Б. А. КАЛАБЕКОВ

# ЦИФРОВЫЕ УСТРОЙСТВА И МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

**Издание второе, переработанное и дополненное**

*Допущено Министерством связи Российской Федерации в качестве учебника для средних специальных учебных заведений связи по специальностям 2004, 2005, 2006*

**Москва**
**ГОРЯЧАЯ ЛИНИЯ - ТЕЛЕКОМ**
**2007**

**ББК.32.97**
**К 17**

**Калабеков Б. А.**

К17 Цифровые устройства и многопроцессорные системы: Учебник для техникумов связи. – Горячая линия–Телеком, 2007. – 336 с.: ил.

**ISBN 5-93517-008-6.**

Излагаются принципы построения и функционирования интегральных логических элементов, методы синтеза логических устройств комбинационного и последовательного типов, различных узлов цифровых устройств, микропрограммных автоматов на основе схемной и программируемой логики, а также методы контроля цифровых устройств. Рассматриваются микропроцессоры серий 580, 1813, 1816, 1830, их программирование и вопросы построения микропроцессорных систем.

Для учащихся колледжей и техникумов, готовящих специалистов телекоммуникационного профиля.

**ББК 32.97**

*Адрес издательства в Интернет WWW.TECHBOOK.RU*
*e-mail: radios_hl@mtu-net.ru.*

Учебное издание

**Калабеков Бениамин Аршакович**

**Цифровые устройства и микропроцессорные системы**

Учебник

Обложка художника В. Г. Ситникова

ЛР № 071825 от 16 марта 1999 г.
Подписано в печать 07.09.06. Формат 60x88/16. Печать офсетная
Уч.-изд. л. 21,5. Доп. тираж 1000 экз. Изд № 6008.

ISBN 5-93517-008-6.

## Предисловие

В любой сфере человеческой деятельности — в науке, технике, производстве — методы и средства вычислительной техники направлены на повышение производительности труда. В связи с этим уровень специалистов в существенной мере определяется их подготовкой в следующих направлениях, связанных с применением средств вычислительной техники:

автоматизированное управление технологическими процессами, включая автоматизированные контроль и диагностику технических средств;

использование ЭВМ для автоматизированного проектирования, научных исследований, административно-организационного управления.

Для первого из этих направлений требуются специалисты в области цифровых методов и цифровых устройств, микропроцессорных систем и ЭВМ. Настоящий учебник ориентирован на это направление и предназначен для базовой подготовки, которая позволила бы в специальных дисциплинах рассматривать различные приложения средств вычислительной техники.

При написании книги материал изданного под тем же названием в 1987 г. учебника подвергся существенной переработке и дополнению. Основные изменения, внесенные в новое издание учебника, состоят в следующем:

исключена глава, посвященная импульсным устройствам, не соответствующая общей направленности учебника;

исключен раздел, связанный с изучением микропроцессорных систем на комплекте серии 589;

рассмотрены микропроцессоры серии 1813, используемые в системах цифровой обработки аналоговых сигналов, и микроконтроллеры серий 1816, 1830, используемые в различных устройствах управления.

К сожалению, ограниченный объем книги не позволил рассмотреть более сложный процессор серии 1867 (аналог американского микропроцессора серии TMS320), ориентированный на системы цифровой обработки сигналов.

# Глава 1. Логические основы цифровой техники

## 1.1. ЛОГИЧЕСКИЕ ФУНКЦИИ

### Понятие о логической функции и логическом устройстве

Для обозначения различной информации — предметов, понятий, действий — мы пользуемся словами. Запись слов производится с помощью букв из некоторого их набора, называемого алфавитом.

В цифровой технике для тех же целей пользуются *кодовыми словами*. Особенность этих слов заключается в том, что все они имеют чаще всего одинаковую длину (т.е. состоят из одного и того же количества букв) и для их построения используется простейший алфавит из двух букв. Эти буквы принято обозначать символами 0 и 1. Таким образом, кодовое слово в цифровой технике есть определенной длины последовательность символов 0 и 1, например 10111011. Такими кодовыми словами могут представляться и числа, в этом случае 0 и 1 совпадают по смыслу с обычными арабскими цифрами. При представлении кодовым словом некоторой нечисловой информации, чтобы отличать символы 0 и 1 от арабских цифр, будем эти символы называть *логическим нулем* и *логической единицей* и обозначать далее *лог.0* и *лог.1*.

Если длина кодовых слов составляет $n$ разрядов, то можно построить $2^n$ различных комбинаций — кодовых слов. Например, при $n = 3$ можно построить $2^3 = 8$ слов: 000, 001, 010, 011, 100, 101, 110, 111.

Информация, которая передается между отдельными узлами (блоками) сложного цифрового устройства, представляется в виде кодовых слов. Таким образом, на входы каждого узла поступают кодовые слова, на выходе узла образуется новое кодовое слово, представляющее собой результат обработки входных слов. Выходное слово зависит от того, какие слова поступают на входы узла. Поэтому можно говорить, что выходное слово есть функция, для которой аргументами являются входные слова. Для того чтобы подчеркнуть особенность таких функций, состоящую в том, что функция и ее аргументы могут принимать значения *лог.0* и *лог.1*, будем эти функции называть *функциями алгебры логики* (ФАЛ).

Устройства, предназначенные для формирования функций алгебры логики, называются *логическими устройствами* или *цифровыми устройствами*. Цифровые устройства (либо их узлы) можно делить на типы по различным признакам.

По способу ввода и вывода кодовых слов различают логические устройства последовательного, параллельного и смешанного действия.

На входы *устройства последовательного действия* символы кодовых слов поступают не одновременно, а последовательно во времени, символ за символом (в так называемой последовательной форме). В такой же последовательной форме выдается выходное слово. Пример такого устройства показан на рис. 1.1,*а*. Как нетрудно сообразить, устройство на рисунке выявляет несовпадение символов на входах, выдавая *лог.1* при несовпадении и *лог.0* при совпадении символов (действительно, при несовпадении входных символов, когда Вх1 = 1 и Вх2 = 0 или Вх1 = 0 и Вх2 = 1, на выходе устройства Вых = 1, при совпадении входных символов, когда Вх1 = 1 и Вх2 = 1 или Вх1 = 0 и Вх2 = 0, на выходе Вых = 0).

На входы *устройства параллельного действия* все *n* символов каждого входного кодового слова подаются одновременно (в так называемой параллельной форме). В такой же форме образуется на выходе выходное слово. Очевидно, при параллельной форме приема и выдачи кодовых слов в устройстве необходимо иметь для каждого разряда входного (выходного) слова отдельный вход (выход). Пример такого устройства показан на рис. 1.1,*б*. Устройство выполняет над разрядами входных слов ту же логическую операцию (выявляя несовпадение символов соответствующих разрядов входных слов), что и устройство, показанное на рис. 1.1,*а*, но в параллельной форме. Входы устройства разделены на две группы (I и II), каждая из которых предназначена для приема трехразрядного входного кодового слова в параллельной форме. На выходах устройства также в параллельной форме получается трехразрядное выходное слово.

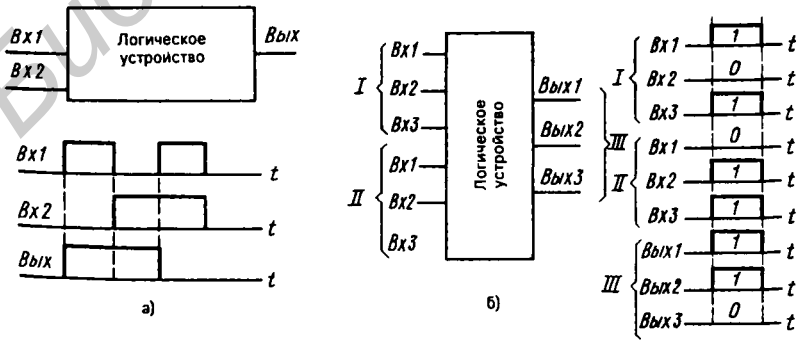



Рис. 1.1

В устройствах *смешанного действия* входные и выходные кодовые слова представляются в разных формах. Например, входные слова — в последовательной форме, выходные — в параллельной. Устройства смешанного действия могут использоваться для преобразования кодовых слов из одной формы представления в другую (из последовательной формы в параллельную или наоборот).

По способу функционирования логические устройства (и их схемы) делят на два класса: комбинационные устройства (и соответственно комбинационные схемы) и последовательностные устройства (последовательностные схемы).

В *комбинационном устройстве* (называемом также *автоматом без памяти*) каждый символ на выходе (*лог.0* или *лог.1*) определяется лишь символами (*лог.0* или *лог.1*), действующими в данный момент времени на входах устройства, и не зависит от того, какие символы ранее действовали на этих входах. В этом смысле комбинационные устройства лишены памяти (они не хранят сведений о прошлом работы устройства).

В *последовательностных устройствах* (или *автоматах с памятью*) выходной сигнал определяется не только набором символов, действующих на входах в данный момент времени, но и внутренним состоянием устройства, а последнее зависит от того, какие наборы символов действовали на входах во все предшествующие моменты времени в процессе работы устройства. Поэтому можно говорить, что последовательностные устройства обладают памятью (они хранят сведения о прошлом работы устройства).

Рассмотрим примеры комбинационного и последовательностного устройства. Пусть устройство (рис. 1.2,*а*) предназначено для формирования на выходе сигнала, определяющего совпадение сигналов на входах: на выходе формируется *лог.1* в случаях, когда на обоих входах действует либо *лог.1*, либо *лог.0*; если на одном из входов действует *лог.1*, а на другом — *лог.0*, то на выходе устройства образуется *лог.0*.

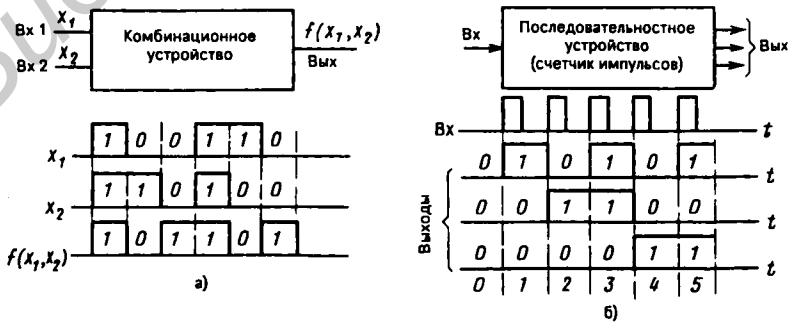



Рис. 1.2

Такое устройство является комбинационным, в котором значение формируемой на выходе логической функции определяется лишь значениями ее аргументов в данный момент времени.

Рассмотрим другой пример. Счетчик на рис. 1.2,*б* подсчитывает импульсы. В каждый момент времени его состояние соответствует числу поступивших на вход импульсов. Выходная информация определяется тем, каково было состояние счетчика до данного интервала времени и поступает или нет на вход импульс в данном интервале времени. Таким образом, данное устройство является последовательностным устройством.

### Способы задания логических функций

В классической математике для задания функции обычно используются два способа: аналитический (запись формулой) и табличный (таблицами значений функции, какие приводятся, например, в справочниках). Подобными же способами могут задаваться логические функции.

При табличном способе строится так называемая таблица истинности, в которой приводятся все возможные сочетания значений аргументов и соответствующие им значения логической функции. Так как число таких сочетаний конечно, таблица истинности позволяет определять значение функции для любых значений аргументов (в отличие от таблиц математических функций, которые позволяют задавать значения функции не для всех, а лишь для некоторых значений аргументов).

Таблица истинности для логических функций одного аргумента приведена в табл. 1.1. Существуют всего четыре функции одного аргумента.

*Таблица 1.1*

| Аргумент $x$ | Функции | | | |
|---|---|---|---|---|
| | $f_0(x)$ | $f_1(x)$ | $f_2(x)$ | $f_3(x)$ |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |

Если число аргументов функции равно $n$, то число различных сочетаний (наборов) значений аргументов составляет $2^n$, а число различных функций $n$ аргументов $2^{2^n}$. Так, при $n = 2$ число наборов значений аргументов равно $2^2 = 4$, число функций $2^4 = 16$. Таблица истинности функций двух аргументов представлена табл. 1.2.

Возможен и аналитический способ записи логической функции. В обычной математике аналитический способ представления функции предполагает запись функции в виде математического выражения, в котором аргументы функции связываются определенными математи-

ческими операциями. Подобно этому аналитический способ задания логической функции предусматривает запись функции в форме логического выражения, показывающего, какие и в какой последовательности должны выполняться логические операции над аргументами функции.

*Таблица 1.2*

| Аргументы | | Функции | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $f_0$ | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ | $f_8$ | $f_9$ | $f_{10}$ | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

В табл. 1.3 приведен перечень логических операций, используемых при записи логических выражений.

Функции одного аргумента (табл. 1.1) представляются следующими выражениями:

$f_0(x) = 0$ (константа 0), $\quad f_1(x) = x$,

$f_2(x) = \overline{x}$, $\quad f_3(x) = 1$ (константа 1).

Устройства, реализующие функции $f_0(x)$, $f_1(x)$ и $f_3(x)$, оказываются тривиальными. Как видно из рис. 1.3, формирование функции $f_0(x)$ требует разрыва между входом и выходом с подключением выхода к общей точке схемы, формирование функции $f_1(x)$ — соединения входа с выходом, формирование функции $f_3(x)$ — подключения выхода к источнику напряжения, соответствующего *лог.1*. Таким образом, из всех функций одного аргумента практический интерес может представлять лишь функция $f_2(x) = \overline{x}$ (логическое НЕ).

Из сравнения таблиц истинности функций $f_0,\ldots,f_{15}$ (табл. 1.2) с таблицами истинности логических операций (табл. 1.3) следует:

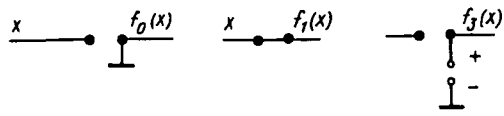



Рис. 1.3

*Таблица 1.3*

| Обозначение логических операций | | Таблица истинности | | | | | Как читается | Название операции |
|---|---|---|---|---|---|---|---|---|
| | | $x_1$ | 0 | 0 | 1 | 1 | | |
| Основное | Дополнительные | $x_2$ | 0 | 1 | 0 | 1 | | |
| $x_1 \cdot x_2$ | $x_1 x_2$<br>$x_1 \wedge x_2$<br>$x_1 \& x_2$ | $x_1 \cdot x_2$ | 0 | 0 | 0 | 1 | $x_1$ и $x_2$ | Конъюнкция; логическое И; логическое произведение |
| $x_1 \vee x_2$ | $x_1 + x_2$ | $x_1 \vee x_2$ | 0 | 1 | 1 | 1 | $x_1$ или $x_2$ | Дизъюнкция; логическое ИЛИ; логическая сумма |
| $x_1 \to x_2$ | $x_1 \supset x_2$ | $x_1 \to x_2$ | 1 | 1 | 0 | 1 | если $x_1$, то $x_2$;<br>$x_1$ влечет $x_2$;<br>$x_1$ имплицирует $x_2$ | Импликация |
| $x_1 \backsim x_2$ | $x_1 \equiv x_2$<br>$x_1 \leftrightarrow x_2$ | $x_1 \backsim x_2$ | 1 | 0 | 0 | 1 | $x_1$ эквивалентно $x_2$ | Эквивалентность; равнозначность |
| $x_1 \oplus x_2$ | $x_1 \supset x_2$ | $x_1 \oplus x_2$ | 0 | 1 | 1 | 0 | либо $x_1$, либо $x_2$;<br>$x_1$ неэквивалентно $x_2$ | Сумма по модулю; неравнозначность; исключающее ИЛИ |
| $x_1 \Delta x_2$ | $x_1 \not\to x_2$<br>$x_1 \not\supset x_2$ | $x_1 \Delta x_2$ | 0 | 0 | 1 | 0 | $x_1$ запрет по $x_2$;<br>$x_1$, но не $x_2$ | Запрет; отрицание импликации |
| $x_1 \mid x_2$ | — | $x_1 \mid x_2$ | 1 | 1 | 1 | 0 | $x_1$ и $x_2$ несовместны | Логическое И-НЕ; элемент (штрих) Шеффера; отрицание конъюнкции |
| $x_1 \downarrow x_2$ | — | $x_1 \downarrow x_2$ | 1 | 0 | 0 | 0 | ни $x_1$, ни $x_2$ | Логическое ИЛИ-НЕ; стрелка Пирса; функция Вебба; отрицание дизъюнкции |
| $\bar{x}$ | $\neg x$ | $x$ | | 0 | 1 | | не x | Логическое НЕ; инверсия; логическое отрицание |
| | | $\bar{x}$ | | 1 | 0 | | | |

$f_1(x_1, x_2) = x_1 \cdot x_2$, $f_2(x_1, x_2) = x_1 \Delta x_2$,

$f_4(x_1, x_2) = x_2 \Delta x_1$, $f_6(x_1, x_2) = x_1 \oplus x_2$,

$f_7(x_1, x_2) = x_1 \vee x_2$, $f_8(x_1, x_2) = x_1 \downarrow x_2$,

$f_9(x_1, x_2) = x_1 \backsim x_2$, $f_{10}(x_1, x_2) = \overline{x}_2$,

$f_{11}(x_1, x_2) = x_2 \rightarrow x_1$, $f_{12}(x_1, x_2) = \overline{x}_1$,

$f_{13}(x_1, x_2) = x_1 \rightarrow x_2$, $f_{14}(x_1, x_2) = x_1 \mid x_2$.

Остальные из приведенных в табл. 1.2 функций не представляют практического интереса: $f_0(x_1, x_2) = 0$, $f_3(x_1, x_2) = x_1$, $f_5(x_1, x_2) = x_2$, $f_{15}(x_1, x_2) = 1$.

В дальнейшем функции одного и двух аргументов будем называть *элементарными логическими функциями*, имея в виду, что логические выражения этих функций, содержащие не более одной логической операции, элементарны.

Рассмотрим способы построения таблиц истинности для сложных функций многих переменных.

В таблице истинности отображается значение функции для каждого набора (комбинации) значений аргументов. Для представления всей совокупности этих наборов удобно пользоваться последовательностью чисел в так называемой *двоичной системе счисления* (подробнее см. § 2.2). Здесь кратко опишем представление чисел в двоичной системе счисления. В этой системе счисления в разрядах числа используются лишь две цифры: 0 и 1. Веса единиц в отдельных разрядах: 1, 2, 4, 8 и т.д., т.е. вес возрастает в два раза в каждом следующем разряде.(Обратите внимание на отличие от обычной десятичной системы счисления, где веса разрядов равны 1, 10, 100, 1000 и т.д.). Таким образом, запись 1101 в двоичной системе счисления означает следующее количество: $1 \cdot 1 + 0 \cdot 2 + 1 \cdot 4 + 1 \cdot 8 = 13$. В табл. 1.4 приведена последовательность десятичных чисел и соответствующие им представления в двоичной системе счисления в форме четырехразрядных чисел.

*Таблица 1.4*

| Десятичные числа | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Соответствующее представление в двоичной системе счисления | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
| Десятичные числа | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Соответствующее представление в двоичной системе счисления | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

В табл. 1.5 представлена одна из форм таблицы истинности некоторой сложной функции четырех аргументов. При $n$ аргументах число наборов их значений составляет $2^n$ и с ростом $n$ быстро увеличивается число столбцов в таблице. При больших $n$ таблица становится весьма неудобной для использования.

*Таблица 1.5*

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1,x_2,x_3,x_4)$ | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

Для обеспечения большей компактности часто отдают предпочтение другой форме таблицы истинности (показана в табл. 1.6 для функции четырех аргументов). Таблица строится следующим образом. Все аргументы функции делятся на две группы. Столбцам и строкам таблицы приписывают комбинации значений аргументов одной и другой группы. В клетках, расположенных на пересечении столбцов и строк, записываются соответствующие значения функции.

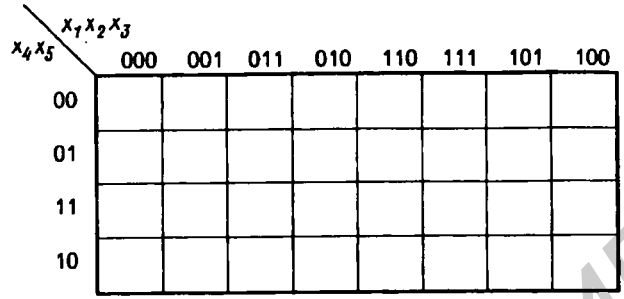
*Таблица 1.6*

| $x_3x_4$ \ $x_1x_2$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 1 |
| 01 | 0 | 1 | 0 | 0 |
| 10 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 |

В дальнейшем при рассмотрении методов минимизации логических функций мы столкнемся с представлением функции в форме таких таблиц истинности, в которых последовательности комбинаций значений аргументов, приписываемых столбцам и строкам таблицы, соответствуют последовательности чисел в так называемом *коде Грея*. Числа в *коде Грея* можно получить из двоичных чисел путем их сложения по модулю 2 (mod 2) с теми же числами, сдвинутыми на один разряд вправо. Например, представление двоичного числа 1101 в *коде Грея* получается следующим образом:

| | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 1 | 0 | 1 | | двоичное число |
| $\oplus$ | | | | | | |
| | | 1 | 1 | 0 | → 1 | сдвинутое вправо число |
| | 1 | 0 | 1 | 1 | | число в *коде Грея* |

*Таблица 1.7*




В табл. 1.7 приведена форма таблицы истинности для функций пяти аргументов. В ней комбинации значений аргументов, приписанные столбцам и строкам таблицы, соответствуют последовательности чисел в *коде Грея*.

### Свойства логических операций конъюнкции, дизъюнкции и инверсии

Конъюнкция переменных $x_1$ и $x_2$ равна *лог.1* в том случае, когда и $x_1$ и $x_2$ равны *лог.1* (отсюда возникло название операции *логическое И*). Дизъюнкция переменных $x_1$ и $x_2$ равна *лог.1*, если или $x_1$ или $x_2$ равна *лог.1* (отсюда название операции *логическое ИЛИ*). В тех случаях, когда число переменных больше двух, конъюнкция их равна *лог.1* при равенстве *лог.1* всех переменных; дизъюнкция равняется *лог.1*, если хотя бы одна из переменных имеет значение *лог.1*.

В табл. 1.8 приведены таблицы истинности для логических операций конъюнкции и дизъюнкции.

В математике установлен определенный порядок выполнения операций в сложном выражении. Например, в выражении $x_1 + x_2x_3$ вначале выполняется операция умножения $x_2x_3$ и затем операция сложения. Если требуется изменить этот порядок, используются скобки. Например, $(x_1 + x_2)x_3$. Здесь вначале выполняется операция в скобках. Подобно этому и для сложного логического выражения установлен определенный порядок выполнения операций: вначале выполняются операции инверсии, затем операции конъюнкции и в последнюю очередь операции дизъюнк-

*Таблица 1.8.*

| Аргументы | | Логические операции | |
|---|---|---|---|
| $x_1$ | $x_2$ | $x_1 \cdot x_2$ | $x_1 \vee x_2$ |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

ции. Например, запись логического выражения $x_1 \vee x_2 \cdot \overline{x}_3 \vee \overline{x}_4 \cdot x_2$ предполагает, что при вычислении выражения вначале выполняются операции инверсии $\overline{x}_3$ и $\overline{x}_4$, затем операции конъюнкции $x_2 \cdot \overline{x}_3$ и $\overline{x}_4 \cdot x_2$ и в последнюю очередь — операции дизъюнкции. Если требуется нарушить это правило, используются скобки. Например, $(x_1 \vee x_2) \cdot (\overline{x} \vee x_4)$. В этом случае вначале выполняются операции в скобках (а если одни скобки вложены в другие, то вначале выполняются операции в самых внутренних скобках).

Операции конъюнкции и дизъюнкции обладают рядом свойств:

$$1 \cdot x = x, \quad x \cdot x = x, \quad 1 \vee x = 1, \quad x \vee x = x, \quad 0 \cdot x = 0,$$

$$x \cdot \overline{x} = 0, \quad 0 \vee x = x, \quad x \vee \overline{x} = 1; \tag{1.1}$$

*сочетательный закон*

$$\begin{cases} x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3, \\ x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3; \end{cases}$$

*переместительный закон*

$$\begin{cases} x_1 \cdot x_2 = x_2 \cdot x_1, \\ x_1 \vee x_2 = x_2 \vee x_1; \end{cases} \tag{1.2}$$

*распределительный закон*

$$\begin{cases} x_1 \cdot (x_2 \vee x_3) = x_1 \cdot x_2 \vee x_1 \cdot x_2, \\ x_1 \vee (x_2 \cdot x_3) = (x_1 \vee x_2) \cdot (x_1 \vee x_3). \end{cases}$$

Выражения (1.2) представляют собой тождества, т.е. равенства, справедливые при любой комбинации значений переменных. Это может быть подтверждено подстановкой в правую и левую части равенств всех возможных комбинаций значений переменных с последующим вычислением логических значений, к которым приводят выражения в обеих частях равенств, и их сравнением. Такая проверка второго выражения распределительного закона показана в табл. 1.9.

Покажем справедливость так называемых *формул де Моргана*:

$$\overline{x_1 \vee x_2} = \overline{x}_1 \cdot \overline{x}_2, \qquad \overline{x_1 \cdot x_2} = \overline{x}_1 \vee \overline{x}_2.$$

В выражении $\overline{x_1 \vee x_2} = \overline{x}_1 \cdot \overline{x}_2$, левая часть обращается *в лог. 1* только в том случае, если $x_1 \vee x_2 = 0$, для чего необходимо, чтобы $x_1 = 0$ и $x_2 = 0$. Правая часть выражения принимает значение *лог. 1* только при $\overline{x}_1 = 1$ и $\overline{x}_2 = 1$, т.е. при $x_1 = 0$ и $x_2 = 0$. Таким образом, только набор $x_1 = 0$ и $x_2 = 0$ обращает в *лог. 1* и левую, и правую части выражения; при остальных

наборах значений переменных левая и правая части выражения будут иметь значение *лог.0*, что и доказывает справедливость рассматриваемого равенства.

*Таблица 1.9*

| Аргументы | | | Результаты вычислений левой части равенства | | Результаты вычислений правой части равенства | | |
|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_2 \cdot x_3$ | $x_1 \vee x_2 \cdot x_3$ | $x_1 \vee x_2$ | $x_1 \vee x_3$ | $(x_1 \vee x_2) \cdot (x_1 \vee x_3)$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

В выражении $\overline{x_1 \cdot x_2} = \bar{x}_1 \vee \bar{x}_2$ и правая, и левая части обращаются в *лог.0* при $x_1 = 1$ и $x_2 = 1$, при остальных наборах значений переменных обе части равны *лог.1*, что и доказывает справедливость данного равенства.

Можно сформулировать следующее правило применения формул де Моргана к сложным логическим выражениям. Инверсия любого сложного выражения, в котором аргументы (либо их инверсии) связаны операциями конъюнкции и дизъюнкции, может быть представлена тем же выражением без инверсии с изменением всех знаков конъюнкции на знаки дизъюнкции, знаков дизъюнкции на знаки конъюнкции и инверсией всех аргументов. Например, $x_1 \vee x_2 \cdot \overline{x_3} \vee \overline{x_1} \cdot x_3 \cdot \overline{x}_4 = \bar{x}_1 \cdot (\bar{x}_2 \vee x_3) \cdot (x_1 \vee \bar{x}_3 \vee x_4)$.

### Выражение элементарных функций через операции И, ИЛИ, НЕ

1. *Операция запрета*

$$x_1 \Delta x_2 = x_1 \cdot \bar{x}_2. \qquad (1.3)$$

Для доказательства этого и последующих выражений будем подставлять в их левую и правую части все возможные наборы значений аргументов и каждый раз проверять справедливость равенств.

| $x_1$ | $x_2$ | $x_1 \Delta x_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| $x_1$ | $x_2$ | $\bar{x}_2$ | $x_1 \cdot \bar{x}_2$ |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

2. *Сумма по модулю 2*

$$x_1 \oplus x_2 = x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2 = (x_1 \vee x_2) \cdot (\bar{x}_1 \vee \bar{x}_2). \quad (1.4)$$

| $x_1$ | $x_2$ | $x_1 \oplus x_2$ | $x_1 \cdot \bar{x}_2$ | $\bar{x}_1 \cdot x_2$ | $x_1 \cdot \bar{x}_2 \vee \vee \bar{x}_1 \cdot x_2$ | $x_1 \vee x_2$ | $\bar{x}_1 \vee \bar{x}_2$ | $(x_1 \vee x_2) \cdot (\bar{x}_1 \vee \bar{x}_2)$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

3. *Операция ИЛИ-НЕ*

$$x_1 \downarrow x_2 = \overline{x_1 \vee x_2}. \quad (1.5)$$

| $x_1$ | $x_2$ | $x_1 \downarrow x_2$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

| $x_1$ | $x_2$ | $x_1 \vee x_2$ | $\overline{(x_1 \vee x_2)}$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |

4. *Логическая равнозначность*

$$x_1 \sim x_2 = \overline{x_1 \oplus x_2} = x_1 \cdot x_2 \vee \bar{x}_1 \cdot \bar{x}_2 = (\bar{x}_1 \vee x_2) \cdot (x_1 \vee \bar{x}_2). \quad (1.6)$$

Справедливость первого равенства может быть установлена непосредственно по таблицам истинности функций логической равнозначности и суммы по модулю 2, а последующих равенств — инвертированием левой и правой частей выражения (1.4) и преобразованием правой части по формулам де Моргана.

5. *Импликация*

$$x_1 \rightarrow x_2 = \bar{x}_1 \vee x_2. \quad (1.7)$$

| $x_1$ | $x_2$ | $x_1 \to x_2$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| $x_1$ | $x_2$ | $\bar{x}_1$ | $\bar{x}_1 \vee x_2$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 |

6. *Операция И-НЕ*

$$x_1 \mid x_2 = \overline{x_1 \cdot x_2}. \tag{1.8}$$

| $x_1$ | $x_2$ | $x_1 \mid x_2$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| $x_1$ | $x_2$ | $x_1 \cdot x_2$ | $(\overline{x_1 \cdot x_2})$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

## Полные системы функций алгебры логики

Элементарные логические функции двух переменных $f_0,\ldots,f_{15}$ могут быть реализованы простейшими логическими элементами. Для реализации сложных логических функций их сначала следует представить элементарными, которые затем последовательно выполнять с помощью простейших логических элементов. Например, функция

$$f(x_1, x_2, x_3, x_4) = (((x_1 \cdot \bar{x}_2) \to x_3) \mid x_4) \vee \bar{x}_1$$

может быть реализована с помощью схемы на рис. 1.4.

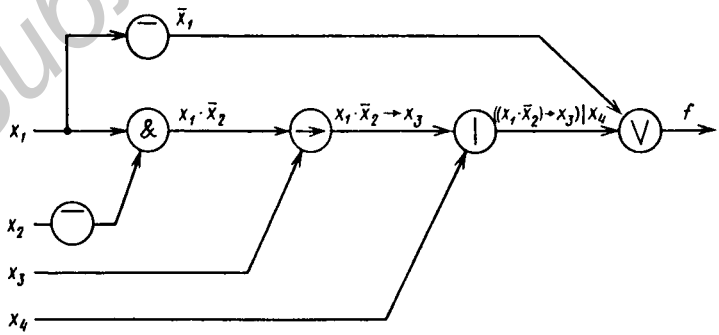



Рис. 1.4

Имея элементы, выполняющие элементарные функции $f_0,...f_{15}$, можно выполнить любую сложную логическую функцию. Такую систему функций можно назвать *полной системой* или *базисом*. Условие наличия 16 различных типов логических элементов, каждый из которых реализует одну из 16 элементарных функций $f_0,...f_{15}$, является достаточным для синтеза логического устройства любой сложности, но это условие не является необходимым, т.е. при синтезе можно ограничиться меньшим набором элементарных функций, взятых из $f_0,...f_{15}$.

Последовательно исключая из базиса функции, можно получить так называемый *минимальный базис*. Под минимальным базисом понимают такой набор функций, исключение из которого любой функции превращает полную систему функций в неполную.

Возможны различные базисы и минимальные базисы, отличающиеся друг от друга числом входящих в них функций и видом этих функций. Выбор того или иного базиса для синтеза логических устройств связан с тем, насколько просто, удобно и экономично выполнить элементы, реализующие входящие в базис функции, и в целом все логическое устройство.

Как показано выше, с помощью логических операций конъюнкции (И), дизъюнкции (ИЛИ) и инверсии (НЕ) можно выразить любую другую из элементарных функций $f_0,...f_{15}$. Следовательно, эта совокупность логических функций образует базис. Это означает, что любая логическая функция, как бы сложна она ни была, может быть представлена через логические операции И, ИЛИ, НЕ. Иначе, можно построить любое логическое устройство, имея лишь три типа логических элементов, выполняющих операции И, ИЛИ, НЕ.

Базис И, ИЛИ, НЕ не является минимальным. Из этой совокупности функций можно исключить функцию И либо функцию ИЛИ, и оставшийся набор функций будет удовлетворять свойствам базиса. Действительно, если исключить функцию И, то операцию И можно выразить через оставшиеся операции ИЛИ и НЕ. Чтобы показать это, дважды инвертируем конъюнкцию и применяем затем правило де Моргана:

$$x_1 \cdot x_2 = \overline{\overline{x_1 \cdot x_2}} = \overline{\overline{x_1} \vee \overline{x_2}}.$$

Хотя операцию И и можно выразить через операции ИЛИ и НЕ, но это сложно (требуется выполнение трех операций НЕ и одной операции ИЛИ), поэтому на практике используется неминимальный базис, включающий функции И, ИЛИ, НЕ.

Рассмотрим некоторые другие базисы. При этом выбранный набор логических функций будет удовлетворять свойствам базиса, если с его помощью можно будет выразить функции И и НЕ (либо функции ИЛИ и НЕ).

1. Базис образует функция И-НЕ. Действительно, операции И и НЕ следующим образом можно выразить через операцию И-НЕ:

$$\bar{x} = \overline{x \cdot x} = x \mid x,$$

$$x_1 \cdot x_2 = \overline{\overline{x_1 \cdot x_2}} = \overline{x_1 \mid x_2} = (x_1 \mid x_2) \mid (x_1 \mid x_2).$$

Таким образом, для построения логического устройства произвольной сложности достаточно иметь элементы, реализующие функцию И-НЕ.

2. Базис образует функция ИЛИ-НЕ. Покажем, что операции НЕ и ИЛИ выражаются через операцию ИЛИ-НЕ:

$$\bar{x} = \overline{x \vee x} = x \downarrow x,$$

$$x_1 \vee x_2 = \overline{\overline{x_1 \vee x_2}} = \overline{x_1 \downarrow x_2} = (x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2).$$

Таким образом, используя однотипные элементы, реализующие операцию ИЛИ-НЕ, можно построить логическое устройство любой сложности.

3. Базис образуют функции запрета $f_4(x_1, x_2) = x_1 \Delta x_2$ и константы единицы $f_{15}(x_1, x_2) = 1$. Действительно,

$$\bar{x} = 1 \cdot \bar{x} = 1 \Delta x,$$

$$x_1 \cdot x_2 = x_1 \cdot \overline{\overline{x_2}} = x_1 \Delta \bar{x}_2 = x_1 \Delta \underbrace{(1 \Delta x_2)}_{\bar{x}_2}.$$

В настоящее время базис И, ИЛИ, НЕ обычно используется при начальной стадии проектирования устройств для построения функциональной схемы. Для реализации устройств чаще всего используются базисы И-НЕ либо ИЛИ-НЕ. Элементы этого базиса широко выпускаются промышленностью в интегральном исполнении.

### *Упражнения*

1. Покажите, что операция ИЛИ представима операциями И и НЕ.

2. Покажите справедливость следующих представлений элементарных функций через операцию И-НЕ:

а) $x_1 \Delta x_2 = \overline{x_1 \mid x_2}$;  б) $x_1 \downarrow x_2 = \overline{\bar{x}_1 \mid \bar{x}_2}$;

в) $x_1 \rightarrow x_2 = x_1 \mid \bar{x}_2$;  г) $x_1 \oplus x_2 = (x_1 \mid \bar{x}_2) \mid (\bar{x}_1 \mid x_2)$;

д) $x_1 \sim x_2 = (x_1 \mid x_2) \mid (\bar{x}_1 \mid x_2)$.

3. Покажите справедливость следующих представлений элементарных функций через операцию ИЛИ-НЕ:

а) $x_1 \Delta x_2 = \bar{x}_1 \downarrow x_2$;  б) $x_1 \mid x_2 = \overline{\bar{x}_1 \downarrow \bar{x}_2}$;

в) $x_1 \rightarrow x_2 = \bar{x}_1 \downarrow x_2$;  г) $x_1 \oplus x_2 = (x_1 \downarrow x_2) \downarrow (\bar{x}_1 \downarrow \bar{x}_2)$;

д) $x_1 \sim x_2 = (x_1 \downarrow \bar{x}_2) \downarrow (\bar{x}_1 \downarrow x_2)$.

4. Пользуясь правилом де Моргана, преобразуйте следующие логические выражения:

а) $\overline{x_1 \cdot x_2 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3}$;

б) $\overline{(x_1 \vee x_2) \cdot (\bar{x}_1 \vee x_2 \vee x_3) \cdot (x_1 \vee \bar{x}_2 \vee \bar{x}_3)}$;

в) $\overline{(x_1 \cdot x_2 \cdot x_3 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot x_3) \cdot (\bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3)}$.

## 1.2. ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ

### Физическое представление логических значений

Выше отмечалось, что логические функции и их аргументы принимают значения *лог.0* и *лог.1*. При этом следует иметь в виду, что в устройствах логическим уровням (*лог.0* и *лог.1*) соответствуют напряжения определенного уровня (или формы). Наиболее часто встречается так называемый *потенциальный способ* представления логических уровней. В этом случае используется напряжение двух уровней (рис. 1.5,*а*,*б*): высокий (по значению модуля) уровень соответствует *лог.1* (*уровень лог.1*), низкий уровень — *лог.0* (*уровень лог.0*). Такой способ представления логических величин называется *положительной логикой*. Относительно редко применяется так называемая *отрицательная логика*, при которой *лог.1* соответствует низкий уровень напряжения, а *лог.0* — высокий уровень. В дальнейшем, если это не оговаривается особо, будем пользоваться только положительной логикой.

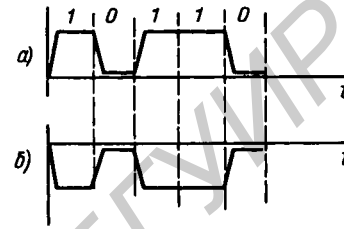
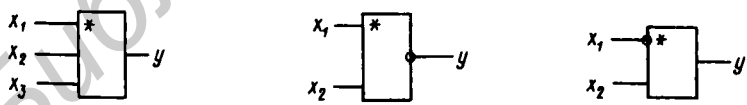



Рис. 1.5

### Обозначения логических элементов в схемах

***Общий принцип обозначения логических элементов***

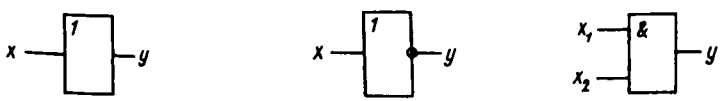
| Логический элемент | Элемент с инверсным выходом | Элемент с инверсным входом |
|---|---|---|
| $y = \varphi^*(x_1, x_2, x_3)$ | $\overline{y = \varphi^*(x_1, x_2)}$ | $y = \varphi^*(\overline{x_1}, x_2)$ |




Здесь * — указатель функции, выполняемой логическим элементом.

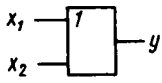
***Обозначение элементов, реализующих логические функции***

| Повторитель | Инвертор (НЕ) | Конъюнктор (И) |
|---|---|---|
| $y = x$ | $y = \overline{x}$ | $y = x_1 \cdot x_2$ |



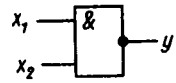
Дизъюнктор (ИЛИ)

$y = x_1 \vee x_2$



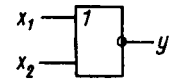

Элемент И-НЕ (элемент Шеффера)

$y = \overline{x_1 \cdot x_2} = x_1 \mid x_2$



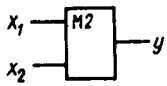

Элемент ИЛИ-НЕ (элемент Пирса)

$y = \overline{x_1 \vee x_2} = x_1 \downarrow x_2$



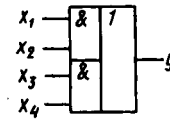

Сложение по модулю 2

$y = x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2 = x_1 \oplus x_2$



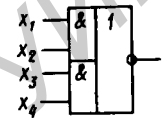

Элемент И-ИЛИ

$y = x_1 \cdot x_2 \vee x_3 \cdot x_4$




Элемент И-ИЛИ-НЕ

$y = \overline{x_1 \cdot x_2 \vee x_3 \cdot x_4}$




**Основные параметры логических элементов**

Логические элементы характеризуются многими параметрами. Среди них следующие параметры могут считаться основными.

*Коэффициент объединения по входу* определяет число входов элемента, предназначенных для подачи логических переменных. Элемент с большим коэффициентом объединения по входу имеет более широкие функциональные возможности.

*Нагрузочная способность* (или *коэффициент разветвления по выходу*) определяет число входов аналогичных элементов, которое может быть подключено к выходу данного элемента. Чем выше нагрузочная способность элементов, тем меньшее число элементов может потребоваться при построении цифрового устройства.

*Быстродействие* логического элемента является одним из важнейших его параметров. Оно оценивается *задержкой распространения сигнала* от входа к выходу элемента. На рис. 1.6 показана идеализированная форма входного и выходного сигналов инвертора; здесь $t_з^{1,0}$ — время задержки переключения выхода элемента из состояния *лог.1* в состояние *лог.0*, $t_з^{0,1}$ — задержка переключения из состояния 0 в состояние 1. Как видно из рисунка, время задержки измеряется на уровне, среднем между уровнями *лог.0* и *лог.1*. *Средняя задержка распространения сигнала* $t_{з\,ср} = 0{,}5\,(t_з^{0,1} + t_з^{1,0})$. Этот параметр используется при расчете задержки распространения сигналов в сложных логических схемах.

*Помехоустойчивость* определяется максимальным значением помехи, не вызывающей нарушения работы элемента. Для количественной оценки помехоустойчивости воспользуемся так называемой *передаточ-*

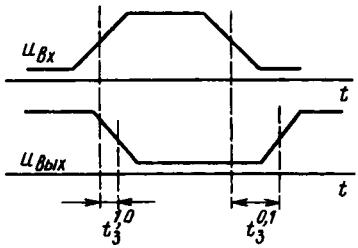



Рис. 1.6

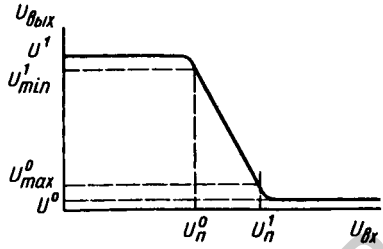



Рис. 1.7

*ной характеристикой* логического элемента (инвертора). На рис. 1.7 приведена типичная идеализированная форма этой характеристики.

*Передаточная характеристика* представляет собой зависимость выходного напряжения от входного. Для ее получения необходимо соединить все входы логического элемента и, изменяя напряжение на входе, отмечать соответствующие значения напряжения на выходе. При увеличении входного напряжения от нуля до порогового $U^0_п$ в области значений, соответствующих уровню *лог.0*, напряжение на выходе уменьшается от уровня *лог.1* до некоторого минимально допустимого $U^1_{min}$ в области значений, соответствующих уровню *лог.1*. Дальнейшее увеличение входного напряжения приводит к резкому снижению выходного напряжения. При больших значениях входного напряжения, превышающих пороговый уровень *лог.1* ($U^1_п$), на выходе устанавливается напряжение, не превышающее максимально допустимого уровня *лог.0* ($U^0_{max}$). Таким образом, при нормальной работе элемента в статическом (установившемся) режиме недопустимы входные напряжения $U^0_п < u_{вх} < U^1_п$. Допустимыми считаются такие помехи, которые, наложившись на входное напряжение, не выведут его в область недопустимых значений.

### Диодный элемент ИЛИ (сборка)

На рис. 1.8,*а* приведена схема элемента для работы с положительными напряжениями. При отрицательных напряжениях полярность включения диодов необходимо изменить, как показано на рис. 1.8,*б*.

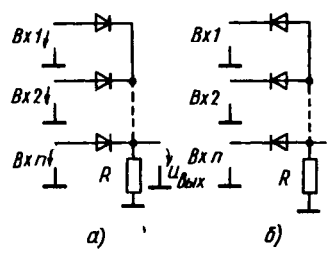



Рис. 1.8

Опишем работу схемы на рис. 1.8,*а* при действии на одном из входов (например, Вх1) напряжения показанной на рис. 1.9,*б* формы, а на остальных входах — напряжения уровня *лог.0*. В момент $t_1$ при возрастании напряжения на Вх1 от уровня *лог.0* до

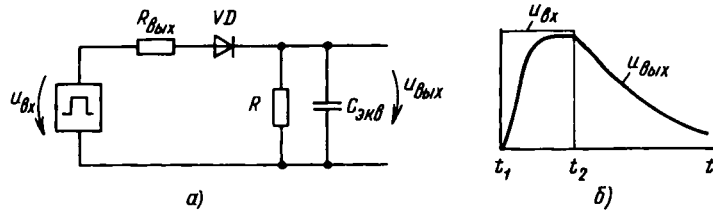



Рис. 1.9

уровня *лог.1* откроется диод и положительное напряжение со входа через открытый диод передастся на резистор R, т.е. на выход. Это напряжение, возникающее на R, для всех других диодов является отрицательным, и эти диоды остаются в закрытом состоянии. Таким образом, на выходе элемента образуется сигнал, соответствующий *лог.1*, если хотя бы на одном из входов действует *лог.1*. Следовательно, элемент реализует операцию дизъюнкции (операцию ИЛИ). Работу схемы рис. 1.8,*б* при входных отрицательных напряжениях предлагается рассмотреть самостоятельно.

Рассмотрим факторы, определяющие скорости нарастания и спада напряжения на выходе элемента ИЛИ. Будем считать, что к выходу (параллельно R) подключен некоторый эквивалентный емкостной элемент, емкость $C_{экв}$ которого включает емкость нагрузки, монтажа и закрытых диодов. Для процессов с момента $t_1$ действия на входе напряжения уровня *лог.1* образуется эквивалентная схема, представленная на рис. 1.9,*а*. Сопротивление, включающее сопротивление открытого диода VD и выходное сопротивление элемента, с которого напряжение поступает на Вх.1, обозначим *r*. Очевидно, из-за наличия емкости $C_{экв}$ напряжение на элементе не может измениться скачком, оно растет по экспоненциальному закону с постоянной времени

$$\tau_{нар} = C_{экв} rR / (r + R) \approx rC_{экв}$$

(так как $r << R$), стремясь к значению $u_{вх} r / (r + R) \approx u_{вх}$.

В момент $t_2$ при переходе напряжения на входе от уровня *лог.1* к уровню *лог.0* закрывается и диод VD. При этом напряжение на выходе снижается по мере разряда емкости $C_{экв}$ через резистор *R* с постоянной времени $\tau_{сп} = RC_{экв}$. Так как $\tau_{сп} >> \tau_{нар}$, скорость спада выходного напряжения оказывается меньше скорости нарастания этого напряжения.

Очередная подача на вход напряжения уровня *лог.1* возможна лишь после того, как остаточное напряжение на выходе снизится до определенного малого значения. Поэтому медленный спад выходного напря-

жения вызывает необходимость увеличения тактового интервала и, следовательно, является причиной снижения быстродействия элемента.

### Диодный элемент И (схема совпадения)

На рис. 1.10,*а* приведена схема элемента И, используемая при положительных входных напряжениях. В случае отрицательных входных напряжений необходимо изменить полярность напряжения источника питания и полярность включения диодов, как показано на рис. 1.10,*б*.

Пусть на одном из входов схемы на рис. 1.10,*а* действует низкий уровень напряжения, соответствующий уровню *лог.0*. Ток будет замыкаться в цепи от источника напряжения $E$ через резистор R, открытый диод и источник низкого входного напряжения. Так как сопротивление открытого диода мало, то низкий потенциал со входа через открытый диод будет передаваться на выход. Диоды, подключенные к остальным входам, на которых действует высокий уровень напряжения, оказываются закрытыми. Напряжение на диоде можно определить суммированием напряжений при обходе внешней по отношению к диоду цепи от его анода к катоду: $U_д = u_{вых} - u_{вх}$. Таким образом, выходное напряжение, прикладываемое к анодам диодов, является для них положительным, стремящимся открыть диоды; входное напряжение, прикладываемое к катоду, — отрицательным, стремящимся закрыть диод. И если $u_{вых} = u_{вх}$, то $U_д$ отрицательно и диод закрыт. Именно поэтому, когда на выходе элемента низкий потенциал (*лог.0*), а на входе высокий потенциал (*лог.1*), подключенный к этому входу диод оказывается закрытым. Таким образом, если хотя бы на одном из входов действует напряжение низкого уровня (*лог.0*), то на выходе элемента образуется напряжение низкого уровня (*лог.0*).

Пусть на всех входах эквивалентной схемы, приведенной на рис. 1.11,*а*, действуют напряжения высокого уровня (*лог.1*). Они могут не-

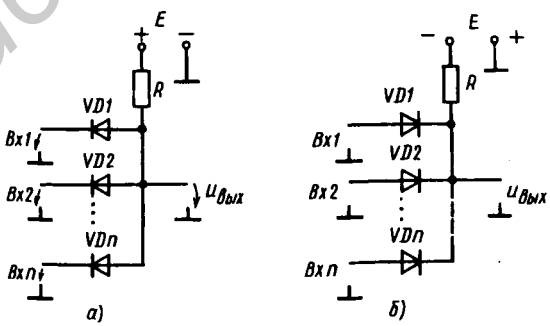



Рис. 1.10

сколько отличаться по значению. При этом будет открыт тот диод, который подключен ко входу с более низким напряжением. Это напряжение через диод будет передаваться на выход.

Остальные диоды будут практически закрыты. На выходе установится напряжение высокого уровня (*лог.1*).Следовательно, на выходе элемента устанавливается напряжение уровня *лог.1* в том, и только в том случае, когда на всех входах действует напряжение уровня *лог.1*. Таким образом элемент выполняет логическую операцию И.

Рассмотрим форму выходного напряжения (рис. 1.11,*б*). Будем считать, что к выходу элемента подключен некоторый эквивалентный емкостной элемент, емкость которого $C_{экв}$ включает емкости нагрузки, монтажа и закрытых диодов. В момент подачи напряжения уровня *лог.1* одновременно на все входы напряжение на емкостном элементе (на выходе элемента И) не может возрасти скачком. Все диоды вначале оказываются закрытыми входными напряжениями, являющимися для диодов отрицательными. Поэтому источники входных напряжений будут отключены от емкостного элемента. Он заряжается от источника напряжения $E$ через резистор R. Напряжение на емкостном элементе (а значит, и на выходе элемента) растет по экспоненциальному закону с постоянной времени $\tau = RC_{экв}$ (рис. 1.11,*б*). В момент времени, когда $u_{вых}$ превысит минимальное из входных напряжений, откроется соответствующий диод, и рост $u_{вых}$ прекратится. Ток от источника напряжения $E$, ранее замыкавшийся через $C_{экв}$, переключается в цепь открытого диода.

В момент перехода от напряжения уровня *лог.1* к напряжению уровня *лог.0* хотя бы на одном из входов соответствующие диоды открываются положительным для них напряжением $u_{вых}$. Происходит относительно быстрый разряд $C_{экв}$ через открытые диоды и малое сопротивление источников входных напряжений. Напряжение на выходе снижается по экспоненциальному закону с малой постоянной времени $\tau_{сп} = rC_{экв}$.

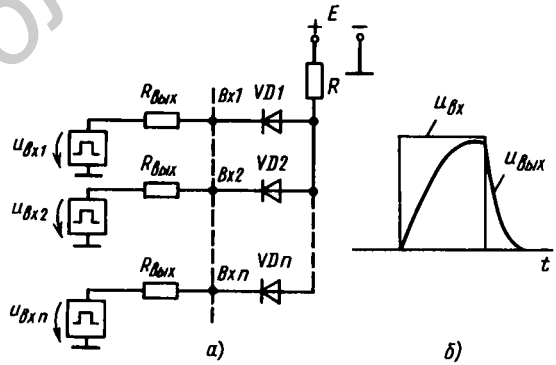



Рис. 1.11

Сравнение формы выходных напряжений диодных элементов ИЛИ и И показывает, что в элементе ИЛИ оказывается более растянутым срез выходного напряжения, в элементе И — его фронт.

### Транзисторный элемент НЕ (инвертор)

Операция НЕ может быть реализована ключевым элементом, представленным на рис. 1.12,*а*. При низком уровне входного напряжения, соответствующем *лог.0*, транзистор закрыт, на его выходе устанавливается напряжение высокого уровня *E* (*лог.1*). И наоборот, при высоком уровне входного напряжения (*лог.1*) транзистор насыщен, на его выходе устанавливается напряжение, близкое к нулю (*лог.0*). Графики входных и выходных напряжений представлены на рис. 1.12,*б*.

### Логический элемент транзисторно-транзисторной логики (ТТЛ)

На рис. 1.13 приведена основная схема элемента. Она состоит из двух последовательно включенных функциональных частей: схемы, выполняющей операцию И, и схемы инвертора, выполняющего операцию НЕ. Особенность элемента ТТЛ заключается в том, что в обеих его частях применены транзисторы (отсюда и название элемента — транзисторно-транзисторная логика). Для реализации операции И используется многоэмиттерный транзистор.

Рассмотрим работу этой части схемы. Пусть на все входы элемента подано напряжение уровня *лог.1* (3,2 В). Возможное при этом распределение потенциалов в отдельных точках схемы показано на рис. 1.14,*а*. В данном случае многоэмиттерный транзистор оказывается в так называемом инверсном включении (эмиттеры выполняют роль коллекторов, коллектор — роль эмиттера). Потенциал базы оказывается выше потенциала коллектора. Переход база—коллектор открыт, и ток базы проходит через переход в коллектор, замыкаясь далее через эмиттерный переход транзистора VT. Потенциал эмиттеров выше потенциала базы, и переход эмиттеры—база закрыт. Многоэмиттерный транзистор вы-

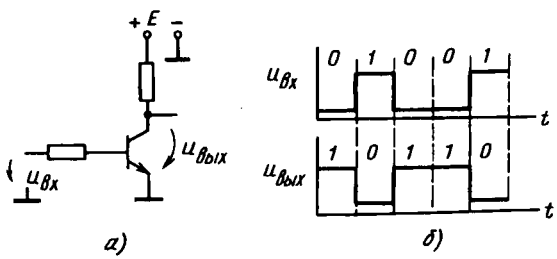



Рис. 1.12

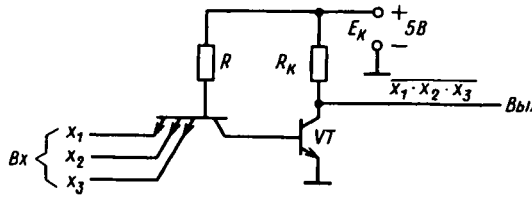



Рис. 1.13

полняется таким образом, чтобы в этом активном режиме с инверсным включением электродов коэффициент усиления по току был бы много меньше единицы, т.е. ток эмиттеров (выполняющих роль коллекторов) был во много раз меньше тока базы. Обратите внимание на направление тока на входах элемента (ток втекает в элемент).

Втекающий в базу транзистора VT ток удерживает транзистор в открытом состоянии. Коллекторный ток транзистора VT замыкается через $R_к$, практически все напряжение источника питания (5 В) падает на $R_к$, на выходе низкое напряжение (0,4 В), соответствующее уровню *лог.0*.

Рассмотрим другое состояние схемы. Пусть хотя бы на одном из входов действует напряжение уровня *лог.0*. Возникающее при этом распределение потенциалов показано на рис. 1.14,*б*. Потенциал базы многоэмиттерного транзистора выше потенциала змиттера и коллектора. Следовательно, оба перехода, эмиттерный и коллекторный, смещены в прямом направлении и многоэмиттерный транзистор находится в режиме насыщения. Весь базовый ток многоэмиттерного транзистора замыкается через эмиттерные переходы. Напряжение между коллектором и эмиттером близко к нулю, и действующий на эмиттере низкий уровень напряжения (*лог.0*) через многоэмиттерный транзистор передается на базу транзистора VT. Транзистор VT закрыт, на его коллекторе высокий уровень напряжения (*лог.1*). При этом практически весь базовый ток многоэмиттерного транзистора замыкается через смещенный в

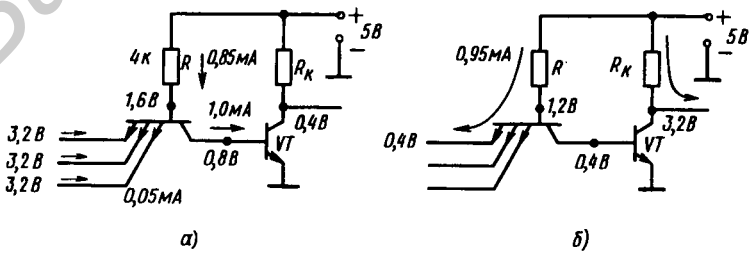



Рис. 1.14

прямом направлении эмиттерный переход многоэмиттерного транзистора.

**Повышение нагрузочной способности элемента.** Улучшение этого параметра достигается применением усложненной схемы инверторной части. Схема элемента с одним из вариантов сложного инвертора приведена на рис. 1.15.

Рисунок 1.15,*а* иллюстрирует режим включенного элемента. Если на всех входах действует напряжение уровня *лог.1*, весь текущий через резистор R1 ток подается в базу транзистора VT2. Транзистор VT2 открывается и переходит в режим насыщения. Эмиттерный ток транзистора VT2 втекает в базу транзистора VT5, удерживая этот транзистор в открытом состоянии. Транзисторы VT3 и VT4 закрываются, так как на эмиттерном переходе каждого из них действует напряжение 0,3 В, недостаточное для открывания транзисторов.

На рис. 1.15,*б* показан режим выключенного элемента. Если хотя бы на одном из входов действует напряжение уровня *лог.0*, ток резистора R1 полностью переключается во входную цепь. Транзисторы VT2, VT5 закрываются, на выходе элемента устанавливается напряжение уровня *лог.1*. Транзисторы VT3, VT4 работают в двух последовательно включенных эмиттерных повторителях, на вход которых подается ток через резистор R2, а эмиттерный ток транзистора VT4 питает нагрузку.

Повышение нагрузочной способности включенного элемента связано с тем, что выходной транзистор VT5, через который замыкается ток нагрузки, удерживается в открытом состоянии большим базовым током, который обеспечивается эмиттерной цепью транзистора VT2.

В выключенном состоянии элемента с простым инвертором ток в нагрузку подается от источника питания через коллекторный резистор $R_к$ с большим сопротивлением (см. рис.1.14,*б*). Этот резистор ограничивает максимальное значение тока в нагрузке (с ростом тока нагрузки

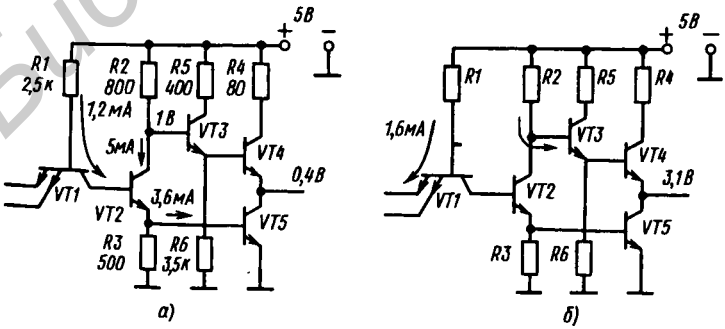



Рис. 1.15

увеличивается падение напряжения на $R_к$, уменьшается напряжение на выходе элемента). В элементе со сложным инвертором в нагрузку подается эмиттерный ток транзистора VT4, работающего в схеме эмиттерного повторителя. Так как выходное сопротивление эмиттерного повторителя мало, то выходное напряжение элемента слабее зависит от тока нагрузки и допустимы большие значения нагрузочного тока.

**Быстродействие логического элемента и методы его повышения.** Быстродействие элемента определяется временем переключения транзисторов в схеме элемента и временем, необходимым для перезаряда емкости нагрузки и паразитных монтажных емкостей $C_{экв}$.

Для уменьшения времени переключения транзисторов в элементе необходимо использовать более высокочастотные транзисторы и переключение транзисторов производить бо́льшими управляющими токами в цепи базы; существенное уменьшение времени переключения транзисторов достигается использованием ненасыщенного режима работы транзисторов (в этом режиме исключается время, необходимое на рассасывание неосновных носителей в базе при выключении транзисторов).

Рассмотрим пути увеличения быстродействия элемента за счет ускорения перезаряда емкости $C_{экв}$. На рис. 1.16,*а* приведена эквивалентная схема элемента, а на рис. 1.16,*б* показан процесс заряда емкости $C_{экв}$ при выключении транзистора. Этот процесс можно ускорить следующими приемами.

1. Уменьшение $R$ (и, следовательно, уменьшение постоянной времени $\tau = RC_{экв}$). Однако непосредственное уменьшение $R$ приводит к росту потребляемого от источника питания тока и мощности. Другой прием уменьшения постоянной времени $\tau$ — применение на выходе элемента эмиттерного повторителя, имеющего малое выходное сопротивление (см. выше схему сложного инвертора).

2. Использование в элементах малых перепадов напряжения. Далее этот прием рассмотрен для повышения быстродействия логического элемента эмиттерно-связанной логики.

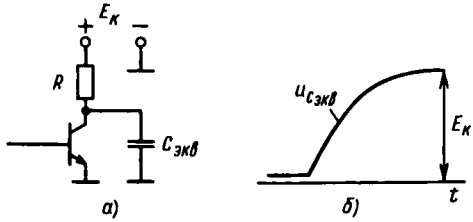



Рис. 1.16

### Логический элемент эмиттерно-связанной логики

Типовая схема элемента эмиттерно-связанной логики (ЭСЛ) приведена на рис. 1.17. Транзисторы VT0, VT1, VT2, VT3 работают в схеме переключателя тока, транзисторы VT4, VT5 — в выходных эмиттерных повторителях. На схеме показаны значения потенциалов в различных точках при подаче на вход напряжения уровня *лог.1*; в скобки заключены значения потенциалов тех же точек для случая, когда на все входы элемента подано напряжение уровня *лог.0*. Значения этих потенциалов соответствуют следующим уровням:

напряжение питания $E_к = 5$ В;

уровень *лог.1* $U^1 = 4{,}3$ В;

уровень *лог.0* $U^0 = 3{,}5$ В;

напряжение между базой и эмиттером открытого транзистора $U_{БЭ} = 0{,}7$ В;

напряжение на базе транзистора VT0 $U = 0{,}5\,(U^1 + U^0) = 3{,}9$ В.

Рассмотрим принцип работы логического элемента ЭСЛ. Пусть на Вх1 подается напряжение $U^1 = 4{,}3$ В. Транзистор открыт, эмиттерный ток этого транзистора создает на резисторе R падение напряжения $U_a = U^1 - U_{БЭ} = 4{,}3 - 0{,}7 = 3{,}6$ В; коллекторный ток создает на резисторе $R_{к1}$ напряжение $U_{Rк1} = 0{,}8$ В; напряжение на коллекторе транзистора $U_б = E_к - U_{Rк1} = 5 - 0{,}8 = 4{,}2$ В.

Между базой и эмиттером транзистора VT0 напряжение $U_{БЭ\ VT0} = U - U_a = 3{,}9 - 3{,}6 = 0{,}3$ В; этого напряжения недостаточно для открывания транзистора VT0. Таким образом, открытое состояние любого из транзисторов VT1, VT2, VT3 приводит к закрытому состоянию транзистора VT0. Ток через резистор $R_{к2}$ весьма мал (течет лишь базовый ток транзистора VT5), и на коллекторе VT0 напряжение $U_в \approx E_к = 5$ В.

Рассмотрим другое состояние логического элемента. Пусть на всех входах действует напряжение уровня *лог.0* $U^0 = 3{,}5$ В. При этом открыт транзистор VT0 (из всех транзисторов, эмиттеры которых объединены,

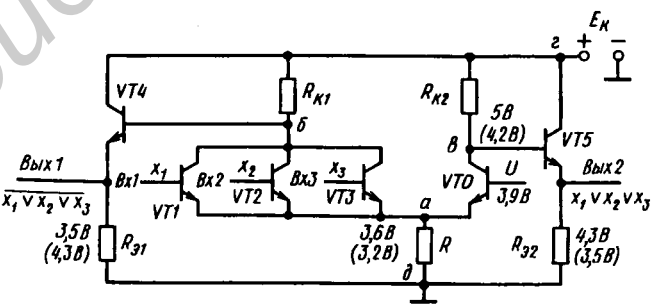



Рис. 1.17

открывается тот, на базе которого действует более высокое напряжение); $U_a = U - U_{БЭ} = 3,9 - 3,2 = 0,7$ В; между базой и эмиттером транзисторов VT1, VT2, VT3 напряжение $U_{БЭ\ VT1 \cdots VT3} = U^0 - U_a = = 3,5 - 3,2 = 0,3$ В, и эти транзисторы закрыты; $U_б = 5$ В; $U_в = 4,2$ В.

Напряжения от точек *б* и *в* передаются на выходы элемента через эмиттерные повторители, при этом уровень напряжения снижается на значение $U_{БЭ} = 0,7$ В. Обратим внимание на важное обстоятельство — напряжения на выходах равны либо $U^1$ (4,3 В), либо $U^0$ (3,5 В).

Выясним, какая логическая функция формируется на выходах элемента. В точке *в* и на Вых2 образуется напряжение низкого уровня при открытом транзисторе VT0, т.е. в случае, когда $x_1 = 0, x_2 = 0, x_3 = 0$. При любой другой комбинации значений входных переменных транзистор VT0 закрыт, и на Вых2 образуется напряжение высокого уровня. Из этого следует, что на Вых2 формируется дизъюнкция переменных $x_1 \vee x_2 \vee x_3$. На Вых1 формируется функция ИЛИ-НЕ $x_1 \vee x_2 \vee x_3 = = x_1 \downarrow x_2 \downarrow x_3$. Следовательно, логический элемент выполняет операции ИЛИ-НЕ и ИЛИ.

В микросхемах ЭСЛ точку *г* делают общей, а точку *д* подключают к источнику питания с напряжением –5 В. В этом случае потенциалы всех точек схемы снижаются на 5 В.

Рассмотренный логический элемент относится к классу наиболее быстродействующих элементов, характеризуемых одновременно относительно высокой потребляемой мощностью.

Высокое быстродействие (малое время задержки распространения сигнала) обеспечивается следующими факторами: открытые транзисторы находятся в активном режиме (не в режиме насыщения); применение на выходах эмиттерных повторителей обеспечивает ускорение процесса перезарядки емкостей, подключенных к выходам; транзисторы включены по схеме, близкой к схеме включения с общей базой, что улучшает частотные свойства транзисторов и ускоряет процесс их переключения; выбран малым перепад логических уровней $U^1 - U^0 = 0,8$ В (однако это приводит к сравнительно низкой помехоустойчивости элемента).

### Логические элементы на МОП-транзисторах

На рис. 1.18,*а* показана схема логического элемента И-НЕ на однотипных МОП-транзисторах с индуцированным каналом типа *n* (так называемая *n* МОП-технология ) . Основные транзисторы VT1 и VT2 включены последовательно, транзистор VT3 выполняет роль нагрузки. В случае, когда на обоих входах элемента действует высокое напряжение $U^1$ ($x_1 = 1, x_2 = 1$), оба транзистора VT1 и VT2 оказываются открытыми и на выходе устанавливается низкое напряжение $U^0$. Во всех остальных случаях хотя бы один из транзисторов VT1 или VT2 закрыт

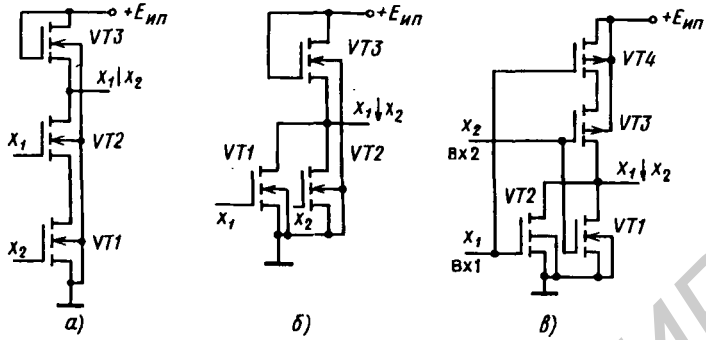



Рис. 1.18

и на выходе устанавливается напряжение $U^1$. Таким образом, элемент выполняет логическую функцию И-НЕ.

На рис. 1.18,*б* приведена схема элемента ИЛИ-НЕ. На его выходе устанавливается низкое напряжение $U^0$, если хотя бы на одном из входов действует высокое напряжение $U^1$ открывающее один из основных транзисторов VT1 или VT2.

На рис. 1.18, *в* приведена схема элемента ИЛИ-НЕ КМОП-технологии. В ней транзисторы VT1 и VT2 — основные, транзисторы VT3 и VT4 — нагрузочные. Пусть на одном из входов этого элемента (например, на Вх1) действует высокое напряжение $U^1$. При этом транзистор VT2 открыт, транзистор VT1 закрыт и независимо от уровня напряжения на другом входе и состояния остальных транзисторов на выходе устанавливается низкое напряжение $U^0$. Элемент реализует логическую операцию ИЛИ-НЕ.

КМОП-схема характеризуется весьма малым потребляемым током (а следовательно, и мощностью) от источника питания.

### Логические элементы интегральной инжекционной логики

На рис. 1.19 показана топология логического элемента интегральной инжекционной логики ($И^2Л$). Для создания такой структуры требуются две фазы диффузии в кремний с проводимостью *n*-типа: в процессе первой фазы образуются области $p_1$ и $p_2$, второй фазы — области $n_2$.

Элемент имеет структуру $p_1$-$n_1$-$p_2$-$n_2$. Такую четырехслойную структуру удобно рассматривать, представив ее соединением двух обычных трехслойных транзисторных структур:

$p_1$-$n_1$-$p_2$

$n_1$-$p_2$-$n_2$.

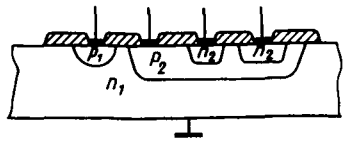



Рис. 1.19

Соответствующая схема показана на рис. 1.20, *а*. Транзистор VT2 со структурой $n_1$-$p_2$-$n_2$ выполняет функции инвертора, имеющего несколько выходов (каждый коллектор образует отдельный выход элемента по схеме с открытым коллектором). Транзистор VT1, называемый *инжектором*, имеет структуру типа $p_1$-$n_1$-$p_2$. Так как область $n_1$ у этих транзисторов общая, эмиттер транзистора VT2 должен быть соединен с базой транзистора VT1; наличие общей области $p_2$ приводит к необходимости соединения базы транзистора VT2 с коллектором транзистора VT1. Так образуется соединение транзисторов VT1 и VT2.

Так как на эмиттере транзистора VT1 действует положительный потенциал, а база находится под нулевым потенциалом, эмиттерный переход оказывается смещенным в прямом направлении и транзистор открыт. Коллекторный ток этого транзистора может замкнуться либо через транзистор VT3 (коллектор предыдущего элемента), либо через эмиттерный переход транзистора VT2.

Если предыдущий логический элемент находится в открытом состоянии (открыт транзистор VT3), то на входе данного элемента наблюдается низкий уровень напряжения, который, действуя на базе VT2, удерживает этот транзистор в закрытом состоянии. Ток инжектора VT1 замыкается через транзистор VT2 . При закрытом состоянии предыдущего логического элемента (закрыт транзистор VT3) коллекторный ток инжектора VT1 втекает в базу транзистора VT2, и этот транзистор устанавливается в открытое состояние.

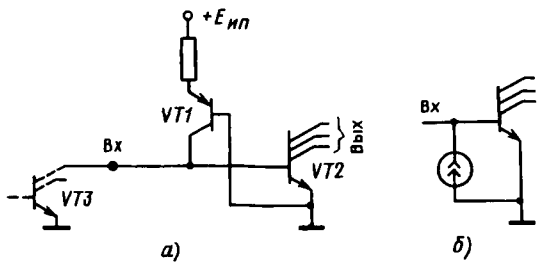



Рис. 1.20

Таким образом, при закрытом VT3 транзистор VT2 открыт и, наоборот, при открытом VT3 транзистор VT2 закрыт. Открытое состояние элемента соответствует состоянию 0, закрытое — состоянию 1.

Инжектор является источником постоянного тока (который может быть общим для группы элементов). Часто пользуются условным графическим представлением элемента, показанным на рис. 1.20,*б*.

На рис. 1.21,*а* приведена схема, реализующая операцию ИЛИ-НЕ. Соединение коллекторов элементов соответствует выполнению операции так называемого *монтажного* И. Действительно, если хотя бы один из элементов находится в открытом состоянии (состоянии 0), то ток инжектора следующего элемента замыкается через открытый инвертор и на объединенном выходе элементов устанавливается низкий уровень *лог.0*. Следовательно, на этом выходе формируется величина, соответствующая логическому выражению $\bar{x}_1 \cdot \bar{x}_2$. Применение к этому выражению преобразования де Моргана приводит к выражению $\bar{x}_1 \cdot \bar{x}_2 = \overline{x_1 \vee x_2}$. Таким образом, данное соединение элементов реализует операцию ИЛИ-НЕ.

Логические элементы И$^2$Л имеют следующие достоинства:

обеспечивают высокую степень интеграции; при изготовлении схем И$^2$Л используются те же технологические процессы, что и при производстве интегральных схем на биполярных транзисторах, но число технологических операций и необходимых фотошаблонов при этом меньше;

используют пониженное напряжение ( $\approx 1$ В);

обеспечивают "обмен" в широких пределах мощности на быстродействие (можно изменять на несколько порядков потребляемую мощность, что соответственно приводит к изменению быстродействия);

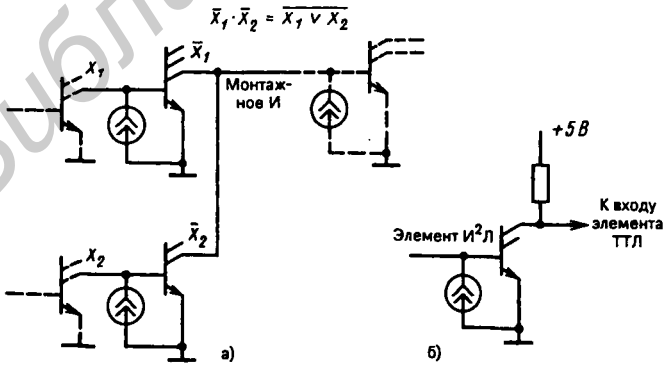



Рис. 1.21

хорошо согласуются с элементами ТТЛ.

На рис. 1.21,*б* показана схема перехода от элемента $И^2Л$ к элементу ТТЛ.

## 1.3. СИНТЕЗ КОМБИНАЦИОННЫХ УСТРОЙСТВ

### Канонические формы представления логических функций

Синтез логического устройства распадается на несколько этапов. На первом этапе функцию, заданную в словесной, табличной или других формах требуется представить в виде логического выражения с использованием некоторого базиса. Дальнейшие этапы сводятся к получению минимальных форм функций, обеспечивающих при синтезе наименьшее количество электронного оборудования и рациональное построение функциональной схемы устройства. Для первого этапа обычно используется базис И, ИЛИ, НЕ независимо от базиса, который будет использован для построения логического устройства.

Для удобства последующих преобразований приняты следующие две исходные канонические формы представления функций: совершенная дизъюнктивная нормальная форма (СДНФ) и совершенная конъюнктивная нормальная форма (СКНФ).

**Совершенная дизъюнктивная нормальная форма** (СДНФ). Дизъюнктивной нормальной формой (ДНФ) называется такая форма представления функции, при которой логическое выражение функции строится в виде дизъюнкции ряда членов, каждый из которых является простой конъюнкцией аргументов или их инверсий. Примером ДНФ может служить выражение

$$f(x_1, x_2, x_3) = x_1 \vee x_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \vee x_2 \cdot x_3. \qquad (1.9)$$

Приведем форму представления функции, не являющуюся ДНФ. Например, функция

$$f(x_1, x_2, x_3) = x_1 \vee x_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \vee \overline{x_2 \cdot x_3}$$

представлена не в ДНФ, так как последний член не является простой конъюнкцией аргументов. Также не является ДНФ следующая форма представления функции:

$$f(x_1, x_2, x_3) = x_1 \cdot (x_2 \cdot \bar{x}_3 \vee \bar{x}_2 \cdot x_3) \vee x_2 \cdot x_3.$$

Если в каждом члене ДНФ представлены все аргументы (или их инверсии) функции, то такая форма называется СДНФ. Выражение (1.9) не является СДНФ, так как в нем лишь третий член содержит все аргументы функции.

Для перехода от ДНФ к СДНФ необходимо в каждый из членов, в которых представлены не все аргументы, ввести выражение вида $x_i \vee \bar{x}_i$, где $x_i$ — отсутствующий в члене аргумент. Так как $x_i \vee \bar{x}_i = 1$, такая операция не может изменить значений функции. Покажем переход от ДНФ к СДНФ на примере следующего выражения:

$$f(x_1, x_2, x_3) = x_1 \vee x_2 \cdot \bar{x}_3.$$

Добавление в члены выражений вида $x_i \vee \bar{x}_i$ приведет к функции

$$f(x_1, x_2, x_3) = x_1 \cdot (x_2 \vee \bar{x}_2) \cdot (x_3 \vee \bar{x}_3) \vee x_2 \cdot \bar{x}_3 \cdot (x_1 \vee \bar{x}_1) =$$

$$= x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \bar{x}_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee x_1 \cdot x_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot x_2 \cdot \bar{x}_3.$$

На основании (1.1)

$$x_1 \cdot x_2 \cdot \bar{x}_3 \vee x_1 \cdot x_2 \cdot \bar{x}_3 = x_1 \cdot x_2 \cdot \bar{x}_3.$$

Отсюда после приведения подобных членов

$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \bar{x}_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee$$

$$\vee \bar{x}_1 \cdot x_2 \cdot \bar{x}_3,$$

т.е. имеем СДНФ. Если исходная функция задана в табличной форме, то СДНФ может быть получена непосредственно.

*Таблица 1.10*

| $x_1$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|
| $x_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_3$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3)$ | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |

Пусть задана функция в форме табл. 1.10. Для этой функции СДНФ имеет вид

$$f(x_1, x_2, x_3) = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot x_3. \qquad (1.10)$$

Каждый член в (1.10) соответствует некоторому набору значений аргументов, при котором $f(x_1, x_2, x_3)$ равна 1. Каждый из наборов аргументов, при которых $f(x_1, x_2, x_3)$ равна 1 (3-, 4-, 6-, 8-й столбцы наборов), обращает в единицу соответствующий член выражения (1.10), вследствие чего и вся функция оказывается равной единице.

Можно сформулировать следующее правило записи СДНФ функции, заданной таблицей истинности. Необходимо записать столько членов в виде конъюнкций всех аргументов, сколько единиц содержит функция в таблице. Каждая конъюнкция должна соответствовать

определенному набору значений аргументов, обращающему функцию в единицу, и если в этом наборе значение аргумента равно нулю, то в конъюнкцию входит инверсия данного аргумента.

Следует отметить, что любая функция имеет единственную СДНФ.

**Совершенная конъюнктивная нормальная форма** (СКНФ). Конъюнктивной нормальной формой (КНФ) называется форма представления функции в виде конъюнкции ряда членов, каждый из которых является простой дизъюнкцией аргументов (или их инверсий).

Примером КНФ может служить следующая форма представления функции:

$$f(x_1, x_2, x_3) = x_1 \cdot (x_2 \vee \bar{x}_3) \cdot (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \cdot (x_2 \vee x_3).$$

Приведем формы представления функций, не являющиеся КНФ:

$$f(x_1, x_2, x_3) = x_1 \cdot (x_2 \vee \bar{x}_3) \cdot (\overline{\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3}) \cdot (x_2 \vee x_3)$$

(здесь третий член не является простой дизъюнкцией аргументов или их инверсий);

$$f(x_1, x_2, x_3) = x_1 \vee (x_2 \vee \bar{x}_3) \cdot (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \cdot (x_2 \vee x_3)$$

( эта форма также не является КНФ, так как в ней первый член не связан с остальными операцией конъюнкции).

В СКНФ в каждом члене КНФ должны быть представлены все аргументы. Для перехода от КНФ к СКНФ необходимо добавить к каждому члену, не содержащему всех аргументов, члены вида $x_i \cdot \bar{x}_i$, где $x_i$ — аргумент, не представленный в члене. Так как $x_i \cdot \bar{x}_i = 0$, то такая операция не может повлиять на значение функции. Добавление $x_i \cdot \bar{x}_i$ к некоторому члену $Y$ образует выражение вида $Y \vee x_i \cdot \bar{x}_i$, которое можно привести к виду

$$Y = Y \vee x_i \cdot \bar{x}_i = (Y \vee x_i) \cdot (Y \vee \bar{x}_i).$$

Справедливость данного равенства вытекает из распределительного закона, она может быть показана также путем раскрытия скобок в правой части выражения

$$(Y \vee x_i) \cdot (Y \vee \bar{x}_i) = Y \cdot Y \vee Y \cdot \bar{x}_i \vee Y \cdot x_i \vee x_i \cdot \bar{x}_i = Y,$$

так как $Y \cdot Y = Y$, $Y \cdot \bar{x}_i \vee Y \cdot x_i = Y \cdot (\bar{x}_i \vee x_i) = Y$, $x_i \cdot \bar{x}_i = 0$.

На примере функции

$$f(x_1, x_2, x_3) = x_1 \cdot (x_2 \vee \bar{x}_3)$$

рассмотрим переход от КНФ к СКНФ:

$$f(x_1, x_2, x_3) = x_1 \cdot (x_2 \vee \bar{x}_3) = (x_1 \vee x_2 \cdot \bar{x}_2 \vee x_3 \cdot \bar{x}_3) \cdot (x_2 \vee \bar{x}_3 \vee x_1 \cdot \bar{x}_1) =$$

$$= (x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee x_2 \vee \bar{x}_3) \cdot (x_1 \vee \bar{x}_2 \vee x_3) \cdot (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \cdot (x_1 \vee x_2 \vee \bar{x}_3) \cdot$$

$$\cdot (\bar{x}_1 \vee x_2 \vee \bar{x}_3) = (x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee x_2 \vee \bar{x}_3) \cdot (x_1 \vee \bar{x}_2 \vee x_3) \cdot (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \cdot$$

$$\cdot (\bar{x}_1 \vee x_2 \vee \bar{x}_3).$$

Покажем применение распределительного закона при проведении использованных здесь преобразований над одним из членов выражения $x_1 \vee x_2 \cdot \bar{x}_2 \vee x_3 \cdot \bar{x}_3$. Обозначим $x_1 \vee x_2 \cdot \bar{x}_2 = Y$. Тогда на основании распределительного закона

$$\underbrace{x_1 \vee x_2 \cdot \bar{x}_2 \vee x_3 \cdot \bar{x}_3}_{Y} = Y \vee x_3 \cdot \bar{x}_3 = (Y \vee x_3) \cdot (Y \vee \bar{x}_3) =$$

$$= \underbrace{(x_1 \vee x_2 \cdot \bar{x}_2 \vee x_3)}_{Y} \cdot \underbrace{(x_1 \vee x_2 \cdot \bar{x}_2 \vee \bar{x}_3)}_{Y}.$$

Далее обозначим $z_1 = x_1 \vee x_3$, $z_2 = x_1 \vee \bar{x}_3$ и вновь применим распределительный закон:

$$(x_1 \vee x_2 \cdot \bar{x}_2 \vee x_3) \cdot (x_1 \vee x_2 \cdot \bar{x}_2 \vee \bar{x}_3) = (z_1 \vee x_2 \cdot \bar{x}_2) \cdot (z_2 \vee x_2 \cdot \bar{x}_2) =$$

$$= (z_1 \vee x_2) \cdot (z_1 \vee \bar{x}_2) \cdot (z_2 \vee x_2) \cdot (z_2 \vee \bar{x}_2).$$

Подставив сюда значения $z_1$ и $z_2$, получим соответствующие члены приведенного выше выражения при переходе от КНФ к СКНФ.

Совершенная КНФ функции легко строится по таблице истинности. Рассмотрим в качестве примера функцию, приведенную в табл 1.10.

$$f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee x_2 \vee \bar{x}_3) \cdot (\bar{x}_1 \vee x_2 \vee x_3) \cdot$$

$$\cdot (\bar{x}_1 \vee \bar{x}_2 \vee x_3). \qquad (1.11)$$

Выражение содержит столько членов, связанных операцией конъюнкции, сколько нулей имеется среди значений функции $f(x_1, x_2, x_3)$ в таблице истинности. Таким образом, каждому набору значений аргументов, на котором функция равна нулю, соответствует определенный член СКНФ, принимающий на этом наборе значений нуль. Так как члены СКНФ связаны операцией конъюнкции, то при обращении в нуль одного из членов функция оказывается равной нулю.

Таким образом, можно сформулировать правило записи СКНФ функции, заданной таблицей истинности. Следует записать столько конъюнктивных членов, представляющих собой дизъюнкции всех аргументов, при скольких наборах значений аргументов функция равна

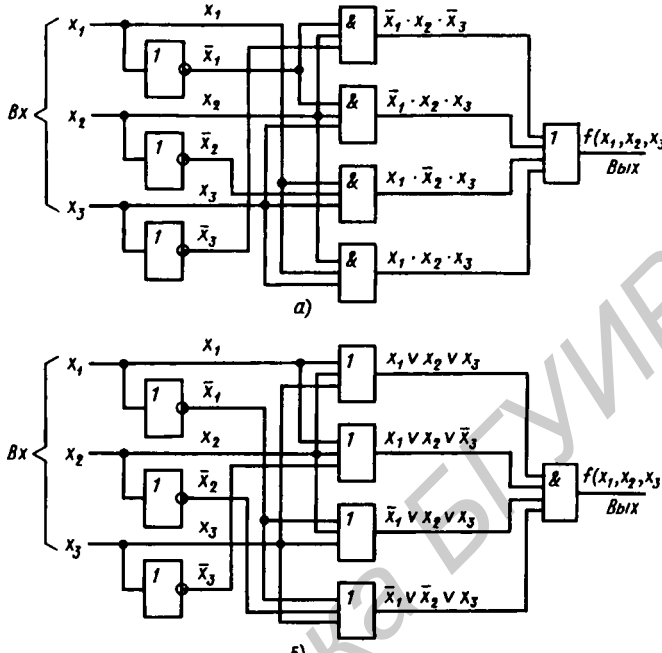



Рис. 1.22

нулю, и если в наборе значение аргумента равно единице, то в дизъюнкцию входит инверсия этого аргумента. Любая функция имеет единственную СКНФ.

Структурная схема логического устройства может быть построена непосредственно по канонической форме (СДНФ или СКНФ) реализуемой функции. Получающиеся при этом схемы для функций (1.10) и (1.11) показаны на рис. 1.22,*а* и *б*. Недостаток такого метода построения структурных схем, обеспечивающего в общем правильное функционирование устройства, состоит в том, что получающиеся схемы чаще всего неоправданно сложные, требуют использования большого числа логических элементов, имеют низкие экономичность и надежность. Во многих случаях удается так упростить логическое выражение, не изменив функции, что соответствующая структурная схема оказывается существенно более простой. Методы такого упрощения функции называются методами *минимизации функций*.

### Минимизация логических функций методом Квайна

Метод Квайна позволяет представлять функции в ДНФ или КНФ с минимальным числом членов и минимальным числом букв в членах.

Этот метод содержит два этапа преобразования выражения функции: на первом этапе осуществляется переход от канонической формы (СДНФ или СКНФ) к так называемой *сокращенной форме*, на втором этапе — переход от сокращенной формы логического выражения к *минимальной форме*.

**Первый этап (получение сокращенной формы).** Пусть заданная функция $f$ представлена в СДНФ. Переход к сокращенной форме основан на последовательном применении двух операций: *операции склеивания* и *операции поглощения*.

Для выполнения операции склеивания в выражении функции выявляются пары членов вида $w \cdot x$ и $w \cdot \bar{x}$, различающихся лишь тем, что один из аргументов в одном из членов представлен без инверсии, а в другом — с инверсией. Затем проводится склеивание таких пар членов: $w \cdot x \vee w \cdot \bar{x} = w \cdot (x \vee \bar{x}) = w$, и результаты склеивания $w$ вводятся в выражение функции в качестве дополнительных членов. Далее выполняется операция поглощения. Она основана на равенстве $w \vee w \cdot z = w \cdot (1 \vee z) = w$ (член $w$ поглощает член $w \cdot z$). При проведении этой операции из логического выражения вычеркиваются все члены, поглощаемые членами, которые введены в результате операции склеивания.

Операции склеивания и поглощения выполняются последовательно до тех пор, пока это возможно. Покажем применение этих операций к функции, представленной табл. 1.11.

