### 1. Алгоритм и его свойства.

Алгоритм — это последовательность команд, предназначенная исполнителю, в результате выполнения которой он должен решить поставленную задачу. Алгоритм должен описываться на формальном языке, исключающем неоднозначность толкования.

Алгоритм обладает следующими свойствами:

1. **Дискретность**. Процесс решения задачи должен быть разбит на последовательность отдельных шагов-команд, которые выполняются одна за другой. Только после завершения одной команды начинается выполнение следующей.
2. **Понятность**. Алгоритм должен содержать только те команды, которые известны исполнителю.
3. **Детерминированность**. Каждый шаг и переход от шага к шагу должны быть точно определены, чтобы его мог выполнить любой другой человек или механическое устройство. У исполнителя нет возможности принимать самостоятельное решение (алгоритм исполняется *формально*).
4. **Конечность**. Обычно предполагают, что алгоритм заканчивает работу за конечное число шагов. Результат работы алгоритма также должен быть получен за конечное время. Можно расширить понятие алгоритма до понятия процесса, который по различным каналам получает данные, выводит данные и потенциально может не заканчивать свою работу.
5. **Массовость**. Алгоритм должен решать не одну частную задачу, а класс задач. Не имеет смысла строить алгоритм нахождения наибольшего общего делителя только для чисел 10 и 15.

### 2. Понятие процесса.

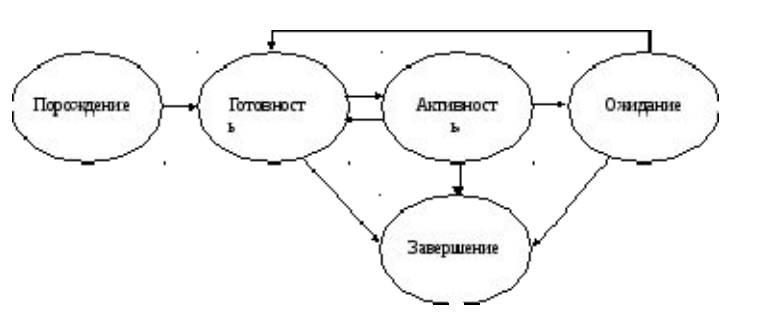
[Процесс (информатика)](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) — идентифицируемая абстракция совокупности взаимосвязанных системных ресурсов на основе отдельного и независимого виртуального адресного пространства, в контексте которой организуется выполнение потоков.

**(Из лекции 2):**

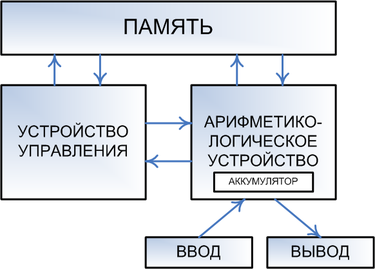
Процессом называется некоторая деятельность, выполняемая на процессоре.

Процесс, выполняем на процессоре, представляет собой программу. При исполнении программы на процессоре выделяются следующие состояния:

* **Порождение** – подготавливаются условия для первого исполнения программы на процессоре;
* **Активность** – программа исполняется на процессоре;
* **Ожидание** – программа не исполняется на процессоре из-за отсутствия какого-либо требуемого ресурса;
* **Готовность** – программа не исполняется, имея все необходимые ресурсы, кроме ЦП;
* **Окончание** – нормальное или аварийное завершение программы, после чего ЦП и другие ресурсы ей не предоставляются.



### 3. Принципы программного управления фон Неймана.



* **Принцип однородности памяти**

Команды и данные хранятся в одной и той же памяти и внешне в памяти неразличимы. Распознать их можно только по способу использования; то есть одно и то же значение в ячейке памяти может использоваться и как данные, и как команда, и как адрес в зависимости лишь от способа обращения к нему. Это позволяет производить над командами те же операции, что и над числами, и, соответственно, открывает ряд возможностей. Так, циклически изменяя адресную часть команды, можно обеспечить обращение к последовательным элементам массива данных. Такой приём носит название модификации команд и с позиций современного программирования не приветствуется. Более полезным является другое следствие принципа однородности, когда команды одной программы могут быть получены как результат исполнения другой программы. Эта возможность лежит в основе трансляции — перевода текста программы с языка высокого уровня на язык конкретной вычислительной машины.

* **Принцип адресности**

Структурно основная память состоит из пронумерованных ячеек, причём процессору в произвольный момент доступна любая ячейка. Двоичные коды команд и данных разделяются на единицы информации, называемые словами, и хранятся в ячейках памяти, а для доступа к ним используются номера соответствующих ячеек — адреса.

* **Принцип программного управления**

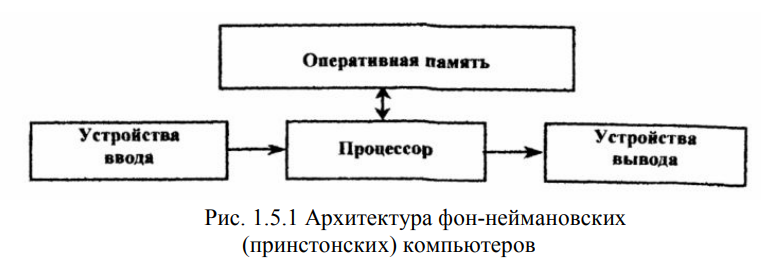
Все вычисления, предусмотренные алгоритмом решения задачи, должны быть представлены в виде программы, состоящей из последовательности управляющих слов — команд. Каждая команда предписывает некоторую операцию из набора операций, реализуемых вычислительной машиной. Команды программы хранятся в последовательных ячейках памяти вычислительной машины и выполняются в естественной последовательности, то есть в порядке их положения в программе. При необходимости, с помощью специальных команд, эта последовательность может быть изменена. Решение об изменении порядка выполнения команд программы принимается либо на основании анализа результатов предшествующих вычислений, либо безусловно.

По плану, первым компьютером, построенным по архитектуре фон Неймана, должен был стать [EDVAC](https://ru.wikipedia.org/wiki/EDVAC) (Electronic Discrete Variable Automatic Computer) — одна из первых электронных вычислительных машин. В отличие от своего предшественника [ЭНИАК](https://ru.wikipedia.org/wiki/%D0%AD%D0%9D%D0%98%D0%90%D0%9A)а, это был компьютер на двоичной, а не десятичной основе. Как и ЭНИАК, EDVAC был разработан в [Институте Мура](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%88%D0%BA%D0%BE%D0%BB%D0%B0_%D0%9C%D1%83%D1%80%D0%B0) [Пенсильванского университета](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D0%BD%D1%81%D0%B8%D0%BB%D1%8C%D0%B2%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82) для Лаборатории баллистических исследований (англ.) [Армии США](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BC%D0%B8%D1%8F_%D0%A1%D0%A8%D0%90) командой инженеров и учёных во главе с [Джоном Преспером Экертом](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D0%BA%D0%B5%D1%80%D1%82,_%D0%94%D0%B6%D0%BE%D0%BD_%D0%9F%D1%80%D0%B5%D1%81%D0%BF%D0%B5%D1%80) и [Джоном Уильямом Мокли](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BA%D0%BB%D0%B8,_%D0%94%D0%B6%D0%BE%D0%BD) при активной помощи математика, однако до 1951 года EDVAC не был запущен из-за технических трудностей в создании надёжной [компьютерной памяти](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) и разногласий в группе разработчиков. Другие научно-исследовательские институты, ознакомившись с ЭНИАКом и проектом EDVAC, сумели решить эти проблемы гораздо раньше. Первыми компьютерами, в которых были реализованы основные особенности архитектуры фон Неймана, были:

1. [прототип](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — [Манчестерская малая экспериментальная машина](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BD%D1%87%D0%B5%D1%81%D1%82%D0%B5%D1%80%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B0%D0%BB%D0%B0%D1%8F_%D1%8D%D0%BA%D1%81%D0%BF%D0%B5%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) — [Манчестерский университет](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BD%D1%87%D0%B5%D1%81%D1%82%D0%B5%D1%80%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82), Великобритания, 21 июня 1948 года;
2. [EDSAC](https://ru.wikipedia.org/wiki/EDSAC) — [Кембриджский университет](https://ru.wikipedia.org/wiki/%D0%9A%D0%B5%D0%BC%D0%B1%D1%80%D0%B8%D0%B4%D0%B6%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82), Великобритания, 6 мая 1949 года;
3. [Манчестерский Марк I](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BD%D1%87%D0%B5%D1%81%D1%82%D0%B5%D1%80%D1%81%D0%BA%D0%B8%D0%B9_%D0%9C%D0%B0%D1%80%D0%BA_I) — [Манчестерский университет](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BD%D1%87%D0%B5%D1%81%D1%82%D0%B5%D1%80%D1%81%D0%BA%D0%B8%D0%B9_%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82), Великобритания, 1949 год;
4. [BINAC](https://ru.wikipedia.org/wiki/BINAC) — США, апрель или август 1949 года;
5. [CSIR Mk 1](https://ru.wikipedia.org/wiki/CSIRAC) — Австралия, ноябрь 1949 года;
6. [EDVAC](https://ru.wikipedia.org/wiki/EDVAC) — США, август 1949 года — фактически запущен в 1952 году;
7. [CSIRAC](https://ru.wikipedia.org/wiki/CSIRAC) — Австралия, ноябрь 1949 года;
8. [SEAC](https://ru.wikipedia.org/wiki/SEAC) — США, 9 мая 1950 года;
9. [ORDVAC](https://ru.wikipedia.org/wiki/ORDVAC) — США, ноябрь 1951 года;
10. [IAS-машина](https://ru.wikipedia.org/wiki/IAS-%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) — США, 10 июня 1952 года;
11. [MANIAC I](https://ru.wikipedia.org/wiki/MANIAC_I) — США, март 1952 года;
12. [AVIDAC](https://ru.wikipedia.org/wiki/AVIDAC) — США, 28 января 1953 года;
13. [ORACLE](https://ru.wikipedia.org/wiki/ORACLE_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)) — США, конец 1953 года;
14. [WEIZAC](https://ru.wikipedia.org/wiki/WEIZAC) — Израиль, 1955 год;
15. [SILLIAC](https://ru.wikipedia.org/wiki/SILLIAC) — Австралия, 4 июля 1956 года.

