Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

И. П. Кобяк

ОСНОВЫ ПРОЕКТИРОВАНИЯ КОМПЬЮТЕРНЫХ УСТРОЙСТВ

Учебно-методическое пособие
по курсу «Структурная и функциональная организация ЭВМ»
для студентов специальности I-40 02 01 «Вычислительные машины, системы и сети» заочной формы обучения

Минск 2008

УДК 004.3 (075.8)
ББК 32.973-26-04 я73
К 55

Р е ц е н з е н т
доц. кафедры ПОИТ, канд. техн. наук П. Ю. Бранцевич

Кобяк, И. П.

К 55 Основы проектирования компьютерных устройств : учеб.-метод. пособие по курсу «Структурная и функциональная организация ЭВМ» для студ. спец. I-40 02 01 «Вычислительные машины, системы и сети» заоч. формы обуч. / И. П. Кобяк. – Минск : БГУИР, 2008. – 78 с. : ил.

ISBN 978-985-488-302-1

Основное назначение представленных материалов – это электронная поддержка курса лекций при изучении дисциплин, связанных с проектированием неймановских компьютеров.

Издание является продолжением методического пособия «Процессоры компьютерных устройств. Синтез операционных автоматов», автор И. П. Кобяк, выпущенного БГУИР в 2003 г.

УДК 004.3 (075.8)
ББК 32.973.26-04 я73

ISBN 978-985-488-302-1

ОГЛАВЛЕНИЕ

1. ОРГАНИЗАЦИЯ ПАМЯТИ КОМПЬЮТЕРНЫХ СИСТЕМ

1.1. Общие сведения

Запоминающее устройство (ЗУ), или память, – это часть компьютера, предназначенная для записи, хранения и выдачи информации, представленной в цифровом виде. Иногда говорят, что с помощью ЗУ информация в ЭВМ передается во времени, причем направление этого движения совпадает с направлением движения реального времени.

Структурная организация запоминающего устройства в общем случае определяет способ записи или извлечения информации из него. Обычно ячейка ЗУ хранит набор двоичных данных фиксированной длины, а вся схема устройства может быть представлена в виде некоторого матричного пространства. Идентификация строк в этой структуре осуществляется с помощью номеров или адресов памяти. В более сложных устройствах при обращении требуется выполнять извлечение слова, расположенного в узле матрицы. При этом необходимо формировать как адреса строк, так и адреса столбцов массива. Стоимость проектирования и изготовления таких устройств на практике оказывается также более высокой, чем схем предыдущего вида.

Существуют различные типы ЗУ, отличающиеся друг от друга способом записи и извлечения информации. С этой точки зрения запоминающие устройства делятся на две большие группы: ЗУ с произвольным доступом или выборкой (ЗУПВ) и ЗУ с последовательным доступом. К первому типу относят схемы, в которых доступ к любому слову требует примерно одного и того же времени, то есть можно наугад выбрать строку матрицы, и этот принцип выбора не отразится на времени, которое затрачивается на чтение или запись. Ко второму типу относят устройства, доступ к которым возможен лишь в определенном порядке. При этом могут быть реализованы различные режимы функционирования. Техническая реализация этих режимов, как правило, позволяет ускорить вычислительный процесс или принципиально организовать эффективные вычисления.

Учитывая практическую значимость, материал данной главы содержит описание принципов организации и функционирования ЗУПВ, стекового и ассоциативного (АЗУ) запоминающих устройств, способов защиты памяти, а также описание принципов организации виртуальных устройств и страничносегментной организации памяти.

1.2. Организация ЗУ с произвольным доступом

В общем случае ЗУ с произвольным доступом строится с использованием нескольких блоков или модулей. Для современных компьютеров выпускаемые

модули обычно реализуются в виде отдельных интегральных БИС или СБИС со статической или динамической записью данных. Состав и функции внешних сигнальных линий микросхем при этом выбираются с таким расчетом, чтобы по возможности обеспечивалось удобство проектирования и простота работы ЗУ в системе с шинной организацией связей. В число таких линий входят: 1) линии для задания адреса слова, к которому производится обращение, 2) линии, по которым осуществляется передача данных из модуля или в модуль, 3) управляющие линии, позволяющие задать нужную операцию, например чтение или запись.

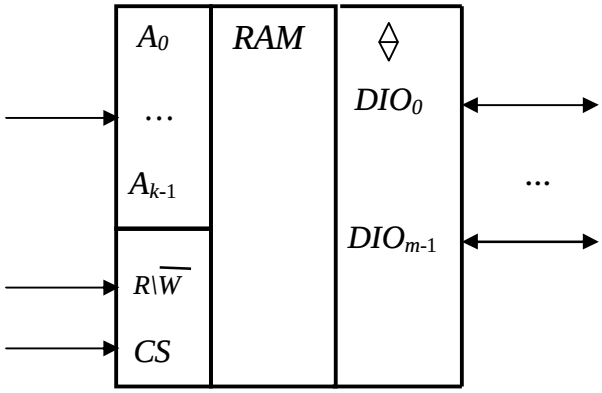


Рис. 1.1

Интегральная схема памяти, показанная на рис. 1.1, содержит адресные линии $A_0 - A_{k-1}$, образующие шину адреса блока, линии $DIO_0 - DIO_{m-1}$, образующие двунаправленную шину данных, а также линии $R\backslash\overline{W}$ (*чтение* \ *запись*), и *CS*-разрешение выборки модуля. Два последних вывода образуют (при наличии системы) часть шины управления компьютерного устройства.

Модуль памяти находится в режиме чтения, если сигнал $R\backslash\overline{W}$ равен логической единице, в противном случае имеет место режим записи. Линия *CS* также может находиться в состоянии 0 или 1. Это дает возможность обратиться к схеме ЗУ при наличии на данном входе потенциала логической единицы. При $CS = 0$ выходы $DIO_0 - DIO_{m-1}$ находятся в выключенном (третьем) состоянии или R_{off}. Если $CS = 1$, на шине данных организуется рабочий режим.

Существует довольно много разновидностей ИМС с набором внешних сигналов, соответствующим различным типам шин. Однако большинство модулей можно отнести к одной из двух основных моделей:

1) модули с одним комплектом линий данных, по которым циркулирует входная и выходная информация при записи или чтении (см. рис. 1.1);

2) модули с разделенной шиной для входной и выходной информации.

Выберем в качестве базовой ячейки ОЗУ модуль с одним комплектом линий данных и в случае необходимости будем использовать его при проектировании системы памяти.

1.2.1. Внутренняя организация ЗУ с произвольным доступом

Рассмотрев внешние характеристики типичного модуля памяти, перейдем теперь к его внутренней организации. Одна из возможных схем, позволяющих выбирать произвольную ячейку ЗУ в нужный момент времени, показана на рис. 1.2.

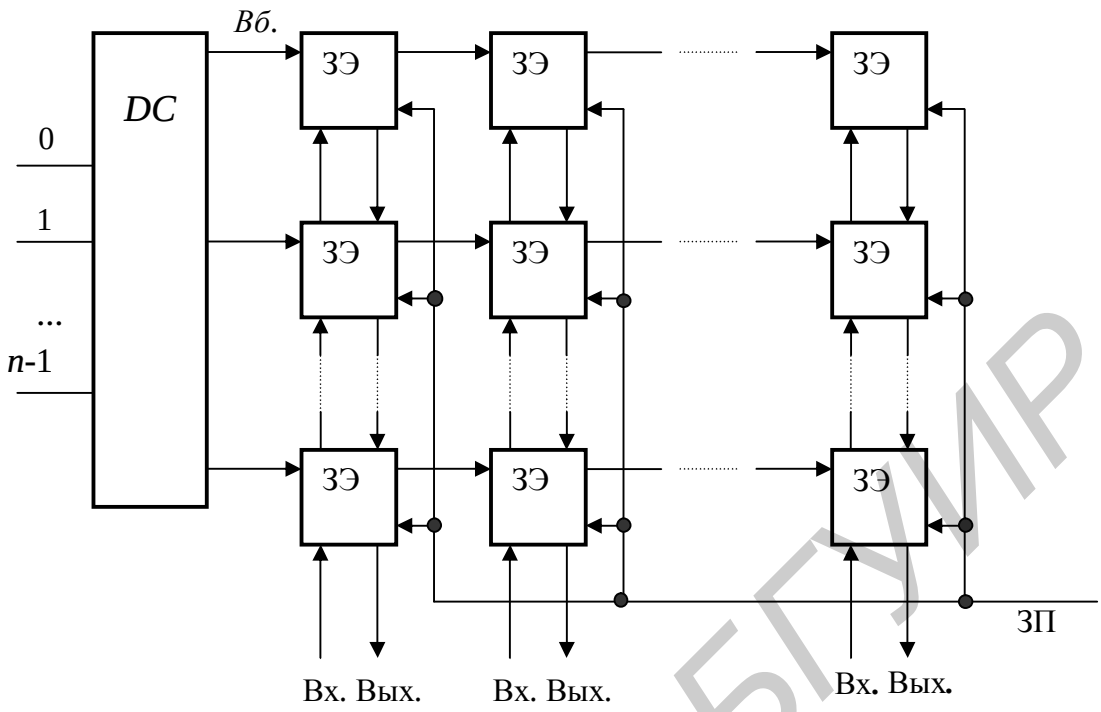


Рис. 1.2

В приведенной схеме операция чтения или записи выполняется одновременно для всех запоминающих элементов (ЗЭ) одной строки. Каждая строка представляет собой ячейку, хранящую одно информационное слово и имеющую свой адрес. Встроенный в ИМС дешифратор «1 из 2^n» служит для выбора ячейки по заданному адресу, и соответственно выходные 2^n линий дешифратора называют линиями выборки слова (*Вб.*).

В запоминающем элементе (рис. 1.3) для хранения информации используется триггер *D*-типа. Это определяет следующий алгоритм функционирования модуля при записи или чтении. В режиме чтения активизируется требуемая линия выборки *Вб.* дешифратора памяти. При этом состояние триггера через элементы 2И и 2ИЛИ выбранной ячейки передается на выход ЗУ и используется системой по назначению.

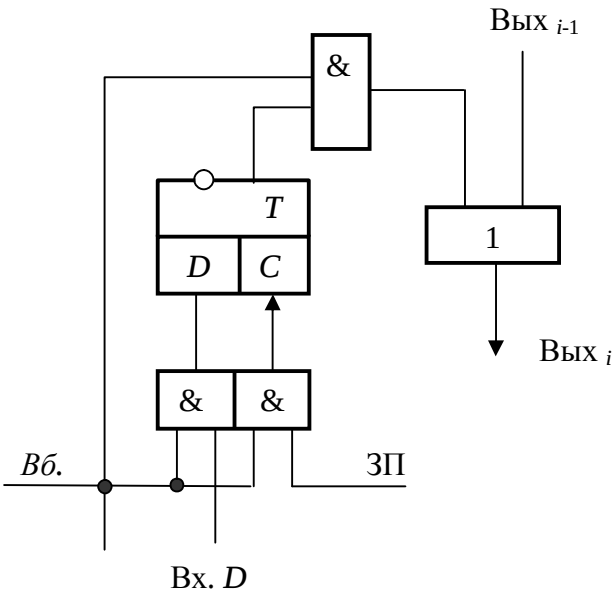


Рис. 1.3

Линия записи в данном случае не должна быть активной, чтобы триггер ЗЭ сохранял своё предыдущее состояние.

Запись информации в ЗЭ производится также после подачи сигнала *Вб*. При этом два логических элемента 2И оказываются разрешенными, что позволяет скоммутировать входную информацию на вход *D*-триггера и при подаче положительного фронта сигнала записи осуществить переключение элемента памяти и запоминание входной информации.

Представленная на рис. 1.2 схема дает возможность организовать обращение со стороны процессора в произвольную строку матрицы и в произвольный момент времени. В связи с этим устройства рассмотренного типа классифицируются как ЗУ с произвольной выборкой (ЗУПВ).

Недостатком рассмотренной схемы является наличие больших затрат аппаратуры на построение дешифратора строк. Так, например, при числе адресных линий, равном $n = 10$, сложность линейного дешифратора составит величину $C = n \cdot 2^n = 10 \cdot 2^{10} = 10\,240$ элементарных входов.

Для устранения данного недостатка выполним реорганизацию схемы дешифрации, передав часть функций блока *DC* самому ЗЭ.

1.2.2. ЗУ с двумерной адресацией и двунаправленной шиной данных

Если множество адресных линий дешифратора памяти разделить на два подмножества, то суммарная сложность устройства выборки будет существенно уменьшена (рис. 1.4). Схемы с таким способом организации адресных шин получили название ЗУ с двумерной адресацией. Сложность дешифратора при этом определяется формулой

$$C = 2\left(\frac{n}{2} 2^{\frac{n}{2}}\right) = 2\left(5 \cdot 2^5\right) = 320 \text{ при } n = 10.$$

В общем случае структурная схема ЗУ с двумерной адресацией и двунаправленной линией данных может быть представлена в следующем виде (рис. 1.5). В приведенном устройстве оба дешифратора работают одновременно.

Это позволяет активизировать две линии выборки и адресовать 1 бит памяти в текущий момент времени. Для считывания *m*-разрядного слова из ЗУ требуется *m* одинаковых интегральных модулей, выбираемых параллельно под управлением одного и того же адреса. Внутренняя структура запоминающего элемента ЗУ с двумерной адресацией и двунаправленной шиной данных в частном случае может быть представлена схемой, приведенной на рис. 1.6.

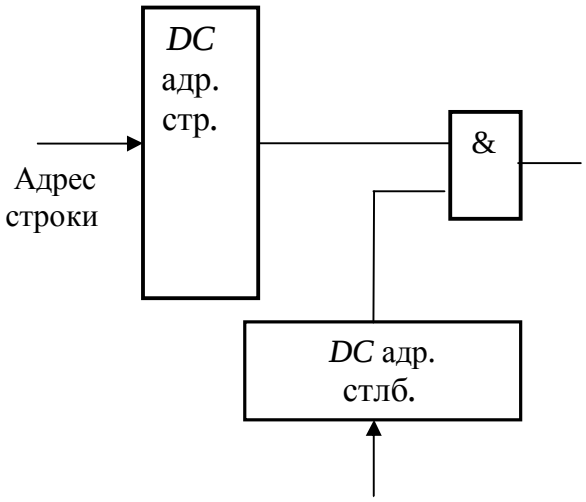


Рис. 1.4

Устройство работает следующим образом. В режиме чтения сигнал $\overline{ЗП}$ = 1, и выход элемента 4 с высоким сопротивлением шины находится в рабочем режиме. При этом если сигналы *Вб*.стр. и *Вб*.стл. равны 1 одновременно, то при наличии потенциального сигнала *strob* = 1 содержимое триггера ЗЭ выводится на трехстабильную шину ячейки. В режиме записи сигналы дешифраторов также указывают на ячейку памяти, но теперь уже приемник данных. Сигнал $\overline{ЗП}$ устанавливается в ноль, что вызывает запирание элемента 4, то есть установку на его выходе третьего состояния (R_{off}).

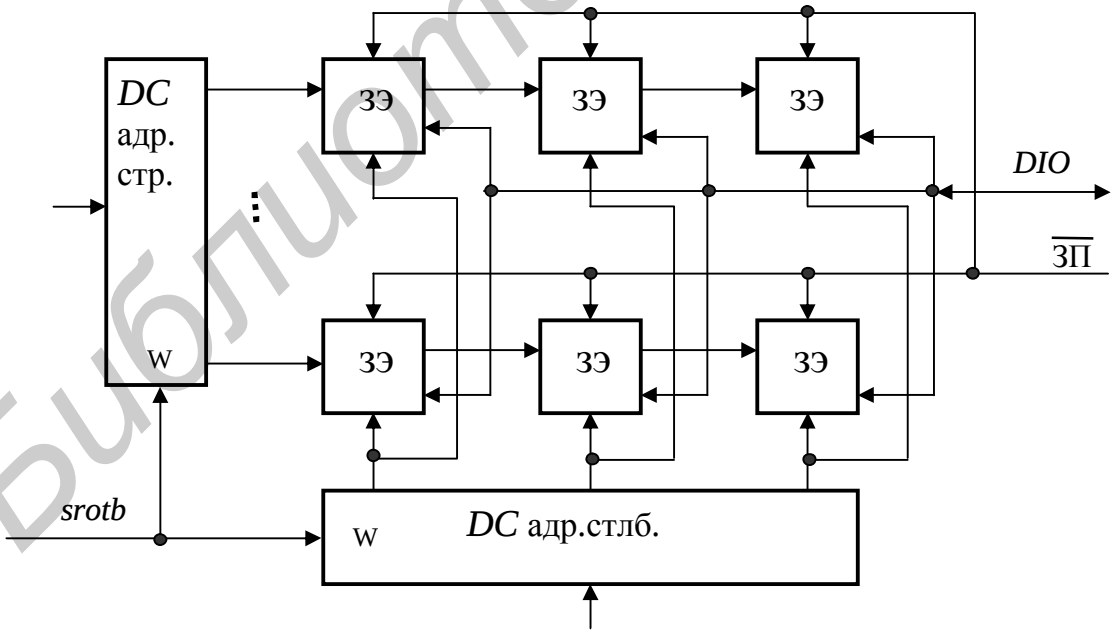


Рис. 1.5

В результате данных действий на входе *DIO* оказывается возможной установка вводимого бита. Далее, на вход *strob* подается короткий импульс стробирова-

ния, который и инициирует запись бита в триггер. Последующее переключение импульса записи в единицу позволяет отпереть выходную шину элемента 4 и включить выходную шину ЗЭ в рабочий режим.

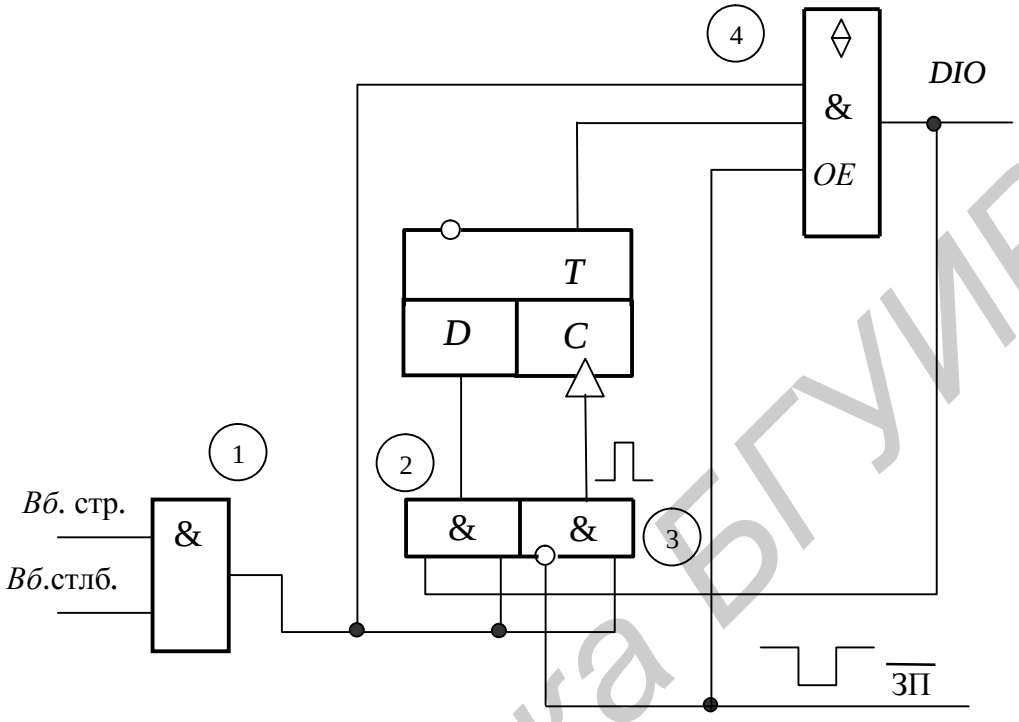


Рис. 1.6

1.2.3. Построение ОЗУ компьютера из модулей памяти

Выберем в качестве базовой ячейки для построения оперативного запоминающего устройства модуль памяти со структурой, приведенной на рис. 1.1. Тогда конфигурация схемы ЗУ из рассмотренных модулей (1 банк памяти) может быть представлена в виде подсистемы, показанной на рис. 1.7.

В состав блока входят следующие шины: шина адреса (ША), шина управления (ШУ), шина данных (ШД). Кроме того, устройство содержит ряд однотипных модулей, позволяющих выполнять двунаправленный обмен данными.

Шина адреса (ША) компьютера содержит линии адреса A_0–A_{n-1}. При этом младшие k разрядов соединены с адресными входами модулей памяти, а старшие n-k разрядов используются для подключения 2^{n-k} однотипных модулей. Выборка модуля осуществляется путем подачи старших бит адреса на входы дешифратора памяти и активизации на его выходе определенной линии выборки. При этом адресуемый модуль подключается к шине данных (ШД) компьютера, а двунаправленные выводы *DIO* остальных модулей переводятся в третье состояние. С этого момента времени процессор системы может обращаться к ОЗУ за очередной командой или набором данных.

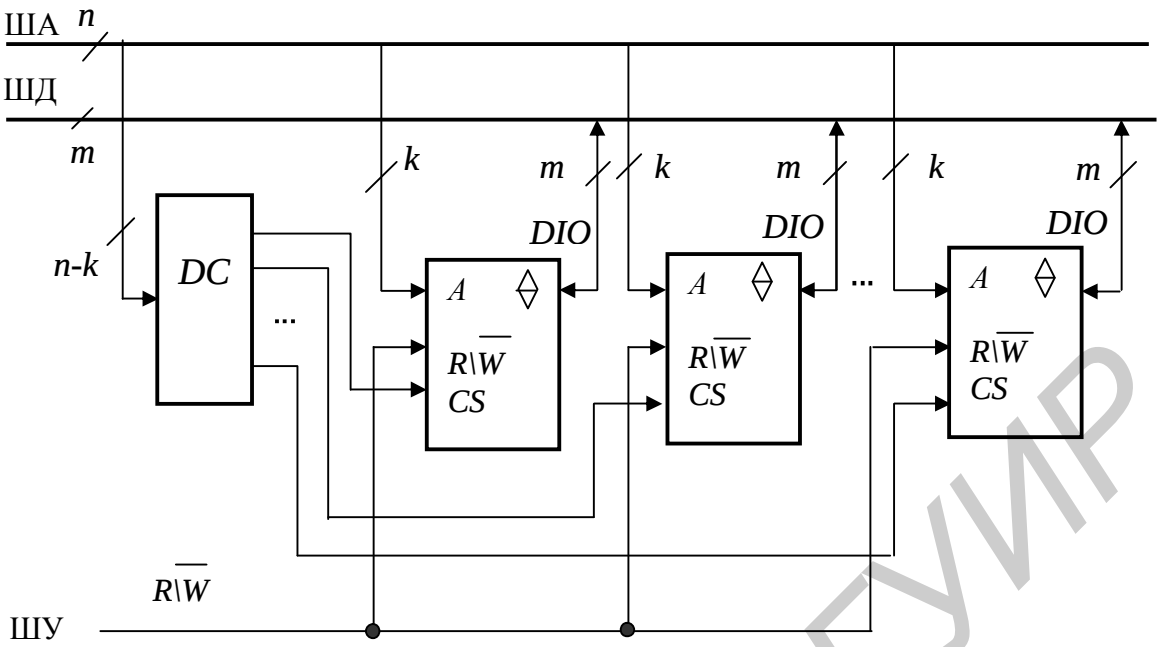


Рис. 1.7

1.2.4. Включение модулей ПЗУ в адресное пространство оперативной памяти

Модули ПЗУ, так же как и другие блоки памяти, включаются в состав ОЗУ путем соединения с шинами адреса и данных компьютера (рис. 1.8).

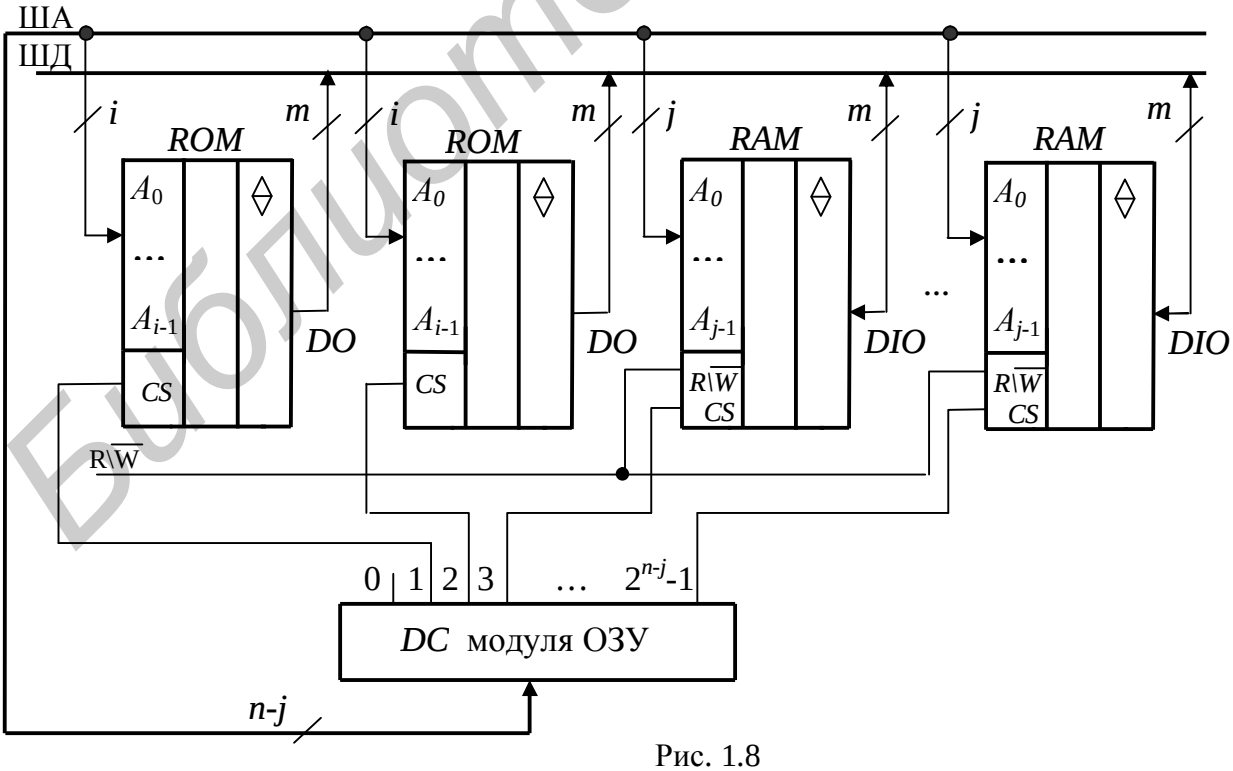


Рис. 1.8

Во всех модулях адресные линии подключены к младшим разрядам адресной шины. Обращение к блоку модулей ОЗУ разрешается при установке на ША кодовой комбинации числа $2^{j+1}+2^{j}=3\cdot 2^{j}$ и более (при $A=2^{n}-2^{j}$ адресуется последний модуль). При обращении к ЗУ активизируется одна из линий выборки, что позволяет подключить выбранный модуль памяти к ШД компьютера. Двунаправленные выходы остальных модулей переключаются в состоянии R_{off}.

При необходимости обращения к ПЗУ на шине адреса устанавливается значение $A=2^{j}$, что определяет чтение первой ИМС ПЗУ в диапазоне адресов $2^{j} \le A < 2^{j}+2^{i}$, $i<j$. Если $A=2^{j+1}$ и более, то организуется обращение (чтение) ко второму модулю ПЗУ в адресном пространстве $2^{j+1} \le A < 2^{j+1}+2^{i}$. При установке на входе дешифратора памяти нулевой комбинации в компьютере возможна адресация устройств ввода–вывода, которые, как правило, подключаются к младшим линиям ША.

1.2.5. Переключаемые банки памяти

В случаях, когда решаемые компьютером задачи требуют использования ЗУ большего объема, чем допускает прямая адресация, применяется методика компоновки блоков памяти из переключаемых банков (рис. 1.9).

Приведенная схема состоит из двух банков, включение в систему которых осуществляется с помощью триггера банка ($Т_б$). При обращении к ЗУ на ША выдается адрес $Т_б$ и выполняется его переключение (если необходимо) в требуемое состояние. При установке триггера в ноль к ШД подключается левый банк (при $\overline{IOW}=1$), если же $Т_б = 1$, то обмен информацией осуществляется с правым банком.

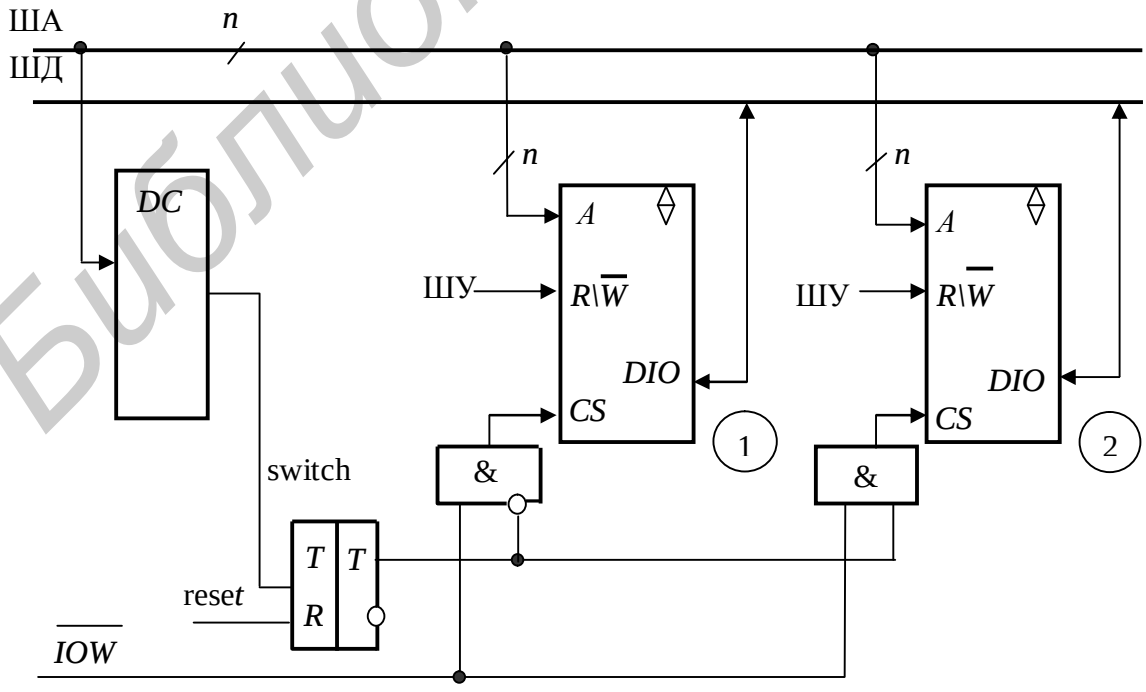


Рис. 1.9

Наличие единичного потенциала на линии $\overline{IOW}$ свидетельствует о том, что компьютер отрабатывает режим обращения к ЗУ. Если же $\overline{IOW} = 0$, то в системе организуется отключение памяти и, например, обмен данными с УВВ.

При построении ЗУ с большим числом банков вместо указателя триггера используется специальный регистр, адресуемый программно как обычный порт ввода–вывода (рис. 1.10).

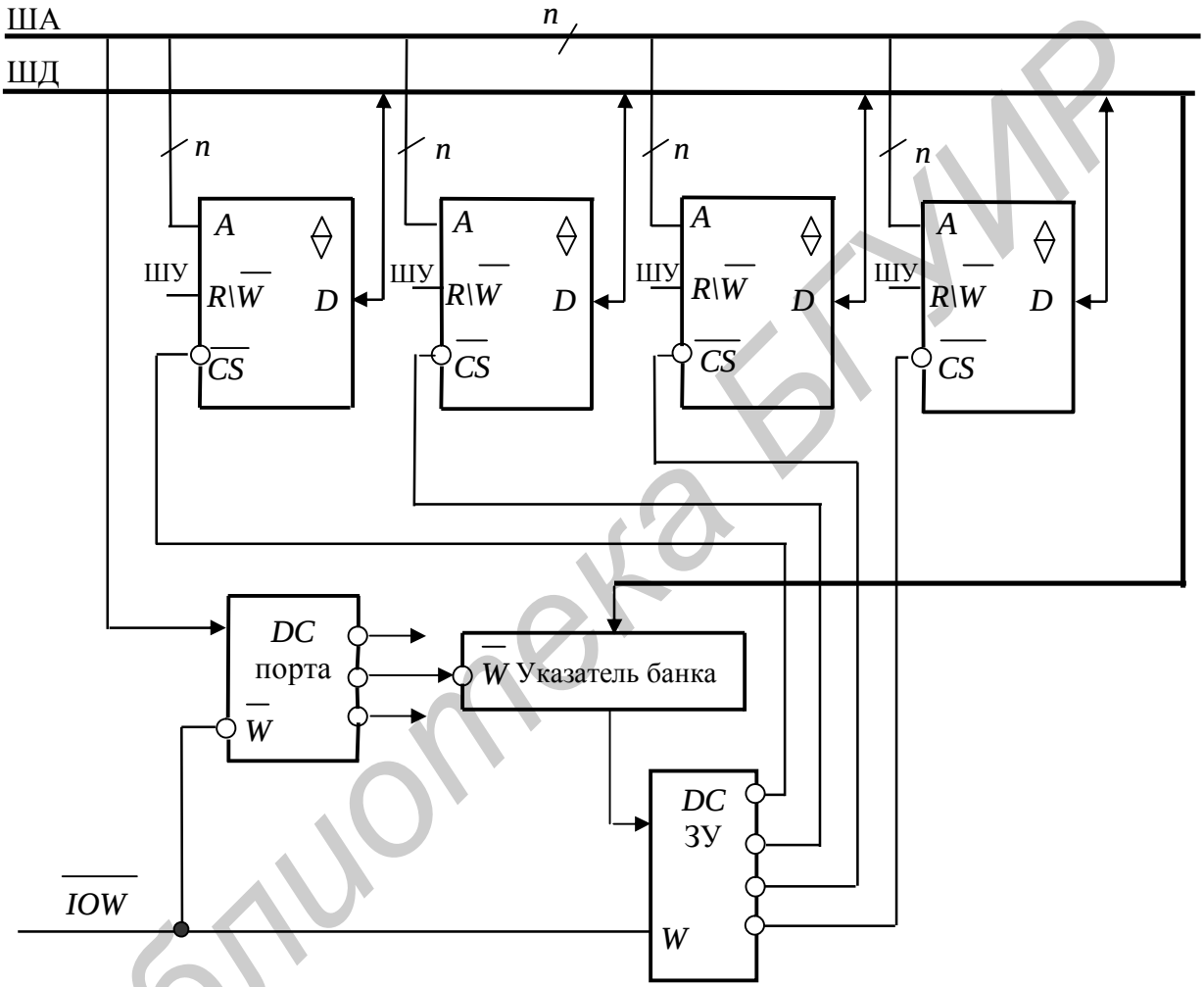


Рис. 1.10

Схема работает следующим образом. Перед началом обмена программируется указатель банка. С этой целью программист должен в ходе написания программы предусмотреть возможность перехода от одного блока ЗУ к другому с помощью команд ввода–вывода. Для изменения номера используемого банка на ША выдается адрес регистра-указателя, на ШД – соответствующий код или номер банка. *DC* порта активизирует на своем выходе одну из линий выборки (при $\overline{IOW} = 0$), что позволяет записать номер адресуемого устройства в регистр-указатель. Далее содержимое указателя передается на вход дешифратора па-

мяти *DC*, который, воздействуя на вход $\overline{CS}$ адресуемого банка, переводит его информационные линии в рабочий режим. Линии данных других модулей (банков ЗУ) остаются в третьем состоянии.