*Таблица 1.11*

| $x_1$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|
| $x_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_3$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3)$ | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |

Записываем СДНФ функции

$$f(x_1, x_2, x_3) = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \vee x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee x_1 \cdot \bar{x}_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \bar{x}_3 \vee$$

$$\vee x_1 \cdot x_2 \cdot x_3. \qquad (1.12)$$

Попарным сравнением членов (каждого из членов со всеми последующими) выявляем склеивающиеся пары членов:

1-й и 4-й члены (результат склеивания $x_2 \cdot \bar{x}_3$);

2-й и 3-й члены (результат склеивания $x_1 \cdot \bar{x}_2$);

2-й и 4-й члены (результат склеивания $x_1 \cdot \bar{x}_3$);

3-й и 5-й члены (результат склеивания $x_1 \cdot x_3$);

4-й и 5-й члены (результат склеивания $x_1 \cdot x_2$).

Результаты операции склеивания вводим в выражение функции и проводим операцию поглощения ими членов исходного выражения:

$$f(x_1, x_2, x_3) = \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \vee x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee x_1 \cdot \overline{x}_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \overline{x}_3 \vee$$
$$\vee x_1 \cdot x_2 \cdot x_3 \vee x_2 \cdot \overline{x}_3 \vee x_1 \cdot \overline{x}_2 \vee x_1 \cdot \overline{x}_3 \vee x_1 \cdot x_3 \vee x_1 \cdot x_2$$

Члены операции склеивания

Член $x_2 \cdot \overline{x}_3$ поглощает те члены исходного выражения, которые содержат $x_2 \cdot \overline{x}_3$, т.е. первый и четвертый. Эти члены вычеркиваются. Член $x_1 \cdot \overline{x}_2$ поглощает второй и третий, а член $x_1 \cdot x_3$ — пятый член исходного выражения.

Повторяем операции склеивания и поглощения:

$$f(x_1, x_2, x_3) = x_2 \cdot \overline{x}_3 \vee x_1 \cdot \overline{x}_2 \vee x_1 \cdot \overline{x}_3 \vee x_1 \cdot x_3 \vee x_1 \cdot x_2 \vee x_1$$

Член операции склеивания

Здесь склеивается лишь пара членов $x_1 \cdot \overline{x}_2$ и $x_1 \cdot x_2$ (склеивание пары членов $x_1 \cdot \overline{x}_3$ и $x_1 \cdot x_3$ приводит к тому же результату), результат склеивания $x_1$ поглощает 2-, 3-, 4-, 5-й члены выражения. Дальнейшее проведение операций склеивания и поглощения оказывается невозможным, сокращенная форма выражения заданной функции (в данном примере она совпадает с минимальной формой)

$$f(x_1, x_2, x_3) = x_2 \cdot \overline{x}_3 \vee x_1. \qquad (1.13)$$

Члены сокращенной формы (в данном примере такими членами служат $x_2 \cdot \overline{x}_3$ и $x_1$) называются *простыми импликантами* функции. Как видим, получено выражение, существенно более простое по сравнению с СДНФ функции.

На рис. 1.23 приведена структурная схема логического устройства в базисе И, ИЛИ, НЕ, построенная с использованием (1.13).

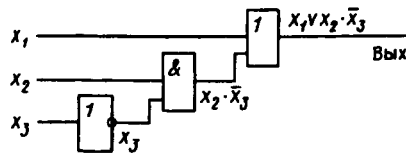



Рис. 1.23

**Второй этап (получение минимальной формы).** Сокращенная форма может содержать лишние члены, исключение которых из выражения не повлияет на значение функции.

Дальнейшее упрощение логического выражения достигается исключением из выражения лишних членов. В этом и заключается содержание второго этапа минимизации. Покажем этот этап минимизации логического выражения на примере построения логического устройства для функции, заданной в табл. 1.12.

*Таблица 1.12*

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3, x_4)$ | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Совершенная ДНФ этой функции

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4 \vee$$

$$\vee \bar{x}_1 \cdot x_2 \cdot x_3 \cdot \bar{x}_4 \vee x_1 \cdot x_2 \cdot x_3 \cdot \bar{x}_4 \vee x_1 \cdot x_2 \cdot x_3 \cdot x_4 . \qquad (1.14)$$

Для получения сокращенной формы проводим операции склеивания и поглощения:

$$f(x_1, x_2, x_3, x_4) = \cancel{\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4} \vee \cancel{\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4} \vee \cancel{\bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4} \vee$$
$$\vee \cancel{\bar{x}_1 \cdot x_2 \cdot x_3 \cdot \bar{x}_4} \vee \cancel{x_1 \cdot x_2 \cdot x_3 \cdot \bar{x}_4} \vee \cancel{x_1 \cdot x_2 \cdot x_3 \cdot x_4} \vee \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee \qquad (1.15)$$
$$\vee \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_4 \vee \bar{x}_1 \cdot x_3 \cdot \bar{x}_4 \vee x_2 \cdot x_3 \cdot \bar{x}_4 \vee x_1 \cdot x_2 \cdot x_3$$

Выражение (1.15) представляет собой сокращенную форму логического выражения заданной функции, а члены его являются простыми импликантами функции. Переход от сокращенной формы к минимальной осуществляется с помощью импликантной матрицы, приведенной в табл. 1.13.

В столбцы импликантной матрицы вписываются члены СДНФ заданной функции, в строки — простые импликанты функции, т.е. члены сокращенной формы логического выражения функции. Отмечаются (например, крестиками) столбцы членов СДНФ, поглощаемых отдельными простыми импликантами. В табл. 1.13 простая импликанта $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3$ поглощает члены $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4$, $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4$ (в первом и во втором столбцах первой строки поставлены крестики).

*Таблица 1.13*

| Простая импликанта | $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4$ | $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4$ | $\bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4$ | $\bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \cdot \bar{x}_4$ | $x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4$ | $x_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4$ |
|---|---|---|---|---|---|---|
| $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3$ | × | × | | | | |
| $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_4$ | × | | × | | | |
| $\bar{x}_1 \cdot x_3 \cdot \bar{x}_4$ | | | × | × | | |
| $x_2 \cdot x_3 \cdot \bar{x}_4$ | | | | × | × | |
| $x_1 \cdot x_2 \cdot x_3$ | | | | | × | × |

Вторая импликанта поглощает первый и третий члены СДНФ (крестики поставлены в первом и третьем столбцах второй строки) и т.д. Импликанты, которые не могут быть лишними и, следовательно, не могут быть исключены из сокращенной формы, составляют *ядро*. Входящие в ядро импликанты легко определяются по импликантной матрице. Для каждой из них имеется хотя бы один столбец, перекрываемый только данной импликантой.

В рассматриваемом примере ядро составляют импликанты $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3$ и $x_1 \cdot x_2 \cdot x_3$ (только ими перекрываются второй и шестой столбцы матрицы). Исключение из сокращенной формы одновременно всех импликант, не входящих в ядро, невозможно, так как исключение одной из импликант может превратить другую уже в нелишний член.

Для получения минимальной формы достаточно выбрать из импликант, не входящих в ядро, такое минимальное их число с минимальным количеством букв в каждой из этих импликант, которое обеспечит перекрытие всех столбцов, не перекрытых членами ядра. В рассматриваемом примере необходимо импликантами, не входящими в ядро, перекрыть третий и четвертый столбцы матрицы. Это может быть достигнуто различными способами, но так как необходимо выбирать минимальное число импликант, то, очевидно, для перекрытия этих столбцов следует выбрать импликанту $\bar{x}_1 \cdot x_3 \cdot \bar{x}_4$.

*Минимальная дизъюнктивная нормальная форма* (МДНФ) заданной функции

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee x_1 \cdot x_2 \cdot x_3 \vee \bar{x}_1 \cdot x_3 \cdot \bar{x}_4. \qquad (1.16)$$

Структурная схема, соответствующая этому выражению, приведена на рис. 1.24.

Переход от сокращенной формы (1.15) к МДНФ (1.16) был осуществлен путем исключения лишних членов — импликант $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_4$ и $x_2 \cdot x_3 \cdot \bar{x}_4$. Покажем допустимость подобного исключения членов из логического выражения.

Импликанты $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_4$ и $x_2 \cdot x_3 \cdot \bar{x}_4$ становятся равными *лог.1* соответственно при следующих наборах значений аргументов:

$$x_1 = 0, x_2 = 0, x_4 = 0 \text{ и } x_2 = 1, x_3 = 1, x_4 = 0. \qquad (1.17)$$

Роль этих импликант в выражении сокращенной формы функции (1.15) заключается лишь в том, чтобы на приведенных наборах (1.17) значений аргументов присваивать функции $f(x_1, x_2, x_3, x_4)$ значение 1. Однако при этих наборах функция равна 1 из-за остальных импликант выражения (1.15). Действительно, подставляя (1.17) в (1.16), получаем:

при $x_1 = 0, x_2 = 0, x_4 = 0$

$$f(0, 0, x_3, 0) = 1 \cdot 1 \cdot \bar{x}_3 \vee 0 \cdot 0 \cdot x_3 \vee 1 \cdot x_3 \cdot 1 = \bar{x}_3 \vee x_3 = 1;$$

при $x_2 = 1, x_3 = 1, x_4 = 0$

$$f(x_1, 1, 1, 0) = \bar{x}_1 \cdot 0 \cdot 0 \vee x_1 \cdot 1 \cdot 1 \vee \bar{x}_1 \cdot 1 \cdot 1 = x_1 \vee \bar{x}_1 = 1.$$

Таким образом, доказана справедливость операции исключения из (1.15) членов $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_4$ и $x_2 \cdot x_3 \cdot \bar{x}_4$.

До сих пор рассматривалось получение минимальной ДНФ. При использовании метода Квайна для получения *минимальной конъюнктив-*

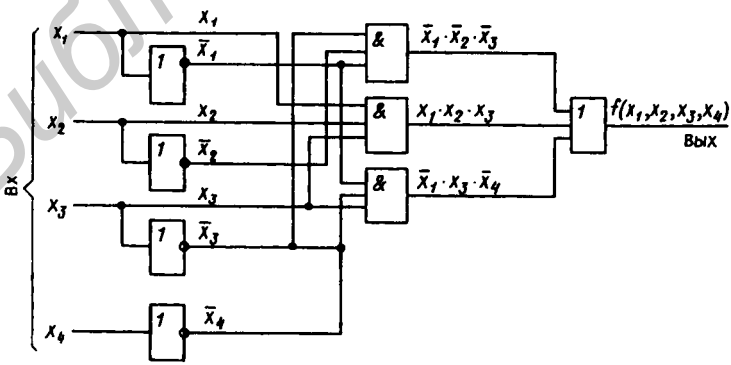



Рис. 1.24

*ной нормальной формы* (МКНФ) логической функции имеются следующие особенности:

исходной для минимизации формой логического выражения заданной функции является СКНФ;

пары склеиваемых членов имеют вид $w \vee x$ и $w \vee \bar{x}$;

операция поглощения проводится в соответствии с выражением

$$z \cdot (z \vee y) = z \vee z \cdot y = z \cdot (1 \vee y) = z.$$

Рассмотрим применение метода Квайна на примере минимизации функции, заданной таблицей истинности (табл. 1.14).

*Таблица 1.14*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_3$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3)$ | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

Совершенная КНФ рассматриваемой функции

$$f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee \bar{x}_3) \cdot (x_1 \vee \bar{x}_2 \vee x_3) \cdot (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \cdot (\bar{x}_1 \vee x_2 \vee \bar{x}_3).$$

Склеивающиеся пары членов:

1-й и 3-й члены (результат склеивания $x_1 \vee \bar{x}_3$);

1-й и 4-й члены (результат склеивания $x_2 \vee \bar{x}_3$);

2-й и 3-й члены (результат склеивания $x_1 \vee \bar{x}_2$).

Проводим операции склеивания и поглощения:

$$f(x_1, x_2, x_3) = \underline{(x_1 \vee x_2 \vee \bar{x}_3)} \cdot \underline{(x_1 \vee \bar{x}_2 \vee x_3)} \cdot \underline{(x_1 \vee \bar{x}_2 \vee \bar{x}_3)} \cdot \underline{(\bar{x}_1 \vee x_2 \vee \bar{x}_3)}$$
$$\underbrace{\cdot (x_1 \vee \bar{x}_3) \cdot (x_2 \vee \bar{x}_3) \cdot (x_1 \vee \bar{x}_2)}$$

**Члены операции склеивания**

Полученное выражение является сокращенной формой функции.

Для перехода к минимальной форме строим импликантную матрицу (табл. 1.15). Все столбцы матрицы перекрываются импликантами $x_2 \vee \bar{x}_3$ и $x_1 \vee \bar{x}_2$. Следовательно, член $x_1 \vee \bar{x}_3$ является лишним и минимальная конъюнктивная нормальная форма (МКНФ) заданной функции $f(x_1, x_2, x_3) = (x_1 \vee \bar{x}_3) \cdot (x_1 \vee \bar{x}_2)$.

*Таблица 1.15*

| Простая импликанта | $x_1 \vee x_2 \vee \bar{x}_3$ | $x_1 \vee \bar{x}_2 \vee x_3$ | $x_1 \vee \bar{x}_2 \vee \bar{x}_3$ | $\bar{x}_1 \vee x_2 \vee \bar{x}_3$ |
|---|---|---|---|---|
| $x_1 \vee \bar{x}_3$ | × | | × | |
| $x_2 \vee \bar{x}_3$ | × | | | × |
| $x_1 \vee \bar{x}_2$ | | × | × | |

## Минимизация логических функций методом Квайна — Мак-Класки

Мак-Класки предложил прием, который на этапе нахождения сокращенных ДНФ и КНФ упрощает процесс минимизации и, кроме того, позволяет описать этот процесс для выполнения на ЭВМ. Прием предусматривает следующую последовательность действий для получения сокращенной ДНФ.

1. СДНФ функции представляют наборами значений аргументов, на которых функция равна *лог.1*.

*Таблица 1.16*

| Десятичный эквивалент набора аргументов | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3, x_4)$ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

Пусть функция задана табл. 1.16. СДНФ функции записываем в виде совокупности наборов (представленных их десятичными эквивалентами), на которых функция принимает значение *лог.1*:

$$f(x_1, x_2, x_3, x_4) = \bigvee_{1} (1, 5, 9, 10, 11, 13).$$

Эта запись читается так: "функция $f$ принимает значение *лог.1* на наборах, соответствующих десятичным эквивалентам или 1, или 5, или

9, или 10, или 11, или 13". Функцию можно записать и через наборы, представленные в двоичной форме:

$f = 0001 \vee 0101 \vee 1001 \vee 1010 \vee 1011 \vee 1101.$

2. Все члены в этой форме СДНФ разбивают на группы по числу единиц, содержащихся в наборах (представленных в двоичной форме). Эта разбивка наборов на группы для рассматриваемой функции представлена в графе I этапа табл. 1.17.

*Таблица 1.17*

| Номер группы | Наборы | | |
|---|---|---|---|
| | I этап | II этап | III этап |
| 0 | — | — | — |
| 1 | ~~0001~~ | ~~0*01~~<br>~~*001~~ | **01 |
| 2 | ~~0101~~<br>~~1001~~<br>~~1010~~ | ~~*101~~<br>10*1<br>~~1*01~~<br>101* | 10*1<br>101* |
| 3 | ~~1011~~<br>~~1101~~ | — | — |

3. Производят склеивание наборов. Склеиваться могут только наборы соседних групп, различающиеся лишь в одном разряде. Результат склеивания пары наборов содержит на месте разряда с различающимися значениями в наборах символ * и заносится в графу следующего этапа, а пара склеивающихся наборов вычеркивается (при этом вычеркнутые наборы должны использоваться в последующих поисках склеивающихся пар наборов). Так, склеивание пары наборов 0001 и 0101 графы I этапа приводит к набору 0 * 01, записываемому в графе II этапа.

Результаты склеивания наборов II этапа заносятся в графу III этапа. Сюда перенесены и невычеркнутые наборы из графы II этапа. Дальнейшее склеивание оказывается невозможным.

Наборы графы последнего этапа изображают простые импликанты функции, т.е. члены сокращенной ДНФ. В рассматриваемом примере сокращенная ДНФ функции

$f(x_1, x_2, x_3, x_4) = **01 \vee 10*1 \vee 101*.$

Эта запись соответствует логическому выражению, получаемому по правилу: каждый набор соответствует отдельной импликанте; каждому символу в наборе соответствует переменная функции с индексом, совпадающим с номером позиции символа в наборе; если символом является $*$, то соответствующая переменная в выражении импликанты отсутствует; если символом является 0, то соответствующая переменная в выражении импликанты присутствует с инверсией; при символе 1 переменная записывается без инверсии:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_3 \cdot x_4 \vee x_1 \cdot \bar{x}_2 \cdot x_4 \vee x_1 \cdot \bar{x}_2 \cdot x_3.$$

Переход от сокращенной ДНФ к минимальной ДНФ может производиться с помощью импликантной матрицы, как и в методе Квайна. Различие может состоять лишь в том, что в импликантной матрице члены СДНФ и сокращенной ДНФ удобней представлять соответствующими им двоичными комбинациями. Для рассматриваемого примера импликантная матрица приведена в табл. 1.18.

*Таблица 1.18*

| Простые импликанты | 0001 | 0101 | 1001 | 1010 | 1011 | 1101 |
|---|---|---|---|---|---|---|
| **01 | × | × | × | | | × |
| 10*1 | | | × | | × | |
| 101* | | | | × | × | |

Из таблицы следует, что МНДФ функции

$$f(x_1, x_2, x_3, x_4) = **01 \vee 101* = \bar{x}_3 \cdot x_4 \vee x_1 \cdot \bar{x}_2 \cdot x_3.$$

Методом Мак-Класки может быть получена и МКНФ функции. По сравнению с описанной выше последовательностью действий различие в этом случае заключается в следующем: функция представляется наборами, на которых она принимает значение *лог.0*, и в полученных в результате минимизации наборах символу 0 соответствует переменная без инверсии, а символу 1 — переменная с инверсией.

Рассмотрим получение МКНФ функции, представленной в табл.1.19.

*Таблица 1.19*

| Десятичные эквиваленты набора аргументов | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3, x_4)$ | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |

Запишем функцию как совокупность наборов, на которых функция имеет значение *лог.0*:

$$f(x_1, x_2, x_3, x_4) = \bigvee_{0} (0, 1, 2, 6, 10, 14).$$

Этапы минимизации показаны в табл.1.20. В графе I этапа приведены наборы, соответствующие значениям функции, равным *лог.0*. В последующих графах приведены результаты склеивания.

*Таблица 1.20*

| Группа | Наборы | | |
|---|---|---|---|
| | I этап | II этап | III этап |
| 0 | ~~0000~~ | 000* | 000* |
| | | 00*0 | 00*0 |
| 1 | ~~0001~~ | ~~0*10~~ | **10 |
| | ~~0010~~ | ~~*010~~ | |
| 2 | ~~0110~~ | ~~*110~~ | |
| | ~~1010~~ | ~~1*10~~ | |
| 3 | ~~1110~~ | | |

Сокращенная КНФ записывается через инверсные комбинации наборов последнего этапа:

$$f(x_1, x_2, x_3, x_4) = (111*) \cdot (11*1) \cdot (**01) = (x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee x_2 \vee x_4) \cdot$$

$$\cdot (\bar{x}_3 \vee x_4).$$

Переход от сокращенной КНФ к минимальной КНФ не имеет особенностей.

### Переход от сокращенной формы к минимальной форме методом Петрика

Метод Петрика формализует (и таким образом, делает возможным применение ЭВМ) нахождение всех тупиковых ДНФ по импликантной матрице, т.е. всех вариантов наборов простых импликант, обеспечивающих перекрытие всех столбцов матрицы. Из полученных тупиковых ДНФ затем необходимо выбрать минимальную ДНФ, содержащую наименьшее число наиболее простых членов.

Суть метода заключается в следующем. Все простые импликанты обозначаются различными буквами (например, прописными латинскими буквами). Затем для каждого $i$-го столбца импликантной матрицы строится дизъюнкция всех букв, обозначающих простые импликанты, перекрывающие данный столбец. Далее импликантная матрица представляется конъюнцией всех построенных для отдельных столбцов дизъюнкций. После выполнения над полученным выражением операций поглощения и перехода к ДНФ каждый член образующегося выражения определяет набор простых импликант тупиковой ДНФ.

Проиллюстрируем применение метода на примере функции, заданной таблицей истинности, приведенной в табл. 1.21. Для получения сокращенной формы воспользуемся методом Квайна — Мак-Класки. Процесс минимизации этим методом показан в табл. 1.22. В табл. 1.23 приведена импликантная матрица.

Далее воспользуемся методом Петрика. Для каждой простой импликанты в табл. 1.23 введено буквенное обозначение. Индексами при буквах, которыми представляются простые импликанты, обозначено количество переменных в простых импликантах. В соответствии с приведенным выше правилом строим логическое выражение в виде конъюнкции членов, каждый из которых представляет собой дизъюнкцию буквенных обозначений простых импликант, перекрывающих отдельные столбцы импликантной матрицы:

*Таблица 1.21*

| $x_3x_4$ \ $x_1x_2$ | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| 00 | | 1 | | 1 |
| 01 | | | | |
| 10 | 1 | 1 | | 1 |
| 11 | 1 | | 1 | 1 |

*Таблица 1.22*

| Группа | Наборы | | |
|---|---|---|---|
| | I этап | II этап | III этап |
| 0 | — | — | — |
| 1 | ~~0 0 1 0~~<br>~~0 1 0 0~~ | 0 0 1 *<br>0 * 1 0<br>~~* 1 0 0~~<br>~~0 1 * 0~~ | 0 0 1 *<br>0 * 1 0<br>* 1 * 0 |
| 2 | ~~1 1 0 0~~<br>~~0 0 1 1~~<br>~~0 1 1 0~~ | ~~1 1 * 0~~<br>* 0 1 1<br>~~* 1 1 0~~ | * 0 1 1 |
| 3 | ~~1 1 1 0~~<br>~~1 0 1 1~~ | 1 1 1 *<br>1 * 1 1 | 1 1 1 *<br>1 * 1 1 |
| 4 | ~~1 1 1 1~~ | — | — |

*Таблица 1.23*

| Простые импликанты | 1100 | 1110 | 1111 | 1011 | 0011 | 0100 | 0110 | 0010 |
|---|---|---|---|---|---|---|---|---|
| $A_3$ 001* | | | | | × | | | × |
| $B_3$ 0*10 | | | | | | | × | × |
| $C_2$ *1*0 | × | × | | | | × | × | |
| $D_3$ *011 | | | | × | × | | | |
| $E_3$ 111* | | × | × | | | | | |
| $F_3$ 1*11 | | | × | × | | | | |

$$C \cdot (C \vee E) \cdot (E \vee F) \cdot (D \vee F) \cdot (A \vee D) \cdot C \cdot (B \vee C) \cdot (A \vee B).$$

Преобразуем выражение. В результате операции поглощения оно приводится к виду

$$C \cdot (E \vee F) \cdot (D \vee F) \cdot (A \vee D) \cdot (A \vee B).$$

Далее в соответствии с распределительным законом

$(E \vee F) \cdot (D \vee F) = F \vee D \cdot E$, $(A \vee D) \cdot (A \vee B) = A \vee B \cdot D$

и выражение приводится к виду

$C \cdot (F \vee D \cdot E) \cdot (A \vee B \cdot D)$.

Наконец, переходя от конъюнктивной формы выражения к его дизъюнктивной форме, получаем окончательно

$A_3 \cdot C_2 \cdot F_3 \vee A_3 \cdot C_2 \cdot D_3 \cdot E_3 \vee B_3 \cdot C_2 \cdot D_3 \cdot F_3 \vee B_3 \cdot C_2 \cdot D_3 \cdot E_3$.

Из данного выражения следует, что имеется четыре тупиковых ДНФ. Из них минимальная ДНФ (с минимальным числом букв) $A_3 \cdot C_2 \cdot F_3$. Следовательно, МДНФ функции

$f(x_1, x_2, x_3, x_4) = x_2 \cdot \bar{x}_4 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \vee x_1 \cdot x_3 \cdot x_4$.

Аналогично ищется МКНФ.

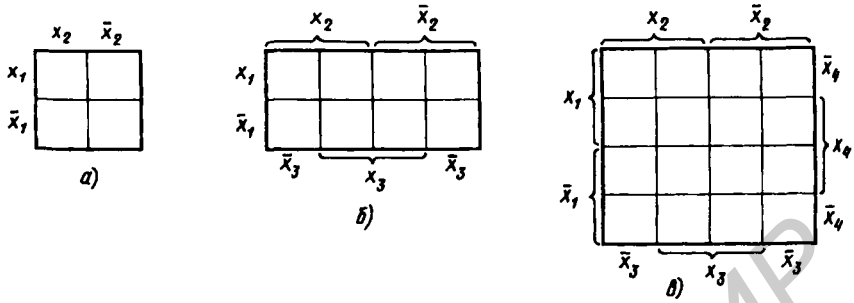
### Минимизация логических функций методом карт Вейча

Метод Квайна имеет четко формулируемые правила проведения отдельных операций. Благодаря этому он может быть применен для минимизации функций с использованием ЭВМ в тех случаях, когда минимизируемая функция достаточно сложна (содержит большое число аргументов и каноническая форма имеет большое число членов). Однако для минимизации функции ручным способом (без ЭВМ) этот метод оказывается весьма трудоемким. Трудоемкость метода Квайна связана с необходимостью попарного сравнения всех членов выражения для выявления склеиваемых членов.

Метод минимизации функции с помощью карт Вейча обеспечивает простоту получения результатов. Он используется при минимизации относительно несложных функций (с числом аргументов до пяти) ручным способом. В отличие от метода Квайна этот метод требует изобретательности и не может быть применен для решения задачи минимизации с помощью ЭВМ. Карта Вейча представляет собой определенную форму таблицы истинности. Таблицы 1.24 являются картами Вейча для функций соответственно двух (*а*), трех (*б*), четырех (*в*) аргументов.

Каждая клетка карты соответствует некоторому набору значений аргументов. Этот набор аргументов определяется присвоением значения *лог.1* буквам, на пересечении строк и столбцов которых расположена клетка. Так, в карте функций четырех аргументов (табл. 1.24,*в*) клетки первой строки соответствуют следующим комбинациям значений аргументов:

1-я клетка $x_1 = 1$, $x_2 = 1$, $\bar{x}_3 = 1$ ($x_3 = 0$), $\bar{x}_4 = 1$ ($x_4 = 0$);




2-я клетка $x_1 = 1, \quad x_2 = 1, \quad x_3 = 1, \quad x_4 = 0;$

3-я клетка $x_1 = 1, \quad x_2 = 0, \quad x_3 = 1, \quad x_4 = 0;$

4-я клетка $x_1 = 1, \quad x_2 = 0, \quad x_3 = 0, \quad x_4 = 0.$

Число клеток карты равно числу всех возможных наборов значений аргументов $2^n$ ( $n$ — число аргументов функций ). В каждую из клеток карты записывается значение функции на соответствующем этой клетке наборе значений аргументов. Пусть функция задана таблицей истинности в форме, которая использовалась ранее (табл. 1.25). Таблица истинности этой функции в форме карты Вейча представлена табл. 1.26.

*Таблица 1.25*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_3$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3)$ | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |

Как видим, карта Вейча определяет значения функции на всех возможных наборах значений аргументов и является таблицей истинности. Карты Вейча компактны, но главное их достоинство состоит в следующем. При любом переходе из одной клетки в соседнюю вдоль столбца или строки изменяется значение лишь одного аргумента функции. Следовательно, если в паре соседних клеток содержится 1, то над соответствующими им членами канонической формы может быть проведена операция склеивания. Таким образом, облегчается поиск склеиваемых членов.

Сформулируем правила получения МДНФ функций с помощью карт Вейча.

Все клетки, содержащие 1, объединяются в замкнутые области. При этом каждая область должна представлять собой прямоугольник с числом клеток $2^k$, где $k = 0, 1, 2,...$ Значит, допустимое число клеток в

области 1, 2, 4, 8,... Области могут пересекаться и одни и те же клетки могут входить в разные области. Затем проводится запись выражения МДНФ функции. Каждая из областей в МДНФ представляется членом, число букв в котором на $k$ меньше общего числа аргументов функции $n$

*Таблица 1.26*

| | $x_2$ | | $\bar{x}_2$ | |
|---|---|---|---|---|
| $x_1$ | 1 | 1 | 0 | 0 |
| $\bar{x}_1$ | 0 | 1 | 1 | 0 |
| | $\bar{x}_3$ | $x_3$ | | $\bar{x}_3$ |

*Таблица 1.27*

| | $x_2$ | | $\bar{x}_2$ | |
|---|---|---|---|---|
| $x_1$ | 1 | 1 | 0 | 0 |
| $\bar{x}_1$ | 0 | 1 | 1 | 0 |
| | $\bar{x}_3$ | $x_3$ | | $\bar{x}_3$ |

I, II

(т.е. равно $n - k$). Каждый член МДНФ составляется лишь из тех аргументов, которые для клеток соответствующей области имеют одинаковое значение (без инверсии либо с инверсией).

Таким образом, при охвате клеток замкнутыми областями следует стремиться, чтобы число областей было минимальным (при этом минимальным будет число членов в МДНФ функции), а каждая область содержала возможно большее число клеток (при этом минимальным будет число букв в членах МДНФ функции).

Рассмотрим минимизацию с помощью карты Вейча функции трех аргументов, представленной табл. 1.27. Все клетки, содержащие 1, охватываются двумя областями. В каждой из областей $2^1$ клеток, для них $n - k = 3 - 1 = 2$, и эти области в МДНФ будут представлены членами, содержащими по две буквы. Первой области соответствует член $x_1 \cdot x_2$ (аргумент $x_3$ здесь не присутствует, так как для одной клетки этой области он имеет значение без инверсии, для другой — с инверсией); второй области соответствует член $\bar{x}_1 \cdot x_3$. Следовательно, МДНФ функции

$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \vee \bar{x}_1 \cdot x_3.$$

Рассмотрим пример минимизации функции четырех аргументов, заданной табл. 1.28. Первая и четвертая области имеют по две клетки, для них $n - k = 4 - 1 = 3$. Эти области в МДНФ будут представлены членами, содержащими по три буквы. Вторая и третья области содержат по четыре клетки и в МДНФ выражаются членами, содержащими по две буквы ($n - k = 4 - 2 = 2$). Минимальная ДНФ функции

$$f(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 \cdot \bar{x}_4 \vee x_1 \cdot x_3 \vee \bar{x}_2 \cdot x_3 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot x_4.$$

*Таблица 1.28*

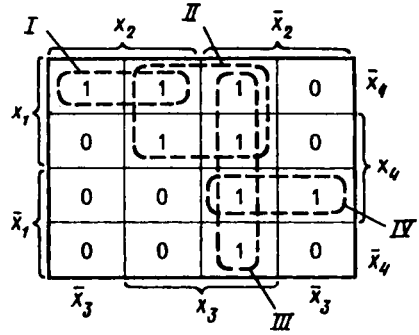
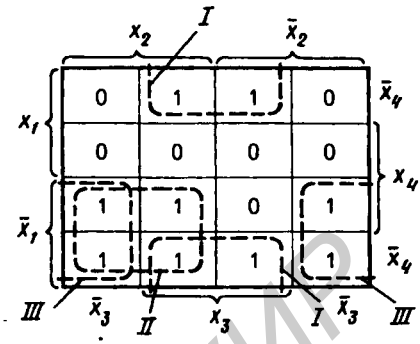



*Таблица 1.29*




При построении замкнутых областей допускается сворачивание карты в цилиндр с объединением ее противоположных граней. В силу этого крайние клетки строки или столбца таблицы рассматриваются как соседние и могут быть объединены в общую область. Иллюстрацию этого приема проведем на примере функции, представленной табл. 1.29. Минимальная ДНФ функции

$$f(x_1, x_2, x_3, x_4) = x_3 \cdot \bar{x}_4 \vee \bar{x}_1 \cdot x_2 \vee \bar{x}_1 \cdot \bar{x}_3.$$

В силу допустимости такого сворачивания карты вдоль горизонтальной и вертикальной осей, например, клетки, расположенные в четырех углах карты функции четырех переменных, оказываются соседними и могут быть объединены в одну область. Покажем это на примере минимизации функции, заданной табл. 1.30. Минимальная ДНФ функции

$$f(x_1, x_2, x_3, x_4) = \bar{x}_3 \cdot \bar{x}_4 \vee \bar{x}_2.$$

*Таблица 1.30*

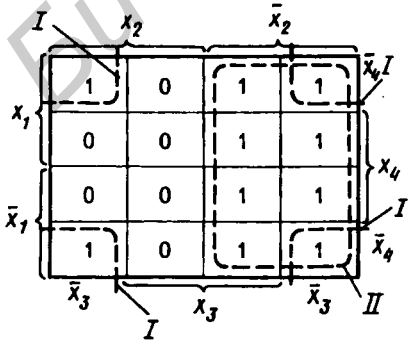
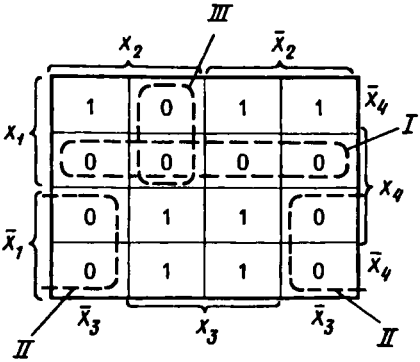



*Таблица 1.31*



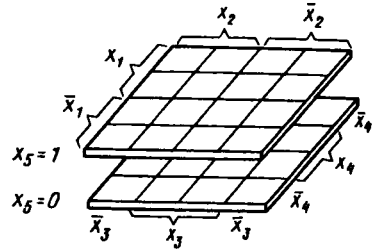



Рис. 1.25

Для получения МКНФ функции замкнутыми областями охватываются клетки с нулевыми значениями функции, и при записи членов логического выражения берутся инверсии аргументов, на пересечении которых находятся области. Так, для функции, приведенной в табл. 1.31, МКНФ

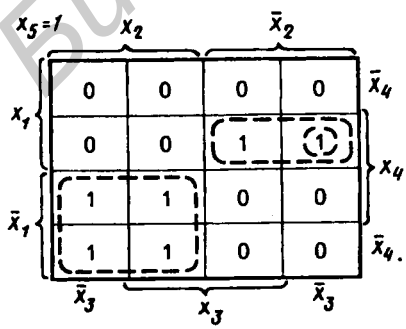
$$f(x_1, x_2, x_3, x_4) = (\bar{x}_1 \vee \bar{x}_4) \cdot (x_1 \vee x_3) \cdot (\bar{x}_1 \vee \bar{x}_2 \cdot \bar{x}_3).$$

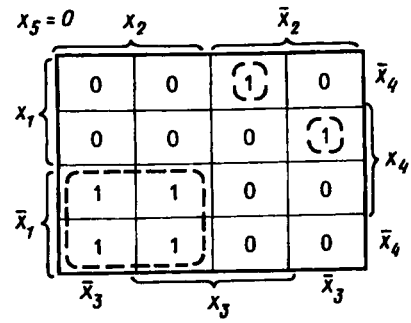
До сих пор рассматривались логические функции с числом аргументов до четырех. Представление функции и минимизация ее с помощью карт Вейча усложняются, если число аргументов больше четырех.

В табл. 1.32 показано представление с помощью карт Вейча функции пяти аргументов. Таблица истинности здесь состоит из двух карт, каждая из которых представляет собой карту четырех переменных. Одна из них соответствует $x_5 = 1$, другая — $x_5 = 0$. Эти карты можно мысленно расположить одна над другой (рис. 1.25). При этом области охвата клеток могут быть трехмерными, т.е. одной областью могут охватываться клетки двух карт.

Для функции, приведенной в табл. 1.32, МДНФ

*Таблица 1.32*



$$f(x_1, x_2, x_3, x_4, x_5) = \bar{x}_1 \cdot x_2 \vee x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4 \vee x_1 \cdot \bar{x}_2 \cdot x_4 \cdot x_5 \vee$$

$$\vee x_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4 \cdot \bar{x}_5.$$

Для минимизации функции с числом аргументов, большим пяти, карты Вейча оказываются неудобными. Минимизация таких функций может быть выполнена методом Квайна.

### Минимизация функций с использованием карт Карно

Отличие карт Карно от карт Вейча заключается в способе обозначения строк и столбцов таблицы истинности. Таблица 1.33 иллюстрирует карты Карно для функций трех и четырех аргументов.

Аргументы функции делятся на две группы, комбинации значений аргументов одной группы приписываются столбцам таблицы, комбинации значений аргументов другой группы — строкам таблицы. Столбцы и строки обозначаются комбинациями, соответствующими последовательности чисел в *коде Грея* (это сделано для того, чтобы склеивающиеся клетки находились рядом). Обозначения столбца и строки, на пересечении которых находится клетка таблицы, образуют набор, значение функции на этом наборе записывается в клетку.

Для получения МДНФ функции охватываются областями клетки таблицы, содержащие 1. Как и в случае минимизации с помощью карт Вейча, области должны быть прямоугольной формы и содержать $2^k$ клеток (при целочисленном значении $k$). Для каждой области составляется набор из двух комбинаций: приписанных столбцам и приписанных строкам, на пересечении которых расположена область. При этом если области соответствуют несколько комбинаций *кода Грея*, приписанных столбцам или строкам, то при составлении набора области записывается общая часть этих комбинаций, а на месте различающихся разрядов комбинаций ставятся звездочки. Например, для функции, представленной табл. 1.34, области I будет соответствовать набор 1∗00 или член

*Таблица 1.33*

| $x_3$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $f(000)$ | $f(010)$ | $f(110)$ | $f(100)$ |
| 1 | $f(001)$ | $f(011)$ | $f(111)$ | $f(101)$ |

| $x_3x_4$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $f(0000)$ | $f(0100)$ | $f(1100)$ | $f(1000)$ |
| 01 | $f(0001)$ | $f(0101)$ | $f(1101)$ | $f(1001)$ |
| 11 | $f(0011)$ | $f(0111)$ | $f(1111)$ | $f(1011)$ |
| 10 | $f(0010)$ | $f(0110)$ | $f(1110)$ | $f(1010)$ |

*Таблица 1.34*

| $x_3x_4$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 1 |
| 01 | 1 | 1 | 0 | 0 |
| 11 | 1 | 1 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

I — 00 11; II — 01 01

*Таблица 1.35*

| $x_3x_4$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 0 | 0 | 1 | 1 |
| 10 | 0 | 0 | 1 | 1 |

I — 01 11; II — 10 00

МДНФ $x_1 \cdot \bar{x}_3 \cdot \bar{x}_4$, области II — набор 0**1 или член МДНФ $\bar{x}_1 \cdot x_4$. Таким образом, для этой функции МДНФ

$$f(x_1, x_2, x_3, x_4) = x_1 \cdot \bar{x}_3 \cdot \bar{x}_4 \vee \bar{x}_1 \cdot x_4.$$

Для получения МКНФ областями охватываются клетки, содержащие 0, и члены МКНФ записываются через инверсии цифр, получаемых для наборов отдельных областей. Так, для функции, представленной в табл. 1.35, области I соответствует набор *100 и член МКНФ $\bar{x}_2 \vee x_3 \vee x_4$, области II — набор 0*1* и член $x_1 \vee \bar{x}_3$. Таким образом, МКНФ функции

$$f(x_1, x_2, x_3, x_4) = (\bar{x}_2 \vee x_3 \vee x_4) \cdot (x_1 \vee \bar{x}_3).$$

### Синтез не полностью заданных логических функций

По условиям работы логического устройства некоторые наборы значений аргументов могут оказаться запрещенными для данного устройства и никогда не появиться на его входах. В этом случае функция задана не на всех наборах аргументов. Такие функции будем называть *не полностью заданными*.

При синтезе логического устройства, реализующего не полностью заданную функцию, допустимо задаваться произвольными значениями функции на запрещенных наборах аргументов. При этом в зависимости от способа задания этих значений функции минимальная форма может оказаться простой или более сложной. Таким образом, возникает проблема целесообразного доопределения функции на запрещенных наборах аргументов.

Может быть использован следующий способ получения минимальной формы не полностью заданной функции $f$:

а) записывается СДНФ (СКНФ) функции $f_0$, полученной из $f$ заданием значения 0 (значения 1 в случае СКНФ) на всех запрещенных наборах аргументов;

б) записывается СДНФ (СКНФ) функции $f_1$, полученной из $f$ заданием значения 1 (значения 0 в случае СКНФ) на всех запрещенных наборах аргументов;

в) функция $f_1$ приводится к сокращенной форме (к форме, содержащей все простые импликанты);

г) составляется импликантная таблица из всех членов функции $f_0$ и простых импликант функции $f_1$;

д) искомая минимальная форма составляется из простых импликант функции $f_1$, поглощающих все члены СДНФ (СКНФ) функции $f_0$.

*Таблица 1.36*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_3$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3)$ | | 0 | 1 | | 1 | | | 1 |

Рассмотрим применение данного метода к минимизации не полностью заданной функции, приведенной в табл. 1.36. Записываем логическое выражение функции $f_0$ в СДНФ:

$$f_0(x_1, x_2, x_3) = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \vee x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee x_1 \cdot x_2 \cdot x_3.$$

Записываем СДНФ функции $f_1$:

$$f_1(x_1, x_2, x_3) = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee x_1 \cdot \bar{x}_2 \cdot$$
$$\cdot x_3 \vee x_1 \cdot x_2 \cdot \bar{x}_3 \vee x_1 \cdot x_2 \cdot x_3.$$

Методом Квайна приводим функцию $f_1$ к сокращенной форме:

$$f_1(x_1, x_2, x_3) = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee$$
$$\vee x_1 \cdot \bar{x}_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot \bar{x}_3 \vee x_1 \cdot x_2 \cdot x_3 \vee$$
$$\vee \bar{x}_1 \cdot \bar{x}_3 \vee \bar{x}_2 \cdot \bar{x}_3 \vee \bar{x}_1 \cdot x_2 \vee x_2 \cdot \bar{x}_3 \vee x_2 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \vee$$
$$\vee x_1 \cdot \bar{x}_3 \vee x_1 \cdot x_3 \vee x_1 \cdot x_2 \vee \bar{x}_3 \vee x_2 \vee x_1$$

Члены операции склеивания

Составляем импликантную таблицу (табл. 1.37).

*Таблица 1.37*

| Простая импликанта функции $f_1$ | $\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$ | $x_1 \cdot \bar{x}_2 \cdot \bar{x}_3$ | $x_1 \cdot x_2 \cdot x_3$. |
|---|---|---|---|
| $\bar{x}_3$ | × | × | |
| $x_2$ | × | | × |
| $x_1$ | | × | × |

Минимальная форма логического выражения функции может быть получена исключением любой из трех простых импликант:

$$f(x_1, x_2, x_3) = \begin{cases} \bar{x}_3 \vee x_2, \\ x_2 \vee x_1, \\ \bar{x}_3 \vee x_1. \end{cases}$$

*Таблица 1.38*

| $x_1$ \ $x_2$ | | | | |
|---|---|---|---|---|
| | | 1 | | 1 |
| | 1 | | 0 | |

($x_3$)

Рассмотрим минимизацию той же функции методом карт Вейча (табл. 1.38). При минимизации функции данным методом следует на запрещенных наборах аргументов задавать функции такие значения, при которых клетки со значением 1 (либо 0) охватываются минимальным числом областей с максимальным числом клеток в каждой из областей. Приме-

*Таблица 1.39*

| $x_1$ \ $x_2$ | | | | |
|---|---|---|---|---|
| | 1 | 1 | 1 | 1 |
| | 1 | 0 | 0 | 1 |

($x_3$)

| $x_1$ \ $x_2$ | | | | |
|---|---|---|---|---|
| | 1 | 1 | 1 | 1 |
| | 1 | 1 | 0 | 0 |

($x_3$)

| $x_1$ \ $x_2$ | | | | |
|---|---|---|---|---|
| | 1 | 1 | 0 | 1 |
| | 1 | 1 | 0 | 1 |

($x_3$)

нительно к рассматриваемой функции такое доопределение функции может быть осуществлено тремя различными способами, представленными в табл. 1.39. Они приводят к полученным выше выражениям МДНФ функции.

### Синтез логических устройств с несколькими выходами

Пусть синтезируемое логическое устройство имеет $n$ входов и $m$ выходов (рис. 1.26). На каждом из выходов должна быть сформирована определенная функция входных переменных. Эта задача могла бы быть решена синтезированием раздельно действующих узлов, каждый из которых реализовывал бы определенную выходную функцию. Однако, если даже каждый из этих узлов будет построен минимальным образом, в целом логическое устройство может оказаться не минимальным. Действительно, такое устройство могло бы быть минимизировано путем использования общих элементов в нескольких узлах, реализующих различные выходные функции.

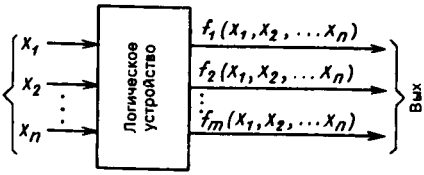



Рис.1.26

Из этих соображений приведение каждой из выходных функций к минимальной форме не является условием получения минимального в целом устройства. При минимизации устройства в целом некоторые из функций могут оказаться представленными в неминимальной форме.

Принцип получения минимальной формы устройства сводится к нахождению минимального набора членов с минимальным числом входящих в них букв, достаточного для получения всех формируемых устройством функций.

Метод построения минимальных логических устройств с несколькими выходами рассмотрим на примере реализации устройства, способ функционирования которого задан табл. 1.40.

*Таблица 1.40*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_3$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f_1(x_1, x_2, x_3)$ | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| $f_2(x_1, x_2, x_3)$ | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| $f_3(x_1, x_2, x_3)$ | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

Записываем наборы аргументов, на которых хотя бы одна из выходных функций имеет значение 1. Рядом в таблице в качестве признака записываем функции, принимающие значение 1 при данном наборе аргументов (табл. 1.41). Затем проводим операцию склеивания и получающиеся при этом члены заносим в табл. 1.42, рядом с членами записываем признаки в виде функций, общих в признаках той пары членов табл. 1.41, склеиванием которых они получены. Так, склеивание членов табл. 1.41 $\bar{x}_1 \cdot \bar{x}_2 \cdot x_3(f_1f_3)$ и $\bar{x}_1 \cdot x_2 \cdot x_3(f_1f_3)$ приводит в табл. 1.42 к члену $\bar{x}_1 \cdot x_3(f_1f_3)$; склеивание членов $\bar{x}_1 \cdot \bar{x}_2 \cdot x_3(f_1f_3)$ и $x_1 \cdot \bar{x}_2 \cdot x_3(f_2f_3)$ приводит к $\bar{x}_2 \cdot x_3(f_3)$ и т.д. Не проводится операция склеивания над членами, в признаках которых не имеется общих функций.

*Таблица 1.41*

| | |
|---|---|
| $\bar{x}_1 \cdot \bar{x}_2 \cdot x_3$ | $f_1 f_3$ |
| $\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$ | $f_2 f_3$ |
| $\bar{x}_1 \cdot x_2 \cdot x_3$ | $f_1 f_3$ |
| $x_1 \cdot \bar{x}_2 \cdot x_3$ | $f_2 f_3$ |
| $x_1 \cdot \bar{x}_2 \cdot \bar{x}_3$ | $f_2 f_3$ |
| $x_1 \cdot x_2 \cdot \bar{x}_3$ | $f_1$ |
| $x_1 \cdot x_2 \cdot x_3$ | $f_1$ |

Далее реализуется операция поглощения членами табл. 1.42 членов табл. 1.41. Операция поглощения может проводиться лишь над членами, имеющими одинаковую комбинацию функций в признаках.

*Таблица 1.42*

| | | | |
|---|---|---|---|
| $\bar{x}_1 \cdot x_3$ | $f_1 f_3$ | $x_2 \cdot x_3$ | $f_1$ |
| $\bar{x}_2 \cdot x_3$ | $f_3$ | $x_1 \cdot \bar{x}_2$ | $f_2 f_3$ |
| $\bar{x}_1 \cdot x_2$ | $f_3$ | $x_1 \cdot x_2$ | $f_1$ |

Указанные операции склеивания и поглощения повторяются до тех пор, пока это возможно. Затем составляется импликантная таблица (табл. 1.43). Определяется набор импликант, обеспечивающий перекрытие всех столбцов импликантной таблицы (табл. 1.44).

Записываем для выходных функций логические выражения, составленные из этих импликант, в признаках которых содержатся заданные функции:

$$f_1(x_1, x_2, x_3) = x_1 \cdot x_2 \vee \bar{x}_1 \cdot x_3,$$

$$f_2(x_1, x_2, x_3) = x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2 \cdot \bar{x}_3,$$

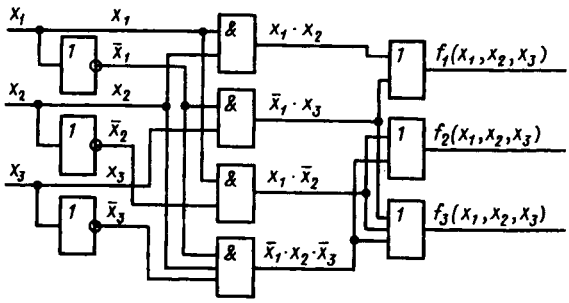
$$f_3(x_1, x_2, x_3) = \bar{x}_1 \cdot x_3 \vee x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2 \cdot \bar{x}_3.$$

Легко убедиться, что выражение для функции $f_3(x_1, x_2, x_3)$ не является минимальным. Минимальная для этой функции форма $f_3(x_1, x_2, x_3) =$

*Таблица 1.43*

| Импликанта | $\bar{x}_1 \cdot \bar{x}_2 \cdot x_3$ | | $\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$ | | $\bar{x}_1 \cdot x_2 \cdot x_3$ | | $x_1 \cdot \bar{x}_2 \cdot \bar{x}_3$ | | $x_1 \cdot \bar{x}_2 \cdot x_3$ | | $x_1 \cdot x_2 \cdot \bar{x}_3$ | $x_1 \cdot x_2 \cdot x_3$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $f_1$ | $f_3$ | $f_2$ | $f_3$ | $f_1$ | $f_3$ | $f_2$ | $f_3$ | $f_2$ | $f_3$ | $f_1$ | $f_1$ |
| $\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$ $(f_2 f_3)$ | | | × | × | | | | | | | | |
| $\bar{x}_1 \cdot x_3$ $(f_1 f_3)$ | × | × | | | × | × | | | | | | |
| $\bar{x}_2 \cdot x_3$ $(f_3)$ | | × | | | | | | | | × | | |
| $\bar{x}_1 \cdot x_2$ $(f_3)$ | | | | × | | × | | | | | | |
| $x_2 \cdot x_3$ $(f_1)$ | | | | | × | | | | | | | × |
| $x_1 \cdot \bar{x}_2$ $(f_2 f_3)$ | | | | | | | × | × | × | × | | |
| $x_1 \cdot x_2$ $(f_1)$ | | | | | | | | | | | × | × |

*Таблица 1.44*




| | |
|---|---|
| $\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$ | $f_2 f_3$ |
| $\bar{x}_1 \cdot x_3$ | $f_1 f_3$ |
| $x_1 \cdot \bar{x}_2$ | $f_2 f_3$ |
| $x_1 \cdot x_2$ | $f_1$ |

Рис. 1.27

$\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$ членом $\bar{x}_1 \cdot x_2$ невыгодна, так как член $\bar{x}_1 \cdot x_2 \cdot \bar{x}_3$ присутствует и в выражении $f_2$ и он должен быть сформирован для этой функции.

На рис. 1.27 приведена функциональная схема устройства, обеспечивающего заданное табл. 1.40 функционирование. Как видно из схемы, ряд элементов участвует в формировании нескольких функций.

### Синтез логических устройств в базисах ИЛИ-НЕ и И-НЕ

Логическое устройство на элементах ИЛИ-НЕ может быть построено при следующей последовательности действий: заданная функция минимизируется с получением МКНФ; производится запись полученного логического выражения через операции ИЛИ-НЕ.

Рассмотрим последовательность синтеза на примере построения логического устройства, реализующего функцию, приведенную в табл. 1.45.

*Таблица 1.45*

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3, x_4)$ | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

Для минимизации функции воспользуемся методам Вейча. В табл. 1.46 приведена карта Вейча для рассматриваемой функции.

Минимальная КНФ функции

$$f(x_1, x_2, x_3, x_4) = (x_1 \vee x_3) \cdot (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \cdot (\bar{x}_1 \vee x_2 \vee \bar{x}_3).$$

Для перехода от базиса И, ИЛИ, НЕ, в котором представлено полученное логическое выражение, к базису ИЛИ-НЕ выполняем следующие действия:
дважды инвертируем правую часть выражения:

$$f(x_1, x_2, x_3, x_4) = \overline{\overline{(x_1 \vee x_3) \vee (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \vee (\bar{x}_1 \vee x_2 \vee \bar{x}_3)}};$$

проводим преобразование по формуле де Моргана:

$$f(x_1, x_2, x_3, x_4) = \overline{(\overline{x_1 \vee x_3}) \vee (\overline{\bar{x}_2 \vee \bar{x}_3 \vee x_4}) \vee (\overline{\bar{x}_1 \vee x_2 \vee \bar{x}_3})};$$

записываем выражение с использованием символа операции ИЛИ-НЕ:

$$f(x_1, x_2, x_3, x_4) = (x_1 \downarrow x_3) \downarrow (\bar{x}_2 \downarrow \bar{x}_3 \downarrow x_4) \downarrow (\bar{x}_1 \downarrow x_2 \downarrow \bar{x}_3). \qquad (1.18)$$

Заметим, что в (1.18) наличие скобок обязательно, иначе исказится функция.

Построенная в соответствии с (1.18) схема логического устройства приведена на рис. 1.28.

Методика синтеза устройства в базисе И-НЕ сходна с рассмотренной выше методикой синтеза в базисе ИЛИ-НЕ. Имеющиеся особенности рассмотрим на примере построения с использованием элементов И-НЕ логического устройства, реализующего функцию, заданную таблицей истинности (табл. 1.45).

Минимизируем функцию. В отличие от синтеза в базисе ИЛИ-НЕ, при котором в процессе минимизации получают МКНФ функции, при синтезе в базисе И-НЕ должна быть получена МДНФ функции. Минимизацию проведем с помощью карты Вейча (табл. 1.47).

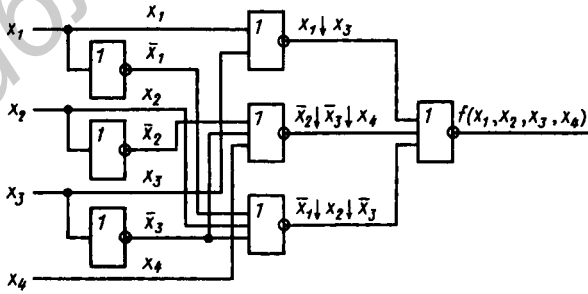



Рис. 1.28

*Таблица 1.46*

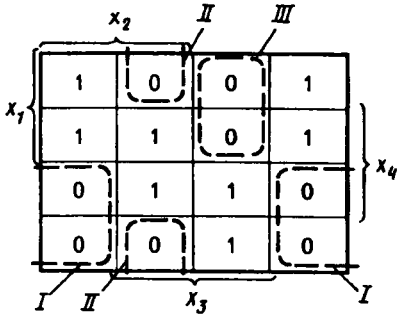
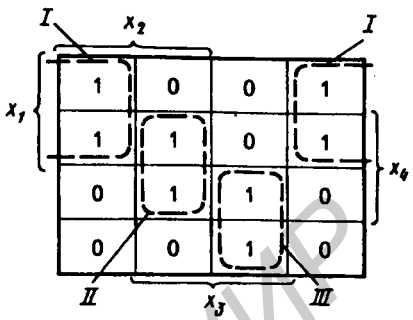



*Таблица 1.47*




Минимальная ДНФ функции

$$f(x_1, x_2, x_3, x_4) = x_1 \cdot \bar{x}_3 \vee x_2 \cdot x_3 \cdot x_4 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot x_3.$$

Дважды инвертируем правую часть выражения:

$$f(x_1, x_2, x_3, x_4) = \overline{\overline{x_1 \cdot \bar{x}_3 \vee x_2 \cdot x_3 \cdot x_4 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot x_3}}. \qquad (1.19)$$

Проводим преобразование по формуле де Моргана:

$$f(x_1, x_2, x_3, x_4) = \overline{(\overline{x_1 \cdot \bar{x}_3}) \cdot (\overline{x_2 \cdot x_3 \cdot x_4}) \cdot (\overline{\bar{x}_1 \cdot \bar{x}_2 \cdot x_3})}.$$

Записываем выражение с использованием символа операции И-НЕ :

$$f(x_1, x_2, x_3, x_4) = (x_1 \mid \bar{x}_3) \mid (x_2 \mid x_3 \mid x_4) \mid (\bar{x}_1 \mid \bar{x}_2 \mid x_3). \qquad (1.20)$$

Выражению (1.20) соответствует схема, приведенная на рис. 1.29.

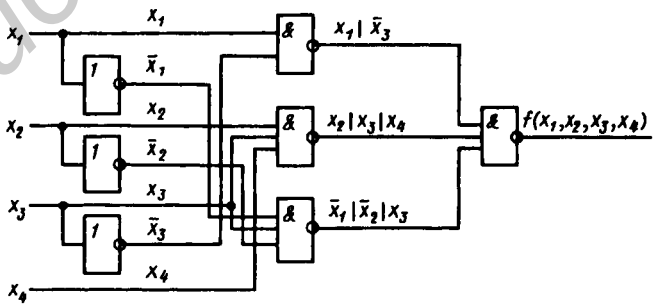



Рис. 1.29

### Некоторые особенности построения схем логических устройств

Построенная схема логического устройства может содержать элементы с разным числом входов. Так, в схеме на рис. 1.29 используются кроме инверторов элементы И-НЕ с двумя и тремя входами. В выпускаемых промышленностью сериях элементов обычно предусматриваются элементы с разным числом входов. Поэтому в большинстве случаев для построения устройств могут быть использованы элементы точно с тем же числом входов, какое требуется в отдельных элементах структурной схемы.

Иногда необходимо иметь в схеме элементы, число входов которых больше или меньше того, которое требуется при рассмотренных выше способах синтеза устройств. Опишем возникающие в этих случаях особенности построения устройств.

Рассмотрим использование элементов, имеющих избыточное число входов. Для определенности примем, что элементы имеют три входа, причем для подачи входных переменных требуется лишь два входа. Избыточный вход мог бы быть оставлен свободным (не подключенным к каким-либо цепям), как это показано на рис. 1.30,*а*. Однако для уменьшения влияния наводимых на этот вход помех нежелательно неиспользуемый вход оставлять свободным. При этом возможны следующие способы его включения. Неиспользуемый вход может быть подключен к любому из используемых входов (рис. 1.30,*б*). Недостаток такого способа соединения состоит в следующем. Объединение входов приводит к тому, что к выходу источника входного сигнала (т.е. к выходу предыдущего элемента, с которого сигнал подается на вход данного элемента) оказывается подключенным большее число входов элемента. Такое увеличение нагрузки вызывает увеличение задержки распространения сигнала, снижение быстродействия элемента. Поэтому наиболее удачным следует считать способ, при котором на неиспользуемый вход подаются логическая константа 0 или 1 (т.е. потенциал, соответствующий *лог.0* либо *лог.1*). Такой способ соединения показан на рис. 1.30,*в*. Здесь на свободные входы элементов ИЛИ и ИЛИ-НЕ подается постоянный потенциал, соответствующий уровню *лог.0*, а для элементов И и И-НЕ — уровню *лог.1*.

Элементы ИЛИ-НЕ и И-НЕ, в которых используется лишь один вход, а остальные входы соединены способом, описанным выше (рис. 1.30,*г*), выполняют операцию НЕ.

Теперь рассмотрим более сложный случай построения устройства на элементах с недостающим числом входов. На рис. 1.31 показан способ реализации трехбуквенного члена логического выражения функции на различных типах элементов с двумя входами.

В логическом устройстве может оказаться несколько членов с числом букв, превышающим число входов элементов. В этом случае для умень-

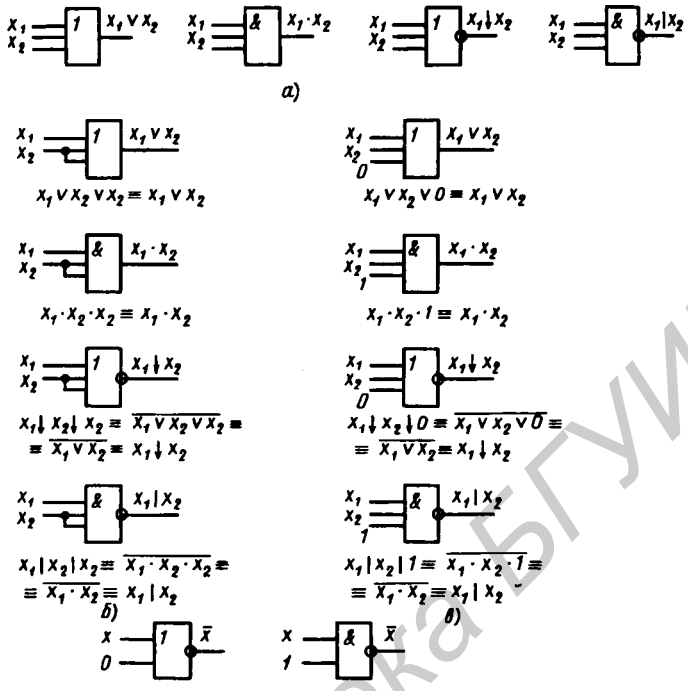



Рис. 1.30

шения числа используемых элементов следует провести соответствующее преобразование групп членов. Этот прием покажем на примере реализации рассмотренной выше логической функции (1.20). Пусть требуется построить устройство, реализующее функцию (1.20) на двухвходовых элементах И-НЕ. Обратимся к (1.19) и сгруппируем два последних члена, вынеся за скобки $x_3$:

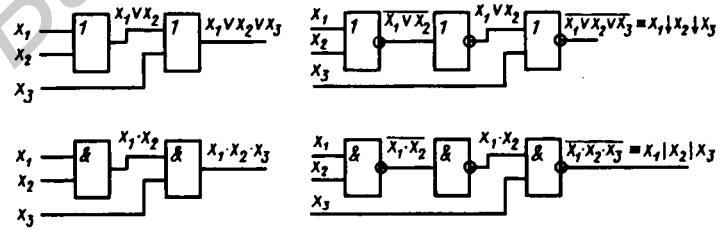



Рис. 1.31

$$f(x_1, x_2, x_3, x_4) = \overline{\overline{x_1 \cdot \bar{x}_3 \vee x_2 \cdot x_3 \cdot x_4 \vee \bar{x}_1 \cdot \bar{x}_2 \cdot x_3}} =$$

$$= \overline{\overline{x_1 \cdot \bar{x}_3 \vee x_3 \cdot (x_2 \cdot x_4 \vee \bar{x}_1 \cdot \bar{x}_2)}}.$$

К полученному выражению применим формулу де Моргана

$$f(x_1, x_2, x_3, x_4) = (\overline{x_1 \cdot \bar{x}_3}) \cdot (\overline{x_3 \cdot (x_2 \cdot x_4 \vee \bar{x}_1 \cdot \bar{x}_2)}).$$

Пользуясь формулой де Моргана, $x_2 \cdot x_4 \vee \bar{x}_1 \cdot \bar{x}_2$ можно преобразовать к виду

$$x_2 \cdot x_4 \vee \bar{x}_1 \cdot \bar{x}_2 = \overline{\overline{(x_2 \cdot x_4)} \cdot \overline{(\bar{x}_1 \cdot \bar{x}_2)}}.$$

После подстановки последнего выражения в предыдущее получим

$$f(x_1, x_2, x_3, x_4) = (\overline{x_1 \cdot \bar{x}_3}) \cdot \overline{(x_3 \cdot \overline{\overline{(x_2 \cdot x_4)} \cdot \overline{(\bar{x}_1 \cdot \bar{x}_2)}})}.$$

Запишем данное выражение через операцию И-НЕ:

$$f(x_1, x_2, x_3, x_4) = (x_1 | \bar{x}_3) | (x_3 | ((x_2 | x_4) | (\bar{x}_1 | \bar{x}_2))).$$

Построенная в соответствии с этим выражением схема приведена на рис. 1.32.

Из сравнения схем на рис. 1.29 и 1.32 видно, что синтез устройств на элементах с уменьшенным числом входов вызвал необходимость применения большего числа элементов.

Однако изложенный прием не всегда применим. Например, им невозможно пользоваться в случаях, когда члены МДНФ не содержат общих букв. При этом необходимое преобразование логического выражения достигается с использованием тождественного соотношения

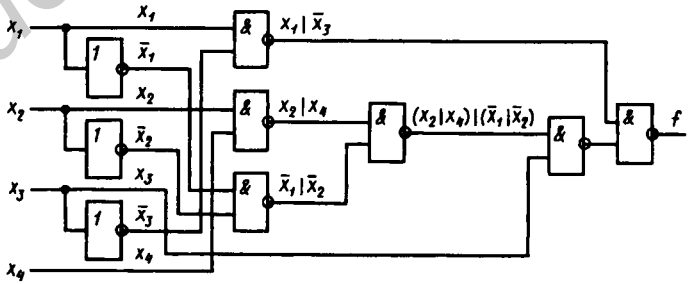



Рис. 1.32

$$a \mid b \mid c = a \mid (\overline{b \mid c}). \qquad (1.21)$$

Покажем справедливость этого тождества:

$$a \mid b \mid c = \overline{a \cdot b \cdot c} = \overline{a \cdot (b \cdot c)} = a \cdot (\overline{b \cdot c}) = a \mid (\overline{b \mid c}).$$

Пусть требуется синтезировать с использованием двухвходовых элементов И-НЕ логическую функцию, МДНФ которой представляется выражением

$$f(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 \vee x_3 \cdot x_4 \vee \bar{x}_1 \cdot \bar{x}_2.$$

Записываем выражение через операцию И-НЕ:

$$f(x_1, x_2, x_3, x_4) = (\overline{x_1 \cdot x_2}) \cdot (\overline{x_3 \cdot x_4}) \cdot (\overline{\bar{x}_1 \cdot \bar{x}_2}) =$$

$$= (x_1 \mid x_2) \mid (x_3 \mid x_4) \mid (\bar{x}_1 \mid \bar{x}_2).$$

Применяя к данному выражению преобразование (1.21), получаем

$$f(x_1, x_2, x_3, x_4) = (x_1 \mid x_2) \mid (\overline{(x_3 \mid x_4) \mid (\bar{x}_1 \mid \bar{x}_2)}).$$

Построенная в соответствии с данным выражением функциональная схема логического устройства приведена на рис. 1.33.

Рассмотрим реализацию на двухвходовых элементах И-НЕ функции

$$f(x_1, x_2, x_3, x_4) = x_3 \cdot x_4 \vee \bar{x}_1 \cdot x_3 \vee x_1 \cdot x_2 \cdot \bar{x}_3 \cdot \bar{x}_4.$$

Перейдем к операции И-НЕ и затем применим соотношение (1.21):

$$f(x_1, x_2, x_3, x_4) = (x_3 \mid x_4) \mid (\bar{x}_1 \mid x_3) \mid (x_1 \mid x_2 \mid \bar{x}_3 \mid \bar{x}_4) =$$

$$= (x_3 \mid x_4) \mid (\bar{x}_1 \mid x_3) \mid (\overline{(x_1 \mid x_2)} \mid (\overline{\bar{x}_3 \mid \bar{x}_4})) =$$

$$= \overline{(x_3 \mid x_4) \mid (\bar{x}_1 \mid x_3)} \mid \overline{(\overline{(x_1 \mid x_2)} \mid (\overline{\bar{x}_3 \mid \bar{x}_4}))}.$$

Аналогичное (1.21) преобразование для случая, когда предполагается реализация логического устройства в базисе ИЛИ-НЕ, имеет вид

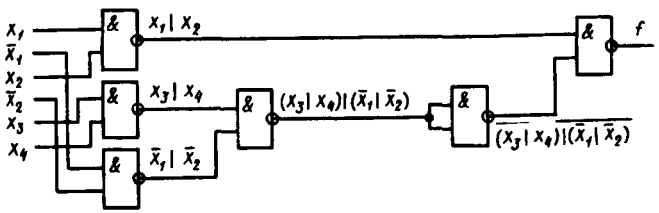



Рис. 1.33

$$a \downarrow b \downarrow c = a \downarrow (\overline{b \downarrow c}). \qquad (1.22)$$

Покажем справедливость этого тождественного соотношения:

$$a \downarrow b \downarrow c = \overline{a \vee b \vee c} = \overline{a \vee (b \vee c)} = a \vee (\overline{\overline{b \vee c}}) = a \downarrow (\overline{b \downarrow c}).$$

### *Упражнения*

1. Для функций $f_1, f_2, f_3$, заданных таблицей истинности (табл. 1.48), найдите МДНФ и МКНФ:

а) методом Квайна,

б) методом Квайна — Мак-Класки, выполнив переход от сокращенной формы к минимальной методом Петрика,

в) методом карт Вейча,

г) методом карт Карно.

Постройте структурные схемы логических устройств в базисе И, ИЛИ, НЕ.

*Таблица 1.48*

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f_1(x_1, x_2, x_3, x_4)$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| $f_2(x_1, x_2, x_3, x_4)$ | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| $f_3(x_1, x_2, x_3, x_4)$ | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

2. Для не полностью заданных функций $f_1, f_2, f_3$ в табл. 1.49 найдите МДНФ и МКНФ:

а) методом Квайна — Мак-Класки,

б) методом карт Карно.

*Таблица 1.49*

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f_1(x_1, x_2, x_3, x_4)$ | | 0 | 1 | | 0 | 0 | 1 | | 1 | | 0 | 1 | | 0 | | 1 |
| $f_2(x_1, x_2, x_3, x_4)$ | 1 | | | 1 | | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| $f_3(x_1, x_2, x_3, x_4)$ | | 1 | | 0 | 1 | | 0 | 1 | 1 | 0 | 0 | | 1 | 0 | 1 | 0 |

3. Синтезируйте в базисе И, ИЛИ, НЕ логическое устройство с тремя выходами. Выходные функции заданы табл. 1.50.

*Таблица 1.50*

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f_1(x_1, x_2, x_3, x_4)$ | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| $f_2(x_1, x_2, x_3, x_4)$ | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| $f_3(x_1, x_2, x_3, x_4)$ | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

4. Синтезируйте заданную табл. 1.51 функцию в базисе ИЛИ-НЕ.

*Таблица 1.51*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_3$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f(x_1, x_2, x_3)$ | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |

5. Синтезируйте заданные табл. 1.52 функции в базисе И-НЕ.

*Таблица 1.52*

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $x_2$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| $x_3$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| $x_4$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $f_1(x_1, x_2, x_3, x_4)$ | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| $f_2(x_1, x_2, x_3, x_4)$ | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| $f_3(x_1, x_2, x_3, x_4)$ | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

6. Синтезируйте заданные табл. 1.52 функции на двухвходовых элементах И-НЕ.

# Глава 2. Арифметические основы цифровой техники

## 2.1. СИСТЕМЫ СЧИСЛЕНИЯ

### Представление чисел в различных системах счисления

Для представления в цифровых устройствах чисел, а также другой информации в процессе программирования наряду с привычной для нас десятичной системой счисления широко используются другие системы. Рассмотрим наиболее употребительные позиционные системы 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:

$$\ldots a_2\, a_1\, a_0, a_{-1}, a_{-2} \ldots$$

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

Цифре разряда приписан вес $p^k$, где $p$ — основание системы счисления; $k$ — номер разряда, равный индексу при обозначениях цифр разрядов. Так, приведенная выше запись означает следующее количество:

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

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 $729{,}324_{10}$ (здесь и далее индекс при числе указывает основание системы счисления, в которой представлено число) означает следующее количество:

$$\begin{array}{ccccccc} 7 & 2 & 9 & , & 3 & 2 & 4_{10} \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ 10^2 & 10^1 & 10^0 & & 10^{-1} & 10^{-2} & 10^{-3} \end{array} = 7 \cdot 10^2 + 2 \cdot 10^1 + 9 \cdot 10^0 + 3 \cdot 10^{-1} + 2 \cdot 10^{-2} + 4 \cdot 10^{-3}.$$

*Весовые коэффициенты разрядов*

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

*В двоичной системе 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:

$$\begin{array}{ccccccccccc} 1 & 1 & 0 & 1 & 1 & , & 1 & 0 & 1_2 & = & (1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow & & +1\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + \\ 2^4 & 2^3 & 2^2 & 2^1 & 2^0 & & 2^{-1} & 2^{-2} & 2^{-3} & & +0\cdot 2^{-2} + 1\cdot 2^{-3})_{10} = 27{,}625_{10}. \end{array}$$

*Весовые коэффициенты разрядов*

*В восьмеричной системе счисления* основание системы счисления $p = 8$. Следовательно, для представления цифр разрядов должно использоваться восемь разных 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 соответствует следующее число:

$$\begin{array}{ccccccccc} 7 & 3 & 5 & , & 4 & 6_8 & = & (7\cdot 8^2 + 3\cdot 8^1 + 5\cdot 8^0 + 4\cdot 8^{-1} + 6\cdot 8^{-2})_{10} = \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & & = 477{,}59375_{10}, \\ 8^2 & 8^1 & 8^0 & & 8^{-1} & 8^{-2} & & \end{array}$$

*Весовые коэффициенты разрядов*

т. е. запись $735{,}46_8$ означает число, содержащее семь раз по $8^2 = 64$, три раза по $8^1 = 8$, пять раз по $8^0 = 1$, четыре раза по $8^{-1} = 1/8$, шесть раз по $8^{-2} = 1/64$.

*В шестнадцатеричной системе 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 символу A в десятичной системе счисления соответствует 10, B — 11, C — 12, D — 13, E — 14, F — 15.

Запись $AB9{,}C2F_{16}$ соответствует следующему числу в десятичной системе счисления:

$$\begin{array}{ccccccccc} A & B & 9 & , & C & 2 & F_{16} & = & (10\cdot 16^2 + 11\cdot 16^1 + 9\cdot 16^0 + \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow & & +12\cdot 16^{-1} + 2\cdot 16^{-2} + 15\cdot 16^{-3})_{10} = \\ 16^2 & 16^1 & 16^0 & & 16^{-1} & 16^{-2} & 16^{-3} & & = 2745{,}7614745\ldots_{10}. \end{array}$$

*Весовые коэффициенты разрядов*

Для хранения $n$-разрядных чисел в цифровой аппаратуре можно использовать устройства, содержащие $n$ элементов, каждый из которых запоминает цифру соответствующего разряда числа. Наиболее просто осуществляется хранение чисел, представленных в двоичной системе счисления. Для запоминания цифры каждого разряда двоичного числа могут использоваться устройства с двумя устойчивыми состояниями (например, триггеры). Одному из этих устойчивых состояний ставится в соответствие цифра 0, другому — цифра 1.

При хранении десятичных чисел каждая цифра десятичного числа представляется в двоичной форме. Такая форма представления чисел называется *двоично-кодированной десятичной системой*. Например, число $765{,}93_{10}$ в двоично-кодированной десятичной системе представляется в следующем виде:

$$765{,}93_{10} = \underbrace{0111}_{7}\ \underbrace{0110}_{6}\ \underbrace{0101}_{5},\ \underbrace{1001}_{9}\ \underbrace{0011}_{3}{}_{2\text{-}10}$$

Следует заметить, что, несмотря на внешнее сходство двоично-кодированного десятичного числа, содержащего в разрядах лишь цифры 0 и 1, с двоичным числом, первое не является двоичным. В этом легко убедиться. Например, если целую часть приведенной выше записи рассматривать как двоичное число, то оно при переводе в десятичную форму означало бы $1893_{10}$, что не совпадает с целой частью исходного числа 765.

Рассмотренный способ двоичного представления (кодирования) десятичных цифр использует так называемый *код 8421* (название кода составлено из весовых коэффициентов разрядов двоичного числа). Наряду с этим кодом при двоичном кодировании десятичных цифр используются различные другие коды, наиболее употребительные из которых приведены в табл. 2.1.

*Таблица 2.1*

| Десятичная цифра | Двоичное кодирование десятичной цифры | | | | | |
|---|---|---|---|---|---|---|
| | *код 8421* | *код 2421* | код 2 из 5 | *код с изб. 3* | *код 3а + 2* | *код 7421* |
| 0 | 0000 | 0000 | 11000 | 0011 | 00010 | 0000 |
| 1 | 0001 | 0001 | 01100 | 0100 | 00101 | 0001 |
| 2 | 0010 | 0010 | 00110 | 0101 | 01000 | 0010 |
| 3 | 0011 | 0011 | 00011 | 0110 | 01011 | 0011 |
| 4 | 0100 | 0100 | 10001 | 0111 | 01110 | 0100 |
| 5 | 0101 | 1011 | 10100 | 1000 | 10001 | 0101 |
| 6 | 0110 | 1100 | 01010 | 1001 | 10100 | 0110 |
| 7 | 0111 | 1101 | 00101 | 1010 | 10111 | 1000 |
| 8 | 1000 | 1110 | 10010 | 1011 | 11010 | 1001 |
| 9 | 1001 | 1111 | 01001 | 1100 | 11101 | 1010 |

*Код 7421* интересен тем, что любая кодовая комбинация содержит не более двух единиц. В *коде 2 из 5* все кодовые комбинации содержат точно две единицы. Это свойство используется для обнаружения ошибочных комбинаций (ошибочное распознавание любого из символов принятой кодовой комбинации изменяет число единиц в этой комбинации).

Пары десятичных цифр, сумма которых равна девяти, составляют цифры, взаимно дополняющие друг друга до девяти (0 и 9, 1 и 8, 2 и 7,...). В *коде 2421* и *коде с избытком 3* кодовая комбинация, соответствующая любой из десятичных цифр, представляет собой инверсию комбинации, соответствующей ее дополнению до девяти. Например, *в коде 2421* паре взаимно дополняющих до девяти цифр 2 и 7 соответствуют комбинации *0010* и *1101*, каждая из которых образуется как инверсия другой. Это свойство упрощает выполнение в цифровых устройствах арифметических операций над десятичными числами. Таким же свойством дополнения до девяти обладает *код 3а + 2*. Кроме того, этот код имеет и другое полезное свойство: любая пара кодовых комбинаций отличается не менее чем а двух разрядах, что позволяет обнаруживать ошибочные комбинации (ошибка, изменяющая цифру одного разряда любой из кодовых комбинаций, приводит к так называемой запрещенной комбинации, не используемой для представления десятичных цифр в этом коде).

## Преобразование 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, представленных в этих системах счисления, в двоичную систему счисления и обратно.

Для перевода чисел из восьмеричной системы счисления в двоичную достаточно каждую цифру восьмеричного числа представить трехразрядным двоичным числом. Например,

$$762{,}35_8 = \underbrace{111}_{7}\ \underbrace{110}_{6}\ \underbrace{010}_{2}\ ,\ \underbrace{011}_{3}\ \underbrace{101}_{5}{}_2.$$

Перевод шестнадцатеричных чисел в двоичную систему счисления достигается представлением цифр шестнадцатеричного числа четырехразрядными двоичными числами. Например,

$$A7B{,}C7_{16} = \underbrace{1010}_{A}\ \underbrace{0111}_{7}\ \underbrace{1011}_{B}{,}\ \underbrace{1100}_{C}\ \underbrace{0111}_{7}{}_2.$$

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 нулями. Затем каждая двоичная группа представляется цифрой той системы счисления, в которую переводится число. Например,

$$\underbrace{001}_{1}\ \underbrace{111}_{7}\ ,\ \underbrace{101}_{5}\ \underbrace{010}_{2}{}_2 = 17{,}52_8\ ;\qquad \underbrace{0101}_{5}\ \underbrace{1100}_{C}\ ,\ \underbrace{1101}_{D}\ \underbrace{0110}_{6}{}_2 = 5C{,}D6_{16}.$$

Бóльшую сложность представляет перевод чисел из десятичной системы в двоичную и обратно. Метод такого перевода зависит от системы счисления, в которой проводятся арифметические операции, необходимые для перевода числа из одной системы счисления в другую. Если перевод осуществляется вручную, то операции будут выполняться в десятичной системе счисления, если цифровым устройством — то в двоичной системе счисления. Рассмотрим эти два случая раздельно.

**Перевод чисел с выполнением операций над десятичными числами.** Так как перевести числа из двоичной системы в шестнадцатеричную и обратно нетрудно, то для простоты выкладок рассмотрим перевод чисел из шестнадцатеричной системы в десятичную и обратно.

В качестве примера перевода числа из шестнадцатеричной системы в десятичную систему выберем число 9A5F, C83B$_{16}$. С учетом весов разрядов шестнадцатеричной системы счисления запишем это число в десятичной системе счисления:

$$9A5F,\ C83B_{16} =$$

$$= \underbrace{(9\cdot 16^3 + \underset{\uparrow}{10}\cdot 16^2 + 5\cdot 16^1 + \underset{\uparrow}{15}\cdot 16^0}_{\text{Целая часть}} + \underbrace{\underset{\uparrow}{12}\cdot 16^{-1} + 8\cdot 16^{-2} + 3\cdot 16^{-3} + \underset{\uparrow}{11}\cdot 16^{-4}}_{\text{Дробная часть}})_{10} =$$

$$\begin{matrix} & \text{A} & & \text{F} & & \text{C} & & & & \text{B} \end{matrix}$$

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

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

Вычисления в приведенном примере дают следующий результат:

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

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

Рассмотрим обратный перевод чисел из десятичной системы счисления в шестнадцатеричную. Воспользуемся приведенным выше примером. Теперь будем считать заданным число $39519,7821502_{10}$ и искать его представление в шестнадцатеричной системе счисления. Преобразуем целую часть числа. Из равенства

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

можно вывести следующее правило получения цифр шестнадцатеричного представления. Деление правой части равенства (т. е. целой части заданного десятичного числа) на 16 дает частное $(9 \cdot 16 + 10) \cdot 16 + 5$ и остаток 15 (т. е. F); деление полученного частного на 16 дает частное $9 \cdot 16 + 10$ и остаток 5; деление последнего частного на 16 приводит 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 |
| ↑ | 5 | 144 | 9 |
| F | | 10 | |
| | | ↑ | |
| | | A | |

Отсюда $39519_{10} = 9A5F_{16}$.

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

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

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:

$$
\begin{array}{lll}
 & & 0,7821502 \\
 & \times & \\
 & & \underline{\qquad 16} \\
C & \leftarrow & 12,5144032 \\
 & \times & \\
 & & \underline{\qquad 16} \\
8 & \leftarrow & 8,2304512 \\
 & \times & \\
 & & \underline{\qquad 16} \\
3 & \leftarrow & 3,6872192 \\
 & \times & \\
 & & \underline{\qquad 16} \\
A & \leftarrow & 10,9955072 \\
 & \times & \\
 & & \underline{\qquad 16} \\
F & \leftarrow & 15,9281152 \\
 & \times & \\
 & & \underline{\qquad 16} \\
E & \leftarrow & 14,8498432 \\
 & & \\
 & & \ldots\ldots\ldots
\end{array}
$$

Таким образом, $07821502_{10} = 0,C83AFE\ldots_{16} \approx 0,C83B_{16}$. 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:

$$937,568_{10} = (9 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0 + 5 \cdot 10^{-1} + 6 \cdot 10^{-2} + 8 \cdot 10^{-3})_{10} =$$

$$= \underbrace{(9 \cdot 10 + 3) \cdot 10 + 7}_{\text{Целая часть}} + \underbrace{10^{-1} \cdot (5 + 10^{-1} \cdot (6 + 10^{-1} \cdot 8))}_{\text{Дробная часть}}.$$

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

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

$$\overbrace{9}\ \overbrace{10}\ \overbrace{3}\ \overbrace{10}\ \overbrace{7}$$

$$0,568_{10} = (((1000 : 1010 + 0110) : 1010 + 0101) : 1010)_2.$$

$$\overbrace{8}\ \overbrace{10}\ \overbrace{6}\ \overbrace{10}\ \overbrace{5}\ \overbrace{10}$$

Получаемые в результате выполнения операций над двоичными числами значения являются двоичными представлениями 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 числа и всех образующихся частных. При этом последнее частное и возникающие при делении остатки являются двоичными представлениями цифр разрядов искомого десятичного представления числа.

Перевод дробной части 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.

## Некоторые правила, облегчающие обращение с двоичными числами

1. Следует на память знать двоичные представления десятичных чисел от 0 до 15 (см. табл. 1.4).

2. Необходимо на память знать десятичные значения чисел $2^k$ от $k = 0$ до $k = 12$ (см. табл. 2.2).

*Таблица 2.2*

| $k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^k$ | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |

3. Следует помнить следующие соотношения:

а) $100\ldots0 = 2^k$, ($k$ разрядов); б) $11\ldots1 = 2^k - 1$, ($k$ разрядов); в) $N\,00\ldots0 = N \cdot 2^k$, ($k$ разрядов).

Эти соотношения в ряде случаев облегчают перевод двоичных чисел в десятичную систему счисления. Например,

г) $1\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0 = (2^6 - 1) \cdot 2^5 + 10 = 63 \cdot 32 + 10 = 2016$,

(6 разрядов; 10; 5 разрядов)

д) $1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0 = 13 \cdot 2^5 = 13 \cdot 32 = 416$,

(13; 5 разрядов)

$1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1 = 13 \cdot 2^5 + 3 = 419$.

(13; 3; 5 разрядов)

4. При чтении дробных двоичных чисел используются соотношения

$$\underbrace{0,0\,0\,\ldots\,0\,1}_{k\text{ разрядов}} = 2^{-k}, \qquad \underbrace{0,1\,1\,\ldots\,1}_{k\text{ разрядов}} = 1 - 2^{-k}$$

Например,

$$\underbrace{0,0\,0\,0\,0\,0\,1}_{6\text{ разрядов}} = 2^{-6} = 1/64, \qquad \underbrace{0,1\,1\,1\,1\,1\,1\,1}_{7\text{ разрядов}} = 1 - 2^{-7} = 1 - 1/128.$$

## Знакоразрядная форма представления двоичных чисел

В этой форме в разрядах числа допускается использование трех символов: 0,1 и $\bar{1}$ (равный –1). Например,

$1\,0\,\bar{1}\,0\,1_2 = 1 \cdot 2^4 - 1 \cdot 2^2 + 1 \cdot 2^0 = 13_{10}$.

Знакоразрядная форма имеет следующие особенности:

а) представление чисел неоднозначно; например, приведенное выше число $13_{10}$ может быть представлено в следующих видах:

$13_{10} = 01101_2 = 1\,0\,0\,\bar{1}\,\bar{1}_2$;

б) исключается необходимость в знаковом разряде:

$-13_{10} = \bar{1}\,0\,1\,0\,\bar{1}_2$

(при изменении знака числа достаточно изменить знак цифр разрядов);

в) возможна минимизация количества ненулевых разрядов.

Преобразование двоичных чисел в знакоразрядную форму с минимальным числом ненулевых разрядов осуществляется следующим образом. На первом этапе все встречающиеся в двоичном числе комбинации вида 01 ... 1 (при числе единиц в комбинации, большем одной) заменяются равнозначными комбинациями вида 10 ... $0\bar{1}$ с тем же числом разрядов. Эти преобразования продолжаются, пока не будут исключены все комбинации указанного вида. На следующем этапе комбинации видов $1\bar{1}$ и $\bar{1}1$ заменяются соответственно комбинациями 01 и $0\bar{1}$. Покажем эти преобразования на примере:

$1\,0\,0\,1\,1\,1\,0\,0\,1\,1,1\,0\,1\,1\;\;0\,1_2$

1-й этап $1\,0\,\overbrace{1\,0\,0\bar{1}}\,0\,\overbrace{1\,0\,0},\overbrace{\bar{1}}\;\overbrace{1\,0\bar{1}}\;0\,1$

2-й этап $1\,0\,1\,0\,0\bar{1}\,0\,1\,0\,0,\overbrace{0\,\bar{1}\,0\bar{1}}\,0\,1$

Как видим, количество ненулевых разрядов, равное в исходном числе 10, сократилось до 7. Сокращение числа ненулевых разрядов может существенно упростить и ускорить операцию умножения, так как сокращается количество суммируемых частичных произведений. Возможно ускорение и операции деления.

### *Упражнения*

Над числами, представленными в вариантах а) — в):

а) $5170{,}236_8$, $A39{,}FB4_{16}$, $4037{,}587_{10}$, $1001111100{,}110111_2$,

б) $6304{,}352_8$, $FBA{,}975_{16}$, $3987{,}654_{10}$, $1011111000{,}101110_2$,

в) $2736{,}503_8$, $EFO{,}B94_{16}$, $3095{,}743_{10}$, $1000111101{,}110111_2$,

проведите следующие операции:

1) преобразование восьмеричных чисел в двоичную систему счисления;

2) преобразование шестнадцатеричных чисел в двоичную систему счисления;

3) преобразование двоичных чисел в восьмеричную систему счисления;

4) преобразование двоичных чисел в шестнадцатеричную систему счисления;

5) преобразование восьмеричных чисел в десятичную систему счисления;

6) преобразование шестнадцатеричных чисел в десятичную систему счисления;

7) преобразование двоичных чисел в десятичную систему счисления;

8) преобразование десятичных чисел в шестнадцатеричную систему счисления;

9) представление двоичных чисел в знакоразрядной двоичной форме с минимальным числом ненулевых разрядов.

## 2.2. ФОРМЫ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ В ЦИФРОВЫХ УСТРОЙСТВАХ

Числа в цифровых устройствах могут представляться в форме целых чисел, чисел с фиксированной запятой и чисел с плавающей запятой.

**Целые числа.** При решении задач целые числа встречаются в случаях представления индексов переменных, подсчета числа повторений каких-либо действий и т. д. Для хранения целых чисел в ячейке памяти предусматривается распределение разрядов (разрядная сетка), показанное на рис. 2,1,*а*. Один из *n* разрядов отводится под знак числа, остальные разряды отводятся под модуль числа.

