ОРГАНИЗАЦИЯ ЭВМ, КОМПЛЕКСОВ И СИСТЕМ

При изучении организации ЭВМ интересуются двумя основными моментами: структурой ЭВМ и ее функциональной организацией или архитектурой ЭВМ.

Под структурой ЭВМ понимается состав и принципы взаимодействия входящих в нее компонентов.

Под архитектурой ЭВМ будем понимать общую логическую организацию ЭВМ, определяющую процесс обработки данных в конкретной ЭВМ и включающую методы представления и кодирования данных, машинные операции, адреса и команды, в терминах которых могут быть представлены алгоритмы и процессы вычислений, принципы взаимодействия технических средств и программного обеспечения ЭВМ. Т.о. архитектура ЭВМ определяется совокупностью ее свойств, существенных для пользователя.

1. **Многоуровневое представление машины**

Вычислительную машину можно представить в виде нескольких уровней.

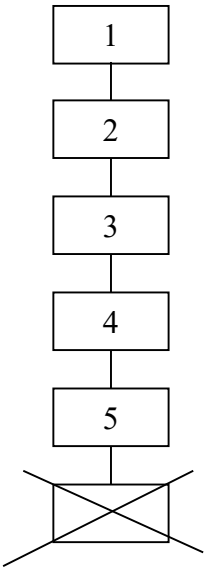


Рис.1. Многоуровневое представление машины

1. Прикладной уровень. На этом уровне реализованы различные пакеты прикладных программ (MS Office, MathCad, PhotoShop…), языки высокого уровня (C, C++, Pascal, Lisp…) и т.д. То есть этот уровень представляет собой профессиональную область, в которой решаются какие-то конкретные задачи.

2. Уровень операционной системы. Здесь находятся трансляторы, загрузчики и прочие программы более низкого уровня. Управление осуществляется на логическом уровне. В данной области работают математики и программисты.

3.Традиционный машинный уровень. Он же ассемблерный уровень, он же уровень системы команд. Здесь решаются задачи управления на физическом уровне в реальном масштабе времени.

4. Микропрограммный уровень. Этот уровень представляет собой описание вычислительной машины на уровне функциональных устройств и взаимодействия этих устройств между собой.
5. Микроэлектронный уровень. Это уровень физиков.

Вычислительная машина работает с программными продуктами, разработка которых происходит в 5 этапов.

Первый этап – это формализация задачи. То есть конкретизация задачи, сроков исполнения и установленных ограничений. Этот этап занимает от 10 до 30 процентов всего времени разработки продукта.

Второй этап – разработка алгоритма реализации задачи. Этот процесс занимает от 20 до 40 процентов времени и является наиболее емким этапом.

Третий этап – кодирование. Этот этап представляет собой собственно претворение алгоритма в жизнь. Например, при написании программного продукта – реализация алгоритма, разработанного на предыдущем этапе в процедурах и функциях используемого языка программирования. Этот этап – это 5-10 процентов всего времени.

Четвертым этапом разработки продукта является тестирование (отладка) реализованного алгоритма и исправление ошибок. В общей сложности этот этап занимает примерно 7,5% времени. Из них 5% - исправление синтаксических ошибок и 2,5% - исправление семантических ошибок.

Последним этапом является сопровождение созданного продукта. То есть поддержка пользователей и создание сопровождающих продуктов. Так, например, если фирма создала принтер, то на этом этапе она будет создавать картриджи для этого принтера.

2. **Принципы структурной организации ЭВМ**

2.1. **Принцип программного управления.**

Современные ЭВМ строятся на одном принципе- принципе программного управления. Принцип программного управления может быть реализован в ЭВМ многими способами. Один из способов реализации программного управления, использующийся в качестве основного принципа построения современных ЭВМ, является Неймановский принцип программного управления. Он заключается в следующем:

1. Информация кодируется в двоичной форме и разделяется на единицы (элементы) информации, наз. словами.
2. Разнотипные слова информации различаются не способом кодирования, а по способу использования.
3. Слова информации размещаются в ячейках памяти машины и идентифицируются номерами ячеек, называемыми адресами.
4. Алгоритм решения задачи представляется в форме последовательности управляющих слов, которые определяют наименование операции и слова информации, участвующие в операции. Эти управляющие слова наз. командами, а алгоритм, представленный в терминах машинных команд - программой.
5. Выполнение вычислений по решению задачи, предписанных алгоритмом сводится к последовательному выполнению команд в порядке, однозначно определяемом программой.

Первый пункт не требует комментариев.

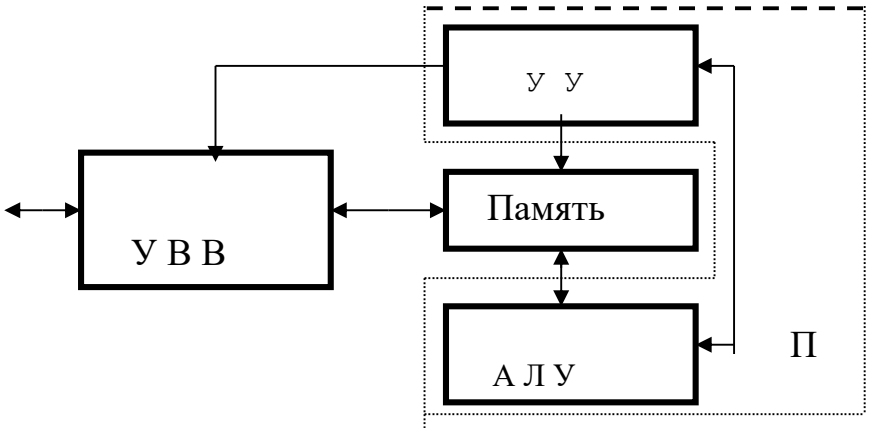
Второй пункт говорит о том, что все слова представляющие числа, команды и прочие объекты выглядят в ЭВМ совершенно одинаково. Только порядок использования слов в программе вносит различия в слова. Т.о. представляется возможность использовать одни и те же операции (у-ва) для обработки слов различной природы (чисел, команд и т.п.).

Третий пункт фиксирует специфику хранения и идентификации информации, порождаемую свойствами машинной памяти. Память выступает как совокупность ячеек, каждая из которых служит местом для хранения слова информации. Т.е. ячейка памяти выделяется для хранения значения константы или команды.

Четвертый пункт указывает на то, что алгоритм решения любой задачи представляется в ЭВМ в виде упорядоченной последовательности команд, в которых определяется код (наименование) операции и указываются адреса операндов. А требуемый порядок вычислений предопределяется алгоритмом и описывается последовательностью команд, образующих программу вычислений. Здесь важно видеть, что вычисления, производимые машиной, определяются программой и сам процесс вычислений состоит в последовательном выполнении команд программы.

2.2. **Состав и порядок функционирования ЭВМ.**

Обобщенная структурная схема ЭВМ и ее состав, соответствующие неймановскому принципу программного управления, может быть представлена в виде:



Фон Нейман впервые в 1946 году выделил 4 основные блока ЭВМ и назвал эти блоки:

- арифметически-логическим устройством (АЛУ), которое производит арифметические и логические преобразования над поступающими машинными словами, отображающими числа или другой вид информации;
- память, которая хранит данные и программу в виде машинных слов;
- управляющее устройство (УУ), которое автоматически без участия человека управляет вычислительным процессом, посылая всем другим устройствам сигналы, предписывающие им те или иные действия;
- устройство ввода-вывода, которое служит для ввода исходных данных и программы и выводит результаты вычислений.

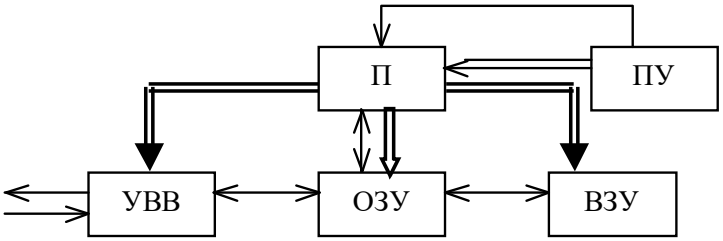
АЛУ и УУ обычно объединяются в один блок, который наз. Процессором.

Впоследствии ЭВМ такой конфигурации получила название машины неймановского типа, которая положена в основу последующих разработок ЭВМ.

Ввиду большой сложности современных машин принято представлять их структуру иерархически, т.е. понятие “элемент” жестко не фиксируется. Так, на самом верхнем уровне сама ЭВМ считается элементом более сложной вычислительной системы, состоящей из нескольких машин. На следующем уровне иерархии элементом структуры ЭВМ является память, процессор, другие операционные устройства, устройства ввода-вывода. На более низком уровне элементами являются узлы и блоки, из которых строятся память, процессор и т.д. Наконец, на самых низких уровнях элементами являются интегральные микросхемы и электро- радио-

элементы. Иерархичность структур облегчает проектирование, использование и изучение ЭВМ.

Современные ЭВМ требуют большого объема памяти и ЗУ, построенные по классической схеме, не могут быть быстродействующими. Поэтому память в современных ЭВМ строится по иерархическому принципу и по отношению к процессору имеет двухуровневую структуру. На первом уровне быстродействующая оперативная память со сравнительно небольшой емкостью, подключаемая непосредственно к процессору. На втором - более медленная внешняя память с большой емкостью. Информация, хранимая во внешней памяти недоступна процессору и может быть подвергнута обработке только после передачи в основную память. В связи с этим структурную схему ЭВМ можно преобразовать к следующему виду.



Пульт управления

Где одинарные стрелки представляют потоки информации, подлежащей обработке, а двойные показывают направления передачи управляющих сигналов.

Работа ЭВМ в соответствии с представленной структурой протекает следующим образом. Программа и исходные данные, представленные на машинном носителе информации (например магнитная лента,, магнитный или лазерный диск) считывается устройством ввода и загружается в память машины. Поскольку ОЗУ имеет ограниченную емкость, часть информации может быть загружена в ВЗУ, откуда извлекается по мере необходимости.

Для инициирования программы начальной загрузки и других вспомогательных программ выделяется отдельное устройство ввода-вывода, которое используется как пульт ручного управления (в ПЭВМ- клавиатура). С пульта посылаются директивы на загрузку и запуск программ, прекращение вычислений и т.п. С него же может вводится непосредственно в память информация, необходимая для решения задачи.

После загрузки программы и исходных данных в процессор посылается пусковой адрес программы и процессор начинает выполнять программу от команды с за-

данным адресом, что сводится к последовательной выборке команд из памяти и их выполнению средствами процессора и устройств ввода-вывода. Этот процесс заканчивается либо в момент выборки команды, отмечающей конец вычислений, либо по команде с пульта ручного управления.

3. **Элементы архитектуры ЭВМ.**

К числу элементов существенных для пользователя, подлежащих рассмотрению при функциональной организации ЭВМ, относятся наборы символов для представления данных, машинные операции, адреса и команды, в терминах которых могут быть представлены алгоритмы и процессы вычислений.

. 3.1. **Представление данных в ЭВМ.**

К основным типам данных (информации), с которыми оперирует ЦВМ можно отнести:

- числа;
- символы;
- логические значения;
- массивы (чисел, символьных строк, списочные);
- графика;
- звук;
- видео.

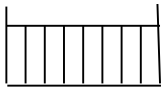
Вне зависимости от типа данных ЭВМ имеет дело исключительно с их двоичным представлением или кодированием. Т.е. в ЭВМ любые данные преобразуются в кодовые последовательности двоичных цифр.

Различные виды информации имеют свои правила кодирования. Коды отдельных значений, относящиеся к различным видам информации, могут совпадать. Поэтому расшифровка кодированных данных осуществляется по контексту при выполнении команд программы.

В ЭВМ используются три вида чисел: двоичные числа с фиксированной запятой (точкой), с плавающей запятой и двоично-кодированные (восьмеричные, шестнадцатиричные, двоично-десятичные) числа

Во всех современных компьютерах важную роль играет представление данных группами по 8 бит называемых байтами. Байт содержит одну из 2 =256 комбинаций двоичных символов.

Байт



7 0

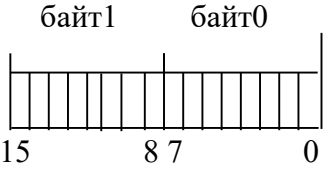
По существу, байт стал стандартной базовой единицей, из которой образуются все остальные единицы машинных данных. В зависимости от того, как интерпретируется содержимое байта, оно может быть: числом, кодированным представлением символа внешнего алфавита, частью команды или более сложной единицей данных. Другими словами интерпретацию байта определяет программист в зависимости от контекста своей программы.

В подавляющем большинстве компьютеров (ЭВМ), принята нумерация битов в байте справа налево.

Единица данных, состоящая из 16 бит или двух байт, называется словом.

(старший байт) (младший байт)

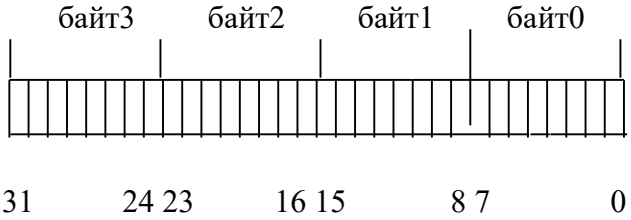
Слово



Слово может содержать любую из 2 = 65536 комбинаций. По аналогии с нумерацией бит, байты в слове также нумеруются справа налево, начиная с нуля: байт 0 называется младшим, а байт 1-старшим байтом.

Следующая единица данных состоит из 4-х байтов и называется двойным словом.

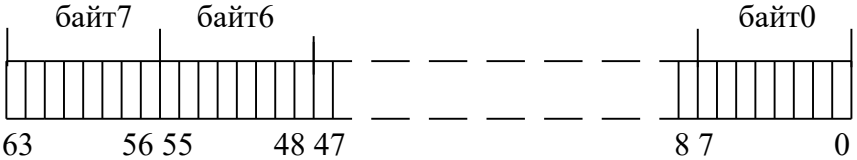
Двойное Слово



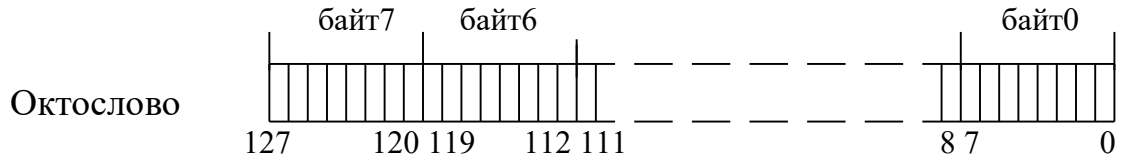
Число возможных комбинаций в двойном слове составляет 2 = 1048576.

Следующая из рассматриваемых единица данных состоит из 64 бит или 8 байт и называется квадро (длинное слово).

Длинное слово (квадро-слово)



Еще одной единицей представления данных, применяемой в современных ЭВМ, является октослово.



Октослово состоит из 16 байт.

Основная или базовая единица данных, с которой оперирует процессор (микропроцессор), называется машинным словом. Практически во всех современных процессорах длина машинного слова кратна байту. Длина слова является важнейшей характеристикой процессора и в соответствии с ней процессоры подразделяются на 8-, 16-, 32-, 64-, 128-битные.

3.2. **Машинные операции.**

Машинные операции принято подразделять на следующие классы:

- арифметические и логические операции
- посылочные операции
- переходы управления
- операции ввода – вывода
- системные операции.

Арифметические и логические операции служат для вычисления значений функций одного или нескольких аргументов. К этому классу относятся следующие операции: сложение, вычитание, умножение и деление; конъюнкция, дизъюнкция и сравнение на равенство; сдвиги влево и вправо на заданное число разрядов; преобразование чисел из одной системы счисления или формы представления в другую.

В зависимости от формата чисел выделяют операции двоичной арифметики, операции арифметики чисел с плавающей запятой и операции десятичной арифметики. В зависимости от формата слов выделяются логические операции над словами фиксированной длины и полями переменной длины.

Посылочные операции служат для передачи информации между процессором и ОП. Типичные посылочные операции - загрузить и записать. Операция загрузить обеспечивает передачу слова из ОП в П, а операция записать – из П в память.

Для повышения быстродействия ЭВМ список посылочных операций дополняется операциями передачи слова с обратным знаком, модуля слова и операциями групповой передачи.

Переходы – это операции, используемые для выполнения команд в порядке, отличном от естественного. Операция перехода обеспечивает возможность передачи управления любой команде программы. Переход может выполняться по значению условия, либо к одной, либо к другой команде. Условия, по которым реализуются операции перехода, обычно называют признаками перехода.

Для увеличения эффективного быстродействия ЭВМ используют специальные операции перехода: переход по счетчику, переход по индексу, переход с возвратом.

