Справочная математическая библиотека

ПРОГРАММИРОВАНИЕ



СМБ
ПРОГРАММИРОВАНИЕ



СПРАВОЧНАЯ МАТЕМАТИЧЕСКАЯ БИБЛИОТЕКА

ПОД ОБЩЕЙ РЕДАКЦИЕЙ
Л. А. ЛЮСТЕРНИКА
И
А. Р. ЯНПОЛЬСКОГО

ИЗДАТЕЛЬСТВО «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
МОСКВА 1966

Н. А. КРИНИЦКИЙ, Г. А. МИРОНОВ, Г. Д. ФРОЛОВ

ПРОГРАММИРОВАНИЕ

ИЗДАНИЕ ВТОРОЕ,
ПЕРЕРАБОТАННОЕ И ДОПОЛНЕННОЕ

ИЗДАТЕЛЬСТВО «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
МОСКВА 1966

518
К 82
УДК 519.95

АННОТАЦИЯ

Этот выпуск является вторым, переработанным и дополненным изданием одноименной книги, вышедшей в этой же серии в 1963 г. Он представляет собой справочное пособие по программированию, содержит краткое изложение основ и элементов теории программно-управляемых машин, их классификацию и описание ряда отечественных электронно-цифровых машин, достаточное для нужд программирования.

Настоящее издание отличается от первого следующим: внесены изменения во все главы книги, связанные с расширением сферы применения электронных цифровых машин, включены разделы, посвященные общим сведениям об алгоритмических языках и описанию международного алгоритмического языка АЛГОЛ, включены описания более современных машин *Урал-4, Минск-2, М-20, Раздан-2,* а также универсальной управляющей машины *Днепр.*

Справочник рассчитан на студентов и аспирантов втузов и университетов, а также на инженеров и научных работников, применяющих электронные цифровые машины для решения различных задач и занимающихся их эксплуатацией.

2-2-4
199-66

ОГЛАВЛЕНИЕ

Предисловие ко второму изданию 11
Из предисловия к первому изданию 12

ЧАСТЬ ПЕРВАЯ

ПРОГРАММИРОВАНИЕ ДЛЯ ПРОГРАММНО-УПРАВЛЯЕМЫХ МАШИН

Глава 1. **Основы теории программно-управляемых машин** 15

§ 1. Некоторые общие сведения о программно-управляемых цифровых машинах 15
1. Буквы и алфавиты (15). 2. Слова (16). 3. Общая структурная схема цифровой программно-управляемой машины (18). 4. Принцип действия программно-управляемой машины (20).

§ 2. Системы счисления 23
1. Позиционные системы счисления (23). 2. Перевод чисел из одной системы счисления в другую (32). 3. Формы представления чисел в машинах (36).

§ 3. Элементы математической логики 39
1. Двоичные функции и алгебрологические операции (39). 2. Понятие логических связей, значения истинности и высказывания (42). 3. Основные свойства алгебрологических операций и логических связей (43). 4. Полные системы алгебрологических операций и логических связей (47). Понятие предиката (49).

§ 4. Элементы теории алгорифмов 51
1. Понятие алгорифма (51). 2. Композиция и расчленение слов (55). 3. Равнозначные алфавиты (57). 4. Предметное тождество слов (57). 5. Операции над словами (58). 6. Транспонирование операций (59). 7. Уточнение понятия алгорифма. Алгорифм выполнения. Абстрактная машина (60). 8. Локально заданные операции над словами (63). 9. Размеченная память (65).

10. Абстрактные машины с размеченной памятью (66). 11. Реальные программно-управляемые машины как физические модели абстрактных машин с размеченной памятью (70).

Глава 2. **Система операций и система команд программно-управляемой машины** 73

§ 1. Система операций программно-управляемой машины и их классификация 73

1. Классификация запоминающих устройств машины (73). 2. Классификация операций машины (74).

§ 2. Кодирование чисел для их изображения в устройствах машины 75

1. Прямой код (76). 2. Сумматоры (77). 3. Обратный код (79). 4. Дополнительный код (81). 5. Кодирование чисел в машинах (83). 6. Кодирование буквенной информации (86).

§ 3. Основные операции машины 89

1. Арифметические операции (89). 2. Неарифметические операции над машинными словами и переносы (93). 3. Логические операции (96). 4. Операции, изменяющие содержимое регистра команд (98). 5. Операции перехода (99). 6. Операции, изменяющие содержимое индексного регистра (101). 7. Операции над буквенной информацией (101).

§ 4. Система команд программно-управляемой машины . . 102

1. Система команд. Общий вид команды (102). 2. Условная структура общего вида команды (103).

Глава 3. **Особенности решения задач на цифровых программно-управляемых машинах** 104

§ 1. Порядок решения задач на машине 104

1. Основные области применения цифровых программно-управляемых машин (104). 2. Основные этапы работы (106). 3. Краткая характеристика основных этапов (107).

§ 2. Разработка решающего алгорифма. Задание функций 108

1. Учет особенностей электронной цифровой машины (108). 2. Учет стоимости решения и расхода времени (109). 3. Способы задания функций (110). 4. Масштабирование (111).

§ 3. Методы программного контроля. Отладка программы на машине 115

1. Программный контроль (115). 2. Отладка программы (116).

§ 4. Организация программ 117

1. Понятие организации программы (117). 2. Схема организации (118).

Глава 4. **Операторный метод программирования** 119

§ 1. Простейшее программирование 119

1. Связь между распределением памяти и составлением команд (119). 2. Символические адреса (119).

§ 2. Операторный метод программирования 124

1. Схема решающего алгорифма (124). 2. Некоторые термины и обозначения (130). 3. Элементарные операторы программы (131). 4. Стандартные операторы. Логическая схема программы (134).

Глава 5. **Равносильные преобразования логических схем** 141

§ 1. Комплексы и их эквивалентные преобразования 141

1. Некоторые общие термины и условные обозначения (141). 2. Память. Операторы (143). 3. Операторы специального вида (144). 4. Комплексы (145). 5. Эквивалент комплекса (148). 6. Эквивалентность комплексов (150). 7. Основные эквивалентные преобразования (153). 8. Сложение комплексов (159). 9. Умножение комплексов (161). 10. Единичные и обратные комплексы (162). 11. Пакеты комплексов. Эквивалентность пакетов (163).

§ 2. Формальное определение логической схемы. Равносильность логических схем 163

1. Основные типы операторов (163). 2. Понятие логической схемы (167). 3. Реализация схемы. Равносильность схем. Выражения. Равносильность выражений (169).

§ 3. Основные равносильные преобразования логических схем 171

1. Равносильные преобразования, не связанные с внутренними свойствами операторов (171). 2. Основные преобразования логических операторов (177). 3. Равносильные преобразования нелогических операторов (179). 4. Перестановки операторов (181). 5. Подчиненность оператора предикату (182).

Глава 6. **Некоторые приемы программирования** 185

§ 1. Операторы, зависящие от параметров, их переадресация и восстановление 185

1. Операторы, зависящие от параметров, и операторы переадресации (185). 2. Операторы восстановления (188).

§ 2. Циклы 189

1. Элементарный итерационный цикл (190). 2. Элементарный цикл с переадресацией (190). 3. Элементарный цикл с переадресацией и восстановлением (191). 4. Сложные циклы (193). 5. Счетчик выполнений цикла (194).

§ 3. Приемы, связанные с применением стандартных ячеек 197

1. Вынесение величин в стандартные ячейки (197). 2. Циркуляция величин в стандартных ячейках (198).

§ 4. Логические шкалы. Операторы формирования 200

1. Логические шкалы (200). 2. Операторы формирования (202).

§ 5. Выделение подпрограммы 206

1. Понятие подсхемы и подпрограммы. Выделение подпрограммы (206). 2. Построение обращений к выделенной подпрограмме (207).

Глава 7. **Автоматизация программирования** 211

§ 1. Автоматизация отдельных работ при программировании вручную 211

1. Автоматизация присвоения истинных адресов (211). 2. Автоматизация исправления некоторых ошибок (212). 3. Автоматизация отладки программ (213).

§ 2. Автоматизация составления программ 215

1. Сущность автоматизации программирования (215). 2. Способы автоматического программирования (215). 3. Компиляция и интерпретация (217).

§ 3. Метод библиотеки стандартных подпрограмм 218

1. Общие сведения о методе БСП (218). 2. Интерпретирующая система ИС-2 (220). 3. Организация интерпретирующей системы ИС-2 (220).

§ 4. Метод программирующих программ 222

1. Появление программирующих программ (222). 2. Стандартные и нестандартные операторы при методе ПП (222). 3. Блочное строение программирующих программ (223). 4. Программирование методом ПП (225).

§ 5. Перспективы развития метода программирующих программ 226

1. Универсальные ПП (226). 2. Метод библиотеки специализированных ПП (227).

Глава 8. **Алгорифмические языки** 230

§ 1. Язык логических схем 230

1. Основные символы алгорифмического языка ЯЛС (230). 2. Ячейки (231). 3. Знаки операций. Функции (232). 4. Классификация ячеек (233). 5. Параметры (233). 6. Объекты (234). 7. Ячейки и функции, зависящие от параметров (236). 8. Комплексы (237). 9. Операторы. Их классификация (237). 10. Знаки перехода (239). 11. Логическая схема (240).

§ 2. Алгорифмический язык АЛГОЛ 244

1. Основные символы алгорифмического языка АЛГОЛ (245). 2. Числа (246). 3. Идентификаторы (248). 4. Строки (249). 5. Переменные. Массивы (249). 6. Указатели функций (250). 7. Логические выражения (251). 8. Арифметическое выражение (252). 9. Именующее выражение (253). 10. Операторы (254). 11. Описания (258). 12. Операторы присваивания (260). 13. Операторы перехода (261). 14. Оператор процедуры (262). 15. Вычисление значений функции (264). 16. Оператор цикла (265). 17. Примечания в программах (267). 18. Некоторые замечания (267).

ЧАСТЬ ВТОРАЯ

ОПИСАНИЯ НЕКОТОРЫХ ПРОГРАММНО-УПРАВЛЯЕМЫХ МАШИН

Вводные замечания 270

Глава 9. **Программно-управляемая машина *Днепр (Дніпро)*** 272

Основные характеристики машины *Днепр* (272). Краткие сведения об основных устройствах машины (273). Представление чисел и команд (277). Система операций и команд (278). Пульт управления вычислительной части машины (298). Пульт оператора (пульт управления УСО) (300). Подготовка данных. Ввод и вывод (302).

Глава 10. **Программно-управляемая машина *Урал-2*** . . . 305

Общая характеристика машины *Урал-2* (305). Краткие сведения об основных устройствах машины (306). Представление чисел и команд (309). Система операций и команд (311). Пульт управления (328). Подготовка данных. Ввод и вывод (329).

Глава 11. **Программно-управляемая машина *Урал-4*** 332

Отличия основных характеристик машины *Урал-4* от основных характеристик машины *Урал-2* (332). Краткие сведения об основных устройствах машины (333). Представление чисел и команд (336). Система операций и команд (338). Пульт управления (353). Подготовка данных. Ввод и вывод (353).

Глава 12. **Программно-управляемая машина *Сетунь*** 358

Основные характеристики машины *Сетунь* (358). Краткие сведения об основных устройствах машины (359). Представление чисел и команд (361). Система операций и команд (362). Пульт управления (370). Подготовка данных. Ввод и вывод (371). Дополнение (372).

Глава 13. **Программно-управляемая машина *Минск-2***. . . 375

Основные характеристики машины *Минск-2* (375). Краткие сведения об основных устройствах машины (376). Представление чисел и команд (379). Система операций и команд (381). Пульт управления (417). Подготовка данных. Ввод и вывод (418).

Глава 14. **Программно-управляемая машина *Раздан-2***. . . 422

Основные характеристики машины *Раздан-2* (422). Краткие сведения об основных устройствах машины (423). Представление чисел и команд (426). Система операций и команд (428). Пульт управления (489). Подготовка данных. Ввод и вывод (491).

Глава 15. **Программно-управляемая машина *Стрела***. . . 493

Основные характеристики машины *Стрела* (493). Краткие сведения об основных устройствах машины (494). Представление чисел и команд (497). Система операций и команд (497). Пульт управления (512). Подготовка данных. Ввод и вывод (513). Машина *Стрела* ВЦ АН СССР (514).

Глава 16. **Программно-управляемая машина *М-20*** 526

Основные характеристики машины *М-20* (526). Краткие сведения об основных устройствах машины *М-20* (527). Представление чисел и команд (532). Система операций и команд (533). Пульт управления (563). Подготовка данных. Ввод и вывод (566).

Глава 17. **Программно-управляемая машина *БЭСМ-2***. . . . 567

Основные характеристики машины *БЭСМ-2* (567). Краткие сведения об основных устройствах машины (568). Представление чисел и команд (571). Система операций и команд (572). Пульт управления (572). Подготовка данных. Ввод и вывод (592).

Библиография 596

ПРЕДИСЛОВИЕ КО ВТОРОМУ ИЗДАНИЮ

С момента выхода в свет первого издания этой книги в области электронной вычислительной техники и программирования произошел ряд серьезных изменений. В производстве цифровых программно-управляемых машин совершился переход от ламповых элементов к полупроводниковым, что послужило причиной начала периода обновления отечественного парка машин. Машины стали оснащаться, помимо устройств для ввода и вывода чисел, устройствами ввода и вывода буквенной информации. В ряде машин предусматривается возможность ввода и вывода информации на телеграфные аппараты, что делает возможным дистанционное решение задач. Намечается тенденция увеличения объема запоминающих устройств, технически уже возможного, ведущая к тому, что «начинка» машины программами, составленными вручную, становится практически невозможной, тогда как автоматизация программирования значительно облегчается.

В области программирования ясно наметилась тенденция развития в сторону разработки и применения для нужд автоматизации программирования алгоритмических языков, позволяющих описывать процессы решения крупных классов задач.

В связи с изложенным второе издание настоящей книги подвергнуто значительной переработке. Ряд изменений внесен в главы 1, 2, 3; из главы 7 исключены описания компилирующих систем стандартных подпрограмм, не получивших широкого распространения; в книгу включена новая глава, посвященная алгорифмическим языкам. Из второй части книги исключены описания цифровых программно-управляемых машин *Урал-1, М-2* и *М-3,* которые морально устарели. Взамен этого приведены описания ряда новых программно-управляемых машин: машины *Урал-4,* обладающей увеличенным объемом запоминающих устройств; получивших широкое

распространение машин *М-20* и *Минск-2*; машины *Раздан-2*, а также универсальной управляющей машины *Днепр*.

Первая часть книги (гл. 1 — 8) написана Н. А. Криницким, вторая часть — Г. А. Мироновым (гл. 10, 11, 14, 16, 17) и Г. Д. Фроловым (гл. 9, 12, 13, 15).

Авторы

ИЗ ПРЕДИСЛОВИЯ К ПЕРВОМУ ИЗДАНИЮ

Электронные цифровые программно-управляемые машины появились в 40-х годах XX века как средство сверхбыстрых вычислений. Потребность в этих машинах назрела в связи с необходимостью решения ряда сложных и трудоемких задач, выдвинутых техникой и естествознанием. Изучение программно-управляемых машин показало, что возможности их далеко выходят за рамки математических вычислений (так, с их помощью можно осуществлять автоматический перевод текстов с одного языка на другой, управление объектами, например станками, получение следствий из посылок, выбор сведений из хранилища информации, например подбор литературы, в которой освещается тот или иной вопрос, и др.). Можно сказать, что цифровые программно-управляемые машины являются средством механизации умственного труда и автоматизации производственных процессов.

Одновременно с появлением цифровых программно-управляемых машин возникла новая область прикладной математики — программирование (целью которого является составление «программ действий» машины для решения различных задач). Сначала программирование представлялось скорее искусством, чем наукой. Составление программ производилось вручную и было подобно решению сложных комбинаторных задач (таких, например, как шахматные). Постепенно был разработан ряд методов программирования. Появление операторного метода программирования, предложенного в 1953 г. А. А. Ляпуновым, позволило по-настоящему поставить вопрос об автоматизации программирования. Сущность операторного метода заключается в том, что алгорифм решения задачи представляется в виде так называемой логической схемы программы, элементами которой являются операторы. Логические схе-

мы позволяют расчленить громоздкий процесс составления программы и допускают «механическое» (выполнимое по формальным правилам) составление частей программы, отвечающих операторам, и объединение их в одно целое. На идеях операторного метода основываются программирующие программы. В настоящее время работы в области автоматизации программирования ведутся в направлении создания алгорифмических языков. Описание на алгорифмическом языке методов решения задач допускает их формальную переработку в программы. В основе этих языков лежит операторный метод. От универсальных программирующих программ намечается переход к библиотекам специализированных программирующих программ, позволяющим более полно автоматизировать процесс программирования.

Постановка все более сложных задач приводит к появлению весьма сложных программ, требующих огромного времени для выполнения даже на самых быстродействующих машинах; в связи с этим возникла потребность исследования возможности равносильных преобразований программ с целью их «улучшения». В настоящее время ведутся работы по равносильным преобразованиям логических схем, с помощью которых преобразование программ может быть произведено еще до программирования.

Исследования процессов программирования и выполнения программ машинами, а также исследование принципов действия машин приводят к выводу, что программирование является прикладной ветвью теории алгорифмов. Программы представляют собой определенное семейство алгорифмов, а электронные цифровые программно-управляемые машины являются автоматами для выполнения этих алгорифмов.

В настоящей книге авторы делают попытку изложить сущность программирования, исходя из современных идей, и по возможности осветить современный уровень этой области. В отличие от других книг, программирование здесь излагается без привязки к какой-либо из существующих или выдуманных программно-управляемых машин. Первая часть книги посвящена программированию, вторая часть содержит описание ряда отечественных электронных программно-управляемых машин. Авторы надеются, что подход, принятый в книге, позволит уяснить сущность программирования для любой программно-управляемой машины.

Приведенные в книге сведения о машинах достаточны для программирования для них, однако чтобы получить «хорошие» программы, кроме этих сведений, необходима еще некоторая тренировка.

Авторы пользуются случаем выразить глубокую признательность Л. А. Люстернику, А. А. Ляпунову, М. Р. Шура-Бура и А. Р. Янпольскому за ценные советы, использованные при написании книги.

Авторы

ЧАСТЬ ПЕРВАЯ
ПРОГРАММИРОВАНИЕ ДЛЯ ПРОГРАММНО-УПРАВЛЯЕМЫХ МАШИН

ГЛАВА 1
ОСНОВЫ ТЕОРИИ ПРОГРАММНО-УПРАВЛЯЕМЫХ МАШИН

§ 1. Некоторые общие сведения о программно-управляемых цифровых машинах

1. Буквы и алфавиты. Символ, рассматриваемый в данном его применении как неизменный и целый, называется *буквой*.

Считается, что в отношении любых двух букв всегда известно, *одинаковы* они или *различны*. Одинаковые буквы можно считать экземплярами одной и той же буквы.

Пример 1.1. Русские буквы, с нашей точки зрения, являются буквами. Индивидуальные буквы, стоящие на первом и третьем местах в слове

анализ,

одинаковы. Можно считать, что эти буквы являются одной и той же буквой.

Конечная последовательность попарно-различных букв называется *алфавитом*. Пусть A — некоторый алфавит. Всякая буква, одинаковая с одной из букв алфавита A, называется *буквой в алфавите* A.

Если каждая буква, входящая в состав алфавита A, является буквой в алфавите B, то говорят, что B является *расширением* A, а A является *частью* B.

Два алфавита называются *равными*, если каждый из них является частью другого.

Иногда мы будем записывать алфавит в виде последовательности букв, разделенных запятыми, а всю последовательность будем заключать в фигурные скобки.

Пример 1.2. Алфавиты

$$\{а, б, в, г\}$$

и

$$\{б, а, г, в\}$$

между собой равны.

Количество букв алфавита называется его *объемом*. Если объем алфавита равен n, то, очевидно, число алфавитов, различных между собой и отличающихся от данного, но равных ему, равно $n! - 1$.

Пусть A и B — два алфавита. Любой алфавит, состоящий из всех таких букв, каждая из которых является как буквой в A, так и буквой в B, называется пересечением алфавитов A и B. Пересечение двух равных алфавитов равно любому из них.

Любой алфавит, являющийся расширением как алфавита A, так и алфавита B и обладающий тем свойством, что каждая его буква является буквой в A или в B, называется *объединением* (суммой) алфавитов A и B. Объединение двух равных алфавитов равно любому из них.

Пример 1.3. Пусть даны алфавиты

$$\{1, б, 2, в, г\}$$

и

$$\{1, б, в, 5\}.$$

Пересечением этих алфавитов будет, например,

$$\{1, б, в\},$$

а их объединением, например,

$$\{1, б, 2, в, 5, г\}.$$

Заметим, что цифры с нашей точки зрения являются буквами, а, например, их последовательность

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

— алфавитом.

2. Слова. Пусть задан некоторый алфавит A. *Словом* в A называется конечная последовательность букв в A. Иногда

удобно допускать существование *пустых* слов (не содержащих ни одной буквы).

Если перенумеровать буквы, образующие слово, в том порядке, в котором они расположены в этом слове, то наибольший номер буквы называется *длиной слова*. Если P означает некоторое слово, то символом $L(P)$ мы будем обозначать его длину. Длину пустого слова считают равной нулю.

Два слова называются *равными*, если они состоят из одинаково расположенных одинаковых букв.

Очевидно, всякое слово в алфавите A является словом в любом расширении алфавита A.

К *написанию* слов в каждом алфавите предъявляется ряд требований. Во-первых, для любых двух букв слова должно быть известно, которая из них является предыдущей и которая последующей. Во-вторых, должны быть известны начало и конец слова. Наконец, в ряде случаев приходится, имея символ, о котором известно, что он является словом в алфавите A, определять буквы, последовательностью которых этот символ является («чтение» слова). Разложение слова на буквы должно быть единственным («разночтение» слов должно быть исключено).

Пример 1.4. Рассмотрим алфавит

$$\{0, 1, 00, 01, 10, 11\}.$$

Буквы этого алфавита, вообще говоря, не исключают возможности разночтения слов.

Например, слово

01001

можно рассматривать как:

1) 0, 1, 0, 0, 1,
2) 01, 00, 1,
3) 0, 10, 01,
4) 01, 0, 01,
5) 0, 1, 0, 01,
6) 0, 1, 00, 1,
7) 0, 10, 0, 1,
8) 01, 0, 0, 1.

Однако если принято правило образования слов, гласящее, что всякое слово, длина которого не более чем 2, получено только с помощью букв 0 и 1, а всякое слово, длина которого более чем 2, начинается с одной из букв 00, 01, 10, 11 и в остальной своей части состоит только из букв 0 и 1, то возможность разночтения оказывается полностью устраненной. Слово 01001 читается теперь единственным образом, как 01, 0, 0, 1.

Заметим, что, приняв вышеприведенное правило образования слов, мы условились пользоваться не всеми возможными словами в нашем алфавите, а только некоторыми определенными словами.

3. Общая структурная схема цифровой программно-управляемой машины. Программно-управляемая цифровая машина состоит из трех основных частей (рис. 1.1):

арифметического устройства, предназначенного для выполнения операций над словами (в частности, над числами);

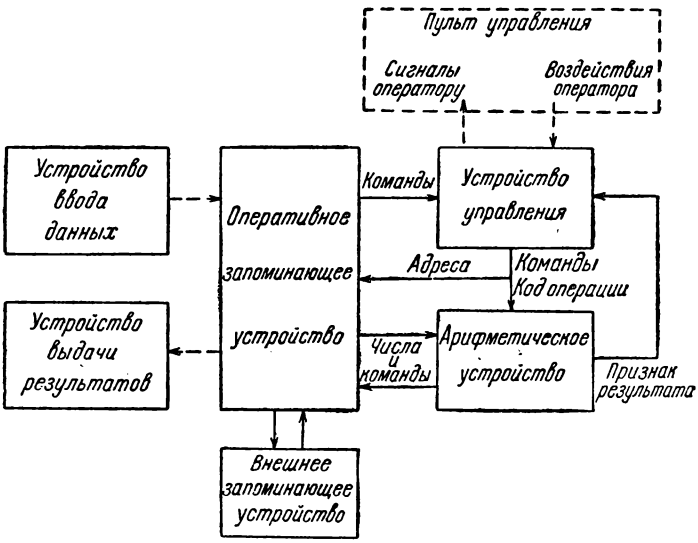


Рис. 1.1. Упрощенная блок-схема цифровой программно-управляемой машины.

запоминающего устройства, предназначенного для хранения слов (в частности, чисел);

устройства управления, предназначенного для управления автоматической работой машины.

Заметим, что названия *цифровые* или *вычислительные*, а также название *арифметическое* устройство применительно к любым программно-управляемым машинам, являются традиционными. Эти названия возникли в то время, когда программно-управляемые машины применялись только для выполнения вычислений и сохранились до сих пор, хотя вы-

числения представляют собой лишь незначительную часть области применения этих машин.

Обычно в машине присутствуют запоминающие устройства нескольких типов: быстродействующее и сравнительно небольшой емкости *оперативное запоминающее устройство* (коротко ОЗУ), называемое также *памятью*, и медленное запоминающее устройство большой емкости — *внешнее запоминающее устройство* (коротко ВЗУ).

При решении задач на цифровых программно-управляемых машинах в памяти хранятся главным образом слова, непосредственно необходимые для выполнения операций. Остальной материал размещается во внешнем запоминающем устройстве. Между памятью и ВЗУ по мере необходимости производится обмен информацией.

Запись слов в память и выдача их из памяти производятся отдельными словами, запись в ВЗУ и выдача из ВЗУ — группами слов. Удобно считать (для большинства существующих машин это действительно так), что память электронной цифровой программно-управляемой машины состоит из частей, предназначенных для хранения отдельных слов и называемых *ячейками*. В свою очередь каждая ячейка представляет собой последовательность элементарных частей, называемых *разрядами* и предназначенных для записи в них отдельных букв.

Запись какого-либо слова в ячейку начинается всегда с того, что «стирается» слово, которое прежде в ней хранилось. После записи слова в ячейку ее содержимое остается неизменным до новой записи.

Внешнее запоминающее устройство считается при этом разбитым на *зоны*, в которых осуществляется хранение групп слов.

Кроме перечисленных трех основных частей, программно-управляемые машины включают в себя *устройства ввода* данных, *устройства вывода* результатов и *пульт ручного управления*, обеспечивающий возможность вмешательства человека в работу машины (см. рис. 1.1).

В большинстве существующих вычислительных программно-управляемых машин ввод данных не может быть произведен в виде, непосредственно заданном человеком. Для ввода данных в память машины их сперва кодируют системой отверстий на *перфокартах* (листы тонкого картона) или *перфолентах* (целлулоидные или бумажные ленты).

Выдача результатов в некоторых машинах производится путем их непосредственной печати на широкой бумажной ленте. Однако не менее распространен способ выдачи путем пробивки результатов на перфолентах или перфокартах.

Для кодировки исходных данных на перфокартах (перфолентах) применяются специальные *клавишные устройства* и так называемые *входные перфораторы*, а для печатания результатов, выданных на перфокарты (перфоленты), — специальные *печатающие устройства*. Все эти устройства уже не входят в состав программно-управляемой машины, а являются ее дополнительным оборудованием. Обычно их называют *внешними устройствами*.

Ближайшей, уже ясно видимой перспективой является ввод в машину текстов, отпечатанных типографским способом или на пишущих машинках, далекой перспективой — ввод текстов, написанных от руки или даже произнесенных голосом. Возможен ввод текстов из каналов связи, например, передаваемых телеграфным аппаратом или устройством вывода другой программно-управляемой машины. Вывод информации (кроме описанных выше способов) может производиться в виде сигналов в каналы связи и в графическом виде, например на экран специальной электронно-лучевой трубки.

4. Принцип действия программно-управляемой машины. Программно-управляемая машина работает под воздействием так называемых команд.

Команда — это информация, представленная в форме, позволяющей ввести ее в машину и определяющая действия машины в течение некоторого отрезка времени.

Совокупность команд, определенным образом расположенных в ячейках памяти машины, образует *программу* работы машины. Программа заранее составляется для каждой задачи и вводится *) в машину вместе с исходными данными перед решением задачи; после этого все решение выполняется машиной автоматически.

В большинстве существующих электронных цифровых программно-управляемых машинах используется *адресный*

*) В специализированных машинах, предназначенных для решения только нескольких определенных задач, программа может быть раз и навсегда установлена в машине.

принцип указания *операндов* (слов, являющихся исходными данными или результатами операций). Сущность адресного принципа заключается в том, что для обозначения слов, хранящихся в памяти, применяется стандартный набор слов, называемых *адресами*. Обычно адресами операндов являются номера (или названия) отведенных для них ячеек (отсюда и происходит название «адрес»). При этом команда как группа символов (букв) делится на несколько подгрупп. Одна подгруппа является *кодом операции* и определяет, что должна сделать машина — характер или вид операции; одна или несколько подгрупп, называемых *адресами*, указывают адреса операндов (например, номера ячеек запоминающего устройства, где находятся числа, над которыми производится операция или куда нужно записать результат).

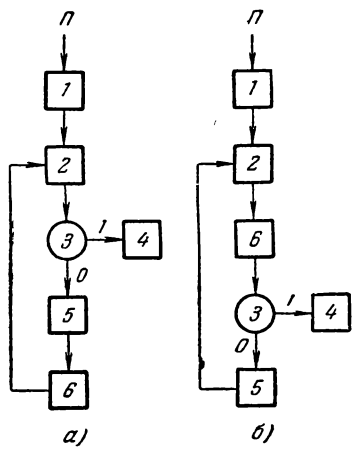


Рис. 1.2. Возможная схема рабочего цикла цифровой программно-управляемой машины.

Следует подчеркнуть, что термин «адрес» может применяться в двух смыслах: как обозначение операнда (в большинстве машин — номер ячейки, в которой он хранится) или как часть команды.

В последнее время появились машины, в которых адресный принцип не используется. В таких машинах (называемых *безадресными*) в состав команд включаются не адреса операндов, а сами операнды. Получат ли такие машины широкое распространение, покажет будущее. Заметим, что и в машинах, использующих адресный принцип, отдельные команды могут требовать непосредственного включения в их состав операндов.

Функционирование электронной программно-управляемой цифровой машины (использующей адресный принцип) после того, как в нее введены программа и исходные данные, представляет собой многократное выполнение так называемого *рабочего цикла*. Возможная схема рабочего цикла представлена в двух вариантах на рис. 1.2, *а* и *б*, где квадратами и кружками изображены отдельные группы действий. При

этом *П* означает пуск машины человеком; *1* — запись в специальную ячейку, называемую *счетчиком-регистром адреса команды* (коротко — СК), адреса первой команды программы; *2* — перенос из оперативной памяти в специальную ячейку, называемую *регистром команд* (коротко — РК), слова, адрес которого находится в СК; *3* — проверка, является ли слово, записанное в РК командой останова, и переход в этом случае по стрелке, отмеченной цифрой 1, а в противном случае по стрелке, отмеченной цифрой 0; *4* — останов (прекращение функционирования машины); *5* — выполнение команды, записанной в РК; *6* — «вычисление» нового адреса команды и запись его в СК.

При всем разнообразии существующих программно-управляемых машин, схемы, приведенные на рис. 1.2, в основном передают сущность рабочего цикла любой из них. Блок *5* при детализации описания рабочего цикла в соответствии с особенностями конкретной машины оказывается сильно разветвленным, причем некоторые ветви не выводят процесс функционирования за пределы рабочего цикла, а другие ведут к новым остановам машины.

При работе машины команды программы выдаются из памяти в устройство управления для выполнения их поочередно в заранее заданном порядке или в порядке, указанном в самих командах. Слова, выбранные из памяти в соответствии с адресами команды (команд), поступают в арифметическое устройство, в котором над ними производится операция, определяемая кодом операции; результат операции может быть послан обратно в память, в ячейку, номер которой указан в соответствующем адресе данной или другой команды.

Из арифметического устройства в устройство управления поступают сигналы о характере результатов, получающихся при выполнении операций. При получении определенных сигналов устройство управления изменяет порядок выполнения команд программы или характер работы машины.

Помимо команд об операциях над словами, машиной выполняются и команды другого рода, необходимые для автоматической работы машины (команды передачи данных из одного места памяти в другое, выдачи данных из машины, остановки и др.).

Программно-управляемые машины в принципе могут быть самой различной физической природы (например, они могут

быть механическими, контактно-релейными, электронными и др.). В настоящее время наибольшее практическое значение имеют *электронные* программно-управляемые машины. Элементы, применяемые для представления слов в таких машинах, являются двухпозиционными (обладают двумя устойчивыми состояниями). Например, полупроводниковый триод может проводить ток или не проводить, магнитный элемент может быть намагничен в «положительную» сторону или в «отрицательную» и т. п. Это обстоятельство явилось одной из причин того, что во многих электронных цифровых машинах используются двухбуквенные алфавиты. В частности, числа представляются в *двоичной* системе счисления или в *двоичнокодированных* системах счисления.

Алфавит, принятый в программно-управляемой машине, называют *машинным алфавитом*, его буквы — *машинными буквами*, слова в этом алфавите, хранение которых возможно в ячейках памяти, — *машинными словами*. Система счисления, используемая для представления чисел в машине и выполнения над ними операций, называется *рабочей системой счисления* машины.

§ 2. Системы счисления

Счислением называется совокупность приемов наименования и обозначения (записи) чисел.

Условные знаки, применяемые при обозначении чисел, обычно называют *цифрами*.

В ряде систем счисления числа записывают как последовательности цифр. Такие системы счисления подразделяются на *позиционные* и *непозиционные*, в зависимости от того, изменяются или нет значения цифр при изменении их положения (позиции) в последовательности. В качестве непозиционной укажем так называемую *римскую* систему счисления.

1. Позиционные системы счисления. Пусть p — некоторое целое число, большее единицы, которое будем называть *основанием* системы счисления. Выберем p попарно различных знаков: p-ичных *цифр*. Выберем, кроме того, p последовательных целых чисел, среди которых содержится нуль. Эту последовательность чисел будем называть *базой* системы счисления. *Между p-ичными цифрами и числами базы установим взаимно однозначное соответствие.*

В дальнейшем рассматриваются только позиционные системы счисления, базы которых либо *неотрицательны* (состоят из нуля и положительных чисел), либо *симметричны* относительно нуля. Цифры системы счисления, отвечающие числам базы, которые симметричны относительно нуля, называются *симметричными.*

Пример 1.5. Примером позиционной системы счисления является десятичная система, широко применяемая в повседневной практике. Для нее основание p равно десяти, база ее неотрицательна и состоит из десяти последовательных целых чисел, начиная с нуля и кончая девятью. В качестве десятичных цифр применяются арабские цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Число записывают в виде последовательности p-ичных цифр, которая разделена запятой на две подпоследовательности. Если каждый из символов a_n, a_{n-1}, ..., a_1, a_0, a_{-1}, ..., a_{-m} означает p-ичную цифру, то запись числа может иметь вид:

$$a_n a_{n-1} \ldots a_1 a_0, \; a_{-1} \ldots a_{-m}. \tag{1.1}$$

Считают, что позиции, на которых в последовательности (1.1) стоят цифры, перенумерованы: позиции, расположенные левее запятой, перенумерованы подряд справа налево числами нуль, один, два и т. д.; позиции, расположенные справа от запятой перенумерованы подряд слева направо с помощью чисел минус один, минус два и т. д. Эти перенумерованные позиции называются *р-ичными разрядами* числа.

Каждой цифре последовательности (1.1) приписано определенное *значение.* Цифра, стоящая в нулевом разряде, имеет своим значением соответствующее ей число базы. Цифра, стоящая в некотором разряде, имеет значение, в p раз большее того, которое она имела бы в разряде с номером, меньшим на единицу (или значение, в p раз меньшее того, которое она имела бы в разряде с номером, большим на единицу).

Последовательность р-ичных цифр обозначает число, равное сумме значений ее цифр.

В соответствии со сказанным, последовательность p-ичных цифр (1.1) означает

$$a_n a_{n-1} \ldots a_1 a_0, \; a_{-1} \ldots a_{-m} = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p + a_0 + a_{-1} p^{-1} + \ldots + a_{-m} p^{-m}. \tag{1.2}$$

В дальнейшем, если это не вызывает недоразумений, вместо слов «запись числа» будем говорить просто «число».

Последовательность цифр числа, расположенных слева от запятой, представляет *целую часть* числа. Если база системы счисления содержит как положительные, так и отрицательные числа, то в этой системе счисления существуют правильные дроби, имеющие целые части (см., например, стр. 28). Последовательность, которая получится, если целую часть числа заменить нулем, называется его *дробной частью*.

Пример 1.6. Приведем примеры десятичной записи чисел: а) основание системы (десять) имеет запись 10; б) $8401{,}302 = 8 \cdot 10^3 + 4 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0 + 3 \cdot 10^{-1} + 0 \cdot 10^{-2} + 2 \cdot 10^{-3}$. Для записи отрицательных чисел в десятичной системе кроме цифр применяется еще знак «—» (*минус*), который ставят перед числом (т. е. слева от последовательности цифр); иногда перед положительным числом ставится знак «+» (*плюс*).

Сложение, вычитание, умножение и деление чисел, представленных в p-ичной позиционной системе, выполняются весьма просто с использованием таблиц сложения, вычитания и умножения, подобно тому, как это делают в общеизвестной десятичной системе счисления. Умножение числа на основание системы p, как это следует из формулы (1.2), сводится к переносу запятой на один разряд вправо, а деление на p — к переносу запятой на один разряд влево.

В программно-управляемых машинах и при подготовке задач для решения на них кроме десятичной могут применяться и другие позиционные системы счисления. Наиболее часто используемые позиционные системы будут разобраны ниже.

Двоичная позиционная система счисления с неотрицательной базой. В этой системе счисления применяются *две* различные цифры 0 и 1. Число «два» (основание системы) записывают как 10. При записи отрицательных чисел перед последовательностью цифр ставят знак минус.

Двоичные таблицы сложения (табл. 1.1), вычитания (табл. 1.2) и умножения (табл. 1.3) весьма просты.

Таблица 1.1.
Двоичная таблица сложения

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=10$

Таблица 1.2.
Двоичная таблица вычитания

$0-0=0$
$1-0=1$
$1-1=0$
$10-1=1$

Таблица 1.3.
Двоичная таблица умножения

$0\times 0=0$
$0\times 1=0$
$1\times 0=0$
$1\times 1=1$

С помощью этих таблиц сложение, вычитание, умножение и деление двоичных чисел выполняются по тем же правилам, по которым мы привыкли складывать, вычитать, умножать и делить десятичные числа.

Пример 1.7.

а) сложение

```
  1 100 111,011
+    10 011,111
-----------------
  1 111 011,010
```

б) вычитание

```
  10 110,1101
-  10 001,1111
-----------------
     100,1110
```

в) умножение

```
      1100111,1101
    ×       11,011
    -----------------
      1100 1111101
     11001 111101
    1100111 1101
   11001111 101
   -----------------
   101011110,0101111
```

г) деление

```
11011101101 | 1001
1001          -------
-----          11000101
 1001
 1001
 -----
    1011
    1001
    -----
     1001
     1001
     ----
     0000
```

Восьмеричная позиционная система счисления с неотрицательной базой. В такой системе счисления используются *восемь* различных цифр 0, 1, 2, 3, 4, 5, 6, 7. Основание системы (восемь) записывают как 10. При записи отрицательных чисел перед последовательностью цифр ставят знак минус.

Восьмеричные таблицы сложения (табл. 1.4) и умножения (табл. 1.5) представлены здесь в форме так называемых таблиц с двумя входами.

Таблица 1.4. Восьмеричная таблица сложения

+	0	1	2	3	4	5	6	7	10
0	0	1	2	3	4	5	6	7	10
1	1	2	3	4	5	6	7	10	11
2	2	3	4	5	6	7	10	11	12
3	3	4	5	6	7	10	11	12	13
4	4	5	6	7	10	11	12	13	14
5	5	6	7	10	11	12	13	14	15
6	6	7	10	11	12	13	14	15	16
7	7	10	11	12	13	14	15	16	17
10	10	11	12	13	14	15	16	17	20

Пример 1.8.

а) сложение

$$\begin{array}{r} 327{,}71102 \\ +\ 35{,}67735 \\ \hline 365{,}61037 \end{array}$$

б) вычитание

$$\begin{array}{r} 11076{,}01 \\ -\ 705{,}62 \\ \hline 10170{,}17 \end{array}$$

Таблица 1.5. Восьмеричная таблица умножения

×	0	1	2	3	4	5	6	7	10
0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	10
2	0	2	4	6	10	12	14	16	20
3	0	3	6	11	14	17	22	25	30
4	0	4	10	14	20	24	30	34	40
5	0	5	12	17	24	31	36	43	50
6	0	6	14	22	30	36	44	52	60
7	0	7	16	25	34	43	52	61	70
10	0	10	20	30	40	50	60	70	100

Пример 1.9.

а) умножение

$$\begin{array}{r} 173{,}261 \\ 16{,}35 \\ \hline 1150565 \\ 562023 \\ 1344046 \\ 173261 \\ \hline 3366{,}56615 \end{array}$$

б) деление

$$\begin{array}{r|l} 336656{,}615 & 1635 \\ 1635 & \overline{173{,}261} \\ \hline 15315 & \\ 14513 & \\ \hline 6026 & \\ 5327 & \\ \hline 4776 & \\ 3472 & \\ \hline 13041 & \\ 12656 & \\ \hline 1635 & \\ 1635 & \\ \hline 0000 & \end{array}$$

Шестнадцатеричная позиционная система счисления с неотрицательной базой. В этой системе счисления применяются *шестнадцать* различных цифр*) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *a*, *b*, *c*, *d*, *e*, *f*. Основание системы (шестнадцать) записывается как 10. При записи отрицательных чисел слева от последовательности цифр ставят знак минус.

*) Для обозначения цифр, больших 9, применяют и другие знаки, например *A*, *B*, *C*, *D*, *E*, *F* или *а*, *б*, *в*, *г*, *д*, *е*, или $\overline{0}$, $\overline{1}$, $\overline{2}$, $\overline{3}$, $\overline{4}$, $\overline{5}$ и т. п.

Шестнадцатеричные таблицы сложения и умножения значительно больше десятичных. Их нетрудно составить по образцу табл. 1.4 и 1.5.

Троичная позиционная система счисления с симметричной базой. В этой системе счисления применяются *три* различные цифры $\bar{1}$, 0, 1. Основание системы счисления (три) записывается как 10. Для различения положительных и отрицательных чисел знак (минус) не нужен. Старшей цифрой отрицательного числа всегда является $\bar{1}$, а всякое положительное число имеет в качестве старшей цифры 1. Некоторые числа, являющиеся правильными дробями, в этой системе счисления имеют не равную нулю целую часть. Например, число 2/3 записывается как $1, \bar{1}$.

Сложение и вычитание чисел выполняются с помощью табл. 1.6.

Таблица 1.6. Троичная таблица сложения

+	$\bar{1}$	0	1
$\bar{1}$	$\bar{1}\,1$	$\bar{1}$	0
0	$\bar{1}$	0	1
1	0	1	$1\,\bar{1}$

Пример 1.10.

а) сложение

$$\begin{array}{r} 1100\bar{1},10\bar{1}\bar{1} \\ +\ 1\bar{1}00,01\bar{1}0 \\ \hline 1\bar{1}\bar{1}10\bar{1},101\bar{1} \end{array}$$

б) вычитание

$$\begin{array}{r} 1\bar{1}\bar{1}10\bar{1}1,01\bar{1} \\ -\ 1\bar{1}000,1\bar{1}0 \\ \hline 0110\bar{1}1,0\bar{1}\bar{1} \end{array}$$

Умножение числа на отрицательную единицу сводится к замене всех цифр этого числа симметричными им цифрами. Это позволяет сократить объем таблицы умножения (табл. 1.7), включая в нее только произведения неотрицательных чисел базы. Произведения остальных чисел базы получают из положительных произведений с помощью общеизвестного правила знаков для умножения.

Таблица 1.7. Троичная таблица умножения (в случае симметричной базы)

×	0	1
0	0	0
1	0	1

С помощью таблиц 1.6 и 1.7 умножение чисел производится по тем же правилам, что и в десятичной системе счисления.

Пример 1.11. Умножение

$$\begin{array}{r} 110\bar{1},\bar{1}01 \\ \times \quad 10,\bar{1}1 \\ \hline \bar{1}\bar{1} \quad 01\bar{1}0\bar{1} \\ 110 \quad \bar{1}\bar{1}01 \\ 110\bar{1}\bar{1} \quad 01 \\ \hline 11011\,,\bar{1}1111 \end{array}$$

Алгорифм деления требует применения табл. 1.6 и 1.7 и несколько отличается от применяемого в десятичной системе счисления.

Отличие это состоит в следующем. Если промежуточное делимое (полученное из остатка путем сноса цифры из делимого) содержит столько же разрядов, сколько делитель, то, независимо от того, больше ли это промежуточное делимое, чем делитель, в частное записывают цифру 1, если первые цифры промежуточного делимого и делителя одинаковы, или цифру $\bar{1}$ в противном случае. Если вслед за этим получается остаток, содержащий столько же разрядов, сколько их в делителе, то повторяют указанное действие, записывая новую цифру частного не справа от ранее написанной, а под нею. После конца процесса (который либо заканчивается получением нулевого остатка, либо искусственно прерывается, как и при делении в десятичной системе счисления) число, полученное в первой строке частного, складывают с числом, полученным во второй строке, считая, что под теми цифрами первой строки, под которыми ничего не подписано, стоят нули.

Пример 1.12. Деление

$$
\begin{array}{l|l}
10\bar{1}\bar{1}0\bar{1}\bar{1}\bar{1} & 1\bar{1}\bar{1} \\
\underline{1\bar{1}\bar{1}} & \\
\underline{1\bar{1}\bar{1}} & 100\bar{1}001 \\
\underline{1\bar{1}\bar{1}} & 1\ \ \bar{1} \\
000\bar{1}0\bar{1} & 1\bar{1}0\bar{1}1001 \\
\underline{\ \ \bar{1}11} & \\
\underline{\ \ \bar{1}11} & \\
\underline{\ \ \bar{1}11} & \\
0001\bar{1}\bar{1} & \\
\underline{\ \ \ 1\bar{1}\bar{1}} & \\
000 &
\end{array}
+
$$

Девятеричная позиционная система счисления с симметричной базой. В такой системе счисления применяются девять различных цифр $\bar{4}$, $\bar{3}$, $\bar{2}$, $\bar{1}$, 0, 1; 2, 3, 4. Основание системы (девять) записывается как 10. Для различения положительных и отрицательных чисел знак (минус) не нужен. Число является отрицательным, если его старшая цифра отвечает отрицательному числу базы, и положительным, если его старшая цифра отвечает положительному числу базы.

Сложение и вычитание чисел выполняются с помощью табл. 1.8.

Таблица 1.8. Девятеричная таблица сложения (в случае симметричной базы)

+	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	0	1	2	3	4
$\bar{4}$	$\bar{1}1$	$\bar{1}2$	$\bar{1}3$	$\bar{1}4$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	0
$\bar{3}$	$\bar{1}2$	$\bar{1}3$	$\bar{1}4$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	0	1
$\bar{2}$	$\bar{1}3$	$\bar{1}4$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	0	1	2
$\bar{1}$	$\bar{1}4$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	0	1	2	3
0	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	0	1	2	3	4
1	$\bar{3}$	$\bar{2}$	$\bar{1}$	0	1	2	3	4	$1\bar{4}$
2	$\bar{2}$	$\bar{1}$	0	1	2	3	4	$1\bar{4}$	$1\bar{3}$
3	$\bar{1}$	0	1	2	3	4	$1\bar{4}$	$1\bar{3}$	$1\bar{2}$
4	0	1	2	3	4	$1\bar{4}$	$1\bar{3}$	$1\bar{2}$	$1\bar{1}$

Пример 1.13.

а) сложение

$$\begin{array}{r} 4\bar{4}3\bar{2}0\bar{1}1 \\ +\ \bar{3}20\bar{4}14 \\ \hline 33\bar{4}\bar{2}\bar{4}1\bar{4} \end{array}$$

б) вычитание

$$\begin{array}{r} 33\bar{4}\bar{2}\bar{4}1\bar{4} \\ -\ 4\bar{4}3\bar{2}0\bar{1}1 \\ \hline 0\bar{3}20\bar{4}14 \end{array}$$

Двоично-кодированные системы счисления. Пусть p — основание позиционной системы счисления. Поставим во взаимно однозначное соответствие p-ичным цифрам не равные между собой целые двоичные числа. Определив количество разрядов k наибольшего из них, уравняем по нему разрядности остальных выбранных двоичных чисел, приписывая к каждому слева необходимое для этого количество нулей. Каждой p-ичной цифре теперь соответствует k-разрядное двоичное число, называемое ее *двоичным кодом.* Любое p-ичное число можно закодировать, заменяя его p-ичные цифры их двоичными кодами. Получаемая при этом совокупность правил записи чисел называется *p-ичной двоично-кодированной* системой счисления. Очевидно, наименьшая возможная разрядность двоичных кодов получится, если k выбрать так, чтобы выполнялось неравенство

$$2^{k-1} < p \leqslant 2^k.$$

Легко сообразить, что количество k-разрядных двоичных чисел, не используемых в качестве кодов p-ичных цифр, равно $2^k - p$. Эти числа обычно называют «запрещенными комбинациями» (нулей и единиц).

Пример 1.14. Десятичная двоично-кодированная система счисления, в которой каждая десятичная цифра заменена четырехразрядным равным ей двоичным числом, называется *двоично-десятичной.* В двоично-десятичной системе счисления количество запрещенных комбинаций равно $2^4 - 10 = 6$ (вот они: 1010, 1011, 1100, 1101, 1110, 1111). Четверки двоичных цифр, являющихся кодами десятичных цифр, называют *тетрадами.* Десятичное число 893,2 изобразится в двоично-десятичной системе следующим образом: 1000 1001 0011, 0010.

В некоторых электронных цифровых машинах принято представление чисел в двоично-десятичной системе счисления. При этом арифметические действия над двоично-десятичными числами выполняются по специальным правилам. В машинах, в которых принята двоичная система счисления, двоично-десятичная запись чисел применяется как промежуточная для ввода чисел в машину.

Комбинированные записи чисел. Для сокращения длины записей p-ичных чисел или любых кодов, являющихся по-

следовательностями p-ичных цифр, иногда применяют так называемую *комбинированную* запись. Группируют p-ичные разряды и вместо каждой группы пишут одну цифру системы счисления, имеющей основанием p^k, где k — количество разрядов в группе. При этом могут применяться системы счисления с различными основаниями (количества разрядов в группах могут быть неодинаковыми). Для того чтобы комбинированную запись можно было расшифровать, то есть определить изображаемое ею p-ичное число, указывают, на каком месте применяются цифры какой системы счисления.

Пример 1.15. Двоичные коды

10111101
01100001

могут быть представлены в виде комбинированных записей

2 75
1 41,

в которых старшая цифра — четверичная, а остальные — восьмеричные.

Комбинированная запись, как правило, применяется для изображения команд и чисел на бланках. Чтобы сделать ненужной ее расшифровку, клавишные устройства, предназначенные для ввода команд и чисел в машину, специально приспосабливают к применяемой комбинированной записи.

2. Перевод чисел из одной системы счисления в другую. 1-й случай. Наиболее прост перевод чисел из p-ичной позиционной системы в q-ичную (или обратно), если имеет место соотношение $p = q^k$ (k — целое положительное) и обе системы имеют либо неотрицательные, либо симметричные базы.

В этом случае перевод из p-ичной системы счисления в q-ичную производят «поразрядно», заменяя каждую p-ичную цифру равным ей k-разрядным числом, записанным в q-ичной системе счисления. Перевод из q-ичной системы в p-ичную при этом производят следующим образом. Двигаясь от запятой вправо и влево, разбивают q-ичную запись числа на группы по k цифр. Если при этом самая левая или самая правая группы окажутся неполными, к ним приписывают соответственно слева и справа столько нулей, чтобы каждая из

них содержала k цифр. После этого каждую группу q-ичных цифр заменяют одной p-ичной цифрой, равной числу, обозначенному этой группой q-ичных цифр.

Пример 1.16. Системы счисления с неотрицательными базами:

а) восьмеричное число 273,54 переводится в двоичную систему счисления ($8 = 2^3$) следующим образом:

$$273{,}54 = (010)\ (111)\ (011),\ (101)\ (100) = 10111011,\ 1011;$$

б) двоичное число 11011,0011 переводится в восьмеричную систему счисления ($8 = 2^3$) следующим образом:

$$11011{,}0011 = 11\ (011),\ (001)\ 1 = (011)\ (011),\ (001)\ (100) = 33{,}14;$$

в) шестнадцатеричное число $a5,\ b\bar{1}e$ переводится в двоичную систему счисления ($16 = 2^4$) следующим образом:

$$a5,\ b\bar{1}e = (1010)\ (0101),\ (1011)\ (0001)\ (1110) = 10100101,\ 10110001111.$$

Пример 1.17. Системы счисления с симметричными базами:

а) из девятеричной системы счисления число $24\bar{1}2,\ 01\bar{3}$ переводим в троичную систему счисления ($9 = 3^2$) следующим образом:

$$24\bar{1}2,\ 01\bar{3} = (1\bar{1})\ (\bar{1}\bar{1})\ (01)\ (\bar{1}1),\ (00)\ (01)\ (\bar{1}0) = 1\bar{1}\bar{1}01\bar{1}1,\ 0001\bar{1};$$

б) из троичной системы счисления число $110\bar{1}\bar{1}01,\ 110\bar{1}\bar{1}$ в девятеричную систему счисления ($9 = 3^2$) переводится следующим образом:

$$110\bar{1}\bar{1}01,\ 110\bar{1}\bar{1} = 1\ (10)\ (\bar{1}\bar{1})\ (01),\ (\bar{1}1)\ (0\bar{1})\ \bar{1} =$$
$$= (01)\ (10)\ (\bar{1}\bar{1})\ (01),\ (11)\ (0\bar{1})\ (\bar{1}0) = 13\bar{4}1,\ 4\bar{1}\bar{3}.$$

2-й случай. Легко осуществляется также перевод из одной системы счисления в другую, если основания систем одинаковы, а базы различны. Он заключается в том, что каждая цифра числа, принадлежащая старой и не принадлежащая новой системе счисления, рассматривается как сумма (или разность) двух цифр новой системы счисления. В записи числа упомянутую цифру старой системы заменяют первой из двух выбранных цифр новой системы счисления, а вторую из них подписывают под первой. Затем под остальными цифрами числа, принадлежащими и старой и новой системам, подписывают нули. Получаются два числа в новой системе счисления, которые складывают (или вычитают) по правилам, принятым в новой системе счисления.

Пример 1.18. Числа $1\bar{1}01\bar{1}$, $0\bar{1}$ и $\bar{1}\bar{1}0011$, $1\bar{1}1$, записанные в троичной системе счисления с симметричной базой, требуется

перевести в троичную систему счисления с неотрицательной базой (в которой используются цифры 0, 1, 2). При переводе каждую цифру $\bar{1}$ рассматриваем как разность 0—1 цифр новой системы счисления.

Перевод числа $1\bar{1}01\bar{1},0\bar{1}$:

$$\begin{array}{r} 10010{,}00 \\ -\ 01001{,}01 \\ \hline 2001{,}22 \end{array}$$

Следовательно,

$$1\bar{1}01\bar{1},0\bar{1} = 2001{,}22.$$

Перевод числа $\bar{1}\bar{1}0011,1\bar{1}1$:

$$\begin{array}{r} 000011{,}101 \\ -\ 110000{,}010. \end{array}$$

Так как в этом случае уменьшаемое меньше вычитаемого, а отрицательных чисел база новой системы счисления не содержит, то из большего числа отнимаем меньшее и ставим перед результатом знак минус. Получаем

$$\bar{1}\bar{1}0011,1\bar{1}1 = -102211{,}202.$$

Пример 1.19. При переводе числа 7803,12, записанного в девятеричной системе счисления с неотрицательной базой, в девятеричную систему счисления с симметричной базой, полагаем

$$7 = 4 + 3; \quad 8 = 4 + 4.$$

Получаем

$$\begin{array}{r} 4403{,}12 \\ +\ 3400{,}00 \\ \hline 1\bar{1}\bar{1}03{,}12 \end{array}$$

Таким образом,

$$7803{,}12 = 1\bar{1}\bar{1}03{,}12.$$

3-й случай. Перевод числа из p-ичной в q-ичную ситему счисления, когда базы обеих систем счисления неотрицательны и $p \neq q^k$, производится отдельно для целой и дробной частей числа (подробно см., например, [37]).

Перевод целой части числа. Число q записывают в p-ичной системе счисления. Делят в p-ичной системе целую часть подвергающегося переводу числа на q; в остатке получают число, равное последней цифре искомой q-ичной записи. Полученное частное снова делят на q; в остатке получают число, равное предпоследней цифре q-ичной записи, и т. д. Процесс повторяют до тех пор, пока в частном не будет получено число, меньшее чем q, которое равно первой цифре q-ичной записи.

Пример 1.20. Десятичное число 191 требуется перевести в восьмеричную систему счисления:

$$\begin{array}{r|l} 191 & 8 \\ \hline 16 & 23 \mid 8 \\ \hline 31 & 16 \mid 2 \\ 24 & \\ \hline 7 & \end{array}$$

Десятичное число 191 имеет восьмеричную запись 277.

Перевод дробной части числа. Число q записывают в p-ичной системе счисления. Умножают в p-ичной системе дробь на q. Целая часть произведения равна первой цифре q-ичной записи дроби. Дробную часть произведения снова умножают на q. Целая часть произведения равна следующей цифре q-ичной записи дроби. Процесс продолжают либо до тех пор, пока он не окончится получением целого произведения, либо до тех пор, пока не получат требуемого количества цифр q-ичной записи дроби.

Пример 1.21. Требуется десятичное число 0,6875 перевести в восьмеричную систему счисления:

$$\begin{array}{r|l} 0 & 6875 \\ & \quad 8 \\ \hline 5 & 5000 \\ & \quad 8 \\ \hline 4 & 0000 \end{array}$$

Получена восьмеричная дробь 0,54.

Комбинируя приведенные правила перевода, можно без труда произвести перевод числа из любой позиционной системы счисления в любую другую позиционную систему счисления.

Перевод из других систем счисления в общепринятую десятичную (в силу того, что выполнение деления в других системах счисления для нас непривычно) иногда производят с помощью формулы (1.2). В ее правой части p-ичные цифры, само число p и показатели степеней записывают в десятичной системе счисления, после чего вычисляют значение полученного выражения уже по правилам действий, принятых в десятичной системе счисления.

3. Формы представления чисел в машинах. Обычно числа в машинах представляются в одной из двух форм: с *фиксированной запятой* или с *плавающей запятой* *).

Представление чисел с фиксированной запятой. В форме с фиксированной запятой число записывается последовательностью изображающих его цифр. Обычно для машины, в которой принята такая форма изображения чисел, положение запятой относительно старшего разряда одинаково для всех чисел и установлено раз и навсегда. Это позволяет запятую в изображении числа опускать. Если рабочая система счисления имеет неотрицательную базу, то, помимо цифр числа, должен быть изображен и его знак. Таким образом, кроме цифровых разрядов, в изображении числа имеются знаковые разряды.

Абсолютная погрешность при изображении в ячейке чисел с запятой, фиксированной после определенного разряда, не превосходит по величине единицы младшего разряда, то есть, как говорят, максимальная абсолютная погрешность при этом постоянна.

Машины, в которых принята форма представления чисел с фиксированной запятой, т. е. арифметические устройства которых специально сконструированы для действий над числами, заданными в этой форме, называют *машинами с фиксированной запятой.* При выполнении некоторых операций над числами, представленными в машине с фиксированной запятой, могут получаться результаты, которые невозможно представить в такой же форме в машине. Например, при умножении неправильных дробей может получиться число разрядов целой части произведения больше, чем допустимое для представления в машине. Такое явление называется *переполнением разрядной сетки.* Если в машине предусмотрено только представление правильных дробей, т. е. запятая фиксирована перед старшим разрядом, то переполнение разрядной сетки при умножении не наступает. В связи с этой особенностью в большинстве машин с фиксированной запятой принято именно такое положение запятой.

Если количество разрядов дробной части числа больше количества разрядов, принятого для изображения дробной

*) Иногда неудачно первую из этих форм называют *естественной*, а вторую — *нормальной.*

части чисел в машине, то избыточные младшие разряды отбрасываются, возможно с округлением последнего из удерживаемых разрядов. При этом некоторые малые числа, отличные от нуля, могут оказаться записанными в виде нуля. Числа, не равные нулю, но представляемые в машине в виде нуля, называются *машинными нулями*.

Представление чисел с плавающей запятой. При форме представления с плавающей запятой число считают приведенным к виду

$$N = mp^n, \tag{1.3}$$

где m — число с запятой, фиксированной после определенного разряда, называемое *мантиссой*, n — целое число, называемое *порядком*, а p — основание рабочей системы счисления, принятой в машине. Число с плавающей запятой записывается в виде пары чисел m, n, т. е. в виде последовательности цифр, состоящей из двух подпоследовательностей, в одну из которых входят цифры мантиссы, а в другую — цифры порядка. Если рабочая система счисления имеет неотрицательную базу, то как мантисса, так и порядок снабжаются знаками, изображение которых осуществляется путем введения знаковых разрядов.

Запятую в мантиссах, как правило, фиксируют перед старшим (цифровым) разрядом, то есть в качестве мантисс берут правильные дроби. Это связано с удобством умножения мантисс (их произведение будет при этом также правильной дробью).

Если в записи (1.3) старшая цифра мантиссы отлична от нуля, то число N называют *нормализованным* (в противном случае — *ненормализованным*). При запятой, фиксированной перед старшим разрядом мантиссы, нормализованными являются числа, для которых

$$\frac{1}{p} \leqslant |m| < 1.$$

Пример 1.22. Десятичное число 25,2 в форме с плавающей запятой (при запятой мантиссы, фиксированной перед старшим разрядом) может иметь любой из видов $0{,}252 \cdot 10^2$, $0{,}0252 \cdot 10^3$, $0{,}00252 \cdot 10^4$ и т. д. Нормализованной является только запись $0{,}252 \cdot 10^2$, а все остальные являются ненормализованными.

Если в ячейке количество цифровых разрядов мантиссы равно μ, то относительная погрешность при изображении в этой ячейке чисел в нормализованной форме с плавающей

запятой не превосходит величины $p^{1-\mu}$, где p — основание рабочей системы счисления. Относительная погрешность при изображении чисел в ненормализованном виде может достигать весьма больших значений.

Машины, в которых принята форма представления чисел с плавающей запятой, т. е. арифметические устройства которых сконструированы для действий над числами, заданными в этой форме, называются *машинами с плавающей запятой.*

В машинах с плавающей запятой также возможно *переполнение разрядной сетки,* которое заключается теперь в том, что в результате какой-либо операции возникает число, имеющее порядок с большей разрядностью, чем допустимая при представлении порядка в машине. В машинах с плавающей запятой возможно появление *машинных нулей,* т. е. нормализованных чисел, отличных от нуля, но имеющих порядок, меньший самого малого порядка, представимого в машине; обычно машинные нули записываются в виде «нулевой» мантиссы и «нулевого» порядка.

Если количество разрядов мантиссы больше количества, принятого для ее изображения в машине, то избыточные младшие разряды отбрасываются, возможно с округлением последнего из удерживаемых.

Диапазон представимых чисел. Существенной характеристикой машины является множество чисел, представимых в ячейке ее памяти. Для машин с фиксированной запятой, использующих систему счисления с неотрицательной или с симметричной базой, это множество содержит нуль, симметрично относительно нуля (то есть вместе с каждым числом N содержит и число $-N$) и может быть представлено в виде возрастающей арифметической прогрессии, разность которой равна наименьшему из представимых положительных чисел. Поэтому для описания множества чисел с фиксированной запятой, представимых в ячейке машины, достаточно указать наибольшее и наименьшее из представимых положительных чисел, или так называемый *диапазон* представимых чисел (точнее — их абсолютных величин).

Диапазон чисел, представимых в ячейке памяти машины с фиксированной запятой, использующей двоичную систему счисления в качестве рабочей, можно оценить с помощью неравенств

$$2^{-\nu} \leqslant |N| \leqslant 2^{n} - 2^{-\nu},$$

где N — число, n — количество цифровых разрядов, отведенных для целой части числа, ν — количество разрядов, предназначенных для дробной части (не считая знаковых разрядов).

Для машин с плавающей запятой, имеющих рабочую систему счисления с неотрицательной или симметричной базой, множество чисел, представимых в ячейке памяти, можно получить, указывая диапазон мантисс (которые являются числами с фиксированной запятой) и наибольшее возможное значение порядка (порядки являются целыми числами, их множество содержит нуль и симметрично относительно нуля). Характеризуя машины с плавающей запятой, обычно вместо полного описания множества представимых чисел приводят их диапазон (точнее — диапазон их абсолютных величин) или даже диапазон нормализованных представимых чисел.

Для машин с плавающей запятой и двоичной рабочей системой счисления диапазон нормализованных представимых чисел, если запятые в мантиссах фиксированы перед старшими разрядами, оценивается неравенствами

$$2^{-2^{\rho}} \leqslant |N| \leqslant (1 - 2^{-\mu})\, 2^{2^{\rho}-1},$$

где N — по-прежнему число, μ — количество цифровых разрядов мантиссы, а ρ — количество цифровых разрядов порядка.

Машины с плавающей запятой обладают значительно более широкими диапазонами представимых чисел, чем машины с фиксированной запятой при тех же количествах цифровых разрядов ячеек.

§ 3. Элементы математической логики

Математическая логика — раздел общей логики, разработанный применительно к потребностям математики при широком использовании математических методов. В течение длительного времени математическая логика была только наукой, изучающей математические доказательства. С появлением программно-управляемых машин она приобрела также прикладное значение.

1. Двоичные функции и алгебрологические операции. Переменная x, принимающая значения 0 и 1, называется *двоичной* переменной.

Однозначная функция n двоичных переменных

$$F(x_1, x_2, \ldots, x_n),$$

принимающая только два значения 0 и 1 или одно из этих значений, называется *двоичной функцией.*

Существует только четыре различные двоичные функции одного двоичного переменного, приведенные в табл. 1.9.

Таблица 1.9. Двоичные функции одного двоичного переменного

x \ $f(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
0	0	0	1	1
1	0	1	0	1

Две из указанных функций являются тождественными постоянными: $f_1(x) = 0$; $f_4(x) = 1$.

Функцию $f_2(x)$ обычно обозначают символом $\overline{\overline{x}}$. Из табл. 1.9 видно, что эта функция удовлетворяет соотношению

$$\overline{\overline{x}} = x. \tag{1.4}$$

Функцию $f_3(x)$ принято обозначать символом $\bar{x}$. Знак «$\bar{}$» можно рассматривать не только как знак функции, но также как знак операции, позволяющей по значению x (которое может быть либо нулем, либо единицей) определять значение функции $\bar{x}$. Такая операция называется *инверсией* и может быть описана табл. 1.10, которая получается из табл. 1.9.

Таблица 1.10. Инверсия

$\bar{0} = 1$
$\bar{1} = 0$

Из табл. 1.10 видно, что

$$\overline{(\bar{x})} = x. \tag{1.5}$$

Формулы (1.4) и (1.5) позволяют истолковать знак «=» как знак двукратного применения операции инверсии. В дальнейшем считается, что

$$\bar{\bar{x}} = \overline{(\bar{x})}. \tag{1.6}$$

Различных двоичных функций от двух двоичных переменных существует шестнадцать (табл. 1.11).

Таблица 1.11. Двоичные функции от двух двоичных переменных

x	y \ F	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Две из указанных функций являются тождественными постоянными: $F_1(x, y) = 0$, $F_{16}(x, y) = 1$.

Функции $F_2(x, y)$, $F_8(x, y)$, $F_{10}(x, y)$, $F_{14}(x, y)$ обычно обозначают соответственно символами $x \wedge y$, $x \vee y$, $x \sim y$, $x \to y$. Символы $\wedge$, $\vee$, $\sim$, $\to$ можно рассматривать как знаки двухместных операций, позволяющих для чисел x и y (каждое из которых равно 0 или 1) получить значение соответственно $x \wedge y$, $x \vee y$, $x \sim y$, $x \to y$. Эти операции известны под названиями соответственно *логического умножения, логического сложения, равнозначности* и *импликации* и полностью описываются табл. 1.12—1.15, получаемыми из табл. 1.11.

Таблица 1.12.
Логическое умножение

$0 \wedge 0 = 0$
$0 \wedge 1 = 0$
$1 \wedge 0 = 0$
$1 \wedge 1 = 1$

Таблица 1.13.
Логическое сложение

$0 \vee 0 = 0$
$0 \vee 1 = 1$
$1 \vee 0 = 1$
$1 \vee 1 = 1$

Таблица 1.14.
Равнозначность

$0 \sim 0 = 1$
$0 \sim 1 = 0$
$1 \sim 0 = 0$
$1 \sim 1 = 1$

Таблица 1.15.
Импликация

$0 \to 0 = 1$
$0 \to 1 = 1$
$1 \to 0 = 0$
$1 \to 1 = 1$

Инверсию, логическое умножение, логическое сложение, равнозначность и импликацию, а также операции, результатами которых являются значения остальных функций, описанных в табл. 1.11, принято называть *алгебрологическими* операциями.

2. Понятие логических связей, значения истинности и высказывания. Простейшим разделом математической логики является исчисление высказываний, в котором рассматриваются сложные предложения, получающиеся из предложений, принимаемых за элементарные высказывания, путем присоединения к ним частицы «не» или путем соединения их в пары с помощью союзов «и», «или», «равнозначно» и «если ..., то...». Частицу «не» и перечисленные союзы называют *логическими связями* или *связками*. Для их обозначений применяют те же знаки, которые были использованы нами для обозначения алгебрологических операций. Теперь эти знаки называются знаками логических связей. Логические связи *не, и, или, равнозначно, если ..., то...* обозначают соответственно через $\overline{\ }$, $\wedge$, $\vee$, $\sim$ и $\to$. Необходимо заметить, что в русском языке союз «или» употребляется в различных смыслах. Знак $\vee$ обозначает неразделительный союз «или», применяемый, например, во фразе: «при звоне будильника Петр или Иван проснется» (здесь «или» не исключает возможности того, что проснутся оба).

В исчислении высказываний предложения оцениваются не с точки зрения их содержания (смысла), а по их истинности или ложности. Про истинное предложение говорят, что его *значение истинности* равно 1, а про ложное, что его *значение истинности* равно 0.

Всякое предложение, для которого имеет смысл утверждение о его истинности или ложности, рассматриваемое

в целом (т. е. без учета его внутренней структуры и содержания), называется *элементарным высказыванием.*

Предложения «9 — нечетное число», «снег красен», «Москва — столица СССР» являются элементарными высказываниями.

Теперь дадим определение *высказывания*:

1) элементарное высказывание является высказыванием;

2) если A и B — высказывания, то $\overline{A}$ (читается: *не* A), $A \wedge B$ (читается: A *и* B), $A \vee B$ (читается: A *или* B), $A \sim B$ (читается: A *равнозначно* B) и $A \to B$ (читается: *если* A, *то* B) — тоже высказывания; значения истинности таких сложных высказываний равны числам, которые получаются с помощью соответствующих логических операций, если в записях сложных высказываний заменить составляющие высказывания A и B их значениями истинности.

Если составляющие высказывания A и B сами являются сложными (т. е. записываются с помощью букв и знаков логических связей), то в записях, упоминаемых в пункте 2) определения, они заключаются в скобки.

Таким образом, при записи сложных высказываний применяются буквы (обозначающие элементарные высказывания), знаки логических связей и скобки (определяющие строение сложных высказываний).

Подчеркнем, что исчисление высказываний не дает способа определения истинности элементарных высказываний, но дает вполне четкие правила определения истинности сложных высказываний по истинности (но не по содержанию) составляющих высказываний. Сложные высказывания $A \wedge B$, $A \vee B$, $A \sim B$, $A \to B$ называются соответственно *конъюнкцией, дизъюнкцией, равнозначностью* и *импликацией* составляющих их высказываний A и B.

3. Основные свойства алгебрологических операций и логических связей. Два высказывания называются *эквивалентными*, если они оба истинны или оба ложны. Эквивалентность двух высказываний обозначают знаком равенства.

Условимся элементарное высказывание и его значение истинности обозначать одной и той же буквой. При соблюдении такого условия между эквивалентностями исчисления высказываний и равенствами исчисления алгебрологических операций существует взаимно однозначное соответствие,

позволяющее изучение свойств сложных высказываний заменить изучением алгебрологических операций. Очевидно, при этом запись $A=B$ может быть прочитана двояко: а) высказывания A и B эквивалентны; б) числа A и B равны. Точно так же запись $A=1$ ($A=0$) может быть прочитана двояко: а) высказывание A истинно (ложно); б) число A равно 1 (равно 0).

В дальнейшем мы будем обозначать истинное высказывание цифрой 1, а ложное — цифрой 0.

Свойства инверсии (и отрицания) видны из формул (1.5) и (1.6):

$$\bar{\bar{A}}=\overline{(\bar{A})}=A, \tag{1.7}$$

что на языке исчисления высказываний гласит: двойное отрицание высказывания эквивалентно самому высказыванию.

Свойства логического умножения (и конъюнкции) легко получаются при рассмотрении табл. 1.12:

$$\left.\begin{aligned} A\wedge B&=B\wedge A,\\ A\wedge 0&=0,\\ A\wedge 1&=A,\\ A\wedge A&=A,\\ A\wedge\bar{A}&=0,\\ A\wedge(B\wedge C)&=(A\wedge B)\wedge C. \end{aligned}\right\} \tag{1.8}$$

Первая из формул (1.8) означает, что для логического умножения (и конъюнкции) справедлив переместительный закон. Последняя из формул (1.8) означает, что для логического умножения (конъюнкции) справедлив сочетательный закон, и позволяет в многочленных конъюнкциях опускать скобки (например, можно писать $A\wedge B\wedge C$).

Основные свойства логического сложения (и дизъюнкции) получаются при исследовании табл. 1.13:

$$\left.\begin{aligned} A\vee B&=B\vee A,\\ A\vee 0&=A,\\ A\vee 1&=1,\\ A\vee A&=A,\\ A\vee\bar{A}&=1,\\ A\vee(B\vee C)&=(A\vee B)\vee C. \end{aligned}\right\} \tag{1.9}$$

Первая из формул (1.9) означает справедливость для логического сложения (и дизъюнкции) переместительного закона, а последняя формула — справедливость сочетательного закона. Из последней формулы вытекает возможность опускать скобки в многочленных дизъюнкциях (например, можно писать $A \vee B \vee C$).

Большое значение имеют формулы

$$A \wedge B = \overline{\bar{A} \vee \bar{B}}, \tag{1.10}$$

$$A \vee B = \overline{\bar{A} \wedge \bar{B}}, \tag{1.11}$$

позволяющие от логического умножения (сложения) перейти к логическому сложению (умножению). Другими словами, эти формулы позволяют конъюнкции (дизъюнкции) преобразовать в отрицание дизъюнкций (конъюнкций).

Наконец, отметим формулы

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C),$$

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C).$$

Первая из этих формул означает, что логическое умножение обладает распределительным свойством относительно логического сложения (аналогом этой формулы в обычной алгебре является $a(b+c) = ab + ac$). Вторая формула означает, что логическое сложение обладает распределительным свойством по отношению к логическому умножению (в обычной алгебре аналога этой формуле нет, ибо формула $a + bc = (a+b)(a+c)$ неверна).

Для импликации и равнозначности справедливы соотношения

$$A \to B = \bar{A} \vee B,$$

$$A \sim B = (A \to B) \wedge (B \to A),$$

$$A \sim B = (\bar{A} \vee B) \wedge (A \vee \bar{B}).$$

Значительный интерес представляет операция (и логическая связь), позволяющая из A и B получить $\overline{A \sim B}$ (отрицание равнозначности). Для этой операции (и логической связи) введен специальный знак $\approx$ (читается: неравнозначно), так что (по определению)

$$A \approx B = \overline{A \sim B}. \tag{1.12}$$

Операция неравнозначности полностью описывается табл. 1.16, которая легко может быть получена по табл. 1.14 и 1.10.

Таблица 1.16.
Отрицание равнозначности

$0 \not\sim 0 = 0$
$0 \not\sim 1 = 1$
$1 \not\sim 0 = 1$
$1 \not\sim 1 = 0$

Функция $x \not\sim y$ представляет собой не что иное, как функцию $F_7(x, y)$ из табл. 1.11. Логическая связь $\not\sim$ имеет смысл исключающего «или», поэтому сложное высказывание $A \not\sim B$ можно произносить так: *или A, или B.*

Наконец, отметим еще *операцию* (и логическую связь) *Шефера,* представляющую собой отрицание конъюнкции и обозначаемую косой чертой «/» (читается *несовместно*):

$$A/B = \overline{A \wedge B}. \tag{1.13}$$

Таблица операции Шефера (табл. 1.17) легко получается из табл. 1.12 и 1.10.

Таблица 1.17.
Операция Шефера

$0/0 = 1$
$0/1 = 1$
$1/0 = 1$
$1/1 = 0$

Функция x/y представляет собой не что иное, как функцию $F_{15}(x, y)$ из табл. 1.11.

Заменяя в (1.13) B на A, получаем после несложных преобразований

$$\overline{A} = A/A, \tag{1.14}$$

а из формул (1.13) и (1.14)

$$A \wedge B = (A/B)/(A/B). \tag{1.15}$$

Для уменьшения количества скобок в формулах вводят «старшинство» для знаков алгебрологических операций и соответственно знаков логических связей, подобно тому как это принято в алгебре (где считают что возведение в степень и извлечение корней старше, чем умножение и деление, а эти последние старше, чем сложение и вычитание). Принято считать, что знак $\vee$ старше знаков $\sim$, $\overline{\sim}$, $\to$ и $/$, знак $\wedge$ старше перечисленных знаков (в том числе и $\vee$), знак $\overline{}$ старше всех остальных знаков.

Это соглашение позволяет, например, вместо

$$(A \vee (B \wedge C)) \to ((A \wedge B) \vee (C \wedge D))$$

писать просто

$$A \vee B \wedge C \to A \wedge B \vee C \wedge D.$$

4. Полные системы алгебрологических операций и логических связей. Пусть $\theta(x_1, x_2, \ldots, x_n)$ — произвольная двоичная функция конечного числа двоичных переменных. Если θ отлична от тождественного нуля и $Q_1, Q_2, \ldots, Q_k$ — все точки ее области определения, в которых $\theta = 1$, то справедлива формула (см. [58], стр. 58—60)

$$\theta(x_1, x_2, \ldots, x_n) = \varphi_1 \vee \varphi_2 \vee \ldots \vee \varphi_k, \tag{1.16}$$

где

$$\varphi_j = y_{j,1} \wedge y_{j,2} \wedge \ldots \wedge y_{j,n} \qquad (j = 1, 2, \ldots, k),$$

причем для $i = 1, 2, \ldots, n$

$$y_{j,i} = \begin{cases} x_i, & \text{если } x_i = 1 \text{ в точке } Q_j; \\ \bar{x}_i, & \text{если } x_i = 0 \text{ в точке } Q_j. \end{cases}$$

Точно так же, если функция θ отлична от тождественной единицы и $R_1, R_2, \ldots, R_l$ — все точки ее области определения, в которых $\theta = 0$, то справедлива формула (см. [58], стр. 59—60)

$$\theta(x_1, x_2, \ldots, x_n) = \psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_l, \tag{1.17}$$

где

$$\psi_j = z_{j,1} \vee z_{j,2} \vee \ldots \vee z_{j,n} \qquad (j = 1, 2, \ldots, l),$$

причем для $i = 1, 2, \ldots, n$

$$z_{j,i} = \begin{cases} x_i, & \text{если } x_i = 0 \text{ в точке } R_j; \\ \bar{x}_i, & \text{если } x_i = 1 \text{ в точке } R_j. \end{cases}$$

Из формул (1.16) и (1.17) вытекает, что любую двоичную функцию конечного числа двоичных переменных можно представить в виде суперпозиции конечного числа операций $\overline{\ }$, $\wedge$, $\vee$ над (двоичными) независимыми переменными этой функции. В этом смысле систему операций $\overline{\ }$, $\wedge$, $\vee$ называют *полной*.

Формула (1.11) выражает операцию $\vee$ через операции $\overline{\ }$ и $\wedge$, следовательно, система операций $\overline{\ }$ и $\wedge$ также является полной. Формула (1.10) выражает операцию $\wedge$ через $\overline{\ }$ и $\vee$, поэтому система операций $\overline{\ }$ и $\vee$ полна. Формулы (1.14) и (1.15) выражают операции $\overline{\ }$ и $\wedge$ через единственную операцию Шефера, так что эта операция, взятая в единственном числе, образует полную систему.

Формулы (1.16) и (1.17) позволяют составлять выражения, являющиеся суперпозициями операций $\overline{\ }$, $\wedge$ и $\vee$ для любой двоичной функции конечного числа двоичных переменных, которая задана таблично.

Пример 1.23. Пусть двоичная функция $\theta(A, B, C)$ задана с помощью табл. 1.18. Требуется записать ее с помощью операций $\overline{\ }$, $\wedge$, $\vee$.

Таблица 1.18. Значения некоторой функции $\theta(A, B, C)$

A	B	C	θ
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Первый способ. Определяем точки, в которых $\theta(A, B, C) = 1$. Имеем $Q_1(0, 0, 1)$, $Q_2(1, 1, 0)$, $Q_3(1, 1, 1)$. Строим φ_i $(i = 1, 2, 3)$:

$$\varphi_1 = \overline{A} \wedge \overline{B} \wedge C,$$

$$\varphi_2 = A \wedge B \wedge \overline{C},$$

$$\varphi_3 = A \wedge B \wedge C.$$

Получаем по формуле (1.16)

$$\theta = \overline{A}\wedge\overline{B}\wedge C \vee A\wedge B\wedge\overline{C} \vee A\wedge B\wedge C.$$

После небольшого упрощения это дает

$$\theta = \overline{A}\wedge\overline{B}\wedge C \vee A\wedge B.$$

Второй способ. Отбирая точки, в которых $\theta(A, B, C) = 0$, имеем $R_1(0, 0, 0)$; $R_2(0, 1, 0)$; $R_3(0, 1, 1)$; $R_4(1, 0, 0)$; $R_5(1, 0, 1)$. Строим функции ψ_i $(i = 1, 2, 3, 4, 5)$:

$$\psi_1 = A \vee B \vee C, \qquad \psi_4 = \overline{A} \vee B \vee C,$$
$$\psi_2 = A \vee \overline{B} \vee C, \qquad \psi_5 = \overline{A} \vee B \vee \overline{C}.$$
$$\psi_3 = A \vee \overline{B} \vee \overline{C},$$

Получаем по формуле (1.17) еще одно выражение для $\theta(A, B, C)$:

$$\theta = (A \vee B \vee C) \wedge (A \vee \overline{B} \vee C) \wedge (A \vee \overline{B} \vee \overline{C}) \wedge (\overline{A} \vee B \vee C) \wedge (\overline{A} \vee B \vee \overline{C}).$$

После небольшого упрощения это дает

$$\theta = (B \vee C) \wedge (A \vee \overline{B}) \wedge (\overline{A} \vee B \vee \overline{C}).$$

Любое из двух выражений, полученных для θ, можно свести путем преобразований к другому.

5. Понятие предиката. Обозначим через M совокупность попарно-различных символов, обозначающих слова, из которых составляются элементарные высказывания. Предположим, что из символов, принадлежащих M, образованы упорядоченные наборы по n символов $(n \geqslant 1)$. Повторение одинаковых символов в пределах отдельного набора допускается. Пусть N — некоторое множество упомянутых наборов. Произвольный, но определенный набор, входящий в N, будем записывать как $(a_1, a_2, \ldots, a_n)$, понимая под a_i $(i = 1, 2, \ldots, n)$ отдельные символы.

Пусть $x_1, x_2, \ldots, x_n$ — последовательность попарно-различных символов, не принадлежащих M. Обозначим ее буквой S и будем называть *совокупностью аргументов* (каждый символ x_i будем называть *аргументом*).

Рассмотрим некоторую конфигурацию $\Phi(x_1, x_2, \ldots, x_n)$ символов, каждый из которых принадлежит либо M, либо S и удовлетворяет условиям:

1) в состав $\Phi(x_1, x_2, \ldots, x_n)$ входит хотя бы один символ из S;

2) если $(a_1, a_2, \ldots, a_n)$ — произвольный набор символов, входящий в N, то, заменяя x_i для $i=1, 2, \ldots, n$ всюду в $\Phi(x_1, x_2, \ldots, x_n)$ соответственно символами a_i, мы превращаем $\Phi(x_1, x_2, \ldots, x_n)$ в запись некоторого высказывания $\Phi(a_1, a_2, \ldots, a_n)$.

Конфигурация символов $\Phi(x_1, x_2, \ldots, x_n)$ называется *соотношением*, а множество N — его *областью задания*.

Пример 1.24. Пусть M — совокупность русских слов. Предположим, что из русских слов образованы наборы, состоящие из одного слова: «лев», «волк», «олень», «корова», «лиса», «белка». Множество N состоит из перечисленных наборов. Конфигурация слов «x — травоядное животное» является соотношением. Заменяя символ «x» словом «лев», получаем ложное высказывание: «лев — травоядное животное»; заменяя символ «x» словом «корова», получаем истинное высказывание: «корова — травоядное животное».

Пример 1.25. Можно считать, что символами, входящими в M, являются обозначения целых чисел в десятичной системе счисления, знаки алгебраических действий и знаки $=, >, \geqslant, <, \leqslant$. Тогда конфигурация символов $x > 2y$ будет соотношением, область задания N которого состоит из всевозможных пар целых чисел. Выбирая пару чисел 2; 5 и заменяя x числом 2, а y числом 5, получаем ложное высказывание $2 > 2 \cdot 5$ (читается: два больше чем дважды пять). Выбирая из N пару символов 5; 2 и заменяя x числом 5, а y числом 2, получаем запись истинного высказывания $5 > 2 \cdot 2$ (пять больше чем дважды два).

Является соотношением и конфигурация

$$\frac{x+y}{1+x^2+y^2} > 3.$$

Если некоторый упорядоченный набор $(a_1, a_2, \ldots, a_n)$ обращает соотношение $\Phi(x_1, x_2, \ldots, x_n)$ в запись истинного высказывания, то говорят, что для $(a_1, a_2, \ldots, a_n)$ соотношение $\Phi(x_1, x_2, \ldots, x_n)$ *выполнено*. Если же упомянутый набор обращает указанное соотношение в ложное высказывание, то говорят, что для $(a_1, a_2, \ldots, a_n)$ оно *не выполнено*.

Два соотношения называются *эквивалентными*, если они имеют общую область задания и для каждого упорядоченного набора, принадлежащего этой области задания, либо оба выполнены, либо оба не выполнены.

Пример 1.26. Пусть M, кроме символов, указанных в примере 1.21, содержит знаки логических связей $-$, $\wedge$ и $\vee$, а N имеет

прежний смысл. Тогда конфигурация

$$\overline{x > y \vee x = y}$$

является соотношением. Для тех же M и N соотношением является

$$x < y,$$

причем последнее соотношение эквивалентно первому.

Поставим в соответствие каждому набору $(a_1, a_2, \ldots, a_n)$, принадлежащему N, если для него соотношение $\Phi(x_1, x_2, \ldots, x_n)$ выполнено, число 1, а если не выполнено, то число 0. Этим самым мы задаем на множестве N наборов некоторую двоичную функцию $F(x_1, x_2, \ldots, x_n)$. Такая двоичная функция называется *предикатом с содержанием* $\Phi(x_1, x_2, \ldots, x_n)$ *и областью задания* N.

Понятия соотношения и предиката являются соответственно обобщениями понятий высказывания и его значения истинности.

В дальнейшем будем считать, что множество M символов, применяемых для записи соотношений, всегда содержит знаки логических связей. При этом, очевидно, отрицание соотношения, а также конъюнкция, дизъюнкция, равнозначность, отрицание равнозначности, импликация и несовместность (связь Шефера) двух соотношений являются соотношениями. Если F_1 — предикат с содержанием Φ_1, то предикат, содержанием которого является $\overline{\Phi_1}$, будем записывать как $\overline{F_1}$. Если, кроме того, F_2 — предикат с содержанием Φ_2, то предикаты с содержаниями $\Phi_1 \wedge \Phi_2$, $\Phi_1 \vee \Phi_2$, $\Phi_1 \sim \Phi_2$ и т. д. будем записывать соответственно как $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \sim F_2$ и т. д.

В дальнейшем часто мы не будем вводить обозначений для предикатов, а, говоря о предикате, будем писать соотношение, являющееся его содержанием.

§ 4. Элементы теории алгорифмов

1. Понятие алгорифма. Понятие алгорифма возникло в области математики в связи с поисками общих методов решения классов однотипных задач и первоначально описывалось следующим образом [51].

Алгорифм — это точное общепонятное предписание, определяющее процесс преобразования исходных данных в искомый результат.

Алгорифм обладает тремя свойствами:

а) *определенностью* (т. е. точностью, не оставляющей места для произвола) и общепонятностью;

б) *массовостью*, т. е. возможностью исходить из любых исходных данных, принадлежащих некоторому, довольно обширному (иногда даже бесконечному) множеству G исходных данных;

в) *результативностью*, то есть свойством определять процесс, который для любых допустимых исходных данных (т. е. принадлежащих вышеупомянутому множеству G) приводит к получению искомого результата. Применение алгорифма к исходным данным, не входящим в G, может оказаться невозможным или безрезультативным (определяемый алгорифмом процесс может никогда не закончиться).

Для алгорифмов также характерны дискретность определяемого алгорифмом процесса и простота операций, выполняемых на каждом шагу.

В простейшем случае алгорифм является просто совокупностью (обозначим ее через R) элементарных предписаний. Рассмотрим сперва этот простейший случай.

Прежде всего заметим, что в множестве R элементарных предписаний, образующих такой алгорифм, должно быть *отмечено* элементарное предписание, определяющее *первый* акт процесса выполнения алгорифма. Кроме того, каждое элементарное предписание из R должно определять *процесс перехода* к другому элементарному предписанию, принадлежащему R, или *прекращение* общего процесса выполнения алгорифма. Последнее можно рассматривать как частный («нулевой») случай перехода.

Затем среди элементарных предписаний (входящих в R) хотя бы одно должно определять *процесс преобразования* исходных для него данных в искомый для него результат. Без этого алгорифм не был бы результативным.

Условимся определяемый элементарным предписанием процесс преобразования исходных (для него) данных в искомый (для него) результат называть *основным процессом*. Удобно считать, что элементарное предписание, не определяющее основного процесса, определяет нулевой основной процесс.

Итак, можно сказать, что *любое элементарное предписание определяет два процесса: основной процесс и процесс перехода*. Совокупность этих процессов представляет собой *акт*.

Пример 1.27. Алгорифм Евклида предназначен для нахождения общего наибольшего делителя двух положительных целых чисел. Если обозначить два положительных целых числа соответственно через x и y, а искомый результат — через z и если считать, что исполнитель алгорифма умеет выполнять операции сложения и вычитания целых чисел, а также операцию вычисления предиката $P(a, b)$, определяемую условием

$$P(a, b) = \begin{cases} 1 \text{ при } a > b, \\ 0 \text{ при } a \leqslant b, \end{cases}$$

то алгорифм Евклида можно записать следующим образом:

1°. Вычислить величину

$$\omega_1 = P(x, y).$$

Перейти к пункту, имеющему номер $2\omega_1 + 2$.

2°. Вычислить величину

$$\omega_2 = P(y, x).$$

Перейти к пункту, имеющему номер $2\omega_2 + 3$.

3°. Положить

$$z = x.$$

Перейти к п. 6°.

4°. Вычислить разность

$$x - y$$

и эту величину в дальнейшем принимать за x. Перейти к п. 1°.

5°. Вычислить разность

$$y - x$$

и эту величину в дальнейшем принимать за y. Перейти к п. 1°.

6°. Прекратить процесс.

Массовость алгорифма Евклида заключается в том, что его можно применить к *любой* паре целых положительных чисел. Результативность его состоит в том, что он определяет процесс, приводящий для любой пары положительных чисел к *получению* их общего наибольшего делителя. Определенность алгорифма Евклида обеспечивается тем, что исполнитель *умеет* выполнять определяемые пунктами алгорифма действия и однозначно *понимает*, каких действий требует каждый пункт.

В примере 1.27 процесс перехода определяется указанием номера элементарного предписания, которое подлежит выполнению вслед за данным актом. Фраза «перейти к пункту номер такой-то» по существу означает — фиксировать внимание на указанном пункте, то есть *перенести его на заранее заданное «место»*: в свою память. Перед началом процесса на этом заранее заданном «месте» должно находиться то элементарное предписание (пункт алгорифма), которое отмечено

как первое. Тогда, считая элементарные предписания и исходные данные словами в некотором алфавите, мы можем как основной процесс, так и процесс перехода рассматривать как процесс преобразования слова, состоящего из алгорифма, подлежащего выполнению элементарного предписания и исходных данных. Прекращение процесса выполнения наступает тогда, когда получается совокупность слов, содержащая на заданном месте элементарное предписание, определяющее прекращение процесса.

Пример 1.28. Вернемся к алгорифму Евклида. Обозначим его элементарные предписания в порядке их номеров, указанных в примере 1.27 соответственно через Q_1, Q_2, Q_3, Q_4, Q_5, Q_6.

Рассмотрим алфавит

$$\{Q_1,\ Q_2,\ Q_3,\ Q_4,\ Q_5,\ Q_6,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ \lambda\},$$

где λ — буква, которую мы будем употреблять как знак раздела между числами. Желая применить алгорифм Евклида к паре целых чисел $x=6$, $y=9$, составим слово

$$Q_1Q_1Q_2Q_3Q_4Q_5Q_66\lambda 9\lambda.$$

Здесь на заранее заданном (на самом левом) месте стоит первое элементарное предписание Q_1. Выполняя Q_1, определяем $\omega_1=0$ и производим переход к элементарному предписанию номер 2, т. е. переходим к слову

$$Q_2Q_1Q_2Q_3Q_4Q_5Q_66\lambda 9\lambda.$$

Дальнейшее выполнение алгорифма Евклида представляет собой нижеприведенную последовательность преобразований

$$\begin{gathered}
Q_5Q_1Q_2Q_3Q_4Q_5Q_66\lambda 9\lambda,\\
Q_1Q_1Q_2Q_3Q_4Q_5Q_66\lambda 3\lambda,\\
Q_4Q_1Q_2Q_3Q_4Q_5Q_66\lambda 3\lambda,\\
Q_1Q_1Q_2Q_3Q_4Q_5Q_63\lambda 3\lambda,\\
Q_2Q_1Q_2Q_3Q_4Q_5Q_63\lambda 3\lambda,\\
Q_3Q_1Q_2Q_3Q_4Q_5Q_63\lambda 3\lambda,\\
Q_6Q_1Q_2Q_3Q_4Q_5Q_63\lambda 3\lambda 3.
\end{gathered}$$

На этом процесс выполнения алгорифма Евклида окончен, и получен искомый результат $z=3$, вхождением которого (в данном случае) кончается последнее полученное слово.

То обстоятельство, что электронные цифровые программно-управляемые машины способны перерабатывать информацию, представленную в виде слов, под воздействием программ, состоящих тоже из слов, позволяет ограничиться рассмотрением алгорифмов, которые сами являются словами и преобразуют исходные данные-слова в искомые результаты-

слова (заметим, что последовательность слов тоже может рассматриваться как слово). Интуитивно ясно, что другие виды информации (графики, матрицы, наборы таблиц) могут быть представлены с помощью слов (в линейном виде) так, что принятое ограничение не является существенным.

Из вышеприведенного описания понятия алгорифма видно, что каждый алгорифм определяет некоторое семейство процессов преобразования (чтобы выделить отдельный процесс, нужно кроме алгорифма указать исходные данные). Другими словами, алгорифмы можно считать средством описания семейств процессов преобразования. Тогда класс родственных между собой алгорифмов можно (учитывая, что алгорифмы являются словами) рассматривать как язык для описания процессов.

2. Композиция и расчленение слов. Будем считать, что буквы каждого слова перенумерованы в порядке их расположения в этом слове.

Пусть σ — последовательность целых чисел

$$\lambda_1, \lambda_2, \ldots, \lambda_n,$$

удовлетворяющая условию

$$1 \leqslant \lambda_i \leqslant m \qquad (i = 1, 2, \ldots, n).$$

Если P и Q — слова в алфавите A соответственно длины m и n такие, что i-я буква слова Q одинакова с λ_i-й буквой слова P, то говорят, что Q является *вхождением* (или входит) в P с характеристикой σ.

Если Q входит в P с характеристикой σ, числа которой удовлетворяют условию

$$\lambda_{i+1} = \lambda_i + 1 \qquad (i = 1, 2, \ldots, n-1),$$

то говорят, что Q *просто входит* (или является *простым вхождением*) в P. Иногда при этом говорят, что Q просто входит в P на λ_1-м месте (при $\lambda_1 = 1$ говорят также, что P *начинается* простым вхождением Q, а при $\lambda_n = m$ — что P *кончается* простым вхождением Q).

Если Q просто входит в P несколько раз с характеристиками $\sigma_1, \sigma_2, \ldots, \sigma_k$, то та характеристика, первый элемент которой не превосходит первых элементов остальных характеристик, определяет так называемое *первое* простое

вхождение Q в P. Можно считать, что любое слово начинается вхождением пустого слова.

Если слова $Q_1, Q_2, \ldots, Q_k$ входят в слово P длины m с характеристиками соответственно $\sigma_1, \sigma_2, \ldots, \sigma_k$, причем каждое натуральное число, не превосходящее m, является элементом хотя бы одной из этих характеристик, то: 1) совокупность $\sum$ характеристик $\sigma_1, \sigma_2, \ldots, \ldots, \sigma_k$ называется *законом композиции*; 2) слово P называется *композицией с законом* $\sum$ *слов* $Q_1, Q_2, \ldots, Q_k$.

Процесс построения слова P по заданным словам $Q_1, Q_2, \ldots, Q_k$ и закону $\sum$ также называется *композицией с законом* $\sum$. Процесс построения слов $Q_1, Q_2, \ldots, Q_k$ по заданному слову P и закону $\sum$ называется *расчленением слова* P *с законом* $\sum$.

Пример 1.29. Пусть в алфавите {а, в, д, з, и, н, о, п, р, я} задано слово P — «производная», имеющее длину $n = 11$. Характеристика $\sigma_1 = (1, 3, 2, 3, 8, 10)$ определяет вхождение в P слова Q — «порода». Характеристики $\sigma_2 = (6, 7, 8, 9, 10, 11)$ и $\sigma_3 = (1, 2, 4, 5)$ определяют вхождения в P слов Q_2 — «водная» и Q_3 — «приз». Вместе с σ_1 они обладают тем свойством, что любое целое число из промежутка

$$1 \leqslant \lambda \leqslant 11$$

является элементом хотя бы одной из них. Совокупность характеристик $\sigma_1, \sigma_2, \sigma_3$ может быть принята за закон $\sum$ композиции, и тогда слово «производная» можно считать композицией с законом $\sum$ слов «порода», «водная» и «приз».

Для того чтобы для слов $Q_1, Q_2, \ldots, Q_k$ совокупность числовых последовательностей $\sigma_1, \sigma_2, \ldots, \sigma_k$ представляла собой закон композиции, необходимо и достаточно:

1) чтобы каждое целое число из промежутка

$$1 \leqslant \lambda \leqslant m,$$

где m — наибольшее из чисел, входящих в перечисленные последовательности, являлось элементом хотя бы одной из последовательностей;

2) чтобы i-я буква слова Q_l была одинакова с j-й буквой слова Q_t, если

$$\lambda_i^l = \lambda_j^t,$$

где λ_i^l и λ_j^t — соответственно элементы последовательностей σ_l и σ_t, принадлежащих Σ.

Если слова P и Q имеют длины $L(P)=m$, $L(Q)=n$, а закон композиции σ_1, σ_2 состоит из последовательностей

$$1, 2, \ldots, m$$

и

$$m+1, m+2, \ldots, m+n,$$

то композицию слов P и Q с законом σ_1, σ_2, называют *объединением* P и Q и записывают в виде PQ.

Объединение пустого слова с любым словом P равно P.

3. Равнозначные алфавиты. Если между буквами двух неравных алфавитов установлено взаимно однозначное соответствие, то эти алфавиты называются *равнозначными*. Например, алфавиты русский письменный, русский печатный и русский алфавит для слепых равнозначны.

Буквы равнозначных алфавитов, связанные взаимно однозначным соответствием, называются *равнозначными*.

Очевидно, объемы равнозначных алфавитов равны.

Пусть A и B — равнозначные алфавиты. Заменяя каждую букву слова в A равнозначной ей буквой в B, мы получим слово в B, которое называется *равнозначным* исходному слову в A. Переход от слова в алфавите A к равнозначному ему слову в алфавите B называется *элементарной перекодировкой*.

Пример 1.30. Алфавиты

$$A — \{0, 1, 2, 3, 4, 5, 6, 7\} \text{ и } B — \{000, 001, 010, 011, 100, 101, 110, 111\},$$

применяемые соответственно в восьмеричной и в восьмеричной двоично-кодированной системах счисления для записи целых неотрицательных чисел, между собой равнозначны.

Запись любого неотрицательного целого числа в восьмеричной системе счисления, например 3201, является словом в алфавите A. Заменяя каждую букву слова в A (например, слова 3201) равнозначной ей буквой в B, получаем слово в B (например, 011010000001), равнозначное исходному слову.

4. Предметное тождество слов. Пусть алфавиты A и B между собой не равны. Если слово P в алфавите A неотличимо от слова Q в алфавите B, то эти слова называются

предметно-тождественными. В простейшем (тривиальном) случае предметно-тождественные слова одинаковы. В самом деле, если пересечение алфавитов AB непусто, то всякое слово в алфавите A, составленное из букв, которые одинаковы с буквами части AB этого алфавита, неотличимо от некоторого слова в алфавите B. В более сложных случаях предметное тождество слов связано с тем, что буквы алфавита A и буквы алфавита B состоят из некоторых элементов, которые по-разному группируются в буквы в зависимости от того, к которому из алфавитов мы относим слово.

Пример 1.31. Рассмотрим алфавиты

$$A\{00,\ 01,\ 10,\ 11\}$$

и

$$B\{000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111\}.$$

Ни одна буква из этих алфавитов не является буквой в другом из них.

Пусть слово P в алфавите A будет

$$001011000110.$$

Оно состоит из букв 00, 10, 11, 00, 01, 10. Очевидно, $L(P)=6$.

Рассмотрим, кроме того, слово Q в алфавите B

$$001011000110.$$

Оно состоит из букв 001, 011, 000, 110 и имеет длину $L(Q)=4$.

Слова P и Q принадлежат неравным алфавитам, но между собой неотличимы. Они предметно-тождественны.

5. Операции над словами. Пусть M — множество, элементами которого являются упорядоченные наборы, состоящие каждый из n слов в алфавите A, а N — множество слов в алфавите B.

Если каждому набору слов $P_1, P_2, \ldots, P_n$, принадлежащему множеству M, поставлено в соответствие одно и только одно слово Q из N, то говорят, что задана *n-местная операция* $\Omega^{(n)}_{A,B}$, преобразующая упорядоченные наборы слов в алфавите A в слова в алфавите B (иногда говорят «операция между алфавитами A и B»). При этом пишут

$$Q=\Omega^{(n)}_{A,B}(P_1,\ P_2,\ldots,\ P_n).$$

В частном случае алфавит B может быть равен алфавиту A. При этом вместо символа $\Omega^{(n)}_{A,A}$ будем применять символ $\Omega^{(n)}_{A}$ и операцию назовем операцией *внутри* алфавита A.

Множество M называют *областью определения* операции, а слово Q — одним из ее *значений*. Если каждое слово, принадлежащее N, является значением операции, то N называют *областью изменения* операции. При $n=1$ упорядоченные наборы слов сводятся к отдельным словам и M оказывается множеством слов в алфавите A. Операция $\Omega^{(1)}_{A,B}$ называется *одноместной*. Вместо символа $\Omega^{(1)}_{A,B}$ будем употреблять символ $\Omega_{A,B}$.

Одноместная операция внутри алфавита A, имеющая в качестве области задания множество всех возможных слов в алфавите A и удовлетворяющая условию

$$\Omega_A(P)=P,$$

называется *тождественной операцией*.

6. Транспонирование операций. Операции внутри алфавита A можно заменить операциями внутри другого (даже неравного и неравнозначного ему) алфавита B.

Пусть $\Omega^{(n)}_A$ — операция внутри алфавита A с областью определения M и областью изменения N.

Из попарно-различных слов в алфавите B, принимая их за буквы, составим новый алфавит C, равнозначный алфавиту A.

Каждое слово P в алфавите A, входящее в упорядоченный набор, принадлежащий M, или принадлежащее N, заменим равнозначным ему словом P' в алфавите C. Тогда найдется слово P'' в алфавите B, предметно-тождественное слову P', которое будет, следовательно, взаимно однозначно соответствовать слову P. Таким образом, построим множество M'', элементами которого являются упорядоченные наборы слов в алфавите B, и множество N'' слов в алфавите B. Определим операцию $\Omega^{(n)}_B$ так, чтобы было

$$Q''=\Omega^{(n)}_B(P''_1,\ P''_2,\ldots,\ P''_n),$$

если словам $P''_1,\ P''_2,\ldots,\ P''_n$, Q'' взаимно однозначно соответствуют слова $P_1,\ P_2,\ldots,\ P_n$, Q, для которых

$$Q=\Omega^{(n)}_A(P_1,\ P_2,\ldots,\ P_n).$$

Для определения значения операции $\Omega^{(n)}_A(P_1,\ P_2,\ldots,\ P_n)$ можно заменить слова $P_1,\ P_2,\ldots,\ P_n$ в алфавите A словами P'_i в алфавите C, а следовательно, предметно-тождественными

им словами P_i'' в алфавите B, затем по формуле

$$Q'' = \Omega_B^{(n)}(P_1'', P_2'', \ldots, P_n'')$$

получить слово Q'' в алфавите B, а следовательно, предметно-тождественное ему слово Q' в алфавите C, и наконец заменить Q' равнозначным ему словом Q в алфавите A. Полученное таким путем слово Q удовлетворяет условию

$$Q = \Omega_A^{(n)}(P_1, P_2, \ldots, P_n).$$

Замена операций, определенных в алфавите A, операциями в алфавите B с помощью введения вспомогательного алфавита C, буквами которого являются слова в B, называется *транспонированием операций.*

Пример 1. 32. Транспонирование операций широко применяется в электронных цифровых машинах. Так, десятичные числа, то есть слова в алфавите

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\},$$

заменяются равнозначными им словами в алфавите D, буквами которого являются группы отверстий перфокарты. Замена букв (десятичных цифр) алфавита A равнозначными им буквами (группами отверстий) алфавита D производится механическим путем: при нажиме клавиши, помеченной некоторой цифрой, производится пробивка группы отверстий, отвечающей этой цифре.

Слова в алфавите D в устройстве ввода машины заменяются словами в алфавите C, буквами которого служат группы электрических импульсов. Эти буквы в свою очередь предметно-тождественны словам в алфавите B, буквами в котором служат, например, одиночные импульсы высокого и низкого напряжения.

Операции электронной цифровой машиной производятся внутри алфавита B. Получаемые результаты предметно-тождественны словам в алфавите C и могут быть преобразованы в устройствах вывода в равнозначные им слова в алфавите D. Последние на печатающем устройстве преобразуются в равнозначные им слова в алфавите A. В конечном итоге разультаты машинного счета получаются в виде чисел, записанных в десятичной системе счисления.

7. Уточнение понятия алгорифма. Алгорифм выполнения. Абстрактная машина. Пусть алфавит C представляет собой объединение алфавитов A и B (которые, в частности, могут быть равны).

Слово Q в алфавите B, которому поставлена в однозначное соответствие определенная операция Ω_C, называется *элементарным предписанием* в алфавите C. Область опреде-

ления операции $\mathfrak{Q}_C$ будем называть также областью определения элементарного предписания Q.

Если слово P принадлежит области определения элементарного предписания Q и если

$$P' = \mathfrak{Q}_C(P),$$

то будем говорить, что элементарное предписание Q *применимо* к слову P и *перерабатывает* его в P'.

Два элементарных предписания в алфавите C называются *различными*, если они не являются равными словами в алфавите C.

Пусть E — конечное или счетное множество попарно различных элементарных предписаний, среди которых присутствует элементарное предписание Q', соответствующее тождественной операции в алфавите C.

Предположим, что G — некоторое непустое множество слов P в алфавите A, что R — слово в алфавите B и что Q_0 — одно из элементарных предписаний (не равное Q'), содержащихся в E.

Допустим, что задан закон композиции $\sum$, позволяющий из слов Q_0, R и любого слова P из G образовать композицию W_0, которая может быть записана как объединение слов $Q_0\ U_0\ V_0$ (очевидно, характеристика простого вхождения слова Q_0 в W_0 содержится в законе композиции $\sum$).

Пусть существует такое натуральное число

$$n = n(P),$$

что для $i = 1, 2, \ldots, n$ операция $\mathfrak{Q}_{i-1}$, названием которой является элементарное предписание Q_{i-1}, осуществляет преобразование слов

$$\mathfrak{Q}_{i-1}(W_{i-1}) = \mathfrak{Q}_{i-1}(Q_{i-1}U_{i-1}V_{i-1}) = Q_iU_iV_i = W_i,$$

причем $Q_1, Q_2, \ldots, Q_{n-1}$ отличны от Q', а Q_n равно Q'. Пусть, кроме того, заранее известна характеристика σ вхождения некоторого слова в алфавите A (обозначим это слово T) в W_n.

Если выполнены все перечисленные условия, то говорят, что R является *алгорифмом* над алфавитом A, перерабатывающим P в T; P называют *исходным данным*, а T — *искомым результатом*, G называют *областью задания* алго-

рифма R, а множество элементарных предписаний E — *основанием* алгорифма.

Простота операций на каждом шагу процесса выполнения алгорифма R заключается в том, что они принадлежат некоторому заранее заданному множеству операций.

Процесс преобразования слов W_{i-1} в слова W_i ($i=1, 2, \ldots, n$, где $n=n(P)$), называется *процессом выполнения* алгорифма R с основанием E.

Алгорифм, которому *подчинен* процесс выполнения алгорифма R с основанием E, называется *алгорифмом выполнения* при основании E или, коротко, в-*алгорифмом* при основании E.

Алгорифмы с основанием E, имеющие один и тот же в-алгорифм, образуют семейство родственных алгорифмов.

Совокупность в-алгорифма F и основания E называется *абстрактной машиной* и обозначается символом (A, B, E, F).

Любой алгорифм из семейства родственных алгорифмов, имеющих в-алгорифм F, называется *выполнимым* абстрактной машиной (A, B, E, F).

Пример 1.33. Программа, составленная для программно-управляемой машины, является алгорифмом. Элементарными предписаниями являются отдельные команды. Характеристика вхождения σ указывает команду, с которой должно начинаться выполнение программы. Закон композиции Σ характеризует размещение в памяти машины программы и исходных данных (то, что называют *распределением памяти*). Характеристика вхождения σ^* указывает то место памяти, в котором будет получен искомый результат. Под словом W_0 понимается начальное содержимое памяти машины, Q_0—содержимое регистра команд в момент начала выполнения машиной программы. Совокупность всевозможных программ представляет семейство родственных алгорифмов, множество всех возможных команд — основание E. Алгорифм, которому подчинен процесс выполнения машиной программ, является в-алгорифмом. Совокупность этого в-алгорифма и основания E представляет абстрактную машину.

Если R_1 и R_2 — алгорифмы над алфавитом A, имеющие общую область задания G и перерабатывающие каждое допустимое для них исходное данное P в один и тот же искомый результат T, то говорят, что алгорифмы R_1 и R_2 *равносильны* и пишут $R_1=R_2$.

Пусть R — алгорифм над алфавитом D, перерабатывающий каждое допустимое исходное данное P в некоторое T_p. Пусть, кроме того, A — алфавит, вообще говоря, не равный алфавиту D. Принимая некоторый набор слов в алфавите A за

буквы алфавита C, равнозначного алфавиту D, построим для каждого P слово P' в алфавите C. Это слово будет предметно-тождественно некоторому слову P' в алфавите A. Предположим, что над алфавитом A существует алгорифм R'', перерабатывающий каждое слово P'' в слово $T''_{P''}$, которое предметно-тождественно некоторому слову $T'_{P'}$ в алфавите C, причем $T'_{P'}$ равнозначно слову T_P. В этом случае говорят, что *алгорифм R'' над алфавитом A равнозначен алгорифму R над алфавитом D при элементарном кодировании исходных данных с помощью C.*

Абстрактная машина (A, B, E, F) называется *универсальной,* если, каковы бы ни были алфавит D и алгорифм R над этим алфавитом, существует алгорифм R'' над алфавитом A, равнозначный алгорифму R при некотором элементарном кодировании, выполнимый абстрактной машиной (A, B, E, F). В противном случае абстрактная машина (A, B, E, F) называется *неуниверсальной.*

8. Локально заданные операции над словами. В некоторых случаях операции над словами, имеющие бесконечную область задания, могут быть заданы эффективно с помощью приема, называемого *локальным заданием операций.* Сущность этого приема заключается в том, что задаются некоторая операция с конечной областью задания и некоторое правило применения ее к словам, являющимся вхождениями в другие слова (представляющие собой объект применения операции с бесконечной областью задания). Упомянутая операция с конечной областью задания называется в этом случае *локальной.*

Локальное задание операций осуществляется следующим образом. Пусть

$$\Omega(P_1, P_2, \ldots, P_n) = Q$$

является локальной операцией. Выполнение операции $\Omega^*(P)$ с бесконечной областью задания определяется как применение локальной операции Ω к упорядоченному набору слов, который получится, если определить слова, являющиеся соответственно простыми вхождениями в слово P, имеющими соответственно характеристики $\sigma_1, \sigma_2, \ldots, \sigma_n$. Результат применения операции Ω^* к слову P получится, если простое вхождение в это слово, равное по длине слову Q и начинающееся буквой,

имеющей номер ν, заменить словом Q (результатом локальной операции). Областью задания операции Ω^* будет, как легко видеть, бесконечное множество G слов P, содержащих вхождения наборов слов, для которых определена операция Ω и достаточно длинных для того, чтобы описанное правило было выполнимо. Действительно, если существует хотя бы одно $P = P_0$, для которого операция Ω^* может быть выполнена, то эта операция может быть выполнена и для любого слова, которое получится из P_0 путем приписывания к нему (вслед за его последней буквой) любой последовательности букв.

Пример 1.34. Предположим, что C — это алфавит

$$\{ а, б, в \}.$$

Пусть локальная операция

$$\Omega (P_1, P_2) = Q$$

задана табл. 1.19 с двумя входами.

Таблица 1.19. Операция $\Omega (P_1, P_2) = Q$

P_1 \ P_2	*а*	*б*	*в*
аа	*аба*	*ааа*	*ббб*
аб	*баа*	*ваб*	*ввб*
ав	*ваа*	*вав*	*ввв*

Пусть, кроме того, характеристиками простых вхождений в слова P, над которыми выполняется локально заданная операция $\Omega^* (P)$, являются

$$\sigma_1 = 1{,}2, \qquad \sigma_2 = 4,$$

и пусть $\nu = 3$. Легко видеть, что область задания G операции $\Omega^* (P)$ состоит из слов, начинающихся с буквы *а* и имеющих длину не меньшую чем 5.

Например,

$$\Omega^* (абаваа) = аббвба,$$

так как $P_1 = аб$ (двухбуквенное простое вхождение, определяемое характеристикой σ_1), а $P_2 = в$ (однобуквенное вхождение, определяемое характеристикой σ_2). При этом по таблице получается $Q = ббб$. Вставляя Q в P вместо трехбуквенного простого вхождения, начинающегося с буквы номер $\nu = 3$, получаем *аббвба*.

Пользуясь несколькими локальными операциями и выполняя соответствующие правила в определенной последовательности,

каждое применительно к результату, полученному после выполнения предыдущего, можно получить более сложные локально заданные операции.

Очевидно, элементарное предписание, соответствующее локально заданной операции, можно строить как совокупность слова, соответствующего локальной операции (или последовательности локальных операций), и слов, означающих характеристики вышеупомянутых вхождений.

К числу локально заданных операций относятся также операции следующих двух видов:

а) операция, заключающаяся в том, что некоторое простое вхождение в исходное для операции слово P зачеркивается (то есть заменяется вхождением пустого слова) и в качестве результата операции принимается полученное более короткое слово;

б) к слову P (после его конца) приписывается некоторое заданное слово и результатом операции считается полученное более длинное слово. При операции второго вида пустое слово, вхождением которого оканчивается P, заменяется заданным словом.

9. Размеченная память. Пусть C — некоторый алфавит. Под *памятью* будем понимать переменное слово π заданной длины N. Конкретное слово, которое получится, если в качестве букв упомянутого переменного слова π взять определенные буквы в алфавите C, будем называть *памятью, находящейся в определенном состоянии.*

Если при определенном состоянии памяти π некоторое вхождение в π равно слову P, то говорят, что память хранит слово P (независимо от того, каковы буквы слова π, не принадлежащие упомянутому вхождению).

Пусть задана совокупность характеристик простых вхождений в слово π такая, что ее можно рассматривать как закон композиции слова π из упомянутых простых вхождений. В этом случае говорят, что память π состоит из *ячеек.* Под ячейками понимают части памяти, соответствующие характеристикам вхождений. Характеристики вхождений называют *названиями* ячеек. Слово, одинаковое с ячейкой при конкретном состоянии памяти, называют *хранящимся* в ячейке. Название ячейки называют *адресом* конкретного слова, хранящегося в ячейке. Память в описанном случае называют *размеченной.*

Так как ячейки размеченной памяти являются простыми вхождениями в нее, то характеристики этих вхождений будут вполне определены, если указать первое число и количество чисел в каждой из них. Таким образом, в качестве названия ячейки можно применить пару чисел: номер ее первой буквы в слове π и ее длину.

Если ячейки размеченной памяти попарно не пересекаются, то характеристики вхождения каждой из них будут взаимно однозначно соответствовать номерам, которые получатся, если характеристики вхождений перенумеровать в порядке возрастания их первых чисел. Это позволяет в описанном случае в качестве названий ячеек применять номера их названий (которые можно считать их номерами).

Наконец, представляет интерес случай, в котором совокупность всех ячеек памяти можно разбить на две частные совокупности, первая из которых содержит непересекающиеся между собой ячейки, покрывающие всю память π, а вторая содержит попарно непересекающиеся ячейки, являющиеся частями ячеек первой из них. В этом случае можно перенумеровать ячейки первой частной совокупности и номера считать их названиями, а ячейкам второй частной совокупности либо приписать номера внутри ячеек первой частной совокупности, либо присвоить названия, отличающиеся от названий первой частной совокупности.

Заметим, что числа, применяемые в качестве названий ячеек, можно представить в виде слов в алфавите C, если некоторые буквы этого алфавита (в данном их применении) рассматривать как цифры некоторой позиционной системы счисления.

10. Абстрактные машины с размеченной памятью. Пусть алфавит C является суммой алфавитов A и B.

Ниже описан важный частный случай абстрактной машины, называемой *абстрактной машиной с размеченной памятью* π.

Будем считать, что для всякого алгорифма R, выполнимого на описываемой абстрактной машине, исходными данными являются слова P вида $P'\pi^*$, где π^* — конкретное состояние памяти π. Далее, будем считать, что всякий закон композиции Σ (см. определение алгорифма в п. 7) заключается в следующем:

а) слово R рассматривается как объединение двух слов $R'R''$;

б) из R' и P' образуется некоторая определяемая законом $\sum$ композиция K';

в) из слов π и Q_0 (Q_0 — элементарное предписание) строится слово $S = Q_0\pi_0'$ путем замены в π его начала, равного по длине слову Q_0, словом Q_0; таким образом, S представляет собой некоторое конкретное состояние памяти π;

г) из слов S, K', R'' строится объединение

$$W_0 = SK'R''.$$

Если положить

$$U_0 = \pi_0' K'R'' \quad \text{и} \quad V_0 = \Lambda$$

(где Λ — пустое слово), то можно считать, что

$$W_0 = Q_0 U_0 V_0.$$

Предположим, далее, что элементарным предписаниям из E соответствуют локально заданные операции, из которых одни являются операциями над памятью π, а другие выходят за ее пределы, но при этом либо представляют собой внесение начального куска слова, примыкающего к π, в одну из ячеек π и вычеркивание этого начального куска на его прежнем месте, либо представляют собой приписывание к концу слова W содержимого одной из ячеек памяти π. Одному или нескольким элементарным предписаниям из E соответствуют тождественные операции.