Обычно применяют следующий способ кодирования знака числа: "+" обозначают цифрой 0 в знаковом разряде, "–" — цифрой 1 в знаковом разряде.

Модуль числа занимает в разрядной сетке ее младшие разряды, свободные старшие разряды заполняются нулями. Например, число $-13_{10}$, представленное в двоичной системе счисления значением $-1101_2$, в 8-разрядной сетке имеет вид, показанный на рис. 2.1,*б*.

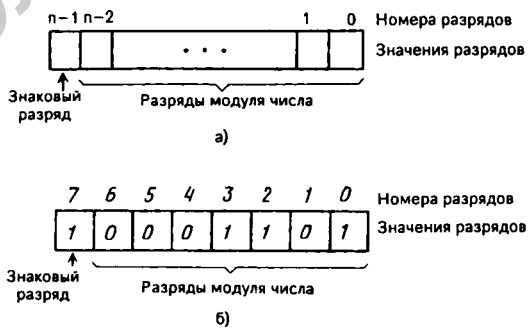



Рис. 2.1

Если количество значащих разрядов модуля числа превышает $n - 1$, происходит потеря старших разрядов модуля. Это явление, называемое *переполнением разрядной сетки*, приводит к ошибке в представлении числа.

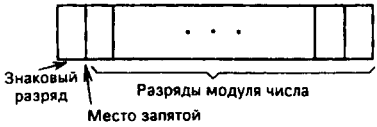



Рис. 2.2

Диапазон модулей чисел, которые могут быть представлены в $n$-разрядной сетке, от 0 (при цифре 0 во всех разрядах модуля) до $2^{n-1} - 1$ (при цифре 1 во всех разрядах модуля).

В универсальных ЭВМ обычно используется два формата целых чисел: *короткий* с числом разрядов $n = 16$ и длинный с $n = 32$. При этом максимальные значения модулей чисел соответственно

$$2^{15} - 1 = 2^5 \cdot 2^{10} - 1 = 32 \cdot 1024 - 1 \approx 32 \cdot 10^3 \text{ (при } n = 16);$$

$$2^{31} - 1 = 2 \cdot 2^{30} - 1 \approx 2 \cdot 10^{30 \cdot 0,3} = 2 \cdot 10^9 \text{ (при } n = 32).$$

**Числа с фиксированной запятой.** При этой форме обычно запятая, отделяющая целую часть числа от ее дробной части, фиксируется перед старшим разрядом модуля числа (рис. 2.2). Таким образом, значение модуля числа всегда оказывается меньше единицы. Это условие путем выбора определенных *масштабных коэффициентов* должно выполняться для исходных данных задачи и всех промежуточных результатов вычислений.

При занесении числа в ячейку памяти свободные младшие разряды заполняются нулями, а если число значащих разрядов модуля больше $n - 1$, то младшие разряды модуля, которые не поместились в разрядной сетке, теряются. Это приводит к погрешности, значение которой меньше единицы младшего разряда разрядной сетки, т. е. $\varepsilon_{абс} < 2^{-(n-1)}$. Так, при $n = 16$ $\varepsilon_{абс} < 2^{-15} = 1/(32 \cdot 10^3)$, при $n = 32$ $\varepsilon_{абс} < 2^{-31} = 1/(2 \cdot 10^9)$.

Если число имеет целую часть, то для ее хранения в разрядной сетке места нет, она теряется, число в разрядной сетке оказывается ошибочным.

Достоинство представления чисел в форме с фиксированной запятой состоит в простоте выполнения арифметических операций; недостатки — в необходимости выбора масштабных коэффициентов и в низкой точности представления чисел с малыми значениями модуля (нули в старших разрядах модуля приводят к уменьшению количества разрядов, занимаемых значащей частью модуля числа).

**Числа с плавающей запятой.** Для научно-технических расчетов необходимо представлять числа в широком диапазоне и с достаточно большой точностью. Указанным требованиям отвечают числа с плавающей запятой (рис. 2.3).

Число состоит из *мантиссы*, старший разряд которой определяет знак числа, и *порядка* со знаком. Значение модуля мантиссы представляется двоичным дробным числом, т. е.

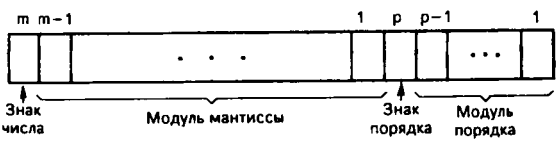



Рис. 2.3

запятая фиксируется перед старшим разрядом модуля мантиссы, порядок представляется целым числом. Порядок указывает действительное положение запятой в числе. Код в приведенном формате представляет значение числа в полулогарифмической форме: $N = M \cdot 2^{П}$, где $M$ и $П$ — мантисса и порядок числа.

Точность представления значений зависит от количества значащих цифр мантиссы. Для повышения точности числа с плавающей запятой представляются в *нормализованной* форме, при которой значение модуля мантиссы лежит в пределах $0{,}5 \le |M| < 1$. Признаком нормализованного числа служит наличие единицы в старшем разряде модуля мантиссы. В нормализованной форме могут быть представлены все числа из некоторого диапазона за исключением нуля.

Нормализованные двоичные числа с плавающей запятой представляют значения модуля чисел в диапазоне

$$0{,}5 \cdot 2^{-П_{\max}} \le |N| \le (1 - 2^{m-1}) 2^{П_{\max}} \approx 2^{П_{\max}},$$

где $П_{\max} = 2^{p-1}$ — максимальное значение модуля порядка.

Так, при $p = 7$ $П_{\max} = 2^{p-1} - 1 = 2^6 - 1 = 63$ и диапазон представления модулей нормализованных чисел

$$|N|_{\min} = 0{,}5 \cdot 2^{-63} = 2^{-64} \approx 10^{-64 \cdot 0{,}3} \approx 10^{-19},$$

$$|N|_{\max} = 2^{-63} \approx 10^{-63 \cdot 0{,}3} \approx 10^{19}.$$

Таким образом, диапазон чисел от $10^{-19}$ до $10^{19}$.

Для расширения диапазона представляемых чисел при фиксированной длине разрядной сетки $(m + p)$ в качестве основания системы счисления выбирается $2^4 = 16$. При этом число, представляемое в разрядной сетке, приобретает значение $N = M \cdot 16^{П}$. Нормализованная мантисса 16-ричного числа с плавающей запятой имеет значение, лежащее в диапазоне $1/16 \le |M| < 1$. Признаком нормализации такого числа является наличие хотя бы одной единицы в четырех старших разрядах модуля мантиссы. Диапазон представления чисел в этом случае существенно расширяется, находясь при том же количестве разрядов порядка в пределах от $10^{-75}$ до $10^{75}$. (Предлагается самостоятельно показать это).

Рассмотрим погрешность представления чисел с плавающей запятой. Абсолютная погрешность представления числа

$$\varepsilon_{абс} = 2^{-(m-1)} \cdot 2^{p}.$$

Предельная относительная погрешность — отношение абсолютной погрешности к числу при минимальном значении модуля мантиссы нормализованного числа:

$$\varepsilon_{отн} = \varepsilon_{абс} / (|M|_{\min} \cdot 2^{p}) = 2^{-(m-1)} \cdot 2^{p} / (0{,}5 \cdot 2^{p}) = 4 \cdot 2^{m}.$$

Отсюда видно, что точность представления чисел определяется количеством разрядов, отводимых в разрядной сетке под мантиссу.

В современных ЭВМ числа с плавающей запятой имеют основание системы счисления 16 и представляются в двух форматах: *коротком* (с числом разрядов 32) и *длинном* (с числом разрядов 64). Длинный формат предусматривает увеличение количества разрядов, отводимых в разрядной сетке под мантиссу, за счет чего повышается точность представления чисел.

**Десятичные числа.** Для кодирования десятичных чисел используются слова переменной длины с применением двух видов формата: *упакованного* и *распакованного*. Каждая десятичная цифра представляется двоичной тетрадой и занимает в разрядной сетке четыре разряда. Четыре разряда отводятся и для представления знака (собственно знак представляется младшим разрядом тетрады, в остальных разрядах тетрады может использоваться постоянная комбинация 110).

При использовании упакованного формата каждый *байт* (8 разрядов двоичного числа) содержит две десятичные цифры (рис. 2.4,*а*).

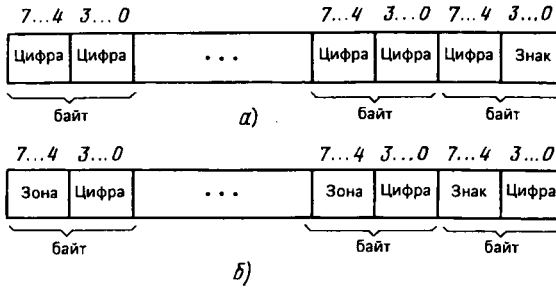



Рис.2.4

Например, число $-6378_{10}$ представляется в упакованном формате в следующем виде:

0000 0110 0011 0111 1000 1101

байт байт байт

В распакованном формате каждый байт содержит лишь одну десятичную цифру в младшей тетраде; старшая тетрада, называемая зоной, заполняется стандартной комбинацией 1111 (рис. 2.4,*б*). Число $-6378_{10}$ представляется в этом формате в следующем виде:

1111 0110 1111 0011 1111 0111 1101 1000

байт байт байт байт

## 2.3. ВЫПОЛНЕНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ

Основной операцией, которая используется в цифровых устройствах при различных вычислениях, является операция алгебраического сложения чисел (сложения, в котором могут участвовать как положительные, 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 запятой, отделяющей целую часть числа от дробной ее части, показывать не будем.

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

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

Цифры разрядов суммы 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 $p = 2$ (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 $N_1 = \mathbf{0}\,831$ и $N_2 = \mathbf{1}\,376$. 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 при выполнении последовательности действий с передачей переносов в старшие разряды (как 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 $N_2 = \mathbf{1}\,376$ в дополнительном коде есть $N_{2доп} = \mathbf{1}\,624$.

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 $N_{2\,доп}$ | + **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 $N = \mathbf{1}\,10110_2$, то $N_{доп} = \mathbf{1}\,01010_2$. Обратное преобразование из дополнительного кода в прямой код производится по тому же правилу.

Рассмотрим примеры выполнения операции сложения.

*Пример 2.1.* Пусть $N_1 = \mathbf{0}\,10110$, $N_2 = \mathbf{1}\,01101$.

| | |
|---|---|
| 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 указывалось выше, перенос, возникающий из знакового разряда, отбрасывается.

*Пример 2.2.* Изменим на обратный знаки слагаемых (по отношению к предыдущему примеру): $N_1 = \mathbf{1}\,10110$, $N_2 = \mathbf{0}\,01101$. Очевидно, ожидаемый ответ $N = N_1 + N_2 = \mathbf{1}\,01001$.

| | |
|---|---|
| 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.** Вначале рассмотрим 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 2.3.* 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 2.4.* 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 разряд, уменьшает сумму не на 10, а на 16 единиц. Уход из суммы шести лишних единиц компенсируется прибавлением шести единиц в ходе коррекции.

*Пример 2.5.* 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 | | 0 0 1 1 |

В данном примере суммирование десятичных цифр в *коде 8421* приводит к числу $1101_2$ ($13_{10}$). Так как сумма больше десяти, то необходимо передать перенос в следующий десятичный разряд, а сумму скорректировать, прибавив к ней шесть единиц. В процессе коррекции возникает 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 представления десятичных цифр правила суммирования отличаются от приведенных выше.

При суммировании многоразрядных десятичных чисел отрицательные числа должны быть предварительно представлены в дополнительном коде. Дополнительный код отрицательного десятичного числа получается путем замены цифр разрядов (кроме знакового) их дополнениями до 9 с прибавлением затем в младший разряд единицы.

*Пример 2.6.* Сложить числа $N_1 = 836$ и $N_2 = -298$.

| | |
|---|---|
| Переносы | 1 |
| $N_1$ | 0 8 3 6 |
| $N_{2доп}$ | + 1 7 0 2 |
| $N = N_1 + N_2$ | 0 5 3 8 |

*Пример 2.7.* Сложить числа $N_1 = -836$ и $N_2 = 298$.

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

Рассмотрим, как можно получить дополнение до 9 (обозначим $a_i'$) к десятичной цифре $a_i$:

$a_i = 9 - a_i = 9 + 6 - (a_i + 6) = 15 - (a_i + 6)$.

Отсюда следует, что дополнение до 9 может быть получено прибавлением 6 и затем нахождением дополнения до 15 для этой суммы. При представлении десятичных цифр четырехразрядным *кодом 8421* дополнение до 15 получается простым инвертированием цифр разрядов двоичного кода. Например, пусть $A = -256$ представляется в двоично-десятичном *коде 8421*: $A_{2\text{-}10} = 1\ 0010\ 0101\ 0110$.

Сначала во все тетрады добавляется $0110_2$ (6):

$$\begin{array}{r} 1\ 0010\ 0101\ 0110 \\ + \\ \quad 0110\ 0110\ 0110 \\ \hline 1\ 1000\ 1011\ 1100 \end{array}$$

После инвертирования цифр разрядов (кроме знакового) получаем обратный код

$$A_{\text{обр}} = 1\ \underbrace{0111}_{7}\ \underbrace{0100}_{4}\ \underbrace{0011}_{3}.$$

Легко убедиться, что каждая тетрада в $A_{\text{обр}}$ представляет собой дополнение до 9 к тетрадам прямого кода $A$.

Для получения дополнительного кода достаточно к младшему разряду $A_{\text{обр}}$ прибавить единицу:

$A_{\text{доп}} = 1\ 0111\ 0100\ 0100$.

При использовании *кода с изб. 3* и *кода 2421* дополнение до 9 получается простым инвертированием цифр тетрады, изображающей десятичную цифру в прямом коде (без таких дополнительных действий, как прибавление $0110_2$ в *коде 8421*), что составляет достоинство этих кодов.

**Умножение двоичных чисел**. Операция умножения чисел включает определение знака и абсолютного значения 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.

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

Пусть производится умножение чисел $1101_2$ и $1011_2$.

$$\begin{array}{rl} & 1101 \quad \text{множимое} \\ \times & \\ & 1011 \quad \text{множитель} \\ \hline & 1101 \quad \text{1-е частичное произведение} \\ & 1101 \quad \text{2-е частичное произведение} \\ + & \\ & 0000 \quad \text{3-е частичное произведение} \\ & 1101 \quad \text{4-е частичное произведение} \\ \hline & 10001111 \quad \text{произведение} \end{array}$$

Как видно из примера, при выполнении умножения формируются частичные произведения (произведения множимого на цифры разрядов множителя), которые суммируются с соответствующими сдвигами относительно друг 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 так далее, пока не будут просуммированы 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 $1011_2$).

```
    1101     4-е частичное произведение
   11010     сдвиг на один разряд влево
 +
    0000     3-е частичное произведение
  -------
   11010     сумма 4- и 3-го частичных произведений
  110100     сдвиг на один разряд влево
 +
    1101     2-е частичное произведение
 ---------
 1000001     сумма 4-, 3- и 2-го частичных произведений

10000010     сдвиг на один разряд влево
+
    1101     1-е частичное произведение
 ---------
10001111     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 подробнее выполнение данной операции.

На рис. 2.5,*а* показана упрощенная схема множительного устройства. Здесь R1, R2, R3 — узлы, называемые *регистрами*. Каждый из них имеет некоторое число разрядов, которые могут устанавливаться в одно из двух состояний: 0 или 1. Таким образом, устанавливая разряды регистра в определенные состояния, можно обеспечить хранение в нем одного многоразрядного двоичного числа (более подробно о регистрах см. в § 2.7). На рисунке См — сумматор, суммирующий многоразрядные числа, поступающие на два его входа; Сч — счетчик.

В регистр R1 поместим $n$- разрядное множимое, в регистр R2 – $n$-разрядный множитель. В $(n + 1)$-разрядном регистре R3 будем накапливать сумму частичных произведений. Перед прибавлением к содержимому регистра R3 очередного частичного произведения хранящаяся в нем сумма предыдущих частичных произведений сдвигается вправо, и если очередное частичное произведение равно множимому, то См суммирует содержимое регистров R3 и R1, полученная сумма помещается в регистр R3. Таким образом к сумме предыдущих частичных произведений прибавляется очередное частичное произведение.

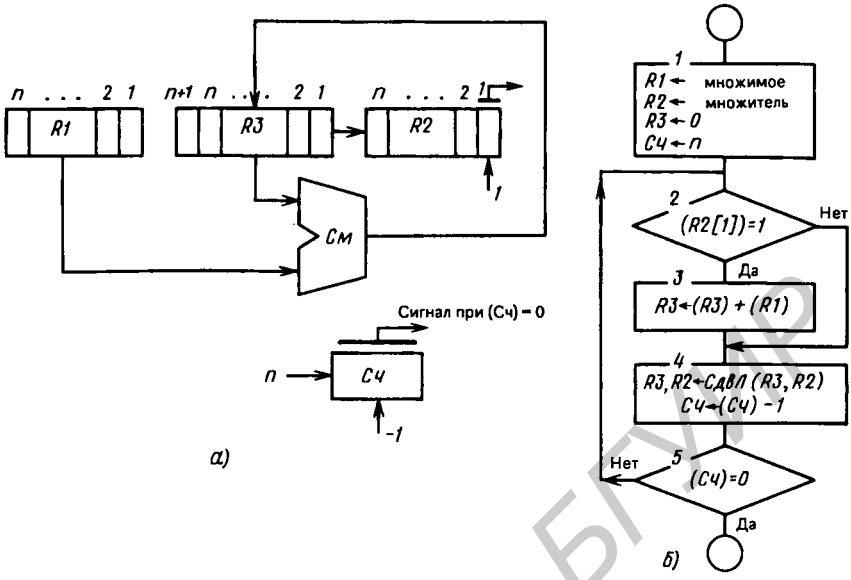



Рис. 2.5

Одновременно со сдвигом содержимого регистра R3 сдвигается вправо и содержимое регистра R2. При каждом таком сдвиге в младший разряд регистра R2 поступает очередной разряд множителя, по которому определяется значение очередного частичного произведения (равное нулю или множимому). В освободившийся в результате сдвига старший разряд регистра R2 можно принять выдвигаемый при сдвиге из регистра R3 младший разряд имеющегося в нем числа.

Подобные вычислительные процессы удобно описывать в форме так называемых схем алгоритмов. На рис. 2.5,*б* приведена схема алгоритма умножения. В блоке 1 показано исходное состояние регистров R1, R2; в регистре R3 устанавливается нулевое значение, в Сч помещается число *n*.

При построении схем алгоритмов для обозначения содержимого регистра наименование регистра заключается в круглые скобки (например, обозначение (R1) означает "содержимое регистра R1"); для указания некоторого разряда регистра вслед за наименованием регистра в прямых скобках записывается номер разряда регистра (например, запись R2 [1] означает "1-й разряд регистра R2", (R2 [1]) — "содержимое 1-го разряда регистра R2").

Процесс умножения носит циклический характер. В блоке *2* проверяется значение младшего разряда числа в регистре R2 (R2 [1]); если в нем содержится 1, то в блоке 3 к содержимому регистра R3 прибавляется множимое, после чего производится сдвиг вправо содержимого регистров R3 и R2; если в R2 [1] обнаруживается 0, частичное произведение равно нулю и блок суммирования обходится.

После *n*-кратного повторения этой группы действий в регистре R3 образуется группа старших разрядов произведения, в регистре R2 — группа младших разрядов произведения. Для подсчета числа повторений этих действий предусмотрен счетчик Сч,

в который предварительно (в блоке 1) помещается число $n$; далее после каждого повторения цикла из содержимого счетчика вычитается единица и проверяется, не равно ли нулю число в счетчике. В случае, если число в счетчике не достигло нулевого значения, производится повторение действий в цикле, при нулевом значении числа в счетчике происходит прекращение действий (выход из цикла).

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