Операции ввода-вывода служат для передачи информации между ОП и внешними устройствами ЭВМ. Состав операций ввода-вывода определяется в основном способом подключения ВУ к ОП и П, т.е. структурой ЭВМ.

Системные операции предназначены для управления режимами работы ЭВМ. С помощью системных операций производится инициирование и прекращение выполнения программ и организуется мультипрограммная и мультимашинная обработка информации, а также работа ЭВМ в реальном масштабе времени.

3.3. **Кодирование команд**

Обработка инф. в ЭВМ осуществляется путем программного управления. Программа - алгоритм решения задачи, записанный в виде последовательности команд, которые должны быть выполнены машиной.

Команда- код определяющий операцию и данные, участвующие в операции.

По характеру выполняемых операций различают след. основные группы к-д:

А) к-ды арифметических операций для чисел с фиксированной и плавающей запятой;

Б) к-ды десятичной арифметики;

В) к-ды логических операций;

Г) к-ды передачи кодов;

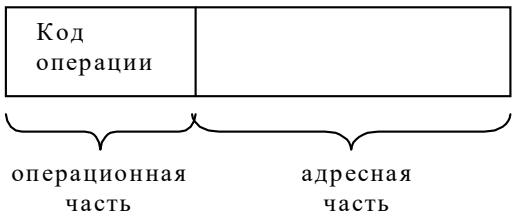
Д) к-ды операций ввода-вывода;

Е) к-ды передачи управления;

Ж) к-ды задания режимов работы машины и др.

В к-де, как правило, содержатся не сами операнды, а информация об адресах ячеек памяти или регистрах, в которых они находятся.

К-да в общем случае состоит из операционной и адресных частей



В свою очередь эти части могут состоять из нескольких полей.

Операционная часть содержит код операции КОП, который задает вид операции (слож., умнож., передача и т.д.).

Адресная часть к-ды содержит информацию об адресах операндов и рез-та операции, а в некоторых случаях инф. об адресах след. к-ды.

Структура к-ды определяется составом, назначением и расположением полей в к-де.

Форматом к-ды называют ее структуру с разметкой номеров разрядов, определяющих границы отдельных полей к-ды.

Для упрощения аппаратуры и повышения быстродействия ЭВМ длина формата к-ды должна быть согласована с длиной машинных слов (16-32 бита). Т.е. формат к-ды должен укладываться в машинное слово или полуслово, а для ЭВМ с коротким словом (8-16 бит) должен быть малократным машинному слову.

Эта проблема решается изменением структуры к-ды и введением различных способов адресации.

Классическая структура к-ды содержала в явном виде всю необходимую информацию и была 4-х адресной.

КОП	A_1	A_2	A_3	A_4

Содержала поле кода операции (КОП), два адреса операндов участвующих в операции (A_1,A_2) адрес ячейки, в которую помещался результат (A_3) и ячейки содержащей след. Команду (A_4). Такой порядок выборки команд наз. принудительным.

Если установить естественный порядок выборки к-д, при котором после к-ды, расположенной по адресу К и занимающей L ячеек выполняется к-да из К+L ячейки,

то отпадает необходимость в указании в явной форме адреса след. к-ды. В трехадресной к-де.

КОП	A_1	A_2	A_3

первый и второй адреса операндов, а третий – адрес рез-та операции.

Операция, описываемая трехразрядной к-дой может символически представлена в виде

$$ОП[A_3] := ОП[A_1] * ОП[A_2]$$

Если условиться, что рез-т операции всегда помещается на месте одного из операндов, то получаем двухадресную к-ду

КОП	A_1	A_2

Задающую операцию

$$ОП[A_1] := ОП[A_1] * ОП[A_2]$$

Т.е. для рез-та используется подразумеваемый адрес.

В одноадресной к-де

КОП	A_1

Задающей операцию

$$A_{КК} := A_{КК} * ОП[A_1],$$

подразумеваемый адрес имеет уже и рез-т и один из операторов. Здесь в качестве второго операнда используется содержимое внутреннего регистра процессора называемого аккумулятором.

При работе со стековой памятью, когда подразумеваются адреса обоих операндов и рез-та операции, возможно использование безадресных к-д

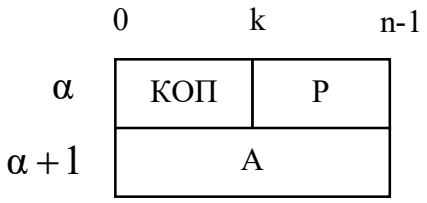
КОП

.

Степень различия затрат памяти и времени, присущих к-дам с различным числом адресов, существенно зависит от свойств алгоритмов, программируемых в терминах различных к-д.

Следует отметить, что с точки зрения затрат памяти на адресацию информации наиболее экономичны для научно-технических расчетов 1-адресные к-ды, а для задач обработки данных- 2-х адресные к-ды. По этой причине в ЭВМ общего назначения преимущественно применяется 2-х адресная система к-д, а в мини- и микро-ЭВМ- 1-адресная.

Затраты оборудования в процессоре уменьшаются, если к-да представляется в формате машинного слова. Но часто оказывается, что слова недостаточно для размещения полноразрядного адреса. В таких случаях длину адреса, указываемого в к-де, сокращают за счет страничной организации памяти. Сокращенный адрес обеспечивает доступ лишь к части информации, хранимой в фиксированной странице. Чтобы обеспечить возможность переходов между страницами необходимы команды с полноразрядными адресами. Поэтому к-ды, обеспечивающие возможность перехода между страницами, кодируются двумя словами, размещаемыми в последовательных ячейках памяти с адресами α и $\alpha+1$. Формат такой к-ды



Здесь Р- совокупность полей, в которых размещаются признаки адресации, адреса внутренней памяти и т.п.; А- полноразрядный адрес, обеспечивающий доступ к любой ячейке ОП.

При использовании к-д длиной в одно или два слова тип (признак) к-ды указывается в поле кода операции или поле Р. При нулевом значении признака к-да состоит из одного слова и следующая к-да размещается в ячейке $(\alpha+1)$, при единичном значении признака к-да состоит из двух слов и следующая к-да размещается в ячейке $(\alpha+2)$.

Использование байтов в кач-ве машинных эл-тов информации позволяет применять к-ды с различным числом адресов и различными способами адресации. За счет многообразия форматов к-д уменьшается размер программ и время их выполнения.

3.4. **Способы и типы адресации информации**

Рассмотренные выше структуры и форматы команд достаточно схематичны. В действительности адресные поля команд большей частью содержат не сами адреса,

а только информацию, позволяющую определить действительные (исполнительные) адреса операндов в соответствии с используемыми в командах способами адресации.

Следует различать понятия адресный код в команде и исполнительный адрес. Адресный код ($А_к$) – это информация об адресе операнда, содержащаяся в команде. Исполнительный адрес ($А_и$) – это номер ячейки памяти, к которой производится фактическое обращение. В современных ЭВМ адресный код, как правило, не совпадает с исполнительным адресом.

Для различных применений ЭВМ разработано большое число способов адресации. Способом адресации называется правило определения адреса и операнда на основе информации, указанной в команде.

Выбор способов адресации, формирование исполнительного адреса и преобразование адресов является одним из важнейших вопросов разработки любой ЭВМ. Эффективность способа адресации характеризуется двумя показателями: затратами оборудования и затратами времени на доступ к адресуемой информации.

Затраты оборудования определяются суммой затрат элементов аппаратуры на обработку адресов в процессоре и затратой памяти на хранение адресов, указываемых в команде программы.

Затраты времени принято характеризовать числом обращений к основной ОП, выполняемых с целью выборки или записи операнда.

Рассмотрим основные способы адресации информации, используемые в современных ЭВМ различных классов.

3.4.1. **Непосредственная адресация.**

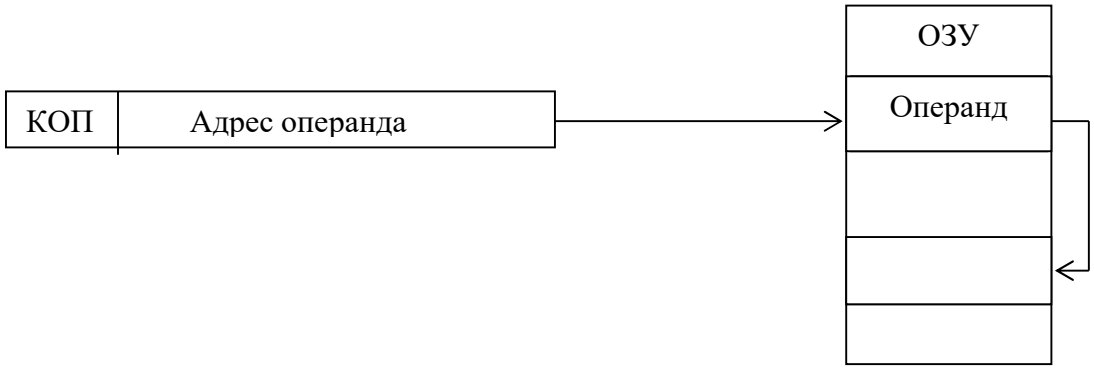
В команде содержится не адрес операнда, а непосредственно сам операнд. При непосредственной адресации не требуется обращение к памяти для выборки операнда. Это способствует уменьшению времени выполнения программы и занимаемого ею объема памяти. Эта адресация удобна для хранения различного рода констант. При значениях констант от 0 до 63 эта адресация часто называется литеральной.

3.4.2. **Прямая адресация.**

Прямая адресация – это основополагающий способ адресации в ЭВМ. В случае прямой адресации исполнительный адрес совпадает с адресной частью команды.

Прямой адрес – это номер ячейки памяти, в которой хранится операнд. Прямой адрес представляется в команде m -разрядным полем, где $m = \log_2 E$ (E – емкость адресной памяти, исчисляемая в машинных элементах – словах или байтах).

Схема прямой адресации может быть представлена в следующем виде:



В настоящее время в чистом виде этот способ находит ограниченное применение из-за трудностей: перемещения программ в памяти ; обработки массивов данных и организации специальных видов памяти; обработки данных, организованных в списочные структуры и т.п.

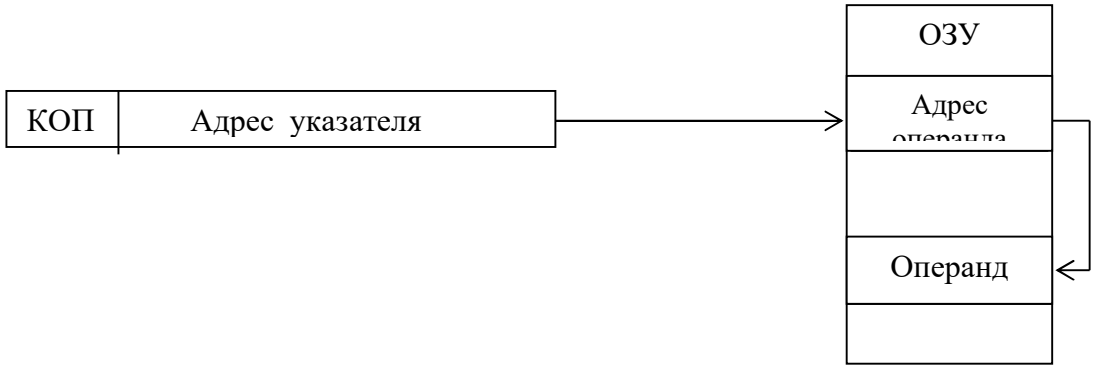
3.4.3. **Косвенная адресация.**

При программировании некоторых задач широко используются указатели, определяющие местоположение операндов в памяти. Указатель – это адрес операнда, хранимый в определенной ячейке памяти. Значение указателя может меняться в процессе выполнения программы по мере обработки одних операндов и перехода к обработке других. Но адрес самого указателя остается постоянным, обеспечивая на каждом этапе вычислений обращение к соответствующему операнду. В таком случае адрес операнда определен в ячейке с заданным адресом.

Адрес, определяющий адрес операнда, называется косвенным адресом. А адресация к операнду через цепочку указателей – косвенной адресацией. Количество указателей в цепочке – кратность косвенной адресации.

На практике кратность не превышает 6 – 8, т.к. при большей кратности программирование значительно затрудняется. Частным случаем косвенной адресации с кратностью 0 может рассматриваться прямая адресация.

Наиболее распространена однократная косвенная адресация, схема которой может быть представлена в виде:



Постоянство адреса указателя при этой адресации при изменении адреса операнда в процесе выполнения программы обеспечивает возможность переадресации данных и обработку массивов или сложной структуры данных с помощью одной и той же программы или участка программы.

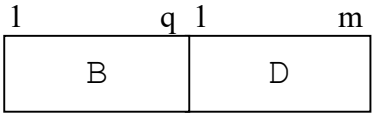
Команда с косвенной адресацией делает дополнительные обращения к памяти за косвенным адресом и выполняется дольше, чем команда с прямой адресацией. Но применение команд с косвенной адресацией выгоднее, т.к. аналогичные действия с использованием только прямой адресации требуют выполнения целой программы. Признак применения косвенной адресации указывается, как правило, в КОП команды. При большой кратности адресации в некоторых ЭВМ адресные поля команды содержат разряд – указатель косвенной адресации (УА).

3.4.4. **Относительная адресация.**

Для эффективного использования основной памяти необходимо обеспечить возможность размещения массивов данных и программ в любом месте памяти, начиная от любого адреса. За счет этого появляется возможность динамического распределения памяти, при котором для каждого массива выделяется область памяти, равная не максимальной, а фактической длине массива, что приводит к уменьшению размера области памяти, выделяемой программе. Возможность размещения информации в любой области памяти необходима для обеспечения мультипрограммного режима работы ЭВМ, при котором для программы может быть выделена любая свободная область памяти. Для обеспечения этого используется относительная адресация информации.

Основная идея относительной адресации в том, чтобы в команде указывать не абсолютный адрес операнда, а смещение его относительно начала программного мо-

дуля. Все исполнительные адреса (кроме непосредственных) при относительной адресации представляются в виде двух полей



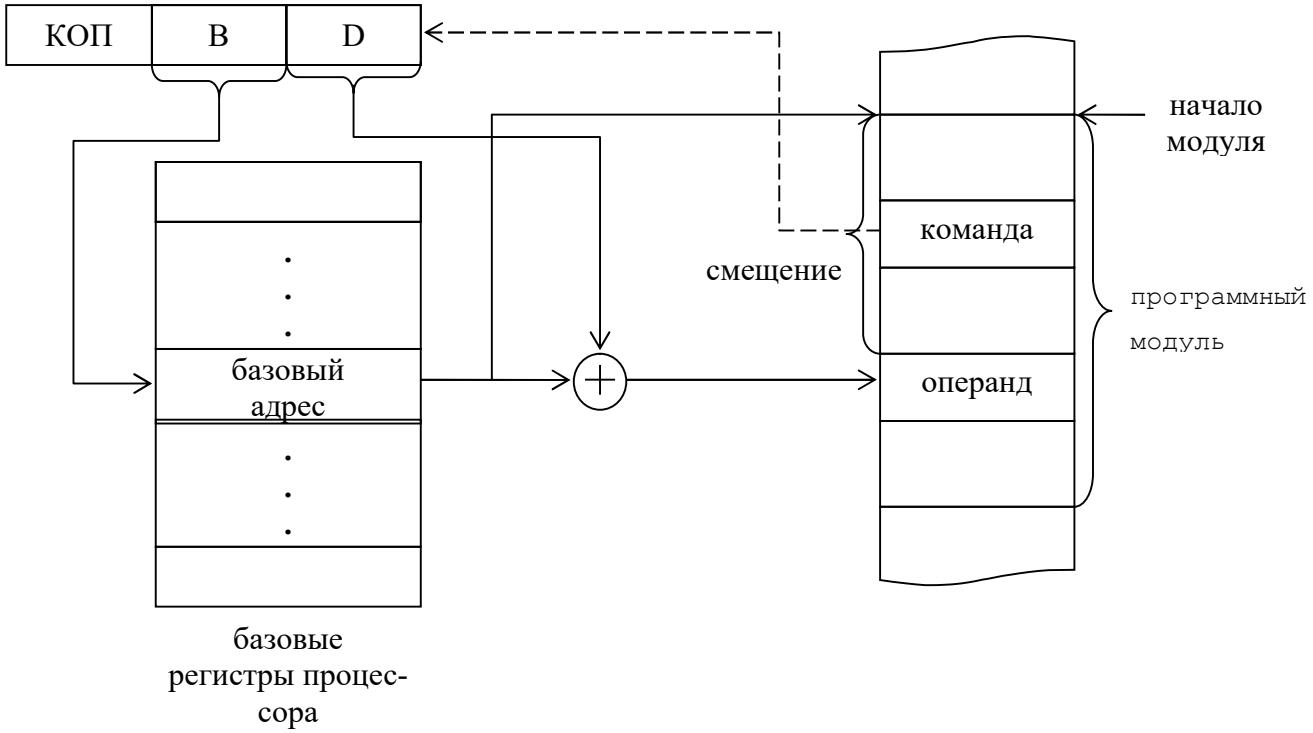
где B – адрес ячейки внутренней памяти (регистра) процессора, в которой размещается некоторое число, называемое базовым адресом и смещением D.