В СССР первой полностью электронной вычислительной машиной, близкой к принципам фон Неймана, стала [МЭСМ](https://ru.wikipedia.org/wiki/%D0%9C%D0%AD%D0%A1%D0%9C), построенная [Лебедевым](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%B1%D0%B5%D0%B4%D0%B5%D0%B2,_%D0%A1%D0%B5%D1%80%D0%B3%D0%B5%D0%B9_%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B5%D0%B5%D0%B2%D0%B8%D1%87) (на базе киевского [Института электротехники АН УССР](https://ru.wikipedia.org/w/index.php?title=%D0%98%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B8_%D0%90%D0%9D_%D0%A3%D0%A1%D0%A1%D0%A0&action=edit&redlink=1)). [МЭСМ](https://ru.wikipedia.org/wiki/%D0%9C%D0%AD%D0%A1%D0%9C) как прототип впервые была публично запущена 6 ноября 1950 года и уже в качестве полноценной машины прошла государственные приёмочные испытания 25 декабря 1951 года.

**ВЕРСИЯ ИЗ ЛЕКЦИЙ:**



1. **Принцип программного управления**. Из него следует, что программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности.

Выборка программы из памяти осуществляется с помощью счетчика команд. Этот регистр процессора последовательно увеличивает хранимый в нем адрес очередной команды на длину команды. Так как команды программы расположены в памяти друг за другом, то тем самым организуется выборка цепочки команд из последовательно расположенных ячеек памяти.

Если после выполнения команды следует перейти не к следующей, а к какой-то другой, используются команды условного или безусловного переходов (ветвления), которые заносят в счетчик команд номер ячейки памяти, содержащей следующую команду. Выборка команд из памяти прекращается после достижения и выполнения команды «стоп».

Таким образом, процессор исполняет программу автоматически, без вмешательства человека.

1. **Принцип однородности памяти**. Программы и данные хранятся в одной и той же памяти. Поэтому компьютер не различает, что хранится в данной ячейке памяти — число, текст или команда. Над командами можно выполнять такие же действия, как и над данными. Это открывает целый ряд возможностей. Например, программа в процессе своего выполнения также может подвергаться переработке, что позволяет задавать в самой программе правила получения некоторых ее частей (так в программе организуется выполнение циклов и подпрограмм). Более того, команды одной программы могут быть получены как результаты исполнения другой программы. На этом принципе основаны методы трансляции — перевода текста программы с языка программирования высокого уровня на язык конкретной машины.
2. **Принцип адресности**. Структурно основная память состоит из перенумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Отсюда следует возможность давать имена областям памяти, так, чтобы к запомненным в них значениям можно было впоследствии обращаться или менять их в процессе выполнения программ с использованием присвоенных имен

### 4. Принстонская архитектура вычислительных машин.

##### **Принстонская архитектура**

***Принстонская архитектура***, которая часто называется ***архитектурой фон Неймана***, характеризуется использованием общей оперативной памяти для хранения программ, данных, а также для организации стека. Для обращения к этой памяти используется общая системная шина, по которой в процессор поступают и команды, и данные.

Архитектура современных персональных компьютеров основана на

***магистрально-модульном принципе***.

Любую вычислительную машину образуют три основные компонента:

* процессор,
* память,
* устройства ввода-вывода (УВВ).

Информационная связь между устройствами компьютера осуществляется через системную шину (системную магистраль).

***Шина*** – это кабель, состоящий из множества проводников. Количество проводников, входящих в состав шины, является

***максимальной разрядностью шины***.

Системная шина, в свою очередь, представляет собой совокупность

* шины данных, служащей для переноса информации;
* шины адреса, которая определяет, куда переносить информацию;
* шины управления, которая определяет правила для передачи информации;
* шины питания, подводящей электропитание ко всем узлам вычислительной машины.

Системная шина характеризуется тактовой частотой и разрядностью. Количество одновременно передаваемых по шине бит называется

***разрядностью шины***.

Тактовая частота характеризует число элементарных операций по передаче данных в 1 секунду. Разрядность шины измеряется в битах, тактовая частота – в мегагерцах.

***Устройство управления*** (УУ) формирует адрес команды, которая должна быть выполнена в данном цикле, и выдает управляющий сигнал на чтение содержимого соответствующей ячейки запоминающего устройства (ЗУ). Считанная команда передается в УУ. По информации, содержащейся в адресных полях команды, УУ формирует адреса операндов и управляющие сигналы для их чтения из ЗУ и передачи в арифметико-логическое устройство (АЛУ). После считывания операндов устройство управления по коду операции, содержащемуся в команде, выдает в АЛУ сигналы на выполнение операции. Полученный результат записывается в ЗУ по адресу приемника результата под управлением сигналов записи. Признаки результата (знак, наличие переполнения, признак нуля и так далее) поступают в устройство управления, где записываются в специальный регистр признаков. Эта информация может использоваться при выполнении следующих команд программы, например команд условного перехода.

***Устройство ввода*** позволяет ввести программу решения задачи и исходные данные в ЭВМ и поместить их в оперативную память. В зависимости от типа устройства ввода исходные данные для решения задачи вводятся непосредственно с клавиатуры, либо они должны быть предварительно помещены на какой-либо носитель (дисковый накопитель).

***Устройство вывода*** служит для вывода из ЭВМ результатов обработки исходной информации. Чаще всего это символьная информация, которая выводится с помощью печатающих устройств или на экран дисплея.

***Запоминающее устройство*** или ***память*** – это совокупность ячеек, предназначенных для хранения некоторого кода. Каждой из ячеек присвоен свой номер, называемый ***адресом***. Информацией, записанной в ячейке, могут быть как команды в машинном виде, так и данные.

Обработка данных и команд осуществляется посредством ***арифметико-логического устройства*** (АЛУ), предназначенного для непосредственного выполнения машинных команд под действием устройства управления. АЛУ и УУ совместно образуют ***центральное процессорное устройство*** (ЦПУ). Результаты обработки передаются в память.

**Основные принципы построения вычислительных машин с архитектурой фон Неймана**

* Принцип двоичности. Для представления данных и команд используется двоичная система счисления.
* Принцип программного управления. Программа состоит из набора команд, которые выполняются процессором друг за другом в определённой последовательности.
* Принцип однородности памяти. Как программы (команды), так и данные хранятся в одной и той же памяти (и кодируются в одной и той же системе счисления, чаще всего – двоичной). Над командами можно выполнять такие же действия, как и над данными.
* Принцип адресуемости памяти. Структурно основная память состоит из пронумерованных ячеек, процессору в произвольный момент времени доступна любая ячейка.
* Принцип последовательного программного управления. Все команды располагаются в памяти и выполняются последовательно, одна после завершения другой.
* Принцип условного перехода. Команды из программы не всегда выполняются одна за другой. Возможно присутствие в программе команд условного перехода (а также команд вызова функций и обработки прерываний), которые изменяют последовательность выполнения команд в зависимости от значений данных. Этот принцип был сформулирован задолго до фон Неймана Адой Лавлейс и Чарльзом Бэббиджем, однако был логически включен в указанный набор как дополняющий предыдущий принцип.

Архитектура фон Неймана имеет ряд важных достоинств.

* Наличие общей памяти позволяет оперативно перераспределять ее объем для хранения отдельных массивов команд, данных и реализации стека в зависимости от решаемых задач. Таким образом, обеспечивается возможность более эффективного использования имеющегося объема оперативной памяти в каждом конкретном случае применения.
* Использование общей шины для передачи команд и данных значительно упрощает отладку, тестирование и текущий контроль функционирования системы, повышает ее надежность.

Поэтому Принстонская архитектура в течение долгого времени доминировала в вычислительной технике.

Однако ей присущи и существенные недостатки. Основным из них является необходимость последовательной выборки команд и обрабатываемых данных по общей системной шине. При этом общая шина становится «узким местом» (bottleneck – «бутылочное горло»), которое ограничивает производительность цифровой системы.

**ИЗ ЛЕКЦИИ:**

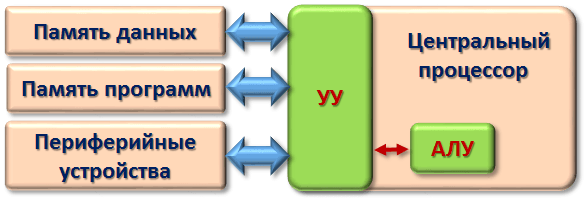
****

Эти Компьютеры относятся к типу фон-неймановских или, как их еще иначе называют, принстонских и характеризуются использованием общей оперативной памяти для хранения программ, данных, а также для организации стека. Для обращения к этой памяти используется общая системная шина, по которой в процессор, по которой в процессор поступают и команда, и данные. Эта архитектура имеет ряд важных достоинств. Наличие общей памяти позволяет оперативно перераспределять ее объем для хранения отдельных массивов команд, данных и реализации стека в зависимости от решаемых задач. Таким образом, обеспечивается возможность более эффективного использования имеющегося объема оперативной памяти в каждом конкретном случае применения МП. Использование общей шины для передачи команд и данных значительно упрощает отладку, тестирование и текущий контроль функционирования системы, повышает ее надежность. Поэтом Принстонская архитектура в течение долгого времени доминировала в вычислительной технике.

Однако ей присущи и существенные недостатки. Основным из них является необходимость последовательной выборки команд и обрабатываемых данных по общей системной шине. При этом общая шина становится «узким местом», которое ограничивает производительность цифровой системы.

Данная проблема решается совершенствованием систем кэширования, что в свою очередь усложняет архитектуру систем и увеличивает риск возникновения побочных ошибок (например, в 2017 году были обнаружены уязвимости Meltdown и Spectre, присутствовавшие в современных процессорах в течение десятилетий, но не обнаруженные ранее из-за сложности современных вычислительных систем и, в частности, их взаимодействия с кэш-памятью).

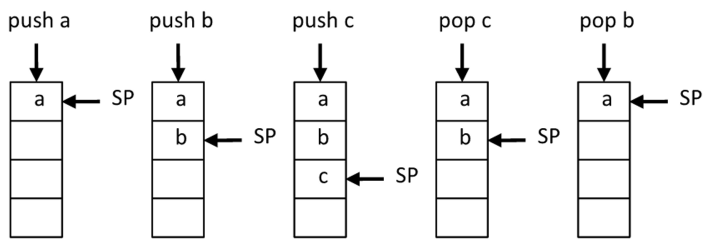
### 5. Гарвардская архитектура вычислительных машин.