В общем случае число банков может быть неограниченно увеличено, однако общее время перехода от устройства к устройству при этом возрастает.

Неудобство вызывает также и необходимость частого программного переключения в системе ЗУ. Это требует дополнительных усилий программиста и предвидения условий переполнения блоков.

1.2.6. Способы увеличения информационной емкости ЗУ

При проектировании ЗУ на заданных модулях БИС часто приходится решать задачу увеличения общей информационной емкости памяти компьютера. Как правило, эта задача решается следующими тремя способами: 1) увеличением разрядности слов, 2) увеличением количества слов, 3) сочетанием первых двух способов.

При решении задачи путем увеличения разрядности слов, хранимых в ОЗУ, можно использовать схему, изображенную на рис. 1.11. Здесь *CS* – сигнал выборки блока ОЗУ; $R/\overline{W}$ – сигнал записи в ОЗУ ($R/\overline{W} = 0$) или чтения ($R/\overline{W} = 1$); A_0–A_{m-1} – адресная шина; DI_0–DI_{nk-1} – входная шина данных; DO_0–DO_{nk-1} – выходная шина данных.

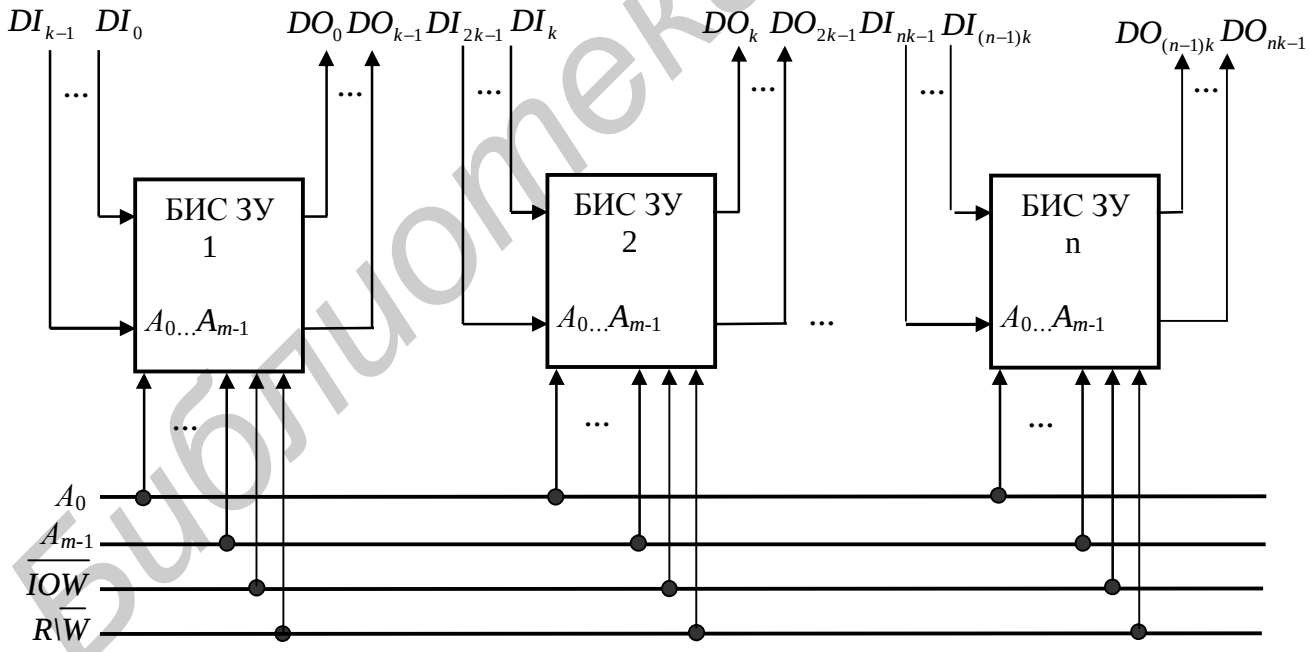


Рис. 1.11

При увеличении информационной емкости запоминающих устройств путем наращивания количества слов требуемое число БИС памяти соединяется параллельно по входам и выходам (рис. 1.12). Выборка кристалла из *n* модулей осуществляется с помощью дешифратора памяти.

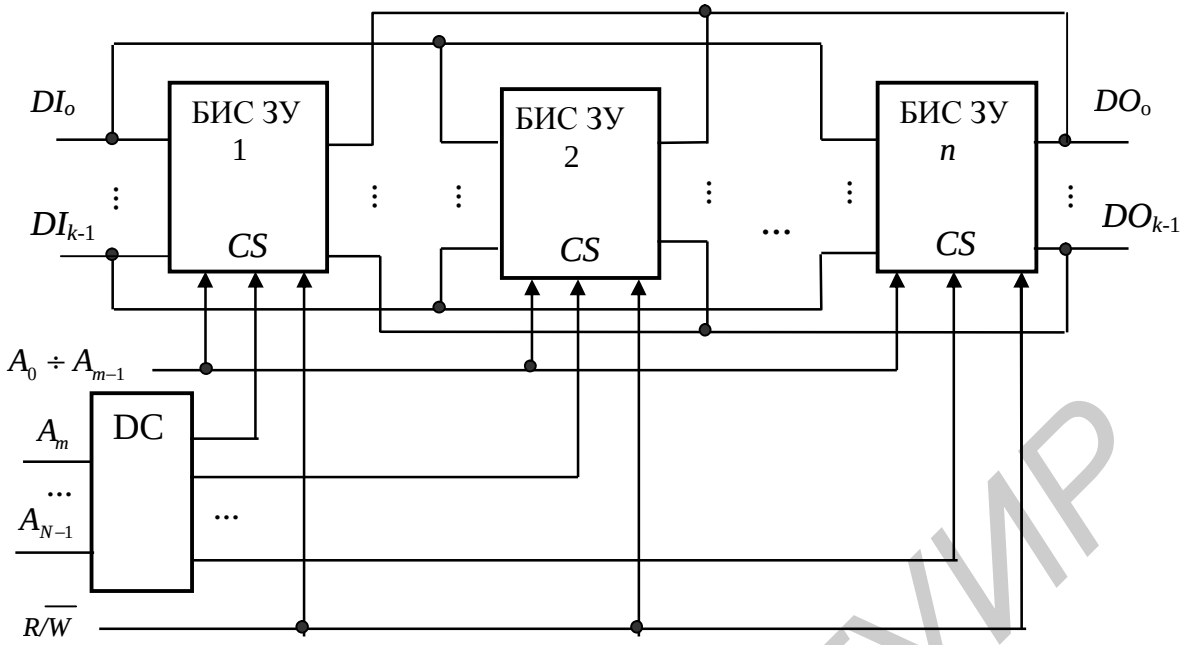


Рис. 1.12

В схеме, приведенной на рис. 1.12, весь адрес делится на две части. Разряды A_0–A_{m-1} позволяют адресоваться к информации в пределах одного из модулей ЗУ: БИС ЗУ$_1$ … БИС ЗУ$_n$, разряды A_m–A_{N-1} – выбрать конкретный модуль памяти.

При обращении к одному из блоков (модулей) дешифратор *DC* расшифровывает входной адрес и активизирует одну из линий выборки. Это позволяет подключить выходы соответствующей БИС к шине данных (ШД), а выходы остальных модулей памяти перевести в состояние высокого сопротивления R_{off}. Теперь при записи информации на шину управления подается сигнал $R\backslash\overline{W} = 0$, а при чтении – потенциал логической единицы.

При увеличении информационной емкости путем совмещения указанных двух способов организуется пространство БИС ЗУ размером $p \times n$, где n – число БИС в строке пространства, p – число строк (рис. 1.13). Адресация строк при этом осуществляется в соответствии со схемотехникой блока, приведенного на рис. 1.12. Чтение слова из выбранного набора БИС выполняется по общей методике с помощью адресной информации на входах A_0–A_{m-1}.

Очевидно, что приведенная методология позволяет сформировать блоки ЗУ произвольной емкости, однако число интегральных модулей при существенном возрастании значений n и p может достигать нескольких сотен. В связи с этим далее будем рассматривать схемы динамической памяти, технология изготовления которых позволяет компактно реализовать практически любые объемы запоминающих устройств. Следует также помнить, что увеличение объемов оперативной памяти существенно сказывается на цикле обращений к ЗУ. Поэтому в реальных системах выполняют кэширование данной подсистемы путем введения промежуточных или буферных устройств, являющихся вторым уровнем в иерархии памятей вида: РЗУ–КЭШ–ОЗУ–КЭШ–ВЗУ.

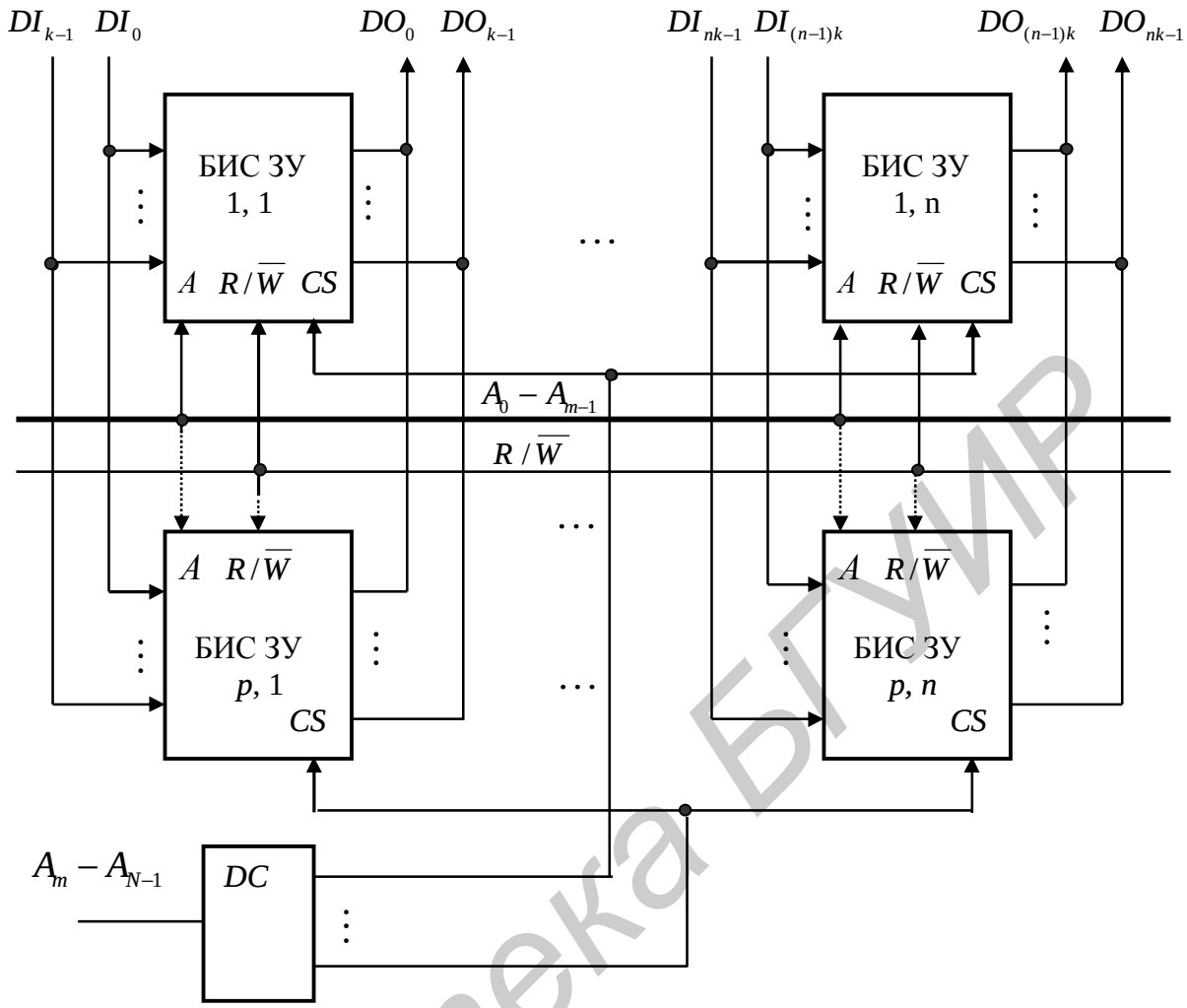


Рис. 1.13

1.3. Динамическое оперативное запоминающее устройство

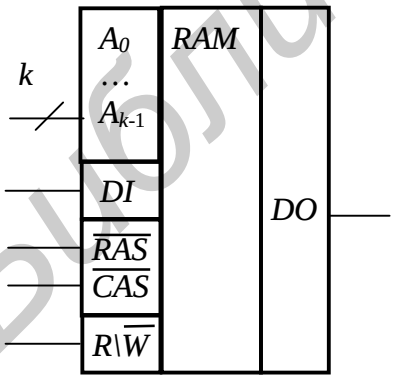


Рис. 1.14

Выберем в качестве исходного модуля для проектирования ОЗУ динамического типа интегральную микросхему со следующими характеристиками (рис. 1.14):

A_0–A_{k-1} – k-разрядный адрес строки или столбца;

DI – однобитные входные данные;

$\overline{RAS}$ – строб записи адреса строки;

$\overline{CAS}$ – строб записи адреса столбца;

$R/\overline{W}$ – 0 – запись, 1 – чтение;

DO – однобитные выходные данные.

Для данной интегральной микросхемы режим записи в матрицу запоминающих элементов предполагает отработку следующей временной диаграммы (рис. 1.15):

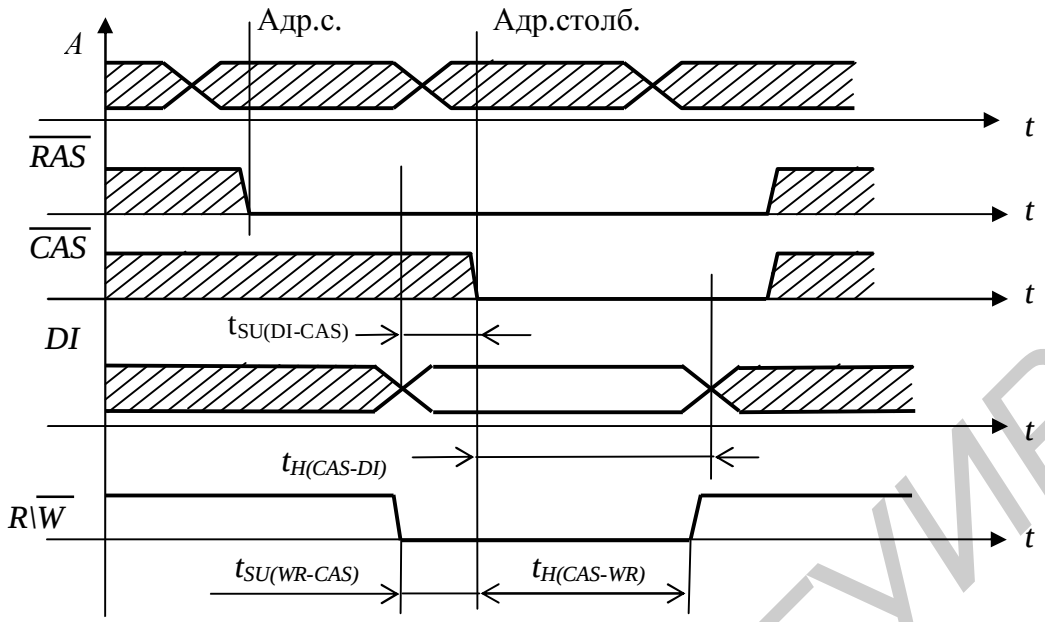


Рис. 1.15

В начале цикла на адресные входы модуля ЗУ подается адрес строки, запись которого во внутренний регистр осуществляется по сигналу $\overline{RAS}$ (стробу записи адреса строки). После этого значение входного адреса изменяется на адрес столбца, запись которого также выполняется по сигналу стробирования $\overline{CAS}$ (или стробу записи адреса столбца). Входные данные и сигнал записи устанавливаются на шинах ОЗУ заблаговременно (до подачи импульса $\overline{CAS}$). При этом момент установки и удержания логических уровней должен строго соответствовать нормативным параметрам, а общее время цикла учитываться при расчете тактовой частоты синхронизации компьютера.

В режиме чтения данных временная диаграмма сигналов на шинах ША, ШУ и ШД может быть представлена в следующем виде (рис. 1.16).

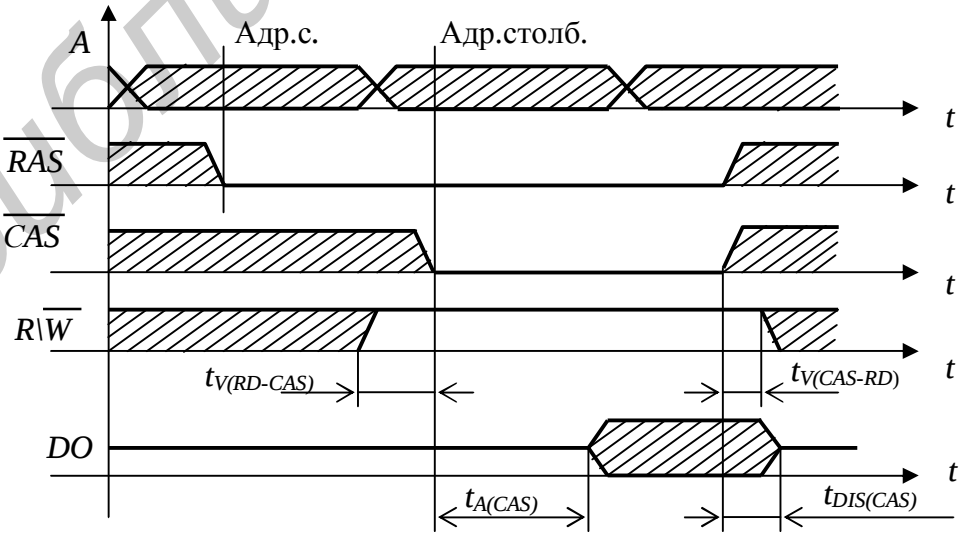


Рис. 1.16

Технология хранения данных в динамическом ЗУ требует периодической перезарядки «запоминающих» емкостей. Этот режим получил название регенерации и технически реализуется путем чтения всех строк или всех столбцов модуля ЗУ в соответствии со следующей временной диаграммой (рис. 1.17).

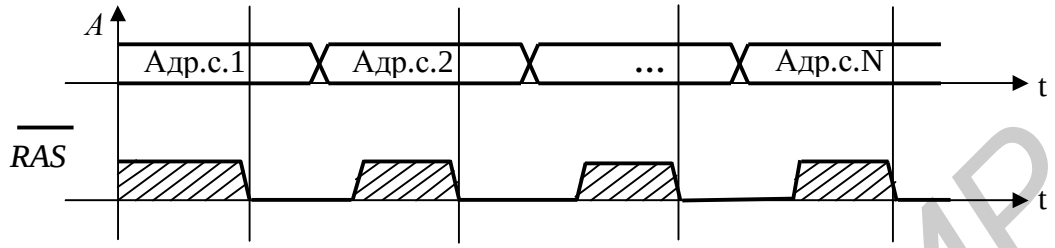
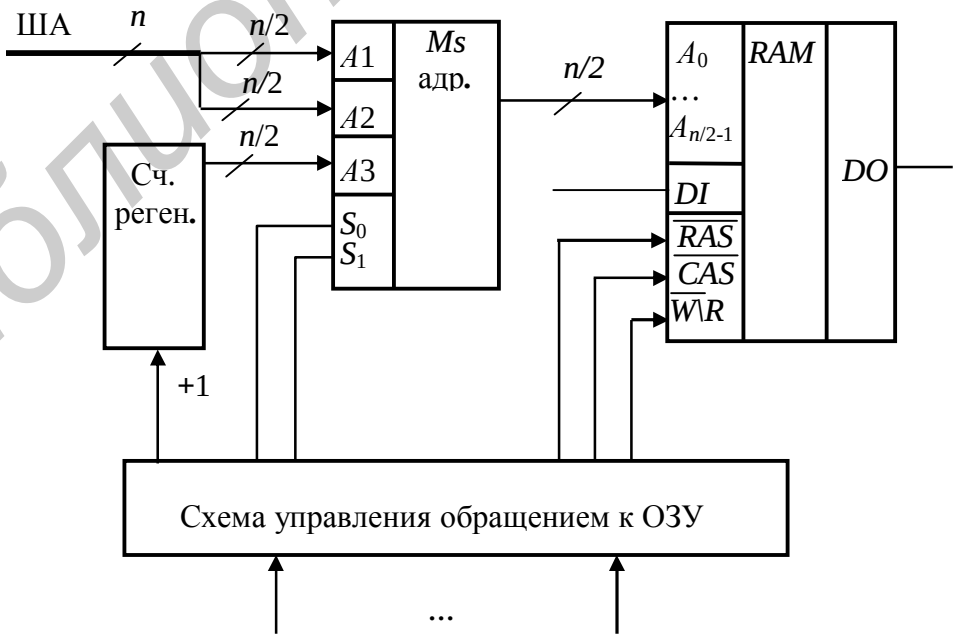


Рис. 1.17

Для совмещения процессов чтения, записи и регенерации в компьютерах используются специальные схемы, получившие название контроллера динамической памяти (КОЗУ – контроллер динамического ОЗУ). Структурная схема такого устройства представлена на рис. 1.18.

В общем случае схема контроллера предназначена для преобразования адресов и обновления информации в ячейках динамической памяти. При обращении к ОЗУ со стороны процессора используемое *n*-разрядное адресное слово условно или явно делится на две части – адрес строки и адрес столбца. Схема управления обращением к ОЗУ должна обеспечить своевременную коммутацию этих адресов и подачу сигналов стробирования на входы $\overline{RAS}, \overline{CAS}$.



Сигналы управления и ТИ

Рис. 1.18

При регенерации необходимо обеспечить чтение всех строк или всех столбцов матрицы с помощью специального счетчика. При этом все биты в строке или в столбце ОЗУ регенерируются одновременно.

1.3.1. Принцип регенерации ДОЗУ по прерыванию

Обновление информации в динамической памяти должно осуществляться через определенные промежутки времени. Это время называется периодом регенерации и, как правило, составляет несколько миллисекунд реального времени.

При восстановлении памяти обращение со стороны процессора должно быть запрещено, каналы ПДП отключены и в целом вычислительный процесс приостановлен (регенерация по прерыванию) (рис. 1.19). Блок КОЗУ при данном способе восстановления функционирует следующим образом.

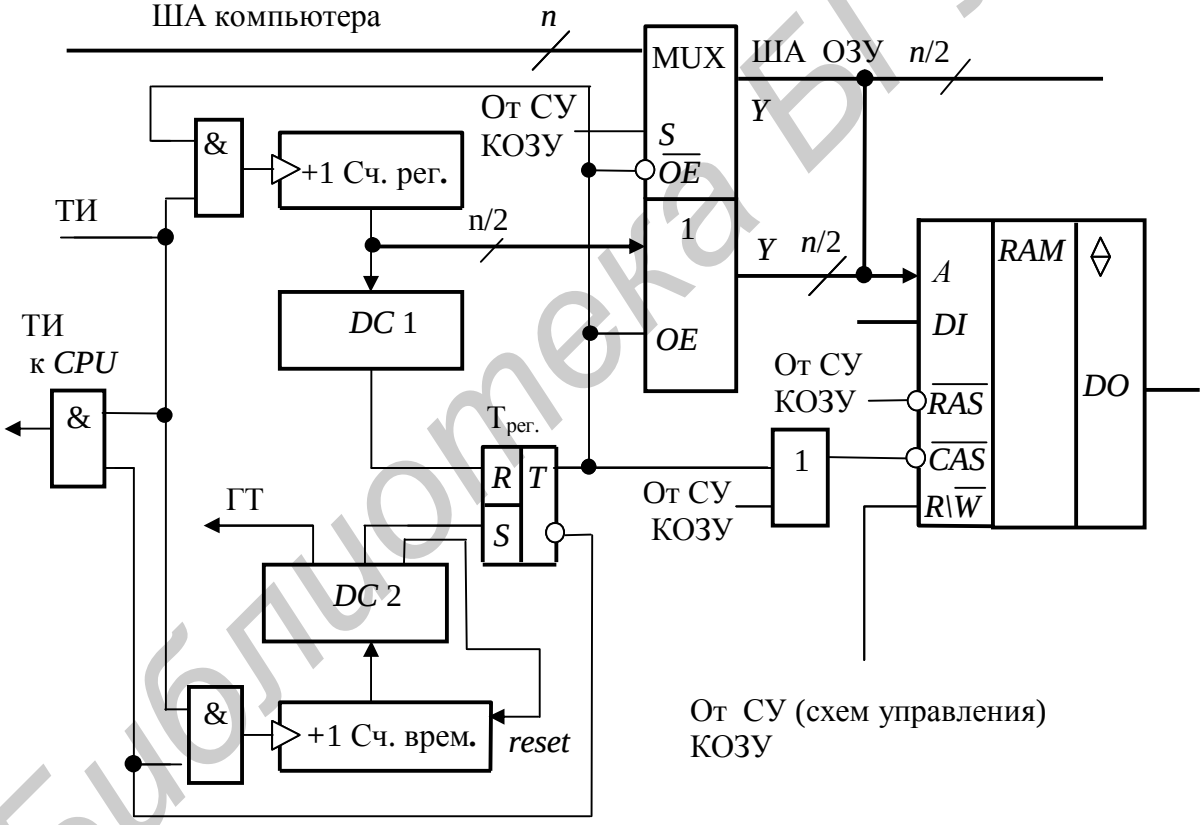


Рис. 1.19

В рабочем режиме $T_{рег}$ устанавливается в «0». При этом разрешается передача тактовых импульсов от ГТИ к *CPU* и к схеме отсчета периода регенерации (счетчику времени). Прямой выход $T_{рег}$ нулевым уровнем запирает *Y*-выходы блока трехстабильных усилителей и отключает выходы счетчика регенерации от адресных шин модуля памяти. Мультиплексор ША включен

в рабочий режим, и вся схема функционирует как ЗУ с двухтактной записью адреса.

О начале цикла регенерации свидетельствует появление импульсного сигнала на выходе ГТ дешифратора *DC2*. Он заблаговременно оповещает процессор о необходимости отключения ОЗУ и возможном сбое в вычислениях из-за регенерации.

Началу периода подзарядки памяти соответствует установление требуемого значения в счетчике времени и формирование схемой *DC2* на соответствующем выводе потенциала логической единицы. Это вызывает установку $T_{рег}$ в единицу и соответственно запрещение (инверсный выход триггера находится в нуле) тактирования *CPU* и счетчика времени. Процессор приостанавливает функционирование и ожидает завершения режима регенерации.

Прямой выход триггера разрешает коммутацию адресов регенерации с выхода счетчика Сч.рег. на адресную шину ОЗУ с помощью блока трехстабильных вентилей. Кроме того, в схеме разрешается прохождение импульсов синхронизации на вход формирователя адресов (Сч.рег.), а на входе $\overline{CAS}$ модуля памяти устанавливается логическая «1». Выходная шина *DO* модуля памяти переводится в третье состояние, и на вход $\overline{RAS}$ подаются импульсы регенерации.

После полного перебора адресов ЗУ *DC*1 формирует сигнал сброса $T_{рег}$ и память снова включается в структуру компьютерной системы.

1.3.2. Схема регенерации ЗУ на базе регистра сдвига

Схема регенерации ЗУ на базе регистра сдвига приведена на рис. 1.20 и работает следующим образом.

В рабочем режиме $T_{рег}$ установлен в нулевое состояние, что позволяет:

1) передавать импульсы синхронизации от ГТИ к процессору;

2) выполнять отсчет длительности рабочего периода счетчиком Сч. врем.;

3) передавать управляющие сигналы от регистра сдвига на вход S_0 мультиплексора адреса (через элемент 2ИЛИ) и на входы $\overline{RAS}$ и $\overline{CAS}$ модуля памяти.

При обращении к ЗУ процессор формирует один из сигналов *MemR* или *MemW* (*Mem x*), что позволяет загрузить необходимый управляющий код в регистр сдвига (в данном случае это значение равно 07_8). На входах S_1S_0 устанавливается нулевая комбинация 00. Это определяет передачу адреса строки на вход *A* модуля памяти и по окончании действия импульса $W = ТИ\ Ч\ Mem(x)$ посредством трехвходового элемента ИЛИ разрешить очередному синхроимпульсу компьютера сдвиг в регистре сдвига. При выполнении данной микрооперации на входах S_1S_0 сохраняется код 00, на входе же $\overline{RAS}$ формируется отрицательный фронт строба записи адреса строки.

Следующий синхросигнал сдвигает управляющий регистр еще на один разряд, что вызывает появление на входах S_1S_0 кода 01 и передачу адреса столбца на входы модуля ЗУ.

Третий импульс сдвига порождает на входе $\overline{CAS}$ ОЗУ строб записи адреса столбца. Таким образом, на данный момент времени весь адрес оказывается записан во внутренний регистр ЗУ. После этого выполняется обмен данными между процессором и памятью в соответствии с кодом выполняемой команды.

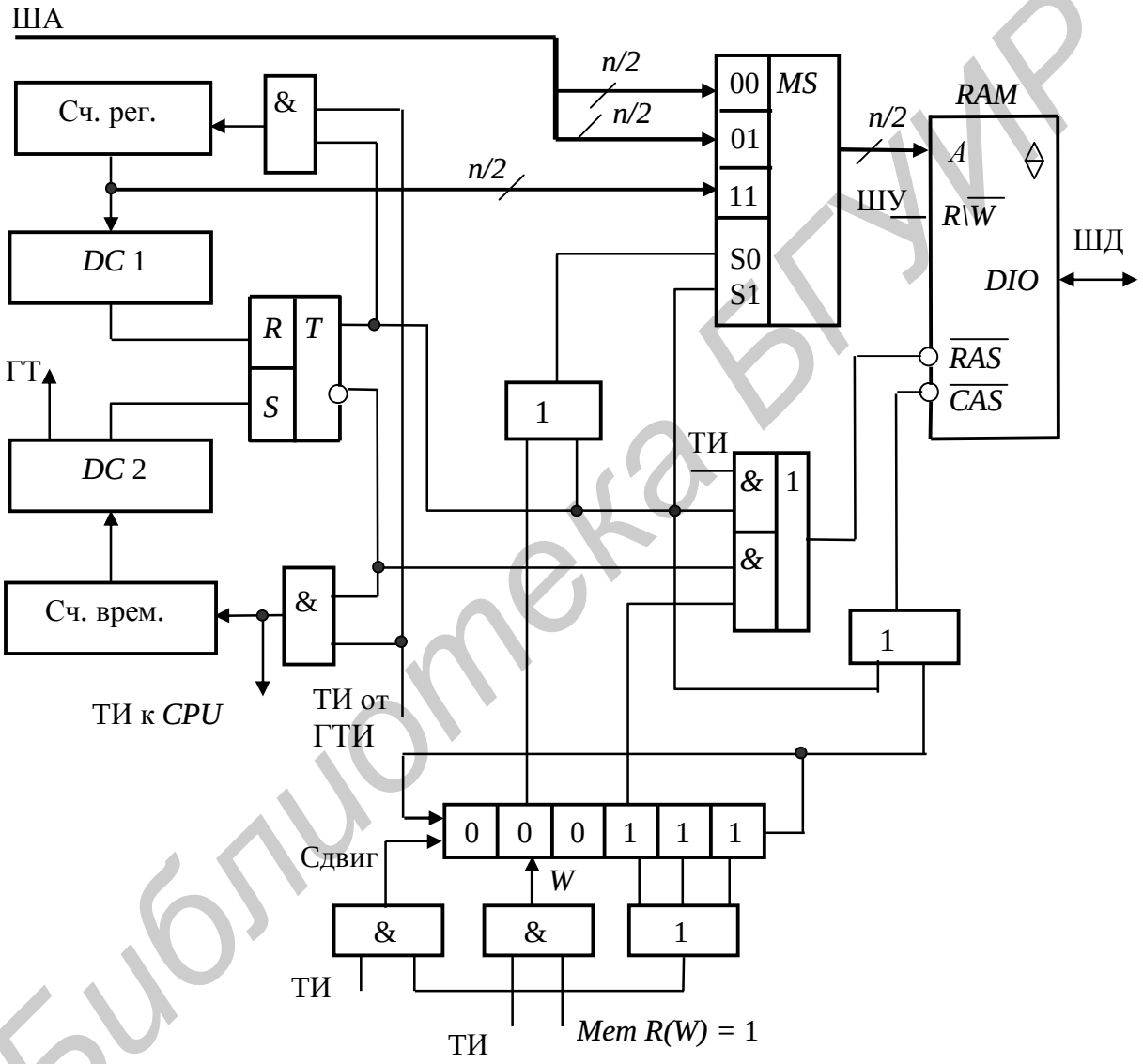


Рис. 1.20

По истечении допустимого времени работы ЗУ дешифратор *DC2* устанавливает $T_{рег}$ в 1. При этом прямой выход триггера фиксирует на входах S_1S_0 мультиплексора значение 11, что разрешает передачу адресов строк регенерации на входы ЗУ, и позволяет включить в работу счетчик *Cm2* регенерации.