Исполнительный адрес операнда определяется суммой

$A_И = [B] + D$, где

[B] – содержимое ячейки внутренней памяти с адресом B.

Схему относительной адресации можно представить в виде



При использовании относительной адресации адреса операндов задаются значениями $D = 0, 1,\ldots, 2^m-1$ относительно произвольных базовых адресов, которые определяются только косвенно – адресами $B = 0, 1, \ldots, Q$ ячеек внутренней памяти, содержащих значения базовых адресов. Базовые адреса (начальные адреса программных модулей или массивов) загружаются в соответствующие ячейки внутренней памяти B при загрузке программы в основную память. При выполнении программы относительные адреса, указанные в командах, обрабатываются процессором путем выборки из внутренней памяти значения базового адреса [B] и сложения его со смещением D. При любом местоположении программы в памяти команды не из-

меняются, а изменяются лишь значения в ячейках внутренней памяти, отведенных для хранения базовых адресов.

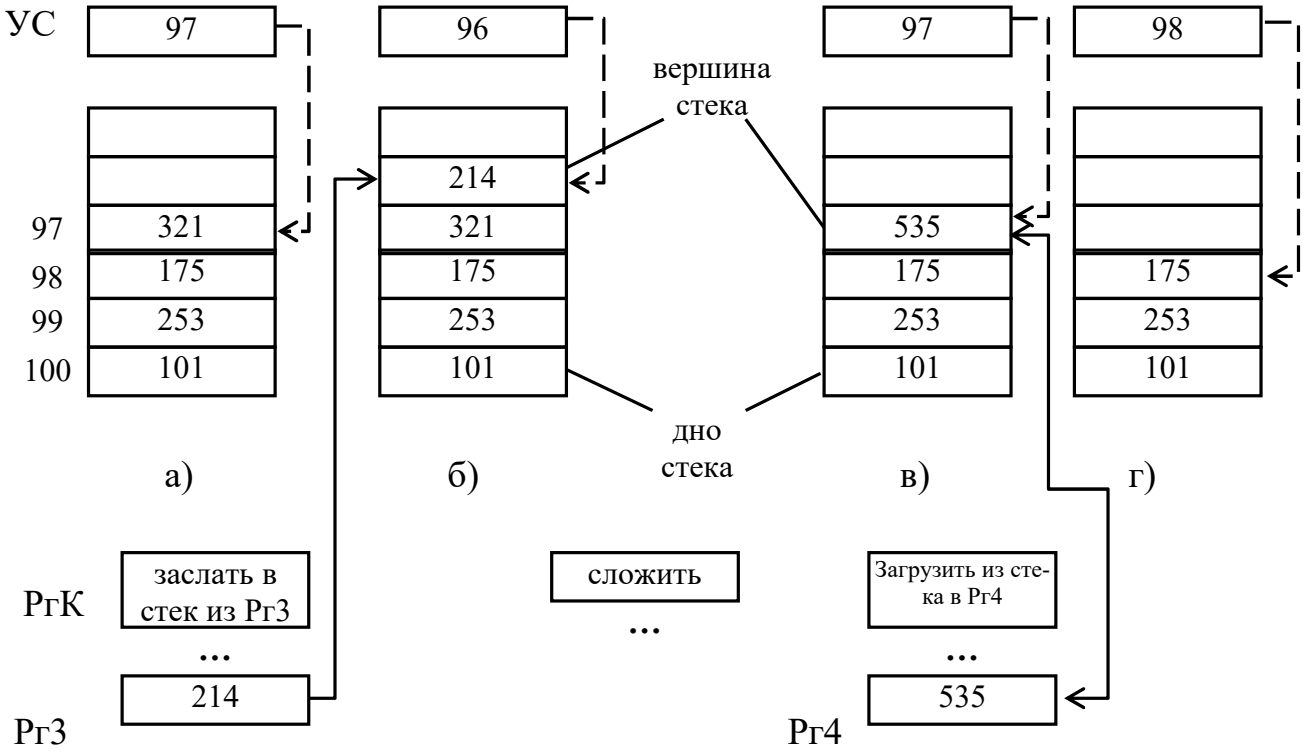
В целях сокращения емкости внутренней памяти процессора, на каждом этапе вычислений используется ограниченное число адресов – например, один базовый адрес для указания начала выполняемой программы (или модуля) и базовые адреса для адресации массивов данных, количество которых определяется структурой данных. Считается, что в ЭВМ общего назначения целесообразно применение четырех-восьми базовых адресов.

При относительной адресации в качестве базового регистра можно использовать и счетчик команд. В этом случае не требуется отдельного регистра для хранения базового адреса и не нужно осуществлять перезагрузку. Недостатком адресации относительно счетчика команд является то, что при использовании одного и того же операнда в разных командах требуется указывать разную величину смещения.

3.4.5. Стековая адресация и польская инверсная запись.

Правила “последний пришел – первый ушел” при обращении к стеку реализуется автоматически, и поэтому при операциях со стеком возможно безадресное задание операнда. Команда в этом случае не содержит адреса ячейки стека, но содержит адрес (или он подразумевается) ячейки памяти или регистра, откуда слово передается в стек или куда загружается из стека.

Механизм стековой адресации можно пояснить следующими рисунками



Здесь УС – указатель стека,

а) - исходное состояние стека

б) – стек после выполнения команды “заслать в стек из Рг3 ”

в) – стек после выполнения команды “ загрузить из стека в Рг4”.

г) – стек после выполнения команды «загрузить из стека в Рг4»

При выполнении команды передачи в стек слова из регистра или ячейки ОП сначала УК увеличивается на 1, а затем слово помещается в ячейку стека, указываемую УС. При команде загрузки из стека регистра или ячейки памяти сначала слово извлекается из вершины стека, а затем указатель стека уменьшается на 1. При соответствующем расположении операндов в стеке можно вычислять выражения полностью безадресными командами, указывающими только вид операции. Такая команда извлекает из стека в соответствии с КОП один или два операнда, выполняет над ними предписанную операцию и заносит результат в стек.

Вычисления с использованием стековой памяти удобно описывать и программировать с помощью плоской инверсной записи арифметических выражений. Эта запись производится последующему правилу

- читается арифметическое выражение слева направо и последовательно друг за другом выписываются встречающиеся операнды; как только окажется, что все операнды некоторой операции выписаны, запивается знак этой операции и далее продолжается выписывание операндов; если операция имеет операндом результат некоторой предыдущей операции и знак последней выписан, то считается этот операнд выписанным.

Для примера рассмотрим польскую запись выражения:

$$(k+l-m)(p-s)$$

Она имеет следующий вид

$$kl+m-ps-x$$

Из примера видно, что польская запись не содержит скобок, но порядок действий определен однозначно.

При использовании стековой памяти последовательность символов в польской записи может рассматриваться как программа вычислений исходного арифметического выражения, если под буквами понимать команды засылки, содержащие только адре-

са в ОП соответствующих операндов, засылаемых в стек, а под знаками операций подразумевать безадресные команды, содержащие только коды операций

адрес k
адрес l
+
адрес m
-
адрес p
адрес s
-
x

Безадресные команды на основе стековой адресации предельно сокращают форматы команд, экономят память и способствуют повышению производительности ЭВМ.

В современной архитектуре процессора (МП) стек и стековая адресация широко используются при организации переходов к подпрограммам и возврате от них, а также в системах прерывания.

3.4.6. Регистровая адресация.

Для сокращения размера адресного поля в команде и, как следствие, уменьшения числа обращений в ОП в современных ЭВМ стали широко использовать для адресации регистры внутренней памяти процессора. Это: регистровая прямая, регистровая косвенная, индексная адресации, адресация через программный счетчик.

При прямой регистровой адресации в регистре хранится операнд, а в команде указывается короткий номер регистра. При косвенной – в регистре находится адрес операнда. При индексной адресации в отличие от относительной в регистре хранится индекс, а адресное поле команды указывает адрес в оперативной памяти.

4. **Принципы организации процессоров.**

П р о ц е с с о р - совокупность аппаратных средств, развертывающих программу в виде вычислительного процесса.

Процессор в машине фон Неймана выполняет следующие функции:

1.Выбирает из памяти очередную команду.

2.Вычисляет адреса операндов, если они есть.

3.Проверяет возможность выполнить выбранную команду. Если команду выполнить невозможно, он прерывает работу по данной программе,

в противном случае

4.выбирает из памяти операнды по адресам, вычисленным в п.2.

5.Выполняет операцию, предписанную кодом операции команды.

6.Записывает результат операции.

7.Вычисляет адрес следующей команды.

8.Проверяет наличие запросов на прерывания от внешних устройств. Если таковые есть, он прерывает работу по данной программе, в противном случае

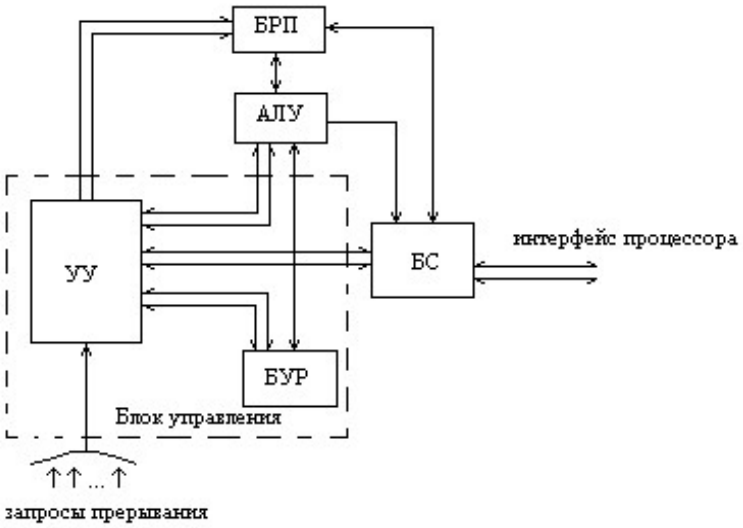
9.Переходит к выполнению п.1.

Так повторяется до тех пор, пока либо не возникнет прерывание программы, либо будет получена команда СТОП.

4.1. **Структура процессора.**

Процессор занимает центральное место в ЭВМ, т.к. кроме выполнения основных функций по расчетам, он осуществляет управление взаимодействием всех устройств, входящих в состав ЭВМ.

Упрощенная схема процессора:



где: БУР - блок управляющих регистров

БС - блок сопряжения с интерфейсом

БРП - блок регистровой памяти

В состав процессора могут входить и другие блоки, участвующие в организации вычислительного процесса: блок прерывания, блок защиты памяти, блок контроля и диагностики процессора и др.

АЛУ предназначено для выполнения арифметических и логических операций над словами данных и адресами. В АЛУ вырабатываются и сохраняются признаки результатов. В процессоре может быть либо одно универсальное АЛУ для выполнения всех основных операций, либо несколько специализированных для отдельных видов операций.

Блок управления (БУ) состоит из УУ и блока управляющих регистров (БУР). Он обеспечивает реализацию алгоритма работы процессора, выдавая последовательности управляющих сигналов в блок процессора и анализирует ответные осведомительные сигналы и запросы прерывания от узлов самого процессора и внешних устройств.

УУ вырабатывает необходимые управляющие сигналы для выборки очередной команды из памяти, дешифрации кода команды, формирования адресов операндов, выборки их из памяти и передачи их в АЛУ, выполнения в АЛУ операции, предусмотренной кодом команды, передачи результата в память, инициирования операций ввода/вывода, организация реакции процессора на запросы прерываний.

БУР предназначен для временного хранения управляющей информации. В нем содержатся регистры, хранящие информацию о состоянии процессора, счетчик команд, счетчик тактов, регистр запросов прерываний и др.

Блок регистров памяти (БРП) включен в состав процессора для повышения его быстродействия и логических возможностей. Регистры этого блока служат для хранения операндов, в качестве аккумулятора, базовых и индексных регистров,

Все регистры могут быть разделены на программно - доступные и программно-недоступные. Программно-доступные - регистры, содержимое которых доступно командам процессора. К ним относят: аккумулятор, базовый и индексный регистры, множителя, частного, счетчик команд, указатель стека и др. В процессе выполнения программы выполняются логические и арифметические операции. В результате вырабатываются некоторые признаки, например, перенос из старшего разряда, переполнение разрядной сетки, равенство результата нулю и т.д. Значения этих признаков фиксируются в регистре слова состояния. Кроме того в нем фикси-

руется значение приоритета, закрепленного за обрабатываемой программой. Регистры могут использоваться для вычисления адресов, тогда их называют адресными регистрами. Если регистры используются только для обработки данных, их называют регистрами данных.

Регистры общего назначения (РОНы) используются и для того, и для другого. Но они обычно не закреплены жестко по назначению, за исключением счетчика команд и указателя стека.. Количество РОНов 80 ÷ 32.

Программно- недоступные регистры предназначены для хранения разнообразной информации в процессе выполнения одной команды. К ним относятся: регистры команд, буферные регистры адресов, слов и некоторые другие.

БС обеспечивает захват шин интерфейса и выработку всех необходимых сигналов для выполнения обмена по шинам.

4.2. **Принципы организации систем прерывания программ.**

Во время выполнения текущей программы, как внутри ЭВМ, так и во внешней среде могут возникать события, требующие немедленной реакции на них со стороны ЭВМ. Эта реакция заключается в прерывании текущей программы и в переходе к другим программам, специально предназначенным для данного события. По завершению выполнения этой программы ЭВМ возвращается к прерванной. Рассмотренный процесс называется прерыванием программы. В этом процессе принципиально то, что моменты возникновения событий неизвестны, и не могут быть учтены при программировании. Сигналы, сопровождающие эти события, называются запросами прерывания. Программа, вызываемая запросом, называется прерывающей.

Возможность прерывания программ - важное архитектурное свойство ЭВМ. Оно позволяет эффективно использовать производительность процессора при наличии нескольких, протекающих параллельно во времени процессов, требующих в произвольные моменты времени управления и обслуживания со стороны процессора.

Для реализации высокого быстродействия прерываний, ЭВМ необходимы аппаратные и программные средства, совокупность которых называется системами прерывания программ или контроллерами прерываний.

Основными функциями системы прерываний являются:

- запоминание состояний прерываемой программы и осуществления перехода к прерывающей программе;
- восстановление состояний прерванной программы и возврат к ней.

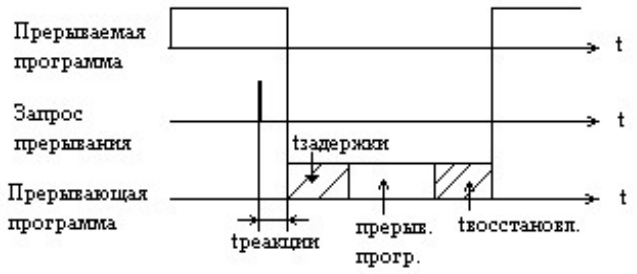
Как правило, существует не один источник запроса прерываний, поэтому между запросами (прерывающими программами) устанавливаются приоритетные соотношения, определяющие какой из нескольких поступивших запросов обслуживается первым и устанавливающие имеет ли данный запрос прерывать текущую программу.

Для оценки эффективности рассматриваемых систем используются следующие характеристики:

а) общее число запросов прерывания (входов в систему прерывания);

б) время реакции - время между появлением запросапрерывания и началом выполнения прерывающей программы.

Приведем упрощенную временную диаграмму процесса прерывания:



где: $t_р$ – время реакции,

$t_з$ – время запоминания состояния прерванной программы;

$t_в$ – время восстановления состояния прерванной программы.

Для одного и того же запроса прерывания задержки в исполнении прерывающей программы зависят от числа программ со старшим приоритетом, ждущих обслуживания. Поэтому время реакции определяют для запроса с наивысшим приоритетом. Затраты времени переключения программ (издержки прерывания) равны

$$t_{изд} = t_з + t_в.$$

в) Глубина прерывания - max число программ, которые могут прерывать друг друга. Глубина равна m, если допускаемое прерывание до m программ.

4.3. **Организация перехода к прерывающей программе.**

Для каждой причины прерывания имеется своя ячейка памяти (или две), содержащая так называемый вектор прерывания, состоящий из начального адреса прерывающей программы и соответствующего ей слова состояния.