При перечисленных условиях выполнение алгорифма R будет состоять из последовательных актов, результатами которых будут слова:

$$\begin{aligned} W_1 &= Q_1\ U_1\ V_1, \\ W_2 &= Q_2\ U_2\ V_2, \\ &\cdots\cdots\cdots \\ W_n &= Q_n\ U_n\ V_n. \end{aligned}$$

Результатом выполнения алгорифма R будем считать слово V_n. Будем, кроме того, предполагать, что для любых слов P, отличающихся друг от друга только составной частью π, результаты (при заданном R) одинаковы и что слово U_n имеет структуру $\pi_n' R'''$, где слово π_n' таково, что вместе с Q_n образует слово $Q_n\pi_n'$, являющееся конкретным состоянием памяти π, а R''' является либо пустым, либо вхождением, которым заканчивается слово R''.

Для описанной нами абстрактной машины с размеченной памятью характерно, что все выполняемые ею операции являются локально заданными. Память этой машины имеет конечную длину.

Остановимся подробнее на элементарных предписаниях, выполняющихся при функционировании абстрактной машины с размеченной памятью.

Предположим, что $\Omega_C^{(k)}$ — некоторая k-местная операция. Зададим $k+1$ определенных ячеек размеченной памяти.

Пусть

$$a_1,\ a_2,\ \ldots,\ a_k,\ a_{k+1}$$

— названия этих ячеек (среди них могут быть и одинаковые). Примем содержимые ячеек, имеющих названия $a_1, a_2 \ldots, a_k$, за исходные данные операции $\Omega_C^{(k)}$, а ее результат запишем в ячейку с названием a_{k+1}.

Предположим, что задан некоторый основной набор H локальных операций.

Пусть символы $x_{i,\,j}$ $(i=1,\ 2,\ \ldots;\ j=1,\ 2,\ \ldots)$ обозначают названия ячеек, а символы Ω_i локальные k_j-местные операции из основного набора. Составим последовательность формул

$$\left.\begin{array}{l} x_{1,\,k_1+1} := \Omega_1 (x_{1,\,1},\ x_{1,\,2},\ \ldots,\ x_{1,\,k_1}), \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_{i,\,k_i+1} := \Omega_i (x_{i,\,1},\ x_{i,\,2},\ \ldots,\ x_{i,\,k_i}), \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_{n,\,k_n+1} := \Omega_n (x_{n,\,1},\ x_{n,\,2},\ \ldots,\ x_{n,\,k_n}), \end{array}\right\} \qquad (1.18)$$

Символ $:=$ называется знаком присваивания и означает, что величине, стоящей слева от него, придается значение, полученное при вычислении выражения, стоящего справа. Вычисления по каждой формуле из (1.18) выполняются после того, как окончены вычисления по предыдущей. В последовательности формул (1.18) среди символов $\Omega_1, \Omega_2, \ldots, \Omega_n$ могут в частном случае повторяться одинаковые. Среди символов $x_{i,\,j}$ также могут быть одинаковые.

Последовательность формул (1.18) после замены в ней символов $x_{i,\,j}$ конкретными названиями ячеек определяет некоторую последовательность локально заданных операций. Некоторые из символов $x_{i,\,j}$ в последовательности формул

(1.18) могут обозначать *произвольные* ячейки, другие — вполне *определенные* ячейки.

Те символы $x_{i,j}$, которые обозначают произвольные ячейки, мы будем называть *переменными*.

Последовательность формул (1.18) будем называть *типом команды*. Количество попарно-различных переменных символов $x_{i,j}$ присутствующих в типе команды, будем называть *адресностью* типа команды, а сами переменные символы $x_{i,j}$ — *адресными* символами. Если адресность типа команды равна ν, то тип команды называется ν-адресным.

Пример 1.34. Пусть алфавит A состоит из двух букв 0,1. Слова в этом алфавите являются двоичными числами. Если среди основного набора локальных операций содержатся сложение (+), определение знака (sign) и перенос (обозначается знаком «=»), то тип команды может иметь вид

$$\left.\begin{aligned} s &:= a + b, \\ \omega &:= \operatorname{sign} s, \\ c &:= s. \end{aligned}\right\} \tag{1.19}$$

Здесь a, b, c, s, ω — символы, обозначающие номера ячеек. Предположим, что s и ω обозначают всегда одни и те же ячейки, а a, b, c обозначают произвольные ячейки. Тогда тип команды (1.19) является трехадресным, причем a, b, c — адресные символы.

Тип команды (1.19) при замене символов a, b, c названиями ячеек будет определять следующее преобразование слова, записанного в размеченной памяти:

1) содержимые ячеек a и b (являющиеся вхождениями в W) складываются и результат записывается в ячейку s;

2) определяется знак суммы, находящейся в s (0 означает плюс, 1 означает минус), и записывается в ячейку ω;

3) содержимое ячейки s (без его стирания в ячейке s) записывается в ячейку c.

Предположим, что задано некоторое конечное множество H' типов команд. Будем считать, что H' содержит также тип команды, определяющий тождественное преобразование слова, записанного в размеченной памяти. Поставим во взаимно однозначное соответствие каждому типу команды некоторое слово $b_t (t = 1, 2, \ldots)$ в алфавите B. Это слово будем называть *кодом операций*. Пусть, кроме того, задан групповой закон композиции $\sum^*$, позволяющий для любого типа команды из H' после замены его адресных символов названиями ячеек получить композицию этих названий и кода операций. Такую композицию будем считать элементарным предписанием и назы-

вать *командой*. Заметим, что закон композиции $\sum^*$ не должен приводить к одинаковым командам для различных между собой кодов операций или для одного и того же кода операций, но при различных заменах адресных символов номерами ячеек.

Вхождение в команду названия ячейки называется ее адресом. Таким образом, каждая команда, имеющая ν-адресный тип, имеет ν адресов.

11. Реальные программно-управляемые машины как физические модели абстрактных машин с размеченной памятью. Существующие программно-управляемые машины можно считать физическими моделями абстрактных машин с размеченной памятью. При этом размеченной памяти соответствует совокупность всех запоминающих устройств машины.

Ее арифметическое устройство и устройство управления вместе взятые соответствуют в-алгорифму и представляют собой автомат, функционирование которого сводится к процессу, определяемому в-алгорифмом. Упомянутые в определении абстрактной машины с размеченной памятью $K'R''$, Q_0, V_n для конкретной программно-управляемой машины означают соответственно программу и исходные данные, подготовленные для ввода в машину, команду, засылаемую в регистр команд при нажатии кнопки «начальный пуск» и результат, выдаваемый устройством вывода. Во многих существующих программно-управляемых машинах алфавиты A и B между собой равны и каждый из них равнозначен алфавиту $\{0,1\}$.

Внешние устройства программно-управляемой машины и ее устройства ввода и вывода являются приборами для элементарной перекодировки слов. В некоторых случаях устройства ввода и вывода обеспечивают также согласование малой скорости поступления слов с большой скоростью действия программно-управляемой машины.

Мы уже знаем, что по способу представления чисел программно-управляемые машины делятся на машины с фиксированной запятой и машины с плавающей запятой.

Кроме такого деления, существует условное деление машин на универсальные и неуниверсальные.

Программно-управляемую машину можно называть *универсальной*, если моделируемая ею абстрактная машина с размеченной памятью является универсальной.

Иногда машины делят на *специализированные* и *неспециализированные. Специализированной* называется машина, созданная для решения некоторого конечного множества задач, то есть машина, реализующая алгорифмы, равнозначные алгорифмам решения этих задач, оптимальным образом (то есть при наиболее выгодных с практической точки зрения количестве оборудования машины, ее стоимости, времени, расходуемом на выполнение этих алгорифмов, и т. п.). Такие машины могут быть как универсальными, так и не универсальными. Машины, не приспособленные к определенным задачам, созданные с расчетом применения их для решения задач, класс которых заранее не установлен, называют *неспециализированными.*

Общепринята также классификация машин по адресности типов их команд. Так, существуют одноадресные, двухадресные, трехадресные, четырехадресные и даже пятиадресные машины. Если машина называется ν-адресной, это отнюдь не значит, что все типы ее команд ν-адресные. Обычно ν-адресные типы составляют большинство ее типов команд.

Закон композиции, с помощью которого из кода операций и номеров (названий) ячеек конструируется команда, как правило, бывает очень простым. Композиция сводится к объединению этих слов в определенном порядке, строго установленном для каждой машины.

Например, если тип команды имеет вид

$$s := x + y,$$
$$\omega := \operatorname{sign} s,$$
$$z := s,$$

где x, y, z — адресные символы, то команда, в которой x, y, z заменены соответственно номерами ячеек 0321, 6721 и 0027, может иметь вид

0321 6721 0027 01

или (для другой машины)

01 0321 6721 0027.

Номера ячеек и код операций представлены здесь в восьмеричной системе счисления, как это делают для многих машин при записи команд на бланках.

Если в командах, применяемых в данной машине, присутствует адрес, являющийся номером ячейки, хранящей очередную команду, то машину называют *машиной со свободным порядком выполнения команд.* Если же такой адрес в командах машины отсутствует (т. е. получение его предусмотрено в в-алгорифме) то машину называют *машиной с заданным порядком выполнения команд* *).

Алгорифмы, выполняемые программно-управляемыми машинами, обычно называют *программами.* Заметим, что название *программно-управляемая машина* не соответствует существу этой машины. Правильнее было бы название *машина, выполняющая программы.*

Литература к главе 1: [2], [15], [19], [20], [35], [36], [37], [51], [57], [58], [69], [70].

---

*) Нередко свободный порядок выполнения команд называют *принудительным* (что неправильно), а заданный порядок — *естественным.* Последнее название происходит, по-видимому, от того, что при заданном порядке чаще всего команды выполняются в порядке номеров хранящих их ячеек.

ГЛАВА 2

СИСТЕМА ОПЕРАЦИЙ И СИСТЕМА КОМАНД ПРОГРАММНО-УПРАВЛЯЕМОЙ МАШИНЫ

§ 1. Система операций программно-управляемой машины и их классификация

Операции, выполняемые цифровой программно-управляемой машиной в процессе ее функционирования, являются локально заданными. *Машинными операциями* принято называть те локальные операции, с помощью которых осуществлено определение общих машинных операций.

Совокупность всех машинных операций, предусмотренных в программно-управляемой машине, называется ее *системой операций*. Следует четко разграничить понятие команды и понятие операции. *Команда* представляет собой элементарное предписание, предусматривающее выполнение некоторой группы операций.

При классификации машинных операций необходимо учитывать, что их характер и роль существенно зависят от того, где находятся исходные для них данные и главным образом куда записываются результаты. Связано это с тем, что в программно-управляемой машине имеется ряд запоминающих устройств, играющих в ее работе принципиально различные роли.

1. Классификация запоминающих устройств машины. Оперативная память, внешние запоминающие устройства, запоминающие ячейки арифметического устройства и перфокарты (перфоленты) предназначены для хранения кодов, представляющих числа и команды (обозначим эти устройства буквой A). *Регистр выполняемой команды* — коротко, «регистр команд» (обозначим его через B) — служит для приема команды,

подлежащей выполнению и, следовательно, определяющей дальнейшие действия машины. В *регистре номера очередной команды* — коротко, «счетчике команд» (обозначим его через C) — в определенный момент выполнения команды должен быть записан номер ячейки, содержащей очередную команду, подлежащую переносу в регистр команд, т. е. команды, к которой надлежит перейти после выполнения данной. (В машинах со свободным порядком выполнения команд счетчик команд обычно бывает частью регистра команд.) Одноразрядные запоминающие устройства (обозначим их совокупность через D) предназначаются для хранения специальных сигналов (например, сигнала, называемого «сигналом ω»), используемых при автоматическом изменении порядка выполнения команд. В ряде машин присутствуют так называемые *индексные регистры* (обозначим их через F), содержимое которых может прибавляться к содержимому регистра команд (или отниматься от него) и тем самым изменять выполняемую команду. В таких машинах обычно для каждого индексного регистра имеется одноразрядное запоминающее устройство, предназначенное для специального сигнала, возникающего, если содержимое индексного регистра удовлетворяет некоторому условию, например делается равным нулю. Эти одноразрядные запоминающие устройства входят в состав D.

В более сложных случаях в машине может присутствовать еще ряд других специальных устройств, на которых мы здесь останавливаться не будем. Устройства A, B, C и D в том или ином виде имеются во всех существующих в настоящее время программно-управляемых машинах. Устройства F присутствуют не во всех отечественных машинах. В конкретных машинах устройства B, C, D и F имеют различное конструктивное решение и носят различные названия.

2. Классификация операций машины. К первому классу машинных операций отнесем все операции, исходные данные для которых содержатся в A, и результаты записываются в A (класс операций «$A \longrightarrow A$»). В этот класс входят *арифметические* операции, так называемые *неарифметические* операции, *переносы* и другие операции над числами, кодами и командами.

Ко второму классу (класс «$\longrightarrow D$») относятся операции, результаты выполнения которых записываются в D. Эти опе-

рации называются *логическими*, так как представляют собой проверки наличия различных свойств у чисел, кодов или команд.

Третьему классу (класс «$\longrightarrow C$») принадлежат операции, результаты которых записываются в C. К этому классу относятся так называемые *переходы*. С помощью операций класса $\longrightarrow C$ машина осуществляет переход от выполнения одной команды к выполнению другой.

В четвертый класс (класс «$\longrightarrow B$») объединим операции, результаты которых записываются в B. В этот класс входят операции, с помощью которых в машине производится так называемая выборка команды из оперативной памяти и различные модификации команд в регистре команд.

Все остальные машинные операции, кроме операции останова, отнесем к пятому классу. В частности, сюда войдут операции, результаты которых записываются в F.

Операцию останова (операцию «стоп»), заключающуюся в тождественном преобразовании содержимого всех запоминающих устройств машины, выделим в самостоятельный шестой класс. Технически эта операция осуществляется путем прекращения функционирования машины.

В некоторых программно-управляемых машинах могут быть предусмотрены и другие виды операций, которые здесь не приведены.

§ 2. Кодирование чисел для их изображения в устройствах машины

В существующих машинах каждая запоминающая ячейка состоит из разрядов, в которых могут храниться только цифры. Если рабочая система счисления машины имеет симметричную базу, то числа могут быть записаны в ячейках без применения каких-либо специальных приемов. Иначе обстоит дело, если рабочая система счисления имеет неотрицательную базу. В этом случае в ячейках, кроме цифр, необходимо изображать также знаки чисел (плюс или минус). Знаки изображают в виде наборов цифр, отводя для них вполне определенные разряды ячеек. Таким образом, в машинах, рабочая система счисления которых имеет неотрицательную базу, числа представляются в виде некоторых цифровых кодов.

В дальнейшем, за исключением специально оговариваемых случаев, речь идет о машинах, для которых рабочей системой счисления является двоичная.

1. Прямой код. Пусть x — двоичное число, имеющее n разрядов в целой части и ν разрядов в дробной части:

$$x = sa_{n-1}a_{n-2} \ldots a_0, a_{-1} \ldots a_{-\nu}, \tag{2.1}$$

где s — знак числа, a_i — двоичные цифры.

Прямым кодом $[x]^k_{\text{пр}}$ числа x называется целое $(n+\nu+k)$-разрядное число ($k \geqslant 1$), определяемое формулой

$$[x]^k_{\text{пр}} = \begin{cases} \underbrace{00 \ldots 0}_{k \text{ нулей}}\, a_{n-1}a_{n-2} \ldots a_0\, a_{-1} \ldots a_{-\nu} \text{ при } x \geqslant 0, \\ \underbrace{11 \ldots 1}_{k \text{ единиц}}\, a_{n-1}a_{n-2} \ldots a_0\, a_{-1} \ldots a_{-\nu} \text{ при } x \leqslant 0. \end{cases} \tag{2.2}$$

Старшие k разрядов прямого кода числа называют *знаковыми.*

Пример 2.1. Как видно из определения, число нуль имеет два прямых кода, называемых соответственно прямыми кодами положительного и отрицательного нулей:

$$[0]^k_{\text{пр}} = [+0]^k_{\text{пр}} = \underbrace{00 \ldots 0}_{n+\nu+k \text{ нулей}},$$

$$[0]^k_{\text{пр}} = [-0]^k_{\text{пр}} = \underbrace{11 \ldots 1}_{k \text{ единиц}}\, \underbrace{0 \ldots 0}_{n+\nu \text{ нулей}}.$$

При каждой системе кодирования величина k имеет фиксированное значение. Для того, чтобы по прямому коду определить изображаемое им число, необходимо знать, чему равны k и ν (значение n можно при этом определить по числу разрядов прямого кода).

На практике чаще всего применяется прямой код с одним знаковым разрядом ($k = 1$). В обозначении $[x]^k_{\text{пр}}$ при $k = 1$ верхний значок будем опускать.

Пример 2.2. Пусть $k = 1$, $n = 4$, $\nu = 5$. Тогда прямые коды чисел будут десятиразрядными числами:

$$[0001,\ 10101]_{\text{пр}} = 0000110101,$$
$$[-0001,\ 10101]_{\text{пр}} = 1000110101.$$

Пример 2.3. При $k=1$, $n=0$, $\nu=9$ прямые коды чисел также будут десятиразрядными целыми числами:

$$[0{,}001001110]_{\text{пр}}=0001001110,$$
$$[-0{,}001001110]_{\text{пр}}=1001001110,$$
$$[-0{,}000110101]_{\text{пр}}=1000110101.$$

Сравнивая последние равенства предыдущего и настоящего примеров, видим, что прямые коды не равных между собой чисел могут быть равны. Это исключается, если совместно рассматриваются только числа, имеющие одинаковые n, ν и k.

2. Сумматоры. В программно-управляемых машинах, рабочая система счисления которых является двоичной, при сложении и вычитании чисел применяются устройства, называемые *двоичными сумматорами*. Пусть α и β — два положительных целых m-разрядных двоичных числа:

$$\alpha=a_{m-1}a_{m-2}\ \ldots\ a_1a_0,$$
$$\beta=b_{m-1}b_{m-2}\ \ldots\ b_1b_0,$$

где a_i и b_i — двоичные цифры ($i=0,\ 1,\ 2,\ \ldots,\ m-1$). В двоичных m-разрядных сумматорах над одноименными разрядами чисел α и β для $i=0,\ 1,\ \ldots,\ m-1$ выполняются операции

$$c_i=a_i+b_i+z_i-10z_{i+1},$$

где

$$z_{i+1}=\begin{cases}0 & \text{при } a_i+b_i+z_i<10,\\ 1 & \text{при } a_i+b_i+z_i\geqslant 10.\end{cases}\qquad(2.3)$$

Здесь z_i — перенос из $(i-1)$-го разряда, в i-й, который может быть равен нулю или единице, c_i — цифра результата, 10 — означает число 2.

В формулах (2.3) не определено значение величины z_0. В зависимости от способа задания этой величины существует два вида двоичных m-разрядных сумматоров:

сумматоры с циклическим переносом из старшего разряда в младший (рис. 2.1), в которых

$$z_0=z_m;$$

сумматоры без переноса из старшего разряда (рис. 2.2), в которых

$$z_0=0,$$

а величина z_m никак не используется.

Очевидно, результат, получаемый с помощью m-разрядных двоичных сумматоров, является арифметической суммой чисел α и β лишь в том случае, если оказывается, что $z_m = 0$, то есть если $\alpha + \beta < 10^m$ (10 — два).

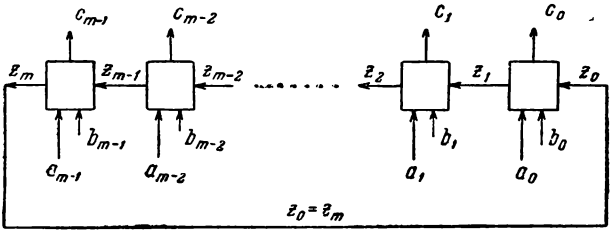


Рис. 2.1. Схема действия сумматора с циклическим переносом из старшего разряда в младший.

Операция, выполняемая сумматором с циклическим переносом (обозначим ее знаком $\bigoplus_m$), определяется формулой

$$\alpha \bigoplus_m \beta = \begin{cases} \alpha + \beta & \text{при } \alpha + \beta < 10^m, \\ \alpha + \beta - 10^m + 1 & \text{при } \alpha + \beta \geqslant 10^m. \end{cases} \tag{2.4}$$

В терминах теории чисел эта операция называется сложением по модулю $10^m - 1$ и обозначается $(\alpha + \beta) \bmod (10^m - 1)$ *).

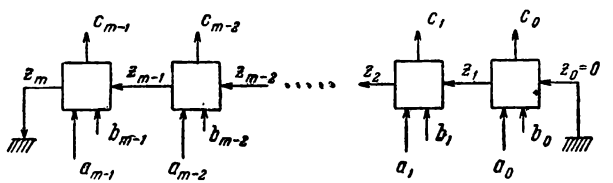


Рис. 2.2. Схема действия сумматора без переноса из старшего разряда.

Сумматор без переноса из старшего разряда выполняет операцию (обозначим ее знаком $\overline{|+|}_m$), определяемую формулой

$$\alpha \overline{|+|}_m \beta = \begin{cases} \alpha + \beta & \text{при } \alpha + \beta < 10^m, \\ \alpha + \beta - 10^m & \text{при } \alpha + \beta \geqslant 10^m. \end{cases} \tag{2.5}$$

*) О сложении по модулю см. И. М. Виноградов, Основы теории чисел, изд. 7, «Наука», 1965.

Эта операция в теории чисел называется сложением по модулю 10^m и обозначается $(\alpha+\beta) \bmod 10^m$.

Сложение и вычитание положительных и отрицательных чисел можно производить с помощью m-разрядных двоичных сумматоров лишь после замены чисел специальными кодами. Такие коды, очевидно, должны удовлетворять условию

$$0 \leqslant f(N) < 10^m, \tag{2.6}$$

где N — число, а $f(N)$ — его код, рассматриваемый как двоичное число. Кроме того, чтобы результат операции сложения получился в том же коде, в котором заданы исходные данные, должны выполняться условия:

1) для сумматора с циклическим переносом

$$f(N_1) \underset{m}{\oplus} f(N_2) = f(N_1 + N_2); \tag{2.7}$$

2) для сумматора без переноса из старшего разряда

$$f(N_1) \underset{m}{\boxed{+}} f(N_2) = f(N_1 + N_2). \tag{2.8}$$

Буквами N_1 и N_2 обозначены исходные числа.

3. Обратный код. Пусть двоичное число x представлено n целыми и ν дробными разрядами, то есть имеет вид:

$$x = s a_{n-1} a_{n-2} \ldots a_0, \; a_{-1} \ldots a_{-\nu}.$$

Обратным кодом $[x]^k_{\text{обр}}$ числа x называется целое $(n+\nu+k)$-разрядное число $(k \geqslant 1)$, определяемое по формуле

$$[x]^k_{\text{обр}} = \begin{cases} \underbrace{00 \ldots 0}_{k \text{ нулей}} \, a_{n-1} a_{n-2} \ldots a_0 a_{-1} \ldots a_{-\nu} & \text{при } x \geqslant 0, \\ \underbrace{11 \ldots 1}_{k \text{ единиц}} \, \bar{a}_{n-1} \bar{a}_{n-2} \ldots \bar{a}_0 \bar{a}_{-1} \ldots \bar{a}_{-\nu} & \text{при } x \leqslant 0, \end{cases} \tag{2.9}$$

где $\bar{a}_i = 1 - a_i \quad (i = n-1, \; n-2, \ldots, \; 0, \; -1, \ldots, \; -\nu)$.

Старшие k разрядов обратного кода называют *знаковыми*, а остальные разряды — *цифровыми*.

Пример 2.3. Число нуль имеет два обратных кода, называемых соответственно обратным кодом положительного и обратным кодом

отрицательного нуля:

$$[0]^k_{\text{обр}} = [+0]^k_{\text{обр}} = \underbrace{00\ \ldots\ 0}_{n+\nu+k \text{ нулей}},$$

$$[0]^k_{\text{обр}} = [-0]^k_{\text{обр}} = \underbrace{11\ \ldots\ 1}_{n+\nu+k \text{ единиц}}.$$

При каждой системе кодирования величина k имеет фиксированное значение. Для того чтобы по обратному коду определить изображаемое им число, необходимо знать, чему равны k и ν (значение n можно при этом определить по числу разрядов обратного кода).

Сравнивая (2.2) и (2.9), легко заметить, что для положительных чисел их обратные и прямые коды (при одинаковых количествах знаковых разрядов) совпадают, а для отрицательных чисел переход от прямого кода к обратному можно осуществить, заменяя все цифры прямого кода, кроме k старших, их дополнениями до 1.

Если числа x и y представлены каждое n целыми и ν дробными разрядами и удовлетворяют условию

$$|x+y| < 10^n, \tag{2.10}$$

(10 — двоичная запись числа два), то, как показывает непосредственная проверка (см., например, [30]), их обратные коды удовлетворяют обоим условиям (2.6) и (2.7) для $m = n+\nu+k$. Таким образом, сумматор с циклическим переносом из старшего разряда в младший позволяет производить сложение чисел, представленных в обратном коде.

При $|x+y| \geqslant 10^n$ условие (2.7) оказывается невыполненным и происходит *переполнение разрядной сетки сумматора.*

Пример 2.4. Пусть $n=0$, $\nu=9$, $k=2$, и пусть $x=0{,}101101101$, $y=-0{,}100001110$. При этом $x+y=0{,}100011111$, то есть $|x+y|<10^n$. По формуле (2.9) получается: $[x]^2_{\text{обр}}=00101101101$, $[y]^2_{\text{обр}}=11110110001$, $[x+y]^2_{\text{обр}}=00100011111$. Выполняя циклическое сложение величин $[x]^2_{\text{обр}}$ и $[y]^2_{\text{обр}}$, убеждаемся в том, что результатом является $[x+y]^2_{\text{обр}}$:

$$\begin{array}{ll} \oplus & 00101101101 \\ 11 & 11110110001 \\ \hline & 100100011110 \\ & \quad\ \uparrow\ \text{(циклический перенос)} \\ \hline & 00100011111 \end{array}$$

Пример 2.5. Пусть $n=0$, $\nu=8$, $k=2$, и пусть $x=0{,}10001001$, $y=0{,}11011011$. В данном случае $x+y=1{,}01100100>10^n=1$ и, следовательно, при циклическом сложении обратных кодов чисел x и y должно произойти переполнение разрядной сетки сумматора. Действительно, сумма $x+y$ обратного кода не имеет. С другой стороны, циклическое сложение величин $[x]^2_{\text{обр}}=0010001001$ и $[y]^2_{\text{обр}}=0011011011$ дает

$$\begin{array}{r} 0010001001 \\ \oplus\ 0011011011 \\ \hline 0101100100 \end{array}$$
$$1\ 0$$

Результат циклического сложения в знаковых разрядах имеет комбинацию 01 и, следовательно, не является обратным кодом.

Замечания. 1. Если $k>1$, то признаком переполнения разрядной сетки сумматора является возникновение недопустимой комбинации нулей и единиц в знаковых разрядах результата. При $k=1$ признаком переполнения разрядной сетки является неравенство старшей цифры результата циклического сложения старшим цифрам кодов $[x]^1_{\text{обр}}$ и $[y]^1_{\text{обр}}$.

2. Вычитание чисел легко сводится к сложению с помощью формулы

$$x-y=x+(-y),$$

из которой следует, что при $|x-y|<10^n$ имеет место

$$[x-y]^k_{\text{обр}}=[x]^k_{\text{обр}}\underset{n+\nu+k}{\oplus}[-y]^k_{\text{обр}}.$$

4. Дополнительный код. Пусть число x имеет n целых и ν дробных разрядов, то есть записывается в виде

$$x=sa_{n-1}\ \ldots\ a_0,\ a_{-1}\ \ldots\ a_{-\nu}.$$

Дополнительным кодом $[x]^k_{\text{доп}}$ числа x называется целое $(n+\nu+k)$-разрядное число, определяемое по формуле

$$[x]^k_{\text{доп}}=\begin{cases} \underbrace{0\ \ldots\ 0}_{k\text{ нулей}}\,a_{n-1}\ldots a_0a_{-1}\ldots a_{-\nu} & \text{при } x\geqslant 0, \\ \underbrace{1\ \ldots\ 1}_{k\text{ единиц}}\,\bar{a}_{n-1}\ldots\bar{a}_0\bar{a}_{-1}\ldots\bar{a}_{-\nu}\underbrace{|+|}_{n+\nu+k}0\ldots01 & \text{при } x\leqslant 0, \end{cases} \tag{2.11}$$

где $\bar{a}_i=1-a_i$ $(i=n-1, n-2, \ldots, 0, -1, \ldots, -\nu)$.

Старшие k разрядов дополнительного кода называются *знаковыми*, а остальные разряды — *цифровыми*. Любое число,

представленное n целыми и ν дробными разрядами, включая и нуль (хотя для нуля справедливы как первая, так и вторая строки формулы (2.11)), при заданном значении k имеет единственный дополнительный код. В частности, $[0]^k_{\text{доп}} = 00 \ldots 0$.

При каждой системе кодирования величина k имеет фиксированное значение. Для того чтобы по дополнительному коду определить изображаемое им число, необходимо знать, чему равны числа k и ν (значение n можно при этом определить по количеству разрядов дополнительного кода).

Если известен дополнительный код некоторого числа x, то, заменив каждую его цифру ее дополнением до 1 и произведя над полученным результатом и числом $0 \ldots 01$ операцию сложения без переноса из старшего разряда $\left(\underset{n+\nu+k}{|\overline{+}|}\right)$, найдем дополнительный код числа $-x$.

Сравнивая формулы (2.2), (2.9) и (2.11), видим, что для положительных чисел, представленных n целыми и ν дробными разрядами, при одинаковом значении k их прямые, обратные и дополнительные коды совпадают. Для отрицательных чисел все три кода между собой отличаются. Дополнительный код числа можно получить, прибавляя к обратному коду единицу младшего разряда.

Если двоичные числа x и y, представленные каждое n целыми и ν дробными разрядами, удовлетворяют условию (2.10), то для их дополнительных кодов справедливы при $m = n + \nu + k$ формулы (2.6) и (2.8), в чем можно убедиться непосредственной проверкой [37]. Таким образом, сумматор без переноса из старшего разряда позволяет производить сложение чисел, представленных в дополнительном коде. При $|x+y| \geqslant 10^n$ (10 — двоичная запись числа два) условие (2.8) оказывается невыполненным и происходит переполнение разрядной сетки сумматора *).

Пример 2.6. Пусть $n=0$, $\nu=8$, $k=2$, и пусть $x=-0{,}10111010$, $y=0{,}00111101$. При этом $x+y=-0{,}01111101$, то есть $|x+y|<10^n$. По формуле (2.11) получаем $[x]^2_{\text{доп}}=1101000110$; $[y]^2_{\text{доп}}=0000111101$; $[x+y]^2_{\text{доп}}=1110000011$. Выполняя сложение без переноса из старшего разряда величин $[x]^2_{\text{доп}}$ и $[y]^2_{\text{доп}}$, убеждаемся в том, что результатом

*) Строго говоря, переполнение разрядной сетки происходит при $x+y \geqslant 10^n$ или $x+y < -10^n$.

является $[x+y]^2_{\text{доп}}$

$$\begin{array}{ll} \overline{|+|}_{10} & 1101000110 \\ & 0000111101 \\ \hline & 1110000011 \end{array}$$

Пример 2.7. Пусть $n=0$, $\nu=10$, $k=2$, и пусть $x=-0{,}1010000001$, $y=-0{,}1100001001$. В этом случае $x+y=-1{,}0110001010$, т. е. $|x+y|>10^n=1$ и, следовательно, $x+y$ дополнительного кода не имеет, а при сложении без переноса из старшего разряда величин $[x]^2_{\text{доп}}=110101111111$ и $[y]^2_{\text{доп}}=110011110111$ должно произойти переполнение разрядной сетки.

Действительно,

$$\begin{array}{ll} \overline{|+|}_{12} & 110101111111 \\ & 110011110111 \\ \hline & 1101001110110 \\ & 101001110110 \end{array}$$

(«потеря» переноса из старшего разряда).

Результат сложения без переноса из старшего разряда имеет в знаковых разрядах недопустимую комбинацию 10 и, следовательно, не является дополнительным кодом.

Замечания 1. Если число знаковых разрядов дополнительного кода $k>1$, то переполнение разрядной сетки проявляется в том, что в знаковых разрядах результата возникает недопустимая комбинация нулей и единиц. Это может служить признаком переполнения. При $k=1$ признаком переполнения является неравенство старших цифр дополнительных кодов исходных чисел и результата.

2. Вычитание чисел легко сводится к сложению с помощью формулы

$$x-y=x+(-y),$$

из которой следует, что при $|x-y|<10^n$ имеет место

$$[x-y]^k_{\text{доп}}=[x]^k_{\text{доп}}\ \underset{n+\nu+k}{\overline{|+|}}\ [-y]^k_{\text{доп}}.$$

5. Кодирование чисел в машинах. Числа с фиксированной запятой в машинах представляют в виде одного из описанных выше кодов — прямого, обратного или дополнительного. В некоторых машинах в различных устройствах приняты различные коды для изображения чисел. Так, в машине *Урал-1* в оперативной памяти числа записываются в прямом коде с одним знаковым разрядом ($k=1$), а в ариф-

метическом устройстве — в обратном коде с двумя знаковыми разрядами ($k = 2$). При переносе чисел из памяти в арифметическое устройство или обратно производится их перевод из одного кода в другой.

Пример 2.8. Рассмотрим машину с фиксированной запятой и двоичной рабочей системой счисления. Пусть для целой части каждого числа отведено шесть, а для дробной — семь разрядов и для изображения чисел в ячейках памяти принят прямой код с одним знаковым разрядом. Представление некоторых чисел в ячейках памяти такой машины приведено в табл. 2.1. (Здесь и дальше расположение знакового разряда слева от цифровых разрядов, вообще говоря, не обязательно.)

Таблица 2.1. Представление чисел с фиксированной запятой в прямом коде

Десятичная запись числа	Двоичная запись числа	Представление числа в машине			Набор цифр, представляющих число в машине
		знак	целая часть	дробная часть	
+21,75	+10101,11	0	010101	1100000	00101011100000
— 4,125	—100,001	1	000100	0010000	10001000010000
— 0,203125	—0,001101	1	000000	0011010	10000000011010

Пример 2.9. Машина с фиксированной запятой имеет в качестве рабочей двоичную систему счисления. Для целой части каждого числа отведено нуль, а для дробной десять разрядов. Для изображения чисел в арифметическом устройстве принят обратный код с двумя знаковыми разрядами. Представление некоторых чисел в арифметическом устройстве такой машины приведено в табл. 2.2.

Таблица 2.2. Представление чисел с фиксированной запятой в обратном коде

Десятичная запись числа	Двоичная запись числа	Представление числа в машине		Набор цифр, представляющих число в машине
		знак	дробная часть	
+0,203125	+0,001101	00	0011010000	000011010000
—0,203125	—0,001101	11	1100101111	111100101111
+0	+0	00	0000000000	000000000000
—0	—0	11	1111111111	111111111111

Пример 2.10. Машина с фиксированной запятой имеет двоичную рабочую систему счисления и использует для изображения чисел в арифметическом устройстве дополнительный код с двумя знаковыми разрядами. Для целой части каждого числа отведено нуль, а для дробной части девять разрядов.

Представление некоторых чисел в арифметическом устройстве такой машины приведено в табл. 2.3.

Таблица 2.3. Представление чисел с фиксированной запятой в дополнительном коде

Десятичная запись числа	Двоичная запись числа	Представление числа в машине		Набор цифр, представляющих число в машине
		знак	дробная часть	
+0,6875	+0,1011	00	101100000	00101100000
—0,6875	—0,1011	11	010100000	11010100000

Числа с плавающей запятой изображаются в машинах в виде кода, представляющего собой объединение двух кодов: мантиссы и порядка. При этом возможно, что и для мантиссы и для порядка принят один и тот же код или различные коды. Например, мантисса может изображаться в прямом коде, а порядок в дополнительном и т. п. Способ кодирования может быть одинаковым во всех устройствах машины или различным в различных устройствах. Так, в машине «Стрела» в оперативной памяти и мантиссы и порядки чисел представлены в прямом коде с одним знаковым разрядом, а в арифметическом устройстве мантиссы и порядки представлены в обратном коде с двумя знаковыми разрядами (но с различными количествами цифровых разрядов).

Пример 2.11. В машине с плавающей запятой в ячейках оперативной памяти для порядков и мантисс принят прямой код с одним знаковым разрядом. Запятая в мантиссе фиксирована перед старшим разрядом. При этом для мантиссы отведено семь разрядов, а для порядка — четыре, не считая знаковых. Изображение некоторых (нормализованных) чисел в оперативной памяти такой машины приведено в табл. 2.4. (Здесь и дальше расположение знаковых разрядов слева от цифровых разрядов соответственно мантиссы и порядка, вообще говоря, не обязательно.)

Таблица 2.4. Изображение в машине некоторых чисел с плавающей запятой в прямом коде

Десятичная запись числа	Нормализованная форма числа в двоичной системе	Представление числа в машине				Набор цифр, представляющий число в машине
		знак мант.	мантисса	знак пор.	порядок	
+11,25	$+0{,}101101 \cdot 10^{+100}$	0	1011010	0	0100	0101101000100
—11,25	$-0{,}101101 \cdot 10^{+100}$	1	1011010	0	0100	1101101000100
+0,23046875	$+0{,}111011 \cdot 10^{-10}$	0	1110110	1	0010	0111011010010
—0,23046875	$-0{,}111011 \cdot 10^{-10}$	1	1110110	1	0010	1111011010010
—0,1	$-0{,}(1100) \cdot 10^{-11}$*)	1	1100110	1	0011	1110011010011

*) Запись 0, (1100) означает периодическую двоичную дробь с периодом 1100.

Пример 2.12. В машине с плавающей запятой в ячейках оперативной памяти для мантисс принят прямой код с одним знаковым разрядом, а для порядков — дополнительный с одним знаковым разрядом. Запятая в мантиссе фиксирована перед старшим разрядом. При этом для мантиссы отведено восемь, а для порядка четыре разряда, не считая знаковых. Изображение некоторых нормализованных чисел в оперативной памяти такой машины показано в табл. 2.5.

Таблица 2.5. Изображение в машине некоторых чисел с плавающей запятой (мантиссы — в прямом коде, а порядки в дополнительном)

Десятичная запись числа	Нормализованная форма числа в двоичной системе счисления	Представление числа в машине				Набор цифр, представляющий число в машине
		знак мант.	мантисса	знак пор.	порядок	
+12,75	$0{,}110011 \cdot 10^{+100}$	0	1100110	0	0100	01100110000100
—0,140625	$0{,}1001 \cdot 10^{-10}$	1	10010000	1	1110	11001000011110

6. Кодирование буквенной информации. Как уже говорилось ранее, в разрядах ячеек машины могут быть записаны только буквы машинного алфавита или, что то же, цифры рабочей системы счисления (мы говорим о машинах, для которых рабочей системой счисления является двоичная). Тем не менее в ряде машин предусмотрена возможность

Таблица 2.6. Один из возможных способов двоичного кодирования букв

№ п/п	Кодируемый алфавит	Значения букв кодируемого алфавита				Двоичный код
		русские буквы	латинские буквы	цифры	знаки	
1	А	А а	A a			000001
2	Б	Б б				000010
3	В	В в	B b			000011
4	Г	Г г				000100
5	Д	Д д				000101
6	Е	Е е	E e			000110
7	Ж	Ж ж				000111
8	З	З з		3		001000
9	И	И и				001001
10	Й	Й й				001010
11	К	К к	K k			001011
12	Л	Л л				001100
13	М	М м	M m			001101
14	Н	Н н	H h			001110
15	О	О о	O o			001111
16	П	П п				010000
17	Р	Р р	P p			010001
18	С	С с	C c			010010
19	Т	Т т	T t			010011
20	У	У у	У y			010100
21	Ф	Ф ф				010101
22	Х	Х х	X x			010110
23	Ц	Ц ц				010111
24	Ч	Ч ч		4		011000
25	Ш	Ш ш				011001
26	Щ	Щ щ				011010
27	Ы	Ы ы				011011
28	Ь	Ь ь				011100
29	Ъ	Ъ ъ				011101
30	Э	Э э				011110
31	Ю	Ю ю				011111
32	Я	Я я				100000
33	D		D d			100001
34	F		F f			100010
35	G		G g			100011
36	I		I i	1		100100
37	J		J j			100101
38	L		L l			100110
39	N		N n			100111
40	Q		Q q			101000

Продолжение табл. 2.6.

№ п/п	Кодируемый алфавит	Значения букв кодируемого алфавита				Двоичный код
		русские буквы	латинские буквы	цифры	знаки	
41	R		R r			101001
42	S		S s			101010
43	U		U u			101011
44	V		V v			101100
45	W		W w			101101
46	2			2		101110
47	5			5		101111
48	6			6		110000
49	7			7		110001
50	8			8		110010
51	9			9		110011
52	.				.	110100
53	,				,	110101
54	:				:	110110
55	;				;	110111
56	?				?	111000
57	!				!	111001
58	—				—	111010
59	+				+	111011
60	=				=	111100
61	(				(	111101
62	)				)	111110
63	Пробел 1				конец слова	111111
64	Пробел 2				конец текста	000000

переработки информации, задаваемой в виде слов, состоящих из букв русского и латинского алфавитов (чаще всего применяются только заглавные буквы этих алфавитов) и из цифр десятичной позиционной системы счисления. Для представления этих букв в машине они должны быть закодированы. В табл. 2.6 показан один из возможных способов кодирования 64-буквенного алфавита в виде так называемых двоичных секстад («шестерок» двоичных цифр).

При размещении буквенной информации в ячейках обычно записываются только полные секстады, так что в случае числа разрядов, не кратного шести, несколько разрядов ячейки не используются (обычно заполняются нулями). Например, в ячейке,

содержащей сорок три разряда, можно записать семь букв и один разряд будет не использован.

Для ввода буквенной информации в машину в большинстве существующих машин применяются перфокарты или перфоленты. Пробивка информации на них производится с помощью входного перфоратора, оборудованного специальным клавишным устройством. Выдача буквенной информации из машины производится либо на специальное печатающее устройство, называемое цифро-буквенным печатающим устройством, либо на перфокарты или перфоленты, с последующей печатью с них на специальном печатающем устройстве.

Кроме описанного выше способа размещения буквенной информации в ячейках оперативной памяти, иногда применяется так называемая *плотная запись*, при которой буквенная информация размещается в разрядах группы последовательных ячеек без учета границ между ячейками.

§ 3. Основные операции машины

1. Арифметические операции. В различных машинах для выполнения арифметических операций применяются различные алгорифмы, определяющие конструкцию их арифметических устройств. Способ выполнения этих операций зависит от формы представления чисел в машине.

Сложение и вычитание чисел с фиксированной запятой. Числа с фиксированной запятой, подлежащие сложению или вычитанию, обычно представляют в обратном или дополнительном коде (в зависимости от типа сумматора, которым располагает машина), и операции производятся по правилам, описанным в предыдущем параграфе. Если в запоминающем устройстве эти числа представлены в другом коде, то предварительно производится их перевод в код, принятый для выполнения действий. При переполнении разрядной сетки из сумматора в устройство управления передается специальный сигнал. Этот сигнал, являющийся одноразрядным двоичным числом, принято называть сигналом φ. Таким образом, при переполнении $\varphi = 1$, а при отсутствии переполнения $\varphi = 0$.

Нормализация и денормализация. Нормализацией называется операция, преобразующая ненормализованное число $m' \cdot 10^{n'}$ и нормализованное число $m \cdot 10^{n}$. (Нормализованным

будем считать число, мантисса которого удовлетворяет условию $\frac{1}{10} \leqslant |m| < 1.$) Если $10^{-r-1} \leqslant |m'| < 10^{-r}$, то нормализация сводится к двум умножениям:

$$m = m' \cdot 10^{r}.$$
$$10^{n} = 10^{n'} \cdot 10^{-r}.$$

Первое из умножений может осуществляться путем сдвига влево на r разрядов кода, содержащегося в цифровых разрядах мантиссы. Последнее из умножений равносильно вычитанию

$$n = n' - r.$$

Участвующее в нормализации число r называют *фактором нормализации.*

В машинах с плавающей запятой числа $x = m' \cdot 10^{n'}$ представлены в виде пар чисел m', n'. Таким образом, нормализация сводится к переходу от пары чисел m', n' к паре чисел m, n.

В машинах с плавающей запятой нормализация чисел является вспомогательной при некоторых других операциях, и обычно среди возможных команд машины нет специальных команд для выполнения нормализации.

В некоторых машинах с фиксированной запятой предусматривается возможность вычислений с плавающей запятой (осуществляемых программным путем). В отличие от машин с плавающей запятой, в таких машинах мантисса и порядок числа хранится не в одной и той же, а в различных ячейках. Если в таких машинах предусмотрена операция нормализации и специальный тип команды для ее выполнения, то содержимое ячейки, отведенной под мантиссу, подвергается умножению на 10^{r}, а содержимое ячейки, отведенной под порядок, — уменьшению на r. Значение фактора нормализации r определяется машиной автоматически.

Денормализация является действием, обратным нормализации, и в машинах с плавающей запятой сводится к переходу от пары чисел m, n к паре чисел $m \cdot 10^{-r}$, $n + r$. Число r при этом называется *фактором денормализации.*

При выполнении операции

$$m' = m \cdot 10^{-r}$$

некоторое количество младших цифр мантиссы выходит «вправо» за пределы разрядной сетки машины. Младший из удерживаемых разрядов при этом обычно округляется.

Сложение и вычитание чисел с плавающей запятой. В машине числа с плавающей запятой, подлежащие сложению или вычитанию, сперва приводятся к одинаковому порядку. Осуществляется это с помощью операции денормализации.

Предположим, что оба слагаемых нормализованы и заданы в виде пар чисел m_1, n_1 и m_2, n_2. Пусть для определенности $n_2 > n_1$. Тогда число m_1, n_1 денормализуется с фактором денормализации

$$r = n_2 - n_1.$$

В результате этого оно оказывается приведенным к виду $m_1 \cdot 10^{n_2 - n_1}$, n_2.

После денормализации числа, имеющего меньший порядок, производится сложение мантисс способом, описанным для сложения чисел в машинах с фиксированной запятой. При сложении (или вычитании) мантисс на двоичном сумматоре возможен один из трех случаев.

1. Сумма $m = m_1 \cdot 10^{n_2 - n_1} + m_2$ удовлетворяет условию $\frac{1}{10} \leqslant |m| < 1$. В этом случае результат сложения фиксируется в машине в виде пары чисел m, n_2.

2. Число m удовлетворяет условию $|m| < \frac{1}{10}$ (т. е. при сложении произошла денормализация «вправо»). В этом случае результат сложения m, n_2 нормализуется, после чего фиксируется в качестве суммы.

3. При сложении на сумматоре кодов чисел $m_1 \cdot 10^r$ и m_2 происходит переполнение разрядной сетки части сумматора, отведенной для оперирования мантиссами (говорят, что произошло нарушение нормализации результата «влево»). В этом случае производится исправление результата, заключающееся в сдвиге всех разрядов кода, выдаваемого сумматором, вправо на один разряд. При этом в самом старшем разряде кода сохраняется его содержимое, а младший из удерживаемых разрядов округляется обычным способом. Одновременно порядок увеличивается на единицу. Такое исправление ре-

зультата равносильно получению в качестве суммы нормализованного числа $(m_1 \cdot 10^r + m_2) \cdot 10^{-1}$, $n_2 + 1$.

Переполнение разрядной сетки той части сумматора, которая отведена для оперирования мантиссами, не мешает получению искомой суммы или разности. Переполнение разрядной сетки в целом после сложения или вычитания может произойти только при увеличении порядка n_2 на 1 и заключается в переполнении той части разрядной сетки, которая отведена под порядок. В этом случае из сумматора в управляющее устройство машины передается сигнал о переполнении, называемый, как и для машин с фиксированной запятой, сигналом φ (при переполнении $\varphi = 1$, в противном случае $\varphi = 0$).

Умножение и деление чисел. Умножение чисел как в машинах с плавающей запятой, так и в машинах с запятой, фиксированной перед старшим разрядом, обычно производится в прямом коде.

Сначала содержимое старших разрядов (знаки чисел) складывается на одноразрядном сумматоре. При таком сложении перенос, если он возникает, теряется. В результате произведению приписывается знак в соответствии с известным правилом алгебры.

Затем производится перемножение чисел.

Умножение чисел в машинах с фиксированной запятой и умножение мантисс в машинах с плавающей запятой производится одинаково.

В двоичной системе счисления первый сомножитель приходится умножать только на нули или единицы, являющиеся цифрами второго сомножителя. Получаемые частичные произведения следует подписывать друг под другом, сдвигая каждое новое произведение на один разряд. Частичные произведения, равные нулю, следует опускать. Благодаря этому умножение в машине можно осуществить путем последовательных сдвигов одного из сомножителей на количество разрядов, определяемое цифрами второго сомножителя, и сложения между собой получающихся результатов. В некоторых машинах умножение первого сомножителя на второй начинается не с младшего, а со старшего разряда второго сомножителя и предусматривает последовательный переход от разряда к разряду второго сомножителя слева направо.

Пример 2.13. Умножение, начиная со старшего разряда, выполняют следующим образом:

×	0,11001101		
	0,11101001		
	01100110	1	— результат первого сдвига
	00110011	01	— результат второго сдвига
+	00011001	101	и т. д.
	00000110	01101	
	00000000	11001101	
	0,10111010	10010101	

Округление: 0,10111011

Кроме того, в машине с плавающей запятой производится алгебраическое сложение порядков сомножителей, дающее порядок произведения. Если после умножения в ячейке машины с плавающей запятой результат получится ненормализованным, то он (до округления) обычно подвергается нормализации.

При делении знак частного получается путем суммирования цифр, изображающих знаки делимого и делителя, на одноразрядном сумматоре, так же как при умножении.

Деление чисел в машинах с фиксированной запятой и деление мантисс в машинах с плавающей запятой производится одинаково. Оно осуществляется путем вычитания из делимого делителя, сдвига влево полученного остатка, нового вычитания (из результата сдвига) делителя и т. д.

Кроме того, в машине с плавающей запятой производится алгебраическое вычитание порядка делителя из порядка делимого; если частное окажется ненормализованным, то производится его нормализация.

При сложении порядков сомножителей или вычитании из порядка делимого порядка делителя может произойти переполнение разрядной сетки. В этом случае из арифметического устройства в управляющее устройство передается сигнал $\varphi = 1$.

2. Неарифметические операции над машинными словами и переносы. Будем условно считать разряды машинного слова, хранимого в ячейке памяти или участвующего в операции, перенумерованными справа налево числами 0, 1, 2, ..., $n-1$, n (заметим, что в описаниях разных программно-управляемых машин мы встречаем разные способы

нумерации разрядов ячеек). Двоичную цифру, стоящую в i-м (по нашей нумерации) разряде кода a, b или c, будем обозначать одним из символов a_i, b_i или c_i.

Сдвиг. Сдвигом на k разрядов (k — целое) кода

$$a = a_n a_{n-1} \dots a_1 a_0$$

называют поразрядную операцию, определяемую формулой

$$c_i = \begin{cases} a_{i-k} & \text{при } 0 \leqslant i - k \leqslant n, \\ 0 & \text{при остальных значениях } i - k, \end{cases}$$

где c_i ($i = 0, 1, \dots, n$) — цифры результата.

Эту операцию иногда при $k > 0$ называют сдвигом на k разрядов *влево*, а при $k < 0$ сдвигом на $|k|$ разрядов *вправо*.

Пример 2.14. В результате сдвига на три разряда (при $n=7$) числа

$$a = 10011101$$

получается число

$$c = 11101000.$$

Сдвиг того же кода a на -3 разряда дает

$$c' = 00010011.$$

Поразрядное логическое умножение. Поразрядное логическое умножение двоичных кодов

$$a = a_n a_{n-1} \dots a_1 a_0,$$
$$b = b_n b_{n-1} \dots b_1 b_0$$

выполняется в соответствии с формулой

$$c_i = a_i \wedge b_i \qquad (i = 0, 1, \dots, n).$$

С помощью описанной операции можно «выделять» различные «части» одного из оперируемых кодов при соответствующем выборе второго из них.

Пример 2.15. Пусть $n = 7$ и

$$a = 10111001.$$

Если a является прямым кодом некоторого числа x (в данном случае отрицательного), то для выделения знака числа x в качестве b необходимо взять код $b = 10000000$. При этом получится $c = 10000000$. Для выделения абсолютной величины числа x необходимо взять $b' = 01111111$. При этом получится $c' = 00111001$.

Поразрядное логическое сложение. Поразрядное логическое сложение кодов a и b выполняется в соответствии с формулой

$$c_i = a_i \vee b_i \qquad (i = 0, 1, \ldots, n).$$

С помощью описанной операции можно из двух частей некоторого кода, хранимых в разных ячейках памяти, «сформировать» этот код.

Пример 2.16. Пусть $n = 7$. Предположим, что $a = 110100000$, $b = 000001010$. С помощью поразрядного логического сложения кодов a и b получим $c = 110101010$.

Сравнение. Сравнением называется поразрядное логическое отрицание равнозначности кодов a и b в соответствии с формулой

$$c_i = a_i \overline{\sim} b_i \qquad (i = 0, 1, \ldots, n).$$

Если в i-х разрядах кодов a и b стоят одинаковые цифры, то в i-м разряде результата получается нуль. Если же в i-х разрядах кодов a и b стоят различные между собой цифры, то в i-м разряде результата будет 1.

Пример 2.17. Пусть $n = 7$. Предположим, что $a = 11010001$, $b = 10011101$. В результате сравнения этих кодов получится $c = 01001100$. Этот результат показывает, что 0-й, 1-й, 4-й, 5-й и 7-й разряды сравниваемых кодов между собой одинаковы, а 2-й 3-й и 6-й — различны.

Контрольное суммирование. В некоторых машинах предусматривается операция сложения кодов с циклическим переносом (из старшего разряда в нулевой)

$$c = a \underset{n+1}{\oplus} b,$$

которую часто называют контрольным суммированием. В машинах с фиксированной запятой, имеющих сумматор с циклическим переносом, эта операция отличается от сложения тем, что сигнал $\varphi = 1$, если он возникает, не передается в управляющее устройство (как говорят, «блокируется»). В машинах с плавающей запятой эта операция не имеет ничего общего со сложением (см. выше п. 1).

Операции переноса. Операции переноса определяются формулой

$$c_i = a_i \qquad (i = 0, 1, \ldots, n), \tag{2.12}$$

причем как ячейка с исходным кодом a, так и ячейка, в которую производится перенос (записывается код c), принадлежат запоминающим устройствам A. Переносы кодов могут производиться:

1) из одних ячеек оперативной памяти в другие ячейки,

2) между группой ячеек оперативной памяти и определенной зоной магнитной ленты или магнитного барабана (групповой перенос между оперативной и внешней памятью),

3) из оперативной памяти на печать, перфокарты или перфоленты — выдача результатов,

4) с перфокарт или перфоленты в оперативную память — ввод данных.

В машинах, в которых сумматор используется не только для выполнения операций, но и для хранения исходных для операции данных и ее результата, предусматривается также перенос из оперативной памяти в сумматор и из сумматора в оперативную память.

3. Логические операции. Логическими операциями будем называть операции, заключающиеся в вычислении того или иного предиката и характеризующиеся тем, что их результаты записываются в D (т. е. в одно из одноразрядных запоминающих устройств, предназначенных для хранения сигналов). Логические операции можно разбить на группы в зависимости от того, откуда для них берутся исходные данные. Одну из таких групп образуют операции вида $A \to D$, исходные данные для которых обычно берутся из сумматора или другой запоминающей ячейки арифметического устройства. Результаты таких операций обычно называют значениями сигнала ω. Одна или несколько из логических операций этой группы называются получением сигнала φ (сигнала переполнения). Другую группу логических операций образуют операции вида $F \to D$, исходные данные для которых берутся из индексных регистров. Возможны и другие группы логических операций. Логические операции, применяемые в машинах, весьма разнообразны. В качестве примеров приводим некоторые из них, используя представления кодов, применяемые в предыдущем разделе.

Проверка знака. Проверкой знака называется логическая операция вида $A \to D$, определяемая формулой

$$\omega = a_n$$

и являющаяся вычислением значения предиката, гласящего «цифра n-го разряда кода a равна 1». Название операции происходит от того, что в случае, когда a является прямым или обратным кодом некоторого числа x, она приводит к образованию сигнала $\omega = 1$ при $x \leqslant -0$, а для $x \geqslant +0$ в результате проверки знака получается сигнал $\omega = 0$. Если a является дополнительным кодом числа x, то при $x \geqslant 0$ операция проверки знака дает $\omega = 0$, при $x < 0$ дает $\omega = 1$.

Проверка переполнения. Операция проверки переполнения при сложении, принадлежащая группе $A \to D$, в случае, когда применяется обратный или дополнительный код с двумя знаковыми разрядами, имеет вид

$$\varphi = a_n \overline{\vee} a_{n-1}.$$

В некоторых машинах существуют режимы работы, при которых сигнал φ используется для изменения порядка выполнения команд. Обычно сигнал $\varphi = 1$ останавливает машину.

Проверка наличия нулей во всех разрядах. Проверка наличия нулей во всех разрядах кода a представляет собой вычисление предиката

$$\omega = \overline{a_n \vee \ldots \vee a_1 \vee a_0},$$

который равен 1, когда $a_i = 0$ $(i = 0, 1, \ldots, n)$ и равен 0, если хотя бы в одном разряде кода a стоит цифра 1.

Данная логическая операция в случае ее принадлежности к группе $A \to D$ может служить для проверки результата такой операции как сравнение. В случае поразрядного тождества сравниваемых кодов она будет давать $\omega = 1$, а при различии их хотя бы в одном из разрядов будет получаться $\omega = 0$. Та же логическая операция может служить для проверки равенства нулю кода, содержащегося в индексном регистре (в этом случае она должна относиться к группе $F \to D$).

Проверка абсолютной величины. В машинах с плавающей запятой часто используется логическая операция, называемая проверкой абсолютной величины. Если μ — номер разряда ячейки, в котором при хранении в ней нормализованного числа помещается знаковый разряд порядка, то описываемую операцию можно задать как предикат

$$\omega = \bar{a}_\mu \wedge (a_{\mu-1} \vee a_{\mu-2} \vee \ldots \vee a_0).$$

Легко сообразить, что эта операция дает сигнал $\omega = 1$ в том и только том случае, если выполнены два условия:

1) в знаковом разряде порядка стоит цифра 0;

2) среди разрядов, отведенных для цифровой части порядка, хотя бы один равен единице.

Если a является кодом нормализованного числа x, то порядок этого числа при $\omega = 1$ (и только при этом условии) удовлетворяет неравенству $p \geqslant 1$. Отсюда следует, что сигнал $\omega = 1$ является признаком того, что

$$|x| = |m \cdot 10^p| \geqslant \frac{1}{10} \cdot 10^1 = 1.$$

Полученная формула

$$|x| \geqslant 1$$

поясняет смысл и название операции.

4. Операции, изменяющие содержимое регистра команд. В различных машинах можно встретить весьма разнообразные операции, изменяющие содержимое регистра команд (относящиеся к классу $\rightarrow B$). Эти операции, как и логические, можно разбить на группы в зависимости от того, откуда берутся для них исходные данные.

Выборка команды. Выборка команды относится к классу операций $A \rightarrow B$ и представляет собой перенос кода в соответствии с формулой (2.12). Она отличается от вышеописанных переносов тем, что ее результат записывается не в одно из запоминающих устройств A, а в регистр команд (B). Содержимое этого регистра определяет действия машины в течение некоторого отрезка времени. Выборка команды производится из той ячейки оперативной памяти, номер которой в этот момент фиксирован в счетчике команд. Описанную операцию выборки команд назовем выборкой команды I рода.

В некоторых машинах (например, *Урал-2*) регистр команд представляет собой сумматор. В таких машинах выборка команд относится к классу операций $A, B \rightarrow B$ и представляет собой сложение кода, находящегося в некоторой ячейке оперативной памяти (A), с прежним содержимым регистра команд (B) и запись результата в регистр команд ($\rightarrow B$). В большинстве случаев предыдущее содержимое регистра команд равно нулю. Такую выборку команды будем называть II рода.

Операция изменения команды. Операция изменения команды обычно применяется в машинах, в которых принята выборка команд II рода, выполняется перед выборкой и относится к виду $A \to B$. Она представляет собой перенос кода из ячейки оперативной памяти в регистр команд. При этом в машине принимаются меры, обеспечивающие сохранение содержимого регистра команд до момента выполнения очередной операции выборки команды.

Изменение команды на содержимое индексного регистра. В машинах, оборудованных индексными регистрами, предусматривают операцию, называемую изменением команды на содержимое индексного регистра. Эта операция относится к виду B, $F \to B$ и заключается в том, что содержимое регистра команд складывается (или вычитается) с содержимым индексного регистра, причем результат записывается в регистр команд. Получаемые при этом адреса команды называются *исполнительными* адресами.

5. Операции перехода. Операции перехода, или, как говорят, передачи управления, характеризуются тем, что их результаты записываются в счетчик команд C. После рассмотрения операции переноса команды из оперативной памяти в регистр команд, называемой выборкой команды, смысл операций перехода нетрудно уяснить, имея в виду, что номер выбираемой команды определяется содержимым счетчика команд.

Безусловные переходы. В машинах с заданным порядком выполнения команд существует две разновидности безусловного перехода. Одна из разновидностей безусловного перехода представляет собой прибавление единицы к содержимому счетчика команд (операция вида $C \to C$). С помощью этой операции обеспечивается выполнение машиной команд в порядке номеров ячеек оперативной памяти.

Вторая разновидность безусловного перехода представляет собой перенос адреса команды из B (т. е. из регистра команд) в C (счетчик команд).

В машинах со свободным порядком выполнения команд эта операция совмещается с выборкой команды, ибо в таких машинах счетчик команд является частью регистра команды.

Условные переходы. В существующих машинах с заданным порядком выполнения команд встречаются одноадресные и двухадресные условные переходы.

Одноадресный условный переход представляет собой операцию вида

$$c=\begin{cases} b & \text{при } \omega=1, \\ a+1 & \text{при } \omega=0, \end{cases} \tag{2.13}$$

где a — число, хранящееся до выполнения этой операции в счетчике команд, b — адресная часть кода команды, находящегося в регистре команд, а ω — содержимое регистра D. Результат этой операции записывается в счетчик команд. Таким образом, при $\omega=0$ происходит увеличение содержимого счетчика команд на 1, а при $\omega=1$ — засылка в этот счетчик адреса команды, находящейся в регистре команд.

В некоторых машинах, кроме условного перехода, отвечающего (2.13), или вместо него, предусматривается одноадресный условный переход, соответствующий формуле

$$c=\begin{cases} a+1 & \text{при } \omega=1, \\ b & \text{при } \omega=0. \end{cases}$$

В одноадресных машинах возможны только одноадресные условные переходы. Однако существуют и трехадресные машины (например, *БЭСМ*), в которых используется эта операция.

Двухадресный условный переход представляет собой операцию, определяемую формулой

$$c=\begin{cases} a & \text{при } \omega=1, \\ b & \text{при } \omega=0, \end{cases}$$

где a и b — адреса команды, хранящейся в регистре команд (B), а ω — содержимое D. Результат этой операции записывается в счетчик команд C.

В результате выполнения этой операции при $\omega=1$ происходит передача управления команде, хранящейся в ячейке, номер которой равен a, а при $\omega=0$ — команде, хранящейся в ячейке с номером b.

Замечание. В связи с операциями перехода необходимо упомянуть операцию формирования команды возврата, которая заключается в том, что с помощью содержимого счетчика команд строится команда безусловного перехода и переносится в одну из ячеек оперативной памяти. Эта операция относится к виду $C \rightarrow A$.

6. Операции, изменяющие содержимое индексного регистра. Операции, изменяющие содержимое индексного регистра относятся к шестому классу. Приводим описание лишь некоторых из них.

Засылка в индексный регистр. В некоторых машинах предусматривается перенос части кода из регистра выполняемой команды в индексный регистр (операция вида $B \rightarrow F$); в других машинах применяется перенос содержимого из ячеек оперативной памяти в индексный регистр (операция вида $A \rightarrow F$). Обе эти операции называются засылками в индексный регистр.

Модификация содержимого индексного регистра. Операция изменения содержимого индексного регистра относится к виду $F \rightarrow F$ и заключается в сложении (или вычитании) содержимого индексного регистра с некоторой константой и записи результата в индексный регистр.

7. Операции над буквенной информацией. Операции над буквенной информацией в большинстве машин являются сложными и представляют собой комбинации описанных выше операций, выполняемых над последовательностями кодов (машинных слов) и их частей. Это связано с тем, что слова, из которых состоит буквенная информация, не укладываются в формат машинного слова.

Основными операциями над буквенной информацией являются:

а) определение длины слова;

б) перенос слова с одного места оперативной памяти в другое;

в) выделение определенной части заданного слова;

г) сравнение двух слов.

В машинах с переменной длиной ячеек и машинных слов (в которых адрес машинного слова задается парой чисел: номером разряда оперативной памяти и длиной машинного слова; в таких машинах все разряды оперативной памяти перенумерованы насквозь) или в машинах с фиксированной длиной ячеек (обычно небольшой), но с переменной длиной машинного слова (в которых адрес машинного слова задается в виде пары чисел: номером ячейки, в которой начинается машинное слово, и числом ячеек, в которых оно хранится) операции над буквенной информацией могут быть элементарными.

§ 4. Система команд программно-управляемой машины

1. Система команд. Общий вид команды. Совокупность различных типов команд, принятых в данной программно-управляемой машине, рассматриваемая вместе с законом композиции, с помощью которого из кодов операций и номеров ячеек конструируются команды, называется системой команд программно-управляемой машины.

Обычно типы команд объединяют в группы (в частном случае группа может состоять из одного типа) и для каждой группы составляют так называемый общий вид команды. *Общим видом* команды называется условная запись в виде последовательности символов, описывающая способ композиции конкретных команд по коду операций и номерам ячеек (названиям запоминающих устройств).

Система команд программно-управляемой машины обычно задается в виде таблицы, в которой приведены общие виды команд, даны пояснения способов получения конкретных команд и описания действий машины, определяемых командами. Систему команд программно-управляемой машины не следует смешивать с ее системой операций. Две машины, имеющие одинаковые системы операций, могут иметь различные системы команд, например отличающиеся количествами входящих в них типов команд и т. п.

Пример 2.18. Система команд машины *Стрела* содержит типы команд

$$x_1\ x_2\ y_1\ 0\ 01,$$
$$x_1\ x_2\ y_1\ 1\ 01,$$

где 0 01 и 1 01 — коды операций. Для этих двух типов команд составляют общий вид команды

$$x_1\ x_2\ y_1\ k\ 01,$$

который называют общим видом команды сложения (или коротко — командой сложения).

Заменяя символы x_1, x_2, y_1 номерами ячеек оперативной памяти, а символ k цифрой 0 или цифрой 1, получаем конкретную команду.

В той же системе команд содержатся для $m = 1, 2, \ldots, 53$ (m задается в восьмеричной системе счисления) типы команд

$$x_1\ y_1\ 4000 + m\ 0\ 14,$$
$$x_1\ y_1\ 4000 + m\ 1\ 14$$

и для $m=-1, -2, \ldots, -53$ типы команд

$$x_1\ y_1\ 4100+|m|\ 0\ 14$$
$$x_1\ y_1\ 4100+|m|\ 1\ 14$$

где $4000+m\ 0\ 14$, $4000+m\ 1\ 14$, $4100+|m|\ 0\ 14$ и $4100+|m|\ 1\ 14$— коды операций. Для этой совокупности типов команд, пользуясь обозначением

$$\operatorname{sign} m=\begin{cases}0 \text{ при } m>0,\\ 1 \text{ при } m<0,\end{cases}$$

можно задать общий вид команды:

$$x_1\quad 4000+100\operatorname{sign} m+|m|\quad y_1\quad k\quad 14$$

который называют общим видом команды сдвига на $|m|$ разрядов по адресу. В этом общем виде команды часть кода операции, определяющая значение показателя сдвига m, содержится в той части кода команды, которая в ряде других общих видов команд является адресом.

2. Условная структура общего вида команды. В реальных машинах коды всех команд делят на несколько групп, (в частном случае все коды команд могут образовать только одну группу), в каждой из которых коды команд условно разделяют на части, называемые *кодом операций, адресами* и *признаками*. Так, любой общий вид команды для машины *Стрела* может быть представлен в виде

$$x_1\ x_2\ y_1\ k\ \theta$$

где x_1 — первый адрес, x_2 — второй адрес, y_1 — третий адрес, k — контрольный знак (признак), θ — код операций. Коды команд машины *Стрелы* образуют только одну группу.

Нужно иметь в виду, что в конкретных командах при этом может оказаться, что код, стоящий в адресе, является не номером ячейки или группы ячеек, а частью кода операций и т. п.

Литература к главе 2: [35], [37], [63].

ГЛАВА 3

ОСОБЕННОСТИ РЕШЕНИЯ ЗАДАЧ НА ЦИФРОВЫХ ПРОГРАММНО-УПРАВЛЯЕМЫХ МАШИНАХ

§ 1. Порядок решения задач на машине

1. Основные области применения цифровых программно-управляемых машин. Первой областью применения цифровых программно-управляемых машин явилась область решения математических задач. Для нее характерно наличие большого количества детально разработанных численных методов. Появление программно-управляемых машин, позволяющих выполнять операции с большой скоростью, привело к появлению новых численных методов.

Развитие методики алгоритмизации процессов позволило математически описывать такие процессы, описание которых прежними методами было невозможно. В результате появилась новая область применения программно-управляемых машин — математическое или, как говорят, программное моделирование процессов.

Автоматическое кодирование и декодирование информации, автоматический перевод текстов с одного языка на другой — задачи, успешно решаемые с помощью программно-управляемых машин.

Важнейшей областью применения машин является автоматизация информационных работ, сводящаяся к тому, что в запоминающих устройствах машины производится накопление больших объемов информации, на основании которой по запросам производится выдача справок.

Не менее важной является область применения машин для экономических расчетов и планирования сложных комплексов работ.

Наконец, необходимо указать такую область применения программно-управляемых машин, как управление другими машинами и их комплексами.

Здесь перечислены лишь важнейшие области применения программно-управляемых машин. При разработке машины обычно учитывают область применения, для которой она создается, отражая ее особенности в системе команд и операций машины, в тех или иных особенностях ее устройств ввода и вывода, в объемах ее запоминающих устройств.

Например, вычислительные машины обладают системой операций, близкой к системе математических операций, применяемых для решения задач численными методами. В таких машинах зачастую не предусматривается возможность ввода и вывода буквенной информации. Объемы их ВЗУ превосходят объем оперативной памяти в небольшое число раз.

Машины, предназначенные для математического моделирования, должны иметь в своем составе датчики случайных чисел, большую по объему оперативную память и, помимо математических операций, располагать развитой системой логических операций.

Автоматическое кодирование информации и автоматический перевод текстов с одного языка на другой связаны с применением машинных словарей, имеющих большие объемы и не изменяющихся в процессе работы машины. Для хранения таких словарей в составе машины могут быть предусмотрены специальные оперативно работающие односторонние запоминающие устройства (позволяющие производить считывание, но не запись). Устройства ввода и вывода должны допускать ввод и выдачу больших объемов буквенной информации.

Информационные машины должны обладать устройствами для ввода и вывода буквенной информации; развитой системой внешних запоминающих устройств, обеспечивающих возможность одновременного обмена информацией с несколькими отдельными блоками оперативной памяти; возможностью одновременного выполнения нескольких программ (пока по одной программе производится ввод в оперативную память информации из ВЗУ, по другой программе или по этой же самой, например, производится обработка информации уже

находящейся в памяти); обладать системой операций, приспособленной к обработке слов и массивов слов. Некоторые информационные машины должны быть оборудованы устройствами для ввода информации из каналов связи и выдачи ее в каналы связи.

Машины для экономических расчетов близки по своим особенностям к информационным.

Управляющие машины обычно снабжаются специальными устройствами преобразования непрерывной информации в дискретную для ввода ее в машину и устройствами преобразования дискретной информации в непрерывную для подачи управляющих сигналов управляемым объектам.

Приведенное здесь описание особенностей программно-управляемых машин различного назначения является лишь приблизительным и весьма неполным. Заметим, что в каждой области применения встречаются машины широкого назначения, называемые *универсальными*, и машины узкого назначения, называемые *специализированными*. Так, существуют универсальные и специализированные вычислительные машины, универсальные и специализированные управляющие машины и т. п.

2. Основные этапы работы. Задача (или класс задач), подлежащая решению на машине, обычно задается в математической (или какой-либо иной) формулировке. Решение задачи с помощью программно-управляемой машины распадается на следующие этапы:

1) разработка алгорифма решения задачи, то есть *решающего алгорифма*;

2) программирование;

3) отладка программы на машине;

4) автоматическое решение задачи.

Кроме осуществления перечисленных главных этапов работы, необходимо также выполнение определенной технической работы, связанной с вводом материала в машину. Подлежащий вводу материал должен быть предварительно перенесен с помощью перфорирующих устройств на перфоленты или перфокарты. При программировании вручную вводу подлежит программа, составленная на бланках, при автоматическом программировании необходим ввод информации о решающем алгорифме.

3. Краткая характеристика основных этапов. Выбор решающего алгорифма в ряде случаев требует высокой квалификации и связан с большими трудностями. Формальных правил для выполнения этой работы не существует. Основой для построения решающего алгорифма при решении математических задач могут служить методы численного анализа. Однако во многих случаях решающий алгорифм приходится специально разрабатывать. Как правило, с этим сталкиваются при разработке программ для решения информационных задач, для управления объектами, для перевода текстов с одного языка на другой и т. п. В дальнейшем всегда, когда это специально не оговаривается, речь идет о применении машин для решения математических задач.

Программирование представляет собой построение алгорифма, равнозначного решающему алгорифму и выполнимого на имеющейся машине.

При разработке решающего алгорифма и особенно при программировании могут быть допущены ошибки. Для устранения ошибок программирования программу тщательно проверяют, однако, как показывает практика, полного устранения ошибок добиться этим путем очень трудно. Окончательно устраняются ошибки при отладке программы на машине. В простейшем случае отладка заключается в решении специально подобранных упрощенных примеров (*тестов*) и сличении получаемых результатов с заранее заготовленными. В некоторых случаях после отладки программы на машине проводят ряд так называемых *контрольных просчетов*, которые иногда необходимы также и для уточнения решающего алгорифма (например, для выбора шага интегрирования и т. п.). Контрольные просчеты при исходных данных, соответствующих имеющимся экспериментальным результатам, можно применить для уточнения математической постановки задачи.

Только после откладки программы на машине (а иногда и после проведения контрольных просчетов) задача может быть автоматически решена на машине. Однако при решении задачи из-за случайных сбоев в работе машины могут получаться неправильные результаты. Для исключения возможности получения неверных результатов необходимо предусматривать либо приборный, либо программный контроль правильности работы машины.

§ 2. Разработка решающего алгорифма. Задание функций

1. Учет особенностей электронной цифровой машины. Применяя электронную цифровую машину для решения математических задач, необходимо учитывать следующие ее особенности:

а) большое, но ограниченное количество цифр в изображениях чисел;

б) большую скорость выполнения операций над числами, хранящимися в оперативной памяти;

в) сравнительно малую скорость ввода исходных данных и вывода результатов;

г) сравнительно малую скорость обмена числами между оперативной памятью и внешними запоминающими устройствами;

д) небольшую емкость оперативной памяти при весьма большой емкости внешних запоминающих устройств;

е) возможность случайных сбоев в работе машины и вытекающую отсюда необходимость контроля работы машины.

Медленность ввода и вывода делает нецелесообразным применение неспециализированных программно-управляемых машин для решения задач, связанных с вводом очень больших количеств исходной информации, выдачей больших количеств результатов и выполнением малого количества операций. Для таких задач время их решения определяется уже не быстродействием машины при выполнении операций, а временем, необходимым на ввод и вывод. В ряде случаев неспециализированные машины непригодны и для решения задач, связанных с необходимостью хранения очень больших количеств информации (превышающих возможности внешних запоминающих устройств машины).

Небольшая скорость обмена числами между оперативной памятью и внешним запоминающим устройством и большое быстродействие машины заставляют стремиться к тому, чтобы программа и промежуточные результаты помещались в оперативной памяти машины и во всяком случае чтобы количество передач из оперативной памяти во внешние запоминающие устройства и обратно было минимальным. Другими словами, программа должна быть по возможности «короткой» (содержать мало команд), а количество промежуточных результатов небольшим. Количество чисел, сохранение которых

необходимо при переходе от одной операции, предусмотренной в решающем алгорифме, к другой операции, характеризует так называемую *связность* алгорифма. Таким образом, решающий алгорифм не должен обладать большой связностью.

Условия задачи, предназначенной для решения на программно-управляемой машине, всегда содержат указания о требуемой точности решения. Обычно задается значение максимальной допустимой погрешности. Только тот решающий алгорифм пригоден, который позволяет получить решение задачи с погрешностью, не превышающей допустимую.

Большое количество разрядов при изображении чисел в машине позволяет производить вычисления с большой точностью. Если, однако, оказывается, что погрешность, допускаемая при вычислениях, слишком велика, то большое быстродействие машины иногда позволяет повысить точность результатов за счет увеличения количества машинных операций. В итерационных методах можно предусмотреть большее количество итераций, а в неитерационных предусмотреть вычисления над числами с количеством разрядов большим, чем разрядность ячейки памяти. При этом, однако, увеличивается количество ячеек, необходимых для хранения исходных данных и промежуточных результатов, и значительно возрастает время, потребное для решения задачи.

2. Учет стоимости решения и расхода времени. При разработке решающего алгорифма необходимо учитывать стоимость всех работ, связанных с решением задачи, и общий расход времени на ее решение. Если решение задачи должно быть осуществлено в кратчайший срок, то ради сокращения общего расхода времени идут даже на увеличение так называемого машинного времени. В этом случае стараются использовать один из уже известных решающих алгорифмов, по возможности простой, не требующий большого времени на программирование.

Если решение задачи не является срочным, то на первый план выступают соображения экономии материальных затрат. Например, для решения многовариантных задач целесообразно особенно тщательно разработать решающий алгорифм и программу, не жалея для этого времени, с тем чтобы по возможности сократить расход дорогостоящего машинного

времени. При ориентировочной прикидке материальных затрат можно за единицу принять стоимость рабочего дня техника-программиста. При этом стоимость рабочего дня младшего научного сотрудника-программиста составит 1,5, стоимость рабочего дня старшего научного сотрудника, разрабатывающего решающий алгорифм, — около 3, стоимость одного часа машинного времени для малой машины — примерно 3, а для средней — приблизительно 8—9. Стоимость машинного времени приведена здесь весьма ориентировочно. Для нее характерна резко выраженная тенденция убывать по мере прогресса электронных программно-управляемых машин.

При разработке программ для автоматического перевода текстов, автоматического выполнения информационных работ, автоматического управления объектами (например, производственными) обычно стоимость работ по разработке решающего алгорифма, программированию и отладке программ на машине складывают со стоимостью самой машины. Стоимость работ по эксплуатации машины и решению задачи в единицу времени отнимают от стоимости материальных выгод, получаемых в единицу времени за счет применения машины и программ. На основании этих данных подсчитывают время, в течение которого окупится машина вместе с программами. Машина и программы должны быть такими, чтобы указанное время было минимальным. Конечно, эти соображения принимаются в расчет в тех случаях, когда не превалируют другие.

3. Способы задания функций. При решении многих задач бывает необходимо пользоваться значениями различных функций, например тригонометрических и других. В большинстве машин значения функций получают с помощью программ, помещаемых в оперативную память. В других машинах (например, в машине *Стрела*) имеется так называемый накопитель стандартных подпрограмм, в котором постоянно закоммутированы подпрограммы для вычисления ряда элементарных функций.

Для получения значений функций применяют различные виды их приближенного задания: функция может быть задана своим дифференциальным уравнением, разложением в степенной или какой-нибудь другой ряд, разложением в бесконечную дробь; в некоторых случаях функцию на отдельных

интервалах изменения ее аргумента задают в виде достаточно близких к ней многочленов.

Весьма широко применяется также табличный способ задания функций. Этот способ связан с хранением в памяти машины большого количества чисел и с существенным расходом машинного времени. В литературе (см., например, [37]) описываются различные приемы «сжатия» таблиц и рационального поиска в них.

4. Масштабирование. Предположим, что решающий алгорифм требует выполнения предусмотренных в машине операций над исходными данными $x_1, x_2, \ldots, x_m$ и получения результатов $y_1, y_2, \ldots, y_{n-m}$, причем некоторые из этих величин или некоторые промежуточные результаты, возникающие при реализации решающего алгорифма, выходят за пределы диапазона чисел, представимых в машине. В этом случае прибегают к приему, известному под названием «масштабирование». Необходимость в масштабировании может возникнуть при пользовании любой машиной, но для машин с фиксированной запятой она возникает, как правило. В дальнейшем масштабирование рассматривается применительно к машинам с фиксированной запятой.

Введем обозначения

$$\alpha_i = \begin{cases} x_i & \text{при } i \leqslant m, \\ y_{i-m} & \text{при } i \geqslant m+1. \end{cases}$$

Сущность масштабирования заключается в том, что величины α_i представляют в виде

$$\alpha_i = \bar{\alpha}_i A_i,$$

где A_i — так называемые *масштабные коэффициенты*. Затем преобразуют решающий алгорифм так, чтобы для него исходными данными были $\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_m$, а искомыми результатами $\bar{\alpha}_{m+1}, \bar{\alpha}_{m+2}, \ldots, \bar{\alpha}_n$ и чтобы ни эти величины, ни промежуточные результаты решающего алгорифма не выходили за пределы диапазона чисел, представимых в машине. Иногда это удается осуществить, задаваясь постоянными значениями для масштабных коэффициентов A_i ($i = 1, 2, \ldots, n$). В этом случае говорят, что для решения задачи использован *метод постоянных масштабов*.

Обычно в качестве масштабных коэффициентов в методе постоянных масштабов применяют целые степени числа десять (*десятичные* масштабы) или целые степени числа два (*двоичные* масштабы). Десятичные масштабы позволяют весьма просто по исходным данным $\alpha_i\,(i=1, 2, \ldots, m)$ определять $\bar{\alpha}_i$ и по полученным результатам $\bar{\alpha}_j\,(j=m+1, m+2, \ldots, n)$ вычислять искомые величины α_j путем простого переноса запятой в десятичной записи этих величин. Двоичные масштабы в этом отношении менее удобны, но их применение позволяет осуществить решение с меньшими погрешностями, чем при десятичных масштабах.

Погрешности, обусловленные применением метода постоянных масштабов, связаны с тем, что в ячейках памяти машины с фиксированной запятой числа $\bar{\alpha}_i$ и промежуточные результаты могут быть представлены лишь с определенным количеством разрядов. Если при этом старшие разряды оказываются равными нулю, то уменьшается количество значащих цифр в изображении этих величин. Уменьшая масштабные коэффициенты, можно добиться того, чтобы старшие разряды числа $\bar{\alpha}_i$ и промежуточных результатов не были нулями. Но при этом некоторые из величин $\bar{\alpha}_i$ или промежуточных результатов могут выйти за пределы допустимого диапазона. Так как целые степени числа два при увеличении показателя степени возрастают медленнее, чем целые степени числа десять, то двоичные масштабы можно подобрать с большей выгодой, чем десятичные.

Пример 3.1. Предположим, что решающий алгорифм сводится к выполнению единственной операции деления

$$\alpha_3=\frac{\alpha_2}{\alpha_1}$$

(α_1, α_2 — исходные данные, α_3 — искомый результат), причем $10\leqslant\alpha_1\leqslant 100$, $20\leqslant\alpha_2\leqslant 50$.

Пусть, кроме того, для решения задачи применяется машина с запятой, фиксированной перед старшим разрядом.

Без труда получаем $1/5\leqslant\alpha_3\leqslant 5$.

Полагая

$$\alpha_i=\bar{\alpha}_i A_i \qquad (i=1, 2, 3),$$

приводим решающий алгорифм к виду

$$\bar{\alpha}_3=\frac{\dfrac{A_2}{A_1\cdot A_3}\,\bar{\alpha}_2}{\bar{\alpha}_3}.$$

Очевидно, должны выполняться условия

$$A_1 > 100,$$
$$A_2 > 50,$$
$$A_3 > 5.$$

Потребуем, кроме того, чтобы было

$$\frac{A_2}{A_1 \cdot A_3} < 1.$$

Все перечисленные условия выполнены, если

$$A_1 = 128 = 2^7,$$
$$A_2 = 64 = 2^6,$$
$$A_3 = 8 = 2^3.$$

При этом

$$\frac{A_2}{A_1 \cdot A_3} = \frac{1}{2^4} = 0{,}0625.$$

Решающий алгорифм имеет теперь вид

$$\bar{\alpha}_3 = \frac{(0{,}0625 \cdot \bar{\alpha}_2)}{\bar{\alpha}_1}.$$

После вычисления $\bar{\alpha}_3$ искомую величину α_3 мы получим по формуле

$$\alpha_3 = \overline{8\alpha_3}.$$

Иногда не удается без допущения чрезмерных погрешностей осуществить масштабирование, пользуясь постоянными масштабными коэффициентами. Однако может оказаться, что решающий алгорифм поддается расчленению на части, каждая из которых позволяет выбрать для себя постоянные масштабные коэффициенты. Для каждой части решающего алгорифма задается свой набор масштабных коэффициентов, а для соединения частей алгорифма в одно целое добавляют к нему участки, обеспечивающие переход от одних наборов масштабных коэффициентов к другим. При этом говорят, что задача решается *методом переменных масштабов*.

Метод переменных масштабов позволяет уменьшить погрешности вычислений (по сравнению с методом постоянных масштабов), однако усложняет решающий алгорифм и программу.

Метод постоянных масштабов связан с трудностью определения диапазонов изменения не только исходных данных и искомых результатов, но и всех промежуточных результатов. В методе переменных масштабов кроме того мы сталкиваемся с трудностью заблаговременного определения поведения некоторых величин. Но анализ поведения величин и выбор новых значений масштабных коэффициентов можно поручить машине, составив программу таким образом, чтобы машина сама определяла масштабные коэффициенты для каждого этапа вычислений. Такой метод решения задач в режиме с фиксированной запятой называется *методом плавающих масштабов* [66]. Этот метод был разработан в вычислительном центре МГУ. Сущность его для машин с запятой, фиксированной перед старшим разрядом, заключается в следующем.

Часть величин, участвующих в решении, представляют в виде

$$\alpha_i = \bar{\alpha}_i \cdot 2^{p_i},$$

где p_i — целые числа, а $\bar{\alpha}_i$ удовлетворяет неравенству

$$|\bar{\alpha}_i| < 1.$$

Масштабными множителями являются величины 2^{p_i}. Величины α_i хранятся каждая в двух ячейках; одна ячейка отводится для $\bar{\alpha}_i$, а другая для p_i. Решение задачи разбивается на этапы, в каждом из которых масштабы остаются постоянными, а при переходе от одного из них к другому определенная часть программы исследует величины $\bar{\alpha}_i$ и при необходимости изменяет некоторые из чисел p_i.

Если предусмотреть в программе возможность изменения масштабов при выполнении каждой операции, то этот метод сведется к *программному введению плавающей запятой.*

При методе плавающих масштабов масштабированию могут подвергаться не все участвующие в вычислениях величины, а лишь те, для которых это необходимо. Для величин, не подлежащих масштабированию, отводится не по две, а по одной ячейке.

Возможности некоторых машин позволяют переходить к пересчету масштабов автоматически, например по сигналу переполнения разрядной сетки.

§ 3. Методы программного контроля. Отладка программы на машине

1. Программный контроль. Сбои в работе машины делятся на систематические и случайные. Наличие систематических сбоев исключает возможность решения задач на машине и является признаком ее неисправности. Возможность случайных сбоев не препятствует решению задач, но требует контроля правильности работы машины.

В некоторых машинах имеются специальные устройства, контролирующие правильность работы отдельных узлов машин. Это так называемый *приборный контроль*, предусмотренный конструкцией машины. Мы же остановимся на некоторых способах *программного контроля*. Одним из весьма распространенных способов программного контроля является так называемый *двойной счет с контрольным суммированием*. Задача решается дважды, и каждый раз результаты суммируются. Равенство контрольных сумм считают признаком правильности работы машины. Двукратный счет, контрольное суммирование и сравнение контрольных сумм предусматривают в программе.

Ввиду того, что случайные сбои могут привести не только к ошибочным результатам, но и к искажению программы, практикуют также контрольное суммирование команд.

Метод двойного счета, как показывает практика, весьма надежен, но увеличивает вдвое расход машинного времени.

При решении некоторых задач применяется для контроля *метод контрольного соотношения*. Существо этого метода сводится к следующему. Если искомые результаты должны удовлетворять некоторому соотношению, которое не используется для их получения, то в программе предусматривают проверку выполнения упомянутого соотношения. Если соотношение выполнено с достаточной степенью точности, то результаты считаются правильными. Например, при вычислении таблицы значений $\sin x$ и $\cos x$ без применения формулы

$$\sin^2 x + \cos^2 x = 1$$

эта формула может быть принята за контрольное соотношение. Задаются достаточно малым числом $\varepsilon > 0$ и заменяют последнюю формулу неравенством

$$|1 - \sin^2 x - \cos^2 x| - |\varepsilon| < 0.$$

Замена равенства неравенством обусловлена тем, что вычисления осуществляются машиной хотя и с малой погрешностью, но приближенно.

В некоторых случаях применяют разновидность метода контрольного соотношения, которую называют *методом контрольной переменной.* Существо этого метода состоит в том, что одновременно с искомыми величинами вычисляют некоторую, не нужную для решения задачи величину, удовлетворяющую вместе с искомыми величинами какому-нибудь заранее известному соотношению, используемому как контрольное соотношение.

Наконец, в некоторых случаях применяют *метод подстановки,* также являющийся частным случаем метода контрольного соотношения. Этот метод можно применять при решении уравнений или дифференциальных уравнений и сводится он, по существу, к тому, что в качестве контрольных соотношений применяются сами решаемые уравнения.

2. Отладка программы. Отладка программы на машине, преследуя цель устранения в программе неправильностей, объединяет в себе три мероприятия: а) контроль программы, б) поиск и определение содержания (диагностику) ошибок, в) исправление обнаруженных ошибок.

Отладку программы на машине обычно производят в два этапа: первый называется *автономной,* а второй — *комплексной* отладкой. При автономной отладке программу разбивают на части, по возможности мало связанные между собой. Для каждой части в отдельности проверяют правильность выполняемых ею вычислений (арифметическая проверка), правильность передачи управления командами условного перехода (логическая проверка), правильность выполнения варьирующих команд (команд, изменяющих другие команды) и т. д. Для арифметической проверки каждой части подбирают для нее исходные данные и путем вычислений вручную получают результаты, отвечающие этим частным исходным данным. Затем вводят программу в память машины и выполняют ее по частям, пользуясь тем или иным специальным режимом работы машины или специальными отладочными программами (см. гл. 7, § 1, п. 3), позволяющими в нужных местах прерывать выполнение отлаживаемой программы, вводить с пульта или другим способом исходные данные, проверять частные

результаты, сравнивая их с заготовленными, проверять результаты выполнения команд, изменяющих другие команды, или вносить в программу исправления и изменения.

После того как в отлаживаемой программе устранены ошибки, обнаруженные при автономной отладке, производят комплексную отладку: с помощью отлаживаемой программы решают несколько примеров, при которых отдельные ее части работают в комплексе. В результате отладки программы на машине можно убедиться, что программа соответствует той логической схеме (см. гл. 4), по которой она составлена. В некоторых случаях может быть установлено, что при переходе от решающего алгорифма к логической схеме были допущены ошибки или что точность, принятая для вычисления промежуточных результатов, недостаточна (об этом можно судить по грубым отклонениям результатов, получающихся при комплексной отладке, от ожидаемых). Наконец, если при комплексной отладке применяются для сравнения результаты, полученные каким-либо способом, отличным от решающего алгорифма, реализованного в программе, то могут быть выявлены и ошибки, допущенные разработчиком решающего алгорифма.

Трудности отладки программы на машине уменьшаются, если отлаживаемая программа составлена с помощью программирующей программы или с применением стандартных подпрограмм (см. гл. 7).

§ 4. Организация программ

1. Понятие организации программы. Из предыдущего видно, что, кроме реализации алгорифма, равнозначного решающему, программа должна предусматривать контроль правильности вычислений, а в ряде случаев — получение значений различных функций. Весьма часто программу приходится делить на куски и размещать не только в оперативной памяти, но и во внешних запоминающих устройствах (например, на магнитных лентах). При этом в программу приходится включать специальные участки, которые обеспечивают правильный обмен числами и частями программы между различными запоминающими устройствами машины. То есть, как говорят, организация программы может быть весьма сложной. Под *организацией программы* понимают

совокупность крупных функционально различных ее частей и взаимодействие между ними. Элементами организации считают:

ввод и вывод программы и исходных данных;

запись на магнитную ленту и считывание с нее;

запись на магнитный барабан и считывание с него;

контрольное суммирование программы и числового материала;

перенос кодов из одной части оперативной памяти в другую;

обращение к некоторой части программы, рабочий счет, останов машины и др.

2. Схема организации. При программировании производится разработка организации программы и составляется схема организации, что облегчает дальнейшее программирование. Например, организация программы, предусматривающей контроль методом двойного счета, может изображаться следующей схемой:

$$VR\Sigma_1 E(R)\,\Sigma_2\; \underset{\uparrow}{P(\sigma_1 = \sigma_2)}\; W''\mathit{Я}\mathit{Я}'.$$

Описанная организация предусматривает ввод (V) исходных данных и программы, рабочий счет (R), первое контрольное суммирование (Σ_1), повторный счет путем выполнения уже один раз использованной части программы ($E(R)$), второе контрольное суммирование (Σ_2), сравнение двух контрольных сумм ($P(\sigma_1 = \sigma_2)$), выдачу результатов (W'') при равенстве контрольных сумм и останов ($\mathit{Я}$). При несовпадении контрольных сумм предусмотрен останов ($\mathit{Я}'$), переход к которому указан с помощью стрелки, расположенной в схеме организации ниже строки. Правила чтения схем организации весьма просты и во многом сходны с правилами чтения логических схем, описанных ниже.

Описанная приведенной выше схемой организация программы может быть применена при самых различных решающих алгорифмах. Хотя организация программы и зависит от особенностей решающего алгорифма, но не определяется им однозначно.

Литература к главе 3: [5], [8], [9], [10], [23], [37].

ГЛАВА 4

ОПЕРАТОРНЫЙ МЕТОД ПРОГРАММИРОВАНИЯ

§ 1. Простейшее программирование

1. Связь между распределением памяти и составлением команд. В период появления первых программно-управляемых машин, обладающих еще небольшими возможностями, методов программирования не существовало. Составление программ носило характер решения сложных комбинаторных задач. Уже тогда заметили, что программирование объединяет две работы: распределение памяти машины и составление команд. Распределение памяти машины, то есть размещение в запоминающих устройствах материала, относящегося к решению задачи (исходных данных, команд, вспомогательных кодов, промежуточных и окончательных результатов), тесно связано с составлением команд. Не зная номеров ячеек, хранящих исходные данные и вспомогательные коды, не определив, куда нужно помещать результаты, нельзя составлять команды программы. С другой стороны, не зная заранее количества команд программы, а также количества промежуточных результатов, которые должны одновременно находиться в памяти, трудно произвести размещение материала в запоминающих устройствах машины.

2. Символические адреса. Первым усовершенствованием процесса программирования явился следующий прием (известный под названием метода символических адресов), позволивший выполнять эти работы раздельно.

Разбивают последовательность M всех ячеек оперативной памяти на массивы (подпоследовательности) M_1, M_2, ..., M_i, ... Ячейкам массива M_i приписывают обозначения $\alpha_i + 1$, $\alpha_i + 2$, $\alpha_i + 3$, ..., где α_i $(i = 1, 2, \ldots)$ — буквы

какого-либо, например латинского, алфавита. Коды $\alpha_i + 1$, $\alpha_i + 2$, ... ($i = 1, 2, \ldots$) называются *символическими адресами* или *буквенно-числовыми обозначениями ячеек.* (Можно считать, например, что исходные данные расположены в ячейках, например $b + 1$, $b + 2$, ..., промежуточные результаты — в ячейках $c + 1$, $c + 2$, ..., программа занимает ячейки $a + 1$, $a + 2$, ... и т. п.)

Пользуясь символическими адресами, программу составляют непосредственно по формулам, входящим в решающий алгорифм. При этом команды составленной программы вместо истинных адресов содержат символические адреса. После того как программа в символических адресах составлена, определяют количества ячеек, входящих в массивы M_1, M_2, ..., M_i, ... Распределение памяти осуществляют, приписывая буквам α_1, α_2, ..., α_i, ... определенные числовые значения. Для получения окончательного вида программы производят замену букв в символических адресах команд назначенными для них значениями. Последний процесс имеет «механический» характер и называется *присвоением истинных адресов.*

Применение символических адресов дает следующие преимущества:

1. Составление команд и распределение памяти выполняются раздельно, что значительно облегчает труд программиста.

2. Возможно успешное разделение труда между квалифицированным программистом (составляющим программу в символических адресах и выполняющим распределение памяти) и группой менее квалифицированных работников, на которых возлагается присвоение истинных адресов.

3. Облегчается процесс составления довольно часто встречающихся в программах команд, предусматривающих модификацию других команд.

4. Процесс присвоения истинных адресов может быть автоматизирован, т. е. по специальной программе, составленной раз и навсегда, может выполняться самой машиной.

Пример 4.1. Требуется составить программу решения на машине *Стрела* системы линейных алгебраических уравнений

$$Ax + By = C,$$
$$Dx + Ey = F,$$

удовлетворяющей условию

$$AE - BD \neq 0.$$

Решающий алгорифм сводится к вычислениям по формулам

$$x = \frac{CE - BF}{AE - BD},$$

$$y = \frac{AF - CD}{AE - BD}.$$

Предположим, что числа должны быть размещены в ячейках памяти следующим образом:

$$b+1)\ A;\quad b+2)\ B;\quad b+3)\ C;$$
$$b+4)\ D;\quad b+5)\ E;\quad b+6)\ F.$$

Здесь слева от закрытой круглой скобки в каждой записи стоит символический номер ячейки, а справа от скобки содержимое этой ячейки.

Для команд программы отведем ячейки с номерами $a+1$, $a+2$, ..., для промежуточных результатов $c+1$, $c+2$, $c+3$, для ответов $d+1$ и $d+2$.

Команда ввода с перфокарт шести исходных чисел будет

$$a+1)\ 0000\ \ 0005\ \ b+1\ \ 0\ \ 41$$

Перевод из десятичной системы счисления в двоичную исходных данных с записью результатов в те же ячейки будет выполнен по команде

$$a+2)\ \ b+1\ \ 0005\ \ b+1\ \ 0\ \ 72.$$

Вычисление величин x и y машина произведет по командам

$a+3)$	$b+1$	$b+5$	$c+1$	0	05	вычисление $A \cdot E$,
$a+4)$	$b+2$	$b+4$	$c+2$	0	05	вычисление $B \cdot D$,
$a+5)$	$c+1$	$c+2$	$c+1$	0	03	вычисление $AE - BD$,
$a+6)$	$c+1$	0000	$c+1$	0	62	вычисление $\frac{1}{AE - BD}$,
$a+7)$	$b+3$	$b+5$	$c+2$	0	05	вычисление $C \cdot E$,
$a+10)$	$b+2$	$b+6$	$c+3$	0	05	вычисление BF,
$a+11)$	$c+2$	$c+3$	$c+2$	0	03	вычисление $CE - BF$,
$a+12)$	$c+2$	$c+1$	$d+1$	0	05	вычисление x,
$a+13)$	$b+1$	$b+6$	$c+2$	0	05	вычисление AF,
$a+14)$	$b+3$	$b+4$	$c+3$	0	05	вычисление CD,
$a+15)$	$c+2$	$c+3$	$c+2$	0	03	вычисление $AF - CD$,
$a+16)$	$c+2$	$c+1$	$d+2$	0	05	вычисление y,
$a+17)$	$d+1$	0001	$d+1$	0	70	перевод результатов в десятичную систему счисления.

Для контроля правильности результатов предусмотрим двойной счет и контрольное суммирование содержимого всех ячеек, хранящих материал, относящийся к решению задачи (кроме самих

контрольных сумм). При этом остановимся на организации программы, предусматривающей:

(1) ввод в оперативную память коэффициентов уравнений и перевод их в двоичную систему счисления;

(2) решение системы уравнений и перевод результатов в десятичную систему счисления;

(3) контрольное суммирование;

(4) проверку: первая или вторая контрольная сумма вычислена;

(5) если вычислена первая сумма, то производится перенос ее в новую ячейку и переход к повторному выполнению этапов (2), (3) и (4);

(6) если вычислена вторая сумма, то производится сравнение контрольных сумм;

(7) если контрольные суммы равны, то производится выдача результатов и останов;

(8) если суммы не равны, то производится останов без выдачи результатов.

Для обеспечения описанной организации программы составляем следующие команды:

$a+20$)	0000	0045	$e+1$	0	34	контрольное суммирование
$a+21$)	$a+1$	$e+1$	$e+1$	0	17	
$a+22$)	$g+1$	0000	0000	0	16	проверка: первая или вторая контрольная сумма получена
$a+23$)	$a+24$	$a+26$	0000	0	20	переход после первого контрольного суммирования к команде $a+24$, а после второго — к команде $a+26$
$a+24$)	$e+1$	0000	$g+1$	0	13	перенос первой контрольной суммы в ячейку $g+1$
$a+25$)	$a+3$	$a+3$	0000	0	20	переход к рабочей и суммирующей части программы для повторного счета
$a+26$)	$e+1$	$g+1$	0000	0	16	сравнение контрольных сумм
$a+27$)	$a+32$	$a+30$	0000	0	20	переход при равенстве контрольных сумм к команде выдачи результатов, а при неравенстве — к команде останова
$a+30$)	$g+1$	$e+1$	0000	0	40	останов при неравенстве контрольных сумм
$a+31$)	$a+30$	$a+30$	0000	0	20	команда, страхующая от выдачи на печать неверных результатов в случае нажатия после останова на пульте управления кнопки «пуск»
$a+32$)	$d+1$	0001	0000	0	44	выдача результатов
$a+33$)	0000	0000	0000	0	40	останов после окончания решения задачи

Во втором адресе команды, расположенной в ячейке $a+20$, стоит 0045. Это число представляет собой $n-1$, где n — количество ячеек, содержимое которых должно быть просуммировано

(n подсчитано после составления всех команд). Команда, расположенная в ячейке $a+30$, включена в программу в связи с тем, что во многих программах в случае неравенства контрольных сумм разрешается нажать на пульте управления кнопку «пуск», после чего происходит третий просчет. В нашей программе третий просчет не предусмотрен. При решении задачи на машине лицом, не знающим нашей программы, нажим кнопки «пуск» при отсутствии страхующей команды привел бы к выдаче на печать неправильных результатов.

Допустим, что первую команду программы мы решили разместить в ячейке с номером 0020. Поэтому, распределяя память, положим

$$a = 0017,$$

затем

$$\begin{aligned} b &= a + 33 = 0052, \\ c &= b + 6 = 0060, \\ d &= c + 3 = 0063, \\ e &= d + 2 = 0065. \end{aligned}$$

Ячейку $g+1$, в которую производится перенос первой контрольной суммы, разместим непосредственно перед первой командой

Таблица 4.1. Программа решения на машине *Стрела* двух линейных алгебраических уравнений с двумя неизвестными

Номер ячейки	Команда или число	Номер ячейки	Команда или число
0017	0000 0000 0000 0 00	0043	0066 0000 0017 0 13
0020	0000 0005 0053 0 41	0044	0022 0022 0000 0 20
0021	0053 0005 0053 0 72	0045	0066 0017 0000 0 16
0022	0053 0057 0061 0 05	0046	0051 0047 0000 0 20
0023	0054 0056 0062 0 05	0047	0017 0066 0000 0 40
0024	0061 0062 0061 0 03	0050	0047 0047 0000 0 20
0025	0061 0000 0061 0 62	0051	0064 0001 0000 0 44
0026	0055 0057 0062 0 05	0052	0000 0000 0000 0 40
0027	0054 0060 0063 0 05	0053	. . . . . A
0030	0062 0063 0062 0 03	0054	. . . . . B
0031	0062 0061 0064 0 05	0055	 C
0032	0053 0060 0062 0 05	0056	. . . . . D
0033	0055 0056 0063 0 05	0057	. . . . . E
0034	0062 0063 0062 0 03	0060	. . . . . F
0035	0062 0061 0065 0 05	0061	} Промежуточные результаты
0036	0064 0001 0064 0 70	0062	
0037	0000 0045 0066 0 34	0063	
0040	0020 0066 0066 0 17	0064	 x
0041	0017 0000 0000 0 16	0065	. . . . . y
0042	0043 0045 0000 0 20	0066	Ячейка для получения контрольной суммы

программы и сделаем ее начальное содержимое равным нулю. Такое размещение ячейки $g+1$ делает удобной запись в нее нуля одновременно с вводом программы. Заметим, что неравенство нулю содержимого этой ячейки служит признаком того, что первая контрольная сумма уже получена, т. е. что первый счет уже произведен.

Итак, положим $g=0016$.

Выбрав значения букв a, b, c, d, e, g, мы тем самым произвели распределение памяти. После присвоения истинных адресов получим программу, приведенную в табл. 4.1.

§ 2. Операторный метод программирования

1. Схема решающего алгорифма. По мере усложнения задач трудности программирования резко возрастают. Возникает вопрос, нельзя ли решающий алгорифм расчленить на части, допускающие раздельное программирование, а программу решения задачи получать соединением частных программ.

Предположим, что решающий алгорифм является совокупностью элементарных предписаний (см. § 4, гл. 1), удовлетворяющих следующим двум условиям:

1) каждое элементарное предписание определяет либо нулевой процесс перехода, либо процесс перехода к одному определенному элементарному предписанию, либо процесс перехода к одному из двух возможных элементарных предписаний (процесс условного перехода);

2) элементарное предписание, определяющее нулевой или условный переход, определяет нулевой основной процесс.

Встречающиеся на практике решающие алгорифмы обычно либо удовлетворяют, либо легко сводятся к алгорифмам, удовлетворяющим перечисленным условиям. Дискретный характер позволяет их расчленять на части. Элементарные предписания, определяющие условный переход, можно формулировать примерно так: «если выполнено условие ϕ, то перейти к такому-то элементарному предписанию, а в противном случае — к такому-то». Про такое элементарное предписание говорят, что оно проверяет выполнение некоторого условия ϕ или значение некоторого предиката с содержанием ϕ.

Совокупность нескольких элементарных предписаний решающего алгорифма будем называть *элементарным оператором,* если:

1) эта совокупность имеет только один *вход* (т. е. одно элементарное предписание, к которому может быть произ-

веден переход извне, от элементарных предписаний, не принадлежащих этой совокупности) и только один *выход* (т. е. одно элементарное предписание, предусматривающее переход вовне, к элементарным предписаниям, не принадлежащим этой совокупности; выход определяет процесс перехода только вовне);

2) эта совокупность *связна* (т. е. если выполняется вход, то после конечного числа актов будут выполнены все входящие в нее элементарные предписания);

3) эта совокупность *линейна* (т. е. каждое ее элементарное предписание, кроме, быть может, выхода, определяет процесс перехода только к одному единственному элементарному предписанию).

Первое элементарное предписание алгорифма считается входом, а элементарное предписание, определяющее нулевой процесс перехода, — выходом того элементарного оператора, которому оно принадлежит.

Возможность рассматривать дискретный алгорифм, являющийся совокупностью элементарных предписаний, как совокупность элементарных операторов, следует из того, что каждое элементарное предписание можно считать оператором.

Пусть M — совокупность нескольких элементарных операторов, входящих в состав алгорифма. Условимся говорить, что элементарный оператор, входящий в M, имеет *внешний вход*, если его вход является первым элементарным предписанием алгорифма или если к нему возможен переход извне, т. е. от элементарного оператора, не принадлежащего M. Если же к данному элементарному оператору предусмотрен переход только от элементарных операторов совокупности M, то будем говорить, что он имеет *внутренний вход*. Кроме того, будем говорить, что элементарный оператор, входящий в M, имеет *внутренний выход*, если от него предусмотрен переход только к другим элементарным операторам, входящим в M. В остальных случаях будем говорить, что упомянутый элементарный оператор имеет *внешний выход*.

Совокупность нескольких элементарных операторов называется *обобщенным оператором*, если: 1) один из входящих в нее элементарных операторов имеет внешний вход, а остальные имеют внутренние входы; 2) каждый элементарный оператор, имеющий внешний выход, является формально связным с элементарным оператором, имеющим внешний вход.

Элементарные операторы и обобщенные операторы будем называть *операторами.*

При разбиении алгорифма на элементарные операторы условимся элементарное предписание, определяющее нулевые основной процесс и процесс перехода, принимать за самостоятельный оператор, который будем называть *оператором конца* и обозначать символом $Я_\nu$.

Элементарные предписания, определяющие условный переход, также условимся считать самостоятельными операторами, которые будем называть *логическими* и обозначать символами вида P_ν. Остальные операторы будем обозначать символами вида Q_ν. Во всех перечисленных обозначениях индекс ν является натуральным числом. Символы, обозначающие операторы, в тех случаях, когда это не ведет к недоразумениям, будем для краткости называть операторами. Кроме вышеописанных символов, введем еще символ $И_0$, который будем называть *оператором начала.*

Предположим, что задан некоторый решающий алгорифм, являющийся совокупностью элементарных предписаний и удовлетворяющий вышеперечисленным условиям. Объединим его элементарные предписания в элементарные операторы и перенумеруем их в каком-либо порядке. Каждому оператору, в зависимости от его особенностей, поставим в соответствие один из символов Q_ν, $Я_\nu$ или P_ν, полагая ν равным номеру оператора.

Затем выпишем строку символов, в которой на первом (самом левом) месте записан оператор начала, а затем в порядке номеров выписаны (слева направо) все элементарные операторы решающего алгорифма.

Затем от оператора $И_0$ проведем стрелку к оператору, которому принадлежит первое элементарное предписание решающего алгорифма и от каждого оператора Q_ν проведем стрелку к тому оператору, к которому происходит переход от него. От каждого оператора P_ν проведем две такие стрелки, причем стрелку, отвечающую переходу при выполнении условия, проверяемого логическим оператором, расположим над строкой, а стрелку, отвечающую переходу в случае невыполнения упомянутого условия, расположим под строкой. Стрелки, имеющие направление слева направо и соединяющие два записанных рядом оператора, будем опускать. Чтобы подчеркнуть отсутствие перехода от оператора,

стоящего слева, к оператору, стоящему справа непосредственно рядом с ним, можно между этими операторами ставить какой-либо разделительный знак, например точку с запятой.

Полученная строка операторов вместе со стрелками, обозначающими переходы от оператора к оператору, называется *схемой* решающего алгорифма.

Пример 4.2. Решающий алгорифм, определяющий процесс вычисления интеграла

$$I=\int_1^a x^m dx \qquad (a>0),$$

состоит из следующих элементарных предписаний:

1°. Проверить, справедливо ли равенство

$$m=-1.$$

Если да, перейти к п. 2°, если нет, — к п. 3°.

2°. Вычислить I по формуле

$$I=\ln a.$$

Перейти к п. 3°.

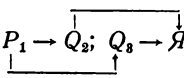
3°. Вычислить I по формуле

$$I=\frac{a^{m+1}-1}{m+1}.$$

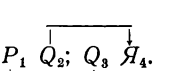
Перейти к п. 4°.

4°. Прекратить процесс.

Принимая каждый пункт этого алгорифма за оператор, получаем схему описанного алгорифма



Опуская стрелки между $И_0$ и P_1, P_1 и Q_2, а также между Q_3 и $Я_4$, окончательно имеем



В схеме решающего алгорифма оператор начала служит только для указания того оператора, который содержит первое элементарное предписание алгорифма.

При записи схем решающих алгорифмов часто вместо стрелок применяют специальные символы, называемые *знаками перехода*. Знак $\lfloor_k$ называют правым знаком перехода, а

знаки $\underset{k}{\llcorner}$, $\overset{l}{\ulcorner}$ и $\underset{k}{\overset{l}{\llcorner}}$ — левыми знаками перехода (здесь k и l — целые неотрицательные числа). Замену стрелок в схеме знаками перехода осуществляют следующим образом.

Просматривают схему (например, слева направо) и каждый раз, когда встречают оператор, к которому ведет группа стрелок (в частности, одна стрелка), нумеруют эту группу. После того как нумерация групп стрелок окончена, *перед* каждым оператором, к которому ведет группа стрелок, т. е. непосредственно слева от него, ставят правый знак перехода с индексом, равным номеру группы стрелок. Затем *после* каждого оператора, от которого ведет стрелка, принадлежащая упомянутой группе, т. е. непосредственно справа от него, ставится левый знак перехода с тем же индексом. При этом после оператора Q_v ставится левый знак перехода вида $\underset{k}{\llcorner}$, а после оператора P_v в случае верхней стрелки ставится левый знак перехода вида $\overset{l}{\ulcorner}$, а в случае нижней стрелки — вида $\underset{k}{\llcorner}$. Если после оператора P_v окажется два левых знака перехода $\underset{k}{\llcorner}\,\overset{l}{\ulcorner}$ или $\overset{l}{\ulcorner}\,\underset{k}{\llcorner}$, то вместо этой пары ставят $\underset{k}{\overset{l}{\llcorner}}$. После этого все стрелки рассмотренной группы стирают.

При использовании описанных знаков перехода знак «точка с запятой» не применяется.

Пример 4.3. Схема алгорифма, приведенная в примере 4.2, при использовании описанных знаков перехода принимает вид

$$И_0 P_1 \underset{1}{\llcorner} A_2 \underset{2}{\llcorner}\underset{1}{\lrcorner} A_3 \underset{2}{\lrcorner} Я_4.$$

В ряде случаев операторы решающего алгорифма действуют над величинами, которые зависят от некоторых параметров (изменяемых другими операторами). Такие операторы называются *зависящими от параметров*; их символы снабжаются справа вверху набором индексов, обозначающих эти параметры. Операторы, изменяющие значения параметров, будем называть *варьирующими*.

Пример 4.4. Пусть решающий алгорифм определяет процесс составления таблицы значений функции $y=f(x)$ для $x=x_1$,

$x_2, \ldots, x_i, \ldots, x_n$, заданной соотношением

$$f(x)=\begin{cases} \varphi(x) & \text{при } x<x^*, \\ \varphi(x)+\theta(x) & \text{при } x\geqslant x^* \end{cases} \qquad (x^*=\text{const}),$$

и состоит из следующих элементарных предписаний:

1°. Положить $i=1$. Перейти к следующему пункту.

2°. Вычислить

$$a_i=\varphi(x_i).$$

Перейти к следующему пункту.

3°. Проверить, справедливо ли неравенство

$$x_i\geqslant x^*.$$

Если да, перейти к п. 4°, если нет — к п. 5°.

4°. Вычислить искомое значение функции

$$y_i=a_i+\theta(x_i).$$

Перейти к п. 6°.

5°. Положить

$$y_i=a_i.$$

Перейти к следующему пункту.

6°. Проверить, справедливо ли равенство

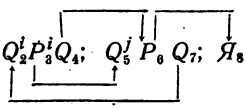
$$i=n.$$

Если да, то перейти к п. 8°, если нет — к следующему пункту.

7°. Увеличить значение i на 1. Перейти к п. 2°.

8°. Конец процесса.

Принимая пункты этого алгорифма за операторы, можем написать следующую схему:



или

$$И_0 Q_1 \lrcorner_4 Q_2^i P_3^i \llcorner_1 Q_4 \llcorner_2 \lrcorner_1 Q_5^i \lrcorner_2 P_6 \overset{3}{\ulcorner} Q_7 \llcorner_4 \lrcorner_3 Я_8.$$

Здесь оператор Q_1 задает начальное значение параметра i, а оператор Q_7 изменяет значение этого параметра. Операторы Q_2^i, P_3^i и Q_5^i зависят от параметра i. Оператор P_6 проверяет условие $i=n$, но сам от параметра i не зависит.

При построении схемы алгорифма можно его расчленить на обобщенные (а не элементарные) операторы. При этом целесообразно ввести три вида символов. Например, с помощью B_v обозначать операторы, имеющие выход только к одному оператору, с помощью C_v — операторы, выходами

которых являются элементарные предписания, определяющие нулевой процесс перехода, наконец, с помощью R_ν — операторы, имеющие выход к одному из двух возможных операторов. Кроме того, вводится символ $И_0$ — оператор начала. Построение схемы алгорифма аналогично описанному для случая элементарных операторов.

Применение обобщенных операторов делает схемы алгорифмов более короткими и обозримыми, но зато хуже передающими структуру алгорифма.

После расчленения решающего алгорифма на части (операторы), можно для каждой части составить программу, а затем частные программы объединить в программу решения задачи. Однако оказывается, что специфика программно-управляемой машины в ряде случаев требует включения в программу еще некоторых дополнительных участков или даже внесения изменений в схему решающего алгорифма (составленного без учета средств, с помощью которых должно быть осуществлено решение задачи), что связано, например, с необходимостью программного контроля правильности решения задачи и т. д.

2. Некоторые термины и обозначения. Если содержимое какой-либо ячейки обозначено некоторым символом, то саму ячейку (ее номер) часто обозначают тем же символом, заключенным в угловые скобки. Например, если N — некоторое число, то $\langle N\rangle$ — ячейка, хранящая число N. Если ячейка (ее номер) обозначена каким-либо символом, то содержимое этой ячейки удобно обозначать тем же символом, взятым в кавычки. Так, если $a+5$ — некоторая ячейка, то код, содержащийся в этой ячейке, обозначают как «$a+5$».

В тех случаях, когда нужно указать ячейку и ее содержимое, пишут номер ячейки, за ним правую круглую скобку (так называемую «закрытую»), а за ней — содержимое ячейки. Например, если в ячейке a хранится число N, то можно писать

$$a)\ N.$$

Принято говорить, что команда во время ее выполнения *управляет* машиной. Переход от одной команды к другой при этом называют *передачей управления*: первая из указанных команд *передает* управление, а вторая — его *получает*.

При выполнении программы одни ее команды могут видоизменяться другими. В этом случае говорят, что в процессе выполнения программа переходит из одного *состояния* в другое. Непосредственно перед пуском машины программа находится в своем *начальном* состоянии. Очевидно, программист составляет программу в ее начальном состоянии.

3. Элементарные операторы программы. Программам (их начальным состояниям) целесообразно придавать такую конструкцию, которая облегчала бы как *программирование*, так и *проверку* составленных программ, не понижая существенно их качества, т. е. не увеличивая заметно времени, необходимого для их выполнения машиной.

Процесс проверки будет значительно облегчен, если программа допускает обозримое описание, которое, будучи составлено до программирования, облегчает и сам процесс программирования.

Предъявим к конструкции программы следующие требования:

1. Программа должна состоять из частей (назовем их временно фрагментами), совокупность которых можно разбить на некоторое число непересекающихся классов так, чтобы

а) каждый фрагмент первого класса допускал программирование (проверку) независимо от других фрагментов и без учета их внутренней структуры;

б) каждый фрагмент i-го класса допускал программирование (проверку) после того, как составлены (проверены) некоторые фрагменты $(i-1)$-го класса (и, может быть, других младших классов), независимо от программирования (проверки) других фрагментов i-го класса и фрагментов старших классов.

2. Каждый фрагмент должен допускать программирование (проверку) путем перехода от команды к команде в определенном порядке.

3. Программа должна допускать обозримое описание, составление которого возможно до программирования.

4. Расписывание команд отдельных фрагментов должно подчиняться определенным формальным правилам, позволяющим выполнять его механически по заданному описанию программы.

Заметим, что составление программ, конструкция которых удовлетворяет перечисленным требованиям, может быть автоматизировано (см. гл. 7).

