Б.А.Калабеков

МИКРОПРОЦЕССОРЫ И ИХ ПРИМЕНЕНИЕ В СИСТЕМАХ ПЕРЕДАЧИ И ОБРАБОТКИ СИГНАЛОВ

Допущено
Министерством высшего и среднего
специального образования СССР
в качестве учебного пособия
для студентов радиотехнических
спецнальностей вузов

Москва
«Радио и связь»
1988

ББК 32.97
К17
УДК 681.325.5-181.4(075)

Калабеков Б. А.

К17 Микропроцессоры и их применение в системах передачи и обработки сигналов: Учеб. пособие для вузов. — М.: Радио и связь, 1988. — 368 с.: ил.

ISBN 5-256-00169-8

Излагаются различные подходы к проектированию процессорных устройств: со специализированным операционным устройством и управляющим устройством, синтезированным на основе схемной и программируемой логики. Рассматриваются устройства на микропроцессорных комплектах БИС серий КР580, К589, КР1804, КР1810, а также приемы программирования на языках различных уровней.

Для студентов радиотехнических специальностей, полезна специалистам, связанным с разработкой микропроцессорных устройств.

$\text{К}\frac{2405000000\text{-}024}{046(01)\text{-}88}$ 154-88 ББК 32.97

Рецензенты: кафедра вычислительной техники МИЭТ (зав. кафедрой чл.-корр. АН СССР Л. Н. Преснухин), чл.-корр. АН УССР К. Г. Самофалов

Редакция литературы по вычислительной технике

Учебное пособие

Калабеков Вениамин Аршакович

МИКРОПРОЦЕССОРЫ И ИХ ПРИМЕНЕНИЕ В СИСТЕМАХ ПЕРЕДАЧИ И ОБРАБОТКИ СИГНАЛОВ

Заведующий редакцией *Г. И. Козырева.* Редактор *Т. М. Бердичевская*
Художественный редактор *Н. С. Шеин.* Переплет художника *Ю. В. Архангельского*
Технический редактор *Т. Н. Зыкина.* Корректор *Н. Л. Жукова*

ИБ № 1415

Сдано в набор 5.06.87. Подписано в печать 22.10.87. Т-19032 Формат 60×88$^1/_{16}$. Бумага офсетная № 2. Гарнитура литературная. Печать офсетная. Усл. печ. л. 22,54. Усл. кр.-отт. 22,54. Уч.-изд. л. 22,96. Тираж 80 000 экз. (1 з. — 1—40.000 экз.). Изд. № 21615. Зак. № 430. Цена 1 р. 10 к.

Издательство «Радио и связь». 101000, Москва, Почтамт, а/я 693

Московская типография № 4 Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли.
Москва, 129041, Б. Переяславская, 46

ISBN 5-256-00169-8

ПРЕДИСЛОВИЕ

Широкая автоматизация процессов в сферах производства, научных исследований, эксплуатации оборудования с использованием средств вычислительной техники является основным направлением интенсификации физического и интеллектуального труда человека, повышения производительности труда. Решение этой задачи требует подготовки инженеров в области автоматизации проектирования (устройств, систем, сетей), научных исследований (при моделировании процессов, сборе и обработке данных эксперимента, планировании эксперимента, постановке машинного эксперимента и т.д.) и управления (управления технологическими процессами, административно-организационного управления).

Основной технической базой автоматизации управления технологическими процессами являются специализированные микропроцессорные устройства (МПУ). Они являются предметом изучения на этапе базовой подготовки, предшествующей рассмотрению в специальных дисциплинах различных применений МПУ.

При изучении специализированных МПУ рассматриваются приемы проектирования как аппаратных, так и программных средств МПУ. Проектирование аппаратных средств требует знания особенностей микропроцессорных комплектов микросхем различных серий и функциональных возможностей микросхем, входящих в состав используемого комплекта, умения правильно выбрать серию. Проектирование программных средств требует знаний, необходимых для выбора метода и алгоритма решения задач, входящих в функции МПУ, для составления программы (часто с использованием языков низкого уровня — языка кодовых комбинаций, языка Ассемблера), а также умения использовать средства отладки программ. Всем этим вопросам и посвящено данное учебное пособие.

Изложение начинается с рассмотрения арифметических основ цифровой техники, необходимых для понимания последующего материала без обращения к другим источникам. Приводятся начальные сведения о методах построения процессорных устройств без использования БИС микропроцессорных комплектов на принципах схемной и программиру-

емой логики. Этот материал позволит расширить представления о способах построения цифровых устройств обработки данных. В последующих разделах книги достаточно подробно изложены приемы проектирования МПУ на немикропрограммируемых микропроцессорных комплектах серий КР580, КР1810, а также МПУ на микропрограммируемых микропроцессорных комплектах серий К589 и КР1804. Даются основные представления о средствах, используемых при отладке аппаратных и программных средств МПУ. Книга завершается рассмотрением ориентированных на реализацию в микропроцессорных устройствах алгоритмов обработки данных, наиболее часто встречающихся при решении радиотехнических задач.

В пособии рассматриваются МПУ на основе универсальных микропроцессорных комплектов (МПК). Вопросы использования специализированных МПК, ориентированных на цифровую обработку сигналов, не освещены в связи с тем, что их изложение при ограниченном объеме книги потребовало бы сокращения представленного в ней материала; по мнению автора, это сделало бы его более трудным для усвоения.

ВВЕДЕНИЕ

В.1. СИСТЕМЫ СЧИСЛЕНИЯ

Для представления чисел в цифровых устройствах, а также для представления разнообразной информации в процессе программирования наряду с привычной для нас десятичной системой счисления широко используются другие. Рассмотрим принцип построения наиболее употребительных позиционных систем счисления.

Числа в таких системах счисления представляются последовательностью цифр (цифр разрядов), разделенных запятой на две группы: группу разрядов, изображающую целую часть числа, и группу разрядов, изображающую дробную часть числа:

$$\ldots\ a_2a_1a_0,\ a_{-1}a_{-2}\ \ldots \qquad \text{(В.1)}$$

Здесь a_0, a_1, ... — цифры нулевого, первого и т. д. разрядов целой части числа, a_{-1}, a_{-2}, ... — цифры первого, второго и т. д. разрядов дробной части числа.

Единице каждого разряда приписан определенный вес p^k, где p — основание системы счисления, k — номер разряда, равный индексу при буквах, изображающих цифры разрядов. Так, представленная выражением (В.1) запись означает следующее колнчество:

$$N = \ldots + a_2\cdot p^2 + a_1\cdot p^1 + a_0\cdot p^0 + a_{-1}\cdot p^{-1} + a_{-2}\cdot p^{-2}\ \ldots$$

Для представления цифр разрядов используется набор из p различных символов. Так, при $p = 10$ (т. е. в обычной десятичной системе счисления) для записи цифр разрядов используется набор из десяти символов: 0, 1, 2, ..., 9. При этом запись числа $729{,}324_{10}$ (здесь и в дальнейшем индекс при числе будет указывать основание системы счисления, в которой представлено число) означает следующее количество:

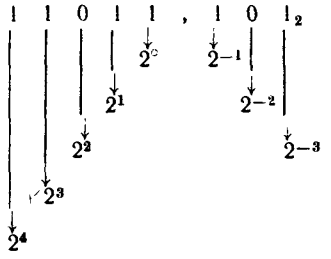
$$7\ 2\ 9\ ,\ 3\ 2\ 4_{10} = 7\cdot 10^2 + 2\cdot 10^1 + 9\cdot 10^0 + 3\cdot 10^{-1} + 2\cdot 10^{-2} + 4\cdot 10^{-3}.$$

весовые коэффициенты разрядов

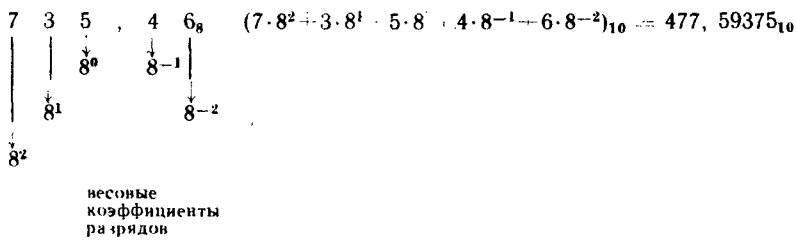
Используя такой принцип представления чисел, но выбирая различные значения основания p, можно строить разнообразные системы счисления.

Двоичная система счисления. Основание системы счисления $p = 2$. Таким образом, для записи цифр разрядов требуется набор всего лишь из двух символов, в качестве которых используются 0 и 1. Следовательно, в двоичной системе счисления число представляется последовательностью символов 0 и 1.

При этом запись $11011,\ 101_2$ соответствует в десятичной системе счисления следующему числу:



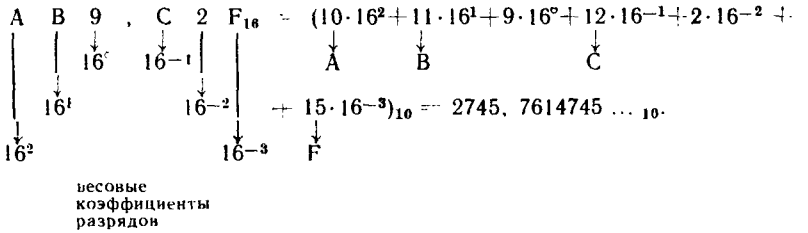
Восьмеричная система счисления. Основание системы счисления $p = 8$. Следовательно, для представления цифр разрядов должно использоваться восемь различных символов, в качестве которых выбраны 0,1, 2,...,7 (заметим, что символы 8 и 9 здесь не используются и в записи чисел встречаться не должны). Например, записи $735,\ 46_8$ в десятичной системе счисления соответствовало бы следующее число:



т. е. запись $735,\ 46_8$ означает число, содержащее 7 раз по $8^2 = 64$, 3 восьмерки, 5 единиц, 4 раза по $8^{-1} = 1/8$ и 6 раз по $8^{-2} = 1/64$.

Шестнадцатеричная система счисления. Основание системы счисления $p = 16$ и для записи цифр разрядов должен использоваться набор из 16 символов: 0,1, 2, ..., 9, A, B, C, D, E, F. В нем используются 10 арабских цифр, и до требуемых шестнадцати их дополняют шестью начальными буквами латинского алфавита. При этом символ A соответствует количеству, в десятичной системе счисления равному 10, B — 11, C — 12, D — 13, E — 14 и F — 15.

При этом запись AB9, $C2F_{16}$ соответствует следующему числу в десятичной системе счисления:



Хранение n-разрядных чисел в цифровой аппаратуре можно осуществить с помощью устройств, содержащих n элемеитов, каждый из которых запоминает цифру соответствующего разряда числа. Наиболее просто осуществляется хранение чисел, представленных в двоичной системе счисления. Для запоминаиия цифры каждого разряда двоичного числа могут использоваться устройства с двумя устойчивыми состояниями (например, триггеры). Одиому из этих устойчивых состояний ставится в соответствие цифра 0, другому — цифра 1.

При хранении десятичных чисел каждая цифра десятичного числа предварительно представляется в двоичной форме. Такая форма представления чисел носит название двоично-кодированной десятичиой системы. Например, число $765{,}93_{10}$ в двоично-коднрованной десятичной системе представляется в следующем виде:

$$765{,}93_{10} \quad \underbrace{011101100101}_{7\ 6\ 5},\ \underbrace{10010011}_{9\ 3}{}_{2-10}.$$

Запоминание цифр 0 и 1 двоичио-кодироваиного десятичного числа может осуществляться с помощью указанных выше элементов с двумя устойчивыми состояниями.

Следует заметить, что несмотря на виешиее сходство двоичио-кодированиого десятичного числа (содержащего в разрядах лишь цифры 0 и 1) с двоичным числом, оно не является двоичным. В этом легко убедиться. Например, если целую часть приведенной выше записи в правой части равеиства рассматривать как двоичное число, то оно при переводе в десятичную систему счисления означало бы 1893_{10}, что не совпадает с целой частью исходиого числа 765.

Рассмотренный способ двоичного представления (кодирования) десятичиых цифр использует так называемый *код 8421* (названиие кода составлено из весовых коэффициентов разрядов в двоичной системе счисления).

В.2. ПЕРЕВОД ЧИСЕЛ ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ

Основания восьмеричной и шестнадцатеричной систем счисления выражаются целой степенью двух ($8=2^3$; $16=2^4$). Этим объясняется простота преобразования чисел между этими системамн и двоичной системой счисления.

Для перевода чисел из восьмеричиой системы счисления в двоичиую достаточио каждую цифру восьмеричного числа заменить соответствующим 3-разрядным двоичным числом. Например,

$$735{,}24_8 = \underbrace{111011101}_{7\ 3\ 5},\ \underbrace{010100}_{2\ 4}{}_2.$$

Перевод в двоичиую систему счисления шестнадцатеричных чисел достигается заменой цифр шестнадцатеричного представления 4-разрядными двоичными числами. Например,

$$A3B{,}C9_{16} = \underbrace{101000111011}_{A\ 3\ B},\ \underbrace{11001001}_{C\ 9}{}_2.$$

При обратном переводе чисел из двоичной системы в восьмеричную или шестнадцатеричную систему счисления необходимо разряды двоичного числа, отсчитывая их от запятой влево и вправо, разбить на группы по три разряда (в случае перевода в восьмеричную систему) или на группы по четыре разряда (в случае перевода в шестнадцатеричную систему счисления). Неполные крайние группы дополняются до полных нулями. Затем каждая двоичная группа представляется цифрой той системы счисления, в которую переводится число. Например,

$$\underset{1}{001}\,\underset{7}{111},\ \underset{5}{101}\,\underset{2}{010_2} \quad 17{,}52_8;$$

$$\underset{5}{0101}\,\underset{C}{1100},\ \underset{B}{1011}\,\underset{6}{0110_2} \quad 5C{,}B6_{16}.$$

Бо́льшую сложность представляет перевод чисел между десятичной и двоичной системами счисления. Метод, используемый для такого перевода, зависит от того, в какой системе счисления представлены числа, над которыми проводятся необходимые для перевода чисел арифметические операции. Если перевод производится человеком, то, очевидно, операции будут выполняться над числами в десятичной системе счисления; если же перевод осуществляется цифровым устройством, арифметические операции удобнее выполнять над числами в двоичной системе счисления. Рассмотрим эти два случая.

Перевод чисел с выполнением операций над десятичными числами. Так как преобразование чисел между двоичной и шестнадцатеричной системами счисления не представляет труда, то для простоты выкладок будем в дальнейшем рассматривать перевод чисел между шестнадцатеричной и десятичной системами счисления.

Пусть требуется перевести число из шестнадцатеричной в десятичную систему счисления. В качестве примера выберем число 9A5F, C83B$_{16}$. Учитывая веса разрядов шестнадцатеричной системы счисления, запишем значение этого числа в десятичной системе счисления:

$$9A5F,C83B_{16} = \underbrace{(9\cdot 16^3+\underset{\downarrow}{10}\cdot 16^2+5\cdot 16^1+\underset{\downarrow}{15}\cdot 16^0}_{\text{целая часть}}\ +$$

$$\underbrace{+\underset{\downarrow}{12}\cdot 16^{-1}+8\cdot 10^{-2}+3\cdot 16^{-3}+\underset{\downarrow}{11}\cdot 16^{-4})_{10}}_{\text{дробная часть}} =$$

$$= \underbrace{(((9\cdot 16+10)\cdot 16+5)\cdot 16+15}_{\text{целая часть}}+\underbrace{16^{-1}\cdot(12+16^{-1}(8+16^{-1}(3+16^{-1}\cdot 11))))_{10}}_{\text{дробная часть}}.$$

Здесь путем группировки членов вычисление полиномов представлено в форме так называемой схемы Горнера, удобной для программирования и обеспечивающей минимальное число выполняемых операций умножения.

Вычисления в приведенном примере дают следующий результат:

$$9A5F,\ C83B_{16} \approx 39519,\ 7821502_{10}.$$

Целая часть числа преобразуется точно, дробная часть — приближенно. В приведенном примере вычисления при нахождении дробной части выполнялись с точностью, определяемой семью десятичными разрядами.

Рассмотрим обратный перевод чисел из десятичной в шестнадцатеричную систему счисления. Воспользуемся приведенным выше примером. Теперь будем считать заданным десятичное число 39519, 7821502$_{10}$ и будем искать его представ-

ление в шестнадцатеричной системе счисления. Рассмотрим преобразование целой части числа. Из равенства

$$39519_{10} = ((9\cdot 16+10)\cdot 16+5)\cdot 16+15$$
$$\qquad\qquad\qquad\quad \downarrow \qquad\qquad \downarrow$$
$$\qquad\qquad\qquad\quad A \qquad\qquad F$$

можно вывести следующее правило получения цифр шестнадцатеричного представления. Деление правой части равенства (т. е. целой части заданного числа) на 16 дает в частном $(9\cdot 16+10)\cdot 16+5$ и в остатке 15 (т. е. F); деление полученного частного на 16 даст частное $9\cdot 16+10$ и остаток 5; деление последнего частного приведет к частному 9 и остатку 10 (т. е. A). Таким образом, последовательно деля на 16 целую часть десятичного числа и образующиеся частные, получаем в последнем частном и остатках цифры всех разрядов шестнадцатеричного представления целой части числа. Покажем эти действия по преобразованию десятичного числа 39519_{10} в шестнадцатеричную систему счисления:

39519	16		
39504	2469	16	
15	2464	154	16
F	5	144	9
		10	
		A	

Отсюда $39519_{10} = 9A5F_{16}$.

Теперь рассмотрим преобразование дробной части десятичного числа в шестнадцатеричную систему счисления. Из равенства

$$0{,}7821502_{10} = 16^{-1}\cdot(12+16^{-1}\cdot(8+16^{-1}\cdot(3+16^{-1}\cdot 11)))$$
$$\qquad\qquad\qquad\quad \downarrow \qquad\qquad\qquad\qquad\quad \downarrow$$
$$\qquad\qquad\qquad\quad C \qquad\qquad\qquad\qquad\quad B$$

следует, что для получения цифр разрядов дробной части шестнадцатеричного числа $(0{,}C83B_{16})$ необходимо последовательно умножать на 16 дробную часть исходного десятичного числа и дробные части образующихся произведений. При этом целые части этих произведений являются цифрами шестнадцатеричного представления:

	$\times$ 0 , 7821502
	16
C ⟶	$\times$ 12 , 5144032
	16
8 ⟶	$\times$ 8 , 2304512
	16
3 ⟶	$\times$ 3 , 6872192
	16
A ⟶	$\times$ 10 , 9955072
	16
F ⟶	$\times$ 15 , 9281152
	16
E ⟶	14 , 8498432
	. . . .

Таким образом, $0{,}7821502_{10} = 0{,}C83AFE\ldots_{16} \approx 0{,}C83B_{16}$. И в этом случае убеждаемся, что дробные числа преобразуются неточно.

Перевод чисел с выполнением операций в двоичной системе счисления. Рассмотрим перевод десятичных чисел в двоичную систему счисления. Для иллюстрации метода перевода выберем десятичное число 937, 568_{10}, которое представим в следующей форме:

$$937{,}568_{10} = (9\cdot10^2+3\cdot10^1+7\cdot10^0+5\cdot10^{-1}+6\cdot10^{-2}+8\cdot10^{-3})_{10} =$$
$$= \underbrace{(9\cdot10+3)\cdot10+7}_{\text{целая часть}} \quad \underbrace{+10^{-1}\cdot(5+10^{-1}\cdot(6+10^{-1}\cdot8))}_{\text{дробная часть}}.$$

Представив числа, входящие в правую часть равенства, 4-разрядными двоичными числами, запишем выражения для преобразования целой и дробной частей:

$$937_{10} = ((\underbrace{1001}_{9}\cdot\underbrace{1010}_{10}+\underbrace{0011}_{3})\cdot\underbrace{1010}_{10}+\underbrace{0111}_{7})_2,$$

$$0{,}568_{10} = (((\underbrace{1000}_{8}:\underbrace{1010}_{10}+\underbrace{0110}_{6}):\underbrace{1010}_{10}+\underbrace{0101}_{5}):\underbrace{1010}_{10})_2.$$

Получаемые в результате выполнения операций над двоичными числами значения представляют собой двоичные представления соответственно целой и дробной частей исходного числа.

Рассмотрим обратный перевод двоичных чисел в десятичную систему счисления. Перевод целых двоичных чисел производится последовательным делением в двоичной системе счисления на число 1010_2 исходного двоичного и всех образующихся частных. При этом последнее частное и возникающие при делении остатки являются двоичными представлениями цифр разрядов искомого десятичного представления числа.

Перевод дробного двоичного числа производится последовательным умножением на двоичное число 1010_2 исходного числа и дробных частей получаемых произведений. При этом целые части произведений являются двоичным представлением цифр разрядов искомого десятичного представления дробного числа.

Преобразование чисел с помощью сдвиговых регистров. Рассмотрим преобразование двоичных чисел в десятичную систему счисления. Пусть число, подлежащее преобразованию в десятичную систему счисления, хранится в регистре R_1 (рис. В.1). Результат преобразования (число в десятичной системе счисления) будем формировать в регистре R_2. Разряды регистра R_2 делятся на 4-разрядные группы R_2', R_2'', R_2''' и т. д., каждая из которых предназначена для хранения одной десятичной цифры, представленной в двоичной системе счисления.

Рассматриваемый способ преобразования потребует выполнения последовательности операций сдвига влево содержимого регистров R_1 и R_2 с передачей выдвигаемого из регистра R_1 содержимого старшего разряда в освобождающийся младший разряд регистра R_2. Двоичное число, выдвигаясь из регистра R_1, будет вдвигаться в регистр R_2. При этом необходимо учитывать следующую особенность выполнения сдвигов в регистре R_2. Единица, выдвигаемая при сдвиге из старшего разряда группы R_2', имеет вес $2^4 = 16$. Однако поступая в группу R_2'' (в раз-

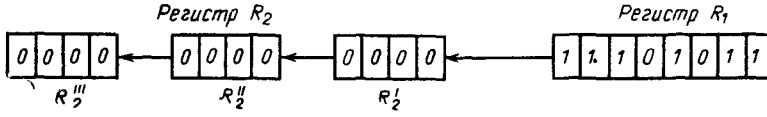


Рис. В.1. Схема преобразования чисел из двоичной в двоично-кодированную десятичную систему счисления

Таблица В.1

R_2'''	R_2''	R_2'	R_1	
0 0 0 0	0 0 0 0	0 0 0 0	1 1 1 0 1 0 1 1	исходное состояние
0 0 0 0	0 0 0 0	0 0 0 1	1 1 0 1 0 1 1	сдвиг
0 0 0 0	0 0 0 0	0 0 1 1	1 0 1 0 1 1	сдвиг
0 0 0 0	0 0 0 0	0 1 1 1	0 1 0 1 1	сдвиг
		+ 0 0 1 1		
		1 0 1 0		коррекция
0 0 0 0	0 0 0 1	0 1 0 0	1 0 1 1	сдвиг
0 0 0 0	0 0 1 0	1 0 0 1	0 1 1	сдвиг
		+ 0 0 1 1		
		1 1 0 0		коррекция
0 0 0 0	+ 0 1 0 1	+ 1 0 0 0	1 1	сдвиг
	0 0 1 1	0 0 1 1		
	1 0 0 0	1 0 1 1		коррекция
0 0 0 1	0 0 0 1	0 1 1 1	1	сдвиг
		+ 0 0 1 1		
		1 0 1 0		коррекция
0 0 1 0	0 0 1 1	0 1 0 1		сдвиг
2	3	5		

ряд десятков), эта единица будет иметь вес 10. Таким образом, при передаче единицы из R_2' в R_2'' происходит потеря 6 единиц. Для компенсации этой потери потребуется прибавить 6 единиц к содержимому R_2'. Можно показать, что выдвижение единицы из любой 4-разрядной группы регистра R_2 требует коррекции содержимого этой группы путем прибавления 6 единиц. Такая же коррекция требуется и в случае, когда после сдвига в 4-разрядной группе возникает число, большее или равное 10. В этом случае прибавление 6 единиц вызывает перенос из старшего разрида группы, который необходимо прибавлить к содержимому следующей 4-разрядной группы.

Более удобным оказывается способ, при котором коррекция производится не после сдвига, а до выполнения сдвига влево. В этом случае коррекция осуществляется прибавлением числа 3 (в результате сдвига оно удваивается и принимает значение 6), а признаком необходимости коррекции является наличие в 4-разрядной группе числа, большего или равного 5 (после сдвига это число оказывается большим или равным 10). При этом передача единицы в следующую 4-разрядную группу осуществляется только путем передачи переноса, возникающего в процессе сдвига (т. е. исключается необходимость прибавления единицы к содержимому группы, как в случае, когда коррекция выполняется после операции сдвига).

Если число разрядов регистра R_1 равно n, то преобразование завершается после n-кратного выполнения сдвига.

В табл. В.1 показан процесс преобразования числа $N=11101011_2$ в десятичное представление 235_{10}.

На рис. В.2 представлена схема алгоритма преобразования рассмотренным способом чисел из двоичной системы счисления в десятичную.

Рассмотрим обратное преобразование числа из десятичной системы счисления в двоичную. Очевидно, такое преобразование может быть осуществлено при использовании описанных выше действий, выполняемых в обратном порядке: осуществляется серия сдвигов вправо содержимого регистров R_2 и R_1 (рис. В.3) с коррекцией результата после каждого сдвига. Коррекция выполняется путем вычитания трех единиц, если содержимое 4-разрядной группы окажется больше

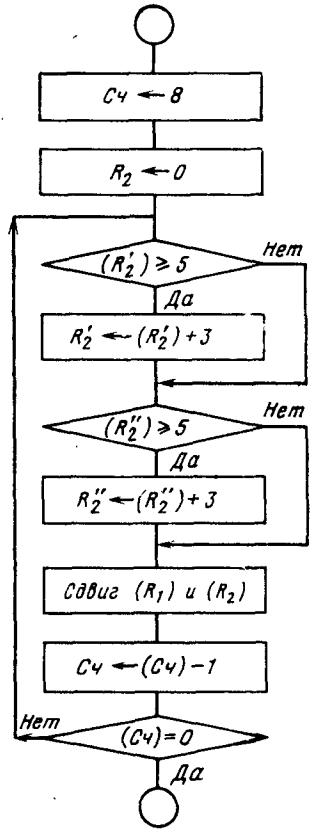


Рис. В.2. Схема алгоритма преобразования чисел из двоичной в десятичную систему счисления

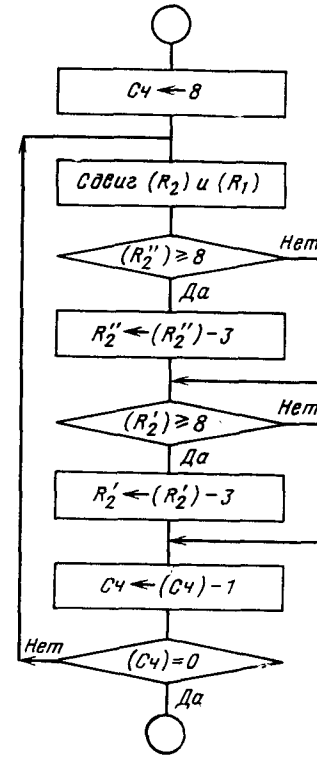


Рис. В.4. Схема алгоритма преобразования чисел из десятичной в двоичную систему счисления

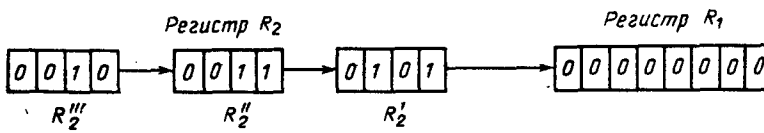


Рис. В.3. Схема преобразоваиия чисел из двоично-кодированиой десятичной в двоичную систему счисления

Таблица В.2

R_2'''	R_2''	R_2'	R_1	
0 0 1 0	0 0 1 1	0 1 0 1		исходное состояние
0 0 0 1	0 0 0 1	− 1 0 1 0	1	сдвиг
		0 0 1 1		
		0 1 1 1		коррекция
0 0 0 0	− 1 0 0 0	− 1 0 1 1	1 1	сдвиг
	0 0 1 1	0 0 1 1		
	0 1 0 1	1 0 0 0		коррекция
0 0 0 0	0 0 1 0	− 1 1 0 0	0 1 1	сдвиг
		0 0 1 1		
		1 0 0 1		коррекция
0 0 0 0	0 0 0 1	0 1 0 0	1 0 1 1	сдвиг
0 0 0 0	0 0 0 0	− 1 0 1 0	0 1 0 1 1	сдвиг
		0 0 1 1		
		0 1 1 1		коррекция
0 0 0 0	0 0 0 0	0 0 1 1	1 0 1 0 1 1	сдвиг
0 0 0 0	0 0 0 0	0 0 0 1	1 1 0 1 0 1 1	сдвиг
0 0 0 0	0 0 0 0	0 0 0 0	1 1 1 0 1 0 1 1	сдвиг

или равно 8. Поясним необходимость в такой коррекции. Указанное условие выполняется, если происходит передача единицы из младшего разряда соседней слева группы в старший разряд данной 4-разрядной группы. При этом, если, например, в процессе сдвига передается единица из R_2'' в R_2', то ее вес в R_2'' был равен 10, а в результате сдвига его значение должно быть уменьшено в два раза и, следовательно, должно быть равно 5. А так как единица, поступающая в старший разряд группы R_2' будет иметь вес 8, то потребуется коррекция вычитанием возникающего избытка в три единицы.

Если число разрядов в регистре R_1 равно n, то преобразование завершается после выполнения серии из n сдвигов.

В табл. В. 3 показан процесс преобразования десятичного числа 235_{10} в двоичную систему счисления.

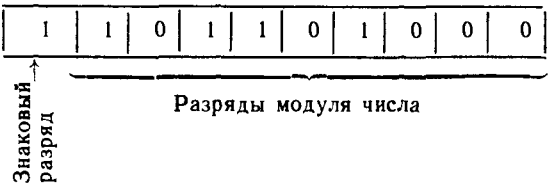
На рис. В. 4 показан алгоритм вычислений.

В.3. ФОРМЫ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ

В цифровых устройствах используются две формы представления чисел: с фиксированной и плавающей точкой.

Числа с фиксированной точкой. В ячейке для хранения числа с фиксированной точкой один разряд используется в качестве знакового, в нем записывается в закодированной форме знак числа: 0 — в случае положительного, 1 — в случае отрицательного числа. Остальные разряды используются для хранения абсолютного значения числа. Точка, отделяющая целую часть числа от ее дробной части, занимает фиксированное положение: часто перед старшим разрядом либо после младшего разряда. В первом случае для всех представляемых в этой форме чисел абсолютное значение меньше еди-

ницы. Например, число — $0{,}101101_2$ следующим образом разместится в элементах запоминающей ячейки:

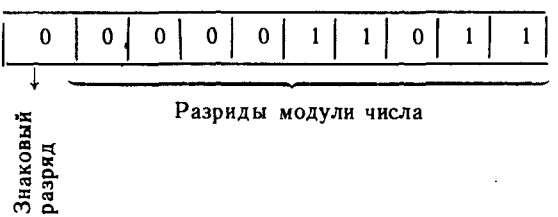


Свободные младшие разряды заполняются нулями.

Так как в этом случае предусматривается хранение лишь дробной части числа, то не только исходные данные, но и результаты всех проведенных над ними операций должны быть числами, абсолютное значение которых меньше единицы. Выполнение этого условия обеспечивается выбором определенных масштабных коэффициентов, на которые умножаются исходные данные задачи. Неправильный выбор коэффициентов может вызвать так называемое *переполнение разрядной сетки* — возникновение ошибки, если в результате выполнения операций в числе образуется целая часть, для хранения которой в разрядной сетке не предусмотрено места, и она теряется.

Необходимость в масштабировании данных составляет один из недостатков представления чисел с фиксированной точкой; другой недостаток этой формы — низкая точность представления чисел, абсолютное значение которых мало́ (нули в старших разрядах приводят к уменьшению числа разрядов, занимаемых значащей частью числа, и к снижению точности представления числа).

Во втором случае, когда точка фиксируется после младшего разряда, числа с фиксированной точкой — целые. Например, число 11011_2 будет размещено в ячейке памяти следующим образом:



Здесь свободные старшие разряды заполняются нулями.

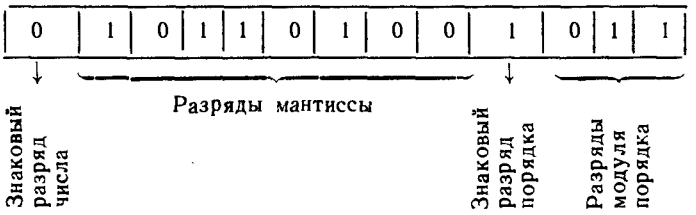
Если n — число разрядов ячейки, то диапазон модулей представимых в ней целых чисел: $0 \ldots (2^{n-1} - 1)$.

Числа с плавающей точкой. Форма с плавающей точкой предусматривает представление числа в показательной форме. Например, десятичное число 685, 73_{10} представляется в форме $0{,}68573 \times \times 10^3$; здесь 0,68573 — *мантисса*, 10 — основание десятичной системы счисления, 3 — *порядок*. Двоичное число $0{,}000101101_2$ представ-

ляется в виде $0{,}101101 \cdot 10^{-11}$; здесь $0{,}101101_2$ — мантисса, 10_2 — основание двоичной системы счисления,—11_2 — порядок.

В ячейке памяти такие числа хранятся в виде двух групп цифр: первая группа, называемая мантиссой, определяет само число, вторая группа, называемая порядком,— место точки в числе.

Приведенное выше двоичное число может иметь следующее размещение в элементах запоминающей ячейки:



Соответствующим выбором значения порядка можно добиться, чтобы старший разряд мантиссы не был равен нулю. При этом образуется так называемая *нормальная форма*.

Определим диапазон двоичных чисел, которые могут быть представлены в ячейке памяти в нормальной форме. Обозначим k — число разрядов, отведенных в ячейке для хранения абсолютного значения порядка. Положительное число в ячейке будет иметь иаименьшее значение, если минимальное значение будет иметь мантисса (все ее разряды, кроме старшего, будут содержать нуль: $0{,}100 \ldots 0_2$), а порядок будет иметь отрицательный знак и максимальное абсолютиое значение (т. е. все разряды модуля порядка будут содержать единицу: $11 \ldots 1_2 = 2^k - 1$). Таким образом, значение минимального положительного числа в нормальной форме

$$N_{\min} = \frac{1}{2} 2^{-(2^k - 1)} = 2^{-2^k}.$$

Максимальное число в ячейке образуется при максимальном значении мантиссы (когда мантисса содержит во всех разрядах единицу: $0{,}11 \ldots 1_2 \approx 1$) и положительном порядке, имеющем максимальное значение (т. е. если все разряды порядка содержат единицу: $11 \ldots 1_2 = 2^k - 1$). Следовательно, максимальное значение числа

$$N_{\max} = 2^{2^k - 1}.$$

Итак, диапазон представимых чисел в нормальной форме равен

$$N_{\min} \ldots N_{\max} = 2^{-2^k} \ldots 2^{2^k - 1}.$$

Как видим, этот диапазои определяется лишь k. Пусть, например, $k = 6$. Тогда

$$N_{\min} \ldots N_{\max} = 2^{-2^6} \ldots 2^{2^6 - 1} = 2^{-64} \ldots 2^{63} \approx 10^{-64 \cdot 0{,}3} \ldots 10^{63 \cdot 0{,}3} \approx$$
$$\approx 10^{-19} \ldots 10^{19}.$$

Если диапазон представимых чисел, как показано выше, определяется числом разрядов, отведенных в ячейке памяти для хранения порядка, то точность представления чисел определяется числом разрядов, выделенных для хранения мантиссы.

Обозначим m — количество разрядов ячейки памяти, предназначенных для хранения мантиссы. Если количество разрядов в мантиссе числа больше m, то в ячейку памяти заносятся m старших разрядов мантиссы числа; младшие ее разряды отбрасываются и может производиться округление сохраняемой части мантиссы. Округление маитиссы чисел в двоичной системе счисления выполняется по следующему правилу: если старший из отбрасываемых разрядов мантиссы содержит единицу, то к младшему разряду сохраняемой части мантиссы прибавляется единица.

При таком округлении абсолютная погрешность ε представления мантиссы не превышает половины весового коэффициента младшего из сохраняемых разрядов мантиссы:

$$\varepsilon \leqslant 1/2 \cdot 2^{-m}.$$

Так как в нормальной форме значение мантиссы не менее 1/2, то относительная погрешность представления числа составит

$$\eta \leqslant \varepsilon/0{,}5 \leqslant 2^{-m}.$$

Пусть, например, $m = 24$. Тогда

$$\eta \leqslant 2^{-24} \approx 10^{-24 \cdot 0{,}3} = 10^{-7{,}2}.$$

Следовательно, при данном значении m двоичные числа в ячейке памяти будут представлены с точностью в 7 десятичных знаков.

Нормальная форма позволяет получать представление чисел в широком диапазоне с одинаковой относительной погрешностью η. Использование формы с плавающей точкой позволяет часто обходиться без масштабирования данных. В тех же случаях, когда оно требуется, выбор масштабных коэффициентов не представляет трудностей. Однако выполнение операций над числами с плавающей точкой сложнее, чем над числами с фиксированной точкой.

В.4. ВЫПОЛНЕНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ

Основной операцией, которая используется в цифровых устройствах при различных вычислениях, является операция алгебраического сложения чисел (сложения, в котором могут участвовать как положи-

тельные, так и отрицательные числа). Вычитание легко сводится к сложению путем изменения на обратный знака вычитаемого. Операции умножения и деления также выполняются с помощью операции сложения и некоторых логических действий. Поэтому именно с операции сложения начнем рассмотрение способов выполнения арифметических операций.

При записи кода числа знак числа будем представлять полужирными цифрами **0** (для положительных чисел) и **1** (для отрицательных чисел). Положение точки в числе показывать не будем.

Сложение положительных двоичных чисел. Выполнение этой операции покажем на примере:

Переносы			1		1		
Первое слагаемое	N_1	**0**	0	1	0	0	1
Второе слагаемое	N_2	+ **0**	0	1	1	0	1
Сумма	$N = N_1 + N_2$	**0**	1	0	1	1	0

Цифры разрядов суммы $N = N_1 + N_2$ формируются последовательно, начиная с младшего разряда. Цифра младшего разряда суммы образуется суммированием цифр младших разрядов слагаемых. При этом, кроме цифры разряда суммы, формируется цифра переноса в следующий, более старший разряд. Таким образом, в разрядах, начиная со второго, суммируются три цифры: цифры соответствующего разряда слагаемых и перенос, поступающий в данный разряд из предыдущего.

Перенос равен 1 во всех случаях, когда результат суммирования цифр в разряде равен или больше $p = 2$ (p — основание системы счисления). При этом в разряд суммы записывается цифра, на p единиц (т. е. на две единицы) меньшая результата суммирования.

Алгебраическое сложение с использованием дополнительного кода. Для пояснения сущности излагаемого ниже метода рассмотрим следующий пример. Пусть требуется сложить два десятичных числа $N_1 = 0\,831$ и $N_2 = 1\,376$. Так как второе слагаемое — отрицательное число, пользование приемом, излагаемым в школьной программе, потребовало бы последовательности действий с заемами из старших разрядов. Предусматривать в цифровом устройстве дополнительно такую последовательность действий не обязательно. Искомый результат может быть получен и с использованием последовательности действий с передачей переносов в старшие разряды, которая используется при сложении положительных чисел. Для этого достаточно отрицательное число 1 376 предварительно преобразовать в так называемый *дополнительный код* следующим образом: во всех разрядах, кроме знакового, запишем дополнение до 9 к цифрам этих разрядов и затем прибавим единицу в младший разряд. Число $N_2 = 1\,376$ в дополнительном коде есть $N_{2\,доп} = 1\,624$.

Далее произведем сложение по правилам сложения с передачей переносов в старшие разряды (т. е. так, как складываются положительные числа):

Переносы	1	1			
Первое слагаемое N_1		0	8	3	1
Второе слагаемое $N_{2доп}$	+	1	6	2	4
Сумма $N = N_1 + N_2$		0	4	5	5

При сложении складываются и двоичные цифры знаковых разрядов с отбрасыванием возникающего из этого разряда переноса. Как видим, получен правильный результат (действительно, 831—376=455).

В двоичной системе счисления дополнительный код отрицательного числа формируется по следующему правилу: инвертируются (путем замены 0 на 1 и 1 на 0) цифры всех разрядов, кроме знакового, и в младший разряд прибавляется единица. Например, если $N = 1\ 10110$, $N_{доп} = 1\ 01010$. Обратное преобразование из дополнительного кода в прямой код производится по тому же правилу.

Рассмотрим примеры выполнения операции сложения.

Пример В.1. Пусть $N_1 = 0\ 10110$, $N_2 = 1\ 01101$.

Переносы	1	1		1	1		
Первое слагаемое N_1		0	1	0	1	1	0
Второе слагаемое $N_{2доп}$	+	1	1	0	0	1	1
Сумма $N = N_1 + N_2$		0	0	1	0	0	1

Как уже указывалось выше, перенос, возникающий из знакового разряда, отбрасывается.

Пример В.2. Изменим на обратный знаки слагаемых (по отношению к предыдущему примеру): $N_1 = 1\ 10110$, $N_2 = 0\ 01101$. Очевидно, ожидаемый ответ: $N = N_1 + N_2 = 1\ 01001$.

Переносы			1				
Первое слагаемое $N_{1доп}$		1	0	1	0	1	0
Второе слагаемое N_2	+	0	0	1	1	0	1
Сумма $N_{доп} = (N_1 + N_2)_{доп}$		1	1	0	1	1	1
Сумма $N = N_1 + N_2$		1	0	1	0	0	1

Таким образом, если результат сложения есть отрицательное число, то оно оказывается представленным в дополнительном коде.

Суммирование десятичных чисел. Рассмотрим вначале операцию суммирования в одном разряде десятичных чисел, т. е. суммирование двух десятичных цифр и единицы переноса, которая при суммировании чисел может поступить из предыдущего десятичного разряда. Способ суммирования десятичных цифр зависит от того, какой двоичный код выбран для представления десятичных цифр. Ниже рассматривается операция суммирования при использовании *кода 8421*.

Двоичные представления десятичных цифр суммируются по обычным правилам сложения двоичных чисел. Если полученная сумма содержит десять или более единиц, то формируется единица переноса, передаваемая в следующий десятичный разряд, а из суммы вычитаются десять единиц. Полученный результат есть цифра соответствующего разряда суммы. Наличие в полученной сумме десяти или более единиц выявляется по следующим признакам: появление переноса из разряда 8, возникающего при суммировании цифр; наличие единиц одновременно в разрядах 8 и 4 либо 8 и 2 в полученной сумме. При этом требуется коррекция суммы прибавлением к ней шести единиц (числа 0110_2).

Покажем эти действия на примерах.

Пример В.3. Сложить десятичные цифры 6 и 2 и перенос 1, поступающий из предыдущего десятичного разряда.

	Десятичная система	*Код 8421*
Переносы	1←	1 1 1←
Первая цифра	6	0 1 1 0
Вторая цифра	+ 2	+ 0 0 1 0
Сумма	9	1 0 0 1
Коррекция		—
Результат		1 0 0 1

В этом случае полученное в результате суммирования число 1001_2 меньше десяти и коррекция суммы не требуется.

Пример В.4. Сложить десятичные цифры *8* и *9*.

	Десятичная система	*Код 8421*
Переносы	1←0←	1← 0←
Первая цифра	8	1 0 0 0
Вторая цифра	+ 9	1 0 0 1
Сумма	7	0 0 0 1
Коррекция		+ 0 1 1 0
Результат		0 1 1 1

В данном случае сложение двух единиц в разряде 8 дает в соответствующем разряде суммы 0 и перенос 1 из разряда 8. Таким образом, появление переноса из разряда 8,передаваемого в следующий десятичный разряд, уменьшает сумму не на 10, а на 16 единиц. Уход из суммы шести лишних единиц компенсируется прибавлением 6 единиц в ходе коррекции.

Пример В.5. Сложить десятичные цифры 6 и 7.

	Десятичная система	*Код 8421*
Переносы	1←0←	1 1 1 0←
Первая цифра	6	0 1 1 0
Вторая цифра	+ 7	0 1 1 1
Сумма	3	+ 1 1 0 1
Коррекция		0 1 1 0
Результат		1←0 0 1 1

В данном примере суммирование десятичных цифр приводит к числу 1101_2 (13). Так как сумма больше десяти, то необходимо передать перенос в следующий десятичный разряд, а сумму скорректировать, прибавив к ней 6 единиц. В процессе коррекции возникает отбрасываемый перенос из разряда 8, уменьшающий сумму на 16 единиц. Таким образом, прибавление 6 и вычитание 16 обеспечивают требуемое уменьшение суммы на 10 единиц.

При использовании других кодов для представлеиия десятичных цифр правила суммирования отличаются от приведенных выше.

При суммировании многоразрядных десятичных чисел отрицательные числа должны быть предварительно представлены в обратном либо дополнительном коде. Обратный код отрицательного десятичиого числа получается путем замены цифр разрядов (кроме знакового) их дополиениями до 9.

Пример В.6. Сложить числа $N_1 = 836$ и $N_2 = -298$.

Переносы	1
N_1	0 8 3 6
$N_{2обр}$	+ 1 7 0 1
	+ 1←0 5 3 7
	+|——→1
$N = N_1 + N_2$	0 5 3 8

Пример В.7. Сложить числа $N_1 = -836$ и $N_2 = 298$.

Переносы	1 1
$N_{1обр}$	1 1 6 3
N_2	+ 0 2 9 8
$N_{обр} = (N_1 + N_{2обр})$	1 4 6 1
$N = N_1 + N_2$	1 5 3 8

Простота формирования дополнения до 9 достигается при пользовании *кодом с избытком 3* либо *кодом 2421*.

Умножение двоичных чисел. Операция умножения чисел, представленных в форме с фиксированной точкой, включает в себя определение знака и абсолютного значения произведения.

Определение знака произведения. Знаковый разряд произведения может быть получен суммированием знаковых разрядов сомножителей без формирования переноса (так называемым суммированием по модулю 2). Действительно, при совпадении цифр знаковых разрядов сомножителей (**0** ... и **0**..., либо **1** ... и **1** ...) их сумма по модулю 2 равна 0, т. е. соответствует знаковому разряду произведения двух сомножителей, имеющих одинаковые знаки; при несовпадении цифр знаковых разрядов эта сумма будет равна 1, что также соответствует знаковому разряду произведения двух сомножителей с разными знаками.

Определение абсолютного значения произведения. Абсолютное значение произведения получается путем перемножения чисел без учета их знаков (так называемого *кодового умножения*).

Пусть производится умножение чисел 1101_2 и 1011_2.

1 1 0 1	множимое
×	
1 0 1 1	множитель
1 1 0 1	1-е частичное произведение
1 1 0 1	2-е частичное произведение
0 0 0 0	3-е частичное произведение
1 1 0 1	4-е частичное произведение
1 0 0 0 1 1 1 1	произведение

Как видно из примера, в процессе выполнения операции умножения формируются частичные произведения (произведения множимого

на цифры разрядов множителя), которые суммируются с соответствующими сдвигами друг относительно друга. В цифровых устройствах процессу суммирования частичных произведений придают последовательный характер: формируется одно из частичных произведений, к нему с соответствующим сдвигом прибавляется следующее частичное произведение, к полученной сумме двух частичных произведений прибавляется с соответствующим сдвигом очередное частичное произведение и т.д., пока не окажутся просуммированными все частичные произведения. Этот процесс суммирования можно начинать с младшего либо старшего частичного произведения.

Ниже показаны процессы при умножении с суммированием частичных произведений, начиная со старшего частичного произведения (используется приведенный выше пример умножения чисел 1101_2 и 1011_2).

```
      1 1 0 1      4-е частичное произведение
    1 1 0 1 0      сдвиг на один разряд влево
  +
      0 0 0 0      3-е частичное произведение
    1 1 0 1 0      сумма 4-го и 3-го частичных произведений
  1 1 0 1 0 0      сдвиг на один разряд влево
  +
      1 1 0 1      2-е частичное произведение
  1 0 0 0 0 0 1    сумма 4-го, 3-го и 2-го частичных произведений
1 0 0 0 0 0 1 0    сдвиг на один разряд влево
+
      1 1 0 1      1-е частичное произведение
1 0 0 0 1 1 1 1    произведение
```

Нетрудно убедиться, что при этом все частичные произведения суммируются с требуемыми сдвигами относительно друг друга, благодаря чему и образуется ранее приведенный результат умножения чисел.

При умножении целых чисел для фиксации произведения в разрядной сетке должно предусматриваться число разрядов, равное сумме числа разрядов множимого и множителя.

Рассмотрим процессы при выполнении операции умножения с суммированием частичных произведений, начиная с младшего частичного произведения, на примере умножения дробных чисел $0{,}1101_2$ и $0{,}1011_2$.

0 , 1 1 0 1		1-е частичное произведение
0 , 0 1 1 0	1	сдвиг на один разряд вправо
+		
0 , 1 1 0 1		2-е частичное произведение
1 , 0 0 1 1	1	сумма 1-го и 2-го частичных произведений
0 , 1 0 0 1	1 1	сдвиг на один разряд вправо
+		
0 , 0 0 0 0		3-е частичное произведение
0 , 1 0 0 1	1 1	сумма 1-го, 2-го и 3-го частичных произведений
0 , 0 1 0 0	1 1 1	сдвиг на один разряд вправо
+		
0 , 1 1 0 1		4-е частичное произведение
1 , 0 0 0 1	1 1 1	сумма частичных произведений
0 , 1 0 0 0	1 1 1 1	сдвиг вправо, произведение

Если требуется сохранять все разряды в произведении, то в устройстве, формирующем произведение, необходимо иметь число разрядов, равное сумме числа разрядов множимого и множителя. При умножении дробных чисел часто в произведении требуется сохранять то же число разрядов, что и в множимом. В таком приближенном представлении результата не фиксируются цифры разрядов, при сдвигах выдвигаемые правее показанной в примере вертикальной линии. Таким образом, цифры четырех младших разрядов в примере окажутся потерянными и будет получен приближенный результат 0,1000. Может быть проведено округление по правилу: если старший из отбрасываемых разрядов содержит единицу, то к младшему из сохраняемых разрядов прибавляется единица (результат с округлением равен в примере 0,1001).

Рассмотренный выше способ умножения предусматривал отделение от сомножителей их знаковых разрядов и раздельное выполнение действий над знаками и модулями чисел. Одним из эффективных алгоритмов умножения является алгоритм Бута. Он не предусматривает раздельных операций над знаковыми разрядами и модулями сомножителей. При выполнении действий по этому алгоритму в получаемом результате образуется произведение со знаковым разрядом.

На рис. В.5 показана схема алгоритма в содержательных обозначениях. Здесь X — множимое; Y — множитель; S — старшие разряды суммы частичных произведений; Сч — счетчик числа повторений цикла, в который в качестве начального значения заносится число разрядов сомножителей, включая и знаковый разряд; *П1*, *П2* — разряды множителя, выдвигаемые при его сдвиге соответственно в предыдущем и текущем повторениях цикла.

В блоке *2* производится арифметический сдвиг вправо S, Y (при арифметическом сдвиге вправо сохраняется содержимое знакового разряда), рассматриваемых как единое число, в котором S образует его старшие, а Y — младшие разряды. Значение появляющегося из младшего разряда Y переноса присваивается *П2*, а содержимое *П2* передается в *П1*.

Далее в блоках *3* ... *7* организуется выполнение действий по следующему правилу:

П1	*П2*	Выполняемое действие
0	0	—
0	1	$S = S - X$
1	0	$S = S + X$
1	1	—

При комбинациях значений *П1* и *П2*, равных 00 и 11, значение S не изменяется; при комбинации 01 значение S уменьшается на величину X, а при комбинации 10 — увеличивается на X.

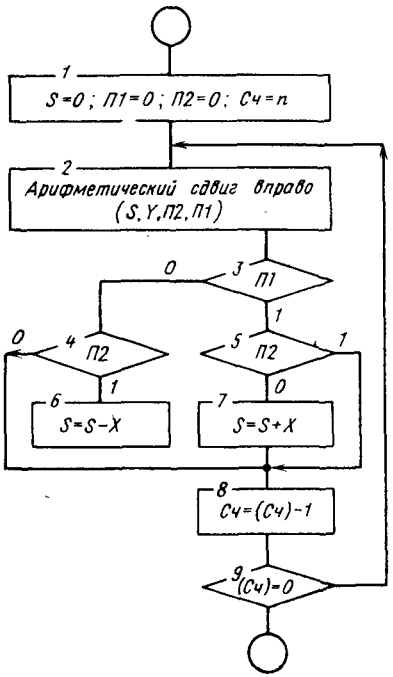


Рис. В.5. Схема алгоритма умножения Бута в содержательных обозначениях

В последующих блоках производится подготовка к повторению цикла. Содержимое счетчика уменьшается на единицу и по результату проверки содержимого счетчика на нуль выясняется необходимость повторения цикла.

После выхода из цикла в S образуется знаковый разряд и группа старших разрядов произведения, в $(n-1)$ разрядах Y — группа младших разрядов произведения.

Следует отметить, что отрицательные сомножители должны представляться в дополнительном коде; если произведение оказывается отрицательным числом, то оно представляется в дополнительном коде.

Покажем описанные действия на примере умножения чисел $X = 01011_2$ (11_{10}) и $Y = 11101_2$ (-13_{10}). Дополнительный код множителя $Y_{доп} = 10011$. Проводимые при умножении действия иллюстрируются табл. В.3.

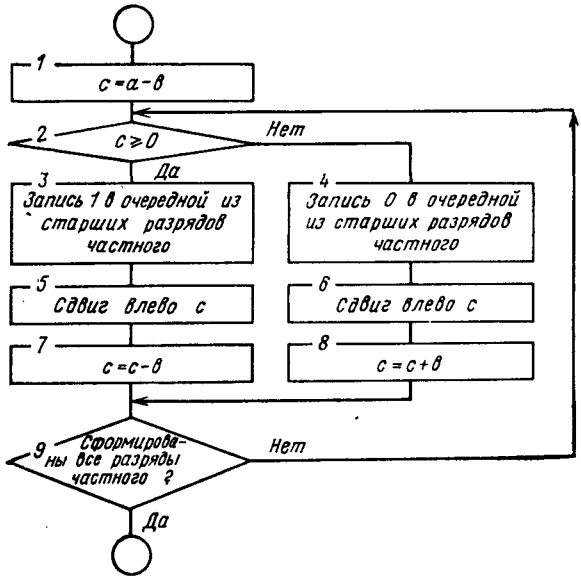


Рис. В.6. Схема алгоритма деления положительных чисел *a* и *b*

Таблица В.3

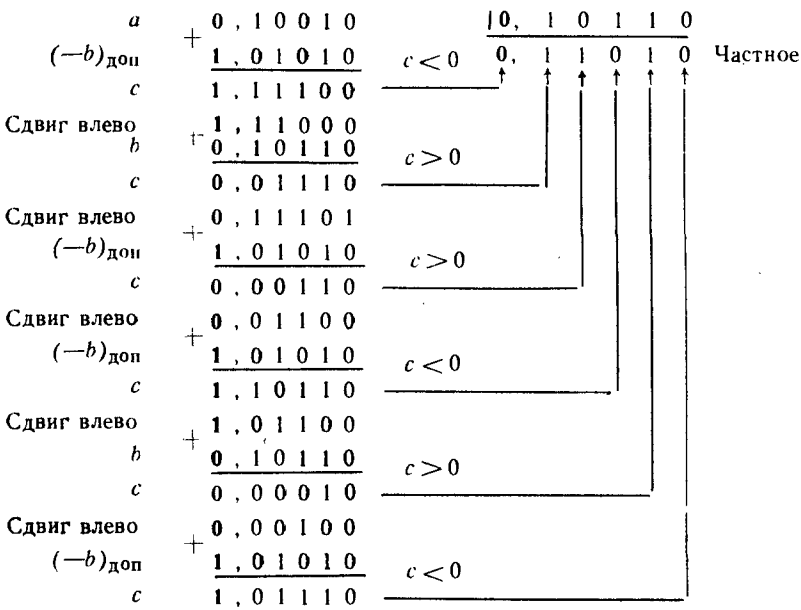
X	$(-X)_{доп}$	S	$Y_{доп}$	$П2$	$П1$	Выполняемые действия
01011	10101	00000	10011	0	0	Исходное состоянне
		00000	01001	1	0	Арифметический сдвиг вправо (S, Y, $П2$, $П1$)
		+ 10101				
		10101				Вычитанне
		11010	10100	1	1	Арнфметическнй сдвиг вправо (S, Y, $П2$, $П1$)
		11101	01010	0	1	Арнфметический сдвиг вправо (S, Y, $П2$, $П1$)
		+ 01011				
		01000				Сложенне
		00100	00101	0	0	Арнфметический сдвиг вправо (S, Y, $П2$, $П1$)
		00010	00010	1	0	Арнфметический сдвиг вправо (S, Y, $П2$, $П1$)
		+ 10101				
		10111				Вычитание
		10111	00010			Произведение в дополнительном коде
		11000	11110_2		-143_{10}	Произведение в прямом коде

Деление двоичных чисел. Будем рассматривать операцию алгебраического деления чисел, представленных в форме с фиксированной точкой. При этом выполнение операции содержит действия, связанные с определением знака частного, и действия, связанные с определением модуля частного. Знак частного может быть найден тем же приемом, что и знак произведения в рассмотренной выше операции умножения с отделением знаковых разрядов. Поэтому ниже рассматривается лишь нахождение модуля частного.

На рис. В. 6 показана схема алгоритма нахождения частного положительных чисел a и b.

Покажем выполнение операции на примере. Пусть после отделения знаковых разрядов модули делимого и делителя представляются соответственно числами $a = 0{,}10010$ и $b = 0{,}10110$.

Встречающуюся в алгоритме операцию вычитания числа заменим прибавлением числа $-b$, представленного в дополнительном коде: $(-b)_{доп} = 1{,}01010$.



1. ПРИНЦИПЫ ПОСТРОЕНИЯ ПРОЦЕССОРОВ

1.1. АНАЛОГОВЫЙ И ЦИФРОВОЙ МЕТОДЫ ОБРАБОТКИ ИНФОРМАЦИИ

Обработка информации может выполняться двумя методами: аналоговым, при котором участвующие в обработке величины представляются в аналоговой форме (обычно уровнями напряжения либо тока), или цифровым, при котором величины представляются в цифровой форме и сама обработка сводится к последовательности действий (операций) над числами.

В зависимости от используемого метода обработки различают два типа аппаратуры: аналоговую, в которой используется аналоговый метод обработки, и цифровую, в которой применяется цифровой метод обработки. В цифровой аппаратуре основным устройством, в котором непосредственно выполняется обработка, является процессорное устройство.

Рассмотрим характерные особенности указанных методов обработки информации.

В аналоговой аппаратуре обработка информации заключается в преобразованиях между токами и напряжениями вида $u_L = L\mathrm{d}i_L/\mathrm{d}t$, $i_C = C\mathrm{d}u_C/\mathrm{d}t$; $u_r = ri_r$, выполняемых соответственно индуктивными, емкостными, резистивными элементами, в изменении масштаба $u_2 = ku_1$, а также в нелинейных преобразованиях $u_2 = f(u_1)$. При этом каждый элемент аналогового устройства в каждый момент времени находится в состоянии активного выполнения характерных для этих элементов операций; таким образом, имеет место параллельное выполнение операций во всех элементах устройства.

В показанном на рис. 1.1, *а* простейшем аналоговом устройстве на выходе формируется величина $u_c(t)$, являющаяся решением следующего дифференциального уравнения:

$$\frac{\mathrm{d}u_C(t)}{\mathrm{d}t} + \frac{1}{rC}u_C(t) = \frac{1}{rC}e(t). \qquad (1.1)$$

На рис. 1.1, *б* показан график функции $u_C(t)$ при некоторой конкретной форме входного воздействия $e(t)$.

Рассмотрим решение дифференциального уравнения (1.1) цифровым методом. Одна из особенностей решения цифровым методом состоит в том, что оно может быть получено для дискретных моментов времени — некоторой последовательности моментов времени, например, следующих с интервалом T (рис. 1.1, *в*). Интервал времени T назовем шагом интегрирования и решения дифференциального уравнения, получаемые для моментов времени, следующих с интервалом T, обозначим ..., $u_C(nT)$, $u_C(nT+T)$, $u_C(nT+2T)$, ...

Представим входящую в выражение (1.1) производную следующим приближением для момента $t = nT$:

$$\left.\frac{\mathrm{d}u_C(t)}{\mathrm{d}t}\right|_{t=nT} \approx \frac{u_C(nT+T)-u_C(nT)}{T}.$$

После подстановки в уравнение и некоторых очевидных преобразований получим

$$u_C(nT+T) = k_1 u_C(nT) + k_2 e(nT), \qquad (1.2)$$

где $k_1 = 1 - T/rC$; $k_2 = T/rC$ — константы.

Выражение (1.2) представляет собой так называемую разностную форму дифференциального уравнения (1.1). Этим выражением пользуются как рекуррентным: по известным значениям $u_C(nT)$, $e(nT)$ вычисляется $u_C(nT+T)$; затем по значениям $u_C(nT+T)$ и $e(nT+T)$ с помощью того же выражения вычисляется $u_C(nT+2T)$ и т. д.

На рис. 1.2 представлен цифровой алгоритм задачи в форме схемы. Вычислительный процесс носит циклический характер. При каждом исполнении предусмотренных блоками *1* ... *5* действий вычисляется очередное значение выходной величины.

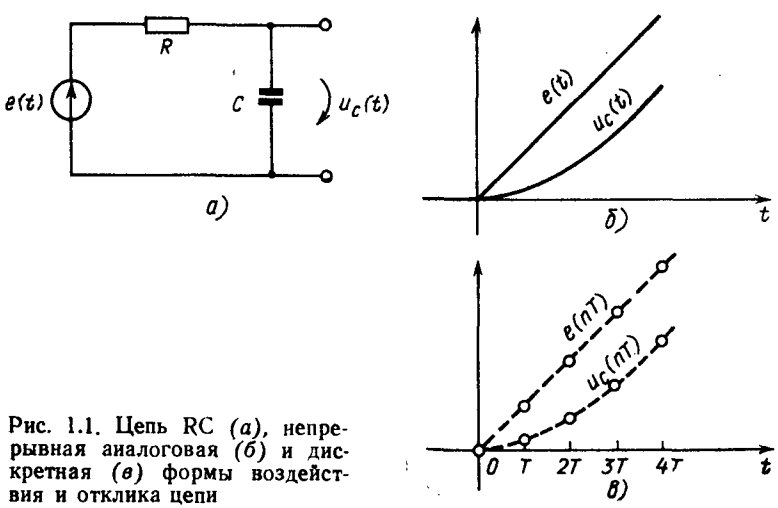


Рис. 1.1. Цепь RC (*а*), непрерывная аналоговая (*б*) и дискретная (*в*) формы воздействия и отклика цепи

Для хранения входящих в выражение (1.2) величин в процессорном устройстве используем регистры: регистр R_0 для формирования в нем вычисляемого в данном шаге интегрирования значения $u_C(nT + T)$; регистр R_1 для хранения вычисленного в предыдущем шаге интегрирования значения $u_C(nT)$; регистры R_2 и R_3 для хранения соответственно k_1 и k_2; регистр R_4 для хранения промежуточных результатов.

При записи выполняемых в блоках действий использованы следующие обозначения: (R_i) — содержимое регистра R_i, стрелка — знак засылки результата операции в соответствующий регистр.

Рассмотрим операции, выполняемые в отдельных блоках. В блоке *1* производится прием очередного значения e в регистр R_4. Блок *2* формирует в регистре R_0 значение второго слагаемого выражения (1.2). Блок *3* вычисляет значение первого слагаемого этого выражения, и оно затем в блоке *4* прибавляется к содержимому регистра R_0. При этом в регистре R_0 образуется вычисленное значение выходной величины $u_C(nT + T)$, которое блоком *5* выдается. Блок *6* производит подготовку к следующему повторению цикла для вычисления $u_C(nT + 2T)$. Необходимой замене в правой части выражения (1.2) $u_C(nT)$ на $u_C(nT + T)$ здесь соответствует передача вычисленного значения $u_C(nT + T)$ в регистр R_1, ранее хранивший $u_C(nT)$. Далее происходит переход к блоку *1* и очередное повторение цикла.

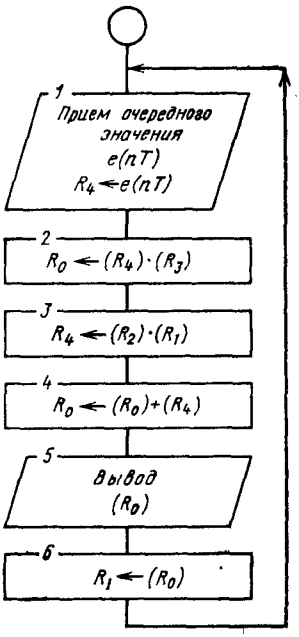


Рис. 1.2. Схема алгоритма при цифровом методе анализа процесса в RC-цепи

Таким образом, для получения одного значения выходной величины требуется последовательное выполнение большого числа операций (в отличие от аналоговой цепи, где операции во всех элементах выполняются параллельно), что затрудняет достижение высокого быстродействия. Кроме того, при цифровом методе обработки возникает необходимость в большом количестве относительно сложных узлов (регистров, сумматора, множительного устройства и др.), а также в устройстве, координирующем их работу.

Однако цифровые методы по сравнению с аналоговыми имеют ряд достоинств: возможность обеспечения любой требуемой точности обработки, высокую помехозащищенность, высокую стабильность характеристик обработки, возможность выполнения таких видов обработки, которые аналоговыми методами трудно либо вовсе невыполнимы.

Аппаратурная сложность цифровых устройств прежде приводила к высокой стоимости, большим габаритным размерам и массе, высо-

кому потреблению энергии, в связи с чем цифровые методы в большинстве случаев оказывались неприемлемыми для практического использования. Ситуация изменилась благодаря достижениям микроэлектроники, обеспечивающей в одной микросхеме упаковку тысяч и десятков тысяч элементов. В настоящее время цифровые устройства имеют малые стоимость, габаритные размеры, массу, потребление энергии. Это объясняет их широкое использование в тех случаях, когда они могут обеспечить требуемую скорость обработки.

1.2. ОБЩАЯ СТРУКТУРА ПРОЦЕССОРА

Процессор синтезируется в виде соединения двух устройств: операционного и управляющего (рис. 1.3).

Операционное устройство (ОУ) — устройство, в котором выполняются операции. Оно включает в себя в качестве узлов регистры, сумматор, каналы передачи информации, мультиплексоры для коммутации каналов, шифраторы, дешифраторы и т. д. Управляющее устройство координирует действия узлов операционного устройства; оно вырабатывает в определенной временной последовательности управляющие сигналы, под действием которых в узлах операционного устройства выполняются требуемые действия.

Процесс функционирования операционного устройства распадается на последовательность элементарных действий в его узлах. Такими элементарными действиями могут быть:

1) установка регистра в некоторое состояние (например, запись в регистр R_1 числа 0, обозначаемая $R_1 \leftarrow 0$);

2) инвертирование содержимого разрядов регистра [например, если регистр R_2 содержал двоичное число 101101, то после инвертирования его содержимое будет равно 010010, такое действие обозначают $R_2 \leftarrow (\overline{R_2})$];

3) пересылка содержимого одного узла в другой [например, пересылка содержимого регистра R_2 в регистр R_1, обозначаемая $R_1 \leftarrow (R_2)$];

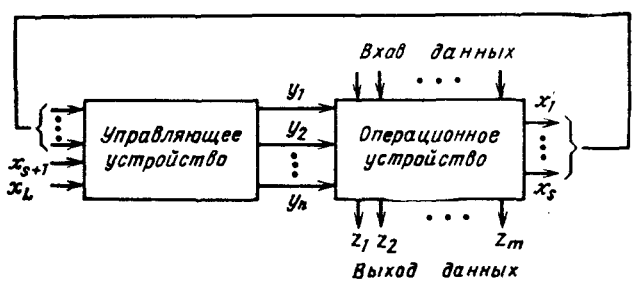


Рис. 1.3. Структура процессора

4) сдвиг содержимого узла влево, вправо [например, сдвиг на один разряд влево содержимого регистра R_1, обозначаемый $R_1 \leftarrow$ $\leftarrow$Сдв Л(R_1)];

5) счет, при котором число в счетчике (регистре) возрастает или убывает на единицу [Сч $\leftarrow$ (Сч) $\pm$ 1];

6) сложение [например, $R_2 \leftarrow (R_2) + (R_1)$];

7) сравнение содержимого регистра на равенство с некоторым числом, результат сравнения — логическая 1 (в случае выполнения равенства) либо логический 0 (в случае невыполнения равенства);

8) некоторые логические действия (поразрядно выполняемые операции конъюнкции, дизъюнкции и др.).

Каждое такое элементарное действие, выполняемое в одном из узлов ОУ в течение одного тактового периода, называется *микрооперацией*.

В определенные тактовые периоды одновременно могут выполняться несколько микроопераций, например $R_2 \leftarrow 0$, Сч $\leftarrow$ (Сч) — 1. Такая совокупность одновременно выполняемых микроопераций называется *микрокомандой*, а весь набор микрокоманд, предназначенный для решения определенной задачи, — *микропрограммой*.

Таким образом, если в операционном устройстве предусматривается возможность исполнения n различных микроопераций, то из управляющего устройства выходят n управляющих цепей, каждая из которых соответствует определенной микрооперации. И если необходимо в операционном устройстве выполнить некоторую микрооперацию, достаточно из управляющего устройства по определенной управляющей цепи, соответствующей данной микрооперации, подать сигнал (например, напряжение уровня логической 1). В силу того, что управляющее устройство определяет микропрограмму, т. е. какие и в какой временной последовательности должны выполняться микрооперации, оно получило название *микропрограммного автомата*.

Формирование управляющих сигналов $y_1 \ldots y_n$ (рис. 1.3) для выполнения определенных микрокоманд может происходить в зависимости от состояния узлов операционного устройства, определяемого сигналами $x_1 \ldots x_s$, которые подаются по определенным цепям с соответствующих выходов операционного устройства на входы управляющего устройства. Управляющие сигналы $y_1 \ldots y_n$ могут также зависеть от внешних сигналов $x_{s+1} \ldots x_L$.

Для сокращения числа управляющих цепей, выходящих из управляющего устройства (в тех случаях, когда оно конструктивно выполняется отдельно от операционного), микрокоманды могут определенным образом кодироваться. Поясним это на примере. Допустим, что в узлах ОУ предусматриваются 20 микроопераций. Пусть выполняемые в различных комбинациях они должны образовывать 470 микрокоманд. В закодированном виде микрокомаиды могут представляться 9-разрядным двоичным кодом. Число комбинаций такого кода составляет $2^9 = 512$. Таким образом, каждой микрокоманде может быть поставлена в соответствие определенная из этих комбинаций 9-разряд-

ного кода (например, 1-й микрокоманде — кодовая комбинация 000 000 000, 2-й микрокоманде — комбинация 000 000 001). При этом микрокоманда на входе операционного устройства будет задаваться некоторой 9-разрядной двоичной комбинацией, для управления же выполнением микроопераций имеется 20 управляющих цепей. Возникает необходимость преобразования 9-разрядной микрокоманды в 20-разрядную комбинацию сигналов в управляющих цепях. Такое преобразование может осуществляться различными способами, например с помощью программируемой логической матрицы (ПЛМ) либо с помощью дешифратора и элементов ИЛИ, объединяющих определенные выходы дешифратора, соответствующие микрокомандам, при которых выполняется одна и та же микрооперация.

Результаты обработки, выполненной в ОУ, снимаются с его выходов $z_1 \ldots z_m$.

1.3. ПОСТРОЕНИЕ ПРОЦЕССОРА

Существует два принципиально разных подхода к проектированию микропрограммного автомата (управляющего устройства): использование принципа схемной логики или использование принципа программируемой логики.

В первом случае в процессе проектирования подбирается некоторый набор цифровых микросхем (обычно малой и средней степени интеграции) и определяется такая схема соединения их выводов, которая обеспечивает требуемое функционирование (т. е. функционирование процессора определяется тем, какие выбраны микросхемы и по какой схеме выполнено соединение их выводов). Устройства, построенные на таком принципе схемной логики, способны обеспечивать наивысшее быстродействие при заданном типе технологии элементов. Недостаток этого принципа построения процессора состоит в трудности использования последних достижений микроэлектроники — интегральных микросхем большой и сверхбольшой степени интеграции (БИС и СБИС). Это связано с тем, что при использовании схемного принципа каждый разрабатываемый процессор окажется индивидуальным по схемному построению и потребует изготовления индивидуального типа БИС. Тогда выпускаемые промышленностью БИС окажутся узкоспециализированными, число выпускаемых типов БИС будет большим, а потребность в каждом типе БИС окажется низкой. Выпуск многих типов БИС малыми сериями по каждому типу для промышленности окажется экономически невыгодным.

Эти обстоятельства заставляют обратиться к другому подходу в проектировании цифровых устройств, основанному на использовании принципа программируемой логики. Этот подход предполагает построение с использованием одной или нескольких БИС некоторого универсального устройства, в котором требуемое функционирование (т. е.

их специализация) обеспечивается занесением в память устройства определенной программы (или микропрограммы). В зависимости от введенной программы такое универсальное управляющее устройство способно обеспечивать требуемое управление операционным устройством при решении самых различных задач. В этом случае число типов БИС, необходимых для построения управляющего устройства, небольшое, а потребность в БИС каждого типа высока. Это обеспечивает целесообразность их выпуска промышленностью.

До сих пор речь шла о построении управляющих устройств процессоров. Теперь рассмотрим условия для широкого использования БИС при построении операционных устройств процессоров. Можно построить операционное устройство с таким набором узлов и такой схемой их соединения, которые обеспечивают решение разнообразных задач. Задача, решаемая таким универсальным операционным устройством определяется тем, какая микропрограмма хранится в управляющем устройстве. Таким образом, независимо от решаемой задачи может быть использовано одно и то же операционное устройство. Благодаря тому, что потребность в таких устройствах окажется высокой, они могут быть построены с использованием БИС.

Следует однако иметь в виду, что наивысшее быстродействие достигается в процессорах, в которых управляющее устройство строится с использованием принципа схемной логики, а операционное устройство выполняется в виде устройства, специализированного для решения конкретной задачи.

Набор БИС, обеспечивающих построение цифровых устройств, образует микропроцессорный комплект (МПК). Они представляют собой комплекты БИС, позволяющие совместно со сравнительно небольшим числом микросхем средней и малой степени интеграции создавать миниатюрные вычислительные устройства для разнообразных применений. Устройства, реализуемые с использованием МПК, — микропроцессорные устройства (МПУ).

Если в устройстве, построенном на принципе схемной логики, всякое изменение или расширение выполняемых фуикций влечет за собой демонтаж устройства и монтаж устройства по новой схеме, то в случае МПУ благодаря использованию принципа программируемой логики такое изменение достигается замеиой хранящейся в памяти программы новой программой, соответствующей новым выполняемым устройством функциям. Такая гибкость применений вместе с другими связаиными с использованием БИС достоинствами (низкой стоимостью, малыми габаритами), а также высокая точность и помехозащищенность, характерные для цифровых методов, обусловили бурное внедрение МПУ в различиые сферы производства, научные исследования и бытовую технику.

Мнкропроцессорные устройства в свою очередь обеспечили широкое использование цифровых методов в различиых технических применениях, и размах внедрения этих новых методов рассматривается как революция в технике.

1.4. СИНТЕЗ ПРОЦЕССОРА СО СПЕЦИАЛИЗИРОВАННЫМ ОПЕРАЦИОННЫМ УСТРОЙСТВОМ И УПРАВЛЯЮЩИМ УСТРОЙСТВОМ НА ОСНОВЕ СХЕМНОЙ ЛОГИКИ

Рассмотрим методику построения процессора на примере реализации устройства, выполняющего операцию умножения двоичных чисел без знака. Проиллюстрируем выполнение операции на примере умножения двоичных чисел 1101 и 1011:

```
      1 1 0 1  — множимое
    ×
      1 0 1 1  — множитель
    ---------
      1 1 0 1  — 1-е частичное произведение
    1 1 0 1    — 2-е частичное произведение
  0 0 0 0      — 3-е частичное произведение
1 1 0 1        — 4-е частичное произведение
---------------
1 0 0 0 1 1 1 1 — произведение
```

Предусмотрим суммирование частичных произведений, начиная с младшего частичного произведения. В табл. 1.1 приведена схема выполнения операции.

В исходном состоянии регистры R_1 и R_2 хранят соответственно множимое и множитель, регистр R_3 установлен в нулевое состояние.

Анализируется содержимое младшего разряда регистра R_2. Так как в рассматриваемом примере этот разряд содержит единицу, то 1-е частичное произведение равно множимому и оно прибавляется к содержимому регистра R_3, используемому для накопления суммы частичных произведений. Далее производится сдвиг на один разряд вправо содержимого регистров R_2 и R_3, причем выдвигаемый при сдвиге из регистра R_3 младший разряд числа (не принимающий участия в последующем суммировании частичных произведений) передается в освобождающийся при сдвиге старший разряд регистра R_2. В младшем разряде регистра R_2 оказывается 2-й разряд множителя. Анализ младшего разряда регистра R_2 вновь обнаруживает в нем единицу и производится прибавление множимого (2-го частичного произведения) к сдвинутому вправо 1-му частичному произведению в регистре R_3. В регистре R_3 образуется сумма двух первых частичных произведений. Заметим, что при суммировании может возникнуть перенос из старшего разряда, его необходимо запомнить, вдвигая затем при сдвиге в старший разряд регистра R_3. Производится очередной сдвиг содержимого регистров R_3 и R_2.

Таблица 1.1

Множимое (R_1)	Старшие разряды произведения (R_3)	Множитель и младшие разряды произведения (R_2)	Выполняемое действие
1 1 0 1	0 0 0 0	1 0 1 1	Исходное состояние
	+ 1 1 0 1		
	0 → 1 1 0 1		Суммирование
	0 ↘ 0 1 1 0	→ 1 1 0 1	Сдвиг (R_3) и (R_2)
	+ 1 1 0 1		
	1 ← 0 0 1 1		Суммирование
	0 ↘ 1 0 0 1	→ 1 1 1 0	Сдвиг (R_3) и (R_2)
	0 ↘ 0 1 0 0	→ 1 1 1 1	Сдвиг (R_3) и (R_2)
	+ 1 1 0 1		
	1 ← 0 0 0 1		Суммирование
	0 ↘ 1 0 0 0	→ 1 1 1 1	Сдвиг (R_3) и (R_2)
	Произведение		

В младшем разряде регистра R_2 обнаруживается нуль (3-й разряд множителя), частичное произведение равно нулю и суммирование не производится. Осуществляется очередной сдвиг.

В младшем разряде регистра R_2 выявляется единица (4-й разряд множителя), прибавляется очередное частичное произведение и после выполнения сдвига регистр R_3 содержит старшие разряды произведения, а регистр R_2 — младшие разряды произведения.

Из приведенного описания видно, что процессы при выполнении умножения носят циклический характер. В каждом повторении цикла выполняются следующие действия: анализируется содержимое младшего разряда регистра R_2; если оно равно единице, производится прибавление множимого к содержимому R_3; осуществляется сдвиг содержимого регистров R_3 и R_2. Число повторений цикла равно числу разрядов множителя.

СИНТЕЗ ОПЕРАЦИОННОГО УСТРОЙСТВА

В соответствии с описанным выше процессом для выполнения операции умножения необходимо в операционном устройстве иметь регистры R_1, R_2, R_3, сумматор (См) и счетчик (Сч) числа повторений цикла.

На рис. 1.4 показана структурная схема операционного устройства. В регистре R_2 предусмотрены микрооперация сдвига содержимого на один разряд вправо, выполняемая под действием управляющего сигнала y_1, и микрооперация пересылки в старший разряд этого регистра содержимого младшего разряда регистра R_3, выполняемая под дей-

ствием управляющего сигнала y_2. Сумматор См производит суммирование чисел, поступающих с выходов регистров R_1 и R_3; для хранения переноса, который может возникнуть из старшего разряда при суммировании, в нем предусмотрен дополнительный $(n+1)$-й разряд. Результат выполненной в сумматоре операции при наличии управляющего сигнала y_3 принимается в регистр R_3, который должен иметь то же число разрядов $n+1$, что и сумматор. Кроме микрооперации приема суммы, в регистре R_3 предусмотрены микрооперации установки нулевого значения и сдвига его содержимого на один разряд вправо, выполняемые соответственно под действием управляющих сигналов y_4 и y_5. При наличии управляющего сигнала y_6 в счетчик Сч принимается установленное на его входе число n; под действием управляющего сигнала y_7 выполняется микрооперация вычитания единицы из содержимого счетчика.

В операционном устройстве формируются следующие признаки: x_1 — содержимое младшего разряда регистра R_2 и x_2 — результат проверки на нуль содержимого счетчика.

Прием в регистр R_3 поступающего на его вход числа (с выхода сумматора См) можно выполнить по схеме, показанной на рис. 1.5, *а*. На рис. 1.5, *б* показаны временные диаграммы, поясняющие работу схемы. В отсутствие управляющего сигнала ($y_3 = 0$) на выходе элемента ИЛИ устанавливается постоянный уровень логической 1. Если в некотором тактовом интервале поступает управляющий сигнал

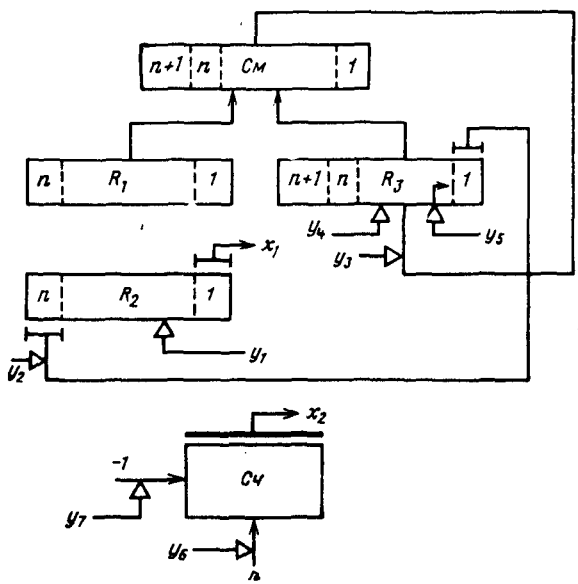


Рис. 1.4. Структурная схема операционного устройства, выполняющего операцию умножения

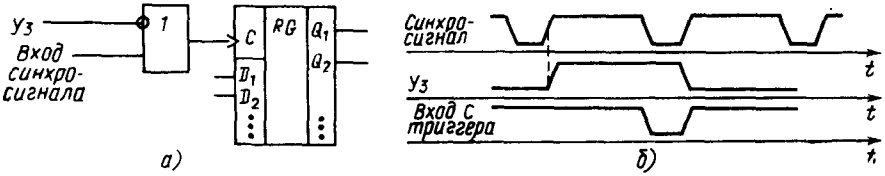


Рис. 1.5. Выполнение микрооперации записи в регистр:
а) схема; *б*) временные диаграммы

($y_3 = 1$), то на выход элемента ИЛИ передается синхросигнал этого тактового периода; на вход С регистра поступает положительный фронт, под действием которого поданное на входы D_1, D_2, ... (предполагается, что регистр построен на D-триггерах) число фиксируется в регистре.

Приведем в условной записи список выполняемых в узлах операционного устройства микроопераций и список формируемых признаков:

$y_1 : R_2 \leftarrow$ Сдв. П (R_2); $\quad y_2 : R_2[n] \leftarrow (R_3[1])$;

$y_3 : R_3 \leftarrow$ (См); $\quad y_4 : R_5 \leftarrow (0)$;

$y_5 : R_3 \leftarrow$ Сдв. П (R_3); $\quad y_6 : $ Сч $\leftarrow n$;

$y_7 : $ Сч $\leftarrow$ (Сч) $- 1$;

$x_1 : (R_2[1]) = 1$; $\quad x_2 : $ (Сч) $= 0$.

Запись $R_2[n]$ и $R_3[1]$ означает соответственно n-й разряд регистра R_2 и 1-й разряд регистра R_3.

Мы здесь не задавались целью показать полный список используемых в операционном устройстве микроопераций. Так, рассматривая процессы с момента, когда в регистры R_1 и R_2 уже помещены соответственно множимое и множитель, мы не показываем микроопераций приема чисел в эти регистры.

СИНТЕЗ УПРАВЛЯЮЩЕГО УСТРОЙСТВА

Процесс синтеза разобьем на этапы, последовательно рассматривая каждый из них.

Построение схемы алгоритма в микрооперациях. На рис. 1.6, *а* показана эта схема алгоритма. Нетрудно понять, что она соответствует приведенному выше описанию функционирования множительного устройства. Такая форма представления функционирования устройства является более наглядной, чем словесная форма его описания.

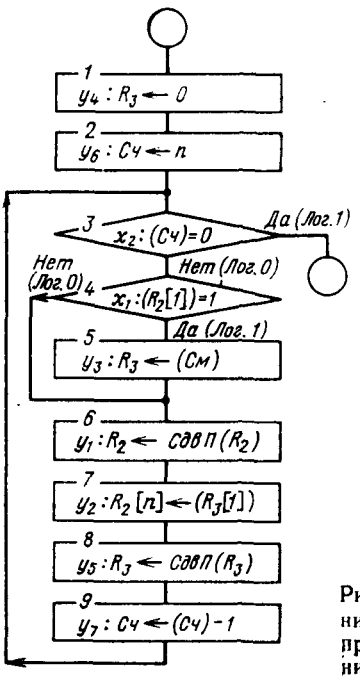
Построение схемы алгоритма в микрокомандах. Для формирования микрокоманд необходимо определить, какие микрооперации могут выполняться одновременно (в одни и те же тактовые периоды).

Очевидно, микрооперации y_4 и y_6 могут быть объединены в общую микрокоманду Y_1, микрооперация y_3 не может быть объединена

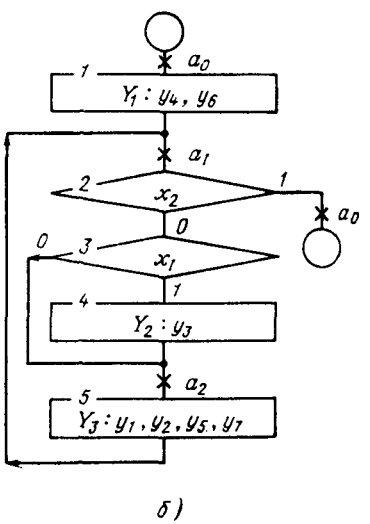
с какими-либо другими микрооперациями, и, следовательно, она одна представляет микрокоманду Y_2; микрооперации y_1, y_2, y_5, y_7 можно выполнять в приведенной на рис. 1.6, *а* последовательности в четырех тактовых периодах, но при построении регистров на триггерах, управляемых фронтом сигнала на их синхронизирующем входе, эти микрооперации могут выполняться одновременно и, следовательно, могут быть объединены в микрокоманду Y_3.

На рис. 1.6, *б* показана схема алгоритма, построенная в микрокомандах.

Построение графа функционирования. Управляющее устройство является логическим устройством последовательностного типа. Микрокоманда, выдаваемая в следующем тактовом периоде, зависит от того, какая микрокоманда выдается в текущем тактовом периоде, или, иначе, от состояния, в котором находится устройство. Для определения состояний устройства производится разметка схемы алгоритма, представленной в микрокомандах (рис. 1.6, *б*), по следующему правилу: символом a_0 отмечаются начало и конец схемы, затем последовательно отмечаются символами a_1, a_2, ... входы блоков, следующих за операторными блоками (блоками, содержащими микрокоманды). Блок *1* является операторным блоком, и отмечается символом a_1 вход следующего за ним блока — блока *2* условного перехода по признаку x_2;



а)



б)

Рис. 1.6. Схемы алгоритма функционирования управляющего устройства при выполнении операции умножения:

а) в микрооперациях; *б*) в командах (размеченная)

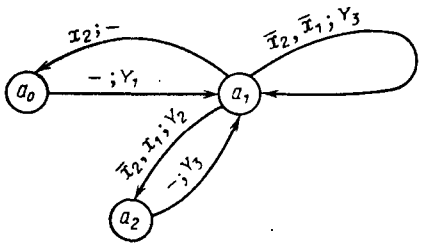


Рис. 1.7. Граф алгоритма управляющего устройства, выполняющего операцию умножения

Таблица 1.2

Состояния	Кодовые комбинации	
	Q_2	Q_1
a_0	0	0
a_1	0	1
a_2	1	0

выбирается следующий операторный блок *4* и отмечается символом a_2 вход следующего за ним блока *5*.

Полученные отметки a_0, a_1, a_2 соответствуют состояниям устройства. Итак, устройство имеет три состояния.

Теперь можно приступить к построению графа функционирования устройства. Состояния устройства в графе представляются узлами (изображаемыми кружками с записью внутри них обозначения соответствующих состояний); дугами, соединяющими узлы, показывают возможные переходы между узлами (на схеме алгоритма эти переходы соответствуют переходам между соответствующими отметками); на дугах записывают условия (значения признаков, поступающих на входы управляющего устройства с выхода операционного), при которых происходит переход, и какая микрокоманда должна выдаваться устройством.

Граф синтезируемого управляющего устройства приведен на рис. 1.7.

Кодирование состояний устройства. В процессе кодирования состояний каждому состоянию устройства должна быть поставлена в соответствие некоторая кодовая комбинация. Число разрядов кода выбирается из следующих соображений: если число состояний равно M, то для обеспечения M кодовых комбинаций требуется k-разрядный код, где k — минимальное целое число, при котором выполняется неравенство $M \leqslant 2^k$.

В рассматриваемом случае $M = 3$ и $k = 2$. Таким образом, состояния устройства отображаются двухразрядными кодовыми комбинациями.

Соответствие между состояниями устройства и кодовыми комбинациями зададим табл. 1.2.

Структурная схема управляющего устройства. Структурная схема рассматриваемого устройства представлена на рис. 1.8.

Триггеры *1* и *2* образуют двухразрядный регистр текущего состояния устройства. Комбинационный узел по состоянию регистра (комбинации значений Q_2 и Q_1) и значениям поступающих с выхода операционного устройства условий x_1 и x_2 определяет новое состояние, в ко-

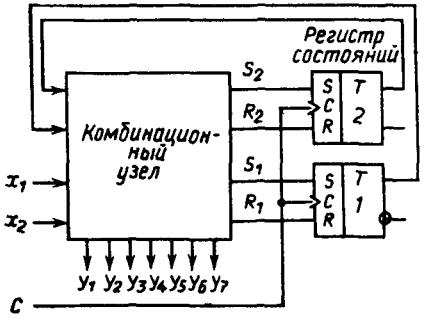


Рис. 1.8. Структурная схема управляющего устройства

торое должно перейти устройство. При этом формируются такие сигналы S_2, R_2, S_1, R_1, которые в момент действия положительного фронта синхронизирующего сигнала C устанавливают в регистре кодовую комбинацию, соответствующую следующему состоянию.

Комбинационный узел формирует также управляющие сигналы $y_1 \ldots y_7$, под действием которых в операционном устройстве выполняются микрооперации.

Дальнейшие шаги по синтезу управляющего устройства сводятся к синтезу его комбинационного узла.

Построение таблицы функционирования комбинационного узла. Таблица функционирования содержит графы, в которые заносятся данные текущего состояния, значение входных условий, данные следующего состояния, в которое должно перейти устройство, и выходные сигналы комбинационного узла. Функционирование комбинационного узла рассматриваемого управляющего устройства представлено в табл. 1.3.

По значению текущего состояния, принимаемому из регистра состояния, и поступающим из операционного устройства значениям условий x_1 и x_2 в таблице определяются следующее состояние, сигналы R_2, S_2, R_1, S_1, необходимые для установки регистра в следующее состояние, и управляющие сигналы $y_1 \ldots y_7$. Заполнение таблицы производится следующим образом. В графе следующего состояния задается состояние a_1; по графу на рис 1.7 находится дуга, ведущая в узел, соответствующий состоянию a_1; найденная дуга выходит из узла a_0,

Таблица 1.3

Текущее состояние			Следующее состояние			Условие перехода	Выходные сигналы	
Обозначение	Кодовая комбинация		Обозначение	Кодовая комбинация			Сигналы установки триггеров регистра	Управляющие сигналы микроопераций
	Q_2	Q_1		Q_2	Q_1			
a_0	0	0	a_1	0	1	—	S_1	Y_1: y_4, y_6
a_1	0	1	a_1	0	1	$\bar{x}_1$, $\bar{x}_2$	—	Y_3: y_1, y_2, y_5, y_7
a_2	1	0	a_1	0	1	—	R_2, S_1	Y_3: y_1, y_2, y_5, y_7
a_1	0	1	a_2	1	0	x_1, $\bar{x}_2$	S_2, R_1	Y_2: y_3
a_1	0	1	a_0	0	0	x_2	R_1	—

следовательно, текущее состояние a_0. Переход из a_0 в a_1 безусловный. Заносим в таблицу кодовые комбинации состояний a_0 и a_1. При этом выясняется, что переход $a_0 \rightarrow a_1$ связан с переходом $Q_1{:}0 \rightarrow 1$. Из таблицы переходов RS = триггера (табл. 1.4) определяем, что $S_1 = 1$. Кроме этого сигнала на выходе комбинационного узла должны формироваться управляющие сигналы микрокоманды Y_1: y_4, y_6.

Таблица 1.4

Вид перехода триггера	Сигналы на входах RS-триггера	
	S	R
0→0	0	×
0→1	1	0
1→0	0	1
1→1	×	0

× — знак «безразлично».

Далее в следующую строку таблицы заносятся данные, соответствующие переходу $a_1 \rightarrow a_1$. Из графа выясняется, что переход происходит при выполнении условий $\overline{x_1} = 1$ и $\overline{x_2} = 1$ с выдачей сигналов микрокоманды Y_3. Принцип заполнения строки аналогичен рассмотренному выше. Каждой из дуг графа в таблице функционирования соответствует отдельная строка. Таким образом заполняется вся таблица.

Запись логических выражений для выходных величин комбинационного узла. Для каждой строки таблицы функционирования комбинационного узла запишем логическое выражение в следующей форме: в левой части выражения перечислим переменные, приведенные в графе выходных величин, в правой части — логическое выражение, представленное через текущее состояние a_i и значения условий перехода.

Для рассматриваемого комбинационного узла получаем следующие логические выражения:

$$S_1;\ y_4, y_6 = a_0;$$

$$y_1, y_2, y_5, y_7 = \overline{x_1} \cdot \overline{x_2} \cdot a_1;$$

$$R_2, S_1;\ y_1, y_2, y_5, y_7 = a_2;$$

$$S_2, R_1;\ y_3 = x_1 \cdot \overline{x_2} \cdot a_1;$$

$$R_1 = x_2 \cdot a_1.$$

Затем определяют логическое выражение для каждой выходной величины. Для этого записывают равенство, в левой части которого указывают выходную величину, в правой части — связанные через операцию дизъюнкции правые части тех из ранее составленных выражений, в которых представлена данная выходная величина.

Полученные логические выражения приводят к минимальной форме:

$$S_2 = x_1 \cdot \overline{x_2} \cdot a_1;\ R_2 = a_2;\ S_1 = a_0 \vee a_2;$$

$$R_1 = x_1 \cdot \overline{x_2} \cdot a_1 \vee x_2 \cdot a_1;\ y_4, y_6 = a_0;$$

$$y_1, y_2, y_5, y_7 = \overline{x_1} \cdot \overline{x_2} \cdot a_1 \vee a_2;\ y_3 = x_1 \cdot \overline{x_2} \cdot a_1.$$

Построение логической схемы комбинационного узла. По полученным выражениям строится логическая схема комбинационного узла.

Входящие в выражения значения a_0, a_1, a_2, определяемые комбинацией значений Q_2 и Q_1, могут быть получены с помощью дешифратора. Остальная часть схемы строится в соответствии с полученными для выходных величин выражениями. Схема комбинационного узла рассматриваемого управляющего устройства приведена на рис. 1.9.

Выполнение программы. Мы рассмотрели реализацию управляющего устройства для выполнения операции умножения. Очевидно, могут быть построены подобные устройства для управления выполнением других операций. И если в управляющем устройстве процессора предусмотреть такие устройства, то, включая то или иное устройство, можно обеспечить выполнение различных операций на одном и том же оборудовании операционного устройства.

Вид операции, подлежащей исполнению в процессоре, будем представлять *командой*. С помощью дешифратора код комаиды можно преобразовать в сигналы, производящие включение устройств, которые управляют выполнением соответствующих операций (рис. 1.10).

При этом возникает возможность записать алгоритм сложной задачи в виде последовательиости команд (которая будет соответствовать последовательности таких выполняемых операций, как умножение, деление и т.д.). Такая последовательность команд образует *программу*, хранимую в оперативной памяти. Считывая из оперативной памяти команды и исполияя их в процессоре, можно решить сложную задачу.

Определим время, в которое реализуется алгоритм умножения при рассмотренном способе построения процессора. Обратимся к графу на рис. 1.7. Каждый переход из одного состояиия в другое происходит за одии тактовый период. Таким образом, переход из состояния a_0 в состояние a_1 требует одиого тактового периода. Далее в цикле прохождение по замкнутому контуру из состояния a_1 с возвратом в то же состояние a_1 возможно либо за один, либо за два тактовых периода (в цепи $a_1 \rightarrow a_2 \rightarrow a_1$). Приняв худший случай, когда во всех n разрядах мно-

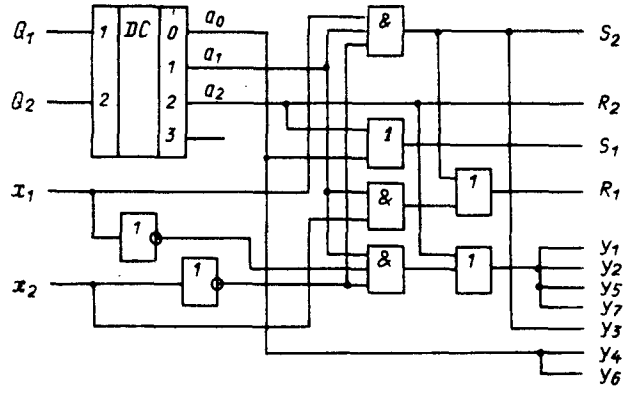


Рис. 1.9. Схема комбинациоиного узла управляющего устройства

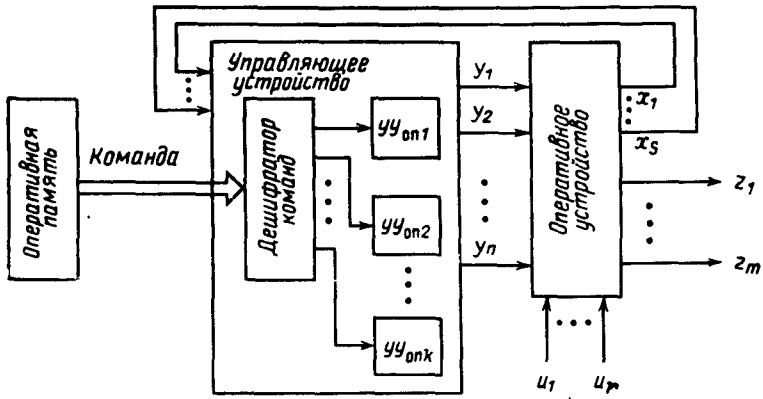


Рис. 1.10. Процессор, программируемый на языке команд

жителя содержатся единицы, получим число тактовых периодов, требуемых для n-кратного повторения цикла, равное $2n$. Наконец, в один тактовый период совершается переход из состояния a_1 в состояние a_0. Итак, общее число тактовых периодов $N_т = 2+2n$. При $n \gg 1$ $N_т \approx 2n$. Если длительность тактового периода T, то время выполнения операции умножения составит $t_{умн} = N_т T = 2nT$.

1.5. СИНТЕЗ УПРАВЛЯЮЩЕГО УСТРОЙСТВА НА ОСНОВЕ ПРОГРАММИРУЕМОЙ ЛОГИКИ

ПРИНЦИП МИКРОПРОГРАММНОГО УПРАВЛЕНИЯ

Мы видели, что выполнение операции в процессоре осуществляется в виде последовательности выполняемых микрокоманд. Можно предусмотреть другой способ формирования в управляющем устройстве управляющих сигналов, под действием которых в операционном устройстве выполняются микрокоманды.

Управляющие сигналы $y_1 \ldots y_n$ на выходе управляющего устройства в каждом тактовом периоде имеют уровни логического 0 и логической 1. Таким образом, каждой микрокоманде на выходе УУ соответствует некоторая кодовая комбинация. Такие кодовые комбинации, называемые кодовыми комбинациями микрокоманд (или просто микрокомандами), можно хранить в *управляющей памяти*. Последовательность микрокоманд, предназначенную для выполнения некоторой операции, называют микропрограммой. При этом выполнение операции сводится к выборке из управляющей памяти последовательно микрокоманд микропрограммы и выдаче с их помощью управляющих сигналов $y_1 \ldots y_n$ в операционное устройство.

В управляющей памяти можно хранить много микропрограмм, предназначенных для выполнения различных операций. По выбранной из оперативной памяти команде в управляющей памяти находится соответствующая команде микропрограмма. Далее путем последовательного считывания микрокоманд найденной микропрограммы и их выполнения в операционном устройстве реализуется предусматриваемая командой операция.

Такой способ реализации операций называется *микропрограммным* способом, а построенное на этом принципе устройство — управляющим устройством *с программируемой логикой*.

На рис. 1.11, *а* изображена структурная схема процессора с управляющим устройством, построенным на принципе программируемой логики. Функции блока микропрограммного управления (БМУ) сводятся к определению адреса очередной микрокоманды (МК) в управляющей памяти (УП). Поступающая из оперативной памяти (ОП) команда содержит адрес первой МК той микропрограммы, которая реализует предусматриваемую командой операцию. Таким образом решается проблема поиска в УП микропрограммы, соответствующей даной команде. Адреса всех последующих МК определяются в БМУ следующим образом.

В формате МК (рис. 1.11, *б*) предусматривается поле адреса, которое содержит адрес очередной МК. Таким образом, считав из УП микрокоманду, по содержимому ее поля адреса узнаем адрес следующей МК. Но так можно получать адреса МК при отсутствии в алгоритме разветвлений, т. е. условных переходов (УсП). Для реализации условных переходов в МК можно предусмотреть поле условных переходов, в котором указывается, имеет ли место условный или безусловный переход, и в случае условного перехода — на значения каких условий следует ориентироваться при определении адреса очередной МК.

Пусть поле условных переходов построено следующим образом. Один из разрядов этого поля указывает вид перехода (например, если содержимое этого разряда 0, то это означает безусловный переход, ес-

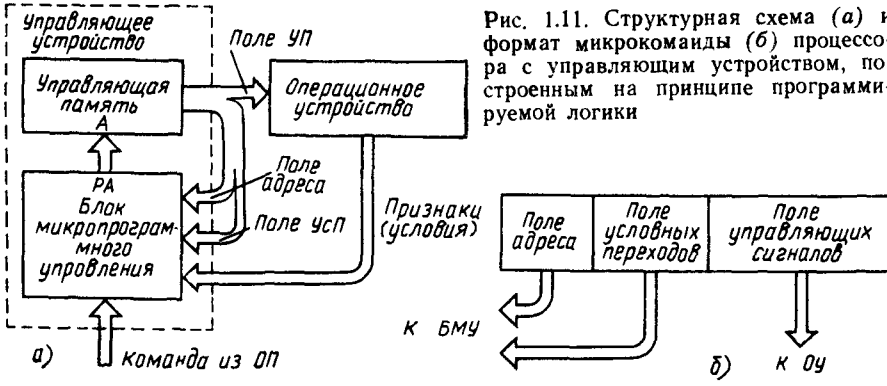


Рис. 1.11. Структурная схема (*а*) и формат микрокоманды (*б*) процессора с управляющим устройством, построенным на принципе программируемой логики

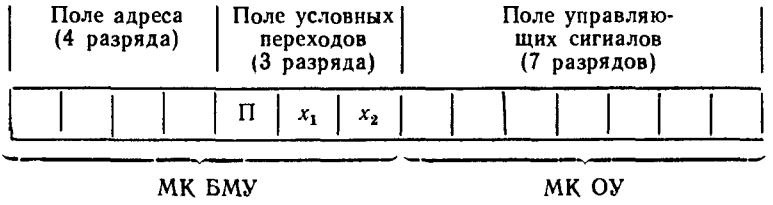
ли содержимое разряда 1 — условный переход). Кроме того, для каждого условия в поле условных переходов имеется разряд, указывающий участие данного условия в определении адреса. Если условный переход осуществляется по некоторому условию, то формирование адреса очередной МК будем осуществлять замещением младшего разряда содержимого поля адреса текущей МК значением соответствующего условия (такую операцию называют *модификацией адреса*). Получается разветвление на два направления: в зависимости от значения условия образуются два различающиеся в младшем разряде адреса и очередная МК считывается из одной либо другой ячейки УП. Если модифицировать два разряда содержимого поля адреса, то можно осуществлять разветвление в четырех направлениях.

Поле управляющих сигналов МК используется для подачи управляющих сигналов в операционное устройство ОУ.

Таким образом, микрокоманда может быть разлита на две части: одна ее часть, включающая в себя поле адреса и поле условных переходов, определяет функционирование БМУ при определении адреса очередной МК и может быть названа микрокомандой БМУ; другая часть — поле управляющих сигналов — определяет функционирование ОУ и может быть названа микрокомандой ОУ.

ПРИМЕР ПОСТРОЕНИЯ МИКРОПРОГРАММЫ

Построим микропрограмму для выполнения рассмотренной выше операции умножения. Выберем следующий формат микрокоманды:



Четырехразрядное поле адреса позволяет обращаться в любую ячейку УП с 16 ячейками (т. е. в УП с возможностью хранения до 16 микрокоманд).

Поле условных переходов содержит 3 разряда: разряд П, наличие единицы в котором указывает на то, что имеет место условный переход; разряды x_1 и x_2, наличие единицы в которых определяет условие, по которому происходит условный переход.

Поле управляющих сигналов содержит 7 разрядов и обеспечивает выдачу сигналов семи различных микроопераций.

Для хранения составляемой микропрограммы используем ячейки УП с последовательно нарастающими адресами: 0000, 0001, 0010, ... Ориентируясь на схему алгоритма, приведенную на рис. 1.6, *б*, построим схему алгоритма в микрокомандах (рис. 1.12) и таблицу размещения микрокоманд в ячейках УП (табл. 1.5).

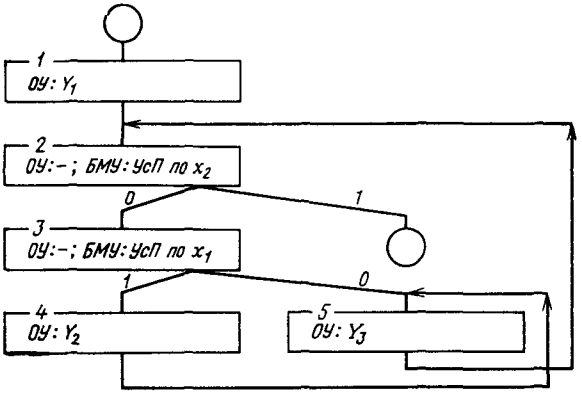


Рис. 1.12. Схема алгоритма

Стрелками в таблице показаны переходы в процессе исполнения микропрограммы.

В ячейку с адресом 0000 помещаем МК1, в которой МК ОУ определяет предусмотренные схемой алгоритма действия Y_1, МК БМУ определяет безусловный переход (БП) к ячейке с адресом 0001.

В ячейку с адресом 0001 помещена МК2, в которой МК ОУ не предусматривает действий в ОУ, МК БМУ определяет условный переход по условию x_2 (при $x_2 = 0$ происходит переход к ячейке с адресом 0010, указанным в поле адреса МК2; при $x_2 = 1$ происходит переход к ячейке с адресом 0011, в которой помещена МК6 продолжения программы после окончания выполнения операции умножения).

Микрокоманда МК3 в ячейке с адресом 0010 так же, как и МК2, не предусматривает действий в ОУ и предназначена для выполнения условного перехода по условию x_1 (значение условия x_1, замещая младший разряд содержимого поля адреса 0100, приводит при $x_1=0$ к переходу к ячейке с адресом 0100, содержащей МК5, и при $x_1 = 1$ — к ячейке с адресом 0101, содержащей МК4).

Таблица 1.5

	Адрес ячейки УП	*Содержимое ячейки (микрокоманда)*
	0 0 0 0	МК1 (МК ОУ: Y_1; МК БМУ: БП)
	0 0 0 1	МК2 (МК ОУ: –; МК БМУ: УсП по x_2
$x_2=0$	0 0 1 0	МК3 (МК ОУ: –; МК БМУ: УсП по x_1
$x_2=1$	0 0 1 1	МК6 (Продолжение)
$x_1=0$	1 0 0 0	МК5 (МК ОУ: Y_3; МК БМУ: БП)
$x_1=1$	0 1 0 1	МК4 (МК ОУ: Y_2; МК БМУ: БП)

Таблица 1.6

Адрес ячейки УП	Микрокоманда											Пояснения
	МК БМУ				МК ОУ							
	Поле адреса	Поле условного перехода			y_1	y_2	y_3	y_4	y_5	y_6	y_7	
		Вид перехода	x_1	x_2								
0000	0001	0	×	×	0	0	0	1	0	1	0	МК1
0001	0010	1	0	1	0	0	0	0	0	0	0	МК2
0010	0100	1	1	0	0	0	0	0	0	0	0	МК3
0011	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	МК6
0100	0001	0	×	×	1	1	0	0	1	0	1	МК5
0101	0100	0	×	×	0	0	1	0	0	0	0	МК4

Микрокоманда МК4 предусматривает в ОУ выполнение микрокоманды Y_2, а в БМУ — безусловный переход к ячейке с адресом 0100, хранящей МК5.

Микрокоманда МК5 предусматривает в ОУ выполнение микрокоманды Y_3, в БМУ — безусловный переход к МК2 , адрес которой содержится в поле адреса МК БМУ.

В табл. 1.6 приведена микропрограмма.

Определнм время выполнения операции умиожения. Оно равно произведению числа тактовых периодов, требуемых для выполнения операции, и длительности тактового периода. В рассматриваемом алгоритме операции умножения требуется однократное выполнение МК1 и n-кратиое (n — число разрядов множителя) выполнение микрокоманд цикла. В цикле в зависимости от значения условия x_1 выполняется либо 4 микрокоманды (МК2, МК3, МК4, МК5) либо 3 микрокоманды (МК2, МК3, МК5). Ведя расчет на наихудший случай, когда во всех разрядах множителя содержится единица, примем, что число выполняемых в цикле микрокоманд равно четырем. Тогда пренебрегая временем выполнения МК1, получаем число тактовых периодов, равиое $N_т = 4\ n$, и время выполнения операции $t_{умн} = N_т \cdot T = 4nT$ (где T — длительность тактового периода).

Вспомним, что в рассматриваемом выше способе построения процессора с использованием схемного принципа построения УУ $N_т = 2n$.

Рассмотрим, как можно было бы снизить $N_т$ (и тем самым повысить скорость выполнения операции умножения) при использовании принципа программируемой логики. На рис. 1.13 показан другой вариант схемы алгоритма, особенность которого в том, что выполняемые в ОУ микрокоманда Y_3 и в БМУ условный переход по x_2 совмещены в одной микрокоманде МК4. При этом $N_т = 3n$. В табл. 1.7 и 1.8 показаны соответственно размещение микрокоманд в УП и микропрограмма.

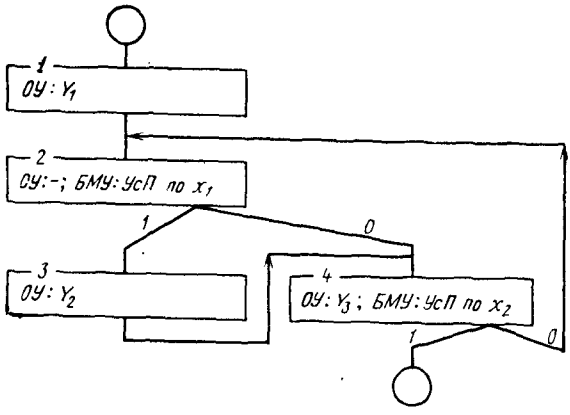


Рис. 1.13. Схема алгоритма

Дальнейшее сокращение $N_т$ возможно лишь при использовании условных переходов одновременно по двумя условиям x_1 и x_2 с выполнением разветвления в четырех направлениях, т. е. замещением не одного, а сразу двух разрядов содержимого поля адреса в МК БМУ значениями условий x_2 и x_1. На рис. 1.14 показана схема алгоритма с таким разветвлением вычислительного процесса. Из схемы видно, что в этом случае $N_т = 2$. В табл. 1.9 и 1.10 приведены соответственно размещение микрокоманд в УП и микропрограмма.

Следует обратить внимание на следующую особенность в записи микрокоманд. Запись 1 в разряде x_1 (либо x_2) микрокоманды означает, что формнрование адреса очередной микрокоманды производится путем замещения в содержимом поля адреса соответствующего разряда на значение условия x_1 (либо x_2). Не следует считать, что такая запись означает $x_1 = 1$ (либо $x_2 = 1$).

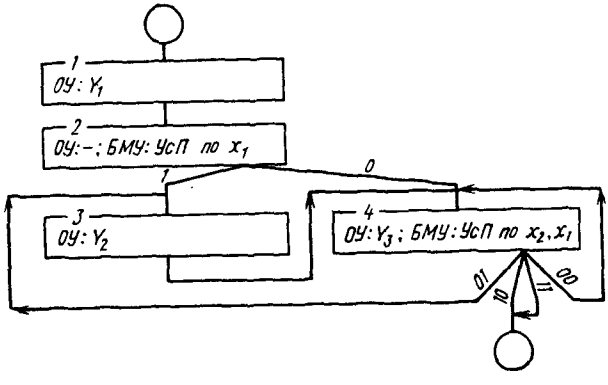


Рис. 1.14. Схема алгоритма

Таблица 1.7

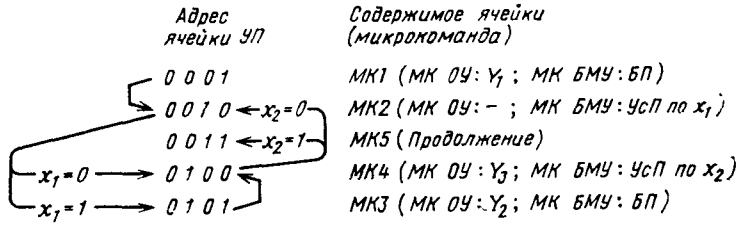


Таблица 1.8

Адрес ячейки УП	Микрокоманда												Пояснения
	МК БМУ				МК ОУ								
	Поле адреса	Поле условного перехода			y_1	y_2	y_3	y_4	y_5	y_6	y_7		
		Вид перехода	x_1	x									
0001	0010	0	×	×	0	0	0	1	0	1	0		МК1
0010	0100	1	1	0	0	0	0	0	0	0	0		МК2
0011	...	...	...	...	...	...	...	...	...	...	...		МК5
0100	0010	1	0	1	1	1	0	0	1	0	1		МК4
0101	0100	0	×	×	0	0	1	0	0	0	0		МК3

Таблица 1.9

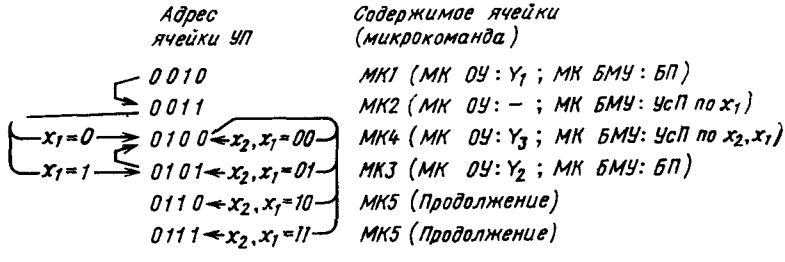


Таблица 1.10

Адрес ячейки УП	Микрокоманда												Пояснения
	МК БМУ				МК ОУ								
	Поле адреса	Поле условного перехода			y_1	y_2	y_3	y_4	y_5	y_6	y_7		
		Вид перехода	x_1	x_2									
0010	0011	0	×	×	0	0	0	1	0	1	0		МК1
0011	0100	1	1	0	0	0	0	0	0	0	0		МК2
0100	0100	1	1	1	1	1	0	0	1	0	1		МК4
0101	0100	0	×	×	0	0	1	0	0	0	0		МК3
0110	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .		МК5
0111	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .		МК5

СРАВНЕНИЕ ПО БЫСТРОДЕЙСТВИЮ ДВУХ ПРИНЦИПОВ ПОСТРОЕНИЯ УПРАВЛЯЮЩИХ УСТРОЙСТВ

Как мы убедились, использование принципа программируемой логики при построении УУ может привести к увеличению числа тактовых периодов, в которое реализуется программа, и, следовательно, к снижению быстродействия процессора.