***Гарвардская архитектура*** была разработана Говардом Эйкеном в конце 1930-х годов в Гарвардском университете с целью увеличить скорость выполнения вычислительных операций и оптимизировать работу памяти. Она характеризуется физическим разделением памяти команд (программ) и памяти данных. В ее оригинальном варианте использовался также отдельный стек для хранения содержимого программного счетчика, который обеспечивал возможности выполнения вложенных подпрограмм. Каждая память соединяется с процессором отдельной шиной, что позволяет одновременно с чтением-записью данных при выполнении текущей команды производить выборку и декодирование следующей команды. Благодаря такому разделению потоков команд и данных и совмещению операций их выборки реализуется более высокая производительность, чем при использовании Принстонской архитектуры.Недостатки Гарвардской архитектуры связаны с необходимостью проведения большего числа шин, а также с фиксированным объемом памяти, выделенной для команд и данных, назначение которой не может оперативно перераспределяться в соответствии с требованиями решаемой задачи. Поэтому приходится использовать память большего объема, коэффициент использования которой при решении разнообразных задач оказывается более низким, чем в системах с Принстонской архитектурой. Однако развитие микроэлектронной технологии позволило в значительной степени преодолеть указанные недостатки, поэтому Гарвардская архитектура широко применяется во внутренней структуре современных высокопроизводительных микропроцессоров, где используется отдельная кэш-память для хранения команд и данных. В то же время во внешней структуре большинства микропроцессорных систем реализуются принципы Принстонской архитектуры.

### 6. Стековая архитектура вычислительных машин.

Стеком называется память, по своей структурной организации отличная от основной памяти ЭВМ, Принципы построения стековой памяти детально рассматриваются позже, здесь же выделим только те аспекты, которые требуются для пояснения особенностей архитектуры на базе стека.

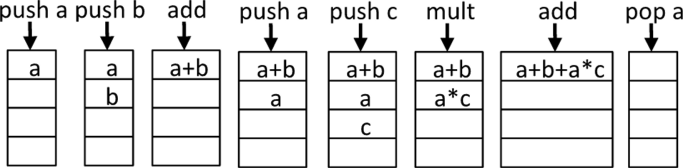
Стек образует множество логически взаимосвязанных ячеек (рисунок 4.5), взаимодействующих по принципу «последним вошел, первым вышел» (LIFO, Last In First Out).

Верхнюю ячейку называют вершиной стека (SP). Для работы со стеком предусмотрены две операции: push (проталкивание данных в стек) и pop (выталкивание данных из стека). Запись возможна только в ячейку стека с адресом SP-1, при этом указатель стека смещается на одну позицию вниз. Чтение допустимо только из вершины стека (ячейка с адресом SP). Извлеченная информация удаляется из стека, а указатель продвигается вверх - принимает значение SP+1.

В вычислительных машинах, где реализована архитектура на базе стека (их обычно называют стековыми), операнды выбираются из двух верхних ячеек стековой памяти. Результат операции заносится в вершину стека. Принцип действия стековой машины рассмотрим на примере вычисления выражения:



При описании вычислений с использованием стека обычно используется иная форма записи математических выражений, известная как обратная польская нотация, которую предложил польский математик Я. Лукашевич. Особенность ее в том, что в выражении отсутствуют скобки, а знак операции располагается не между операндами, а следует за ними (постфиксная форма). Последовательность операций определяется их приоритетами. Рассмотренное выше выражение в польской нотации примет вид:

Данная форма записи однозначно определяет порядок загрузки операндов и выполнения операций в стековой архитектуре. Порядок выполнения команд в ЭВМ со стековой архитектурой представлен на рисунке 4.6.Основные узлы и информационные тракты одного из возможных вариантов ЭВМ на основе стековой архитектуры показаны на рисунке 4.7.

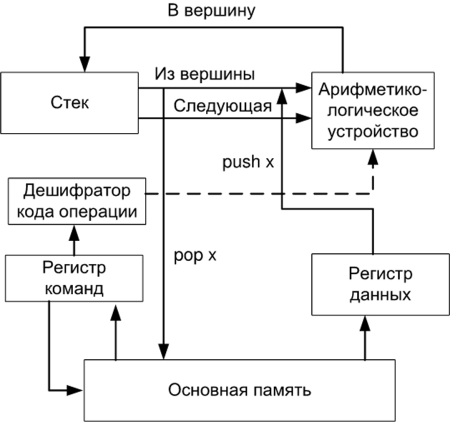
Информация может быть занесена в вершину стека из памяти или из АЛУ. Для записи в стек содержимого ячейки памяти с адресом х выполняется команда *push* х, по которой информация считывается из ячейки памяти, заносится в регистр данных, а затем проталкивается в стек. Результат операции из АЛУ заносится в вершину стека автоматически.

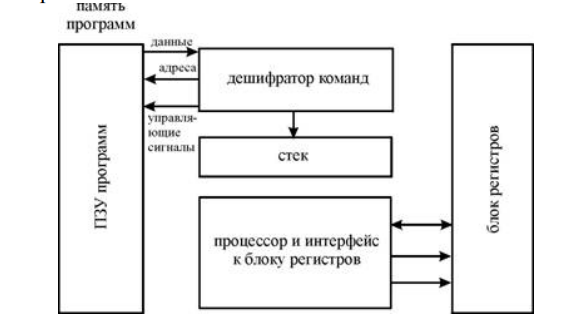
Сохранение содержимого вершины стека в ячейке памяти с адресом *х* производится командой *pop х.* По этой команде содержимое верхней ячейки стека подается на шину, с которой и производится запись в ячейку *х,* после чего производится инкремент указателя стека: SP+1.

Для выполнения арифметической или логической операции на вход АЛУ по дается информация, считанная из двух верхних ячеек стека (при этом содержимое стека продвигается на две позиции вверх, то есть операнды из стека удаляются). Результат операции заталкивается в вершину стека. Возможен вариант, когда результат сразу же переписывается в память с помощью автоматически выполняемой операции *pop х.*

Верхние ячейки стековой памяти, где хранятся операнды и куда заносится результат операции, как правило, делаются более быстродействующими и размещаются в процессоре, в то время как остальная часть стека может располагаться в основной памяти и частично даже на магнитном диске.

К достоинствам ЭВМ на базе стека следует отнести возможность сокращения адресной части команд, поскольку все операции производятся через вершину стека, то есть адреса операндов и результата в командах арифметической и логической обработки информации указывать не нужно. Код программы получается компактным. Достаточно просто реализуется декодирование команд.





С другой стороны, стековая архитектура по определению не предполагает произвольного доступа к памяти, из-за чего компилятору трудно создать эффективный программный код, хотя создание самих компиляторов упрощается. Кроме того, стек становится «узким местом» ЭВМ в плане повышения производительности.

### 7. Пост неймановские архитектуры вычислительных машин.

**Архитектура постнеймановских компьютеров.**

В настоящее время различают архитектуру больших универсальных компьютеров-мейнфреймов (наиболее типичными представителями являются компьютеры серий IBM 360/370 и их «потомков» ES9000) и архитектуру мини-, микро- и персональных компьютеров.

Особенностью универсальных компьютеров является параллельная и асинхронная работа процессора и специализированных процессоров ввода-вывода - каналов ввода-вывода. Каналы ввода-вывода полностью управляют всеми периферийными устройствами. Взаимодействие периферийных устройств с каналами и каналов с процессором обеспечивается системой прерывания. Если при выполнении программы возникает необходимость в работе периферийного устройства, то процессор инициализирует канал на выполнение данной операции, после чего продолжает выполнять основную программу. О завершении своей работы канал сообщает процессору прерыванием. Такая архитектура наиболее эффективная в понимании быстродействия, но требует больших аппаратных затрат (каналы ввода-вывода по своей архитектуре более сложные, чем процессор), сложного управления и имеют более низкую архитектурную надежность.

Универсальные компьютеры типа IBM 360/370 используются в режиме мультипрограммной обработки информации для многих пользователей и имеют широкий набор периферийных устройств. Типовая архитектура изображена на рис 1.5.3.



Процессор имеет арифметико-логическое устройство (АЛУ), устройство центрального управления (ЦУ), устройство управления памятью (УУП) и устройство контроля и диагностики (УКиД). Арифметико-логическое устройство (АЛУ) выполняет арифметические и логические операции над двоичными и двоично-десятичными числами. Устройство центрального управления (ЦУ) обеспечивает микропрограммное управление всего процессора, обработку прерываний и отсчет времени. Устройство управления памятью (УУП) обеспечивает связь процессора и каналов ввода-вывода с оперативным запоминающим устройством (ОЗУ), решения конфликтов при обращении к памяти и буферизацию информации, которая передается. Устройство контроля и диагностики (УКиД) обеспечивает текущий контроль функционирования компьютера при инициализации системы.

Мультиплексный канал является специализированным процессором ввода-вывода и обеспечивает ввод/вывод информации из медленнодействующих периферийных устройств (ПУ). Он работает в мультиплексном режиме, то есть после чтения/записи одного байта информации из одного периферийного устройства возможный обмен байтом информации с другим более приоритетным устройством (если канал получи запрос на обмен);

Селекторные каналы также являются специализированными процессорами ввода-вывода, но они предназначены для работы с периферийными быстродействующими устройствами, например, с устройствами внешней памяти, накопителями на дисках. Селекторный канал работает в селекторном режиме, то есть, если начался обмен информацией с одним устройством, то он не может быть прерван другим, даже более приоритетным, устройством.

Все периферийные устройства подключаются к каналу через свои устройства управления (УУ), что обеспечивают стандартное подключение разнотипных устройств к каналам.

### 8. Семантический разрыв и способы его преодоления.

– это сложности, возникающие во время преобразования оператор языков высокого уровня в машинный код.

Решения: специализация машин, при которой операторы проблемных языков могут выполняться аппаратными средствами машины и не требовать трансляции.

Примеры: аппаратная реализация графических преобразований; аппаратная реализация операций с векторами и матрицами.