Чтобы при составлении и проверке команд одних фрагментов, передающих управление другим фрагментам, можно было не учитывать внутренней структуры последних, каждый фрагмент должен содержать одну-единственную команду, могущую получать управление извне (такая команда называется первой). Чтобы составление или проверку каждого фрагмента можно было выполнять в одном определенном порядке, внутри фрагмента не должно быть команд условного перехода к другим командам фрагмента. Наконец, для того чтобы при проверке каждого фрагмента не нужно было учитывать внутренней структуры фрагментов, от которых происходит переход к командам данного фрагмента, каждый фрагмент должен иметь единственную команду, передающую управление вовне. Такая команда называется последней командой фрагмента. В некоторых фрагментах последней может быть команда, предусматривающая останов.

Команда, видоизменяющая другую команду, может быть составлена или проверена лишь после того, как составлена или проверена изменяемая команда. Поэтому внутри фрагмента изменяемые команды не должны предшествовать изменяющим. Если же такое расположение команд необходимо в программе, то изменяемые и изменяющие команды должны быть объединены в различные между собой фрагменты. Так как описание программы должно быть описанием фрагментов и их взаимосвязи, то для большей обозримости описания число фрагментов программы должно быть возможно малым. Уменьшение числа фрагментов можно достигнуть путем их «укрупнения», которое возможно, если команды одинакового функционального назначения расположены по возможности вместе (а не «вперемежку» с командами другого функционального назначения).

Команду, вырабатывающую сигнал ω, и следующую за ней команду условного перехода, выполнение которой определяется значением сигнала ω, следует относить к одному и тому же фрагменту (чтобы не увеличивать числа фрагментов).

Таким образом, каждому фрагменту программы должны быть присущи следующие свойства:

1. Упорядоченность. Только одна (первая) команда фрагмента может получать управление извне. Внутри фрагмента управление передается от команды команде в одном

определенном порядке. Только одна команда фрагмента (последняя) может передавать управление вовне или предусматривать останов.

2. Связность. Если передать управление первой команде фрагмента, то после выполнения конечного числа команд управление получит его последняя команда.

3. Автономность. Команда условного перехода, принадлежащая фрагменту, должна управляться значением сигнала ω, выработанным командами этого же самого (а не другого) фрагмента.

4. Простота. Команды фрагмента должны быть функционально едины и (с точки зрения их роли в программе).

5. Допускается, чтобы два фрагмента, из которых первый может передавать управление только второму, а второй может получать управление только от первого, между собой пересекались, т. е. чтобы существовала группа команд, содержащая первую команду второго и последнюю команду первого фрагмента, принадлежащая одновременно обоим фрагментам.

Фрагменты программы, обладающие перечисленными свойствами, называются *элементарными операторами программы*. В операторы программы при ее начальном состоянии могут включаться коды, не являющиеся командами, но такие, которые должны быть переработаны в команды до момента их выполнения. При этом обычно известно заранее, каково будет функциональное назначение каждой такой «будущей команды».

Необходимо сразу отметить, что возможность пересечения элементарных операторов связана со спецификой программно-управляемой машины и зависит от того, что некоторые команды могут объединять в себе операции, имеющие в программе различное функциональное значение. Пересекающиеся элементарные операторы в один оператор не объединяют. Пересечение элементарных операторов можно было бы исключить, вводя в программу дополнительные команды, что, однако, нецелесообразно.

Пример 4.5. Требуется в ячейках оперативной памяти машины *Стрела*, начиная с ячейки 0025, расположить команды, позволяющие вычислить разность квадратов чисел «0400» и «0401», записать ее в ячейку 0500 и проверить, будет ли эта величина отрицательна. Если она окажется отрицательной, то управление передать в ячейку 0300, а в противном случае — в ячейку 0250.

Составим сперва элементарный оператор арифметического счета

0025) 0400 0400 0500 0 05,
0026) 0401 0401 0501 0 05,
0027) 0500 0501 0500 0 03.

Затем составим элементарный оператор, проверяющий знак результата и производящий требуемый условный переход (логический оператор)

0030) 0500 0000 0000 0 01,
0031) 0250 0300 0000 0 20.

Заметим, что команда «0027» вырабатывает то же значение сигнала ω, что и команда «0030». Поэтому команду «0030» можно исключить и на ее место записать команду «0031». Получится

0025) 0400 0400 0500 0 05,
0026) 0401 0401 0501 0 05,
0027) 0500 0501 0500 0 03,
0030) 0250 0300 0000 0 20.

Теперь оказалось, что команда «0027» принадлежит одновременно арифметическому и логическому элементарным операторам, которые, следовательно, пересекаются.

Обычно команду останова считают самостоятельным элементарным оператором.

В некоторых случаях бывает полезным понятие *обобщенного оператора программы:* совокупности элементарных операторов программы, удовлетворяющей определению (стр. 125) обобщенного оператора.

4. Стандартные операторы. Логическая схема программы. В практике программирования выявилось, что в большинстве программ чаще всего встречается некоторое количество типов элементарных операторов. Для этих операторов приняты постоянные обозначения и разработаны формальные приемы их программирования. Эти операторы принято называть *стандартными* (см. табл. 4.2).

Имея готовую программу (начальное состояние), ее можно разбить на элементарные операторы, последние перенумеровать в порядке возрастания номеров команд программы, а затем способом, описанным для построения схем решающих алгорифмов (см. п. 1 настоящего параграфа), построить *схему программы.* Схему программы принято называть ее *логической схемой.* Если в логической схеме присутствует оператор

Таблица 4.2. Стандартные элементарные операторы программ

Название оператора	Обозначение	Функции, выполняемые оператором в программе
Арифметический оператор	A_v	Вычисления по формулам
Оператор переноса	$П_v$	Перенос чисел из одних ячеек оперативной памяти в другие или из одного запоминающего устройства в другое
Оператор начала	$И_0$	Символ, указывающий первую команду программы
Оператор конца	$Я_v$	Останов машины
Оператор переадресации по параметру i	$F_v(i)$	Переадресация команд, зависящих от параметра i, в результате которой их адреса, соответствующие значению параметра $i = i_k$, заменяются адресами, соответствующими значению параметра $i = i_{k+1}$ (см. гл. 6, § 1)
Оператор изменения параметра	$F_v(i)$	Изменение значения параметра i
Оператор восстановления по параметру i	$O_v(i)$	Приведение адресов команд, измененных оператором $F_v(i)$ к виду, отвечающему начальному значению параметра i (см. гл.6, § 1)
Оператор восстановления параметра i	$O_v(i)$	Восстановление начального значения параметра i
Оператор формирования	$Ф_v$	Формирование команд других операторов (см. гл. 6, § 4)
Засылка	$З_v$	Вынесение величин в стандартные ячейки (см. гл. 6, § 3)
Оператор циркуляции	$Ц_v$	Циркуляция чисел в стандартных ячейках (см. гл. 6, § 3)
Обращение	$E_v(m; n)$	Переход к оператору с номером m и запись команды возврата непосредственно после оператора с номером n (см. гл. 6, § 5)
Подвод ленты	$У_v$	Подвод нужной зоны магнитной ленты
Перенос на ленту	L_v	Перенос из оперативной памяти в некоторую зону магнитной ленты
Считывание с ленты	$Л_v$	Перенос с магнитной ленты в оперативную память
Контрольное суммирование	Σ_v	Вычисление контрольной суммы
Нестандартный оператор	H_v	Всякий оператор, отличный от перечисленных (в частности, может обозначать обобщенный оператор)

$E_\nu(m; n)$, то символ оператора, имеющего индекс $\nu = n$, снабжается сверху чертой (например, $\bar{A}_n$, $\bar{F}_n(i)$ и т. п.). Эта черта над оператором в схеме программы является напоминанием программисту, что после команд оператора с номером n необходимо оставить место для команды возврата.

Логическая схема программы, составленная по уже готовой программе, может служить ее описанием и быть полезной при проверке программы лицом, не участвовавшим в ее составлении, и при отладке программы на машине. Необходимы логические схемы и при анализе программ, например с целью определения общих групп операторов в различных программах.

Однако программирование без использования логической схемы допустимо лишь в самых легких случаях. Наиболее целесообразно разработать логическую схему программы до программирования и использовать ее уже в процессе программирования.

При программировании вручную схему программы составляют, исходя из схемы решающего логарифма. Затем каждый элементарный оператор схемы программы, пользуясь методом символических адресов, представляют в виде некоторой последовательности команд. При этом каждый раз, когда необходимо написать команду условного или безусловного перехода к оператору, составление которого еще невозможно, для обозначения ячеек, которые будут хранить этот еще несоставленный оператор, вводят новые символические адреса.

Объединение отдельных элементарных операторов в программу производят в процессе распределения памяти машины и присвоения истинных адресов.

Перечисленная совокупность приемов получила название «операторный метод программирования».

Пример 4.6. Составить логическую схему программы для вычисления на машине *Минск-2* таблицы значений функции

$$y = \alpha_0 \sin^8 x + \alpha_1 \sin^7 x + \ldots + \alpha_7 \sin x + \alpha_8, \tag{4.1}$$

для заданных значений независимой переменной $x = x_1, x_2, \ldots, x_n$, удовлетворяющих условию

$$0 < x_i < \frac{\pi}{4},$$

если коэффициенты $\alpha_0, \alpha_1, \ldots, \alpha_8$ удовлетворяют условию

$$\sum_{j=0}^{8} |\alpha_j| < 1. \tag{4.2}$$

С помощью подстановки

$$z = \sin x$$

преобразуем (4.1) к виду

$$y = \alpha_0 z^8 + \alpha_1 z^7 + \ldots + \alpha_7 z + \alpha_8. \qquad (4.3)$$

Для вычислений по формуле (4.3) воспользуемся так называемой схемой Горнера, придающей вычислениям однообразный циклический характер, для чего приведем (4.3) к виду

$$y = ((((((\alpha_0 z + \alpha_1) z + \alpha_2) z + \alpha_3) z + \alpha_4) z + \alpha_5) z + \alpha_6) z + \alpha_7) z + \alpha_8.$$

Решающий алгорифм описанной задачи, полагая, что вначале i имеет значение 1, можно записать следующим образом:

1°. Положить $j = 1$. Перейти к п. 2°.

2°. Вычислить $\sin x_i$ и принимать эту величину в дальнейшем за z. Перейти к п. 3°.

3°. Положить $\alpha = \alpha_0$. Перейти к п. 4°.

4°. Вычислить величину $\alpha z + \alpha_j$ и принимать ее в дальнейшем за α. Перейти к п. 5°.

5°. Увеличить значение j на 1. Перейти к п. 6°.

6°. Проверить, выполнено ли условие $j = 9$.

Если нет, то перейти к п. 4°, в противном случае — к п. 7°.

7°. Полагать $y_i = \alpha$. Перейти к следующему пункту.

8°. Увеличить значение i на 1. Перейти к следующему пункту.

9°. Проверить, удовлетворяет ли i условию

$$i = n + 1.$$

Если нет, то перейти к пункту 1°, если да, то к следующему пункту.

10°. Закончить процесс.

Условия (4.1) и (4.2) убеждают нас в том, что масштабирование величин для решения задачи в случае применения чисел с запятой, фиксированной перед старшим разрядом, не требуется. В машине *Минск-2* предусмотрена возможность применения как чисел с фиксированной запятой, так и чисел с плавающей запятой. В данном случае следует оказать предпочтение числам с фиксированной запятой, так как скорость выполнения арифметических операций при этом будет больше.

Принимая пункты алгорифма соответственно за операторы Q_2, Q_4^i, Q_5, Q_6^j, Q_7, P_8, Q_9^i, Q_{10}, P_{11} и $Я_{16}$, получаем схему решающего алгорифма

$$И_0 Q_2 Q_4^i Q_5 Q_6^j Q_7 P_8 Q_9^i Q_{10} P_{11} Я_{16}$$

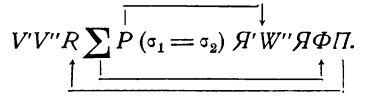
или

$$И_0 \underset{1}{\rfloor} Q_2 Q_4^i Q_5 \underset{2}{\rfloor} Q_6^j Q_7 P_8 \underset{2}{\lfloor} Q_9^i Q_{10} P_{11} \underset{1}{\lfloor} Я_{16}.$$

В этой схеме, забегая вперед, мы для облегчения дальнейших пояснений приписали операторам номера, которые они будут иметь в логической схеме программы.

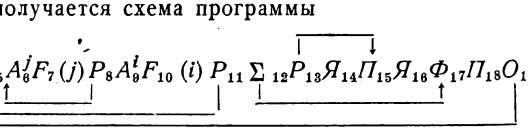
В программе предусмотрим контроль правильности вычислений способом двойного счета; при этом остановимся для нее на

организации (см. § 4 гл. 3) вида



Эта организация предусматривает раздельные вводы программы (V') и исходных данных (V''). Предпоследний в ее схеме оператор Φ изменяет переход от оператора контрольного суммирования Σ таким образом, чтобы после второго счета оператор Σ передавал управление оператору $P(\sigma_1 = \sigma_2)$. Оператор $П$ переносит первую контрольную сумму σ_1 на хранение в специально отведенную ячейку и тем самым освобождает место для второй контрольной суммы σ_2. Предположим, что ввод исходных данных должен быть программный (см. описание машины *Минск-2*).

Перестраивая схему решающего алгорифма вместе со схемой организации, получим схему программы. При этом после оператора $И_0$ включим оператор ввода исходных данных $П_1$; оператор Q_2 обозначим через $O_2(j)$; после него включим оператор A_3^i перевода числа из десятичной системы счисления в двоичную; операторы Q_4^i, Q_5, Q_6^j, Q_7, Q_9^i, Q_{10} обозначим соответственно через A_4^i, A_5, A_6^j, $F_7(j)$, A_9^i, $F_{10}(i)$; здесь оператор A_9^i переводит числа из двоичной системы счисления в десятичную и полагает их равными величинам y_i; затем включим операторы, предусмотренные схемой организации: Σ_{12} — для контрольного суммирования, P_{13} — для сравнения контрольных сумм, $Я_{14}$ — для останова в случае их неравенства, $П_{15}$ — для вывода из машины результатов, Φ_{17} — для изменения перехода от оператора Σ_{12}, $П_{18}$ — для переноса первой контрольной суммы на хранение и, наконец, не показанный в схеме организации, но необходимый для проведения второго счета оператор $O_{19}(i)$ (после $П_{18}$). Таким образом, получается схема программы



или

$$И_0 П_1 \underset{1}{\lrcorner} O_2(j)\, A_3^i A_4^i A_5 \underset{2}{\lrcorner} A_6^j F_7(j)\, P_8 \underset{2}{\lfloor} A_9^i F_{10}(i)\, P_{11} \underset{1}{\lfloor} \Sigma_{12} \underset{4}{\lfloor} \underset{5}{\lrcorner} P_{13} \overset{3}{\lceil} Я_{14} \underset{3}{\lrcorner} П_{15} Я_{16} \underset{4}{\lrcorner} \Phi_{17} П_{18} O_{19}(i) \underset{1}{\lfloor}. \quad (4.4)$$

Примечание. Оператор Φ_{17} изменяет в схеме (4.4) левый знак перехода $\underset{4}{\lfloor}$ на знак $\underset{5}{\lfloor}$.

Поскольку в машине *Минск-2* нет специальных устройств или закоммутированных подпрограмм для получения $\sin x$ по заданному значению x, для перевода числа из десятичной (вернее двоично-кодированной десятичной) системы счисления в двоичную и для

обратного перевода, то эти действия необходимо осуществлять программным способом. Наиболее целесообразно заранее составить подпрограммы для вычисления значений часто встречающихся функций и упомянутых переводов чисел и при составлении новых программ применять эти подпрограммы (см. гл. 7). Для контрольного суммирования необходимо тоже иметь специальную подпрограмму.

Предположим, что мы располагаем подпрограммами:

1) для вычисления по заданному числу x величины $\sin x$;

2) для перевода десятичного числа в двоичную систему счисления;

3) для перевода двоичного числа в десятичную систему счисления;

4) для контрольного суммирования.

Будем считать, что первые три подпрограммы требуют следующей информации: номер α_1 ячейки, в которой расположено исходное число, и номер α_2 ячейки, в которой должен быть записан результат. Для четвертой подпрограммы нужно задавать номер α_1 ячейки, начиная с которой в последовательных ячейках расположены числа, подлежащие суммированию, и количество n этих чисел; результат суммирования подпрограмма помещает в ячейку с номером $\alpha_1 - 1$.

Предположим, что указанные данные должны быть представлены в виде кодов

$$+00\ 0\ 00\ \alpha_1\ \alpha_2 \quad \text{и} \quad +00\ 0\ 00\ \alpha_1\ n_1,$$

имеющих структуру команд для машины *Минск-2*, причем α_1, α_2 и n должны быть записаны как восьмеричные четырехразрядные числа.

Будем считать, что для любой из перечисленных подпрограмм указанные коды должны быть записаны в ячейку 3776 и что любая из подпрограмм после окончания работы передает управление в ячейку 3777. Перед тем как передать управление подпрограмме, нужно в последнюю ячейку заслать команду, которая обеспечит возврат в основную программу.

В полученной нами схеме программы, чтобы приспособить ее к машине *Минск-2* надлежит оператор A_3^i заменить оператором E_3^i, производящим обращение к подпрограмме перевода из десятичной системы счисления в двоичную, оператор A_4^i — оператором E_4^i, производящим обращение к подпрограмме вычисления значений синусов, оператор A_9^i оператором E_9^i, производящим обращение к подпрограмме перевода чисел из двоичной системы счисления в десятичную, оператор Σ_{12} — оператором E_{12}, производящим обращение к подпрограмме контрольного суммирования для получения контрольной суммы.

Заметим, что операторы $П_1$ и $П_{18}$ (ввода исходных данных и выдачи на печать результатов) благодаря тому, что машина М-3 позволяет производить программные ввод и выдачу на печать только по одному числу, будут обобщенными операторами.

Наконец, ввиду того, что *Минск-2* не имеет команды сравнения, равенство контрольных сумм σ_1 и σ_2 придется устанавливать путем

проверки двух неравенств: $\sigma_1 - \sigma_2 \leqslant -0$; $\sigma_2 - \sigma_1 \leqslant -0$. Это заставляет нас оператор $P_{13} \overset{3}{\Gamma}$ заменить двумя операторами

$$P'_{13} \underset{6}{\llcorner} P''_{13} \overset{3}{\Gamma} \underset{6}{\lrcorner}.$$

После внесения в схему (4.4) указанных изменений получим логическую схему программы, учитывающую особенности машины *Минск-2*

$$И_0 П_1 \underset{1}{\lrcorner} O_2(j) E_3^i E_4^i A_5 \underset{2}{\lrcorner} A_6^j F_7(j) P_8 \underset{2}{\llcorner} E_9^i F_{10}(i) P_{11} \underset{1}{\llcorner} E_{12} \underset{4}{\llcorner}$$

$$\underset{5}{\lrcorner} P'_{13} \underset{6}{\llcorner} P''_{13} \overset{3}{\Gamma} \underset{6}{\lrcorner} Я_{14} \underset{3}{\lrcorner} П_{15} Я_{16} \underset{4}{\lrcorner} \Phi_{17} П_{18} O_{19}(i) \underset{1}{\llcorner}. \qquad (4.5)$$

После того как составлена логическая схема программы (4.5), приступаем к составлению программ отдельных операторов.

Литература к главе 4: [11], [31], [35], [36], [37], [38], [47], [49].

ГЛАВА 5

РАВНОСИЛЬНЫЕ ПРЕОБРАЗОВАНИЯ ЛОГИЧЕСКИХ СХЕМ

§ 1. Комплексы и их эквивалентные преобразования

1. Некоторые общие термины и условные обозначения. Если A — некоторое множество, произвольный элемент которого обозначен буквой x, то будем применять запись

$$A = \{x\}.$$

Упорядоченный набор конечного числа элементов какого-либо множества будем называть *кортежем*. Если кортеж X состоит из элементов $x_1, x_2, \ldots, x_n$, то будем писать

$$X = \{x_i\}^n_{i=1}.$$

В тех случаях, когда кортеж состоит из одного *произвольного* элемента x_i, будем писать

$$X = \{x_i\}_{i=\text{const}}.$$

Если бы кортеж состоял из одного *определенного* элемента x_1, мы вместо $X = \{x_i\}_{i=1}$ написали бы

$$X = \{x_1\}.$$

В тех случаях, когда кортеж состоит из нескольких элементов, обозначенных различными буквами, будем его записывать в виде перечня этих букв, разделенных знаками «;» (точка с запятой), заключенного в фигурные скобки, как, например,

$$X = \{x; y; z\}.$$

Если каждый элемент множества B принадлежит множеству A, то будем писать

$$B \subseteq A.$$

Знак $\subseteq$ читается как «является частью или равно», так что последняя запись означает «B является частью или равно A».

Если каждый элемент множества B принадлежит множеству A, но известно, что существуют элементы множества A, не принадлежащие B, то будем писать

$$B \subseteq A.$$

Знак $\subseteq$ читается как «является частью».

Если $B \subseteq A$ и $A \subseteq B$, то будем писать

$$A = B$$

и говорить, что множества A и B между собой *равны*.

Предложение «x_1 является элементом множества A» будем записывать в виде

$$x_1 \in A,$$

а предложение «x_1 не является элементом множества A» — в виде

$$x_1 \overline{\in} A.$$

Мы в дальнейшем будем употреблять понятие функции в более широком смысле, чем это принято в математическом анализе. Именно, будем считать, что значениями аргумента и функции могут быть любые элементы, а не обязательно числа.

Если $y = f(x)$ определена для любого значения x, принадлежащего множеству X, и принимает значения, являющиеся элементами множества Y, то иногда говорят, что «$f(x)$ *отображает* X в Y» или что «$f(x)$ является *отображением* X в Y». Последнее предложение условимся записывать в виде

$$y = f(x) : X \to Y.$$

В некоторых случаях вместо $y = f(x)$ мы будем для краткости употреблять $f(x)$ или даже f. При этом введенная нами запись будет принимать вид

$$f(x) : X \to Y \quad \text{или} \quad f : X \to Y.$$

Заметим, что значениями независимого переменного могут быть упорядоченные наборы значений некоторых других переменных. Например, значениями аргумента x функции $y = f(x)$ могут быть наборы значений переменных x_1,

$x_2, \ldots, x_n$. В этом случае вместо $y=f(x)$ можно писать

$$y=\varphi(x_1, x_2, \ldots, x_n).$$

Такую функцию, как и в классической математике, мы будем называть *функцией многих переменных*, аргументами которой являются величины $x_1, x_2, \ldots, x_n$.

Пересечение (т. е. общую часть) множества A и B будем обозначать как $A \cdot B$, а сумму (т. е. объединение) этих множеств — как $A+B$. Наконец, *разность* этих множеств (т. е. совокупность элементов, входящих в множество A, но не входящих в множество B) будем обозначать как $A-B$.

2. Память. Операторы. Пусть Ω — множество элементов, называемых *ячейками*. Если α — произвольная ячейка, то

$$\Omega=\{\alpha\}.$$

Множество Ω будем называть *памятью*. Пусть, кроме того, имеется множество (обозначим его буквой E), элементы которого называются *состояниями ячеек*. Произвольное состояние ячейки обозначим буквой γ. Тогда

$$E=\{\gamma\}.$$

Функцию $\gamma=f(\alpha)$, отображающую Ω в E

$$\gamma=f(\alpha): \Omega \to E,$$

будем называть *состоянием памяти*. Эта функция каждой ячейке ставит в соответствие вполне определенное состояние ячейки.

Пример 5.1. Функция, названная нами состоянием памяти, может быть в частном случае задана табличным способом. Остановимся на «машинном» истолковании понятий ячейки и состояния ячейки. При разработке программы программист обычно составляет так называемое распределение памяти, представляющее собой табличное изображение начального состояния памяти. Это осуществляется следующим образом. Лист бумаги разграфляют на клетки, которые считают изображениями ячеек. В каждой клетке указывают содержимое ячейки, которая соответствует этой клетке.

Рассматривая теперь состояния памяти как самостоятельные объекты, обозначим множество всех состояний памяти буквой H, т. е. положим, что

$$H=\{f(\alpha)\} \quad \text{или} \quad H=\{f\}.$$

Говоря о состояниях памяти как самостоятельных объектах, будем их обозначать символами вида f и $\bar{f}$; если же речь будет идти о связи между ячейками и их состояниями, будем состояния памяти обозначать как $f(\alpha)$ или $\bar{f}(\alpha)$.

Оператором будем называть отображение

$$\bar{f} = A(f): G \to G',$$

где $G \subseteq H$, $G' \subseteq H$. Другими словами, оператором называется функция, заданная на некотором множестве G состояний памяти и отображающая их на другие состояния памяти, входящие в множество G'.

Произведением AB двух операторов $A(f): G \to G'$ и $B(f): G' \to G''$ называется оператор

$$B(A(f)): G \to G''.$$

3. Операторы специального вида. Пусть X, Y и Z — кортежи, состоящие соответственно из n, m и l ячеек, причем Y и Z между собой не пересекаются. Введем обозначение

$$U = Y + Z.$$

Оператор $\bar{f} = A(f)$ с областью задания G называется $(n, m+l)$-оператором, если он удовлетворяет следующим трем условиям:

1) если $\alpha \overline{\in} U$, то $\bar{f}(\alpha) = f(\alpha)$;

2) коль скоро $f_1(\alpha) = f_2(\alpha)$ при $\alpha \in X$, то $\bar{f}_1(\alpha) = \bar{f}_2(\alpha)$ при $\alpha \in U$;

3) если $f_1 \in G$ и $f_2(\alpha) = f_1(\alpha)$ при $\alpha \in X$, то $f_2 \in G$. Подчеркнем, что символами f, f_1 и f_2 мы обозначили значения аргумента, а символами $\bar{f}$, $\bar{f}_1$ и $\bar{f}_2$ — соответствующие им значения функции $A(f)$.

Приведенные выше три условия можно словесно сформулировать следующим образом:

1) оператор $A(f)$ не изменяет состояний ячеек, которые не принадлежат кортежу $U = Y + Z$;

2) оператор $A(f)$ придает ячейкам из U состояния, полностью определяемые начальными состояниями ячеек кортежа X;

3) если какое-либо состояние f_1 памяти входит в область задания G оператора $A(f)$, то вместе с f_1 в G входят все

мыслимые состояния памяти, совпадающие с f_1 на ячейках кортежа X (и отличающиеся от f_1 вне X).

Последнее условие позволяет нам в качестве области задания $(n, m+l)$-оператора указывать не множество состояний всей памяти $\mathfrak{Q}$, а множество состояний кортежа X (т. е. множество функций вида $\gamma = \varphi(\alpha)$, определенных только для тех ячеек α, которые входят в X). В дальнейшем так и будем поступать, не оговаривая этого дополнительно.

Кортеж X называется *входным*, кортеж Y — *выходным*, а кортеж Z — *рабочим* (или кортежем *рабочих ячеек*) для оператора $A(f)$.

Пример 5.2. При «машинном» истолковании понятий ячейки и состояния ячейки видно, что арифметические элементарные операторы программы являются $(n, m+l)$-операторами. Действительно, арифметический оператор программы имеет входной, выходной и рабочий кортежи ячеек. Являясь группой команд, он в зависимости от содержимого своих входных ячеек вырабатывает содержимое своих выходных и рабочих ячеек (условие 2). Содержимого ячеек, не принадлежащих его выходному и рабочему кортежам, он не меняет (условие 1). Его «работа» не зависит от содержимого ячеек, не являющихся для него входными (условие 3). При этом, конечно, сами ячейки, хранящие команды арифметического оператора, должны быть исключены из рассмотрения.

Определение $(n, m+l)$-оператора не устанавливает различия между ячейками выходного и рабочего кортежей. Различие между ними будет установлено несколько ниже.

4. Комплексы. Остановимся на n, $1+0$-операторах. Для них кортеж Z является пустым. Предположим, что

$$X = \{\alpha_i\}_{i=1}^{n}; \quad Y = \{\alpha_i\}_{i=\text{const}}.$$

Очевидно, преобразование состояния памяти $\mathfrak{Q}$, осуществляемое n, $1+0$-оператором, можно полностью описать формулами

$$\overline{f}(\alpha_i) = \varphi(f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)),$$

$$\overline{f}(\alpha) = \varphi(\alpha) \quad \text{при } \alpha \neq \alpha_i.$$

Вторую из этих формул можно подразумевать, не выписывая ее, а в первой формуле можно для упрощения записи отбросить символы $\overline{f}(\,)$ и $f(\,)$, заменив при этом знак $=$ знаком $:=$,

который называется *знаком присваивания* *). Тогда описание n, $1+0$-оператора примет вид формулы

$$\alpha_i := \varphi(\alpha_1, \alpha_2, \ldots, \alpha_n).$$

Произведение операторов, изобразимых с помощью таких формул, будем записывать в виде столбца формул.

Рассмотрим столбец формул

$$\left.\begin{aligned} \alpha_{t_1} &:= \varphi_1(\alpha_1, \alpha_2, \ldots, \alpha_n), \\ \alpha_{t_2} &= \varphi_2(\alpha_1, \alpha_2, \ldots, \alpha_n), \\ &\ldots\ldots\ldots\ldots \\ \alpha_{t_N} &:= \varphi_N(\alpha_1, \alpha_2, \ldots, \alpha_n). \end{aligned}\right\} \qquad (5.1)$$

При этом будем считать, что как в левых, так и в правых частях этих формул не указаны никакие ячейки, кроме α_1, α_2, ..., α_n. Это вовсе не значит, что любая функция $\varphi_j(\alpha_1, \alpha_2, \ldots, \alpha_n)$, присутствующая в (5.1), в самом деле имеет в качестве своих аргументов все ячейки α_1, α_2, ..., α_n. Некоторые из перечисленных ячеек могут быть *фиктивными* аргументами, как это, например, имеет место для аргумента y в записи $f(x, y)$, обозначающей $\sin x$. Если какая либо ячейка α_i является нефиктивным аргументом функции φ_j, будем применять запись (читается: α_i является существенным аргументом φ_j)

$$\alpha_i = sa\,\varphi_j.$$

Если же α_i является фиктивным аргументом функции φ_j, то будем писать (читается: α_i не является существенным аргументом φ_j)

$$\alpha_i \neq sa\,\varphi_j.$$

Пусть для последовательности (5.1) задан входной кортеж $X = \{\alpha_{s_i}\}_{i=1}^{m}$ и выходной кортеж **) $Y = \{\alpha_{r_i}\}_{i=1}^{M}$.

*) В теории логических схем первоначально применялся знак $=$, смысл которого, таким образом, зависел от контекста. При записи предикатов он обозначал отношение равенства, а при записи операторов — присваивание ячейке, обозначение которой стоит слева от него, состояния, определяемого формулой, стоящей справа от него. Знак $:=$ был позаимствован позже из алгоритмического языка АЛГОЛ.

**) Ячейки α_{s_i} и ячейки α_{r_i} тоже должны содержаться среди ячеек α_1, α_2, ..., α_n.

Выпишем все нефиктивные аргументы функции φ_1, а затем последовательно для $k=2, 3, \ldots, N$ выпишем все нефиктивные аргументы функции φ_k, отличные от ранее выписанных и от $\alpha_{t_1}, \alpha_{t_2}, \ldots, \alpha_{t_{k-1}}$. Получим кортеж ячеек, который обозначим через L. Кроме того, обозначим через L' кортеж, состоящий из всех попарно-различных ячеек, встречающихся в последовательности $\alpha_{t_1}, \alpha_{t_2}, \ldots, \alpha_{t_N}$.

Столбец формул (5.1) будем называть *нормальной последовательностью* (формул) с входным кортежем X и выходным кортежем Y, если выполнены два условия:

$$L \subseteq X, \quad Y \subseteq X + L'.$$

Нормальная последовательность с входным кортежем X и выходным кортежем Y, рассматриваемая вместе с некоторым множеством G состояний кортежа X, называется комплексом, если произведение n, $1+0$-операторов, изображением которого является нормальная последовательность, определено на G, т. е. если, как говорят, нормальная последовательность *вычислима* для любого входящего в G состояния кортежа X. Множество состояний G кортежа X называется *областью задания* комплекса.

Очевидно, комплекс является изображением некоторого $(m, M+l)$-оператора, имеющего входной кортеж X, выходной кортеж Y и рабочий кортеж $Z=L'-Y$.

Пример 5.3. Арифметический оператор может быть записан в виде комплекса. При этом должно быть известно, в каких ячейках содержатся исходные данные для этого оператора и в каких ячейках будет получен результат. Областью задания, если она специально не указана, можно считать максимальное множество состояний входного кортежа, для которых нормальная последовательность комплекса вычислима.

Например, дан арифметический оператор в коде машины *Стрела*:

$a+1)$	0040	0041	0060	0	03
$a+2)$	0040	0042	0061	0	01
$a+3)$	0061	0060	0061	0	05,
$a+4)$	0041	0061	0062	0	03
$a+5)$	0062	0000	0063	0	62
$a+6)$	0063	0062	0062	0	03

Предположим, что исходные данные этого оператора размещены в ячейках 0040, 0041 и 0042, а результаты получаются в ячейках 0040 и 0062.

Обозначим ячейку 0040 буквой x, ячейку 0041 — буквой y, ячейку 0042 — буквой z. Тогда входной кортеж искомого комплекса

будет иметь вид

$$X=\{x;\ y;\ z\}.$$

Ячейку 0060 обозначим через u, ячейку 0061 через v, ячейку 0062 через w, а ячейку 0063 — через t. Нормальная последовательность комплекса будет:

$$u:=x-y,$$

$$v:=x+z,$$

$$v:=v\cdot u,$$

$$w:=y-v,$$

$$t:=1:w,$$

$$w:=t-w.$$

Выходной кортеж комплекса будет

$$Y=\{x;\ w\},$$

а его рабочий кортеж —

$$Z=\{u;\ v;\ t\}.$$

Областью задания комплекса можно считать множество тех состояний кортежа X, для которых выполнима операция $1:w$, присутствующая в предпоследней формуле нормальной последовательности.

Будем допускать комплексы, нормальные последовательности которых не содержат ни одной формулы. Для каждого такого комплекса должно выполняться условие $Y\subseteq X$. При этом считается, что $N=0$ (где N — число формул в нормальной последовательности).

5. Эквивалент комплекса. Рассмотрим некоторый комплекс K с областью задания G, входным кортежем X, состоящим из ячеек $\alpha_{s_1}, \alpha_{s_2}, \ldots, \alpha_{s_m}$, выходным кортежем Y, состоящим из ячеек $\alpha_{r_1}, \alpha_{r_2}, \ldots, \alpha_{r_M}$, и нормальной последовательностью

$$\begin{aligned}
\alpha_{t_1} &:= \varphi_1(\alpha_1, \alpha_2, \ldots, \alpha_n),\\
\alpha_{t_2} &:= \varphi_2(\alpha_1, \alpha_2, \ldots, \alpha_n),\\
&\ldots\ldots\ldots\ldots\\
\alpha_{t_N} &:= \varphi_N(\alpha_1, \alpha_2, \ldots, \alpha_n).
\end{aligned}$$

Выполним процесс, определяемый описанным ниже алгорифмом подстановок последовательно для $k=1, 2, \ldots, M$.

Алгорифм подстановок.

1°. Положить $i = N$. Перейти к п. 2°.

2°. Положить $\theta_{k,\,i} = \alpha_{r_k}$. Перейти к п. 3°.

3°. Проверить, справедливо ли равенство $i = 0$. Если да, принять $\theta_{k,\,0}$ за θ_k и прекратить процесс; если нет, перейти к п. 4°.

4°. Подставить в $\theta_{k,\,i}$ вместо α_{t_i} функцию φ_i. Результат подстановки принять за $\theta_{k,\,i-1}$. Перейти к п. 5°.

5°. Уменьшить значение i на единицу. Перейти к п. 3°.

После выполнения алгорифма подстановок для $k = 1, 2, \ldots, M$, получим последовательность функций

$$\begin{aligned}
&\theta_1(\alpha_{s_1}, \alpha_{s_2}, \ldots, \alpha_{s_m}),\\
&\theta_2(\alpha_{s_1}, \alpha_{s_2}, \ldots, \alpha_{s_m}),\\
&\ldots\ldots\ldots\ldots\\
&\theta_M(\alpha_{s_1}, \alpha_{s_2}, \ldots, \alpha_{s_m}).
\end{aligned}$$

Выберем два непересекающихся кортежа ячеек $X' = \{\delta_j\}_{j=t+1}^{t+m}$ и $Y' = \{\beta_j\}_{j=1}^{M}$ и составим последовательность формул

$$\left.\begin{aligned}
\beta_1 &:= \theta_1(\delta_{t+1}, \delta_{t+2}, \ldots, \delta_{t+m}),\\
\beta_2 &:= \theta_2(\delta_{t+1}, \delta_{t+2}, \ldots, \delta_{t+m}),\\
&\ldots\ldots\ldots\ldots\\
\beta_M &:= \theta_M(\delta_{t+1}, \delta_{t+2}, \ldots, \delta_{t+m}).
\end{aligned}\right\} \qquad (5.2)$$

Легко видеть, что полученный столбец формул представляет собой нормальную последовательность с входным кортежем X' и выходным кортежем Y'.

Обозначим через G' множество состояний кортежа X', удовлетворяющее следующему условию: каждому состоянию Φ' кортежа X', входящему в G', взаимно однозначно соответствует состояние Φ кортежа X, принадлежащее G, причем

$$\Phi'(\delta_{t+j}) = \Phi(\alpha_{s_j}) \qquad (j = 1, 2, \ldots, m).$$

Нормальная последовательность (5.2) с входным кортежем X' и выходным кортежем Y', рассматриваемая вместе с G', представляет собой комплекс, который мы будем называть *эквивалентом* комплекса K и будем обозначать символом $ЭК$.

6. Эквивалентность комплексов. Пусть K_1 и K_2 — два комплекса. Предположим, что эквивалент $\mathcal{Э}K_1$ комплекса K_1 имеет область задания G' и нормальную последовательность

$$\left.\begin{aligned} \beta_1 &:= \theta'_1(\delta_{t+1}, \delta_{t+2}, \ldots, \delta_{t+m}),\\ \beta_2 &:= \theta'_2(\delta_{t+1}, \delta_{t+2}, \ldots, \delta_{t+m}),\\ &\ldots\ldots\ldots\ldots\\ \beta_{M'} &:= \theta'_{M'}(\delta_{t+1}, \delta_{t+2}, \ldots, \delta_{t+m}) \end{aligned}\right\} \qquad (5.3)$$

с входным кортежем X' и выходным кортежем Y', а эквивалент $\mathcal{Э}K_2$ комплекса K_2 — область задания G'' и нормальную последовательность

$$\left.\begin{aligned} \beta_1 &:= \theta''_1(\theta_{\tau+1}, \delta_{\tau+2}, \ldots, \delta_{\tau+\mu}),\\ \beta_2 &:= \theta''_1(\delta_{\tau+1}, \delta_{\tau+2}, \ldots, \delta_{\tau+\mu}),\\ &\ldots\ldots\ldots\ldots\\ \beta_{M''} &:= \theta''_{M''}(\delta_{\tau+1}, \delta_{\tau+2}, \ldots, \delta_{\tau+\mu}) \end{aligned}\right\} \qquad (5.4)$$

с входным кортежем X'' и выходным кортежем Y''.

Предположим, кроме того, что пересечение $X' \cdot X''$ входных кортежей непусто и что для каждого состояния, принадлежащего одному из множеств G' и G'', в другом из этих множеств найдется хотя бы одно состояние, совпадающее с ним на $X' \cdot X''$.

Пусть $X^* = X' + X''$, и пусть G^* есть совокупность состояний кортежа X^*, которые связаны с состояниями кортежей X' и Y' следующим образом:

а) для каждой пары состояний, первое из которых входит в G', а второе в G'' и которые совпадают на $X' \cdot X''$, в множестве G^* найдется состояние, совпадающее с первым на X' и со вторым — на X'';

б) для каждого состояния кортежа X^*, принадлежащего G^*, найдется пара состояний, из которых одно входит в G', а другое — в G'', для которых выполняется условие а).

Выберем из правых частей формул нормальной последовательности (5.3) эквивалента $\mathcal{Э}K_1$ все попарно между собой нетождественные функции θ'_{a_i}. Произведем аналогичный выбор функций θ''_{b_i} среди правых частей формул нормальной последовательности (5.4) эквивалента $\mathcal{Э}K_2$.

Комплексы K_1 и K_2 называются *эквивалентными*, если выполнены два следующих условия:

1. Количества отобранных функций равны, т. е. получены две последовательности функций:

$$\theta'_{a_1},\ \theta'_{a_2},\ \ldots,\ \theta'_{a_n},\qquad \theta''_{b_1},\ \theta''_{b_2},\ \ldots,\ \theta''_{b_n}.$$

2. На множестве G^* справедливы тождества

$$\theta'_{a_1} \equiv \theta''_{b_1},$$
$$\theta'_{a_2} \equiv \theta''_{b_2},$$
$$\cdot\quad\cdot\quad\cdot\quad\cdot$$
$$\theta'_{a_n} \equiv \theta''_{b_n}.$$

Если комплексы K_1 и K_2 эквивалентны, то будем писать

$$K_1 \sim K_2.$$

Легко видеть, что эквивалентность комплексов а) рефлексивна, т. е. $K_1 \sim K_1$; б) симметрична, т. е. если $K_1 \sim K_2$, то $K_2 \sim K_1$; в) транзитивна, т. е. если $K_1 \sim K_2$ и $K_2 \sim K_3$, то $K_1 \sim K_3$. Кроме того, очевидно, что каждый комплекс эквивалентен своему эквиваленту, т. е. $K \sim \mathcal{E}K$.

Пример 5.4. Будем под ячейками понимать буквы, являющиеся обозначениями величин, а под состояниями ячеек — значения этих величин (т. е. числа).

Пусть комплекс K_1 имеет нормальную последовательность

$$u := x + \sin^2 z + \cos^2 z,$$
$$v := y \cdot u,$$
$$w := v$$

с входным кортежом $X_1 = \{x;\ y;\ z\}$ и выходным кортежем $Y_1 = \{u,\ v,\ w\}$, а его область задания G_1 состоит из всевозможных состояний кортежа X_1, т. е. может быть представлена системой неравенств

$$-\infty < x < +\infty,$$
$$-\infty < y < +\infty,$$
$$-\infty < z < +\infty.$$

Пусть, кроме того, комплекс K_2 имеет нормальную последовательность

$$u := x + \frac{2}{\pi}(\arcsin t + \arccos t),$$
$$v := y \cdot u$$

с входным кортежем $X_2 = \{x, y, t\}$ и выходным кортежем $Y_2 = \{u; v\}$, причем его область задания G_2 может быть описана системой неравенств

$$-\infty < x < +\infty,$$
$$-\infty < y < +\infty,$$
$$0 \leqslant t \leqslant 1.$$

Покажем, что $K_1 \sim K_2$.

Действительно, после выполнения алгорифма подстановок для комплекса K_1 получим последовательность функций

$$x + \sin^2 z + \cos^2 z,$$
$$y\,(x + \sin^2 z + \cos^2 z),$$
$$y\,(x + \sin^2 z + \cos^2 z).$$

Заменяя в них буквы x, y, z соответственно через δ_2, δ_3, δ_4, получим $\mathcal{Э}K_1$ с нормальной последовательностью

$$\beta_1 := \delta_2 + \sin^2 \delta_4 + \cos^2 \delta_4,$$
$$\beta_2 := \delta_3\,(\delta_2 + \sin^2 \delta_4 + \cos^2 \delta_4),$$
$$\beta_3 := \delta_3\,(\delta_2 + \sin^2 \delta_4 + \cos^2 \delta_4),$$

имеющей входной кортеж $X_1' = \{\delta_i\}_{i=2}^{4}$. Область задания $\mathcal{Э}K_1$ может быть описана неравенствами

$$-\infty < \delta_2 < +\infty,$$
$$-\infty < \delta_3 < +\infty,$$
$$-\infty < \delta_4 < +\infty.$$

Выполняя аналогичные действия над комплексом K_2 и заменяя x, y, t соответственно на δ_2, δ_3, δ_1, получим эквивалент $\mathcal{Э}K_2$ комплекса K_2 с нормальной последовательностью

$$\beta_1 := \delta_2 + \frac{2}{\pi}\,(\arcsin \delta_1 + \arccos \delta_1),$$

$$\beta_2 := \delta_3 \left[\delta_2 + \frac{2}{\pi}\,(\arcsin \delta_1 + \arccos \delta_1)\right],$$

имеющей входной кортеж $X_2' = \{\delta_i\}_{i=1}^{3}$. Область задания $\mathcal{Э}K_2$ может быть описана неравенствами

$$0 \leqslant \delta_1 \leqslant 1,$$
$$-\infty < \delta_2 < +\infty,$$
$$-\infty < \delta_3 < +\infty.$$

Обозначим через G^* совокупность состояний кортежа $X^* = X_1' + X_2' = \{\delta_i\}_{i=1}^{4}$, которая может быть описана системой

неравенств

$$0 \leqslant \delta_1 \leqslant 1,$$
$$-\infty < \delta_2 < +\infty,$$
$$-\infty < \delta_3 < +\infty,$$
$$-\infty < \delta_4 < +\infty.$$

Отберем среди правых частей формул сперва для $\mathcal{E}K_1$, а затем для $\mathcal{E}K_2$ попарно нетождественные между собой функции и сравним полученные при этом последовательности функций. Очевидно, для любого состояния из G^* справедливы тождества

$$\delta_2 + \sin^2\delta_4 + \cos^2\delta_4 \equiv \delta_2 + \frac{2}{\pi}(\arcsin\delta_1 + \arccos\delta_1),$$

$$\delta_3(\delta_2 + \sin^2\delta_4 + \cos^2\delta_4) \equiv \delta_3\left[\delta_2 + \frac{2}{\pi}(\arcsin\delta_1 + \arccos\delta_1)\right],$$

ибо для состояний, входящих в G^*,

$$\sin^2\delta_4 + \cos^2\delta_4 \equiv 1,$$

$$\arcsin\delta_1 + \arccos\delta_1 \equiv \frac{\pi}{2}.$$

Результат, к которому мы пришли, означает, что комплексы K_1 и K_2 эквивалентны, т. е. что $K_1 \sim K_2$.

7. Основные эквивалентные преобразования. Если, внося некоторые изменения в комплекс K, мы получим комплекс K', эквивалентный K, то будем говорить, что в комплексе K *можно* произвести указанные изменения. При этом переход от K к K' будем называть *эквивалентным преобразованием* комплекса K в комплекс K'.

Символом Λ будем обозначать пустую формулу, считая, что перед любой формулой и после любой формулы нормальной последовательности всякого комплекса присутствует пустая формула. Если каждая из букв A и B обозначает последовательность формул и если в нормальной последовательности любого комплекса последовательность формул A можно заменить последовательностью формул B, то будем это изображать в виде записи

$$A \to B.$$

Совокупность записей $A \to B$, $B \to A$ будем изображать как

$$A \rightleftarrows B.$$

При этом запись

$$A \rightleftarrows \Lambda \tag{5.5}$$

обозначает, что группу формул A можно заменить пустой формулой, т. е., другими словами, можно *вычеркнуть*, и что эту группу формул можно в произвольном месте *вписать* в нормальную последовательность комплекса. Запись вида (5.5), в которой на месте A стоит не формула, а ячейка, мы будем применять, когда нужно показать, что ячейку можно вычеркнуть из кортежа или включить в него.

Условимся предложение «ячейки (буквы) α_{s_i} и α_j одинаковы» изображать в виде записи

$$\alpha_{s_i} (=) \alpha_j,$$

а предложение «ячейки (буквы) α_{s_i} и α_j различны» — в виде записи

$$\alpha_{s_i} (\neq) \alpha_j.$$

Преобразование С-1. Если имеет место тождество $\varphi_i \equiv \psi$, то

$$[\alpha_{t_i} := \varphi_i] \to [\alpha_{t_i} := \psi].$$

Преобразования А-2 и обратное ему В-2:

$$[\alpha_j := \alpha_j] \rightleftarrows \Lambda.$$

Если ячейка α_j не принадлежит входному кортежу комплекса, то одновременно с преобразованием В-2 необходимо ее включить во входной кортеж.

Преобразование С-3. Если выполнены условия $\alpha_{t_i} (\neq) \alpha_{t_{i+1}}$, $\alpha_{t_i} \neq sa\, \varphi_{i+1}$ и $\alpha_{t_{i+1}} \neq sa\, \varphi_i$, то

$$\begin{bmatrix} \alpha_{t_i} := \varphi_i \\ \alpha_{t_{i+1}} := \varphi_{i+1} \end{bmatrix} \to \begin{bmatrix} \alpha_{t_{i+1}} := \varphi_{i+1} \\ \alpha_{t_i} := \varphi_i \end{bmatrix}.$$

Преобразование А-4. Если выполнены условия $\alpha_{t_{i+1}} \neq sa\, \varphi_i$, $\alpha_{t_{i+1}} (\neq) \alpha_{t_i}$, то

$$\begin{bmatrix} \alpha_{t_i} := \varphi_i \\ \alpha_{t_{i+1}} := \varphi_{i+1} (\alpha_{t_i},\ \alpha_{j \neq t_i}) \end{bmatrix} \to \begin{bmatrix} \alpha_{t_{i+1}} := \varphi_{i+1} (\varphi_i,\ \alpha_{j \neq t_i}) \\ \alpha_{t_i} := \varphi_i \end{bmatrix}.$$

Здесь вместо $\varphi_{i+1}(\alpha_1, \alpha_2, \ldots, \alpha_n)$ употребляем символ $\varphi_{i+1}(\alpha_{t_i}, \alpha_{j\neq t_i})$, чтобы выделить аргумент α_{t_i}, заменяемый при преобразовании функцией φ_i.

Преобразование В-4. Если выполнены условия $\alpha_{t_i} \neq sa\,\varphi_{i+1}$ и $\alpha_{t_{i+1}} (\neq) \alpha_{t_i}$, то

$$\begin{bmatrix} \alpha_{t_i} & := \varphi_i(\varphi_{i+1}, \alpha_{j\neq t_{i+1}}) \\ \alpha_{t_{i+1}} & := \varphi_{i+1} \end{bmatrix} \to \begin{bmatrix} \alpha_{t_{i+1}} := \varphi_{i+1} \\ \alpha_{t_i} \quad := \varphi_i(\alpha_{t_{i+1}}, \alpha_{j\neq t_{i+1}}) \end{bmatrix}.$$

Будем называть *элементами* комплекса его входной кортеж $\{\alpha_{s_i}\}_{i=1}^{m}$; левые части формул его нормальной последовательности $\alpha_{t_1}, \alpha_{t_2}, \ldots, \alpha_{t_N}$; правые части формул его нормальной последовательности с предшествующими им знаками присваивания $:=\varphi_1, :=\varphi_2, \ldots, :=\varphi_N$; его выходной кортеж $\{\alpha_{r_i}\}_{i=1}^{M}$. Упорядочим элементы комплекса, приписывая им номера: 1 — входному кортежу; $2i$ — элементу $:=\varphi_i$; $2i+1$ — элементу a_{t_i}; $2N+2$ — выходному кортежу.

Пусть α — любая буква. Будем говорить, что перед первым элементом комплекса присутствует α-барьер; если $\alpha_{t_i}(=)\alpha$, то между $2i$-м и $(2i+1)$-м элементами комплекса находится α-барьер; после последнего элемента комплекса присутствует α-барьер; нигде больше α-барьеров нет. Перенумеруем α-барьеры, переходя от одного из них к другому в направлении возрастания номеров элементов комплекса. Совокупность элементов комплекса, заключенных между двумя последовательными α-барьерами, будем называть α-звеном. Каждому α-звену припишем номер, одинаковый с номером предшествующего ему α-барьера.

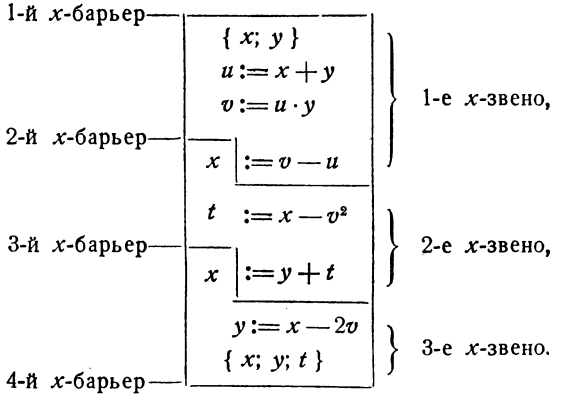
Пример 5.5. Рассмотрим комплекс, имеющий входной и выходной кортежи

$$X = \{x; y\}; \quad Y = \{x; y; t\}$$

и нормальную последовательность

$$\begin{aligned} u &:= x + y, \\ v &:= u \cdot y, \\ x &:= v - u, \\ t &:= x - v^2, \\ x &:= y + t, \\ y &:= x - 2v. \end{aligned}$$

Комплекс K будет состоять из следующих x-звеньев:



Принадлежащие α-звену элементы комплекса будем называть элементами α-звена. Удобно считать элементы α-звена перенумерованными в порядке возрастания номеров, приписанных им как элементам комплекса.

Совокупность элементов α-звена, состоящая из его первого элемента и некоторого количества следующих за этим элементом формул, будем обозначать символом M_α.

Совокупность некоторого количества последовательных формул, принадлежащих звену, и непосредственно следующего за ними последнего элемента α-звена будем обозначать символом N_α.

Предположим, что в комплексе K присутствуют α-звено и β-звено, удовлетворяющие следующим условиям:

1) элемент, являющийся первым в α-звене, предшествует элементу, являющемуся первым в β-звене;

2) между упомянутыми элементами расположен только один β-барьер и ни одного α-барьера.

Отбросим первый элемент β-звена и найдем общую часть остатка β-звена и α-звена. Полученную совокупность элементов обозначим через $O_{\alpha,\beta}$.

Подчеркнем, что первым элементом в M_α является либо входной кортеж X (который может содержать, а может и не содержать букву α_{s_i}, одинаковую с α), либо некоторая буква α_{t_i}, одинаковая с α. Остальные левые части формул, входящие в M_α, отличны от α. Левые части формул, вхо-

дящие в N_α, все отличны от α, а входящие в $O_{\alpha,\beta}$, все отличны и от α и от β.

Записывая справа от какого-либо из символов M_α, N_α, $O_{\alpha,\beta}$ или $:=\varphi$ в скобках букву δ, мы будем отмечать, что в элементах множества, обозначенного этим символом, быть может, присутствует буква δ. Чтобы показать отсутствие буквы δ в элементах множества, обозначенного упомянутым символом, будем записывать в скобках $\bar{\delta}$.

Производя замену букв в элементах множеств M_α, N_α, $O_{\alpha,\beta}$ и $:=\varphi$, мы будем соответствующую замену букв производить и внутри скобок, стоящих справа от каждого из этих символов. При некоторых заменах букв может происходить изменение индекса в символах M_α, N_α, $O_{\alpha,\beta}$. Чтобы показать замену буквы (например, δ) другой буквой (например, буквой γ) *всюду или в некоторых местах*, мы будем заменяемую букву внутри скобок повторять дважды и заменять новой буквой только одну из повторяющихся букв; например, замена в некоторых местах или всюду буквы δ на γ в элементах множества $O_{\alpha,\beta}$ изображается так: до замены букв пишем $O_{\alpha,\beta}(\delta, \delta)$, и после замены букв $O_{\alpha,\beta}(\delta, \gamma)$ или $O_{\alpha,\beta}(\gamma, \delta)$.

Преобразования А-5 и обратное ему В-5:

$$\begin{bmatrix} M_\alpha(\alpha, \bar{\beta}) \\ \beta := \varphi(\alpha, \bar{\beta}) \\ N_\alpha(\bar{\alpha}) \end{bmatrix} \rightleftarrows \begin{bmatrix} M_\beta(\beta, \bar{\alpha}) \\ \beta := \varphi(\beta, \bar{\alpha}) \\ N_\alpha(\bar{\alpha}) \end{bmatrix}.$$

Преобразования А-6 и обратное ему В-6:

$$\begin{bmatrix} \alpha := \varphi \\ N_\alpha(\bar{\alpha}) \end{bmatrix} \rightleftarrows \begin{bmatrix} N_\alpha(\bar{\alpha}) \end{bmatrix}.$$

Преобразования А-7 и обратное ему В-7:

$$\begin{bmatrix} \alpha := \varphi(\bar{\alpha}) \\ \beta := \varphi(\bar{\alpha}) \\ O_{\alpha,\beta}(\alpha, \alpha, \beta) \end{bmatrix} \rightleftarrows \begin{bmatrix} \alpha := \varphi(\bar{\alpha}) \\ \beta := \varphi(\bar{\alpha}) \\ O_{\alpha,\beta}(\alpha, \beta, \beta) \end{bmatrix}.$$

Если преобразование А-7 применяется к комплексу, для которого $\alpha \in Y$ и $\beta \in Y$, и после его применения в кортеже

Y окажется дважды повторенной ячейка β (и, следовательно, исчезнет ячейка α), то одну из ячеек β из кортежа Y необходимо вычеркнуть.

Если к комплексу применяется преобразование В-7, затрагивающее его выходной кортеж Y, причем $\beta \in Y$, то перед выполнением В-7 в кортеж Y можно включить «второй экземпляр» ячейки β, с тем чтобы (при выполнении В-7) его заменить ячейкой α.

Пример 5.6. Рассмотрим комплекс

$$\begin{array}{l|l|}
\hline
 & \{x;\ y\} \\
 & u := x + 5 \\
 & v := u - y \\
 & z := v - u \\
\hline
x & := v + u \\
\hline
 & \{x;\ z\}
\end{array}.$$

К его первому x-звену можно применить преобразование А-5

$$\begin{bmatrix} M_x(x,\ \bar{v}) \\ v := u - y \\ N_x(\bar{x}) \end{bmatrix} \to \begin{bmatrix} M_v(v,\ \bar{x}) \\ v := u - y \\ N_x(\bar{x}) \end{bmatrix},$$

понимая формулу $v := u - y$ как $v := \varphi(x,\ \bar{v})$. В результате получится комплекс

$$\begin{gathered}
\{v;\ y\}, \\
u := v + 5, \\
v := u - y, \\
z := v - y, \\
x := v + u, \\
\{x;\ z\}.
\end{gathered}$$

Преобразования А-8 и обратное ему В-8. Если выполнены условия $\alpha_{s_j} \neq sa\,\varphi_i$ при $i = 1, 2, \ldots, N$ и $\alpha_{s_j} \overline{\in} Y$, то

$$[\alpha_{s_j}] \rightleftarrows [\Lambda],$$

т. е. при указанных условиях букву можно вычеркнуть из кортежа X или, наоборот, включить ее в кортеж.

Преобразования А-9 и обратное ему В-9. Если нормальная последовательность комплекса заканчивается

формулой $\alpha_{r_k} = \alpha_{r_l}$, причем $\alpha_{r_l} \in Y$, то

$$[\alpha_{r_k}] \rightleftarrows [\Lambda],$$

т. е. при указанных условиях букву α_{r_k} можно вычеркнуть из кортежа Y или, наоборот, включить ее в этот кортеж (если она в нем отсутствует).

Вышеприведенные шестнадцать эквивалентных преобразований образуют *полную систему* в том смысле, что если два комплекса эквивалентны, то любой из них можно с помощью конечного числа таких преобразований свести к другому. Следующие преобразования могут быть выведены из предыдущих:

$$\begin{bmatrix} \alpha_k := \varphi_i \\ \alpha_k := \varphi_{i+1}(\alpha_k,\ \alpha_{j\neq k}) \end{bmatrix} \rightleftarrows [\alpha_k := \varphi_{i+1}(\varphi_i,\ \alpha_{j\neq k})].$$

Если θ — нормальная последовательность комплекса K и если состояния входного кортежа X, принадлежащие области задания этого комплекса, удовлетворяют соотношению $\alpha_j \equiv \psi(\alpha_1, \alpha_2, \ldots, \alpha_n)$, то

$$[\theta] \rightleftarrows \begin{bmatrix} \alpha_j := \psi \\ \theta \end{bmatrix},$$

т. е. впереди нормальной последовательности комплекса K можно приписать формулу $\alpha_j = \psi$ или вычеркнуть формулу нормальной последовательности $\alpha_j = \psi$, если она является первой.

8. Сложение комплексов. Предположим, что комплексы K_1 и K_2 отличаются только (быть может) своими областями задания G_1 и G_2, причем их входные кортежи, нормальные последовательности и выходные кортежи — соответственно графически одинаковы. Тогда комплекс K, отличающийся от каждого из комплексов K_1, K_2 только (быть может) своей областью задания G, где

$$G = G_1 + G_2,$$

называется *слабой суммой* комплексов K_1 и K_2. Операция, с помощью которой по комплексам K_1 и K_2 получается их слабая сумма, называется *слабым сложением* комплексов

и обозначается знаком «$+$». При этом пишут

$$K = K_1 + K_2.$$

Пусть даны комплексы K_1 и K_2 и существует комплекс K такой, что $K = K_1' + K_2'$, причем $K_1' \sim K_1$, $K_2'' \sim K_2$. Комплекс K будем называть *суммой* комплексов K_1 и K_2 и будем писать

$$K \sim K_1 + K_2.$$

Слабая сумма комплексов является частным случаем их суммы.

Операция, с помощью которой по комплексам K_1 и K_2 получается их сумма, называется сложением, а сами комплексы K_1 и K_2 — *слагаемыми*.

Сложение комплексов подчиняется *переместительному закону*

$$K_1 + K_2 \sim K_2 + K_1,$$

а также *сочетательному закону*

$$K_1 + (K_2 + K_3) \sim (K_1 + K_2) + K_3.$$

Последнее позволяет в многочленных суммах опускать скобки; например, можно писать

$$K_1 + K_2 + K_3.$$

Имея некоторый комплекс K с областью задания G, можно разбить G на две непересекающиеся части G_1 и G_2 так, чтобы было

$$G = G_1 + G_2,$$

и рассматривать комплексы K_1 и K_2 с областями задания соответственно G_1 и G_2, каждый из которых отличается от K лишь своей областью задания. При этом, очевидно,

$$K = K_1 + K_2.$$

Описанная операция, приводящая к получению для комплекса K двух комплексов K_1 и K_2, суммой которых является K, называется *расчленением* комплекса K на слагаемые. Ввиду того, что область задания G комплекса K можно различными способами разделять на две непересекающиеся части, расчленение комплекса K на слагаемые не является однозначным.

9. Умножение комплексов. Пусть комплекс K_1 имеет область задания G_1 и нормальную последовательность θ_1 с входным кортежем X_1, выходным кортежем Y_1 и рабочим кортежем Z_1, а комплекс K_2 имеет область задания G_2 и нормальную последовательность θ_2 с входным кортежем X_2, выходным кортежем Y_2 и рабочим кортежем Z_2. Предположим, что выполнено условие: пересечения $X_2 \cdot Z_1$ и $Y_1 (Z_2 - X_2)$ являются пустыми. Обозначим

$$X = X_1 + (X_2 - Y_1); \quad Y = Y_2 + (Y_1 - Z_2).$$

Легко видеть, что столбец формул, получающийся, если к нормальной последовательности θ_1 приписать нормальную последовательность θ_2, представляет собой новую нормальную последовательность θ с входным кортежем X и выходным кортежем Y.

Пусть G означает совокупность состояний кортежа X, удовлетворяющую следующим условиям.

1. Каково бы ни было состояние Φ из G, в G_1 и G_2 найдутся соответственно состояния Φ_1 и Φ_2 такие, что

$$\Phi_1(\alpha) = \Phi(\alpha) \quad \text{при } \alpha \in X_1,$$
$$\Phi_2(\alpha) = \Phi(\alpha) \quad \text{при } \alpha \in X_2 - Y_1.$$

2. Каковы бы ни были состояния Φ_1 и Φ_2, входящие соответственно в G_1 и G_2, если $\Phi_1(\alpha) = \Phi_2(\alpha)$ при $\alpha \in X_1 \cdot (X_2 - Y_1)$, то в G содержится состояние Φ такое, что

$$\Phi_1(\alpha) = \Phi(\alpha) \quad \text{при } \alpha \in X_1,$$
$$\Phi_2(\alpha) = \Phi(\alpha) \quad \text{при } \alpha \in X_2 - Y_1.$$

Если произведение n, $1+0$-операторов, изображаемое столбцом формул θ, определено на множестве G состояний кортежа X, то нами получен некоторый комплекс K, который мы будем называть *произведением* комплексов K_1 и K_2 и записывать в виде $K_1 K_2$.

Операция, с помощью которой по двум комплексам получается их произведение, называется *умножением* комплексов.

Условие, которое было нами выше наложено на кортежи комплексов K_1 и K_2, означает, что промежуточные результаты оператора A_1, изображаемого комплексом K_1, не должны быть исходными данными для оператора A_2, отвечающего комплексу K_2, а промежуточные результаты комплекса K_2

не должны уничтожать результаты комплекса K_1, если последние не были использованы в качестве исходных данных комплекса K_2. Заметим, что оператор A, соответствующий комплексу K_1K_2, является произведением операторов A_1 и A_2.

Можно говорить о произведении нескольких (более чем двух) комплексов, если для каждого предыдущего и для каждого за ним следующего (не только для непосредственно следующего) выполнены вышеприведенные условия, налагаемые на кортежи.

Для умножения комплексов справедлив *сочетательный закон.*

$$K_1(K_2K_3) \sim (K_1K_2)K_3.$$

Это позволяет в многочленных произведениях опускать скобки. Например, можно писать

$$K_1K_2K_3.$$

Переместительному закону умножение комплексов *не подчиняется.*

Пусть K — некоторый комплекс с нормальной последовательностью θ. Можно разбить (в порядке возрастания номеров формул) нормальную последовательность θ на части $\theta_1, \theta_2, \ldots, \theta_k$ и рассматривать комплексы $K_1, K_2, \ldots, K_k$ с нормальными последовательностями $\theta_1, \theta_2, \ldots, \theta_k$ и соответствующим образом выбранными входными и выходными кортежами. Иногда будет требоваться предварительное несложное преобразование комплекса K, которое обеспечило бы выполнение соответствующих условий для кортежей образуемых комплексов. При этом будет

$$K = K_1K_2 \ldots K_k.$$

Описанный процесс будем называть *расчленением комплекса K на сомножители.* Очевидно, расчленение комплекса на сомножители можно производить различными способами.

10. Единичные и обратные комплексы. Пусть K — комплекс с областью задания G, для которого $X = Y$. Если K ставит в соответствие каждому состоянию кортежа X, входящему в G, это же состояние (кортежа Y), то комплекс K называется *единичным.* При этом пишут

$$K \sim \Lambda.$$

Легко видеть, что если $K \sim \Lambda$, то $K \sim K'$, где K' имеет ту же область задания G, что и K, и нормальную последовательность

$$\alpha_{s_i} = \alpha_{s_i} \qquad (i = 1, 2, \ldots, m).$$

Пусть K_n — произвольный комплекс. Если существует такой комплекс K_{-n}, что

$$K_n K_{-n} = \Lambda,$$

то K_{-n} называется *обратным справа* для K_n, а K_n — *обратным слева* для K_{-n}.

Если комплекс K_{-n} является для комплекса K_n обратным справа и обратным слева, т. е. если

$$K_n K_{-n} \sim K_{-n} K_n \sim \Lambda,$$

то комплекс K_{-n} называют *обратным* для K_n.

11. Пакеты комплексов. Эквивалентность пакетов. Пусть $K_1, K_2, \ldots, K_i, \ldots$ — конечное или счетное множество комплексов с общим входным кортежем X и непересекающимися областями задания $G_1, G_2, \ldots, G_i, \ldots$ Обозначим

$$G = G_1 + G_2 + \ldots + G_i + \ldots$$

Совокупность комплексов $K_1, K_2, \ldots, K_i, \ldots$ называется *пакетом комплексов* с входным кортежем X и областью задания G.

Два пакета комплексов назовем *эквивалентными*, если с помощью операций расчленения комплексов на слагаемые и объединения комплексов в суммы каждый из пакетов можно свести к новому пакету таким образом, что каждому комплексу первого нового пакета взаимно однозначно соответствует эквивалентный ему комплекс второго нового пакета.

§ 2. Формальное определение логической схемы. Равносильность логических схем

1. Основные типы операторов. Перед тем как приступить к выделению некоторых типов операторов, введем понятие n-функции. Пусть G — некоторое множество состояний памяти Ω, X — кортеж, состоящий из n ячеек этой памяти, а $U = \{u\}$ — множество элементов u произвольной природы.

Рассмотрим функцию

$$u = \mu^*(f)\colon\ G \to U',$$

где $U' \subseteq U$.

Функция $u = \mu^*(f)$ называется n-функцией с входным кортежем X, если она удовлетворяет следующим двум условиям:

1) если f_1 и f_2 входят в G и если $f_1(\alpha) = f_2(\alpha)$ при $\alpha \in X$, то $\mu^*(f_1) = \mu^*(f_2)$;

2) если $f_1 \in G$ и $f_2(\alpha) = f_1(\alpha)$ при $\alpha \in X$, то $f_2 \in G$.

Первое из приведенных условий означает, что значения функции $\mu^*(f)$ целиком определяются состояниями ячеек, принадлежащих кортежу X. Если $X = \{\alpha_i\}_{i=1}^{n}$, то n-функцию можно представить в виде

$$u = \mu(f(\alpha_1),\ f(\alpha_2),\ \ldots,\ f(\alpha_n)).$$

Опуская в последней записи для упрощения символы $f(\ )$, как мы уже делали ранее для n, $1+0$-операторов, получим

$$u = \mu(\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_n).$$

Вместо последней записи мы, однако, будем пользоваться более удобной для нас записью

$$u = \mu^{\alpha_1,\ \alpha_2\ \ldots,\ \alpha_n},$$

которой в тех случаях, когда

$$\alpha_j = \varphi_j(\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_k) \qquad (j = 1,\ 2,\ \ldots,\ n),$$

будем придавать вид

$$u = \mu^{\varphi_1(\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_k),\ \varphi_2(\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_k),\ \ldots,\ \varphi_n(\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_k)}.$$

Второе условие, которому подчинена n-функция, означает, что все состояния памяти Ω, совпадающие друг с другом на кортеже X, входят в область задания функции $\mu^*(f)$, если в нее входит хотя бы одно из них. Это позволяет считать областью задания функции $\mu^*(f)$ множество состояний кортежа X, а не множество состояний всей памяти Ω (в дальнейшем так и будем поступать).

Пусть Ω_x, Ω_i и Ω_ω являются частями памяти Ω (так, что $\Omega_x + \Omega_i + \Omega_\omega \subset \Omega$). Будем их называть соответственно *памятью величин, памятью параметров* и *памятью сигналов*. Будем ячейки памяти величин обозначать символами

x_1, x_2, ..., а состояния этой памяти или принадлежащих ей кортежей ячеек — символами вида $g(x)$. Ячейки памяти параметров будем обозначать символами i_1, i_2, ..., а ее состояния или состояния принадлежащих ей кортежей — символами вида $j(i)$. При этом будем считать, что состояниями ячеек памяти параметров являются целые числа. Наконец, будем считать, что память сигналов состоит только из одной ячейки ω, которая может находиться лишь в одном из двух состояний: 0 или 1.

Пусть A — некоторый оператор с входным кортежем X и выходным кортежем Y, причем выполнено условие

$$X+Y\subseteq\mathfrak{Q}_x+\mathfrak{Q}_i+\mathfrak{Q}_\omega.$$

Будем оператор A называть

действующим, если $Y\subseteq\mathfrak{Q}_x$;

варьирующим, если $Y\subseteq\mathfrak{Q}_i$;

логическим, если $Y=\mathfrak{Q}_\omega$.

Ограничимся изучением случаев, в которых операторы удовлетворяют следующим дополнительным условиям:

1) входные кортежи действующих операторов принадлежат памяти величин, т. е. $X\subseteq\mathfrak{Q}_x$;

2) входные кортежи варьирующих операторов принадлежат памяти параметров, т. е. $X\subseteq\mathfrak{Q}_i$;

3) нормальная последовательность комплекса, соответствующего логическому оператору, состоит из одной формулы.

Действующий оператор будем либо изображать в виде записи комплекса (которая содержит входной кортеж X, нормальную последовательность формул и выходной кортеж Y), либо будем его обозначать буквой D, снабженной снизу целым положительным числовым индексом. Например, действующие операторы могут быть изображены символом D_1, D_2 и т. п.

Варьирующий оператор либо будем представлять в виде записи его комплекса, либо будем обозначать символом V_ν, где ν — натуральное число.

Логический оператор либо будем изображать в виде записи комплекса, либо будем его обозначать буквой P, снабженной снизу целым положительным числовым индексом.

Например, логические операторы могут изображаться символами P_1, P_5 и т. п.

Нормальная последовательность комплекса, соответствующего логическому оператору, сводится к формуле, в левой части которой всегда стоит буква ω. Поэтому без ущерба для ясности левую часть формулы и знак присваивания можно опустить и представлять формулу ее правой частью. Функция, присутствующая в правой части формулы, может принимать лишь одно из двух значений: 0 или 1. Такие функции являются *предикатами*. Соотношение, которому должны удовлетворять аргументы предиката для того, чтобы он был равен 1, и не удовлетворять для того, чтобы он был равен 0, является *содержанием предиката*. Например, если предикат равен 1 при x_1 большем, чем x_2, и равен 0 при x_1 меньшем или равном x_2, то содержанием предиката будет соотношение (неравенство) $x_1 > x_2$. В качестве обозначения предиката (и соответствующего ему логического оператора) будем применять соотношение, являющееся его содержанием. В некоторых случаях предикаты, как и логические операторы, будем обозначать знаками P_ν.

Более сложные предикаты могут быть заданы как функции алгебры логики от более простых предикатов.

Пример 5.7. Если содержанием предиката P_1 является соотношение $\alpha_1 > 0$, а содержанием предиката P_2 — соотношение $\alpha_1^2 + \alpha_2^2 = 1$, то сложный предикат $P_1 \vee \overline{P_2}$ может быть обозначен знаком P_3 или формулой $P_1 \vee \overline{P_2}$ или строкой $\alpha_1 > 0 \vee \overline{\alpha_1^2 + \alpha_2^2 = 1}$. Так как отрицание соотношения $\alpha_1^2 + \alpha_2^2 = 1$ может быть записано в виде соотношения $\alpha_1^2 + \alpha_2^2 \neq 1$, то тот же сложный предикат можно записать в виде формулы $\alpha_1 > 0 \vee \alpha_1^2 + \alpha_2^2 \neq 1$. Логический оператор, отвечающий сложному предикату, обозначаем одинаково с предикатом.

Пусть $I = \{i_k\}_{k=1}^n$ — кортеж ячеек памяти параметров, J — некоторое множество состояний этого кортежа, а U — некоторое множество операторов определенного типа (действующих или логических). Тогда n-функция

$$\mu^{i_1, i_2, \ldots, i_n}: J \to U,$$

имеющая входной кортеж I, называется *оператором, зависящим от параметров* (соответственно действующим или логическим). Варьирующих операторов, зависящих от параметров, вводить в рассмотрение не будем.

Действующий оператор, зависящий от параметров, будем обозначать символом вида $D^{i_1, i_2, \ldots, i_n}$, а логический — символом вида $P^{i_1, i_2, \ldots, i_n}$.

Всякий оператор (не зависящий от параметров) можно рассматривать как оператор, зависящий от параметров, который при любых возможных значениях параметров принимает одно и то же значение. Говоря о различных операторах, зависящих или не зависящих от параметров, будем для краткости называть их просто *операторами*.

2. Понятие логической схемы. Условимся символ $И_0$ называть *оператором начала* и считать его действующим оператором (не зависящим от параметров), выполняющим тождественное преобразование состояния всей памяти величин, а символ $Я_\nu$, где ν — натуральное число, условимся называть *оператором конца*. Смысл этих символов будет пояснен ниже.

Левыми знаками перехода, соответственно нижними и верхними, будем называть символы вида $\underset{z}{\lfloor}$ и $\overset{z}{\lceil}$, а правыми знаками перехода — символы вида $\underset{z}{\rfloor}$, где z — натуральное число.

Введем понятие терма:

1) оператор конца $Я$ является термом;

2) упорядоченная пара нелогического оператора, отличного от оператора конца, и левого нижнего знака перехода является термом;

3) упорядоченная тройка, состоящая из логического оператора и двух взятых в любом порядке левых знаков перехода — верхнего и нижнего, является термом;

4) правый знак перехода является термом;

5) никаких больше термов нет.

Графиком называется упорядоченная слева направо конечная строка термов, удовлетворяющая следующим условиям:

1) терм, содержащий оператор $И_0$, присутствует в этой строке на первом и только на первом (т. е. на самом левом) месте;

2) правые знаки перехода, присутствующие в строке, попарно отличаются друг от друга своими индексами;

3) каждому левому знаку перехода с индексом z, входящему в один из термов строки, соответствует правый знак перехода с тем же индексом z.

Предположим, что заданы: некоторый график Γ; множество G состояний кортежа X, принадлежащего памяти величин; некоторое состояние $j_0(i)$ кортежа I, принадлежащего памяти параметров. Упорядоченную тройку $(\Gamma, G, j_0(i))$ назовем *агрегатом*.

Если для любого входящего в G состояния $g(x)$ кортежа X выполним и заканчивается после конечного числа шагов процесс, определяемый описанным ниже алгорифмом выполнения, то агрегат $(\Gamma, G, j_0(i))$ называется *логической схемой*.

Алгорифм выполнения логической схемы.

1°. Полагаем $k=0$, $g_0=g(x)$, $j_0=j_0(i)$, $\omega_0=0$. Рассматриваем первый терм графика. Приступаем к выполнению п. 2°.

2°. Проверяем, является ли рассматриваемый терм оператором конца. Если да, процесс окончен, если нет, переходим к следующему пункту.

3°. Проверяем, является ли рассматриваемый терм правым знаком перехода. Если да, переходим к п. 4°, если нет, — к п. 5°.

4°. Переходим в графике к ближайшему следующему терму. Возвращаемся к п. 2°.

5°. Определяем значение Z оператора, входящего в рассматриваемый терм. Обозначим это значение соответственно через $И_0$, D, V или P, в зависимости от того, является ли оно оператором начала, действующим, варьирующим или логическим оператором. Полагаем

$$g_{k+1}=\begin{cases} D(g_k), & \text{если } Z=D,\\ g_k & \text{в остальных случаях;}\end{cases}$$

$$j_{k+1}=\begin{cases} V(j_k), & \text{если } Z=V,\\ j_k & \text{в остальных случаях;}\end{cases}$$

$$\omega_{k+1}=\begin{cases} P(g_k, j_k), & \text{если } Z=P,\\ \omega_k & \text{в остальных случаях.}\end{cases}$$

Увеличиваем значение k на 1. Переходим к п. 6°.

6°. Если рассматриваемый терм содержит два левых знака перехода (верхний и нижний), переходим к п. 7°, если только один левый знак перехода (нижний), — к п. 8°.

7°. Если $\omega=1$, рассматриваем левый верхний знак перехода, а если $\omega=0$, то левый нижний знак перехода. Приступаем к выполнению п. 8°.

8°. Находим в графике правый знак перехода, индекс которого одинаков с индексом рассмотренного левого знака перехода. Возвращаемся к п. 4°.

Если $(\Gamma, G, j_0(i))$ является схемой, то G называется *областью задания схемы*, кортеж X, состояния которого входят в G, называется *входным кортежем схемы*, $j_0(i)$ называется начальным состоянием схемы, а Γ — ее графиком. Любое состояние $j(i)$ кортежа I, получающееся в процессе выполнения схемы, называется *состоянием схемы*.

В дальнейшем схему $(\Gamma, G, j_0(i))$ будем обозначать одной буквой Σ.

Выполнением оператора будем называть: а) для действующего оператора — получение g_{k+1} по формуле $g_{k+1} = D(g_k)$ (п. 5 алгорифма выполнения), б) для варьирующего оператора — получение j_{k+1} по формуле $j_{k+1} = V(j_k)$ (п. 5° алгорифма выполнения), в) для оператора конца — прекращение процесса выполнения схемы (п. 2° алгорифма выполнения), г) для логического оператора — получение ω_{k+1} по формуле $\omega_{k+1} = P(g_{k+1}, j_{k+1})$ (п. 5° алгорифма выполнения) и выбор левого знака перехода (пп. 6° и 7° алгорифма выполнения). Оператор начала $И_0$ служит в схеме для указания того оператора, с которого начинается ее выполнение. Выполнением правого знака перехода называется переход к следующему за ним элементарному выражению, а выполнением левого знака перехода с индексом i — отыскание правого знака перехода с тем же индексом.

Из определения схемы следует, что ее график должен содержать хотя бы один оператор конца. В противном случае схема не была бы выполнима ни для одного $g(x)$, т. е. не была бы схемой.

В дальнейшем, говоря о термах и их последовательностях, присутствующих в графике схемы, мы будем для краткости говорить, что они присутствуют в схеме. Производя изменения в графике схемы, мы будем говорить, что производим изменения схемы.

3. Реализация схемы. Равносильность схем. Выражения. Равносильность выражений. После выполнения схемы Σ для некоторого $g(x)$, принадлежащего ее области задания G, мы получаем состояния букв $y_1, y_2, \ldots, y_M$, являющихся ячейками выходного кортежа Y действующего

оператора, последнего при выполнении схемы. Будем говорить, что схема Σ перерабатывает исходное состояние $g(x)$ кортежа X в состояние $\overline{g}(y)$ кортежа Y.

При выполнении схемы можно каждый раз, когда выполняется действующий оператор, выписывать нормальную последовательность его комплекса. При этом после выполнения схемы будет получена некоторая последовательность формул. Выберем все состояния $g(x)$ кортежа X, входящие в область задания схемы (обозначим их совокупность через G_1), которые перерабатываются в состояния одного и того же кортежа Y и приводят к получению одной и той же последовательности формул. Получаемая последовательность формул будет нормальной последовательностью с входным кортежем X и выходным кортежем Y. Взятая вместе с G_1, она представляет комплекс, который назовем *реализацией схемы.*

Реализация схемы описывает произведение значений действующих операторов в том порядке, в котором они выполнялись при выполнении схемы.

Область задания схемы представляет собой сумму непересекающихся множеств, являющихся областями задания ее реализаций.

Легко видеть, что множество реализаций схемы не более чем счетно.

Пусть схема Σ удовлетворяет тому условию, что, каков бы ни был стоящий в ее графике левый знак перехода, отвечающий ему правый знак перехода стоит в графике правее него.

Такую схему будем называть *прямой.*

Очевидно, множество реализаций прямой схемы конечно.

Совокупность реализаций каждой схемы представляет собой пакет комплексов. Две схемы будем называть *равносильными*, если пакеты их реализаций эквивалентны. Для обозначения равносильности схем будем пользоваться знаком равенства. Запись

$$\Sigma = \Sigma'$$

означает, что схема Σ равносильна схеме Σ'. Нетрудно убедиться в том, что знак равносильности обладает

1) свойством рефлексивности: $\Sigma = \Sigma$;

2) свойством симметрии: если $\Sigma = \Sigma'$, то $\Sigma' = \Sigma$;

3) свойством транзитивности: если $\Sigma = \Sigma'$ и $\Sigma' = \Sigma''$, то $\Sigma = \Sigma''$.

Выражением будем называть конечную строку термов, в которой для каждого натурального числа z имеется не больше одного правого знака перехода с этим индексом, и терм, содержащий $И_0$, может присутствовать только на первом месте. В дальнейшем, если это специально не оговорено, считаем, что выражение не содержит оператора $И_0$.

Содержательно выражения следует рассматривать как куски графиков схем.

Термы являются наиболее простыми выражениями и потому называются *элементарными выражениями.*

Символом Λ обозначим пустое выражение. Считается, что пустое выражение присутствует после любого элементарного выражения графика схемы. Для того чтобы отметить вхождение в график схемы Σ на определенных местах выражений $W_1, W_2, \ldots, W_n$, будем писать $\Sigma\{W_1, W_2, \ldots, W_n\}$. Агрегат, который получится при замене в последней схеме выражений $W_1, W_2, \ldots, W_n$ соответственно выражениями $W_1', W_2', \ldots, W_n'$, будем обозначать $\Sigma\{W_1', W_2', \ldots, W_n'\}$.

Два выражения W_1 и W_2 называются *равносильными*, если, коль скоро $\Sigma\{W_1\}$ — схема, то:

1) $\Sigma\{W_2\}$ тоже схема и 2) эти схемы равносильны. Для обозначения равносильности выражений будем применять знак равенства.

Если некоторое изменение схемы Σ приводит к получению новой схемы Σ', такой, что $\Sigma = \Sigma'$, то будем говорить, что схему Σ *можно* подвергнуть упомянутому изменению, а это изменение будем называть *равносильным* преобразованием.

Равносильность выражений обладает свойствами симметрии, рефлексивности и транзитивности, однако иногда замена формулы $W_1 = W_2$ формулой $W_2 = W_1$ требует некоторых оговорок, которые в дальнейшем приводятся лишь в случаях, когда они не являются совершенно очевидными.

§ 3. Основные равносильные преобразования логических схем

1. Равносильные преобразования, не связанные с внутренними свойствами операторов. Ниже приведены равносильные преобразования схем, обусловленные свойствами их алгорифма выполнения.

Правые знаки перехода, стоящие рядом, можно менять местами:

$$\underset{i}{\lrcorner}\,\underset{j}{\lrcorner} = \underset{j}{\lrcorner}\,\underset{i}{\lrcorner}.$$

Если в схеме отсутствует левый знак перехода с индексом, равным k, то правый знак перехода с этим индексом можно вычеркнуть:

$$\underset{k}{\lrcorner} = \Lambda. \tag{5.6}$$

Левые знаки перехода (верхний и нижний), стоящие непосредственно после логического оператора, можно менять местами

$$P \overset{i}{\ulcorner}\underset{i}{\llcorner} = P \underset{i}{\llcorner}\overset{j}{\ulcorner}.$$

Последняя равносильность позволяет вместо $\overset{j}{\ulcorner}\underset{i}{\llcorner}$ писать $\underset{i}{\overset{j}{\sqcap}}$. Таким образом,

$$P \overset{j}{\ulcorner}\underset{i}{\llcorner} = P \underset{i}{\llcorner}\overset{j}{\ulcorner} = P \underset{i}{\overset{j}{\sqcap}}.$$

Любой из знаков $\underset{i}{\llcorner}$, $\overset{i}{\ulcorner}$ будем обозначать через λ_i, а элементарное выражение, содержащее λ_i, — через $R\lambda_i$, понимая под R совокупность всех символов элементарного выражения, кроме λ_i.

Если в схеме $\Sigma\{R\lambda_i, \underset{i}{\lrcorner}\}$ отсутствуют знаки перехода с индексом j, то

$$\Sigma\{R\lambda_i, \underset{i}{\lrcorner}\} = \Sigma\{R\lambda_j, \underset{j}{\lrcorner}\,\underset{i}{\lrcorner}\}.$$

Для упрощения записи логических схем принято соглашение

$$R\lambda_i \underset{i}{\lrcorner} = R \underset{i}{\lrcorner}. \tag{5.7}$$

Если, кроме того, в схеме кроме λ_i нет других левых знаков перехода с индексом i, то в силу (5.6)

$$R\lambda_i \underset{i}{\lrcorner} = R. \tag{5.8}$$

Соглашение (5.7) означает справедливость формул

$$D\underset{i}{\lfloor}\underset{i}{\rfloor}=D\underset{i}{\rfloor},$$

$$V\underset{i}{\lfloor}\underset{i}{\rfloor}=V\underset{i}{\rfloor},$$

$$P\overset{j}{\underset{i}{\lceil}}\underset{i}{\rfloor}=P\overset{j}{\underset{i}{\lceil}}\underset{i}{\rfloor},$$

$$P\overset{j}{\underset{i}{\lceil}}\underset{j}{\rfloor}=P\underset{i}{\lfloor}\underset{j}{\rfloor}.$$

Пример 5.8. Приведенные выше равносильности значительно упрощают запись графика схемы. Так, вместо

$$И_0\underset{1}{\lfloor}\underset{1}{\rfloor}D_1\underset{2}{\lfloor}\underset{2}{\rfloor}P_1\overset{4}{\underset{3}{\lceil}}\underset{4}{\rfloor}D_2\underset{5}{\lfloor}\underset{3}{\rfloor}\underset{5}{\rfloor}D_3\underset{6}{\lfloor}\underset{6}{\rfloor}Я$$

можно написать

$$И_0D_1P_1\underset{3}{\lfloor}D_2\underset{3}{\rfloor}D_3Я.$$

Если все элементарные выражения графика схемы записаны с полными наборами входящих в них левых знаков перехода, то мы будем говорить, что в графике все знаки перехода присутствуют *явно*. При сокращенной записи графика схемы (в соответствии с соглашением (5.7)) говорят, что некоторые левые, а может быть и правые, знаки перехода присутствуют *неявно*.

Будем говорить, что левые знаки перехода, входящие в то же элементарное выражение, что и оператор Q, *принадлежат* оператору Q.

Просмотрим справа налево символы графика схемы, начиная с оператора Q, до тех пор, пока не встретим элементарное выражение A, содержащее какой-либо оператор. Введем явно все левые знаки перехода, входящие в A (если этого нет). Очевидно, *после* элементарного выражения A и *перед* оператором Q будет стоять некоторое количество правых знаков перехода. Эти правые знаки перехода будем называть *принадлежащими* оператору Q.

Последовательность элементарных выражений, состоящую из всех правых знаков перехода, принадлежащих оператору Q, и элементарного выражения, содержащего оператор Q (в котором все левые знаки перехода присутствуют явно), будем называть *замыканием* оператора Q.

Пусть каждый из знаков T_1 и T_2 обозначает замыкание оператора. Тогда

$$\Sigma\{T_1,\ T_2\}=\Sigma\{T_1T_2,\ \Lambda\}. \tag{5.9}$$

Рассмотрим схему $\Sigma\{TW\}$, где T — замыкание оператора, а W — произвольное выражение. Если все индексы правых знаков перехода, присутствующих в TW, отличны от индексов левых знаков перехода, присутствующих в остальной части графика схемы $\Sigma\{TW\}$, то

$$TW=\Lambda. \tag{5.10}$$

Последняя равносильность справедлива и для случая, когда W является пустым выражением.

Пусть w — элементарное выражение, входящее в выражение TW равносильности (5.10), не являющееся правым знаком перехода. Тогда

$$w=\Lambda.$$

Из (5.10) вытекает справедливость равносильностей

$$D\underset{i}{\llcorner}\,Q\underset{k}{\llcorner}=D\underset{i}{\llcorner},\qquad P\overset{j}{\underset{i}{\sqsubset}}\,Q\underset{k}{\llcorner}=P\overset{j}{\underset{i}{\sqsubset}},$$

$$D\underset{i}{\llcorner}\,P\overset{l}{\underset{k}{\sqsubset}}=D\underset{i}{\llcorner},\qquad P\overset{j}{\underset{i}{\sqsubset}}\,P_1\overset{l}{\underset{k}{\sqsubset}}=P\overset{j}{\underset{i}{\sqsubset}},$$

$$D\underset{i}{\llcorner}\,Я=D\underset{i}{\llcorner},\qquad P\overset{j}{\underset{i}{\sqsubset}}\,Я=P\overset{j}{\underset{i}{\sqsubset}},$$

$$V\underset{i}{\llcorner}\,Q\underset{k}{\llcorner}=V\underset{i}{\llcorner},\qquad ЯQ\underset{k}{\llcorner}=Я,$$

$$V\underset{i}{\llcorner}\,P\overset{l}{\underset{k}{\sqsubset}}=V\underset{i}{\llcorner},\qquad ЯP\overset{l}{\underset{k}{\sqsubset}}=Я,$$

$$V\underset{i}{\llcorner}\,Я=V\underset{i}{\llcorner},\qquad ЯЯ_1=Я.$$

Пример 5.9. Рассмотрим схему

$$\Sigma=И_0D_1\underset{1}{\llcorner}\underset{2}{\lrcorner}\,D_2\underset{3}{\llcorner}\underset{1}{\lrcorner}\,P_3\overset{5}{\underset{4}{\sqsubset}}\underset{3}{\lrcorner}\,D_6\underset{2}{\llcorner}\underset{4}{\lrcorner}\,D_7\underset{5}{\lrcorner}\,D_8Я_9.$$

С помощью (5.9) поменяем местами $\underset{1}{\lrcorner}\,P_3\overset{5}{\underset{4}{\sqsubset}}$ (замыкание оператора P_3) и $\underset{3}{\lrcorner}\,D_6\underset{2}{\llcorner}$ (замыкание оператора D_6). Получим

$$\Sigma=И_0D_1\underset{1}{\llcorner}\underset{2}{\lrcorner}\,D_2\underset{3}{\llcorner}\underset{3}{\lrcorner}\,D_6\underset{2}{\llcorner}\underset{1}{\lrcorner}\,P_3\overset{5}{\underset{4}{\sqsubset}}\underset{4}{\lrcorner}\,D_7\underset{5}{\lrcorner}\,D_8Я_9.$$

Учитывая, что в этой схеме $D_2 \underset{3}{\llcorner}\,\underset{3}{\lrcorner} = D_2$ и $P_3 \overset{5}{\underset{4}{\sqsubset}}\,\underset{4}{\lrcorner} = P_3 \overset{5}{\sqsubset}$, можем написать

$$\Sigma = И_0 D_1 \underset{1}{\llcorner}\,\underset{2}{\lrcorner}\, D_2 D_6 \underset{2}{\llcorner}\,\underset{1}{\lrcorner}\, P_3 \overset{5}{\sqsubset}\, D_7 \underset{5}{\lrcorner}\, D_8 Я_9.$$

Здесь $\underset{2}{\lrcorner}\, D_2 D_6 \underset{2}{\llcorner}$ представляет собой выражение TW, указанное в (5.10) и не содержащее правых знаков перехода с индексами, равными индексам левых знаков перехода, стоящих в схеме вне этого выражения. Следовательно, $\underset{2}{\lrcorner}\, D_2 D_6 \underset{2}{\llcorner} = \Lambda$, откуда

$$\Sigma = И_0 D_1 \underset{1}{\llcorner}\,\underset{1}{\lrcorner}\, P_3 \overset{5}{\sqsubset}\, D_7 \underset{5}{\lrcorner}\, D_8 Я_9.$$

Учитывая, что $D_1 \underset{1}{\llcorner}\,\underset{1}{\lrcorner} = D_1$, окончательно получим

$$\Sigma = И_0 D_1 P_3 \overset{5}{\sqsubset}\, D_7 \underset{5}{\lrcorner}\, D_8 Я_9.$$

Пусть W — некоторое выражение, в котором все знаки перехода присутствуют явно.

Левый знак перехода, принадлежащий W, будем называть *внутренним*, если в этом выражении присутствует правый знак перехода с тем же индексом. Остальные левые знаки перехода, содержащиеся в W, будем называть *внешними*.

Предположим, что существует натуральное число n, такое, что для W выполнены следующие условия:

1) каков бы ни был оператор Q, принадлежащий W, в этом выражении содержится еще $n-1$ операторов, одинаковых с ним;

2) если некоторому оператору Q из W принадлежит внешний левый знак перехода с индексом i, то такой же левый знак перехода (и с тем же индексом i) принадлежит каждому оператору из W, одинаковому с Q;

3) если λ_i и λ_j — внутренние левые знаки перехода, принадлежащие одинаковым операторам из W, то правые знаки перехода $\underset{i}{\lrcorner}$ и $\underset{j}{\lrcorner}$ тоже принадлежат одинаковым операторам из W (или одному и тому же оператору).

Выражение W, удовлетворяющее вышеприведенным условиям, называется *n-кратным*. Присутствующие в нем одинаковые операторы называют *подобными*.

Выражение W, в котором не все знаки перехода присутствуют явно, но которое с помощью формул (5.7) и (5.8) сводится к n-кратному, тоже называется n-кратным.

Пример 5.10. Выражение

$$\underset{4}{\lrcorner} D_1 \underset{2}{\llcorner} \underset{3}{\lrcorner} \underset{1}{\lrcorner} P_1 \overset{10}{\underset{4}{\sqsubset}} \underset{5}{\lrcorner} D_1 \underset{3}{\llcorner} \underset{2}{\lrcorner} P_1 \overset{10}{\underset{5}{\sqsubset}}$$

является двукратным. Вместе с каждым оператором (D_1 и P_1) в этом выражении присутствует еще один подобный ему оператор. Левые верхние знаки перехода $\overset{10}{\sqcap}$, принадлежащие операторам P_1, являются внешними, а все остальные левые знаки перехода — внутренними.

Любой правый знак перехода, принадлежащий Q — одному из операторов п-кратного выражения, можно перенести и поставить перед любым оператором Q', подобным оператору Q. При этом получается новое п-кратное выражение.

Совокупность правых знаков перехода, входящих в замыкание какого-либо оператора, обозначим буквой π.

Из предыдущего правила вытекают равносильности

$$\left.\begin{aligned} \Sigma\{\underset{k}{\lrcorner} Q \underset{i}{\llcorner}, \pi Q \underset{i}{\llcorner}\} &= \Sigma\{Q \underset{i}{\llcorner}, \underset{k}{\lrcorner} \pi Q \underset{i}{\llcorner}\}, \\ \Sigma\{\underset{k}{\lrcorner} P \overset{j}{\underset{i}{\sqsubset}}, \pi P \overset{j}{\underset{i}{\sqsubset}}\} &= \Sigma\{P \overset{j}{\underset{i}{\sqsubset}}, \underset{k}{\lrcorner} \pi P \overset{j}{\underset{i}{\sqsubset}}\}, \end{aligned}\right\} \qquad (5.11)$$

где Q — нелогический, а P — логический операторы. Из последних равносильностей в свою очередь следует, что

$$\left.\begin{aligned} \underset{k}{\lrcorner}\underset{l}{\lrcorner} Q \underset{i}{\llcorner} &= \underset{k}{\lrcorner} Q \underset{i}{\llcorner} \underset{l}{\lrcorner} Q \underset{i}{\llcorner}, \\ \underset{k}{\lrcorner}\underset{l}{\lrcorner} P \overset{j}{\underset{i}{\sqsubset}} &= \underset{k}{\lrcorner} P \overset{j}{\underset{i}{\sqsubset}} \underset{l}{\lrcorner} P \overset{j}{\underset{i}{\sqsubset}}. \end{aligned}\right\} \qquad (5.12)$$

Формула (5.11) справедлива и для случая, когда Q является оператором конца. В этом случае она принимает вид

$$\Sigma\{\underset{k}{\lrcorner} Я_\nu, \pi Я_\mu\} = \Sigma\{Я_\nu, \underset{k}{\lrcorner} \pi Я_\mu\};$$

при этом (5.12) записывается как

$$\underset{k}{\lrcorner}\underset{l}{\lrcorner} Я = \underset{k}{\lrcorner} Я \underset{l}{\lrcorner} Я.$$

Необходимо отметить также следующие правила.

Если в схеме Σ нет знаков перехода с индексом k, то i, являющееся индексом знаков перехода этой схемы, можно заменить на k.

Если в схеме содержатся знаки перехода с индексами k и l, то можно все индексы k заменить на l и одновременно все индексы l заменить на k.

2. Основные преобразования логических операторов. Следующие две равносильности можно отнести еще к числу не связанных с внутренними свойствами операторов:

$$\left.\begin{aligned} \Sigma\{\pi P \overset{k}{\underset{k}{\sqcap}}, \underset{k}{\lrcorner}\} &= \Sigma\{\Lambda, \pi \underset{k}{\lrcorner}\}, \\ P \overset{k}{\underset{k}{\sqcap}} \underset{k}{\lrcorner} &= \underset{k}{\lrcorner}. \end{aligned}\right\} \tag{5.13}$$

Тождественно постоянный предикат и отвечающий ему логический оператор будем обозначать через 0 или 1, в зависимости от того, которому из этих чисел равен упомянутый предикат. К тождественно постоянным можно относить также предикаты, которые при выполнении схемы для любого $g(x)$ из G в момент выполнения отвечающих им логических операторов принимают всегда одни и те же значения. Для тождественно-постоянных предикатов

$$0 \overset{j}{\underset{i}{\sqcap}} = 0 \overset{i}{\underset{i}{\sqcap}},$$

$$1 \overset{j}{\underset{i}{\sqcap}} = 1 \overset{j}{\underset{j}{\sqcap}}.$$

Последние формулы вместе с (5.13) дают

$$0 \overset{j}{\underset{i}{\sqcap}} \underset{i}{\lrcorner} = \underset{i}{\lrcorner},$$

$$1 \overset{j}{\underset{i}{\sqcap}} \underset{j}{\lrcorner} = \underset{j}{\lrcorner},$$

откуда получается

$$0 \overset{j}{\underset{i}{\sqcap}} = P \overset{i}{\underset{i}{\sqcap}},$$

$$1 \overset{j}{\underset{i}{\sqcap}} = P \overset{j}{\underset{j}{\sqcap}}.$$

Если содержания предикатов P_1 и P_2 имеют одни и те же существенные аргументы и для одинаковых комбинаций значений существенных аргументов эти предикаты равны, т. е. $P_1 \equiv P_2$, то

$$P_1 \sqsubset_{i}^{j} = P_2 \sqsubset_{i}^{j}.$$

В частности, это равенство имеет место для случая, когда предикаты P_1 и P_2 логически эквивалентны.

Следующие ниже три формулы позволяют расчленять (отвечающие сложным предикатам) и объединять логические операторы

$$\overline{P} \sqsubset_{i}^{j} = P \sqsubset_{j}^{i},$$

$$P_1 \wedge P_2 \sqsubset_{i}^{j} = P_1 \llcorner_{i} P_2 \sqsubset_{i}^{j},$$

$$P_1 \vee P_2 \sqsubset_{i}^{j} = P_1 \ulcorner^{j} P_2 \sqsubset_{i}^{j}.$$

Большое значение имеют также равносильности

$$\Sigma\{P \sqsubset_{i}^{j}, \lrcorner_{i} P \sqsubset_{k}^{l}\} = \Sigma\{P \sqsubset_{k}^{j}, \lrcorner_{i} P \sqsubset_{k}^{l}\},$$

$$\Sigma\{P \sqsubset_{i}^{j}, \lrcorner_{j} P \sqsubset_{k}^{l}\} = \Sigma\{P \sqsubset_{i}^{l}, \lrcorner_{j} P \sqsubset_{k}^{l}\},$$

из которых вытекают формулы

$$P \sqsubset_{i}^{j} W \lrcorner_{i} P \sqsubset_{k}^{l} = P \sqsubset_{k}^{j} W \lrcorner_{i} P \sqsubset_{k}^{l},$$

$$P \sqsubset_{i}^{j} W \lrcorner_{j} P \sqsubset_{k}^{l} = P \sqsubset_{i}^{l} W \lrcorner_{j} P \sqsubset_{k}^{l},$$

$$\lrcorner_{i} P \sqsubset_{k}^{l} WP \sqsubset_{i}^{j} = \lrcorner_{i} P \sqsubset_{k}^{l} WP \sqsubset_{k}^{j},$$

$$\lrcorner_{j} P \sqsubset_{k}^{l} WP \sqsubset_{i}^{j} = \lrcorner_{j} P \sqsubset_{k}^{l} WP \sqsubset_{i}^{l}.$$

Отметим еще некоторые равносильности, вытекающие из предыдущих:

$$P_1 \underset{i}{\llcorner} P_2 \overset{k}{\underset{j}{\sqsubset}} = P_1 \wedge P_2 \overset{k}{\ulcorner} P_1 \wedge \overline{P}_2 \overset{j}{\underset{i}{\sqsubset}},$$

$$P_1 \overset{i}{\ulcorner} P_2 \overset{k}{\underset{j}{\sqsubset}} = P_1 \vee \overline{P}_2 \underset{k}{\llcorner} P_1 \vee P_2 \overset{i}{\underset{j}{\sqsubset}},$$

$$P_1 \overset{i}{\ulcorner} P_2 \overset{l}{\underset{k}{\sqsubset}} = P_2 \underset{m}{\llcorner} P_1 \overset{i}{\underset{l}{\sqsubset}} \sqcup P_1 \overset{i}{\underset{k}{\sqsubset}},$$

$$P_1 \underset{i}{\llcorner} P_2 \overset{l}{\underset{k}{\sqsubset}} = P_2 \underset{m}{\llcorner} P_1 \overset{l}{\underset{i}{\sqsubset}} \sqcup P_1 \overset{k}{\underset{i}{\sqsubset}}.$$

Если $P_1 = P_1 \wedge P_2$, то

$$P_1 \overset{i}{\ulcorner} P_2 \overset{l}{\underset{k}{\sqsubset}} = P_2 \underset{k}{\llcorner} P_1 \overset{i}{\underset{l}{\sqsubset}}.$$

Если $P_1 \wedge P_2 = 0$, то

$$P_1 \overset{i}{\ulcorner} P_2 \overset{l}{\underset{k}{\sqsubset}} = P_2 \overset{l}{\ulcorner} P_1 \overset{i}{\underset{k}{\sqsubset}}.$$

3. Равносильные преобразования нелогических операторов. Если при каждом состоянии I значениям действующих операторов D_1 и D_2 отвечают эквивалентные комплексы, у которых входные кортежи и выходные кортежи соответственно одинаковы, то

$$D_1 \underset{i}{\llcorner} = D_2 \underset{i}{\llcorner}.$$

При тех же условиях, но без требования, чтобы выходные кортежи комплексов состояли из одних и тех же ячеек,

$$D_1 Я = D_2 Я.$$

Если варьирующим операторам V_1 и V_2 отвечают эквивалентные комплексы, у которых входные кортежи и выходные кортежи соответственно одинаковы, то

$$V_1 \underset{i}{\llcorner} = V_2 \underset{i}{\llcorner}.$$

Если при каждом состоянии I комплекс значения действующего оператора D является произведением комплексов значений операторов D_1 и D_2, то

$$D \underset{i}{\llcorner} = D_1 D_2 \underset{i}{\llcorner}.$$

Если комплекс варьирующего оператора V является произведением комплексов варьирующих операторов V_1 и V_2, то

$$V\underset{i}{\llcorner} = V_1 V_2 \underset{i}{\llcorner}.$$

Последние две равносильности позволяют нелогический оператор расчленить на два или, наоборот (если в равносильностях обменять местами левые и правые части), два последовательных действующих или варьирующих оператора объединить в один.

Если входной и выходной кортежи нелогического оператора D или V при любом состоянии I не пересекаются, то

$$D\underset{i}{\llcorner} = DD\underset{i}{\llcorner},$$

$$V\underset{i}{\llcorner} = VV\underset{i}{\llcorner}.$$

Если при любом состоянии I комплекс значения оператора D_2 обратен справа комплексу значения оператора D_1, то

$$\Sigma\{\pi D_1 D_2 \underset{i}{\llcorner},\ \underset{i}{\lrcorner}\} = \Sigma\{\Lambda,\ \pi \underset{i}{\lrcorner}\}$$

или в частном случае

$$\pi D_1 D_2 \underset{i}{\llcorner}\ \underset{i}{\lrcorner} = \pi \underset{i}{\lrcorner}.$$

Если комплекс оператора V обратен справа комплексу оператора V_1, то

$$\Sigma\{\pi V_1 V_2 \underset{i}{\llcorner},\ \underset{i}{\lrcorner}\} = \Sigma\{\Lambda,\ \pi \underset{i}{\lrcorner}\}$$

или в частном случае

$$\pi V_1 V_2 \underset{i}{\llcorner}\ \underset{i}{\lrcorner} = \pi \underset{i}{\lrcorner}.$$

Если схемы $\Sigma\{\pi V\underset{i}{\llcorner},\ \underset{i}{\lrcorner}\}$ и $\Sigma\{\Lambda,\ \pi\underset{i}{\lrcorner}\}$ не содержат операторов, существенно зависящих от параметров, принадлежащих выходному кортежу оператора V, то

$$\Sigma\{\pi V\underset{i}{\llcorner},\ \underset{i}{\lrcorner}\} = \Sigma\{\Lambda,\ \pi\underset{i}{\lrcorner}\},$$

или в частном случае

$$\pi V \underset{i}{\llcorner} \underset{i}{\lrcorner} = \pi \underset{i}{\lrcorner}.$$

Пусть $i_1, i_2, \ldots, i_t$ — параметры схемы Σ. Можно ввести новый параметр α, удовлетворяющий соотношению

$$\alpha \equiv \varphi(i_1, i_2, \ldots, i_t),$$

и заменить всюду или в некоторых местах в соотношениях, являющихся содержаниями логических операторов, и в индексах операторов, зависящих от параметров, выражение $\varphi(i_1, i_2, \ldots, i_t)$ буквой α. При этом необходимо в конце нормальной последовательности каждого варьирующего оператора приписать формулу

$$\alpha = \varphi(i_1, i_2, \ldots, i_t)$$

и в кортеж I включить ячейку α.

4. Перестановки операторов. Если при каждом состоянии I ячейки выходного кортежа действующего оператора D_i $(i = 1, 2)$ являются несущественными аргументами комплекса оператора D_j $(j = 1, 2;\ j \neq i)$, то

$$D_1 D_2 \underset{k}{\llcorner} = D_2 D_1 \underset{k}{\llcorner}.$$

Если ячейки выходного кортежа варьирующего оператора V_i $(i = 1, 2)$ являются несущественными аргументами комплекса оператора V_j $(j = 1, 2;\ j \neq i)$, то

$$V_1 V_2 \underset{k}{\llcorner} = V_2 V_1 \underset{k}{\llcorner}.$$

Если ячейки выходного кортежа оператора V не являются существенными параметрами действующего оператора D, то

$$VD \underset{i}{\llcorner} = DV \underset{i}{\llcorner}.$$

Пусть K — комплекс действующего оператора D при произвольном состоянии схемы, а X и Y — его входной и выходной кортежи. Пусть, кроме того, $\Psi(Y, Z)$ — содержание (при том же состоянии схемы) логического оператора P, причем Y и Z не пересекаются. Тогда

$$DP \underset{i}{\overset{j}{\sqsubset}} = P' \overset{m}{\sqcap} D \underset{i}{\llcorner} \underset{m}{\lrcorner} D \underset{j}{\llcorner},$$

где P' — логический оператор, содержанием которого является соотношение $\Psi[K(X), Z]$, получаемое из $\Psi(Y, Z)$ путем замены ячеек, входящих в Y, их выражениями через ячейки кортежа X с помощью описанного в § 1, п. 5 алгорифма подстановок.

Если ни при одном состоянии I ячейки выходного кортежа действующего оператора D не являются существенными аргументами предиката логического оператора P, то

$$DP\underset{i}{\overset{j}{\square}} = P\overset{m}{\Gamma} D\underset{i}{\llcorner} \underset{m}{\lrcorner} D\underset{j}{\llcorner}.$$

Пусть $\varphi_1 = \varphi_1(I)$, $\varphi_2 = \varphi_2(I)$, ..., $\varphi_k = \varphi_k(I)$ — верхние индексы действующего оператора, зависящего от параметров, и пусть V — варьирующий оператор, имеющий комплекс K. Обозначим

$$\psi_1 = \varphi_1[K(I)], \quad \psi_2 = \varphi_2[K(I)], \quad \ldots, \quad \psi_k = \varphi_k[K(I)].$$

Тогда

$$VD^{\varphi_1, \varphi_2, \ldots, \varphi_k}\underset{i}{\llcorner} = D^{\psi_1, \psi_2, \ldots, \psi_k} V\underset{i}{\llcorner}.$$

При тех же обозначениях для индексов и для варьирующего оператора, если содержанием логического оператора при произвольном состоянии схемы является $\Psi(I, Z)$, где кортежи I и Z пересекаются, то

$$VP^{\varphi_1, \varphi_2, \ldots, \varphi_k}\underset{i}{\overset{j}{\square}} = P'^{\psi_1, \psi_2, \ldots, \psi_k}\overset{m}{\Gamma} V\underset{i}{\llcorner} \underset{m}{\lrcorner} V\underset{j}{\llcorner}.$$

Здесь $P'^{\psi_1, \psi_2, \ldots, \psi_k}$ — логический оператор с содержанием $\Psi[K(I), Z]$.

5. Подчиненность оператора предикату. Будем буквой w обозначать элементарное выражение, содержащее оператор Q, отличный от $И_0$ (Q может быть как логическим, так и нелогическим).

Будем рассматривать схему $\Sigma\{w\}$, индексы знаков перехода которой отличны от натурального числа α. Обозначим через $\Sigma_\alpha\{w\}$ схему, получающуюся из $\Sigma\{w\}$, если после ее произвольного элементарного выражения вписать $\underset{\alpha}{\lrcorner}$. Очевидно,

$$\Sigma\{w\} = \Sigma_\alpha\{w\}.$$

Если найдется предикат P_0 такой, что при любом положении правого знака перехода $\underset{\alpha}{\lrcorner}$ справедлива равносильность

$$\Sigma\{w\} = \Sigma_\alpha\{P_0 \underset{\alpha}{\llcorner} w\},$$

то говорят, что «элементарное выражение w, а также оператор Q подчинены предикату P_0», и пишут

$$w < P_0; \quad Q < P_0.$$

Содержание предиката P_0, вообще говоря, может иметь в качестве своих аргументов параметры схемы, а также в зависимости от состояния кортежа I может изменять свой вид.

Если в схеме $\Sigma\{w\}$

$$w < P_0,$$

то непосредственно перед выполнением w всегда $P_0 = 1$.

Для того чтобы оператор Q схемы $\Sigma\{w\}$ не выполнялся ни при одном X, принадлежащем G, необходимо и достаточно, чтобы этот оператор был подчинен нулю, т. е. предикату, тождественно равному нулю.

Если $Q < 0$, то $Q < P_0$, где P_0 — любой допустимый для схемы предикат.

Если $Q < P_0$; $Q < P^*$, то

$$Q < P_0 \wedge P^*.$$

Если $Q < P_0$, то при любом допустимом для схемы предикате P^*

$$Q < P_0 \vee P^*.$$

Если логический оператор P удовлетворяет условию $P < P_0$, то справедливы равносильности

$$P\underset{i}{\overset{j}{\lrcorner}} = P_0 \wedge P\underset{i}{\overset{j}{\lrcorner}},$$

$$P\underset{i}{\overset{j}{\lrcorner}} = \overline{P}_0 \vee P\underset{i}{\overset{j}{\lrcorner}}.$$

В схеме $\Sigma\{P \underset{i}{\llcorner} w\}$ имеет место $w < P$.

Если в схеме $\Sigma\{P\lambda_i w\}$ имеет место $P < P_0$, то $w < P_0$.

Если в схеме $\Sigma\{w\}$ оператор Q при всяком ее состоянии подчинен предикату P_0 с содержанием $\alpha_j \equiv \psi(\alpha_1, \alpha_2, \ldots, \alpha_n)$, то

$$w = Q_1 w,$$

где Q_1 — нелогический оператор, которому соответствует комплекс с нормальной последовательностью, состоящей из одной формулы

$$\alpha_j = \psi(\alpha_1, \alpha_2, \ldots, \alpha_n).$$

Если среди букв $\alpha_1, \alpha_2, \ldots, \alpha_n$ нет параметров, то Q_1 является действующим оператором, если все существенные аргументы ψ, а также α_j являются параметрами, то Q_1 является варьирующим оператором. Другие случаи здесь не рассматриваются.

Если в $\Sigma\{w\}$ $Q < P_0$, причем P_0 имеет содержание

$$i_j = \theta(i_1, i_2, \ldots, i_t),$$

то (при введенных ранее обозначениях для индексов) индексы $\varphi_1, \varphi_2, \ldots, \varphi_k$ оператора Q, все или некоторые, можно заменить индексами

$$\psi_1 = \varphi[K(I)], \quad \psi_2 = \varphi[K(I)], \quad \ldots, \quad \psi_k = \varphi_k[K(I)].$$

Здесь K — комплекс, нормальная последовательность которого состоит из формулы

$$i_j = \theta(i_1, i_2, \ldots, i_t).$$

В процессе программирования программист нередко приходит к программам, которые он признает неудовлетворительными по тем или другим причинам (например, программа содержит больше команд, чем в данном случае допустимо, и т. п.). В таких случаях программы приходится переделывать. Переделку программы можно осуществить на уровне логической схемы путем равносильных преобразований. Заметим, что часто недостатки будущей программы можно установить заранее, путем анализа ее логической схемы. Таким образом, равносильные преобразования могут быть с выгодой использованы при программировании.

Как показала практика последних лет, равносильные преобразования программ или других алгорифмов, заданных с помощью того или другого алгорифмического языка, являются необходимыми частями процесса автоматического программирования. Таким образом, автоматизация программирования является еще одной областью применения равносильных преобразований.

Литература к главе 5: [37], [38], [62], [74].

ГЛАВА 6

НЕКОТОРЫЕ ПРИЕМЫ ПРОГРАММИРОВАНИЯ

§ 1. Операторы, зависящие от параметров, их переадресация и восстановление

1. Операторы, зависящие от параметров, и операторы переадресации. Некоторые адреса команд в операторах программы зависят от параметров, которые обычно считаются целочисленными переменными. В случае, когда такие операторы выполняются при определенной последовательности значений параметров, в этих операторах программы адресные части некоторых команд подвергаются изменению. Такое изменение команд (и операторов) называется *переадресацией.*

В зависимости от конструкции программно-управляемой машины может применяться *переадресация в ячейках*, осуществляемая путем изменения кода команды в ячейке оперативной памяти, и *переадресация в устройстве управления*, при которой в ячейке оперативной памяти код команды остается неизменным, а прибавление чисел к адресам команды происходит перед ее выполнением в устройстве управления (например, в регистре команд). Переадресация в устройстве управления бывает двух видов: на содержимое ячейки оперативной памяти (коротко — *переадресация в УУ*) и на содержимое индексного регистра (коротко — *переадресация по индексу*).

В случае переадресации в ячейках оператор, зависящий от параметров, составляют в виде его значения, отвечающего начальным значениям параметров. Переадресация его команд осуществляется с помощью специальных операторов, называемых *операторами переадресации.*

Пусть адрес команды, изменяемый оператором переадресации $F(i)$, до первого выполнения $F(i)$ имеет вид a_0.

Обозначим через $h(k)$ число, прибавляемое к этому адресу при k-м выполнении $F(i)$.

Если адрес команды не подвергался изменениям другими операторами, то после i-го выполнения $F(i)$ он принимает вид

$$a(i) = a_0 + \sum_{k=1}^{i} h(k). \tag{6.1}$$

В частном, наиболее распространенном случае $h(k) = h = \text{const}$, так что

$$a(i) = a_0 + ih. \tag{6.2}$$

При этом говорят, что изменяемый адрес подвергается переадресации *с постоянным шагом.*

При зависимости адреса от многих параметров $i_1, i_2, \ldots, i_n$, при условии, что адрес подвергался изменению только операторами $F(i_s)$ $(s = 1, 2, \ldots, n)$, после того как каждый оператор $F(i_s)$ $(s = 1, 2, \ldots, n)$ выполнялся i_s раз, изменяемый адрес получает вид

$$a(i_1, i_2, \ldots, i_n) = a_0 + \sum_{k=1}^{i_1} h_1(k) + \sum_{k=1}^{i_2} h_2(k) + \ldots + \sum_{k=1}^{i_n} h_n(k), \tag{6.3}$$

где $h_s(k)$ — число, прибавляемое к рассматриваемому адресу оператором $F(i_s)$ при k-м выполнении этого оператора. В частном случае, если $h_s(k) = h_s = \text{const}$, получается

$$a(i_1, i_2, \ldots, i_n) = a_0 + i_1h_1 + i_2h_2 + \ldots + i_nh_n. \tag{6.4}$$

В случае переадресации в УУ оператор программы, зависящий от параметров, отличается от описанного выше тем, что перед каждой его командой, подлежащей видоизменению, ставится команда «изменение команд», в адресе которой указан номер ячейки, содержащей вспомогательное число, прибавляемое к изменяемой команде перед ее выполнением. Изменение этого вспомогательного числа (его «переадресацию») осуществляет оператор, называемый обычно *оператором изменения параметров* (обозначается F, как и оператор переадресации). Формулы (6.1) — (6.4) можно рассматривать как формулы, описывающие зависимость от параметров вспомогательных чисел $a(i)$ и $a(i_1, i_2, \ldots, i_n)$, применяемых при переадресации в регистре.

При использовании переадресации по индексу, в зависимости от конструкции машины, оператор, зависящий от параметра, составляют в виде либо его начального, либо его конечного значения, смотря по тому, прибавляется к коду команды или от него отнимается содержимое индексного регистра. Коды изменяемых команд содержат признак переадресации. До выполнения этого оператора должна быть произведена засылка вспомогательного числа в индексный регистр. После его выполнения, содержимое индексного регистра должно быть видоизменено.