Кроме того, быстродействие дополнительно может снизиться за счет большей длительности тактового периода. На рис. 1.15 приведена временна́я диаграмма работы процессора с УУ, построенным по принципу программируемой логики. В момент времени t_1 (на положительном фронте синхросигнала C) происходит прием в регистр адреса (РА) БМУ сформированного адреса следующей микрокоманды. Далее

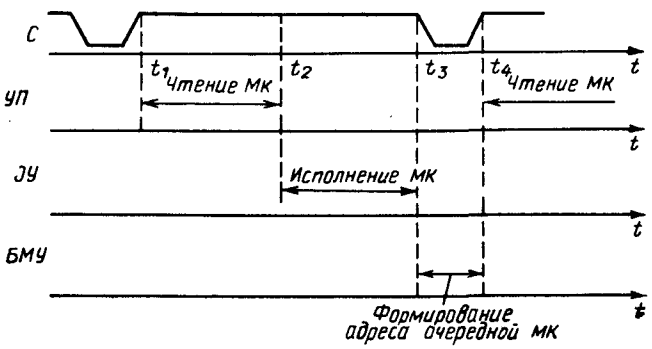


Рис. 1.15. Временны́е диаграммы работы процессора, построенного на принципе программируемой логики

адрес из РА выдается в шину адреса и поступает в УП, где происходит чтение очередной МК. Процесс чтения завершается к моменту времени t_2, и с этого момента в ОУ поступает МК ОУ и начинается процесс исполнения МК, который завершается к моменту t_3. В момент t_4 (на положительном фронте синхросигнала) полученный в ОУ результат фиксируется в соответствующем регистре. В БМУ в интервале времени t_3 ... t_4 под действием МК БМУ происходит формирование адреса очередной МК и фиксация его в момент t_4 в РА и т. д.

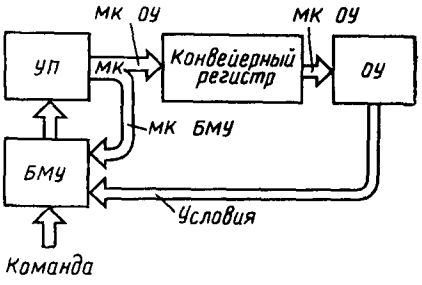


Рис. 1.16. Структура микропроцессора с конвейерным регистром

Такой последовательный способ чтения и исполнения МК вызывает увеличение длительности тактового периода на время, необходимое для чтения МК из УП, что является дополнительным фактором, снижающим быстродействие процессора.

Последний недостаток может быть устранен использованием конвейерного способа чтения и исполнения МК. При этом способе осуществляется параллельный принцип чтения и исполнения МК: в процессе исполнения в ОУ n-й МК в УП производится чтение $(n + 1)$-й МК, в том же тактовом периоде в БМУ формируется адрес $(n + 2)$-й МК. Реализация этого способа требует использования конвейерного регистра (рис. 1.16). Временна́я диаграмма работы процессора с конвейерным регистром показана на рис. 1.17.

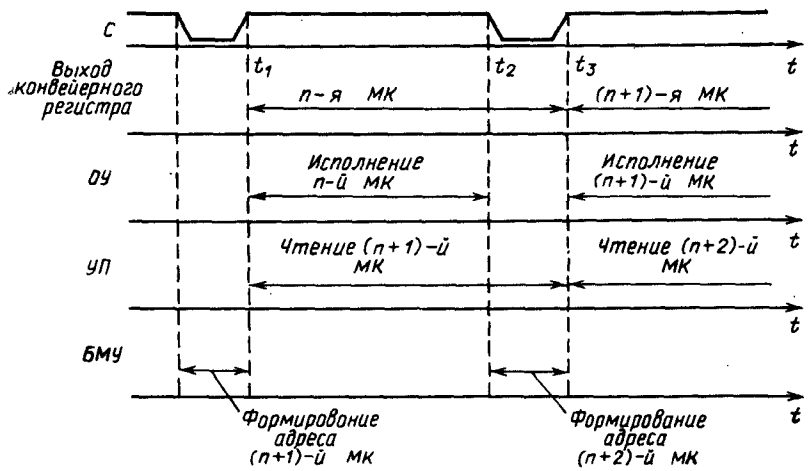


Рис. 1.17. Временны́е диаграммы работы процессора с конвейерным регистром

В интервале времени t_1 ... t_3 из конвейерного регистра выдается n-я МК ОУ, считанная из УП на предыдущем тактовом периоде. В интервале времени $t_1 \ldots t_2$ она исполняется в ОУ. В этом же тактовом интервале производится чтение из УП $(n + 1)$-й МК по адресу, сформированному в БМУ в предыдущем тактовом периоде. В интервале времени $t_2 \ldots t_3$ в БМУ формируется адрес $(n + 2)$-й МК.

При использовании схемы с конвейерным регистром длительность тактового периода может оказаться той же (при соответствующем высоком быстродействии УП), что и в устройстве, построенном на принципе схемной логики. Однако в этом случае дополнительно увеличивается число тактовых периодов, во время которых исполняется алгоритм.

Применение конвейерного регистра приводит к некоторому усложнению процесса составления микропрограммы. Особенности построения микропрограмм для таких схем будут рассмотрены в разделах, посвященных программированию микропроцессорных устройств с использованием микропрограммируемых МПК.

1.6. ПОСТРОЕНИЕ МИКРОПРОЦЕССОРОВ ПРИ ИСПОЛЬЗОВАНИИ РАЗЛИЧНЫХ МИКРОПРОЦЕССОРНЫХ КОМПЛЕКТОВ

Все элементы микропроцессора с программируемой логикой — операционное устройство (ОУ), управляющая память (УП) и блок микропрограммного управления (БМУ) — могут размещаться на одном кристалле, т. е. весь микропроцессор может быть выполнен в виде одной микросхемы. Так реализованы микропроцессоры в отечественных сериях микропроцессорных комплектов КР580 и КР1810. Управляющая память микропроцессоров такого типа хранит набор микропрограмм, записанный в нее уже на этапе изготовления микросхемы на заводе. Каждая микропрограмма представляет собой последовательность микрокоманд, обеспечивающую выполнение некоторой несложной операции. При поступлении в микропроцессор команды из оперативной памяти (ОП) в УП находится соответствующая команде микропрограмма и путем последовательного считывания ее микрокоманд осуществляется прием из ОП операндов, выполнение над ними некоторых простейших действий и вызов из ОП очередной команды. В микропроцессоре серии КР580 такие микропрограммы содержат от 4 до 17 микрокоманд. Применение микропроцессора, выполненного на одной микросхеме, естественно, упрощает построение микропроцессорного устройства, сокращая количество используемых в нем элементов. Кроме того, упрощается процесс программирования, так как от программиста не требуется записывать выполняемые в каждом также микрокоманды. Составляя программу, он оперирует командами, т. е. группами микрокоманд, которые соответствуют командам.

Однако такое облегчение программирования сопровождается существенным снижением скорости решения задачи. Это связано со следующим. Система команд, которой снабжается микропроцессор при его заводском изготовлении, универсальна в том смысле, что она позволяет программировать решение любой задачи. Но при решении конкретной задачи такая фиксированная система команд может оказаться неэффективной: пользование ею потребует большого числа команд, на выполнение которых микропроцессор будет затрачивать много времени. Программа оказывается более эффективной (требующей меньшей емкости памяти для ее хранения и меньшего времени для исполнения), если для ее построения используется специально подобранная для данной конкретной задачи система команд. Такой прием с введением новых составленных программистом команд оказывается невозможным в микропроцессорах, реализованных в виде одной микросхемы.

В тех случаях, когда требуется обеспечивать высокую скорость решения задачи, у разработчика микропроцессорного устройства возникает желание самому разработать систему команд, наилучшим образом приспособленную к решению конкретной задачи. При этом он должен знать, что ему придется преодолеть ряд трудностей, связанных с необходимостью определения состава команд и построения для каждой команды соответствующей ей микропрограммы, если программирование ведется на языке микрокоманд. Составленные таким образом микропрограммы затем записываются в постоянное запоминающее устройство управляющей памяти.

Рассмотрим, к каким изменениям в структуре микропроцессора приводит обеспечение указанной выше возможности программирования на языке микрокоманд.

При создании микросхемы приходится решать трудную проблему сокращения числа выводов. В представленном на рис. 1.18, *а* варианте с совмещением в общей микросхеме всех элементов микропроцессора (ОУ, БМУ, УП) эта задача решается обычно путем мультиплексирования шин. Например, в микропроцессоре серии КР580 для 8-разрядных выходов и входов используются общие выводы, которые переключаются в зависимости от направления передачи данных либо на ввод, либо на вывод данных; в микропроцессоре серии КР1810, оперирующем с 16-разрядными данными и 20-разрядными адресами ОП, кроме объединения входов и выходов данных, предусматривается использование этих выводов и для выдачи части разрядов адресной информации (при этом, очевидно, необходимо предусмотреть выдачу адреса и выдачу или прием данных в различные временные интервалы).

Для того чтобы разработчик микропроцессорного устройства имел возможность программировать на языке микрокоманд, он должен иметь доступ к УП для записи в нее составленных им микропрограмм. Такой доступ можно обеспечить, если УП вынести из микросхемы процессора, как показано на рис. 1.18, *б*.

Как видно из сравнения схем на рис. 1.18, *а* и *б*, вариант построения микропроцессора, представленный на рис. 1.18, *б*, потребует в

микросхеме, содержащей ОУ и БМУ, предусмотреть существенно большее число выводов, чем в микросхеме на рис. 1.18, *а*. Это связано с необходимостью предусматривать в варианте на рис. 1.18, *б* выходы для передачи в УП адреса и входы для приема микрокоманды из УП. В результате такое построение практически окажется нереализуемым из-за чрезмерно большого числа выводов, которые пришлось бы предусмотреть в микросхемах. В этом случае для сокращения числа выводов потребуется ОУ и БМУ выполнять не в общей микросхеме, а раз-

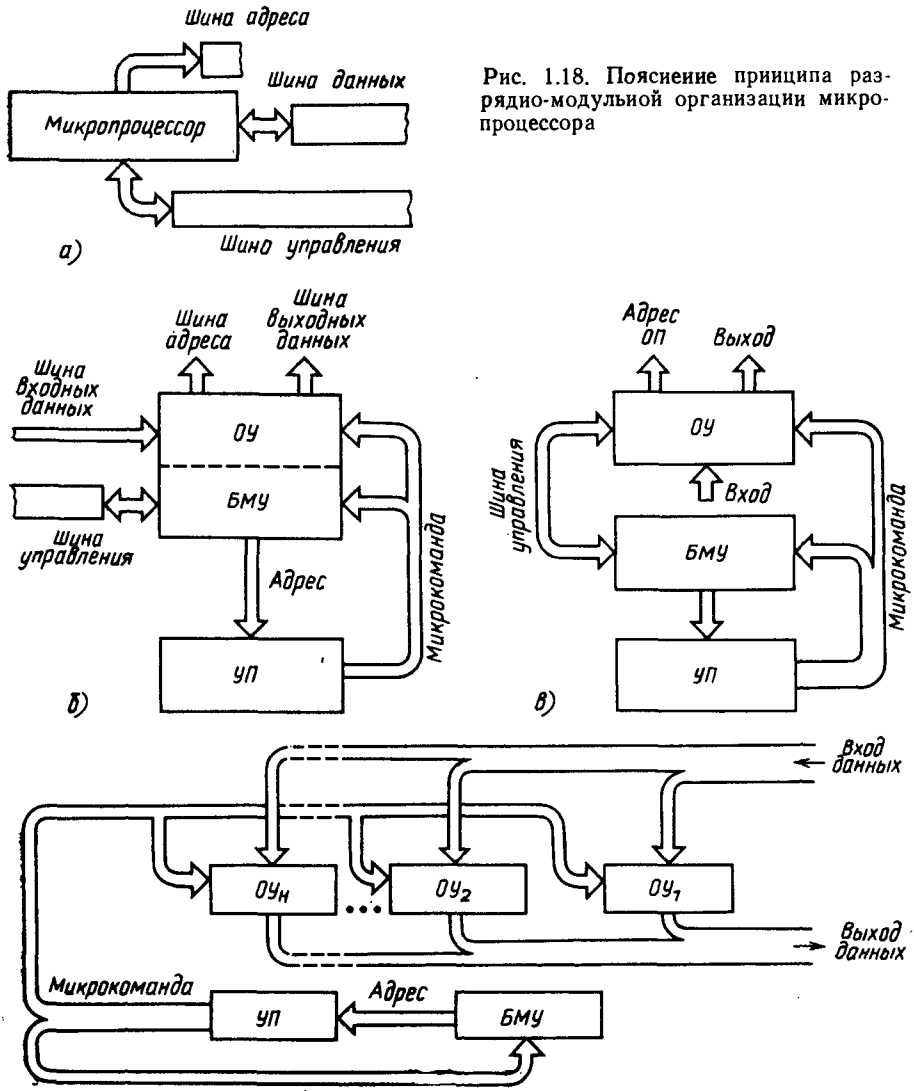


Рис. 1.18. Поясиение прииципа разрядио-модульиой организации микропроцессора

нести в разные микросхемы, как показано на рис. 1.18, *в*. Так как обеспечение высокого быстродействия требует отказа от мультиплексирования шин, то и в данном варианте число выводов в микросхеме ОУ окажется недопустимо большим. Число выводов можно сократить, если построить микросхему ОУ на небольшое число разрядов обрабатываемых данных (2-, 4-, 8-разрядных данных) и обеспечить возможность наращивать разрядность ОУ путем объединения соответствующего числа микросхем, как показано на рис. 1.18, *г*. При этом каждая микросхема, называемая центральным процессорным элементом (ЦПЭ), представляет собой не все ОУ, а лишь определенную его секцию. Так, используя восемь 4-разрядных секций, можно построить ОУ для обработки 32-разрядных данных. Подобную организацию микропроцессора называют *разрядно-модульной*. Такую организацию имеют микропроцессоры, которые строятся, например, на МПК серий К589 и К1804 (описание этих микропроцессоров приведено в гл. 5 и 6).

1.7. МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА

Рассмотрим структурную схему микропроцессорного устройства (МПУ), приведенную на рис. 1.19. Функционирование МПУ сводится к следующей последовательности действий: получение данных от различных периферийных устройств (с клавиатуры терминала, от дисплеев, из каналов связи, различного типа внешних запоминающих устройств), обработка данных и выдача результата обработки на периферийные устройства (ПУ). При этом данные от ПУ, подлежащие обработке, могут поступать и в процессе их обработки.

Для выполнения этих процессов в МПУ, кроме микропроцессора, предусматриваются следующие устройства:

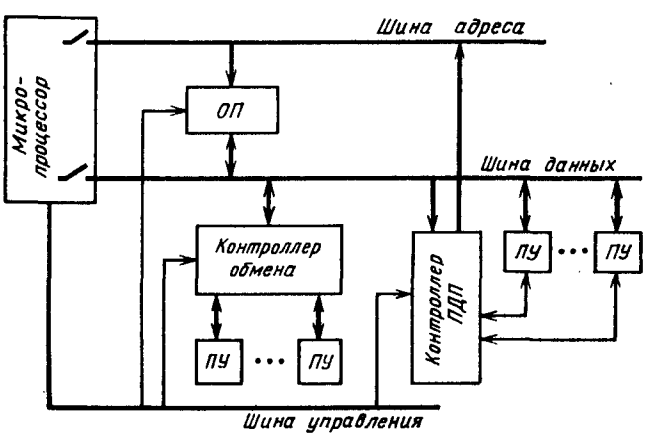


Рис. 1.19. Микропроцессорное устройство

оперативная память, предназначенная для хранения и выдачи по запросам команд программ, определяющих работу микропроцессора, различных данных (исходных данных, промежуточных и конечных результатов обработки данных в микропроцессоре), контроллеры — устройства, обеспечивающие обмен данными различных ПУ с микропроцессором и ОП.

В ходе работы микропроцессор выдает на шину адреса номер (адрес) ячейки ОП, в которой хранится очередная команда, и из шины управления в ОП поступают сигналы, обеспечивающие чтение содержимого указываемой шиной адреса ячейки памяти. Запрошенная команда оперативной памятью выдается на шину данных, откуда она принимается в микропроцессор. Здесь команда расшифровывается. Если данные, действия над которыми предусматривает команда, находятся в регистрах микропроцессора, то микропроцессор приступает к выполнению указанной в команде операции. Если же при расшифровке команды выясняется, что участвующие в операции данные находятся в ОП, то микропроцессор выставляет на шину адреса адрес ячейки, хранящей эти данные, и после их выдачи из ОП принимает их через шину данных, затем выполняется операция над данными. После завершения выполнения текущей команды на шину адреса выдается адрес следующей команды и описанный процесс повторяется.

В процессе функционирования МПУ может потребоваться выдача результата на ПУ (для управления объектами отображения на экране дисплея и т. д.) либо прием данных от ПУ (например, прием данных, набираемых оператором на клавиатуре, и т.д.). Такой обмен данными может осуществляться следующим образом.

Группа ПУ подключается к шине данных МПУ через *контроллер обмена* (*устройства сопряжения*), управляющий процессом обмена данными. До начала непосредственного обмена данными с ПУ микропроцессор через шину данных должен выдать в контроллер информацию о режимах, используемых при передаче, направлениях передачи данных (от микропроцессора к ПУ либо, наоборот, от ПУ к микропроцессору), используемых в дальнейшем при обмене данными с каждым из подключенных к контроллеру ПУ. Затем в момент, когда потребуется, например, передать в ОП выдаваемые из ПУ данные, микропроцессор, выполняя команду ввода, подает на контроллер соответствующие управляющие сигналы; данные из ПУ принимаются в регистр контроллера, откуда они затем контроллером выдаются на шину данных. Далее эти данные с шины данных принимаются в микропроцессор, после чего в процессе выполнения соответствующей команды они передаются в ОП.

Аналогично происходит обмен данными в обратном направлении — от ОП к ПУ. По соответствующей команде программы осуществляется прием из ОП в микропроцессор данных, подлежащих передаче, после чего по одной из следующих команд эти данные выдаются на шину данных и через контроллер обмена передаются на ПУ.

Описанный обмен предполагает, что моменты обмена данными известны заранее уже на этапе программирования, и в программе предусматриваются в определенных местах соответствующие команды, обеспечивающие обмен. Моменты обмена могут определяться и самим ПУ. Тогда эти моменты программисту оказываются неизвестными, и он не может в программе предусмотреть соответствующие команды обмена. В этих случаях ПУ, подавая в микропроцессор определенные сигналы, переводит его в состояние так называемого *прерывания*. В этом состоянии микропроцессор прекращает выполнение основной программы и переходит к исполнению команд другой хранящейся в ОП программы (*прерывающей программы*), обеспечивающей обмен данными, требуемый периферийным устройством. После окончания выполнения такой прерывающей программы микропроцессор возвращается к выполнению основной программы.

Описанные способы обеспечивают низкую скорость обмена и применять их целесообразно при обмене данными с низкоскоростными ПУ. При работе с высокоскоростными ПУ (такими, как запоминающие устройства на дисках и др.) используется так называемый режим *прямого доступа к памяти* (ПДП). В этом режиме микропроцессор отключается от шин адреса и данных, предоставляя их в распоряжение ПУ для непосредственного обмена данными с ОП (без участия микропроцессора). Обмен при этом организуется специальным контроллером ПДП. В режиме ПДП ПУ обменивается с ОП не одиночными данными, а большими блоками данных. В контроллер ПДП микропроцессор предварительно помещает информацию, необходимую для управления обменом (адрес ячейки ОП, куда помещается или откуда считывается первое подлежащее обмену слово, количество слов в блоке и др.). В процессе обмена контроллер ПДП выдает на шину адреса адрес ячейки ОП, после окончания передачи слова между ОП и ПУ через шину данных контроллер ПДП увеличивает на единицу значение адреса, выдаваемого на шину адреса. После завершения передачи заданного количества слов контроллер ПДП прекращает обмен, информируя об этом микропроцессор. Последний восстанавливает связь с шинами адреса и данных и продолжает выполнение программы.

1.8. МИКРОПРОЦЕССОРНЫЕ КОМПЛЕКТЫ, ВЫПУСКАЕМЫЕ ОТЕЧЕСТВЕННОЙ ПРОМЫШЛЕННОСТЬЮ

В соответствии с ОСТ 11.073.915—80 все многообразие выпускаемых отечественной промышленностью интегральных микросхем делится по конструктивно-технологическому исполнению на три группы, которым присвоены следующие обозначения: 1, 5, 6, 7 — полупроводниковые микросхемы; 2, 4, 8 — гибридные микросхемы; 3 — прочие (пленочные, вакуумные, керамические).

Таблица 1.11

Подгруппа	Вид	Буквенные обозначения подгруппы и вида
Схемы вычислительных средств	МикроЭВМ	ВЕ
	Микропроцессоры	ВМ
	Микропроцессорные секции	ВС
	Схемы микропрограммного управления	ВУ
	Функциональные расширители (в том числе расширители разрядных данных)	ВР
	Схемы синхронизации	ВВ
	Схемы управления прерыванием	ВН
	Схемы управления вводом-выводом (схемы интерфейса)	ВО
	Схемы управления памятью	ВТ
	Функциональные преобразователи информации (арифметические, тригонометрические, логарифмические, быстрого преобразования Фурье и др.)	ВФ
	Схемы сопряжения с магистралью	ВИ
	Микрокалькуляторы	ВХ
	Контроллеры	ВГ
	Комбинированные схемы	ВК
	Специализированные схемы	ВЖ
	Прочие	ВП
Многофункциональные схемы (схемы, выполняющие одновременно несколько функций)	Аналоговые	ХА
	Цифровые	ХЛ
	Комбинированные	ХК
	Прочие	ХП

По выполняемым функциям микросхемы МПК подразделяются на подгруппы, указанные в табл. 1.11.

По принятой системе обозначение микросхемы должно состоять из четырех элементов:

1) цифры, обозначающей группу микросхем;

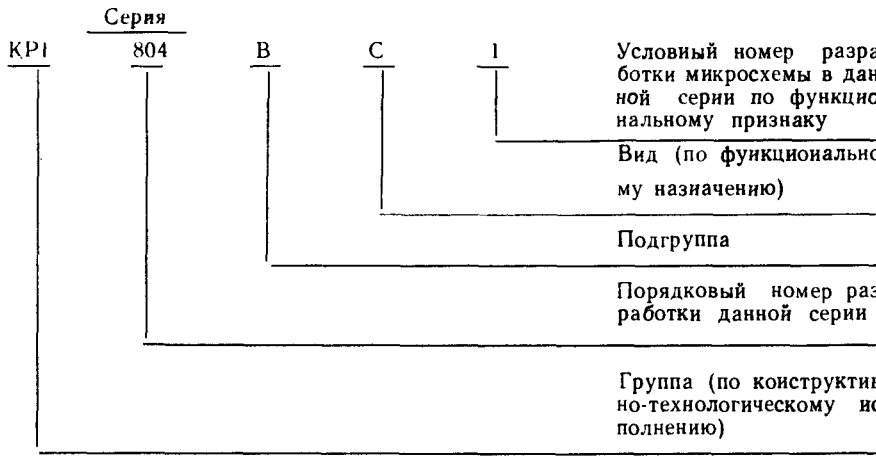
2) трех цифр (от 000 до 999) или двух цифр (от 00 или 99), обозначающих порядковый номер разработки серии микросхем;

3) двух букв, соответствующих подгруппе и виду микросхемы;

4) условного номера разработки микросхемы в данной серии по функциональному признаку.

Два первых элемента обозначают серию микросхемы (допускается трех- и четырехзначное обозначение серии).

Пример условного обозначения микросхемы КР1804ВС1 — микропроцессорной секции:



Для микросхем, используемых в устройствах широкого применения, в начале обозначения добавляется буква К или буквы КР. Как правило, микросхемы с индексом К или КР отличаются от соответствующей серии микросхем без этого индекса диапазоном температур, при котором они могут быть использованы.

Для микросхем МПК, выпуск которых начат до издания приведенной системы условных обозначений, в наименовании подгруппы и вида сохранено ранее использовавшееся обозначение ИК.

В табл. 1.12 приведены серии отечественных МПК и некоторые их данные.

Таблица 1.12

Серия МПК	Число микросхем в МПК	Обозначение ЦПЭ	Данные ЦПЭ			
			Разрядность, бит	Тактовая частота, МГц	Напряжение питания, В	Тип техиологии
КР580	15	КР580ИК80А	8	2,5	+5; +12; −5	*n*МДП
К581	3	КР581ВЕ1	16	2,5—3,3	+5; +12	*n*МДП
КР582	2	КР582ИК1	4	0,6	1,5	И²Л
К583	4	К583ИК3	8	1,0	1,5	И²Л
К584	3	К584ИК1	4	0,5	5,0	И²Л
КР588	5	КР588ВС2	16	1,0	5,0	КМДП
К1800	4	К1800ВС1	4	36	−5,2; −2	ЭСЛ
К1801	3	К1801ВЕ1	16	8,0	5,0	*n*МДП
		К1801ВМ1	16	5,0	5,0	*n*МДП
КР1802	6	КР1802ВС1	8	8,0	5,0	ТТЛШ
КР1804	6	КР1804ВС1	4	8,0	5,0	ТТЛШ
К589	8	К589ИК02	2	10	5,0	ТТЛШ

2. ПОЛУПРОВОДНИКОВЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

2.1. ТИПЫ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Для хранения небольших массивов кодовых слов могут использоваться регистры. Но уже при необходимости хранить десятки слов применение регистров приводит к неоправданно большим аппаратурным затратам. Для хранения больших массивов слов строят запоминающие устройства (ЗУ) с использованием специальных микросхем, в каждой из которых может храниться информация объемом в тысячи бит.

По выполняемым функциям различают следующие типы запоминающих устройств: оперативное запоминающее устройство (ОЗУ), постоянное запоминающее устройство (ПЗУ), перепрограммируемое постоянное запоминающее устройство (ППЗУ).

Оперативное ЗУ предназначено для использования в условиях, когда необходимо выбирать и обновлять хранимую информацию в высоком темпе работы процессора цифрового устройства. Вследствие этого в ОЗУ предусматриваются три режима работы: режим хранения при отсутствии обращения к ЗУ, режим чтения хранимых слов и режим записи новых слов. При этом в режимах чтения и записи ОЗУ должно функционировать с высоким быстродействием (обычно время чтения или записи слова в ОЗУ составляет доли микросекунды). В микропроцессорных устройствах ОЗУ используются для хранения данных (исходных данных, промежуточных и конечных результатов обработки данных).

Постоянное ЗУ предназначено для хранения некоторой однажды записанной в него информации, не нарушаемой и при отключении источников питания. В ПЗУ предусматриваются два режима работы: режим хранения и режим чтения с высоким быстродействием. Режим записи не предусматривается. Используются ПЗУ для хранения программ, по которым микропроцессорные устройства функционируют длительное время, многократно выполняя действия по одному и тому же алгоритму при различных исходных данных.

Перепрограммируемое ПЗУ в процессе функционирования микропроцессорного устройства используется как ПЗУ. Оно отличается от ПЗУ тем, что допускает обновление однажды занесенной информации, т. е. в нем предусматривается режим записи. Однако в отличие от ОЗУ запись информации требует отключения ППЗУ от микропроцессорного устройства, производится с использованием специальных предназначенных для записи устройств (программаторов) и занимает длительное время, достигающее десятков минут. Перепрограммируемые ПЗУ дороже ПЗУ, и их применяют в процессе отладки программы, после чего их можно заменить более дешевым ПЗУ.

2.2. ОСНОВНЫЕ ПАРАМЕТРЫ ЗАПОМИНАЮЩИХ УСТРОЙСТВ

Запоминающее устройство содержит некоторое число N ячеек, в каждой из которых может храниться слово с определенным числом разрядов n. Ячейки последовательно нумеруются двоичными числами. Номер ячейки называется адресом. Если для представления адресов используются комбинации m-разрядного двоичного кода, то число ячеек в ЗУ может составить $N = 2^m$.

Количество информации, которое может храниться в ЗУ, определяет его *емкость*. Емкость можно выражать числом ячеек N с указанием разрядности n хранимых в них слов в форме $N \times n$ либо ее можно определять произведением N и n: $M = N \cdot n$ бит. Часто разрядность ячеек выбирают кратным байту (1 байт равен 8 битам). Тогда и емкость удобно представить в *байтах*. Большие значения емкости часто выражаются в единицах $К = 2^{10} = 1024$. Например, $M = 64$ Кбайт означает емкость, равную $M = 64 \cdot 1024$ байт $= 64 \cdot 1024 \cdot 8$ бит.

Быстродействие ЗУ характеризуется двумя параметрами: *временем выборки* $t_в$, представляющим собой интервал времени между моментом подачи сигнала выборки и появлением считанных данных на выходе, и *циклом записи* $t_{цз}$, определяемым минимально допустимым временем между моментом подачи сигнала выборки при записи и моментом, когда допустимо последующее обращение к памяти.

Запоминающие устройства строятся из набора однотипных микросхем ЗУ с определенным их соединением. Каждая микросхема ЗУ, кроме времени обращения и емкости, характеризуется потребляемой мощ-

Таблица 2.1

Мнкросхема	Информационная емкость, бит (организация, слово × разряд)	Время выборки, нс	Напряжение источников питания, В	Потребляемая от источников мощность, мВт	Число выводов
К155РУ5	256 (256×1)	60	5	700	16
КР188РУ2А	256 (256×1)	500	5	$P_{ст}=0{,}05$ $P_{дин}=10$	16
К500РУ410	256 (256×1)	25	—5,2	750	16
К500РУ415	1024 (1024×1)	30	—5,2	730	16
КР565РУ2А	1024 (1024×1)	450	5	300	16
КР537РУ2А	4096 (4096×1)	300	5	50	18
КР541РУ1А	4096 (4096×1)	120	5	450	18
КР565РУ1А	4096 (4096×1)	200	12; 5; —5	3; 0,25; 0,125	22
КР541РУ31	8192 (8192×1)	150	5	550	20
КР541РУ3	16 384 (16 384×1)	150	5	550	20
КР581РУ4	16 384 (16 384×1)	200	12; 5; —5	500; 0,05; 2	22

Таблица 2.2

Микросхема	Информационная емкость, бит (организация, слово × разряд	Время выборки, нс	Напряжение источников питания, В	Потребляемая от источников мощность, мВт	Число выводов
К155РЕ3	256 (32×8)	50	5	550	16
К500РЕ149	1024 (256×4)	35	—5,2	730	16
К541РТ1	1024 (256×4)	80	5	400	16
КР556РТ4	1024 (256×4)	70	5	650	16
КР556РТ5	4096 (512×8)	70	5	950	24
КР565РТ1	4096 (1024×4)	300	—12; 5; —5	$P_{ст}$: 3; 10; 0,5 $P_{дин}$: 130; 1; 95	22

ностью, набором питающих напряжений, типом корпуса (числом выводов). Микросхемы ППЗУ дополнительно характеризуются временем хранения записанной в них информации (по истечении которого хранящаяся в ячейках информация может самопроизвольно измениться), допустимым количеством циклов перезаписи (после чего микросхема считается негодной для использования).

Перечень и основные характеристики разных типов ЗУ, рекомендуемых для широкого использования, приведены в табл. 2.1 — 2.3: табл. 2.1 — ОЗУ, табл. 2.2 — ПЗУ с однократным электрическим программированием, табл. 2.3 — ППЗУ.

Таблица 2.3

Микросхема	Информационная емкость, бит (организация, слово × разряд)	Время выборки, нс	Время хранения, ч	Число циклов перезаписи	Напряжение источников, В	Потребляемая мощность, мВт	Число выводов
ПЗУ с многократным электрическим перепрограммированием							
К505РР4А	512 (256×2)	1200	3000	10^4	—9; 5	350; 200	24
К505РР4	1024 (512×2)	1200	3000	10^4	—9; 5	350; 200	24
КР558РР11	1024 (256×4)	5000	3000	10^4	—12; 5	120; 50	24
КР558РР1	2048 (256×4)	5000	3000	10^4	—12; 5	120; 50	24
ПЗУ с ультрафиолетовым стиранием и электрической записью							
К573РФ1	8192 (1024×8)	450	15 000	10	12; —5; 5	850; 225; 75	24
К573РФ11	4096 (512×8)	450	15 000	10	12; —5; 5	850; 225; 75	24
К573РФ13	4096 (1024×4)	450	15 000	10	12; —5; 5	850; 225; 75	24
К573РФ2	16 384 (2048×8)	900	10 000	10	5	225	24
К573РФ21	8192 (1024×8)	900	10 000	10	5	225	24
К573РФ23	8192 (2048×4)	900	10 000	10	5	225	24

2.3. ОПЕРАТИВНЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

На рис. 2.1 приведена типичная структура микросхемы ОЗУ. Информация хранится в накопителе. Он представляет собой матрицу, составленную из *элементов памяти* (ЭП), расположенных вдоль строк и столбцов. Элемент памяти может хранить 1 бит информации (лог. 0 либо лог. 1). Кроме того, он снабжен управляющими цепями для установки элемента в любой из трех режимов: режим хранения, в котором он отключается от входа и выхода микросхемы; режим чтения, в котором содержащаяся в ЭП информация выдается на выход микросхемы; режим записи, в котором в ЭП записывается новая поступающая со входа микросхемы информация.

Каждому ЭП приписан номер, называемый *адресом* элемента. Для поиска требуемого ЭП указываются строка и столбец, соответствующие положению ЭП в накопителе. Адрес ЭП в виде двоичного числа принимается по шине адреса в регистр адреса. Число разрядов адреса связано с емкостью накопителя. Число строк и столбцов накопителя выбираются равными целой степени двух. И если число строк $N_{стр} = 2^{n_1}$ и число столбцов $N_{столб} = 2^{n_2}$, то общее число ЭП (емкость накопителя)

$$N = N_{стр} N_{столб} = 2^{n_1} \cdot 2^{n_2} = 2^{n_1+n_2} = 2^n,$$

где $n = n_1 + n_2$ — число разрядов адреса, принимаемого в регистр адреса.

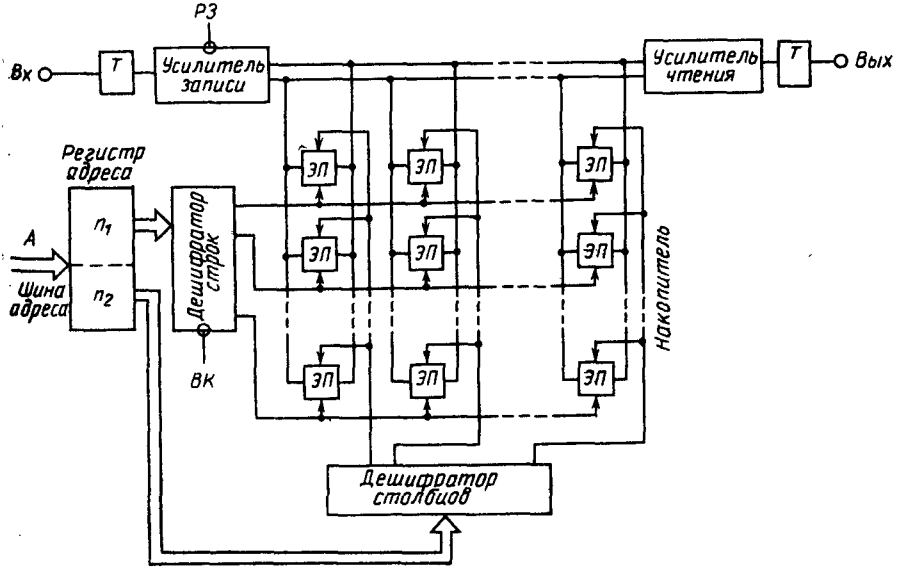


Рис. 2.1. Структура микросхемы ОЗУ

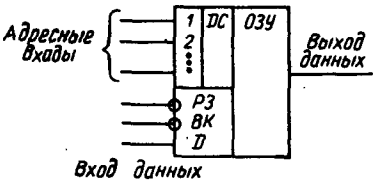


Рис. 2.2. Условное обозначение микросхемы ОЗУ

Например, при емкости $N = 2^{10} = 1024$ число разрядов адреса $n = 10$; при этом выбирается $n_1 = n_2 = n/2 = 5$, в этом случае число строк и число столбцов накопителя равно $2^{n_1} = 2^{n_2} = 32$.

Разряды регистра адреса делятся на две группы: одна группа в n_1 разрядов определяет двоичный номер строки, в которой в накопителе расположен ЭП, другая группа в n_2 разрядов определяет двоичный номер столбца, в котором расположен выбираемый ЭП. Каждая группа разрядов адреса подается на соответствующий дешифратор: дешифратор строк и дешифратор столбцов. При этом каждый из дешифраторов создает на одной из своих выходных цепей уровень лог. 1 (на остальных выходах дешифратора устанавливается уровень лог. 0); выбранный ЭП оказывается под воздействием уровня лог. 1 одновременно по цепям строки и столбца. При чтении содержимое ЭП выдается на усилитель чтения и с него на выходной триггер и выход микросхемы. Режим записи устанавливается подачей сигнала на вход разрешения записи (РЗ). При уровне лог. 0 на входе РЗ открывается усилитель записи и бит информации со входа данных поступает в выбранный ЭП и запоминается в нем.

Указанные процессы происходят в том случае, если на входе выбора кристалла (ВК) действует активный уровень лог. 0. При уровне лог. 1 на этом входе на всех выходах дешифратора устанавливается уровень лог. 0 и ЗУ оказывается в режиме хранения.

На рис. 2.2 показано условное графическое обозначение микросхемы ОЗУ.

Рассмотрим последовательность подачи сигналов в режимах чтения и записи. На рис. 2.3, *а* представлена временная диаграмма сиг-

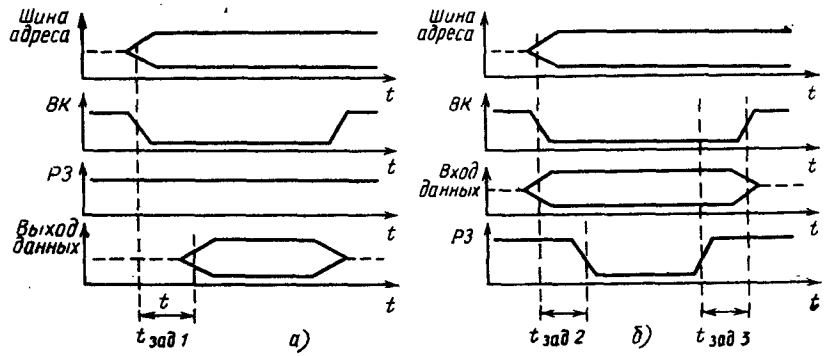


Рис. 2.3. Временные диаграммы сигналов:
а) в режиме чтения; *б)* в режиме записи

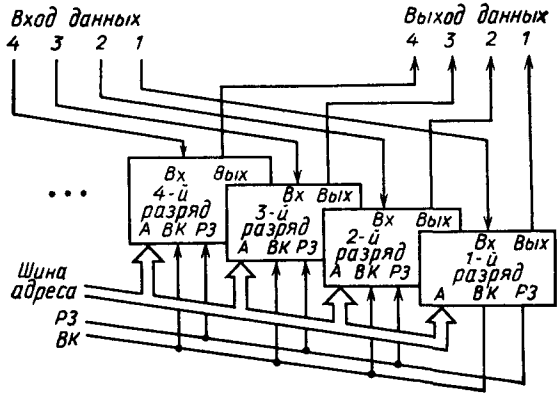


Рис. 2.4. Схема наращивания разрядности ячеек ЗУ

налов в режиме чтения. С определенной задержкой $t_{зад1}$ относительно момента подачи адреса и сигнала в цепь ВК (связанной с процессами дешифрации адреса и включения выходных цепей выбранного ЭП) на выходе микросхемы возникает содержимое выбранного ЭП. В режиме записи (рис. 2.3, *б*) должны быть соблюдены условия, которые исключили бы нарушение содержимого ячеек, в которые не производится обращение. Это обеспечивается тем, что сигнал в цепь РЗ подается с задержкой $t_{зад\,2}$ относительно момента подачи сигналов в цепи адреса, ВК и входных данных и снимается сигнал в цепи РЗ прежде, чем будет снят сигнал в цепи ВК. В противном случае, при преждевременной подаче сигнала РЗ, может произойти запись в ячейку с адресом, не совпадающим с информацией на адресных входах микросхемы.

Микросхемы ОЗУ допускают наращивание емкости памяти путем наращивания разрядности (и, следовательно, разрядности хранимых в них слов) и наращивания числа ячеек (и, значит, числа слов, которые можно хранить в памяти).

Таким образом, используя соответствующее число микросхем в определенном соединении, можно построить память с требуемой организацией.

Рассмотрим схему наращивания разрядности ячеек (рис. 2.4). На все микросхемы подается один и тот же адрес. При чтении каждой микросхемой выдается определенный разряд считываемого слова. При записи входное слово поразрядно заносится в ЭП отдельных микросхем. Таким образом, если микросхемы имеют организацию $N \times 1$ (N одноразрядных ячеек), то для блока памяти с организацией $N \times n$ (N ячеек с разрядностью каждой из них, равной n) потребуется n микросхем.

На рис. 2.5 показана схема наращивания числа и разрядности ячеек. Блок памяти состоит из микросхем, образующих отдельные линейки (ряды), каждая из которых строится по схеме наращивания разряд-

ности (рис. 2.4). Разряды адреса блока памяти в этом случае делятся на две группы A_1 и A_2. Группа разрядов A_2 определяет номер линейки, группа разрядов A_1 — номер ячейки в выбранной линейке. Выбор линейки осуществляется с помощью дешифратора, на вход которого подается A_2, а каждый из выходов подключен к входу ВК определенной линейки. Таким образом, в зависимости от кодовой комбинации, содержащейся в A_2, на соответствующем выходе дешифратора появляется уровень лог. 0, который обеспечнвает выбор определенной линейки микросхем. На входы ВК остальных линеек с выходов дешифратора поступает уровень лог. 1, и микросхемы этих линеек устанавливаются в режим хранения, в котором они не реагируют на адресную группу A_1.

Рассмотрим пример наращнвания емкости блока памяти. Пусть на микросхемах с организацией 1024 × 1 необходимо построить блок памяти, имеющий организацию 4096 × 8, т. е. блок памяти на 4096 8-разрядных ячеек. Наращивание разрядности потребует в каждой линейке схемы на рис. 2.5 использовать 8 микросхем; для увеличения числа ячеек с 1024 до 4096 (в 4 раза) необходимо предусмотреть 4 линейки микросхем. Таким образом, общее число требуемых микросхем 8×4= = 32. Адрес, по которому в таком блоке памяти будет производиться обращение, формируется следующим образом. Для выбора линейки в адресе потребуется двухразрядная группа A_2, каждой из четырех кодовых комбинаций этой группы (00, 01, 10, 11) будет соответствовать определенная линейка в блоке памяти. Выбор ячейки в линейке микросхем потребует наличия в адресе 10-разрядной группы A_1 (число комбинаций 10-разрядной группы $2^{10} = 1024$ равно числу ЭП в микросхеме). Таким образом, адрес рассматриваемого блока памяти должен иметь 12 разрядов.

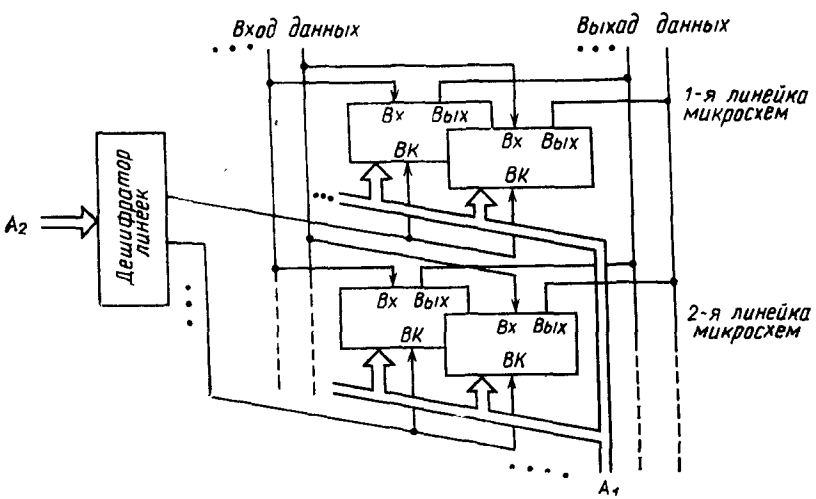


Рис. 2.5. Схема иаращивания числа и разрядности ячеек ЗУ

В каждом столбце матрицы микросхем на рис. 2.5 выходы всех микросхем объединяются в цепь соответствующего разряда выхода данных блока, все входы данных — в цепь соответствующего разряда входа данных блока памяти.

2.4. ПОСТОЯННЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Как и ОЗУ, ПЗУ состоит из ячеек, обратившись к которым, можно вывести их содержимое. Отличие от ОЗУ состоит в том, что информация в ячейкн записывается однократно, после чего в процессе эксплуатации используется лишь режим чтения.

По способу занесения информации ПЗУ делятся на два вида: ПЗУ, программируемые маской на предприятии-изготовителе, и ПЗУ, программируемые пользователем.

В первых информация заносится в процессе изготовления микросхемы с помощью соответствующего фотошаблона. Очевидно, такой способ записи пригоден в тех случаях, когда производится выпуск крупной партии ПЗУ с одной и той же записанной в них информацией. Промышленность выпускает такие ПЗУ, например, для использования в качестве преобразователя двоичного кода в определенные двоично-десятичные коды и других преобразователей. В них входная кодовая комбинация служит адресом ячейки, а содержимое ячейки — выходной кодовой комбинацией (являющейся, например, кодовой комбинацией двоично-десятичного кода).

В ПЗУ, программируемых пользователем, запись информации производится непосредственно пользователем с помощью специальных устройств, называемых программаторами. Программатор выдает в микросхему соответствующие напряжения для записи информации, набираемой на клавиатуре либо предварительно нанесенной путем пробивок на перфоленту. Этими напряжениями осуществляется прожигание плавких перемычек в элементах памяти. Очевидно, однажды записанная в ПЗУ информация в дальнейшем не может быть изменена. При необходимости изменить содержимое ПЗУ микросхемы с ранее записанной информацией заменяются новыми, в которые записываются новые данные.

На рис. 2.6 приведена структура ПЗУ, программируемого пользователем. Как и в ОЗУ, матрица-накопитель состоит из элементов памяти (ЭП), образующих строки и столбцы, но в отличие от ОЗУ при считывании из накопителя выдается содержимое целой строки элементов памяти. Такая строка обычно содержит несколько слов. С помощью селектора из строки выделяется и передается на выход требуемое слово.

Пусть, например, ПЗУ имеет емкость $M = 2^{10}$ бит, разбивающихся на $N = 2^8$ слов по $2^2 = 4$ разрядов в каждом слове. Накопитель будет содержать 2^{10} элементов памяти, расположенных вдоль $2^5 = 32$ строк и $2^5 = 32$ столбцов. При обращении должен указываться адрес слова;

этот адрес в рассматриваемом примере будет содержать 8 разрядов, разбивающихся на две группы разрядов A_2 и A_1: 5-разрядную группу A_1 и 3-разрядную группу A_2.

Группа A_1 подается на дешифратор $Дш_1$, который выбирает одну из $2^5 = 32$ строк накопителя. Содержимое строки состоит из 32 бит или восьми 4-разрядных слов. Номер слова в строке задается группой A_2. Дешифратор $Дш_2$ преобразует эту адресную группу в сигнал на одном из восьми своих выходов. По этому сигналу в селекторе из содержимого строки выделяется требуемое слово, которое передается через буфер ввода-вывода на выход микросхемы.

На рис. 2.7 показана принципиальная схема накопителя и селектора для рассматриваемого выше примера. Схема построена на биполярных транзисторах (биполярные транзисторы используются для построения ПЗУ с высоким быстродействием).

Накопитель содержит 2^{10} транзисторов (элементов памяти), образующих 32 строки и 32 столбца. Коллекторы транзисторов накопителя подключены к источнику питания (для упрощения рисунка эти цепи, соединяющие коллекторы транзисторов с источником питания, не показаны). В цепь эмиттера каждого транзистора включена плавкая перемычка (на рисунке перемычки показаны кружками). Перемычка изготовляется из нихрома, поликремния или титаната вольфрама и имеет сопротивление в несколько десятков ом. При программировании для пережигания перемычки достаточно через транзистор пропустить импульс тока 20 ... 30 мА длительностью порядка 1 мс. При работе в режиме чтения токи в транзисторах накопителя существенно меньше,

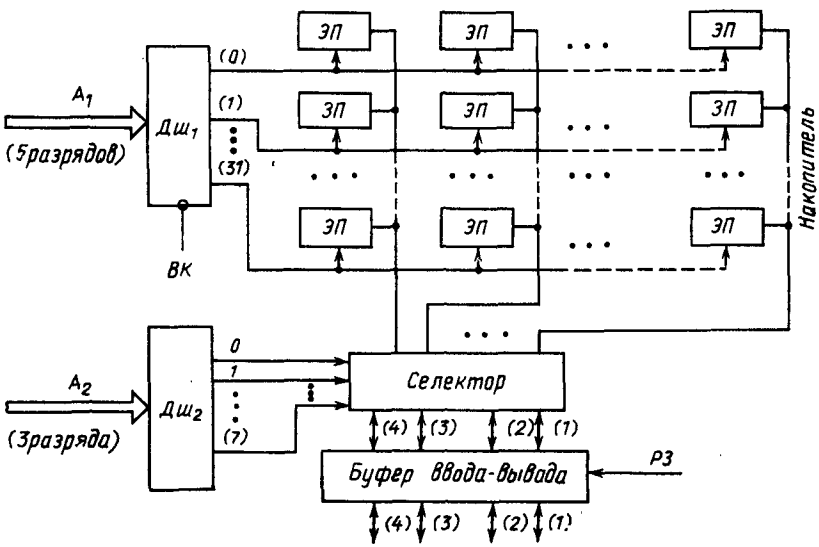


Рис. 2.6. Структура ПЗУ, программируемого пользователем

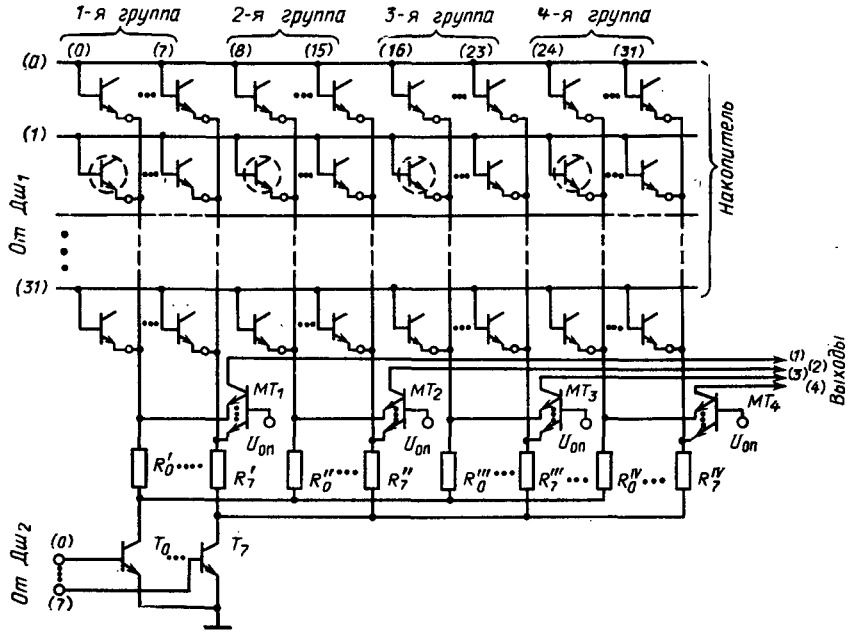


Рис. 2.7. Принципиальная схема накопителя и селектора ПЗУ

и они не могут вызвать прожигание тех перемычек, которые в процессе программирования оставлены непережженными.

Транзисторы VT_0 ... VT_7 работают в схеме селектора, многоэмиттерные транзисторы MT_1 ... MT_4 — в схеме буфера ввода-вывода.

Рассмотрим процессы при записи информации. На входы дешифраторов адреса подаются адресные группы A_1 и A_2 и на одном из выходов каждого дешифратора образуется уровень лог. 1. Пусть от дешифратора $Дш_1$ уровень лог. 1 поступает в строку с номером 1, а от дешифратора $Дш_2$ уровень лог. 1 поступает в цепь с номером 0 и в селекторе открывается транзистор VT_0. При этом в накопителе окажутся открытыми обведенные штриховой линией (рис. 2.7) транзисторы (назовем эти транзисторы выбранными). Далее повысим напряжение $U_{оп}$ и подадим на выходы микросхемы кодовую комбинацию, записываемую в выбранную четверку разрядов накопителя (выводы микросхемы, при чтении используемые в качестве выходов считываемого слова, при записи используются в качестве входов для подачи записываемого слова). Пусть на второй выход микросхемы подан уровень лог. 1. При этом открывается многоэмиттерный транзистор MT_2 в буфере ввода-вывода; эмиттерный ток этого транзистора, протекая через резистор R_0'', создает напряжение, запирающее второй транзистор в выбранной четверке транзисторов накопителя. Таким образом, состояние транзисторов выбранной четверки определяется записываемым словом (выбранный

транзистор открыт, если в соответствующем разряде записываемого слова содержится лог. 0, и, наоборот, этот транзистор закрыт, если разряд записываемого слова содержит лог. 1). Затем повысим значение напряжения коллекторного питания транзисторов накопителя. Через открытые выбранные транзисторы потечет большой ток, вызывающий пережигание перемычек в эмиттерной цепи этих транзисторов. Итак, перемычка в цепи эмиттера выбранного транзистора пережигается, если на соответствующий выход подан лог. 0.

Таким образом может быть записана требуемая информация во все элементы накопителя.

Рассмотрим процессы при чтении информации из ПЗУ. При подаче адреса (адресных групп A_1 и A_2) происходит, как уже рассматривалось выше, выборка определенной четверки транзисторов накопителя. Если перемычка в цепи эмиттера выбранного транзистора не пережжена, ток этого транзистора создает на резисторе напряжение, запирающее соответствующий многоэмиттерный транзистор; если же перемычка пережжена, то многоэмиттерный транзистор открыт. Открытое либо закрытое состояние многоэмиттерных транзисторов $MT_1 \ldots MT_4$ определяет значение разрядов считанного слова.

Рассмотрим ПЗУ типа К556РТ4 и его программирование. Микросхема ПЗУ (рис. 2.8, *а*) имеет организацию 256 × 4 (см. табл. 2.2). В соответствии с этим в ней предусмотрено 8 адресных входов (выводы микросхемы 5,6, 7, 4, 3, 2, 1, 15) и 4 входа-выхода данных (выводы 12, 11, 10, 9), являющиеся выходами в режиме чтения и входами в режиме записи. Вывод 16 используется для подключения источника питания, вывод 8 — общий, вывод 14 (С) — программирующий.

Режим чтения устанавливается подачей напряжения 5 В на вывод 16, на выводы 13 ($\overline{\text{ВК}}$) и 14 (С) — напряжения уровня лог. 0. Выходы данных 12, 11, 10, 9 построены по схеме с открытым коллектором, поэтому для снятия данных они требуют включения по схеме, показанной на рис. 2.8, *б*.

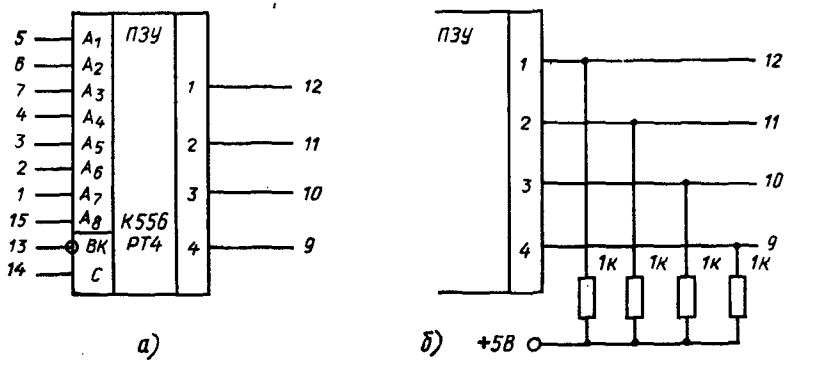


Рис. 2.8. Микросхема ПЗУ К556РТ4:
а) условное обозначение; *б)* схема включения выходов типа «открытый коллектор»

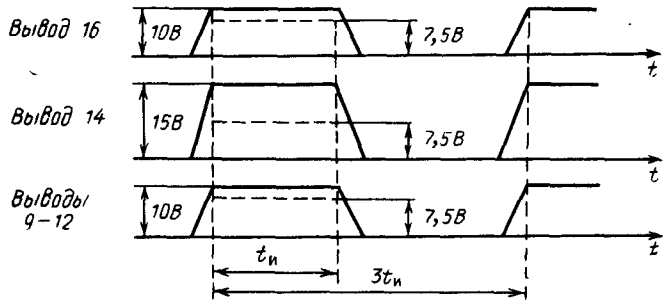


Рис. 2.9. Временны́е диаграммы сигналов при записи информации в ПЗУ

При программировании (в режиме записи) подаются импульсы в соответствии с временно́й диаграммой, показанной на рис. 2.9:

1) на адресных входах устанавливается адрес ячейки;

2) напряжение питания (на выводе 16) повышается от 5 В до 10 В (источник питания должен быть рассчитан на ток не менее 400 мА);

3) на программирующий вывод 14 подается напряжение 15 В (ток источника должен быть ограничен уровнем 100 мА);

4) на программируемый вывод через резистор 300 Ом подается напряжение 10 В (при записи лог. 1). В одном цикле можно программировать только один разряд.

Наращивание емкости ПЗУ производится по тем же схемам, что и наращивание емкости ОЗУ.

2.5. ПЕРЕПРОГРАММИРУЕМЫЕ ПОСТОЯННЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

Перепрограммируемые ПЗУ обладают всеми достоинствами ПЗУ, храня записанную в них информацию неопределенно долго и при отключенном питании. В то же время они допускают стирание записанной информации и запись новой информации. Однако если чтение осуществляется за доли микросекунды, то запись требует на много порядков большего времени.

Рассмотрим принцип работы приведенного на рис. 2.10, *а* элемента памяти с электрической записью информации и стиранием ультрафиолетовым светом. Транзистор VT_1 служит для выборки элемента памяти. Хранение информации осуществляется в транзисторе VT_2. Особенность транзистора VT_2, структура которого показана на рис. 2.10, *б*, состоит в том, что он имеет изолированный затвор. При подаче достаточно большого напряжения к p — n -переходу истока либо стока происходит инжекция электронов в затвор, после чего этот заряд может удерживаться на затворе длительное время. Отрицательный заряд на затворе, притягивая дырки, создает в n-области проводящий p-канал

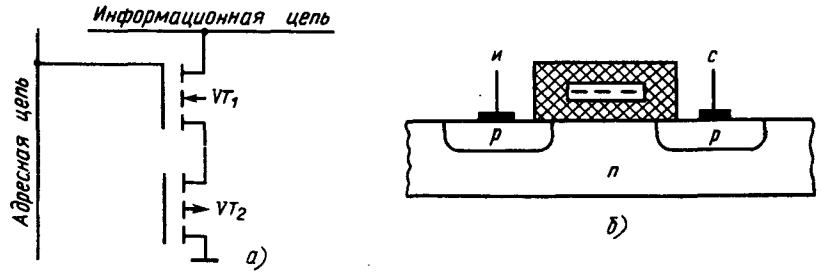


Рис. 2.10. Элемент памяти ППЗУ:
а) схема; *б)* структура

между истоком и стоком. Транзистор оказывается в состоянии лог. 0 Если же к р — *n*-переходу не прикладывалось повышенного напряжения, заряд на затворе отсутствует, транзистор оказывается в непроводящем состоянии (состоянии лог. 1).

Стирание информации в одних микросхемах производится путем подачи соответствующих напряжений, в других — путем подачи ультрафиолетового излучения через прозрачную кварцевую крышку в корпусе микросхемы. Под действием напряжений либо светового излучения, действующего в течение примерно 10 мин, сиимается заряд с затворов транзисторов и все транзисторы накопителя оказываются установленными в непроводящее состояние. Обычное комнатное освещение практически не оказывает влияния на состояние транзисторов.

Рассмотрим микросхему ППЗУ типа К573РФ1 (рис. 2.11) и его программирование.

Данная микросхема имеет организацию 1024×8 (см. табл. 2.3), в ней предусмотрено 10 адресных входов (выводы с номерами 8, 7, 6, 5, 4, 3, 2, 1, 23, 22) и 8 входов-выходов данных (выводы 9, 10, 11, 13, 14 15, 16, 17), совместимые с ТТЛ-логикой.

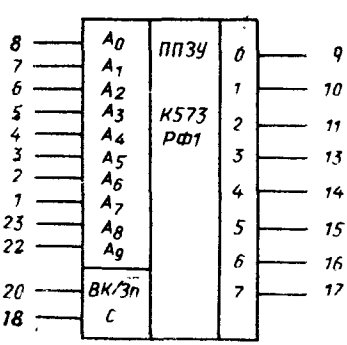


Рис. 2.11. Условное обозначение мнкросхемы К573РФ1

Режим чтения информации. Микросхема требует трех источников питания: +12 В (вывод 19), +5 В (вывод 24), —5 В (вывод 21) относительно общего вывода 12. На входе $\overline{ВК}/Зп$ (вывод 20) и программирующем входе (вывод 18) устанавливается напряжение уровня лог. 0. Выводы 9... 11, 13 ... 17 используются в качестве выходов данных, на которых возникает кодовая комбинация содержимого ячейки памяти, адрес которой подан на адресные входы микросхемы.

Режим стирания информации. Стирание информации осуще-

ствляется подачей на микросхему излучения высокой интенсивности с длиной волны не более 400 мкм через прозрачную для лучей крышку. В качестве источника ультрафиолетового излучения могут быть использованы лампы типа ДРТ-220 или ДРТ-375. В процессе стирания выводы микросхемы должны быть закорочены. После стирания во всех разрядах всех ячеек памяти микросхемы устанавливается лог. 1.

Режим записи информации. Цикл записи начинается подачей на вход $\overline{ВК}$/Зп (вывод 20) напряжения +12 В. На адресных входах последовательно устанавливаются адреса ячеек памяти, записываемая в ячейки информация в виде параллельных 8-разрядных кодовых комбинаций подается на выводы 9 ... 11, 13 ...17, которые в режиме записи используются в качестве входов записываемых данных. После подачи кодовых комбинаций адреса и записываемого числа с некоторой временной задержкой подается программирующий импульс уровня 26 В на программирующий вход С (вывод 18). Временная диаграмма импульсов в режиме записи представлена на рис. 2.12. Подача по одному программирующему импульсу по каждому из адресов определяет цикл записи. В течение каждого цикла записи должны программироваться все ячейки памяти. Программирование отдельных ячеек или групп ячеек недопустимо. Необходимое число циклов программирования памяти определяется следующей формулой:

$$N = 100 \text{ мс}/t_{и},$$

где $t_{и}$ — длительность программирующего импульса, мс. Так, если $t_{и} = 0{,}5$ мс, то для уверенной записи информации потребуется $N = 200$ циклов записи.

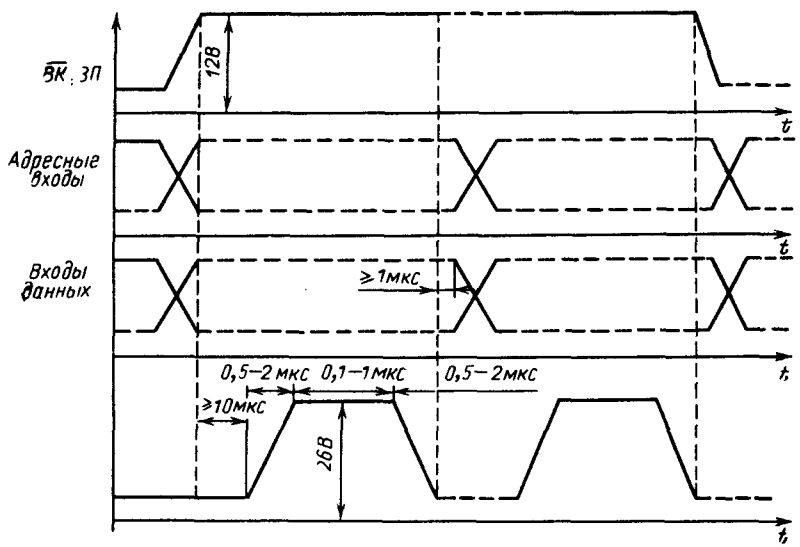


Рис. 2.12. Временные диаграммы процесса записи информации в ППЗУ

Перепрограммируемые ПЗУ дороже ПЗУ и их применяют в процессе отладки микропроцессорных устройств, когда необходимо уточнить информацию, которая должна храниться в памяти. После отладки ППЗУ можно заменить более дешевым ПЗУ.

3. МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА НА ОСНОВЕ МПК СЕРИИ КР580

3.1. СОСТАВ МИКРОПРОЦЕССОРНОГО КОМПЛЕКТА

Микропроцессорный комплект серии КР580 содержит набор БИС для построения микропроцессорных устройств относительно невысокого быстродействия, работающих с тактовой частотой до 2,5 МГц. В основном на МПК данной серии строятся микропроцессорные устройства, решающие задачи, связанные с управлением разнообразными технологическими процессами.

Комплект имеет следующие особенности. В нем предусмотрена БИС центральиого процессора, содержащая в одной микросхеме операционное и управляющее устройства. Это существенно упрощает построение микропроцессорного устройства. Кроме того, из соображений упрощения программирования для управления микросхемами МПК применяется фиксированный набор команд. Использование такого фиксированного набора команд, облегчая составление программ, в то же время является фактором, снижающим быстродействие микропроцессорного устройства. Это связано с тем, что предлагаемый пользователю стандартиый набор команд может оказаться плохо приспособленным для описания действий, требуемых для решения конкретной задачи.

Все микросхемы, входящие в состав МПК, выполнены по *n*МДП-технологии, однако входные и выходные сигналы соответствуют уровням логических схем ТТЛ-технологии. Это упрощает переходы между микросхемами серии КР580 и микросхемами ТТЛ-технологии любых серий. Следовательно, не возникает трудностей, если при построении микропроцессорного устройства используются также некоторые микросхемы ТТЛ-технологии, имеющие широкое применение.

Микросхемы МПК КР580 и их функции представлены в табл. 3.1.

Микросхемы характеризуются следующими параметрами:

температурный диапазон: КР580 —10 ... +70° С; 580 —60 ... ... + 85° С;

потребляемая мощность: КР580ИК80 $\leqslant$ 1,25 Вт; 580ВМ80 $\leqslant$ 1,7 Вт; остальные БИС $\leqslant$ 0,7 Вт;

напряжение питания: КР580ИК80, 580ВМ80 +5, +12, —5 В, остальные БИС + 5 В;

допустимое отклонение напряжения: КР580 ±5 %; 580 ±10%;

нагрузочная способность каждого выхода БИС — один вход элемента ТТЛ;

время спада и нарастания входных напряжений на выводах БИС ⩽ 30 нс.

Таблица 3.1

Тип микросхемы	Назначение микросхемы	Выполняемая функция
КР580ИК80А 580ВМ80	Восьмиразрядный параллельный центральный процессор	Центральный процессор с фиксированной системой команд для обработки параллельной 8-разрядной информации
КР580ИК51 580ВВ51	Программируемый последовательный интерфейс	Универсальное синхронно-асинхронное программируемое приемно-передающее устройство последовательной связи
КР580ВИ53 580ВИ53	Программируемый таймер	Формирует программно-управляемые временные задержки для синхронизации управляемых объектов в реальном масштабе времени
КР580ИК55 580ВВ55	Программируемый параллельный интерфейс	Программируемый ввод-вывод параллельной информации различного формата
КР580ИК57 580ВТ57	Программируемый контроллер прямого доступа к памяти	Высокоскоростной обмен информацией между памятью МПУ и периферийными устройствами
КР580ВН59 580ВН59	Программируемый контроллер прерываний	Обслуживает до 8 запросов на прерывание от внешних устройств
КР580ГФ24 580ГФ24	Генератор тактовых импульсов	Формирует две последовательности тактовых импульсов, необходимые для работы центрального процессора
КР580ВК28 580ВК28 КР580ВК38 580ВК38	Системный контроллер	Формирует сигналы, предназначенные для управления различными устройствами, входящими в состав МПУ
КР580ВА86	Шинный формирователь	Двунаправленный 8-разрядный шинный формирователь с неинвертирующим выходом с высокой нагрузочной способностью и тремя состояниями
КР580ВА87	Шинный формирователь	То же с инвертирующим выходом
КР580ИР82	Буферный регистр	8-разрядный буферный регистр с неинвертирующим выходом с тремя состояниями
КР580ИР83	Буферный регистр	То же с инвертирующим выходом
КР580ВГ75	Программируемый интерфейс ЭЛТ	Контроллер вывода информации из памяти МПУ на экран ЭЛТ
КР580ВГ79	Программируемый интерфейс клавиатуры и дисплея	Контроллер ввода-вывода для клавиатуры и дисплея

3.2. МИКРОПРОЦЕССОР КР580ИК80

БИС КР580ИК80 представляет собой 8-разрядный процессор, в котором совмещены операционное и управляющее устройства. Управляющая память недоступна пользователю, в ней уже в процессе изготовления БИС записываются микропрограммы операций (микропрограммы, по которым выполняются команды). Таким образом, предусматривается использование некоторой фиксированной системы команд, в которую пользователь не может внести изменений. В связи с этим данный микропроцессор относится к числу немикропрограммируемых, т. е. программируемых не на уровне микрокоманд, а на уровне команд.

СТРУКТУРНАЯ СХЕМА

На рис. 3.1 приведена структурная схема БИС КР580ИК80. Опишем кратко ее узлы.

Регистры данных. Для хранения участвующих в операциях данных предусмотрено семь 8-разрядных регистров. Регистр *A*, называемый аккумулятором, предназначен для обмена информацией с внешними устройствами (т. е. либо содержимое этого регистра может быть выдано на выход, либо со схода в него может быть принято число),

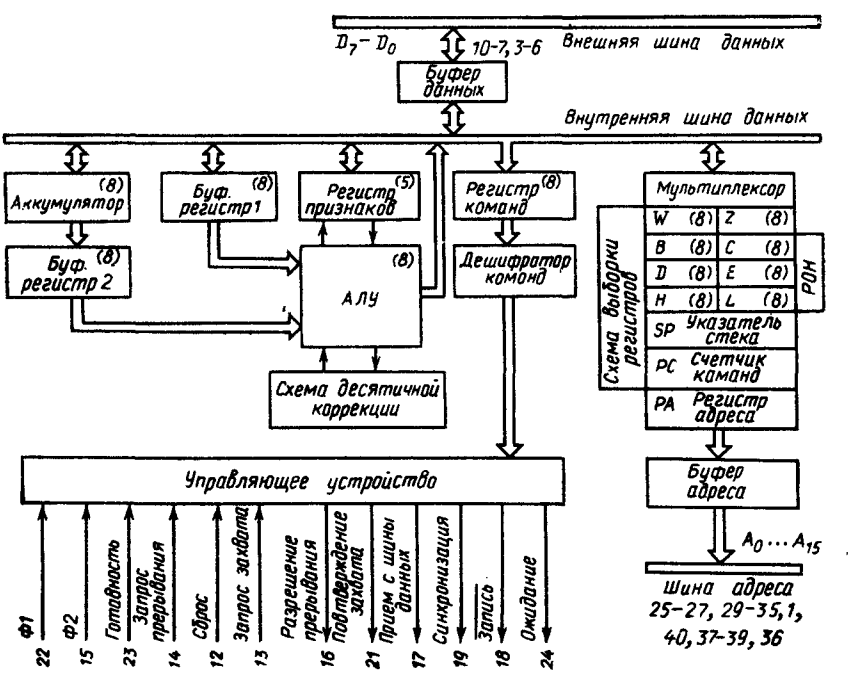


Рис. 3.1. Структурная схема БИС КР580ИК80

при выполнении арифметических, логических операций и операций сдвига он служит источником операнда (числа, участвующего в операции), в него помещается результат выполненной операции.

Шесть других регистров, обозначенных B, C, D, E, H, L, образуют так называемый блок регистров общего назначения РОН (название связано с тем, что эти регистры могут использоваться для хранения как данных, так и адресов). Эти регистры могут использоваться как одиночные 8-разрядные регистры. В случаях, когда возникает необходимость хранить 16-разрядные двоичные числа, они объединяются в пары BC, DE, HL.

Регистры $БР_1$, $БР_2$, W, Z используются как буферные, программно-недоступные регистры (т. е. регистры, к которым программист при составлении программы не может обращаться).

Указатель стека SP (16-разрядный) служит для адресации особого вида памяти, называемого *стеком* (организация стека будет рассмотрена ниже).

Счетчик команд PC (16-разрядный) предназначен для хранения адреса команды; после выбора из оперативной памяти текущей команды содержимое счетчика увеличивается на единицу и таким образом формируется адрес очередной команды (при отсутствии безусловных и условных переходов).

При обращении к памяти в качестве адреса может использоваться и содержимое любой пары регистров блока РОН.

При выдаче адреса содержимое соответствующих регистров передается в 16-разрядный регистр адреса РА, из которого далее через буферы БА адрес поступает на 16-разрядную шину адреса. С этой шины адрес может быть принят в оперативную память. Число кодовых комбинаций 16-разрядного адреса равно 2^{16}, каждая из этих кодовых комбинаций может определять адрес (иомер) одной из ячеек оперативной памяти. Таким образом, обеспечивается возможность обращения к памяти, содержащей до $2^{16} = 2^6 \cdot 2^{10} = 64К$ 8-разрядных слов (байт).

Арифметическо-логическое устройство (АЛУ). В 8-разрядном АЛУ предусмотрена возможность выполнения четырех арифметических операций (сложение с передачей переноса в младший разряд и без учета этого переноса, вычитание с передачей заема в младший разряд и без него), четырех видов логических операций (операций конъюнкции, дизъюнкции, неравнозначности, сравнения), а также четырех видов циклического сдвига.

При выполнении арифметических и логических операций одним из операндов служит содержимое аккумулятора и результат выполненной операции помещается в аккумулятор. Циклический сдвиг выполняется только над содержимым аккумулятора.

Предусмотрена возможность выполнения арифметических операций над десятичными числами. При хранении десятичного числа разряды регистра делятся на две группы по 4 разряда и в каждой группе разрядов хранится одна десятичная цифра, представленная в *коде*

8421. Таким образом, в регистре можно хранить 2-разрядное десятичное число. Как указывалось в § 3.4, при выполнении операции суммирования десятичных цифр может потребоваться коррекция результата путем прибавления к нему числа 0110_2. Такая коррекция результата в каждой 4-разрядной группе результата в микропроцессоре выполняется схемой десятичной коррекции (СДК).

Регистр признаков (РП). Этот 5-разрядный регистр предназначен для хранения определенных признаков, выявляемых в числе, которое представляет собой результат выполнения некоторых операций. Пять триггеров этого регистра имеют следующее назначение:

триггер Tc (триггер переноса) устанавливается в состояние, соответствующее переносу из старшего разряда при выполнении арифметических операций и содержимое выдвигаемого из аккумулятора разряда при выполнении операции сдвига;

триггер Tz (триггер нуля) — устанавливается в состояние лог. 1, если результат операции АЛУ или операции приращения содержимого регистра равен нулю;

триггер Ts (триггер знака) — устанавливается в состояние, соответствующее значению старшего разряда результата операции АЛУ или операции приращения содержимого регистра;

триггер Tp (триггер четности) — устанавливается в состояние лог. 1, если число единиц в разрядах результата четно;

триггер Tv (триггер дополнительного переноса) — хранит перенос, возникающий при выполнении операции из 4-го разряда.

Блок управления. Состоит из регистра команд, куда принимается первый байт команды, и управляющего устройства, формирующего управляющие сигналы, под действием которых выполняются микрооперации в отдельных узлах. Управляющее устройство содержит выполненную на программируемой логической матрице управляющую память, в которой хранятся микропрограммы отдельных операций. Однако, как уже указывалось, пользователь не может изменить содержимого управляющей памяти, а значит, и состава команд.

Буферы. Буферы данных и буферы адреса обеспечивают связь центрального процессора с внешними шинами данных и адреса. Особенность буферов состоит в том, что в каждом разряде они используют логические элементы с тремя состояниями. В них, кроме состояний лог. 0 и лог. 1, предусмотрено еще третье состояние, в котором они имеют практически бесконечное выходное сопротивление и оказываются отключенными от соответствующих шин. Использование таких буферов позволяет процессору отключаться от внешних шин (шин данных и адреса), предоставляя их в распоряжение внешних

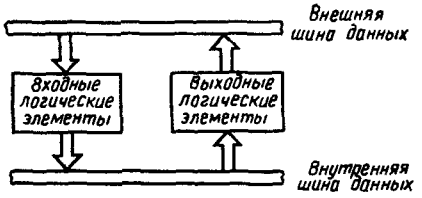


Рис. 3.2. Принцип двунаправленной передачи между внутренней и внешней шинами данных

устройств, а также позволяет использовать одну и ту же шину данных как для приема данных (т. е. в качестве входной шины), так и для выдачи данных (т. е. в качестве выходной шины). Такое использование шины данных позволяет сократить число выводов микросхемы.

На рис. 3.2 показан принцип двунаправленного обмена данными между внутренней и внешней шинами данных. Если осуществляется прием данных (передача данных с внешней шины данных на внутреннюю шину данных), отключаются, переходя в третье состояние, выходные логические элементы; при выдаче данных (передаче с внутренней шины на внешнюю шину) отключаются входные логические элементы.

ФОРМАТ ДАННЫХ И КОМАНД

Данные (обрабатываемая информация и результаты обработки) хранятся в оперативной памяти и в процессоре в виде 8-разрядных двоичных чисел. Таким образом, слово данных имеет следующий формат:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

(байт данных)

Для команд используются одно-, двух-, трехбайтовые форматы. Однобайтовый формат команды:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

(код операции)

Большинство команд является однобайтовыми.

Двухбайтовый формат команды:

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
B_1									(код операции)
B_2									(операнд или номер устройства ввода-вывода)

В первом байте двухбайтовой команды указывается вид выполняемой операции, во втором байте приводится число, являющееся операндом при выполнении операции, либо номером устройства ввода или вывода при обмене данными.

Трехбайтовый формат команды:

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
B_1									(код операции)
B_2									(младший байт адреса или младший байт операнда)
B_3									(старший байт адреса или старший байт операнда)

Байты трехбайтовой команды имеют следующее назиачение: в первом указывается вид выполняемой операции, следующие два байта используются для указания двухбайтового адреса команды (при выполнении безусловных, условных переходов, обращении к подпрограммам), или адреса ячейки оперативной памяти, содержимое которого является операндом, или двухбайтового операнда. Во всех случаях байт B_2 является младшим, байт B_3 — старшим.

СПОСОБЫ АДРЕСАЦИИ

Для того чтобы могла быть выполнена определенная операция, в команде, кроме вида операции, должно содержаться указание, откуда берутся участвующие в операции числа и куда помещается результат выполненной операции (иначе говоря, указание об источниках и приемнике операндов). Под способами адресации понимают способы, используемые для указания источников и приемников операндов.

В микропроцессоре используются следующие способы адресации.

Прямая адресация. При этом способе адресом операнда является указанный в команде (в байте кода операции) адрес регистра микропроцессора. Адреса регистров приведены в следующей таблице.

Регистр	Адрес регистра, используемый в командах	Регистр	Адрес регистра, используемый в командах
B	0 0 0	H	1 0 0
C	0 0 1	L	1 0 1
D	0 1 0	M	1 1 0
E	0 1 1	A	1 1 1

Под M понимается ячейка оперативной памяти, адресом которой служит содержимое пары регистров HL.

Покажем некоторые примеры команд с прямой адресацией, взятых из приведенного в табл. 3.2 списка команд микропроцессора.

Мнемоника команды	Кодовая комбинация команды	Выполняемая операция
MOV C, D	0 1 0 0 1 0 1 0	C ← (D)
ADD D	1 0 0 0 0 0 1 0	A ← (A) + (D)

Здесь под мнемоникой команды понимают сокращенное ее обозначение, облегчающее запоминание команды.

В кодовой комбинации команды 01 001 010 два старших разряда (01) определяют вид операции (операция пересылки содержимого одиого регистра в другой), в последующих двух 3-разрядных группах (001 и 010) приведены адреса регистров C и D. Команда представляет операцию пересылки в регистр C содержимого регистра D.

В команде 10 000 010 пять старших разрядов (10 000) представляют вид выполняемой операции (операция суммирования); в трех младших разрядах (010) указан адрес регистра D, служащего источником операнда. При выполнении операции суммирования источником другого операнда и приемником результата выполненной операции является аккумулятор A.

Непосредственная адресация При этом способе адресации операнды (один либо два) задаются непосредственно в команде вслед за байтом кода операции во втором либо втором и третьем байтах.

Ниже приведены примеры команд с непосредственной адресацией.

1) ADI	B_1	11	000	110	$A \leftarrow (A) + <B_2>$
	B_2	01	001	100	
2) MVI D	B_1	00	010	110	$D \leftarrow <B_2>$
	B_2	01	001	110	
3) LXI D	B_1	00	010	001	$D \leftarrow <B_3>$; $E \leftarrow <B_2>$
	B_2	01	100	101	
	B_3	10	100	101	

Команда с мнемоникой ADI предусматривает суммирование содержимого аккумулятора с числом, приведенным во втором байте команды (в примере это число равно $4C_{16}$).

Команда MVI производит пересылку числа, приведенного во втором байте команды (в примере это число равно $4E_{16}$), в регистр D, адрес которого (010) указан в разрядах $D_5D_4D_3$ первого байта команды.

Команда LXI производит пересылку чисел, приведенных во втором и третьем байтах (в примере — чисел 65_{16} и $A5_{16}$), соответственно в младший и старший регистры пары регистров DE. В разрядах $D_5D_4D_3$ первого байта пара регистров указана адресом (010) одного из регистров этой пары.

Косвенная адресация. При этом способе адресации в команде указывается пара регистров блока РОН (путем указания адреса одного из регистров этой пары), содержимое которой служит адресом, по которому в оперативной памяти находится операнд.

Примеры команд с косвенной адресацией:

1) LDAX	B	00	001	010	$A \leftarrow [(BC)]$
2) STAX	B	00	000	010	$[(BC)] \leftarrow (A)$

Здесь запись [(BC)] означает ячейку памяти, адресом которой служит содержимое пары регистров BC.

Таблица 3.2. **Система команд микропроцессора КР580ИК80.** Основные обозначения

Обозначение	Содержание
$\langle B_i \rangle$	Содержимое i-го байта команды
R	Обозначение регистра

R	Наименование регистра
0 0 0	B
0 0 1	C
0 1 0	D
0 1 1	E
1 0 0	H
1 0 1	L
1 1 0	M (память)
1 1 1	A (аккумулятор)

Обозначение	Содержание
F	Обозначение состояния регистра признаков

Триггеры	Условие истинности
Tc (перенос)	Наличие переноса из 8-го разряда
Tz (нуль)	Нулевой результат
Ts (знак)	Старший разряд результата равен 1
Tp (четность)	Число единиц в результате четно
Tv (дополнительный перенос)	Наличие переноса из 4-го разряда

(R)	Содержимое регистра R
$[(R_iR_{i+1})]$	Содержимое ячейки памяти, адрес которой (R_iR_{i+1})
$\wedge$	Логическое И
$\veebar$	Исключающее ИЛИ
$\vee$	Логическое ИЛИ
(A_m)	m-й бит содержимого аккумулятора
PC	Счетчик команд
SP	Указатель стека
$\leftarrow$	Пересылка

Продолжение табл. 3.2

Структура кода команды	Байты	Циклы	Такты	Выполняемая операция	Мнемоника	Признаки Z	S	C	V	P
				I. Команды пересылки						
01 R_iR_j	1	1/2	5/7	1. Регистр — регистр $R_i \leftarrow (R_j)$	MOV R_i, R_{i+1}	—	—	—	—	—

00 R_i 110	2	2/3	7/10	2. Непосредственная загрузка регистра $R_i \leftarrow \langle B_2 \rangle$	MVI R_i	—	—	—	—	—
00 R_i 001	3	3	10	3. Непосредственная загрузка пары регистров $R_i \leftarrow \langle B_3 \rangle$; $R_{i+1} \leftarrow \langle B_2 \rangle$ при $R_i = 110$: SP $\leftarrow \langle B_3 \rangle \langle B_2 \rangle$	LXI R_i	—	—	—	—	—
				4. Запоминание/загрузка A и HL						
00 K_1 010										
	1	2	7	0 0 0 $[(BC)] \leftarrow (A)$	STAX B	—	—	—	—	—
				0 1 0 $[(DE)] \leftarrow (A)$	STAX D	—	—	—	—	—
				0 0 1 $A \leftarrow [(BC)]$	LDAX B	—	—	—	—	—
				0 1 1 $A \leftarrow [(DE)]$	LDAX D	—	—	—	—	—
	3	4	13	1 1 0 $[\langle B_3 \rangle \langle B_2 \rangle] \leftarrow (A)$	STA	—	—	—	—	—
				1 1 1 $A \leftarrow [\langle B_3 \rangle \langle B_2 \rangle]$	LDA	—	—	—	—	—
	3	5	16	1 0 0 $[\langle B_3 \rangle \langle B_2 \rangle] \leftarrow (L)$; $[\langle B_3 \rangle \langle B_2 \rangle] + 1 \leftarrow (H)$	SHLD	—	—	—	—	—
				1 0 1 $L \leftarrow [\langle B_3 \rangle \langle B_2 \rangle]$; $H \leftarrow [\langle B_3 \rangle \langle B_2 \rangle] + 1$	LHLD	—	—	—	—	—
				5. Ввод из пар регистров в стек						
11R_i 101	1	3	11	$[SP - 1] \leftarrow (R_i)$; $[SP - 2] \leftarrow (R_{i+1})$; $SP \leftarrow (SP) - 2$	PUCH R_i	—	—	—	—	—
				6. Ввод (A) и (F) в стек						
11110101	1	3	11	$[SP - 1] \leftarrow (A)$; $[SP - 2] \leftarrow (F)$; $SP \leftarrow (SP) - 2$	PUCH PSW	—	—	—	—	—
				7. Вывод из стека в пары регистров						
11R_i001	1	3	10	$R_{i+1} \leftarrow [SP]$; $R_i \leftarrow [SP + 1]$; $SP \leftarrow (SP) + 2$	POP R_i	—	—	—	—	—
				8. Выбор (A) и (F) из стека						
11110001	1	3	10	$F \leftarrow [SP]$; $A \leftarrow [SP + 1]$; $SP \leftarrow (SP) + 2$	POP PSW	+	+	+	+	+
				9. Обмен между DE и HL						
11101011	1	1	4	$(H) \leftarrow \rightarrow (D)$; $(L) \leftarrow \rightarrow (E)$	XCH D	—	—	—	—	—
				10. Обмен вершины стека с HL						
11100011	1	5	13	$(L) \leftarrow \rightarrow [SP]$; $(H) \leftarrow [SP + 1]$	XTH L	—	—	—	—	—
				11. Пересылка (HL) в указатель стека						
11111001	1	1	5	$SP \leftarrow (HL)$	SPHL	—	—	—	—	—
				12. Пересылка (HL) в PC						
11101001	1	1	5	$PC \leftarrow (HL)$	PCHL	—	—	—	—	—

Продолжение табл. 3.2

Структура кода команды	Байты	Циклы	Такты	Выполняемая операция	Мнемоника	Признаки Z	S	C	V	P
				II. Положительное/отрицательное приращение						
00 R_i 100	1	1/3	5/10	1. Положительное приращение регистра $R_i \leftarrow (R_i) + 1$	INP	+	+	—	+	+
00 R_i 101	1	1/3	5/10	2. Отрицательное приращение регистра $R_i \leftarrow (R_i) - 1$	DCP	+	+	—	+	+
00 R_i 011	1	1	5	3. Положительное приращение пары регистров $R_iR_{i+1} \leftarrow (R_iR_{i+1}) + 1$; при $R_i = 110$: приращение SP	INX	—	—	—	—	—
00 R_i 011	1	1	5	4. Отрицательное приращение пары регистров $R_{i-1}R_i \leftarrow (R_{i-1}R_i) - 1$; при $R_i = 111$: отрицательное приращение SP	DCX	—	—	—	—	—
				III. Арифметические и логические операции						
10 K_2R_i	1	1/2	4/7	1. Над (A) и (R_i) $A \leftarrow (A)^* (R_i)$ *—операция, определяемая K_2						
				K_2 | Операция						
				0 0 0 | $A \leftarrow (A) + (R_i)$	ADD	+	+	+	+	+
				0 0 1 | $A \leftarrow (A) + (R_i) + (Tc)$	ADC	+	+	+	+	+
				0 1 0 | $A \leftarrow (A) - (R_i)$	SUB	+	+	+	+	+
				0 1 1 | $A \leftarrow (A) - (R_i) - (Tc)$	SBB	+	+	+	+	+
				1 0 0 | $A \leftarrow (A) \wedge (R_i)$	ANA	+	+	0	0	+
				1 0 1 | $A \leftarrow (A) \forall (R_i)$	XRA	+	+	0	0	+
				1 1 0 | $A \leftarrow (A) \vee (R_i)$	ORA	+	+	0	0	+
				1 1 1 | $(A) - (R_i)$ Сравнение	CMR	+	+	+	+	+

00 R_i 001	1	3	10	2. Сложение содержимого пар регистров $HL \leftarrow (HL) + (R_{i-1}R_i)$; при $R_i = 111$: $HL \leftarrow (HL) + (SP)$	DAD	—	—	+	—	—
11 K_2 110	2	2	7	3. Операции с непосредственной адресацией $A \leftarrow (A) * \langle B_2 \rangle$						

K_2	Операция									
0 0 0	$A \leftarrow (A) + \langle B_2 \rangle$	ADI	+	+	+	+	+
0 0 1	$A \leftarrow (A) + \langle B_2 \rangle + (Tc)$	ACI	+	+	+	+	+
0 1 0	$A \leftarrow (A) - \langle B_2 \rangle$	SUI	+	+	+	+	+
0 1 1	$A \leftarrow (A) - \langle B_2 \rangle - (Tc)$	SBI	+	+	+	+	+
1 0 0	$A \leftarrow (A) \wedge \langle B_2 \rangle$	ANI	+	+	0	0	+
1 0 1	$A \leftarrow (A) \forall \langle B_2 \rangle$	XRI	+	+	0	0	+
1 1 0	$A \leftarrow (A) \vee \langle B_2 \rangle$	ORI	+	+	0	0	+
1 1 1	$(A) - \langle B_2 \rangle$, сравнение	CPI	+	+	+	+	+

IV. Операция циклического сдвига

				K_3	Операция						
00 K_3 111	1	1	4	0 0 0	$A_{m+1} \leftarrow (A_m)$; $A_0 \leftarrow (A_7)$; $Tc \leftarrow (A_7)$	RLC	—	—	+	—	—
				0 0 1	$A_m \leftarrow (A_{m+1})$; $A_7 \leftarrow (A_0)$; $Tc \leftarrow (A_0)$	RRC	—	—	+	—	—
				0 1 0	$A_{m+1} \leftarrow (A_m)$; $A_0 \leftarrow (Tc)$; $Tc \leftarrow (A_7)$	RAL	—	—	+	—	—
				0 1 1	$A_m \leftarrow (A_{m+1})$ $A_7 \leftarrow (Tc)$; $Tc \leftarrow (A_0)$	RAR	—	—	+	—	—

V. Операции переходов

11000011	3	3	10	1. Безусловный переход $PC \leftarrow \langle B_3 \rangle \langle B_2 \rangle$	JMP	—	—	—	—	—

Продолжение табл. 3.2

Структура кода команды	Байты	Циклы	Такты	Выполняемая операция	Мнемоника	Признаки Z	S	C	V	P
				2. Условные переходы						
11 K_4 010	3	3	10	K_4 — Вид перехода						
				0 0 0 Если (Tz) = 0, то PC ← ⟨B_3⟩ ⟨B_2⟩; иначе PC ← (PC) + 3	JNZ	—	—	—	—	—
				0 0 1 Если (Tz) = 1, то PC ← ⟨B_3⟩ ⟨B_2⟩; иначе PC ← (PC) + 3	JZ	—	—	—	—	—
				0 1 0 Если (Tc) = 0, то PC ← ⟨B_3⟩ ⟨B_2⟩; иначе PC ← ⟨PC⟩ + 3	JNC	—	—	—	—	—
				0 1 1 Если (Tc) = 1, то PC ← ⟨B_3⟩ ⟨B_2⟩; иначе PC ← (PC) + 3	JC	—	—	—	—	—
				1 0 0 Если (Tp) = 0, то PC ← ⟨B_3⟩ ⟨B_2⟩; иначе PC ← (PC) + 3	JPO	—	—	—	—	—
				1 0 1 Если (Tp) = 1, то PC ← ⟨B_3⟩ ⟨B_2⟩; иначе PC ← (PC) + 3	JPE	—	—	—	—	—
				1 1 0 Если (Ts) = 0, то PC ← ⟨B_3⟩ ⟨B_2⟩; иначе PC ← (PC) + 3	JP	—	—	—	—	—
				1 1 1 Если (Ts) = 1, то PC ← ⟨B_3⟩ ⟨B_2⟩; иначе PC ← (PC) + 3	JM	—	—	—	—	—
				3. Вызов подпрограммы безусловный						
11001101	3	5	17	[SP − 1] [SP − 2] ← (PC); SP ← (SP) − 2; PC ← ⟨B_3⟩ ⟨B_2⟩	CALL	—	—	—	—	—
				4. Вызов подпрограммы условный						
11 K_5 100	3	3/5	11/17	K_5 — Вид перехода		—	—	—	—	—

				0 0 0	Если (Tz) = 0, то [SP − 1] [SP − 2] ← (PC); SP ← (SP) − 2; PC ← ⟨B₃⟩ ⟨B₂⟩; иначе PC ← (PC) + 3	CNZ	—	—	—	—	—
				0 0 1	Если (Tz) = 1, то [SP − 1] [SP − 2] ← (PC); SP ← (SP) − 2; PC ← ⟨B₃⟩ ⟨B₂⟩; иначе PC ← (PC) + 3	CZ	—	—	—	—	—
				0 1 0	Если (Tc) = 0, то [SP − 1] [SP − 2] ← (PC); SP ← (SP) − 2; PC ← ⟨B₃⟩ ⟨B₂⟩; иначе PC ← (PC) + 3	CNC	—	—	—	—	—
				0 1 1	Если (Tc) = 1, то [SP − 1] [SP − 2] ← (PC); SP ← (SP) − 2; PC ← ⟨B₃⟩ ⟨B₂⟩; иначе PC ← ← (PC) + 3	CC	—	—	—	—	—
				1 0 0	Если (Tp) = 0, то [SP − 1] [SP − 2] ← (PC); SP ← (SP) − 2; PC ← ⟨B₃⟩ ⟨B₂⟩; иначе PC ← ← (PC) + 3	CPO	—	—	—	—	—
				1 0 1	Если (Tp) = 1, то [SP − 1] [SP − 2] ← (PC); SP ← (SP) − 2; PC ← ⟨B₃⟩ ⟨B₂⟩; иначе PC ← ← (PC) + 3	CPE	—	—	—	—	—
				1 1 0	Если (Ts) = 0, то [SP − 1] [SP − 2] ← (PC); SP ← (SP) − 2; PC ← ⟨B₃⟩ ⟨B₂⟩; иначе PC ← ← (PC) + 3	CP	—	—	—	—	—
				1 1 1	Если (Ts) = 1, то [SP − 1] [SP − 2] ← (PC); SP ← (SP) − 2; PC ← ⟨B₃⟩ ⟨B₂⟩; иначе PC ← ← (PC) + 3	CM	—	—	—	—	—
11001001	1	3	10	5. Возврат из подпрограммы PC ← [SP] [SP + 1]; SP ← (SP) + 2		RET	—	—	—	—	—
				6. Условный возврат из подпрограммы							
11K₆000	1	1/3	5/11	K_6	Вид перехода						
				0 0 0	Если (Tz) = 0, то PC ← [SP] [SP + 1]; SP ← (SP) + 2, иначе PC ← (PC) + 1	RNZ	—	—	—	—	—
				0 0 1	Если (Tz) = 1, то PC ← [SP] [SP + 1]; SP ← (SP) + 2, иначе PC ← (PC) + 1	RZ	—	—	—	—	—
				0 1 0	Если (Tc) = 0, то PC ← [SP] [SP + 1]; SP ← (SP) + 2, иначе PC ← (PC) + 1	RNC	—	—	—	—	—

Продолжение табл. 3.2

Структура кода команды	Байты	Циклы	Такты	Выполняемая операция	Мнемоника	Признаки Z	S	C	V	P
11K$_6$000	1	1-3	5-11	K_6 — Вид перехода						
				0 1 1 — Если (Tc) = 1, то PC ← [SP] [SP + 1]; SP ← (SP) + 2, иначе PC ← (PC) + 1	RC	—	—	—	—	—
				1 0 0 — Если (Tp) = 0, то PC ← [SP] [SP + 1]; SP ← (SP) + 2, иначе PC ← (PC) + 1	RPO	—	—	—	—	—
				1 0 1 — Если (Tp) = 1, то PC ← [SP] [SP + 1]; SP ← (SP) + 2, иначе PC ← (PC) + 1	RPE	—	—	—	—	—
				1 1 0 — Если (Ts) = 0, то PC ← [SP] [SP + 1]; SP ← (SP) + 2, иначе PC ← (PC) + 1	RP	—	—	—	—	—
				1 1 1 — Если (Ts) = 1, то PC ← [SP] [SP + 1]; SP ← (SP) + 2, иначе PC ← (PC) + 1	RM	—	—	—	—	—
				VI. Операции ввода и вывода						
				1. Ввод данных						
11011011	2	3	10	A ← (входные данные)	IN	—	—	—	—	—
				2. Вывод данных						
11010011	2	3	10	Шина данных ← (A)	OUT	—	—	—	—	—

				VII. Прочие операции							
				1. Рестарт							
11AAA111	2	3	11	[SP—1] [SP—2] ← (PC); SP ← (SP) — 2; PC ← 00 000 000 00 AAA 000	RST	—	—	—	—	—
				2. Дополнение (A)							
00101111	1	1	4	A ← $\overline{(A)}$	SMA	—	—	—	—	—
				3. Установка переноса							
00110111	1	1	4	Tc ← 1	STC	—	—	1	—	—
				4. Дополнение переноса							
00111111	1	1	4	Tc ← $\overline{(Tc)}$	CMC	—	—	+	—	—
11111011	1	1	4	5. Разрешение прерываний	EI	—	—	—	—	—
11110011	1	1	4	6. Блокировка прерываний	DI	—	—	—	—	—
00100111	1	1	4	7. Двоично-десятичное представление (A)	DAA	+	+	+	+	+
00000000	1	1	4	8. Отсутствие операции	NOP	—	—	—	—	—
01110110	1	1	7	9. Останов	HLT	—	—	—	—	—

По команде LDAX B аккумулятор загружается содержимым ячейки оперативной памяти, адресом которой служит содержимое пары регистров BC (для указания именно этой пары регистров в разрядах $D_5D_4D_3$ команды приведен адрес 001 регистра C).

По комаиде STAX B содержимое аккумулятора запоминается в ячейке, адресом которой служит содержимое пары регистров BC (для указания пары регистров в разрядах $D_5D_4D_3$ команды приведен адрес 000 регистра B).

ПРИНЦИП РАБОТЫ МИКРОПРОЦЕССОРА

На рис. 3.3 показана структурная схема микропроцессорного устройства на МПК серии КР580. Генератор тактовых импульсов (ГТИ) формирует две импульсные последовательности Ф1 и Ф2, необходимые для тактирования работы микропроцессора (рис. 3.4). Импульсы двух последовательностей не должны перекрываться во времени, должны иметь амплитуду 12 В. ПЗУ может быть использовано для хранения программы, ОЗУ — для хранения данных.

Общий принцип функционирования микропроцессорного устройства заключается в следующем. Из микропроцессора на шину адреса выдается адрес очередной команды. Считанная по этому адресу из памяти (например, из ПЗУ) команда поступает на шину данных и принимается в микропроцессор, где она исполняется. В счетчике команд микропроцессора формируется адрес следующей команды. После окончания исполнения данной команды на шину адреса поступает адрес

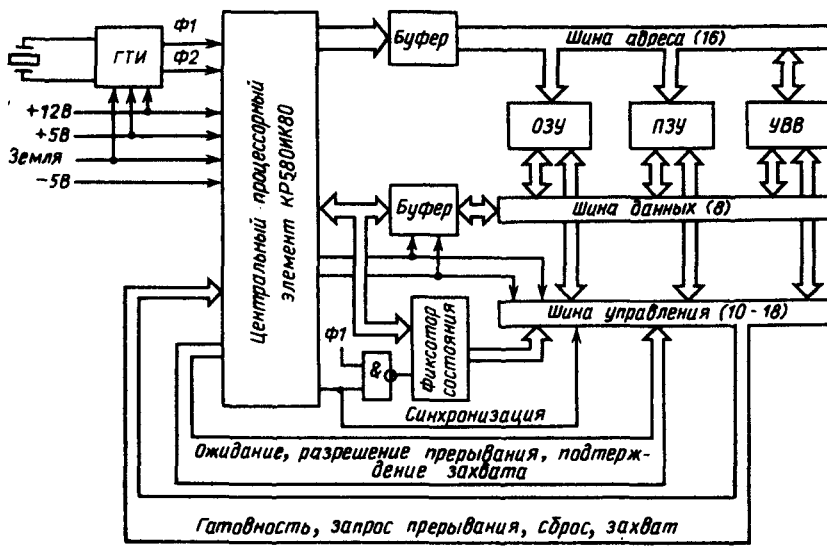


Рис. 3.3. Структурная схема микропроцессорного устройства

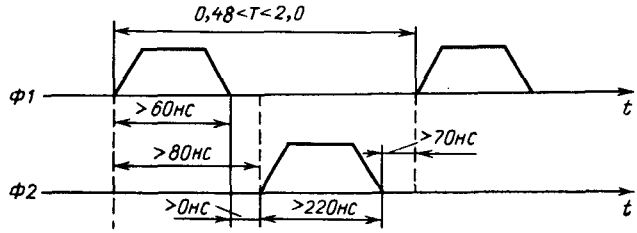


Рис. 3.4. Тактовые последовательности импульсов

следующей команды и т. д. В процессе исполнения команды могут потребоваться дополнительные обращения к памяти для вызова в микропроцессор дополнительных байтов команды (в случае двух-, трехбайтовых команд), операндов или записи в память числа, выдаваемого из микропроцессора.

Рассмотрим подробнее процесс выполнения команды. Этот процесс разбивается на циклы, обозначаемые M_1, M_2, M_3, M_4, M_5. В каждом цикле производится одно обращение микропроцессора к памяти или УВВ (исключение составляет лишь выполнение команды DAD).

В зависимости от типа команда может быть выполнена за один цикл (M_1), либо за два цикла (M_1, M_2), либо за три цикла (M_1, M_2, M_3) и т. д. Самые длинные по времени исполнения команды выполняются в пять циклов (M_1 ... M_5).

Каждый цикл включает в себя несколько тактов, обозначаемых T_1, T_2, T_3, T_4, T_5. Циклы могут содержать три (T_1 ... T_3), четыре (T_1... ...T_4) либо пять (T_1 ... T_5) тактов. Первые три такта во всех циклах используются для организации обмена с памятью и УВВ, такты T_4 и T_5 (если они присутствуют в цикле) — для выполнения внутренних операций в микропроцессоре. На рис. 3.5 показана временна́я диаграмма цикла из пяти тактов.

Отсчет тактов производится от положительных фронтов импульсов Ф1. Рассмотрим цикл M_1. В такте T_1 содержимое счетчика команд выдается на шину адреса, адрес принимается памятью, где начинается

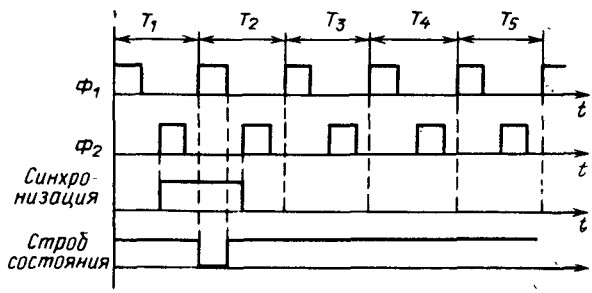


Рис. 3.5. Сигиалы синхронизации и строба состояния

процесс чтения байта команды из указанной ячейки. В такте T_2 проверяется наличие сигнала (уровня лог. 1) на входе *Готовность*. Этот сигнал подается на вход микропроцессора через интервал времени, достаточный для завершения процесса чтения из памяти. Если на входе *Готовность* сигнал отсутствует (действует уровень лог. 0), то микропроцессор устанавливается в режим ожидания, в котором каждый следующий такт рассматривается как такт T_2 до тех пор, пока не появится сигнал на входе *Готовность*. С приходом этого сигнала микропроцессор выходит из режима ожидания, переходя в такт T_3. В этом такте выданный из памяти байт команды с шины данных принимается в микропроцессор, где он помещается в регистр команд. В такте T_4 анализируется принятый байт команды и выясняется, нужны ли дополнительные обращения в оперативную память. Если такие обращения не требуются (команда однобайтовая и операнды находятся в регистрах микропроцессора), то в этом же такте либо с использованием дополнительно такта T_5 выполняется предусматриваемая командой операция.

Если необходимы дополнительные обращения в оперативную память, то после такта T_4 цикл M_1 завершается и происходит переход к к циклу M_2. Пусть, например, команда однобайтовая, но в операции должен участвовать операнд, хранящийся в оперативной памяти. Тогда в цикле M_2 происходят следующие процессы: в такте T_1 выдается адрес ячейки памяти, в такте T_2 проверяется наличие сигнала на входе *Готовность* (сигнала о том, что прошел интервал времени, достаточный для чтения из памяти). С появлением этого сигнала происходит переход к такту T_3, в котором выданное из памяти число с шины данных принимается в микропроцессор и в этом же такте выполняется операция, предусматриваемая командой.

При исполнении большинства команд в случаях, когда происходят дополнительные обращения к памяти, первый цикл M_1 содержит четыре такта, в каждом следующем цикле содержится три такта и происходит одно дополнительное обращение к памяти.

ИНФОРМАЦИЯ О СОСТОЯНИИ МИКРОПРОЦЕССОРА

В каждом цикле в интервале времени от момента положительного фронта импульса последовательности Ф2 в такте T_1 и до момента положительного фронта импульса Ф2 в такте T_2 микропроцессор выдает на выход *Синхронизация* уровень лог. 1 и на шину данных—информацию о состоянии. Элемент И-НЕ (см. рис. 3.3) формирует строб, которым осуществляется прием информации о состоянии микропроцессора с шины данных в регистр состояния (временно́е положение строба состояния показано на рис. 3.5). В табл. 3.3 показано назначение отдельных разрядов кода состояния.

В табл. 3.4 приведено соответствие сигналов состояния отдельным видам циклов.

Таблица 3.3

Разряд	Назначение сигнала
D_0	Подтверждение прерываиия: используется для стробированнн команды RST в микропроцессор из устройства, запрашивающего прерывание
D_1	Запись-вывод: уровень лог. 0 свидетельствует о том, что в данном цикле будет происходить запнсь (выдача информации из микропроцессора в оперативную память) или вывод (передача иифорлации из микропроцессора в УВВ); уровень лог. 1 озиачает, что происходит чтение (прием информации из оперативиой памяти) или ввод (прием из УВВ)
D_2	Свидетельствует о том, что в даином цикле на адресной шине устаиовлеио содержимое указателя стека
D_3	Подтверждение останова: свидетельствует о том, что микропроцессор в состояиии остаиова
D_4	Свидетельствует о том, что в даиион цикле на адресиой шиие устаиовлеи номер виешнего устройства и осуществляется вывод содержимого аккумулятора иа устройство вывода
D_5	Свидетельствует о том, что в даиион цикле микропроцессор прииимает первый байт комаиды
D_6	Свидетельствует о том, что в даиион пикле на адресной шиие устаиовлен иомер устройства ввода и осуществляется ввод иифорлации из устройства ввода в аккумулятор микропроцессора
D_7	Свидетельствует о том, что в данном цикле производится чтение (прием информации из памяти в микропроцессор)

Таблица 3.4

Вид цикла	Состояние микропроцессора							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Выборка первого байта команды	1	0	1	0	0	0	1	0
Чтение из памяти	1	0	0	0	0	0	1	0
Запись в память	0	0	0	0	0	0	0	0
Чтение стека	1	0	0	0	0	1	1	0
Запись в стек	0	0	0	0	0	1	0	0
Ввод из УВВ	0	1	0	0	0	0	1	0
Вывод иа УВВ	0	0	0	1	0	0	0	0
Подтверждение прерываиия	0	0	1	0	0	0	1	1
Подтверждение остаиова	1	0	0	1	1	0	1	0
Подтверждение прерываиия прн остаиове	0	0	1	1	1	0	1	1

СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА

Система команд микропроцессора приведена в табл. 3.2. Команды разбиты на семь групп. Группа может содержать несколько типов операций. Каждый тип операций характеризуется некоторой структурой кодовых комбинаций команд, где вместо R_i должен быть подставлен адрес регистра и вместо K_i — 3-разрядная кодовая комбинация, определяющая конкретный тип команды.

В таблице указано число байтов, содержащихся в команде, число циклов и тактов, в которые выполняется команда (в знаменателе указано число циклов и тактов в случаях, когда указан адрес регистра 110 и требуется дополнительное обращение в оперативную память для выборки операнда, адресом которого служит содержимое пары регистров HL).

Для каждого типа команды показано, как формируются призиаки в пяти триггерах регистра признаков. Принята следующая система обозначений:

«+» означает, что признак в данном триггере формируется;

«—» означает, что соответствующий признак при выполнении данной команды не формируется и в триггере сохраняется значение признака, сформированное при выполнении предыдущих команд;

0 означает установку триггера в состояние лог. 0;

1 означает установку триггера в состояние лог. 1.

Отметим следующие особенности формирования признаков:

команды пересылки и переходов не изменяют состояния триггеров призиаков;

команды увеличения или уменьшения содержимого одиночного регистра используют все признаки, за исключением признака переноса С;

команды увеличения или уменьшения содержимого пар регистров не изменяют состояния триггеров признаков;

команды арифметических операций используют все признаки; при выполнении логических операций триггеры переносов Tc и Tv сбрасываются в состояние лог. 0; команды сложения содержимого пар регистров используют только признак переноса С.

ОПЕРАЦИИ ЦИКЛИЧЕСКОГО СДВИГА

Операции циклического сдвига выполняются над содержимым аккумулятора и предусматривают четыре вида сдвига.

1. Сдвиг циклический влево (СЦЛ) без переноса (рис. 3.6, *а*): содержимое каждого разряда аккумулятора А передается в соседний старший разряд ($A_{m+1} \leftarrow (A_m)$), содержимое старшего разряда передается в младший разряд ($A_0 \leftarrow (A_7)$) и одновременно в триггер переноса ($Tc \leftarrow (A_7)$).

2. Сдвиг циклический вправо (СЦП) без переноса (рис. 3.6, *б*): содержимое каждого разряда аккумулятора А передается в соседний

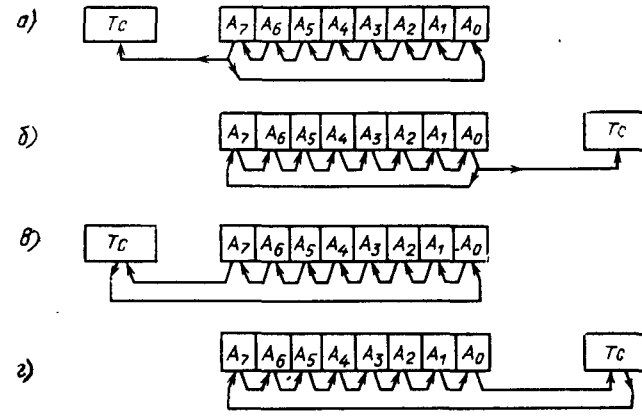


Рис. 3.6. Виды сдвигов:
а) влево без переноса, *б)* вправо без переноса, *в)* влево с переносом, *г)* вправо с переносом

младший разряд ($A_m \leftarrow (A_{m+1})$), содержимое младшего разряда передается в старший разряд ($A_7 \leftarrow (A_0)$) и одновременно в триггер переноса (Тс $\leftarrow (A_0)$).

3. Сдвиг циклический влево с переносом (рис. 3.6, *в*): отличие от сдвига без переноса состоит в том, что триггер переноса Тс вводится в замкнутый контур, в котором осуществляется сдвиг; триггер переноса передает свое содержимое в младший разряд аккумулятора ($A_0 \leftarrow$ $\leftarrow$ (Тс)) и принимает выдвигаемое из аккумулятора содержимое старшего разряда (Тс $\leftarrow (A_7)$).

4. Сдвиг циклический вправо с переносом (рис. 3.6, *г*): отличие от сдвига без переноса в том, что триггер переноса передает свое содержимое в старший разряд аккумулятора ($A_7 \leftarrow$ (Тс)) и принимает выданное из аккумулятора содержимое младшего разряда (Тс $\leftarrow (A_0)$).

СТЕК

Стек — память с определенной (упрощенной) формой адресации. В микропроцессорном устройстве на МПК КР580 стек организуется следующим образом. В оперативной памяти (ОЗУ) команды размещаются в ячейках с младшими, последовательно нарастающими адресами. Стек использует ячейки со старшими адресами и по мере заполнения стека занимаются ячейки с адресами, последовательно убывающими. Таким образом, адреса этих двух частей памяти изменяются навстречу друг другу (рис. 3.7, *а*).

Особенность организации стека состоит в следующем. Указатель стека SP содержит так называемый адрес входа в стек; при чтении из стека производится выборка содержимого ячейки по адресу входа в стек (по адресу, хранящемуся в SP); при записи в стек вводимое в стек число помещается в ячейку с адресом, на единицу меньшим содержимо-

го SP; одновременно с записью и чтением изменяется содержимое SP: при записи уменьшается, а при чтении увеличивается на единицу.

Обмен со стеком производится двухбайтовыми словами, занимающими две ячейки памяти. Пусть указатель стека хранит адрес А. При вводе нового слова его байты должны быть помещены в пару соседних со входом в стек ячеек, имеющих адреса А-1 и А-2. Таким образом, ввод в стек сводится к следующей последовательности действий: содержимое SP уменьшается на единицу и по образующемуся в SP адресу помещается старший байт вводимого двухбайтового слова; за-

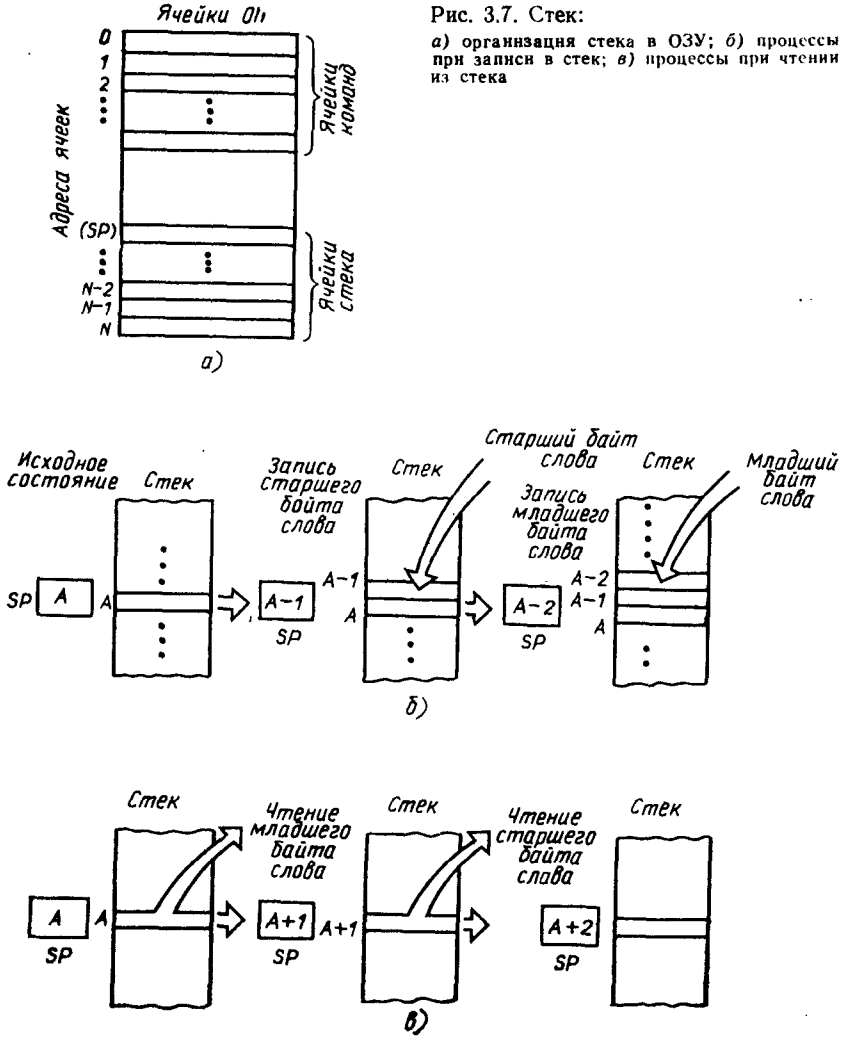


Рис. 3.7. Стек:

а) организация стека в ОЗУ; *б)* процессы при записи в стек; *в)* процессы при чтении из стека

тем содержимое SP вновь уменьшается на единицу и по образующемуся в нем адресу помещается младший байт вводимого слова (рис. 3.7,*б*).