### 9. Понятие о виртуальной машине.

- виртуальная вычислительная среда, в которой может быть запущена гостевая операционная система.

Данная операционная система запущена второй и работает в отдельном окне. В ней также можно запускать программы и работать как обычно. Таких окон с разными операционными системами можно создать несколько. Количество установленных на одном компьютере виртуальных машин ограничивается лишь ресурсами самого компьютера.

Виртуальные машины могут использоваться для следующего:

* Создание и развертывание приложений в облаке.
* Тестирование новых операционных систем, в том числе бета-версий.
* Развертывание новой среды, чтобы упростить и ускорить выполнение сценариев разработки и тестирования для разработчиков.
* Резервное копирование существующей ОС.
* Доступ к зараженным вирусом данным или выполнение старого приложения путем установки старой версии ОС.
* Запуск программного обеспечения или приложений в операционных системах, для которых они изначально не предназначались.

### 10. Многоуровневая организация ЭВМ.

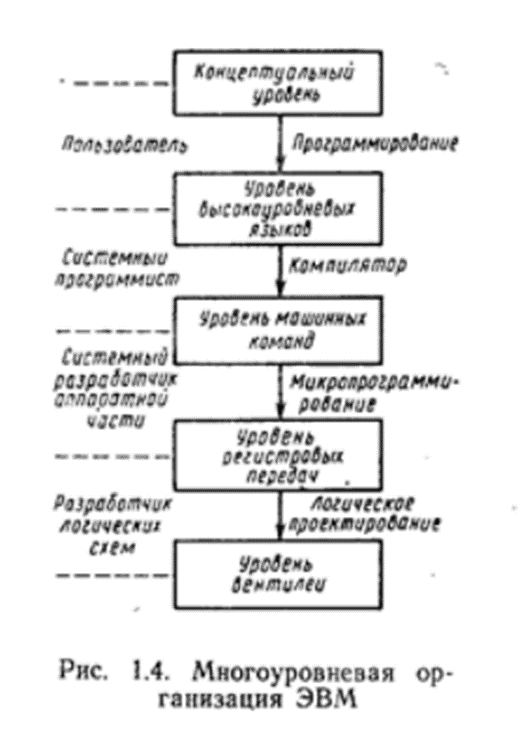
1. Прикладной уровень. На этом уровне реализованы различные пакеты прикладных программ (MS Office, MathCad, PhotoShop…), языки высокого уровня (C, C++, Pascal, Lisp…) и т.д. То есть этот уровень представляет собой профессиональную область, в которой решаются какие-то конкретные задачи.

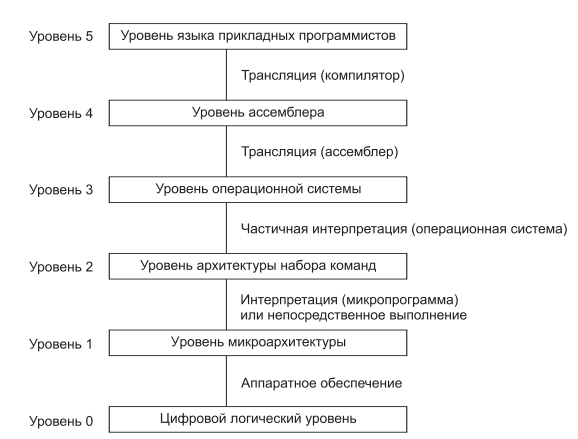
2. Уровень операционной системы. Здесь находятся трансляторы, загрузчики и прочие программы более низкого уровня. Управление осуществляется на логическом уровне. В данной области работают математики и программисты.

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.

****

****

На концептуальном уровне пользователь ЭВМ анализирует задачу, разрабатывает алгоритм ее решения, определяет содержимое обработки информации.

На уровне языков программирования высо­кого уровня изучается алгоритм решения задачи, состав­ляется детальный проект решения (определяются структуры данных, содержание отдельных программных модулей, свя­зи между ними), пишется программа на одном из языков вы­сокого уровня.

На уровне машинных команд обеспечивается связь программных и аппаратных средств. На этом уровне состав­ляется список команд, определяются

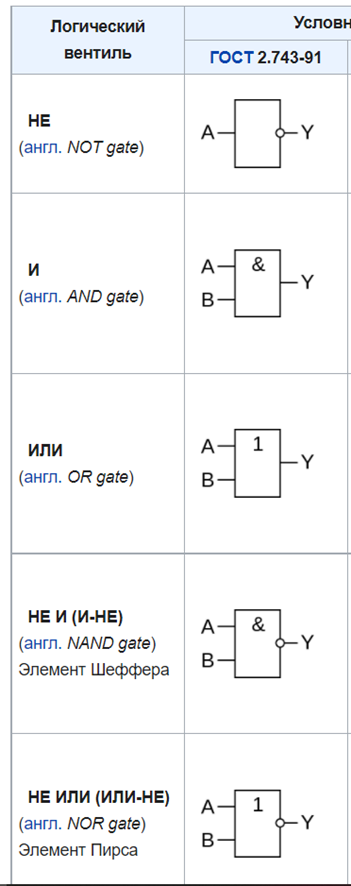
способы кодирования кодов операций и адресов, число адресных полей и другие параметры, заложенные в структуру ЭВМ.

На уровне регистровых передач осуществляются элементарные операции, выполняемые аппаратурой ЭВМ. Это операции преобразования информации, операции чтения и записи в запоминающее устройство и регистры, операции коммутации, обеспечиваю­щие передачу слов между отдельными модулями и блоками.

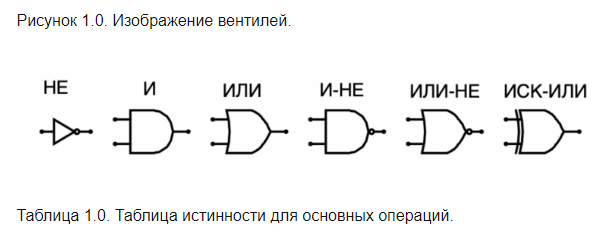
### 11. Основные вентили. Представление логических функций.

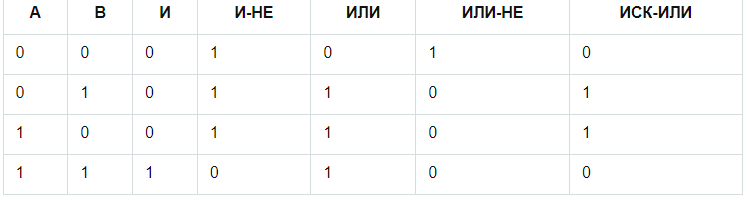
***Логический вентиль*** – это своего рода атом, из которого состоят электронные узлы ЭВМ.

**Логи́ческий ве́нтиль** — базовый элемент цифровой схемы, выполняющий элементарную логическую операцию[[1]](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B2%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C#cite_note-1), преобразуя таким образом множество входных логических сигналов в выходной логический сигнал. Логика работы вентиля основана на [битовых операциях](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82%D0%BE%D0%B2%D1%8B%D0%B5_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8)[[2]](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B2%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C#cite_note-2) с входными цифровыми сигналами в качестве [операндов](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D0%BD%D0%B4).

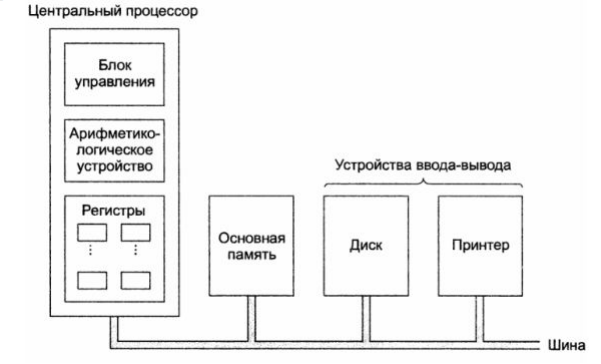


Вентиль (рис. 1.0) является устройством, которое принимает входные сигналы и выдает выходные.





### 12.Основные блоки и функции процессора.

Основные блоки процессора: Блок управления и АЛУ. 

**Блок управления** предназначен для реализации выборки команд, их дешифрации, и на основе этого – для управления обменом и обработкой информации путем генерации последовательности управляющих сигналов. Управляющий блок выдает последовательность сигналов, которые обеспечивают выполнение данной команды. Информационные сигналы зависят не только от исходных значений обрабатываемых данных, но и от результатов получаемых в процессе обработки.

**Арифметико-логическое устройство** служит для обработки цифровой информации (арифметические и логические операции, сдвиги, анализ чисел и т.п.)

Арифметической операцией называют процедуру обработки данных, аргументы и результат которой являются числами (сложение, вычитание, умножение, деление). Логической операцией именуют процедуру, осуществляющую построение сложного высказывания (операции И, ИЛИ, НЕ, ...). АЛУ состоит из регистров, сумматора с соответствующими логическими схемами и блока управления выполняемым процессом.

### 13.Система прерываний, ее функции.

Система прерываний является неотъемлемой частью любого компьютера и предназначена для обеспечения быстрой реакции процессора на ряд ситуаций, требующих его внимания, которые могут возникнуть не только в самом компьютере, но и за его пределами.

*Программные* прерывания связаны с выполняемой программой и являются синхронными по отношению к этой программе.

*Аппаратные* прерывания могут возникать в произвольные моменты времени и являются асинхронными по отношению к выполняемой программе. С помощью аппаратного прерывания осуществляется взаимодействие процессора с периферийными устройствами (клавиатура, магнитные диски, таймер и т.п.), а также сообщается о различных ошибках аппаратуры (ошибка памяти, ошибка данных по шине и т.п.).

**Функции системы прерываний**:

* Прием и хранение запросов прерываний от многих источников.
* Выделение наиболее приоритетного запроса из множества поступивших.
* Проверка возможности обработки запроса центральным процессором (проверка замаскированности запросов или сравнение уровня приоритетности запросов с так называемым порогом прерываний).
* Сохранение состояния (контекста) прерываемой программы.
* Вызов соответствующего обработчика прерываний.
* Обработка прерываний (выполнение программы обработчика прерываний).
* Восстановление состояния (контекста) прерванной программой и возобновление ее выполнения.