```
  0,1101                  1-е частичное произведение
  0,0110|1                сдвиг на один разряд вправо
+       |
 .0,1101|                 2-е частичное произведение
 ---------+
  1,0011|1                сумма 1- и 2-го частичных произведений
  0,1001|11               сдвиг на один разряд вправо
+       |
  0,0000|                 3-е частичное произведение
 ---------+
  0,1001|11               сумма 1-, 2, и 3-го частичных произведений
  0,0100|111              сдвиг на один разряд вправо
+       |
  0,1101|                 4-е частичное произведение
 ---------+
  1,0001|111              сумма частичных произведений
  0,1000|1111             сдвиг вправо, произведение
```

Если требуется сохранить 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).

При рассмотренном выше способе умножения выполнялись отделение от сомножителей их знаковых разрядов и раздельные действия над знаками и модулями чисел. Одним из эффективных алгоритмов умножения является *алгоритм Бута*. Он не предусматривает раздельных операций над знаковыми разрядами и модулями сомножителей. По этому алгоритму в результате образуется произведение со знаковым разрядом.

На рис. 2.6,*а* показана упрощенная схема устройства, выполняющего умножение с использованием алгоритма Бута. В регистры R1 и R2 предварительно помещаются множимое и множитель. В регистре R3 формируются старшие разряды произведения. В процессе выполнения операции по мере сдвига содержимого R2 вправо освобождающиеся старшие разряды заполняются младшими разрядами произведения; T1 и T2 — триггеры, хранящие одноразрядные двоичные числа (0 или 1). Счетчик предназначен для подсчета числа повторений цикла.

Процесс выполнения операции умножения на рис. 2.6,*б* представлен в форме схемы алгоритма. В блоке 1 в регистры R1 и R2 принимаются множимое и множитель, регистр R3, триггеры T2 и T1 устанавливаются в нулевое состояние, в счетчик заносится $n$ — число

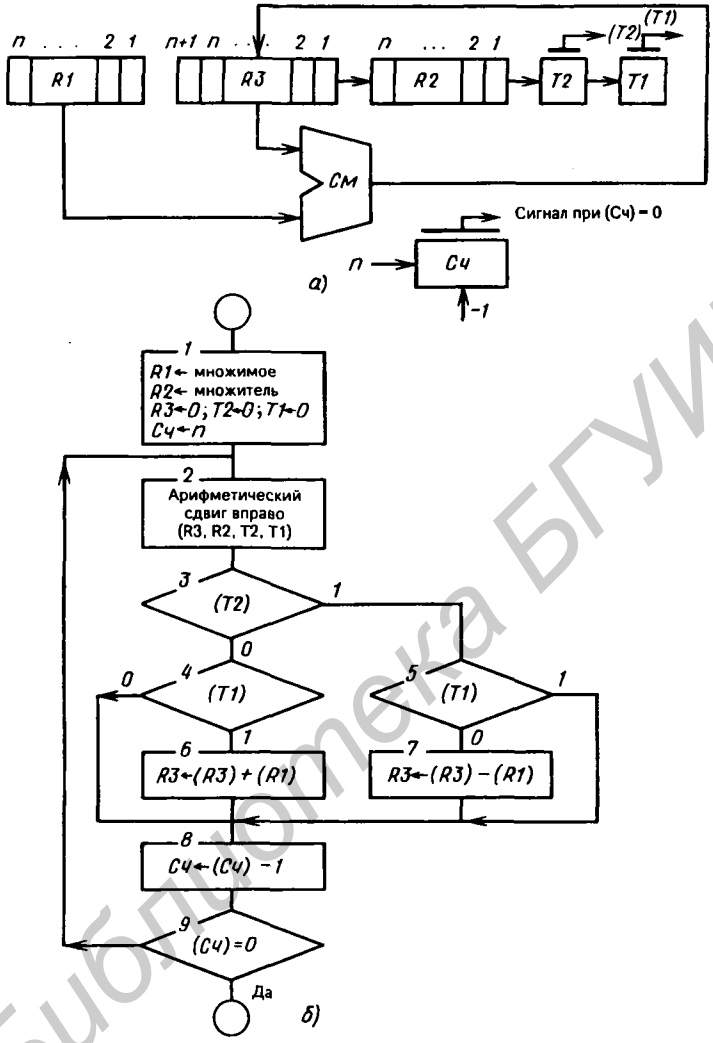



Рис. 2.6

разрядов в сомножителях (равное числу повторений цикла). В блоке 2 производится арифметический сдвиг содержимого регистров R3, R2 (при арифметическом сдвиге вправо в регистре R3 сохраняется неизменным содержимое знакового разряда), рассматриваемого как единое число, в котором содержимое R3 образует его старшие разряды, содержимое R2 — младшие разряды. Значение выдвигаемого из регистра младшего разряда принимается в T2, хранившееся в T2 значение передается в T1. Таким образом, в T2 и T1 образуется последняя выдвинутая из R2 пара разрядов множителя. В блоках 3 — 7 в зависимости от значений, образующихся в T2 и T1, выполняются действия, показанные в таблице на с.92.

| (T2) | (T1) | Выполняемые действия |
|---|---|---|
| 0 | 0 | — |
| 0 | 1 | Суммирование R3 ← (R3) + (R1) |
| 1 | 0 | Вычитание R3 ← (R3) – (R1) |
| 1 | 1 | — |

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 R3 образуется знаковый разряд и группа старших разрядов произведения, а в ($n$ – 1) разрядах R2 — группа младших разрядов произведения.

Следует отметить, что отрицательные сомножители должны представляться в дополнительном коде, в этом же коде оказывается представленным произведение.

Рассмотрим пример умножения чисел **0** $1011_2$ ($11_{10}$) и **1** $1101_2$ ($-13_{10}$). Дополнительный код второго числа **1** 0011. Проводимые при умножении действия иллюстрируются табл. 2.3.

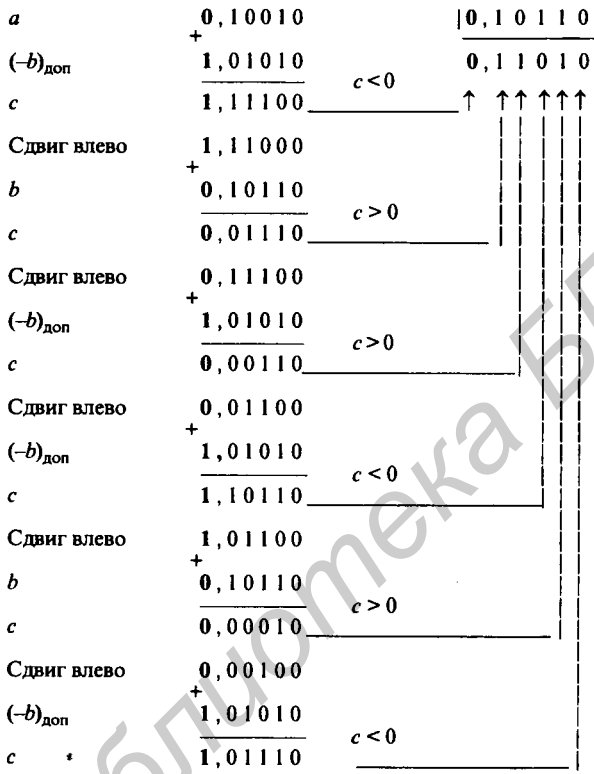
**Деление двоичных чисел.** При алгебраическом делении чисел выполняются действия по определению знака частного и действия по определению модуля частного. Знак частного может быть найден тем же способом, что и знак произведения в рассмотрен-

*Таблица 2.3*

| Множимое (R1) | Старшие разряды произведения (R3) | Множитель и младшие разряды произведения (R2) | (T2) | (T1) | Выполняемое действие |
|---|---|---|---|---|---|
| 01011 | 00000 | 10011 | 0 | 0 | Исходное состояние |
| | 00000 | 01001 | 1 | 0 | Арифметический сдвиг вправо |
| | + 10101 | | | | |
| | 10101 | | | | Вычитание |
| | 11010 | 10100 | 1 | 1 | Арифметический сдвиг вправо |
| | 11101 | 01010 | 0 | 1 | Арифметический сдвиг вправо |
| | + 01011 | | | | |
| | 01000 | | | | Сложение |
| | 00100 | 00101 | 0 | 0 | Арифметический сдвиг вправо |
| | + 10101 | | | | |
| | 10111 | 00010 | | | Вычитание |
| | Произведение в дополнительном коде | | | | |

ной выше операции умножения с отделением знаковых разрядов. Поэтому опишем далее лишь нахождение модуля частного.

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 $b$ заменим прибавлением $-b$, представленного в дополнительном коде: $(-b)_{доп} = 1{,}01010$




**Умножение десятичных чисел**. Рассмотрим процесс умножения на примере. Пусть перемножаются числа $A = -75$ и $B = 23$. Определив отдельно знак произведения так, как это производилось при умножении двоичных чисел (сложением по модулю 2 знаковых разрядов сомножителей), будем затем умножать модули чисел $|A| = 75$ и $|B| = 23$:

| | | |
|---|---|---|
| $\|A\|$ | 75 | множимое |
| $\|B\|$ | × 23 | множитель |
| | 225 | 1-е частичное произведение |
| | + 150 | 2-е частичное произведение |
| | 1725 | произведение |

Процессы в множительном устройстве, связанные с выполнением этих действий, представлены в табл. 2.4.

*Таблица 2.4*

| (R1) | (R3) | (R2) | Выполняемое действие |
|---|---|---|---|
| 75 | 000 | 23 | Исходное состояние |
| | + 75 = 075 | − 1 = 22 | |
| | + 75 = 150 | − 1 = 21 | |
| | + 75 = 225 | − 1 = 20 | |
| | 022 | 52 | Сдвиг вправо |
| | + 75 = 097 | − 1 = 51 | |
| | + 75 = 172 | − 1 = 50 | |
| | 017 | 25 | Сдвиг вправо |
| | Старшие | Младшие | |
| | разряды произведения | | |

Пусть в регистрах R1 и R2 содержатся соответственно модули множимого ($|A| = 75$) и множителя ($|B| = 23$). Для накопления суммы частичных произведений используем регистр R3 с числом десятичных разрядов, на единицу большим числа разрядов в регистрах R1 и R2.

Для получения 1-го частичного произведения — произведения множимого на 1-й разряд множителя ($75 \cdot 3 = 225$) будем суммировать множимое столько раз, сколько единиц содержится в 1-м разряде множителя (в таблице выделен полужирным шрифтом). Это можно выполнить следующим образом. Предварительно установим в регистре R3 нулевое значение. Далее будем анализировать значение 1-го разряда регистра множителя R2. Если значение цифры в этом разряде не равно нулю, прибавим множимое к содержимому регистра R3 и вычтем единицу из содержимого 1-го разряда регистра R2. Эти действия будем повторять до тех пор, пока в 1-м разряде регистра R2 не окажется нуль. К этому моменту в регистре R3 будет сформировано 1-е частичное произведение. Далее сдвинем содержимое регистров R3 и R2 на один десятичный разряд вправо так, чтобы выдвигаемая из регистра R3 десятичная цифра заняла освобождающийся в регистре R2 старший разряд.

Затем аналогично описанному в регистре R3 производятся накопление следующего частичного произведения и сдвиг вправо содержимого регистров R3 и R2. В регистре R3 образуются старшие разряды произведения, в регистре R2 — младшие разряды произведения.

*Таблица 2.5*

| (R1) | (R3) | (R2) | Выполняемое действие |
|---|---|---|---|
| 0111 0101 | 0000 0000 0000 | 0010 **0011** | Исходное состояние |
| | + | – | |
| | 0111 0101 | 1 | |
| | 0000 0111 0101 | 0010 **0010** | |
| | + | – | |
| | 0111 0101 | 1 | |
| | 0000 1110 1010 | 0010 **0001** | |
| | + | | |
| | 0110 1010 | | Коррекция |
| | 0001 0101 0000 | 0010 **0001** | |
| | + | – | |
| | 0111 0101 | 1 | |
| | 0001 1100 0101 | 0010 **0000** | |
| | + | | |
| | 0110 | | Коррекция |
| | 0010 0010 0101 | 0010 **0000** | |
| | 0000 0010 0010 | 0101 **0010** | Сдвиг вправо |
| | + | – | |
| | 0111 0101 | 1 | |
| | 0000 1001 0111 | 0101 **0001** | |
| | + | – | |
| | 0111 0101 | 1 | |
| | 0001 0000 1100 | 0101 **0000** | |
| | + | | |
| | 0110 0110 | | Коррекция |
| | 0001 0111 0010 | 0101 **0000** | |
| | 0000 0001 0111 | 0010 0101 | Сдвиг вправо |
| | Старшие разряды произведения | Младшие | |



В табл. 2.5 показан тот же процесс умножения с представлением десятичных чисел в *коде 8421* (т. е. с представлением чисел в двоично-кодированной десятичной системе счисления). Именно такая форма обычно используется в цифровых устройствах. Особенность показанных здесь действий состоит в том, что при суммировании в тетрадах регистра R3 могут возникать значения, превосходящие $1001_2$ ($9_{10}$). В этих случаях необходима коррекция с прибавлением в эти тетрады числа $0110_2$ ($6_{10}$). Кроме того, сдвигу на один десятичный разряд соответствует сдвиг на одну тетраду (на четыре двоичных разряда).

**Деление десятичных чисел.** Пусть участвующие в операции деления числа представлены в форме с фиксированной запятой и модуль делимого меньше модуля делителя (в противном случае результат операции — частное — окажется представленным с перепол-

нением разрядной сетки). При делении знак частного определяется так же, как и при умножении — суммированием по модулю 2 значений знаковых разрядов делимого и делителя. Модуль частного находится делением модулей делимого и делителя. Эту операцию деления модулей чисел и будем рассматривать далее.

Делимое сдвигается на один десятичный разряд влево. Из него вычитается делитель. Если получающийся остаток положителен, в младший разряд регистра частного прибавляется единица. Эти действия (вычитание делителя и прибавление в разряд частного единицы) продолжаются до изменения знака остатка (до получения отрицательного остатка).

С появлением отрицательного остатка производится сдвиг влево остатка и содержимого регистра частного, после чего в младший разряд регистра частного записывается цифра 9. Далее выполняется серия прибавлений делителя к образующимся остаткам и вычитания из младшего разряда регистра частного единицы. Эти действия продолжаются до тех пор, пока не произойдет очередное изменение знака остатка (остаток не станет положительным). При этом после сдвига остатка и содержимого регистра частного повторяется серия вычитаний делителя из остатков с прибавлением единицы к содержимому младшего разряда регистра частного. Действия продолжаются до образования отрицательного остатка и т. д.

Проиллюстрируем правило деления на примере получения частного от деления чисел $A = 0{,}159$, $B = 0{,}565$. Для лучшего понимания действий, связанных с выполнением деления, используем обычное представление чисел в десятичной системе счисления.

| | | |
|---|---|---|
| Сдвиг влево | 1,59 | частное |
| Вычитание | – 0,565 | |
| Остаток | 1,025 | 0,001 |
| Вычитание | – 0,565 | |
| Остаток | 0,460 | 0,002 |
| Вычитание | – 0,565 | |
| Остаток | –0,105 | |
| Сдвиг влево | –1,050 | 0,029 |
| Сложение | + 0,565 | |
| Остаток | –0,485 | 0,028 |
| Сложение | + 0,565 | |
| Остаток | 0,080 | |
| Сдвиг влево | 0,800 | 0,280 |
| Вычитание | – 0,565 | |
| Остаток | 0,235 | 0,281 |

### *Упражнения*

Выполните арифметические операции, используя в качестве операндов числа $A$ и $B$, представленные ниже в таблице.

| Номер варианта | $A$ | $B$ |
|---|---|---|
| 1 | + 0,0101111 | + 0,1100110 |
| 2 | – 0,0111101 | + 0,1101101 |
| 3 | + 0,0100101 | – 0,1010111 |
| 4 | – 0,0110011 | – 0,1001101 |

1. Найдите сумму $A + B$.

2. Найдите разность $A - B$.

3. Вычислите произведение $A \cdot B$, используя для представления мантисс операндов: а) прямой код, б) дополнительный код.

4. Вычислите частное $A / B$.

# Глава 3. Цифровые устройства

## 3.1. ТРИГГЕРЫ

### Общие сведения

**Активные и пассивные логические уровни элементов И-НЕ и ИЛИ-НЕ.** Интегральные триггеры обычно реализуются на логических элементах И-НЕ, ИЛИ-НЕ. Обратимся к таблицам истинности функций, реализуемых логическими элементами И-НЕ и ИЛИ-НЕ (табл. 3.1). Каждый из этих элементов характеризуется некоторым логическим уровнем (*лог.0* или *лог.1*), наличие которого на одном из входов полностью определяет логический уровень на выходе. При этом логический уровень на выходе элемента не зависит ни от каких комбинаций логических уровней на других входах этого элемента. Таким логическим уровнем для элемента И-НЕ является *лог.0*, а для элемента ИЛИ-НЕ — *лог.1*.

*Таблица 3.1*

| $x_1$ | $x_2$ | $x_1 \mid x_2$ | $x_1 \downarrow x_2$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

Действительно, если на одном из входов элемента И-НЕ лог.0, то на выходе этого элемента возникает *лог.1* независимо от того, каковы логические уровни на других входах; *лог.1* на одном из входов элемента ИЛИ-НЕ установит на выходе уровень *лог.0*, который не будет зависеть от логических уровней на других входах элемента.

Логический уровень, наличие которого на одном из входов элемента однозначно определяет логический уровень на его выходе независимо от уровней на других входах, будем называть *активным логическим уровнем*. Таким образом, активный логический уровень для элементов И-НЕ — *лог.0*, для элементов ИЛИ-НЕ — *лог.1*.

Так как наличие активного логического уровня на одном из входов элемента определяет уровень на выходе элемента (выходной уровень элемента при этом не зависит от уровней на других входах), можно

говорить, что при этом происходит логическое отключение остальных входов элемента.

Уровни, обратные активным, будем называть *пассивными логическими уровнями*. Пассивным уровнем для элементов И-НЕ является уровень *лог.1*, для элемента ИЛИ-НЕ — *лог.0*. При пассивном логическом уровне на одном из входов элемента уровень на выходе элемента определяется логическими уровнями на других его входах. Пользование понятиями активного и пассивного логических уровней облегчает анализ функционирования триггеров, построенных на элементах И-НЕ или ИЛИ-НЕ.

**Назначение триггера.** Триггер предназначен для хранения значения одной логической переменной (или значения одноразрядного двоичного числа; при хранении многоразрядных двоичных чисел для запоминания значения каждого разряда числа используется отдельный триггер). В соответствии с этим триггер имеет два состояния: одно из них обозначается как состояние 0, другое — как состояние 1. Воздействуя на входы триггера, его устанавливают в нужное состояние.

**Основные обозначения.** Триггер имеет два выхода: *прямой* $Q$ и *инверсный* $\overline{Q}$. Состояние, в котором находится триггер, определяется уровнями напряжения на этих выходах: если напряжение на выходе $Q$ соответствует уровню *лог.0* ($Q = 0$), то принимается, что триггер находится в состоянии 0, при $Q = 1$ триггер находится в состоянии 1. Логический уровень на инверсном выходе $\overline{Q}$ представляет собой инверсию состояния триггера (в состоянии 0 $\overline{Q} = 1$, и наоборот).

Триггеры имеют различные типы входов. Приведем их обозначения и назначения:

$R$ (от англ. Reset) — *раздельный вход установки в состояние 0*;

$S$ (от англ. Set) — *раздельный вход установки в состояние 1*;

$K$ — *вход установки универсального триггера в состояние 0*;

$J$ — *вход установки универсального триггера в состояние 1*;

$T$ — счетный вход;

$D$ (от англ. Delay) — *информационный вход установки триггера в состояние, соответствующее логическому уровню на этом входе*;

$C$ — *управляющий (синхронизирующий) вход*.

Наименование триггера определяется типами его входов. Например, RS-триггер — триггер, имеющий входы типов $R$ и $S$.

По характеру реакции на входные сигналы триггеры делятся на два типа: асинхронные и синхронные. В асинхронном триггере входные сигналы воздействуют на состояние триггера непосредственно с момента их подачи на входы, в синхронных триггерах — только при подаче синхронизирующего сигнала на управляющий вход $C$.

**Типы триггеров**. Рассмотрим общие характеристики основных типов триггеров. Каждый тип триггера характеризуется таблицей переходов (табл. 3.2). Таблица переходов, приведенная в табл. 3.2,*а*, соответствует

работе RS-триггера. Здесь $Q_0$ — текущее состояние триггера (состояние до подачи на вход активного сигнала). При отсутствии на входах $R$ и $S$ активного уровня триггер сохраняет текущее состояние. Активный сигнал $R = 1$ устанавливает триггер в состояние 0, а сигнал $R = 1$ — в состояние 1. Звездочкой в таблице отмечено состояние, соответствующее запрещенной комбинации входных сигналов.

Таблица 3.2,*б* является таблицей переходов JK-триггера. Этот тип триггера отличается от RS-триггера отсутствием запрещенной комбинации входных сигналов, при $J = K = 1$ триггер устанавливается в состояние, противоположное текущему состоянию $Q_0$.

*Таблица 3.2*

| $S$ | $R$ | $Q$ |
|---|---|---|
| 0 | 0 | $Q_0$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | * |

*а)*

| $J$ | $K$ | $Q$ |
|---|---|---|
| 0 | 0 | $Q_0$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{Q}_0$ |

*б)*

| $D$ | $Q$ |
|---|---|
| 0 | 0 |
| 1 | 1 |

*в)*

| $T$ | $Q$ |
|---|---|
| 0 | $Q_0$ |
| 1 | $\overline{Q}_0$ |

*г)*

Таблица 3.2,*в* является таблицей переходов D-триггера. Триггер устанавливается в состояние, соответствующее уровню сигнала на входе $D$.

Таблица 3.2,*г* определяет работу Т-триггера. При входном сигнале $T = 0$ триггер сохраняет текущее состояние $Q_0$, при входном сигнале $T = 1$ триггер переключается в состояние, противоположное текущему.

### Асинхронные триггеры

**RS-триггер с прямыми входами.** Логическая структура триггера представлена на рис. 3.1,*а*. Триггер построен на двух логических элементах ИЛИ-НЕ, связанных таким образом, что выход каждого элемента подключен к одному из входов другого. Такое соединение элементов в устройстве обеспечивает два устойчивых состояния, в чем легко убедиться.

Пусть на входах $R$ и $S$ действуют пассивные для элементов ИЛИ-НЕ уровни *лог.0*, которые не влияют на состояние триггера. В состоянии 0 триггера на выходе элемента А имеем $Q = 0$; это значение подается на вход элемента В; при этом на обоих входах элемента В действует уровень *лог.0*, а на его выходе $\overline{Q} = 1$; с выхода элемента В это значение поступает на вход элемента А, что обеспечивает на его выходе $Q = 0$. Это одно из устойчивых состояний триггера. В состоянии 1 триггера на выходе элемента А имеем $Q = 1$, что обусловливает на выходе

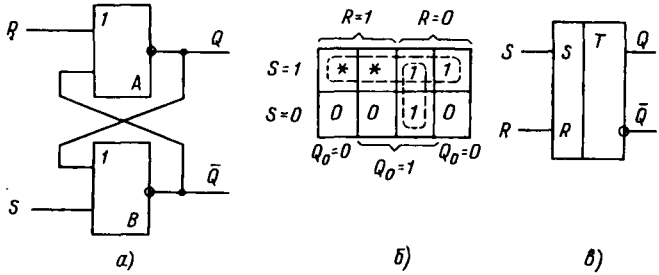



Рис. 3.1

элемента В $\overline{Q} = 0$, при этом на обоих входах элемента А действуют уровни *лог.0*, что обеспечивает на выходе этого элемента уровень *лог.1*. Таким образом, в каждом из состояний триггера элементы А и В оказываются в противоположных состояниях.

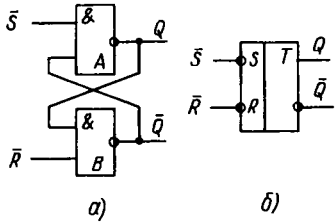
Переключение триггера из одного устойчивого состояния в другое происходит при подаче активных сигналов на входы.

При $R = 1$ элемент А устанавливается в состояние, в котором на его выходе $Q = 0$, следовательно, на инверсном выходе $\overline{Q} = 1$, и таким образом, триггер устанавливается в состояние 0. Если триггер до подачи сигнала $R = 1$ находился в состоянии 0, то его состояние не изменится. Если же триггер находился в состоянии 1, то при $R = 1$ произойдет переключение элемента А и на его выходе установится $Q = 0$; это значение подается на вход элемента В, переключает его и на выходе элемента В устанавливается $\overline{Q} = 1$, после чего триггер оказывается в состоянии 0.

Таким образом, при переключении триггера из одного состояния в другое его элементы последовательно переключаются и время переключения равно удвоенному среднему времени задержки распространения сигнала в логическом элементе ИЛИ-НЕ: $t_п = 2t_з$. Очевидно, чем меньше $t_п$, тем большее число переключений триггера удастся произвести в единицу времени, т.е. будет выше допустимая частота переключений или, иначе говоря, быстродействие триггера.

Процесс установления триггера в состояние 1 при подаче на его вход $S = 1$ аналогичен описанному.

Одновременная подача активных уровней *лог.1* на оба входа $R$ и $S$ не допускается, так как при этом на обоих выходах установится уровень *лог.0*, а после снятия со входов активных логических уровней состояние триггера окажется неопределенным: в силу случайных причин триггер может установиться либо в состояние 0, либо в состояние 1. На рис. 3.1,*б* приведена таблица состояний RS-триггера в форме таблицы Вейча. Из




*а)* *б)*

Рис. 3.2

этой таблицы может быть построено логическое выражение, определяющее функционирование RS-триггера:

$$Q = S \vee \bar{R} \cdot Q_0,$$

т.е. триггер устанавливается в состояние 1, если $S = 1$, либо остается в этом состоянии 1, если $R = 0$ и прежнее состояние триггера $Q_0 = 1$.

На рис. 3.1,*в* показано условное обозначение асинхронного RS-триггера.

**RS-триггер с инверсными входами**. Логическая структура триггера приведена на рис. 3.2,*а*. Отличие от логической структуры рассмотренного выше RS-триггера с прямыми входами состоит лишь в том, что здесь использованы логические элементы И-НЕ.

При этом активным логическим уровнем на входах является *лог.0*, пассивным — *лог.1*. Для того чтобы активными были, как и в предыдущем триггере, входные сигналы $S = 1$ и $R = 1$, будем считать, что на входы подаются инверсии $\bar{S}$ и $\bar{R}$. Тогда при $\bar{S} = 1$ (или $\bar{R} = 1$) полученная $S = 0$ (или $R = 0$) и на входе триггера будет действовать активный уровень *лог.0*. Другое удобство такого обозначения входных величин состоит в том, что триггер с инверсными входами описывается той же таблицей состояний (рис. 3.1,*б*), что и триггер с прямыми входами.

Рассмотрим устойчивые состояния триггера. Пусть на входах действуют пассивные уровни $S = 0$ и $R = 0$ ($\bar{S} = 1$ и $\bar{R} = 1$). В состоянии 0 триггера $Q = 0$ этот уровень передается на вход элемента В и вызывает на его выходе $\bar{Q} = 1$, это значение с выхода элемента В подается на вход элемента А, и так как на обоих входах элемента А уровень *лог.1*, то на выходе элемента $Q = 0$. Аналогично определяется второе устойчивое состояние триггера.

При подаче активного уровня $\bar{S} = 0$ ($S = 1$) на выходе элемента А устанавливается $Q = 1$, на выходе элемента В устанавливается $\bar{Q} = 0$ и триггер оказывается в состоянии 1. При подаче активного уровня $\bar{R} = 0$ ($R = 1$) триггер устанавливается в состояние 0. Как и для триггера с прямыми входами, одновременная подача активных логических уровней на оба входа не допускается.

На рис. 3.2,*б* показано условное обозначение RS-триггера с инверсными входами.

### Синхронные триггеры со статическим управлением

Отличие *синхронного триггера* от асинхронного состоит в том, что синхронный триггер снабжен дополнительным входом, называемым

*синхронизирующим* (этот вход часто называют также *тактирующим входом*). Назначение синхронизирующего входа в том, чтобы сигналом на этом входе разрешать прием сигналов с информационных входов (входов, сигналами на которых производится переключение триггера) в заданные временные интервалы. При отсутствии сигнала на синхронизирующем входе информационные входы логически отключаются и сигналы на этих входах не влияют на состояние триггера.

Достоинство синхронных триггеров заключается в том, что они позволяют устранять влияние различий в значении задержек в распространении сигнала в отдельных элементах схемы. При этом обеспечивается одновременный прием сигналов разными частями схемы в заданные временные отрезки.

Синхронные триггеры, в свою очередь, делятся на два типа: *синхронные триггеры со статическим управлением* и *синхронные триггеры с динамическим управлением*. В первых триггеры реагируют на изменения сигналов на информационных входах, происходящие во время действия сигнала на синхронизирующем входе. Поэтому эти изменения допускаются только при отсутствии сигнала на синхронизирующем входе. В синхронных триггерах с динамическим управлением прием сигналов с информационных входов происходит в течение малой длительности фронта (положительного или отрицательного) сигнала на синхронизирующем входе. В остальное время информационные входы оказываются логически отключенными и допускаются изменения сигналов на информационных входах и в течение действия синхронизирующего сигнала (исключая длительность его фронта).

**RS-триггер.** На рис. 3.3,*а*,*б* показаны логические структуры синхронного RS-триггера. Как видно из этих структур, синхронный RS-триггер состоит из асинхронного триггера с прямыми (либо инверсными) входами, на входах $R$ и $S$ которого включены логические элементы И (И-НЕ). С помощью логических элементов И (И-НЕ) обеспечивается передача активных логических уровней информационных входов $S$ и $R$

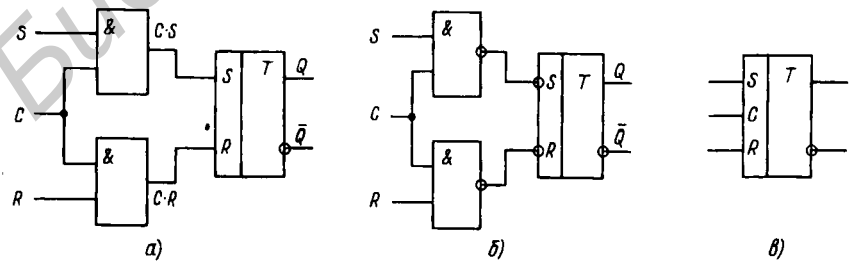



Рис. 3.3

синхронного триггера на входы $S$ и $R$ входящего в его состав асинхронного триггера только при уровне *лог.1* на синхронизирующем входе $C$.

Таким образом, при $C = 0$ на входы асинхронного триггера не передаются активные уровни и триггер сохраняет ранее установленное в нем состояние $Q_0$. При $C = 1$ состояние триггера определяется действующими на входах уровнями так же, как и в рассмотренном выше асинхронном RS-триггере. Следовательно, функционирование синхронного RS-триггера может быть описано логическим выражением

$$Q = \overline{C} \cdot Q_0 \vee C \cdot (S \vee \overline{R} \cdot Q_0).$$

Нормальная работа синхронного RS-триггера требует, чтобы за время действия *лог.1* на синхронизирующем входе $C$ уровни на информационных входах $S$ и $R$ оставались неизменными. Смена уровней на входах допускается лишь в то время, когда $C = 0$ и триггер не реагирует на уровни на входах $S$ и $R$.

На рис. 3.3,*в* показано условное обозначение синхронного RS-триггера.

**D-триггер.** Этот тип триггера имеет лишь один информационный вход $D$. Вход $C$ — управляющий и служит для подачи синхронизирующего сигнала.

Функционирование D-триггера определяется таблицей состояний, приведенной на рис. 3.4,*а*. Как видно из таблицы, при $C = 1$ триггер устанавливается в состояние, определяемое логическим уровнем на входе $D$ (при $C = 0$ он сохраняет ранее установленное состояние $Q_0$). Такое функционирование может быть описано логическим выражением

$$Q = \overline{C} \cdot Q_0 \vee C \cdot D.$$

На рис. 3.4,*б,в* представлены логические структуры D-триггера, состоящего из асинхронного RS-триггера с логическими элементами на входах. При $C = 0$ на выходах элементов И (И-НЕ) образуются пассивные для входов асинхронного RS-триггера уровни. При $C = 1$ уровень,

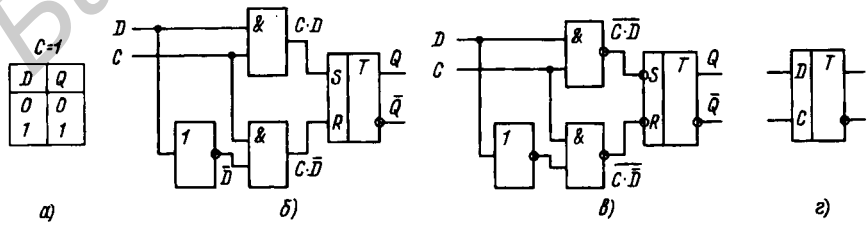



Рис. 3.4

поданный на информационный вход $D$, создает активный уровень либо на входе $R$ (при $D = 0$), либо на входе $S$ (при $D = 1$) асинхронного RS-триггера, и триггер устанавливается в состояние, соответствующее логическому уровню на входе $D$. Таким образом, D-триггер воспринимает информацию со входа $D$ при $C = 1$ и затем может хранить ее неопределенное время, пока $C = 0$.

На рис. 3.4,*г* показано условное изображение D-триггера.

### Синхронные триггеры, построенные по принципу двухступенчатого запоминания информации

Особенность триггеров с двухступенчатым запоминанием информации состоит в том, что они содержат две триггерные структуры: одна из них образует так называемый *ведущий триггер*, другая — *ведомый триггер* (рис. 3.5). Оба триггера функционируют как синхронные триггеры со статическим управлением. Если на синхронизирующем входе $C = 1$, ведущий триггер устанавливается в состояние, соответствующее сигналам, поступающим на информационные входы. Ведомый триггер, имеющий инверсный синхронизирующий вход, при этом невосприимчив к информации, поступающей на его вход с выхода ведущего триггера. Он продолжает находиться в состоянии, в которое был ранее установлен (в предыдущем тактовом периоде).

При изменении значения $C$ (с $C = 1$ на $C = 0$) ведущий триггер отключается от информационных входов и перестает реагировать на изменения значений сигналов на этих входах; ведомый триггер устанавливается в состояние, в котором находится ведущий триггер. С этого момента на выходах устанавливаются значения, соответствующие входным сигналам, поступавшим к моменту рассматриваемого фронта сигнала на синхронизирующем входе.

Таким образом, управление процессами в триггере с двухступенчатым запоминанием информации за время тактового периода осуществляется двумя фронтами сигнала на синхронизирующем входе: на положительном фронте происходит установка ведущего триггера, на

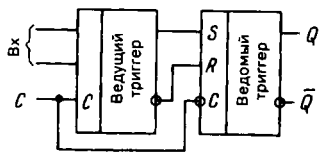



Рис. 3.5

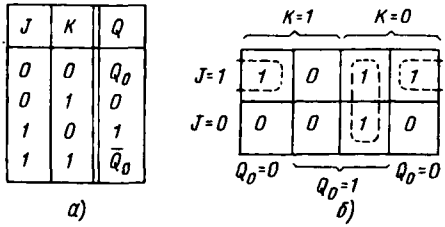



Рис.3.6

отрицательном фронте — ведомого триггера. В качестве примера рассмотрим JK-триггер с двухступенчатым запоминанием информации.

**JK-триггер.** На рис. 3.6,*б* таблица состояний JK-триггера представлена в форме диаграммы Вейча, из которой можно получить следующее логическое выражение, определяющее функционирование JK-триггера:

$$Q = J \cdot \overline{Q}_0 \vee \overline{K} \cdot Q_0. \qquad (3.1)$$

Из (3.1) следует, что состояние $Q$, в которое устанавливается триггер, определяется не только логическими уровнями на информационных входах $J$ и $K$, но и состоянием $Q_0$, в котором ранее находился триггер. Это определяет возможность построения логической структуры JK-триггера с использованием двух RS-триггеров. Один из RS-триггеров (ведомый) предназначен для хранения текущего состояния $Q_0$; снимаемые с его выходов сигналы $Q_0$ и $\overline{Q}_0$ совместно с информационными сигналами входов $J$ и $K$ используются для формирования нового состояния $Q$ в другом RS-триггере (ведущем). JK-триггер с подобной логической структурой представлен на рис. 3.7,*а*. Так как каждый из

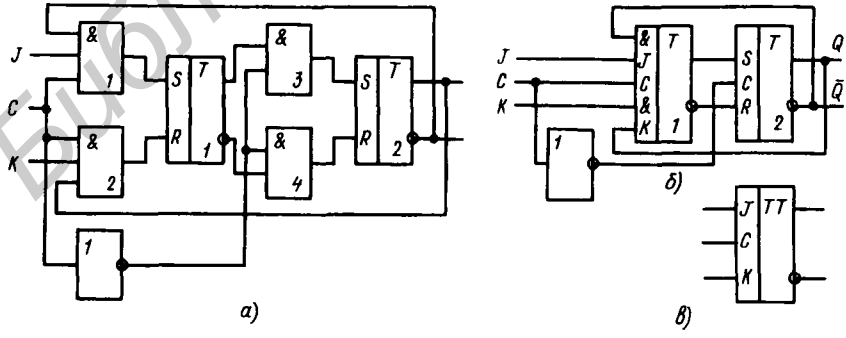



Рис. 3.7

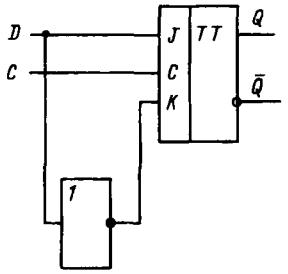



Рис. 3.8

триггеров совместно с элементами И на входах образует схему синхронного RS-триггера, то логическая структура может быть такой, как показано на рис. 3.7,*б*. Здесь ведущий триггер имеет по две пары связанных операцией И входов *J* и *K*.

При уровне *лог.0* на входе *C* триггер *1* не реагирует на сигналы входов *J* и *K*. На синхронизирующий вход триггера *2* при этом подается уровень *лог.1*, и состояние ведущего триггера *1* передается ведомому триггеру *2*. Оба триггера оказываются в одном и том же состоянии. При переходе на входе *C* к уровню *лог.1* на синхронизирующий вход триггера *2* через инвертор подается уровень *лог.0*, и логическая связь между триггерами обрывается. Триггер *1* устанавливается в состояние *Q*, определяемое выражением (3.1). Подача вновь на вход *C* уровня *лог.0* приводит к передаче состояния *Q* из триггера *1* в триггер *2*. Символическое изображение описанного JK-триггера приведено на рис. 3.7,*в*.

На рис. 3.8 показано включение JK-триггера, при котором он выполняет функции D-триггера.

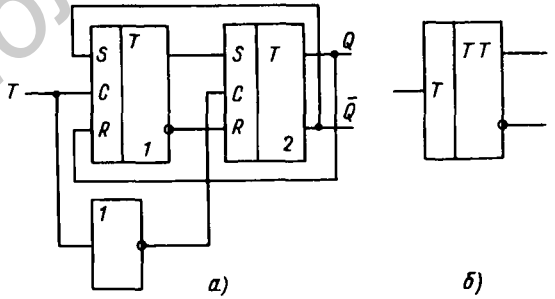



Рис. 3.9

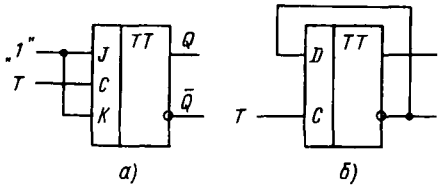



Рис. 3.10

**Т-триггер**. На рис. 3.9,*а* представлена логическая структура Т-триггера. При положительном фронте импульса, поступающего на вход *Т*, ведущий триггер *1* устанавливается в состояние, противоположное состоянию ведомого триггера *2*, при отрицательном фронте входного импульса происходит передача сигнала, соответствующего состоянию триггера *1*, в триггер *2*.

На рис. 3.9,*б* приведено условное изображение Т-триггера. Режим Т-триггера может быть получен с помощью JK-триггера либо D-триггера, как показано на рис. 3.10,*а*,*б*.

### Триггеры с динамическим управлением

В триггерах с динамическим управлением (управлением фронтом синхронизирующего сигнала) процессы, связанные с переключением, происходят в течение короткого времени вблизи фронта сигнала на синхронизирующем входе. Если переключение триггера происходит при положительном фронте сигнала на этом входе, то вход называется *прямым динамическим входом* (условное обозначение прямого динамического входа триггера показано на рис. 3.11,*а*); если при отрицательном — то *инверсным динамическим входом* (условное обозначение инверсного динамического входа триггера приведено на рис. 3.11,*б*).

**D-триггер**. На рис. 3.12,*а* приведена логическая структура D-триггера. Здесь элементы И-НЕ 1 и И-НЕ 2 составляют простейшую выходную

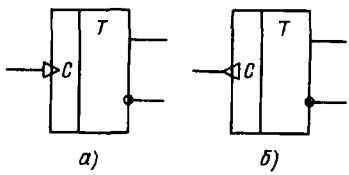



Рис.3.11

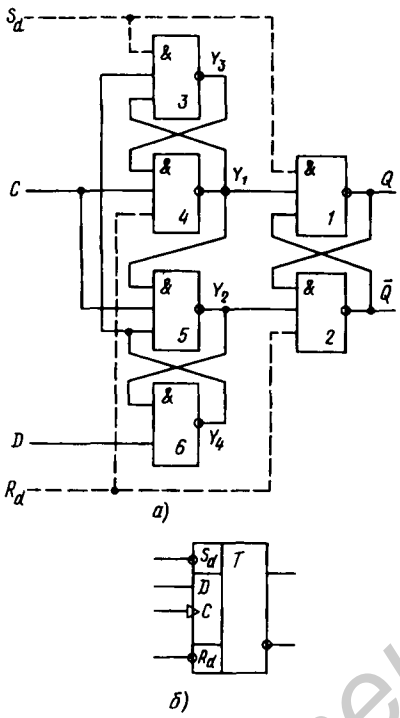



Рис 3.12

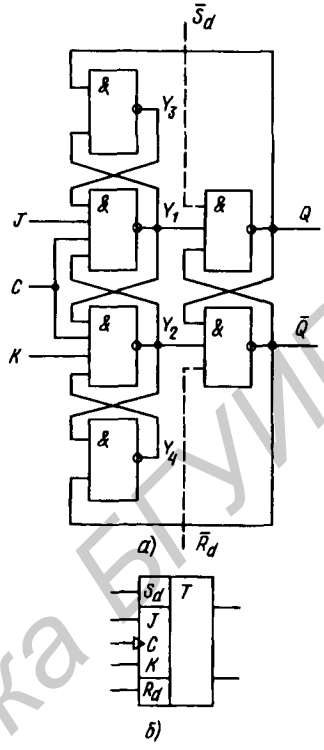



Рис.3.13

триггерную структуру, состояние которой определяет состояние D-триггера. Элементы И-НЕ 3 — 6 образуют схему, формирующую сигналы $Y_1$ и $Y_2$, которыми переключается выходная триггерная структура. На положительном фронте синхронизирующего сигнала на входе $C$ выходная триггерная структура устанавливается в состояние, соответствующее логическому уровню на входе $D$. Убедитесь в этом самостоятельно, задавая различные значения $D$ при разных исходных состояниях триггера. Входы $S_d$ и $R_d$ — установочные, сигналами *лог.0* на этих входах триггер устанавливается в состояние соответственно 1 и 0.

На рис. 3.12,*б* приведено условное обозначение триггера.

**JK-триггер.** Одна из логических схем JK-триггера и его условное обозначение приведены на рис. 3.13. Убедитесь самостоятельно, что при действии положительного фронта сигнала на синхронизирующем входе $C$ триггер устанавливается в состояние 1 при $J = 1$ и в состояние 0 при $K = 1$.

## 3.2. ШИФРАТОРЫ, ДЕШИФРАТОРЫ, ПРЕОБРАЗОВАТЕЛИ КОДОВ

### Шифраторы

*Шифратор* (называемый также *кодером*) осуществляет преобразование десятичных чисел в двоичную систему счисления. Пусть в шифраторе имеется $m$ входов, последовательно пронумерованных десятичными числами (0, 1, 2,...,$m-1$), и $n$ выходов. Подача сигнала на один из входов приводит к появлению на выходах $n$-разрядного двоичного числа, соответствующего номеру возбужденного входа.

Очевидно, трудно строить шифраторы с очень большим числом входов $m$, поэтому они используются для преобразования в двоичную систему счисления относительно небольших десятичных чисел. Преобразование больших десятичных чисел осуществляется методами, приведенными в § 2.1.

Шифраторы широко используются в разнообразных устройствах ввода информации в цифровые системы. Такие устройства могут снабжаться клавиатурой, каждая клавиша которой связана с определенным входом шифратора. При нажатии выбранной клавиши подается сигнал на соответствующий вход шифратора, и на его выходе возникает двоичное число, соответствующее выгравированному на клавише символу.

На рис. 3.14 приведено символическое изображение шифратора, преобразующего десятичные числа 0, 1, 2,...,9 в двоичное представление в *коде 8421*. Символ CD образован из букв, входящих в английское слово Coder. Слева показаны 10 входов, обозначенных десятичными цифрами 0, 1, 2,...,9, справа — выходы шифратора; цифрами 1, 2, 4, 8 обозначены весовые коэффициенты двоичных разрядов, соответствующих отдельным выходам.

Из приведенного в табл. 3.3 соответствия десятичного и двоичного кодов следует, что переменная $x_1$ на выходе, обозначенном цифрой 1, равна *лог.1*, если это значение имеет одна из входных переменных $y_1$, $y_3$, $y_5$, $y_7$, $y_9$. Следовательно,

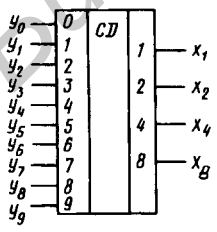



Рис. 3.14

$$x_1 = y_1 \vee y_3 \vee y_7 \vee y_9.$$

Для остальных выходов

$$x_2 = y_2 \vee y_3 \vee y_6 \vee y_7,$$

$$x_4 = y_4 \vee y_5 \vee y_6 \vee y_7,$$

$$x_8 = y_8 \vee y_9.$$

Этой системе логических выражений соответствует схема на рис. 3.15,*а*.

*Таблица 3.3*

| Номер входа (в десятичной системе) | Выходной *код 8421* | | | |
|---|---|---|---|---|
| | $x_8$ | $x_4$ | $x_2$ | $x_1$ |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |

На рис. 3.15,*б* изображена схема шифратора на элементах ИЛИ-НЕ. Шифратор построен в соответствии со следующими выражениями:

$$\bar{x}_1 = \overline{y_1 \vee y_3 \vee y_5 \vee y_7 \vee y_9} = y_1 \downarrow y_3 \downarrow y_5 \downarrow y_7 \downarrow y_9,$$

$$\bar{x}_2 = y_2 \downarrow y_3 \downarrow y_6 \downarrow y_7,$$

$$\bar{x}_4 = y_4 \downarrow y_5 \downarrow y_6 \downarrow y_7,$$

$$\bar{x}_8 = y_8 \downarrow y_9.$$

При этом шифратор имеет инверсные выходы.

При выполнении шифратора на элементах И-НЕ следует пользоваться следующей системой логических выражений:

$$x_1 = \overline{\bar{y}_1 \vee \bar{y}_3 \vee \bar{y}_5 \vee \bar{y}_7 \vee \bar{y}_9} = \overline{\bar{y}_1 \cdot \bar{y}_3 \cdot \bar{y}_5 \cdot \bar{y}_7 \cdot \bar{y}_9} = \bar{y}_1 \mid \bar{y}_3 \mid \bar{y}_5 \mid \bar{y}_7 \mid \bar{y}_9,$$

$$x_2 = \bar{y}_2 \mid \bar{y}_3 \mid \bar{y}_6 \mid \bar{y}_7,$$

$$x_4 = \bar{y}_4 \mid \bar{y}_5 \mid \bar{y}_6 \mid \bar{y}_7,$$

$$x_8 = \bar{y}_8 \mid \bar{y}_9.$$

В этом случае предусмотрена подача на входы инверсных значений, т.е. для получения на выходе двоичного представления некоторой десятичной цифры необходимо на соответствующий вход подать *лог.0*, на остальные входы — *лог.1*. Схема шифратора, выполненная на элементах И-НЕ, приведена на рис. 3.15,*в*.

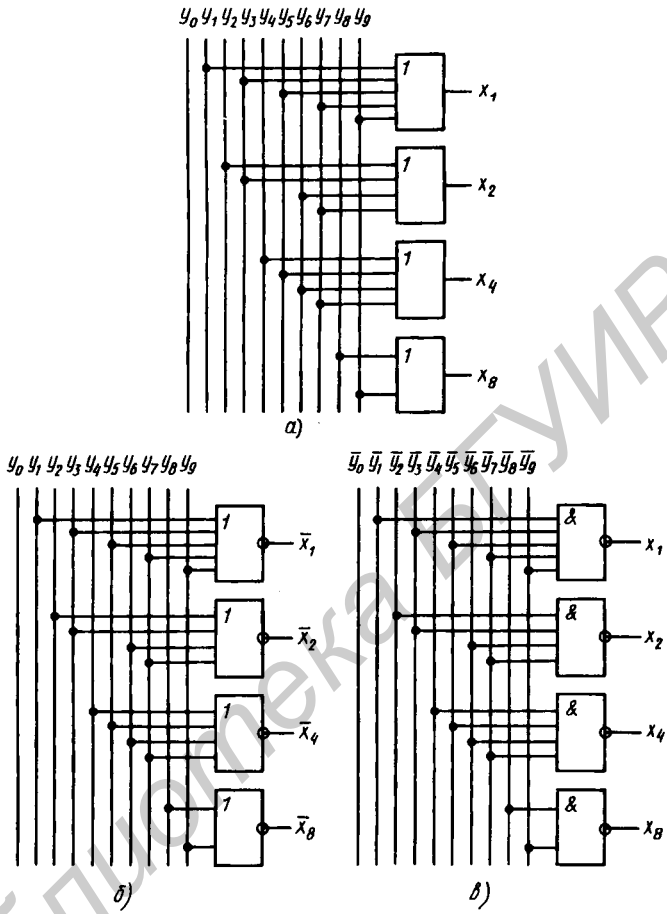



Рис. 3.15

Изложенным способом могут быть построены шифраторы, выполняющие преобразование десятичных чисел в двоичное представление с использованием любого двоичного кода.

### Дешифраторы

Для обратного преобразования двоичных чисел в небольшие по значению десятичные числа используются *дешифраторы* (называемые также *декодерами*). Входы дешифратора предназначаются для подачи двоичных чисел, выходы последовательно нумеруются деся-

тичными числами. При подаче на входы двоичного числа появляется сигнал на определенном выходе, номер которого соответствует входному числу.

Дешифраторы имеют широкое применение. В частности, они используются в устройствах, печатающих на бумаге выводимые из цифрового устройства числа или текст. В таких устройствах двоичное число, поступая на вход дешифратора, вызывает появление сигнала на определенном его выходе. С помощью этого сигнала производится печать символа, соответствующего входному двоичному числу.

На рис. 3.16,*а* приведено символическое изображение дешифратора. Символ DC образован из букв английского слова Decoder. Слева показаны входы, на которых отмечены весовые коэффициенты двоичного кода, справа — выходы, пронумерованные десятичными числами, соответствующие отдельным комбинациям входного двоичного кода. На каждом выходе образуется уровень *лог.1* при строго определенной комбинации входного кода. Дешифратор может иметь парафазные входы для подачи наряду с входными переменными их инверсий, как показано на рис. 3.16,*б*.

По способу построения различают *линейные* и *прямоугольные дешифраторы.*

**Линейный дешифратор**. Рассмотрим построение дешифратора, осуществляющего преобразование, заданное табл. 3.4.

*Таблица 3.4*

| Выходной *код 8421* | | | | Номер выхода (в десятичной системе) |
|---|---|---|---|---|
| $x_8$ | $x_4$ | $x_2$ | $x_1$ | |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |

Значения выходных переменных определяются следующими логическими выражениями:

$$
\begin{aligned}
y_0 &= \bar{x}_8 \cdot \bar{x}_4 \cdot \bar{x}_2 \cdot \bar{x}_1, & y_5 &= \bar{x}_8 \cdot x_4 \cdot \bar{x}_2 \cdot x_1,\\
y_1 &= \bar{x}_8 \cdot \bar{x}_4 \cdot \bar{x}_2 \cdot x_1, & y_6 &= \bar{x}_8 \cdot x_4 \cdot x_2 \cdot \bar{x}_1,\\
y_2 &= \bar{x}_8 \cdot \bar{x}_4 \cdot x_2 \cdot \bar{x}_1, & y_7 &= \bar{x}_8 \cdot x_4 \cdot x_2 \cdot x_1,\\
y_3 &= \bar{x}_8 \cdot \bar{x}_4 \cdot x_2 \cdot x_1, & y_8 &= x_8 \cdot \bar{x}_4 \cdot \bar{x}_2 \cdot \bar{x}_1,\\
y_4 &= \bar{x}_8 \cdot x_4 \cdot \bar{x}_2 \cdot \bar{x}_1, & y_9 &= x_8 \cdot \bar{x}_4 \cdot \bar{x}_2 \cdot x_1;
\end{aligned}
\qquad (3.2)
$$

$$
\begin{aligned}
\bar{y}_0 &= \overline{\bar{x}_8 \cdot \bar{x}_4 \cdot \bar{x}_2 \cdot \bar{x}_1} = \\
&= \bar{x}_8 \,|\, \bar{x}_4 \,|\, \bar{x}_2 \,|\, \bar{x}_1, & \bar{y}_5 &= \bar{x}_8 \,|\, x_4 \,|\, \bar{x}_2 \,|\, x_1,\\
\bar{y}_1 &= \bar{x}_8 \,|\, \bar{x}_4 \,|\, \bar{x}_2 \,|\, x_1, & \bar{y}_6 &= \bar{x}_8 \,|\, x_4 \,|\, x_2 \,|\, \bar{x}_1,\\
\bar{y}_2 &= \bar{x}_8 \,|\, \bar{x}_4 \,|\, x_2 \,|\, \bar{x}_1, & \bar{y}_7 &= \bar{x}_8 \,|\, x_4 \,|\, x_2 \,|\, x_1,\\
\bar{y}_3 &= \bar{x}_8 \,|\, \bar{x}_4 \,|\, x_2 \,|\, x_1, & \bar{y}_8 &= x_8 \,|\, \bar{x}_4 \,|\, \bar{x}_2 \,|\, \bar{x}_1,\\
\bar{y}_4 &= \bar{x}_8 \,|\, x_4 \,|\, \bar{x}_2 \,|\, \bar{x}_1, & \bar{y}_9 &= x_8 \,|\, \bar{x}_4 \,|\, \bar{x}_2 \,|\, x_1.
\end{aligned}
\qquad (3.3)
$$

В линейном дешифраторе выходные переменные формируются по (3.2) либо (3.3). При выполнении дешифратора на элементах И-НЕ пользуются (3.3), получая инверсии выходных функций. В этом случае каждой комбинации входного кода будет соответствовать уровень *лог.0* на строго определенном выходе, на остальных выходах устанавливается уровень *лог.1*. На рис. 3.17, *а*, *б* показана структура дешифратора, построенного на элементах И-НЕ, и его изображение в схемах. Структура имеет особенности, характерные для дешифраторов в интегральном исполнении:

для уменьшения числа входов формирование инверсий входных переменных осуществляется в самом дешифраторе;

подключенные непосредственно к входам дополнительные инверторы уменьшают нагрузку со стороны дешифратора на его входные цепи.

Дешифратор с 16 выходами для дешифрирования всех возможных комбинаций четырехразрядного двоичного *кода 8421* можно построить на двух рассмотренных дешифраторах с 10 выходами. На рис. 3.18 показана структура такого дешифратора. В каждом из дешифраторов используется по восемь выходов, которые и образуют требуемые 16 выходов ($y_0, y_1, \ldots, y_{15}$).

**Прямоугольный дешифратор.** Рассмотрим принцип построения прямоугольного дешифратора на примере дешифратора с 4 входами и16 выходами.

Разобьем входные переменные $x_8$, $x_4$, $x_2$, $x_1$ на две группы по две переменные в каждой: $x_8, x_4$ и $x_2, x_1$.

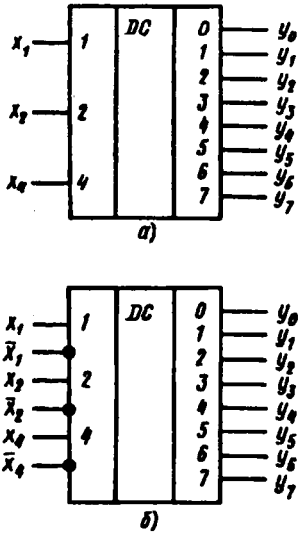
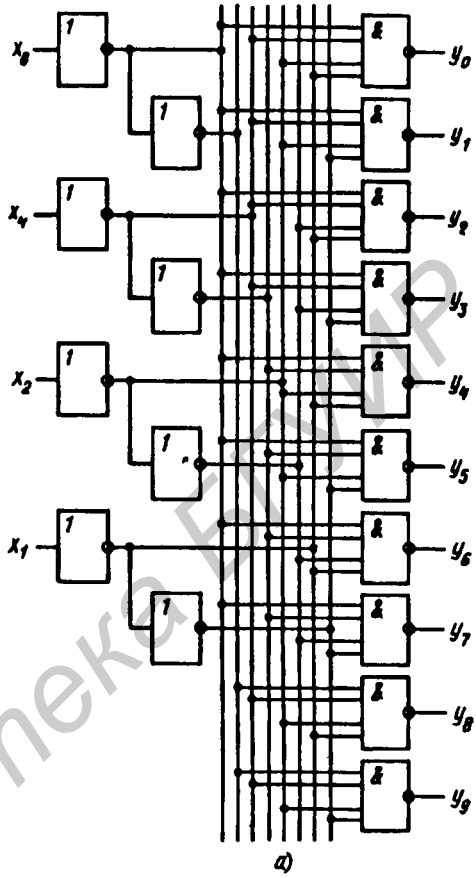



Рис. 3.16



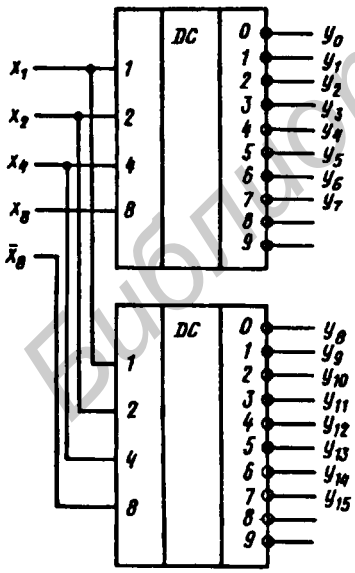




Рис.3.17

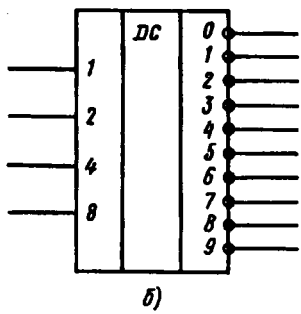



Рис.3.18

Каждую пару переменных используем в качестве входных переменных отдельного линейного дешифратора на четыре выхода, как показано на рис. 3.19,*а*. Выходные переменные линейных дешифраторов определяются следующими выражениями:

$$y'_0 = \bar{x}_8 \cdot \bar{x}_4, \qquad y''_0 = \bar{x}_2 \cdot \bar{x}_1,$$

$$y'_1 = \bar{x}_8 \cdot x_4, \qquad y''_1 = \bar{x}_2 \cdot x_1,$$

$$y'_2 = x_8 \cdot \bar{x}_4, \qquad y''_2 = x_2 \cdot \bar{x}_1,$$

$$y'_3 = x_8 \cdot x_4, \qquad y''_3 = x_2 \cdot x_1,$$

Эти дешифраторы выполняют функции первой ступени дешифратора.

Выходные переменные $y_0, y_1, \ldots, y_{15}$ прямоугольного дешифратора можно представить логическими выражениями, используя в них в качестве аргументов выходные переменные $y'_0,\ldots,y'_3$ и $y''_0,\ldots,y''_3$ линейных дешифраторов:

$$y_0 = \bar{x}_8 \cdot \bar{x}_4 \cdot \bar{x}_2 \cdot \bar{x}_1 = y'_0 \cdot y''_0,$$

$$y_1 = \bar{x}_8 \cdot \bar{x}_4 \cdot \bar{x}_2 \cdot x_1 = y'_0 \cdot y''_1,$$

$$y_2 = \bar{x}_8 \cdot \bar{x}_4 \cdot x_2 \cdot \bar{x}_1 = y'_0 \cdot y''_2,$$

.....

$$y_{15} = x_8 \cdot x_4 \cdot x_2 \cdot x_1 = y'_3 \cdot y''_3.$$

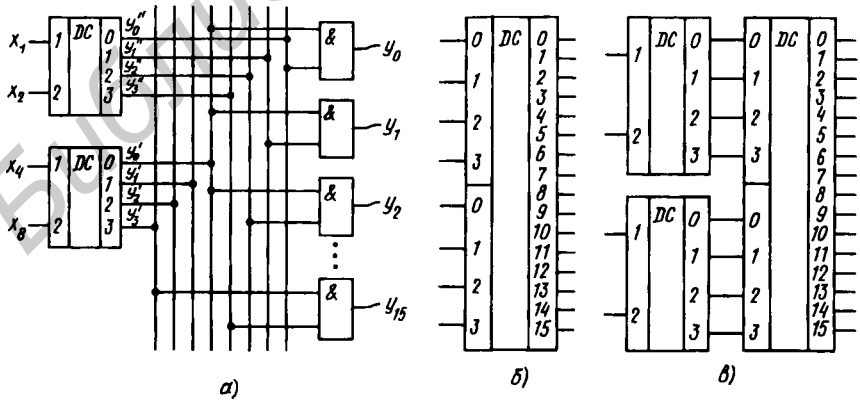



Рис. 3.19

Эти логические операции выполняются в отдельном дешифраторе второй ступени, называемом *матричным* и состоящим из двухвходовых элементов. На рис. 3.19,*б* показано условное обозначение матричного дешифратора, где помеченные десятичными числами две группы входов служат для подключения к выходам двух предварительных ступеней дешифрации. На рис. 3.19,*в* представлена структура прямоугольного дешифратора с использованием символов линейного и матричного дешифраторов.

Могут быть построены прямоугольные дешифраторы с числом ступеней, большим двух.

Применение прямоугольного дешифратора может оказаться более выгодным, чем линейного дешифратора, в тех случаях, когда велико число входов и нежелательно использовать требующиеся для построения линейного дешифратора элементы с большим числом входов. Однако прохождение сигналов последовательно через несколько ступеней приводит в прямоугольном дешифраторе к большей задержке распространения сигнала.

### Преобразователи кодов

В цифровых устройствах часто возникает необходимость преобразования числовой информации из одной двоичной системы в другую (из одного двоичного кода в другой). Примером такого преобразования может служить преобразование чисел из двоичного *кода 8421*, в котором выполняются арифметические операции, в двоичный *код 2 из 5* для передачи по линии связи. Эта задача выполняется устройствами, называемыми *преобразователями кодов*. Для преобразования кодов можно пользоваться двумя методами:

методом, основанным на преобразовании исходного двоичного кода в десятичный и последующем преобразовании десятичного представления в требуемый двоичный код;

методом, основанным на использовании логического устройства комбинационного типа, непосредственно реализующего данное преобразование.

Первый метод структурно реализуется соединением дешифратора и шифратора и удобен в тех случаях, когда можно использовать стандартные дешифраторы и шифраторы в интегральном исполнении.

Рассмотрим подробнее второй метод на конкретных примерах преобразования двоичных кодов.

**Преобразование** *кода 8421* в *код 2421*. Обозначим переменные, соответствующие отдельным разрядам *кода 8421*, $x_4, x_3, x_2, x_1$; то же для *кода 2421*: $y_4, y_3, y_2, y_1$. В табл. 3.5 приведено соответствие комбинаций обоих кодов.

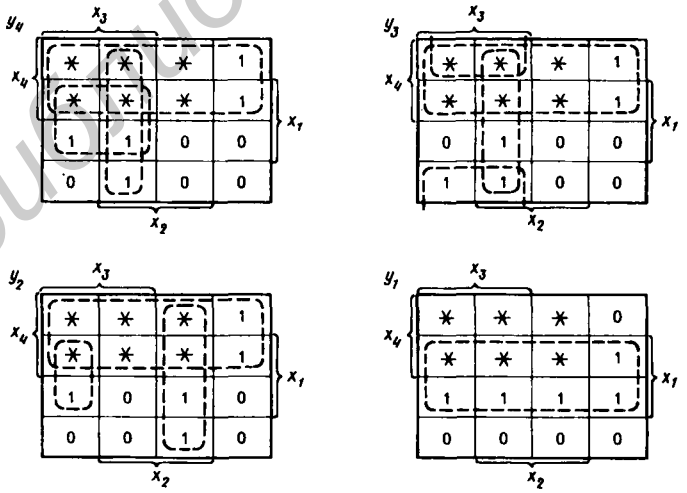
*Таблица 3.5*

| *Код 8421* | | | | *Код 2421* | | | |
|---|---|---|---|---|---|---|---|
| $x_4$ | $x_3$ | $x_2$ | $x_1$ | $y_4$ | $y_3$ | $y_2$ | $y_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

Каждая из переменных $y_4, y_3, y_2, y_1$ может рассматриваться функцией аргументов $x_4, x_3, x_2, x_1$ и, следовательно, представлена через эти аргументы соответствующим логическим выражением. Для получения указанных логических выражений представим переменные $y_4$, $y_3$, $y_2$, $y_1$ таблицами истинности в форме таблицы Вейча (табл.3.6).

Получим минимальную форму логических выражений, представленных через операции И, ИЛИ, НЕ и через операцию И-НЕ:

*Таблица 3.6*



$y_4 = x_4 \vee x_3 \cdot x_2 \vee x_3 \cdot x_1$, $\quad y_4 = \overline{x}_4 \mid (x_3 \mid x_2) \mid (x_3 \mid x_1)$,

$y_3 = x_4 \vee x_3 \cdot x_2 \vee x_3 \cdot \overline{x}_1$, $\quad y_3 = \overline{x}_4 \mid (x_3 \mid x_2) \mid (x_3 \mid \overline{x}_1)$,

$y_2 = x_4 \vee \overline{x}_3 \cdot x_2 \vee x_3 \cdot \overline{x}_2 \cdot x_1$, $\quad y_2 = \overline{x}_4 \mid (\overline{x}_3 \mid x_2) \mid (x_3 \mid \overline{x}_2 \mid x_1)$,

$y_1 = x_1$, $\quad y_1 = x_1$.

На рис. 3.20 приведена логическая структура преобразователя кодов, построенная на элементах И-НЕ с использованием полученных логических выражений.

**Преобразование** *кода 2421* в *код 8421*. Для реализации данного преобразования (обратного по отношению к рассмотренному выше) требуется получить логические выражения для переменных $x_4$, $x_3$, $x_2$, $x_1$, используя в качестве аргументов переменные $y_4$, $y_3$, $y_2$, $y_1$.

Таблицы Вейча для переменных $x_4$, $x_3$, $x_2$, $x_1$ представлены табл.3.7. Логические выражения для переменных $x_4$, $x_3$, $x_2$, $x_1$:

$x_4 = y_3 \cdot y_2$, $\quad x_4 = \overline{y_3 \mid y_2}$,

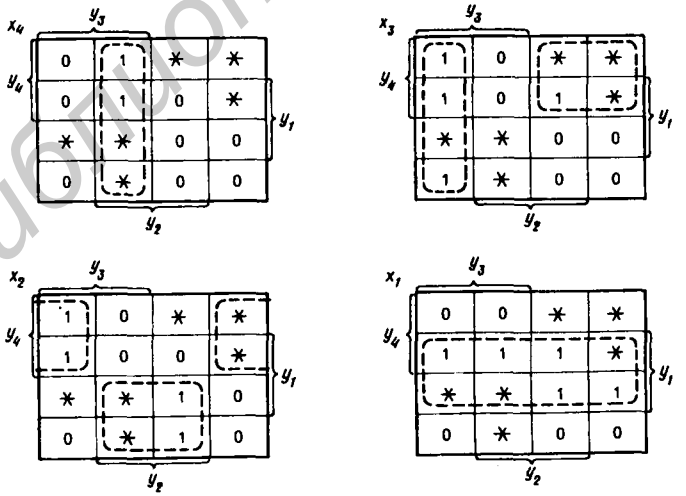
$x_3 = y_3 \cdot \overline{y}_2 \vee y_4 \cdot \overline{y}_3$, $\quad x_3 = (y_3 \mid \overline{y}_2) \mid (y_4 \mid \overline{y}_3)$,

$x_2 = y_4 \cdot \overline{y}_2 \vee \overline{y}_4 \cdot y_2$, $\quad x_2 = (y_4 \mid \overline{y}_2) \mid (\overline{y}_4 \mid y_2)$,

$x_1 = y_1$, $\quad x_1 = y_1$.

Логическая структура преобразователя приведена на рис. 3.21.

*Таблица 3.7*



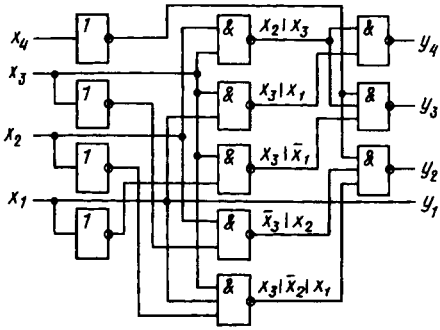



Рис. 3.20

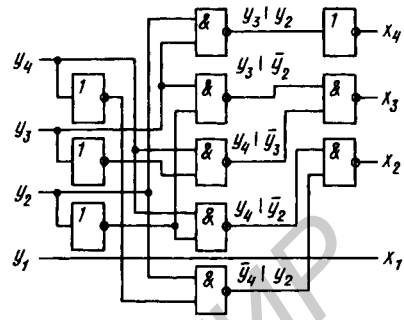



Рис. 3.21

**Преобразователь для цифровой индикации.** Один из способов цифровой индикации состоит в следующем. Имеется семь элементов, расположенных так, как показано на рис. 3.22,*а*. Каждый может светиться либо не светиться, в зависимости от значения соответствующей логической переменной, управляющей его свечением. Вызывая свечение элементов в определенных комбинациях, можно получить изображение десятичных цифр 0, 1, ... , 9 (рис. 3.22,*б*).

Десятичные цифры, отображение которых необходимо вызвать, задаются обычно в двоичном коде. При этом возникает задача формирования логических переменных $y_1$, $y_2$, ... , $y_7$ для управления отдельными элементами в устройстве индикации. Таблица истинности для этих переменных представлена в табл. 3.8.

При построении таблицы были приняты следующие условия: если элемент индикатора светится, то это означает, что он находится в состоянии 1, если погашен — то в состоянии 0, управление элементом осуществляется таким образом, что *лог.1* на некотором входе индикатора вызывает гашение соответствующего элемента (т.е. чтобы *i*-й элемент был погашен и $z_1 = 0$, необходимо подать на *i*-й вход индикатора управляющий сигнал $y_1 = 1$). Таким образом, $y_i = \overline{z}_i$. Например, для высвечивания цифры 0 необходимо погасить седьмой элемент ($z_7 = 0$), оставив остальные элементы в состоянии свечения; следовательно, при этом управляющий сигнал $y_7 = 1$, остальные управляющие сигналы $y_1, \ldots, y_6$ должны иметь уровень *лог.0*.

Формирование управляющих сигналов производится логическим устройством, для синтеза которого в табл. 3.9 построены таблицы истинности в форме таблиц Вейча отдельно для каждой переменной $y_1, \ldots, y_7$. Синтезируемое устройство является устройством с несколькими выходами, и для получения минимальной схемы необходимо в таблицах Вейча построить минимальное число областей, обеспечивающих покрытие клеток, содержащих 1 во всех семи таблицах. Построение этих областей имеет следующие особенности. В таблицах переменных $y_5$ и $y_6$ использованы области I и V, которые входят в таблицы других переменных. Если вместо этих областей в таблицах пременных $y_5$ и $y_6$ построить области с большим охватом клеток, это вызовет увеличение общего количества областей и, следовательно, увеличится количество логических элементов, требуемых для формирования соответствующих им логических выражений. Выделенным областям соответствуют следующие логические выражения:

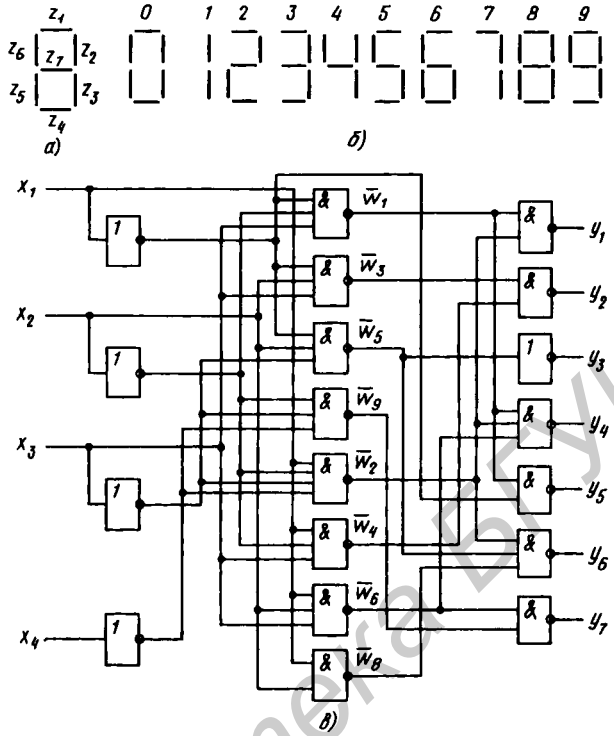
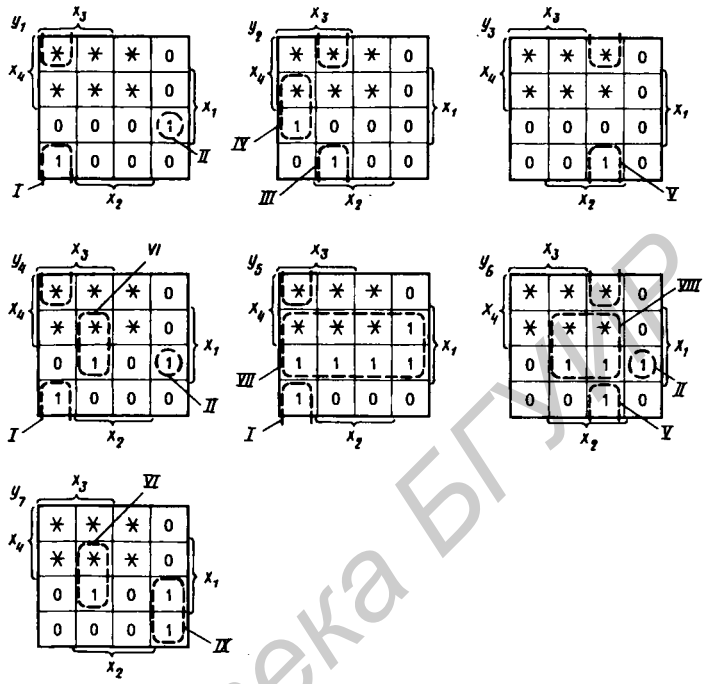



Рис.3.22

*Таблица 3.8*

| Десятичная цифра | *Код 8421* | | | | Состояние элементов $z_1, \ldots, z_7$ и значение управляющих сигналов $y_1, \ldots, y_7$ | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $x_4$ | $x_3$ | $x_2$ | $x_1$ | $\bar{z}_1$ | $\bar{z}_2$ | $\bar{z}_3$ | $\bar{z}_4$ | $\bar{z}_5$ | $\bar{z}_6$ | $\bar{z}_7$ |
| | | | | | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |




| | |
|---|---|
| область I | $\bar{x}_1 \cdot \bar{x}_2 \cdot x_3 = w_1$, |
| область II | $x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4 = w_2$, |
| область III | $\bar{x}_1 \cdot x_2 \cdot x_3 = w_3$, |
| область IV | $\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 = w_4$, |
| область V | $\bar{x}_1 \cdot x_2 \cdot \bar{x}_3 = w_5$, |
| область VI | $x_1 \cdot x_2 \cdot x_3 = w_6$, |
| область VII | $x_1 = w_7$, |
| область VIII | $x_1 \cdot x_2 = w_8$, |
| область IX | $x_2 \cdot \bar{x}_3 \cdot \bar{x}_4 = w_9$. |

Теперь нетрудно записать логические выражения для выходных величин $y_1, \ldots, y_7$:

$y_1 = w_1 \vee w_2 = \overline{\bar{w}_1 \cdot \bar{w}_2} = \overline{\bar{w}_1 \mid \bar{w}_2} = (\bar{x}_1 \mid \bar{x}_2 \mid x_3) \mid (x_1 \mid \bar{x}_2 \mid \bar{x}_3 \mid \bar{x}_4)$,

$y_2 = w_3 \vee w_4 = (\bar{x}_1 \mid x_2 \mid x_3) \mid (x_1 \mid \bar{x}_2 \mid x_3)$,

$y_3 = w_5 = \overline{\bar{x}_1 \mid x_2 \mid \bar{x}_3}$,

$y_4 = w_1 \vee w_2 \vee w_6 = (\bar{x}_1 \mid \bar{x}_2 \mid x_3) \mid (x_1 \mid \bar{x}_2 \mid \bar{x}_3 \mid \bar{x}_4) \mid (x_1 \mid x_2 \mid x_3)$,

$y_5 = w_1 \vee w_7 = (\bar{x}_1 \mid \bar{x}_2 \mid x_3) \mid \bar{x}_1$,

$y_6 = w_2 \vee w_5 \vee w_6 = (x_1 | \bar{x}_2 | \bar{x}_3 | \bar{x}_4) | (\bar{x}_1 | x_2 | \bar{x}_3) | (x_1 | x_2)$,

$y_7 = w_6 \vee w_9 = (x_1 | x_2 | x_3) | (\bar{x}_2 | \bar{x}_3 | \bar{x}_4)$.

Построенная в соответствии с этими выражениями схема преобразователя приведена на рис. 3.22,*в*.

Определим количество микросхем, необходимых для построения преобразователя. При этом следует учитывать, что в корпусе выпускаемых промышленностью микросхем может содержаться несколько логических элементов. В табл.3.10 приведен расчет количества корпусов микросхем.

*Таблица 3.10*

| Тип логического элемента | Число элементов в корпусе микросхемы | Число элементов в преобразователе | Число корпусов микросхем |
|---|---|---|---|
| Инвертор | 6 | 6 | 1 |
| Двухвходовый элемент И-НЕ | 4 | 5 | 5/4 |
| Трехвходовый элемент И-НЕ | 3 | 8 | 8/3 |
| Четырехвходовый элемент И-НЕ | 2 | 1 | 1/2 |
| Общее количество корпусов микросхем | | | $5^5/12$ |

### *Упражнения*

1. Постройте шифратор для преобразования десятичных чисел 0, 1, 2, ... ,15 в двоичный *код 8421*.

2. С использованием элементов ИЛИ-НЕ постройте линейный дешифратор, осуществляющий преобразование двоичных чисел в *коде 8421* в десятичные числа 0, 1, 2, ... ,9.

3. На элементах ИЛИ-НЕ постройте прямоугольный дешифратор, осуществляющий преобразование двоичных чисел в *коде 8421* в десятичные числа 0, 1, 2, ... ,15.

4. Постройте преобразователи двоичного *кода 8421* в следующие виды кодов (см. табл. 2.1): а) *2 из 5*; б) *с избытком 3*; в) *3а + 2;* г) *7421*.

5. Постройте преобразователи, осуществляющие преобразование приведенных ниже видов кода в двоичный *код 8421*: а) *2 из 5*; б) *с избытком 3*; в) *3а + 2;* г) *7421*.

## 3.3. МУЛЬТИПЛЕКСОРЫ И ДЕМУЛЬТИПЛЕКСОРЫ

### Мультиплексоры

**Назначение и принцип работы**. Устройство, которое осуществляет выборку одного из нескольких входов и подключает его к своему выходу, называется *мультиплексором*. Мультиплексор имеет несколько информационных входов ( $D_0$, $D_1$, ...), адресные входы ($A_0$, $A_1$, ...), вход для подачи стробирующего сигнала $C$ и один выход $Q$. На рис. 3.23, *а* показано символическое изображение мультиплексора с четырьмя информационными входами.

Каждому информационному входу мультиплексора присваивается номер, называемый *адресом*. При подаче стробирующего сигнала на

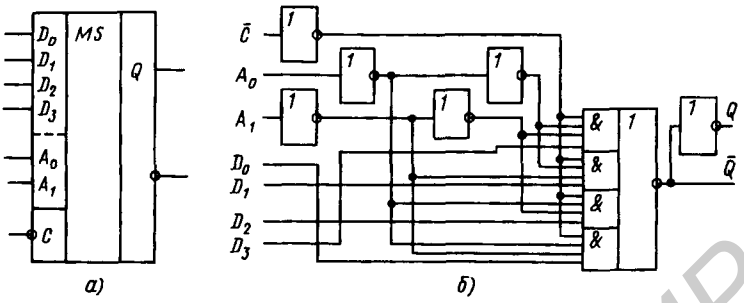



Рис. 3.23

вход $C$ мультиплексор выбирает один из входов, адрес которого задается двоичным кодом на адресных входах, и подключает его к выходу.

Таким образом, подавая на адресные входы адреса различных информационных входов, можно передавать цифровые сигналы с этих входов на выход $Q$. Очевидно, число информационных входов $n_{\text{инф}}$ и число адресных входов $n_{\text{адр}}$ связаны соотношением $n_{\text{инф}} = 2^{n_{\text{адр}}}$.

Функционирование мультиплексора определяется табл. 3.11.

*Таблица 3.11*

| Адресные входы | | Стробирующий сигнал | Выход |
|---|---|---|---|
| $A_1$ | $A_0$ | $C$ | Q |
| × | × | 0 | 0 |
| 0 | 0 | 1 | $D_0$ |
| 0 | 1 | 1 | $D_1$ |
| 1 | 0 | 1 | $D_2$ |
| 1 | 1 | 1 | $D_3$ |

При отсутствии стробирующего сигнала ($C = 0$) связь между информационными входами и выходом отсутствует ($Q = 0$). При подаче стробирующего сигнала ($C = 1$) на выход передается логический уровень того из информационных входов $D_i$, номер которого $i$ в двоичной форме задан на адресных входах. Так, при задании адреса $A_1 A_0 = 11_2 = 3_{10}$ на выход $Q$ будет передаваться сигнал информационного входа с адресом $3_{10}$, т.е. $D_3$.

По этой таблице можно записать следующее логическое выражение для выхода $Q$:

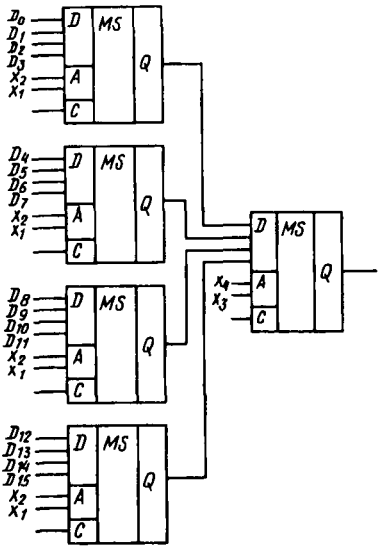



Рис. 3.24

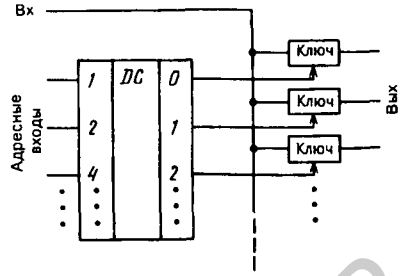



Рис.3.25

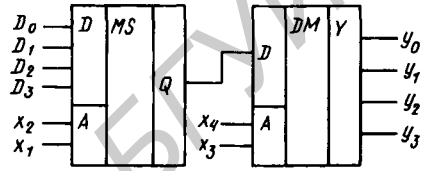



Рис. 3.26

$$Q = (D_0 \cdot \overline{A}_1 \cdot \overline{A}_0 \vee D_1 \cdot \overline{A}_1 \cdot A_0 \vee D_2 \cdot A_1 \cdot \overline{A}_0 \vee D_3 \cdot A_1 \cdot A_0) \cdot C. \qquad (3.4)$$

Построенная по этому выражению принципиальная схема мультиплексора приведена на рис. 3.23,*б*.

В тех случаях, когда требуется передавать на выходы многоразрядные входные данные в параллельной форме, используется параллельное включение мультиплексоров по числу разрядов передаваемых данных.

**Мультиплексорное дерево.** Максимальное число информационных входов мультиплексоров, выполненных в виде интегральных схем, равно 16. Если требуется построить мультиплексорное устройство с большим числом входов, можно объединить мультиплексоры в схему так называемого *мультиплексорного дерева*. Такое мультиплексорное дерево, построенное на четырехвходовых мультиплексорах, показано на рис. 3.24. Схема состоит из четырех мультиплексоров первого уровня с адресными переменными $x_1$, $x_2$ и мультиплексора второго уровня с адресными переменными $x_3$, $x_4$. Мультиплексорное устройство имеет 16 входов, разбитых на четверки, которые подключены к отдельным мультиплексорам первого уровня. Мультиплексор второго уровня, подключая к общему выходу устройства выходы отдельных мультиплексоров первого уровня, переключает четверки входов. Внутри четверки требуемый вход выбирается мультиплексором первого уровня. По такой

схеме, используя восьмивходовые мультиплексоры, можно построить мультиплексорное устройство, имеющее 64 входа.

На первом и втором уровнях мультиплексорного дерева можно использовать мультиплексоры с разным числом входов. Если на первом уровне такого дерева используются мультиплексоры с числом адресных переменных $n_{адр1}$, на втором — с числом переменных $n_{адр2}$, то общее число входов мультиплексорного дерева $n_{инф} = 2^{n_{адр1} + n_{адр2}}$, а число мультиплексоров в схеме составит $2^{n_{адр2}} + 1$.

### Демультиплексор

Демультиплексор имеет один информационный вход и несколько выходов и осуществляет коммутацию входа к одному из выходов, имеющему заданный адрес (номер). На рис. 3.25 показана структура демультиплексора. Она включает в себя дешифратор, выходы которого управляют ключами. В зависимости от поданной на адресные входы кодовой комбинации, определяющей номер выходной цепи, дешифратор открывает соответствующий ключ, и вход демультиплексора подключается к определенному его выходу.

Объединяя мультиплексор с демультиплексором, можно построить устройство, в котором по заданным адресам один из входов подключается к одному из выходов (рис. 3.26). Таким образом может быть выполнена любая комбинация соединений входов с выходами. Например, при комбинации значений адресных переменных $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 0$ вход $D_2$ окажется подключенным к выходу $y_0$.

Если требуется большое число выходов, может быть построено демультиплексорное дерево.

## 3.4. РЕГИСТРЫ

### Общие сведения

Основная функция регистров — хранение одного многоразрядного числа. При этом число должно быть представлено в двоичной системе счисления или в любой другой системе, но с двоичным представлением цифр разрядов (т.е. в любой двоично-кодированной системе счисления). Регистр строится в виде набора триггеров, каждый из которых предназначается для хранения цифр определенного числа. Таким образом, регистр для хранения $n$-разрядного двоичного числа должен содержать $n$ триггеров.

Регистры могут использоваться для выполнения и некоторых других функций: сдвиг хранимого в регистре числа на определенное число разрядов влево или вправо, преобразование числа из последовательной формы (при которой оно передается последовательно разряд за

разрядом) в параллельную (с передачей всех разрядов одновременно), преобразование из параллельной формы представления числа в последовательную и др.

В зависимости от формы представления числа (параллельной или последовательной), вводимого в регистр, различают два типа регистров: *параллельные и последовательные*. В параллельный регистр предназначенное для хранения число подается одновременно всеми разрядами, т.е. в параллельной форме. В последовательный регистр ввод числа производится путем последовательной во времени подачи цифр отдельных разрядов (обычно начиная с цифры младшего разряда), т.е. в последовательной форме.

### Параллельный регистр

Пусть на вход регистра поступает парафазный код числа. При этом для каждого разряда числа предусматривается два входа, на один из которых поступает прямой код, на другой — инверсный. Прием такого числа может производиться в регистр, построенный с использованием простейших синхронных RS- триггеров, как показано на рис. 3.27,*а*.

Если цифра *i*-го разряда $a_i = 1$, то на вход *S* соответствующего триггера поступает 1 и при подаче уровня *лог.1* на вход *C* триггер устанавливается в состояние 1. Если $a_i = 0$ ($\bar{a}_i = 1$), то 1 поступает на вход *R* и этот триггер устанавливается в состояние 0. Таким образом, триггеры устанавливаются в состояния, определяемые поступающими на их входы цифрами разрядов числа.

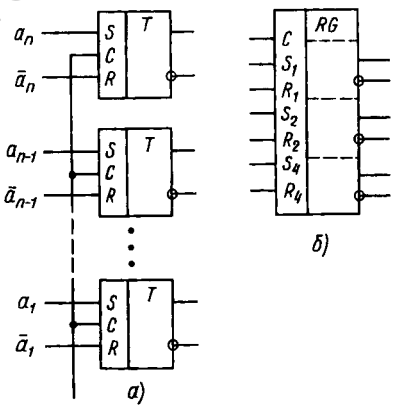



Рис. 3.27

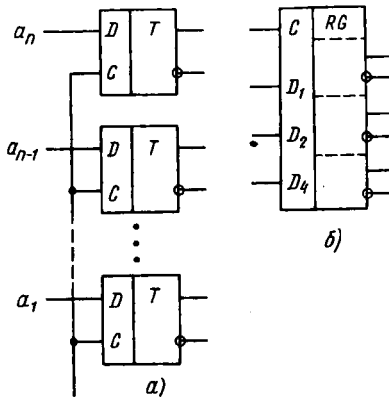



Рис. 3.28

Когда на вход регистра поступает однофазный код числа (без подачи инверсных значений цифр разрядов), регистр может быть построен с использованием простейших синхронных D-триггеров (рис. 3.28,*а*). В таком регистре при уровне *лог.1* на входе *C* триггеры устанавливаются в состояния, определяемые действующими на входах *D* цифрами разрядов. На рис. 3.27,*б* и 3.28,*б* приведены условные обозначения рассмотренных типов регистров.

### Сдвиговый регистр

Покажем пример сдвига числа на один разряд вправо.

| | |
|---|---|
| Номер разряда | 4 3 2 1 |
| Число в регистре до сдвига | 1 0 1 1 |
| | ↘ ↘ ↘ ↘ |
| Число в регистре после сдвига | 0 1 0 1 |

Суть сдвига состоит в том, что цифра, имевшаяся до сдвига в *i*-м разряде регистра, передается в соседний справа (*i* – 1)-й разряд (т.е. значение четвертого разряда передается в третий разряд, значение третьего разряда — во второй разряд и т.д.). В крайний левый разряд заносится значение, подаваемое извне, а цифра крайнего правого разряда числа выдвигается из регистра во внешнюю цепь. Такого рода сдвиги вправо (либо влево) выполняются так называемым *сдвиговым регистром*.

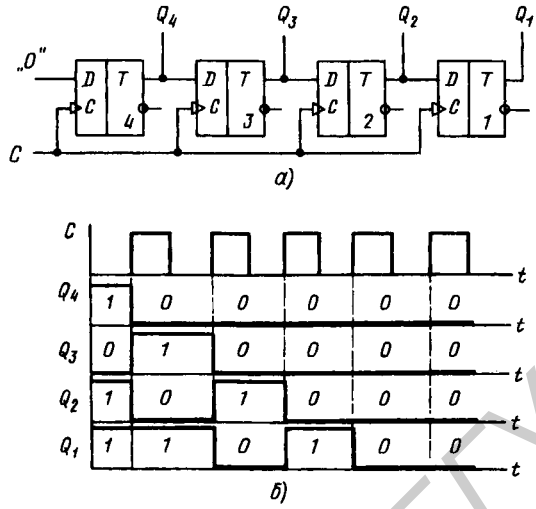



Рис. 3.29

Для построения сдвигового регистра чаще всего используются D-триггеры, управляемые одним фронтом синхронизирующего сигнала, но могут использоваться и другие типы триггеров, управляемых одним фронтом синхронизирующего сигнала, либо триггеры, построенные по принципу двухступенчатого запоминания информации.

Рассмотрим работу показанного на рис. 3.29,*а* сдвигового регистра, построенного на D-триггерах. Выход $Q$ триггера каждого разряда подключен к входу $D$ триггера соседнего более младшего разряда. Таким образом, при низком уровне синхронизирующего сигнала хранящееся в триггере значение разряда числа передается на вход триггера соседнего справа разряда и производит в нем подготовку управляющих цепей. В момент положительного фронта синхронизирующего сигнала каждый из триггеров устанавливается в состояние, соответствующее действовавшему на входе $D$ сигналу, и число в регистре оказывается сдвинутым вправо на один разряд; в старший разряд заносится значение, подаваемое извне на вход $D$ триггера этого разряда. На рис. 3.29,*б* приведено содержимое регистра при выполнении последовательных сдвигов вправо.

Для осуществления сдвига влево необходимо в сдвиговом регистре изменить связи между триггерами, подключая выход триггера ко входу $D$ триггера соседнего слева (более старшего) разряда.

Сдвиговые регистры имеют разнообразные применения. Рассмотрим некоторые из них.

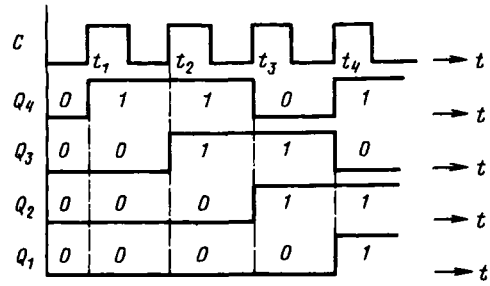



Рис. 3.30

**Последовательный регистр**. Представляет собой сдвиговый регистр, в который многоразрядное число вводится последовательно цифра за цифрой (обычно начиная с цифры младшего разряда) через один из его крайних разрядов (обычно через старший). Таким образом, представленный на рис. 3.29,*а* сдвиговый регистр может выполнять функции последовательного регистра, если на вход *D* триггера старшего разряда не постоянно подавать уровень *лог.0* (как показано на рисунке), а вводить в регистр число в последовательной форме.

Временны́е диаграммы на рис. 3.30 иллюстрируют работу такого последовательного регистра при вводе числа 1011 последовательно разряд за разрядом, начиная с младшего разряда. В момент $t_1$ появление синхронизирующего импульса на входе *C* вызывает сдвиг информации в регистре на один ряд вправо. Если до этого момента в регистре было число $0000_2$, то в результате сдвига в первом, втором, третьем разрядах сохранится значение 0; в четвертый разряд будет со входа принято значение 1. Таким образом, в регистре образуется число $1000_2$. В момент $t_2$ появления следующего синхронизирующего импульса процессы сдвига и приема очередного разряда вводимого числа приводят регистр в состояние $1100_2$. Далее в момент $t_3$ в регистре образуется число $0110_2$ и, наконец, в момент $t_4$ — число $1011_2$. Поданное на вход число оказывается зафиксированным в регистре.

**Преобразование формы представления чисел из последовательной в параллельную**. Данное преобразование может быть выполнено сдвиговым регистром, в котором предусматривается выдача числа в параллельной форме (одновременно с выходов триггеров всех разрядов).

В процессе преобразования формы представления числа *n*-разрядное двоичное число в последовательной форме подается на вход последовательного регистра. Через *n* тактов (после подачи *n* синхронизирующих импульсов) число окажется принятым в регистр и может быть затем снято в параллельной форме с выходов триггеров всех разрядов.

**Преобразование формы представления чисел из параллельной в последовательную.** Это преобразование может быть выполнено сдвиговым регистром, в котором предусмотрены входы для приема числа в параллельной форме.

После принятия в регистр $n$-разрядного числа на выходе триггера первого разряда образуется младший разряд числа. При сдвиге числа в регистре на один разряд вправо в триггер первого разряда передается цифра второго разряда введенного в регистр числа, и, таким образом, с выхода этого триггера снимается цифра следующего разряда. При повторении сдвигов вправо $(n-1)$ раз на выходе триггера первого разряда регистра появляются цифры всех $n$ разрядов, и, таким образом, число может быть снято в последовательной форме.

Для выдвижения из регистра числа в последовательной форме, начиная с его старшего разряда, необходимо производить серию сдвигов влево, снимая цифры разрядов числа с выхода триггера старшего разряда регистра.

## 3.5. СЧЕТЧИКИ

### Назначение и типы счетчиков

*Счетчик* — это цифровое устройство, определяющее, сколько раз на его входе появился некоторый определенный логический уровень. В дальнейшем во всех случаях, когда это не оговаривается специально, будем полагать, что счетчик подсчитывает содержащиеся во входном сигнале переходы с уровня *лог. 0* к уровню *лог.1*. При входном сигнале, имеющем форму последовательности импульсов, счетчик ведет счет поступающих на вход импульсов. Числа в счетчике представляются некоторыми комбинациями состояний триггеров. При поступлении на вход очередного уровня *лог.1* в счетчике устанавливается новая комбинация состояний триггеров, соответствующая числу, на единицу большему предыдущего числа. Таким образом, счетчик представляет собой логическое устройство последовательностного типа, в котором новое состояние определяется предыдущим состоянием и значением логической переменной на входе.

Для представления чисел в счетчике могут использоваться двоичная или десятичная система счисления. При использовании двоичной систе-

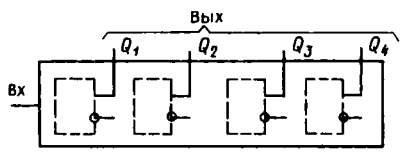



Рис. 3.31

мы состояния триггеров и соответствующие им логические уровни на прямых выходах триггеров определяют цифры двоичных разрядов числа. Если для регистрации двоичного числа в счетчике используется $n$ триггеров, то максимальное значение числа, до которого может вестись счет, $N = 2^n - 1$. Так, при $n = 4$ $N = 15$. На рис. 3.31 показаны вход и выходы счетчика (без раскрытия схемы счетчика), а в табл. 3.12 приведено состояние триггеров, соответствующее различному числу поступивших на вход импульсов.

*Таблица 3.12*

| Число поступивших импульсов | Состояние триггеров | | | | Число поступивших импульсов | Состояние триггеров | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $Q_4$ | $Q_3$ | $Q_2$ | $Q_1$ | | $Q_4$ | $Q_3$ | $Q_2$ | $Q_1$ |
| 0 | 0 | 0 | 0 | 0 | 8 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 9 | 1 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 10 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 11 | 1 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 12 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 13 | 1 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 | 14 | 1 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 | 15 | 1 | 1 | 1 | 1 |

При использовании десятичной системы счисления цифры разрядов десятичного числа в счетчике представляются в четырехразрядной двоичной форме, т.е. используется двоично-кодированная десятичная система счисления. Таким образом, для представления цифр каждого разряда десятичного числа (каждой декады) требуется четыре триггера, и если число десятичных разрядов $k$, то число триггеров, необходимое для регистрации чисел в счетчике, равно $4k$, а максимальное значение чисел $N = 10^k - 1$. В табл. 3.13 показана последовательность состояний триггеров в двухразрядном десятичном счетчике, приведенном на рис. 3.32.

Наряду с *суммирующими счетчиками*, в которых в процессе счета каждое очередное число в счетчике на единицу превышает предыдущее,

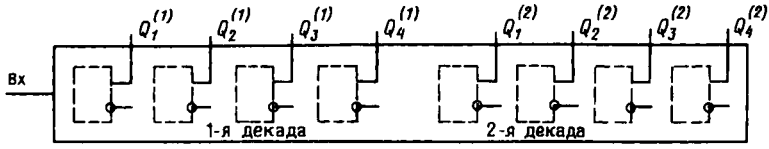



Рис. 3.32

*Таблица 3.13*

| Число поступивших импульсов | Состояние триггеров | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $Q_4$ | $Q_3$ | $Q_2$ | $Q_1$ | $Q_4$ | $Q_3$ | $Q_2$ | $Q_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 10 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 99 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

используются и такие счетчики, в которых в процессе счета числа убывают (эти счетчики называются *вычитающими*). Находят применение счетчики, которые допускают в процессе работы автоматическое переключение (*реверс*) из режима суммирующего счетчика в режим вычитающего счетчика и наоборот. Такие счетчики называются *реверсивными.* Хотя для построения счетчиков могут использоваться любые типы триггеров, на которых может быть организован счетный вход, в дальнейшем будем пользоваться только одним типом — JK-триггерами.

### Суммирующие двоичные счетчики

В суммирующем счетчике поступление на вход очередного уровня *лог.1* (очередного импульса) вызывает увеличение на единицу хранимого в счетчике числа. Таким образом, в счетчике устанавливается число, которое получается путем суммирования предыдущего значения с единицей. Это суммирование проводится по обычным правилам сложения в двоичной системе счисления. Например:

| | | |
|---|---|---|
| Переносы | | 1 1 1 |
| Исходное число | 1 0 1 1 0 | 1 0 1 1 1 |
| | + 1 | + 1 |
| Результат | 1 0 1 1 1 | 1 1 0 0 0 |

Заметим, что в процессе такого суммирования имеют место следующие особенности:

1) если цифра некоторого разряда остается неизменной либо изменяется с 0 на 1, то при этом цифры более старших разрядов не изменяются;

2) если цифра некоторого разряда изменяется с 1 на 0, то происходит инвертирование цифры следующего за ним более старшего разряда.

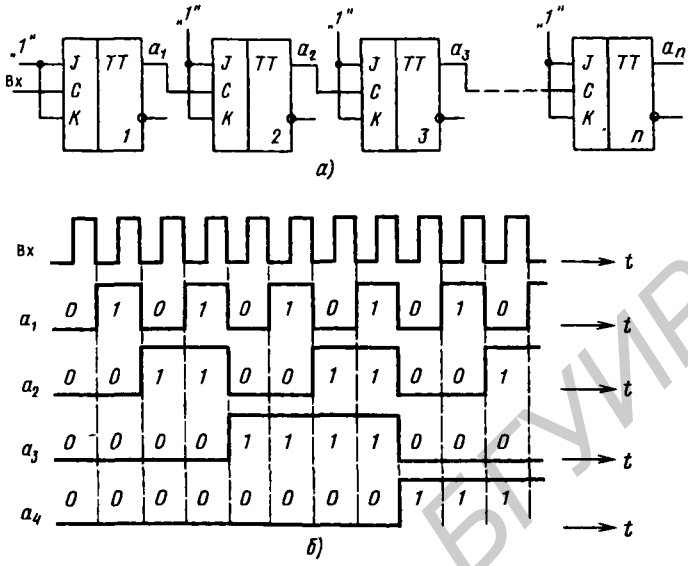



Рис. 3.33

Этот принцип использован при построении схемы счетчика, представленной на рис. 3.33,*а*. Схема имеет следующие особенности:

входы *J* и *K* в каждом триггере ТТ объединены, и на эти входы подан уровень *лог.1*, таким образом, в каждом триггере синхронизирующий вход *C* является счетным входом триггера;

сигнал с прямого выхода триггера каждого разряда поступает на счетный вход *C* триггера следующего, более старшего разряда, а на счетный вход триггера 1 первого разряда подаются входные импульсы.

Если на счетном входе *C* триггера действует импульс, то его положительным фронтом переключается ведущая часть триггера, отрицательным — ведомая. Итак, при каждом изменении сигнала на счетном входе с уровня *лог.1* на уровень *лог.0* изменяется на противоположное состояние выхода триггера. Таким образом, при отрицательном фронте сигнала на выходе триггера происходит переключение следующего за ним триггера более старшего разряда. На рис. 3.33,*б* показана временная диаграмма работы данного счетчика.

С каждым входным импульсом число в счетчике увеличивается на единицу. Такое нарастание числа происходит до тех пор, пока после $(2^n - 1)$-го входного импульса ($n$ — число разрядов в счетчике) в счетчике не устанавливается двоичное число 11...1. Далее с приходом $2^n$-го импульса в счетчике устанавливается исходное состояние 00...0, после чего счет ведется сначала. Таким образом, при непрерывной подаче на вход

импульсов счетчик циклически с периодом $2^n$ входных импульсов устанавливается в исходное состояние.

### Вычитающий и реверсивный счетчики

В *вычитающем счетчике* поступление на вход очередного уровня *лог.1* (очередного импульса) вызывает уменьшение хранившегося в счетчике числа на единицу.

Покажем примеры такого вычитания единицы:

| | | |
|---|---|---|
| Переносы | | 1 1 1 |
| Исходное число | 1 1 0 0 1 | 1 1 0 0 0 |
| | – 1 | – 1 |
| Результат | 1 1 0 0 0 | 1 0 1 1 1 |

Из первого примера видно, что если в младшем разряде числа содержится 1, то получающееся в результате вычитания 1 число отличается от исходного лишь в младшем разряде.

Если в младшем разряде числа содержится 0, то вычитание сопровождается возникновением переносов. В отличие от суммирования, при котором перенос прибавляется в разряд, в который он поступает, в вычитании перенос имеет смысл заема из следующего, более старшего разряда и вычитается из этого разряда. Последовательная передача таких заемов из разряда в разряд продолжается до тех пор, пока в очередном разряде, в который передается заем, не обнаруживается 1.

Так, во втором из приведенных выше примеров такая 1 обнаруживается в четвертом разряде. В результате заема этой 1 в четвертом разряде образуется 0, а занятая из этого разряда 1 передается в третий разряд, где она имеет уже вес 2. Из этих двух единиц в третьем разряде остается одна, а другая передается во второй разряд, где она приобретает также вес 2 и т.д.

Таким образом, в результате вычитания часть числа левее первого из разрядов, содержащих 1, остается неизменной, цифры остальных разрядов инвертируются.

Функционирование $i$-го разряда счетчика при выполнении операции вычитания единицы представлено в табл. 3.14.

*Таблица 3.14*

| $a_i$ | $p_i$ | $c_i$ | $p_{i+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

Из этой таблицы истинности следуют логические выражения

$$c_i = a_i \cdot \overline{p}_i \vee \overline{a}_i \cdot p_i,\ p_{i+1} = \overline{a}_i \cdot p_i.$$

Цифры разрядов $c_i$ определяются тем же логическим выражением, что и в суммирующем

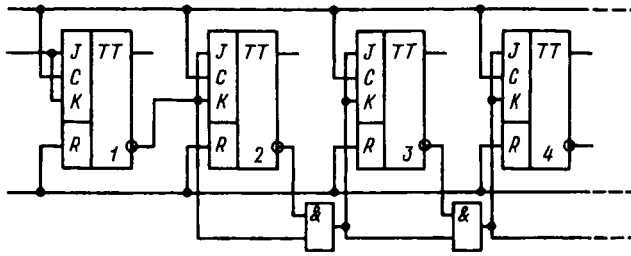



Рис. 3.34

счетчике. Следовательно, как и в суммирующем счетчике, перенос должен подаваться на счетный вход, образованный соединением информационных входов J и K триггера. Отличие выражения $p_{i+1}$ от соответствующего выражения суммирующего счетчика состоит в том, что вместо $a_i$ использовано $\bar{a}_i$. Таким образом, в вычитающем счетчике на элементы И, формирующие переносы, подаются сигналы с инверсных выходов триггеров.

На рис. 3.34 показана схема вычитающего счетчика с последовательной передачей переносов. Для повышения скорости работы счетчика могут быть использованы последовательно-параллельные цепи передачи переносов. Вычитающий счетчик , как и суммирующий, имеет период циклической работы, равный $2^n$ импульсов.

*Реверсивный счетчик* допускает в процессе работы переключение из режима суммирования в режим вычитания и наоборот. На рис.3.35 приведена схема такого счетчика. В ней предусмотрены две цепи передачи переносов, одна из которых соответствует схеме суммирующего

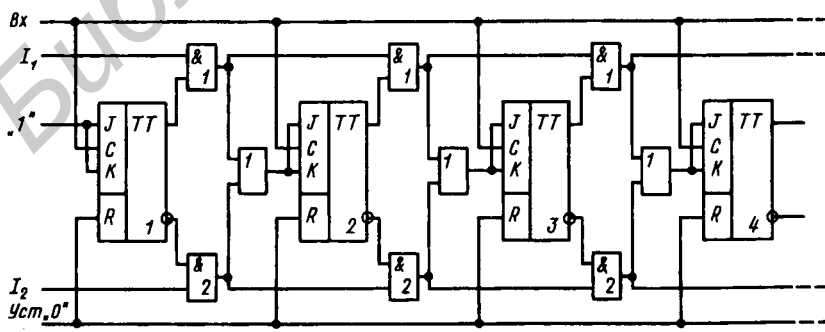



Рис. 3.35

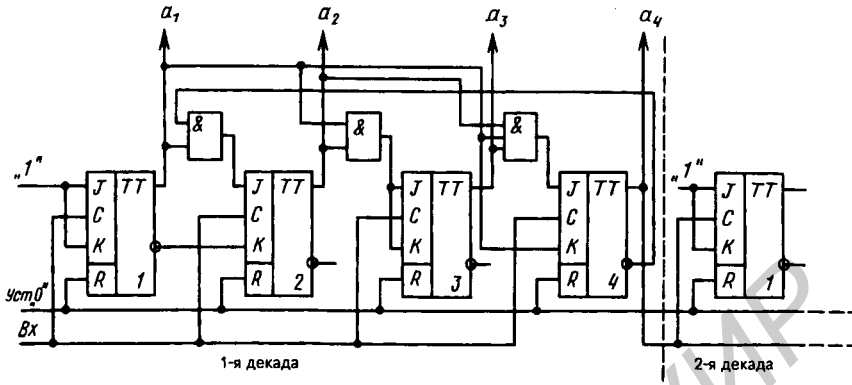



Рис. 3.36

счетчика, другая — схеме вычитающего счетчика. Управляющие сигналы $I_1$ и $I_2$ включают в работу одну или другую цепь.

При $I_1 = 1$ и $I_2 = 0$ оказывается закрытым элемент И2 и, следовательно, отключена цепь передачи переносов режима вычитания. Счетчик работает в режиме суммирования. При $I_1 = 0$ и $I_2 = 1$ закрыт элемент И1 и отключена, таким образом, цепь передачи переносов режима суммирования, счетчик работает в режиме вычитания.

### Десятичный счетчик

На рис. 3.32 была показана структура десятичного счетчика. Каждый десятичный разряд такого счетчика — *декада* — является двоичным счетчиком с периодом цикла $N = 10$.

На рис. 3.36 приведена схема декады и показана связь со следующей декадой. Как видно из схемы, входными импульсами следующей декады являются импульсы, возникающие на выходе триггера старшего разряда данной декады. В момент отрицательного фронта десятого импульса, поступающего на вход данной декады, триггеры этой декады переходят в состояние 0, на выходе триггера четвертого разряда возникает изменение уровня от *лог.1* до *лог.0*. Это вызывает переход следующей декады в состояние, соответствующее двоичному числу, на единицу большему. Убедитесь самостоятельно в правильности функционирования данной схемы.

Десятичные счетчики находят широкое применение в тех случаях, когда число поступающих импульсов необходимо представлять в привычной для человека десятичной системе счисления.

### Кольцевой счетчик

В рассмотренных выше счетчиках число поступлений на вход импульсов представляется в форме двоичного числа, цифры разрядов которого выражаются через состояния

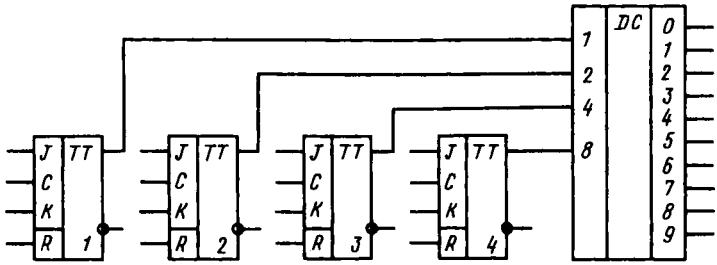



Рис. 3.37

триггеров. При этом, если требуется получить десятичное представление числа импульсов, к выходам счетчика подключается дешифратор.

На рис. 3.37 показано подключение дешифратора к декаде десятичного счетчика. В этой схеме уровень *лог.1* появляется на том из выходов дешифратора, десятичный номер которого соответствует двоичному числу в счетчике. В процессе счета с каждым поступлением на вход импульса происходит переход *лог.1* на следующий выход, номер которого на единицу больше.

Неудобства, связанные с необходимостью применения дешифратора, устраняются в кольцевом счетчике: в нем число поступлений импульсов выражается непосредственно в десятичной системе счисления.

*Кольцевой счетчик* строится в виде сдвигового регистра, в котором выдвигаемая из старшего разряда информация вводится в младший разряд. Схема счетчика показана на рис. 3.38. В счетчике использовано $N$ триггеров ТТ. Перед началом счета импульсом начальной установки триггер 0 устанавливается в состояние 1, остальные триггеры устанавливаются в состояние 0. Этому состоянию счетчика соответствует число 0. На выход счетчика, обозначенный цифрой 0, с прямого выхода триггера 0 передается *лог.1*. Далее каждый из приходящих на вход импульсов переписывает в счетчике 1 в следующий триггер и *лог.1* передается на следующий выход, обозначенный цифрой, на единицу большей. Таким образом, по тому, какой из триггеров находится в состоянии 1, т.е. на выходе какого из триггеров возникает уровень *лог.1*, выявляется число поступивших на вход импульсов непосредственно в десятичной системе счисления.

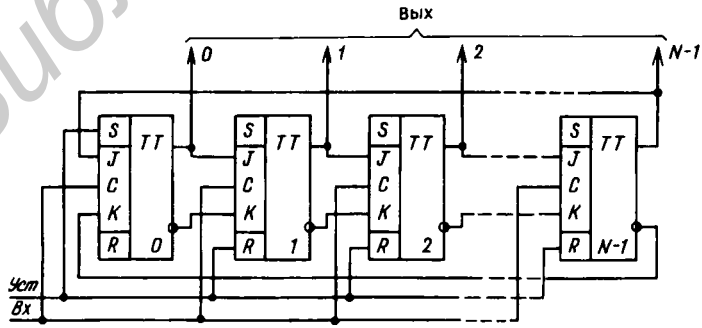



Рис. 3.38

Кольцевой счетчик обеспечивает высокую скорость работы. Это связано с тем, что единица из одного триггера в другой передается непосредственно (без использования в цепи передачи логических элементов) путем подключения входов $J$ и $K$ каждого триггера соответственно к прямому и инверсному выходам предыдущего триггера.

После подачи $N-1$ импульсов в состоянии 1 окажется $(N-1)$-й триггер, а с приходом $N$-го импульса единица из этого триггера перепишется в триггер 0 и счет импульсов начнется сначала. Следовательно, период цикла кольцевого счетчика равен числу использованных в нем триггеров. Например, для построения декады десятичного счетчика потребуется 10 триггеров (вместо четырех триггеров в двоичном счетчике). Таким образом, возможность построения счетчика, выдающего числа непосредственно в десятичной системе счисления, достигается существенным увеличением числа используемых в схеме счетчика элементов.

### Делители частоты импульсной последовательности

*Делитель частоты* — устройство, которое при подаче на его вход периодической последовательности импульсов формирует на выходе такую же последовательность, но имеющую частоту повторения импульсов, в некоторое число раз меньшую, чем частота импульсов входной последовательности.

Отличие делителей частоты от счетчиков состоит в следующем. В счетчике каждая комбинация состояний триггеров определяет в некоторой системе счисления число импульсов, поступивших к данному моменту времени. В делителе частоты последовательность состояний может быть выбрана произвольной, важно лишь обеспечить заданный период цикла $N$. Последовательность состояний выбирается из соображений обеспечения при заданном $N$ наибольшей простоты межтриггерных связей. Эти связи должны выполняться непосредственным соединением выходов одних триггеров со входами других без логических элементов. Счетчик, имеющий то же значение $N$, может исполнять роль делителя частоты, однако следует иметь в виду, что такое решение будет неэкономичным.

Рассмотрим схемы делителей частоты с различными коэффициентами деления $N$.

**Делитель частоты с коэффициентом деления $N = 2$.** Схема делителя приведена на рис. 3.39,*а*. В моменты отрицательного фронта входных импульсов триггер переключается в новое состояние. Как видно из временно́й диаграммы на рис. 3.39,*б*, период импульсной последовательности $T_{вых}$ на выходе триггера оказывается вдвое больше периода следования импульсов на входе. Следовательно, $f_{вых} = 1/T_{вых} = 1/(2T_{вх}) = f_{вх}/2$, т.е. частота следования импульсов на выходе в два раза ниже, чем на входе.

**Делитель частоты с коэффициентом деления $N = 2^n$.** На рис. 3.40,*а* показано последовательное соединение делителей частоты с коэффициентом деления, равным двум, при котором выход каждого из делителей подключен к входу следующего. На выходе каждого делителя частота следования импульсов вдвое ниже, чем на входе. Так, если частота

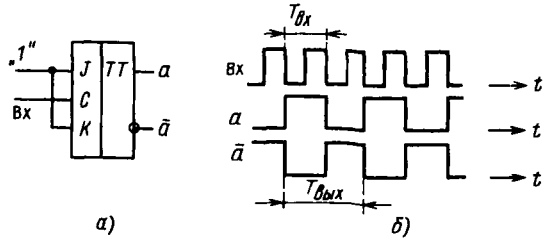



Рис. 3.39

следования импульсов на входе первого делителя $f_{вх}$, то на выходе первого делителя $f_{вых1} = f_{вх}/2$, на выходе второго $f_{вых2} = f_{вых1}/2 = f_{вх}/2^2$, на выходе третьего $f_{вых3} = f_{вых2}/2 = f_{вх}/2^3$ и т.д. При $n$ каскадах подобного деления частота выходной последовательности окажется равной $f_{вых} = f_{вх}/2^n$, т.е. будет осуществляться деление частоты в $N = 2^n$ раз.

**Делитель частоты с коэффициентом деления $N = 3$.** Счетчик с периодом цикла $N = 3$ имеет простейшие межтриггерные связи без логических элементов. Этот счетчик может одновременно служить и делителем частоты с коэффициентом $N = 3$.

**Делитель частоты с коэффициентом деления $N = 5$.** Схема делителя и временна́я диаграмма работы приведены на рис. 3.40, таблица состояний — в табл. 3.15.

*Таблица 3.15*

| Номер входного импульса | Состояние триггеров | | | | | |
|---|---|---|---|---|---|---|
| | текущее | | | следующее | | |
| | $a_3$ | $a_2$ | $a_1$ | $a_3$ | $a_2$ | $a_1$ |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 |
| 3 | 0 | 1 | 1 | 1 | 1 | 0 |
| 4 | 1 | 1 | 0 | 1 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 1 | 0 | 1 | 1 |
| ... | ... | ... | ... | ... | ... | ... |

Как видно из временно́й диаграммы, на выходах триггеров всегда образуется последовательность импульсов с частотой в пять раз более низкой, чем частота импульсов на входе делителя.

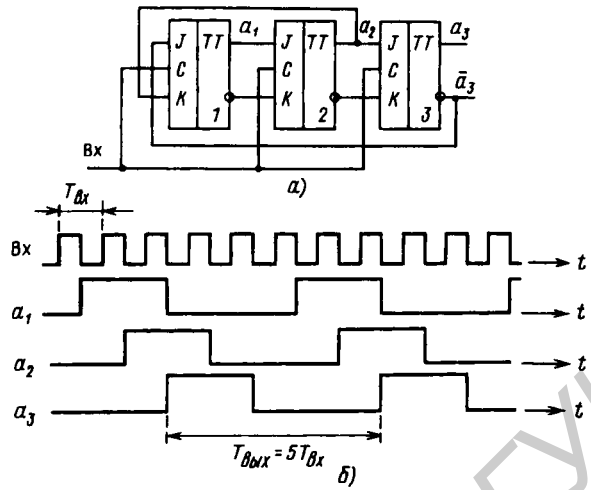



Рис. 3.40

**Каскадные делители частоты**. В тех случаях, когда коэффициент деления $N$ не является простым числом и может быть представлен произведением вида $N = N_1 \cdot N_2 \cdot N_3 \cdot \ldots \cdot N_k$, схема делителя строится в виде каскадного соединения делителей, имеющих коэффициенты деления $N_1, N_2, N_3, \ldots, N_k$ (рис. 3.41).

Примером такого каскадного построения делителей является рассмотренный ранее делитель с коэффициентом $2^n$ (см. рис. 3.33).

Построим делитель с коэффициентом деления $N = 6$. Этот коэффициент можно представить произведением $N = 3 \cdot 2$. Таким образом, данный делитель может быть построен в виде каскадного соединения делителей, имеющих коэффициенты деления $N_1 = 3$ и $N_2 = 2$. Схема делителя показана на рис. 3.42.

Чтобы построить делитель с коэффициентом деления $N = 9$, представим этот коэффициент в форме $N = 3 \cdot 3$ и реализуем каскадным соединением двух делителей с коэффициентами $N_1 = N_2 = 3$ (рис. 3.43).

Рассмотрим еще пример: пусть коэффициент $N = 10$. Представим $N = N_1 \cdot N_2 = 5 \cdot 2$, при этом схема делителя строится в виде каскадного

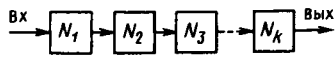



Рис.3.41

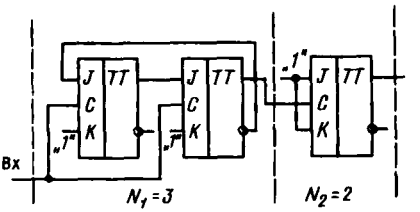



Рис. 3.42

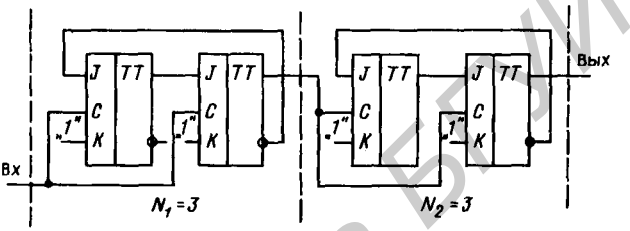



Рис. 3.43

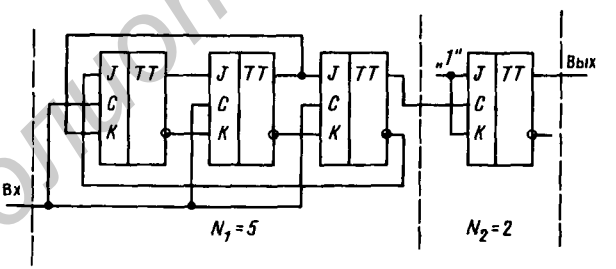



Рис.3.44

соединения делителей с коэффициентами $N_1 = 5$ и $N_2 = 2$, как показано на рис. 3.44.

Таким образом, имея набор схем, реализующих коэффициенты деления, которые представляют собой простые числа, можно их каскадным соединением получать делители с разнообразными коэффициентами деления.

### *Упражнения*

1. Постройте счетчики со следующими значениями периода:

а) $N = 7$; б) $N = 9$; в) $N = 11$; г) $N = 12$; д) $N = 21$; е) $N = 14$; ж) $N = 15$.

2. Постройте делители частоты со следующими значениями коэффициента деления :

а) $N = 14$; б) $N = 15$; в) $N = 18$; г) $N = 20$; д) $N = 21$; е) $N = 24$; ж) $N = 25$.

## 3.6. СУММАТОРЫ

### Одноразрядный двоичный сумматор

Из рассмотренного в § 2.2 принципа сложения многоразрядных двоичных чисел следует, что в каждом из разрядов производятся однотипные действия: определяется цифра суммы путем сложения по модулю 2 цифр слагаемых и поступающего в данный разряд переноса и формируется перенос, передаваемый в следующий разряд. Эти действия реализуются *одноразрядным двоичным сумматором*. Символическое изображение такого сумматора показано на рис. 3.45,*а*. Он имеет три входа для подачи цифр разрядов слагаемых $a_i$, $b_i$ и переноса $p_i$; на выходах формируются сумма $s_i$ и перенос $p_{i+1}$, предназначенный для передачи в следующий разряд. В одноразрядном сумматоре могут предусматриваться входы для подачи как прямых $a_i$, $b_i$, $p_i$, так и инверсных значений $\bar{a}_i$, $\bar{b}_i$, $\bar{p}_i$ входных переменных. Пример такого одноразрядного сумматора приведен на рис. 3.45,*б*.

В табл. 3.16 показано функционирование одноразрядного сумматора.

Пользуясь этой таблицей истинности, запишем логические выражения для выходных величин $s_i$ и $p_i$ в базисе И-ИЛИ-НЕ:

$$s_i = \overline{\bar{a}_i \cdot \bar{b}_i \cdot \bar{p}_i \vee a_i \cdot b_i \cdot \bar{p}_i \vee \bar{a}_i \cdot b_i \cdot p_i \vee a_i \cdot \bar{b}_i \cdot p_i},$$

$$p_{i+1} = \overline{\bar{a}_i \cdot \bar{b}_i \vee \bar{a}_i \cdot \bar{p}_i \vee \bar{b}_i \cdot \bar{p}_i}.$$

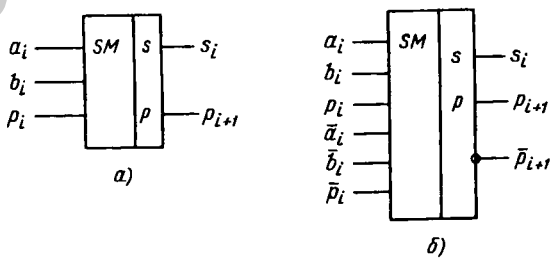



Рис. 3.45

*Таблица 3.16*

| Входы | | | Выходы | |
|---|---|---|---|---|
| Слагаемые | | Перенос | Сумма | Перенос |
| $a_i$ | $b_i$ | $p_i$ | $s_i$ | $p_{i+1}$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

На рис. 3.46 приведена схема сумматора, построенного с использованием логических выражений.

### Многоразрядные двоичные сумматоры

В зависимости от способа ввода кодов слагаемых сумматоры делятся на два типа: *последовательного* и *параллельного действия*. В сумматоры первого типа коды чисел вводятся в последовательной форме, т.е. разряд за разрядом (младшим разрядом вперед), в сумматоры второго типа каждое слагаемое подается в параллельной форме, т.е. одновременно всеми разрядами.

**Сумматор последовательного действия** (рис. 3.47). Состоит из одноразрядного сумматора, выход $p_{i+1}$ которого соединен с входом $p_i$ через D-триггер. Изображенные на рисунке сдвиговые регистры RG не входят непосредственно в схему сумматора, они служат для подачи на вход сумматора разрядов слагаемых (регистры 1 и 2) и приема выдаваемых сумматором разрядов суммы (регистр 3). Операция суммирования во всех разрядах слагаемых осуществляется с помощью одного и того же одноразрядного сумматора. Такое построение сумматора возможно за счет того, что слагаемые поступают в последовательной форме.

С первым тактовым импульсом на входы сумматора поступают из регистров 1 и 2 цифры первого разряда слагаемых $a_1$ и $b_1$, из D-триггера на вход $p_i$ подается уровень *лог.0*. Суммируя поданные на входы цифры, одноразрядный сумматор формирует первый разряд суммы $S_1$, выдаваемый на вход регистра 3, и перенос $p_2$, принимаемый в D-триггер. Второй тактовый импульс осуществляет в регистрах сдвиг на один разряд вправо; при этом на входы одноразрядного сумматора подаются цифры

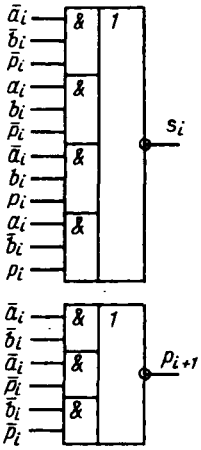



Рис. 3.46

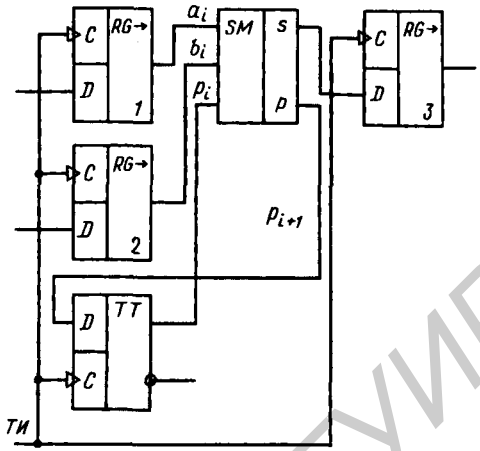



Рис. 3.47

второго разряда слагаемых $a_2$, $b_2$ и перенос $p_2$, получающаяся цифра второго разряда суммы вдвигается в регистр 3, перенос $p_3$ принимается в триггер и т.д.

Очевидное достоинство сумматора последовательного действия заключается в малом объеме оборудования, требуемого для его построения. Однако связанная с этим необходимость в последовательной обработке разрядов приводит к низкому быстродействию.

**Сумматор параллельного действия.** Состоит из отдельных разрядов, каждый из которых содержит одноразрядный сумматор (рис.3.48).

При подаче слагаемых цифры их разрядов поступают на соответствующие одноразрядные сумматоры. Каждый из одноразрядных сумматоров формирует на своих выходах цифру соответствующего разряда суммы и перенос, передаваемый на вход одноразрядного сумматора следующего, более старшего разряда.

**Повышение быстродействия параллельных сумматоров.** Для обеспечения высокого быстродействия параллельные сумматоры должны строиться на элементах, также имеющих высокое быстродействие. Трудности в достижении высокого быстродействия сумматора, построенного по схеме на рис. 3.48, связаны с тем, что процесс распространения переносов в нем носит последовательный характер. Импульс переноса в каждом разряде формируется после того, как будет сформирован и передан импульс переноса из предыдущего разряда. В наиболее неблагоприятном случае возникший в младшем разряде перенос может последовательно вызывать переносы во всех остальных разрядах. При этом

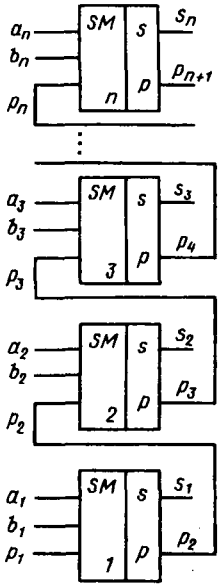



Рис. 3.48

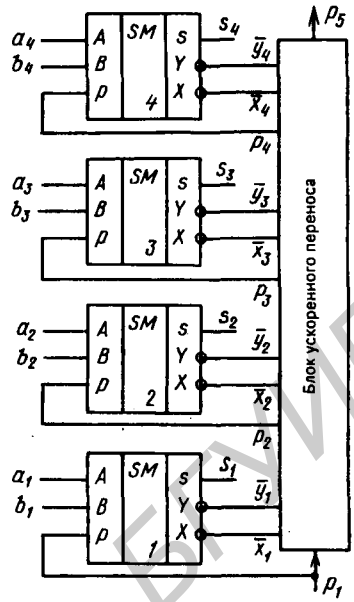



Рис. 3.49

время передачи переносов $\tau = n\tau_1$, где $\tau_1$ — задержка распространения переноса в одном разряде.

Уменьшение $\tau$ достигается следующими приемами.

1. При построении схем одноразрядных сумматоров стремятся к уменьшению числа элементов в цепи между входом, на который поступает импульс переноса $p_i$, и выходом, на котором формируется передаваемый в следующий разряд импульс переноса $p_{i+1}$. Этот принцип реализован в схеме сумматора на рис. 3.46, в которой цепь от $p_i$ к $p_{i+1}$ содержит один логический элемент И-ИЛИ-НЕ.

2. В цепях от $p_i$ к $p_{i+1}$ применяют элементы с повышенным быстродействием.

3. Схемы сумматора следует строить таким образом, чтобы сигналы с выхода каждого логического элемента в цепи от $p_i$ к $p_{i+1}$ поступали на возможно меньшее число других логических элементов, так как присоединение каждого дополнительного элемента к той или иной точке цепи переносов, как правило, приводит к увеличению паразитной емкости и длительности фронтов сигналов и, следовательно, к увеличению задержки распространения сигнала и снижению быстродействия сумматора.

4. Применяют устройства формирования переносов в параллельной форме. В показанном на рис. 3.49 сумматоре с помощью устройства, назы-

ваемого *блоком ускоренного переноса*, производится формирование переносов в параллельной форме, т.е. одновременно для всех разрядов. Переносы из этого блока поступают во все разряды сумматора одновременно. При этом разрядные сумматоры не содержат цепей формирования переносов, они формируют только сумму $s_i$ и величину $y_i$ и $x_i$, для получения которых переносы не требуются. Эти величины $y_i$ и $x_i$ необходимы для формирования переносов в блоке ускоренного переноса, они определяют следующие ситуации: $y_i = 1$ означает, что в $i$-м разряде перенос $p_{i+1}$ в следующий $(i + 1)$-й разряд необходимо формировать независимо от поступления в данный разряд переноса из предыдущего разряда; $x_i = 1$ означает, что в $i$-м разряде перенос $p_{i+1}$ должен формироваться только при условии поступления переноса $p_i$ из предыдущего разряда.

Рассмотрим принцип построения блока ускоренного переноса. Перенос $p_2$ во второй разряд должен формироваться при условии $y_1 = 1$ или при условии $x_i = 1$ и наличии переноса на входе $p_1$, т.е. $p_2 = y_1 \vee x_1 \cdot p_1$. После преобразований получим

$$p_2 = \overline{\overline{y_1 \vee x_1 \cdot p_1}} = \overline{\overline{y}_1 \cdot (\overline{x}_1 \vee \overline{p}_1)} = \overline{\overline{y}_1 \cdot \overline{x}_1 \vee \overline{y}_1 \cdot \overline{p}_1}.$$

Аналогичные выражения можно построить для переносов в другие разряды:

$$p_3 = y_2 \vee x_2 \cdot p_2 = y_2 \vee x_2 \cdot \underbrace{(y_1 \vee x_1 \cdot p_1)}_{p_2} = \overline{\overline{y_2 \vee y_1 \cdot x_2 \vee x_2 \cdot x_1 \cdot p_1}} =$$

$$= \overline{\overline{y}_2 \cdot (\overline{y}_1 \vee \overline{x}_2) \cdot (\overline{x}_2 \vee \overline{x}_1 \vee \overline{p}_1)} = \overline{\overline{y}_2 \cdot (\overline{x}_2 \vee \overline{y}_1 \cdot \overline{x}_1 \vee \overline{y}_1 \cdot \overline{p}_1)} =$$

$$= \overline{\overline{y}_2 \cdot \overline{x}_2 \vee \overline{y}_2 \cdot \overline{y}_1 \cdot \overline{x}_1 \vee \overline{y}_2 \cdot \overline{y}_1 \cdot \overline{p}_1}.$$

Предлагаем самостоятельно убедиться в справедливости следующих выражений:

$$p_4 = \overline{\overline{y}_3 \cdot \overline{x}_3 \vee \overline{y}_3 \cdot \overline{y}_2 \cdot \overline{x}_2 \vee \overline{y}_3 \cdot \overline{y}_2 \cdot \overline{y}_1 \cdot \overline{x}_1 \vee \overline{y}_3 \cdot \overline{y}_2 \cdot \overline{y}_1 \cdot \overline{p}_1},$$

$$p_5 =$$

$$= \overline{\overline{y}_4 \cdot \overline{x}_4 \vee \overline{y}_4 \cdot \overline{y}_3 \cdot \overline{x}_3 \vee \overline{y}_4 \cdot \overline{y}_3 \cdot \overline{y}_2 \cdot \overline{x}_2 \vee \overline{y}_4 \cdot \overline{y}_3 \cdot \overline{y}_2 \cdot \overline{y}_1 \cdot \overline{x}_1 \vee \overline{y}_4 \cdot \overline{y}_3 \cdot \overline{y}_2 \cdot \overline{y}_1 \cdot \overline{p}_1}.$$

На рис. 3.50,*а* и *б* показаны схема блока ускоренного переноса и его условное обозначение.

Входящие в выражения $p_2$ ,..., $p_5$ величины $y_i$, $x_i$ формируются одновременно во всех разрядных сумматорах, одновременно поступают на

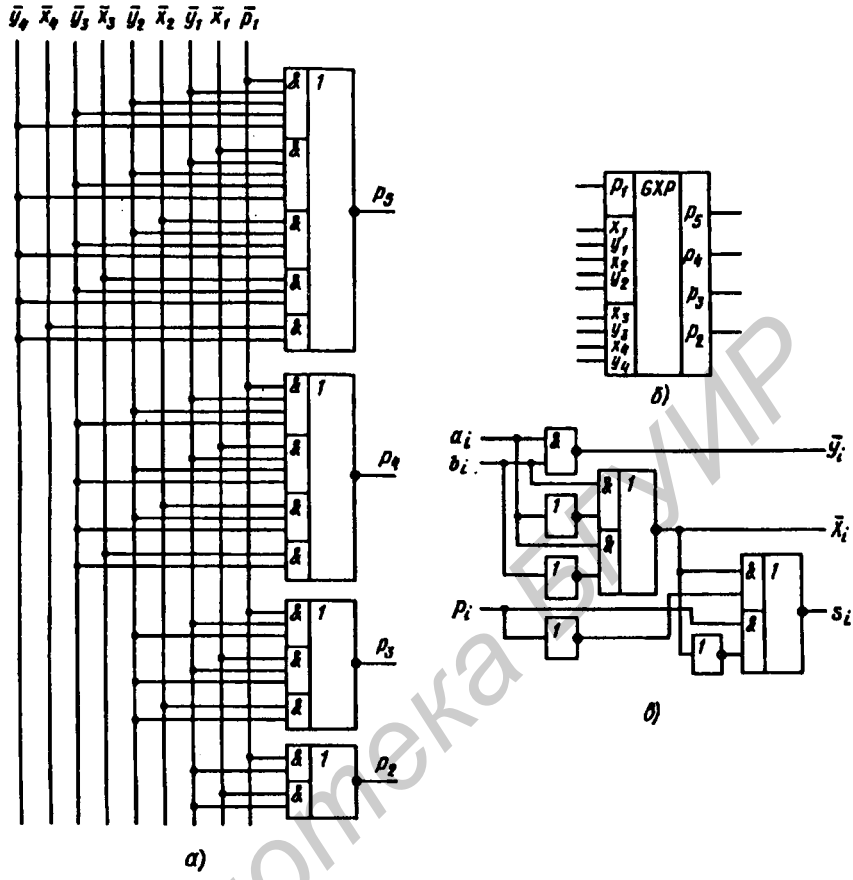



Рис. 3.50

входы блока ускоренного переноса, и, следовательно, в этом блоке одновременно формируются переносы, подаваемые в разрядные сумматоры. После поступления переносов из блока ускоренного переноса в разрядных сумматорах формируются суммы $\underline{s_i}$.

Формирование инверсных значений $\overline{y_i}$ и $\overline{x_i}$ и суммы $s_i$ в разрядном сумматоре может быть выполнено по следующим логическим выражениям:

$$\overline{y}_i = \overline{a_i \cdot b_i}, \qquad \overline{x}_i = \overline{\overline{a_i \cdot b_i} \vee \overline{a}_i \cdot b_i}, \quad s_i = x_i \oplus p_i = \overline{\overline{x_i \cdot p_i} \vee \overline{x}_i \cdot \overline{p}_i}.$$

Схема разрядного сумматора, построенного в соответствии с этими выражениями, показана на рис. 3.50,*в*.

### Десятичные сумматоры

Для построения многоразрядных двоичных сумматоров, как было показано выше, необходимы одноразрядные двоичные сумматоры. Аналогично многоразрядные десятичные сумматоры строятся с использованием одноразрядных десятичных сумматоров. Последние выполняют операцию суммирования десятичных цифр $a_i$, $b_i$ и переноса $p_i$, поступающих в разряд, и формируют на выходах десятичную цифру суммы $s_i$ и перенос $p_{i+1}$ для передачи в следующий десятичный разряд.

При использовании десятичной системы счисления цифры разрядов десятичного числа представляются в двоичной форме (см. § 2.2). В связи с этим одна из особенностей одноразрядных десятичных сумматоров связана с тем, что суммируемые десятичные цифры $a_i$ и $b_i$ представляются многоразрядными двоичными числами (переносы $p_i$ независимо от используемой системы счисления могут иметь лишь значения 0 либо 1). Рассмотрим построение одноразрядного сумматора десятичных цифр, представляемых в *коде 8421*.

**Сумматор для** *кода 8421*. В работе сумматора этого типа имеются особенности в формировании переноса и суммы по сравнению с работой двоичного сумматора.

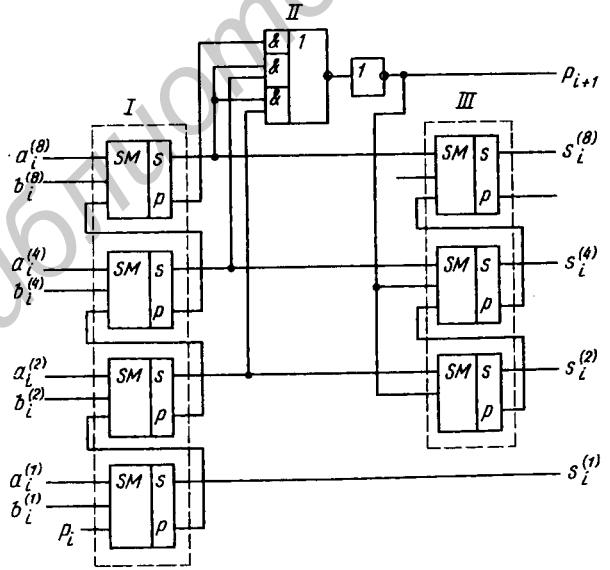



Рис. 3.51

Правила сложения в десятичной системе счисления с использованием *кода 8421* рассмотрены в § 2.3. Построенная в соответствии с этими правилами схема одноразрядного десятичного сумматора (рис. 3.51) включает четырехразрядный двоичный сумматор I, схему формирования переноса $p_{i+1}$ в следующий десятичный разряд II и схему коррекции суммы III. Последняя представляет собой трехразрядный сумматор, в котором при $p_{i+1} = 1$ производится прибавление единицы в разрядах нескорректированной суммы с весовыми коэффициентами 2 и 4.

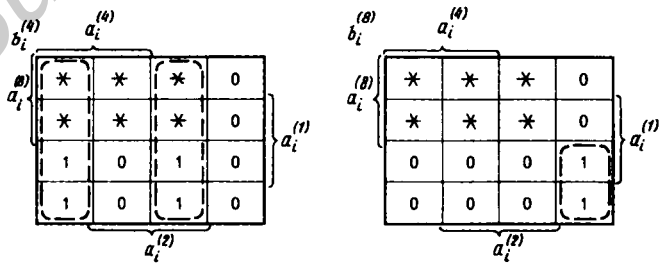
Операция суммирования в случае, когда либо одно слагаемое, либо оба слагаемых имеют отрицательные значения, может производиться с представлением таких слагаемых в обратном коде.

**Схема формирования обратного кода.** В десятичной системе счисления обратный код образуется путем преобразования каждой цифры числа в дополнение до 9. В табл. 3.17 приведены для десятичных цифр 0, 1, ... , 9 прямые коды и соответствующие им обратные коды.

*Таблица 3.17*

| Десятичная цифра | Прямой *код 8421* | | | | Обратный *код 8421* | | | |
|---|---|---|---|---|---|---|---|---|
| | $a_i^{(8)}$ | $a_i^{(4)}$ | $a_i^{(2)}$ | $a_i^{(1)}$ | $b_i^{(8)}$ | $b_i^{(4)}$ | $b_i^{(2)}$ | $b_i^{(1)}$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

*Таблица 3.18*



Из сопоставления приведенных в таблице значений $a_i^{(2)}$, $a_i^{(1)}$ и соответствующих им $b_i^{(2)}$, $b_i^{(1)}$ нетрудно заключить, что

$$b_i^{(1)} = \overline{a}_i^{(1)}, \quad b_i^{(2)} = a_i^{(2)}.$$

Логические выражения для $b_i^{(4)}$ и $b_i^{(8)}$ можно получить из карт Вейча (табл. 3.18):

$$b_i^{(4)} = a_i^{(4)} \cdot \overline{a}_i^{(2)} \vee \overline{a}_i^{(4)} \cdot a_i^{(2)},$$

$$b_i^{(8)} = \overline{a}_i^{(8)} \cdot \overline{a}_i^{(4)} \cdot \overline{a}_i^{(2)}.$$

На рис. 3.52 приведена схема, формирующая обратный код по полученным выше логическим выражениям.

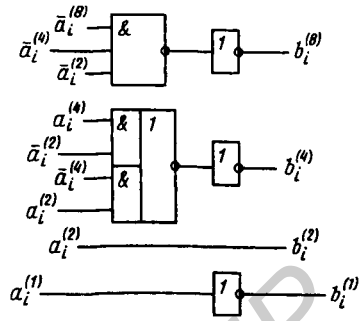



Рис. 3.52

## 3.7. ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ УСТРОЙСТВА С МАТРИЧНОЙ СТРУКТУРОЙ

В настоящее время для построения логических устройств может быть использован универсальный элемент, называемый *программируемой логической матрицей* (ПЛМ). Такая матрица может быть настроена (запрограммирована) на выполнение любой логической функции определенной сложности.

На рис. 3.53 показана структура ПЛМ. Цепи входных переменных $x_1$, $x_2$, ... и их инверсий $\overline{x}_1$, $\overline{x}_2$, ... составляют горизонтальные цепи матрицы $M_1$, вертикальными цепями которой служат так называемые *цепи конъюнкции*. Другую матрицу $M_2$ образуют цепи конъюнкции с горизонтальными цепями выходов $y_1$, $y_2$, ... В узлах матрицы $M_1$ включены элементы, с помощью которых на цепях конъюнкции могут формироваться любые требуемые конъюнкции входных переменных, имеющиеся в узлах матрицы $M_2$ элементы позволяют формировать на выходных цепях любые требуемые дизъюнкции функций, полученных на цепях конъюнкций. В процессе программирования ПЛМ в узлах матриц $M_1$ и $M_2$ производят подключение элементов, которые необходимы для реализации требуемых выходных логических функций $y_1$, $y_2$, ... В зависимости от того, прямая или инверсная функция реализуется, в выходные цепи могут включаться инверторы.

Матрица $M_1$ содержит горизонтальные цепи, на которых действуют входные переменные $x_1$, $x_2$, ... и их инверсии $\overline{x}_1$, $\overline{x}_2$, ..., и вертикальные цепи, на которых формируются коньюнкции $p_1, p_2$, ... В отдельных узлах матрицы между ее вертикальными и горизонтальными цепями включены диоды. На вертикальной цепи образуется высокий потенциал (уровень *лог.1*) в том случае, когда на всех входах, идущих к узлам, содержащим диоды, действует высокий потенциал (уровень *лог.1*), закрывающий диоды. Если хотя бы на одном из таких входов низкий

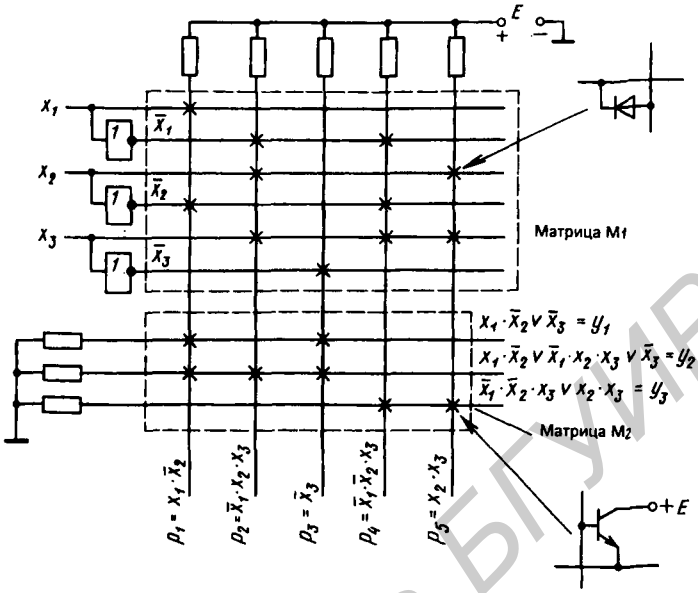



Рис. 3.53

потенциал (уровень *лог.0*), открывается диод и уровень *лог.0* с этого входа через открытый диод передается на вертикальную цепь матрицы.

На рис. 3.53 крестиками показаны участки, в которых в процессе программирования создаются соединения. Таким образом, в этой схеме

$$p_1 = x_1 \cdot \bar{x}_2,\ p_2 = \bar{x}_1 \cdot x_2 \cdot x_3,\quad p_3 = \bar{x}_3,\quad p_4 = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3,\quad p_5 = x_2 \cdot x_3.$$

Включая в соответствующие узлы диоды, можно на выводах $p_i$ сформировать любые конъюнкции входных переменных и их инверсий.

В узлах матрицы $M_2$ между цепями $p_i$ и $y_j$ включены транзисторы, базы которых подключены к цепям $p_i$, а эмиттеры — к цепям $y_j$. Если в цепи $p_i$ действует высокий потенциал (уровень *лог.1*), транзистор оказывается в открытом состоянии и высокий потенциал через открытый транзистор передается в цепь $y_j$ и $y_j = 1$ независимо от уровней на других выходах матрицы $M_1$.

Таким образом, в схеме на рис. 3.53

$$y_1 = x_1 \cdot \bar{x}_2 \vee \bar{x}_3,\ \ y_2 = x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2 \cdot x_3 \vee \bar{x}_3,\quad y_3 = \bar{x}_1 \cdot \bar{x}_2 \cdot x_3 \vee x_2 \cdot x_3.$$

Программа функционирования приведенной на рис. 3.53 ПЛМ может быть представлена табл. 3.19.

*Таблица 3.19*

| | $x_1$ | $x_2$ | $x_3$ | $y_1$ | $y_2$ | $y_3$ |
|---|---|---|---|---|---|---|
| $p_1$ | 1 | 0 | — | 1 | 1 | . |
| $p_2$ | 0 | 1 | 1 | . | 1 | . |
| $p_3$ | — | — | 0 | 1 | 1 | . |
| $p_4$ | 0 | 0 | 1 | . | . | 1 |
| $p_5$ | — | 1 | 1 | . | . | 1 |

Таблица строится по следующему правилу. На пересечении $j$-й строки и столбца $x_i$ записывается 1, если $x_i$ входит в конъюнкцию $p_j$ на выходе матрицы $M_1$ без инверсии, и 0 — если с инверсией, если $x_i$ не входит в $j$-ю конъюнкцию, то ставится прочерк. На пересечении $i$-й строки и столбца $y_k$ записывается 1, если $i$-я элементарная конъюнкция входит в ДНФ $y_k$, и точка в противном случае.

Программирование ПЛМ может осуществляться на заводе в процессе изготовления микросхемы на этапе формирования элементов в узлах матриц. Программирование может выполняться пользователем. В этом случае завод — изготовитель микросхемы выпускает ПЛМ со вставленными элементами во все узлы матриц. Пользователь, пропуская импульсы тока через определенные элементы, пережигает плавкие перемычки, последовательно включенные с элементами, и таким образом отключает их. Остающиеся элементы должны обеспечить требуемые функции на выходах ПЛМ.

В выпускаемых ПЛМ число входов может достигать 24, число выходов — 16, число цепей конъюнкций — 96. Структуры с программированием на заводе—изготовителе широко используются при выпуске БИС.

Ниже рассматриваются типовые узлы цифровых устройств, выполненные на программируемых логических устройствах с матричной структурой.

**Шифратор.** Рассмотрим построение шифратора, преобразующего унитарный десятичный код (с отображением десятичной цифры уровнем *лог.1* на одной из десяти цепей) в двоичный *код 8421*. Воспользуемся полученными в § 3.5 для дешифратора логическими выражениями

$$x_1 = \bar{y}_1 \,|\, \bar{y}_3 \,|\, \bar{y}_5 \,|\, \bar{y}_7 \,|\, \bar{y}_9,\ x_2 = \bar{y}_2 \,|\, \bar{y}_3 \,|\, \bar{y}_6 \,|\, \bar{y}_7,$$

$$x_4 = \bar{y}_4 \,|\, \bar{y}_5 \,|\, \bar{y}_6 \,|\, \bar{y}_7,\ x_8 = \bar{y}_8 \,|\, \bar{y}_9,$$

где $y_i$ — входные сигналы; $x_j$ — выходные сигналы (значения разрядов *кода 8421*).

На рис. 3.54 показана ПЛМ, реализующая функции шифратора (для упрощения схемы здесь и в последующих схемах отсутствуют некоторые элементы схемы рис. 3.53 — источник питания и резисторы).

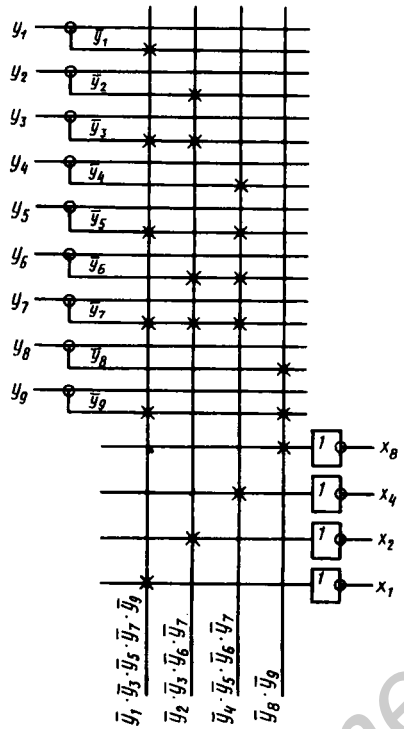



Рис. 3.54

**Дешифратор**. Реализацию на ПЛМ дешифратора рассмотрим на примере дешифратора, преобразующего трехразрядный двоичный код ($x_1$, $x_2$, $x_4$), подаваемый на вход, в унитарный 8-разрядный код на выходе ($y_0$, ... , $y_7$).

Функционирование такого дешифратора определяется следующими логическими выражениями:

$$y_0 = \bar{x}_4 \cdot \bar{x}_2 \cdot \bar{x}_1,\ y_1 = \bar{x}_4 \cdot \bar{x}_2 \cdot x_1,$$

$$y_2 = \bar{x}_4 \cdot x_2 \cdot \bar{x}_1,\ y_3 = \bar{x}_4 \cdot x_2 \cdot x_1,$$

$$y_4 = x_4 \cdot \bar{x}_2 \cdot \bar{x}_1,\ y_5 = x_4 \cdot \bar{x}_2 \cdot x_1,$$

$$y_6 = x_4 \cdot x_2 \cdot \bar{x}_1,\ y_7 = x_4 \cdot x_2 \cdot x_1.$$

Настроенная на реализацию данных функций ПЛМ приведена на рис. 3.55. На рис. 3.56 показана схема мультиплексора с четырьмя входами ($D_3$, $D_2$, $D_1$, $D_0$). Здесь $A_1$, $A_0$ — адресные входы; $C$ — вход для подачи сигнала разрешения выдачи; $y$ — выход.

На рис. 3.57 приведена схема демультиплексора с четырьмя выходами ($y_3$, $y_2$, $y_1$, $y_0$). Здесь $D$ — вход; $A_1$, $A_0$ — адресные входы; $C$ — вход сигнала разрешения выдачи.

**Регистр.** Регистр является устройством с памятью (устройством последовательностного типа). Реализация такого типа устройства на ПЛМ требует наличия в ПЛМ элементов памяти — триггеров, образующих регистр. В такой ПЛМ матрицы $M_1$ и $M_2$ используются для построения комбинационной схемы, с помощью которой регистру придаются дополнительные свойства, кроме простого хранения кода. Одним из таких свойств может быть, например, сдвиг содержимого регистра влево или вправо (на один или несколько разрядов). Рассмотрим построение универсального регистра, обладающего следующими возможностями:

прием извне в регистр кодовой комбинации, поступающей в регистр;

циклический сдвиг содержимого регистра влево или вправо (сдвиг с передачей выдвигаемой из регистра цифры в освобождающий при сдвиге разряд регистра);

сдвиг вправо с приемом в освобождающийся старший разряд регистра цифры, подаваемой на вход;

выдача содержимого регистра на выход.

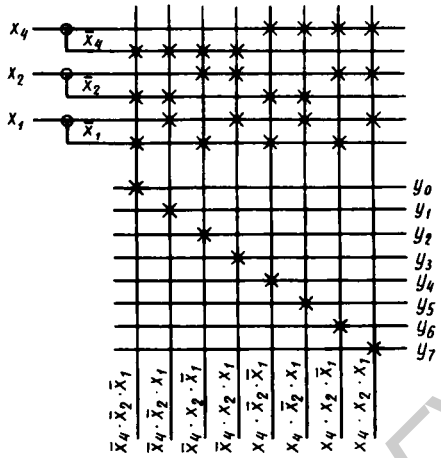



Рис. 3.55

Пусть вид выполняемой в регистре функции задается комбинацией $v_1$, $v_2$, $v_3$ в соответствии с табл. 3.20. Число выполняемых в регистре функций ограничим пятью для того, чтобы схема не оказалась чрезмерно громоздкой и трудной для рассмотрения.

Схема четырехразрядного регистра с указанными функциями представлена на рис. 3.58. Здесь $a_4, a_3, a_2, a_1$ — выводы для подачи входного кода; $y$ — вывод для подачи на вход цифры, вводимой при сдвиге

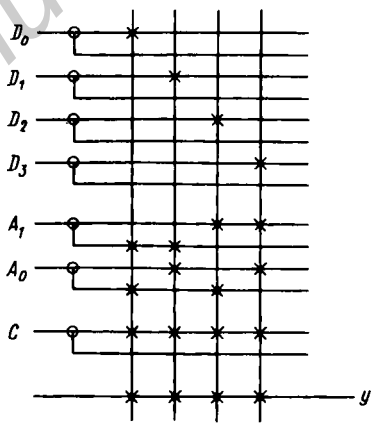



Рис. 3.56

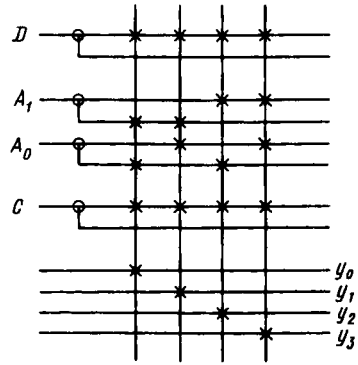



Рис. 3.57

*Таблица 3.20*

| $v_1$ | $v_2$ | $v_3$ | Выполняемая функция |
|---|---|---|---|
| 0 | 0 | 1 | Выдача содержимого регистра |
| 0 | 1 | 1 | Сдвиг вправо с приемом *y* в старший разряд |
| 1 | 0 | 0 | Циклический сдвиг влево |
| 1 | 0 | 1 | Циклический сдвиг вправо |
| 1 | 1 | 1 | Прием в регистр |

вправо в освобождающийся старший разряд регистра; $Q_4$, $Q_3$, $Q_2$, $Q_1$ — выводы содержимого регистра; $v_1$, $v_2$, $v_3$ — выводы для выбора выполняемой регистром функции в соответствии с табл. 3.20. Предлагается самостоятельно разобрать выполнение функций в схеме рис. 3.58.

## 3.8. АНАЛОГО-ЦИФРОВЫЕ И ЦИФРОАНАЛОГОВЫЕ ПРЕОБРАЗОВАТЕЛИ ИНФОРМАЦИИ

### Принцип аналого-цифрового преобразования информации

В большинстве случаев получаемый непосредственно от источника информации сигнал представлен в форме непрерывно меняющегося по значению напряжения либо тока (рис. 3.59). Таков, в частности, характер электрического сигнала, соответствующего телефонным, телевизионным и другим видам сообщений. Для передачи таких сообщений по линии связи или для их обработки (например, при отфильтровании помех) могут быть использованы две формы: *аналоговая* или *цифровая*.

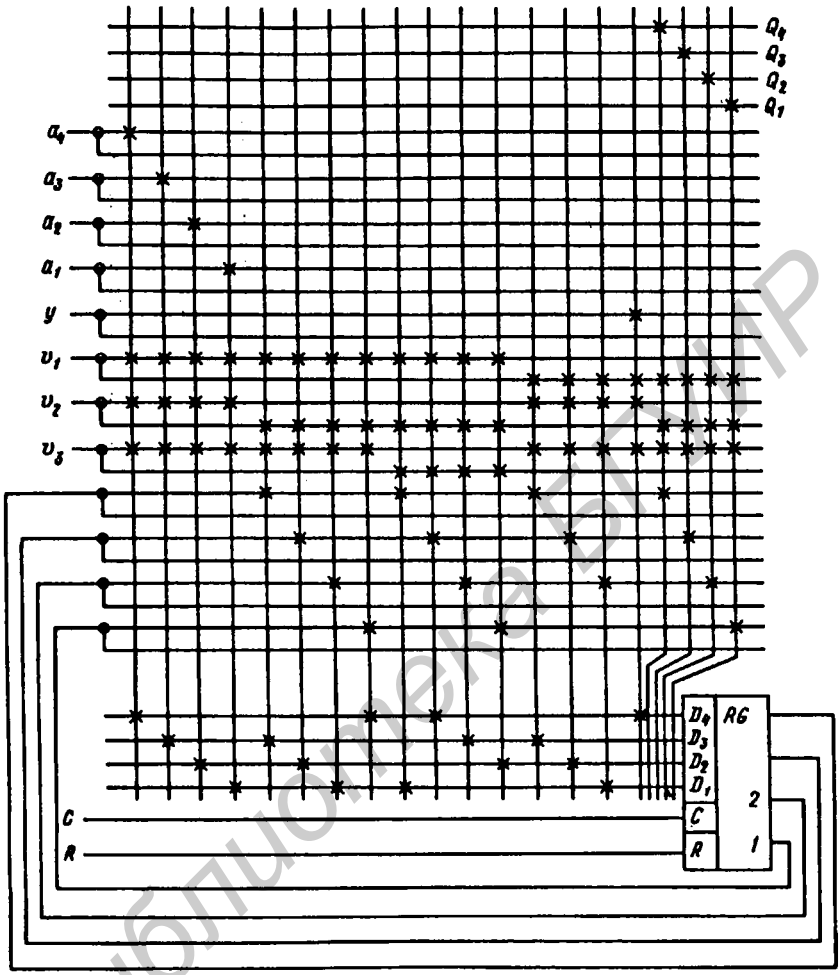



Рис. 3.58

Аналоговая форма предусматривает оперирование всеми значениями сигнала, цифровая форма — отдельными его значениями, представленными в форме кодовых комбинаций.

Преобразование сигналов из аналоговой формы в цифровую выполняется в устройстве, называемом *аналого-цифровым преобразователем* (АЦП). В преобразователе сигналов из аналоговой формы в цифровую можно выделить следующие процессы: *дискретизацию*, *квантование*, *кодирование*. Рассмотрим сущность этих процессов. При этом для определенности в последующем изложении будем считать, что преобразова-

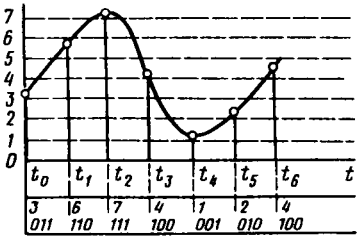



Рис. 3.59

ние в цифровую форму осуществляется над сигналом, представленным в форме меняющегося во времени напряжения.

**Дискретизация непрерывных сигналов.** Процесс дискретизации заключается в том, что из непрерывного во времени сигнала выбираются отдельные его значения, соответствующие моментам времени, следующим через определенный временной интервал $T$ (на рис. 3.59 моменты $t_0, t_1, ...$). Интервал $T$ называется *тактовым интервалом* времени, а моменты $t_0, t_1, ...$, в которые берутся отсчеты, — *тактовыми моментами* времени.

Дискретные значения сигнала следует отсчитывать с таким малым тактовым интервалом $T$, чтобы по ним можно было бы восстановить сигнал в аналоговой форме с требуемой точностью.

**Квантование и кодирование.** Сущность этих операций заключается в следующем. Создается сетка так называемых уровней квантования (рис. 3.59), сдвинутых друг относительно друга на величину $\Delta$, называемую *шагом квантования*. Каждому уровню квантования можно приписать порядковый номер (0, 1, 2, 3, 4 и т.д.). Далее полученные в результате дискретизации значения исходного аналогового напряжения заменяются ближайшими к ним уровнями квантования. Так, на диаграмме рис. 3.59 значение напряжения в момент $t_0$ заменяется ближайшим к нему уровнем квантования с номером 3, в тактовый момент $t_1$ значение напряжения ближе к уровню 6 и заменяется этим уровнем и т.д.

Описанный процесс носит название *операции квантования*, смысл которой состоит в округлении значений аналогового напряжения, выбранных в тактовые моменты времени. Как и всякое округление, процесс квантования приводит к погрешности (к *ошибкам квантования*) в представлении дискретных значений напряжения, создавая так называемый *шум квантования*. При проектировании АЦП стремятся снизить шум квантования до такого уровня, при котором он еще обеспечивает требуемую точность. Подробнее шум квантования будет рассмотрен далее.

Следующая операция, выполняемая при аналого-цифровом преобразовании сигналов, — *кодирование*. Смысл ее состоит в следующем. Округление значения напряжения, осуществляемое при операции квантования, позволяет эти значения представлять числами — номерами соответствующих уровней квантования. Для диаграммы, представленной на рис. 3.59, образуется последовательность чисел: 3, 6, 7, 4, 1, 2 и

т.д. Получаемая таким образом последовательность чисел представляется двоичным кодом.

Вернемся к искажениям, связанным с процессом квантования, названным шумом квантования. При телефонной связи шум квантования воспринимается ухом человека действительно в виде шума, сопровождающего речь.

Так как в процессе квантования значение напряжения в каждый тактовый момент округляется до ближайшего уровня квантования, ошибка в представлении значений напряжения оказывается в пределах $-\Delta/2 \le \varepsilon_{кв} \le +\Delta/2$. Следовательно, чем больше шаг квантования $\Delta$, тем больше ошибки квантования $\varepsilon_{кв}$. Считая, что в указанных пределах любые значения $\varepsilon_{кв}$ равновероятны, можно получить выражение среднеквадратичного значения ошибки квантования $\sigma_{\varepsilon} = \Delta/(3\sqrt{2})$.

Уменьшение шума квантования достигается только уменьшением шага квантования $\Delta$. Так как $\Delta$ — промежуток между соседними уровнями квантования, то с уменьшением $\Delta$, очевидно, должно возрасти число уровней квантования в заданном диапазоне значений напряжения. Пусть $A = U_{max} - U_{min}$ — ширина диапазона изменений напряжения. Тогда требуемое число уровней квантования $N = A/\Delta + 1$. Обычно $A/\Delta >> 1$ и $N \approx A/\Delta$. Отсюда видно, что уменьшение шума квантования путем уменьшения $\Delta$ приводит к увеличению числа уровней квантования $N$. Это увеличивает число разрядов при представлении номеров уровней квантования двоичными кодами. При организации телефонной связи номера уровней квантования обычно выражаются семи- восьмиразрядными двоичными числами, а число уровней квантования $N = 2^7 ... 2^8 = 128 ... 256$.

Наряду с рассмотренными выше погрешностями квантования при аналого-цифровом преобразовании возникают аппаратурные погрешности, связанные с неточностью работы отдельных узлов АЦП. Эти погрешности будут выявляться при рассмотрении различных схемных построений АЦП.

### Цифроаналоговые преобразователи

Рассмотрим цифроаналоговые преобразователи (ЦАП), построенные по принципу суммирования напряжений или токов, пропорциональных весовым коэффициентам двоичного кода.

**Схема ЦАП с суммированием напряжений.** Одна из таких схем с суммированием напряжений на операционном усилителе приведена на рис. 3.60. Триггеры 1 ... $n$ образуют регистр, в который помещаются двоичные числа, предназначенные для перевода в пропорциональные им значения напряжения на выходе. Будем считать, что напряжение на

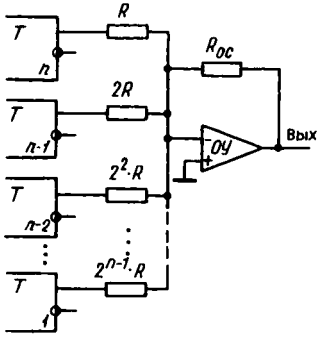



Рис. 3.60

выходе каждого из триггеров может принимать одно из двух возможных значений: $E$ при состоянии 1 и 0 при состоянии 0.

Напряжения с выходов триггеров передаются на выход ЦАП через операционный усилитель (ОУ), работающий в режиме взвешенного суммирования напряжений (аналогового сумматора). Для каждого триггера предусматривается отдельный вход в сумматоре с коэффициентом передачи

$$K_i = R_{\text{ос}} / R_{\text{вх}i} = R_{\text{ос}} / (R \cdot 2^{n-i}) =$$

$$= R_{\text{ос}} \cdot 2^{-n-i} / R.$$

Таким образом, напряжение с выхода триггера $n$-го разряда передается на выход усилителя с коэффициентом передачи: $K_n = R_{\text{ос}} / R$; этот коэффициент для $(n-1)$-го разряда $K_{n-1} = 2^{-1} R_{\text{ос}} / R$; для $(n-2)$-го разряда $K_{n-2} = 2^{-2} R_{\text{ос}} / R$ и т.д.

Обратим внимание на то, что коэффициенты передачи усилителя с отдельных его входов находятся в том же соотношении, что и весовые коэффициенты соответствующих разрядов двоичного числа. Так, $K_n$ в два раза больше $K_{n-1}$ и весовой коэффициент $n$-го разряда в два раза больше весового коэффициента $(n-1)$-го разряда. Следовательно, напряжения, передаваемые на выход усилителя с выходов триггеров отдельных разрядов, находящихся в состоянии 1, пропорциональны весовым коэффициентам разрядов.

Если в состоянии 1 находятся одновременно триггеры нескольких разрядов, то напряжение на выходе усилителя равно сумме напряжений, передаваемых на этот выход от отдельных разрядов двоичного числа в регистре: $a_n, a_{n-1}, \ldots, a_1$. Тогда напряжение на выходе усилителя

$$U_{\text{вых}} = E\frac{R_{\text{ос}}}{R}a_n + E\frac{R_{\text{ос}}}{R}2^{-1}a_{n-1} + \ldots + E\frac{R_{\text{ос}}}{R}2^{-(n-1)}a_1 =$$

$$= E\frac{R_{\text{ос}}}{R}2^{-(n-1)}\underbrace{(2^{n-1}a_n + 2^{n-2}a_{n-1} + \ldots + a_1)}_{N} = E\frac{R_{\text{ос}}}{R}2^{-(n-1)}N.$$

Здесь $N$ — десятичное значение двоичного числа, введенного в регистр. Из последнего выражения видно, что напряжение на выходе ЦАП пропорционально числу в регистре.

Рассмотрим работу ЦАП в случае, когда на триггерах 1 ... $n$ построен двоичный счетчик. Если подать на вход этого счетчика последователь-

ность импульсов, то с приходом каждого очередного импульса число в счетчике будет увеличиваться на единицу и напряжение на выходе ЦАП будет возрастать на ступеньку Δ, соответствующую единице младшего разряда счетчика: $\Delta = ER_{oc}2^{-(n-1)}/R$. Таким образом, напряжение на выходе ЦАП будет иметь ступенчатую форму, как показано на рис. 3.61. После поступления $2^n - 1$ импульсов все разряды счетчика будут содержать 1, на выходе ЦАП образуется максимальное напряжение

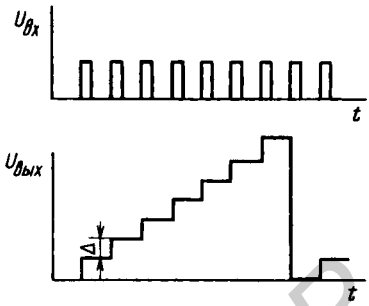



Рис. 3.61

$$U_{\text{вых max}} = (2^n - 1)\Delta = E\frac{R_{oc}}{R}2^{-(n-1)}(2^{n-1} - 1) = E\frac{R_{oc}}{R}(2 - 2^{-(n-1)}).$$

При большом числе разрядов $2 >> 2^{(n-1)}$ и $U_{\text{вых max}} \approx 2ER_{oc}/R$.

Далее очередным импульсом счетчик будет сброшен в нулевое состояние, нулевым будет и выходное напряжение ЦАП. После этого счетчик начинает счет импульсов сначала, и на выходе ЦАП вновь формируется напряжение ступенчатой формы.

Суммарная абсолютная погрешность преобразования $\Delta U_{\text{вых}}$ должна быть меньше выходного напряжения, соответствующего единице младшего разряда входного двоичного числа:

$$\Delta U_{\text{вых}} < E\frac{R_{oc}}{R}2^{-(n-1)}.$$

Отсюда можно получить условие для относительной погрешности:

$$\eta = \Delta U_{\text{вых}} / U_{\text{вых max}} < 2^{-(n-1)} / (2 - 2^{-(n-1)}) \approx 2^{-n}.$$

Это соотношение определяет связь между относительной погрешностью преобразователя η и числом его разрядов $n$. Так, при $n = 10$ имеем $\eta < 2^{-10} \approx 0{,}001 \approx 0{,}1\ \%$.

Недостатки рассмотренной схемы преобразователя:

используются высокоточные резисторы с различными значениями сопротивления;

трудно обеспечить высокую точность выходного напряжения триггеров.

Эти недостатки устранены в схеме ЦАП, приведенной на рис. 3.62, где показана схема трехразрядного преобразователя. Нетрудно построить схему с любым заданным числом разрядов. Особенности этой

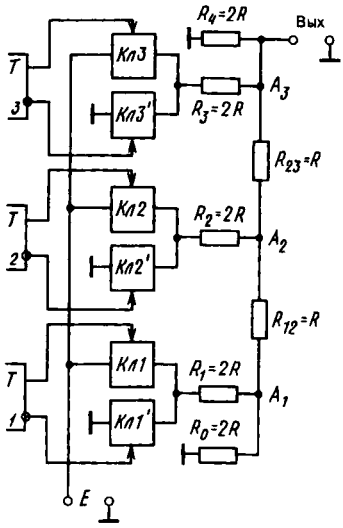



Рис. 3.62

схемы, называемой схемой с суммированием напряжений на резисторной матрице, состоит в том, что, во-первых, используются резисторы лишь с двумя значениями сопротивления ($R$ и $2R$) и, во-вторых, выходные напряжения триггеров непосредственно не участвуют в формировании выходного напряжения ЦАП, а используются лишь для управления состоянием ключей, т.е. устранены отмеченные выше недостатки предыдущей схемы ЦАП (см. рис. 3.60).

Рассмотрим подробнее работу такого преобразователя. В каждом разряде имеется два ключа, через один из них в резисторную матрицу подается напряжение $E$, через другой — нулевое напряжение. Определим напряжения на выходе ЦАП, соответствующие единицам разрядов числа, помещаемого в регистр. Пусть в регистр введено число $100_2$. Триггер 3 в состоянии 1, и в третьем разряде открыт ключ (Кл 3), в остальных разрядах триггеры в состоянии 0 и открыты ключи 2′ и 1′ (рис. 3.63,*а*). Последовательными преобразованиями можно получить схему (рис. 3.63,*д*), из которой следует, что напряжение в точке $A_3$ равно $U_{A_3} = U_{\text{вых}} = E/3$.

Если в регистр поместить число $010_2$, то резисторную матрицу можно представить схемой, показанной на рис. 3.64,*а*. Путем преобразования ее можно привести к схеме, представленной на рис. 3.64,*в*.

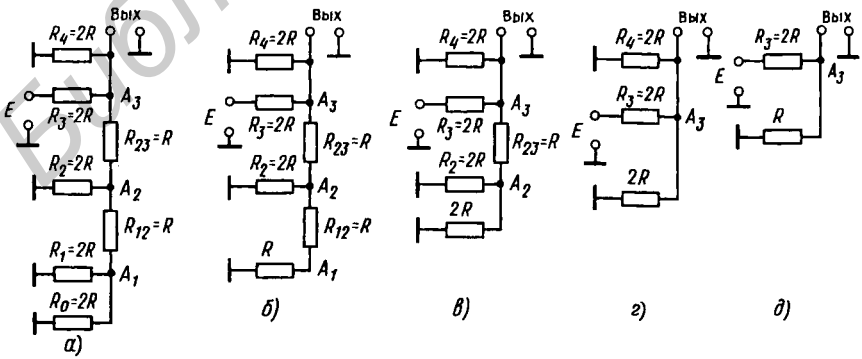



Рис. 3.63

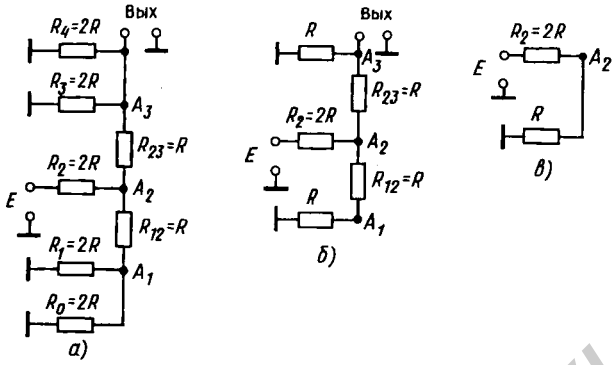



Рис. 3.64

Возникающее в точке $A_2$ напряжение имеет то же значение, что и в точке $A_3$ схемы на рис. 3.63. Из рис. 3.64,*б* видно, что при передаче на выход преобразователя это напряжение делится на два и, таким образом, $U_{вых} = 0{,}5\,U_{A_2} = 0{,}5\,E\,/\,3$.

Можно показать,что при числе $001_2$ напряжение в точке $A_1$ равно $U_{A_1} = E\,/\,3$. При передаче этого напряжения в точку $A_2$ и далее от точки $A_2$ к точке $A_3$ напряжение каждый раз делится на два и $U_{вых} = 0{,}25\,E\,/\,3$.

Итак, напряжение на выходе, соответствующее единицам отдельных разрядов двоичного числа в регистре, пропорционально весовым коэффициентам разрядов. При $n$-разрядном регистре, обозначив цифры разрядов двоичного числа $a_n, a_{n-1}, \ldots, a_1$, получим выражение напряжения на выходе ЦАП:

$$U_{вых} = \frac{1}{3}E(a_n + 2^{-1}a_{n-1} + 2^{-2}a_{n-2} + \ldots + 2^{-(n-1)}a_1) =$$

$$= \frac{1}{3}E2^{-(n-1)}\underbrace{(2^{(n-1)}a_n + 2^{(n-2)}a_{n-1} + 2^{(n-3)}a_{n-2} + \ldots + a_1)}_{N} = \frac{1}{3}E2^{-(n-1)}N.$$

Из выражения видно, что выходное напряжение ЦАП пропорционально числу $N$, помещаемому в регистр.

Аппаратурные погрешности преобразования в данной схеме связаны с отклонениями сопротивления резисторов от их номинальных значений, неидеальностью ключей (сопротивление реального ключа в закрытом состоянии не равно бесконечности, а в открытом — не равно нулю), нестабильностью источника напряжения $E$. Наибольшее влияние на погрешность ЦАП оказывают эти отклонения в старших разрядах.

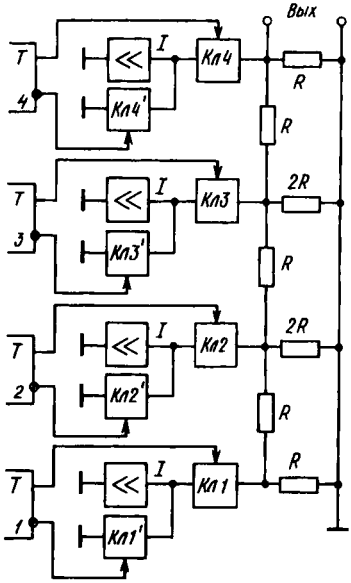



Рис. 3.65

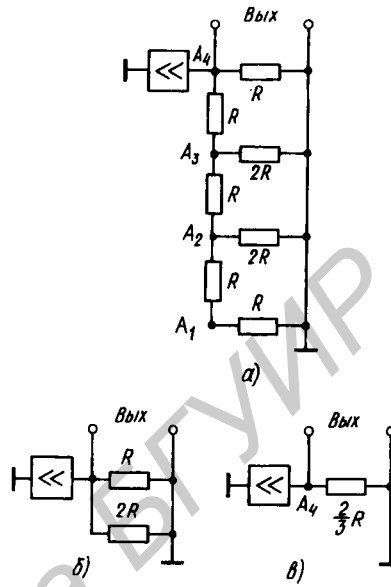



Рис. 3.66

**Схема ЦАП с суммированием токов**. На рис. 3.65 показан еще один вариант схемы ЦАП — схема с суммированием токов в резисторной матрице. Вместо источника стабильного напряжения $E$ в данной схеме используются источники стабильного тока $I$. Если триггер находится в состоянии 1, ток $I$ источника через открытый ключ втекает в резисторную матрицу, если триггер в состоянии 0, то открывается другой ключ, который замыкает источник. На рис. 3.66,*а* показана схема, соответствующая числу $1000_2$. Путем преобразования она приводится к эквивалентным схемам на рис. 3.66,*б* и *в*, откуда следует $U_{A_4} = U_{\text{вых}} = 2/3RI$.

Такое же напряжение образуется в любой из точек $A_1$, $A_2$, $A_3$, $A_4$, если соответствующий разряд регистра содержит 1. При передаче напряжения между этими точками напряжение делится на два и, следовательно, выходное напряжение

$$U_{\text{вых}} = \frac{2}{3} RI2^{-(n-1)}(2^{(n-1)}a_n + 2^{(n-2)}a_{n-1} + \ldots + a_1) = \frac{2}{3} RI2^{-(n-1)}N.$$

Рассмотрим схемные решения элементов, используемых в ЦАП.

**Источник стабильного напряжения.** На рис. 3.67 представлена схема простого стабилизатора напряжения. В цепь между входом и выходом стабилизатора последовательно включен транзистор VT1. Стабилизация выходного напряжения $U_{\text{ст}}$ обеспечивается тем, что при возраста-

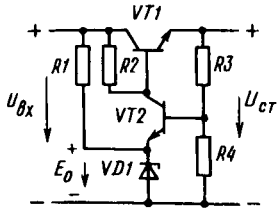



Рис. 3.67

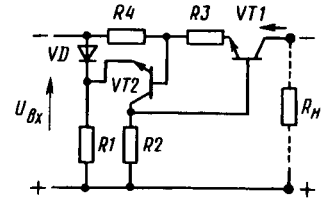



Рис.3.68

нии входного напряжения $U_{вх}$ увеличивается напряжение на транзисторе VT1 и, наоборот, при снижении $U_{вх}$ напряжение на транзисторе уменьшается. Таким образом, все изменения входного напряжения гасятся на транзисторе VT1. Такой режим транзистора VT1 обеспечивается усилителем, построенным на транзисторе VT2. Пусть, например, $U_{вх}$ растет и вследствие этого имеется тенденция к увеличению и $U_{ст}$. Малое увеличение $U_{ст}$, далее усиливаясь, уменьшает напряжение на коллекторе VT2 и базе VT1, увеличивается падение напряжения между коллектором и эмиттером транзистора VT1.

Цепочка из резистора R1 и стабитрона VD1 обеспечивает в цепи эмиттера VT2 постоянное напряжение $E_0$, которое стремится запереть транзистор. Для компенсации этого отрицательного смещения используется положительное напряжение, снимаемое с резистора R4 делителя напряжения, составленного из резисторов R3 и R4. Чем больше $E_0$, тем бóльшая часть напряжения $U_{ст}$ должна передаваться с R4 на базу VT2 и вместе с этим и бóльшая часть изменений напряжения $U_{ст}$ будет прикладываться к базе VT2 и, усиливаясь, передаваться на базу VT1.

**Источник стабильного тока.** Стабилизатор тока, схема которого приведена на рис. 3.68, работает аналогично стабилизатору напряжения. Отличие состоит в том, что входное напряжение усилителя на транзисторе VT2 снимается с резистора R4, который в схеме стабилизатора тока включен последовательно с нагрузкой (ток нагрузки $I$ проходит через $R_н$, VT1, R3, R4). Если, например, $U_{вх}$ возрастает или сопротивление $R_н$ уменьшается и, таким образом, ток имеет тенденцию к росту, возрастает напряжение на R4 и на базе транзистора VT2. Это приводит к снижению потенциала коллектора VT2 и базы VT1, растет напряжение между коллектором и базой транзистора VT1, что препятствует росту тока $I$.

**Ключевые устройства.** Ключи преобразователя с суммированием напряжений на резисторной матрице могут быть выполнены по схеме, представленной на рис. 3.69,*а*. Транзисторы VT1 и VT2 управляются напряжениями с выходов триггера. Выход схемы подключается к резисторной матрице.

Пусть триггер находится в состоянии 1. На его инверсном выходе низкий потенциал и транзистор VT2, на базу которого поступает этот

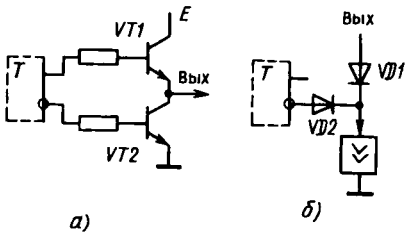



Рис. 3.69

потенциал, закрыт. На прямом выходе триггера напряжение высокого уровня. Оно поступает на вход транзистора VT1 и удерживает его в открытом состоянии. Через открытый транзистор VT1 в резисторную матрицу подается напряжение, близкое к $E$. Если триггер находится в состоянии 0, закрыт транзистор VT1, а через открытый транзистор VT2 в резисторную матрицу поступает напряжение низкого уровня. Таким образом, реализованное по данной схеме устройство выполняет роль двух ключей в разряде преобразователя.

В преобразователе с суммированием токов не обязательно стремиться к малому сопротивлению открытого ключа. В этом преобразователе, может быть использован диодный переключатель, схема которого представлена на рис. 3. 69,*б*. Если триггер находится в состоянии 0, высокое напряжение, поступающее с инверсного выхода триггера, удерживает диод VD2 в открытом состоянии. Ток источника замыкается через диод VD2 и триггер. Если триггер находится в состоянии 1, диод VD2 закрыт и ток $I$ замыкается через диод VD1 и резисторную матрицу.

### Аналого-цифровые преобразователи

По своей структуре схемы АЦП делятся на два типа: схемы, содержащие цифроаналоговый преобразователь (ЦАП), и схемы, не содержащие ЦАП.

**АЦП с промежуточным преобразованием напряжения во временно́й интервал.** Схема преобразователя данного типа приведена на рис. 3.70,*а*, временны́е диаграммы, иллюстрирующие процессы в преобразователе, — на рис. 3.70,*б*. В схеме этого типа ЦАП не используется.

Рассмотрим работу преобразователя. Очередным тактовым импульсом счетчик сбрасывается в нулевое состояние и одновременно запускается генератор линейно изменяющегося напряжения (ГЛИН). Выходное напряжение ГЛИН поступает на входы компараторов К1 и К2, на другие входы которых подаются соответственно нулевое напряжение и подлежащее преобразованию в числовую форму напряжение $U_{вх}$ на входе схемы (Вх). В момент времени, когда линейно изменяющееся напряжение, нарастая от небольших отрицательных значений, проходит нулевое значение, выдает импульс первый компаратор. Этим импульсом триггер устанавливается в состояние 1. В момент, когда линейно изменяющееся напряжение достигает значения $U_{вх}$, выдается

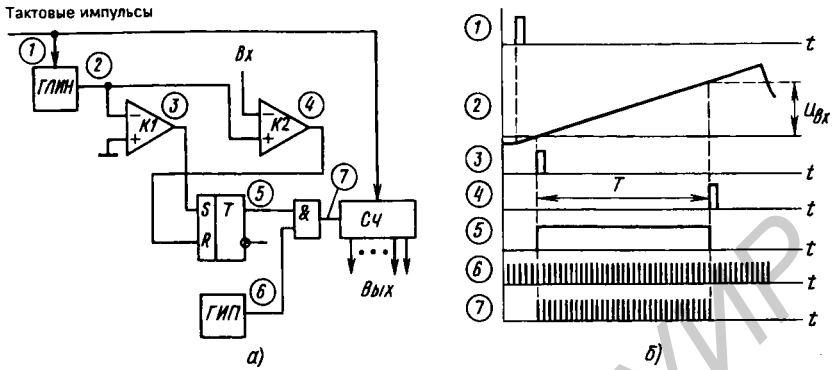



Рис. 3.70

импульс вторым компаратором. Этим импульсом триггер возвращается в состояние 0.

Время $T$, в течение которого триггер находится в состоянии 1, пропорционально входному напряжению. Таким образом, входное напряжение преобразуется во временной интервал, длительность которого пропорциональна значению входного напряжения.

В течение времени $T$ с выхода триггера подается высокое напряжение на вход элемента И, и импульсы генератора импульсной последовательности (ГИП) проходят через элемент на вход счетчика (Сч). Очевидно, устанавливающееся в счетчике число пропорционально $T$, а следовательно, и $U_{вх}$.

Для получения нового отсчета напряжения следует вновь подать импульс запуска. Таким образом, импульсы запуска должны следовать с частотой дискретизации входного напряжения. Покажем, как определяются параметры элементов преобразователя.

По заданной относительной погрешности $\eta$ преобразователя определяется максимальное число $N_{max}$, до которого счетчик должен производить счет: $N_{max} = 1/\eta$. *Число разрядов счетчика* находится как минимальное $n$, удовлетворяющее неравенству $N_{max} \le 2^n$.

Процесс преобразования значения $U_{вх}$ в число занимает время $T$, пропорциональное $U_{вх}$. Максимальное значение $T_{max}$ называется *временем преобразования*:

$$T_{пр} = T_{max} = \tau N_{max} = N_{max} / F,$$

где $\tau$ и $F$ — соответственно *период и частота генератора импульсов*.

Отсюда

$$F = N_{max} / T_{пр}.$$

При проектировании преобразователя время $T_{пр}$ бывает задано. Этот параметр определяет так называемую динамическую погрешность преобразователя, связанную с тем, что за время преобразования входное напряжение $U_{вх}$ может измениться. Изменение $U_{вх}$ за время $T_{пр}$ должно быть меньше напряжения, соответствующего единице младшего разряда счетчика.

*Крутизна напряжения ГЛИН* $\beta = U_{max} / T_{max} = U_{max} / T_{пр}$.

Аппаратурные погрешности преобразователя связаны с неточностью работы отдельных его элементов: нелинейностью напряжения ГЛИН; отклонениями момента времени, в который компаратором выдается импульс, от момента точного равенства входных напряжений компаратора; конечным временем срабатывания триггера, элемента И; нестабильностью частоты следования импульсов генератора.

**АЦП с двойным интегрированием.** Схема АЦП приведена на рис. 3.71,*а*. В ней, как и в схеме рассмотренного выше типа АЦП, не используется ЦАП, который для своего построения требует применения резисторной матрицы с высокоточными значениями сопротивлений.

Рассмотрим работу преобразователя. В момент $t_0$ (рис. 3.71,*б*) подачей импульса $U_п$ в цепь "Пуск" осуществляется запуск схемы: сбрасывается в 0 счетчик (Сч), первый ключ (Кл1) устанавливается в замкнутое состояние, второй ключ (Кл2) — в разомкнутое.

Предварительно разряженный конденсатор С начинает заряжаться током от источника входного напряжения $U_{вх}$. Так как входное напряжение операционного усилителя (ОУ) близко к нулю, практически все напряжение $U_{вх}$ падает на резисторе R1 и ток в цепи резистора $I_{зар} = U_{вх}/R_1$. Этот ток замыкается через конденсатор С. Если за время длительности импульса $U_п (T_п = t_1 - t_0)$ значение напряжения $U_{вх}$ считать неизменным, конденсатор будет заряжаться постоянным током

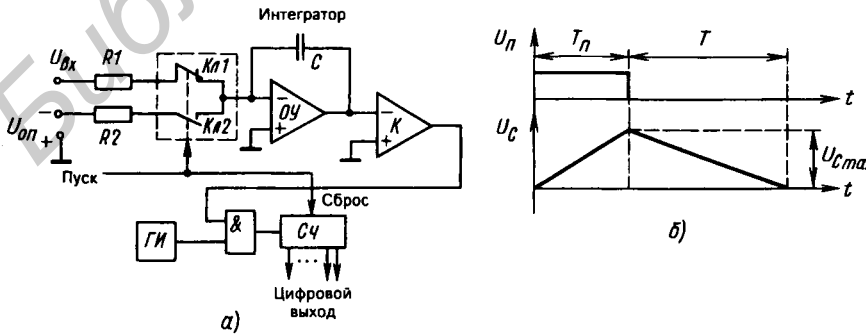



Рис. 3.71

и напряжение на нем будет изменятъся по линейному закону, достигая к моменту $t_2$ значения

$$U_{C\max} = I_{зар}\frac{T_п}{C} = \frac{U_{вх}}{R_1}\frac{T_п}{C}.$$

В момент окончания импульса на входе "Пуск" (в момент $t_1$) счетчик начинает счет импульсов, поступающих в него из генератора импульсной последовательности (ГИ) через элемент И. В этот же момент ключ Кл1 устанавливается в разомкнутое состояние, ключ Кл2 — в замкнутое. В цепи конденсатара возникает ток обратного направления $I_{раз} = U_{оп}/R_2$. Конденсатор разряжается постоянным током $I_{раз}$, и напряжение на нем снижается по линейному закону. В момент $t_2$ напряжение на конденсаторе $U_C$ и напряжение на выходе операционного усилителя $U_{ОУ} = -U_C$ проходят нулевое значение, на выходе компаратора (К) устанавливается уровень *лог.0*, прекращается прохождение импульсов ГИ через элемент И на вход счетчика (Сч). Образующееся к этому моменту в Сч число $N$ есть значение $U_{вх}$, представленное в цифровой форме.

Определим значение $N$. Время разряда конденсатора

$$T = \frac{CU_{C\max}}{I_{раз}} = \frac{U_{C\max}C}{U_{оп}}R_2.$$

Подставляя выражение $U_{Cmax}$, получаем

$$T = \frac{U_{вх}}{R_1}\frac{T_п}{U_{оп}}R_2.$$

Если период следования импульсов ГИ равен τ, то количество импульсов $N$, поступающих в счетчик за время $T$, определится выражением

$$N = \frac{T}{\tau} = U_{вх}\left(\frac{1}{U_{оп}}\frac{T_п}{\tau}\frac{R_2}{R_1}\right) = U_{вх}k.$$

Как видим, $N$ пропорционально $U_{вх}$. Величина

$$k = \frac{1}{U_{оп}}\frac{T_п}{\tau}\frac{R_2}{R_1}$$

определяет масштаб, в котором представляется значение $U_{вх}$.

**Аналого-цифровой преобразователь последовательного счета.** Структурная схема преобразователя данного типа приведена на рис. 3.72,*а*. Тактовым импульсом (ТИ) счетчик (Сч) сбрасывается в нулевое состояние. Нулевое напряжение $U_{ЦАП} = 0$ возникает на выходе ЦАП, преобразующего числа в счетчике в пропорциональное напряжение. Устанавливается неравенство $U_{вх} > U_{ЦАП}$, при котором компаратор (К) подает на вход элемента И уровень *лог.1*. При этом импульсы генератора

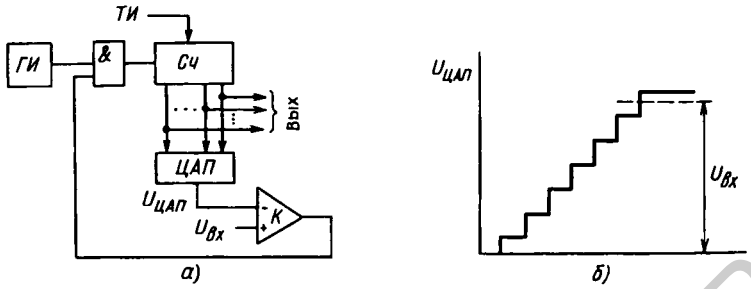



Рис. 3.72

импульсной последовательности (ГИ) проходят через элемент И на вход счетчика. Каждый поступивший на вход счетчика импульс вызывает увеличение на единицу хранившегося в нем числа, на одну элементарную ступеньку возрастает напряжение на выходе ЦАП. Таким образом, напряжение $U_{ЦАП}$ растет по ступенчатому закону, как показано на рис. 3.72,*б*. В момент времени, когда $U_{ЦАП}$ достигает значения, превышающего $U_{вх}$, компаратор выдает уровень *лог.0*, и в дальнейшем прекращается доступ импульсов генератора в счетчик. Полученное к этому моменту времени в счетчике число пропорционально напряжению $U_{вх}$.

Из-за того, что в АЦП рассматриваемого типа не используется генератор линейно изменяющегося напряжения, его аппаратурные погрешности меньше, чем могут быть в АЦП с промежуточным преобразованием напряжения во временной интервал.

**Аналого-цифровой преобразователь следящего типа.** Рассмотренные выше типы АЦП работают в циклическом режиме. В них каждый очередной тактовый импульс устанавливает преобразователь в исходное состояние, после чего начинается процесс преобразования. Быстродействие таких преобразователей ограничивается главным образом быстродействием счетчика (а именно быстродействием триггеров его младших разрядов, в которых переключение происходит с высокой частотой).

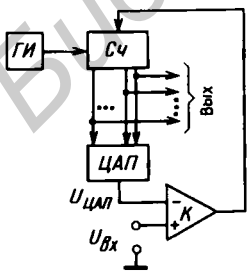



Рис. 3.73

На практике часто используется нециклический преобразователь, структурная схема которого представлена на рис. 3.73. Эта схема отличается от предыдущей тем, что в ней используется реверсивный счетчик (Сч), управляемый сигналами с выхода компаратора (К). При $U_{вх} > U_{ЦАП}$ счетчик устанавливается в режим прямого счета, поступающие на вход им-

пульсы генератора последовательно увеличивают в нем число, растет $U_{ЦАП}$, пока не достигнет значения $U_{вх}$.

При $U_{вх} < U_{ЦАП}$ счетчик переводится в режим обратного счета, при котором убывает число в счетчике и, следовательно, убывает напряжение $U_{ЦАП}$, пока не будет достигнуто значение $U_{вх}$. Таким образом, все происходящие во времени изменения напряжения $U_{вх}$ отслеживаются напряжением $U_{ЦАП}$ на выходе ЦАП.

В необходимые моменты времени с выхода счетчика могут сниматься числа, пропорциональные значениям $U_{вх}$.

**Аналого-цифровой преобразователь последовательного приближения**. Структурная схема преобразователя приведена на рис. 3.74. В схеме предусмотрен построенный на RS-триггерах 1 ... $n$ регистр числа. В этом регистре формируется число, пропорциональное напряжению $U_{вх}$.

Вначале записывается единица только в триггер $n$ старшего разряда этого регистра. Получающееся в регистре число с помощью ЦАП преобразуется в напряжение $U_{ЦАП}$, которое сравнивается с напряжением $U_{вх}$. Если выполняется неравенство $U_{вх} \geq U_{ЦАП}$, то число, в которое преобразуется $U_{вх}$, действительно содержит единицу в старшем разряде. При невыполнении неравенства триггер $n$ сбрасывается в нуль. Далее производится запись единицы в триггер $n - 1$ следующего разряда регистра и вновь сравнением напряжения $U_{вх}$ с $U_{ЦАП}$, соответствующим имеющемуся к этому моменту времени числу в регистре, выясняется, должна ли быть сохранена единица в данном разряде или триггер этого

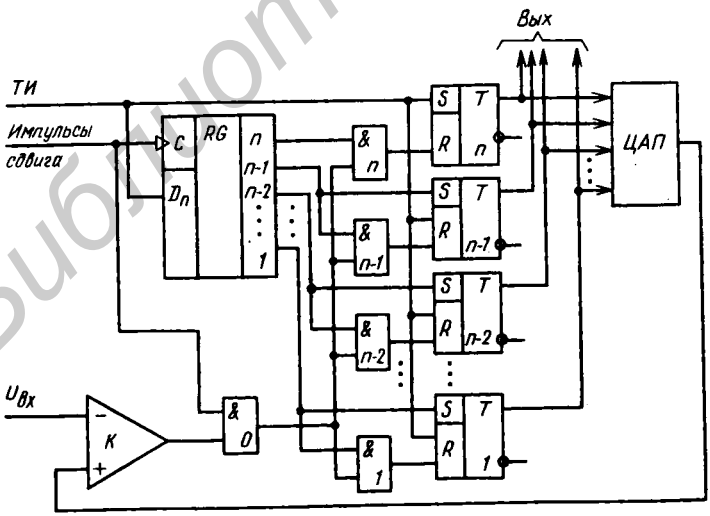



Рис. 3.74

разряда должен быть возвращен в состояние 0. Аналогичные операции выполняются во всех разрядах, после чего получающееся в регистре число может быть выдано на выход.

Рассмотрим выполнение указанных действий в преобразователе, схема которого представлена на рис. 3.74. Тактовый импульс устанавливает триггер $n$ в состояние 1, остальные триггеры 1 ... $n-1$ — в состояние 0. Этим же импульсом одновременно производится запись единицы в старший разряд сдвигового регистра RG, и на $n$-м выходе регистра появляется уровень *лог.1*.

Компаратор сравнивает $U_{вх}$ с $U_{ЦАП}$, соответствующим имеющемуся к этому моменту числу в регистре числа, и при выполнении условия $U_{вх} < U_{ЦАП}$ выдает уровень *лог.1*. При поступлении импульса сдвига логический уровень с выхода компаратора через элемент И0 передается на вход элемента И$n$, и если этот уровень был уровнем *лог.1*, то триггер n возвращается в состояние 0. В момент окончания импульса сдвига завершается процесс сдвига на один разряд вправо содержимого регистра, появляется уровень *лог.1* на $(n-1)$-м выходе этого регистра, триггер n устанавливается в состояние 1. Далее с приходом очередного импульса сдвига определяется требуемое состояние триггера $n-1$ и в момент окончания импульса триггер $n-2$ устанавливается в состояние 1. Эти действия повторяются до тех пор, пока не будет определено состояние всех триггеров.

### Схема выборки и хранения

В тех случаях, когда аналоговый сигнал на входе АЦП изменяется с большой скоростью, за время преобразования может произойти существенное изменение входного напряжения. Получаемое при этом на выходе АЦП числовое значение не будет соответствовать значению входного сигнала в тактовый момент времени. Устранение этого явления достигается использованием так называемой схемы выборки и хранения. Эта схема производит из входного напряжения выборку значения, соответствующего тактовому моменту времени, и хранит эту выборку неизменной в течение времени, необходимого для ее преобразования в числовую форму.

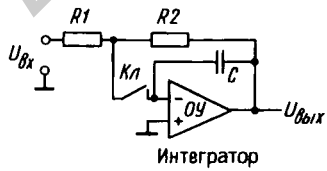



Рис. 3.75

На рис. 3.75 приведена упрощенная схема выборки и хранения. В исходном состоянии ключ (Кл) замкнут. При малой постоянной времени $R_1C$ напряжение на конденсаторе (С) следует за изменениями напряжения $U_{вх}$ с требуемой точностью. В тактовый момент времени ключ переводится в разомкнутое состояние. Напряжение на конденсаторе $U_C$, имевшееся к моменту размыка-

ния ключа (представляющее собой выборку из напряжения $U_{вх}$), может сохраняться практически неизменным в течение достаточно длительного времени. Напряжение с выхода операционного усилителя $U_{ОУ} = -U_C$ поступает в АЦП и преобразуется в числовую форму. После окончания преобразования ключ вновь замыкается.

## 3.9. ПОЛУПРОВОДНИКОВЫЕ ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

### Классификация и параметры 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 битов.

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

Оперативное ЗУ используется 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 цифровых устройствах ОЗУ используются для хранения данных (исходных данных, промежуточных и конечных результатов обработки данных) и программ.

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 в таких специализированных цифровых устройствах, которые, функционируя длительное время, многократно выполняют действия по одному и тому же алгоритму при различных исходных данных.

Перепрограммируемое ПЗУ в процессе функционирования цифрового 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 цифрового устройства, про-

изводится 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.

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 кратной 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 $M = 64 \cdot 1024 \cdot 8$ 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 мощностью, 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).

### Оперативное запоминающее устройство

На рис. 3.76 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 любом из трех режимов: режиме хранения, в котором он отключается от входа и выхода микросхемы, режиме чтения, в котором содержащаяся в ЭП информация выдается на выход микросхемы, режи-

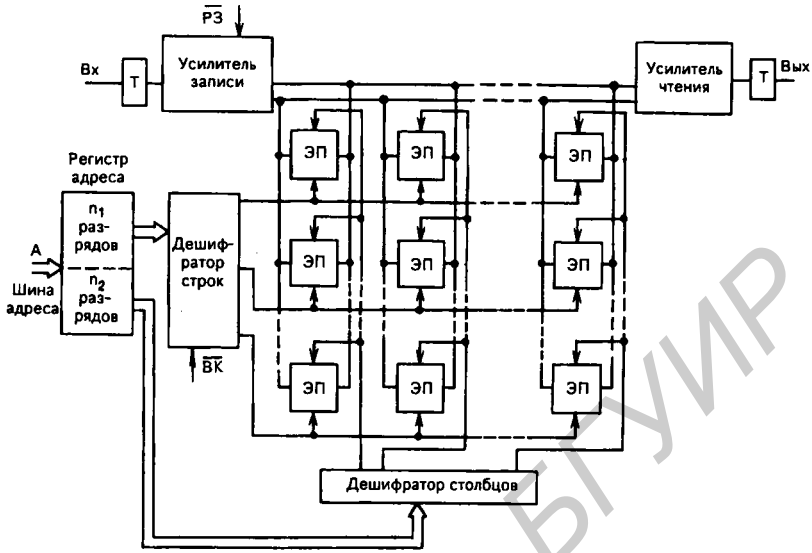



Рис. 3.76

ме записи, в котором в ЭП записывается новая поступающая со входа микросхемы информация.

Каждому ЭП приписан номер, называемый *адресом* элемента. Для поиска требуемого ЭП указывается 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 число столбцов накопителя выбираются равными целой степени двух. И если число строк $N_{стр} = 2^{n_1}$ и число столбцов $N_{ст} = 2^{n_2}$, то общее число ЭП (емкость накопителя)

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

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

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 $2^{n_1} = 2^{n_2} = 32$.

Разряды регистра адреса делятся на две группы: одна группа из $n_1$ разрядов определяет двоичный номер строки, в которой в накопителе расположен ЭП, другая группа из $n_2$ разрядов — двоичный номер столбца, в котором расположен выбираемый ЭП. Каждая группа разрядов

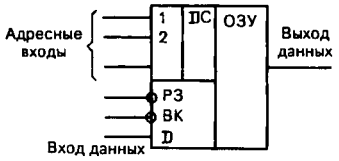



Рис.3.77

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 (Т) и выход микросхемы. Режим записи устанавливается подачей сигнала на вход разрешения записи ($\overline{РЗ}$). При уровне *лог.0* на входе $\overline{РЗ}$ 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 ($\overline{ВК}$) 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. 3.77 показано условное графическое обозначение микросхемы ОЗУ.

Рассмотрим последовательность подачи сигналов в режимах чтения и записи. На рис. 3.78,*а* представлена временна́я диаграмма сигналов в режиме чтения. С определенной задержкой $t_{з1}$ относительно момента подачи адреса и сигнала в цепь $\overline{ВК}$ (связанной с процессами дешифрации адреса и включения выходных цепей выбранного ЭП) на выход микросхемы передается содержимое выбранного ЭП. В режиме записи (рис. 3.78,*б*) должны быть соблюдены условия, которые исключали бы нарушение содержимого ячеек, в которые не производится обращение. Это обеспечивается тем, что сигнал в цепь $\overline{РЗ}$ подается с задержкой

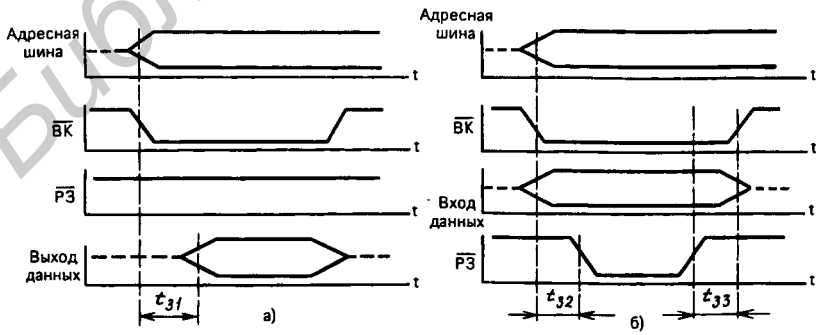



Рис. 3.78

$t_{32}$ относительно момента подачи сигналов в цепь адреса, $\overline{ВК}$ и входных данных и снимается сигнал в цепи $\overline{РЗ}$ прежде, чем будет снят сигнал в цепи $\overline{ВК}$. В противном случае, при преждевременной подаче сигнала $\overline{РЗ}$, может произойти запись в ячейку с адресом, не совпадающим с адресом на входах микросхемы.

Микросхемы ОЗУ допускают наращивание емкости памяти *наращиванием разрядности* (и, следовательно, разрядности хранимых слов) и *наращиванием числа ячеек* (и, значит, числа слов, которые можно хранить в памяти). Таким образом, используя соответствующее число микросхем в определенном их соединении, можно строить память с требуемой организацией.

Рассмотрим схему наращивания разрядности ячеек (рис. 3.79). 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 $n$) потребуется $n$ микросхем.

На рис. 3.80 показана схема наращивания числа ячеек и их разрядности. Блок памяти состоит из микросхем, образующих отдельные *линейки (ряды)*, каждая из которых строится по схеме наращивания разрядности (рис. 3.79). 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.

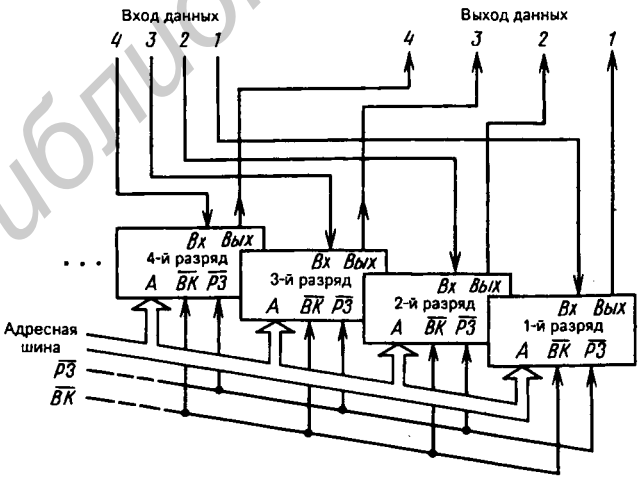



Рис. 3.79

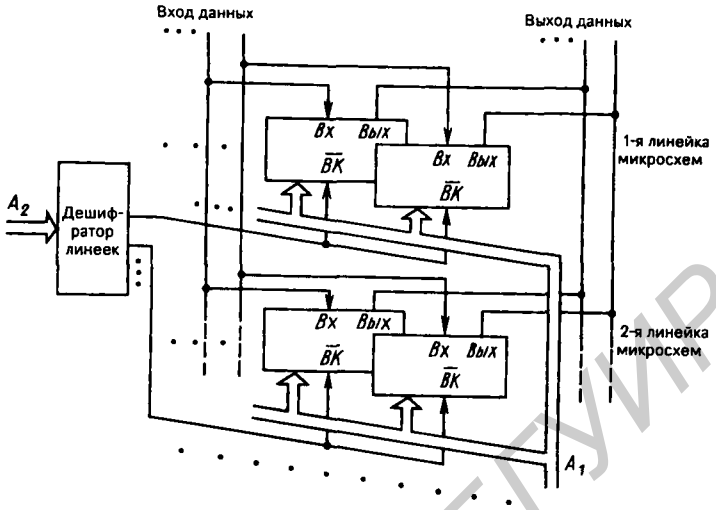



Рис.3.80

Выбор линейки осуществляется с помощью дешифратора, на вход которого подается $A_2$, а каждый из выходов подключен к входу $\overline{\text{ВК}}$ определенной линейки. Таким образом, в зависимости от кодовой комбинации, содержащейся в $A_2$, на соответствующем выходе дешифратора появляется уровень *лог.0*, который обеспечивает 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 наращивания 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. 3.80 использовать восемь микросхем; для увеличения числа ячеек с 1024 до 4096 (в четыре раза) необходимо предусмотреть четыре 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 номбинаций 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.

В каждом столбце матрицы микросхем на рис. 3.80 выходы всех микросхем объединяются в цепь соответствующего разряда выхода данных блока, все входы данных — в цепь соответствующего разряда входа данных блока памяти.

### Постоянные запоминающие устройства

Как и ОЗУ, ПЗУ состоит из ячеек, обратившись к которым можно вывести их содержимое. Отличие от ОЗУ заключается в том, что информация в ячейки 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 первые информация заносится в процессе изготовления микросхем 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. 3.81 приведена структура ПЗУ, программируемого пользователем. Как и в ОЗУ, матрица-накопитель состоит из элементов памяти, 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.

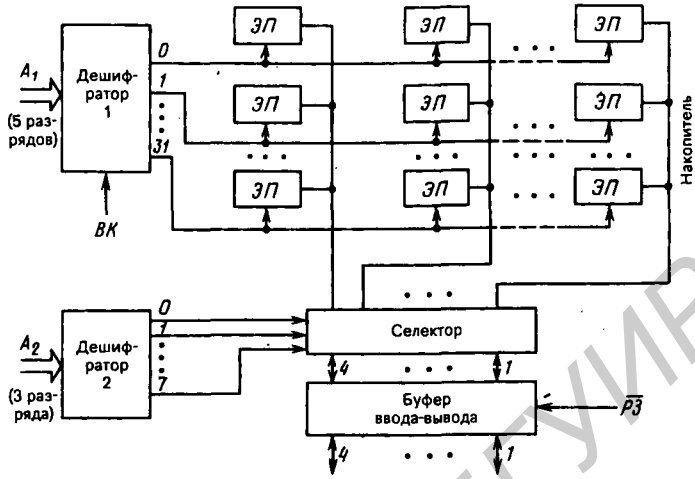



Рис. 3.81

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 восемь разрядов, разбивающихся на две группы разрядов $A_2$ и $A_1$: пятиразрядную группу $A_1$ и трехразрядную группу $A_2$. Группа $A_1$ подается на дешифратор 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 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.

### Перепрограммируемые постоянные запоминающие устройства

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

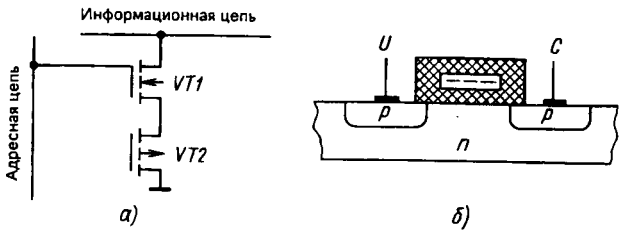



Рис. 3.82

Рассмотрим принцип работы приведенного на рис. 3.82,*а* элемента памяти с электрической записью информации и стиранием ультрафиолетовым светом.

Транзистор VT1 служит для выборки элемента памяти. Хранение информации осуществляется в транзисторе VT2. Особенность транзистора VT2, структура которого показана на рис. 3.82,*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 области проводящей *p*-канал между истоком и стоком.

Транзистор оказывается в состоянии 0. Если же к *p*—*n*-переходу не прикладывалось повышенного напряжения, заряд на затворе отсутствует, транзистор оказывается в непроводящем состоянии (состояние 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, снимается заряд с затворов транзисторов, и все транзисторы накопителя оказываются установленными в непроводящее состояние. Обычное комнатное освещение практически не оказывает влияния на состояние транзисторов.

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

## 3.10. КОНТРОЛЬ ЦИФРОВЫХ УСТРОЙСТВ

### Общие сведения

В процессе работы цифрового устройства иногда возникают ошибки, искажающие информацию. Причинами таких ошибок могут быть:

выход из строя какого-либо элемента, из-за чего устройство теряет работоспособность;

воздействие различного рода помех, возникающих из-за проникновения сигналов из одних цепей в другие через различные паразитные связи.

Выход из строя элемента устройства рассматривается как *неисправность*. При этом в устройстве наблюдается постоянное искажение информации.

Иной характер искажений информации имеет место под воздействием помех. Вызвав ошибку, помехи могут затем в течение длительного времени не проявлять себя. Такие ошибки называют *случайными сбоями.*

В связи с возникновением ошибок необходимо снабжать цифровые устройства системой контроля правильности циркулирующей в ней информации. Такие системы контроля могут предназначаться для решения задач двух типов: задачи *обнаружения* и задачи *исправления ошибок*. Система обнаружения ошибок, производя контроль информации, способна лишь выносить решения: нет ошибок и есть ошибка, причем в последнем случае она не указывает, какие разряды слов искажены. Система исправления ошибок сигнализирует о наличии ошибок и указывает, какие из разрядов искажены. При этом непосредственное исправление цифр искаженных разрядов представляет собой уже несложную операцию. Так, если известно, что некоторый разряд двоичного слова ошибочен, то появление в нем ошибочного *лог.0* означает, что правильное значение — *лог.1* и наоборот.

Таким образом, трудно локализовать ошибку, т.е. указать, в каких разрядах слова она возникла. После решения этой задачи само исправление сводится лишь к инверсии цифр искаженных разрядов, поэтому обычно под исправлением ошибок понимают решение задачи локализации ошибок.

При постоянном нарушении правильности информации, обнаружив ошибку, можно принять меры для поиска неисправного элемента и заменить его исправным. Причины же случайных сбоев обычно выявляются чрезвычайно трудно, и такие изредка возникающие ошибки желательно было бы устранять автоматически, восстанавливая правильное значение слов с помощью системы исправления ошибок. Однако следует иметь в виду, что система исправления ошибок требует значительно большего количества оборудования, чем система обнаружения ошибок.

Ниже раздельно рассматриваются методы контроля цифровых устройств двух типов: устройств хранения и передачи информации, уст-

ройств обработки информации. К устройствам первого типа могут быть отнесены запоминающие устройства, регистры, цепи передачи и другие устройства, в которых информация не должна изменяться. На выходе этих устройств информация та же, что и на входе. К устройствам второго типа относятся устройства, у которых входная информация не совпадает с выходной и в тех случаях, когда ошибки не возникают. Примером могут служить арифметические и логические устройства.

### Обнаружение одиночных ошибок в устройствах хранения и передачи информации

Для дальнейшего изложения потребуется понятие *кодовое расстояние по Хеммингу*. Для двух двоичных слов кодовое расстояние по Хеммингу есть число разрядов, в которых разнятся эти слова. Так, для слов $11011_2$ и $10110_2$ кодовое расстояние $d = 3$, так как эти слова различаются в трех разрядах (первом, третьем и четвертом).

Пусть используемые слова имеют $m$ разрядов. Для представления информации можно использовать все $2^m$ возможных комбинаций от 00 ... 0 до 11 ... 1. Тогда для каждого слова найдутся другие такие слова, которые отличаются от данного не более чем в одном разряде. Например, для некоторого слова 1101 можно найти следующие слова: 0101, отличающиеся только в четвертом разряде; 1001, отличающееся только в третьем разряде, и т.д. Таким образом, *минимальное кодовое расстояние* $d_{\min} = 1$. Обнаружить ошибки в таких словах невозможно. Например, если передавалось слово $N_1 = 1101$, а принято $N_2 = 0101$, то в принятом слове невозможно обнаружить никаких признаков наличия ошибки (ведь могло бы быть передано и слово $N_2 = 0101$). Для того чтобы можно было обнаружить одиночные ошибки (ошибки, возникающие не более чем в одном из разрядов слова), минимальное кодовое расстояние должно удовлетворять условию $d_{\min} \geq 2$. Это условие требует, чтобы любая пара используемых слов отличалась друг от друга не менее чем в двух разрядах. При этом, если возникает ошибка, она образует такую комбинацию цифр, которая не используется для представления слов, т.е. образует так называемую *запрещенную* комбинацию.

Для получения $d_{\min} = 2$ достаточно к словам, использующим любые комбинации из $m$ информационных двоичных разрядов, добавить один дополнительный разряд, называемый *контрольным*. При этом значение цифры контрольного разряда будем выбирать таким, чтобы общее число единиц в слове было четным. Например:

| Информационные разряды | Контрольные разряды |
|---|---|
| 11001110111 | 0 |
| 11010100111 | 1 |

В первом из приведенных примеров число единиц в информационной части четно (8), поэтому контрольный разряд должен содержать 0. Во втором примере число единиц в информационной части слова нечетно (7), и для того, чтобы общее число единиц в слове было четным, контрольный разряд должен содержать единицу. Таким способом во все слова вводится определенный признак — четность числа единиц. Принятые слова проверяются на наличие в них этого признака, и, если он оказывается нарушенным (т.е. обнаруживается, что число содержащихся в разрядах слова единиц нечетно), принимается решение, что слово содержит ошибку.

Этот метод позволяет обнаруживать ошибку. Но с его помощью нельзя определить, в каком разряде слова содержится ошибка, т.е. нельзя исправить ее. Кроме того, при этом методе не могут обнаруживаться ошибки четной кратности, т.е. ошибки одновременно в двух, четырех и т.д. разрядах, так как при таком четном числе ошибок не нарушается четность числа единиц в разрядах слова. Однако наряду с одиночными ошибками могут обнаруживаться ошибки, возникающие одновременно в любом нечетном числе разрядов.

На практике часто вместо *признака четности* используется *признак нечетности*, т.е. цифра контрольного разряда выбирается такой, чтобы общее число единиц в разрядах слова было нечетным. При этом, если имеет место, например, обрыв линии связи, это обнаруживается, так как принимаемые слова будут иметь 0 во всех разрядах и нарушится принцип нечетности числа единиц.

Рассмотрим схемы, выполняющие операцию проверки на четность (нечетность). Проверка на четность требует суммирования по модулю 2 цифр разрядов слова. Если $a_1$, $a_2$, ... , $a_n$ — цифры разрядов, результат проверки на четность определится выражением $p = a_1 \oplus a_2 \oplus a_3 \oplus ... \oplus a_n$. Если $p = 0$, то число единиц в разрядах слова четно, в противном случае оно нечетно.

Наиболее просто эта операция реализуется, когда контролируемое слово передается в последовательной форме. Суммирование в этом случае может быть выполнено в последовательности $p = ...((a_1 \oplus a_2) \oplus a_3) \oplus ... \oplus a_n$. К результату суммирования $p_i$ первых разрядов прибавляется цифра очередного поступающего разряда $(i + 1)$, находится результат суммирования $(i + 1)$ разрядов $p_{i+1} = p_i \oplus a_{i+1}$, и так до тех пор, пока не будут просуммированы цифры всех разрядов.

*Таблица 3.21*

| $p_i$ | $p_{i+1}$ | $p_{i+1} = p_i \oplus a_{i+1}$ | $p_i$ | $p_{i+1}$ | $p_{i+1} = p_i \oplus a_{i+1}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |

Из таблицы истинности для операции $p_{i+1} = p_i \oplus a_{i+1}$ (табл. 3.21) видно, что *лог.0* не должен менять состояния устройства суммирования ($p_{i+1} = p_i$), *лог.1* переводит устройство в новое состояние ($p_{i+1} = \bar{p}_i$). Эта логика соответствует работе триггера со счетным входом (рис. 3.83). Действительно, пусть триггер был предварительно установлен в состояние 0, после чего на его синхронизирующий вход стали поступать логические уровни, соответствующие цифрам контролируемого слова. При этом первая *лог.1* переведет триггер в состояние 1, вторая *лог.1* вернет триггер в состояние 0 и т.д. Следовательно, после подачи четного числа единиц триггер окажется в состоянии $p = 0$; при поступлении нечетного числа единиц — в состоянии $p = 1$.

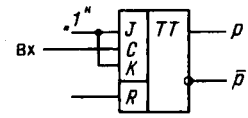



Рис. 3.83

Если разряды контролируемого слова передаются в параллельной форме, то последовательность действий при проверке на четность может быть следующая:

$$p = \ldots (((a_1 \oplus a_2) \oplus (a_3 \oplus a_4)) \oplus ((a_5 \oplus a_6) \oplus (a_7 \oplus a_8)))\oplus \ldots$$

Согласно этому выражению для нахождения $p$ вначале попарно суммируются по модулю 2 цифры разрядов контролируемого слова, далее полученные результаты также суммируются попарно и т.д. Этот принцип вычисления $p$ использован в схеме проверки на четность на рис. 3.84,*а*. Цифры разрядов (и их инверсии) поступают на входы элементов (на рис. обозначены =1) первого яруса схемы, в которых они попарно суммируются по модулю 2. Полученные результаты попарно суммируются в элементах второго яруса и т.д. Результат проверки на четность образуется на выходе элемента старшего яруса. Каждый из элементов схемы реализует следующую логическую функцию:

$$y = x_1 \oplus x_2 = x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2 = \overline{\overline{x_1 \cdot \bar{x}_2 \vee \bar{x}_1 \cdot x_2}} = (x_1 \mid \bar{x}_2) \mid (\bar{x}_1 \mid x_2).$$

Построенная по данному выражению схема элемента, выполняющего операцию суммирования по модулю 2, приведена на рис.3.84,*б*.

Определим число ярусов и число элементов в этой схеме проверки на четность. Пусть число разрядов $n$ контролируемого слова составляет целую степень двух. Число элементов в отдельных ярусах $a_i$ составляет геометрическую прогрессию 1, 2, 4, 8, ... , $n/2$, знаменатель которой $q = 2$. Для последнего $k$-го яруса $a_k = 1$, для первого яруса $a_1 = a_k q^{k-1} = n/2$, откуда $2^{k-1} = n/2$ или $2^k = n$.

Из этого соотношения можно найти число ярусов $k$. Число суммирующих элементов в схеме равно сумме членов приведенной выше геометрической прогрессии:

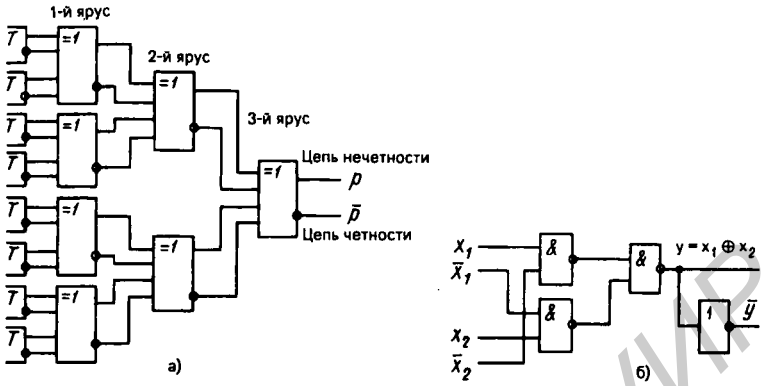



Рис.3.84

$$N_{эл} = \frac{a_k(q^{k-1} - 1)}{q - 1} = \frac{1 \cdot (2^k - 1)}{2 - 1} = 2^k - 1 = n - 1.$$

**Контроль арифметических операций**

В устройствах хранения и передачи информации одиночная ошибка вызывала искажение цифры лишь одного разряда слова. При выполнении арифметических операций одиночная ошибка в получаемом результате может вызвать искажение одновременно группы разрядов. Действительно, пусть в суммирующем счетчике хранится число $N_1 = 10111_2$ и на вход поступает очередная единица. Произойдет сложение: $N_2 = N_1 + 1$. При этом

| | |
|---|---|
| Переносы | 1 1 1 |
| $N_1$ | 1 0 1 1 1 |
| + | 1 |
| $N_2$ | 1 1 0 0 0 |

Пусть в процессе суммирования из-за ошибочной работы устройства не будет передан перенос из 2-го разряда в 3-й. Такая одиночная ошибка приведет к следующему результату:

| | |
|---|---|
| Переносы | 1 |
| $N_1$ | 1 0 1 1 1 |
| + | 1 |
| $N_{2ош}$ | 1 0 1 0 0 |

Сравнивая ошибочный результат $N_{2ош}$ с правильным $N_2$, видим, что они различаются в двух разрядах. Тем не менее считаем, что в $N_{2ош}$ содержится одиночная ошибка.

Во всех случаях, когда ошибочный результат связан с арифметическим прибавлением (или вычитанием) ошибочной единицы к одному из разрядов, имеет место одиночная ошибка. И если ошибочный результат может быть получен из правильного результата путем арифметического суммирования (или вычитания) единицы не менее чем в $k$ разрядах, кратность ошибки равна $k$.

Для контроля арифметических операций чаще всего используется *контроль по модулю q*. Этот метод более универсален и годится также для контроля устройств хранения и передачи информации. Сущность метода состоит в следующем.

Контролируемое число $N$ арифметически делится на $q$, и выделяется остаток $r_N$. Остаток вписывается в контрольные разряды числа $N$ вслед за его информационными разрядами. Принятое число $N^*$ делится на $q$ и выделяется остаток $r^*_N$. Эту операцию выполняет устройство свертки по модулю $q$ (рис. 3.85,*а*). Элемент сравнения сравнивает $r_N$ и $r^*_N$, в случае их несовпадения выносит решение о наличии ошибки в принятом слове. Схема на рис. 3.85,*б* иллюстрирует принцип контроля суммирующего устройства. Пусть в результате суммирования чисел $N_1$ и $N_2$ получено $N^*$. Остатки $r_{N1}$ и $r_{N2}$ также суммируются с выделением остатка $r_N$. Если остаток $r^*_N$, полученный от деления числа $N^*$ на модуль $q$, не совпадает с $r_N$, то элемент сравнения сигнализирует о наличии ошибок в работе устройства.

Чаще всего используется $q = 3$, иногда выбирается $q = 7$. При увеличении значения $q$ возрастает способность метода к обнаружению ошибок, но одновременно увеличивается объем контролирующего оборудования.