Мы видим, что SP каждый раз указывает адрес последней ячейки, занятой под стек,— так называемый вход в стек.

Вывод данных из стека производится также двухбайтовыми словами. При этом каждый раз доступна для чтения лишь ячейка, адрес которой содержится в SP. Если указатель стека хранит адрес A, то байты выводимого из стека слова выбираются из ячеек памяти, имеющих адреса A и A+1. Таким образом, выбор слова из стека сводится к такой последовательности действий: чтение младшего байта выводимого слова из ячейки, адресом которой служит содержимое SP, и увеличение содержимого SP на единицу; затем чтение старшего байта выводимого слова по хранящемуся в SP адресу и увеличение содержимого SP на единицу (рис. 3.7, *в*).

О таком принципе функционирования, когда читается последняя помещенная в память информация, говорят как о принципе «последним вошел — первым вышел». Как видим, при записи и чтении производится обращение в ячейку, адрес которой связан с содержимым SP. Это упрощает адресацию памяти, но исключает возможность обращения в произвольную ячейку памяти.

Рассмотрим команды некоторых операций со стеком.

Установка в SP некоторого начального значения производится по команде пересылки SPHL (11 111 001). По этой команде в SP пересылается содержимое пары регистров HL.

Команда ввода из пары регистров DE в стек: PUSH D (11 010 101). В разрядах $D_5D_4D_3$ кодовой комбинации команды указан адрес 010 старшего регистра пары DE. По этой команде выполняются следующие действия: SD ← (SP) — 1; [(SP)] ← (D); SP ← (SP) — 1; [(SP)] ← (E).

Команда пересылки из стека в пару регистров DE : POP D (11 010 001). Здесь в разрядах $D_5D_4D_3$ кодовой комбинации команды указан адрес 010 старшего регистра пары DE. По данной команде выполняются действия: E ← [(SP)]; SP ←(SP) + 1; D ← [(SP)]; SP ← (SP) + 1.

ЗАПУСК МИКРОПРОЦЕССОРА

После подачи на соответствующие входы микропроцессора питающих напряжений и тактовых импульсов последовательностей Ф1 и Ф2 подается сигнал уровня лог. 1 на вход *Сброс*. Этим сигналом сбрасываются в состояние лог. 0 счетчик команд PC, регистр команд, размещенные в управляющем устройстве триггеры разрешения прерывания, подтверждения захвата и ожидания. После окончания действия сигнала *Сброс* (при переходе сигнала от уровня лог. 1 к уровню лог. 0) микропроцессор начинает работать с такта T_1 цикла M_1 и выдает на шину адреса нулевое значение адреса. Содержимое регистров блока РОН, аккумулятора, регистра признаков меняется только в процессе выполнения команд.

СОСТОЯНИЕ ЗАХВАТА

Состояние захвата характеризуется тем, что микропроцессор, заканчивая выполнение текущего цикла команды, переводит буферы шины данных и буферы шины адреса в третье состояние. При этом микропроцессор отключается от внешних шин, предоставляя их в распоряжение некоторого внешнего устройства, и останавливает работу.

Переход в состояние захвата происходит следующим образом. От внешнего устройства поступает сигнал уровня лог. 1 на вход *Запрос захвата*. Этот сигнал на отрицательном фронте импульса Ф2 такта T_2 принимается в триггер захвата управляющего устройства. Управляющее устройство заканчивает выполнение текущего цикла, переходит в состояние захвата и подтверждает это выдачей сигнала на выходе *Подтверждение захвата*. Сигнал на выходе *Подтверждение захвата* выдается на положительном фронте импульса Ф1 в такте T_3, если текущий цикл не является циклом записи; в противном случае этот сигнал выдается на положительном фронте импульса Ф1 такта, следующего за тактом T_3.

После окончания действия сигнала *Захват* (при переходе от уровня лог. 1 к уровню лог. 0) микропроцессор начинает выполнение следующего цикла с места, где было приостановлено исполнение программы.

СОСТОЯНИЕ ПРЕРЫВАНИЯ

В микропроцессоре предусмотрена возможность по запросам внешних устройств прерывать выполнение текущей программы и переходить на выполнение новой программы, так называемой *прерывающей программы* (или *программы обслуживания прерывания*). После окончания выполнения прерывающей программы микропроцессор возвращается к выполнению осиовной программы с команды, на которой произошло прерывание.

Если на некотором участке программы допускается ее прерывание, то при составлении программы в начале этого участка предусматривается команда EI, по которой триггер разрешения прерывания в управляющем устройстве микропроцессора устанавливается в состояние лог. 1, а в конце участка — команда DI, при выполнении которой триггер сбрасывается в состояние лог. 0. Состояние триггера выдается на выход *Разрешение прерывания*.

Процесс прерывания связан со следующими действиями. От внешнего устройства поступает сигнал уровня лог. 1 на вход *Запрос прерывания*. Если прерывание разрешено (т. е. на выходе *Разрешение прерывания* имеется уровень лог. 1), то после окончания выполнения текущей команды триггер разрешения прерывания сбрасывается в состояние лог. 0, а в информации о состоянии микропроцессора, выдаваемой на шину данных, появляются сигналы *Подтверждение прерывания* (в разряде D_0), *Ввод* (в разряде D_1) и сигнал о том, что в данном

цикле производится прием первого байта команды (в разряде D_5). Сигнал *Подтверждение прерывания* используется в качестве строба для выдачи внешним устройством на шину данных команды RST (команды *Рестарт*).

При выполнении команды RST содержимое счетчика команд PC запоминается в стеке, а в счетчик команд PC записывается адрес первой команды прерывающей программы. Этот адрес задается следующим образом. Команда RST имеет структуру 11 AAA 111 и в счетчик комаид заносится значение 00 000 000 00 AAA 000, которое и служит адресом первой команды прерывающей программы. Задавая определенную трехразрядную кодовую комбинацию AAA, внешнее устройство может задать адрес первой команды одной из восьми прерывающих программ.

После окончания выполнения прерывающей программы возврат в основную программу происходит следующим образом. Прерывающая программа заканчивается командой RET (*Возврат из подпрограммы*). В процессе выполнения этой комаиды адрес команды основной программы, перед которой произошло прерывание, выбирается из стека и передается в регистр адреса, а увеличенное на единицу значение заносится в счетчик команд.

СОСТОЯНИЕ ОСТАНОВА

В системе команд микропроцессора имеется команда HLT (*Останов*), которая вызывает прекращение выполнения программы и переход в состояние останова. Это состояние характеризуется тем, что буферы шины адреса и шины данных переходят в третье состояние, микропроцессор отключается от внешних шин и на выходе *Ожидание* устанавливается уровень лог. 1.

Состояние останова может быть прервано сигналами запуска микропроцессора либо перевода его в состояние прерывания.

3.3. ПРИЕМЫ ПРОГРАММИРОВАНИЯ МИКРОПРОЦЕССОРА НА ЯЗЫКЕ КОДОВЫХ КОМБИНАЦИЙ

ПРОГРАММИРОВАНИЕ ПОСЛЕДОВАТЕЛЬНЫХ УЧАСТКОВ АЛГОРИТМА

Будем рассматривать программирование участков алгоритма, ие содержащих разветвлений.

Пример 3.1. Требуется принять из ОЗУ два числа, хранящихся в соседних ячейках, и, вычислив разность чисел, поместить ее в ОЗУ на место второго числа. Будем считать, что адрес первого числа хранится в паре регистров HL, адрес второго числа на единицу больше содержимого этих регистров.

На рис. 3.8 приведена схема алгоритма решения данной задачи, построенная в операциях, выполняемых микропроцессором серии КР580. Рассмотрим операции, выполняемые в каждом из блоков схемы алгоритма.

Блок *1* производит прием в аккумулятор содержимого ячейки ОЗУ (М), адресом которой служит содержимое пары регистров HL; таким образом, в регистр А принимается первое из чисел; эта операция может быть выполнена командой пересылки регистр — регистр (мнемоническое обозначение команды MOV A, M).

Блок *2* формирует в паре регистров HL адрес второго числа; эта операция выполняется командой приращения пары регистров (мнемоника команды INX H).

Блок *3* производит вычисление разности содержимого аккумулятора (А) и содержимого ячейки ОЗУ (М), адресом которой служит содержимое пары регистров HL; операция выполняется командой вычитания SUB M.

Блок *4* пересылает в память полученную в аккумуляторе разность; выполняющая эту операцию команда имеет мнемонику MOV M, A.

В табл. 3.5 приведена программа рассматриваемой задачи с представлением команд в кодовых комбинациях.

Команды программы при отсутствии условных и безусловных переходов размещаются в ячейках памяти с последовательно нарастающими адресами. При построении данной программы размещение команд произведено, начиная с ячейки, имеющей адрес 0050_{16}.

Общее число тактов, необходимых для выполнения приведенных четырех команд.

$$N_т = 7+5+7+7 = 26$$

и общее время их исполнения

$$t_{исп} = N_т\, T = 26 \cdot 0{,}5 = 13 \text{ мкс}.$$

Рассмотрим пример программирования с использованием содержимого триггера регистров признаков.

Пример 3.2. Требуется выполнить операцию арифметического сдвига вправо над содержимым регистра *D*.

Таблица 3.5

Адрес команды в ОП (в 16-ричной системе)	Команда			Пояснения
	Кодовая комбинация	Число байтов	Число тактов	
0 0 5 0	0 1 1 1 1 1 0	1	7	Блок *1*:A ← (M)
0 0 5 1	0 0 1 0 0 0 1 1	1	5	Блок *2*: HL ← (HL) + 1
0 0 5 2	1 0 0 1 0 1 1 0	1	7	Блок *3*:A ← (A) — (M)
0 0 5 3	0 1 1 1 0 1 1 1	1	7	Блок *4*:M ← (A)
. . .	. . .	. . .	. . .	. . .

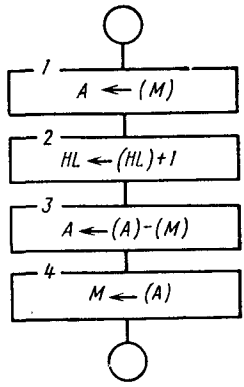


Рис. 3.8. Схема алгоритма вычитания чисел, хранящихся в памяти

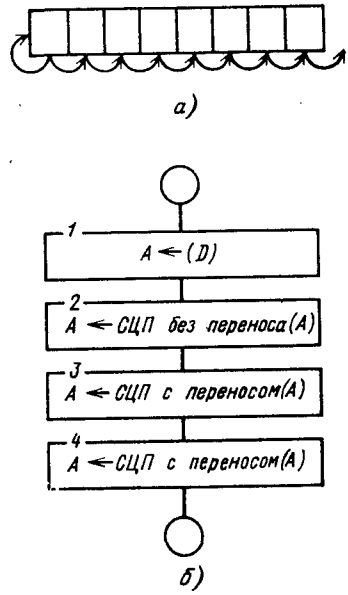


Рис. 3.9. Арифметический сдвиг вправо:
а) принцип; *б)* схема алгоритма

Особенность арифметического сдвига вправо состоит в том, что при сдвиге содержимое старшего (знакового) разряда регистра сохраняется неизменным (рис. 3.9, *а*). Таким образом, при выполнении этой операции требуется предварительно запомнить содержимое старшего разряда и затем, сдвинув вправо содержимое регистра, вписать в его старший разряд цифру, которую ранее запомнили. Эти действия выполняются в фрагменте программы, схема алгоритма которой представлена на рис. 3.9, *б*. Так как операции сдвига выполняются только над содержимым аккумулятора, блок *1* пересылает в аккумулятор содержимое регистра D. Операции, выполняемые в последующих блоках, иллюстрируются табл. 3.6.

Выполняется операция циклического сдвига влево без переноса (блок 2). В результате выполнения этой операции знак числа (в табл. 3.6 он выделен полужирным шрифтом) передается в младший разряд аккумулятора и в триггер переноса Тс.

Затем дважды выполняется операция циклического сдвига вправо с переносом (блоки *3* и *4*). В результате выполнения первой операции

Таблица 3.6

(Тс)	(А)	Выполняемая операция
X	**1 0 1 1 0 1 1 0**	Исходное состояние
1	0 1 1 0 1 1 0 1	Сдвиг влево без переноса
1	**1 0 1 1 0 1 1 0**	Сдвиг вправо с переносом
0	**1 1 0 1 1 0 1 1**	Сдвиг вправо с переносом

Таблица 3.7

Адрес комаиды в ОП (в 16-ричной системе)	Команда в кодовой комбинации	Поясиения
0 0 7 1	0 1 1 1 0 1 0	Блок *1*: A←(D)
0 0 7 2	0 0 0 0 0 1 1 1	Блок *2*: СЦЛ без переноса
0 0 7 3	0 0 0 1 1 1 1 1	Блок *3*: СЦП с переносом
0 0 7 4	0 0 0 1 1 1 1 1	Блок *4*: СЦП с переносом
...	...	...

сдвига в аккумуляторе восстанавливается исходное число, а в триггере Тс оказывается продублированным знак числа. После выполнения второй операции сдвига в аккумуляторе оказывается число, являющееся результатом выполнения арифметического сдвига вправо.

В табл. 3.7 приведена соответствующая схеме алгоритма на рис.3.9 программа с представлением команд в кодовых комбинациях.

ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЕНИЙ

Пусть требуется проанализировать содержимое младшего разряда числа, хранящегося в регистре В. Если оно равно нулю, то к содержимому регистра В следует прибавить содержимое регистра С; если оно равно единице, то к содержимому регистра В следует прибавить содержимое регистра D.

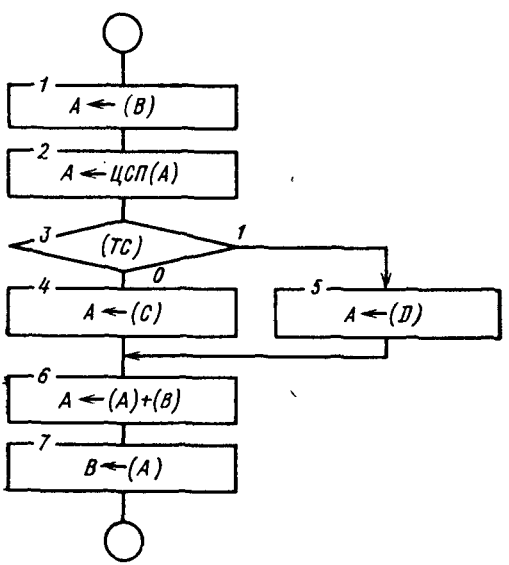


Рис. 3.10. Схема алгоритма разветвляющегося процесса

На рис. 3.10 показана схема алгоритма этой задачи. Здесь блоки *1* и *2* осуществляют передачу содержимого младшего разряда регистра В в триггер Тс регистра признаков. Блок *3* реализует разветвление по содержимому триггера Тс: в зависимости от значения содержимого этого триггера в аккумулятор передается либо содержимое регистра С (блок *4*) либо содержимое регистра D (блок *5*). Блок *6* осуществляет сложение. Блок *7* полученную в аккумуляторе сумму пересылает в регистр В.

Таблица 3.8

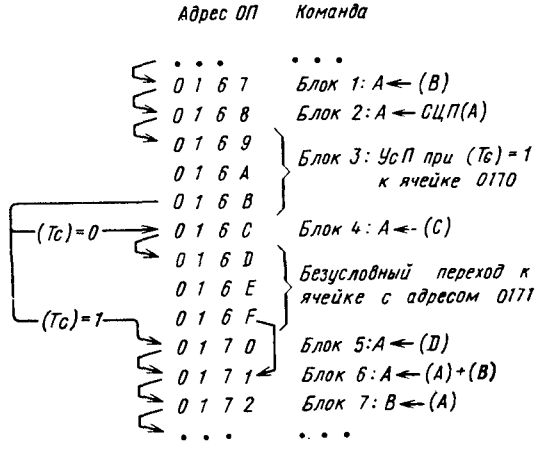


Таблица 3.9

Адрес комаиды в ОП	Комаида в кодовой комбинации	Число байтов	Пояснеиия
...	...	...	...
0 1 6 7	0 1 1 1 1 0 0 0	1	Блок *1*: A←(B)
0 1 6 8	0 0 0 0 1 1 1 1	1	Блок *2*: A←СЦП (A)
0 1 6 9 0 1 6 A 0 1 6 B	1 1 0 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1	3	Блок *3*: УсП при (Тс) = 1 к ячейке $(B_3B_2) = 0170_{16}$
0 1 6 C	0 1 1 1 1 0 0 1	1	Блок *4*: A←(C)
0 1 6 D 0 1 6 E 0 1 6 F	1 1 0 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1	3	Безусловиый переход к ячейке $(B_3B_2) = 0171$
0 1 7 0	0 1 1 1 1 0 1 0	1	Блок *5*: A←(D)
0 1 7 1	1 0 0 0 0 0 0 0	1	Блок *6*: A←(A)+(B)
0 1 7 2 ...	0 1 0 0 0 1 1 1 ...	1 ...	Блок *7*: B←(A) ...

В табл. 3.8 показано размещение в ОП команд, реализующих рассмотренный алгоритм.

Пусть команда, реализующая операцию блока *1*, помещается в ячейку ОП с адресом 0167. При выполнении программы следующая команда будет считываться из соседней ячейки 0168 и в этой ячейке должна храниться команда, реализующая операцию блока *2*. Затем из трех очередных ячеек 0169, 016A, 016B должна считываться трехбайтовая команда условного перехода по (Тс) = 1 (блок *3*): при (Тс) = 0 не происходит нарушения естественного порядка следования ячеек, из которых при исполнении программы считываются команды, и очередная команда (блок *4*) считывается из ячейки с адресом 016C; при (Тс) = 1 происходит переход к ячейке 0170 (этот адрес приводится во втором и третьем байтах команды условного перехода), хранящей команду блока *5*. Далее, независимо от того, выполняется ли команда блока *4* либо команда блока *5*, следующим действием должно быть выполнение операции, предусматриваемой блоком *6*. После выполнения команды, считываемой из ячейки 0170, очередная команда блока *6* считывается из соседней ячейки 0171. Но если выполняется команда блока *4*, то после ее выполнения переход к ячейке 0171 может быть выполнен трехбайтовой командой безусловного перехода, помещаемой в ячейки 016D, 016E, 016F (во втором и третьем байтах этой команды указывается адрес перехода 0171). Затем из очередной ячейки 0172 считывается команда, выполняющая операцию блока *7*.

В табл. 3.9 приведена программа данной задачи.

ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Рассмотрим выполнение операции кодового умножения двух восьмиразрядных чисел без знака. Пусть множимое хранится в паре регистров DE, где оно занимает младший регистр E, старший регистр D пары регистров установлен в нуль. Будем считать, что множитель хранится в аккумуляторе A. Шестнадцатиразрядное произведение будем формировать в паре регистров HL.

Процесс получения произведения организуем следующим образом. Будем анализировать разряды множителя, начиная с его старшего разряда. В соответствии с этим частичные произведения будут формироваться, начиная со старшего частичного произведения. Накопление суммы частичных произведений будем производить в паре регистров HL, т. е. к содержимому предварительно сброшенной в нуль пары регистров HL вначале прибавим восьмое частичное произведение; затем, сдвинув на один разряд влево содержимое пары регистров HL, прибавим седьмое частичное произведение и т. д., пока не будут просуммированы все частичные произведения.

Таким образом, этот процесс носит циклический характер: цикл, содержащий операции сдвига влево содержимого пары регистров HL,

формирования и прибавления к содержимому пары регистров HL очередного частичного произведения, должен быть повторен восемь раз.

Для счета числа повторений цикла организуем счетчик на регистре B. В этот регистр предварительно занесем число 8 и после каждого повторения цикла будем вычитать единицу из содержимого регистра B, проверяя затем, равно ли нулю его содержимое. При достижении нулевого значения производится выход из цикла.

На рис. 3. 11 представлена схема алгоритма. Блок *1* производит установку нулевого значения в паре регистров HL. Блок *2* устанавливает в регистре B (счетчике) начальное значение 8. Блок *3* производит сдвиг на один разряд влево содержимого пары регистров HL; эта операция выполняется путем удвоения содержимого этой пары регистров: HL ← (HL) + (HL). Блок *4* предназначен для анализа очередного разряда множителя; для этого содержимое аккумулятора A сдвигается влево, в результате чего очередной разряд хранимого в нем множителя передается в триггер Tc регистра признаков. Блок *5* производит разветвление по содержимому триггера Tc. При (Tc) = 1 в блоке *6* выполняется операция прибавления множимого (содержимого пары регистров DE) к сумме предыдущих частичных произведений в паре регистров HL. При (Tc) = 0 операция суммирования не выполняется, по команде условного перехода осуществляется переход к команде блока *7*. Блок *7* производит вычитание единицы из содержимого счетчика (регистра B), после чего блок *8* выполняет разветвление по содержимому триггера Tz регистра признаков. Если при выполнении команды блока *7* в регистре B образуется нулевое значение, в триггере Tz устанавливается значение лог. 1, происходит выход из цикла и переход к очередной команде. Если содержимое регистра B не равно нулю, то в триггере Tz устанавливается значение 0 и команда условного перехода производит переход к команде блока *3*, вызывая очередное повторение цикла.

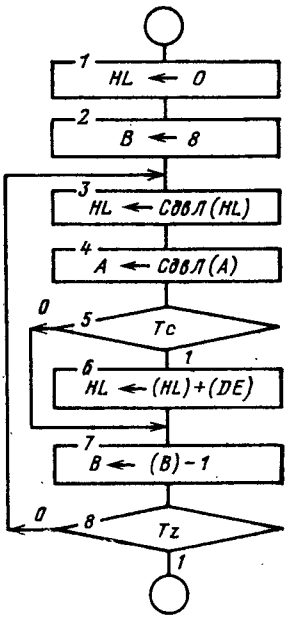


Рис. 3.11. Схема алгоритма операции кодового умножения

В табл. 3.10 показано размещение команд в ячейках ОП.

В табл. 3.11 приведена программа рассматриваемой операции умножения.

В цикле выполняются команды блоков *3* ... *8*. Определим количество тактов N_{T1}, требуемое для однократного прохождения цикла алгоритма. При этом будем полагать, что во всех разрядах множите-

Таблица 3.10

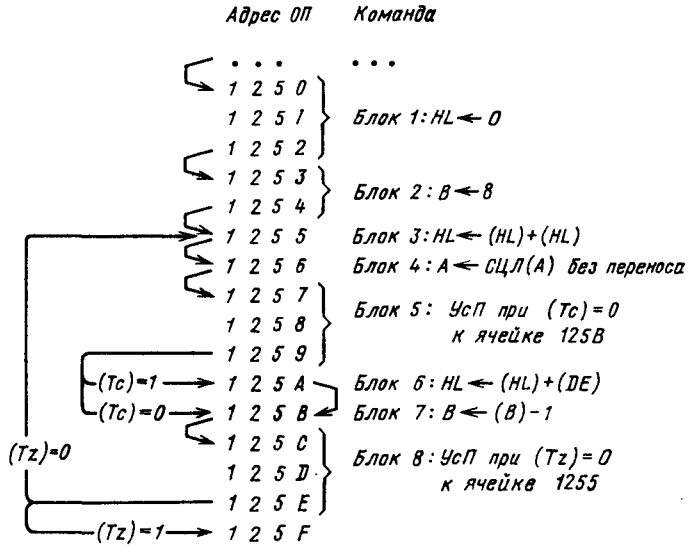


Таблица 3.11

Адрес команды в ОП	Команда в кодовой комбинации	Число байтов	Число тактов	Пояснения
. . .	. . .	. . .	. . .	. . .
1 2 5 0 1 2 5 1 1 2 5 2	0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	3	10	Блок *1*: HL←(B_3B_2); (B_3)=0; (B_2)=0
1 2 5 3 1 2 5 4	0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0	2	7	Блок *2*: B←(B_2) (B_2)=8
1 2 5 5	0 0 1 0 1 0 0 1	1	10	Блок *3*: HL←(HL)+(HL)
1 2 5 6	0 0 0 0 0 1 1 1	1	4	Блок *4*: A←СЦЛ (A) без переноса
1 2 5 7 1 2 5 8 1 2 5 9	1 1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 1 0 0 1 0	3	10	Блок *5*: УсП при (Tc)=0 к ячейке 125B (B_3)=12_{16}; (B_2)=$5B_{16}$
1 2 5 A	0 0 0 1 1 0 0 1	1	10	Блок *6*: HL←(HL)+(DE)
1 2 5 B	0 0 0 0 0 1 0 1	1	5	Блок *7*: B←(B) — 1
1 2 5 C 1 2 5 D 1 2 5 E	1 1 0 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 1 0 0 1 0	3	10	Блок *8*: УсП при (Tz)=0 к ячейке 1255; (B_3)=12_{16}; (B_2)=55_{16}
1 2 5 F	. . .	. . .	. . .	. . .

ля содержатся единицы (случай с точки зрения быстродействия наиболее тяжелый):

$N_{т1} = 10+4+10+10+5+10=49$ тактов.

Число тактов при восьмикратном прохождении цикла

$N_т = 8 \cdot N_{т1} = 8 \cdot 49 = 392$ такта.

В § 1.4 было показано, что в процессоре, в котором используется специализированное операционное устройство и управляющее устройство, построенное на принципе схемной логики, число тактов для однократного прохождения цикла алгоритма составляло 2. Следовательно, реализация рассматриваемой операции умножения в микропроцессоре требует в 49/2 = 24,5 раза большего числа тактов. Такой проигрыш в быстродействии при использовании микропроцессора КР580ИК80 может оказаться еще бо́льшим, если учесть, что в случае применения в микропроцессорном устройстве оперативной памяти с низким быстродействием в цикле работы микропроцессора появятся «пустые» такты T_2, связанные с ожиданием появления сигнала на входе *Готовность*.

СИСТЕМА СБОРА ДАННЫХ

Рассмотрим пример, в котором микропроцессор используется для выполнения логических действий. Пусть устройство должно выполнять следующие функции: поступающие по восьми каналам аналоговые сигналы последовательно подключаются к АЦП и после преобразования их в цифровую форму запоминаются в оперативной памяти.

Работа устройства, структурная схема которого представлена на рис. 3.12, происходит в следующей последовательности. Адрес очередного канала указывается в трех младших разрядах данных, выдавае-

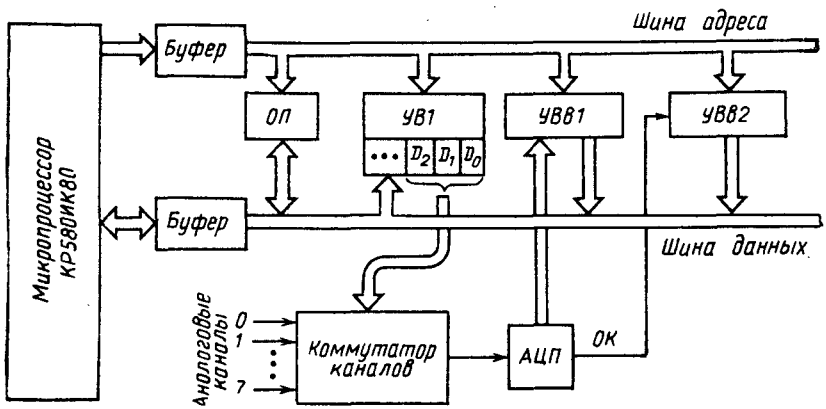


Рис. 3.12. Устройство сбора данных

мых из микропроцессора на шину данных. С шины данных адрес принимается устройством вывода УВ 1, откуда он поступает на адресные входы коммутатора. Сигнал выбранного канала передается на вход АЦП. После окончания преобразования АЦП выдает сигнал ОК = 1, устройством ввода УВв 2 он передается на шину данных, откуда принимается микропроцессором. После этого микропроцессор через устройство УВв 1 принимает сигнал канала, преобразованный в цифровую форму. Принятые данные микропроцессор передает в ОП.

Пусть для хранения данных в ОП выделены ячейки с адресами, начинающимися с адреса 1350_{16}. Адреса ячеек будем хранить в паре регистров HL.

Номер очередного канала будем хранить в трех младших разрядах регистра B, пять старших разрядов регистра заполним единицами. После каждого опроса канала будем увеличивать на единицу содержимое регистра B, формируя таким образом в нем номер очередного канала.

Таблица 3.12

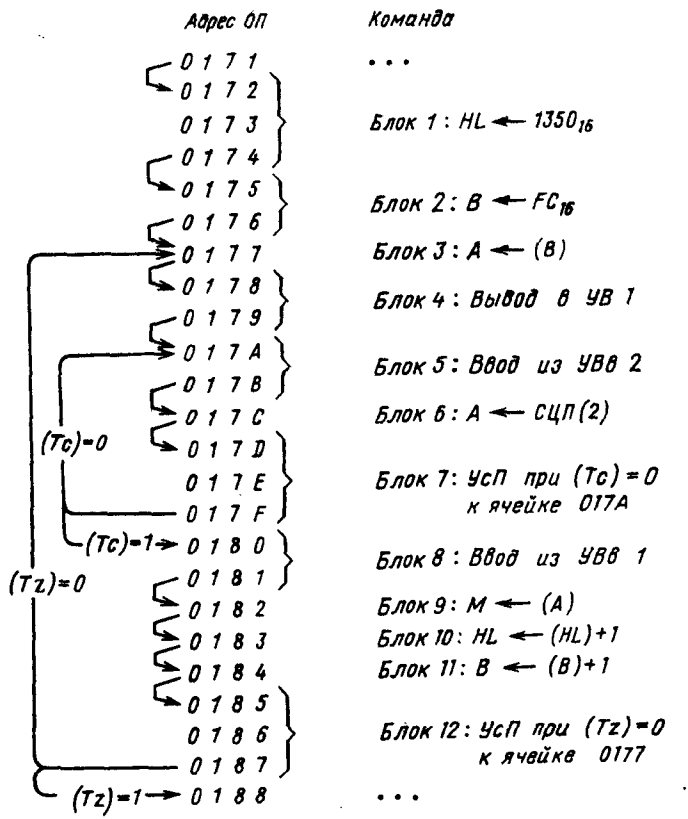


Таблица 3.13

Адрес команды в ОП	Комаида в кодовой комбиниации	Число байтов	Поясиения
0 1 7 1	...	...	...
0 1 7 2 0 1 7 3 0 1 7 4	0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 1	3	Блок *1*: HL←1350_{16}; $(B_3)=13_{16}$; $(B_2)=50_{16}$
0 1 7 5 0 1 7 6	0 0 0 0 0 1 1 0 1 1 1 1 1 1 0 0	2	Блок *2*: B←FC_{16}; $(B_2)=FC_{16}$
0 1 7 7	0 1 1 1 1 0 0 0	1	Блок *3*: A←(B)
0 1 7 8 0 1 7 9	1 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1	2	Блок *4*: Вывод в УВ 1; $(B_2)=01_{16}$
0 1 7 A 0 1 7 B	1 1 0 1 1 0 1 1 0 0 0 0 0 0 1 0	2	Блок *5*: Ввод из УВв 2; $(B_2)=02_{16}$
0 1 7 C	0 0 0 0 1 1 1 1	1	Блок *6*: A←СЦП (A)
0 1 7 D 0 1 7 E 0 1 7 F	1 1 0 1 0 0 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 1	3	Блок *7*: УсП при (Tc)=0 к ячейке 017A; $(B_3)=01_{16}$; $(B_2)=7A_{16}$
0 1 8 0 0 1 8 1	1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 1	2	Блок *8*: Ввод из УВв 1; $(B_2)=01_{16}$
0 1 8 2	0 1 1 1 0 1 1 1	1	Блок *9*: M←(A)
0 1 8 3	0 0 1 0 0 0 1 1	1	Блок *10*: HL←(HL)+1
0 1 8 4	0 0 0 0 0 1 0 0	1	Блок *11*: B←(B)+1
0 1 8 5 0 1 8 6 0 1 8 7	1 1 0 0 0 0 1 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 1	3	Блок *12*: УсП при (Tz)=0 к ячейке 0177; $(B_3)=01_{16}$; $(B_2)=77_{16}$
0 1 8 8	...	...	...

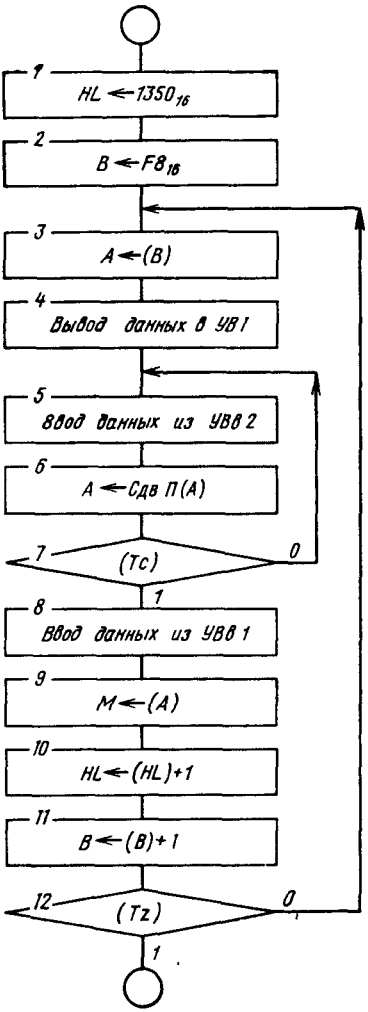


Рис. 3.13. Схема алгоритма сбора данных

После опроса последнего канала очередное прибавление единицы к содержимому регистра B сбросит регистр в нулевое состояние, что будет использовано в качестве признака окончания сбора данных.

После выдачи номера очередного канала микропроцессор принимает в аккумулятор A сигнал из УВв 2. Путем сдвига вправо содержимого аккумулятора принятое значение передается в триггер Tc. Если при этом (Tc) = 0 (это означает, что АЦП не закончил преобразование принятого аналогового сигнала), то микропроцессор повторяет прием из УВв 2, и так до тех пор, пока не будет принято значение ОК = 1.

После этого производится прием данных из УВв 1. Принятый байт данных запоминается в памяти. Затем прибавлением единицы к содержимому пары регистров HL в них формируется адрес ячейки, в которую будут переданы данные, полученные в результате преобразования сигнала следующего канала.

На рис. 3.13 показана схема алгоритма функционирования устройства.

В табл. 3.12 показано размещение программы в ОП.

В табл. 3.13 приведена программа.

3.4. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРА

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

До сих пор, записывая команды, мы пользовались языком кодовых комбинаций, единственно понятным для микропроцессора. Пользование этим языком вызывает трудности, связанные, во-первых, с необходимостью записи громоздких, труднозапоминаемых двоичных кодовых

комбинаций (использование для этих кодовых комбинаций представления в восьмеричной либо шестнадцатеричной системах счисления не приводит к существенному облегчению записи программы), во-вторых, сложностью поиска ошибок в составленной программе из-за того, что записанная с помощью кодовых комбинаций программа оказывается трудно читаемой, плохо обозримой, и, в-третьих, трудиостью внесеиия изменений в составленную программу.

Рассмотрим, в чем состоят указанные трудности, которые свазаны с внесением исправлений в программу, записанную в кодовых комбинациях.

Пусть исправление некоторого участка программы привело к тому, что после коррекции этот участок занимает меньшее число ячеек в ОП. Таким образом, в последовательности адресов ячеек, занимаемых программой, возникает разрыв (на рис. 3.14 A_{k+1} ... A_{k+m} — адреса m освободившихся ячеек памяти). Такие разрывы недопустимы. Это связано с тем, что счетчик команд микропроцессора после выборки очередной команды формирует адрес следующей команды путем увеличения своего содержимого на единицу. Следовательно, после выполнения последней команды участка программы перед разрывом счетчик в качестве адреса очередной команды укажет адрес ячейки A_{k+1}. Для устранения образовавшихся разрывов можио воспользоваться следующими приемами. Если $m \geqslant 3$, то в первые три ячейки разрыва следует поместить трехбайтовую команду безусловного перехода к ячейке A_{k+m+1} — первой после разрыва. Если число ячеек в разрыве меиьше трех, в эти ячейки помещается однобайтовая команда *Отсутствие операции*, имеющая кодовую комбинацию 00 000 000.

Рассмотрим другой случай, когда в результате коррекции некоторого участка программы выясняется, что исправленный участок программы не помещается в той группе ячеек, которая ранее отводилась под этот участок. В этом случае в указанную группу ячеек размещают начальную часть скорректированного участка, заканчивая ее трехбайтовой командой безусловного перехода к некоторой свободной ячейке, например следующей за ячейкой, которую занимает последняя команда программы (рис. 3.15). Начиная с этой ячейки, производится размещение команд конечной части скорректированного участка программы. Завершить эту часть участка программы необходимо командой безусловного перехода, как показано на рис. 3.15.

Следует иметь в виду, что подобные приемы приводят к появлению в программе дополнительных команд, за счет чего возрастают расходуемая емкость оперативной памяти и время исполнения программы. При большом числе исправляемых участков программы, по-видимому, целесообразно пересоставить всю программу, не пользуясь описанными выше приемами.

Наряду с указанными недостатками, использование языка кодовых комбинаций при программировании имеет и достоинства. Программируя на этом языке, программист может лучше использовать особенности микропроцессора и каждой его команды, наилучшим образом

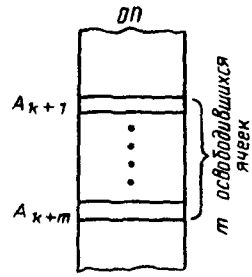


Рис. 3.14. Освобождение ячеек памяти в процессе отладки программы

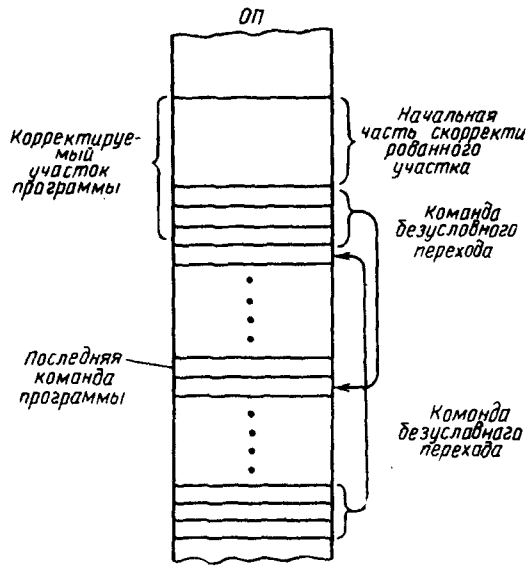


Рис. 3.15. Введение новых команд в процессе отладки программы

выполнять адресацию. При этом программа оказывается наиболее эффективной, она занимает минимальный объем в памяти и быстрее исполняется. Уменьшение объема занимаемой программой памяти (часто выполняемой в виде ПЗУ) снижает затраты на этот наиболее дорогостоящий узел микропроцессорного устройства, а уменьшение времени исполнения программы позволяет решать более сложные задачи. Кроме того, записанная на бумаге программа после ее набивки на перфоленту может быть непосредственно введена в память микропроцессорного устройства.

Трудности программирования уменьшаются при использовании *языка Ассемблера.* В этом языке вместо кодовых комбинаций используется мнемоническая форма записи операций, выполняемых в БИС микропроцессора. Такой мнемонической записью (в виде сочетания букв, взятых из соответствующих английских слов) представляют вид выполняемой операции, операнды и адреса. Каждой команде на языке Ассемблера соответствует команда на языке кодовых комбинаций.

Язык Ассемблера упрощает запись команд, обеспечивает лучший обзор программы, облегчает поиск в ней ошибок, обеспечивает простоту внесения исправлений в записанную на этом языке программу (не требуя привлечения специальных приемов, подобных тем, которые описывались выше для исправления написанных на языке кодовых комбинаций программ).

Перед исполнением программа должна быть переведена с языка Ассемблера на язык кодовых комбинаций и в таком виде помещена в память микропроцессорного устройства. Такой перевод осуществляется на ЭВМ с помощью программы трансляции, называемой *Ас-*

семблером. Языком Ассемблера можно пользоваться для программирования и в тех случаях, когда отсутствует программа для трансляции (отсутствует Ассемблер). Выполнение трансляции в этом случае производится вручную (такая трансляция называется *ручным ассемблированием*).

Язык Ассемблера (так же, как и язык кодовых комбинаций) индивидуален для каждого микропроцессорного комплекта, т. е. каждый микропроцессорный комплект имеет свой язык Ассемблера, отличный от языков Ассемблера других комплектов.

Следующий уровень языка программирования — *язык Макроассемблера*. В нем предусматривается возможность присвоения имени некоторой последовательности команд, и в любых местах программы, в которых должна быть использована эта последовательность, указывается лишь имя последовательности. Использование языка Макроассемблера сокращает запись программы (в среднем на 5...20 %) и тем самым улучшает ее обозримость.

Язык Макроассемблера (как и язык Ассемблера) индивидуален для МПК, т. е. программа, составленная на этом языке МПК одной серии, непригодна для использования в микропроцессорах, построенных на комплектах других серий.

Следующий уровень языка программирования — *язык высокого уровня*. Языки высокого уровня близки к обычному математическому языку, описывающему процесс решения задачи, поэтому они легко усваиваются.

Кроме того, они обеспечивают бóльшую компактность программы (сложные вычислительные процессы представляются короткими записями), что улучшает обзор программы и выявление в ней ошибок.

Различают *машинно-независимые* и *машинно-зависимые* языки высокого уровня. Первые позволяют вести запись прогрммы независимо от серии микропроцессорного комплекта, используемого для построения микропроцессорного устройства (к таким языкам относятся Бейсик, Фортран и др.). Вторые пригодны при применении определенных серий МПК. Для программирования устройств, построенных с использованием комплекта серии КР580, разработан язык высокого уровня PL M-80, относящийся к классу машинно-зависимых языков высокого уровня.

Языки высокого уровня требуют более сложных трансляторов для перевода программы на язык кодовых комбинаций (для машинно-независимых языков они сложнее, чем для машинно-зависимых языков), кроме того, полученная после трансляции программа занимает больший объем памяти (на 10...100 %) и медленнее исполняется, чем в том случае, когда эта программа составляется непосредственно в кодовых комбинациях. При этом эффективность программы, для составления которых используются машинно-независимые языки, обычно ниже, чем в случае использования машинно-зависимых языков программирования.

Для относительно несложных программ (например, объемом до одной тысячи команд) целесообразно использовать языки низкого уровня: язык кодовых комбинаций, язык Ассемблера или язык Макроассемблера.

ЯЗЫК АССЕМБЛЕРА

Программа на языке Ассемблера представляется в виде последовательности предложений, каждое из которых занимает отдельную строку. В табл. 3.14 показана запись на языке Ассемблера той же программы, которая на языке кодовых комбинаций приведена в табл. 3.9.

Каждое предложение языка Ассемблера содержит четыре фиксированных поля: поле метки, поле кода, поле операнда и поле комментария.

Поле метки. Если предложение снабжается именем, то оно записывается в поле метки и после имени ставится двоеточие. Имя строится в виде произвольно выбранной последовательности заглавных букв латинского алфавита и цифр, причем первым символом в имени должна быть буква. В приведенной в табл. 3.14 программе использованы имена M1 и M2. Обычно именами снабжаются предложения, на которые производится условный либо безусловный переход. Одно и то же имя не может встречаться в поле метки более одного раза. В противном случае возникает неясность, к какому предложению должен производиться переход по соответствующим командам условного и безусловного перехода.

Поле кода. В этом поле записывается мнемоническое обозначение кода операции, приводимое в системе команд микропроцессора.

Поле операнда. В поле операнда приводятся участвующие в операции числа (непосредственные данные), указания об источниках и приемниках данных, участвующих в операции; в предложениях условных и безусловных переходов в этом поле указывается имя (метка) предложения, на которое осуществляется переход. Числовые данные могут представляться в различных системах счисления. Для указания выбранной для представления числа системы счисления после шестнад-

Таблица 3.14

Метка	Код	Операнд	Комментарий
	MOV	A, B	; Блок *1*: A←(B)
	RRC		; Блок *2*: A←СЦП (A)
	JC	M1	; Блок *3*: УсП при (Tc) = 1
	MOV	A, C	; Блок *4*: A←(C)
	JMP	M2	; Безусловный переход
M1:	MOV	A, D	; Блок *5*: A←(D)
M2:	ADD	B	; Блок *6*: A←(A)+(B)
	MOV	B, A	; Блок *7*: B←(A)

цатеричного числа ставится символ H (а если число начинается с букв A, ..., F, то перед числом ставится цифра 0), после десятичного числа можно ставить символ D (либо не записывать никакого символа), восьмеричное число заканчивают символом Q, двоичное — символом B.

Например, пусть требуется загрузить в регистры E, C, D соответственно числа 101101_2, 217_8, 37_{10} и в пару регистров HL число $A195_{16}$. Указанные действия описываются следующими предложениями на языке Ассемблера:

```
Метка   Код    Операнд      Комментарий
        MVI    E, 101101B   ; загрузка регистра E
        MVI    C, 217Q      ; загрузка регистра C
        MVI    D, 37        ; загрузка регистра D
        LXI    H, 0A195H    ; загрузка пары регистров HL
```

Вместо идентификаторов (имен) внутренних регистров микропроцессора B, C, D, E, H, L, M, A допустимо применять их адреса в любой системе счисления. Например, приведенные выше действия можно записать следующими предложениями:

```
Метка   Код    Операнд       Комментарий
        MVI    3, 101101B    ; загрузка регистра E
        MVI    1, 217Q       ; загрузка регистра C
        MVI    10B, 37       ; загрузка регистра D
        LXI    100B, 0A195H  ; загрузка пары регистров HL
```

Здесь в первом и втором предложениях адреса регистров E (011_2) и C (001_2) представлены в десятичной системе счисления; в третьем и четвертом предложениях в поле операнда адреса регистров D и H записаны в двоичной системе счисления.

В качестве операндов могут быть указаны счетчик команд идентификатором PC и двухбайтовое содержимое аккумулятора вместе с регистром признаков — идентификатором PSW. В командах ввода (IN) и вывода (OUT) в поле операнда указывается номер устройства, с которым процессор обменивается данными.

В поле операнда допускается использование выражений, которые строятся путем связывания рассмотренных выше данных символами арифметических операций: + (сложение), — (вычитание), * (умножение), / (деление с выделением целой части частного), MOD (целый остаток от деления) и символами логических операций: NOT (инвертирование всех разрядов), AND (поразрядная конъюнкция), OR (поразрядная дизъюнкция), XOR (поразрядное суммирование по модулю 2), SHR и SHL (сдвиг первого операнда соответственно вправо и влево на число

разрядов, задаваемое значением второго операнда; освобождающиеся при сдвиге разряды заполняются нулями). Например,

```
Метка  Код      Операнд          Комментарий
       MVI  E,  0AH+17*3/2   ; загрузка в регистр
; числа 10+17·3/2=10+51/2=10+25=35
       JMP      M1+2         ; переход к команде
; с адресом, на две единицы бóльшим адреса предложения с
; меткой M1
```

Однако использование выражений лишает программу на языке Ассемблера наглядности, простоты ее чтения. Поэтому выражения при программировании на языке Ассемблера применяются редко и в данном пособии не рассматриваются.

Поле комментария. Начинается символом ; (точка с запятой). Оно служит для записи любых пояснений смысла выполняемых действий, которые могли бы облегчить чтение программы. Под комментарий можно выделять полные строки, начиная их символом ;. Приведенная в комментарии запись нужна лишь программисту, при трансляции она игнорируется Ассемблером.

ПРИМЕРЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ АССЕМБЛЕРА

В табл. 3.15 показана программа умножения, представляющая собой перевод на язык Ассемблера программы, приведенной в кодовых комбинациях в табл. 3.11.

Сложение многобайтных чисел. Пусть требуется сложить два четырехбайтных числа, каждое из которых занимает в ОП четыре ячейки с последовательно нарастающими адресами; адреса младших байтов первого и второго числа хранятся соответственно в парах регистров BC и HL. Результат сложения необходимо поместить в память на место второго слагаемого.

Таблица 3.15

Метка	Код	Операнд	Комментарий
	LXI	H, 0	; Блок *1*: Обнуление HL←0
	MVI	B, 8	; Блок *2*: Подготовка счетчика
LOOP2:	DAD	H	; Блок *3*: Сдвиг влево H
	RAL		; Блок *4*: Сдвиг влево A
	JNC	LOOP1	; Блок *5*: Условный переход
LOOP1:	DAD	D	; Блок *6*: Суммирование
	DCR	B	; Блок *7*: Счет
	JNZ	LOOP2	; Блок *8*: Условный переход

Принцип сложения таких многобайтных чисел состоит в том, что вначале в микропроцессор вызываются младшие байты слагаемых. Байты суммируются, результат суммирования помещается в память на место младшего байта второго слагаемого; возникающий в процессе суммирования перенос из старшего разряда запоминается в триггере Тс регистра признаков. Затем в парах регистров BC и HL формируется адрес вторых байтов слагаемых, которые затем вызываются в микропроцессор и суммируются вместе с хранящимся в триггере Тс переносом, возникшим при сложении первых байтов, и т. д. Выполнение операции завершится после четырехкратного повторения указанных действий. На рис. 3.16 приведена схема алгоритма и в табл. 3.16 — программа на языке Ассемблера.

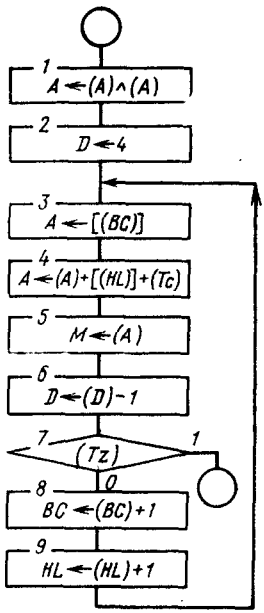


Рис. 3.16. Схема алгоритма многобайтового сложения

Программа деления. Операция алгебраического деления чисел содержит действия, связанные с определением знака частного, и действия, связанные с определением модуля частного и положительного остатка. Знак частного может быть определен выделением из чисел содержимого знаковых разрядов, затем суммированием их по модулю 2 и введением в знаковый разряд частного после того, как будет найден модуль частного. Ниже будем рассматривать наиболее сложную часть алгоритма деления, связанную с нахождением модуля частного.

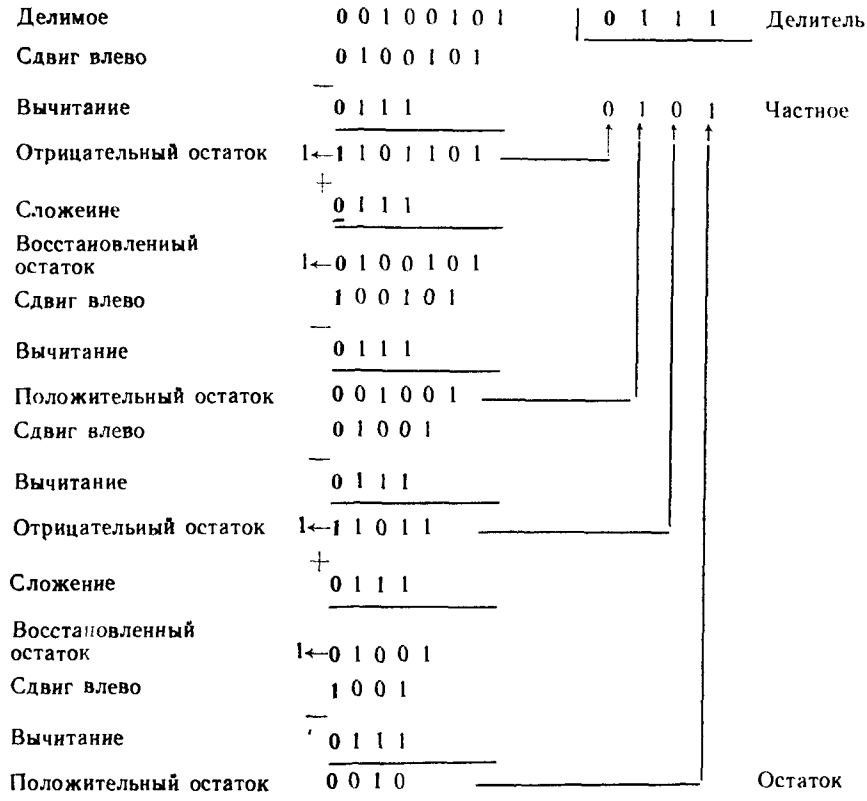
Пусть делимое и делитель — целые положительные числа; делимое имеет $2n$ разрядов, делитель n разрядов, их старшие разряды — знаковые и содержат 0.

Таблица 3.16

Метка	Код	Операнд	Комментарий
	ANA	A	; Установка Тс←0
	MVI	D, 4	; Подготовка счетчика
	LDAX	B	; Блок *3*
CYCLE:	ADC	M	; Блок *4*
	MOV	M, A	; Блок *5*
	DCR	D	; Блок *6*
	JZ	KI	; Блок *7*
	INX	B	; Блок *8*
	INX	H	; Блок *9*
	JMP	CYCLE	; Безусловный переход
KI:	...	...	

Рассмотрим алгоритм деления методом с восстановлением остатка. В качестве примера, на котором будет иллюстрирован алгоритм деления, выберем деление числа 37 на 7. При этом должно получиться частное 5 и остаток 2. Представим делимое 8-разрядным двоичным числом $\mathbf{0}\ 0100101_2$, делитель — 4-разрядным двоичным числом $\mathbf{0}\ 111_2$.

Ниже показан процесс деления:



Процесс деления сводится к следующей циклически повторяемой последовательности действий. В первом повторении цикла — делимое, а в последующих повторениях цикла — остаток сдвигаются на один разряд влево и затем из него вычитается делитель; если полученный новый остаток — положительное число, то в очередной разряд частного (начиная с его старшего разряда) записывается 1; если новый остаток — отрицательное число, то в разряд частного заносится 0, а к остатку прибавляется делитель, и таким образом восстанавливается предыдущий сдвинутый остаток. Эти действия циклически повторяются n раз (n — число разрядов делителя). В результате образуется частное, а последний остаток является результирующим остатком операции деления.

Таблица 3.17

Метка	Код	Операнд	Комментарий
	MVI	E, 8	; Блок *1*
	ANA	A	; Блок *2*
M2:	MOV	A, C	; Блок *3*
	RAL		; Блок *4*
	MOV	C, A	; Блок *5*
	MOV	A, B	; Блок *6*
	RAL		; Блок *7*
	MOV	B, A	; Блок *8*
	SUB	D	; Блок *9*
	CMC		; Блок *10*
	JC	M1	; Блок *11*
	ADD	D	; Блок *12*
M1:	DCR	E	; Блок *13*
	JNZ	M2	; Блок *14*
	MOV	B, A	; Блок *15*
	MOV	A, C	; Блок *16*
	RAL		; Блок *17*
	MOV	C, A	; Блок *18*

Для построения программы выполнения операции деления в микропроцессе примем $n = 8$. При этом делимое будет иметь $2n = 16$ разрядов (2 байта) и для своего хранения потребует пары регистров. Используем для хранения делимого пару регистров BC. При каждом сдвиге влево содержимого пары регистров BC в освобождающийся правый разряд будем заносить значение очередного разряда частного. Таким образом, после окончания выполнения операции в регистре C образуется частное. Так как действия вычитания (или сложения) делителя должны производиться над старшими восемью разрядами остатка, то эти операции будут выполняться над содержимым регистра B, а после окончания операции деления его содержимое будет результирующим остатком операции. Для хранения однобайтового делителя используем регистр D. На регистре E, предварительно загружаемом числом 8, построим счетчик числа повторений цикла.

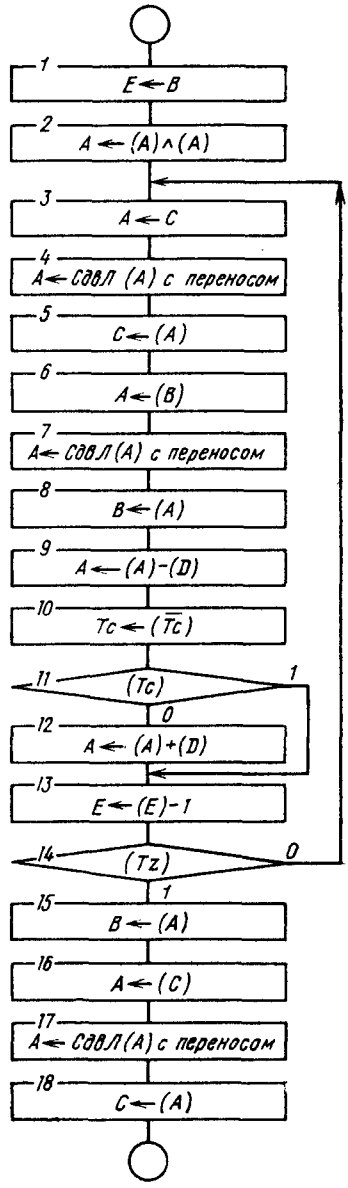


Рис. 3.17. Схема алгоритма деления

На рис. 3.17 показана схема алгоритма. Поясним некоторые ее особенности. Блок *2* осуществляет сброс в 0 содержимого триггера Тс регистра признаков, что может быть обеспечено выполнением любой логической операции АЛУ. Для выполнения сдвига влево содержимого пары регистров ВС использованы блоки *3* ... *8*. Вначале содержимое регистра С передается в аккумулятор А, где оно сдвигается, после чего возвращается в регистр С. При сдвиге в младший разряд принимается из триггера Тс сформированное в нем значение очередного разряда частного, выдвигаемое содержимое старшего разряда принимается в триггер Тс, откуда оно при выполнении сдвига содержимого регистра В будет принято в младший разряд.

Далее аналогичные действия производятся с содержимым регистра В. При выполнении в блоке *9* вычитания в случае отрицательного остатка возникает перенос 1 из старшего разряда, фиксируемый в триггере Тс, в случае положительного остатка этот перенос равен 0. Таким образом, значение очередного разряда частного может быть получено инвертированием содержимого триггера Тс, что и выполняет блок *10*. Блоки *11*, *12* выполняют действия, связанные с восстановлением положительного остатка. Блок *13* организует счет числа повторений цикла. После выхода из цикла в блоках *15* ... *18* производится передача в регистр В результирующего остатка операции деления и сдвиг содержимого регистра С с записью последнего сформированного в триггере Тс разряда частного.

В табл. 3.17 приведена записанная на языке Ассемблера программа, соответствующая рассмотренной схеме алгоритма операции деления.

ПСЕВДОКОМАНДЫ АССЕМБЛЕРА

Команды программы определяют действия, выполняемые над данными в процессе решения задачи. *Псевдокоманды* не связаны с действиями над данными, они сообщают необходимые Ассемблеру сведения и используются лишь в процессе трансляции программы на язык кодовых комбинацнй. С помощью таких псевдокоманд транслятору можно сообщить, например, сведения о том, с какого адреса памяти следует производить размещение команд программы; о том, что нужно зарезервировать в ОП ячейки (и в каком количестве) для хранения переменных, используемых в программе с определенными именами, или массивов переменных.

В табл. 3.18 приведены наиболее употребительные псевдокоманды.

Рассмотрим использование псевдокоманд в программе.

Пример 3.3. Пусть требуется определить в массиве данных количество элементов, значение которых совпадает с заданным значением некоторой константы.

В табл. 3.19 приведена программа.

Подобная программа может оказаться удобной в тех случаях, когда приходится, варьируя параметрами, многократно выполнять программу. Для изменения размерности массива данных и зиачения констан-

Таблица 3.18

Имя псевдокоманды	Пример записи			
	Метка	Код	Операнд	Комментарий
ORG		ORG	0200H	; Команды размещаются, начиная с ячейки ; с адресом 0200H
EQU	TIME	EQU	56	; Константе с именем TIME присваивается ; значение 56
DB	PR:	DB	37	; Для однобайтовой переменной с именем ; PR резервируется ячейка, в которую за- ; носится значение 37
DW	INIT:	DW	0	; Для двухбайтовой переменной с именем ; INIT резервируются в памяти две ячейки, ; в которые заносится 0
DS	ARR:	DS	100	; Для массива с именем ARR резервируют- ; ся 100 ячеек в памяти
END		END		; Конец программы

ты, с которой сравниваются значения элементов массива, здесь достаточно задать новые значения CN и NR в соответствующих псевдокомандах EQU. Кроме того, использование псевдокоманд DB и DS позволяет пользоваться символическими адресами TABLE и RES и избавляет от необходимости знать числовые значения адресов.

Таблица 3.19

```
        ORG     300H
        MVI     E, CN       ; Засылка в E размерности массива CN
        MVI     C, 0        ; Установка счетчика совпадений
        MVI     D, NR       ; Засылка константы в регистр D
        LXI     B, TABLE    ; Засылка в B адреса TABLE 1-го эле-
                            ; мента
M1:     LDAX    B           ; Загрузка A очередным элементом
                            ; массива
        CMP     D           ; Сравнение элемента массива с кон-
                            ; стантой
        JNZ     M2
        INR     C
M2:     INX     B           ; В регистре B адрес очередного эле-
                            ; мента
        DCR     E
        JNZ     M1
        MOV     A, C
        STA     RES         ; Засылка результата в память
        END
CN      EQU     100
NR      EQU     57
RES:    DB      0
TABLE:  DS      CN
```

3.5. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ PL/M-80

ЭЛЕМЕНТЫ ЯЗЫКА

Символы языка. Для записи программы язык PL/M-80 использует следующий набор символов:

а) 26 заглавных букв латинского алфавита: A, B, C, D, E, F,G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z;

б) при записи комментариев допускается использование заглавных букв русского алфавита, отсутствующих в латинском алфавите;

в) 10 арабских цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;

г) специальные символы:

¤ денежный знак,	= равно или присвоить,
+ плюс,	— минус,
* звездочка,	/ косая черта,
; точка с запятой,	: двоеточие,
. точка	' апостроф,
(левая скобка,	) правая скобка,
> больше,	< меньше
, запятая,	

Программа записывается в виде последовательности операторов. Конец каждого оператора обозначается символом «;».

Числовая константа. Числовая константа — это величина, значение которой, представленное числом, в процессе вычислений неизменно. В языке PL/M применяется беззнаковая целочисленная арифметика и, таким образом, все числа — целые без знака.

Для представления числовой константы могут использоваться различные системы счисления. Используемая система счисления обозначается соответствующей буквой вслед за числом: после шестнадцатеричного числа ставится буква H, если число начинается с одного из символов A, B, C, D, E, F, то дополнительно перед числом ставится цифра 0; после десятичного числа ставится буква D (допускается не ставить никакой буквы); после восьмеричного числа ставится буква O (либо Q); после двоичного числа — буква B.

Диапазон представимых чисел определяется условием, чтобы в двоичном представлении длина числа не превышала 2 байт, т. е. 0 ... 65535, (0FFFFH, 177777Q, 1111¤ 1111¤ 1111¤ 1111B). Символ ¤ при трансляции игнорируется и, таким образом, его можно ставить для облегчения чтения числа.

Символьная константа. Это последовательность символов, заключенная в апострофы. Такая символьная константа называется также литералом.

Переменная. Это величина, снабженная именем и в процессе выполнения программы могущая принимать различные значения. Имя переменной выбирается программистом в виде произвольной последова-

тельности букв и цифр, начинающейся обязательно с буквы и имеющей длину не более 31 символа. Символ ¤ при трансляции игнорируется, он может включаться в имя для разделения его на части, облегчающие чтение имени.

Примеры имен переменных:

I, TIME, ALPHA, A, X, Y, Z.

Каждая переменная должна быть описана в операторе DECLARE (операторе объявления). В описании переменной обязательно указывается ее тип и могут приводиться некоторые другие ее характеристики. В типе переменной программист определяет длину переменной, а именно: число ячеек ОП с разрядностью в 1 байт, необходимых для хранения значения переменной. Допускается использование только двух типов переменных: BYTE, означающий длину в 1 байт, и ADDRESS, означающий длину в 2 байта.

Например,

DECLARE X1 BYTE, TIME ADDRESS;

Здесь переменная с именем X1 объявлена как однобайтовая, переменная с именем TIME — двухбайтовая.

Слова BYTE и ADDRESS в операторе DECLARE могут выноситься за скобки, например,

DECLARE (X, Y, VAL) BYTE, (TIME, Z) ADDRESS;

Здесь переменные X, Y, VAL объявлены типа BYTE, переменные TIME, Z — типа ADDRESS.

Значения переменных, так же как и числовые константы, при выполнении операций рассматриваются целыми без знака.

Массив. Под массивом переменных понимают группу переменных, имеющих одно и то же имя. Это имя и является именем массива. Входящие в массив переменные различаются значениями индексов и называются переменными с индексами.

Например, пусть массив MATR содержит три переменных. Тогда эти переменные запишутся так: MATR (0), MATR (1), MATR (2). Особенность записи переменных с индексом состоит в том, что вслед за именем массива, в который входит переменная, записывается индекс, заключенный в круглые скобки (в отличие от принятого в обычной математике способа записи индексов ниже строки). Индекс может представляться любым выражением. Например, X (2*A+1). Здесь индексом переменной служит значение выражения 2*A+1.

Для каждого массива в операторе DECLARE должна быть объявлена размерность, т. е. количество переменных в массиве. Размерность указывается заключенным в скобки числом вслед за именем массива. Например,

DECLARE A (128) BYTE, (X, ALPHA) (100) ADDRESS;

Массив A содержит 128 однобайтовых переменных. При этом первая переменная массива имеет индекс 0, последняя — индекс 127. Таким образом, переменные массива A: A (0), A (1), ..., A (127).

Каждый из массивов X и ALPHA содержит по 100 двухбайтовых переменных.

Имя массива в программе не должно использоваться в качестве простой переменной (переменной без индекса).

Для представления двухмерных массивов (матриц), в которых переменные имеют два индекса, используется так называемая *структура*. Например, объявленная структура

```
DECLARE X (100) STRUCTURE (Y (50) BYTE);
```

может быть использована для описания матрицы с размерностью 100×50. Здесь каждый из 100 элементов массива X есть в свою очередь массив из 50 однобайтовых элементов Y. Примерами обращения к элементам такого двухмерного массива являются

X (5). Y (10), X (0). Y (6), ...

Таким образом, переменные с двумя индексами представляются двумя именами: первое имя X — имя массива строк матрицы, второе имя Y — имя массива переменных в строке. Первая из приведенных выше записей соответствует обращению к переменной, расположенной в 5-й строке и в 10-м столбце; вторая запись — обращению к переменной в нулевой строке и 6-м столбце.

Арифметические выражения. Строятся с использованием следующих знаков операций: + (сложение без использования содержимого триггера переноса Tc регистра признаков микропроцессора), — (вычитание без использования содержимого триггера Tc), * (умножение), /(деление), MOD (определение положительного остатка от деления), PLUS (сложение с прибавлением в младший разряд содержимого триггера Tc), MINUS (вычитание, при котором из полученной разности вычитается содержимое триггера Tc).

Рассмотрим некоторые особенности выполнения операций, связанные с использованием беззнаковой целочисленной арифметики по модулю $2^8 = 256$ или модулю $2^{16} = 65536$ в зависимости от типа результата (BYTE или ADDRESS). Пусть переменная X типа BYTE имеет значение 255. Тогда

X + 1 приводит к значению 0,
X + 2 приводит к значению 1.
Пусть переменная X типа BYTE имеет значение 1. Тогда
X — 1 приводит к значению 0,
X — 2 приводит к значению 255.

Таким образом, при выполнении операций +, —, PLUS и MINUS к результату прибавляется или из него вычитается число 256 (в случае типа ADDRESS прибавляется или вычитается 65536) так, чтобы образовалось положительное число заданного типа.

Допускается одноместная операция «—». Например, — X имеет то же значение, что и результат операции 0 — X.

Если хотя бы один из операндов имеет тип ADDRESS, то этот тип будет иметь и результат операции; в противном случае результат имеет тип BYTE.

Операции умножения и деления выполняются в предположении, что операнды — целые без знака, тип результата всегда ADDRESS. Если результат умножения не может быть представлен двухбайтовым числом, то он оказывается ошибочным. Деление выполняется нацело с сохранением целого частного; положительный остаток отбрасывается. При выполнении операции MOD отбрасывается частное от деления и сохраняется положительный остаток.

Порядок, в котором выполняются арифметические операции, определяется присвоенными операциям уровнями приоритета:

1-й уровень: одноместная операция — ,
2-й уровень: *, /, MOD,
3-й уровень: +, —, PLUS, MINUS.

При вычислении значения выражения вначале выполняются все операции 1-го уровня, затем операции 2-го уровня и в последнюю очередь — операции 3-го уровня. Операции с одним уровнем приоритета выполняются в том порядке, в каком они встречаются в выражении при его просмотре слева направо. Если необходимо нарушить указанную последовательность выполнения операций, то используются круглые скобки, как и в обычной математике.

При присвоении вычисленного значения выражения некоторой переменной это значение преобразуется к типу переменной. Преобразование вида ADDRESS ← BYTE выполняется добавлением слева к значению выражения нулевого байта; при преобразовании вида BYTE ← ← ADDRESS отбрасывается левый байт значения.

Пример 3.4. Рассмотрим порядок вычислений выражения в следующей программе:

```
DECLARE (U, X, Y, Z) BYTE;
. . . . . .
U=X—Y/Z*2;
. . . . . .
```

Пусть X = 60H (96D); Y = 0A3H (163D); Z = 03H.

Операция Y/Z, выполняемая первой, приводит к двухбайтовому значению 0036H (54D); выполняемая затем операция умножения на 2 дает результат 006CH (108 D); наконец, выполняется операция вычитания 60H — 006CH = 0060H — 006CH. Так как образуется отрицательный двухбайтовый результат, то к нему прибавляется число 2^{16} = 0FFFFH+1. Следовательно, 0060H+(0FFFFH+1—006CH)= = 0060H+0FF94H = 0FFF4H. Полученное двухбайтовое значение выражения должно быть присвоено однобайтовой переменной X. При этом происходит преобразование типа BYTE ← ADDRESS и X = 0F4H.

Логическое выражение. Строится с использованием операций отношения: = (равно), < (меньше), > (больше), <= (меньше или равно), >= (больше или равно), <> (не равно). Результатом операции отношения является логическая величина, имеющая тип BYTE и равная 0FFH (в случае выполнения условия) либо 00H (при невыполнении условия). Логические выражения могут содержать логические операции NOT (инверсия НЕ), AND (конъюнкция И), OR (дизъюнкция ИЛИ), XOR (исключающее ИЛИ), выполняемые над логическими величинами. Логические операции выполняются над операндами поразрядно. Порядок их выполнения в выражении определяется следующими уровнями приоритета:

1-й уровень: NOT,

2-й уровень: AND,

3-й уровень: OR, XOR.

В выражении, содержащем арифметические операции, операции отношения и логические операции, выполняются вначале все арифметические операции, далее все операции отношения и затем логические операции.

Пример 3.5. Рассмотрим пример вычисления логического выражения в следующей программе:

```
DECLARE B ADDRESS, (X, A) BYTE;
. . .
X = 100; A = 200;
B = NOT X - 22 <= -A AND X + 20 > (50 - A)/3;
END;
```

При вычислении выражения операции выполняются в следующей последовательности:

```
VIII   IV   VI    II    IX    V    VII   I    III
NOT X - 22 <= -A AND X + 20 > (50 - A)/3;
I: 50 - A => 50 - 200 = 256 - 150 = 106;
II: -A => 256 - 200 = 56;
III: (50 - A)/3 => 0035 (результат деления - двухбайтовое значение);
IV: X - 22 => 100 - 22 = 78;
V: X + 20 => 100 + 20 = 120;
VI: X - 22 <= -A => 78 <= 56 => 00H;
VII: X + 20 > (50 - A)/3 => 0FFH (результат операции отношения - однобайтовая величина)
VIII: NOT X - 22 <= -A => NOT 00H => 0FFH;
IX: B = 00FFH (однобайтовое значение 0FFH преобразуется в двухбайтовое 00FFH).
```

ОПИСАНИЕ РАЗВЕТВЛЯЮЩИХСЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Для описания разветвлений используется оператор IF. Этот оператор имеет две формы.

Первая форма оператора:

```
IF выражение THEN оператор 1;
оператор 2;
...
```

Этому разветвляющемуся процессу соответствует схема алгоритма, представленная на рис. 3.18. В вычисленном значении двоичного представления выражения проверяется младший разряд. Если он равен 1, то выполняются оператор 1 и затем оператор 2, ...; в противном случае оператор 1 не выполняется и происходит переход непосредственно к выполнению оператора 2.

Пример 3.6. Описать вычисление $y^2 = x^2/R$, где x — переменная, представленная графиком на рис. 3.19, *а*. На рис. 3.19, *б* показана

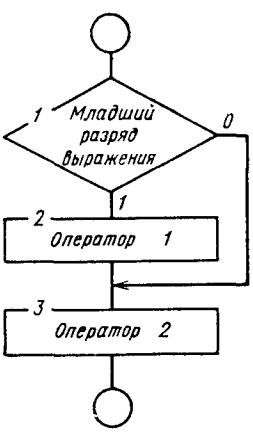
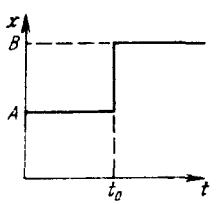
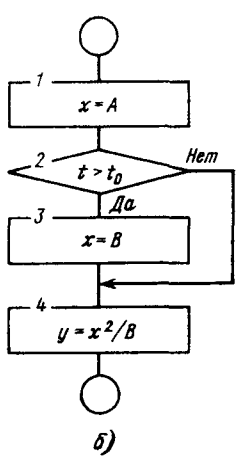


Рис. 3.18. Схема алгоритма, соответствующая первой форме оператора разветвления IF



а)



б)

Рис. 3.19. Пример разветвляющегося процесса:
а) сигнал; *б)* схема алгоритма, описывающего сигнал

схема алгоритма вычислений. Ей соответствует следующее описание на языке PL/M:

```
. . .
X = A;
IF T > T0 THEN X = B;
Y = X*X/R;
. . .
Вторая форма оператора;
IF выражение THEN оператор 1;
                ELSE оператор 2;
оператор 3;
```

Описываемому этим оператором разветвлению соответствует схема алгоритма на рис. 3.20. Вычисляется выражение и проверяется значение младшего разряда полученного результата. Если оно равно 1, то выполняется оператор 1 и затем оператор 3; если младший разряд выражения содержит 0, то выполняется оператор 2 и затем оператор 3.

Рассмотренный выше пример с использованием данной формы оператора разветвления решается представленной на рис. 3.21 схемой алгоритма и следующим описанием алгоритма на языке PL/M:

```
. . .
IF T > T0 THEN X = B;
          ELSE X = A;
Y = X*X/R;
```

Пример 3.7. Рассмотрим пример, представленный на рис. 3.22, *а*. Его решение в форме схемы алгоритма показано на рис. 3.22, *б*.

Особенность этой схемы алгоритма в том, что каждая ветвь разветвлений содержит не один, а два оператора. Это требует при записи оператора разветвления указания о том, что при каждом исходе проверки условия должна выполняться группа из нескольких (в данном примере из двух) операторов. Объединение операторов в группу производится путем их заключения в так называемые операторные скобки:

DO; <u>оператор 1; оператор 2; . . .</u>; END;
<u>группа операторов</u>

Здесь DO — открывающая скобка, END — закрывающая скобка образуют так называемую простую DO-группу. Таким образом, оператор разветвления для примера на рис. 3.22, *б* примет следующий вид:

```
. . .
IF T > T0 THEN DO; X = B; Y = C; END;
          ELSE DO; X = A; Y = D; END;
Z = (X*X + Y*Y)/R;
. . .
```

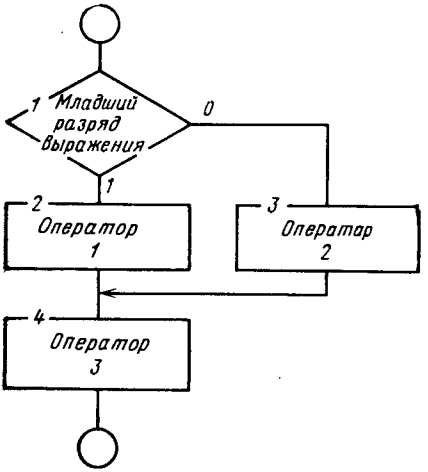


Рис. 3.20. Схема алгоритма, соответствующего второй форме оператора разветвления IF

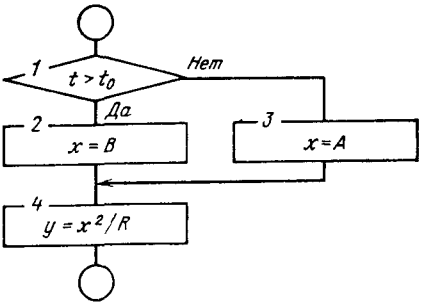


Рис. 3.21. Схема алгоритма разветвляющего процесса

Пример 3.8. Требуется выполнить операцию алгебраического сложения однобайтовых чисел X и Y, если старший разряд чисел — знаковый и числа представлены в прямом коде.

На рис. 3.23 приведена схема алгоритма рассматриваемого вычисления. В ней предусматривается получение дополнительных кодов X1 и Y1 исходных чисел X и Y, сложение дополнительных кодов чисел и преобразование суммы в прямой код.

Рассмотрим формирование дополнительного кода X1 числа X. Дополнительный код положительного числа совпадает с прямым кодом числа. Преобразование требуется в случае отрицательного числа, т. е.

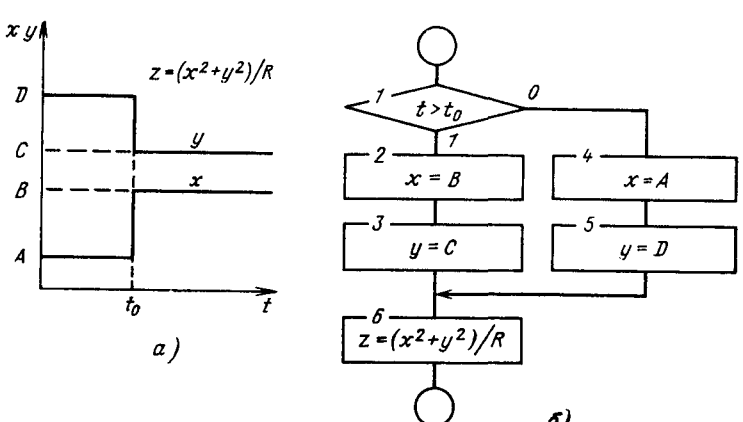


Рис. 3.22. Использование простой DO-группы:
а) пример; *б)* схема алгоритма

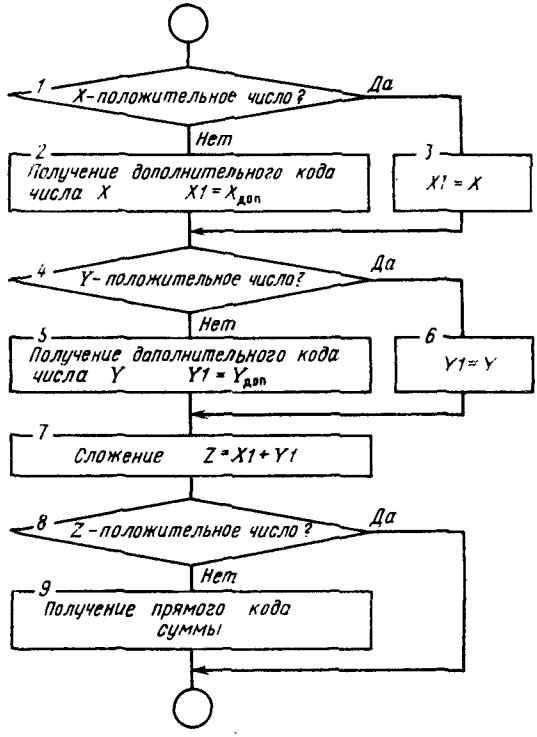


Рис. 3.23. Схема алгоритма алгебраического сложения чисел

если знаковый разряд числа содержит 1. Знак числа может быть выявлен проверкой условия X < 80H. Число 80H в двоичном представлении равно 1000000B, т. е. содержит 1 в старшем разряде и нули в остальных разрядах. Так как при выполнении операций в языке PL/M числа рассматриваются как положительные, то неравенство окажется выполненным, если X будет содержать 0 в старшем разряде, т. е. будет представлять положительное число. В случае, если X — отрицательное число, т. е. неравенство X < 80H окажется невыполненным, дополнительный код числа может быть получен оператором X1 = = (X XOR 7FH) + 1;

Число 7FH = 01111111B и выполнение операции поразрядного исключающего ИЛИ X XOR 7FH приведет к тому, что старший (знаковый) разряд X останется неизменным, значения остальных разрядов числа X будут инвертированы. Таким образом получается обратный код, а прибавление 1 приводит к дополнительному коду.

Теперь нетрудно написать программу алгебраического суммирования чисел:

```
DECLARE (X; X1, Y1, Y, Z) BYTE;
. . .
IF X < 80H THEN X1 = X;
                ELSE X1 = (X XOR 7FH) + 1;
IF Y < 80H THEN Y1 = Y; ELSE Y1 = (Y XOR 7FH) + 1;
Z = X1 + Y1;
IF Z < 80H THEN;
                ELSE Z = (Z XOR 7FH) + 1;
. . .
```

В последнем операторе IF после THEN оператор отсутствует (в этом случае говорят, что присутствует пустой оператор) и не выполняется никаких действий.

ОПИСАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Для описания циклических процессов используется оператор DO. Ранее был рассмотрен простой DO-оператор при объединении нескольких операторов в группу. Для организации циклов используются три других типа оператора DO: DO-WHILE-оператор, итеративный DO-оператор и DO-CASE-оператор.

DO-WHILE-оператор. Форма оператора

```
DO WHILE выражение;
   оператор 1;
   . . .                  } тело цикла
   . . .
   оператор n;
END;
```

Такая запись вызывает многократное исполнение группы операторов тела цикла: оператор 1; ...; оператор *n*; , пока, младший разряд результата вычисления выражения в двоичном представлении содержит 1 (нечетный результат). При этом выражение вычисляется при первом вхождении в блок до выполнения оператора 1 и каждый раз при достижении оператора END. Если при очередном вычислении выражения результат окажется четным, управление передается следующему за END оператору. Этот процесс будем представлять схемой алгоритма, показанной на рис. 3.24.

Пример 3.8. Вычислить значение sin x при $0 \leqslant x \leqslant 1$ суммированием ряда Маклорена

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \ldots = \sum_{i=1}^{k} (-1)^{i-1} \frac{x^{2i-1}}{(2i-1)!}.$$

Суммирование будем продолжать, пока модуль очередного члена не окажется меньше допустимой погрешности ε.

Пусть a_i, a_{i+1} — модули соответствующих членов ряда. Они определяются выражениями

$$a_i = \frac{x^{2i-1}}{(2i-1)!}, \quad a_{i+1} = \frac{x^{2(i+1)-1}}{(2(i+1)-1)!}.$$

Выразим значение a_{i+1} через значение a_i:

$$a_{i+1} = a_i \frac{x^2}{2i(2i+1)}.$$

Знак члена определяется следующим условием: i-й член положителен, если i нечетно.

Приведенная на рис. 3.25 схема алгоритма описывает процесс суммироваиия членов ряда. В блоке *1* значению первого члена присваивается a и сумме членов s, после чего происходит вхождение в цикл. Пока выполняется условие $a > \varepsilon$ (т. е. пока не достигнута требуемая точность), в блоке *3* вычисляется модуль очередного члена, в блоке *4* — его индекс; далее в блоке *5* выявляется, нечетно или четно значение i.

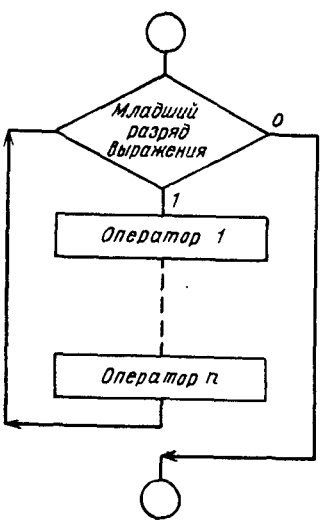


Рис. 3.24. Схема алгоритма, соответствующего циклическому оператору DO-WHILE

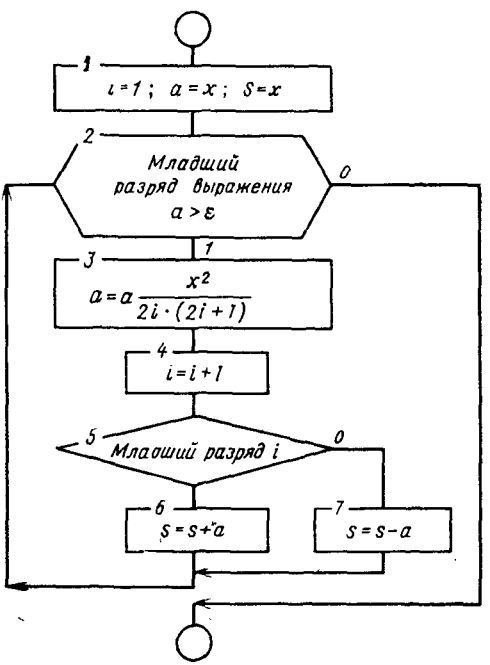


Рис. 3.25. Схема алгоритма вычисления sin x

При нечетном i выполняется оператор *6*, при четном i — оператор *7*. После достижения требуемой точности происходит выход из цикла.

Программа на языке PL/M:

```
DECLARE (I, A, X, EPS, S) BYTE;
. . .
I = 1; A = X; S = X;
DO WHILE A > EPS;
A = A*X/256*X/256/(2*I)/(2*I + 1);
I = I + 1;
IF I THEN S = S + A;
     ELSE S = S — A;
END;
```

По смыслу переменные A, X, S, EPS являются величинами, меньшими единицы. Поэтому их значения в программе следует рассматривать как дробные части чисел. В операторе, вычисляющем значение A, предусмотрено дважды деление на 256. Введение этой операции связано с необходимостью исключения переполнения разрядной сетки в процессе выполнения операций (деление на 256 может быть заменено выполнением так называемой встроенной процедуры выделения старшего байта, которая будет рассмотрена в дальнейшем).

Итеративный DO-оператор. Форма оператора:

```
DO i = a TO b BY c;
   оператор 1;  \
   . . .         } тело цикла
   . . .         /
   оператор n;  /
END;
```

Здесь i — простая переменная, называемая индексом цикла; a, b, c — выражения, определяющие соответственно начальное значение, конечное значение и шаг изменения индекса цикла.

Выполнение оператора происходит в следующем порядке. Индексу цикла i присваивается значение выражения a, вычисляется выражение b и его значение сравнивается с i. Если $i > b$, то управление передается следующему за END оператору; в противном случае выполняются операторы тела цикла, после чего вычисляются вновь выражения b и c, индексу цикла присваивается значение $i = i + c$ и вновь проверяется неравенство $i > b$ и т. д. Часть BY c может быть опущена. В этом случае шаг изменения индекса цикла принимается равным единице.

Таким образом, итеративный DO-оператор позволяет повторять выполнение некоторой группы операторов фиксированное число раз.

На рис. 3.26 показано представление итеративного DO-оператора в схеме алгоритма.

Пример 3.9. В массиве A из 100 однобайтовых переменных найти переменную, имеющую наибольшее значение.

Пусть x — искомое наибольшее значение переменных в массиве A. На рис. 3.27 приведена схема алгоритма решения сформулированной задачи. Алгоритм описывает циклический процесс. В каждом повторении тела цикла производится сравнение значения очередной выбранной в массиве A переменной a_i с максимальным из значений перемеиных, просмотренных в предыдущих повторениях тела цикла. Если неравенство $a_i > x$ выполняется, то x присваивается значение a_i. После 100-кратного повторения тела цикла происходит выход из цикла, при этом значение x есть искомое решение задачи.

Программа решения рассмотренной задачи на языке PL/M имеет следующий вид:

```
DECLARE (X, I, A (100)) BYTE;
X = 0;
DO I = 0  TO 99;
IF A (I) > X THEN X = A (I);
END;
```

Пример 3.10. Решить такую же задачу поиска максимального значения переменных в двухмерном массиве M размерностью 100×50.

Представленная на рис. 3.28 схема алгоритма решения данной задачи содержит два вложенных цикла: внешний цикл, в котором ведется

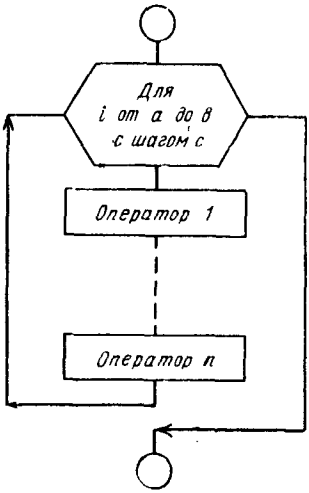


Рис. 3.26. Схема алгоритма, соответствующая итеративному DO-оператору

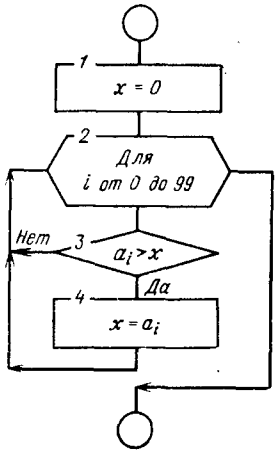


Рис. 3.27. Схема алгоритма вычисления максимального элемеита в одномерном массиве

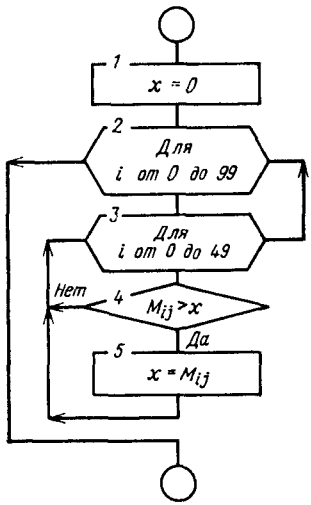


Рис. 3.28. Схема алгоритма вычисления максимального элемента в двумерном массиве

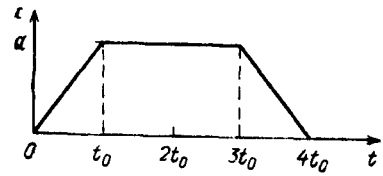


Рис. 3.29. Трапецеидальный импульс

просмотр строк i, и внутренний цикл, в котором просматриваются столбцы j. В каждом повторении тела цикла элемент матрицы M_{ij} сравнивается с максимальным значением x элементов, просмотренных в предыдущих повторениях тела цикла.

При построении программы положим, что массив строк матрицы M образует структуру A из 100 элементов. Каждый элемент структуры A есть массив B из 50 элементов.

Программа на языке PL/M имеет следующий вид:

```
DECLARE (X, I, J) BYTE, A (100) STRUCTURE (B (50) BYTE);
X = 0;
DO I = 0 TO 99;
DO J = 0 TO 49;
IF A (I).B (J) > X THEN X = A (I).B (J);
END;
END;
```

DO-CASE-оператор. Форма оператора:

```
DO CASE выражение;
    оператор 1;
    ...
    ...
    оператор n;
END;
```

Из группы операторов: оператор 1, ..., оператор n выполняется только один, номер которого определяет значение выражения. Значение выражения должно находиться в пределах 0... $(n - 1)$. Если оно равно k, то выполняется оператор с номером $(k + 1)$, после чего

происходит выход из блока и передача управления оператору, следующему за оператором END.

Пример 3.11. Требуется описать трапецеидальный импульс, представленный на рис. 3.29.

В аналитической форме импульс представляется выражением следующего вида:

$$x=\begin{cases} a\dfrac{t}{t_0} & \text{при } 0 \leqslant t \leqslant t_0; \\ a & \text{при } t_0 \leqslant t \leqslant 3t_0; \\ a\left(4-\dfrac{t}{t_0}\right) & \text{при } 3t_0 \leqslant t \leqslant 4t_0; \\ 0 & \text{при } t \geqslant 4t_0. \end{cases}$$

Описание этой функции на языке PL/M можно представить с помощью блока типа DO-CASE-оператора:

```
DECLARE (I, T, T0, X, A) BYTE;
IF T<4*T0 THEN I=T/T0;
          ELSE I=4;
DO CASE I;
X=A*T/T0;
X=A;
X=A;
X=4*A-A*T/T0;
X=0;
END;
```

Оператор IF вычисляет значение переменной I. Если выполняется условие T < 4*T0, то переменной I присваивается значение частного T/T0, которое принимает значение 0 при $t < t_0$, значение 1 при $t_0 \leqslant t < 2t_0$, значение 2 при $2t_0 \leqslant t < 3t_0$, значение 3 при $3t_0 \leqslant t < 4t_0$. Если неравенство T < 4*T0 не выполняется, то переменной I присваивается значение 4. В зависимости от присвоенного перемеиной I значеиия выполияется определенный оператор в блоке DO-CASE.

ПРОЦЕДУРЫ

Процедура — подпрограмма, в которой описано неоднократно встречающееся при решении задачи вычисление. Если в процессе исполнения главной программы встречается необходимость в проведении такого вычисления, производится обращение к процедуре. При обращении к процедуре приостанавливается выполнение главной программы и производится выполнение действий, предусматриваемых процедурой. После окончания исполнения процедуры происходит возврат в

главную программу в ту точку, в которой было приостановлено ее выполнение, и продолжается выполнение главной программы с использованием значений, полученных в результате выполнения процедуры.

Различают два вида процедур: процедуры-функции и процедуры общего вида. Каждая процедура снабжается именем, по которому производится обращение к ней.

В случае обращения к процедуре-функции ее имя используется при записи выражений как имя некоторой переменной, которая при выполнении процедуры приобретает некоторое значение. В отличие от переменной вслед за именем процедуры-функции в скобках приводятся аргументы, разделенные запятыми. Каждый из аргументов может быть представлен переменной либо выражением.

Пример 3.12. Используем ранее рассмотренный пример алгебраического суммирования чисел, представленных в прямом коде. Пусть имеется процедура-функция с именем D, описывающая способ преобразования числа из прямого кода в дополнительный код:

```
D: PROCEDURE (A, B);
DECLARE (A, B) BYTE;
IF A < 80H  THEN  B = A;
            ELSE  B = (X XOR 7FH) + 1;
RETURN B;
END D;
```

Здесь для некоторого A, у которого старший разряд — знаковый, показан способ вычисления дополнительного кода; A и B — так называемые *формальные параметры*, представляемые простыми скалярными переменными. Формальными эти параметры названы из следующих соображений. К процедуре можно обращаться для преобразования в дополнительный код значения любой переменной. Например, при преобразовании значения перемеиной X в дополнительный код X1 в процедуре формальные параметры A и B замещаются соответственно аргументами X и X1.

Пусть теперь в главной программе с именем OPT требуется произвести алгебраическое суммирование чисел X и Y с представлением их суммы в прямом коде. Эта программа может быть записана в следующем виде:

```
OPT: . . .
DECLARE (X, X1, Y, Y1, Z, Z1) BYTE;
/*ВЫЧИСЛЕНИЕ ДОПОЛНИТЕЛЬНОГО КОДА СУММЫ Z1*/
Z1 = D (X, X1) + D (Y, Y1);
/* ВЫЧИСЛЕНИЕ ПРЯМОГО КОДА СУММЫ Z*/
Z = D (Z1, Z);
. . .
END OPT;
```

Здесь заключенный между символами /* ...*/ текст есть комментарий, который игнорируется транслятором и служит лишь для облегчения чтения программы (для записи комментария допускается использование букв русского алфавита).

В этой программе при вычислении дополнительного кода суммы дважды производится обращение к процедуре-функции с именем D. При первом обращении D (X, X1) в процедуре-функции на место формальных параметров ставятся аргументы X и X1. Вычисленное в процедуре-функции значение дополнительного кода оператором RETURN B возвращается в главную программу. При втором обращении D (Y, Y1) в процедуру-функцию передаются аргументы Y и Y1 и оператор RETURN B возвращает в главную программу значение дополнительного кода Y1. Далее в главной программе вычисляется Z1, после чего оператор Z = D (Z1, Z) путем третьего обращения в процедуру-функцию преобразует дополнительный код суммы Z1 в прямой код Z.

Мы видели, что использование процедуры-функции имеет следующие особенности. В главной программе обращение к процедуре-функции пронзводится непосредственно из выраженя, в котором вместо переменной указывается имя процедуры-функции и за ним в скобках перечисляются аргументы, разделенные запятыми. Другая особенность состоит в том, что процедура-функция может передавать в главную программу значение лишь одной переменной.

Покажем решение той же задачи с использованием так называемой процедуры общего вида.

```
/* ПРОЦЕДУРА ОБЩЕГО ВИДА*/
D: PROCEDURE (A, B);
DECLARE (A, B) BYTE;
IF A<80H THEN B=A;
          ELSE B=(X XOR 7FH)+1;
RETURN;
END D;
/*ГЛАВНАЯ ПРОГРАММА*/
OPT:...
DECLARE (X, X1, Y, Y1, Z, Z1) BYTE;
CALL D (X, X1);
CALL D (Y, Y1);
Z1=X1+Y1;
CALL D (Z1, Z);
...
END OPT;
```

Здесь представлены следующие особенности использования процедуры общего вида, отличающие ее от процедуры-функции. В процедуре общего вида вместо оператора RETURN В использован оператор RETURN; в главной программе обращение к процедуре производится с помощью оператора CALL.

В отличие от процедуры-функции процедура общего вида может передавать в главную программу значения нескольких переменных. Покажем это на следующем примере. Пусть требуется получить в дополнительном коде попарную сумму трех переменных X, Y, Z.

```
/*ПРОЦЕДУРА ОБЩЕГО ВИДА*/
D: PROCEDURE (A1, A2, B1, B2, C1, C2);
DECLARE (A1, A2, B1, B2, C1, C2) BYTE;
IF A1 < 80H  THEN A2 = A1;
             ELSE A2 = (A1 XOR 7FH) + 1;
IF B1 < 80H  THEN B2 = B1;
             ELSE B2 = (B1 XOR 7FH) + 1;
IF C1 < 80H  THEN C2 = C1;
             ELSE C2 = (C1 XOR 7FH) + 1;
RETURN;
END D;
/*ГЛАВНАЯ ПРОГРАММА*/
OPT: . . .
DECLARE (X, X1, Y, Y1, Z, Z1, U1, U2, U3) BYTE;
CALL (X, X1, Y, Y1, Z, Z1);
U1 = X1 + Y1;
U2 = Y1 + Z1;
U3 = Z1 + X1;
. . .
END OPT;
```

Здесь при обращении к процедуре общего вида с именем D в главную программу передаются вычисленные в процедуре значения трех переменных X1, Y1, Z1.

ВСТРОЕННЫЕ ПРОЦЕДУРЫ И ПЕРЕМЕННЫЕ

Язык PL/M допускает пользование некоторым набором так называемых *встроенных процедур* и *переменных*, особенность которых в том, что от программиста не требуется их описания.

Ниже приводятся наиболее употребительные встроенные процедуры и переменные.

Процедура-функция INPUT (ввод). Процедура имеет тип BYTE и один формальный параметр. Форма обращения к ней:

INPUT (*a*)

где *a* — числовая константа, находящаяся в диапазоне 0 ... 255. Константа указывает номер устройства ввода. В результате обращения к процедуре из устройства ввода в процессор передается одно 8-разрядное двоичное число.

Пример 3.13. X = INPUT (5) +7;

Оператор присваивает переменной X сумму числа, полученного из устройства ввода *5*, и константы 7.

Процедура-функция OUTPUT (вывод). Форма обращения к процедуре:

OUTPUT (*a*) = *b*;

где *a* — числовая константа, находящаяся в диапазоне 0 ... 255 и являющаяся номером устройства вывода, *b* — выражение, байтовое значение которого подлежит выдаче из микропроцессора в устройство вывода.

Процедура-функция преобразования типов LOW, HIGH, DOUBLE. Обращение LOW (*a*) производит выборку из двухбайтового значения выражения *a* его младшего байта;

обращение HIGH (*a*) выбирает из двухбайтового значения выражения *a* его старшего байта;

обращение DOUBLE (*a*) байтовое значение выражения *a* преобразует в тип ADDRESS путем добавления старшего нулевого байта.

Пример 3.14. Выдать на устройство вывода 3 последовательно младший и старший байты двухбайтовой величины Z.

```
OUTPUT (3) = LOW (Z);
OUTPUT (3) = HIGH (Z);
```

Процедура-функция сдвигов SHL, SHR, SCL, SCR, ROL, ROR. Форма обращения к процедуре:

ИМЯ (*a*, *b*)

Здесь ИМЯ — имя процедуры-функции; *a* — выражение, значение которого (байтовое либо двухбайтовое) подлежит сдвигу; *b* — число разрядов, на которое производится сдвиг.

Процедуры-функции SHL, SHR выполняют логический сдвиг соответственно влево и вправо, вписывая в освобождающиеся разряды нуль;

процедуры-функции SCL, SCR выполняют циклический сдвиг соответственно влево и вправо с переносом (с включением в замкнутую цепь сдвига триггера переноса Тс регистра признаков микропроцессора);

процедуры-функции ROL, ROR выполняют циклический сдвиг соответственно влево и вправо без переноса (без включения в замкнутую цепь сдвига триггера Тс).

Пример 3.15. Выполнить операцию алгебраического умножения байтовых чисел X и Y, вводимых соответственно из устройств ввода 5 и 7; результат выдать в устройство вывода 3.

```
DECLARE (X, Y) BYTE, Z ADDRESS;
X = INPUT (5); Y = INPUT (7);
IF X >= 80H THEN U = 1; ELSE U = 0;
IF Y >= 80H THEN U = U XOR 1;
Z = ROR (SHR (SHL (X, 1), 1)*SHL (Y, 1) + U, 1);
OUTPUT (3) = LOW (Z); OUTPUT (3) = HIGH (Z);
END;
```

В программе определяется значение знаковых разрядов сомножителей и формируется знаковый разряд произведения U. Далее в результате выполнения операций SHR (SHL (X, 1), 1) множимое X сдвигается логически влево с потерей знакового разряда и затем логический сдвиг вправо восстанавливает число с вписанным в знаковый разряд нулем. Полученный модуль числа X умножается на сдвинутый логически влево множитель Y (SHL (Y, 1)). В результате образуется модуль произведения, сдвинутый на один разряд влево. К нему в младший разряд прибавляется значение знакового разряда произведения U, после чего циклический сдвиг вправо (ROR) устанавливает значение знакового разряда в старший разряд числа.

Процедура-функция DEC. Обращение DEC (*a*) преобразует байтовое значение *a* в две двоично-кодированные десятичные цифры.

Процедура общего вида ТИМЕ. Обращение к процедуре имеет вид:

```
CALL TIME (a);
```

Процедура обеспечивает задержку в работе микропроцессора, равную произведению байтовой константы *a* на 100 мкс. Таким образом, наибольшее значение задержки равно 25,5 мс.

Скалярная переменная STACKPTR. Двухбайтовая переменная STACKPTR является именем указателя стека и позволяет вызвать значение этого регистра либо, если эта переменная используется в левой части оператора присваивания, то производит засылку в регистр некоторого значения.

3.6. УЗЛЫ МИКРОПРОЦЕССОРНОГО УСТРОЙСТВА

ГЕНЕРАТОР ТАКТОВЫХ ИМПУЛЬСОВ КР580ГФ24

Структурная схема. Микросхема формирует две последовательности тактовых импульсов Ф1 и Ф2. Кроме того, она выполняет ряд других функций.

На рис. 3.30 приведены структурная схема микросхемы и схема подключения микросхемы генератора тактовых импульсов к микросхеме КР580ИК80.

К выводам КВ микросхемы КР580ГФ24 подключается кварцевый резонатор с частотой, в 9 раз более высокой, чем частота следования тактовых импульсов Ф1 и Ф2. Сформированные генератором гармонические колебания поступают на вывод *Осц* для контроля работы генератора и синхронизируют работу формирователя тактовых импульсов. На выводы Ф1 и Ф2 выдаются требуемые для работы микропроцессора

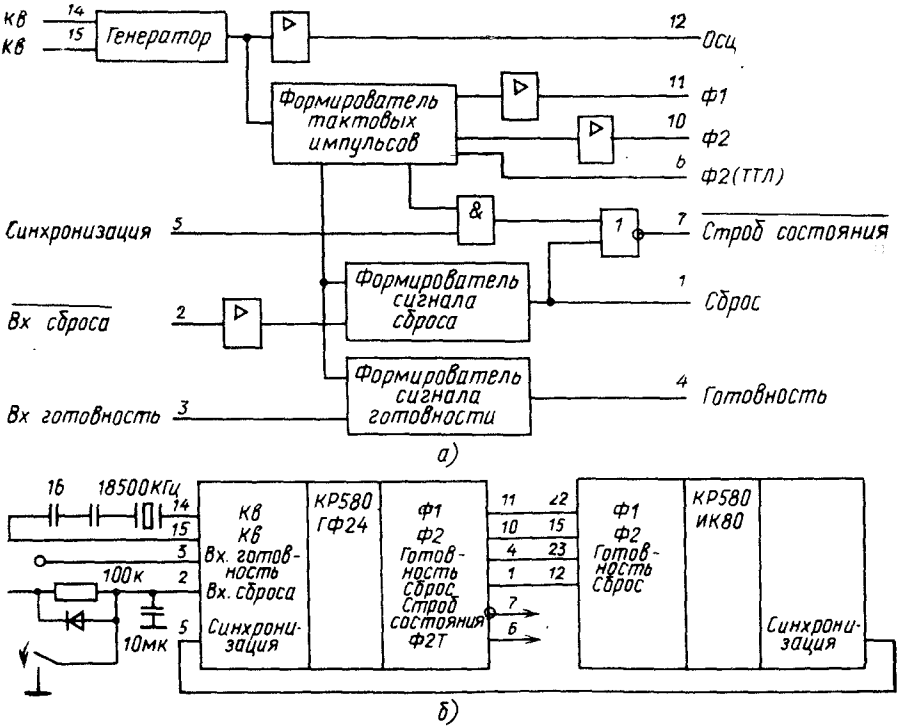


Рис. 3.30. Генератор тактовых импульсов КР580ГФ24:
а) структурная схема; *б)* подключение к процессорному элементу

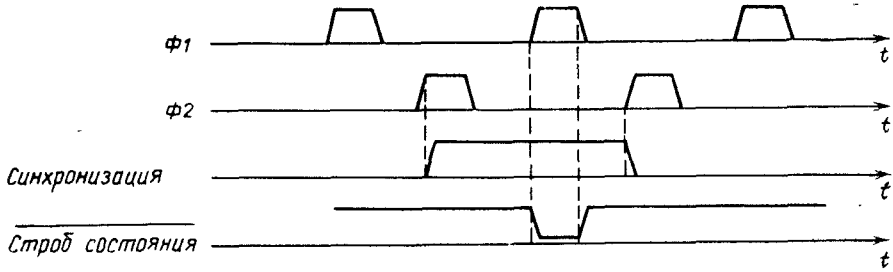


Рис. 3.31. Временные диаграммы формирования сигнала *Строб состояния*

высоковольтные последовательности тактовых импульсов. Кроме того, на специальный вывод выдается последовательность тактовых импульсов Ф2 с уровнями, характерными для микросхем ТТЛ.

С управляющего выхода *Синхронизация* микропроцессора на вход *Синхронизация* микросхемы КР580ГФ24 поступают импульсы, формируемые микропроцессором на границе между первым и вторым тактами каждого цикла. С помощью этого сигнала на вывод $\overline{Строб\ состояния}$ передаются импульсы Ф1, соответствующие началу каждого второго тактового период циклов работы микропроцессора (см. рис. 3.31).

Кроме того, предусмотрены вход и выход сигнала сброса ($\overline{Вх.\ сброса}$ и *Сброс*), вход и выход сигнала готовности (*Вх. готовность* и *Готовность*). Рассмотрим назначение и способы формирования этих сигналов.

Использование сигнала Сброс. Сигнал *Сброс* в микропроцессоре производит сброс в нуль счетчика команд РС. И если в ячейке ОП с нулевым значением адреса хранится 1-я команда программы, то с этой команды начинается исполнение программы. Таким образом, сигнал *Сброс* производит запуск микропроцессора. Входной сигнал $\overline{Вх.\ сброс}$, под действием которого в микросхеме КР580ГФ24 формируется сигнал *Сброс*, может подаваться одновременно с включением источников питания либо он может подаваться с пульта управления (нажатием соответствующей кнопки). В момент включения источников питания конденсатор 10 мк разряжен, напряжение на входе $\overline{Вх.\ сброс}$ равно нулю. При этом на выходе микросхемы КР580ГФ24 формируется сигнал *Сброс*, осуществляющий сброс микропроцессора. Далее током через резистор 100 к конденсатор начинает заряжаться. Когда напряжение на конденсаторе достигает определенного значения, снимается сигнал *Сброс* с выхода микросхемы и микропроцессор начинает выполнять программу с первой ее команды.

Сброс микропроцессора может быть выполнен замыканием показанного на рисунке ключа (путем нажатия соответствующей кнопки на пульте управления). При этом конденсатор разряжается и на выходе микросхемы возникает сигнал *Сброс*. После размыкания ключа (при

отжатии кнопки на пульте) конденсатор начинает заряжаться и в некоторый момент снимается сигнал *Сброс*, микропроцессор начинает выполнение программы.

Использование сигнала Готовность. В ряде случаев возникает необходимость приостановить выполнение микропроцессором программы и перевести его в режим ожидания. Режим ожидания может быть использован в тех случаях, когда узлы (ЗУ, УВВ), с которыми микропроцессор обменивается данными, не могут работать в высоком темпе работы микропроцессора. Если, например, ЗУ не может обеспечить выдачу запрашиваемых данных на шину данных за время, равное одному тактовому периоду работы микропроцессора, то потребуется перевод микропроцессора в режим ожидания на время, необходимое ЗУ для выдачи данных.

Перевод микропроцессора в режим ожидания может потребоваться в процессе отладки программы для установки пошагового режима выполнения программы, в котором после каждого цикла работы микропроцессор возвращается в режим ожидания. При этом по содержимому шин адреса и данных может быть определена правильность выполнения требуемых действий. Выход из режима ожидания производится вручную подачей соответствующего сигнала с пульта управления.

Другой используемый при отладке режим — командный — обеспечивает автоматический переход микропроцессора в режим ожидания после завершения выполнения каждой команды.

Во всех случаях перевод микропроцессора в режим ожидания достигается установкой низкого уровня лог. 0 на входе *Готовность* микропроцессора. В рассмотренном выше случае согласования темпов работы ЗУ и микропроцессора при каждом обращении к ЗУ потребуется установка на входе *Готовность* микропроцессора уровня лог. 0 в течение определенного числа тактовых периодов. В случае установки микропроцессора в шаговый или командный режим перевод сигнала *Готовность* на высокий уровень лог. 1 осуществляется вручную с пульта управления, перевод на уровень лог. 0 — автоматически с окончанием цикла работы микропроцессора либо после окончания выполнения команды.

Рассмотрим способы формирования сигнала *Вх. Готовность*, обеспечивающего в рассмотренных выше случаях необходимую форму сигнала *Готовность*.

Установка микропроцессора в режим ожидания на заданное время. Пусть необходимо обеспечить сопряжение микропроцессора с устройством, требующим для выдачи или приема данных времени, равного двум тактовым периодам. На рис. 3.32, *а* показана схема формирования сигнала *Вх. Готовность*, обеспечивающего в микросхеме КР580ГФ24 формирование сигнала *Готовность* установки микропроцессора в режим ожидания в течение двух тактовых периодов. Сигнал $\overline{Строб\ состояния}$ уровня лог. 0, появляющийся в момент положительного фронта тактового импульса Ф1 на границе 1-го и 2-го тактов каж-

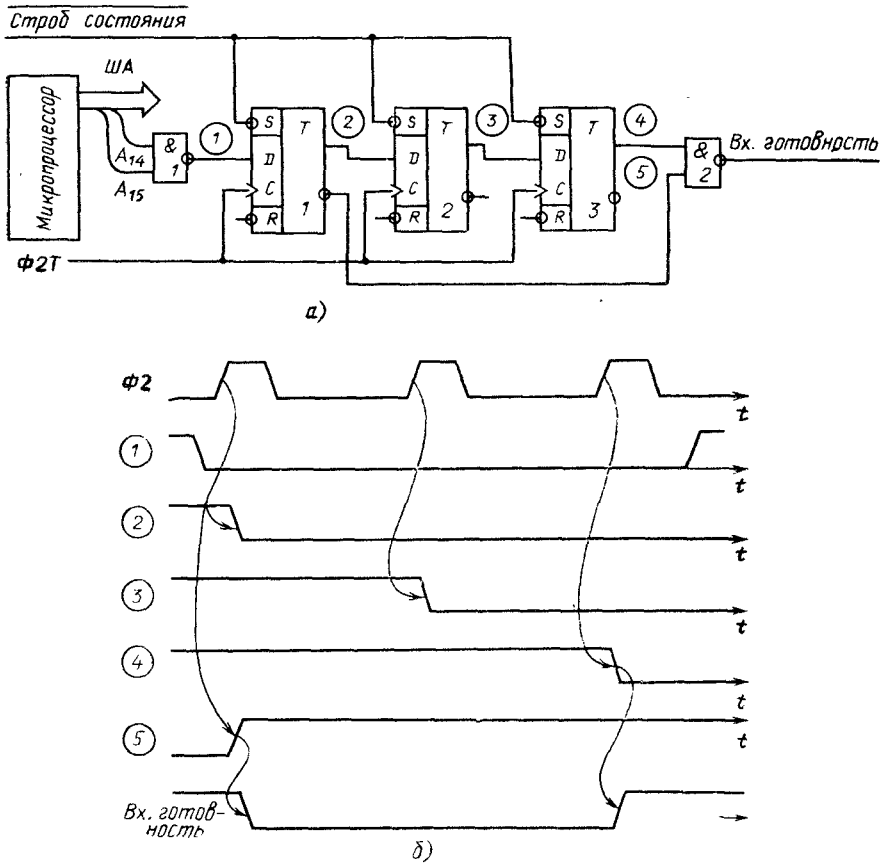


Рис. 3.32. Формирование сигнала *Вх. готовность* с заданной длительностью: *а*) схема; *б*) временные диаграммы

дого цикла, устанавливает все триггеры в состояние лог. 1. При этом *Вх. готовность* = 1.

Пусть адрес блока ЗУ, требующего установки микропроцессора в режим ожидания, содержит лог. 1 в разрядах A_{15} и A_{14} адресов. При обращении к этому блоку ЗУ образуется лог. 0 на выходе элемента И-НЕ *1*. С появлением положительного фронта тактового импульса Ф2 (см. временну́ю диаграмму на рис. 3.32, *б*) триггер *1* переводится в состояние лог. 0 и на выходе схемы устанавливается *Вх. Готовность*= =0. Далее в моменты положительного фронта очередных импульсов Ф2 состояние лог. 0 последовательно передается в триггеры *2* и *3*, после чего на выходе *Вх. готовность* восстанавливается уровень лог. 1. Таким образом, в течение двух тактов, пока *Вх. Готовность* =0, на выходе *Готовность* микросхемы КР580ГФ24 действует низкий уровень

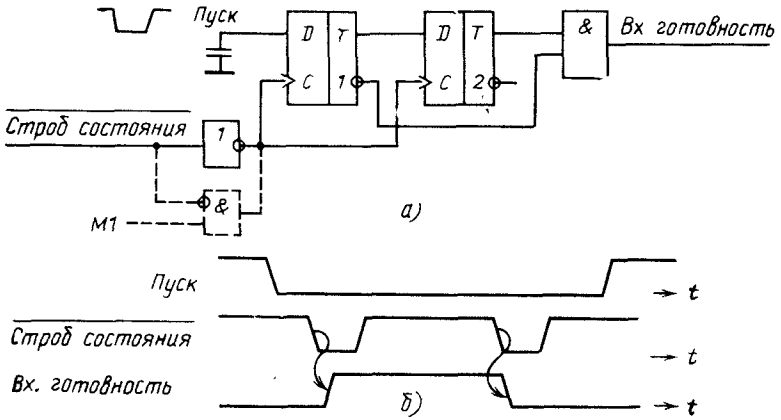


Рис. 3.33. Формирование сигнала *Вх. готовность* с длительностью цикла или времени исполнения команды:
а) схема; *б)* временные диаграммы

лог. 0, устанавливающий микропроцессор на это время в режим ожидания.

Шаговый режим. На рис. 3.33 приведены схема формирования сигнала *Вх. Готовность* для шагового режима и временная диаграмма ее работы.

В исходном состоянии сигнал на входе *Пуск* имеет уровень лог. 1 и оба триггера оказываются установленными в состояние лог. 1, на выходе схемы *Вх. Готовность* = 0 и микропроцессор находится в режиме ожидания. При установке на входе *Пуск* уровня лог. 0 отрицательный фронт сигнала *Строб состояния* устанавливает триггер *1* в состояние лог. 0, на выходе схемы *Вх. готовность* = 1, микропроцессор выходит из режима ожидания. По окончании одного цикла работы микропроцессора отрицательным фронтом очередного импульса сигнала $\overline{Строб\ состояния}$ устанавливается в состояние лог. 0 триггер *2*, на выходе *Вх. Готовность* = 0 и микропроцессор прекращает работу, возвращаясь в режим ожидания.

Для установки командного режима используется сигнал M1, образующийся в разряде D5 на выходе регистра фиксации слова состояния в 1-м цикле каждой выполняемой микропроцессором команды. Формирование сигнала *Вх. Готовность* в этом режиме может осуществляться схемой рис. 3.33, *а*, если на вход С триггеров подавать конъюнкцию *Строб состояния* $\wedge$ M1, как показано на рисунке штриховой линией.