Инверсный выход $T_{рег}$ запрещает прохождение сигналов от регистра сдви-

га на вход $\overline{RAS}$ модуля ОЗУ. При этом в качестве источника стробов по данному входу используются синхросигналы ГТИ компьютера.

Прямой выход триггера фиксирует на входе $\overline{CAS}$ модуля ОЗУ уровень логической единицы. Следовательно, на все время регенерации выходы модуля переводятся в состояние высокого сопротивления R_{off}.

По окончании перебора всех строк дешифратор *DC*1 сбрасывает $T_{рег}$ в ноль, и схема ЗУ вновь оказывается готова к очередному циклу обмена с процессором.

Для предварительного оповещения процессора о начале регенерации и недостатке времени на обращение к памяти может быть использован сигнал ГТ = 1, который формируется с требуемым опережением времени.

1.4. Постоянное запоминающее устройство

Постоянные запоминающие устройства (ПЗУ) применяются для хранения служебных программ, констант, таблиц кодирования данных или другой служебной и неизменяемой в процессе вычислений информации. Структурная схема ПЗУ может быть представлена в следующем виде (рис. 1.21).

В состав устройства входят: дешифратор адреса *DC* и требуемое число (по числу выходов) логических элементов ИЛИ, подключенных к линиям выборки. При подаче адреса на вход ПЗУ осуществляется его расшифровка и активизация соответствующей линии *Вб*. Логический потенциал линии воздействует на входы логических элементов и переключает их выходы из нулевого состояния в единичное. Таким образом, разработчик постоянной памяти, самостоятельно задав закон соединения дешифратора и элементов ИЛИ, может получить практически любой набор выходных управляющих или информационных сигналов.

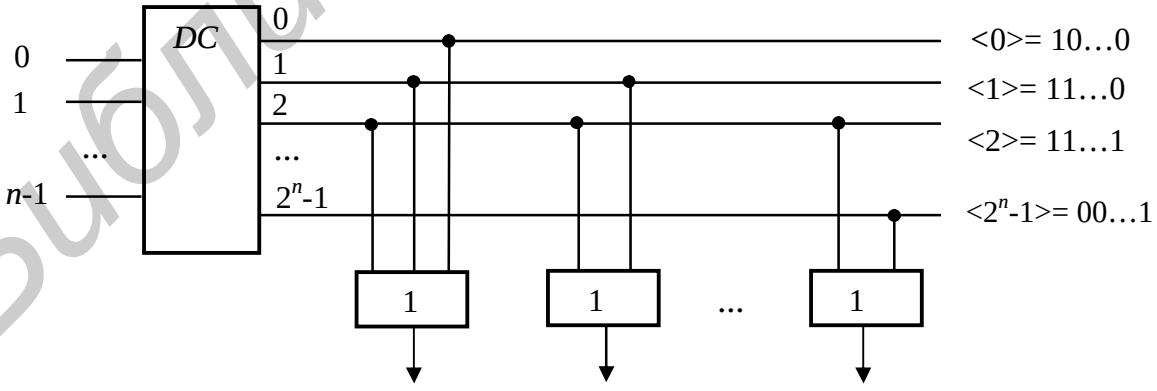


Рис. 1.21

Изготовление таких схем и программирование выполняется по следующей технологии (рис. 1.22).

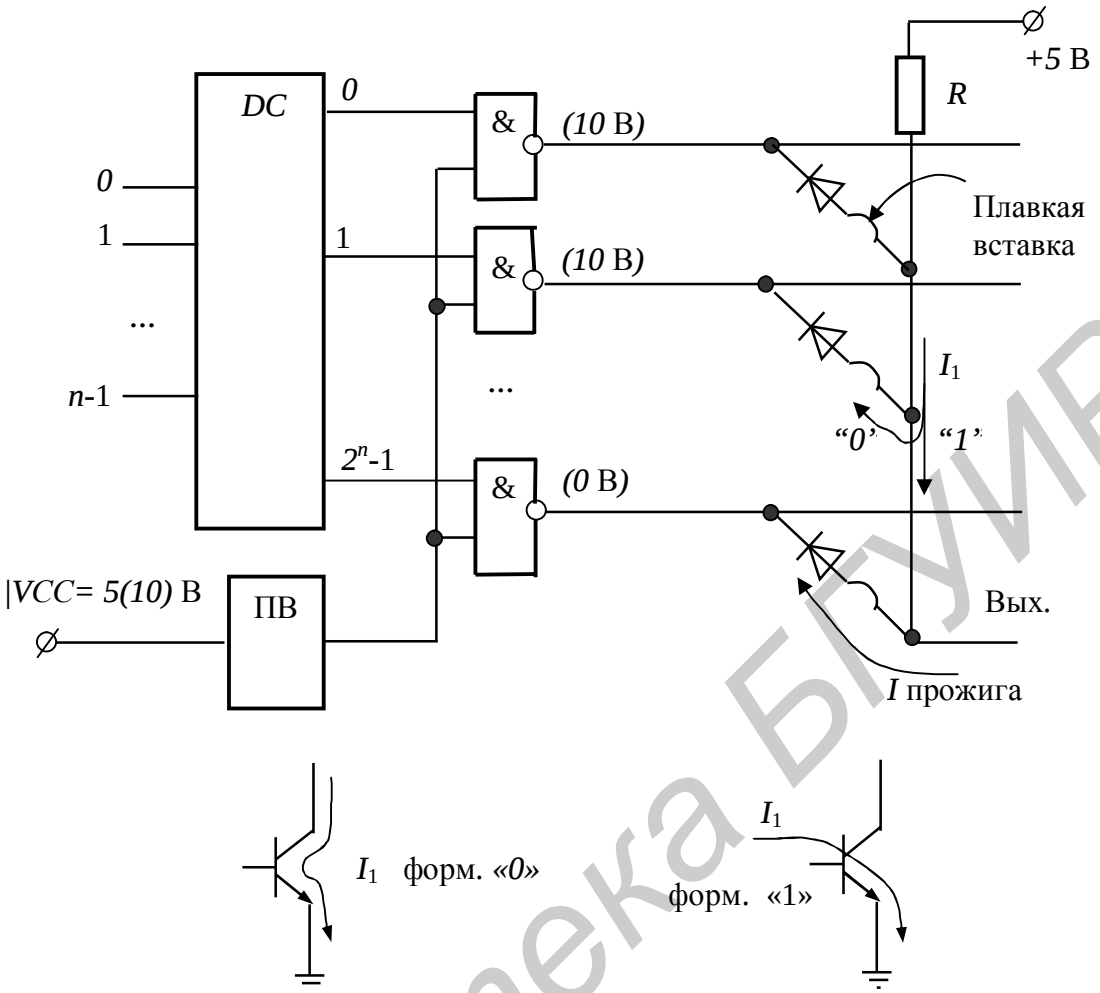


Рис. 1.22

Устройство, показанное на рис. 1.22, может находиться в одном из двух режимов – чтение или программирование. Конкретную реализацию режима определяет напряжение питания на входе *VCC* порогового вентиля (ПВ). Если напряжение *VCC=5* В, то в ПЗУ реализуется процесс чтения. В этом случае на выходе ПВ устанавливается логическая единица, а состояние выходной шины определяется наличием плавкой вставки в данном каскаде. Если плавкая вставка цела, на выходе устанавливается нулевой уровень. Данный потенциал обусловлен протеканием тока по пути (пути наименьшего сопротивления): *VCC* = =+5 В – плавкая вставка – выходной каскад элемента «2И-НЕ» – внутренняя шина «земля» логического элемента. Если вставка прожжена, ток протекает во входную цепь следующего каскада и на выходе ячейки устанавливается потенциал единицы.

В режиме программирования VCC=10 В. При этом пороговый вентиль также устанавливает на своем выходе уровень логической единицы, но с рабочим напряжением 10 В. Для прожига перемычки на входы ПЗУ подается адрес

требуемой ячейки, в результате чего в схеме активизируется соответствующий элемент «2И-НЕ». На выходе элемента устанавливается ноль, затем от источника тока на выходную шину подают питающее напряжение с уровнем 10 В. Данное напряжение вызывает протекание тока » 65 мА по цепи: плавкая вставка–диод–выход элемента «2И-НЕ». Ток плавит перемычку, затем внешний источник отключается. Наличие высокого (10 В) напряжения на выходах остальных элементов определяет малый ток в неадресуемых каскадах, что позволяет сохранить перемычки в других ячейках ПЗУ.

1.5. Стековое запоминающее устройство

Стеком называют запоминающее устройство с последовательным доступом, обеспечивающее считывание слов в порядке, обратном записи. Стек иногда представляют в виде магазинного ЗУ, доступ в котором в текущий момент времени разрешается к верхней ячейке (рис. 1.23).

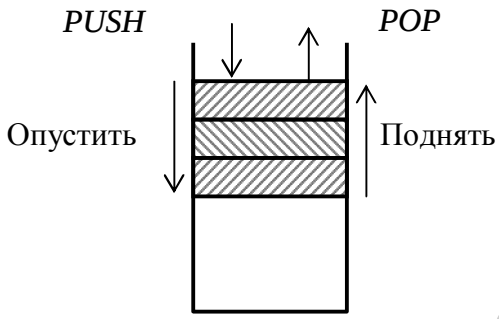


Рис. 1.23

Эта ячейка получила называние «вершины стека». Адрес вершины в каждый конкретный момент хранится в специальном регистре, который называют указателем стека.

Как правило, в аппаратных реализациях явно изготовленный указатель может отсутствовать (рис. 1.24). Адреса ячеек, к которым происходит очередное обращение, при этом определяются текущим моментом времени и последовательностью обращения к ЗУ. Когда слово записывается в стек, занимавшее вершину значение и все хранимые слова сдвигаются вниз на одну позицию, содержимое нижней ячейки теряется. Количество запоминающих регистров в стековом ЗУ называют глубиной стека.

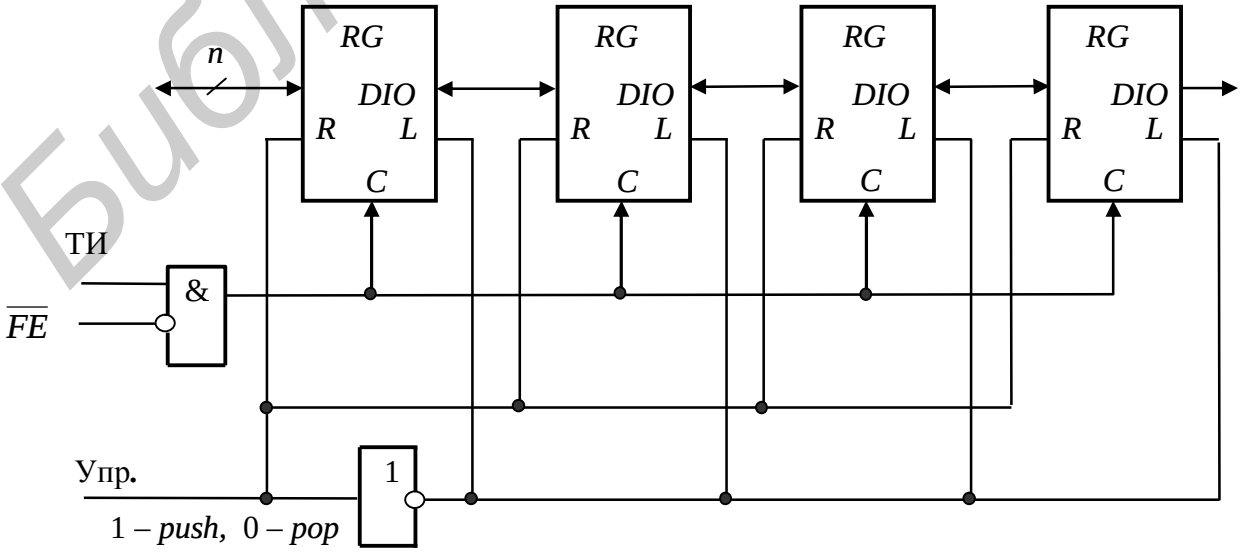


Рис. 1.24

Стековая память в общем случае является безадресной. Однако если стек является подразделом ОП, то в процессоре организуется адресный регистр, который также называют указателем стека. Основное назначение такого стекового ЗУ состоит в запоминании ССП при прерываниях или переходах от одной подпрограммы к другой (рис. 1.25).

На рис. 1.25 показан переход к подпрограмме №1 из ячейки с номером 550 на адрес 650 с запоминанием точки (адреса) возврата 551 в стеке. Аналогично из подпрограммы №1 из ячейки с номером 679 выполняется переход на адрес 730. Точка возврата 680 размещается в вершине стека, сдвигая предыдущее значение на одну позицию вверх. Начиная с адреса 770, осуществляется выход из подпрограммы №2 в точку 680 подпрограммы №1. Далее выполняется завершение данного программного модуля и возврат на адрес 551 основной программы.

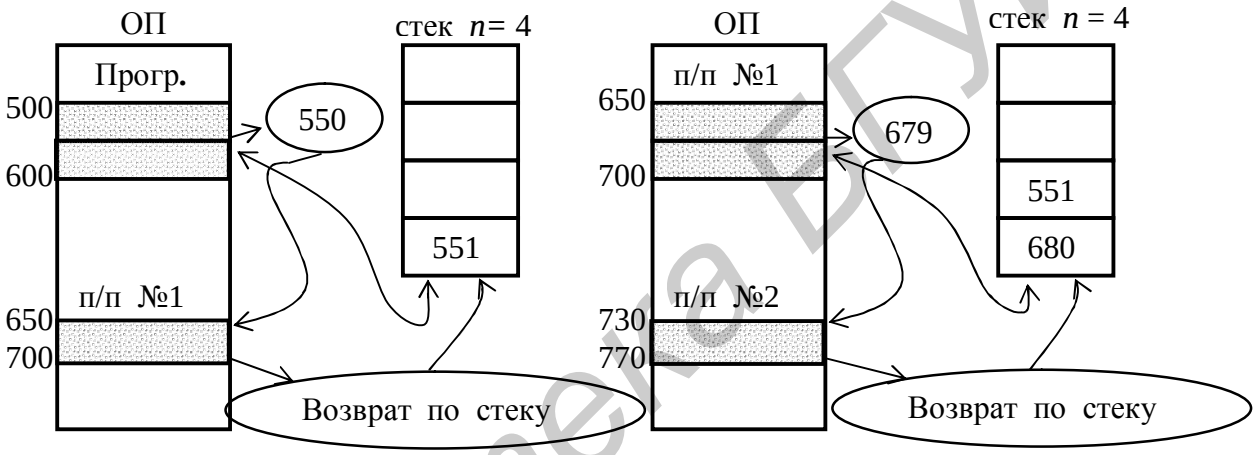


Рис. 1.25

1.6. Ассоциативная память

Типичная структура ассоциативной памяти имеет вид, показанный на рис. 1.26. Схема включает в свой состав:

RG АП – регистр ассоциативного признака, по которому осуществляется поиск в ЗУ;

ЗМ – запоминающий массив;

RG M – регистр маски, необходимый для выделения нужных разрядов в словах, расположенных в ЗМ;

RG I – регистр информации, в который пересылаются данные при записи в ЗУ или считывании;

КС – комбинационная схема, предназначенная для установки в единицу разрядов регистра совпадения *RG* СОВ, соответствующих номерам ячеек ЗМ, в которых произошло совпадение хранимой информации с ассоциативным признаком;

ФП – формирователь признаков α_i по содержимому регистра совпадения.

Причем:

a_0 =1, если искомые слова отсутствуют в ЗМ;

a_1 =1 – указывает на присутствие одного слова в массиве;

a_2 =1 – говорит о наличии в массиве нескольких слов с заданными ассоциативными признаками.

Поиск информации в АЗУ производится не по адресу, а по содержимому ячеек или, еще точнее, по ассоциативному признаку из *RG* АП. При этом обращение в ходе ассоциативного поиска происходит параллельно ко всем ячейкам массива. При чтении информации из ЗМ вначале по входной шине данных в *RG* АП в разряды 0…*n*-1 засылается *n*-разрядный ассоциативный запрос, а в регистр маски *RG M* в разряды, соответствующие коду поиска, помещаются значения «1». Для слов в ЗМ, в которых цифры в выделенных разрядах совпали с незамаскированными разрядами *RG* АП (маскирование выполняется нулем), схема КС устанавливает в единицу соответствующие разряды *RG* СОВ и в ноль – остальные разряды. Схема ФП формирует из слова, образовавшегося в *RG* СОВ, осведомительные сигналы a_i. В общем случае состав этих сигналов определяется разработчиками схемы АЗУ и может быть любым. Как правило, на практике формирование a_i по содержимому *RG* АП называют операцией контроля ассоциации.

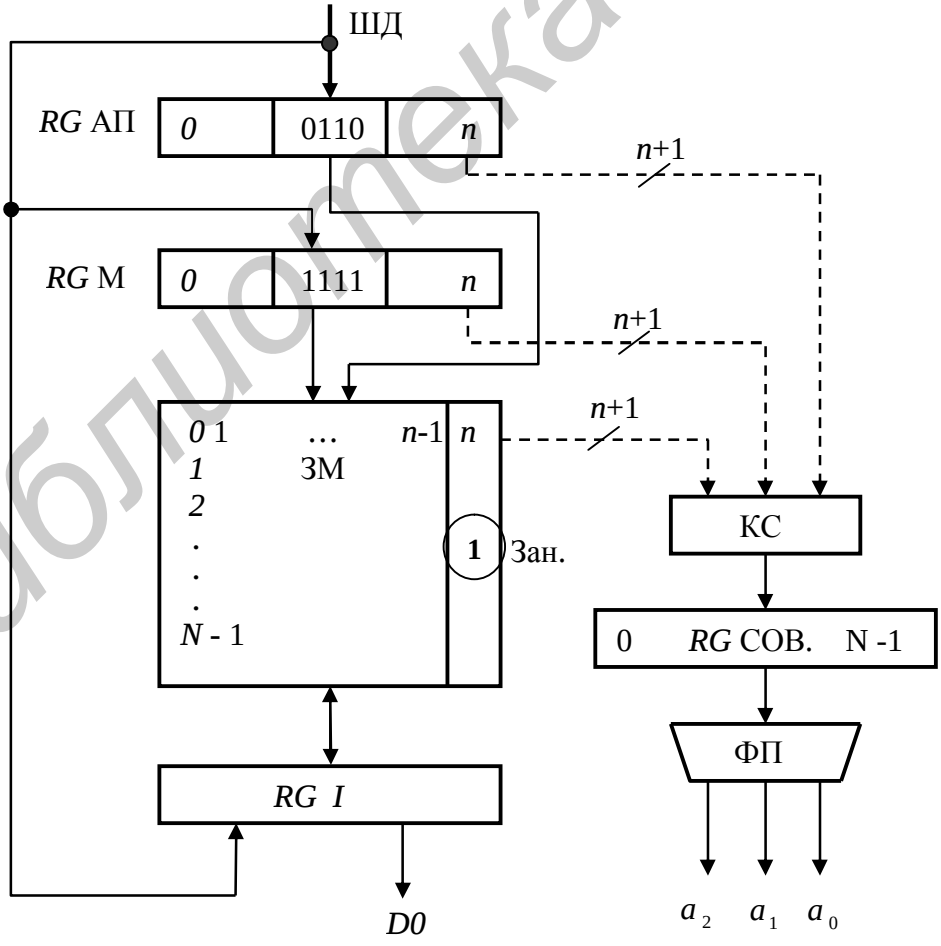


Рис. 1.26

Итак, в нашем случае если при контроле ассоциации получен признак a_0 = 1, то считывание отменяется из-за отсутствия искомой информации. При a_1 = 1 в *RG I* читается единственное найденное слово. Если в результате поиска получен признак a_2=1, то в регистр информации читается слово, имеющее наименьший номер среди номеров ячеек, отмеченных единицами в *RG* СОВ.

Разряд *n* запоминающего массива АЗУ имеет специальное назначение. В данном бите регистрируется факт занятости текущей ячейки ЗМ. Это обстоятельство определяет следующие действия системы при записи слова.

Во-первых, выполняется операция контроля ассоциации при <*RG* АП> = = *xx…x*0 и <*RG M*>= 00…01. При этом свободные ячейки отмечаются единицами в *RG* СОВ, а для записи автоматически выбирается ячейка с наименьшим номером.

Если свободных ячеек нет, то запись в АЗУ не производится и программистам следует пересмотреть правила заполнения *ЗМ*, уменьшив при этом количество слов или переформировав весь массив.

1.7. Организация многоуровневой памяти

Память современных компьютеров имеет многоуровневую структуру. Причем чем выше уровень, тем выше быстродействие и меньше емкость ЗУ. К верхнему уровню относят регистровые, ассоциативные ЗУ и ОП, то есть те ЗУ, с которыми процессор непосредственно взаимодействует в ходе вычислений.

ВЗУ – это память нижнего уровня. Она обладает малым быстродействием, и, следовательно, в процессе выполнения программ пользователи должны как можно реже обращаться к ВЗУ, чтобы не утратить эффективности (скорости) расчетов.

В сложных ВС могут одновременно выполняться несколько задач. Это порождает необходимость в обеспечении сохранности и неприкосновенности частных программных продуктов от команд хакеров, то есть выполнить защиту памяти.

1.7.1. Защита памяти

Один из методов защиты памяти предполагает использование граничных регистров для хранения адресной информации или прямых адресов границ защищаемой области ОЗУ. Эти регистры получили название граничных и обозначаются: *RG н.гр.* – регистр нижней границы и *RG в.гр.* – регистр верхней границы. Структурная схема блока адресации с граничными регистрами имеет вид, показанный на рис. 1.27.

При обращении к ОЗУ $A_{исп}$ проверяется на принадлежность к заданной области памяти. Если выхода за границы защищаемого пространства нет, то схемы сравнения формируют на выходах «*b*» единичный логический потенциал, разрешающий обращение к ЗУ. При выходе адреса за установленные границы одна из схем сравнения устанавливает на выходе «*b*» логический ноль, а на

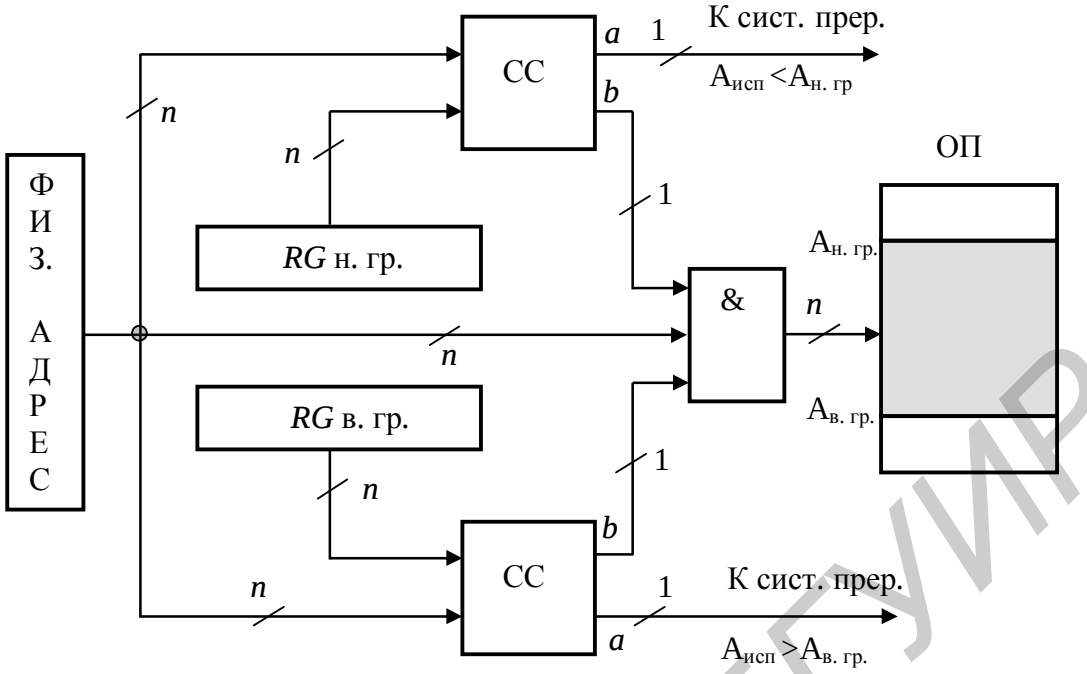


Рис. 1.27

выходе «*a*» – логическую единицу. Обращение к ОЗУ запрещается, а в системе формируется сигнал прерывания, передаваемый на вход устройства управления.

Более гибким по отношению к методу граничных регистров является метод ключей защиты. Он позволяет выполнить защиту программы, части которой расположены в различных разделах памяти. В данном случае ОП делится на одинаковые блоки, число которых равно:

$$C=2^r,$$

где r – длина поля «код блока» в регистре адреса.

Схема устройства, реализующая описанный способ защиты, приведена на рис. 1.28. В состав блока входят: модуль АЗУ и элемент 2ИЛИ.

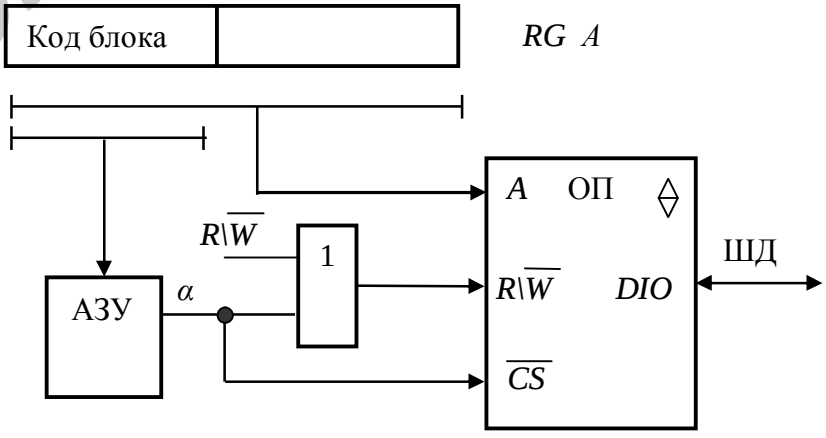


Рис. 1.28

Перед исполнением программ в ассоциативное ЗУ должны быть помещены коды блоков, закрепленные за соответствующими пользователями. При произвольном обращении к ОП старшие разряды адреса (код блока) передаются на вход АЗУ, где происходит их сравнение с кодами блоков защищаемых программ. Если код блока из *RG A* и код блока в АЗУ не совпали, то формируется признак $a = 1$, который запрещает запись в ОП (защита от записи) и отключает шину *DIO* от шины данных компьютера (защита от чтения).

Если код блока из *RG A* совпал с хранимой в АЗУ информацией, то признак a устанавливается в 0, в результате чего разрешается воздействие линий ШУ на вход $R/\overline{W}$ ОП. Одновременно линии *DIO* ЗУ подключаются к ШД компьютера.

Существуют также блоки защиты, в которых режим обращения к памяти может быть санкционирован с помощью специального триггера, называемого триггером «режима обращения» $T_{р.о}$. При этом организуется:

1) защита от записи и считывания,

2) защита от записи с возможностью считывания.

Схема обращения к ЗУ, реализующая описанный способ защиты, показана на рис. 1.29.

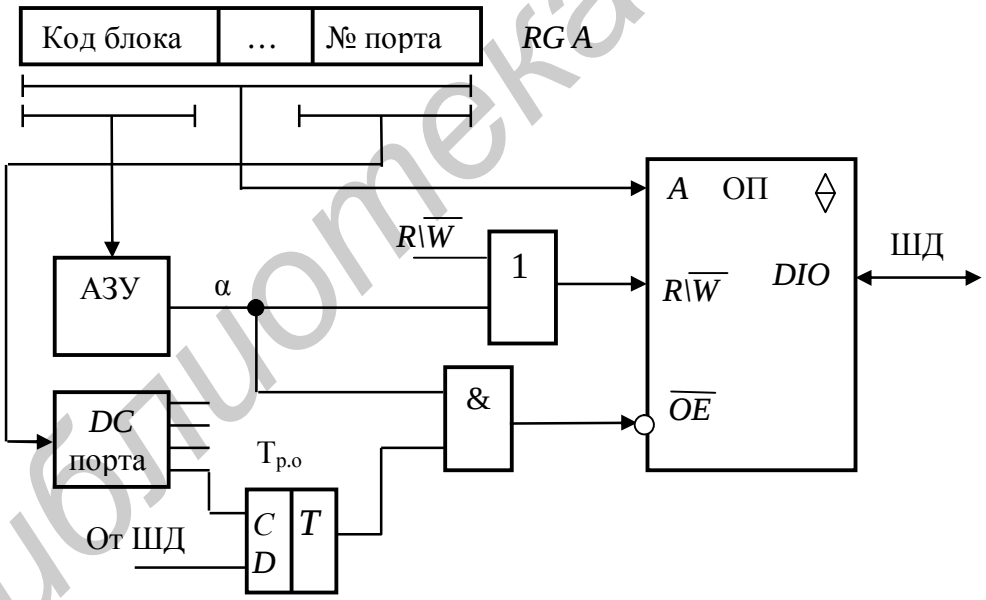


Рис. 1.29

Организация первого режима осуществляется программной установкой триггера $T_{р.о}$ в состояние 1. При этом, если:

$a = 0$, то $R/\overline{W}$ =ШУ, $\overline{OE}$ =0 и в схеме реализуется рабочий режим;

$a = 1$, то $R/\overline{W}$ =1, $\overline{OE}$ =1 и имеет место режим защиты, а шина *DIO* переводится в третье состояние.

Второй режим имеет место при установке триггера $T_{р.о}$ в состояние нуля.

При этом, если:

a = 0, то $R/\overline{W}$ = ШУ, $\overline{OE}$ = 0 и в схеме организуется рабочий режим;

a = 1, то $R/\overline{W}$ = 1, $\overline{OE}$ = 0 и имеет место режим чтения с защитой от записи.

Использование схем защиты в общем случае предполагает распределение памяти между программами с помощью специальных операционных средств. Пользователь только принимает решение: защищать или нет данную область. Специальная программа компьютера выполняет расчет, запоминание ключей и установку $T_{p.o}$ с помощью имеющегося набора своих команд. В противном случае хакер имеет возможность самостоятельно изменить ключ или, записав его в АЗУ, несанкционировано воспользоваться памятью.

Вместо дорогостоящего АЗУ для хранения ключей может быть использовано регистровое ЗУ или так называемое ЗУ ключей (рис. 1.30).

В приведенной схеме регистр адреса содержит поле «код блока» и поле «ключ». В процессе обращения к памяти поле «код блока» передается на адресный вход регистрового ЗУ и инициирует чтение ключа на вход схемы сравнения (СС). Содержимое поля «ключ» регистра *RG A* сравнивается со считанным значением и в случае совпадения СС формирует сигнал управления $a = 0$. Оперативная память при этом оказывается доступна для чтения или записи, а при необходимости программной установкой $T_{p.o}$ в состояние 0 организуется защищенный режим «только чтение».

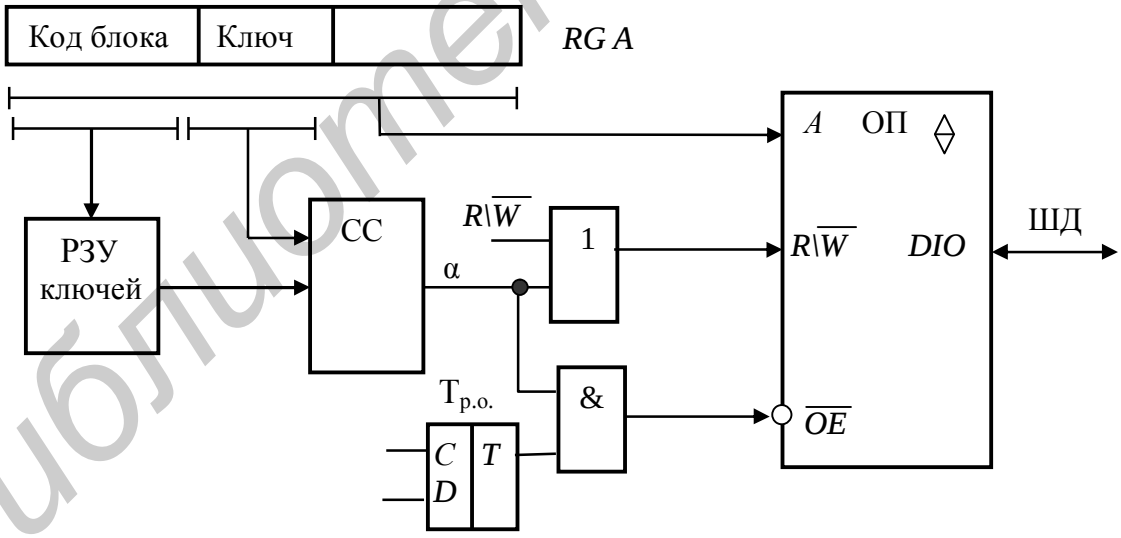


Рис. 1.30

Однако применение указанной схемы оказывается не всегда эффективно из-за большой разрядности регистра адреса *RG A*. Необходимость хранения ключа в каждой команде соответственно обусловливает и дополнительный объем аппаратных средств на реализацию памяти. В связи с этим в реальных схемах ключ помещается в специальный регистр, в котором и сохраняется на все время выполнения программы пользователя (рис. 1.31).

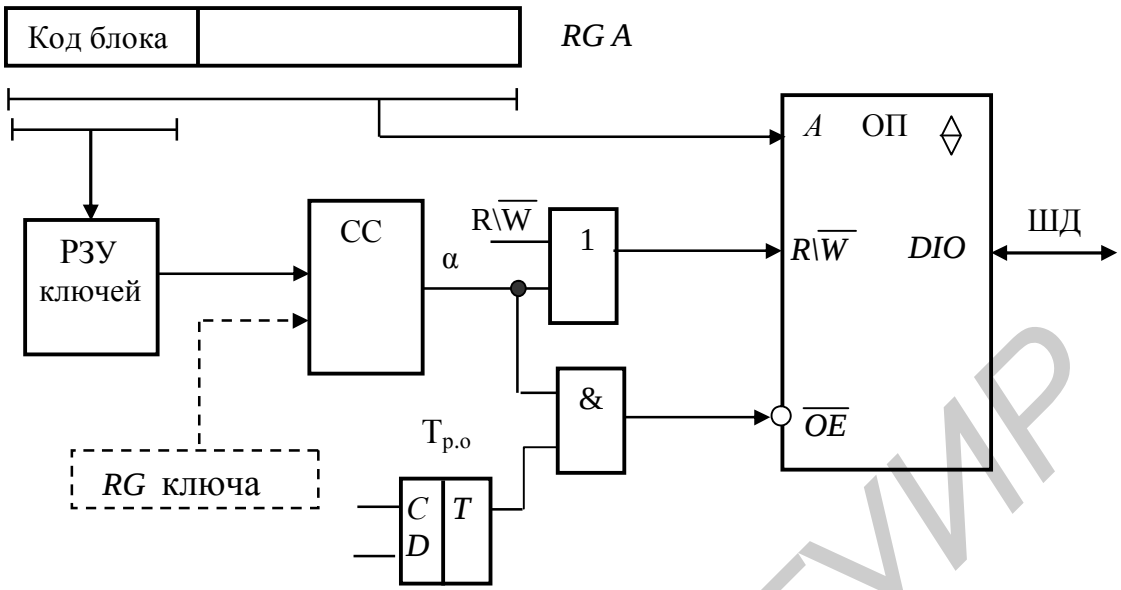


Рис. 1.31

Замечание. Содержимое *RG* ключа при прерываниях включается в состав слова состояния программы и эвакуируется в стек. В свою очередь стековая область ЗУ должна быть защищена с помощью специальных программных средств операционной системы компьютера. Если данное требование не выполняется, то программы, защищаемые пользователями, могут быть несанкционировано прочитаны программистами, имеющими более высокую квалификацию.