### 14.Виды наборов команд. Основные типы процессоров.

Набор команд — соглашение о предоставляемых архитектурой средствах программирования, а именно: определённых типах данных, инструкций, системы регистров, методов адресации, моделей памяти, способов обработки прерываний и исключений, методов ввода и вывода.

Основные типы процессоров:

**CISC -** есть традиционная архитектура, в которой ЦП использует микропрограммы для выполнения исчерпывающего набора команд

**RISC -** процессор, функционирующий с сокращенным набором команд. Так, в процессоре CISC для выполнения одной команды необходимо в большинстве случаев 10 и более тактов. Что же касается процессоров RISC, то они близки к тому, чтобы выполнять по одной команде в каждом такте

**MISC -** работающий с минимальным набором длинных команд и увеличенной разрядностью с идеей укладки нескольких команд в одно слово размером 128 бит. Оперируя с одним словом, процессор получил возможность обрабатывать сразу несколько команд. Это позволило использовать возросшую производительность компьютера и его возможность обрабатывать одновременно несколько потоков данных.

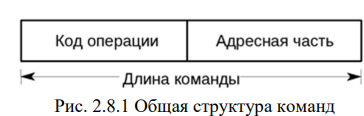
**Процессор VLIW —** процессор, работающий с системой команд сверхбольшой разрядности. Идея технологии VLIW заключается в том, что создается специальный компилятор планирования, который перед выполнением прикладной программы проводит ее анализ, и по множеству ветвей последовательности операций определяет группу команд, которые могут выполняться параллельно. Каждая такая группа образует одну сверхдлинную команду. Это позволяет решать две важные задачи. Во-первых, в течение одного такта выполнять группу коротких («обычных») команд. И, во-вторых, упростить структуру процессора.

### 15.Основные виды машинных операций.

Базовые команды:

* передача данных
* арифметические операции
* логические операции
* сдвиги двоичного кода влево и вправо
* ввод и вывод
* команды управления (переходы, обращения к подпрограмме)

### 16.Машинные команды. Структуры и форматы.



* Операционная часть – содержит код, который задает вид операции (сложение, умножение, передача и т.д.).
* Адресная часть – содержит информацию об адресах операндов и результата операции, а в некоторых случаях и следующей команды.
* Структура команды – определяется составом, назначением и расположением полей в команде.
* Формат команды – это ее структура с разметкой номеров разрядов, определяющих границы отдельных полей команды

### 17.Типы адресации. Способы адресации

Способы адресации:

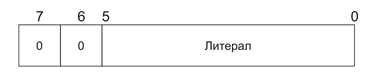
* **Подразумеваемый операнд** В команде не содержится явных указаний о самом операнде или его адресе. Операнд подразумевается и фактически задается кодом операции команды.
* **Подразумеваемый адрес** В команде не содержится явных указаний об адресе участвующего в операции операнда или адресе результата операции
* **Непосредсвенная адресация.** В команде содержится не адрес операнда, а непосредственно сам операнд Это способ уменьшения объема программы и занимаемой памяти, так как не требует операций обращения к памяти и самой ячейки памяти.
* **Прямая адресация.** Исполнительный адрес совпадает с адресной частью команды, т.е. адресный код совпадает с исполнительным адресом
* **Относительная адресация, или базирование** Исполнительный адрес (АИ) определяется как сумма адресного кода команды (АК) и некоторого числа АБ, называемого базовым адресом. АИ = АБ + АК
* **Регистровая адресация.** Это частный случай так называемой укороченной адресации, суть которой сводится к тому, что используется только небольшая группа фиксированных ячеек памяти с начальными (короткими) адресами
* **Косвенная адресация** Адресный код (АК) команды указывает адрес ячейки ОП, в которой находится исполнительный адрес (АИ) операнда или команды, т.е. это адрес адреса – АА
* **Автоинкрементная и автодекрементная адресации** *Автоинкрементная адресация*– сначала (при каждом обращении) содержимое регистра используется как адрес операнда, а затем получает приращение, равное числу байт в элементе массива, т.е. формируется адрес следующего элемента. *Автодекрементная адресация* – сначала содержимое соответствующего регистра уменьшается на число, равное числу байт в элементе массива, а затем используется как адрес операнда.
* **Стековая адресация** При операциях со стеком возможно безадресное задание операнда – команда не содержит адреса ячейки стека, а содержит только адрес (или он подразумевается) регистра или ячейки ОП, откуда слово загружается в стек или куда выгружается из стека.

### 18.Непосредственная, литеральная и прямая адресации.

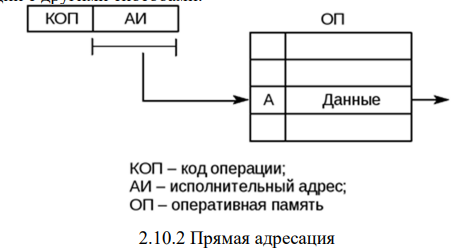
**Непосредсвенная адресация.** В команде содержится не адрес операнда, а непосредственно сам операнд Это способ уменьшения объема программы и занимаемой памяти, так как не требует операций обращения к памяти и самой ячейки памяти.

****

**Литеральная адресация** Литеральная адресация является особым специальным случаем короткой непосредственной адресации. Признаком этой адресации являются два нуля в старших (6 и 7) разрядах байта-спецификатора. В младших шести разрядах (с 0 по 5) байта-спецификатора вместо номера регистра и младших разрядов кода способа адресации помещается значение операнда. Таким образом, литеральная адресация (один способ адресации) использует как бы четыре режима адресации (с 0 по 3). Поле операнда для способа литеральной адресации имеет длину 1 байт и в нем в явном виде отсутствует указание регистра. Для способа литеральной адресации формат поля операнда имеет следующий вид

****

**Прямая адресация.** Исполнительный адрес совпадает с адресной частью команды, т.е. адресный код совпадает с исполнительным адресом



### 19.Косвенная адресация.

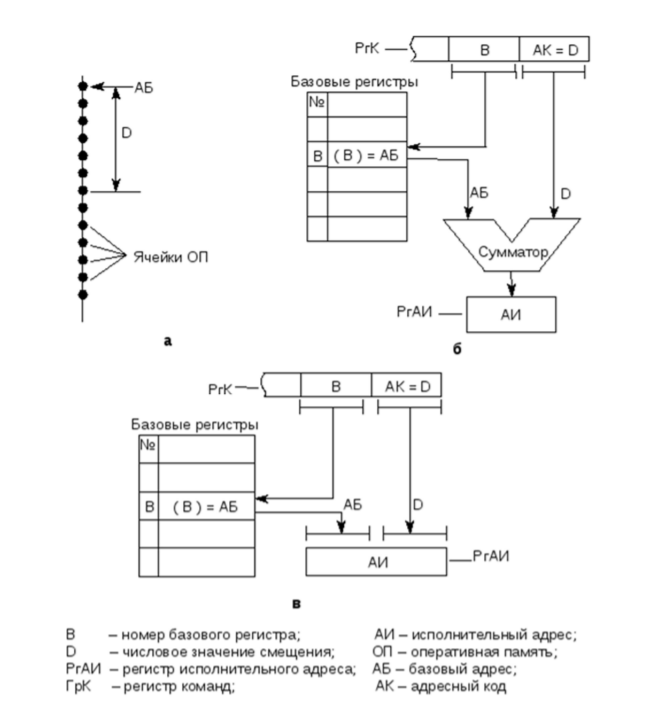
Адресный код (АК) команды указывает адрес ячейки ОП- оперативной памяти, в которой находится исполнительный адрес (АИ) операнда или команды, т.е. это адрес адреса – АА. Схема косвенной адресации представлена на рис. 2.10.5.

На косвенную адресацию указывает код операции (КОП) команды. Такая адресация позволяет микропроцессору адресоваться к ОП-оперативной память

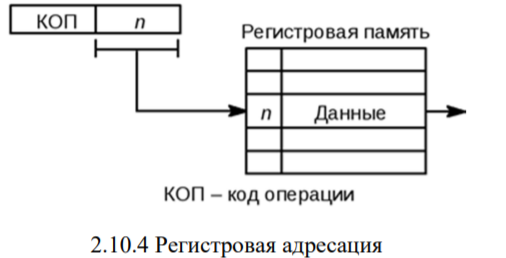
достаточно большого объема при небольшой длине адресного поля команды (адресного поля достаточно только для указания номеров нескольких внутренних регистров микропроцессора).

****

### 20.Относительная адресация.

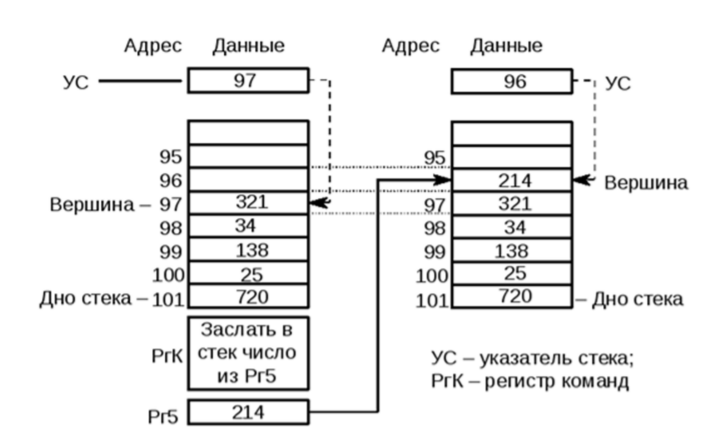
Исполнительный адрес (АИ) определяется как сумма адресного кода команды (АК) и некоторого числа АБ, называемого базовым адресом. АИ = АБ + АК (рис. 2.10.3).   


### 21. Регистровая адресация.

Это частный случай так называемой укороченной адресации, суть которой сводится к тому, что используется только небольшая группа фиксированных ячеек памяти с короткими адресами. Такая адресация используется только совместно с другими типами адресации. При использовании укороченной адресации длина команды существенно сокращается, так как используются только младшие разряды адресов.

В случае регистровой адресации (рис. 2.10.4) в качестве фиксированных ячеек с короткими адресами используются регистры внутренней памяти процессора, которых обычно немного. Поэтому разрядность АК также невелика. 2.10.4 Регистровая адресация Это, фактически, прямая адресация к сверхбыстрой памяти процессора. Достоинства данного способа адресации – укорочение команд, увеличение скорости выполнения операций. Недостаток – малое число адресов.