БУФЕРЫ

Информация, выдаваемая микропроцессором на шины адреса и данных, может предназначаться большому числу различных устройств, подключенных к этим шинам (ОЗУ, ПЗУ, устройства ввода-вывода).

Однако выходы микросхемы КР580ИК80 допускают потребление подключенными к ним устройствами относительно небольшого тока. Значение тока через эти выходы при высоком уровне напряжения (уровне лог. 1) $I^1_{вых} \leqslant 0{,}1$ мА, при низком уровне напряжения (уровне лог. 0) $I^0_{вых} \leqslant 1{,}6$ мА. Такая нагрузочная способность обеспечивает подключение к выходам микропроцессора не более одного входа микросхемы ТТЛ. Низкая нагрузочная способность выходов микропроцессора связана с тем, что на кристалле микропроцессора размещено большое число транзисторов и для обеспечения требуемого теплового режима должно быть малым тепло, выделяемое каждым транзистором. Следовательно, малыми должны быть токи через транзисторы. Увеличение же нагрузочной способности выходов потребовало бы использования на выходах мощных транзисторов, через которые протекали бы большие токи, а это привело бы к большому выделению тепла и недопустимому повышению температуры кристалла.

Так как токи, потребляемые нагрузкой микропроцессора, обычно превышают указанные выше допустимые значения, в шины адреса и данных включаются буферы. Для построения таких буферов в МПК серии КР580 предусмотрены шинные формирователи КР580ВА86 и КР580ВА87.

Шинные формирователи КР580ВА86 и КР580ВА87. На рис. 3.34 показана логическая схема формирователя КР580ВА86, осуществляющего передачу 8-разрядных данных. На рисунке подробно изображе-

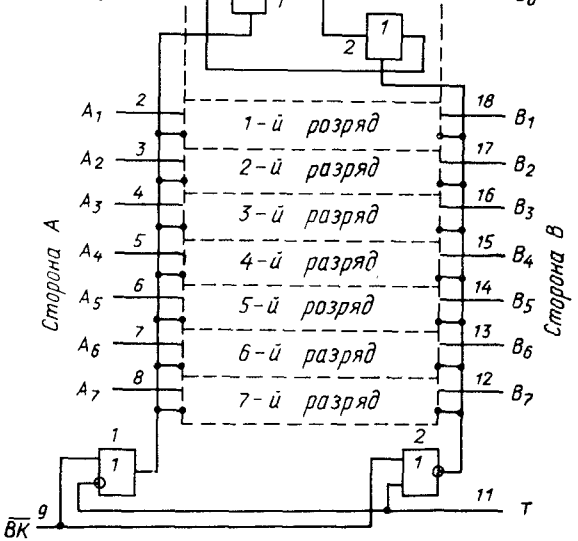


Рис. 3.34. Структурная схема шинного формирователя КР580ВА86

на схема лишь одного разряда, схемы остальных разрядов аналогичны. В цепи передачи включены два повторителя, имеющие три состояния. При этом если один из повторителей находится в включенном состоянии, другой будет находиться в выключенном (третьем) состоянии. Таким образом, если повторитель *1* установлен в включенное состояние, то повторитель *2* оказывается в выключенном состоянии и передача (в разряде, схема которого на рисунке представлена в развернутом виде) будет осуществляться через повторитель *1* в направлении от вывода 1 к выводу 19 (от A_0 к B_0). Если переключить повторители в обратное состояние, установив во включенное состояние повторитель *2*, повторитель *1* окажется в выключенном состоянии и передача будет происходить через повторитель *2* в направлении от вывода 19 к выводу 1 (от B_0 к A_0), т. е. в обратном направлении.

Управление состоянием повторителей осуществляется элементами ИЛИ-НЕ *1* и *2* с помощью управляющих сигналов $\overline{ВК}$ и Т. Если на входе $\overline{ВК}$ установлен высокий уровень лог. 1, то независимо от значения сигнала Т на выходах элементов ИЛИ-НЕ устанавливается низкий уровень лог. 0 и во всех разрядах оба повторителя оказываются установленными в выключенное состояние, и не происходит передачи информации ни в прямом, ни в обратном направлениях. При комбинации сигналов $\overline{ВК}=0$ и $Т=1$ на выходе элемента ИЛИ-НЕ *1* образуется высокий уровень лог. 1 и повторители *1* во всех разрядах оказываются во включенном состоянии; на выходе элемента ИЛИ-НЕ *2* — низкий уровень лог. 0, устанавливающий повторители *2* в выключенное состояние. Происходит передача 8-разрядных данных в направлении от А к В. При комбинации сигналов $\overline{ВК} = 0$ и $Т = 0$, наоборот, на выходе элемента ИЛИ-НЕ *2* устанавливается напряжение уровня лог. 1, открываются повторители *2*, на выходе элемента ИЛИ-НЕ *1* устанавливается напряжение уровня лог. 0 и повторители *1* оказываются в выключенном состоянии. Происходит передача 8-разрядных данных от стороны В к стороне А. Таким образом, шинный формирователь обеспечивает управляемую двунаправленную передачу 8-разрядных данных в соответствии с табл. 3.20.

Выходы В (при передаче в направлении от А к В) имеют бо́льшую нагрузочную способность, чем выходы А (когда происходит передача

Таблица 3.20

Значения управляющих сигналов		Направление передачи информации
$\overline{ВК}$	Т	
0	0	От стороны В к стороне А
0	1	От стороны А к стороне В
1	×	Передача отсутствует

в направлении от В к А). К выходам В допускается включение нагрузки, потребляющей ток $I^0_{вых} = 32$ мА, $I^1_{вых} = -5$ мА. Для выходов А эти токи $I^0_{вых} = 10$ мА, $I^1_{вых} = -1$ мА. Очевидно, шинный формирователь должен включаться стороной А к выводам микропроцессора, стороной В — к системным шинам адреса и данных.

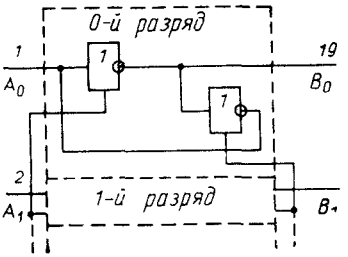


Рис. 3.35. Схема шинного формирователя КР580ВА87

На рис. 3.35 приведена схема шинного формирователя КР580ВА87. Ее отличие от КР580ВА86 состоит лишь в том, что включенные в разряды повторители имеют инвертирующие выходы и при передаче происходит инвертирование передаваемых данных. В остальном работа этой микросхемы аналогична работе рассмотренной выше микросхемы КР580ВА86.

Буфер шины адреса. Шина адреса имеет 16 разрядов, и так как шинный формирователь содержит 8-разрядный канал, для построения буфера потребуются 2 микросхемы. Их включение показано на рис.3.36.

Шина адреса однонаправленная, в качестве входного канала в шинных формирователях выбран канал А, в качестве выходного — канал В. Передача информации от А к В в шинном формирователе обеспечивается при напряжении уровня лог. 1 на входе Т, поэтому выводы Т

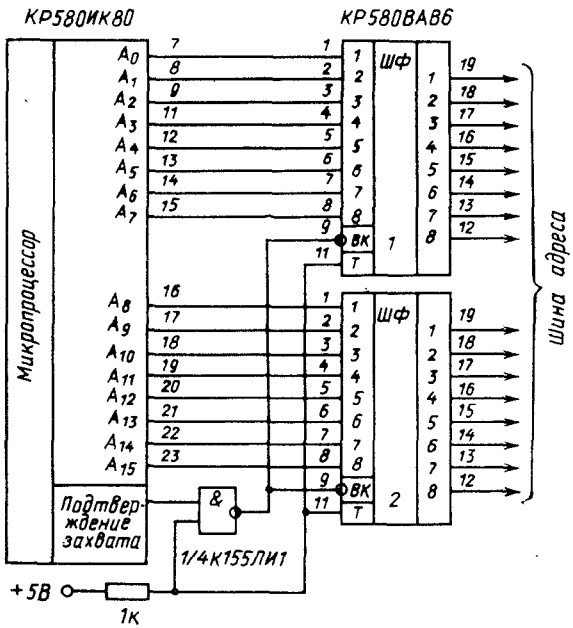


Рис. 3.36. Схема буфера шины адреса

микросхем шинных формирователей подключены к источнику питающего напряжения + 5 В. К выводам $\overline{ВК}$ микросхем подается формируемый микропроцессором сигнал *Подтверждение захвата*. Передача адреса с выхода микропроцессора через шинный формирователь происходит при низком уровне сигнала *Подтверждение захвата*. В режиме захвата микропроцессор выдает сигнал *Подтверждение захвата* высокого уровня, этим сигналом выходы шинных формирователей переводятся в высокоомное (выключенное) состояние, микропроцессор оказывается отключенным от шины адреса. Элемент 1/4 К155ЛИ1 (один из четырех двухвходовых элементов И в микросхеме) используется для увеличения нагрузочной способности выхода *Подтверждение захвата*.

Если в микропроцессорном устройстве не предусматривается режим захвата, то вместо шинных формирователей можно использовать инверторы К155ЛН1, К155ЛН3 или К155ЛН5 в зависимости от нагрузки на адресную шину.

Буфер шины данных. Шина данных имеет 8 разрядов с двунаправленной передачей информации. Для построения буфера достаточно одной микросхемы шинного формирователя, включенной по схеме с управляемой двунаправленной передачей информации. Схема включения шинного формирователя показана на рис. 3.37. Управление направлением передачи осуществляется с помощью сигнала *Прием*, формируемого микропроцессором. При высоком уровне сигнала *Прием* обеспечивается передача от шины данных к микропроцессору, при низком уровне — в обратном направлении.

Далее будет показано построение буфера шины данных с использованием микросхемы КР580ВК28, который наряду с функциями буфера способен выполнять ряд других функций (функции фиксатора состояния, формирователя системных управляющих сигналов). Шинный формирователь может выполнять функции буфера между шиной данных и устройствами ввода-вывода.

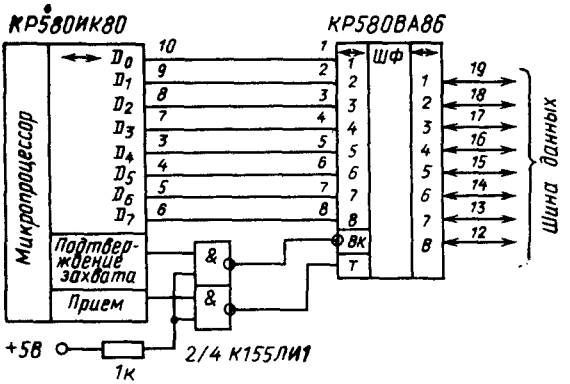


Рис. 3.37. Схема буфера шины данных

ФОРМИРОВАНИЕ УПРАВЛЯЮЩИХ СИГНАЛОВ МИКРОПРОЦЕССОРНОГО УСТРОЙСТВА

Шины данных и адреса в микропроцессорном устройстве являются общими для многих узлов, которые, будучи подключены к ним, могут принимать с шин либо выдавать в эти шины информацию. При таком обобществлении шин возникает необходимость в согласовании работы узлов: при выдаче информации с шин обеспечение ее поступления в соответствующие узлы, при приеме информации в шины ее поступление с определенного узла. Эти действия требуют выработки управляющих сигналов, формирование которых рассматривается ниже.

Сопряжение ОЗУ с шиной данных. Рассмотрим, какие управляющие сигналы требуются для сопряжения ОЗУ с шиной данных. Микросхемы ОЗУ могут иметь раздельные входы и выходы данных либо общие выводы данных. В последнем случае в режиме записи на общие выводы данных принимаются записываемые данные, а в режиме чтения на эти выводы выдаются считанные из ОЗУ данные.

Рассмотрим сопряжение с шиной данных микросхем с раздельными входами и выходами данных. Если выходы в самой микросхеме ОЗУ не снабжены элементами с тремя состояниями, то между этими выходами и шиной данных необходимо включить буферы с тремя состояниями, как показано на рис. 3.38.

В режиме хранения или записи буфер устанавливается в выключеиное состояние. В шину данных информация может поступать из микропроцессора либо из устройств ввода. В режиме чтения буфер устанавливается в открытое состояние и считанная из ОЗУ информация поступает в шину данных. Режим микросхем ОЗУ устанавливается сигнала-

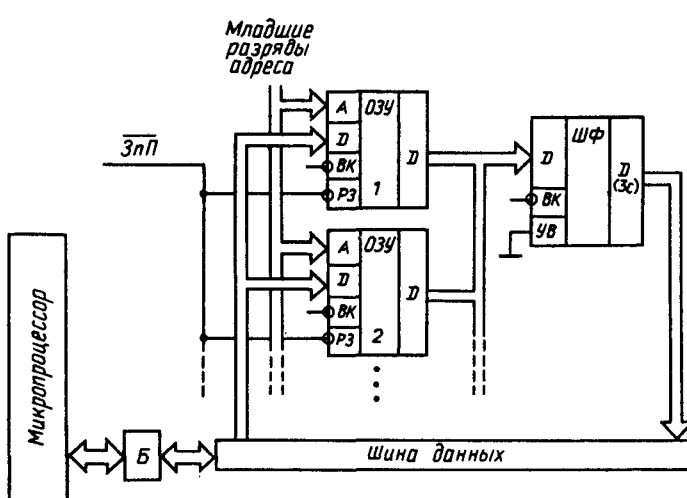


Рис. 3.38. Схема сопряжении ОЗУ с шиной данных

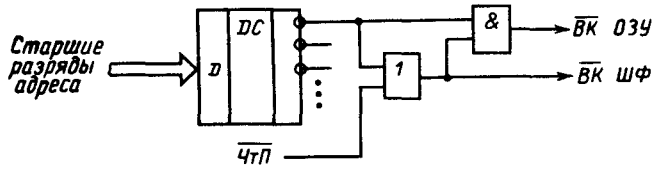


Рис. 3.39. Формирование сигналов $\overline{ВК}$ для управления микросхемами ОЗУ и буфером

ми, подаваемыми на управляющие входы $\overline{ВК}$ и $\overline{РЗ}$ (*Разрешение записи*: при $\overline{РЗ} = 0$ — запись, при $\overline{РЗ} = 1$ — чтение). Для формирования сигналов $\overline{ВК}$, необходимых для управления микросхемами ОЗУ и буфером, может быть использована схема, приведенная на рис. 3.39.

Принимаемая с адресной шины информация дешифрируется. Если происходит обращение к ОЗУ, то старшие разряды адреса определяют микросхемы ОЗУ, в ячейки которых производится обращение; младшие разряды в результате дешифрации в самих микросхемах определяют выбираемые в них ячейки. Дешифратор DC выдает низкий уровень лог. 0 на выходе, соответствующем выбираемым микросхемам. На выходе $\overline{ВК}$ ОЗУ сигнал будет иметь активный уровень лог. 0, если уровень лог. 0 возникает на соответствующем выходе DC. Если при этом подается сигнал чтения из памяти $\overline{ЧтП}$ уровня лог. 0, то $\overline{ВК}$ ШФ = = 0 и одновременно в выбранных микросхемах ОЗУ производится чтение, а буфер ШФ устанавливается в открытое состояние. В режиме записи подается сигнал разрешения записи в память $\overline{ЗпП} = 0$, при этом сигнал чтения $\overline{ЧтП} = 1$, следовательно, $\overline{ВК}$ ШФ = 1 и буфер оказывается в выключенном состоянии.

Если память строится с использованием микросхем ОЗУ, выходы которых имеют три состояния, то надобность в буфере отпадает. Входы и выходы данных в таких микросхемах могут быть объединеиы и подключены непосредственно к шине данных. В ряде микросхем ОЗУ такое объединеиие входов и выходов предусмотрено в самих микросхемах. В них предусматриваются общие выводы данных, которые при записи используются как входы, при чтении — как выходы данных. Формирование сигналов управления такими микросхемами ОЗУ может выполняться схемой, приведенной на рис. 3.40.

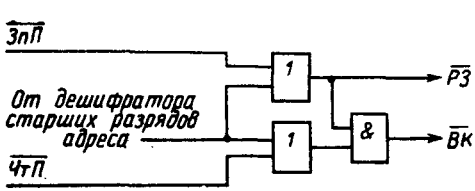


Рис. 3.40. Формнрование сигналов управления микросхемами ОЗУ, выходы которых имеют три состояния

Из изложенного видно, что для управления микросхемами ОЗУ требуются сигналы $\overline{ЗпП}$ и $\overline{ЧтП}$. Для управления устройствами ввода и вывода требуются соответст-

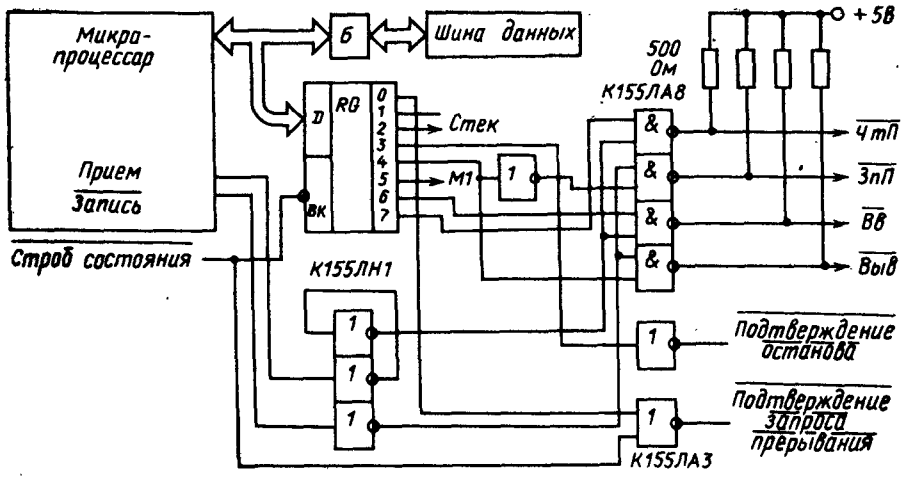


Рис. 3.41. Схема формирования управляющих сигналов

вующие сигналы: $\overline{ВВ}$ (*Ввод*) и $\overline{Выв}$ (*Вывод*). Эти сигналы являются управляющими сигналами микропроцессорных устройств.

Схема формированиия управляющих сигналов. На рис. 3.41 приведена схема, формирующая управляющие сигналы в микропроцессорном устройстве. В начале каждого цикла работы микропроцессор выдает одновременно сигнал *Синхронизация* на соответствующий выход и информацию о состоянии на шину данных. Информация о состоянии микропроцессора предназначена для формирования управляющих сигналов. Так как она выдается в течение коротких интервалов времени, а для формирования управляющих сигналов она необходима в течение длительности всего цикла, то информация о состоянии в каждом цикле стробом $\overline{Строб\ состояния}$ с шины данных принимается для хранения в регистр, называемый *фиксатором состояния*. Таким образом, содержимое фиксатора состояния в каждом цикле работы микропроцессора обновляется.

Отдельные разряды байта состояния дают информацию, указанную в табл. 3.3.

С помощью получаемой на выходе фиксатора состояния информации о состоянии микропроцессора и управляющих сигналов *Прием* и $\overline{Запись}$, выдаваемых микропроцессором, формируются управляющие сигналы

$$\overline{ЧтП} = \overline{D7 \wedge Прием};$$

$$\overline{ЗпП} = \overline{D4 \wedge Запись};$$

$$\overline{ВВ} = \overline{D6 \wedge Прием};$$

$$\overline{ВЫВ} = \overline{D4 \wedge Запись}$$

Системный контроллер КР580ВК28 (КР580ВК38). Показанная на рис. 3.41 схема формирования системных управляющих сигналов в основном предназначена для иллюстрации способа управления внешними устройствами, осуществляемого микропроцессором. При практическом выполнении микропроцессорного устройства необязательно использование этой схемы. Те же функции обеспечивает микросхема системного контроллера КР580ВК28 (КР580ВК38).

На рис. 3.42 показана структурная схема микросхемы системного контроллера. В микросхеме предусмотрен двунаправленный шинный формирователь, выполняющий функции двунаправленного буфера, включаемого между выводами шины данных (ШД) микропроцессора и ШД системы. Выдаваемая из микропроцессора в начале цикла информация о состоянии микропроцессора поступает на вход регистра состояния и при поступлении сигнала *Строб состояния* фиксируется в регистре, где она хранится до наступления следующего цикла (до момента поступления очередного сигнала *Строб состояния*). Контрольно-декодирующая матрица использует содержимое регистра состояния и управляющие сигналы с выхода микропроцессора *Прием*, *Запись*, *Подтверждение захвата*, формируя на выходах контроллера системные управляющие сигналы $\overline{\text{ЧтП}}$, $\overline{\text{ЗпП}}$, $\overline{\text{Вв}}$, $\overline{\text{Выв}}$, *Подтверждение прерывания*.

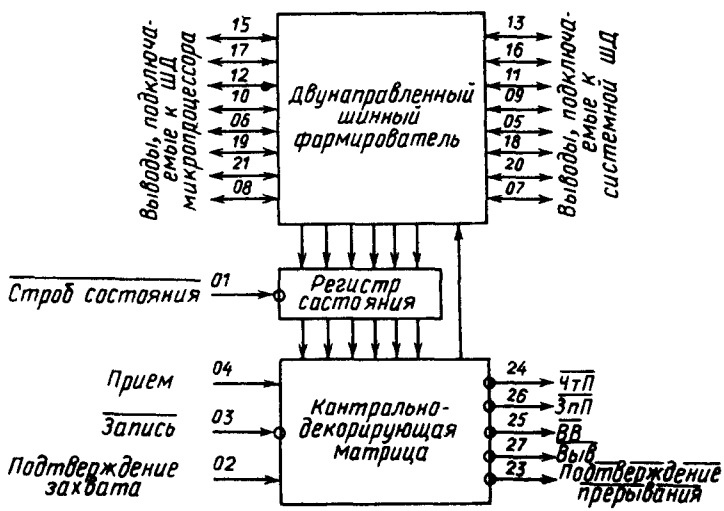


Рис. 3.42. Структурная схема микросхемы системиого контроллера КР580ВК28

Способы обмена данными между микропроцессорным устройством и периферийным устройством. В процессе функционирования МПУ возникает необходимость приема в него данных от различных устройств ввода. Принятые данные подвергаются обработке. Полученные в результате обработки данные выводятся из МПУ и передаются в различные устройства вывода. В качестве таких устройств ввода и вывода, называемых *периферийными устройствами* (ПУ), могут использоваться телетайпы, дисплеи, аналого-цифровые и цифро-аналоговые преобразователи информации, линии связи и т. п. Очевидно, для обеспечения такого обмена данными требуются определенные средства — система команд, сигналов и соответствующие устройства сопряжения. Эти средства объединяются под наименованием *интерфейс ввода-вывода*.

Рассмотрим способы обмена данными. Обмен данными между МПУ и ПУ может быть *программно управляемым* либо может осуществляться способом так называемого *прямого доступа к памяти* (ПДП). При программно-управляемом вводе микропроцессор в ходе выполнения соответствующей программы ввода побайтно принимает данные от ПУ через шину данных в аккумулятор. Прежде чем принимать очередной байт информации микропроцессор пересылает содержимое аккумулятора в ОП. Аналогично при выводе данных из МПУ в ПУ байты данных принимаются из ОП в аккумулятор, затем из аккумулятора они выдаются на шину данных, откуда принимаются в соответствующие ПУ.

Бóльшая скорость обмена данными между ОП и ПУ может быть обеспечена в режиме прямого доступа к памяти. В этом режиме микропроцессор отключается от шин адреса и данных (переходя в режим *Захвата*) и не принимает участия в процессе обмена. Обмен между ОП и ПУ осуществляется непосредственно.

Рассмотрим подробнее принципы программно-управляемой передачи данных.

Синхронная передача. Синхронная передача предполагает, что при каждом выполнении встречающихся в программе команд обмена *Ввод* и *Вывод* ПУ готово к выдаче на шину данных запрашиваемого микропроцессором байта или готово к приему с шины данных байта, выданного на эту шину микропроцессором.

На рис. 3.43 показана схема, иллюстрирующая данный способ обмена. Здесь сопряжение с ПУ обеспечивают буферные регистры *1* и *2* (регистр *1* обеспечивает связь МПУ с устройством вывода, регистр *2*— с устройством ввода). Дешифратор DC, получая с шины адреса номер ПУ, обеспечивает уровень лог. 1 на входе ВК регистра *1*; далее с приходом из шины управления сигнала вывода $\overline{\text{Выв}}$ регистр устанавливается в состояние, в котором поступающий с шины данных байт принимается в регистр; содержимое регистра постоянно передается к ПУ.

Если дешифратор выбирает регистр *2*, то при поступлении сигнала ввода — $\overline{Вв}$ выход регистра выводится из третьего (выключенного) состояния и его содержимое передается на шину данных; прием информации в регистр осуществляется от устройства ввода при подаче *Строба приема* на сход С.

Асинхронная передача. При асинхронной передаче, прежде чем производить обмен данными, микропроцессор выясняет готовность ПУ к такому обмену. Приведенная на рис. 3.44 схема алгоритма иллюстрирует этот процесс. Микропроцессор получает из ПУ информацию о состоянии; анализируя ее, он выясняет готовность ПУ к обмену; если ПУ не готово к обмену, то микропроцессор повторяет чтение состояния ПУ; если ПУ готово к обмену, то осуществляется передача данных между микропроцессором и ПУ.

На рис. 3.43 штриховой линией показан триггер состояния ПУ. Если в дешифратор с шины адреса поступает номер, присвоенный этому триггеру, то с приходом сигнала ввода $\overline{Вв}$ (поступающего из шины управления) управляемый вентиль, подключенный к выходу триггера, выводится из третьего (закрытого) состояния, содержимое триггера передается в цепь одного из разрядов шины данных и принимается в аккумулятор микропроцессора. Здесь принятая информация анализируется для выяснения факта готовности ПУ.

Передача данных с прерыванием программы. В рассмотренных случаях обмен данными инициировался микропроцессором. Встречаются задачи, в которых обмен должен осуществляться в произвольных

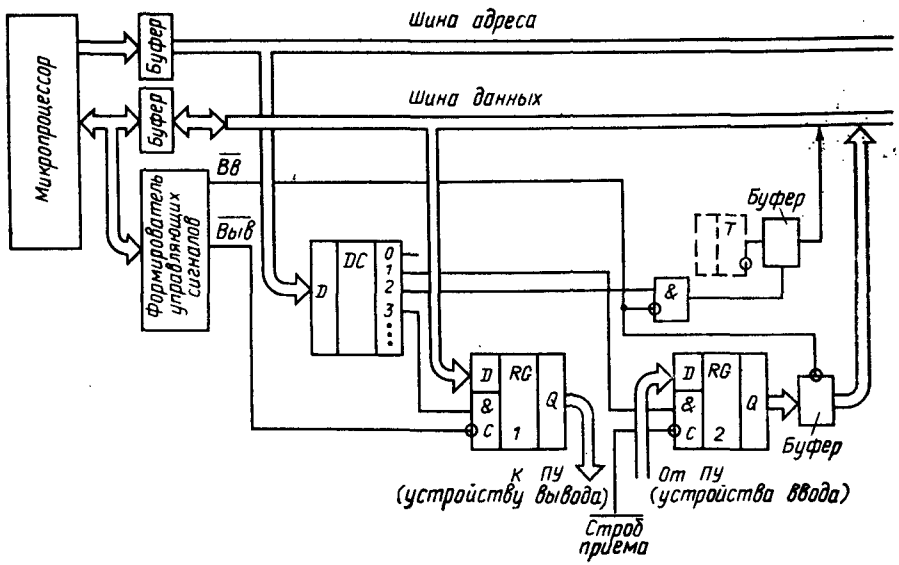


Рис. 3.43. Схема синхронного обмена между периферийным устройством и шиной данных МПУ

точках программы в моменты, определяемые периферийным устройством. При выполнении такого вида обмена данными по запросу, поступающему из ПУ, осуществляется прерывание выполняемой микропроцессором программы и переход к выполнению специальной программы обмена.

В МПК КР580 имеются микросхемы, предназначенные для построения интерфейса ПУ: КР580ВВ55 — программируемый параллельный интерфейс и КР580ВВ51 — программируемый последовательный интерфейс. Описание этих устройств приводится ниже.

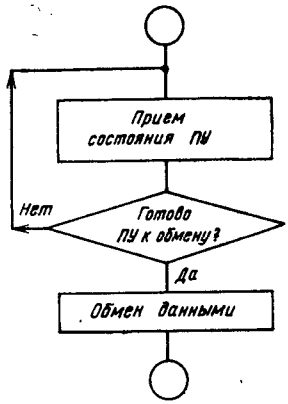


Рис. 3.44. Схема алгоритма асинхронного обмена между периферийным устройством и шиной данных МПУ

Программируемый параллельный интерфейс КР580ВВ55. На рис. 3.45 приведена упрощенная структурная схема программируемого параллельного интерфейса (ППИ).

С помощью ППИ осуществляется обмен данными (рис. 3.46) между микропроцессором (МП) и различными ПУ. Для подключения ППИ к шине данных (ШД) МПУ в ППИ предусмотрен 8-разрядный канал КД. Периферийные устройства могут подключаться к 8-разрядным каналам ППИ КА, КВ, КС. Канал КС состоит из двух 4-разрядных каналов КС1 и КС2. Каналы КА, КВ, КС снабжены регистрами. В канале КА предусмотрено два регистра, один из них используется для приема данных, поступающих из ШД МПУ, и выдачи их к ПУ, другой — для приема данных, поступающих от ПУ, и выдачи их на ШД МПУ. В каналах КВ, КС1 и КС2 имеется по одному регистру, который обеспечивает передачу данных между МП и ПУ в требуемом направлении. Все каналы снабжены буферными устройствами (входными и выходными формирователями с тремя состояниями), через которые осуществляется связь ППИ с внешними шинами.

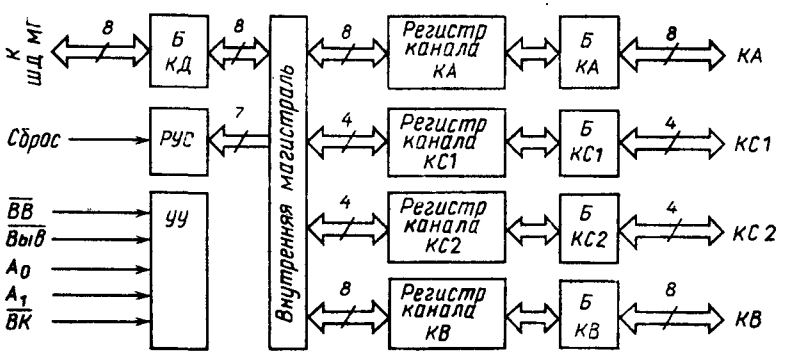


Рис. 3.45. Структура программируемого параллельного интерфейса КР580ВВ55

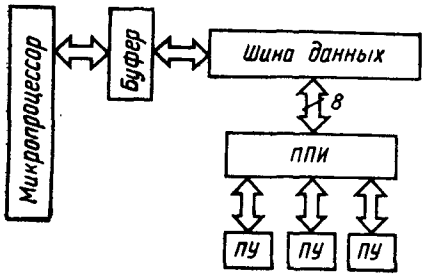


Рис. 3.46. Схема обмена данными между периферийными устройствами и шиной данных через ППИ

Таким образом, обмен между МП и ПУ распадается на две фазы обмена: обмен между регистром выбранного канала ППИ (каналов КА, КВ и КС) и ШД МПУ и обмен между регистрами каналов ППИ и ПУ. Рассмотрим, как организуется каждая из этих фаз обмена.

Обмен между ШД МПУ и регистром ППИ организуется под управлением сигналов, подаваемых на входы устройства управления (УУ) ППИ. A_0, A_1 — содержимого двух младших разрядов шины адреса (ША) МПУ, $\overline{ВК}$ — сигнала выборки микросхемы. В качестве последнего сигнала в системах с малым числом интерфейсных устройств может быть выбрано содержимое одного из шести старших разрядов шины адреса, в системах с большим числом интерфейсных устройств этот сигнал формируется дешифратором шести старших разрядов адреса. $\overline{Вв}$ и $\overline{Выв}$ — сигналы, формируемые в цепях управления МПУ. В табл. 3.21 показаны виды обмена данными между ШД МП и регистрами ППИ и соответствующие им наборы значений сигналов выборки.

По командам микропроцессора IN (ввод данных) и OUT (вывод данных) буферы канала КД обеспечивают обмен данными между ШД МПУ и внутренней магистралью данных ППИ. Принятая с ШД МПУ на внутреннюю магистраль данных ППИ информация либо представляет собой данные, которые через внутреннюю магистраль принимаются в регистр одного из каналов для дальнейшей их выдачи к ПУ, подключенному к этому каналу, либо представляет собой так называемое *управляющее слово*. Управляющее слово (УС) принимается в регистр управляющего слова (РУС) и организует обмен данными между регистрами каналов ППИ и ПУ.

С помощью УС производится установка ППИ в один из режимов работы (называемых *режимами 0, 1, 2*) для выполнения каналами определенных функций и задается направление передачи.

Таблица 3.21

Входы	Ввод			Вывод			
	КА→КД	КВ→КД	КС→КД	КД→КА	КД→КВ	КД→КС	КД→РУС
$\overline{ВК}$	0	0	0	0	0	0	0
$\overline{Вв}$	0	0	0	1	1	1	1
$\overline{Выв}$	1	1	1	0	0	0	0
A_1	0	0	1	0	0	1	1
A_0	0	1	0	0	1	0	1

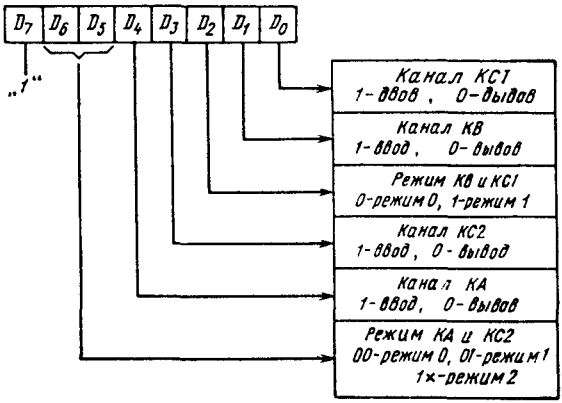


Рис. 3.47. Формат управляющего слова ППИ

На рис. 3.47 представлен формат управляющего слова. При поступлении из шины управления МПУ сигнала *Сброс* все канальные регистры сбрасываются в состояние лог. 0, а в РУС заносится информация, при которой все каналы устанавливаются на прием в режиме 0 (при этом выходные формирователи каналов оказываются в третьем — выключенном — состоянии).

Опишем функционирование каналов в отдельных режимах работы.

Р е ж и м 0. В этом режиме любой из каналов КА, КВ, КС1 и КС2 может быть установлен на ввод или вывод информации. При этом, если производится ввод информации, то регистр канала (в канале КА — входной регистр) непрерывно следит за всеми изменениями информации на входе канала; если осуществляется вывод информации, то содержимое регистра канала (в канале КА — выходного регистра) непрерывно передается на выход канала. Сигналы управления (*квитирования*) в этом режиме не формируются.

В табл. 3.22 показаны состояния каналов в зависимости от значения разрядов управляющего слова.

На рис. 3.48 показаны временные диаграммы для режима 0.

Р е ж и м 1. В этом режиме передача данных осуществляется через каналы КА и КВ, а канал КС используется в основном для приема и выдачи сигналов управления.

На рис. 3.49 показано функционирование канала КС, когда канал КА либо канал КВ установлен в режим 1.

Если канал КА установлен на ввод информации в режиме 1 (рис. 3.49, *а*), то одновременно с подачей на вход КА данных периферийное устройство подает в цепь $\overline{Строб\ приема\ КА}$ уровень лог. 0, сигнализируя о выдаче информации. ППИ выдачей сигнала *Подтверждение приема* сигнализирует о том, что в регистр канала приняты данные из периферийного устройства. При этом, если в разряде *Разрешение пре-*

Таблица 3.22

Значение разрядов УС				Состояние каналов			
D_4	D_3	D_1	D_0	КА	КС2 (4...7 разряды КС)	КВ	КС1 (0...3 разряды КС)
0	0	0	0	Вывод	Вывод	Вывод	Вывод
0	0	0	1	Вывод	Вывод	Вывод	Ввод
0	0	1	0	Вывод	Вывод	Ввод	Вывод
0	0	1	1	Вывод	Вывод	Ввод	Ввод
0	1	0	0	Вывод	Ввод	Вывод	Вывод
0	1	0	1	Вывод	Ввод	Вывод	Ввод
0	1	1	0	Вывод	Ввод	Ввод	Вывод
0	1	1	1	Вывод	Ввод	Ввод	Ввод
1	0	0	0	Ввод	Вывод	Вывод	Вывод
1	0	0	1	Ввод	Вывод	Вывод	Ввод
1	0	1	0	Ввод	Вывод	Ввод	Вывод
1	0	1	1	Ввод	Вывод	Ввод	Ввод
1	1	0	0	Ввод	Ввод	Вывод	Вывод
1	1	0	1	Ввод	Ввод	Вывод	Ввод
1	1	1	0	Ввод	Ввод	Ввод	Вывод
1	1	1	1	Ввод	Ввод	Ввод	Ввод

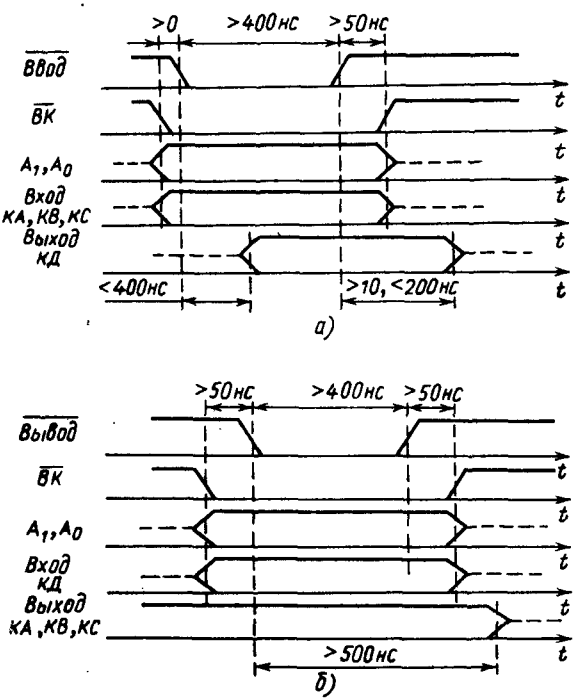


Рис. 3.48. Временные диаграммы:
а) ввод информации в режиме 0; *б)* вывод информации в режиме 0

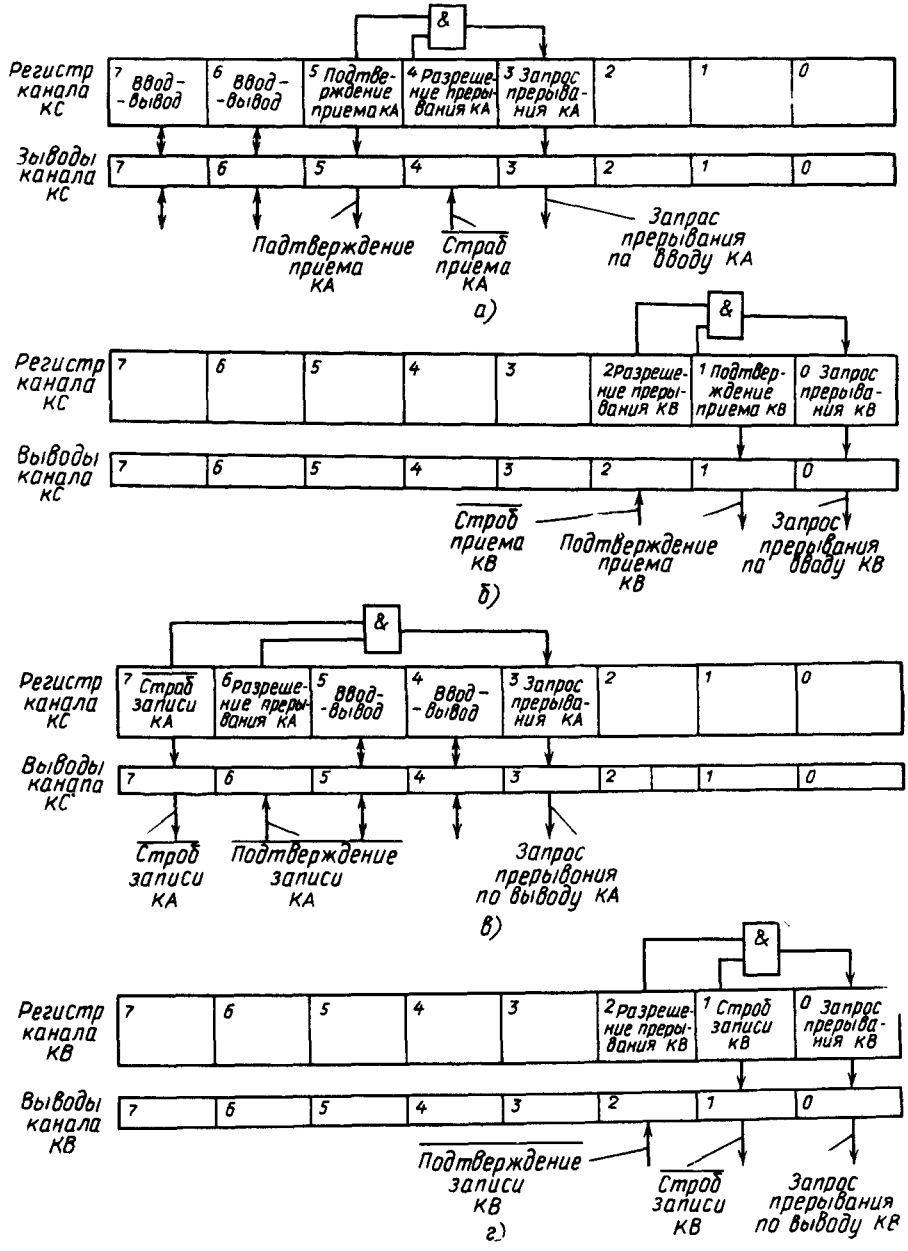


Рис. 3.49. Использование канала КС в режиме 1:

а) ввод информации по каналу КА; *б)* ввод информации по каналу КВ; *в)* вывод информации по каналу КА; *г)* вывод информации по каналу КВ

рывания по вводу КА регистра КС установлена лог. 1, то выдается сигнал *Запрос прерывания КА*.

На рис. 3.49, *б* показано функционирование канала КС при работе канала КВ на ввод информации в режиме 1.

При выводе данных (рис. 3.49, *в* и *г*) лог. 0 на выходе $\overline{Строб\ записи}$ сигнализирует о том, что МП произвел запись данных в регистр канала (КА или КВ); лог. 0 на входе $\overline{Подтверждение\ записи}$ сигнализирует о том, что ПУ приняло выдаваемую информацию.

Каналы КА и КВ путем записи в РУС соответствующего управляющего слова независимо друг от друга могут быть запрограммированы для работы на ввод или вывод данных в режиме 1. Вводом в МП содержимого регистра канала КС обеспечивается возможность проверки состояния каждого из подключенных к ППИ периферийных устройств и выбора в зависимости от него процесса выполнения программы.

Р е ж и м 2. В режимах 0 и 1 направление передачи между каналами ППИ и подключенными к ним периферийными устройствами задается управляющим словом, предварительно засылаемым из МП в ППИ. Следовательно, в указанных режимах всякое изменение направления передачи между ППИ и ПУ требует предварительной посылки в ППИ соответствующего управляющего слова. Особенность режима 2 состоит в том, что сигналами $\overline{Ввод}$ и $\overline{Вывод}$, посылаемыми в ППИ, не только устанавливается направление передачи между МП и ППИ, но эти сигналы задают также направление передачи между ППИ и подключенным к нему ПУ. Таким образом, обеспечивается возможность быстрого переключения направления передачи информации в целом между МП и ПУ без необходимости предварительной засылки управляющего слова в ППИ при каждом изменении направления обмена.

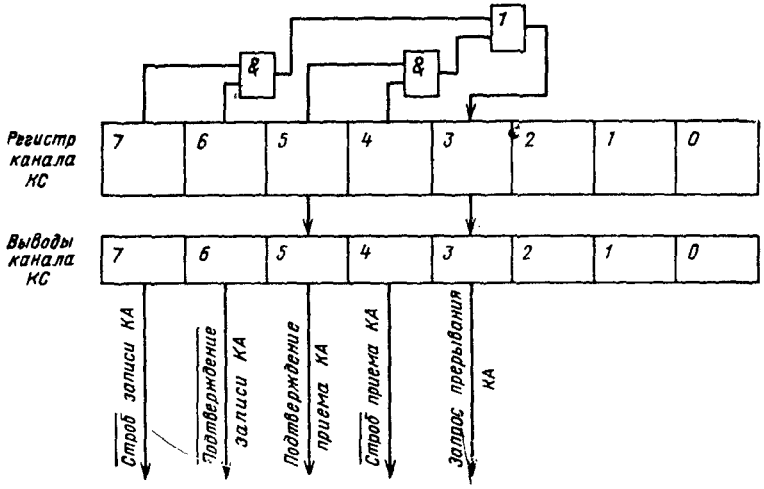


Рис. 3.50. Использование канала КС в режиме 2

В режиме 2 может работать только канал КА. Для передачи управляющих сигналов в этом режиме используются пять линий канала КС, как показано на рис. 3.50.

Запись информации в разряд регистра канала КС. В режимах 1 и 2 по соответствующим линиям канала КС выдается сигнал *Запрос прерывания*, который предназначен для подачи на соответствующий вход МП. Как видно из рис. 3.49 и 3.50, одним из условий, при которых формируется этот сигнал, является наличие лог. 1 в соответствующем разряде регистра канала КС. Устанавливая в этом разряде значение лог. 0 либо лог. 1, программист имеет возможность соответственно запрещать либо разрешать прерывание исполняемой программы для перехода на выполнение программы ПУ.

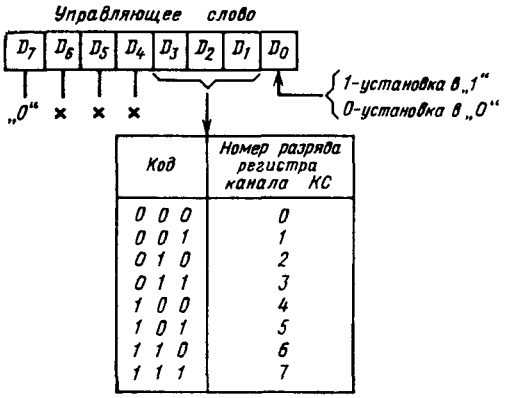


Рис. 3.51. Формат управляющего слова установки-сброса разряда регистра канала КС

Таким образом, возникает необходимость записи информации в отдельные разряды регистра канала КС. Для записи необходимо путем обмена вида ШД → РУС передать из МП в ППИ управляющее слово установки-сброса разряда регистра канала КС. Формат этого слова приведен на рис. 3.51. Его особенность в том, что в старшем разряде содержится лог.0. По этому признаку выявляется, что это слово не предназначено для помещения в РУС и служит для записи информации в разряд регистра канала КС. В управляющем слове записи указывается номер разряда регистра КС и значение, которое должно быть в этот разряд записано.

Программируемый последовательный интерфейс КР580ВВ51. Микросхема КР580ВВ51 представляет собой универсальное синхронно-асинхронное программируемое приемно-передающее устройство (УСАПП).

Микропроцессор через шину данных способен осуществлять обмен байтами данных в параллельной форме (одновременно всеми разрядами). Передача данных по линии связи может осуществляться в последовательной форме (разряд за разрядом). Для сопряжения ШД МП с линией связи может использоваться УСАПП. Это устройство преобразует снимаемые с ШД данные из параллельной формы в последовательную, пригодную для передачи их в линию связи; принимаемые из линии связи (в последовательной форме) данные преобразуются в параллельную форму, пригодную для выдачи на ШД МП.

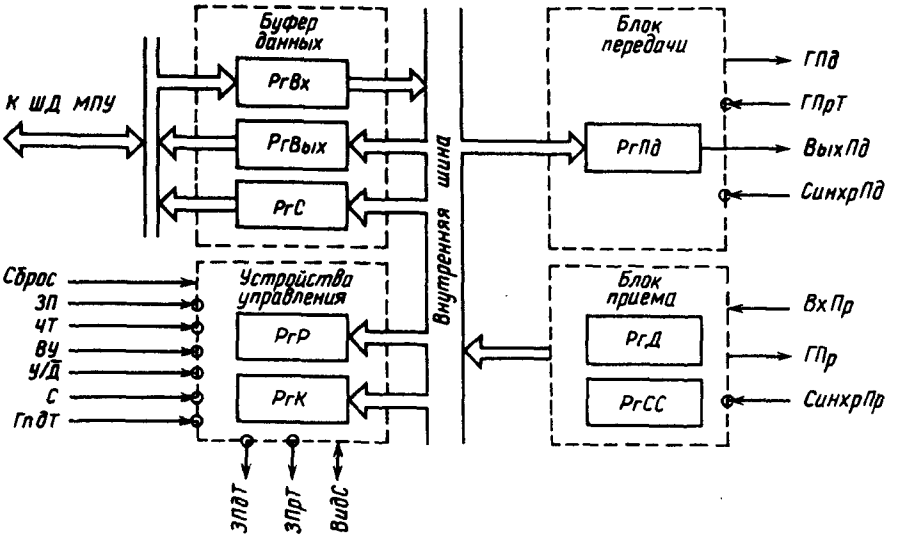


Рис. 3.52. Структурная схема УСАПП

На рис. 3.52 приведена укрупненная структурная схема УСАПП. Приведем краткое описание основных узлов устройства.

Принимаемый с ШД МП байт данных фиксируется во входном регистре РгВх, откуда он через внутреннюю шину передается в регистр передатчика РгПд. Путем серии сдвигов содержимое этого регистра выдвигается в последовательной форме на выход передатчика ВыхПд. При вводе данных в МП поступающие на вход приемника ВхПр в последовательной форме данные фиксируются в регистре РгД, откуда они через внутреннюю шину передаются в выходной регистр РгВых. Отсюда данные в параллельной форме выдаются на ШД МП. В синхронном режиме данные между УСАПП и ПУ (линией связи) сопровождаются одним либо двумя синхронизирующими словами. Для хранения кодовых комбинаций синхронизирующих слов в блоке приема предусмотрены регистры РгСС. Состояние устройства, формируемое в регистре состояния РгС в виде слова состояния, может быть запрошено в МП. Устройство управления содержит регистр режима РгР, предназначенный для хранения передаваемой из МП информации о режиме, в котором предусматривается обмен данными, и регистр команд РгК для хранения принимаемой из МП комаиды на обмен данными.

Подробное описание функционирования устройства и назначения сигиалов на его выходах будет дано ниже.

В табл. 3.23 показаны сигналы управления, определяющие направление передачи и вид передаваемой информации.

В качестве сигнала выборки устройства ВУ в системах с малым числом интерфейсиых устройств может быть выбрано содержимое од-

Таблица 3.23

Входные сигналы				Направление передачи	Вид передаваемой информации
У/$\overline{Д}$	ЧТ	ЗП	ВУ		
0	0	1	0	УСАПП→ШД	Данные
0	1	0	0	ШД→УСАПП	Даиные
1	0	1	0	УСАПП→ШД	Слово состояния
1	1	0	0	ШД→УСАПП	Управляющие слова

ного из старших разрядов шины адреса МПУ. В системах с большим числом интерфейсных устройств этот сигнал формируется путем дешифрирования адреса (исключая младший разряд адреса). Низкий уровень сигнала ВУ обеспечивает включение устройства в работу.

Сигнал У/$\overline{Д}$ определяет вид информации: при низком уровне лог. 0 передаваемая информация представляет собой данные; при высоком уровне лог. 1 передаваемая информация является словом состояния (служебной информацией, определяющей состояние УСАПП) либо представляет собой управляющие слова, передаваемые из МП для обеспечения в УСАПП требуемых функций. В качестве сигнала У/$\overline{Д}$ обычно используется содержимое младшего разряда A_0 шины адреса. Таким образом, при программировании обращения к УСАПП в младшем разряде его адреса указывается значение сигнала У/$\overline{Д}$.

Направление передачи (УСАПП → ШД либо ШД → УСАПП) определяет команда МП: при команде ввода (IN) формирователь системных управляющих сигналов выдает сигнал $\overline{Вв} = 0$, который поступает на вход $\overline{ЧТ}$ УСАПП и настраивает устройство на передачу в направлении УСАПП → ШД; при команде вывода (OUT) формируется сигнал $\overline{Выв} = 0$, который поступает на вход $\overline{ЗП}$ и настраивает устройство на передачу в направлении ШД → УСАПП.

Управляющими словами являются инструкция режима (формат инструкции представлен на рис. 3.53) и инструкция команды (формат команды приведен на рис. 3.54). Формат слова состояния приведен на рис. 3.55.

Сейчас не будем обсуждать форматы управляющих слов и слова состояния. Они будут рассмотрены при описании работы устройства в отдельных режимах.

А с и н х р о н н ы й в ы в о д . В схеме алгоритма на рис.5.56 представлены действия, которые должны быть предусмотрены в программе МП для обеспечения процессов асинхрониого вывода.

Подачей системного сигнала *Сброс* УСАПП устанавливается в исходиое состояиие. Далее командой МП OUT (*Вывод*) из МП передается в УСАПП ииструкция режима (так как по команде OUT МП выдает содержимое аккумулятора А, то, очевидио, предварительно в А должно быть сформировано соответствующее слово инструкции

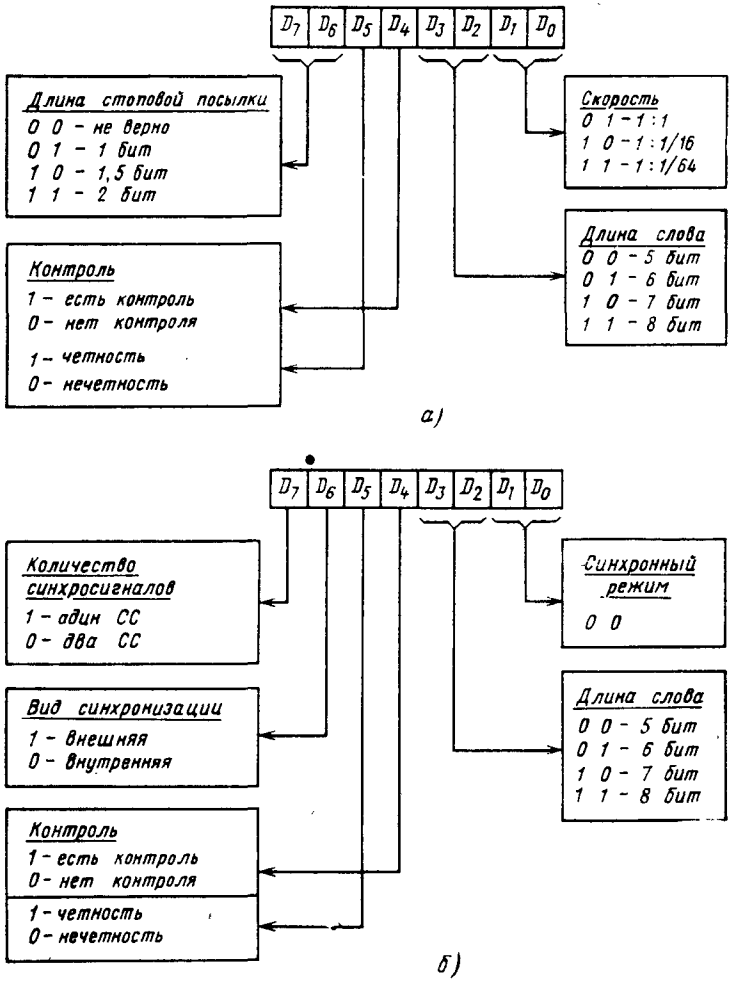


Рис. 3.53. Формат инструкцин режима:
а) для синхронного режима; *б)* для асинхронного режима

режима). В режиме асинхрониого вывода УСАПП к каждому выдаваемому байту данных подключает вначале стартовый сигнал уровня лог. 0 длиной в 1 бит и вслед за битами данных — бит контроля (если контроль предусматривается) и стоповый сигнал. Длина стопового сигнала может быть программно установлена равной 1; 1,5; 2 битам. Таким образом, даниые выдаются в следующем формате:

ВыхПд	Стартовый сигнал	Биты данных	Бит контроля	Стоповый сигнал

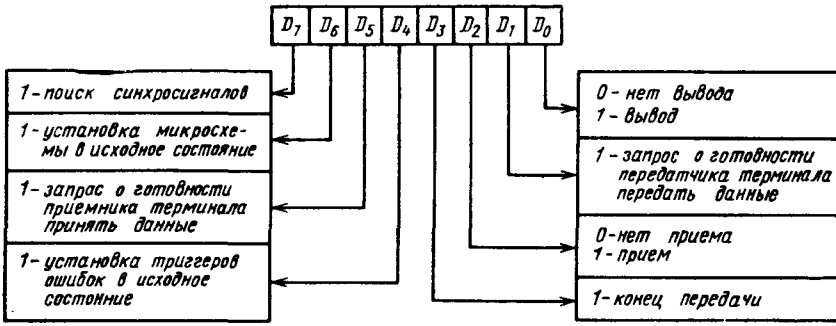


Рис. 3.54. Формат инструкции команды

Выдача этого слова на выход передатчика может осуществляться с тактовой частотой, равной 1/1; 1/16; 1/64 частоты сигнала синхронизации передатчика на входе *Синхр Пд.*

Эти сведения по формату выдаваемого из УСАПП слова и тактовой частоте выдачи его битов программно указываются в инструкции режима.

В соответствии с представленным на рис.3.53 форматом разряды D_1, D_0 в инструкции режима определяют синхронный (при комбинации значений 00) либо асинхронный режим; в последнем случае комбинации значений 01, 10, 11 задают тактовую частоту выдачи, соответственно равную 1/1, 1/16, 1/64 частоты сигнала синхронизации передатчика на входе *Синхр Пд.* Разряды D_3, D_2 определяют количество битов в выдаваемых данных (5...8), при этом свободные биты заполняются нулями. Разряд D_4 определяет, требуется ли контроль; если он предусматривается ($D_4 = 1$), то разряд D_5 определяет вид контроля (на четность либо нечетность количества единиц в битах даиных). Разряды D_7, D_6 задают длину стопового сигнала.

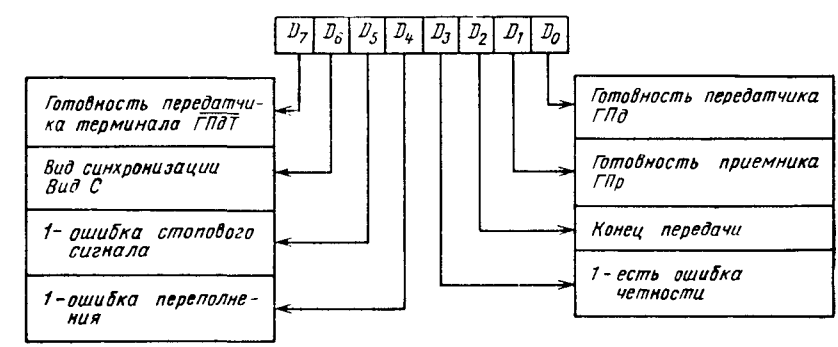


Рис. 3.55. Формат слова состояния

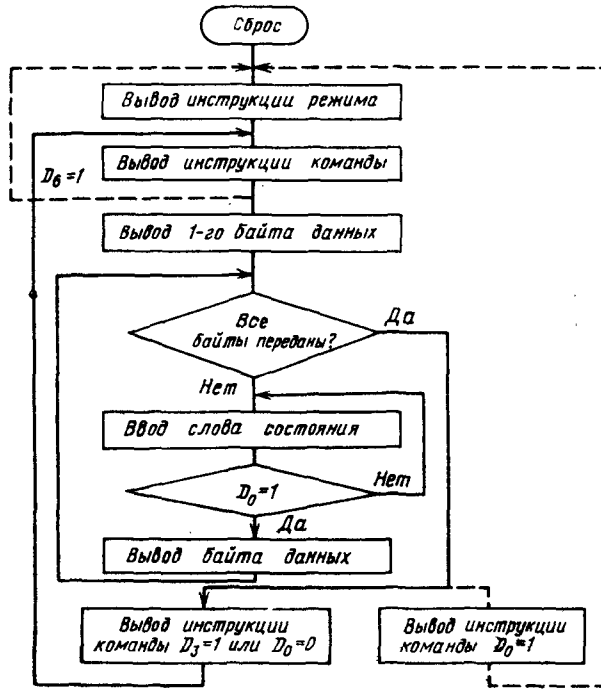


Рис. 3.56. Схема алгоритма асинхронного вывода

Пусть, например, тактовая частота выдачи битов должна быть равна частоте сигнала синхронизации передатчика ($D_1 = 0$, $D_0 = 1$); длина данных должна составлять байт ($D_3 = 1$, $D_2 = 1$); контроль не предусматривается ($D_5 = 0$, $D_4 = 0$); длина стопового сигнала равна 2 бита ($D_7 = 1$; $D_6 = 1$). При этих условиях инструкция режима будет иметь следующее значение:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	0	1	1	0	1

Вслед за выдачей в УСАПП инструкции режима в программе следует предусмотреть выдачу (команду МП OUT) в УСАПП инструкции команды (инструкция команды предварительно должна быть сформирована в аккумуляторе А). Так как предполагается вывод данных из УСАПП, то в команде $D_0 = 1$ (см. формат инструкции команды на рис.3.54). Не рассматривая смысла всех разрядов инструкции команды, представим ее в следующем виде:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	0	0	0	0	1

Значение $D_5 = 1$ означает выдачу устройством сигнала $\overline{\text{ЗПрТ}}$ (запрос готовности приемника терминала), т.е. сигнала запроса готовности приемника ПУ к приему данных, которые будут выдаваться УСАПП.

Далее командой OUT МП выдает в УСАПП содержимое аккумулятора А в качестве 1-го байта данных. Принятый в УСАПП байт данных совместно со стартовым сигналом, битом контроля и стоповым сигналом будет выдаваться на выход передатчика УСАПП при условии, если на вход $\overline{\text{ГПрТ}}$ поступит уровень лог.0, свидетельствующий о готовности приемника ПУ к приему данных.

Если необходимо выдавать из МП не один байт данных, а массив байтов, то прежде чем выдавать из МП каждый очередной байт, необходимо удостовериться, что выдача из УСАПП предыдущего байта завершена. Завершение выдачи УСАПП сигнализирует уровнем лог. 1 на своем выходе ГПд и значением $D_0 = 1$ в слове состояния (см. формат слова состояния на рис. 3.55). Следовательно, окончание процесса выдачи УСАПП предыдущего байта можно установить, циклически считывая из УСАПП слово состояния (по командам МП IN и установки в младшем разряде адреса $A_0 = У/\overline{Д} = 1$) и анализируя в МП значение младшего разряда принятого в аккумулятор слова состояния (например, путем сдвига вправо и анализа содержимого триггера переноса Тс в регистре признаков).

При готовности передатчика УСАПП в аккумулятор МП передается очередной подлежащей выдаче байт и командой OUT он выдается в УСАПП.

После выдачи последнего байта данных в аккумуляторе МП формируется инструкция команды, содержащая либо $D_3 = 1$, либо $D_0 = 0$ (если следующий обмен данными с ПУ не потребует изменения режима УСАПП), либо $D_6 = 1$ (если следующий обмен данными потребует иного режима УСАПП).

В рассматриваемом процессе информация об окончании выдачи передатчиком принятого из МП байта данных выбиралась из слова состояния. Как отмечалось выше, эта информация содержится и на выходе ГПд. Этот сигнал может быть использован в качестве сигнала запроса прерывания. При этом по сигналу ГПд = 1 МП переходит на выполнение прерывающей программы вывода данных через УСАПП.

Асинхронный ввод. Программа процесса асинхронного ввода данных в МП от ПУ через УСАПП представлена на рис. 3.57.

Пусть из МП в УСАПП передается инструкция режима того же содержания, что и в рассмотренном выше случае асинхронного вывода

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	0	1	1	0	1

Затем передается инструкция команды

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	1	1	0

В ней $D_1 = 1$ предусматривает выдачу из УСАПП в ПУ сигнала запроса готовности передатчика терминала $\overline{\text{ЗПдТ}}$; $D_2 = 1$ свидетельствует о том, что предстоит ввод данных ПУ в МП.

Затем в программе предусматривается циклический ввод слова состояния из УСАПП в МП и проверка в нем содержимого разряда D_1 (готовность приемника УСАПП), т.е. проверка завершения приема очередного слова из ПУ в УСАПП. При $D_1 = 1$ МП должен выполнить команду ввода (IN) принятого в УСАПП слова.

О готовности приемника УСАПП сообщает не только значением лог.1 в разряде слова состояния, но и сигналом уровня лог.1 на выходе ГПр. Этот сигнал может быть использован в качестве сигнала запроса прерывания, по которому МП можно перевести на выполнение прерывающей программы ввода данных из ПУ.

После ввода данных из УСАПП в МП снимается сигнал готовности с выхода ГПд и УСАПП принимает следующий байт из периферийного устройства. Прекращение ввода данных обеспечивается переда-

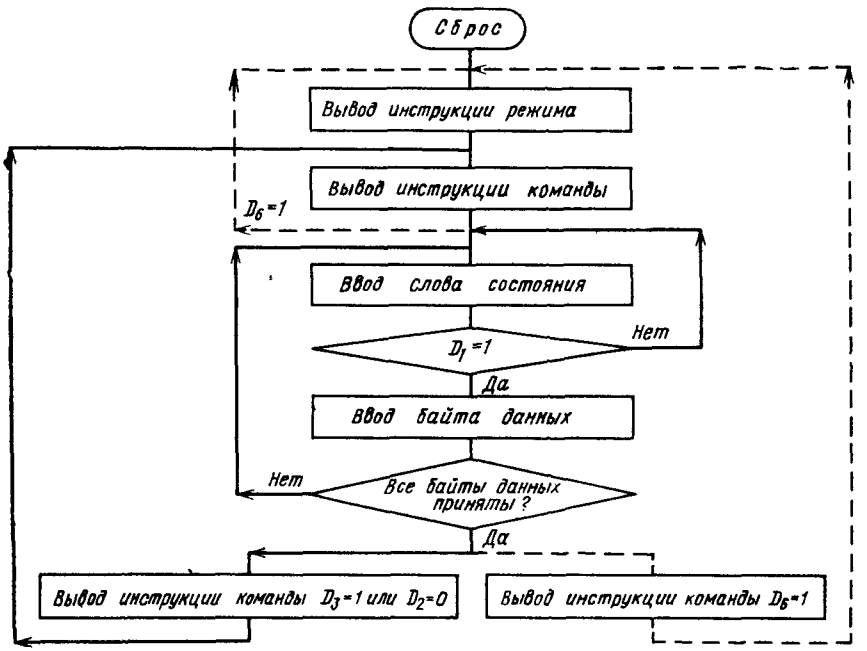


Рис. 3.57. Схема алгоритма асинхронного ввода

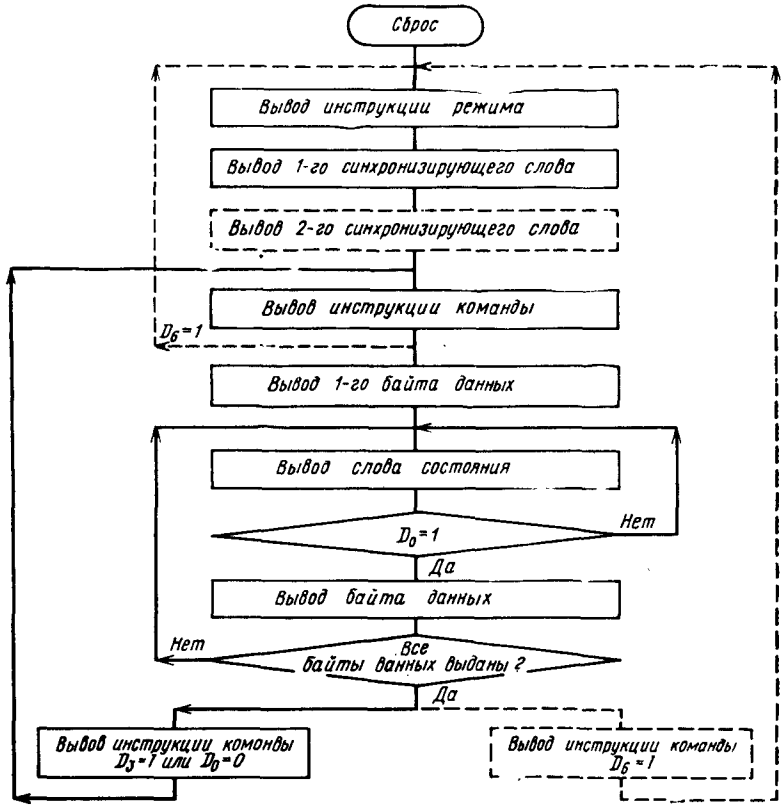


Рис. 3.58. Схема алгоритма синхронного вывода

чей из МП в УСАПП инструкции команды, содержащей $D_2 = 0$, или $D_3 = 1$, или $D_6 = 1$.

Синхронный вывод. На рис.3.58 представлена схема алгоритма синхронного вывода данных из МП через УСАПП в периферийное устройство.

В этом режиме данные из передатчика УСАПП выдаются в следующем виде

ВыхПд	Биты данных	Синхронизирующее слово 1	Синхронизирующее слово 2	Биты данных	Синхронизирующее слово 1	Синхронизирующее слово 2	Биты данных

Инструкция режима (см. рис. 3.53,*б*) в этом случае содержит в разрядах D_1, D_0 комбинацию значений 00, соответствующую синхронному режиму; значение разряда D_6 определяет вид синхронизации (внутренняя или внешняя), значение разряда D_7 определяет количество синхронизирующих слов.

Инструкция режима может иметь следующее значение:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	1	1	0	0

После вывода из МП в УСАПП инструкции режима в программе предусматривается вывод в УСАПП кодовой комбинации, выбранной для синхронизирующего слова (если в инструкции режима указано, что предусматривается использование двух синхронизирующих слов, то должна быть запрограммирована передача кодовых комбинаций обоих синхронизирующих слов).

Затем в программе МП предусматривается передача в УСАПП инструкции команды. Команда строится так же, как и при асинхронном выводе.

Далее командой вывода (OUT) из МП в УСАПП передается слово данных. При этом следует иметь в виду, что этому первому слову данных на выходе передатчика УСАПП не предшествуют синхронизирующие слова. Поэтому это слово данных не принимается в приемник ПУ и при программировании оно может быть задано произвольным. В остальном программа совпадает с программой асинхронного вывода.

Синхронный ввод с внутренней синхронизацией. В этом режиме информация, поступающая от ПУ на вход приемника УСАПП ВхПр, имеет следующий формат:

ВхПр	Синхронизирующее слово 1	Синхроинзирующее слово 2	Биты данных

Алгоритм синхронного ввода данных ПУ через УСАПП в МП представлен на рис. 3.59.

Программируется передача в УСАПП инструкции режима, которая может иметь то же значение, что и при синхронном выводе:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	1	1	0	0

Затем программируется передача в УСАПП кодовой комбинации синхронизирующих слов, после чего предусматривается передача инструкции команды. Инструкция команды может иметь такое же значение, что и при асинхронном вводе.

После этого в УСАПП происходят следующие процессы. Из УСАПП в ПУ через выход $\overline{\text{ЗПдТ}}$ уровнем лог.0 выдается сигнал запроса готовности передатчика терминала. Периферийное .устройство сигнализирует готовность к передаче подачей на вход УСАПП ГПдТ сигнала уровня лог.0, после чего передает на вход УСАПП ВхПр в последовательной форме синхронизирующие слова и слово данных.

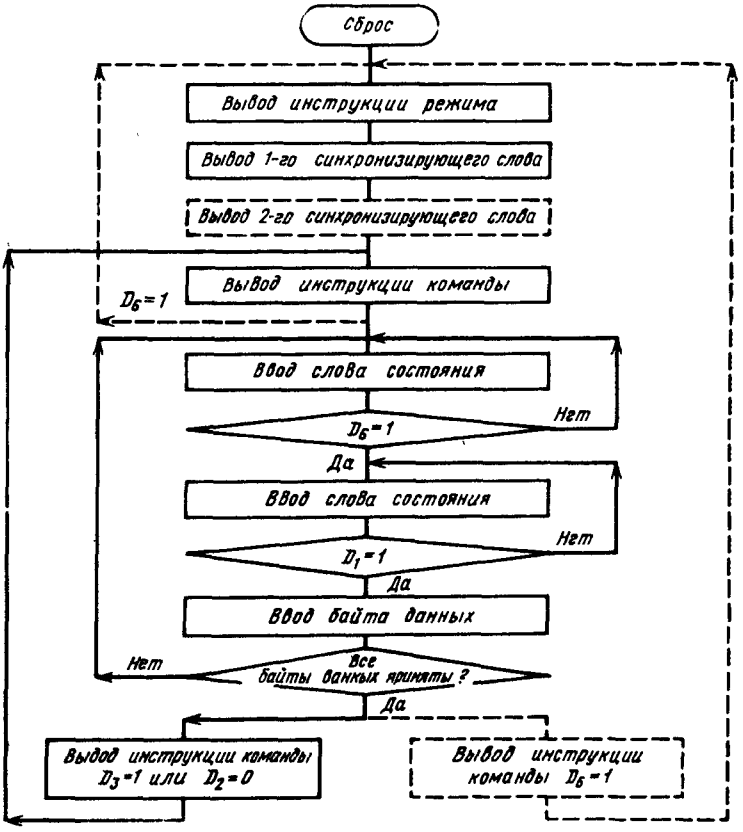


Рис. 3.59. Схема алгоритма синхроиого ввода

При совпадении синхронизирующих слов с их кодовой комбинацией (ранее переданной из МП в УСАПП) в регистр приемника УСАПП принимается слово данных, на выводе ВидС устанавливается выходное напряжение уровня лог.1 и записывается лог. 1 в разряд D_6 слова состояния. Сигнал готовности приемника ГПр = 1 и $D_1 = 1$ в слове состояния свидетельствуют об окончании приема слова данных и готовности данных для выдачи из УСАПП в МП.

Эти процессы протекают в УСАПП как реакция на поступление из МП инструкции команды.

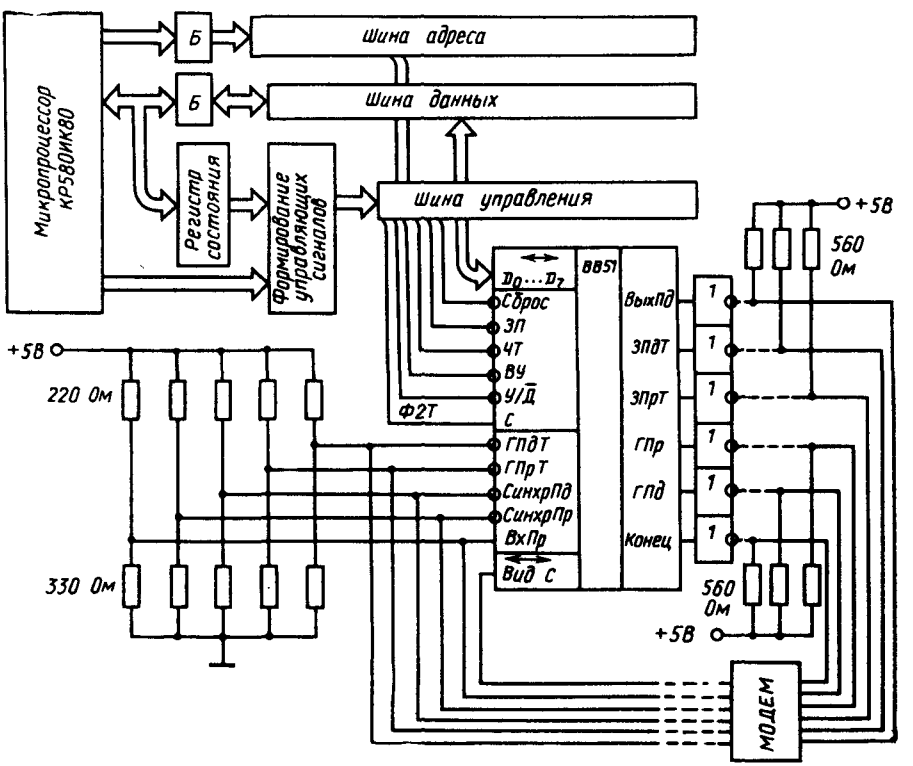
Вернемся к рассмотрению программы МП. После передачи в УСАПП инструкции команды предусматривается циклический ввод в МП слова состояния и проверка в нем содержимого разряда D_6. При $D_6 = 1$, продолжая циклический ввод слова состояния, МП проверяет в нем содержимое разряда D_1. При обнаружении $D_1 = 1$ выполняется команда МП на ввод данных (IN) и принятое в УСАПП слово

данных передается в аккумулятор МП. Дальнейшее в алгоритме аналогично схеме алгоритма асинхронного ввода.

Сигиал на выходе ВидС может быть использован в качестве сигнала запроса прерывания для перехода к выполнению прерывающей программы ввода.

Синхронный ввод с внешней синхронизацией. Для установки УСАПП в этот режим необходимо в передаваемой из МП в УСАПП инструкции режима предусмотреть $D_6 = 1$. В этом случае вывод ВидС является входом, на который из ПУ подают сигналы разрешения приема данных со входа ВхПр. При этом с тактовой частотой синхронизирующего сигнала на входе СинхрПр байты данных принимаются в приемник УСАПП. Процесс программирования для работы с УСАПП в данном режиме тот же, что и при использовании режима синхронного ввода с внутренней синхронизацией. Различие лишь в значении разряда D_6 инструкции режима, передаваемого в УСАПП.

Рассмотрим назначение разрядов D_3, D_4, D_5 слова состояния.



Рнс. 3.60. Использование УСАПП в качестве устройства сопряжения между мнкропроцессором и модемом

В разряде D_3 устанавливается значение лог.1 при обнаружении ошибки в принимаемых данных (четность или нечетность числа единиц в разрядах данных не соответствует значению бита контроля). Разряд D_4 устанавливается в состояние лог.1 при наличии так называемой ошибки переполнения, возникающей в следующих случаях: если МП не ввел подготовленный байт данных или ввод этого байта во времени совпал с передачей байта данных из приемника в буферы, через которые данные выводятся из УСАПП на ШД; если при синхронном вводе код данных совпал с кодовой комбинацией синхронизирующего слова.

В разряде D_5 устанавливается значение лог.1, если в режиме асинхронного ввода в конце посылки не обнаружены стоповые сигналы. Ошибки не влияют на работу УСАПП. Триггеры ошибок сбрасываются значением $D_4 = 1$ инструкции команды.

В заключение на рис. 3.60 приводим схему включения УСАПП в качестве устройства сопряжения между МП и модемом, работающим на линию связи.

4 МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА НА ОСНОВЕ МПК СЕРИИ КР1810

4.1. МИКРОПРОЦЕССОР КР1810ВМ86

Микропроцессорный комплект серии КР1810 представляет собой развитие МПК серии КР580. В комплект входят следующие микросхемы: КР1810ВМ86 — 16-разрядный микропроцессор, КР1810ГФ84 — тактовый генератор, КР1810ВГ86 — контроллер шин, КР1810ВН59А — контроллер прерываний, КР1810ВБ89 — арбитр шины.

По сравнению с микропроцессором КР580ИК80 микропроцессор КР1810ВМ86 отличается следующим:

при сохранении той же технологии *n* МДП достигнута более высокая степень интеграции (на кристалле размером 5,5 × 5,5 мм размещено около 30 тыс. транзисторных структур);

уменьшена задержка в логических элементах и тактовая частота повышена до 5 — 8 МГц; благодаря повышению тактовой частоты и совершенствованию структуры производительность микропроцессора повышена примерно на порядок;

расширена разрядность шины данных и тем самым обеспечена возможность выполнения операций обмена и обработки над 16-разрядными данными;

расширена разрядность адреса до 20 и, таким образом, обеспечена возможность адресации памяти емкостью до 1 Мбайта;

расширен набор команд.

СТРУКТУРНАЯ СХЕМА МИКРОПРОЦЕССОРА

На рис. 4.1 приведена структурная схема микропроцессора КР1810ВМ86. По общему функциональному назначению узлы схемы можно разбить на три части: операционное устройство, устройство сопряжения с шиной и управляющее устройство. В операционном устройстве выполняются команды, т.е. действия, связанные с обработкой данных; в устройстве сопряжения с шиной — действия, связанные с формированием адресов, по которым производится обращение в оперативную память, вызов команд и их хранение до начала выполнения; управляющее устройство формирует сигналы управ-

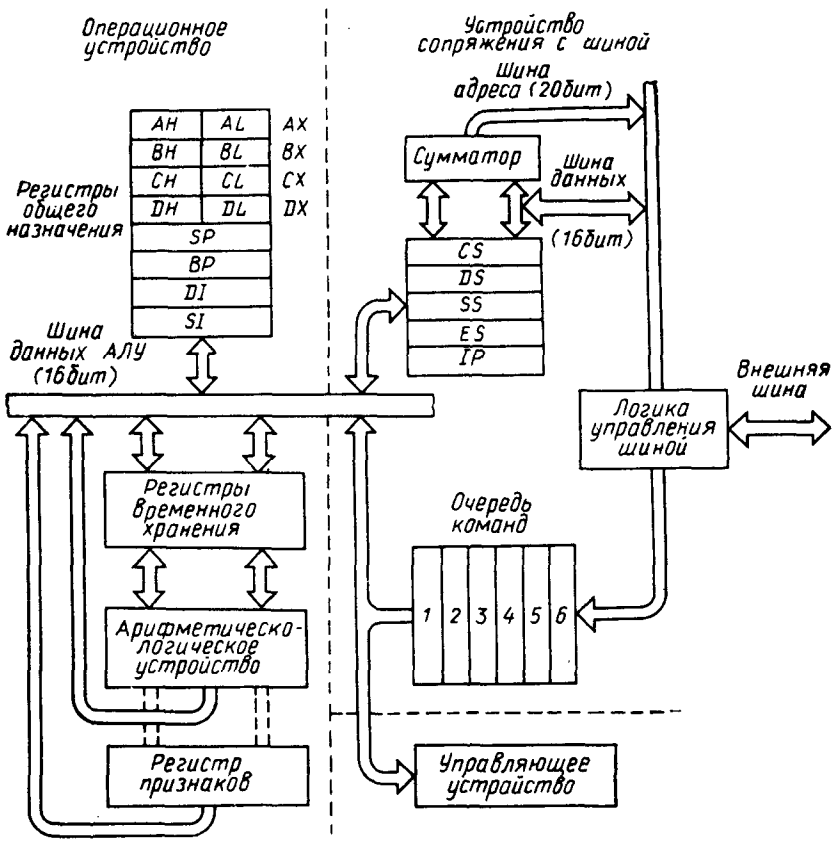


Рис. 4.1. Структурная схема микропроцессора КР1810ВМ86

ления функционированием операционного устройства и устройства сопряжения с шиной.

Регистры операционного устройства. Блок регистров общего назначения составлен из четырех 16-разрядных регистров A,B,C,D. Если эти регистры используются как 16-разрядные, то к их наименованию добавляется символ X (AX, BX, CX, DX); при их использовании для хранения 8-разрядных слов каждый регистр разбивается на два 8-разрядных, из которых наименование левого образуется добавлением к имени регистра символа H, наименование правого регистра —добавлением символа L (например, AH, AL и т.д.), как показано на рис.4.1. Эти регистры могут выполнять функции:

регистр A — функции аккумулятора,

регистр B — функции базового регистра, связанные с адресацией данных в оперативной памяти; они близки функциям, выполняемым в микропроцессоре серии КР580 парой регистров HL,

регистр C — функции счетчика,

регистр D — функции хранения данных.

Кроме указанных регистров, имеющих аналоги в микропроцессоре серии КР580, в микропроцессоре серии КР1810 предусмотрены еще четыре 16-разрядных регистра SP, BP, DI, SI. Из них лишь регистр SP (указатель стека) имеет аналог в микропроцессоре серии КР580. Эти регистры используются при формировании адресов; более подробно они будут рассмотрены далее. В табл.4.1 приведено назначение регистров операционного устройства.

Регистры устройства сопряжения с шиной. Из пяти 16-разрядных регистров, содержащихся в данном устройстве CS, DS, SS, ES, IP, лишь регистр IP имеет аналог в микропроцессоре серии КР580 (в микропроцессоре серии КР580 к нему близок по функциональному назначению регистр PC — счетчик команд). Остальные регистры предназначены для указания области (сегмента) памяти, в которой находится адресуемая ячейка памяти. Приведем наименование регистров, связанное с основным их назначением: CS — сегмент программы, DS — сегмент данных, SS — сегмент стека, ES — дополнительный сегмент, IP — указатель команд.

Таблица 4.1

Обозначение регистра	Назначение регистра
AX (AH, AL)	Аккумулятор
BX (BH, BL)	Базовый регистр
CX (CH, CL)	Счетчик
DX (DH, DL)	Регистр данных
SP	Указатель стека
BP	Базовый указатель
DI	Индексный регистр операнда
SI	Индексный регистр результата операции

Формирование адреса операнда. При регистровой адресации в команде указывается регистр, содержимое которого участвует в операции. При косвенной адресации возможны более сложные, чем в микропроцессоре серии КР580, приемы формирования адреса, показанные на рис.4.2.

В оперативной памяти могут выделяться области (сегменты), состоящие из 64К ячеек с последовательно нарастающими адресами. Для указания начальных адресов сегментов в зависимости от информации, для хранения которой предназначен сегмент (команды, стек, данные), используются регистры CS, SS, DS, ES в устройстве сопряжения с шиной. Программным путем регистры могут загружаться новой информацией. Таким образом, в памяти могут быть обозначены новые сегменты.

При определении адреса вначале формируется так называемый *исполнительный адрес*. Он может быть представлен содержимым регистров BX, BP, SI или DI. Исполнительный адрес операнда можно представить суммой содержимого указанного в команде регистра и представленного в непосредственной форме (в форме числа) *смещения*. Исполнительный адрес может формироваться и более сложным путем, обычно используемым при обработке последовательности знаков: к некоторой базе, хранящейся в базовом регистре BX или

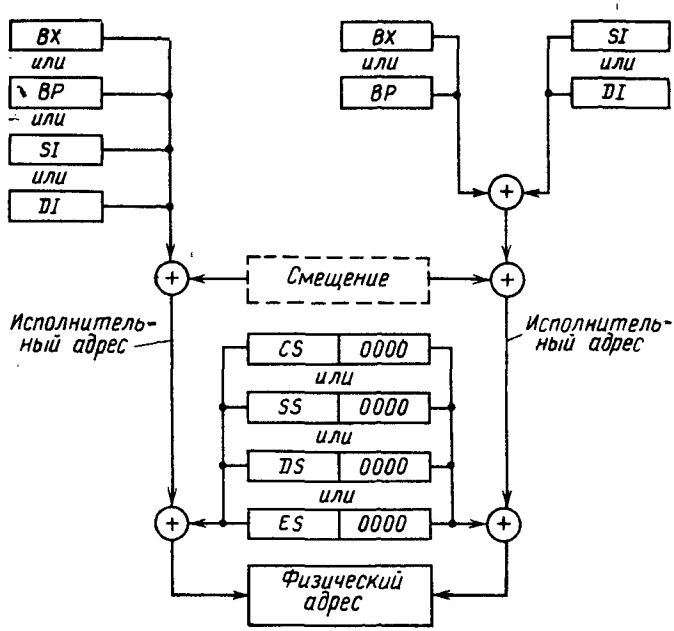


Рис. 4.2. Способы формирования физического адреса

BP, прибавляется содержимое индексного регистра DI или SI и смещение.

Исполнительный адрес используется для формирования *физического адреса* ячейки памяти, т.е. адреса, выдаваемого на шину адреса и поступающего в память. Предполагается, что ячейка находится в некоторой области (сегменте) памяти емкостью 64 Кбайт. В зависимости от характера хранимой в ячейке информации начальным адресом сегмента является содержимое регистра CS, SS, DS или ES, а положение ячейки в сегменте определяется исполнительным адресом. Начальным адресом сегмента служит содержимое регистров CS, SS, DS или ES, сдвинутое влево на 4 разряда. Суммирование начального адреса сегмента с 16-разрядным исполнительным адресом дает физический адрес в форме 20-разрядной кодовой комбинации, которая и выдается на адресную шину.

Регистр признаков. Регистр признаков имеет девять разрядов и хранит, таким образом, девять признаков, из которых пять аналогичны соответствующим признакам, хранимым в триггерах регистра признаков микропроцессора серии КР580: признаки CF, PF, AF, ZF, SF аналогичны признакам, формируемым в микропроцессоре серии КР580 соответственно в триггерах регистра признаков Tc, Tp, Tv, Tz, Ts. Дополнительно формируемые признаки: OF — признак переполнения разрядной сетки при выполнении арифметических операций и три признака управления: IF — разрешение прерывания, DF — признак направления, TF — признак захвата. На рис.4.3 показано назначение признаков. Признак DF используется лишь при

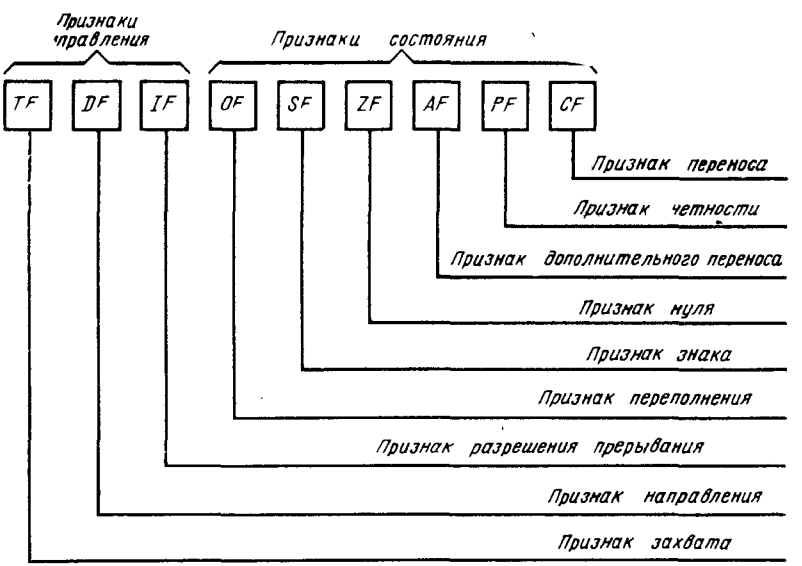


Рис. 4.3. Назначение признаков

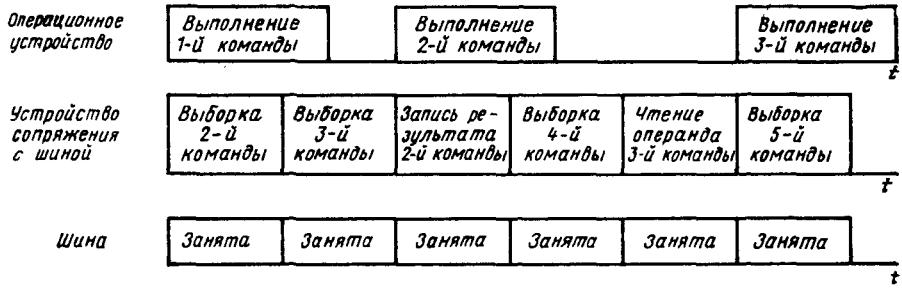


Рис. 4.4. Пример обработки последовательности команд

обработке последовательности знаков. При DF = 0 последовательность адресов обрабатываемых данных формируется в порядке их нарастания, а при DF = 1 — в порядке убывания адресов. Признак IF = 0 используется для игнорирования сигнала запроса прерывания, поступающего на вход маскируемого прерывания. Признак TF применяется в тех случаях, когда предусматривается покомандное выполнение программы. Когда TF = 1, процессор после выполнения каждой команды переходит в состояние особого прерывания (не связанного с сигналами запроса прерывания от периферийных устройств).

Блок регистров очереди команд. Блок регистров очереди команд обеспечивает возможность накопления команд объемом до 6 байт. Это позволяет подавать команды из блока регистров в операционное устройство, где происходит их исполнение, без задержки и, кроме того, достигается возможность совместить во времени процессы, связанные с выборкой команд из памяти, и процессы, связанные с их выполнением.

На рис.4.4 показан пример обработки последовательности команд в которой 1-я команда (ранее принятая в блок регистров очереди команд) предусматривает выполнение некоторой операции над даиными, хранящимися в регистрах микропроцессора, и запись результата в память; 2-я команда также выполняет операцию над хранящимися в процессоре данными, но в отличие от 1-й команды не предусматривает записи результата в память; 3-я команда требует извлечения операнда из оперативной памяти. В интервалах времени, свободных от записи или чтения данных из оперативной памяти, производится обращение к ней для выборки очередных команд, устанавливаемых в очередь в блоке регистров очереди команд. Таким образом, можно обеспечить высокую плотность загрузки шины и повышение скорости выполнения программы.

Очевидно, такое опережающее чтение из оперативной памяти команд до выполнения предыдущих команд возможно лишь в процессе исполнения последовательных участков программы, т. е. участков,

не содержащих условных и безусловных переходов. При таких переходах очередная команда должна поступать не из блока регистров очереди команд, а непосредственно из оперативной памяти.

4.2. СТРУКТУРА МИКРОПРОЦЕССОРНОГО УСТРОЙСТВА

На рис.4.5 показан простейший вариант построения микропроцессорного устройства на микропроцессоре КР1810ВМ86. Микросхема микропроцессора имеет 40 выводов. Из-за увеличенной разрядности шин, очевидно, в этом случае невозможно предусмотреть раздельные шины адреса и данных, как это сделано в микропроцессоре серии КР580. В микропроцессоре серии КР1810 сокращение числа выводов для шин достигнуто за счет использования совмещенных функций выводов: 16 выводов, обозначенные на рис.4.5 $АД_{15}...АД_0$, в разные временные интервалы используются либо как адресные выводы либо как выводы данных. Так как адрес имеет 20 разрядов, предусмотрено дополнительно 4 адресных вывода $A_{19}...A_{16}$, на которые выдаются старшие разряды адреса. С выводов $A_{19}...A_{16}$ и $АД_{15}...АД_0$ адрес через буфер $Б_1$ поступает в 20-разрядиую шину адреса системы. С этой шины адрес принимается в блок памяти и периферийные устройства ввода-вывода данных. Относится ли выдаваемый из микропроцессора на шину адреса адрес к блоку памяти или к периферийному устройству, определяется значением сигнала, выдаваемого микропроцессором на вывод П/$\overline{УВВ}$. При уровне лог.1

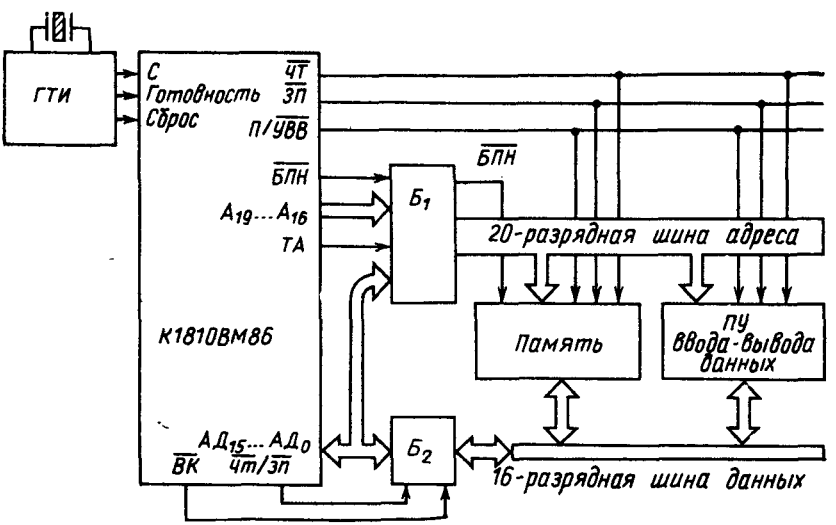


Рис. 4.5. Вариант построения МПУ на мнкропроцессоре КР1810ВМ86

на этом выводе инициируется работа блоков оперативной памяти, при уровне лог.0 — работа периферийных устройств. Вид обмена между микропроцессором и этими устройствами определяется сигналом на выводах $\overline{ЧТ}$ и $\overline{ЗП}$: при $\overline{ЧТ} = 0$ осуществляется чтение и прием данных через шину данных в микропроцессор, при $\overline{ЗП} = 0$ производится запись выводимой из микропроцессора на шину данных информации в блоки памяти или в периферийные устройства.

Операции чтения и записи занимают в микропроцессоре один цикл, состоящий из четырех тактовых интервалов. В тактовом интервале T_1 на выводы $A_{19}...A_{16}$ и $A_{15}...A_0$ микропроцессор выдает адресную информацию, которая должна быть принята в $Б_1$ и зафиксирована в нем на время, равное полной длительности цикла. Тактовый интервал T_2, как и в микропроцессоре серии КР580, используется для проверки наличия сигнала *Готовность*. Собственно обмен данными (чтение или запись) через шину данных осуществляется в тактовых интервалах T_3 и T_4.

Переключение буфера $Б_2$ на передачу по шине данных в требуемом направлении производится сигналом $ЧТ/\overline{ЗП}$.

Рассмотрим адресацию блоков памяти (рис.4.6). Память строится в виде двух блоков памяти, каждая емкостью 512К байт. Один из блоков связан со старшим байтом шины данных, другой — с младшим байтом этой шины.

Адресация блоков осуществляется разрядами $A_{19}...A_1$ адреса. В зависимости от значения сигнала $\overline{БПН}$ (*выборки блока памяти с нечетными адресами*) и значения младшего разряда адреса A_0 происходит инициирование того или другого либо обоих блоков в соответствии с табл.4.2.

При обмене двухбайтовой величиной (запись в память или чтение из памяти) адресом этой величины служит адрес ее младшего байта.

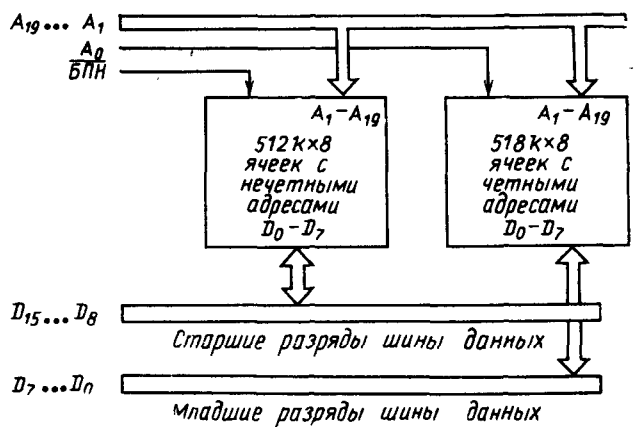


Рис. 4.6. Организация памяти с возможностью обмена двухбайтовыми данными

Таблица 4.2

$\overline{БПН}$	A_0	Пересылаемый байт
0	0	Оба байта
0	1	Верхний байт (D_8...D_{15}), нечетные адреса
1	0	Нижний байт (D_0...D_7), четные адреса
1	1	Ни тот ни другой

Если младший байт размещается в ячейке блока памяти с четными адресами, то при поступлении на блок четного адреса ($A_0 = 0$), и управляющего сигнала $\overline{БПН} = 0$ происходит обращение одновременно в оба блока. В блок с нечетными адресами (или из него) передается младший байт величины, а в блок с четными адресами (или из него) передается старший байт величины. Таким образом, передача двухбайтовой величины осуществляется при одном обращении к памяти. Если же младший байт передаваемой двухбайтовой величины размещается в ячейке блока памяти с нечетными адресами, то обмен такой величиной потребует двух обращений к памяти: при первом обращении подается нечетный адрес и сигнал $\overline{БПН} = 0$, происходит обращение в блок с нечетными адресами и осуществляется передача младшего байта, затем адрес увеличивается на единицу (при этом он принимает четное значение) и подается сигнал $\overline{БПН} = 1$, происходит обращение в блок с четными адресами и осуществляется передача старшего байта. Таким образом, при обращении к памяти требуется не только указывать адреса, но и сообщать сведения о том, является ли передаваемая величина однобайтовой или двухбайтовой.

Адресация устройств ввода-вывода (УВВ) осуществляется с использованием разрядов адреса A_{15}...A_0; УВВ, обмениваясь с микропроцессорным устройством 8-разрядными данными, этот обмен могут осуществлять, подключаясь либо к группе разрядов D_{15}...D_8 или к группе разрядов D_7...D_0 шины данных. В первом случае устройству присваивается нечетный адрес ($A_0 = 1$), во втором — четный адрес ($A_0 = 0$). Если УВВ обменивается 16-разрядными даниыми, то оно использует все 16 разрядов D_{15}...D_0 шины данных и имеет четный адрес.

В микропроцессоре серии КР1810 предусмотрены различные способы прерывания, показанные на рис. 4.7.

NMI и INTR — внешние сигналы, под действием которых в микропроцессоре происходят процессы прерываний. Обычно сигнал INTR формируется контроллером прерываний, который принимает сигналы запроса прерывания от многих источников. При установке в регистре признаков IF = 0 прерывания под действием сигнала INTR предотвращаются.

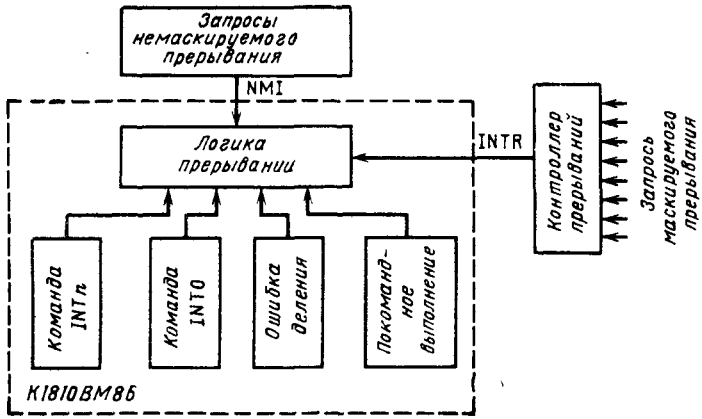


Рис. 4.7. Способы прерывания

Прерывания могут быть внутренними программными, возникающими при выполнении определенных команд:

команда INTO производит прерывание, если предыдущая арифметическая операция вызвала переполнение (в регистре признаков DF = 1),

ошибочный результат команд деления вызывает прерывание,

команда INT*n* (*n* = 0...255) вызывает прерывание,

если в регистре признаков TF = 1, то происходит прерывание после выполнения каждой команды (это используется при отладке программ).

Каждое прерывание имеет численное значение, определяющее адрес, с которого начинается выполнение программы после прерывания. Этот адрес представляется 8-разрядным числом (0...255), которое микропроцессор получает либо явно из команды (INT*n*), либо из контроллера прерываний, либо неявно.

4.3. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРА

ФОРМАТ КОМАНД

На рис.4.8 приведен общий формат команд микропроцессора. Первый байт команды содержит код операции и присутствует во всех командах, другие байты могут не быть в команде.

Младший разряд w байта кода операции в подавляющем большинстве команд определяет разрядность операндов. Если w = 0, то операнд однобайтовый; если w = 1, то операнд двухбайтовый.

Во втором байте команды поля mod и r/m определяют способы формирования исполнительного адреса в сегменте данных и стека,

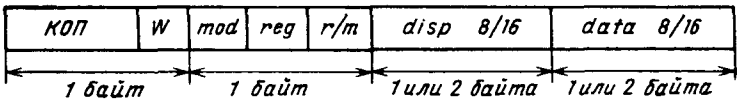


Рис. 4.8. Формат команд

показанные в табл.4.3. Здесь disp 8, disp 16 — приводимые в командах значения однобайтового и двухбайтового смещения. При disp 8 используемое при формировании исполнительного адреса смещение получается путем добавления к нему старшего байта, во всех разрядах которого устанавливается значение, совпадающее со знаковым (старшим) разрядом disp 8. Если в формируемом исполнительном адресе используется содержимое регистра BP, то адрес принадлежит (если это не оговаривается особо) сегменту стека, в остальных случаях адрес принадлежит сегменту данных.

При значении mod = 1 1 поля r/m и reg определяют адрес регистра общего назначения (РОН) в соответствии с табл.4.4.

Таблица 4.3

r/m	mod		
	0 0	0 1	1 0
0 0 0	BX + SI	BX + SI + disp 8	BX + SI + disp 16
0 0 1	BX + DI	BX + DI + disp 8	BX + DI + disp 16
0 1 0	BP + SI	BP + SI + disp 8	BP + SI + disp 16
0 1 1	BP + DI	BP + DI + disp 8	BP + DI + disp 16
1 0 0	SI	SI + disp 8	SI + disp 16
1 0 1	DI	DI + disp 8	DI + disp 16
1 1 0	disp 16	BP + disp 8	BP + disp 16
1 1 1	BX	BX + disp 8	BX + disp 16

Таблица 4.4

Поле r/m или reg	w=0	w=1	Поле r/m или reg	w=0	w=1
0 0 0	AL	AX	1 0 0	AH	SP
0 0 1	CL	CX	1 0 1	CH	BP
0 1 0	DL	DX	1 1 0	DH	SI
0 1 1	BL	BX	1 1 1	BH	DI

Таблица 4.5

Мнемоника	Операнды	Формируемые признаки	Размер операнда	Символическое обозначение операции	Содержание операции
			1. Общие пересылки		
MOV	dst, src	—	B, W	(dst)←(src)	Пересылка данных
MOV	dst, data	—	B, W	(dst)←(data)	
MOV	dst, sreg	—	W	(dst)←(sreg)	
MOV	sreg, src	—	W	(sreg)←(src)	
PUS H	src	—	W	(stack)←(src)	Запись в стек
PUS H	sreg	—	W	(stack)←(sreg)	
POP	dst	—	W	(dst)←(stack)	Обмен
POP	sreg	—	W	(sreg)←(stack)	Перекодировка
			2. Ввод-вывод		
IN	A, port	—	B, W	(AL или Ax)←(port)	Ввод
OUT	port, A	—	B, W	port←(AL или AX)	Вывод
			3. Пересылка содержимого регистра признаков		
LAHF		—	B	(AH)←(FLAGS)$_{мл.б}$	Загрузка младшего байта FLAGS в AH
SAHF		—	B	(FLAGS)$_{мл.б}$←(AH)	Загрузка AH в младший байт FLAGS
PUSHF		—	W	(stack)←(FLAGS)	Загрузка FLAGS в стек
POPF		—	W	(FLAGS)←(stack)	Загрузка FLAGS из стека
			4. Сложение		
ADD	dst, src	A, C, O, P, S, Z	B, W	(dst)←(dst)+(src)	Сложение без переноса
ADD	dst, data	A, C, O, P, S, Z	B, W	(dst)←(dst)+(data)	

ADC ADC	dst, src dst, data	A, C, O, P, S, Z A, C, O, P, S, Z	B, W B, W	(dst)←(dst)+ +(src)+(CF) (dst)←(dst)+ +(data)+(CF)	Сложение с переносом
INC	dst	A, O, P, S, Z	B, W	(dst)←(dst)+1	Приращение
AAA DAA		A, C A, C, P, S, Z	B B		Десятичная коррекция А при сложении (неупакованный формат) То же при упакованном формате

5. Вычитание

SUB SUB	dst, src dst, data	A, C, O, P, S, Z A, C, O, P, S, Z	B, W B, W	(dst)←(dst)—(src) (dst)←(dst)—(data)	Вычитание без заема
SBB SBB	dst, src dst, data	A, C, O, P, S, Z A, C, O, P, S, Z	B, W B, W	(dst)←(dst)—(src)— —(CF) (dst)←(dst)— —(data)—(CF)	Вычитание с заемом
DEC	dst	A, O, P, S, Z	B, W	(dst)←(dst)—1	Отрицательное приращение
NEG	dst	—	B W	(dst)←0FFH—(dst)+1 (dst)←0FFFFH— —(dst)+1	Получение дополнительного кода (с изменением знака)
CMP CMP	dst, src dst, data	A, C, O, P, S, Z A, C, O, P, S, Z	B, W B, W	(dst)—(src) (dst)—(data)	Сравнение

Продолжение табл. 4.5

Мнемоника	Операнды	Формируемые признаки	Размер операнда	Символическое обозначение операции	Содержание операции
AAS		A, C	B		Коррекция неупакованной десятичной цифры при вычитании
DAS		A, C, P, S, Z	B		Коррекция упакованного десятичного числа при вычитании
		6. Умножение			
MUL	src	(AH)≠0⇒ CF=1, OF=1	B	(AX)←(AL)·(src)	Умножение однобайтовых чисел без знака
MUL	src	(DX)≠0⇒ CF=1, OF=1	W	(DX, AX)←(AX)·(src)	Умножение двухбайтовых чисел без знака
IMUL	src	(AH)≠0 или OFFH⇒CF=1, OF=1	B	(AX)←(AL)·(src)	Умножение однобайтовых чисел со знаком
IMUL	src	(DX)≠0 или OFFFFH⇒CF=1, OF=1	W	(DX, AX)←(AX)·(src)	Умножение двухбайтовых чисел со знаком
AAM			B		Десятичная коррекция неупакованных цифр в AH и AL при умножении
		7. Деление			
DIV	src	—	B	(AL)←(AX)/(src) (AH)←MOD (AX, src)	Деление без знака двухбайтового AH на однобайтовый src

DIV	src	—	W	(AX)←(DX, AX)/(src) (DX)← MOD ((DX, AX), (src))	Деление без знака четырехбайтового DX, AX на двухбайтовый src
IDIV	src	—	B	(AL)←(AX)/(src) (AH)←MOD (AX, src)	Деление со знаком двухбайтового AX на однобайтовый src
IDIV	src	—	W	(AX)←(DX, AX)/(src) (DX)← MOD ((DX, AX), (src))	Деление со знаком четырехбайтового DX, AX на двухбайтовый src
AAD		P, S, Z	B		Коррекция AL до деления двух неупакованных десятичных цифр
CBW		—	B	(AL)<0⇒(AH)←0FFH (AL)≥0⇒(AH)←00H	Преобразование байта в слово
CWD		—	W	(AX)<0⇒(DX) 0FFFFH (AX)≥0⇒(DX) 0000H	Преобразование слова в двойное слово

8. Логические операции

NOT	dst	O=0, C=0 S, P, Z	B W	(dst)←0FFH−(dst) (dst)←0FFFFH−(dst)	Инверсия
AND AND	dst, src dst, data	O=0, C=0 S, P, Z	B, W B, W	(dst)←(dst)∧(src) (dst)←(dst)∧data	Логическая операция И
OR OR	dst, src dst, data	O=0, C=0 S, P, Z	B, W B, W	(dst)←(dst)∨(src) (dst)←(dst)∨data	Логическая операция ИЛИ

Продолжение табл. 4.5

Мнемоника	Операнды	Формируемые признаки	Размер операнда	Символическое обозначение операции	Содержание операции
XOR XOR	dst, src dst, data	O=0, C=0 S, P, Z	B, W B, W	(dst)←(dst)⊕(src) (dst)←(dst)⊕data	Логическая операция ИСКЛЮЧАЮЩЕЕ ИЛИ
TEST TEST	dst, src dst, data	O=0, C=0 S, P, Z	B, W B, W	(dst)∧(src) (dst)∧data	Логическая операция И без записи результата
		9. Сдвиг			
CHL/CAL	dst, cnt	C, P, S, Z	B, W		Логический/арифметнческий сдвиг влево на cnt разрядов
SHR	dst, cnt	C, P, S, Z	B, W		Логический сдвиг вправо на cnt разрядов
SAR	dst, snt	C, P, S, Z	B, W		Арифметический сдвиг вправо на cnt разрядов
ROL	dst, cnt	C, P, S, Z	B, W		Циклический сдвиг влево без переноса
ROR	dst, cnt	C, P, S, Z	B, W		Циклическнй сдвиг вправо без переноса
RCL	dst, cnt	C, P, S, Z	B, W		Циклический сдвиг влево через перенос
RCR	dst, cnt	C, P, S, Z	B, W		Циклический сдвиг вправо через перенос

10. Безусловный переход

CALL	addr	—	—	(IP)←(IP)+addr	Вызов подпрограммы по адресу addr
CALL	src	—	—	(IP)←(src)	Вызов подпрограммы по адресу в src
RET		—	—		Возврат из подпрограммы
JMP	addr	—	—	(IP)←(IP)+addr	Переход по адресу addr
JMP	src	—	—	(IP)←(src)	Переход по адресу в src

11. Условный переход по значению флажка

JC	addr 8	—	—	(IP)←(IP)+addr 8 если (CF)=1	Переход по переносу
JNC	addr 8	—	—	если (CF)=0	Переход по отсутствию переноса
JS	addr 8	—	—	если (SF)=1	Переход по отрицательному результату
JNS	addr 8	—	—	если (SF)=0	Переход по положительному результату
JE, JZ	addr 8	—	—	если (ZF)=1	Переход по нулевому результату
JNE, JNS	addr 8	—	—	если (ZF)=0	Переход по ненулевому результату

Окончание табл. 4.5

Мнемоника	Операнды	Формируемые признаки	Размер операнда	Символическое обозначение операции	Содержание операции
JO	addr 8	—	—	если (OF) = 1	Переход по наличию переполнения
JNO	addr 8	—	—	если (OF) = 0	Переход по отсутствию переполнения
JP, JPE	addr 8	—	—	если (PF) = 1	Перенос по четности
JNP, JPO	addr 8	—	—	если (PF) = 0	Перенос по нечетности
JBE, JNA	addr 8	—	—	если (CF) ∨ (ZF) = 1	Переход по меньше или равно
JA, JNBE	addr 8	—	—	если (CF) ∨ (ZF) = 0	Переход по больше
		12. Управление циклами			
LOOP	addr 8	—	—	(CX) ← (CX) − 1 (IP) ← (IP) + addr 8 если (CX) <> 0	Циклический переход по адресу
JCXZ	addr 8	—	—	если CX = 0	
		13. Установка значения признака			
CLC				CF ← 0	Установка CF в нуль
CMC				CF ← 1 − CF	Инвертирование признака CF
STC				CF ← 1	Установка CF в состояние 1
CLD				DF ← 0	Установка DF в нуль
STD				DF ← 1	Установка DF в состояние 1
CLI				IF ← 0	Установка IF в нуль
STI				IF ← 1	Установка IF в состояние 1
		14. Прочие команды			
HLT					Останов
WAIT					Перевод в состояние ожидания
NOP					Отсутствие операции

Из-за большой сложности программирования на языке кодовых комбинаций, этот способ практически не находит применения. Широко используется программирование на языке Ассемблера. Поэтому ниже приводится описание системы команд и программирование только на этом языке.

Каждое имя в языке Ассемблера имеет определенный тип. Наиболее часто используются следующие типы:

BYTE PTR — обозначает 16-разрядный адрес (смещение) в сегменте памяти для однобайтовой переменной,

WORD PTR — то же для двухбайтовой переменной,

DWORD PTR — то же для четырехбайтовой переменной,

NEAR PTR — то же для команды, на которую выполняется переход внутри сегмента,

FAR PTR — то же для межсегментного перехода

NUMBER — обозначает произвольное 16-разрядное число.

Преобразование типа данных производится с помощью следующих операторов:

оператор OFFSET преобразует адрес, представленный 16-разрядным смещением в сегменте, в число (PTR → NUMBER),

оператор SEG преобразует 16-разрядный базовый адрес сегмента в число (PTR → NUMBER).

В приводимом в табл.4.5 списке команд микропроцессора приняты следующие обозначения:

B — байт, W — слово, D — двойное слово,

dst и src — операнды (один из них должен быть регистром),

data — обозначение непосредственного операнда, который должен быть выражением типа NUMBER,

cnt — число разрядов, на которое производится сдвиг операнда dst; если в поле cnt записывается CL, число сдвигов равно содержимому регистра CL,

reg — любой регистр AX ... DI в операциях над словами или любой однобайтовый регистр в операциях над байтами,

sreg — любой сегментный регистр CS ... ES,

FLAGS — регистр признаков.

ПСЕВДОКОМАНДЫ АССЕМБЛЕРА

В табл.4.6 приведен список псевдокоманд, используемых при программировании для сообщения транслятору необходимой информации.

Псевдокоманда EQU используется для задания используемой в программе переменной некоторого значения. Эту псевдокоманду удобно использовать в тех случаях, когда предусматривается повторное выполнение программы с измененным значением некоторой константы, встречающейся во многих местах программы. Тогда в

Таблица 4.6

Имя псевдокоманды	Пример	Выполняемые действия
EQU	TIME EQU 5	В программе везде транслятор заменяет имя переменной TIME значением 5
DB	COR DB?	Резервируется память для однобайтовой переменной COR
DW	RM DW?	То же для двухбайтовой переменной RM
DD	MAS DD?	То же для четырехбайтовой переменной MAS
SEGMENT	CSEG SEGMENT	Определяется начало CSEG в сегменте
ENDS	CSEG ENDS	Конец в сегменте
ASSUME	ASSUME CS: CSEG	Установка содержимого сегментного регистра CS при выполнении программы
ORG	ORG 0200H	Заданне исполнительного адреса 1-й команды программы
PROC	TM PROC NEAR	Заголовок процедуры TM
ENDP	TM ENDP	Конец процедуры TM
END	END PR	Конец программы с именем 1-й команды PR

программе этой константе присваивается имя, а в псевдокоманде EQU имени присваивается значение. В процессе трансляции везде, где в программе встречается это имя, Ассемблер его заменяет значением, приведенным в псевдокоманде EQU. И если требуется изменить значение такой константы, встречающейся во многих местах программы, то достаточно изменить его значение в псевдокоманде EQU.