1.7.2. Динамическое распределение памяти

Эффективное распределение памяти между программами не может быть выполнено до «пуска» вычислительного процесса, то есть статически. Это обусловлено тем, что задачи пользователей в ходе вычислений требуют непредсказуемых объемов памяти, которые определяются массивами промежуточных результатов и формой представления окончательных решений. Таким образом, секционирование ЗУ должно выполняться динамически, то есть непосредственно в ходе решения задач или отработки заданий в системе.

Один из способов динамического распределения памяти основывается на использовании базовых регистров. При этом каждой программе специальные операционные средства ставят в соответствие свой базовый адрес, а при исполнении программ физические адреса образуются путем суммирования содержимого базовых регистров и смещения, задаваемого в той или иной форме. Кроме того, в ходе вычислений свободная память компьютера может состоять из нескольких несвязных областей или фрагментов памяти. Это приводит к невозможности размещения в ОЗУ других программ и задач, обрабатываемых, например, в многопрограммном режиме. В связи с изложенными обстоятельствами при вводе каждого очередного задания необходимо вы-

полнять сжатие оперативной памяти, что в общем случае сопровождается назначением новых базовых адресов с учетом нового размещения программ в ОП.

Недостаток способа – большие затраты времени на редактирование связей в перемещаемых модулях или новую загрузку программ из ВЗУ.

От данных недостатков свободен процесс вычислений в компьютерах, имеющих виртуальную организацию памяти.

1.7.3. Виртуальная память

Виртуальная память – это способ организации ОЗУ и ВЗУ в систему, при котором программист имеет дело не с конкретным запоминающим устройством, а с кажущейся одноуровневой компьютерной памятью. Объем этой памяти задается длиной адресных полей в командах и разрядностью используемых базовых регистров. Теперь в процессе программирования пользователь имеет дело с виртуальными адресами, и лишь при исполнении конкретной команды выполняется автоматическое преобразование виртуальных адресов в реальные адреса ОЗУ и ВЗУ.

Для упрощения преобразования виртуальных адресов вся память разбивается на блоки или страницы одинакового размера. Страницам виртуальной и физической памяти присваивают свои номера, а по объему каждая физическая страница соответствует (или равна) одной виртуальной странице.

Страничная организация дает ряд преимуществ программисту. Так, например, загружаемая программа может быть записана на любую свободную страницу физической памяти, причем порядок расположения страниц в ЗУ в данном случае не имеет никакого значения.

Соответствие между физическими и виртуальными страницами устанавливается с помощью страничной таблицы (рис. 1.32). Эта таблица хранится в служебной области ОЗУ, и для каждой достаточно большой программы составляется своя таблица.

Виртуальные страницы программы в текущий момент времени могут перемещаться из ОП в ВЗУ и наоборот. Поэтому страничная таблица под управлением операционных средств должна перерабатываться и уточняться при каждом изменении в распределении памяти.

Основной принцип образования физических адресов из виртуальных заключается в следующем. Из виртуального адреса извлекается номер виртуальной страницы, который далее используется для входа в страничную таблицу. Из таблицы извлекается номер физической страницы в ОП или в ВЗУ и далее вместе с номером ячейки памяти, взятым из виртуального адреса, образует физический адрес, по которому происходит фактическое обращение. При выполнении каждой очередной программы в оперативную память загружается начальная страница и ей передается управление. Далее по мере надобности недостающие страницы подгружаются из ВЗУ в ОП с последующей коррекцией страничной таблицы.

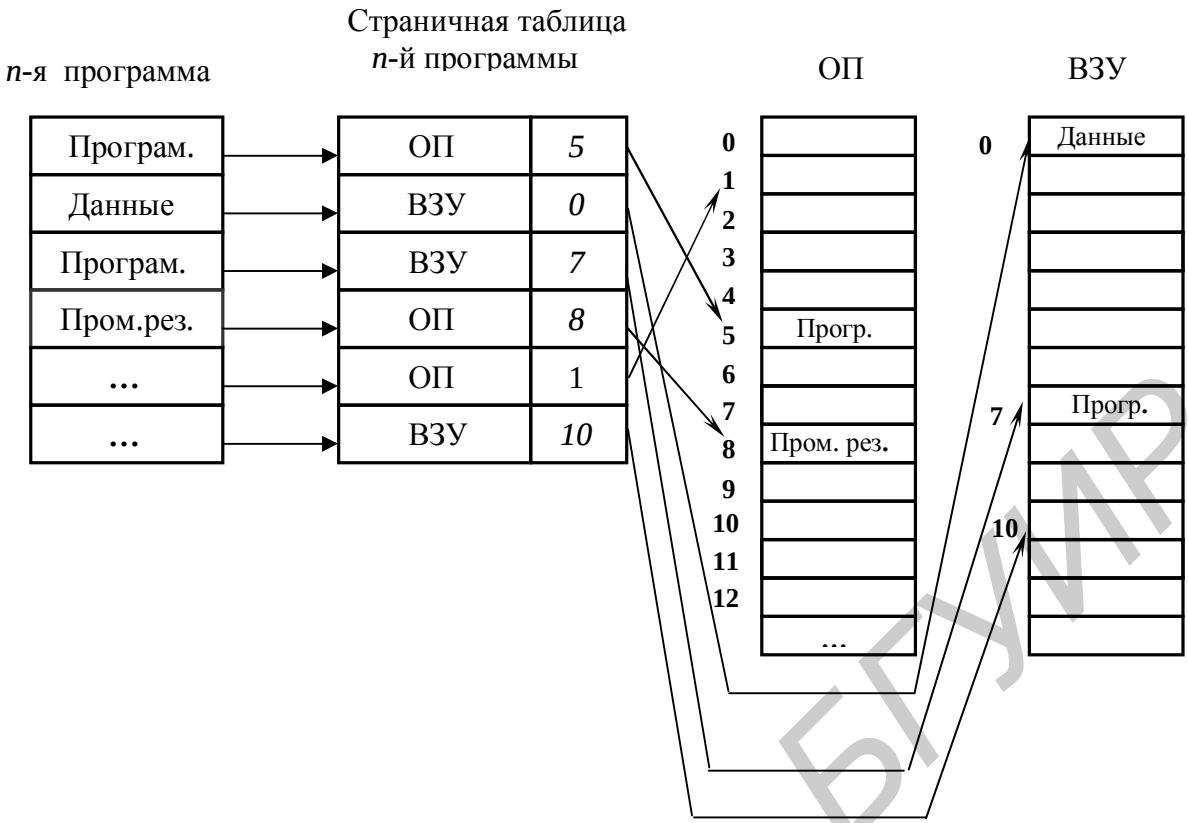


Рис. 1.32

Для ускорения преобразования адресов может быть использована небольшая сверхоперативная память, куда из ОП передается страничная таблица выполняемой программы. Например, в схеме, приведенной на рис. 1.33, номер виртуальной страницы (старшие разряды) передается на адресные входы СОЗУ. При этом на выход *DO* считывается номер физической страницы и признак *p*, характеризующий тип памяти, хранящей страницу. Исполнительный адрес формируется путем составления номера физической страницы и номера байта из регистра виртуального адреса *RG VA* в одно адресное слово.

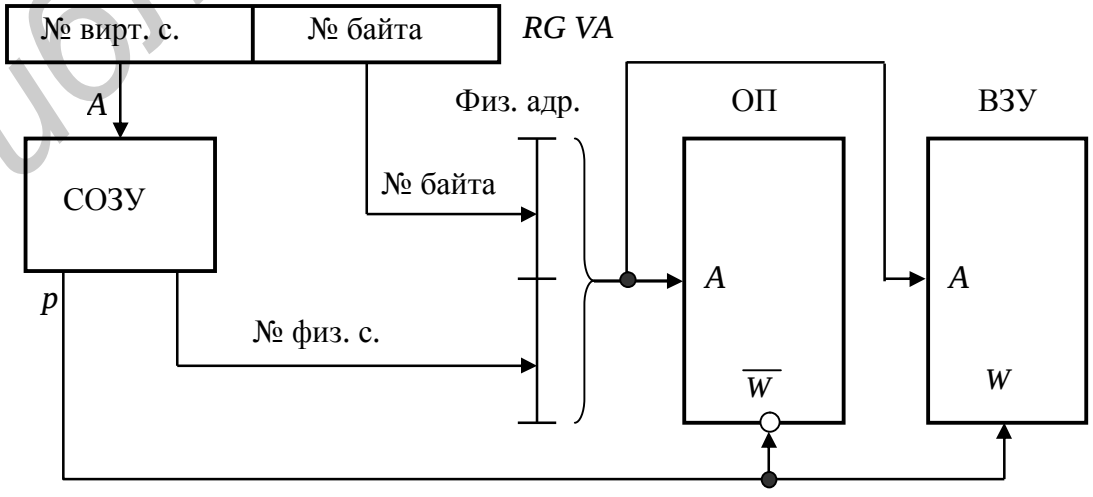


Рис. 1.33

Если признак $p = 0$, то информация читается из ОП, при $p = 1$ страница находится в ВЗУ и система формирует соответствующее обращение через блок УВВ.

Очевидно, что схема, приведенная на рис. 1.33, только поясняет принцип преобразования адресов. Реально обращение к ВЗУ более сложная процедура.

В мультипрограммном режиме в СОЗУ могут размещаться страничные таблицы или их фрагменты для различных программ. В этом случае блок преобразования адресов удобно строить с использованием схемы ассоциативной памяти (рис. 1.34). В данной реализации схема АЗУ содержит информацию о номерах виртуальных и соответствующих им физических страницах. Причем для различных программ хранятся свои таблицы или их фрагменты, которые «адресуются» полями: «номер программы» и «номер страницы». Данные составляющие виртуального адреса передаются на вход АЗУ, где выполняется поиск по соответствующему ассоциативному признаку. В результате совпадения входной информации с содержимым некоторой ячейки на выходную шину памяти выдается физический адрес страницы, находящийся в ОП или ВЗУ. Принадлежность адреса к тому или иному устройству характеризует признак *p*.

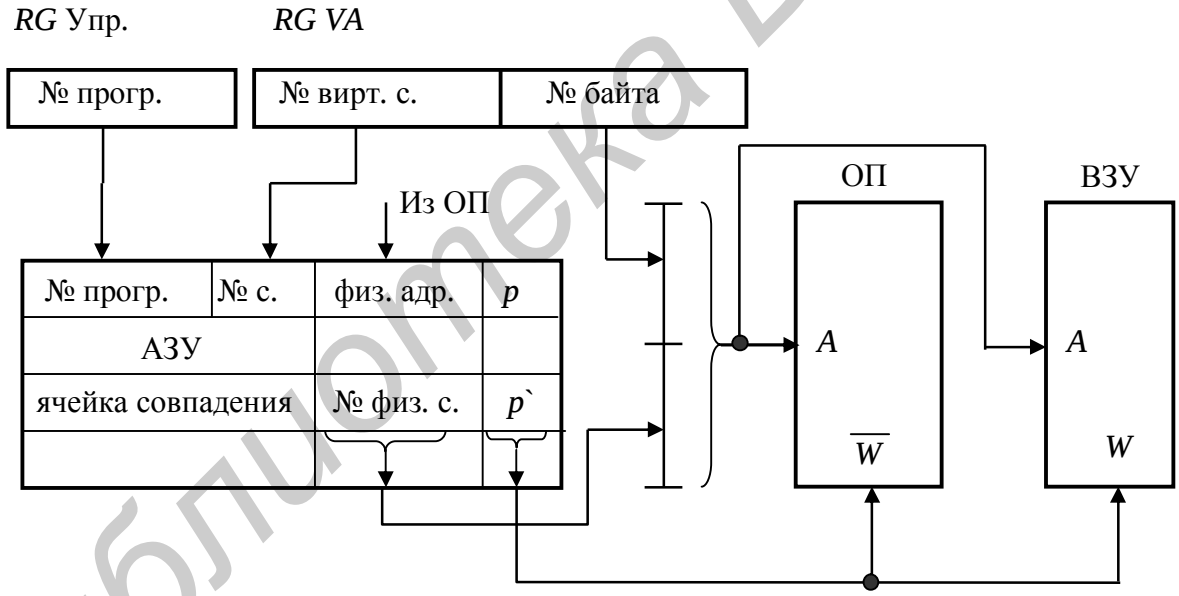


Рис. 1.34

В случае необходимости кроме признака *p* ассоциативная память может формировать и ряд дополнительных сигналов, назначение которых определяется при проектировании разработчиком схемы.

1.7.4. Принцип удаления страниц из ОП

В процессе исполнения программ недостающие страницы заданий по мере надобности могут загружаться из ВЗУ в ОП. Однако на некотором этапе вы-

числений оперативная память оказывается полностью загруженной, и возникает проблема удаления неактивно используемых фрагментов программ, то есть стирания их путем записи активных страниц.

Простейший вариант данной процедуры предполагает замену частей задания без использования какой-либо дополнительной аппаратуры. При этом, как правило, применяется стратегия случайного поиска, что, очевидно, может привести к явным промахам – в результате замены будет удалена часто используемая страница.

Другой подход к решению задачи «выбора» основывается на использовании принципа регистрации активности страниц. При этом каждое очередное обращение программы к некоторому фрагменту определяется как процесс активизации страницы, а сам факт обращения регистрируется специальной схемой (рис. 1.35).

Приведенная схема содержит: АЗУ, *DC* физической страницы, регистр на триггерах *RS*-типа, комбинационную схему КС и логические элементы, используемые для сброса триггеров активности.

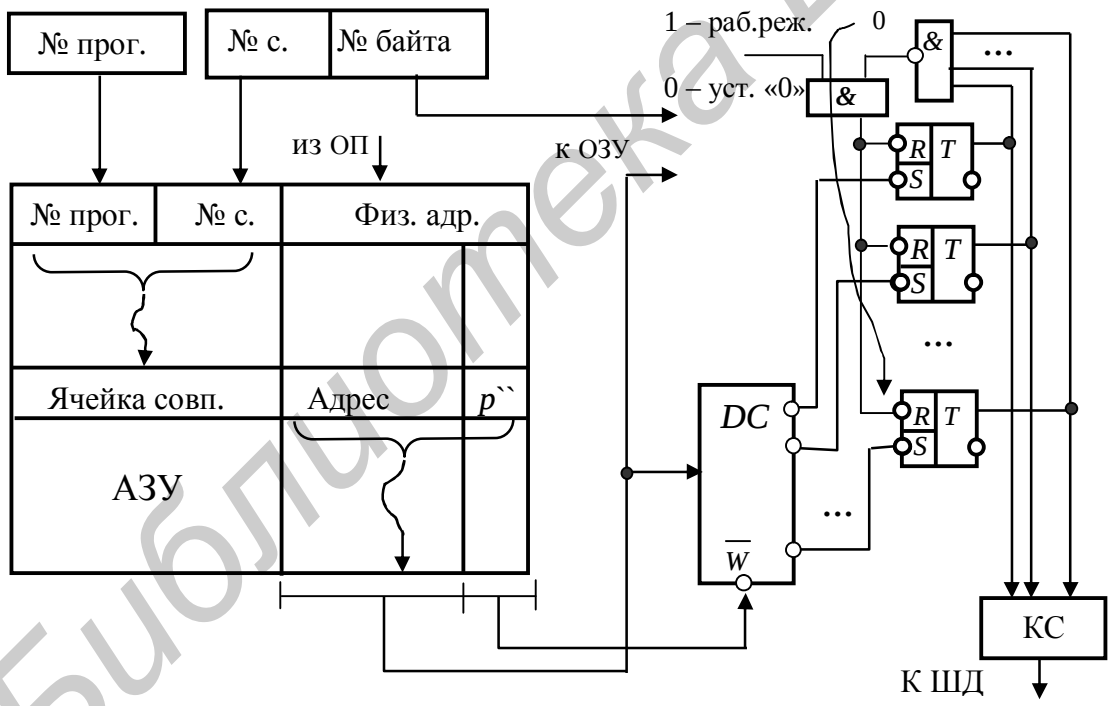


Рис. 1.35

DC физической страницы включается при каждом обращении к оперативной памяти. При этом его выходы из состояния 11…1 переключаются в рабочий режим, а один из выходных сигналов, воздействуя на инверсный *S*-вход триггера активизируемой страницы, устанавливает элемент памяти в состояние «1». Это позволяет в произвольный момент времени получать информацию как о страницах, отмеченных единицами в регистре активности, так и о страницах,

еще не использовавшихся в процессе вычислений. Если в результате длительных расчетов оказалось, что все страницы использовались хотя бы по разу, то многовходовой элемент «И-НЕ» переключится в рабочее состояние и сформирует на своем выходе уровень логического нуля. Этот сигнал установит весь регистр активности в нулевое состояние, и процесс регистрации обращений к ОЗУ начнется заново.

При необходимости замены *N*-й страницы её номер, кодированный схемой КС, передается в процессор и далее обрабатывается программой обслуживания виртуальной памяти.

1.7.5. Сегментно-страничная организация памяти

Программы, выполняемые на компьютере, обычно состоят из нескольких массивов информации, например: текста программы, набора исходных данных, содержимого регистров, сохраняемого при прерываниях, и т.д. В связи с этим виртуальная память каждой программы также делится на отдельные части или сегменты с независимой адресацией внутри сегмента. Соответственно при формировании исполнительных адресов появляется определенная иерархия: адрес сегмента – номер страницы – номер байта. Этой иерархии в свою очередь ставится в соответствие иерархия таблиц, необходимых для преобразования виртуальных адресов в физические.

В режиме загрузки программы операционные средства системы строят программную таблицу, которая, в частности, содержит начальные адреса сегментных таблиц для соответствующих сегментов памяти. Схема преобразования адресов при этом имеет вид, показанный на рис. 1.36.

При выполнении программы адрес начала сегментной таблицы текущей программы A_s читается из программной таблицы в регистр управления. Этот адрес и содержимое поля *s* регистра виртуального адреса суммируются и образуют строку сегментной таблицы, в которой размещается начальный адрес страничной таблицы в сегменте *s*. На следующем этапе считанное значение суммируется с номером виртуальной страницы, образуя при этом адрес строки страничной таблицы. По этому адресу из ОЗУ читается номер физической страницы, и если эта страница оказывается в ОП, то адрес искомой информации формируется путем составления номера физической страницы и номера байта, взятого из регистра *RG VA*.

Если требуемая страница оказывается в ВЗУ, то в общем случае происходит перезапись информации в ОЗУ и коррекция страничной таблицы специальной программой. Очевидный недостаток метода – многократное обращение к ОЗУ и большая длительность формирования исполнительных адресов.

Указанный недостаток может быть устранен путем применения в системе специального блока АЗУ, ориентированного на поддержку интерфейса «процессор–память» (рис. 1.37). В данном случае дополнительный блок ЗУ должен хранить информацию о наиболее активных страницах одной или нескольких исполняемых программ.

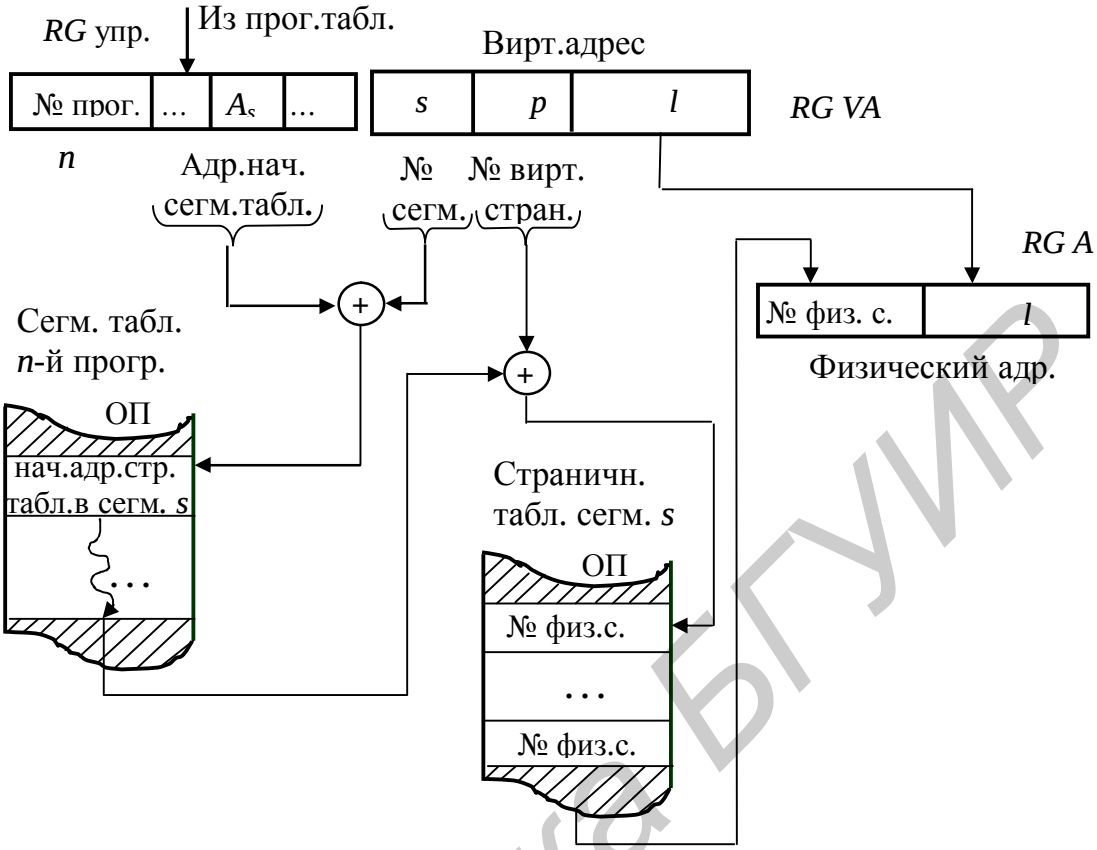


Рис. 1.36

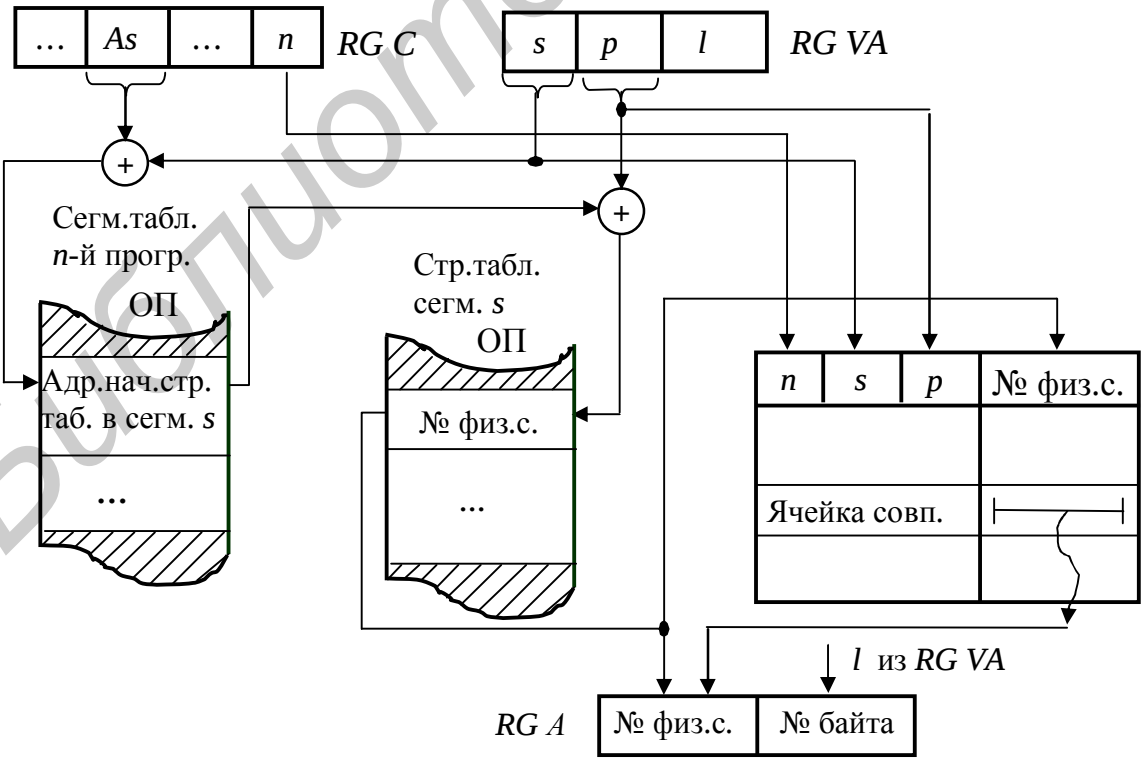


Рис. 1.37

Преобразование адресов в указанной схеме всегда начинается по двум направлениям: 1) выполняется просмотр информации в АЗУ по ассоциативному признаку *АП* = «*n, s, p*» и 2) организуется обращение к ОП по стандартному алгоритму. Если «адресуемый» № физ.с. находится в АЗУ, то дальнейшее обращение к ОП автоматически отменяется и слово читается в *RG A*. Если же информация, к которой происходит обращение, в АЗУ не обнаружена, то формирование физического адреса продолжается по известному алгоритму.

1.7.6. Применение АЗУ для ускорения процессов обмена с памятью

Для ускорения выборки операндов из памяти в вычислительных системах используются специальные буферные ЗУ. Это позволяет в процессе выполнения программ осуществить подмену обращений к ОП обращениями к более быстродействующему БЗУ (как правило АЗУ), содержащему активные составляющие программы. Подмена ячеек ОП ячейками более быстродействующей схемы предполагает определение степени активности информации, используемой задачей. Это необходимо для пересылки слова из ОП в БЗУ с тем, чтобы последующее обращение к данному слову обслуживалось посредством БЗУ. Схема, реализующая соответствующий подход на базе ассоциативного ЗУ, приведена на рис. 1.38.

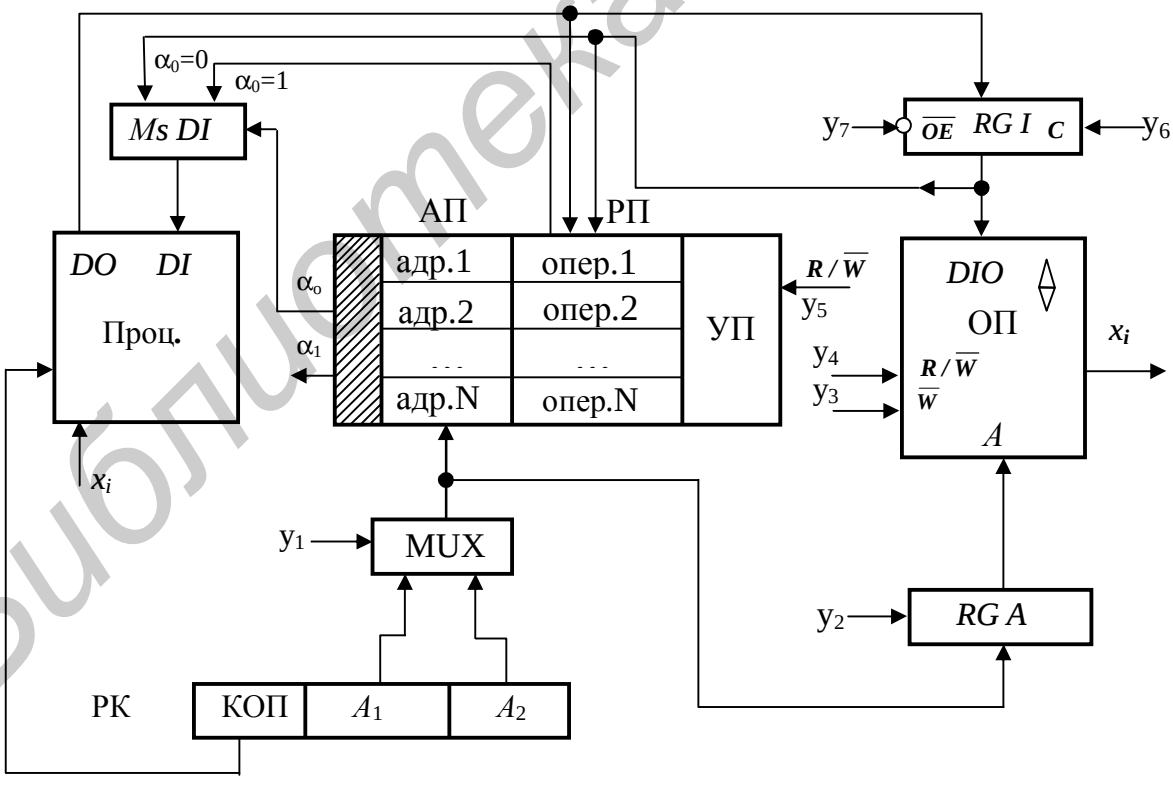


Рис. 1.38

В составе используемого АЗУ выделяют: АП – адресное поле, РП – рабочее поле, УП – управляющее поле.

В N ячейках адресного поля хранятся N адресов оперативной памяти, а в соответствующих регистрах рабочего поля N операндов. Обращение к ОЗУ начинается с передачи адреса из *РК* через *MUX A* на входы адресного поля АЗУ. Одновременно исполнительный адрес передаётся на адресные входы ОП через регистр адреса. Если искомое значение находится в ассоциативной памяти, то схема формирует признак $a_0 = 1$, что позволяет передать данные через мультиплексор данных *MUX DI* на вход *DI* процессора. При $a_0 = 0$ выполняется чтение операнда из ОП.

Если в процессе выполнения программы один из операндов, расположенный в ОП, оказывается достаточно активен, то выполняется его перемещение в КЭШ ЗУ (АЗУ) после анализа активности содержимого кэша. Активный операнд может быть переписан также и в свободную ячейку АЗУ. При этом о наличии свободных мест свидетельствует признак $a_1 = 0$. Если свободных мест нет, то признак $a_1 = 1$.

Для разрешения конфликтной ситуации с $a_1 = 1$ необходимо выбрать определенный алгоритм, ориентированный на замену слов, записанных в АЗУ. Наиболее часто с этой целью используют управляющее поле УП самого кэша, сложность которого определяет точность решения задачи. Очевидно, что при наличии в структуре УП N двоичных счетчиков можно получить 100 %-ную точность в определении степени активности данных. Однако стоимость АЗУ при этом оказывается достаточно большой, и разработчику памяти на практике приходится использовать некоторые приближенные алгоритмы анализа активности.

В современных процессорах (например P3) с высокой производительностью все чаще используют двухуровневые кэши. При этом кэш первого уровня состоит из подсистемы команд объемом 16 Кб и такого же кэша данных. Последний имеет четырехканальную множественно-ассоциативную организацию, что означает разбиение пространства памяти на страницы и объединение страниц в подмножества. Такой подход к организации кэш ЗУ уменьшает время ассоциативного поиска, поскольку область сканирования уменьшается за счет предварительного просмотра кодов блоков. Соответственно число страниц в множестве в данном случае равно четырем, что указывает на внутреннюю канальность кэша. При записи информации в данную подсистему используются различные протоколы взаимодействия.

Кэш команд имеет четырехканальную множественно-ассоциативную организацию. Поскольку в ходе выполнения программ команды не модифицируются, особая стратегия записи для данного кэша не нужна.

Кэш второго уровня имеет гораздо больший объем. В нем содержатся команды и данные. Этот кэш соединен с остальной частью системы через шинный интерфейс компьютера.

2. УСТРОЙСТВА УПРАВЛЕНИЯ КОМПЬЮТЕРОВ

2.1. Управляющие автоматы с жесткой логикой

К устройствам управления с жесткой логикой (с неизменяемым алгоритмом управления) относят микропрограммные автоматы (МПА) Мили и Мура. Эти устройства позволяют сформировать как импульсные, так и потенциальные сигналы микроопераций, необходимые для управления вычислительным процессом в операционном автомате (ОА).

2.1.1. Синтез МПА Мили по ГСА

Теоретически закон функционирования автомата Мили задается функцией переходов δ и функцией выходов λ. Функция переходов δ определяет состояние автомата в момент времени $t+1$ по отношению к моменту времени t. Иными словами, функция δ ставит в соответствие паре параметров «состояние–входной сигнал» a_m, x_i – состояние автомата a_s, в которое он переходит из состояния a_m под действием сигнала x_i, то есть $a_s = \delta(a_m, x_i)$. Функция выходов λ ставит в соответствие паре параметров «состояние–входной сигнал» a_m, x_i – выходной сигнал y_j, который определяется функцией вида $y_j = \lambda(a_m, x_i)$.

Рассмотрим методику синтеза МПА Мили по ГСА, представленной на рис. 2.1. В данном примере автомат имеет 4 логических условия x_1, x_2, x_3, x_4 и формирует 5 сигналов микроопераций y_1, y_2, y_3, y_4, y_5:

$$X = \{x_1, x_2, x_3, x_4\},$$
$$Y = \{y_1, y_2, y_3, y_4, y_5\}.$$

Таким образом, автомат имеет 4 входа и 5 управляющих выходов.