. Каждому внутреннему прерывающему состоянию процессора и каждому ВУ соответствует свой вектор прерывания (ВП), способный инициализировать выполнение соответствующей прерывающей программы. Главное место в процедуре перехода к прерывающей программе занимает передача из соответствующего регистра процессора в память на сохранение текущего вектора состояния прерываемой программы и загрузка в регистр вектора прерывания прерывающей программы, к которой и переходит управление процессором. Эта процедура включает в себя и выделение из выставленных запросов такого, который имеет ***наибольший*** приоритет. Различают абсолютный и относительный приоритеты. Запрос с абсолютным приоритетом прерывает выполняемую программу. Запросы с относительным - является первым кандидатом на обслуживание после завершения текущей программы.

Простейший способ установления приоритетных отношений между запросами прерываний заключается в порядке присоединения линий сигналов запросов ко входам системы прерывания. В этом случае приоритет является жестко фиксированным. При этом используются различные процедуры установления приоритетных отношений между запросами и перехода к прерывающей программе. Среди них: прерывание с опросом источников (флажков) прерывания; цикличный (многотактный) опрос запросов прерывания; цепочный (однотактный) опрос; векторное прерывание.

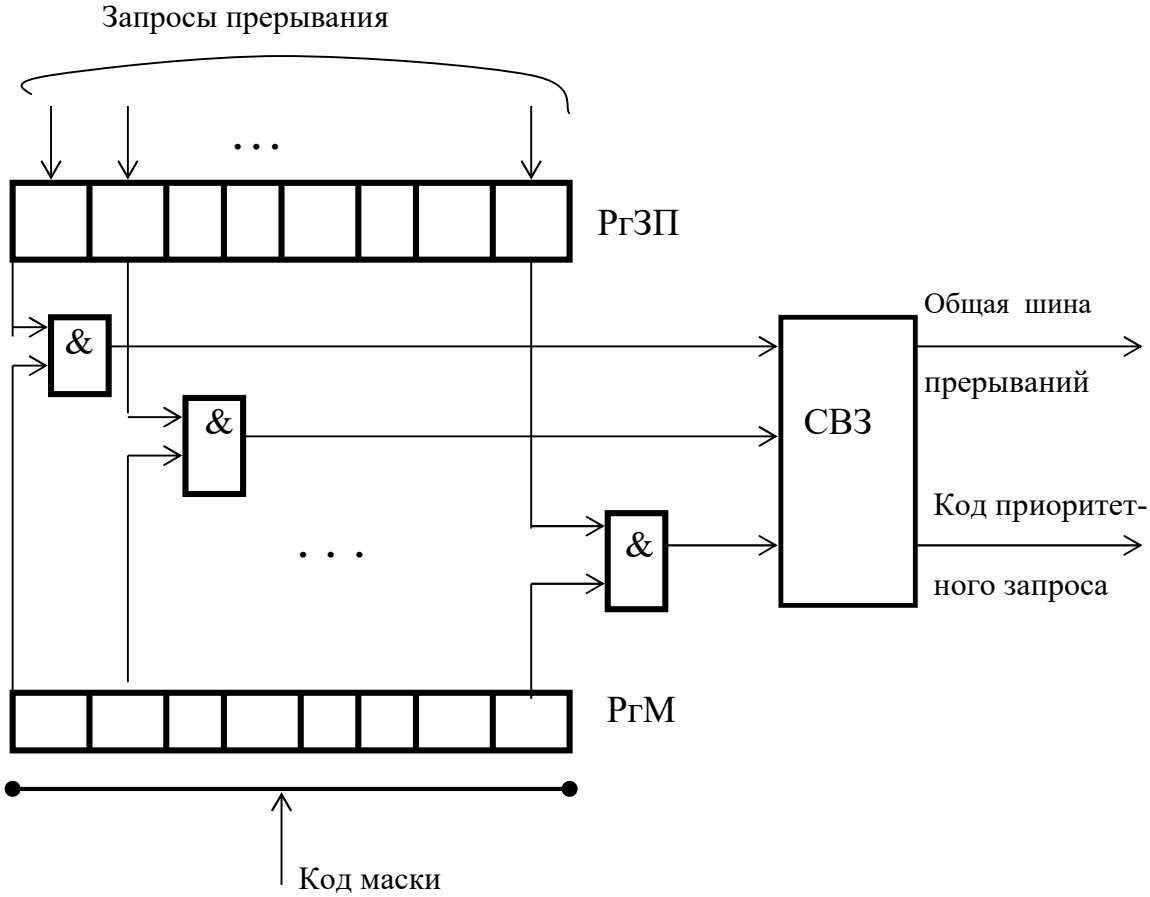
Во многих случаях приоритет между прерывающими программами не может быть зафиксирован раз и навсегда. Т.е. приоритет должен быть динамичным, программно управляемым.

Широко применяют два способа реализации программно- управляемого приоритета прерывающих программ, в которых используется соответственно порог прерывания (в малых- и микро- ЭВМ) и маски прерывания (в ЭВМ общего назначения).

Первый способ позволяет в ходе вычислительного процесса программным путем изменять уровень приоритета обрабатываемой процессором программы относительно приоритетов запросов источников прерывания. Другими словами задается минимальный уровень приоритета запросов, которым разрешается прерывать программу, идущую на процессоре, т.е. задавать порог прерывания. Порог прерывания

задается командой программы, устанавливающей в регистре порога прерывания код порога прерывания.

Второй способ реализации программно-управляемого приоритета проиллюстрируем схемой.



. где РгЗП - регистр запросов прерывания;

РгМ - регистр маски;

СВЗ - схема выделения немаскированного запроса старшего приоритета.

Программное управление приоритетом на основе маски прерывания получило наибольшее распространение в современных ЭВМ общего назначения. Маска прерывания - двоичный код, разряды которого поставлены в соответствие запросам прерывания. Маска загружается командой программы в РгМ, состояние 1 в данном разряде маски разрешает, а 0 - маскирует прерывание текущей программы от соответствующего запроса. Таким образом программа, изменяя маску, может устанавливать произвольные приоритетные соотношения между запросами без перекоммутации линий. При формировании маски в 1 устанавливается только в разряды, соответствующие прерывающим программам с более высоким, чем у выполняемой про-

граммы приоритетом. Схемы «И» выделяют незамаскированные запросы, из которых СВЗ выделяют наибольший приоритетный и формируют код его номера.

С замаскированным запросом, в зависимости от причины прерывания поступают двояким образом: или он игнорируется, или запоминается с тем, чтобы осуществить прерывание, когда запрет будет снят.

4.4. **CISC , RISC и MISC архитектуры**

Процессор с большим числом различных по формату и длине команд, сложной системой кодирования операций, с большим числом различных режимов адресации называют процессор с CISC-архитектурой. CISC (complex instruction set computer). CISC-архитектура используется с1964 года и дошла до наших дней, например, в таких современных мейнфреймах как IBM ES/9000.

Эта архитектура является практическим стандартом для рынка микрокомпьютеров серией x86 и Pentium.

Для CISC-процессоров характерно: сравнительно небольшое число регистров общего назначения; большое количество машинных команд (до 250), некоторые из которых выполняются за много тактов; большое количество методов адресации; большое количество форматов команд различной разрядности; преобладание двухадресного формата команд; наличие команд обработки типа регистр - память.

Основное преимущество CISC архитектуры: облегчение отладки программ на ассемблере.

Иногда же достаточно иметь небольшое число основных команд, одинакового формата с простой кодировкой. Такие ЭВМ называют ЭВМ с RISC - архитектурой (reduced instruction set computer).

Основой архитектуры современных рабочих станций и серверов является именно эта архитектура. Упрощения архитектуры С. Крэй с успехом применил при создании широко известной серии суперкомпьютеров компании Cray Research.

Система команд разрабатывалась таким образом, чтобы выполнение любой команды занимало небольшое количество машинных тактов (предпочтительно один машинный такт). Сама логика выполнения команд с целью повышения производительности ориентировалась на аппаратную, а не на микропрограммную реализацию. Чтобы упростить логику декодирования команд использовались команды фиксированной

длины и фиксированного формата. Для обработки, как правило, используются трехадресные команды.

Благодаря небольшому числу команд упрощаются аппаратные средства ЭВМ типа RISC, время выполнения команд примерно одинаково, увеличивается скорость выполнения команд. Однако отладка программ в RISC архитектурах более сложна.

Повышение степени интеграции при производстве процессоров создало возможность их реализации, обладающей преимуществами вышеуказанных архитектур. Это так называемая MISC-архитектура (multipurpose instruction set computer).

Элементная база состоит из двух частей, которые выполнены в отдельных корпусах либо объединены.Основная часть (host)- RISC процессор, расширяемый подключением второй части- ПЗУ микропрограммного управления. Основные команды работают на host, а для реализации сложных операций составляются микропрограммы (программы) на командах RISC -архитектуры, которые записываются в ПЗУ. т.о. вторая часть становится эквивалентна процессору со сложным набором команд. И система приобретает свойства CISC.

Практика показывает, что использование MISC - архитектуры и быстродействующего ядра, реализованного на RISC -архитектуре, позволяет получить большее быстродействие, чем при прямой реализации CISC -архитектуры. При этом сама реализация по MISC -архитектуре значительно технологичнее.

Процессоры Intel с 8086 по 80486 и Pentium реализованы на CISC -архитектуре, а Pentium II и выше - на MISC -архитектуре.

4.5. **Совмещение операций. Конвейеризация.**

При проектировании процессоров используется совмещение операций, позволяющее в каждый момент времени выполнять более одной операции.

Это совмещение производится как при обработке команд в УЦУ, так и при выполнении операций в АЛУ и может осуществляться на уровне команд, микрокоманд и схем.

Совмещение операций базируется на двух принципах:

А) Параллелизм. Делается несколько одинаковых устройств и все они работают одновременно, решая одну и ту же задачу или ее части.

Б) Конвейеризация. Подлежащая вычислению функция разбивается на подфункции, которые можно выполнять последовательно. За каждой из этих подфункций закрепляется некоторый набор оборудования, позволяющий выполнить эту подфункцию. Подфункция, вместе с закрепленным за ней набором оборудования, называется ступенью конвейера. Объект, двигаясь по конвейеру, в последующий момент времени занимает ступень со следующим номером. Объект, поступивший на конвейер раньше, занимает ступень с большим номером, чем объект, поступивший позднее.

Т.е. для выполнения каждого этапа необходимо иметь свой аппаратный блок в конвейере операций. Общая схема конвейера:

Здесь, Фi - фиксаторы,

Сті - аппаратные средства i -ой ступени.

Обычно в качестве фиксаторов используют один или несколько регистров, а аппаратные средства ступеней - комбинационные схемы.

Результатом работы каждой ступени конвейера, которая является исходным данным для следующей ступени, фиксируется в фиксаторах и регистрах. Для того, чтобы можно было реализовать такой конвейер, необходимо выполнить 4 условия:

1) Вычисления функции должны быть относительно независимы.
2) Для вычисления каждой функции требуется один и тот же набор подфункций.
3) Все подфункции тесно связаны между собой.
4) Для вычисления каждой подфункции требуется одно и то же время.

Конвейер, удовлетворяющий этим условиям, называется синхронным или конвейером со статической структурой.

4.5.1. **Синхронный конвейер операций.**

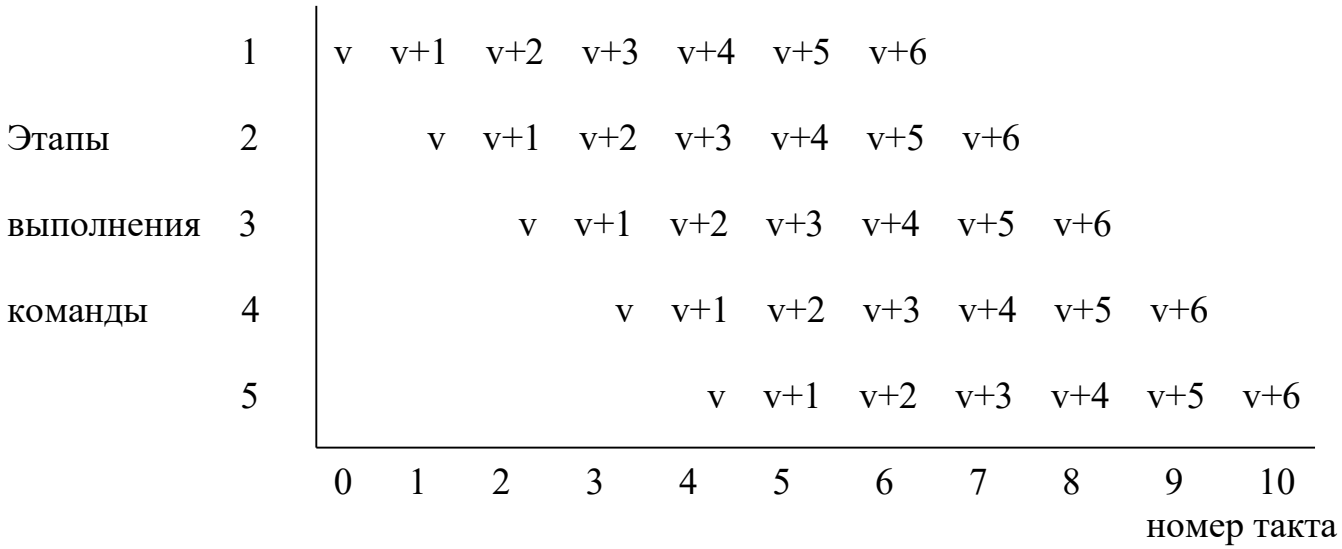
Если конвейер работает в принудительном и постоянном темпе, то его называют синхронным. Разбиение процедуры на этапы и выбор длительности этапа или такта конвейера производится, согласно условиям:

$$t_т. = \max \{ t_i \}, i = 1, 2, ..., k \qquad (1)$$

$t_i + t_{i+1} > t$ т., $i = 1, 2, ..., k$ (2)

При этом в неравенстве (2) можем принять $t_{k+1} = t_1$, т.к. работа конвейера циклична. Если для каких-либо смежных этапов условие (2) не выполняется, то их объединяют в один этап, либо наиболее длительный этап разбивают на несколько. В последнем случае заново выбирается $t_т$. И вновь проверяется условие.

Временная диаграмма выполнения команды в синхронном ступенчатом конвейере:

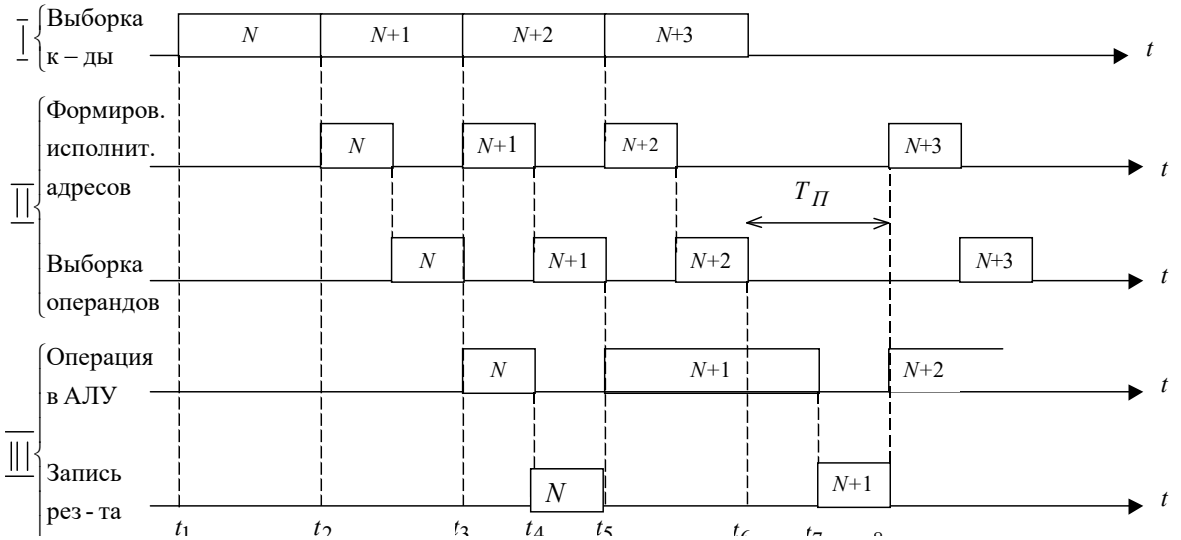


На рисунке одинаковыми символами помечены разные этапы работы цикла одной и той же команды. После того, как все позиции конвейера заполняются, параллельно во времени выполняется столько команд, сколько ступеней в конвейере. Скорость вычислений теоретически увеличивается в n раз. В действительности рост реальной производительности процессора ниже из-за простоев отдельных ступеней конвеера. По схеме синхронного конвейера реализуется операционное устройство в целом.