Псевдокоманды DB, DW, DD определяют память для переменных длиной соответственно 1,2,4 байта. Для обозначения переменной используется символ «?». Запись *n* DUP (?) означает массив из *n* переменных, а приведенное перед псевдокомандой в графе метки имя есть имя массива и одновременно имя его первого элемента. После метки псевдокоманды двоеточие не ставится (в отличие от меток команд).

СПОСОБЫ АДРЕСАЦИИ

По сравнению с микропроцессором серии КР580, микропроцессор серии КР1810 предоставляет более широкие возможности для адресации операндов. Ниже излагаются эти возможности представления на языке Ассемблера различных способов адресации операндов.

Регистровая адресация. При регистровой адресации адресом операнда служит содержимое регистра общего назначения.

Например, команда

ADD AX, BX

выполняет операцию (AX) ← (AX) + (BX).

В качестве регистров могут использоваться двухбайтовые регистры AX, ..., DI (при 16-разрядных операндах) или однобайтовые регистры AH, ...,DL (при 8-разрядных операндах).

Непосредственная адресация. При непосредственной адресации операнд представляется константой в самой команде.

Например, команда

```
ADD AX, 5
```

выполняет операцию (AX)←(AX) + 5.

Операнд 5 не требует обращения к памяти или содержимому регистра, он непосредственно представлен в команде. В качестве такого операнда может быть использовано любое выражение, которое в процессе трансляции приводит к числовой константе.

Прямая адресация. При прямой адресации в команде приводится исполнительный адрес участвующего в операции операнда.

Например, пусть ARRAY — начальный адрес массива однобайтовых данных. Тогда по команде

ADD AL, ARRAY + 7 будет выполнена операция (AL) ← ← (AL) + MEM (ARRAY + 7), по которой к содержимому регистра AL будет прибавлено содержимое ячейки памяти с адресом ARRAY + 7, т. е. в качестве второго операнда будет использован восьмой элемент массива ARRAY. Так как в программе массив ARRAY с помощью соответствующей псевдокоманды DB будет объявлен массивом однобайтовых переменных, здесь не требуется указаний о типе операнда (т.е. является ли операнд однобайтовым или двухбайтовым).

Косвенная регистровая адресация. При данном виде адресации операндом является содержимое ячейки памяти, адресом которой служит содержимое одного из регистров BX, SI или DI.

Например, команда

```
ADD AL, BYTE PTR [BX]
```

производит суммирование содержимого регистра AL и ячейки памяти, адресом которой служит содержимое регистра BX. Стоящий перед BX символ типа BYTE PTR указывает на то, что из памяти выбирается однобайтовая величина.

Адресация по базе или с индексированием. Этот вид адресации используется в тех случаях, когда операнд является одним из элементов массива данных и при обращении к такому операнду используется начальный адрес массива (адрес его первого элемента) и смещение относительно этого адреса. При этом начальный адрес (база) или смещение (индекс) представляются с использованием содержимого одного из базовых регистров BX, BP или одного из индексных регистров SI, DI. Исполнительный адрес операнда представляется выражением, в которое входит содержимое одного из четырех регистров. В процессе выполнения программы этот адрес может изменяться (вместе с содержимым соответствующего регистра).

Например, в команде

ADD AX, WORD PTR [BX + 7]

или

ADD AX, WORD PTR [BX]. 7

в качестве второго операнда используется содержимое ячейки памяти, адресом которой служит сумма содержимого регистра BX и константы 7. Здесь WORD PTR указывает, что из памяти по адресу (BX) + 7 выбирается двухбайтовое слово.

При адресации с индексированием возможна и следующая форма представления команды с использованием содержимого индексного регистра (SI или DI):

ADD AX, ARRAY + *n* [DI]

Здесь ARRAY — начальный адрес массива, *n* — числовая константа. Формируемый при этом исполнительный адрес равен ARRAY + + *n* + (DI). По этому адресу из памяти считываются данные той длины, какова длина, определенная соответствующей псевдокомандой для элементов массива ARRAY.

Адресация по базе с индексированием. При этом способе адресации исполнительный адрес определяется суммированием содержимого базового регистра (BX или BP), индексного регистра (SI или DI) и константы (присутствие константы необязательно).

Например,

ADD AL, BYTE PTR [BX]. 5 [DI]

или

ADD AL, BYTE PTR [BX + 5 + DI]

Данная команда в качестве операнда использует содержимое байта с адресом (BX) + 5 + (DI).

ПРИМЕРЫ ПРОГРАММ

Рассмотрим примеры составления программ с использованием различных способов адресации.

Пример 4.1. Требуется найти среднее арифметическое шестнадцати элементов массива W, хранимых в памяти в дополнительном коде, полученный результат RES поместить в память.

Программа решения задачи приведена в табл. 4.7.

Программа предусматривает помещение команд и данных в общий сегмент CS с базовым адресом SEG0 (0000H). Псевдокоманда ASSUME сообщает, что во время выполнения программы сегментный регистр CS будет содержать адрес SEG0. Далее следует псевдокоманда, сообщающая, что размещение команд в сегменте памяти производится, начиная с ячейки, имеющей смещенне 0200H относительно начала сегмента.

Таблица 4.7

```
SEG0    SEGMENT AT 0000H
        ASSUME CS: SEG0          ; Смещение программы в сегменте
        ORG 0200H                ; Первая команда
START:  MOV BX, OFFSET W         ; Установка исходного состояния в
        MOV AX, 0                ; AX
M1:     ADD AX, WORD PTR [BX]    ; Суммирование очередного элемента
        ADD BX, 2                ; Изменение адреса в BX
        CMP BX, OFFSET W+32      ; Проверка окончания суммирования
        JNZ M1                   ; Переход на M1, если есть еще эле-
                                 ; менты
        SAR AX, 4                ; Арифметический сдвиг вправо
        MOV RES, AX              ; Засылка результата в память
        DW 16 DUP (?)            ; Резервирование памятн для W
W       DW?                      ; Резервироваиие памяти для RES
RES     ENDS                     ; Конец в сегменте
SEG0    END START                ; Конец программы для траислятора
```

Командой, имеющей метку START, начинается собственно программа, определяющая вычисления.

Команда MOV BX, OFFSET W предусматривает пересылку начального адреса массива W в регистр BX. Представление этой команды в виде MOV BX, W было бы ошибочным, так как Ассемблер эту запись воспринял бы как пересылку первого элемента массива (хранящегося в ячейке памяти с адресом W) в регистр BX. Чтобы показать, что имеется в виду не элемент массива с адресом W, а сам адрес W, необходимо использовать оператор OFFSET, преобразующий W из адреса в данные. Таким образом, OFFSET W есть число, равное адресу W. Следовательно, эта команда, как и следующая MOV AX, 0 (обнуляющая регистр AX), построена с использованием непосредственной адресации.

Далее команда M1: ADD AX, WORD PTR [BX] предусматривает прибавление к содержимому регистра AX двухбайтового слова, взятого из памяти по адресу, хранящемуся в регистре BX. Здесь заключение в прямые скобки имени регистра BX свидетельствует о том, что содержимое регистра BX рассматривается в качестве адреса памяти; WORD PTR указывает, что по этому адресу выбирается двухбайтовое слово. Таким образом, в этой команде используется косвенная регистровая адресация.

После прибавления элемента массива к содержимому регистра AX в регистре BX формируется адрес очередного элемента массива (так как каждый элемент массива имеет двухбайтовое значение и занимает в памяти две ячейки, производится прибавление 2 к содержимому регистра BX).

Далее следует команда CMP BX, OFFSET W + 32.

Последний элемент массива имеет адрес, равный W + 30. Таким образом, данная команда производит сравнение содержимого регистра BX с адресом, следующим за адресом последнего элемента массива.

Далее команда INZ M1 осуществляет переход к команде M1, если результат сравнения, выполненного в предыдущей команде, не приводит к нулю.

Команда SAR AX, 4 производит арифметический сдвиг вправо содержимого регистра AX на 4 разряда, что обеспечивает деление на 16. Для деления можно использовать команду IDIV. Однако применение команды деления привело бы к 8-разрядному результату в регистре AL. Использование же операции сдвига обеспечило здесь получение результата в AX с бóльшим числом значащих разрядов (в данном случае получается 12 значащих разрядов).

Наконец, последняя в программе команда MOV RES, AX пересылает в память по адресу RES полученный в регистре AX результат вычислений.

Далее следует псевдокоманда определения W DW 16 DUP (?) с начальным адресом W для хранения 16 двухбайтовых данных.

Псевдокоманда RES DW? резервирует в памяти ячейку для хранения двухбайтовой переменной RES.

Псевдокоманда SEGO ENDS указывает, что больше никаких данных не предполагается помещать в сегмент памяти, а псевдокоманда END START информирует Ассемблер об окончании программы.

Пример 4.2. Составить программу нахождения наибольшего элемента в массиве ARR из 100 однобайтовых чисел без знака. Программа приведена в табл. 4.8.

Программа построена для поиска максимального элемента в массиве длиной NR, равной 100 (это значение присвоено NR псевдокомандой EQU). Далее в программе в качестве длины массива использовано имя NR. Следовательно, для того чтобы программа была пригодна для

Таблица 4.8

```
DATASEG     SEGMENT AT 0000H
NR          EQU 100
ARR         DB NR DUP (?)
MX          DB?
DATASEG     ENDS
CODESEG     SEGMENT AT 0100H
            ASSUME CS:CODESEG, DS: DATASEG
MAX:        MOV AL, 0              ; Установка AL в нуль
            MOV DI, 0              ; Установка DI в нуль
M1:         MOV BL, ARR [DI]       ; Прием в BL элемента массива
            CMP AL, BL             ; Сравнение содержимого AL и BL
            JNC M2                 ; Переход в M2, если (AL) > (BL)
            MOV AL, BL             ; Пересылка в AL большего элемента
M2:         INC DI                 ; Приращение содержимого DI
            CMP DI, NR             ; Проверка окончания элементов
            JNZ M1                 ; Переход к M1, если еще есть эле-
                                   ; меиты
            MOV MX, AL             ; Пересылка результата в память
GODESEG     ENDS
            END MAX
```

длины массива, отличного от 100, достаточно в ней изменить лишь оператор EQU, присваивая в нем NR другое значение.

В команде M1: MOV BL, ARR [DI] использована адресация с индексированием. Адрес элемента массива здесь определяется суммированием ARR + (DI). Этот элемент пересылается в регистр BL, чтобы затем в следующей команде его сравнить с содержимым регистра AL (хранящего наибольший из ранее просмотренных элементов массива).

Программа использует в памяти два сегмента: для хранения кодов команд — сегмент с базовым адресом CODESEG в регистре CS и для хранения данных — сегмент с базовым адресом DATASEG в регистре DS.

5. МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА НА ОСНОВЕ МПК СЕРИИ К589

5.1. СОСТАВ МИКРОПРОЦЕССОРНОГО КОМПЛЕКТА

Микропроцессорный комплект серии К589 предназначен для построения микропроцессорных устройств высокого быстродействия с тактовой частотой до 10 МГц.

В табл. 5.1 приведены некоторые данные микросхем, входящих в состав серии. Приведенное в графе разрядности центрального процессорного элемента (ЦПЭ) значение $2n$ означает, что микросхема имеет разрядность 2 бита, но допускает при построении операционного уст-

Таблица 5.1

Тип микросхемы	Назначение микросхемы	Разрядность, бит	Быстродействие, мкс	Потребляемая мощность, Вт	Количество выводов
К589ИК01	Блок микропрограммного управления	—	0,02	0,4	40
К589ИК02	Центральный процессорный элемент	$2n$	0,1	0,85	28
К589ИК03	Схема ускоренного переноса	8	0,02	0,4	28
К589ИР12	Многорежимный буферный регистр	8	0,08	0,45	24
К589АП16	Шинный формирователь	4	0,02	0,45	16
К589АП26	Шинный формирователь с инверсией	4	0,02	0,45	16
К589ИК14	Блок приоритетного прерывания	8	0,1	0,65	24
К589РЕ4	Микропрограммное ЗУ на (256×4) бит	4	0,04	0,5	16

ройства расширение разрядности обрабатываемых данных до $2n$ бит путем объединения n микросхем ЦПЭ.

Все микросхемы серии выполнены по технологии ТТЛШ, используют напряжение питания 5В±5 % и имеют допустимый диапазон рабочих температур —10 ... + 70° С. Они совместимы со всеми серийными интегральными микросхемами типа ТТЛ (серии К155,К555 и др.), т. е. не требуют устройств согласования при передаче выходных сигналов на входы микросхем типа ТТЛ либо при передаче выходных сигналов микросхем типа ТТЛ на входы микросхем микропроцессорного комплекта.

5.2. ПОСТРОЕНИЕ ОПЕРАЦИОННОГО УСТРОЙСТВА

ЦЕНТРАЛЬНЫЙ ПРОЦЕССОРНЫЙ ЭЛЕМЕНТ К589ИК02

Структурная схема. Центральный процессорный элемент предназначен для обработки двухразрядных данных. При построении операционного устройства с бо́льшим числом разрядов объединяется соответствующее число микросхем ЦПЭ.

Рассмотрим назначение отдельных узлов ЦПЭ (рис. 5.1).

Для хранения данных предусмотрены 12 регистров: аккумулятор АС, близкий к нему по функциональным возможностям регистр Т и 10 регистров общего назначения R_0 ... R_9. Блок регистров снабжен демультиплексором выборки входов и мультиплексором выборки выходов регистров. С помощью этих устройств по коду адреса в блоке регистров находится соответствующий регистр в зависимости от выполняемой операции используемый в качестве источника либо приемника операндов.

Кроме указанных 12 регистров предусмотрен регистр адреса (РА) для хранения двух разрядов адреса оперативной памяти.

В АЛУ выполняются операции над двухразрядными данными, для приема которых в АЛУ предусмотрены два входа. На каждый вход данные принимаются с выходов соответствующих мультиплексоров А и В. Мультиплексор А осуществляет подключение к первому входу АЛУ одного из трех источников данных: входа М, одного из регистров (выбранного мультиплексором блока регистров) либо аккумулятора АС. Мультиплексор В имеет также три входа, на которые принимаются двухразрядные данные со входа В, с выхода аккумулятора и входа К. Особенность работы мультиплексора В состоит в том, что в зависимости от выполняемой операции он может передавать на второй вход АЛУ поразрядную конъюнкцию данных, поступающих из АС и входа К, со входов В и К либо непосредственно данные со входа К. Такое построение мультиплексора В обеспечивает возможность выделения (так называемого *маскирования*) определенных разрядов данных, принимаемых со входа В либо из АС, путем установления соответствующего

кода на входе К либо возможность приема со входа К констант, участвующих в операциях АЛУ.

Результат выполненной в АЛУ операции принимается в АС, в РА либо в один из регистров R_0 ... R_9, Т блока регистров.

АЛУ имеет ряд дополнительных входов и выходов: вход переноса C_1 и выход переноса C_0; вход $СП_1$ и выход $СП_0$, используемые при операции сдвига вправо; выходы X и Y, используемые для ускоренной передачи переносов с помощью микросхемы ускоренного переноса К589ИК03. Выводы C_0 и $СП_0$ подключены к выходам буферов $ВБ_3$ и $ВБ_4$ с тремя состояниями, что позволяет соединять эти выводы в общую цепь (при выполнении операции сдвига вправо открывается буфер $ВБ_4$, а буфер $ВБ_3$ устанавливается в закрытое состояние, в котором он оказывается отключенным от внешней цепи, при всех других операциях открыт буфер $ВБ_3$ и в закрытое состояние устанавливается $ВБ_4$).

АЛУ построено как асинхронное устройство. Синхросигналы управляют лишь процессами, связанными с работой триггеров. В интервале

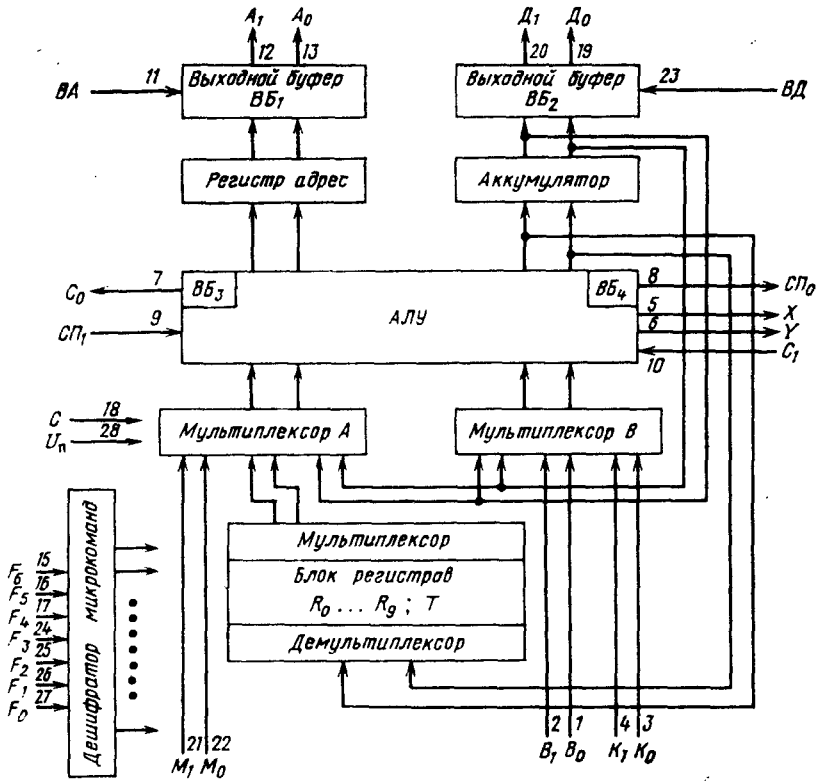


Рис. 5.1 Структурная схема ЦПЭ К589ИК02

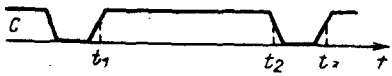


Рис 5.2. Синхросигнал

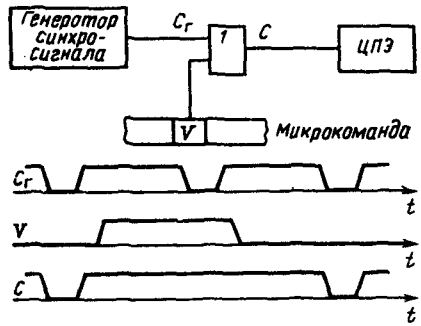


Рис. 5.3. Схема управления подачей синхроимпульсов в ЦПЭ

времени t_1 ... t_2 при высоком уровне синхросигнала (рис. 5.2) АЛУ выполняет соответствующую операцию. При этом входы триггеров регистров, являющихся приемниками результата выполненной операции, открыты. На отрицательном фронте синхросигнала происходит установка триггеров в состояния, соответствующие выдаваемому из АЛУ результату операцин, и входы триггеров логически отключаются на время t_2 ... t_3, когда синхросигнал имеет низкий уровень. При такой работе один и тот же регистр может служить как источником операнда, так и приемником результата выполненной операции.

Кроме того, если синхросигнал подавать в ЦПЭ через элемент ИЛИ, управляемый разрядом V микрокоманды (рис. 5.3), то при V = 1 будет подавлен спад сннхроснгнала, поступающего на вход ЦПЭ. При этом результат выполненной в АЛУ операции не будет приниматься в регистр.

Выходиые буферы $ВБ_1$ и $ВБ_2$ при наличии на входах ВА, ВД сигналов разрешення выдачи из третьего состояния переходят в открытое состояние, передавая на двухразрядные шины адреса А и данных Д содержимое регистров РА, АС.

Дешифратор мнкрокоманд преобразует подаваемый на входы F_6...F_0 семиразрядный код микрокоманды ЦПЭ в систему управляющих сигналов, под действием которых выполняется операция.

Следует иметь в виду следующую особенность: на всех выводах, кроме X, Y и F_6... F_0, информация представляется в обратном коде (т. е. лог. 1 представляется напряжением низкого уровня).

Система микрокоманд ЦПЭ. Мнкрокомаида ЦПЭ имеет следующий формат:

F_6...F_4	F_3...F_0	К	ВА	ВД

ВА, ВД — одноразрядные поля, которымн определяется значение сигнала на соответствующих входах микросхемы. Любое из этих полей либо оба поля в микрокоманде (МК) могут отсутствовать. Тогда на соответствующие отсутствующим полям входы микросхемы подается низкий уровень напряжения, при котором выходные буферы устанавливаются в открытое состояние.

Поле К в общем случае имеет то же число разрядов, что и данные процессора. В тех случаях, когда в реализуемой программе не возникает необходимости в выделении из данных некоторых их разрядов либо в передаче из МК в операционное устройство констант, все входы маскнрующей шины К объединяются в общую цепь. При этом поле К оказывается одноразрядным.

Поле $F_6 \ldots F_4$ образует F-группу, определяющую предусматриваемую МК операцию.

Поле $F_3 \ldots F_0$ задает регистр, который служит источником либо приемником, либо одновременно источником и приемником операнда.

Значение К = 0 во всех разрядах, за исключением операций, соответствующих значениям F-группы, равным 1 и 5, запрещает поступление операнда в АЛУ через мультнплексор В. В операциях, соответствующих значениям F-группы 1 и 5, через мультиплексор В на входы АЛУ передается значение К как некоторая константа.

В табл. 5.2 приведена кодировка F- и R-групп. Комбинации кода $F_3 \ldots F_0$ разбиты на три группы, названные первой, второй и третьей R-группами. В первой R-группе адресуются все 11 регистров блока регистров и аккумулятор АС; во второй и третьей R-группах адресуются лишь два регистра: Т и АС.

В табл. 5.3 приведен список выполняемых в ЦПЭ операций. Здесь обозначение АТ означает аккумулятор АС или регистр Т, в зависимости от кодовой комбинацин в поле $F_3 \ldots F_0$ микрокоманды; R_n — один из 12 регистров, определяемый значением поля $F_3 \ldots F_0$; AT_0, AT_1 — соответственно нулевой и первый разряды регистров АС либо Т.

Таблица 5.2

Группа функций (F-группа)	F_6	F_5	F_4	Группа регистра (R-группа)	Регистр	F_3	F_2	F_1	F_0
0	0	0	0		0	0	0	0	0
1	0	0	1		1	0	0	0	1
2	0	1	0		2	0	0	1	0
3	0	1	1		3	0	0	1	1
4	1	0	0		4	0	1	0	0
5	1	0	1	1	5	0	1	0	1
6	1	1	0		6	0	1	1	0
7	1	1	1		7	0	1	1	1
					8	1	0	0	0
					9	1	0	0	1
					T	1	1	0	0
					AC	1	1	0	1
				2	T	1	0	1	0
					AC	1	0	1	1
				3	T	1	1	1	0
					AC	1	1	1	1

Таблица 5.3

F-группа	R-группа	Произвольное значение K
0	1	$R_n+(AC\wedge K)+C_1\to R_n$, AC
	2	$M+(AC\wedge K)+C_1\to AT$
	3	$AT_0\wedge(B_0\wedge K_0)\to СП_0$ $СП_1\vee[(B_1\wedge K_1)\wedge AT_1]\to AT_1$
		$[AT_0\wedge(B_0\wedge K_0)]\vee[AT_1\vee(B_1\wedge K_1)]\to AT_0$
1	1	$K\vee R_n\to PA$ $R_n+K+C_1\to R_n$
	2	$K\vee M\to PA$ $M+K+C_1\to AT$
	3	$(\overline{AT}\vee K)+(AT\wedge K)+C_1\to AT$
2	1	$(AC\wedge K)-1+C_1\to R_n$
	2	$(AC\wedge K)-1+C_1\to AT$
	3	$(B\wedge K)-1+C_1\to AT$
3	1	$R_n+(AC\wedge K)+C_1\to R_n$
	2	$M+(AC\wedge K)+C_1\to AT$
	3	$AT+(B\wedge K)+C_1\to AT$
4	1	$C_1\vee(R_n\wedge AC\wedge K)\to C_0$ $R_n\wedge(AC\wedge K)\to R_n$
	2	$C_1\vee(M\wedge AC\wedge K)\to C_0$ $M\wedge(AC\wedge K)\to AT$
	3	$C_1\vee(AT\wedge B\wedge K)\to C_0$ $AT\wedge(B\wedge K)\to AT$
5	1	$C_1\vee(R_n\wedge K)\to C_0$ $K\wedge R_n\to R_n$
	2	$C_1\vee(M\wedge K)\to C_0$ $K\wedge M\to AT$
	3	$C_1\vee(AT\wedge K)\to C_0$ $K\wedge AT\to AT$
6	1	$C_1\vee(AC\wedge K)\to C_0$ $R_n\vee(AC\wedge K)\to R_n$
	2	$C_1\vee(AC\wedge K)\to C_0$ $M\vee(AC\wedge K)\to AT$
	3	$C_1\vee(B\wedge K)\to C_0$ $AT\vee(B\wedge K)\to AT$
7	1	$C_1\vee(R_n\wedge AC\wedge K)\to C_0$ $R_n\,\overline{\oplus}\,(AC\wedge K)\to R_n$
	2	$C_1\vee(M\wedge AC\wedge K)\to C_0$ $M\,\overline{\oplus}\,(AC\wedge K)\to AT$
	3	$C_1\vee(AT\wedge B\wedge K)\to C_0$ $AT\,\overline{\oplus}\,(B\wedge K)\to AT$

K=00	Мнемоника	K=11	Мнемоника
$R_n+C_1 \to R_n$, AC	ILR	$AC+R_n+C_1 \to R_n$, AC	ALR
$M+C_1 \to AT$	ACM	$M+AC+C_1 \to AT$	AMA
$AT_0 \to СП_0$ $AT_1 \to AT_0$	SRA	(см. общее описание)	
$СП_1 \to AT_1$			
$R_n \to PA$ $R_n+C_1 \to R_n$	LMI	$11 \to PA$ $R_n-1+C_1 \to R_n$	DSM
$M \to PA$ $M+C_1 \to AT$	LMM	$11 \to PA$ $M-1+C_1 \to AT$	LDM
$AT+C_1 \to AT$	CIA	$AT-1+C_1 \to AT$	DCA
$C_1-1 \to R_n$	CSR	$AC-1+C_1 \to R_n$	SDR
$C_1-1 \to AT$	CSA	$AC-1+C_1 \to AT$	SDA
см. CSA		$B-1+C_1 \to AT$	LDI
$R_n+C_1 \to R_n$	INR	$AC+R_n+C_1 \to R_n$	ADR
см. ACM		см. AMA	
$AT+C_1 \to AT$	INA	$B+AT+C_1 \to AT$	AIA
$C_1 \to C_0$ $0 \to R_n$	CLR	$C_1 \vee (R_n \vee AC) \to C_0$ $R_n \wedge AC \to R_n$	ANR
$C_1 \to C_0$ $0 \to AT$	CLA	$C_1 \vee (M \wedge AC) \to C_0$ $M \wedge AC \to AT$	ANM
см. CLA		$C_1 \vee (AT \wedge B) \to C_0$ $AT \wedge B \to AT$	ANI
см. CLR		$C_1 \vee R_n \to C_0$ $R_n \to R_n$	TZR
см. CLA		$C_1 \vee M \to C_0$ $M \to AT$	LTM
см. CLA		$C_1 \vee AT \to C_0$ $AT \to AT$	TZA
$C_1 \to C_0$ $R_n \to R_n$	NOP	$C_1 \vee AC \to C_0$ $R_n \vee AC \to R_n$	ORR
$C_1 \to C_0$ $M \to AT$	LMF	$C_1 \vee AC \to C_0$ $M \vee AC \to AT$	ORM
см. NOP		$C_1 \vee B \to C_0$ $B \vee AT \to AT$	ORI
$C_1 \to C_0$ $\overline{R}_n \to R_n$	CMR	$C_1 \vee (R_n \wedge AC) \to C_0$ $R_n \overline{\oplus} AC \to R_n$	XNR
$C_1 \to C_0$ $\overline{M} \to AT$	LCM	$C_1 \vee (M \wedge AC) \to C_0$ $M \overline{\oplus} AC \to AT$	XNM
$C_1 \to C_0$ $\overline{AT} \to AT$	CMA	$C_1 \vee (AT \wedge B) \to C_0$ $B \overline{\oplus} AT \to AT$	XNI

Поясним смысл некоторых из приведенных в табл. 5.3 записей. F-группа = 0, R-группа = 1:

при произвольном значении K пронзводится суммирование содержимого регистра R_n, поразрядной конъюнкции содержимого регистра AC и шины K, значения на входной цепи переноса C_1; результат заносится в регистры R_n и AC;

при K = 00 суммируется содержимое регистра R_n и значение на входной цепи переноса C_1; результат заносится в регистры R_n и AC;

при K = 11 суммируются содержимое регистров AC и R_n и значение на входе переноса C_1; результат заносится в регистры R_n и AC;

F-группа = 0, R-группа = 3, K = 00: выполняется операция сдвига вправо содержимого регистров AC или T. При выполнении этой операции содержимое младшего разряда регистра выдвигается на выход $СП_0$, в старший разряд регистра передается значение, поступающее на вывод $СП_1$.

Операция сдвига влево может быть выполнена удвоением содержимого AC. Для этого используется микрокоманда ALR либо ADR при задании в поле $F_3 \ldots F_0$ адреса AC в первой R-группе, т. е. $F_3 \ldots F_0 = 1101$. При выполнении этой операции содержимое старшего разряда выдвигается в цепь C_0, а в младший разряд AC заносится значение, поступающее на вход C_1.

Для установки в нуль всех разрядов регистра могут быть использованы МК CLR и CLA. Та же операция установки в нуль может быть выполнена с использованием МК CSR и CSA при подаче в цепь C_1 значения 1. Если в цепи C_1 действует значение лог. 0, то при выполнении МК CSR и CSA происходит запись значения 1 во все разряды адресуемого регистра.

Инвертирование содержимого регистра осуществляется с помощью МК CMR и CMA.

Прием данных с входных шин M или B в регистры AC и T может быть произведен с помощью МК LMF, LTM и LDI (при использовании МК LDI в цепи C_1 должно быть установлено значение 1).

Для выполнения логической операции конъюнкции (операции И) используются МК ANR, ANM, ANI. По МК ANR выполняется поразрядная конъюнкция содержимого адресуемого полем $F_3 \ldots F_0$ регистра R_n и аккумулятора AC; результат помещается в регистр R_n; дизъюнкция (операция ИЛИ) всех разрядов этого результата и значения на входе C_1 передается в цепь C_0 (последнее при $C_1 = 0$ может быть использовано для индикации нулевого значения результата). Микрокоманды ANM и ANI отличаются лишь источниками и приемниками операндов.

Для выполнения операции поразрядной дизъюнкции используются МК ORR, ORM, ORI; для выполнения поразрядной операции равнозначности (символ операции $\overline{\oplus}$) МК XNR, XNM, XNI.

В группе МК TZR, LTM, TZA предусмотрена выдача в цепь C_0 дизъюнкции соответственно R_n, M, AT и значения на входе C_1. Эта операция используется для проверки содержимого регистра или шины M на нуль (при этом на входе C_1 устанавливается значение 0).

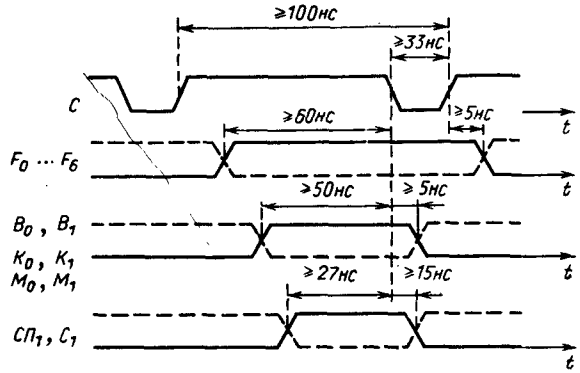


Рис. 5.4. Временные соотношения при работе ЦПЭ

Некоторые временные соотношения при работе ЦПЭ. На рис. 5.4 приведена временная диаграмма входных сигналов ЦПЭ и указаны требования к их длительности и временному положению. Диаграмма представляет минимально допустимые временные соотношения.

Время задержки распространения сигнала от входов F_6 ... F_0 до выходов X, Y, $СП_0$, C_0 составляет примерно 40 нс и от входа C_1 до выхода C_0 — 14 нс, время включения и отключения выходных буферов 12 нс.

СХЕМА УСКОРЕННОЙ ПЕРЕДАЧИ ПЕРЕНОСОВ К589ИК03

Принцип сквозной передачи переносов в СУП. На рис. 5.5 приведена схема объединения ЦПЭ для образования 2N-разрядного операционного устройства. В такой схеме объединения предусмотрена последовательная передача переносов, и так как задержка распространения переноса в каждом ЦПЭ составляет 14 нс, то общая задержка 14·N нс. Ускорение передачи переносов может быть обеспечено применением микросхемы ускоренной передачи переносов (СУП). Для осуществления сквозной передачи переносов в ЦПЭ предусмотрены выводы X и Y. На этих выводах формируются следующие сигналы:

Y определяет возникновение в микросхеме ЦПЭ переноса без учета поступления переноса на вход C_1;

X определяет условие, при котором перенос на выходе C_0 микросхемы ЦПЭ возникает только при наличии переноса на входе C_1 этой микросхемы.

Формирование значений X и Y во всех объединенных ЦПЭ происходит параллельно во времени, таким образом, затрачиваемое на это время мало. Блок СУП должен, анализируя значение переноса C_1 на входе операционного устройства (т. е. на входе C_1 ЦПЭ, находящегося в младшей позиции) и состояние X и Y на выходах всех секций операционного устройства (всех микросхем ЦПЭ), сформировать значение переноса одиовременно на входах всех секций (рис. 5.6).

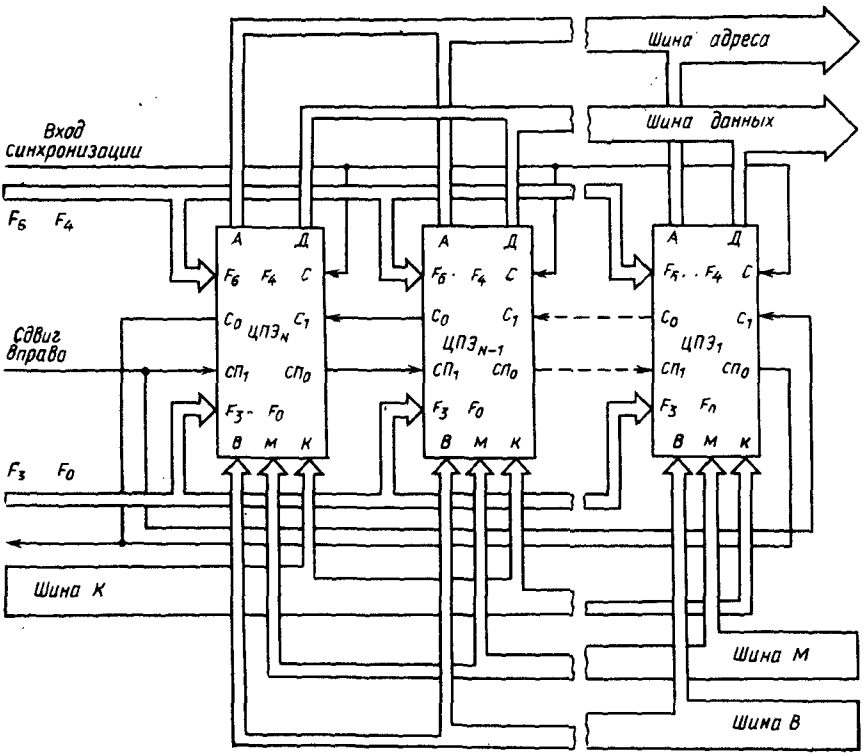


Рис. 5.5. Схема объединения ЦПЭ с последовательной передачей переносов

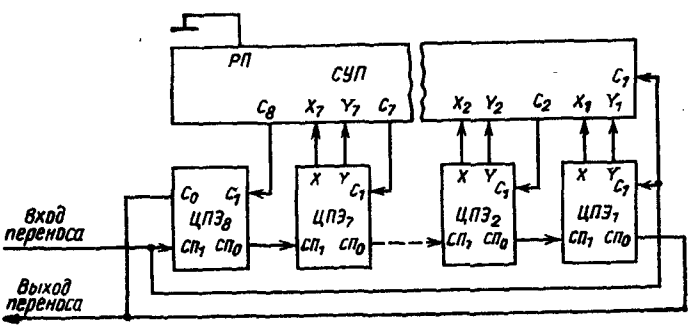


Рис. 5.6. 16-разрядное операционное устройство со сквозной передачей переносов

Пусть X_i, Y_i — подаваемые в СУП сигналы с выводов соответственно X и Y i-й секции операционного устройства и C_i — формируемый в СУП перенос, подаваемый на вход C_1 ЦПЭ i-й секции.

Определим логические выражения для формирования в СУП переносов C_i. При этом будем учитывать, что для подачи на вход C_1 ЦПЭ требуется инверсное значение переноса.

Перенос C_2 определится следующим логическим выражением:

$$\overline{C}_2 = \overline{Y_1 \vee X_1 \cdot \overline{C}_1} = Y1 \cdot (X_1 \vee \overline{C}_1) = Y_1 \cdot X_1 \vee Y_1 \cdot \overline{C}_1.$$

Выражение для инверсии переноса, подаваемого на вход C_1 третьей секции:

$$\overline{C}_3 = Y_2 \cdot X_2 \vee Y_2 \cdot Y_1 \cdot X_1 \vee Y_2 \cdot Y_1 \cdot \overline{C}_1.$$

Подобные выражения можно построить для всех секций операционного устройства. В СУП К589ИК03 указанные логические выражения реализованы для восьми секций.

Перенос C_0 на выходе 2N-разрядного операционного устройства можно получить как перенос, возникающий на выходе C_0 восьмой секции либо непосредственно из СУП, где этот перенос формируется как перенос C_9. Этот выход имеет три состояния, в отключенное состояние он переводится уровнем лог. 0 на выводе РП (*разрешение переноса*) СУП.

Время сквозного переноса в СУП составляет 13 нс, задержка выдачи переноса C_9 — 20 нс.

Варианты использования СУП. На рис. 5.6 представлено включение СУП в схеме 16-разрядного операционного устройства. Перенос C_0 здесь получается как перенос из старшей секции, поэтому на входе РП СУП установлен уровень лог.0. Следует обратить внимание на то, что для передачи переносов при сдвиге вправо используется последовательная цепь путем подключения выхода $СП_0$ секции с входом $СП_1$ соседней младшей секции.

Аналогичная схема операционного устройства на 32 разряда приведена на рис. 5.7. Здесь использованы два блока СУП, каждый из которых обслуживает группу из восьми секций ЦПЭ. В блоке СУП младшей группы секций вход РП находится под уровнем лог. 1, и перенос C_9 с выхода этого блока подается на вход следующего блока. В блоке, обслуживающем старшую группу секций, можно было бы использовать такое же включение, что и на рис. 5.6. На рис. 5.7 показан другой вариант включения, в котором выход C_0 операционного устройства снимается с выхода C_9 блока СУП. Так как этот вывод обычно объединяется в общую цепь с выводом $СП_0$, то возникает необходимость отключения этого вывода C_9 (перевода в состояние высокого выходного сопротивления) при выполнении операции сдвига вправо. Это достигается тем, что уровень напряжения на входе РП второго блока СУП определяется логическим выражением $РП = \overline{\overline{F}_6 \cdot \overline{F}_5 \cdot \overline{F}_4 \cdot F_3 \cdot F_2 \cdot F_1}$ (сдвигу вправо в МК соответствует поле F_6, ..., F_4 с комбинацией 000 и поле

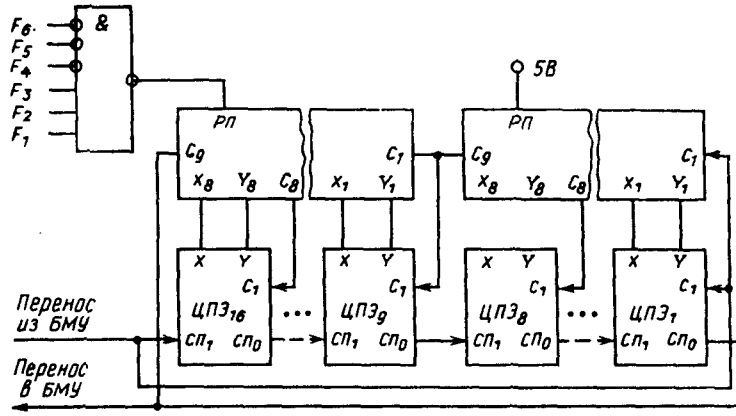


Рис. 5.7. 32-разрядное операционное устройство с сквозной передачей переносов

F_3, ..., F_0 с комбинациями 1110 либо 1111). При операции сдвига вправо это выражение принимает значение лог. 0 и выход переноса C_9 второго блока СУП устанавливается в выключенное состояние. При всех других операциях это логическое выражение принимает значение лог. 1 и с выхода C_9 второго блока СУП снимается перенос C_0 операционного устройства (при этом выход $СП_0$ устанавливается в отключенное состояние).

5.3. ПОСТРОЕНИЕ УПРАВЛЯЮЩЕГО УСТРОЙСТВА

БЛОК МИКРОПРОГРАММНОГО УПРАВЛЕНИЯ К589ИК01

Структурная схема и общий принцип функционирования. На рис. 5.8 приведена упрощенная схема, поясняющая взаимодействие блока микропрограммного управления (БМУ) с памятью МК (управляющей памятью) и операционным устройством.

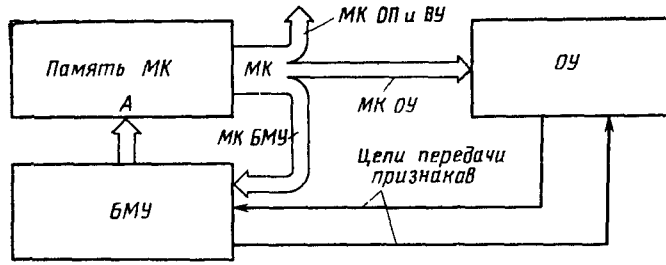


Рис. 5.8. Взаимодействие МБУ с памятью МК и операционным устройством

Сформированный в БМУ адрес МК с момента положительного фронта синхросигнала выдается на шину МА, с которой он далее принимается на адресный вход А блока памяти микрокоманд. Память МК выдает считанную по принятому адресу общую микрокоманду микропроцессорного устройства. Эта МК включает в себя микрокоманды, предназначенные для управления работой различных блоков, входящих в состав МПУ, и имеет следующий формат:

МК ОУ	МК БМУ	МК ОП и ВУ	...

Здесь МК ОУ — микрокоманда, управляющая работой операционного устройства; МК БМУ — микрокоманда, управляющая работой БМУ; МК ОП и ВУ — микрокоманда, управляющая работой оперативной памяти и внешних устройств.

Выделенные из общей МК поля микрокоманд отдельных блоков подаются к соответствующим блокам.

Микрокоманда БМУ, под управлением которой организуется функционирование БМУ, имеет следующий формат:

Поле управления переходами $УА_6 \ldots УА_0$	Поле управления признаками $УФ_3 \ldots УФ_0$	Поле управления загрузкой ЗМ

Под управлением отдельных полей МК БМУ выполняются основные функции этого блока, заключающиеся в формировании адреса очередной МК и хранении и выдаче признаков (переносов), поступающих из ОУ по объединенной цепи $С_0 — СП_0$.

На рис. 5.9 приведена структурная схема БМУ.

Поле управления переходами $УА_6 \ldots УА_0$ задает способ, которым в логической схеме определения адреса следующей МК (ЛСх) формируется адрес очередной МК. При низком уровне синхросигнала открываются входы триггеров регистра адреса МК (РАМК) и происходит прием сформированного в ЛСх девятиразрядного адреса в триггеры регистра. На положительном фронте синхросигнала происходит переключение триггеров регистра в состояния, соответствующие разрядам адреса, входы триггеров логически отключаются от выходов ЛСх. Этот адрес через выходные буферы $ВБ_1$ и $ВБ_2$ выдается на адресную шину МА в виде групп разрядов: $МА_3 \ldots МА_0$ и $МА_8 \ldots МА_4$. Группа разрядов $МА_8 \ldots МА_4$ предназначена для определения в двухмерном массиве ячеек памяти МК адреса строки (номера строк 0 ... 31), группа разрядов $МА_3 \ldots МА_0$ определяет адрес колонки (номера колонок 0...15). Таким образом, обеспечивается адресация памяти емкостью $2^5 \cdot 2^4 = 512$ ячеек.

Выходные буферы имеют три состояния. Буфер адреса колонки $ВБ_1$ выводится из отключенного состояния (состояния с высоким выходным сопротивлением) при уровне лог. 1 на входе общего строба ОС; буфер

адреса строки $ВБ_2$ для вывода из отключенного состояния требует, чтобы уровень лог. 1 действовал одновременно на входе ОС и входе разрешения выдачи адреса строки РС.

Выдаваемый операционным устройством по объединенной цепи $С_0$—$СП_0$ перенос подается на вход Ф БМУ, откуда он при высоком уровне синхросигнала принимается в триггер признака Ф. На отрицательном фронте синхросигнала происходит отключение входа триггера от входной цепи Ф блока, после чего при низком уровне синхросигнала триггер продолжает хранить принятую перед отрицательным фронтом сигнала информацию.

На положительном фронте синхросигнала состояние триггера Ф может быть передано в один либо оба триггера (триггер С и триггер Z) регистра признаков (регистр признаков часто называют регистром флажков).

Поле управления признаками $УФ_3$ ... $УФ_0$ МК БМУ определяет, следует ли производить передачу содержимого регистра Ф в регистр признаков и в какие из его триггеров, а также определяет, следует ли

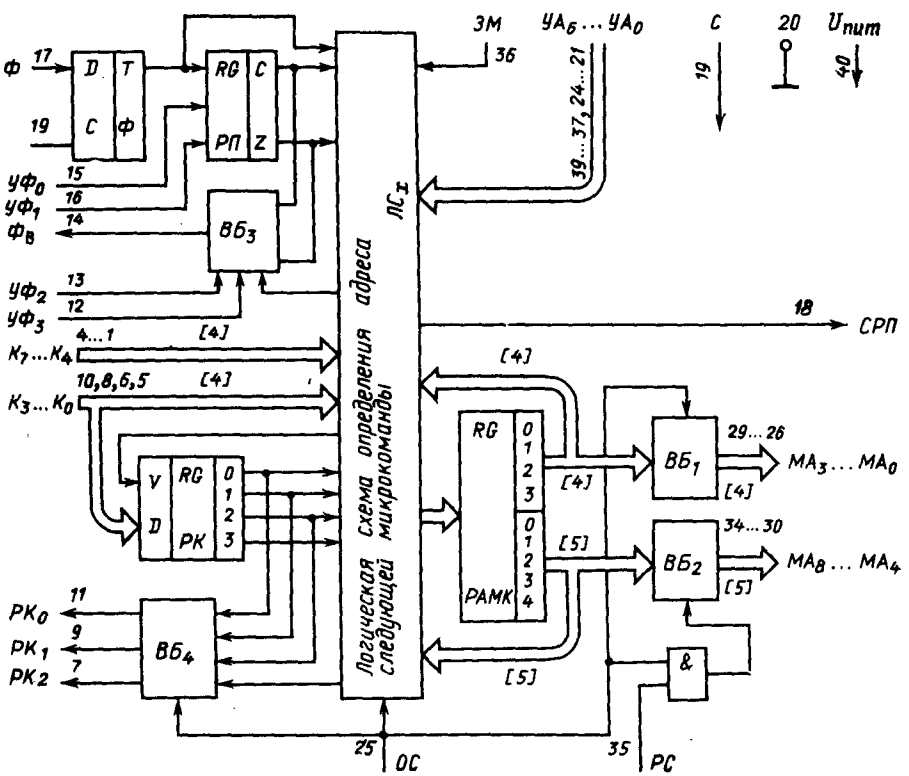


Рис. 5.9. Структурная схема БМУ К589ИК01

передавать на выход $Ф_в$ (для передачи в цепь $С_1$ — $СП_1$ ОУ) содержимого регистра признаков либо установить на этом выходе уровень лог. 0 или лог. 1.

В памяти МК может храниться несколько микропрограмм. Для обращения к некоторой микропрограмме следует указать адрес первой МК этой микропрограммы. Так как в каждой общей МК МПУ поле МК БМУ содержит указания о том, как должен формироваться адрес следующей МК, то после считывания первой МК микропрограммы будет определена вся последовательность МК, входящих в микропрограмму.

Адрес первой МК микропрограммы задается командой, принимаемой на входы $К_7$ ... $К_0$ БМУ. Поле управления загрузкой ЗМ в МК БМУ определяет, следует ли поступающую на входы $К_7$ ... $К_0$ кодовую комбинацию заносить в РАМК.

Управление формированием адреса очередной микрокоманды. В табл. 5.4 показаны способы формирования адреса очередной МК.

Все виды переходов разбиты на три группы: безусловные переходы, условные переходы, переходы по коду команды.

Группа безусловных переходов. Старшие 2 ... 4 разряда поля управления переходом в МК БМУ имеют фиксированные значения для каждого вида перехода. Значения остальных разрядов передаются в младшие разряды адреса строки либо адреса колонки в РАМК, содержимое остальных разрядов РАМК сохраняется, т. е. эти разряды адреса следующей МК сохраняют те значения, какие они имеют в адресе текущей МК. Например, при переходе в текущей колонке (JCC) кодовая комбинация 00 в разрядах $УА_6$ и $УА_5$ определяет вид перехода, содержимое остальных разрядов $УА_4$ ... $УА_0$ служит адресом строки следующей МК; адрес колонки следующей МК сохраняется тем же, что и в адресе текущей МК.

При переходе в текущей колонке в группе адресов строки (JCE) кодовая комбинация 1110 в старших четырех разрядах поля $УА_6$ ... ... $УА_0$ определяет вид перехода, три младших разряда этого поля $У_2$ ... $У_0$ определяют значения соответствующих разрядов адреса строки следующей МК, значения остальных разрядов (двух старших разрядов адреса строки $М_8$, $М_7$ и разряды адреса колонки $М_3$ ... $М_0$) сохраняются теми же, что и в адресе текущей МК. Кроме того, при выполнении этого вида перехода открывается буфер $ВБ_4$, через который на шину $РК_2$ ... $РК_0$ выдается содержимое регистра команд (РК).

Особенность перехода в нулевую строку (JZR) заключается в том, что задается нулевое значение всех разрядов адреса строки следующей МК и, если адресом колонки следующей МК оказывается $МА_3$ ... $МА_0$ = = 1111 (15), выдается сигнал на вывод *строба разрешения прерывания* СРП.

Группа условных переходов. Вид перехода задается тремя-четырьмя старшими разрядами поля $УА_6$ ... $УА_0$ в МК БМУ, остальные разряды этого поля определяют соответствующие младшие

разряды адреса строки следующей МК, старшие 1—2 разряда адреса строки сохраняют прежнее значение.

Адрес колонки следующей МК формируется так: старший разряд $МА_3$ сохраняет то же значение, что и в адресе текущей МК; двум средним разрядам $МА_2$, $МА_1$ сообщается комбинация значений 01; в младший разряд заносится значение признака (т. е. содержимое триггера Ф, триггера С либо триггера Z), по которому производится условный переход. Таким образом, в зависимости от значения признака происходит обращение в одну из двух соседних ячеек памяти МК.

Группа переходов по коду команды. При этих переходах 4—5 старших разрядов поля $УА_6$ ... $УА_0$ определяют вид перехода, остальные разряды этого поля заносятся в соответствующие младшие разряды адреса строки следующей МК (старшие разряды адреса строки сохраняют прежнее значение). Адрес колонки следующей МК формируется следующим образом. При переходе JPR этот адрес определяется содержимым РК; при переходе JLL два старших разряда принимают значение кодовой комбинации 01, остальные два разряда — значение $РК_3$, $РК_2$; при переходе JPL старшие два разряда — 11, младшие два разряда адреса колонки — $РК_1$, $РК_0$; при переходе JPX адресом колонки служит действующая на входах $К_7$ ... $К_4$ кодовая комбинация. При последнем переходе происходит загрузка РК.

Управление признаками. В поле управления признаками МК БМУ $УФ_3$ ... $УФ_0$ младшие два разряда $УФ_1$ и $УФ_0$ определяют прием в регистр признаков РП содержимого триггера Ф,

Таблица 5.4

Мнемоника	Вид перехода	Значение поля переходом			
		$УА_6$	$УА_5$	$УА_4$	$УА_3$
	Безусловные переходы:				
JCC	в текущей колонке	0	0	$У_4$	$У_3$
JZR	в нулевую строку	0	1	0	$У_3$
JCR	в текущей строке	0	1	1	$У_3$
JCE	в текущей колонке в группе адресов строк	1	1	1	0
	Условные переходы:				
JFL	по содержимому триггера Ф	1	0	0	$У_3$
JCF	по содержимому триггера С	1	0	1	0
JZF	по содержимому триггера Z	1	0	1	1
	Переходы по коду команды:				
JPR	по содержимому регистра команд	1	1	0	0
JLL	по левым разрядам регистра команд	1	1	0	1
JRL	по правым разрядам регистра команд	1	1	1	1
JPX	по разрядам команды $К_4$...$К_7$	1	1	1	1

Примечание. $Уi$ – данные по шине $УАi$; $Мi$ – данные в i-ом разряде РАМК; $РКi$

старшие два разряда — выдачу признаков на выход $Ф_в$. Соответствие кодовых комбинаций поля $УФ_3 \ldots УФ_0$ выполняемым под их управлением действиям приведено в табл 5.5.

Управление загрузкой адреса микропрограммы. При значении поля ЗМ = 1 старший разряд РАМК устанавливается в состояние лог. 0, в остальные восемь разрядов РАМК передается кодовая комбинация, содержащаяся в поступающей на входы $К_7 \ldots К_0$ команде. При этом $К_7 \ldots К_4$ определяют адрес колонки следующей МК, $К_3 \ldots К_0$ определяют четыре младших разряда адреса строки следующей МК (табл. 5.6).

Следует заметить, что при ЗМ = 1 блокируются переходы, предусматриваемые полем $УА_6 \ldots УА_0$ МК БМУ, однако не блокируются разрешение выдачи содержимого РК на шину $РК_2 \ldots РК_0$ (при кодовой комбинации $УА_6 \ldots УА_0$, соответствующей переходу JCE) и разрешение на прием в РК информации с шины $РК_2 \ldots РК_0$ (при кодовой комбинации $УА_6 \ldots УА_0$, соответствующей переходу JPX), не запрещается также выдача сигнала СРП и выполнение функций под управлением поля $УФ_3 \ldots УФ_0$.

Строб разрешения прерывания СРП. Если при выполнении перехода JZR адресом очередной МК оказывается (0,15), т. е. в нулевой строке 15-я колонка, то из БМУ по цепи СРП выдается сигнал (уровень лог 1). Этот сигнал принимается в блок приоритетного прерывания (БПП). Если при этом БПП отвечает на этот сигнал уровнем лог. 0 на своем выходе, то последний, поступая на вход раз-

управления в МК БМУ			Адрес строки следующей МК					Адрес колонки следующей МК			
$УА_2$	$УА_1$	$УА_0$	$МА_8$	$МА_7$	$МА_6$	$МА_5$	$МА_4$	$МА_3$	$МА_2$	$МА_1$	$МА_0$
$У_2$	$У_1$	$У_0$	$У_4$	$У_3$	$У_2$	$У_1$	$У_0$	$М_3$	$М_2$	$М_1$	$М_0$
$У_2$	$У_1$	$У_0$	0	0	0	0	0	$У_3$	$У_2$	$У_1$	$У_0$
$У_2$	$У_1$	$У_0$	$М_8$	$М_7$	$М_6$	$М_5$	$М_4$	$У_3$	$У_2$	$У_1$	$У_0$
$У_2$	$У_1$	$У_0$	$М_8$	$М_7$	$У_2$	$У_1$	$У_0$	$М_3$	$М_2$	$М_1$	$М_0$
$У_2$	$У_1$	$У_0$	$М_8$	$У_3$	$У_2$	$У_1$	$У_0$	$М_3$	0	1	Ф
$У_2$	$У_1$	$У_0$	$М_8$	$М_7$	$У_2$	$У_1$	$У_0$	$М_3$	0	1	С
$У_2$	$У_1$	$У_0$	$М_8$	$М_7$	$У_2$	$У_1$	$У_0$	$М_3$	0	1	Z
$У_2$	$У_1$	$У_0$	$М_8$	$М_7$	$У_2$	$У_1$	$У_0$	$РК_3$	$РК_2$	$РК_1$	$РК_0$
$У_2$	$У_1$	$У_0$	$М_8$	$М_7$	$У_2$	$У_1$	$У_0$	0	1	$РК_3$	$РК_2$
1	$У_1$	$У_0$	$М_8$	$М_7$	1	$У_1$	$У_0$	1	1	$РК_1$	$РК_0$
0	$У_1$	$У_0$	$М_8$	$М_7$	$М_6$	$У_1$	$У_0$	$К_7$	$К_6$	$К_5$	$К_4$

данные в i-м разряде РК; Ф, С, Z — содержимое триггеров Ф, С, Z соответственно.

Таблица 5.5

Мнемоника	Управление записью в триггеры C и Z	$УФ_1$	$УФ_0$
SCZ	Установить триггеры C и Z по входу Ф	0	0
STZ	Установить триггер Z по входу Ф	0	1
STC	Установить триггер C по входу Ф	1	0
HCZ	Хранить признаки в триггерах C и Z	1	1

Мнемоника	Управление выходом $Ф_в$	$УФ_3$	$УФ_2$
FF0	Выдать на выход $Ф_в$ лог. 0	0	0
FFC	Выдать на выход $Ф_в$ признак из триггера C	0	1
FFZ	Выдать на выход $Ф_в$ признак из триггера Z	1	0
FF1	Выдать на выход $Ф_в$ лог. 1	1	1

Таблица 5.6

ЗМ	Адрес следующей строки					Адрес следующей колонки			
	MA_8	MA_7	MA_6	MA_5	MA_4	MA_3	MA_2	MA_1	MA_0
0	См. таблицу 5.4								
1	0	K_3	K_2	K_1	K_0	K_7	K_6	K_5	K_4

решения выдачи строки РС, вызывает отключение $ВБ_2$ (перевод в третье состояние с высоким выходным сопротивлением). Адрес строки в этом случае может быть подан извне, минуя БМУ, что позволит прервать выполнение текущей программы и перейти к исполнению новой, так называемой прерывающей программы. Процессы, связанные с прерываниями, будут рассмотрены ниже.

ФУНКЦИОНИРОВАНИЕ МИКРОПРОЦЕССОРА С ПОСЛЕДОВАТЕЛЬНЫМ ПРИНЦИПОМ ЧТЕНИЯ И ИСПОЛНЕНИЯ МИКРОКОМАНД

Принцип функционирования мнкропроцессора рассмотрим на упрощенной схеме, представленной на рис. 5.10. Здесь не показаны оперативная память, устройство прерывания, устройства ввода-вывода данных, устройства сопряжения элементов, которые могут присутствовать в МПУ. На рис. 5.11 показана временна́я диаграмма работы микропроцессора. К моменту t_1 в ЛСх БМУ завершается процесс формирования адреса текущей МК и на положительном фронте синхросигнала С происходит прием адреса в РАМК. На шине адреса MA_8 ... MA_0 по-

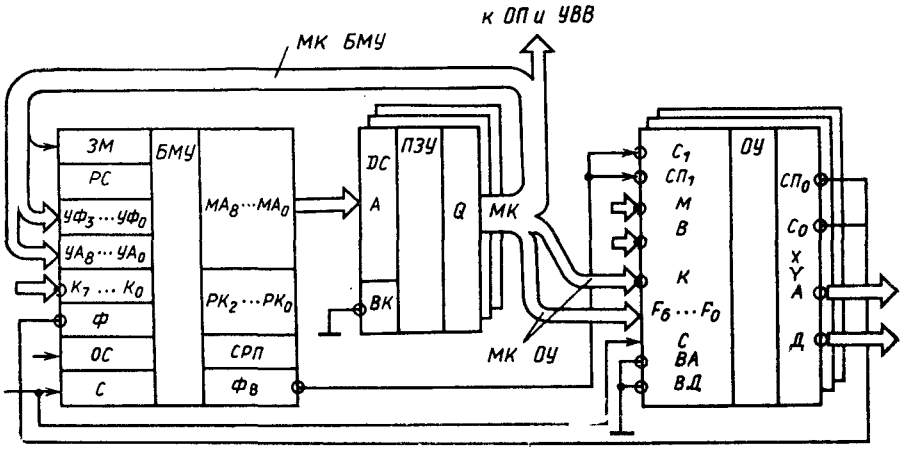


Рис. 5.10. Схема микропроцессора

является адрес текущей МК, и с задержкой, связанной с процессом чтения из ПЗУ микрокоманд, с момента времени t_2 на выходе ПЗУ появляется МК. Микрокоманда содержит ряд полей, управляющих работой БМУ, СУ, ОП и УВВ.

Поля МК БМУ управляют процессом формирования адреса следующей МК, занимающим интервал времени t_3 .. t_4, и выдачей признака на выход $Ф_в$.

Поля МК ОУ управляют процессом выполнения операции в ОУ. В момент времени t_3 на отрицательном фронте синхросигнала С результат операции принимается в назначенный для него регистр. Перенос ($С_0$ или $СП_0$), возникающий в результате выполнения операции в ОУ, в момент t_3 (на отрицательном фронте синхросигнала С) принима-

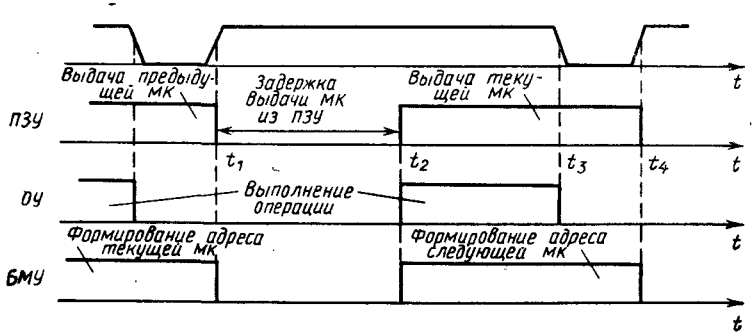


Рис. 5.11. Временные диаграммы работы микропроцессора с последовательным принципом чтения и исполнения микрокоманд

ется со входа Ф и фиксируется в триггере Ф БМУ. В момент времени t_4 (на положительном фронте синхросигнала С) содержимое триггера Ф может быть передано в один либо в оба триггера (триггер С и триггер Z) регистра признаков БМУ.

В момент времени t_4 сформированный адрес следующей МК принимается в РАМК БМУ, начинается процесс чтения из ПЗУ следующей МК.

При описанном способе функционирования микропроцессора процессы чтения из ПЗУ микрокоманды и исполнения ее в ОУ и БМУ носят последовательный характер. Ниже рассматривается режим функционирования микропроцессора, при котором эти процессы совмещаются во времени, что повышает быстродействие мнкропроцессора.

ФУНКЦИОНИРОВАНИЕ МИКРОПРОЦЕССОРА С КОНВЕЙЕРНЫМ ПРИНЦИПОМ ЧТЕНИЯ И ИСПОЛНЕНИЯ МИКРОКОМАНД

Режим функционирования с конвейерным принципом чтения и исполнения МК реализуется в схеме микропроцессора, показанной на рис. 5.12. Отличие этой схемы от схемы на рис. 5.10 состоит в том, что в нее введен *конвейерный регистр*. Пока конвейерный регистр хранит текущую МК, под действием которой в ОУ выполняется операция, из РАМК БМУ выдается адрес следующей МК, и в ПЗУ протекают процессы, связанные с чтением; в это же время в ЛСх БМУ формируется адрес очередной МК. Этн процессы иллюстрируются временно́й диаграммой на рис. 5.13.

В момент времени t_1 в РАМК БМУ поступает адрес $(n+1)$-й МК и с некоторым запаздыванием, связанным с установкой триггеров РАМК и

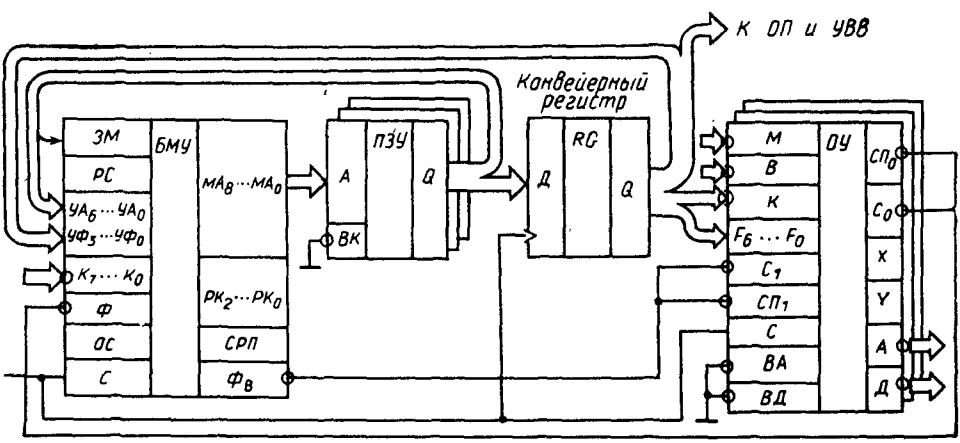


Рис. 5.12. Схема микропроцессора с конвейерным принципом чтения и исполнения микрокоманд

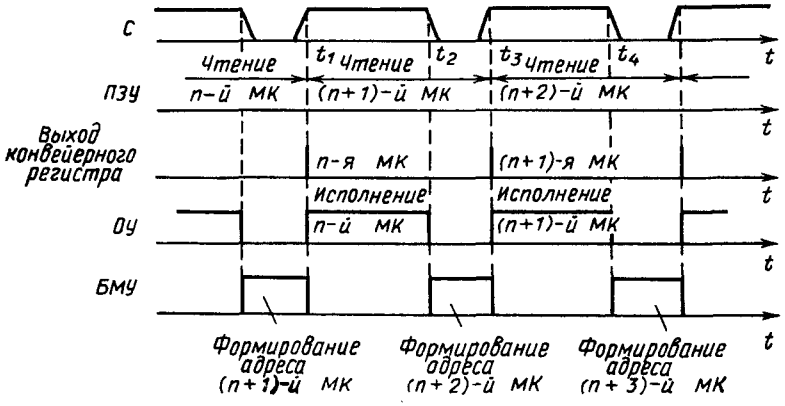


Рис. 5.13. Временны́е диаграммы работы микропроцессора с конвейерным регистром

прохождением адреса через $ВБ_1$ и $ВБ_2$, в ПЗУ начинаются процессы дешифрации адреса и чтения содержимого ячейки. К моменту времени t_2 на выходе ПЗУ устанавливается $(n + 1)$-я МК. Поля $УА_6 \ldots УА_0$ и ЗМ микрокоманды БМУ непосредственно с выхода ПЗУ поступают на соответствующие входы БМУ, и в БМУ происходит формирование адреса $(n + 2)$-й МК. Остальные поля $(n + 1)$-й МК поступают на информационные входы конвейерного регистра и на положительном фронте синхросигнала в момент t_3 фиксируются в этом регистре. Принятые в конвейерный регистр поля МК появляются на выходе регистра, откуда они поступают в соответствующие блоки МПУ: поле $УФ_3 \ldots УФ_0$ — в БМУ, поля $F_6 \ldots F_0$ и К — в ОУ и т. д.

Как видно из временно́й диаграммы на рнс. 5.13, происходит совмещение во времени процессов в БМУ, ПЗУ и ОУ, а именно: в одном и том же тактовом периоде в ОУ выполняется n-я МК, в ПЗУ происходит чтение $(n + 1)$-й МК, а в БМУ формируется адрес $(n + 2)$-й МК. Такой принцип параллельного функционирования блоков МПУ, называемый *конвейерным*, существенно повышает быстродействие МПУ.

ОПРЕДЕЛЕНИЕ ДЛИТЕЛЬНОСТИ ТАКТОВОГО ПЕРИОДА

Определим длительность тактового периода при конвейерном принципе и при использовании блоков СУП в ОУ. Как показано на рис 5.14, *а*, длительность тактового периода для этого случая

$$\tau_1 = \tau_{КР} + \tau_{ОУ} + \tau_и,$$

где $\tau_{КР}$ — задержка выдачи МК из конвейерного регистра (примем $\tau_{КР} = 30$ нс), $\tau_и$ — длительность импульса синхросигнала (пусть

$\tau_и = 33$ нс), $\tau_{ОУ}$ — длительность выполнения операции в ОУ, определяемая выражением

$$\tau_{ОУ} = \tau_{XY} + \tau_{СУП} + \tau_С.$$

Здесь τ_{XY} — задержка выдачи сигналов X и Y ($\tau_{XY} = 37$ нс), $\tau_{СУП}$ — время сквозной передачи переносов через блоки СУП (при m блоках СУП $\tau_{СУП} = 13\ m$ нс), $\tau_С$ — время установки переноса ($\tau_С \geqslant 27$ нс).

Пусть число разрядов ОУ $n = 32$, для организации ускоренного переноса потребуется $m = 2$ блока СУП. При этом $\tau_{ОУ} \geqslant 37 + 27 + 13 \cdot 2 = 90$ нс. Следовательно, $\tau_1 \geqslant 30 + 90 + 33 = 153$ нс.

Рассмотрим вариант без использования конвейерного регистра, но с применением СУП.

Составляющие длительности тактового периода для этого случая показаны на рис. 5.14, *б*:

$$\tau_2 = \tau_{БМУ} + \tau_{ПЗУ} + \tau_{ОУ} + \tau_и,$$

где $\tau_{БМУ}$ — задержка выдачи адреса из БМУ ($\tau_{БМУ} = 30$ нс); $\tau_{ПЗУ}$ — время обращения к ПЗУ.

Пусть $\tau_{ПЗУ} = 70$ нс. Тогда при числе разрядов ОУ $n = 32$ $\tau_2 \geqslant \geqslant 30+70+90+33=223$ нс.

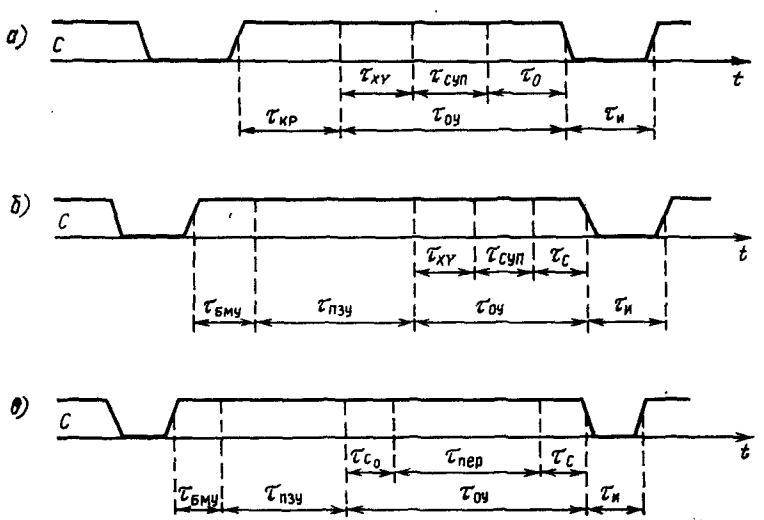


Рис. 5.14. Определение длительности тактового периода:

а) УУ с конвейерным регистром, ОУ с использованием блоков СУП; *б)* УУ без конвейерного регистра, ОУ с использованием блоков СУП; *в)* УУ без конвейерного регистра, ОУ без использования блоков СУП

Наконец, рассмотрим вариант без использования конвейерного регистра и блоков СУП. Длительность $\tau_{ОУ}$ для этого случая (рис. 5.14, *в*) определяется соотношением

$$\tau_{ОУ} = \tau_{C_0} + \tau_{пер} + \tau_{C},$$

где τ_{C_0} — задержка выдачи переноса из первой секции ОУ ($\tau_{C_0} = 43$ нс), $\tau_{пер}$ — время передачи переносов с выхода первой секции до входа последней секции [$(n/2 - 2) \cdot 14$ нс].

При $n = 32$ $\tau_{ОУ} \geqslant 43 + 196 + 27 = 266$ нс. Следовательно, $\tau_з \geqslant 30 + 70 + 266 = 366$ нс.

5.4. ПРИЕМЫ ПРОГРАММИРОВАНИЯ

ПРОГРАММИРОВАНИЕ ПОСЛЕДОВАТЕЛЬНЫХ УЧАСТКОВ АЛГОРИТМА

Рассмотрим такие участки алгоритма, которые не содержат разветвлений.

Пример 5.1. Требуется принять из ОЗУ числа a_1 и a_2 и, вычислив разность $a_1 - a_2$, поместить ее в ОЗУ на место числа a_2. Адрес числа a_1 хранится в регистре R_3 ОУ, адрес числа a_2 на единицу больше адреса числа a_1.

На рис. 5.15 приведена схема алгоритма рассматриваемого примера.

В блоке *1*: в ОУ осуществляется пересылка адреса числа a_1 из регистра R_3 в РА, в R_3 формируется адрес числа a_2.

В блоке *2*: в ОЗУ производится чтение числа a_1, которое принимается в подключенный к выходу ОЗУ регистр; в ОУ при этом выполняется холостая операция (например, операция типа $R_n \rightarrow R_n$, не изменяющая содержимого регистров ОУ).

В блоке *3*: поступающее из регистра ОЗУ число a_1 принимается в АС ОУ.

В блоке *4*: в ОУ адрес числа a_2 из регистра R_3 пересылается в РА, содержимое R_3 сохраняется неизменным.

В блоке *5*: в ОУ содержимое АС передается в R_4, аккумулятор АС освобождается для приема следующего числа; в ОЗУ производится чтение числа a_2.

В блоке *6*: в ОУ производится прием инверсии числа a_2 в АС.

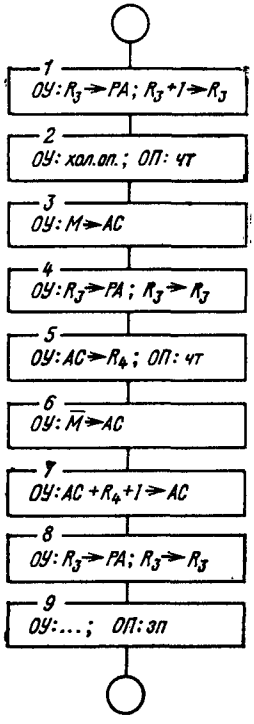


Рис. 5.15. Схема алгоритма вычисления разности чисел, хранящихся в оперативной памяти

В блоке *7*: в ОУ содержимое АС суммируется с единицей и инвертированным значением числа a_2.

Блоки *8* и *9* производят запись результата операции в ОЗУ.

Описанным способом выполняется вычитание в тех случаях, когда участвующие в операции числа представлены в форме с фиксированной точкой и отрицательные числа в ОЗУ хранятся в дополнительном коде. При этом изменение знака числа a_2 на обратный, необходимое перед суммированием, осуществляется инвертированием в a_2 всех разрядов с прибавлением затем единицы в младший разряд. Инвертирование разрядов a_2 выполняется в блоке *6*, а прибавление единицы — при суммировании в блоке *7*.

Пусть МПУ управляется микрокомандами, имеющими следующий формат:

$УА_6 \ldots УА_0$	$УФ_3 \ldots УФ_0$	ЗМ	$F_6 \ldots F_0$	К	$\overline{ВК}$	Чт/$\overline{Зп}$
МК БМУ			МК ОУ		МК ОЗУ	

На входах микросхем БМУ и ЦПЭ устанавливаем сигналы: в БМУ РС = 1, ОС = 1; в ЦПЭ ВД = 0, ВА = 0.

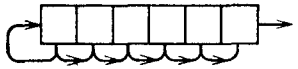
В табл. 5.7 приведена микропрограмма решения рассматриваемой задачи.

Рассмотрим подробно построение первой микрокоманды.

Пусть адресом этой МК в ПЗУ является 00001 1100_2 (1,12). Рассмотрим содержание хранящейся по этому адресу МК. Так как не предусматривается обращения к ОЗУ, то в разряд $\overline{ВК}$ занесена лог. 1, значение разряда Чт/$\overline{Зп}$ при этом безразлично. Операция $R_3 \rightarrow РА$; $R_3 + 1 \rightarrow R_3$ в ОУ выполняется микрокомандой LMI при $C_1 = 1$.

Таблица 5.7

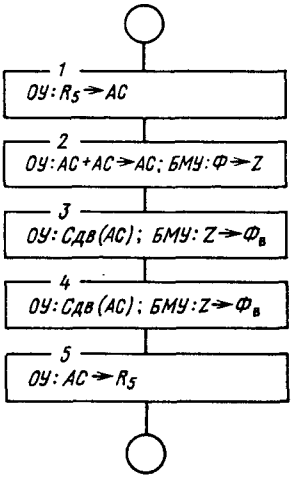
Адрес МК в ПЗУ		Поля МК БМУ			Поля МК ОУ		Поля ОЗУ		Пояснения
Адрес строки	Адрес колонки	$УА_6 \ldots УА_0$	$УФ_3 \ldots УФ_0$	ЗМ	$F_6 \ldots F_0$	К	$\overline{ВК}$	Чт/$\overline{Зп}$	
00001	1100	011 1101	11 11	0	001 0011	0	1	/	Блок *1*
00001	1101	011 1110	00 11	0	110 0000	0	0	1	Блок *2*
00001	1110	011 1111	00 11	0	110 1011	0	1	0	Блок *3*
00001	1111	00 00010	00 11	0	001 0011	0	1	>	Блок *4*
00010	1111	011 1110	11 11	0	010 0100	1	0	1	Блок *5*
00010	1110	011 1101	00 11	0	111 1011	0	1	0	Блок *6*
00010	1101	011 1100	11 11	0	000 1101	1	1	×	Блок *7*
00010	1100	011 1011	00 11	0	001 0011	0	1	0	Блок *8*
00010	1011	...	...	...	...	...	0	0	Блок *9*



а)

	АС	Действие
0	1 0 1 1 0 1	Исходное состояние
1	0 1 1 0 1 0	Сдвиг влево
1	1 0 1 1 0 1	Сдвиг вправо
	1 1 0 1 1 0	Сдвиг вправо

б)



в)

Рис. 5.16. Арифметический сдвиг вправо:

а) принцип сдвига; *б)* иллюстрация операции сдвига на примере; *в)* схема алгоритма операции

Микрокоманда LMI определяется: F-группа = 2 (следовательно, $F_6 \ldots F_4 = 010$), R-группа = 1 (следовательно, регистру R_3 соответствует $F_3 \ldots F_0 = 0011$). Значение $C_1 = 1$ обеспечивается комбинацией $УФ_3УФ_2 = 11$ в МК БМУ. Значение разрядов $УФ_1\ УФ_0$ можно выбрать равным 11, что соответствует хранению признаков, содержащихся в триггерах C и Z регистра признаков. Адрес следующей МК определяем полем $УА_6 \ldots УА_0$ (следовательно, ЗМ = 0). Выберем переход в текущей строке ($УА_6 \ldots УА_4 = 011$) к ячейке в соседней колонке ($УА_3 \ldots УА_0 = 1101$).

Относительно некоторых из остальных МК дадим лишь краткие пояснения. В МК блока *2* производится обращение к ОЗУ ($\overline{ВК} = 0$) для чтения ($Чт/\overline{Зп} = 1$). В МК блока *3* значение $Чт/\overline{Зп} = 0$ обеспечивает сохранение в регистре ОЗУ информации, полученной из ОЗУ, в течение тактового периода, следующего за тактом чтения из ОЗУ. В МК блока *4* особенность построения поля $УА_6 \ldots УА_0$ состоит в том, что оно предусматривает переход в текущей колонке ($УА_6УА_5 = 00$) в соседнюю строку с адресом $УА_4 \ldots УА_0 = 00010$.

Рассмотрим пример программирования с использованием триггера регистра признаков.

Пример 5.2. Требуется выполнить операцию арифметического сдвига вправо над содержимым регистра R_5 ОУ.

Особенность арифметического сдвига вправо состоит в том, что восстанавливается содержимое старшего (знакового) разряда (рис. 5.16,*а*). Эта операция может быть выполнена следующим образом. Так как операция сдвига может выполняться над содержимым аккумулятора АС, то необходимо предварительно содержимое регистра R_5 передать в АС. Далее путем сдвига влево, осуществляемого удвоением содер-

Таблица 5.8

Адрес МК в ПЗУ		Поля МК БМУ			Поля МК ОУ		Пояснения
Адрес строки	Адрес колонки	$УА_6...УА_0$	$УФ_3...УФ_0$	ЗМ	$F_6...F_0$	К	
00010	0000	011 0001	00 11	0	000 0101	0	МК1
00010	0001	011 0010	00 01	0	011 1101	1	МК2
00010	0010	011 0011	10 11	0	000 1111	0	МК3
00010	0011	011 0100	10 11	0	000 1111	0	МК4
00010	0100	011 0101	11 11	0	010 0101	1	МК5
00010	0101	...	...	...	...	...	...

жимого АС, из АС выдвигается знаковый разряд числа (в виде переноса C_0, принимаемого в триггер Ф). На положительном фронте синхросигнала знаковый разряд из триггера Ф передается в один из триггеров регистра признаков (например, в триггер Z). Затем следуют два последовательно выполняемых сдвига вправо. При каждом сдвиге знаковый разряд из регистра признаков выдается на выход $Ф_в$, откуда он поступает на вход $СП_1$ ОУ и вписывается в старший разряд АС. Эти процессы иллюстрируются приведенным на рис. 5.16, *б* примером.

На рис. 5.16, *в* дана схема алгоритма арифметического сдвига вправо, в табл. 5.8 — соответствующая ей микропрограмма.

Здесь во второй МК комбинация $УФ_1УФ_0 = 01$ обеспечивает передачу содержимого триггера Ф в триггер Z регистра признаков; в третьей и четвертой МК комбинацией $УФ_3\ УФ_2 = 10$ производится выдача содержимого триггера Z на выход $Ф_в$ БМУ.

ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЕНИЙ

Разветвления в программах реализуются с помощью условных переходов. При этом в качестве признака, по которому выполняется условный переход, может быть использовано содержимое триггера Ф либо триггеров С и Z регистра признаков БМУ. Следует иметь в виду, что содержимое триггера Ф при низком уровне синхросигнала (когда оно используется в БМУ для выполнения условного перехода) в каждом тактовом периоде оказывается обновленным. Оно соответствует признаку, формируемому в ОУ в результате выполнения операции.

Следовательно, признаком, по которому осуществляется условный переход, может служить содержимое триггера Ф лишь в том случае, если выдача этого признака нз ОУ на вход Ф БМУ и условный переход в БМУ происходят в одном и том же тактовом периоде. Для выполнения этого условия в МПУ с конвейерным регистром, как это будет показано ниже, может потребоваться введение микрокоманды, не представленной в схеме алгоритма в виде блока. Это происходит в тех случаях, когда в схеме алгоритма блок условного перехода непосредст-

венно следует за блоком, в котором формируется признак для этого перехода.

Если же указанное выше условие нарушается за счет того, что формирование признака в ОУ происходит в более ранних тактовых периодах по сравнению с периодом, в котором должен в БМУ выполняться условный переход по этому признаку, то переход по содержимому триггера Ф оказывается невозможным. В этом случае необходимо в тактовом периоде, в котором формируется в ОУ признак, передать содержимое триггера Ф в один из триггеров регистра признаков (в триггер С либо в триггер Z). Затем условный переход осуществляется по содержимому триггера С либо триггера Z.

Рассмотрим эти особенности на примерах.

Пример 5.3. Требуется проанализировать значение младшего разряда числа в регистре R_0 ОУ. Если значение этого разряда равно 0_2, то содержимое регистра R_0 следует сложить с содержимым регистра R_1, в противном случае оно должно быть просуммировано с содержимым регистра R_2.

На рис. 5.17 приведена схема алгоритма решения этой задачи. Признак, по которому осуществляется разветвление, формируется как перенос, возникающий при сдвиге вправо переданного из R_0 в АС числа. Так как условный переход выполняется непосредственно после формирования признака разветвления, переход можно осуществлять по содержимому триггера Ф БМУ.

Пусть МПУ построено с использованием последовательного принципа чтения и исполнения микрокоманд (т. е. без конвейерного регист-

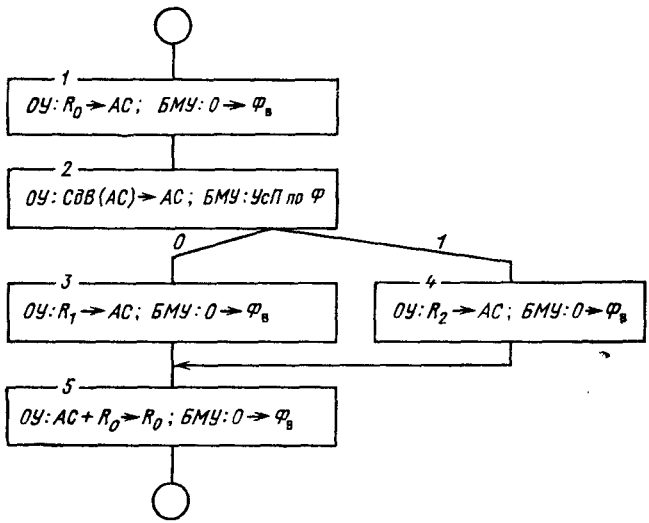


Рис. 5.17. Схема алгоритма разветвляющегося процесса в микропроцессоре без конвейерного регистра

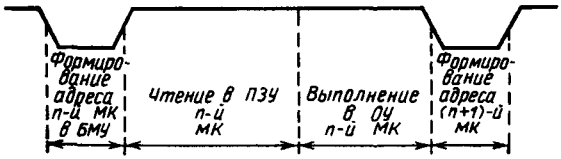


Рис. 5.18. Временна́я диаграмма процессов в микропроцессоре без конвейерного регистра

ра). Соответствующая этому режиму временная диаграмма представлена на рис. 5.18.

В табл. 5.9 представлены действия, выполняемые в отдельные тактовые периоды в блоках ОУ, БМУ и ПЗУ. В конце $(n + 1)$-го тактового периода формируется адрес МК1. В $(n + 2)$-м тактовом периоде из ПЗУ считывается МК1 и затем в этом же тактовом периоде исполняется в ОУ, в конце этого тактового периода в БМУ формируется адрес МК2. В $(n + 3)$-м тактовом периоде считывается из ПЗУ и исполняется в ОУ МК2. К моменту отрицательного фронта синхросигнала в ОУ завершается формирование признака (при сдвиге вправо выдвигается младший разряд содержимого АС), по которому в этом же тактовом периоде (при низком уровне синхросигнала) в БМУ выполняется условный переход по содержимому триггера Ф. В $(n + 4)$-м тактовом периоде, в зависимости от значения признака, будет считываться и исполняться МК3 либо МК4, и в БМУ формируется адрес МК5. В $(n + 5)$-м тактовом периоде происходит считывание и выполнение МК5.

В табл. 5.10 показана последовательность обращений к ячейкам ПЗУ в процессе исполнения рассматриваемого фрагмента микропрограммы. В табл. 5.11 приведена микропрограмма, осуществляющая решение задачи.

Для МК1 выделена ячейка ПЗУ с адресом (2,11). Следующую МК предусматривается поместить в ячейку с адресом (2,4). Пропуск двух очередных ячеек с адре́сами (2,2) и (2,3) связан с тем, что в соответствии с табл. 5.4 при условных переходах адрес колонки следующей МК имеет структуру M_3 01 П, где П — значение признака (т. е. содержимого триггеров Ф, С или Z), по которому осуществляется условный переход. Пропускаемые ячейки имеют в двух средних разрядах двоичного адреса колонки кодовую комбинацию 01 и предназначаются для выполнения условного перехода.

Рассмотрим последовательность действий при реализации данного алгоритма в МПУ с конвейерным принципом чтения и исполнения микрокоманд. В табл. 5.12 показаны действия, выполняемые в отдельных тактовых периодах в БМУ, ПЗУ и ОУ.

Функционирование блоков в этом случае имеет особенность, состоящую в том, что если в $(n + 1)$-м тактовом периоде в БМУ формируется адрес МК1, то эта микрокоманда считывается из ПЗУ в $(n + 2)$-м тактовом периоде, а выполняется в ОУ в $(n + 3)$-м тактовом периоде.

Таблица 5.9

Номер тактового периода	Формирование в БМУ адреса МК	Чтение МК из ПЗУ	Исполнение МК в ОУ
$n+1$	МК1	...	...
$n+2$	МК2	МК1	МК1
$n+3$	УсП по Ф	МК2	МК2
$n+4$	МК5	МК3 или МК4	МК3 или МК4
$n+5$	...	МК5	МК5

Таблица 5.10

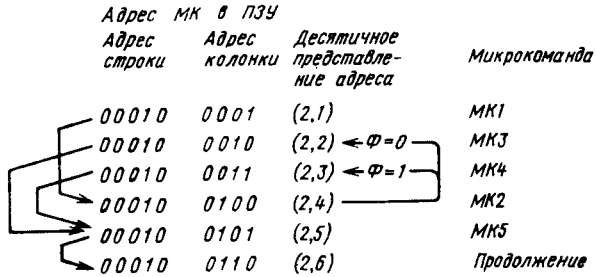


Таблица 5.11

Адрес МК в ПЗУ		Поля МК БМУ			Поля МК ОУ		Пояснения
Адрес строки	Адрес колонки	$УА_6...УА_0$	$УФ_3...УФ_0$	ЗМ	$F_8...F_0$	К	
00010	0001	011 0100	00 11	0	000 0000	0	МК1
00010	0010	011 0101	00 11	0	000 0001	0	МК3
00010	0011	011 0101	00 11	0	000 0010	0	МК4
00010	0100	100 0010	00 11	0	000 1111	0	МК2
00010	0101	011 0110	00 11	0	000 0000	1	МК5
00010	0110	...	...	...	...	...	Продолжение

Таблица 5.12

Номер тактового периода	Формирование в БМУ адреса МК	Чтение МК из ПЗУ	Исполнение МК в ОУ
$n+1$	МК1	...	...
$n+2$	МК2	МК1	...
$n+3$	*	МК2	МК1
$n+4$	УсП по Ф	*	МК2
$n+5$	МК5	МК3 или МК4	*
$n+6$	...	МК5	МК3 или МК4
$n+7$	...	...	МК5

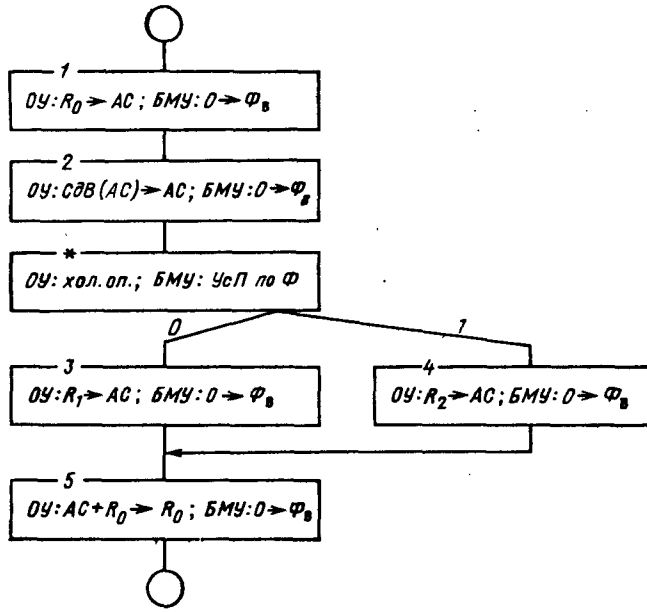


Рис. 5.19. Схема алгоритма разветвляющегося процесса в микропроцессоре с конвейерным регистром

Поле $УА_6$ ... $УА_0$ МК, считанной из ПЗУ в некотором тактовом периоде, в том же тактовом периоде поступает в БМУ и управляет формированием адреса следующей МК. Следовательно, это поле в МК1 должно определить способ формирования адреса МК2 и т. д. Как видно из табл. 5.12, операция МК2 исполняется в ОУ в $(n + 4)$-м тактовом периоде; в этом тактовом периоде формируется признак, по которому в БМУ выполняется условный переход. Таким образом, в последовательности считываемых МК образуется разрыв, отмеченный в таблице звездочкой.

Таблица 5.13

Адрес МК в ПЗУ: Адрес строки	Адрес колонки	Десятичное представление адреса	Микрокоманда
0 0 0 1 0	0 0 0 1	(2,1)	МК1
0 0 0 1 0	0 0 1 0	(2,2)	МК3
0 0 0 1 0	0 0 1 1	(2,3)	МК4
0 0 0 1 0	0 1 0 0	(2,4) Ф=1 Ф=0	МК2
0 0 0 1 0	0 1 0 1	(2,5)	МК*
0 0 0 1 0	0 1 1 0	(2,6)	МК5
0 0 0 1 0	0 1 1 1	(2,7)	*Продолжение*

Таблица 5.14

Адрес МК в ПЗУ		Поля МК БМУ			Поля МК ОУ		Пояснения
Адрес строки	Адрес колонки	$УА_6...УА_0$	$УФ_3...УФ_0$	ЗМ	$F_6...F_0$	К	
00010	0001	011 0100	00 11	0	000 0000	0	МК1
00010	0010	011 0110	00 11	0	000 0001	0	МК3
00010	0011	011 0110	00 11	0	000 0010	0	МК4
00010	0100	011 0101	00 11	0	000 1111	0	МК2
00010	0101	100 0010	00 11	0	110 0000	0	МК*
00010	0110	011 0111	00 11	0	000 0000	0	МК5
...	...	...	...	...	...	...	...

Так как в каждом тактовом периоде БМУ должен формировать адрес некоторой МК, которая в следующем тактовом периоде считывается из ПЗУ, необходимо на месте, отмеченном в таблице звездочкой, указать некоторую МК, предусматривающую выполнение в ОУ холостой операции (операции, не изменяющей содержимого регистров, например, операции вида $R_n \rightarrow R_n$). В поле $УА_6 \ldots УА_0$ этой МК заносится код условного перехода по содержимому триггера Ф.

На рис. 5.19 показана схема алгоритма, в табл. 5.13 и 5.14 приведены последовательность выборки ячеек ПЗУ в процессе исполнения микропрограммы МПУ с конвейерным регистром и сама микропрограмма.

Из сравнения приведенных в табл. 5.11 и 5.14 микропрограмм видно, что применение конвейерного принципа чтения и исполнения микропрограмм привело к необходимости использования дополнительной микрокоманды МК*, содержащей в поле МК ОУ кодовую комбинацию холостой операции пересылки вида $R_0 \rightarrow R_0$. Следует заметить, что можно избежать введения холостой операции, если в микрокоманде условного перехода окажется возможность предусмотреть выполнение в ОУ одной из тех операций, которые должны следовать после МК5.

Перейдем к рассмотрению программы с более сложным видом разветвления по содержимому триггера Ф и одного из триггеров регистра признаков.

Пример 5.4. Рассмотрим реализацию алгоритма Евклида для нахождения наибольшего общего делителя двух положительных чисел a и b (схема алгоритма показана на рис. 5.20).

Алгоритм в микрокомандах приведен на рис. 5.21. Пусть исходные числа a и b хранятся соответственно в регистрах R_1 и R_2 ОУ. Микрокоманды МК1 и МК2 производят дублирование

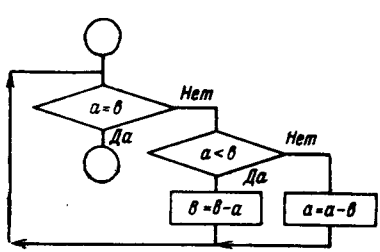


Рис. 5.20. Схема алгоритма Евклида для нахождения общего наибольшего делителя двух положительных чисел

содержимого регистра R_1 в регистре R_4 (необходимость этого действия будет поясиена ниже). МК3 и МК4 формируют в АС обратный код содержимого регистра R_2, что необходимо для выполнения операции вычитания $a—b$ в МК5. Необходимость дублирования содержимого R_1 в R_4, о чем говорилось выше, вызвана тем, чтобы при выполнении операции МК5 не нарушать содержимого регистра R_1. МК5 передает в триггер С значение возникающего в результате выполнения МК5 переноса; МК6 производит проверку на нуль содержимого АС. Результат такой проверки в виде дизъюнкции всех разрядов АС выдается в цепь C_0 и принимается в триггер Ф БМУ.

Разветвление по содержимому триггера Ф (МК7) соответствует разветвлению по результату проверки условия $a = b$. Если $a < b$, то при выполнении операции МК5 формируется $C_0 = 0$, в противном случае $C_0 = 1$. Убедимся в этом на примере.

Пусть $a = 0101_2$ (5_{10}), $b = 0111_2$ (7_{10}).

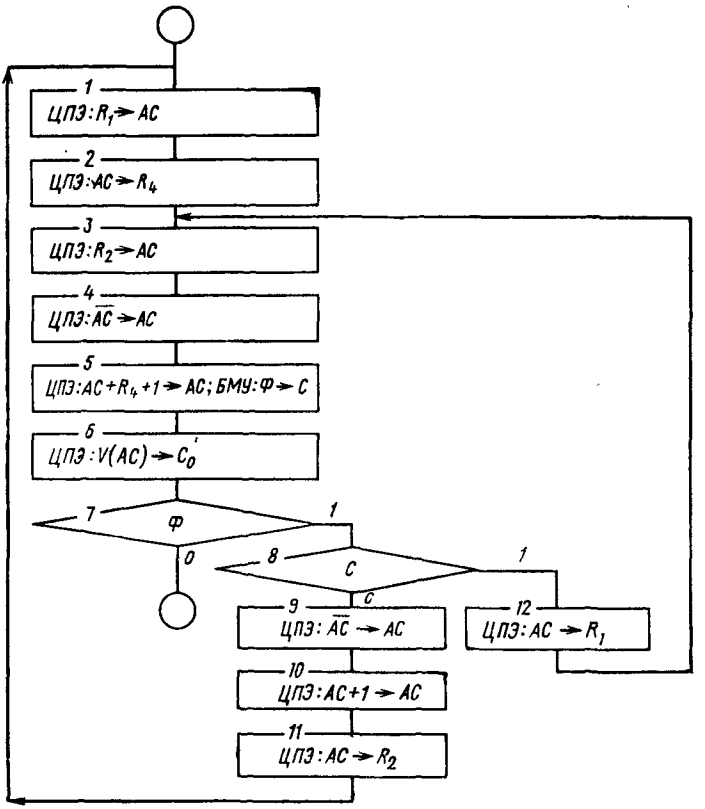


Рис. 5.21. Схема алгоритма Евклида в микрокомаидах

Определяемая МК5 сумма: $a + 1 + b_{\text{обр}} = {}_{+}\begin{array}{r}0101\\1000\\1\\\hline 1110\end{array}$

Перенос C_0 равен 0, что соответствует случаю $a < b$, а полученный результат есть дополнительный код разности чисел $(a - b)_{\text{доп}}$. Перенос C_0 в конце выполнения операции МК5 принимается в триггер Ф и передается затем из этого триггера в триггер С. В рассмотренном случае С = 0 и выполняются операции МК9 ... МК11; здесь формируется прямой код модуля разности $|a - b| = \overline{(a - b)}_{\text{доп}} + 1 = 0010$, который передается в регистр R_2.

Рассмотрим случай, когда условие $a < b$ не выполняется. Например, пусть $a = 0111_2\ (7_{10})$, $b = 0101_2\ (5_{10})$. При этом

$$a + b_{\text{обр}} + 1 = {}_{+}\begin{array}{r}0111\\1010\\1\\\hline 1\leftarrow 0010\end{array}$$

Перенос $C_0 = 1$. В этом случае будет выполняться операция МК12.

Составляя микропрограмму, реализующую рассматриваемый алгоритм Евклида, будем иметь в виду микропроцессорное устройство с конвейерным принципом чтения и исполнения микрокоманд.

В табл. 5.15 показано размещение микрокоманд в ячейках ПЗУ и последовательность, в которой происходит обращение к этим ячей-

Таблица 5.15

Адрес МК в ПЗУ: Адрес строки	Адрес колонки	Десятичное представление адреса		Микрокоманда
00010	0000	(2,0)		МК1
00010	0001	(2,1)		МК2
00010	0010	(2,2)		Продолжение
00010	0011	(2,3)		МК*
00010	0100	(2,4)	Ф=0	МК3
00010	0101	(2,5)	Ф=1	МК4
00010	0110	(2,6)		МК5
00010	0111	(2,7)		МК7
00010	1000	(2,8)		МК6
00010	1001	(2,9)		МК8
00010	1010	(2,10)	C=0	МК9
00010	1011	(2,11)	C=1	МК12
00010	1100	(2,12)		МК10
00010	1101	(2,13)		МК11

Таблица 5.16

Адрес МК в ПЗУ		Поля МК БМУ			Поля МК ОУ		Пояснения
Адрес строки	Адрес колонки	$УА_6...УА_0$	$УФ_3...УФ_0$	ЗМ	$F_8...F_0$	К	
00010	0000	011 0001	00 11	0	000 0001	0	МК1
00010	0001	011 0100	11 11	0	010 0100	1	МК2
00010	0010	...	...	...	...	.	Продолжение
00010	0011	011 1001	00 11	0	110 0000	0	МК*
00010	0100	011 0101	00 11	0	000 0010	0	МК3
00010	0101	011 0110	00 11	0	111 1111	0	МК4
00010	0110	011 1000	11 00	0	000 0100	1	МК5
00010	0111	100 0010	00 11	0	110 0000	0	МК7
00010	1000	011 0111	00 11	0	101 1111	1	МК6
00010	1001	101 0010	00 11	0	110 0000	0	МК8
00010	1010	011 1100	00 11	0	111 1111	0	МК9
00010	1011	011 0100	11 11	0	010 0001	1	МК12
00010	1100	011 1101	11 11	0	000 1101	0	МК10
00010	1101	011 0000	11 11	0	010 0010	1	МК11

кам в процессе исполнения микропрограммы; в табл. 5.16 приведена микропрограмма, реализующая рассматриваемый алгоритм.

Необходимость в МК* связана со следующим. Из ячейки (2,3), в которой размещена эта микрокоманда, требуемое микрокомандой МК8 разветвление по содержимому регистра С в данной строке можно произвести лишь в пару ячеек с адресами (2,2) и (2,3); однако в эту пару ячеек уже производилось разветвление микрокомандой МК7. Поэтому возможны два приема: либо осуществление разветвления в подобную пару ячеек, но расположенную в какой-либо другой строке, либо переход в какую-либо другую ячейку данной строки, из которой возможно разветвление в пару ячеек с адресами (2, 10) и (2, 11). Последний прием и был принят при составлении микропрограммы: МК* используется для выполнения безусловного перехода к ячейке с адресом (2,9), в которой размещается МК8, осуществляющая разветвление по содержимому регистра С в пару ячеек с адресами (2, 10) и (2, 11).

МИКРОПРОГРАММА СЛОЖЕНИЯ

Рассмотрим алгебраическое сложение, т. е. операцию суммирования, в которой могут участвовать как положительные, так и отрицательные числа. Как это принято в цифровой технике, знак числа будем определять значением его старшего (так называемого знакового) разряда, записывая в нем 0 в случае положительного числа и 1 в случае отрицательного числа. Для того чтобы процесс суммирования с участием отрицательных чисел происходил по тем же правилам, что и суммирование положительных чисел, для представления отрицательных чи-

сел используется дополнительный либо обратный код. В дальнейшем будем полагать, что используется дополнительный код.

При суммировании чнсел, имеющих одинаковые знаки, возможно переполнение разрядной сетки, при котором результат суммирования не помещается в отведенных для него разрядах и, занимая зиаковый разряд, искажает знак числа. В процессе выполнения операции необходимо выявлять случаи возникновения переполнения разрядной сетки. Обнаруживая такие случаи, следует предусматривать соответствующую индикацию и останов вычислений либо переход к выполнению спецнальной микропрограммы. Рассмотрим подобные случаи на примерах.

Пример 5.5. Суммирование положительных чисел. Пусть $N_1 =$ $= \mathbf{0}\ 1011010$, $N_2 = \mathbf{0}\ 0111011$ (здесь полужирным шрифтом выделены значения знакового разряда чисел).

Переносы		1	1	1	1		1	
N_1	+	**0**	1	0	1	1	0	1	0
N_2		**0**	0	1	1	1	0	1	1
$N_2 + N_1$		**1**	0	0	1	0	1	0	1

Получен ошибочный результат — сумма положительных чисел не может иметь отрицательный знак.

Пример 5.6. Суммнрование двух отрицательных чисел. Пусть

$N_1 = \mathbf{1}\ 1\ 0\ 1\ 1\ 0\ 1\ 0$ ($N_{1доп} = \mathbf{1}\ 0\ 1\ 0\ 0\ 1\ 1\ 0$),
$N_2 = \mathbf{1}\ 0\ 1\ 1\ 1\ 0\ 1\ 1$ ($N_{2доп} = \mathbf{1}\ 1\ 0\ 0\ 0\ 1\ 0\ 1$).

Переносы		1					1	
$N_{1доп}$	+	**1**	0	1	0	0	1	1	0
$N_{2доп}$		**1**	1	0	0	0	1	0	1
$N_{1доп} + N_{2доп}$		**0**	1	1	0	1	0	1	1

Результат ошибочен — сумма отрицательных чисел не может иметь положительный знак.

Из приведенных примеров нетрудно понять, что факт переполнения разрядной сетки может быть выявлен по следующему признаку: при равенстве знаковых разрядов слагаемых знаковый разряд суммы не совпадает со знаковым разрядом слагаемых.

При построении микропрограммы сложения равенство знаковых разрядов слагаемых будем определять следующим образом: занесем знаковый разряд 1-го слагаемого в младший разряд регистра T и затем прибавим к содержимому этого регистра значение знакового разряда 2-го слагаемого. При равенстве знаков слагаемых в младшем разряде регистра образуется 0.

Пусть одно из слагаемых хранится в регистре R_1, другое слагаемое— в регистре T. На рис. 5.22 приведена схема алгоритма операции сложения.

Микрокоманды МК1 и МК2 путем передачи 1-го слагаемого из регистра R_1 в аккумуляторный регистр АС и последующего сдвига влево содержимого АС выдвигают знаковый разряд слагаемого в триггер Ф БМУ, из которого он затем принимается в триггер С регистра признаков БМУ.

Микрокоманды МК3 и МК4 производят сдвиг влево содержимого триггера Т с приемом в младший разряд этого регистра знакового разряда 1-го слагаемого из триггера С; выдвигаемый из регистра Т знаковый разряд 2-го слагаемого передается в триггер Z регистра признаков БМУ.

Микрокоманды МК5 и МК6 формируют сумму по модулю 2 знаковых разрядов слагаемых путем их суммирования в младшем разряде регистра Т; полученное значение выдвигается из регистра Т и передается для хранения в триггер С. При этом равенству знаковых разрядов слагаемых будет соответствовать значение 0 в триггере С.

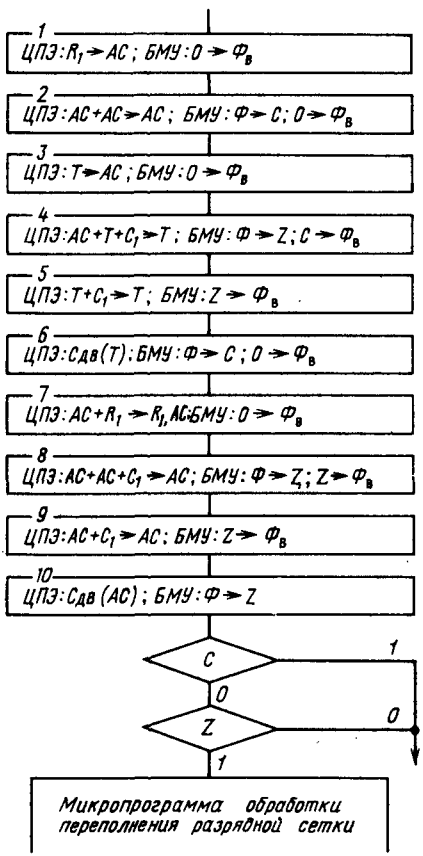


Рис. 5.22. Схема алгоритма операции сложения

Микрокоманда МК7 формирует сумму слагаемых в регистрах АС и R_1.

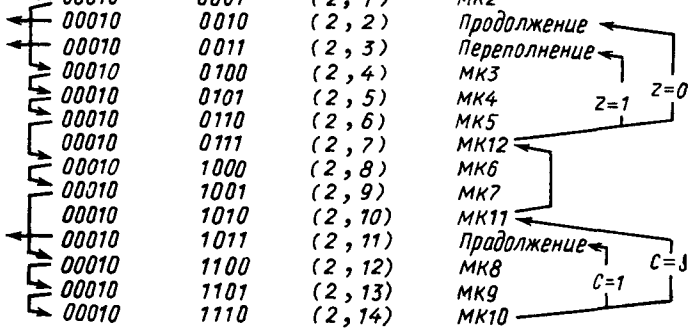
Микрокоманда МК8 осуществляет сдвиг влево хранящейся в АС суммы слагаемых и прием в младший разряд регистра знакового разряда 2-го слагаемого, который к данному моменту хранится в триггере Z; выдвигаемый из АС знаковый разряд суммы слагаемых передается в триггер Z.

Микрокоманды МК9 и МК10 производят суммирование в младшем разряде АС знаковых разрядов 2-го слагаемого и суммы, после чего полученное значение выдвигается из АС и передается в триггер Z. Получаемое в триггере Z значение равно нулю в случае совпадения знаковых разрядов суммы и 2-го слагаемого.

Комбинация полученных в триггерах С и Z значений С = 0 и Z = 1 соответствует сформулированному выше условию возникновения переполнения разрядной сетки. При этом осуществляется переход к специальной микропрограмме обработки переполнения. При всех других сочетаниях значений, возникающих

Таблица 5.17

Адрес строки	Адрес колонки	Десятичный адрес	Микрокоманда
00010	0000	(2, 0)	МК1
00010	0001	(2, 1)	МК2
00010	0010	(2, 2)	Продолжение
00010	0011	(2, 3)	Переполнение
00010	0100	(2, 4)	МК3
00010	0101	(2, 5)	МК4
00010	0110	(2, 6)	МК5
00010	0111	(2, 7)	МК12
00010	1000	(2, 8)	МК6
00010	1001	(2, 9)	МК7
00010	1010	(2, 10)	МК11
00010	1011	(2, 11)	Продолжение
00010	1100	(2, 12)	МК8
00010	1101	(2, 13)	МК9
00010	1110	(2, 14)	МК10



к данному моменту в триггерах C и Z, операция сложения считается завершенной, результат операции хранится в регистре R_1.

В табл. 5.17 показано возможное размещение микрокоманд в ячейках управляющей памяти и последовательность, в которой производится обращение к этим ячейкам памяти в процессе выполнения микропрограммы.

При размещении микрокоманд МК1 ... МК10 были оставлены свободными две пары ячеек с адресами (2,2), (2,3) и (2,10), (2,11), используемые для выполнения разветвлений по признакам C и Z.

После перехода по признаку C = 0 к ячейке с адресом (2,10) по алгоритму требуется выполнение разветвления по признаку Z. Однако разветвление в ячейках (2,2) и (2,3) может быть выполнено лишь из ячейки, имеющей в старшем разряде адреса колонки то же значение 0, что и у ячеек (2,2) и (2,3). Так как ячейка (2, 10) имеет в старшем разряде адреса колонки значение 1, то возникает необходимость размещения микрокоманды условного перехода по признаку Z в одной из ячеек с адресом колонки, меньшим 8. В качестве такой ячейки нами выбрана ячейка с адресом (2,7), которая также оставлена свободной при размещении первых десяти микрокоманд. Таким образом, микрокоманда МК11 служит для перехода к ячейке с адресом (2,7), а размещенная в этой ячейке микрокоманда МК12 реализует условный переход по признаку Z. Эти микрокоманды отсутствуют в схеме на рис. 5.21, их необходимость определяется в процессе размещения микрокоманд в управляющей памяти. Рассмотренная реализация условных переходов показана на рис. 5.23.

В данном варианте размещения микрокоманд для выполнения обоих условных переходов используются ячейки памяти, расположенные на одной и той же строке, что вызвало необходимость в дополнительной микрокоманде МК12. В табл. 5.18 показан другой вариант размещения той же микропрограммы в управляющей памяти. В этом варианте раз-

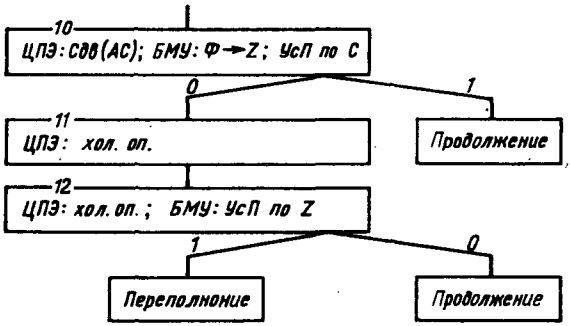


Рис. 5.23. Схема алгоритма двух последовательных разветвлений с использованием ячеек памяти в общей строке

мещения микропрограммы условные переходы выполняются в двух последовательных микрокомандах МК10 и МК11 (в предыдущем варианте эти условные переходы выполнялись в микрокомандах МК10 и МК12). При этом сокращаются общее число микрокоманд в микропрограмме и время ее выполнения. Однако, как видно из табл. 5.18, при таком выполнении условных переходов в двух последовательных микрокомандах необходимо пару ячеек для второго разветвления выбирать, переходя на другую строку. Соответствующая этому варианту схема разветвлений показана на рис. 5.24.

В табл. 5.19 приведена микропрограмма, соответствующая последнему варианту ее размещения в управляющей памяти.

Программирование на языке Ассемблера. До сих пор, записывая микрокоманды, мы пользовались языком кодовых комбинаций, единственно понятным для микропроцессора языком. Пользование этим языком вызывает трудности, связанные, во-первых, с необходимостью за-

Таблица 5.18

	Адрес строки	Адрес колонки	Десятичный адрес	Микрокоманда
	00010	0000	(2, 0)	МК1
	00010	0001	(2, 1)	МК2
	00010	0010	(2, 2)	МК3
	00010	0011	(2, 3)	МК4
	00010	0100	(2, 4)	МК5
	00010	0101	(2, 5)	МК6
	00010	0110	(2, 6)	МК7
	00010	0111	(2, 7)	МК8
	00010	1000	(2, 8)	МК9
	00010	1001	(2, 9)	МК10
C=0 →	00010	1010	(2, 10)	МК11
C=1 →	00010	1011	(2, 11)	Продолжение
	.....	.....		
Z=0 →	00011	1001	(3, 10)	Продолжение
Z=1 →	00011	1010	(3, 11)	Переполнение

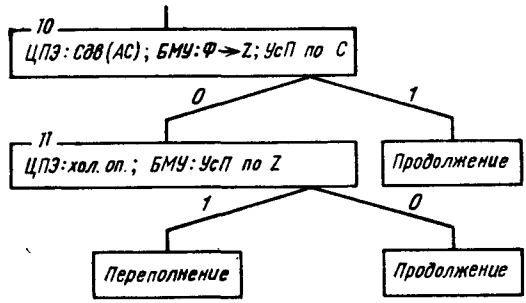


Рис. 5.24. Схема алгоритма двух последовательных разветвлений с использованием пар ячеек в различных строках

писи громоздких, трудно запоминаемых двоичных кодовых комбинаций и, во-вторых, со сложностью поиска ошибок в составленной микропрограмме из-за того, что записанная с помощью кодовых комбинаций микропрограмма оказывается трудно читаемой, плохо обозримой.

Микрокоманды на языке Ассемблера МПК К589 состоят из ряда обязательных полей, содержащих информацию, предназначенную для управления БМУ и ОУ, и дополнительных полей, вводимых программистом для управления выходными буферами ЦПЭ, оперативной памятью, устройствами ввода, устройствами вывода.

В табл. 5.20 показана запись на языке Ассемблера микропрограммы, приведенной на языке кодовых комбинаций в табл. 5.19.

Таблица 5.19

Адрес МК в ПЗУ		Поля МК БМУ		Поля МК ОУ		Пояснения
Адрес строки	Адрес колонки	$УА_3...УА_0$	$УФ_3 ... УФ_0$	$F_3...F_0$	К	
00010	0000	011 0001	00 11	000 1101	0	МК1
00010	0001	011 0010	00 10	000 1101	1	МК2
00010	0010	011 0011	00 11	000 1100	0	МК3
00010	0011	011 0100	01 01	000 1100	1	МК4
00010	0100	011 0101	10 11	011 1100	0	МК5
00010	0101	011 0110	00 10	000 1110	0	МК6
00010	0110	011 0111	00 11	000 0001	1	МК7
00010	0111	011 1000	10 01	000 1101	1	МК8
00010	1000	011 1001	10 11	011 1101	0	МК9
00010	1001	1010 010	00 01	000 1111	0	МК10
00010	1010	1011 011	00 11	110 0000	0	МК11
00010	1011	...	...	...	...	Продолжение
...	...					
00011	1010	...	...	...	...	Продолжение
00011	1011	...	...	...	...	Переполнение

В микрокомандах безусловного перехода в поле метки перехода указывается метка той микрокоманды, к которой осуществляется переход. В случае микрокоманды условного перехода в поле метки перехода приводятся метки двух микрокоманд, к которым осуществляется переход: на первом месте — метка микрокоманды, к которой производится переход при значении признака разветвления, равном нулю, на втором месте — метка микрокоманды, к которой происходит переход при значении признака, равном единице. Очевидно, не должно быть более одной микрокоманды с одной и той же меткой, в противном случае возникает неопределенность в выборе микрокоманды, к которой должеи быть произведен переход.