### 22.Стековая адресация.

Основной принцип работы стекового ЗУ-запоминающее устройство соответствует правилу: "последний пришел – первый ушел". Это правило реализуется автоматически. Поэтому при операциях со стеком возможно безадресное задание операнда – команда не содержит адреса ячейки стека, а содержит только адрес регистра или ячейки ОП, откуда слово загружается в стек или куда выгружается из стека. Стек может быть реализован как аппаратным путем, так и программно. В первом случае стек представляет собой одномерный массив регистров, связанных между собой разрядными цепями передачи данных. Обычно он снабжен счетчиком стека, по содержимому которого можно контролировать переполнение стека. Во втором случае стек организуется на последовательно расположенных ячейках ОП. Для его реализации требуется еще один регистр – указатель стека, в котором хранится адрес вершины, т.е. последней занятой ячейки ОП из массива ячеек, отведенных под стек. Стек является эффективным элементом архитектуры современных ЭВМ, позволяющим во многих случаях существенно повысить скорость обработки информации. В универсальных ЭВМ общего применения (таких как персональный компьютер) программисту в большинстве случаев доступен только программный стек. На рис. 2.10.7 приведена схема записи числа в "перевернутый" программный стек, который используется наиболее широко.

### 23.Конвейерная обработка команд.

Разработчики архитектуры компьютеров издавна прибегали к методам проектирования, известным под общим названием "совмещение операций", при котором аппаратура компьютера в любой момент времени выполняет одновременно более одной базовой операции. Этот общий метод включает два понятия: параллелизм и конвейеризацию. Хотя у них много общего и их зачастую трудно различать на практике, эти термины отражают два совершенно различных подхода. При параллелизме совмещение операций достигается путем воспроизведения в нескольких **копиях аппаратной структуры**. Высокая производительность достигается за счет одновременной работы всех элементов структур, осуществляющих решение различных частей задачи.

Конвейеризация (или конвейерная обработка) в общем случае основана на разделении подлежащей исполнению функции на более мелкие части, называемые ступенями, и выделении для каждой из них отдельного блока аппаратуры. Так обработку любой машинной команды можно разделить на несколько этапов (несколько ступеней), организовав передачу данных от одного этапа к следующему. При этом конвейерную обработку можно использовать для совмещения этапов выполнения разных команд. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько команд. Конвейерная обработка такого рода широко применяется во всех современных быстродействующих процессорах.

Уровни конвейеризации:

1. Макроконвейер – конвейеризация на уровне процессоров
2. Конвейер команд - конвейеризация команд процессора
3. Конвейер арифметический - конвейеризация на уровне выполнения команд процессора

### 24.Конфликты при конвейерной обработке команд.

1. Конфликты по ресурсам - возникают из-за конфликтов по ресурсам, когда аппаратные средства не могут поддерживать все возможные комбинации команд в режиме одновременного выполнения с совмещением.
2. Конфликты по данным, возникающие в случае, когда выполнение одной команды зависит от результата выполнения предыдущей команды.
3. Конфликты по управлению, которые возникают при конвейеризации команд переходов и других команд, которые изменяют значение счетчика команд.

Конфликты в конвейере приводят к необходимости приостановки выполнения команд (pipeline stall). Обычно в простейших конвейерах, если приостанавливается какая-либо команда, то все следующие за ней команды также приостанавливаются. Команды, предшествующие приостановленной, могут продолжать выполняться, но во время приостановки не выбирается ни одна новая команда.

### 25.Организация памяти ЭВМ. Основные определения и характеристики.

В любой ВМ, вне зависимости от ее архитектуры, программы и данные хранятся в памяти. Функции памяти обеспечиваются запоминающими устройствами (ЗУ), предназначенными для фиксации, хранения и выдачи информации в процессе работы ВМ. Процесс фиксации информации в ЗУ называется записью, процесс выдачи информации — чтением или считыванием, а совместно их определяют как процессы обращения к ЗУ.

Перечень основных характеристик, которые необходимо учитывать, рассматривая конкретный вид ЗУ, включает в себя:

* место расположения;
* емкость;
* стоимость;
* единица пересылки;
* метод доступа;
* быстродействие;
* физический тип;
* физические особенности.

По **месту расположения** ЗУ разделяют на процессорные, внутренние и внешние. Наиболее скоростные виды памяти (регистры, кэш-память первого уровня) обычно размещают на общем кристалле с центральным процессором, а регистры общего назначения вообще считаются частью ЦП. Вторую группу (внутреннюю память) образуют ЗУ, расположенные на системной плате. К внутренней памяти относят память, а также кэш-память второго и последующих уровней (кэш-память второго уровня может также размещаться на кристалле процессора). Медленные ЗУ большой емкости (магнитные и оптические диски, магнитные ленты) называют внешней памятью, поскольку к ядру ВМ они подключаются аналогично устройствам ввода/вывода.

**Емкость** ЗУ характеризуют числом битов либо байтов, которое может храниться в запоминающем устройстве. На практике применяются более крупные единицы, а для их обозначения к словам «бит» или «байт» добавляют приставки кило, мега, гига, тера, пета, экза (kilo, mega, giga, tera, peta, exa). Стандартно эти приставки означают умножение основной единицы измерений на 103 , 106 , 109 , 1012; 1018 соответственно. В вычислительной технике, ориентированной на двоичную систему счисления, они соответствуют значениям достаточно близким к стандартным, но представляющим собой целую степень числа 2, то есть 2^10, 2^20, 2^30, 2^50 ,2^60 . Во избежание разночтений, в последнее время ведущие международные организации по стандартизации, например IEEE (Institute of Electrical and Electron'\* Engineers), предлагают ввести новые обозначения, добавив к основной пристав слово binary (бинарный): kilobinary, megabinary, gigabinary, terabinary, petabinar exabinary. В результате вместо термина «килобайт» предлагается термин «кибибайт», вместо «мегабайт» — «мебибайт» и т. д. - 2 - Для обозначения новых единиц предполагаются сокращения: Ki, Mi, Gi, Ti, Pi и Ei [133].

Важной характеристикой ЗУ является **единица пересылки**. Для основной памяти (ОП) единица пересылки определяется шириной шины данных, то есть количеством битов, передаваемых по линиям шины параллельно. Обычно единица пересылки равна длине слова, но не обязательно. Применительно к внешней памяти данные часто передаются единицами, превышающими размер слова, и такие единицы называются блоками.

При оценке быстродействия необходимо учитывать применяемый в данном типе ЗУ **метод доступа к данным**. Различают четыре основных метода доступа:

* ***Последовательный доступ***. ЗУ с последовательным доступом ориентировано на хранение информации в виде последовательности блоков данных, называемых записями. Для доступа к нужному элементу (слову или байту) необходимо прочитать все предшествующие ему данные. Время доступа зависит от положения требуемой записи в последовательности записей на носителе информации и позиции элемента внутри данной записи. Примером может служить ЗУ на магнитной ленте.
* ***Прямой доступ***. Каждая запись имеет уникальный адрес, отражающий ее физическое размещение на носителе информации. Обращение осуществляется как адресный доступ к началу записи, с последующим последовательным доступом к определенной единице информации внутри записи. В результате время доступа к определенной позиции является величиной переменной. Такой режим характерен для магнитных дисков.
* ***Произвольный доступ***. Каждая ячейка памяти имеет уникальный физический адрес. Обращение к любой ячейке занимает одно и то же время и может производиться в произвольной очередности. Примером могут служить запоминающие устройства основной памяти.
* ***Ассоциативный доступ***. Этот вид доступа позволяет выполнять поиск ячеек содержащих такую информацию, в которой значение отдельных битов совпадает с состоянием одноименных битов в заданном образце. Сравнение осуществляется параллельно для всех ячеек памяти, независимо от ее емкости ассоциативному принципу построены некоторые блоки кэш-памяти.

**Быстродействие** ЗУ является одним из важнейших его показателей. Для количественной оценки быстродействия обычно используют три параметра:

***Время доступа (Тд)***. Для памяти с произвольным доступом оно соответствует интервалу времени от момента поступления адреса до момента, когда данные заносятся в память или становятся доступными. В ЗУ с подвижным носителем информации — это время, затрачиваемое на установку головки записи/считывания (или носителя) в нужную позицию.

***Длительность цикла памяти или период обращения (Тц)***. Понятие применяется к памяти с произвольным доступом, для которой оно означает минимальное время между двумя последовательными обращениями к памяти. Период обращения включает в себя время доступа плюс некоторое дополнительное время. Дополнительное время может требоваться для - 3 - затухания сигналов на линиях, а в некоторых типах ЗУ, где считывание информации приводит к ее разрушению, — для восстановления считанной информации.

***Скорость передачи***. Это скорость, с которой данные могут передаваться в память или из нее. Для памяти с произвольным доступом она равна 1/Гц. Для других видов памяти скорость передачи определяется соотношением: где TN — среднее время считывания или записи N битов; ТА — среднее время доступа; R — скорость пересылки в битах в секунду.

Говоря о **физическом типе запоминающего устройства**, необходимо упомянуть три наиболее распространенных технологии ЗУ — это полупроводниковая память, память с магнитным носителем информации, используемая в магнитных дисках и лентах, и память с оптическим носителем — оптические диски.

В зависимости от примененной технологии следует учитывать и **ряд физических особенностей ЗУ**, например энергозависимость. В энергозависимой памяти информация может быть искажена или потеряна при отключении источника питания. В энергонезависимых ЗУ записанная информация сохраняется и при отключении питающего напряжения. Магнитная и оптическая память — энергонезависимы. Полупроводниковая память может быть как энергозависимой, так и нет, зависимости от ее типа. Помимо энергозависимости нужно учитывать, приводит и считывание информации к ее разрушению.

**Стоимость** ЗУ принято оценивать отношением общей стоимости ЗУ к его емкости в битах, то есть стоимостью хранения одного бита информации.