Пример 6.1. Арифметический оператор A_y^i, осуществляющий счет по формуле

$$y_i = x_i + x_i^2 + x_i^3 = [(x_i + 1)\, x_i + 1]\, x_i$$

при условии, что числа $x_1, x_2, \ldots, x_m$ размещены соответственно в ячейках $b+1, b+2, \ldots, b+m$, для машины *Стрела*, использующей переадресацию в ячейках, может иметь вид

$g+1)$	$b+1$	$e+1$	$c+1$	0	01
$g+2)$	$b+1$	$c+1$	$c+1$	0	05
$g+3)$	$c+1$	$e+1$	$c+1$	0	01
$g+4)$	$c+1$	$b+1$	$d+1$	0	05

где $c+1$ — рабочая ячейка. В ячейке $e+1$ хранится число $+1$, а $d+1, d+2, \ldots, d+m$ — ячейки для записи результатов $y_1, y_2, \ldots, y_m$.

Для переадресации оператора A_y^i необходимы вспомогательные коды

$e+2)$	0001	0000	0000	0	00
$e+3)$	0000	0001	0001	0	00

Оператор переадресации имеет вид

$h+1)$	$g+1$	$e+2$	$g+1$	0	02
$h+2)$	$g+2$	$e+2$	$g+2$	0	02
$h+3)$	$g+4$	$e+3$	$g+4$	0	02

Пример 6.2. Для машины *Урал-2*, при использовании переадресации в УУ, если числа $x_1, x_2, \ldots, x_m$ размещены в полных ячейках $b+2, b+4, \ldots, b+2m$ (b четное), а вспомогательный код для переадресации находится в ячейке $e+1$ (неполной), оператор A_y^i имеет вид (знак $+$ перед кодом операции команды обычно подразумевают, но не пишут)

$g+1)$	30	$e+1$	0	$g+6)$	01	$c+2$	4
$g+2)$	02	$b+2$	4	$g+7)$	30	$e+1$	0
$g+3)$	01	$c+2$	4	$g+10)$	06	$b+2$	4
$g+4)$	30	$e+1$	0	$g+11)$	30	$e+1$	0
$g+5)$	06	$b+2$	4	$g+12)$	16	$d+2$	4

Здесь $c+2$ (c — четное) — ячейка (полная), хранящая число $+1$, а для результатов отведены ячейки $d+2$, $d+4$, ... (полные, d — четное). Оператор $F(i)$ изменения параметра будет

$$\begin{array}{llll} h+1) & 02 & e+1 & 0 \\ h+2) & 01 & e+2 & 0 \\ h+3) & 16 & e+1 & 0 \end{array}$$

причем в ячейке $e+2$ должен храниться код $+00\ 0002\ 0$. В операторе A_v^i команды «$g+1$», «$g+4$», «$g+7$» и «$g+11$» предназначены для выполнения переадресации в регистре команд.

На машине *Урал*-2 в ряде случаев более выгодна переадресация по индексу, при которой оператор A_v^i вместо вышеприведенного имеет вид

$$\begin{array}{llllllll} g+1) & 25 & 2n-2 & 4 & g+5) & 01 & c+2 & 4 \\ g+2) & -02 & b+2n & 4 & g+6) & -06 & b+2n & 4 \\ g+3) & 01 & c+2 & 4 & g+7) & -16 & d+2n & 4 \\ g+4) & -06 & b+2n & 4 & g+10) & 24 & g+2 & 0 \end{array}$$

Здесь команда «$g+1$» задает начальное значение содержимого индексного регистра — число $2(n-1)$, а команда «$g+10$» производит изменение содержимого индексного регистра, уменьшая его при каждом своем выполнении на 2.

Наличие знака «—» перед кодом операции команды является признаком переадресации. Содержимое индексного регистра при выполнении команд, снабженных знаком «—», отнимается от их адресов. Признак полной ячейки (последняя цифра 4 в коде команды) в команде «$g+1$» означает, что при каждом выполнении команды «$g+10$» содержимое индексного регистра должно уменьшаться на две единицы.

2. Операторы восстановления. В некоторых случаях необходимо предусмотреть в программе, чтобы после того, как оператор, зависящий от параметров, был выполнен при различных значениях параметра i, он был выполнен при начальном значении этого параметра.

Если переадресация выполнялась в ячейках, то возврат к начальному значению параметра требует изменения адресов ряда команд оператора, зависящего от параметров. Это изменение называется *восстановлением* оператора (или изменяемых команд) по параметру i. Восстановление по параметру i может быть осуществлено одним из двух способов: 1) с помощью оператора переадресации по другому параметру j,

уничтожающего результаты всех предыдущих переадресаций по параметру i; 2) с помощью оператора восстановления $O(i)$.

Работа оператора восстановления заключается в том, что в восстанавливаемом операторе на места команд, зависящих от параметров, производится запись кодов, одинаковых по виду с этими командами при начальном значении параметра i. Если восстанавливаемый оператор зависит только от одного параметра i или если его восстановление по i производится после того, как он уже восстановлен по всем другим параметрам, то применяется так называемое *восстановление в начальный вид*. Коды, используемые при восстановлении в начальный вид, могут быть заготовлены заранее (при составлении программы) и называются *константами восстановления*. В тех случаях, когда заранее составить константы восстановления нельзя, получение кодов для восстановления команд предусматривают в самой программе.

Если используется переадресация в УУ, то оператор, зависящий от параметров в ячейках памяти, неизменен. В этом случае возврат к начальному значению параметра производится путем изменения вспомогательных чисел, применяемых для переадресации в УУ. Оператор, производящий указанное изменение, также называется оператором восстановления и обозначается $O(i)$. Все сказанное выше о восстановлении команд справедливо и для восстановления вспомогательных чисел, применяемых при переадресации в УУ.

Если используется переадресация по индексу, то возврат к начальному значению параметра производится путем изменения содержимого индексных регистров.

§ 2. Циклы

Реальная программа, как правило, обеспечивает выполнение значительно большего количества команд, чем их число в самой программе. Это возможно благодаря тому, что некоторые серии команд программы выполняются многократно, подвергаясь в случае необходимости преобразованиям. Такое свойство программ обеспечивают, включая в их состав группы операторов, называемые *циклами*.

Наиболее простыми являются неразветвленные циклы с одним входом и одним выходом. Такие циклы называются элементарными.

1. Элементарный итерационный цикл. Элементарный итерационный цикл содержит оператор переноса, арифметический оператор и логический оператор. Его выражение в логической схеме имеет вид

$$\pi \underset{m}{\lrcorner} \Pi_k A_{k+1} P_{k+2} \lambda_m \overline{\lambda}_j,$$

где π — некоторая совокупность правых знаков перехода, а $\overline{\lambda}_j$ обозначает левый верхний (нижний) знак перехода, если λ_m является левым нижним (верхним) знаком перехода. При замене знаков перехода стрелками имеем

$$\longrightarrow \Pi_k A_{k+1} P_{k+2} \longrightarrow \quad \text{или} \quad \longrightarrow \Pi_k A_{k+1} P_{k+2} \longrightarrow.$$

Пример 6.3. Решение методом итераций уравнения

$$x = f(x)$$

при начальном значении неизвестного x_0 и заданной оценке погрешности $\varepsilon > 0$ осуществляют путем вычисления величин x_i для $i = 1, 2, \ldots$ по формуле

$$x_i = f(x_{i-1}).$$

Процесс считают оконченным, как только будет получено x_i, удовлетворяющее условию

$$| x_i - x_{i-1} | < \varepsilon.$$

Найденное x_i считают искомым корнем уравнения. Соответствующая часть логической схемы программы решения описанной задачи представляет собой итерационный цикл и имеет вид (для $k = 3$, $m = 1$)

$$\underset{1}{\lrcorner} \Pi_3 A_4 P_5 \underset{1}{\llcorner}.$$

Предположим, что число x_0 записано в ячейке α. Оператор Π_3 переносит «α» в ячейку β. Оператор A_4 вычисляет величину f[«β»] и записывает ее в ячейку α. Оператор P_5 проверяет справедливость неравенства | «α» — «β» | $< \varepsilon$ и управляет выполнениями цикла.

2. Элементарный цикл с переадресацией. Элементарный цикл с переадресацией может изображаться в логической схеме выражением

$$\pi \underset{m}{\lrcorner} A_k^i F_{k+1}(l) P_{k+2} \lambda_m \overline{\lambda}_j$$

и содержит оператор переадресации. Взаимное расположение операторов в цикле может быть различным.

Пример 6.4. Основная часть логической схемы программы вычисления таблицы значений функции $y = f(x)$ для заданных зна-

чений аргумента $x = x_1, x_2, \ldots, x_n$ является циклом с переадресацией и может иметь вид

$$\underset{1}{\lrcorner} A_3^i F_4(i) P_5 \underset{1}{\llcorner}.$$

В ней A_3^i — оператор счета по формуле $y_i = f(x_i)$; $F_4(i)$ — оператор переадресации, изменяющий (в программе) команды оператора A_3^i таким образом, чтобы при следующем выполнении цикла вычисления велись по формуле $y_{i+1} = f(x_{i+1})$, а не по формуле $y_i = f(x_i)$; P_5 — проверка условия $i = n + 1$ (предполагается, что до начала выполнения программы и соответственно ее логической схемы $i = 1$).

3. Элементарный цикл с переадресацией и восстановлением. При восстановлении по i с помощью оператора переадресации по другому параметру j выражение цикла с переадресацией и восстановлением в логической схеме программы может иметь, например, вид

$$\pi \underset{m}{\lrcorner} A_k^{i,j} F_{k+1}(i) P_{k+2} \lambda_m F(j) \lambda_n.$$

Если применялась переадресация в ячейках, то для восстановления применяется также переадресация в ячейках. В случае переадресации в УУ восстановление производится путем видоизменения кода, с помощью которого предусмотрено изменение команд.

Заметим, что если оператор $F_{k+1}(i)$ производит переадресацию с постоянным шагом и если число выполнений цикла наперед задано, то вспомогательный код для переадресации по параметру j может быть заранее составлен и записан в одной из ячеек памяти. При переадресации по параметру i с переменным шагом или при неизвестном заранее числе выполнений цикла в состав оператора $F(i)$ необходимо включить команды, суммирующие числа $h(k)$ (см. формулу (6.1)) и тем самым подготавливающие вспомогательный код, вычитанием которого от измененных адресов команд оператором $F(j)$ будет произведено восстановление.

При восстановлении по параметру i с помощью оператора $O(i)$, выражение цикла в логической схеме может иметь вид

$$\pi \underset{m}{\lrcorner} A_k^i F_{k+1}(i) P_{k+2} \lambda_m O_{k+2}(i) \lambda_n \tag{6.5}$$

или

$$\pi O_{k-1}(i) \underset{m}{\lrcorner} A_k^i F_{k+1}(i) P_{k+2} \lambda_m \overline{\lambda_n}. \tag{6.6}$$

Выражения (6.5) и (6.6) между собой равносильны и сводятся друг к другу с помощью равносильных преобразований (см. гл. 5). Если операторы цикла зависят только от одного параметра или, по крайней мере, команды, зависящие от параметра i, не зависят от других параметров, то удобно применять восстановление в начальный вид.

Если все или некоторые из переадресуемых по i команд зависят сразу от нескольких параметров, может быть применена следующая схема цикла:

$$\pi O'_{k-1} \underset{m}{\lrcorner} A_k^{i,j} F_{k+1}(i) P_{k+2} \lambda_m O_{k+3}(i) \lambda_n.$$

Здесь оператор O'_{k-1} (в случае переадресации в ячейках) «фотографирует» команды, зависящие от i, и запасает их в специальных ячейках, а оператор $O_{k+3}(i)$ ставит запасенные «фотографии» на место изменившихся команд. При переадресации в регистре команд оператор O'_{k-1} «фотографирует» вспомогательный код, применяемый для изменения команд, а оператор $O_{k+3}(i)$ заносит эту «фотографию» на место изменившегося кода. Описанный способ «фотографирования» является одним из способов программного получения кодов, необходимых для восстановления по параметру

Пример 6.5. Логическая схема программы для получения суммы попарных произведений $x_i x_j$ (включая также $i=j$) величин $x_1, x_2, \ldots, x_n$ может быть следующей:

$$И_0 \underset{2}{\lrcorner} O'_1 \underset{1}{\lrcorner} A_2^{i,j} F_3(j) P_4 \underset{1}{\llcorner} O_5(j) F_6(i) P_7 \underset{2}{\llcorner} Я_8.$$

Эта схема содержит цикл с переадресацией и восстановлением

$$O'_1 \underset{1}{\lrcorner} A_2^{i,j} F_3(j) P_4 \underset{1}{\llcorner} O_5(j),$$

входящий в состав другого цикла. В этой схеме $A_2^{i,j}$ — оператор, вычисляющий $x_i x_j$ и прибавляющий эту величину к содержимому некоторой ячейки α, начальное содержимое которой равно нулю; $F_3(j)$ — оператор переадресации по j; P_4 — проверка условия $j=n+1$ (предполагается, что начальное значение j равно 1); O'_1 — оператор, «фотографирующий» команды из $A_2^{i,j}$, зависящие от j; $O_5(j)$ — оператор восстановления по параметру j; $F_6(i)$ — оператор переадресации по i; P_7 — проверка условия $i=n+1$ (предполагается, что начальное значение параметра i равно 1); $Я_8$ — останов машины.

В машинах, оборудованных индексными регистрами, например таких, как *Урал-2*, *Урал-4*, цикл с переадресацией

и восстановлением может иметь в логической схеме следующий вид:

$$\pi H_{k-2}\, Q_{k-1} \underset{m}{\lrcorner} A_k^i F_{k+1}(i) P_{k+2} \underset{m}{\llcorner}. \qquad (6.7)$$

Здесь H_{k-2} — оператор (состоящий из одной команды), который заносит некоторое число в индексный регистр; Q_{k-1} — обобщенный оператор (в частном случае может отсутствовать), не зависящий от параметра i; оператор $F_{k+1}(i)$ изменяет число, занесенное в индексный регистр, а оператор P_{k+2} проверяет, достигло ли содержимое индексного регистра некоторого определенного значения. Пока упомянутое значение не достигнуто, P_{k+2} передает управление оператору A_k^i; как только это значение будет достигнуто, P_{k+2} передает управление оператору, стоящему в логической схеме непосредственно после него. Операторы $F_{k+1}(i) P_{k+2} \lambda_m$ представлены в программе одной-единственной командой. Эту команду (эту пару операторов) обычно обозначают символом «закрытая скобка» $\Big\}_s^i{}_m$; при этом вместо правого знака перехода $\underset{m}{\lrcorner}$ ставят символ ${}_m\Big\{_s$, а оператор H_{k-2} обозначают символом R_{k-2}^s, где s — номер индексного регистра (если в машине присутствует только один индексный регистр, как это имеет место для машины *Урал-1*, то индекс s опускают). При этом вместо записи (6.7) получается более компактная запись

$$\pi R_{k-2}^s Q_{k-1}\, {}_m\Big\{A_k^i\Big\}_{s\,m}^{i}.$$

Цикл ${}_m\Big\{A_k^i\Big\}_{s\,m}^{i}$ называется циклом по индексу.

4. Сложные циклы. Циклы, встречающиеся в программах, могут быть значительно сложнее описанных выше. Например, могут встречаться циклы с внутренними разветвлениями, имеющие логические схемы вида

$$\pi \underset{m}{\lrcorner} A_k^{i,\,j} P_{k+1} \lambda_n F_{k+2}(i) \underset{t}{\llcorner} \underset{n}{\lrcorner} F_{k+3}(j) \underset{t}{\lrcorner} P_{k+4}\, \lambda_m \overline{\lambda}_s.$$

Весьма часто встречаются циклы, содержащие внутри себя другие циклы. Например, возможен цикл, описываемый в логической схеме выражением

$$\underset{1}{\lrcorner} A_k^{i,\,s} F_{k+1}(s) P_{k+2} \lambda_1 A_{k+3} \underset{2}{\lrcorner} A_{k+4}^{i,\,j} F_{k+5}(j) P_{k+6} \lambda_2 Q_{k+7}(j) F_{k+8}(i) P \lambda_1.$$

В такой схеме «большой» цикл содержит внутри себя два меньших.

К сложным циклам относятся также циклы, имеющие несколько (больше чем один) входов или несколько выходов. Например, в логических схемах такие циклы могут иметь вид выражений

$$\pi_1 \bigsqcup_m A_k^i \pi_2 A_{k+1}^i F_{k+2}(i) P_{k+3} \lambda_m \overline{\lambda}_t$$

или

$$\pi_1 \bigsqcup_m A_k^i P_{k+1} \lambda_{m+1} A_{k+2}^i F_{k+3}(i) P_{k+4} \lambda_m \overline{\lambda}_t.$$

Особо отметим случай так называемых пересекающихся циклов, когда существует группа операторов, входящих одновременно в два или более циклов. Логическая схема двух пересекающихся циклов может иметь, например, вид

$$\pi \bigsqcup_m A_k^i F_{k+1}(i) \bigsqcup_n A_{k+2}^{i,j} P_{k+3} \lambda_m A_{k+4}^j F_{k+5}(j) P_{k+6} \lambda_n \overline{\lambda}_t.$$

Приведенные здесь примеры не исчерпывают возможных видов структуры циклов.

5. Счетчик выполнений цикла. В состав каждого цикла программы должен входить логический оператор, управляющий «работой» цикла, т. е. выполнениями входящих в этот цикл команд. Такой оператор в зависимости от значения проверяемого им предиката либо передает управление начальной команде цикла, либо передает управление вовне. Кроме логического оператора, управляющего работой цикла, в ряде случаев в состав цикла включают некоторые команды, изменяющие какую-либо величину, являющуюся аргументом предиката, проверяемого логическим оператором, и осуществляющие как бы счет выполнений цикла.

Один из наиболее часто применяемых счетчиков выполнений цикла представляет собой совокупность одной или нескольких команд и двух ячеек памяти. В одной из ячеек записывают какое-либо число τ, а в другой — число ϱ, удовлетворяющее неравенству

$$(n-1)\tau < \varrho < n\tau,$$

где n — заданное число выполнений цикла. Предположим, что упомянутые числа размещены следующим образом:

$$\beta+1)\ \varrho:\quad \beta+2)\ \tau.$$

Команды счетчиков обеспечивают выполнение операции

$$«\beta+1»:=«\beta+1»-«\beta+2»,$$

т. е. вычитания из «$\beta+1$» числа «$\beta+2$» и записи результата в ячейку $\beta+1$. Логический оператор, управляющий работой цикла, проверяет знак числа «$\beta+1$» и передает управление при знаке плюс начальной команде цикла, а при знаке минус вовне.

Пример 6.6. В программах, составленных для машины *БЭСМ-2*, обобщенный оператор, управляющий с помощью счетчика работой цикла, может состоять из команд

$$\alpha+\mu+1)\quad 02\quad \beta+1\quad \beta+2\quad \beta+1,$$
$$\alpha+\mu+2)\quad 36\quad 0000\quad \beta+1\quad \alpha+1.$$

Последняя команда является логическим оператором, проверяющим условие $0<(\beta+1)$ и передающим при выполнении этого условия управление в ячейку $\alpha+1$ (начальную в цикле), а при его невыполнении — в ячейку $\alpha+\mu+3$ (за пределы цикла).

Если в случае переадресации в ячейках при выполнениях цикла изменяется только один параметр, например l, то команда, зависящая от этого параметра, может быть использована в качестве элемента счетчика выполнений цикла. Наиболее прост случай, когда упомянутая команда зависит только от одного параметра i. Предположим, что эта команда находится в ячейке $\mu+\varkappa$ и при выполнениях цикла, если ее трактовать как число, монотонно возрастает или монотонно убывает. Для определенности остановимся на первом. Пусть «$\mu+\varkappa$»′ и «$\mu+\varkappa$»″ — содержимое ячейки $\mu+\varkappa$ соответственно при предпоследнем и при последнем выполнениях цикла. По предположению, трактуя модификации команды «$\mu+\varkappa$» как числа, имеем

$$«\mu+\varkappa»'<«\mu+\varkappa»''.$$

Выбираем число ϱ, удовлетворяющее условию

$$«\mu+\varkappa»'<\varrho<«\mu+\varkappa»'',$$

и размещаем его в ячейке $\beta+1$.

Для управления работой цикла можно использовать логический оператор, проверяющий выполнение предиката

$$«\mu + x» < «\beta + 1»,$$

или, что то же,

$$«\mu + x» - «\beta + 1» < 0.$$

Если проверяемое условие выполнено, управление должно передаваться первой команде цикла, если не выполнено, то — некоторой команде, не принадлежащей циклу.

Пример 6.7. Рассмотрим программу для машины *Минск-2*, содержащую цикл, изображаемый в логической схеме выражением

$$\pi \bigsqcup_{m} A^{i}_{k+1} F_{k+2}(i) P_{k+3} \lambda_{m}.$$

Пусть в ячейке $\alpha + \mu$ содержится команда

$$\alpha + \mu) \quad 00 \quad 0325 \quad 0112,$$

второй адрес которой является постоянным, а первый адрес при каждом выполнении цикла увеличивается на $h = 2$. Предположим, что цикл должен выполняться двадцать пять раз (или в восьмеричной системе счисления $n = 31$). Очевидно, при этом

$$«\alpha + \mu»'_1 = 0325 + h(n-2) = 0325 + 2 \cdot 27 = 0325 + 56 = 0403,$$

$$«\alpha + \mu»''_1 = 0325 + h(n-1) = 0325 + 2 \cdot 30 = 0325 + 60 = 0405,$$

где $\alpha + \mu_1$ — первый адрес команды $«\alpha + \mu»$.

Здесь $«\alpha + \mu»''_1$ содержит слагаемое $h(n-1)$ потому, что первый раз команда $«\alpha + \mu»$ выполняется в ее неизменном виде, второй раз — после увеличения на 00 h 0000, третий раз — после увеличения на 00 $2h$ 0000 и т. д.

В качестве Q можно взять, например, 00 0404 0112.

Оператор, управляющий работой цикла, может иметь вид

$$\alpha + \mu + 1) \quad 11 \quad \beta + 1 \quad \alpha + \mu$$
$$\alpha + \mu + 2) \quad 34 \quad g + 1 \quad \alpha + 1$$

При этом предполагается, что в ячейке $g + 1$ расположена команда, не принадлежащая циклу.

Если применяется переадресация команд в УУ, то в качестве счетчика выполнений цикла можно использовать изменяемый вспомогательный код, применяемый для переадресации команд.

§ 3. Приемы, связанные с применением стандартных ячеек

В ряде случаев группы исходных данных для работы оператора Q размещены (или вычисляются в программе и размещаются) в некоторой последовательности групп ячеек.

Для того чтобы оператор Q «брал» эти исходные данные, ряд команд оператора необходимо видоизменять перед каждым его выполнением. Иногда бывает выгодным вместо видоизменения команд оператора переносить исходные для него данные в фиксированные ячейки. Эти фиксированные ячейки принято называть *стандартными.*

1. Вынесение величин в стандартные ячейки. Включение в программу оператора переноса, переписывающего числа из последовательности ячеек в стандартные ячейки (такой оператор называется *засылкой* и имеет стандартное обозначение $З_v$), и соответствующее изменение оператора, использующего эти числа, называется *вынесением величин в стандартные ячейки.*

При вынесении величин в стандартные ячейки оператор Q, использующий упомянутые величины, заменяется парой операторов $ЗQ$.

В ряде случаев вынесение величин в стандартные ячейки позволяет значительно уменьшить количество команд переадресации и восстановления. Это дает экономию как рабочих тактов (машинного времени), так и ячеек.

Пример 6.8. Предположим, что в программе для машины *Стрела* присутствует цикл, имеющий в логической схеме выражение

$$\pi \underset{m}{\sqcup} A_k^i F_{k+1}(i) P_{k+2} \lambda_m \overline{\lambda}_n, \tag{6.8}$$

причем оператор A_k^i производит вычисление значений $y_i = f(x_i)$, где x_i пробегает $x_1, x_2, \ldots, x_n$, а

$$f(x) = x\{\ln[\sqrt{x \sin(x+1)} + x] + x\}.$$

В результате вынесения величины x в стандартную ячейку выражение (6.8) цикла заменяется следующим:

$$\pi \underset{m}{\sqcup} З_k^i, A_k^i F_{k+1}(i) P_{k+2} \lambda_m \overline{\lambda}_n.$$

Если числа x_i размещены в ячейках $b+i$, а число 1 — в ячейке b, и если заданы вспомогательные коды

$d+1)$ 0001 0000 0000 0 00

$d+2)$ 0000 0000 0001 0 00

причем y_i должны быть записаны на места x_i в ячейки $b+i$, то операторы $\mathcal{Z}_k^i$ и A_k^i (с вынесением величины x в стандартные ячейки) имеют вид (адреса команд, зависящие от i, помечены звездочкой)

$a+1)$	$b+1$*	0000	$c+3$	0	13
$a+2)$	$c+3$	b	$c+1$	0	01
$a+3)$	$c+1$	0000	$c+1$	0	67
$a+4)$	$c+3$	$c+1$	$c+1$	0	05
$a+5)$	$c+1$	0000	$c+1$	0	63
$a+6)$	$c+3$	$c+1$	$c+1$	0	01
$a+7)$	$c+1$	0000	$c+1$	0	66
$a+10)$	$c+3$	$c+1$	$c+1$	0	01
$a+11)$	$c+3$	$c+1$	$b+1$*	0	05

Оператор переадресации $F_{k+1}(i)$ состоит из двух команд

$a+16$ $a+1$ $d+1$ $a+1$ 0 02,

$a+17$ $a+15$ $d+2$ $a+11$ 0 02.

Если бы программа была составлена без вынесения величин в стандартные ячейки, то команда «$a+1$» в ней отсутствовала бы, а адрес $c+3$ в командах оператора A_k^i был бы заменен адресом $b+1$*. При этом оператор переадресации состоял бы из 5 команд.

Вынесение величин в стандартные ячейки выгодно и для машин, в которых применяется переадресация в УУ. Разумеется этим приемом пользуются в том случае, когда нельзя воспользоваться переадресацией по индексу.

2. Циркуляция величин в стандартных ячейках. В некоторых случаях экономия команд переадресации и рабочих тактов машины достигается с помощью приема, называемого *циркуляцией (величин) в стандартных ячейках.*

Сущность циркуляции состоит в том, что с каждым выполнением команд цикла в некоторой группе ячеек $c+1$, $c+2$, ... , $c+tk$ производятся переносы чисел из ячеек $c+r$ в ячейки $c+r+k$ ($r=1, 2, \ldots, k(t-1)$), а в освободившиеся ячейки $c+1$, $c+2$, ... , $c+k$ записываются новые числа.

Иногда циркуляцию производят «в обратном направлении», производя переносы из ячеек с большими номерами в ячейки с меньшими номерами (это в значительной мере связано с конструктивными особенностями машины и со спецификой ее системы команд).

Оператор циркуляции обозначают символом $Ц_{\vee}$.

Пример 6.9. Функция $y=f(x)$ задана своими значениями $y_0, y_1, y_2, \ldots, y_n, y_{n+1}$ в точках $x_0, x_1, x_2, \ldots, x_n, x_{n+1}$ (причем $x_{i+1}-x_i=h$ для $i=0, 1, 2, \ldots, n$). Требуется вычислить вторую производную $y''=f''(x)$ в точках $x_1, x_2, \ldots, x_n$ с помощью формулы

$$y_i''=\frac{y_{i-1}-2y_i+y_{i+1}}{h^2} \qquad (i=1, 2, \ldots, n).$$

Решение этой задачи описывает следующее операторное выражение (которое может входить в различные логические схемы):

$$П_1 \sqcup_1 Ц_2\, П_3^i\, A_4^i\, F_5(i)\, P_6 \sqcap_1^2.$$

Здесь оператор $П_1$ переносит числа y_0 и y_1 соответственно в стандартные ячейки $c+2$ и $c+3$ из ячеек, скажем, $b+0$ и $b+1$, в которых они заданы. Оператор $Ц_2$ производит циркуляцию величин в ячейках $c+1$, $c+2$, $c+3$, перенося их из ячеек $c+r$ ($r=2, 3$) в ячейки $c+r-1$. Циркуляцию можно описать нормальной последовательностью (см. гл. 5, § 1, п. 4)

$$«c+1»:=«c+2»,$$
$$«c+2»:=«c+3».$$

Оператор $П_3^i$ производит перенос числа из ячейки $b+i+1$ (начальное значение i равно 1, а конечное — n), хранящей y_{i+1}, в ячейку $c+3$. Оператор A_4^i имеет нормальную последовательность, состоящую, например, из формулы

$$«d+i»=\frac{«c+1»-2«c+2»+«c+3»}{h^2}.$$

Оператор $F_5(i)$ производит переадресацию по параметру i с постоянным шагом $\Delta i=1$. Оператор P_6 управляет работой цикла (например, с помощью счетчика). После n-кратного выполнения цикла, входящего в приведенное выше операторное выражение, величины $y_1'', y_2'', \ldots, y_n''$ будут получены соответственно в ячейках $d+1$, $d+2, \ldots, d+n$.

§ 4. Логические шкалы. Операторы формирования

1. Логические шкалы. Предположим, что некоторый оператор Q программы в ходе выполнения последней получает управление заданное число (например, n) раз, причем известно, что после i-го выполнения оператора Q должен происходить переход к оператору с номером N_i. Другими словами, задана целочисленная функция

$$N_i = f(i).$$

Выпишем все значения этой функции $N_1, N_2, \ldots, N_n$ и составим из них последовательность попарно-различных чисел $M_1, M_2, \ldots, M_{n'}$ (такую, чтобы для каждого N_i существовало равное ему M_j и для каждого M_j существовало хотя бы одно равное ему N_i).

Определим целое число k, удовлетворяющее неравенствам

$$2^{k-1} \leqslant n' \leqslant 2^k - 1,$$

то есть

$$\log_2 (n' + 1) \leqslant k \leqslant 1 + \log_2 n'.$$

Составим всевозможные k-разрядные целые двоичные числа и по произволу выберем из них n' чисел. Каждому числу M_j поставим во взаимно однозначное соответствие k-разрядное двоичное число m_j из выбранных. Теперь каждому N_i поставим в соответствие то двоичное число m_{j_i}, которое соответствует M_j, равному N_i.

Последовательность двоичных цифр, которая получается, если в одну строку (подряд) записать двоичные числа $m_{j_1}, m_{j_2}, \ldots, m_{j_n}$, называется логической шкалой.

Выделяя первые k разрядов логической шкалы после первого выполнения оператора Q и сравнивая их последовательно с числами $m_1, m_2, \ldots, m_{n'}$, мы можем узнать номер оператора, к которому необходимо произвести переход. Затем нужно первые k разрядов шкалы зачеркнуть, а справа приписать к ней какое-либо k-разрядное двоичное число, отличное от m_{j_n}. Последнее делается для того, чтобы число разрядов в шкале оставалось постоянным и вместе с тем была бы возможность установить момент, когда шкала исчерпается. Теперь шкала готова к использованию при новом выполнении оператора Q.

Обычно взаимно однозначное соответствие между двоичными числами m_j и числами $M_j (j = 1, 2, \ldots, n')$ устанавливают таким образом, чтобы номеру оператора N_n отвечало двоичное число m_{jn}, не равное нулю. Тогда при вычеркивании уже использованных разрядов шкалы справа можно приписывать к ней k-разрядный нуль. Признаком того, что шкала исчерпана, будет служить равенство всех разрядов шкалы нулю. Мы видим, что преобразование шкалы после выделения ее первых k разрядов в этом случае эквивалентно операции сдвига шкалы на k разрядов влево.

Во многих случаях бывает удобно сперва производить сдвиг шкалы, а затем выделять k разрядов, оказавшихся на первом (самом левом) месте. Чтобы иметь такую возможность, левее m_{j_1} в шкалу вписывают произвольное k-разрядное двоичное число m_{j_0}.

Техническое осуществление логической шкалы зависит от значений чисел k и n. Шкалу обычно размещают в последовательных ячейках памяти, а для выделения ее первых k разрядов (с помощью операции поразрядного логического умножения) составляют набор двоичных цифр, в котором на первом месте стоят k единиц, а на остальных местах нули. Этот набор называется указателем шкалы. В простейшем случае шкала и ее указатель занимают по одной ячейке.

Сравнение выделенного из шкалы k-разрядного набора двоичных цифр с числами $m_1, m_2, \ldots, m_n$, и необходимую передачу управления осуществляет группа логических операторов, размещаемая непосредственно после оператора Q.

Пример 6.10. Дана логическая схема

$$И_0\, П_1\, A_2\, A_8^j F_4 (j) \ldots A_7\, П_8\, Я_0, \qquad (6.9)$$

в которой многоточием обозначены пропущенные логические операторы, необходимые для использования логической шкалы. Известно, что после i-го выполнения оператора $F_4 (j)$ управление должно передаваться в соответствии с табл. 6.1.

Таблица 6.1. Значения целочисленной функции N_i

i	N_i	i	N_i
1÷6	3	8; 9	3
7	4	10	7

Требуется построить логическую шкалу, обеспечивающую описанные передачи управления, и включить в схему необходимые логические операторы.

Из табл. 6.1 видно, что $M_1=3$, $M_2=4$, $M_3=7$, следовательно, $n'=3$. Определяем $k=2$ (ибо $2^1\leqslant 3\leqslant 2^2-1$). Двухразрядных двоичных чисел всего существует четыре. Из них отберем три (ибо $n'=3$):

$$00,\quad 01,\quad 10.$$

Положим $m_1=00$, $m_2=01$, $m_3=10$, $m_0=11$ (m_0 произвольное). Логическая шкала будет иметь вид

$$11\quad 00\quad 00\quad 00\quad 00\quad 00\quad 00\quad 01\quad 00\quad 00\quad 10.$$

Если шкала будет использована в программе для машины *Стрела*, имеющей 43-разрядные ячейки, то справа к шкале следует приписать 21-разрядный нуль. При этом получится

11 00 00 00 00 00 00 01 00 00 10 00 00 00 00 00 00 00 00 00 0.

Указатель шкалы будет иметь при этом вид

11000000000000000000000000000000000000000000.

Если шкалу разметить в ячейке $c+1$, ее указатель — в ячейке $c+2$, а в ячейке $c+3$ поместить набор

01 00000000000000000000000000000000000000000,

то, как легко видеть, для использования шкалы необходимы два логических оператора P_5 и P_6. P_5 должен сдвинуть шкалу влево на 2 разряда и выделить с помощью указателя две первые цифры. Если при этом будет получен набор нулей, то управление должно быть передано оператору A_5^i. В противном случае должен быть произведен переход к оператору P_6. Оператор P_6 должен выделенный набор цифр сравнить с «$c+3$». Если наборы «$c+3$» и выделенный совпадут, то управление должно быть передано оператору $F_4(j)$, а в противном случае оператору A_7. Таким образом, логическая схема (6.9) принимает вид

$$И_0П_1A_2\underset{1}{\lrcorner}A_3^j\underset{2}{\lrcorner}F_4(j)\,P_5\overset{1}{\urcorner}P_6\overset{2}{\urcorner}A_7П_8Я_9.$$

2. Операторы формирования. Иногда при программировании отдельная команда или группа команд не может быть составлена, потому что ее адреса заранее неизвестны и станут известными после того, как часть программы будет выполнена. В таком случае в программу включают так называемый оператор формирования (обозначается Φ_y), который должен составить недостающую команду (группу команд).

В программах для машин с переадресацией в УУ оператор формирования может составлять заранее неизвестный вспомогательный код для видоизменения команды.

Наиболее простым является формирование по параметру. В этом случае формируемый оператор должен работать после некоторого цикла, а его заранее неизвестные команды определяются значением параметра, при котором прекращается выполнение цикла. При этом логическая схема программы может содержать выражение

$$\pi \underset{m}{\lrcorner} A_k^i F_{k+1}(i) P_{k+2} \lambda_m \Phi_{k+3} Q_{k+4} \lambda_n.$$

В ряде случаев формирование равносильно переадресации по параметру оператора, внешнего для цикла, в котором стоит оператор переадресации. В программах для машин с переадресацией в УУ замена формирования переадресацией может быть выгодной.

Пример 6.11. Задана монотонно возрастающая последовательность чисел $x_1, x_2, \ldots, x_n$ и требуется составить часть логической схемы программы, которая определяла бы 1) выбор из последовательности $x_1, x_2, \ldots, x_n$ чисел, удовлетворяющих условию $x_i > k$, и 2) перенос их в последовательные ячейки, начиная с $c+1$.

Интересующая нас часть логической схемы может иметь вид

$$\underset{2}{\lrcorner} P_1^i \underset{1}{\llcorner} F_2(i) P_3 \underset{2}{\llcorner} Я_4 \underset{1}{\lrcorner} \Phi_5 \Pi_6 \lambda_3, \tag{6.10}$$

где P_1^i проверяет условие $x_i < k$, если оно выполнено, передает управление оператору $F_2(i)$, если же не выполнено, то оператору Φ_5. Оператор P_3 проверяет, все ли числа $x_1, x_2, \ldots, x_n$ уже обследованы, т. е. проверяет условие $i = n$. Если это условие выполнено, то $Я_4$ останавливает машину, если же не выполнено, то управление получает снова P_1^i. Оператор Φ_5 формирует команды переноса, входящие в Π_6 (для многих машин Π_6 будет являться обобщенным оператором; именно для машин, система команд которых предусматривает только одиночные переносы чисел). Легко видеть, что формирование оператора Π_6 является формированием по параметру i, причем формируемые команды зависят от i линейно.

Оператор Π_6 не может быть составлен в окончательном виде: заранее неизвестно, в какой ячейке стоит первое из чисел $x_i \geqslant k$.

Составим часть программы, изображаемую схемой (6.10), в системе команд машины *Стрела*.

Предположим, что числа x_i ($i = 1, 2, \ldots, n$) и k размещены в памяти следующим образом:

$$\beta)\ k;\ \beta+1)\ x_1;\ \beta+2)\ x_2;\ \ldots;\ \beta+n)\ x_n.$$

Вспомогательные коды по мере их появления в процессе программирования будем размещать в ячейках $\gamma+1, \gamma+2, \ldots$ Числа x_i ($x_i \geqslant k$) будем переносить в ячейки $\nu+1, \nu+2, \ldots$

Рабочие ячейки обозначим через $\mu+1, \mu+2, \ldots$ Адреса команд, зависящие от параметра i, будем отмечать звездочкой.

Оператор P_1 состоит из двух команд:

$$\begin{array}{llll} \alpha+1) & \beta+1^* & \beta & 0000\ 0\ 03 \\ \alpha+2) & \delta+1 & \alpha+3 & 0000\ 0\ 03 \end{array}$$

Здесь $\delta+1$ — номер ячейки, хранящей первую команду оператора Φ_5. При обозначении этой ячейки использована буква δ, ввиду того что мы не знаем заранее, сколько команд будут содержать операторы $F_2(i)$, P_3 и $Я_4$.

Оператор $F_2(i)$ будет

$$\alpha+3 \quad \alpha+1 \quad \gamma+1 \quad \alpha+1 \quad 0 \quad 02$$

Эта команда для переадресации команды «$\alpha+1$» использует код

$$\gamma+1) \quad 0001 \quad 0000 \quad 0000 \quad 0 \quad 00$$

Оператор P_3:

$$\begin{array}{llll} \alpha+4) & \alpha+1 & \gamma+2 & 0000\ 0\ 16 \\ \alpha+5) & \alpha+6 & \alpha+1 & 0000\ 0\ 20 \end{array}$$

Команда «$\alpha+4$» использует вспомогательный код

$$\gamma+2) \quad \beta+n+1 \quad \beta \quad 0000 \quad 0 \quad 03$$

Оператор $Я_4$:

$$\alpha+6) \quad 0000 \quad 0000 \quad 0000 \quad 0 \quad 40$$

Оператор Φ_5 мы сможем составить лишь после того, как примерно будет составлен формируемый оператор $П_6$. Этот последний сделаем обобщенным (т. е. некоторым циклом), несмотря на то, что групповой перенос с помощью одной команды в системе команд машины *Стрела* (см. табл. 15.1) специально предусмотрен.

Оператор $П_6$:

$$\begin{array}{llll} \varepsilon+1) & 0000 & 0000 & \nu+1\ 0\ 45 \\ \varepsilon+2) & \varepsilon+1 & \gamma+3 & \varepsilon+1\ 0\ 02 \\ \varepsilon+3) & \varepsilon+1 & \gamma+4 & \mu+1\ 0\ 11 \\ \varepsilon+4) & \mu+1 & \gamma+5 & 0000\ 0\ 16 \\ \varepsilon+5) & \varepsilon+6 & \varepsilon+1 & 0000\ 0\ 20 \end{array}$$

Первый адрес команды «$\varepsilon+1$» нам заранее неизвестен, и мы его временно положили равным 0000.

Окончательный вид этого адреса определить нетрудно, ибо перенос чисел x_i, удовлетворяющих неравенству $x_i \geqslant k$, предусматриваемый командой «$\varepsilon+1$», закончится переносом числа x_n из ячейки $\beta+n$, после чего команда «$\varepsilon+1$» подвергнется еще один раз переадресации с помощью кода $\gamma+3)$ 0001 0000 0001 0 00. Этот последний вид упомянутого адреса мы «запасли» в ячейке $\gamma+5$ в виде кода

$$\gamma+5) \quad \beta+n+1 \quad 0000 \quad 0000 \quad 0 \quad 00$$

В ячейке $\gamma+4$ помещаем вспомогательный код для выделения из команды «$\varepsilon+1$» ее первого адреса:

$$\gamma+4) \quad 7777 \quad 0000 \quad 0000 \quad 0 \quad 00$$

Команда $\varepsilon+3$ выделяет первый адрес команды «$\varepsilon+1$», а команда «$\varepsilon+4$» сравнивает его с «$\gamma+5$». Таким образом проверяется,

все ли числа, подлежащие переносу, уже перенесены. Если перенесены не все, то управление получает «$\varepsilon+1$», если все, то управление получит команда «$\varepsilon+6$», которая уже не входит в $П_6$.

Оператор $Ф_5$:

$$\begin{array}{llllll} \delta+1) & \alpha+1 & \gamma+4 & \mu+1 & 0 & 11 \\ \delta+2) & \varepsilon+1 & \mu+1 & \varepsilon+1 & 0 & 13 \end{array}$$

Этот оператор выделяет с помощью вспомогательного кода «$\gamma+4$» первый адрес команды «$\alpha+1$» и записывает временно в рабочую ячейку $\mu+1$, а затем с помощью операции специального сложения «вставляет» выделенный адрес в команду «$\varepsilon+1$».

При распределении памяти необходимо положить

$$\delta=\alpha+6;\quad \varepsilon=\delta+2.$$

Само собой разумеется, что команды, составленные в действительных адресах, должны быть переписаны на бланки в порядке номеров отведенных для них ячеек памяти.

В данном случае вместо формирования команды «$\varepsilon+1$» ее можно было бы получить с помощью оператора переадресации, в который следовало бы получить после команды «$\alpha+3$» команду

$$\alpha'+3)\quad \varepsilon+1\quad \gamma+1\quad \varepsilon+1\quad 0\quad 02$$

При этом команде «$\varepsilon+1$» нужно было бы вместо начального вида 0000 0000 $\nu+1$ 0 45 придать начальный вид

$$\varepsilon+1)\quad \beta+1^*\quad 0000\quad \nu+1\quad 0\quad 45.$$

Это было бы менее выгодно (по количеству расходуемых рабочих тактов), чем вышеописанное формирование.

При составлении программы в системе команд машины *Урал-1* (эта машина предусматривает переадресацию в УУ) оператор формирования был бы не нужен, достаточно было бы предусмотреть изменение команд переноса с помощью кода, примененного для переадресации в УУ команды оператора P^i_1, зависящей от параметра i. Другими словами, для этой машины замена формирования по параметру переадресацией по этому параметру в данном случае выгодна.

Выражение (6.10) следовало бы заменить следующим:

$$\underset{2}{\lrcorner} P^i_1 \underset{1}{\llcorner} F_2\,(i)\; P_3 \overset{2}{\ulcorner} Я_4 \underset{1}{\llcorner} П^i_6 \lambda_3.$$

Значительно более сложным является случай формирования по исходным данным или промежуточным результатам решаемой задачи. В этом случае при переадресации в УУ «исчезновение» оператора формирования не происходит, хотя бы формировалась не команда оператора Q, а код, прибавляемый к УУ к некоторому начальному виду этой команды.

Пример 6.12. Предположим, что в памяти машины задана таблица значений функции $y_i=f(x_i)$, где $x_i=x_0+ih$ ($h=\text{const}$), причем сами значения x_i в память не введены, а лишь в ячейках $b+i$ ($i=1, 2, \ldots, n$) размещены y_i (таблица с равномерным шагом).

Для того чтобы по заданному x найти y_i и y_{i+1}, для которых

$$x_i \leqslant x < x_{i+1},$$

адреса $b+i$ и $b+i+1$ вычисляются с помощью формулы

$$i = E\left(\frac{x-x_0}{n}\right),$$

где $E(z)$ означает целую часть числа z.

Найденные адреса $b+i$ и $b+i+1$ должны быть включены в команды, с помощью которых машина «возьмет» числа y_i и y_{i+1} для вычислений (например, для получения с их помощью путем интерполяции приближенного значения $y=f(x)$). В этом случае применяется формирование по исходному данному x (подробнее см. [30, стр. 433]).

§ 5. Выделение подпрограммы

1. Понятие подсхемы и подпрограммы. Выделение подпрограммы. Пусть

$$\Sigma_1 = \Sigma\{W_1, W_2, \ldots, W_n\} \tag{6.11}$$

— логическая схема программы, в которую входят выражения $W_1, W_2, \ldots, W_n$. Допустим, что при анализе схемы установлена равносильность каждого выражения $W_2, W_3, \ldots, W_n$ с выражением W_1. Заменим каждое выражение W_k $(k \geqslant 2)$ в схеме Σ_1 соответственно символом E_k, обозначающим совокупность перехода к W_1 и получения управления от него после его выполнения. Этот символ будем называть *оператором обращения* к W_1. При этом схема $\Sigma\{W_1, W_2, \ldots, W_n\}$ преобразуется в равносильную ей схему

$$\Sigma_2 = \Sigma(W_1, E_2, E_3, \ldots, E_n\}.$$

В этом случае выражение W_1 называют *подсхемой*, а описанное преобразование исходной логической схемы называют *выделением подсхемы*.

Логическую схему Σ_2 можно также привести к виду

$$\Sigma_3 = \Sigma\{E_1, E_2, E_3, \ldots, E_n\}\, W_1,$$

где E_1 — также оператор обращения к W_1. Такое преобразование называют *выносом* подсхемы W_1.

Если составить программы T_1, T_2 и T_3, соответствующие логическим схемам Σ_1, Σ_2 и Σ_3, то выражениям $W_1, W_2, \ldots, W_n$, присутствующим в Σ_1, в программе T_1 будут отвечать некоторые части этой программы $A_1, A_2, \ldots, A_n$.

В программе T_2 части, отвечающие выражениям W_2, W_3, ..., W_n, будут заменены группами команд, осуществляющими обращения к части программы A_1, соответствующей W_1. При этом A_1 называют *подпрограммой*.

Построение программы T_2 вместо T_1 представляет собой прием программирования, называемый *выделением подпрограммы*.

О программе T_3, отвечающей логической схеме $\sum_3$, говорят, что в ней произведен *вынос* подпрограммы.

Выделение подсхемы в тех случаях, когда выражения W_i состоят каждое из многих операторов, упрощает запись логической схемы и делает ее более обозримой.

2. Построение обращений к выделенной подпрограмме. Пусть $A_1, A_2, \ldots, A_n$ — части программы T_1, соответствующие выражениям W_1, W_2, ..., W_n в схеме $\sum_1$ (см. (6.11)). Те команды части A_i, которые получают управление извне, назовем *входами*, а те, которые передают управление вовне, — *выходами*. В случае, когда возможно вынесение подпрограммы A_1, каждому входу этой подпрограммы соответствует некоторый вход части A_i ($l \geqslant 2$). Такие входы называются между собою *подобными*. Точно также каждому выходу из A_1 соответствует некоторый выход на A_i. Такие выходы называются между собой *подобными*.

Группу команд, осуществляющую обращение к подпрограмме и соответствующую оператору обращения в логической схеме программы, называют элементарным оператором обращения.

Выделение A_1 в качестве подпрограммы можно произвести следующим образом:

1. Ввести по одной «свободной» ячейке после каждого выхода из A_1, отличного от команды условного или безусловного перехода (каждый такой выход передает управление в следующую за ним ячейку). Эти свободные ячейки отводятся для засылки в них команд возврата (см. п. 2, б)) и в дальнейшем считаются выходами из подпрограммы A_1.

2. Заменить каждую часть программы A_i оператором обращения к A_1. Оператор обращения должен предусматривать:

а) засылку необходимых исходных данных в ячейки, из которых подпрограмма их «берет» для своих вычислений

(если эти исходные данные не записываются в упомянутые ячейки другими операторами программы);

б) засылку в выходы подпрограммы так называемых команд возврата, т. е. команд перехода, передающих управление туда, куда его передавали подобные выходы из A_i;

в) передачу управления соответствующему входу подпрограммы;

г) в некоторых случаях после конца работы подпрограммы необходимо производить стирание команд возврата или восстановление прежнего содержимого ячеек, содержащих команды возврата, а также, может быть, перенос результатов работы подпрограммы в другие рабочие ячейки. Иногда удобно считать, что все это выполняет группа команд обращения к подпрограмме. Тогда эта группа команд уже не является оператором, ибо имеет несколько входов и выходов. Однако для удобства ее можно считать элементарным оператором.

Наиболее удобны подпрограммы, имеющие только один выход. Если входом такой подпрограммы является оператор с номером m, а выходом — оператор с номером n, то оператор обращения обозначают символом E_v $(m; n)$. Некоторые машины располагают специальными командами для осуществления таких обращений. Например, машина *Стрела* располагает командой с кодом операций 27. По команде

$$k)\ a\ a\ c\ 0\ 27$$

производится запись в ячейку c команды возврата

$$c)\ k+1\ \ k+1\ \ c\ 0\ 20$$

и передача управления в ячейку a. При этом команда возврата является «самогасящейся» или, как ее иногда называют, «самоубийцей». Будучи при ее выполнении перенесенной в регистр команд, эта команда обусловливает запись в ячейку c пустого кода 0000 0000 0000 0 00 и безусловный переход к команде с номером $k+1$. Команды, облегчающие обращение к подпрограммам, имеют также машины *БЭСМ* и *Урал-2*.

Пример 6.13. Необходимо составить программу вычисления величины

$$y = \sin^2\left[\sum_{j=1}^{n} \sin x_j\right]$$

по заданным числам $x_1, x_2, \ldots, x_j, \ldots, x_n$.

Известно, что

$$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

Последний ряд является при любом значении x знакочередующимся, следовательно, абсолютная величина ошибки при замене суммы ряда его частной суммой не превосходит абсолютной величины последнего из удерживаемых членов ряда.

Если обозначить через u_i член ряда с номером i, т. е.

$$u_i = \frac{x^{2i-1}}{(2i-1)!},$$

то

$$u_{i+1} = u_i\left[-\frac{x^2}{2i\,(2i+1)}\right]. \qquad (6.12)$$

Последней зависимостью можно воспользоваться для вычисления $\sin x$.

При этом логическая схема нужной нам программы будет иметь вид

$$И_0 \underset{2}{\lrcorner} З_3^j A_4 \underset{1}{\lrcorner} A_5 P_6 \underset{1}{\llcorner} П_7^j F_8(j)\, P_9 \underset{2}{\llcorner} A_{10} A_{11} \underset{3}{\lrcorner} A_{12} P_{13} \underset{3}{\llcorner} A_{14} Я_{15}. \qquad (6.13)$$

В приведенной логической схеме оператор $З_3^j$ переносит число x_j в рабочие ячейки оператора A_4; оператор A_4 вычисляет величину $-x_j^2$; оператор A_5 по известной величине u_i с помощью формулы (6.12) определяет u_{i+1} и, складывая величины $u_1 + u_2 + \ldots$, последовательно получает частные суммы ряда s_1, s_2, ...; оператор P_6 проверяет выполнение неравенства $|u_{i+1}| < \varepsilon$ ($\varepsilon > 0$ задано) и, пока это условие не выполнено, передает управление оператору A_5; оператор $П_7^j$ вычисленное значение $\sin x_j$ из рабочей ячейки оператора A_5 переносит на хранение; оператор $F_8(j)$ производит переадресацию операторов $З_3^j$ и $П_3^j$; оператор P_9 проверяет условие $j = n + 1$ и, пока это условие не выполнено, передает управление оператору $З_3^j$; оператор A_{10} вычисляет величину $z = \sum_{j=1}^{n} \sin x_j$ и записывает ее в рабочую ячейку оператора A_{11}; оператор A_{11} вычисляет величину $-z^2$. Оператор A_{12} по известной величине u_k вычисляет u_{k+1} (где u_k — член разложения в ряде функции $\sin z$) и, складывая величины $u_1 + u_2 + \ldots$, последовательно получает частные суммы s_1, s_2, ... степенного ряда для $\sin z$; оператор P_{13} проверяет неравенство $|u_{k+1}| < \varepsilon$ и, пока оно не выполнено, передает управление оператору A_{12}; оператор A_{14} вычисляет квадрат значения $\sin z$, полученного оператором A_{12}; оператор $Я_{15}$ останавливает машину.

Рассматривая схему программы (6.13), видим, что

$$A_4 \underset{1}{\lrcorner} A_5 P_6 \underset{1}{\llcorner} \quad \text{и} \quad A_{11} \underset{3}{\lrcorner} A_{12} P_{13} \underset{3}{\llcorner}$$

представляют одинаковые части программы. Выделяя

$$A_4 \underset{1}{\lrcorner} A_5 P_6 \underset{1}{\llcorner}$$

в качестве подпрограммы, приводим (6.13) к виду

$$И_0 \underset{2}{\rfloor} З_3^j A_4 \underset{1}{\rfloor} A_5 \overline{P}_6 \underset{1}{\lfloor} П_7^j F_8 (j)\, P_6 \underset{2}{\lfloor} A_{10} E_{11}\ (4;\ 6)\ A_{14} Я_{15}.$$

В последней схеме команда возврата засылается в свободную ячейку, непосредственно следующую за командой условного перехода, принадлежащей оператору P_6 (что отмечено чертой над знаком этого оператора). Оператор обращения E_{11} (4; 6) предусматривает засылку упомянутой команды возврата и передачу управления оператору A_4. Команда возврата после работы подпрограммы должна передать управление оператору A_{14}. Оператор A_{10} вычисляет величину $z = \sum_{j=1}^{n} \sin x_j$ и записывает ее в рабочую ячейку оператора A_4. Оператор A_{14} вычисляет квадрат значения $\sin z$, полученного оператором A_4.

Литература к главе 6: [7], [8], [11], [16], [37], [68].

ГЛАВА 7
АВТОМАТИЗАЦИЯ ПРОГРАММИРОВАНИЯ

§ 1. Автоматизация отдельных работ при программировании вручную

Стремление облегчить труд программистов и уменьшить количество допускаемых ими ошибок породило идею использовать саму электронную цифровую машину для выполнения технической работы, связанной с программированием, т. е. идею автоматизации такой работы. При этом (а) выявлялись правила, по которым такую работу мог бы выполнять человек, (б) устанавливалась форма, в которой задается исходная информация, и форма, в которой должен быть выдан результат, (в) разрабатывались алгорифмы получения искомого результата по исходной информации, (г) составлялась программа для выполнения указанных алгорифмов на электронной цифровой машине.

1. Автоматизация присвоения истинных адресов. После составления программы в символических адресах (в буквенно-числовых обозначениях), подсчета количества ее команд и числа ячеек, необходимых для всего материала, относящегося к программе, и распределения памяти (см. гл. 4) должна быть выполнена формальная работа — присвоение истинных адресов. Эта работа заключается в замене буквенных компонент символических адресов определенными числами и сложении последних с числовыми компонентами символических адресов. Исходной информацией является таблица значений букв и программа в символических адресах, требуемым результатом — программа в коде команд машины. Если машина допускает как цифровой, так и буквенный ввод, то составление программы присвоения истинных адресов не представляет

трудностей. Если машина оборудована только цифровым вводом, то целесообразно уже при программировании пользоваться цифровыми символическими адресами. Например, вместо $a+1$, $a+2$, ... можно писать 1001, 1002, ..., вместо $b+1$, $b+2$, ... — писать 2001, 2002, ... и т. п. При этом в состав информации, исходной для программы присвоения истинных адресов, войдет вместо таблицы значений букв таблица соответствия символических адресов истинным адресам. В такой таблице достаточно указать истинные адреса, соответствующие первым символическим адресам каждой группы.

Присвоение истинных адресов в ряде случаев можно совместить с распределением памяти. При этом информация, исходная для программы присвоения истинных адресов, может вместо таблицы значений букв или таблицы соответствия истинных адресов символическим содержать таблицу последовательности групп символических адресов. Совмещенный процесс присвоения истинных адресов и распределения памяти заключается в том, что символические адреса в порядке, заданном таблицей их последовательности, заменяются последовательными истинными адресами, начиная с заданного истинного адреса.

В качестве примера программы присвоения истинных адресов можно указать разработанную для машины *Стрела* программу автоматического присвоения адресов (ПАПА), обладающую высокой универсальностью [52].

2. Автоматизация исправления некоторых ошибок. При проверке программы, в частности при отладке ее на машине, нередко обнаруживают ошибки, состоящие в пропуске одной или нескольких команд. Исправление программы путем «латания», т. е. путем записи пропущенных команд в свободных ячейках памяти (которые могут оказаться оторванными от массива ячеек, хранящих программу) и построения перехода к ним из программы и возврата от них в программу, представляет ряд неудобств. Такое исправление усложняет структуру программы и может сделать неудобным ее ввод. Второй способ исправления состоит в том, что программу раздвигают, на освободившееся место вписывают пропущенные команды, а остальные команды программы просматривают и, если нужно, изменяют их адреса. При выполнении вручную это равносильно пересоставлению программы заново. Переделка про-

граммы, подобная описанной, может потребоваться не для исправления программы, а для приспособления ее к решению новой задачи. При этом, кроме вставления новых команд, может потребоваться удаление из программы некоторых групп команд, а также замена некоторых команд или групп команд новыми. Иногда бывает желательным перемещение какой-либо группы чисел из одних ячеек памяти в другие (т. е. «перенумерация» группы чисел). Для описанных переделок и исправлений программ можно составить специальную исправляющую программу. Исходная информация для исправляющей программы будет слагаться из исправляемой программы и таблицы переделок, указывающей места и характер переделок и содержащей группы команд, подлежащие включению в исправляемую программу. Примером исправляющей программы может служить программа ВУЗП (вставки, удаления, замены, перенумерации), разработанная для машины *Стрела* [37], а затем для *М-20*.

3. Автоматизация отладки программ. Для облегчения трудоемкого процесса отладки программ на машине можно применять специальные отлаживающие программы. Исходной информацией для таких программ является отлаживаемая программа и дополнительная информация, необходимая для отладки. Выдаваемые результаты могут быть весьма разнообразными и в значительной своей части являются материалом, который должен подвергаться анализу программистом. Большинство таких программ копирует ту или иную работу, выполняемую программистом при отладке вручную (см. гл. 3, § 3). Такие программы могут производить арифметическую и логическую проверку частей отлаживаемой программы, контроль правильности переадресации, восстановления и формирования команд, проверку количества выполнения циклов и т. п. Сравнение промежуточных результатов с их заготовленными значениями эти программы производят автоматически и в случае несовпадения результатов выдают информацию о том, какая часть отлаживаемой программы дает неправильные результаты.

Наиболее обстоятельную проверку отлаживаемой программы можно произвести, выполняя по отдельности каждую ее команду и анализируя результат ее выполнения. При отладке вручную это практически невозможно, так как требует огромного расхода машинного времени. При автоматической

отладке программы можно имитировать отладку в однотактном режиме, выдавая на печать информацию о порядке выполнения команд, о содержимом некоторых интересующих программиста ячеек и т. д. Для осуществления такой имитации, отлаживающая программа должна содержать специальную часть, называемую «прокруткой», которая моделировала бы работу машины, выбирая из отлаживаемой программы команды, исполняя их и фиксируя необходимые результаты. Так как программисту нужна информация о результатах работы не всех команд, отлаживающая программа должна содержать часть, называемую «расшифровкой», назначение которой — определять, нужно ли и какую выдавать информацию о проработавшей команде. Примером подобной отлаживающей программы может служить так называемая «Программа автоматического контроля программ», разработанная в ВЦ МГУ [65]. Примером более простой отлаживающей программы, предназначенной для узкой цели проверки логики отлаживаемой программы, является разработанная в ВЦ МГУ программа ППУ — «печать передач управления» [65]. Из отладочных программ узкого назначения укажем еще разработанные в ВЦ АН СССР программы ПИК — «печать изменяемых команд» [61], которая служит для проверки правильности варьирующих операторов, и ЛУЧ — «линейные участки» [60], которая служит для проверки правильности так называемых линейных участков отлаживаемой программы, т. е. участков, не содержащих внутри себя логических операторов.

Существенным недостатком способов отладки, выполняемой вручную, а также способов, основанных на применении описанных отлаживающих программ, является то, что если автономную отладку с их помощью можно произвести с большой полнотой, то комплексная отладка при этом проводится весьма неполно. Это связано с тем, что для всесторонней проверки работы программы в комплексе необходимо решить большое число примеров, что часто практически невозможно.

Указанный недостаток заставляет искать новые способы отладки.

Г. Д. Фроловым разработан новый метод отладки, заключающийся в автоматическом декодировании отлаживаемой программы (с помощью «декодирующей программы»), и получения по ней математического описания решаемой задачи на одном из алгорифмических языков. Это описание (в частности

логическая схема) должно быть сопоставлено с исходным описанием (в частности, логической схемой), по которому велось программирование. В настоящее время этот метод реализован в виде программы, выдающей результаты на алгорифмическом языке АЛГОЛ (см. гл. 8, § 2).

§ 2. Автоматизация составления программ

1. Сущность автоматизации программирования. Программирование состоит из двух основных этапов: разработки логической схемы программы по математическому описанию решающего алгорифма и разработки программы по ее логической схеме. В дальнейшем под автоматизацией программирования понимается автоматизация второго этапа. Автоматизация программирования слагается из

а) выбора формы задания исходной информации,

б) разработки алгорифмов получения требуемого результата,

в) составления программы для выполнения указанных алгорифмов на электронной цифровой машине.

Результатом автоматического программирования является готовая программа, составленная в коде команд определенной машины. Таким образом, форма представления требуемого результата (т. е. *выходной язык*) заранее установлена. От формы задания исходной информации (т. е. от *входного языка*) в значительной степени зависят

а) сложность алгорифмов переработки исходной информации в требуемый результат, а значит и сложность и трудоемкость разработки программы, осуществляющей автоматическое программирование;

б) трудности, связанные с заданием исходной информации, а следовательно и успех внедрения автоматического программирования взамен программирования вручную.

В настоящее время уделяется большое внимание вопросу разработки входных языков для автоматического программирования.

2. Способы автоматического программирования. При автоматическом программировании, как и при программировании вручную, возможны два способа. Первый из них заключается в том, что каждый раз по исходной инфор-

мации производится составление всей программы заново. Именно этот способ был описан в предыдущих главах. Программы, осуществляющие автоматическое программирование первым способом, получили название *программирующих программ* (коротко — ПП). Метод программирования, основанный на использовании ПП, называют *методом программирующих программ* (коротко — методом ПП). Второй способ программирования заключается в том, что отдельные части программ сохраняют и как целое применяют при составлении новых программ. При втором способе заново программируют лишь такие части программ, которые еще не были «запасены». Коллекции заготовленных частей программ (так называемых *подпрограмм*) получили название *библиотек подпрограмм*. Метод программирования, связанный со вторым способом, называют *методом библиотечных подпрограмм*.

При методе библиотечных подпрограмм, кроме собственно подпрограмм необходимо иметь все сведения, нужные для использования подпрограмм. Без наличия таких сведений, программисту приходилось бы каждую подпрограмму перед ее применением расшифровывать. Необходимые сведения о подпрограммах можно значительно сократить, если в библиотеку собирать не «случайно» полученные части программ, а специально разрабатываемые подпрограммы, удовлетворяющие определенным требованиям, дающим возможность описать совокупность действий, необходимых для применения произвольной подпрограммы, в виде единого алгорифма. В этом случае подпрограммы называют *стандартными подпрограммами* (СП), а их библиотеку — *библиотекой стандартных подпрограмм* (БСП). Библиотека стандартных подпрограмм в совокупности с алгорифмом для их использования (который, в частности, может быть оформлен в виде программы) и правилами задания исходной информации образует *систему стандартных подпрограмм*. Метод библиотечных подпрограмм при программировании вручную широкого распространения не получил. Значительно большие успехи достигнуты на пути автоматизации программирования, основанного на применении БСП. Автоматизированный метод программирования с помощью БСП в дальнейшем называется *методом библиотеки стандартных подпрограмм* (коротко — методом БСП).

Возможна автоматизация программирования, при которой сочетаются метод ПП и метод БСП.

3. Компиляция и интерпретация. Если программирование и выполнение программы на машине рассматривать как единый процесс решения задачи на машине, то возможны два порядка осуществления этого процесса. Первый порядок, назовем его *раздельным*, предусматривает: а) составление всей программы; б) выполнение программы на машине при заданных исходных данных (т. е. собственно решение задачи). Этот порядок широко распространен при ручном программировании и в ряде случаев предусматривается при автоматическом программировании. Второй порядок назовем *чередующимся*. Он заключается в том, что исходная информация о программе (или неполная программа и исходная информация о недостающих ее «кусках») вводится в машину вместе с исходными данными задачи. Общий процесс решения задачи на машине состоит из чередующихся этапов автоматической переработки частей исходной информации о программе в «куски» программы и этапов выполнения «кусков» программы и заканчивается получением искомого результата задачи. При этом перерабатываемыми частями исходной информации являются группы кодов, которым соответствуют законченные элементы программы, например, команды, операторы или подпрограммы. «Куски» программы, получающиеся на этапах автоматического программирования, после выполнения не сохраняются.

Автоматическое программирование при раздельном порядке решения задач называется *компиляцией* программы, а при чередующемся порядке — *интерпретацией* (исходной информации о программе). Возможно сочетание компиляции с интерпретацией, заключающееся в том, что компилируется программа, в состав которой кроме команд включены специальные коды, называемые *псевдокомандами*. Псевдокоманды интерпретируются уже при выполнении составленной программы на машине. Возможна и такая комбинация компиляции с интерпретацией, при которой основной является интерпретация, однако получаемые при ней куски программы, если позволяет объем оперативной памяти, сохраняются и включаются в исходную информацию о программе вместо соответствующих этим кускам групп кодов (см. § 3 этой главы).

Если при компиляции программа, осуществляющая автоматическое программирование, нужна только для получения программируемой программы, то при интерпретации интерпретирующая программа находится в оперативной памяти в

течение всего решения задачи. Каждый из методов (компиляции и интерпретации) имеет свои достоинства, однако метод интерпретации является более гибким. Например, в программе могут быть предусмотрены операторы формирования псевдокоманд в тех случаях, когда заранее неизвестна подпрограмма, которая потребуется для решения задачи, и т. п. Кроме того, метод интерпретации упрощает задачу распределения памяти, хотя и требует большого дополнительного расхода памяти для хранения самой интерпретирующей программы. При методе компиляции более экономно расходуются ячейки оперативной памяти, так как компилирующая программа в ней во время решения задачи не присутствует.

Ввиду того что способы программирования можно любым образом сочетать с порядком решения задач на машине, возможны ПП как компилирующие, так и интерпретирующие; точно так же метод БСП может осуществляться как путем компиляции, так и путем интерпретации.

§ 3. Метод библиотеки стандартных подпрограмм

1. Общие сведения о методе БСП. Как уже было сказано, система стандартных подпрограмм состоит из библиотеки подпрограмм и организующей программы (компилирующей или интерпретирующей). Остановимся подробнее на подпрограммах.

В каждой системе подпрограмм применяется свой метод составления подпрограмм, отличный от обычных способов программирования в кодах той машины, на которой применяются подпрограммы. Это связано с двумя обстоятельствами. Во-первых, место оперативной памяти, на котором будет выполняться подпрограмма, заранее неизвестно и подпрограмма должна допускать так называемую *настройку по месту*. Во-вторых, часто взамен большого количества *конкретных* подпрограмм составляют небольшое количество подпрограмм общего вида. Например, вместо подпрограмм для решения системы одного, двух, трех и т. д. алгебраических линейных уравнений можно составить подпрограмму для решения системы n уравнений.

Подпрограмма общего вида должна быть составлена так, чтобы ее можно было перед употреблением путем незначительной переработки, называемой *конкретизацией*, преобразовать в любую из объединяемых ею конкретных подпрограмм.

Конкретизация подпрограмм общего вида заключается в том, что некоторым величинам, называемым *параметрами* подпрограммы, придают определенные числовые значения. Например, параметром подпрограммы для решения системы n уравнений является величина n.

Таким образом, адреса ряда команд и вспомогательных кодов стандартных подпрограмм, а также некоторые входящие в их состав коды, не имеющие структуры команд (например, коды чисел), перед использованием подпрограммы должны быть переработаны. Эти переменные элементы подпрограмм составляются по особым правилам, зависящим от алгорифма, лежащего в основе организующей программы, что и порождает специфику методов составления подпрограмм.

Адреса в подпрограммах делятся, во-первых, на *зависящие* и *не зависящие от параметров* и, во-вторых, на *зависящие* и *не зависящие от места установки* подпрограммы. Адреса, зависящие от места установки, делят на *внутренние* (являющиеся адресами команд и кодов самой подпрограммы) и *внешние* (являющиеся адресами команд или кодов остальной части составляемой программы). Легко видеть, что всего существует шесть разновидностей адресов. Адреса, не зависящие ни от параметров, ни от места установки подпрограммы, называются *постоянными*. Элементы подпрограммы, не имеющие структуры команд, делятся на зависящие от параметров и не зависящие от них.

Стандартные подпрограммы, не содержащие внешних адресов, называются *замкнутыми*, в отличие от подпрограмм, содержащих внешние адреса и называемых *открытыми*.

Внутренние адреса, не зависящие от параметров, можно составлять как символические адреса (см. гл. 4, § 1). Существует и другой способ их составления — в виде так называемых *условных адресов*, при котором они записываются как действительные адреса, соответствующие какому-либо определенному способу расположения подпрограмм в памяти (например, в последовательных ячейках, начиная с ячейки, имеющей номер 2000 и кончая ячейкой с номером 3777).

Метод БСП может встречаться как самостоятельный метод программирования или на более скромных ролях как способ расширения системы команд при каком-либо другом методе программирования. Именно в последней роли метод БСП и завоевал себе твердое место.

2. Интерпретирующая система ИС-2. Интерпретирующая система ИС-2 разработана в Математическом институте АН СССР коллективом сотрудников под руководством М. Р. Шура-Бура. Принципы этой системы могут быть применены при разработке интерпретирующей системы для любой программно-управляемой машины.

Необходимо заметить, что в ИС-2 метод интерпретации комбинируется с методом компиляции, что дает некоторый выигрыш машинного времени по сравнению с «чистым» методом интерпретации.

Стандартные подпрограммы системы ИС-2 являются замкнутыми и конкретными. Каждая из них имеет один вход и один выход и вместе со своими константами представляет единый массив. Стандартные подпрограммы составлены в условных адресах (т. е. в действительных адресах, начиная с 2000). Константы каждой подпрограммы расположены непосредственно вслед за ее командами. Стандартная подпрограмма, будучи введена в оперативную память, начиная с ячейки 2000, для своего применения не требует никакой переработки. Если же она введена начиная с ячейки $a_{\text{нач}}$, то ее внутренние адреса (бо́льшие или равные 2000) должны быть переработаны путем прибавления к каждому из них поправки, равной $a_{\text{нач}} - 2000$.

3. Организация интерпретирующей системы ИС-2. Интерпретирующая система ИС-2 состоит из системы стандартных подпрограмм, интерпретирующей программы (ИП) и постоянной таблицы характеристик (ПТХ).

Интерпретирующая программа (ИП) вводится в оперативную память вместе с программой решаемой задачи (которую будем называть основной программой).

Для того чтобы при программировании основной программы можно было не учитывать расположения стандартных подпрограмм на магнитной ленте (или барабане), вызов подпрограмм осуществляется путем указания в псевдокомандах их номеров. Остальная информация, необходимая для вызова стандартных подпрограмм, сосредоточена в ПТХ *).

*) Для вызова подпрограммы с магнитной ленты необходимо указать номер зоны, в которой она записана, количество строк в этой зоне и номер ячейки оперативной памяти, начиная с которой должна быть введена подпрограмма.

Каждая строка ПТХ содержит данные, нужные для ввода подпрограммы, имеющей номер, равный номеру этой строки (считая от начала ПТХ).

Каждый раз, когда при выполнении основной программы встречается псевдокоманда, требующая выполнения некоторой стандартной подпрограммы, происходит передача управления интерпретирующей программе. ИП по номеру стандартной подпрограммы находит в ПТХ необходимые данные для вызова подпрограммы. Часть оперативной памяти — так называемое *рабочее поле* — специально предназначена для записи стандартных подпрограмм. Вызываемые подпрограммы записываются на рабочем поле одна вслед за другой до тех пор, пока позволяет емкость рабочего поля. После вызова каждой подпрограммы, начиная с последней ячейки рабочего поля в порядке убывания номеров ячеек, ИП записывает строки так называемой таблицы характеристик (ТХ).

Каждая такая строка представляет команду перехода к соответствующей подпрограмме, записанной на рабочем поле. Кроме того, ИП преобразует псевдокоманды основной программы, относящиеся к вызванной подпрограмме, в команды передачи управления соответствующей строке ТХ (команды засылки в некоторую фиксированную ячейку информации, необходимой для возврата в основную программу, содержащиеся в псевдокомандах, при этом сохраняются). Таким образом, наряду с интерпретацией псевдокоманд (вызывом отвечающих им подпрограмм) ИП производит включение вызванных подпрограмм в основную программу (компиляцию).

Если при вызове новой стандартной подпрограммы оказывается, что свободного места на рабочем поле для записи этой подпрограммы недостаточно, производится стирание. При этом в ТХ строки, отвечающие стертым подпрограммам, заменяются командами перехода к ИП, для того чтобы при обращении из основной программы к одной из стертых подпрограмм ИП снова произвела ее вызов. При повторном вызове подпрограммы ИП исправляет команду условного перехода, находящуюся в строке ТХ, отвечающей этой подпрограмме, заменяя ее командой перехода к этой подпрограмме в соответствии с ее новым местом в оперативной памяти.

§ 4. Метод программирующих программ

1. Появление программирующих программ. Появление программирующих программ (ПП) было связано с возникновением операторного метода программирования (см. гл. 4), предложенного проф. А. А. Ляпуновым в 1953 г. В 1954 г. в СССР появилась первая программирующая программа ПП-1, явившаяся как бы макетом для проверки новых идей [71]. В 1955 г. разработана уже более совершенная ПП-2 [32]. Успех ПП-2 послужил стимулом к созданию ряда других ПП (в 1955 г. — ПП для машины *Стрела*-2 [71], в 1956 г. — ПП *БЭСМ* [24], в 1957 г. — ПП-С [37], в 1958 г. — ПП для машины *Стрела*-4 [65] и т. д.). Перечисленные ПП являются универсальными компилирующими программирующими программами. Примерно в это же время был разработан ряд интерпретирующих программирующих программ, получивших название «прорабов» (см., например, [67]). Подтверждением связи между операторным методом и появлением ПП является и то обстоятельство, что в США с возникновением в 1954 г. алгебраического подхода к описанию программ, по существу совпадающего с операторным методом, также стал развиваться метод ПП. В 1956 г. в корпорации «Интернешенл бизнес мэшинз» была разработана мощная универсальная ПП *Фортран* для программирования на наиболее массовой из крупных американских электронных цифровых машин *ИМБ-704* [1]. В 1958 г. создана ПП *Юникод* для крупной вычислительной машины *Юнивак Сайнтифик 1103А*, во многом похожая на *Фортран*. В настоящее время метод ПП прочно вошел в практику программирования.

Логические схемы программ послужили исходным материалом для разработки входных языков, применяемых для задания исходной информации для ПП. Поскольку интерпретирующие ПП до настоящего времени не получили широкого распространения, в дальнейшем говорится главным образом о компилирующих ПП.

2. Стандартные и нестандартные операторы при методе ПП. При разработке ПП выбирают некоторое количество типов операторов, которые ПП должна составлять автоматически. Такие операторы называют *стандартными* (для дан-

ной ПП). Выбор типов стандартных операторов накладывает ограничения на характер программ, которые могут быть получены с помощью ПП. Эти ограничения связаны главным образом с тем, что при заданном наборе типов операторов программы некоторых задач становятся явно неудовлетворительными (например, очень «длинными» или требующими для своего выполнения чрезмерного расхода машинного времени). Для того чтобы снять эти ограничения и сделать метод ПП более гибким, в некоторых ПП предусматривают возможность включения в составляемые программы так называемых *нестандартных* операторов, составляемых вручную (вообще говоря, по специальным правилам, отличным от обычных правил программирования). Другой путь повышения гибкости метода ПП состоит в том, чтобы придать ПП структуру, допускающую расширение набора ее стандартных операторов без значительных переделок ПП.

Наборы стандартных типов операторов различны для различных ПП. Так ПП-С [37], составленная для машины *Стрела*, предусматривает в качестве стандартных операторов арифметические, логические, переадресации, восстановления, операторы «стандартные подпрограммы» и допускает нестандартные операторы. Составленная для машины *БЭСМ* ПП *БЭСМ* [24] предусматривает в качестве стандартных операторов арифметические, логические, стандартные циклы (обобщенные операторы) и допускает нестандартные операторы; циклы могут быть вложенными, т. е. содержащимися внутри других циклов. ПП *Фортран* предусматривает тридцать два типа стандартных операторов: оператор «арифметическая формула» (частный случай арифметического оператора), пятнадцать операторов управления (различные условные и безусловные переходы, останов, повторение группы операторов), тринадцать операторов переноса (различных видов ввода и вывода) и три оператора спецификации, которые собственно являются не операторами, а «блоками» информации, необходимой для обеспечения высокого качества составляемых программ.

3. Блочное строение программирующих программ. Кроме составления операторов, компилирующая ПП должна производить экономию рабочих ячеек, экономию команд в составляемых операторах, распределение памяти и присвое-

ние истинных адресов. Кроме того, весьма целесообразно, чтобы ПП начинала свою работу с проверки исходной информации для выявления в ней наиболее часто допускаемых формальных ошибок. Таким образом, ПП является весьма сложной программой. Для того чтобы ПП занимала небольшую часть оперативной памяти, ее целесообразно составить в виде отдельных блоков, хранящихся во внешних запоминающих устройствах и по мере надобности вызываемых в оперативную память. В противном случае либо значительно уменьшается возможная максимальная длина составляемых программ, либо усложняется организация работы ПП (приходится программируемую программу составлять по частям).

Для того чтобы расширение набора типов стандартных операторов не требовало переделки всей ПП в целом (т. е. по существу составления новой ПП), каждый из блоков должен автоматически составлять операторы одного определенного стандартного типа или выполнять вспомогательную работу одного определенного вида (экономию рабочих ячеек, присвоение истинных адресов и т. д.). Многократный вызов каждого блока ПП в оперативную память нецелесообразен (значительно увеличивает время работы ПП), поэтому исходной информации придают такой вид, а ПП — такую организацию, чтобы каждый блок ПП нужно было вызывать только один раз.

Состав операций, выполняемых ПП, обычно определяет ее структуру. Так, программирующая программа ПП-С состоит из следующих блоков:

K — контроль исходной информации,

$(AP)'$ — блок, подготовляющий информацию об арифметических и логических операторах,

A — блок арифметических операторов, составляющий, кроме того, ряд команд, входящих в логические операторы и осуществляющий экономию команд,

P — блок логических операторов,

$Э$ — блок экономии рабочих ячеек,

C — блок включения стандартных подпрограмм,

F — блок операторов переадресации,

O — блок операторов восстановления,

$П$ — блок распределения памяти и присвоения истинных адресов.

ПП-*БЭСМ* также состоит из ряда блоков:

I-ПП-1 — блок логических операторов,

II-ПП-1 — блок арифметических операторов,

III-ПП-1 — блок экономии рабочих ячеек,

I-ПП-2 — блок формирования цикла,

II-ПП-2 — блок построения команд управления циклом,

III-ПП-2 — блок расстановки команд управления в цикле,

I-ПП-3 — блок образования относительных адресов (в данном случае — номеров команд составляемого оператора, отсчитываемых от команд, в адресах которых они присутствуют),

II-ПП-3 — блок распределения памяти,

III-ПП-3 — блок присвоения истинных адресов,

IV-ПП-3 — блок печати информации.

Кроме того, в состав ПП-*БЭСМ* входят три так называемые ведущие программы, организующие работу блоков:

ВП-1 (для блоков I-ПП-1, II-ПП-1, III-ПП-1), ВП-2 (для блоков I-ПП-2, II-ПП-2, III-ПП-2) и ВП-3 (для блоков I-ПП-3, II-ПП-3, III-ПП-3, IV-ПП-3).

4. Программирование методом ПП. С помощью указанных выше ПП можно составить программу любой математической задачи. Для составления программ логических задач они могут оказаться мало пригодными, ибо логические задачи требуют весьма специфического и пока еще неполностью выявленного набора стандартных типов операторов. Работа программиста, эксплуатирующего универсальную компилирующую ПП, заключается в разработке логической схемы программы, записи ее на входном языке ПП и в проверке правильности полученной исходной для ПП информации. Кроме логической схемы для работы ПП может требоваться та или иная дополнительная информация, например о зависимости величин от параметров, о желательном распределении памяти и т. п. Эта информация по установленным для данной ПП правилам включается в исходную информацию. Если машина, на которой осуществляется автоматическое программирование, допускает только цифровой ввод, то исходную информацию, содержащую не только числа, но и буквы, приходится кодировать. Это значительно снижает выгоды от применения ПП.

Приготовленную исходную информацию вместе с ПП вводят в машину и после выполнения ПП получают програм-

мируемую программу в готовом виде. Возможность ошибок в исходной информации заставляет подвергать полученную программу отладке на машине. При использовании ПП весьма целесообразно отладку проводить автоматически с помощью отлаживающей программы. Очень удобной может оказаться декодирующая программа, получающая по отлаживаемой программе ее логическую схему из стандартных операторов (при условии, что нестандартных операторов в программе не было).

В настоящее время все большее и большее распространение получает автоматическое программирование (математических задач), основанное на применении алгорифмического языка АЛГОЛ, подробное описание которого приведено в гл. 8. Программирующие программы, применяемые при этом, обычно называют *трансляторами*, а процесс автоматического программирования — *трансляцией*. Все сказанное выше о методе ПП справедливо и для автоматического программирования с использованием АЛГОЛа.

§ 5. Перспективы развития метода программирующих программ

1. Универсальные ПП. Выше было дано краткое описание метода программирующих программ при использовании универсальных компилирующих ПП.

С помощью таких ПП экономия работы программистов (связанной с заданием решающего алгорифма на входном языке) и удовлетворительное качество автоматически составляемых программ достигаются лишь для некоторого класса задач (для которой входной язык ПП «удобен»). Улучшение качества программ решения других задач может быть достигнуто путем включения в программы ранее упомянутых нестандартных операторов, составляемых вручную. При этом увеличивается доля ручного труда и в некоторых случаях сводится почти к нулю выгода от автоматизации программирования.

Таким образом, каждая универсальная ПП практически оказывается далеко не универсальной.

По мере совершенствования применяемой ПП путем включения в нее новых блоков, которые составляют новые виды операторов (расширение числа стандартных типов операторов) и выполняют новые операции по улучшению состав-

ляемых программ (например, вместо экономии рабочих ячеек в отдельных операторах экономию их во всей составляемой программе), класс задач, для программирования которых пригодна данная универсальная ПП, будет расширяться.

Однако при таком методе полная автоматизация процесса программирования не достигается, ибо наиболее квалифицированная часть работы, в том числе разработка решающего алгорифма и составление логической схемы программы, в каждом случае применения ПП выполняется человеком. На машину возлагается менее квалифицированная формальная часть работы: переработка решающего алгорифма, заданного на входном языке, в программу.

Однако процесс автоматизации программирования принципиально можно довести до его логического конца — полного исключения при составлении отдельных программ этапов разработки решающего алгорифма и построения логической схемы программы. Точнее говоря, можно выделить сколь угодно обширный класс задач, не требующих для автоматического составления программ никакой предварительной подготовки. Исходной информацией для программирующей программы, применяемой для задач этого класса будет служить, например, их словесная формулировка, закодированная побуквенно [17]. Необходимо подчеркнуть, что программирующая программа не будет в этом случае универсальной. Такие программы могут быть составлены для ряда классов задач. Тем самым мы приходим к методу программирования, который может быть назван методом *библиотеки специализированных программирующих программ* (предложен А. А. Ляпуновым и В. М. Глушковым).

2. Метод библиотеки специализированных ПП. Сущность метода библиотеки специализированных программирующих программ заключается в следующем.

1. Выбирается несколько классов задач. При объединении задач в отдельный класс руководствуются не только сходством их формулировок, но и определенной общностью применяемых для них рещающих алгорифмов. Например, в один класс могут включаться задачи, предусматривающие решение методом Адамса систем дифференциальных уравнений первого порядка, разрешенных относительно производных, и т. п.

2. Для каждого класса задач составляется специализированная программирующая программа, требующая минимума исходной информации, заданной в общепонятной естественной форме и не учитывающей специфику применяемой машины.

3. Каждая специализированная ПП должна предусматривать определенную организацию составляемых ею программ, например определенный способ контроля правильности вычислений и т. д.

4. Отдельные специализированные ПП могут содержать некоторые общие блоки, например арифметический блок и т. п. Такие блоки естественно хранить в одном экземпляре во внешних запоминающих устройствах и в нужное время вызывать в оперативную память.

5. Для задач, не принадлежащих ни одному из классов, которым соответствуют специализированные ПП, следует применять либо программирование вручную, либо универсальную программирующую программу.

На первом этапе можно ограничиться библиотекой специализированных программирующих программ. После того как библиотека программирующих программ сделается достаточно обширной, ее можно разбить на отделы и построить специальную *программу выбора метода.* Такая программа должна по закодированной словесной формулировке задачи и присутствующим в ней формулам определять сначала отдел библиотеки, а затем номер специализированной программирующей программы.

По-видимому, наиболее эффективно может быть использована библиотека специализированных ПП, снабженная программой выбора метода, на машине, располагающей достаточно большим объемом запоминающих устройств для размещения в них всей библиотеки.

Заметим, однако, что специализированные ПП оказываются значительно более короткими, чем универсальные, что́ позволяет применять их (без объединения в комплекс) на машинах со сравнительно малым объемом запоминающих устройств.

Применение библиотеки специализированных ПП не освобождает программистов от работы. Вместо того чтобы тратить усилия на составление большого количества программ для решения отдельных задач программисты должны постоянно работать над расширением библиотеки специализированных ПП.

А. А. Ляпунов выдвинул идею оригинальной организации библиотеки специализированных ПП, заключающуюся в следующем. Библиотека должна содержать большое количество блоков, объединенных в упорядоченную систему групп. В каждую группу должны входить функционально родственные блоки (например, в одну группу — арифметические блоки, в другую группу — логические блоки и т. д.). Специализированные ПП получаются путем выбора по одному блоку из каждой группы и объединения их в порядке групп. Такая организация библиотеки, по-видимому, облегчит работу по ее постоянному расширению.

Работы в области автоматизации программирования, выполненные за последние несколько лет, по-видимому, являются началом движения по описанному выше пути. Разрабатываются специализированные ПП для программирования отдельных классов задач. Для этих ПП разрабатываются свои, приспособленные для соответствующих классов задач входные алгорифмические языки. Необходимо отметить тенденцию разрабатывать входные языки, не отражающие специфики вычислительных машин. Это позволяет в области разработки решающих алгорифмов объединять усилия всех специалистов, разрабатывающих методы решения задач данного класса. Становится реальной возможность (даже в международных масштабах) накапливать решающие алгорифмы. Последнее обстоятельство повышает интерес к автоматизации программирования.

В настоящее время изготовители программно-управляемых машин придают каждой поставляемой ими машине определенный набор стандартных подпрограмм, отладочных программ, программирующих программ (трансляторов) и инструкции для их применения. Совокупность этих программ и инструкций называется *математическим обеспечением* машины. Удобство пользования машиной в значительной мере зависит от математического обеспечения.

Литература к главе 7: [1], [14], [16], [17], [22], [24], [25], [27], [32], [37], [42], [52], [56], [60], [61], [65], [66], [67], [71], [73].

ГЛАВА 8

АЛГОРИФМИЧЕСКИЕ ЯЗЫКИ

§ 1. Язык логических схем

В гл. 5 при описании равносильных преобразований логических схем фактически изложен алгорифмический язык, который называется языком логических схем (коротко ЯЛС). Ниже приводится синтаксис этого языка.

ЯЛС разрабатывался, с одной стороны, как язык, удобный для равносильных преобразований алгорифмов, с другой стороны, как язык, предназначенный не только для нужд программирования, но и для описания дискретных процессов преобразования, имеющих различную природу. Следствием этого явилась большая общность ЯЛС как средства описания процессов; наличие в языке специальных приемов укрупненной записи (что необходимо как при описании реальных дискретных процессов преобразования, так и при равносильных преобразованиях алгорифмов, в ходе которых запись алгорифма приходится многократно переписывать); логическая простота ЯЛС, необходимая для упрощения правил равносильных преобразований.

Неизбежным следствием большой общности является неполная формализация ЯЛС. При различных конкретных применениях ЯЛС необходимо производить дополнительную его формализацию, заключающуюся в том, что понятию состояний ячеек приписывается определенный смысл (чего нет в самом ЯЛС), после чего приписывается точный смысл применяемым функциям и операциям (под операцией здесь понимается получение по значениям аргументов значения функции).

1. Основные символы алгорифмического языка ЯЛС. Основными символами языка логических схем являются:

1° общепринятые арабские десятичные цифры;

2° латинские курсивные строчные буквы и греческая буква φ;

3° латинские заглавные буквы D, V, P и русские заглавные буквы *И*, *Ф*, *Я*;

4° знаки (общепринятые) математических действий и функций;

5° знаки логических связей

$$\wedge\ (и),\ \vee\ (или),\ \rightarrow (влечет),\ \overline{\ }\ (не),$$

$$\sim (равнозначно),\ \simeq (неравнозначно);$$

6° знаки математических отношений

$$>,\ \geqslant,\ <,\ \leqslant,\ =,\ \neq;$$

7° знак присваивания $:=$;

8° знаки перехода левые $\llcorner$, Γ, $\sqsubset$ и правый $\lrcorner$;

9° скобки ( ).

Допускается расширение наборов всех видов знаков, кроме 7°, 8°. При записях на ЯЛС употребляются два шрифта: крупный и мелкий.

2. Ячейки. Понятие *ячейки* в ЯЛС (см. гл. 5, п. 2) является фундаментальным. Каждой ячейке всегда соответствуют некоторое ее *состояние* (в котором она находится) и некоторое множество возможных для нее состояний. Что представляют собой состояния ячеек, в ЯЛС не определяется; это делается при различных конкретных применениях ЯЛС. Каждая ячейка устойчива в своем состоянии, которое может меняться только в результате особого действия.

С синтаксической точки зрения ячейки являются одним из видов элементарных конструкций языка логических схем. Если ξ обозначает любую букву из упомянутых в п. 1, 2° кроме p, f, φ, o, а $\alpha_1, \alpha_2, \ldots, \alpha_\varkappa$ и $\beta_1, \beta_2, \ldots, \beta_\lambda$ — целые неотрицательные числа (для которых принят общеизвестный способ записи в виде последовательности цифр), то записи

$$\xi,\ \xi_{\alpha_1, \alpha_2, \ldots, \alpha_\varkappa},\ \xi^{(\beta_1, \beta_2, \ldots, \beta_\lambda)},\ \xi^{(\beta_1, \beta_2, \ldots, \beta_\lambda)}_{\alpha_1, \alpha_2, \ldots, \alpha_\varkappa}$$

называются *ячейками*.

Примерами ячеек могут служить x, y_1, y_2, $z^{(1,\ 5)}_{3,\ 4,\ 6}$, $u^{(1,\ 7)}$.

3. Знаки операций. Функции. Кроме знаков математических действий и функций, упомянутых в п. 1, 4° (о них будет сказано ниже), в ЯЛС употребляется неформализованные знаки действий, представляющие собой записи вида

$$f,\ \varphi,\ f_{\alpha_1, \alpha_2, \ldots, \alpha_\chi},\ \varphi_{\alpha_1, \alpha_2, \ldots, \alpha_\chi},\ p_{\alpha_1, \alpha_2, \ldots, \alpha_\chi}.$$

Эти знаки действий подлежат точному определению в каждом конкретном случае применения ЯЛС. Они определяют действия над состояниями ячеек и могут быть заданы только тогда, когда известно, что такое состояния ячеек. Некоторые из неформализованных знаков действий обозначают операции над объектами, определенными в ЯЛС, однако остаются неформализованными потому, что заранее неизвестно, какие знаки операций могут потребоваться при конкретном применении ЯЛС, и таким образом представляют собой резерв для расширения набора знаков операций. При формализации эти знаки операций могут быть заменены другими.

Пусть символы $\zeta_1, \zeta_2, \ldots$ означают произвольные ячейки, а θ — произвольный неформализованный знак операции. Тогда записи вида

$$\theta(\zeta_1, \zeta_2, \ldots, \zeta_\rho), \qquad \theta_{\alpha_1, \alpha_2, \ldots, \alpha_\chi}(\zeta_1, \zeta_2, \ldots, \zeta_\rho)$$

называются *функциями*.

В ЯЛС считается, что операндами (исходными данными для операций и аргументами функций, а также результатами операций и значениями функций) являются состояния ячеек, хотя в записях указаны сами ячейки. Неформализованным функциям и знакам операций может быть придан точный смысл лишь при условии, что будет определен смысл термина «состояния ячеек». В частном случае состояниями ячеек являются числа, как это имеет место при применении ЯЛС для нужд программирования. При этом знаки операций и функции приобретают смысл, принятый для них в математике или заданный в машинном языке. Обычно (но не обязательно) в этом случае применяются знаки операций и функций, упомянутые в п. 1, 4°.

В ЯЛС предполагается, что суперпозиции функций (при определенных условиях) также являются функциями, которые могут быть обозначены с помощью еще не использованных неформализованных знаков операций.

4. Классификация ячеек. Совокупность всех ячеек, используемых в данном применении ЯЛС, называется *памятью*. В ЯЛС предполагается, что память состоит из трех частей, называемых соответственно *памятью объектов* (в гл. 5 мы ее называли памятью величин, что является частным случаем), *памятью параметров* и *логической памятью* (в гл. 5 была названа памятью сигналов). Последняя состоит из единственной, так называемой *логической ячейки*, для которой существуют два возможных состояния, обозначаемых соответственно цифрами 0 и 1, первая из которых означает логическое значение «ложь», а вторая — логическое значение «истина». Логическая ячейка обозначается буквой *о*. Ячейки, входящие в память объектов, называются *объектами*, а входящие в память параметров — *параметрами*.

5. Параметры. Параметры записываются в виде букв i, j без индексов или с нижними индексами. Состояниями этих ячеек являются натуральные (целые неотрицательные) числа. Чтобы не отказываться от привычных терминов, будем состояния параметров называть их *значениями*.

Если значение какого-либо параметра является тождественно постоянным, то для обозначения этого параметра может применяться его значение (т. е. целое число). Таким образом, примерами параметров могут служить i, j, i_0, $i_{1,\,5}$, $j_{2,\,3,\,7}$, 1, 54.

Считается, что над значениями параметров возможно выполнение любых операций, приводящих к получению целого неотрицательного результата, т. е. к получению нового значения параметра. Так как не все мыслимые операции такого характера имеют принятые в математике обозначения, то для обозначения операций над параметрами могут применяться и неформализованные знаки операций, а именно буква φ с индексами или без них. Примерами операций над параметрами могут служить

$$i+1, \quad i+j,$$

$$\varphi_5(i_1, i_2, i_3), \quad E\left(\frac{i+j}{2}\right), \quad \left|\sin\frac{\pi i}{2}\right| \text{ и т. п.}$$

Функции от состояний параметров, значениями которых являются состояния параметров (объектов), будем называть *функциями первого (второго) рода от параметров*.

Функции от параметров, значениями которых являются 0 и 1 (т. е. «ложь» и «истина»), называются *предикатами от параметров*. К числу таких функций прежде всего относятся соотношения, каждое из которых представляет собой пару функций математического характера, соединенных знаком отношения (п. 1, 6°). В частности, в соотношения могут входить отдельные параметры и числа. Примерами соотношений являются

$$i \geqslant 2, \quad i-j<4, \quad i_1+j_1 \neq i_2+j_2 \quad \text{и т. п.}$$

Для тех значений параметров, для которых соотношение выполнено, оно считается равным единице; для тех, для которых обе входящие в него функции определены, но соотношение не выполнено, оно считается равным нулю; для остальных значений параметров оно считается неопределенным.

Неформализованные предикаты от параметров обозначаются как функции с помощью буквы p с индексами или без индексов.

Наконец, с помощью знаков логических связей по общеизвестным правилам алгебры логики (гл. 1, § 3) из предикатов составляются более сложные предикаты. Впрочем, любой сложный предикат можно обозначить при помощи какого-нибудь из еще не использованных неформализованных предикатов.

Пример 8.1. Предикатами от параметров являются функции

$$i>j; \quad p_1^{(1)}(i_1, i_2, i_3); \quad p_1^{(2)}(i_1, i_3, i_4).$$

Из этих предикатов можно построить, например, сложный предикат

$$i>j \wedge p_1^{(1)}(i_1, i_2, i_3) \sim \overline{p_1^{(2)}(i_1, i_3, i_4)}.$$

Этот сложный предикат можно обозначить, например, с помощью не использованного нами еще неформализованного предиката $p(i, j, i_1, i_2, i_3, i_4)$.

6. Объекты. Объекты обозначаются с помощью любых букв, перечисленных в п. 3, 2°, кроме букв p, f, φ, o i, j.

Состояния объектов в ЯЛС не определены и подлежат определению при конкретном применении ЯЛС. При программировании считается (при составлении логической схемы), что объектами являются величины, а их состояниями — числа (значения величин). Это позволяет в качестве операций над

состояниями объектов взять математические операции и функции. На втором этапе, при переходе от логических схем к программам, считают, что объектами являются ячейки памяти программно-управляемой машины, а их состояниями — содержимое ячеек. Операции истолковываются как машинные операции, соответствующие математическим операциям.

Чтобы подчеркнуть общность языка логических схем, заметим, что он может быть использован для описания процессов, не имеющих математического характера. Например, на ЯЛС можно описать производственный процесс, хотя бы процесс сборки из отдельных деталей автомобильных двигателей. При этом состояниями объектов будут отдельные детали и соединения из этих деталей. Операциями будут операции (тоже производственные), с помощью которых производится сборка.

Способ задания операций в ЯЛС не установлен. Этот способ зависит от природы состояний объектов. Иногда операции могут задаваться математическими средствами, в других случаях — с помощью словесных описаний или рисунков. Если бы мы захотели описать с помощью ЯЛС процесс сборки автомобильных двигателей, то, возможно, операции нам пришлось бы задавать в виде кинофильмов.

Функции от состояний объектов, значениями которых являются состояния объектов (параметров), будем называть *функциями первого (второго) рода от объектов.*

Предикаты, аргументами которых являются состояния объектов, называются *предикатами от объектов* и обозначаются с помощью буквы p с индексами или без них.

Предикаты, аргументами которых являются как объекты, так и параметры, образуются из предикатов от объектов и предикатов от параметров с помощью знаков логических связей по правилам алгебры логики.

Пример 8.2. Предикатами от объектов являются

$$p(x_1, x_2); \qquad p_3(a_1, b_2^{(1)}, a_{3,4}^{(2,7)}).$$

В качестве примера предиката от параметров и объектов приведем

$$p(x_1, x_2) \wedge i^2 + i > j^3 \vee \overline{p_2(x_1) \to i \leqslant j}.$$

Этот предикат построен из более простых предикатов «чистого» вида:

$$p(x_1, x_2); \qquad i^2 + i > j^3, \qquad p_2(x_1), \qquad i \leqslant j.$$

7. Ячейки и функции, зависящие от параметров. Символы, которые получатся, если в записях ячеек и функций вместо числовых индексов подставить функции первого рода от параметров, называются *ячейками, зависящими от параметров, и функциями, зависящими от параметров.*

Заметим, что ячейки и функции, зависящие от параметров, не являются ячейками и функциями, а являются лишь конструкциями в ЯЛС. Поэтому наше определение не позволяет нам говорить о ячейках и функциях, зависящих от «параметров, зависящих от параметров». В ЯЛС принято, что, придавая состояниям параметров, входящих в индексы, конкретные значения, вычисляя функции первого рода, стоящие в индексах, и заменяя их полученными натуральными числами, мы получим (обычные) ячейки и функции. Эти ячейки и функции называются конкретизациями ячеек и функций, зависящих от параметров. Таким образом, ячейки и функции, зависящие от параметров, можно считать функциями, аргументы которых суть значения параметров, а значения которых (их конкретизации) — соответственно ячейки и функции.

В дальнейшем под ячейками и функциями мы будем понимать как ячейки и функции, не зависящие от параметров, так и зависящие от них. Правила записи, изложенные выше для функций, не зависящих от параметров, с состояниями ячеек, не зависящих от параметров, в качестве аргументов, мы распространяем на функции и ячейки, зависящие от параметров. В ЯЛС считается, что для вычисления функции, зависящей от параметров, необходимо сначала получить конкретизации функции и ячеек, входящих в ее запись, и лишь потом вычислить значение конкретизации функции, которое и принимается за значение функции, зависящей от параметров (при определенных значениях параметров и определенных состояниях ячеек-аргументов).

Пример 8.3. От параметров могут зависеть объекты и параметры, но не логическая ячейка (которая в своем обозначении не содержит числовых индексов). Ячейками, зависящими от параметров, являются x_i, $y_i^{(j)}$, a_{i_1, i_2}, $b_{\varphi(i_1, i_2)}$, i_j, j_i. Функциями, зависящими от параметров, являются $p_j(x, y)$, $f_{i+j}(x_1, y^{(2)})$, $\varphi^{(j)}(i_5)$. Функции $f_1^{[2]}(x_i)$ и $p_2(u_i, v_{i+j})$ не являются зависящими от параметров, хотя их аргументами и являются состояния ячеек, зависящих от параметров.

Конкретизациями вышеприведенных ячеек и функций при $i=0$, $j=1$, $i_1=i_2=3$ и при $\varphi(0, 1)=10$ будут ячейки x_0, $y_0^{[1]}$,

$a_{3,8}$, b_{10}, i_1, j_0 и функции $p_1(x, y)$, $f_1(x, y^{(2)})$, $\varphi^{(1)}_{(i_5)}$), $f^{(2)}_1(x_0)$ и $p_2(u_0, v_1)$.

Отметим еще более «трудный» случай: конкретизацией функции $i_{2j+1}+3j$ при $j=2$ будет функция i_5+6.

8. Комплексы. Запись вида

$$\zeta: =\theta(\zeta_1, \zeta_2, \ldots, \zeta_k),$$

где ζ, ζ_1, ζ_2, ..., ζ_k — ячейки, а θ — функция, называется *формулой*.

Последовательность, элементами которой являются:

а) перечень X ячеек, называемый *входным кортежем*;

б) конечное число формул вида

$$\zeta_{s_i}: =\theta_i(\zeta_1, \zeta_2, \ldots, \zeta_n), \qquad 1 \leqslant s_i \leqslant n, \quad i=1, 2, \ldots, N;$$

в) перечень Y ячеек, называемый *выходным кортежем*, называется *комплексом* *), если выполнены следующие условия:

1° Каждая ячейка, из числа действительно присутствующих **) в правой части формулы $\zeta_{s_i}=\theta_i(\zeta_1, \zeta_2, \ldots, \zeta_n)$, либо входит в X, либо присутствует среди ячеек $\zeta_{s_1}, \zeta_{s_2}, \ldots, \zeta_{s_{i-1}}$.

2° Каждая ячейка из Y либо входит в X, либо присутствует хотя бы один раз в последовательности $\zeta_{s_1}, \zeta_{s_2}, \ldots, \zeta_{s_N}$.

3° Если ячейка ζ_{s_i} является параметром, то для $i+1$, $i+2$, ..., N формулы не содержат функций и ячеек, зависящих от этого параметра.

Примеры комплексов приведены в гл. 5, § 1; нужно только помнить, что в гл. 5 применяется частный вид языка логических схем (состояниями объектов считаются числа). В частном случае последовательность формул в комплексе может сводиться к одной формуле или даже быть пустой.

9. Операторы. Их классификация. Комплекс, все ячейки которого являются объектами, а функции — функциями первого рода, называется *действующим оператором*. Если

---

*) В гл. 5 такая последовательность называлась нормальной последовательностью комплекса.

**) Согласно терминологии гл. 5 являющаяся нефиктивным аргументом.

некоторые ячейки или функции этого действующего оператора зависят от параметров, то он называется зависящим от параметров. Действующие операторы обозначаются заглавной латинской буквой *D*, снабженной снизу (а может быть и сверху) числовыми индексами (неотрицательными целыми). Действующий оператор, зависящий от параметров, вместо всех или нескольких верхних числовых индексов имеет функции первого рода от параметров. Действующий оператор, который получится, если все ячейки и функции, зависящие от параметров, которые содержатся в нем, заменить их конкретизациями, соответствующими определенной системе значений параметров, называется конкретизацией действующего оператора (зависящего от параметров). Примеры записей действующих операторов можно найти в гл. 5, §§ 2 и 3.

Комплекс, все ячейки которого являются параметрами, а функции — функциями первого рода, называется *варьирующим оператором*. Как и действующие операторы, варьирующие операторы могут быть зависящими от параметров. Обозначения варьирующих операторов строятся аналогично обозначениям действующих операторов, с той лишь разницей, что при обозначении оператора вместо буквы *D* употребляется буква *V*. В гл. 5 рассматривались только варьирующие операторы, не зависящие от параметров.

Комплекс, содержащий только одну формулу, функция в которой является предикатом, и имеющий в качестве выходного кортежа одну логическую ячейку *о*, называется *логическим оператором*. Как и все вышеописанные операторы, логические операторы могут быть зависящими от параметров. Обозначения для логических операторов строятся по тем же правилам, что и для действующих, только вместо буквы *D* применяется буква *P*. Так как во всех логических операторах формула начинается одинаково с пары знаков «*о* :=», то в ЯЛЄ принято соглашение эту пару знаков опускать. Таким образом, вместо формулы в записи логического оператора может присутствовать предикат.

Комплекс, все функции которого являются функциями от объектов, выходной кортеж состоит из параметров, а все ячейки, не входящие в выходной кортеж, являются объектами, называется *оператором формирования*. Такие операторы тоже могут зависеть от параметров. Обозначения для них строятся по правилам, одинаковым с правилами для действующих опе-

раторов, только вместо буквы D применяется буква Ф. В гл. 5 такие операторы не рассматривались.

Комплекс, все функции которого являются функциями от параметров, выходной кортеж состоит из объектов, а все ячейки, не входящие в состав выходного кортежа, являются параметрами, называется *оператором вызова* (объектов, в частном случае объекта). Обозначения операторов вызова строятся по общему правилу с применением буквы F. Операторы вызова, как и все другие, могут быть зависящими от параметров. В гл. 5 операторы вызова не рассматривались.

Выполнением оператора называется следующий процесс. Если оператор зависит от параметров, то строится его конкретизация для определенной системы значений параметров (таким образом, оператор, зависящий от параметров, может быть выполнен только при определенных значениях параметров, от которых он зависит). Оператор, не зависящий от параметров, можно при этом считать зависящим от параметров, но таким, что все его конкретизации между собой тождественны. Далее производится преобразование памяти путем последовательного выполнения формул полученной конкретизации. Каждое выполнение формулы заключается в том, что определяется значение функции, стоящей в ее правой части, и это значение в качестве нового состояния присваивается ячейке, стоящей слева от знака :=.

Кроме вышеописанных видов операторов в ЯЛС называются еще операторами знаки вида $И_0$ и $Я_\nu$, где ν — целое положительное число. Знак $И_0$ называется оператором начала, а любой знак $Я_\nu$ — оператором конца (смысл этих операторов известен читателям из гл. 5).

10. Знаки перехода. *Знаки перехода* делятся на левые и правые. *Левыми знаками перехода* называются символы вида $\underset{\nu}{\lfloor}$ и $\overset{\nu}{\lceil}$, где ν — либо целое положительное число, либо функция первого рода от параметров, принимающая целые положительные значения. В последнем случае левый знак перехода называется зависящим от параметров. Если вычислить для заданной системы значений параметров значение функции и вписать его на место функции, получится левый знак перехода (не зависящий от параметров), который является кон-

кретизацией левого знака перехода, зависящего от параметров. Левые знаки перехода делятся на верхние (вида $\overset{\nu}{\Gamma}$) и нижние (вида $\underset{\nu}{\llcorner}$).

Правые знаки перехода имеют только один вид $\underset{\nu}{\lrcorner}$, где ν — целое положительное число.

Для пары левых знаков перехода $\underset{\nu_1}{\llcorner}$, $\overset{\nu_2}{\Gamma}$, один из которых нижний, а другой верхний, введено специальное обозначение $\overset{\nu_2}{\underset{\nu_1}{\sqsubset}}$. Такой знак мы будем ниже называть *двойным* (левым) *знаком перехода.*

11. Логическая схема *Логической схемой* называется конечная строка, состоящая из операторов и знаков перехода, для которой выполнены следующие условия:

1° На первом (самом левом) месте в логической схеме стоит оператор $И_0$, который нигде более в логической схеме не встречается.

2° Индексы всех правых знаков перехода между собой различны.

3° Левый нижний знак перехода может стоять после любого оператора, кроме оператора $Я_\nu$, и только после оператора.

4° Левый верхний знак перехода может стоять только после логического оператора или левого нижнего знака перехода, следующего за логическим оператором, и более нигде.

5° Правый знак перехода может стоять в логической схеме перед любым оператором, кроме $И_0$.

6° В логической схеме не может быть левых знаков перехода с числовыми индексами, отличными от всех числовых индексов правых знаков перехода, а также не может быть левых знаков перехода, зависящих от параметров, области изменения функций у которых содержат числа, не являющиеся индексами правых знаков перехода.

Приведенное здесь определение логической схемы является более широким, чем данное в гл. 5. Таким образом, правила равносильных преобразований, изложенные в гл. 5, рассчитаны только на некоторый класс логических схем, который, однако, достаточно широк и охватывает все наиболее важные случаи, встречающиеся на практике.

Всякая логическая схема рассматривается вместе с некоторым набором параметров, имеющих определенные (так называемые *начальные* значения).

Логические схемы образуют некоторый класс алгорифмов. Семантику ЯЛС определяет алгорифм выполнения логических схем. В общем случае этот алгорифм отличается от приведенного в гл. 5, § 2.

Пусть α — состояние памяти объектов, β — состояние памяти параметров, ω — состояние логической памяти. Если ρ_ν — конкретный вид одного из этих состояний и если $\rho_{\nu+1}$ — конкретный вид того же самого или другого из этих состояний, устанавливающийся благодаря выполнению оператора Z, то будем писать

$$\rho_{\nu+1} = Z(\rho_\nu).$$

Условимся, кроме того, называть элементами логической схемы операторы; правые знаки перехода; двойные левые знаки перехода (имея в виду, что это пара левых знаков перехода); пары левых знаков перехода, следующих за логическим оператором; отдельные левые нижние знаки перехода, стоящие после нелогических операторов.

Алгорифм выполнения логических схем

1° Полагаем $\omega_0 = 0$, $\alpha_0 = \alpha$, $\beta_0 = \beta$, $k = 0$. Находим элемент $И_0$ логической схемы и приступаем к выполнению п. 3°.

2° Если рассматриваемый элемент является правым знаком перехода, приступаем к выполнению следующего пункта, иначе переходим к п. 4°.

3° Переходим в логической схеме к ближайшему элементу справа. Возвращаемся к п. 2°.

4° Если рассматриваемый элемент является оператором, то переходим к п. 5°, иначе к п. 7°.

5° Если рассматриваемый элемент является оператором $Я_\nu$, то процесс окончен, иначе переходим к п. 6°.

6° Определяем конкретизацию Z рассматриваемого оператора (обозначим ее соответственно через D, V, P, Ф или F, в зависимости от того, является ли она оператором действующим, варьирующим, логическим, формирования или вызова).

Полагаем

$$\omega_{k+1} = \begin{cases} P(\alpha_k, \beta_k), & \text{если } Z = P, \\ \omega_k & \text{в остальных случаях;} \end{cases}$$

$$\alpha_{k+1} = \begin{cases} D(\alpha_k), & \text{если } Z = D, \\ F(\beta_k), & \text{если } Z = F, \\ \alpha_k & \text{в остальных случаях;} \end{cases}$$

$$\beta_{k+1} = \begin{cases} V(\beta_k), & \text{если } Z = V, \\ \Phi(\alpha_k), & \text{если } Z = \Phi, \\ \beta_k & \text{в остальных случаях.} \end{cases}$$

Увеличиваем значение k на 1. Возвращаемся к п. 3°.

7° Определяем конкретизацию рассматриваемого левого знака перехода (в том числе двойного или пары левых знаков перехода). Если полученная конкретизация является двойным знаком перехода или парой знаков перехода, идем к п. 8°, иначе — к п. 9°.

8° Если $\omega = 1$, рассматриваем верхний левый знак перехода, а если $\omega = 0$, то нижний левый знак перехода. Приступаем к выполнению п. 9°.

9° Находим в логической схеме правый знак перехода, индекс которого равен индексу рассматриваемого левого знака перехода. Возвращаемся к п. 3°.

Язык логических схем имеет три уровня: *отвлеченный, смешанный* и *конкретный.*

На отвлеченном уровне алгорифмы записываются в виде логических схем, в которых операторы обозначены буквенными символами, снабженными целочисленными нижними индексами и при зависимости их от параметров последовательностями верхних буквенных индексов. Например, в гл. 5 мы встречали $И_0$, $D_1^{i,j}$, V_5, P_7^j, $Я_\nu$ и т. п., в настоящей главе, кроме того, Ф и F, а в гл. 6 — $A_5^{i,j}$, P_7, $F_3(i)$, $O_4(j)$, $П_0$, $Я_{15}$ и т. д. Описания операторов, например, в виде комплексов и логических формул должны быть заданы отдельно. Отвлеченный уровень удобен для описания структуры алгорифма (логической схемы или программы) и для равносильных преобразований, не затрагивающих внутреннее содержание операторов.

При конкретном уровне все операторы логических схем представлены как комплексы, т. е. последовательностями фор-

мул или логическими формулами. Такая запись логических схем применяется при подробном описании алгорифмов. Логические схемы программ как исходная информация для автоматического программирования, кроме операторов, должны включать в себя так называемые *описания*, содержащие дополнительную информацию, необходимую для программирования. Такой информацией могут быть сведения о размерности матриц и способе развертывания их в «линию», о взаимном размещении величин в оперативной памяти, об ожидаемой частоте получения логическими операторами значений предикатов 1 и 0 и т. п.

При смешанном уровне часть операторов обозначена символами, другая часть представлена в виде комплексов и логических формул. Смешанный уровень удобен при равносильных преобразованиях логических схем, затрагивающих как их структуру, так и внутреннее содержание операторов.

Нужно еще заметить, что логические схемы можно составлять не только из элементарных, но и из обобщенных операторов, вводя для последних сокращенные способы записи. Например, операторами могут быть стандартные циклы. Такие операторы предусматриваются в качестве стандартных для ПП-*БЭСМ*. Запись таких операторов на отвлеченном уровне может быть $[D_5^j]_{j=1}^n$, где D_5^j — действующий оператор, зависящий от параметра j (этот оператор должен быть выполнен для всех последовательных значений $j=1, 2, \ldots, n$). Запись вложенных циклов может иметь вид $[D_1[D_2^{i,j}]_{i=1}^n D_3^j]_{j=1}^m$. Цикл, который должен снова и снова выполняться до тех пор, пока не окажется выполненным некоторое условие, например $x > y$, можно записывать в виде $[D_1]_{x>y}$.

Вообще говоря, *обобщенным оператором* называется строка операторов и знаков перехода, являющаяся частью логической схемы, начинающаяся оператором отличным как от $И_0$, так и от $Я_v$, не содержащая правых знаков перехода с индексами, которые равны или могут быть значениями индексов левых знаков перехода, стоящих вне этой строки. Если мы желаем, применяя обобщенные операторы, пользоваться символикой, разработанной в ЯЛС, то нужно еще потребовать, чтобы обобщенный оператор заканчивался обязательно оператором и чтобы все входящие в него левые знаки перехода были внутренними, т. е. не приводили к переходу к

операторам, не принадлежащим обобщенному оператору. Обобщенные операторы, входящие в логическую схему и обозначенные буквами, должны быть пояснены в приложении к логической схеме.

§ 2. Алгорифмический язык АЛГОЛ

В 1960 г. группой ученых различных стран был выработан алгорифмический язык АЛГОЛ 60. По замыслу авторов этот язык должен стать международным входным языком автоматического программирования математических задач и средством облегчения обмена алгорифмами. По своему характеру этот язык приспособлен к описанию математических вычислений и мало пригоден, например, для описания процессов переработки экономической информации, автоматического управления и т. п.

При разработке АЛГОЛа 60 учитывались следующие три основных требования:

а) получить язык, по возможности близкий к общепринятому языку математических описаний;

б) обеспечить возможность задания на этом языке алгорифмов, решения вычислительных задач;

в) обеспечить удобство автоматического перевода алгорифмов заданных на этом языке, в программы для конкретных вычислительных машин.

В соответствии с этими требованиями были выработаны три рода вариантов языка АЛГОЛ.

Один из вариантов объявлен *эталонным языком.* Он содержит фиксированный набор основных символов и фактически только видом этих символов отличается от других вариантов.

Группа вариантов называется *языками публикаций.* В языках публикаций разрешается пользоваться любыми знаками в качестве основных символов при условии, что эти знаки облегчают применение языка. При этом должно быть соблюдено четкое соответствие с эталонным языком, заключающееся в том, что точно установлено, чему в эталонном языке соответствует каждый знак или прием записи. Между всеми основными символами, кроме букв *), эталонного языка и языка

*) Смысл термина «буква» здесь не совпадает со смыслом этого термина в теории алгорифмов и будет определен ниже.

публикаций должно существовать взаимно однозначное соответствие. Что же касается букв языка публикаций, то достаточно, чтобы было известно, какие именно знаки являются его буквами.

Наконец, еще группа вариантов языка АЛГОЛ называется *конкретными представлениями*. Эти варианты отличаются от эталонного языка и языков публикаций тем, что основным символам эталонного языка в конкретном представлении соответствуют знаки или комбинации знаков, принятые в этом конкретном представлении и являющиеся знаками входного алфавита некоторой конкретной цифровой программно-управляемой машины.

Ряд основных символов эталонного языка представляет собой слова английского языка. Стремясь к тому, чтобы алгорифмический язык был возможно близок к принятому в нашей стране языку математических описаний, мы заменяем их соответствующими русскими словами. Таким образом, ниже приводится краткое описание одного из языков публикации. Каждый раз, когда описываемый основной символ будет отличаться от принятого в эталонном языке, это будет особо отмечаться.

Приводя примеры записей на языке АЛГОЛ, мы лишены возможности пользоваться знаками препинания, так как последние входят в число основных символов самого языка АЛГОЛ. В связи с этим при необходимости вместо знаков препинания (точек и запятых) для отделения записей на АЛГОЛе друг от друга мы будем применять вертикальную черту | , не являющуюся основным символом АЛГОЛа.

1. Основные символы алгорифмического языка АЛГОЛ. *Основными символами* языка АЛГОЛ называются:

1) общепринятые арабские десятичные *цифры*;

2) строчные и заглавные курсивные *буквы* латинского и русского алфавитов (в эталонном языке только латинского алфавита);

3) *логические значения*;

4) *ограничители*.

Логическими значениями называются символы **истина** и **ложь** (в эталонном языке соответственно **true** и **false**). Ограничители (они перечислены в табл. 8.1 и 8.2) подразделяются на:

а) *знаки операций*,

б) *разделители*,

в) *описатели* [44].