В поле вида перехода приводится мнемоническое обозначение вида перехода, осуществляемого БМУ при формировании адреса очередной микрокоманды.

В поле управления признаками приводится мнемоническое обозначение действий, связанных с выдачей и хранением признаков в БМУ.

В поле операции в ЦПЭ приводится мнемоническое обозначение операции, выполняемой в ОУ.

В поле операнда записывается имя регистра, который используется в качестве источника и (или) приемника информации.

Поле комментариев служит для записи любых пояснений смысла выполняемых по микрокоманде действий, которые облегчают чтение микропрограммы.

Приведенная форма записи микрокоманд на языке Ассемблера не единственна.

Микропрограмма сложения с использованием модифицированного кода. Микропрограмма сложения упрощается, если числа представляются в модифицированном коде. Особенность модифицированного

Таблица 5.20

Метка	Вид перехода	Метка перехода	Управление признаками	Операция в ЦПЭ	Операнд	Комментарий
	JCR		FF0 HCZ	ILR	RI	МК1
	JCR		FF0 STC	ALR	AC	МК2
	JCR		FF0 HCZ	ILR	T	МК3
	JCR		FFC STZ	ALR	T	МК4
	JCR		FFZ HCZ	INR	T	МК5
	JCR		FF0 STC	SRA	T	МК6
	JCR		FF0 HCZ	ALR	RI	МК7
	JCR		FFZ STZ	ALR	AC	МК8
	JCR		FFZ HCZ	INR	AC	МК9
	JCF	M1, M2	FF0 STZ	SRA	AC	МК10
M1:	JZF	M3, M4	FF0 HCZ	NOP		МК11
M2:	. . .	. . .	. . .	. . .	. . .	Продолжение
M3:	. . .	. . .	. . .	. . .	. . .	Продолжение
M4:	. . .	. . .	. . .	. . .	. . .	Переполнение

кода состоит в том, что он предусматривает использование двух знаковых разрядов с одинаковыми значениями. Несовпадение значений знаковых разрядов имеет место лишь в случае возникновения переполнения разрядной сетки, что и служит признаком, по которому выявляется это событие.

Пусть слагаемые хранятся в регистрах R_1 и АС. На рис. 5.25 приведена схема алгоритма сложения. Микрокоманда МК1 осуществляет сложение с занесением суммы в регистры R_1 и АС. Далее проводятся операции по проверке совпадения значений знаковых разрядов суммы. Для этого в микрокоманде МК2 предусматривается сдвиг влево содержимого АС с передачей выдвигаемого старшего знакового разряда в триггер Z регистра признаков БМУ. Затем в микрокоманде МК3 предусматривается повторный сдвиг влево содержимого АС с занесением в младший разряд АС хранящегося в триггере Z значения старшего знакового разряда; выдвигаемый из АС младший знаковый разряд суммы передается в триггер Z. Микрокоманда МК4 производит прибавление младшего знакового разряда суммы к содержимому младшего разряда АС. При этом в младшем разряде АС образуется сумма по модулю 2 знаковых разрядов суммы слагаемых, она микрокомандой МК5 выдвигается из АС и затем в микрокоманде МК6 используется в качестве признака, по которому выполняется разветвление. В случае несовпадения знаковых разрядов их сумма по модулю 2 равна 1 и происходит переход к микрокоманде, определяющей действия при возникновении переполнения разрядной сетки.

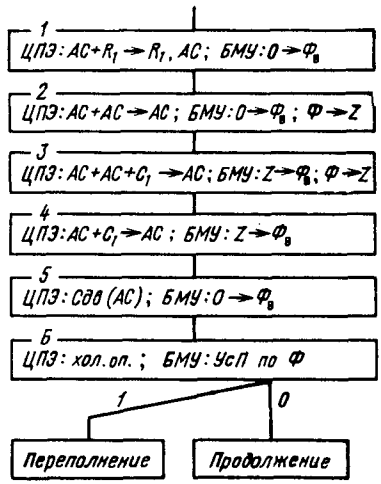


Рис. 5.25. Схема алгоритма сложения с использованием модифицированного кода

В табл. 5.21 приведено размещение микропрограммы в ячейках управляющей памяти, в табл. 5.22 и 5.23 — микропрограммы соответственно в кодовых комбинациях и на языке Ассемблера.

Таблица 5.21

Адрес строки	Адрес колонки	Десятичный адрес	Микрокоманда
00011	0000	(3, 0)	МК1
00011	0001	(3, 1)	МК2
00011	0010	(3, 2)	Продолжение
00011	0011	(3, 3)	Переполнение
00011	0100	(3, 4)	МК3
00011	0101	(3, 5)	МК4
00011	0110	(3, 6)	МК5
00011	0111	(3, 7)	МК6

Таблица 5.22

Адрес МК в ПЗУ		Поля МК БМУ		Поля МК ОУ		Пояснения
Адрес строки	Адрес колонки	$УА_6...УА_0$	$УФ_3...УФ_0$	$F_8...F_0$	K	
00011	0000	011 0001	00 11	000 0001	1	МК1
00011	0001	011 0100	00 11	000 1101	1	МК2
00011	0010	...	...	...	...	Продолжение
00011	0011	...	...	...	...	Переполнение
00011	0100	011 0101	10 01	000 1101	1	МК3
00011	0101	011 0110	10 11	011 1101	0	МК4
00011	0110	011 0111	00 11	000 1111	0	МК5
00011	0111	100 0011	00 11	110 0000	0	МК6

Таблица 5.23

Метка	Вид перехода	Метка перехода	Управление признаками	Операция ЦПЭ	Операнд	Комментарий
	JCR		FF0 HCZ	ALR	RI	МК1
	JCR		FF0 STZ	ALR	AC	МК2
	JCR		FFZ STZ	ALR	AC	МК3
	JCR		FFZ HCZ	INR	AC	МК4
	JCR		FF0 HCZ	SRA	AC	МК5
	JFL	M1, M2	FF0 HCZ	NOP		МК6
M1:	...	...	...	...	...	Продолжение
M2:	...	...	...	...	...	Переполнение

МИКРОПРОГРАММА ВЫЧИТАНИЯ

Будем считать, что отрицательные числа хранятся в микропроцессоре (а также в оперативной памяти) в дополнительном коде. Пусть уменьшаемым является содержимое регистра R_1, вычитаемым — содержимое аккумулятора АС. Очевидно, можно воспользоваться операцией сложения, если предварительно изменить на обратный знак вычитаемого. Так как вычитаемое представлено в дополнительном коде, то изменение знака потребует инвертирования всех разрядов, включая и знаковый, а затем прибавления единицы в младший разряд числа. Для вычитаемого, хранящегося в АС, такая операция может быть выполнена одной микрокомандой:

ЦПЭ: $AC + C_1 \rightarrow AC$; БМУ: $1 \rightarrow Ф_в$

Умножение чисел без знака. Покажем процесс умножения чисел без знака на примере умножения чисел 1011_2 и 1101_2:

```
      1 0 1 1   множимое
+     1 1 0 1   множитель
      1 0 1 1   1-е частичное произведение
+   0 0 0 0     2-е частичное произведение
  1 0 1 1       3-е частичное произведение
1 0 1 1         4-е частичное произведение
1 0 0 0 1 1 1 1 произведение
```

Схема умножения предусматривает последовательный анализ разрядов множителя и формирование частичных произведений, каждое из которых определяется значением соответствующего разряда множителя: если в разряде множителя содержится 1, то частичное произведение равно множимому, если в разряде множителя содержится 0, то частичное произведение равно нулю. Затем произведение находится суммированием соответствующим образом сдвинутых относительно друг друга частичных произведений.

В дальнейшем будем рассматривать способ выполнения операции, в котором анализ множителя проводится, начиная с его младшего разряда, и соответственно формирование частичных произведений начинается с 1-го частичного произведения.

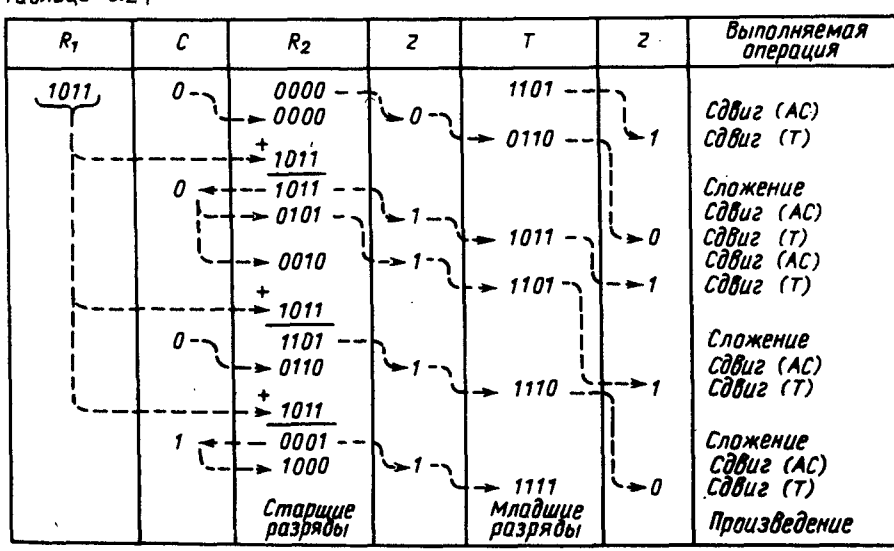
Если сомножители — n-разрядные числа, то процесс получения произведения состоит из n-кратного повторения цикла, включающего в себя прибавление очередного частичного произведения к сумме предыдущих частичных произведений и сдвиг полученной суммы на один разряд вправо (и, таким образом, подготовка этой суммы к прибавлению к ней частичного произведения в очередном повторении цикла). Заметим, что число разрядов в произведении равно $2n$. Однако в каждом акте суммирования n-разрядное частичное произведение прибавляется лишь к n старшим разрядам сдвинутой текущей суммы частичных произведений. Из этого следует, во-первых, что младшие разряды этой суммы остаются неизменными, не принимая участия в процессе суммирования, и для их хранения может быть использован отдельный регистр младших разрядов сумм, и во-вторых, что может использоваться n-разрядный сумматор и результат суммирования может быть помещен в n-разрядный регистр старших разрядов суммы (следует только предусмотреть хранение переноса, возникающего при суммировании из старшего разряда, и при сдвиге суммы вправо его передачу в старший разряд суммы).

Рассмотрим реализацию операции в микропроцессорном устройстве. Процесс получения произведения чисел 1011_2 и 1101_2 иллюстрируется табл. 5.24.

Здесь хранение множимого предусматривается в регистре R_1, множителя — в регистре Т. Регистр Т одновременно используется для хранения младших разрядов произведения, не участвующих в последующих операциях суммирования частичных произведений. Старшие разряды произведения формируются в регистре R_2. Значения отдельных разрядов множителя (начиная с его младшего разряда) анализируются путем последовательного сдвига вправо содержимого регистра Т. Выдвигаемый из регистра младший разряд множителя принимается в триггер Z БМУ, и по содержимому этого триггера осуществляется разветвление процесса: при Z = 1 к содержимому регистра R_2 прибавляется множимое, при Z = 0 акт суммирования пропускается. Возникающий в процессе суммирования перенос помещается в триггер С регистра признаков БМУ. При сдвиге содержимого АС в старший разряд регистра принимается содержимое триггера С, а выдвигаемый из регистра младший разряд запоминается в триггере Z. Затем при сдвиге содержимого регистра Т в старший разряд регистра принимается содержимое триггера Z, выдвигаемый из регистра младший разряд заносится в тот же триггер Z.

Не каждая из указанных в таблице операций может быть выполнена одной микрокомандой. Так, чтобы хранящееся в регистре R_1 множимое прибавить к содержимому регистра R_2, потребуется предварительно передать содержимое регистра R_1 в аккумулятор АС, после чего можно суммировать содержимое регистров R_2 и АС. Невозможно также непосредственно выполнить сдвиг вправо содержимого регистра R_2. Для получения в регистре R_2 сдвинутого значения потребуется дубли-

Таблица 5.24



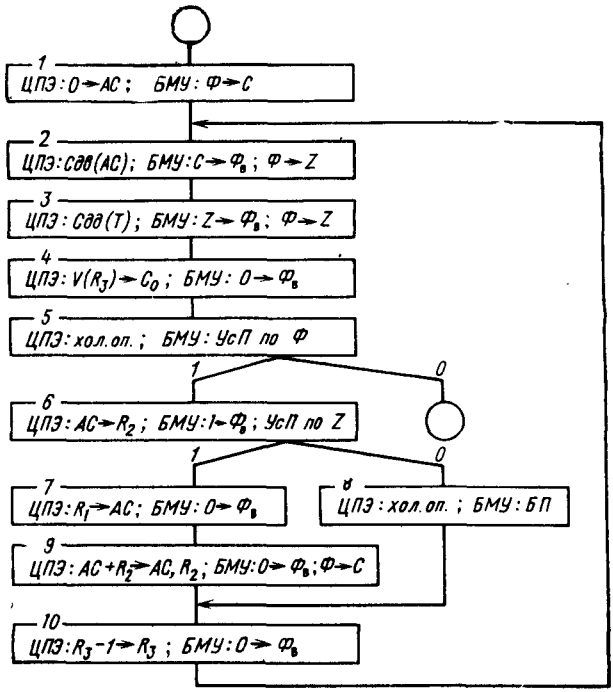


Рис. 5.26. Схема алгоритма операции кодового умножения

ровать содержимое R_2 в АС, произвести сдвиг содержимого АС и затем передать содержимое АС в регистр R_2.

Кроме того, при построении схемы алгоритма потребуется организовать счетчик числа повторений цикла, при каждом прохождении которого формируется очередное частичное произведение и оно прибавляется к сумме предыдущих частичных произведений.

На рис. 5.26 приведена схема алгоритма рассматриваемой операции умножения.

Счетчик числа повторений цикла выполнен здесь на регистре R_3, в который в качестве начального значения помещается число, равное количеству разрядов в сомножителях.

Микрокоманда МК1 производит установку в АС нулевого значения, которое затем при первом прохождении цикла микрокомандой МК6 передается в регистр R_2 и в последнем устанавливается исходное значение. В той же микрокоманде МК1 производится установка нулевого значения в триггере С регистра признаков БМУ.

Микрокоманды МК2 и МК3 производят сдвиг содержимого АС и регистра Т с соответствующим табл. 5.24 использованием содержимого триггеров С и Z регистра признаков БМУ; выдвигаемый из регистра Т младший разряд передается в триггер Z.

Микрокоманда МК4 производит проверку на нуль содержимого счетчика R_3, по результатам этой проверки микрокоманда МК5 выполняет разветвление. В случае нулевого значения содержимого регистра R_3 происходит выход из цикла и переход к следующему участку микропрограммы, начинающемуся с микрокоманды, обозначеиной *Продолжение*. Если содержимое счетчика не достигло нулевого значения, то происходит переход к микрокоманде МК6, в которой выполняется разветвление по содержимому триггера Z, хранящего очередной выдвинутый из триггера T разряд множителя. В случае $Z = 1$ в ветви из микрокоманд МК7 и МК9 выполняется операция прибавлеиия взятого из регистра R_1 множимого к сумме предыдущих частнчных произведений, старшие разряды которой формируются в регистре R_2. При $Z = 0$ микрокоманда МК8 производит безусловный переход к МК10 и, таким образом, операция суммирования пропускается.

После каждого прохождения цикла в МК10 производится вычитание единицы из содержимого счетчика на регистре R_3 и безусловный переход к МК2.

На рис. 5.27 показано размещение микропрограммы в управляющей памяти. Наличие в схеме алгоритма двух непосредственно друг за другом следующих микрокоманд условного перехода (микрокоманд МК5 и МК6) требует их размещения в ячейках с различающимися адресами строк. Как видно из таблицы, в этом случае достигается возможность после выполнения разветвления по признаку Ф в пару ячеек с адресами (3, 2) и (3,3) из ячейки с адресом (3,3) осуществлять разветвление по признаку Z в пару ячеек с адресами (4, 2) и (4, 3),

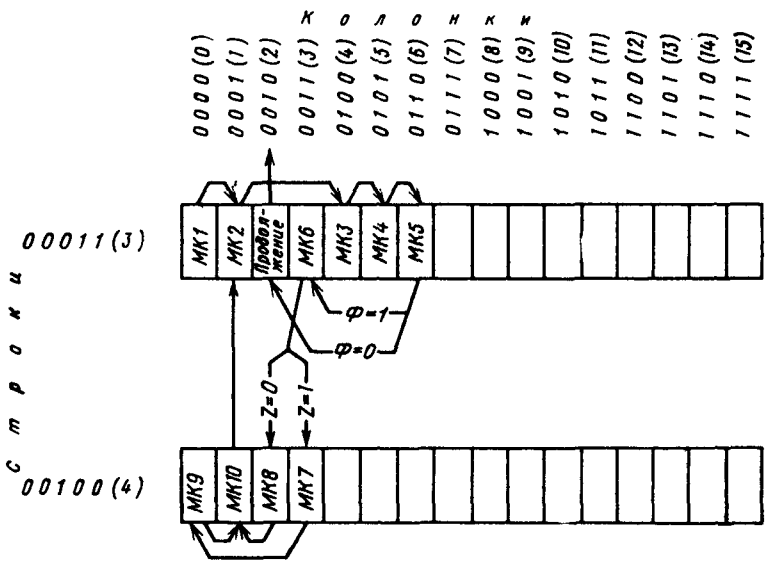


Рис. 5.27. Размещенне микропрограммы умножеиия в управляющей памяти

Таблица 5.25

Адрес МК в ПЗУ		Поля МК БМУ		Поля МК ОУ		Пояснения
Адрес строки	Адрес колонки	$УА_6...УА_0$	$УФ_3...УФ_0$	$F_6...F_0$	К	
00011	0000	011 0001	00 10	100 1101	0	МК1
00011	0001	011 0100	01 01	000 1111	0	МК2
00011	0010	...	...	...	...	Продолжение
00011	0011	1011 100	11 11	010 0010	1	МК6
00011	0100	011 0101	10 01	000 1110	0	МК3
00011	0101	011 0110	00 11	101 0011	1	МК4
00011	0110	100 0011	00 11	110 0000	0	МК5
...	...					
00100	0000	011 0001	00 10	000 0010	1	МК9
00100	0001	00 00011	00 11	001 0011	1	МК10
00100	0010	011 0001	00 11	110 0000	0	МК8
00100	0011	011 0000	00 11	000 0001	0	МК7

расположенную в памяти в другой строке. Размещение всей микропрограммы в ячейках одной строки потребовало бы введения между микрокомандами условных переходов дополнительной микрокоманды безусловного перехода.

В табл. 5.25 приведена микропрограмма, представленная в кодовых комбинациях, в табл. 5.26 — та же микропрограмма на языке Ассемблера.

Алгебраическое умножение. Алгоритм с отделением знаковых разрядов. Пусть сомножители представлены в прямом коде и хранятся в регистрах R_1 и T в форме с фиксированной точкой. Старший разряд чисел является знаковым. Алгоритм умножения включает в себя две операции: операцию над знаковыми разрядами для определения знака произведения и операцию над модулями со-

Таблица 5.26

Метка	Вид перехода	Метка перехода	Управление признаками	Операция ЦПЭ	Операнд	Комментарии
	JCR		FF0 STC	CLR	AC	МК1
M6:	JCR		FFC STZ	SRA	AC	МК2
	JCR		FFZ STZ	SRA	T	МК3
	JCR		FF0 HCZ	TZR	R3	МК4
	JFL	M1, M2	FF0 HCZ	NOP		МК5
M1:	...	...	...	...	...	Продолжение
M2:	JZF	M3, M4	FF1 HCZ	SDR	R2	МК6
M3:	JCR	M5	FF0 HCZ	NOP		МК8
M4:	JCR		FF0 HCZ	ILR	R1	МК7
	JCR		FF0 STC	ALR	R2	МК9
M5:	JCC	M6	FF0 HCZ	DSM	R3	МК10

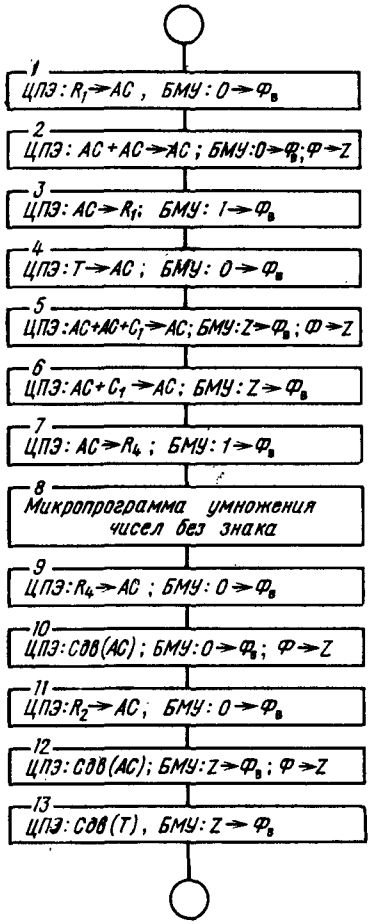


Рис. 5.28. Схема алгоритма алгебраического умножения с отделением знаковых разрядов

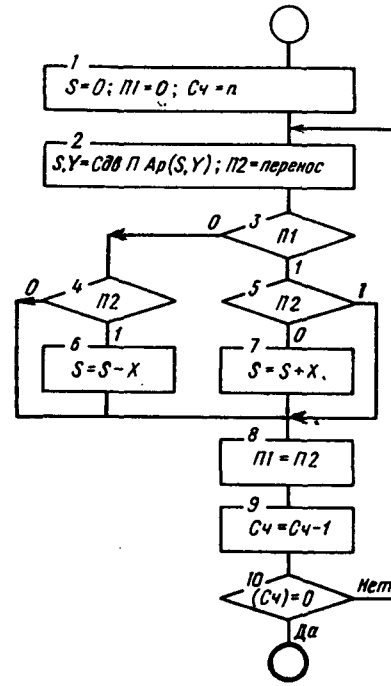


Рис. 5.29. Схема алгоритма умножения Бута в содержательных обозначениях

множителей для нахождения модуля произведения. Для проведения этих операций необходимо выделить из регистров R_1 и Т знаковые разряды путем сдвига их содержимого влево и над знаковыми разрядами провести операцию суммирования по модулю 2; содержимое регистров R_1 и Т будет представлять собой модули сомножителей, операция умножения которых может быть проведена как операция умножения чисел без знака, рассмотренная выше.

На рис. 5.28 показана схема алгоритма умножения. В микрокомандах МК1, МК2, и МК3 производятся: передача сомножителя из регистра R_1 в АС; путем сдвига содержимого АС выделение знакового разряда сомножителя и передача его для хранения в триггер Z БМУ; передача сомножителя, лишенного знакового разряда, из АС в регистр R_1.

В микрокомандах МК4...МК7 переданный из регистра Т в АС второй сомножитель сдвигается влево, в младший разряд заносится хра-

нящийся в триггере Z знаковый разряд первого сомножителя, а выдвигаемый из АС знаковый разряд второго сомножителя передается для хранения в триггер Z; далее к содержимому АС прибавляется выбираемый из триггера Z знаковый разряд второго сомножителя и результат, в котором младший разряд есть знаковый разряд произведения, передается для хранения в регистр R_4.

Затем производится по ранее рассмотренному алгоритму умножение чисел без знака с числом повторения цикла, на единицу меньшим числа разрядов процессора. При этом содержимое регистра R_1, в котором хранится модуль первого сомножителя, умножается на содержимое регистра Т без учета его старшего разряда, в котором содержится знак. Получаемый в результате умножения модуль произведения хранится в регистре R_2 (старшие разряды произведения) и в регистре Т (младшие разряды произведения).

Последующие микрокоманды МК9...МК13 предназначены для вписывания в произведение его знакового разряда. В МК9 и МК10 сдвигом вправо из содержимого регистра R_4 выдвигается младший разряд, представляющий собой знаковый разряд произведения, и передается в триггер Z. В МК11 и МК12 производится сдвиг вправо хранящихся в регистре R_2 старших разрядов произведения и прием знакового разряда произведения. Выдвигаемый при сдвиге младший разряд микрокомандой МК13 вписывается через триггер Z в старший разряд регистра Т.

Алгоритм без отделения знаковых разрядов. Одним из эффективных алгоритмов умножения является *алгоритм Бута*. Он не предусматривает выполнения раздельных операций над знаковыми разрядами и модулями сомножителей. В результате выполнения действий по этому алгоритму в получаемом результате образуется произведение со знаковым разрядом.

Таблица 5.27

X	$-X_{доп}$	S	Y	П2	П1	Выполняемая операция
0 1011	0 0101	0 0000	10011		0	
		0 0000	01001	1	0	Сдвиг (S,Y)
		+1 0101				
		1 0101				Вычитание
		1 1010	10100	1	1	Арифметический сдвиг (SY)
		1 1101	01010	0	1	Арифметический сдвиг (SY)
		+0 1011				
		0 1000				Сложение
		0 0100	00101	0	0	Арифметический сдвиг (SY)
		0 0010	00010	1	0	Арифметический сдвиг (SY)
		+1 0101				
		1 0111	00010			Дополнительный код произведения
		1 1000	$1111_2 = -143_{10}$			Прямой код произведения

На рис. 5.29 показана схема алгоритма в содержательных обозначениях (X — множимое, Y — множитель, S — старшие разряды суммы частичных произведений, Сч — счетчик числа повторений цикла, в который в качестве начального значения заносится число разрядов сомножителей, включая и знаковый разряд; П1, П2 — разряды множителя, выдвигаемые при его сдвиге соответственно в предыдущем и текущем повторениях цикла).

В блоке *2* производится арифметический сдвиг вправо S, Y, рассматриваемых как единое число, в котором S образует его старшие, а Y — младшие разряды. Значение появляющегося из младшего разряда Y переноса присваивается П2.

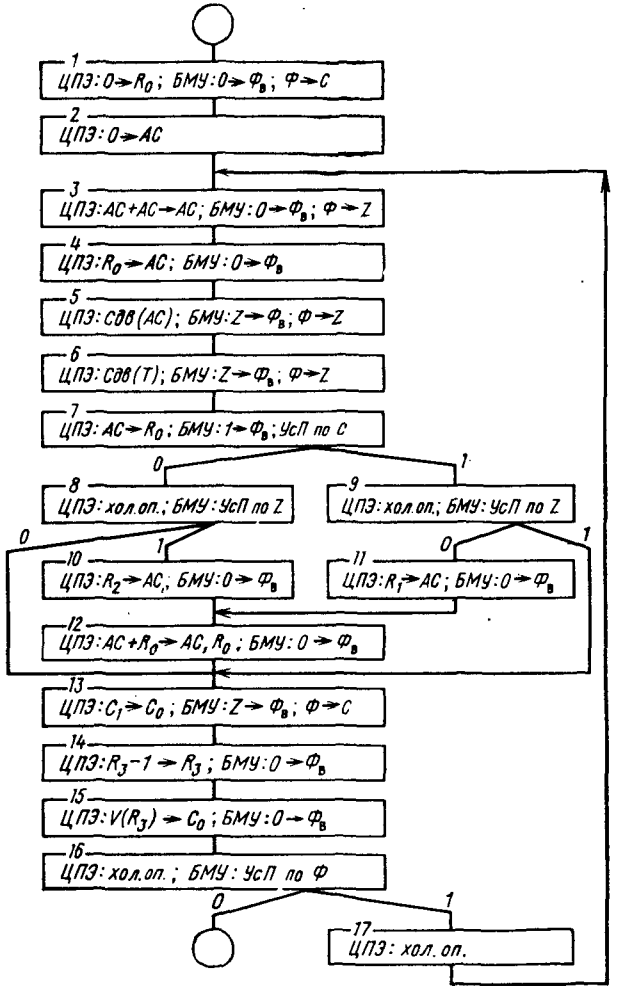


Рис. 5.30. Схема алгоритма умножения Бута в микрокомандах

Далее в блоках *3...7* организуется выполнение действий по следующему правилу:

П1	П2	Выполняемое действие
0	0	—
0	1	S=S—X
1	0	S=S+X
1	1	—

При комбинациях значений П1 и П2, равных 00 и 11, содержимое S не изменяется; при комбинации 01 значение S уменьшается на X, а при комбинации 10 — увеличивается на X.

В последующих блоках производится подготовка к повторению цикла. Значение П2 передается П1, а содержимое счетчика уменьшается на единицу, и по результату проверки содержимого счетчика на нуль выясняется необходимость повторения цикла.

После выхода из цикла в S образуется знаковый разряд и группа старших разрядов произведения, в (n — 1) разрядах Y — группа младших разрядов произведения.

Следует отметить, что отрицательные сомножители должны представляться в дополнительном коде; если произведение оказывается отрицательным числом, то оно оказывается представленным в дополнительном коде.

Покажем описанные действия на примере умножения чисел $X = 01011_2\ (11_{10})$ и $Y_{доп} = 10011_2\ (-13_{10})$ (выделенный полужирным рязряд — знаковый, второй сомножитель представлен в дополнительном коде). Проводимые при умножении действия иллюстрируются табл. 5.27.

На рис. 5.30 приведена схема реализации данного алгоритма в микропроцессорном устройстве.

Алгоритм построен в предположении, что в регистре R_1 хранится дополнительный код множимого, в регистре R_2 — дополнительный код взятого с обратным знаком множимого, в регистр T помещен дополнительный код множителя. Регистр R_0 используется для формирования в нем старших разрядов произ-

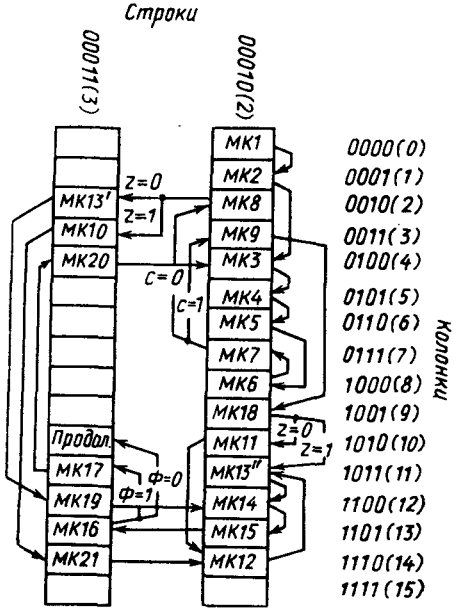


Рис. 5.31. Размещение микропрограммы умножения по алгоритму Бута в управляющей памяти

ведения. Для хранения признака П1 используется триггер С и для хранения признака П2 — триггер Z регистра признаков БМУ.

Микрокоманды МК1 и МК2 осуществляют запись начального нулевого значения в регистры R_0 и АС и триггер С. Микрокоманды МК3... ...МК6 выполняют операцию арифметического сдвига вправо содержимого пары регистров АС — Т; МК3 удвоением содержимого АС выдвигает из этого регистра знаковый разряд, передавая его для хранения в триггер Z, МК4 восстанавливает число в АС; МК5 сдвигает вправо содержимое АС, при этом знаковый разряд из триггера Z вдвигается в старший разряд АС, а выдвигаемый из АС младший разряд принимается в триггер Z; МК6 осуществляет сдвиг вправо содержимого регистра Т, при этом в старший разряд принимается хранящийся в триггере Z выдвинутый из АС младший разряд, а выдвигаемый из регистра Т младший разряд принимается в триггер Z и служит признаком П2.

Микрокоманды МК7...МК12 соответствуют блокам *3...7* схемы алгоритма на рис. 5.29. Вычитание множителя производится передачей в АС из регистра R_2 множителя с измененным на обратный знаком (МК10) и последующим сложением содержимого регистров АС и R_0

Таблица 5.28

Адрес МК в ПЗУ		Поля МК БМУ		Поля МК ОУ		Пояснения
Адрес строки	Адрес колонки	$УА_5...УА_0$	$УФ_3...УФ_0$	$F_5...F_0$	К	
00010	0000	011 0001	00 10	100 0000	0	МК1
00010	0001	011 0100	00 11	100 1101	0	МК2
00010	0010	1011 011	00 11	110 0000	0	МК8
00010	0011	011 1001	00 11	110 0000	0	МК9
00010	0100	011 0101	00 01	000 1101	1	МК3
00010	0101	011 0110	00 11	000 0000	0	МК4
00010	0110	011 1000	10 01	000 1111	0	МК5
00010	0111	1010 010	11 11	010 0000	1	МК7
00010	1000	011 0111	10 01	000 1110	0	МК6
00010	1001	1011 010	00 11	110 0000	0	МК18
00010	1010	011 1110	00 11	000 0001	0	МК11
00010	1011	011 1100	10 10	110 0000	0	МК13″
00010	1100	011 1101	00 11	001 0011	1	МК14
00010	1101	00 00011	00 11	101 0011	1	МК15
00010	1110	011 1011	00 11	000 0000	1	МК12
00010	1111					
00011	0000					
00011	0001					
00011	0010	011 1100	10 10	110 0000	0	МК13′
00011	0011	011 1110	00 11	000 0010	0	МК10
00011	0100	00 00010	00 11	110 0000	0	МК20
...	...					
00011	1010	...	...	...	...	Продолжение
00011	1011	011 0100	00 11	110 0000	0	МК17
00011	1100	00 00010	00 11	110 0000	0	МК19
00011	1101	100 0011	00 11	110 0000	0	МК16
00011	1110	00 00010	00 11	110 0000	0	МК21

(МК12); в случае прибавления множителя в АС передается множимое из регистра R_1 (МК11).

В МК13 производится выдача признака П2 из триггера Z на выход $Ф_в$ БМУ, отсюда он поступает на вход C_1 ОУ и в ОУ проходит на выход переноса C_0, далее он принимается в триггер Ф БМУ и затем передается в триггер С, где он служит уже признаком П1. Таким образом реализуется операция П1 = П2, предусмотренная в блоке *8* схемы алгоритма на рис. 5.29.

Микрооперация МК14 уменьшает на единицу содержимое счетчика, выполненного на регистре R_3; МК15 выполняет операцию дизъюнкции над разрядами содержимого регистра R_3 для его проверки на равенство нулю; МК16 осуществляет условный переход по содержимому триггера Ф и в случае, если содержимое регистра R_3 оказывается не равным нулю, МК17 осуществляет безусловный переход на начало цикла (к МК3).

На рис. 5.31 показаны размещение микропрограммы в управляющей памяти и последовательность переходов в процессе исполнения микропрограммы. В процессе размещения микропрограммы обнаруживается необходимость в дополнительных четырех микрокомандах: МК18, МК19, МК20 и МК21, осуществляющих безусловные переходы в тех случаях, когда приведенный на рис. 5.30 алгоритм требует таких переходов, при которых должны быть изменены адреса и строки и колонкн одновременно.

Таблица 29

Метка	Вид перехода	Метка перехода	Управление признаками	Операция ЦПЭ	Операнд	Комментарии
	JCR		FF0 STC	CLR	R0	MK1
	JCR		FF0 HCZ	CLR	AC	MK2
M13:	JCR		FF0 STZ	ALR	AC	MK3
	JCR		FF0 HCZ	ILR	R0	MK4
	JCR		FFZ STZ	SRA	AC	MK5
	JCR		FFZ STZ	SRA	T	MK6
	JCF	M1, M2	FF1 HCZ	SDR	R0	MK7
M1:	JCR	M3, M4	FF0 HCZ	NOP		MK8
M2:	JZF	M5	FF0 HCZ	NOP		MK9
M4:	JCR	M6	FF0 HCZ	ILR	R2	MK10
M5:	JZF	M7, M8	FF0 HCZ	NOP		MK18
M7:	JCR	M9	FF0 HCZ	ILR	R1	MK11
M6:	JCC		FF0 HCZ	NOP		MK21
M9:	JCR		FF0 HCZ	ALR	R0	MK12
M8:	JCR	M10	FFZ STC	NOP		MK13″
M3:	JCR		FFZ STC	NOP		MK13′
	JCC		FF0 HCZ	NOP		MK19
M10:	JCR		FF0 HCZ	DSM	R3	MK14
	JCC		FF0 HCZ	TZR	R3	MK15
	JZF	M11, M12	FF0 HCZ	NOP		MK16
M11:	. . .		. . .	. . .		Продолжение
M12:	JCR		FF0 HCZ	NOP		MK17
	JCC	M13	FF0 HCZ	NOP		MK20

В табл. 5.28 приведена микропрограмма в кодовых комбинациях.
В табл. 5.29 приведена микропрограмма, представленная на языке Ассемблера.

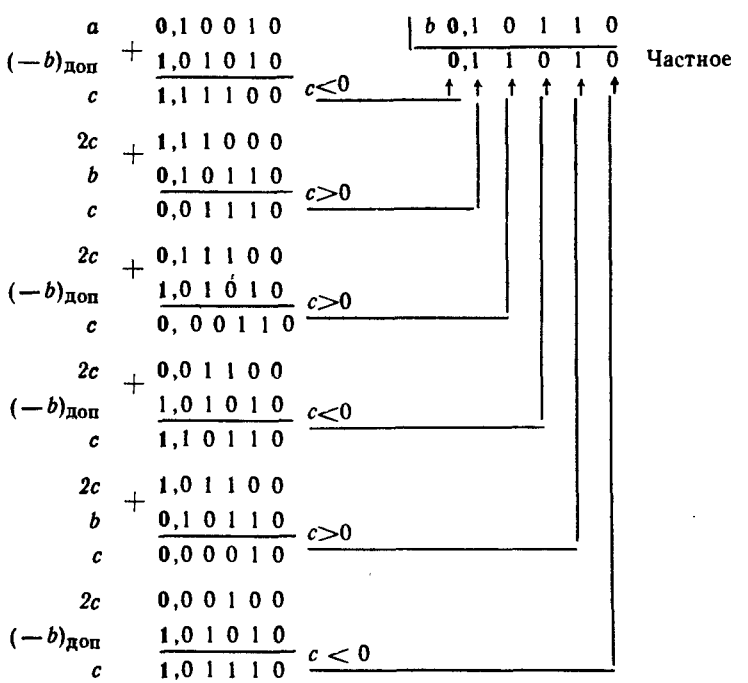
МИКРОПРОГРАММА ДЕЛЕНИЯ

Рассмотрим операцию алгебраического деления чисел, представленных в форме с фиксированной точкой. При этом выполнение операции содержит действия, связанные с определением знака частного, и действия, связанные с определением модуля частного. Знак частного может быть найден тем же приемом, что и знак произведения в рассмотренной выше микропрограмме алгебраического умножения с отделением знаковых разрядов. Поэтому ниже рассматривается лишь нахождение модуля частного.

На рис. 5.32 показана построенная в содержательных обозначениях схема алгоритма нахождения частного положительных чисел a и b.

Покажем выполнение операции на примере. Пусть после отделения знаковых разрядов модули делимого и делителя представляются числами $a = 0,\ 10010_2$ и $b = 0,\ 10110_2$.

Встречающуюся в алгоритме операцию вычитания числа b заменим прибавлением числа $-b$, представленного в дополнительном коде: $(-b)_{доп} = 1,\ 01010$.



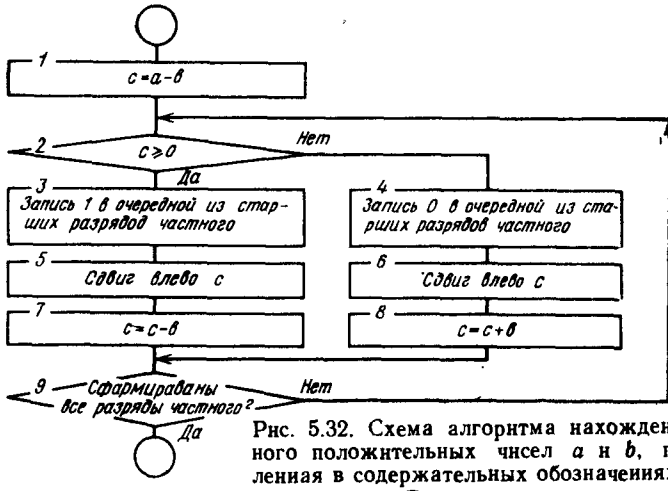


Рис. 5.32. Схема алгоритма нахождения частного положительных чисел a и b, представленная в содержательных обозначениях

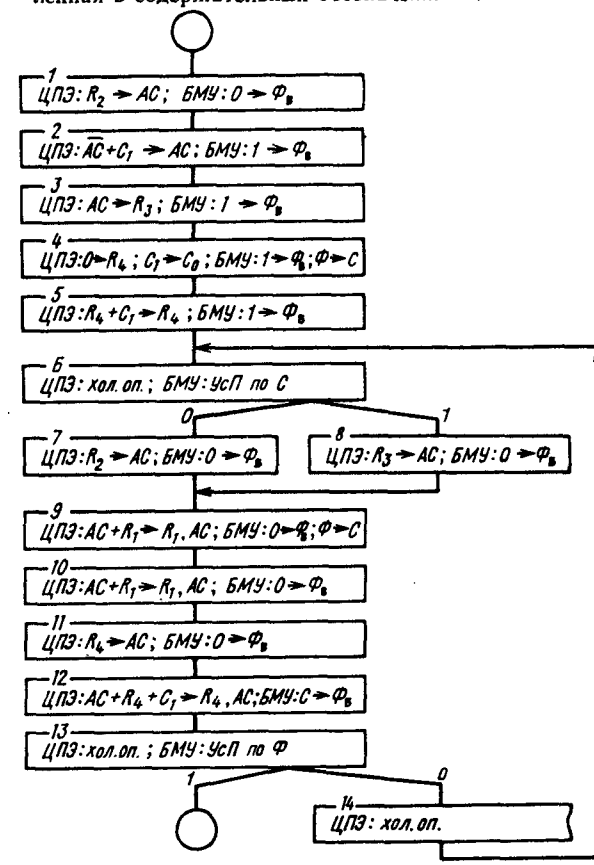


Рис. 5.33. Схема алгоритма деления в микрокомандах

В табл. 5.30 показан процесс реализации этого примера в микропроцессорном устройстве. Для хранения делимого (a) и возникающих остатков использован регистр R_1; в регистрах R_2 и R_3 хранятся соответственно делитель (b) и дополнительный код взятого с обратным знаком делителя ($-b_{доп}$). Частное формируется в регистре R_4.

Предусматриваемое блоком 2 алгоритма на рис. 5.32 разветвление будем осуществлять по признаку, в качестве которого используем перенос, возникающий при формировании очередного остатка. Из показанного выше примера видно, что очередной остаток получается всегда в результате суммирования чисел с разными знаками. При таком суммировании перенос из знакового разряда будет равен 1 в случае образования положительного остатка и равен 0 в случае образоваиия отрицательного остатка. Заметим, что значение очередного разряда частного совпадает со значением этого переноса из знакового разряда прн формировании остатка.

Запись формируемых разрядов частного будем осуществлять в младший разряд частного в регистре R_4. При этом перед записью очередного

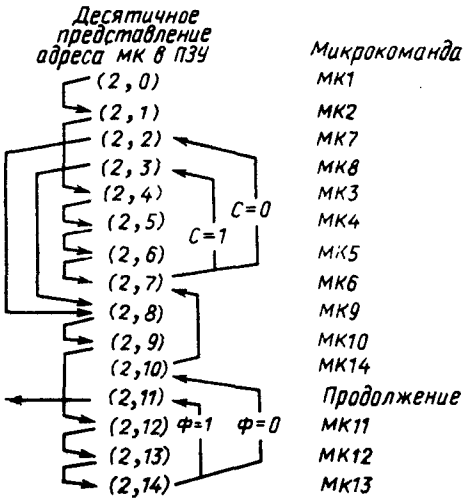
Таблица 5.30

R_2	R_3	C	R_1	Φ	R_4	Выполняемая операция
0, 10110	1, 01010		0, 10010		000001	
			+ 1, 01010			
		0	1, 11100			Вычитание
			1, 11000			Сдвиг влево (R_1)
				0	000010	Сдвиг влево (R_4)
			+ 0, 10110			
		1	0, 01110			Суммирование
			0, 11100			Сдвиг влево (R_1)
				0	000101	Сдвиг влево (R_4)
			+ 1, 01010			
		1	0, 00110			Вычитание
			0, 01100			Сдвиг влево (R_1)
				0	001011	Сдвиг влево (R_4)
			+ 1, 01010			
		0	1, 10110			Вычитание
			1, 01100			Сдвиг влево (R_1)
				0	010110	Сдвиг влево (R_4)
			+ 0, 10110			
		1	0, 00010			Суммирование
			0, 00100			Сдвиг влево (R_1)
				0	101101	Сдвиг влево (R_4)
			+ 1, 01010			
		0	1, 01110			Вычитание
			0, 11100			Сдвиг влево (R_1)
				1	0, 11010	Сдвиг влево (R_4)

разряда частного содержимое регистра R_4 должно быть сдвинуто на один разряд влево.

Проверку на окончание операции, осуществляемую блоком *9* алгоритма на рис. 5.32, можно выполнить, организовав счет числа повторений цикла. Однако такая проверка проще реализуется следующим образом. После обнуления содержимого регистра R_4 занесем в его младший разряд единицу. После каждого сдвига содержимого регистра эта единица будет продвигаться по регистру влево. При сдвиге, в процессе которого происходит запись в регистр последнего разряда частного, из регистра R_4 будет выдвинута единица, что и будет служить признаком окончания выполнения операции.

Таблица 5.31



Рассмотренной последовательности действий в процессе выполнения операции деления соответствует показанная на рис. 5.33 схема алгоритма, представленная в микрокомандах.

Таблица 5.32

Адрес МК в ПЗУ		Поля МК БМУ		Поля МК ОУ		Пояснения
Адрес строки	Адрес колонки	$УА_6...УА_0$	$УФ_3...УФ_0$	$F_6..F_0$	К	
00010	0000	011 0001	00 11	000 0010	0	МК1
00010	0001	011 0100	11 11	001 1111	0	МК2
00010	0010	011 1000	00 11	000 0010	0	МК7
00010	0011	011 1000	00 11	000 0011	0	МК8
00010	0100	011 0101	11 11	010 0011	1	МК3
00010	0101	011 0110	11 10	100 0100	0	МК4
00010	0110	011 0111	11 11	011 0100	0	МК5
00010	0111	1010 010	00 11	110 0000	0	МК6
00010	1000	011 1001	00 10	000 0001	1	МК9
00010	1001	011 1100	00 11	000 0001	1	МК10
00010	1010	011 0111	00 11	110 0000	0	МК14
00010	1011	...	...	...	...	Продолжение
00010	1100	011 1101	00 11	000 0100	0	МК11
00010	1101	011 1110	01 11	000 0100	1	МК12
00010	1110	100 0010	00 11	110 0000	0	МК13

Таблица 5.33

Метка	Вид перехода	Метка перехода	Управление признаками	Операция ЦПЭ	Операнд	Комментарии
	JCR		FF0 HCZ	ILR	R2	МК1
	JCR		FF1 HCZ	CIA	AC	МК2
	JCR		FF1 HCZ	SDR	R3	МК3
	JCR		FF1 STC	CLR	R4	МК4
	JCR		FF1 HCZ	INR	R4	МК5
M6:	JCF	M1, M2	FF0 HCZ	NOP		МК6
M1:	JCR	M3	FF0 HCZ	ILR	R2	МК7
M2:	JCR	M3	FF0 HCZ	ILR	R3	МК8
M3:	JCR		FF0 STC	ALR	R1	МК9
	JCR		FF0 HCZ	ALR	R1	МК10
	JCR		FF0 HCZ	ILR	R4	МК11
	JCR		FFC HCZ	ALR	R4	МК12
	JFL	M4, M5	FF0 HCZ	NOP		МК13
M4:	JCR	M6	FF0 HCZ	NOP		МК14
M5:	. . .	. . .	. . .	. . .		Продолжение

В табл. 5.31, 5.32 и 5.33 приведены соответственно размещение микропрограммы в управляющей памяти, микропрограмма в кодовых комбинациях и микропрограмма на языке Ассемблера.

ПОДПРОГРАММЫ

При построении микропрограммы в ней могут обнаружиться однотипные участки, предусматривающие выполнение операции одного и того же типа. Например, в нескольких точках микропрограммы может встретиться необходимость выполнения операции умножения. Такие повторяющиеся операции могут быть оформлены в виде отдельных микропрограмм — подпрограмм. При этом в соответствующих точках основной программы предусматривается обращение к таким подпрограммам с возвратом после окончания выполнения подпрограммы в ту же точку основной микропрограммы, в которой произошел выход из нее при обращении к подпрограмме.

Процесс обращения к подпрограмме иллюстрирует рис. 5.34. Пусть в основной микропрограмме после выполнения микрокоманды $МК_m$, считанной по адресу A_m, должна выполняться подпрограмма, состоящая из последовательности микрокоманд $МК_1^п$ ... $МК_k^п$, размещенных в ячейках управляющей памяти с адресами $B_1 ... B_k$.

Переход к подпрограмме реализуется просто. Для этого достаточно в микрокоманде $МК_m$ предусмотреть переход от ячейки с адресом A_m к ячейке с адресом B_1. Сложнее обеспечивается возврат в основную микропрограмму, т. е. после выполнения микрокоманды $МК_k^п$, взятой из ячейки с адресом B_k, произвести выбор очередной микрокоманды из ячейки A_{m+1}. Для этого необходимо при каждом выходе из основной

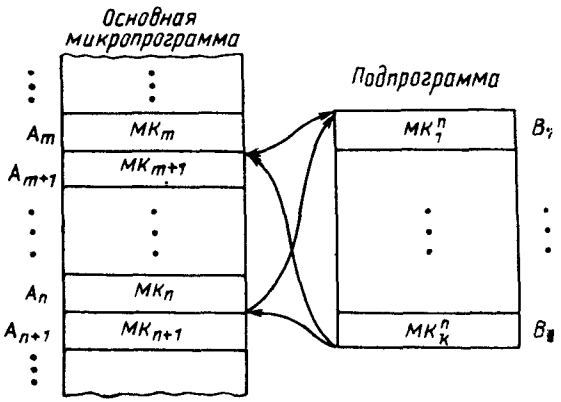


Рис. 5.34. Иллюстрация процесса обращения к подпрограмме

микропрограммы запоминать адрес ячейки, к которой должен производиться возврат после завершения выполнения подпрограммы.

Будем считать, что адрес A_{m+1} микрокоманды $МК_{m+1}$, к которой осуществляется переход при возврате в основную микропрограмму, связан с адресом A_m микрокоманды $МК_m$, от которой осуществляется переход к подпрограмме, соотношением $A_{m+1} = A_m + 1$ (аналогично для другой точки: $A_{n+1} = A_n + 1$).

Предусмотрим возможность приема в ОУ (например, через входы М ЦПЭ) выдаваемого из БМУ адреса A_m (A_n), как показано на рис. 5.35. Микрокоманда обращения к подпрограмме должна обеспечить хранение в ОУ адреса очередной микрокоманды (для определения точки возврата в основную микропрограмму) и переход к первой микрокоманде под-

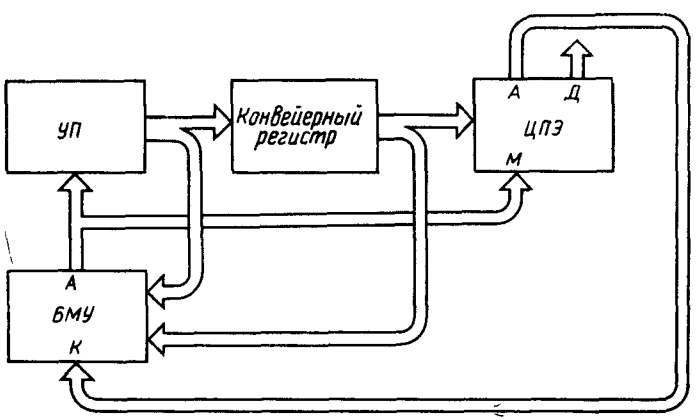


Рис. 5.35. Схема микропроцессора с передачей в ОУ адреса точки выхода из основной микропрограммы

программы. Таким образом, для показанного на рис. 5.35 случая микрокоманда обращения к подпрограмме может иметь следующее содержание:

МК	ОУ: М C_1→АС; БМУ: 1→$Ф_в$; переход к адресу B_1

В подпрограмме принятый в АС адрес возврата в основную микропрограмму уже первой микрокомандой может быть передан в регистр адреса РА:

МК	ОУ: АС→РА; БМУ:...

Адрес возврата с выхода регистра РА поступает через адресную шину на входы $K_7...K_0$ БМУ.

Последняя микрокоманда подпрограммы в поле ЗМ должна содержать 1:

$МК_k^п$	ОУ:...; БМУ: ЗМ=1;...

Это обеспечивает переход по адресу, поступающему на входы K_7... ...K_0, т. е. к очередной микрокоманде основной микропрограммы.

Подобным образом можно обеспечить обращение к одной и той же подпрограмме из нескольких точек основной микропрограммы.

Если в процессе выполнения одной подпрограммы необходимо обеспечить возможность обращения к другой подпрограмме, то описанное построение подпрограммы следует использовать для 2-й подпрограммы. В 1-й же подпрограмме принятый в АС адрес возврата в основную микропрограмму необходимо передавать не в регистр РА, а в один из регистров блока РОН. На рис. 5.36 в качестве регистра для хранения адреса возврата использован регистр R_9. Предпоследней микрокомандой подпрограммы $МК_{k-1}^п$ содержимое этого регистра передается в регистр адреса РА, откуда он затем поступает на входы K_7... K_0 БМУ. Последней микрокомандой $МК_k^п$ осуществляется переход к микрокоманде по этому адресу.

В качестве примера построения микропрограммы с обращением к подпрограмме рассмотрим реализацию цифрового фильтра.

Пример 5.7. Требуется построить микропрограмму, реализующую цифровой фильтр 2-го порядка, описываемый разностным уравнением следующего вида:

$$y(nT) = k_1 \cdot y(nT - T) + k_2 \cdot y(nT - 2T) + k_3 \cdot x(nT - T) + x(nT).$$

Для хранения входящих в правую часть выражения величин выделим следующие регистры:

$$y(nT) \rightarrow R_4;\ y(nT - T) \rightarrow R_5;$$
$$y(nT - 2T) \rightarrow R_6;\ x(nT - T) \rightarrow R_7.$$

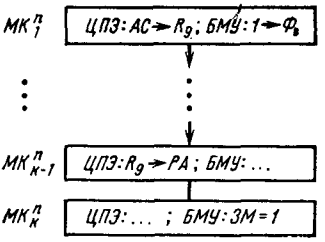


Рис. 5.36. Использование регистра R_9 для хранения адреса возврата

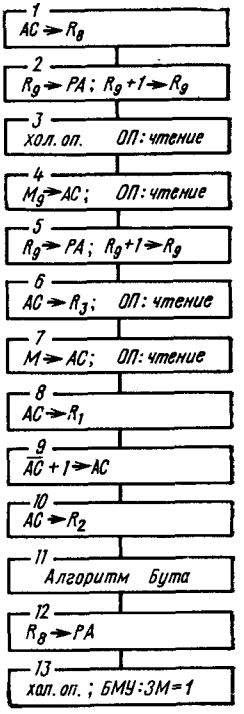


Рис. 5.37. Схема подпрограммы умножения по алгоритму Бута

Хранение коэффициентов k_1, k_2, k_3 будем производить в оперативной памяти (ОЗУ).

Выражение разностного уравнения используем как рекуррентное, строя на нем циклически повторяющийся вычислительный процесс. При каждом повторении цикла будут производиться прием через вход В ОУ очередного значения входного сигнала $x(nT)$ и вычисление очередного значения выходного сигнала $y(nT)$. До начала следующего повторения цикла должны быть выполнены пересылки:

$$R_5 \rightarrow R_6;\ R_4 \rightarrow R_5;\ x(nT) \rightarrow R_7,$$

после чего принимается $x(nT + T)$ и в R_4 формируется $y(nT + T)$.

Как видно из приведенного выше разностного уравнения цифрового фильтра, осуществляемые в цикле вычисления требуют трехкратного выполнения операции умножения. Следовательно, если не предусматривать подпрограммы, то в микропрограмме будет трижды повторяться участок, реализующий операцию умножения. Существенного сокращения длины микропрограммы (а значит, и требуемой емкости управляющей памяти) можно достигнуть, оформив выполнение операции умножения в виде подпрограммы, к которой в каждом повторении цикла трижды производится обращение из основной микропрограммы в точках, где возникает необходимость в выполнении операции умножения.

В качестве подпрограммы может быть использована рассмотренная выше микропрограмма алгоритма Бута, расширенная микрокомандами подготовки операции (с засылкой участвующих в операции чисел в соответствующие регистры) и микрокомандами, необходимыми для осуществления возврата в основную микропрограмму. Схема подпрограммы показана на рис. 5.37.

Здесь предусматривается следующее использование регистров ОУ: R_0 — формирование старших разрядов произведения, R_1 — множимое,

R_2 — множимое, взятое с обратным знаком, R_3 — счетчик в микропрограмме умножения, T — множитель, R_4 — формирование $y\ (nT)$, R_5 — хранение $y\ (nT - T)$, R_6 — хранение $y\ (nT - 2T)$, R_7 — хранение $x\ (nT - T)$, R_8 — адрес возврата в основную микропрограмму, R_9 — адрес данных в оперативной памяти.

Подготовка к операции умножения должна предусматривать передачу в регистры T, R_1, R_2, R_3 соответственно множителя, множимого (коэффициентов k_1, k_2, k_3), множимого, взятого с обратным знаком, и числа разрядов сомножителей n. Предусмотрим пересылку множителя в регистр T в осиовной микропрограмме. В связи с отсутствием в ЦПЭ свободных регистров, хранение значений k_1, k_2, k_3 и n предусматриваем в оперативной памяти. Кроме того, следует предусмотреть хранение в оперативной памяти числа — 6, необходимого для восстановления в регистре R_9 начального адреса массива данных в оперативной памяти.

В табл. 5.34 показано размещение указанных данных в памяти. Будем считать, что в начале каждого повторения цикла в регистре R_9 восстанавливается начальный адрес A.

В процессе каждого выполнения операции умножения в подпрограмме необходимо предусмотреть следующую подготовку к операции: при выполнении умножения $k_3 \cdot x\ (nT - T)$ из оперативной памяти считываются и помещаются в соответствующие регистры числа n и k_3, затем при выполнении операции $k_2 \cdot y\ (nT - 2T)$ производится считывание из оперативной памяти чисел n и k_2 и, наконец, при выполнении операции $k_1 \cdot y\ (nT - T)$ — чисел n и k_1. Принятые в ОУ коэффициенты уравнения после изменения их знака помещаются в регистр R_2. Таким образом, при каждой передаче адреса оперативной памяти из регистра R_9 в регистр РА подготовка в R_9 адреса очередной ячейки памяти, к которой далее производится обращение, сведется к увеличению на единицу содержимого этого регистра.

Рассмотрим порядок действий в приведенной на рис. 5.37 схеме алгоритма подпрограммы умножения.

В основной микропрограмме микрокоманда обращения к подпрограмме осуществляет прием в АС увеличенного на единицу адреса этой микрокоманды в управляющей памяти и переход к микрокоманде $МК_1^п$ подпрограммы. Микрокоманда $МК_1^п$ передает содержимое АС в регистр R_8, где этот адрес возврата в основную микропрограмму хранится до окончания действий в подпрограмме. Мнкрокоманды $МК_2^п$, $МК_3^п$, $МК_4^п$ осуществляют выдачу адреса, чтение из оперативной памяти и прием n в АС; микрокоманды $МК_5^п$, ..., $МК_8^п$ — выдачу следующего адреса, чтение из оперативной памятн, передачу содержащегося в АС числа n

Таблица 5.34

Адрес ячейки оперативной памяти	Содержимое ячейки памяти	Адрес ячейки оперативной памяти	Содержимое ячейки памяти
A	n	A+4	n
A+1	k_3	A+5	k_1
A+2	n	A+6	—6
A+3	k_2		

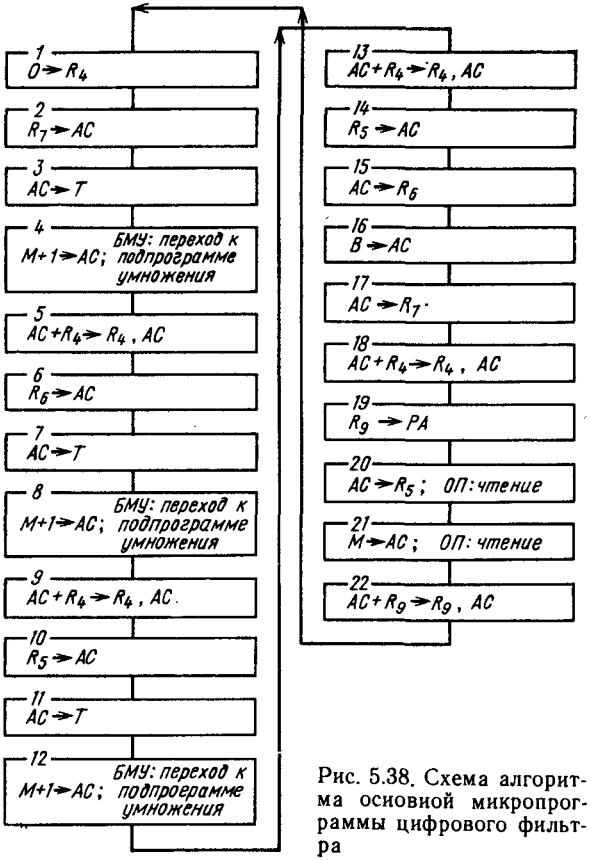


Рис. 5.38. Схема алгоритма осиовиой микропрограммы цифрового фильтра

в регистр R_3, прием в АС участвующего в операции умножения коэффициента (множимого) и передачу его в регистр R_1; микрокоманды $МК^{п}_{9}$ и $МК^{п}_{10}$ формируют в регистре R_2 значение коэффициента с обратным знаком; блок *11* реализует алгоритм Бута; микрокоманды $МК^{п}_{12}$, $МК^{п}_{13}$ осуществляют возврат в основную микропрограмму.

На рис. 5.38 приведена схема алгоритма основной микропрограммы. Здесь пары МК2 и МК3, МК6 и МК7, МК10 и МК11 осуществляют перед обращением к подпрограмме умножения передачу множителя в регистр Т; МК16 выполняет прием через вход В от внешнего устройства ввода очередного значения x; МК14 и МК15, МК17, МК20 осуществляют сдвиг информации, необходимый для подготовки к действиям в следующем повторении цикла; микрокомандами МК19 и МК21 производится чтение из оперативной памяти числа — 6 и прием его в АС. Прибавлением этого числа к содержимому регистра R_9 микрокоманда МК22 восстанавливает в этом регистре начальный адрес массива данных, считываемых из оперативной памяти.

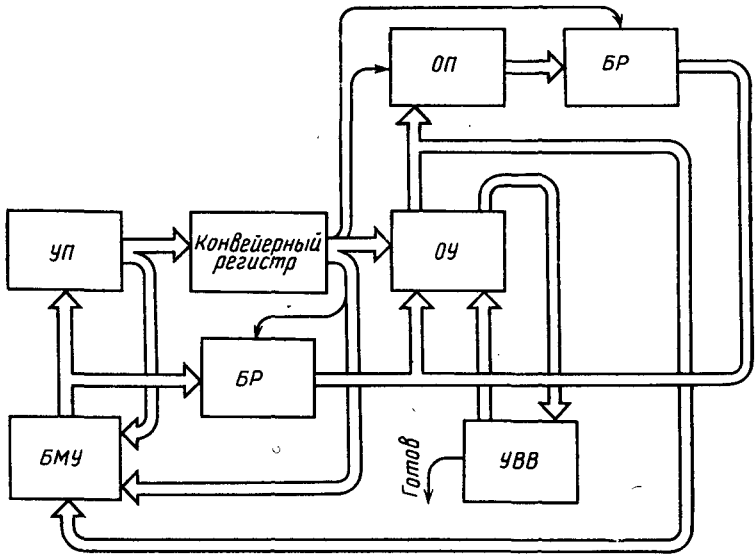


Рис. 5.39. Структурная схема микропроцессорного устройства

На рис. 5.39 показана структурная схема микропроцессорного устройства. Так как на один и тот же вход М ОУ поступает информация с адресного выхода БМУ и выхода данных оперативной памяти, возникла необходимость включения в эти цепи управляемых буферных регистров (БР) с тремя состояниями. Из двух этих буферных регистров в выведенном из 3-го состояния (выключенного) может быть не более чем один регистр. Для управления буферными регистрами необходимо предусмотреть соответствующее поле в микрокомандах.

Учтем следующую особенность работы устройства. В разрядах, участвующих в операции умножения, множитель может содержать различное число единиц и нулей. Таким образом, операция умножения

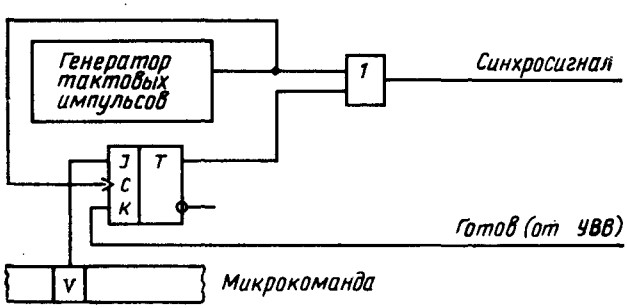


Рис. 5.40. Схема управления синхросигналом

выполняется в различное число тактовых периодов и различным оказывается общее число тактовых периодов, в которое происходит выполнение действий, связанных с однократным повторением цикла. Между тем прием очередного значения x должен происходить в определенные моменты времени, следующие с периодом выдачи их из устройства ввода.

Необходимое согласование в работе ОУ и устройства ввода можно выполнить следующим образом. Предусмотрим в формате микрокоманд одноразрядное поле управления цепью синхронизации. Микрокоманда МК15, предшествующая МК16, осуществляющей прием x, в поле управления цепью синхронизации содержит лог. 1. После завершения выполнения МК15 эта информация, действуя на входе J триггера (рис. 5.40), в момент положительного фронта сигнала генератора тактовых импульсов устанавливает триггер в состояние лог 1. С этого момента триггер подает лог. 1 на вход логического элемента ИЛИ и на выходе этого элемента устанавливается постоянный уровень лог. 1. Постоянный уровень лог. 1 действует в цепи синхронизации устройства. Отсутствие спадов уровня в синхросигнале приведет к тому, что будет оставаться неизменным содержимое регистров в БМУ и ОУ. Следовательно, на адресном выходе БМУ будет действовать один и тот же адрес, одна и та же микромкоманда МК16 будет считываться из управляющей памяти, но действующая на входе В ОУ информация не будет приниматься в АС. В микропроцессорном устройстве устанавливается режим *ожидания*. Такое состояние продолжается до тех пор, пока устройство ввода не подаст сигнал *Готов*. Этот сигнал поступает на вход К триггера и в момент очередного положительного фронта сигнала генератора импульсов переводит триггер в состояние лог. 0. С этого момента устройство выходит из режима ожидания, осуществляется прием числа со входа В в ОУ и продолжается выполнение действий по алгоритму, приведенному на рис. 5.38.

Итак, в рассматриваемом микропроцессорном устройстве микрокоманды должны иметь формат, представленный на рис. 5.41.

В заключение отметим, что использование подпрограмм приводит к снижению быстродействия микропроцессорного устройства. Это свя-

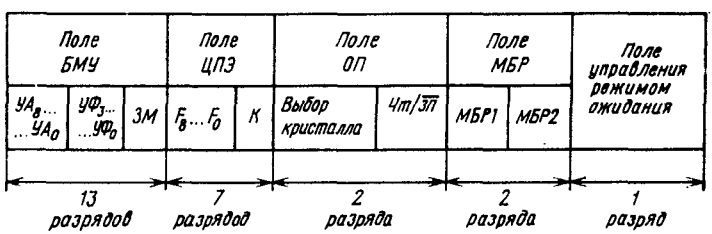


Рис. 5.41. Формат микрокоманды (поле БМУ — 14 разрядов; в поле ЦПЭ — $F_6 \ldots F_0$); в поле БМУ — $УА_6 \ldots УА_0$

зано с тем, что при использовании подпрограмм затрачивается дополнительное время на исполнение микрокоманд, связанных с обращением к подпрограммам и возвратом из подпрограмм в основную микропрограмму.

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ КОМАНД

В случае сложных алгоритмов описание их на языке микрокоманд приводит к тому, что образуется весьма громоздкая микропрограмма (содержащая, например, многие сотни и тысячи микрокоманд). Это вызывает трудности как в составлении микропрограммы, так и в проверке ее правильности. Использование подпрограмм может не разрешить возникающих трудностей. Они преодолеваются при переходе к программированию на *языке команд*.

Рассмотрим этот способ программирования. Изучив алгоритм задачи, определяют минимальный перечень операций, используя который можно описать данный алгоритм. На каждую включенную в список операцию строится микропрограмма операции и команда, которой эта микропрограмма может быть вызвана. Команды хранятся в оперативной памяти, микропрограммы — в управляющей памяти. Для того чтобы по вызванной из оперативной памяти команде можно было бы в управляющей памяти отыскать соответствующую команде микропрограмму, достаточно в команде указать адрес ячейки управляющей памяти, в которой хранится первая микрокоманда этой микропрограммы.

Итак, образуется список команд. Пользуясь таким перечнем команд, с гораздо меньшими трудностями можно описать сложный алгоритм.

Назовем описание алгоритма на языке команд *программой* (в отличие от микропрограммы, дающей это описание на языке микрокоманд). Хранение программы производится в оперативной памяти.

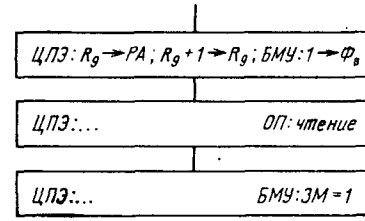
Работа микропроцессорного устройства в этом случае происходит в следующем порядке. Пусть программа не требует привлечения безусловных и условных переходов. Работа с такой программой сводится к последовательной выборке команд из оперативной памяти и их исполнению путем обращения к соответствующим этим командам микропрограммам, хранящимся в управляющей памяти.

Для хранения адреса очередной команды необходимо в ОУ выделить регистр. Пусть таким регистром будет регистр R_9.

Микропрограмма текущей команды завершается микрокомандами чтения из оперативной памяти очередной команды и выдачи из БМУ адреса управляющей памяти, взятого из принятой на входы $K_7 \ldots K_0$ БМУ очередной команды (рис. 5.42). Таким образом осуществляется переход к микропрограмме, реализующей очередную команду.

Программирование на языке команд не только упрощает процесс программирования, но может привести к существенному сокращению

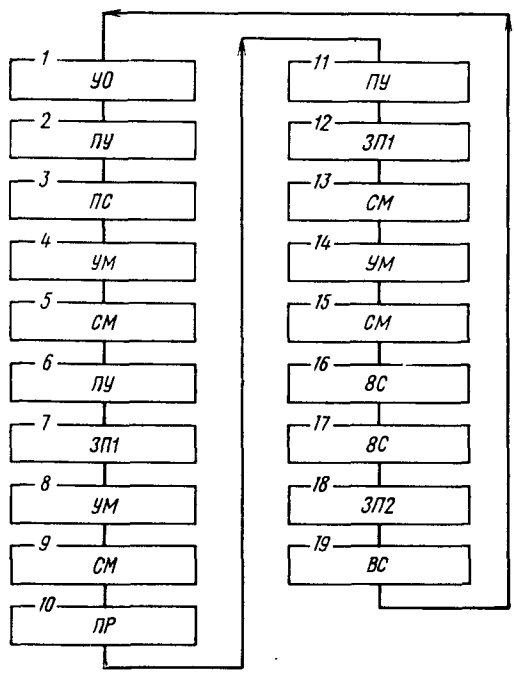
емкости управляющей и оперативной памяти, требуемой для хранения программы и микропрограмм операций. Однако следует иметь в виду, что быстродействие микропроцессорного устройства при этом существенно снижается. Последнее нетрудно понять хотя бы потому, что в каждой мнкропрограмме выполняются микрокоманды, связанные с выборкой команды из оперативной памяти. При описании алгоритма на языке микрокоманд необходимость в таких микрокомандах обращения к оперативной памяти отпадает. Очевидно, чем меньше средний размер микропрограмм, соответствующих отдельным операциям, тем выше в этих микропрограммах удельный вес микрокоманд обращения к оперативной памяти и тем ниже быстродействие микропроцессорного устройства.



Рнс. 5.42. Переход к мнкрокоманде очередиой команды

Покажем программирование на языке команд в рассмотренном выше примере построения цифрового фильтра 2-го порядка.

Пусть значения входящих в разностное уравнение цифрового фильтра величин $y(nT - T)$, $y(nT - 2T)$, $x(nT - T)$ хранятся в оператив-



Рнс. 5.43. Программа цифрового фильтра

ной памяти в ячейках с адресами, соответственно равными $R_8 + 1$, $R_8 + 2$, $R_8 + 3$; коэффициенты k_1, k_2, k_3 имеют в оперативной памяти адреса, соответственно равные R_7, $R_7 + 1$, $R_7 + 2$. Для хранения числа разрядов сомножителей используем один из регистров ОУ, например R_6.

Из анализа алгоритма функционирования фильтра следует, что входящие в алгоритм действия могут быть представлены системой операций, представленной в табл. 5.35.

В табл. 5.36 приведены микропрограммы этих операций.

Пользуясь сформулированной системой команд, нетрудно описать алгоритм цифрового фильтра. Программа, реализующая данный алгоритм, приведена на рис. 5.43. Она реализует следующую последовательность действий.

Команда К1 очищает регистр R_4, подготавливая его к накоплению суммы членов разностного уравнения. Команда К2 производит подготовку к первому умножению $k_1 \cdot y(nT - T)$. Команда К3 осуществляет пересылку $y(nT - T)$ из регистра R_1 в регистр R_5. Здесь это значение хранится до момента, когда из ячейки ОП с адресом $R_8 + 2$ командой К6 будет осуществлено считывание $y(nT - 2T)$ и окажется возможным командой К7 записать $y(nT - T)$ из регистра R_5 в ячейку ОП, где

Таблица 5.35

Условное обозначение команды	Операции, выполняемые командой
ПУ	Подготовка операции умножения: а) засылка в R_3 числа n из регистра R_6, б) засылка миожимого из ОП в регистр R_1, в) формирование в R_2 отрицательиого множимого, г) засылка миожителя из ОП в регистр Т
ПС	Пересылка содержимого R_1 в регистр R_5
УМ	Умиожение с получением старших разрядов произведения в АС
СМ	Суммирование содержимого регистров R_4 и АС
ЗП1	Запись содержимого R_5 в ОП по адресу R_8
ЗП2	Запись содержимого АС в ОП по адресу R_8
ПР	Прием данных от виешиего устройства в R_5
ВС	Умеиьшеиие иа единицу содержимого регистров R_7 и R_8 для восстановления в иих исходиого значеиия
У0	Установка иуля в регистре R_4

Таблица 5.36

Команда	Микропрограмма	Команда	Микропрограмма
ПУ	ЦПЭ: $R_8+1 \to R_8$; БМУ: $1 \to Ф_в$	СМ	ЦПЭ: $AC+R_4 \to R_4$, AC; БМУ: $0 \to Ф_в$
	ЦПЭ: $R_8 \to PA$; БМУ: $0 \to Ф_в$	ЗП1	ЦПЭ: $R_8 \to PA$; БМУ: $0 \to Ф_в$
	ЦПЭ: М→AC; ОП: чт		ЦПЭ: $R_5 \to AC$; БМУ: $0 \to Ф_в$
	ЦПЭ: $AC \to R_1$; БМУ: $1 \to Ф_в$		ЦПЭ: хол. оп.; ОП: зп
	ЦПЭ: $\overline{AC}+1 \to AC$; БМУ: $1 \to Ф_в$	ЗП2	ЦПЭ: $R_8 \to RA$; БМУ: $0 \to Ф_в$
	ЦПЭ: $AC \to R_2$; БМУ: $1 \to Ф_в$		ЦПЭ; хол. оп.; ОП: зп
	ЦПЭ: $R_7 \to PA$; БМУ: $1 \to Ф_в$	ПР	ЦПЭ: хол. оп. Режим ожидания
	ЦПЭ: М→T; ОП: чт		ЦПЭ: В→AC; БМУ: $1 \to Ф_в$
	ЦПЭ: $R_6 \to AC$; БМУ: $0 \to Ф_в$		ЦПЭ: $AC \to R_5$; БМУ: $1 \to Ф_в$
	ЦПЭ: $AC \to R_3$; БМУ: $1 \to Ф_в$	ВС	ЦПЭ: $R_7-1 \to R_7$; БМУ: $0 \to Ф_в$
ПС	ЦПЭ: $R_1 \to AC$; БМУ: $0 \to Ф_в$		ЦПЭ: $R_8-1 \to R_8$; БМУ: $0 \to Ф_в$
	ЦПЭ: $AC \to R_5$; БМУ: $1 \to Ф_в$	У0	ЦПЭ: $0 \to R_4$
УМ	Микропрограмма алгоритма Бута		

ранее хранилось $y\,(nT-2T)$. После выполнения командой К4 операции умножения $k_1 \cdot y\,(nT-T)$ команда К5 накапливает этот результат в регистре R_4. Затем команда К8 производит умножение $k_2 \times y(nT-2T)$ и команда К9 суммирует его с содержащимся в регистре R_4 предыдущим произведением. Команда К10 производит прием $x\,(nT)$ с передачей его для хранения в регистр R_5. После выполнения команды К11 обеспечивается командой К12 возможность передачи $x\,(nT)$ из регистра R_5 в ячейку ОП, в которой ранее хранилось значение $x\,(nT-T)$. Команда К13 прибавляет к хранящейся в регистре R_4 сумме двух произведений значение $x\,(nT)$. Команда К14 формирует $k_3 \cdot x\,(nT-T)$, полученное произведение командой К15 прибавляется к содержимому регистра R_4 и в этом регистре (а также в АС) образуется значение $y\,(nT)$. После выполнения команд К16 и К17 в регистре R_8 образуется адрес ячейки ОП, в которой хранилось значение $y\,(nT-T)$. Командой К18 в эту ячейку записывается сформированное значение $y\,(nT)$ и, таким образом, после выполнения еще одной команды К19 содержимое ОП оказывается подготовленным к повторению цикла.

5.5. УЗЛЫ МИКРОПРОЦЕССОРНОГО УСТРОЙСТВА

МНОГОРЕЖИМНЫЙ БУФЕРНЫЙ РЕГИСТР К589ИР12

Многорежимный буферный регистр (МБР) (рис. 5.44) содержит 8-разрядный информационный регистр, снабженный схемой управления и выходным буфером (ВБ) с тремя состояниями. Кроме того, в блоке имеется устройство формирования сигнала запроса прерывания текущей программы (выделено штриховой линией).

Работа информационного регистра. Информационный регистр построен на D-триггерах, которые при уровне лог. 1 на внутренней цепи разрешения приема данных (РПД) повторяют на своих выходах информацию, поступающую по входной шине данных. При уровне лог. 0 в цепи РПД входы триггеров логически отключаются от входной шины данных, и регистр переходит в режим хранения, сохраняя на своих выходах ранее поступавшую по входной шине информацию. Выходы регистра подключены к выходному буферу ВБ, имеющему три состояния. Выходной буфер управляется сигналом, поступающим по внутренней цепи разрешения выдачи данных (РВД). При поступлении по цепи РВД уровня лог. 1 содержимое информационного регистра передается на выходную шину данных; при поступлении уровня лог. 0 ВБ переходит в третье состояние с высоким выходным сопротивлением, отключая регистр от выходной шины.

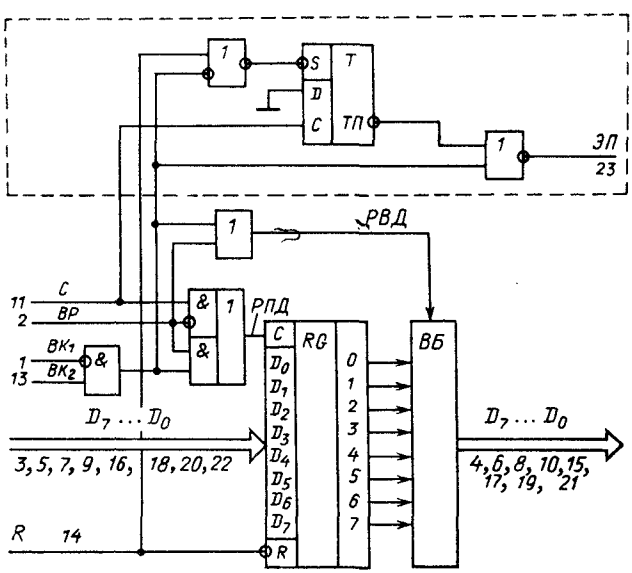


Рис. 5.44. Структуриая схема МБР К589ИР12

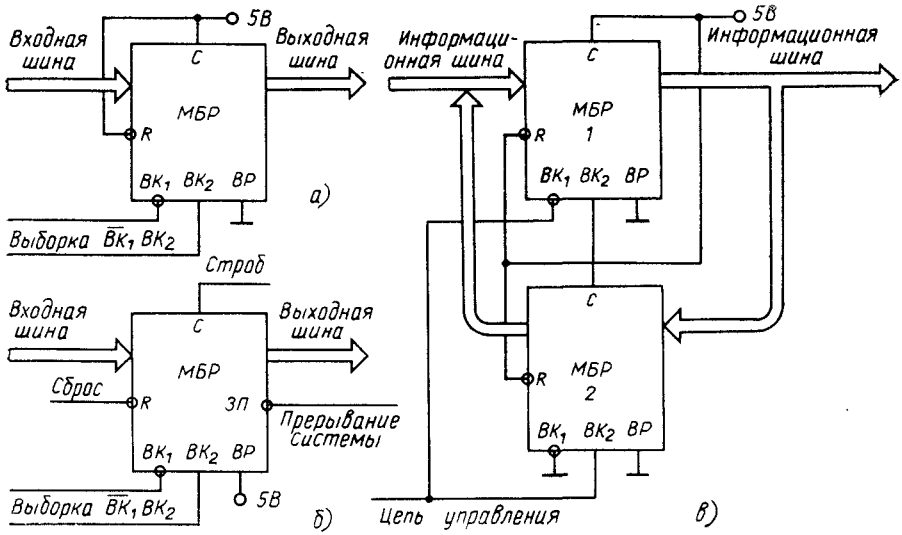


Рис. 5.45. Применение МБР:

а) буферное устройство с отключаемым выходом; *б)* буферное устройство с отключаемым входом; *в)* устройство двунаправленной передачи данных

Уровни в цепях РПД и РВД определяются следующими логическими выражениями:

$$РПД = С \cdot \overline{ВР} \vee \overline{ВК1} \cdot ВК2 \cdot ВР;$$

$$РВД = ВР \vee \overline{ВК1} \cdot ВК2.$$

Формирование сигнала запроса прерывания (ЗП). Сигналу ЗП соответствует уровень лог. 0. Такое представление ЗП удобно для непосредственной подачи на вход блока приоритетного прерывания БПП. Как видно из схемы на рис. 5.44, ЗП формируется при сбросе триггера прерывания ТП в состояние лог. 0 путем подачи строба на вход С либо при выборе устройства, т. е. $\overline{ВК1} \cdot ВК2 = 1$,

$$\overline{ЗП} = С \vee \overline{ВК1} \cdot ВК2.$$

Установка ТП в состояние лог. 1 происходит в соответствии с логическим выражением

$$ТП = \overline{R} \vee \overline{\overline{ВК1} \cdot ВК2}$$

Применения МБР. На рис. 5.45, *а* представлен МБР в режиме, в котором $С \cdot \overline{ВР} = 1$. Таким образом, вход информационного регистра постоянно открыт и происходит непрерывно прием информации с входной шины данных. Выходные буферы в этом режиме открываются лишь

при комбинации сигналов разрешения выборки устройства, удовлетворяющей логическому выражению $\overline{ВК1} \cdot ВК2 = 1$. Образуется буферное устройство с отключаемым от нагрузки выходом.

На рис. 5.45, *б* представлена схема МБР в режиме, в котором за счет установки уровня лог. 1 на входе ВР выходной буфер постоянно открыт. Прием информации с входной шины данных происходит лишь при разрешении выборки устройства, т. е. при выполнении условия $\overline{ВК1} \cdot ВК2 = 1$. Образуется буферное устройство с отключаемым входом.

На рис. 5.45, *в* показано применение МБР для построения устройства двунаправленной передачи данных, в котором сигналом, поданным в управляющую цепь, устанавливается режим передачи в прямом либо обратном направлении. При подаче в управляющую цепь уровня лог. 0 в МБР *1* выходной буфер оказывается включенным, в МБР *2* — отключенным и передача данных по информационной шине происходит в направлении слева направо через МБР *1*. При установке в управляющей цепи уровня лог. 1 МБР *2* устанавливается в режим с включенным выходным буфером, МБР *1* — в режим с отключенным буфером и передача данных осуществляется справа налево через МБР *2*.

БЛОК ПРИОРИТЕТНОГО ПРЕРЫВАНИЯ К589ИК14

Применения блока приоритетного прерывания (БПП). Пусть программирование ведется на уровне команд. Составленная в командах программа хранится в ОП. Каждой предусматриваемой командой операции в ПЗУ микрокоманд соответствует микропрограмма. Считанная из ОП очередная команда поступает на входы $K_7 \ldots K_0$ БМУ. Команда содержит адрес первой МК микропрограммы, которой соответствует операция, предусматриваемая этой командой. Выбранная из ПЗУ

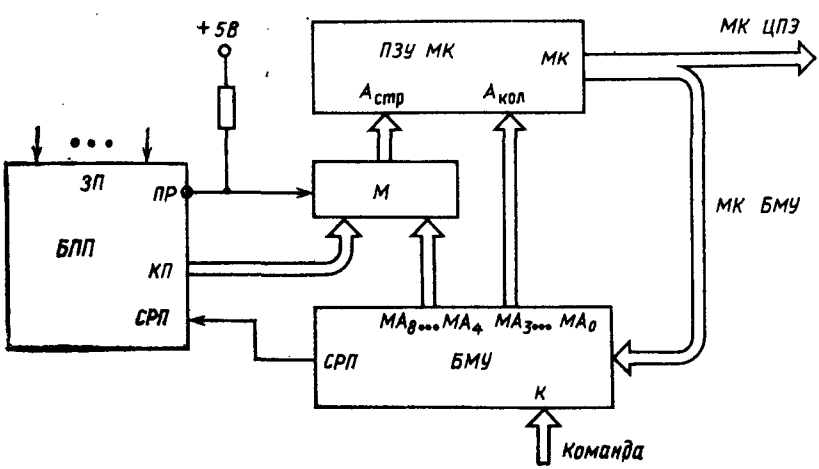


Рис. 5.46. Схема, реализующая прерывания путем переключения адреса строки

первая МК микропрограммы в поле $УА_6...УА_0$ МК БМУ содержит указание о том, как сформировать адрес второй МК, а эта МК — указание о способе формирования адреса третьей МК и т. д., пока не будут выбраны все МК, входящие в микропрограмму данной команды. Последняя МК каждой микропрограммы содержит указание о переходе в нулевую строку (переход вида JZR) и 15-ю колонку. Расположенная по этому адресу (0, 15) ячейка ПЗУ хранит МК, которая в поле ЗМ в МК БМУ содержит 1. К этому моменту на входах $К_7...К_0$ БМУ действует очередная команда и, так как ЗМ = 1, то она загружается в РАМК БМУ. Таким образом, происходит считывание первой МК микропрограммы, соответствующей очередной команде и т. д.

Так происходит выполнение программы в отсутствие прерываний. При этом на выходе прерываний ПР БПП действует сигнал, который в схеме, приведенной на рис 5.46, устанавливает мультиплексор М в состояние передачи на вход ПЗУ адреса строк с выхода $МА_8...МА_4$ БМУ.

Рассмотрим процессы, связанные с прерываниями. При считывании последней МК микропрограммы ЛСх БМУ обнаруживает в разрядах $УА_6...УА_0$ МК БМУ кодовую комбинацию 0111111 перехода вида JZR по адресу (0,15) и выдает сигнал на выход *строба разрешения прерывания* СРП. Этот сигнал поступает в БПП. И если к этому моменту в БПП был принят от какого-либо источника сигнал *запроса прерывания* ЗП, то БПП формирует на выходе ПР сигнал переключения мультиплексора М (наличие цепи R — источник + 5 В связано с тем, что выход ПР построен по типу «открытый коллектор») и на вход $А_{стр}$ ПЗУ адрес строки поступает не из БМУ, а с выхода *кода прерывания* КП БПП. При этом адресом ячейки ПЗУ оказывается не (0, 15), а адрес некоторой другой ячейки в 15-й колонке. Строка ее задается кодовой комбинацией на выходе КП БПП и определяется номером источника, запросившего прерывание текущей программы. Из этой ячейки считывается первая МК прерывающей программы. Действующая на входе БМУ невыполненная команда должна быть помещена в стек. Далее в каждой микропрограмме прерывающей программы обеспечивается считывание из ОП очередной команды, которая к концу выполнения микропрограммы поступает на вход $К_7...К_0$ БМУ.

Так же, как и при исполнении основной программы, последняя МК предусматривает переход к ячейке с адресом (0,15), и оказывается возможным прерывание текущей прерывающей программы с передачей в стек команды, действующей на входе $К_7...К_0$ БМУ.

В процессе выполнения последней команды последней прерывающей программы производится считывание из стека команды и прием ее через входы $К_7...К_0$ в РАМК БМУ. Таким образом осуществляется возврат в последнюю прерванную программу и продолжение ее исполнения.

Если прерывающая программа единственна, то прерывание может быть выполнено по схеме, приведенной на рис. 5.47. Сигнал с выхода ПР БПП подается на вход разрешения выдачи адреса строки РС БМУ.

Таким образом, при появлении на выходе ПР сигнала прерывания (уровня лог. 0) буфер $ВБ_2$ в БМУ закрывается и на входы $А_{стр}$ ПЗУ поступает высокий уровень напряжения (уровня лог. 1) от источника питания + 5 В через резисторы R. В ПЗУ происходит обращение к ячейке с адресом (31, 15), где помещается первая МК прерывающей программы.

На рис. 5.48 показан еще один вариант включения БПП. В общей микрокоманде микропроцессорного устройства предусматривается разряд СРП, содержимое которого подается на соответствующий вход БПП. Этот разряд только в последней МК каждой микропрограммы содержит лог. 1. Кроме того, последняя МК микропрограммы, как и в рассмотренном выше случае, в поле $УА_6 ... УА_0$ МК БМУ предусматривает переход вида JZR к ячейке с адресом (0, 15). Отличие состоит в том, что разряд $УА_0$ поступает в БМУ через инвертор, на выходе которого выполнена операция монтажного И: $УА_0 \cdot \overline{ПР}$. Таким образом, при отсутствии прерывания на входе $УА_0$ БМУ устанавливается уровень лог. 1 и происходит переход к ячейке с адресом (0,15). При прерывании на выходе ПР БПП возникает уровень лог. 0 и происходит переход к ячейке с адресом (0,14); в этой ячейке хранится первая МК прерывающей микропрограммы.

До сих пор рассматривалось программирование на уровне команд. В случае программирования на уровне микрокоманд отсутствуют команды, считываемые из ОП, и вся программа выполняется в виде одной микропрограммы, размещаемой в ПЗУ микрокоманд. В такой микропрограмме можно отметить несколько точек, в которых допускается прерывание. Таким образом, указанные точки разбивают микро-

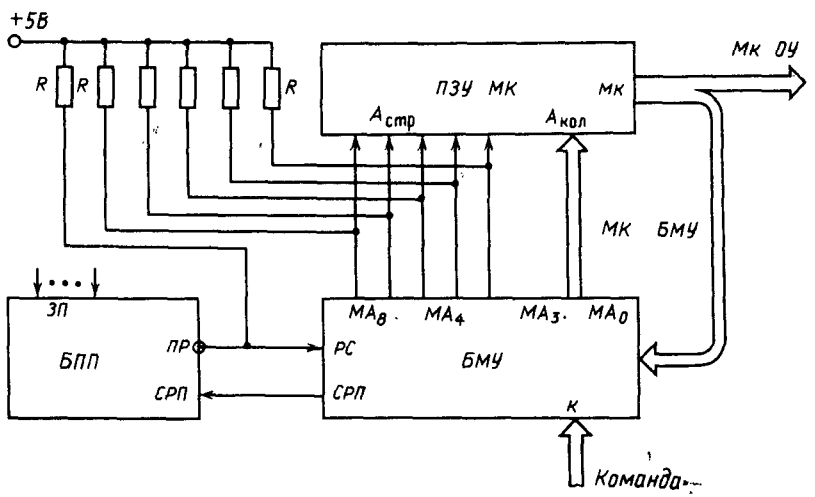


Рис. 5.47. Схема, реализующая прерывания с переходом к единственной прерывающей программе

программу на ряд участков. В каждом из участков последняя МК в поле $УА_6...УА_0$ МК БМУ предусматривает переход вида JZR (в нулевую строку) с номером колонки, равным 15. При этом из БМУ выдается сигнал СРП, принимаемый в БПП. Если в БПП к этому моменту не поступали сигналы запроса прерывания ЗП, то РАМК БМУ загружается со входа $K_7...K_0$ адресом первой микрокоманды следующего участка микропрограммы. Этот адрес подается на входы $K_7...K_0$ БМУ с шины адреса ОУ. Очевидно, к соответствующему моменту времени в РА ОУ должен быть из соответствующего регистра общего назначения передан адрес этой МК, а содержимое регистра общего назначения ОУ увеличивается на единицу и в нем подготавливается адрес первой МК очередного участка микропрограммы.

При наличии запроса прерывания БПП устанавливает на выходе ПР уровень лог. 0; этим сигналом в БМУ закрывается буфер $ВБ_2$, происходит обращение к ячейке ПЗУ с адресом (31, 15), откуда считывается первая МК прерывающей программы. В процессе выполнения прерывающей микропрограммы в регистре ОУ должен быть восстановлен (путем вычитания единицы из содержимого регистра) адрес первой МК участка микропрограммы, переход к которому не произошел из-за выполнения прерывания.

Последняя МК прерывающей микропрограммы должна осуществлять переход к ячейке с адресом (0,15), а к этому моменту на входе $K_7...K_0$ БМУ должен быть выставлен адрес первой МК очередного участка прерванной микропрограммы.

Может быть использован иной способ адресации первой МК участков микропрограммы, не предусматривающий использование регистра ОУ. Адресами этих микрокоманд могут быть выбраны (0,0), (0, 1),..., т. е. адреса ячеек в нулевой строке, исключая адрес (0,15) (указанные ячейки не обязательно должны находиться в нулевой строке, выбор строки может быть произвольным). В процессе исполнения МК каждого участка микропрограммы в РК БМУ запоминается номер колонки ячейки, содержащей первую МК очередного участка микропрограммы. При этом считываемая по адресу (0,15) микрокоманда в поле $УА_6...УА_0$ должна предусматривать переход по содержимому РК (JPR). Прием в РК БМУ соответствующего номера колонки может быть произведен по схеме, приведенной на рис. 5.49. Здесь в БМУ выход адреса колонки $МА_3...МА_0$ заведен ко входам $K_7...K_4$ и ко входам $K_3...K_0$. Если адрес колонки текущей МК совпадает с адресом колонки первой МК очередного участка микропрограммы, то выполняется переход по разрядам команды $K_4...K_7$ (JPX), при котором в РК осуществляется прием информации со входов $K_3...K_0$ и, таким образом, в этом регистре запоминается требуемый адрес колонки.

Структурная схема БПП. Как было показано выше, БПП используется в микропроцессорных устройствах, в которых по сигналам от внешних устройств необходимо прервать выполнение текущей программы и перейти к выполнению специальной, так называемой прерывающей программы этого внешнего устройства. При этом прерывающая

программа может быть в свою очередь прервана другой прерывающей программой по сигналу запроса прерывания ЗП от некоторого другого внешнего устройства.

Программа каждого источника ЗП снабжается уровнем приоритета. При этом прерывание происходит лишь в случае, если уровень приоритета программы, пославшей сигнал ЗП, выше уровня приоритета текущей программы.

Структурная схема БПП приведена на рис. 5.50. Блок обеспечивает прием сигналов ЗП от восьми источников с восемью уровнями приоритета. Запросы прерывания запоминаются в регистре РЗП, причем если одновременно поступает несколько сигналов ЗП, записывается тот из запросов прерывания, уровень приоритета которого выше. Номер запроса на выходе шифратора CD представляется трехразрядным двоичным кодом. В регистре текущего состояния (РТС) хранится в двоичной форме уровень приоритета текущей программы. Компаратор СМР производит сравнение уровней приоритета. Если уровень приоритета запроса выше, то при поступлении из БМУ сигнала *строб разрешения прерывания* СРП триггер прерывания ТП устанавливается в состояние лог. 1 и с инверсного выхода этого триггера на выход ПР подается уровень лог. 0. Этот сигнал с выхода ПР поступает на вход триггера запрета прерывания (ТЗП), устанавливая его в состояние, при котором кратковременно блокируется режим приема запросов в РЗП. Выход ПР можно внешней цепью скоммутировать на вход *разрешения считывания кода прерывания* РСЧ. При этом на выходы *кода прерывания* КП поступит двоичный номер запроса, который может быть использован для определения адреса прерывающей программы.

В микропроцессорном устройстве, построенном с использованием микропроцессора КР580ИК80, эта образующаяся на выходе БПП кодо-

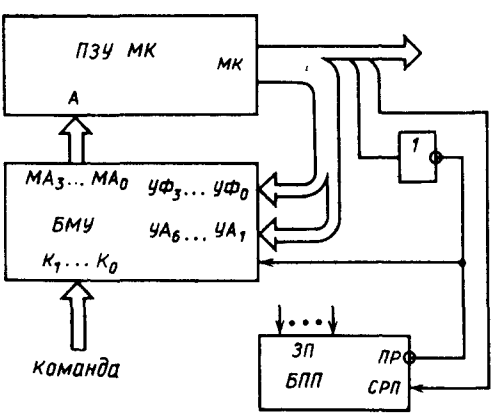


Рис. 5.48. Вариант включення БПП

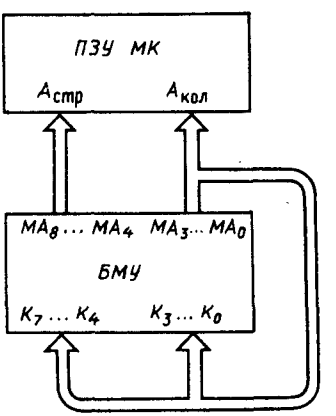


Рис. 5.49. Прием в РК БМУ адреса колонки

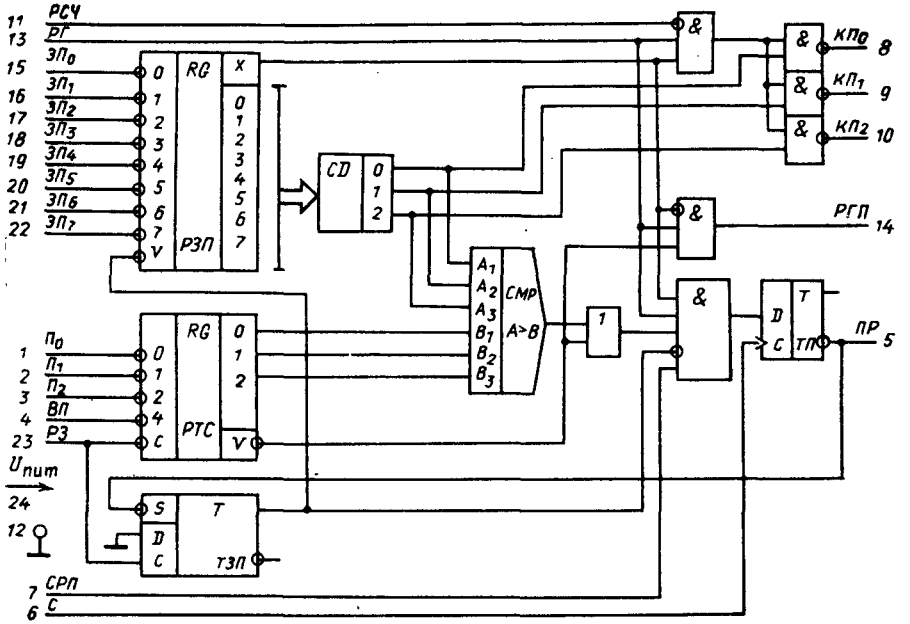


Рис. 5.50. Структурная схема БПП К589ИК14

вая комбинация вписывается в разряды ААА команды прерывания RST (код этой команды 11ААА111) и определяет в оперативной памяти адрес прерывающей программы 0...0ААА000.

Вход *разрешения группы прерываний* РГ и *выход разрешения группы прерываний* РГП позволяют, объединив несколько БПП, построить систему приоритетного прерывания на число входов ЗП, большее восьми.

СОПРЯЖЕНИЕ МИКРОПРОЦЕССОРА С ОПЕРАТИВНОЙ ПАМЯТЬЮ

На рис. 5.51 показаны схема сопряжения микропроцессора с оперативной памятью с помощью регистра, в качестве которого использован многорежимный буферный регистр МБР, и временна́я диаграмма процессов. В момент t_1 завершается выполнение операции в АЛУ ЦПЭ и происходит прием результатов операции в регистры ОУ. С этого же момента могут осуществляться выдача содержимого регистра адреса РА ОУ на шину адреса и прием адреса в ОЗУ. Рассмотрим микропроцессорное устройство с конвейерным регистром.

При чтении на вход $\overline{\text{ВК}}$ ОЗУ подается активный уровень лог. 0, на вход Чт/$\overline{\text{Зп}}$ — высокий уровень лог. 1, соответствующий операции чтения. Выдаваемая из ОЗУ информация в интервале времени $t_2...t_3$

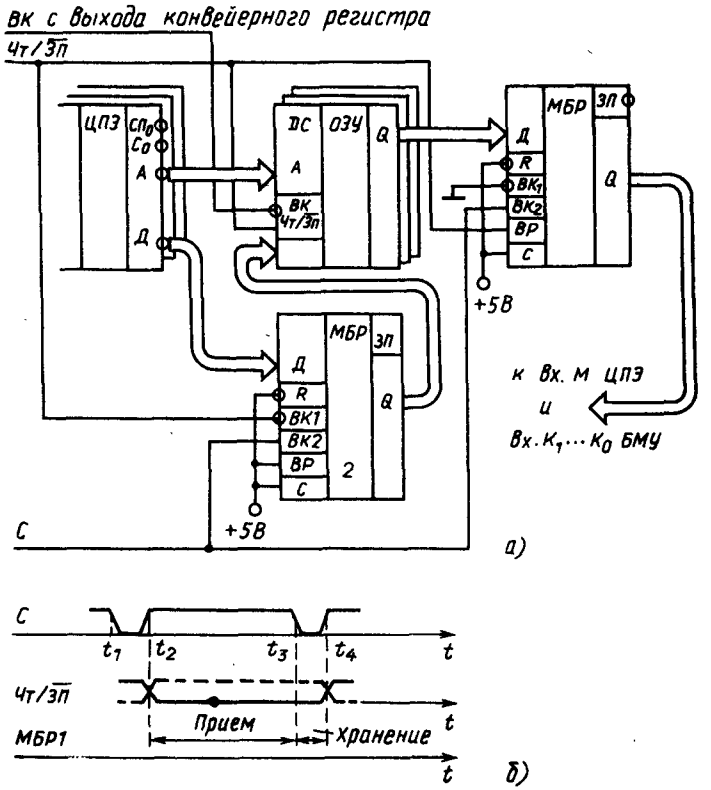


Рис. 5.51. Сопряжение микропроцессора с оперативной памятью:
а) схема; *б)* временные диаграммы процессов

принимается в МБР *1*; далее в интервале $t_3 \ldots t_4$ МБР *1* переводится в режим хранения, продолжая поддерживать на выходе ранее считанную из ОЗУ информацию. Данные с выхода МБР *1* могут поступать на вход М ОУи на входы $К_7 \ldots К_0$ БМУ и принимать участие в операциях, проводимых в этих блоках в следующем тактовом периоде.

Для установки ОЗУ в режим записи на вход Чт/$\overline{\text{Зп}}$ подается уровень лог. 0. Содержимое аккумулятора АС ОУ выдается на шину данных и далее через МБР *2* поступает на вход Д ОЗУ. С момента t_3 МБР *2* переходит в режим хранения, продолжая в интервале времени $t_3 \ldots t_4$ выдавать на выход ранее принятую информацию.

6. МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА НА ОСНОВЕ МПК СЕРИИ КР1804

6.1. СОСТАВ МИКРОПРОЦЕССОРНОГО КОМПЛЕКТА

Входящие в микропроцессорный комплект типы микросхем представлены в табл. 6.1.

Таблица 6.1

Тип микросхемы	Назначение микросхемы	Параметры			
		Разрядность	Быстродействие, нс	Потребляемая мощность, мВт	Количество выводов
КР1804ВС1	Микропроцессорная секция	4	95	1250	40
КР1804ВР1	Схема ускоренного переноса	—	7	545	16
КР1804ИР1	Параллельный регистр	4	100 МГц	650	16
КР1804ВУ1	Схема управления адресом микрокоманды	4	95	650	28
КР1804ВУ2	Схема управления адресом микрокоманды	4	95	650	28
КР1804ВУ3	Схема управления следующим адресом	8	35	575	16

Микропроцессорный комплект (МПК) позволяет строить быстродействующие микропроцессорные устройства с разрядно-модульной организацией, предназначенные для использования в системах обработки сигналов. На МПК серии КР1804 построена выпускаемая промышленностью серийная мини-ЭВМ типа СМ-1420.

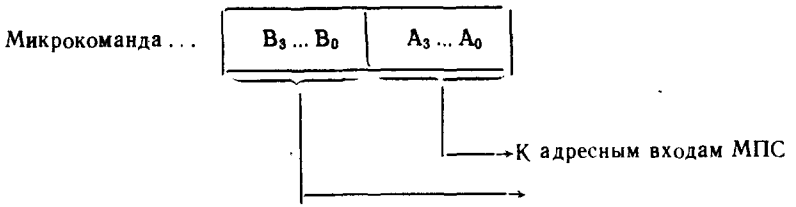
6.2. ПОСТРОЕНИЕ ОПЕРАЦИОННОГО УСТРОЙСТВА

МИКРОПРОЦЕССОРНАЯ СЕКЦИЯ КР1804ВС1

Микропроцессорная секция (МПС) представляет собой 4-разрядную секцию, в которой возможны хранение и обработка 4-разрядных данных. Объединением n МПС можно построить $4n$-разрядное операционное устройство.

На рис. 6.1 показана структурная схема МПС. В ней узлы микросхемы сгруппированы в четыре блока. Рассмотрим их построение и функционирование.

Блок внутренней памяти (БВП). В блоке имеется регистровое запоминающее устройство (РЗУ), содержащее 16 4-разрядных регистров. Адреса регистров представляются 4-разрядными кодовыми комбинациями 0000...1111. РЗУ имеет два адресных входа $A_3...A_0$ и $B_3...B_0$, на которые информация поступает из микрокоманды:



Задавая в полях микрокоманды адреса $A_3...A_0$ и $B_3...B_0$, можно одновременно производить чтение и выдачу на выходы А и В РЗУ содержимого любой пары регистров (при совпадении адресов $A_3...A_0$ и $B_3...B_0$ на оба выхода А и В РЗУ передается содержимое одного и того же регистра). Выданное на выходы А и В содержимое регистров РЗУ принимается соответственно в регистры РгА и РгВ. Далее эти регистры служат источниками операндов, над которыми выполняются операции.

Запись в РЗУ в каждом тактовом периоде может производиться лишь в один из регистров, адрес которого задается шиной $B_3...B_0$. Записываемые в РЗУ данные поступают на вход РЗУ с выхода арифметическо-логического устройства (АЛУ) через узел сдвигателя даиных АЛУ (СДА). Данные через СДА могут передаваться без сдвига либо со сдвигом на один разряд влево или вправо. Таким образом, за один так-

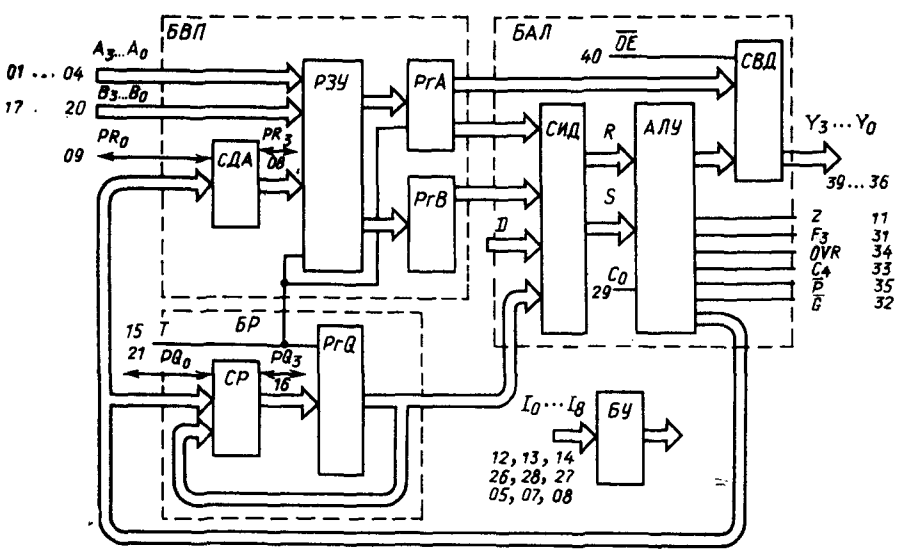


Рис. 6.1. Структурная схема микропроцессорной секции КР1804ВС1

товый период из РЗУ может быть выдано содержимое двух регистров, над ними в АЛУ выполнена некоторая операция и полученный в АЛУ результат операции сдвинут вправо либо влево и записан в регистр РЗУ. Выводы PR_0 и PR_3 в зависимости от направления сдвига служат входом или выходом, через которые производятся запись значения в освобождающийся при сдвиге разряд и выдача содержимого выдвигаемого разряда.

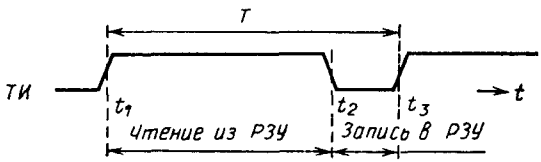


Рис. 6.2. Временная диаграмма чтения и записи в РЗУ

Чтение из регистров РЗУ, адресуемых шинами $A_3...A_0$ и $B_3...B_0$, происходит при высоком уровне тактового сигнала Т (на рис. 6.2 временной интервал $t_1...t_2$). Вход РЗУ при этом логически отключен и РЗУ не реагирует на поступающую на вход информацию. Выдаваемые из РЗУ данные принимаются в регистры операндов РгА и РгВ, построенные на простых триггерах с защелкой. При низком уровне тактового сигнала (временной интервал $t_2...t_3$) входы регистров РгА и РгВ логически отключаются от выходов РЗУ и регистры продолжают хранить принятую информацию. При этом в регистр РЗУ, адресуемый шиной $B_3...B_0$, производится запись передаваемой через СДА информации. Таким образом, чтение и запись в РЗУ разнесены во времени.

Блок рабочего регистра Q (БР). Блок содержит одиночный 4-разрядный регистр Q, построенный на триггерах D-типа. Содержимое регистра постоянно передается в узел АЛУ (в блоке БАЛ). Запись же в регистр может производиться на положительном фронте тактовых импульсов. Данные на вход регистра передаются через узел сдвигателя регистра Q (СР), который работает аналогично узлу СДА блока БВП, передавая записываемые в регистр данные без сдвига либо со сдвигом на один разряд влево или вправо. В отличие от СДА, через который передается лишь полученный на выходе АЛУ результат выполненной операции, через СР на вход регистра Q может передаваться либо результат операции с выхода АЛУ либо содержимое самого регистра Q. Последнее обеспечивает возможность выполнения сдвига содержимого регистра Q, производимого параллельно с операцией в АЛУ.

Блок арифметическо-логический (БАЛ). АЛУ имеет два 4-разрядных входа R и S. Данные на эти входы поступают с выхода селектора источников данных (СИД). Кроме этих входов АЛУ имеет вход для подачи переноса C_0.

На вход R АЛУ СИД коммутирует или выход регистра РгА блока БВП или внешнюю шину данных $D_3...D_0$ либо передает на этот вход нулевое значение. На вход S СИД коммутирует один из трех источников (РгА, РгВ, РгQ) или передает нулевое значение.

Результат операции с выхода АЛУ, как отмечалось выше, подается

на сдвигатели СДА и СР блоков БВП и БР. Кроме того, результат операции подается на селектор выходных данных (СВД), который коммутирует в выходную шину данных $Y_3 \ldots Y_0$ содержимое регистра РгА блока БВП либо выход АЛУ. Селектор выходных данных построен на элементах с тремя состояниями и управляется сигналом $\overline{OE}$. Передача информации на шину $Y_3 \ldots Y_0$ происходит при управляющем сигнале $\overline{OE} = 0$, при сигнале $\overline{OE} = 1$ СВД переводится в третье (выключенное) состояние и микросхема МПС отключается от шины $Y_3 \ldots Y_0$.

АЛУ имеет выходы, на которых формируются следующие признаки результата выполненной операции:

Z — признак нулевого результата (Z = 1, если результат операции F = 0),

F_3 — старший разряд результата, который может рассматриваться как знаковый разряд,

C_4 — признак переноса (при выполнении арифметических операций $C_4 = 1$, если возникает перенос из старшего разряда),

OVR — признак переполнения (при выполнении арифметических операций $OVR = C_3 \oplus C_4$, где C_3 — перенос, передаваемый в старший разряд); если числа представляются со знаковым разрядом, то при OVR = 1 искажается знаковый разряд, т. е. результат оказывается ошибочным.

Выходы АЛУ $\overline{P}$ и $\overline{G}$ предназначены для подключения микросхемы МПС к микросхеме СУП.

Блок управления (БУ). Блок предназначен для преобразования содержимого поля кода операции $I_8 \ldots I_0$ микрокоманды в систему управляющих сигналов, под действием которых в узлах микросхемы МПС выполняются микрооперации. Таким образом, для управления рассмотренными процессами требуется микрокоманда, содержащая следующие поля:

Микрокоманда ...	OE ... C_0	$I_8 \ldots I_0$	$B_3 \ldots B_0$	$A_3 \ldots A_0$

Операции, выполняемые в МПС. Девятиразрядное поле кода операции $I_8 \ldots I_0$, определяющее выполняемую в микросхеме операцию, делится на три поля: поле управления источниками данных I_{210}, поле

Таблица 6.2

Кодовая комбинация				Источник операндов АЛУ		Кодовая комбинация				Источник операндов АЛУ	
I_2	I_1	I_0	8-ричный код	R	S	I_2	I_1	I_0	8-ричный код	R	S
0	0	0	0	A	Q	1	0	0	4	0	A
0	0	1	1	A	B	1	0	1	5	D	A
0	1	0	2	0	Q	1	1	0	6	D	Q
0	1	1	3	0	B	1	1	1	7	D	0

управления операциями АЛУ I_{543} и поле управления приемником I_{876}.

Поле управления источниками данных управляет источниками операндов, подаваемых на входы R и S АЛУ. В табл. 6.2 приведены комбинации значений I_{210} и соответствующие им операнды на входах R и S.

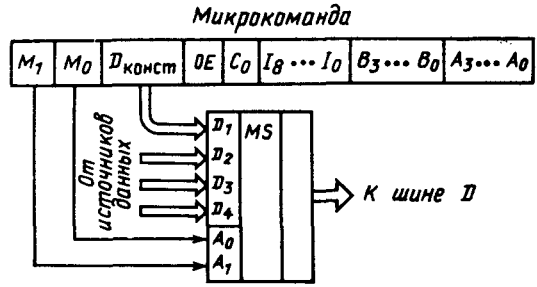


Рис. 6.3. Схема коммутации источников на шину D

Под A и B понимается содержимое соответственно регистров РгА и РгВ, в которые передается содержимое регистров РЗУ, адресуемых полями $A_3 \ldots A_0$ и $B_3 \ldots B_0$ микрокоманды; Q – содержимое регистра Q; D — данные, поступающие из внешней входной цепи данных D.

В тех случаях, когда предусматривается возможность приема с шины данных D информации от нескольких источников (одним из таких источников может быть константа, записанная в соответствующее поле микрокоманды), возникает необходимость коммутации этих источников на шину D. Принцип такой коммутации показан на рис. 6.3.

В поле микрокоманды предусматривается поле $D_{конст}$, в которое при программировании заносится константа, требуемая при выполнении операции. Следует иметь в виду, что это поле в микрокоманде имеет такое же число разрядов, какова разрядность обрабатываемых в операционном устройстве данных. Например, если в операционном устройстве обрабатываются 12-разрядные данные, то для его построения потребуется три микросхемы МПС и из 12-разрядного поля $D_{конст}$ отдельные четверки разрядов будут подаваться в соответствующие МПС.