Рассмотрим пример применительно к схеме на рис. 3.85,*б*. Пусть $q = 3$, $N_1 = 32_{10} = 100000_2$, $N_2 = 29_{10} = 011101_2$. Соответствующие этим числам

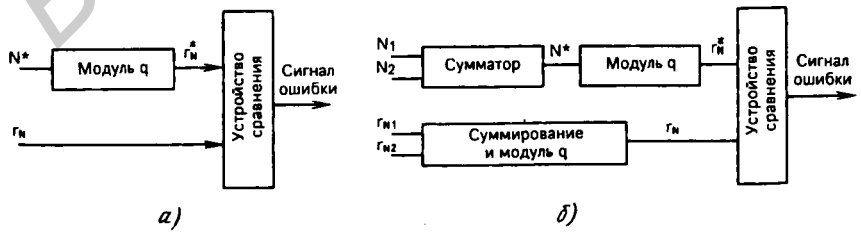



Рис. 3.85

остатки равны $r_{N1} = 2_{10} = 10_2$, $r_{N2} = 2_{10} = 10_2$ (при $q = 3$ остатки могут принимать значения 0, 1, 2 и для их представления в двоичной форме достаточно двух контрольных разрядов). При отсутствии ошибок в работе устройства результат суммирования чисел $N^* = N_1 + N_2 = 61_{10} = 111101_2$, значение свертки по модулю 3 равно $r_N^* = 01_2$. Суммируя $r_{N1}$ и $r_{N2}$ и выделяя остаток по модулю 3, получаем $r_N = 01_2$. Совпадение $r_N^* = r_N$ указывает на отсутствие ошибок. При наличии ошибок не имело бы места совпадение остатков $r_N^*$ и $r_N$.

Эффективность контроля по модулю характеризуется данными, приведенными в табл. 3.22. В таблице указано, какую часть всех возможных

*Таблица 3.22*

| Значение модуля | Доля необнаруживаемых ошибок | | |
|---|---|---|---|
| | 1-й кратности | 2-й кратности | 3-й кратности |
| 3 | 0 | 1/2 | 1/4 |
| 7 | 0 | 1/6 | 1/7 |

комбинаций ошибок составляют ошибки, которые не обнаруживаются при контроле по модулю. Как видно из приведенных данных, обнаруживаются все однократные ошибки; доля ошибок высокой кратности, оказывающихся необнаруженными, при модуле 7 меньше, чем при модуле 3. Тем самым эффективность контроля по модулю 7 выше, чем при модуле 3. Однако при контроле по модулю 7 контрольная часть слов содержит три двоичных разряда (вместо двух разрядов при модуле 3) и, кроме того, сложнее схемы формирования остатков (схемы свертки).

В заключение рассмотрим построение схем свертки по модулю 3. Общим для этих схем является следущий метод получения остатка. Каждый разряд числа вносит определенный вклад в формируемый остаток. В табл. 3.23 приведены остатки от деления на 3 значений, выражаемых единицами отдельных разрядов (т.е. весовых коэффициентов разрядов). Эти остатки для единиц нечетных разрядов равны 1, для четных разрядов они равны 2. Следовательно, для получения остатка от деления на 3 всего числа достаточно просуммировать остатки для единиц отдельных его разрядов и затем для получения суммы найти остаток от деления на 3.

Например, пусть $N = 11001011_2$; сумма остатков, создаваемых отдельными разрядами, $S = 1 \cdot 2 + 1 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 1 \cdot 2 + 0 \cdot 1 + 1 \cdot 2 + 1 \cdot 1 = 8$; далее, деля 8 на 3, получаем остаток $r_N = 2$.

*Таблица 3.23*

| Номер разряда | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |
|---|---|---|---|---|---|---|---|---|---|
| Весовой коэффициент разряда | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | ... |
| Остаток от деления на 3 значения, выражаемого единицей разряда | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | ... |

**Схема свертки по модулю 3 для последовательной формы передачи чисел.** Схема может быть выполнена в виде двухразрядного счетчика с циклом 3, построенного таким образом, что единицы нечетных разрядов поступающего на вход числа вызывают увеличение содержимого счетчика на единицу, а единицы четных разрядов вызывают увеличение числа в счетчике на два. Функционирование такого счетчика описывается табл. 3.24.

*Таблица 3.24*

| Четность номера в разряде $b$ | Текущее состояние счетчика | | Следующее состояние счетчика | |
|---|---|---|---|---|
| | $a_2$ | $a_1$ | $a_2^*$ | $a_1^*$ |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |

Здесь $b$ — код, определяющий четность номера очередного разряда числа, поступающего на вход счетчика; примем для четных разрядов $b = 0$, для нечетных разрядов $b = 1$.

По этой таблице и таблице переходов JK-триггера (табл. 3.25) по-

*Таблица 3.25*

| Вид перехода триггера | Сигналы на входах | | Вид перехода триггера | Сигналы на входах | |
|---|---|---|---|---|---|
| | $J$ | $K$ | | $J$ | $K$ |
| $0 \to 0$ | 0 | — | $1 \to 0$ | — | 1 |
| $0 \to 1$ | 1 | — | $1 \to 1$ | — | 0 |

строены приведенные на рис. 3.86,*а* карты, по которым находятся логические выражения для входов триггеров ТТ1 и ТТ2 счетчика:

$$J_1 = a_2 \cdot \bar{b} \vee \bar{a}_2 \cdot b, \quad K_1 = 1;$$

$$J_2 = \bar{a}_1 \cdot \bar{b} \vee a_1 \cdot b, \quad K_2 = 1.$$

Представив выражения для $J_1$ и $J_2$ в базисе И-НЕ: $J_1 = (a_2 \mid \bar{b}) \mid (\bar{a}_2 \mid b)$, $J_2 = (a_1 \mid \bar{b}) \mid (a_1 \mid b)$, получим схему межтриггерных связей на рис. 3.86,*б*. Логическая переменная $b$ формируется триггером *3*.

Этот триггер переключается тактовыми импульсами (ТИ), следующими с частотой поступления разрядов числа на вход счетчика. Таким образом, в моменты поступления нечетных разрядов триггер *3* устанавливается в состояние 1 и $b = 1$, в моменты поступления четных разрядов $b = 0$.

**Схема свертки для параллельной формы представления числа.** При параллельной форме представления числа обычно используется пирамидальный способ построения схемы свертки, показанный на рис. 3.87.

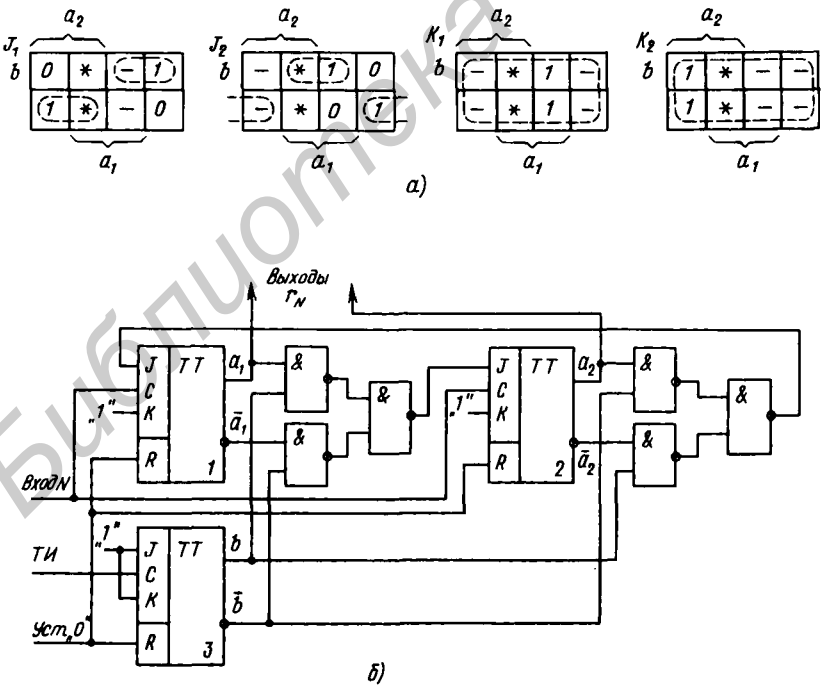



Рис. 3.86

Элементы А первого яруса формируют остатки для пар разрядов числа, выдавая уровень *лог.1* на один из выходов $A_0, A_1, A_2$ в зависимости от значения остатка (0, 1, 2). В последующих ярусах используются однотипные элементы В, которые формируют остатки по результатам, выдаваемым парой элементов предыдущего яруса. На выходе элемента последнего яруса образуется остаток для всего числа.

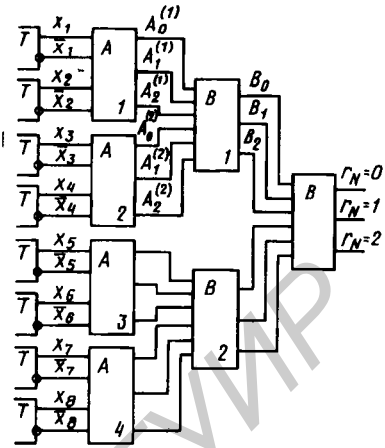



Рис.3.87

Выходы элементов определяются следующими логическими выражениями:
для первого яруса

$$A_0 = x_1 \cdot x_2 \vee \bar{x}_1 \cdot \bar{x}_2, \quad A_1 = x_1 \cdot \bar{x}_2,$$

$$A_2 = \bar{x}_1 \cdot x_2;$$

для остальных ярусов

$$B_0 = A_0' \cdot A_0'' \vee A_1' \cdot A_2'' \vee A_2' \cdot A_1'', \; B_1 = A_0' \cdot A_1'' \vee A_1' \cdot A_0'' \vee A_2' \cdot A_2'',$$

$$B_2 = A_0' \cdot A_2'' \vee A_0'' \cdot A_2' \vee A_1' \cdot A_1''.$$

Если число разрядов $n = 2^k$, то число ярусов в схеме свертки равно $k$, а число элементов составляет $n - 1$.

# Глава 4. Процессор

## 4.1. ПРИНЦИП РАБОТЫ ЭВМ

### Поколения ЭВМ

Первая ЭВМ была создана в 1946 г. В последующий период до 1955 г. происходило становление вычислительной техники. В это время определились основные принципы построения ЭВМ. Затем с периодичностью 5 — 7 лет происходил переход к ЭВМ принципиально новых типов, использующих более совершенную элементную базу, имеющих новую структуру, расширяющую их возможности и обеспечивающую бóльшие удобства при работе с ними человека. В связи с этим появилось понятие *поколение* ЭВМ.

Для *ЭВМ первого поколения* (40-е — начало 50-х годов) характерны следующие признаки. Строились они на дискретных компонентах с использованием электровакуумных приборов, имели низкую надежность, в них применялись ЗУ на ультразвуковых линиях задержки и электронно-лучевых трубках. Ориентировались машины в основном на решение научно-технических задач, для которых характерны относительно небольшие объемы исходных данных и результатов решения.

В *ЭВМ второго поколения* (середина 50-х — 60-е годы) в качестве элементной базы применялись дискретные компоненты и полупроводниковые приборы (транзисторы и диоды), монтаж осуществлялся с использованием печатных плат, ЗУ выполнялись на тороидальных ферритовых сердечниках. Все это повысило быстродействие и надежность машин. В ЭВМ второго поколения обеспечивалась возможность обмена данными между ЭВМ и большим числом внешних устройств. ЭВМ стали успешно применяться и для решения экономических задач.

В *ЭВМ третьего поколения* (60-е годы) в качестве элементной базы используются интегральные микросхемы. Благодаря этому ЭВМ третьего поколения по сравнению с ЭВМ второго поколения имеют меньшие габаритные размеры и потребляемую мощность, бóльшие быстродейст-

вие и надежность, широко применяются в самых разнообразных областях деятельности человека.

В *ЭВМ четвертого поколения* (70-е годы) в качестве элементной базы используются интегральные микросхемы высокой степени интеграции — *большие интегральные схемы* (БИС). С их помощью на одном кристалле можно создать устройства, содержащие тысячи и десятки тысяч транзисторов. Компактность узлов при использовании БИС позволяет строить ЭВМ с большим числом вычислительных устройств — процессоров (так называемые многопроцессорные вычислительные системы).

В последнее время создаются *ЭВМ и вычислительные системы пятого поколения*. Эти ЭВМ будут обладать высокой производительностью, компактностью и низкой стоимостью (эти характеристики улучшаются в каждом следующем поколении ЭВМ). Основная особенность ЭВМ пятого поколения будет состоять в их высокой интеллектуальности, обеспечивающей возможность общения человека с ЭВМ на естественном языке, способности ЭВМ к обучению и т.д. Быстродействие ЭВМ пятого поколения будет достигать десятков миллиардов операций в секунду, они будут обладать памятью в десятки мегабайтов и строиться на *сверхбольших* БИС, на кристалле которых будут размещаться миллионы транзисторов.

### Структура ЭВМ

ЭВМ состоит из ряда устройств, взаимодействующих друг с другом в процессе решения задачи. Рассмотрим кратко основные устройства и их функции (рис. 4.1).

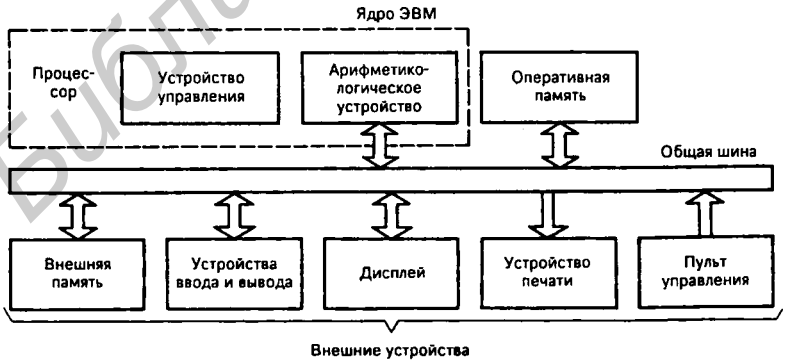



Рис. 4.1

*Оперативная память* (ОП) служит для хранения программы, исходных данных задачи, промежуточных и конечных результатов решения задачи.

*Арифметико-логическое устройство* (АЛУ) предназначено для выполнения предусмотренных в ЭВМ арифметических и логических операций. Участвующие в операциях данные выбираются из ОП, результаты операций отсылаются в ОП. Для ускорения выборки операндов (данных, участвующих а операциях) АЛУ может снабжаться собственной местной памятью (сверхоперативным запоминающим устройством — СОЗУ) на небольшое число данных, но обладающей быстродействием, превышающим быстродействие ОП. При этом результаты операций, если они участвуют в последующих операциях, могут не отсылаться в ОП, а храниться в СОЗУ.

*Устройство управления* (УУ), посылая в определенной временно́й последовательности управляющие сигналы в устройства ЭВМ, обеспечивает их соответствующее функционирование и взаимодействие друг с другом.

АЛУ и УУ объединяют под общим названием *процессор*. Процессор совместно с оперативной памятью образует *ядро ЭВМ*.

Кроме этих узлов ЭВМ снабжается рядом других устройств, называемых *внешними*.Они обеспечивают расширение возможностей ЭВМ, облегчают пользование ими. В состав внешних устройств могут входить следующие узлы.

*Внешняя память* (ВП) — память, имеющая относительно невысокое быстродействие, но по сравненив с ОП существенно более высокую емкость. В силу того что быстродействие внешней памяти значительно ниже быстродействия АЛУ, последнее в процессе работы взаимодействует лишь с ОП, получая из нее команды и данные, отсылая в эту память результаты операций. Часто при решении сложных задач емкость ОП оказывается недостаточной. В этих случаях в процессе решения задач данные определенными порциями могут пересылаться из внешней памяти в ОП, откуда они затем выбираются для обработки в АЛУ.

*Дисплей* позволяет отображать на экране разнообразную информацию, связанную с процессом решения задачи.

*Устройство печати* обеспечивает возможность выдачи разнообразной документации.

С помощью *устройств ввода* и *вывода* ЭВМ может обмениваться данными, передаваемыми по линиям связи.

Через *пульт управления* человек вводит в ЭВМ данные и команды выполнения разнообразных действий.

**Команды**

В процессоре предусматривается возможность выполнения большого числа различных операций. Несмотря на то что число таких операций может быть более 100, каждая из них представляет собой простейшие арифметические либо логические действия, такие, например, как сложение, вычитание, умножение и деление чисел, пересылка кодов и т.п. При этом в каждой операции участвует не более двух операндов. В связи с этим решаемая задача должна быть предварительно представлена последовательностью таких операций, которые способна выполнять ЭВМ. Затем на каждую из этих операций должна быть составлена так называемая *команда*. Совокупность команд, образующая *программу* решения задачи, должна быть помещена в ОЗУ.

Команда должна содержать все необходимые для выполнения операции указания: вид операции, место, где хранятся операнды данной операции и куда должен быть помещен результат операции. Такая команда имеет следующий формат:

| КОп | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|

Здесь КОп — код операции. В каждой ЭВМ предусматривается определенная система кодирования операций. Например, может быть принят следующий способ записи вида операции: 01 — сложение, 02 — вычитание, 03 — умножение и т.д.; $A_1$ — первый адрес — адрес ячейки оперативной памяти, в которой хранится первый операнд; А2 — второй адрес — адрес второго операнда; $A_3$ — третий адрес — адрес ячейки оперативной памяти, в которую должен помещаться результат операции. Команда с таким содержанием называется трехадресной.

ЭВМ могут использовать двухадресные команды, имеющие следующий формат:

| КОп | $A_1$ | $A_2$ |
|---|---|---|

Результат операции в этом случее помещается в ячейку одного из аперандов либо остается в АЛУ.

Одноадресные команды имеют формат

| КОп | $A_1$ |
|---|---|

При таком формате для выполнения одного арифметического действия над двумя числами от машины может потребоваться исполнение нескольких команд. Например, для сложения двух чисел необходимо выполнить три команды:

ввести в АЛУ число, хранящееся в оперативной памяти (ОЗУ) по приведенному в команде адресу;

прибавить к принятому числу число, хранящееся в памяти по указанному в команде адресу;

поместить полученный в АЛУ результат в память по адресу, указанному в данной команде.

Широкое распростренение нолучили машины с переменной адресностью. В них при выполнении операций операнды (один либо оба) могут выбираться не из оперативной памяти, а из местной (СОЗУ). Команды этих машин по существу являются двухадресными, но оба адреса либо один из них могут быть адресами не оперативной памяти, а регистров местной памяти АЛУ.

Рассмотрим взаимодействие устройств ЭВМ в процессе решения задачи. Для определенности примем, что рассматриваемая ЭВМ является трехадресной. Перед решением задачи набор команд, образующий программу решения, помещается в последовательные ячейки оперативной памяти так, что адрес ячейки, содержащей следующую команду, на единицу больше адреса ячейки, в которую помещена предыдущая команда.

Процесс реализации программы состоит в последовательной выборке из ОП команд и их исполнении. Вызванная из ОЗУ в центральное устройство управления (ЦУУ) очередная команда хранится в нем все время исполнения операции. ЦУУ выбирает из команды первый адрес $A_1$, пересылает его в ОЗУ и подает сигнал считывания. Из ОЗУ выдается первый операнд. ЦУУ подает в АЛУ сигнал отпирания входов регистра, в который должен быть принят этот операнд. Аналогично по второму адресу $A_2$ производится передача из ОЗУ в АЛУ второго операнда. Затем ЦУУ подает в АЛУ управляющие сигналы, под действием которых выполняется предусмотренная командой операция. После получения результата операции ЦУУ передает в ОЗУ третий адрес $A_3$, подает сигнал записи и открывает выход регистра АЛУ, хранящего результат операции.

Далее в ОЗУ передается адрес очередной команды, сформированной в ЦУУ (например, путем увеличения на единицу адреса предыдущей команды), в ЦУУ поступает следующая команда и т.д.

## 4.2. ОБЩИЕ ВОПРОСЫ ПОСТРОЕНИЯ ПРОЦЕССОРА

### Аналоговые и цифровые методы обработки информации

Обработка информации может осуществляться двумя методами: *аналоговым*, при котором участвующие в обработке величины представляются в аналоговой форме (обычно уровнями напряжения или тока), или *цифровым*, при котором величины представляются в цифровой форме и обработка сводится к последовательности действий (операций) над числами. В зависимости от используемого метода обработки различают

два типа аппаратуры: *аналоговая*, в которой используется аналоговый метод обработки, и *цифровая*, в которой применяется цифровой метод обработки. В цифровой аппаратуре основным устройством, в котором непосредственно выполняется обработка, является процессор.

Рассмотрим характерные особенности указанных двух методов обработки информации.

В аналоговой аппаратуре обработка информации реализуется путем преобразований между токами и напряжениями вида

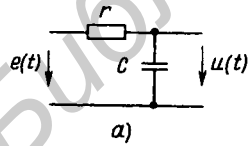
$$u_L = L\frac{di_L}{dt},\quad i_C = C\frac{du_C}{dt},\quad u_r = i_r \cdot r,$$

выполняемых соответственно индуктивными, емкостными, резистивными элементами, а также элементами усиления $u_2 = ku_1$ и нелинейных преобразований $u_2 = f(u_1)$. Каждый элемент аналогового устройства в каждый момент времени находится в режиме активного выполнения характерных для этих элементов операций; таким образом, выполнение операций носит параллельный характер (одновременно выполняются операции во многих элементах устройства).

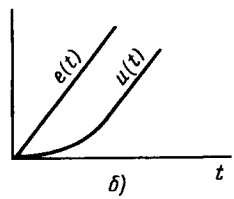
В показанном на рис. 4.2,*а* простейшем аналоговом устройстве на выходе формируется величина $u(t)$, являющаяся решением дифференциального уравнения (рис.4.2,*б*):

$$\frac{du(t)}{dt} + \frac{1}{rC}u(t) = \frac{1}{rC}e(t). \qquad (4.1)$$

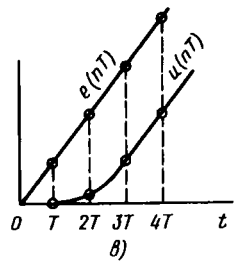
В процессоре обработка информации осуществляется последовательным выполнением простейших арифметических и логических опе-



*а)*



*б)*



*в)*

Рис. 4.2

раций над числами (например, сложение пары чисел, сдвиг числа влево, вправо).

Рассмотрим решение дифференциального уравнения (4.1) цифровым методом. Решение ищется для дискретных моментов времени, следующих через определенные временны́е интервалы, называемые шагом интегрирования $T$ (рис. 4.2,*в*). Пусть для этих моментов времени искомые значения ..., $u(nT)$, $u(nT + T)$, $u(nT + 2T)$, ... Представим в (4.1) значение производной в дискретный момент $nT$ приближенным выражением

$$\left.\frac{du(t)}{dt}\right|_{t=nT} \approx \frac{u(nT+T)-u(nT)}{T}. \qquad (4.2)$$

Подставляя (4.2) в (4.1), после простых преобразований получаем разностную форму дифференциального уравнения:

$$u(nT + T) = k_1 u(nT) + k_2 e(nT), \qquad (4.3)$$

где $k_1 = 1 - T/(rC)$, $k_2 = T/(rC)$.

Выражением (4.3) пользуются как рекуррентным: по известным $u(nT)$, $e(nT)$ вычисляется значение $u(nT + T)$; затем для вычисления $u(nT + 2T)$ используется то же выражение (4.3), в котором во всех членах увеличивается значение $n$ на единицу подстановкой $n + 1$ вместо $n$. При этом (4.3) преобразуется к виду

$$u(nT + 2T) = k_1 u(nT + T) + k_2 e(nT + T). \qquad (4.4)$$

Подставляя в правую часть этого выражения ранее вычисленное значение $u(nT + T)$ и значение $e(t)$ для момента $nT + T$, находят значение $u(nT + 2T)$. Далее в (4.4) вновь увеличивают на единицу значение $n$ и т.д.

На рис. 4.3 представлен алгоритм решения рассматриваемой задачи. Вычисление отклика цепи для дискретных моментов времени сводится к многократному выполнению действий в соответствии с (4.3). Такие вычислительные процессы, предусматривающие многократное использование одной и той же группы операций, называют *циклическими*. Схема на рис. 4.3 описывает циклический алгоритм. В ней при каждом исполнении блоков 1 — 7 вычисляется очередное значение выходной величины. Выполнение операций этой группы блоков повторяется столько раз, каково число рассчитываемых значений выходной величины $u(t)$.

Для хранения значений встречающихся в алгоритме величин используются следующие регистры процессора: в регистре R0 хранятся значения $u(nT)$, подставляемые в правую часть (4.3); в регистре R1 формируется результат вычисления правой части выражения (4.3), т.е. $u(nT + T)$; значения коэффициентов $k_1$ и $k_2$ хранятся соответственно в регистрах R2 и R3; регистр R4 используется в качестве счетчика, в него предварительно помещается требуемое число вычисляемых значений отклика $u(T)$.

Рассмотрим операции, выполняемые в блоках. В блоке 1 производится прием очередного значения $e$ в регистр R1. В блоке 2 вычисляется произведение $k_2 \cdot e$, для чего перемножаются содержимое регистров R3 и R1 (чтобы показать, что имеется в виду содержимое регистра, его обозначение здесь и далее заключают в круглые скобки), результат выполненной операции помещается в тот же регистр R1, из которого выбирается один из сомножителей.

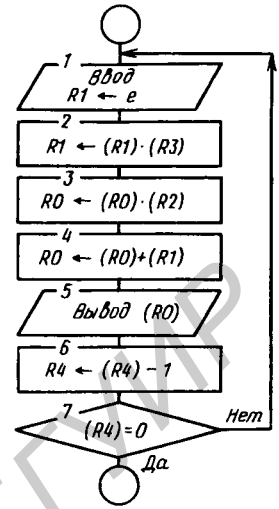



Рис. 4.3

Блок 3 формирует в регистре R0 произведение $k_1 u(nT)$. В блоке 4 производится суммирование членов правой части выражения (4.3) с получением в регистре R0 значения отклика для очередного момента времени. Блок 5 производит вывод значения отклика. В блоке 6 из содержимого регистра R4 (счетчика) вычитается единица и затем в блоке 7 производится проверка на равенство нулю содержимого счетчика. Если условие (R4) = 0 выполняется, то происходят выход из цикла, при невыполнении этого условия происходит переход к блоку 1 и очередное повторение цикла.

Таким образом, в пределах шага интегрирования $T$ (значение которого для обеспечения необходимой точности выбирается достаточно малым) выполняется большое число арифметических и логических операций. Кроме того, для получения результатов, которые в аналоговой аппаратуре могут быть получены с помощью относительно простой цепи, при цифровом методе обработки требуется привлечение большого числа относительно сложных узлов (регистров, сумматора, множительного устройства) и устройства, координирующего их работу.

Следовательно, для цифровой обработки требуются устройства с большим числом элементов. При этом обеспечивается более низкое быстродействие, чем при аналоговых методах обработки. Однако цифровые методы по сравнению с аналоговыми имеют ряд достоинств: возможность обеспечения любой точности обработки, высокая помехозащищенность, высокая стабильность характеристик обработки, возможность выполнения таких видов обработки, которые аналоговыми методами либо трудновыполнимы, либо вовсе невыполнимы.

Аппаратурная сложность цифровых устройств прежде приводила к высокой стоимости оборудования, большим его габаритным размерам и массе, высокому 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.

### Структура процессора

Процессор осуществляет непосредственно обработку данных и программное управление процессом обработки данных. Он синтезируется в виде соединения двух устройств: операционного и управляющего (рис. 4.4).

*Операционное устройство* (ОУ) — устройство, в котором выполняются операции. Оно включает в качестве узлов регистры, сумматоры, каналы передачи информации, мультиплексоры для коммутации каналов, шифраторы, дешифраторы и т.д. *Управляющее устройство* (УУ) координирует действия узлов операционного устройства; оно вырабатывает в некоторой 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:

1) установка регистра в некоторое состояние (например, запись в регистр R1 числа 0, обозначаемая $R1 \leftarrow 0$);

2) инвертирование содержимого разрядов регистра (например, если регистр R2 содержал двоичное число 101101, то после инвертирования его содержимое будет равно 010010; такое действие обозначают $R2 \leftarrow (\overline{R2})$);

3) пересылка содержимого одного узла в другой (например, пересылка содержимого регистра R2 в регистр R1, обозначаемая $R1 \leftarrow (R2)$);

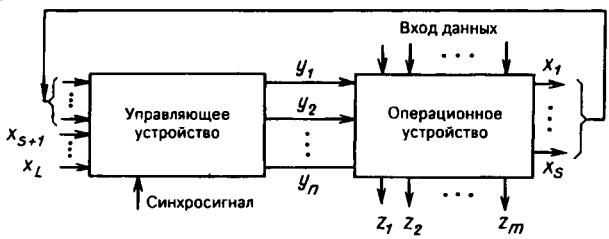



Рис. 4.4

4) сдвиг содержимого узла влево, вправо (например, сдвиг на один разряд влево содержимого регистра R1, обозначаемый R1 ← СдвЛ (R1));

5) счет, при котором число в счетчике (регистре) возрастает или убывает на единицу (Сч ← (Сч ± 1));

6) сложение (например, R2 ← (R2) + (R1));

7) сравнение содержимого регистра на равенство с некоторым числом; результат сравнения: *лог.1* (при выполнении равенства) либо *лог.0* (при невыполнении 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 R2 ← 0, Сч ← (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 этой микрооперации, подать сигнал (например, напряжение уровня *лог.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. 4.4) для выполнения микрокоманд может происходить в зависимости от состояния узлов операционного устройства, определяемого сигналами $x_1, \ldots, x_s$, которые подаются 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 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 микрокоманды могут представляться 9-разрядным двоичным кодом. Число комбинаций такого кода составляет $2^9 = 512$. Таким образом, каждой

микрокоманде может быть поставлена в соответствие одна из этих комбинаций 9-разрядного кода (например, первой микрокоманде может быть поставлена в соответствие кодовая комбинация 000 000 000, второй микрокоманде — комбинация 000 000 001 и т.д.). При этом микрокоманда на входе операционного устройства будет задаваться некоторой 9-разрядной кодовой 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 $z_1, \ldots, z_m$.

### Два подхода к построению процессоров

Существует два принципиально разных подхода к проектированию микропрограммного автомата (управляющего устройства): использование принципа схемной логики и 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, основанные на таком принципе схемной логики, способны обеспечивать наивысшее быстродействие при заданном типе технологии элементов. Недостаток этого принципа построения процессора состоит в трудности использования 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. специализация устройства на выполнение определенных функций) обес-

печивается 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 разнообразных задач. В этом случае число типов БИС, необходимых для построения управляющего устройства, окажется небольшим, а потребность в БИС каждого типа высокой, что обеспечит целесообразность их выпуска промышленностью.

До сих пор речь шла о построении управляюших устройств процессоров. Теперь рассмотрим условия для широкого использования БИС в операционных устройствах процессоров. Можно построить операционное устройство с таким набором узлов и такой схемой их соединения, которые обеспечили бы решение разнообразных задач. Задача, решаемая подобным 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 устройстаа, специализированного для решения конкретной задачи.

Процессор, построенный на одной или нескольких БИС, называется *микропроцессором*.

Набор БИС, обеспечивающих построение цифровых устройств, образует *микропроцессорный комплект* (МПК). Он позволяет совместно со сравнительно небольшим числом микросхем средней и малой степени интеграции создавать миниатюрные вычислительные устройства для разнообразных применений.

С помошью МПК реализуются *микропроцессорные системы* (МПС). Если в устройстве, построенном на принципе схемной логики, любое изменение или расширение выполняемых функций влечет демонтаж устройства и монтаж другого устройства по новой схеме, то в МПС благодаря использованию принципа программируемой логики изменение функций может быть достигнуто заменой хранящейся в памяти программы новой программой, соответствующей новым функциям устройства. Подобная гибкость вместе с другими связанными с использованием БИС достоинствами (низкой стоимостью, малыми размерами), а также высокая точность и помехозащищенность, характерные для цифровых методов, обусловили бурное внедрение МПС в различные сферы производства, научные исследования и бытовую технику.

Микропроцессорные системы в свою очередь обеспечили широкое

использование цифровых методов в различных технических применениях, и размах внедрения этих новых методов рассматривается как революция в технике.

### Цифровые автоматы

Процессор является примером цифрового автомата — устройства, осуществляющего прием, хранение и преобразование дискретной информации по некоторому алгоритму. Теорию автоматов подразделяют на *абстрактную* и *структурную*. Абстрактная теория изучает поведение автомата, отвлекаясь от структуры (т.е. способа его построения, схемной реализации).

Автомат под действием входных сигналов принимает состояния в соответствии с набором значений входных сигналов и выдает сигнал, зависящий от внутреннего состояния либо от внутреннего состояния и входных сигналов. Для хранения внутреннего состояния автомат должен иметь память; таким образом, автомат является устройством с памятью, т.е. устройством последовательностного типа.

Несмотря на то что реальные автоматы могут иметь несколько входов и выходов, на каждом из которых в дискретные моменты времени (определяемые тактом работы) образуются сигналы, соответствующие *лог.0* и *лог.1*, в абстрактной теории удобно рассматривать автоматы с одним входом и одним выходом. Возможность такого рассмотрения заключается в следующем. Пусть число реальных входов равно двум. Так как на каждом входе может быть *лог.0* или *лог.1*, автомат оказывается под воздействием в каждый тактовый момент одного из четырех входных сигналов: $x_1 = (0,0)$, $x_2 = (0,1)$, $x_3 = (1,0)$, $x_4 = (1,1)$, $x_1, x_2, x_3, x_4$ — отдельные значения переменной $X$.

Аналогично несколько реальных выходов приводятся к одному выходу (рис. 4.5). Функционирование цифрового автомата происходит на трех множествах:

| | |
|---|---|
| *множестве возможных входных сигналов* | $x_1, x_2, \ldots, n_n$; |
| *множестве внутренних состояний* | $a_0, a_1, \ldots, a_k$; |
| *множестве возможных выходных сигналов* | $y_1, y_2, \ldots, y_m$. |

Одно из состояний является начальным (состояние $a_0$), и перед началом работы автомат всегда устанавливается в это состояние.

Работа автомата определяется следущими функциями:

*функцией переходов f*, которая определяет состояние автомата $a(t + 1)$ в момент $t + 1$ в зависимости от состояния автомата $a(t)$ и значения входного сигнала $x(t)$ в момент $t$:

$$a(t + 1) = f(a(t); x(t));$$


Рис. 4.5

*функцией выходов* $\varphi$, определяющей зависимость выходного сигнала автомата $y(t)$ от состояния автомата $a(t)$ и входного сигнала $x(t)$:

$$y(t) = \varphi(a(t); x(t)).$$

Автомат с такой функцией выходов называется *автоматом Мили*. Другой тип автомата — *автомат Мура*. Особенность автомата Мура в том, что в нем выходной сигнал зависит лишь от внутреннего состояния $a(t)$ и не зависит от входного сигнала. Функции переходов и выходов для него имеют вид

$$a(t + 1) = f(a(t); x(t)), \quad y(t) = \varphi(a(t)).$$

Функционирование автомата может быть задано в форме *таблиц переходов* и *выходов* либо с помощью так называемого *графа*.

Задание автомата Мили в виде таблицы переходов и выходов представлено в табл. 4.1. Здесь в клетках, расположенных на пересечении столбцов текущего состояния автомата $(a_0, \ldots, a_2)$ со строками текущего значения входного сигнала $(x_1, x_2)$, указываются следующее состояние (состояние в следующем такте) и значение выходного сигнала (например, $a_1/y_2$).

*Таблица 4.1*

| Входной сигнал | $a_0$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|---|
| $x_1$ | $a_1/y_2$ | $a_3/y_2$ | $a_1/y_1$ | $a_0/y_1$ |
| $x_2$ | $a_0/y_1$ | $a_2/y_1$ | $a_3/y_1$ | $a_3/y_3$ |

Функционирование этого же автомата в форме графа представлено на рис. 4.6. Граф состоит из узлов, отождествляемых с состояниями автомата. Связи между узлами показывают переходы автомата из одного состояния в другое под воздействием входных сигналов. На каждой связи указывается формируемый на выходе автомата сигнал. Задавая произвольное входное слово в виде последовательности сигналов $x_1$ и $x_2$, можно определять соответствующее выходное слово для данного автомата:

| | |
|---|---|
| входное слово | $x_2\ x_1\ x_2\ x_2\ x_1\ x_2\ x_1\ x_1\ x_2\ \ldots$ |
| состояние автомата | $a_0\ a_0\ a_1\ a_2\ a_3\ a_0\ a_0\ a_1\ a_3\ a_3$ |
| выходное слово | $y_1\ y_2\ y_1\ y_1\ y_1\ y_1\ y_2\ y_2\ y_3\ \ldots$ |

Пример задания автомата Мура в форме таблицы переходов и выходов показан в табл. 4.2. Соответствующий этому автомату граф приведен на рис. 4.7.

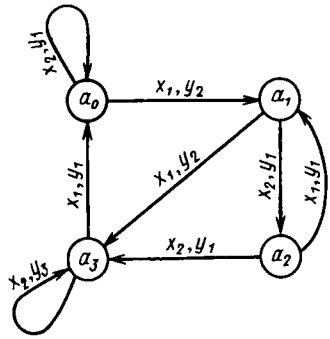



Рис. 4.6

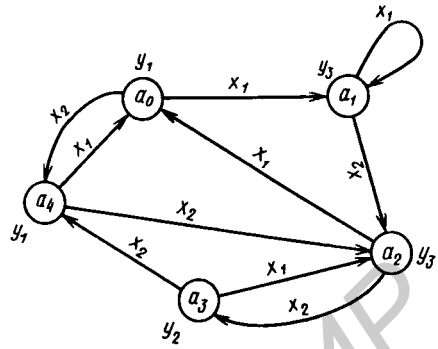



Рис. 4.7

*Таблица 4.2*

| Входной сигнал | $y_1$ | $y_3$ | $y_3$ | $y_2$ | $y_1$ |
|---|---|---|---|---|---|
| | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ |
| $x_1$ | $a_1$ | $a_1$ | $a_0$ | $a_2$ | $a_0$ |
| $x_2$ | $a_4$ | $a_2$ | $a_3$ | $a_4$ | $a_2$ |

Данный автомат Мура производит следующее преобразование данных:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| входное слово | $x_2$ | $x_2$ | $x_2$ | $x_1$ | $x_1$ | $x_2$ | $x_1$ | $x_1$ | $x_2$ | ... |
| состояние автомата | $a_0$ | $a_4$ | $a_2$ | $a_3$ | $a_2$ | $a_0$ | $a_4$ | $a_0$ | $a_1$ | $a_2$ |
| выходное слово | $y_1$ | $y_1$ | $y_3$ | $y_2$ | $y_3$ | $y_1$ | $y_1$ | $y_1$ | $y_3$ | ... |

## 4.3. СИНТЕЗ ПРОЦЕССОРА С ИСПОЛЬЗОВАНИЕМ ПРИНЦИПА СХЕМНОЙ ЛОГИКИ

Рассмотрим методику построения процессора на примере реализации устройства, выполняющего операцию умножения двоичных чисел без знаков 1101 и 1011. Как ранее было показано (§ 2.3), для получения произведения необходимо формировать частичные произведения (произведения множимого на цифры отдельных разрядов множителя) и суммировать их с определенным сдвигом относительно друг друга. Предусмотрим суммирование частичных произведений, начиная с младшего частичного произведения. В табл. 4.3 приведена последовательность действий при умножении чисел $1101_2$ и $1011_2$.

*Таблица 4.3*

| Множимое (R1) | Старшие разряды произведения (R3) | Множитель и младшие разряды произведения (R2) | Выполняемое действие |
|---|---|---|---|
| 1101 | 0000 | 1011 | Исходное состояние |
| | + | | |
| | 1101 | | |
| | 0←1101 | | Суммирование |
| | ↘ | | |
| | 0 0110 | → 1101 | Сдвиг (R3) и (R2) |
| | + | | |
| | 1101 | | |
| | 1←0011 | | Суммирование |
| | ↘ | | |
| | 0 1001 | → 1110 | Сдвиг (R3) и (R2) |
| | ↘ | | |
| | 0 0100 | → 1111 | Сдвиг (R3) и (R2) |
| | + | | |
| | 1101 | | |
| | 1←0001 | | Суммирование |
| | ↘ | | |
| | 0 1000 | → 1111 | Сдвиг (R3) и (R2) |
| | Произведение | | |



### Синтез операционного устройства

В соответствии с описанным выше процессом для выполнения операции умножения необходимо в операционном устройстве иметь регистры R1, R2, R3, сумматор (См) и счетчик (Сч) числа повторений цикла. Регистр R3′ — регистр временного хранения; при низком уровне синхронизирующего сигнала *C* в него передается содержимое регистра R3; благодаря этому в течение времени суммирования чисел в См при высоком уровне *C* числа на входах См поддерживаются неизменными.

На рис. 4.8 показана структурная схема операционного устройства. В регистре R2 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 R3, выполняемая под действием управляющего сигнала $y_2$. Сумматор производит суммирование чисел, поступающих с выходов регистров R1 и R3′; 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 R3, который должен иметь то же число разрядов $n + 1$, что и сумматор. Кроме микрооперации приема суммы в регистре R3 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 R2 и $x_2$ — результат проверки на нуль содержимого счетчика. Приведем в условной записи список выполняемых в узлах операционного устройства микроопераций и список формируемых признаков:

$y_1$: R2 ← Сдв П(R2), $y_2$: R2[$n$] ← R3[1],

$y_3$: R3 ← (См), $y_4$: R3 ← 0,

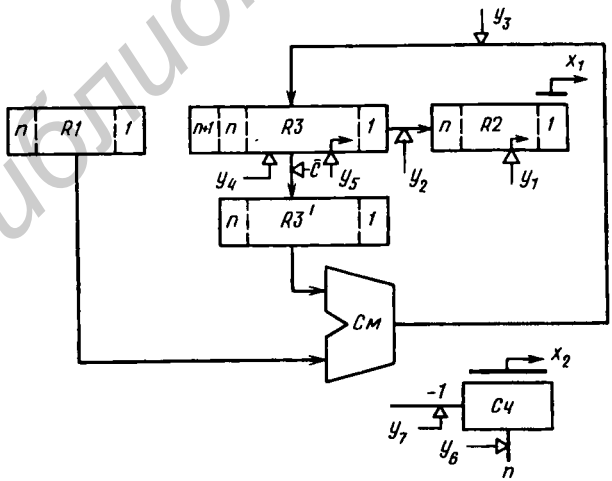



Рис. 4.8

$y_5$: R3 ← Сдв П(R3), $y_6$: Сч ← $n$,

$y_7$: Сч ← (Сч) – 1,

$x_1$: (R2[1]) = 1, $x_2$: (Сч) = 0.

Запись R2[$n$] и R3[1] означает соответственно $n$-й разряд регистра R2 и 1-й разряд регистра R3.

Мы здесь не задавались целью показать полный список используемых в операционном устройстве микроопераций. Так, рассматривая процессы с момента, когда в регистры R1 и R2 уже помещены соответственно множимое и множитель, мы не показываем микроопераций приема чисел в эти регистры.

### Синтез управляющего устройства в форме автомата Мили

Процесс синтеза разобьем на этапы и последовательно рассмотрим каждый из них.

**Построение схемы алгоритма в микрооперациях.** На рис. 4.9,*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.

**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. 4.9,*а* последовательности в четырех тактовых периодах, но при построении регистров на триггерах, управляемых фронтом синхросигнала, эти микрооперации могут выполняться одновременно и, следовательно, могут быть объединены в микрокоманду $Y_3$. На рис. 4.9,*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. 4.9,*б*), по следующему правилу: символом $a_0$ отмечаются начало и конец схемы, затем последовательно символами $a_1$, $a_2$, ... — входы блоков, следующих за операторными блоками (блоками, содержащими микрокоманды). В рассматриваемой схеме алгоритма блок 1 является операторным блоком; символом $a_1$ отмечается

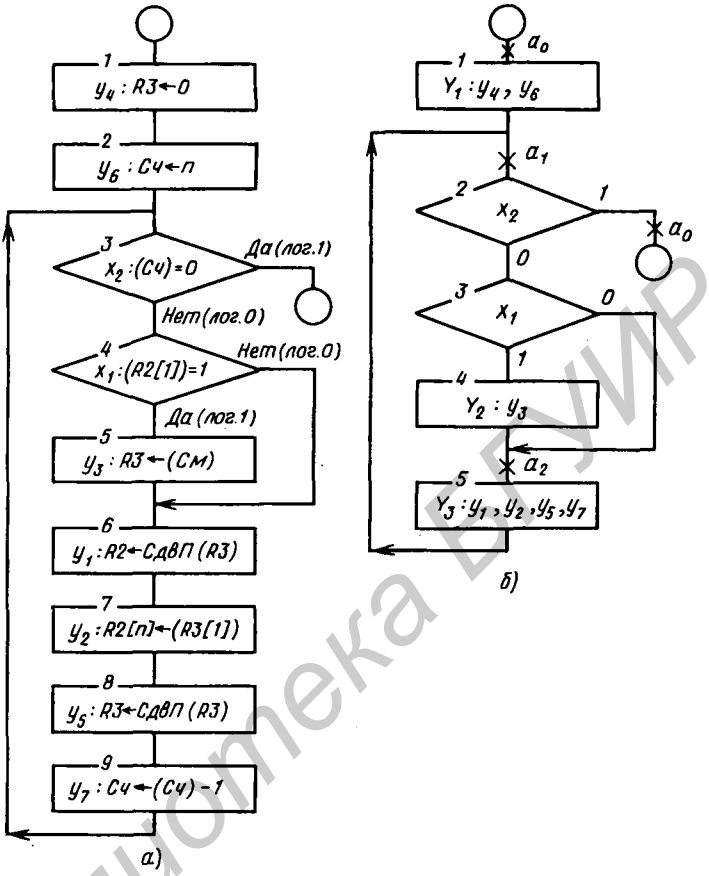



Рис. 4.9

вход следующего за ним блока 2 условного перехода по признаку $x_2$; затем выбирается следующий операторный блок 4 и символом $a_2$ отмечается вход следующего за ним блока 5. Полученные отметки $a_0$, $a_1$, $a_2$ соответствуют состояниям устройства. Итак, рассматриваемое устройство имеет три состояния.

Теперь можно приступить к построению графа функционирования устройства. Состояния устройства в графе представляются узлами (изображаемыми кружками с записью внутри них обозначений соответствующих состояний), дугами, соединяющими узлы, показываются возможные переходы между узлами (на схеме алгоритма эти переходы соответствуют переходам между соответствующими отметками), на

дугах записываются условия (значения признаков, поступающих на входы управляющего устройства с выхода операционного), при которых происходит переход, и микрокоманда, которая должна выдаваться устройством; отсутствие признака или микрокоманды обозначено знаком "-". Граф синтезируемого управляющего устройства приведен на рис. 4.10.

**Кодирование состояний устройства.** При кодировании 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 $M \le 2^k$.

*Таблица 4.4*

| Состояние устройства | Кодовая комбинация | |
|---|---|---|
| | $Q_2$ | $Q_1$ |
| $a_0$ | 0 | 0 |
| $a_1$ | 0 | 1 |
| $a_2$ | 1 | 0 |

В рассматриваемом случае $M = 3$ и $k = 2$. Таким образом, состояния управляющего устройства отображаются двухразрядными кодовыми комбинациями, задаваемыми состояниями триггеров 1 и 2 ($Q_1$ и $Q_2$). Соответствие между состояниями устройства и кодовыми комбинациями зададим табл. 4.4.

**Структурная схема управляющего устройства**. Структурная схема рассматриваемого устройства представлена на рис. 4.11. 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 которое должно перейти управляющее устройство. При этом формируются также сигналы $S_2$, $R_2$, $S_1$, $R_1$, которые в момент положительного фронта синхросигнала $C$ устанавливают в регистре кодовую комбинацию, соответствующую следующему состоянию устройства.

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, значения входных условий, данные следующего состояния, в которое должно перейти устройство, и выходные сигналы комбинационного узла. Функционирование комбинационного узла рассматриваемого управляющего устройства представлено в табл. 4.5.

По значению текущего состояния, принимаемому из регистра состояния, и поступающим из операционного устройства значениям

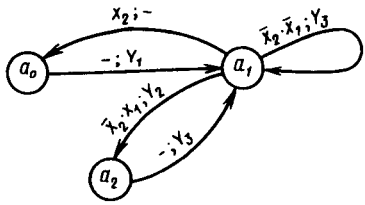



Рис. 4.10

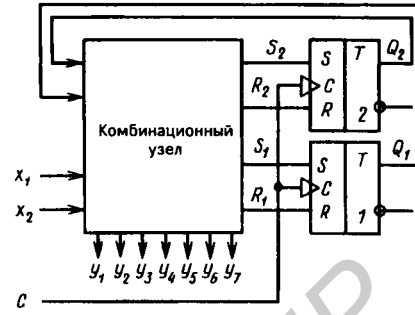



Рис. 4.11

условий перехода $x_1$ и $x_2$ в таблице определяются следующее состояние управляющего устройства, сигналы $S_2$, $R_2$, $S_1$, $R_1$ для установки регистра в соответствующее 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. 3.11 находится дуга, ведущая в узел, соответствующий состоянию $a_1$; найденная дуга выходит из узла $a_0$, 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 $a_0 \to a_1$ связан с переходом $Q_1 : 0 \to 1$. Из таблицы переходов RS-триггера (табл. 4.6) определяем, что $S_1 = 1$. Кроме этого сигнала на выходе комбинационного узла должны формироваться управляющие сигналы микрокоманды $Y_1 : y_4, y_6$.

*Таблица 4.5*

| 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 | $x_2$ | $R_1$ | — |

Далее в следующую строку таблицы заносятся данные, соответствующие переходу $a_1 \to a_1$. Из графа на рис. 4.10 выясняется, что переход происходит при выполнении условий $\bar{x}_1 = 1$ и $\bar{x}_2 = 1$ 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.

*Таблица 4.6*

| Вид перехода триггера | Сигналы на входах RS-триггера | |
|---|---|---|
| | $S$ | $R$ |
| $0 \to 0$ | 0 | × |
| $0 \to 1$ | 1 | 0 |
| $1 \to 0$ | 0 | 1 |
| $1 \to 1$ | × | 0 |

**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:

$$S_1, y_4, y_6 = a_0; \quad y_1, y_2, y_5, y_7 = \bar{x}_1 \cdot \bar{x}_2 \cdot a_1;$$

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

$$S_2, R_1, y_3 = x_1 \cdot \bar{x}_2 \cdot a_1; \quad R_1 = x_2 \cdot a_1.$$

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 (если это необходимо) к минимальной форме:

$$S_2 = x_1 \cdot \bar{x}_2 \cdot a_1; \; R_2 = a_2; \; S_1 = a_0 \vee a_2;$$

$$R_1 = x_1 \cdot \bar{x}_2 \cdot a_1 \vee x_2 \cdot a_1; \; y_4, y_6 = a_0;$$

$$y_1, y_2, y_5, y_7 = \bar{x}_1 \cdot \bar{x}_2 \cdot a_1 \vee a_2; \; y_3 = x_1 \cdot \bar{x}_2 \cdot a_1.$$

**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

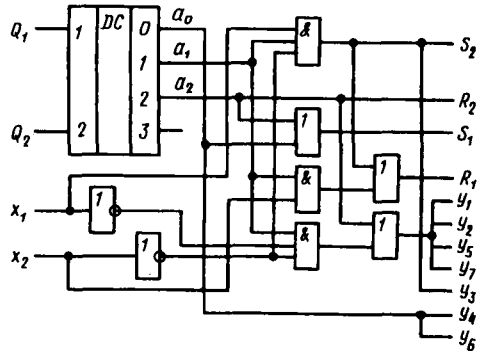



Рис. 4.12

логическими выражениями. Схема комбинационного узла рассматриваемого управляющего устройства приведена на рис. 4.12.

**Выполнение программы.** Мы рассмотрели синтез управляющего устройства для реализации операции умножения. Очевидно, подобные устройства могут быть построены для управления выполнением других операций. И если в управляющем устройстве процессора предусмотреть такие устройства $УУ_{оп1}$ ... $УУ_{опk}$, то, включая то или иное устройство можно обеспечить реализацию различных операций на одном и том же оборудовании операционного устройства.

Вид операции, подлежащей исполнению в процессоре, будем представлять *командой*. С помощью дешифратора код команды можно преобразовать в сигналы, производящие включение устройств, которые управляют выполнением соответствующих операций (рис. 4.13). При этом возникает возможность записать алгоритм сложной задачи в виде последовательности команд (которая будет соответствовать последова-

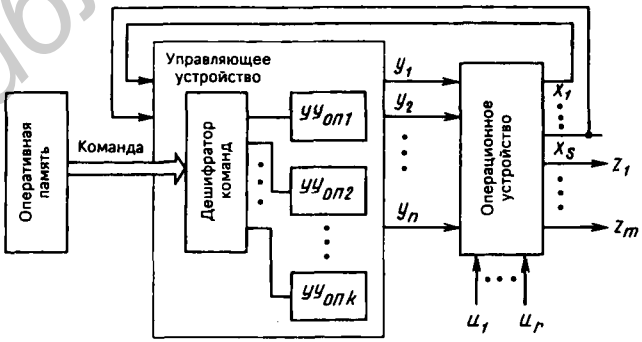



Рис. 4.13

тельности таких выполняемых операций, как умножение, деление и др.). Эта последовательность команд образует *программу*, хранимую в оперативной памяти. Считывая из оперативной памяти команды и исполняя их в процессоре, можно решить сложную задачу.

Определим время реализации алгоритма умножения при рассмотренном способе построения процессора. Обратимся к графу на рис. 4.10. Каждый переход из одного состояния в другое происходит за один тактовый период. Таким образом, переход из состояния $a_0$ в состояние $a_1$ требует одного 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 множителя 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 переходов $N_т = 2 + 2n$. При $n >> 1$ имеем $N_т \approx 2n$. Если длительность тактового периода равна 1, то время выполнения операции умножения $t_{умн} = N_т T = 2nT$.

### Синтез управляющего устройства в форме автомата Мура

**Построение графа функционирования.** Для определения состояния управляющего устройства в схеме алгоритма в микрокомандах, представленной на рис.4.9,*б*, выполним разметку, которая для автомата Мура производится по иному правилу, чем для автомата Мили. Воспользуемся следующим правилом: символом $a_0$ отметим начало и конец схемы алгоритма; символами $a_1$, $a_2$, ... — операторные блоки. Отмеченная таким образом схема алгоритма представлена на рис. 4.14. Из рисунка следует, что устройство управления, синтезированное в форме автомата Мура, имеет четыре состояния: $a_0, a_1, a_2, a_3$.

*Таблица 4.7*

| Состояние устройства | Кодовая комбинация | |
|---|---|---|
| | $Q_2$ | $Q_1$ |
| $a_0$ | 0 | 0 |
| $a_1$ | 0 | 1 |
| $a_2$ | 1 | 0 |
| $a_3$ | 1 | 1 |

**Кодирование состояний устройства.** Для кодирования состояний устройства можно использовать двухразрядный код. Выберем представленное в табл. 4.7 соответствие между состояниями устройства и кодовыми комбинациями.

**Построение таблицы функционирования комбинационного узла.** В табл. 4.8 приведена таблица функционирования комбинационного узла устройства управления процессора, соответствующая графу на рис. 4.15.

*Таблица 4.8*

| 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 | — | $S_1$ | — |
| $a_1$ | 0 | 1 | $a_2$ | 1 | 0 | $\bar{x}_2 \cdot x_1$ | $S_2, R_1$ | $Y_1 : y_4, y_6$ |
| $a_1$ | 0 | 1 | $a_3$ | 1 | 1 | $\bar{x}_2 \cdot x_1$ | $S_2$ | $Y_1 : y_4, y_6$ |
| $a_2$ | 1 | 0 | $a_3$ | 1 | 1 | — | $S_1$ | $Y_2 : y_3$ |
| $a_3$ | 1 | 1 | $a_3$ | 1 | 1 | $\bar{x}_2 \cdot \bar{x}_1$ | — | $Y_3 : y_1, y_2, y_5, y_7$ |
| $a_3$ | 1 | 1 | $a_2$ | 1 | 0 | $\bar{x}_2 \cdot x_1$ | $R_1$ | $Y_3 : y_1, y_2, y_5, y_7$ |
| $a_3$ | 1 | 1 | $a_0$ | 0 | 0 | $x_2$ | $R_1, R_2$ | $Y_3 : y_1, y_2, y_5, y_7$ |

**Запись логических выражений для выходных величин комбинационного узла.** Так же как и при синтезе устройства в форме автомата Мили, записываем логические выражения, соответствующие отдельным строкам табл. 4.8:

$y_4, y_6 = a_1$;
$y_3 = a_2$;
$y_1, y_2, y_5, y_7 = a_3$;

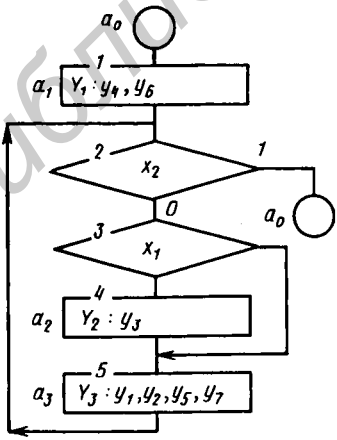



Рис. 4.14

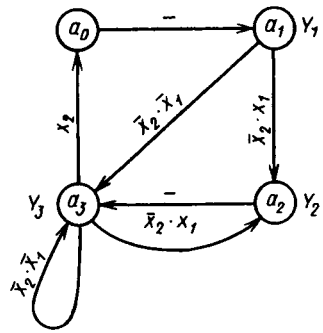



Рис. 4.15

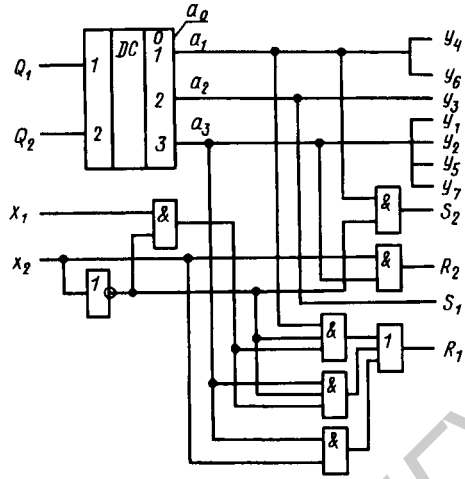



Рис. 4.16

$$
\left.\begin{array}{l} S_2, R_1 = a_1 \cdot \overline{x}_2 \cdot x_1; \\ S_2 = a_1 \cdot \overline{x}_2 \cdot \overline{x}_1; \end{array}\right\} \quad S_2 = a_1 \cdot \overline{x}_2 \cdot x_1 \vee \alpha_1 \cdot \overline{x}_2 \cdot \overline{x}_1 = a_1 \cdot \overline{x}_2;
$$

$$
R_2 = a_3 \cdot x_2;
$$

$$
\left.\begin{array}{l} S_1 = a_2; \\ R_1 = a_3 \cdot \overline{x}_2 \cdot x_1; \end{array}\right\} \Rightarrow S_1 = a_2;
$$

$$
\left.\begin{array}{l} R_1, R_2 = a_3 \cdot x_2; \end{array}\right\} \quad R_1 = a_1 \cdot \overline{x}_2 \cdot x_1 \vee a_3 \cdot \overline{x}_2 \cdot x_1 \vee a_3 \cdot x_2.
$$

**Построение логической схемы комбинационного узла.** По полученным логическим выражениям может быть построена схема комбинационного узла устройства управления, представленная на рис. 4.16.

### *Упражнения*

1. Синтезируйте процессор в форме автомата Мили для выполнения операции алгебраического суммирования двух чисел с фиксированной запятой, представленных в прямом коде.

2. Решите задачу п.1, построив процессор в форме автомата Мура.

3. Синтезируйте процессор в форме автомата Мили для выполнения операции алгебраического вычитания двух чисел с фиксированной запятой, представленных в прямом коде.

4. Решите задачу п.3, построив процессор в форме автомата Мура.

5. Синтезируйте процессор в форме автомата Мили для выполнения операции умножения двух чисел со знаком, представленных в дополнительном коде.

6. Решите задачу п.5, построив процессор в форме автомата Мура.

7. Реализуйте процессор в форме автомата Мили для выполнения операции деления двух чисел с фиксированной запятой, представленных в прямом коде.

8. Решите задачу п.7, построив процессор в форме автомата Мура.

## 4.4. СИНТЕЗ ПРОЦЕССОРА С ИСПОЛЬЗОВАНИЕМ ПРИНЦИПА ПРОГРАММИРУЕМОЙ ЛОГИКИ

### Принцип микропрограммного управления

Мы рассмотрели выполнение операций процессором в виде последовательности микрокоманд. Можно предусмотреть другой способ формирования 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 имвют уровни *лог. 0* и *лог.1*. Таким образом, каждой микрокоманде на выходе управляющего устройства соответствует некоторая кодовая комбинация. Такие кодовые комбинации, называемые *кодовыми комбинациями микрокоманд* (или просто микрокомандами), можно хранить в специально предназначенной для них *управляющей памяти*. 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*, а построенное на этом способе управляющее устройство — управляющим устройством с *программируемой логикой*.

На рис. 4.17,*а* изображена структурная схема процессора с управляющим устройством, построенным по принципу 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. Так решается задача поиска в УП микропрограммы, соответствующей данной команде. Адреса всех последующих МК определяются в БМУ следующим образом.

В формате МК (рис. 4.17,*б*) предусматривается поле адреса, которое содержит адрес очередной МК. Считав из УП микрокоманду, по содержимому ее поля адреса определим адрес следующей МК. Но так можно получить адреса МК при отсутствии в алгоритме разветвлений, т.е. условных переходов (УсП). Для реализации условных переходов в МК надо предусмотреть поле условных переходов, в котором указывается, имеет ли место условный или безусловный переход и при условном переходе — на значения каких условий следует ориентироваться при определении адреса очередной МК.

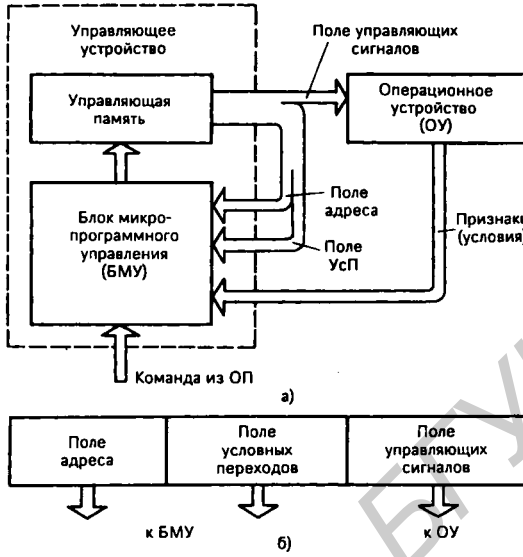



Рис. 4.17

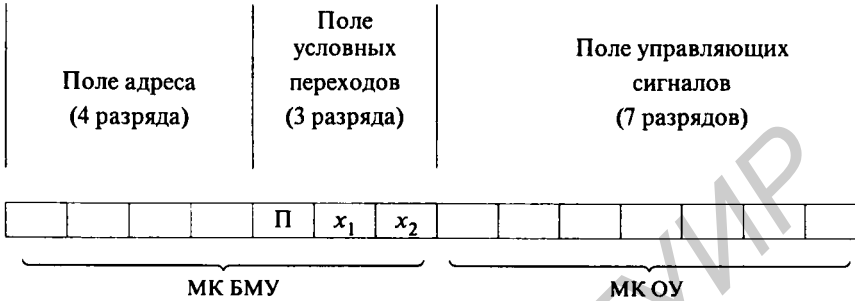
Пусть поле условных переходов построено следующим образом. Один из разрядов поля указывает вид перехода (например, 0 в этом разряде означает безусловный переход, 1 — условный переход). Кроме того, для каждого условия в поле условных переходов имеется разряд, указывающий участие данного условия в определении адреса. Если условный переход осуществляется по некоторому условию, то адрес очередной МК будем формировать 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 различающихся в младшем разряде адреса и очередная МК считывается из одной либо другой ячейки УП. Если модифицировать два разряда содержимого поля адреса, то можно осуществить разветвление на четыре направления.

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

Таким образом, микрокоманда может быть разбита на две части: одна часть — поле адреса и поле условных переходов — определяет функционирование БМУ при нахождении 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 три 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 семь 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. 4.9,*б*, построим схему алгоритма в микрокомандах (рис. 4.18) и таблицу

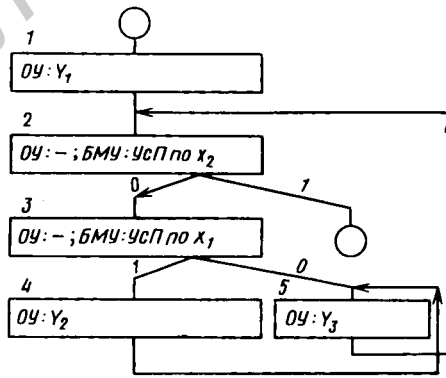
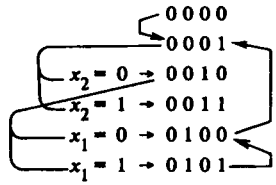



Рис. 4.18

*Таблица 4.9*

| Адрес ячейки УП | Содержимое ячейки (микрокоманда) |
|---|---|
|  | МК1 (МК ОУ: $Y_1$ ; МК БМУ:БП)<br>МК2 (МК ОУ: - ; МК БМУ:УсП по $x_2$)<br>МК3 (МК ОУ: - ; МК БМУ:УсП по $x_1$)<br>МК6 (Продолжение)<br>МК5 (МК ОУ: $Y_3$ ; МК БМУ:БП)<br>МК4 (МК ОУ: $Y_2$ ; МК БМУ:БП) |



размещения микрокоманд в ячейках УП (табл. 4.9). 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 — 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 $x_1$ замещает младший разряд содержимого поля адреса 0100; при $x_1 = 0$ происходит переход к ячейке с адресом 0100, содержащей МК5, при $x_1 = 1$ — к ячейке с адресом 0101, содержащей 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 таблице 4.10 приведена микропрограмма.

Определим время выполнения операции умножения. Оно равно произведению числа тактовых периодов, требуемых для выполнения операции, и длительности тактового периода. В рассматриваемом алгоритме операции умножения требуется однократное выполнение МК1 и $n$-кратное ($n$ — число разрядов множителя) выполнение мик-

рокоманд цикла. В цикле в зависимости от значения условия $x_1$ выполняется либо четыре микрокоманды (МК2, МК3, МК4, МК5), либо три микрокоманды (МК2, МК3, МК5). В наихудшем случае, 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 $N_т = 4n$ и время выполнения операции $t_{умн} = N_т T = 4nT$ (где $T$ — длительность тактового периода).

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

*Таблица 4.10*

| Адрес УП | Микрокоманда | | | | | | | | | | | Пояснения |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Поле адреса | Поле условных переходов | | | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ | |
| | | вид перехода | $x_1$ | $x_2$ | | | | | | | | |
| | МК БМУ | | | | МК ОУ | | | | | | | |
| 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 |

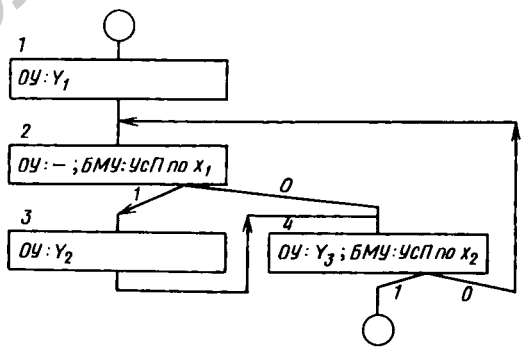
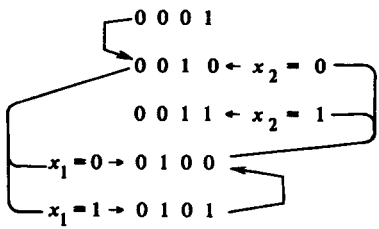



Рис. 4.19

*Таблица 4.11*

| Адрес ячейки УП | Содержимое ячейки (микрокоманда) |
|---|---|
|  | МК1 (МК ОУ:$Y_1$; МК БМУ: БП)<br>МК2 (МК ОУ: - ; МК БМУ:УсП по $x_1$)<br>МК5 (Продолжение)<br>МК4 (МК ОУ:$Y_3$; МК БМУ:УсП по $x_2$ )<br>МК3 (МК ОУ: $Y_2$; МК БМУ:БП) |



Рассмотрим, как можно снизить $N_т$ (и тем самым повысить скорость выполнения операции умножения) при использовании принципа программируемой логики. На рис. 4.19 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. 4.11 и 4.12 показаны соответственно размещение микрокоманд в УП и микропрограмма.

*Таблица 4.12*

| Адрес УП | Микрокоманда | | | | | | | | | | | | Пояснения |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Поле адреса | Поле условных переходов | | | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ | | |
| | | вид перехода | $x_1$ | $x_2$ | | | | | | | | | |
| | МК БМУ | | | | МК ОУ | | | | | | | | |
| 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 |

Дальнейшее сокращение $N_т$ возможно лишь при использовании условных переходов одновременно по двум 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

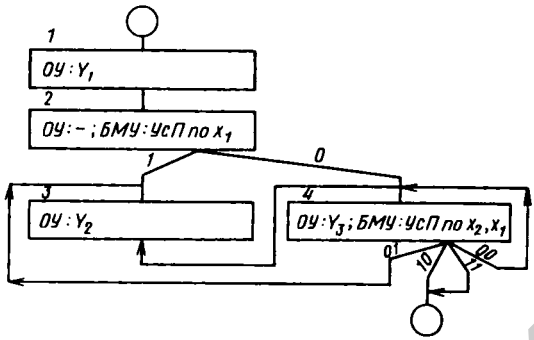



Рис. 4.20

условий $x_1$ и $x_2$. На рис. 4.20 показана схема алгоритма с таким разветвлением вычислительного процесса. Из схемы видно, что в этом случае $N_т = 2n$. В табл. 4.13 и 4.14 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 формирование 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 $x_2 = 1$).

*Таблица 4.13*

| Адрес ячейки УП | Содержимое ячейки (микрокоманда) |
|---|---|
| 0 0 1 0 | МК1 (МК ОУ:$Y_1$; МК БМУ:БП) |
| 0 0 1 1 | МК2 (МК ОУ: - ; МК БМУ: УсП по $x_1$) |
| $x_1 = 0 \rightarrow$ 0 1 0 0 $\leftarrow x_2, x_1 = 00$ | МК4 (МК ОУ:$Y_3$; МК БМУ: УсП по $x_2$, $x_1$) |
| $x_1 = 1 \rightarrow$ 0 1 0 1 $\leftarrow x_2, x_1 = 01$ | МК3 (МК ОУ:$Y_2$; МК БМУ: БП) |
| 0 1 1 0 $\leftarrow x_2, x_1 = 10$ | МК5 (Продолжение) |
| 0 1 1 1 $\leftarrow x_2, x_1 = 11$ | МК5 (Продолжение) |

*Таблица 4.14*

| Адрес УП | Микрокоманда | | | | | | | | | | | Пояснения |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Поле адреса | Поле условных переходов | | | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ | |
| | | вид перехода | $x_1$ | $x_2$ | | | | | | | | |
| | МК БМУ | | | | МК ОУ | | | | | | | |
| 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 быстродействия управляющих устройств**

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

Кроме того, быстродействие может снизиться дополнительно из-за большей длительности тактового периода. На рис. 4.21 приведена временная диаграмма работы процессора с УУ, построенным по принципу программируемой логики. В момент $t_1$ (при положительном фронте синхросигнала С) происходит прием в регистр адреса (РА) БМУ сфор-

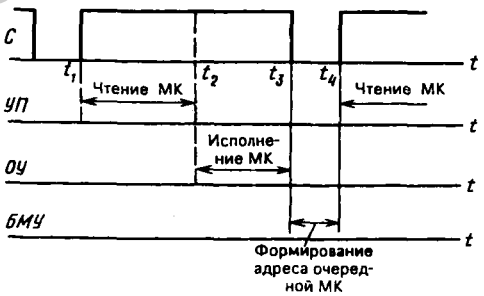



Рис. 4.21

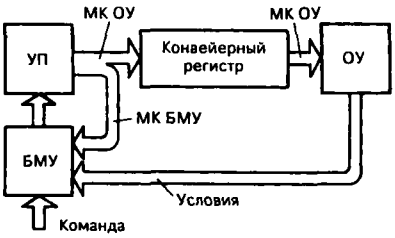



Рис. 4.22

мированного адреса следующей микрокоманды. Далее адрес из РА выдается на шину адреса и поступает в УП, где происходит чтение очередной МК. Процесс чтения завершается к моменту времени $t_2$, в этот момент в ОУ поступает МК ОУ и начинается процесс исполнения МК, который завершается к моменту $t_3$. В момент $t_4$ (при 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.

Такой последовательный способ чтения и исполнения МК вызывает увеличение 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. Для реализации этого способа требуется конвейерный регистр (рис. 4.22). Временная диаграмма работы процессора с конвейерным регистром приведена на рис. 4. 23.

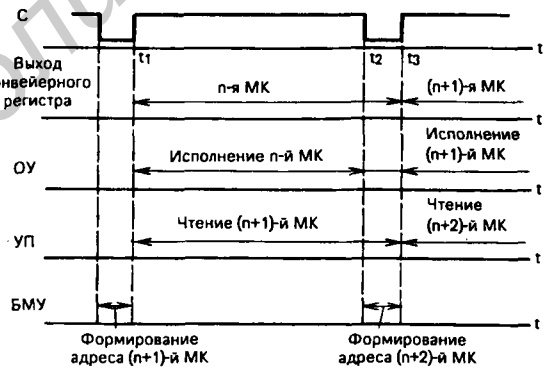



Рис. 4.23

В интервале времени $t_1$ ... $t_2$ из конвейерного регистра выдается $n$-я МК ОУ, считанная из УП на предыдущем тактовом периоде. В интервале времени $t_1$ ... $t_2$ она исполняется в ОУ. В этом же тактовом интервале производится чтение из УП ($n$ + 1)-й МК по адресу, сформированному в БМУ в предшествующем тактовом периоде. В интервале времени $t_2$ ... $t_3$ 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 по принципу схемной логики. Однако в этом случае дополнительно увеличивается число тактовых периодов, во время которых исполняется алгоритм.

## 4.5. МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

### Структура микропроцессорной системы

Рассмотрим структурную схему микропроцессорной системы (МПС), приведенную на рис. 4.24. Функционирование МПС сводится к следующей последовательности действий: получение данных от различных периферийных устройств (с клавиатуры терминала, от дисплеев, из каналов связи, от различного типа внешних запоминающих устройств), обработка данных и выдача результатов обработки на периферийные устройства (ПУ). При этом данные от ПУ, подлежащие обработке, могут поступать и в процессе их обработки.

Для выполнения этих действий в МПС кроме микропроцессора предусматриваются следующие устройства:

оперативная память (ОП), предназначенная для хранения и выдачи по запросам команд программ, определяющих работу микропроцессо-

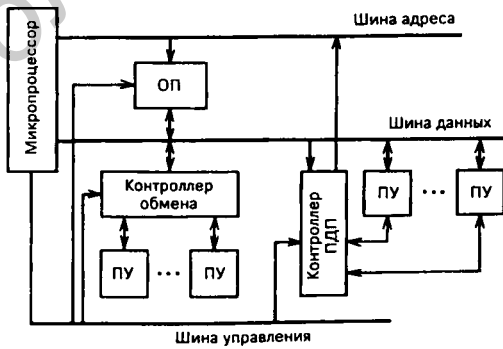



Рис. 4.24

ра, различных данных (исходных данных, промежуточных и конечных результатов обработки данных в микропроцессоре);

контроллеры — устройства, обеспечивающие обмен данными различных ПУ с микропроцессором и ОП.

Микропроцессор выдает на шину адреса номер (адрес) ячейки ОП, в которой хранится очередная команда, и из шины управления в ОП поступают сигналы, обеспечивающие считывание содержимого указываемой шиной адреса ячейки памяти. Оперативная память выдает запрошенную команду 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 при расшифровке команды выяснится, что участвующие в операции данные находятся в ОП, то микропроцессор выставляет на шину адреса адрес ячейки, хранящей эти данные; после выдачи данных из ОП микропроцессор принимает их через шину данных, затем выполняется операция над данными. После завершения текущей команды на шину адреса выдается адрес следующей команды, и описанный процесс повторяется.

Обмен данными с ПУ может осуществляться следующим образом. Группа ПУ подключается к шине данных МПС 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 УП.

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, он не может предусмотреть в программе 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 такой прерывающей программы микропроцессор возвращается к выпсднению 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.

### Построение микропроцессоров с использованием различных микропроцессорных комплектов

Все элементы микропроцессоров 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 мик-

рокоманд, обеспечивающую выполнение некоторой несложной операции. При поступлении в микропроцессор команды из 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 микропроцессорной системы, сокращая число используемых в ней элементов. Кроме того, упрощается процесс программирования, так как от программиста не требуется записывать выполняемые в каждом такте микрокоманды. Составляя программу, он оперирует командами, т.е. хранящимися в микропроцессоре 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 (требуюшей 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 (т.е. модификация системы 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 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.4.25,*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 16-разрядными данными и 20-разрядными адресами ОП, кроме объединения входов и выходов данных предусматривается использование этих выводов и для части разрядов адресной информации (при этом, очевидно, необходимо предусмотреть выдачу адреса и выдачу или прием данных в различные временные интервалы).

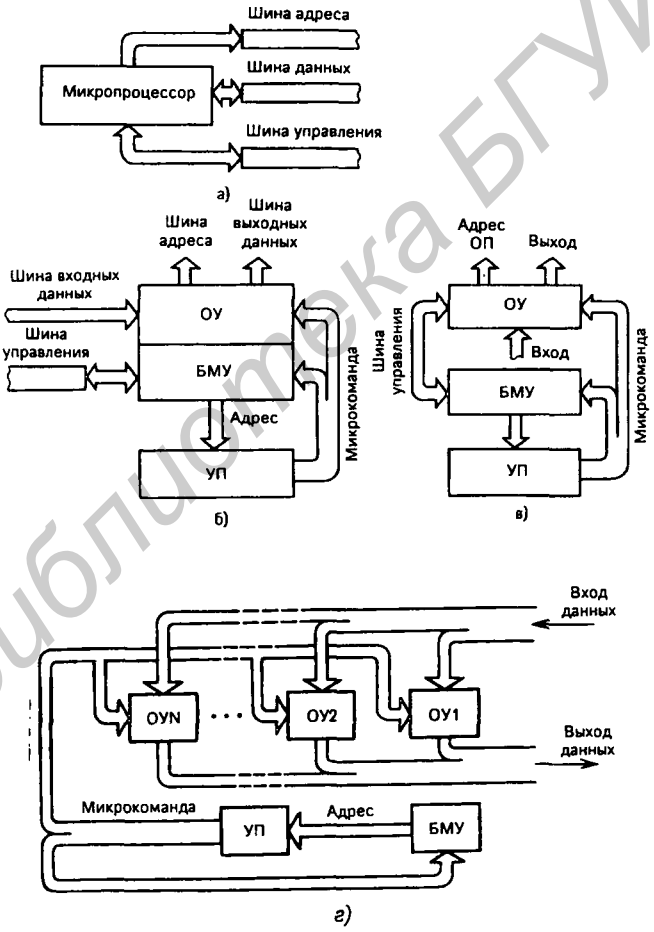



Рис. 4.25

Для того чтобы разработчик микропроцессорного устройства имел возможность программировать на языке микрокоманд, он должен иметь доступ к УП для записи в нее составленных микропрограмм. Такой доступ можно обеспечить, если УП вынести из микросхемы процессора, как показано на рис. 4.25,*б*.

Из сравнения схем на рис. 4.25,*а* и 4.25,*б* видно, что вариант на рис. 4.25,*б* потребует в микросхеме, содержащей ОУ и БМУ, предусмотреть существенно большее число выводов, чем в микросхеме на рис. 4.25,*а*. Это связано с необходимостью иметь в варианте на рис. 4.25,*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 микросхеме. В этом случае для сокращения числа выводов следует ОУ и БМУ выполнять не в общей микросхеме, а разнести в разные микросхемы, как показано на рис. 4.25,*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. 4.25,*г*.

### *Упражнения*

1. Синтезируйте процессор на основе принципа программируемой логики для выполнения операции алгебраического суммирования чисел с фиксированной запятой, представленных в прямом коде.

2. Синтезируйте процессор на основе принципа программируемой логики для выполнения операции алгебраического вычитания чисел с фиксированной запятой, представленных в прямом коде.

3. Синтезируйте процессор на основе принципа программируемой логики для выполнения операции умножения чисел с фиксированной запятой, представленных в дополнительном коде (используйте алгоритм Бута)

4. Синтезируйте процессор на основе принципа программируемой логики для выполнения операции деления чисел с фиксированной запятой, представленных в прямом коде.

# Глава 5. Микропроцессорные системы на основе микропроцессорного комплекта серии КР580

## 5.1. СОСТАВ МИКРОПРОЦЕССОРНОГО КОМПЛЕКТА

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

Комплект имеет следующие особенности. В нем предусмотрена БИС центрального процессора, содержащая в одной микросхеме операционное и управляющее устройства. Это существенно упрощает построение микропроцессорной системы. Кроме того, для облегчения программирования при управлении микросхемами МПК применяется фиксированный набор команд. Однако использование такого фиксированного набора команд снижает быстродействие микропроцессорной системы. Это связано с тем, что предлагаемый пользователю стандартный набор команд может оказаться плохо приспособленным для решения конкретной задачи.

Ряд микросхем, входящих в состав МПК серии КР580, выполнены по МОП-технологии, другие — по технологии ТТЛШ. Независимо от технологии входные и выходные сигналы соответствуют уровням логических схем ТТЛ-технологии. Это упрощает согласование микросхем серии КР580 с микросхемами ТТЛ-технологии любых серий. Следовательно, не возникает трудностей, если при построении микропроцессорной системы используются микросхемы ТТЛ-технологии, не входящие в МПК данной серии.

Все микросхемы МПК серии КР580 предназначены для работы в диапазоне температур –10...+70 °С. Микросхема центрального процессора КР580ВМ80А требует трех источников напряжения питания: +12 В ±5 %, +5 В ±5 %, –5 В ±5 %; микросхема генератора тактовых импульсов КР580ГФ24 — двух источников: +12 В ±5 %, +5 В ±5 %: все остальные микросхемы — одного источника +5 В ±5 %. В табл. 5.1 приведен состав МПК серии КР580.

*Таблица 5.1*

| Тип микросхемы | Наименование микросхемы | Выполняемая функция |
|---|---|---|
| КР580ВМ80А | 8-разрядный параллельный центральный процессор | Центральный процессор с фиксированной системой команд для обработки параллельной 8-разрядной информации |
| КР580ВВ51А | Программируемый последовательный интерфейс | Универсальное синхронно-асинхронное приемно-передающее устройство последовательной связи |
| КР580ВИ53 | Программируемый таймер | Формирует программно-управляемые временные задержки для синхронизации управляемых объектов в реальном масштабе времени |
| КР580ВВ55А | Программируемый параллельный интерфейс | Программируемый ввод-вывод параллельной информации различного формата |
| КР580ВТ57 | Программируемый контроллер прямого доступа к памяти | Высокоскоростной обмен информацией между памятью МПС и периферийными устройствами |
| КР580ВН59 | Программируемый контроллер прерываний | Обслуживает до восьми запросов на прерывания от внешних устройств |
| КР580ГФ24 | Генератор тактовых импульсов | Формирует две последовательности тактовых импульсов, необходимые для работы центрального процессора |
| КР580ВК28<br>КР580ВК38 | Системный контроллер | Формирует сигналы, предназначенные для управления различными устройствами, входящими в МПС |
| КР580ВА86<br>КР580ВА87 | Шинный формирователь | Двунаправленный 8-разрядный шинный формирователь с высокой нагрузочной способностью и тремя состояниями |
| КР580ИР82<br>КР580ИР83 | Буферный регистр | 8-разрядный буферный регистр с тремя состояниями |
| КР580ВГ75 | Программируемый интерфейс электронно-лучевой трубки | Контроллер вывода информации из памяти МПС на экран электронно-лучевой трубки |
| КР580ВВ79 | Программируемый интерфейс клавиатуры и дисплея | Контроллер ввода-вывода для клавиатуры и дисплея |

| Тип микросхемы | Наименование микросхемы | Выполняемая функция |
|---|---|---|
| КР580ВК91А | Интерфейс микропроцессор — канал общего пользования | Устройство сопряжения микропроцессора с информационно-измерительной системой |
| КР580ВА93 | Приемопередатчик микропроцессор — канал общего пользования | Программируемый приемопередатчик |

## 5.2. МИКРОПРОЦЕССОР КР580ВМ80А

### Структурная схема

На рис.5.1 приведена структурная схема БИС КР580ВМ80А. Кратко опишем ее узлы.

**Регистры данных.** Для хранения участвующих в операциях данных предусмотрено семь 8-разрядных регистров. Регистр А, называемый *аккумулятором*, предназначен для обмена информацией с внешними устройствами (т.е. содержимое этого регистра может быть выдано либо на вход микропроцессора, либо со входа микропроцессора в него может быть принято от внешнего устройства число), при выполнении арифметических, логических операций и операций сдвига он служит источником операнда (числа, участвующие в операции), в него помещается результат выполненной операции.

Шесть других регистров, обозначенных B, C, D, E, H, L, образуют блок *регистров общего назначения* (РОН) (название связано с тем, что в этих регистрах могут храниться как данные, так и адреса). Эти регистры могут использоваться как одиночные 8-разрядные регистры. Если необходимо хранить 16-разрядные двоичные числа, регистры объединяются в пары BC, DE, HL.

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

**Счетчик команд.** Регистр PC (16-разрядный) предназначен для хранения адреса команды; после выборки 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 пары регистров блока РОН.

При выдаче адреса содержимое соответствующего регистра передается в 16-разрядный регистр адреса РА, из которого далее через буферы адреса адрес поступает на 16-разрядную шину адреса. С этой шины

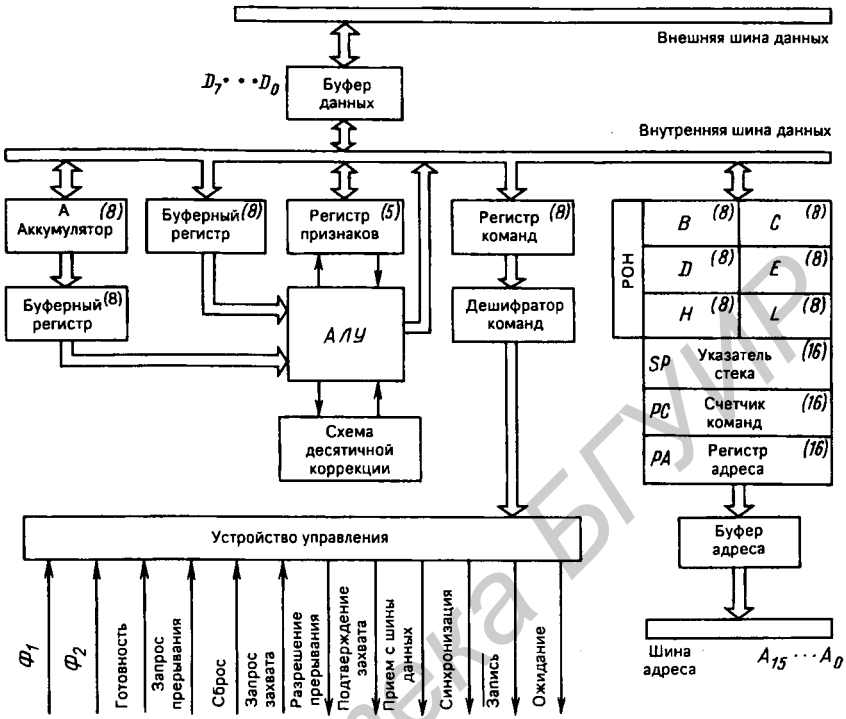



Рис. 5.1

адрес может быть принят в оперативную память. Число кодовых комбинаций 16-разрядного адреса равно $2^{16}$, каждая из этих кодовых комбинаций может определять адрес (номер) одной из ячеек оперативной памяти. Таким образом обеспечивается возможность обращения к памяти, содержащей до $2^{16} = 2^6 \cdot 2^{10} = 64$ К 8-разрядных слов (байтов).

**Арифметико-логическое устройство.** 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 (конъюнкции, дизъюнкции, неравнозначности, сравнения), а также четырех видов циклического сдвига. При реализации арифметических и логических операций одним из операндов служит содержимое аккумулятора, результат операции помещается в аккумулятор. Циклический сдвиг выполняется только над содержимым аккумулятора.

Предусмотрена возможность выполнения арифметических операций над десятичными числами, представленными в *коде 8421*. При хранении

десятичного числа разряды регистра делятся на две группы по четыре разряда, и в каждой группе разрядов хранится одна десятичная цифра, представленная в *коде 8421*. Таким образом, в регистре можно хранить 2-разрядное десятичное число. В § 2.3 показывалось, что 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 переноса* Tc при выполнении арифметических операций устанавливается в состояние, соответствующее переносу из старшего разряда числа, при выполнении операции сдвига — в состояние, соответствующее содержимому выдвигаемого из аккумулятора разряда;

*триггер нуля* Tz устанавливается в состояние 1, если результат операции АЛУ или операции приращения содержимого регистра равен нулю;

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

*триггер четности* Tp устанавливается в состояние 1, если число единиц в разрядах результата четное;

*триггер дополнительного переноса* Tv хранит возникающий при выполнении операции перенос из 4-го разряда.

**Блок управления.** Состоит из регистра команд, куда принимается первый байт команды, и *устройства управления*, 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. 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 0 и 1 предусмотрено еще третье состояние, в котором они имеют практически бесконечное выходное сопротивление и оказываются от-

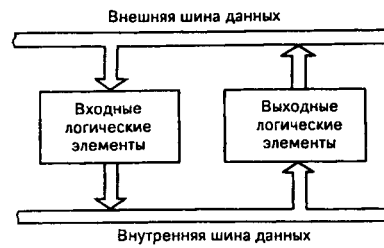


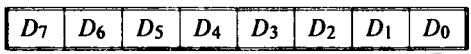

Рис. 5.2

ключенными от соответствующих шин. Такие буферы позволяют процессору отключаться от внешних шин (шин данных и адреса), предоставляя их в распоряжение внешних устройств, а также 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), что сокращает число выводов микросхемы.

На рис.5.2 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.

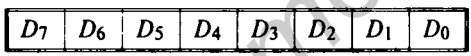
**Формат данных и команд**

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

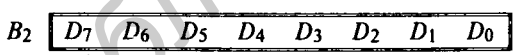



Для команд используются одно-, двух- и трехбайтовые форматы. Большинство команд является однобайтовыми.

Однобайтовый формат команды:

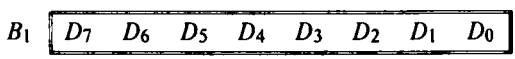



Двухбайтовый формат команды:


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

Трехбайтовый формат команды:




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B_3$ | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |

Старший байт адреса ОП или старший байт операнда

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

### Способы адресации

Для выполнения какой-либо операции в команде должно содержаться указание вида операции, а также откуда берутся участвующие в операции числа и куда помещается результат выполненной операции (т.е. указание об источниках и приемнике операндов). Под способами адресации понимают способы указания источников и приемников операндов. Опишем способы адресации, которые используются в микропроцессоре.

**Прямая адресация**. При этом способе адресом операнда является указанный в команде (в байте кода операции) адрес регистра микропроцессора (см. рис. 5.1). Адреса регистров приведены в следующей таблице:

| Регистр | Адрес регистра |
|---|---|
| B | 000 |
| C | 001 |
| D | 010 |
| E | 011 |
| H | 100 |
| L | 101 |
| M | 110 |
| A | 111 |

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

Покажем некоторые примеры команд с прямой адресацией, взятых из приведенной в табл. 5.2 системы команд микропроцессора. Здесь под мнемоникой команды понимают ее сокращенное обозначение, облегчающее запоминание команды.

В кодовой комбинации команды 01 001 010 два старших разряда (01) определяют вид операции (операция пересылки содержимого одного 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 (операции 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.

*Таблица 5.2.*

**Система команд КР580ВМ80А**

<table>
<tr><th>Обозначение</th><th colspan="2">Содержание</th><th>Обозначение</th><th colspan="2">Содержание</th></tr>
<tr><td rowspan="11"><$B_i$><br>$r_i$</td><td colspan="2">$i$ -й байт команды<br>Обозначение одного из регистров</td><td rowspan="11">F</td><td colspan="2">Обозначение состояния регистра признаков</td></tr>
<tr><td>$r_i$</td><td>Наименование регистра</td><td>Триггеры регистра F</td><td>Условие истинности</td></tr>
<tr><td>000</td><td>B</td><td>Tc (перенос)</td><td>Наличие переноса или заема</td></tr>
<tr><td>001</td><td>C</td><td>Tz (нуль)</td><td>Нулевой результат</td></tr>
<tr><td>010</td><td>D</td><td>Ts (знак)</td><td>Старший разряд результата равен 1</td></tr>
<tr><td>011</td><td>E</td><td>Tp(четность)</td><td>Число единиц четно</td></tr>
<tr><td>100</td><td>H</td><td>Tv (дополнительный перенос)</td><td>Перенос из младшей четверки разрядов</td></tr>
<tr><td>101</td><td>L</td><td>(r)</td><td colspan="2">Содержимое регистра r</td></tr>
<tr><td>110</td><td>M (память)</td><td>[(r)]</td><td colspan="2">Ячейка памяти, адрес которой (r)</td></tr>
<tr><td>111</td><td>A (аккумулятор)</td><td>$\wedge$</td><td colspan="2">Логическое И</td></tr>
<tr><td></td><td></td><td>$\oplus$</td><td colspan="2">Исключающее ИЛИ</td></tr>
<tr><td></td><td></td><td>$\vee$</td><td colspan="2">Логическое ИЛИ</td></tr>
<tr><td></td><td></td><td>$r_m$</td><td colspan="2">$m$-й бит регистра r</td></tr>
<tr><td></td><td></td><td>SP</td><td colspan="2">Указатель стека</td></tr>
<tr><td></td><td></td><td>PC</td><td colspan="2">Счетчик команд</td></tr>
<tr><td></td><td></td><td>$\leftarrow$</td><td colspan="2">Пересылка</td></tr>
</table>

| Структура кода команды | Байты | Циклы | Такты | Выполняемая операция | Мнемоника | Tz | Ts | Tc | Tv | Tp |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | **I. Команды пересылки** | | | | | | |
| $01r_ir_j$ | 1 | 1/2 | 5/7 | 1. Регистр – регистр: $r_i \leftarrow (r_j)$ | MOV $r_ir_j$ | – | – | – | – | – |
| $00r_i110$ | 2 | 2/3 | 7/10 | 2. Непосредственная загрузка регистра $r_i \leftarrow <B_2>$ | MVI $r_i$ | – | – | – | – | – |
| $00r_i001$ | 3 | 3 | 10 | 3. Непосредственная загрузка пары регистров $r_i \leftarrow <B_3>$; $r_{i+1} \leftarrow <B_2>$; при $r_i$ = 110: SP$\leftarrow <B_3> <B_2>$ | LXI $r_i$ | – | – | – | – | – |
| | | | | 4. Запоминание/загрузка (A) и (HL) | | | | | | |
| $00K_1010$ | | | | $K_1$ \| Операция | | | | | | | |
| | 1 | 2 | 7 | 000 \| [(BC)] $\leftarrow$ (A) | STAX B | – | – | – | – | – |
| | | | | 010 \| [(DE)] $\leftarrow$ (A) | STAX D | – | – | – | – | – |
| | | | | 001 \| A $\leftarrow$ [(BC)] | LDAX B | – | – | – | – | – |
| | | | | 011 \| A $\leftarrow$ [(DE)] | LDAX D | – | – | – | – | – |
| | 3 | 4 | 13 | 110 \| [$<B_3\,B_2>$] $\leftarrow$ (A) | STA | – | – | – | – | – |
| | | | | 111 \| A $\leftarrow$ [$<B_3\,B_2>$] | LDA | – | – | – | – | – |
| | 3 | 5 | 16 | 100 \| [$<B_3\,B_2>$] $\leftarrow$ (L); [$<B_3\,B_2>$ + 1] $\leftarrow$ (H) | SHLD | – | – | – | – | – |
| | | | | 101 \| L $\leftarrow$ [$<B_3\,B_2>$]; H $\leftarrow$ [$<B_3\,B_2>$+ 1] | LHLD | – | – | – | – | – |
| | | | | 5. Ввод из пар регистров в стек | | | | | | |
| $11r_i101$ | 1 | 3 | 11 | [SP – 1] $\leftarrow$ ($r_i$); [SP – 2] $\leftarrow$ ($r_{i+1}$); SP $\leftarrow$ (SP) – 2 | PUSH $r_i$ | – | – | – | – | – |
| 11110101 | 1 | 3 | 11 | 6. Ввод (A) и (F) в стек [SP – 1] $\leftarrow$ (A); [SP – 2] $\leftarrow$ (F); SP $\leftarrow$ (SP) – 2 | PUSH PSW | – | – | – | – | – |

| Структура кода команды | Байты | Циклы | Такты | Выполняемая операция | Мнемоника | Tz | Ts | Tc | Tv | Tp |
|---|---|---|---|---|---|---|---|---|---|---|
| $11r_i001$ | 1 | 3 | 10 | 7. Выбор из стека в пары регистров $(r_{i+1}) \leftarrow [SP]$; $r_i \leftarrow [SP+1]$; $SP \leftarrow (SP)+2$ | POP $r_i$ | – | – | – | – | – |
| 11110001 | 1 | 3 | 10 | 8. Выбор (A) и (F) из стека $F \leftarrow [SP]$; $A \leftarrow [SP+1]$; $SP \leftarrow (SP)+2$ | POP PSW | + | + | + | + | + |
| 11101011 | 1 | 1 | 4 | 9. Обмен между DE и HL $(H) \leftarrow\rightarrow (D)$; $(L) \leftarrow\rightarrow (E)$ | XCH D | – | – | – | – | – |
| 11100011 | 1 | 5 | 13 | 10. Обмен вершины стека с HL $(L) \leftarrow\rightarrow [(SP)]$; $(H) \leftarrow\rightarrow [(SP)+1]$ | XTH L | – | – | – | – | – |
| 11111001 | 1 | 1 | 5 | 11. Пересылка (HL) в SP $SP \leftarrow (HL)$ | SPHL | – | – | – | – | – |
| 11101001 | 1 | 1 | 5 | 12. Пересылка (HL) в PC $PC \leftarrow (HL)$ | PCHL | – | – | – | – | – |
| | | | | **II. Положительное/отрицательное приращение** | | | | | | |
| $00r_i100$ | 1 | 1/3 | 5/10 | 1. Положительное приращение регистра $r_i \leftarrow (r_i)+1$ | INR $r_i$ | + | + | – | + | + |
| $00r_i101$ | 1 | 1/3 | 5/10 | 2. Отрицательное приращение регистра $r_i \leftarrow (r_i)-1$ | DCR $r_i$ | + | + | – | + | + |
| $00r_i011$ | 1 | 1 | 5 | 3. Положительное приращение пары регистров $r_i r_{i+1} \leftarrow (r_i r_{i+1})+1$; при $r_i = 110$: приращение SP | INX $r_i$ | – | – | – | – | – |
| $00r_i011$ | 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 $r_{i-1}$ | – | – | – | – | – |
| | | | | **III. Арифметические и логические операции** | | | | | | |
| $10K_2r_i$ | 1 | 1/2 | 4/7 | 1. Над (A) и $(r_i)$: $A \leftarrow (A) \circledast (r_i)$ | | | | | | |

| Структура кода команды | Байты | Циклы | Такты | Выполняемая операция | Мнемоника | Tz | Ts | Tc | Tv | Tp |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | ❄ – операция, определяемая $K_2$ | | | | | | |
| | | | | $K_2$ \| Операция | | | | | | |
| | | | | 000 \| $A \leftarrow (A) + (r_i)$ | ADD $r_i$ | + | + | + | + | + |
| | | | | 001 \| $A \leftarrow (A) + (r_i) + (Tc)$ | ADC $r_i$ | + | + | + | + | + |
| | | | | 010 \| $A \leftarrow (A) - (r_i)$ | SUB $r_i$ | + | + | + | + | + |
| | | | | 011 \| $A \leftarrow (A) - (r_i) - (Tc)$ | SBB $r_i$ | + | + | + | + | + |
| | | | | 100 \| $A \leftarrow (A) \wedge (r_i)$ | ANA $r_i$ | + | + | 0 | 0 | + |
| | | | | 101 \| $A \leftarrow (A) \veebar (r_i)$ | XRA $r_i$ | + | + | 0 | 0 | + |
| | | | | 110 \| $A \leftarrow (A) \vee (r_i)$ | ORA $r_i$ | + | + | 0 | 0 | + |
| | | | | 111 \| $A - (r_i)$ Сравнение | CMR $r_i$ | + | + | + | + | + |
| $00r_i001$ | 1 | 3 | 10 | 2. Сложение содержимого пар регистров<br>$HL \leftarrow (HL) + (r_{i-1}\, r_i)$;<br>при $r_i = 111$: $HL \leftarrow (HL) + (SP)$ | DAD $r_i$ | – | – | + | – | – |
| $11K_2110$ | 2 | 2 | 7 | 3. Операции с непосредственной адресацией<br>$A \leftarrow (A)$ ❄$<B_2>$ | | | | | | |
| | | | | $K_2$ \| Операция | | | | | | |
| | | | | 000 \| $A \leftarrow (A) + <B_2>$ | ADI | + | + | + | + | + |
| | | | | 001 \| $A \leftarrow (A) + <B_2> + (Tc)$ | ACI | + | + | + | + | + |
| | | | | 010 \| $A \leftarrow (A) - <B_2>$ | SUI | + | + | + | + | + |
| | | | | 011 \| $A \leftarrow (A) - <B_2> - (Tc)$ | SBI | + | + | + | + | + |

| Структура кода команды | Байты | Циклы | Такты | Выполняемая операция | | Мнемоника | Tz | Ts | Tc | Tv | Tp |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 100 | $A \leftarrow (A) \wedge \langle B_2 \rangle$ | ANI | + | + | 0 | 0 | + |
| | | | | 101 | $A \leftarrow (A) \oplus \langle B_2 \rangle$ | XRI | + | + | 0 | 0 | + |
| | | | | 110 | $A \leftarrow (A) \vee \langle B_2 \rangle$ | ORI | + | + | 0 | 0 | + |
| | | | | 111 | $A - \langle B_2 \rangle$ Сравнение | CPI | + | + | + | + | + |
| | | | | **IV. Операции циклического сдвига** | | | | | | | |
| $00K_3111$ | 1 | 1 | 4 | $K_3$ | Операция | | | | | | |
| | | | | 000 | $A_{m+1} \leftarrow (A_m)$; $A_0 \leftarrow (A_7)$; $Tc \leftarrow (A_7)$ | RLC | – | – | + | – | – |
| | | | | 001 | $A_m \leftarrow (A_{m+1})$; $A_7 \leftarrow (A_0)$; $Tc \leftarrow (A_0)$ | RRC | – | – | + | – | – |
| | | | | 010 | $A_{m+1} \leftarrow (A_m)$; $A_0 \leftarrow (Tc)$; $Tc \leftarrow (A_7)$ | RAL | – | – | + | – | – |
| | | | | 011 | $A_m \leftarrow (A_{m+1})$; $A_7 \leftarrow (Tc)$; $Tc \leftarrow (A_0)$ | RAR | – | – | + | – | – |
| | | | | **V. Операции переходов** | | | | | | | |
| 11000011 | 3 | 3 | 10 | 1. Безусловный переход: $PC \leftarrow \langle B_3\ B_2 \rangle$ | | JMP | – | – | – | – | – |
| | | | | 2. Условные переходы | | | | | | | |
| $11K_4010$ | 3 | 3 | 10 | $K_4$ | Операция | | | | | | |
| | | | | 000 | Если (Tz) = 0, то $PC \leftarrow \langle B_3\ B_2 \rangle$ | JNZ | – | – | – | – | – |
| | | | | 001 | Если (Tz) = 1, то $PC \leftarrow \langle B_3\ B_2 \rangle$ | JZ | – | – | – | – | – |
| | | | | 010 | Если (Tc) = 0, то $PC \leftarrow \langle B_3\ B_2 \rangle$ | JNC | – | – | – | – | – |
| | | | | 011 | Если (Tc) = 1, то $PC \leftarrow \langle B_3\ B_2 \rangle$ | JC | – | – | – | – | – |
| | | | | 100 | Если (Tp) = 0, то $PC \leftarrow \langle B_3\ B_2 \rangle$ | JPO | – | – | – | – | – |

| Структура кода команды | Байты | Циклы | Такты | Выполняемая операция | | Мнемоника | Tz | Ts | Tc | Tv | Tp |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 101 | Если (Tp) = 1, то PC ← <$B_3$ $B_2$> | JPE | – | – | – | – | – |
| | | | | 110 | Если (Ts) = 0, то PC ← <$B_3$ $B_2$> | JP | – | – | – | – | – |
| | | | | 111 | Если (Ts) = 1, то PC ← <$B_3$ $B_2$> | JM | – | – | – | – | – |
| 11001101 | 3 | 5 | 17 | 3. Вызов подпрограммы безусловный<br>[(SP) – 1] [(SP) –2] ← (PC); SP ←(SP) – 2;<br>PC ← <$B_3$ $B_2$><br>4. Вызов подпрограммы условный | | CALL | – | – | – | – | – |
| 11$K_5$100 | 3 | 3/5 | 11/17 | $K_5$ | Операция | | | | | | |
| | | | | 000 | Если (Tz) = 0, то [(SP) – 1] [(SP) – 2] ← (PC); SP ← (SP) – 2; PC ← <$B_3$ $B_2$> | CNZ | – | – | – | – | – |
| | | | | 001 | Если (Tz) = 1, то [(SP) – 1] [(SP) – 2] ← (PC); SP ← (SP) – 2; PC ← <$B_3$ $B_2$> | CZ | – | – | – | – | – |
| | | | | 010 | Если (Tc) = 0, то [(SP) – 1] [(SP) – 2] ← (PC); SP ← (SP) – 2; PC ← <$B_3$ $B_2$> | CNC | – | – | – | – | – |
| | | | | 011 | Если (Tc) = 1, то [(SP) – 1] [(SP) – 2] ← (PC); SP ← (SP) – 2; PC ← <$B_3$ $B_2$> | CC | – | – | – | – | – |
| | | | | 100 | Если (Tp) = 0, то [(SP) – 1] [(SP) – 2] ← (PC); SP ← (SP) – 2; PC ← <$B_3$ $B_2$> | CPO | – | – | – | – | – |
| | | | | 101 | Если (Tp) = 1, то [(SP) – 1] [(SP) – 2] ← (PC); SP ← (SP) – 2; PC ← <$B_3$ $B_2$> | CPE | – | – | – | – | – |

| Структура кода команды | Байты | Циклы | Такты | Выполняемая операция | | Мнемоника | Tz | Ts | Tc | Tv | Tp |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 110 | Если (Ts) = 0, то [(SP) – 1] [(SP) – 2] ← (PC); SP ← (SP) – 2; PC ← <$B_3$ $B_2$> | CP | – | – | – | – | – |
| | | | | 111 | Если (Ts) = 1, то [(SP) – 1] [(SP) – 2] ← (PC); SP ← (SP) – 2; PC ← <$B_3$ $B_2$> | CM | | | | | |
| 11001001 | 1 | 3 | 10 | 5. Возврат из подпрограммы<br>PC ← [(SP)] [(SP)+ 1]; SP ← (SP) + 2<br>6. Условный возврат из подпрограммы | | RET | – | – | – | – | – |
| 11$K_6$000 | 1 | 1/3 | 5/11 | $K_6$ | Операция | | | | | | |
| | | | | 000 | Если (Tz) = 0, то PC ← [(SP)] [(SP)+ 1]; SP ← (SP) + 2 | RNZ | – | – | – | – | – |
| | | | | 001 | Если (Tz) = 1, то PC ← [(SP)] [(SP)+ 1]; SP ← (SP) + 2 | RZ | – | – | – | – | – |
| | | | | 010 | Если (Tc) = 0, то PC ← [(SP)] [(SP)+ 1]; SP ← (SP) + 2 | RNC | – | – | – | – | – |
| | | | | 011 | Если (Tc) = 1, то PC ← [(SP)] [(SP)+ 1]; SP ← (SP) + 2 | RC | – | – | | – | – |
| | | | | 100 | Если (Tp) = 0, то PC ← [(SP)] [(SP)+ 1]; SP ← (SP) + 2 | RPO | – | – | – | – | – |
| | | | | 101 | Если (Tp) = 1, то PC ← [(SP)] [(SP)+ 1]; SP ← (SP) + 2 | RPE | – | – | – | – | – |

| Структура кода команды | Байты | Циклы | Такты | Выполняемая операция | Мнемоника | Tz | Ts | Tc | Tv | Tp |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 110 Если (Ts) = 0, то PC ← [(SP)] [(SP)+ 1]; SP ← (SP) + 2 | RP | – | – | – | – | – |
| | | | | 111 Если (Ts) = 1, то PC ← [(SP)] [(SP)+ 1]; SP ← (SP) + 2 | RM | – | – | – | – | – |
| | | | | **VI. Операции ввода и вывода** | | | | | | |
| 11011011 | 2 | 3 | 10 | 1. Ввод данных: A ← (Шина данных) | IN | – | – | – | – | – |
| 11010011 | 2 | 3 | 10 | 2. Вывод данных: Шина данных ← (A) | OUT | – | – | – | – | – |
| | | | | **VII. Прочие операции** | | | | | | |
| 11AAA111 | 2 | 3 | 11 | 1. Рестарт [(SP) – 1] [(SP) – 2 ] ← PC; SP ← (SP) – 2; PC ← 00000000 00AAA000 | RST | – | – | – | – | – |
| 00101111 | 1 | 1 | 4 | 2. Дополнение (A): A ← $(\overline{A})$ | SMA | – | – | – | – | – |
| 00110111 | 1 | 1 | 4 | 3. Установка переноса: Tc ← 1 | STC | – | – | 1 | – | – |
| 00111111 | 1 | 1 | 4 | 4. Дополнение переноса: 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 | – | – | – | – | – |

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

| Мнемоника команды | Кодовая комбинация команды | Выполняемая операция |
|---|---|---|
| MOV C,D | 01 001 010 | C ← (D) |
| ADD D | 10 000 010 | A ← (A) + (D) |

Примеры команд с непосредственной адресацией.

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

<$B_i$> — содержимое байта $B_i$ команды.

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 отмечается 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 ← [(BC)] |
| 2) STAX B | 00 000 010 | [(BC)] ←(A) |

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

По команде LDAX B аккумулятор загружается содержимым ячейки оперативной памяти, адресом которой служит содержимое пары реги-

стров ВС (для указания именно этой пары регистров в разрядах $D_5D_4D_3$ команды приведен адрес 001 регистра С).

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

### Принцип работы микропроцессора

На рис.5.3 показана структурная схема микропроцессорной системы на МПК КР580. Генератор тактовых импульсов (ГТИ) формирует две импульсные последовательности $\Phi_1$ и $\Phi_2$, необходимые для тактирования работы микропроцессора (рис. 5.4). Импульсы двух последовательностей не должны перекрываться во времени и должны иметь амплитуду 12 В. ПЗУ может быть использовано для хранения программы, ОЗУ — для хранения данных.

Общий принцип функционирования микропроцессорной системы 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 исполнения данной команды на шину адреса поступает адрес следующей команды и т.д. При исполнении команды могут потребоваться дополнительные обращения к памяти для вызова в микропроцессор дополнительных байтов команды (в случае двух-, трех- байтовых команд), операндов или для записи в память числа, выдаваемого из микропроцессора.

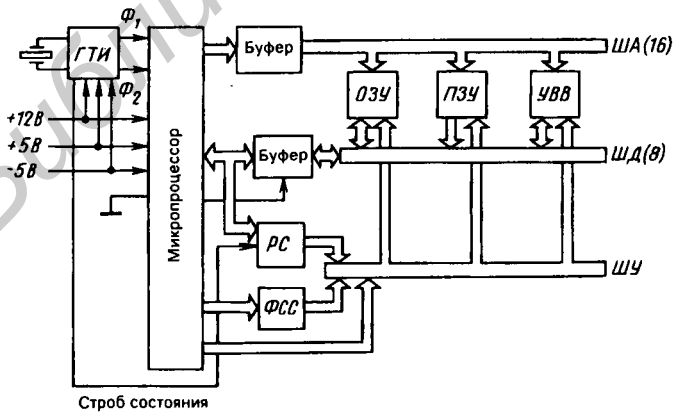



Рис. 5.3

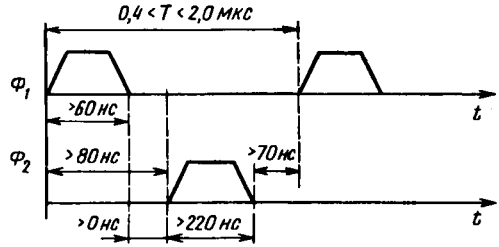



Рис. 5.4

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 к устройству ввода или вывода (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 несколько тактов, обозначаемых $T_1$, $T_2$, $T_3$, $T_4$, $T_5$. Циклы могут содержать три ($T_1$, ..., $T_3$), четыре ($T_1$, ..., $T_4$) такта 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. 5.5 показана временная диаграмма цикла из пяти тактов.

Отсчет тактов производится от положительных фронтов импульсов $Ф_1$. Рассмотрим цикл $M_1$. В такте $T_1$ содержимое счетчика команд

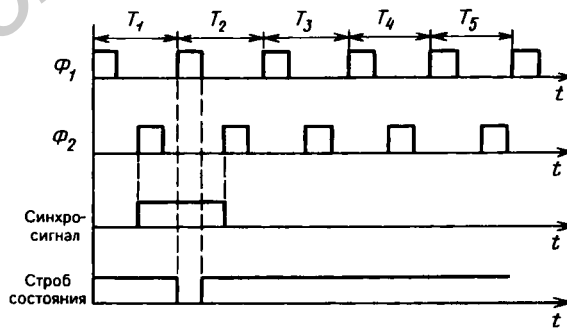



Рис. 5.5

выдается на шину адреса, адрес принимается памятью, где начинается процесс чтения байта команды из указанной ячейки. В такте $T_2$ проверяется наличие сигнала (уровня *лог.1*) на входе *Готовность* (см. рис.5.1). 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 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 $\Phi_1$ в такте $T_1$ до момента положительного фронта импульса $\Phi_2$ в такте $T_2$ микропроцессор выдает на выход *Синхронизация* (рис. 5.1) уровень *лог.1* и на шину данных — *информацию о состоянии*. ГТИ (см. рис. 5.3) формирует *строб состояния*, которым осуществляется прием информации о состоянии микропроцессора с шины данных в *регистр состояния* (временное положение строба состояния показано на рис. 5.5). В табл. 5.3 показаны назначения сигналов в разрядах кода состояния микропроцессора. В табл. 5.4 приведено соответствие этих сигналов отдельным видам циклов.

*Таблица 5.3*

| Разряд кода состояния | Назначение сигнала в разряде |
|---|---|
| $D_0$ | *Подтверждение прерывания*: используется для стробирования команды RST в микропроцессор из устройства, запрашивающего прерывание |
| $D_1$ | Запись-вывод: уровень *лог.0* свидетельствует о том, что в данном цикле будет происходить *запись* (выдача информации из микропроцессора в оперативную память) или *вывод* (передача информации из микропроцессора в УВВ); уровень *лог.1* означает, что происходит *чтение* (прием информации из оперативной памяти) или *ввод*, (прием из УВВ) |
| $D_2$ | На адресной шине установлено *содержимое указателя стек*а |
| $D_3$ | *Подтверждение останова*: микропроцессор в состоянии останова |
| $D_4$ | На *адресной шине* установлен *номер внешнего устройства* и осуществляется *вывод* содержимого аккумулятора на устройство вывода |
| $D_5$ | Микропроцессор *принимает первый байт команды* |
| $D_6$ | На *адресной шине* установлен *номер устройства ввода* и осуществляется ввод информации из устройства ввода в аккумулятор микропроцессора |
| $D_7$ | В данном цикле производится *чтение* из памяти в микропроцессор |

*Таблица 5.4*

| 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 из УВВ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Подтверждение прерывания | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| Подтверждение останова | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| Подтверждение прерывания при останове | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |

## Система команд микропроцессора

Система команд микропроцессора приведена в табл. 5.2. Команды разбиты на семь групп. Группа может содержать несколько видов операций. Каждый вид 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 в качестве адреса регистра указана комбинация 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 признаки 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 0;

1 означает установку триггера в состояние 1.

Отметим следующие особенности формирования признаков:

команды пересылки и переходов не изменяют состояния триггеров признаков;

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

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 0;

команды сложения содержимого пар регистров используют только признак переноса *C*.

## Стек

*Стек* — память с определенной (упрощенной) формой адресации. В микропроцессорной системе на МПК КР580 стек организуется следующим образом. В 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 (рис. 5.6,*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 (адресу, хранящемуся в SP); при записи в стек вводимое в стек число помещается в ячейку с адресом, на единицу меньшим содержимого 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 A. При вводе нового слова его байты должны быть помещены в пару соседних со входом в стек ячеек, имеющих адреса A – 1 и A – 2. Таким образом,

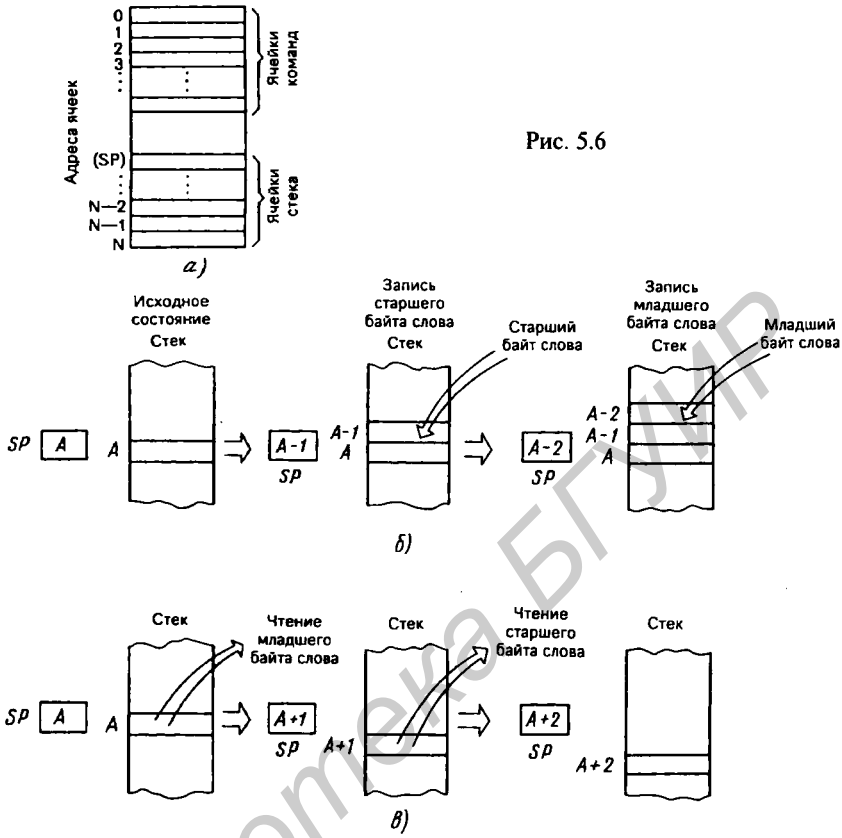



Рис. 5.6

ввод в стек сводится к следующей последовательности действий: содержимое SP уменьшается на единицу и по образующемуся адресу помещается старший байт вводимого двухбайтового слова; затем содержимое SP вновь уменьшается на единицу и по образующемуся адресу помещается младший байт вводимого слова (рис. 5.6.*б*). Мы видим, что SP каждый раз указывает адрес ячейки, являющейся входом 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 адресу, хранящемуся в SP, и увеличение содержимого SP на единицу (рис. 5.6,*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 некоторые команды операций со стеком.

Установка в SP некоторого начального значения производится по команде пересылки SPHL (11 111 001), по которой 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: SP ← (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 $\Phi_1$ и $\Phi_2$ 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 при отрицательном фронте импульса $\Phi_2$ такта $T_2$ принимается в *триггер захвата* управляющего устройства. Управляю-

щее устройство заканчивает выполнение текущего цикла, переходит в состояние захвата и подтверждает это выдачей сигнала на выходе *Подтверждение захвата*. Сигнал на выходе *Подтверждение захвата* выдается при положительном фронте импульса $\Phi_1$ в такте $T_3$, если текущий цикл не является циклом записи; в противном случае этот сигнал выдается при положительном фронте импульса $\Phi_1$ 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 основной 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, выдаваемом 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 PC записывается адрес первой команды прерывающей программы. Этот адрес задается следующим образом. Команда RST имеет структуру 11 AAA 111, и в счетчик команд заносится значение 00 000 000 00 AAA 000, которое и служит адресом первой

команды прерывающей программы. Задавая определенную трехразрядную 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 команды 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.

## 5.3. ПРИЕМЫ ПРОГРАММИРОВАНИЯ МИКРОПРОЦЕССОРА НА ЯЗЫКЕ КОДОВЫХ КОМБИНАЦИЙ

### Программирование последовательных участков алгоритма

Рассмотрим программирование участков алгоритма, не содержащих разветвлений.

*Пример 5.1.* Требуется принять из ОЗУ два числа, хранящихся в соседних ячейках с адресами $0276_{16}$ и $0277_{16}$ и, вычислив сумму чисел, поместить ее в ОЗУ на место второго из суммируемых чисел.

На рис. 5.7 приведена схема алгоритма решения данной задачи микропроцессором серии КР580. Рассмотрим операции, выполняемые в каждом из блоков схемы алгоритма.

Блок 1: в пару регистров HL заносится адрес первого числа.

Блок 2: в аккумулятор принимается содержимое ячейки ОЗУ (первое число), адресом которой служит содержимое пары регистров HL; эта операция выполняется командой *пересылки "регистр — регистр"* (мнемоническое обозначение команды MOV A, M).

Блок 3: в паре регистров HL формируется адрес второго числа; эта операция выполняется командой *приращения пары регистров* (мнемоника команды INX H).

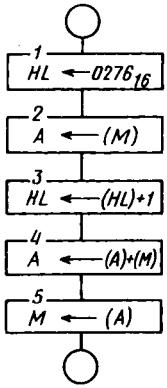



Рис. 5.7

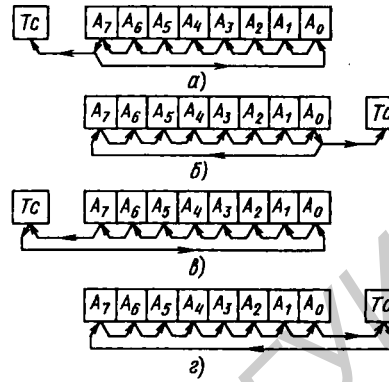



Рис. 5.8

Блок 4: вычисляется сумма содержимого аккумулятора (А) и содержимого ячейки ОЗУ (М), адресом которой служит содержимое пары регистров HL; операция выполняется командой *суммирования* ADD M.

Блок 5: полученная в аккумуляторе сумма пересылается в память; выполняющая эту операцию команда имеет мнемонику MOV M, A.

В табл. 5.5 приведена программа рассматриваемой задачи с представлением команд в кодовых комбинациях.

*Таблица 5.5*

| Адрес команды в ОП (в 16-ричной системе) | Команда | | | Пояснения |
|---|---|---|---|---|
| | Кодовая комбинация | Число байтов | Число тактов | |
| 0050 | 00101110 | 3 | 10 | Блок 1: HL ← $0276_{16}$ |
| 0051 | 01110110 | | | |
| 0052 | 00000010 | | | |
| 0053 | 01111110 | 1 | 7 | Блок 2: A ← (M) |
| 0054 | 00100011 | 1 | 5 | Блок 3: HL ← (HL) + 1 |
| 0055 | 10000110 | 1 | 7 | Блок 4: A ← (A)+ (M) |
| 0056 | 01110111 | 1 | 7 | Блок 5: M ← (A) |

Команды программы при отсутствии условных и безусловных переходов (так называемой *последовательной программы*) 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 пяти команд, $N_т = 10 + 7 + 5 + 7 + 7 = 36$, и при длительности тактового периода $T = 0{,}5$ мкс общее время их исполнения $t_{исп} = N_т \cdot T = 36 \cdot 0{,}5 = 18$ мкс.

Рассмотрим пример программирования с использованием содержимого триггера регистра признаков.

*Пример 5.2.* Требуется выполнить операцию арифметического сдвига вправо содержимого регистра D.

В системе команд микропроцессора предусмотрены четыре команды операций сдвига содержимого аккумулятора — по две команды операций сдвига влево и вправо. Каждая операция сдвига носит циклический характер, т.е. осуществляется в замкнутом кольце, в которое кроме аккумулятора может входить триггер переноса Тс регистра признаков ("сдвиг через Тс", рис. 5.8,*в* и *г*) либо этот триггер может быть вне кольца ("сдвиг без Тс", рис.5.8,*а* и *б*). В обоих случаях выдвигаемое из аккумулятора значение разряда числа (в зависимости от направления сдвига старшего или младшего разряда) поступает в триггер Тс.

Вернемся к решаемому примеру. Особенность арифметического сдвига вправо состоит в том, что при сдвиге содержимое старшего (знакового) разряда регистра сохраняется неизменным (рис. 5.9,*а*). Таким образом, при выполнении этой операции необходимо предварительно запомнить содержимое старшего разряда и затем, сдвинув вправо содержимое регистра, вписать в его старший разряд цифру, которую ранее запомнили.

Эти действия выполняются во фрагменте программы, схема алгоритма которой представлена на рис. 5.9,*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. 5.6. В результате выполнения операции "сдвиг влево без Тс" (блок 2) знак числа (в табл. 5.6 знак выделен полужирным шрифтом) передается в младший разряд аккумулятора и в триггер переноса Тс. Затем дважды выполняется операция "сдвиг вправо через Тс" (блоки 3 и 4). В результате выполнения первой операции сдвига в аккумуляторе восстанавливается исходное число, а в триггере Тс оказывается продублированным знак числа. После выполнения второй операции сдвига вправо в аккумуляторе оказывается число, являющееся результатом выполнения арифметического сдвига вправо.

*Таблица 5.*

| (Тс) | (А) | Выполняемая операция |
|---|---|---|
| × | 1 0 1 1 0 1 1 0 | Исходное состояние |
| 1 ← | 0 1 1 0 1 1 0 1 | Сдвиг влево без Тс |
| 1 → | 1 0 1 1 0 1 1 0 | Сдвиг вправо через Тс |
| 0 → | 1 1 0 1 1 0 1 1 | Сдвиг вправо через Тс |

В табл. 5.7 приведена соответствующая схеме алгоритма на рис. 5.9 программа с представлением команд в кодовых комбинациях.

*Таблица 5.7*

| Адрес команды в ОП ( в 16-ричной системе) | Команда в кодовой комбинации | Пояснения |
|---|---|---|
| 0071 | 01 111 010 | Блок 1: А ← (D) |
| 0072 | 00 000 111 | Блок 2: Сдвиг влево без Тс |
| 0073 | 00 011 111 | Блок 3: Сдвиг вправо через Тс |
| 0074 | 00 011 111 | Блок 4: Сдвиг вправо через Тс |

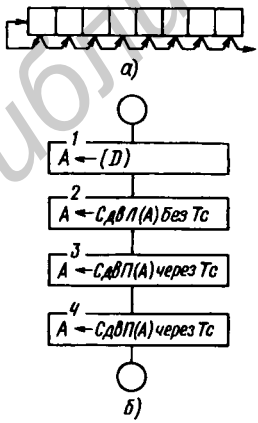



Рис. 5.9

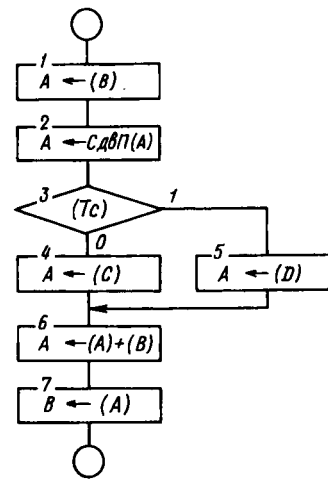



Рис. 5.10

### Программирование разветвлений

*Пример 5.3.* Требуется проанализировать содержимое младшего разряда числа, хранящегося в регистре B микропроцессора. Если оно равно нулю, то к содержимому регистра B следует прибавить содержимое регистра C; если оно равно единице, то к содержимому регистра B следует прибавить содержимое регистра D.

На рис. 5.10 показана схема алгоритма решения этой задачи. Здесь блоки 1 и 2 осуществляют передачу содержимого младшего разряда регистра B в триггер Tc регистра признаков. Блок 3 реализует разветвление по содержимому триггера Tc: в зависимости от значения содержимого этого триггера в аккумулятор передается либо содержимое регистра C (блок 4), либо содержимое регистра D (блок 5). Блок 6 осуществляет сложение. Блок 7 полученную в аккумуляторе сумму пересылает в регистр B.

В табл. 5.8 показано размещение команд в ячейках ОП, реализующих рассмотренный алгоритм.

*Таблица 5.8*

| Адрес ячейки УП | Содержимое ячейки (микрокоманда) |
|---|---|
| 0000 | МК1 (МК ОУ: $Y_1$ ; МК БМУ:БП) |
| 0001 | МК2 (МК ОУ: - ; МК БМУ:УсП по $x_2$) |
| $x_2 = 0 \rightarrow$ 0010 | МК3 (МК ОУ: - ; МК БМУ:УсП по $x_1$) |
| $x_2 = 1 \rightarrow$ 0011 | МК6 (Продолжение) |
| $x_1 = 0 \rightarrow$ 0100 | МК5 (МК ОУ: $Y_3$ ; МК БМУ:БП) |
| $x_1 = 1 \rightarrow$ 0101 | МК4 (МК ОУ: $Y_2$ ; МК БМУ:БП) |

Пусть команда, реализующая операцию блока 1, помещается в ячейку ОП с адресом 0167. При выполнении программы следующая команда будет считываться из соседней ячейки с адресом 0168, и в этой ячейке должна храниться команда, реализующая операцию блока 2. Затем из трех очередных ячеек с адресами 0169, 016A, 016B должна считываться трехбайтовая команда условного перехода (УсП) по (Tc) = 1 (блок 3): при (Tc) = 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 (Tc) = 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 команда блока 4 либо команда блока 5, должна быть выполнена операция, предусматриваемая блоком 6. После выполнения команды, считываемой из ячейки 0170, очередная команда блока 6 считывается из соседней ячейки с адресом 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 с адресами 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.5.9 приведена программа данной задачи.

*Таблица 5.9*

| Адрес команды в ОП ( в 16-ричной системе) | Команда в кодовой комбинации | Число байтов | Пояснения |
|---|---|---|---|
| ... | ... | ... | ... |
| 0167 | 01 111 000 | 1 | Блок 1: A ← (B) |
| 0168 | 00 001 111 | 1 | Блок 2: A ← СдвП(A) |
| 0169 | 11 011 010 | 3 | Блок 3: УсП при (Тс) = 1 |
| 016A | 01 110 000 | | к ячейке $(B_3B_2) = 0170_{16}$ |
| 016B | 00 000 001 | | |
| 016C | 01 111 001 | 1 | Блок 4: A ← (C) |
| 016D | 11 000 011 | 3 | Безусловный переход |
| 016E | 01 110 001 | | к ячейке $(B_3B_2) = 0171_{16}$ |
| 016F | 00 000 001 | | |
| 0170 | 01 111 010 | 1 | Блок 5: A ← (D) |
| 0171 | 10 000 000 | 1 | Блок 6: A ← (A) + (B) |
| 0172 | 01 000 111 | 1 | Блок 7: B ← (A) |
| ... | ... | ... | ... |

**Программирование циклических вычислительных процессов**

Рассмотрим выполнение операции кодового умножения двух восьмиразрядных чисел без знака. Пусть множимое хранится в паре реги-

стров DE, где оно занимает младший регистр E, старший регистр D пары регистров установлен в нуль. Будем считать, что множитель хранится в аккумуляторе A. 16-разрядное 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 так далее, 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. 5.11 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 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 осу-

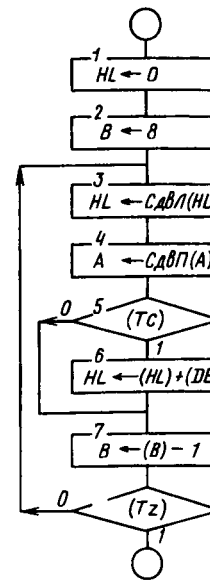
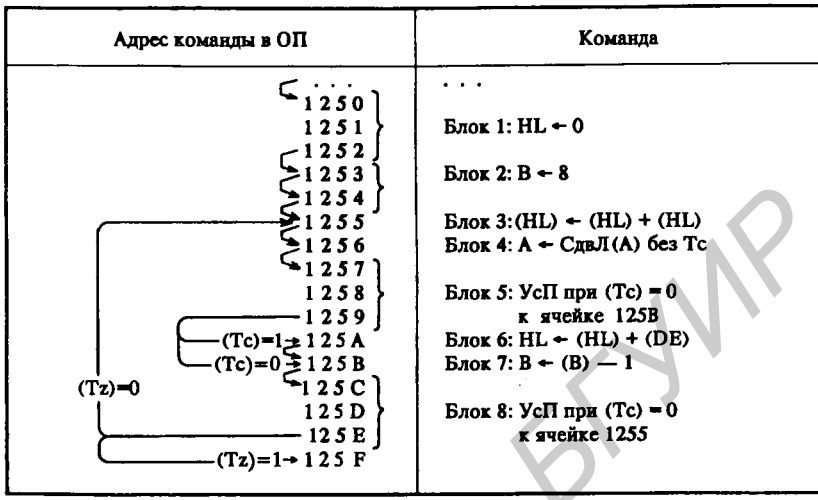



Рис. 5.11

*Таблица 5.10*




ществляется 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 1, происходит выход из цикла и переход к очередной команде. Если содержимое регистра B не равно нулю, то в триггере Tz устанавливается 0 и команда условного перехода производит переход к команде блока 3, вызывая очередное повторение выполнения тела цикла.

В табл. 5.10 показано размещение команд в ячейках ОП. В табл. 5.11 приведена программа рассматриваемой операции умножения.

В цикле выполняются команды блоков 3 — 8. Определим количество тактов $N_{т1}$, требуемое для однократного прохождения цикла алгоритма. При этом будем полагать, что во всех разрядах множителя 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.

*Таблица 5.11*

| Адрес команды в ОП (в 16-ричной системе) | Команда кодовой комбинации | Число байтов | Число тактов | Пояснения |
|---|---|---|---|---|
| 1250 | 00 100 001 | 3 | 10 | Блок 1: HL ← ($B_3B_2$) |
| 1251 | 00 000 000 | | | ($B_3$) = 0, ($B_2$) = 0 |
| 1252 | 00 000 000 | | | |
| 1253 | 00 000 110 | 2 | 7 | Блок 2: B ← ($B_2$) |
| 1254 | 00 001 000 | | | ($B_2$) = 8 |
| 1255 | 00 101 001 | 1 | 10 | Блок 3: HL ← (HL) + (HL) |
| 1256 | 00 000 111 | 1 | 4 | Блок 4: A ← СдвЛ (A) |
| 1257 | 11 010 010 | 3 | 10 | Блок 5: УсП при (Tc) = 0 |
| 1258 | 01 011 011 | | | к ячейке 125B |
| 1259 | 00 010 010 | | | ($B_3$) = $12_{16}$, ($B_2$) = $5B_{16}$ |
| 125A | 00 011 001 | 1 | 10 | Блок 6: HL ← (HL) + (DE) |
| 125B | 00 000 101 | 1 | 5 | Блок 7: B ← (B) – 1 |
| 125C | 11 000 010 | 3 | 10 | Блок 8: УсП при (Tz) = 0 |
| 125D | 01 010 101 | | | к ячейке 1255 |
| 125E | 00 010 010 | | | |
| 125F | . . . | | | |

Ранее было показано, что в процессоре, в котором используется специализированное операционное устройство и управляющее устройство, построенное на принципе схемной логики, требовалось два такта для однократного прохождения цикла алгоритма. Следовательно, реализация рассматриваемой операции умножения в микропроцессоре потребовала в 49/2 = 24,5 раза большего числа тактов. Такой проигрыш в быстродействии при использовании микропроцессора КР580ВМ80А может оказаться еще большим, если учесть, что в случае применения в микропроцессорной системе оперативной памяти с низким быстродействием в цикле работы микропроцессора появятся "пустые" такты $T_2$, связанные с ожиданием появления сигнала на входе *Готовность*.

## 5.4. ПРОГРАММИРОВАНИЕ МИКРОПРОЦЕССОРА НА ЯЗЫКЕ АССЕМБЛЕРА

### Языки программирования

До сих пор, записывая команды, мы применяли язык кодовых комбинаций, единственно понятный микропроцессору. 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 системе счисления не приводит к существенному облегчению записи программы), во-вторых, со сложностью поиска ошибок в составленной программе из-за того, что записанная с помощью кодовых комбинаций программа оказывается трудно читаемой, плохо обозримой, и, в-третьих, с трудностью внесения изменений в составленную программу.

Рассмотрим, в чем состоят последние трудности. 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. 5.12 $A_{k+1}, \ldots, A_{k+m}$ — 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 можно воспользоваться следующими приемами. Если $m \geq 3$, то в первые три ячейки разрыва следует поместить трехбайтовую коман-

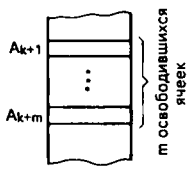



Рис. 5.12

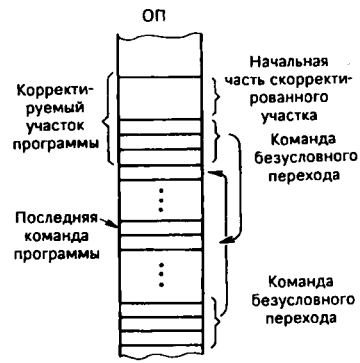



Рис. 5.13

ду безусловного перехода к ячейке $A_{k+m+1}$ — первой после разрыва. Если число ячеек в разрыве меньше 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 помещают начальную часть скорректированного участка, заканчивая ее трехбайтовой командой безусловного перехода к некоторой свободной ячейке, например следующей за ячейкой, которую занимает последняя команда программы (рис. 5.13). Начиная с этой ячейки, производится размещение команд конечной части скорректированного участка программы. Завершить эту часть участка программы необходимо командой безусловного перехода.

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 составить программу заново, не пользуясь описанными выше приемами.

Наряду с указанными недостатками язык кодовых комбинаций имеет и достоинства. Программа на этом языке оказывается наиболее эффективной, она занимает минимальный объем памяти и быстрее исполняется. Уменьшение памяти (часто выполняемой в виде ПЗУ) снижает затраты на этот наиболее дорогостоящий узел микропроцессорной системы, а уменьшение времени позволяет решать более сложные задачи. Кроме того, записанная на бумаге программа после ее набивки на перфоленту может быть непосредственно введена в память микропроцессорной системы.

Трудности программирования уменьшаются при использовании *языка ассемблера*. В этом языке вместо кодовых комбинаций применяется мнемоническая форма записи операций, выполняемых в микропроцессоре. 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, облегчает поиск в ней ошибок, обеспечивает лучший обзор программы и простоту внесения исправлений в программу (без 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 микропроцессорной системы. Этот перевод осуществляется на ЭВМ с помощью программы трансляции, называемой *ассемблером*. 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. Применение языка макроассемблера сокращает запись программы (в среднем на 5...20 %) и тем самым улучшает ее обозримость. Каждая серия МПК имеет свой индивидуальный язык макроассемблера (как и язык ассемблера), т. е. программа, составленная на этом языке для 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 большую 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 программы независимо от серии микропроцессорного комплекта, используемого для построения микропроцессорного устройства (к таким языкам относятся Бейсик, Фортран, Паскаль и др.). Вторые пригодны для определенных серий МПК. Для программирования устройств, построенных с использованием комплекта серии КР580, разработан язык высокого уровня PLM-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); 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 программ, 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. 5.12 показана запись на языке ассемблера той же программы, которая на языке кодовых комбинаций приведена в табл. 5.9.

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. 5.12 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 переходов.

*Таблица 5.12*

| Метка | Код | Операнд | Комментарий |
|---|---|---|---|
| | MOV | A, B | ; Блок 1: A ← (B) |
| | RRC | | ; Блок 2: A ← СдвП(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) |

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 без-

условных 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 шестнадцатеричного 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 заканчивается 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 HL 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, OAH + 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 большим 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 относительно редко и в данном пособии не рассматриваются.

**Поле комментария**. Начинается символом ";" (точка с запятой). Оно служит для записи любых пояснений 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. 5.13 показана программа умножения, представляющая собой перевод на язык ассемблера программы, приведенной в кодовых комбинациях в табл. 5.11.

**Сложение многобайтовых чисел.** Пусть требуется сложить два четырехбайтовых 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 первого слагаемого.

Принцип сложения таких многобайтовых 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; возникший в процессе сум-

мирования перенос из старшего разряда запоминается в триггере Тс регистра признаков. Затем в парах регистров BC и HL формируются адреса вторых байтов слагаемых, которые затем вызываются в микропроцессор и суммируются с хранящимся в триггере Тс переносом, возникшим при сложении первых байтов, и т. д. Выполнение операции завершается после четырехкратного повторения указанных действий.

*Таблица 5.13*

| 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 (HL) |
| | 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 |

На рис. 5.14 приведена схема алгоритма и в табл. 5.14 — программа на языке ассемблера.

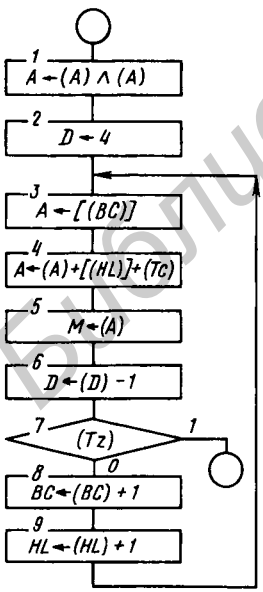



Рис. 5.14

**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 в цифровую форму запоминаются в оперативной памяти.

Работа устройства, структурная схема которого представлена на рис. 5.15, происходит в следующей последовательности. Адрес очередного канала указывается в трех младших разрядах данных, выдаваемых из микропроцессора на шину данных. С шины данных адрес принимается устройством вывода УВ1, откуда он поступает на адресные входы коммутатора. Сигнал выбранного канала передается на вход АЦП. После окончания преобразования АЦП выдает сигнал готовности $Г = 1$, который устройством ввода УВв2 передается на шину данных, откуда принимается микропроцессором.

После этого микропроцессор через УВв1 принимает значение выборки сигнала, представленное в цифровой форме. Принятые данные микропроцессор передает в ОП.

*Таблица 5.14*

| Метка | Код | Операнд | Комментарий |
|---|---|---|---|
| | ANA | A | ; Установка Тс ← 0 |
| | MVI | D, 4 | ; Подготовка счетчика |
| CYCLE: | LDAX | B | ; Блок 3 |
| | ADC | M | ; Блок 4 |
| | MOV | M, A | ; Блок 5 |
| | DCR | D | ; Блок 6 |
| | JZ | K1 | ; Блок 7 |
| | INR | B | ; Блок 8 |
| | INX | H | ; Блок 9 |
| | JMP | CYCLE | ; Безусловный переход |
| K1: | ... | ... | |

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

Номер очередного канала будем формировать 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. После опроса последнего канала очередное прибавление едини-

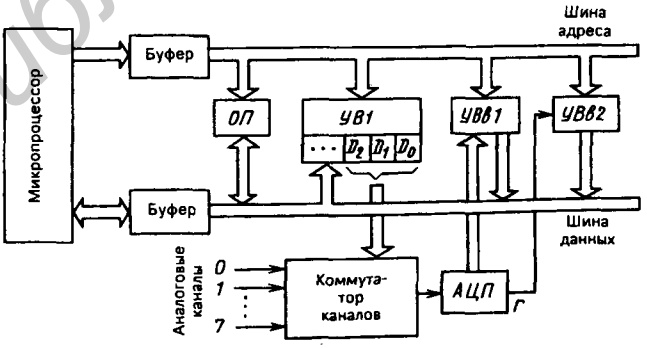



Рис. 5.15

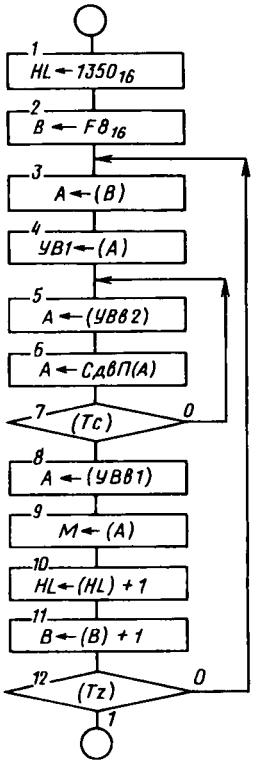



Рис. 5.16

цы 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 УВв2. 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 УВв2, и так до тех пор, пока не будет принято значение $\Gamma = 1$.

После этого производится прием данных из УВв1. 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. 5.16 приведена схема алгоритма функционирования устройства.

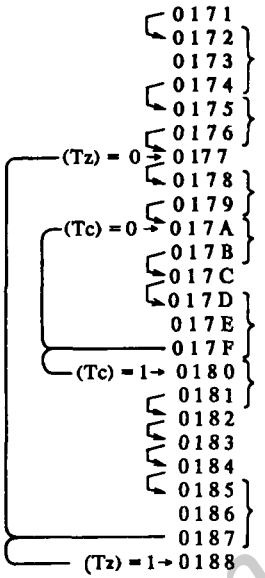
В табл.5.15 приведена программа.

В табл. 5.16 показано размещение программы в ОП.

*Таблица 5.15*

| Метка | Код | Операнд | Комментарий |
|---|---|---|---|
| | LXI | H, 1350H | ; Блок 1 |
| | MVI | B, 0F8H | ; Блок 2 |
| CYCLE: | MOV | A, B | ; Блок 3 |
| | OUT | 1 | ; Блок 4 |
| GOTOWN: | IN | 2 | ; Блок 5 |
| | RRC | | ; Блок 6 |
| | JNC | GOTOWN | ; Блок 7 |
| | IN | 1 | ; Блок 8 |
| | MOV | M, A | ; Блок 9 |
| | INX | H | ; Блок 10 |
| | INR | B | ; Блок 11 |
| | JNZ | CYCLE | ; Блок 12 |
| | ... | ... | ... |

*Таблица 5.16*

| Адрес команды в ОП | Команда |
|---|---|
|  | ...<br><br>Блок 1: HL ← $1350_{16}$<br><br>Блок 2: B ←$F8_{16}$<br><br>Блок 3: A ← (B)<br>Блок 4: Вывод в УВ1<br><br>Блок 5: Вывод из УВв2<br><br>Блок 6: A ← СдвП (A)<br><br>Блок 7: УсП при (Tc) = 0<br>к ячейке 017A<br>Блок 8: Ввод из УВв1<br><br>Блок 9: M ← (A)<br>Блок 10: HL ← ( HL) + 1<br>Блок 11: B ← (B) + 1<br><br>Блок 12:УсП при (Tz) = 0<br>к ячейке 0177<br>... |



### *Упражнения*

1. Составьте программу, выполняющую преобразование хранимых в регистрах B и D алгебраических чисел (чисел со знаком) из прямого кода в дополнительный.

2. Составьте программу, выполняющую прием из УВв2 шестнадцати восьмиразрядных алгебраических чисел, которые представлены в прямом коде, и выдачу в УВ3 их суммы. Определите число тактов, в течение которых решается задача.

3. Составьте программу, выполняющую прием из УВв12 шестнадцати восьмиразрядных чисел без знака (положительных чисел, не содержащих знакового разряда) и выдайте в оперативную память по адресу $007A_{16}$ максимальное из этих чисел.

## 5.5. УЗЛЫ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

### Генератор тактовых импульсов КР580ГФ24

На рис. 5.17 показано подключение микросхемы КР580ГФ24 к микропроцессору. Основное назначение микросхемы — формирование двух последовательностей тактовых импульсов (рис. 5.18) $\Phi_1$ и $\Phi_2$. Кроме того, микросхема выдает последовательность импульсов с уров-

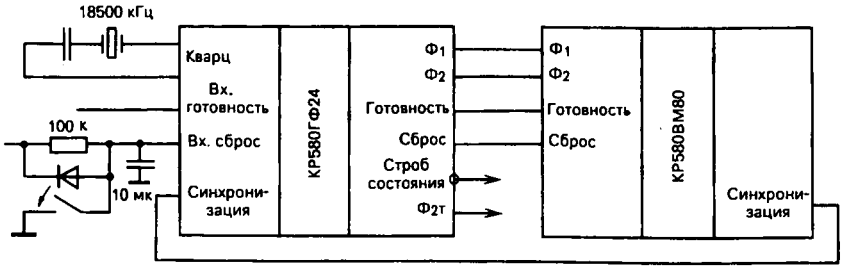



Рис. 5.17

нями, например, согласованными с уровнями схем ТТЛ ($Ф_{2T}$), формирует сигналы *Сброс, Готовность* и *Строб состояния.*

**Использование сигнала *Сброс***. Этот сигнал производит сброс в нуль счетчика команд РС в микропроцессоре. И если в ячейке ОП с нулевым значением адреса хранится первая команда программы, то с этой команды начинается исполнение программы. Таким образом, сигнал *Сброс* производит запуск микропроцессора. Сигнал *Вх. сброс*, под действием которого в микросхеме КР580ГФ24 формируется сигнал *Сброс*, может подаваться одновременно с включением источников питания либо с пульта управления (нажатием соответствующей кнопки). В момент включения источников питания (см. рис. 5.18) конденсатор 10 мкФ разряжен, напряжение на входе *Вх. сброс* равно нулю. При этом на выходе микросхемы формируется сигнал *Сброс*, осуществляющий сброс микропроцессора. Далее током через резистор 100 кОм 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. При этом конденсатор разряжается и на выходе микросхемы возникает сигнал *Сброс*. После размыкания ключа конден-

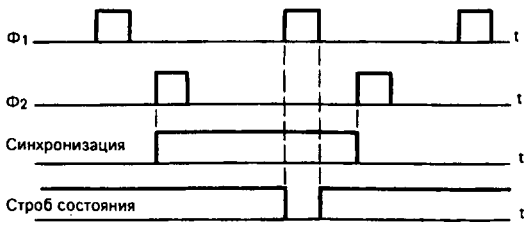



Рис. 5.18

сатор начинает заряжаться, в некоторый момент снимается сигнал *Сброс*, микропроцессор начинает выполнять программу.

### Буферы

Информация, выдаваемая микропроцессором на шины адреса и данных, может предназначаться большому числу различных устройств, подключенных к этим шинам (ОЗУ, ПЗУ, устройства ввода и вывода).

Однако выходы микросхемы КР580ВМ80А допускают потребление подключенными к ним устройствами относительно небольшого тока. Значение тока через эти выходы при высоком уровне напряжения (уровне *лог.1*) $I^1_{вых} \le 0{,}1$ мА, при низком уровне напряжения (уровне лог.0) $I^0_{вых} \le 1{,}6$ мА. При такой нагрузочной способности к выходам микропроцессора может быть подключено 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 тепло, выделяемое каждым транзистором, должно быть малым. Следовательно, малыми должны быть токи через транзисторы. Для увеличения нагрузочной способности выходов потребовалось бы использование 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.** На рис. 5.19 показана логическая схема формирователя КР580ВА86, осуществляющего передачу 8-разрядных данных. На рисунке подробно изображена схема лишь нулевого разряда, схемы остальных разрядов аналогичны. В цепи передачи включены два повторителя, имеющие три состояния. При этом если один из повторителей находится в включенном состоянии, то другой — в выключенном (третьем). Так, если повторитель 1 находится в включенном состоянии, то повторитель 2 оказывается в выключенном состоянии и передача (в нулевом разряде) осуществляется через повторитель 1 в направлении от вывода $A_0$ к выводу $B_0$. 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 $B_0$ к выводу $A_0$, 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 сиг-

нала Т на выходах элементов ИЛИ-НЕ устанавливается низкий уровень *лог.0*, во всех разрядах оба повторителя оказываются в выключенном состоянии 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. 5.17.

Выходы В (при передаче в направлении от А к В) имеют бóльшую нагрузочную способность, чем выходы А (когда происходит передача в направлении от В к А). К выходам В допускается подключение нагрузки, потребляющей ток $I^0_{вых} \le 32$ мА, $I^1_{вых} \le 5$ мА. Для выходов А эти токи $I^0_{вых} \le 10$ мА, $I^1_{вых} \le 1$ мА. Очевидно, шинный формирователь должен включаться стороной А к выводам микропроцессора, стороной В — к системным шинам адреса и данных.

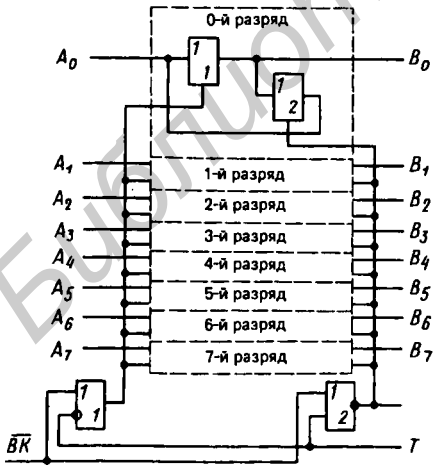



Рис. 5.19

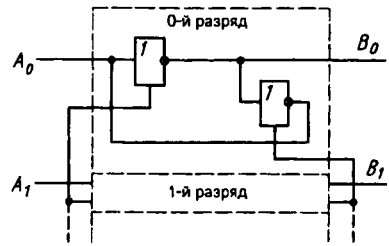



Рис. 5.20

*Таблица 5.17*

| 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 |

На рис. 5.20 приведена схема шинного формирователя КР580ВА87. Ее отличие от схемы 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.

### Формирование управляющих сигналов микропроцессорной системы

Шины данных и адреса в микропроцессорной системе являются общими для многих подключенных к ним узлов, которые либо принимают с шин, либо выдают в эти шины информацию. При таком обобществлении шин возникает необходимость в согласовании работы узлов: при выдаче информации — обеспечение ее поступления с шины данных в соответствующий узел микропроцессорной системы, при приеме информации — ее поступление в шину данных из какого-либо узла. Эти действия требуют выработки управляющих сигналов, называемых *системными управляющими сигналами*. К числу таких сигналов относятся: для управления микросхемами памяти — сигналы записи (ЗпП) и чтения (ЧтП) памяти, для управления устройствами ввода и вывода — сигналы ввода (Вв) и вывода (Выв). Для формирования этих сигналов используется микросхема — системный контроллер КР580ВК28 (КР580ВК38).

На рис. 5.21 показана структура микросхемы системного контроллера. В микросхеме предусмотрен двунаправленный шинный формирователь, выполняющий функции двунаправленного буфера, включаемого между выводами шины данных микропроцессора и шиной данных системы. Выдаваемая из микропроцессора в начале цикла информация о состоянии микропроцессора поступает на вход регистра состояния и при появлении сигнала $\overline{Строб\ состояния}$ фиксируется в регистре, где она хранится до наступления следующего цикла (до момента поступления очередного сигнала $\overline{Строб\ состояния}$). Контрольно-декодирующая матрица использует содержимое регистра состояния и управляющие сигналы с выхода микропроцессора *Прием, Запись, Подтверждение*

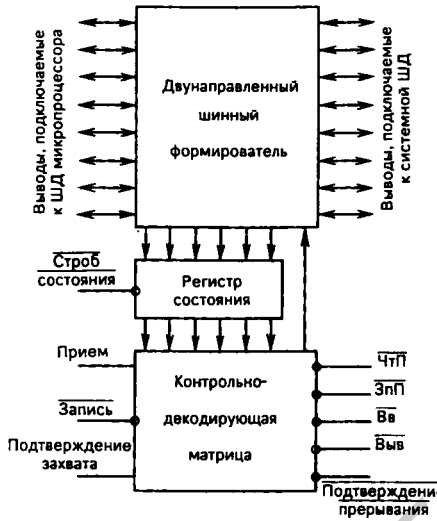



Рис. 5.21

захвата, формируя на выходах контроллера системные управляющие сигналы *ЧтП*, *ЗпП*, *Вв*, *Выв*, *Подтверждение прерывания*.

### Интерфейсы ввода-вывода

В процессе функционирования МПС возникает необходимость приема в него данных от различных устройств ввода. Принятые данные подвергаются обработке. Полученные в результате обработки данные выводятся из МПС и передаются в различные устройства вывода. В качестве таких устройств ввода и вывода, называемых периферийными (ПУ), могут использоваться телетайпы, дисплеи, аналого-цифровые и цифроаналоговые 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 МПС и ПУ может либо быть *программно-управляемым*, либо осуществляться способом 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 ОП в ПУ байты данных принимаются из ОП в аккумулятор микропроцессора, затем из аккумулятора они выдаются на шину данных, откуда принимаются в соответствующее ПУ.

Большая 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 состояние Захват) 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 Вв и Выв ПУ готово к выдаче на шину данных запрашиваемого микропроцессором байта или к приему с шины данных байта, выданного на эту шину микропроцессором.

**Асинхронная передача.** При асинхронной передаче, прежде чем производить обмен данными, микропроцессор выясняет готовность ПУ к такому обмену. Приведенная на рис.5.22 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.

**Передача данных с прерыванием программы.** В рассмотренных случаях обмен данными инициировался микропроцессором. Встречаются задачи, в которых обмен должен осуществляться в произвольных точках программы в моменты, определяемые периферийным устройством. При выполнении такого вида обмена данными по запросу, поступившему из ПУ, производится прерывание выполняемой микропроцессором программы и переход к выполнению специальной программы обмена.

В МПК серии 580 имеются микросхемы, предназначенные для построения интерфейса ПУ; одной из таких микросхем является программируемый параллельный интерфейс КР580ВВ55, описание которого приводится ниже.

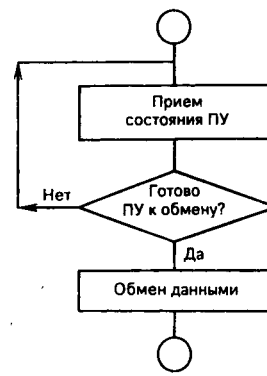


Рис. 5.22

**Программируемый параллельный интерфейс**. На рис.5.23 приведена упрощенная структурная схема *программируемого параллельного интерфейса* (ППИ).

С помощью ППИ осуществляется обмен данными (рис.5.24) между микропроцессором (МП) и различными ПУ. Для подключения ППИ к шине данных (ШД) микропроцессорной системы в ППИ предусмотрен 8-разрядный канал данных (КД). Периферийные устройст-

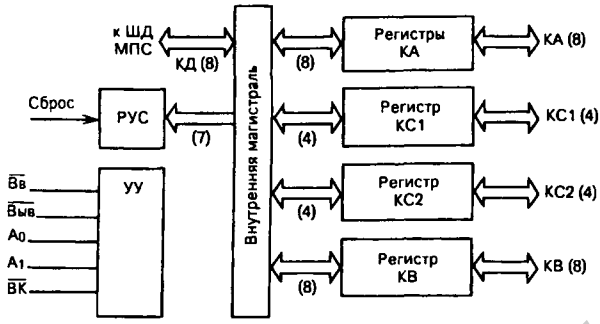



Рис. 5.23

ва могут подключаться к 8-разрядным каналам ППИ: КА, КВ, КС. Канал КС состоит из двух 4-разрядных подканалов 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 МПС, и выдачи их к ПУ, другой — для приема данных, поступающих от ПУ, и выдачи их на шину данных МПС. 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.

Таким образом, обмен между МП и ППИ распадается на две фазы обмена: обмен между регистром выбранного канала ППИ (регистром каналов КА, КВ, КС) и ШД МПС и обмен между регистрами каналов ППИ и ПУ. Рассмотрим, как организуется каждая из этих фаз обмена.

Обмен между ШД МПС и регистром ППИ организуется под управлением сигналов, подаваемых на входы устройства управления (УУ) ППИ. $A_0$, $A_1$ — содержимое двух младших разрядов шины адреса (ША) МПС, ВК — сигнал 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 МПС (системные управляющие сигналы на выходах микросхемы системного контроллера КР580ВК28). В табл. 5.18 показаны виды обмена данными между ШД

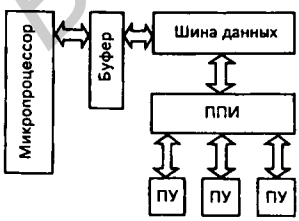



Рис. 5.24

МПС и регистрами ППИ и соответствующие им наборы значений сигналов выборки.

*Таблица 5.18*

| Вход | 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 |

По командам микропроцессора IN (ввод данных) и OUT (вывод данных) буферы канала КД обеспечивают обмен данными между ШД МПС и внутренней магистралью данных ППИ. Принятая с ШД МПС на внутреннюю магистраль данных ППИ информация представляет собой либо данные, которые через внутреннюю магистраль принимаются в регистр одного из каналов для дальнейшей их выдачи к ПУ, подключенному к этому каналу, либо 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. На рис. 5.25 представлен формат управляющего слова.

При поступлении из шины управления МПС сигнала Сброс все канальные регистры сбрасываются в нулевое состояние, 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.

**Режим 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.

**Режим 1**. В этом режиме передача данных может производиться через каналы КА и КВ, а канал КС используется в основном для приема и выдачи сигналов управления.

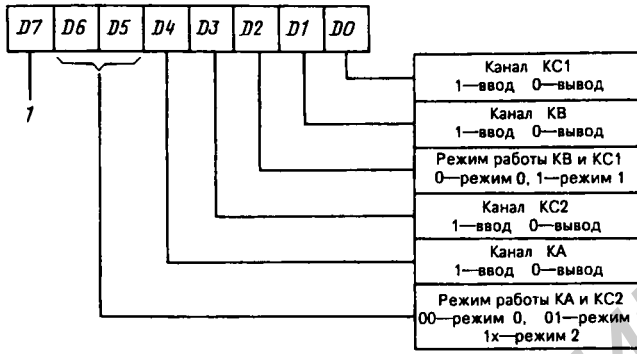



Рис. 5.25

Если канал КВ установлен на ввод информации, периферийное устройство одновременно с подачей данных в канал КВ подает в ППИ через разряд 2 канала КС *лог.0*, сигнализируя о выдаче информации. При этом через разряд 1 канала КС периферийное устройство получает от ППИ сигнал (*лог.1*) подтверждения приема выданных периферийным устройством данных.

В случае установки канала КА на ввод информации сигнал о выдаче данных периферийным устройством ППИ осуществляется через разряд 4 канала КС, сигнал подтверждения приема ППИ выдает на ПУ через разряд 5 канала КС.

При выводе информации через канал КВ разряд 1 канала КС используется для выдачи на ПУ сигнала (*лог.0*) вывода данных, через разряд 2 канала КС осуществляется прием от ПУ сигнала (*лог.0*) подтверждения записи выданных из ППИ данных в ПУ.

При выводе данных через канал КА для передачи указанных сигналов используются соответственно разряды 7 и 6 канала КС.

**Режим 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 $\overline{\text{Вв}}$ и $\overline{\text{Выв}}$, посылаемыми в ППИ, устанавливается направление передачи не только между МП и ППИ, но и между ППИ и подключенным к нему ПУ. Таким образом обеспечивается возможность быстрого переключения направления передачи информации в целом между МП и ПУ без предварительной засылки управляющего слова в ППИ при каждом изменении направления обмена.

В режиме 2 может работать только КА. Для передачи управляющих сигналов в этом режиме используются следующие разряды канала КС: разряд 4 для приема в ППИ сигнала (*лог.0*) выдачи данных из ПУ, разряд 5 для выдачи из ППИ сигнала (*лог.1*) подтверждения приема данных, разряд 7 для выдачи на ПУ сигнала (*лог.0*) вывода данных из ППИ, разряд 6 для приема от ПУ сигнала (*лог.0*) подтверждения записи в ПУ данных, выданных из ППИ.

## 5.6. РАЗВИТИЕ МИКРОПРОЦЕССОРНОГО КОМПЛЕКТА СЕРИИ 580

### Микропроцессор К1821ВМ85А

Микропроцессор ВМ85 представляет собой 8-разрядный микропроцессор, полностью совместимый с КР580ВМ80. При его разработке стремились усовершенствовать микропроцессор ВМ80. Введены следующие изменения:

повышена производительность за счет повышения тактовой частоты с 2,5 до 3 МГц;

уменьшено число дополнительных микросхем, требуемых для построения микропроцессорного устройства (в микропроцессоре ВМ85 предусмотрены функции генератора тактовых импульсов ГФ24, системного контроллера ВК28, частично контроллера прерываний);

обеспечен ввод-вывод данных в последовательной форме;

требуется лишь один источник питания 5 В.

На рис. 5.26 приведена структурная схема микропроцессора ВМ85.

При использовании микропроцессора уже не требуется отдельный тактовый генератор, достаточно к соответствующим выводам X1 и X2 подключить кварцевый резонатор (или внешний генератор).

Вместо системного контроллера для формирования системных управляющих сигналов ЧтП (чтение из памяти), ЗпП (запись в память), Ввв(ввод из ПУ), Выв (вывод в ПУ) может быть использована несложная схема, приведенная на рис. 5.27. Выходы, имеющие три состояния, обозначены z.

С точки зрения программиста архитектура микропроцессора ВМ85 практически тождественна архитектуре микропроцессора ВМ80. В нем присутствует тот же набор программно доступных регистров. Система команд включает весь набор команд ВМ80 в той же их кодировке, благодаря чему обеспечивается полная программная совместимость с микропроцессором ВМ80. Введены только две новые команды, связанные с расширением средств прерываний и последовательным вводом-выводом данных.

Отличия состоят в следующем. В микропроцессоре ВМ80 имеется лишь один вход запроса прерывания, который может быть подключен к одному источнику прерываний. Если необходимо иметь несколько источников прерываний, используют программируемый контроллер прерываний ВН59. В микропроцессоре ВМ85 наряду с входом прерываний RST*n*, аналогичным соответствующему входу микропроцессора ВМ80, предусмотрено несколько дополнительных входов для принятия запросов прерываний. В табл. 5.19 приведены начальные адреса прерывающих программ, к которым происходит обращение при поступлении запросов прерывания на соответствующие входы, и их приоритеты. Наивысшим приоритетом обладает вход RST 4.5. Если при реализации прерываний возникает запрос прерывания с более высоким приоритетом, происходит прерывание текущей прерывающей программы.

Все запросы прерываний, за исключением запросов по цепи RST 4.5, запрещаются или разрешаются командами DI и EI. Кроме того, они могут быть раздельно разрешены или запрещены с помощью команд SIM и RIM. С помощью команд SIM и RIM осуществляется также ввод и вывод данных в последовательной форме.

*Таблица 5.19*

| Вход | Приоритет | Стартовый адрес |
|---|---|---|
| RST 4.5 | 1 | 36 |
| RST 7.5 | 2 | 60 |
| RST 6.5 | 3 | 52 |
| RST 5.5 | 4 | 44 |
| RST *n* (0 ... 7) | 5 | 8*n* |

Наличие дополнительных (по сравнению с микропроцессором ВМ80) выводов, связанных с цепями прерываний и последовательным вводом-выводом, потребовало (при сохранении корпуса с 40 выводами) объединения функций вывода младшего байта адреса и данных $АД_7$ ... $АД_0$. В такте $T_1$ эти выводы используются для выдачи младшего байта адреса, в последующих тактах они используются для связи с шиной данных. Очевидно, такое совмещение функций требует фиксации выдаваемого через эти выводы в такте $T_1$ байта адреса во внешнем регистре, как показано на рис. 5.28.

В такте $T_1$ на выводы $АД_7$ ... $АД_0$ выдается младший байт адреса, он поступает на информационные входы D регистра КР580ИР82. Одновременно микропроцессор на вывод СтрА выдает сигнал, который служит стробом, открывающим информационные входы регистра, байт адреса фиксируется в регистре. В последующих тактах сигнал на выходе СтрА отсутствует, и информационные входы регистра оказываются логически отключенными. Принятая в такте $T_1$ информация сохраняется в течение последующих тактов цикла. В тактах обмена данными устанавливается связь с системной шиной данных через буфер, реализован-

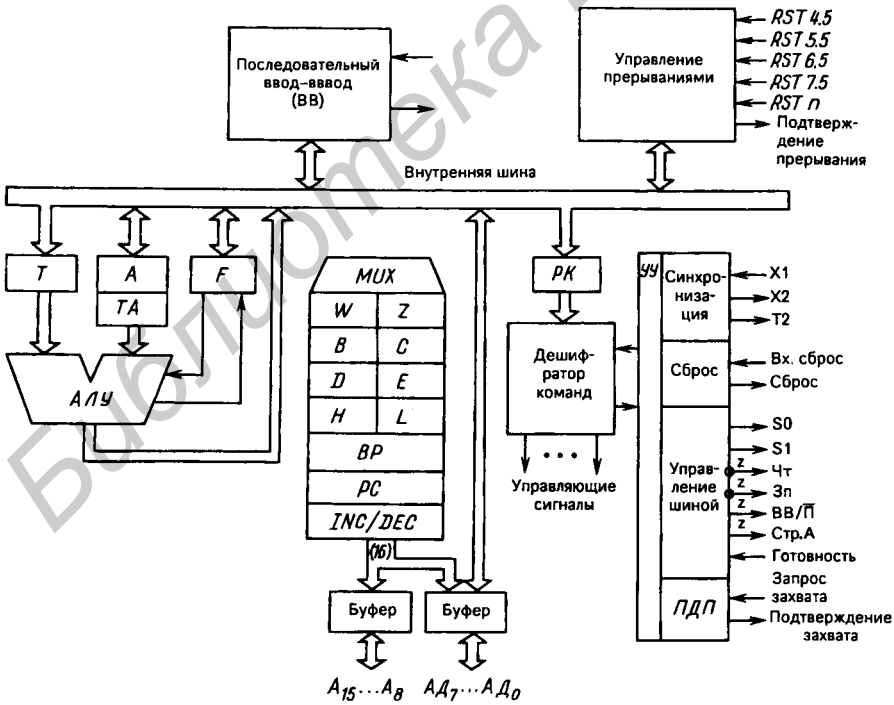



Рис. 5.26

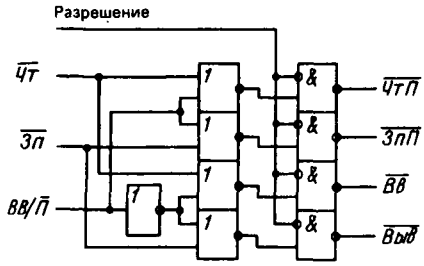



Рис. 5.27

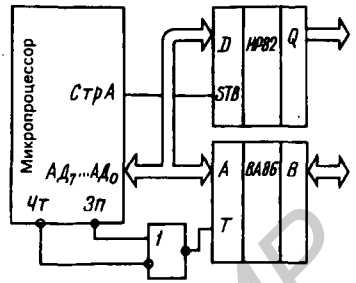



Рис. 5.28

ный на микросхеме ВА86. Направление передачи данных устанавливается сигналом, формируемым на входе *T* буфера.

## Микропроцессорный комплект серии 1810

**Состав микропроцессорного комплекта**. Микропроцессорный комплект серии 1810 является развитием микропроцессорного комплекта серии 580. Состав комплекта приведен в табл. 5.20.

*Таблица 5.20*

| Тип микросхемы | Назначение микросхемы |
|---|---|
| К1810ВМ86 | Центральный процессор |
| К1810ВМ88 | Центральный процессор с 8-разрядной внешней шиной данных |
| К1810ВМ87 | Арифметический сопроцессор |
| К1810ВМ89 | Специализированный процессор ввода-вывода |
| К1810ГФ84 | Генератор тактовых импульсов |
| К1810ВГ88 | Системный контроллер |
| К1810ВБ89 | Арбитр системной шины |
| К1810ВТ02 | Контроллер динамической памяти (16К) |
| К1810ВТ03 | Контроллер динамической памяти (64К) |
| К1810ВИ54 | Интервальный таймер |
| К1810ВТ37 | Контроллер прямого доступа к памяти |
| К1810ВН59 | Программируемый контроллер прерываний |
| К1810ИР82/83 | Регистр-защелка |
| К1810ВА86/87 | Шинный формирователь |

Отличия микропроцессора КР1810ВМ86 от микропроцессора КР580ВМ80 заключаются в следующем:

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 увеличилась примерно на порядок;

расширена разрядность шины данных и тем самым обеспечена возможность выполнения операций обмена и обработки 16-разрядных данных;

расширена разрядность шины адреса до 20 и, таким образом, обеспечена возможность адресации памяти емкостью до 1 Мбайта;

расширен набор команд.

**Структурная схема микропроцессора КР1810ВМ86**. На рис. 5.29 приведена структурная схема микропроцессора. По общему функциональному назначению узлы схемы можно разбить на три части: операционное устройство, устройство сопряжения с шиной и устройство управления. В операционном устройстве выполняются команды, 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; устройство управления формирует сигналы управления функционированием операционного устройства и устройства сопряжения с шиной.

Операционное устройство содержит следующие регистры. 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 регистра, из которых наименование левого (старшего) образуется

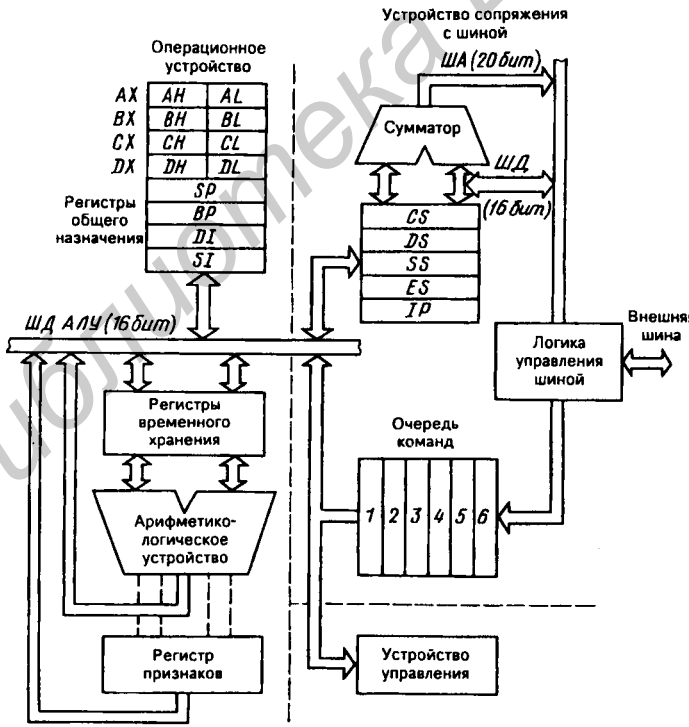



Рис. 5.29

добавлением к имени регистра символа H, наименование правого (младшего) регистра — добавлением символа L (например, AH, AL и т.д.).

Кроме указанных регистров, имеющих аналоги в микропроцессоре серии КР580, в микропроцессоре серии КР1810 предусмотрены 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 устройстве сопряжения с шиной из пяти 16-разрядных регистров CS, DS, SS, ES, IP только регистр IP имеет аналог в микропроцессоре серии КР580 (в микропроцессоре серии 580 к нему близок по функциональному назначению регистр PC — счетчик команд). Остальные регистры предназначены для указания области (сегмента) памяти, в которой находится адресуемая ячейка памяти. Указанные регистры имеют следующее наименование, связанное с их основным назначением: CS — сегмент программы, DS — сегмент данных, SS — сегмент стека, ES — дополнительный сегмент, IP — указатель команд.

Рассмотрим 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 580, приемы формирования адреса.

В оперативной памяти могут выделяться области (сегменты), состоящие из 64K 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 операнда вначале формируется так называемый *эффективный адрес*. Он может быть представлен содержимым регистров BX, BP, SI или DI. Эффективный адрес операнда может выражаться суммой содержимого регистра, указанного в команде, и представленного в непосредственной форме (в форме числа) *смещения*. Эффективный адрес может формироваться и более сложным путем, обычно используемым при обработке последовательности знаков: к некоторому базовому содержанию регистров BX или BP прибавляется содержимое индексного регистра DI или SI и смещение.

Эффективный 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 Кбайта. 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 эффективным адресом. Начальным адресом сегмента служит содержимое регистров CS, SS, DS или ES, сдвинутое влево на четыре разряда. Суммирование начального адреса сегмента с 16-разрядным эффективным адресом дает физический адрес в форме 20-разрядной кодовой комбинации, которая и выдается на адресную шину.

В АЛУ выполняются арифметические и логические операции. Операндами могут быть однобайтовые и двухбайтовые данные. По сравнению с микропроцессором серии 580 в данном микропроцессоре существенно расширен набор выполняемых операций. Список команд предусматривает выполнение операций умножения и деления чисел со знаком и без знака, три вида операций сдвига: логический, арифметический и циклический сдвиги на заданное число разрядов и ряд других операций. Очевидно, программы, составленные на языке ассемблера для микропроцессора КР1810ВМ86, не годятся для реализации на микропроцессоре серии 580. В то же время обеспечивается программная совместимость "снизу вверх", т. е. программы, написанные для микропроцессора серии 580, могут выполняться на микропроцессоре 1810.

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, совместить во времени процессы выборки команд из памяти и процессы их выполнения.

# Глава 6. Однокристальный микроконтроллер КМ1816ВЕ48

## 6.1. МИКРОКОНТРОЛЛЕРЫ СЕРИЙ 1816 И 1830

Особенностью построения современных технических систем является широкая автоматизация процессов контроля их состояния и управления их состоянием с помощью так называемых контроллеров (устройств управления). Именно для создания подобных устройств используется в настоящее время бóльшая часть выпускаемой электронной продукции. С целью сокращения аппаратурных затрат при построении контроллеров и снижения их стоимости производятся однокристальные микроконтроллеры. Выполненные в виде отдельных микросхем они несут в себе многие черты микроЭВМ.

В табл.6.1 приведены характеристики выпускаемых отечественной промышленностью микроконтроллеров серий 1816 и 1830. Первые шесть типов микросхем образуют семейство МК48, остальные типы относятся к семейству МК51. В таблице: РПП — резидентная память программ, т.е. память для хранения команд программы, организованная непосредственно в микросхеме (МПЗУ — масочное ПЗУ, программируемое непосредственно в процессе изготовления микросхемы, ППЗУ — перепрограммируемое ПЗУ); РПД — резидентная память данных, т.е. ОЗУ для хранения переменных, организованное внутри микросхемы.

Микросхемы серии 1816 выполнены по *n*МОП технологии, микросхемы серии 1830 — по КМОП технологии; этим определяется их низкое энергопотребление.

В пределах каждого семейства МК48 или МК51 микросхемы имеют одинаковые структурную организацию, систему команд, разводку выводов и, таким образом, полностью совместимы.

Микросхемы семейства МК51 по сравнению с МК48 имеют более сложную структурную организацию, характеризуются увеличенным объемом памяти, бóльшим быстродействием, наличием последовательного интерфейса (для ввода и вывода данных в последовательной форме), более развитой системой команд (включающей команды по

*Таблица 6.1*

| Тип микросхемы | Аналог микросхемы фирмы Intel, США | Тип и емкость РПП, байт | Емкость РПД, байт | Тактовая частота, МГц | Ток потребления, мА |
|---|---|---|---|---|---|
| КР1816ВЕ35 | 8035 | — | 64 | 6 | 135 |
| КР1816ВЕ48 | 8748 | ППЗУ, 1К | 64 | 6 | 135 |
| КР1816ВЕ39 | 8039 | — | 128 | 11 | 110 |
| КР1816ВЕ49 | 8049 | ПЗУ, 2К | 128 | 11 | 110 |
| КР1830ВЕ35 | 80С35 | — | 64 | 6 | 8 |
| КР1830ВЕ48 | 80С48 | ПЗУ, 1К | 64 | 6 | 8 |
| КР1816ВЕ31 | 8031АН | — | 128 | 12 | 150 |
| КР1816ВЕ51 | 8051АН | ПЗУ, 4К | 128 | 12 | 150 |
| КР1816ВЕ751 | 8751Н | ППЗУ, 4К | 128 | 12 | 220 |
| КР1830ВЕ31 | 80С31ВН | — | 128 | 12 | 18 |
| КР1830ВЕ51 | 80С51ВН | ПЗУ, 4К | 128 | 12 | 18 |

выполнению таких операций, как вычитание, умножение, деление и др.), бо́льшим числом портов для обмена данными с внешними устройствами.

Для упрощения дальнейшее изложение ориентировано на один тип микросхемы — КР1816ВЕ48.

## 6.2. СТРУКТУРНАЯ СХЕМА МИКРОКОНТРОЛЛЕРА КМ1816ВЕ48

Рассмотрим структурную схему микроконтроллера, приведенную на рис. 6.1. Обмен информацией между всеми узлами внутри микроконтроллера обеспечивается внутренней 8-разрядной шиной. Рассмотрим функции, выполняемые отдельными узлами.

**Арифметико-логическое устройство**. АЛУ выполняет арифметические и логические операции над 8-разрядными данными. При этом в качестве одного из операндов выступает содержимое аккумулятора, второй операнд (в случае двухоперандной операции) принимается с внутренней шины (R1 и R2 — регистры временного хранения информации). Результат операции с выхода АЛУ через внутреннюю шину засылается в аккумулятор.

**Десятичный корректор.** Обеспечивает коррекцию результата при выполнении операции *сложения* двухразрядных десятичных чисел с использованием *кода 8421* для изображения десятичных цифр.

**Флаги (признаки).** Для выполнения условных переходов предусмотрена возможность использования ряда признаков. Одни признаки (равенство или неравенство нулю содержимого аккумулятора, равенство единице содержимого любого указанного разряда аккумулятора, нали-

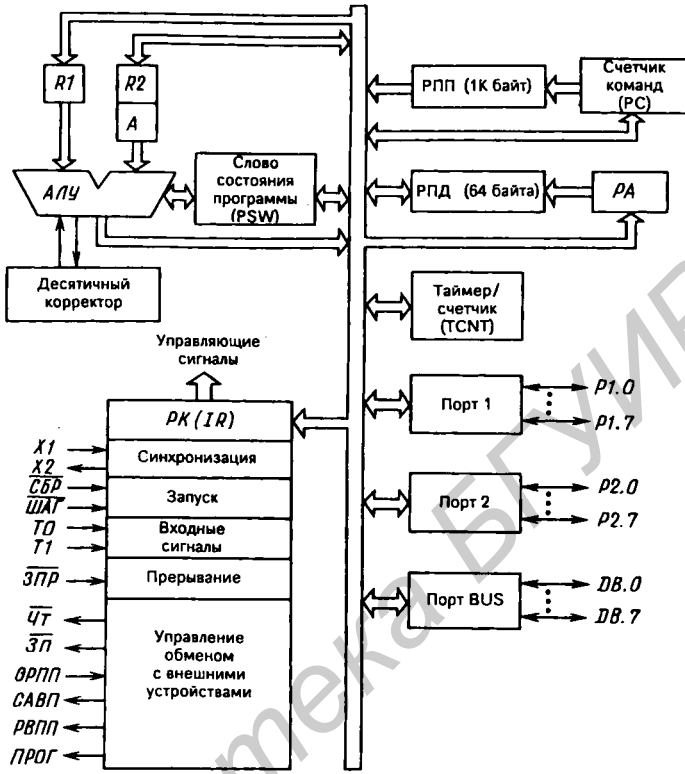



Рис. 6.1

чие переноса C при выполнении последней операции в АЛУ и признак переполнения таймера TF) формируются внутри микроконтроллера. Другие признаки (наличие *лог. 0* или *лог. 1* на входах T0 и T1, наличие *лог.0* на входе запроса прерывания ЗПР) поступают извне. Наконец, третьи (F0 = 1 и F1 = 1) задаются программным путем.

Некоторые из этих признаков записываются в триггерах регистра *слова состояния программы* (PSW). Формат слова состояния программы (ССП) приведен на рис. 6.2. Здесь АС — перенос из младшей тетрады при выполнении арифметической операции, используемый при десятичной коррекции. Три младших разряда регистра ССП выполняют роль указателя стека. При обращении к подпрограмме содержимое старшей тетрады ССП вместе с содержимым 12-разрядного счетчика команд PC пересылается в стек, а при выходе из подпрограммы эти данные возвращаются из стека в соответствующие узлы. Таким образом, при выходе из подпрограммы восстанавливаются

значения признаков, имевшиеся в ССП перед обращением к подпрограмме. Значения остальных флагов при обращении к подпрограмме не передаются в стек и, следовательно, не могут быть восстановлены при выходе из подпрограммы.

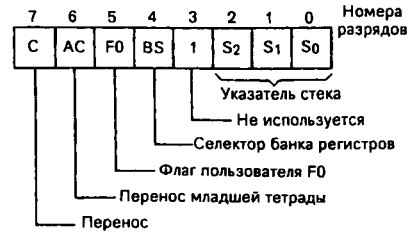



Рис. 6.2

**Память программ**. Для хранения команд программы в микроконтроллере предусмотрена внутренняя (резидентная) память, выполненная в виде перепрограммируемого ПЗУ емкостью 1К байт. Все адресное пространство памяти делится на два так называемых *банка*: банк 0 и банк 1. Адресация памяти производится счетчиком команд РС, который содержит 12 разрядов. Таким образом, адресное пространство для команд программы составляет 4К байта (рис. 6.3). В нем младшие адреса (1К байт) отведены для адресации *резидентной памяти программ* (РПП), остальные 3К байта — для адресации *внешней памяти программ* (ВПП).

В процессе выполнения программы происходит увеличение содержимого счетчика команд (СК) от 0 до 2047 (т.е. в пределах младших 11 разрядов PC0...PC10), старший, 12-й, разряд счетчика ($PC_{11}$) определяет номер банка, он устанавливается программным путем с помощью специальной команды.

В РПП предусмотрены три специализированных адреса:

0 — адрес начала программы (ее первой команды), устанавливается в счетчике команд по сигналу СБР;

3 — адрес, по которому размещается адрес прерывающей программы, вызываемой внешним устройством;

7 — адрес, по которому размещается адрес прерывающей программы, вызываемой при переполнении таймера/счетчика.

В командах условных переходов задается 8-разрядный адрес и переход осуществляется к команде, адрес которой формируется путем замещения в СК содержимого младших восьми разрядов 8-разрядным адресом из команды с сохранением в СК его старших разрядов. Таким образом, условный переход выполняется в пределах текущих 256 ячеек памяти, т.е. в пределах так называемой *страницы памяти*.

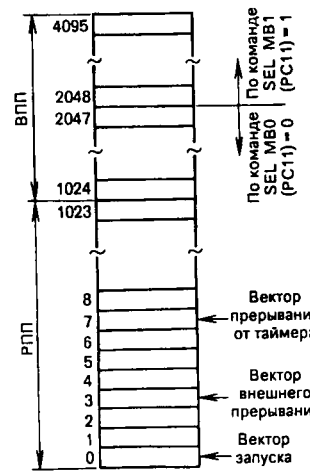



Рис. 6.3

При безусловном переходе или обращении к подпрограмме в команде задаются 11 разрядов адреса, и, таким образом, в СК сохраняется значение лишь старшего, 12-го разряда, т.е. переход осуществляется в пределах банка памяти емкостью 2048 ячеек.

При прерываниях старший разряд СК устанавливается в 0, и, таким образом, все прерывающие программы оказываются размещенными в пределах банка 0 с адресами ячеек 0...2047.

**Память данных.** Резидентная память данных (РПД) имеет емкость 64 байта. Адресное пространство данных включает (рис. 6.4) два банка по восемь рабочих регистров R0...R7 и R0*... R7* с адресами соответственно 0...7 и 24...31; выбор банка регистров производится командами SEL RB0 и SEL RB1; ячейки с адресами 8...23 могут использоваться в качестве стека для хранения в каждой паре ячеек содержимого 12-разрядного счетчика команд PC и старшей тетрады регистра PSW при обращении к подпрограммам (по команде CALL) (при отсутствии в программе восьми вложений подпрограмм свободные ячейки могут быть использованы для хранения данных); ячейки с адресами 32...63 предусмотрены для хранения данных.

При наличии внешней памяти данных (ВПД) возможно хранение в ней 256 байт по адресам, в качестве которых используется содержимое регистров R0 и R1.

**Порты ввода-вывода.** Для обмена информацией с внешними устройствами предусмотрены три порта: P1, P2 и BUS. Порты снабжены буферными 8-разрядными регистрами, в которых фиксируется выводимая информация. Перед вводом информации необходима настройка портов. Она осуществляется путем записи единицы в те разряды буферного регистра, номера которых совпадают с номерами разрядов порта, по которым предусматривается последовательный ввод данных.

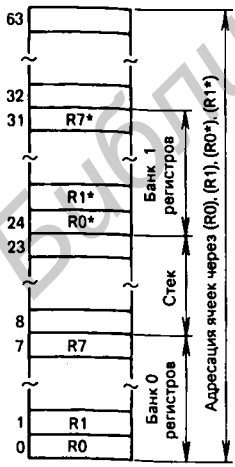



Рис. 6.4

При вводе данных выполняется операция поразрядного логического И над вводимыми данными и содержимым буферного регистра. Таким образом может осуществляться логическая операция И над вводимыми данными и зафиксированными в буферном регистре последними выведенными данными. Если такая операция не требуется, то перед вводом данных следует в буферный регистр занести единицу в те разряды, через которые осуществляется ввод данных. В составе команд микроконтроллера имеются команды, обеспечивающие выполнение операций И и ИЛИ над вводимыми данными и некоторой задаваемой в команде константой.

Особенность порта P2 состоит в том, что:

при обращении к внешней памяти программ этот порт участвует в адресации: через младшую тетраду порта передается содержимое старших четырех разрядов счетчика команд (разряды 8...11);

эта же младшая тетерада порта P2 может быть использована для организации четырех дополнительных 4-разрядных портов P4...P7 (подробное изложение такого расширения ввода-вывода приведено в § 6.5).

Порт BUS имеет особенность, заключающуюся в том, что обмен с внешними устройствами через этот порт сопровождается выдачей стробирующих сигналов (при выводе — сигнала $\overline{Зп}$, при вводе — сигнала $\overline{Чт}$); этот порт может быть использован для организации обмена информацией с внешней памятью программ и данных и для расширения ввода-вывода ( см. § 6.5).

**Синхронизация микроконтроллера.** Частота синхронизации задается кварцевым резонатором (с частотой 1...6 МГц), подключенным к входам X1 и X2 микроконтроллера. При работе от внешнего источника синхронизации последний подключается к входу X1. С помощью деления частоты в микроконтроллере формируются сигналы системной синхронизации с частотой, равной 1/3 основной частоты синхронизации, и сигналы с частотой циклов, равной 1/15 основной частоты синхронизации (с периодом 2,5 мкс при основной частоте синхронизации 6 МГц). Сигнал цикловой частоты выдается на выход Строб адреса внешней памяти (САВП) и используется для приема во внешний регистр адреса памяти, выдаваемого из микроконтроллера.

**Таймер/счетчик.** 8-разрядный суммирующий счетчик может быть поставлен в режим таймера или счетчика. В режиме таймера на его вход подаются синхросигналы частоты циклов, деленной на 32. Таким образом, просчитываемые им сигналы имеют период 80 мкс (при основной частоте синхронизации 6 МГц). При переполнении таймер сбрасывается в нуль с установкой признака переполнения таймера TF в 1. По этому признаку может быть осуществлен условный переход. Устанавливая в таймере различные начальные значения, можно получать требуемые временные интервалы между моментами возникновения переполнения. Эти интервалы могут быть в пределах от 80 мкс до $80 \cdot 256 = 20{,}48$ мс (при основной частоте синхронизации 6 МГц). Бóльшие временные интервалы можно получить программным путем, подсчитав определенное число переполнений.

В режиме счетчика производится подсчет числа переходов от *лог.1* к *лог.0* в сигнале, поступающем извне на вход T1.

**Режим прерываний.** В каждом машинном цикле производится проверка сигнала на входе запроса прерывания $\overline{ЗПР}$. При низком уровне сигнала после окончания выполнения текущей команды содержимое 12-разрядного счетчика команд PC и содержимое старшей тетрады

слова состояния программы PSW передаются в стек и производится переход к команде, выбираемой из ячейки РПП по адресу 3.

Участки программы, на которых разрешено прерывание, должны начинаться командой разрешения прерывания EN I и заканчиваться командой запрещения прерывания DIS I. На участках программы, на которых запрещено прерывание, сигнал на входе $\overline{\text{ЗПР}}$ может быть использован для выполнения условного перехода по команде JNI.

Сигнал запроса прерывания должен быть снят со входа ЗПР до выполнения команды возврата из прерывающей программы.

Следует иметь в виду, что вхождение в прерывающую программу, первая команда которой размещена в ячейке с адресом 3 (т.е. в банке 0 памяти программ), сопровождается установкой в нуль старшего разряда счетчика команд PC11. Поэтому при желании разместить команды прерывающей программы в банке 1 перед ними необходимо предусмотреть команду SEL MB1 перехода в памяти программ в банк 1.

Если в подпрограмме переключаются банки регистров в памяти данных с помощью команд SEL RB0, SEL RB1, возврат из подпрограммы может быть осуществлен по команде RETR с восстановлением содержимого регистра слова состояния программ (следовательно, с восстановлением и банка регистров). Если используется команда возврата RET, то не происходит восстановления содержимого ССП.

### 6.3. СИСТЕМА КОМАНД

В табл. 6.2 приведена система команд микроконтроллера КМ1816ВЕ48.

*Таблица 6.2.*

| Мнемоника команды | Код операции | Количество байтов в команде | Время выполнения команды, циклы | Выполняемые действия |
|---|---|---|---|---|
| **Арифметические операции** | | | | |
| ADD A, Rn | 01101rrr | 1 | 1 | A ← (A) + (Rn) |
| ADD A,@Ri | 0110000i | 1 | 1 | A ← (A) + ((Ri)) |
| ADD A, #d | 0000011 | 2 | 2 | A ← (A) + #d |
| ADDC A, Rn | 01111rrr | 1 | 1 | A ← (A) + (Rn) + (C) |
| ADDC A,@Ri | 0111000i | 1 | 1 | A ← (A) + ((Ri)) +(C) |
| ADDC A, #d | 00010011 | 2 | 2 | A ← (A) + #d + (C) |
| DA A | 01010111 | 1 | 1 | Десятичная коррекция (A) |
| INC A | 00010111 | 1 | 1 | A ← (A) + 1 |

| Мнемоника команды | Код операции | Количество байтов в команде | Время выполнения команды, циклы | Выполняемые действия |
|---|---|---|---|---|
| INC Rn | 00011rrr | 1 | 1 | Rn ← (Rn) + 1 |
| INC @Ri | 0001000i | 1 | 1 | ((Ri)) ← ((Ri)) + 1 |
| DEC A | 00000111 | 1 | 1 | A ← (A) – 1 |
| DEC Rn | 11001rrr | 1 | 1 | Rn ←(Rn) – 1 |
| **Логические операции** | | | | |
| ANL A, Rn | 01011rrr | 1 | 1 | A ← (A) ∧ (Rn) |
| ANL A, @Ri | 0101000i | 1 | 1 | A ← (A) ∧ ((Ri)) |
| ANL A, #d | 01010011 | 2 | 2 | A ← (A) ∧ #d |
| ORL A, Rn | 01001rrr | 1 | 1 | A ← (A) ∨ (Rn) |
| ORL A, @Ri | 0100000i | 1 | 1 | A ← (A) ∨ ((Ri)) |
| ORL A, #d | 01000011 | 2 | 2 | A ← (A) ∨ #d |
| XRL A, Rn | 11011rrr | 1 | 1 | A ← (A) ⊕ (Rn) |
| XRL A, @Ri | 1101100i | 1 | 1 | A ← (A) ⊕ ((Ri)) |
| XRL A, #d | 11010011 | 2 | 2 | A ← (A) ⊕ #d |
| CLR A | 00100111 | 1 | 1 | A ← 0 |
| CPL A | 00110111 | 1 | 1 | A ← $\overline{(A)}$ |
| SWAP A | 01000111 | 1 | 1 | $A_{0...3} \leftrightarrow (A_{4...7})$ |
| RL A | 11100111 | 1 | 1 | Циклический сдвиг влево (A) |
| RLC A | 11110111 | 1 | 1 | Сдвиг влево (A) через перенос |
| RR A | 01110111 | 1 | 1 | Циклический сдвиг вправо (A) |
| RRC A | 01100111 | 1 | 1 | Сдвиг вправо (A) через перенос |
| ANL Pp, #d | 100110pp | 2 | 2 | Pp ← (Pp) ∧ #d, p = 1, 2 |
| ANL BUS, #d | 10011000 | 2 | 2 | BUS ← (BUS) ∧ #d |
| ANLD Pp, A | 100111pp | 1 | 2 | Pp ← (Pp) ∧ ( $A_{0...3}$), p = 4...7 |
| ORL Pp, #d | 100010pp | 2 | 2 | Pp ← (Pp) ∨ #d, p = 1, 2 |
| ORL BUS, #d | 10001000 | 2 | 2 | BUS ← (BUS) ∨ #d |
| ORLD Pp, A | 100011pp | 1 | 2 | Pp ← (Pp) ∨ ( $A_{0...3}$), p = 4...7 |
| CLR C | 10010111 | 1 | 1 | C ← 0 |
| CLR F0 | 10000101 | 1 | 1 | F0 ← 0 |
| CLR F1 | 10100101 | 1 | 1 | F1 ← 0 |
| CPL C | 10100111 | 1 | 1 | C ← $\overline{(C)}$ |

| Мнемоника команды | Код операции | Количество байтов в команде | Время выполнения команды, циклы | Выполняемые действия |
|---|---|---|---|---|
| CPL F0 | 10010101 | 1 | 1 | F0 ← ($\overline{F0}$) |
| CPL F1 | 10110101 | 1 | 1 | F1 ← ($\overline{F1}$) |
| **Пересылка данных** | | | | |
| MOV A, Rn | 11111rrr | 1 | 1 | A ← (Rn) |
| MOV A, @Ri | 1111000i | 1 | 1 | A ← ((Ri)) |
| MOV A, #d | 00100011 | 2 | 2 | A ← #d |
| MOV Rn, A | 10101rrr | 1 | 1 | Rn ← (A) |
| MOV Rn, #d | 10111rrr | 2 | 2 | Rn ← #d |
| MOV @Ri, A | 1010000i | 1 | 1 | ((Ri)) ← (A) |
| MOV @Ri, #d | 1011000i | 2 | 2 | ((Ri)) ← #d |
| MOV A, PSW | 11000111 | 1 | 1 | A ← (PSW) |
| MOV PSW, A | 11010111 | 1 | 1 | PSW ← (A) |
| MOV A, T | 01000010 | 1 | 1 | A ← (T) |
| MOV T, A | 01100010 | 1 | 1 | T ← (A) |
| MOVX A,@Ri | 1000000i | 1 | 2 | A ← ((Ri)) |
| MOVX @Ri,A | 1001000i | 1 | 2 | ((Ri)) ← (A) |
| MOVP A, @A | 10100011 | 1 | 2 | $PC_{0...7}$ ← (A); A ← ((PC)) |
| MOVP3 A,@A | 11100011 | 1 | 2 | $PC_{0...7}$ ← (A); $PC_{8...11}$ ← 0011 A ← ((PC)) |
| XCH A, Rn | 00101rrr | 1 | 1 | A ← (Rn) |
| XCH A, @Ri | 0010000i | 1 | 1 | A ← ((Ri)) |
| XCHD A, @Ri | 0011000i | 1 | 1 | $A_{0...3}$ ← $((Ri)_{0...3})$ |
| IN A, Pp | 000010pp | 1 | 2 | A ← (Pp), p = 1,2 |
| INS A, BUS | 00001000 | 1 | 2 | A ← (BUS) |
| OUTL Pp, A | 001110pp | 1 | 2 | Pp ← (A), p = 1,2 |
| OUTL BUS, A | 00000010 | 1 | 2 | BUS ← (A) |
| MOVD A, Pp | 000011pp | 1 | 2 | $A_{0...3}$ ← (Pp), $A_{4...7}$ ← 0000, p = 4...7 |
| MOVD Pp, A | 001111pp | 1 | 2 | Pp ← $(A_{0...3})$, p = 4...7 |
| **Передача управления** | | | | |
| JMP ad 11 | $a_{10}a_9a_8$00100 | 2 | 2 | Безусловный переход по ad 11 |

| Мнемоника команды | Код операции | Количество байтов в команде | Время выполнения команды, циклы | Выполняемые действия |
|---|---|---|---|---|
| JMPP @A | 10110011 | 1 | 2 | $PC_{0...7} \leftarrow ((A))$. Переход в текущей странице памяти программ |
| DJNZ Rn, ad | 11101rrr | 2 | 2 | Декремент (Rn) и переход по ad, если (Rn) ≠ 0 |
| JC ad | 11110110 | 2 | 2 | Переход по ad, если (C) = 1 |
| JNC ad | 11100110 | 2 | 2 | Переход по ad, если (C) = 0 |
| JZ ad | 11000110 | 2 | 2 | Переход по ad, если (A) = 0 |
| JNZ ad | 10010110 | 2 | 2 | Переход по ad, если (A) ≠ 0 |
| JT0 ad | 00110110 | 2 | 2 | Переход по ad, если T0 = 1 |
| JNT0 ad | 00100110 | 2 | 2 | Переход по ad, если T0 = 0 |
| JT1 ad | 01010110 | 2 | 2 | Переход по ad, если T1 = 1 |
| JNT1 ad | 01000110 | 2 | 2 | Переход по ad, если T1 = 0 |
| JF0 ad | 10110110 | 2 | 2 | Переход по ad, если (F0) = 1 |
| JF1 ad | 01110110 | 2 | 2 | Переход по ad, если (F1) = 1 |
| JTF ad | 00010110 | 2 | 2 | Переход по ad, если TF = 1 |
| JNI ad | 10000110 | 2 | 2 | Переход по ad, если $\overline{ЗПР}$ = 0 |
| JBb ad | bbb10010 | 2 | 2 | Переход по ad, если (Bb) = 1 |
| CALL ad | $a_{10}a_9a_8$10100 | 2 | 2 | Вызов подпрограммы по ad |
| RET | 10000011 | 1 | 2 | Возврат из подпрограммы |
| RETR | 10010011 | 1 | 2 | Возврат из подпрограммы с восстановлением ССП |
| **Управление режимами** | | | | |
| STRT T | 01010101 | 1 | 1 | Запуск таймера |
| STRT CNT | 01000101 | 1 | 1 | Запуск счетчика |
| STOP TCNT | 01100101 | 1 | 1 | Останов таймера/счетчика |
| EN TCNTI | 00100101 | 1 | 1 | Разрешение прерывания от таймера/счетчика |
| DIS TCNTI | 00110101 | 1 | 1 | Запрещение прерывания от таймера/счетчика |
| EN I | 00000101 | 1 | 1 | Разрешение внешнего прерывания |
| DIS I | 00010101 | 1 | 1 | Запрещение внешнего прерывания |
| SEL RB0 | 11000101 | 1 | 1 | Выбор банка 0 РПД; (BS) ← 0 |

| Мнемоника команды | Код операции | Количество байтов в команде | Время выполнения команды, циклы | Выполняемые действия |
|---|---|---|---|---|
| SEL RB1 | 11010101 | 1 | 1 | Выбор банка 1 РПД; (BS) ← 1 |
| SEL MB0 | 11100101 | 1 | 1 | Выбор банка 0 памяти программ; (DBF) ← 0 |
| SEL MB1 | 11110101 | 1 | 1 | Выбор банка 1 памяти программ; (DBF) ← 1 |
| ENT0 CLC | 01110101 | 1 | 1 | Выдача синхросигнала на выход T0 |
| NOP | 00000000 | 1 | 1 | Холостая команда |

В таблице 6.2. используются следующие значения операндов:

| | |
|---|---|
| Rn (n = 0, ... , 7) | рабочий регистр |
| @Ri (i = 0, 1) | адрес (РПД и в командах с мнемоникой операции, заканчивающейся буквой X, ВПД), в качестве которого используется содержимое регистров R0, R1 |
| #d | операнд, задаваемый числовым значением |
| A | аккумулятор |
| @A | 8-разрядный адрес памяти программ, в качестве которого используется содержимое аккумулятора |
| PSW | регистр слова состояния программы |
| Pp (p = 1, 2) | порты P1, P2 |
| Pp (p = 4 ... 7) | порты схемы расширения в командах с мнемоникой операции, заканчивающейся буквой D |
| BUS | порт BUS |
| C, F0, F1 | флаги |
| ad | 8-разрядный адрес памяти программ, задаваемый в команде |
| ad 11 | 11-разрядный адрес памяти программ, задаваемый в команде |
| T | таймер в команде STRT |
| T | таймер/счетчик в команде MOV |
| TCNT | таймер/счетчик в команде STOP |
| TCNTI | таймер/счетчик в командах EN, DIS |
| CNT | счетчик |
| I | в командах EN, DIS соответственно разрешение, запрещение внешнего прерывания |
| RB0, RB1 | соответственно нулевой и первый банки регистров |
| MB0, MB1 | соответственно нулевой и первый банки памяти программ |
| CLC | в команде ENT0 разрешение выдачи синхросигнала основной частоты на выход T0 |

Команды имеют однобайтовый и двухбайтовый форматы (большинство команд однобайтовые). Время выполнения большинства команд составляет один цикл (2,5 мкс при тактовой частоте 6 МГц); команды с числовым значением операнда, команды ввода-вывода и передачи управления выполняются за два цикла (за время 5 мкс при тактовой частоте 6 МГц).

### 6.4. СИСТЕМА С ВНЕШНЕЙ ПАМЯТЬЮ И РАСШИРЕННЫМ ВВОДОМ-ВЫВОДОМ

Микроконтроллер может быть использован как функционально законченное устройство. В тех случаях, когда его возможности оказываются недостаточными (не хватает емкостей РПП и РПД, возникает необходимость обмена данными с большим числом устройств ввода-вывода), с помощью дополнительных средств память программ может быть расширена до 4К байт, память данных — до 320 байт, количество устройств ввода-вывода увеличено неограниченно.

**Система с внешней памятью программ** (ВПП). На рис.6.5 показано подключение ВПП к микроконтроллеру. ВПП представлена тремя блоками емкостью 1К байт.

Если адрес команды превышает верхнюю границу адресного пространства резидентной памяти программ (т.е. оказывается равным 1024 и больше), микроконтроллер формирует сигналы, предназначенные для управления ВПП (за исключением сигнала САВП, формируемого в каждом машинном цикле):

содержимое счетчика команд выдается через BUS и P2 (младшие восемь разрядов СК выводятся через порт BUS, старшие четыре разряда СК — через младшую тетраду порта P2);

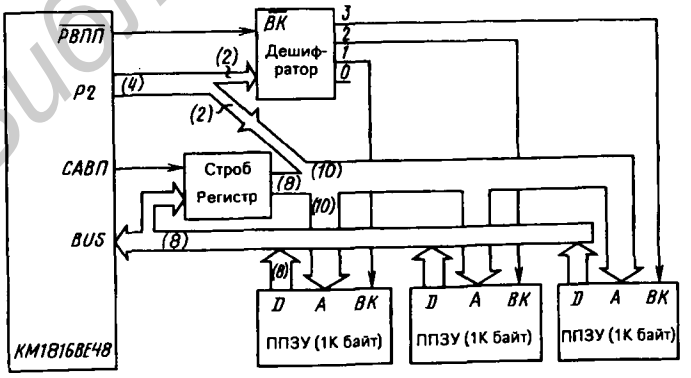



Рис. 6.5

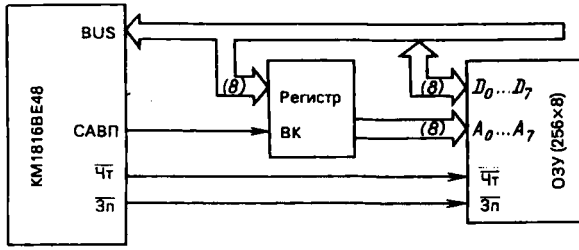



Рис. 6.6

содержимое порта BUS в момент снятия сигнала на выходе САВП фиксируется в регистре R и порт BUS освобождается для ввода в микроконтроллер выводимой из памяти команды;

содержимое регистра и двух младших разрядов порта P2 образуют 10-разрядный адрес, подаваемый на все три блока ВПП, два старших разряда порта P2 используются для выбора одного из трех блоков памяти;

при снятии сигнала разрешения ВПП ($\overline{РВПП}$) порт BUS переводится в режим ввода поступающего из ВПП байта команды в микроконтроллер.

**Система с внешней памятью данных** (ВПД). На рис.6.6 показана схема с ВПД. Обращение к этой памяти производится по командам MOVX. По этим командам содержимое указанного в команде регистра Ri (R0 или R1) выдается через порт BUS и при снятии сигнала САВП фиксируется в буферном регистре, и порт BUS освобождается для передачи байта данных. Сигналами Чт и Зп устанавливается режим работы блока памяти (чтение или запись). При чтении поступающий из памяти байт данных передается в микроконтроллер, при записи выдаваемый из микроконтроллера на порт BUS байт записывается в ячейку памяти.

**Система с расширенным вводом-выводом.** На рис.6.7 приведен вариант подключения устройств ввода-вывода к микроконтроллеру через

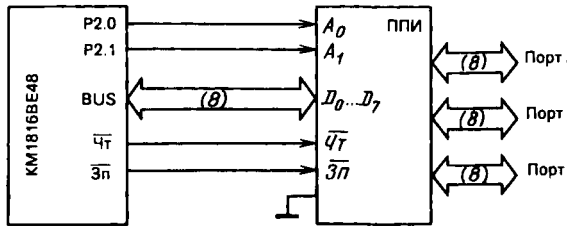



Рис. 6.7.

порты программируемого параллельного интерфейса (ППИ) КР580ВВ55. Адресация порта ППИ производится через два младших разряда порта P2 (P2.0 и P2.1), передача байта данных — через порт BUS.

### 6.5. ПРИЕМЫ ПРОГРАММИРОВАНИЯ

*Пример 1.* Прибавить к содержимому рабочего регистра R0 в банке 0 содержимое восьми ячеек РПД с последовательно нарастающими адресами, начиная с ячейки с адресом $46_{16}$.