4.5.2. **Асинхронный конвейер операций.**

При большой зависимости действительных продолжительностей выполнения процедур отдельных этапов от типа команды и вида операндов целесообразно применение асинхронного конвейера. В нем отсутствует единый такт работы ступеней, а информация с одной ступени на другую передается при наличии двух условий:

- данная ступень закончила свою процедуру;
- следующая ступень полностью освободилась от обработки предыдущей команды.



В качестве примера приведем временную диаграмму совмещения выполнения четырех команд в трехступенчатом конвейере

Здесь N – команда.

Из диаграммы видно, что, начиная с момента t_3 выполняются одновременно три этапа цикла. А для данного примера момента t_6 из-за большой длительности в команде N+1 операции в АЛУ приостанавливается работа аппаратуры ступеней 1 и 2.

Если в синхронных конвейерах аппаратные средства каждой ступени, как правило, не имеют собственных элементов памяти, т.е. являются комбинационными схемами; то в асинхронных наличие памяти для запоминания состояния ступени обязательно.

По схеме асинхронного конвейера реализуется выполнение команд в центральном процессоре, процессоре подпрограмм и т.п.

Можно сказать, что синхронные конвейеры используются на более низком, аппаратном, микропрограммном уровне, а асинхронные на более высоком, командном, программном уровне.

5. **Организация машинной памяти**

5.1. **Определения и основные характеристики памяти.**

Памятью ЭВМ называется комплекс технических средств, служащих для запоминания, хранения и выдачи информации. Отдельные средства, входящие в этот

комплекс, называют запоминающими устройствами или памятями того или иного типа.

Процесс фиксации сигналов в ЗУ называется записью информации, а процесс выдачи сигналов - чтением или считыванием информации. Процессы записи или чтения информации называются процессами обращения к ЗУ.

Элементы памяти - элементы, каждый из которых способен запоминать один разряд слова.

Ячейка памяти - совокупность запоминающих элементов, обращение к которым при записи и чтении информации производится одновременно.

Совокупность бит информации, хранимых в одной ячейке, называется словом памяти. Слово памяти может не совпадать с машинным словом, являющимся основной информационной единицей.

К основным характеристикам ЗУ относятся быстродействие, емкость, надежность, стоимость.

Быстродействие ЗУ определяется периодом обращения Тобр и временем выборки Твыб.

Период обращения - минимально допустимый интервал времени между двумя очередными обращениями к ЗУ. Ряд ЗУ имеет различные периоды обращения при записи и при чтении информации.

Время выборки - интервал времени от начала обращения к ЗУ при чтении до момента появления информации на выходе ЗУ.

Емкость ЗУ ("объем памяти") задается количеством бит или байт информации, которое может одновременно хранится в ЗУ. Часто емкость ЗУ определяют числом хранимых одновременно слов определенной разрядности.

Удельная емкость - отношение емкости ЗУ к его физическому объему.

Надежность ЗУ - свойство ЗУ выполнять функции фиксации хранения и выдачи информации. Надежность оценивается вероятностью безотказной работы ЗУ в пределах заданного интервала времени.

Удельная стоимость ЗУ - отношение стоимости всего ЗУ к емкости ЗУ в битах.

По способу обращения к ячейкам памяти ЗУ делят на 3 класса:

1) ЗУ с последовательным доступом к информации (ЗУ на магнитных лентах); в таких ЗУ при обращении к какой-либо ячейке требуется прохождение через другие ячейки.
2) ЗУ с периодическим (циклическим) доступом; к ним относятся накопители на магнитных барабанах, магнитных и лазерных дисках. В таких ЗУ используются вращающиеся носители информации и ячейки ЗУ оказываются доступными только через периодически повторяющиеся интервалы.
3) ЗУ с произвольным (непосредственным) доступом; примером таких ЗУ являются ЗУ на ферритовых сердечниках или полупроводниковые ЗУ; для таких ЗУ характерна независимость времени доступа к ячейке от ее размещения в ЗУ.

Под временем доступа понимается промежуток времени между началом обращения и моментом, когда становится возможным доступ к данной ячейке памяти.

В зависимости от реализуемых в памяти обращений различают:

а) память с произвольным обращением; здесь возможны запись и считывание данных;

б) постоянная или односторонняя память; здесь возможно только считывание информации, а запись производится либо в процессе изготовления, либо при настройке

Эти типы памяти соответствуют терминам:

RAM(random-access memory - память с произвольным обращением)

ROM(read only memory - память только для считывания)

По кратности считывания различают ЗУ со считыванием без разрушения информации и ЗУ со считыванием с разрушением информации. В последнем случае для сохранения информации необходимо регенерировать считанную информацию в каждом цикле обращения к ЗУ, чтобы иметь возможность ее последующего использования.

По характеру хранения информации ЗУ подразделяются на статические и динамические. В статических ЗУ физическое состояние, кодирующее информацию, остается неподвижным относительно носителя информации. Элементы памяти (ЭП) статических ЗУ способны сохранять свое состояние (0 или 1) неограниченно долго при включенном питании. В динамических ЗУ кодирующее информацию физическое состояние периодически перемещается по отношению к среде носителя информации. Динамические ЗУ нуждаются в периодическом восстановлении записанной в

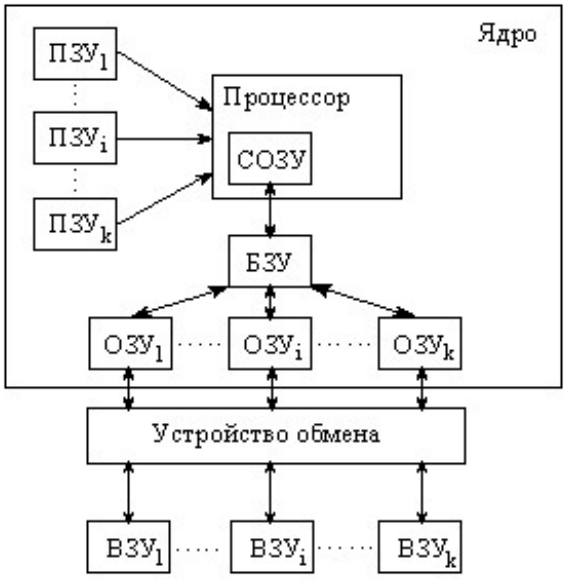
них информации – в регенерации. ЭП динамических ЗУ отличаются от ЭП статических меньшим числом компонентов в одном ЭП и поэтому могут иметь меньшие габариты. Однако из-за необходимости регенерации информации динамические ЗУ имеют более сложные системы управления.

По назначению различают ЗУ сверхоперативные (СОЗУ), буферные (БЗУ), оперативные (ОЗУ), постоянные (ПЗУ) и внешние (ВЗУ). Такое деление связано с особенностями использования и специфичностью характеристики отдельных типов иерархии ЗУ.

5.2. **Иерархия памяти ЭВМ**

Информационная емкость и быстродействия - противоречивые характеристики ЗУ. Чем больше быстродействие, тем технически труднее достигается и дороже обходится увеличение емкости памяти. А стоимость памяти составляет значительную часть общей стоимости ВМ. Поэтому память ЭВМ организуется в виде иерархической структуры запоминающих устройств. В иерархии памяти используют ЗУ с различными характеристиками, начиная со сверхскоростных сверхоперативных или буферных устройств со сравнительно малой информационной емкостью и кончая сравнительно медленными внешними ЗУ с очень большой емкостью.

Приведем обобщенную схему иерархии ЗУ ЭВМ.



В конкретных условиях реализации ЦВМ набор ступеней иерархии, а также количество блоков ОЗУ, ВЗУ, ПЗУ может быть различными.

Оперативные запоминающие устройства (ОЗУ) имеют информационную емкость достаточную для выполнения программы или их частей, но их быстродействие в не-

сколько раз ниже быстродействия СОЗУ. Разделение ОЗУ на ряд модулей позволяет увеличить его быстродействие за счет совмещения различных фаз временных циклов при параллельном обращении к различным модулям. Выполняются на тонких магнитных пленках, а в последнее время на БИС.

Сверхоперативные ЗУ (СОЗУ) служат для хранения ряда чисел, необходимых для выполнения некоторой текущей последовательности команд программы. Они строятся на интегральных микросхемах. Быстродействие СОЗУ соизмеримо с быстродействием АЛУ и блоков устройства управления процессора (в 2-10 раз превышает быстродействие ОЗУ. СОЗУ (встроенный КЭШ, cache - тайник) используется как для временного хранения участков программы, так и данных, участвующих в вычислениях.

Буферные ЗУ (БЗУ) - предназначены для промежуточного хранения информации при ее обмене между устройствами, работающими с разной скоростью. Их быстродействия и емкость соизмеримы с аналогичными характеристиками СОЗУ. Регистры этой памяти недоступны для программиста, поэтому буферную память часто называют внешней КЭШ-памятью.

В качестве ОЗУ, СОЗУ и БЗУ используются быстродействующие ЗУ с произвольным обращением и непосредственным доступом.

Достаточно большое количество информации, требуемое для работы ЦВМ, не изменяется в процессе вычислений, поэтому технически целесообразно построить ЗУ, в которое информация записывается только при изготовлении или настройке, а при работе только считывается. Такие устройства называются ПЗУ. Они предназначены для хранения некоторых программ (начальной загрузки), отдельных микропрограмм, различных констант, таблиц функций и т.п. Они, как правило имеют большее быстродействие и меньшую аппаратную сложность, чем ОЗУ.

Внешние ЗУ(ВЗУ) - предназначены для хранения больших массивов информации и работают со сравнительно малой скоростью. В качестве ВЗУ используются либо ЗУ с прямым доступом на магнитных барабанах, магнитных или оптических дисках, либо ЗУ с последовательным доступом на магнитных лентах. ВЗУ на магнитных барабанах, магнитных и лазерных дисках относятся к быстродействующими, а на магнитных лентах к медленнодействующим.

При иерархическом принципе построения структуры ЗУ, логическая организация потоков информации должна быть такой, чтобы все информационное поле ЦВМ выступало в виде единого внутреннего ЗУ. Это абстрактное внутреннее ЗУ называют виртуальные ЗУ.

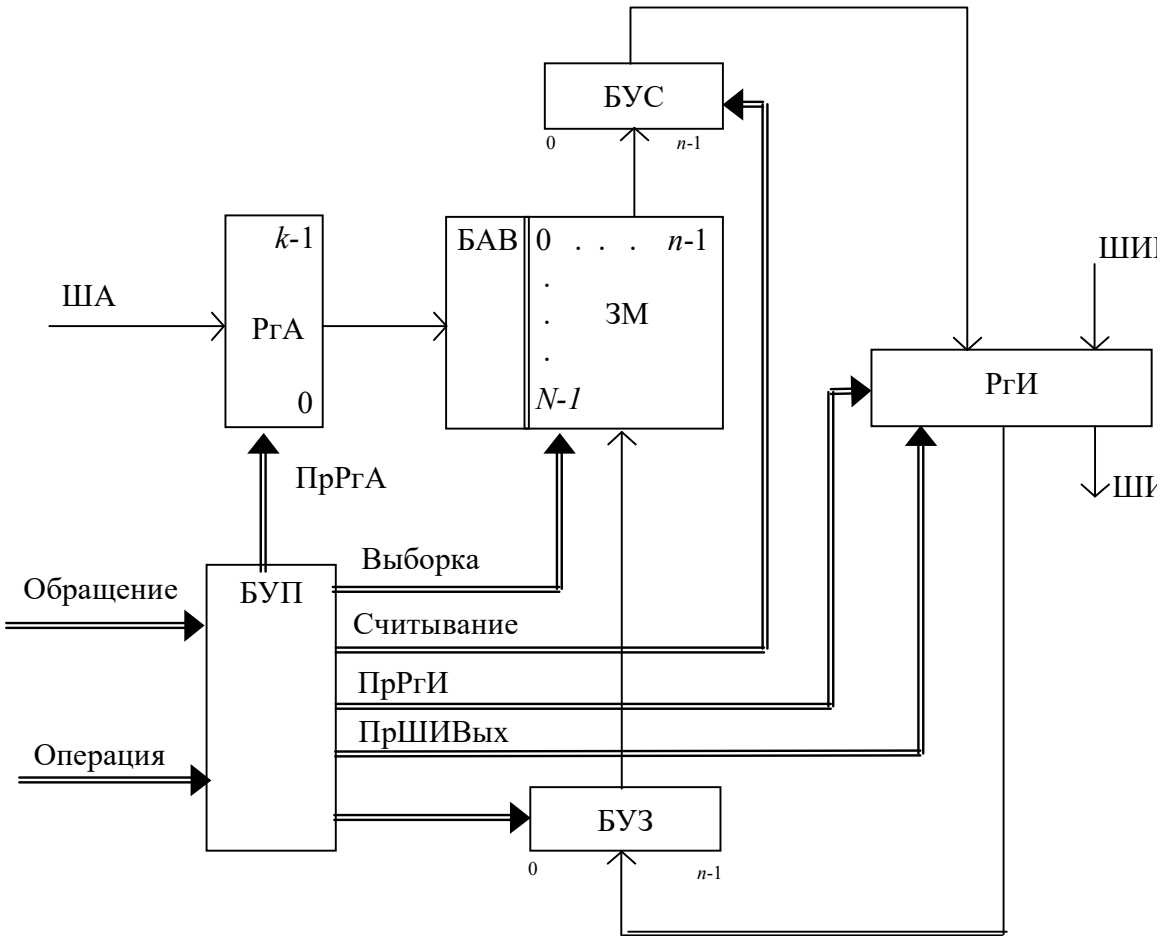
5.3. **Адресная, ассоциативная и стековая организация памяти**

ЗУ, как правило, содержит множество одинаковых запоминающих элементов, образующих запоминающий массив. Массив разделен на ячейки памяти, каждая из которых предназначена для хранения двоичного кода, число разрядов в котором определяется шириной выборки памяти (одно, половина или несколько машинных слов). Способ организации памяти зависит от методов размещения и поиска информации в запоминающем массиве. По этому признаку различают адресную, ассоциативную и стековую (магазинную) памяти.

5.3.1. **Адресная память**

В памяти с адресной организацией размещение и поиск информации в массиве основаны на использовании адреса хранения слова (числа, команды и т.п.); адресом служит номер ячейки запоминающего массива, в котором это слово размещается .

При записи (или считывании) слова в ЗМ инициирующая эту операцию команда должна указывать адрес (номер ячейки), по которому производится запись (считывание). Приведем типичную структуру адресной памяти.



Эта структура содержит ЗМ из N n - разрядных ячеек; регистр адреса РгА, имеющий k (k >= log2N) разрядов; информационный регистр РгИ; блок адресной выборки БАВ; блок усилителей считывания БУС, блок разрядных усилителей формирователей сигналов записи БУЗ и блок управления памяти БУП.

По коду адреса с РгА БАВ формирует в соответствующей ячейке памяти сигналы, позволяющие произвести в ячейке считывание или запись слова.

Цикл обращения к памяти инициируется поступлением в БУП сигнала Обращение (извне). Общая часть цикла обращения включает в себя прием в РгА с ША адреса обращения и прием в БУП и расшифровку управляющего сигнала "Операция", указывающего вид запрашиваемой операции (считывания или запись).

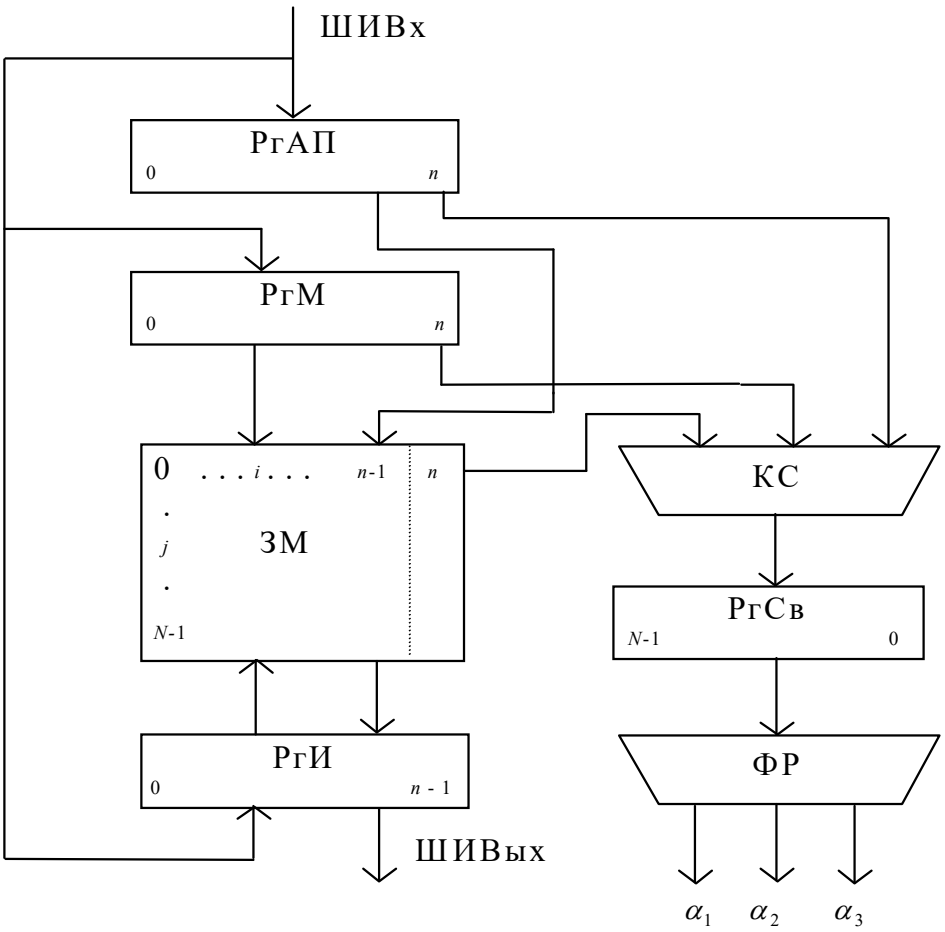
Далее при считывании БАВ дешифрирует адрес и посылает сигнал считывания в заданную адресом ячейку ЗМ. При этом слово записанное в ячейке считывается БУС и передается в РгИ. В памяти с разрушающем считыванием производится регенерация информации в ячейке путем записи в нее из РгИ через БУЗ считанного слова. Операция считывания завершается выдачей слова из РгИ на информационную выходную шину ШИВых.