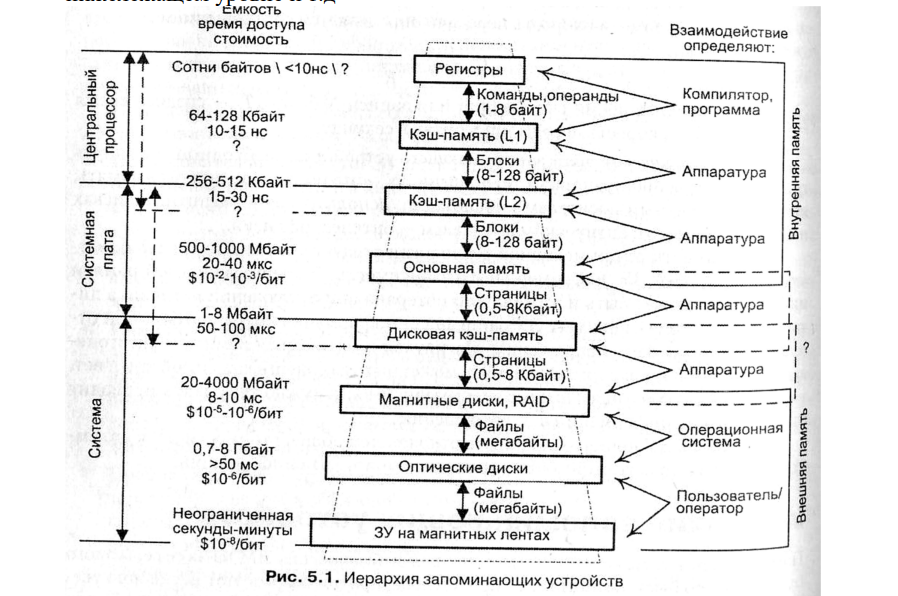
### 26.Иерархическая система памяти.

Часто выделяют 4 основных (укрупнённых) уровня иерархии:[[1]](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8#cite_note-toyzee-1)

1. Внутренняя память процессора ([регистры](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0), организованные в [регистровый файл](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) и [кэш процессора](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0)).
2. [ОЗУ](https://ru.wikipedia.org/wiki/%D0%9E%D0%97%D0%A3) системы ([RAM](https://youtu.be/S3ePt_3g3Hg)) и вспомогательных карт памяти.
3. Накопители с «горячим» доступом (On-line mass storage) — или вторичная компьютерная память. [Жесткие диски](https://ru.wikipedia.org/wiki/%D0%96%D1%91%D1%81%D1%82%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA) и [твердотельные накопители](https://ru.wikipedia.org/wiki/SSD), не требующие длительных (секунды и больше) действий для начала получения данных.
4. Накопители, требующие переключения носителей (Off-line bulk storage) — или третичная память. Сюда относятся [магнитные ленты](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BD%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D0%BD%D0%B0%D0%BA%D0%BE%D0%BF%D0%B8%D1%82%D0%B5%D0%BB%D1%8C), [ленточные](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BD%D1%82%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0) и [дисковые библиотеки](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), требующие длительной перемотки либо механического (или ручного) переключения носителей информации.

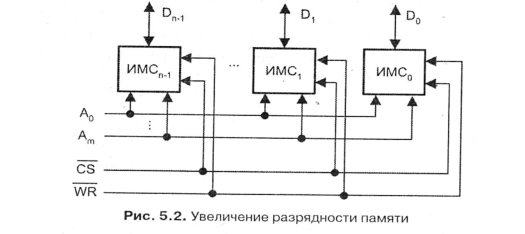
В большинстве современных [ПК](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) используется следующая иерархия памяти:

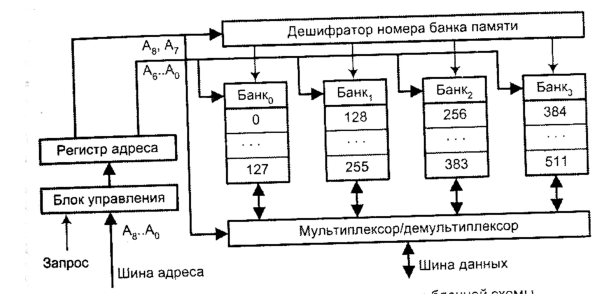
1. [Регистры процессора](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D1%8B_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0), организованные в [регистровый файл](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) — наиболее быстрый доступ (порядка 1 такта), но размером лишь в несколько сотен или, редко, тысяч байт.
2. [Кэш процессора](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0) 1го уровня (L1) — время доступа порядка нескольких тактов, размером в десятки килобайт
3. Кэш процессора 2го уровня (L2) — большее время доступа (от 2 до 10 раз медленнее L1), около полумегабайта или более
4. Кэш процессора 3го уровня (L3) — время доступа около сотни тактов, размером от нескольких мегабайт до сотен
5. Кэш процессора 4го уровня (L4) — время доступа несколько сотен тактов, размером одну-несколько сотен мегабайт. Применялся в процессорах Intel 5го поколения
6. [ОЗУ](https://ru.wikipedia.org/wiki/%D0%9E%D0%97%D0%A3) системы — время доступа от сотен до, возможно, тысячи тактов, но огромные размеры, от нескольких гигабайт до нескольких терабайт. Время доступа к ОЗУ может варьироваться для разных его частей в случае комплексов класса [NUMA](https://ru.wikipedia.org/wiki/NUMA) (с неоднородным доступом в память)
7. [Дисковое хранилище](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2) — многие миллионы тактов, если данные не были закэшированны или забуферизованны заранее, размеры до нескольких терабайт
8. Третичная память — задержки до нескольких секунд или минут, но практически неограниченные объёмы ([ленточные библиотеки](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BD%D1%82%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0)).

****

### 27.Блочная организация основной памяти.

Емкость основной памяти современных ВМ слишком велика, чтобы ее можно было реализовать на базе единственной интегральной микросхемы (ИМС). Необходимость объединения нескольких ИМС ЗУ возникает также, когда разрядность ячеек в микросхеме ЗУ меньше разрядности слов ВМ.

Увеличение разрядности ЗУ реализуется за счет объединения адресных входов объединяемых ИМС ЗУ. Информационные входы и выходы микросхем являются входами и выходами модуля ЗУ увеличенной разрядности (рис. 9.4.1). Полученную совокупность микросхем называют модулем памяти. Модулем можно считать и единственную микросхему, если она уже имеет нужную разрядность. Один или несколько модулей образуют банк памяти. 9.4.1 Увеличение разрядности памяти

Для получения требуемой емкости ЗУ нужно определенным образом объединить несколько банков памяти меньшей емкости. В общем случае основная память ВМ практически всегда имеет блочную структуру, то есть содержит несколько банков. - 9 - При использовании блочной памяти, состоящей из В банков, адрес ячейки А преобразуется в пару (b, w), где b — номер банка, w — адрес ячейки внутри банка. Известны три схемы распределения разрядов адреса А между b и w: - блочная (номер банка b определяет старшие разряды адреса); - циклическая (b-A modВ; w = A div B); - блочно-циклическая (комбинация двух предыдущих схем). Рассмотрение основных структур блочной ОП будем проводить на примере памяти емкостью 512 слов (29 ), построенной из четырех банков по 128 слов в каждом. Типовая структура памяти, организованная в соответствии с блочной структурой, показана на рис. 9.4.2. Рис 9.4.2 Структура основной памяти на основе блочной схемы

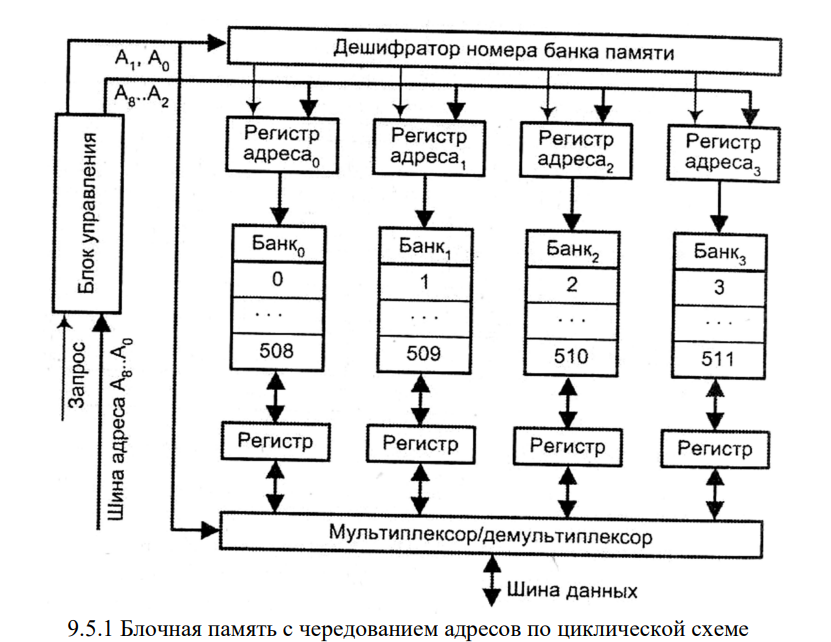
Адресное пространство памяти разбито на группы последовательных адресов, и каждая такая группа обеспечивается отдельным банком памяти. Для обращения к ОП используется 9-разрядный адрес, семь младших разрядов которого (А6-А0) поступают параллельно на все банки памяти и выбирают в каждом из них одну ячейку. Два старших разряда адреса (А8, А7) содержат номер банка. Выбор банка обеспечивается либо с помощью дешифратора номера банка памяти, либо путем мультиплексирования информации (на рис. 9.4.2 показаны оба варианта). В функциональном отношении такая ОП может рассматриваться как единое ЗУ, емкость которого равна суммарной емкости составляющих, а быстродействие — быстродействию отдельного банка**.**

### 28. Память с расслоением обращений.

Для одновременного доступа ко многим банкам памяти используется *расслоение памяти.* В ее основе лежит так называемое чередование адресов (address interleaving), заключающееся в изменении системы распределения адресов между банками памяти.

Прием чередования адресов базируется на ранее рассмотренном свойстве локальности по обращению, согласно которому последовательный доступ в память обычно производится к ячейкам, имеющим смежные адреса. Иными словами, если в данный момент выполняется обращение к ячейке с адресом 5, то следующее обращение, вероятнее всего, будет к ячейке с адресом 6, затем 7 и т. д. Чередование адресов обеспечивается за счет циклического разбиения адреса.