```
Программа:
        SEL     RB0         ; выбор банка 0 регистров
        MOV     R1, #46H    ; задание начального адреса РПД
        MOV     R2, #8      ; установка счетчика суммируемых чисел
        MOV     A, R0       ; пересылка в аккумулятор содержимого R0
LOOP:   ADD     A, @R1      ; прибавление очередного числа из РПД
        INC     R1          ; формирование адреса очередного числа
        DJNZ    R2, LOOP    ; декремент R2 и условный переход, если (R2) ≠ 0
        MOV     R0, A       ; передача в R0 суммы чисел из аккумулятора
```

*Пример 2.* Вычесть из содержимого ячейки ВПД с адресом 5 содержимое регистра R0.

Для вычитания 8-разрядных чисел без знака $Y \leftarrow Y - X$ может быть использовано выражение $Y \leftarrow \overline{\overline{Y} + X}$, где $\overline{Y}$ — поразрядная инверсия $Y$. Покажем справедливость этого выражения. Для 8-разрядного двоичного числа инверсия есть дополнение до $2^8 - 1 = 255$. Следовательно, $\overline{\overline{Y} + X} = 255 - (\overline{Y} + X) = 255 - (255 - Y + X) = Y - X$.

```
Программа:
MOV     R1, #5      ; пересылка в регистр R1 адреса ячейки ВПД
MOVX    A, @R1      ; пересылка в A содержимого ячейки ВПД
CPL     A           ; инверсия содержимого аккумулятора
ADD     A, R0       ; сложение
CPL     A           ; инверсия содержимого аккумулятора
MOVX    @R1, A      ; пересылка содержимого A в ВПД
```

Возникновение переноса $C = 1$ в процессе выполнения операции сложения свидетельствует о том, что полученный результат — отрицательное число (представленное в дополнительном коде). Другой способ

вычитания при решении данной задачи может быть основан на следующем выражении: $Y - X = Y + \overline{X} + 1$.

Программа с использованием этого выражения:

```
XCH   A, R0     ; обмен содержимого аккумулятора и регистра R0
CPL   A         ; инверсия содержимого аккумулятора
XCH   A, R0     ; обмен содержимого аккумулятора и регистра R0
MOV   R1, #5    ; установка в R1 адреса ВПД
MOV   A, @R1    ; пересылка в А содержимого ячейки ВПД
ADD   A, R0     ; сложение
INC   A         ; инкремент содержимого аккумулятора (A) ← (A) + 1
MOVX  @R1, A    ; пересылка содержимого А в ячейку ВПД
```

В этом случае отсутствие переноса $C = 0$ при сложении есть свидетельство получения отрицательного числа, представленного в дополнительном коде.

*Пример 3.* Составить программу умножения 8-разрядных данных.

Пусть множимое и множитель содержатся соответственно в регистрах R0 и R1. Произведение будем формировать в аккумуляторе (старшие разряды) и регистре R2 (младшие разряды). Используем регистр R3 в качестве счетчика числа повторений тела цикла программы.

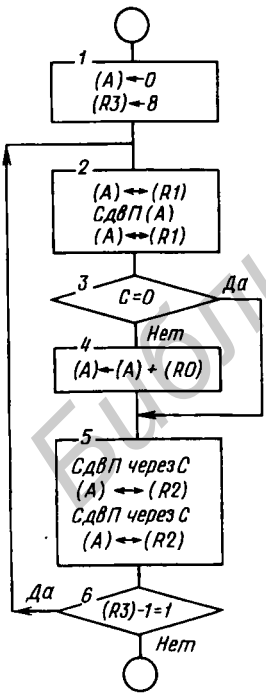



Рис. 6.8

На рис.6.8 приведена схема алгоритма операции. При построении алгоритма использован метод умножения с суммированием частичных произведений, начиная с младшего частичного произведения. После прибавления к сумме предыдущих частичных произведений очередного частичного произведения производится сдвиг суммы вправо. Выдвигаемый при сдвиге младший разряд суммы, не участвующий в дальнейшем суммировании, вдвигается в регистр младших разрядов через его старший разряд.

Алгоритм и составленная на его основе программа содержат следующие особенности.

В системе команд микроконтроллера выполнение операции сдвига предусмотрено только над содержимым аккумулятора. Поэтому, если требуется сдвинуть содержимое некоторого регистра, производится обмен содержимым между аккуму-

лятором и этим регистром и после выполнения сдвига содержимое аккумулятора вновь обменивается с содержимым регистра.

После прибавления к содержимому аккумулятора очередного частичного произведения может возникнуть перенос С из старшего разряда. С помощью команды "сдвиг вправо через С" этот перенос вдвигается в старший разряд аккумулятора, а выдвигаемый при этом из аккумулятора младший разряд его содержимого вдвигается в старший разряд регистра R2 при команде "сдвиг вправо через С" его содержимого.

Программа умножения:

```
MUL:    CLR     A       ; сброс аккумулятора: (A) ← 0
        MOV     R3, #8  ; установка начального значения в счетчике
M1:     XCH     A, R1   ; обмен регистра R1 с аккумулятором: (A) ↔ (R1)
        RR      A       ; сдвиг вправо содержимого аккумулятора
        XCH     A, R1   ; обмен регистра R1 с аккумулятором: (A) ↔ (R1)
        JNC     M2      ; переход, если нет переноса: (C) = 0
        ADD     A, R0   ; прибавление множимого: (A) ← (A) + (R0)
M2:     RRC     A       ; сдвиг вправо через перенос C
        XCH     A, R2   ; обмен регистра R2 с аккумулятором: (A) ↔ (R2)
        RRC     A       ; сдвиг вправо через перенос C
        XCH     A, R2   ; обмен регистра R2 с аккумулятором: (A) ↔ (R2)
        DJNZ    R3, M1  ; декремент R3 и переход на M1, если не нуль
        RETR            ; возврат
```

Программа оформлена в форме подпрограммы с именем MUL и возвратом с восстановлением слова состояния программы RETR.

*Пример 4.* Выдать во внешнее устройство через порт BUS число импульсов, поступивших в микроконтроллер за некоторый временной интервал.

Рассмотрим три варианта решения задачи.

В а р и а н т 1. Пусть имеется два источника (рис. 6.9), один из которых задает временной интервал в виде импульса (положительной полярности) большой длительности, другой является источником импульсов отрицательной полярности, подлежащих счету.

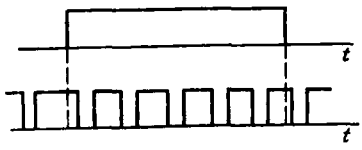


Рис. 6.9

Используем узел таймер/счетчик микроконтроллера в качестве счетчика (для нормальной работы счетчика период

просчитываемых импульсов должен быть не меньше длительности трех машинных циклов). При этом подлежащие счету импульсы следует подавать на вход T1 микроконтроллера, а в программе предусмотреть после поступления положительного перепада напряжения импульса, задающего временной интервал, запуск счетчика командой STRT CNT и после отрицательного перепада — команду останова счетчика STOP TCNT. Полученное после останова число в счетчике есть искомый результат.

На рис. 6.10 приведена схема алгоритма решения данной задачи. Предполагается, что импульс, задающий временной интервал, подается на вход T0 микроконтроллера. Блоки 2 и 4 осуществляют ожидание появления на входе T0 соответственно *лог.1* и *лог.0*. Таким образом они обеспечивают фиксацию моментов, соответствующих перепадам напряжения импульса, задающего временной интервал.

```
Программа:
        CLR     A       ; сброс аккумулятора: (A) ← 0
        MOV     T, A    ; сброс счетчика: (T) ← (A)
M1:     JNT0    M1      ; ожидание положительного фронта на входе T0
        STRT    CNT     ; запуск счетчика
M2:     JT0     M2      ; ожидание отрицательного фронта на входе T0
        STOP    CNT     ; останов счетчика
        MOV     A, T    ; пересылка в аккумулятор содержимого счетчика
        OUTL    BUS A   ; вывод
```

В а р и а н т 2. Пусть временной интервал задан числовым значением, а узел таймер/счетчик по-прежнему используется в режиме счетчика и просчитываемые импульсы подаются на вход T1.

Временну́ю задержку, равную заданному временно́му интервалу, можно реализовать, построив цикл, удовлетворяющий определенным условиям. Пусть время однократного прохождения тела цикла программы $t_ц$ и выполнение тела цикла происходит $k$ раз. Тогда время, затрачиваемое на прохождение цикла $t_з = kt_ц$, можно сделать равным заданному временно́му интервалу, в котором следует производить подсчет числа импульсов.

На рис. 6.11 приведена схема алгоритма цикла, обеспечивающего требуемую задержку. Каждая из команд здесь выполняется за два машинных цикла. При тактовой частоте 6 МГц длительность машинного цикла равна 2,5 мкс и на выполнение каждой команды затрачивается 5 мкс. Следовательно, время задержки равно $t_з = (5k + 1)$ мкс. Если требуется иметь $t_з = 600$ мкс, то $k = t_з/5 - 1 = (600 - 1)/5 \approx 120$.

Программа решения задачи:

```
        CLR     A           ; сброс аккумулятора
        MOV     T, A        ; сброс счетчика
        STRT    CNT         ; запуск счетчика
        MOV     R0, #120    ; передача в регистр R0 числа повторений цикла
WAIT:   DJNZ    R0, WAIT    ; цикл, задающий временной интервал

        STOP    CNT         ; останов счетчика
        MOV     A, T        ; пересылка содержимого счетчика в аккумулятор
        OUTL    BUS, A      ; выдача результата в порт BUS
```

В а р и а н т 3. Используем узел таймер/счетчик в режиме таймера для задания требуемого временного интервала.

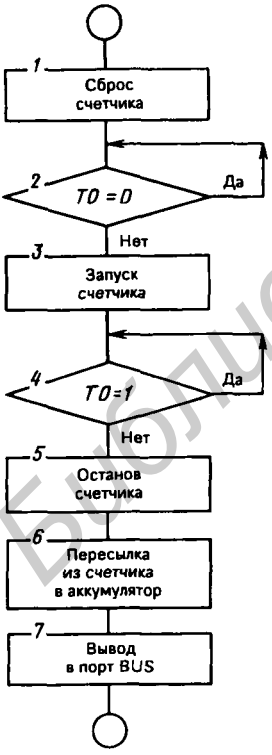



Рис. 6.10

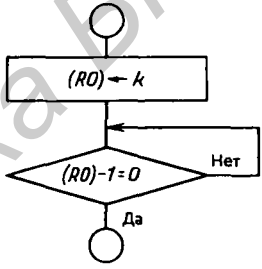



Рис. 6.11

В режиме таймера ведется подсчет сигналов машинного цикла (с периодом 2,5 мкс при тактовой частоте 6 МГц), частота которых предварительно делится на 32. Таким образом, при тактовой частоте 6 МГц период поступления сигналов в таймер $t_{з\,min} = 2{,}5 \cdot 32 = 80$ мкс. Максимальный интервал времени задержки, обеспечиваемый таймером, $t_{з\,max} = 80 \cdot 256 = 20480$ мкс $\approx 20$ мс.

Для получения требуемой задержки в таймер предварительно заносится число $256 - t_з/80$, где $t_з$ выражено в микросекундах, и после

запуска таймера ожидается момент переполнения таймера, т.е. достижения в нем числа 256, при котором происходит сброс в нуль и выдача переноса TF = 1.

Пусть требуется подсчитать количество импульсов, поступающих на вход микроконтроллера за 16 мс. Предварительно заносимое в таймер число равно 256 – 16000/80 = 56.

Программа решения задачи:

```
        MOV     A, #56      ; передача в аккумулятор числа 56
        MOV     T, A        ; пересылка содержимого аккумулятора в таймер
        CLR     A           ; сброс аккумулятора
        START   T           ; запуск таймера
M1:     JTF     M2          ; выход из цикла при переполнении таймера (TF = 1)
        JT0     M1          ; переход, если T0 = 1 (нет импульса)
WAIT:   JNT0    WAIT        ; ожидание конца импульса
        INC     A           ; инкремент аккумулятора
        JMP     M1          ; безусловный переход
M2:     OUTL    BUS, A      ; вывод результата в порт
```

Приведенные программы построены в предположении, что количество просчитываемых импульсов за заданный временной интервал не достигает значения 256 и, таким образом, не происходит переполнения узла, в котором ведется счет импульсов. Это ограничение легко снимается. Предполагается самостоятельно построить программы для случая, когда количество просчитываемых импульсов должно представляться двухбайтовым числом.

## *Упражнения*

1. Составьте программу решения следующей задачи. Выполните опрос сигнала на входе T0 микроконтроллера с интервалом времени 100 мкс; после 10-кратного обнаружения *лог.1* на входе T0 содержимое аккумулятора вывести через порт P1.

У к а з а н и е: рассмотрите схему алгоритма на рис. 6.12 и воспользуйтесь ею для составления программы.

2. Составьте программу решения следующей задачи. Подсчитайте количество импульсов, поступающих на вход T1 в течение времени, задаваемого длительностью импульса на входе T0 (см. рис. 6.13,*а*).

У к а з а н и е: рассмотрите схему алгоритма на рис. 6.13,*б* и воспользуйтесь ею для составления программы.

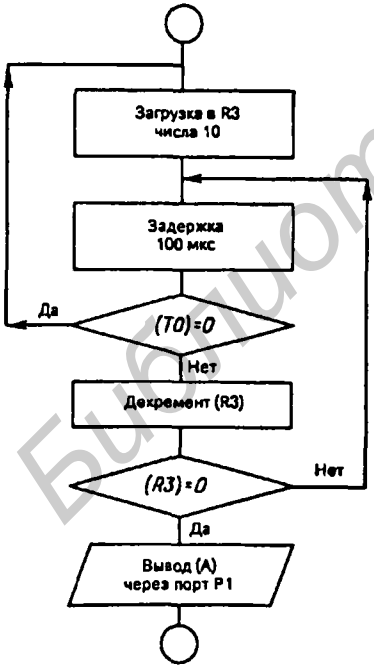



Рис. 6.12

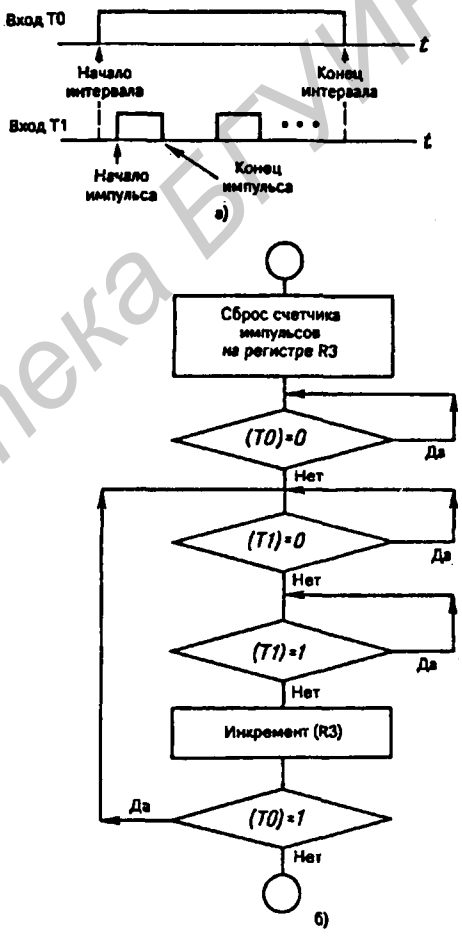



Рис.6.13

3. Составьте программу решения следующей задачи. Подсчитайте число импульсов, поступающих на вход T1 за время 1 с (рис. 6.14,*а*). Для счета импульсов используйте счетчик микроконтроллера.

У к а з а н и е: рассмотрите схему алгоритма на рис. 6.14,*б* и воспользуйтесь ею для составления программы.

4. Составьте программу решения следующей задачи. Подсчитайте число импульсов за временной интервал 10 мс (рис. 6.15,*а*).

У к а з а н и е: рассмотрите схему алгоритма на рис. 6.15.*б* и воспользуйтесь ею для составления программы.

5. Составьте программу решения следующей задачи. Определите длительность импульса, поступающего на вход T0, приняв за единицу временно́го интервала 100 мкс.

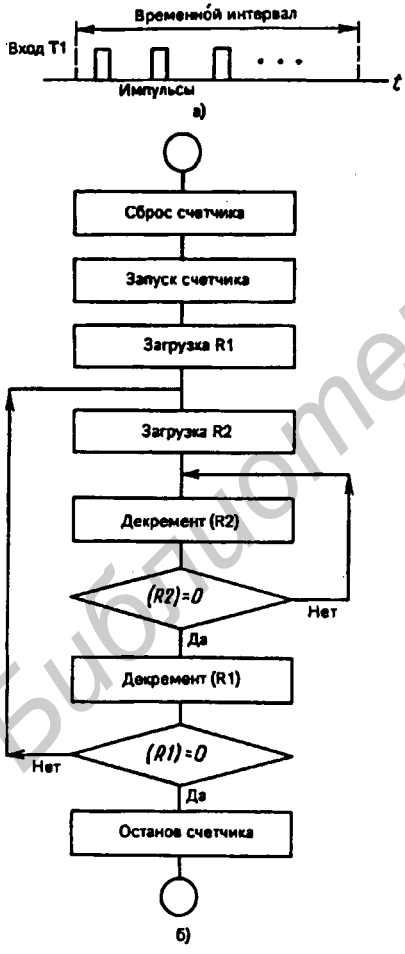



Рис. 6.14

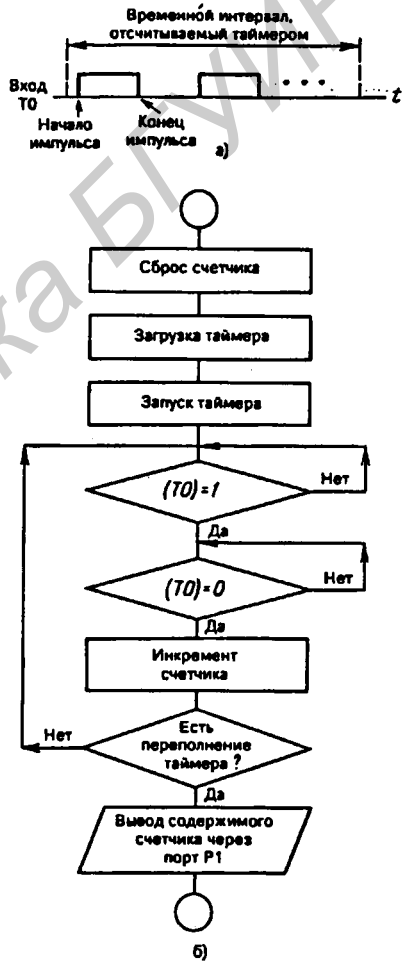



Рис. 6.15

У к а з а н и е: рассмотрите схему алгоритма на рис. 6.16 и воспользуйтесь ею для составления программы.

6. Составьте программу решения следующей задачи. Определите период импульсной последовательности, поступающей на вход T0, приняв за единицу временного интервала 1 мс (рис. 6.17, *а*).

У к а з а н и е: рассмотрите схему алгоритма на рис. 6.17,*б* и воспользуйтесь ею для составления программы.

7. Составьте программу преобразования параллельного кода в последовательный. Обычно при таком преобразовании используется стартстопный (асинхронный) режим обмена данными. Передача последовательного байта предваряется посылкой старт-бита (*лог.0*) и завершается выдачей стоп-бита (*лог.1*).

Выдачу байта производить через разряд 3 порта P1 (P1.3).

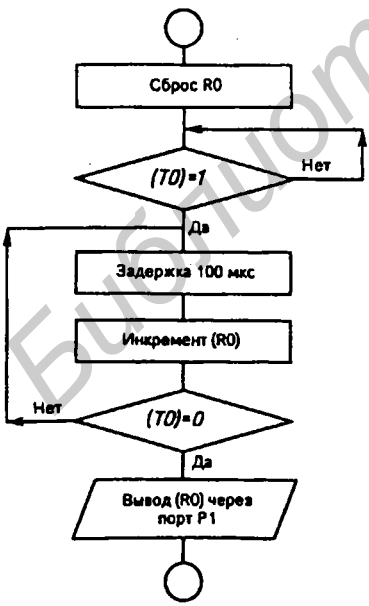



Рис. 6.16

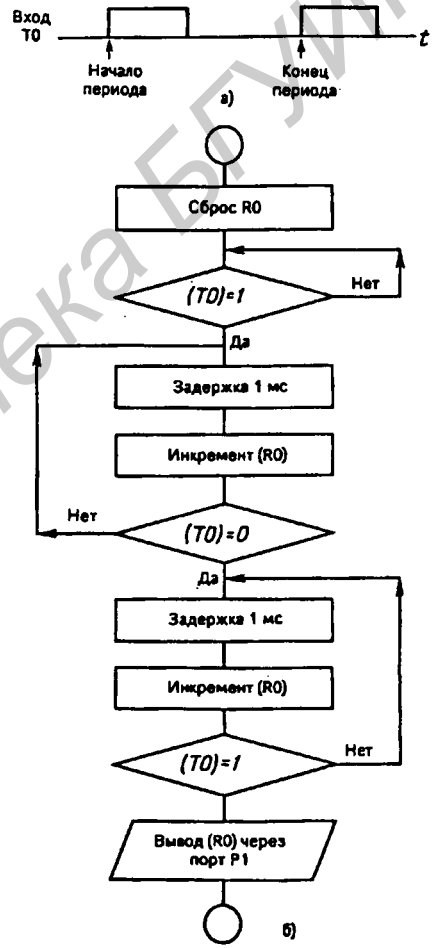



Рис. 6.17

У к а з а н и е: рассмотрите схему алгоритма на рис. 6.18 и воспользуйтесь ею для составления программы.

8. Составьте программу преобразования последовательного кода в параллельный, используя стартстопный (асинхронный) режим обмена данными. Передача последовательного кода байта предваряется посылкой старт-бита (*лог.0*) и завершается выдачей стоп-бита (*лог.1*).

Прием последовательного кода производить через вход T0 микроконтроллера.

У к а з а н и е: рассмотрите схему алгоритма на рис. 6.19 и воспользуйтесь ею для составления программы.

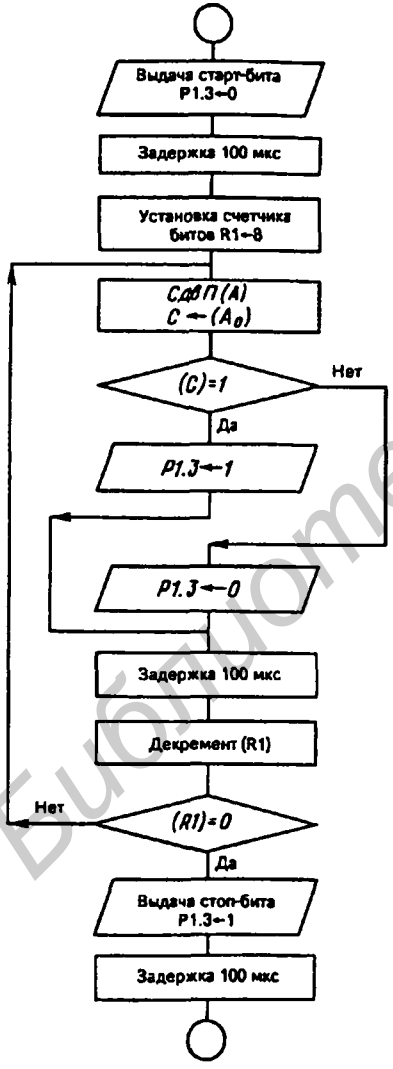



Рис. 6.18

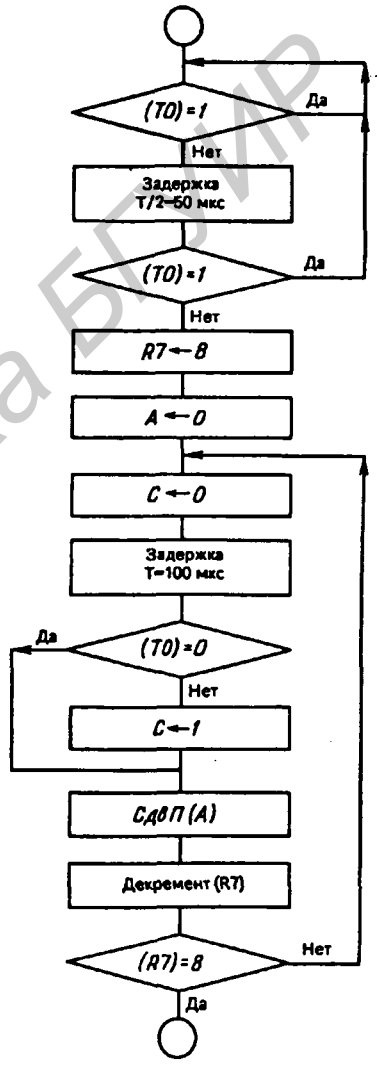



Рис. 6.19

# Глава 7. Однокристальная микроЭВМ КМ1813ВЕ1 для цифровой обработки сигналов

## 7.1. ОСОБЕННОСТИ МИКРОСХЕМЫ

Микросхема КМ1813ВЕ1 содержит процессор, память данных и программ, т.е. набор узлов, характерных для микроЭВМ. Кроме того, в ней имеются аналого-цифровой преобразователь (АЦП) для преобразования отсчетов, взятых из поданного на вход аналогового сигнала, в цифровую форму, и цифроаналоговый преобразователь (ЦАП) для преобразования результатов проведенных в процессоре вычислений из цифровой формы в аналоговую для выдачи на выход.

Состав команд микросхемы и высокая точность выполнения операций позволяют с ее помощью строить сложные системы цифровой обработки сигналов и цифровые системы управления объектами.

Цифровая обработка сигналов включает много видов обработки. Среди них основными являются линейная цифровая фильтрация и спектральный анализ. Спектральный анализ предусматривает определение спектрального состава сигнала. При цифровой фильтрации по заданному закону изменяется спектральный состав сигнала, т. е. воспроизводятся те изменения сигнала, которые возникают при его прохождении через фильтр — линейную цепь с определенной частотной характеристикой.

На одной микросхеме КМ1813ВЕ1 можно построить фильтр достаточно высокого порядка — такого, какой в аналоговой форме достигается с использованием до 40 реактивных элементов. Либо можно построить систему фильтров более низкого порядка. Кроме фильтров микросхема позволяет реализовывать функции многих других типовых узлов аппаратуры: детекторов, ограничителей, генераторов колебаний различной формы, преобразователей частоты и др.

Микросхема выполнена по *n*МОП-технологии, корпус микросхемы имеет прозрачную крышку для стирания содержимого памяти программ ультрафиолетовым излучением перед записью в нее новой программы.

## 7.2 СТРУКТУРНАЯ СХЕМА

На рис. 7.1 приведена структурная схема КМ1813ВЕ1. Она может быть разбита на три части: аналоговую часть (I), устройство цифровой обработки (II), память команд (III).

В устройство цифровой обработки входит сверхоперативное запоминающее устройство (СОЗУ) , содержащее 40 ячеек для хранения 25-разрядных слов, 9-разрядный регистр данных DAR, через который осуществляется ввод и вывод данных (по содержимому отдельных раз-

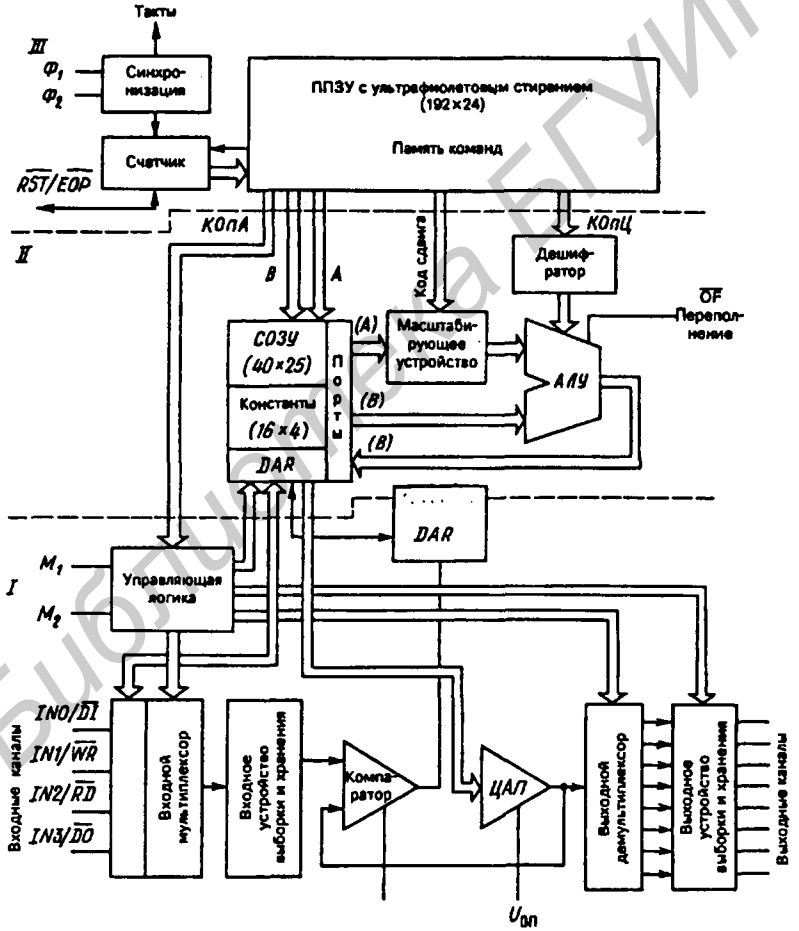



Рис. 7.1

рядов этого регистра могут выполняться условные переходы), 16 ячеек для хранения констант. Адреса ячеек СОЗУ представляются 6-разрядными двоичными комбинациями. Начальные адреса соответствуют ячейкам, предназначенным для хранения значений переменных. Адреса, имеющие структуру 11xxxx (т.е. содержащие 1 в двух старших разрядах), являются адресами ячеек, хранящих константы. Значения констант равны $0{,}125 \cdot m$, где $m$ может принимать целочисленные значения, лежащие в пределах –8...+7. Значения констант и соответствующие им мнемонические обозначения приведены в табл. 7.1.

*Таблица 7.1*

| Мнемоника константы | Значение константы | | Мнемоника константы | Значение константы | |
|---|---|---|---|---|---|
| | Десятичное | Двоичное | | Десятичное | Двоичное |
| KP0 | 0 | **0**.000 | KM1 | –0,125 | **1**.111 |
| KP1 | +0,125 | **0**.001 | KM2 | –0,250 | **1**.110 |
| KP2 | +0,250 | **0**.010 | KM3 | –0,375 | **1**.101 |
| KP3 | +0,375 | **0**.011 | KM4 | –0,500 | **1**.100 |
| KP4 | +0,500 | **0**.100 | KM5 | –0,625 | **1**.011 |
| KP5 | +0,625 | **0**.101 | KM6 | –0,750 | **1**.010 |
| KP6 | +0,750 | **0**.110 | KM7 | –0,875 | **1**.001 |
| KP7 | +0,875 | **0**.111 | KM8 | –1,000 | **1**.000 |

Сверхоперативное ОЗУ снабжено двумя портами: A и B, через которые производится выдача операндов для их обработки в АЛУ. Результат выполненной в АЛУ операции передается в СОЗУ через порт B. Операнд, получаемый с порта B, подается на вход АЛУ непосредственно. Операнд, выдаваемый с порта A, поступает в АЛУ через масштабирующее устройство, в котором этот операнд умножается на $2^n$ путем выполнения сдвига на соответствующее число разрядов влево или вправо. Коэффициент $n$ имеет целочисленное значение в пределах –13...+2. Выбор одного из 16 значений этого коэффициента задается в команде.

АЛУ выполняет операции в модифицированном дополнительном коде. Старший разряд полученного из СОЗУ 25-разрядного операнда рассматривается как знаковый, остальные 24 разряда слова образуют дробную часть числа. В АЛУ знаковый разряд операнда дублируется в четырех знаковых разрядах для образования модифицированного кода и, таким образом, каждый операнд в АЛУ представляется в форме

$$\underline{\text{ssss}}.\, a_{-1}a_{-2} \ldots a_{-24},$$

где ssss — четыре знаковых разряда; $a_{-1} \ldots a_{-24}$ — разряды дробной части числа со значениями разрядных коэффициентов, соответственно равны-

ми $2^{-1}$ ... $2^{-24}$. Над такими 28-разрядными операндами в АЛУ выполняются операции.

Использование модифицированного кода позволяет обнаруживать переполнение разрядной сетки. При возникновении переполнения АЛУ выдает сигнал на выход $\overline{OF}$. При выполнении некоторых команд в случае переполнения положительное значение результата заменяется максимально возможным (т. е. значение **0001**. x...x заменяется на значение **0000**. 1...1) и отрицательное значение результата заменяется минимально возможным (т.е. значение **1110**. x...x заменяется на значение **1111**.0...0, представляющее собой дополнительный код числа –1). В АЛУ могут использоваться два режима работы: с описанной выше коррекцией результата при возникновении переполнения и без коррекции результата.

Память команд построена на перепрограммируемом ПЗУ емкостью 192 24-разрядных команды. Для сокращения числа выводов микросхемы для ввода данных при записи информации в ППЗУ используются четыре аналоговых выхода (со старшими номерами). Таким образом, ввод в ППЗУ одного 24-разрядного слова команды требует 6 циклов записи.

На рис. 7.2 приведена структура аналоговой части микросхемы. Входной мультиплексор обеспечивает выборку одного из четырех аналоговых каналов. Эта операция выполняется по команде IN(*k*), в которой *k* указывает номер коммутируемого аналогового канала. Из сигнала, выделенного с помощью мультиплексора канала, схема выборки и хранения выбирает отсчет, значение которого запоминается в виде напряжения на конденсаторе. Затем производится преобразование по-

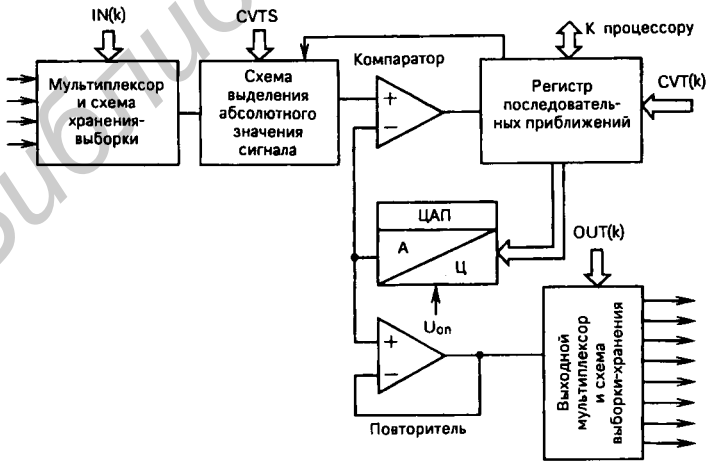



Рис. 7.2

лученного на конденсаторе напряжения в цифровую форму. По команде CVTS формируется знаковый разряд и выделяется абсолютное значение напряжения, подлежащего преобразованию. Абсолютное значение поступает в схему аналого-цифрового преобразования, работающую по методу последовательных приближений. Формирование каждого разряда цифрового представления абсолютного значения напряжения (начиная с его старшего разряда) производится по соответствующей команде CVT(*k*), где *k* — номер разряда. Допустимый диапазон входных сигналов задается внешним источником опорного напряжения. Необходимый при аналого-цифровом преобразовании ЦАП используется для преобразования выводимых данных из цифровой формы в аналоговую. При выводе содержимое регистра DAR (см. рис. 7.1) подвергается преобразованию в ЦАП и в аналоговой форме поступает в выходной мультиплексор. При поступлении команды вывода OUT(*k*) мультиплексор выдает преобразованное значение в *k*-й аналоговый канал.

Возможны прием и вывод сигналов в цифровой форме. Требуемый режим работы аналоговой части задается комбинацией напряжений на входах M1 и M2 микросхемы в соответствии с табл. 7.2.

*Таблица 7.2*

| Управляющие напряжения, В, на входах | | Режим ввода | Режим вывода |
|---|---|---|---|
| M1 | M2 | | |
| +5 | +5 | IN0...IN3 — аналоговые каналы | OUT0...OUT7 — аналоговые каналы |
| +5 | –5 | DI — цифровой последовательный канал | D0 — цифровой последовательный канал<br>OUT0...OUT3 — аналоговые каналы<br>OUT4...OUT7 — цифровые каналы |
| –5 | +5 | IN0...IN3 — аналоговые каналы | OUT0...OUT3 — цифровые каналы<br>OUT4...OUT7 — аналоговые каналы |
| –5 | –5 | IN0...IN3 — аналоговые каналы | OUT0...OUT7 — цифровые каналы |

В режиме M1 = +5 В, M2 = –5 В обеспечивается последовательный ввод и вывод цифровых данных через регистр DAR (см. рис. 7.1). В этом режиме для организации ввода—вывода используются четыре входных

канала. Через канал $\overline{DI}$ осуществляется ввод информации по командам CVTS и CVT($k$) в последовательности: знак, инверсные значения разрядов в последовательности от старшего ($k$ = 7) к младшему ($k$ = 0). Одновременно в процессе выполнения команды CVT по каналу $\overline{RD}$ выдаются стробирующие импульсы. При последовательном выводе канал $\overline{D0}$ используется для передачи информации и канал $\overline{RD}$ для выдачи стробирующих импульсов

### 7.3. СИСТЕМА КОМАНД

Команды имеют 24 разряда и представляются в следующем формате:

| КОпЦ (3 разряда) | Адрес операнда B (6 разрядов) | Адрес операнда A (6 разрядов) | Код сдвига (4 разряда) | КОпА (5 разрядов) |
|---|---|---|---|---|

Поле кода цифровой операции (КОпЦ) определяет вид выполняемой цифровой операции (операции, выполняемой в цифровой части с помощью АЛУ). Система цифровых операций и их представление в поле КОпЦ приведены в табл.7.3.

*Таблица 7.3*

| Цифровая операция | | Содержание операции |
|---|---|---|
| Мнемоника | Кодовая комбинация в поле КОпЦ | |
| ADD | 110 | $B + 2^n \cdot A \to B$ |
| SUB | 101 | $B - 2^n \cdot A \to B$ |
| LDA | 111 | $2^n \cdot A \to B$ |
| XOR | 000 | $B \oplus 2^n \cdot A \to B$ |
| ABS | 011 | $\lvert 2^n \cdot A \rvert \to B$ |
| ABA | 100 | $B + \lvert 2^n \cdot A \rvert \to B$ |
| LIM | 010 | $+1 \to B$, если $A \geq 0$<br>$-1 \to B$, если $A < 0$ |
| AND | 001 | $B \wedge 2^n \cdot A \to B$ |

В табл.7.4 приведена группа условных команд. Эти команды в поле кода аналоговой операции (КОпА) записываются с помощью мнемоники CND; CND может иметь вид CNDS, что означает проверку содержимого знакового разряда регистра DAR, или вид CND($k$), что означает

проверку $k$-го разряда содержимого регистра DAR. Сказанное относится к условным командам LDA CND и ADD CND.

*Таблица 7.4*

| Условная команда | | Содержание операции |
|---|---|---|
| КОпЦ | КОпА | |
| ADD | CND($k$) | $B + 2^n \cdot A \rightarrow B$, если $k$-й разряд DAR равен 1<br>$B \rightarrow B$, если $k$-й разряд DAR равен 0 |
| SUB | CND($k$) | $B - 2^n \cdot A \rightarrow B$, если предыдущий перенос $CY_p = 1$<br>$B + 2^n \cdot A \rightarrow B$, если предыдущий перенос $CY_p = 0$ |
| LDA | CND($k$) | $\lvert 2^n \cdot A \rvert \rightarrow B$, если $k$-й разряд DAR равен 1<br>$B \rightarrow B$, если $k$-й разряд DAR равен 0 |
| ABA | CND($k$) | $2^n \cdot A + B \rightarrow B$, установка режима без коррекции |
| XOP | CND($k$) | $B \oplus 2^n \cdot A \rightarrow B$, установка режима с коррекцией |
| RNZ | CND($k$) | Возврат на начало программы, если $k$-й разряд DAR = 1; продолжение программы, если $k$-й разряд DAR равен 0 |
| JNZ | CND($k$) | Переход на 32 команды вперед, если $k$-й разряд DAR = 1; продолжение программы, если $k$-й разряд DAR равен 0 |

В случае команды SUB CND($k$) проверяется значение хранящегося в АЛУ переноса в знаковый разряд CY при выполнении предыдущей команды и записывается перенос CY, возникающий при выполнении данной команды, в $k$-й разряд регистра DAR. В случае SUB CNDS инверсное значение переноса CY, возникающего в текущей команде, записывается в знаковый разряд регистра DAR.

Команда XOR CND обеспечивает в последующих командах выполнение коррекции результата в АЛУ при возникновении переполнения разрядной сетки. Команда ABA CND отменяет выполнение коррекции при переполнении АЛУ в последующих командах.

Команда RNZ CND обеспечивает условный переход в начало программы по значению проверяемого разряда регистра DAR: возврат в начало программы, если значение $k$-го разряда регистра DAR равно 1, и продолжение программы, если значение $k$-го разряда регистра DAR равно 0.

Команда JNZ производит переход на 32 команды вперед, если проверяемый $k$-й разряд регистра DAR равен 1. Эти команды имеют следующие особенности. Команда RNZ должна размещаться в памяти программ в ячейке с адресом, делящимся на четыре; при этом возврат в начало программы может происходить после выполнения остальных

трех команд четверки. Команда JNZ должна занимать третий адрес в четверке адресов; переход на 32 команды вперед может происходить после выполнения следующих пяти команд.

Поле КОпА определяет операции в аналоговой части схемы (табл. 7.5). Рассмотрим выполнение этих операций.

*Таблица 7.5*

| Аналоговая операция | | | Содержание операции |
|---|---|---|---|
| Мнемоника | Кодовая комбинация в поле КОпА | | |
| | Функция | Значение $k$ | |
| IN( $k$) | 00<br>00<br>00<br>00 | 000 (k = 0)<br>001 ($k$ = 1)<br>010 ($k$ = 2)<br>011 ($k$ = 3) | Ввод по каналу $k$ |
| NOR | 00 | 100 | Отсутствие операции |
| EOP | 00 | 101 | Возврат в ППЗУ к ячейке 0 |
| CVTS | 00 | 110 | Преобразование знакового разряда |
| CNDS | 00 | 111 | Условная операция по знаковому разряду в DAR |
| CVT( $k$) | 10<br>...<br>10 | 000 ($k$ = 0)<br>...<br>111 ($k$ = 7) | АЦП $k$-го разряда |
| OUT( $k$) | 01<br>...<br>01 | 000 ($k$ = 0)<br>...<br>111 ($k$ = 7) | Вывод через канал $k$ |
| CND( $k$) | 11<br>...<br>11 | 000 ($k$ = 0)<br>...<br>111 ($k$ = 7) | Условная операция по $k$ -му разряду регистра DAR |

В команде ввода IN($k$) $k$ определяет номер одного из четырех входных аналоговых каналов. Для фиксации значения сигнала выбранного канала на конденсаторе схемы выборки и хранения с погрешностью, не большей 0,5 весового коэффициента младшего разряда, требуется 4,8 мкс. Поэтому при длительности цикла выполнения команд 0,6 мкс предусматривается восемь следующих подряд команд IN($k$). Полученное на конденсаторе напряжение далее подвергается преобразованию в цифровую форму методом последовательного приближения при котором последовательно формируются разряды цифрового представления отсче-

та сигнала. Для получения каждого разряда числа после команды CVT используется не менее двух команд NOR ("Отсутствие операции"). Это необходимо, чтобы обеспечить паузу длительностью 1,2 мкс для установления процессов в ЦАП после преобразования каждого разряда. Результат преобразования фиксируется в регистре DAR.

В табл.7.6 показана последовательность команд для ввода и преобразования в цифровую форму отсчета аналогового сигнала *k*-го входного канала. Очевидно, если требуется производить преобразование в цифровую форму с меньшей точностью, т.е. с числом разрядов, меньшим девяти, то не понадобится получать в DAR значения младших разрядов и количество команд, необходимых для ввода данных, сократится.

*Таблица 7.6*

| Номер команды | Содержимое поля | Номер команды | Содержимое поля | Номер команды | Содержимое поля | Номер команды | Содержимое поля |
|---|---|---|---|---|---|---|---|
| 0 | IN(*k*) | 9 | CVTS | 18 | CVT5 | 27 | CVT2 |
| 1 | IN(*k*) | 10 | NOP | 19 | NOP | 28 | NOP |
| 2 | IN(*k*) | 11 | NOP | 20 | NOP | 29 | NOP |
| 3 | IN(*k*) | 12 | CVT7 | 21 | CVT4 | 30 | CVT1 |
| 4 | IN(*k*) | 13 | NOP | 22 | NOP | 31 | NOP |
| 5 | IN(*k*) | 14 | NOP | 23 | NOP | 32 | NOP |
| 6 | IN(*k*) | 15 | CVT6 | 24 | CVT3 | 33 | CVT0 |
| 7 | IN(*k*) | 16 | NOP | 25 | NOP | | |
| 8 | NOP | 17 | NOP | 26 | NOP | | |

В команде вывода OUT(*k*) *k* — номер одного из каналов, используемого для вывода результата преобразования содержимого регистра DAR в аналоговую форму. После пересылки выводимого числа в регистр DAR следует в трех следующих командах предусмотреть операцию NOP (на время установления выходной схемы выборки и хранения) и далее в семи— восьми командах — операцию OUT(*k*).

Команда NOP указывает на отсутствие аналоговой операции.

Команда EOP является последней в программе. Она помещается в ячейку памяти, адрес которой делится на четыре. После этой команды выполняются следующие три команды четверки и производится передача управления из конца программы на начальную команду программы, хранящуюся в ячейке с адресом 0 памяти программ.

Команды CNDS и CND($k$) используются в условных командах (их описание приведено выше).

### 7.4. ПРИЕМЫ ПРОГРАММИРОВАНИЯ

Рассмотрим пример моделирования цепи второго порядка, приведенной на рис.7.3. Построим дифференциальное уравнение, описывающее процессы в приведенной цепи. Выразим ток и напряжения на элементах цепи через напряжение $u_C$ на емкостном элементе:

ток в контуре $i = Cdu_C/dt$;

напряжение на резистивном элементе $u_r = ir = rCdu_C/dt$;

напряжение на индуктивном элементе $u_L = Ldi/dt = LCd^2u_C/dt^2$.

Так как $u_L + u_r + u_C = u_{вх}$, то, подставив соответствующие напряжениям $u_L$ и $u_r$ выражения, получим

$$LC\frac{d^2u_C}{dt^2} + rC\frac{du_C}{dt} + u_C = u_{вх}.$$

Перейдем в данном выражении от $u_C$ к $u_{вых} = \frac{1}{k}u_C$:

$$\frac{LC}{k}\frac{d^2u_{вых}}{dt^2} + \frac{rC}{k}\frac{du_{вых}}{dt} + \frac{1}{k}u_{вых} = u_{вх}$$

или

$$\frac{d^2u_{вых}}{dt^2} + \frac{r}{L}\frac{du_{вых}}{dt} + \frac{1}{LC}u_{вых} = \frac{k}{LC}u_{вх}.$$

Заменив $u_{вых}$ на $y$ и $u_{вх}$ на $x$, запишем дифференциальное уравнение в виде

$$y'' + \frac{r}{L}y' + \frac{1}{LC}y = \frac{k}{LC}x, \qquad (7.1)$$

где $y'$ и $y''$ — соответственно первая и вторая производные по времени.

При интегрировании такого уравнения с помощью цифровой ЭВМ переменные $y$ и $x$ и производные $y'$, $y''$ должны представляться для дискретных моментов времени и в цифровой форме. На рис. 7.4,*а* показаны дискретные моменты времени $t_1, t_2, t_3, \ldots$ и соответствующие этим моментам времени значения переменных $x$ и $y$, которые с помощью АЦП

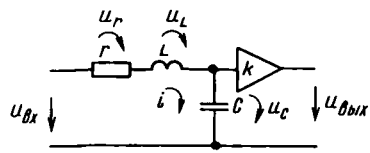



Рис.7.3

могут быть преобразованы в цифровую форму.

Рассмотрим, как могут быть представлены производные $y'$ и $y''$. Через значения $y_{n-1}$ и $y_n$ для моментов $t_{n-1}$ и $t_n$ (рис. 7.4,*б*) можно приближенно выразить производную $y'_n$ для момента $t_n$:

$$y'_n \approx (y_n - y_{n-1})/T$$

и через значения $y_{n-2}$ и $y_{n-1}$ — производную $y'_{n-1}$:

$$y'_{n-1} \approx (y_{n-1} - y_{n-2})/T.$$

Аналогично вторую производную можно представить выражением

$$y''_n \approx (y'_n - y'_{n-1}) / T.$$

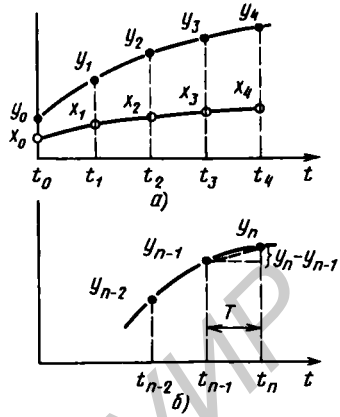



Рис.7.4

После подстановки выражений $y'_n$ и $y'_{n-1}$ имеем

$$y''_n = (y_n - 2y_{n-1} + y_{n-2})/T^2.$$

Полученные приближенные выражения $y'_n$ и $y''_n$ подставим в дифференциальное уравнение (7.1):

$$\frac{y_n - 2y_{n-1} + y_{n-2}}{T^2} + \frac{r}{L}\frac{y_n - y_{n-1}}{T} + \frac{1}{LC}y_n = \frac{k}{LC}x_n.$$

Разрешим полученное выражение относительно $y_n$:

$$y_n = \frac{k}{LC\left(\frac{1}{T^2} + \frac{r}{LT} + \frac{1}{LC}\right)}x_n + \left(\frac{2}{T^2} + \frac{r}{LT}\right)\frac{1}{\frac{1}{T^2} + \frac{r}{LT} + \frac{1}{LC}} \times$$

$$\times y_{n-1} - \frac{1}{T^2\left(\frac{1}{T^2} + \frac{r}{LT} + \frac{1}{LC}\right)}y_{n-2} \qquad (7.2)$$

или

$$y_n = ax_n + b_1 y_{n-1} + b_n y_{n-2}, \qquad (7.3)$$

где $a$, $b_1$, $b_2$ — значения коэффициентов в (7.2) перед соответствующими переменными.

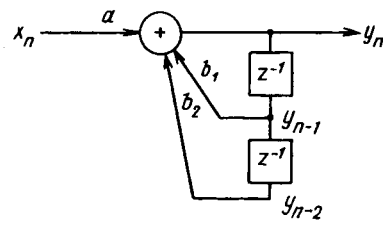



Рис. 7.5

Выражение (7.3) называется разностным уравнением. Его решение в ЭВМ сводится к следующей последовательности действий. Подставим начальные значения переменных в правую часть уравнения в (7.3). Вычислив правую часть уравнения, находим значение $y_n$. Для следующего момента времени $t_{n+1}$ выражение разностного уравнения получается увеличением на единицу индексов при всех переменных:

$$y_{n+1} = ax_{n+1} + b_1 y_n + b_2 y_{n-1}.$$

Отсюда следует, что для вычисления $y_{n+1}$ достаточно ранее найденные $y_n$ и $y_{n-1}$ подставить в разностное уравнение (7.3) вместо $y_{n-1}$ и $y_{n-2}$ и повторить вычисление правой части при новом значении входной переменной $x - x_{n+1}$ и т. д. В результате многократного повторения этих действий мы получим последовательность значений $y_0, y_1, y_2, \ldots$, соответствующих дискретным значениям выходного напряжения в схеме рис. 7.3.

Последовательность вычислений для подобных фильтров удобно представлять в виде схемы, показанной на рис.7.5. Здесь над цепями, подходящими к суммирующему элементу, показаны коэффициенты $a$, $b_1$, $b_2$, с которыми производится суммирование соответствующих переменных; $z^{-1}$ — элемент задержки на один тактовый интервал времени $T$. Легко обнаружить соответствие между данной схемой и решением разностного уравнения (7.3).

Все рассмотренное относилось к полосовому фильтру, представленному схемой рис. 7.3. Очевидно, фильтры других типов и высоких порядков описываются соответствующими разностными уравнениями и их вычисление может быть представлено схемами, подобными приведенной на рис. 7.5.

Вернемся к расчету разностного уравнения (7.3). Зададимся значениями коэффициентов, представив их в двоичной знакоразрядной форме: $a = 0{,}00293 = 2^{-8} - 2^{-10}$; $b_1 = 1{,}7656 = 2^1 - 2^{-2} + 2^{-6}$; $b_2 = -0{,}99414 = -2^0 + 2^{-7} - 2^{-9}$.

Составим программу реализации данного фильтра на КМ1813ВЕ1 (табл. 7.7) . Здесь для переменных $y_n, y_{n-1}, y_{n-2}, x_n$, входящих в разностное уравнение (7.3), введены мнемонические обозначения Y0, Y1, Y2, X, используемые также в качестве мнемонических обозначений адресов ячеек СОЗУ, в которых хранятся значения этих переменных. Таким образом, разностное уравнение, представленное через эти имена переменных, будет иметь вид

$$Y0 = a \cdot X + b_1 \cdot Y1 + b_2 \cdot Y2. \qquad (7.4)$$

*Таблица 7.7*

| Номер команды | Команда | | | | | Выполняемые действия |
|---|---|---|---|---|---|---|
| | КОпЦ | Адрес B | Адрес A | Код сдвига | КОпА | |
| 0 | LDA | Y0 | Y1 | L01 | IN2 | $Y0 \leftarrow 2^1 \cdot Y1$ |
| 1 | SUB | Y0 | Y1 | R02 | IN2 | $Y0 \leftarrow 2^1 \cdot Y1 - 2^{-2} \cdot Y1$ |
| 2 | ADD | Y0 | Y1 | R06 | IN2 | $Y0 \leftarrow 2^1 \cdot Y1 - 2^{-2} \cdot Y1 + +2^{-6} \cdot Y1 = b_1 \cdot Y1$ |
| 3 | SUB | Y0 | Y2 | R00 | IN2 | $Y0 + b_1 \cdot Y1 - 2^0 \cdot Y2$ |
| 4 | ADD | Y0 | Y2 | R07 | IN2 | $Y0 \leftarrow b_1 \cdot Y1 - 2^0 \cdot Y2 + 2^{-7} \cdot Y2$ |
| 5 | SUB | Y0 | Y2 | R09 | IN2 | $Y0 \leftarrow b_1 \cdot Y1 + b_2 \cdot Y2$ |
| 6 | LDA | Y0 | Y0 | R00 | IN2 | $Y0 \leftarrow Y0$ |
| 7 | | | | | IN2 | |
| 8 | | | | | CVTS | |
| 9 | | | | | NOP | |
| 10 | | | | | NOP | |
| 11 | | | | | CVT7 | |
| 12 | | | | | NOP | |
| 13 | | | | | NOP | |
| 14 | | | | | CVT6 | |
| ⋮ | | | | | ⋮ | |
| 33 | | | | | CVT0 | |
| 34 | ADD | Y0 | DAR | R08 | NOP | $Y0 \leftarrow b_1 \cdot Y1 + b_2 \cdot Y2 + 2^{-8}X$ |
| 35 | SUB | Y0 | DAR | R10 | NOP | $Y0 \leftarrow b_1 \cdot Y1 + b_2 \cdot Y2 + aX$ |
| 36 | LDA | DAR | Y0 | R00 | NOP | $DAR \leftarrow Y0$ |
| 37 | LDA | Y2 | Y1 | R00 | NOP | $Y2 \leftarrow Y1$ |
| 38 | LDA | Y1 | Y0 | R00 | NOP | $Y1 \leftarrow Y0$ |
| 39 | LDA | Y0 | Y0 | R00 | NOP | $Y0 \leftarrow Y0$ |
| 40 | | | | | OUT3 | |
| ⋮ | | | | | ⋮ | |
| 46 | | | | | OUT3 | |
| 47 | | | | | OUT3 | |
| 48 | | | | | EOP | |

Примем, что входная величина X вводится в микросхему через ее входной аналоговый канал *2*. В первых 34 командах (0...33) предусматриваются прием и преобразование в цифровую форму отсчета входной величины. В это время в цифровой части командами (0...5) производится вычисление значения $b_1 \cdot Y1 + b_2 \cdot Y2$. Вычисление этого выражения завершается командой 5. Последующие команды до команды 33 —"холостые" команды Y0 ← Y0, не изменяющие содержимое ячеек памяти и обеспечивающие задержку до момента получения в DAR значения входной величины в цифровой форме. Очевидно, что в этой части программы (до команды 33) можно было обработать более сложное выражение, описывающее фильтры более высокого порядка.

Команды 34 и 35 завершают вычисление значения $Y0 = a \cdot X + + b_1 \cdot Y1 + b_2 \cdot Y2$. Команда 36 передает вычисленное значение Y0 в регистр DAR, откуда оно после преобразования в аналоговую форму будет выдано на выход 3. Команды 37, 38, 39 обеспечивают задержку, необходимую для установления процессов в ЦАП. Последующие команды 40 — 47 осуществляют вывод рассчитанного значения Y0 в аналоговой форме. В это время в цифровой части командами 37 и 38 производится подготовка к повторению цикла путем передач Y2 ← Y1, Y1 ← Y0.

Если в цикле не предусматривается никаких других действий, то последней в программе должна быть команда, содержащая в поле КОпА команду EOP возврата в начало программы.

Таким образом, для реализации фильтра (рассмотренного ранее, а также и фильтра более высокого порядка) может потребоваться 48 команд. Отсюда время $T_{отс}$ между отсчетами входной величины при длительности машинного цикла (времени выполнения одной команды) 0,6 мкс составит $T_{отс} = 51 \cdot 0,6 = 30,6$ мкс. Следовательно, по теореме Котельникова спектр входного сигнала должен быть ограничен частотой $F_{max} = 0,5/T_{отс} = 0,5/(30,6 \cdot 10^{-6}) \approx 15$ кГц.

В памяти программ микросхемы КМ1813ВЕ1, содержащей 192 ячейки, можно записать программы четырех фильтров, обрабатывающих четыре входных сигнала. В этом случае программа состоит из четырех частей. Первая часть соответствует приему сигнала из первого входного канала и выдаче его после обработки на один из восьми выходных каналов, вторая — приему сигнала из второго канала и выдаче результата обработки на другой выходной канал и т.д. После выдачи результата обработки сигнала четвертого канала происходят переход в начало программы и повторение описанного процесса. При этом период отсчетов сигналов каждого из каналов возрастает в четыре раза по сравнению с ранее найденным значением и спектр входного сигнала должен быть уже в четыре раза.

## 7.5. ПРИМЕНЕНИЕ МИКРОСХЕМЫ КМ1813ВЕ1 ДЛЯ ЦИФРОВОГО МОДЕЛИРОВАНИЯ УЗЛОВ АППАРАТУРЫ

На одной микросхеме КМ1813ВЕ1 можно реализовать анализатор спектра низкочастотных сигналов (рис. 7.6). Принцип его работы заключается в следующем.

Пилообразное напряжение, выдаваемое соответствующим генератором, подается на вход осциллографа "Горизонтальное отклонение". Под действием этого напряжения луч на экране осциллографа периодически совершает равномерное смещение в горизонтальном направлении. Одновременно по закону пилообразного напряжения происходит изменение частоты колебаний, близких по форме к гармоническому колебанию. В момент, когда в преобразователе частоты разность частот этих колебаний и компоненты в спектре входного сигнала будет равна промежуточной частоте, возникает сигнал на выходе узкополосного фильтра промежуточной частоты. А после выпрямления и сглаживания в фильтре нижних частот образуется всплеск напряжения, который, действуя на входе "Вертикальное отклонение" осциллографа, вызовет на экране выброс луча вверх. Положение выброса будет соответствовать частоте компоненты входного сигнала, размер выброса — амплитуде компоненты. Таким образом, на экране вычерчивается спектр входного сигнала.

Все узлы данной схемы могут быть промоделированы на одной микросхеме КМ1813ВЕ1. Для реализации функций узлов схемы требуется 95 команд. Следовательно, программа действительно реализуема на КМ1813ВЕ1, имеющей память емкостью 192 команд. При 95 командах и длительности выполнения команды 0,6 мкс период отсчетов входного сигнала $T_{отс} = 95 \cdot 0{,}6 = 57$ мкс и максимальная частота спектра входного сигнала $F_{max} = 0{,}5/T_{отс} = 8{,}8$ кГц.

Ниже рассматриваются программы, реализующие функции отдельных узлов анализатора спектра.

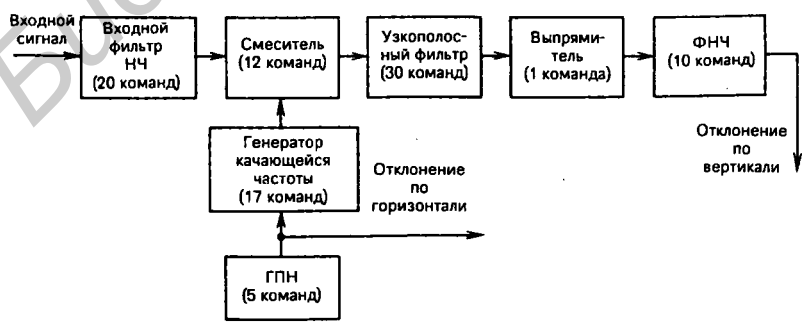



Рис. 7.6

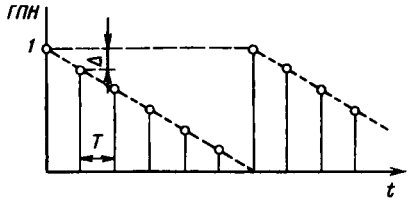



Рис. 7.7

**Генератор пилообразных колебаний.** Пилообразное напряжение моделируется последовательностью значений для моментов $t_0$, $t_1$, $t_2$, ... , следующих с периодом $T$ и совпадающих со значениями пилообразного напряжения в соответствующие моменты времени (рис. 7.7). Очевидно, для любой пары таких значений разность равна некоторой константе Δ. Пусть требуется за период пилообразного напряжения иметь 200 отсчетов, т.е. Δ = 0,005.

Рассмотрим определение значения Δ. Его можно выразить, подобрав константы, приведенные в табл. 7.1: $0{,}005 = 0{,}375 \cdot 2^{-6} - 0{,}875 \cdot 2^{-10} + 0{,}000004882$. Отбрасывая член 0,000004882, можно считать, что значение $0{,}375 \cdot 2^{-6} - 0{,}875 \cdot 2^{-10}$ является достаточно хорошим приближением к требуемому значению Δ = 0,005. Здесь 0,375 и 0,875 — константы, берущиеся из ячеек СОЗУ с мнемоническими именами КР3 и КР7.

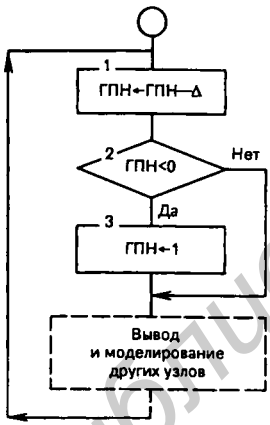



Рис. 7.8

Схема алгоритма формирования пилообразного колебания представлена на рис. 7.8. Здесь для значений ординат этих колебаний использовано обозначение ГПН. В блоке 1 производится вычитание значения Δ из ГПН и затем в блоке 2 — проверка возникновения отрицательного значения. В случае, если полученное значение отрицательно, производится восстановление начального значения ГПН = 1.

Далее предусмотрен вывод полученного значения ГПН. Если генератор должен использоваться в сложной схеме, как это имеет место в рассмотренном выше анализаторе спектра, то в схеме алгоритма после блока вывода (или вместо этого блока) необходимо отразить действия остальных узлов схемы на рис. 7.6.

Программа, соответствующая схеме алгоритма на рис. 7.8, имеет вид

| КОпЦ | В | А | *n* | КОпА |
|---|---|---|---|---|
| SUB | ГПН | КР3 | R06 | NOP |
| ADD | ГПН | КР7 | R10 | NOP |
| LDA | DAR | ГПН | R00 | NOP |
| LDA | ГПН | КР4 | L01 | CNDS |
| ... | ... | ... | ... | ... |

**Генератор гармонических колебаний**. На рис. 7.9,*а* приведена схема формирования колебаний, близких к гармоническому колебанию. Генератор пилообразных колебаний (ГПН) задает период колебаний. Пилообразные колебания генератора далее преобразуются в блоках 2 — 5. В последнем блоке при возникновении значений, выходящих за пределы –1 ... +1, происходит ограничение. Из временной диаграммы на рис. 7.9,*а* видно, что полученное колебание достаточно близко по форме к гармоническому. Если бы потребовалось лучшее приближение к гармоническому колебанию, можно было бы полученное колебание пропустить через полосовой фильтр. Программа, соответствующая схеме алгоритма, имеет вид

```
LDA   Y   ГПН   R00   NOP   ; пила от +1 до 0
SUB   Y   КР4   R00   NOP   ; пила от +0,5 до –0,5
ABS   Y   Y     L01   NOP   ; треугольник от 0 до 1
SUB   Y   КР4   R00   NOP   ; треугольник от –0,5 до +0,5
ADD   Y   Y     L01   NOP   ; срезанный треугольник
```

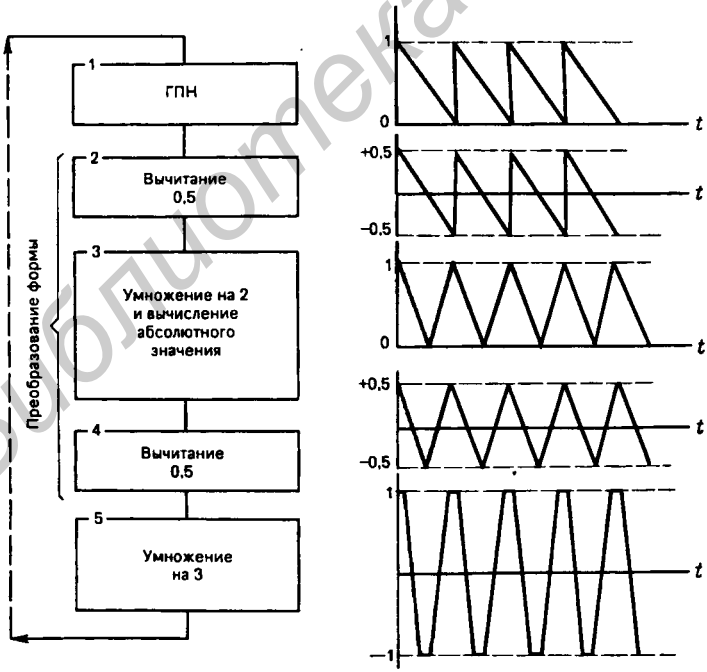



Рис. 7.9

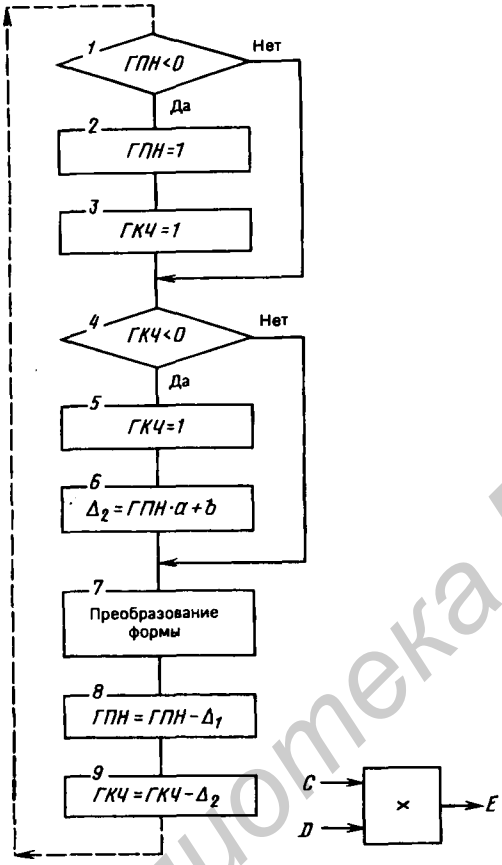



Рис. 7.10




Рис. 7.11

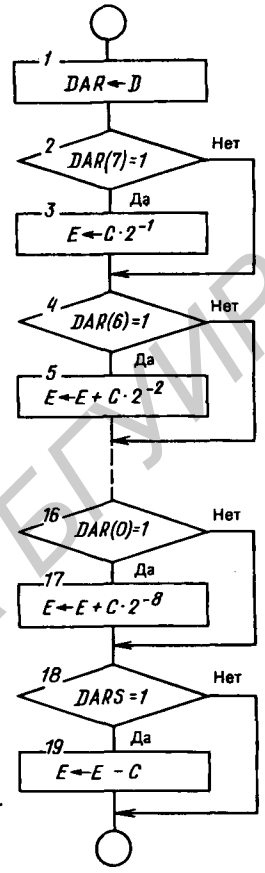



Рис. 7.12

**Генератор качающейся частоты**. Рассмотрим генератор, формирующий колебания, по форме близкие к гармоническому с частотой, меняющейся по пилообразному закону. На рис. 7.10 представлена схема алгоритма такого генератора. Здесь использованы следующие обозначения: ГПН — генератор пилообразного напряжения низкой частоты, задающий закон изменения частоты; ГКЧ — генератор пилообразного напряжения, задающий период модулируемых колебаний высокой частоты. При этом под напряжением следует понимать его цифровое представление.

В блоке 6 вычисляется $\Delta_2$, определяющее период ГКЧ. Так как $\Delta_2$ линейно зависит от напряжения ГПН (*a*, *b* — константы), то период (а следовательно, и частота) ГКЧ изменяется по закону напряжения ГПН. Преобразователь формы преобразует пилообразное напряжение ГКЧ в колебание, близкое по форме к гармоническому.

**Умножитель**. Показанный на рис. 7.11 умножитель формирует на выходе значение E, равное произведению входных чисел C и D.

На рис. 7.12,*а* приведена схема алгоритма умножения. Множитель помещается в регистр DAR. Последовательно анализируется содержимое разрядов этого регистра. При обнаружении единицы в *k*-м разряде регистра DAR к E добавляется множимое, умноженное на весовой коэффициент *k*-го разряда. Это умножение осуществляется путем арифметического сдвига на соответствующее число разрядов множимого C. Участвующие в умножении числа представляются в дополнительном коде, в дополнительном коде получается и результат операции. По правилам умножения чисел в дополнительном коде при обнаружении единицы в знаковом разряде множителя D из произведения модулей перемножаемых чисел следует вычесть множимое.

Так как нет команды, которая по условию DARS = 1 выполняла бы операцию E ← E – C, то в программе, соответствующей схеме алгоритма, реализация этой операции потребовала трех последних команд: ячейка F сбрасывается в нуль (операция F ← F – F), из содержимого F вычитается множимое C (операция F ← F – C) и, таким образом, в ячейке F образуется значение –C, затем при DARS = 1 к содержимому E прибавляется F (операция E ← E + F, если DARS = 1).

Программа, соответствующая схеме алгоритма:

| | | | | |
|---|---|---|---|---|
| LDA | DAR | D | R00 | NOP |
| LDA | E | C | R01 | CND(7) |
| ADD | E | C | R02 | CND(6) |
| ... | ... | ... | ... | ... |
| ADD | E | C | R08 | CND(0) |
| LDA | F | KP0 | R00 | NOP |
| SUB | F | C | R00 | NOP |
| ADD | E | F | R00 | CNDS |

## *Упражнения*

1. Составьте программы, обеспечивающие реализацию фильтра 2-го порядка для аналоговых сигналов.

Структура фильтра приведена на рис. 7.13. Этой схеме соответствует разностное уравнение

$$y_n = s(b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2}) - a_1 y_{n-1} - a_2 y_{n-2}.$$

Значение коэффициентов фильтра для восьми вариантов приведены в табл.7.8.

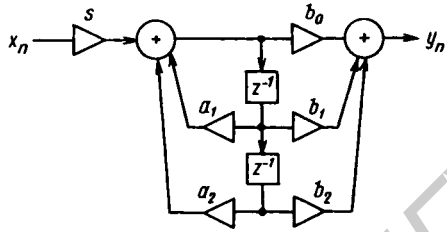



Рис. 7.13

*Таблица 7.8*

| Номер варианта | Тип фильтра | Значения коэффициентов | | | | | |
|---|---|---|---|---|---|---|---|
| | | $s$ | $a_1$ | $a_2$ | $b_0$ | $b_1$ | $b_2$ |
| 1 | ФНЧ | 0,25 | 0,21875 | –0,640625 | 1,0 | 1,28125 | 1,0 |
| 2 | ФНЧ | 1,0 | –0,25 | 0,9375 | 1,0 | 0,5625 | 1,0 |
| 3 | ФВЧ | 0,125 | –0,5625 | –0,875 | 1,0 | 0,21875 | 1,0 |
| 4 | ФВЧ | 1,0 | –0,875 | –0,6875 | 2,0 | –1,75 | 2,0 |
| 5 | ПФ | 0,003906 | –0,679687 | –0,980469 | 1,0 | 0,0 | –1,0 |
| 6 | ПФ | 0,0625 | –0,597656 | –0,980469 | 1,0 | 0,0 | –1,0 |
| 7 | РФ | 0,003906 | –0,75 | 0,976562 | 1,0 | 0,0 | –1,0 |
| 8 | РФ | 0,0625 | –0,820312 | 0,976562 | 1,0 | 0,0 | –1,0 |

Примечания: ФНЧ — фильтр нижних частот; ФВЧ — фильтр верхних частот; ПФ — полосовой фильтр; РФ — режекторный фильтр.

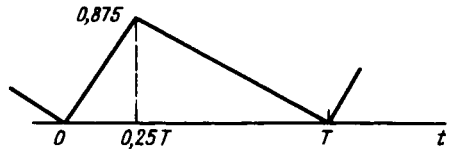
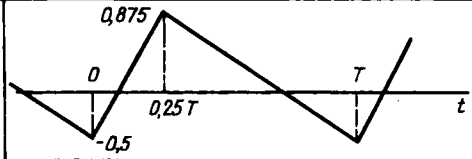
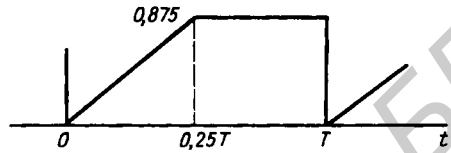
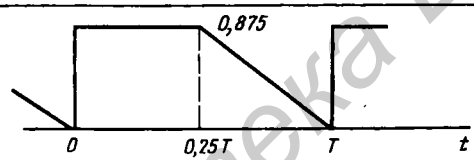
| Номер варианта | Форма генерируемого колебания | Число ординат за период T |
|---|---|---|
| 1 |  | 128 |
| 2 |  | 128 |
| 3 |  | 64 |
| 4 |  | 64 |

Рис. 7.14

2. Составьте программу формирования колебаний приведенных на рис. 7.14 форм.

## Список литературы


1. **Поспелов Д.А.** Логические методы анализа и синтеза схем. — М.: Энергия, 1968.

2. **Поспелов Д.А.** Арифметические основы вычислительных машин дискретного действия: Учеб. пособие. — М.: Высшая школа, 1970.

3. **Баранов С.И.** Синтез микропрограммных автоматов (граф-схемы и автоматы). — Л.: Энергия, 1979.

4. **Алексенко А.Г., Шагурин И.И.** Микросхемотехника: Учеб.пособие. — М.: Радио и связь, 1982.

5. **Преснухин Л.Н., Воробьев Н. В., Шишкевич А.А.** Расчет элементов цифровых устройств: Учеб. пособие. — М.: Высшая школа, 1982.

6. **Калабеков Б.А.** Микропроцессоры и их применение в системах передачи и обработки сигналов: Учеб. пособие. — М.: Радио и связь, 1988.

7. **Цифровые** и аналоговые интегральные микросхемы: Справочник / С.В. Якубовский и др.; Под ред. С.В. Якубовского. — М.: Радио и связь, 1989.

8. **Микропроцессоры** и микропроцессорные комплекты интегральных микросхем: Справочник: В 2 т. / Н. Н. Аверьянов и др.; Под ред. В. А. Шахнова. — М.: Радио и связь, 1988.

9. **Однокристальные** микроЭВМ: Справочник / А. В. Боборыкин, Г. П. Липовецкий, Г. В. Литвинский и др. — М.: МИКАП, 1994.

## Оглавление