Кроме $D_{конст}$, которое берется из микрокоманды, может потребоваться коммутирование на шину D МПС данных от других источников. В этом случае в микрокоманде следует предусмотреть поле (на рис. 6.3 таким полем является поле M_1, M_0), содержимое которого подается на

Таблица 6.3

Кодовая комбинация				Операция АЛУ	Кодовая комбинация				Операция АЛУ
I_5	I_4	I_3	8-ричный код		I_5	I_4	I_3	8-ричный код	
0	0	0	0	$R+S+C_0$	1	0	0	4	$R \wedge S$
0	0	1	1	$S-R-1+C_0$	1	0	1	5	$\overline{R} \wedge S$
0	1	0	2	$R-S-1+C_0$	1	1	0	6	$R \oplus S$
0	1	1	3	$R \vee S$	1	1	1	7	$\overline{R \oplus S}$

адресные входы мультиплексора, коммутирующего соответствующие источники на шину D.

Поле кода операций АЛУ (табл. 6.3) определяет вид выполняемой в АЛУ операции. Как видно из таблицы, предусмотрено выполнение трех арифметических и пяти логических операций. При выполнении арифметических операций учитывается перенос C_0, поступающий по входной цепи переноса. Он прибавляется к результату выполненной операции сложения или вычитания.

Рассмотрим особенности операции вычитания. Пусть $I_{543} = 001$. Эту операцию можно было бы представить в виде $S - R - 1 + C_0 = S + \overline{R} + C_0$. Действительно, пусть $S = 6\ (0110_2)$, $R = 5\ (0101_2)$, $C_0 = 0$. Тогда $S + \overline{R} = 0110_2 + \overline{0101_2} =$

$$\begin{array}{rl} & 0110_2 \\ + & \overline{1010_2} \\ 1\leftarrow & \overline{0000} \end{array}$$

Получен нулевой результат, что соответствует операции $S - R - 1$. Таким образом, эта операция при $C_0 = 0$ является операцией суммирования обратных кодов, при $C_0 = 1$ — операцией суммирования дополнительных кодов.

Логические операции выполняются поразрядно, т. е. каждый разряд результата получается путем выполнения логической операции над соответствующими разрядами операндов. Например, пусть $I_{543} = 011$, $I_{210} = 001$. Из табл. 6.2 следует, что на входы R и S АЛУ поступают соответственно операнды A и B. Из табл. 6.3 следует, что над операндами выполняется операция дизъюнкции. Следовательно, будет выполнена операция $A \vee B$. И если $A = 0110_2$ и $B = 0101_2$, то результат операции будет

$$\begin{array}{rl} & A\ 0110 \\ & B \vee 0101 \\ \hline A \vee B & \quad 0111 \end{array}$$

При тех же значениях операндов A, B и коде операции $I_{543} = 100$ и $I_{210} = 001$ выполняется операция конъюнкции:

$$\begin{array}{rl} & A\ 0110 \\ & B \wedge 0101 \\ \hline A \wedge B & \quad 0100 \end{array}$$

При коде операции $I_{543} = 101$ и $I_{210} = 001$ операция над теми же операндами приведет к следующему результату:

$$\begin{array}{rl} & \bar{A}\ 1001 \\ & B \wedge 0101 \\ \hline \bar{A} \wedge B & \quad 0001 \end{array}$$

Обозначенная символом $\oplus$ операция — операция поразрядного суммирования по модуля 2 (эту операцию называют также операцией ИСКЛЮЧАЮЩЕЕ ИЛИ или операцией НЕРАВНОЗНАЧНОСТИ). При $I_{543} = 0110$ и $I_{210} = 001$ над операндами A и B будет выполнена операция:

$$\begin{array}{ll} A & 0110 \\ B & 0101 \\ \hline A \oplus B & 0011 \end{array}$$

Операция $\overline{R \oplus S}$ является операцией логической РАВНОЗНАЧНОСТИ. Например, если $I_{543} = 111$ и $I_{210} = 001$, то результат операции

$$\begin{array}{ll} A & 0110 \\ B & 0101 \\ \hline \overline{A \oplus B} & 1100 \end{array}$$

В табл. 6.4 представлены данные табл. 6.2 и 6.3 в форме, облегчающей поиск кодовых комбинаций I_{543} и I_{210}, соответствующих определенным операциям.

Поле управления приемником операнда I_{876} определяет следующие процессы:

должен ли на выход Y быть выдан результат операции с выхода АЛУ или содержимое регистра РгА;

должен ли результат операции с выхода АЛУ загружаться в регистр Q либо в регистр РЗУ, адресуемый полем $B_3 \ldots B_0$;

должен ли при загрузке регистра РЗУ передаваемый с выхода АЛУ результат операции сдвигаться вправо или влево и должен ли одновременно с этим производиться сдвиг содержимого РгQ. Эти процессы определяются табл. 6.5.

Комбинация $I_{876} = 000$ производит загрузку полученного на выходе АЛУ результата операции F в регистр Q и выдает его на выход Y; при $I_{876} = 001$ результат операции F не фиксируется в регистрах МПС и лишь выдается на выход Y; при $I_{876} = 010$ результат операции F запоминается в регистр РЗУ, а на выход Y выдается содержимое регистра РгА; при $I_{876} = 011$ результат операции F передается в регистр РЗУ и на выход Y; при $I_{876} = 100$ результат операции F выдается на выход Y, кроме того, он сдвигается вправо (сдвигу вправо соответствует уменьшение в два раза значения числа) и заносится в регистр РЗУ, сдвигается вправо содержимое РгQ; при $I_{876} = 101$, в отличие от предыдущей комбинации, содержимое РгQ не сдвигается; при $I_{876} = 110$, в отличие от комбинации $I_{876} = 100$, производится сдвиг не вправо, а влево (сдвигу влево соответствует увеличение значения числа в два раза); при $I_{876} = 111$, в отличие от предыдущей комбинации, содержимое регистра Q не сдвигается.

Таблица 6.4

Арифметические операции АЛУ					Логические операции АЛУ					
I_{543}, I_{210}	Группа	Функция	Группа	Функция	I_{543}, I_{210}	Группа	Функция	I_{543}, I_{210}	Группа	Функция
$C_0=0$			$C_0=1$							
0 0	Сложение	A+Q	Сложение	A+Q+1	4 0	И	$A \wedge Q$	6 2	Пропуск	Q
0 1		A+B		A+B+1	4 1		$A \wedge B$	6 3		B
0 5		D+A		D+A+1	4 5		$D \wedge A$	6 4		A
0 6		D+Q		D+Q+1	4 6		$D \wedge Q$	6 7		D
0 2	Пропуск	Q	Счет	Q+1	3 0	ИЛИ	$A \vee Q$	3 2	Пропуск	Q
0 3		B		B+1	3 1		$A \vee B$	3 3		B
0 4		A		A+1	3 5		$D \vee A$	3 4		A
0 7		D		D+1	3 6		$D \vee Q$	3 7		D
1 2	Обратный счет	Q—1	Пропуск	Q	6 0	ИСКЛЮЧАЮЩЕЕ ИЛИ	$A \oplus Q$	4 2	Нуль	0
1 3		B—1		B	6 1		$A \oplus B$	4 3		0
1 4		A—1		A	6 5		$D \oplus A$	4 4		0
2 7		D—1		D	6 6		$D \oplus Q$	4 7		0
2 2	Обратный код	—Q—1	Дополнительный код	—Q	7 0	Равнозначность	$\overline{A \oplus Q}$	5 0	Маска	$\overline{A} \wedge Q$
2 3		—B—1		—B	7 1		$\overline{A \oplus B}$	5 1		$\overline{A} \wedge B$
2 4		—A—1		—A	7 5		$\overline{D \oplus A}$	5 5		$\overline{D} \wedge A$
1 7		—D—1		—D	7 6		$\overline{D \oplus Q}$	5 6		$\overline{D} \wedge Q$
1 0	Вычитание (обратный код)	Q—A—1	Вычитание (дополнительный код)	Q—A	7 2	Инверсия	$\overline{Q}$			
1 1		B—A—1		B—A	7 3		$\overline{B}$			
1 5		A—D—1		A—D	7 4		$\overline{A}$			
1 6		Q—D—1		Q—D	7 7		$\overline{D}$			
2 0		A—Q—1		A—Q						
2 1		A—B—1		A—B						
2 5		D—A—1		D—A						
2 6		D—Q—1		D—Q						

Таблица 6.5

Кодовая комбинация				РЗУ		PrQ		Вых Y	СДА		СР	
I_8	I_7	I_6	8-ричный код	Сдвиг	Загрузка	Сдвиг	Загрузка		PR_0	PR_3	PQ_0	PQ_3
0	0	0	0	—	—	—	F→Q	F	×	×	×	×
0	0	1	1	—	—	—	—	F	×	×	×	×
0	1	0	2	—	F→B	—	—	A	×	×	×	×
0	1	1	3	—	F→B	—	—	F	×	×	×	×
1	0	0	4	Вправо	F/2→B	Вправо	Q/2→Q	F	F_0	Вход	Q_0	Вход
1	0	1	5	Вправо	F/2→B	—	—	F	F_0	Вход	Q_0	×
1	1	0	6	Влево	2F→B	Влево	2Q→Q	F	Вход	F_3	Вход	Q_3
1	1	1	7	Влево	2F→B	—	—	F	Вход	F_3	×	Q_3

ОБЪЕДИНЕНИЕ МИКРОПРОЦЕССОРНЫХ СЕКЦИЙ В ОПЕРАЦИОННОМ УСТРОЙСТВЕ

Требуемая разрядность операционного устройства обеспечивается объединением некоторого числа МПС. Каждая МПС в операционном устройстве хранит и обрабатывает 4-разрядную группу данных, и если используется n микропроцессорных секций, то разрядность операционного устройства равна $4 \cdot n$. На рис. 6.4 показано объединение четырех секций в 16-разрядном операционном устройстве. Рассмотрим вопросы, связанные с таким объединением.

Одной из задач, которые приходится при этом решать, является обеспечение малого времени задержки переноса, поступающего на

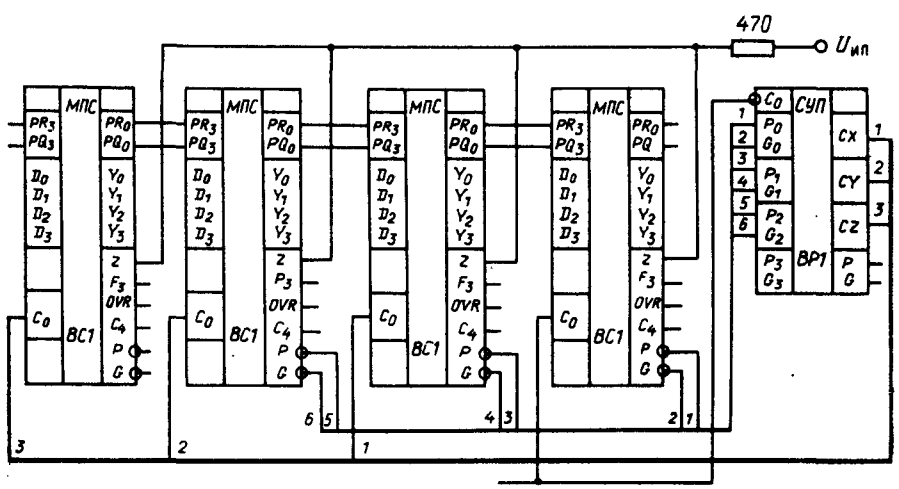


Рис. 6.4. Схема 16-разрядного операционного устройства

вход C_0 секций. Один из возможных способов построения цепи передачи переносов при объединении МПС — последовательный, при котором выход C_4 секции подключается к входу C_0 следующей, более старшей секции. При этом на вход C_0 каждой секции сигнал переноса поступает с задержкой, с которой проходят переносы через все предыдущие секции. Эта задержка в цепи от входа C_0 до выхода C_4 в одной секции составляет 20 нс, и если в операционном устройстве объединены n секций, то задержка в поступлении переноса на вход C_0 старшей секции относительно момента поступления переноса C_0 на вход младшей секции составит $(n - 1)$ 20 нс. При большом числе n объединяемых секций эта задержка существенно отразится на быстродействии операционного устройства. Уменьшение задержки в формировании и подаче переносов на входы C_0 МПС обеспечивает применение схемы ускоренного переноса (СУП) КР1804ВР1. Информация, необходимая для формирования в СУП переносов, выдаваемых на входы C_0 секций, подается в виде сигналов с выходов P и G секций, как показано на рис. 6.4.

Определим, какова задержка между моментом подачи адресов в шины $A_3...A_0$ и $B_3...B_0$ и моментом поступления на вход четвертой секции переноса C_0 в схеме с последовательной передачей переносов и в схеме с использованием СУП.

В схеме операционного устройства с последовательной передачей переносов время распространения сигнала от входов $A_3...A_0$ и $B_3...B_0$ до выхода переноса C_4 в первой секции составляет 70 нс, далее задержка во 2-й и 3-й секциях в цепи от входа C_0 до выхода C_4 составит $20 \cdot 2 = 40$ нс и, таким образом, искомая задержка составит 110 нс.

В схеме с СУП время распространения сигнала от входов $A_3...A_0$ и $B_3...B_0$ до выходов P и G, сигналы с которых поступают в СУП, равно 59 нс, задержка в СУП равна 5 нс и, таким образом, задержка в поступлении сигнала переноса на вход C_0 4-й секции составит $59 + 5 = 64$ нс.

Различие в значении задержки при двух способах построения цепей переносов растет с ростом числа объединяемых МПС. При использовании СУП с ростом числа объединяемых секций быстродействие снижается незначительно.

Другая задача, решаемая при объединении МПС, состоит в построении цепей передачи переносов при выполнении операций сдвигов. Если производится сдвиг вправо в СДА и СР микропроцессорных секций, то выдвигаемое из секций на выходы PR_0 и PQ_0 содержимое младших разрядов должно передаваться на входы PR_3 и PQ_3 следующих младших секций для ввода их в освобождающиеся при сдвиге старшие разряды регистров. При сдвиге влево из секций на выходы PR_3 и PQ_3 выдвигается содержимое старших разрядов, оно должно вдвигаться через входы PR_0 и PQ_0 в освобождающиеся при сдвиге младшие разряды следующих старших МПС. Таким образом, при объединении МПС необходимо обеспечить соединение выводов PR_0 и PQ_0 секции с выводами PR_3 и PQ_3 следующей старшей секции.

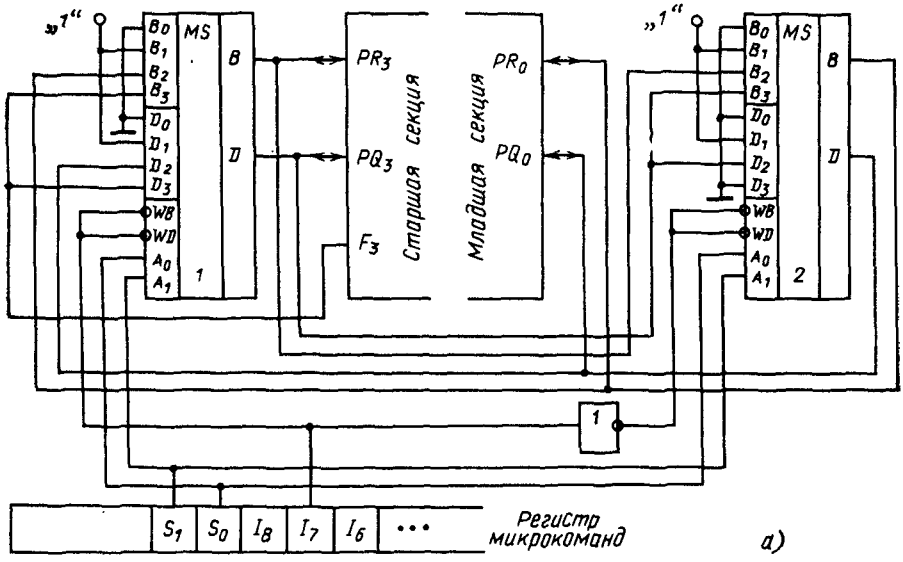
В операционном устройстве оказываются свободными выводы PR_3

и PQ_3 старшей секции и выводы PR_0 и PO_0 младшей секции. Решение задачи может потребовать выполнения различных видов сдвигов. На рис. 6.5, *б* показан некоторый набор сдвигов, который может быть представлен в операционном устройстве. Каждый вид сдвига требует определенного соединения оставшихся свободными выводов сдвига операционного устройства или подачи на эти выводы определенной информации. Это может быть выполнено с помощью мультиплексоров, показанных на рис. 6.5,*а*. Управление мультиплексорами осуществляется разрядом I_7 кода микрокоманды МПС ($I_7 = 1$ соответствует сдвигу влево, $I_7 = 0$ — сдвигу вправо) и сигналами S_1, S_0, снимаемыми из специального поля управления мультиплексорами, предусмотренного в микрокоманде.

Третья задача, решаемая при построении операционного устройства, — формирование слова состояния (признаков, предназначенных для выполнения условных переходов). При объединении МПС необходимо объединить выходы Z секций и подключить через резистор к источнику питания, как показано на рис. 6.4. Такая объединенная цепь служит выходом признака нуля. В качестве остальных выводов признаков используются выходы признаков из старшей МПС. Выходы признаков остальных секций остаются неиспользованными. Из этих признаков и значений, появляющихся на выходах PR_0 и PQ_0 младшей секции при выполнении операции сдвига вправо и на выходе PR_3 старшей секции при выполнении операции сдвига влево, формируется *слово состояния* операционного устройства.

Слово состояния (CC) формируется группой мультиплексоров MS CC, состоящей из четырех мультиплексоров, и оно в начале каждого тактового периода (на положительном фронте тактовых импульсов) принимается в регистр слова состояния R G CC (рис. 6.6). Как видно из схемы, в 4-разрядном слове состояния либо сохраняются значения отдельных разрядов, сформированных в предыдущих тактовых периодах (за счет подачи на входы мультиплексоров MS CC выходов регистра R G CC), либо эти значения обновляются. Три разряда слова состояния хранят признаки Z, F_3, OVR, принимаемые с выходов признаков операционного устройства. В четвертом разряде слова состояния предусматривается хранение одного из следующих признаков: C_4, PR_3, PR_0, PQ_0, либо в этом разряде устанавливается фиксированное значение лог. 0, лог. 1, либо может быть повторено значение этого разряда, сформированное в предыдущем тактовом периоде. Этот разряд используется для формирования в мультиплексоре MS C_0 значения, предназначенного для подачи во входную цепь C_0 младшей МПС. С помощью мультиплексора MS C_0 в цепь C_0 операционного устройства может быть выдано одно из четырех значений: лог. 0, лог. 1, четвертый разряд слова состояния либо его инверсия.

Управление четырьмя мультиплексорами MS CC производится содержимым соответствующего 6-разрядного поля микрокоманды, для управления мультиплексором MS C_0 в микрокоманде следует предусмотреть соответствующее двухразрядное поле.



Код			Схема сдвига при I=334/6
I_7	S_1	S_0	ст РЗУ мл ст Q мл
1	0	0	0 0
1	0	1	1 1
1	1	0	
1	1	1	0
0	0	0	0 0
0	0	1	1 1
0	1	0	
0	1	1	F3

б)

Рис. 6.5. Реализация сдвигов:
а) схема; *б)* таблица сдвигов

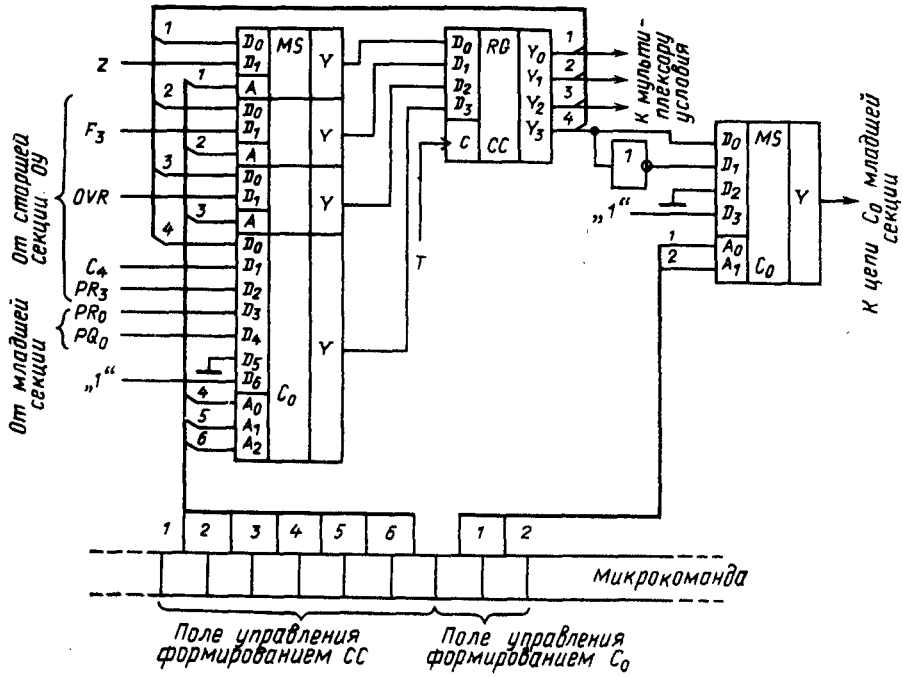
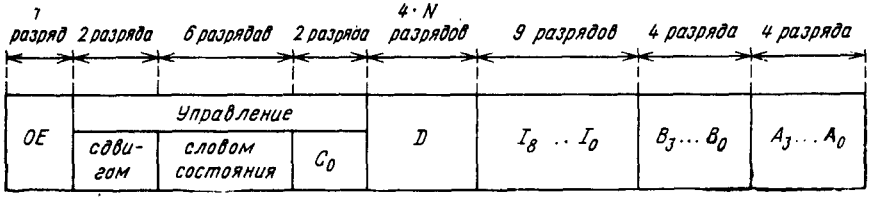


Рис. 6.6. Схема хранения и выдачи признаков

Мы рассмотрели один из возможных вариантов формирования слова состояния. В зависимости от решаемой микропроцессорным устройством задачи разрядность слова состояния и схема его формирования могут быть различными. Схема и разрядность соответствующих полей микрокоманды, предназначенных для управления, определяются проектировщиком микропроцессорного устройства.

В рассматриваемом варианте построения управления операционным устройством потребуется микрокоманда следующего формата:



6.3. ПОСТРОЕНИЕ УПРАВЛЯЮЩЕГО УСТРОЙСТВА

СХЕМЫ УПРАВЛЕНИЯ АДРЕСОМ МИКРОКОМАНДЫ КР1804ВУ1, КР1804ВУ2

Одна схема управления адресом микрокоманды (СУАМ) формирует на своем выходе четыре разряда адреса, по которому из управляющей памяти производится считывание очередной микрокоманды. Объединением определенного числа микросхем СУАМ обеспечивается формирование адреса с требуемой разрядностью.

На рис. 6.7 приведена структурная схема КР1804ВУ1. В микросхеме предусматривается четыре источника адреса, каждый из которых может выдать 4-разрядный двоичный адрес: счетчик микрокоманд (СМК), регистр адреса (РА), стек и входная шина адреса D. Блок выбора адреса (БВА) в соответствии с комбинацией управляющих сигналов S_1, S_0 передает содержимое одного из этих источников адреса на свой выход. Соответствие сигналов S_1, S_0 выбираемому источнику адреса приведено в табл. 6.6.

Рассмотрим несколько подробнее узлы, служащие источниками адреса. Счетчик микрокоманд состоит из 4-разрядного регистра, в который на положительном фронте тактовых импульсов заносится значение, имеющееся на выходе БВА (при значении на входной цепи переноса C_0, равном лог. 0), либо значение выхода БВА, увеличенное

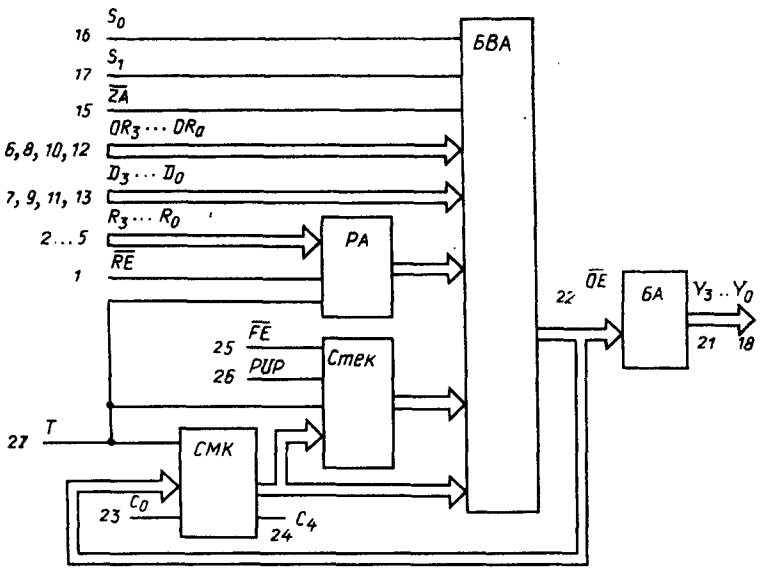


Рис. 6.7. Структурная схема СУАМ КР1804ВУ1

на 1 (при $C_0 = 1$). Узел снабжен выходной цепью переноса C_4. При объединении микросхем СУАМ (рис. 6.8) выходная цепь переноса C_4 подключается к входной цепи переноса C_0 следующей, старшей секции СУАМ. На вход C_0 младшей секции подается уровень лог. 1 (подключением к источнику питания), выходная цепь переноса C_4 старшей секции не используется. Таким образом, в начале каждого тактового периода в СМК заносится значение адреса, увеличенное на единицу по сравнению со значением адреса в предыдущем тактовом периоде. Так формируется адрес микрокоманды, если не нарушается естественный порядок следования адресов, т. е. в отсутствие условных и безусловных переходов.

Таблица 6.6

S_1	S_0	Источник адреса
0	0	(СМК)
0	1	(РА)
1	0	(CT_0)
1	1	(D)

Регистр адреса — 4-разрядный регистр, информация в который может приниматься по 4-разрядной шине $R_3 \ldots R_0$. Вход $\overline{RE}$ является управляющим, на этот вход подается сигнал разрешения записи в РА. При $\overline{RE} = 0$ на положительном фронте тактового импульса информация, поступающая по шине $R_3 \ldots R_0$, принимается в РА.

Стек содержит накопитель из четырех 4-разрядных регистров CT_0, CT_1, CT_2, CT_3 и 2-разрядного указателя стека, хранящего адрес входа в накопитель. Работой стека управляют сигналы $\overline{FE}$ и PUP. Сигнал $\overline{FE}$ служит сигналом разрешения изменения содержимого указателя стека, сигнал PUP — сигналом, определяющим направление изменения содержимого указателя стека (при PUP = 0 — уменьшение, при PUP = 1 — увеличение содержимого указателя стека).

Пусть регистры накопителя CT_0, CT_1, CT_2, CT_3 хранят соответственно адреса A, B, C, D. В дальнейшем под регистром CT_0 будем понимать регистр накопителя, адресуемый указателем стека. Рассмотрим процессы в стеке при различных комбинациях управляющих сигналов $\overline{FE}$ и PUP.

Рассмотрим случай, когда в текущем N-м такте поступает сигнал $\overline{FE} = 1$, при этом значение сигнала PUP безразлично. Значение $\overline{FE} = 1$ задает режим чтения без изменения содержимого указателя стека. При этом в текущем N-м такте из стека на вход БВА поступает

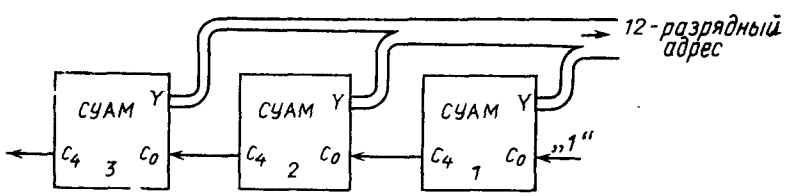


Рис. 6.8. Схема объединения микросхем СУАМ

Таблица 6.7

Такт	Сигналы				СМК	РА
	S_1	S_0	$\overline{FE}$	PUP		
N $N+1$	0	0	0	0	(СМК) (СМК)+1	(РА) (РА)
N $N+1$	0	0	0	1	(СМК) (СМК)+1	(РА) (РА)
N $N+1$	0	0	1	×	(СМК) (СМК)+1	(РА) (РА)
N $N+1$	0	1	0	0	(СМК) (РА)+1	(РА) (РА)
N $N+1$	0	1	0	1	(СМК) (РА)+1	(РА) (РА)
N $N+1$	0	1	1	×	(СМК) (РА)+1	(РА) (РА)
N $N+1$	1	0	0	0	(СМК) ($СТ_0$)+1	(РА) (РА)
N $N+1$	1	0	0	1	(СМК) ($СТ_0$)+1	(РА) (РА)
N $N+1$	1	0	1	×	(СМК) ($СТ_0$)+1	(РА) (РА)
N $N+1$	1	1	0	0	(СМК) (D)+1	(РА) (РА)
N $N+1$	1	1	0	1	(СМК) (D)+1	(РА) (РА)
N $N+1$	1	1	1	×	(СМК) (D)+1	(РА) (РА)

Накопитель стека				Вых. БВА	Комментарий
CT_0	CT_1	CT_2	CT_3		
(CT_0) (CT_1)	(CT_1) (CT_2)	(CT_2) (CT_3)	(CT_3) (CT_0)	(СМК) —	Выталкивание из стека
(CT_0) (СМК)	(CT_1) (CT_0)	(CT_2) (CT_1)	(CT_3) (CT_2)	(СМК) —	Засылка (СМК)
(CT_0) (CT_0)	(CT_1) (CT_1)	(CT_2) (CT_2)	(CT_3) (CT_3)	(СМК) —	Продолжить
(CT_0) (CT_1)	(CT_1) (CT_2)	(CT_2) (CT_3)	(CT_3) (CT_0)	(РА) —	Выталкивание, адрес из РА
(CT_0) (СМК)	(CT_1) (CT_0)	(CT_2) (CT_1)	(CT_3) (CT_2)	(РА) —	Засылка (СМК), адрес из РА
(CT_0) (CT_0)	(CT_1) (CT_1)	(CT_2) (CT_2)	(CT_3) (CT_3)	(РА) —	Адрес из РА
(CT_0) (CT_1)	(CT_1) (CT_2)	(CT_2) (CT_3)	(CT_3) (CT_0)	(CT_0) —	Адрес из CT_0, выталкивание
(CT_0) (СМК)	(CT_1) (CT_0)	(CT_2) (CT_1)	(CT_3) (CT_2)	(CT_0) —	Адрес из CT_0, засылка (СМК)
(CT_0) (CT_0)	(CT_1) (CT_1)	(CT_2) (CT_2)	(CT_3) (CT_3)	(CT_0) —	Адрес из CT_0
(CT_0) (CT_1)	(CT_1) (CT_2)	(CT_2) (CT_3)	(CT_3) (CT_0)	(D) —	Адрес с шины D, выталкивание
(CT_0) (СМК)	(CT_1) (CT_0)	(CT_2) (CT_1)	(CT_3) (CT_2)	(D) —	Адрес с шины D, засылка (СМК)
(CT_0) (CT_0)	(CT_1) (CT_1)	(CT_2) (CT_2)	(CT_3) (CT_3)	(D) —	Адрес с шины D

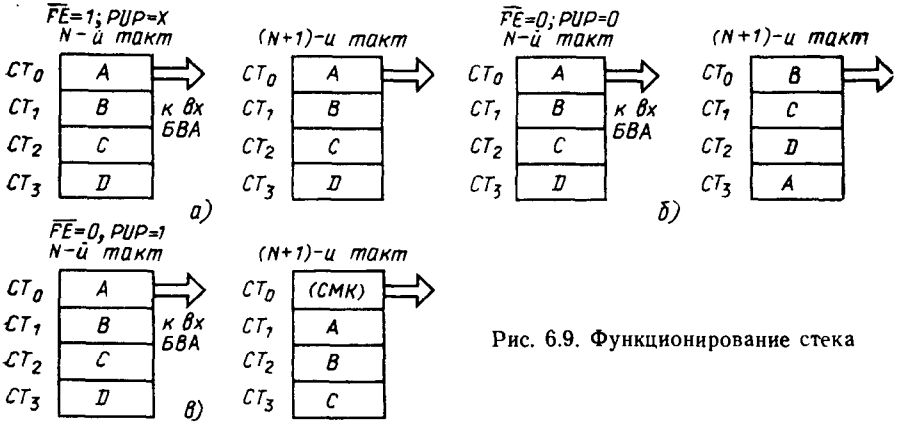


Рис. 6.9. Функционирование стека

содержимое регистра CT_0 (на рис. 6.9, *а* — адрес А). При переходе к следующему $(N + 1)$-му такту размещение информации в регистрах накопителя остается прежним.

Если в N-м такте подается комбинация управляющих сигналов $\overline{FE} = 0$ и PUP = 0, то устанавливается так называемый режим выталкивания из стека. В этом случае в текущем N-м такте на вход БВА выдается хранившийся в регистре CT_0 адрес А (рис. 6.9, *б*), но при переходе к следующему $(N + 1)$-му такту происходит перемещение информации в регистрах, показанное на рис. 6.9, *б*.

При подаче в N-м такте комбинации сигналов $\overline{FE} = 0$ и PUP = 1 устанавливается режим записи, при котором в текущем N-м такте на вход БВА выдается содержимое регистра CT_0 (адрес А), а при переходе к следующему $(N+ 1)$-му такту происходит перемещение информации в регистрах накопителя в обратном направлении (по сравнению с режимом выталкивания) и в регистр CT_0 принимается содержимое счетчика микрокоманд СМК (рис. 6.9, *в*).

В табл. 6.7 показаны процессы в СУАМ при различных комбинациях сигналов S_1, S_0, $\overline{FE}$, PUP.

Рассмотрим комбинацию сигналов $S_1 = 0$, $S_0 = 0$, $\overline{FE} = 0$, PUP = = 0. В N-м такте БВА в качестве источника адреса выбирает СМК (что связано с комбинацией сигналов $S_1 = 0$ и $S_0 = 0$). При переходе к $(N + 1)$-му такту в накопителе стека происходит сдвиг содержимого регистров (что связано с комбинацией $\overline{FE} = 0$, PUP = 0, устанавливающей режим выталкивания). Процессы при других комбинациях управляющих сигналов предлагается рассмотреть самостоятельно.

Стек используется при обращении к подпрограммам. При переходе к подпрограмме адрес ее 1-й микрокоманды выдается на выход СУАМ из РА либо с шины D. Стек устанавливается в режим записи и при переходе к следующему такту в регистр CT_0 накопителя стека принимается (СМК), соответствующее адресу очередной микрокоманды, на

которой было остановлено выполнение главной программы. После окончания выполнения подпрограммы производится выдача адреса из стека и происходит возврат в главную программу.

Кроме входов, предназначенных для приема содержимого четырех рассмотренных выше источников адреса, и входов для подачи сигналов S_1 и S_0, комбинацией значений которых определяется выбор источника адреса, БВА имеет входы $\overline{ZA}$ и $OR_3...OR_0$. Вход $\overline{ZA}$ используется для установки на выходе БВА нулевого значения адреса, обеспечиваемого при подаче $\overline{ZA} = 0$. Вход маски $OR_3...OR_0$ используется для модификации адреса на выходе БВА: может быть установлена «1» в любом разряде адреса путем подачи «1» в соответствующий разряд шины $OR_3...OR_0$.

Адрес с выхода БВА передается на выход $Y_3...Y_0$ микросхемы через буфер адреса (БА), который построен на элементах с тремя состояниями, управляемых сигналом $\overline{OE}$. При $\overline{OE} = 0$ БА устанавливается в открытое состояние, в котором он передает адрес с выхода БВА на выход $Y_3...Y_0$. При $\overline{OE} = 1$ БА устанавливается в 3-е (выключенное) состояние, в котором он отключает микросхему от внешней шины адреса, предоставляя ее в распоряжение других устройств.

Микросхема КР1804ВУ2 отличается от рассмотренной микросхемы КР1804ВУ1 тем, что в ней не предусмотрена шина маски $OR_3...OR_0$, и шины $D_3...D_0$, $R_3...R_0$ объединены в одну общую шину $D_3...D_0$. Эти упрощения позволили в микросхеме КР1804ВУ2 иметь 20 выводов вместо 28 у микросхемы КР1804ВУ1.

СХЕМА УПРАВЛЕНИЯ СЛЕДУЮЩИМ АДРЕСОМ КР1804ВУ3

Данная микросхема предназначена для формирования сигналов управления блоками, входящими в состав управляющего устройства микропроцессора и участвующими в формировании адреса микрокоманды. В частности, эта микросхема формирует управляющие сигналы S_1, S_0, $\overline{FE}$, PUP для микросхемы СУАМ.

Структурная схема микросхемы управления следующим адресом (УСА) показана на рис. 6.10. Дешифратор (ДШ) имеет пять входов, на

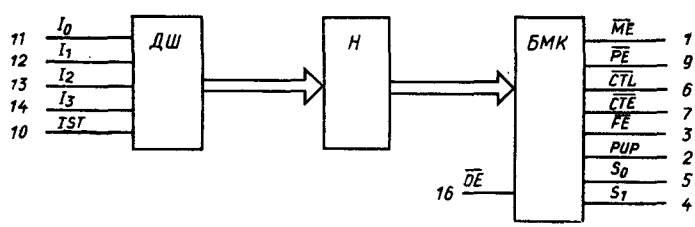


Рис. 6.10. Структурная схема УСА КР1804ВУ3

Таблица 6.8.

Мнемоника	Микрокоманда	Входы					Выходы							
		I_3	I_2	I_1	I_0	TST	S_1	S_0	$\overline{FE}$	PUP	CL	CE	ME	PE
JZ	Переход на нулевой адрес	0	0	0	0	X	1	1	1	1	0	0	1	0
CJS	Условный переход к подпрограмме по (РМК)	0 0	0 0	0 0	1 1	0 1	0 1	0 1	1 0	1 1	1 1	1 1	1 1	0 0
JMAP	Переход по адресу в ОП	0	0	1	0	X	1	1	1	1	1	1	0	1
CJP	Условный переход	0 0	0 0	1 1	1 1	0 1	0 1	0 1	1 1	1 1	1 1	1 1	1 1	0 0
PUSH	Засылка в стек и условная загрузка счетчика	0 0	1 1	0 0	0 0	0 1	0 0	0 0	0 0	1 1	1 0	1 1	1 1	0 0
JSRP	Условный переход к подпрограмме по (РА) или (РМК)	0 0	1 1	0 0	1 1	0 1	0 1	1 1	0 0	1 1	1 1	1 1	1 1	0 0
CJV	Условный переход по адресу вектора	0 0	1 1	1 1	0 0	0 1	0 1	0 1	1 1	1 1	1 1	1 1	1 1	1 1
JRP	Условный переход по (РА) СУАМ или (РМК)	0 0	1 1	1 1	1 1	0 1	0 1	1 1	1 1	1 1	1 1	1 1	1 1	0 0
RFCT	Повторить цикл	1 1	0 0	0 0	0 0	0 1	1 0	0 0	1 0	0 0	1 1	0 1	1 1	0 0
RPCT	Повторить по адресу (РМК), если (Сч) $\neq 0$	1 1	0 0	0 0	1 1	0 1	1 0	1 0	1 1	1 1	1 1	0 1	1 1	0 0
CRTN	Условный возврат из подпрограммы	1 1	0 0	1 1	0 0	0 1	0 1	0 0	1 0	0 0	1 1	1 1	1 1	0 0
CJPP	Условный переход по (РМК) и прием из стека	1 1	0 0	1 1	1 1	0 1	0 1	0 1	1 0	0 0	1 1	1 1	1 1	0 0
LDCT	Загрузка счетчика и продолжение	1	1	0	0	X	0	0	1	1	0	1	1	0
LOOP	Проверка конца цикла	1 1	1 1	0 0	1 1	0 1	1 0	0 0	1 0	0 0	1 1	1 1	1 1	0 0
CONT	Продолжение (переход на следующий адрес)	1	1	1	0	X	0	0	1	1	1	1	1	0
JP	Безусловный переход по (РМК)	1	1	1	1	X	1	1	1	1	1	1	1	0

которые подаются 4-разрядный код $I_3...I_0$ управления микросхемой и сигнал признака ветвления TST, используемый при выполнении условных переходов. Каждой из комбинаций сигналов на входах ДШ соответствует сигнал на определенном из 32 его выходов.

Накопитель (Н) выполнен в виде узла памяти с 32 8-разрядными ячейками. При каждой комбинации значений входных сигналов $I_3...$ $...I_0$ и TST дешифратор производит чтение содержимого определенной ячейки накопителя. Считанное содержимое ячейки представляет собой набор восьми управляющих сигналов, выдаваемых из микросхемы. Этот набор сигналов передается на выходы микросхемы через буфер микрокоманды (БМК), построенный на элементах с тремя состояниями. Управление БМК производится сигналом $\overline{OE}$. При подаче $\overline{OE} = 1$ БМК устанавливается в выключенное состояние, при $\overline{OE} = 0$ на выходы микросхемы поступают управляющие сигналы.

В табл. 6.8 показан набор комбинаций выходных сигналов.Назначение отдельных выходных сигналов и использование их комбинаций будут рассмотрены ниже при рассмотрении общей структурной схемы управляющего устройства.

СТРУКТУРНАЯ СХЕМА УПРАВЛЯЮЩЕГО УСТРОЙСТВА

На рис. 6.11 показан один из возможных вариантов построения управляющего устройства. Рассмотрим его узлы и их взаимодействие в процессе формирования адреса микрокоманды.

Узлом, непосредственно выдающим в управляющую память (УП) адрес очередной микрокоманды, является СУАМ, в качестве которой могут быть использованы микросхемы КР1804ВУ1 или КР1804ВУ2, рассмотренные выше. Считанная из УП микрокоманда на положительном фронте тактового импульса фиксируется в регистре микрокоманд (РМК). Микрокоманда, наряду с полями, предназначенными для управления операционным устройством, содержит поля для управления работой узлов управляющего устройства.

Как отмечалось при рассмотрении СУАМ, источниками адреса в этой микросхеме используются счетчик микрокоманд (СМК), регистр адреса (РА), стек $СТ_0$ и входная шина адреса D. Первые три источника находятся в самой микросхеме. Рассмотрим, откуда поступают адреса в шину D.

Источники адреса на входной шине D СУАМ. Типовая схема управляющего устройства предусматривает возможность подачи адреса на шину D СУАМ от трех различных источников. Одним из этих источников является микрокоманда, принятая в РМК. В поле микрокоманды предусматривается поле адреса, содержимое которого может служить адресом, по которому из УП считывается очередная микрокоманда.

При программировании задач со сложным алгоритмом бывает удобно описывать процессы в алгоритме, используя некоторую группу достаточно сложных операций. Каждая из таких операций представляется

некоторой микропрограммой. В этом случае УП хранит не микропрограмму решаемой задачи, а набор микропрограмм указанных операций. При этом алгоритм задачи представляется последовательностью команд, каждая из которых указывает, какая из набора операций должна выполняться, иначе говоря, какая из хранящихся в УП микропрограмм должна исполняться. Набор команд, образующий программу решения задачи, хранится в оперативной памяти.

Вызванная из оперативной памяти очередная команда содержит информацию о том, какая из микропрограмм должна исполняться. Эта информация заключена в одном из полей команды, содержащем код операции. Код операции с помощью преобразователя начального адреса (ПНА) преобразуется в адрес 1-й микрокоманды, соответствующей данной операции микропрограммы. В этой микрокоманде содержится указание о том, как получить адрес следующей, 2-й микрокоманды

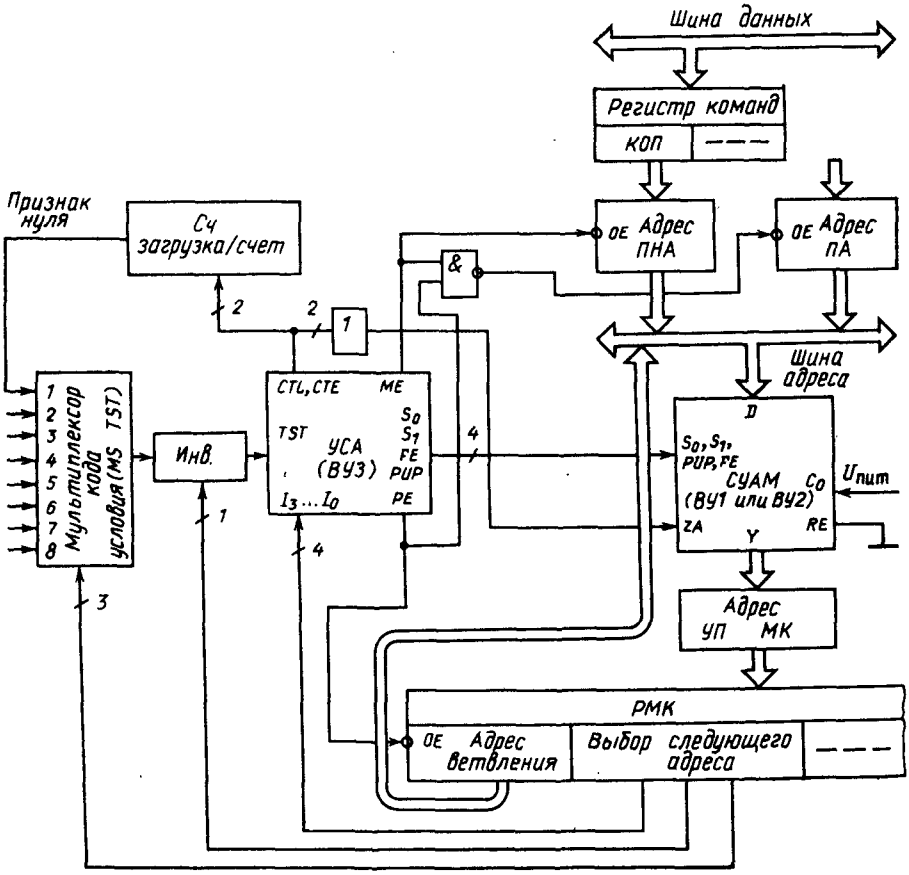


Рис. 6.11. Схема управляющего устройства

и т. д. После окончания выполнения микропрограммы из оперативной памяти вызывается очередная команда, которая определяет адрес 1-й микрокоманды следующей микропрограммы. Таким образом, ПНА служит вторым источником адреса, подаваемого на шину D СУАМ. Он может быть выполнен на ПЗУ либо на программируемой логической матрице (ПЛМ).

Таблица 6.9

$\overline{ME}$	$\overline{PE}$	Сигналы $\overline{OE}$ источников адреса		
		РМК	ПНА	ПА
1	0	0	1	1
0	1	1	0	1
1	1	1	1	0

По запросам от внешних устройств может потребоваться прерывание выполнения текущей программы и переход к исполнению программы внешних устройств. В этих случаях от внешнего устройства должна поступить информация о микропрограмме, исполнение которой запрашивает внешнее устройство. Эта информация в виде некоторой кодовой комбинации, называемой вектором прерывания, подается на вход преобразователя адреса (ПА), который на своем выходе формирует адрес 1-й микрокоманды микропрграммы, исполнение которой запрашивается внешним устройством. Этот адрес подается на шину D, и ПА служит, таким образом, третьим источником адреса на этой шине.

Так как к шине D оказываются подключенными выходы трех источников адреса, необходимо, чтобы выходы этих источников были построены на элементах с тремя состояниями и в каждый момент времени во включенном состоянии находились выходы не более чем одного из источников. Другие источники при этом должны быть отключены от шины D путем установки их выходов в третье (выключенное) состояние. Такое управление указанными источниками осуществляется подачей соответствующих сигналов в цепи $\overline{OE}$ источников. Эти сигналы получаются с помощью управляющих сигналов $\overline{ME}$, $\overline{PE}$, формируемых микросхемой УСА, в соответствии с табл. 6.9: $\overline{PE}$ — сигнал включения выхода регистра микрокоманд РМК, $\overline{ME}$ — сигнал включения выхода преобразователя начального адреса ПНА. При значениях этих сигналов, равных лог. 0, в цепях $\overline{OE}$ соответствующих устройств (РМК либо ПНА) устанавливается сигнал включения. При $\overline{ME} = 1$ и $\overline{PE} = 1$ с помощью элемента И-НЕ формируется уровень лог. 0, который, поступая в цепь управления $\overline{OE}$ преобразователя адреса ПА, переводит это устройство во включениое состояние.

Формирование признаков ветвления. Микросхема ВУ3 имеет вход TST для приема сигнала признака, по значению которого осуществляется ветвление в микропрограмме, т. е. в зависимости от значения признака на входе TST производится переход к одной либо другой микрокоманде. Формирование на выходах ВУ3 управляющих сигналов, необходимых для выполнения такого процесса ветвления, рассмотрим позже. Сейчас же рассмотрим те узлы в управляющем устройстве, которые формируют на входе TST ВУ3 сигнал признака ветвления.

Этот сигнал получается путем выборки с помощью мультиплексора кода условия MS TST одного из сигналов, поступающих на его входы. На входы мультиплексора подается слово состояния, формируемое в регистре слова состояния (РСС) операционного устройства в каждом тактовом периоде. Кроме того, в управляющем устройстве может быть предусмотрен счетчик циклов (Сч). Перед вхождением в цикл в Сч помещается определенное число от некоторого источника и затем в процессе каждого повторного исполнения тела цикла из содержимого Сч вычитается единица. Счетчик снабжается логической схемой, проверяющей его содержимое на равенство нулю. При обнаружении в Сч нуля эта схема выдает сигнал, который может быть использован в качестве признака для выхода из цикла. Сигнал с выхода Сч подается на один из входов мультиплексора и, наряду с разрядами слова состояния, может быть использован в качестве сигнала признака ветвлеиия. Управление мультиплексором осуществляется сигналами, получаемыми с соответствующего поля микрокомаиды.

Выделенный мультиплексором признак передается на вход TST ВУЗ через элемент Инв. Этот элемент управляется одноразрядным полем микрокоманды и может передавать на вход TST ВУЗ либо непосредственно сигнал признака ветвления, поступающий с выхода MS TST, либо инверсное его значение.

Формирование управляющих сигналов микросхемой ВУЗ. Микросхема ВУЗ функционирует под управлением 4-разрядного поля микрокоманды $I_3...I_0$ и значения сигнала признака ветвления на входе TST. При этом ВУЗ создает на восьми своих выходах сигналы, предназначенные для управления работой различных узлов управляющего устройства.

В соответствии с 16 комбинациями кода $I_3...I_0$ образуется 16 видов переходов, обеспечиваемых выходными сигналами ВУЗ (см. табл. 6.8). Процессы при этих переходах схематично показаны на рис. 6.12. Все виды переходов могут быть сгруппированы в 5 типов переходов, как это показано в табл. 6.10. Приведенная в этой таблице форма представления функционирования управляющего устройства удобна для использования при программировании. Рассмотрим по этой таблице и табл. 6.8 выполнение отдельных типов переходов.

При естественном типе переходов адрес следующей микрокоманды на единицу превышает адрес текущей и выбирается из счетчика микрокоманд СМК блока СУАМ. Это обеспечивается выдачей из ВУЗ комбинации сигналов $S_1 = 0$ и $S_0 = 0$. Возможна в СУАМ запись в стек содержимого СМК, которая выполняется под действием выдаваемых ВУЗ управляющих сигналов $\overline{FE} = 0$ и PUP = 1 (при комбинации $I_3...I_0$, равной $0100_2 = 4_{16}$), выдаваемых из микросхемы ВУЗ в СУАМ (см. табл. 6.8). При этом типе переходов возможна загрузка счетчика циклов (при комбинациях $I_3...I_0$, равных 4_{16} и C_{16}), для выполнения которой ВУЗ выдает в Сч комбинацию управляющих сигналов $\overline{CL} = 0$, $\overline{CE} = 1$ (см. табл. 6.8).

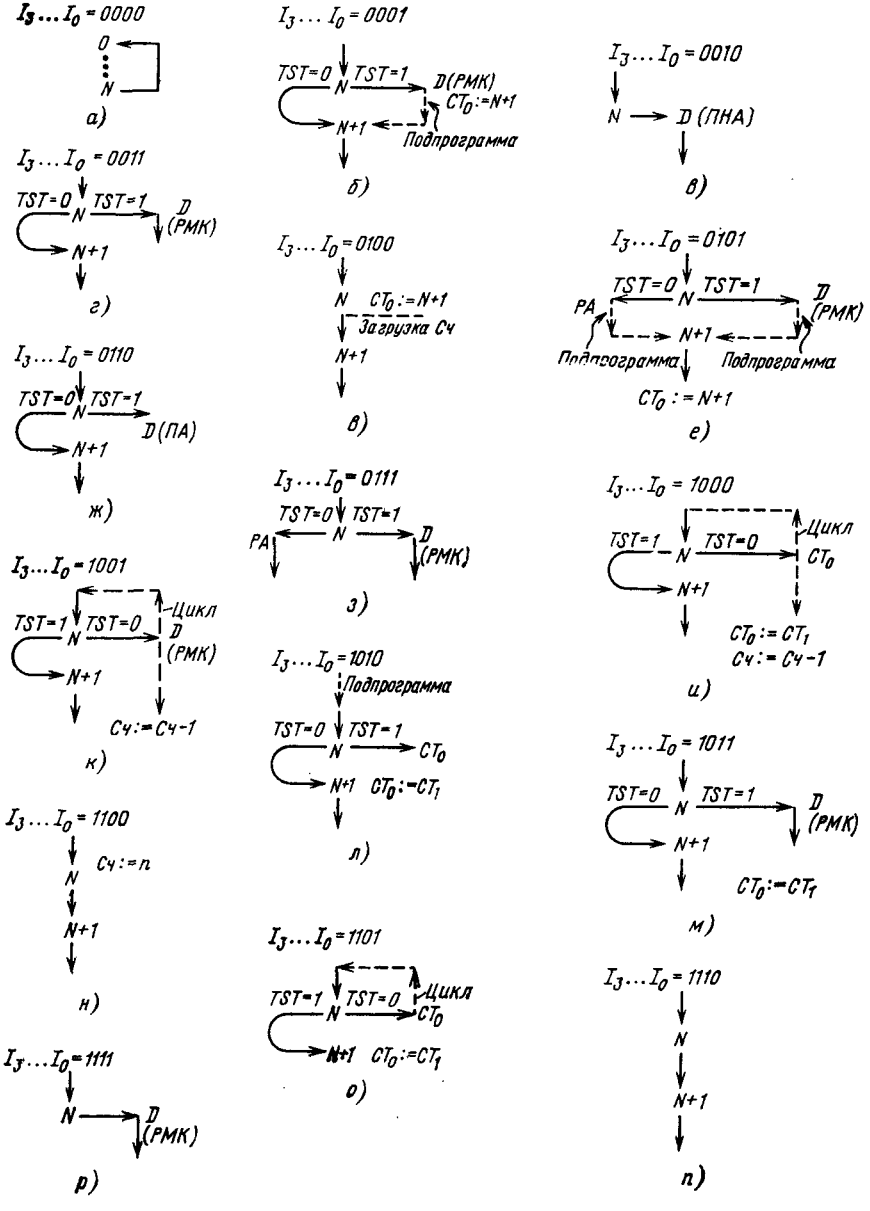


Рис. 6.12. Виды переходов, обеспечиваемые выходными сигналами ВТ3

При безусловных переходах ВУЗ выдает управляющие сигналы $S_1 = 1$, $S_0 = 1$, под действием которых в СУАМ происходит выдача в качестве адреса микрокоманды содержимого шины D.

В этом типе предусмотрены следующие типы переходов. Переход к микрокоманде в ячейке УП с нулевым адресом (при комбинации $I_3 \ldots I_0 = 0$); этот переход выполняется путем выдачи из ВУЗ комбинации $\overline{CL} = 0$ и $\overline{CE} = 0$, которая обеспечивает на входе $\overline{ZA}$ СУАМ уро-

Таблица 6.10

Тип перехода	$I_3 \ldots I_0$ (16-ричный код)	TST	Адрес МК	Стек	Счетчик циклов
Естественный	4	0	(СМК)	Запоминание	—
		1	(СМК)	Запоминание	Загрузка
	C	×	(СМК)	—	Загрузка
	E	×	(СМК)	—	—
Безусловный	0	×	0	—	0
	2	×	(ПНА)	—	—
	F	×	(РМК)	—	—
Условный	3	0	(СМК)	—	—
		1	(РМК)	—	—
	6	0	(СМК)	—	—
		1	(ПА)	—	—
	7	0	(РА)	—	—
		1	(РМК)	—	—
	A	0	(СМК)	—	—
		1	(CT_0)	Выталкивание	—
	B	0	(СМК)	—	—
		1	(РМК)	Выталкивание	—
Условиый переход к подпрограмме	1	0	(СМК)	—	—
		1	(РМК)	Запоминание	—
	5	0	(РА)	Запоминание	—
		1	(РМК)	Запомнинание	—
Условиый переход в циклах	8	0	(CT_0)	—	Уменьшение
		1	(СМК)	Выталкивание	—
	9	0	(РМК)	—	Уменьшение
		1	(СМК)	—	—
	D	0	(CT_0)	—	—
		1	(СМК)	Выталкивание	—

вень лог. 0. Другой вид безусловного перехода предусматривает использование адреса, поступающего на шину D с выхода преобразователя начального адреса ПНА (при $I_3...I_0 = 2$); этот переход осуществляется путем выдачи из ВУЗ сигнала $\overline{ME} = 0$, под действием которого выход блока ПНА переводится в открытое состояние. Третий вид безусловного перехода — переход по адресу из регистра микрокоманд РМК, выполняемый при $I_3... I_0 = F_{16}$, предусматривает выдачу из ВУЗ сигнала $\overline{PE} = 0$, под действием которого открывается выход соответствующей группы разрядов РМК.

При условных переходах значения выдаваемых из ВУЗ сигналов S_1 и S_0 зависят от значения поступающего на вход ВУЗ сигнала признака TST и, таким образом, в зависимости от значения TST в СУАМ используются разные источники адреса. Например, если $I_3...I_0 = 3$, то при TST = 0 выдачей из ВУЗ в СУАМ комбинации $S_1 = 0$, $S_0 = 0$ в качестве адреса используется содержимое СМК СУАМ (т. е. используется адрес, увеличенный на единицу по сравнению с адресом текущей микрокоманды); при TST = 1 выдается комбинация $S_1 = 1$, $S_0 = 1$ и в качестве адреса используется содержимое РМК, передаваемое на шину D ($\overline{PE} = 0$). При $I_3...I_0 = A_{16}$ или B_{16} в стеке возможен процесс выталкивания путем выдачи из ВУЗ в СУАМ комбинации сигналов $\overline{FE} = 0$ и PUP = 0.

Переходы типа условного перехода к подпрограмме имеют особенность, состоящую в том, что путем выдачи из ВУЗ сигналов $\overline{FE} = 0$ и PUP = 1 в СУАМ производится запись содержимого СМК в стек. Тем самым происходит запоминание в стеке адреса микрокоманды, которую необходимо вызвать после окончания выполнения подпрограммы.

Условные переходы в циклах при TST = 0 обеспечивают переход на начало тела цикла по адресу, содержащемуся в $СТ_0$ или РМК; при TST = 1 происходит выход из цикла (с процессом выталкивания из стека, если в нем хранился адрес начала цикла). При $I_3...I_0 = 8$ или 9 происходит уменьшение содержимого счетчика циклов Сч и тем самым равенство содержимого Сч нулю может быть использовано в качестве признака для выхода из цикла.

Выше были рассмотрены схемы операционного и управляющего устройств. Их объединение, образующее схему процессора, показано на рис. 6.13.

Формируемый в СУАМ (ВУ1 или ВУ2) адрес микрокоманды подается на адресный вход управляющей памяти (ПЗУ МК), хранящей микропрограммы. Считанная из памяти общая микрокоманда микропроцессорного устройства в начале тактового интервала принимается в конвейерный регистр (в регистр МК), где она хранится в течение тактового периода. Микрокоманда содержит поля, предназначенные для управления работой отдельных блоков микропроцессорного устройства. Поле МК схемы управления следующим адресом (УСА ВУЗ) совместно с поступающим на вход TST признаком формирует сигналы, под действием

которых в СУАМ (ВУ1 или ВУ2) происходит формирование адреса очередной микрокоманды. При условных и безусловных переходах по адресу ветвления, содержащемуся в МК, УСА (ВУ3) содержимое этого поля МК коммутирует на вход D СУАМ. МК ОУ (с полями A, B, I, C_0, D) поступает в блоки ВС1 и управляет в них выполнением операции. При выполнении операции сдвига соответствующее поле МК ОУ, воздействуя на мультиплексор сдвига, обеспечивает требуемую коммутацию цепей сдвига (PR_3, PR_0, PQ_3, PQ_0). Поле управления мультиплексором признаков определяет признак (содержимое определенного разряда регистра состояния), по которому выполняется условный переход, и передает его на вход УСА. Вновь формируемые признаки принимаются в регистр состояния. В общей микрокоманде могут быть поля, управляющие работой других блоков микропроцессорного устройства (например, блоками оперативной памяти, устройств ввода-вывода).

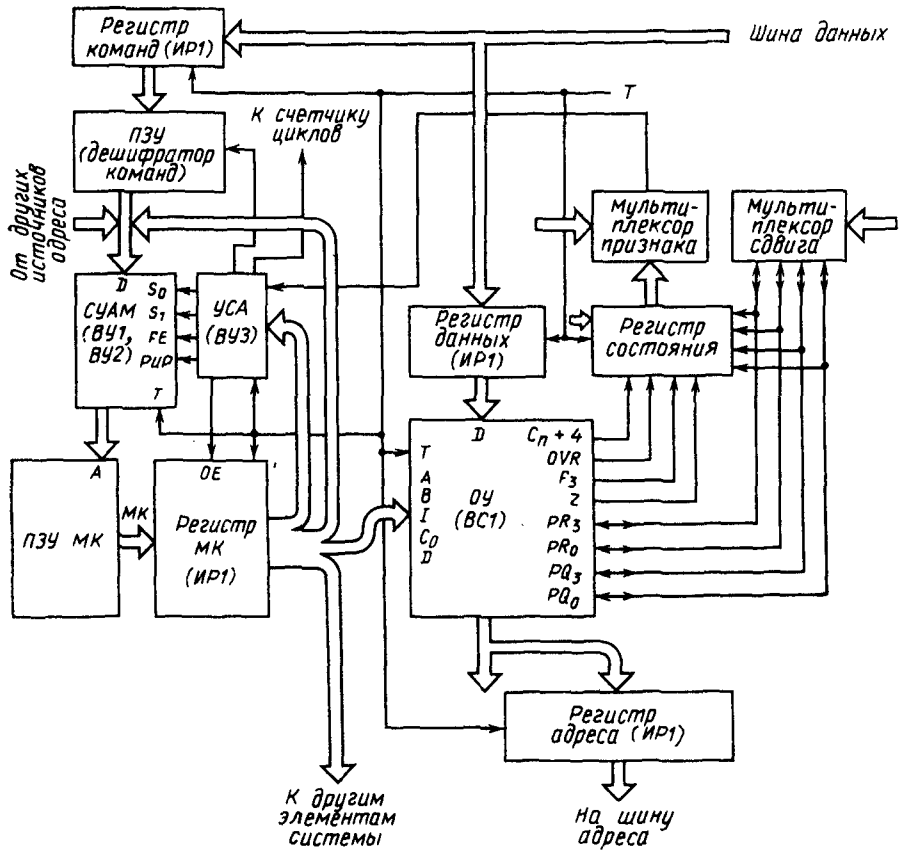


Рис. 6.13. Схема процессора

Используемые в схеме регистры могут быть построены на входящей в состав МПК микросхеме КР1804ИР1. Структурная схема регистра КР1804ИР1 приведена на рис. 6.14. Регистр содержит четыре D-триггера, информация в которые принимается со входов $D_3...D_0$ на положительном фронте тактового сигнала T. Содержимое триггеров передается на выходы $Q_3...Q_0$ непосредственно и на выходы $Y_3...Y_0$ через буферы с тремя состояниями, управляемые сигналом со входа $\overline{OE}$. При $\overline{OE} = 1$ буферы переходят в выключенное состояние (состояние с высоким выходным сопротивлением), запрещая выдачу содержимого регистра на выходы $Y_3...Y_0$.

На рис. 6.15 показаны некоторые применения регистра. На рис. 6.15, *а* показано наращивание разрядности регистра (построение 8-разрядного регистра). На рис. 6.15, *б* представлен 8-разрядный преобразователь кода из последовательной формы в параллельную. В схеме предусмотрены цепи обратной связи, подающие информацию с выходов регистров на входы со сдвигом на один разряд влево (выход Q_0 подключен к входу D_1, выход Q_1 — к входу D_2 и т. д., выход Q_3 регистра *1*, хранящего младшую четверку 8-разрядного содержимого регистров, подается на вход D_0 регистра *2*, хранящего старшую четверку разрядов). На положительном фронте тактового импульса содержимое регистров сдвигается на один разряд влево, а в освобождающийся младший разряд (D_0 регистра *1*) принимается очередной разряд кода, поступающего на вход в последовательной форме. После восьмикратного сдвига с выходов $X_7...X_0$ либо с выходов $W_7...W_0$ принятая в регистр информация может быть снята в параллельной форме. На рис. 6.15, *в* показано использование микросхемы для построения устройства с управляемой двунаправленной передачей информации. При подаче сигнала $\overline{OE} = 1$ на вход регистра *1* и сигнала $\overline{OE} = 0$ на вход регистра *2* выходы $Y_3...Y_0$ регистра *1* оказываются отключенными и информация

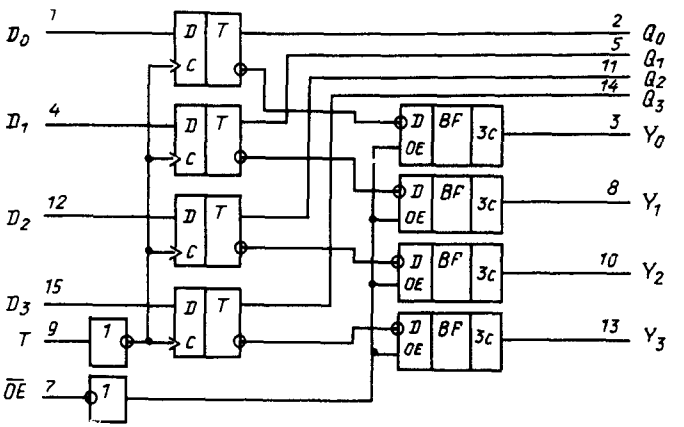


Рис. 6.14. Структурная схема регистра КР1804ИР1

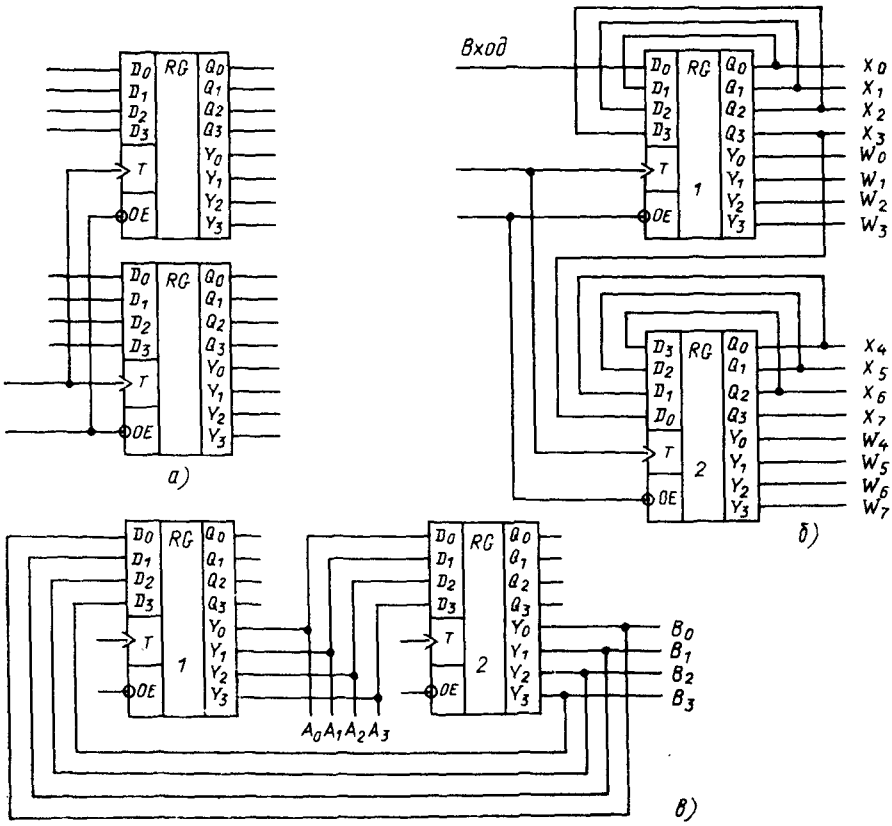


Рис. 6.15. Схемы применений регистра КР1804ИР1:

а) наращивание разрядности регистра; *б)* 8-разрядный преобразователь кода из последовательной формы в параллельную форму; *в)* устройство управляемой двунаправленной передачи информации

с входов $A_3...A_0$ на положительном фронте тактового импульса принимается в регистр *2* и появляется на выходах $B_3...B_0$. Если установить сигналы $\overline{OE} = 0$ на входе регистра *1* и $\overline{OE} = 1$ на входе регистра *2*, то в выключенном состоянии окажутся выходы $Y_3...Y_0$ регистра *2*. При этом информация, поданная на входы $B_3...B_0$ на положительном фронте тактового импульса будет приниматься в регистр *1* и появится на выходах $A_3...A_0$.

6.4. ПРИЕМЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ КОДОВЫХ КОМБИНАЦИЙ

ФОРМАТ МИКРОКОМАНДЫ

Для программирования рассматриваемых ниже задач примем формат микрокоманды, приведенный на рис. 6.16.

Формат микрокоманды содержит две группы полей: одну группу образует МК БМУ, другую — МК ОУ. Поля МК БМУ осуществляют управление устройствами, образующими БМУ, т. е. устройствами, участвующими в формировании адреса очередной микрокоманды для подачи в память МК (см. рис. 6.11). МК БМУ включает в себя четыре поля: поле *Адрес ветвления* используется при выполнении условных и безусловного переходов по адресу в РМК, остальные три поля, объединенные под наименованием *Выбор следующего адреса*, предназначаются для выбора источника адреса следующей микрокоманды. Выбор источника адреса осуществляет УСА (ВУЗ) при задании кода $I_3 \ldots I_0$ и признака TST, поступающего с выхода мультиплексора кода условия через инвертор. Для управления инвертором предназначено одноразрядное поле *Инвертор*, для управления мультиплексором кода условия — 3-разрядное поле *Мультиплексор условия*. Будем считать, что при коде *Инвертор* = 0 информация с выхода мультиплексора кода условия передается на вход TST ВУЗ без инвертирования, а при *Инвертор* = 1 она инвертируется. С помощью 3-разрядного поля *Мультиплексор условия* возможно переключение восьми признаков; примем, что функционирование мультиплексора кода условия задается табл. 6.11.

Микрокоманда ОУ содержит группу полей для управления микросхемами ВС1: $I_8 \ldots I_6$, $I_5 \ldots I_3$, $I_2 \ldots I_0$, предназначенные для задания соответственно кодов управления приемником АЛУ (см. табл. 6.5), кода управления операцией (см. табл. 6.3) и кода управления источниками АЛУ (см. табл. 6.2); одноразрядное поле C_0 для установки информации на входной цепи переноса; поля A и B для выборки регистра в блоке РЗУ. Кроме того, в МК ОУ предусмотрено 2-разрядное поле управления мультиплексорами сдвига *Мультиплексор сдвига*. Виды сдвигов,

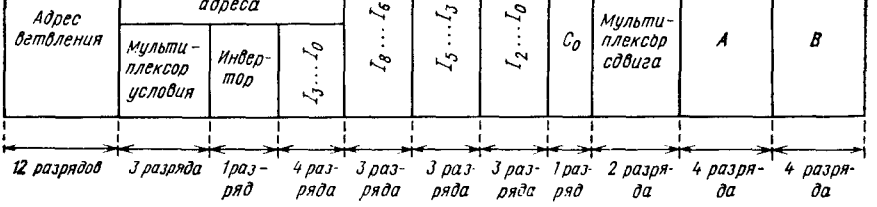


Рис. 6.16. Формат микрокоманды

Таблица 6.11

Мультиплексор условия	Признак на выходе мультиплексора кода условия	Мультиплексор условия	Признак на выходе мультиплексора кода условия
0 0 0	Счетчик циклов	1 0 0	OVR
0 0 1	C_{n+4}	1 0 1	PQ_0
0 1 0	F_3	1 1 0	PR_3
0 1 1	Z	1 1 1	PR_0

задаваемые этим полем, и способы их кодирования примем теми же, что и приведенные на рис. 6.5.

Дополнительные поля, которые могут в МК потребоваться при решении отдельных задач, будем вводить по мере возникновения необходимости в них (при выборе формата МК мы полагали, что в решаемых задачах не будет необходимости в хранении признаков и, следовательно, в пользовании схемой, приведенной на рис. 6.6).

ПОСЛЕДОВАТЕЛЬНАЯ МИКРОПРОГРАММА

Пример 6.1. Требуется реализовать цифровой фильтр 2-го порядка, функционирование которого задается уравнением

$$y(nT) = k_3 \cdot x(nT) + k_4 \cdot x(nT - T) + k_1 \cdot y(nT - T) + k_2 \cdot y(nT - 2T).$$

Для того чтобы микропрограмма не оказалась чрезмерно громоздкой, примем для модулей коэффициентов $k_1 \ldots k_4$ значения, представляемые 4-разрядными двоичными числами, не большими единицы. Пусть $k_1 = -0{,}101_2$, $k_2 = 0{,}001_2$, $k_3 = 1{,}000_2$, $k_4 = -0{,}110_2$.

Представим правую часть уравнения фильтра в форме с положительными коэффициентами, отнеся знак коэффициентов к соответствующим переменным, при которых они стоят:

$$\begin{aligned} y(nT) = {} & 1{,}000 \cdot x(nT) + \\ & + 0{,}110 \cdot (-x(hT - n)) + \\ & + 0{,}101 \cdot (-y(nT - T)) + \\ & + 0{,}001 \cdot y(nT - 2T). \end{aligned}$$

Будем входящие в выражение переменные хранить в дополнительном коде, выделив для них следующие регистры РЗУ ОУ:

$$x(nT)_{доп} \to R_1,\ (-x(nT - T))_{доп} \to R_2,\ (-y(nT - T))_{доп} \to$$
$$\to R_3,\ y(nT - 2T)_{доп} \to R_4,$$

суммирование членов правой части выражения будем производить в регистре R_0, получая в результате суммирования в этом регистре значение $y(nT)$.

Процесс вычисления $y(nT)$ представим следующей последовательностью действий. Формируется сумма первых частичных произведений

(для рассматриваемого выражения это: $-y(nT-T)+y(nT-2T)$, т. е. сумма произведений младших разрядов коэффициентов на соответствующие переменные). Далее, сдвинув полученное значение арифметически на один разряд вправо, прибавим сумму вторых частичных произведений (т.е. произведений вторых разрядов коэффициентов на соответствующие переменные). Сдвинув полученное значение арифметически на один разряд вправо, прибавим сумму третьих частичных произведений. И, наконец, после очередного сдвига, прибавив сумму четвертых частичных произведений, получим результат $y(nT)$.

Рассмотрим эти действия в представленной на рис. 6.17 схеме алгоритма.

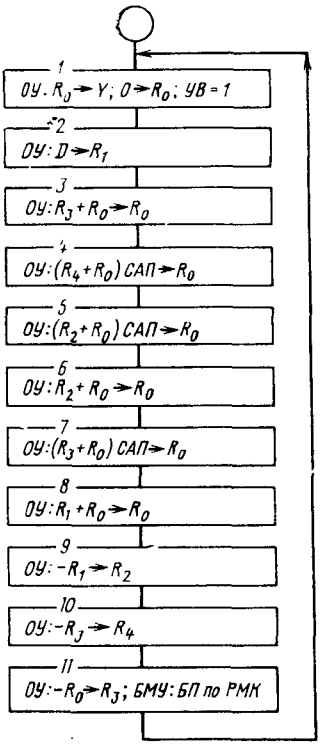


Рис. 6.17. Схема алгоритма цифрового фильтра

В МК1 содержимое регистра R_0 (сформированное в предыдущем повторении цикла значение y) выдается на выход Y с одновременной выдачей на устройство вывода стробирующего сигнала, извещающего устройство о готовности данных для приема. Для выдачи стробирующего сигнала предусмотрим в формате МК дополнительно одноразрядное поле УВ, содержимое которого будет выдаваться на соответствующий вход устройства вывода. В МК1 в это дополнительное поле УВ занесем 1, в остальных МК в этом поле будет установлено значение 0. На этапе записи (при низком уровне синхронизирующего сигнала Т) сбросим регистр R_0 в нуль, подготовив его к накоплению нового значения y в очередном повторении цикла.

В МК2 осуществим прием в регистр R_1 через внешнюю шину D очередного значения $x(nT)$.

В МК3 к содержимому регистра R_0 прибавляется содержимое регистра R_3, хранящего $(-y(nT-T))_{доп}$, т. е. прибавляется первое частичное произведение в третьем слагаемом выражения для $y(nT)$.

В МК4 формируется сумма содержимого регистров R_0 и R_4 (т. е. к содержимому регистра R_0 прибавляется 1-е частичное произведение из четвертого слагаемого выражения $y(nT)$). Эта сумма, полученная на выходе АЛУ, поступает в регистр R_0 через СДА (см. рис. 6.1), где она сдвигается арифметически на один разряд вправо (в микрокоманде указанный сдвиг обозначен САП). Таким образом, в регистре R_0 подготавливается значение, к которому можно прибавлять вторые частичные произведения.

В МК5 подобно тому, как это выполнялось в МК4, формируется

сумма содержимого регистра R_0 и второго частичного произведения 2-го члена выражения y (nT) и после сдвига вправо заносится в регистр R_0.

В МК6 к содержимому регистра R_0 прибавляется третье частичное произведение 2-го члена выражения y (nT).

В МК7 формируется сумма содержимого регистра R_0 и третьего частичного произведения 3-го члена выражения y (nT), эта сумма сдвигается арифметически на один разряд вправо и заносится в регистр R_0.

В МК8 к содержимому регистра R_0 прибавляется четвертое частичное произведение 1-го члена выражения y (nT) (т. е. x (nT)). После этого в регистре R_0 образуется сформированное значение y.

Последующие действия в МК9...МК11 связаны с подготовкой в регистрах R_2...R_4 данных для очередного повторения цикла. Значение x (nT) в текущем цикле в следующем повторении цикла становится значением x ($nT-T$), и взятое с обратным знаком оно должно быть помещено в регистр R_2. Эта пересылка выполняется в МК9. Аналогично, значение y ($nT-T$) текущего цикла в следующем повторении цикла рассматривается в качестве $-y$ ($nT-2T$). Так как в регистре R_3 хранится значение $-y$ ($nT-T$), то перед записью в регистр R_4 (в котором должно храниться y ($nT-2T$)) его знак должен быть изменен на обратный. Эти действия выполняются МК10. Наконец, МК11 передает в регистр R_3, предназначенное для хранения $-y$ ($nT-T$) содержимое регистра R_0 (т. е. последнее сформированное значение y) с измененным на обратный знаком. В этой микрокоманде осуществляется безусловный переход к МК1.

В микропроцессорном устройстве, построенном на МПК К589, для выполнения операции арифметического сдвига вправо требуется

Таблица 6.12

Адрес МК	МК БМУ				МК ОУ							УВ	Пояснения
	Адрес ветвлений	Мультиплексор условия	Инвертор	I_3...I_0	I_8...I_6	I_5...I_3	I_2...I_0	C_0	Мультиплексор сдвига	A	B		
020	X	X	X	1110	010	100	011	X	X	0000	0000	1	МК1
021	X	X	X	1110	011	000	111	0	X	X	0001	0	МК2
022	X	X	X	1110	011	000	001	0	X	0011	0000	0	МК3
023	X	X	X	1110	101	000	001	0	11	0100	0000	0	МК4
024	X	X	X	1110	101	000	001	0	11	0010	0000	0	МК5
025	X	X	X	1110	011	000	001	0	X	0010	0000	0	МК6
026	X	X	X	1110	101	000	001	0	11	0011	0000	0	МК7
027	X	X	X	1110	011	000	001	0	X	0001	0000	0	МК8
028	X	X	X	1110	011	010	100	1	X	0001	0010	0	МК9
029	X	X	X	1110	011	010	100	1	X	0011	0100	0	МК10
02A	020	X	X	1111	011	010	100	1	X	0000	0011	0	МК11

выполнение трех микрокоманд (исполнение которых занимает время в три тактовых интервала). Как мы видим, при построении микропроцессорного устройства на комплекте серии КР1804 для выполнения этой операции сдвига не потребовалось отдельных микрокоманд, реализация операции сдвига совмещается в одном и том же тактовом интервале с выполнением операции в АЛУ. Это обеспечивает высокую скорость обработки данных.

В табл. 6.12 приведена микропрограмма, в которой содержание полей микрокоманды представлено в двоичной системе счисления, адреса ячеек управляющей памяти, в которых производится хранение микрокоманд, представлены в шестнадцатеричной системе счисления.

Так как алгоритм задачи не содержит разветвлений, то микрокоманды помещаются в ячейки управляющей памяти с последовательно нарастающими адресами и в поле $I_3...I_0$ МК БМУ во всех микрокомандах, кроме МК11, должна содержаться соответствующая информация. По табл. 6.8 *Продолжение* (переход на следующий адрес) обеспечивается значением $I_3...I_0 = 1110$. В МК11 предусмотрен безусловный переход к МК1. Это обеспечивается значением $I_3...I_0 = 1111$, по табл. 6.8 соответствующим безусловному переходу по содержимому РМК, т. е. по содержимому поля *Адрес ветвления*.Поэтому в этом поле должен быть записан адрес МК1.

Рассмотрим запись МК ОУ.

Микрокоманда МК1. Для выдачи на выход Y содержимого регистра R_0 необходимо $I_8...I_6 = 010$ (см. табл. 6.5) и $A = 0000$. При этом происходит выдача содержимого регистра R_0 в регистр А, откуда оно выдается на выход Y. Для обнуления регистра R_0 необходимо на выходе АЛУ ОУ получить значение 0. Это обеспечивается выполнением операции конъюнкции (по табл. 6.3 $I_5...I_3 = 100$) н установкой одного из операндов равным нулю (по табл. 6.2 $I_2...I_0 = 010$, 011, 100 или 111). Для того чтобы полученное на выходе АЛУ нулевое значение было принято в регистр R_0, требуется в поле В установить номер этого регистра 0000. Для выдачи на устройство вывода сигнала строба в поле УВ записывается значение 1.

В дальнейшем для определения значений $I_5...I_3$, $I_2...I_0$ рекомендуется пользоваться табл. 6.4.

Микрокоманда МК2. Для пропуска на выход АЛУ значения с шины D можно выбрать $I_5...I_3 = 000$ и $I_2...I_0 = 111$ при $C_0 = 0$ (тот же результат достигается и другими значениями, соответствующими операции *Пропуск* в табл. 6.4). Для записи значения с выхода АЛУ в регистр R_1 необходимо в поле В занести 4-разрядный номер регистра 0001, в поле $I_8...I_6$ при этом должно быть значение 011, обеспечивающее передачу с выхода АЛУ в РЗУ. Значение в поле А безразлично.

Микрокоманда МК3. По табл. 6.4 операция сложения A + B выполняется при $I_5...I_3 = 000$ и $I_2...I_0 = 001$ при $C_0 = 0$. В поля А и В необходимо впнсать соответственно номер регистра R_3 (0011) и регистра R_0 (0000). Результат операции будет занесен по адресу В, т. е. в регистр R_0.

В микрокомандах МК4, МК5, МК7 в поле управления мультиплексором сдвига записано значение, соответствующее арифметическому сдвигу, а в поле $I_8...I_6$ — значение 101, соответствующее сдвигу вправо перед его запнсью в регнстр РЗУ.

Предлагаем подробно запись остальных микрокоманд рассмотреть самостоятельно.

РАЗВЕТВЛЯЮЩИЕСЯ МИКРОПРОГРАММЫ

Пример 6.2. Требуется проанализировать знак числа, хранящегося в регистре R_{12}. Если это число положительное, то к содержимому регистра R_{12} следует прибавить содержимое регистра R_5, в противном случае следует из содержимого регистра R_{12} вычесть содержимое регистра R_4.

На рис. 6.18, *а* приведена схема алгоритма решения данной задачи.

Микрокоманда МК1 предусматривает передачу содержимого регистра R_{12} на выход АЛУ операционного устройства без записи в РЗУ. Знаковый разряд с выхода F_3 старшей секции ВС1 будет принят в регистр состояния. Микрокоманда МК2 осуществляет условный переход по признаку F_3, выбираемому мультиплексором из регистра состояния. При F_3 1 происходит переход к МК3, при $F_3 = 0$ — переход к МК4. Микрокоманда МК3 выполняет суммирование и безусловный переход по адресу ветвления из РМК к микрокоманде, следующей за рассматриваемым фрагментом микропрограммы (эта микрокоманда обозначена *Продолжение*). Микрокоманда МК4 выполняет вычитание и переход к МК *Продолжение* по счетчику микрокоманд СМК блока СУАМ. На рис. 6.18, *б* показаны размещение микрокоманд в управляющей памяти и переходы в процессе выполнения микропрограммы.

В табл. 6.13 приведена микропрограмма.

Рассмотрим запись микрокоманд в микропрограмме.

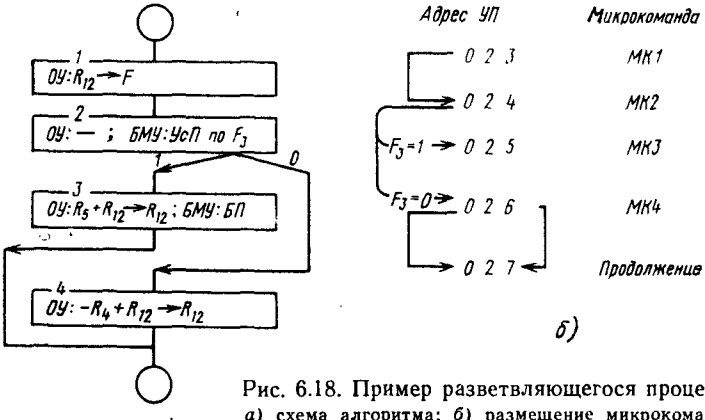


Рис. 6.18. Пример разветвляющегося процесса:
а) схема алгоритма; *б)* размещение микрокоманд в управляющей памяти (от адреса 025 переход к 027)

Микрокоманда МК1. В БМУ предусматривается переход к следующему адресу по СМК в блоке СУАМ, т. е. к адресу, на единицу большему адреса текущей микрокоманды. По табл. 6.8 этот вид перехода выполняется при подаче в блок ВУЗ управляющего кода $I_3...I_0 = 1110$. В ОУ не предусматривается записи в РЗУ; по табл. 6.5 это обеспечивается кодом $I_8...I_6 = 001$. Для передачи содержимого регистра R_{12} на выход АЛУ следует выбрать одну из кодовых комбинаций, соответствующих операции *Пропуск* по табл. 6.4. Например, комбинацией $I_5...I_3 = 011$ и $I_2...I_0 = 100$ осуществляется *Пропуск* содержимого регистра РЗУ, адресуемого содержимым адресного поля А. Поэтому в поле А заносим адрес регистра R_{12} (А = 1100). Содержимое поля В не имеет значения.

Микрокоманда МК2. В БМУ выполняется условный переход по содержимому РМК (т. е. по содержимому поля адреса ветвления в МК). В поле адреса ветвления МК должен быть установлен адрес перехода 026. В поле управления ВУЗ заносится код операции *Условный переход* (по *табл.* 6.8 $I_3...I_0 = 0011$), а мультиплексор кода состояния и инвертор признака должны быть настроены так, чтобы на вход TST ВУЗ подавалось бы значение 0 при переходе по приведенному в МК адресу ветвления. Для этого в соответствии с табл. 6.11 необходимо в поле управления мультиплексором кода состояния установить значение 010 и в поле управления инвертором значение 1. В операционном устройстве запрещается запись в РЗУ установкой $I_8...I_6 = 001$. При этом содержимое остальных полей МК ОУ не имеет значения.

Микрокоманда МК3. МК БМУ предусматривает безусловный переход по содержимому поля адреса ветвления в РМК, для чего в этом поле устанавливается адрес перехода 027, а в поле кода управления ВУЗ $I_3...I_0 = 1111$. В МК ОУ устанавливается код $I_8...I_6 = 011$ записи информации с выхода АЛУ в регистр РЗУ по адресу 1100, содержащемуся в поле В. В поле А — адрес 0101 (адрес регистра R_5).

По табл. 6.4 операции сложения соответствует $I_5...I_3 = 000$ и $I_2...I_0 = 001$ при $C_0 = 0$.

Таблица 6.13

Адрес МК	МК БМУ				МК ОУ							Пояснения
	Адрес ветвления	Мультиплексор условия	Инвертор	$I_3...I_0$	$I_8...I_6$	$I_5...I_3$	$I_2...I_0$	C_0	Мультиплексор сдвига	A	B	
023	×	×	×	1110	001	011	100	×	×	1100	×	МК1
024	026	010	1	0011	001	×	×	×	×	×	×	МК2
025	027	×	×	1111	011	000	001	0	×	0101	1100	МК3
026	×	×	×	1110	011	001	001	1	×	0100	1100	МК4
027	...	...	...	...	...	...	...	...	...	...	...	Продолжение

Микрокоманда МК4. В МК ОУ предусматривается запись в регистр R_{12} (B = 1100) информации с выхода АЛУ ($I_8...I_6$ = 011), а в АЛУ — выполнение операции B — A (A = 0100 и по табл. 6.4 $I_5...I_3$ = 001, $I_2...I_0$ = 001, C_0 = 1).

ЦИКЛИЧЕСКАЯ МИКРОПРОГРАММА

Пример 6.3. Рассмотрим выполнение операции кодового умножения (умножения чисел без знака). Пусть в регистре R_0 содержится множимое, в регистре R_1 — множитель, в регистре R_3 — число разрядов множителя. В регистре R_2 будем формировать группу старших разрядов произведеиия, группу младших разрядов произведения будем накапливать в регистре Q операционного устройства. Операцию будем выполнять следующим способом. Множитель из регистра R_1 передается в регистр Q, после чего устанавливается в исходное нулевое состояние регистр R_2, содержимое регистра Q сдвигается вправо. Затем циклически повторяются следующие действия. Если выдвинутое из регистра Q значение очередного разряда множителя равно 1, то к содержимому регистра R_2 прибавляется множимое, которое берется из регистра R_0, и затем сдвигается вправо содержимое регистров R_2 и Q, объединенных в пару регистров таким образом, чтобы выдвигаемое из регистра R_2 содержимое младшего разряда вдвигалось в освобождающийся старший разряд регистра Q. Если значение выдвигаемого из регистра Q разряда множителя равно нулю, то осуществляется сдвиг без выполнения суммирования. При выполнении суммирования может возникнуть перенос C_4 из старшего разряда. Этот перенос при выполнении сдвига должен быть вдвинут в освобождающийся старший разряд регистра R_2. Среди приведенных на рис. 6.5, *б* видов нет сдвига, удовлетворяющего описанным выше требованиям. Предусмотрев дополнительный (третий) разряд в управлении мультиплексорами сдвига, введем требуемый вид сдвига (рис. 6.19), обозначим этот вид сдвига СП1 и соответствующий ему код управления мультиплексором сдвига примем 100.

На рис. 6.20, *а* приведена схема алгоритма решения данной задачи.

Микрокоманда МК1 в ОУ пересылает содержимое регистра R_1 в регистр Q. Микрокоманда МК2 устанавливает на выходе АЛУ нулевое значение и осуществляет сдвиг вида СП1 содержимого пары регистров R_2 и Q. Микрокоманда МК3 выполняет условный переход по признаку PQ_0 (значению выдвигаемого из регистра Q младшего разряда содержимого при выполнении предыдущей МК). При значении признака PQ_0 = 1 к содержимому R_2 прибавляется множимое из регистра R_0 и результат совместно с содержимым регистра Q сдвигается операцией СП1 и записывается в регистры R_2 и Q. При значении признака PQ_0 = 0 микрокоманда МК5 выполняет лишь сдвиг содержимого пары регистров

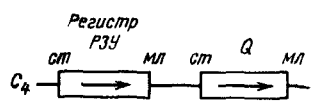


Рис. 6.19. Схема сдвига

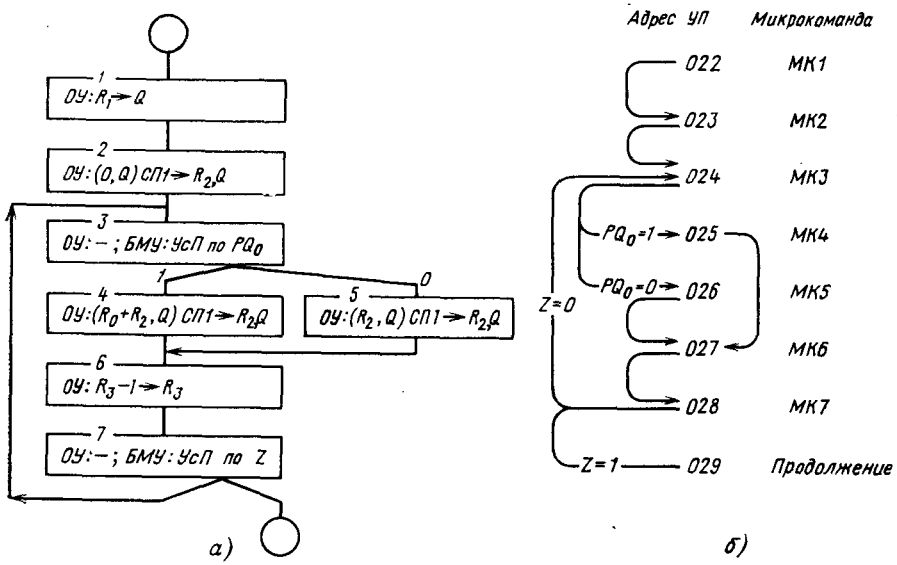


Рис. 6.20. Операция кодового умножения:
а) схема алгоритма; б) размещение микрокоманд в управляющей памяти

R_2, Q. Микрокоманда МК6 вычитает единицу из содержимого регистра R_3, используемого в качестве счетчика числа повторений цикла и затем микрокоманда МК7 выполняет условный переход по признаку Z (признаку нулевого результата). При Z = 0 производится передача управления МК3 и происходит повторение цикла.

На рис. 6.20, *б* приведено размещение микрокоманд в управляющей памяти и показаны переходы в процессе исполнения микропрограммы. В табл. 6.14 приведена микропрограмма рассматриваемой операции умножения.

Таблица 6.14

Адрес МК	МК БМУ				МК ОУ							Пояснения
	Адрес ветвления	Мультиплексор условия	Инвертор	$I_3 \ldots I_0$	$I_8 \ldots I_6$	$I_5 \ldots I_3$	$I_2 \ldots I_0$	C_0	Мультиплексор сдвига	А	Б	
022	×	×	×	1110	000	011	011	×	×	×	0001	МК1
023	×	×	×	1110	100	100	010	×	100	×	0010	МК2
024	026	101	1	0011	×	×	×	×	×	×	×	МК3
025	027	×	×	1111	100	000	001	0	100	0000	0010	МК4
026	×	×	×	1110	100	011	011	×	100	×	0010	МК5
027	×	×	×	1110	011	001	011	0	×	×	0011	МК6
028	024	011	1	0011	001	×	×	×	×	×	×	МК7
029	...	...	...	...	...	...	...	...	...	...	...	Продолжение

Из схемы алгоритма на рис. 6.20, *а* видно, что в каждом повторении цикла выполняются четыре микрокоманды. Если используется внешний счетчик циклов, то отпадает необходимость в микрокомандах МК6 и МК7 и длительность каждого повторения цикла при этом составляет лишь длительность выполнеиия двух микрокоманд, т. е. длительность двух тактовых интервалов. Отсюда следует, что в микропроцессорном устройстве, построенном иа МПК серии КР1804, может быть достигнуто высокое быстродействие.

7. СРЕДСТВА ОТЛАДКИ ПРОГРАММ. ДИАГНОСТИКА МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВ

7.1. ВВЕДЕНИЕ

Процесс разработки микропроцессорных устройств (МПУ) можно представить последовательностью этапов, показанных на рис. 7.1.

На этапе составления технического задания формулируются требования к МПУ и решаемым им задачам. Затем на этапе разработки возможно одновременное выполнение работ по разработке программных и аппаратных средств. В процессе разработки программных средств производится формализация решаемых МПУ задач и построение алгоритма решения; затем составляется программа путем описания алгоритма с использованием некоторого языка программирования (языка кодовых комбинаций, языка Ассемблера и др.). Составленная программа вероят-

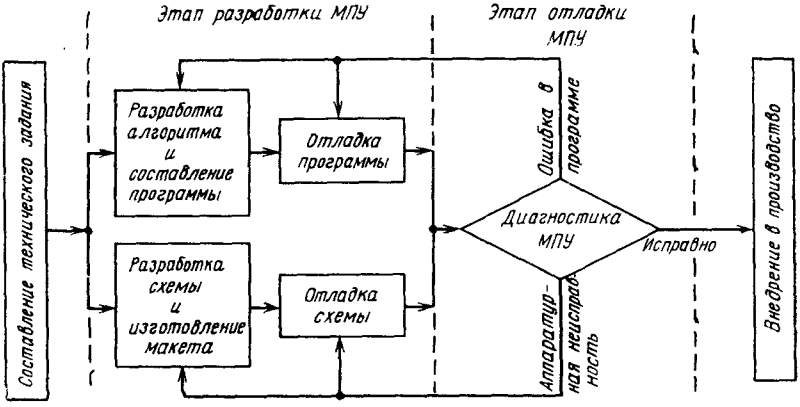


Рис. 7.1. Процесс разработки микропроцессорных устройств

ней всего будет содержать ошибки (их, как правило, не удается избежать и опытным разработчикам программных средств), требуется выявить и устранить ошибки, т. е. отладить программу.

Тем временем разработчики аппаратных средств могут производить разработку схемы МПУ, определив состав блоков, структурную и принципиальную схемы МПУ, по построенной принципиальной схеме производят сборку макета и его испытание для выявления и устранения неисправностей, т. е. осуществляется отладка схемы.

Разумеется, нельзя рассматривать процессы разработки программных и аппаратных средств совершенно независимыми. В выборе структурной схемы МПУ должны принимать участие разработчики программных средств, а в ходе разработки программных средств может выявиться необходимость во внесении изменений в схему МПУ.

Отладка порознь программных и аппаратных средств МПУ не всегда обеспечивает их правильное совместное функционирование. На этапе отладки МПУ производится проверка работы и выявление неисправностей в комплексе. При этом в память макета МПУ вводится полученная на этапе разработки программа и воспроизводится процесс решения задач, для выполнения которых предназначено МПУ, во всех (либо в большинстве) возможных ситуаций, которые могут иметь место на практике. При этом могут обнаружиться погрешности как в программных, так и в аппаратных средствах МПУ, оставшиеся невыявленными на этапе разработки.

Далее окончательно отлаженное МПУ поступает в производство, где его функционирование также должно подвергаться проверке.

Из сказанного очевидно, что в проектировании (а также в производстве и эксплуатации) МПУ значительное место занимают процессы отладки. Пользуются специальными отладочными средствами, облегчающими контроль правильности функционирования МПУ (либо отдельных его частей) и в случае неверного функционирования отыскание неисправностей. Ниже рассматриваются принципы построения таких средств. Причем использование отдельных видов средств эффективно на определенных этапах разработки МПУ. Средства *системного программного обеспечения* используются для трансляции программы на язык кодовых комбинаций и ее отладки. С помощью логических анализаторов можно указать место неисправности с точностью до блока. Для выявления неисправного элемента (микросхемы) в блоке удобно применять сигнатурный анализатор.

7.2. СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Программа, которая пишется разработчиком МПУ с использованием языка Ассемблера или другого языка программирования (кроме языка кодовых комбинаций), называется *исходной*. С ней затем проводятся следующие действия:

редактирование с внесением исправлений, необходимость в которых выявляется при чтении программы программистом;

трансляция — перевод программы с языка программирования на язык кодовых комбинаций, в результате трансляции образуется *объектная программа*;

отладка, в результате которой получается скорректированная и проверенная *рабочая программа*.

Полученная таким образом окончательная программа затем записывается в память МПУ, определяя его функционирование.

Для выполнения всей этой работы, связанной с получением рабочей программы из исходной, используется набор вспомогательных (служебных) программ, объединяемых под названием системное программное обеспечение. Системное программное обеспечение разделяют на *резидентное программное обеспечение* и *кросс-программное обеспечение*. Резидентное программное обеспечение включает в себя те служебные программы (необходимые для разработки прикладных программ), которые используются в микроЭВМ, построенной на той же элементной базе (на МПК той же серии) и имеющей тот же язык кодовых комбинаций, что и МПУ, на котором будут установлены прикладные программы. Кросс-программное обеспечение включает в себя служебные программы, предназначенные для разработки рабочих программ с использованием ЭВМ, построенной на другой элементной базе и имеющей иной язык кодовых комбинаций, чем МПУ, для которого предназначены рабочие программы.

МОНИТОР

Задача программы «Монитор» — управлять работой ЭВМ в процессе трансляции, тестирования, корректировки и ввода прикладных программ пользователя. «Монитор» хранится в ПЗУ.

На рис. 7.2 показан принцип функционирования «Монитора». Пуск осуществляется включением питания микроЭВМ, после чего программа раскрутки устанавливает в счетчике команд адрес первой команды «Монитора». Далее «Монитор» переводится в режим ожидания, в котором он находится до тех пор, пока пользователь нажатием соответствующей клавиши на терминале не запросит выполнение одной из подпрограмм «Монитора». После выполнения вызванной подпрограммы «Монитор» возвращается в режим ожидания.

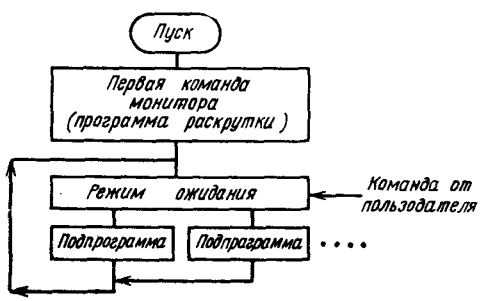


Рис. 7.2. Принцип функционирования «Монитора»

«Монитор», таким образом, управляет несколькими подпрограммами. Из них укажем две подпрограммы. Первая

«Загрузчик» — подпрограмма ввода данных с некоторого носителя данных (например, с накопителя на магнитных дисках). «Загрузчик» включает в работу устройство ввода, обеспечивая ввод программы, хранящейся на внешнем носителе, и размещение ее в памяти ЭВМ. Вторая — подпрограмма вывода данных иа внешний носитель информации (например, на магнитный диск).

РЕДАКТОР ТЕКСТА

Написаниая на языке Ассемблера исходная программа должна быть отперфорирована на ленте либо записана на магнитном диске. Чтобы избежать ошибок при такой передаче программы на внешний носитель, используется программа «Редактор текста». Получение программы на внешнем носителе показано на рис. 7.3.

После подачи оператором команды ввода «Редактора», «Монитор» выходит из режима ожидания и вызывает соответствующую подпрограмму «Загрузчик», которая управляет работой устройства ввода. С внешнего носителя информации (например, с перфоленты либо магнитного диска) считывается и вводится в память микроЭВМ программа «Редактор». «Монитор» возвращается в режим ожидания.

Оператор подает с терминала команду запуска «Редактора». При этом «Монитор» передает управление находящейся в оперативной памяти программе «Редактор». С этого момента может начаться ввод ассемблерных команд. Каждый набранный на терминале символ высвечи-

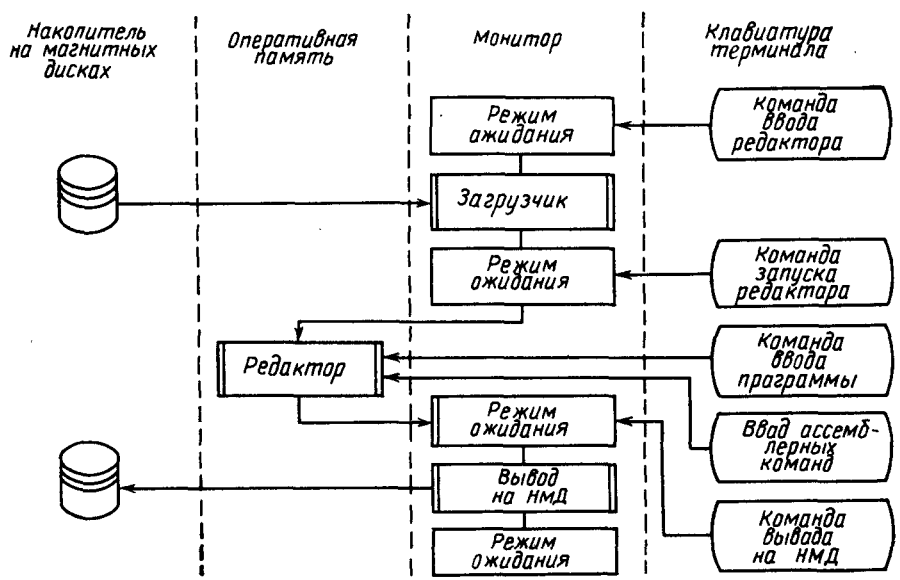


Рис. 7.3. Процессы при работе с «Редактором текста»

вается на экране дисплея и засылается в память. Когда введено несколько команд программы, оператор может проверить правильность введенных символов. Если он обнаружит ошибку, то может ввести соответствующую команду (строку) снова.

Когда вся программа после ее проверки оказалась введенной в память микроЭВМ, «Редактор» с помощью программы «Моиитор» может вызвать подпрограмму вывода данных на внешний носитель. Эта подпрограмма выводит на внешний носитель прикладную программу, находящуюся в оперативной памяти ЭВМ. Таким образом получают на внешнем носителе исходную программу. После этого «Монитор» переходит в режим ожидания.

АССЕМБЛЕР

Программа «Ассемблер» — служебная программа, преобразующая исходную программу, написанную на языке Ассемблера, в объектную. «Ассемблер» может выявлять синтаксические (несмысловые, связанные с несоблюдением правил записи команд) ошибки в тексте исходной программы. Например, могут обнаруживаться ошибки, связанные с повторным использованием метки, отсутствием в команде указания об операнде и т. п.

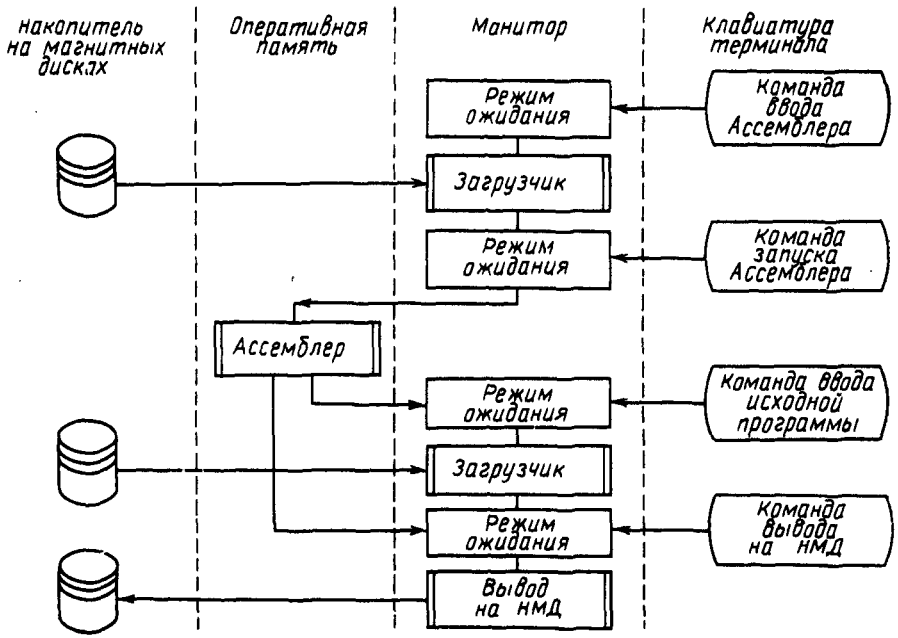


Рис. 7.4. Процесс получеиия объектиой программы

«Ассемблер» несколько раз просматривает исходный текст программы. В процессе первого просмотра исходной программы «Ассемблер» формирует список синтаксических ошибок. Ассемблирование не может быть прервано для исправления ошибок. Если обнаружена ошибка, то соответствующая команда может быть исправлена с помощью «Редактора», после чего процесс ассемблирования следует повторить.

Процесс получения объектной программы иллюстрируется рис. 7.4.

Оператор подает команду ввода «Ассемблера». Монитор выходит из состояния ожидания и передает управление подпрограмме «Загрузчик». Последняя, управляя работой устройства ввода, вводит «Ассемблер» с внешнего носителя в оперативную память микроЭВМ. По окончании этой операции «Монитор» возвращается в режим ожидания. Затем оператором может быть подана команда запуска «Ассемблера». «Ассемблер» вызывает «Загрузчик», с помощью которого производится ввод исходной программы с внешнего носителя в память микроЭВМ. После этого начинается процесс трансляции.

Если в процессе трансляции не обнаруживаются ошибки, то «Ассемблер» вызывает подпрограмму вывода, которая управляет процессом выдачи объектной программы на внешний носитель.

Программа «Ассемблер» может располагаться в ПЗУ. При этом из рассмотренного процесса очевидно, что этап ввода «Ассемлера» в память исключается.

ОТЛАДЧИК

С помощью «Редактора» и «Ассемблера» получают объектную программу, не содержащую синтаксических ошибок. Однако в программе могут быть смысловые (иначе семантические) ошибки (например, использование команд, выполняющих не те действия, которые должны быть при решении задачи; неверное использование меток; смысловые ошибки в схеме алгоритма и т. д.). В этом случае объектная программа, введенная и выполненная в микроЭВМ, не дала бы правильных результатов. Необходимо протестировать программу. Это выполняется с помощью программы «Отладчик».

«Отладчик» — служебная программа, обеспечивающая:

индикацию содержимого ячеек памяти. Введя с клавиатуры адрес памяти, можно вызвать содержимое ячейки на экран дисплея либо на печать;

изменение содержимого ячейки памяти; для этого, используя клавиатуру терминала, программист-оператор вводит адрес ячейки и новое содержимое, которое должно быть установлено в ячейке;

индикацию содержимого регистров центральиого процессора (регистров общего назначения, счетчика команд, аккумулятора, регистра адреса, указателя стека, регистра признаков) на экране дисплея;

останов по адресу. Если указать один или несколько адресов (точек останова), то выполнение объектной программы будет прерываться

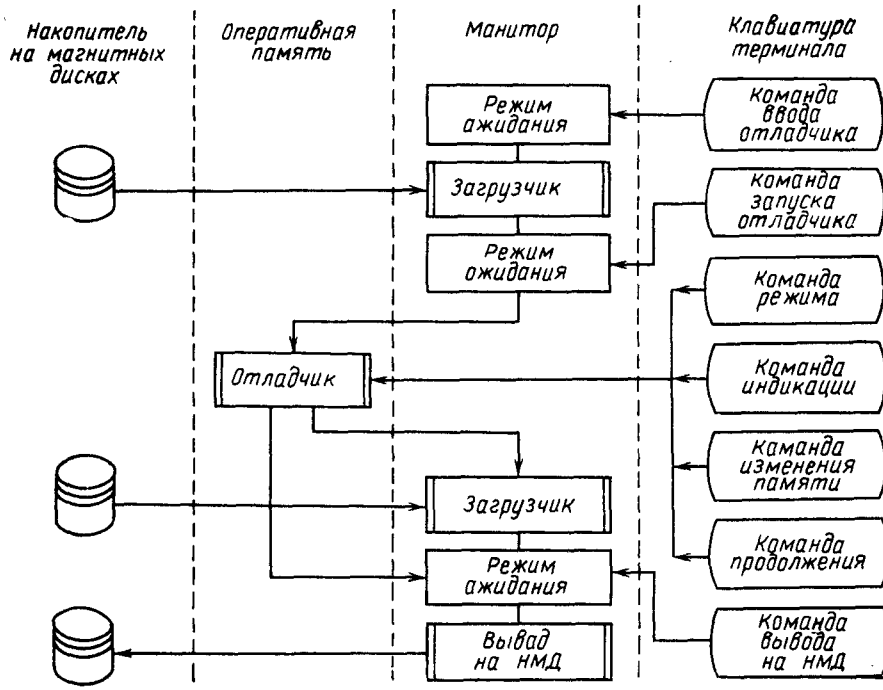


Рис 7.5. Процессы при работе с «Отладчиком»

при достижении соответствующих команд, находящихся в ячейках памяти с указанными адресами. После останова можно, проверяя содержимое памяти и регистров, установить правильность работы выполненной части программы. Если возникают ошибки, то их можно исправить, изменяя содержимое памяти. После этого программист подает комаиду на продолжение выполнения программы;

пошаговый режим, в котором происходит останов после каждой выполненной команды, либо n-тактный режим с остановом после указанного числа n команд. После выполнения заданного числа команд «Отладчик» переходит в режим ожидания команд: «Продолжить», «Индицировать содержимое памяти» или «Индицировать содержимое регистров».

При работе с «Отладчиком» выполняются следующие действия (рис. 7.5).

Команда ввода «Отладчика» выводит «Монитор» из режима ожидания, вызывается «Загрузчик». Производятся чтение «Отладчика» с внешнего носителя и ввод в память.

По команде запуска «Отладчика» последний передает управление «Загрузчику», который считывает с внешнего носителя объектную программу и вводит ее в память микроЭВМ.

Далее программист-оператор с клавиатуры может задать либо значение параметра n, либо совокупность адресов (точек останова), в которых будет останавливаться выполнение программы. Затем он выдает команду продолжения и в зависимости от введенного значения n выполняется одна либо несколько команд, после чего «Отладчик» возвращается в состояние ожидания.

Программист-оператор имеет возможность посмотреть содержимое ячеек памяти или регистров и произвести изменение их содержимого. Затем по команде продолжения обеспечивается выполнение следующей группы команд.

После завершения отладки программы (с помощью рассмотренных средств) «Отладчик» передает управление подпрограмме вывода, которая выводит окончательный вариант объектной программы на внешний носитель. Эта отлаженная программа называется рабочей.

ИМИТАТОР

С помощью «Ассемблера», «Редактора», «Монитора» и «Отладчика» можно получать откорректированную объектную программу (т. е. рабочую программу). Оборудование, на котором происходит процесс превращения исходной программы в рабочую, называется системой разработки. На одной и той же системе разработки можно имитировать различные МПУ. Это осуществляется с помощью служебиой программы, называемой «Имитатор», которая выполняет следующие функции:

имитацию характеристик конкретного типа микропроцессора. Это означает, что система команд имитируемого микропроцессора должна быть преобразована в систему команд микропроцессора, лежащего в основе системы разработки;

имитацию периферийных устройств МПУ, для которого создается рабочая программа;

отладку объектной программы на микроЭВМ с элементной базой и языком кодовых комбинаций, отличными от элементной базы и языка кодовых комбинаций МПУ, для которого разрабатывается программа. Отсюда следует, что в состав имитатора входит «Отладчик».

7.3. ЛОГИЧЕСКИЕ АНАЛИЗАТОРЫ

На рис. 7.6 приведена упрощенная структурная схема логических анализаторов.

При испытании аналоговых устройств успешно применяются обычные осциллографы, с помощью которых на экране отображается временной ход мгновенных значений исследуемых процессов. В цифровых устройствах мгновенные значения не представляют интереса. Важно лишь знать, какому из двух уровней лог. 0 и лог. 1 соответствуют эти мгновенные значения. Например, в логических схемах технологии типа ТТЛ любые напряжения, превышающие 2,4 В, производят эффект,

соответствующий лог. 1, а уровни напряжения, не превышающие 0,4 В, — эффект, соответствующий лог. 0. Напряжения, появляющиеся в различных точках МПУ, логическим анализатором преобразуются в последовательность логических уровней (последовательность уровней лог. 0 и лог. 1). Это преобразование осуществляется с помощью компараторов, на которые поступают входные напряжения, снимаемые с цепей шин данных, адреса, управления (в показанной на рис. 7.6 схеме предусмотрено 16 входов, на которые могут быть поданы напряжения с шин данных и адреса, где информация представляется в параллельной форме, либо могут быть приняты сигналы с различных цепей шины управления для установления правильности временных соотношений в них).

Напряжение, поступающее на каждый вход анализатора, с помощью компаратора сравнивается с пороговым напряжением (которое для схем ТТЛ может быть выбрано (2,4 + 0,4)/2 1,4 В, для других схем оно выбирается аналогично), устанавливаемым с помощью генератора порогового напряжения. Если входное напряжение превышает уровень порогового напряжения, принимается, что входное напряжение соответствует уровню лог. 1, в противном случае оно принимается соответствующим уровню лог. 0.

Таким образом, напряжение, поступающее на каждый вход, преобразуется в двухуровневые сигналы на выходе компаратора. Этот процесс показан на рис. 7.7.

В моменты, соответствующие тактовым импульсам, полученный на выходах компараторов многоразрядный код поступает в буферный регистр.

Из буферного регистра тактовыми импульсами информация передается в запоминающее устройство (ЗУ) и логический компаратор.