При записи помимо выполнения указанной выше общей части цикла обращения производится прием записываемого слова с входной информационной шины ШИВх в РгИ. Сама запись состоит из двух операций: очистки требуемой ячейки (сброса в 0) и собственно записи. Для этого БАВ сначала производит выборку и очистку ячейки заданной адресом в РгА. Очистка производится сигналами считывания слова в ячейке. Но в это время БУС блокированы и информация из БУС в РгИ не поступает. Затем производится запись слова из РгИ в очищенную ячейку.

БУП генерирует необходимые последовательности управляющих сигналов (показаны двойной стрелкой), инициирующих работу отдельных узлов памяти.

5.3.2. **Ассоциативная память**

В памяти этого типа поиск нужной информации производится не по адресу, а по ее содержанию (по ассоциативному признаку). Под поиском по ассоциативному признаку понимается поиск числа в памяти по наибольшему значению; по наименьшему значению; по значению, заданному в определенных пределах; большего или меньшего, чем заданное; ближайшего меньшего или ближайшего большего и т.п. Поиск по ассоциативному признаку производится параллельно во времени для всех ячеек ЗМ. Приведем типичную структуру ассоциативной памяти



ЗМ содержит N (n+1)-разрядных ячеек. Для указания занятости ячейки исполь-

зуется служебный n-й разряд.

На схеме: РгАП - регистр ассоциативного признака; РгМ - регистр маски; РгСв - регистр совпадения; ФР - комбинационная схема формирования результата ассоциативной выборки; КС – комбинационная схема.

По входной информационной шине ШИВх в регистр ассоциативного признака РгАП в разряды $0 \div n-1$ поступает n-разрядный ассоциативный запрос, а в регистр маски РгМ - код маски поиска, при этом N-й разряд РгМ устанавливается в 0. Ассоциативный поиск производится лишь для совокупности разрядов РгАП, которым соответствуют 1 в РгМ (незамаскированные разряды РгАП). Для слов, в которых цифры в разрядах совпали с незамаскированными разрядами РгАП, комбинационная схема КС устанавливает 1 в соответствующие разряды регистра совпадения РгСв

Комбинационная схема формирования результата ассоциативной выборки ФР обращения формирует из слова, образовавшегося в РгСв сигналы $\alpha_0, \alpha_1, \alpha_2$, соответствующие случаям: отсутствие слов в ЗМ, удовлетворяющих ассоциативному признаку, наличию одного или более чем одного такого слова.

Формирование содержимого РгСв и сигналов $\alpha_0, \alpha_1, \alpha_2$ по содержимому РгАП, РгМ и ЗМ называется операцией контроля ассоциации. Эта операция является составной частью операций считывания и записи, хотя она имеет и самостоятельное значение.

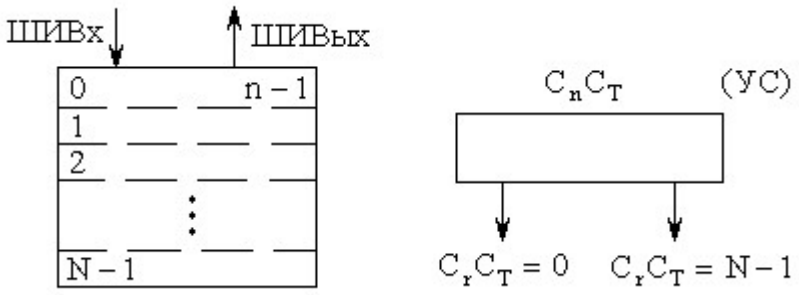
При считывании сначала производится контроль ассоциации ассоциативному признаку в РгАП, затем при $\alpha_0 = 1$ считывание отменяется из-за отсутствия искомой информации. При $\alpha_1 = 1$ считывается в РгИ найденное слово. При $\alpha_2 = 1$ в РгИ считывается слово из ячейки, имеющей наименьший номер среди ячеек, отмеченных 1 в РгСв. Из РгИ считанное слово выдается на ШИВых.

При записи сначала отыскивается свободная ячейка. Для этого выполняется операция контроля ассоциации при РгАП=111...10 и РгМ=00...01. При этом свободные ячейки отмечаются 1 в РгСв. Для записи выбираются свободная ячейка с наименьшим номером. В нее записывается слово, поступившее с ШИВх в РгИ.

Отметим, что для ассоциативной памяти необходимы ЗУ, допускающие считывание без разрушения информации.

5.3.3. **Стековая память**

В стековой памяти ячейки образуют одномерный массив, в котором



соседние ячейки связаны друг с другом разрядными цепями передачи слов. Запись нового слова производится в ячейку (0). При этом все ранее записанные слова (включая в ячейке 0) сдвигаются вниз, в соседние ячейки с большим на 1 номерами. Считывание возможно только из верхней (нулевой) ячейки. В этой памяти порядок считывания слов соответствует правилу: последним поступил - первым обслуживается. Иногда стековая память снабжается счетчиком стека CrCm, показывающим количество занесенных в память слов. Часто стековую память используют для организации стековой адресации. Широкое применение эта память находит при обработке вложенных структур данных.

5.4. **Организация оперативной памяти**

Функционально ОП доступна процессору и другим устройствам ЭВМ, связанным с ОП. Равнодоступность достигается присваиванием адреса каждой ячейке ОП и обеспечением возможности доступа к информации при любом порядке поступления адресов.

5.4.1. **Модульная организация памяти**

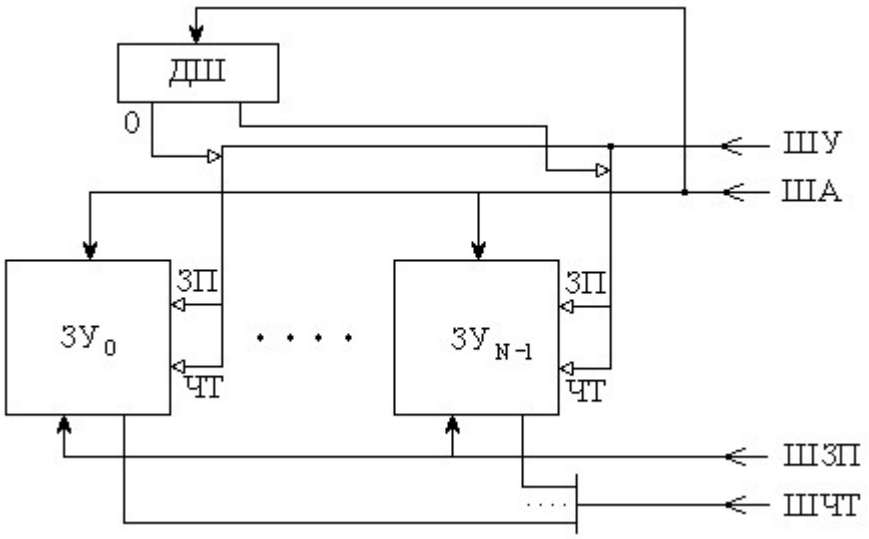
Т.к. быстродействие запоминающих устройств уменьшается с ростом их емкости, то в структурном отношении ОП состоит из комплекса быстродействующих ЗУ, охваченных общей схемой управления. Т.е. применяется принцип модульного построения ОП. Используются модули (блоки) емкостью, например, 32 или 64 Кслов и строится на их основе ОП любой большей емкости. Таким образом ОП является многоблочной (многомодульной).

Адреса ячеек многоблочной ОП имеют следующую структуру

B		C	
0	k-1	0	l-1

Здесь B-K разрядный адрес блока; C-l-разрядный адрес ячейки в блоке B.

Многомодульная ОП строится по следующей схеме



Здесь поле адреса команды подаваемое с ША дешифрируется ДШ и вырабатывается сигнал, подключающий ШУ к заданному блоку. Предполагается, что по сигналу записи ЗП или чтения ЧТ одно ЗУ (один блок) принимает адрес и возможно слово с шины записи ШЗП. После чего выполняется цикл записи или чтения под управлением автономного устройства управления, встроенного в каждый блок ЗУ.

Здесь ДШ - дешифратор адреса. ШУ - шина управления; ША - шина адреса; ШЗП - шина записи; ШЧТ - шина чтения.

В функциональном отношении N-блочная ОП может рассматриваться как одно ЗУ с емкостью, равной сумме емкостей блоков, и быстродействием, примерно равным быстродействию одного блока.

5.4.2. **Оперативная память с многоканальным доступом**

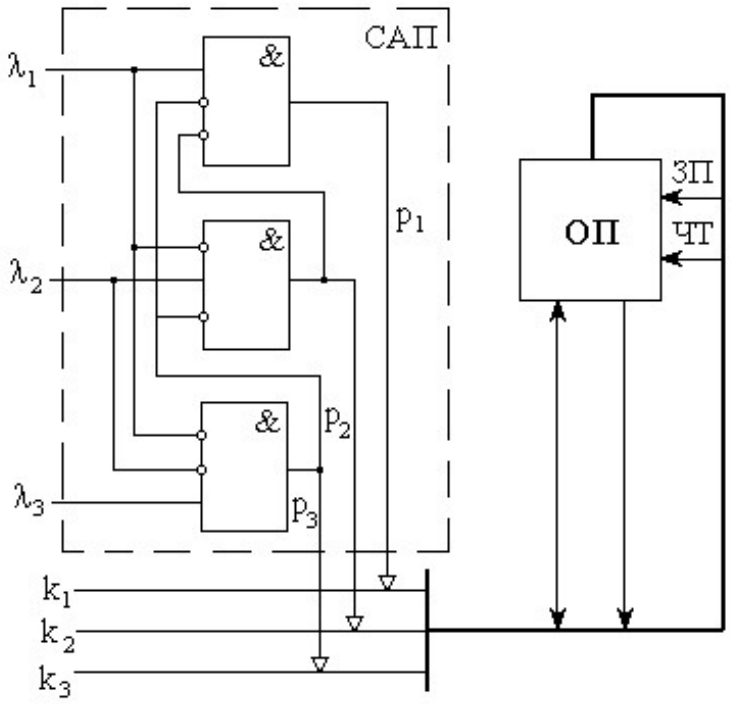
Т.к. устройства, связанные с ОП функционируют одновременно и независимо друг от друга, а память в каждый момент времени может обслуживать только одно обращение, возникает задача распределения ресурсов памяти между несколькими устройствами. Память, ресурсы которой распределяются между несколькими потребителями, называются памятью с многоканальным доступом.

Правило выбора канала, которому будет предоставлено обслуживание, определяется дисциплиной обслуживания. Наиболее естественной является дисциплина

обслуживания в порядке поступления запросов. Конфликтная ситуация, возникающая при одновременном поступлении запросов от нескольких каналов, разрешается путем присваивания каналом различных приоритетов, задаваемых числом. Меньшее значение числа соответствует более высокому приоритету. Описанная дисциплина обслуживания называется обслуживанием с относительными приоритетами.

Приоритеты распределяются следующим образом. Низший присваивается процессору, более высокие - каналам обмена, причем каналу с большим быстродействием присваивается больший приоритет. Указанное распределение приоритетов приводит к тому, что процессор работает в режиме приостановок: в моменты обмена информацией между внешним устройством и ОП процессор приостанавливает. В результате быстродействие процессора зависит от интенсивности потока информации между ОП и ВУ.

Принцип построения памяти с многоканальным доступом можно проиллюстрировать схемой



Здесь ОП - оперативная память

ЗП, ЧТ - сигналы записи, чтение соответственно

k_1, k_2, k_3 - совокупность цепей, составляющих интерфейс ОП.

САП - схема анализа приоритетов.

Каналы генерируют запросы λ_K, принимающие значения 1 в момент обращения к памяти с целью записи или чтения слова информации. Эти сигналы обрабаты-

ваются САП, реализующей систему вышеприведенных функций. В любой момент времени только один сигнал p_K может принять значение 1. Этот сигнал подключает один канал к ОП. В течение цикла записи - чтения все остальные запросы ожидают момента окончания обслуживания, после которого схема выберет на обслуживание очередной запрос с наивысшим приоритетом.

5.4.3. **Организация виртуальной памяти**

В зависимости от количества информации, составляющей задачу, и величины области ОП, отводимой для ее размещения, задача может размещаться в ОП одним из следующих способов:

1) полное размещение программы и данных;
2) полное размещение программы и частичное - данных;
3) частичное размещение программы и полное данных;
4) частичное размещение программы и данных.

Первая ситуация возможна только для коротких задач и встречается достаточно редко. Типичной является ситуация, когда только часть информации размещается в ОП, а остальная - хранится на ВЗУ. Таким образом программист имеет дело с многоуровневой памятью и, планируя процесс решения задачи, включает в программу операции, вызывающие обмен информации между уровнями памяти. Однако всякое априорное планирование обмена информации в многоуровневой памяти, которым занимается программист, не может быть оптимальным, для любой реализации задачи. Издержки из-за неоптимальности процессов обмена информации могут быть значительными, что приводит к необходимости автоматизации работы с многоуровневой памятью.

Автоматическое планирование передач информации в многоуровневой памяти основывается на построении виртуальной (кажущейся) одноуровневой памяти. При этом местом хранения информации является совокупность оперативных и внешних ЗУ с суммарной емкостью E, достаточной для хранения слов с адресами 0, 1, ..., E-1. Указанная совокупность адресов рассматривается в функциональном отношении как виртуальная память. В терминах виртуальных адресов 0, 1, ..., E-1 программируется процесс решения задачи. При этом предполагается, что слова, идентифицированые виртуальными адресами, являются доступными для процессора.

Предположение о равнодоступности E слов информации должно быть физически реализовано путем встраивания в ЭВМ средств, обеспечивающих преобразование виртуальных адресов в адреса ячеек памяти и передачу слов информации в ОП, если адресуемое слово на момент обращения к нему размещается вне ОП. Таким образом, в физическом соотношении виртуальная память - это совокупность оперативных и внешних ЗУ, охваченных средствами преобразования виртуальных адресов в физические адреса ячеек и средствами, автоматизирующими перемещение информации между устройствами памяти.

5.4.4. **Страничная адресация памяти**

Процессы преобразования адресов и перемещения информации наиболее просто реализуется при страничной адресации памяти. Метод страничной адресации состоит в следующем.

Множество адресов (слов, ячеек) разделяется на сегменты, состоящие из 2^K соседних адресов и называемые страницами. Так адреса 0, 1, ..., 2^{K-1} относятся к странице 0, адреса $2^K, 2^{K+1}, ..., 2^{2K}-1$ - к странице 1 и т.д. В результате адрес рассматривается как совокупность двух полей

P		A	
0	l-1	0	k-1

Где P - адрес страницы; A - адрес слова (ячейка) в странице P.