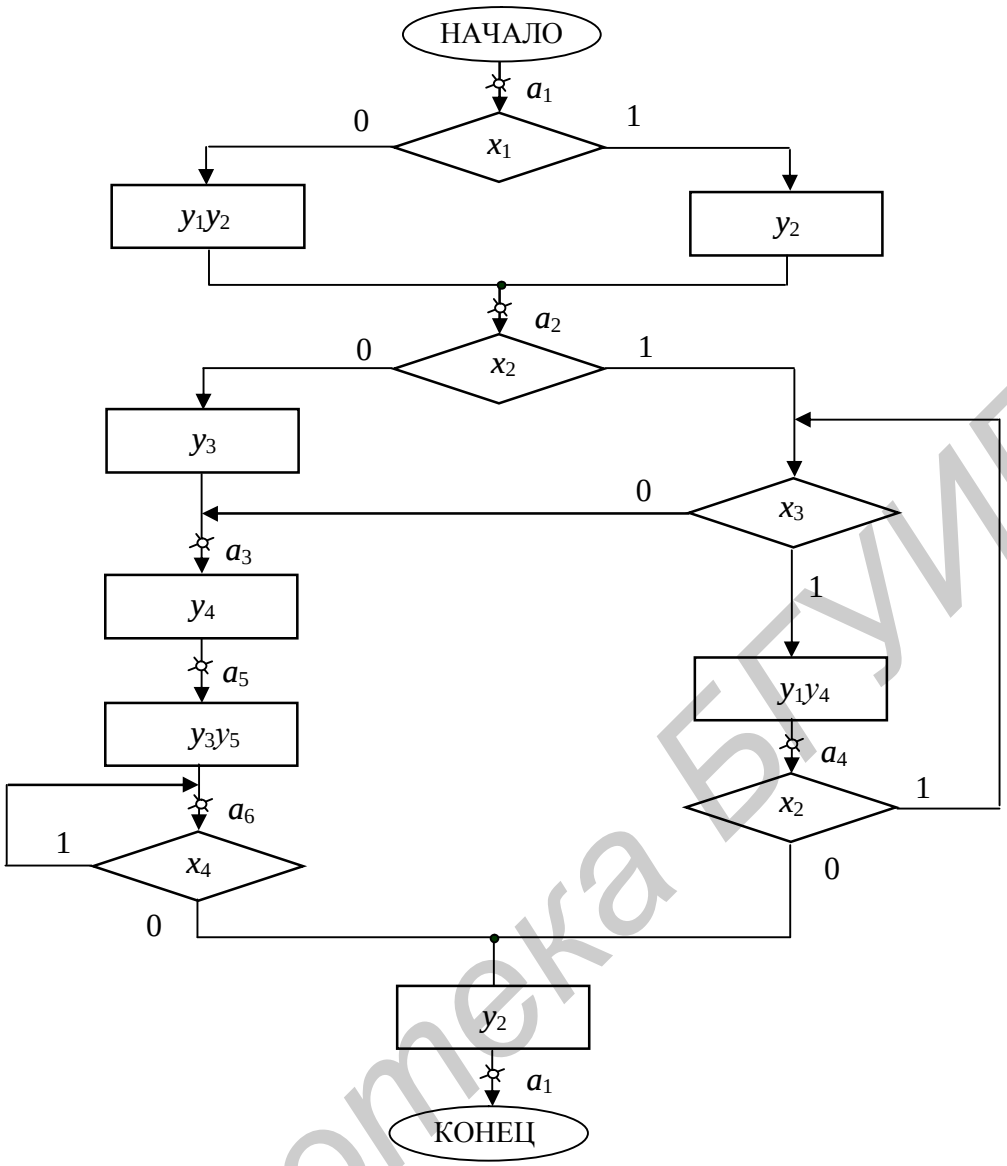


Рис. 2.1

Классическая методика синтеза автомата Мили состоит из следующих этапов проектирования. На первом этапе выполняется разметка ГСА по правилам:

а) символом a_1 отмечается выход начальной и вход конечной вершин;

б) входы вершин, следующих за операторными, отмечаются символами a_2, a_3 и т. д.;

в) входы различных вершин должны иметь различные метки;

г) вход вершины может отмечаться только одним символом.

Предполагается, что в начальном состоянии входные сигналы y_i имеют нулевые значения. По окончании функционирования автомат возвращается в исходное состояние a_1 и останавливается. Для запуска автомата в реальных системах используется специальный сигнал A (рис. 2.2), который относится к группе входных сигналов. Если $A = 0$, то МПА находится в ждущем режиме,

при A=1 начинается выполнение программы формирования управляющих сигналов.

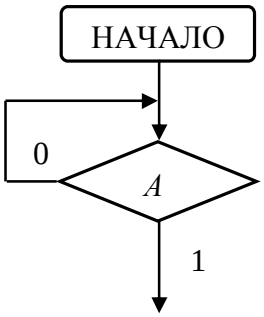


Рис. 2.2

На втором этапе синтеза строится структурная таблица переходов с учетом выполненной разметки (табл. 2.1).

Таблица 2.1

a_m	$K(a_m)$	a_s	$K(a_s)$	$X(a_m a_s)$	$Y(a_m, a_s)$	$F(a_m a_s)$
a_1	000	a_1	000	$\overline{A}$	-	-
		a_2	001	$A\overline{x}_1$	y_1y_2	D_3
		a_2	001	Ax_1	y_2	D_3
a_2	001	a_3	010	$\overline{x}_2$	y_3	D_2
		a_3	010	$x_2\overline{x}_3$	-	D_2
		a_4	011	x_2x_3	y_1y_4	D_2D_3
a_3	010	a_5	100	1	y_4	D_1
a_4	011	a_1	000	$\overline{x}_2$	y_2	-
		a_3	010	$x_2\overline{x}_3$	-	D_2
		a_4	011	x_2x_3	y_1y_4	D_2D_3
a_5	100	a_6	101	1	y_3y_5	D_1D_3
a_6	101	a_6	101	x_4	-	D_1D_3
		a_1	000	$\overline{x}_4$	y_2	-

На третьем этапе выполняется кодирование состояний автомата, занесенных в таблицу переходов. Для этого используется двоичный код длиной

$$R = \text{int}\log_2 M,$$

где R – число элементов памяти МПА,

M – число состояний автомата или число операторных вершин исходной граф-схемы. В нашем примере, очевидно,

$R = \text{int}\log_2 6 = 3$.

Выберем в качестве запоминающих элементов триггеры D-типа. Функции возбуждения в этом случае ставятся в соответствии с правилом: переход элемента памяти вида 11 или 01 определяет единичное значение функции $F(a_m a_s)$

и отмечается символом D_i в i-м разряде кода, переходы 00 и 10 отмечаются прочерком. Иными словами функции возбуждения для D-триггеров ставятся в соответствии с единичными значениями в колонке $K(a_s)$ табл. 2.1.

На четвертом этапе выписываются логические уравнения для функций возбуждения (строится схема $КС_1$) и выходных переменных (схема $КС_2$):

$$\begin{cases} D_1 = a_3 \vee a_5 \vee a_6 x_4, \\ D_2 = a_2 \overline{x_2} \vee a_2 x_2 \overline{x_3} \vee a_2 x_2 x_3 \vee a_4 x_2 \overline{x_3} \vee a_4 x_2 x_3, \\ D_3 = a_1 A \overline{x_1} \vee a_1 A x_1 \vee a_2 x_2 x_3 \vee a_4 x_2 x_3 \vee a_5 \vee a_6 x_4. \end{cases}$$

После минимизации сложность схемы может быть уменьшена:

$$\begin{cases} D_1 = a_3 \vee a_5 \vee a_6 x_4, \\ D_2 = a_2 \vee a_4 x_2, \\ D_3 = a_1 A \vee a_2 x_2 x_3 \vee a_4 x_2 x_3 \vee a_5 \vee a_6 x_4. \end{cases}$$

Функции выходов автомата описываются следующими соотношениями:

$$\begin{cases} y_1 = a_1 A \overline{x_1} \vee a_2 x_2 x_3 \vee a_4 x_2 x_3, \\ y_2 = a_1 A \vee a_4 \overline{x_2} \vee a_6 \overline{x_4}, \\ y_3 = a_2 \overline{x_2} \vee a_5, \\ y_4 = a_2 x_2 x_3 \vee a_3 \vee a_4 x_2 x_3, \\ y_5 = a_5. \end{cases}$$

По полученным логическим соотношениям строится функциональная схема МПА (рис. 2.3).

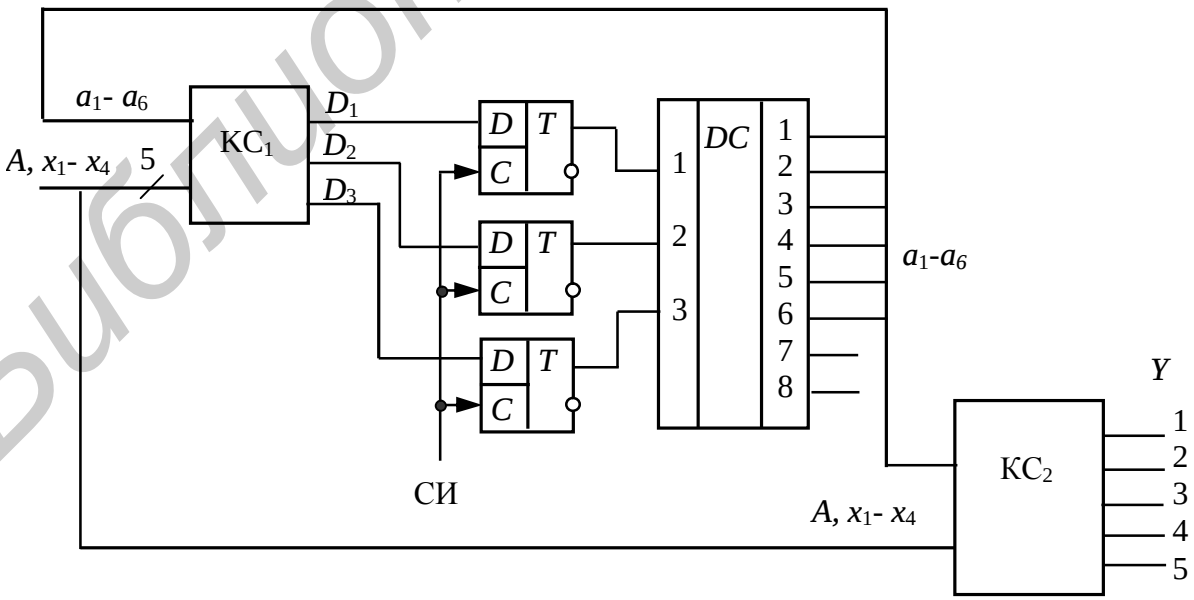


Рис. 2.3

2.1.2. Синтез МПА Мура по ГСА

Закон функционирования автомата Мура так же, как и автомата Мили, задается функцией переходов и функцией выходов. Функция переходов δ ставит в соответствие паре параметров «состояние–входной сигнал» a_m, x_i состояние автомата a_s, в которое он переходит из состояния a_m под действием сигнала x_i, то есть $a_s = \delta(a_m, x_i)$.

Однако функция выходов λ, в отличие от автомата Мили, ставит в соответствие каждому состоянию автомата выходной сигнал y_j, который определяется равенством $y_j = \lambda(a_m)$, то есть имеет место зависимость только от состояния автомата.

Пусть задан МПА с помощью ГСА, приведенной на рис. 2.4 (без учета сигнала запуска A). При синтезе МПА Мура, аналогично, как и для автомата Мили, ГСА должна быть размечена по определенным правилам. Общее правило разметки состоит из двух пунктов:

а) символом a_1 отмечаются начальная и конечная вершины;

б) символами a_2, a_3 … отмечаются все операторные вершины.

Число элементов памяти в МПА определяется величиной $R = \text{int}\log_2 M$, где M – число состояний автомата (или операторных вершин ГСА).

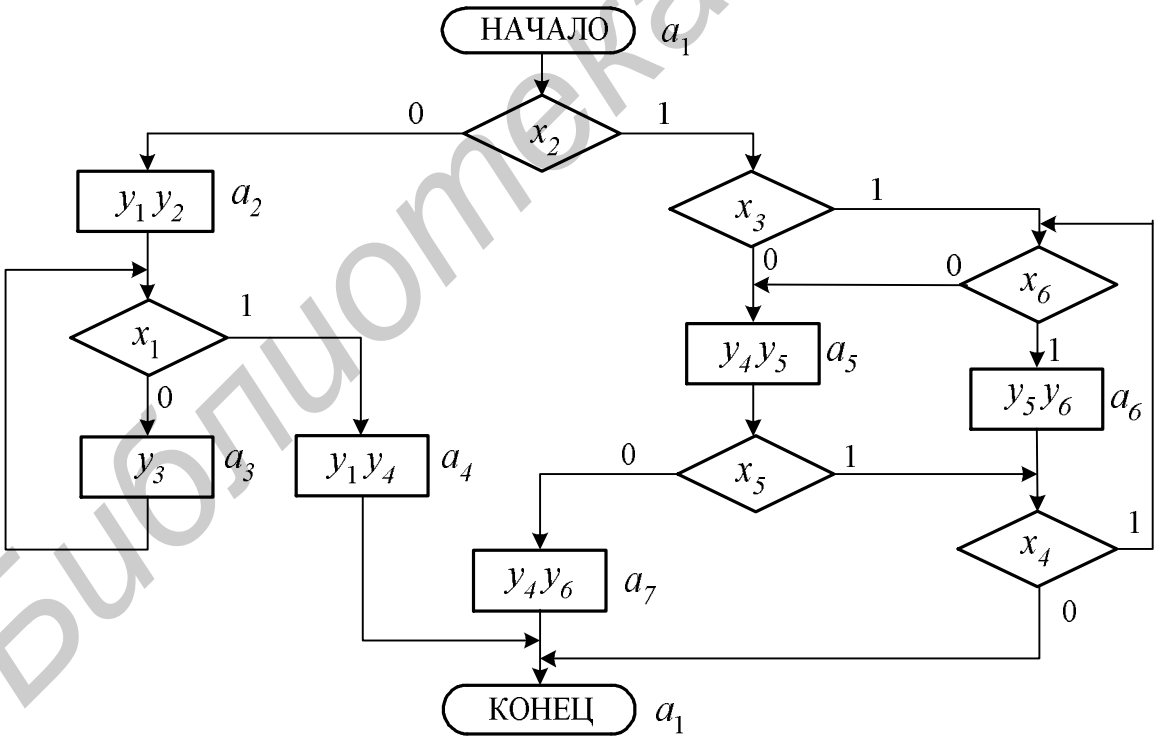


Рис. 2.4

По заданной граф-схеме алгоритма строим структурную таблицу МПА Мура (табл. 2.2). Для кодирования состояний автомата используется код длиной 3 (столбцы таблицы $K(a_m), K(a_s)$), функции возбуждения $F(a_m, a_s)$ про-

ставляются аналогично предыдущему случаю. Однако при записи булевых соотношений для выходных переменных необходимо помнить, что данные параметры не зависят от входных сигналов. Таким образом, автомат Мура – это микропрограммное устройство со статическим состоянием выходов. Данное обстоятельство следует учитывать при синтезе АЛУ, где статическое управление характерно для схем коммутации (управление мультиплексорами), схем задания режимов работы ОА, механизмов арбитража шин вычислительного устройства между источниками и приемниками информации и т.д.

Таблица 2.2

a_m	$K(a_m)$	a_s	$K(a_s)$	$X(a_m a_s)$	$Y(a_m)$	$F(a_m a_s)$
a_1	000	a_2	001	$\overline{x_2}$	-	D_3
		a_5	100	$x_2\overline{x_3}$		D_1
		a_5	100	$x_2x_3\overline{x_6}$		D_1
		a_6	101	$x_2x_3x_6$		D_1D_3
a_2	001	a_3	010	$\overline{x_1}$	y_1y_2	D_2
		a_4	011	x_1		D_2D_3
a_3	010	a_3	010	$\overline{x_1}$	y_3	D_2
		a_4	011	x_1		D_2D_3
a_4	011	a_1	000	1	y_1y_4	-
a_5	100	a_7	110	$\overline{x_5}$	y_4y_5	D_1D_2
		a_1	000	$x_5\overline{x_4}$		-
		a_6	101	$x_4x_5x_6$		D_1D_3
		a_5	100	$x_4x_5\overline{x_6}$		D_1
a_6	101	a_1	000	$\overline{x_4}$	y_5y_6	-
		a_6	101	x_4x_6		D_1D_3
		a_5	100	$x_4\overline{x_6}$		D_1
a_7	110	a_1	000	1	y_4y_6	-

Выпишем равенства для функций возбуждения *D*-триггеров:

$$\begin{cases} D_1 = a_1x_2 \vee a_5(\overline{x}_5 \vee x_4) \vee a_6x_4, \\ D_2 = a_2 \vee a_3 \vee a_5\overline{x}_5, \\ D_3 = a_1x_2x_3x_6 \vee a_2x_1 \vee a_3x_1 \vee a_5x_4x_5x_6 \vee a_6x_4x_6. \end{cases}$$

Выходные переменные сформируем по закону:

$$\begin{cases} y_1 = a_2 \vee a_4, \\ y_2 = a_2, \\ y_3 = a_3, \\ y_4 = a_4 \vee a_5 \vee a_7. \end{cases} \qquad \begin{cases} y_5 = a_5 \vee a_6, \\ y_6 = a_6 \vee a_7. \end{cases}$$

Теперь по данным соотношениям можно построить функциональную схему МПА Мура в виде, представленном на рис. 2.5.

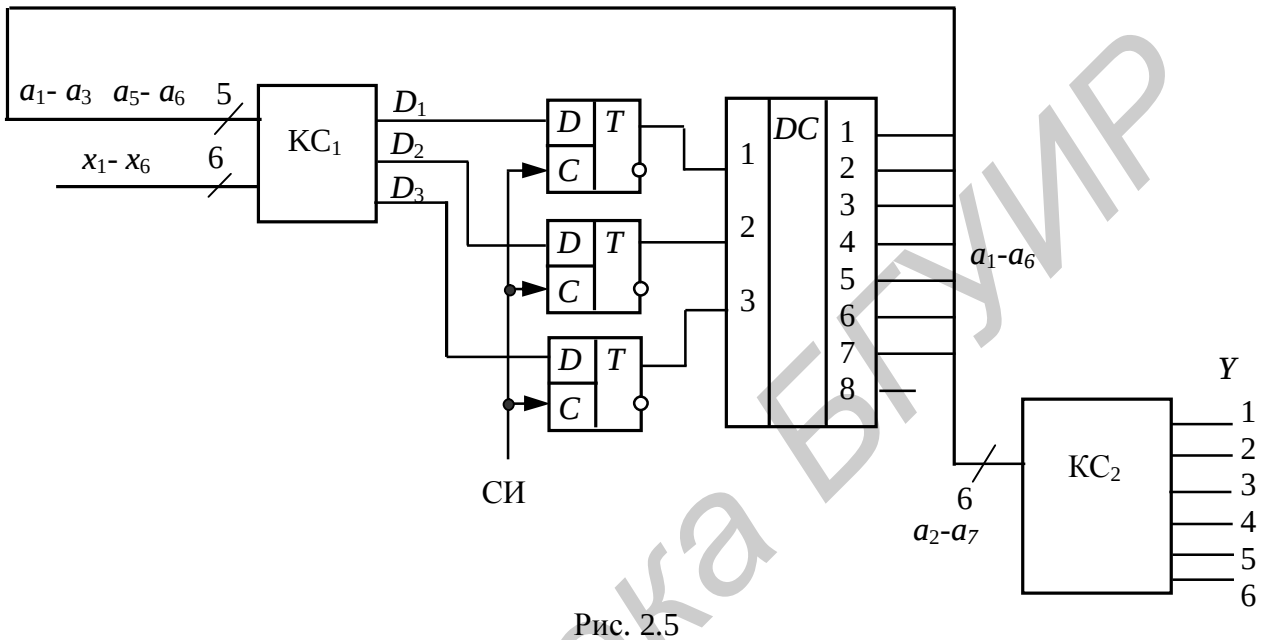


Рис. 2.5

В приведенной структуре используется специальный дешифратор для формирования состояний автомата. В его отсутствии состояния могут быть сформированы с помощью логических элементов И, ИЛИ, НЕ. Однако при этом необходимо решать задачу минимизации длины кода состояния. Рассмотрим данную ситуацию на примере.

Пусть выход дешифратора a_i определяется установкой триггеров в некоторый текущий момент времени в состояние z_1, z_2, z_3, где z_i есть переменная z_i или $\overline{z}_i$. При этом минимизация осуществляется с использованием карты Карно–Вейча (рис. 2.6):

z_1	z_2z_3 = 00	01	11	10
1	a_5	a_6	*	a_7
0	a_1	a_2	a_4	a_3

a_1=000 a_5=100
a_2=001 a_6=1-1
a_3=010 a_7=11-
a_4= -11

Рис. 2.6

Состояние $z_1z_2z_3 = 111$ в автомате отсутствует, поэтому данный код используется для минимизации длины конъюнкции состояний a_4, a_6, a_7, а в таблице

отмечается «звездочкой». Уравнения для выходных переменных в данном случае будут иметь следующий вид:

$$y_1 = \left(\overline{z_1} \vee z_2\right) z_3,$$
$$y_2 = \overline{z_1}\,\overline{z_2}\,z_3,$$
$$y_3 = \overline{z_1} z_2\,\overline{z_3},$$
$$y_4 = z_1\overline{z_3} \vee z_2\left(z_3 \vee z_1\right),$$
$$y_5 = z_1\overline{z_2},$$
$$y_6 = z_1(z_3 \vee z_2).$$

Цена данной схемы по Квайну (с учетом инверсий) составляет 27 входов.

2.2. Управляющие автоматы с программируемой логикой

Данный тип автоматов реализует принцип управления по хранимой программе. В соответствии с этим функция проектируемого управляющего устройства определяется следующими параметрами:

1) множеством входных сигналов $x_1 \ldots x_m$, состоящим из кода операции, а также признаков результата, сформированных на выходе АЛУ (ОА);

2) множеством выходных управляющих сигналов $y_1 \ldots y_n$, инициирующих микрооперации в операционном автомате;

3) микропрограммой, задающей порядок следования выходных сигналов в зависимости от алгоритма вычислений.

В результате функционирование управляющего автомата сводится к генерированию последовательности сигналов микроопераций в соответствии с заданным порядком компьютерных расчетов.

Схема проектируемого УА может быть построена на основе принципа программного управления, использующего операционно-адресную структуру управляющих слов. При этом под управляющими словами будем понимать наборы сигналов микроопераций, определяющие порядок функционирования устройства в течение одного такта. Такие наборы (или списки воздействий) в задачах проектирования устройств управления получили название микрокоманд. Соответственно множество микрокоманд, реализующих заданный алгоритм преобразования данных, называют микропрограммой.

В структуре автомата с программируемой логикой отдельные элементы – микрокоманды, хранимые в ПЗУ, выделяются посредством адреса. В частном случае, адрес может быть выбран, например, равным номеру текущей микрокоманды.

Управляющее слово в общем случае должно содержать информацию о микрооперациях, выполняемых в текущем такте, а также адресную информацию о следующей микрокоманде. Таким образом, структура управляющего слова, достаточная для представления микрокоманды, определяется следующими действиями.

1. Пусть задан граф алгоритма вычислений, содержащий множество сигналов $Y=\{y_1, \ldots, y_n\}$. Тогда номера микроопераций можно закодировать двоичными кодами разрядностью $r=\text{int}\log_2 n$ и разместить в операционной части управляющего слова. Для расшифровки управляющей информации в схеме УА будем использовать дешифратор *DC Y*.

2. Для формирования адреса следующей микрокоманды воспользуемся методом принудительной адресации. Сущность метода состоит в том, что в специальном поле текущей микрокоманды указывается адрес следующей микрокоманды. Этот адрес может извлекаться из разрядов управляющего слова безусловно, то есть независимо от значений сигналов обратной связи, или выбираться по условию, сформированному по результату вычислений.

3. Пусть УА анализирует только одно условие x_i из множества $X=\{x_1 \ldots x_m\}$. Тогда для выбора любого из m условий в схеме управляющего слова необходимо предусмотреть l-разрядное поле, управляющее дешифратором условия *DC X*. Таким образом, структура УС может быть представлена на рис. 2.7.

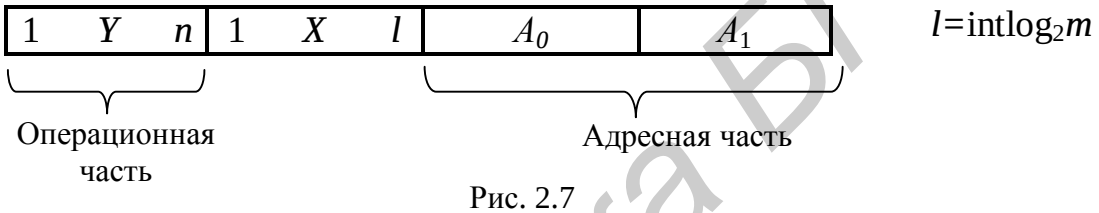


Рис. 2.7

В данной структуре поле X определяет номер условия, по которому выполняется переход, причем будем считать:

- если $x_i=0$, то адрес перехода извлекается из разрядов поля A_0;
- если $x_i=1$, то адрес исполнительный расположен в поле A_1.

Для хранения микрокоманд, как было указано выше, используется специальное ПЗУ Мк (микрокоманд), обращение к которому осуществляется путем подачи адреса на адресные входы. При этом с выходов памяти читается требуемое слово, которое далее помещается в регистр микрокоманды. Простейшая реализация УА с принудительной адресацией может быть представлена схемой, приведенной на рис. 2.8.

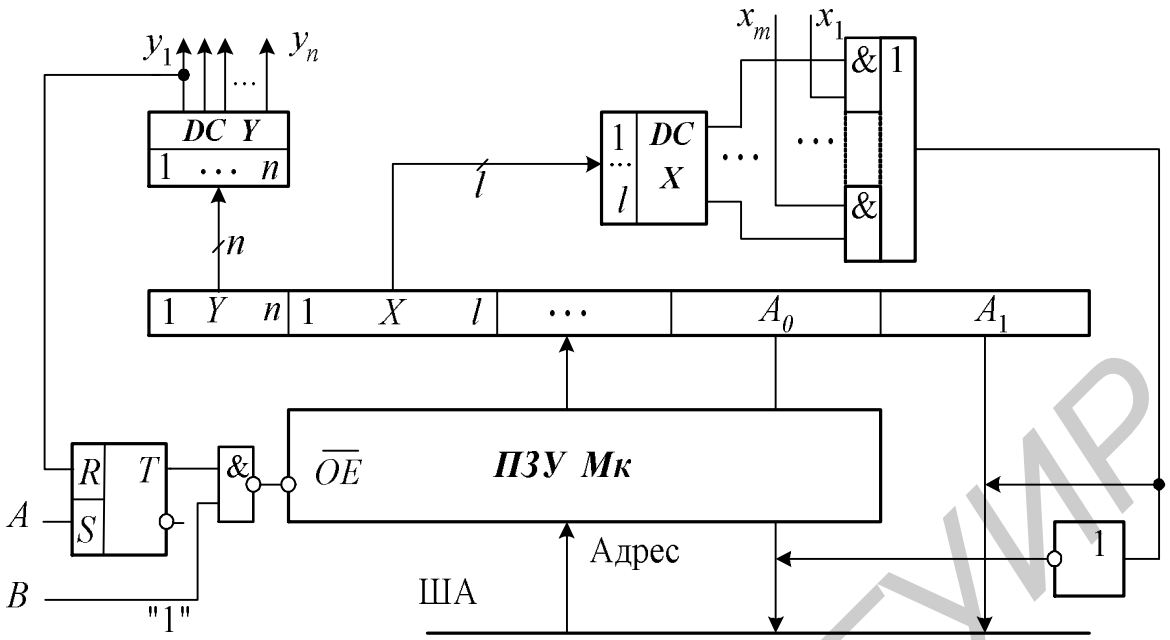


Рис. 2.8

Устройство работает следующим образом.

Перед началом работы триггер управления установлен в нулевое состояние, поэтому на входе $\overline{OE}$ ПЗУ формируется логическая единица:

$$\overline{(B=1)\wedge(T=0)}=\overline{OE}=1,$$

где T – состояние триггера управления.

При этом выходы ПЗУ устанавливаются в R_{off} (третье состояние) и в регистр микрокоманды читается код 00…0. Этот код воспринимается схемой как команда *NOP*.

Запуск автомата осуществляется сигналом A, который переключает триггер в единичное состояние. Это позволяет прочитать микрокоманду из ячейки 00…0 и выполнить анализ сигналов $\{X\}$. После отработки микропрограммы распознавания КОП осуществляется переход к микропрограмме вычислений, по окончании которой формируется сигнал y_1, сбрасывающий триггер управления в ноль. Автомат выключается.

Вход B автомата может быть использован для отключения устройства (установкой $B = 0$) при управлении схемой от некоторого центрального устройства управления.

2.3. Принудительная адресация в МПУУ Уилкса

Автомат Уилкса относят к классу автоматов с хранимой логикой. Структурная схема такого устройства имеет вид, приведенный на рис. 2.9.

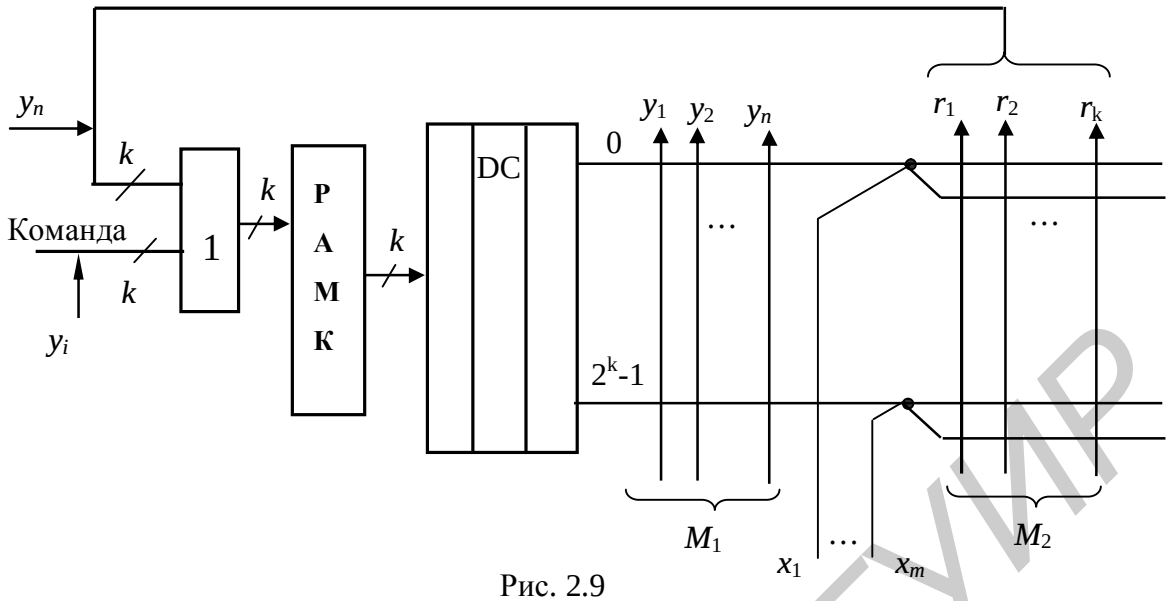


Рис. 2.9

Здесь M_1 – матрица (ПЗУ) *у*правляющих сигналов; M_2 – матрица (ПЗУ) адресов переходов.

Линия выборки дешифратора адресует строку матрицы M_1 и пару строк матрицы M_2. Таким образом, схема M_2 содержит в 2 раза больше строк, чем ПЗУ матрицы M_1.

Блок выборки следующего адреса в схеме автомата Уилкса может быть представлен в виде, приведенном на рис. 2.10.

Приведенная реализация поясняет принцип ветвления в микропрограмме. В частности, элементы 2И осуществляют анализ признака x_i и выбор соответствующего адреса из ячейки *A* или *B* ПЗУ. При этом, если $x_i = 0$, адрес извлекается из ячейки *A*, при $x_i = 1$ – из ячейки *B*. Для организации условных переходов содержимое ячейки *A* и ячейки *B* должно быть различным. Если же в микропрограмме реализуется безусловный переход, то содержимое строки *A* и строки *B* ПЗУ должно быть одинаковым.

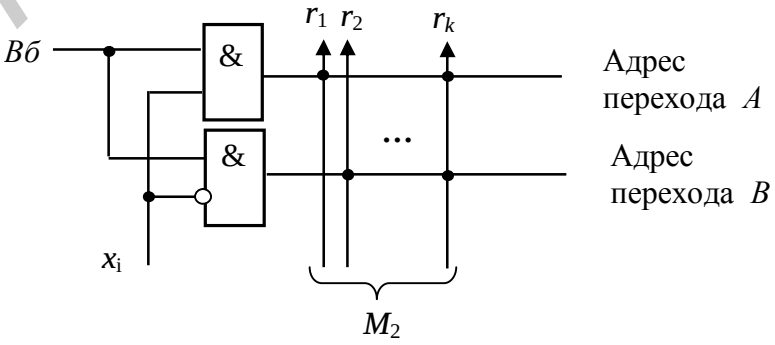


Рис. 2.10

Функциональная схема автомата Уилкса может быть представлена в виде, показанном на рис. 2.11.