Таблица 8.1. Знаки операций

Арифметические операции		Логические операции		Операции отношения	
символ	как читается	символ	как читается	символ	как читается
$+$	плюс	$\vee$	или	$<$	меньше
$-$	минус	$\wedge$	и	$\leqslant$	не больше
$\times$	умножить	$\rightarrow$	влечет (импликация)	$>$	больше
$/$	разделить на	$\sim$	равнозначно	$\geqslant$	не меньше
$\div$	разделить в целом			$=$	равно
$\uparrow$	возвести в степень			$\neq$	не равно

В эталонном языке вместо знаков $\rightarrow$, $\sim$, $\leqslant$, $\geqslant$, $\neq$, приведенных в табл. 8.1, приняты соответственно знаки $\supset$, $\equiv$, $\leqq$, $\geqq$, $\neq$. Операция «разделить в целом» на общепринятом математическом языке определяется как вычисление целой части абсолютного значения частного, взятой со знаком частного, т. е.

$$a \div b = \operatorname{sign} \frac{a}{b} E\left|\frac{a}{b}\right|.$$

Операторные скобки **начало** и **конец** (см. табл. 8.2) в эталонном языке изображаются соответственно символами **begin** и **end**, а вместо кавычек » и « применяются символы ' и ,.

2. Числа. Любая последовательность цифр называется *целым числом без знака* (коротко, *целым без знака*). *Целым* называется как целое без знака, так и целое без знака, перед которым поставлен знак $+$ или $-$. Например, целыми без знака являются

$$00 \mid 133 \mid 012$$

целыми являются как только что приведенные числа, так и

$$-00 \mid +245 \mid -117$$

Таблица 8.2. Разделители и описатели

Разделители						Описатели	
Указатели следования		Конструктивные разделители		Скобки			
В рассматриваемом языке публикаций	В эталонном языке	Символ	Название	Символ	Название	В рассматриваемом языке публикаций	В эталонном языке
на	**go to**	,	запятая	(	скобки	**целый**	**integer**
если	**if**	.	точка	)	скобки	**вещественный**	**real**
то	**then**	10	десять	[	индексные скобки	**логический**	**Boolean**
иначе	**else**	:	двоеточие	]	индексные скобки	**массив**	**array**
для	**for**	;	точка с запятой	**начало**	операторные скобки	**переключатель**	**switch**
				конец	операторные скобки	**процедура**	**procedure**
цикл	**do**	:=	знак присваивания	«	кавычки	**собственный**	**own**
шаг	**step**	⎵	пробел	»	кавычки	**значение**	**value**
						метка	**label**
до	**until**					**строка**	**string**
пока	**whil**					**примечание**	**comment**

Правильной дробью называется запись, состоящая из точки и следующего за ней целого без знака. *Дробью* называется как целое без знака, так и правильная дробь, а также запись, состоящая из целого без знака и следующей за ним правильной дроби.

Например, правильными дробями являются

$$.000 \mid .12$$

(в общепринятой записи это будут 0,000 и 0,12). Дробями являются

$$145 \mid .57 \mid 12.73$$

(последние две дроби в общепринятой записи имеют вид 0,57 и 12,73).

Запись, состоящая из разделителя $_{10}$ и следующего за ним целого числа (без знака или со знаком), называется (*десятичным*) *порядком*, например:

$$_{10}12 \mid _{10} - 241 \mid \ _{10} + 002 \mid .$$

Числом без знака является дробь, порядок или запись, состоящая из дроби и следующего за ней порядка. Примерами последнего вида записей являются

$$14\ _{10} - 8 \mid 12.5\ _{10} + 32$$

Числом называется как число без знака, так и запись, состоящая из знака + или — и следующего за ним числа без знака. Так, все вышеописанные записи являются числами и, кроме того, числами являются

$$+\ _{10} - 43 \mid -\ _{10} - 24 \mid + 678.03_{10} - 65$$

3. Идентификаторы. Слово *идентификатор* (или, что то же, *название*) означает условное обозначение. Идентификатором может быть любая буква или любая последовательность букв и цифр, начинающаяся буквой. В АЛГОЛе идентификаторы применяются как названия переменной, массива, процедуры, функции или переключателя (эти термины определены ниже), а также в качестве меток операторов.

4. Строки. Любая последовательность основных символов, не содержащая кавычек, называется *чистой строкой.* Это название не совсем удачно, так как чистая строка, как это видно из дальнейшего, еще не является строкой. Заметим, что пустую последовательность основных символов тоже можно считать чистой строкой.

Строкой называется заключенная в кавычки чистая строка, а также любая последовательность, состоящая из строк и чистых строк и заключенная в кавычки.

Например, чистыми строками являются

$$a+b\,|\ \textit{вычислить}\ x\ 1\,|$$

Строками являются

$$«a+b;\ »\ |«\quad »|\ «\ x\ [[1\leqslant «a+b»\ »$$

5. Переменные. Массивы. Термин *простая переменная* обозначает величину, принимающую числовые или логические значения и обозначенную идентификатором.

Массивом называется конечная совокупность величин, обозначенных одним и тем же идентификатором и поставленных в соответствие упорядоченным наборам значений некоторого количества целочисленных параметров, каждый из которых может принимать любое значение, начиная с некоторого своего наименьшего и кончая некоторым своим наибольшим значением.

Упомянутые наибольшее и наименьшее значения называются *граничной парой* параметра. Общий идентификатор всех величин, входящих в состав массива, называется идентификатором массива. Массив обозначается записью, состоящей из этого идентификатора и следующего за ним заключенного в индексные скобки списка граничных пар. Граничные пары в этой записи разделены запятыми, а между наименьшим и наибольшим значениями каждого параметра (в каждой граничной паре) стоит знак : . В более общем случае элементами граничных пар могут быть арифметические выражения (см. п. 8), которые в этом случае называются *индексными выражениями.* Массивы могут иметь обозначения

$$a[0:3]\,|\ \textit{Табл}\ [1:5,\ 4:7]\ |\ u\ [k:k+3,\ l:l\uparrow 2].$$

Число граничных пар массива называется его *размерностью.*

Для выделения отдельной величины массива служит так называемая *переменная с индексами*, которая записывается в виде идентификатора массива, за которым следуют квадратные (индексные) скобки, охватывающие список индексов (разделенных запятыми). Число индексов в списке должно быть равно размерности массива. Индексами могут быть значения параметров или арифметические (называемые при этом индексными) выражения (см. п. 8). Для вышеприведенных массивов переменные с индексами могут записываться в виде

$$a[2] \mid a[i] \mid \textit{Табл}\ [i, j] \mid u[j+1, i+j]$$

Простые переменные и переменные с индексами объединяются в понятие *переменной.*

6. Указатели функций. *Функцией* называется алгорифм, заданный описанием процедуры (см. п. 11), который определяет одно числовое или логическое значение функции для каждой совокупности значений фактических параметров. Фактическими параметрами могут при этом быть строки, идентификаторы массивов, идентификаторы процедур, идентификаторы переключателей и выражения (см. пп. 11 и 14).

Указатель функции записывается в виде идентификатора процедуры, за которым следует заключенный в круглые скобки список фактических параметров, разделенных ограничителями параметра. Ограничителями параметра являются либо запятые, либо записи вида

$$)\ldots:($$

где многоточие означает произвольную последовательность букв. Например, указатели функций могут иметь вид

$$r \mid f(x, y) \mid F(3, t, p) \mid F(3)\ \textit{температура}:(t)\ \textit{давление}:(p)$$

В первом из приведенных указателей список фактических параметров отсутствует или, как говорят, является пустым. Последний содержит в своем составе слова, поясняющие смысл его фактических параметров.

За некоторыми часто встречающимися функциями в языке АЛГОЛ закреплены стандартные обозначения:

$abs(A)$ — абсолютная величина значения выражения A,

$sqrt(A)$ — квадратный корень из значения выражения A,

sin (*A*), *cos* (*A*) — тригонометрические функции;
tg (*A*), *ctg* (*A*)
arcsin (*A*), *arctg* (*A*) — главные значения обратных круговых функций,
ln (*A*) — натуральный логарифм значения выражения *A*,
entier (*A*) — наибольшее целое, не превосходящее значение выражения *A*,
sign (*A*) — функция, равная $+1$ при *A* положительном, равная -1 при *A* отрицательном и 0 при $A=0$,
exp (*A*) — функция e^A.

7. Логические выражения. Определение логического выражения является рекурсивным. *Первичными логическими выражениями* называются логические значения, логические переменные, указатели логических функций, соотношения и, наконец, логические выражения, каждое из которых заключено в скобки (круглые).

Соотношением называется пара простых арифметических выражений (см. п. 8), в частности чисел и идентификаторов числовых переменных, соединенных знаком операции отношения (см. табл. 8.1). Примерами соотношений являются

$$x > 1 \mid sin(x) \leqslant 0,\ 5 \mid \textit{скорость} \neq \textit{ускорение}$$

Логическим множителем называется первичное логическое выражение, снабженное или не снабженное слева знаком $\neg$ (*не*).

Логическим одночленом называется как отдельный логический множитель, так и последовательность, состоящая из логического одночлена, знака $\wedge$ (*и*) и логического множителя. Другими словами, логическим одночленом может быть последовательность конечного числа логических множителей, соединенных знаками $\wedge$.

Логическим многочленом называется как отдельный логический одночлен, так и последовательность, состоящая из логического многочлена, знака $\vee$ (*или*) и логического одночлена. Таким образом, логический многочлен представляет собой последовательность конечного числа логических одночленов, соединенных знаками $\vee$.

Импликацией называется как отдельный логический многочлен, так и последовательность, состоящая из импликации, знака $\rightarrow$ и логического многочлена.

Простым логическим выражением называется как отдельная импликация, так и последовательность, состоящая из простого логического выражения, знака $\sim$ и импликации. Простое логическое выражение является понятием, объединяющим в себе также понятия первичного логического выражения, логического множителя, логического одночлена, логического многочлена и импликации.

Условием называется последовательность, состоящяя из символа **если** (произвольного) логического выражения и символа **то**. Примерами условий являются

если $x>0$ **то** | **если** $x>0 \wedge y<2$ **то** |

если $x>0 \rightarrow y<0$ **то**

Логическим выражением называется как простое логическое выражение, так и последовательность, состоящая из условия, простого логического выражения, символа **иначе** и (произвольного) логического выражения.

Пример 8.4. Логическими выражениями являются **если** $x>0$ **то** $x+1>1$ **иначе** $x+1\leqslant 1$ | **если** $x>0$ **то** $y>1$ **иначе если** $y>1$ **то** $z\geqslant 3$ **иначе** $z\leqslant 8$ | **если** $x>0$ **то** (**если** $y>1$ **то** $z\geqslant 3$ **иначе** $z\leqslant 8$) **иначе** $y>1$ | $x\neq y \wedge y=z$ | $p1 \wedge p2 \vee p1 \wedge \neg p2$ | **ложь** и т. п.

8. Арифметическое выражение. Определение арифметического выражения рекурсивно и построено аналогично определению логического выражения.

Первичным арифметическим выражением называется число, переменная, указатель функции или (произвольное) арифметическое выражение, заключенное в (круглые) скобки.

Множителем называется либо первичное арифметическое выражение, либо последовательность, состоящая из множителя, знака $\uparrow$ и первичного выражения.

Одночленом называется как отдельный множитель, так и последовательность, состоящая из одночлена, одного из знаков $\times$ или $/$ или $\div$ и множителя.

Простым арифметическим выражением называется как отдельный одночлен, снабженный или не снабженный слева одним из знаков $+$ или $-$, так и последовательность, со-

стоящая из простого арифметического выражения, одного из знаков + или — и одночлена.

Арифметическим выражением называется как отдельное простое арифметическое выражение, так и последовательность, состоящая из условия (см. п. 7), простого арифметического выражения, символа **иначе** и (произвольного) арифметического выражения.

Пример 8.5. Следующие записи являются арифметическими выражениями:

-25.456 |

если $x>0$ **то** x **иначе** 1 |

если $x>y$ **то** $(x+y)$ **иначе** $(x-y)\uparrow 2+1$ |

$2+$ **если** $x+y<0$ **то** 2 **иначе** $y+3z$

Пример 8.6. Значения функции, которая на традиционном математическом языке задается формулой

$$y=\begin{cases} x+1 & \text{при } x<0 \\ 1 & \text{при } x=0 \\ 2-x & \text{при } x>0 \end{cases}$$

можно задать с помощью арифметического выражения

если $x<0$ **то** $(x+1)$ **иначе если** $x=0$ **то** 1 **иначе** $2-x$

Арифметическое выражение, применяемое для вычисления значения целочисленного параметра (индекса), называется *индексным выражением.* Индексное выражение должно принимать только целые неотрицательные значения. Если же при вычислении будет получено вещественное значение (но не целое), то в качестве значения индексного выражения принимается *entier* $(A+0.5)$, где A — вычисленное значение, а операция *entier* имеет смысл, описанный в п. 6.

9. Именующее выражение. Операторы языка АЛГОЛ (см. п. 10), входящие в состав алгорифма, можно помечать, т. е. давать им индивидуальные названия. Эти названия операторов называются *метками.* Метка является идентификатором. Именующие выражения служат в конечном счете для «вычисления» меток.

Простым именующим выражением называется метка, указатель переключателя или произвольное именующее выражение, заключенное в (круглые) скобки.

Указателем переключателя (переключатель — см. п. 11) называется последовательность, состоящая из идентификатора, переключателя и заключенного в индексные скобки произвольного индексного выражения (т. е. арифметического выражения, предназначенного для вычисления индекса).

Именующим выражением называется как отдельное простое именующее выражение, так и последовательность, состоящая из условия (см. п. 7), простого именующего выражения, символа **иначе** и произвольного именующего выражения.

Пример 8.7. Простыми именующими выражениями (а значит, именующими выражениями) являются метка *счет* и указатель переключателя *переход* $[i+j]$. Именующими сложными выражениями являются

если $i \leqslant 2$ **то** *счет* **иначе если** $i \leqslant 5$ **то** *переход* $[i]$ **иначе** *переход* $[i+j]$

а также

если $i > 2$ **то** (**если** $i \leqslant 5$ **то** *переход* $[i]$ **иначе** *переход* $[i+j]$) **иначе** *счет*

10. Операторы. Основными понятиями в АЛГОЛе являются понятия операторов и различных их типов, подтипов и видов. Эти понятия можно ввести лишь рекурсивным образом. Прежде чем дать такое определение, отметим, что всякий оператор может быть «помеченным», т. е. снабженным (слева) меткой или несколькими метками. Метка соответствует правому знаку перехода в языке логических схем. Мы, следуя [44], в качестве меток допускаем только идентификаторы, т. е. последовательности букв и цифр, начинающиеся обязательно буквой. Заметим, что в эталонном языке в качестве меток допускаются также целые (числа) без знака. При определении понятия операторов мы не будем указывать, помечены они или не помечены, допуская и первое и второе.

При записи метка отделяется от оператора или от других меток, если оператор помечен несколькими метками, разделителем : (двоеточие).

Рекурсивное определение операторов.

1°. Операторами по определению являются либо безусловные операторы, либо условные операторы. Таким образом, существует два типа операторов. Будем для удобства пояснений произвольный оператор обозначать буквой S, а произвольный безусловный оператор — буквой Q.

2°. Безусловный оператор — это либо оператор цикла, либо составной оператор, либо блок, либо основной оператор. Таким образом, существует четыре подтипа безусловных операторов.

Для удобства дальнейших пояснений будем произвольный безусловный оператор, не являющийся циклом, обозначать буквой Q.

3°. Составным оператором называется строка операторов, разделенных точками с запятыми, заключенная между операторными скобками **начало** и **конец.** Таким образом, составной оператор имеет запись вида

начало S_1; S_2; ...; S_k **конец.**

4°. Блоком называется строка описаний и следующая за ней строка операторов, заключенные в операторные скобки; описания и операторы разделены точками с запятыми, т. е. блок записывается как

начало M_1; M_2; ...; M_l; S_1; S_2; ...; S_n

конец

Здесь M_i — описания. В описаниях должны быть заданы обозначения локализованных в данном блоке величин. Если такие обозначения используются вне блока, то на время выполнения блока они приобретают новые значения. Остальные величины, фигурирующие в блоке, имеют тот же смысл, что и вне блока. Все метки, стоящие внутри блока, имеют локальный смысл, так что войти в блок можно только через его начало.

5°. Оператор цикла состоит из заголовка цикла и оператора, т. е. может быть изображен как ZS. Здесь Z — заголовок цикла. В заголовке цикла указываются параметр цикла и способ получения его последовательных значений, при которых должен быть выполнен оператор S.

6°. Основной оператор по определению может быть либо оператором присваивания, либо оператором перехода, либо оператором процедуры, либо оператором «пусто». Таким образом, подтип основных операторов (являющихся безусловными) охватывает собой четыре вида операторов.

7°. Оператором присваивания называется формула, которая имеет одну или несколько равноправных между

собой левых частей и правую часть. Формула имеет вид

$$x_1 := x_2 := \ldots := x_s : W,$$

где x_i — переменные или идентификаторы процедур, W — арифметическое или логическое выражение, а знак «:=» называется знаком присваивания; он означает, что значение величины или выражения, записанного в правой части формулы, нужно присвоить величине, записанной слева от него. Объяснение случая, когда слева от знака := стоит идентификатор процедуры, см. в п. 15.

8°. Оператор перехода записывается так:

на W

Здесь W — именующее выражение. Оператору перехода в языке логических схем соответствует понятие левого знака перехода. Выполнение оператора перехода заключается, в конечном счете, в получении некоторой метки и в переходе к оператору, имеющему такую метку.

9°. Оператором процедуры называется совокупность идентификатора процедуры и перечня так называемых фактических параметров. Выполнение оператора процедуры заключается в следующем. Среди описаний, присутствующих в блоке, в котором содержится выполняемый оператор процедуры, находят описание процедуры. Если его там не окажется, ищут описание процедуры в блоке, содержащем данный, и так далее до тех пор, пока описание процедуры не будет найдено. Оно состоит из заголовка процедуры и тела процедуры, которое является оператором. В заголовке содержатся перечень формальных параметров процедуры и так называемый список значений, в котором повторены некоторые из формальных параметров. Тем формальным параметрам процедуры, которые указаны в списке значений, производится присваивание значений фактических параметров из числа указанных в операторе процедуры. Остальные формальные параметры процедуры заменяются в теле процедуры отвечающими им фактическими параметрами. Соответствие между формальными параметрами процедуры и фактическими параметрами, указанными в операторе процедуры, получается путем сопоставления в одном и том же порядке элементов перечня фактических параметров и перечня формальных параметров. Преобразованное описанным способом тело процедуры вы-

полняется, чем и завершается выполнение оператора процедуры.

Таким образом, оператор процедуры осуществляет преобразование тела процедуры и выполнение его. Преобразование тела процедуры осуществляется с помощью двух операций: а) присваивания (подстановки) значений формальным параметрам и б) замены формальных параметров фактическими (в частности, это может быть замена одной переменной величины другой переменной величиной или выражением).

10°. Оператор «пусто» не пишется. Он обладает двумя свойствами: а) его выполнение заключается в переходе к следующему за ним оператору; б) ему можно присвоить метку.

11°. Условные операторы могут быть двух видов, т. е., по определению, условный оператор это либо оператор «если», записываемый как

если P **то** Q

либо оператор, записываемый как

если P **то** Q' **иначе** S

В обеих записях P означает некоторое логическое выражение. На языке логических схем условные операторы указанных типов могут быть изображены соответственно как

$$P\underset{\alpha}{\lfloor}\,Q\,\underset{\alpha}{\rfloor} \quad \text{и} \quad P\underset{\alpha}{\lfloor}\,Q'\underset{\beta}{\lfloor}\,\underset{\alpha}{\rfloor}\,S\,\underset{\beta}{\rfloor}$$

Особо подчеркнем, что включение цикла в оператор

если P **то** Q' **иначе** S

на место безусловного оператора Q' (не являющегося циклом) недопустимо, так как может приводить к двусмысленным записям.

Приведенное определение операторов, их типов, подтипов и видов не является «порочным кругом» и позволяет путем многократного применения строить сколь угодно сложные операторы. Программа определяется в АЛГОЛе как оператор, чаще всего блок, не являющийся частью другого оператора и содержащий в себе все свои части (в том числе и необходимые описания). В соответствии с этим программа представляет собой строку операторов и описаний, после каждого из

которых стоит знак ; (точка с запятой), сгруппированных с помощью операторных скобок. Кроме того, в программе могут содержаться примечания, которые при ее выполнении не принимаются в расчет. После выполнения каждого оператора программы производится переход к непосредственно за ним следующему, за исключением оператора перехода, изменяющего порядок выполнения операторов. Отдельные виды операторов несколько ниже будут описаны более подробно.

11. Описания. Описания подразделяются на *описания типа, описания массивов, описания переключателей и описания процедур.*

Описание типа (простой переменной) характеризует одну или несколько простых переменных и представляет собой последовательность символов, состоящую из двух частей: названия типа и списка простых переменных. *Названием типа* может быть описатель или пара описателей (в произвольном роде и числе);

целый | вещественный | логический | собственный целый | собственный вещественный | собственный логический

Список простых переменных представляет собой последовательность идентификаторов, разделенных запятыми. Описатели

целый | вещественный | логический

не требуют пояснения. Смысл описателя **собственный** состоит в следующем. Как уже было сказано, величины, описанные внутри некоторого блока, локализованы внутри этого блока. После «выхода» из блока они теряют смысл или получают новый смысл и при новом входе в блок определяются заново. Это иногда бывает неудобно. Если требуется, чтобы величина сохраняла свое значение (внутри блока; вне блока она по-прежнему не имеет смысла или имеет другой смысл), то в описание включают описатель **собственный** |.

Пример 8.8. Примером описаний типа могут служить

целые *x, y. Номер, z*2

собственные вещественные *u, v*1, *v*2, *Вес, Время*

логические *p*1, *p*2, *Условие 1, Условие 2*

Описание массива представляет собой последовательность следующих своих частей: названия типа (описано

выше), описателя **массив** и некоторого числа разделенных запятыми так называемых сегментов. Каждый сегмент представляет собой последовательность идентификаторов массивов, разделенных запятыми, и следующего за последним идентификатором списка граничных пар, заключенного в индексные скобки (см. п. 5). Если название типа сводится к единственному описателю **действительный**, то оно может быть опущено. Таким образом, описание массивов может иметь один из следующих видов:

массивы R_1, R_2, ..., R_k | T **массивы** R_1, R_2, ..., R_k |.

Здесь T означает название типа, а R_i — сегмент.

Пример 8.9. Описания массивов могут иметь вид

целые массивы *A*, *B* [*1*:*6*, *3*:*40*, *2*:*122*], *u*, *v* [*1*:*20*]
логический массив *Возможность* [*1*:*3*, *1*:*5*]
собственный вещественный массив *w* [*1*:*120*]
массивы *U*, *V*, *W* [*1*:*100*, *20*:*400*]
массив *X* [*1*:*n*, *m*:m^2+1]

В каждом сегменте список граничных пар относится ко всем предшествующим ему идентификаторам массивов. Таким образом, описание массивов может давать сведения либо об одном, либо сразу о нескольких массивах. Название типа относится сразу ко всем массивам, о которых задается информация в описании.

Описание переключателя представляет собой последовательность символов, состоящую из описателя **переключатель**, идентификатора переключателя, разделителя := и переключательного списка, состоящего из именующих выражений, отделенных друг от друга запятыми.

Пример 8.10. Описания переключателей могут иметь вид

переключатель *R*:=*M1*, *M2*, *M3*, *вход 4*

переключатель *переход* := **если** $x>1$ **то** *проверка* **иначе** *выход*, **если** $x<5$ **то** *M* [*k*+*1*] **иначе** *M* [*k*+*2*]

Описание процедуры состоит из заголовка процедуры и следующего за ним тела процедуры, разделенных точкой с запятой. Тело процедуры представляет собой оператор, обычно блок. Заголовок процедуры состоит из описателя **процедура** и идентификатора процедуры; далее могут следовать заключенный в круглые скобки список формальных

параметров, список значений и одна или несколько спецификаций. Описателю **процедура** может предшествовать название типа процедуры (см. п. 15):

целая | вещественная | логическая

Список формальных параметров представляет собой последовательность идентификаторов, отделенных друг от друга ограничителями параметра. Ограничителями параметров являются либо запятые, либо записи вида) ... : (, где многоточие означает произвольную последовательность букв.

Список значений представляет собой последовательность, состоящую из описателя **значение** и списка идентификаторов (т. е. последовательности идентификаторов, разделенных запятыми). В список значений могут входить только те идентификаторы, которые являются формальными параметрами.

Спецификация начинается одним описателем или парой описателей из следующих:

строка | метка | переключатель | массив | процедура | целый | вещественный | логический | целый массив | вещественный массив | логический массив | целая процедура | вещественная процедура | логическая процедура.

Далее в спецификацию входит список идентификаторов (разделенных запятыми) из числа фактических параметров.

Роль каждого элемента в описании процедуры можно объяснить только в процессе рассмотрения оператора процедуры.

Пример 8.11. Заголовок процедуры может иметь следующий вид:

процедура *сумма* (*y*) *пределы*: (*m*, *n*) *параметр*: (*i*) *результат*: (*s*); **значения** *m*, *n*; **целые** *m*, *n*, *i*; **вещественное** *y*;

Здесь *сумма* — идентификатор процедуры; *y*, *m*, *n*, *i*, *s* — формальные параметры, текст

значения *m*, *n*

является списком значений; далее следуют две спецификации.

12. Операторы присваивания. Для оператора присваивания (см. п. 10, 7°) установлен следующий порядок его выполнения.

1° Вычисляются все индексные выражения, встречающиеся в переменных левой части последовательно слева направо.

2° Вычисляется значение выражения, стоящего в правой части.

3° Значение выражения присваивается всем переменным левой части с индексными значениями, вычисленными в соответствии с 1°.

Все переменные левой части должны иметь одинаковый тип.

Если тип значения правой части **вещественный**, а тип переменных в левой части — **целый**, то переменным присваивается значение *entier* $(A + 0.5)$, где A — вычисленное значение арифметического выражения.

13. Операторы перехода. Выполнение оператора перехода на W

(см. п. 10, 8°) заключается в вычислении значения именующего выражения (см. п. 9) W. Если при этом получится идентификатор метки, то производится переход к оператору с этой меткой, содержащемуся в том же блоке, что и оператор перехода. Если в этом блоке искомой метки нет, ее ищут в блоке, содержащем данный блок, и так далее, до тех пор, пока метка не будет найдена.

Если при вычислении значения W будет получен результат, представляющий собой идентификатор переключателя и значение индексного выражения (целое без знака), то подобно тому как производится в вышеописанном случае поиск метки, в этом случае производится поиск описания переключателя по его идентификатору (см. п. 11). После того как описание переключателя будет найдено, в переключательном списке описания выбирается индексное выражение, номер которого, если считать в списке слева направо, равен вычисленному значению индексного выражения. Далее, вычисляется значение нового индексного выражения и весь процесс, описанный выше, повторяется.

Если при этом именующее выражение, находящееся во внешнем блоке, содержит идентификаторы меток, переменных, переключателей или процедур, описанных также во внутреннем блоке, то считается (временно), что эти идентификаторы имеют те значения, которые им были присвоены при выполнении операторов внешнего блока. В конце концов будет найдена метка, поиск которой производится вышеописанным способом.

Пример 8.12. Рассмотрим такой участок программы:

```
начало вещественное x; целое i;
       переключатель N:= если x>0 то M2
       иначе M3, N [i];
       . . . . . .
            x:=1; i:=1;
       начало вещественное x; целое i;
            . . . . . .
            x:=-1; i:=2;
       на N [i];
            M2: x:=3;
       . . . . . .
       конец;
       . . . . . .
конец;
```

При выполнении оператора перехода **на** $N[i]$, стоящего во внутреннем блоке, вычисляя именующее выражение $N[i]$, получим $N[2]$ (так как непосредственно перед выполнением оператора перехода переменной i было присвоено значение 2). Но описание переключателя с идентификатором N находится во внешнем блоке. Выбираем второй элемент в списке переключателя N. Этот элемент является индексным выражением $N[i]$. Хотя переменная i описана во внутреннем блоке и равна там 2, но во внешнем блоке она была равна 1. Поэтому при вычислении значения последнего индексного выражения получим $N[1]$. Это значит, что мы снова должны обратиться к переключателю N, но теперь в его описании взять первое именующее выражение, а именно:

если $x>0$ **то** *M2* **иначе** *M3*

Переменная x, имевшая во внутреннем блоке значение -1, во внешнем блоке имеет значение 1. Поэтому значением последнего индексного выражения будет метка *M2*. Начинаем ее поиск с внутреннего блока, так как именно там находится оператор перехода и находим там эту метку. Переходим к выполнению оператора присваивания $x:=3$;.

14. Оператор процедуры. Оператор процедуры состоит из идентификатора процедуры и, возможно, следующего за ним списка фактических параметров, заключенного в круглые скобки. В этом списке фактические параметры разделены запятыми или ограничителями параметра вида

)...:(

как в указателе функции. Фактическим параметром может быть либо выражение (в частности, число или переменная), либо идентификатор массива, переключателя или процедуры. Выполнение оператора процедуры в общих чертах описано в п. 10, 9°.

После выполнения тела процедуры производится переход к оператору, непосредственно следующему за оператором процедуры.

Необходимо подчеркнуть, что в случае, когда описание процедуры отсутствует в том блоке, в котором стоит оператор процедуры, его ищут в блоке, содержащем внутри себя данный. Если его и там не находят, то выходят в блок, содержащий внутри себя последний блок, и действуют так до тех пор, пока не будет найдено искомое описание процедуры. Как и в случае оператора перехода, идентификаторы, локализованные внутри блока, содержащего оператор процедуры, если они имеют иной смысл в блоке, содержащем описание процедуры, считаются имеющими этот последний смысл (т. е. присущий внешнему блоку, в котором стоит описание процедуры).

Язык АЛГОЛ допускает процедуры, описанные на каком-либо языке, отличном от АЛГОЛа. Предполагается, что каждый, кто пользуется такими процедурами, понимает язык, на котором они описаны. Фактическими параметрами таких процедур могут быть кроме выражений и идентификаторов также строки (см. п. 4).

Описание такой процедуры состоит из заголовка, построенного по общим правилам, и тела процедуры, на которое в языке АЛГОЛ не накладывается никаких ограничений. В связи с тем, что фактическими параметрами описываемой процедуры могут быть строки, в заголовке ее описания (см. п. 11) могут содержаться спецификации особого вида, состоящие из описателя **строка** и следующего за ним списка идентификаторов, разделенных запятыми. Формальным параметрам, включенным в этот список, должны в операторе процедуры соответствовать фактические параметры, являющиеся строками.

Строки могут быть также фактическими параметрами процедуры, описанной на АЛГОЛе, если в теле такой процедуры имеется обращение к процедуре, описанной не на АЛГОЛе.

Пример 8.13. Дано описание процедуры:

Процедура *U1* (*a*, *y*); **вещественное** *a*;

начало ...; *U2* (*y*);... **конец**

Предположим, что формальный параметр *y* встречается в теле процедуры *U1* только в качестве фактического параметра оператора процедуры *U2*, которая описана не на АЛГОЛе. В таком случае фактическим параметром оператора процедуры *U1*, соответствующим формальному параметру *y*, может быть строка. Например,

упомянутый оператор процедуры может иметь вид

U1 (0.512, «0025 ⎵ 0031 ⎵ 0452 ⎵ 0⎵ 01»)

Входящая в его состав строка представляет собой команду машины (в данном случае — *Стрелы*).

Процедура, при выполнении которой производится обращение к этой же самой процедуре, называется рекурсивной. При каждом таком обращении величины, локализованные в блоке, содержащем описание рекурсивной процедуры, получают новые значения (а прежние их значения «запасаются» до момента окончания обращения), т. е. получается как бы переход в новый блок, внутренний для того, в котором мы находились в момент обращения.

При выполнении тела процедуры иногда имеет место так называемый *побочный эффект*, который заключается в том, что содержащиеся в теле процедуры нелокализованные в ней переменные при выполнении тела процедуры получают новые значения, хотя это не является целью процедуры.

При выполнении тела процедуры с целью вычисления значения функции (см. п. 6) побочный эффект может выразиться в изменении значений фактических параметров функции, а также в выходе из тела процедуры к нелокализованной в ней метке (благодаря чему значение функции никогда не будет вычислено). Возможность побочного эффекта необходимо учитывать при программировании.

Пример 8.14. При выполнении оператора присваивания

$$z := f_1(x) + f_2(x)$$

побочный эффект, сопровождающий вычисление $f_1(x)$ и заключающийся в изменении значения фактического параметра x, привел бы к вычислению $f_2(x)$ не для того значения x, которое предусмотрено в операторе присваивания, а для другого. Однако влияние побочного эффекта можно устранить, если вышеприведенный оператор заменить парой операторов

$$y := x;\ z := f_1(y) + f_2(x).$$

В этом случае побочный эффект выразится в изменении значения переменной y, что не повлияет на вычисление $f_2(x)$.

15. Вычисление значений функции. Описание процедуры, предназначенной для вычисления значения функции, начинается парой описателей:

логическая процедура | целая процедура | вещественная процедура

В теле процедуры должен содержаться оператор присваивания, который идентификатору этой процедуры присваивает значение, равное вычисленному значению функции. Такой оператор присваивания может присутствовать только в теле данной процедуры и нигде более. Тип значения, присваиваемого идентификатору процедуры, должен соответствовать самому первому описателю, стоящему в заголовке процедуры.

16. Оператор цикла. Оператор цикла состоит из заголовка и произвольного оператора (п. 10, 5°), подлежащего многократному выполнению (в частности, однократному или нулькратному).

Заголовок цикла представляет собой последовательность, состоящую из указателя следования **для** переменной, называемой параметром цикла, знака присваивания, списка цикла и указателя следования **цикл**.

Списком цикла называется последовательность разделенных запятыми *элементов списка цикла*. Элементы списка цикла бывают трех типов. Элемент первого типа, называемый *элементом типа отдельного значения*, представляет собой арифметическое выражение (в частности, переменную или число). Элемент второго типа, называемый *элементом типа арифметической прогрессии*, состоит из трех арифметических выражений, разделенных указателями следования **шаг** и **до**. Этот элемент имеет вид

A1 **шаг** *A2* **до** *A3*

где *A1*, *A2* и *A3* — произвольные арифметические выражения. Элемент третьего типа, называемый *элементом итеративного типа*, представляет собой последовательность, состоящую из арифметического выражения, указателя следования **пока** и логического выражения. Он имеет вид

A **пока** *B*

где *A* и *B* — соответственно арифметическое и логическое выражения. В списке цикла элементы могут сочетаться любым образом.

Выполнение оператора цикла осуществляется следующим образом. Присваивается значение параметру цикла, после чего

выполняется внутренний оператор цикла, затем, если это возможно, присваивается новое значение параметру цикла и снова выполняется внутренний оператор цикла и так до тех пор, пока присвоение значения параметру цикла не перестанет быть возможным. После этого производится переход к оператору, непосредственно следующему за оператором цикла.

Присвоение значений параметру цикла производится с помощью элементов списка цикла. Начинается этот процесс с первого элемента. После того как действие очередного элемента заканчивается, начинается действие следующего за ним. После окончания действия последнего элемента присвоение новых значений параметру невозможно (до тех пор, пока не будет произведен новый переход к рассматриваемому оператору цикла).

Действие элемента первого типа заключается в одноразовом вычислении арифметического выражения и присвоении его значения параметру цикла.

Действие элемента второго типа заключается в том, что вычисляются значения трех арифметических выражений, первое принимается за первый член, второе за разность, а третье за последний член арифметической прогрессии, и параметру цикла последовательно присваиваются значения всех членов арифметической прогрессии, начиная с ее первого члена и кончая ее последним членом, после чего действие элемента оканчивается.

Действие элемента третьего типа заключается в том, что вычисляется значение арифметического выражения и присваивается параметру цикла, затем вычисляется значение логического выражения и, если оно есть **истина**, выполняется внутренний оператор цикла. Если значение логического выражения окажется **ложь**, то считается, что элемент уже прекратил свое действие.

Переход извне цикла на метку внутри цикла невозможен, выход же из цикла на метку вне цикла возможен.

Пример 8.15. Вычисление значения алгебраической суммы

$$y = \sum_{i=1}^{n} a_i$$

можно произвести с помощью пары операторов, из которых первым

является оператор присваивания

$$y := 0$$

а вторым — один из нижеприведенных операторов цикла:

для $i := 1$ **шаг** 1 **до** n цикл $y := y + ai\ [i]$

для $i := 1, j$ **пока** $j < n + 1$ **цикл начало** $j := i$;

$y := y + a\ [i]$; $j := i + 1$ **конец**;

В первом из приведенных операторов цикла список цикла состоит из одного элемента типа арифметической прогрессии. Во втором операторе цикла список цикла состоит из двух элементов: элемента типа отдельного значения и элемента итеративного типа.

17. Примечания в программах. Как уже говорилось, программа представляет собой либо составной оператор, либо блок. Соседние операторы программы разделены точками с запятой. Существует три способа включения в программы примечаний — текста, поясняющего людям смысл программы и отдельных ее частей, но не принимаемого во внимание при выполнении программы.

1° Произвольный текст, начинающийся описателем **примечание**, кончающийся точкой с запятой и не содержащий внутри себя точек с запятой, можно поместить после любого ограничителя **начало** и после любой точки с запятой.

2° Произвольный текст, не содержащий символов **конец, иначе** и точек с запятой, можно вставить между любым ограничителем **конец** и следующим за ним символом.

3° Любой текст, состоящий из букв и цифр, но начинающийся с буквы (и, значит, подобный идентификатору), можно перед любым оператором (помеченным или не помеченным) вставить в виде метки.

4° Текст, состоящий только из букв, можно включить в список формальных параметров заголовка процедуры (п. 11) или в список фактических параметров оператора процедуры или указателя функции (пп. 6 и 14) в качестве составных частей ограничителей параметра.

18. Некоторые замечания. Вышеприведенное описание АЛГОЛа позволяет сделать вывод, что этот язык далеко не общедоступен. Лишь его начатки могут быть легко усвоены нематематиками.

Построение алгорифма на АЛГОЛе по своей трудности мало чем отличается от программирования на языке машины. По мнению ряда работников-практиков при построении АЛГОЛ-программ целесообразно в качестве языка, промежуточного между математической формулировкой задачи и АЛГОЛ-программой, пользоваться некоторыми элементами языка логических схем, например логическими схемами, составленными на абстрактном уровне и со словесным описанием операторов этих схем.

При разработке АЛГОЛ-программ значительные трудности возникают в связи с возможностью побочного эффекта процедур и сложностью процессов выполнения операторов перехода и тел процедур. Это часто приводит к ошибкам в АЛГОЛ-программах. При автоматическом переводе на язык машины обнаруживаются без труда лишь некоторые синтаксические ошибки. Надежная работа трансляторов позволяет не сомневаться в том, что полученная программа соответствует своему прототипу — АЛГОЛ-программе. Остается не устраненным сомнение в том, что сама АЛГОЛ-программа соответствует математической формулировке метода решения задачи. Поэтому вместо отладки программы на машине пользование АЛГОЛом вынуждает производить отладку АЛГОЛ-программ на машине. Протекает этот процесс примерно следующим образом. АЛГОЛ-программу транслируют, решают задачу на машине, анализируют результаты и промежуточные результаты и, если они вызывают сомнения, начинают исправлять АЛГОЛ-программу. Затем процесс повторяется. Методы отладки АЛГОЛ-программ на машине еще мало разработаны и подробнее не могут быть освещены в этой книге.

Несмотря на перечисленные обстоятельства, популярность АЛГОЛа непрерывно возрастает. Разрабатываются трансляторы с АЛГОЛа на языки различных программно-управляемых машин, публикуется большое число статей, посвященных АЛГОЛу, в СССР и за рубежом публикуются описания различных вычислительных процессов на АЛГОЛе.

В заключение заметим, что конкретный уровень языка логических схем, если под объектами понимать величины, представляет собой алгорифмический язык, который с точностью до обозначений совпадает с некоторым подъязыком АЛГОЛа (при условии ограничения числа и вида применяемых

в ЯЛС операций). По-видимому, этот алгорифмический язык, являющийся пересечением ЯЛС и АЛГОЛа, и определяет ту часть АЛГОЛа, которая успешно может быть усвоена и применена на практике неспециалистами в области программирования для подготовки задач к решению их на цифровых программно-управляемых машинах.

Кроме алгорифмических языков, описанных в этой главе, существует и разрабатывается еще ряд алгорифмических языков, из которых можно указать КОБОЛ, разработанный в США для решения экономических задач, и *адресный* универсальный алгорифмический язык, разработанный в Киеве [16, 72].

Литература к главе 8: [4], [26], [28], [39], [44], [50], [72].

ЧАСТЬ ВТОРАЯ

ОПИСАНИЯ НЕКОТОРЫХ ПРОГРАММНО-УПРАВЛЯЕМЫХ МАШИН

ВВОДНЫЕ ЗАМЕЧАНИЯ

Во второй части книги приведены описания отечественных программно-управляемых машин, которые выпускались или в настоящее время выпускаются серийно.

Описания машин даются примерно в одном плане: основные характеристики машины, краткое описание устройства машины, представление чисел и команд в машине, система операций и команд, функции пульта управления и правила кодирования данных для ввода в машину. Описания содержат, как правило, все необходимые сведения для возможности программирования задач, отладки и решения их на машине.

Основное содержание каждого описания составляет система команд и операций машины, которая дается в форме таблицы. В таблице указывается код операции, соответствующий описываемой команде; наименование команды и ее условное обозначение, отражающее деление кода команды на части (адреса, признаки и т. п.); краткое содержание команды, записываемое условными обозначениями, условия выработки машиной сигналов $\omega = 1$ и $\varphi = 1$ после каждой команды и полное словесное описание команды. Краткое описание действий, выполняемых машиной по команде (содержание команды), дается в виде комплекса на языке логических схем (гл. 8, § 1), то есть, если пользоваться терминологией АЛГОЛа, в виде последовательности операторов присваивания. Условия образования сигналов $\omega = 1$ и $\varphi = 1$ записаны по правилам, принятым для записи соотношений как в языке логических схем, так и в АЛГОЛе. Заметим, что в указанных записях названия ячеек и регистров являются обозначениями их содержимых. В тех случаях, когда названием ячейки является число (например ее номер) или алгебраическое выражение, отличное от отдельной буквы, перед названием ячейки, которое во вто-

ром случае должно быть заключено в скобки, ставится буква *я*. Этим исключается возможность неоднозначности записей. Например, вместо названий ячеек 0345, $a+12$, $a+b$ пишутся *я*0345, *я*$(a+12)$, *я*$(a+b)$.

Знаки x_M, x_P, x_I, x_II, x_III, x_θ, x_{τ_i} означают соответственно мантиссу и порядок числа, хранящегося в ячейке x и первый, второй, третий адреса, код операции и код признака τ_i команды, хранящейся в ячейке x. «C» и «F» означают соответственно содержимое регистра-счетчика команд и регистра переадресации. «x» и «x_i» означают соответственно содержимое ячейки с номером x и i-го разряда этой ячейки (или устройства). Индекс в скобках у числа показывает основание системы счисления, в которой записано число; например $63_{(10)}=77_{(8)}$.

Если в графе условия выработки сигналов $\omega=1$ или $\varphi=1$ оставлено пустое место, это означает, что при выполнении описываемой команды всегда вырабатывается соответственно только сигнал $\omega=0$ или $\varphi=0$. Знак $\equiv$ означает сохранение того значения сигнала ω, которое имело место до выполнения описываемой команды.

В описании структуры команды вместо ее структурного элемента (адреса, признака или кода операции), не влияющего на работу машины, ставится * (звездочка).

Символы $\operatorname{sign} x$ и $\overline{\operatorname{sign} x}$ означают соответственно знак числа «x» и знак, обратный знаку числа «x».

ГЛАВА 9

ПРОГРАММНО-УПРАВЛЯЕМАЯ МАШИНА ДНЕПР (*Дніпро*)

Основные характеристики машины *Днепр*

Тип машины	управляющая
Класс машины	средняя
Среднее число основных операций в секунду	15 000
Рабочая система счисления	двоичная
Форма представления чисел	с фиксированной запятой
Длина кода числа	26 разрядов
Диапазон представимых чисел	$2^{-25} \leqslant \lvert x \rvert \leqslant 1 - 2^{-25}$
Точность вычислений	7 ÷ 8 десятичных знаков
Порядок выполнения команд	заданный
Адресность машины	двухадресная
Виды запоминающих устройств	оперативное запоминающее устройство на магнитных сердечниках емкостью 512 26-разрядных двоичных чисел (предусмотрена возможность иметь четыре блока емкостью 512 26-разрядных чисел каждый), время обращения — 14 *мксек*; пассивное запоминающее устройство емкостью 512 26-разрядных двоичных чисел (предусмотрена возможность иметь шесть блоков по 512 26-разрядных двоичных чисел каждый), время обращения (считывания) — 9 *мксек*; внешнее запоми-

	нающее устройство на магнитной ленте *) емкостью 200 000 26-разрядных двоичных чисел, числа на магнитную ленту записываются зонами, среднее время поиска зоны 75 *сек*, скорость выборки (считывания с магнитной ленты) — 500 чисел в секунду
Виды ввода и вывода	вводное устройство фотоэлектрическое со скоростью ввода 45 чисел или 40 команд в секунду, ввод с перфорационной бумажной ленты; выводное устройство — электрифицированная пишущая машина, работает со скоростью 7 знаков в секунду
Связь машины с управляемыми объектами	связь вычислительной части машины с управляемыми объектами производится через устройство связи, которое может опрашивать максимально 250 датчиков; время одного аналого-дискретного преобразования 2 *мсек*, точность преобразования 0,4%
Количество каналов для двухпозиционного регулирования	30

Машина *Днепр* может быть использована как универсальная программно-управляемая машина вычислительного типа.

Краткие сведения об основных устройствах машины

Машина *Днепр* состоит из двух частей: из вычислительной части (универсальная программно-управляемая машина вычислительного типа) и устройства связи вычислительной части машины с управляемыми объектами. Ниже приводятся краткие сведения об устройствах, входящих в состав вычислительной части, и об устройстве связи.

Арифметическое устройство (АУ). У машины *Днепр* АУ параллельного действия. Основную часть АУ составляют 31-разрядный сумматор, 26-разрядный регистр r и 31-разрядный регистр R. Сумматор и регистры выполнены на триггерах.

*) Внешнее запоминающее устройство на магнитной ленте имеется в образце машины *Днепр*, находящемся в институте Кибернетики АН УССР. В серийно выпускаемых образцах машины такое запоминающее устройство отсутствует.

В момент выполнения операции в АУ исходные числа и результат размещаются в сумматоре и регистрах следующим образом:

1) операция сложения (вычитания) — на s первое слагаемое (уменьшаемое), на R второе слагаемое (вычитаемое), результат операции записывается на s и на R;

2) операция умножения — на R множимое, на r множитель, на s произведение;

3) операция деления — на s делимое, на R делитель, на r частное, остаток на s;

4) операция арифметического (логического) сдвига — на r сдвигаемое число, на s количество сдвигов со знаком и результат;

5) операция логического умножения — на s первое число, на R второе число, результат операции на s;

6) операция сложения по $\mathrm{mod}\,2$ — на s первое слагаемое, на r второе слагаемое, результат операции на s и R.

Запоминающее устройство (ЗУ). В машине *Днепр* существуют три вида ЗУ: оперативное (ОЗУ), пассивное (ПЗУ) и внешнее (ВЗУ).

ОЗУ выполнено на магнитных сердечниках. Время обращения — 14 *мсек*. В ячейках ОЗУ могут храниться одновременно 512 26-разрядных двоичных кодов (предусмотрена возможность иметь четыре блока емкостью 512 26-разрядных двоичных кодов). Ячейки ОЗУ занумерованы восьмеричными числами 000, 001, ..., 777. Блоки ОЗУ занумерованы числами 000, 001, 002, 003. Номер блока ОЗУ задается на пульте управления.

ПЗУ трансформаторного типа. Состоит из шести блоков по 512 26-разрядных двоичных кодов. Каждый блок выполнен на четырех сменных линейках по 128 26-разрядных кодов. Блоки ПЗУ занумерованы числами 000÷005. Номер блока ПЗУ задается на пульте управления.

Замечание. Если на пульте управления номер блока ОЗУ (ПЗУ) не задан, то в качестве ОЗУ (ПЗУ) используется блок с номером 000. Обращение к ячейкам других блоков ОЗУ (ПЗУ) выполняется по команде с кодом операции 27. Во включенном состоянии машина может содержать одновременно три блока памяти; два ПЗУ (программный и констант) и один ОЗУ (рабочих ячеек) или один ПЗУ (констант) и два ОЗУ (программный и рабочих ячеек). Переключение номеров блоков ОЗУ и ПЗУ можно выполнить по командам с кодом операции 23÷26.

ПЗУ используется для хранения констант, стандартных программ, программ по изучению объектов, по управлению объектами и т. п. Команда с кодом операции 26 позволяет любую программу поместить в ПЗУ. Переменные команды этой про-

граммы переносятся в ОЗУ и выполняются в нужном месте, не нарушая естественного порядка выполнения команд программы.

ВЗУ на магнитной ленте емкостью 200 000 26-разрядных чисел. Числа на магнитную ленту записываются из ОЗУ зонами; количество чисел, записываемых на зону, не более 512. Среднее время поиска зоны 75 *сек*, скорость выборки (считывание с магнитной ленты) — 500 чисел в секунду. Разметка магнитной ленты на зоны производится с пульта управления.

Устройство управления (УУ) включает в себя следующие основные элементы: а) регистр номера (счетчик) команды; б) регистр команд.

В машине имеется возможность параллельно с выдачей на печать продолжать вычисления по программе. Как только начала выполняться команда с кодом операции 34, в машине вырабатывается признак $\omega_3 = 1$, и устройство управления готово к выполнению следующей команды. Если следующей командой является команда с кодом операции 34, то машина, приняв эту команду в устройство управления, ждет окончания команды «выдача на печать» и только после этого переходит на печать. Если следующей после команды с кодом операции 34 является команда, отличная от команды с кодом операции 34, то она выполняется независимо от того, выполнилась команда «выдача на печать» или не выполнилась. После окончания команды с кодом операции 34 значение признака ω_3 принимается равным нулю. Для выяснения, окончена выдача на печать или не окончена, используется команда условного перехода по признаку ω_3 с кодом операции 23.

Устройство ввода и вывода (УВВ). Устройство ввода — фотоэлектрическое, работает со скоростью 40 команд или 45 чисел в секунду. Ввод производится с пятипозиционной бумажной перфоленты. Перфорация ленты выполняется на СТА-2М.

Устройство вывода — электрифицированная пишущая машинка с шириной каретки 450 *мм*. Вывод производится со скоростью 7 знаков в секунду. Максимальное число знаков в строке 150 ÷ 165. Печать может осуществляться черным или красным цветом.

Устройство связи с объектом (УСО). УСО обеспечивает связь вычислительной части машины с управляемыми объектами и выполняет следующие функции:

а) производит опрос датчиков управляемых объектов;

б) обеспечивает связь оператора с управляемыми объектами через вычислительную часть машины;

в) вырабатывает сигналы двухпозиционного регулирования (включить-выключить) исполнительных элементов управляемых объектов в соответствии с кодами, полученными из вычислительной части машины;

г) по командам из вычислительной части машины принимает и хранит коды чисел, предназначенных для выработки сигналов пропорционального регулирования на исполнительные механизмы управляемых объектов.

УСО имеет 250 каналов опроса датчиков непрерывного сигнала (это значит, что к машине можно подключить максимально 250 датчиков непрерывного сигнала). Время одного опроса стандартного датчика непрерывного сигнала — 2 *мсек*. Точность преобразования непрерывного сигнала стандартного датчика в цифровую форму 0,4%.

УСО допускает подключение 400 каналов опроса двухпозиционных датчиков *). Время опроса группы двухпозиционных датчиков (не более 26 в группе) — 28 *мксек*. Задание регуляторам и адреса регуляторов записываются на *выходном 17-разрядном регистре*, входящем в состав УСО, в двоичном коде (выходной регистр предназначен для хранения результатов расчетов, выполненных машиной, которые могут быть использованы для выработки управляющих сигналов, поступающих непосредственно на исполнительные органы управляемого объекта). Имеется 30 каналов двухпозиционного регулирования для выдачи сигналов в схемы дистанционного управления, включение канала сопровождается зажиганием лампочки на пульте оператора. Имеется *счетчик времени*, отсчитывающий секунды, минуты, часы и полностью заполняющийся за сутки, после чего он сбрасывается (т. е. устанавливается в нуль при показании 24 часа). Начальную установку счетчика времени может произвести оператор с пульта при помощи клавишного устройства (оператор может ввести только часы и минуты). Работа счетчика времени начинается с нажатия кнопки «Пуск счетчика времени» на пульте оператора. Разряды счетчика времени распределяются следую-

*) Чтобы иметь каналы опроса двухпозиционных датчиков, необходимо добавить специальный блок.

щим образом: в 1 ÷ 8 разрядах считаются секунды, в 9 ÷ 16 — минуты, в 17 ÷ 22 — часы.

В процессе работы программы необходимо бывает ввести какие-либо дополнительные данные. Для этой цели предназначен 26-разрядный *регистр ввода* числа. Вводимое число набирается на пульте оператора и сопровождается признаком, набираемым на клавишном регистре. Этот признак может указывать на физический смысл вводимой величины (время, температура, давление и т. д.).

В состав УСО входят также *регистр режимов* и *регистр признака печати.*

Датчики, подсоединяемые к УСО, должны иметь:

а) выходные сигналы непрерывно меняющихся параметров в пределах 5 ÷ 25 *ма* или 0 ÷ 25 *в*;

б) токовые датчики должны допускать нагрузку в 1 ÷ 3 *ком*;

в) внутреннее сопротивление датчиков напряжения не должно превышать 1 *ком*;

г) датчики двухпозиционного типа должны иметь на выходе напряжение 0 ÷ 12 *в* и внутреннее сопротивление 1,5 *ком.*

Взаимный обмен информацией между объектом регулирования и вычислительной частью машины осуществляется с помощью команд «прием — выдача» и «включение — выключение». С помощью этих же операций осуществляется обмен информацией между вычислительной частью машины и оператором.

Представление чисел и команд

На рис. 9.1 показано распределение разрядов ячейки памяти машины *Днепр* при хранении в ней двоичного и двоично-десятичного числа с фиксированной запятой. Знак числа определяется содержимым 26-го разряда: знак плюс, если содержимое 26-го разряда 0, и знак минус, если 1.

На том же рисунке показано распределение разрядов ячейки памяти при хранении в ней команды. Команды машины *Днепр*, кроме команды «поиск зоны магнитной ленты», имеют следующую структуру:

$$\theta\ \tau\ M\ a\ b,$$

где θ — код операции, τ — признак, указывающий вид памяти ($\tau = 1$ соответствует ПЗУ, $\tau = 0$ — ОЗУ), M — код модификации,

a и *b* — соответственно первый и второй адреса (если $\tau = 1$, то *a* — ячейка ПЗУ, если $\tau = 0$, то *a* — ячейка ОЗУ; *b* всегда является ячейкой ОЗУ).

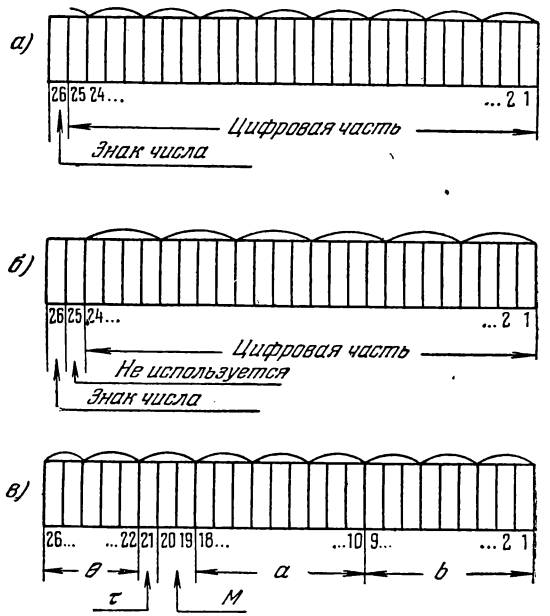


Рис. 9.1. Распределение разрядов ячейки памяти: *а*) для двоичного числа; *б*) для двоично-десятичного числа; *в*) для команды.

На рис. 9.1 указаны объединения двоичных разрядов в восьмеричные и десятичные цифры, применяемые при записи.

Система операций и команд

Все операции в машине *Днепр* разбиты на два типа: арифметические и логические операции (операции первого типа) и операции управления (операции второго типа).

К числу арифметических и логических операций относятся: сложение, вычитание, вычитание модулей, умножение, деление, сдвиг арифметический, сдвиг логический, логическое умножение,

Таблица 9.1. Система арифметических и логических операций и команд

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения	Условия выработки $\varphi = 1$
01	Сложение $01\ \tau\ M\ a\ b$	$s := s + a$, если $M = 0$; $s := s + a$, $b := s$, } если $M = 1$	Числа «s» и «a» алгебраически складываются. Результат операции остается в сумматоре и записывается в ячейку b, если $M = 1$; остается только в сумматоре, если $M = 0$	$\lvert s \rvert \geqslant 1$
		$s := a + b$, если $M = 2$; $s := a + b$, $b := s$, } если $M = 3$	Числа «a» и «b» алгебраически складываются. Результат операции остается в сумматоре и записывается в ячейку b, если $M = 3$; остается только в сумматоре, если $M = 2$. (При сложении двух нулей с разными знаками получается отрицательный нуль.)	
02	Вычитание $02\ \tau\ M\ a\ b$	$s := a - s$, если $M = 0$; $s := a - s$, $b := s$, } если $M = 1$	Из числа «a» вычитается число «s». Результат остается в сумматоре и записывается в ячейку b, если $M = 1$; остается только в сумматоре, если $M = 0$	$\lvert s \rvert \geqslant 1$

Продолжение табл. 9.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения	Условия выработки $\varphi = 1$
02	Вычитание $02\ \tau\ M\ a\ b$	$s := a - b$, если $M = 2$; $s := a - b$, $b := s$, } если $M = 3$	Из числа «a» вычитается число «b». Результат операции остается в сумматоре и записывается в ячейку b, если $M = 3$; остается только в сумматоре, если $M = 2$	$\lvert s \rvert \geqslant 1$
03	Вычитание модулей $03\ \tau\ M\ a\ b$	$s := \lvert a \rvert - \lvert s \rvert$, если $M = 0$; $s := \lvert a \rvert - \lvert s \rvert$, $b := s$, } если $M = 1$	Из числа $\lvert$ «a» $\rvert$ вычитается число $\lvert$ «s» $\rvert$. Результат операции остается в сумматоре и записывается в ячейку b, если $M = 1$; остается только в сумматоре, если $M = 0$	
		$s := \lvert a \rvert - \lvert b \rvert$, если $M = 2$; $s := \lvert a \rvert - \lvert b \rvert$, $b := s$, } если $M = 3$	Из числа $\lvert$ «a» $\rvert$ вычитается $\lvert$ «b» $\rvert$. Результат операции остается в сумматоре и записывается в ячейку b, если $M = 3$; остается только в сумматоре, если $M = 2$	
04	Умножение без округления $04\ \tau\ M\ a\ b$	$s := s \cdot a$, если $M = 0$; $s := s \cdot a$, $b := s$, } если $M = 1$	Число «s» умножается на число «a». Результат операции в прямом коде остается в сумматоре и записывается в ячейку b, если $M = 1$; остается только в сумматоре, если $M = 0$	

Продолжение табл. 9.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения	Условия выработки φ = 1
04	Умножение без округления 04 τ *M a b*	$s := a \cdot b$, если $M = 2$; $s := a \cdot b$, $b := s$, } если $M = 3$	Число «*a*» умножается на число «*b*». Результат операции (в прямом коде) остается в сумматоре и записывается в ячейку *b*, если $M = 3$; остается только в сумматоре, если $M = 2$	
05	Деление 05 τ *M a b*	$s := a : s$, если $M = 0$; $s := a : s$, $b := s$, } если $M = 1$	Число «*a*» делится на число «*s*». Результат операции (в прямом коде) остается в сумматоре и записывается в ячейку *b*, если $M = 1$; остается только в сумматоре, если $M = 0$	$\lvert s \rvert \geqslant 1$ или $s = 0$
		$s := a : b$, если $M = 2$; $s := a : b$, $b := s$, } если $M = 3$	Число «*a*» делится на число «*b*». Результат операции (в прямом коде) остается в сумматоре и записывается в ячейку *b*, если $M = 3$; остается только в сумматоре, если $M = 2$	$\lvert s \rvert \geqslant 1$ или $b = 0$

Продолжение табл. 9.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения	Условия выработки $\varphi = 1$
06	Пересылка числа из ячейки ОЗУ в ячейку ОЗУ 06 0 1 *a b*	$s := a$, $b := s$	Число из ячейки *a* ОЗУ переносится в ячейку *b* ОЗУ	
	Пересылка числа из ПЗУ в ячейку ОЗУ 06 1 1 *a b*	$s := a$, $b := s$	Число из ячейки *a* ПЗУ переносится в ячейку *b* ОЗУ	
10	Логический сдвиг 10 0 0 *a* 000	$s_i := \begin{cases} s_{i-n}, \text{ если } 1 \leqslant i-n \leqslant 26; \\ 0, \text{ если } i-n < 1 \text{ или } i-n > 26 \end{cases}$ ($i = 1,2,\ldots, 26$; n—целое число, находящееся в младших шести разрядах ячейки *a*)	Содержимое сумматора сдвигается на число разрядов, равное целому числу, находящемуся в шести младших разрядах ячейки *a*. Сдвиг выполняется влево, если «*a*»>0, и вправо, если «*a*»<0. Результат операции остается в сумматоре	

Продолжение табл. 9.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Пояснения	Условия выработки $\varphi = 1$
10	Логический сдвиг	10 0 1 *a b*	$s_i := \begin{cases} s_{i-n}, & \text{если } 1 \leqslant i-n \leqslant 26, \\ 0, & \text{если } i-n < 1 \text{ или } i-n > 26 \end{cases}$ ($1 \leqslant i \leqslant 26$; n — целое число, находящееся в младших шести разрядах ячейки *a*); $b := s$	Операция выполняется аналогично предыдущей с той лишь разницей, что результат операции остается в сумматоре и записывается в ячейку *b*	
		10 0 2 *a b*	$s_i := \begin{cases} b_{i-n}, & \text{если } 1 \leqslant i-n \leqslant 26, \\ 0, & \text{если } i-n < 1 \text{ или } i-n > 26 \end{cases}$ ($1 \leqslant i \leqslant 26$; n — целое число, находящееся в младших шести разрядах ячейки *a*)	Содержимое ячейки *b* сдвигается на число разрядов, равное целому числу, находящемуся в шести младших разрядах ячейки *a*. Сдвиг производится влево, если «*a*» > 0, и вправо, если «*a*» < 0. Результат операции остается в сумматоре	
		10 0 3 *a b*	$s_i := \begin{cases} b_{i-n}, & \text{если } 1 \leqslant i-n \leqslant 26, \\ 0, & \text{если } i-n < 1 \text{ или } i-n > 26 \end{cases}$ ($1 \leqslant i \leqslant 26$; n — целое число, находящееся в младших шести разрядах ячейки *a*); $b := s$	Операция выполняется аналогично предыдущей с той лишь разницей, что результат операции остается в сумматоре и записывается в ячейку *b*	

Продолжение табл. 9.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Пояснения	Условия выработки φ = 1
11	Арифметический сдвиг	11 0 0 *a* 000	$s_i := \begin{cases} s_{i-n}, & \text{если } 1 \leqslant i-n \leqslant 25, \\ 0, & \text{если } i-n < 1 \text{ или } i-n > 25 \end{cases}$ ($1 \leqslant i \leqslant 25$; n — целое число, находящееся в младших шести разрядах ячейки *a*)	Содержимое цифровых разрядов сумматора сдвигается на число, равное целому числу, находящемуся в младших шести разрядах ячейки *a*. Сдвиг производится влево, если «*a*» > 0, и вправо, если «*a*» < 0. Результат операции остается в сумматоре	
		11 0 1 *a b*	$s_i := \begin{cases} s_{i-n}, & \text{если } 1 \leqslant i-n \leqslant 25, \\ 0, & \text{если } i-n < 1 \text{ или } i-n > 25 \end{cases}$ ($1 \leqslant i \leqslant 25$; n — целое число, находящееся в младших шести разрядах ячейки *a*) $b := s$	Операция выполняется аналогично предыдущей с той лишь разницей, что результат операции остается в сумматоре и записывается в ячейку *b*	
		11 0 2 *a b*	$s_i := \begin{cases} b_{i-n}, & \text{если } 1 \leqslant i-n \leqslant 25, \\ 0, & \text{если } i-n < 1 \text{ или } i-n > 25 \end{cases}$ ($1 \leqslant i \leqslant 25$; n — целое число, находящееся в младших шести разрядах ячейки *a*)	Содержимое цифровых разрядов ячейки *b* сдвигается на число разрядов, равное целому числу, находящемуся в шести младших разрядах ячейки *a*. Сдвиг производится влево, если «*a*» > 0, и вправо, если «*a*» < 0. Результат операции остается в сумматоре	

Продолжение табл. 9.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Пояснения	Условия выработки $\varphi = 1$
	Арифметический сдвиг	11 0 3 *a b*	$s_i := \begin{cases} b_{i-n}, \text{ если } 1 \leqslant i-n \leqslant 25, \\ 0, \text{ если } i-n < 1 \text{ или } \\ \quad i-n > 25 \end{cases}$ ($1 \leqslant i \leqslant 25$; n — число, находящееся в младших шести разрядах ячейки *a*); $b := s$	Операция выполняется аналогично предыдущей операции с той лишь разницей, что результат операции остается в сумматоре и записывается в ячейку *b*	
	Нормализация числа	12 0 0 *a b*		Число «*s*» нормализуется. Мантисса нормализованного числа записывается в ячейку *a*, модуль порядка — в младшие шесть разрядов ячейки *b*, знак порядка — в знаковый разряд ячейки *b*	
		12 0 1 *a b*		Число «*s*» нормализуется. Мантисса нормализованного числа остается в сумматоре, модуль порядка записывается в младшие шесть разрядов ячейки *b*, знак порядка — в знаковый разряд ячейки *b*	

Продолжение табл. 9.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения	Условия выработки φ = 1
	Нормализация числа: 12 0 2 *a b*		Число «*a*» нормализуется. Мантисса нормализованного числа записывается в ячейку *a*, модуль порядка — в младшие шесть разрядов ячейки *b*, знак порядка — в знаковый разряд ячейки *b*	
	Нормализация числа: 12 0 3 *a b*		Число «*a*» нормализуется. Мантисса нормализованного числа остается на сумматоре, модуль порядка — в младшие шесть разрядов ячейки *b*, знак порядка — в знаковый разряд ячейки *b*	
13	Логическое умножение 13 τ *M a b*	$s_i := s_i \wedge a_i$, если $M = 0$ $(i = 1,2,\ldots, 26)$; $s_i := s_i \wedge a_i$ $(i = 1,2,\ldots, 26)$, $b := s$, } если $M = 1$	Содержимое сумматора поразрядно логически умножается на содержимое ячейки *a*. Результат операции остается в сумматоре и записывается в ячейку *b*, если $M = 1$, и остается в сумматоре, если $M = 0$	

Продолжение табл. 9.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения	Условия выработки $\varphi = 1$
		$s_i := b_i \wedge a_i$, если $M = 2$ ($i = 1,2,\ldots, 26$); $s_i := a_i \wedge b_i$ ($i = 1,2,\ldots, 26$), если $M = 3$; $b := s$,	Содержимое ячейки a поразрядно логически умножается на содержимое ячейки b. Результат операции остается в сумматоре и записывается в ячейку b, если $M = 3$; и остается в сумматоре, если $M = 2$	
14	Поразрядное отрицание равнозначности 14 т M a b	$s_i := s_i \overline{\sim} a_i$, если $M = 0$ ($i = 1,2,\ldots, 26$); $s_i := s_i \overline{\sim} a_i$ ($i = 1,2,\ldots, 26$), если $M = 1$; $b := s$,	Производится поразрядное отрицание равнозначности содержимого сумматора с содержимым ячейки a. Результат операции остается в сумматоре и записывается в ячейку b, если $M = 1$, и остается в сумматоре, если $M = 0$	
		$s_i := a_i \overline{\sim} b_i$, если $M = 2$ ($i = 1,2,\ldots, 26$); $s_i := a_i \overline{\sim} b_i$ ($i = 1,2,\ldots, 26$), если $M = 3$; $b := s$,	Производится поразрядное отрицание равнозначности содержимого ячейки a с содержимым ячейки b. Результат операции остается в сумматоре и записывается в ячейку b, если $M = 3$, и остается в сумматоре, если $M = 2$	•

Продолжение табл. 9.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения	Условия выработки $\varphi = 1$
27	Пересылка из сумматора в ячейку памяти 27 0 0 0 1 p b	$b := s$	Содержимое сумматора переносится в ячейку b блока ОЗУ с номером p	
	Пересылка из ячейки памяти в сумматор 27 0 1 0 m p b	$s := b$	Содержимое ячейки b блока ОЗУ с номером p, если $m = 1$, и блока ПЗУ с номером p, если $m = 0$, переносится в сумматор	
31	Запись на магнитную ленту 31 0 0 $\bar{n}$ b		Числа «b», «$b + 1$», ..., «$b + n$» из ОЗУ переносятся на зону магнитной ленты, номер которой стоит в данный момент времени под считывающими — записывающими головками (здесь $\bar{n}$ — дополнение n до 777_8, т. е. $\bar{n} + n = 777_8$)	
	Чтение с магнитной ленты 31 0 1 $\bar{n}$ b		n чисел с зоны магнитной ленты, номер которой стоит в данный момент времени под считывающими—записывающими головками, переносятся соответственно в ячейки b, $b + 1$,..., $b + n$ ОЗУ (здесь $\bar{n}$ — дополнение n до 777_8, т. е. $\bar{n} + n = 777_8$)	

Таблица 9.2. Система операций управления и команд

Код операции θ		Название операции и вид команды	Краткое содержание команды	Пояснения
21	Прием — выдача	Перенос показаний счетчика времени в ячейку памяти 21 0 0 001 *b*	$b := h$	Содержимое счетчика времени h переносится в ячейку b ОЗУ
		Перенос числа из регистра ввода в ячейку памяти 21 0 0 002 *b*	$b := R^{(1)}$	Содержимое регистра ввода числа $R^{(1)}$ переносится в ячейку b ОЗУ
		Перенос числа из регистра признака печати в ячейку памяти 21 0 0 003 *b*	$b := R^{(2)}$	Содержимое регистра признака печати $R^{(2)}$ УСО переносится в ячейку b ОЗУ
		Перенос числа из регистра режимов в ячейку памяти 21 0 0 004 *b*	$b := R^{(3)}$	Содержимое регистра режимов $R^{(3)}$ УСО переносится в ячейку b ОЗУ

Продолжение табл. 9.2

Код операции θ		Название операции и вид команды	Краткое содержание команды	Пояснения
21	Прием — выдача	Перенос числа из ячейки памяти в регистр режимов 21 0 1 001 *b*	$R^{(3)} := b$	Содержимое ячейки *b* ОЗУ переносится в регистр режимов $R^{(3)}$ УСО
		Перенос из ячейки памяти в выходной регистр 21 0 1 002 *b*	$R^{(4)} := b$	Содержимое ячейки *b* ОЗУ переносится в выходной регистр $R^{(4)}$ УСО
		Опрос датчика непрерывного сигнала 21 0 2 *a b*		Производится опрос датчика непрерывного сигнала с номером *a* ($a \geqslant 6$). В узле аналого-дискретного преобразования УСО снятый в датчике непрерывный сигнал преобразуется в число и записывается в ячейку *b* ОЗУ. Точность преобразований 0,4%

Продолжение табл. 9.2

Код операции θ	Название операции и вид команды		Краткое содержание команды	Пояснения
22	Включение — выключение реле	22 0 0 000 *b*		Выключается реле, номер которого указан в ячейке *b* ОЗУ
		22 0 1 *a* 000		Выключается реле с номером *a* ($0 \leqslant a \leqslant 30_{10}$)
		22 0 2 000 *b*		Включается реле, номер которого указан в ячейке *b* ОЗУ
		22 0 3 *a* 000		Включается реле с номером *a*

Продолжение табл. 9.2

Код операции b	Название операции и вид команды		Краткое содержание команды	Пояснения
23	Условный переход по признаку	Условный переход по признаку ω_1 23 0 1 *птр b*	$C:=k_{II}$, если $\omega_1=1$; $C:=C+1$, если $\omega_1=0$ (вид памяти выбирается в зависимости от значения m)	Машина выполняет: а) передачу управления команде «*b*», если $\omega_1=1$, и следующей команде (команде с номером $k+1$), если $\omega_1=0$ (ячейка *b* относится к блоку ОЗУ с номером *p*, если $m=1$ или 3, и к блоку ПЗУ с номером *p*, если $m=0$ или 2); б) включает в работу блок констант с номером *n* и блок рабочих ячеек с номером *m* (*n*, *m*, *p*— восьмеричные цифры)
		Условный переход по признаку ω_2 23 0 2 *птр b*	$C:=k_{II}$, если $\omega_2=1$; $C:=C+1$, если $\omega_2=0$ (вид памяти выбирается в зависимости от значения m)	Машина выполняет: а) передачу управления команде «*b*», если $\omega_2=1$, и следующей команде (команде с номером $k+1$), если $\omega_2=0$ (ячейка *b* относится к блоку ОЗУ с номером *p*, если $m=1$ или 3, и к блоку ПЗУ с номером *p*, если $m=0$ или 2); б) включает в работу блок констант с номером *n* и блок рабочих ячеек с номером *m* (*n*, *m* и *p*— восьмеричные цифры)

Продолжение табл. 9.2

Код операции θ	Название операции и вид команды		Краткое содержание команды	Пояснения
23	Условный переход по признаку	Условный переход по признаку φ 23 0 3 *nmp b*	$C:=k_{II}$, если $\varphi=1$; $C:=C+1$, если $\varphi=0$ (вид памяти выбирается в завимости от m)	Машина выполняет: а) передачу управления команде «*b*», если $\varphi=1$, и следующей команде (команде с номером $k+1$), если $\varphi=0$ (ячейка b относится к блоку ОЗУ с номером p, если $m=1$ или 3, и к блоку ПЗУ с номером p, если $m=0$ или 2); б) включает в работу блок констант с номером n и блок рабочих ячеек с номером m (n, m и p — восьмеричные цифры)
		Условный переход по признаку печати ω_3 23 1 1 *nmp b*	$C:=k_{II}$, если $\omega_3=1$; $C:=C+1$, если $\omega_3=0$ (вид памяти выбирается в зависимости от m)	Машина выполняет: а) передачу управления команде «*b*», если $\omega_3=1$, и следующей команде (команде с номером $k+1$), если $\omega_3=0$ (ячейка b относится к блоку ОЗУ с номером p, если $m=1$ или 3, и к блоку ПЗУ с номером p, если $m=0$ или 2); б) включает в работу блок констант с номером n и блок рабочих ячеек с номером m (n, m и p — восьмеричные цифры)

Продолжение табл. 9.2

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения
24	Условный переход по ключу 24 τ *M пгр b*	$C:=k_{\text{П}}$, если $N=\tau M$; $C:=C+1$, если $N\neq\tau M$ Здесь N — номер ключа, включенного на пульте управления (вид памяти выбирается в зависимости от m)	Машина выполняет: а) передачу управления команде «*b*», если на пульте управления включен ключ с номером, совпадающим с числом τM, и следующей команде (команде с номером $k+1$) в противном случае (ячейка *b* относится к блоку ОЗУ с номером p, если $m=1$ или 3, и к блоку ПЗУ, если $m=0$ или 2); б) включает в работу блок констант с номером n и блок рабочих ячеек с номером m (n, m и p — восьмеричные цифры)

Продолжение табл. 9.2

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения
25	Безусловный переход I 25 0 1 *nmp b*	$C:=k_{II}$ (вид памяти выбирается в зависимости от m)	Машина выполняет: а) передачу управления команде «*b*» блока ОЗУ с номером p, если $m=1$ или 3, и блока ПЗУ с номером p, если $m=0$ или 2; б) включает в работу блок констант с номером n и блок рабочих ячеек с номером m (n, m и p — восьмеричные цифры)
	Безусловный переход II 25 0 0 01 *p b*	$C:=k_{II}$ (вид памяти выбирается в зависимости от m)	Машина выполняет: а) передачу управления команде «β» блока ОЗУ с номером γ, если $\sigma=1$ или 3, и блока ПЗУ с номером γ, если $\sigma=0$ или 2; б) включает в работу блок констант с номером α и блок рабочих ячеек с номером σ. Здесь β, γ, σ и α — части команды «*b*» = 00 0 0 ασγ β (p, α, σ, γ — восьмеричные цифры)

Продолжение табл. 9.2

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения
26	Формирование возврата 26 0 0 01*p b*		В ячейку *b* блока ОЗУ с номером *p* записывается «команда» 00 0 0 α β γ *k*+1 где α, β и γ — номера блока констант, блока рабочих ячеек и блока ОЗУ, если β=1 или 3, и ПЗУ, если β=0 или 2, работающих в момент выполнения команды «формирование возврата»; *k* — номер команды «формирование возврата». (Здесь *p*, α, β и γ — восьмеричные цифры.)
	Безусловный переход с возвратом (операция по фиксатору) 26 0 1 01*p b*		Выполняется команда «*b*» блока ОЗУ с номером *p* и передается управление команде «*k*+1», где *k* — номер команды «безусловный переход с возвратом»

Продолжение табл. 9.2

Код операции θ	Название операции и вид команды	Краткое содержание команды	Пояснения
30	Поиск зоны магнитной ленты 30 0 00 *n*		Производится подвод зоны магнитной ленты с номером *n* под «считывающие — записывающие» головки лентопротяжного механизма. Подвод зоны магнитной ленты осуществляется одновременно с выполнением других команд программы, не обращающихся к магнитной ленте. В начальный момент операции вырабатывается признак $\omega_4 = 1$ и после окончания — $\omega_4 = 0$
34	Вывод на печать 34 0 0 0*mp b*		Содержимое ячейки *b* блока ОЗУ с номером *p*, если $m = 1$, и блока ПЗУ с номером *p*, если $m = 0$, выдается на печатающее устройство и вырабатывается значение признака ω_3, равное единице. После окончания выполнения команды признак ω_3 принимает значение, равное нулю (*m*, *p* — восьмеричные цифры)
37	Останов 37 0 0 000 000		Машина останавливается

логическое сложение, нормализация, пересылка, запись на магнитную ленту, чтение с магнитной ленты (операции сложения, вычитания и вычитания модулей выполняются над числами в обратном коде, результат этих операций, получаемый в сумматоре, — число в обратном коде. Все остальные операции выполняются над числами в прямом коде, результат операции — число в прямом коде).

В табл. 9.1 приведены операции и команды первого типа.

При выполнении операций первого типа в АУ вырабатываются признаки ω_1, ω_2 и φ.

Признак $\omega_1 = 1$ вырабатывается во всех операциях, кроме операции «арифметический сдвиг», если результат операции — число отрицательное; при выполнении операции «арифметический сдвиг» вырабатывается признак $\omega_1 = 1$, если результат операции равен ± 0.

Признак $\omega_2 = 1$ вырабатывается во всех операциях, кроме операции «арифметический сдвиг», если результат операции — положительное число; при выполнении операции «арифметический сдвиг» вырабатывается признак $\omega_2 = 1$, если результат операции отличен от числа ± 0 (таким образом, признак $\omega_2 = \bar{\omega}_1$). В начальный момент выполнения команды «поиск зоны магнитной ленты» вырабатывается признак $\omega_4 = 1$, и после окончания ее выполнения $\omega_4 = 0$. В начальный момент выполнения команды «выдача на печать» вырабатывается признак $\omega_3 = 1$ и после окончания ее выполнения $\omega_3 = 0$.

Время выполнения основных арифметических операций: сложение (вычитание) — 29,5 ÷ 57,5 *мксек*; умножение — 31,5 ÷ ÷ 395 *мксек*; деление 379,5 ÷ 407,5 *мксек*.

В табл. 9.2 приведены операции и команды второго типа. В этой таблице используются следующие обозначения: C — регистр номера команды; h — счетчик времени; $R^{(1)}$ — регистр ввода; $R^{(2)}$ — регистр печати; $R^{(3)}$ — регистр режимов; $R^{(4)}$ — выходной регистр; k — номер описываемой команды.

Пульт управления вычислительной части машины

Пульт управления состоит из визуальной панели и панели управления.

На визуальной панели распределяются ряды неоновых лампочек, отражающих содержимое сумматора, регистра команд, регистра (счетчика) номера команд. Отдельные лампочки реги-

стрируют значение признака φ, останов машины, обращение к магнитной ленте и обращение к печатающему устройству.

На панели управления имеются следующие кнопки и клавиши:

Клавишный регистр набора команды, выполненный в виде матрицы из 8 рядов клавиш: нижний ряд — клавиши с цифрой нуль, следующий ряд — клавиши с цифрой единица и т. д., верхний ряд — клавиши с цифрой семь; каждый ряд содержит по 9 клавиш, из которых три крайние справа (белые) соответствуют второму адресу команды, следующие три (красные) соответствуют первому адресу и три левые (черные) соответствуют коду операции, признаку τ вида памяти и коду модификации. На этом регистре набирается восьмеричное число, которое следует записать на сумматор или в ячейку памяти.

Кнопка «ВЗ». При нажатии этой кнопки происходит сброс клавиш регистра набора команды.

Клавишный регистр номера команды. Этот регистр также выполнен в виде матрицы из 8 рядов клавиш, таких же как и в клавишном регистре набора команды; каждый ряд имеет по 6 клавиш: правые три (белые) и левые три (красные); с помощью этих клавиш можно задать начальный номер команды программы, т. е. номер команды, с которой следует начать выполнение программы, и номер блока памяти, из которого должны выбираться команды и числа в момент начала выполнения программы. Номера блоков памяти и номер команды набирается на этом регистре точно так же, как они записываются в команде с кодом операции 25 и кодом модификации 1, при этом белые клавиши соответствуют второму адресу команды, а красные — первому адресу команды. На том же регистре набирается начальный адрес вводимого массива чисел (команд) в машину с перфоленты. Номер ячейки набирается на белых клавишах; номер блока — на правых клавишах из числа красных; вид памяти — на средней линейке красных клавиш. Блоку ОЗУ соответствует 1.

Кнопка «Установ адреса». При нажатии на эту кнопку число, набранное на регистре номера команды пульта управления, передается в регистр номера команды устройства управления.

Кнопка «Ввод». При нажатии на эту кнопку производится ввод в машину, начиная с ячейки, номер которой указан в регистре номера команды устройства управления, и выполняется

контрольное суммирование. После чего машина останавливается, а контрольная сумма сохраняется на сумматоре.

Клавишный регистр «Останов по адресу». Выполнен в виде клавишной матрицы из 8 рядов клавиш, таких же как и в регистре набора команды, каждый ряд содержит 4 клавиши: правые три (белые), на которых набирается номер команды, с тем, чтобы при выполнении этой команды произошел останов машины и одна (красная), предназначенная для набора блока памяти (ОЗУ или ПЗУ). Вид памяти указывается путем нажатия на кнопку «ОЗУ» или «ПЗУ».

Кнопка «Останов».

Кнопка «Пуск».

Кнопка «Вызов на сумматор». При нажатии этой кнопки содержимое ячейки, номер которой указан на клавишном регистре номера команды, переносится в сумматор.

Кнопка «Запись в ячейку». При нажатии на эту кнопку содержимое клавишного регистра набора команды переносится в ячейку, номер которой указан на клавишном регистре набора номера команды.

Регистр набора номера ключа, состоящий из 7 клавиш с номерами $1 \div 7$. На этом регистре включается одна из клавиш (номер ключа), по которому следует выполнить в программе условный переход по ключу.

Пульт оператора (пульт управления УСО)

Пульт оператора состоит из визуальной панели и панели управления.

Таблица 9.3. Соответствие разрядов

Название регистра	Раз										
	26	25	24	23	22	21	20	19	18	17	16
Выходной регистр										×	×
Регистр ввода числа	×	×	×	×	×	×	×	×	×	×	×
Регистр признака печати											
Регистр режимов											
Счетчик времени					×	×	×	×	×	×	×

На панели управления имеется:

Регистр набора числа, представляющий собой клавишную матрицу из 10 рядов клавиш: нижний ряд имеет клавиши с цифрой нуль, следующий — с цифрой единица и т. д., верхний — с цифрой 9, каждый такой ряд имеет 5 клавиш белых и одну черную, и клавиш «+» и «—», дополняющих матрицу до полного числового регистра. На этом регистре можно набрать 5-разрядное десятичное число (на белых клавишах) и признак (на черных клавишах), характеризующий число.

Регистр вызова на печать, выполненный в виде клавишной матрицы из трех рядов клавиш; один ряд имеет клавиши с номерами 1, 2, ... , 10, второй ряд — с номерами 11, 12, ... , 20 и третий ряд — с номерами 21, 22, ... , 30. На этом регистре записывается (путем нажатия клавиши) любой из 30 кодов вызова на печать на регистр признаков внеочередной печати регистра печати, расположенного в УСО.

Регистр режима, состоящий из шести клавиш с номерами 0, 1, ... , 5. На этом регистре набирается один из шести допустимых режимов (под режимом понимается переход от одной подпрограммы к другой. Таких программ 6).

Клавиши «Одиночный», «Импульс» и *«Пуск счетчика времени»*. Клавиши «Одиночный» и «Импульс» используются для ввода десятичного числа, набранного на регистре набора числа. При этом соблюдается постоянный порядок нажатия этих клавиш: сперва «Одиночный», затем «Импульс». Клавиша «Пуск счетчика времени» используется для включения в работу счетчика времени.

регистров УСО разрядам ячейки памяти

ряды															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
										×	×	×	×	×
												×	×	×
×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

На *визуальной панели* имеется два ряда неоновых лампочек, из которых один состоит из 30 лампочек и предназначен для контроля набранного числа на регистре набора числа (двадцать первых лампочек высвечивают само двоично-десятичное число, а десять других — признак числа), другой состоит из 6 лампочек и предназначен для контроля выбранного режима на регистре режима панели управления.

Кнопка останов.

Часы.

В табл. 9.3 указано соответствие регистров УСО разрядам ячейки памяти.

Подготовка данных. Ввод и вывод

Подлежащие вводу числа (десятичные или восьмеричные) программист записывает на специальных бланках. Образец такого бланка приведен на рис. 9.2.

Номер ячейки памяти	Код операции	ω и код модификации	Первый адрес	Второй адрес	Пояснения
125	+	09	81	04	+0,098104 } десятичные
126	—	15	00	06	—0,150006 } числа
020	01	1	537	105	
021	04	3	017	106	

Рис. 9.2. Образец записи чисел и команд на бланке.

Далее этот материал с помощью телетайпа переносится на перфоленту в виде пробивок. На рис. 9.3 приведены коды цифр и некоторых специальных знаков на перфоленте.

При перфорировании десятичных чисел перед каждым числом следует пробивать *признак десятичного числа*, затем *признак начала числа*. При перфорировании восьмеричных чисел (команд) следует перед числом (командой) пробивать *признак начала числа*. В конце всего материала следует пробивать цифру нуль, признак десятичного числа, признак начала числа, шесть синхронизирующих импульсов (нулей) и *признак конца ввода*.

Ввод осуществляется с пульта управления. Для этого следует:

а) нажать на пульте управления кнопку «Начальная установка»;

б) на регистре номера команды пульта управления набрать номер ячейки, в которую следует поместить первое число, и номер блока ОЗУ (рекомендуется всегда начинать ввод с нулевой ячейки нулевого блока, если программа начинается с другой ячейки, то в первых ячейках рекомендуется записывать нули);



Рис. 9.3. Образец кодирования цифр и знаков на перфоленте.

в) нажать кнопку «Установ адреса» (при этом на регистр (счетчик) номера команды и регистр номера блока устройства управления переносится набранный на пульте начальный адрес памяти);

г) включить клавиши «Останов по переполнению 1 и 2»;

д) нажать кнопку «Ввод».

Ввод прекращается как только с перфоленты будет считан признак конца ввода, при этом на сумматоре образуется *контрольная сумма* введенных чисел, полученная по операции сложения с циклическим переносом единицы и из старшего разряда в младший.

Вывод из машины производится по команде с кодом операции 34. При этом печатающее устройство может работать в двух режимах: в режиме печати букв и в режиме печати цифр. Конкретно режим задается в 26 разряде ячейки, содержимое которой выводится на печать. Кроме того, в разрядах 19 ÷ 25 этой же ячейки задаются коды других управляющих операций. В табл. 9.4 приведены управляющие операции устройства вывода и управляющие коды этих операций.

Печатающее устройство в режиме печати цифр печатает знак числа (содержимое 17-го разряда: если содержимое 17-го разряда 1, то печатается минус, если нуль — делается пропуск печати на один разряд, что соответствует знаку плюс) и четыре цифры (первая цифра — содержимое 13 ÷ 16-го разрядов, вторая цифра — 9 ÷ 12-го разрядов, третья цифра —

Таблица 9.4. Управляющие операции устройства вывода

Управляющий код		Наименование операции
Управляющий разряд	Содержимое управляющих разрядов	
26	1	Печать цифр
	0	Печать букв
25	1	Печать красным цветом
	0	Печать черным цветом
24	1	Возврат каретки перед печатью
	0	Нет возврата каретки перед печатью
19 ÷ 22	n	Сдвиг каретки влево на n позиций ($0 \leqslant n \leqslant 15_{10}$)

1 ÷ 4-го разрядов). В режиме печати букв печатается слово из трех букв (первая буква — содержимое 13 ÷ 18-го разрядов, вторая буква — 7 ÷ 12-го разрядов и третья буква — 1 ÷ 6-го разрядов). Кодировка букв приведена в табл. 9.5.

Таблица 9.5. Кодировка букв

Буква	Восьмеричный код	Десятичный код	Буква	Восьмеричный код	Десятичный код
А	20	16	Р	41	33
Б	21	17	С	42	34
В	22	18	Т	43	35
Г	23	19	У	44	36
Д	24	20	Ф	45	37
Е	25	21	Х	46	38
Ё	26	22	Ц	47	39
Ж	27	23	Ч	50	40
З	30	24	Ш	51	41
И	31	25	Щ	52	42
Й	32	26	Ь	53	43
К	33	27	Ъ	54	44
Л	34	28	Ы	55	45
М	35	29	Э	56	46
Н	36	30	Ю	57	47
О	37	31	Я	60	48
П	40	32			

Литература к главе 9: [73].

ГЛАВА 10

ПРОГРАММНО-УПРАВЛЯЕМАЯ МАШИНА *УРАЛ-2*

Общая характеристика машины *Урал-2*

Тип машины	вычислительная		
Класс машины	средняя		
Среднее число основных операций в секунду	5000		
Рабочая система счисления	двоичная		
Форма представления чисел	с плавающей запятой	или с фиксированной запятой	
Длина кода числа	40 разрядов (мантисса со знаком—33 разряда, порядок со знаком — 7 разрядов)	40 разрядов (39 цифровых и один знаковый)	20 разрядов (19 цифровых и один знаковый)
Диапазон представления чисел	$2^{-64} \leqslant \lvert x \rvert \leqslant (1 - 2^{-32})\, 2^{63}$ или	$2^{-39} \leqslant \lvert x \rvert \leqslant 1 - 2^{-39}$ или	$2^{-19} \leqslant \lvert x \rvert \leqslant 1 - 2^{-19}$
Точность вычислений	$9 \div 10$ десятичных значащих цифр	$11 \div 12$ десятичных знаков	или $5 \div 6$ десятичных знаков
Порядок выполнения команд	заданный (естественный)		
Адресность машины	одноадресная		
Виды запоминающих устройств	оперативное запоминающее устройство выполнено на магнитных сердечниках; емкость 4096 20-разрядных или 2048 40-разрядных двоичных кодов; время обращения: для 20-разрядных кодов — 15 *мксек*, для 40-разрядных кодов — 30 *мксек*; накопитель на магнитном барабане:		

	емкость одного барабана 8192 40-разрядных кодов; скорость последовательной записи или считывания 3000 кодов в секунду; среднее время ожидания кода 20 *мсек*; возможно подключение до 8 барабанов; накопитель на магнитной ленте: емкость 100 000 40-разрядных кодов; скорость последовательной записи или считывания 1000 кодов в секунду
Виды ввода и вывода	ввод с перфорированной киноленты: емкость 12 000 40-разрядных кодов; скорость последовательной выборки 150 кодов в секунду; вывод на печатающее устройство со скоростью 20 чисел в секунду и на перфорированную киноленту со скоростью 160 кодов в минуту; предусмотрена возможность подключения к машине до 12 независимых устройств ввода и до 12 устройств вывода
Количество электронных ламп	2280
Потребляемая мощность	25 *квт*
Площадь, необходимая для размещения машины	150 ÷ 200 $м^2$

Машина *Урал-2* является универсальной. Эксплуатируется круглосуточно. Допустимы перерывы в работе с выключением питания. Время восстановления работоспособности — 30 минут.

Краткие сведения об основных устройствах машины

Арифметическое устройство (АУ). Основным блоком АУ является сумматор, разделенный на две части (сумматор чисел и сумматор порядков) и работающий в обратном коде. Сумматор чисел имеет 41 разряд, сумматор порядков — 8 разрядов.

В состав АУ входят также следующие регистры:

регистр I *(входной)* разделен на две части — регистр чисел (40 разрядов) и регистр порядков (7 разрядов);

регистр II *(выходной)* имеет 40 разрядов.

При выполнении операций с фиксированной запятой сумматор порядков и регистр порядков не принимают участия в работе.

При выполнении операций с плавающей запятой работают все регистры и сумматоры АУ. При этом сумматор чисел и регистры чисел (все 40 разрядов каждого из них) выпол-

няют действия с мантиссами чисел, сумматор и регистр порядков — с порядками чисел. Благодаря тому, что сумматор чисел и регистры чисел содержат на 8 разрядов больше, чем требуется для изображения мантиссы, обеспечивается повышенная точность вычислений при выполнении «цепочки» арифметических действий. Округление выполняется только при передаче кода из сумматора в ОЗУ.

Результат операций фиксируется в прямом коде либо в сумматоре чисел (фиксированная запятая), либо и в сумматоре чисел и в сумматоре порядков (плавающая запятая).

При выполнении операций обмена информацией между ОЗУ и НМЛ или ОЗУ и НМБ в АУ производится суммирование передаваемых кодов для целей контроля.

При соответствующих условиях АУ вырабатывает сигнал переполнения φ и сигнал управления условным переходом — ω.

Оперативное запоминающее устройство. ОЗУ выполнено на ферритовых сердечниках. Время обращения к ОЗУ для 20-разрядных кодов 15 *мксек*. Сорокаразрядный код записывается (считывается) в два приема; время обращения 30 *мксек*.

Двадцатиразрядные коды хранятся в так называемых «неполных» ячейках. Коды, состоящие из 40 разрядов, записываются в две «неполные» ячейки, составляющие одну «полную». Неполные ячейки нумеруются подряд от 0000 до 7777 в восьмеричной системе счисления. Полные ячейки имеют четные номера от 0000 до 7776. Для отличия четных неполных ячеек от полных в коде команды имеется специальный разряд, указывающий на признак «полноты» ячейки. Полная ячейка с номером $2k$ состоит из двух неполных с номерами $2k$ и $2k+1$.

Устройство управления (УУ). УУ содержит следующие основные логические элементы:

Регистр-счетчик номера команды («счетчик команд») имеет 12 разрядов. Его содержимое задает номер ячейки ОЗУ, из которой производится пересылка команды на регистр команд для исполнения.

Регистр-сумматор команд («регистр команд») имеет 20 разрядов. Служит для приема исполняемой команды из ОЗУ, для изменения ее по содержимому регистра переадресации и для хранения в процессе исполнения.

Регистр переадресации (иначе «счетчик циклов»). Имеет 12 разрядов. Регистр переадресации служит для автоматического

изменения адресной части исполняемой команды. Совместно с регистром переадресации работает специальный триггер, хранящий значение признака *f*, необходимого для возможности переадресации по полным и по неполным ячейкам.

Длительность стандартного такта (цикла) работы машины 80 *мксек*. Управление осуществляется сериями синхронизирующих импульсов или отдельными импульсами, выдаваемыми задающим генератором.

Накопитель на магнитном барабане (НМБ). Магнитный барабан представляет собой цилиндр, вращающийся со скоростью 1500 оборотов в минуту. Емкость барабана — 8192 40-разрядных или 16 384 20-разрядных двоичных кодов. 40-разрядный код занимает две последовательные 20-разрядные ячейки. Все ячейки перенумерованы подряд с 00000 до 37777 в восьмеричной системе счисления. Считывание (запись) группы кодов можно выполнять, начиная с любой из ячеек. Ячейки расположены зонами по 512 в каждой зоне. Время считывания группы из одной зоны (записи в одну зону) от 90 до 130 *мсек*. На переход от одной зоны к другой затрачивается 40 *мсек*.

Накопитель на магнитной ленте (НМЛ). НМЛ использует в качестве носителя информации кинопленку с ферромагнитным покрытием. Запись производится группами (зонами). НМЛ может содержать 256 зон. Зоны нумеруются подряд от 0000 до 0377 в восьмеричной системе счисления.

Номера зон перфорируются. Максимальная емкость одной зоны 2048 40-разрядных или 4096 20-разрядных двоичных кодов. Емкость ленты при максимальной длине (200—250 *м*) достигает 100 000 40-разрядных двоичных кодов. Лента, склеенная в кольцо, движется в одну сторону. Среднее время поиска произвольной зоны при максимальной длине ленты 70—80 секунд. Скорость ленты 2 *м/сек*.

Устройства ввода и вывода (УВВ). Для ввода используется перфорированная кинолента. Коды располагаются группами (зонами); номера зон: $0000 \div 0377_{(8)}$. Максимальная емкость одной зоны определяется емкостью ОЗУ. Максимальная длина ленты 250—300 *м*; общая емкость 10 000—12 000 40-разрядных кодов.

Считывание фотоэлектрическое. Лента, склеенная в кольцо, движется в одну сторону. Скорость движения ленты 2,8 *м* в секунду. Скорость считывания 150 чисел в секунду. Вре-

мя поиска нужной зоны при максимальной длине ленты составляет в среднем 40—50 *сек.*

Для вывода служат: печатающее устройство, работающее в восьмеричной и десятичной системах счисления со скоростью 20 чисел в секунду, и перфоратор со скоростью перфорации 160 кодов в минуту. При печати десятичных чисел в строке 12 разрядов, из них два разряда помещают знаки мантиссы и порядка, 8 разрядов — мантиссу, 2 — порядок. При печати восьмеричных чисел и команд в строке 14 разрядов (одно число из 40-разрядной ячейки или две команды).

Представление чисел и команд

Схема распределения разрядов полной ячейки при хранении в ней двоичных и двоично-десятичных чисел с фиксированной и с плавающей запятой показана на рис. 10.1.

В неполной ячейке ОЗУ может храниться либо 20-разрядное двоичное число с фиксированной запятой, либо команда (рис. 10.2).

Команда машины *Урал-2* имеет следующий основной формат:

$$\tau_1 \theta \overset{\frown}{a} \tau_2,$$

где τ_1 — признак переадресации; θ — код операции; a — адрес; τ_2 — признак разрядности числа, с которым производится действие (признак длины ячейки).

На приведенных рисунках скобками $\frown$ указаны объединения двоичных разрядов в восьмеричные, десятичные и двоичные цифры, принятые для записи чисел и команд; если используется содержимое полной ячейки, то $\tau_2 = 4$, при использовании содержимого неполной ячейки $\tau_2 = 0$; при $\tau_1 = 1$ перед командой ставится знак минус. Указанная кодировка признаков обусловливается конструктивными особенностями клавишного устройства. В знаковых разрядах чисел 0 обозначает плюс, 1 обозначает минус. Мантиссы и порядки представляются в прямом коде.

Кроме основного формата команды в системе команд применяется также следующий вспомогательный формат:

$$* \overset{\frown}{b} \tau_2,$$

где b занимает 4 ÷ 18 разряды команды и представляет собой номер зоны магнитного барабана, τ_2 — признак длины ячейки.

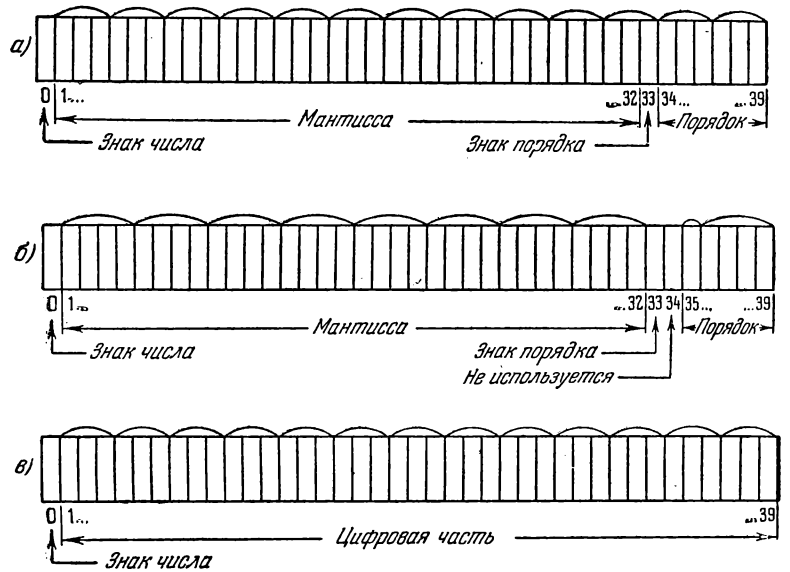


Рис. 10.1. Распределение разрядов полной ячейки: *а*) для двоичного числа с плавающей запятой; *б*) для двоично-десятичного числа; *в*) для 40-разрядного двоичного числа с фиксированной запятой.

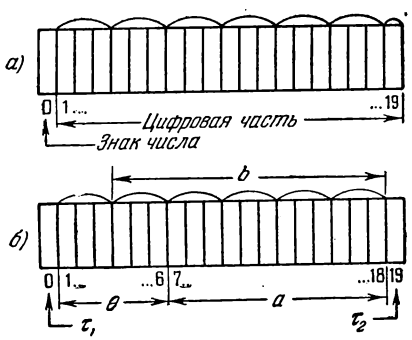


Рис. 10.2. Распределение разрядов неполной ячейки памяти: *а*) для 20-разрядного двоичного числа с фиксированной запятой; *б*) для команды.

Система операций и команд

Система операций и команд приведена в табл. 10.1. В ней помимо введенных ранее (см. стр. 271) приняты следующие обозначения:

«$s^{ч}$» — содержимое сумматора чисел,

«$s^{п}$» — содержимое сумматора порядков,

«s» — содержимое сумматора (сумматора чисел и сумматора порядков, рассматриваемых в совокупности),

«R» — содержимое регистра команд,

f — значение признака типа переадресации (по полным или неполным ячейкам).

Во всех командах таблицы (исключая особо отмеченные) признак τ_1 служит для формирования исполнительного адреса $\tilde{a}$ по правилу:

$$\tilde{a} = a - \tau_1 \cdot «F».$$

Здесь $\tilde{a}$ — исполнительный адрес команды, «F» — содержимое регистра переадресации. (Если $F > a$ и $\tau_1 = 1$, то $\tilde{\theta} = \theta - 1$. В дальнейшем предполагается, что при формировании $\tilde{a}$ код операции θ не меняется.)

В графе 6 табл. 10.1 используются также обозначения q и t, имеющие следующий смысл:

q — число сдвигов на один разряд, выполняющихся без коррекции порядков (сдвиг и выравнивание порядков);

t — число сдвигов на один разряд с коррекцией порядков (нормализация при действиях с плавающей запятой).

Замечания. 1. Нулевой результат операций с фиксированной или плавающей запятой представляется в виде кода, имеющего нули во всех разрядах.

2. При операциях с фиксированной запятой (см. $\theta = 01$, 02, 03, 06, 07) сумматор порядков сбрасывается ($s^{п} = 0$).

3. При операциях с плавающей запятой (см. $\theta = 41$, 43, 44, 46) числа, над которыми производится операция, могут быть ненормализованными; при $\theta = 47$ необходимо, чтобы выполнялось условие $2 \cdot «s_м» < «a_м»$.

4. Перед началом операций обращения к НМБ и НМЛ сумматор автоматически устанавливается в нуль и после выполнения передачи кодов в нем образуется контрольная сумма, полученная по операции с $\theta = 26$. При операциях чтения с НМБ и НМЛ ($\theta = 51$ и 53), если $\tau_2' = 4$, то запись считываемых кодов в ОЗУ не производится; на сумматоре получается контрольная сумма кодов, хранящихся в соответствующих ячейках барабана или в соответствующей зоне магнитной ленты («контрольное считывание»).

Таблица 10.1. Система операций и команд машины *Урал-2*

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
01	Сложение чисел с фиксированной запятой $\tau_1\, 01\, a\, \tau_2$	$s^{ч} := s^{ч} + \tilde{a}$, $s^{п} := 0$	$s < 0$	$\lvert s \rvert > 1 - 2^{-39}$	80	К содержимому сумматора чисел прибавляется содержимое ячейки $\tilde{a}$. Результат фиксируется в сумматоре чисел. Сумматор порядков сбрасывается
02	Посылка в сумматор числа с фиксированной запятой $\tau_1\, 02\, a\, \tau_2$	$s^{ч} := \tilde{a}$, $s^{п} := 0$	$s < 0$		80	Сумматор сбрасывается, затем в сумматор чисел записывается содержимое ячейки $\tilde{a}$. Сумматор порядков сбрасывается
03	Вычитание чисел с фиксированной запятой $\tau_1\, 03\, a\, \tau_2$	$s^{ч} := s^{ч} + \overline{\operatorname{sign} \tilde{a}} \lvert \tilde{a} \rvert$, $s^{п} := 0$	$s < 0$	$\lvert s \rvert > 1 - 2^{-39}$	80	Числу, переданному в АУ из ячейки $\tilde{a}$, присваивается обратный знак, после чего выполняется операция «сложение с фиксированной запятой» (см. $\theta = 01$)

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
06	Умножение чисел с фиксированной запятой $\tau_1\ 06\ a\ \tau_2$	$s^{ч} := s^{ч} \times \tilde{a}$, $s^{п} := 0$	$s < 0$		470 при $\tau_2 = 4$; 330 при $\tau_2 = 0$	Содержимое сумматора чисел умножается на содержимое ячейки $\tilde{a}$. Результат фиксируется в сумматоре чисел. Сумматор порядков сбрасывается
07	Деление чисел с фиксированной запятой $\tau_1\ 07\ a\ \tau_2$	$s^{ч} := s^{ч} : \tilde{a}$, $s^{п} := 0$	$s < 0$	$\lvert s \rvert > 1 - 2^{-39}$ или $\tilde{a} = 0$	810 при $\tau_2 = 4$; 500 при $\tau_2 = 0$	Содержимое сумматора чисел делится на содержимое ячейки $\tilde{a}$. Результат фиксируется в сумматоре чисел. Сумматор порядков сбрасывается
10	Присвоение знака $\tau_1\ 10\ a\ \tau_2$	$s_0^{ч} := \tilde{a}_0$, $s_i^{ч} := \tilde{a}_i$, $1 \leqslant i \leqslant 39$	$s < 0$		80	Содержимому сумматора присваивается знак числа, стоящего в ячейке $\tilde{a}$. Результат фиксируется в сумматоре. Нормализации нет

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
11	Сдвиг τ_1 11 $*$ n τ_2 $\tau_2 = 0$ Сдвиг «логический»	$s^{ч}_i := \begin{cases} s^{ч}_{i+k_{\tilde{n}}} \text{ при} \\ 0 \leqslant i + k_{\tilde{n}} \leqslant 39, \\ 0 \text{ при} \\ 0 > i + k_{\tilde{n}}, \\ i + k_{\tilde{n}} > 39, \end{cases}$ где $\tilde{n} = [(-1)^{\langle k_{12} \rangle} \times$ $\times \sum_{j=13}^{18} \langle k_j \rangle \cdot 2^{18-j} -$ $- \langle F \rangle \cdot r_1,$ $s^{п} := 0$	$s^{ч}_0 = 1$		$130 + 10q$	Содержимое сумматора чисел сдвигается влево или вправо на число разрядов, определяемое значением $\tilde{n}$, занимающим 12 ÷ 18-й разряды. Направление сдвига определяет 12-й разряд команды (вправо — когда в нем 1, влево — когда 0), величину сдвига — 13 ÷ ÷ 18-й разряды. Производится сдвиг всех 40 разрядов сумматора. Сумматор порядков сбрасывается
11	$\tau_2 = 4$ Сдвиг «арифметический»	$s^{ч}_0 := s^{ч}_0,$ $s^{ч}_i := \begin{cases} s^{ч}_{i+k_{\tilde{n}}} \text{ при} \\ 1 \leqslant i + k_{\tilde{n}} \leqslant 39, \\ 0 \text{ при} \\ 1 > i + k_{\tilde{n}}, \\ i + k_{\tilde{n}} > 39, \end{cases}$ где $\tilde{n}$ — то же, что и для $\tau_2 = 0$	$s^{ч}_0 = 1$ так как $s^{ч} := s^{ч} \cdot 2^{\tilde{n}}$	$s^{ч} \geqslant \lvert 1 \rvert$	$130 + 10q$	Производится сдвиг разрядов 1 ÷ 39 сумматора. Содержимое нулевого разряда (знак числа) не меняется. Сумматор порядков сбрасывается

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
12	Выделение части числа τ_1 12 a τ_2	$s_i^{\text{ч}} := s_i^{\text{ч}} \wedge \tilde{a}_i$, $0 \leqslant i \leqslant 39$, $s^{\text{п}} := s^{\text{п}}$	$s^{\text{ч}} = 0$		80	Содержимое сумматора чисел поразрядно логически умножается на «$\tilde{a}$». Результат фиксируется в сумматоре чисел. Содержимое сумматора порядков не изменяется
13	Формирование τ_1 13 a τ_2	$s_i^{\text{ч}} := s_i^{\text{ч}} \vee \tilde{a}_i$, $0 \leqslant i \leqslant 39$, $s^{\text{п}} := s^{\text{п}}$	$s^{\text{ч}} = 0$		80	Содержимое сумматора чисел поразрядно логически складывается с «$\tilde{a}$». Результат фиксируется в сумматоре чисел. Содержимое сумматора порядков не изменяется
14	Сравнение τ_1 14 a τ_2	$s_i^{\text{ч}} := s_i^{\text{ч}} \not\sim \tilde{a}_i$, $0 \leqslant i \leqslant 39$, $s^{\text{п}} := s^{\text{п}}$	$s^{\text{ч}} = 0$		80	Производится поразрядная операция отрицания равнозначности между содержимым сумматора чисел и ячейки $\tilde{a}$. Результат фиксируется в сумматоре чисел. Содержимое сумматора порядков не изменяется

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
16	Посылка в ОЗУ содержимого сумматора с фиксированной запятой τ_1 16 a τ_2	$\tilde{a} := s^{ч}$, $s := s$	$s < 0$		160	В ячейку $\tilde{a}$ записывается содержимое сумматора чисел. Содержимое сумматора порядков и чисел не изменяется
21	Условный переход τ_1 21 a 0	$s := s$; при $\omega = 1$, $C := k_1$; в противном случае $C := C + 1$	$\equiv$		80	Если сигнал $\omega = 1$, то управление передается команде «$\tilde{a}$»; в противном случае выполняется следующая команда
	τ_1 21 a 4	$s := s$; при $\omega = 0$, $C := k_1$; в противном случае $C := C + 1$	$\equiv$		80	Отличается от предыдущей команды тем, что переход в зависимости от значения сигнала ω осуществляется противоположным образом

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
22	Безусловный переход τ_1 22 *a* 0	$C := k_1$, $s := s$	$\equiv$		80	Отличается от предыдущей команды тем, что управление передается команде «$\tilde{a}$» всегда, независимо от значения сигнала ω
22	τ_1 22 *a* 4	$s := s$, $\tilde{a}_{\tau_1} := 0$, $\tilde{a}_0 := 22$, $\tilde{a}_1 := C + 1$, $\tilde{a}_{\tau_2} := 0$, $C := k_1 + 1$	$\equiv$		80	В ячейку $\tilde{a}$ записывается команда возврата вида 0.22. «C» + 1.0 и управление передается команде с номером $\tilde{a} + 1$
23	Передача управления по ключу τ_1 23 $*n*$	$s := s$; При включенном n-м ключе $C := C + 2$, в противном случае $C := C + 1$	$\equiv$		80	Если ключ с номером n, занимающим $16 \div 19$ разряды, включен, то пропускается следующая команда; если он выключен, то выполняется следующая команда

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
24	Конец цикла τ_1 24 a *	$s := s$; при $F \neq 0$ $F := F - 2^f$, $C := k_1$, при $F = 0$ $F := F$, $C := C + 1$	$\equiv$		80	В зависимости от значения признака переадресации f содержимое регистра переадресации уменьшается на единицу или на два (см. $\theta = 25$). Если до этого на регистре переадресации был нуль, то управление передается следующей команде. В противном случае управление передается команде, стоящей в ячейке с номером $\tilde{a}$
25	Начало цикла τ_1 25 a τ_2	$s := s$, $F := k_1$; при $\tau_2 = 0$ $f := 0$, при $\tau_2 = 4$ $f := 1$	$\equiv$		80	На регистр переадресации записывается содержимое адресной части команды. Содержимое 19-го разряда «k_{19}» команды передается на триггер, хранящий значение f. Если $f = 0$, то в дальнейшем, при выполнении команды с

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
						$\theta = 24$, из содержимого регистра переадресации вычитается 1, в противном случае вычитается 2
26	Суммирование τ_1 26 *a* *	$s^{ч} := s^{ч} \oplus \tilde{a}$	$\equiv$		80	К содержимому сумматора чисел прибавляется содержимое неполной ячейки $\tilde{a}$. Суммирование производится в $20 \div 39$-м разрядах сумматора чисел. Единицы переноса передаются в старшие разряды сумматора чисел. Сумматор порядков сбрасывается
27	Посылка в ячейку ОЗУ содержимого регистра переадресации τ_1 27 *a* τ_2	$\tilde{a}_1 := F$, $\tilde{a}_{\tau_1} := 0$, $\tilde{a}_{\tau_1} := f$; при $F \neq 0$	$\equiv$		80	Содержимое регистра переадресации и признака f записывается в ячейку $\tilde{a}$ в виде команды с $\theta = 25$ (при $F \neq 0$) или $\theta = 00$ (при $F = 0$). Если

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
		$\tilde{a}_\theta := 25$, при $F = 0$ $\tilde{a}_\theta := 0$				$\tau_2 = 4$, то в нечетную неполную ячейку, входящую в состав полной ячейки $\tilde{a}$, записывается нуль
30	Изменение команд τ_1 30 a τ_2	$R := [\tilde{a}]_{0k}$, $C := C + 1$; при $C = k + 1$ $R := R \oplus k + 1$	$\equiv$		80	Следующей выполняемой командой является код, получаемый на регистре команд суммированием обратного кода 20-разрядного числа «$\tilde{a}$», со следующей командой, выбираемой из ячейки $k + 1$ (если $\tau_2 = 0$); при $\tau_2 = 4$ $R := [я\ (\tilde{a} + 1)]_{0k}$, если $\tilde{a}$ четное, $R := [я\ (\tilde{a} - 1)]_{0k}$, если $\tilde{a}$ нечетное. Суммирование производится по всем двадцати разрядам регистра-сумматора команд с круговым переносом из нулевого разряда в девятнадцатый

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
32; 33	Выдача результатов либо τ_1 32 a *, либо τ_2 33 a *	При $\theta = 32$ производится выдача содержимого сумматора чисел в десятичной системе счисления. При $\theta = 33$ производится выдача содержимого сумматора чисел в восьмеричной системе счисления	$\equiv$		80 для передачи на приемный регистр соответствующего выходного устройства; задержка возникает, когда, прежде чем произведена выдача, происходит вновь обращение к тому же устройству по команде $\theta = 32$ или $\theta = 33$	Выдача на то или иное устройство определяется значением адресной части команды — $\tilde{a}$: при $\tilde{a} = 0001$ производится печать на бумажной ленте; при $\tilde{a} = 0002$ производится выдача на перфоратор; при $\tilde{a} = 0003$ одновременно работают перфоратор и печатающее устройство. (Для выдачи числа с плавающей запятой его необходимо переслать в сумматор чисел.)

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
34	Интервал * 34 * *		$\equiv$			Бумажная лента, на которой печатаются числа, передвигается на одну строку
37	Останов * 37 * *	Машина останавливается	$\equiv$		80	
40	Посылка в сумматор с входа $\tilde{a}$ τ_1 40 a τ_2		$s < 0$			Код из вводного устройства с номером $\tilde{a}$ посылается в сумматор ($\tilde{a} = 2n$, $0 \leqslant n \leqslant 11$)
41	Сложение чисел с плавающей запятой τ_1 41 a τ_2	$s := s + \tilde{a}$ (при $s^{ч} = 0$ $s := \tilde{a}$, при $\tilde{a}_i = 0$ $1 \leqslant i \leqslant 39$, $s := 3$)	$s < 0$	$\lvert s \rvert \geqslant 2^{63}$	$130 + 10q + 20t$ ($130 \div 1000$)	К содержимому сумматора прибавляется содержимое ячейки $\tilde{a}$. Результат фиксируется в сумматоре
42	Посылка в сумматор числа с плавающей запятой τ_1 42 a τ_2	$s := \tilde{a}$	$s < 0$	$s^{п} < -63$	$80 + 20t$	Сумматор сбрасывается, затем в него записывается число из ячейки $\tilde{a}$. Результат нормализуется всегда, кроме случая $\tilde{a} = \pm 0 \cdot 2^{п}$, при этом $s^{ч} := 0$; $s^{п} := п$

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
43	Вычитание чисел с плавающей запятой τ_1 43 a τ_2	$s := s + \overline{\operatorname{sign} \tilde{a}}\,\lvert a\rvert$	$s < 0$	$\lvert s\rvert \geqslant 2^{63}$	$130 + 10q + 20t$ $(130 \div 1000)$	Числу, переданному в АУ из ячейки $\tilde{a}$, присваивается обратный знак, после чего выполняется операция «сложение с плавающей запятой» (см. $\theta = 41$)
44	Вычитание модулей чисел с плавающей запятой τ_1 44 a τ_2	$s := \lvert s\rvert + [-\lvert \tilde{a}\rvert]$	$s < 0$		$130 + 10q + 20t$ $(130 \div 1000)$	Выполняется операция «сложение с плавающей запятой» ($\theta = 41$) с предварительным изменением знаков: первоначальному содержимому сумматора присваивается знак плюс, числу, передаваемому из ячейки $\tilde{a}$ — знак минус
46	Умножение чисел с плавающей запятой τ_1 46 a τ_2	$s := s \times \tilde{a}$	$s < 0$	$\lvert s\rvert \geqslant 2^{63}$	$470 + 20t$ при $\tau_2 = 4$, $330 + 20t$ при $\tau_2 = 0$	Содержимое сумматора умножается на содержимое ячейки $\tilde{a}$. Результат фиксируется в сумматоре

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
47	Деление чисел с плавающей запятой τ_1 47 a τ_2	$s := s:\tilde{a}$	$s < 0$	$\lvert s \rvert \geqslant 2^{63}$ или $\tilde{a} = 0$	$530 + 20t$ при $\tau_2 = 4$, $810 + 20t$ при $\tau_2 = 0$	Содержимое сумматора делится на содержимое ячейки $\tilde{a}$. Результат фиксируется в сумматоре
56	Посылка в ОЗУ содержимого сумматора с плавающей запятой τ_1 56 a τ_2	$\tilde{a} := s$ $s := s$	$s < 0$	$s^{\text{п}} = 64$	$160 + 20t$	Содержимое сумматора округляется, нормализуется и посылается в ячейку $\tilde{a}$. Округление производится путем прибавления единицы к содержимому 33-го разряда сумматора чисел. После округления 33 ÷ 39-й разряды сумматора чисел устанавливаются в нуль. В сумматоре остается то же число, которое посылается в ячейку $\tilde{a}$. Округление и нормализация не производятся (при сохранении других действий), когда $s = \pm 0{\cdot}2^{\text{п}}$

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
	Обращение к внешним запоминающим устройствам $\tau_1\ \theta\ a'\ \tau_2$ $*\ b\ *$ $*\ *\ a''\ *$		$\equiv$			Выполняется по трем командам, записанным в последовательные ячейки ОЗУ
	$\theta = 50$ Чтение с перфоленты					Производится перепись кодов из зоны b перфоленты в ячейки ОЗУ с номерами, начиная с a' и кончая a''. Если $\tau_2 = 0$, то вводятся 20-разрядные двоичные коды в неполные ячейки. В противном случае вводятся 40-разрядные двоичные коды в полные ячейки. Для ввода двоично-десятичных кодов к номеру зоны b необходимо прибавить 1000 ($b = 00000 \div \div 00377$)

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
	$\theta = 51$ Чтение с магнитного барабана					Производится считывание $a'' - a' + 1$ кодов из магнитного барабана, начиная с ячейки b ($b = 00000 \div 37777$) в неполные ячейки ОЗУ с номерами $a' \div a''$
	$\theta = 52$ Запись на магнитный барабан					На магнитный барабан, начиная с ячейки, имеющей номер b ($b = 00000 \div 37777$), записывается $a'' - a' + 1$ кодов, хранящихся в неполных ячейках ОЗУ с номерами $a' \div a''$
	$\theta = 53$ Чтение с магнитной ленты					Производится перепись $a'' - a' + 1$ кодов из зоны b магнитной ленты в неполные ячейки ОЗУ с номерами $a' \div a''$

Продолжение табл. 10.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
	$\theta = 54$ Запись на магнитную ленту					В зону b магнитной ленты записывается $a'' - a' + 1$ кодов, хранящихся в неполных ячейках ОЗУ с номерами $a' \div a''$
						Исключение. После выполнения всех указанных операций регистр переадресации сбрасывается
00; 20; 04; 31; 05; 35; 06; 36; 15; 45; 17; 55; 57÷77	Не используются		$\equiv$		80	Выполняется один «холостой» такт работы машины

5. Все операции, исключая особо отмеченные, выполняются как над 40-разрядными, так и над 20-разрядными числами. При этом поступающие в АУ или в ОЗУ (для операций 16 и 56) 20-разрядные числа воспринимаются как 40-разрядные, у которых в $20 \div 39$-м разрядах содержатся нули.