Применительно к виртуальной памяти выделяется два типа адресов: виртуальные и физические. Виртуальный адрес - это адрес, которым идентифицируется некоторое слово в программе. (имеет вышеприведенную структуру)

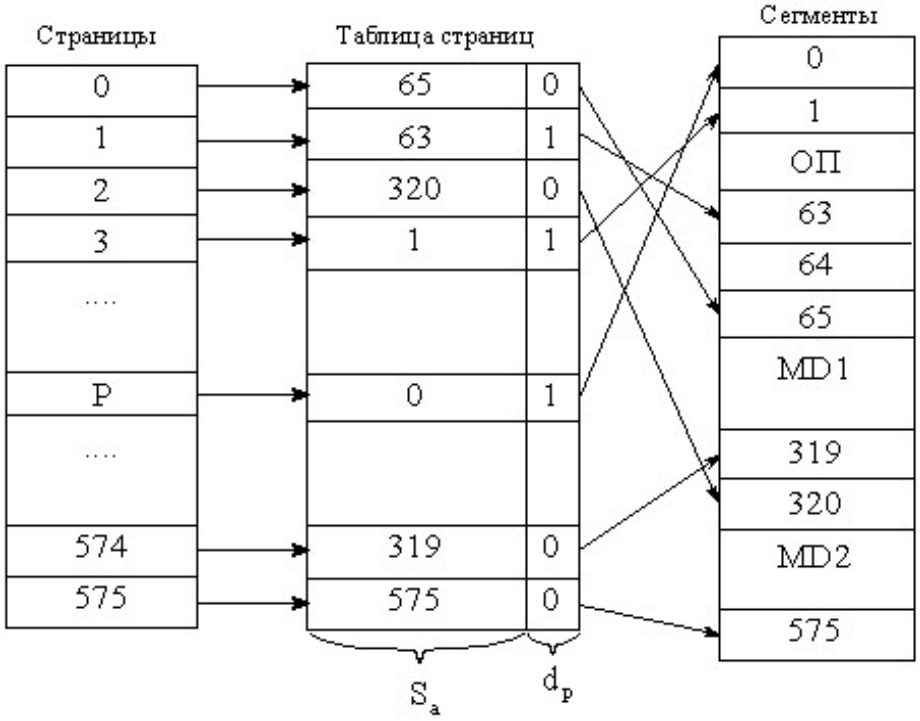
Физический адрес - адрес ячейки памяти. Для физического адреса поля идентифицируются следующим образом.

S		A	
0	l-1	0	k-1

где S - адрес сегмента, состоящего из 2^K соседних ячеек; A - адрес ячейки в сегменте S. Таким образом страница - совокупность из 2^K слов информации, а сегмент - совокупность из 2^K ячеек, являющихся местом хранения страницы.

В процессе решения задачи страницы перемещаются между ОЗУ и ВЗУ. В результате перемещения страница может быть помещена в любой сегмент памяти.

Текущее состояние памяти ЭВМ характеризуется таблицей страниц, которая размещается в ОП и в любое время доступна процессору. Порядок использования таблицы страниц иллюстрируется рисунком



Здесь S_a=0, 1, ..., M физический адрес страницы (адрес сегмента)

Бит d_p- признак доступности страницы P.

Если $d_p = 1$, то P размещается в ОП, если $d_p = 0$ то P→ВЗУ.

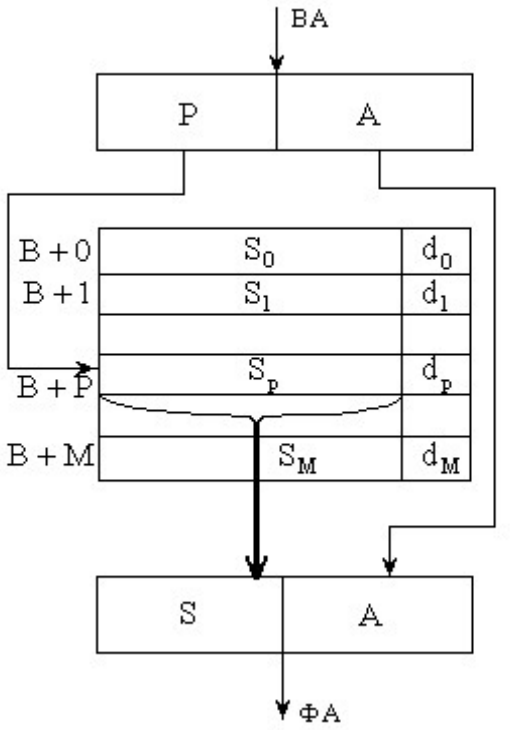
Каждый элемент таблицы страниц содержит номер физической страницы основной памяти и специальный индикатор. Единичное состояние этого индикатора свидетельствует о наличии этой страницы в основной памяти. Нулевое состояние индикатора означает отсутствие страницы в оперативной памяти.

В данном примере виртуальная память имеет емкость 576 страниц, которые могут размещаться в 64 сегментах ОП и на магнитных дисках MD1 и MD2 емкостью по 256 страниц. Каждой странице 0, 1, ...575 соответствует сегмент, адрес которого определен в таблице страниц. Естественно, что любое перемещение страницы сопровождается корректировкой ее физического адреса в таблице страниц.

Для обращения к слову (команда или данные) необходимо преобразовывать виртуальный адрес в физический, для чего в процессор встраивается схема преобразования адресов.

5.4.5. **Преобразование виртуальных адресов**

Приведем наиболее простую схему преобразования адресов.



Виртуальный адрес ВА слова определяется путем обработки относительно адреса, указанного в команде. Для нахождения адреса ячейки в которой хранится слово, производится обращение к ячейке (В+Р) ОП, где В - фиксированный базовый адрес таблицы страниц. И выбираются значения S_p и d_p описывающие страницу Р. Если d_p=0, то в данный момент страница недоступна для обработки. Работа процессора прерывается через систему прерывания и управление передается средствам перемещения страниц.

Если $d_p = 1$ то $S := S_p$ и производится обращение к ячейке S.A оперативной памяти.

Если число физических страниц, отводимых задаче в ОП, достаточно велико, то схема преобразования виртуальных адресов строится на основе АЗУ.

5.5. Сверхоперативные ЗУ

СОЗУ классифицируются в зависимости от способа доступа к хранимой в них информации, который определяется принципом адресации ячеек ЗУ. Наиболее широко используются ЗУ с прямой, стековой, магазинной и ассоциативной адресацией информации.

СОЗУ с прямой адресацией имеют сравнительно небольшую емкость и их функции обычно ограничиваются хранением модификаторов адресов и операндов, относящихся к стандартным программам системы математического обеспечения

ЭВМ. Поэтому эти СОЗУ используются в счете недостаточно эффективно и увеличивают производительность процессора не более чем на 20%.

Обычно СОЗУ с прямой адресацией встраивается в процессор и рассматривается как КЭШ - память (внутренняя) процессора.

Принцип магазинной адресации оказывается весьма удобным для вычисления арифметических и булевых выражений. При вычислениях исходные значения загружаются в СОЗУ со стековой адресацией из ОП. Все промежуточные результаты и конечный результат автоматически записываются в СОЗУ. Вычисления заканчиваются записью результата в ОП.

Выше уже отмечено, что принцип магазинной адресации порождает 0-адресные команды, состоящие только из кода операции. Малая длина таких команд позволяет размещать в одном слове памяти сразу несколько команд, что приводит к уменьшению числа обращений к ОП и, следовательно, увеличивает быстродействие ЭВМ в целом.

В течение некоторого промежутка времени вычислительного процесса лишь часть слов, из числа хранимых в ОП, является активной. Поэтому для повышения быстродействия ЭВМ целесообразно эти активные слова переместить в СОЗУ.

СОЗУ с ассоциативной адресацией, как буферное ЗУ между АЛУ и ОП выполняет следующие действия:

1) хранение активных для некоторого этапа вычислений слов из ОП;
2) контроль ассоциации, т.е. проверка наличия требуемого для вычислений слова и определения его адреса в ассоциативном СОЗУ;
3) чтение слова при обращении АЛУ;
4) запись слова промежуточного или конечного результата операции АЛУ;
5) освобождение некоторой ячейки α СОЗУ в случае отсутствия требуемого для поведения операции слова путем передачи слова из ячейки α СОЗУ в ячейку **A** ОП;
6) обращение к ячейке ОП, с целью записи нового активного слова;

Требуют некоторого пояснения пункты 5, 6. Существует несколько стратегий замещения активных слов в СОЗУ с ассоциативной адресацией:

а) случайное замещение слова, при котором каждое из слов, хранимых в СОЗУ, удаляется с одинаковой вероятностью. Очевидно, что данная стратегия часто приводит к промахам (может быть удалено наиболее активное слово и сохранено

слово, потерявшее активность), что приводит к снижению производительности процессора;

б) удаление слова, к которому не было обращений в течение некоторого промежутка времени. Такая стратегия приводит к усложнению схемы СОЗУ.

в) удаление слова по алгоритму “первый пришел – первый ушел”, по которому замещается слово дольше других, находившееся в СОЗУ;

г) удаление слова по алгоритму “последний пришел – первый ушел”.

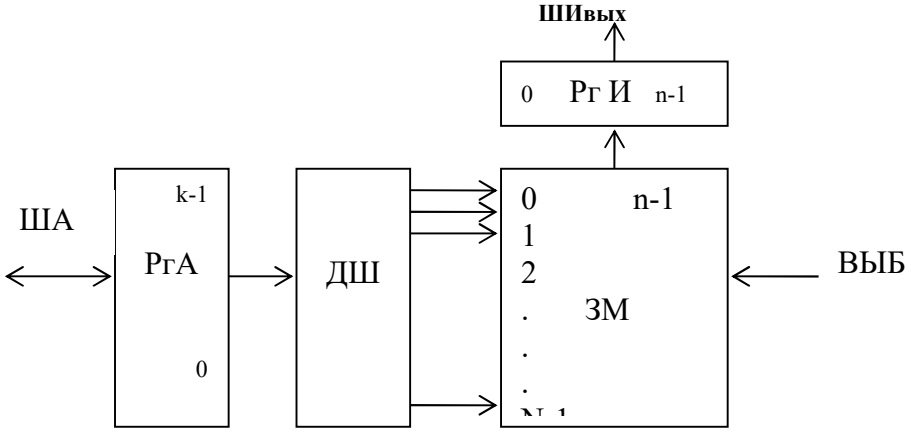
Последние две стратегии обладают тем же недостатком, что и случайное замещение.

5.6. **Постоянные ЗУ.**

Как уже указывалось ПЗУ в рабочем режиме ЭВМ допускают только считывание хранимой информации, а запись производится только в процессе изготовления или в эксплуатационных условия путем настройки, предваряющей использование ПЗУ в вычислительном процессе. В последнем случае ПЗУ называют программируемым постоянным запоминающим устройством (ППЗУ).

ПЗУ строятся обычно как адресные ЗУ. По сравнению с ЗУ с произвольным обращением, допускающим как считывание, так и запись информации, структура ПЗУ значительно проще, их быстродействие и надежность выше, а стоимость ниже. Это объясняется отсутствием цепей для записи информации и реализацией неразрушающего считывания.

Структура простейшего ПЗУ выглядит следующим образом:



По коду адреса, поступающего с шины адреса через регистр адреса, дешифратор выбирает ячейку запоминающей матрицы в которую подается сигнал выборки. Слово информации из выбранной ячейки поступает через Рг И на ШИвых.

В зависимости от типа запоминающих элементов ячеекпамяти ЗМ различают резисторные, емкостные, индуктивные, полупроводниковые интегральные ПЗУ и др. В настоящее время наиболее распространенным типом являются интегральные ПЗУ.

5.7. **Защита памяти.**

Если в памяти одновременно могут находиться несколько программ, необходимы специальные меры для предотвращеня или ограничения обращений одной программы к областям памяти, используемым другими программами. Отдельные программы могут содержать такие ошибки, которые если этому не воспрепятствовать, приводят к искажению информации, принадлежащей другим программам.

Чтобы воспрепятствовать разрушению одних программ другими достаточно защитить область памяти данной программы от попыток записи в нее со стороны других программ.

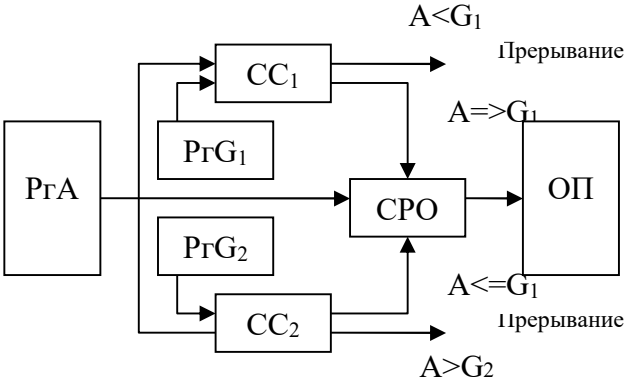
Для защиты памяти в мульти программных ЭВМ используются следующие способы: защита памяти по граничным адресам, защита памяти по ключам.

5.7.1. **Защита памяти по граничным адресам.**

Каждой программе выделяется своя область основной памяти. Предполагается, что различным программам выделяются не перекрывающиеся области. Адреса А команд и операндов, генерируемые в процессе выполнения программы, проверяются на корректность путем сравнения их с граничными адресами: $G_1 <= A <= G_2$, $(G_1 < G_2)$

Адрес считается некорректным, если $A < G_1$ или $A > G_2$

Проверка производится устройством, встраиваемым в процессор, схема которого выглядит следующим образом:



Здесь РгА - регистр адреса; $СС_1$, $СС_2$ - схемы сравнения; $РгG_1$ и $РгG_2$ - регистры граничных адресов; СРО - схема разрешения обращения.

При инициировании программы в $РгG_1$ и $РгG_2$ загружаются значения граничных адресов G_1 и G_2 . Адрес А, сформированный в программе перед обращением к памяти сравнивается на меньше - больше с граничным адресом в СС. Если $A<G_1$ или $A>G_2$ формируется сигнал прерывания, по которому прекращается выполнение программы и она исключается из процесса обработки.

Если в ЭВМ используется страничная организация памяти, то программе выделяются области, состоящие из целого числа страниц. К основному недостатку рассмотренного способа защиты памяти следует отнести необходимость в выделении сплошного участка памяти для размещения программы.

5.7.2. **Защита памяти по ключам.**

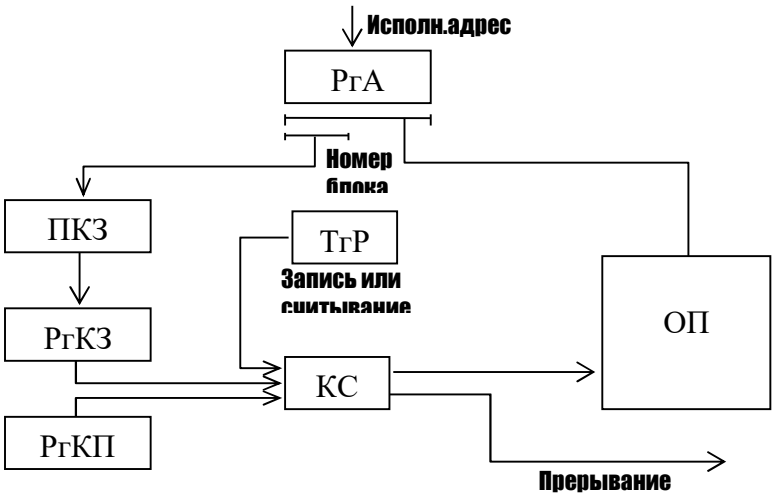
По сравнению с предыдущим, данный метод позволяет организовать доступ программы к областям памяти, расположенным не подряд.

Память в логическом отношении делится на одинаковые блоки. Каждому блоку памяти ставится в соответствие код, называемый ключом защиты памяти, а каждой программе, принимающей участие в мультипрограммной обработке, присваивается код ключа программы. Доступ программы к данному блоку памяти разрешается, если ключи защиты совпадают или один из них имеет код 0. Ключ программы с кодом 0 присваивается программам операционной системы. Ключ защиты памяти с кодом 0 присваивается блокам, хранящим стандартные подпрограммы.

ключ защиты блока памяти		ключ программы	
1 1 0 1	0	1 1 0 1	зп: сч разрешено
0 0 1 1	1	1 1 0 1	зп: сч запрещено
1 1 0 1	0	0 0 0 0	зп: сч разрешено

Дополнительный разряд ключа защиты – разряд режима защиты. Если 0, то защита действует только при попытке записи в блок; если 1 – то защита действует и при записи и при считывании.

Функционирование защиты памяти по ключам можно пояснить схемой



Здесь РгА – регистр адреса; ПКЗ – память ключей защиты, в которой хранятся коды ключей защиты; РгКЗ – регистр ключа защиты памяти; РгКП – регистр ключа программы, регистр процессора, в который ключ заносится при инициировании данной программы; ТгРО – триггер режима обращения (запись или считывание); КС – комбинационная схема.

6. **ОРГАНИЗАЦИЯ СИСТЕМ ВВОДА - ВЫВОДА**

ЭВМ содержит помимо процессора (процессоров) и основной памяти, образующих ее ядро, многочисленные и разнообразные по выполняемым функциям и принципам действия внешние (переферийные) устройства (ВУ), предназначенные для хранения больших объемов информации (внешние запоминающие устройства) и для ввода в ЭВМ и для вывода из нее информации, в том числе для ее регистрации и отображения (устройства ввода-вывода).