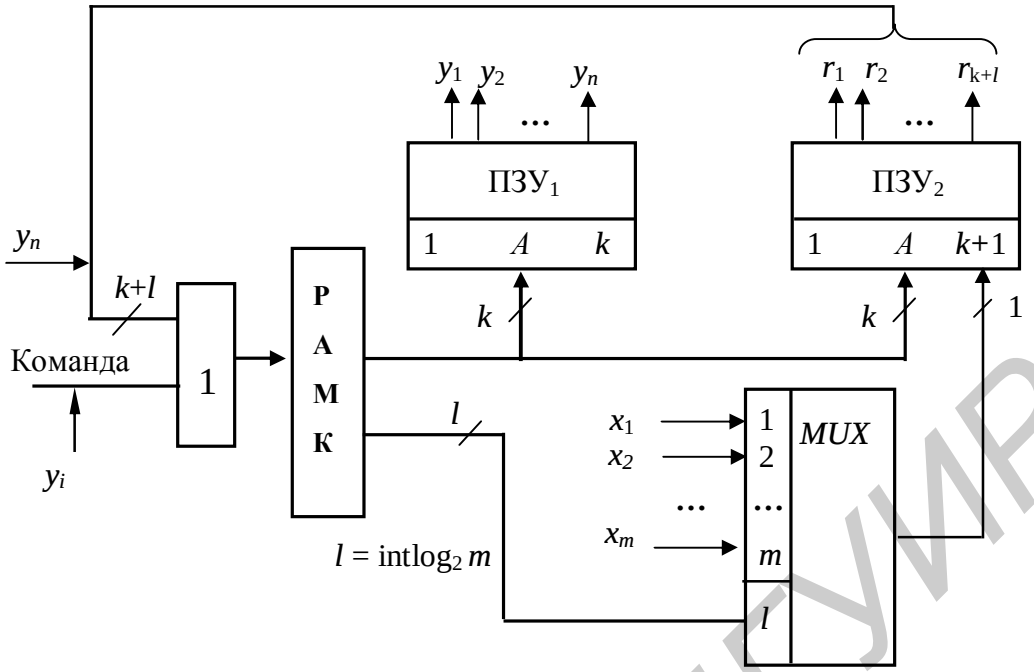


Рис. 2.11

В данной схеме ПЗУ$_1$ формирует сигналы управления, ПЗУ$_2$ – адрес следующей микрокоманды, номера условий перехода определяет состояние входа *l* *MUX*. При запуске автомата команда с помощью сигнала y_i заносится в РАМК и адресует первую микрокоманду выполняемой микропрограммы. УУ формирует сигналы управления вычислительным процессом $y_1, y_2, ..., y_n$, один из которых – y_n – используется для разрешения записи в регистр адреса микрокоманды (РАМК) следующего $k+l$-разрядного адреса с выхода ПЗУ$_2$. В процессе функционирования младший разряд адреса k+1 ПЗУ$_2$ используется для выбора одной из двух ячеек памяти, адресуемых k разрядами РАМК. Иными словами этот разряд конкретизирует адрес перехода при выборке следующей микрокоманды в процессе функционирования устройства.

2.4. Естественная адресация. Функциональная схема УА с естественной адресацией

При данном способе адресации адрес следующей микрокоманды определяется равным адресу предыдущей микрокоманды плюс единица:

$$A_{i+1} = A_i + 1.$$

При формирования исполнительных адресов указанный способ легко реализуется с помощью простого устройства, а именно двоичного счетчика микрокоманд (СМК).

Принцип ветвления в микропрограмме при использовании схемы, представленной на рис. 2.12, может быть реализован двумя способами: с помощью поля адреса безусловного перехода *BR*, выделяемого в адресной части микро-

команды, и с помощью блока СМК, реализующего линейный участок микропрограммы.

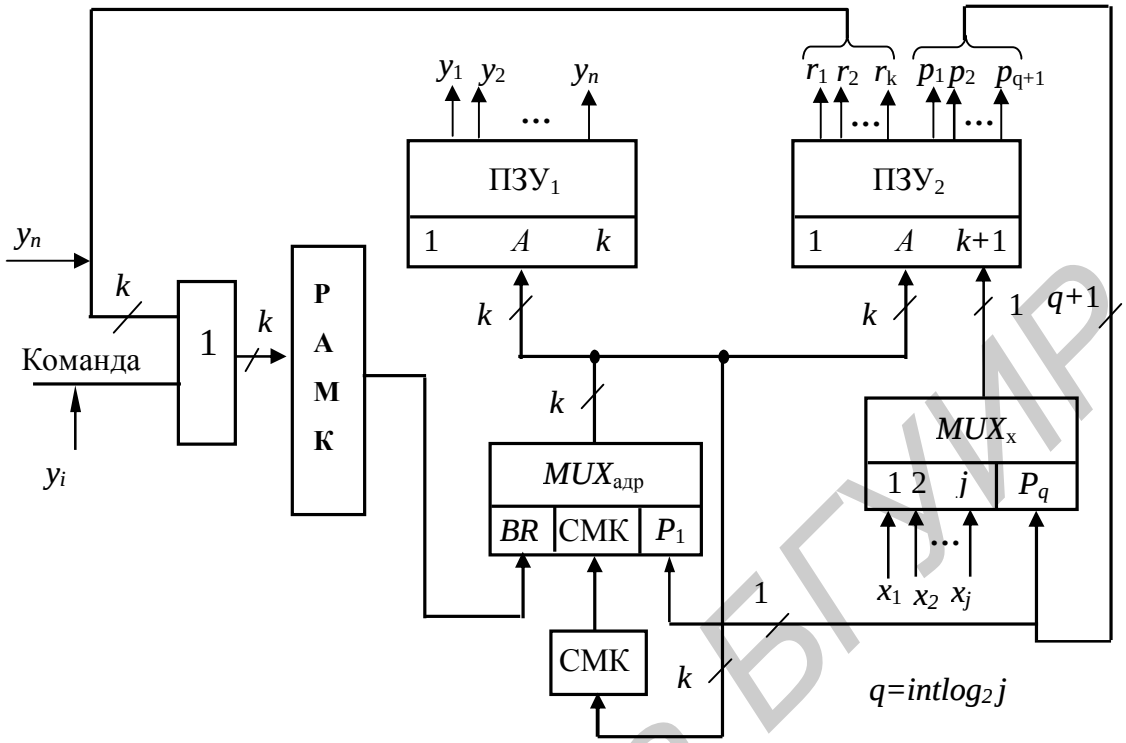


Рис. 2.12

Функционирование схемы начинается с занесения команды в РАМК. Эта микрооперация выполняется в конце каждой микропрограммы путем выдачи сигнала y_i.

Состояние входа P_1 (см. $MUX_{адр}$), определяемое полем P_1–P_{q+1}, позволяет коммутировать на адресные входы ПЗУ$_1$ и ПЗУ$_2$ информацию от двух источников. При естественном следовании адресов на входе P_1 устанавливается логическая «1», что позволяет осуществить чтение адреса из счетчика микрокоманд и циклическое наращивание СМК на 1. При организации принудительного перехода P_1 устанавливается в «0» и на выход $MUX_{адр}$ передается поле перехода *BR*.

Организация условных переходов осуществляется путем задания в текущей микрокоманде кода условия перехода (или номера условия x_i). Разряды P_1–P_q передаются на управляющие входы MUX_x, и выбранное условие коммутируется на адресный вход A_0 ПЗУ$_2$. Если условие $x_i = 0$, то выполняется чтение четной ячейки ПЗУ$_2$. При $x_i = 1$ адресуется нечетное управляющее слово. Различное информационное содержание этих двух слов позволяет изменить источник адреса, а следовательно, выполнить ветвление в текущей микропрограмме.

2.5. Микроминиатюрная реализация управляющих устройств. СУАМ К1804ВУ1

Совершенствование принципов управления и необходимость создания миниатюрных схем УУ привело к созданию специальных БИС, реализующих

сложные функции при выборке адресов. Примером такой БИС является секция управления адресом микрокоманды К1804ВУ1 (СУАМ).

Эта БИС предназначена для управления адресами в схемах блоков микропрограммного управления (БМУ). Основная функция ее заключается в формировании адреса микрокоманды под воздействием внешних управляющих сигналов (рис. 2.13).

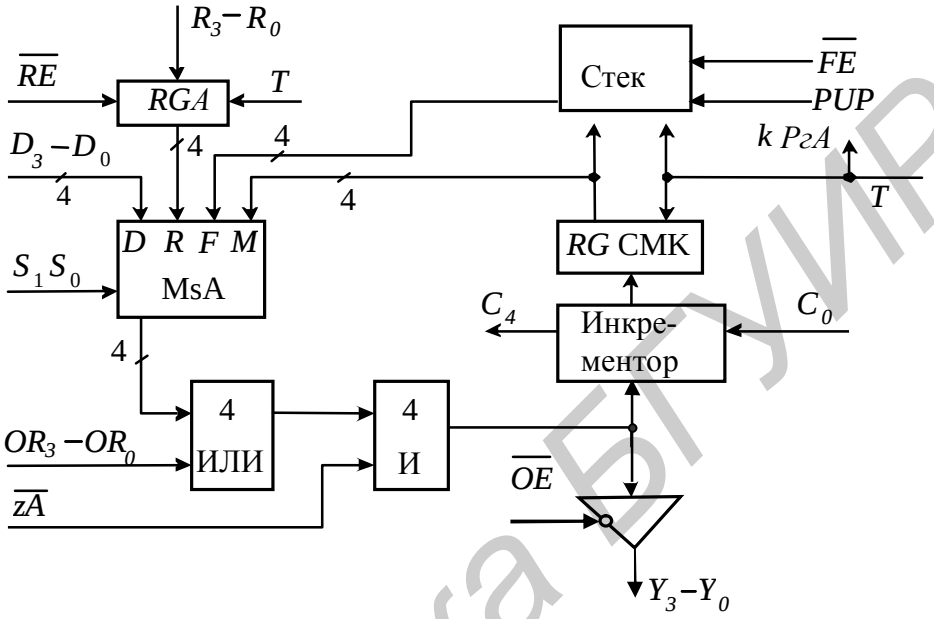


Рис. 2.13

К особенностям СУАМ относят:

- возможность наращивания разрядности БМУ до любого значения, кратного 4;
- наличие внутреннего регистра адреса, СМК и стека глубиной 4;
- наличие трех стабильных *Y*-выходов.

Рассмотрим основное назначение структурных составляющих модуля СУАМ.

Мультиплексор адреса позволяет выбрать в качестве следующего адреса микрокоманды один из четырех источников: шину данных D_3–D_0, регистр адреса *RGA*, стек, регистр-счетчик микрокоманд РгСМК. Управление мультиплексором осуществляется сигналами S_1S_0, коммутирующими требуемый источник информации.

Выходы мультиплексора соединены с четырьмя двухвходовыми схемами ИЛИ, которые позволяют маскировать выбранные разряды адреса с помощью сигналов маски OR_3–OR_0.

Далее по схеме включены 4 двухвходовых элемента И, управляемые сигналом $\overline{zA}$. Если $\overline{zA}$ = 0, то на выходе СУАМ устанавливается нулевой адрес. При $\overline{zA}$=1 значение адреса определяется состоянием линий управления мультиплексором S_1S_0.

Табл. 2.3 определяет зависимость $Y=f(S_1,S_0)$ и задает источник адреса на выходе мультиплексора адреса в зависимости от конкретных значений управляющих сигналов.

Совместное действие маски OR_3–OR_0 и управляющих сигналов $\overline{zA}$ и $\overline{OE}$ представлено в табл. 2.4.

Таблица 2.3

S_0	S_1	Y
0	0	*СМК*
0	1	*RGA*
1	0	*Стек*
1	1	*D*

Таблица 2.4

OR_i	$\overline{zA}$	$\overline{OE}$	Y
x	*x*	1	*Roff*
x	0	0	0
1	1	0	1
0	1	0	*

* источник выбирается сигналами S_1, S_0 *MUX A*

Регистр адреса *RGA* рассматриваемой БИС предназначен для хранения адреса перехода, принятого от внешнего источника по входам OR_3–OR_0. Запись в *RGA* возможна только при $\overline{RE} = 0$ и наличии положительного фронта на входе синхронизации.

Счетчик микрокоманд состоит из *RG* СМК и инкрементора. Увеличение содержимого счетчика возможно только при $C_0 = 1$. Если же $C_0 = 0$, то адрес не изменяется. Если содержимое *RG* СМК = 11…1 и вход $C_0 = 1$, то при наличии фронта тактового сигнала *T* на выходе формируется нулевой код и C_4 устанавливается в «1».

Аппаратный стек имеет глубину 4. Управление им осуществляется сигналами $\overline{FE}$ и *PUP* в соответствии с табл. 2.5.

В качестве указателя стека (УС) в схеме используется двухразрядный реверсивный счетчик. При этом, если $\overline{FE} = 0$, $PUP = 1$, то происходит изменение указателя по правилу

$$2 \rightarrow 3 \rightarrow 0 \rightarrow 1 \rightarrow 2.$$

Если $\overline{FE} = 0$, $PUP = 0$, то выполняется обратный счет:

$$2 \rightarrow 1 \rightarrow 0 \rightarrow 3 \rightarrow 2.$$

Таблица 2.5

$\overline{FE}$	*PUP*	Операция
1	*X*	Стек отключен
0	1	*PUSH*: *СМК* → стек
0	0	*POP*: циклический сдвиг

Стек имеет динамический вход синхронизации, поэтому запись выполняется при изменении сигнала синхроимпульса из 0 в 1.

В общем случае данная схема обеспечивает 3 режима работы:

1) увеличение содержимого указателя стека и запись: $\overline{FE} = 0$, $PUP = 1$;

2) считывание и уменьшение указателя стека: $\overline{FE} = 0$, $PUP = 0$;

3) считывание без изменения указателя стека: $\overline{FE} = 1$, $PUP = x$.

2.6. Блок микропрограммного управления. Аппарат условных и безусловных переходов

Рассмотрим реализацию блока микропрограммного управления (БМУ) с использованием секции К1804ВУ1 (рис. 2.14). Данная реализация БМУ служит для формирования и выдачи адресов на входы микропрограммной памяти в компьютере, имеющем разрядно-модульную организацию связей.

Схема БМУ включает в свой состав следующие блоки: MUX_1 – мультиплексор условия, MUX_2 – мультиплексор режима работы, ПЗУ емкостью 32 × 8 слов, СУАМ К1804ВУ1 и клавиатуру адреса.

ПЗУ имеет 5 адресных входов, причем 4 старших разряда определяются содержимым поля P_0–P_3 регистра микрокоманды. Младший разряд адреса A_0 – это один из признаков результата, коммутируемый из регистра состояния с помощью мультиплексора MUX_1. Выбор признака *z*, F_3, *OVR* или C_4 производится с помощью сигналов P_0 P_1, т.е. младших разрядов поля P_0–P_3 регистра микрокоманды. Любой из этих флагов может быть выбран для формирования условных переходов при ветвлении в микропрограмме.

Реализация ветвления в данной схеме осуществляется следующим образом: если признак, считанный из регистра состояния *RGF*, равен 0, то из ПЗУ читается четное управляющее слово. При единичном признаке адресуется нечетная строка, содержащая в принципе другую информацию. Различное информационное содержание этих двух строк обеспечивается переводом СУАМ из одного режима формирования адреса в другой.

При организации безусловных переходов адреса перехода, например, указывается в специальном поле микрокоманды BR_3–BR_0. В такте отработки микрокоманды содержимое этого поля, минуя *RGMk*, передается непосредственно на схему К1804ВУ1 во внутренний регистр адреса. В данном случае *RGA* СУАМ используется как часть регистра микрокоманды. Это позволяет передавать адрес безусловного перехода на выход СУАМ в такте выполнения текущей микрокоманды, то есть без опоздания на 1 такт.

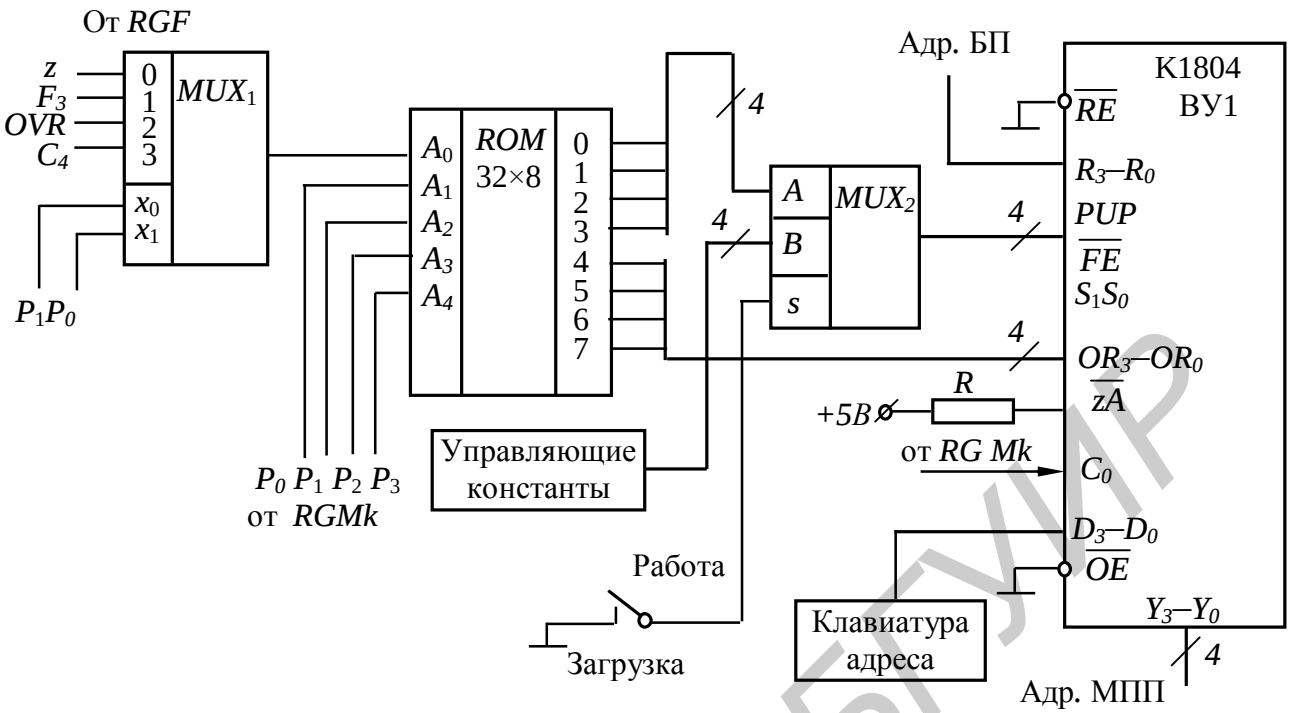


Рис. 2.14

Одинаковое информационное содержание двух строк ПЗУ обеспечивает перевод СУАМ в требуемый режим формирования адреса независимо от сформированного АЛУ условия.

Схема БМУ функционирует в двух режимах:

1. В режиме «Загрузка» на входе *s* MUX_2 устанавливается уровень логической единицы, при этом управляющие константы передаются на входы СУАМ и позволяют подключить клавиатуру адреса к выходам *Y* схемы ВУ1; таким образом, в режиме «Загрузка» клавиатура непосредственно соединяется с адресными входами микропрограммной памяти. Это позволяет выполнить запуск заданной микропрограммы с любой точки и реализовать процедуры проверки программного обеспечения и коррекции ошибок.

2. В режиме «Работа» на входе *s* присутствует уровень логического 0. При этом на выход MUX_2 передаются сигналы с выходов ПЗУ и определяют режим работы БМУ в течение одного такта машинного времени. То есть в режиме «Работа» коды ПЗУ определяют характер перехода и источник следующего адреса в микропрограмме.

Пусть, например, в ПЗУ по адресам A_4–A_0 записана информация, приведенная в табл. 2.6.

Таблица 2.6

A_4	A_3	A_2	A_1	A_0		Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	*RGA*	1	0	1	*x*
0	0	0	0	1	СМК	0	0	1	*x*

z			
$P_3 \ldots P_0$	F		$\overline{FE}$ PUP

Адрес 00000 определяет переход в микропрограмме на адрес из *RGA*. При появлении признака $z = 1$ осуществляется переход к следующей микрокоманде в соответствии с кодом СМК.

2.7. Устройства управления на ПЛИС

Программируемые логические устройства, называемые иногда программируемыми интегральными схемами (ПЛИС), представляют собой элементную базу, обладающую гибкостью заказных БИС и доступностью традиционной «жесткой» логики. Главным отличительным свойством этих устройств является возможность их настройки на выполнение заданных функций самим пользователем. Процесс проектирования цифровых устройств на основе ПЛИС заключается в описании их функционирования на входном языке САПР, выполнении автоматизированного синтеза, проведении моделирования и настройки ПЛИС с помощью программатора.

Программируемые пользователем БИС делятся на два класса устройств в соответствии с двумя основными подходами к проектированию схем: микропрограммным и аппаратным. Первый подход позволяет построить устройство на базе некоторых универсальных элементов с зашиваемой в ПЗУ микропрограммой, второй – предполагает синтез управляющего автомата с использованием базовых теорий и распределения его функций на программируемых БИС с учетом заданного комплекта ИМС.

Основное достоинство использования ПЛИС в системах управления состоит в достижении высокого функционального быстродействия в задачах координации вычислительного процесса для любой сколь угодно сложной конфигурации компьютера.

3. ОРГАНИЗАЦИЯ ВВОДА–ВЫВОДА В КОМПЬЮТЕРНЫХ СИСТЕМАХ

3.1. Общие сведения

В соответствии с общепринятым соглашением направление потоков вводимой и выводимой информации в вычислительных устройствах рассматривают относительно процессора. Следовательно, портом ввода компьютера будем называть любой источник информации, подключенный к его шине данных и позволяющий вводить параметры решаемых задач из внешней памяти или от других периферийных устройств компьютера. Аналогично портом вывода будем называть любой приемник информации, подключенный к ШД и предназначенный для вывода на внешние носители результатов решенных задач.

В большинстве случаев для адресации портов используется шина адреса компьютера или ее часть. Однако при необходимости адреса портов ввода могут отличаться от адресов портов вывода (и адресов оперативной памяти) не значениями на адресных линиях, а сигналами идентификации *IOR* (чтение УВВ) и *IOW* (запись в УВВ) на линиях шины управления.

3.2. Подключение портов к общей шине

Как правило, для ввода–вывода данных в компьютер используются регистры с тремя состояниями выходной шины *DO*. На рис. 3.1 приведена схема соединения таких регистров и элементов системы, применяемых для ввода информации. При вводе по данной схеме периферийное устройство (ПУ) устанавливает вводимое слово на входы *DI* порта и посылает по линии *WR* активный уровень записи. Процессор принимает запрос на обслуживание в виде сигнала

INT, после чего на ША машины выдается адрес порта, сформировавшего запрос. Дешифратор внешних устройств расшифровывает адрес и возбуждает на своем выходе соответствующую линию выборки. Это определяет на входе $\overline{OE}$ адресуемого регистра активный нулевой уровень, что определяет подключение выходной шины *DO* используемого порта к ШД компьютера. Информационные выходы остальных портов сохраняют состояние высокого сопротивления.

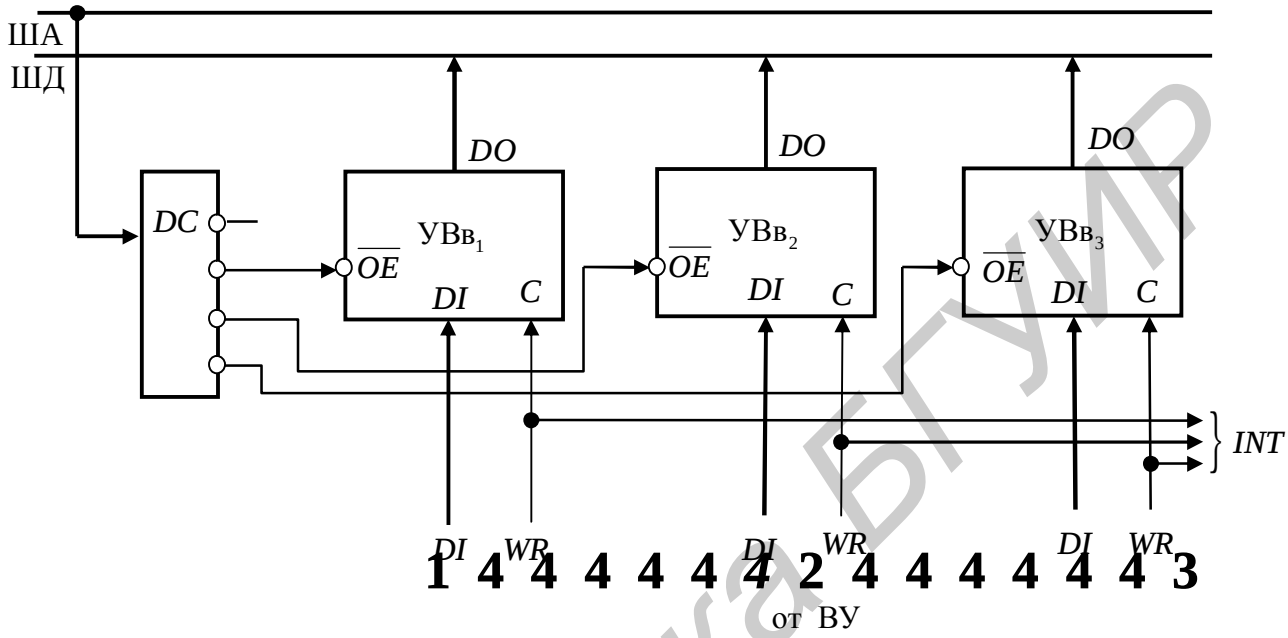


Рис. 3.1

В результате выполнения команды *IN* («Ввод») вводимая информация передается на шину данных и записывается в один из регистров процессора. На этом цикл ввода с периферийного устройства завершается.

При выводе данных на внешние носители может быть использована схема, приведенная на рис. 3.2.

В данной схеме инициатором обмена является компьютер. В режиме вывода процессор системы выдает на ША адрес порта, на ШД – выводимую информацию. Дешифратор порта возбуждает на своем выходе требуемую линию выборки, и входная шина *DI* адресуемого устройства оказывается подключенной к ШД компьютера. Входные информационные шины остальных портов отключены от ШД, так как соответствующие выходы дешифратора находятся в пассивном единичном состоянии.

Для непосредственной записи данных в порт на линии *IOW* ШУ устанавливается активный уровень управляющего сигнала. Это соответствует появлению синхросигнала на входе *C* регистра и предполагает запись данных в адресуемый порт. Внешнее устройство оповещается о запросе со стороны компьютера сигналом готовности $\overline{RDY_1}$. В качестве такого сигнала может быть использован специальный интерфейсный вывод процессора или, например, выход дешифратора порта, как показано на рис. 3.2.

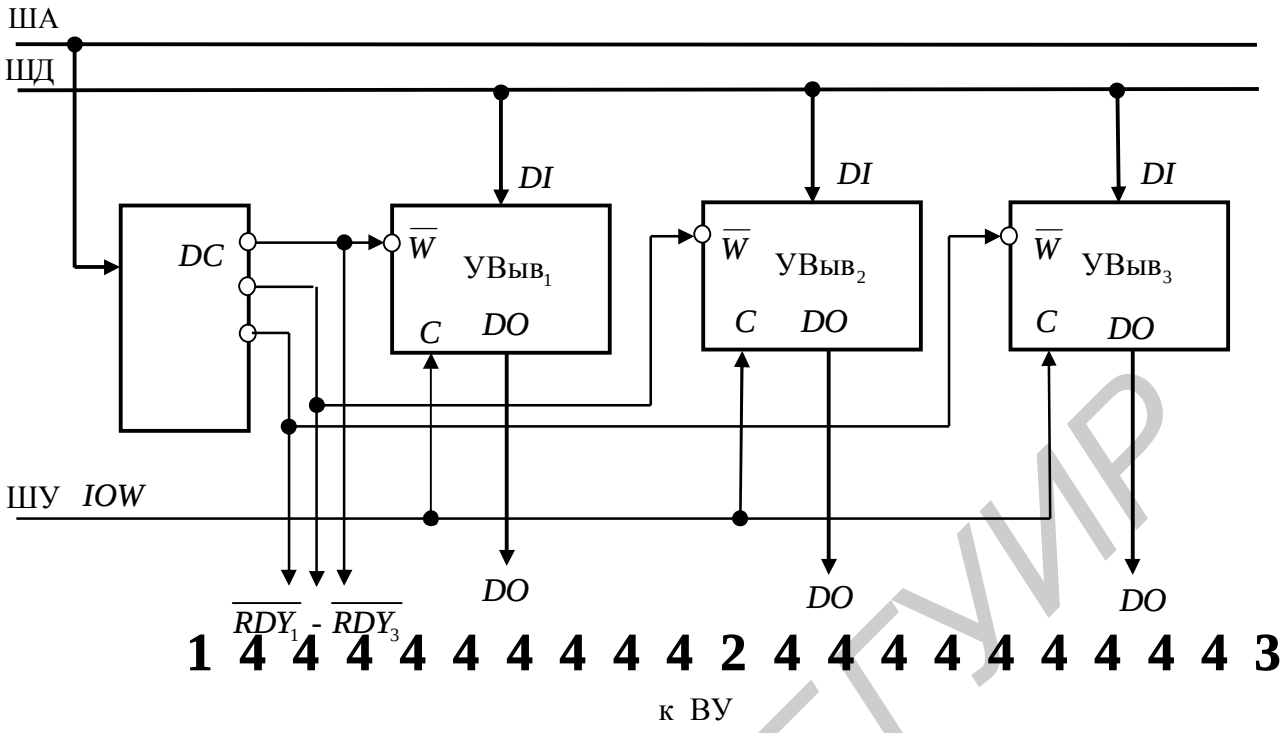


Рис. 3.2

3.3. Схема и временные диаграммы сигналов при вводе и выводе информации по разделенной шине

При вводе–выводе данных через порт с фиксированным номером применяется принцип идентификации регистров сигналами $\overline{IOW}$ и $\overline{IOR}$ на шине управления. Схема устройства в данном случае представлена на рис. 3.3.

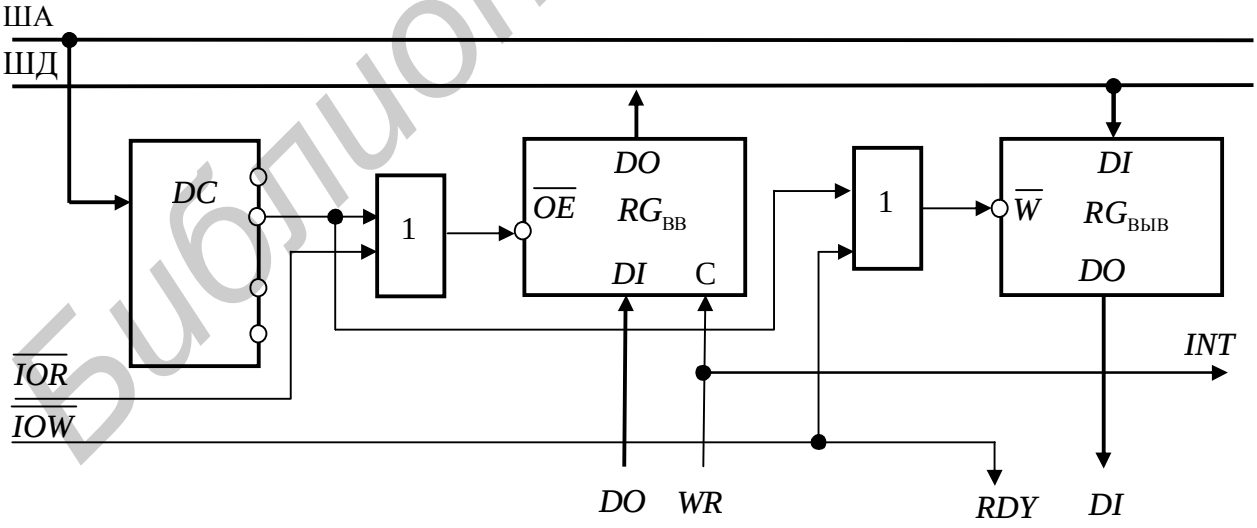


Рис. 3.3

Приведенные на схеме регистры порта ввода–вывода имеют одинаковый адрес, однако различаются сигналами $\overline{IOR}$ и $\overline{IOW}$ на линиях ШУ. В режиме приема информации функционирует регистр ввода. При этом внешнее устрой-

ство записывает в регистр вводимое слово, одновременно формируя запрос на обслуживание на входе INT процессора. Реагируя на запрос, процессор выдает на ША адрес порта, а на ШУ активный уровень сигнала $\overline{IOR}$. На входе $\overline{OE}$ регистра ввода устанавливается нулевой уровень. После этого вводимая информация через трехстабильные выходы DO передается на ШД компьютера.

В режиме вывода данных используется регистр вывода. В этом случае на ШД устанавливается выводимое слово, а выдача адреса на ША сопровождается сигналом $\overline{IOW}$ на шине управления. На входе $\overline{W}$ $RG_{ВЫВ}$ устанавливается нулевой уровень. Таким образом, сигнал $\overline{IOW}$ разрешает запись информации в порт и одновременно сообщает внешнему устройству о готовности компьютера к обмену. Выходная шина порта находится в разрешенном режиме, в связи с этим данные беспрепятственно передаются к периферийному устройству.

При вводе и выводе информации через порт имеют место временные диаграммы сигналов, приведенные на рис. 3.4 и 3.5.

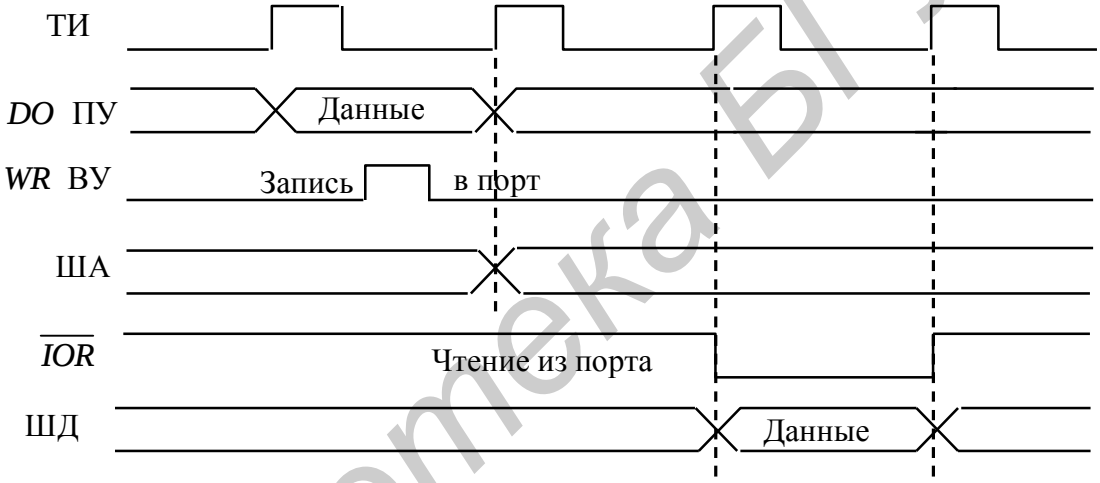


Рис. 3.4

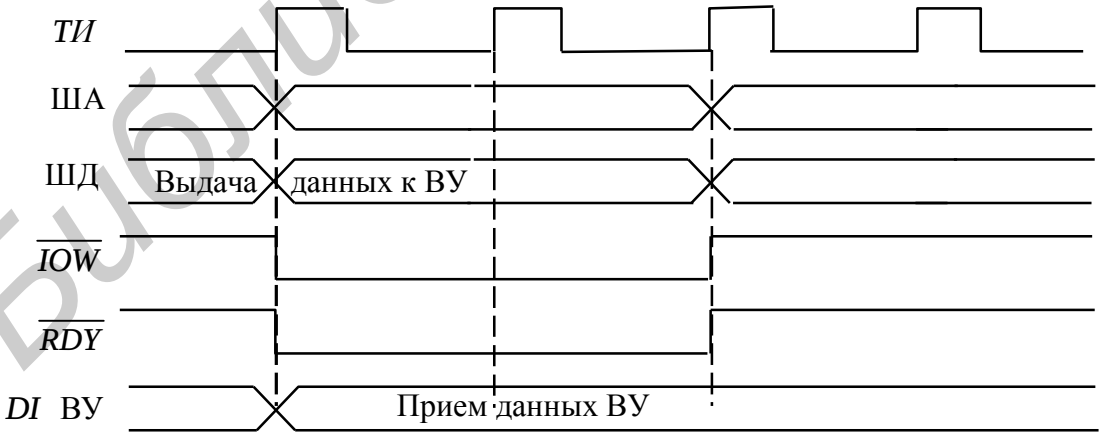


Рис. 3.5

К недостатку рассмотренного порта следует отнести наличие разделенных информационных входов и выходов в схеме УВВ, что предполагает дополнительные аппаратные затраты на коммутацию шин во внешнем устройстве.