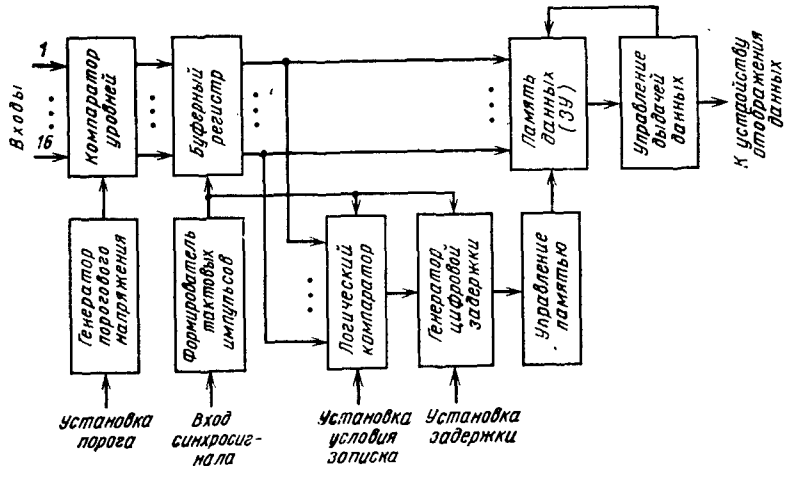


Рис. 7.6. Структурная схема логического анализатора

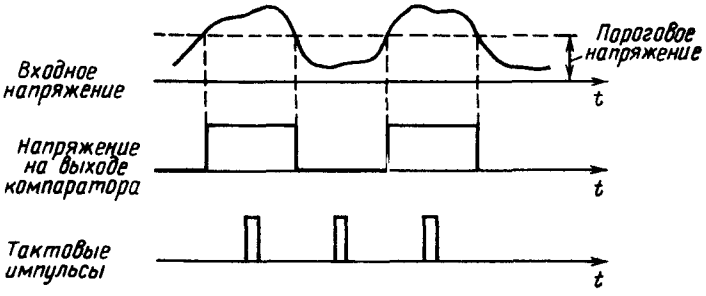


Рис. 7.7. Преобразование входного напряжения в логические уровни

По характеру протекающих процессов ЗУ можно представить состоящим из сдвиговых регистров, число которых равно числу разрядов буферного регистра. В моменты тактовых импульсов содержимое сдвиговых регистров сдвигается и в освобождающийся разряд каждого из них принимается содержимое соответствующего разряда буферного регистра. При этом выдвигаемая из сдвиговых регистров информация теряется. Таким образом, в таких регистрах удерживаются данные, поступившие из буферного регистра за определенное число последних тактов (например, за последние 16 тактов). С наступлением очередного такта принимается новое слово, а самое раннее принятое слово теряется.

Такие процессы можно воспроизвести в ЗУ типа стек, если предусмотреть в нем 16 16-разрядных ячеек и 4-разрядный счетчик в качестве указателя стека. Тактовые импульсы изменяют на единицу содержимое указателя стека и по образовавшемуся адресу производится запись в стек информации, поступающей из буферного регистра. Так как работа счетчика носит циклический характер, то через каждые 16 тактов в ячейке памяти запоминается новая информация, при этом ранее введенное в нее слово теряется.

Если в процессе отладочных работ обнаружится выдача МПУ неверных результатов, то возникает необходимость выявления точек программы, которые могут быть виновными в возникновении такого результата. Определение таких точек в программе требует просмотра содержимого шин за некоторое число тактов на участках программ, где происходит выполнение команд, подозреваемых в ошибочном исполнении. Эту информацию можно получить из рассмотренного выше ЗУ. Пусть ввод в ЗУ должен прекращаться в момент, когда в подключенной ко входу анализатора шине (и, следовательно, в буферном регистре) появляется слово, возникающее при выполнении данной команды. Тогда в ЗУ оказываются 16 данных, принятых в течение 16 предшествующих тактов. Можно после обнаружения в буферном регистре указанного слова закрыть доступ в ЗУ новых данных с задержкой на некоторое число k тактовых периодов. При этом в ЗУ окажутся зафиксированными данные, принятые в течение $16 - k$ предшествующих и

k последующих тактов, отсчитываемых относительно такта, в котором обнаружено данное слово.

В схеме логического анализатора на рис. 7.6 выявление момента, когда в буферном регистре появляется заданное слово, осуществляется логическим компаратором, на входы которого поступает информация из буферного регистра и слово, набранное на переключателях оператором. Генератор цифровой задержки по установленному оператором значению задержки, воздействуя на устройство управления памятью, закрывает доступ в ЗУ новых данных в определенном такте.

С помощью устройства управления выдачей данных содержимое ячеек ЗУ отображается на экране. При этом могут быть использованы различные формы представления данных. Этим данным может быть придана форма временных диаграмм, таблиц данных либо карт состояния. Рассмотрим эти формы вывода и отображения данных.

Форма временных диаграмм удобна при отладке аппаратных средств, когда возникает необходимость анализа взаимных временных положений между сигналами, возникающими в различных цепях, выявления ложных импульсов. На рис. 7.8 показан вид получаемых на экране временных диаграмм. Получение таких диаграмм достигается в результате последовательного циклического считывания содержимого ячеек ЗУ и подачи разрядов считанных данных в отдельные каналы многоканального осциллографа. Такие временные диаграммы, одновременно отображающие логические уровни сигналов, получаемых по многим каналам (например, по 8 и 16 каналам), позволяют достаточно полно проанализировать функционирование определенных блоков цифрового устройства. Представленные на рис. 7.8 диаграммы соответствуют работе двоичного счетчика. На диаграмме, обозначенной номером 0, представлена последовательность синхронизирующих импульсов, на

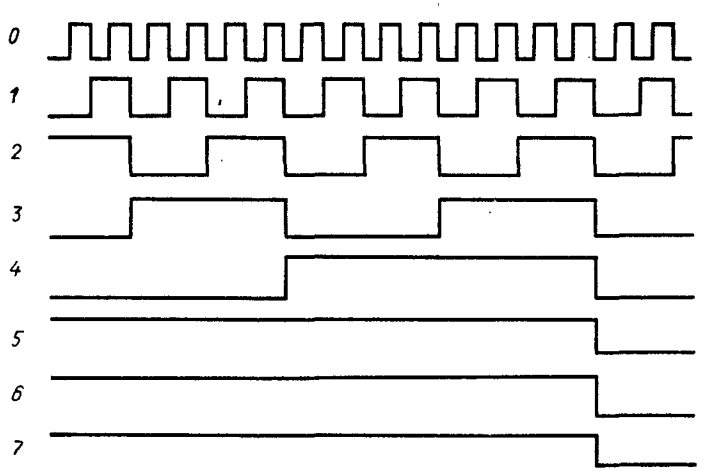


Рис. 7.8. Временные диаграммы на экране логического анализатора

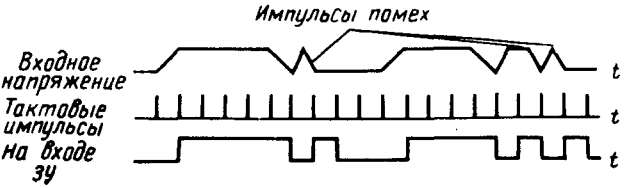


Рис. 7.9. Процессы регистрации коротких импульсов

каждой из последующих диаграмм — состояние на выходах соответственно 1-го, 2-го и т. д. разрядов счетчика.

Для регистрации коротких импульсов помех, возникающих в интервале времени между моментами появления синхронизирующих импульсов, можно пользоваться следующими приемами. Частоту тактовых импульсов, под действием которых поступающая на входы анализатора информация считывается и передается в буферный регистр, а затем в ЗУ, выбирают в несколько раз более высокой, чем частота синхронизирующего сигнала испытываемого устройства. Кроме того, для регистрации импульсов, длительность которых меньше периода следования тактовых импульсов, можно в каналах использовать триггеры, которые взводятся от таких импульсов. С появлением тактовых импульсов состояние триггера передается в буферный регистр, после чего триггер сбрасывается в исходное состояние. Эти процессы показаны на рис. 7.9.

Отображение содержимого ЗУ в форме таблиц данных удобно на этапе комплексной отладки МПУ, когда производится совместное испытание ранее порознь отлаженных программных и аппаратных средств. При этой форме содержимое ЗУ отображается в виде чисел, представляемых в различных системах счисления (двоичной, восьмеричной, шестнадцатеричной). Каждое такое число изображает комбинацию логических уровней сигнала, поступающего на входы анализатора, в некоторый тактовый момент. Например, на рис. 7.10 показана таблица данных, поступающих на входы логического анализатора с выходов разрядов двоичного счетчика.

При использовании формы карт состояния разряды выводи-

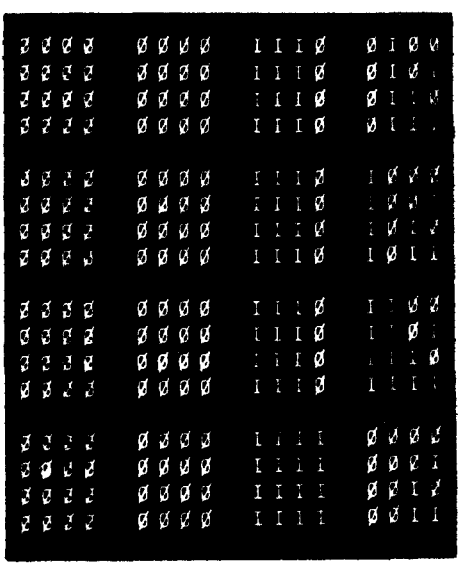


Рис. 7.10. Таблица данных на экране логического анализатора

мых из ЗУ анализатора данных разбиваются на две группы. Каждая группа разрядов подается на отдельный цифро-аналоговый преобразователь информации (ЦАП). Выходное напряжение одного ЦАП используется в качестве напряжения горизонтальной развертки электронно-лучевой трубки, напряжение другого ЦАП — в качестве напряжения вертикальной развертки трубки. При этом каждому значению выдаваемых из ЗУ данных соответствует определеиное положение точки на экране трубки. Последовательности поступающих из ЗУ данных соответствует последовательность точек, образующих на экране некоторую фигуру. Такая форма отображения удобна для контроля на этапе производства МПУ или их эксплуатации. Если форма фигуры при исправном функционировании МПУ известна, то при наличии неисправности форма этой фигуры резко отличается и по этому признаку выявляется неправильное функционирование МПУ.

7.4. СИГНАТУРНЫЙ АНАЛИЗАТОР

С помощью логических анализаторов может быть локализована неисправность с точностью до некоторого крупного элемента замены (некоторого коиструктивного элемента), например платы. Выявленный неисправный конструктивный элемент может быть при этом заменен исправным, а неисправный элемент подлежит ремонту. Восстановление работоспособности неисправного элемента требует локализации неисправности в нем с точностью до компонента (микросхемы), т. е. выявления неисправно функционирующего компонента и замены его на исправиый.

Таким образом, при проведении ремонтных работ основная трудность состоит в выявлении неисправного компонента. В аналоговом оборудовании для облегчения выполнения этой задачи используются схемы, на которых указываются уровни и форма напряжений в различных точках исправно функционирующей аппаратуры. Поиск неисправного компонента при этом ведется следующим образом. На входы испытываемого устройства подаются соответствующие воздействия и затем с помощью вольтметра и осциллографа проверяются сигналы в точках, для которых их уровни или форма указаны в схеме. Компонент, у которого выходные сигналы не соответствуют указанным в схеме, хотя входные сигналы правильны, является неисправным.

Использование такого приема выявления неисправного компонента в цифровой аппаратуре невозможно. Это связано с тем, что в цифровой аппаратуре оценка сигнала должна осуществляться не по форме напряжения, а по последовательности логических уровней (т. е. последовательности лог. 0 и лог. 1), которой соответствует это иапряжение. Эти последовательности, как правило, имеют большую длину, и отличить осциллограмму верной последовательности от осциллограммы неверной практически невозможно.

Сигнатурный метод позволяет сжать длинные последовательности

логических уровней в отдельных точках аппаратуры в короткие (обычно представляемые 4-разрядиыми числами в шестнадцатеричной системе счисления), называемые *сигнатурами*. Такие сигнатуры для исправио функционирующей цифровой аппаратуры (подобно форме напряжения в точках аналоговой аппаратуры) могут приводиться в схеме. При этом способ поиска неисправного цифрового компоиента сводится к преобразованию в сигнатуры логических последовательностей, имеющихся в отдельных точках, и проверке соответствия полученых сигиатур приведенным в документации для этих точек эталонным сигнатурам. Если обнаруживается несоответствие сигнатур, то последовательно проверяются точки в направлении ко входам аппаратуры, пока не выявится элемент, у которого при правильных входных сигнатурах окажутся неверными выходные сигнатуры. На рис. 7.11 показан пример записи эталонных сигнатур в схеме дешифратора.

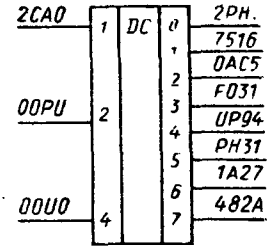


Рис. 7.11. Пример записи эталоиных сигиатур

Рассмотрим формирование сигнатуры. Для получения сигнатуры длинной логической последовательности используется сдвиговый регистр с цепями линейной обратной связи. Пример такого регистра показан на рис. 7.12. Регистр имеет 16 разрядов. Содержимое регистра сдвигается влево тактовыми импульсами с тем же периодом, что и длительность тактов входной двоичной последовательности. В каждом тактовом интервале содержимое регистра сдвигается влево, после чего содержимое 7, 9, 12 и 16-го разрядов регистра и значение входиой последовательности суммируются по модулю 2 и полученое значеиие заносится в освободившийся после сдвига 1-й разряд регистра. После определенного числа тактов, обычно существенно большего числа разрядов регистра, в регистре образуется значение, являющееся сигнатурой данной входной последовательности. Для облегчения чтения сигнатуры она представляется в шестнадцатеричной системе счисления, т. е. содержимое каждой четверки разрядов регистра представляется символом шестнадцатеричной системы счисления. Для возможности

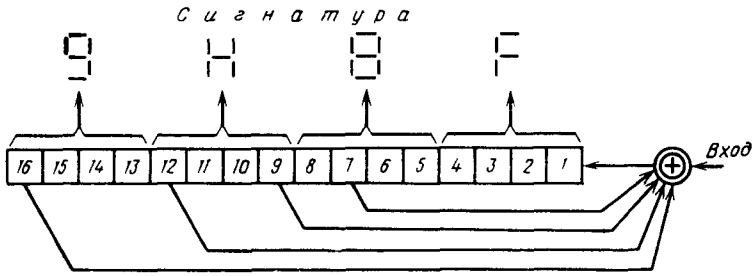


Рис. 7.12. Схема формнрования сигнатуры

использования семисегмеитных индикаторов в качестве таких символов используются 0, 1, 2,..., 9, A, C, F, H, P, U. Использование стандартных для шестнадцатеричной системы счисления символов B, D невозможно, так как на семисегментном индикаторе нельзя отличить символ B от цифры 8 и символ D от цифры 0, а символ E неудобен, так как он имеет изображение, обратное изображению цифры 3.

Число различиых кодовых комбинаций в 16-разрядном регистре составляет $2^{16} = 65536$. Следовательно, число различных сигнатур, которое может быть зафиксировано в сдвиговом регистре, составит 65535. Поэтому вероятность того, что две различные входные последовательности могут привести к одной и той же сигнатуре (и, следовательно, невозможно будет распознать ошибочную последовательность) может быть порядка $1/65536 \approx 0{,}000015$. Таким образом, обеспечивается возможность обнаружения ошибок с вероятностью 0,999985.

Для использования сигнатурного анализа в ПЗУ МПУ должна храниться тестовая программа, при которой определены эталонные сигнатуры для различных точек исправно функционирующего устройства.

7.5. ТЕСТОВОЕ ДИАГНОСТИРОВАНИЕ

При тестовом диагностировании исправность МПУ проверяется с помощью специальных программных средств, называемых *тестовыми программами*.

Принцип проверки работы МПУ с помощью тестовых программ состоит в том, что МПУ выполняет некоторые специальио предиазначенные для диагностирования устройства программы. Причем результаты, получаемые на отдельных этапах в процессе исполнения тестовых программ, заранее известны. Поэтому для выявления неисправности в программе оказывается достаточным предусмотреть выполнение операций, связанных с сопоставлением получаемых результатов с ожидаемыми, и если обнаруживается несовпадение указанных данных, то в программе происходит переход на ветвь выдачи соответствующей информации о наличии неисправности.

Тестовые программы строятся из отдельных подпрограмм, предназначенных для проверки работы отдельных блоков МПУ. Вначале производится проверка микропроцессора. Если подпрограмма проверки этого блока не выявила неисправности, производится переход к следующим подпрограммам, предназначенным, например, для проверки оперативной и постоянной памяти. Затем могут проверяться программируемые интерфейсы ввода-вывода и т. д. Кроме того, если не обнаруживаются неисправности простейшими тестовыми программами малого объема, предназначенными, например, для обнаружения постоянных одиночных ошибок в разрядах узлов (типа «залипания» в состоянии лог. 0 или лог. 1), то можио включить более сложные программы, выявляющие многократные ошибки, случайно возникающие ошибки (сбои) и т. д.

Тестовое диагностирование обычно проводится после пуска МПУ, но оно может проводиться и в процессе нормальной работы. В последнем случае по истечении определенного времени, отмечаемого таймером, либо при возникновении некоторых событий производится прерывание микропроцессора с переходом на выполнение хранящейся в постоянной памяти МПУ тестовой программы. Выполнение тестовой программы может также предусматриваться в режимах, когда микропроцессор не производит какой-либо полезной работы.

Тестовые программы большого объема, предназначенные для глубокой проверки МПУ, целесообразно хранить вне МПУ, используя внешнюю память.

8 МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА В СИСТЕМАХ ПЕРЕДАЧИ И ОБРАБОТКИ СИГНАЛОВ

8.1. АНАЛОГО-ЦИФРОВОЕ ПРЕОБРАЗОВАНИЕ СИГНАЛОВ

Обычно выдаваемый источником сигнал (речевой сигнал в телефонии и радиовещании, телевизионный сигнал и др.) имеет аналоговую форму, описываемую непрерывной во времени функцией с мгновенными значениями, находящимися в некотором интервале. Передача и обработка таких сигналов могут осуществляться также в аналоговой форме. Однако в последнее время все более широкое распространение получают системы передачи и обработки, в которых поступающие на вход аналоговые сигналы преобразуются в цифровую форму, полученные цифровые сигналы передаются или обрабатываются, на выходе системы производится обратное преобразование сигналов из цифровой формы в аналоговую.

Использование цифровой формы представления сигналов может обеспечить более высокую помехоустойчивость при передаче сигналов, стабильность параметров обработки (независимость от времени и влияния изменений в окружающей среде — температуры, влажности и т. д.) при обработке сигналов, возможность построения аппаратуры с использованием последних достижений микроэлектроники, обеспечивающей компактность, экономичность, гибкость функционирования (легкость модификации выполняемых функций) аппаратуры.

Преобразование сигналов из аналоговой формы в цифровую включает в себя следующие операции: *дискретизацию, квантование и кодирование* (рис. 8.1.). В процессе дискретизации из непрерывного сигнала

x (t) берутся отсчеты (мгновенные значения), которые следуют через определенный временной интервал T, называемый *тактовым интервалом*. Согласно теореме Котельникова, если сигнал имеет ограниченный спектр, т. е. все его спектральные составляющие имеют частоты не выше некоторой частоты F_{max}, то для восстановления аналогового сигнала из последовательности его дискретных значений тактовый интервал должен удовлетворять условию $T \leqslant 1/(2F_{max})$. Сущность операции квантования состоит в следующем. Создается сетка так называемых *уровней квантования*, смещенных друг относительно друга на величину, называемую *шагом квантования*, каждому уровню квантования приписывается порядковый номер (0, 1, 2, 3, ...). Полученные в результате дискретизации отсчеты заменяются ближайшими к ним уровнями квантования. Так, на рис. 8.1 отсчет в момент t_0 заменяется ближайшим к нему уровнем квантования с номером 3, а взятый в тактовый момент t_1 отсчет — ближайшим к нему уровнем квантования с номером 6 и т. д. Очевидно, процесс квантования вносит погрешность в представление значений сигнала. Однако выбором достаточно малого шага квантования эту погрешность можно снизить до допустимых значений. Таким образом, последовательность отсчетов сигнала в процессе квантования преобразуется в последовательность соответствующих чисел (номеров уровней квантования). Для представленного на рис. 8.1 сигнала эта последовательность чисел: 3, 6, 7, 4, 1, 2 и т. д. Наконец, в процессе операции кодирования числа этой последовательности представляются в определенной системе счисления, например двоичной.

Преобразование сигналов из аналоговой формы в цифровую осуществляется устройствами, называемыми *аналого-цифровыми преобразователями* (АЦП). Одним из основных параметров АЦП является количество разрядов в выдаваемых данных, характеризующее точность представления отсчетов в цифровой форме; другой его важный параметр — время преобразования (максимальный интервал времени между началом проведения операции по преобразованию одного отсчета и готовностью выходных цифровых данных), определяющее быстродействие устройства преобразования.

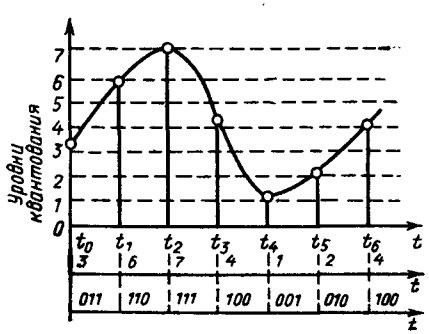


Рис. 8.1. Преобразование сигнала из аналоговой формы в цифровую

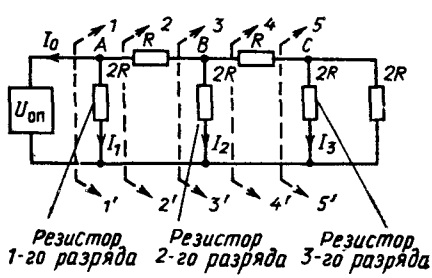


Рис. 8.2. Резисторная матрица R–2R

Обратное преобразование сигналов из цифровой формы в аналоговую выполняется *цифро-аналоговыми преобразователями* информации (ЦАП).

Рассмотрим устройства, реализующие эти преобразования, и обмен данными между этими устройствами и микропроцессором в МПУ.

ЦИФРО-АНАЛОГОВЫЙ ПРЕОБРАЗОВАТЕЛЬ ИНФОРМАЦИИ

Для построения ЦАП могут использоваться различные принципы. Наиболее часто используется принцип формирования токов, пропорциональных весовым коэффициентам разрядов двоичного кода, с последующим их суммированием в разрядах кода, содержащих 1. Формирование указанных токов обычно производится с помощью так называемой резисторной матрицы R — 2R. Схема такого формирователя токов на 3 разряда приведена на рис. 8.2.

Входное сопротивление схемы правее точек 5...5′ равно R, правее точек 4...4′ равно $2R$, правее точек 3...3′ — R, правее точек 2...2′ — $2R$ и, наконец, правее точек 1...1′ — R. Таким образом, источник опорного напряжения выдает в матрицу ток $I_0 = U_{оп}/R$. Так как сопротивление между точками 2...2′ равно $2R$, то в точке A этот ток I_0 разветвляется в две ветви с равными сопротивлениями $2R$ и, следовательно, по разрядному резистору 1-го разряда потечет ток $I_1 = I_0/2 = I_0 \cdot 2^{-1}$ и ток $I_0/2$ потечет по резистору R от точки A к точке B. Так как сопротивление между точками 4...4′ также равно $2R$, то в силу тех же причин из притекшего к точке B тока $I_0/2$ половина его, равная $I_2 = I_0/4 = I_0 \cdot 2^{-2}$, потечет через разрядное сопротивление 2-го разряда и ток $I_0/4$ пройдет через резистор R от точки B к точке C. В точке C этот ток вновь поделится пополам и в разрядном резисторе 3-го разряда возникнет ток $I_3 = I_0/8 = I_0 \cdot 2^{-3}$. Очевидно, такую матрицу можно было бы расширить, предусмотрев в ней любое число разрядных резисторов. При этом ток в i-м разрядном резисторе будет равен $I_i = I_0 \cdot 2^{-i}$. Коэффициент 2^{-i} равен весовому коэффициенту i-го разряда дробного двоичного числа (если отсчет разрядов вести от старшего разряда в сторону младших разрядов).

Обратимся к полной схеме ЦАП, приведенной на рис. 8.3. Эта схема соответствует структурной схеме выпускаемой промышленностью микросхемы ЦАП К572ПА1. На цифровые входы в параллельной форме подается подлежащее преобразованию в аналоговую форму двоичное число $a_1a_2a_3\ldots$ Входное напряжение каждого разряда поступает в отдельный усилитель-инвертор, имеющий два выхода: прямой и инверсный. На прямом выходе создается напряжение логического уровня, совпадающего со входным, на инверсном — напряжение уровня, инверсного по отношению ко входному. Выходные напряжения усилителя управляют состоянием токовых ключей. Если на цифровом входе i-го разряда $a_i = 1$, то усилитель откроет ключ, через который разрядный ток данного разряда резистивной матрицы поступит на выход $I_{\text{}}$

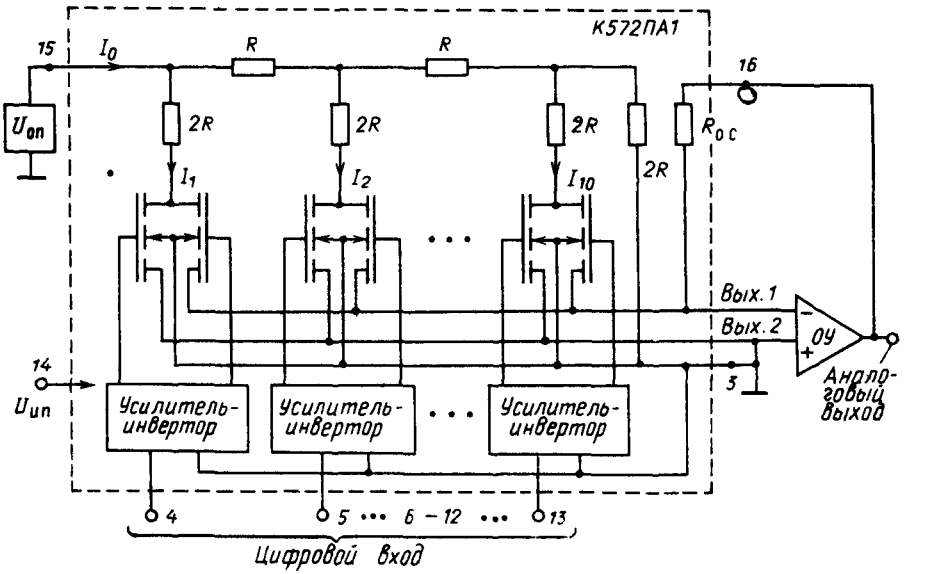


Рис. 8.3. Схема цифро-аналогового преобразователя

Если $a_i = 0$, то усилитель откроет другой ключ, через который этот ток поступит на выход *2*.

Таким образом, на выходе *1* образуется суммарный ток $I_{вых1} = \sum_{i=1}^{n} a_i I_i = I_0 \sum_{i=1}^{n} a_i 2^{-i}$.

На выходе *2* этот ток равен

$$I_{вых2} = \sum_{i=1}^{n} \bar{a}_i I_i = I_0 \sum_{i=1}^{n} \bar{a}_i 2^{-i}.$$

Если к выходу *1* подключить операционный усилитель, как показано на схеме, на аналоговом выходе ЦАП образуется напряжение

$$U_{вых} = I_{вых1} R_{ос} = I_0 R_{ос} \sum_{i=1}^{n} a_i 2^{-i} = U_{оп} \frac{R_{ос}}{R} \sum_{i=1}^{n} a_i 2^{-i}.$$

Так как $\sum_{i=i}^{n} a_i 2^{-i} = N$ — десятичное представление входного двоичного числа, то $U_{вых} = U_{оп} \frac{R_{ос}}{R} N$, т. е. на аналоговом выходе образуется напряжение, пропорциональное значению входного числа.

Микросхема К572ПА1 может обеспечить преобразование 10-разрядного двоичного числа.

АНАЛОГО-ЦИФРОВОЙ ПРЕОБРАЗОВАТЕЛЬ ИНФОРМАЦИИ

Рассмотрим АЦП, построенный на принципе последовательных поразрядных приближений. На рис. 8.4 приведена функциональная схема микросхемы АЦП К1113ПВ1, использующего данный принцип преобразования. В преобразователе предусмотрен 10-разрядный регистр, в котором последовательно разряд за разрядом (начиная со старшего разряда) формируется двоичное число, соответствующее цифровой форме представления поданного на вход (вывод *13*) микросхемы аналогового напряжения. Процесс формирования этого числа состоит в следующем.

Регистр сбрасывается в нуль и затем записывается единица в триггер старшего (десятого) разряда регистра. Получающееся в регистре число с помощью ЦАП преобразуется в пропорциональный ему ток $I_{ос}$, который с помощью компаратора сравнивается с током $I_{вх} = U_{вх}/R$, возникающим под воздействием входного напряжения $U_{вх}$. Если выполняется неравенство $I_{вх} \geqslant I_{ос}$, то число, в которое преобразуется $U_{вх}$, действительно содержит единицу в старшем разряде. При невыполнении неравенства триггер старшего разряда возвращается в состояние лог. 0.

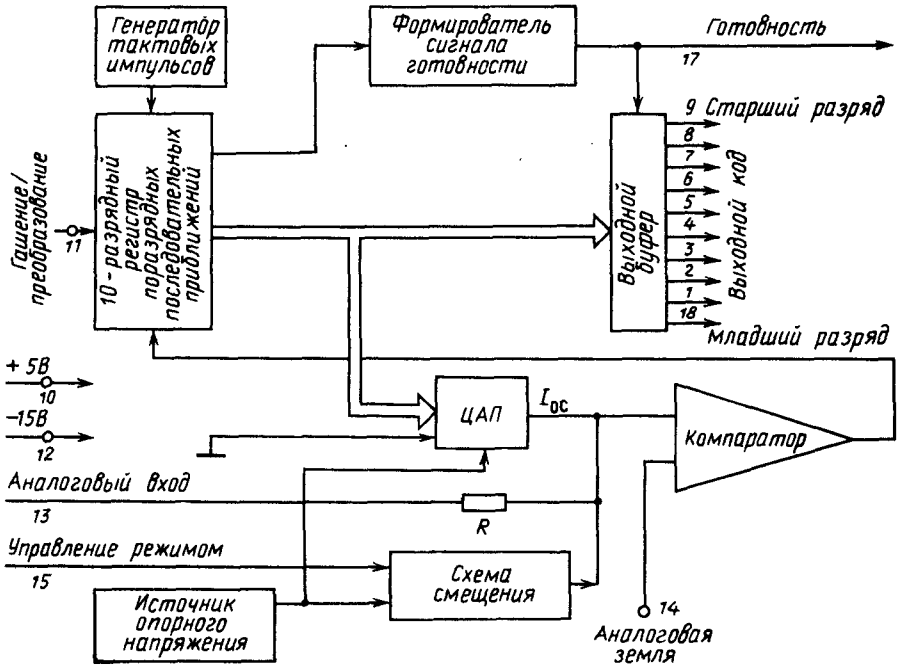


Рис. 8.4. Схема аналого-цифрового преобразователя

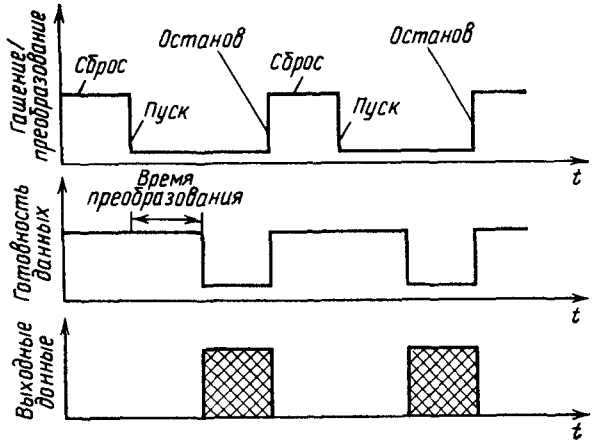


Рис. 8.5. Временные диаграммы работы АЦП

Далее производится запись единицы в триггер следующего 9-го разряда и вновь сравнивается $I_{вх}$ с $I_{ос}$, соответствующим имеющемуся к этому моменту времени числу в регистре; выясняется, должна ли быть сохранена единица в данном разряде или триггер этого разряда должен быть возвращен в состояние лог. 0.

Таким образом, производится опробование во всех десяти разрядах, после чего получающееся в регистре число выдается на выход через буфер, построенный на элементах с тремя состояниями.

На рис. 8.5 приведены временные диаграммы работы АЦП. Запуск преобразователя производится подачей сигнала на вход *Гашение/ преобразование* (вывод *11*). При высоком уровне напряжения на этом входе регистр удерживается в нулевом состоянии. Затем при переходе сигнала на низкий уровень (на отрицательном фронте сигнала) происходит запуск преобразователя и выполняются описанные выше действия. После окончания преобразования АЦП выдает на вывод *17* сигнал (напряжение низкого уровня) *Готовность данных*, выходные буферы вы-

Таблица 8.1

Тип микросхемы	Тип преобразователя	Число разрядов	Время преобразования, мкс	Число выводов	Тип микросхемы	Тип преобразователя	Число разрядов	Время преобразования, мкс	Число выводов
К572ПА1	ЦАП	10	5	16	К1107ПВ1	АЦП	6	0,1	42
К572ПА2	ЦАП	12	15	48	К1108ПА1	ЦАП	12	0,4	24
К572ПВ1	АЦП	12	110	48	К1113ПВ1	АЦП	10	30	18
К594ПА1	ЦАП	12	3,5	24					

водятся из 3-го состояния (из отключенного состояния с высоким выходным сопротивлением) и полученное в регистре число через буферы выдается на цифровые выходы.

В табл. 8.1 приведены некоторые данные выпускаемых промышленностью микросхем ЦАП и АЦП.

8.2. АНАЛОГОВЫЕ И ЦИФРОВЫЕ СИГНАЛЫ

Пусть $x(t)$ — аналоговый сигнал, а соответствующий ему цифровой сигнал (получеиный в результате аналого-цифрового преобразования сигнала $x(t)$) есть $x(nT)$, где T — тактовый период, n — номер отсчета аналогового сигнала при его преобразовании в цифровую форму, nT — тактовые моменты (моменты отсчета аналогового сигнала). При этом будем полагать, что аналоговый сигнал имеет ограниченный спектр и тактовый период удовлетворяет условию $T \leqslant 1/(2F_{\max})$.

В дальнейшем будем считать, что при $t < 0$ $x(t) = 0$ и, следовательно, отличные от нуля значения $x(nT)$ могут иметь место лишь при $nT \geqslant 0$ и n представляет собой последовательность 0, 1, 2, ... чисел натурального ряда.

Известно, что операция получения спектральной функции $X(j\omega)$ аналогового сигнала $x(t)$ и обратная операция получения сигнала $x(t)$ по известной его спектральной функции $X(j\omega)$ производится с помощью пары преобразований Фурье

$$X(j\omega) = \int_0^\infty x(t)\, e^{-j\omega t}\, dt, \tag{8.1}$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega)\, e^{j\omega t}\, d\omega. \tag{8.2}$$

Для цифрового сигнала спектральная функция последовательности $x(nT)$ обозначается $X(e^{j\omega T})$, а преобразования Фурье определяются следующими выражениями:

$$X(e^{j\omega T}) = \sum_{n=0}^{\infty} x(nT)\, e^{-jn\omega T}, \tag{8.3}$$

$$x(nT) = \frac{T}{2\pi} \int_{-\pi/T}^{\pi/T} X(e^{j\omega T})\, e^{jn\omega T}\, d\omega. \tag{8.4}$$

Преобразование Фурье, независимо от того, проводится ли оно над аналоговым или дискретным сигналом, и независимо от того, является оно прямым или обратным, характеризуется следующим свойством: преобразование Фурье, выполняемое над периодической функцией,

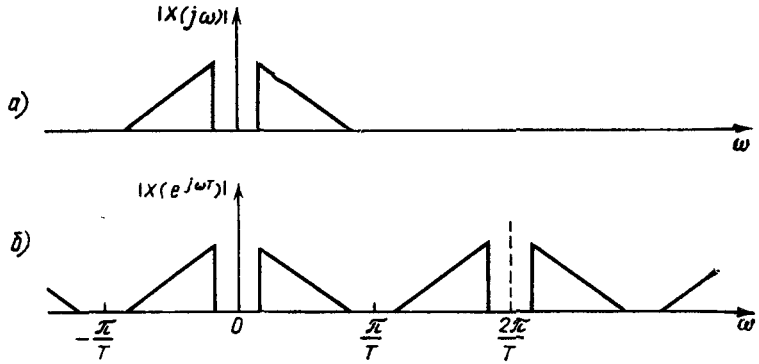


Рис. 8.6. Спектральные функции:
а) непрерывного сигнала; *б)* дискретного сигнала

приводит к дискретной функции, и, наоборот, преобразование Фурье дискретной функции является периодической функцией. Из этого следует, что в случае, если аналоговая функция $x(t)$ является дискретной, то ее спектральная функция является периодической. Если спектр $X(j\omega)$ аналогового сигнала $x(t)$ представляется функцией, изображенной на рис. 8.6, *а*, то после преобразования в цифровую форму сигнал будет описываться дискретной функцией $x(nT)$ и его спектральная функция будет периодической, как показано на рис. 8.6, *б*. Как видно из рис. 8.6, в пределах интервала $-\pi/T \leqslant \omega \leqslant \pi/T$ модуль спектральной функции аналогового и цифрового сигналов подобны. При ограниченном спектре аналогового сигнала спектр цифрового сигнала оказывается неограниченным и имеет периодическую структуру с периодом $2\pi/T$. Отсюда следует прием, используемый для получения аналогового сигнала $x(t)$ из цифрового сигнала $x(nT)$: достаточно цифровую последовательность преобразовать в последовательность импульсов, имеющих малую длительность, и амплитуды, равные $x(nT)$, а затем из спектра такого дискретного сигнала с помощью фильтра нижних частот выделить ту ее часть в интервале $0 \leqslant \omega \leqslant \pi/T$, которая совпадает со спектром аналогового сигнала. При этом на выходе фильтра образуется аналоговый сигнал $x(t)$, соответствующий цифровому сигналу $x(nT)$.

8.3. ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

Вычисление выражения (8.3) не может быть реализовано, так как в нем предусматривается суммирование бесконечного числа членов. При решении практических задач используется конечное число N отсчетов аналогового сигнала и, следовательно, в выражении (8.3) может производиться суммирование конечного числа членов. В этом

случае пара преобразований Фурье принимает вид так называемого *дискретного преобразования Фурье* (ДПФ):

$$X(k)=\frac{1}{N}\sum_{n=0}^{N-1}x(nT)\,e^{-j2\pi nk/N}=\frac{1}{N}\sum_{n=0}^{N-1}x(nT)\,W^{nk},$$

$$k=0,\ 1,\ldots\ N-1; \tag{8.5}$$

$$x(nT)=\sum_{k=0}^{N-1}X(k)\,e^{j2\pi nk/N}=\sum_{k=0}^{N-1}X(k)\,W^{-nk},\ n=0,\ 1,\ldots,\ N-1. \tag{8.6}$$

Здесь $W^{nk}=e^{-j2\pi nk/N}$. Выражение (8.5) определяет прямое дискретное преобразование Фурье (ДПФ), выражение (8.6) — обратное дискретное преобразование Фурье (ОДПФ).

Рассмотрим различие в результатах, получаемых при пользовании выражениями (8.3), (8.4) и (8.5), (8.6).

В случае ДПФ предполагается, что исходная функция $x(nT)$ является периодической с периодом NT, в силу чего для суммирования в (8.5) из последовательности $x(nT)$ выбираются значения в пределах одного периода (рис. 8.7, *а*). Тогда в силу сформулированных в § 8.2 свойств преобразования Фурье дискретность функции $x(nT)$ приводит к периодичности спектральной функции $X(k)$, а так как функция $x(nT)$ принимается периодической, то спектральная функция $X(k)$ оказывается дискретной. Следовательно, в случае ДПФ результат

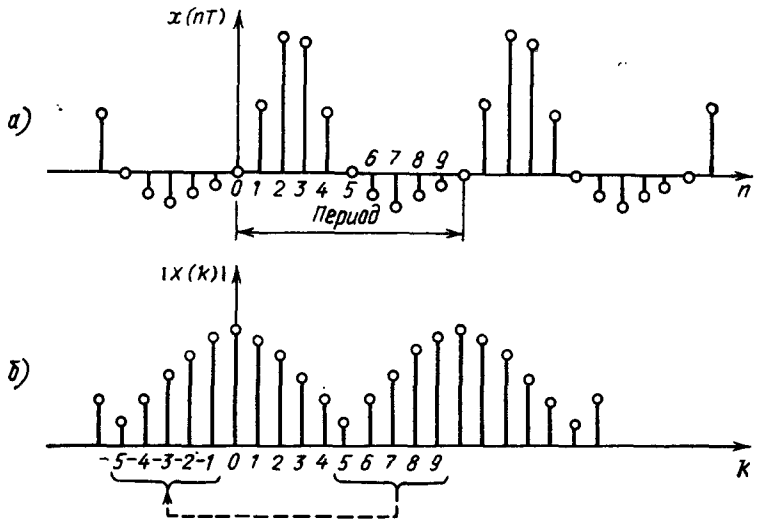


Рис. 8.7. Дискретный периодический сигнал (*а*) и его спектр (*б*)

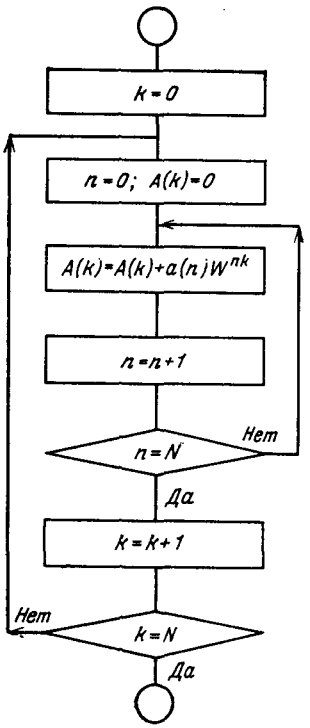


Рис. 8.8. Схема алгоритма ДПФ и ОДПФ

преобразования всегда будет периодической и дискретной функцией, и, таким образом, первые $N/2-1$ точек $X(k)$ определяют спектральные составляющие на положительных частотах, следующие $N/2-1$ точек соответствуют спектральным составляющим при отрицательных частотах (рис. 8.7, *б*).

Перейдем к рассмотрению алгоритма ДПФ и ОДПФ. Оба преобразования можно вычислить с помощью одиого и того же алгоритма. Для этого используется выражение

$$A(k) = \sum_{n=0}^{N-1} a(n)\, W^{nk},$$

$$k = 0, 1, \ldots, N-1. \qquad (8.7)$$

Для прямого ДПФ $a(n) = x(nT)$, а искомое $X(k) = \frac{1}{N} A(k)$; при вычислении обратного ДПФ $a(n) = X(N-n)$, $n = 1, 2, \ldots, N-1$; $a(0) = X(0)$, а искомое $x(kT) = A(k)$.

Таким образом, для вычисления ДПФ и ОДПФ можно пользоваться единым алгоритмом, предусматривающим расчет по (8.7). Схема такого алгоритма приведена на рис. 8.8.

8.4. БЫСТРОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

Представленный на рис. 8.8 алгоритм дискретного преобразования Фурье предусматривает большой объем вычислений. Для вычисления одного значения $A(k)$ требуется N-кратное повторение внутреннего цикла и, таким образом, выполнение N умиожений (в общем случае выполняемых над парами комплексных чисел). А для нахождения N значений $A(k)$ потребуется N-кратное повторение внешнего цикла и количество умножений будет равно N^2. Этот алгоритм содержит большое количество избыточных операций, в том числе и таких, когда одни и те же операции многократно выполняются над одними и теми же зиачениями величин. Это связано со следующим. Весовая функция $W^{nk} = e^{-j2\pi nk/N}$ является периодической функцией аргумента nk. Так как n и k принимают значения из последовательности 0,1, ..., $N = 1$, то произведение nk, принимающее значения $0,1,\ldots,(N-1)^2$, будет содержать большое число периодов N и соответствующие им значения весовой функции W^{nk} будут повторяться через период N. В табл. 8.2 это показано для $N = 8$.

В пределах одного периода первые $N/2$ значений W^{nk} отличаются от вторых $N/2$ значений лишь знаком. Действительно,

$$W^4 = e^{-j2\pi \cdot 4/8} = e^{-j\pi} = -1 = -W^0,$$
$$W^5 = e^{-j2\pi \cdot 5/8} = e^{-j\pi} \cdot e^{-j2\pi \cdot 1/8} = -W^1 \text{ и т. д.} \tag{8.8}$$

Устранение избыточных операций умножения приводит к так называемому алгоритму *быстрого преобразования Фурье* (БПФ).

Рассмотрим один из способов построения алгоритма БПФ, называемый способ *прореживания по времени*.

Считая, что N делится на 2, представим (8.7) двумя суммами, соответствующими четным и нечетным значениям n:

$$A(k) = \sum_{n=0}^{N-1} a(n)\, e^{-j2\pi kn/N} = \sum_{n=0}^{N/2-1} a(2n)\, e^{-j2\pi 2nk/N} +$$
$$+ \sum_{n=0}^{N/2-1} a(2n+1)\, e^{-j2\pi(2n+1)k/N} = \sum_{n=0}^{N/2} a(2n)\, e^{-j2\pi 2nk/N} +$$
$$+ e^{-j2\pi k/N} \sum_{n=0}^{N/2-1} a(2n+1)\, e^{-j2\pi 2nk/N} = \sum_{n=0}^{N/2-1} a(2n)\, W^{2nk} +$$
$$+ W^k \sum_{n=0}^{N/2-1} a(2n+1)\, W^{2nk} = B(k) + W^k C(k), \tag{8.9}$$

где $B(k)$ и $C(k)$ — суммы соответственно первого и второго слагаемых.

Таким образом, вычисление N-точечного преобразования $A(k)$, $k = 0,1, \ldots, N-1$, можно произвести путем вычисления двух $N/2$-точечных преобразований: $B(k)$, $k = 0,2, \ldots, N-2$ и $C(k)$, $k = 1,3, \ldots, N-1$, с последующим их объединением по формуле (8.9). На рис. 8.9, *а* показан этот прием для $N = 8$. Такая же схема, но использующая соотношения (8.8) для сокращения числа умножений при объединении двух 4-точечных преобразователей, показана на рис. 8.9, *б*. Прямое вычисление N-точечного преобразования требует N^2 комплексных умножений, при рассмотренном приеме прямое вычисление двух $N/2$-точечных преобразований потребует $2(N/2)^2 = N^2/2$ комплексных умножений, а их объединение — еще $N/2$ умножений. Таким образом, количество умножений станет равным $(N + N^2)/2$, что при больших N примерно вдвое сокращает требуемое количество умножений.

Таблица 8.2

nk	0	1	2	3	4	5	6	7	8	9	10
W^{nk}	$W^0 = 1$	W^1	W^2	W^3	$W^4 = -W^0$	$W^5 = -W^1$	$W^6 = -W^2$	$W^7 = -W^3$	$W^8 = W^0$	$W^9 = W^1$	$W^{10} = W^2$

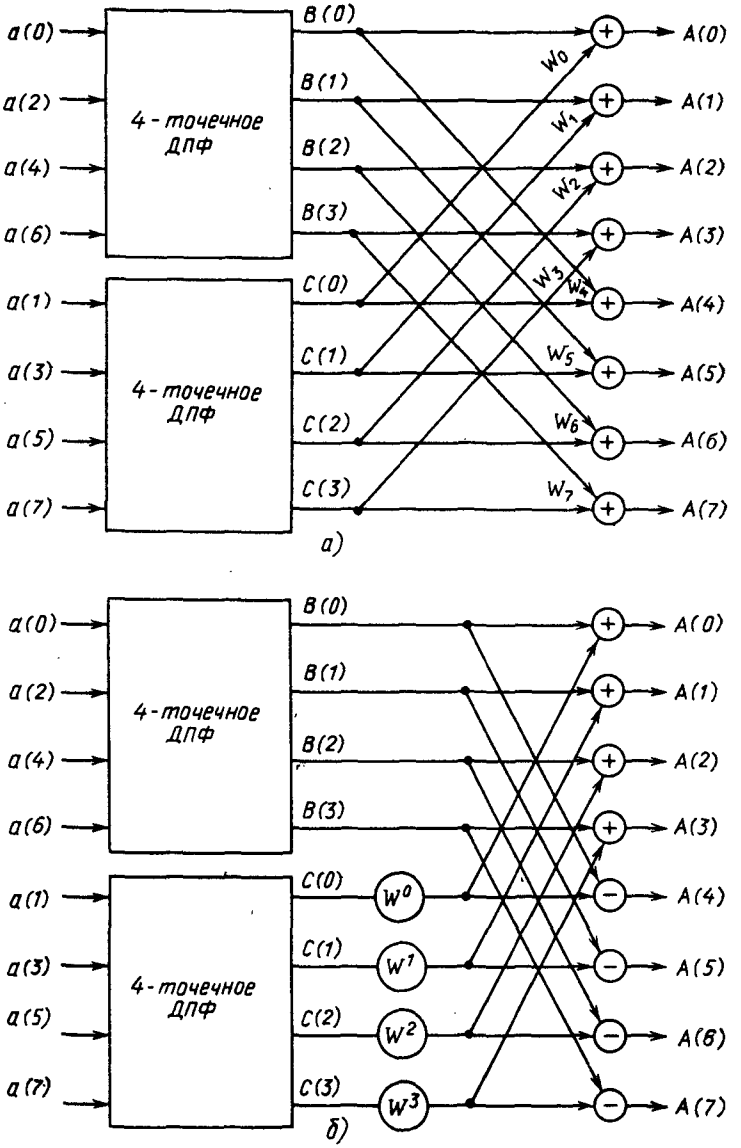


Рис. 8.9. Восьмиточечное ДПФ, представленное через два четырехточечных ДПФ *(а)* и сокращенне числа умножений при объединенни двух четырехточечных ДПФ *(б)*

Если $N/2$ в свою очередь делится на 2, то вычисление каждого из преобразований $B(k)$ и $C(k)$ можно также свести к двум $N/4$-точечным преобразованиям, что вызовет дополнительное уменьшение требуемого количества операций умножения и т. д. На рис. 8.10 показано сведение 8-точечного преобразования к четырем 2-точечным, на рис. 8.11 — полная схема вычислений, в которой дополнительно показаны вычисления, требуемые для 2-точечных преобразований.

Если N представляется целой степенью двух ($N = 2^m$), то вычисления разбиваются на $m = \log_2 N$ этапов, в каждом из которых требуется $N/2$ умножений. Таким образом, общее количество умножений равно $\frac{N}{2}\log_2 N$. Например, при $N = 2^{10} = 1024$-точечном преобразовании число умножений окажется равным $0{,}5 \cdot 1024 \cdot 10 \approx 10^4/2$, в то время как при $N =$ точечном ДПФ потребовалось бы $N^2 \approx 10^6$ операций умножения. Как видим, БПФ обеспечивает существенное сокращение (в данном примере в 200 раз) объема вычислений.

Следует обратить внимание на то, что исходные отсчеты подаются на входы преобразователя не в естественном порядке (на рис. 8.11 эта последовательность: $a(0)$, $a(4)$, $a(2)$,...). Для получения номеров отсчетов в требуемой последовательности необходимо номера отсчетов, следующих в естественном порядке, представить в двоичной системе счисления, а затем в каждом из этих двоичных представлений переставить разряды, записав их в обратном (так называемом *двоично-инверсном*) порядке:

естественный порядок	000 001 010 011 100 101 110 111
двоично-инверсный порядок	000 100 010 110 001 101 011 111
	(0) (4) (2) (6) (1) (5) (3) (7).

Убеждаемся, что полученная последовательность входных отсчетов соответствует приведенной на рис. 8.11.

Рассмотрим способы выполнения перестановок элементов массива в соответствии с двоично-инверсным порядком следования. Пусть имеются два m-разрядных регистра R_1 и R_2 (рис. 8.12,*а*). Поместим в регистр R_1 порядковый номер входных данных, представленный m-разрядным двоичным числом. Затем проведем серию сдвигов содержимого регистров R_1 и R_2, причем содержимое регистра R_1 будем сдвигать вправо, а содержимое регистра R_2 — влево, выдвигаемое при сдвиге содержимое младшего разряда регистра R_1 будем передавать в младший разряд регистра R_2. После m-кратного повторения таких сдвигов в R_2 образуется двоично-инверсный номер. На рис. 8.12, *б* показана схема алгоритма выполнения перестаиовок для получения двоично-инверсного порядка следования элементов массива. Другой более быстрый алгоритм таких перестановок представлен на рис. 8.13. Предлагаем самостоятельно, задавшись значением N числа элементов в массиве, проследить выполняемые в последнем алгоритме операции и убедиться в том, что оии действительно приводят к образованию двоично-инверсной последовательности элементов.

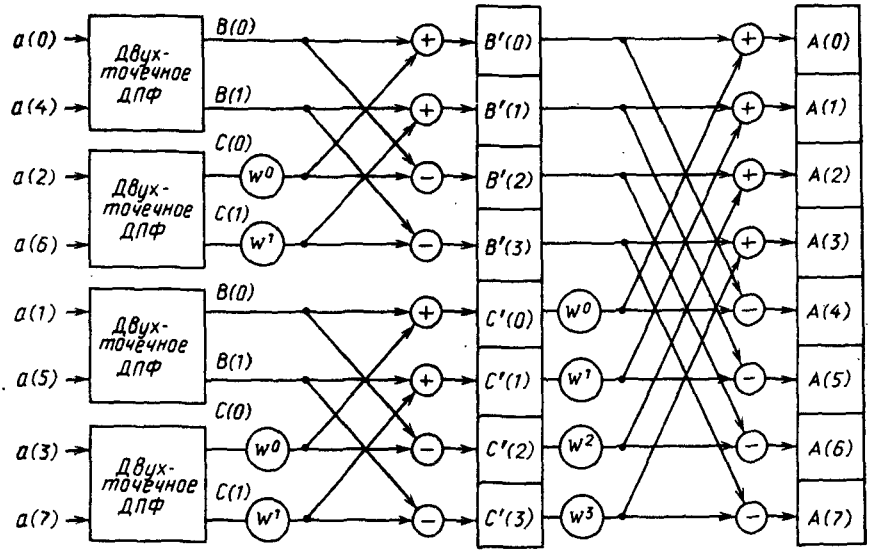


Рис. 8.10. Восьмиточечное ДПФ, представленное через четыре двухточечных ДПФ

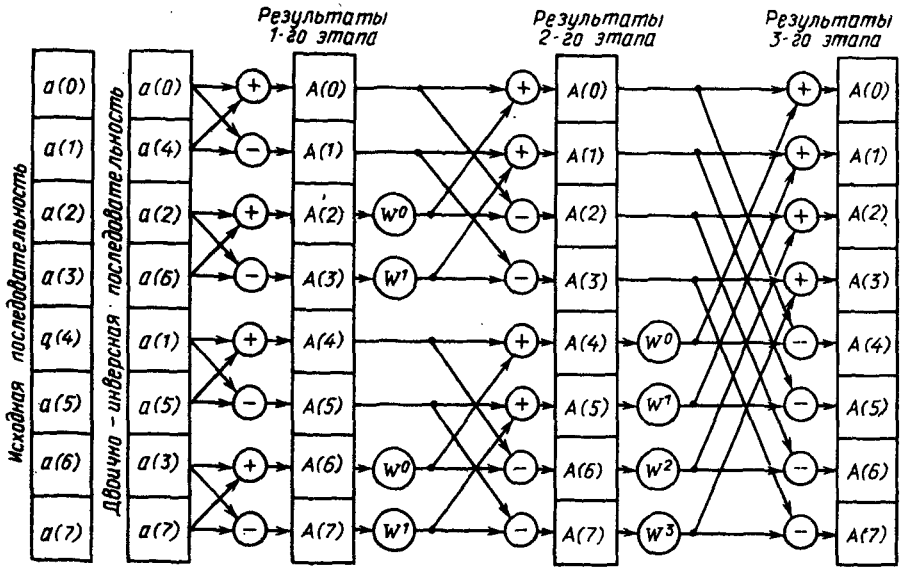


Рис. 8.11. Полная схема восьмиточечного БПФ

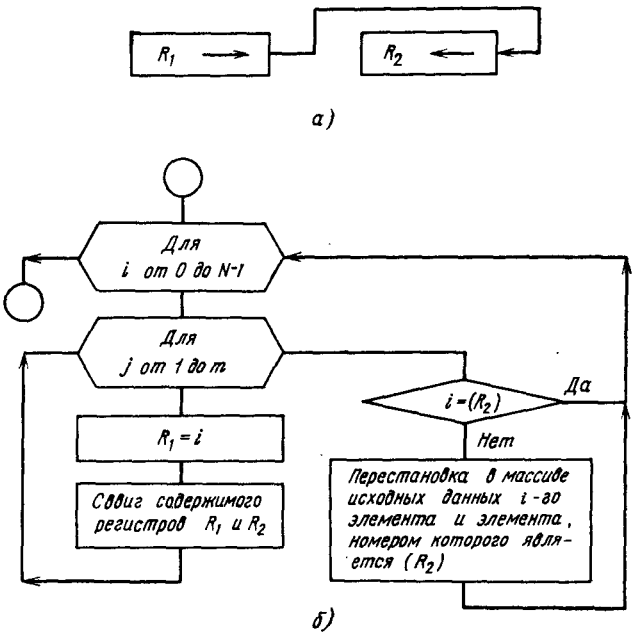


Рис. 8.12. Перестановки элемеитов массива в соответствии с двоично-инверсным порядком следования:

а) схема формирования двоично-инверсиых номеров; *б)* схема алгоритма получения двоично-инверсного порядка следования

Обратимся к рис. 8.11 и рассмотрим, какая емкость требуется для хранения исходных данных, промежуточных и конечных результатов БПФ.

Как мы видели выше, двоично-инверсная последовательность получается путем перестановок пар элементов, производимых в исходной последовательности. Такие перестановки можно осуществлять непосредственно над содержимым ячеек, хранящих данные исходной последовательности. Таким образом, получение двоично-инверсной последовательности не требует использования дополнительного массива ячеек памяти, она формируется в том же массиве ячеек, в котором хранилась исходная последовательность.

В процессе выполнения БПФ (рис. 8.11) на любом его этапе результат вычисления соответствующих пар значений $A'(p)$ и $A'(q)$ получается путем использования только значений $A(p)$ и $A(q)$, которые берутся из результата предыдущего этапа преобразований (рис. 8.14, *а*). Поэтому для хранения вычисленных значений $A'(p)$ и $A'(q)$ не обязательно использование новых ячеек памяти, эти значения можно помещать в ячейки, хранившие результаты предыдущего этапа $A(p)$ и $A(q)$. Соответственно, результаты, получаемые в ходе 1-го этапа преобразования, могут помещаться на место элементов двоично-инверсной

последовательности. Этот принцип использования памяти отражен на рис. 8.11: результаты каждого этапа преобразований имеют обозначения, совпадающие с обозначением результатов предыдущего этапа. Таким образом, требуемая в БПФ емкость памяти для хранения исходных данных, промежуточных и конечных результатов преобразований равна N (числу элементов в исходном массиве данных).

На рис. 8.14, *б* показана схема алгоритма БПФ, где $m = \log_2 N$ — число этапов преобразования. Выпускаемые в настоящее время микропроцессорные средства позволяют выполнять БПФ в реальном масштабе времени над сигналами с полосой частот до сотен килогерц. При обработке сигналов изображений полоса частот, занимаемая сигналом, может оказаться шире. Необходимое при этом повышение скорости обработки может быть достигнуто использованием аппаратной реализации операции умножения и применением мультипроцессорных устройств. Рассмотрим принцип реализации БПФ.

Пусть в МПУ, реализующем БПФ, с временным интервалом T вводятся массивы данных, содержащие $2^8 = 256$ данных. В том же темпе на выходе необходимо получать массивы, представляющие собой результат преобразования входных массивов. Если обработка будет вестись одним микропроцессором, к моменту поступления на вход очередного массива результат обработки предыдущего массива должен

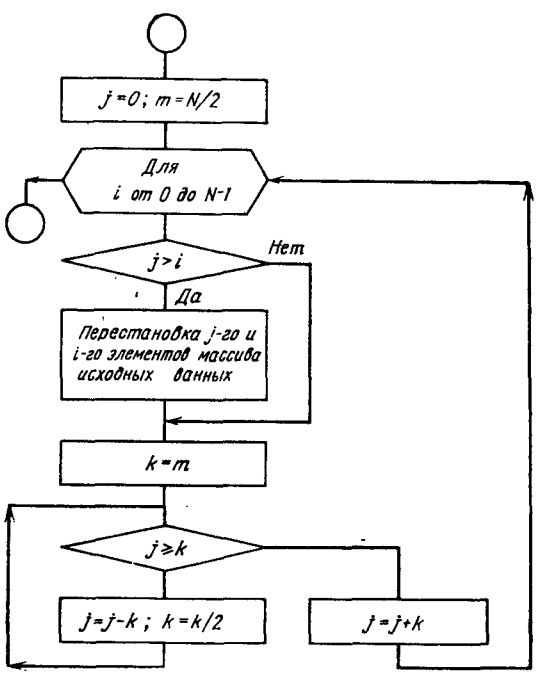


Рис. 8.13. Быстрый алгоритм двоично-инверсных перестановок

быть выдан из микропроцессорного устройства. Таким образом, время преобразования не должно превышать T. Для реализации БПФ можно использовать несколько микропроцессоров, например четыре. При этом 1-й микропроцессор может выполнять не все преобразование, а лишь часть его (1-й и 2-й этапы). Передавая результат такой частичной обработки 2-му микропроцессору, 1-й микропроцессор приступает к выполнению 1-го и 2-го этапов преобразования над очередным массивом данных; 2-й микропроцессор, получая от 1-го микропроцессора результат 1-го и 2-го этапов преобразования, выполняет 3-й и 4-й этапы преобразования, передавая результат следующему микропроцессору и т. д. Последний 4-й микропроцессор выполняет 7-й и 8-й этапы преобразования и полученный окончательный результат выдает на выход. В такой системе обработки на все преобразование может затрачиваться время $4T$ и за время T один микропроцессор должен выполнять не все преобразование, но лишь часть его этапов. При такой обработке для хранения исходных данных и данных, представляющих собой результаты выполнения отдельных этапов, в памяти можно выделить отдельные массивы ячеек (рис. 8.15). При этом 1-й микропроцессор, выбирая данные из массива A, будет помещать результат 1-го этапа преобразования в массив B; одновременно 2-й микропроцессор, используя данные массива C, будет формировать результат обработки в массиве D и т. д. Затем 1-й микропроцессор выбирает данные из массива B и результат 2-го этапа преобразования помещает в

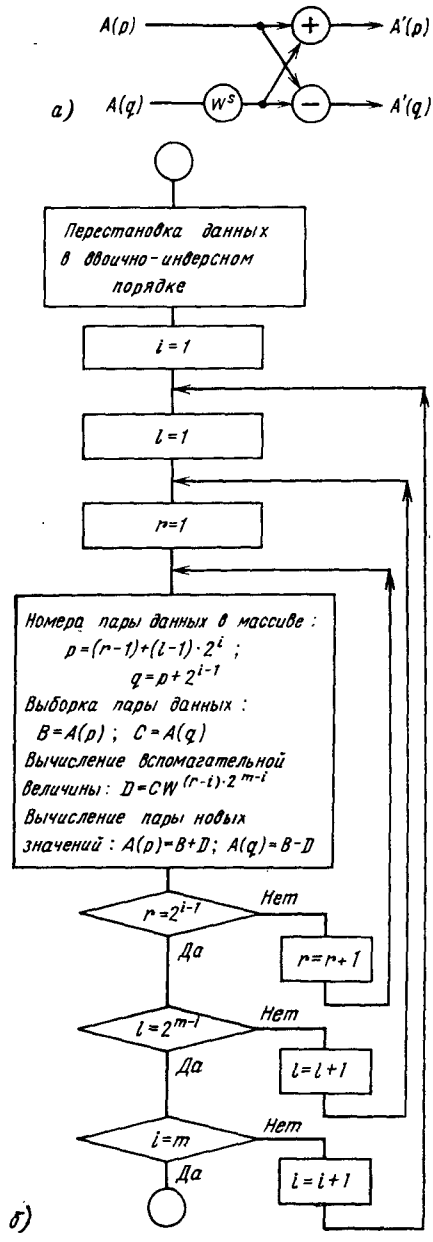


Рис. 8.14. Схема вычисления пар значений при выполнении БПФ *(а)* и схема алгоритма БПФ *(б)*

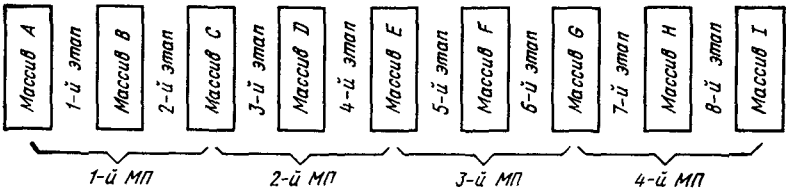


Рис. 8.15. Повышение скорости выполнения БПФ путем использования нескольких микропроцессоров

массив C; 2-й микропроцессор в это время выбирает данные из D и результат 4-го этапа преобразования помещает в массив E и т. д. Использование подобной мультимикропроцессорной системы обеспечивает многократное уменьшение интервала времени, через который на вход микропроцессорного устройства подаются массивы подлежащих преобразованию данных.

Так как в приведенном описании предполагалось, что все микропроцессоры используют общую память, у которой имеется лишь один вход и один выход, то в каждый тактовый период к такой памяти может обращаться лишь один микропроцессор системы. Этот режим функционирования системы нетрудно обеспечить, соответствующим образом сдвинув начала выполнения этапов в отдельных микропроцессорах.

8.5. ПЕРЕНОС СПЕКТРА СИГНАЛОВ ИЗ ОДНОЙ ЧАСТОТНОЙ ОБЛАСТИ В ДРУГУЮ

Пусть исходный сигнал имеет действительные значения $x\,(nT)$ и спектр $X\,(k)$, модуль которого $|X\,(k)|$ показан на рис. 8.16, *а*. Необходимо преобразовать сигнал таким образом, чтобы его спектр оказался

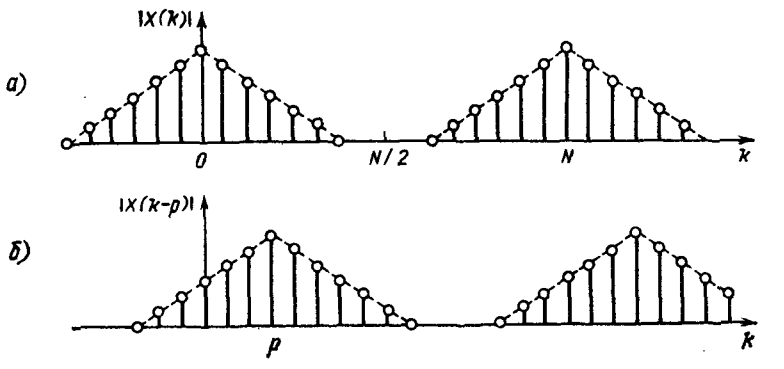
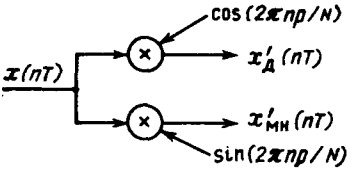


Рис. 8.16. Перенос спектра сигнала из одной частотной области в другую: *а)* исходный спектр; *б)* сдвинутый спектр

Рис. 8.17. Операции при сдвиге спектра



сдвинутым вправо на p дискретных значений частоты (на частоту $\Delta\omega = 2\pi p/N$), как показано на рис. 8.16, *б*. Очевидно, спектральная функция преобразованного сигнала будет описываться выражением $X(k-p)$. Так как

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(nT)\cdot W^{nk},$$

то

$$X(k-p) = \frac{1}{N}\sum_{n=0}^{N-1} x(nT)\cdot W^{n(k-p)} = \frac{1}{N}\sum_{n=0}^{N-1} x(nT)\cdot W^{-np}\,W^{nk} =$$

$$= \frac{1}{N}\sum_{n=0}^{N-1} x'(nT)\cdot W^{nk}, \qquad (8.10)$$

где $x'(nT) = x(nT)\cdot W^{-np} = x(nT)\cdot e^{j2\pi pn/N}$ —

преобразованный сигнал, имеющий требуемый спектр.

Таким образом, для сдвига спектра вправо необходимо дискретные значения сигнала $x(nT)$ умножить на $e^{j2\pi pn/N} = \cos(2\pi pn/N) + j\sin(2\pi pn/N)$, как показано на рис. 8.17, где $x'_д(nT)$ — действительная, $x'_{мн}(nT)$ — мнимая части значений преобразованного сигнала.

Рассмотрим использование сдвига спектра для выделения одной боковой полосы частот (рис. 8.18, *г*). Эта операция может быть выпол-

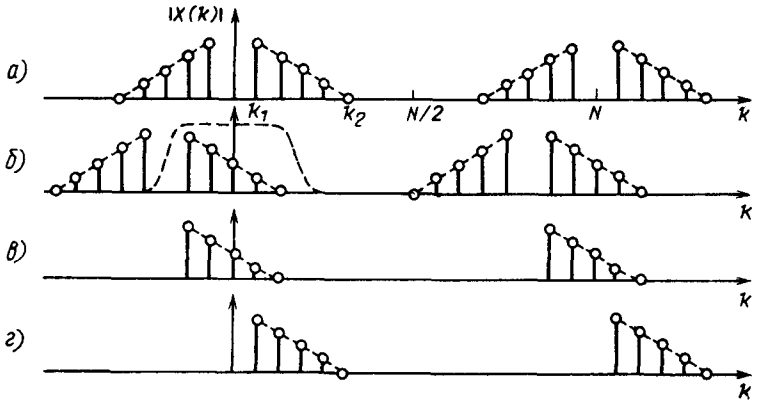


Рис. 8.18. Использование сдвига спектра для выделения одной боковой полосы частот

нена следующим образом. Спектр исходного сигнала сдвигается влево на $(k_1 + k_2)/2$ дискретных значений частоты (рис. 8.18, *б*), затем с помощью цифрового фильтра (цифровые фильтры рассматриваются в § 8.7) выделяется требуемая часть спектра (рис. 8.18, *в*), содержащая верхнюю боковую полосу частот, после чего производится сдвиг спектра вправо на $(k_1 + k_2)/2$ дискретных значений частоты (рис. 8.18, *г*).

8.6. ВЫЧИСЛЕНИЕ ЭНЕРГЕТИЧЕСКОГО СПЕКТРА, КОРРЕЛЯЦИОННОЙ ФУНКЦИИ, СВЕРТКИ

Как было показано выше, ДПФ позволяет рассчитать спектр $X(k)$ периодической последовательности $x(nT)$. Наряду с $X(k)$ часто представляет интерес *энергетический спектр*, составляющие которого представляют собой мощность соответствующих составляющих спектра $X(k)$.

Мощность k-й составляющей спектра $X(k)$ равна

$$P(k) = X(k)\,X^*(k) = |X(k)|^2, \quad k = 0, 1, \ldots, N-1, \tag{8.11}$$

где $X^*(k)$ — комплексное значение, сопряженное с $X(k)$ [$X^*(k)$ отличается от $X(k)$ лишь знаком мнимой части].

Корреляционная функция последовательности $x(nT)$ определяется выражением

$$R(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(nT) \cdot x(kT + nT), \quad k = 0, 1, \ldots, N-1. \tag{8.12}$$

Корреляционная функция $R(k)$ и энергетический спектр $P(k)$ связаны парой преобразования Фурье:

$$R(k) = \sum_{n=0}^{N-1} P(n)\, W^{-hk}, \quad k = 0, 1, \ldots, N-1; \tag{8.13}$$

$$P(k) = \frac{1}{N} \sum_{n=0}^{N-1} R(k)\, W^{nk}, \quad k = 0, 1, \ldots, N-1. \tag{8.14}$$

Перейдем к рассмотрению *свертки*. Операция свёртки двух периодических последовательностей $x(nT)$ и $g(nT)$ выражается формулой

$$y(nT) = \sum_{m=0}^{N-1} x(mT) \cdot g(nT - mT), \quad n = 0, 1, \ldots, N-1, \tag{8.15}$$

или эквивалентной формулой

$$y(nT) = \sum_{m=0}^{N-1} x(nT - mT) \cdot g(mT), \quad n = 0, 1, \ldots, N-1. \tag{8.16}$$

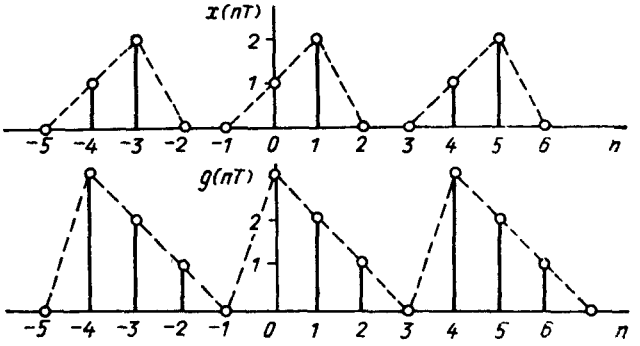


Рис. 8.19. К вычислению свертки функций

Свертка может использоваться при расчете отклика линейной цепи с импульсной характеристикой $g\,(nT)$ на входное воздействие $x\,(nT)$.

Рассмотрим пример вычисления свертки двух N-точечных функций $x\,(nT)$ и $g\,(nT)$, приведенных на рис. 8.19. Вычисление свертки $y\,(T)$ этих функций с помощью выражения (8.15) приводит к следующему значению:

$$y(T) = g(T)\,x(0) + g(0)\,x(T) + g(-T)\,x(2T) +$$
$$+ g(-2T)\,x(3T) = 2\cdot 1 + 3\cdot 2 + 0\cdot 0 + 1\cdot 0 = 8.$$

Как видим, при вычислении свертки используются N значений каждой функции, причем эти N значений могут браться не только из основного, но и из соседнего с ним периода. Поэтому такая свертка, предусматривающая периодичность входящих в нее функций, называется *круговой*.

Прямое выполнение круговой свертки по формулам (8.15) и (8.16) над функциями, выражаемыми последовательностями действительных чисел, требует N^2 умножений действительных чисел. Возможно быстрое выполнение свертки, предусматривающее следующую последовательность действий:

с использованием БПФ вычисление спектров $X\,(k)$ и $G\,(k)$, участвующих в свертке функций $x\,(nT)$ и $g\,(nT)$;

вычисление произведения спектров $Z\,(k) = X\,(k)\,G\,(k)$;

с использованием БПФ вычисление обратного ДПФ от $Z\,(k)$, которое и представляет собой искомый результат свертки $y\,(nT)$.

Таким образом,

$$y(nT) = \sum_{k=0}^{N-1} Z(k)\,W^{-nk} = \sum_{k=0}^{N-1} X(k)\,G(k)\,W^{-nk} =$$
$$= \sum_{k=0}^{N-1} \left(\frac{1}{N} \sum_{n=0}^{N-1} x(nT)\,W^{nk} \right) \left(\frac{1}{N} \sum_{n=0}^{N-1} g(nT)\,W^{nk} \right) W^{-nk}. \qquad (8.17)$$

Несмотря на то, что этот способ вычисления свертки предусматривает трехкратное вычисление ДПФ, он оказывается более экономным, чем прямое вычисление свертки по выражениям (8.15) и (8.16). Определим количество умножений, выполняемых при таком способе вычисления свертки. Быстрое преобразование Фурье, используемое для выполнения прямого ДПФ, как было показано выше, требует $\frac{N}{2}\log_2 N$ умножений комплексных чисел или $2N\log_2 N$ умножений действительных чисел, а для преобразования двух функций $x(nT)$ и $g(nT)$ потребуется $4N\log_2 N$ умножений действительных чисел. Так как спектры $X(k)$ и $G(k)$ представляют собой N-точечные последовательности в общем случае комплексных чисел, то на выполнение операции умножения спектров будет затрачиваться $4N$ умножений действительных чисел. Наконец, ОДПФ, выполняемое с использованием БПФ, потребует $\frac{N}{2}\log_2 N$ умножений комплексных чисел или $2N\log_2 N^2$ умножений действительных чисел. Таким образом, общее количество умножений действительных чисел составит $M = 6N\log_2 N + 4N$. В табл. 8.3 приведено количество умножений при прямом методе выполнения свертки и методе с использованием БПФ для различных значений N. Как видим, при $N > 2^5 = 32$ метод с использованием БПФ по сравнению с прямым методом обеспечивает экономичность тем более высокую, чем больше N.

До сих пор мы рассматривали круговую свертку, предполагая, что используемые в свертке функции являются периодическими с периодом N. Однако на практике чаще возникает необходимость в вычислении свертки от непериодических функций, например при использовании свертки для вычисления отклика линейной системы (если входное воздействие и может выражаться периодической функцией, то импульсная характеристика системы является существенно апериодической функцией). В этом случае допущение периодичности импульсной характеристики привело бы к неверным результатам. В подобных случаях, когда исходные функции не могут быть приняты периодическими, последовательность их отсчетов необходимо дополнить столькими нулевыми значениями, чтобы при вычислении свертки значения исходных функций брались бы лишь из одного основного периода. Например, пусть $x(nT)$ — апериодическая последовательность длиной N_1,

Таблица 8.3

Метод	M при N, равном				
	$2^4=16$	$2^5=32$	$2^6=64$	$2^7=128$	...
Прямой метод ($M=N^2$)	256	1024	4096	16384	...
Метод с использованием БПФ ($M=6N\times\times\log_2 N+4N$)	448	1088	2560	5888	...

$g\ (nT)$ — апериодическая последовательность длиной N_2 отсчетов. В этом случае формируются последовательности отсчетов $x_1\ (nT)$ и $g_1\ (nT)$, каждая длиной $N_1 + N_2 - 1$ отсчетов путем включения дополнительных нулевых значений:

$$x_1(nT) = \begin{cases} x(nT) & \text{при } n = 0, 1, \ldots, N_1 - 1; \\ 0 & \text{при } n = N_1, \ldots, N_1 + N_2 - 1; \end{cases}$$

$$g_1(nT) = \begin{cases} g(nT) & \text{при } n = 0, 1, \ldots, N_2 - 1; \\ 0 & \text{при } n = N_2, \ldots, N_1 + N_2 - 1. \end{cases}$$

При этом искомая свертка последовательностей $x\ (nT)$ и $g\ (nT)$ определяется $(N_1 + N_2 - 1)$-точечной сверткой

$$y(nT) = \sum_{m=0}^{N_1+N_2-1} x_1(m)\, g_1(nT - mT), \quad n = 0, 1, \ldots, N_1 + N_2 - 2$$

или

$$y(nT) = \sum_{m=0}^{N_1+N_2-1} x_1(nT - mT)\, g_1(mT), \quad n = 0, 1, \ldots, N_1 + N_2 - 2.$$

Вычисление такой свертки с использованием БПФ потребует $(N_1 + N_2 - 2)$-точечного ДПФ.

8.7. ЦИФРОВОЙ ФИЛЬТР

ОСНОВНЫЕ ПОНЯТИЯ

Аналоговый фильтр представляет собой частотно-избирательную цепь, осуществляющую некоторое линейное преобразование над непрерывным входным сигналом $u_{вх}\ (t)$. Результатом такого преобразования является непрерывный выходной сигнал $u_{вых}\ (t)$. Особенность цифрового фильтра состоит в том, что указанное выше преобразование выполняется не над непрерывным сигналом $u_{вх}\ (t)$, а над входной цифровой последовательностью $x\ (nT)$, и получаемый на выходе результат преобразования $y\ (nT)$ представляет собой также цифровую последовательность.

На рис. 8.20 представлен простейший аналоговый фильтр. Рассмотрим отклик $u_{вых}\ (t)$ данной rC-цепи на входное воздействие $u_{вх}\ (t)$. Ток $i\ (t)$ в цепи определяется выражением

$$i(t) = C\,\frac{d(u_{вх}(t) - u_{вых}(t))}{dt} = \frac{u_{влх}(t)}{r}. \qquad (8.18)$$

Имея в виду цифровой способ решения задачи, представим входное $u_{вх}\ (t)$ и выходное $u_{вых}\ (t)$ напряжения соответствующими цифровыми

последовательностями $x_n = x(nT)$ и $y_n = y(nT)$. Тогда производная в (8.18) может быть заменена следующим приближением:

$$\left.\frac{d(u_{вх}(t)-u_{вых}(t))}{dt}\right|_{t=nT} \approx \frac{(x_n-y_n)-(x_{n-1}-y_{n-1})}{T}. \qquad (8.19)$$

Подставив (8.19) в (8.18) и решив полученное выражение относительно y_n, получим:

$$y_n = a_0 x_n + a_1 x_{n-1} - b_1 y_{n-1}, \qquad (8.20)$$

где $a_0 = 1/(1 + T/rC)$, $a_1 = -1/(1 + T/rC)$, $b_1 = -1/(1 + T/rC)$.

Полученное разностное уравнение может быть использовано для построения цифрового фильтра 1-го порядка.

Известно, что одной из характеристик фильтра, полностью определяющих выполняемые им преобразования, является переходная характеристика. Переходная характеристика аналогового фильтра есть отклик на воздействие, имеющее форму единичного скачка (единичной функции). Переходная характеристика цифрового фильтра есть цифровая последовательность, представляющая собой определенное линейное преобразование, выполненное над входной последовательностью следующего вида:

$$x(nT) = \begin{cases} 1 \text{ при } n \geqslant 0, \\ 0 \text{ при } n < 0. \end{cases}$$

Решая разностное уравнение (8.20) при данной входной последовательности $x(nT)$, можно получить переходную характеристику h цифрового фильтра 1-го порядка. В табл. 8.4 приведены результаты вычислений при различных значениях шага интегрирования T. В этой же таблице приведена переходная характеристика аналогового фильтра. На рис. 8.21 эти характеристики представлены в форме графиков. Как следует из этих данных, выбирая достаточно малым значение шага интегрирования T, можно с помощью цифрового фильтра воспроизвести с любой точностью переходную характеристику аналогового фильтра.

Обратим внимание на следующую особенность цифрового фильтра. Он выполняет то же преобразование, что и аналоговый фильтр при оп-

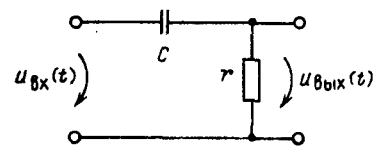


Рис. 8.20. Простейший аналоговый фильтр

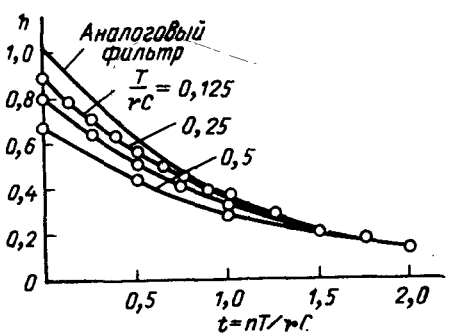


Рис. 8.21. Характеристики аналогового и цифровых фильтров

Таблица 8.4

Фильтр	h при t/rC, равном				
	0	0,5	1,0	1,5	2,0
Аналоговый фильтр	1	0,7788	0,3679	0,2231	0,1353
Цифровой фильтр					
$T=0{,}5\ rC$	0,6667	0,4444	0,2963	0,1975	0,1317
$T=0{,}25\ rC$	0,8000	0,5120	0,3277	0,2097	0,1342
$T=0{,}125\ rC$	0,8889	0,5549	0,3464	0,2163	0,1350

ределенных значениях коэффициентов в выражении (8.20). При иных значениях коэффициентов цифровой фильтр выполняет преобразование, которое может оказаться не реализуемым с помощью аналогового фильтра.

Наряду с переходной характеристикой, выполняемое аналоговым фильтром преобразование может быть описано с помощью передаточной функции, определяемой отношением $H(s) = y(s)/x(s)$, где $x(s)$ и $y(s)$ — преобразование по Лапласу соответственно входного $x(t)$ и выходного $y(t)$ сигналов. Передаточная функция цифрового фильтра определяется выражением $H(z) = y(z)/x(z)$, где $x(z)$ и $y(z)$ представляют собой z-преобразования соответственно входной и выходной цифровых последовательностей. z-преобразование цифровой последовательности $x(nT)$, $n = 0, 1, \ldots$ определяется выражением

$$X(z) = \sum_{n=0}^{\infty} x(nT)\, z^{-n}. \tag{8.21}$$

Обратное z-преобразование выражается контурным интегралом

$$x(nT) = \frac{1}{2\pi j} \oint X(z)\, z^{n-1}\, dz. \tag{8.22}$$

Не вдаваясь в более глубокие подробности z-преобразования, приводим результаты такого преобразования для некоторых частных случаев (табл. 8.5).

Таким образом, применяя z-преобразование к последовательностям, входящим в выражение (8.20), можно это разностное уравнение представить в следующем виде:

$$Y(z) = a_0 X(z) + a_1 X(z)\, z^{-1} - b_1 Y(z)\, z^{-1}. \tag{8.23}$$

Решение данного уравнения реализуется представленной на рис. 8.22 схемой цифрового фильтра 1-го порядка. В схеме цифрового фильтра элемент, обозначенный z^{-1}, выполняет задержку цифровой последовательности на один тактовый период T, числа, показанные на линиях, — коэффициенты, на которые умножаются соответствующие последовательности.

Таблица 8.5

Цифровая последовательность	z-преобразование последовательности
Единичный импульс $x(nT)=\begin{cases}1 & n=0,\\ 0 & \lvert n\rvert>0.\end{cases}$	$X(z)=1.$
Единичный скачок $x(nT)=\begin{cases}1 & n\geqslant 0\\ 0 & n<0\end{cases}$	$X(z)=\dfrac{1}{1-z^{-2}}$
Задержка на один тактовый период T $y(nT)=x(nT-T).$	$Y(z)=X(z)\,z^{-1}.$

Решим выражение (8.23) относительно $Y(z)$:

$$Y(z)=\frac{a_0+a_1 z^{-1}}{1+b_1 z^{-1}}X(z).$$

Отсюда передаточная функция цифрового фильтра 1-го порядка

$$H(z)=(a_0+a_1 z^{-1})/(1+b_1 z^{-1}). \tag{8.24}$$

Передаточная функция аналоговой цепи позволяет получать частотную характеристику цепи, для чего достаточно произвести в $H(s)$ замену s на $j\omega$. Таким образом, $H(j\omega)$ представляет собой коэффициент передачи для гармоники входного сигнала частоты ω. Соответственно передаточная функция цифрового фильтра $H(z)$ при подстановке вместо z цифровой последовательности экспоненциального колебания $e^{j\omega T}$ позволяет определить $H(e^{j\omega T})$ — частотную характеристику цифрового фильтра, которая определяет коэффициент передачи линейной цифровой системы для входной цифровой последовательности, отображающей комплексное экспоненциальное колебание $x(nT)=e^{jn\omega T}$. Так, частотная характеристика цифрового фильтра 1-го порядка, передаточная функция которого определяется выражением (8.24), имеет следующий вид:

$$H(e^{j\omega T})=(a_0+a_1 e^{-j\omega T})/(1+b_1\times e^{-j\omega T}). \tag{8.25}$$

Для цифрового фильтра 2-го порядка разностное уравнение

$$y_n=a_0 x_n+a_1 x_{n-1}+a_2 x_{n-2}-b_1 y_{n-1}-b_2 y_{n-2}, \tag{8.26}$$

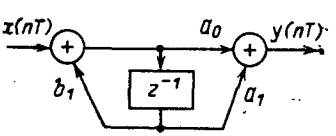


Рис. 8.22. Схема цифрового фильтра 1-го порядка

передаточная функция

$$H(z)=(a_0+a_1 z^{-1}+a_2 z^{-2})/(1+ \\ +b_1 z^{-1}+b_2 z^{-2}). \qquad (8.27)$$

Функциональная схема такого цифрового фильтра показана на рис. 8.23.

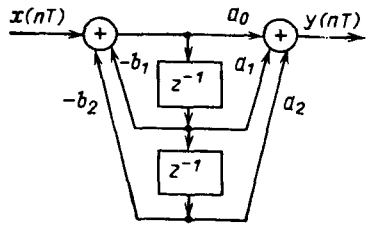


Рис. 8.23. Функциональная схема цифрового фильтра 2-го порядка

В общем случае разностное уравнение цифрового фильтра представляется выражением следующего вида:

$$y_n=\sum_{i=0}^{M} a_i x_{n-i}-\sum_{i=1}^{N} b_i y_{n-i}. \qquad (8.28)$$

При $N=0$ образуется *нерекурсивный* цифровой фильтр, характеризующийся тем, что отклик его представляется суммой некоторого числа членов входной последовательности $x(nT)$.

При $N>0$ образуется *рекурсивный* цифровой фильтр с откликом, выражаемым суммой, в которой оказываются представленными не только члены входной цифровой последовательности, но и предыдущие члены выходной последовательности. Рекурсивные фильтры при определенных условиях могут быть неустойчивыми и значения в выходной последовательности могут неограниченно нарастать.

Общей форме разностного уравнения (8.28) соответствует следующее выражение передаточной функции:

$$H(z)=\frac{a_0+a_1 z^{-1}+a_2 z^{-2}+\ldots+a_M z^{-M}}{1+b_1 z^{-1}+b_2 z^{-2}+\ldots+b_N z^{-N}}. \qquad (8.29)$$

Обратим внимание на следующую особенность цифровых фильтров. Аналоговые фильтры физически реализуемы, если в их передаточных функциях степень полинома числителя не выше степени полинома знаменателя. Цифровые фильтры не предъявляют таких ограничений, и, таким образом, они могут иметь характеристики, добиться которых в аналоговых фильтрах невозможно.

Обратимся к вопросам, связанным с реализацией цифровых фильтров на основании микропроцессорных устройств. При построении цифрового фильтра на МПУ часто сталкиваются с необходимостью решения проблемы повышения быстродействия. На вход цифрового фильтра поступает исходная цифровая последовательность $x(nT)$, на выход выдается последовательность $y(nT)$, представляющая собой результат обработки входной последовательности. Быстродействие может оцениваться допустимым для данной реализации минимальным значением тактового периода $T_{\min}$ входной цифровой последовательности $x(nT)$ или связанным с $T_{\min}$ максимальным значением ширины полосы частот $F_{\max}=1/(2\,T_{\min})$ сигнала, дискретизацией которого получена входная последовательность $x(nT)$. Чем меньше $T_{\min}$ или больше $F_{\max}$, тем выше быстродействие цифрового фильтра. Необходимость

в высоком быстродействии связана со стремлением обрабатывать в реальном масштабе времени широкополосные сигналы.

Однокристальные универсальные микропроцессоры мало пригодны для построения быстродействующих цифровых фильтров. Большее быстродействие достигается при использовании микропроцессоров с разрядно-модульной организацией. Однако и последние часто не способны обеспечить требуемое быстродействие без использования специальных приемов организации обработки информации в фильтре, рассмотрению которых посвящается дальнейшее изложение.

РЕАЛИЗАЦИЯ ЦИФРОВОГО ФИЛЬТРА ПУТЕМ ПОСЛЕДОВАТЕЛЬНОГО ВКЛЮЧЕНИЯ ЭЛЕМЕНТАРНЫХ ФИЛЬТРОВ

Если обозначить $p = z^{-1}$, то числитель выражения $H(z)$ (8.29) представится полиномом $a_0 + a_1 p + \ldots + a_M p^M$. Приравняв полином нулю и вычислив корни полученного уравнения p_i, $i = 1, \ldots, M$, можно представить полином произведением

$$a_0 + a_1 p + \ldots + a_M p^M = a_M (p - p_1)(p - p_2) \ldots (p - p_M). \quad (8.30)$$

Здесь p_i могут быть действительными или комплексными числами. В последнем случае к каждому комплексному p_i среди корней найдется корень p_j, имеющий сопряженное с p_i значение. Пары членов, соответствующие таким комплексно-сопряженным членам в (8.30), можно представить трехчленами вида $p^2 - (p_i + p_j) p + p_i p_j$. При этом коэффициенты трехчлена $-(p_i + p_j)$ и $p_i p_j$ будут действительными членами. Можно также объединить в трехчлены пары двухчленов, соответствующих действительным корням. Тогда полином может быть представлен выражением

$$a_0 + a_1 z^{-1} + \ldots + a_M z^{-M} = \prod_{i=1}^{m} (a_{0i} + a_{1i} z^{-1} + a_{2i} z^{-2}). \quad (8.31)$$

Если M нечетно, число двухчленов будет нечетным и один из двухчленов не будет иметь пары. Этот двухчлен в (8.31) отобразится трехчленом, в котором $a_{2i} = 0$.

Аналогично в форме произведения может быть представлен и знаменатель выражения (8.29)

$$1 + b_1 z^{-1} + \ldots + b_N z^{-N} = \prod_{i=1}^{n} (1 + b_{1i} z^{-1} + b_{2i} z^{-2}). \quad (8.32)$$

Таким образом, отношение рассмотренных полиномов, т. е. передаточная функция $H(z)$, может быть представлено произведением дробей,

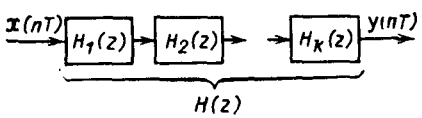


Рис. 8.24. Последовательное включение цифровых фильтров

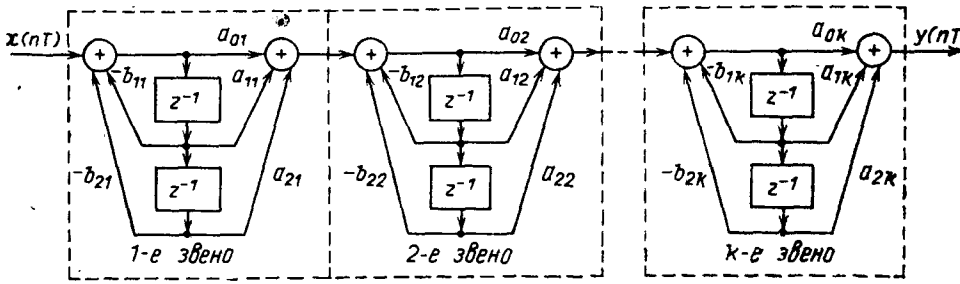


Рис. 8.25. Последовательное соединение фильтров 2-го порядка

у которых числитель и знаменатель являются полиномами степени не выше второй:

$$H(z)=\prod_{i=1}^{k}\frac{a_{0i}+a_{1i}z^{-1}+a_{2i}z^{-2}}{1+b_{1i}z^{-1}+b_{2i}z^{-2}}=H_1(z)H_2(z)\ \ldots\ H_k(z), \qquad (8.33)$$

где

$$H_i(z)=\frac{a_{0i}+a_{1i}z^{-1}+a_{2i}z^{-2}}{1+b_{1i}z^{-1}+b_{2i}z^{-2}}.$$

Для получения произведения передаточных функций $H(z)=\prod_{i=1}^{k}H_i(z)$ необходимо устройства, реализующие элементарные функции $H_i(z)$, включить последовательно, как показано на рис. 8.24. Для каждой элементарной передаточной функции $H_i(z)$ может быть построен элементарный цифровой фильтр (2-го или 1-го порядка), а последовательное их соединение (рис.8.24) обеспечит результирующую передаточную функцию $H(z)$. Соответствующее выражению (8.33) соединение фильтров 2-го порядка показано на рис. 8.25.

Пример 8.1. Пусть требуется построить цифровой фильтр 5-го порядка с передаточной функцией

$$H(z)=\frac{2-3z^{-1}+2z^{-2}-0,5z^{-3}}{1+1,9z^{-1}+1,8z^{-2}+0,95z^{-3}+0,3z^{-4}+0,05z^{-5}}.$$

Нули передаточной функции: $1+j$; $1-j$; 2; полюсы: $-1+j$; $-1-j$; $-1+2j$; $-1-2j$; -2.

Следовательно, передаточная функция может быть представлена в следующем виде:

$$H(z)=\frac{-0,5(z^{-1}-1-j)(z^{-1}-1+j)(z^{-1}-2)}{0,05(z^{-1}+1-j)(z^{-1}+1+j)(z^{-1}+1-2j)(z^{-1}+1+2j)(z^{-1}+2)}=$$

$$=\frac{-0,5(z^{-2}-2z^{-1}+2)(z^{-1}-2)}{0,05(z^{-2}+2z^{-1}+2)(z^{-2}+2z^{-1}+5)(z^{-1}+2)}=$$

$$=\frac{1}{1+z^{-1}+0,5z^{-2}}\,\frac{-1+z^{-1}-0,5z^{-2}}{1+0,4z^{-1}+0,2z^{-2}}\,\frac{-2+z^{-1}}{1+0,5z^{-1}}.$$

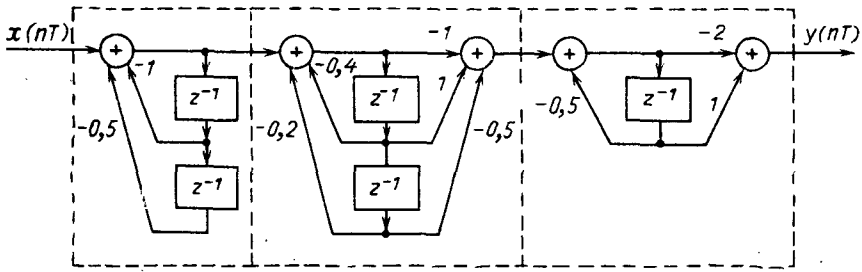


Рис. 8.26. Пример структуры фильтра с последовательным включением звеньев 1-го и 2-го порядков

Таким образом, передаточная функция оказалась представленной в форме произведения передаточных функций трех элементарных цифровых фильтров и реализуемый фильтр может иметь структуру, представленную на рис. 8.26.

При таком последовательном соединении элементарных цифровых фильтров выполнение операций, связанных с реализацией отдельных элементарных цифровых фильтров, может производиться в отдельных микропроцессорах. Каждый микропроцессор в этом случае выполняет не всю обработку, но лишь ее часть, выдавая промежуточные значения для дальнейшей обработки в следующий микропроцессор. Последний микропроцессор будет выдавать конечный результат обработки $y(nT)$. Если последовательно включено k микропроцессоров, то при поступлении на вход данных исходной цифровой последовательности $x(nT)$ с временны́м интервалом T, время, которое затрачивается на обработку, связанную с формированием каждого выходного значения, может составлять kT. Очевидно, если бы вся обработка была бы сосредоточена в одном микропроцессоре, он затрачивал бы на обработку время kT и допускал поступление на вход данных с временны́м интервалом kT. Следовательно, в рассматриваемом случае, когда обработка распределяется между k микропроцессорами быстродействие (а значит, и широкополосность) цифрового фильтра вырастает в k раз.

РЕАЛИЗАЦИЯ ЦИФРОВОГО ФИЛЬТРА ПАРАЛЛЕЛЬНЫМ ВКЛЮЧЕНИЕМ ЭЛЕМЕНТАРНЫХ ФИЛЬТРОВ

Возможен и другой способ распараллеливания процесса обработки сигнала. Выражение $H(z)$ (8.29) может быть представлено не произведением, как в рассмотренном выше случае, а суммой элементарных дробей, знаменатели которых формируются, как и в предыдущем случае, путем определения полюсов выражения (8.29):

$$H(z) = \sum_{i=1}^{k} \frac{c_{0i} + c_{1i} z^{-1}}{1 + b_{1i} z^{-1} + b_{2i} z^{-2}} = \sum_{i=1}^{k} H_i(z). \qquad (8.34)$$

Для получения результирующей передаточной функции $H(z)$ в этом случае потребуется параллельное включение элементарных цифровых фильтров, каждый из которых реализует одну из составляющих передаточной функции в выражении (8.34). Как и в последовательной реализации, для построения элементарных фильтров могут быть использованы отдельные микропроцессоры. При этом быстродействие фильтра возрастает во столько раз, каково число микропроцессоров, участвующих в обработке сигналов. Отличие от предыдущего случая в том, что результаты обработки появляются на выходе с запаздыванием, равным не kT, а T.

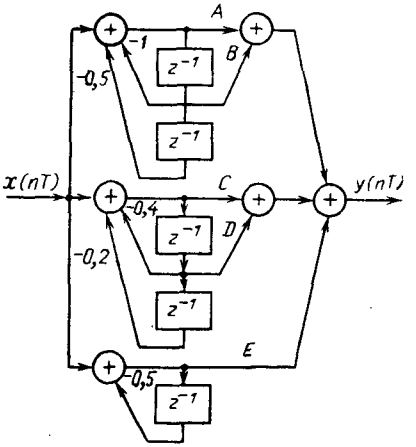


Рис. 8.27. Схема цифрового фильтра с параллельным включением элементарных фильтров

Пример 8.2. Рассмотрим реализацию той же передаточной функции, что и в предыдущем примере, и представим ее в форме

$$H(z) = \frac{A+Bz^{-1}}{1+z^{-1}+0,5z^{-2}} + \frac{C+Dz^{-1}}{1+0,4z^{-1}+0,2z^{-2}} + \frac{E}{1+0,5z^{-1}}.$$

Здесь A, B, C, D, E — коэффициенты, значения которых находятся из следующего условия: после приведения суммы дробей к общему знаменателю выражение в числителе должно совпасть с числителем выражения заданной передаточной функции $H(z)$. Эти коэффициенты имеют значения: $A = 16,551726$; $B = -8,965525$; $C = -2,55124$; $D = -4,85517$; $E = 21,10345$.

Таким образом, передаточная функция представляется в виде следующей суммы элементарных дробей:

$$H(z) = \frac{-16,55173-8,96553z^{-1}}{1+z^{-1}+0,5z^{-2}} + \frac{-2,55124-4,85517z^{-1}}{1+0,4z^{-1}+0,2z^{-2}} + \frac{21,10345}{1+0,5z^{-1}}.$$

Этой форме передаточной функции соответствует схема цифрового фильтра с параллельным включением элементарных фильтров, приведенная на рис. 8.27.

ПАРАЛЛЕЛЬНОЕ ВЫПОЛНЕНИЕ ОПЕРАЦИЙ УМНОЖЕНИЯ В ЭЛЕМЕНТАРНЫХ ЦИФРОВЫХ ФИЛЬТРАХ

На рис. 8.28 показана схема цифрового фильтра 2-го порядка и алгоритм обработки сигнала в нем. Нумерация точек в схеме цифрового фильтра соответствует нумерации переменных y_i, хранящих формируемые в этих точках значения, т. е. y_1, y_2, y_3, y_4 (и, следовательно, ячейки памяти, выделенные для хранения значений этих переменных) имеют значения, совпадающие со значениями величин в точках *1*, *2*, *3*, *4* схемы цифрового фильтра.

Блоки *1*, *2* схемы алгоритма значения y_2, y_1, сформированные в предыдущем повторении цикла, передают соответственно в y_3, y_2 и отражают, таким образом, задержку, предусмотренную в схеме цифрового фильтра между точками 3 и 2, 2 и 1. Ячейка y_5 используется как вспомогательная ячейка для формирования произведений.

Из 14 содержащихся в схеме алгоритма блоков пять блоков (блоки *4*, *6*, *8*, *10*, *12*) предусматривают выполнение операции умножения, и основное время, затрачиваемое на исполнение алгоритма, связано именно с выполнением этих блоков. Возможно ускорение исполнения алгоритма, если предусмотреть параллельное выполнение операций умножения, т. е. пять предусмотренных в алгоритме операций умножения выполнять одновременно, используя пять различных устройств умножения. Так как время, затрачиваемое на исполнение других блоков в схеме алгоритма относительно невелико, то при параллельном выполнении умножения примерно в пять раз сократится время одно-

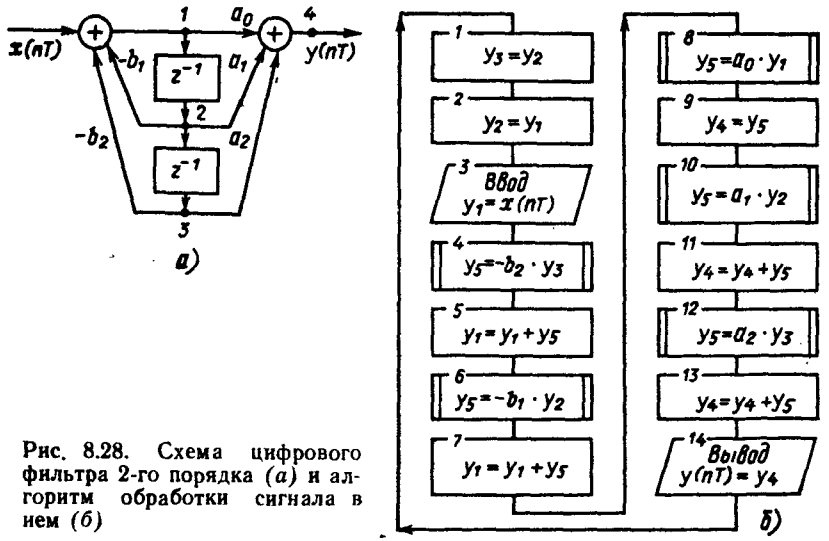


Рис. 8.28. Схема цифрового фильтра 2-го порядка (*а*) и алгоритм обработки сигнала в нем (*б*)

кратного исполнения цикла алгоритма и, следовательно, в пять раз возрастут быстродействие цифрового фильтра и предельная широкополосность обрабатываемых фильтром сигналов.

УСКОРЕНИЕ ВЫПОЛНЕНИЯ ОПЕРАЦИИ УМНОЖЕНИЯ В ЭЛЕМЕНТАРНОМ ЦИФРОВОМ ФИЛЬТРЕ

Дальнейшее увеличение быстродействия цифрового фильтра может быть достигнуто при использовании приемов, ускоряющих выполнение операции умножения в элементарных фильтрах. Одним из таких приемов является аппаратная реализация операции умножения, т. е. выполнение операции умножения в специализированной БИС, предназначенной для реализации этой операции. Ниже будут рассматриваться методы ускорения при программной реализации операции умножения. В схеме алгоритма на рис. 8.28,*б* предполагалось, что операция умножения реализуется подпрограммой, к которой производится обращение из различных точек главной программы. Использование подпрограммы умножения сокращает объем программы и емкость памяти, требуемой для ее хранения. Однако процессы, связанные с обращением к подпрограмме и возвратом в главную программу, снижают скорость выполнения программы и быстродействие цифрового фильтра. Если стремиться к увеличению быстродействия фильтра, следует, исключив подпрограмму, операцию умножения описывать непосредственно в главной программе. Так как это описание операции умножения в главной программе будет повторяться во всех местах, где требуется выполнение этой операции, то объем программы увеличится и для ее хранения потребуется бóльшая емкость памяти.

Следует иметь в виду, что участки программы, описывающие операцию умножения (или подпрограмма умножения), должны быть построены так, чтобы время их исполиения не зависело от выбора ветви программы при выполнении команд условного перехода. Выполнение этого условия, необходимое для того, чтобы время исполнения цикла обработки было бы постоянно, требует включения в программу команд холостой операции. Это приводит к тому, что наличие в разрядах множителя нулей не приводит к уменьшению времени выполнения операции умножения. Этот недостаток в случае, когда ие предполагается варьирование значений коэффициентов a_0, a_1, a_2, b_1, b_2, можно устранить, используя следующий прием умножения.

Представим значение коэффициента a_0 в форме $a_0 = |a_0| \operatorname{sign} a_0$, где $|a_0|$ — модуль коэффициента a_0,

$$\operatorname{sign} a_0 = \begin{cases} -1 & \text{при } a_0 < 0, \\ +1 & \text{при } a_0 \geqslant 0. \end{cases}$$

Представив в подобной форме и все другие коэффициенты, разностное уравнение (8.26) цифрового фильтра 2-го порядка можно записать в следующем виде:

$$y_n = \underbrace{|a_0|}_{c_0}\underbrace{\operatorname{sign} a_0 x_n}_{z_0} + \underbrace{|a_1|}_{c_1}\underbrace{\operatorname{sign} a_1 x_{n-1}}_{z_1} + \underbrace{|a_2|}_{c_2}\underbrace{\operatorname{sign} a_2 x_{n-2}}_{z_2} +$$
$$+ \underbrace{|b_1|}_{c_3}\underbrace{(-\operatorname{sign} b_1) y_{n-1}}_{z_3} + \underbrace{|b_2|}_{c_4}\underbrace{(-\operatorname{sign} b_2) y_{n-2}}_{z_4} = c_0 z_0 + c_1 z_1 +$$
$$+ c_2 z_2 + c_3 z_3 + c_4 z_4. \qquad (8.35)$$

При такой записи разностного уравнения все коэффициенты c_0, $c_1, \ldots, c_4$ имеют положительные значения. Будем считать, что значения переменных $z_1, \ldots, z_4$ представляются в дополнительном коде. Пусть коэффициенты c_i выражаются n-разрядными числами, меньшими единицы:

$$c_i = c_i^{(1)} 2^{-1} + c_i^{(2)} 2^{-2} + \ldots + c_i^{(n)} 2^{-n}, \qquad (8.36)$$

где $c_i^{(j)}$ — значение j-го разряда коэффициента c_i (отсчет разрядов принят от старшего разряда в сторону младших разрядов). Представим выражение (8.36) в форме Горнера

$$c_i = (\ldots((c_i^{(n)} 2^{-1} + c_i^{(n-1)}) 2^{-1} + c_i^{(n-2)}) 2^{-1} + \ldots + c_i^{(1)}) 2^{-1}. \qquad (8.37)$$

Аналогично в форме Горнера можно представить значения переменных z_i, входящих в выражение (8.35)

$$z_i = (\ldots((z_i^{(n)} 2^{-1} + z_i^{(n-1)}) 2^{-1} + z_i^{(n-2)}) 2^{-1} + \ldots + z_i^{(1)}) 2^{-1} + z_i^{(0)}. \qquad (8.38)$$

Здесь член $z_i^{(0)}$ соответствует знаковому разряду в z_i.

После подстановки (8.38) в (8.35) и соответствующей группировки членов получим

$$y_n = \left(\ldots\left(\left(\sum_{i=1}^{5} c_i z_i^{(n)} 2^{-1} + \sum_{i=1}^{5} c_i z_i^{(n-1)}\right) 2^{-1} + \right.\right.$$
$$\left.\left. + \sum_{i=1}^{5} c_i z_i^{(n-2)}\right) 2^{-1} + \ldots\right) 2^{-1} + \sum_{i=1}^{5} c_i z_i^{(0)}. \qquad (8.39)$$

Сумма $\sum_{i=1}^{5} c_i z_i^{(j)}$ является при заданном значении коэффициентов c_i функцией значений j-го разряда пяти величин: $z_1^{(j)}, z_2^{(j)}, \ldots, z_5^{(j)}$, т. е.

$$\sum_{i=1}^{5} c_i z_i^{(j)} = \psi(z_1^{(j)}, z_2^{(j)}, \ldots, z_5^{(j)}). \qquad (8.40)$$

Такая функция ψ пяти аргументов задается на $2^5 = 32$ комбинациях значений аргументов. Следовательно, для задания значений коэффициентов c_i можно предварительно вычислить значение этой суммы для всех 32 возможных комбинаций значений $z_1^{(j)}, z_2^{(j)}, \ldots, z_5^{(j)}$. Вычисленные таким образом значения функции ψ можно хранить в ПЗУ. При этом адресами ячеек ПЗУ могут служить комбинации значений аргументов $z_1^i \ldots z_5^i$, а содержимое ячеек — значение ψ. В этом случае для

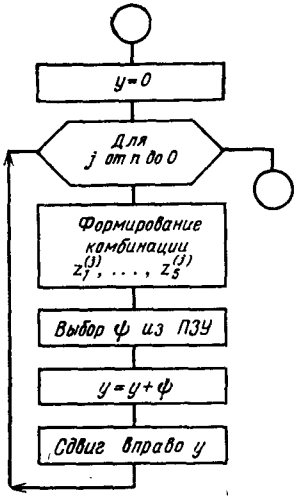


Рис. 8.29. Алгоритм ускоренного выполнения умножений

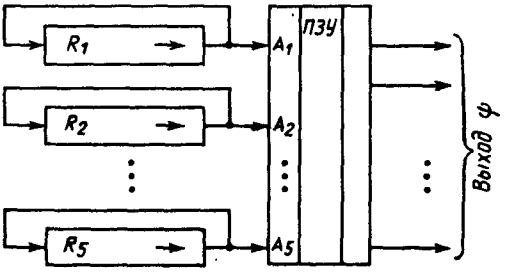


Рис. 8.30. Аппаратная реализация аргументов функции ψ

нахождения каждой из входящих в выражение (8.39) сумм достаточно лишь одного обращения в ПЗУ. Алгоритм рассмотренного способа вычислений показан на рис. 8.29.

Трудности реализации этого метода могут быть связаны с формированием комбинаций значений $z_1^{(j)} \ldots z_5^{(j)}$, составленных из значений j-го разряда пяти переменных z_i. Наиболее эффективна в этом случае аппаратиая реализация, представленная на рис. 8.30. Здесь регистры $R_1 \ldots R_5$ хранят соответственно переменные $z_1 \ldots z_5$. Регистры построеиы так, что в них обеспечивается выполнеиие операции циклического сдвига вправо. При каждом сдвиге, выполняемом одновременно над содержимым всех регистров, на их выходах образуется кодовая комбинация $z_1^{(j)} \ldots z_5^{(j)}$ (j — номер сдвига). Эта кодовая комбинация используется в качестве адреса, по которому производится обращение в постоянное ЗУ, хранящее функцию ψ.

Таким образом, если разрядиость регистров равна m, то вычисление выражения y_n завершается после m-кратного сдвига содержимого регистров.

Этот метод ускоренного умножения рассматривался примеиительно к фильтру 2-го порядка с пятью коэффициентами, на которые умножались переменные. Увеличение порядка фильтра на единицу может вызвать увеличение числа коэффициентов на две единицы и, следовательно, число коэффициентов в фильтре k-го порядка может составить $2k + 1$. Таким же будет и число аргументов функции ψ, и емкость памяти, требуемая для хранения значений этой функции, составит $2^{2k+1} = 2 \cdot 4^k = 2\,e^{k \ln 4}$. Таким образом, с ростом порядка фильтра происходит экспоненциальиый рост емкости памяти, предназначенной для хранения функции ψ. Рост емкости памяти в свою очередь приводит к увеличению времени выборки чисел из памяти. Этот недостаток не проявляется, если фильтр высокого порядка строится путем

каскадного или параллельного соединения звеньев фильтра 1-го и 2-го порядков.

Выполнение фильтра с помощью рассматриваемого метода ускоренного умножения оказывается невозможным, если предусматривается изменения значений коэффициентов фильтра в процессе его работы.

СПИСОК ЛИТЕРАТУРЫ

1. **Балашов Е. П., Пузанков Д. В.** Микропроцессоры и микропроцессорные системы: Учеб. пособие для вузов /Под ред. В. Б. Смолова.— М.: Радио и связь, 1981.
2. **Березенко А. И., Корягин Л. Н., Назарьян А. Р.** Микропроцессорные комплекты повышенного быстродействия.— М.: Радио и связь, 1981.
3. **Березенко А. И.** Микропроцессорные комплекты общего применения.— М.: Машиностроение, 1982.
4. **Васильев Н. П., Горовой В. Р.** Микропроцессоры. Аппаратурно-программные средства отладки: Учеб. пособие для вузов/Под ред. Л. Н. Преснухина. — М.: Высшая школа, 1984.
5. **Воробьев Н. В., Вериер В. Д.** Микропроцессоры: Элементная база и схемотехника средств сопряжения: Учеб. пособие для вузов/Под ред. Л. Н. Преснухина. — М.: Высшая школа, 1984.
6. **Гольденберг Л. М., Матюшкин Б. Д., Поляк М.Н.** Цифровая обработка сигналов: Справочник.— М.: Радио и связь, 1985.
7. **Каган Б. М., Сташин В.В.** Микропроцессоры в цифровых системах. —М.: Энергия, 1979.
8. **Клингман Э.** Проектирование специализированных микропроцессорных систем: Пер. с англ.— М.: Мир, 1985.
9. **Мик Дж., Брик Дж.** Проектирование микропроцессорных устройств с разрядно-модульной организацией: Пер. с англ. — В 2-х кн. — М.: Мир, 1984.
10. **Микропроцессорные** комплекты интегральных схем: Состав и структура: Справочник/Под ред. А.А. Васенкова, В.А. Шахнова. — М.: Радио и связь, 1982.
11. **Прангишвили И. В., Стецюра Г. Г.** Микропроцессорные системы. — М.: Наука, 1980.
12. **Проектирование** цифровых систем на комплектах микропрограммируемых БИС/Булгаков С. С. и др.; Под ред. В. Г. Колесникова. — М.: Радно и связь, 1984.
13. **Уокерли Дж.** Архитектура и программирование микро ЭВМ: Пер. с англ.— В 2-х кн.— М.: Мир, 1984.
14. **Шаньгин В. Ф., Костин А. Е.** Микропроцессоры: Организация вычислительных процессов на микроЭВМ.— М.: Высшая школа, 1984.
15. Микопроцессоры и микропроцессорные комплекты интегральных микросхем: Справочник. В 2-х т. / В.-Б. Б. Абрайтис, Н. Н. Аверьяиов, А. И. Белоус и др.; Под ред. В. А. Шахиова. — М.: Радио и связь, 1988.

ОГЛАВЛЕНИЕ