6. Если при нечетном $\tilde{a}$ $\tau_2 = 4$, то операция выполняется так же, как и при $\tau_2 = 0$.

Пульт управления

Пульт управления имеет управляющую и сигнальную панели.

На сигнальную панель выведены ряды неоновых лампочек, отображающие содержимое сумматора чисел, сумматора порядков (с разрядами переполнения), регистра команд, счетчика команд, регистра переадресации.

Кроме указанного, на сигнальной панели имеется ряд лампочек, показывающих причины останова, режим работы машины и др. (см. ниже).

На управляющей панели имеется также ряд клавиш, ключей и кнопок, важнейшими из которых являются следующие:

Клавиши для ручного набора команд и переноса их на регистр команд (исполнение набранной команды осуществляется нажатием кнопки «Пуск» или «Однотактный режим»).

Семь *ключей передачи управления* (см. операцию с $\theta = 23$).

Клавиши набора кода в восьмеричной системе счисления (с фиксированной или плавающей запятой) для занесения его в сумматор.

Тумблеры «Блокировка φ*» и «Остановка по* φ*»* для управления работой машины при возникновении сигнала $\varphi = 1$. Если тумблер «Блокировка φ» выключен, а тумблер «Остановка по φ» включен, то машина останавливается; при выключенном тумблере «Остановка по φ» пропускается следующая команда. Если ключ «Блокировка φ» включен, то, независимо от положения второго тумблера, выполняется следующая команда.

Клавиатура для набора адреса, при совпадении которого со счетчиком команд производится останов (останов по заданному адресу).

Кнопки «Пуск» и «Останов» машины.

Кнопка «Однотактный режим», нажатие на которую вызывает выполнение одной команды.

Кнопка «Стирание ОЗУ», при нажатии которой одновременно с кнопкой «Пуск» во все ячейки ОЗУ записывается

пустой код. (Можно стереть часть накопителя, пользуясь тем, что в режиме стирания последовательные адреса возникают на счетчике номера команды; для этого необходимо применить останов «по заданному адресу».)

Кнопка «Начальный пуск», после нажатия на которую производится ввод зоны 00000 перфоленты в неполные ячейки 0000 ÷ 0020 ОЗУ и передача управления команде, стоящей в ячейке 0001.

Ряд кнопок и ключей, относящихся к управлению работой внешних запоминающих устройств (стирание МБ, блокировка стирания МБ, останов при обращениях к МБ, НМЛ, НПЛ, блокировка записи в НМЛ).

Подготовка данных. Ввод и вывод

Числа и команды, предназначенные для пробивки и ввода в машину, записываются на стандартных бланках.

Один из возможных видов бланка с образцами записи двоичных чисел, команд и двоично-десятичных чисел приведен на рис. 10.3. На этом рисунке показано содержимое зоны с номером 000, помещающей программу начального ввода, контрольную сумму, а также двоичные и десятичные числа. Программа осуществляет начальный ввод в полные ячейки с номерами от 0040 до 2420 из зоны с номером 001, суммирование введенных кодов, сравнение с контрольной суммой и передачу управления в ячейку 0040.

Номера зон перед записью увеличиваются на 400 и записываются в трех последовательных строках (по одному восьмеричному разряду в строке). После номера зоны пропускается одна строка. Напомним, кроме того, что знак порядка числа с плавающей запятой входит в младшую восьмеричную цифру мантиссы, а при $\tau_1 = 0$ во второй колонке знак «+» ради простоты не ставится.

Перфорация исходной информации осуществляется с помощью клавишного устройства, соединенного с входным перфоратором.

Образцы пробивки различных кодов на перфоленте показаны на рис. 10.4. Пробивки кодов располагаются по 11 отдельным дорожкам, причем каждая цифра пробивается в четырех строках (исключение составляет пробивка 40-разрядного числа — см. рис. 10.1, *в*).

Номер ячейки	Признак τ_1 (или знак числа)	Код операции	Адрес	Признак τ_2	Знак	Порядок	Пояснения
		3	4				Информация о зоне с номером 000
			0				
			0				
			0				
0000			—				
0001		50	0040				Команды
0002		00	0001				
0003		00	2420				
0004		11	0050				
0005		25	2400				
0006	—	26	2420				
0007		24	0006				
0010		14	0016	4			
0011		21	0040	4			
0012		37					
0013		22	0001				
0014		56	3050	27245		03	Двоичные 40-разрядные числа
0015		60	0000	00001		01	$0{,}75_{(10)} \cdot 2^{-1}$
0016		00	0032	65043		74	Контрольная сумма
0017			3333	3333		00	1/3
0020	—		1549	0729	—	03	Десятичные числа: $-0{,}15490729_{(10)} \cdot 10^{-3}$

Содержимое зоны

Рис. 10.3. Образец записи команд и чисел на бланке.

Имеется также контрольно-считывающее устройство, которое может работать в двух режимах;

а) режим контроля, при котором производится автоматическое сравнение пробивок на двух лентах со скоростью 200

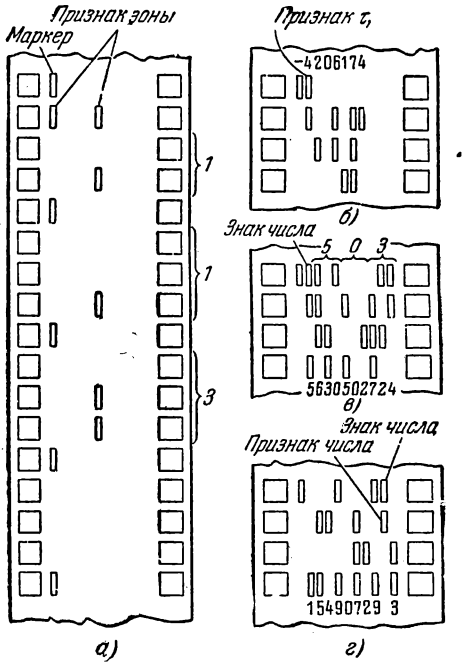


Рис. 10.4. Образцы кодирования на перфоленте: *а*) номера зоны 0113; *б*) команды — 42 0617 4; *в*) 40-разрядного двоичного числа 5630502724503; *г*) десятичного числа 154907293.

кодов в минуту. Если пробивки не совпадают, то устройство останавливается и выдает сигнал;

б) режим реперфорирования, в котором пробивки одной из лент дублируются на другой ленте.

Литература к главе 10: [7].

ГЛАВА 11

ПРОГРАММНО-УПРАВЛЯЕМАЯ МАШИНА *УРАЛ-4*

Машина *Урал-4* построена на основе машины *Урал-2*, от которой отличается некоторыми характеристиками и командами.

Отличия основных характеристик машины *Урал-4* от основных характеристик машины *Урал-2*

Виды запоминающих устройств	оперативное запоминающее устройство то же, что и на машине *Урал-2*; накопитель на магнитном барабане: емкость одного барабана 32 768 20-разрядных кодов, скорость записи или считывания до 12 000 20-разрядных кодов в секунду; в состав машины может быть включено до восьми барабанов; накопитель на магнитной ленте: емкость ленты на одном лентопротяжном блоке 262 144 40-разрядных кодов, скорость последовательной записи или считывания — 2000 40-разрядных кодов в секунду, скорость движения ленты 2 *м/сек*, в состав машины может быть включено 48 лентопротяжных блоков (12 шкафов по 4 блока в каждом), лента движется в обе стороны
Виды ввода и вывода	ввод с перфокарт со скоростью 60 40-разрядных кодов в секунду; ввод с телеграфной ленты со скоростью 400 ÷ 600 алфавитно-цифровых кодов в секунду; вывод на перфокарты со скоростью 30 40-разрядных кодов в секунду; вывод на телеграфную ленту со скоростью 20 алфавитно-цифровых знаков в секунду; вывод на алфавитно-цифровое печатающее устройство, имеющее 128 знаков в строке и 63 различных знака, со скоростью 640 алфавитно-цифровых знаков в секунду (5 строк в секунду)
Количество электронных ламп	около 5000

Потребляемая мощность	40 ÷ 60 *ква*, в зависимости от числа одновременно работающих устройств
Занимаемая площадь	200 ÷ 250 $м^2$

Машина *Урал-4* является универсальной. Может быть использована как информационная. Режим работы машины круглосуточный. Допустимо выключение питания. Время восстановления работоспособности 30 ÷ 60 минут.

Краткие сведения об основных устройствах машины

Арифметическое устройство, устройство управления и оперативное запоминающее устройство в основном не отличаются от тех же устройств машины *Урал-2*. Ниже приводится описание накопителей на магнитном барабане, на магнитной ленте и устройств ввода и вывода.

Накопитель на магнитном барабане (НМБ). Магнитный барабан представляет собой цилиндр, вращающийся со скоростью 750 *об/мин*. Емкость одного барабана 32 768 20-разрядных ячеек. В них может помещаться либо 32 768 20-разрядных кодов, либо, 163 840 десятичных цифр, либо 98 200 алфавитно-цифровых знаков. Запись или считывание могут производиться начиная с любой ячейки магнитного барабана группами, содержащими от одной до 4096 ячеек. Скорость последовательной записи или считывания 12 000 20-разрядных кодов в секунду. Минимальное время поиска первого кода — 15 *мсек*, максимальное — 160 *мсек*. Устройство может содержать до восьми барабанов. Ячейки всех восьми барабанов нумеруются подряд от 000000 до 777777 в восьмеричной системе счисления. При обращении к МБ в АУ автоматически подсчитывается сумма передаваемой информации.

Накопитель на магнитной ленте имеет общую емкость 3 145 728 40-разрядных чисел (31 457 280 десятичных чисел или 18 874 368 алфавитно-цифровых знаков) на 48 лентопротяжных блоках (12 шкафов по четыре лентопротяжных блока в каждом). Скорость записи или считывания 2000 кодов в секунду, возможны считывание и запись начиная с любой строки зоны. В каждом лентопротяжном блоке лента разбита на 32 зоны по 2048 40-разрядных кодов в зоне. Ячейки всех зон перенумерованы подряд от 00000000 до 27777777 в восьмеричной системе счисления.

Лента при поиске зоны движется в обе стороны. Считывание и запись производится при движении в одну сторону. Максимальное время подвода требуемой зоны 40 секунд, при максимальной длине ленты (80 метров).

В НМЛ имеются специальные блоки, позволяющие производить выборочное считывание кодов. Такими блоками являются: регистр ρ признака, помещающий некоторый 40-разрядный код, в котором комбинация значений некоторых разрядов является признаком; регистр r шкалы, имеющий 40 разрядов и помещающий код с нулями только в тех разрядах, в которых на регистре ρ записывается комбинация значений, образующая признак; блок сравнения, позволяющий записывать в ОЗУ только те 40-разрядные коды, поступающие с ленты, которые содержат признак. При этом в процессе считывания производится сравнение содержимого тех одноименных разрядов регистра признака и каждого 40-разрядного кода, поступающего с магнитной ленты, в которых регистр шкалы содержит нули.

Кроме того, для выборочного считывания служат также: регистр δ шага, имеющий 13 двоичных разрядов и позволяющий посылать в блок сравнения не все коды, поступающие с магнитной ленты, а только те, номера ячеек которых на магнитной ленте отличаются друг от друга на «δ»; счетчик π количества просматриваемых кодов, на котором задается число кодов, поступающих с магнитной ленты, имеет 23 разряда; с приходом каждого кода содержимое этого счетчика уменьшается на единицу.

При операциях обращения к магнитной ленте в АУ может автоматически определяться контрольная сумма считываемой или записываемой группы кодов. Для повышения надежности работы магнитной ленты запись значения каждого двоичного разряда производится дважды, с помощью двух магнитных головок на разных участках ленты. При считывании сигналы, поступающие с обоих магнитных головок, попадают на общий усилитель, работающий так, что на его выходе возникает, сигнал, когда хотя бы одна из головок произвела считывание. Благодаря этому повышается вероятность правильной работы магнитной ленты при наличии на ней участков с некачественным ферромагнитным покрытием.

Так как при осуществлении выборочного считывания с магнитной ленты групп кодов произвольной длины применение

контроля по контрольным суммам затруднено, на магнитной ленте контроль правильности записи, считывания и хранения кодов производится по четности. Сорокаразрядный код разбит на 8 групп разрядов, контролируемых независимо.

Устройства ввода и вывода. Для ввода и вывода может быть использовано до 64 или 32 входных и выходных устройств для 20-разрядных или 40-разрядных кодов (соответственно). Ниже перечислены устройства, входящие в комплект машины.

Для ввода используются стандартные 80-колонные или 40-колонные перфокарты и телеграфная лента. Устройство ввода с перфокарт позволяет вводить перфокарты с любым расположением информации. Для этой цели служит специальная коммутационная доска. Правильность ввода каждой перфокарты может контролироваться автоматически. Для этой цели служит сумматор, установленный в клавишном устройстве, который обеспечивает суммирование 11 кодов, пробиваемых на 11 строках перфокарты, и автоматическую перфорацию суммы иа 12-й строке. Контроль может производиться в процессе ввода по специальной программе.

Для ввода с перфоленты предназначен фототрансмиттер, использующий пятидорожечную телеграфную ленту. Скорость ввода 400 ÷ 600 кодов в секунду. Одновременно с вводом может производиться автоматическое преобразование кода СТА-35 в шестизначный код, принятый для выдачи информации на алфавитно-цифровое печатающее устройство.

Для выдачи информации на перфокарты используется выходной перфоратор, работающий со скоростью 30 кодов в секунду (150 карт в минуту). Перфоратор не имеет коммутационной доски, поэтому перфокарта имеет заданное расположение пробивок.

Для выдачи на перфоленту применяется перфоратор типа ПРЛ-20 со скоростью выдачи 20 5 ÷ 7-разрядных кодов в секунду. Выдача производится на семидорожечную бумажную ленту. В комплекте машины два ПРЛ-20, которые могут работать как по отдельности, так и в режиме дублирования.

Для выдачи на печать используется алфавитно-цифровое печатающее устройство (АЦПУ), работающее со скоростью 5 строк в секунду. В одной строке 128 знаков. Печатающее устройство имеет 128 печатающих колес, на каждом из которых 63 различных знака. Подлежащие выводу знаки

передаются в R' — приемный регистр АЦПУ, выполненный в виде оперативного запоминающего устройства на ферритовых сердечниках. Двоичные разряды этого регистра разделены на 128 групп. Каждая группа соответствует определенному печатающему колесу, содержит шесть двоичных разрядов, называется m-м разрядом регистра АЦПУ и обозначается R'_m. Двоичные разряды регистра R', соответствующие R'_m, нумеруются справа налево и обозначаются через R'_{m_i} $(1 \leqslant i \leqslant 6)$. При этом значения младших разрядов шестиразрядных кодов (см. табл. 11.2) помещаются в R'_{m_1}.

Представление чисел и команд

Распределение разрядов полной и неполной ячеек при хранении в них двоичных, двоично-десятичных чисел и команд совпадает с распределением в машине *Урал-2* (см. рис. 10.1). Схемы распределения разрядов полной (40-разрядной) и неполной (20-разрядной) ячеек при хранении алфавитно-цифровых знаков показаны на рис. 11.1. На рис. 11.2 показаны

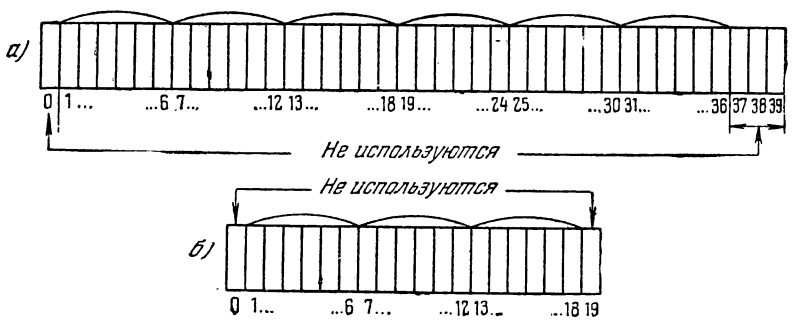


Рис. 11.1. Распределение разрядов при хранении алфавитно-цифровых знаков в *а*) полной и *б*) неполной ячейках.

схемы распределения разрядов полной и неполной ячеек при хранении в них двоично-десятичных чисел с фиксированной запятой. Такая форма представления чисел используется для операций перевода чисел ($\theta = 35{,}36$). Младшие цифры десятично-двоичных чисел всегда четные.

На рис. 11.3 показаны схемы распределения разрядов неполной ячейки для записи адресов ячеек НМБ и НМЛ,

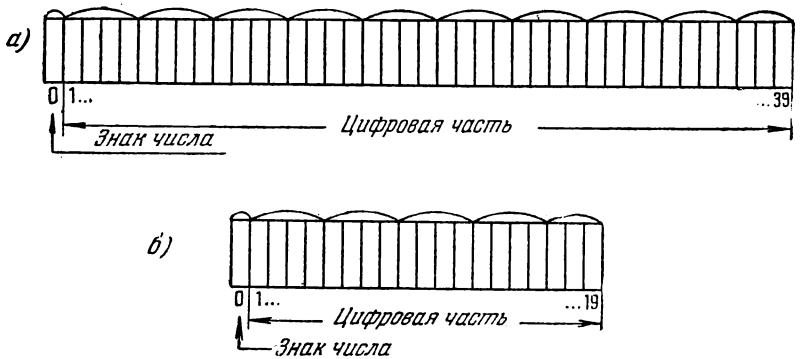


Рис. 11.2. Распределение разрядов при хранении двоично-десятичных чисел с фиксированной запятой в *а*) полной и *б*) неполной ячейках.

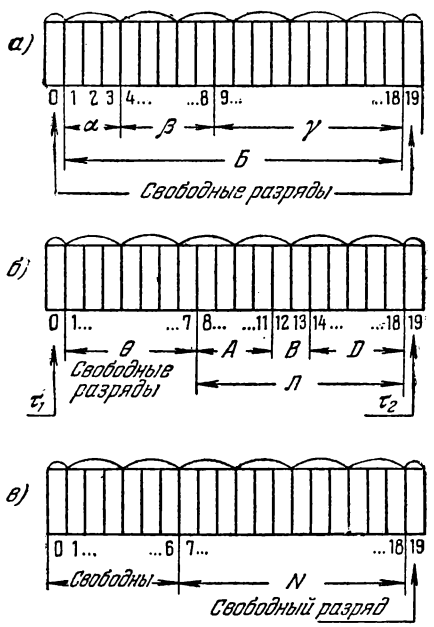


Рис. 11.3. Распределение разрядов неполной ячейки: *а*) для номера ячейки барабана *Б*; *б*) для номера зоны магнитной ленты *Л*; *в*) для номера ячейки в зоне *N*.

применяемые для записи команд обращения к этим устройствам. Номер ячейки магнитного барабана, обозначаемый через *Б*, состоит из

α — номера барабана,

β — номера зоны барабана,

γ — номера ячейки в зоне ($000000 \leqslant Б \leqslant 777777$).

Номер ячейки магнитной ленты (от 0000 0000 до 2777 7777) может быть записан в двух неполных ячейках. В одной из них указывается *Л* — номер зоны, состоящий из номера шкафа *А*, номера блока *В*, номера зоны в блоке *D*. В другой указывается *N* — номер ячейки в зоне.

На приведенных рисунках скобками $\frown$ обозначены объединения двоичных разрядов в восьмеричные, десятичные, двоичные цифры, а также в шестиразрядные группы (для помещения алфавитно-цифровых кодов).

Кроме основного формата команд, совпадающего с основным форматом команд машины *Урал-2* и имеющего вид

$$\tau_1 \ \theta \ a \ \tau_2,$$

в машине *Урал-4* для записи команд обращения к НМБ и НМЛ применяются вспомогательные форматы:

$$* \ Б \ *, \ * \ N \ *,$$

где *Б* и *N* имеют значения и составные части, показанные на рис. 11.3.

Система операций и команд

Система операций и команд мало отличается от системы операций машины *Урал-2*. Отличия касаются главным образом команд, по которым выполняются передачи кодов из ОЗУ в НМБ, НМЛ и обратно и выдача результатов. Команды, имеющие коды операций $\theta = 01 \div 25$; 27; 30; 34; $37 \div 47$; 56, совпадают. В табл. 11.1 описаны операции, выполняемые иначе, чем в машине *Урал-2*, и операции, добавленные в систему команд. При этом обозначения совпадают с принятыми для описания системы команд *Урал-2*. Для операций перевода чисел из двоичной системы счисления в десятичную и обратно обозначают через «$s^{ч}_{(2)}$», «$s^{ч}_{(10)}$» содержимое сумматора чисел в двоичной и десятичной системах счисления.

В команде $\theta = 40$ при $\bar{a} = 0002$ происходит прием в сумматор кодов с перфокарт, при $\bar{a} = 0010$; 0012 — с перфоленты.

Таблица 11.1. Операции и команды машины *Урал-4*

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
26	Контрольное суммирование τ_1 26 *a* τ_2	$s^{ч} := s^{ч} \oplus \tilde{a}$, $s^{п} := s^{п}$	$\equiv$		80	К содержимому сумматора чисел прибавляется содержимое ячейки $\tilde{a}$. При $\tau_2 = 0$ содержимое неполной ячейки $\tilde{a}$ прибавляется к $0 \div 19$-му разрядам сумматора, а к $20 \div 39$-му разрядам прибавляются нули
35	Перевод числа из десятичной системы в двоичную * 35 *a* *	$s^{ч}_{(2)} := \tilde{a}$	$s < 0$		810	Содержимое ячейки $\tilde{a}$, рассматриваемое как десятично-двоичное число с фиксированной запятой, переводится в двоичную систему счисления. Результат является двоичным числом с фиксированной запятой. Образуется в сумматоре чисел

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
36	Перевод числа из двоичной системы в десятичную. * 36 * *	$s^{ч}_{(10)} := s^{ч}_{(2)}$	$s < 0$		530	Содержимое сумматора рассматривается как двоичное число с фиксированной запятой и переводится в десятичное число с фиксированной запятой. Результат образуется в сумматоре чисел
32	Выдача в десятичной системе счисления τ_1 32 0000 + m τ_2	$s := s$	$\equiv$		80 для передачи на приемный регистр соответствующего выходного устройства. Задержка возникает, когда, прежде чем произведена перфорация, происходит вновь обращение к тому же устройству	Содержимое сумматора в десятичной системе счисления выдается на одно из m выходных устройств; m занимает 6 младших разрядов адреса. При $m = 02$ происходит выдача на перфокарты, при $m = 10$ — на один из перфораторов ПРЛ-20, при $m = 11$ — на оба перфоратора ПРЛ-20, работающие в режиме дублирования

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
	Перенос содержимого сумматора в регистр АЦПУ в десятичной системе счисления τ_1 32 4000$+$«k_{10}»$\times$ $\times 2^8 + m$ τ_2	$s := s$; при «k_{10}» $= 1$ и «k_{10}» $= 0$ $R'_{m_1} := s_0^{ч}$, $R'_{m_2} := 1$, $R'_{m_3} := 0$, $R'_{m_4} := 1$, $R'_{m_5} := 0$, $R'_{m_0} := 0$; при «k_{10}» $= 0$, если $\tau_2 = 4$, то $R'_{(m+1)_1} := s_{33}^{ч}$, $R'_{(m+1)_2} := 1$, $R'_{(m+1)_3} := 0$, $R'_{(m+1)_4} := 1$, $R'_{(m+1)_5} := 0$, $R'_{(m+1)_6} := 0$; если $\tau_2 = 0$, то $R'_{(m+j)_i} := s_{4_{j+1-i}}$,	$\equiv$			Содержимое сумматора чисел переписывается в регистр АЦПУ, начиная с разряда, имеющего номер m, m занимает $12 \div 18$-й двоичные разряды команды, «k_{10}» — значение 10-го разряда команды. Если «k_{10}» $= 1$, то в регистр переписывается знак числа, независимо от значения признака τ_2. При «k_{10}» $= 0$; если $\tau_2 = 4$, то в регистр переписываются знаки мантиссы и порядка, если $\tau_2 = 0$, то переписываются знак и первые четыре десятичные цифры мантиссы

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
		$R_{(m+j)_5} := 0$, $R_{(m+j)_6} := 0$, $1 \leqslant j \leqslant 4$, $1 \leqslant i \leqslant 4$				
33	Выдача в восьмеричной системе счисления τ_1 33 0000 + m τ_2	$s := s$	$\equiv$		То же, что и для команды с $\theta = 32$	Отличается от операции $\theta = 32$, выполняемой по команде τ_1 320000 + m τ_2, тем, что содержимое сумматора выдается в восьмеричной системе счисления
	Перенос содержимого сумматора в регистр АЦПУ в восьмеричной системе счисления τ_1 33 4000 + «k_{10}» × × 2^8 + m τ_2	$s := s$; при «k_{10}» = 1 и «k_{10}» = 0 $R'_{m_1} := s^{\text{ч}}_0$, $R'_{m_2} := 1$, $R'_{m_3} := 0$, $R'_{m_4} := 1$, $R'_{m_5} := 0$,	$\equiv$		210	Отличается от операции $\theta = 32$, выполняемой по команде τ_1 32 4000 + «k_{10}» · 2^8 + m τ_2 следующим: если «k_{10}» = = 1, то в разряд регистра с номером m переписывается содержимое

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
		$R'_{m_6} := 0$; при «k_{10}» $= 0$ $R_{(m+j)_i} := s_{3j+1-i}$, $1 \leqslant i \leqslant 3$, $R'_{(m+j)_4} := 0$, $R'_{(m+j)_5} := 0$, $R'_{(m+j)_6} := 0$, если $\tau_2 = 4$, то $1 \leqslant j \leqslant 7$; если $\tau_2 = 0$, то $1 \leqslant j \leqslant 14$				нулевого разряда сумматора. При «k_{10}» $= 0$, если $\tau_2 = 4$, в регистр переписывается 14 восьмеричных разрядов, если $\tau_2 = 0$, то в регистр переписываются семь первых восьмеричных знаков сумматора в виде кода команды, начиная с $m + 1$ разряда R'. В разряд m, как и в предыдущем случае, переписывается «$s_0^{\text{ч}}$»
34	Пропуск строки (интервал) τ_1 34 * *		$\equiv$			На перфоленте пробивается нуль. На остальные выходные устройства выполнение команды не влияет

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
45	Подвод зоны магнитной ленты τ_1 45 *Л* *				80 для запуска устройства подвода. Подвод прекращается, когда зона подведена под считывающие головки	Производится подвод зоны магнитной ленты под считывающие головки. Выполняется одновременно со всеми операциями машины, исключая обращения к тому же лентопротяжному блоку НМЛ
	Обращения к накопителю на магнитной ленте τ_1' 45 *Л* τ_2 τ_1'' θ a' * * *N* * * * a'' *				2000 40-разрядных кодов в секунду без учета времени поиска зоны и требуемой ячейки в зоне	Выполняется по четырем командам, записанным в последовательные ячейки ОЗУ. При всех операциях *Л* представляет собой номер зоны магнитной ленты
	$\theta = 55$ Считывание с магнитной ленты		При неправильном считывании хотя бы одного кода.			Производится считывание $a'' - a' + 1$ 40-разрядных кодов из зоны с номером *Л*, начиная со строки *N*. При $\tau_2 = 0$ про-

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
			При $\omega = 1$ операция прекращается, и машина останавливается			изводится запись в ячейки $a' \div a''$ ОЗУ, при $\tau_2 = 4$ запись в ОЗУ не производится. В обоих случаях работает система контроля правильности считывания информации каждого кода в отдельности (по четности) и всей группы кодов (с помощью контрольного суммирования). Если в зоне *Л*, начиная с ячейки *N* меньше $a'' - a' + 1$ кодов, то в качестве недостающих считываются коды из следующей зоны
	$\theta = 57$ Запись на магнитную ленту					Содержимое ячеек ОЗУ с номерами $a' \div a''$ (40-разрядные коды) переписывается в зону *Л*, начиная с ячейки *N*

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
	$\theta = 65$ Первая операция выборки	$\pi := \pi$, $\rho := \rho$, $r := r$, $\delta := \delta$, $s^{\text{п}} := 0$, $F := 0$,	При неверном считывании кода. При $\omega = 1$ возникает останов			По команде выполняются следующие действия: 1. считывание $a'' - a' + 1$ кодов из зоны $Л$ начиная с ячейки N с шагом «δ», записанным в двенадцати младших разрядах регистра шага δ; 2. сравнение считываемых кодов с признаком, записанным в регистре признака ρ, по тем разрядам, в которых на регистре шкалы r содержатся нули; 3. запись в ячейки ОЗУ, начиная с a', которая производится следующим образом: если старший разряд регистра шага содержит 0 ($«\delta_0» = 0$), то записываются только коды, имеющие признак;

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
						если «δ_0» $= 1$, то в ячейки записываются группы кодов, первым кодом в группе является код, имеющий признак, совпавший с заданным, последующие коды являются теми кодами, которые считываются с магнитной ленты вслед за первым кодом независимо от того, совпадают их признаки с заданным или нет, число кодов в группе равно содержимому двенадцати младших разрядов регистра шага. Операция заканчивается, когда либо число кодов, признаки которых совпали с заданным, равно $a'' - a' + 1$, либо запись произведена в ячейку с номером a''. В сум-

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
						маторе чисел образуется количество кодов, признаки которых совпали с заданным признаком. Содержимое регистров признака, шкалы, счетчика считываемых кодов и шага не меняется. Сумматор порядков и регистр переадресации сбрасывается
	Вторая операция выборки τ_1' 45 *Л* * τ_1' 66 * * * *N* * * *n* * Условие: $n \neq 0000$; $n \neq 7777$;	$\rho := \rho$, $r := r$, $\delta := \delta$, $s^{\text{п}} := 0$, $F := 0$	Такое как и при $\theta = 65$			При выполнении операции: 1) на счетчик числа считываемых кодов π записывается n; 2) производится считывание и сравнение кодов по правилам операции $\theta = 65$; 3) запись в ОЗУ не производится;

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
						4) на сумматоре чисел образуется число кодов, признаки которых совпали с заданным. Операция заканчивается, когда с ленты считано n кодов, содержимое регистров признака, шкалы, счетчика шага не меняется. Сумматор порядков и регистр переадресации сбрасываются
60	Перенос содержимого сумматора на регистр АЦПУ в алфавитно-цифровой форме $\tau_1\ 60\ 4000 + «k_{10}» \times \times 2^8 + m\ \tau_2$	$s := s$; $R_{(m+j)_i} = s_{6j+1-i}$, $1 \leqslant i \leqslant 6$; при $«k_{10}» = 1$ $j = 1$; при $«k_{10}» = 0$, если $\tau_2 = 0$, то $1 \leqslant j \leqslant 3$,	$\equiv$		210	Отличается от операции $\theta = 32$, выполняемой по команде $\tau_1\ 32\ 4000 + + «k_{10}» \cdot 2^8 + m\ \tau_2$, следующим: в разряды регистра АЦПУ, начиная с R_m, переписывается, если $«k_{10}» = 1$, содержимое $1 \div 6$-го разрядов сумматора чисел, независимо

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
		если $\tau_2 = 4$, то $1 \leqslant j \leqslant 6$				от значения τ_2; если «k_{10}» $= 0$, то при $\tau_2 = 0$ — содержимое $1 \div 18$-го разрядов, при $\tau_2 = 4$ — содержимое $1 \div 36$-го разрядов. Каждая шестиразрядная группа переносится в разряд регистра АЦПУ
61	Печать строки на АЦПУ τ_1 61 4000 τ_2	$R' := 0$, $s := s$	$\equiv$		160	Содержимое регистра АЦПУ печатается на бумажной ленте. После печати содержимое R' сбрасывается. При $\tau_2 = 4$ после печати происходит протяжка ленты на одну строку. Если $R' = 0$, то происходит только протяжка ленты. При $\tau_2 = 0$ производится только печать

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
62	Включение звонка * 62 * *	$s := s$	$\equiv$		80	Включается звонок, установленный на пульте управления (для прекращения его работы необходимо нажать кнопку на пульте управления). Выполнение программы продолжается
67	Запись в регистр признака * 67 * *	$\rho := s^{ч}$, $s := s$	$\equiv$		80	В регистр признака записывается содержимое сумматора чисел, содержимое сумматора сохраняется
70	Запись в регистр шага * 70 * *	$\delta_j := s_i^{ч}$, $27 \leqslant i \leqslant 39$, $0 \leqslant j \leqslant 12$, $s := s$	$\equiv$		80	Производится перенос содержимого $27 \div 39$-го разрядов сумматора чисел в регистр шага. Содержимое сумматора сохраняется

Продолжение табл. 11.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
71	Запись в регистр шкалы * 71 * *	$r := s^{ч}$, $s := s$	$\equiv$		80	Выполняется перенос содержимого сумматора чисел в регистр шкалы, содержимое сумматора сохраняется
72	Запись в счетчик числа считываемых кодов * 72 * *	$\pi_j := s_i^{ч}$, $17 \leqslant i \leqslant 39$, $0 \leqslant j \leqslant 22$, $s := s$	$\equiv$		80	Содержимое 17 ÷ 39-го разрядов сумматора чисел переносится в счетчик количества просматриваемых кодов, содержимое сумматора сохраняется
73	Считывание содержимого счетчика количества считываемых кодов * 73 * *	$s^{ч} := 0$, $s_i^{ч} := \pi_j$, $17 \leqslant i \leqslant 39$, $0 \leqslant j \leqslant 22$, $s^{п} := s^{п}$	$\equiv$		80	Содержимое счетчика числа считываемых кодов переносится в 17 ÷ ÷ 39-й разряды сумматора чисел

Пульт управления

Пульт управления машины *Урал-4* отличается от пульта управления *Урал-2* тем, что в нем имеются дополнительные лампочки сигнализации о работе НМЛ, НМБ и устройств ввода и вывода. Изменен также порядок работы машины при нажатии кнопки «Начальный пуск». При этом автоматически выполняются команды, закоммутированные в четырех ячейках блока начального пуска устройства управления. Обозначим эти ячейки номерами: 0000*; 0001*; 0002*; 0003*.

Номер ячейки блока начального пуска	τ_1	Код операции θ	Адрес	τ_2
0000 *		25	0026	4
0001 *		40	0002	4
0002 *	—	16	0032	4
0003 *		24	0001 *	0

Рис. 11.4. Команды, выполняемые при нажатии кнопки «Начальный пуск».

Тогда выполняемые команды имеют вид, представленный на рис. 11.4. По этим командам вводится содержимое 11 строк перфокарты в ячейки ОЗУ 0004 ÷ 0016. После ввода управление передается ячейке с номером 0004.

Подготовка данных. Ввод и вывод

Числа и команды, предназначенные для пробивки и ввода в машину, записываются на стандартных бланках.

Один из возможных видов бланка с образцами записи двоичных чисел, десятичных чисел и команд приведен на рис. 11.5. В этом бланке содержится 33 строки для размещения содержимого трех перфокарт. Напомним, что 12-я строка перфокарты служит для помещения контрольной суммы. Автоматический контроль, осуществляемый при вводе с перфокарт, производится с помощью программы ввода. С

	0100	25 0534 4	Команды
	1	— 42 3406 4	
	2	22 0326 4	
	3	32 5003	
	4	— 16 3406 4	
	5	24 0101	
	6		
	7		
	0110	11 0050	
	1	25 0677	
	2	— 26 0777	
	3	24 0112	
	4	16 1000 4	
	5	25 0001	
	6	27 0126	
	7		
	0120		
	1		
	2		
	3		
	4	+ 93 864 6 73 + 11	$0{,}938\ 64\ 673 \cdot 10^{11}$
	5	— 50 000 0 00 — 01	$-0{,}5 \cdot 10^{-1}$
	6	— 25 000 0 00	— 0,25
	7		
	0130	14 512 0 02 + 02	$0{,}145\ 12\ 002 \cdot 10^{22}$
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	0140		

Десятичные числа

Рис. 11.5. Образец записи команд и чисел на бланке.

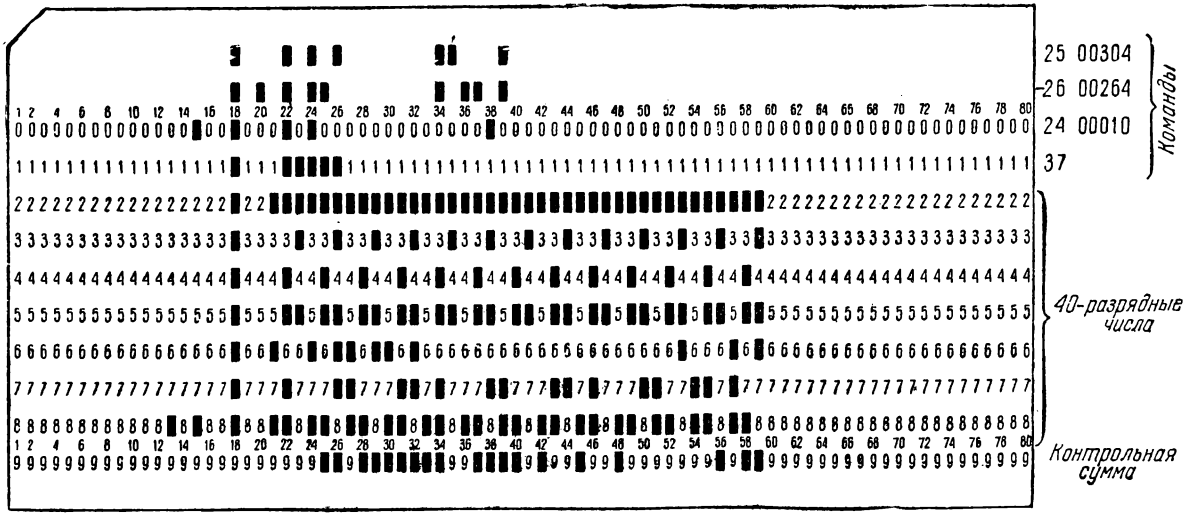


Рис. 11.6. Перфокарта машины *Урал-4*.

Таблица 11.2. Перевод шестизначного алфавитно-цифрового кода в пятизначный телеграфный код *)

Символ	Регистр	Пятизначный код	Шестизначный код	Символ	Регистр	Пятизначный код	Шестизначный код	Символ	Регистр	Пятизначный код	Шестизначный код
0	Циф.	11110	000000	К	Рус.	10011	010110	b	Лат.	00011	100111
1	Циф.	10000	000001	Л	Рус.	11011	010111	Э	Циф.	01110	101000
2	Циф.	01000	000010	М	Рус.	01011	011600	V*	Лат.	10100	101000
3	Циф.	00100	000011	Н	Рус.	01111	011001	Ю	Рус.	01111	101001
4	Циф.	10100	000100	О	Рус.	11100	011010	Q*	Лат.	10111	101001
5	Циф.	11100	000101	П	Рус.	11000	011011	Я	Рус.	00011	101010
6	Циф.	10010	000110	Р	Рус.	00111	011100	*O**	Циф.	00011	101010
7	Циф.	01010	000111	С	Рус.	00101	011101	?	Циф.	10011	101011
8	Циф.	00110	001000	Т	Рус.	10101	011110	.	Циф.	00101	101100
9	Циф.	10110	001001	Т*	Лат.	10101	011110	,	Циф.	01001	101101
+	Циф.	11010	001010	У	Рус.	10100	011111	:	Циф.	11001	101110
—	Циф.	00111	001011	Y	Лат.	00100	011111	;	Циф.	01101	101111
А	Рус.	10000	001100	Ф	Рус.	01110	100000	×	Циф.	11101	110000
Б	Рус.	00110	001101	F*	Лат.	01110	100000	/	Циф.	10111	110001
В	Рус.	01101	001110	Х	Рус.	11010	100001	=	Циф.	11011	110010
Г	Рус.	01010	001111	X*	Лат.	01001	100001	<	Лат.	10000	110011
Д	Рус.	11110	010000	Ц	Рус.	10110	100010	>	Лат.	00110	110100
D*	Лат.	11110	010000	W*	Лат.	01101	100010	∧	Лат.	01000	110101
Е	Рус.	01000	010001	Ч	Циф.	10101	100011	v	Лат.	11101	110110
Ж	Рус.	11101	010010	G*	Лат.	01010	100011	(	Циф.	11000	110111
L	Лат.	11011	010010	Ш	Рус.	01100	100100	)	Циф.	01011	111000
З	Рус.	11001	010011	Z*	Лат.	11001	100100	[*	Лат.	10011	110001
N*	Лат.	01111	010011	Щ	Рус.	10111	100101	]*	Лат.	01011	111010
И	Рус.	01100	010100	R*	Лат.	00111	100101	↑	Лат.	11010	111011
I	Лат.	01100	010100	Ы	Рус.	00100	100110	↓	Лат.	11100	111100
Й	Рус.	10010	010101	S*	Лат.	00101	100110	→	Лат.	11000	111101
J*	Лат.	10010	010101	Ь	Рус.	01001	100111	*	Лат.	10110	111110
								Пробел	Рус. Лат. Циф.	10001	111111

* Регистровые посылки имеют коды: русский 11111, латинский 00001, цифровой 00010.

помощью программы перекодировки возможно также преобразование кода СТА-35 в код АЦПУ или любой другой. Такие программы могут выполняться в процессе ввода, без дополнительных затрат времени. Это обеспечивается тем, что после ввода одной строки по команде $\theta = 40$, служащей для ввода, в машине можно выполнить серию команд, предназначенную для преобразования кодов или их контроля. Время выполнения такой серии не должно превышать при вводе с перфокарт 16 *мсек*, при вводе с перфоленты — 1,6 *мсек*.

Наличие коммутационной доски в читающем устройстве для перфокарт позволяет вводить перфокарты, содержащие результаты обработки информации на счетно-перфорационных машинах, а также использовать для перфорации вводимой информации любые перфораторы.

На рис. 11.6 приведен образец пробивки команд, двоичных и десятичных чисел на перфокарте, выданной выходным перфоратором. Каждая строка перфокарты снабжается маркером.

При выполнении операций выдачи и ввода используется табл. 11.2 перевода шестизначного алфавитно-цифрового кода в пятизначный телеграфный код. Символы, не отмеченные знаком *, имеются на печатающих колесах алфавитно-цифрового печатающего устройства.

ГЛАВА 12

ПРОГРАММНО-УПРАВЛЯЕМАЯ МАШИНА *СЕТУНЬ*

Основные характеристики машины *Сетунь*

Тип машины	вычислительная
Класс машины	малая
Среднее число основных операций в секунду	4500
Рабочая система счисления	троичная с симметричной базой
Форма представления числа	с фиксированной запятой
Длина кода числа	18 разрядов или 9 разрядов
Точность вычислений	7 — 8 десятичных знаков или 3 — 4 десятичных знака
Диапазон представления чисел	$3^{-16} \leqslant \lvert x \rvert < \frac{1}{2}\, 3^{2}$ или $3^{-7} \leqslant \lvert x \rvert < \frac{1}{2}\, 3^{2}$.
Порядок выполнения команд	заданный (естественный)
Адресность машины	одноадресная
Виды запоминающих устройств	оперативное запоминающее устройство на магнитных сердечниках емкостью 162 9-разрядных или 81 18-разрядное троичное число; время обращения к ОЗУ 45 *мксек*; внешнее запоминающее устройство на магнитном барабане емкостью 3888 9-разрядных или 1944 18-разрядных троичных числа; время обращения к магнитному барабану 7,5 *мсек* на считывание (запись) 54 9-разрядных троичных чисел; скорость вращения магнитного барабана 6000 *об/мин*
Виды ввода и вывода	вводные устройства: электромеханическое со скоростью ввода 7 знаков в секунду и фотоэлектрическое со скоростью ввода 800 знаков в секунду; ввод с перфорированной бумажной пятипозиционной ленты:

	выводное устройство — телетайп — работает со скоростью печатания 7 знаков в секунду; наряду с печатью производится перфорация
Количество электронных ламп	40
Потребляемая мощность	2,5 *квт*
Занимаемая рабочая площадь	20 $м^2$

Программно-управляемая машина *Сетунь* — универсальная; временной порядок эксплуатации — произвольный.

Краткие сведения об основных устройствах машины

Запоминающие устройства. Запоминающих устройств в машине два: оперативное (ОЗУ) и внешнее (ВЗУ).

ОЗУ выполнено на магнитных сердечниках и может хранить одновременно 162 9-разрядных или 81 18-разрядное троичное число.

Восемнадцатиразрядные (длинные) ячейки ОЗУ занумерованы в смешанной девятерично-троичной системе счисления числами $\bar{4}\bar{4}\bar{1} \div 44\bar{1}$. Первые две цифры этих чисел — девятеричные, третья — троичная. Одна длинная ячейка с номером $y_1y_2\,\bar{1}$ состоит из двух коротких (9-разрядных) ячеек с номерами $y_1y_2\,0$ (левая) и $y_1y_2\,1$ (правая). Все ячейки ОЗУ разделены на три зоны (группы). Первую зону составляют длинные ячейки ОЗУ с номерами от $\bar{4}\bar{4}\bar{1}$ до $\bar{2}4\bar{1}$ ($\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}_{(3)} \div \div \bar{1}1111_{(3)}$), вторую зону — ячейки с номерами от $\bar{1}4\bar{1}$ до $14\bar{1}$ ($0\bar{1}\bar{1}\bar{1}\bar{1}_{(3)} \div 01111_{(3)}$), третью зону — ячейки с номерами от $2\bar{4}\bar{1}$ до $44\bar{1}$ ($1\bar{1}\bar{1}\bar{1}\bar{1}_{(3)} \div 1111\bar{1}_{(3)}$). В соответствии со старшими троичными цифрами номеров ячеек зон условимся, что первая зона имеет номер $\bar{1}$, вторая — номер 0 и третья — номер 1.

ВЗУ, выполненное на магнитном барабане, имеет емкость 3888 коротких или 1944 длинных ячеек.

Числа и команды на магнитном барабане записываются только зонами (группами) по 54 9-разрядных троичных числа, либо по 27 18-разрядных троичных чисел. Зоны магнитного барабана в девятеричной системе счисления имеют номера от $\bar{4}\bar{4}$ до 44. (Зоны с номерами от $0\bar{4}$ до 04 считаются запрещенными.)

Запись (считывание) на магнитный барабан производится только через ОЗУ. При этом одна из зон ОЗУ переписывается полностью на выбранную зону магнитного барабана (или зона барабана — на зону ОЗУ при считывании).

Время считывания (записи) одной зоны магнитного барабана составляет 7,5 *мсек*. Скорость вращения магнитного барабана — 6000 оборотов в мин.

Арифметическое устройство (АУ) предназначено для производства операций над кодами. Все операции в арифметическом устройстве выполняются с 18-разрядными числами.

К числу основных узлов АУ относятся 19-разрядный сумматор s и 18-разрядный регистр R.

Содержание нулевого разряда сумматора соответствует значению сигнала «переполнения» φ: $\varphi = \bar{1}$ или $\varphi = 1$, если $|(s)| \geqslant 4{,}5$; при $|(s)| < 4{,}5$ $\varphi = 0$.

Устройство управления (УУ). К числу основных узлов УУ относятся следующие элементы:

регистр команд, имеющий 9 разрядов;

регистр номера команды C, имеющий пять разрядов. По номеру ячейки, указанному в C, машина производит выбор команды из ОЗУ и запоминает ее в регистре команд. Отметим, что ввиду особенностей кодирования коротких ячеек в ОЗУ номер следующей команды (при отсутствии передачи управления) получается либо прибавлением единицы к содержимому регистра C, если $\tau = 0$, либо прибавлением числа 2, если $\tau = 1$ (τ — признак длины ячейки в команде).

регистр переадресации F, имеющий пять разрядов. Регистр F предназначен для переадресации команды, находящейся в регистре команд, если признак переадресации $\tau_1 \neq 0$. При $\tau_1 = 1$ содержимое регистра F прибавляется, а при $\tau_1 = \bar{1}$ — вычитается от адресной части кода команды. При $\tau_1 = 0$ адрес команды, находящейся в регистре команд, остается без изменения;

блок управления вводом и выводом. Этот блок содержит два 9-разрядных триггерных регистра — «входной» и «выходной».

В устройстве управления в результате выполнения команды вырабатывается сигнал ω, управляющий выполнением команд условной передачи управления. Его значение определяется либо содержимым сумматора, либо содержимым одного из

регистров R, C или F. В зависимости от того, чем определяется значение сигнала ω, будем его обозначать через ω_s, ω_R, ω_C или ω_F. Сигнал ω равен

$$\omega_s = \begin{cases} \bar{1} \text{ при } 0 < |(s)| < 0{,}5, \\ 0 \text{ при } 0{,}5 \leqslant |(s)| \leqslant 1{,}5 \text{ или при } (s) = 0, \\ 1 \text{ при } |(s)| > 1{,}5 \end{cases}$$

или (понимая под r один из регистров R, C, F)

$$\omega_r = \begin{cases} 1 \text{ при } (r) < 0, \\ 0 \text{ при } (r) = 0, \\ \bar{1} \text{ при } (r) > 0. \end{cases}$$

Вводные и выводные устройства (УВВ). Вводных устройств в машине два: фотоэлектрическое и электромеханическое. Фотоэлектрическое входное устройство обладает скоростью ввода 800 знаков (строк) в секунду; электромеханическое — 7 знаков в секунду. Ввод производится с перфорированной пятипозиционной бумажной ленты, на которой кодированные цифры чисел или команд размещаются последовательно одна за другой.

Выводное устройство (телетайп) предназначается для вывода команд и чисел из ОЗУ на печать и на перфоленту. Скорость печатания 7 знаков в секунду.

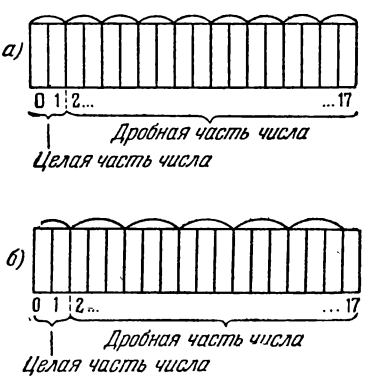
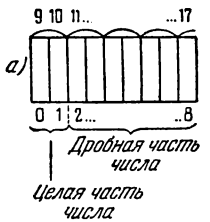


Рис. 12.1. Распределение разрядов длинной ячейки памяти: *а*) для 18-разрядного троичного числа; *б*) для троично-десятичного числа.

Представление чисел и команд

На рис. 12.1 показано распределение разрядов длинной ячейки памяти машины при хранении в ней 18-разрядного троичного числа и троично-десятичного числа. (Знак троичного и троично-десятичного чисел определяется знаком цифры, находящейся в старшем (нулевом) разряде.)

На рис. 12.2 показано распределение разрядов ячейки памяти при хранении в ней 9-разрядного троичного числа и команды. Команда машины *Сетунь* имеет следующую структуру: $a\tau\theta\tau_1$, где a — адрес команды, τ — признак длинной или короткой ячейки, θ — код операции, τ_1 — признак переадресации.



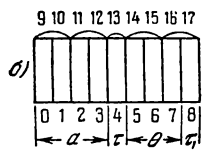


Рис. 12.2. Распределение разрядов короткой ячейки памяти: *а*) для 9-разрядного троичного числа; *б*) для команды.

Скобками ⌢ на рис. 12.1 и 12.2 указаны объединения троичных разрядов в девятеричные и десятичные цифры, применяемые при записи чисел и команд.

Система операций и команд

Система операций и команд приведена в табл. 12.1.

Кроме обозначений, введенных ранее (см. стр. 271), применяются следующие обозначения:

s — сумматор АУ,

R — регистр АУ,

F — регистр переадресации УУ,

$\tilde{a}$ — исполнительный адрес команды, формируемый по правилу

$$\tilde{a} = a \cdot \tau \boxplus\limits_{s} «F» \cdot \tau_1;$$

a^* — ячейка, состоящая из старших пяти разрядов ячейки a; a_* — из младших четырех разрядов ячейки a.

В таблице код операции представляется двумя девятеричными цифрами, которые включают в себя коды θ и τ_1 для случая $\tau_1 = 0$.

Действия, выполняемые машиной, в ряде случаев зависят не только от кода операции, но и от значения некоторых разрядов адреса команды. В таких случаях вместо сокращенной записи адресной части $a\tau$ будем употреблять развернутую запись

$$x_0 y_1 y_2,$$

где x_0 — троичная, а y_1 и y_2 — девятеричные цифры. Под обозначениями $\tilde{x}_0$, $\tilde{y}_1$, $\tilde{y}_2$ будут подразумеваться те же цифры, соответствующие исполнительному адресу $\tilde{a}$.

Таблица 12.1. Система операций и команд машины *Сетунь*

Код операции	Название операции и вид команды	Краткое содержание команды	Вид сигнала ω	Время выполнения операции, *мксек*	Пояснения
$\bar{3}\bar{3}$	Перенос чисел с магнитного барабана в память $x_0\ y_1y_2\ \bar{3}\bar{3}$		Сохраняется	7500	Все числа из зоны магнитного барабана, имеющей номер $\tilde{y}_1\tilde{y}_2$, переносятся в зону с номером $\tilde{x}_0$ ОЗУ
$\bar{3}0$	Ввод в ОЗУ (перенос чисел с перфоленты в память) $x_0\ 00\ \bar{3}0$		Сохраняется	7 знаков в секунду	Числа с перфоленты переносятся в зону оперативной памяти, имеющую номер $\tilde{x}_0$. Ввод прекращается либо после заполнения всей зоны оперативной памяти, имеющей номер $\tilde{x}_0$, либо после считывания с перфоленты комбинации «стоп»
$\bar{3}0$	Троичный вывод $x_0\ 03\ \bar{3}0$		Сохраняется		Печатаются все команды (или девятиразрядные числа) зоны ОЗУ с номером $\tilde{x}_0$
$\bar{3}0$	Вывод в один столбец $x_0\ 0\bar{3}\ \bar{3}0$		Сохраняется		Все числа зоны ОЗУ с номером $\tilde{x}_0$ печатаются в десятичной системе счисления в один столбец

Продолжение табл. 12.1

Код операции	Название операции и вид команды	Краткое содержание команды	Вид сигнала ω	Время выполнения операции, *мксек*	Пояснения
$\bar{3}0$	Вывод в два столбца $x_0\ 0\bar{1}\ \bar{3}0$		Сохраняется		То же, что и в предыдущей команде, но печать выводимых чисел производится в два столбца
$\bar{3}0$	Вывод в три столбца $x_0\ 01\ \bar{3}0$		Сохраняется		То же, что и в предыдущих двух командах, но печать выводимых чисел производится в три столбца
$\bar{3}3$	Перенос чисел из ячеек памяти на магнитный барабан $x_0\ y_1y_2\ \bar{3}3$		Сохраняется	7500	Все числа зоны ОЗУ с номером $\tilde{x}_0$ записываются на зону магнитного барабана с номером $\tilde{y}_1\ \tilde{y}_2$
$\bar{2}\bar{3}$	Нормализация $a\tau\ \bar{2}\bar{3}$		ω_s	285	Содержимое сумматора нормализуется. Полученная мантисса числа переносится в ячейку $\tilde{a}$. В ячейку s^* записывается порядок числа. После выполнения операции $0{,}5 \leqslant \lvert \langle \tilde{a} \rangle \rvert \leqslant 1{,}5$

Продолжение табл. 12.1

Код операции	Название операции и вид команды	Краткое содержание команды	Вид сигнала ω	Время выполнения операции, *мксек*	Пояснения
$\bar{2}0$	Сдвиг $a \tau \bar{2}0$		ω_s	220	Число «s» сдвигается на число «$\tilde{a}^*$». Сдвиг производится вправо, если «$\tilde{a}^*$» > 0, и влево, если «$\tilde{a}^*$» < 0. Результат операции остается в сумматоре s
$\bar{2}3$	Перенос числа из сумматора в ячейку памяти $a \tau \bar{2}3$	$\tilde{a} := s$	Сохраняется	225	Число «s» переносится в ячейку $\tilde{a}$. Содержимое сумматора не изменяется
$\bar{1}\bar{3}$	Сложение в регистре переадресации $a \tau \bar{1}\bar{3}$	$F := F + \tilde{a}^*$	ω_F	180	Содержимое регистра переадресации F складывается с числом «$\tilde{a}$». Результат операции записывается в F
$\bar{1}0$	Перенос числа из ячейки памяти в F $a \tau \bar{1}0$	$F := \tilde{a}^*$	ω_F	180	Число «$\tilde{a}^*$» переносится в регистр переадресации F

Продолжение табл. 12.1

Код операции	Название операции и вид команды	Краткое содержание команды	Вид сигнала ω	Время выполнения операции, *мксек*	Пояснения
$\overline{1}3$	Сложение в регистре номера команды $a\ \tau\ \overline{1}3$	$F:=C+\tilde{a}^*$, $C:=F$	ω_F	180	Содержимое регистра счетчика команд C складывается с числом «$\tilde{a}^*$», результат операции посылается в регистр C и в регистр переадресации F
$0\overline{3}$	Перенос числа из регистра переадресации в ячейку памяти $a\ \tau\ 0\overline{3}$	$\tilde{a}^*:=F$, $\tilde{a}_*:=0$	ω_F	180	Содержимое регистра переадресации F переносится в ячейку $\tilde{a}^*$. В других разрядах этой ячейки устанавливаются нули
00	Безусловный переход $a\ \tau\ 00$	$C:=\tilde{a}$	Сохраняется	100	Передается управление команде «$\tilde{a}$»
03	Перенос числа из регистра номера команды в ячейку памяти $a\ \tau\ 03$	$\tilde{a}^*:=C$, $\tilde{a}_*:=0$	Сохраняется	180	Содержимое регистра номера команды C переносится в старшие пять разрядов ячейки $\tilde{a}$. В остальных разрядах этой ячейки устанавливаются нули

Продолжение табл. 12.1

Код операции	Название операции и вид команды	Краткое содержание команды	Вид сигнала ω	Время выполнения операции, *мксек*	Пояснения
$1\bar{3}$	Условный переход (УП $=\bar{1}$) $a\ \tau\ 1\bar{3}$	$C := \tilde{a}$, если $\omega = \bar{1}$	Сохраняется	100÷180	При $\omega \neq \bar{1}$ управление передается следующей команде, при $\omega = \bar{1}$ — команде «$\tilde{a}$»
10	Условный переход (УП $=0$) $a\ \tau\ 10$	$C := \tilde{a}$, если $\omega = 0$	Сохраняется	100÷180	Управление передается к следующей команде, если $\omega \neq 0$, и команде «$\tilde{a}$», если $\omega = 0$
13	Условный переход (УП $=1$) $a\ \tau\ 13$	$C := \tilde{a}$, если $\omega = 1$	Сохраняется	100÷180	Управление передается к следующей команде, если $\omega \neq 1$, и команде «$\tilde{a}$», если $\omega = 1$
$2\bar{3}$	Останов машины $a\ \tau\ 2\bar{3}$				Производится останов машины. При нажатии после останова на пульте управления кнопки «пуск» происходит перенос числа «$\tilde{a}$» в регистр R и вырабатывается сигнал ω_R, после чего машина переходит к автоматическому выполнению программы, начиная с команды, следующей за командой останова

Продолжение табл. 12.1

Код операции	Название операции и вид команды	Краткое содержание команды	Вид сигнала ω	Время выполнения операции, *мксек*	Пояснения
20	Логическое поразрядное умножение a τ 20	$s_i := \tilde{a}_i \wedge s_i$ $(i = 0,1,\ldots, 17)$	ω_s	180	Содержимое ячейки $\tilde{a}$ поразрядно умножается на содержимое сумматора s. Результат операции остается в сумматоре s
23	Перенос числа из ячейки памяти в регистр R a τ 23	$R := \tilde{a}$	ω_R	180	Число «$\tilde{a}$» переносится в регистр R
33̄	Вычитание a τ 33̄	$s := s - \tilde{a}$	ω_s	180	Из числа «s», находящегося в сумматоре АУ, вычитается число «$\tilde{a}$». Результат операции остается в сумматоре s
30	Перенос числа из ячейки a памяти в сумматор s a τ 30	$s := \tilde{a}$	ω_s	180	Число «$\tilde{a}$» переносится в сумматор s

Продолжение табл. 12.1

Код операции	Название операции и вид команды	Краткое содержание команды	Вид сигнала ω	Время выполнения операции, *мксек*	Пояснения
33	Сложение $a\ \tau\ 33$	$s := s + \tilde{a}$	ω_s	180	Число «s», находящееся в сумматоре АУ, алгебраически складывается с числом «$\tilde{a}$». Результат остается в сумматоре s
$4\bar{3}$	Умножение — $\bar{1}$ $a\ \tau\ 4\bar{3}$	$s := \tilde{a} + s \cdot R$	ω_s	325	Число «s» умножается на число «R». Результат алгебраически складывается с числом «$\tilde{a}$». Результат операции остается в сумматоре s
40	Умножение — 0 $a\ \tau\ 40$	$R := s$, $s := \tilde{a} \cdot s$	ω_s	325	Число «s» переносится в регистр R. Далее число «$\tilde{a}$» умножается на число «s». Результат остается в сумматоре s
43	Умножение — 1 $a\ \tau\ 43$	$s := \tilde{a} \cdot R + s$	ω_s	325	Число «$\tilde{a}$» умножается на число «R». Результат алгебраически складывается с числом «s». Результат остается в сумматоре s

Замечание. Если в команде $\tau \neq 1$, то действие производится с числом «$\tilde{a}$», у которого 9 младших разрядов равны нулю; из сумматора в ячейку $\tilde{a}$ посылаются лишь 9 старших разрядов сумматора.

Пульт управления

Пульт управления имеет две панели: сигнальную панель и панель управления.

На сигнальной панели располагаются ряды неоновых лампочек, отражающих состояние регистра команд, регистра переадресации F, регистра C, сумматора s и регистра R. Каждому регистру соответствуют два ряда лампочек. Свечение лампочки какого-либо разряда верхнего ряда соответствует значению $\bar{1}$ соответствующего разряда в регистре; свечение лампочки нижнего ряда соответствует значению 1 того же разряда регистра; отсутствие свечения лампочек в верхнем и нижнем рядах соответствует значению нуль разряда регистра.

Управляющая панель содержит следующие тумблерные регистры, тумблеры и кнопки:

Тумблерный регистр набора команды, которую необходимо выполнить с пульта.

Тумблерный регистр «Адрес останова», на котором набирается номер ячейки ОЗУ для «останова по адресу команды» или для «останова по адресной части команды» (справа от этого регистра имеется переключающий тумблер). При совпадении содержимого регистра «Адрес останова» с содержимым регистра C или с адресной частью регистра команд машина останавливается.

Восемнадцатиразрядный тумблерный регистр набора числа, предназначенный для изменения содержимого сумматора s или регистра R арифметического устройства (в зависимости от положения находящегося слева от этого регистра переключающего тумблера).

Тумблер, включающий «Однотактный режим», при котором машина останавливается после выполнения каждой команды.

Кнопки «Останов», «Пуск», «Начальный пуск» и «Команда с пульта управления».

Кнопка «Команда с пульта управления» предназначена для выполнения команды, набранной на тумблерном регистре «Набор команды». (Если на тумблерном регистре набрана команда условного или безусловного перехода, то содержимое регистра C изменяется, в регистр команд посылается команда из ячейки памяти C, которая затем выполняется, и машина останавливается.)

Кнопка «Начальный пуск» служит для ввода кодов во вторую зону ОЗУ, начиная с первой ячейки этой зоны. Ввод прекращается либо при считывании с перфоленты комбинации «Стоп», либо после заполнения всей зоны ОЗУ, после чего управление передается в ячейку с номером 00001.

Замечание. Все тумблеры пульта управления имеют три возможных положения: соответствующие значениям $\bar{1}$, 0, 1.

Подготовка данных. Ввод и вывод

Команды и числа, предназначенные для ввода в машину, первоначально записываются на специальных бланках. Образец таких бланков с примерами записи чисел и команд приведен на рис. 12.3.

Номер команды	Адрес команды	т-признак	Код операции	Пояснения
$2\,\bar{4}\,\bar{1}$	—275	6	30	—2,7563 } Десятичные
$2\,\bar{3}\,\bar{1}$	+039	6	16	0,396157 } числа
$\bar{1}\,\bar{4}\,0$	$\bar{4}\bar{4}$	$\bar{1}$	23	} Команды
$\bar{1}\,\bar{4}\,1$	$\bar{4}\bar{3}$	$\bar{1}$	30	
$\bar{1}\,\bar{3}\,0$	$\bar{4}\bar{2}$	$\bar{1}$	$4\bar{3}$	

Рис. 12.3. Образец записи чисел и команд на бланке.

Перфорирование ленты производится на телетайпе, имеющем клавиши десятичных цифр, девятеричных цифр с симметричной базой, троичных цифр с симметричной базой и специальные признаки:

признак «К» команды или девятиразрядного троичного числа, признак «стоп» и знаки «+» и «—». На рис. 12.4 приведена кодировка цифр и знаков на перфоленте.

На телетайпе можно производить и репродукцию перфоленты.

При пробивке команд или чисел, записанных в смешанной троично-девятеричной системе счисления, необходимо перед каждой командой или числом пробивать признак «К». Этот признак при вводе обозначает конец предыдущей команды.

Ввод чисел и команд в ОЗУ производится по команде с кодом операции $\bar{3}0$. Начальный ввод производится с пульта управления путем нажатия кнопки «начальный пуск». При этом коды (числа или команды) с перфоленты вводятся во вторую зону ОЗУ, начиная с первой ячейки этой зоны. Ввод прекращается либо при считывании с перфоленты комбинации «Стоп», либо после заполнения всей зоны, после чего управление передается в ячейку с номером 00001.

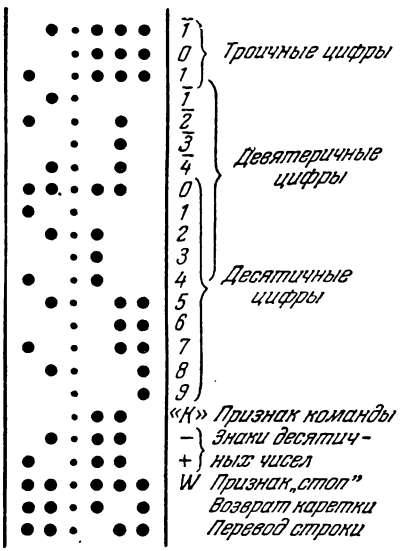


Рис. 12.4. Образец кодирования цифр и знаков на перфоленте.

Вывод из ОЗУ на печать производится по командам с кодом операции $\bar{3}0$.

Команда печатается в смешанной девятерично-троичной системе счисления следующим образом:

$$k y_1 y_2 x_1 y_3 y_4,$$

где k — признак команды, y_1, y_2, y_3 и y_4 — цифры девятеричной системы счисления с симметричной базой, а x_1 — цифра троичной системы счисления с симметричной базой. При этом $y_1 y_2$ образуют адрес команды, x_1 — признак длины ячейки, $y_3 y_4$ — код операции.

Отрицательные троичные и девятеричные цифры печатаются в перевернутом виде, например $\bar{2}$ печатается как 2.

Форма печати десятичных чисел зависит от подпрограммы перевода чисел из троичной системы в десятичную и может быть произвольной.

Дополнение

Кроме описанного выше варианта машины *Сетунь* имеется другой ее вариант, который отличается вводными и выводными устройствами, а также способом кодирования вводной информации.

Таблица 12.2. Кодировка символов

Символы на цифровом регистре	Символы на буквенном регистре		Изображение тремя троичными цифрами	Изображение двумя троичными цифрами	Комбинация пробивок
	стоп		$\bar{1}\ \bar{1}\ \bar{1}$		
—	F	Б	$\bar{1}\ \bar{1}\ 0$		
	подчеркивание		$\bar{1}\ \bar{1}\ 1$		
	перевод строки		$\bar{1}\ 0\ \bar{1}$		
10	G	Щ	$\bar{1}\ 0\ 0$		
●	H	Н	$\bar{1}\ 0\ 1$		
	возврат каретки		$\bar{1}\ 1\ \bar{1}$		
+	I	И	$\bar{1}\ 1\ 0$		
A	J	Ы	$\bar{1}\ 1\ 1$		
B	K	К	$0\ \bar{1}\ \bar{1}$	$\bar{1}\ \bar{1}$	
C	L	Г	$0\ \bar{1}\ 0$	$\bar{1}\ 0$	
D	M	М	$0\ \bar{1}\ 1$	$\bar{1}\ 1$	
E	N	П	$0\ 0\ \bar{1}$	$0\ \bar{1}$	
0	P	Р	$0\ 0\ 0$	$0\ 0$	
1	Q	Ф	$0\ 0\ 1$	$0\ 1$	
2	R	Я	$0\ 1\ \bar{1}$	$1\ \bar{1}$	
3	S	Ь	$0\ 1\ 0$	$1\ 0$	
4	T	Т	$0\ 1\ 1$	$1\ 1$	
5	U	П	$1\ \bar{1}\ \bar{1}$		
6	V	Л	$1\ \bar{1}\ 0$		
7	W	Ш	$1\ \bar{1}\ 1$		
8	X	Х	$1\ 0\ \bar{1}$		
9	Y	У	$1\ 0\ 0$		
пробел	Z	Э	$1\ 0\ 1$		
	цифровой регистр		$1\ 1\ \bar{1}$		
	буквенный регистр		$1\ 1\ 0$		
)	(	Ж	$1\ 1\ 1$		

Этот вариант машины имеет два фотосчитывающих вводных устройства, каждое из которых производит: а) ввод команд (первое устройство по команде $x_0 01\ \bar{3}0$, второе — $x_0 03\ \bar{3}0$); б) ввод десятичных чисел или других символов (первое устройство по команде $x_0 0\bar{1}\ \bar{3}0$, второе — $x_0 0\bar{3}\ \bar{3}0$). Окончание ввода производится либо при считывании с перфоленты комбинации «стоп», либо при заполнении всей зоны x_0 ОЗУ.

Вывод на печать производится одним из двух печатающих устройств, каждое из которых по команде $x_0 10\ \bar{3}0$ производит печать команд, а по команде $x_0\bar{1}0\ \bar{3}0$ — печать десятичных чисел или других символов. В этих командах x_0 — номер зоны ОЗУ, из которой производится вывод на печать. Печатающие устройства отличаются друг от друга наборами символов (одно устройство имеет литеры букв русского алфавита, другое — латинского алфавита). На печатающих устройствах отрицательные цифры $\bar{1}$, $\bar{2}$, $\bar{3}$, $\bar{4}$ печатаются соответственно буквами E, D, C, B. (Полный набор символов на двух печатающих устройствах приведен в табл. 12.2.)

При выводе на печать предусмотрена возможность параллельного включения в работу перфорирующего телетайпа.

При подготовке данных для ввода в машину используется перфорирующий телетайп, имеющий клавиши десятичных цифр, букв русского и латинского алфавитов, некоторых математических знаков и специальных управляющих признаков (см. табл. 12.2).

Литература к главе 12: [27].

ГЛАВА 13

ПРОГРАММНО-УПРАВЛЯЕМАЯ МАШИНА *МИНСК-2*

Основные характеристики машины *Минск-2*

Тип машины	вычислительная
Класс машины	средняя
Число основных операций в секунду	5000
Рабочая система счисления	двоичная
Форма представления чисел	с фиксированной запятой или с плавающей запятой
Диапазон представления чисел	$2^{-36} \leqslant \lvert x \rvert \leqslant 1 - 2^{-36}$ $2^{-64} \leqslant \lvert x \rvert \leqslant (1 - 2^{-28}) \cdot 2^{63}$
Порядок выполнения команд	естественный
Адресность машины	двухадресная
Виды запоминающих устройств	оперативное запоминающее устройство на магнитных сердечниках; емкость 4096 37-разрядных кодов; время обращения 24 *мксек*; внешнее запоминающее устройство на магнитной ленте; емкость 400 000 кодов; время записи (чтения) 2048 кодов за 40 *сек*
Виды ввода и вывода	ввод фотоэлектрический со скоростью ввода 800 знаков в секунду; ввод с перфорированной пятипозиционной ленты; выводное устройство — быстродействующий печатающий механизм — БПМ-20 — работает со скоростью 20 знаков в секунду; телетайп работает со скоростью 7 знаков в секунду; наряду с печатью можно выполнять перфорацию выводимых кодов
Потребляемая мощность	4 *квт*
Занимаемая площадь	50 $м^2$

Машина *Минск-2* универсальная, допускающая произвольный временной порядок эксплуатации.

Краткие сведения об основных устройствах машины

Запоминающие устройства. Запоминающих устройств в машине два: оперативное (ОЗУ) и внешнее (ВЗУ).

ОЗУ выполнено на магнитных сердечниках. Емкость ОЗУ 4096 37-разрядных двоичных кодов; время обращения к ОЗУ 24 *мксек*.

ОЗУ разбито на два блока, в каждом из которых может храниться одновременно 2048 37-разрядных двоичных кодов. Блоки ОЗУ имеют номера 0 и 1. Ячейки блока с номером 0 имеют восьмеричные номера $0000 \div 1777$. Из ячейки с номером 0000 всегда выдается число нуль. Ячейки блока с номером 1 имеют восьмеричные номера 2000, ... , 3777.

ВЗУ выполнено на магнитной ленте; емкость ВЗУ 400 тыс. кодов. Конструктивно ВЗУ выполнено в виде шкафа, состоящего из четырех лентопротяжных механизмов магнитной ленты (в машине предусмотрена возможность иметь несколько таких шкафов). Запись (чтение) на магнитную ленту производится только из ОЗУ. На магнитную ленту коды записываются (считываются) зонами. Наибольшее число допустимых кодов в зоне — 2048; наибольшее число зон — 49. Номерами зон могут быть двоичные числа $00000 \div 11111$; лентопротяжные механизмы имеют номера 0, 1, ... , 7; шкафы накопителя магнитных лент имеют номера 0, 1, ... , 4. Скорость движения магнитной ленты 2 *м/сек*; плотность записи 10 *импульсов/мм*.

Арифметическое устройство (АУ). Основную часть АУ составляют два регистра (r и R) и сумматор s, выполненные на триггерах.

Устройство управления содержит следующие основные регистры:

регистр-счетчик номера команд C, имеющий 13 двоичных разрядов;

регистр кода операции, имеющий 7 двоичных разрядов.

Машина *Минск-2* регистра команд не имеет. Для размещения команды используется сумматор и регистр r арифметического устройства.

Основной цикл выполнения команд машины *Минск-2* состоит из IV этапов:

Первый этап — чтение команды. Машина производит перенос команды из ячейки ОЗУ, номер которой указан в регистре C,

в сумматор, где происходит сложение адресов этой команды с «адресами» команды, находящейся в индексной ячейке, номер которой есть содержимое 9 ÷ 12 двоичных разрядов сумматора (подробное описание способа индексных ячеек приведено ниже), код операции передается в регистр кода операции устройства управления.

Второй этап — чтение первого числа. Содержимое сумматора переносится в регистр *r*, число из ячейки ОЗУ, номер которой указан в первом адресе команды, находящейся в регистре *r*, переносится в сумматор, а затем в регистр *R*.

Третий этап — чтение второго числа. Число из ячейки ОЗУ, номер которой указан во втором адресе команды, находящейся на регистре *r*, переносится в сумматор.

Четвертый этап — выполнение операции и запись результата в арифметическом устройстве. Выполняется операция с числами «*R*» и «*s*», соответствующая коду операции, указанному в регистре кода операции; результат операции записывается в регистр *R* и остается в сумматоре (в ряде случаев запись производится в ячейку памяти, номер которой указан во втором адресе команды, находящейся в *r*); содержимое регистра *C* увеличивается на единицу.

Действия выполняемые машиной, по командам условного и безусловного перехода, ввода и вывода, обращения к НМЛ и др. отличны от вышеописанных.

В машине для автоматического изменения адресов переменных команд программы, т. е. для автоматической переадресации, применяется так называемый *способ индексных ячеек*, с использованием первых 15 ячеек ОЗУ под индексные

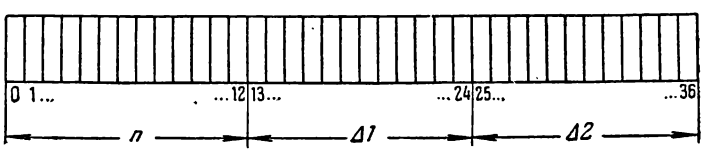


Рис. 13.1. Распределение разрядов индексной ячейки.

ячейки. Распределение разрядов индексной ячейки приведено на рис. 13.1. Переадресация по способу индексных ячеек осуществляется в машине следующим образом: перед выполнением любой команды «*K*» программы, кроме команды «конец цикла», происходит увеличение адресов команды «*K*» на соответствующие «шаги переадресации» (Δ1 и Δ2, см. рис. 13.1), указанные в индекс-ячейке, имеющей номер, равный индекс-адресу команды «*K*», т. е. равный содержимому 9 ÷ 12 разрядов команды «*K*». При этом, если код операции команды «*K*» отличен от кода операции — 40, — 41,

— 46, — 47, то выполняется блокировка единицы переноса из 25-го и 13-го разрядов соответственно в 24-й и 12-й разряды, а единица переноса из указанных разрядов логически складывается соответственно с содержимым 8-го и 7-го разрядов команды «*K*». Если же код операции команды «*K*» совпадает с одним из кодов операций — 40, — 41, — 46, — 47, то при автоматической переадресации адресов команды «*K*» единица переноса из 25-го разряда в 24-й и из 13-го в 12-й не блокируется.

УУ включает в себя и программу прерывания (см. стр. 379).

В УУ в результате выполнения команды вырабатываются сигналы ω_1, ω_2 и $\omega_3 = \varphi$, управляющие выполнением команд условного перехода. В табл. 13.1 указаны условия выработки признаков $\omega_1 = 1$, $\omega_2 = 1$ и $\varphi = 1$.

Устройство ввода и вывода (УВВ). Вводное устройство фотоэлектрическое, скорость ввода 800 *знаков/сек*. Ввод производится с перфорированной бумажной ленты, на которой цифры (буквы) кода размещаются последовательно одна за другой.

Ввод может быть двух видов: ввод по адресу и ввод по команде (подробнее см. стр. 420).

Выводное устройство предназначено для вывода кодов из ОЗУ на печать и на бумажную перфоленту (перфорация выводимой информации может производиться на одном из двух перфораторов, имеющих соответственно номера 1 и 2). Одновременная печать и перфорация на перфораторе 1 невозможна; на перфораторе 2 возможна.

Выводимая из машины информация может быть представлена в одной из следующих форм:

а) в форме числа в двоично-десятичной системе счисления с фиксированной запятой;

б) в форме числа в двоично-десятичной системе счисления с плавающей запятой;

в) в форме команды и числа в восьмеричной системе счисления;

г) в форме текста во втором международном телеграфном коде.

Вывод на печать выполняется со скоростью 20 *знаков/сек*; на перфоратор — 7 *знаков/сек*.

В машине предусмотрена возможность выдачи информации на выводные устройства и одновременного продолжения

вычислений по программе. Делается такое совмещение с помощью так называемой программы прерывания, входящей в состав устройства управления. При выполнении команды — 06 0 0 a b, находящейся в ячейке k, выполняются: команда «0020», если $a = 0001$, и команда «0021», если $a = 0002$. Если при этом в ячейке 0020 в первом случае (т. е. при $a = 0001$) или в ячейке 0021 во втором случае (т. е. при $a = 0002$) находится команда — 31 0 0 α β, то машина передает управление команде «α» и в ячейку β записывается команда возврата — 30 0 0 $k+1$ 0000. Затем выполняется команда «α» и после этого по команде «β» происходит передача управления к команде «$k+1$». И, наконец, если команда «α» есть команда выдачи на перфоратор 1 в первом случае и на перфоратор 2 во втором случае, то при выполнении команды «α» происходит совмещение выдачи числа на перфоратор и продолжение счета по программе, так как возврат по команде «β» происходит в начальный момент времени выполнения команды «α». При этом может оказаться, что команда «$k+1$» есть выдача на выходное устройство машины, или есть команда «разрешение — запрещение прерывания». Тогда в этом случае машина будет выполнять до конца только команду «α» и лишь после окончания ее выполнения перейдет к выполнению команды «$k+1$».

Представление чисел и команд

На рис. 13.2 показано распределение разрядов ячейки памяти машины *Минск-2* при хранении в ней двоичного числа с фиксированной и плавающей запятой, двоично-десятичного с фиксированной и плавающей запятой. Наличие в знаковом разряде цифры 1 соответствует знаку минус, а цифры 0 — знаку плюс. Цифровая часть числа с фиксированной запятой и мантисса числа в нормальной форме изображается в прямом коде.

На том же рисунке показано распределение разрядов ячейки памяти при хранении в ней слова в буквенном (русском или латинском) алфавите и команды первого и второго типа. Все команды в машине *Минск-2* можно разбить на два типа: команды, при выполнении которых машина выполняет обращение к магнитным лентам (команды второго типа), и все остальные команды (команды первого типа).

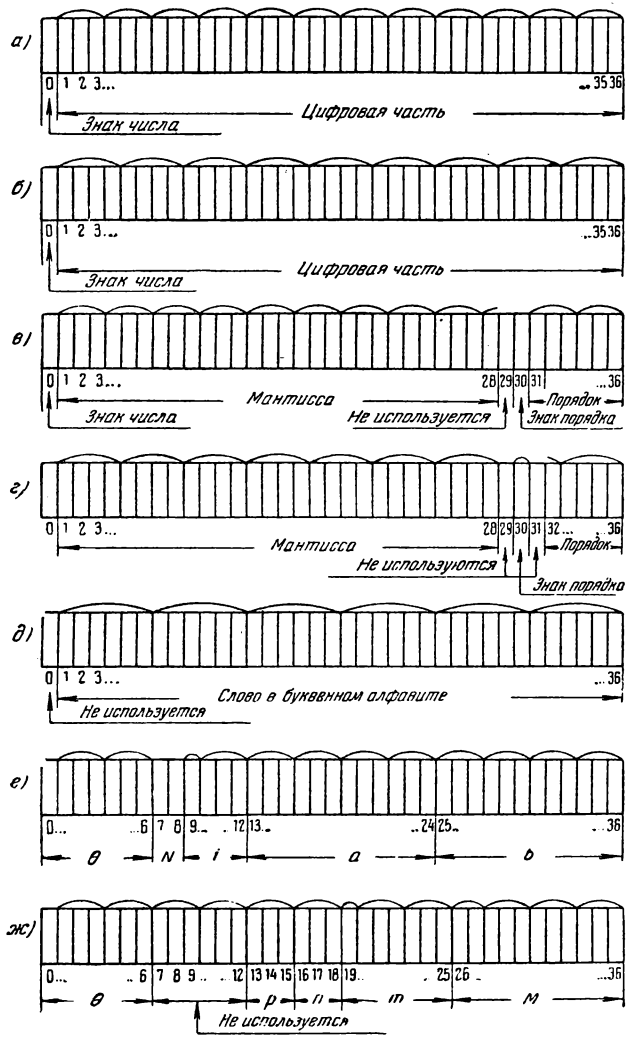


Рис. 13.2. Распределение разрядов ячейки памяти: *а*) для двоичного числа с фиксированной запятой; *б*) для двоично-десятичного числа с фиксированной запятой; *в*) для двоичного числа с плавающей запятой; *г*) для двоично-десятичного числа с плавающей запятой; *д*) для слова, состоящего из букв цифро-буквенного алфавита; *е*) для команды первого типа; *ж*) для команды второго типа.

Команды первого типа имеют структуру $\theta\ N\ i\ a\ b$, где θ — код операции, N — номер блока ОЗУ, i — индексный адрес, a и b — первый и второй адреса; команды второго типа имеют структуру $\theta\ p\ n\ m\ M$, где θ — код операции, p — номер шкафа ВЗУ, n — номер лентопротяжного механизма магнитной ленты, m — номер зоны магнитной ленты и M — порядковый номер первого числа из группы считываемых (записываемых) чисел с зоны магнитной ленты с номером m.

Скобками $\frown$ на рис. 13.2 указаны объединения двоичных разрядов в буквы буквенного алфавита, в восьмеричные и десятичные цифры, применяемые при записи текстовых кодов, чисел и команд.

Система операций и команд

Описание системы операции и команд машины *Минск-2* приведено в табл. 13.1.

В табл. 13.1 $\tilde{a}$ и $\tilde{b}$ — исполнительные адреса команды $\theta\ N\ i\ a\ b$, которые получаются путем увеличения a и b соответственно на величины i_{I} и i_{II}, хранящиеся в первом и втором адресах индексной ячейки с номером i (подробнее о способе автоматической переадресации с помощью индексных ячеек см. на стр. 377); r, R и s — соответственно первый регистр, второй регистр и сумматор арифметического устройства; $\equiv$ соответствует тому, что значение признака ω_1 (ω_2 или φ) не изменяется; A — код, у которого мантисса и знак мантиссы — нули, а порядок равен 63.

Замечания. 1. Арифметические операции над числами с плавающей запятой выполняются правильно, если числа нормализованные. Результат арифметических операций — нормализованное число.

2. Машина при выполнении команд с кодом операции $+00$, $+01$, $+02$ и $+03$ оставляет содержимое регистра R прежним и переходит к выполнению следующей команды; при выполнении команды с кодом операции, не указанным в табл. 13.1 и отличным от $+00$, $+01$, $+02$ и $+03$, машина останавливается.

3. Команда «Снятие блокировки прерывания» используется в программе прерывания. Однако не рекомендуется ею пользоваться вне программы прерывания, так как особенности технического исполнения могут приводить к ошибкам, связанным с округлением результата вычислений.

Таблица 13.1. Система операций и команд машины *Минск-2*

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+04	Поразрядное отрицание равнозначности	+04 *N i a b*	$s_i := \tilde{b}_i \approx \tilde{a}_i$ $(i = 0,1,\ldots,36)$, $\tilde{b} := s$, $R := s$	$s = 0$			Производится поразрядное отрицание равнозначности содержимого ячеек $\tilde{a}$ и $\tilde{b}$	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	120
+05		+05 *N i a b*	$s_i := \tilde{b}_i \approx \tilde{a}_i$ $(i = 0,1,\ldots,36)$, $R := s$					Результат записывается в регистр R	96
+06		+06 *N i a b*	$s_i := R_i \approx \tilde{a}_i$ $(i = 0,1,\ldots,36)$, $\tilde{b} := s$, $R := s$				Производится поразрядное отрицание равнозначности содержимого регистра R и ячейки $\tilde{a}$	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	120
+07		+07 *N i a b*	$s_i := R_i \approx \tilde{a}_i$ $(i = 0,1,\ldots,36)$, $R := s$					Результат записывается в регистр R	96

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+10	Сложение с фиксированной запятой	+10 *N i a b*	$s := \tilde{b} + \tilde{a}$, $\tilde{b} := s$, $R := s$	$s = 0$		$\lvert s \rvert \geqslant 1$	Производится алгебраическое сложение чисел «$\tilde{b}$» и «$\tilde{a}$» с фиксированной запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	120
+11		+11 *N i a b*	$s := \tilde{b} + \tilde{a}$, $R := s$					Результат операции записывается в регистр R	96
+12		+12 *N i a b*	$s := R + \tilde{a}$, $\tilde{b} := s$, $R := s$				Производится алгебраическое сложение чисел «R» и «$\tilde{a}$» с фиксированной запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	120
+13		+13 *N i a b*	$s := R + \tilde{a}$, $R := s$					Результат операции записывается в регистр R	96

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+14	Сложение с плавающей запятой	+14 *N i a b*	$s := \tilde{b} + \tilde{a}$, $\tilde{b} := s$, $R := s$		$s = A$	$s_p > 63$	Производится алгебраическое сложение чисел «$\tilde{a}$» и «$\tilde{b}$» с плавающей запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	194÷334
+15		+15 *N i a b*	$s := b + \tilde{a}$, $R := s$					Результат операции записывается в регистр R	170÷310
+16		+16 *N i a b*	$s := R + \tilde{a}$, $\tilde{b} := s$, $R := s$				Производится алгебраическое сложение чисел «R» и «$\tilde{a}$» с плавающей запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	170÷310
+17		+17 *N i a b*	$s := R + \tilde{a}$, $R := s$					Результат операции записывается в регистр R	146÷286

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+20	Вычитание с фиксированной запятой	+20 *N i a b*	$s := \tilde{b} - \tilde{a}$, $\tilde{b} := s$, $R := s$	$s = 0$		$\lvert s \rvert \geqslant 1$	Из числа «$\tilde{b}$» вычитается число «$\tilde{a}$». Операция выполняется над числами с фиксированной запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	194 ÷ 334
+21		+21 *N i a b*	$s := \tilde{b} - \tilde{a}$, $R := s$					Результат операции записывается в регистр R	96
+22		+22 *N i a b*	$s := R - \tilde{a}$, $\tilde{b} := s$, $R := s$				Из числа «R» вычитается число «$\tilde{a}$». Операция выполняется над числами с фиксированной запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	96
+23		+23 *N i a b*	$s = R - \tilde{a}$, $R := s$					Результат операции записывается в регистр R	120

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\omega_3 = 1$	Пояснения		Время выполнения операции, *мксек*
+24	Вычитание с плавающей запятой	+24 *N i a b*	$s := b - \tilde{a}$, $\tilde{b} := s$, $R := s$		$s = A$	$s_p > 63$	Из числа «$\tilde{b}$» вычитается число «$\tilde{a}$». Операция выполняется над числами с плавающей запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	194÷334
+25		+25 *N i a b*	$s := b - \tilde{a}$, $R := s$					Результат операции записывается в регистр R	170÷310
+26		+26 *N i a b*	$s := R - \tilde{a}$, $\tilde{b} := s$, $R := s$				Из числа «R» вычитается число «$\tilde{a}$». Операция выполняется над числами с плавающей запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	170÷310
+27		+27 *N i a b*	$s := R - \tilde{a}$, $R := s$					Результат операции записывается в регистр R	146÷286

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+30	Умножение с фиксированной запятой	+30 *N i a b*	$s = \tilde{b} \cdot \tilde{a}$, $\tilde{b} := s$, $R := s$	$s = 0$			Производится умножение чисел «$\tilde{b}$» и «$\tilde{a}$». Операция выполняется над числами с фиксированной запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	208÷436
+31		+31 *N i a b*	$s := b \cdot \tilde{a}$, $R := s$					Результат записывается в регистр R	184÷412
+32		+32 *N i a b*	$s := R \cdot \tilde{a}$, $\tilde{b} := s$, $R := s$				Производится умножение чисел «R» и «$\tilde{a}$». Операция выполняется над числами с фиксированной запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	184÷412
+33		+33 *N i a b*	$s := R \cdot \tilde{a}$, $R := s$					Результат операции записывается в регистр R	160÷398

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+34	Умножение с плавающей запятой	+34 *N i a b*	$s := \tilde{b} \cdot \tilde{a}$, $\tilde{b} := s$, $R := s$		$s = A$	$s_p > 63$	Производится умножение чисел «$\tilde{b}$» и «$\tilde{a}$». Операция выполняется над числами с плавающей запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	228÷400
+35		+35 *N i a b*	$s := \tilde{b} \cdot \tilde{a}$, $R := s$					Результат операции записывается в регистр R	204÷376
+36		+36 *N i a b*	$s := R \cdot \tilde{a}$, $\tilde{b} := s$, $R := s$				Производится умножение чисел «R» и «$\tilde{a}$». Операция выполняется над числами с плавающей запятой	Результат операции записывается в регистр R и ячейку $\tilde{b}$	204÷376
+37		+37 *N i a b*	$s := R \cdot \tilde{a}$, $R := s$					Результат операции записывается в регистр R	180÷352

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+40	Деление с фиксированной запятой	+40 *N i a b*	$s := \tilde{b} : \tilde{a}$, $\tilde{b} := s$, $R := s$	$s = 0$		$\lvert s \rvert \geqslant 1$ или $\tilde{a} = 0$	Число «$\tilde{b}$» делится на число «$\tilde{a}$». Операция выполняется над числами с фиксированной запятой	Результат операции заносится в ячейку $\tilde{b}$ и регистр R	740
+41		+41 *N i a b*	$s := \tilde{b} : \tilde{a}$, $R := s$					Результат операции записывается в регистр R	716
+42		+42 *N i a b*	$s := R : \tilde{a}$, $\tilde{b} := s$, $R := s$				Число «R» делится на число «$\tilde{a}$». Операция выполняется над числами с фиксированной запятой	Результат операции заносится в ячейку $\tilde{b}$ и регистр R	716
+43		+43 *N i a b*	$s := R : \tilde{a}$, $R := s$					Результат операции записывается в регистр R	692

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+44	Деление с плавающей запятой	+44 *N i a b*	$s := \tilde{b} : \tilde{a}$, $\tilde{b} := s$, $R := s$		$s = A$	$s_p > 63$ или если «$\tilde{b}$»—ненормализованное число	Число «$\tilde{b}$» делится на число «$\tilde{a}$». Операция выполняется над числами с плавающей запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	620÷628
+45		+45 *N i a b*	$s := \tilde{b} : \tilde{a}$, $R := s$					Результат операции записывается в регистр R	596÷604
+46		+46 *N i a b*	$s := R : \tilde{a}$, $\tilde{b} := s$, $R := s$				Число «R» делится на число «$\tilde{a}$». Операция выполняется над числами с плавающей запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	596÷604
+47		+47 *N i a b*	$s := R : \tilde{a}$, $R := s$					Результат операции записывается в регистр R	572÷580

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+50	Вычитание модулей с фиксированной запятой	+50 *N i a b*	$\underline{s} := \lvert\tilde{b}\rvert - \lvert\tilde{a}\rvert$, $\tilde{b} := s$, $R := s$	$s = 0$			Из абсолютной величины числа $\lvert\langle\tilde{b}\rangle\rvert$ вычитается абсолютная величина $\lvert\langle\tilde{a}\rangle\rvert$. Операция выполняется над числами с фиксированной запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	120
+51		+51 *N i a b*	$s := \lvert\tilde{b}\rvert - \lvert\tilde{a}\rvert$, $R := s$					Результат операции записывается в регистр R	96
+52		+52 *N i a b*	$\underline{s} := \lvert R\rvert - \lvert\tilde{a}\rvert$, $\tilde{b} := s$, $R := s$				Из абсолютной величины числа $\lvert\langle R\rangle\rvert$ вычитается абсолютная величина числа $\lvert\langle\tilde{a}\rangle\rvert$. Операция выполняется над числами с фиксированной запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	96
+53		+53 *N i a b*	$s := \lvert R\rvert - \lvert\tilde{a}\rvert$, $R := s$					Результат операции записывается в регистр R	72

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды		Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+54	Вычитание модулей с плавающей запятой	+54 *N i a b*	$\underline{s} := \lvert \tilde{b} \rvert - \lvert \tilde{a} \rvert$, $\tilde{b} := s$, $R := s$		$s = A$		Из абсолютной величины числа $\lvert «\tilde{b}» \rvert$ вычитается абсолютная величина числа $\lvert «\tilde{a}» \rvert$. Операция выполняется над числами с плавающей запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	194÷334
+55		+55 *N i a b*	$\underline{s} := \lvert \tilde{b} \rvert - \lvert \tilde{a} \rvert$, $R := s$					Результат операции записывается в регистр R	170÷310
+56		+56 *N i a b*	$\underline{s} := \lvert R \rvert - \lvert \tilde{a} \rvert$, $\tilde{b} := s$, $R := s$				Из абсолютной величины числа $\lvert «R» \rvert$ вычитается абсолютная величина числа $\lvert «\tilde{a}» \rvert$. Операция выполняется над числами с плавающей запятой	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	170÷310
+57		+57 *N i a b*	$\underline{s} := \lvert R \rvert - \lvert \tilde{a} \rvert$, $R := s$					Результат операции записывается в регистр R	146÷286

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1=1$	Условия выработки $\omega_2=1$	Условия выработки $\varphi=1$	Пояснения		Время выполнения операции, *мксек*
+60	Сдвиг 1	+60 *N i a b*	$s_i := \begin{cases} \tilde{a}_{i+n}, & 0 \leqslant i+n \leqslant 36, \\ 0, & i+n<0 \text{ или} \\ & i+n>36 \end{cases}$ $(0 \leqslant i \leqslant 36;\ n = \langle\tilde{b}_{\text{р}}\rangle)$, $R := s$, $\tilde{b} := s$	$s=0$			Содержимое всех разрядов ячейки $\tilde{a}$ сдвигается на $\lvert\langle\tilde{b}_{\text{р}}\rangle\rvert$ разрядов влево, если $\langle\tilde{b}_{\text{р}}\rangle > 0$, и вправо, если $\langle\tilde{b}_{\text{р}}\rangle < 0$	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	$132 \div 4 \times \times \lvert\langle\tilde{b}_{\text{р}}\rangle\rvert$
+61		+61 *N i a b*	$s_i := \begin{cases} \tilde{a}_{i+n}, & 0 \leqslant i+n \leqslant 36, \\ 0, & i+n<0 \text{ или} \\ & i+n>36 \end{cases}$ $(0 \leqslant i \leqslant 36;\ n = \langle\tilde{b}_{\text{р}}\rangle)$, $R := s$					Результат операции записывается в регистр R	$108 \div 4 \times \times \lvert\langle\tilde{b}_{\text{р}}\rangle\rvert$
+62		+62 *N i a b*	$s_i := \begin{cases} R_{i+n}, & 0 \leqslant i+n \leqslant 36, \\ 0, & i+n>36 \text{ или} \\ & i+n<0 \end{cases}$ $(0 \leqslant i \leqslant 36;\ n = \langle\tilde{b}_{\text{р}}\rangle)$, $R := s$, $\tilde{b} := s$				Содержимое всех разрядов регистра R сдвигается на $\lvert\langle\tilde{b}_{\text{р}}\rangle\rvert$ разрядов влево, если $\langle\tilde{b}_{\text{р}}\rangle > 0$, и вправо, если $\langle\tilde{b}_{\text{р}}\rangle < 0$	Результат операции записывается в ячейку $\tilde{b}$ и в регистр R	$108 \div 4 \times \times \lvert\langle\tilde{b}_{\text{р}}\rangle\rvert$
+63		+63 *N i a b*	$s_i := \begin{cases} R_{i+n}, & 0 \leqslant i+n \leqslant 36, \\ 0, & i+n<0 \text{ или} \\ & i+n>36 \end{cases}$ $(0 \leqslant i \leqslant 36;\ n = \langle\tilde{b}_{\text{р}}\rangle)$, $R := s$					Результат операции записывается в регистр R	$84 \div 4 \times \times \lvert\langle\tilde{b}_{\text{р}}\rangle\rvert$

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условая выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+64	Сдвиг 2	+64 *N i a b*	$s_i := \begin{cases} \tilde{a}_{i+n}, & 1 \leqslant i+n \leqslant 36, \\ 0, & i+n < 1 \text{ или} \\ & i+n > 36 \end{cases}$ $(1 \leqslant i \leqslant 36;\ n = \langle \tilde{b}_p \rangle)$ $R := s,$ $\tilde{b} := s$				Содержимое 1÷36-го разрядов ячейки $\tilde{a}$ сдвигается на $\lvert \langle \tilde{b}_p \rangle \rvert$ разрядов влево, если $\langle \tilde{b}_p \rangle > 0$, и вправо, если $\langle \tilde{b}_p \rangle < 0$	Результат операции записывается в ячейку $\tilde{b}$ и в регистр R	132÷4× ×$\lvert \langle \tilde{b}_p \rangle \rvert$
+65		+65 *N i a b*	$s_i := \begin{cases} \tilde{a}_{i+n}, & 1 \leqslant i+n \leqslant 36, \\ 0, & i+n < 1 \text{ или} \\ & i+n > 36 \end{cases}$ $(1 \leqslant i \leqslant 36;\ n = \langle \tilde{b}_p \rangle)$ $R := s$					Результат операции записывается в регистр R	108÷4× ×$\lvert \langle \tilde{b}_p \rangle \rvert$
+66		+66 *N i a b*	$s_i := \begin{cases} R_{i+n}, & 1 \leqslant i+n \leqslant 36, \\ 0, & i+n < 1 \text{ или} \\ & i+n > 36 \end{cases}$ $(1 \leqslant i \leqslant 36;\ n = \langle \tilde{b}_p \rangle)$ $R := s$ $\tilde{b} := s$				Содержимое 1÷36-го разрядов регистра R сдвигается на $\lvert \langle \tilde{b}_p \rangle \rvert$ разрядов влево, если $\langle \tilde{b}_p \rangle > 0$, и вправо, если $\langle \tilde{b}_p \rangle < 0$	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	108÷4× × $\tilde{b}_p$
+67		+67 *N i a b*	$s_i := \begin{cases} R_{i+n}, & 1 \leqslant i+n \leqslant 36, \\ 0, & i+n < 1 \text{ или} \\ & i+n > 36 \end{cases}$ $(1 \leqslant i \leqslant 36;\ n = \langle \tilde{b}_p \rangle)$ $R := s$					Результат операции записывается в регистр R	84÷4× × $\tilde{b}_p$

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения		Время выполнения операции, *мксек*
+70	Логическое умножение	+70 *N i a b*	$s_i := \tilde{b}_i \wedge \tilde{a}_i$ $(i = 0,1,\ldots,36)$, $\tilde{b} := s$, $R := s$	$s = 0$		Производится поразрядное логическое умножение содержимого ячеек $\tilde{b}$ и $\tilde{a}$	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	120
+71		+71 *N i a b*	$s_i := \tilde{b}_i \wedge \tilde{a}_i$ $(i = 0,1,\ldots,36)$, $R := s$				Результат операции записывается в регистр R	96
+72		+72 *N i a b*	$s_i := R_i \wedge \tilde{a}_i$ $(i = 0,1,\ldots,36)$, $\tilde{b} := s$, $R := s$			Производится поразрядное логическое умножение содержимого регистра R и ячейки $\tilde{a}$	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	96
+73		+73 *N i a b*	$s_i := R_i \wedge \tilde{a}_i$ $(i = 0,1,\ldots,36)$, $R := s$				Результат операции записывается в регистр R	72

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\omega_3 = 1$	Пояснения		Время выполнения операции, *мксек*
+74	Логическое сложение: +74 $N\,i\,a\,b$	$s_i := \tilde{b}_i \vee \tilde{a}_i$ $(i = 0, 1, \ldots, 36)$, $\tilde{b} := s$, $R := s$	$s = 0$			Производится поразрядное логическое сложение содержимого ячеек $\tilde{b}$ и $\tilde{a}$	Результат операции записывается в ячейку $\tilde{b}$ и регистр R	120
+75	+75 $N\,i\,a\,b$	$s_i := \tilde{b}_i \vee \tilde{a}_i$ $(i = 0, 1, \ldots, 36)$, $R := s$					Результат операции записывается в регистр R	96
+76	+76 $N\,i\,a\,b$	$s_i := R_i \vee \tilde{a}_i$ $(i = 0, 1, \ldots, 36)$, $\tilde{b} := s$, $R := s$				Производится поразрядное логическое сложение содержимого регистра R и ячейки $\tilde{a}$	Результат операции записывается в ячейку $\tilde{b}$ и в регистр R	96
+77	+77 $N\,i\,a\,b$	$s_i := R_i \vee \tilde{a}_i$ $(i = 0, 1, \ldots, 36)$, $R := s$					Результат операции записывается в регистр R	72

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—00	Останов	$R := s$, $r := \tilde{a}$, $s := \tilde{b}$				Содержимое сумматора переносится в регистр R, содержимое ячеек $\tilde{a}$ и $\tilde{b}$ переносится соответственно в регистр r и сумматор s. Затем машина останавливается	96
—04	Блокировка округления —04 $N\ i\ a\ b$	$s := s$, $R := R$				Производится блокировка округления, т. е. дальнейшее выполнение операции по командам программы будет производиться без округления. Значения N, i, a и b на выполнение операции не влияют	24
—05	Снятие блокировки округления —05 $N\ i\ a\ b$	$s := s$, $R := R$				Производится разблокировка округления, т. е. дальнейшее выполнение операции по командам программы осуществляется с округлением результатов. Значения N, i, a и b на выполнение команды не влияют	24

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—06	Разрешение—запрещение прерывания —06 $N\,i\,a\,b$	$r := r$, $R := R$				Если $a = 0001$ или $a = 0002$, то машина выполняет следующие операции: а) разрешает прерывание; б) выполняет команду «0020», если $a = 0001$ и команду «0021», если $a = 0002$. (Если при этом в ячейке 0020 в первом случае или 0021 во втором случае находится команда — 31 0 0 α β, то происходит передача управления команде «α», а в ячейку β записывается команда —30 0 0 $k+1$ 0000. Если к тому же в ячейке α находится команда выдачи на перфоратор 1 в первом случае и на перфоратор 2 во втором, то, перейдя к команде «α», машина производит выдачу числа на соответствующий перфоратор с прерыванием, т. е. по команде «β» машина переходит к выполнению команды «$k+1$».) Если	24

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
						$a \neq 0001$ или $a \neq 0002$, то происходит запрещение прерывания и машина переходит к выполнению следующей команды	
—07	Реверс перфоленты —07 *N i a b*					На устройстве ввода производится обратная перемотка перфоленты на участок, заключенный между двумя знаками «граница ввода». Одновременно машина продолжает выполнение команд программы. Значения *N*, *i*, *a* и *b* на выполнение команды не влияют	2
—10	Перенос числа —10 *N i a b*	$s := \tilde{a}$, $\tilde{b} := s$, $R := s$	$s = 0$			Число «$\tilde{a}$» переносится в ячейку $\tilde{b}$ и регистр *R*	96
—11	Изменение знака числа —11 *N i a b*	$s := \overline{\operatorname{sign} \tilde{a}} \mid \tilde{a} \mid$, $\tilde{b} := s$, $R := s$	$s = 0$			Число «$\tilde{a}$» с обратным знаком переносится в регистр *R* и в ячейку $\tilde{b}$	96

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—12	Вычисление модуля числа —12 *N i a b*	$\underline{s} := \lvert \tilde{a} \rvert$, $\tilde{b} := s$, $R := s$	$s = 0$			Модуль числа «$\tilde{a}$» пересылается в регистр R и ячейку $\tilde{b}$	96
—13	Перенос числа с клавишного регистра пульта управления в ячейку памяти —13 *N i a b*	$\underline{s} := R^*$, $\tilde{b} := s$, $R := s$	$s = 0$			Производится пересылка двоичного 37-разрядного кода, набранного на клавишном регистре R^* пульта управления, в сумматор, регистр R и ячейку $\tilde{b}$	96
—14	Перенос числа с присвоением знака другого числа —14 *N i a b*	$\underline{s} := \operatorname{sign} \tilde{b} \lvert \tilde{a} \rvert$, $\tilde{b} := s$, $R := s$	$s = 0$			В регистр R и ячейку $\tilde{b}$ записывается число, имеющее модуль числа «$\tilde{a}$» и знак числа «$\tilde{b}$»	120

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—15	Перенос числа из регистра R в ячейку памяти —15 0 0 *a b*	$\tilde{b} := R$, $s := s$				Содержимое регистра R переносится в ячейку $\tilde{b}$. Содержимое сумматора не изменяется. Значение величины *a* на результат операции не влияет	
—16	Присвоение числу порядка другого числа —16 *N i a b*	$\underline{s} := \tilde{b}_м \cdot 2^{\tilde{a}_{p_s}}$, $\tilde{b} := s$, $R := s$				Порядок числа «$\tilde{b}$» заменяется на порядок числа «$\tilde{a}$», результат операции записывается в регистр R и засылается в ячейку $\tilde{b}$	120
—20	Конец цикла —20 *N i a b*					«Адреса» числа «*i*» увеличиваются на соответствующие «адреса» числа «*b*»; переносы из старших разрядов адресов блокируются. Одновременно из числа, находящегося в старших 13 разрядах ячейки *i*, вычитается единица. Если результат вычитания оказывается	120

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\rho = 1$	Пояснения	Время выполнения операции, *мксек*
						больше или равен нулю, то происходит передача управления к команде «*a*» и запись числа «*s*» в ячейку *i*, если результат вычитания меньше нуля, то происходит переход к «*k* + 1» команде	
—30 или —37	Безусловный переход —30 *N i a b* или —37 *N i a b*	$C := \tilde{a}$, $\tilde{b} := R$	≡	≡		Происходит передача управления команде «$\tilde{a}$» и одновременно содержимое регистра *R* пересылается в ячейку $\tilde{b}$	72
—31	Безусловный переход с возвратом —31 *N i a b*	$C := \tilde{a}$, «$\tilde{b}$» := 0 0 *C* 0000	≡	≡		Происходит передача управления команде «$\tilde{a}$» и одновременно в ячейку $\tilde{b}$ записывается «команда возврата», т. е. команда —30 0 0 *C* 0000	72

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—32	Условный переход по ω_2 —32 *N i a b*	$C := \tilde{a}$, если $\omega_2 = 0$; $C := \tilde{b}$, если $\omega_2 = 1$; $R := R$	≡	≡		Происходит передача управления команде «$\tilde{a}$», если $\omega_2 = 0$ и команде «$\tilde{b}$», если $\omega_2 = 1$	48
—33	Условный переход по φ —33 *N i a b*	$C := \tilde{a}$, если $\varphi = 0$; $C := \tilde{b}$, если $\varphi = 1$; $R := R$	≡	≡		Происходит передача управления команде «$\tilde{a}$», если $\varphi = 0$ и команде «$\tilde{b}$», если $\varphi = 1$; признак φ принимает значение, равное нулю; содержимое регистра R не изменяется	48
—34	Условный переход по ω_1 —34 *N i a b*	$C := \tilde{a}$, если $\omega_1 = 0$; $C := \tilde{b}$, если $\omega_1 = 1$; $R := R$	≡	≡		Передается управление команде «$\tilde{a}$», если $\omega_1 = 0$ и команде «$\tilde{b}$», если $\omega_1 = 1$. Содержимое регистра R не изменяется	48

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—35	Условный переход по ключу —35 *N i a b*	$C := \tilde{b}$, если ключ с номером a включен; $C := C + 1$, если выключен; $R := R$	$\equiv$	$\equiv$		Передается управление команде «$\tilde{b}$», если на пульте управления включен ключ с номером a ($a = 0001$, 0002, 0004, 0010, 0020, 0040, 0100) и следующей команде, т. е. команде «$k+1$», если ключ с номером a выключен. Если a есть сумма номеров ключей (например, $a = 0046$ — сумма номеров ключей 0002; 0004 и 0040), то происходит передача управления команде «$\tilde{b}$» при включенном хотя бы одном из этих ключей. Содержимое регистра R не изменяется	48
—36	Безусловный переход с разблокировкой прерывания —36 *N i a b*	$C := \tilde{a}$, $R := \tilde{b}$	$s = 0$			Число «$\tilde{b}$» переносится в регистр R; происходит передача управления команде «$\tilde{a}$» и снятие блокировки прерывания	72

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—40	Подвод зоны магнитной ленты в прямом направлении —40 *p n m M*					Магнитная лента лентопротяжного механизма с номером *n* в шкафу НМЛ с номером *p* начинает двигаться в прямом направлении с одновременным поиском зоны с номером *m* на этой ленте. Одновременно с движением магнитной ленты происходит дальнейшее выполнение команд программы. Движение магнитной ленты прекращается, как только будет найдено число с номером *M* на зоне с номером *m*	От 50 *мсек* до 40 *сек*
—41	Подвод зоны магнитной ленты в обратном направлении —41 *p n m M*					Команда выполняется точно так же, как и команда с кодом операции $\theta = -40$, с той лишь разницей, что движение ленты происходит в обратном направлении	От 50 *мсек* до 40 *сек*

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции; *мксек*
—46 —45	Перенос чисел с магнитной ленты в ячейки ОЗУ при движении магнитной ленты в прямом направлении —46 $p\ n\ m\ M$ —45 $N\ i\ a\ b$					a чисел с зоны с номером m магнитной ленты лентопротяжного механизма с номером n в шкафу НМЛ с номером p переносятся, начиная с числа с номером M, в ячейки ОЗУ $\tilde{b}$, $\tilde{b}+1, \ldots, \tilde{b}+\tilde{a}$. При этом лента движется в прямом направлении. По окончании переписи в сумматоре s образуется контрольная сумма этих чисел, полученная с помощью операции сложения с циклическим переносом единицы из старшего разряда в младший. При переносе чисел с магнитной ленты в ячейки ОЗУ происходит контроль правильности считывания информации с магнитной ленты (чисел, номеров зоны или информации со служебных дорожек) и если обнаружится сбой в работе	От 50 *мсек* до 40 *сек*

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
						машины, то происходит передача управления команде с номером $k+2$; если же такого сбоя не обнаружится, происходит передача управления команде с номером $k+3$. (Здесь k и $k+1$ — номера ячеек, в которых размещаются рассматриваемые команды.)	
—47 —45	Перенос чисел с магнитной ленты в ячейки памяти ОЗУ при движении магнитной ленты в обратном направлении —47 *p n m M* —45 *N i a b*					Команды выполняются так же, как и команда «Перенос чисел с магнитной ленты в ячейки памяти ОЗУ при движении магнитной ленты в прямом направлении», с этой лишь разницей, что магнитная лента движется в обратном направлении	От 50 *мсек* до 40 *сек*

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—46 —44	Контрольное суммирование чисел, считываемых с магнитной ленты при движении ее в прямом направлении —46 *p n m M* —44 *N i a b*					*a* чисел, начиная с числа с номером *M*, с зоны с номером *m* магнитной ленты, лентопротяжного механизма с номером *n*, шкафа НМЛ с номером *p*, переносятся последовательно в сумматор, образуя сумму этих чисел, при этом используется операция суммирования с циклическим переносом единицы из старшего разряда в младший; движение ленты происходит в прямом направлении. При переносе чисел с магнитной ленты в сумматор происходит контроль правильности считывания информации (см. код операции —46 / —45). На выполнение команды значение *b* не влияет	От 50 *мсек* до 40 *сек*

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—47 —44	Контрольное суммирование чисел, считываемых с магнитной ленты при движении ее в обратном направлении —47 *p n m M* —44 *N i a b*					Команда выполняется так же, как и команда с кодом операции { —46 / —44, с той лишь разницей, что движение магнитной ленты производится в обратном направлении	От 50 *мсек* до 40 *сек*
—50	Ввод чисел с перфоленты —50 *N i a b*					В ячейки $\tilde{b}$, $\tilde{b}+1$, ... ОЗУ вводятся числа с перфоленты устройства ввода. Однако, если на перфоленте информация подготовлена для «ввода по адресу» или для группового «ввода по адресу», то ввод производится в ячейки, номера которых указаны на перфоленте. Ввод прекращается, как только с перфоленты будет считана комбинация «граница ввода». Значение величины *a* на выполнение команды не влияет	От 50 *мсек* до 40 *сек*

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—51	Контрольное суммирование чисел, считываемых с перфоленты при числовом вводе —51 *N i a b*					С перфоленты устройства ввода числа, находящиеся между двумя комбинациями «Граница ввода», переносятся в сумматор, при этом образуется их сумма (используется операция сложения с циклическим переносом единицы из старшего разряда в младший). Значения *N*, *i*, *a* и *b* на выполнение команды не влияют	800 *строк/сек*
—52	Буквенно-цифровой ввод с перфоленты —52 *N i a b*					Производится буквенный ввод с перфоленты устройства ввода в ячейки ОЗУ, начиная с ячейки *b*. При этом каждый считанный знак с перфоленты преобразуется в шестиразрядное двоичное «слово» с последующей записью в младшие шесть разрядов входного регистра. После этого содер-	800 *строк/сек*

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
						жимое входного регистра сдвигается на 6 разрядов влево и снова производится считывание с перфоленты следующей комбинации, и т. д. После шестикратного повторения этого цикла содержимое входного регистра записывается в ячейку памяти, при этом каждый раз к содержимому сумматора с помощью операции сложения с циклическим переносом единицы из старших разрядов в младшие прибавляется содержимое входного регистра и таким образом накапливается контрольная сумма вводимой информации. Ввод прекращается, как только считается комбинация «Граница ввода». Значение *a* на выполнение команды не влияет	

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—53	Контрольное суммирование чисел, считываемых с перфоленты при буквенно-цифровом вводе —53 *N i a b*					Команда выполняется аналогично команде с $\theta = -52$, с той лишь разницей, что записи в ячейки памяти не производится	800 *строк/сек*
—60	Вывод на БПМ-20 или перфоратор 1 —60 *N i a b*					Содержимое 1÷6-го разрядов ячейки $\tilde{b}$ или содержимое всей ячейки $\tilde{b}$ выводится на печатающее устройство БПМ-20 или выходной перфоратор 1 в зависимости от значения величины a: $a = 1000$ — производится вывод содержимого ячейки $\tilde{b}$ на перфоратор 1 в восьмеричной системе счисления; $a = 2000$ — производится вывод содержимого ячейки $\tilde{b}$ на перфоратор 1 в десятичной системе счисления;	

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
						$a = 3000$ — на перфоленте перфоратора 1 пробивается комбинация «запись»; $a = 4000$ — содержимое 1÷6-го разрядов выводится на перфоратор 1 в русском алфавите; $a = 6000$ — содержимое 1÷6-го разрядов выводится на перфоратор 1 в латинском алфавите; $a = 0400$ — число «$\tilde{b}$» печатается на БПМ-20 в десятичной системе счисления с плавающей запятой; $a = 1400$ — число «$\tilde{b}$» печатается на БПМ-20 в восьмеричной системе счисления; $a = 2400$ — число «$\tilde{b}$» печатается на БПМ-20 в десятичной системе счисления с фиксированной запятой; $a = 3400$ — производится поворот рулона на БПМ-20 на один интервал	50÷100 *мсек* на знак 50÷100 *мсек* на знак 50÷100 *мсек*

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—61	Вывод на телетайп или перфоратор 2 —61 *N i a b*					Содержимое 1÷6-го разрядов ячейки $\tilde{b}$ выводится на телетайп или на перфоратор 2 в зависимости от значения величины *a*: $a = 4400$ — на телетайп в русском алфавите; $a = 6400$ — на телетайп в латинском алфавите; $a = 4000$ — на перфоратор 2 в русском алфавите; $a = 6000$ — на перфоратор 2 в латинском алфавите	150 150 50 50
—70	Вычисление младших разрядов произведения —70 *N i a b*				$s \neq 0$	Число «$\tilde{b}$» умножается на число «$\tilde{a}$». Операция выполняется над числами с фиксированной запятой; результат операции — число с удвоенным числом разрядов (старшие разряды числа находятся в сумматоре *s*, младшие — в регистре *R*). Режим округления на операцию не влияет	184÷412

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1=1$	Условия выработки $\omega_2=1$	Условия выработки $\varphi=1$	Пояснения	Время выполнения операции, *мксек*
—71	Вычисление остатка от деления —71 *N i a b*				$\lvert R\rvert\geqslant 1$ или $a=0$	Число «$\tilde{b}$» делится на число «$\tilde{a}$». Операция выполняется над числами с фиксированной запятой; результат операции — остаток от деления числа «$\tilde{b}$» на число «$\tilde{a}$», знак остатка совпадает со знаком числа «$\tilde{b}$» (результат операции размещается в регистре R). Режим округления на операцию не влияет	716
—72	Сложение порядков —72 *N i a b*	$s:=\tilde{b}_{\text{м}}\times$ $\times 2^{\tilde{b}_{\text{р}}+\tilde{a}_{\text{р}}}$, $\tilde{b}:=s$, $R:=s$	$\tilde{b}_{\text{р}}+\tilde{a}_{\text{р}}<$ <-63		$\tilde{b}_{\text{р}}+\tilde{a}_{\text{р}}>$ >63	Порядок числа «$\tilde{b}$» складывается с порядком числа «$\tilde{a}$». В ячейку $\tilde{b}$ записывается число, имеющее мантиссу числа «$\tilde{b}$» и порядок, равный «$\tilde{b}_{\text{р}}$»+«$\tilde{a}_{\text{р}}$»	120
—73	Вычитание порядков —73 *N i a b*	$s:=\tilde{b}_{\text{м}}\times$ $\times 2^{\tilde{b}_{\text{р}}-\tilde{a}_{\text{р}}}$, $\tilde{b}:=s$, $R:=s$	$\tilde{b}_{\text{р}}-\tilde{a}_{\text{р}}<$ <-63		$\tilde{b}_{\text{р}}-\tilde{a}_{\text{р}}>$ >63	Из порядка числа «$\tilde{b}$» вычитается порядок числа «$\tilde{a}$». В ячейку $\tilde{b}$ записывается число, имеющее мантиссу числа «$\tilde{b}$» и порядок, равный «$\tilde{b}_{\text{р}}$»+«$\tilde{a}_{\text{р}}$»	120

Продолжение табл. 13.1

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega_1 = 1$	Условия выработки $\omega_2 = 1$	Условия выработки $\varphi = 1$	Пояснения	Время выполнения операции, *мксек*
—74	Контрольное суммирование —74 *N i a b*	$s := \tilde{b} \boxplus\limits_{37} \tilde{a}$, $\tilde{b} := s$, $R := s$				Число «$\tilde{b}$» складывается по всем разрядам с числом «$\tilde{a}$» с циклическим переносом единицы из нулевого разряда в 36-й разряд. Результат записывается в ячейку $\tilde{b}$	120
—75	Нормализация —75 *N i a b*		$R = 0$			Содержимое ячейки $\tilde{a}$ нормализуется. Результат операции записывается в регистр R и ячейку $\tilde{b}+1$. В разряды 30÷36-й ячейки $\tilde{b}$ записывается число, определяющее количество сдвигов числа «$\tilde{a}$» влево, необходимых для нормализации этого числа	$120 \div 4 \times$ $\times$ «$\tilde{b}_p$»
—76	Сумма разрядов ячейки памяти —76 *N i a b*	$s := \sum\limits_{i=0}^{36} \tilde{a}_i$, я $(\tilde{b}+1) := s$, $R := s$				Содержимое всех разрядов ячейки $\tilde{a}$ суммируется. Результат записывается в регистр R и ячейку $\tilde{b}+1$	268

Пульт управления

Пульт управления имеет две панели: визуальную и управляющую.