3.4. Организация двунаправленного обмена через порт ввода–вывода

Применение коммутирующих элементов с тремя состояниями на выходе позволило создавать простые устройства ввода–вывода и организовывать двунаправленный обмен с периферийным устройством по одной информационной шине (рис. 3.6).

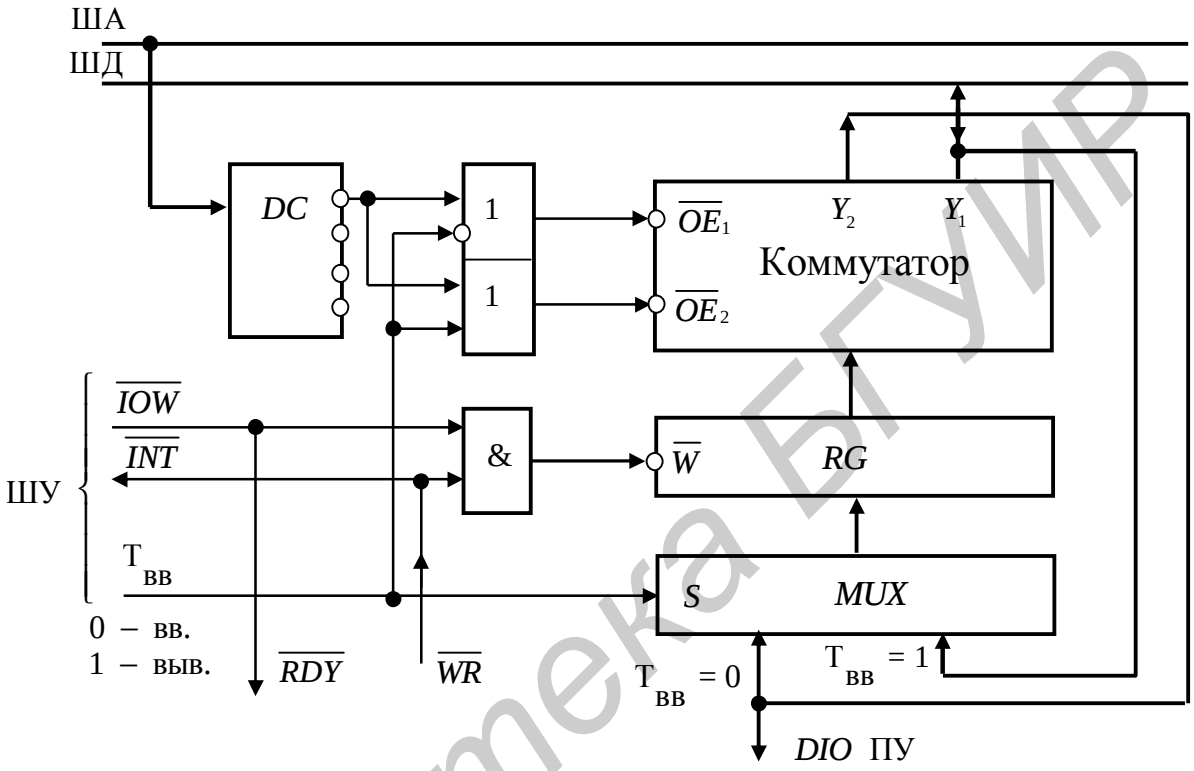


Рис. 3.6

При вводе информации триггер ввода $T_{ВВ}$ устанавливается в 0 и данные от ВУ через первый вход мультиплексора передаются на вход внутреннего регистра порта. Сигнал $\overline{IOW}$ удерживается в единичном состоянии, следовательно, по нулевому уровню $\overline{WR}$ вводимая информация может быть записана в порт. Процессор компьютера информируется о записи слова в порт нулевым значением сигнала $\overline{INT}$. Реагируя на этот запрос, процессор выдает на ША адрес порта, и сигнал выборки с выхода дешифратора разрешает функционирование коммутатора. Так как $T_{ВВ}$= 0, то в коммутаторе разрешаются Y_1-выходы. Линии Y_2 находятся в третьем состоянии. Данные из регистра передаются на выходы Y_1 коммутатора и далее на ШД компьютера.

При выводе информации $T_{ВВ}$ устанавливается в единицу. На шину адреса выдается адрес порта, и дешифратор сигналом $\overline{Вб}$ разрешает его работу. Сигнал $\overline{IOW}$ на шине управления устанавливается в нулевое состояние, после чего данные по пути ШД $\rightarrow$ второй вход мультиплексора $\rightarrow$ *RG* порта передаются в порт.

Выдача информации непосредственно к ВУ осуществляется по пути: *RG* порта $\rightarrow Y_1$ выходы коммутатора $\rightarrow$ ШД внешнего устройства.

3.5. Прерывания программ от устройств ввода–вывода

Под прерыванием программ понимают способность компьютера прервать вычислительный процесс в ответ на внешний запрос и выполнить специальную подпрограмму, предназначенную для обслуживания устройств ввода–вывода.

Прерывание программы пользователя осуществляется путем перехода компьютерной системы в режим анализа запросов и выполнения процедур распознавания портов. Причем этот переход инициируется не командой в текущей программе, а внешним сигналом, который называют запросом на прерывание.

Различают два способа обслуживания запросов.

Первый способ – обработка прерываний системой с программным опросом – предполагает наличие главной программы обработки прерываний, которая в специально отведенные моменты времени проверяет состояние каждого из периферийных устройств и находит ВУ, запросившее обслуживание.

Второй способ – обработка прерываний векторной системой – предполагает наличие аппаратных средств, которые текущему множеству запросов ставят в соответствие номер ВУ, имеющего наивысший приоритет среди устройств, сформировавших сигналы прерывания на данный момент времени. Достоинство системы заключается в малых затратах времени на расшифровку приоритетов и обслуживание портов-источников.

3.5.1. Элементы управления системы прерываний (координация взаимодействия ВУ и компьютера)

Большинство периферийных устройств работает асинхронно по отношению к компьютеру. Поэтому часто возникает задача согласования моментов срабатывания регистров портов и регистров ВУ в процессе обмена двоичными данными. Чтобы не происходило накладок в циклах ввода–вывода, взаимодействие компьютера и ВУ происходит по определенным правилам. Эти правила называют протоколом взаимодействия устройств и системы.

В общем случае обмен данными через порт осуществляется в 4 этапа.

1. Запрос порта состояния о текущем состоянии внешних устройств.
2. Выдача сигналов управления в порт управления.
3. Собственно обмен данными через порт ввода–вывода.
4. Возврат к прерванной программе.

Один из методов синхронизации процессов обмена информацией через порт предполагает использование специальных триггеров при выполнении операций ввода–вывода (рис. 3.7). Эти элементы памяти получили название квитирующих триггеров и используются для организации квитирующих пар, синхронизирующих ввод и вывод информации при работе устройств с разными скоростями.

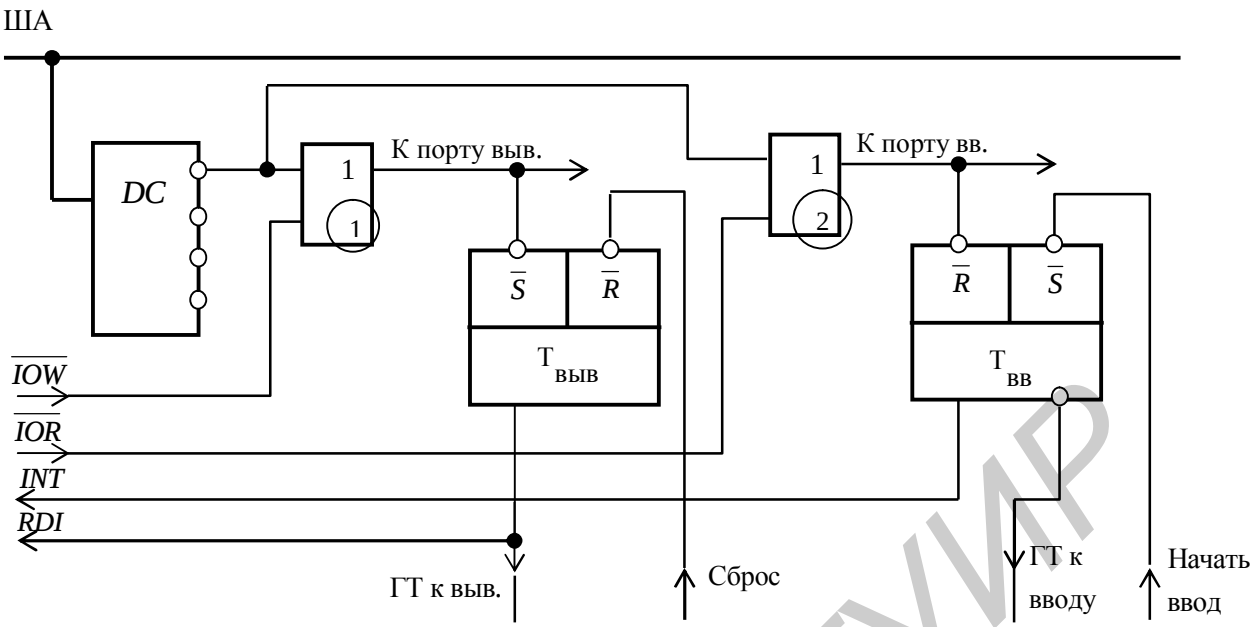


Рис. 3.7

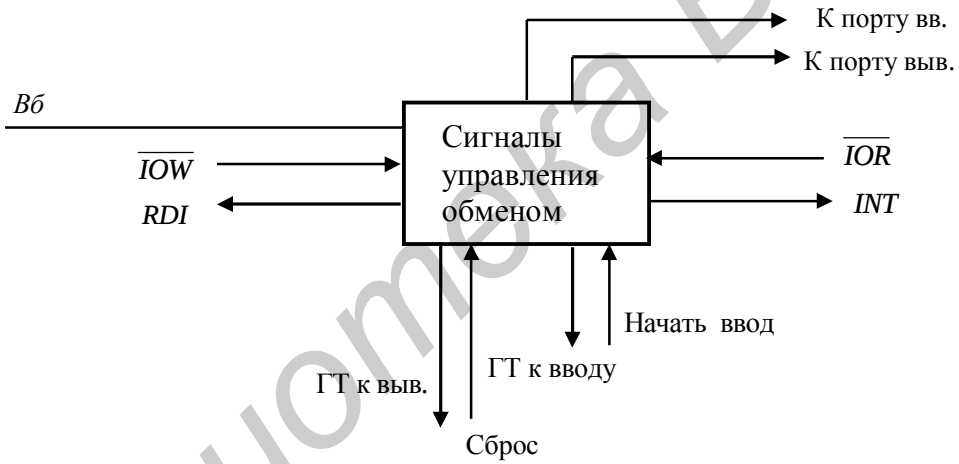


Рис. 3.8

При выводе данных на шину адреса выдается адрес порта, который расшифровывается дешифратором внешних устройств. Дешифратор возбуждает на своем выходе соответствующую линию выборки и готовит элементы 2ИЛИ к переключению. Далее на линии $\overline{IOW}$ шины управления устанавливается нулевой уровень, что вызывает переключение первого элемента 2ИЛИ и установку триггера вывода в состояние единицы. ВУ воспринимает этот сигнал как сигнал готовности компьютера к обмену через порт. После завершения очередного цикла вывода периферийное устройство переключает соответствующий триггер в ноль подачей на вход $\overline{R}$ сигнала «Сброс».

Для приема данных компьютер предоставляет каждому из внешних устройств свой триггер ввода. Этот триггер устанавливается ВУ, если существует необходимость передать информацию в систему или транзитно на другое

внешнее устройство. Триггеры ввода всех портов неявно объединяются в единый регистр запросов. Наличие хотя бы одного запроса на ввод влечет за собой выполнение специальной программы обслуживания периферийных устройств. При этом на шину адреса выдается адрес порта, на шину управления – активный уровень сигнала $\overline{IOR}$. Вводимая информация читается на ШД компьютера, одновременно триггер ввода сбрасывается в ноль. Внешнее устройство оповещается о готовности к новому циклу обмена сигналом с инверсного вывода элемента памяти.

Пары сигналов, функционально связанные при обмене, указаны на рис. 3.8.

3.5.2. Обработка прерываний от УВВ системой с программным опросом

При обработке прерываний системой с программным опросом все триггеры внешних устройств ($Т_{ВВ}$) и триггеры состояния портов вывода ($Т_{ВЫВ}$) объединяются в общий регистр состояния (рис. 3.9). Обслуживание схем УВВ организуется путем чтения и расшифровки содержимого регистра, в процессе чего выясняется порядок обработки поступивших заявок в зависимости от заранее назначенных приоритетов.

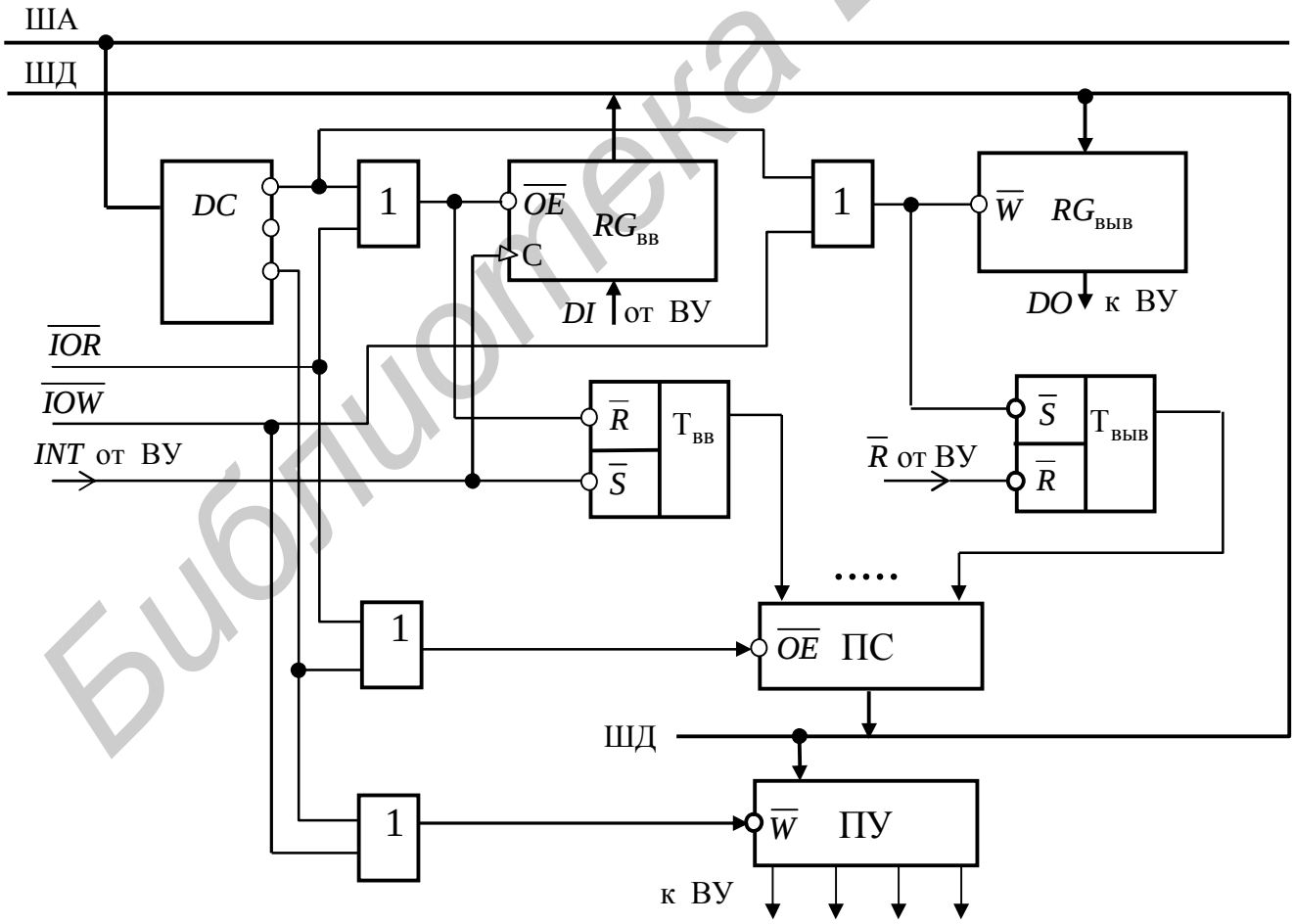


Рис. 3.9

При выводе информации по данной схеме вначале выполняется программирование порта управления ПУ программой инициализации УВВ. Это позволяет установить требуемый режим обмена между компьютером и периферийным устройством. После того как режим установлен, в регистр порта записывается выводимая информация, а в порт управления – сигнал Гт – готовности к выводу. Одновременно триггер вывода порта переключается в единицу. Далее компьютер циклически (или в заданные моменты времени) опрашивает порт состояния ПС, ожидая сброса $Т_{ВЫВ}$ от схемы ВУ.

ВУ принимает данные в один из своих регистров и посылает по линии связи «$\overline{R}$ от ВУ» сигнал установки $Т_{ВЫВ}$ в ноль. Компьютер реагирует сбросом сигнала готовности в соответствующем разряде ПУ и организует новый цикл обмена между компьютером и УВВ.

При вводе информации на входе данных порта ввода устанавливается информационное слово и выполняется его запись путем подачи импульса синхронизации на *C* вход регистра (сигнал «*INT* от ВУ»). Одновременно триггер ввода устанавливается в единицу и в режиме опроса в составе вектора состояния УВВ (из порта состояния) читается в один из регистров процессора. Расшифровав состояние УВВ, процессор формирует управляющее слово для порта управления. В частности, это слово содержит сигнал «Не готов», в ответ на который ВУ отключается, ожидая завершения фазы приема данных компьютером. Для чтения данных на ША выдается адрес порта ввода, на ШУ – сигнал $\overline{IOR}$. При этом выполняется подключение порта к ШД компьютера и сброс $Т_{ВВ}$ в нулевое состояние. Далее в порт управления посылается сигнал Гт – готовности компьютера к новому циклу обмена информацией.

3.5.3. Векторная система обработки запросов с идентификацией устройств при помощи адресов

К достоинствам векторной системы следует отнести оперативность перехода вычислительной системы к подпрограмме обслуживания устройств ввода–вывода. Это обусловлено тем, что определение номеров ВУ, запросивших обмена, и расшифровка их приоритетов выполняется в компьютере с использованием набора аппаратных средств. Таким образом, особенностью данной системы обработки прерываний является формирование аппаратурой порта на ША компьютера непосредственно адреса порта или области оперативной памяти, в которой хранится программа обработки запроса от данного УВВ.

При поступлении запросов от ВУ процессы обработки прерываний отрабатываются так, как показано на рис. 3.10. Порт, запросивший обслуживания, устанавливает соответствующий триггер ввода в единичное состояние. Установка хотя бы одного из триггеров в единицу, если обслуживание порта не запрещено (M_i = 1), определяет формирование на специальном входе процессора *INT* активного логического уровня. Это предполагает выдачу на шину управления сигнала *INTA*, что при Q =1 определяет подключение к шине данных компьюте-

ра регистра адреса *RG A*. Предварительно в этот регистр загружается адрес подпрограммы обслуживания прерываний от текущего УВВ.

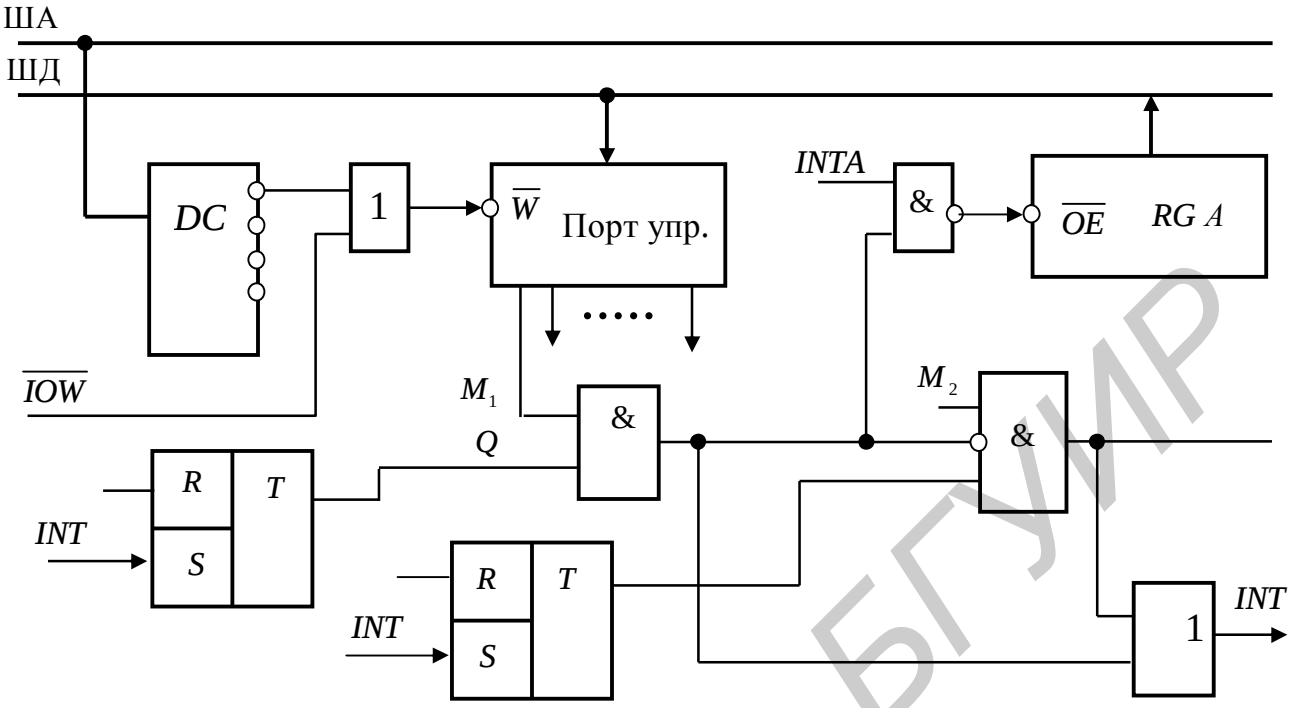


Рис. 3.10

В режиме опроса устройств ввода–вывода информация из *RG A* читается на шину данных компьютера. Процессор принимает идентификатор порта в один из своих регистров и далее, если принятый код является адресом порта, выдает его на системную шину адреса. Если считанный код представляет собой отвлеченное значение и не является физическим адресом порта, то формируется обращение к соответствующей подпрограмме, которая позволяет вычислить исполнительные адреса и выдать их на ША компьютера. После завершения обслуживания УВВ триггер прерывания сбрасывается в ноль.

Рассмотрим схему задания приоритетов устройствам ввода–вывода (рис. 3.11) и распространение запрещающих сигналов в разрядах, соответствующих низкоприоритетным УВВ.

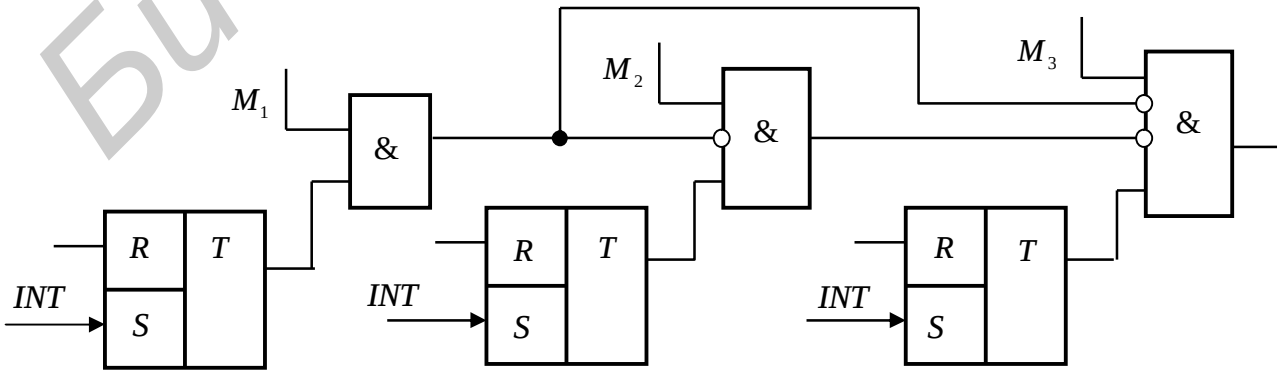


Рис. 3.11

В приведенной схеме последовательно включенные элементы 2И, 3И, 4И и т.д. используются для задания приоритетов внешним устройствам. Если из *i*-го разряда порта управления поступает сигнал маски $M_i = 0$, то соответствующий логический элемент 2И оказывается заперт. Соответствующее УВВ с номером *i* с обслуживания снимается. Нулевой сигнал с его выхода разрешает обработку запросов во всех следующих *i*+1, *i*+2, *i*+3… устройствах системы. Кроме каналов ввода с большими чем *i* номерами в системе оказываются разрешенными УВВ с *i*–1, *i*–2, *i*–3… номерами, приоритеты которых тем выше, чем меньше номер устройства. Причем в текущий момент времени компьютер аппаратно связан только с одним наиболее приоритетным (левым по схеме) портом. После обслуживания соответствующего прерывания приоритетным становится очередное, разрешенное сигналом M_i устройство, стоящее правее по схеме.

Принцип предпочтения УВВ с меньшими номерами обусловлен тем, что срабатывание триггера ввода в *i*-м канале схемы автоматически вызывает распространение запрещающего единичного сигнала во всех каналах выдачи адреса с номерами *i*+1, *i*+2, *i*+3 … . Данный сигнал воздействует на инверсные входы элементов И, устанавливая на их выходах уровни логического нуля.

3.5.4. Векторная система обработки запросов с шифраторами приоритетов

В общем случае расшифровка приоритетов устройств ввода информации может быть выполнена с использованием схемы шифратора (рис. 3.12).

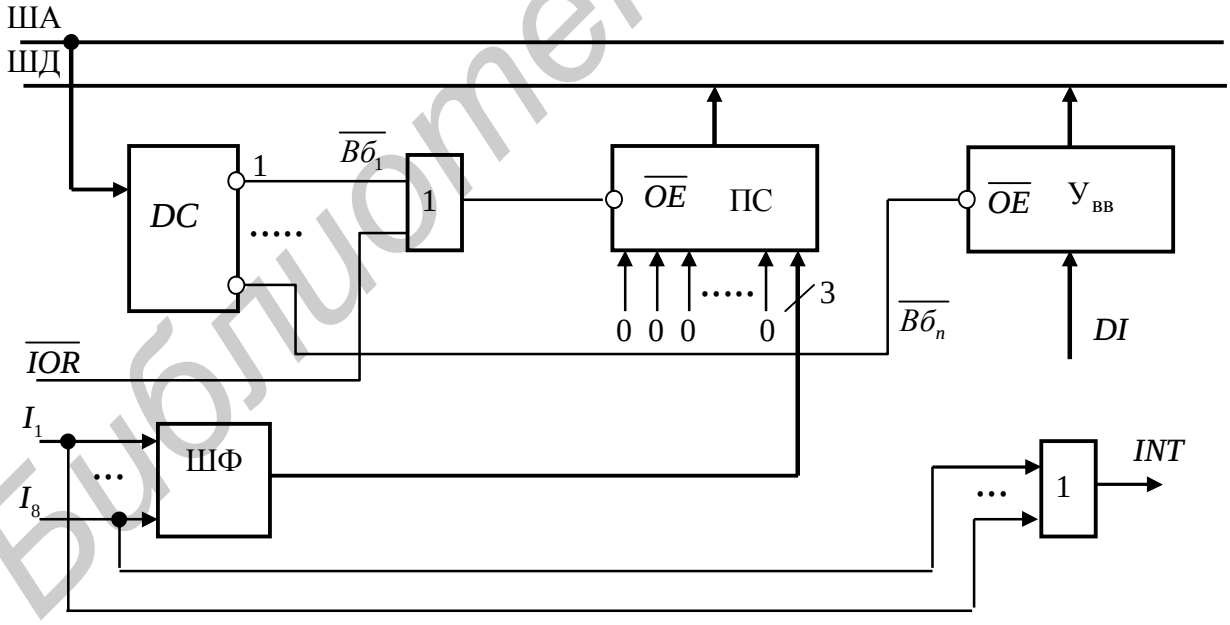


Рис. 3.12

Данная схема предполагает наличие входа «Запрос на прерывание» (*INT*) в блоке центрального процессора. При поступлении нескольких запросов шифратор ШФ формирует код внешнего устройства, имеющего наивысший приоритет. Этот код заносится в порт состояния и считывается в один из регистров

процессора при реакции последнего на сигнал *INT*. Далее принятое значение обрабатывается специальной программой и выдается в виде адреса на ША компьютера. Сформированный адрес порта расшифровывается схемой дешифратора, на выходе которого формируется сигнал $\overline{Вб_n}$. Это определяет на входе $\overline{OE}$ порта ввода нулевой уровень, в результате чего информация по линиям шины данных передается в процессор.

В общем случае схема шифратора приоритетов может быть построена по известной методике синтеза комбинационных схем.

3.5.5. Векторная система обработки запросов с преобразователем адреса

Использование в системе в качестве преобразователя «начального» адреса стандартной схемы ПЗУ приводит к конфигурации устройства обслуживания запросов, изображенной на рис. 3.13.

В приведенной схеме сформированный вектор запросов используется как адрес ячейки памяти ПЗУ. Эта ячейка хранит адрес подпрограммы обслуживания одного из устройств ввода–вывода, приславших на данный момент времени сигналы прерывания. Причем каждому подмножеству запросов в ячейках ПЗУ ставится в соответствие свой двоичный эквивалент, кодирующий адрес подпрограммы обработки прерываний от УВВ, имеющего наивысший приоритет для данного подмножества заявок.

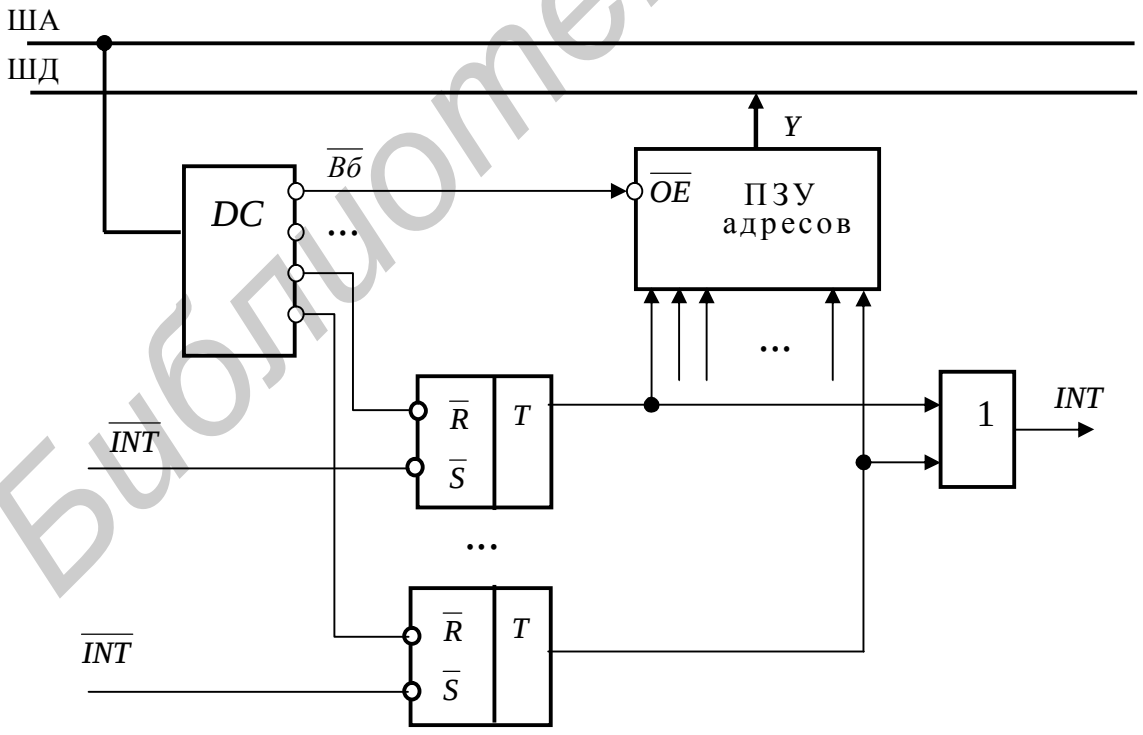


Рис. 3.13

При поступлении сигнала прерывания от ВУ формируется запрос на обслуживание *INT*, поступающий в центральный процессор. Процессор отвечает выдачей на адресную шину номера преобразователя адресов, в результате чего на выходе дешифратора порта формируется нулевой уровень $\overline{Вб}$. Данный сигнал подключает выходную шину ПЗУ к шине данных компьютера, и адрес наиболее приоритетного из портов передается из ПЗУ в процессор. После завершения обмена триггер запроса *T* сбрасывается в ноль путем адресации входа триггера $\overline{R}$ как стандартного порта для пересылки данных.

Особенностью схемы является использование модуля ПЗУ, которое выполняет следующие системные функции:

1. Регистрацию запросов на прерывание.
2. Распознавание порта с наивысшим приоритетом для текущего подмножества заявок.
3. Формирование адресов подпрограмм обслуживания прерываний от ВУ.

3.5.6. Обмен данными через двунаправленный порт с квитированием

Рассмотрим процесс передачи данных через порт при наличии двунаправленной информационной шины (рис. 3.14).