Передача информации с внешнего устройства в ядро ЭВМ (память и процессор) называется операцией ввода, а передача из ядра ЭВМ в периферийное устройство - операцией вывода.

Производительность ЭВМ в значительной степени зависит от состава ВУ, их технических данных и способа организации их совместной работы с ядром.

Связь устройств ЭВМ друг с другом осуществляется с помощью интерфейсов, от характеристик которых во многом зависят производительность и надежность вычислительной машины.

6.1. **Элементы организации интерфейсов**.

Интерфейс представляет собой совокупность линий и шин, сигналов, электронных схем и алгоритмов (протоколов), предназначенную для осуществления обмена информацией между устройствами ЭВМ

Физически интерфейс представляет собой совокупность линий и схем формирования сигналов. Все линии интерфейса разбиты на группы линий - шины.

Шины интерфейса в зависимости от их назначения можно разделить на:

- информационные, предназначенные для передачи команд, данных и адресов;
- идентификации типа информации, передаваемой по информационным шинам;
- управляющие, предназначенные для синхронизации, инициирования и завершения передачи информации.

В случае если для передачи адресов, данных и команд используются физически разные шины, необходимость в шинах идентификации отпадает.

Различные структуры шин интерфейса подразделяются на индивидуальные, коллективные и комбинированные.

Наиболее надежной является структура с индивидуальными шинами, поскольку выход из строя группы шин не влияет на работу других устройств. При использовании индивидуальных шин упрощается адресация и идентификация, но увеличивается кол-во оборудования.

Структура с коллективными шинами имеет меньшую надежность. Но при необходимости организации связи с большим числом устройств такая структура позволяет уменьшить объем оборудования.

Интерфейсы характеризуются следующими параметрами:

- пропускной способностью интерфейса - количеством информации, которое может быть передано через интерфейс в единицу времени;
- максимальной частотой передачи информационных сигналов через интерфейс;
- максимально допустимым расстоянием между соединяемыми устройствами;
- динамическими параметрами интерфейса - временем передачи отдельного слова или блока данных с учетом продолжительности процедур подготовки и завершения передачи;
- общим числом проводов (линий) в интерфейсе;
- информационной шириной интерфейса - числом бит или байтов данных,

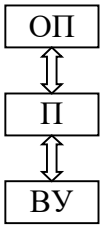
передаваемых параллельно через интерфейс.

Кол-во информационных цепей интерфейса определяется типом элементов информации, которыми обмениваются устройства. Чаще всего по интерфейсу передаются байты, 16-ти или 32-х разрядные слова.

6.3. **Программно-управляемая передача данных и прямой доступ к памяти**

В системах ввода – вывода ЭВМ используется два основных способа организации передачи информации между памятью и ВУ: программно–управляемая передача и прямой доступ к памяти (ПДП).

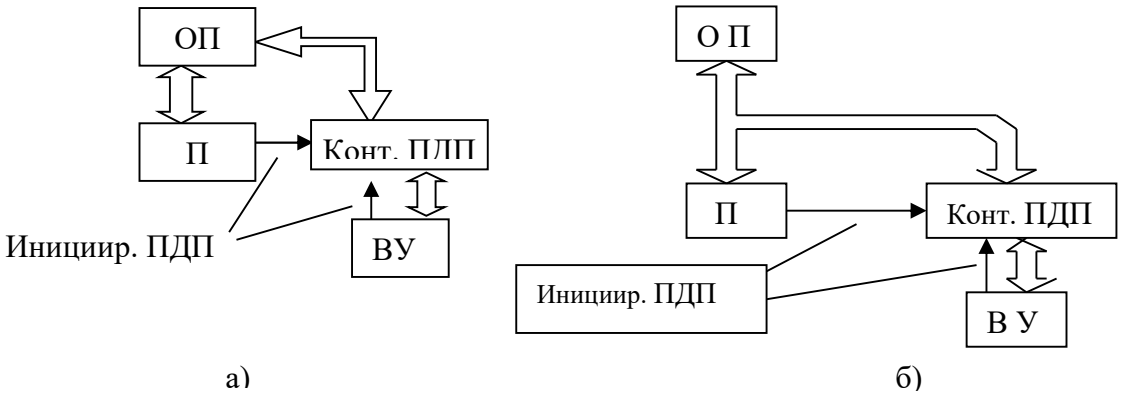
Программно-управляемая передача данных осуществляется при непосредственном участии и под управлением процессора, который при этом выполняет специальную подпрограмму процедуры ввода-вывода. Данные между памятью и ВУ пересылаются через процессор.



Операция ввода-вывода инициируется текущей командой программы или запросом прерывания от ВУ. При программно - управляемой передаче данных процессор на все время этой передачи отвлекается от выполнения основной программы решения задачи. Вместе с тем при пересылке блока данных процессору приходится для каждой единицы передаваемых данных (байт, слово) выполнять довольно много команд, чтобы обеспечить: буферизацию данных, преобразование форматов, подсчет количества переданных данных, формирование адресов памяти и т.д. В результате скорость передачи данных при пересылке блока данных даже через высокопроизводительный процессор может оказаться неприемлемой для работы ЭВМ в реальном времени. Т.е. программно-управляемый обмен данными приемлем только для передачи небольших объемов информации и в основном используется в микро-ЭВМ. Для быстрого ввода-вывода информации и разгрузки процессора от управления операциями ввода-вывода используют прямой доступ к памяти (ПДП).

Прямым доступом к памяти называется способ обмена данными, обеспечивающий автономно от процессора установление связи и передачу данных между ОП и ВУ.

Прямой доступ к памяти организуется по двум основным схемам:



а) прямой доступ к памяти при наличии отдельной шины в памяти для ПДП;

б) прямой доступ к памяти при использовании процессора и ПДП одной шины для связи с памятью.

ПДП освобождает П или МП от управления операциями ввода-вывода. П и ВУ работают независимо, поочередно обращаясь к памяти для чтения-записи слов информации, причем приоритет на доступ к памяти предоставляется ВУ. ПДП позволяет осуществлять параллельно во времени выполнение процессором текущей программы и обмен данными между ВУ и ОП. Т.о., ПДП, разгружая процессор от обслуживания операций ввода-вывода, способствует возрастанию общей производительности ЭВМ.

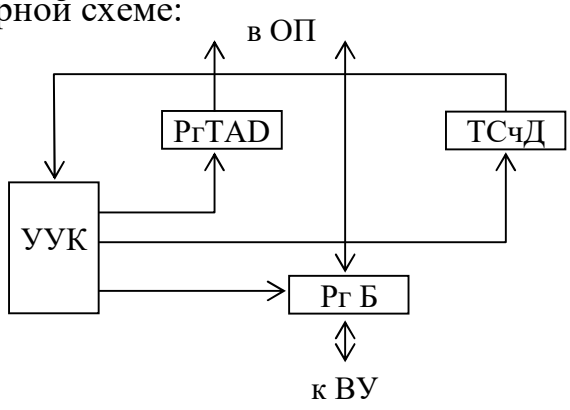
Прямым доступом к памяти управляет контроллер ПДП, который выполняет следующие функции: - управление инициируемой процессором или ВУ передачей данных между ОП и ВУ;

- задание размера блока данных, который подлежит передаче, и области памяти, используемой при передаче;

- формирование адресов ячеек ОП, участвующих в передаче;

- подсчет числа единиц данных (байтов, слов), передаваемых от ВУ в ОП или обратно, и определение момента завершения заданной операции ввода-вывода.

Эти функции могут быть реализованы контроллером ПДП по следующей структурной схеме:



В состав схемы контроллера ПДП входят несколько буферных регистров (РгБ) (в зависимости от числа подключенных ВУ), регистр-счетчик текущего адреса данных (РгTAD), текущий счетчик данных (ТСчД) и устройство управления контроллера (УУК).

При инициировании операции ввода-вывода в ТСчД заносится размер подлежащего передаче блока (число байт или число слов), а в РгТАД - начальный адрес области памяти, используемой при передаче. После передачи каждого байта содержимое РгТАД увеличивается на 1, при этом формируется адрес очередной ячейки ОП, участвующей в передаче. Одновременно уменьшается на 1 содержимое ТСЧД. Обнуление ТСЧД указывает на завершение передачи.

Контроллер ПДП обычно имеет более высокий приоритет в занятии цикла памяти по сравнению с процессором. Поэтому управление памятью переходит к контроллеру ПДП, как только завершится цикл работы памяти, выполняемый для текущей команды процессора.

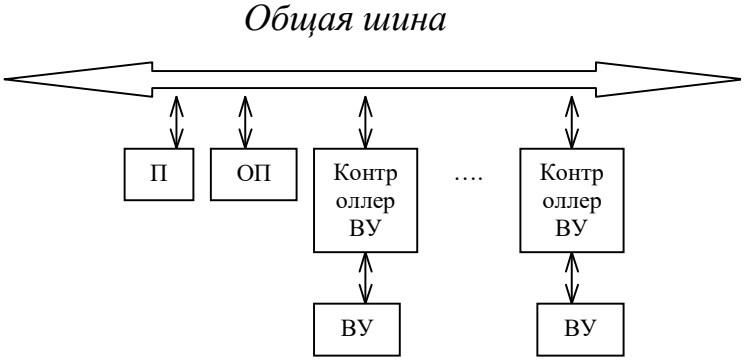
Прямой доступ к памяти обеспечивает высокую скорость обмена данными за счет того, что управление обменом производится не программным путем, а аппаратными средствами.

Как для программного обмена данными, так и для управления работой ВУ с ПДП используется один и тот же набор команд ввода-вывода. Такая унификация управления режимами передачи данных обеспечивается в основном за счет особой адресации ВУ. При этом каждому источнику и приемнику информации присваивается соответственный адрес. ВУ, работающее в режиме программного обмена данными, получает один адрес, обращаясь по которому можно записать или прочитать слово информации.

6.4. **Основные принципы построения и структуры систем ввода-вывода.**

В зависимости от способа реализации интерфейса различают два характерных принципа построения и соответствующие им структуры систем ввода-вывода: ЭВМ с одним общим интерфейсом и ЭВМ с множеством интерфейсов и процессорами (каналами) ввода - вывода.

Структура первого типа имеет следующий вид.



Внешние устройства подключаются к единому интерфейсу через контроллер – устройство управления ВУ.

Контроллер ВУ относится к классу операционных устройств и выполняет следующие функции:

- согласуют форматы данных используемые в ВУ, с форматом, принятым для передачи по единому интерфейсу;
- обеспечивают синхронизацию работы ВУ с другими устройствами;
- обеспечивают буферизацию информации.

Каждый тип ВУ требует применение специфического контроллера.

Применение единого интерфейса порождает следующие правила обмена информацией:

1) информация передается словами, а информационная ширина интерфейса равна длине слова ОП;
2) в каждый момент времени обеспечивается информацией только одна пара у-в;
3) прямой обмен информацией между двумя ВУ невозможен, источником или приемником информации всегда является П или ОП.

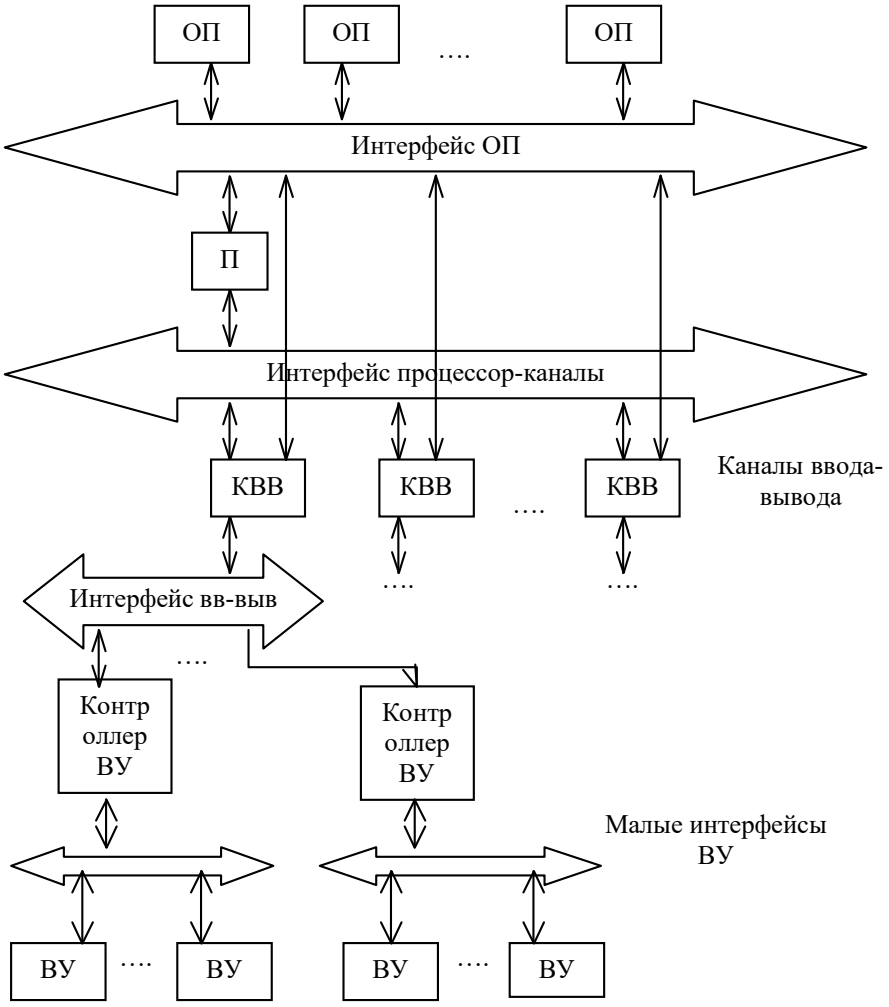
Если необходимость обмена возникает одновременно в нескольких устройствах, то конфликт между ними разрешается с помощью специального устройства - контроллера шин, который часто включается в состав процессора.

Причина обмена информацией между ВУ только через П или ОП - различие быстродействия ВУ, которое приводит к необходимости буферизации. Для буферизации в контроллеры вводят внутреннюю память небольшого объема.

Единый интерфейс - высокоэффективный способ организации обмена информацией в ЭВМ, комплектуемых небольшим количеством ВУ. В едином интерфейсе обеспечивается большое многообразие режимов обмена.

При общем интерфейсе аппаратура управления вводом-выводом рассредоточена по отдельным модулям и ее объем существенно зависит от числа ВУ в составе ЭВМ. Поэтому по схеме с единым интерфейсом строятся мини- и микро- ЭВМ с небольшим числом ВУ. При этом структуры мини ЭВМ строятся по схеме с общей шиной, с микро ЭВМ с мультишиной (модификацией единого интерфейса). Мультишина обладает большими логическими возможностями.

Для построения высокопроизводительных ЭВМ общего назначения, работащих с многобайтными словами, с большим набором ВУ, используется более сложная иерархическая структура системы ввода-вывода с процессорами (каналами) ввода - вывода..



Эта структура часто называется структура ЭВМ на основе канала ввода-вывода. Здесь отсутствует однородность в структуре потоков и форматах представления данных, что приводит к необходимости иметь в ЭВМ несколько специализированных интерфейсов.

В данной структуре используются интерфейсы четырех типов:

- оперативной памяти. Через интерфейс основной памяти производится обмен ин-

формацией между ОП, с одной стороны, и процессором (процессорами) и каналами ввода-вывода - с другой;

- процессор - каналы. Интерфейс "процессор - каналы" предназначается для передачи управляющей информации между процессорами и каналами ввода-вывода;

- ввода-вывода. Через интерфейс ввода-вывода производится обмен информацией между каналами и контроллерами ВУ;

-малые интерфейсы внешних у-в. Через малые интерфейсы осуществляется передача информации между контроллерами ВУ и ВУ.

Наиболее быстродействующими являются интерфейс ОП и интерфейс “процессор-канал”.

При проектировании ЭВМ интерфейсы стремятся унифицировать, в первую очередь интерфейсы, обеспечивающие сопряжение с периферийными устройствами (интерфейсы ввода-вывода). Интерфейсы периферийных устройств не могут быть унифицированы, т.к. сами эти устройства весьма разнообразны по принципу действия, по выполняемым операциям и по используемым форматам данных и сигналам.

Каналы ввода-вывода разгружают процессор от операций ввода-вывода. Они осуществляют прямой доступ к памяти.

Функции контроллеров ВУ в основном остаются такие же, что и в структуре с общим интерфейсом, но общее для всех контроллеров оборудование вынесено в канал, а в контроллере оставлены только схемы специфичные для конкретного типа ВУ.

При большом числе ВУ использование КВВ экономит оборудование за счет централизации в канале весьма сложных функций по обслуживанию ВУ.