На панели управления имеются следующие тумблеры, клавиши и кнопки:

Тумблерный регистр «Переход по ключу», имеющий 7 тумблеров (ключей с номерами 0001, 0002, 0004, 0010, 0020, 0040, 0100).

Тумблерный регистр «Останов по адресу», имеющий 13 тумблеров, один из которых предназначен для выбора блока ОЗУ, 12 других — для набора номера ячейки, и три клавиши «Команда», «Запись» и «Чтение». Нажатие клавиши «Команда» включает режим, при котором машина останавливается, если номер команды на регистре *С* совпадает с номером на тумблерном регистре «Останов по адресу». Нажатие клавиши «Запись» включает режим, при котором машина останавливается, если номер ячейки, в которую производится запись результата операции совпадает с номером ячейки, набранным на тумблерном регистре «Останов по адресу». Нажатие клавиши «Чтение» включает режим, при котором машина останавливается, если номер ячейки, из которой при выполнении операции производится перенос кода в сумматор, совпадает с номером ячейки набранным на тумблерном регистре «Останов по адресу».

Клавишный регистр «Набор кода», имеющий 37 клавиш с надписями «зн», 1, 2, ... , 36;

Тумблер «Команда с пульта». При включении этого тумблера выполняется команда, набранная на клавишном регистре «Набор команды».

Клавишный регистр «Набор адреса», имеющий 12 клавиш с номерами 1, 2, ... , 12, и тумблер, предназначенный для выбора блока ОЗУ.

Кнопка «Занесение в сумматор». При нажатии этой кнопки код, набранный на клавишном регистре «Набор кода», переносится в сумматор.

Кнопка «СчАК» (счетчика адреса команды). При нажатии на эту кнопку адрес ОЗУ, набранный на клавишном регистре «Набор адреса», переносится в регистр *С*.

Кнопки «Гашение общее», «Гашение сумматора», «Гашение СчАК». При нажатии на одну из этих кнопок происходит

соответственно «сброс» содержимого всех регистров машины, сумматора, регистра *C*.

Клавиши «Ввод с перфоленты включен», «Останов при обращении к вводу с перфоленты».

Клавиша «Печать включена».

Кнопки «Цифровой ввод», «Реверс», «Текстовой ввод».

Кнопки «Пуск машины», «Останов машины», «Включение машины», «Выключение машины». Последние две кнопки предназначены для включения машины в сеть и ее выключения из сети.

Кнопки «Включен ввод с перфокарт», «Останов при обращении к вводу с перфокарт», клавиши «Включено АЦПУ», «Блокировка АЦПУ». Эти кнопки и клавиши в машине *Минск-2* не задействованы.

Кнопки режимов работы машины «Автоматический режим», «Циклический режим», «Тактовый режим» и «Шаговый режим».

На визуальной панели находятся ряды неоновых лампочек *r*, *R* и *s*, каждый из которых состоит из 37 лампочек, ряд регистра *C* (СчАК), состоящий из 13 лампочек. Имеются также лампочки, указывающие на причины останова машины (обращение к магнитной ленте по адресу, сбой при считывании с магнитной ленты или при записи на магнитную ленту, переполнение и др.); лампочки, указывающие на устройства, которые находятся в работе (чтение с магнитной ленты, вывод на перфоратор, вывод на печать, ввод с перфоленты, ввод с перфокарт и др.).

Подготовка данных. Ввод и вывод

Подлежащий вводу в машину материал (программа, числа, вспомогательные константы и др.) программист записывает на специальный бланк. Образец бланков приведен на рис. 13.3. На этом бланке в графе «Адрес ячейки» для первого блока ОЗУ адреса записываются четырьмя восьмеричными цифрами, а для второго блока ОЗУ — пятью цифрами, из которых первая всегда единица. В графе «Признак» пишется информация о том, какого типа слово находится в ячейке (десятичное число с фиксированной запятой, десятичное число с плавающей запятой, восьмеричное число, слово в буквенном алфавите). В графе «Индекс» при написании команды записываются номер блока ОЗУ и индекс-ячейка в виде двух восьмеричных цифр.

Далее материал с бланков переносится на пятипозиционную перфоленту с помощью клавишного устройства УПД, имеющего клавиши цифр 0, 1, ..., 9, клавиши специальных признаков: «Группа зоны», «Запись», «Передача», «Пробел»,

Адрес ячейки	Признак	Код операции		Индекс	Первый адрес	Второй адрес	Примечание
		±	—				
0131	8	—	32	36	4274	1603	Команда
4276	10ф	—	29	47	5128	6	Десятичное число с фиксированной запятой: —0,294751286
10426	10п	+	81	39	357	—04	Десятичное число с плавающей запятой: $+0{,}8139357 \cdot 10^{-04}$
17624		+	12	HI	OC		Слово в буквенно-цифровом алфавите

Рис. 13.3. Образец записи чисел, команд и слов в буквенно-цифровом алфавите на бланках.

«Запятая», «Пробел десятичный», знаки «Признаки» восьмеричных и десятеричных чисел («+ восьмеричное», «— восьмеричное», «+ десятичное» и «— десятичное»). На рис. 13.4 приведены образцы кодирования на перфоленте знаков клавиатуры устройства УПД.

Слова в буквенном алфавите переносятся с бланков на перфоленту с помощью телетайпа (СТА-2М), имеющего клавиатуру «второго телеграфного международного кода». На рис. 13.5 приведены образцы кодирования на перфоленте знаков клавиатуры второго телеграфного международного кода. Параллельно с перфорированием на телетайпе СТА-2М или на аппаратуре УПД осуществляется печать перфорируемого материала для контроля правильности набора кодов на перфорируемых устройствах.

Замечание. Перфорация должна начинаться с пробивки на перфоленте комбинации «Граница ввода» и кончаться этой же комбинацией.

Ввод информации в машину может быть трех видов: ввод по адресу, групповой ввод по адресу и ввод по команде.

Для ввода по адресу на перфоленте вместе с кодом (восьмеричным или десятичным числом) перфорируется и номер ячейки ОЗУ, в которую следует поместить вводимый код, причем порядок перфорации в этом случае следующий: в первую очередь перфорируется служебная комбинация «Передача», затем номер ячейки ОЗУ, служебная комбинация «Запись», восьмеричное или десятичное число.

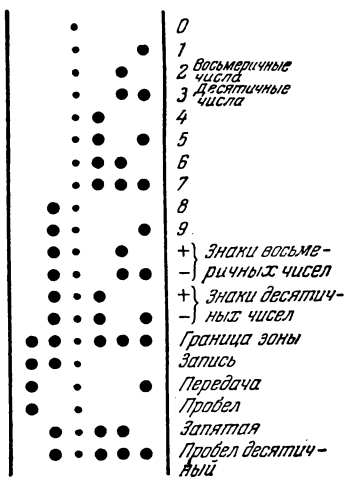


Рис. 13.4. Образец кодирования цифр и знаков на перфоленте с помощью УПД.

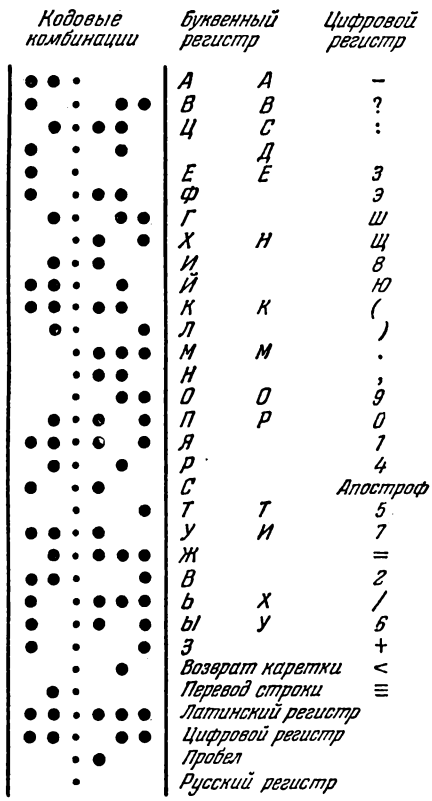


Рис. 13.5. Образец кодирования знаков на перфоленте с помощью телетайпа СТА-2М.

Для группового ввода по адресу на перфоленте пробивается служебная комбинация «Передача», затем адрес ячейки ОЗУ, в которую следует поместить первое число (восьмеричное или десятичное), затем служебную комбинацию «Запись», второе число (восьмеричное или десятичное), служебную ком-

бинацию «Запись», третье число (восьмеричное или десятичное) и т. д. В этом случае ввод в машину производится следующим образом: первое число записывается в ячейку, номер которой указан на перфоленте; второе — в ячейку с номером на единицу больше и т. д.

Для ввода по команде перфорация осуществляется так же, как и для группового ввода по адресу, с той лишь разницей, что адрес для первого числа на перфоленте не пробивается. Ввод по команде см. в табл. 13.1 в пояснении к команде с кодом операции — 50.

Слова в буквенном алфавите перфорируются на перфоленте сплошным массивом (как текст) без каких-либо служебных комбинаций, исключая комбинации «русский», «цифровой» и «латинский», которые являются управляющими в работе СТА-2М, определяющие выбор алфавита русского, цифрового или латинского и без адресов ячеек ОЗУ. Ввод с перфоленты в буквенном алфавите осуществляется по команде с кодом операции — 52.

ГЛАВА 14

ПРОГРАММНО-УПРАВЛЯЕМАЯ МАШИНА *РАЗДАН-2*

Основные характеристики машины *Раздан-2*

Тип машины	вычислительная
Класс машины	средняя
Среднее число основных операций в секунду	5000
Рабочая система счисления	двоичная
Форма представления чисел	с плавающей запятой
Длина кода числа	36 двоичных разрядов (мантисса со знаком — 30 разрядов, порядок со знаком — 6 разрядов)
Диапазон представления чисел	$2^{-60} \leqslant x \leqslant (1 - 2^{-29}) \cdot 2^{31}$ а для нормализованных $2^{-32} \leqslant x \leqslant (1 - 2^{-29}) \cdot 2^{31}$
Точность вычислений	$7 \div 8$ десятичных значащих цифр
Порядок выполнения команд	заданный (естественный)
Адресность машины	двухадресная
Виды запоминающих устройств	оперативное запоминающее устройство выполнено на магнитных сердечниках, емкость 2048 36-разрядных ячеек, время обращения — 20 *мксек*; магнитная лента емкостью 120 000 36-разрядных двоичных кодов, скорость последовательной записи или считывания 2000 кодов в секунду, скорость движения ленты 2 метра в секунду, лента движется в двух направлениях, максимальное время подвода требуемой зоны 60 секунд.

Виды ввода и вывода	ввод с перфоленты со скоростью 6÷7 36-разрядных кодов в секунду; вывод на перфоленту со скоростью 15 36-разрядных кодов в секунду и на печатающее устройство со скоростью 10÷20 36-разрядных двоичных кодов в секунду.
Количество полупроводниковых триодов	около 5000
Количество полупроводниковых диодов	около 30 000
Потребляемая мощность	2,5 *ква*
Занимаемая площадь	10÷20 $м^2$

Программно-управляемая машина *Раздан-2* является вычислительной машиной широкого назначения (универсальной). Допускается произвольный режим эксплуатации.

Краткие сведения об основных устройствах машины

Арифметическое устройство (АУ). Основным блоком АУ является сумматор *s*, работающий в дополнительном коде. Сумматор имеет 44 разряда, из них 36 разрядов служат для помещения мантиссы, порядка и знака числа при выполнении арифметических операций, 6 разрядов служат для получения произведения с требуемой точностью, 2 разряда — для хранения знака числа и знака порядка в модифицированном дополнительном коде. В сумматоре при выполнении соответствующих операций помещаются одно из слагаемых, уменьшаемое, делимое, а также накапливается произведение при умножении. Сумматор имеет цепи сдвига влево.

В состав АУ входят также два регистра. Регистр I (*RI*) состоит из 44 разрядов. В этом регистре, так же как и в сумматоре, 6 разрядов служат для получения мантиссы произведения с требуемой точностью, 2 разряда — для помещения знаков. *RI* имеет цепи сдвига вправо, служит для помещения одного из слагаемых, вычитаемого, делителя, множимого при соответствующих операциях, а также используется для обмена кодами между магнитной лентой и оперативным запоминающим устройством.

Регистр II (*RII*) состоит из 39 разрядов, из которых 36-разрядов служат для помещения кода числа при выполнении

арифметических операций, 2 разряда — для хранения знаков числа и порядка, 1 разряд используется для округления при выполнении операции деления. При умножении в *RII* помещается множитель, при делении — частное. Передача кодов с перфоленты в ОЗУ и обратно производится через *RII*. Этот регистр имеет цепи сдвига влево.

В состав АУ входит также шестиразрядный счетчик, который подсчитывает число элементарных сдвигов и сложений (вычитаний) при выполнении операций сдвига, сложения, вычитания, умножения и деления. По результату каждой из операций, возникающему в сумматоре, вырабатываются сигналы ω; γ; ν и φ. Сигналы ω; γ и ν предназначены для управления условным переходом. При этом $\omega = 1$, когда знаковый разряд сумматора содержит единицу ($«s_{30}» = 1$); $\gamma = 1$, когда разряд сумматора, помещающий знак порядка, содержит единицу ($«s_{36}» = 1$); $\nu = 1$, когда во всех разрядах сумматора содержатся нули ($«s_i» = 0$; для $1 \leqslant i \leqslant 36$). Сигнал $\varphi = 1$ вырабатывается, когда порядок результата превышает 2^{31}.

Оперативное запоминающее устройство (ОЗУ) выполнено на ферритовых сердечниках. Время обращения 20 *мксек*.

Емкость ОЗУ — 2048 36-разрядных ячеек. Все ячейки перенумерованы от 0000 до 3777 в восьмеричной системе счисления. Первые восемь ячеек ОЗУ с номерами 0000 ÷ 0007 могут быть использованы для осуществления автоматической переадресации команд перед их исполнением. По требованию заказчика обеспечивается выдача кода, содержащего нули во всех 36 разрядах из ячейки с номером 0000.

Устройство управления (УУ). Основными блоками УУ являются регистр команд R и счетчик команд C.

Регистр команд имеет 33 разряда и служит для приема кода операции и адресов исполняемой команды из ОЗУ и хранения их в процессе исполнения.

Счетчик команд имеет 11 разрядов. Его содержимое задает номер ячейки, из которой производится пересылка команды на регистр команд для исполнения.

В состав УУ входят также схемы, обеспечивающие: расшифровку кода выполняемой операции, задание стандартного такта (цикла) работы машины, передачу кода исполняемой команды в сумматор для ее изменения, передачу содержимого некоторой из первых семи ячеек ОЗУ в сумматор для авто-

матической переадресации команд, удлинение стандартного такта при выполнении некоторых команд и т. д.

Работа машины слагается из выполнения элементарных тактов, в каждом из которых последовательно производятся следующие действия:

1) считывание команды из ячейки ОЗУ, номер которой указан на счетчике команд, передача кода этой команды на сумматор, считывание содержимого ячейки ОЗУ, содержащей код для автоматической переадресации команды, суммирование адресов команды с кодом для переадресации и запись полученной команды на регистр команд, увеличение содержимого счетчика команд на единицу, восстановление первоначального содержимого сумматора; при отсутствии признаков переадресации выполняется только считывание команды из ОЗУ, увеличение содержимого счетчика команд и восстановление содержимого сумматора;

2) прием первого и второго чисел из оперативной памяти и выполнение операции в АУ, изменение содержимого счетчика команд путем записи на него нового адреса; в ряде случаев новый адрес определяется путем суммирования, для чего используется сумматор АУ;

3) запись результата в указанную ячейку ОЗУ; при некоторых модификациях операций запись в ОЗУ не производится.

При выполнении операций, требующих работы местного управления АУ (сложение, вычитание, умножение, деление, сдвиг) или осуществляющих выдачу на печать (перфоратор) и обмен между ОЗУ и накопителем на магнитной ленте, перед записью результата в ОЗУ происходит задержка элементарного такта. Во время задержки управляющие сигналы поступают либо из АУ, либо от печатающего (перфорирующего) устройства, либо от магнитной ленты.

Запоминающее устройство на магнитной ленте (НМЛ) состоит из одного лентопротяжного блока, на котором устанавливается магнитная лента длиной до $100 \div 120$ метров. На ленте максимальной длины может храниться до 120 000 36-разрядных двоичных кодов. Скорость движения ленты 2 метра в секунду. Лента движется в двух направлениях.

Магнитная лента размечается на зоны. Максимальное число зон — 511. Они имеют номера от 0001 до 0777 в восьмеричной системе счисления. Номера зон записываются на ленту перед содержимым зоны. Каждая зона может содержать не более 2048 36-разрядных кодов. Если на ленте 511 зон, то обеспечивается средняя длина зоны не менее чем на 256 кодов. Считывание и запись производится зонами. При передаче кодов в сумматоре образуется контрольная сумма переданных кодов, используемая для проверки правильности передачи.

Представление чисел и команд

Схема распределения разрядов ячейки ОЗУ с номером x при хранении в ней 36-разрядного двоичного числа представлена на рис. 14.1, *а*.

Числа представляются с плавающей запятой. Мантисса и порядок представляются в дополнительном коде. Машинный нуль представляется в виде кода, содержащего нули во всех разрядах.

Схема распределения разрядов ячейки ОЗУ с номером y при хранении в ней двоично-десятичного числа представлена на рис. 14.1, *б*.

Десятичное число представляется с плавающей запятой. 1-й и 35-й разряды не используются. Порядок и мантисса хранятся в прямом коде.

Схема распределения разрядов ячейки ОЗУ при хранении в ней команды представлена на рис. 14.1, *в*. Команда машины *Раздан-2* имеет следующую структуру:

$$r\ M\ n\ \theta'\ \tau_a\ a\ \tau_b\ b,$$

где r — адрес ячейки ОЗУ, используемой как регистр переадресации, M — код модификации операции θ', n — признак, значение которого указывает наличие блокировки округления при арифметических операциях и модификацию операции в командах логических операций и операций управления; θ' — код операции; τ_a — признак переадресации первого адреса на содержимое ячейки ОЗУ, используемой в качестве регистра переадресации; a — первый адрес; τ_b — то же, что и τ_a, но для второго адреса, b — второй адрес команды. Для удобства описания в дальнейшем разряды, обозначаемые символом n и θ', объединяют и обозначают одной буквой θ. В ряде случаев попарно объединяют символы $\tau_a a$ и $\tau_b b$ и обозначают соответственно буквами A и B.

Схема распределения разрядов ячейки ОЗУ с номером z при использовании ее в качестве регистра переадресации показана на рис. 14.1, *г*. На этом рисунке z_n — число повторений групповой операции, z_i — код, служащий для формирования исполнительного адреса, z_Δ — шаг изменения z_i. Такое же распределение разрядов применяется и для описания некоторых операций над содержимым произвольных (0000 $\leqslant$

$\leqslant z \leqslant 3777$), а не только первых восьми ($0000 \leqslant z \leqslant 0007$) ячеек ОЗУ.

На приведенном рисунке скобками ⌒ указаны объединения двоичных разрядов в восьмеричные и десятичные числа,

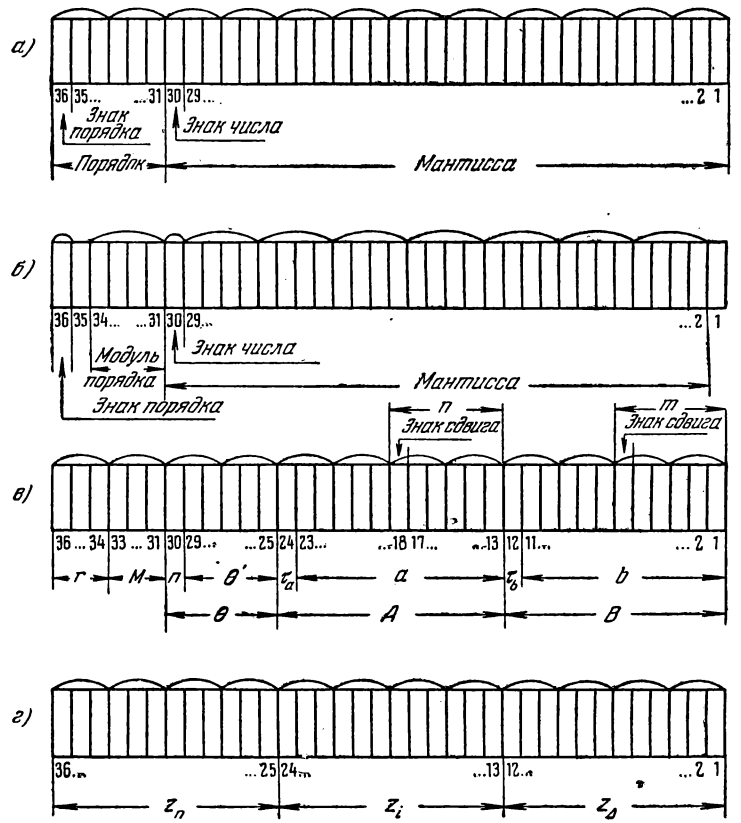


Рис. 14.1. Распределение разрядов ячейки ОЗУ: *а*) для хранения двоичного числа; *б*) для хранения двоично-десятичного числа; *в*) для хранения команды; *г*) для хранения кода переадресации.

принятые для записи команд и чисел. В знаковых разрядах 0 обозначает плюс, 1 — минус.

Система операций и команд

Во всех командах машины *Раздан-2* признаки τ_a и τ_b служат для формирования исполнительных адресов $\tilde{a}$ и $\tilde{b}$ по правилам:

$$\tilde{a} = a \boxplus_{11} \tau_a \cdot «r_i»,$$

$$\tilde{b} = b \boxplus_{11} \tau_b \cdot «r_i»,$$

где $«r_i»$ — содержимое $13 \div 24$ разрядов ячейки ОЗУ с номером r, указанным в $34 \div 36$ разрядах команды.

Операции машины *Раздан-2* имеют модификации. Значение M указывает вид модификации. Большинство операций имеют по восемь стандартных модификаций, показанных в таблице 14.1. В этой таблице θ обозначает символ конкретной операции ($+$, $\times$ и т. д.), выполняемой по команде с кодом операции θ. Эта команда располагается в ячейке k. Символы a_{I}; a_{II} обозначают 1-й и 2-й адреса команды, хранящейся в ячейке с номером a.

При выполнении всех операций, кроме операций с $\theta = 30$, $\theta = 31$, $\theta = 37$, вырабатываются сигналы

$$\omega = 1 \text{ при } s < 0,$$
$$\gamma = 1 \text{ при } s_p < 0,$$
$$\nu = 1 \text{ при } s_j = 0 \text{ для } 1 \leqslant j \leqslant 36,$$

в остальных случаях сигналы принимают значение 0.

При выполнении операций $\theta = 30$, $\theta = 31$, $\theta = 37$ значения сигналов сохраняются теми же, которые выработаны при выполнении предыдущей операции.

Сигнал $\varphi = 1$ вырабатывается при $s > (1 - 2^{-29}) \cdot 2^{31}$.

Как видно из таблицы 14.1, при модификациях 2 и 3 после выполнения основной операции θ производится сложение с содержанием ячейки $\tilde{b}$. Сложение производится по правилам для чисел с плавающей запятой. При этом округление и нормализация выполняются так же, как и для основной операции θ.

Система операций и команд приведена в табл. 14.2.

В машине *Раздан-2* предусмотрена возможность выполнения абсолютной и относительной передачи управления. В первом случае на счетчик команд записывается значение

Таблица 14.1. Стандартные модификации операций

Значение M	Выполнение операции	Пояснения
0	$s := \tilde{a}\theta\tilde{b}$, $\tilde{b} := s$	Операция выполняется над содержимым ячеек с номерами $\tilde{a}$ и $\tilde{b}$, образованными при исполнении команды. Результат сохраняется в сумматоре и посылается в ячейку $\tilde{b}$
1	$s := \tilde{a}\theta\tilde{\tilde{b}}$	Отличается от предыдущей модификации тем, что нет засылки в ячейку $\tilde{b}$
2	$s := \tilde{\tilde{b}} + (s\theta\tilde{a})$, $\tilde{b} := s$	Операция θ выполняется над содержимым сумматора и ячейки $\tilde{a}$. К полученному числу прибавляется содержимое ячейки $\tilde{b}$. Результат сохраняется в сумматоре и посылается в ячейку $\tilde{b}$
3	$s := \tilde{b} + (s\theta\tilde{a})$	Отличается от предыдущей модификации тем, что нет засылки в ячейку $\tilde{b}$
4	$s := s\theta\tilde{a}$, $\tilde{b} := s$	Операция θ выполняется над содержимым сумматора и ячейки $\tilde{a}$. Результат сохраняется в сумматоре и посылается в ячейку $\tilde{b}$
5	$s := s\theta\tilde{a}$ В зависимости от значения кода операции и комбинаций значений сигналов ω; γ; ν n, либо: $C := k_{II}$,	Операция θ выполняется над содержимым сумматора и ячейки $\tilde{a}$. Результат остается в сумматоре. В зависимости от результата и вида операции θ вырабатываются значения сигналов ω; γ; ν.

Продолжение табл. 14.1

Значение M	Выполнение операции	Пояснения
5	либо $C := C + 1$	Если при соответствующем коде операции определенный сигнал принимает значение 1, то происходит передача управления к ячейке с номером $\tilde{b}$. В противном случае выполняется следующая команда
6	$s := s\theta\tilde{a}_{\text{I}}$; $\tilde{a}_{\text{II}} := s$; $C := k_{\text{II}} \underset{11}{\boxed{+}} (C + 1)$	Операция θ выполняется над содержимым сумматора и ячейки, номер которой указан в первом адресе (13 ÷ 23-й разряды) команды, стоящей в ячейке с номером $\tilde{a}$. Результат сохраняется в сумматоре и посылается в ячейку, номер которой указан во втором адресе (2 ÷ 11-й разряды) команды, стоящей в ячейке с номером $\tilde{a}$. Происходит безусловная передача управления к ячейке, номер которой является суммой значения второго адреса k_{II} и содержимого счетчика команд
7	$s := s\theta\tilde{a}$ В зависимости от значения кода операции и комбинаций значений сигналов ω; γ; ν либо: $C := k_{\text{II}}$, либо: $C := C + 1$	Отличается от модификации с $M = 5$ тем, что передача управления производится, когда один из сигналов ω; γ; ν принимает значение 0

Таблица 14.2. Система операций и команд

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
01	Сложение с нормализацией и округлением $r\ M\ 01\ \tau_a\ a\tau_b\ b$	0	$s:=\tilde{a}+\tilde{b}$, $b:=s$	124	Сложение выполняется по правилам для чисел с плавающей запятой на сумматоре, включая дополнительные шесть разрядов. Округление производится по старшему из шести дополнительных разрядов после нормализации. Если после округления произошло нарушение нормализации, то нормализация производится второй раз. Округление при этом не выполняется
		1	$s:=\tilde{a}+\tilde{b}$		
		2	$s:=\tilde{b}+(s+\tilde{a})$, $\tilde{b}:=s$		
		3	$s:=\tilde{b}+(s+\tilde{a})$		
		4	$s:=s+\tilde{a}$, $\tilde{b}:=s$		
		5	$s:=s+\tilde{a}$, $C:=k_{\text{II}}$, если $\omega=1$, $C:=C+1$ в противном случае		
		6	$s:=s+\tilde{a}_{\text{I}}$, $\tilde{a}_{\text{II}}:=s$, $C:=k_{\text{II}}\ \underset{11}{\boxed{+}}\ (C+1)$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		7	$s := s + \tilde{a}$, $C := k_{II}$, если $\omega = 0$, $C := C + 1$ в противном случае		
11	Сложение без нормализации с округлением *r M* 11 τ_a *a* τ_b *b*	Те же, что и для операции 01	То же, что и для операции $\theta = 01$	124	Отличается от операции $\theta = 01$ тем, что нормализация не производится
41	Сложение с нормализацией без округления *r M* 41 τ_a *a* τ_b *b*	То же, что и для операции $\theta = 01$	То же, что и для операции $\theta = 01$	124	Отличается от операции $\theta = 01$ тем, что округление не производится
51	Сложение без нормализации и без округления *r M* 51 τ_a *a* τ_b *b*	Те же, что и для операции $\theta = 01$	То же, что и для операции $\theta = 01$	124	Отличается от операции $\theta = 01$ тем, что округление и нормализация не производятся

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
02	Вычитание I с нормализацией и округлением $r\ M\ 02\ \tau_a\ a\ \tau_b\ b$	0	$RI := \overline{\operatorname{sign} \tilde{b}}\,|\,b\,|$, $s := \tilde{a} + RI$, $\tilde{b} := s$		Вычитаемому, выбранному из ОЗУ и записанному на RI, присваивается противоположный знак, после чего выполняется сложение с нормализацией и округлением ($\theta = 01$)
		1	$RI := \overline{\operatorname{sign} \tilde{b}}\,|\,\tilde{b}\,|$, $s := \tilde{a} + RI$		
		2	$RI := \overline{\operatorname{sign} \tilde{a}}\,|\,\tilde{a}\,|$, $s := \tilde{b} + (s + RI)$, $\tilde{b} := s$		
		3	$RI := \overline{\operatorname{sign} \tilde{a}}\,|\,\tilde{a}\,|$, $s := \tilde{b} + (s + RI)$		
		4	$RI := \overline{\operatorname{sign} \tilde{a}}\,|\,\tilde{a}\,|$, $s := s + RI$, $\tilde{b} := s$		
		5	$RI := \overline{\operatorname{sign} \tilde{a}_I}\,|\,\tilde{a}_I\,|$, $s := s + RI$, $C := k_{II}$, если $\omega = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		6	$RI := \overline{\operatorname{sign} \tilde{a}_I} \mid \tilde{a}_I \mid$, $s := s + RI$, $\tilde{a}_{II} := s$, $C := k_{II} \overline{\underset{11}{+}} (C + 1)$		
		7	$RI := \overline{\operatorname{sign} \tilde{a}} \mid \tilde{a} \mid$, $s := s + RI$, $C := k_{II}$, если $\omega = 0$, $C := C + 1$ в противном случае		
12	Вычитание I без нормализации с округлением *r M* 12 τ_a *a* τ_b *b*	Те же, что и для операции $\theta = 02$	То же, что и для операции $\theta = 02$	124	Отличается от операции с $\theta = 02$ тем, что после изменения знака вычитаемого производится сложение без нормализации с округлением ($\theta = 11$)

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
42	Вычитание I с нормализацией без округления $r\ M\ 42\ \tau_a\ a\ \tau_b\ b$	Те же, что и для операции $\theta = 02$	То же, что и для операции $\theta = 02$	124	Отличается от операции с $\theta = 02$ тем, что после изменения знака вычитаемого производится сложение с нормализацией без округления ($\theta = 42$)
52	Вычитание I без нормализации и без округления $r\ M\ 52\ \tau_a\ a\ \tau_b\ b$	Те же, что и для операции $\theta = 01$	То же, что и для операции $\theta = 02$	124	Отличается от операции с $\theta = 02$ тем, что после изменения знака вычитаемого производится сложение без нормализации и без округления ($\theta = 52$)
03	Вычитание II с нормализацией и округлением $r\ M\ 03\ \tau_a\ a\ \tau_b\ b$	0	$R\mathrm{I} := \overline{\operatorname{sign} \tilde{a}}\ \lvert \tilde{a} \rvert,$ $s := \tilde{b} + R\mathrm{I},$ $\tilde{b} := s$	124	Отличается от операции вычитания I с нормализацией и округлением ($\theta = 02$) тем, что уменьшаемое и вычитаемое меняются местами
		1	$R\mathrm{I} := \overline{\operatorname{sign} \tilde{a}}\ \lvert \tilde{a} \rvert,$ $s := \tilde{b} + R\mathrm{I}$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		2	$RI := \overline{\text{sign } s}\,\lvert s \rvert$, $s := \tilde{b} + (\tilde{a} + RI)$, $\tilde{b} := s$		
		3	$RI := \overline{\text{sign } s}\,\lvert s \rvert$, $s := \tilde{b} + (\tilde{a} + RI)$		
		4	$RI := \overline{\text{sign } s}\,\lvert s \rvert$, $s := \tilde{a} + RI$, $\tilde{b} := s$		
		5	$RI := \overline{\text{sign } s}\,\lvert s \rvert$, $s := \tilde{a}_I + RI$, $C := k_{II}$, если $\omega = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		6	$RI := \overline{\operatorname{sign} s} \mid s \mid$, $s := \tilde{a}_I + RI$, $\tilde{a}_{II} := s$, $C := k_{II} \boxplus\limits_{II} (C+1)$		
		7	$RI := \overline{\operatorname{sign} s} \mid s \mid$, $s := \tilde{a} + RI$, $C := k_{II}$, если $\omega = 0$, $C := C + 1$ в противном случае		
13	Вычитание II без нормализации с округлением *r M* 13 τ_a *a* τ_b *b*	Те же, что и для операции с $\theta = 03$	То же, что и для операции с $\theta = 03$	124	Отличается от операции вычитание I без нормализации с округлением (θ=12) тем, что уменьшаемое и вычитаемое меняются местами

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
43	Вычитание II с нормализацией без округления *r M* 43 τ_a *a* τ_b *b*	Те же, что и для операции с $\theta = 03$	То же, что и для операции с $\theta = 03$	124	Отличается от операции вычитание I с нормализацией без округления (θ=42) тем, что уменьшаемое и вычитаемое меняются местами
53	Вычитание II без нормализации и без округления *r M* 53 τ_a *a* τ_b *b*	Те же, что и для операции $\theta = 03$	То же, что и для операции $\theta = 03$	124	Отличается от операции вычитание I без нормализации и без округления (θ=52) тем, что уменьшаемое и вычитаемое меняются местами
04	Вычитание модулей с нормализацией и округлением *r M* 04 τ_a *a* τ_b *b*	0	$RI := -\lvert\tilde{b}\rvert$, $s := \lvert\tilde{a}\rvert$, $s := s + RI$, $\tilde{b} := s$	124	Вычитаемому, выбранному из ОЗУ и записанному на *RI*, присваивается знак минус, уменьшаемому, которое записано на сумматоре, присваивается знак плюс. После этого выполняется операция сложения с нормализацией и округлением ($\theta = 01$)
		1	$RI := -\lvert\tilde{b}\rvert$, $s := \lvert\tilde{a}\rvert$, $s := s + RI$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		2	$RI := -\lvert \tilde{a} \rvert$, $s := \lvert s \rvert$, $s := \tilde{b} + (s + RI)$, $\tilde{b} := s$		
		3	$RI := -\lvert \tilde{a} \rvert$, $s := \lvert s \rvert$, $s := (s + RI)$		
		4	$RI := -\lvert \tilde{a} \rvert$, $s := \lvert s \rvert$, $s := s + RI$, $\tilde{b} := s$		
		5	$RI := -\lvert \tilde{a} \rvert$, $s := \lvert s \rvert$, $s := s + RI$, $C := k_{II}$, если $\omega = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		6	$RI := -\lvert \tilde{a}_I \rvert$, $s := \lvert s \rvert$, $a_{II} := s + RI$, $C := k_{II} \boxplus (C+1)$ 11		
		7	$RI := -\lvert \tilde{a} \rvert$, $s := \lvert s \rvert$, $s := s + RI$, $C := k_{II}$, если $\omega = 0$, $C := C + 1$ в противном случае		
14	Вычитание модулей без нормализации с округлением *r M* 14 τ_a *a* τ_b *b*	Те же, что и для операции $\theta = 04$	То же, что и для операции $\theta = 04$	124	Отличается от операции вычитания модулей с округлением ($\theta = 04$) и нормализацией тем, что после изменения знаков выполняется операция сложения без нормализации с округлением ($\theta = 11$)

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
44	Вычитание модулей с нормализацией без округления *r M* 44 τ_a *a* τ_b *b*	Те же, что и для операции $\theta = 04$	То же, что и для операции $\theta = 04$	124	Отличается от операции вычитания модулей с округлением и нормализацией тем, что после изменения знаков выполняется операция сложения с нормализацией без округления ($\theta = 41$)
54	Вычитание модулей без нормализации и без округления *r M* 54 τ_a *a* τ_b *b*	Те же, что и для операции $\theta = 04$	То же, что и для операции $\theta = 04$	124	Отличается от операции вычитания модулей с округлением и нормализацией ($\theta = 04$) тем, что после изменения знаков выполняется операция сложения без нормализации и без округления ($\theta = 51$)
05	Умножение с нормализацией и округлением *r M* 05 τ_a *a* τ_b *b*	0 .	$s := \tilde{a} \cdot \tilde{b}$, $\tilde{b} := s$	457	При умножении мантисса определяется с 35 разрядами. Из них шесть дополнительных обеспечивают правильное вычисление 29-го и 30-го разрядов мантиссы произведения. Округление производится после нормализации путем прибавления
		1	$s := \tilde{a} \cdot \tilde{b}$		
		2	$s := \tilde{b} + (s \cdot \tilde{a})$, $\tilde{b} := s$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		3	$s := \tilde{b} + (s \cdot \tilde{a})$		содержимого 30-го разряда к 29-му разряду. Если исходные числа не нормализованы, то произведение может быть получено с недостатком
		4	$s := s \cdot \tilde{a}$, $\tilde{b} := s$		
		5	$s := s \cdot \tilde{a}$, $C := k_{II}$, если $\gamma = 1$ или $\omega = 1$, $C := C + 1$ в противном случае		
		6	$s := s \cdot \tilde{a}_I$, $\tilde{a}_{II} := s$, $C := k_{II} \boxplus (C + 1)$ (11)		
		7	$s := s \cdot \tilde{a}$, $C := k_{II}$, если $\omega = 0$ или $\gamma = 0$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
15	Умножение без нормализации с округлением $r\ M\ 15\ \tau_a\ a\ \tau_b\ b$	Те же, что и для операции $\theta = 05$	То же, что и для операции $\theta = 05$	457	Отличается от операции умножение с нормализацией и округлением ($\theta = 05$) тем, что нормализация не производится
45	Умножение с нормализацией без округления $r\ M\ 45\ \tau_a\ a\ \tau_b\ b$	Те же, что и для операции $\theta = 05$	То же, что и для операции $\theta = 05$	457	Отличается от операции умножение с нормализацией и округлением ($\theta = 05$) тем, что округление не производится
55	Умножение без нормализации и без округления $r\ M\ 55\ \tau_a\ a\ \tau_b\ b$	Те же, что и для операции $\theta = 05$	То же, что и для операции $\theta = 05$	457	Отличается от операции умножение с нормализацией и округлением ($\theta = 05$) тем, что ни нормализация, ни округление не производятся
06	Деление с нормализацией и округлением $r\ M\ 06\ \tau_a\ a\ \tau_b\ b$	0	$s := \tilde{a} : \tilde{b}$, $\tilde{b} := s$	800	Мантисса частного образуется как результат деления мантиссы делимого на мантиссу делителя. Порядок образуется путем вычитания из порядка делимого порядка делителя. Частное
		1	$s := \tilde{a} : \tilde{b}$		
		2	$s := \tilde{b} + (s : \tilde{a})$, $\tilde{b} := s$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		3	$s := \tilde{b} + (s : \tilde{a})$		нормализуется и округляется по 30-му разряду мантиссы частного, который специально определяется для этой цели
		4	$s := \tilde{s} : \tilde{a}$, $\tilde{b} := s$		
		5	$s := s : a_{\text{I}}$, $C := k_{\text{II}}$, если $\omega = 1$ или $\gamma = 1$, $C := C + 1$ в противном случае		
		6	$s := s : \tilde{a}_{\text{I}}$, $\tilde{a}_{\text{II}} := s$, $C := (C + 1) \underset{11}{\boxplus} k_{\text{II}}$		
		7	$s := s : \tilde{a}$, $C := k_{\text{II}}$, если $\omega = 0$ или $\gamma = 0$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
16	Деление без нормализации с округлением *r M* 16 τ_a *a* τ_b *b*	Те же, что и для операции $\theta = 06$	То же, что и для операции $\theta = 06$	800	Отличается от операции деления с нормализацией и округлением ($\theta = 06$) тем, что нормализация не производится
46	Деление с нормализацией без округления *r M* 46 τ_a *a* τ_b *b*	Те же, что и для операции $\theta = 06$	То же, что и для операции $\theta = 06$	800	Отличается от операции деления с нормализацией и округлением ($\theta = 06$) тем, что округление не производится
56	Деление без нормализации и без округления *r M* 56 τ_a *a* τ_b *b*	Те же, что и для операции $\theta = 06$	То же, что и для операции $\theta = 06$	800	Отличается от операции деления с нормализацией и округлением ($\theta = 06$) тем, что нормализация и округление не производятся

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
07	Сложение команд *r M* 07 τ_a *a* τ_b *b*	0	$s_\Delta := \tilde{a}_\Delta \boxplus\limits_{12} \tilde{b}_\Delta$, $s_i := \tilde{a}_i \boxplus\limits_{12} \tilde{b}_i$, $s_n := \tilde{a}_n \boxplus\limits_{12} b_n$, $\tilde{b} := s$	84	Содержимое сумматора образуется путем сложения исходных кодов, каждый из которых рассматривается как состоящий из трех целых двоичных чисел, расположенных в разрядах 1÷12; 13÷24 и 25÷36. Эти числа обозначаются через z_Δ, z_i и z_n соответственно. При сложении переносы из 12-го в 13-й, из 24-го в 25-й и из 36-го разряда блокируются
		1	$s_\Delta := \tilde{a}_\Delta \boxplus\limits_{12} \tilde{\tilde{b}}_\Delta$, $s_i := \tilde{a}_i \boxplus\limits_{12} \tilde{\tilde{b}}_i$, $s_n := \tilde{a}_n \boxplus\limits_{12} \tilde{\tilde{b}}_n$		
		2	$s_\Delta := s_\Delta \boxplus\limits_{12} \tilde{a}_\Delta$, $s_i := s_i \boxplus\limits_{12} \tilde{a}_i$, $s_n := s_n \boxplus\limits_{12} \tilde{a}_n$, $C := k_{II}$ при «s_{12}» $= 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		3	$s_\Delta := s_\Delta \boxplus\limits_{12} \tilde{a}_\Delta$, $s_i := s_i \boxplus\limits_{12} \tilde{a}_i$, $s_n := s_n \boxplus\limits_{12} \tilde{a}_n$, $C := k_{11}$ при «s_{24}» $= 1$, $C := C + 1$ в противном случае		
		4	$s_\Delta := s_\Delta \boxplus\limits_{12} \tilde{a}_\Delta$, $s_i := s_i \boxplus\limits_{12} \tilde{a}_i$, $s_n := s_n \boxplus\limits_{12} \tilde{a}_n$, $b := s$		
		5	$s_\Delta := s_\Delta \boxplus\limits_{12} \tilde{a}_\Delta$, $s_i := s_i \boxplus\limits_{12} \tilde{a}_i$, $s_n := s_n \boxplus\limits_{12} \tilde{a}_n$, $C := k_{11}$ при $\gamma = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		6	$s_\Delta := s_\Delta \underset{12}{\boxplus} \tilde{a}_{1\Delta}$, $s_i := s_i \underset{12}{\boxplus} \tilde{a}_{1i}$, $s_n := s_n \underset{12}{\boxplus} \tilde{a}_{1n}$, $\tilde{a}_{1\mathrm{I}} := s$, $C := k_{\mathrm{II}} \underset{11}{\boxplus} (C+1)$		
		7	$s_\Delta := s_\Delta \underset{12}{\boxplus} \tilde{a}_\Delta$, $s_i := s_i \underset{12}{\boxplus} \tilde{a}_i$, $s_n := s_n \underset{12}{\boxplus} \tilde{a}_n$, $C := k_{\mathrm{II}}$ при $\gamma = 0$, $C := C+1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
10	Вычитание команд *r M* 10 τ_a *a* τ_b *b*	0	$s_\Delta := \tilde{a}_\Delta \boxplus_{12} [\tilde{b}_\Delta]_g$, $s_i := \tilde{a}_i \boxplus_{12} [\tilde{b}_i]_g$, $s_n := \tilde{a}_n \boxplus_{12} [\tilde{b}_n]_g$, $b := s$	84	Отличается от операции сложения команд ($\theta = 07$) тем, что перед сложением один из исходных кодов представляется в дополнительном коде. Такие коды обозначаются символом $[x]_g$, где x — номер ячейки, хранящей код, принимающий участие в операции
		1	$s_\Delta := s_\Delta \boxplus_{12} [\tilde{a}_\Delta]_g$, $s_i := s_i \boxplus_{12} [\tilde{a}_i]_g$, $s_n := s_n \boxplus_{12} [\tilde{a}_n]_g$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		2	$s_\Delta := s_\Delta \boxplus_{12} [\tilde{a}_\Delta]_g$, $s_i := s_i \boxplus_{12} [\tilde{a}_i]_g$, $s_n := s_n \boxplus_{12} [\tilde{a}_n]_g$; $C := k_{II}$ при «s_{12}»=1, $C := C + 1$ в противном случае		
		3	$s_\Delta := s_\Delta \boxplus_{12} [\tilde{a}_\Delta]_g$, $s_i := s_i \boxplus_{12} [\tilde{a}_i]_g$, $s_n := s_n \boxplus_{12} [\tilde{a}_n]_g$; $C := k_{II}$ при «s_{24}»=1, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		4	$s_\Delta := s_\Delta \boxplus\limits_{12} [\tilde{a}_\Delta]_g$, $s_i := s_i \boxplus\limits_{12} [\tilde{a}_i]_g$, $s_n := s_n \boxplus\limits_{12} [\tilde{a}_n]_g$, $b := s$		
		5	$s_\Delta := s_\Delta \boxplus\limits_{12} [\tilde{a}_\Delta]_g$, $s_i := s_i \boxplus\limits_{12} [\tilde{a}_i]_g$, $s_n := s_n \boxplus\limits_{12} [\tilde{a}_n]_g$; $C := k_{II}$ при $\gamma = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		6	$s_\Delta := s_\Delta \boxplus_{12} [\tilde{a}_{1\Delta}]_g$, $s_i := s_i \boxplus_{12} [\tilde{a}_{1i}]_g$, $s_n := s_n \boxplus_{12} [\tilde{a}_{1n}]_g$. $C := k_{11} \boxplus_{11} (C+1)$		
		7	$s_\Delta := s_\Delta \boxplus_{12} [\tilde{a}_\Delta]_g$, $s_i := s_i \boxplus_{12} [\tilde{a}_i]_g$, $s_n := s_n \boxplus_{12} [\tilde{a}_n]_g$. $C := k_{11}$ при $\gamma = 0$, $C := C+1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
17	Циклическое сложение *r M* 17 τ_a *a* τ_b *b*	0	$s := \tilde{a} \oplus \tilde{b}$ $\tilde{b} := s$	84	Содержимое сумматора представляет собой сумму исходных кодов с круговым переносом из 36-го разряда в 1-й
		1	$s := \tilde{a} \oplus \tilde{b}$		
		2	$s := \tilde{b} + (s \oplus \tilde{a})$, $\tilde{b} := s$		
		3	$s := \tilde{b} + (s \oplus \tilde{a})$		
		4	$s := s \oplus \tilde{a}$, $\tilde{b} := s$		
		5	$s := s \oplus \tilde{a}$, $C := k_{II}$ при $\nu = 1$, $C := C + 1$ в противном случае		
		6	$s := s \oplus \tilde{a}_I$, $\tilde{a}_{II} := s$, $C := k_{II} \boxplus_{II} (C + 1)$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		7	$s:=s\oplus\tilde{a}$, $C:=k_{II}$ при $\nu=0$, $C:=C+1$ в противном случае		
57	Циклическое сложение (считывание с пульта управления) *r M* 57 τ_a *a* τ_b *b*	0	$s:=\tilde{a}\oplus\rho$, $\tilde{b}:=s$	84	Отличается от операции $\theta=07$ тем, что один из кодов выбирается с регистра ρ пульта управления (см. стр. 489)
		1	$s:=\tilde{a}\oplus\rho$		
		2	$s:=\tilde{b}+(s\oplus\rho)$, $\tilde{b}:=s$		
		3	$s:=\tilde{b}+(s\oplus\rho)$		
		4	$s:=s\oplus\rho$, $\tilde{b}:=s$		
		5	$s:=s\oplus\rho$, $C:=k_{II}$ при $\nu=1$, $C:=C+1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		6	$s := s \oplus \rho_I$, $\rho_{II} := s$, $C := k_{II} \boxplus\limits_{II} (C+1)$		
		7	$s := s \oplus \rho$, $C := k_{II}$ при $\nu = 0$, $C := C+1$ в противном случае		
20	Сравнение *r M* 20 τ_a *a* τ_b *b*	0	$s_i := \tilde{a}_i \approx \tilde{b}_i$, $\tilde{b} := s$	84	Содержимое сумматора является результатом поразрядной операции отрицание равнозначности над исходными кодами, $1 \leqslant i \leqslant 36$
		1	$s_i := \tilde{a}_i \approx \tilde{b}_i$		
		2	$s_i := s_i \approx \tilde{a}_i$, $s := s + \tilde{b}$, $\tilde{b} := s$		
		3	$s_i := s_i \approx \tilde{a}_i$ $s := s + \tilde{b}$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		4	$s_i := s_i \approx \tilde{a}_i$, $\tilde{b} := s$		
		5	$s_i := s_i \approx \tilde{a}_i$, $C := k_{11}$ при $\nu = 1$ $C := C + 1$ в противном случае		
		6	$s := s_i \approx \tilde{a}_{1i}$, $\tilde{a}_{11} := s$, $C := k_{11} \boxplus_{11} (C + 1)$		
		7	$s_i := s_i \approx \tilde{a}_i$, $C := k_{11}$ при $\nu = 0$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
60	Сравнение (считывание с пульта управления) $r\ M\ 60\ \tau_a\ a\ \tau_b\ b$	0	$s_i := \tilde{a}_i \approx \rho_i$, $\tilde{b} := s$	84	Отличается от операции $\theta = 20$ тем, что один из кодов выбирается с регистра ρ пульта управления (см. стр. 489)
		1	$s_i := \tilde{a}_i \approx \rho_i$		
		2	$s_i := s_i \approx \rho_i$, $s := s + \tilde{b}$, $\tilde{b} := s$		
		3	$s_i := s_i \approx \rho_i$, $s := s + \tilde{b}$		
		4	$s_i := s_i \approx \rho_i$, $\tilde{b} := s$		
		5	$s_i := s_i \approx \rho_i$; $C := k_{II}$ при $\nu = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		6	$s := s \approx \rho_{Ii}$, $\rho_{II} := s$, $C := k_{II} \boxplus\limits_{II} (C+1)$		
		7	$s_i := s_i \approx \rho_i$; $C := k_{II}$ при $\nu = 0$, $C := C+1$ в противном случае		
21	Сложение порядков *r M* 21 τ_a *a* τ_b *b*	0	$s_M := \tilde{a}_M$, $s_P := \tilde{a}_P + \tilde{b}_P$, $\tilde{b} := s$	84	Порядок результата равен сумме порядков исходных чисел. Мантисса результата равна мантиссе одного из исходных чисел. При модификациях 1 и 2 происходит нормализация (в тех случаях когда мантисса результата ненормализована) и округление. Однако при округлении результат не меняется, так как в дополнительных разрядах сумматора всегда содержатся нули
		1	$s_M := \tilde{a}_M$, $s_P := \tilde{a}_P + \tilde{b}_P$, содержимое сумматора нормализуется и округляется $\tilde{b} := s$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		2	$s_M := s_M$, $s_P := s_P + \tilde{a}_P$. $s := s + \tilde{b}$, содержимое сумматора нормализуется и округляется		
		3	$s_M := s_M$, $s_P := s_P + \tilde{a}_P$, $s := s + \tilde{b}$		
		4	$s_M := s_M$, $s_P := s_P + \tilde{a}_P$, $\tilde{b} := s$		
		5	$s_M := s_M$, $s_P := s_P + \tilde{a}_P$; $C := k_{II}$ при $\gamma = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		6	$s_M := s_M$, $s_P := s_P + \tilde{a}_{1P}$, $\tilde{a}_{11} := s$, $C := k_{11} \boxplus_{11} (C+1)$		
		7	$s_M := s_M$, $s_P := s_P + \tilde{a}_P$; $C := k_{11}$, если $\gamma = 0$, $C := C+1$ в противном случае		
22	Вычитание порядков *r M* 22 τ_a *a* τ_b *b*	0	$s_M := \tilde{a}_M$, $s_P := \tilde{a}_P + [\tilde{b}_P]_g$, $b := s$	84	Отличается от операции сложение порядков тем, что порядок результата равен разности порядков исходных данных
		1	$s_M := s_M$, $s_P := s_P + [\tilde{b}_P]_g$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		2	$s_M := s_M$, $s_P := s_P + [\tilde{a}_P]_g$, $s := s + \tilde{b}$, содержимое сумматора нормализуется и округляется $\tilde{b} := s$		
		3	$s_M := s_M$, $s_P := s_P + [\tilde{a}_P]_g$, содержимое сумматора нормализуется и округляется		
		4	$s_M := s_M$, $s_P := s_P + [\tilde{a}_P]_g$, $\tilde{b} := s$		
		5	$s_M := s_M$, $s_P := s_P + [\tilde{a}_P]_g$; $C := k_{II}$ при $\gamma = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		6	$s_M := s_M$, $s_P := s_P + [\tilde{a}_{1P}]_g$, $\tilde{a}_{II} := s$, $C := k_{II} \boxplus_{11} (C+1)$		
		7	$s_M := s_M$, $s_P := s_P + [\tilde{a}_P]_g$; $C := k_{II}$, если $\gamma = 0$, $C := C + 1$ в противном случае		
23	Логическое сложение *r M* 23 τ_a *a* τ_b *b*	0	$s_i := \tilde{a}_i \vee \tilde{b}_i$, $\tilde{b} := s$	84	Содержимое *i*-го разряда сумматора $1 \leqslant i \leqslant 36$ образуется путем выполнения дизъюнкции над содержимым одноименных разрядов исходных кодов
		1	$s_i := \tilde{a}_i \vee \tilde{b}_i$		
		2	$s_i := s_i \vee \tilde{a}_i$, $s := s + \tilde{b}$, $\tilde{b} := s$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
		3	$s_i := s_i \vee \tilde{a}_i$, $s := s + \tilde{b}$		
		4	$s_i := s_i \vee \tilde{a}_i$, $\tilde{b} := s$		
		5	$s_i := s_i \vee \tilde{a}_i$; $C := k_{II}$ при $\nu = 1$, $C := C + 1$ в противном случае		
		6	$s_i := s_i \vee \tilde{a}_{1i}$, $\tilde{a}_{II} := s$, $C := k_{II} \boxplus\limits_{11} (C + 1)$		
		7	$s_i := s_i \vee \tilde{a}_i$; $C := k_{II}$, если $\nu = 0$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
63	Логическое сложение (считывание с пульта управления) *r M* 63 τ_a *a* τ_b *b*	0	$s_i := \tilde{a}_i \vee \rho_i$, $\tilde{b} := s$	84	Отличается от операции $\theta = 23$ тем, что один из кодов выбирается с регистра ρ пульта управления (см. стр. 489)
		1	$s_i := \tilde{a}_i \vee \rho_i$		
		2	$s_i := s_i \vee \rho_i$, $s := s + \tilde{b}$, $\tilde{b} := s$		
		3	$s_i = s_i \vee \rho_i$, $s := s + \tilde{b}$		
		4	$s_i := s_i \vee \rho_i$, $\tilde{b} := s$		
		5	$s_i := s_i \vee \rho_i$, $C := k_{\text{II}}$ при $\nu = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
63		6	$s_i := s_i \vee \rho_{1i}$, $\rho_{II} := s$, $C := k_{II} \boxplus\limits_{11} (C+1)$		
		7	$s_i := s_i \vee \rho_i$, $C := k_{II}$, если $\nu = 0$, $C := C+1$ в противном случае		
24	Логическое умножение *r M* 24 τ_a *a* τ_b *b*	0	$s_i := \tilde{a}_i \wedge \tilde{b}_i$, $b := s$	96	Содержимое *i*-го разряда сумматора ($1 \leqslant i \leqslant 36$) образуется путем выполнения конъюкции над содержимым одноименных разрядов исходных кодов
		1	$s_i := \tilde{a}_i \wedge \tilde{b}_i$		
		2	$s_i := s_i \wedge \tilde{a}_i$, $s := s + b$, $b := s$		
		3	$s_i := s_i \wedge \tilde{a}_i$, $s := s + b$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
24		4	$s_i := s_i \wedge \tilde{a}_i$, $s := s + b$		
		5	$s_i := s_i \wedge \tilde{a}_i$, $C := k_{II}$ при $\nu = 1$, $C := C + 1$ в противном случае		
		6	$s_i := s_i \wedge \tilde{a}_{Ii}$, $b_{II} := s$, $C := k_{II} \boxplus\limits_{II} (C + 1)$		
		7	$s_i := s_i \wedge \tilde{a}_i$, $C := k_{II}$, если $\nu = 0$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
25	Сдвиг *r M* 25 τ_a *a* τ_b *b*	0	$s_i := \begin{cases} \tilde{b}_{i-\tilde{a}_I} \text{ при} \\ 1 \leqslant i - \tilde{a}_I \leqslant 36; \\ 0 \text{ при } 36 < i - \tilde{a}_I, \\ i - \tilde{a}_I \leqslant 0, \end{cases}$ $\tilde{b}_i := s_i$	от 60 до 350	Содержимое сумматора является результатом сдвига исходного кода влево или вправо на *n* разрядов, знак сдвига определяется знаком *n*. Значения остальных разрядов адресов, в которых помещается *n*, могут быть любыми
		1	$s_i := \begin{cases} \tilde{b}_{i-\tilde{a}_I} \text{ при } 1 \leqslant i - \\ \quad - \tilde{a}_I \leqslant 36; \\ 0 \text{ при } 36 < i - \tilde{a}_I, \\ \quad i - \tilde{a}_I \leqslant 0 \end{cases}$		Модуль целого двоичного числа *n* занимает 13÷17-й разряды ячейки *x*, знак записывается в 18-м разряде, 1 соответствует знаку «—», 0 — знаку «+»; *n*, записанное в ячейке *x*, обозначается x_I $n = (-1)^{\langle x_{18} \rangle} \sum_{j=13}^{17} \langle x_j \rangle \, 2^{j-13}$
		2	$s_i := \begin{cases} \tilde{b}_{i-k_I} \text{ при } 1 \leqslant i - \\ \quad - k_I \leqslant 36; \\ 0 \text{ при } 36 < i - k_I, \\ \quad i - k_I \leqslant 0, \end{cases}$ $\tilde{b}_i := s_i$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
25		3	$s_i := \begin{cases} \tilde{b}_{i-k_I} \text{ при } 1 \leqslant i - k_I \leqslant 36; \\ 0 \text{ при } 36 < i - k_I, \\ i - k_I \leqslant 0 \end{cases}$		При модификации $M=7$ сдвиг производится на m разрядов (см. стр. 427); m занимает $1 \div 5$ разряды ячейки x, знак записывается в 6-м разряде; при этом m обозначается x_{II} $m = (-1)^{«x_6»} \sum_{j=1}^{5} «x_j» \cdot 2^{j-1}$
		4	$s_i := \begin{cases} s_{i-k_I} \text{ при } 1 \leqslant i - k_I \leqslant 36; \\ 0 \text{ при } 36 < i - k_I, \\ i - k_I \leqslant 0, \end{cases}$ $\tilde{b}_i := s_i$		
		5	$s_i := \begin{cases} s_{i-k_I} \text{ при } 1 \leqslant i - k_I \leqslant 36; \\ 0 \text{ при } 36 < i - k_I, \\ i - k_I \leqslant 0, \end{cases}$ $C := k_{II}$ при $\gamma = 1$ $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
25		6	$s_i := \begin{cases} s_{i-\tilde{a}_I} & \text{при } 1 \leqslant i - \tilde{a}_I \leqslant 36; \\ 0 & \text{при } 36 < i - \tilde{a}_I, \\ & i - \tilde{a}_I \leqslant 0, \end{cases}$ $C := k_{II} \boxplus_{II} (C + 1)$		
		7	$s_i := \begin{cases} s_{i-k_{II}} & \text{при } 1 \leqslant i - k_{II} \leqslant 36; \\ 0 & \text{при } 36 < i - k_{II}, \\ & i - k_{II} \leqslant 0, \end{cases}$ $C := k_{II}$ при $\gamma = 0$ $C := C + 1$ в противном случае		
65	Циклический сдвиг *r M* 65 τ_a *a* τ_b *b*	0	$s_i := \begin{cases} \tilde{\tilde{b}}_{i-\tilde{a}_{II}} & \text{при } 1 \leqslant i - \tilde{a}_{II} \leqslant 36; \\ \tilde{b}_{36-\tilde{a}_{II}+i} & \text{при } i - \tilde{a}_{II} \leqslant 0; \\ \tilde{b}_{i-36+\tilde{a}_{II}} & \text{при } i - \tilde{a}_{II} > 36, \end{cases}$ $\tilde{b}_i := s_i$	от 60 до 350	Отличается от операции сдвиг ($\theta = 25$) тем, что содержимое сумматора представляет собой результат циклического сдвига

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
65		1	$s_i := \begin{cases} \tilde{b}_{i-\tilde{a}_{II}} \text{ при } 1 \leqslant i - \tilde{a}_{II} \leqslant 36, \\ \tilde{b}_{36-\tilde{a}_{II}+i} \text{ при } i - \tilde{a}_{II} \leqslant 0, \\ \tilde{b}_{i-36+\tilde{a}_{II}} \text{ при } i - \tilde{a}_{II} > 36 \end{cases}$		
		2	$s_i := \begin{cases} \tilde{b}_{i-k_{II}} \text{ при } 1 \leqslant i - k_{II} \leqslant 36, \\ \tilde{b}_{36-k_{II}+i} \text{ при } i - k_{II} \leqslant 0, \\ \tilde{b}_{i-36+k_{II}} \text{ при } i - k_{II} > 36, \end{cases}$ $\tilde{b}_i := s_i$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
65		3	$s_i := \begin{cases} \tilde{b}_{i-k_{II}} \text{ при} \\ 1 \leqslant i - k_{II} \leqslant 36, \\ \tilde{b}_{36-k_{II}+i} \text{ при} \\ i - k_{II} \leqslant 0, \\ \tilde{b}_{i-36+k_{II}} \text{ при} \\ i - k_{II} > 36, \end{cases}$		
		4	$s_i := \begin{cases} s_{i-k_{II}} \text{ при} \\ 1 \leqslant i - k_{II} \leqslant 36, \\ s_{36-k_{II}+i} \text{ при} \\ i - k_{II} \leqslant 0, \\ s_{i-36+k_{II}} \text{ при} \\ i - k_{II} > 36, \end{cases}$ $\tilde{b}_i := s_i$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
65		5	$s_i := \begin{cases} s_{i-k_{II}} & \text{при } 1 \leqslant i - k_{II} \leqslant 36, \\ s_{36-k_{II}+i} & \text{при } i - k_{II} \leqslant 0, \\ s_{i-36+k_{II}} & \text{при } i - k_{II} > 36, \end{cases}$ $C := k_{II}$ при $\gamma = 1$, $C := C + 1$ в противном случае		
		6	$s_i := \begin{cases} s_{i-k_{II}} & \text{при } 1 \leqslant i - k_{II} \leqslant 36, \\ s_{36-k_{II}+i} & \text{при } i - k_{II} \leqslant 0, \\ s_{i-36+k_{II}} & \text{при } i - k_{II} > 36, \end{cases}$ $C := k_{II} \underset{11}{\boxplus} (C + 1)$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
65		7	$s_i := \begin{cases} s_{i-k_{II}} \text{ при} \\ 1 \leqslant i - k_{II} \leqslant 36, \\ s_{36-k_{II}+i} \text{ при} \\ i - k_{II} \leqslant 0, \\ s_{i-36+k_{II}} \text{ при} \\ 36 < i - k_{II}, \end{cases}$ $C := k_{II}$ при $\gamma = 0$, $C := C + 1$ в противном случае		
26	Пересылка *r M* 26 τ_a *a* τ_b *b*	0	$\tilde{a} := \underline{s}$, $s := \tilde{b}$		Операция предназначена для пересылки содержимого ячеек (сумматора) в другие ячейки (сумматор). При пересылке никаких преобразований кода не производится
		1	$s := \tilde{b}$, $\tilde{a} := 0$		
		2	$s := \tilde{a}$, $C := k_{II}$ при $\gamma = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
26		3	$s := \tilde{a}$, $C := k_{II}$ при $\nu = 1$, $C := C + 1$ в противном случае		
		4	$s := \tilde{a}$, $\tilde{b} := s$		
		5	$s := \tilde{a}$, $C := k_{II}$ при $\omega = 1$ $C := C + 1$ в противном случае		
		6	$\tilde{a}_{II} := \tilde{a}_{I}$, $C := k_{II} \boxplus_{II} (C + 1)$		
		7	$s := \tilde{a}$, $C := k_{II}$ при $\omega = 0$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
30	Абсолютная передача управления *r M* 30 τ_a *a* τ_b *b*	0	$\tilde{a}_\theta := 30$, $\tilde{a}_r := 0$, $\tilde{a}_M := 0$, $\tilde{a}_{\text{I}} := C + 1$, $\tilde{a}_{\text{II}} = 0$, $C := k_{\text{II}}$, $s := s$	56	В зависимости от модификаций производится условная или безусловная передача управления. При модификации 0 в ячейку записывается команда возврата, которая в восьмеричной записи имеет вид: 0 0 30 $k+1$ 0000
		1	$C := k_{\text{II}}$		
		2	$C := k_{\text{II}}$ при $\omega = 1$, $C := C + 1$ в противном случае		
		3	$C := k_{\text{II}}$ при $\omega = 0$, $C := C + 1$ в противном случае		
		4	$C := k_{\text{II}}$ при $\gamma = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
30		5	$C:=k_{II}$ при $\gamma=0$, $C:=C+1$ в противном случае		
		6	$C:=k_{II}$ при $\nu=1$, $C:=C+1$ в противном случае		
		7	$C:=k_{II}$ при $\nu=0$, $C:=C+1$ в противном случае		
70	Относительная передача управления *r M* 70 τ_a *a* τ_b *b*	0	$a_\theta:=30$, $a_r:=0$, $a_M:=0$, $a_I:=C+1$, $a_{II}:=0$, $C:=k_{II}\boxplus_{II}(C+1)$, $s:=s$	56	Отличается от команды абсолютная передача управления тем, что вместо абсолютной передачи управления производится относительная передача управления, при которой содержимое счетчика команд образуется путем прибавления к его первоначальному содержимому передаваемого на него кода
		1	$C:=k_{II}\boxplus_{II}(C+1)$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
70		2	$C := k_{II} \underset{II}{\boxplus} (C+1)$ при $\omega = 1$, $C := C+1$ в противном случае		
		3	$C := k_{II} \underset{II}{\boxplus} (C+1)$ при $\omega = 0$, $C := C+1$ в противном случае		
		4	$C := k_{II} \underset{II}{\boxplus} (C+1)$ при $\gamma = 1$, $C := C+1$ в противном случае		
		5	$C := k_{II} \underset{II}{\boxplus} (C+1)$ при $\gamma = 0$, $C := C+1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
70		6	$C:=k_{\text{II}} \boxplus\limits_{\text{II}} (C+1)$ при $\nu=1$, $C:=C+1$ в противном случае		
		7	$C:=k_{\text{II}} \boxplus\limits_{\text{II}} (C+1)$ при $\nu=0$, $C:=C+1$ в противном случае		
31	Изменение содержимого регистров переадресации *r M* 31 τ_a *a* τ_b *b*	0	$\tilde{a}_n:=\tilde{a}_n-1$, $\tilde{a}_i:=\tilde{a}_i+\tilde{a}_\Delta$, $\tilde{a}_\Delta:=\tilde{a}_\Delta$, $C:=k_{\text{II}}$ при «$\tilde{a}_n$»$\geqslant 0$, $C:=C+1$ в противном случае	58	При выполнении операции содержимое ячейки с номером $\tilde{a}$ изменяется в соответствии с указанными модификациями. Если эта ячейка используется в дальнейшем для автоматической переадресации, то значение $\tilde{a}$ должно лежать в пределах $0\leqslant a\leqslant 7$
		1	$\tilde{a}_n:=\tilde{a}_n-1$, $\tilde{a}_\Delta:=\tilde{a}_\Delta$, $\tilde{a}_i:=\tilde{a}_i+\tilde{a}_\Delta$, $\tilde{a}_\Delta:=\tilde{a}_\Delta$, $C:=k_{\text{II}}$ при «$\tilde{a}_n$»<0, $C:=C+1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
31		2	$\tilde{a}_n := \tilde{a}_n - 1$, $\tilde{a}_i := \tilde{a}_i - \tilde{a}_\Delta$, $C_i := k_{II}$ при «$\tilde{a}_n$» $\geqslant 0$, $C := C + 1$ в противном случае		
		3	$\tilde{a}_n := \tilde{a}_n - 1$, $\tilde{a}_i := \tilde{a}_i - \tilde{a}_\Delta$, $\tilde{a}_\Delta := \tilde{a}_\Delta$, $C := k_{II}$ при «$\tilde{a}_n$» $= 0$, $C := C + 1$ в противном случае		
		4	$\tilde{a}_n := \tilde{a}_n - 1$, $\tilde{a}_i := \tilde{a}_i + \tilde{a}_\Delta$, $\tilde{a}_\Delta := \tilde{a}_\Delta$, $C := k_{II}$ при «$\tilde{a}_n$» $\geqslant 0$ и $\omega = 1$, $C := C + 1$ в противном случае		
		5	$\tilde{a}_n := \tilde{a}_n - 1$, $\tilde{a}_i := \tilde{a}_i + \tilde{a}_\Delta$, $\tilde{a}_\Delta := \tilde{a}_\Delta$, $C := k_{II}$ при «$\tilde{a}_n$» < 0 и $\omega = 1$, $C := C + 1$ в противном случае		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
31		6	$\tilde{a}_n := \tilde{a}_n - 1$, $\tilde{a}_i := \tilde{a}_i - \tilde{a}_\Delta$, $C := k_{II}$ при $\tilde{a}_n = 1$, $C := C + 1$ в противном случае		
		7	$\tilde{a}_n := \tilde{a}_n - 1$, $\tilde{a}_i := \tilde{a}_i - \tilde{a}_\Delta$, $\tilde{a}_\Delta := \tilde{a}_\Delta$, $C := k_{II}$ при $\tilde{a}_n < 0$ и $\omega = 1$, $C := C + 1$ в противном случае		
33	Обращение к магнитной ленте $* M\ 33\ \tau_a\ a\ 0\ N$	0	Производится запись на ленту в зону с номером N $\tilde{a}_\Delta$ кодов, хранящихся в ОЗУ, начиная с ячейки $\tilde{a}_i$. Лента начинает движение в прямом направлении	После выполнения поиска скорость обмена 2000 кодов в секунду	При поиске зоны начальное направление движения ленты определяется модификацией операции. Если при прямом начальном направлении движения первый считанный адрес зоны больше N, то лента начинает двигаться в обратном на-

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
33		1	То же, что и для модификации 0, но лента начинает движение в обратном направлении	(500 *мксек* для каждого кода)	правлении. Обратное направление переключается на прямое, если первый адрес меньше N. Если $\tilde{a}_{24}=1$ или, что то же самое, $\tilde{a}_i \geqslant 4096$, то при всех модификациях выполняется разметка ленты, при которой записывается номер зоны N и производится прогон на расстояние, достаточное для записи $\tilde{a}_\Delta$ кодов
		2	Производится считывание из зоны с номером N $\tilde{a}_\Delta$ кодов. Считываемые коды записываются в ячейки ОЗУ, начиная с $\tilde{a}_i$. Лента начинает движение в прямом направлении		
		3	То же, что и для модификации 2, но лента начинает движение в обратном направлении		
		4	Выполняется то же, что и при модификации 0, после записи размечается адрес следующей зоны, с номером $N+1$		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
33		5	Выполняется то же, что и при модификации 1, после записи размечается адрес следующей зоны с номером $N+1$		
		6	Производится суммирование $\tilde{a}_\Delta$ кодов из зоны N по правилам операции с $\theta=17$, сумма остается в сумматоре Лента начинает движение в прямом направлении		
		7	То же, что и для модификации 6, но лента начинает движение в обратном направлении		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
73	Стирание зон r M 73 τ_a a 0 N	4	Стирается номер и содержимое зоны с номером N, помещающей $\tilde{a}_\Delta$ кодов. Номер следующей зоны заменяется номером зоны N. Лента движется в прямом направлении	То же, что и для $\theta = 33$	
		5	То же, что и для модификации 4, но лента движется в обратном направлении		
34	Ввод без перемотки r * 34 τ_a a * *	$0 \div 7$	При всех модификациях выполняется ввод в ОЗУ, начиная с ячейки $\tilde{a}$ до появления на перфоленте признака «Конец ввода». Если среди вводимых кодов имеется признак «ввод по адресу», в котором указан номер некоторой	≈ 30 *мсек* на каждый код	Если $\tilde{a} = 0$ и на перфоленте нет признака начала ввода, то ввод осуществляется, начиная с нулевой ячейки ОЗУ. Порядок выполнения операции $\theta = 34$ с пульта управления описан в разделе «Пульт управления»

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
34			ячейки ОЗУ — a', то коды, расположенные вслед за признаком, будут вводиться в ячейки, начиная с a'		
74	Ввод с перемоткой $r * 34\ \tau_a\ a * *$	$0 \div 7$	Отличается от операции «Ввод без перемотки» $\theta = 34$ тем, что перфолента предварительно перематывается до прихода признака «Конец перемотки»	30 *мсек* на каждый код (без перемотки)	То же, что и для операции с $\theta = 34$
35	Вывод на перфоленту и печатающее устройство $r\ M\ 35\ \tau_a\ a\ \tau_b\ b$	0	Содержимое сумматора в восьмеричной системе счисления выводится на печатающее устройство	80 для выдачи на приемный регистр выходных устройств. Задержка возникает, если	
		1	Содержимое сумматора в десятичной системе счисления выводится на печатающее устройство		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
35		2	Отличается от модификации 0 тем, что выдается содержимое ячейки $\tilde{a}$	до осуществления вывода опять выполняется команда, осуществляющая вывод на то же устройство	
		3	Отличается от модификации 1 тем, что выдается содержимое ячейки $\tilde{a}$		
		4	Содержимое сумматора в восьмеричной системе счисления выводится на перфоленту		
		5	Содержимое ячейки $\tilde{a}$ в восьмеричной системе счисления выводится на перфоленту		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
37	Останов и передача управления по ключам $r\ M\ 37\ \tau_a\ a\ \tau_b\ b$	0	Машина останавливается $s := \tilde{a}$, $R\text{I} := \tilde{b}$, $\tilde{a} := 0$. Содержимое s и $R\text{I}$ выдается на пульт управления	70	На пульте управления имеются три ключа, служащие для выполнения передачи управления по команде $\theta = 37$
		1	То же, что и для модификации 0		
		2	Если на пульте управления включен ключ 1, то производится переход к ячейке с номером b. $C := k_{\text{II}}$ при включении ключа 1, $C := C + 1$ в противном случае		
		3	То же, что и для модификации 2		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
37		4	Отличается от модификации 2 тем, что переход выполняется при включенном ключе 2		
		5	То же, что и для модификации 4		
		6	Отличается от модификации 3 тем, что переход выполняется при включенном ключе 3		
		7	То же, что и для модификации 6		

Продолжение табл. 14.2

Код операции	Наименование команды и ее вид	Модификация операции	Содержание операции	Время выполнения, *мксек*	Пояснения
00 27 32 36	Пропуск такта $r\ M\ 0\ *\ *\ \tau_b\ b$	0 ÷ 5 и 7	$\underline{s} := \underline{s},$ $\underline{b} := \underline{b},$ $C := C + 1$	56	При всех модификациях, кроме $M = 6$, выполняется «холостой» такт работы машины. При $M = 6$ происходит передача управления
40 61 62 64 66 67 71 72 73 75 76 77		6	$\underline{s} := \underline{s},$ $\underline{b} := \underline{b},$ $C := k_{\text{II}} \boxplus_{11} (C + 1)$		

некоторого адреса — k_{II} ($C := k_{\text{II}}$), во втором случае к содержимому счетчика команд, увеличенному на единицу, прибавляется значение адреса по формуле

$$C := k_{\text{II}} \underset{\text{II}}{\boxplus} (C + 1).$$

Пульт управления

Пульт управления имеет управляющую и сигнальную панели.

На сигнальную панель выведены ряды неоновых лампочек, отображающие содержимое сумматора, регистров арифметического устройства, регистра команд и счетчика команд, а также причины останова машины и режимы работы.

На управляющей панели имеется также ряд клавиш, ключей и кнопок, важнейшими из которых являются следующие:

Клавиши для ручного набора 36-разрядного двоичного кода и перенесения его на регистр команд или на сумматор и регистры АУ (исполнение набранной команды осуществляется при нажатии кнопки «Пуск»); эти клавиши называются в системе команд регистром ρ ($\theta = 57$; 60; 63): клавиши для набора адреса, при совпадении которого с содержимым счетчика команд происходит останов машины.

Кнопки «Пуск» и «Останов» машины.

Четыре клавиши, обеспечивающие четыре режима работы машины:

1) автоматический, при котором команды последовательно выполняются;

2) однотактный, при котором после выполнения каждой команды машина останавливается;

3) повторный, при котором одна и та же команда повторяется многократно;

4) одиночных импульсов, при котором после выполнения действий по одному импульсу такта машина останавливается.

Кнопка «Ввод», при включении которой на регистр команд записывается команда с кодом операции $\theta = 34$; по этой команде производится ввод в ОЗУ, начиная с ячейки, имеющей номер 0001; при поступлении признака «конец ввода» управление передается команде, стоящей в ячейке с номером 0010. Если необходим останов после ввода, то перед нажатием кнопки «Ввод» включают клавишу, обеспечивающую однотактный режим.

0010	0426	0014	0001		Команды
1	1426	4016	5011		
2	0031	0001	0010		
3	0037				
4	0005		0001		
5					
6	0220				2
7	0230				3
0020	0320				4
1	7631	4631	4631		0,2
2	0556				—18
3	7747	0243	6561		—0,39
4					
5					
6					
7	+ 1 + 100	0000			1
0030	+ 0 — 333	3333			$-1/8$
1	+ 0 + 200	0000			0,2
2	+ 3 + 987	3456			+ 987, 3456
3					
4					
5					
6					
7					

Двоичные числа

Десятичные числа

Рис. 14.2. Образец записи команд и чисел на бланке.

Подготовка данных. Ввод и вывод

Числа и команды, предназначенные для пробивки и ввода в машину, записываются на стандартных бланках.

Один из возможных видов бланка с образцами записи команд, двоичных чисел и десятичных чисел представлен на рис. 14.2. Двоичные числа и команды записываются в виде

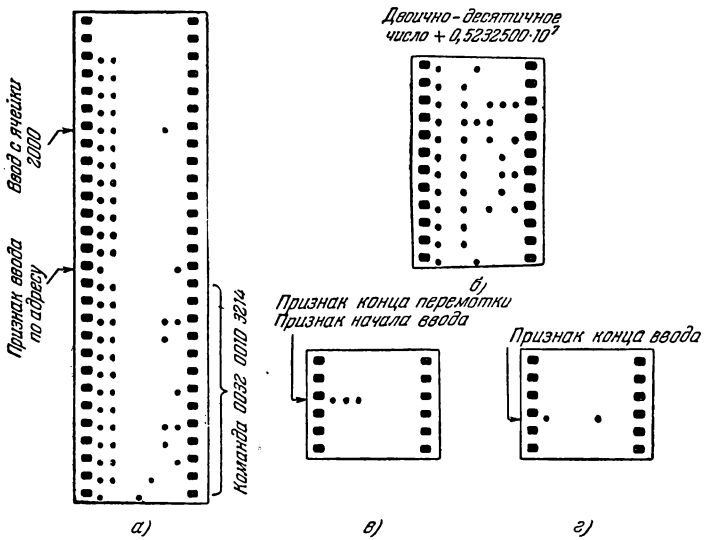


Рис. 14.3. Образцы кодирования на перфоленте.

восьмеричных чисел. Команда и двоичное число кодируются двенадцатью восьмеричными числами, которые для удобства чтения объединяют в четыре группы, обозначающие соответственно: r; $M\theta$; A; B. При записи десятичных чисел знаки числа и порядка обозначаются символами «+» и «—».

Перфорация исходной информации производится на клавишном устройстве, соединенном с перфоратором. Перфорация сопровождается печатью кодов на пишущей машинке для визуального контроля.

Образцы пробивки различных кодов на перфоленте приведены на рис. 14.3. Пробивки располагаются на семи дорожках, из которых три — управляющие, четыре — кодовые.

Комбинации пробивок, обозначающие признаки, приведены в табл. 14.3. Двоичные числа и команды пробиваются на 12 строках ленты, десятичные на 11 строках.

Таблица 14.3. Признаки, перфорируемые на ленте

Дорожки, имеющие перфорацию	Название признака	Пояснения
7	Маркер	Сопровождает каждую цифру или признак
3	Конец ввода	Используется при операциях $0=34$ и $0=74$
4	Признак записи	Разрешает запись кода в ячейку
5	Признак двоично-десятичного числа	Пробивается во всех строках
6	Признак команды и двоичного числа	Пробивается во всех строках
1	Ввод по адресу	На 5 и 6 дорожках пробивки отсутствуют
5—6	Конец перемотки	

Отрицательный знак порядка двоично-десятичного числа пробивается на 2-й дорожке. Положительный знак двоично-десятичного числа пробивается на 3-й и 4-й дорожках, отрицательный — на 1-й, 3-й и 4-й дорожках.

Из машины на перфоленту выдаются только двоичные числа или команды.

ГЛАВА 15

ПРОГРАММНО-УПРАВЛЯЕМАЯ МАШИНА *СТРЕЛА*

Основные характеристики машины *Стрела*

Тип машины	вычислительная
Класс машины	средняя
Среднее число основных операций в секунду	2000
Рабочая система счисления	двоичная
Форма представления числа	с плавающей запятой
Длина кода числа	43 двоичных разряда (мантисса со знаком — 36 разрядов, порядок со знаком — 7 разрядов)
Диапазон представления чисел	$2^{-64} \leqslant \lvert x \rvert \leqslant (1 - 2^{-35})\, 2^{63}$
Точность вычислений	10—11 десятичных значащих цифр
Порядок выполнения команд	заданный (естественный)
Адресность команды	трехадресная
Виды запоминающих устройств	оперативное запоминающее устройство на электронно-лучевых трубках; емкость 2048 43-разрядных двоичных кодов; внешнее запоминающее устройство на магнитных лентах; количество лент две; емкость ленты 50 000 43-разрядных кодов; скорость записи (считывания) 750 чисел в секунду; долговременное запоминающее устройство: накопитель стандартных программ емкостью 208 команд, накопитель констант емкостью 256 43-разрядных кодов

Виды ввода и вывода	вводное устройство: электромеханическое со скоростью ввода 2400 кодов в минуту; ввод с восьмидесятиколонных перфокарт; выводное устройство — перфоратор со скоростью вывода 600 кодов в минуту; вывод осуществляется на перфокарты
Количество электронных ламп	5851
Потребляемая мощность (без охлаждения)	124 *квт*
Занимаемая площадь	400 $м^2$

Машина *Стрела* — универсальная; временной порядок эксплуатации — круглосуточный.

Краткие сведения об основных устройствах машины

Все устройства машины, кроме внешних, которые работают автономно, соединены между собой системой магистралей. Первая, вторая и третья магистрали («числовые») служат для передачи соответственно первого и второго исходных чисел и результата операции. Четвертая магистраль (командная) служит для передачи команд из оперативного запоминающего устройства или из накопителя стандартных программ в устройство управления, в блок управления внешним запоминающим устройством и др. Пятая магистраль (сигнальная) служит для передачи управляющих сигналов между различными устройствами.

Запоминающие устройства (ЗУ). В машине *Стрела* существует три вида ЗУ: оперативное (ОЗУ), внешнее (ВЗУ) и долговременное (ДЗУ).

ОЗУ выполнено на электронно-лучевых трубках. Время обращения к ОЗУ: выборка 15 *мксек*, запись 30 *мксек*. В ячейках ОЗУ могут храниться одновременно 2048 43-разрядных кодов. Ячейки ОЗУ пронумерованы восьмеричными числами 0000, 0001, ..., 3777. Из ячейки с номером 0000 всегда выдается число нуль.

ВЗУ состоит из двух блоков магнитной ленты. Запись (считывание) на магнитную ленту производится только из ОЗУ. Емкость магнитной ленты 50 000 кодов. На магнитную ленту коды записываются (считываются) зонами. Наибольшее число допустимых кодов в зоне определяется

емкостью ОЗУ (2048). Наибольшее число зон 512. Зоны магнитной ленты каждого блока имеют восьмеричные номера 000, 001, ..., 777. Блоки магнитной ленты имеют номера: один 5000_8 или 7000_8, другой 4000_8 или 6000_8. Чтобы указать определенную зону определенного блока, задают сумму их номеров. Скорость движения магнитной ленты — 0,6 *м/сек*, плотность записи 1,5 *имп/мм*.

ДЗУ выполнено на диодной сетке. В машине *Стрела* различают два вида ДЗУ: устройство выдачи констант (УВК) и накопитель стандартных подпрограмм (НСП). УВК и НСП состоят каждое из 16 сменных секций. В каждой секции размещаются 16 ячеек. Секции НСП имеют номера $60_8 \div 77_8$. В командах обращения к СП НСП адрес секции пишется в коде операции.

Арифметическое устройство (АУ) работает в обратном коде.

Все операции, выполняемые в АУ, разбиты на три группы. В соответствии с этим имеется три узла АУ, каждый из которых реализует операции одной из групп. Узел сложения и вычитания (УСВ) предназначен для выполнения операций сложения, вычитания, вычитания модулей, специального сложения, специального вычитания, контрольного суммирования. Узел умножения предназначен для выполнения операций умножения, сложения порядков, вычитания порядков, переноса числа с присвоением знака другого числа. Узел дополнительных операций предназначен для производства операции поразрядного логического умножения, поразрядного логического сложения, формирования, сдвига.

Устройство управления (УУ). Назначение УУ сводится к выполнению такта (цикла) работы машины, за время которого производится выбор из ОЗУ очередной команды, выбор исходных чисел, выполняется операция над этими числами и результат записывается в ОЗУ. Кроме того, УУ обеспечивает выполнение групповых команд (см. ниже).

УУ включает в себя следующие основные узлы:

Регистр номера команды имеет 11 разрядов. На этом регистре находится номер команды, которая должна быть выполнена в очередном такте работы машины.

Регистр команд имеет 43 разряда. На этот регистр записывается команда в начальный момент такта работы машины. Регистр команд при выполнении групповых команд работает как счетчик адресов чисел.

Одиннадцатиразрядный регистр-счетчик, определяющий конец выполнения групповых операций.

Два 43-разрядных регистра чисел. На этих регистрах записываются два исходных числа (первое и второе) для выполнения арифметических и других операций. Регистры чисел фактически являются входными регистрами арифметического устройства.

Задающий генератор и специальный счетчик (программный датчик), вырабатывающий все основные управляющие напряжения и импульсы.

Устройство ввода и вывода (УВВ). Ввод и вывод на машине *Стрела* производится с перфокарт. Устройство ввода — электрическое с ощупывающим механизмом. Перед вводом колоду перфокарт помещают в магазин вводного устройства. Скорость ввода 2400 чисел в минуту.

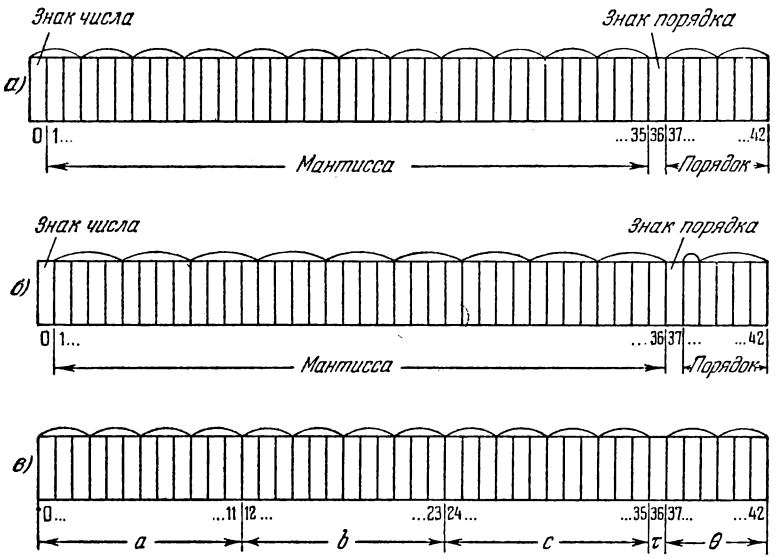


Рис. 15.1. Распределение разрядов ячейки памяти: *а*) для двоичного числа; *б*) для двоично-десятичного числа; *в*) для команды

Устройство вывода (перфоратор результатов) предназначено для выдачи чисел или команд из ОЗУ на перфокарты. Скорость перфорации результатов 600 чисел (50 перфокарт) в минуту.

Представление чисел и команд

На рис. 15.1 показано распределение разрядов ячейки памяти машины *Стрела* при хранении в ней двоичного и двоично-десятичного чисел с плавающей запятой.

Наличие в знаковом разряде мантиссы и порядка цифры 1 соответствует знаку минус, а цифры 0 — знаку плюс. Порядок и мантисса представляются в прямом коде.

На том же рисунке показано распределение разрядов ячейки памяти при хранении в ней команды. Команды машины *Стрела* имеют следующую структуру: $\overset{\frown}{abc}\tau\theta$, где a, b и c — соответственно I, II и III адреса команды; θ — код операции; τ — контрольный сигнал.

Скобками $\overset{\frown}{\ }$ на рис. 15.1 указаны объединения двоичных разрядов в восьмеричные и десятичные цифры, применяемые при записи чисел и команд.

Система операций и команд

Описание системы операций и команд машины *Стрела* приведено в табл. 15.1.

Замечания. 1. Если в команде первый и второй адреса являются адресами констант УВК, то по обоим адресам выдается в первую и вторую магистрали одна и та же константа УВК, адрес которой получается путем поразрядного логического сложения первого и второго адресов данной команды.

2. Восьмеричная цифра x кода операции $30 + x$ указывает на переменные адреса в исполнительной команде следующим образом:

Значение x	Двоичное представление x	Переменные адреса		
		a	b	c
0	000	нет	нет	нет
1	001	нет	нет	да
2	010	нет	да	нет
3	011	нет	да	да
4	100	да	нет	нет
5	101	да	нет	да
6	110	да	да	нет
7	111	да	да	да

Таблица 15.1. Система операций и команд машины «Стрела»

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
01	Сложение $a\ b\ c\ \tau$ 01	$c := a + b$	$c < 0$	$c_P > 63$	Числа «a» и «b» алгебраически складываются. Результат операции записывается в ячейку c
02	Поадресное сложение $a\ b\ c\ \tau$ 02	$c_{\mathrm{I}} := a_{\mathrm{I}} \underset{12}{\boxplus} b_{\mathrm{I}}$, $c_{\mathrm{II}} := a_{\mathrm{II}} \underset{12}{\boxplus} b_{\mathrm{II}}$, $c_{\mathrm{III}} := a_{\mathrm{III}} \underset{12}{\boxplus} b_{\mathrm{III}}$, $c_{\tau,\theta} := a_{\tau,\theta}$			«Адреса» числа «a» складываются с одноименными «адресами» числа «b», переносы из старших разрядов адресов блокируются. Порядок числа «c» совпадает с порядком числа «a». Результат операции записывается в ячейку c
03	Вычитание $a\ b\ c\ \tau$ 03	$c := a - b$	$c < 0$	$c_P > 63$	Из числа «a» вычитаются числа «b». Результат записывается в ячейку c
04	Вычитание модулей $a\ b\ c\ \tau$ 04	$c := \lvert a \rvert - \lvert b \rvert$	$c < 0$		Из абсолютной величины числа «a» вычитается абсолютная величина «b», результат записывается в ячейку c

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
05	Умножение *a b c* τ 05	$c := a \cdot b$	$\lvert c \rvert \geqslant 1$	$c_P > 63$	Числа «*a*» и «*b*» перемножаются. Результат записывается в ячейку *c*. Если хотя бы один из сомножителей ненормализован, то результат операции равен нулю
06	Сложение порядков *a b c* τ 06	$c := a \cdot 2^{(a_P + b_P)}$	$\lvert c \rvert \geqslant 1$	$c_P > 63$	Порядок числа «*a*» складывается с порядком числа «*b*». В ячейку *c* записывается число, имеющее мантиссу числа *a* и порядок, равный «a_P» + «b_P». Нормализация результата не производится
07	Вычитание порядков *a b c* τ 07	$c := a \cdot 2^{(a_P - b_P)}$	$\lvert c \rvert \geqslant 1$	$c_P > 63$	Из порядка числа «*a*» вычитается порядок числа «*b*». В ячейку *c* записывается число, имеющее мантиссу числа *a*, и порядок, равный «a_P» — «b_P». Нормализация результата не производится

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
10	Перенос числа с присвоением знака другого числа $a\ b\ c\ \tau\ 10$	$c := \operatorname{sign} b \cdot \lvert a \rvert$	$\lvert c \rvert \geqslant 1$		В ячейку c записывается число, имеющее мантиссу и порядок числа «a» и знак числа «b». Нормализация результата не производится
11	Выделение части $a\ b\ c\ \tau\ 11$	$c_i := a_i \wedge b_i$ $(i = 0, 1, \ldots, 42)$	$c = 0$		Содержимое ячейки a поразрядно логически умножается на содержимое ячейки b. Результат заносится в ячейку c
12 *)	Сложение без округления $a\ b\ c\ \tau\ 12$	$c := a + b$	$c = 0$	$c_P > 63$	Число «a» алгебраически складывается с числом «b». Результат записывается в ячейку c. Операция выполняется так же, как и сложение с $\theta = 01$, с тем лишь различием, что результат не округляется

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
13	Формирование *a b c* т 13	$c_i := a_i \vee b_i$ $(i = 0, 1, \ldots, 42)$	$c = 0$		Содержимое ячейки *a* поразрядно логически складывается с содержимым ячейки *b*. Результат записывается в ячейку *c*
14	Сдвиг *a b c* т 14	$c_i := \begin{cases} a_{i-n}, 0 \leqslant i-n \leqslant 42, \\ 0, \ i-n < 0 \\ \quad \text{или} \\ \quad i-n > 42 \end{cases}$ $(0 \leqslant i \leqslant 42_{10})$. $n = -b_P$, если $b < 4000_8$; $n = -m$, если $b = 4000_8 + m$ или $b = 5000_8 + m$; $n = +m$, если $b = 4100_8 + m$ или $b = 5100_8 + m$	$c = 0$		При $b < 4000_8$ содержимое всех разрядов ячейки *a* сдвигается на $\lvert «b_P» \rvert$ разрядов. При $«b_P» > 0$ сдвиг производится влево, при $«b_P» < 0$ сдвиг производится вправо. При $b > 4000_8$ содержимое всех разрядов ячейки *a* сдвигается на *m* разрядов влево, если $b = 4000_8 + m$ (или если $b = 5000_8 + m$), и на *m* разрядов вправо, если $b = 4100_8 + m$ (или $b = 5100_8 + m$)

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
15	Поадресное вычитание *a b c* τ 15	$c_{\text{I}} := a_{\text{I}} \underset{12}{\boxplus} [b_{\text{I}}]_{\text{обр}}$, $c_{\text{II}} := a_{\text{II}} \underset{12}{\boxplus} [b_{\text{II}}]_{\text{обр}}$, $c_{\text{III}} := a_{\text{III}} \underset{12}{\boxplus} [b_{\text{III}}]_{\text{обр}}$, $c_{\tau,\theta} := a_{\tau,\theta}$			Из «адресов» числа «*a*» вычитаются одноименные «адреса» числа «*b*». Результат записывается в ячейку *c*. Переносы из старших разрядов адресов блокируются. Порядок числа «*c*» совпадает с порядком числа «*a*»
16	Сравнение *a b c* τ 16	$c_i := a_i \approx b_i$ $(i = 0, 1, \ldots, 42)$	$c \neq 0$		Производится поразрядное отрицание равнозначности содержимого ячейки *a* с содержимым ячейки *b*. Результат записывается в ячейку *c*
17	Контрольное суммирование *a b c* τ 17	$c := a \underset{43}{\boxplus} b$			Число «*a*» складывается по всем разрядам с числом «*b*» с циклическим переносом единицы из нулевого разряда в 42-й разряд. Результат записывается в ячейку *c*

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
20	Условный переход первого типа *a b c* τ 20	$C := a$, если $\omega = 0$; $C := b$, если $\omega = 1$; $c := 0$	≡		Передается управление к команде «*a*», если при выполнении предыдущей команды $\omega = 0$, и к команде «*b*», если $\omega = 1$. Ячейка *c* очищается
21 *)	Подвод магнитной ленты *a* ∗ *c* τ 21				Производится подвод зоны магнитной ленты, причем, если $a = 4000_8 + m$ (или $6000_8 + m$) $(1 \leqslant m \leqslant 777_8)$, то производится подвод *m*-й зоны первого блока магнитной ленты, если $a = 5000_8 + m$ (или $7000_8 + m$), то производится подвод *m*-й зоны второго блока магнитной ленты. Подвод магнитной ленты осуществляется одновременно с выполнением других команд, не обращающихся к магнитной ленте. Ячейка с номером *c* очищается

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
26 *)	Сравнение с остановом при несовпадении $a\ b\ c\ \tau\ 26$	$c_i := a_i \dot{\approx} b_i$ $(i = 0, 1, \ldots, 42)$	$c \neq 0$		Отличается от команды «сравнение» с кодом операции 16 тем, что при «c» $\neq 0$ машина останавливается
27	Условный переход второго типа $a\ b\ c\ \tau\ 27$	$C := a$, если $\omega = 0$; $C := b$, если $\omega = 1$; $c_{\text{I}} := я\,(k+1)$, $c_{\text{II}} := я\,(k+1)$, $c_{\text{III}} := k_{\text{III}}$, $c_{\tau,\theta} := 020$	$\equiv$		Команда выполняется так же, как и команда с кодом операции $\theta = 20$, с тем лишь различием, что в ячейку c записывается «команда возврата»
$30 + x$	Предварительная команда групповой операции $0000\ n\ *\ \tau\ 30 + x$				Следующая («исполнительная») команда групповой операции выполняется $n+1$ раз; после каждого исполнения команды ее переменные адреса увеличиваются на единицу. Сигнал ω групповой операции равен 1, если хотя бы один раз при выполнении исполнительной команды $\omega = 1$.

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
					Машина остановится, если при выполнении «исполнительной команды» вырабатывается $\varphi = 1$. Нажатием кнопки «пуск» на пульте управления машина начинает продолжать выполнять групповую операцию
40	Останов *a b c* τ 40		Всегда		Машина останавливается и на пульт управления выдаются числа «*a*» и «*b*». Ячейка с номером *c* очищается
41	Перенос чисел с перфокарт в память * *n c* τ 41				С перфокарт через вводное устройство в ячейки $c, c+1, \ldots, c+n$ ОЗУ вводится $n+1$ число
42	Групповой останов * *n c* τ 42		Всегда		Машина останавливается. Останов будет повторяться *n* раз, если нажимать кнопку «пуск» на пульте управления. После этого машина перейдет к выполнению следующей команды. Ячейка *c* очищается.

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
43	Перенос чисел с МЛ в память *а п с т* 43				Производится перенос $n+1$ чисел из зоны с номером P магнитной ленты первого блока, если $a = 4000_8 + P$ (или $a = 6000_8 + P$), и второго блока, если $a = 5000_8 + P$ (или $a = 7000_8 + P$), в ячейки c, $c+1$, ..., $c+n$ ОЗУ. (Если в зоне P количество чисел m меньше чем $n+1$, то после считывания m чисел в ячейки c, $c+1$, ..., $c+m-1$ ОЗУ машина вновь начнет переносить числа с той же зоны P, начиная с первого, в ячейки $c+m$, $c+m+1$, ..., и этот процесс будет продолжаться до тех пор, пока общее количество чисел, перенесенных в ОЗУ, не окажется равным $n+1$.) Если $a < 4000_8$, то машина останавливается и $c{:=}0$

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
44	Перенос чисел из памяти на перфокарты *а п с* т 44				Производится перенос чисел «*a*», «$a+1$», ..., «$a+n$» из ОЗУ на перфокарты. При этом ячейка *c* очищается
45	Перенос чисел из памяти в память *а п с* т 45	$я(c+i) := я(a+i)$ $(i = 0, 1, \ldots, n)$			Производится перенос чисел «*a*», «$a+1$», ..., «$a+n$» из ОЗУ соответственно в ячейки *c*, $c+1, \ldots, c+n$ ОЗУ
46	Перенос чисел из памяти на магнитную ленту *а п с* т 46				Производится перенос чисел «*a*», «$a+1$», ..., «$a+n$» из ОЗУ в зону *m* магнитной ленты первого блока, если $c = 4000_8 + m$ (или $c = 6000_8 + m$), и второго блока, если $c = 5000_8 + m$ (или $c = 7000_8 + m$). Если $c < 4000_8$, то машина останавливается и в ячейку *c* записывается число нуль

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
47	Разметка зоны МЛ $a\ n * \tau\ 47$				На участке магнитной ленты, находящейся под магнитными головками, записывается номер зоны, указанный в первом адресе кода «a». Далее лента начинает двигаться вперед, производя стирание ранее хранившихся кодов. Движение магнитной ленты заканчивается, как только длина участка, пройденного под магнитными головками, может вместить $n+1$ кодов
62	Вычисление обратной величины $a\ n\ c\ \tau\ 62$	$я\ (c+i) := 1 : я\ (a+i)$ $(i = 0, 1, \ldots, n)$		Если хотя бы одно из $a+i=0$	Вычисляются числа $1 :$ «a»; $1 :$ «$a+1$»; ...; $1 :$ «$a+n$» и записываются в ячейки с номерами c, $c+1$, ..., $c+n$

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
63	Извлечение квадратного корня *а п с т* 63	$я(c+i) := \sqrt{я(a+i)}$ $(i = 0, 1, \ldots, n)$, $я\,0002 := 1 : \sqrt{я(a+n)}$			Вычисляются числа $\sqrt{«a»}$, $\sqrt{«a+1»}$, ..., $\sqrt{«a+n»}$ и записываются соответственно в ячейки с номерами c, $c+1$, ..., $c+n$. В ячейке с номером 0002 получается величина $1 : \sqrt{«a+n»}$
64	Вычисление показательной функции *а п с т* 64	$я(c+i) := e^{я(a+i)}$ $(i = 0, 1, \ldots, n)$			Вычисляются числа $e^{«a»}$; $e^{«a+1»}$; ...; $e^{«a+n»}$ и записываются соответственно в ячейки c, $c+1$, ..., $c+n$
66	Вычисление логарифма *а п с т* 66	$я(c+i) := \ln я(a+i)$ $(i = 0, 1, \ldots, n)$			Вычисляются числа $\ln «a»$; $\ln «a+1»$; ...; $\ln «a+n»$ и записываются соответственно в ячейки c, $c+1$, ..., $c+n$

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
67	Вычисление синуса *а п с т* 67	$я(c+i) := \sin я(a+i)$ $(i = 0, 1, \ldots, n)$, $я\,0002 := \cos я(a+n)$			Вычисляются числа sin «a»; sin «$a+1$»; ...; sin «$a+n$» и записываются соответственно в ячейки c, $c+1$, ..., $c+n$. В ячейке с номером 0002 получается величина cos «$a+n$»
70	Перевод чисел в десятичную систему *а п с т* 70				Числа «a», «$a+1$», ..., «$a+n$» переводятся из двоичной системы счисления в двоично-десятичную и записываются соответственно в ячейки c, $c+1$, ..., $c+n$

Продолжение табл. 15.1

Код операции	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
72	Перевод в двоичную систему *a n c* τ 72				Числа «a», «$a+1$», ..., «$a+n$» переводятся из двоично-десятичной системы счисления в двоичную и записываются соответственно в ячейки c, $c+1$, ..., $c+n$
73	Вычисление арктангенса *a n c* τ 73	$я(c+i) := \operatorname{arctg} я(a+i)$ $(i = 0, 1, \ldots, n)$			Вычисляются числа arctg «a»; arctg «$a+1$»; ...; arctg «$a+n$» и записываются соответственно в ячейки c, $c+1$, ..., $c+n$
74	Вычисление арксинуса *a n c* τ 74	$я(c+i) := \arcsin я(a+i)$ $(i = 0, 1, \ldots, n)$			Вычисляются числа arcsin «a»; arcsin «$a+1$»; ... ... arcsin «$a+n$» и записываются соответственно в ячейки c, $c+1$, ..., $c+n$

*) Эти команды введены в процессе модернизации машины и могут иметь на разных образцах машины коды, отличные от указанных в таблице.

3. Если в команде первый или второй адрес больше 3777_8 и меньше 7400_8, исключая, конечно, команды обращения к магнитной ленте (коды $\theta = 43, 46, 21$), то при выполнении команды в соответствующую магистраль посылается нуль.

4. При выполнении команд с кодами операции, не предусмотренными системой команд машины, в ячейку c посылается нуль.

5. При выполнении команд с $\theta = 62 \div 74$ $\varphi = 1$, если порядок хотя бы одного из результатов $(c + i)$ больше 63.

Пульт управления

Пульт управления имеет две панели: визуальную и управляющую.

На визуальной панели располагаются ряды неоновых лампочек, отражающих содержимое первой, второй и третьей числовых магистралей, регистра команд, счетчика команд. Отдельные лампочки регистрируют значение сигнала ω, сигнала переполнения φ, случаи останова машины, условного останова, останова по заданному адресу или по 3-му адресу.

На панели управления имеются следующие тумблеры, клавиши и кнопки:

Тумблерные регистры «Запись числа» и «Запись команды», имеющие по 43 тумблера. При включенном разрешающем тумблере позволяют число или команду, набранную на соответствующем тумблерном регистре, записать в ОЗУ.

Тумблерный регистр набора команды, с помощью которого при включенном разрешающем тумблере можно производить изменение содержимого регистра команд.

Тумблер «Однотактный режим». После выполнения каждой команды в этом режиме машина останавливается.

Тумблер «Условный останов», включенное положение которого вызывает останов машины после выполнения команды, имеющей контрольный сигнал $\tau = 1$.

Тумблер «Останов по 3-му адресу» включает режим, при котором машина останавливается, если третий адрес выполняемой команды совпадает с номером ячейки, набранным на тумблерном регистре записи команд.

Кнопка «Пуск». При нажатии на нее машина переходит либо к выполнению последующих невыполненных операций из всей группы операций, если останов был произведен на

групповой команде, либо к выполнению следующей очередной команды.

Кнопка «Останов».

Кнопка «Начальный пуск», при нажатии на которую в ячейки 0004 ÷ 0017 вводится содержимое одной перфокарты, находящейся в УВВ, и управление передается команде (0004).

Кнопки «Стирание Н» и «блокировка Н», при одновременном нажатии на которые во все ячейки ОЗУ посылается нуль.

Кнопка «Выдача команды» позволяет менять в ОЗУ или вызывать из ОЗУ на пульт управления содержимое ячейки памяти. При этом на регистре записи числа набирается код, а на регистре набора команды по третьему (при записи) или по первому (второму) адресу набирается номер ячейки ОЗУ.

Тумблеры, предназначенные для блокировки движения первого или второго блоков магнитной ленты и записи на блоки магнитной ленты.

Тумблер модификации адресов блоков магнитных лент, при верхнем положении которого первый блок магнитной ленты имеет адрес 5000_8 (или 7000_8), второй блок — 4000_8 (или 6000_8). При нижнем положении этого тумблера адреса блоков меняются на обратные.

В центре управляющей панели размещается электронная трубка («растр»), позволяющая визуально наблюдать содержимое того или иного разряда всех ячеек ОЗУ. Переключение «растра» с одного разряда на другой осуществляется путем нажатия на кнопку с надписью «точно». При этом номер разряда ячеек ОЗУ фиксируется на семи неоновых лампочках, расположенных в ряд и размещенных над «растром».

Подготовка данных. Ввод и вывод

Подлежащий вводу в машину материал (программа, вспомогательные константы и т. д.) программист записывает на специальных бланках. Образец бланков приведен на рис. 15.2.

Далее этот материал с помощью клавишного устройства и перфоратора переносится на перфокарты в виде пробивок. Образец перфокарты с пробивками см. на рис. 15.3 (на этом

рисунке пробивки соответствуют материалу, приведенному на рис. 15.2). Для чтения пробивок на перфокарте применяется специальный шаблон. Содержимое перфокарт может быть отпечатано на печатающем устройстве.

Номер ячейки памяти	Знак числа	I адрес	II адрес	III адрес	Контрольный сигнал	Код операции	Пояснения
3171	+	237	601	000	—	04	$+0{,}237601 \cdot 10^{-4}$ } Десятичные
3172	—	100	001	961	+	03	$-0{,}100001961 \cdot 10^{+8}$ } числа
0040		0115	0000	0010	0	67	}
0041		0010	0114	0010	0	01	} Команды
0042		0040	7423	0040	0	02	}

Рис. 15.2. Образец записи чисел и команды на бланке.

Следует особо отметить тот факт, что в 18-й колонке перфокарты пробивается так называемый «маркер кода», который в устройстве ввода воспринимается как признак чтения кода (команды или числа) с соответствующей строки перфокарты.

Отперфорированный материал следует ввести в машину. Ввод осуществляется по команде с кодом операции 41. Начальный ввод производится путем нажатия на пульте управления кнопки «Начальный пуск». При этом числа и команды с одной перфокарты вводятся в ячейки 0004 — 0017 и передается управление команде «0004».

Вывод из машины производится по команде с кодом операции 44.

Машина *Стрела* ВЦ АН СССР

В ВЦ АН СССР имеется машина *Стрела* с системой операций и команд, совпадающей, за некоторым исключением, с системой операций и команд машины *Стрела*, приведенной выше.

В настоящем описании машины *Стрела* ВЦ АН СССР мы приведем лишь отличительные особенности этой машины.

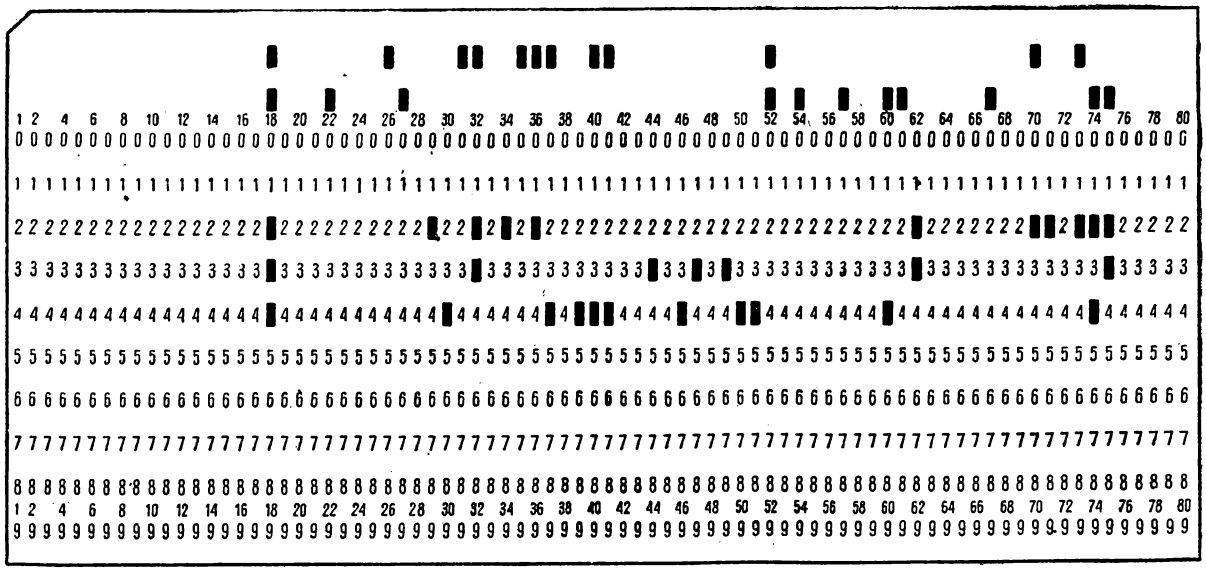


Рис. 15.3. Перфокарта машины *Стрела*.

Основные характеристики машины *Стрела* ВЦ АН СССР

Среднее число основных операций в секунду	20 000
Виды запоминающих устройств	оперативное запоминающее устройство на магнитных сердечниках; емкость 4096 43-разрядных двоичных кодов; время выборки 10 *мксек*; запоминающее устройство на магнитном барабане; емкость 5120 43-разрядных двоичных кодов; среднее время обращения — 40 *мксек*; запоминающее устройство на трех блоках магнитной ленты.
Вывод из машины	печатающее устройство; работает со скоростью печати 18 чисел в секунду.

Краткие сведения об основных устройствах машины

Арифметическое устройство счетно-импульсное, параллельно-последовательного действия. Время выполнения основных операций: сложение (вычитание) 30—40 *мксек*; сравнение чисел, логическое умножение, выделение целой части — 5 *мксек*; сложение порядков — 10 *мксек*; умножение — 70 *мксек*; деление нормализованных чисел — 550 *мксек*.

Запоминающие устройства. Оперативное запоминающее устройство на магнитных сердечниках емкостью 4096 43-разрядных двоичных кодов; время обращения — 10 *мксек*. Ячейки занумерованы восьмеричными числами 0000, 0001, ..., 7777. 43-разрядный код, записанный по команде в ячейку с номером 0000, может быть использован только в следующей команде.

Запоминающее устройство на магнитном барабане емкостью 5120 43-разрядных двоичных кодов. Среднее время обращения к магнитному барабану — 40 *мксек*. Запись (считывание) на магнитный барабан производится зонами (группами). Максимальное количество зон 5. На каждую зону может быть записано не более 1024 43-разрядных двоичных кодов. Запись (считывание) на магнитный барабан производится по командам. Зоны номеруются восьмеричными числами 0001, 0002, ..., 0005. На зоне с номером 0001 записана библиотека стандартных программ.

Запоминающее устройство на магнитных лентах емкостью 250 000 43-разрядных двоичных кодов. Имеются три блока магнитных лент с номерами 4000, 5000 и 6000.

В машине *Стрела* ВЦ АН СССР нет долговременного запоминающего устройства, а команды с кодом операции $60_8 + x$ ($0 \leqslant x \leqslant 17_8$) используются для других целей.

Печатающее устройство работает со скоростью печати 18 чисел в секунду. Вывод на печатающее устройство производится по команде с кодом операции 56.

Система операций и команд

В табл. 15.2 приведены операции и команды, отличающиеся от операций и команд, описанных в табл. 15.1, или не содержащиеся в этой таблице.

Структура команды осталась прежней, т. е. имеет вид $a\ b\ c\ \tau\ \theta$, но значение τ на действия машины не влияет.

Замечание 1. Восьмеричная цифра x кода операции $60_8 + x$ указывает на переменные адреса таким же образом, как и в команде с кодом операции $30 + x$ (см. стр. 497).

Библиотека стандартных подпрограмм (БСП) включает в свой состав следующие подпрограммы: СП1 (подпрограмма обмена информации или иначе подпрограмма «диспетчер»). По этой подпрограмме производится: а) ввод чисел (двоично-десятичных или двоичных) с перфокарт в ячейки ОЗУ; б) перенос чисел (двоичных или двоично-десятичных) из ячеек памяти на магнитную ленту; г) перенос чисел с магнитной ленты в ячейки ОЗУ; д) перенос чисел из ячеек ОЗУ на магнитный барабан; е) перенос чисел с магнитного барабана в ячейки ОЗУ; СП2-подпрограмма перевода чисел из десятичной системы счисления в двоичную и обратно; СП3, СП4, СП5, СП6, СП7, СП8 — соответственно подпрограммы вычисления элементарных функций $\sqrt{x}$, $\sin x$, e^x, $\ln x$, $\operatorname{arctg} x$ по заданному аргументу x (x — нормализованное двоичное число), которое должно находиться в ячейке 0001. Результат выполнения каждой из этих подпрограмм помещается в ячейку 0003, при этом содержимое ячейки 0001 не изменяется. В ячейку 0002 в результате выполнения СП3 получается число $1 : \sqrt{x}$, а в результате выполнения СП8 — $\arccos x$.

БСП записана в первую зону магнитного барабана в ячейках $0000 \div 0647_8$. Запись в эту зону заблокирована.