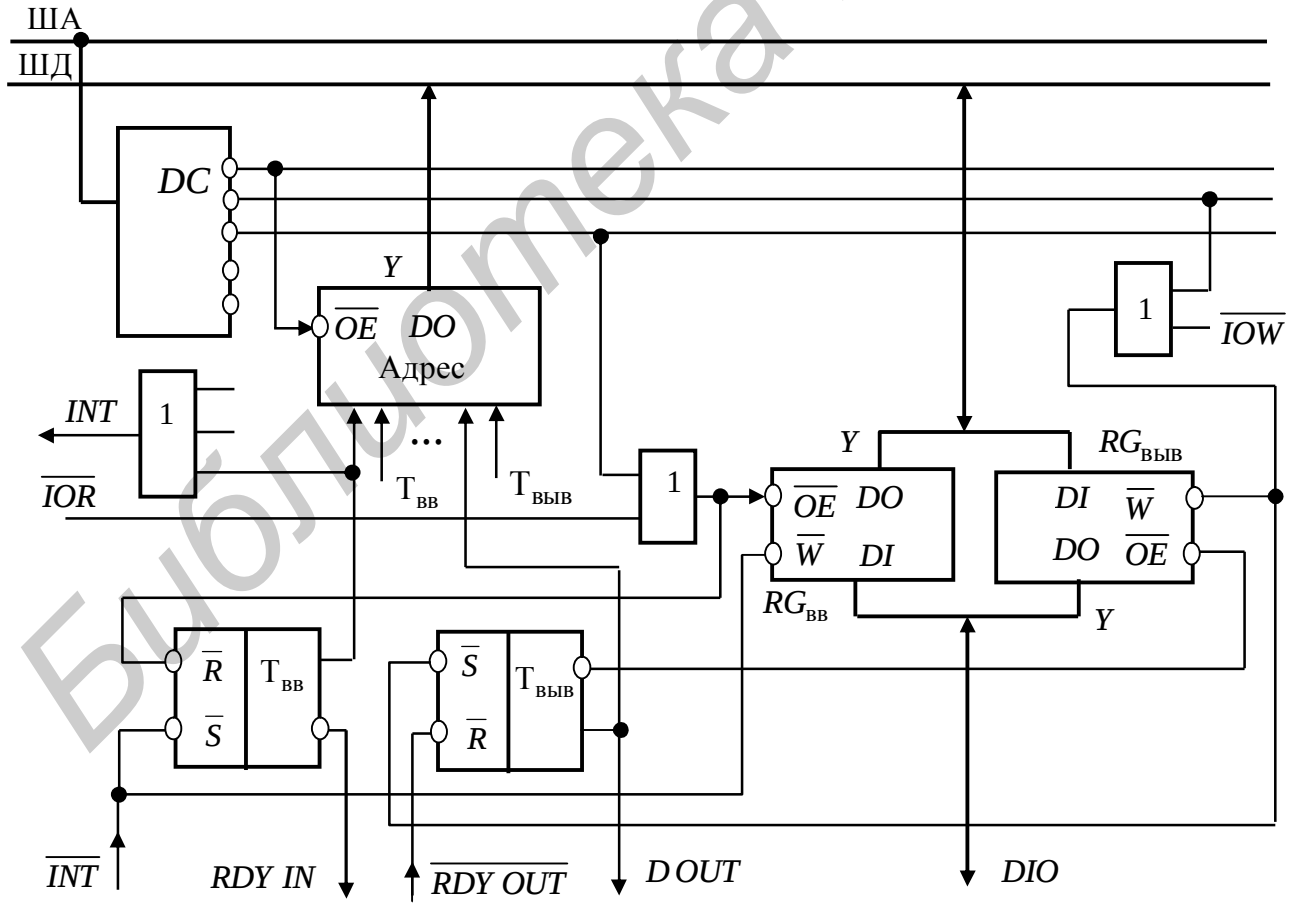


Рис. 3.14

В режиме ввода информации инициатором обмена является внешнее устройство. ВУ-источник данных посылает в регистр запросов сигнал $\overline{INT}$ и устанавливает свой триггер ввода в единичное состояние. Одновременно нулевой уровень данного сигнала отпирает входную шину регистра ввода, так как $\overline{W} = 0$, что позволяет записать вводимое слово с двунаправленной шины непосредственно в порт. Линии *DO* регистра вывода в данном режиме отключены. Режим отключения обусловлен тем, что инверсный выход триггера, используемого при выводе, в текущий момент времени установлен в единицу.

Поступивший запрос на ввод (выход $T_{ВВ}$) через схему многовходового элемента ИЛИ сигнализирует процессору компьютера о необходимости обмена информацией через порт. Процессор отвечает выдачей на шину адреса номера преобразователя адреса, в качестве которого используется ПЗУ с тремя состояниями на выходе.

Дешифратор периферийных устройств принимает адрес и переводит соответствующую линию выборки в нулевое состояние, при этом информационные выходы ПЗУ переключаются в рабочий режим. Сформированный к данному моменту времени вектор состояния УВВ адресует в преобразователе ячейку памяти, которая содержит адрес устройства наиболее приоритетного из текущего подмножества УВВ, приславших запросы на ввод.

Считанный код принимается в регистр-аккумулятор процессора, а затем выдается на системную шину адреса. Дешифратор порта преобразует двоичный код адреса в унитарный, что позволяет подготовить элемент 2ИЛИ регистра ввода к переключению. Моменту переключения элемента соответствует появление на линии $\overline{IOR}$ шины управления нулевого логического уровня. Таким образом, совместное действие двух сигналов определяет на входе $\overline{OE}$ регистра ввода уровень логического нуля, и его выходная шина *Y* переводится в рабочий режим. Вводимые данные передаются на шину данных компьютера, одновременно триггер $T_{ВВ}$ сбрасывается в ноль.

Единичный уровень с инверсного выхода триггера ввода *RDY IN* сигнализирует источнику данных о завершении цикла обмена. Канал ввода информации снова готов к пересылке слова.

Процесс вывода данных сопровождается выдачей на шину адреса адреса порта, на шину данных – выводимого слова. Адрес порта преобразуется в унитарный код дешифратором устройств ввода–вывода, и соответствующая линия выборки готовит к срабатыванию элемент 2ИЛИ регистра вывода.

Далее, на шину управления выдается сигнал записи в порт вывода $\overline{IOW}$, что вызывает срабатывание логического элемента 2ИЛИ и установку на входе $\overline{W}$ регистра вывода нулевого уровня. Входная информационная шина регистра вывода оказывается разрешенной, и слово с шины данных компьютера записывается в порт. Этим же сигналом триггер вывода устанавливается в единицу, оповещая периферийное устройство сигналом *D OUT* о загрузке данных в регистр вывода. Реагируя на данный сигнал, внешнее устройство принимает выводимую информацию и сбрасывает триггер вывода в нулевое состояние, пере-

сылая через интерфейс сигнал $\overline{RDY\ OUT}$. На этом текущий цикл вывода завершается.

3.6. Принципы организации последовательного ввода–вывода информации

Различают два основных способа передачи данных через последовательный канал: 1) синхронная передача информации, 2) асинхронная передача информации.

Синхронный способ передачи предполагает наличие двухпроводной линии связи между источником и приемником сообщений.

В источнике информации предусматривается сдвиговый регистр, который преобразует параллельный код компьютера в последовательный код канала связи. Одновременно с записью информации в этот регистр вычисляется бит контроля четности, формируемый как сумма по модулю два всех информационных разрядов. Запись в порт девятиразрядного слова осуществляется при подаче положительного фронта управляющего сигнала на вход записи *W*. Этому моменту времени соответствует установка в счетчике тактов *Q* кодовой комбинации 1001 и появление отрицательного фронта синхроимпульса на выходе ГТИ.

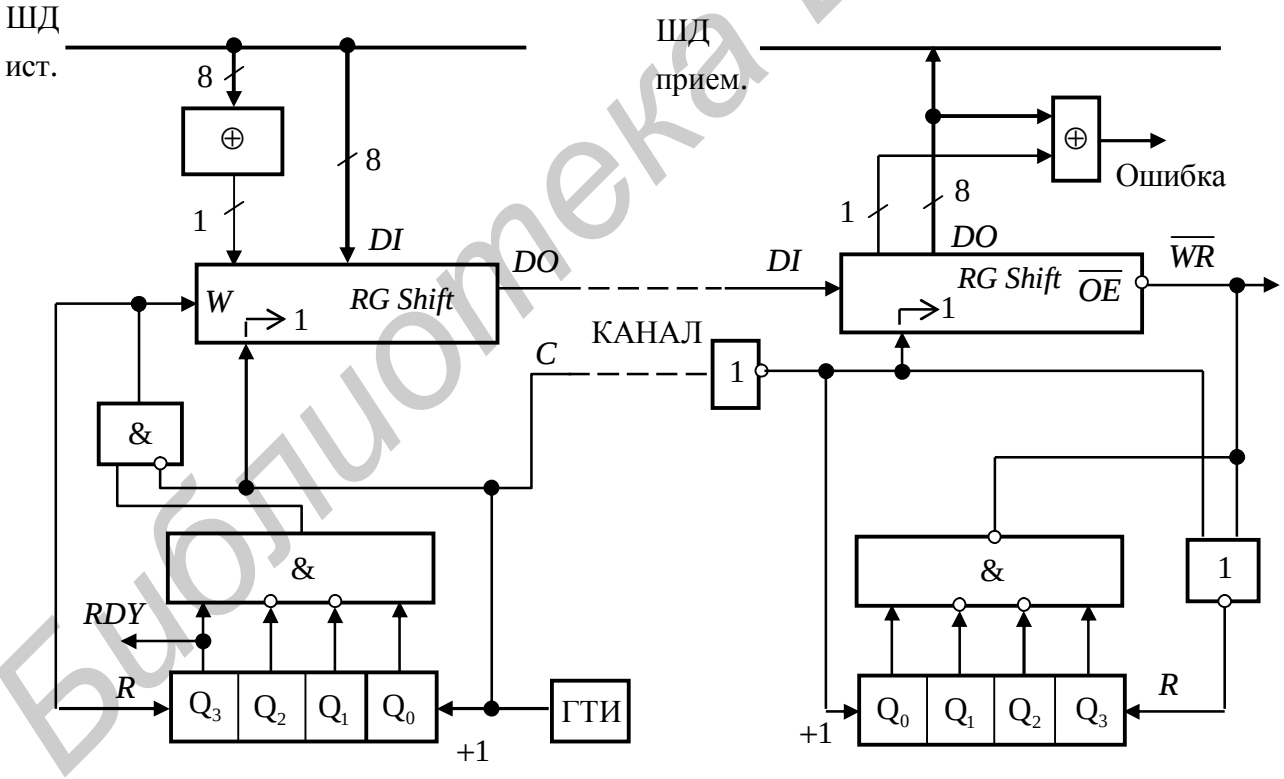


Рис. 3.15

Сигнал записи в сдвиговый регистр источника используется также для сброса счетчика в ноль перед началом очередной серии сдвиговых операций. Непосредственно вывод байта в канал сопровождается битом контроля четности, который передается из последовательного порта последним.

Приемник информации так же, как и источник, включает в свой состав девятиразрядный регистр сдвига, используемый для преобразования последовательного кода канала в параллельный код системы. В режиме приема данных биты поступают на вход *DI* порта и записываются в регистр сдвига по приходу синхроимпульсов из канала. Число принятых бит определяется наполнением синхронного счетчика. Если последний устанавливается в состояние 1001, то на выходе схемы 4И-НЕ формируется сигнал нулевого уровня. Он передается на вход $\overline{OE}$ регистра сдвига, переводя его выходную шину в рабочий режим. Принятый код выдается на шину данных приемника, после чего записывается в буферный регистр при помощи сигнала низкого уровня $\overline{WR}$. Очередное переключение импульса сдвига в состояние нуля вызывает формирование на выходе элемента 2ИЛИ сигнала «Сброс» счетчика. После этого цикл приема данных повторяется снова.

При выдаче информации на шину данных приемника выполняется контроль байта на четность. При этом изменение информации в нечетном числе бит проявляется единичным потенциалом сигнала на линии «Ошибка».

Асинхронный обмен по однонаправленной линии связи представляет собой более сложный вид обмена информацией. При этом схема передатчика в последовательный канал представляет собой устройство, показанное на рис. 3.16.

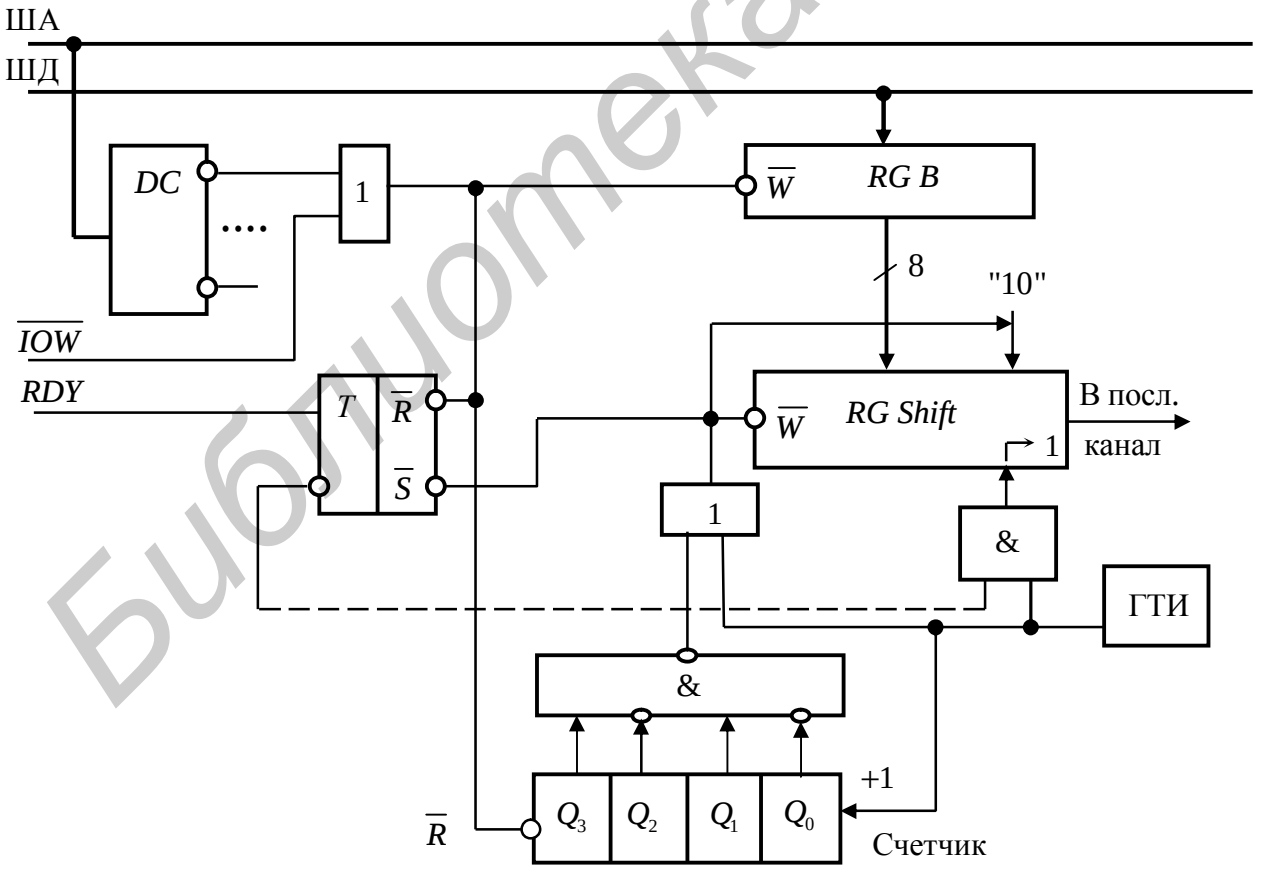


Рис. 3.16

В приведенной схеме триггер управления обменом через порт в исходном состоянии установлен в единичное состояние. Это состояние воспринимается процессором как готовность буферного регистра *RG B* к приему пересылаемых данных. Кроме того, инверсный выход триггера может отключать (не обязательно) регистр сдвига от шины тактирования, и схема устройства в целом находится в состоянии ожидания. Если сдвиговые операции не выключены, то в канал связи передаются нулевые значения из регистра.

Для вывода информации на шине адреса устанавливается адрес порта, а на шине данных – байт пересылаемого сообщения. Линии $\overline{IOW}$ шины управления придается нулевой уровень, который совместно с сигналом дешифратора $\overline{Bб}$ разрешает по входу $\overline{W}$ запись информации в буферный регистр. При этом триггер обмена переключается в ноль, разрешая вывод предыдущего байта из регистра сдвига в последовательный канал.

По истечении десяти сдвиговых операций триггер обмена устанавливается в единицу. Это состояние информирует процессор о выполнении действий, связанных с пересылкой байта. Требуемое управление переключением триггера и прекращение передачи информации в канал достигается за счет того, что элемент 4И-НЕ канала регистрирует двоичную комбинацию 1010, устанавливающуюся в счетчике порта в процессе счета. Инверсный выход элемента формирует уровень логического нуля, который передается на вход схемы 2ИЛИ. В свою очередь элемент 2ИЛИ с приходом инверсного сигнала от ГТИ формирует управляющие воздействия: 1) сигнал $\overline{W}$ записи информации в регистр сдвига; 2) импульс установки триггера обмена в единичное состояние; 3) запись синхропары в регистр сдвига.

Для обеспечения непрерывного функционирования сдвигового регистра процессор в течение ограниченного промежутка времени (рис. 3.17) должен переслать очередной байт информации в буферный регистр. Если этого не происходит, то передача данных в канал автоматически запрещается. Кроме байта данных в канал связи передается пара синхросигналов 0 и 1, которые определяют начало каждого слова и используются приемником для взаимной синхронизации.

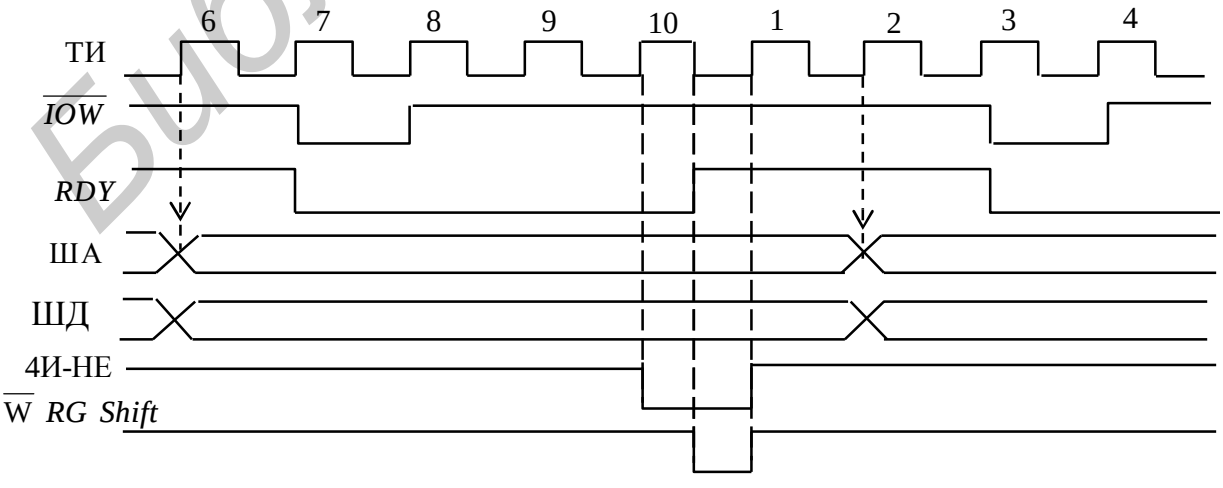


Рис. 3.17

Асинхронный приемник информации из последовательного канала может быть построен по схеме, приведенной на рис. 3.18.

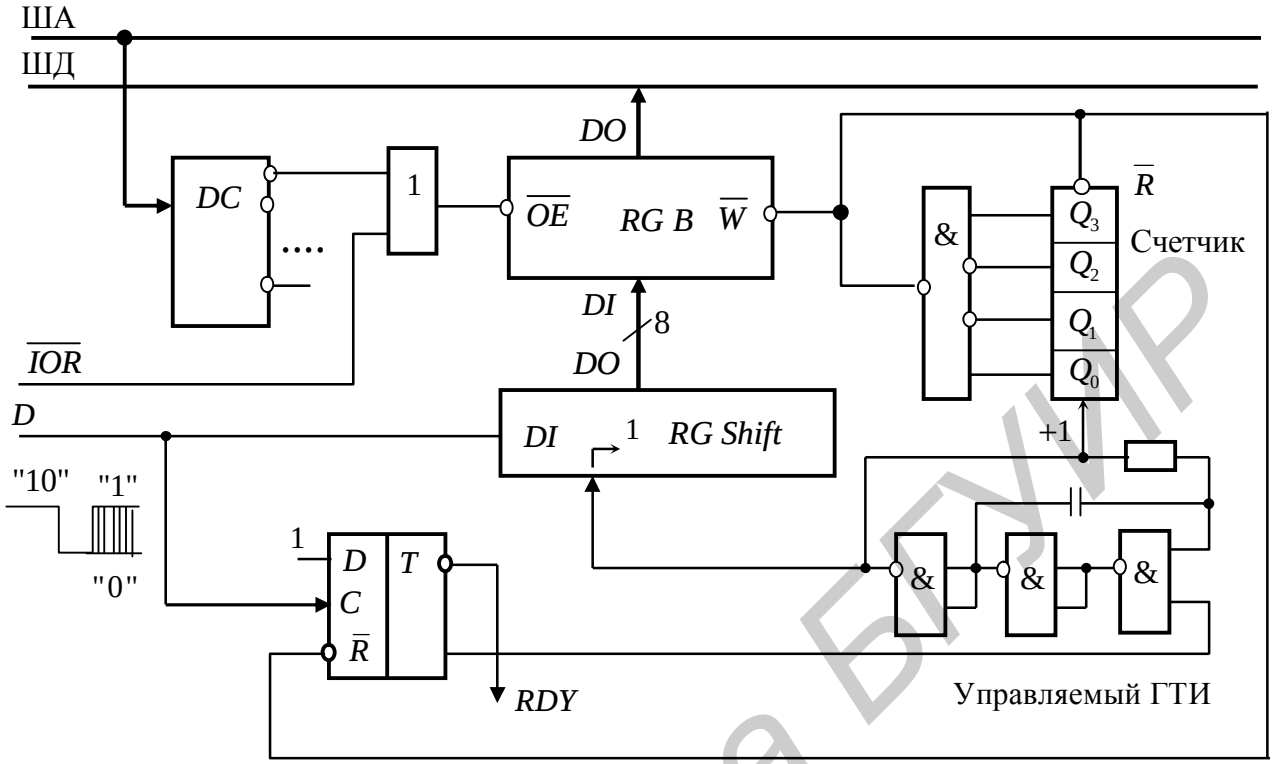


Рис. 3.18

В исходный момент времени триггер порта находится в состоянии нуля: в связи с этим управляемый генератор импульсов отключен от шины тактирования и переведен в режим ожидания. Счетчик количества принятых бит сброшен в ноль и также находится в состоянии ожидания.

В режиме ожидания приема на входе *D* устройства присутствует либо высокий, либо низкий уровень сигнала. Это определяет присутствие на синхровходе триггера управления пассивного потенциального состояния. Если состояние входа *C* равно нулю, то имеет место совпадение первого бита синхропары и текущего потенциального уровня, следовательно, процесс приема информации развивается по стандартной процедуре (см. ниже). Если состояние входа *C* равно единице, то процесс синхронизации приема задерживается до прихода синхропары «0,1».

Приход нулевого сигнала синхронизации вызывает появление низкого уровня на входе *C* триггера, однако состояние его не изменяется.

Приход единичного синхросигнала вызывает переключение элемента памяти в единицу, что влечет за собой включение генератора тактовых импульсов и последовательный ввод данных в регистр приемника.

В момент начала приема на входе регистра сдвига присутствует нулевой уровень синхросигнала. Этот сигнал не фиксируется, так как сдвиг регистра в данный момент времени запрещен. Далее схема приемника формирует девять

импульсов сдвига, что позволяет принять полезную информацию и выдвинуть из регистра второй бит синхронизации, который записывается при первом такте сдвига.

В процессе функционирования счетчик сдвигов регистрирует количество принимаемых бит и вырабатывает импульс переключения триггера в ноль. Это позволяет: 1) переслать единичный сигнал готовности к процессору (инверсный выход триггера); 2) переслать байт данных в буферный регистр ($\overline{W}_{RG\,B} = 0$); 3) сбросить в ноль счетчик сдвигов; 4) последовательный порт ввода перевести в режим ожидания.

В промежутке времени между соседними двумя синхропарами процессор должен принять данные из $RG\,B$, в противном случае происходит сбой устройства при приеме информации.

3.7. Организация прямого доступа в память

Такой тип обращения к памяти организуется при обслуживании внешних устройств, которые требуют быстрого ввода данных в режиме наиболее приоритетного прерывания. Как правило, при вводе используются аппаратные средства самого канала без участия процессорного блока или устройства управления (рис. 3.19).

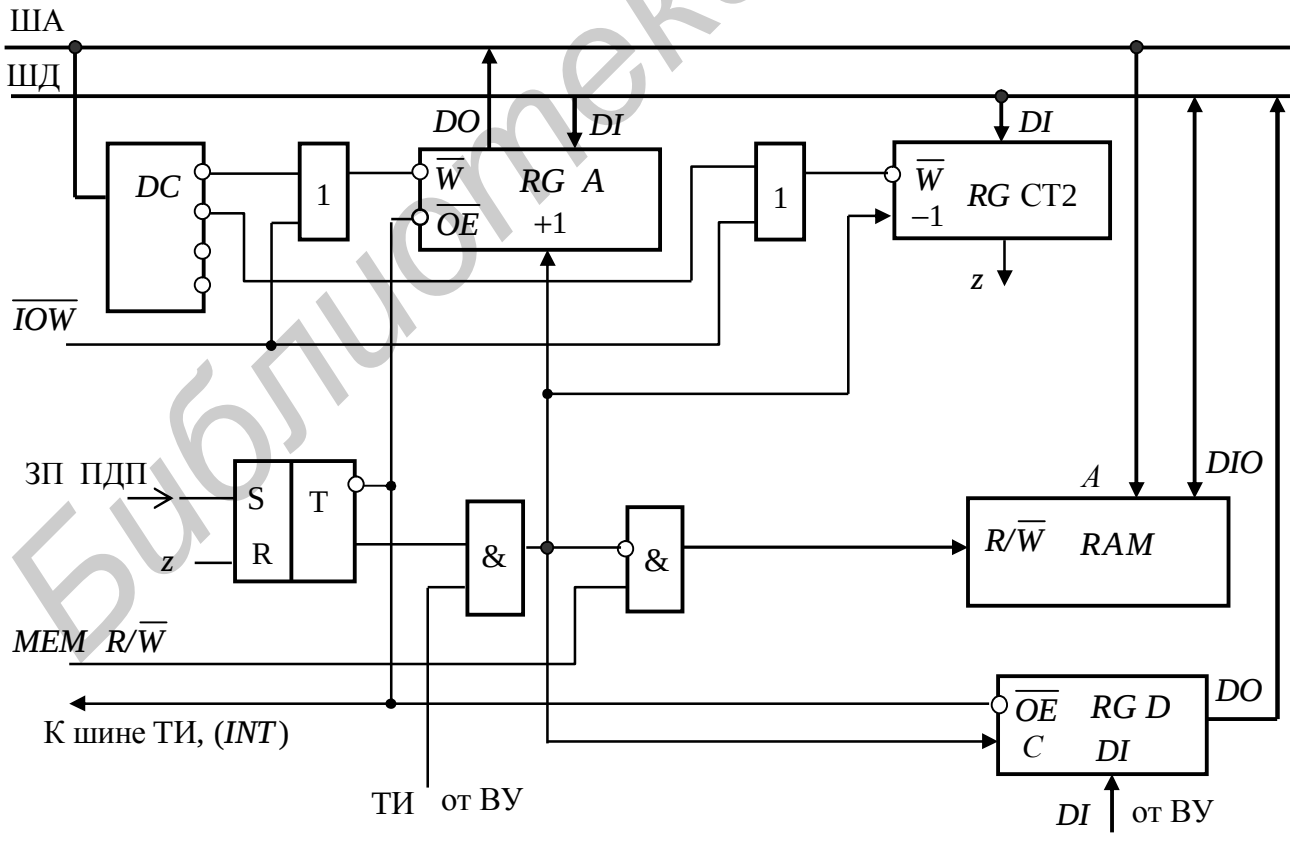


Рис. 3.19

ВУ, использующие канал ПДП, обычно передают информацию в виде сообщений заданной длины. При этом, если компьютер ожидает обмена через порт, то специальная программа операционной системы предварительно инициализирует регистр A канала начальным адресом области памяти, используемой для приема данных. В регистр $RG\ CT2$ – счетчик длины сообщения – заносится информация о количестве слов, принимаемых от ВУ. В целом адресация регистров канала ПДП осуществляется точно так же, как и обычных портов ввода–вывода.

Классически при поступлении запроса на ПДП (*INT*) процессор приостанавливает вычислительный процесс и переводит информационную шину и шину адреса в третье состояние. После этого выдается импульс подтверждения ПДП, устанавливающий соответствующий триггер состояние единицы. С этого момента времени шины компьютера полностью передаются в распоряжение канала.

Схема, приведенная на рис. 3.19, предполагает, что компьютер, использующий блок ПДП, имеет разрядно-модульную организацию связей. При этом сигнал *INT* не анализируется процессором, а напрямую отключает синхронизацию системы на время ввода данных.

В режиме ввода информации нулевое значение с инверсного выхода триггера активизирует входы $\overline{OE}$ регистра $RG\ A$ и регистра входных данных $RG\ D$ и переводит их шины *DO* в рабочий режим. Прямой выход триггера разрешает прохождение импульсов синхронизации от ВУ через двухвходовый элемент И к элементам канала. Это обеспечивает:

1) инкрементирование адреса памяти в *RG A*;

2) вычитание счетчика слов в $RG\ CT2$;

3) запись входной информации в *RG D* и

4) выдачу содержимого *RG D* на шину данных компьютера.

В режиме ПДП сигнал $MEM\ R/\overline{W}$ шины управления должен находиться в состоянии единицы. Таким образом, инверсные синхроимпульсы, приходящие на соответствующий вход ОЗУ от ВУ, используются как импульсы записи информации в память противофазно относительно данных, принимаемых в *RG D*.

После того как блок данных принят, регистр-счетчик канала ПДП устанавливается в ноль. На его специальном выходе формируется признак нулевого результата *z*. Этот сигнал сбрасывает триггер ПДП в ноль, и процессор информируется о завершении обмена потенциалом логической единицы с инверсного выхода этого же триггера. Канал ПДП отключается от компьютера, шинные ресурсы системы передаются процессору.

ЛИТЕРАТУРА

1. Гивоне, Д. Микропроцессоры и микрокомпьютеры. Вводный курс / Д. Гивоне, Р. Россер; пер. с англ. – М. : Мир, 1983. – 464 с.

2. Хвощ, С. Т. Микропроцессоры и микроЭВМ в системах автоматического управления : справочник / С. Т. Хвощ, Н. Н. Варлинский, Е. А. Попов; под общ. ред. С. Т. Хвоща. – Л. : Машиностроение, 1987. – 640 с.

3. Проектирование цифровых систем на комплектах микропрограммируемых БИС / С. С. Булгаков [и др.]; под ред. В. Г. Колесникова. – М. : Радио и связь, 1984. – 240 с.

4. Каган, Б. М. Электронные вычислительные машины и системы / Б. М. Каган. – М. : Энергоатомиздат, 1985. – 464 с.

5. Майоров, С. А. Структура электронных вычислительных машин / С. А. Майоров, Г. И. Новиков. – Л. : Машиностроение, 1979. – 384 с.

6. Мик, Дж. Проектирование микропроцессорных устройств с разрядно- модульной организацией связей / Дж. Мик, Дж. Брик; пер. с англ. – М. : Мир, 1984. – 289 с.

7. Столингс, У. Структурная организация и архитектура компьютерных систем / У. Столингс; пер. с англ. – 5-е изд. – М. : Вильямс, 2001. – 892 с.

8. Таненбаум, Э. Архитектура компьютерных систем / Э. Таненбаум; пер. с англ. – 4-е изд. – М. : Питер, 2002. – 698 с.

9. Угрюмов, Е. Цифровая схемотехника / Е. Угрюмов. – М. : СПб, 2001. – 518 с.

10. Майоров, С. А. Введение в микроЭВМ / С. А. Майоров. – Л. : Машиностроение, 1988. – 484 с.

11. Мотока, Т. Компьютеры на СБИС. Кн. 1, 2 / Т. Мотока; пер. с англ. – М. : Мир, 1988. – 534 с.

12. Мячев, А. А. Интерфейсы вычислительных систем на базе мини- и микроЭВМ / А. А. Мячев. – М. : Радио и связь, 1986. – 342 с.

13. Калабеков, Б. А. Микропроцессоры и их применение в системах передачи и обработки сигналов / Б. А. Калабеков. – М. : Радио и связь, 1988. – 368 с.

14. Гук, М. Процессоры «PENTIUM». Архитектура, интерфейс, программирование... / М. Гук. – СПб. : Питер, 1999. – 288 с.

15. Шагурин, И. И. Процессоры семейства Intel P6. Архитектура, программирование, интерфейс / И. И. Шагурин. – М. : Телеком, 2000. – 248 с.

16. Грушвицкий, Р. Проектирование систем на микросхемах программируемой логики / Р. Грушвицкий. – СПб. : Питер, 2002. – 608 с.

17. Гук, М. Аппаратные интерфейсы. Энциклопедия / М. Гук. – СПб. : Питер, 2002. – 528 с.

Св. план 2008, поз. 56

Учебное издание

Кобяк Игорь Петрович

ОСНОВЫ ПРОЕКТИРОВАНИЯ КОМПЬЮТЕРНЫХ УСТРОЙСТВ

Учебно-методическое пособие
по курсу «Структурная и функциональная организация ЭВМ»
для студентов специальности I-40 02 01
«Вычислительные машины, системы и сети»
заочной формы обучения

Редактор Н. В. Гриневич
Корректор М. В. Тезина

Подписано в печать Формат 60×84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л.
Уч.-изд. л. 4,4. Тираж 100 экз. Заказ 166.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6