Рекомендуется считывание БСП осуществлять по подпрограмме, приведенной в табл. 15.3. По этой подпрограмме

Таблица 15.2. Система операций и команд машины *Стрела* ВЦ АН СССР

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
14	Сдвиг по порядку $a\ b\ c\ \tau\ 14$	$c_i := \begin{cases} a_{i-n}, \text{ если } 0 \leqslant i-n \leqslant 42, \\ 0, \text{ если } i-n<0 \text{ или } \\ \qquad i-n>42 \end{cases}$ $(0 \leqslant i \leqslant n;\ n = \langle\!\langle -b_P \rangle\!\rangle)$	$c = 0$		Содержимое всех разрядов ячейки a сдвигается на $\lvert\langle\!\langle b_P \rangle\!\rangle\rvert$ разрядов. При $\langle\!\langle b_P \rangle\!\rangle > 0$ сдвиг производится влево, при $\langle\!\langle b_P \rangle\!\rangle < 0$ — вправо
21	Пересылка числа с 1-го тумблерного регистра пульта управления в ячейку памяти $* * c\ \tau\ 21$				Число, набранное на первом тумблерном регистре набора числа пульта управления, переносится в ячейку c памяти
22	Пересылка числа со 2-го тумблерного регистра пульта управления в ячейку памяти $* * c\ \tau\ 22$				Число, набранное на втором тумблерном регистре набора числа пульта управления, переносится в ячейку c памяти

Продолжение табл. 15.2

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
23	Деление *a b c* т 23	$c := a : b$	$\lvert c \rvert \geqslant 1$	$c > 2^{88}$ или $b = 0$	Число «*a*» делится на число «*b*». Результат записывается в ячейку *c*
24	Сдвиг по адресу *a b c* т 24	$c_i := \begin{cases} a_{i-n}, \text{ если } 0 \leqslant i-n \leqslant 42_{10}, \\ 0, \text{ если } i-n < 0 \text{ или } \\ \quad i-n > 42_{10} \end{cases}$ $(0 \leqslant i \leqslant 42_{10}$, $n = -p$, если $b = 4000_8 + p$, и $n = +p$, если $b = 0000 + p)$	$c \neq 0$		При $b = 4000_8 + p$ содержимое всех разрядов ячейки *a* сдвигается на *p* разрядов вправо, и на *p* разрядов влево, если $b = 0000 + p$, где $000 \leqslant p \leqslant 777_8$
56	Десятичная печать *a n c* т 56				Производится десятичная печать чисел «*a*», «*a*+1», ..., «*a* + *n*». При этом ячейка *c* очищается

Продолжение табл. 15.2

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
$60 + x$	Предварительная команда групповой операции *а п с* 0 $60 + x$				$n + 1$ раз выполняются операция считывания числа с магнитного барабана, если $0001 \leqslant a \leqslant 0005$, или с магнитной ленты, если $a = 4000_8 + q$ $(0001 \leqslant q \leqslant 2777_8)$, и исполнительной команды. При выполнении операции считывания числа с магнитного барабана каждый раз происходит запись во вторую числовую магистраль числа с зоны a магнитного барабана. При первом выполнении считывается число с порядковым номером c, при втором — $c+1$ и т. д. При выполнении операции считывания числа с магнитной ленты каждый раз происходит запись во вторую магистраль числа с зоны $4000_8 + p$, если $0001 \leqslant p \leqslant 0777_8$, с зоны $5000_8 + p$, если $1000_8 \leqslant p \leqslant 1777_8$, и с зоны $6000_8 + p$, если $2000_8 \leqslant p \leqslant 2777_8$. При первом выполнении операции считывается первое число зоны, при втором — второе и т. д.

Продолжение табл. 15.2

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$	Пояснения
					При выполнении исполнительной команды каждый раз производится запись в первую числовую магистраль числа, находящегося в ячейке, номер которой — первый адрес исполнительной команды; выполняется операция, соответствующая коду операции исполнительной команды, над числами, находящимися в первой и второй числовых магистралях; результат операции записывается в ячейку, номер которой—третий адрес исполнительной команды. Переменные адреса исполнительной команды, определяемые числом x (см. замечание 1), увеличиваются на единицу (x может принимать значения 0, 1, 4, 5).

Продолжение табл. 15.2

Код операции θ	Название операции и вид команды	Краткое содержание команды	Условия выработки ω = 1	Условия выработки φ = 1	Пояснения
64	Запись на магнитный барабан *а п с* 1 64				Команда выполняется машиной совместно со следующей командой, имеющей всегда вид: *а* 0000 0000 0 00. При выполнении команд происходит запись чисел «*а*», «*а* + 1», ..., «*а* + *n*» в зону *а* магнитного барабана со смещением на число *с*, т. е. на зоне *а* магнитного барабана число «*а*» будет иметь порядковый номер *с*, «*а* + 1» — номер *с* + 1 и т. д.

происходит: а) считывание (по первым двум командам) подпрограммы «диспетчер» вместе с константами БСП в ячейки 7613, 7614, ..., 7776 ОЗУ; б) считывание (по следующим трем командам) всей БСП в ячейки 7127, ..., 7776 ОЗУ с помощью подпрограммы «диспетчер», т. е. считывание БСП с контролем.

Таблица 15.3. Подпрограмма считывания БСП

0001 0000	0163 0000	0464 7613	0 0	61 13	Считывание программы «диспетчер» вместе с константами в ячейки 7613, 7614, ..., 7776 без контроля
7761	7761	7777	0	27	Обращение к подпрограмме «диспетчер»
0001 0000	0647 0000	0000 7127	0 0	61 13	Считывание БСП в ячейки памяти 7127, ..., 7776 с помощью подпрограммы «диспетчер» с контролем

В табл. 15.4 указаны все необходимые сведения о стандартных подпрограммах БСП.

Обращение к стандартной подпрограмме с номером N рекомендуется выполнять через подпрограмму «диспетчер» по команде

$$7760+N \quad 7760+N \quad 7777 \quad 0 \quad 27.$$

Обращение к СП1 выполняется в несколько команд, из которых первая есть команда обращения к подпрограмме «диспетчер», т. е. команда 7761 7761 7777 0 27, а последующие — команды обращения к тому или иному запоминающему устройству (команды считывания с МБ или записи на МБ; считывания с МЛ или записи на МЛ; считывания с перфокарт или записи на перфокарты). Обращение к СП2 выполняется в две команды, из которых первая есть команда обращения к подпрограмме «диспетчер», т. е. команда вида

$$7762 \quad 7762 \quad 7777 \quad 0 \quad 27,$$

вторая имеет вид:

$$a \quad n \quad c \quad 0 \quad 20$$

Таблица 15.4. Стандартные программы БСП

Номер СП (восьмеричный)	Содержание СП	Номера ячеек МБ, в которых размещается СП	Номера стандартных ячеек ОЗУ, в которых размещается СП	Номер рабочих ячеек, используемых в СП
1	Обмен информации между различными видами запоминающих устройств с контролем	0465 ÷ 0570	7614 ÷ 7617	0001 ÷ 0013
2	Перевод чисел из десятичной системы счисления в двоичную или обратно	0315 ÷ 0462	7444 ÷ 7613	0001 ÷ 0011
3	Вычисление значения функции $y = \sqrt{x}$	0256 ÷ 0312	7405 ÷ 7443	0001 ÷ 0012
4	Вычисление значения функции $y = \sin x$	0214 ÷ 0254	7343 ÷ 7404	0001 ÷ 0003
5	Вычисление значения функции $y = e^x$	0152 ÷ 0211	7301 ÷ 7342	0001 ÷ 0006
6	Вычисление значений функции $y = \ln x$	0112 ÷ 0147	7241 ÷ 7276	0001 ÷ 0003
7	Вычисление значений функции $y = \operatorname{arctg} x$	0050 ÷ 0111	7177 ÷ 7240	0001 ÷ 0006
10	Вычисление значений функции $y = \arcsin x$	0000 ÷ 0047	7127 ÷ 7176	0001 ÷ 0006

для перевода чисел «a», «$a+1$», ..., «$a+n$» из двоичной системы счисления в десятичную, или

$$a\ n\ c\ 0\ 02$$

— из десятичной системы счисления в двоичную.

Если при составлении рабочей программы программист не использует какие-либо СП, то занимаемые этими СП стандартные ячейки ОЗУ программист может использовать по своему усмотрению.

Пульт управления в машине несколько изменен: нет растра, вместо режима «условный останов» введен режим останова по команде условного или безусловного перехода, т. е. режим «останов по E». В этом режиме (при включенном тумблере «E», который заменил тумблер «условный останов») машина останавливается после изменения содержимого регистра C по команде условного или безусловного перехода.

Вывод на печать выполняется по команде с кодом операции 56. Число печатается следующим образом: знак числа, мантисса числа из 9 цифр, знак порядка из 2 цифр.

Литература к главе: [35], [37].

ГЛАВА 16

ПРОГРАММНО-УПРАВЛЯЕМАЯ МАШИНА *М-20*

Основные характеристики машины *М-20*

Тип машины	вычислительная
Класс машины	средняя
Число основных операций в секунду	20 000
Рабочая система счисления	двоичная
Форма представления чисел	с плавающей запятой
Длина кода числа	всего 45 разрядов, из них: мантисса — 36 разрядов, порядок — 7 разрядов, один разряд — знак числа, один разряд — признак числа.
Диапазон представимых положительных чисел	$2^{-99} \leqslant x \leqslant 2^{63}(1-2^{-36})$, а для нормализованных чисел: $2^{-64} \leqslant x \leqslant 2^{63}(1-2^{-36})$
Точность вычислений	10 ÷ 11 значащих цифр
Порядок выполнения команд	заданный (естественный)
Адресность машины	трехадресная
Виды запоминающих устройств	оперативное запоминающее устройство на ферритовых сердечниках; емкость: 4096 45-разрядных двоичных кодов; время обращения 6 *мксек*; магнитные барабаны: три барабана, каждый емкостью по 4096 45-разрядных двоичных кодов; среднее время ожидания требуемого кода—30 *мсек*, скорость последовательной записи или чтения 6400 кодов в секунду; магнитная лента: емкость 300 000 45-разрядных двоичных кодов; скорость считывания или записи 3000 кодов в секунду.

Виды ввода и вывода	ввод с перфокарт со скоростью 20 45-разрядных кодов в секунду; вывод на печатающее устройство со скоростью 15—20 кодов в секунду и на перфокарты со скоростью 10 кодов в секунду
Количество электронных ламп	3500
Потребляемая мощность	50 *квт*
Комплект машины	электронная цифровая машина на одной стойке, 2 клавишных устройства, 2 перфоратора входных, 2 печатающих устройства (с перфокарт и непосредственно из машины), система питания, система вентиляции
Занимаемая площадь	170—200 $м^2$

Машина *М-20* является универсальной вычислительной машиной. Рекомендуется круглосуточный режим эксплуатации.

Краткие сведения об основных устройствах машины *М-20*

Арифметическое устройство (АУ) — основным блоком АУ является сумматор, разделенный на две части: сумматор чисел, содержащий 36 цифровых разрядов, один дополнительный разряд для округления и 3 знаковых разряда, и сумматор порядков, содержащий 7 цифровых и 2 знаковых разряда. Сумматор чисел работает в обратном коде, сумматор порядков в дополнительном. Сумматор служит для выполнения всех арифметических действий. Имеет цепи сдвига вправо и влево.

В состав АУ входят также два регистра (РІ и РІІ) по 44 разряда. Каждый из этих регистров разделяется на регистр числа, содержащий 36 цифровых разрядов и один дополнительный (младший), и регистр порядка, содержащий 6 цифровых разрядов и один знаковый. РЧІ, РЧІІ, РПІ, РПІІ — обозначения для этих частей регистров.

РІ и РІІ служат для приема кодов из МОЗУ, передачи их на СМ в прямом, обратном (сумматор чисел) и дополнительном (сумматор порядков) кодах. При умножении РЧІІ помещает множимое, при делении — делитель. РІІ не имеет цепей сдвига, РЧІ служит для помещения множителя и младших разрядов произведения при умножении, частного — при делении. РЧІ имеет цепи сдвига вправо и влево. В состав

АУ входит также схема местного управления (МУОП АУ), служащая для управления выравниванием порядков (денормализацией) при сложении, нормализацией, последовательностью сложений и вычитаний при умножении, делении и извлечении корня.

Для повышения скорости работы АУ применяется алгорифм одновременного умножения на два разряда множителя; выполнение сдвига на 2 разряда при умножении и на один разряд при делении совмещено со сложением.

В необходимых случаях АУ вырабатывает сигнал переполнения φ и сигнал управления командой условного перехода — ω. Для выработки ω и φ служит схема, проверяющая наличие нулей во всех разрядах сумматора. Эта же схема служит для упрощения операций с нулевыми исходными данными.

При выполнении операций обмена информацией между ОЗУ и магнитным барабаном, ОЗУ и магнитной лентой, а также при вводе с перфокарт в АУ производится суммирование передаваемых кодов для целей контроля.

Оперативное запоминающее устройство выполнено на ферритовых сердечниках, время обращения для одного сорокапятиразрядного кода — 6 *мксек*. Имеет 4096 45-разрядных ячеек.

Ячейки имеют номера от 0000 до 7777 в восьмеричной системе счисления. При чтении из ячейки с номером 0000 всегда выдается машинный нуль — код, содержащий 0 во всех разрядах.

Устройство управления содержит основные логические элементы:

регистр номера команды (КРА) — *C*; имеет 12 разрядов;

регистр команд (РК), на который записывается выдаваемая из ОЗУ команда; имеет 45 разрядов;

регистр изменения адреса (РА) — δ; имеет 12 разрядов;

сумматор адреса (СМА), работающий в прямом коде, имеет 12 разрядов.

К устройству управления можно отнести также *регистр результата* (РР), через этот регистр передаются коды, полученные в АУ для записи в ОЗУ.

Магнитные запоминающие устройства (МЗУ) состоят из трех магнитных барабанов и четырех лентопротяжных блоков магнитных лент.

Магнитные барабаны (МБ). Устройство содержит три магнитных барабана емкостью по 4096 45-разрядных кодов каждый. Общая емкость 12 288 кодов. Каждый из барабанов представляет собой цилиндр с ферромагнитным покрытием, вращающийся со скоростью 1500 оборотов в минуту. Вдоль образующей цилиндра установлено 66 магнитных головок, служащих для записи и чтения кодов (64 головки), для чтения адресов кодов (1 головка) и для чтения синхронизирующих импульсов (1 головка). По окружности барабана располагается 256 кодов. Имеется 8 трактов (по 8 головок на каждый тракт), на каждом из которых размещается соответственно по 512 кодов. Запись кодов и передача их из МОЗУ в МБ и обратно—параллельно-последовательная. Передача кодов производится через АУ, в котором с помощью РІ производится преобразование из параллельно-последовательной формы в параллельную и обратно. Способ чтения и записи групповой с указанием номера первого кода в группе.

Магнитные ленты (МЛ). Машина имеет 4 лентопротяжных блока, или, как их еще иначе называют, — 4 магнитофона емкостью по 75 000 45-разрядных кодов каждый. Общая емкость — 300 000 45-разрядных кодов. Считывание и запись производятся с помощью 9 головок: 8 из них для записи и чтения кодов, одна — для чтения синхронизирующих импульсов. Запись кода параллельно-последовательная. При обмене кодами между МОЗУ и магнитной лентой, АУ является буферным и преобразующим устройством. Скорость движения ленты — 2 метра в секунду. Лента может двигаться как в одну, так и в другую сторону. Способ записи и чтения групповой — по адресам зон.

Устройства ввода и вывода. В качестве устройства ввода в машине применяется читающее устройство, работающее на перфокартах (20 кодов в секунду). Выдача производится на перфоратор (10 кодов в секунду) или на печатающее устройство (15 кодов в секунду).

Для обеспечения большой скорости вывода применяется буферное устройство — дополнительная секция одного из магнитных барабанов емкостью 511 45-разрядных кодов. По команде выводимый материал передается на барабан, откуда коды поступают на выходное устройство. Машина может продолжать вычисления. Задержка в решении возникает только в том случае, когда при поступлении новой команды

выдачи еще не все коды, расположенные на барабане, переданы на печать или на перфоратор. Указанные 511 кодов размещаются в 256 зонах (по 2 кода в зоне). Запись на буфер может производиться только с начала некоторой зоны. При записи происходит поразрядное логическое сложение старого содержимого ячейки с записываемым кодом. Поэтому перед записью нужные ячейки буфера необходимо очистить. Выдача более 511 кодов производится обычно по специальной стандартной программе.

Работа машины слагается из выполнения элементарных тактов, каждый из которых в свою очередь состоит из стандартного такта, длительностью 24 *мксек*, и временной задержки, определяемой логикой выполняемых операций. За время одного стандартного такта задающий генератор УУ выдает 16 импульсов с интервалом 1,5 *мксек*. Эти импульсы являются основными в цикле и для удобства им присваивают порядковые номера с 1 до 16.

Рассматривая выполнение операций над числами, над командами и кодами (табл. 16. 1; 16. 2; 16. 3), предположим, что на РК установлена некоторая команда, на КРА — ее номер. На регистре переадресации (РА) также записано некоторое число. В начале цикла (с 1-го по 3-й импульсы) на сумматор адреса СМА последовательно передаются: 1) содержимое РА (в том случае, если признак переадресации 1-го адреса $\tau_1 = 1$), 2) содержимое 1-го адреса РК и происходит суммирование. Так образуется исполнительный адрес. По этому адресу происходит считывание 1-го числа из ОЗУ.

Далее, за время с 5-го по 7-й импульсы на СМА аналогичным образом формируется исполнительный адрес, служащий для считывания 2-го числа, и передается в ОЗУ. (Содержимое РА поступает на СМА, если $\tau_2 = 1$.) Предыдущее чтение к этому времени закончилось и 1-е число записано на РI АУ. После выборки 2-го числа и передачи его на РII АУ (а при выполнении некоторых операций после выдачи 1-го числа) начинается выполнение операции в АУ. Запись результата по 3-му адресу можно будет производить только после окончания работы АУ. Имеющийся промежуток времени используется для: 1) формирования номера следующей команды, 2) считывания этой команды, 3) формирования исполнительного адреса записи результата. Для этого за промежуток времени между 7-м и 9-м импульсами на СМА поступает адрес из КРА, его значение увеличивается на единицу, а затем результат посылается обратно на КРА. Таким образом, на КРА попадает номер следующей команды. Между 11-м и 13-м импульсами на СМА образуется и хранится в дальнейшем исполнительный адрес записи результата. Таким образом, все три адреса, указанные в исполняемой команде, сформированы, и на РК может быть записана следующая команда. (Разряды кода операции исполняемой команды были предварительно переписаны на РОП.) Для чтения следующей команды, в момент прихода 13-го импульса, в МОЗУ передается

ее номер непосредственно из КРА, и новая команда принимается на РК. После окончания арифметической операции в АУ, в момент прихода 16-го импульса, происходит запись результата по адресу, переданному из СМА в ОЗУ.

Некоторые операции (их время выполнения превышает 24 *мксек*) не успевают выдать свой результат до прихода последнего импульса стандартного такта. При выполнении этих операций в необходимых местах такта прекращается выдача основных импульсов и включается МУОП АУ, с помощью которого производится задержка выдачи главных импульсов и выдается серия сигналов, используемых в схемах АУ. В частности, для операций, выполняющих выравнивание порядков, нормализацию результата и для операции сдвига, МУОП через каждые 1,5 *мксек* выдает импульс, управляющий сдвигом на 1 разряд. МУОП прекращает свою работу по окончании всех сдвигов. Этим объясняется то, что время указанных операций может колебаться в зависимости от исходных данных.

После того как операция закончена, продолжается выдача главных импульсов и происходит запись результата.

При выполнении элементарного такта кроме указанных действий происходит сравнение исполнительного третьего адреса и адреса следующей команды. При совпадении этих адресов на РК сбрасывается ранее считанная из ОЗУ команда и принимается код, посылаемый в ОЗУ во время записи. Этим обеспечивается верная работа машины в том случае, когда результат выполнения операции является следующей командой.

При выполнении операций передачи управления за время между 7-м и 9-м импульсами вместо увеличения содержимого КРА на 1 с помощью СМА производится передача содержимого 2-го адреса регистра команд на СМА. Это выполняется для команд безусловного перехода ($\theta = 56$; 16 см. табл. 16.4) всегда, а для команд условного перехода ($\theta = 36, 76$) и переходов после цикла ($\theta = 12, 32, 11, 31, 51, 71$) — при удовлетворении соответствующих условий. В том случае, если условия не удовлетворены, производится, как и раньше, увеличение содержимого КРА на 1. При работе этих операций АУ либо выполняет операцию передачи кода, либо формирует команду возврата, либо сравнивает 1-й адрес команды, стоящей на РК, с содержимым регистра РА для проверки условия (переход после циклов). В последних двух случаях используются специальные цепи, соединяющие РК и АУ.

В машине выполняются операции записи кода на РА с переносом прежнего содержимого его в ячейку ОЗУ; ($\theta = 52, 72$). В этом случае в момент прихода 4-го импульса содержимое СМА, полученное, как описано выше, передается на сумматор АУ, где окончательно формируется команда, которая будет в дальнейшем записана в ОЗУ. Значение второго адреса, полученное за время с 5-го по 7-й импульс, с приходом 7-го импульса передается на РI АУ, где хранится до тех пор, пока не будет сформирован 3-й адрес и только с приходом 14-го импульса передается на РА. Таким образом, все три адреса (при наличии соответствующих значений признаков переадресации) изменяются на старое содержимое РА и только после этого на РА принимается новое число.

При выполнении группового обмена информацией между ОЗУ и МБ или ОЗУ и МЛ, а также при вводе производится синхронизация работы машины от соответствующих электромеханических устройств. После обработки каждого из передаваемых кодов на данном устройстве (запись на МЛ, МБ, прием из устройства ввода; или считывание из МЛ, МБ) в УУ поступает импульс, который запускает машину на выполнение одного элементарного такта, во время которого производится преобразование этого кода (при считывании) или следующего (при выдаче из ОЗУ) кода из последовательно-параллельной формы в параллельную или обратно, а также запись в ОЗУ или чтение из него. Кроме этого, существуют режимы поиска зоны МЛ или начального адреса МБ, при выполнении которых с приходом каждого нового адреса АУ производит сравнение его с требуемым и вырабатывает соответствующие сигналы.

Представление чисел и команд

Распределение разрядов ячейки ОЗУ при хранении в ней 45-разрядного двоичного числа $x = x_M \cdot 2^{x_П - 64}$ показано на рис. 16.1, *а*. Число представляется с плавающей запятой, мантисса записывается в ячейке в прямом коде, а фактический порядок x_P связан с порядком в ячейке $x_П$ соотношением

$$x_P = x_П - 64 = \begin{cases} 1 \text{ в 43-м разряде и } |x_P| \text{ в } 37 \div 42\text{-м} \\ \text{разрядах при } x_P \geqslant 0; \\ 0 \text{ в 43-м разряде и } 2^6 - |x_P| \text{ в } 37 \div 42\text{-м} \\ \text{разрядах при } x_P < 0. \end{cases}$$

Машинный нуль представляется кодом, имеющим нули во всех разрядах.

Распределение разрядов ячейки ОЗУ при хранении в ней двоично-десятичного числа показано на рис. 16.1, *б*. Десятичное число представляется с плавающей запятой, порядок и мантисса представляются в прямом коде.

В знаковых разрядах двоичного и двоично-десятичного чисел 0 соответствует знаку плюс; 1 — знаку минус.

Распределение разрядов ячейки ОЗУ при хранении в ней команды показано на рис. 16.1, *в*. Структура кода команды имеет следующий вид:

$$\tau_1 \tau_2 \tau_3 \quad \theta \, a \, b \, c,$$

где τ_1, τ_2, τ_3 — признаки изменения адресов; θ — код операции; *a*, *b*, *c* — первый, второй и третий адреса команды. Весь код команды занимает 45 двоичных разрядов.

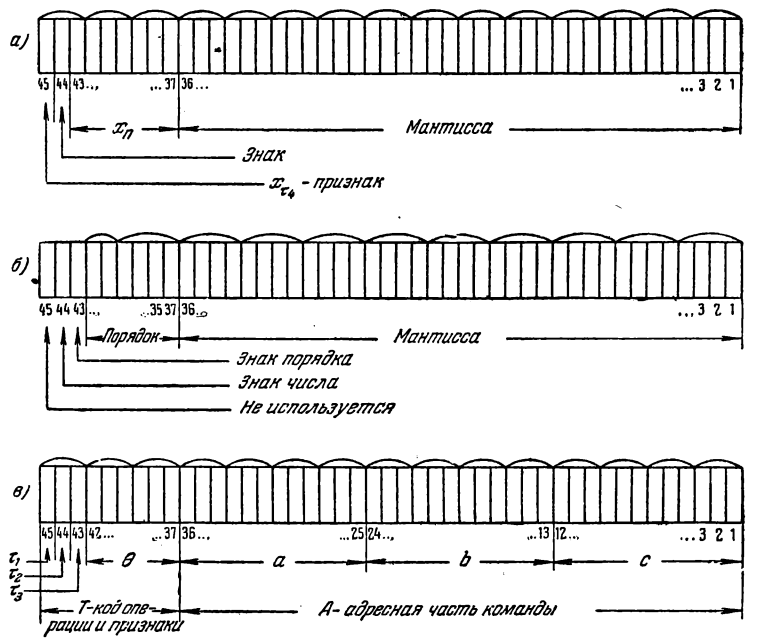


Рис. 16.1. Распределение разрядов ячейки ОЗУ: *а*) для хранения двоичного числа; *б*) для хранения двоично-десятичного числа; *в*) для хранения команды.

На рис. 16.1 скобками ⌒ указаны объединения двоичных разрядов в восьмеричные и десятичные цифры, применяемые при записи чисел и команд.

Система операций и команд

При выполнении различных групп операций 45-разрядные коды, представляемые в машине, рассматриваются по-разному.

В соответствии с этим система операций и команд помещена в пяти таблицах (16.1; 16.2; 16.3; 16.4; 16.5).

Операции над числами (табл. 16.1) служат для выполнения вычислений, иногда применяются для преобразования кодов. В операциях над числами 45-разрядный двоичный код рассматривается как двоичное число в соответствии с рис. 16.1, *а*. При выполнении любой из операций этой группы признак числа «$c_{\tau_ч}$» результата «c» равен логической сумме признаков числа «$a_{\tau_ч}$»; «$b_{\tau_ч}$», участвующих в операции чисел «a» и «b»:

$$c_{\tau_ч} := a_{\tau_ч} \vee b_{\tau_ч}.$$

Операции над кодами (табл. 16.2) применяются для преобразования различных кодов. Для большинства этих операций рассматриваются 45-разрядные коды без какого-либо деления на мантиссу, порядок, адреса, код операции или какие-либо признаки. Все разряды считаются равноценными. Для некоторых операций ($\theta = 07, 27$) код, хранящийся в ячейке x, удобно разбивать на два кода «x_T» и «x_A» (см. рис. 16.1, *в*), характеризуемые целыми положительными двоичными числами:

$$«x_A» = \sum_{i=1}^{36} «x_i» \cdot 2^{i-1}; \qquad «x_T» = \sum_{i=37}^{45} «x_i» \cdot 2^{i-37}. \qquad (1)$$

Здесь «x_i» — содержимое двоичного разряда с номером i ячейки x.

При выполнении операций этой группы $\varphi = 0$.

Операции над кодами команд (табл. 16.3) чаще всего применяются для преобразования команд. При этих операциях 45-разрядный код рассматривается как команда, состоящая из трех признаков, кода операции и адресной части согласно рисунку 16, 1 *в*. При описании операций, входящих в эту группу, удобно пользоваться формулами (1).

Операции управления (табл. 16.4) применяются для изменения последовательности выборки команд, в частности для организации циклов. Группа может быть разделена на две подгруппы:

а) операции передачи управления;

б) операции, организующие выполнение циклов (запись кода на регистр РА или изменение содержимого этого регистра).

Таблица 16.1. Команды, выполняющие операции над числами

Наименование команды, ее вид и пояснения, общие для всех модификаций	Код операции (значения θ)	Модификация команды	Время выполнения, *мксек*	Условия выработки ω = 1	Условия выработки φ = 1
Сложение $\tau_1\tau_2\tau_3\ \theta\ a\ b\ c$ Число, хранящееся в ячейке $\tilde{a}$, складывается с числом, хранящимся в ячейке $\tilde{b}$, и результат записывается в ячейку $\tilde{c}$: $\tilde{c} := \tilde{a} + \tilde{b}.$ «$\tilde{a}$» и «$\tilde{b}$» могут быть ненормализованными. Предварительная нормализация их не производится. Сдвиг и суммирование не производятся, если мантисса одного из чисел равна «+ 0». При этом мантисса ненулевого слагаемого есть результат суммирования. Если перед записью результата в ячейку $\tilde{c}$ оказывается, что его мантисса равна нулю, или что $\tilde{c}_P < -64,$ то из АУ выдается машинный нуль	01	С округлением и нормализацией. Во время суммирования мантисс производится округление по 37-му (дополнительному) разряду суммы. После суммирования выполняется нормализация результата. При нормализации вправо производится доокругление результата по младшему разряду, сдвигаемому за пределы мантиссы. При нормализации влево дополнительный (37-й) разряд попадает в мантиссу	от 25,5 до 82,5, точнее, $25{,}5+1{,}5l$	$\tilde{c} < 0$	$\tilde{c}_P > 63$
	21	Без округления и с нормализацией. Округления и доокругления нет, нормализация выполняется как при θ = 01			
	41	С округлением и без нормализации. Нормализация влево не производится, округление и доокругление производится так же, как при θ = 01			
	61	Без округления и без нормализации. Округление и нормализация влево не производится			

Продолжение табл. 16.1

Наименование команды, ее вид и пояснения, общие для всех модификаций	Код операции (значения θ)	Модификация команды	Время выполнения, *мксек*	Условия выработки $\omega = 1$	Условия выработки $\varphi = 1$
Вычитание $\tau_1\tau_2\tau_3\ \theta\ a\ b\ c$ Операция отличается от операции «сложение» тем, что знак числа, поступающего из ячейки b, предварительно изменяется на обратный $\tilde{c} := \tilde{a} + \overline{\operatorname{sign} \tilde{b}}\,|\tilde{b}|$	02 22 42 62	Коды операций обозначают те же модификации, что и для операции «сложение» соответственно	То же, что и для сложения	$\tilde{c} < 0$	$\tilde{c}_P > 63$
Вычитание модулей $\tau_1\tau_2\tau_3\ \theta\ a\ b\ c$ Операция отличается от операции «сложение» тем, что предварительно числу, выбираемому из ячейки $\tilde{a}$, присваивается знак «+», а числу, выбираемому из ячейки $\tilde{b}$, — знак «—» $\tilde{c} := \|\tilde{a}\| - \|\tilde{b}\|$	03 23 43 63	Коды операций обозначают те же модификации, что и для операции «сложение» соответственно	То же, что и для сложения	$\tilde{c} < 0$	

Продолжение табл. 16.1

Наименование команды, ее вид и пояснения, общие для всех модификаций	Код операции (значения θ)	Модификация команды	Время выполнения, *мксек*	Условия выработки ω = 1	Условия выработки φ = 1
Умножение $\tau_1\tau_2\tau_3\ \theta\ a\ b\ c$ Число, хранящееся в ячейке $\tilde{a}$ (множитель), умножается на число, хранящееся в ячейке $\tilde{b}$ (множимое). Результат (36 старших разрядов мантиссы произведения, порядок и знак) записывается в ячейку $\tilde{c}$ $\tilde{c} := \tilde{a} \cdot \tilde{b}.$ «$\tilde{a}$» и «$\tilde{b}$» могут быть ненормализованными. Предварительная нормализация их не производится. Если либо $\tilde{a}_M = 0$, либо $\tilde{b} = 0$, либо $\tilde{c}_P < -64$, то выдается машинный нуль	05 25 45 65	Коды операции обозначают соответственно те же модификации, что и для операции «сложение». Так же, как и при сложении, округление и нормализация производятся после образования мантиссы результата на сумматоре числа. Мантисса может быть: 1) нормализованной или ненормализованной вправо на 1 разряд (при нормализованных «$\tilde{a}$» и «$\tilde{b}$»); 2) ненормализованной влево на 1 или более разрядов (для ненормализованных «$\tilde{a}$» и «$\tilde{b}$»). Производится округление 37-го разряда полного произведения по 38-му. Это дает верный результат при наличии нормализации. При отсутствии нормализации округляется 36-й разряд по 37-му.	От 49,5 до 76,5	$\tilde{c}_P > 0$	$\tilde{c}_P > 63$

Продолжение табл. 16.1

Наименование команды, ее вид и пояснения, общие для всех модификаций	Код операции (значения θ)	Модификация команды	Время выполнения, *мксек*	Условия выработки ω = 1	Условия выработки φ = 1
Вывод младших разрядов произведения $\tau_1\tau_2\tau_3$ θ *a b c* В ячейку $\tilde{c}$ записывается код, образованный путем объединения содержимого Р1 (мантисса кода) и сумматора порядков (порядок его), оставшихся от предыдущей операции. Знак кода равен знаку результата предыдущей операции. Если предыдущей операцией была операция умножения, то полученный код представляет собой младшие разряды произведения с его порядком и знаком	47	Модификаций нет. Округления нет. Так как Р1 имеет 36 разрядов, нормализация не производится	24	$\tilde{c} = 0$	

Продолжение табл. 16.1

Наименование команды, ее вид и пояснения, общие для всех модификаций	Код операции (значения θ)	Модификация команды	Время выполнения, *мксек*	Условия выработки ω = 1	Условия выработки φ = 1
Деление $\tau_1\tau_2\tau_3\ \theta\ a\ b\ c$ Число, стоящее в ячейке $\tilde{a}$, делится на число, стоящее в ячейке $\tilde{b}$. Частное записывается в ячейку $\tilde{c}$ $\tilde{c} := \tilde{a} : \tilde{b}$. «$\tilde{a}$» и «$\tilde{b}$» могут быть ненормализованными, предварительная нормализация их не производится. При этом частное может быть ненормализованным вправо на любое количество разрядов. Нормализация влево не производится. Если мантисса частного $(\tilde{a} : \tilde{b})_M$ лежит в пределах $1 \leqslant (\tilde{a} : \tilde{b})_M \leqslant 2$, то производится нормализация вправо на 1 разряд. Если $(\tilde{a} : \tilde{b})_M = 0$, то $c := 0$	04	С округлением. Округление производится после нормализации. Поэтому в результате 36-й разряд мантиссы округляется по 37-му, который специально определяется для целей округления	136,5	$\tilde{c}_P > 0$	$\tilde{c}_P > 63$ $\tilde{c}_M \geqslant 2$ Последнее возможно при ненормализованных «$\tilde{a}$» и «$\tilde{b}$»
	24	Без округления. Как и при θ = 04, определяется 37 разрядов частного, но округление не производится			

Продолжение табл. 16.1

Наименование команды, ее вид и пояснения, общие для всех модификаций	Код операции (значения θ)	Модификация команды	Время выполнения, *мксек*	Условия выработки ω = 1	Условия выработки φ = 1
Извлечение квадратного корня $\tau_1\tau_2\tau_3\ \theta\ a\ *\cdot c$ Из числа, стоящего в ячейке $\tilde{a}$, извлекается квадратный корень, результат записывается в ячейку $\tilde{c}$ $\tilde{c} := \sqrt{\tilde{a}}$. При ненормализованном «a» результат может быть ненормализованным. Нормализация вправо не производится, так как $\tilde{a}_M < 1$, а следовательно и $\tilde{c}_M < 1$. Если $\tilde{a}_M = 0$, или $\tilde{c}_P < -64$, то $\tilde{c} = 0$	44	С округлением. 36-й разряд мантиссы результата округляется по 37-му разряду, который определяется специально для этой цели	От 197 до 654, среднее 275	$\tilde{c}_P > 0$	$\tilde{a} < 0$
	64	Без округления. Как и при θ = 44 определяется 37 разрядов мантиссы результата, но округление не производится			

Продолжение табл. 16.1

Наименование команды, ее вид и пояснения, общие для всех модификаций	Код операции (значения θ)	Модификация команды	Время выполнения, *мксек*	Условия выработки ω = 1	Условия выработки φ = 1
Сложение порядка с адресом $\tau_1\tau_2\tau_3$ 06 * *n b c* Здесь $\tilde{n}$ — семь младших разрядов адреса, * занимает пять его старших разрядов. В ячейку $\tilde{c}$ записывается число, мантисса и знак которого равны знаку и мантиссе числа «$\tilde{b}$», а порядок получен путем прибавления n к порядку числа «$\tilde{b}$» $\tilde{c}_{14} := \tilde{b}_{44}$, $\tilde{c}_M := \tilde{b}_M$, $\tilde{c}_P := \tilde{b}_P + k_n$. Если $k_n < 0$, то оно должно быть представлено так, как представляются отрицательные порядки чисел	06	Модификации нет. Нормализация мантиссы результата, возможная в случае, когда «$\tilde{b}$» ненормализовано, не производится	61,5	$\tilde{c}_P > 0$	$\tilde{c}_P > 63$

Продолжение табл. 16.1

Наименование команды, ее вид и пояснения, общие для всех модификаций	Код операции (значения θ)	Модификация команды	Время выполнения, *мксек*	Условия выработки ω = 1	Условия выработки φ = 1
Сложение порядков $\tau_1\tau_2\tau_3$ 26 *a b c* Отличается от предыдущей операции тем, что для изменения порядка используется порядок числа, стоящего в ячейке «$\tilde{a}$» $\tilde{c}_{44} := \tilde{b}_{44}$, $\tilde{c}_M := \tilde{b}_M$, $\tilde{c}_P := \tilde{b}_P + \tilde{a}_P$	26	То же, что и для θ = 06	24	$\tilde{c} > 0$	$\tilde{c}_P > 63$
Вычитание из порядка адреса $\tau_1\tau_2\tau_3$ 46 **n b c* Отличается от операции θ = 06 тем, что *n* вычитается из порядка числа «$\tilde{b}$» $\tilde{c}_{44} := \tilde{b}_{44}$, $\tilde{c}_M := \tilde{b}_M$, $\tilde{c}_P := \tilde{b}_P + \overline{\operatorname{sign} k_n}\,\lvert k_n \rvert$	46	То же, что и для θ = 06	61,5	$\tilde{c} > 0$	$\tilde{c}_P > 63$

Продолжение табл. 16.1

Наименование команды, ее вид и пояснения, общие для всех модификаций	Код операции (значения θ)	Модификация команды	Время выполнения, *мксек*	Условия выработки ω = 1	Условия выработки φ = 1
Вычитание порядков $\tau_1\tau_2\tau_3$ 66 *a b c* Значение 43-го разряда кода «$\tilde{a}$» при передаче в АУ изменяется на обратное, затем выполняется операция θ = 26 $\tilde{c}_{44} := \tilde{b}_{44}$ $\tilde{c}_M := \tilde{b}_M$ $\tilde{c}_P := \tilde{b}_P + \overline{\operatorname{sign} \tilde{a}_P} \lvert \tilde{a}_P \rvert$	66	То же, что и для θ = 06	24	$\tilde{c} > 0$	$\tilde{c}_P > 63$

Таблица 16.2. Команды, выполняющие операции над кодами

Наименование команды, ее вид. Пояснения к адресной части команды	Код операции (значения θ)	Содержание операции	Время выполнения, *мксек*	Условия выработки ω = 1
Сдвиг мантиссы по адресу $\tau_1\tau_2\tau_3$ 14 $*n\ b\ c$ Здесь $\tilde{n}$ (младшие 7 разрядов адреса) — величина сдвига. В старшем разряде $\tilde{n}$ располагается его знак. * занимает остальные пять разрядов адреса	14	Мантисса и знак кода, хранящегося в ячейке $\tilde{b}$, сдвигаются на $\tilde{n}$ разрядов влево, если $k_n > 0$ и вправо, если $k_n < 0$. $\tilde{n}$ воспринимается машиной в той же форме, что и порядки чисел. Результат записывается в ячейку $\tilde{c}$. Содержимое 37 ÷ 45-го разрядов ячейки $\tilde{c}$ (порядок) равно содержимому соответствующих разрядов ячейки $\tilde{b}$. $\tilde{c}_i := \begin{cases} \tilde{b}_{i-k_n} \text{ при } 1 \leqslant i - k_n \leqslant 36, \\ 0 \text{ при } i - k_n > 36;\ i - k_n < 1, \\ \tilde{b}_i \text{ при } 37 \leqslant i \leqslant 45, \end{cases}$ $1 \leqslant i \leqslant 45$	$61{,}5 + 1{,}5\,\lvert k_n \rvert$	$\tilde{c}_i = 0$ $1 \leqslant i \leqslant 36$
Сдвиг мантиссы по порядку $\tau_1\tau_2\tau_3$ 34 $a\ b\ c$	34	Отличается от предыдущей операции тем, что сдвиг кода «$\tilde{b}$» производится по порядку кода, хранящегося в ячейке $\tilde{a}$. $\tilde{c}_i := \begin{cases} \tilde{b}_{i-\tilde{a}_P} \text{ при } 1 \leqslant i - \tilde{a}_P \leqslant 36, \\ 0 \text{ при } i - \tilde{a}_P > 36;\ i - \tilde{a}_P < 1, \\ \tilde{b}_i \text{ при } 37 \leqslant i \leqslant 45, \end{cases}$ $1 \leqslant i \leqslant 45$	$24 + 1{,}5\,\lvert a_P \rvert$	$\tilde{c}_i = 0$ $1 \leqslant i \leqslant 36$

Продолжение табл. 16.2

Наименование команды, ее вид. Пояснения к адресной части команды	Код операции (значения θ)	Содержание операции	Время выполнения, *мксек*	Условия выработки ω = 1
Сдвиг кода по адресу $\tau_1\tau_2\tau_3$ 54 $*n\ b\ c$ Здесь n имеет то же значение, что и в команде θ=14	54	Код, хранящийся в ячейке $\tilde{b}$, сдвигается на $\tilde{n}$ разрядов влево, если $k_n > 0$, и вправо, если $k_n < 0$. Результат записывается в ячейку $\tilde{c}$. $\tilde{c}_i := \begin{cases} \tilde{b}_{i-k_n} \text{ при } 1 \leqslant i - k_n \leqslant 45, \\ 0 \quad \text{при } i - k_n < 1; i - k_n > 45, \end{cases}$ $1 \leqslant i \leqslant 45$	от 61,5 до 130, точнее, $61{,}5 + 1{,}5\,\lvert k_n \rvert$	$\tilde{c}_i = 0$ $1 \leqslant i \leqslant 45$
Сдвиг кода по порядку $\tau_1\tau_2\tau_3$ 74 $a\ b\ c$	74	Отличается от предыдущей операции тем, что сдвиг кода «$\tilde{b}$» производится по порядку кода, хранящегося в ячейке $\tilde{a}$. $\tilde{c}_i := \begin{cases} \tilde{b}_{i-\tilde{a}_P} \text{ при } 1 \leqslant i - \tilde{a}_P \leqslant 45, \\ 0 \quad \text{при } i - \tilde{a}_P < 1; i - \tilde{a}_P > 45 \end{cases}$	От 24 до 93, точнее, $24 + 1{,}5\,\lvert \tilde{a}_P \rvert$	$\tilde{c}_i = 0$ $1 \leqslant i \leqslant 45$

Продолжение табл. 16.2

Наименование команды, ее вид. Пояснения к адресной части команды	Код операции (значения θ)	Содержание операции	Время выполнения *мксек*	Условия выработки ω = 1
Сравнение $\tau_1\tau_2\tau_3$ 15 *a b c*	15	Производится поразрядная операция отрицания равнозначности между содержимым ячеек $\tilde{a}$ и $\tilde{b}$. Результат записывается в ячейку $\tilde{c}$. $\tilde{c}_i := \tilde{a}_i \not\sim \tilde{b}_i,$ $1 \leqslant i \leqslant 45$	24	$\tilde{c}_i = 0$ $1 \leqslant i \leqslant 45$
Сравнение с остановом $\tau_1\tau_2\tau_3$ 35 *a b c*	35	Отличается от операции «сравнение» (θ = 15) тем, что при ω = 0 (коды не совпадают между собой) машина останавливается	24	$\tilde{c}_i = 0$ $1 \leqslant i \leqslant 45$
Логическое сложение $\tau_1\tau_2\tau_3$ 75 *a b c*	75	Код, хранящийся в ячейке $\tilde{b}$, поразрядно логически складывается с кодом, хранящимся в ячейке $\tilde{a}$. Результат записывается в ячейку $\tilde{c}$: $\tilde{c}_i := \tilde{b}_i \vee \tilde{a}_i,$ $1 \leqslant i \leqslant 45$	24	$\tilde{c}_i = 0$ $1 \leqslant i \leqslant 45$

Продолжение табл. 16.2

Наименование команды, ее вид. Пояснения к адресной части команды	Код операции (значения 0)	Содержание операции	Время выполнения, *мксек*	Условия выработки $\omega = 1$
Логическое умножение $\tau_1\tau_2\tau_3$ 55 *a b c*	55	Код, хранящийся в ячейке $\tilde{b}$ поразрядно логически умножается на код, хранящийся в ячейке $\tilde{a}$. Результат записывается в ячейку $\tilde{c}$: $\tilde{c}_i := \tilde{b}_i \wedge \tilde{a}_i,$ $1 \leqslant i \leqslant 45$	24	$\tilde{c}_i = 0$ $1 \leqslant i \leqslant 45$
Суммирование для контроля (циклическое сложение) $\tau_1\tau_2\tau_3$ 07 *a b c*	07	Производится суммирование с круговым переносом кодов, стоящих в ячейках $\tilde{a}$ и $\tilde{b}$. Результат записывается в ячейку $\tilde{c}$. Имеется две цепи кругового переноса: 1) из 45-го разряда в 37-й, 2) из 36-го в 1-й. Переноса из 36-го в 37-й разряд нет. Таким образом, одновременно и независимо проходят две операции сложения с круговым переносом над разными частями кодов. $\tilde{c}_T := \tilde{a}_T \oplus \tilde{b}_T,$ $\tilde{c}_A := \tilde{a}_A \oplus \tilde{b}_A$	24	Если возникает круговой перенос из 36-го разряда в 1-й, $\tilde{a}_A + \tilde{b}_A > 2^{36} - 1$

Продолжение табл. 16.2

Наименование команды, ее вид. Пояснения к адресной части команды	Код операции (значения θ)	Содержание операции	Время выполнения, *мксек*	Условия выработки ω = 1
Циклическое вычитание $\tau_1\tau_2\tau_3$ 27 *a b c*	27	Отличается от предыдущей операции тем, что в циклическом сложении принимает участие поразрядная инверсия кода «$\tilde{b}$». $\tilde{c}_T := \tilde{a}_T \oplus [\bar{\tilde{b}}_T]$, $\tilde{c}_A := \tilde{a}_A \oplus [\bar{\tilde{b}}_A]$	24	Если возникает круговой перенос из 36-го разряда в 1-й, $\tilde{a}_A + [\bar{\tilde{b}}_A] > 2^{36} - 1$
Передача кода $\tau_1\tau_2\tau_3$ 00 *a* * *c*	00	Код, хранящийся в ячейке $\tilde{a}$, без преобразования передается в ячейку $\tilde{c}$	24	
Передача кода из регистров пульта управления $\tau_1\tau_2\tau_3$ 20 **n b c* *n* занимает четыре разряда адреса; три младших разряда адреса указывают номер одного из четырех регистров пульта управления. Используются комбинации 001; 010; 011 и 100. Единица в четвертом разряде соответствует регистру результата	20	Код, набранный на соответствующем регистре пульта управления, переносится без преобразования в ячейку $\tilde{c}$. При $\tilde{n}$ равном 0101; 0110; 0111, в ячейку $\tilde{c}$ записывается логическая сумма кодов, набранных на двух регистрах. Такая же логическая сумма образуется и в том случае, когда при наличии 1 в 4-м разряде $\tilde{n}$ имеется отличная от нуля комбинация в трех младших разрядах	24	

Продолжение табл. 16.2

Наименование команды, ее вид. Пояснения к адресной части команды	Код операции (значения θ)	Содержание операции	Время выполнения, *мксек*	Условия выработки ω = 1
Гашение $\tau_1\tau_2\tau_3\ \theta\ *\ *\ c$	40 60	В ячейку $\tilde{c}$ записывается машинный нуль: $\tilde{c}_i := 0,$ $1 \leqslant i \leqslant 45$	24	
Циклический сдвиг $\tau_1\tau_2\tau_3\ 67\ *\ b\ c$	67	Производится сдвиг кода, хранящегося в ячейке $\tilde{b}$, влево на 24 разряда. Сдвиг осуществляется на 48-разрядном сумматоре так, что 49-й разряд при каждом сдвиге на один разряд попадает в 1-й разряд результата. Результат записывается в ячейку $\tilde{c}$: $\tilde{c}_i := \begin{cases} \tilde{b}_{i-24} & \text{при } 25 \leqslant i \leqslant 45, \\ \tilde{b}_{i+24} & \text{при } 1 \leqslant i \leqslant 21, \\ 0 & \text{при } 21 \leqslant i \leqslant 24 \end{cases}$	24	

Таблица 16.3. Команды, выполняющие операции над кодами команд

Наименование команды, ее вид	Код операции θ	Результат операции и пояснения	Время выполнения, *мксек*	Условия выработки $\omega = 1$
Сложение адресной части $\tau_1\tau_2\tau_3$ 13 *a b c*	13	В ячейку $\tilde{c}$ записывается код, у которого адресная часть равна сумме адресных частей кодов «$\tilde{a}$» и «$\tilde{b}$», а разряды кода операции и признаков равны соответствующим разрядам кода «$\tilde{a}$». Единица переноса из 36-го разряда отбрасывается $\tilde{c}_T := \tilde{a}_T,$ $\tilde{c}_A := \tilde{a}_A \boxplus_{36} \tilde{b}_A$	24	При возникновении 1-го переноса из 36-го разряда, то есть: $\tilde{a}_A + \tilde{b}_A \geqslant 2^{36}$
Вычитание адресной части $\tau_1\tau_2\tau_3$ 33 *a b c*	33	Отличается от предыдущей операции тем, что из адресной части кода «$\tilde{a}$» вычитается адресная часть кода «$\tilde{b}$». (В сложении принимает участие обратный код 1÷36-го разрядов «$\tilde{b}$»; используется круговой перенос из 36-го разряда в 1-й) $\tilde{c}_T := \tilde{a}_T,$ $\tilde{c}_A := \tilde{a}_A \oplus [\bar{\tilde{b}}_A]$	24	При возникновении 1-го переноса из 36-го разряда, то есть: $\tilde{a}_A + [\bar{\tilde{b}}_A] \geqslant 2^{36}$

Продолжение табл. 16.3

Наименование команды, ее вид	Код операции θ	Результат операции и пояснения	Время выполнения, *мксек*	Условия выработки $\omega = 1$
Сложение кодов операции $\tau_1\tau_2\tau_3$ 53 *a b c*	53	В ячейку $\tilde{c}$ записывается код, у которого адресная часть равна адресной части кода «$\tilde{a}$», код операции и признаки равны сумме соответствующих разрядов «$\tilde{a}$» и «$\tilde{b}$». Перенос из 45-го разряда отбрасывается $\tilde{c}_A := \tilde{a}_A,$ $\tilde{c}_T := \tilde{a}_T \overset{\theta}{\boxplus} \tilde{b}_T$	24	При наличии переноса из 45-го разряда в 37-й $\tilde{a}_T + \tilde{b}_T \geqslant 2^9$
Вычитание кодов операций $\tau_1\tau_2\tau_3$ 73 *a b c*	73	Отличается от предыдущей операции тем, что сложение производится с поразрядной инверсией кода «$\tilde{b}_T$». Имеется циклический круговой перенос из 45-го разряда в 37-й. $\tilde{c}_A := \tilde{a}_A,$ $\tilde{c}_T := \tilde{a}_T \oplus [\bar{\tilde{b}}_T]$	24	$\tilde{a}_T + [\bar{\tilde{b}}_T] \geqslant 2^9$

Таблица 16.4. Команды управления

Наименование команды и ее вид	Код операции θ	Содержание операции и пояснения	Время выполнения, *мксек*
Безусловный переход с передачей кода $\tau_1\tau_2\tau_3$ 56 *a b c*	56	Всегда происходит передача управления команде, стоящей в ячейке с номером *b*. В ячейку $\tilde{c}$ без преобразования переносится код из ячейки $\tilde{a}$. $C := k_{II}$, $\tilde{c} := \tilde{a}$	24
Безусловный переход с записью команды возврата $\tau_1\tau_2\tau_3$ 16 *a b c*	16	Отличается от предыдущей команды тем, что в ячейку $\tilde{c}$ записывается «команда возврата»; $\tilde{c} := 0\ 16\ 0000\ k_I\ 0000$, представляющая собой команду безусловного перехода к ячейке $\tilde{a}$. (При выполнении команды возврата запись в ячейку с номером 0000 не важна, так как из нее нет выдачи.) Если «k_I» $= k + 1$, где k — номер ячейки, из которой выдана на РК команда с кодом операции $\theta = 16$, то команда возврата при своем исполнении действительно «возвращает» управление команде с номером $k + 1$. $\tilde{c}_{\tau_1} := 0$, $\tilde{c}_I := 0$, $\tilde{c}_{\tau_2} := 0$, $\tilde{c}_{II} := k_I$, $\tilde{c}_{\tau_3} := 0$, $\tilde{c}_{III} := 0$ $\tilde{c}_\theta := 16$,	24

Продолжение табл. 16.4

Наименование команды и ее вид	Код операции θ	Содержание операции и пояснения	Время выполнения, *мксек*
Условный переход по $\omega = 1$ $\tau_1\tau_2\tau_3$ 36 *a b c*	36	Если $\omega = 1$, то происходит передача управления ячейке с номером $\tilde{b}$, если $\omega = 0$, то выполняется следующая команда. В ячейку $\tilde{c}$ без преобразования переносится код из ячейки $\tilde{a}$: $\tilde{c} := \tilde{a}$, $C := \begin{cases} k_{II} \text{ при } \omega = 1, \\ C + 1 \text{ при } \omega = 0 \end{cases}$	24
Условный переход по $\omega = 0$ $\tau_1\tau_2\tau_3$ 76 *a b c*	76	Выполняется то же самое, что и при выполнении предыдущей команды, но при противоположных значениях сигнала ω $\tilde{c} := \tilde{a}$, $C := \begin{cases} k_{II} \text{ при } \omega = 0, \\ C + 1 \text{ при } \omega = 1 \end{cases}$	24

Продолжение табл. 16.4

Наименование команды и ее вид	Код операции θ	Содержание операции и пояснения	Время выполнения, *мксек*
Посылка на РА 2-го адреса команды $\tau_1\tau_2\tau_3$ 52 *a b c*	52	При выполнении команды происходит: 1) перенос начального содержимого регистра адреса — $[\langle\delta\rangle]_k$ в ячейку $\tilde{c}$ для запоминания, 2) запись на РА содержимого 2-го адреса команды $\tilde{c}_{\tau_1} := 0, \quad \tilde{c}_{\mathrm{I}} := 0,$ $\tilde{c}_{\tau_2} := 0, \quad \tilde{c}_{\mathrm{II}} := k_{\mathrm{I}} \underset{12}{\boxplus} \delta \cdot k_{\tau_{\mathrm{I}}},$ $\tilde{c}_{\tau_3} := 0, \quad \tilde{c}_{\mathrm{III}} := 0.$ $\tilde{c}_{\theta} := 52, \quad \delta := k_{\mathrm{II}}.$ Для того чтобы запомнить только начальное содержимое РА, необходимо положить в команде $k_{\tau_1} = 1$ и $k_{\mathrm{II}} = 0$. Если $k_{\tau_1} = 0$, то $\tilde{c}_{\mathrm{II}} := k_{\mathrm{I}}$. В дальнейшем при выполнении команды, записанной в ячейку $\tilde{c}$, $\delta := \begin{cases} k_{\mathrm{I}} & \text{при } k_{\tau_1} = 0, \\ k_{\mathrm{I}} \underset{12}{\boxplus} [\delta]_k & \text{при } k_{\tau_1} = 1;\ k_{\mathrm{I}} \neq 0, \\ [\delta]_k & \text{при } k_{\tau_1} = 1;\ k_{\mathrm{I}} = 0; \text{ в этом случае происходит восстановление содержимого РА} \end{cases}$	24

Продолжение табл. 16.4

Наименование команды и ее вид	Код операции θ	Содержание операции и пояснения	Время выполнения, *мксек*
Посылка на РА числа из ячейки МОЗУ $\tau_1\tau_2\tau_3$ 72 *a b c*	72	Отличается от предыдущей команды тем, что на РА записывается число, заданное во 2-м адресе кода, хранящегося в ячейке $\tilde{b}$. $\delta := \tilde{b}_{II}$	24
Передача управления после цикла (Конец цикла) $\tau_1\tau_2\tau_3$ θ *a b c*	12	Происходит передача управления к команде, стоящей в ячейке с номером $\tilde{b}$, если содержимое регистра переадресации меньше значения первого адреса $\delta < k_I$. В противном случае выполняется следующая команда. Кроме этого, в конце выполнения команды на РА записывается значение третьего адреса: $\delta := k_{III}$	24
	32	Отличается от предыдущей команды тем, что передача управления команде с номером $\tilde{b}$ производится тогда, когда $\delta \geqslant k_{II}$	24

Продолжение табл. 16.4

Наименование команды и ее вид	Код операции θ	Содержание операции и пояснения	Время выполнения, *мксек*
	11	Отличается от команды $\theta = 12$ тем, что передача управления происходит тогда, когда $\delta < k_1$ и $\omega = 1$ (сигнал ω выработан в предыдущем такте)	24
	31	Отличается от команды $\theta = 12$ тем, что передача управления происходит при $\delta \geqslant k_1$ и $\omega = 1$	24
	51	Отличается от команды $\theta = 12$ тем, что передача управления происходит при $\delta < k_1$ и $\omega = 0$	24
	71	Отличается от команды $\theta = 12$ тем, что передача управления происходит при $\delta \geqslant k_1$ и $\omega = 0$	24
Останов $\tau_1\tau_2\tau_3$ θ *a b c*	17 37 57 77	Машина останавливается. На PI и PII записываются коды, хранящиеся в ячейках $\tilde{a}$ и $\tilde{b}$. Они могут быть прочитаны на пульте управления. В ячейку $\tilde{c}$ записывается машинный нуль: $\tilde{c} := 0$	24 — выполнение команды, затем машина остановлена

После выполнения операций управления сохраняется значение сигнала ω, выработанное в предыдущем такте. $\varphi = 0$.

Операции обмена информацией между ОЗУ и МБ или ОЗУ и МЛ; операции ввода и вывода (табл. 16.5) осуществляют групповую передачу кодов, а также некоторые вспомогательные действия, связанные с работой электромеханических запоминающих устройств. Благодаря тому, что при выполнении этих операций производится контроль передачи, в некоторых случаях вырабатывается сигнал ω, который определяет логику контроля и остается после выполнения команды. $\varphi = 0$.

Время выполнения этих операций определяется быстродействием соответствующих устройств, количеством кодов в передаваемой группе, а для МЛ еще и временем ожидания требуемой зоны.

Общие замечания к системе операций и команд. Во всех командах таблиц 16.1 ÷ 5, исключая особо отмеченные, признаки τ_1, τ_2, τ_3 служат для формирования исполнительных адресов по правилу:

$$\tilde{a} = a \underset{12}{\boxplus} \tau_1 \cdot «\delta»,$$

$$\tilde{b} = b \underset{12}{\boxplus} \tau_2 \cdot «\delta»,$$

$$\tilde{c} = c \underset{12}{\boxplus} \tau_3 \cdot «\delta»,$$

«δ» — содержимое регистра переадресации во время выполнения данной команды. Для исключения возможных сомнений при рассмотрении системы операций следует принять правило: команда $\tau_1\ \tau_2\ \tau_3\ \theta\ a\ b\ c$ при своем выполнении эквивалентна команде

$$000 \quad \theta\ (a \underset{12}{\boxplus} \tau_1 \cdot «\delta»)\ (b \underset{12}{\boxplus} \tau_2 \cdot «\delta»)\ (c \underset{12}{\boxplus} \tau_3 \cdot «\delta»),$$

где θ — любое.

В машине может быть предусмотрен любой порядок чередования большинства команд. Независимость выполнения нарушается в случае выполнения команд $\theta = 50$ и $\theta = 70$. При этом выполнение команды $\theta = 50$ равносильно выполнению «холостого» такта, при выполнении $\theta = 70$ происходит останов.

Таблица 16.5. Команды, осуществляющие операции обмена информации между ОЗУ и МБ, ОЗУ и МЛ, операции ввода и вывода

Наименование команды и ее вид	Код операции (значения θ)	Содержание операции	Условия выработки ω = 1
Ввод с перфокарт $\tau_1\tau_2\tau_3\ \theta\ a\ b\ c$	10	В последовательность ячеек ОЗУ, начиная с $\tilde{a}$, вводится группа кодов, пробитых на перфокартах, $\tilde{c}$ — номер ячейки, в которую вводится сумма кодов, накопленная в процессе ввода на сумматоре АУ по правилам операции θ = 07. $\tilde{b}$ — номер ячейки, в которую производится передача управления, если сумма, найденная в АУ, и последний код из вводимого материала не совпадают (поразрядно) между собой. При несовпадении вначале возникает останов, а переход осуществляется при нажатии кнопки «Пуск». Последний код имеет два маркера (справа и слева). Начальный адрес может указываться на перфокарте в виде так называемого «адресного кода», который имеет один маркер справа (суммируется так же, как и все остальные коды). Адресный код прерывает ввод в последовательность ячеек, начинающуюся с номера $\tilde{a}$, и продолжает ввод в последовательность, начинающуюся с ячейки, имеющей номер, указанный в разрядах первого адреса строки перфокарты, помещающей адресный код. Таким образом, возможен ввод нескольких групп (массивов) кодов в различные последовательности ячеек ОЗУ с помощью одной команды путем пробивки адресных кодов.	При несовпадении сумм

Продолжение табл. 16.5

Наименование команды и ее вид	Код операции (значения θ)	Содержание операции	Условия выработки ω = 1
		Предусмотрен ряд блокировок. Если $\tilde{a}=0$, то запись в ОЗУ блокируется. В адресном коде 1-й, 12-й и 13-й разряды выполняют соответственно блокировки: останова, контроля и записи в ОЗУ. Блокировки останова и контроля, указанные в одном из адресных кодов, действуют на ввод всех массивов. Блокировка записи действует на ввод всех массивов, следующих за адресным кодом, снабженным признаком блокировки.	
Ввод с перфокарт без останова $\tau_1\tau_2\tau_3$ θ *a b c*	30	Отличается от предыдущей операции тем, что при несовпадении сумм управление передается к ячейке с номером $\tilde{b}$ без останова	
Обращение к магнитным лентам, магнитным барабанам, к устройствам вывода $\tau_1\tau_2\tau_3$ 50 *a b c* $\tau'_1\tau'_2\tau'_3$ 70 *a′ b′ c′*	50 70	Здесь: $\tilde{b}'$ — начальный адрес, $\tilde{c}$ — конечный адрес последовательности ячеек ОЗУ, в которые записывается или из которых считывается группа кодов; $\tilde{a}$ — адрес зоны МЛ или адрес первого кода на МБ. Характер передачи определяется значениями разрядов адреса $\tilde{a}$ согласно таблице *):	

*) 1-й разряд адреса $\tilde{a}$ является 25-м разрядом команды, 2-й разряд — 26-м разрядом, и так далее.

Продолжение табл. 16.5

Наименование команды и ее вид	Код операции (значения θ)	Содержание операции	Условия выработки ω = 1
Выполняется всегда по двум командам, записываемым в две последовательные ячейки ОЗУ. Первая команда является подготовительной, вторая — исполнительной.		Разряд / Значение разряда и характер операции	
		1, 2 — Указывается номер МБ или номер блока МЛ	
		3 — 0 — считывание, 1 — запись	
		4 — 1 — операции производятся с МБ	
		5 — 1 — операции производятся с МЛ	
		6 — 1 — выполняется разметка ленты, т. е. запись на МЛ адреса зоны и передаваемых кодов в эту новую зону. Старая запись, имевшаяся на этом месте, стирается	
		7 — 1 — производится выдача на печать (через буфер)	
		8 — 1 — производится выдача на перфоратор (через буфер)	

Продолжение табл. 16.5

Наименование команды и ее вид	Код операции (значения θ)	Содержание операции	Условия выработки ω = 1

Разряд	Значение разряда и характер операции
9	1 — блокируется останов при несовпадении сумм (см. ниже)
10	0 — лента движется в прямом направлении 1 — лента движется в обратном направлении
11	1 — блокировка контроля (см. ниже)
12	1 — блокировка записи в ОЗУ (см. ниже)

При выполнении всех этих операций производится суммирование кодов в АУ (по правилам операции θ = 07).

$\tilde{c}'$ — адрес ячейки ОЗУ, в которую записывается найденная сумма.

При передаче из ОЗУ на МБ и МЛ найденная сумма записывается как добавочный код, помещаемый после передаваемой группы кодов. При чтении с МБ

Продолжение табл. 16.5

Наименование команды и ее вид	Код операции (значения θ)	Содержание операции	Условия выработки $\omega = 1$
		и МЛ этот добавочный код сравнивается (по правилам $\theta = 15$) с найденной суммой и при совпадении после выполнения команд $\theta = 50$, $\theta = 70$ управление передается следующей команде; в противном случае машина останавливается, при нажатии кнопки «Пуск» выполняется команда, стоящая в ячейке $\tilde{b}'$. В момент останова на пульте управления PI содержит добавочный код, сумматор содержит найденную сумму. Для проверки записи на МБ и МЛ используется также режим чтения из этих устройств с блокировкой записи в ОЗУ (1 в 12-м разряде $\tilde{a}$). При выдаче на печать или на перфоратор контроль применяется для проверки правильности передачи в буферное устройство. На перфокартах в последней строке всей выдачи пробивается также сумма выводимых кодов. При комбинациях разрядов, определяющих выполнение несовместных операций, машина останавливается ($\varphi = 1$). Это бывает, когда одновременно заданы единицы в разрядах: 5 и 6; 4 и 5; 4 и 6; 4 и 7 (или 8); 6 и 10; 5 и 7 (или 8); 4 и 10; 6 и 11; 5 и 11. Машина останавливается также, если ни в одном из разрядов адреса $\tilde{a}$ с номерами: 4; 5; 6; 7; 8 нет 1.	

В табл. 16.1 используется символ *l*, показывающий число сдвигов на один разряд, выполняемых в АУ при формировании результата (выравнивание порядков, нормализация).

Пульт управления

Пульт управления предназначен для контроля правильности решения задачи и ручного управления машиной.

На панели сигнализации имеются неоновые лампочки, отображающие содержимое следующих регистров: регистра результата, на котором хранится последний записанный в ОЗУ код, сумматора и регистров АУ, регистра команд и регистра номера команды (КРА).

На панели сигнализации имеются также дополнительные лампочки, показывающие режим работы, причины останова, код выполняемой операции, значения сигналов ω и φ, характер работы МБ, МЛ, устройств ввода и вывода и т. д.

На пульте управления имеются следующие клавиши и кнопки:

Шесть рядов клавиш по 45 штук в каждом. Из них 4 ряда представляют собой четыре дополнительные ячейки, из которых при выключении тумблера «Блокировка ДЗУ» могут в темпе работы ОЗУ считываться коды. Их номера: 7771, 7772, 7773, 7774. Пятый ряд клавиш служит для передачи кода в регистр результата, откуда он может быть записан в ОЗУ. Номер регистра результата: 7775. Шестой ряд клавиш служит для ввода кодов команд на РК. Слева у каждого ряда клавиш имеется клавиша исполнения, справа — клавиша сброса;

Три ряда клавиш по 12 штук в каждом из них, служащие: 1-й ряд — для останова машины при совпадении набранного значения с КРА (останов по адресу команды); 2-й ряд — для останова машины при совпадении набранного значения с адресом, по которому производится запись (останов по 3-му адресу); 3-й ряд — для вызова кода, адрес которого набран на клавишах, на ПУ (код фиксируется на РР). Эта же клавиатура (после переключения соответствующего тумблера) служит для останова по содержимому РА.

Тумблеры, осуществляющие блокировку чтения для записи для МБ и МЛ.

Кнопка «Пуск», при нажатии которой УУ начинает выдачу основных импульсов такта.

Адрес	Команды и числа					Пояснения	№№ поз.
4020	0					Команды	
1		52					2
2					0020		3
3	2	07	0020	4034	0020		4
4	1	12	0013	4023	0001		5
5		35	0020	0100			6
6		76		2010			7
7		50	0200		4047		8
4030		70	4034				9
1		50	0500		4041		10
2		70	4034				11
3		56		4131			12
4	1	01	4000			1 — Двоичные числа	
5	2		4000			—0,5	2
6		77	4000			0,25	3
7	1	07	6200			100	4
4040		77	5252	5252	5253	$^1/_8$	5
1	2	77	5252	5252	5253	$-^1/_8$	6
2	0						7
3	0						8
4	0						9
5		02	144			14,4 — Десятично-двоичные числа	10
6	3	01	333	333	333	$-^1/_{30}$	11
7		01	500			5	12

Рис. 16.2. Образец записи команд и чисел на бланке.

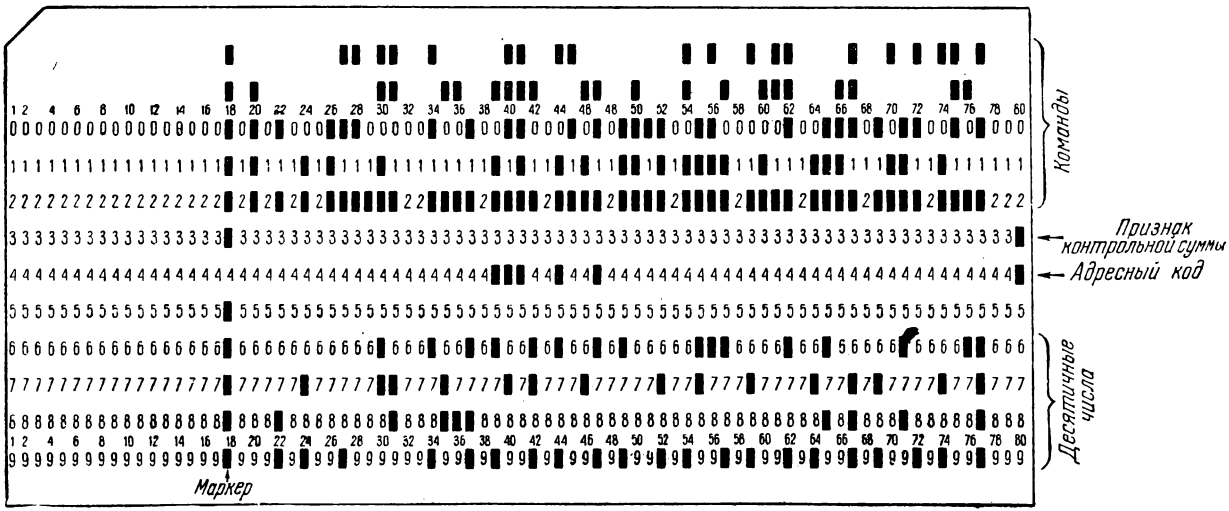


Рис. 16.3. Перфокарта машины *М-20*.

Две кнопки «Стирание», при одновременном нажатии которых во все ячейки ОЗУ записывается нуль.

Кнопка «Останов», при нажатии на которую машина останавливается.

Две кнопки сброса содержимого регистров и сумматора АУ.

Кнопка «Ввод», нажатие которой эквивалентно выполнению команды: 0 10 0001 0001 0000 с предварительным гашением регистров РК, РА и регистров АУ (регистр результата не сбрасывается). По этой команде в случае совпадения сумм после ввода кодов, пробитых на перфокартах, в ячейки ОЗУ, начиная с ячейки 0001, производится передача управления ячейке 0001. В случае несовпадения машина остановится в ячейке 0000, при нажатии кнопки «Пуск» управление передается ячейке 0001.

Подготовка данных. Ввод и вывод

Программа задачи и исходные данные записываются на бланках, а затем переносятся на перфоленту или на перфокарты. На машине *М-20* для записи чисел и команд применяют бланки, возможный вид которых показан на рис. 16.2.

На рис. 16.3 даны образцы пробивки на перфокарте.

ГЛАВА 17

ПРОГРАММНО-УПРАВЛЯЕМАЯ МАШИНА *БЭСМ-2*

Основные характеристики машины *БЭСМ-2*

Тип машины	вычислительная
Класс машины	средняя
Число основных операций в секунду	$8 \div 10$ тысяч
Рабочая система счисления	двоичная
Форма представления чисел	с плавающей запятой
Длина кода числа	39 двоичных разрядов (мантисса со знаком — 33 разряда, порядок со знаком — 6 разрядов)
Диапазон представимых чисел	$2^{-88} \leqslant \lvert x \rvert \leqslant 2^{81}(1 - 2^{-32})$
Точность вычислений	десять десятичных значащих цифр
Порядок выполнения команд	заданный (естественный)
Адресность машины	трехадресная
Виды запоминающих устройств	оперативное запоминающее устройство: выполнено на магнитных сердечниках; емкость 2047 39-разрядных двоичных кодов; время обращения 10 *мксек*;
	магнитное запоминающее устройство: два магнитных барабана, каждый емкостью по 6144 кодов; скорость последовательной записи или чтения 800 кодов в секунду; магнитная лента: емкость 200 000 кодов; скорость записи или чтения $350 \div 400$ кодов в секунду
Виды ввода и вывода	ввод с бумажной перфоленты со скоростью 20 кодов в секунду и с перфокарт со скоростью 20 кодов в секунду;
	вывод на печатающее устройство со скоростью 15 кодов в секунду и на перфокарты со скоростью 10 кодов в секунду

Количество электронных ламп	5000
Потребляемая мощность	35 *квт*
Площадь, необходимая для размещения машины	собственно машина занимает 150 ÷ 200 $м^2$

Машина *БЭСМ-2* является универсальной.

Эксплуатируется круглосуточно, допустимы перерывы в работе с выключением питания с временем восстановления работоспособности 30 ÷ 50 минут.

Краткие сведения об основных устройствах машины

Арифметическое устройство (АУ). АУ работает в дополнительном коде.

В состав АУ входят следующие основные регистры: сумматор мантисс чисел (35 разрядов), сумматор порядков чисел (7 разрядов), регистры мантиссы и порядка 1-го числа, регистры мантиссы и порядка 2-го числа (по 34 и 6 разрядов соответственно).

Сумматоры служат для выполнения всех операций над кодами. При выполнении большинства операций результат образуется на сумматорах и передается для записи в ОЗУ. Регистры 1-го и 2-го чисел служат для приема кодов из ОЗУ и передачи их на сумматор, а также для других вспомогательных действий в АУ.

Оперативное запоминающее устройство (ОЗУ) построено на ферритовых сердечниках и имеет 2048 39-разрядных ячеек, занумерованных в восьмеричной системе от 0000 до 3777. В ОЗУ имеется ячейка (адрес 0000), из которой при считывании всегда выдается код, содержащий нули во всех разрядах.

Устройство управления (УУ). Основными элементами УУ являются следующие:

Регистры-счетчики номера команды («счетчики команд»). Таких счетчиков два: счетчик центрального управления командами (*счетчик* ЦУК — *C*) и счетчик местного управления командами (*счетчик* МУК — *D*); счетчики имеют по 11 разрядов. Машина может работать в режиме центрального или местного управления командами. При этом содержимое соответствующего счетчика является адресом, по которому производится выборка команды из ОЗУ для исполнения. Существуют команды, переключающие управление с одно-

го счетчика на другой. Счетчик МУК применяется, кроме того, для подсчета количества передаваемых кодов при групповом обмене между ОЗУ и МЗУ или устройствами ввода — вывода. Благодаря этому счетчик МУК при программировании используется для вспомогательных целей (выполнение подпрограмм, не содержащих команд группового обмена информацией между запоминающими устройствами и т. д.). В остальном оба счетчика равноценны.

Регистр команд. Имеет 39 двоичных разрядов. Служит для приема кода команды, выдаваемой из ОЗУ по адресу ЦУК (или МУК) и для его хранения в процессе исполнения.

В состав УУ входят также схемы, обеспечивающие: расшифровку кода выполняемой операции, задание стандартного такта (цикла) работы машины, удлинение этого такта при выполнении некоторых операций и т. д.

Работа машины слагается из выполнения элементарных тактов, каждый из которых в свою очередь состоит из стандартного такта (цикла), длительностью 65 *мксек*, и временной задержки, определяемой логикой выполняемых операций.

Элементарный такт начинается с приходом импульсов «выдача 1-го числа» и «выдача 2-го числа». По этим импульсам, согласно 1-му и 2-му адресам команды, стоящей на регистре команд, из ОЗУ в АУ передаются 1-е и 2-е числа. Затем, в соответствии с кодом операции команды, установленной на регистре команд, УУ выдает серию импульсов, управляющих работой АУ. После выполнения операции выдается импульс «запись», по которому результат, образованный в АУ, передается в ОЗУ согласно 3-му адресу команды, установленной на регистре команд. Время выполнения большинства операций (при блокировке нормализации), за исключением операций умножения и деления, укладывается в стандартный такт работы машины (65 *мксек*).

Для операций умножения и деления, а также в случае нормализации результатов производится удлинение такта путем задержки импульса «запись» примерно на 180 *мксек* (для операций умножение и деление) и на 25 *l мксек* при нормализации (*l* — количество сдвигов, необходимых для выравнивания порядков при нормализации).

Во время работы АУ производится увеличение содержимого счетчика ЦУК (или МУК) на единицу. После импульса «запись» выдается импульс «выдача команды», по которому, согласно новому содержимому счетчика ЦУК (или МУК), из ОЗУ выбирается очередная команда и записывается на регистр команд. На этом выполнение элементарного такта заканчивается.

Все перечисленные и другие управляющие импульсы образуются в устройстве управления путем преобразования сигналов

задающего генератора, поступающих через 2,5 *мксек*. Переход от такта к такту осуществляется автоматически.

Порядок выполнения операций передачи управления, а также обращения к МЗУ несколько отличается от описанного.

Магнитное запоминающее устройство (МЗУ) состоит из двух магнитных барабанов и магнитных лент.

Магнитные барабаны (МБ). Устройство содержит два магнитных барабана, вращающихся со скоростью 780 *об/мин*.

Барабаны имеют номера «0» и «1». Последовательная выборка и запись кодов производится со скоростью 800 кодов в секунду. Среднее время ожидания требуемого кода 40 *мсек*. Максимальное время — 80 *мсек*.

Емкость каждого из барабанов 6144 двоичных 39-разрядных ячеек. Все ячейки разбиты на три группы по 2048 в каждой. Номера ячеек первой группы от 00000 до 03777, второй — от 10000 до 13777, третьей — от 20000 до 23777. Считывание и запись может производиться, начиная с любой ячейки МБ. При обращении к последовательности ячеек, содержащей ячейки разных групп, после использования ячеек одной группы используются ячейки следующей группы, начиная с имеющей наименьший номер.

Магнитная лента (МЛ). Имеется четыре независимых лентопротяжных блока; им присвоены номера 0, 1, 2, 3. Емкость каждого лентопротяжного блока 30 000 ÷ 200 000 39-разрядных кодов; коды на ленте располагаются зонами. Длина ленты на одном лентопротяжном блоке — 300 *м*. Скорость записи или чтения 400 кодов в секунду. Скорость движения ленты 2 *м* в секунду. Лента движется в обе стороны. Запись и считывание производится зонами. На одной ленте может быть не более 63 зон. Ширина ленты — 6,5 *мм*. Запись двухдорожечная, на одной дорожке записывается последовательно число, на другой — синхронизирующие импульсы. Бобины с лентами сменные и могут быть переставлены с одного лентопротяжного блока на другой. Максимальное время подвода требуемой группы кодов 100 *сек*.

Устройства ввода и вывода (УВВ). В машине имеется два типа устройств ввода: с перфорированной ленты и с перфокарт. Перфолента размещается на бобинах, аналогичных бобинам МЛ. Считывание фотоэлектрическое со скоростью 20 кодов в секунду. Ввод с перфокарт осуществляется со скоростью 20 кодов в секунду. Вывод осуществляется

на печатающее устройство со скоростью 20 чисел в секунду или на перфокарты со скоростью 10 кодов в секунду. С перфокарт печать осуществляется на внешних устройствах в виде команд или чисел со скоростью 75 кодов в минуту.

Представление чисел и команд

Распределение разрядов ячейки ОЗУ при хранении в ней двоичного 39-разрядного числа представлено на рис. 17.1, *а*. Число представляется с плавающей запятой. Порядок записывается в ячейке в дополнительном коде, а мантисса в прямом.

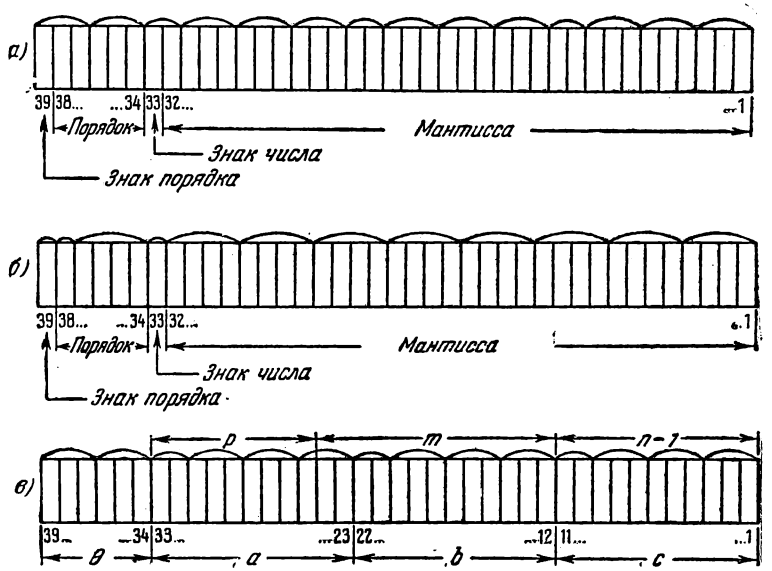


Рис. 17.1. Распределение разрядов ячейки памяти: *а*) для двоичного числа: *б*) для двоично-десятичного числа; *в*) для команды.

Машинный нуль представляется кодом, имеющим нули в разрядах мантиссы и минимальный порядок $P = -2^5$. Код, состоящий из нулей, не является машинным нулем, а представляет собой ненормализованное число, в котором $M = 0$ и $P = 0$.

Распределение разрядов ячейки ОЗУ при хранении в ней двоично-десятичного кода представлено на рис. 17.1, *б*. Десятичное число представляется с плавающей запятой. Порядок и мантисса выражаются в прямом коде.

В знаковых разрядах двоичного и десятичного чисел 0 соответствует знаку плюс; 1 — знаку минус.

Распределение разрядов ячейки при хранении в ней команды показано на рис. 17.1, *в*.

Основной формат команды машины *БЭСМ* имеет следующий вид:

$$\theta\ a\ b\ c,$$

где θ — код операции, a, b, c — соответственно первый, второй и третий адреса команды.

Вспомогательный формат команды, применяемый для обращений к МБ и МЛ, имеет вид

$$\theta\ p\ m\ n-1,$$

где θ — код операции, p — признак, m — номер ячейки или зоны, n — число передаваемых кодов.

На рис. 17.1 скобками указаны объединения двоичных разрядов в восьмеричные, четверичные и десятичные цифры, применяемые при записи чисел и команд.

Система операций и команд

Система операций и команд приведена в табл. 17.1.

Замечания: 1. Операции $\theta = 41 \div 43$; $45 \div 55$, 65 выполняются аналогично операциям с $\theta = 01 \div 03$; $05 \div 15$, 25, но без нормализации результата.

2. Арифметические операции могут производиться и над ненормализованными числами. При делении получается правильный результат при $M(a) < 2M(b)$.

Пульт управления

Пульт управления предназначен для контроля правильности решения задачи и ручного управления машиной.

На панели сигнализации имеются неоновые лампочки, отображающие содержимое следующих регистров: триггерного контрольного запоминающего устройства (в котором находится

Таблица 17.1. Система операций и команд машины *БЭСМ-2*

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
00 (40)	Передача числа 00 *a* * *c*	$c := a$		65	Содержимое ячейки *a* без преобразования записывается в ячейку *c*
01 (41)	Сложение 01 *a b c*	$c := a + b$	$c_P > 31$	65 ÷ 225 среднее 75	В ячейку *c* записывается сумма чисел, хранящихся в ячейках *a* и *b*. Округление производится перед нормализацией по 33-му (дополнительному) разряду мантиссы суммы. Округление не производится, когда при выравнивании порядков сдвигается вправо отрицательная мантисса суммы. Доокругление после нормализации вправо не производится
02 (42)	Вычитание 02 *a b c*	$c := a + \overline{\operatorname{sign} b} \mid b \mid$	$c_P > 31$	То же	После передачи в АУ производится изменение знака числа, поступившего из *b* на обратный, после чего выполняется операция «сложение» ($\theta = 01$)

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
03 (43)	Умножение 03 *a b c*	$c := a \times b$	$c_P > 31$	230	В ячейку *c* записывается произведение чисел, хранящихся в ячейках *a* и *b*. Округление производится так же, как и при операции $\theta = 01$
04	Деление 04 *a b c*	$c := a : b$	$c_P > 31$	до 235	В ячейку *c* записывается частное от деления числа «*a*» на число «*b*». Округление производится после нормализации по 33-му разряду мантиссы частного, который специально определяется для этой цели
44	Деление без округления 44 *a b c*	$c := a : b$	$c_P > 31$	до 235	Отличается от операции $\theta = 04$ тем, что округление не производится
05 (45)	Сложение порядков 05 *a b c*	$c := a \cdot 2^{b_P}$	$c_P > 31$	65	В ячейку *c* записывается код, у которого мантисса равна мантиссе числа «*a*», порядок равен сумме порядков «a_P» и «b_P»

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
06 (46)	Вычитание порядков 06 *a b c*	$c := a : 2^{bP}$	$c_P > 31$	65	После передачи в АУ производится изменение знака порядка числа, поступившего из ячейки *b* на обратный, после чего выполняется операция «сложение порядков» ($\theta = 05$)
07 (47)	Выделение порядка 07 *a* * *c*	$c := a_P$	$a_P = = -32$	$65 \div 225$	В ячейку *c* записывается число, равное порядку числа «*a*». При выполнении операции порядок числа передается в $28 \div 32$ разряды мантиссы сумматора $s_M := a_P \cdot 2^{-5}$, а в разряды сумматора порядка записывается число 5 $s_P := 5$ Содержимое сумматора передается в ячейку *c* с предварительной нормализацией
10 (50)	Умножение без округления 10 *a b c*	$c := a \cdot b$	31	230	Отличается от операции $\theta = 03$ тем, что округление не производится

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
11 (51)	Вывод младших разрядов произведения 11 *a b c*		31	230	Умножается «*a*» на «*b*». В ячейку *c* записывается число, у которого мантисса содержит 32 младших разряда 64-разрядной мантиссы произведения «a_M»·«b_M»; порядок и знак *c* равны порядку и знаку «*a*»·«*b*»
14 или 54	Выделение целой и дробной частей числа 14 *a b c*	$c := E(a) \cdot 2^{-32}$, $b := D(a)$		65 ÷ 225	В ячейку *c* записывается целая часть числа «*a*», в ячейку *b* — дробная часть «*a*». Ненормализованное число, выражающее целую часть «*a*», размещается в ячейке *c* следующим образом: младший разряд его мантиссы, соответствующий 2^0, попадает в 1-й разряд ячейки *c*. Порядок «*c*» равен нулю, знак положительный. Число, выражающее дробную часть, денормализуется так, чтобы «b_p» = 0

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
15 (55)	Присвоение знака 15 *a b c*	$c := a$ при $b > 0$; $c := \overline{\operatorname{sign} a} \mid a \mid$ при $b < 0$		65	В ячейку *c* записывается число «*a*», если «*b*» > 0 и «$-a$», если «*b*» < 0
16 или 56	Логическое сложение 16 *a b c*	$c_i := a_i \vee b_i$, $1 \leqslant i \leqslant 39$		65	В ячейку *c* записывается результат поразрядного логического сложения кодов, стоящих в ячейках *a* и *b*
17 или 57	Логическое умножение 17 *a b c*	$c_i := a_i \wedge b_i$, $1 \leqslant i \leqslant 39$		65	В ячейку *c* записывается результат поразрядного логического умножения кодов, стоящих в ячейках *a* и *b*
20	Отрицание равнозначности 20 *a b c*	$s_i := a_i \not\sim b_i$, $1 \leqslant i \leqslant 39$		65	В ячейку *c* записывается результат поразрядной логической операции «сложение по модулю 2» над кодами, стоящими в ячейках *a* и *b*

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
60	Циклическое сложение 60 *a b c*	$c := a \oplus b$		65	В ячейку *c* записывается результат сложения с круговым переносом (из 39 разряда в первый) кодов, содержащихся в ячейках *a* и *b*
22	Сложение команд 22 *a b c*	$c_P := a_P$, $c_M := a_M \underset{32}{\boxplus} b_M$		65	В ячейку *c* записывается код, у которого разряды кода операции равны соответствующим разрядам кода, стоящего в ячейке *a*. Адресная часть этого результирующего кода равна сумме адресных частей кодов, стоящих в ячейках *a* и *b*. При суммировании перенос из 33-го разряда в 34-й блокируется
62	Вычитание команд 62 *a b c*	$c_P := a_P$ $c_M := a_M \underset{32}{\boxplus} [b_M]_{\text{доп}}$		65	Адресная часть кода, поступившего в АУ из ячейки *b*, представляется в дополнительном коде, после чего выполняется операция «сложение команд» ($\theta = 22$)

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
32	Печать 32 *a* * *c*	$c := a$		0,05 *сек*	Содержимое ячейки *a* печатается в шестнадцатеричной системе счисления и передается в ячейку *c*
72	Пропуск строки («интервал») 72 *a* * *c*	$c := a$		То же	Бумажная лента печатающего устройства передвигается на одну строку. Содержимое ячейки *a* передается в ячейку *c*
24	Подсчет единиц 24 *a* * *n c*	При $k_{18} = 1$ $c := l \cdot 2^{-32}$; при $k_{18} = 0$ $c := 0$		$65 \div 225$	При $k_{18} = 1$ ($n \geqslant 64$) среди $n - 64$ правых разрядов кода «*a*» определяются содержащие единицы и подсчитывается их количество. Полученное целое число l записывается в ячейку *c* так, что значение его младшего разряда, соответствующего 2^0, попадает в 1-й разряд мантиссы. Порядок «*c*» равен нулю. При $k_{18} = 0$ ($n < 64$) в ячейку *c* записывается нуль. *n* занимает $12 \div 18$-й разряды команды

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
25 (65)	Изменение порядка 25 *a* **n* *c*	$c := a \cdot 2^{k_n}$	31	65	Операция отличается от операции сложения порядков $\theta = 05$ тем, что вместо порядка числа «*b*» действие производится с числом *n*, представленным в дополнительном коде, со знаком в 6-м разряде адреса (как порядки двоичных чисел). *n* занимает 12 ÷ 17-й разряды команды
26	Полный сдвиг 26 *a* **n* *c*	$c_i := \begin{cases} a_{i-k_n} \text{ при} \\ 1 \leqslant i - k_n \leqslant 39, \\ 0 \quad \text{при} \\ i - k_n > 39, \\ i - k_n < 1, \end{cases}$ $1 \leqslant i \leqslant 39$		65 ÷ 225	В ячейку *c* записывается код «*a*», сдвинутый на *n* разрядов влево, если $n > 0$, или вправо, если $n < 0$; *n* представляется в дополнительном коде со знаком в 18-м разряде команды. *n* занимает 12 ÷ 18-й разряды команды

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
66	Неполный сдвиг 27 *a* **n c*	$c_i := \begin{cases} a_{i-k_n} \text{ при} \\ 1 \leqslant i - k_n \leqslant 32, \\ 0 \quad \text{при} \\ 1 > i - k_n, \\ 32 < i - k_n, \end{cases}$ $1 \leqslant i \leqslant 32,$ $c_j := 0,$ $34 \leqslant j \leqslant 39,$ $c_{39} := a_{39}$		$65 \div 225$	Осуществляется сдвиг мантиссы кода, стоящего в ячейке *a* на *n* разрядов влево, если $n > 0$, или вправо, если $n < 0$. Порядок результата всегда равен нулю. Знак результата равен знаку числа «*a*». Результат записывается в ячейку *c*. *n* представляется так же, как и в предыдущей команде
27 или 67	Пульт 27 *a b c*			65	Над содержимым ячейки *a* и регистров, указываемых адресом *b*, производится поразрядное сравнение. Результат записывается в ячейку *c*. В операции участвуют: при $b = 0001$ содержимое верхнего ряда клавиш пульта управления, при $b = 0002$ содержимое нижнего ряда клавиш пульта управления, при $b = 0004$ со-

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
					держимое ячейки ТКЗУ (результат предыдущей операции), при $b = 0010$ содержимое счетчика ЦУК. В тех случаях, когда b содержит единицы более чем в одном из управляющих разрядов, происходит сравнение с поразрядной логической суммой содержимого соответствующих регистров. Содержимое счетчика ЦУК представляется в виде 39-разрядного кода, помещающего «*C*» в $1 \div 11$ разрядах
30 31 или 30 71	Обращение к МЗУ, ввод и вывод				Выполняется с помощью двух команд, стоящих в ячейках с последовательными номерами. Конкретная операция определяется значением признака p, θ и a.

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
$p = 14$	Запись на магнитный барабан '30 14 *m n* — 1 31 * *b c*			1250 для передачи каждого кода без учета времени ожидания	В ячейки МБ, начиная с *m*, записывается *n* кодов из ячеек ОЗУ, начиная с *b*. В ячейку *c* записывается сумма передаваемых кодов, найденная по правилам операции $\theta = 60$
$p = 04$	Считывание с магнитного барабана '30 04 *m n* — 1 31 * *b c*			То же	В ячейки ОЗУ, начиная с *b*, записываются *n* кодов из ячеек МБ, начиная с ячейки *m*. В ячейку *c* записывается контрольная сумма, как и при операции записи на МБ ($p = 04$)
$p = 04$	Фиктивное считывание с магнитного барабана '30 04 *m n* — 1 71 * *b c*			То же	Отличается от предыдущей команды тем, что все коды записываются последовательно в ячейку *b* (обычно $b = 0000$)

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
$p = 12$	Запись на магнитную ленту '30 12 *m n* — 1 31 * *b c*			2400 для передачи каждого кода без учета времени поиска зоны	В зону *m* магнитной ленты записывается *n* кодов из ячеек ОЗУ, начиная с *b*. В ячейку *c* записывается сумма передаваемых кодов, найденная по правилам операции $\theta = 60$
$p = 02$	Считывание с магнитной ленты '30 02 *m n* — 1 31 * *b c*			То же	В ячейки ОЗУ, начиная с *b*, записываются *n* кодов из зоны *m*. В ячейку *c* записывается контрольная сумма, как и при операции записи на магнитную ленту
$p = 22$	Обратная перемотка магнитной ленты 30 22 *m* * 31 * * *			до 100 *сек*	Лента движется в обратном направлении до зоны *m*

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
$p = 02$	Фиктивное считывание с магнитной ленты '30 02 *m n* — 1 71 * *b c*			То же, что и для считывания с ленты	Отличается от команды считывания с магнитной ленты тем, что все коды записываются последовательно в ячейку *b* (обычно $b = 0000$)
	Ввод с перфоленты '30 0204 *m n* — 1 31 * *b c*			0,05 *сек* для одного кода	В ячейки ОЗУ, начиная с *b*, вводится *n* кодов из зоны *m* перфоленты
	Ввод с перфокарт '30 0100 * *n* — 1 31 * *a c*			0,05 *сек* для ввода каждого кода	С перфокарт в ОЗУ вводится *n* кодов, начиная с ячейки *a*. В ячейку записывается контрольная сумма вводимых кодов, найденная по правилам операции с $\theta = 60$

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
	Вывод на перфокарты '30 1100 * $n-1$ 31 * b c			0,1 *сек* для вывода каждого кода	Из ОЗУ, начиная с ячейки b, на перфокарты выводится n кодов. В ячейку c записывается контрольная сумма выданных кодов Примечания: 1. После выполнения указанных команд на счетчике МУК устанавливается 0000. 2. Отдельное выполнение команд $\theta = 30$; $\theta = 31$ и $\theta = 30$; $\theta = 71$ не имеет смысла. 3. Для записи команд, отмеченных знаком ' применяется вспомогательный формат
33	Останов 33 * * *			65	Машина останавливается. Запись в ОЗУ не производится
73	Условный останов 73 * * *			65	Если на пульте управления включен ключ «условный останов», то машина останавливается. Запись в ОЗУ не производится

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
34	Изменение содержимого счетчика ЦУК с записью команды возврата 34 * *b c*	$b_0 := 74$, $b_I := 0$, $b_{II} := 0$, $b_{III} := C + 1$, $C := k_{III}$		65	Производится изменение содержимого счетчика ЦУК путем записи на него третьего адреса команды $-k_{III}$. Предварительно в ячейку *b* производится запись команды возврата 74 0000 0000 $C + 1$
74	Изменение содержимого счетчика ЦУК 74 * *b c*	$b := 0$, $C := k_{III}$		65	Отличается от команды $\theta = 34$ тем, что в ячейку *b* записывается код, содержащий нули во всех разрядах

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
35	Условный переход при совпадении 35 *a b c*	При $a_i = b_i$ $(1 \leqslant i \leqslant 39)$ $C := k_{\text{III}}$ или $D := k_{\text{III}}$, в противном случае $C := C + 1$ или $D := D + 1$		65	Если «*a*» совпадает с «*b*» по всем разрядам, то управление передается команде, стоящей в ячейке с номером *c*; передача управления достигается тем, что на счетчик ЦУК или МУК (в зависимости от того, по какой системе управления работает машина) записывается k_{III}
75	Условный переход при несовпадении 75 *a b c*	При $a_i \neq b_i$ хотя бы для одного i $(1 \leqslant i \leqslant 39)$ $C := k_{\text{III}}$ или $D := k_{\text{III}}$, в противном случае $C := C + 1$ или $D := D + 1$		65	Отличается от команды $\theta = 35$ тем, что передача управления производится в тех случаях, когда хотя бы один из соответствующих разрядов кодов «*a*» и «*b*» не совпадает

Продолжение табл. 17.1

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
36	Сравнение и условный переход 36 *a b c*	При $a < b$ $C := k_{III}$ или $D := k_{III}$, в противном случае $C := C + 1$ или $D := D + 1$		65	Отличается от команды $\theta = 35$ тем, что передача управления производится в тех случаях, когда: 1) «a_P» = «b_P» и «a_M» < «b_M»; 2) «a_P» > «b_P» и «a» < 0; 3) «a_P» < «b_P» и «b» ≥ 0. Для нормализованных «a» и «b» это означает, что «a» < < «b»
76	Сравнение по модулю и условный переход 76 *a b c*	При $\|a\| < \|b\|$ $C := k_{III}$ или $D := k_{III}$, в противном случае $C := C + 1$ или $D := D + 1$		65	Отличается от команды $\theta = 36$ тем, что в операции участвуют модули чисел «a» и «b»

Продолжение табл. 17.7

Код операции	Наименование команды и ее вид	Краткое содержание команды	Условия выработки $\varphi = 1$	Время выполнения, *мксек*	Пояснения
37	Передача управления счетчику ЦУК 37 * * *	$C := C + 1$		65	Производится прибавление единицы к содержимому счетчика ЦУК, после чего управление передается центральной системе управления командами. Если команда $\theta = 37$ выдавалась по счетчику ЦУК, то выполняется один «холостой» такт работы машины
77	Передача управления счетчику МУК 77 * * *c*	$D := k_{III}$			На счетчик МУК записывается содержимое 3-го адреса — *c* и передается управление местной системе
12 52 13 53 21 61 23 63	Не используются θ *a* * *c*	$c := a$		65	Выполнение любой из этих команд эквивалентно работе команды $\theta = 00$

результат предыдущей операции), сумматоров и регистров АУ, счетчика ЦУК, счетчика МУК, регистра команд.

На панели сигнализации имеются также дополнительные лампочки, показывающие режим работы, причины останова, код выполняемой операции и т. д.

На пульте управления имеются следующие клавиши и кнопки:

Кнопка «Стирание», при нажатии которой во все разряды всех ячеек ОЗУ записывается 0.

Кнопка «Установка в 0», при нажатии которой производится сброс сумматора и всех важнейших регистров машины.

Переключатель «Род работы», имеющий положения: «Автоматический режим», «Однотактный режим» (в котором машина останавливается после выполнения каждого такта работы машины), «Одиночные импульсы» (служит для наладки машины).

Кнопка «Останов».

Кнопка «Пуск».

Кнопка «Начальный пуск», при нажатии на которую, а затем и на «Пуск», производится ввод 12 кодов с перфокарт. При этом выполняются закоммутированные в машине команды:

37 0000 0000 0000

30 0100 0000 0013

31 0000 0000 0000

34 0000 0000 0001

Кнопки, позволяющие переводить машину с центрального управления на местное и обратно.

Три ряда клавиш (по 11 в каждом) с соответствующими тумблерами разрешения для останова машины по счетчику ЦУК, счетчику МУК и по 3-му адресу. Машина останавливается, если набранный адрес совпадает с содержимым соответствующего устройства.

Два ряда клавиш (по 39 в каждом), на которых можно набирать числа, вводимые в ОЗУ по команде «Пульт» ($\theta = 27$).

Тридцать девять клавиш с кнопкой передачи набранной команды на регистр команд для исполнения.

Подготовка данных. Ввод и вывод

Программа задачи и исходные данные записываются на бланках, а затем переносятся на перфоленту или на перфокарты. На машине *БЭСМ-2* для записи чисел и команд применяют бланки, возможный вид которых показан на рис. 17.2.

Номер ячейки	Код операции	1-й адрес	2-й адрес	3-й адрес	Пояснения
20					
1	77	1252	1252	1252	≈1/3 } Нормализованные
2	04	3200			—10 } числа
3					
4	40				Машинный нуль
5					
6	00	3777			} Константы для преобра-
7	00	3777	3776	0001	} зования команд
30					
1	1 01	058	993	052	+0,058993052 } Десятичные
2	0 06	133	398	787	—333987,87 } числа
3					
4	00	0034		0002	} Команды
5	33				

Рис. 17.2. Образец записи команд и чисел на бланке.

Отметим следующие особенности записи чисел на бланках:

а) знак мантиссы двоичного числа входит в старшую четверичную цифру цифровой части числа;

б) знак мантиссы десятичного числа пишется перед мантиссой в виде двоичной цифры.

На том же рисунке даны примеры, показывающие, как записываются команды и числа на бланках.

На рис. 17.3 и 17.4 даны образцы пробивок команд и десятичных чисел на перфоленте и на перфокарте. На перфоленте в верхнем ряду пробиты маркеры, соответствующие двоичным разрядам. В нижнем ряду пробиваются числа.

На перфокарте каждый код снабжается маркером. Если его нет, то код в машину не вводится. Чтение перфокарт производится с помощью специального шаблона.

На приведенных рисунках скобками ⌒ указаны объединения в восьмеричные и десятичные цифры.

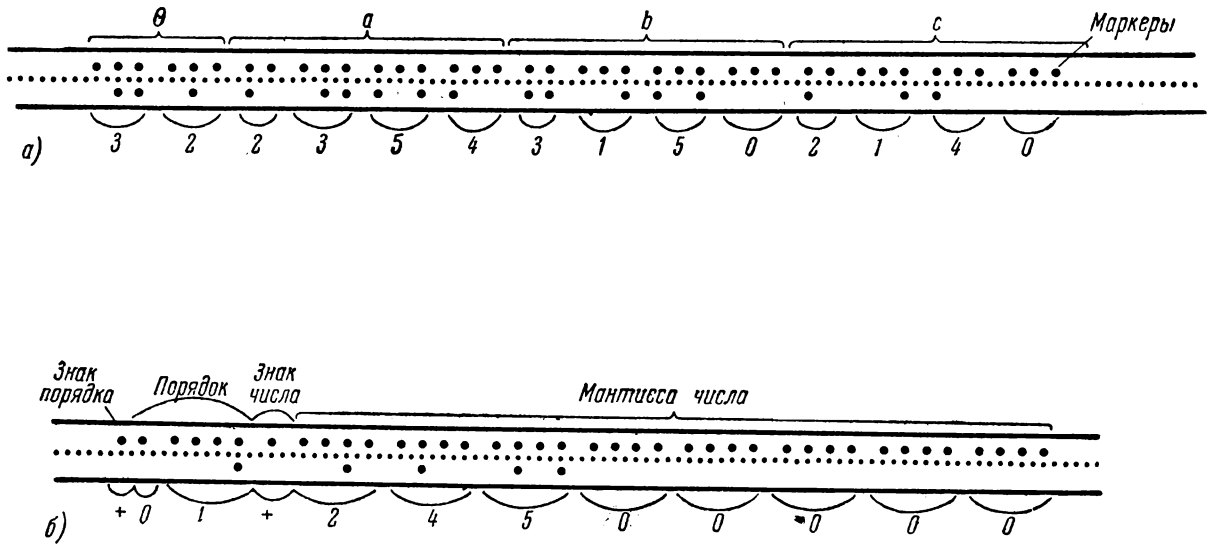


Рис. 17.3. Образцы кодирования на перфоленте: *а*) команды; *б*) двоично-десятичного числа.

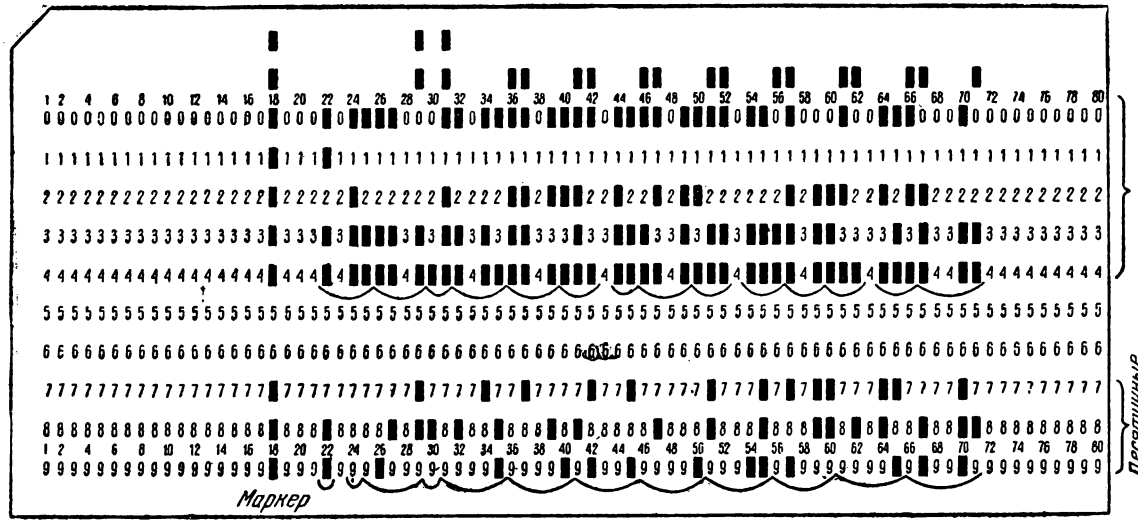


Рис. 17.4. Перфокарта машины *БЭСМ-2*.

Печатающее устройство способно выдавать шестнадцатеричные и десятичные числа. При этом код в ячейке группируется по четыре разряда. Шестнадцати последовательным двоичным числам, которые могут размещаться в каждой группе из четырех разрядов, соответствуют знаки: 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; $\bar{0}$; $\bar{1}$; $\bar{2}$; $\bar{3}$; $\bar{4}$; $\bar{5}$. Содержимое 39 и 33 разрядов печатается в виде знаков «+» и «—». Содержимое 34÷39-го разрядов печатается справа от содержимого 1÷33-го разрядов. Например, десятичное число— $0{,}69657812 \cdot 10^{-3}$ печатается:

— 69657812 — 03

Литература к главе: [3].

БИБЛИОГРАФИЯ

1. Автоматизация программирования, Сб. переводов под ред. А. П. Ершова, Физматгиз, 1961.
2. Айзерман М. А., Гусев Л. А., Розеноэр Л. И., Смирнова И. М., Таль А. А., Логика, автоматы, алгоритмы, Физматгиз, 1963.
3. Базилевич В. Л., Базилевич Л. В., Система команд и программирование для *БЭСМ-2*, «Судостроение», 1964.
4. Бекус Дж. и др., Сообщение об алгорифмическом языке АЛГОЛ 60, Журнал вычисл. матем. и матем. физ., т. 1, № 2, Изд. АН СССР, 1961.
5. Березин И. С., Жидков Н. П., Методы вычислений, Физматгиз, т. I, 1959, т. II, 1960.
6. Беркли Э., Символическая логика и разумные машины, ИЛ, 1961.
7. Биркган А. Ю. и Воскресенский Г. П., Программирование для цифровой вычислительной машины *Урал-2*, «Сов. Радио», 1962.
8. Бондаренко В. Н., Плотников И. Т., Полозов П. П., Программирование задач для машины *Урал*, Изд. Арт. инж. академии им. Дзержинского, 1957.
9. Бусленко Н. П., Шрейдер Ю. А., Метод статистических испытаний, Физматгиз, 1961.
10. Бусленко Н. П., Голенко Д. И., Соболь И. М., Срагович В. Г., Шрейдер Ю. А., Метод статистических испытаний (метод Монте-Карло) СМБ, Физматгиз, 1962.
11. Бухтияров А. М., Зикевская Л. М., Фролов Г. Д., Сборник задач по программированию, «Наука», 1965.
12. Быстродействующая вычислительная машина *М-2*, под ред. И. С. Брука, Гостехиздат, 1957.
13. Вычислительные машины (*СЕАК и ДИСЕАК*), ИЛ, 1958.
14. Вязалов Л. Х. и Морозов Ю. И., Вспомогательная стандартная программа для отладки программы (СПО-1), сб. «Проблемы кибернетики», вып. 6, Физматгиз, 1961.
15. Гильберт Д., Аккерман В., Основы теоретической логики, ИЛ, 1947.
16. Гнеденко Б. В., Королюк В. С., Ющенко Е. Л., Элементы программирования, Физматгиз, 1961.
17. Глушков В. М., Об одном методе автоматизации программирования, сб. «Проблемы кибернетики», вып. 2, Физматгиз, 1959.

18. Глушков В. М., Дашевский Л. Н., Ющенко Е. Л. и др., Электронная вычислительная машина *Киев*, сб. Трудов конференции ВЦ АН УССР, 1958.
19. Глушков В. М., Синтез цифровых автоматов, Физматгиз, 1962.
20. Гудстейн Р. Л., Математическая логика, ИЛ, 1961.
21. Гутенмахер Л. И., Электронные информационно-логические машины, Изд. АН СССР, 1960.
22. Даугавет О. К., Озерова Е. Ф., Программирующая программа компилирующего типа, Журнал вычисл. матем. и матем. физ., т. 1, № 4, Изд. АН СССР, 1961.
23. Демидович Б. П., Марон И. А., Шувалова Э. Э., Численные методы анализа, Физматгиз, 1962.
24. Ершов А. П., Программирующая программа для быстродействующей электронной счетной машины, Изд. АН СССР, 1958.
25. Ершов А. П., Ким К. В., Программирующая программа для вычислительной машины *Стрела-3* (ППС). Изд. ВЦ АН СССР, 1961.
26. Ефимова М. Н., Алгоритмические языки, «Сов. радио», 1965.
27. Жоголев Е. А., Система команд и интерпретирующая система для машины *Сетунь*, Журнал вычисл. матем. и матем. физ., т. 1, № 3, Изд. АН СССР, 1961.
28. Жоголев Е. А., Трифонов Н. П., Курс программирования «Наука», 1964.
29. Зарипов Р. Х., О программировании процесса сочинения музыки, сб. «Проблемы кибернетики», вып. 7, Физматгиз, 1962.
30. Илзиня И. Г., Программирование для двухадресных цифровых вычислительных машин, Изд. АН Латв. ССР, 1962.
31. Калужнин Л. А., Об алгорифмизации математических задач, сб. «Проблемы кибернетики», вып. 2, Физматгиз, 1959.
32. Камынин С. С., Любимский Э. З., Шура-Бура М. Р., Об автоматизации программирования при помощи программирующей программы, сб. «Проблемы кибернетики», вып. 1, Физматгиз, 1958.
33. Канторович Л. В., О проведении численных и аналитических вычислений на машинах с программным управлением, Изд. АН Арм. ССР, № 2, 3—16, 1957.
34. Карцев М. А., Арифметические устройства электронных цифровых машин, Физматгиз, 1958.
35. Китов А. И., Электронные цифровые машины, «Сов. Радио», 1956.
36. Китов А. И., Криницкий Н. А., Электронные вычислительные машины, изд. 2-е, «Наука», 1965.
37. Китов А. И., Криницкий Н. А., Электронные цифровые машины и программирование, изд. 2-е, Физматгиз, 1961.
38. Китов А. И., Криницкий Н. А., Комолов П. Н., Элементы программирования, Изд. Арт. инж. академии им. Дзержинского, 1956.
39. Криницкий Н. А., Язык логических схем, сб. «Цифровая вычислительная техника и программирование», «Сов. Радио», 1966.
40. Клини С. К., Введение в метаматематику, ИЛ, 1957.

41. Королюк В. С., О понятии адресного алгоритма, сб. «Проблемы кибернетики», вып. 4, Физматгиз, 1960.
42. Кулагина О. С., Об операторном описании алгорифмов перевода и автоматизации процесса их программирования, сб. «Проблемы кибернетики», вып. 2, Физматгиз, 1959.
43. Кулагина О. С., Вакуловская Г. В., Опытные переводы с французского языка на русский на машине *Стрела*, сб. «Проблемы кибернетики», вып. 2, Физматгиз, 1959.
44. Лавров С. С., Универсальный язык программирования, «Наука», 1964.
45. Лебедев С. А., Мельников В. А., Общее описание *БЭСМ* и методика выполнения операций, Физматгиз, 1959.
46. Люстерник Л. А., О вычислении функций одного переменного, сб. «Математическое просвещение», вып. 3, 4, Физматгиз, 1958.
47. Люстерник Л. А., Абрамов А. А., Шестаков В. М., Шура-Бура М. Р., Решение математических задач на автоматических цифровых машинах, Изд. АН СССР, 1952.
48. Ляпунов А. А., Математические исследования, связанные с эксплуатацией электронных вычислительных машин, сб. «Математика в СССР за 40 лет», т. 1, Физматгиз, 1959.
49. Ляпунов А. А., О логических схемах программ, сб. «Проблемы кибернетики», вып. 1, Физматгиз, 1958.
50. Мак-Кракен Дж., Программирование на АЛГОЛе, «Мир», 1964.
51. Марков А. А., Теория алгорифмов, Труды математического института имени В. А. Стеклова, т. XLII, Изд. АН СССР, 1954.
52. Мартынюк В. В., О методе символических адресов, сб. «Проблемы кибернетики», вып. 6, Физматгиз, 1961.
53. Машинный перевод, Сб. переводов под ред. Т. Н. Молошной и В. А. Пурто, ИЛ, 1957.
54. Мельчук И. А., О машинном переводе с венгерского языка на русский, сб. «Проблемы кибернетики», вып. 1, Физматгиз, 1958.
55. Миронов Г. А., Испытательные программы для контроля электронных цифровых машин, «Наука», 1964.
56. Морозов Ю. И., Петров А. А., Программирующая система компилирующего типа (ПСК-1), сб. «Проблемы кибернетики», вып. 5, Физматгиз, 1961.
57. Нечаев А., Устройство и работа электронных цифровых машин, «Московск. рабоч.», 1964.
58. Новиков П. С., Элементы математической логики, Физматгиз, 1959.
59. Первин Ю. А., Об алгоритмизации и программировании игры в домино, сб. «Проблемы кибернетики», вып. 3, Физматгиз, 1960.
60. Поддерюгин В. Д., Программа контроля для *Стрелы-3* (ЛУЧ), Изд. ВЦ АН СССР, 1961.
61. Поддерюгин В. Д., Программа контроля для *Стрелы-3* (ПИК), Изд. ВЦ АН СССР, 1961.
62. Подловченко Р. И., О преобразованиях схем программ и их применении в программировании, сб. «Проблемы кибернетики», вып. 7, Физматгиз, 1962.

63. Ричардс Р. К., Арифметические операции на цифровых вычислительных машинах, ИЛ, 1957.
64. Ричардс Р. К., Элементы и схемы цифровых вычислительных машин, ИЛ, 1961.
65. Система автоматизации программирования, под ред. Н. П. Трифонова и М. Р. Шура-Бура, «Библиотека прикладного анализа и вычислительной математики», Физматгиз, 1961.
66. Система стандартных подпрограмм, под ред. М. Р. Шура-Бура, «Библиотека прикладного анализа и вычислительной математики», Физматгиз, 1958.
67. Смирнова Т. Н., Полиномиальный прораб на электронную вычислительную машину *Стрела*, Журнал вычисл. матем. и матем. физ., т. 1, № 5, Изд. АН СССР, 1961.
68. Смольников Н. Я., Основы программирования для цифровой машины *Урал*, «Сов. радио», 1961.
69. Трахтенброт Б. А., Алгорифмы и машинное решение задач, Физматгиз, 1957.
70. Успенский В. А., Лекции о вычислимых функциях, Физматгиз, 1960.
71. Шура-Бура М. Р., Программирование, сб. «Математика в СССР за 40 лет», т. 1, Физматгиз, 1959.
72. Ющенко Е. Л., Адресное программирование, Гостехиздат УССР, 1963.
73. Ющенко Е. Л., Малиновский Б. Н., Полищук Г. А., Ядренко Э. К., Никитина А. И., Управляющая машина широкого назначения *Днипро*, и программирующая программа к ней, «Наукова думка», Киев, 1964.
74. Янов Ю. И., О логических схемах алгоритмов, сб. «Проблемы кибернетики», вып. 1, Физматгиз, 1958.

Николай Андреевич Криницкий,
Георгий Акимович Миронов,
Геннадий Дмитриевич Фролов

ПРОГРАММИРОВАНИЕ

М., 1966 г., 600 стр. с илл.

Редактор *М. М. Горячая.*
Техн. редактор *С. Я. Шкляр.*
Корректор *Н. Д. Дорохова.*

Сдано в набор 5/I 1966 г. Подписано к печати 25/V 1966 г. Бумага 84×108$^{1}/_{32}$. Физ. печ. л. 18,75. Условн. печ. л. 31,5. Уч.-изд. л. 28.46. Тираж 25 000 экз. Т-08210. Цена книги 1 р. 57 к. Заказ № 151.

Издательство «Наука».
Главная редакция
физико-математической литературы.
Москва, В-71, Ленинский проспект, 15.

Ленинградская типография № 1 «Печатный Двор» имени А. М. Горького Главполиграфпрома Комитета по печати при Совете Министров СССР, Гатчинская, 26.

Цена 1 р. 57 к.

СПРАВОЧНАЯ МАТЕМАТИЧЕСКАЯ БИБЛИОТЕКА

1. В. Л. ДАНИЛОВ, А. Н. ИВАНОВА и др., Математический анализ (функции, пределы, ряды, цепные дроби).

2. И. Г. АРАМАНОВИЧ, Р. С. ГУТЕР и др., Математический анализ (дифференцирование и интегрирование).

3. Л. А. ЛЮСТЕРНИК и др., Математический анализ (вычисление элементарных функций).

4. А. П. МИШИНА и И. В. ПРОСКУРЯКОВ, Высшая алгебра (линейная алгебра, многочлены, общая алгебра).

5. В. А. ДИТКИН и А. П. ПРУДНИКОВ, Интегральные преобразования и операционное исчисление.

6. Н. П. БУСЛЕНКО и др., Метод статистических испытаний (метод Монте-Карло).

7. Н. А. КРИНИЦКИЙ и др., Программирование.

8. Р. С. ГУТЕР, Л. Д. КУДРЯВЦЕВ, Б. М. ЛЕВИТАН, Элементы теории функций (функции действительного переменного, приближение функций, почти-периодические функции).

9. Н. Я. ВИЛЕНКИН и др., Функциональный анализ.

10. В. М. БАБИЧ и др., Линейные уравнения математической физики.

11. С. Г. МИХЛИН и Х. Л. СМОЛИЦКИЙ, Приближенные методы решения дифференциальных и интегральных уравнений.

12. Г. БЕЙТМЕН и А. ЭРДЕЙИ, Высшие трансцендентные функции (гипергеометрическая функция, функции Лежандра).

13. Г. БЕЙТМЕН и А. ЭРДЕЙИ, Высшие трансцендентные функции (функции Бесселя, функции параболического цилиндра, ортогональные многочлены).