Поскольку в каждом такте на шине адреса может присутствовать адрес только одной ячейки, параллельное обращение к нескольким банкам невозможно, однако оно может быть организовано со сдвигом на один такт. Адрес ячейки запоминается в индивидуальном регистре адреса, и дальнейшие операции по доступу к ячейке в каждом банке протекают независимо. При большом количестве банков среднее время доступа к ОП сокращается почти в В раз (В — количество банков), но при условии, что ячейки, к которым производится последовательное обращение относятся к разным банкам. Если же запросы к одному и тому же банку следуют друг за другом, каждый следующий запрос должен ожидать завершения обслуживания предыдущего. Такая ситуация называется конфликтом по доступу. При частом возникновении конфликтов по доступу метод становится неэффективным



### 29. Ассоциативная память.

Зачастую значительно удобнее искать информацию не по адресу, а опираясь на какой-нибудь характерный признак, содержащийся в самой информации. Такой принцип лежит в основе ЗУ, известного как ассоциативное запоминающее устройство (АЗУ).

Ассоциативное ЗУ — это устройство, способное хранить информацию, сравнивать ее с некоторым заданным образцом и указывать на их соответствие или несоответствие друг другу. Признак, по которому производится поиск информации, будем называть ассоциативным признаком, а кодовую комбинацию, выступающую в роли образца для поиска, — признаком поиска. Ассоциативный признак может быть частью искомой информации или дополнительно придаваться ей. В последнем случае его принято называть тегом или ярлыком.



Один из вариантов построения ассоциативной памяти показан на рис. 10.1 АЗУ включает в себя:

- запоминающий массив для хранения N m -разрядных слов, в каждом из которых несколько младших разрядов занимает служебная информация;

- регистр ассоциативного признака, куда помещается код искомой информации (признак поиска). Разрядность регистра k обычно меньше длины слова т;

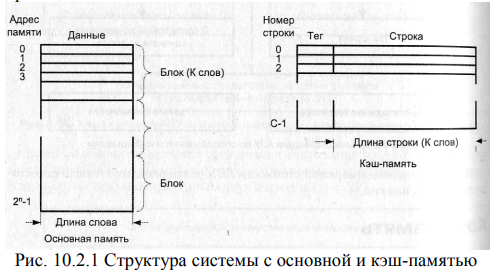
- схемы совпадения, используемые для параллельного сравнения каждого бита всех хранимых слов с соответствующим битом признака поиска и выработки сигналов совпадения;   
- регистр совпадений, где каждой ячейке запоминающего массива соответствует один разряд, в который заносится единица, если все разряды соответствующей ячейки совпали с одноименными разрядами признака поиска;

- регистр маски, позволяющий запретить сравнение определенных битов;

- комбинационную схему, которая на основании анализа содержимого регистра совпадений формирует сигналы, характеризующие результаты поиска информации.

### 30.Организация кэш-памяти.

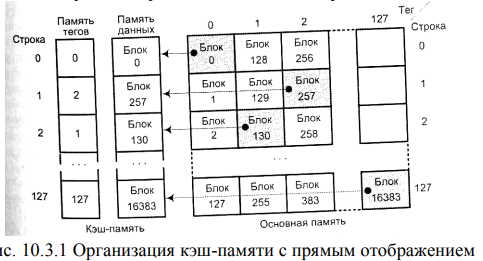
Двухуровневая память, когда между ОП и процессором размещается небольшая, но быстродействующая буферная память. В процессе работы такой системы в буферную память копируются те участки ОП, к которым производится обращение со стороны процессора. В общепринятой терминологии — производится отображение участков ОП на буферную память. Выигрыш достигается за счет свойства локальности — если отобразить участок ОП в более быстродействующую буферную память и переадресовать на нее все обращения в пределах скопированного участка, можно добиться существенного повышения производительности ВМ.



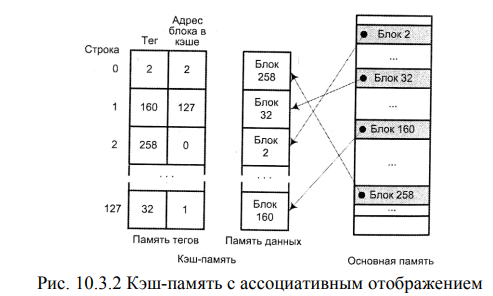
На рис. 10.2.1 приведена структура системы с основной кэш-памятью. ОП состоит из 2n адресуемых слов, где каждое слово имеет уникальный n разрядный адрес. При взаимодействии с кэшем эта память рассматривается как М блоков фиксированной длины по К слов в каждом (М = 2п /К). Кэшпамять состоит из С блоков аналогичного размера (блоки в кэш-памяти принято называть строками), причем их число значительно меньше числа блоков в основной памяти (С<<М). При считывании слова из какого-либо блока ОП этот блок копируется в одну из строк кэша. Поскольку число блоков ОП больше числа строк, отдельная строка не может быть выделена постоянно одному и тому же блоку ОП. По этой причине каждой строке кэшпамяти соответствует тег (признак), содержащий сведения о том, копия какого блока ОП в данный момент хранится в данной строке. В качестве тега обычно используется часть адреса ОП.

### 31.Способы отображения оперативной памяти на кэш-память.

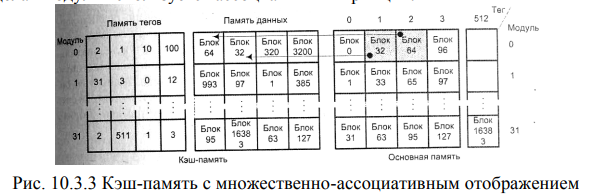
При **прямом отображении** адрес строки i кэш-памяти, на которую может быть отображен блок j из ОП, однозначно определяется выражением: i=j mod т, где т — общее число строк в кэш-памяти (рис. 10.3.1).



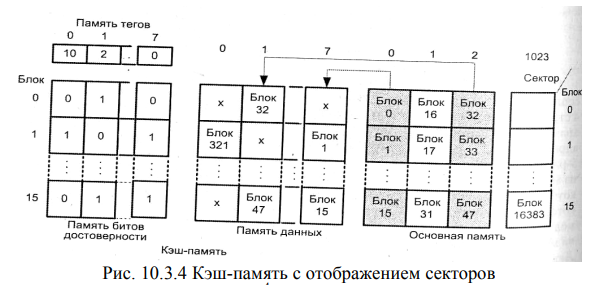
**Полностью ассоциативное отображение** позволяет преодолеть недостаток прямого, разрешая загрузку любого блока ОП в любую строку кэш-памяти. Логика управления кэш-памяти выделяет в адресе ОП два поля: поле тега и поле слова. Поле тега совпадает с адресом блока основной памяти. Для проверки наличия копии блока в кэш-памяти логика управления кэша должна одновременно проверить теги всех строк на совпадение с полем тега адреса. Этому требованию наилучшим образом отвечает ассоциативная память, то есть тег должен храниться в ассоциативной памяти тегов кэша. Логика работы такой кэш-памяти иллюстрируется рис. 10.3.2.



**Множественно-ассоциативное отображение** относится к группе методов частично-ассоциативного отображения. Кэш-память (как тегов, так и данных) разбивается на v подмножеств (в дальнейшем будем называть такие подмножества модулями), каждое из которых содержит k строк (принято говорить, что модуль имеет k входов). Зависимость между модулем и блоками ОП такая же, как и при прямом отображении: на строки, входящие в модуль, могут быть отображены только вполне определенные блоки основной памяти, в соответствии с соотношением i =j mod v, где j — адрес блока ОП. В то же время размещение блоков по строкам модуля — произвольное, и для поиска нужной строки в пределах модуля используется ассоциативный принцип.



**Частично-ассоциативное отображение**. Согласно данному методу, основная память разбивается на секторы, состоящие из фиксированного числа последовательных блоков. Кэш-память также разбивается на секторы, содержащие такое же число строк. Расположение блоков в секторе ОП и секторе кэш-памяти полностью совпадает. Отображение сектора на кэш-память осуществляется ассоциативно, то есть любой сектор из ОП может быть помещен в любой сектор кэш-памяти



### 32. Алгоритмы замещения и согласования информации в кэш-памяти.

**Алгоритмы замещения**

**Алгоритм замещения на основе наиболее давнего использования** (LRU — Least Recently Used), при котором замещается та строка кэш-памяти, к которой дольше всего не было обращения. Наиболее известны два способа аппаратурной реализации этого алгоритма.

В первом из них с каждой строкой кэш-памяти ассоциируют счетчик. К содержимому всех счетчиков через определенные интервалы времени добавляется единица. При обращении к строке ее счетчик обнуляется. Таким образом, наибольшее число будет в счетчике той строки, к которой дольше всего не было обращений, и эта строка — первый кандидат на замещение.

Второй способ реализуется с помощью очереди, куда в порядке заполнения строк кэш-памяти заносятся ссылки на эти строки. При каждом обращении к строке ссылка на нее перемещается в конец очереди. В итоге первой в очереди каждый раз оказывается ссылка на строку, к которой дольше всего не было обращений. Именно эта строка прежде всего и заменяется.

**Алгоритм, работающий по принципу «первый вошел, первый вышел»** (FIFO — First In First Out). Здесь заменяется строка, дольше всего находившаяся в кэш-памяти. Алгоритм легко реализуется с помощью очереди.

**Замена наименее часто использовавшейся строки** (Lr и Least Frequently Used). Заменяется та строка в кэш-памяти, к которой было меньше всего обращений

**Произвольный выбор строки для замены**. Замещаемая строка выбирается случайным образом. Реализовано это может быть, например с помощью счетчика, содержимое которого увеличивается на единицу с каждым тактовым импульсом, вне зависимости от того, имело место попадание или промах. Значение в счетчике определяет заменяемую строку в полностью ассоциативной кэш-памяти или строку в пределах модуля для множественно-ассоциативной кэш-памяти

**Алгоритмы согласования**

**Метод сквозной записи**. Прежде всего обновляется слово, хранящееся в основной памяти. Если в кэш-памяти существует копия этого слова, то она также обновляется. Если же в кэш-памяти отсутствует нужная копия, то либо из основной памяти в кэш-память пересылается блок, содержащий обновленное слово (сквозная запись с отображением), либо этого не делается (сквозная запись без отображения).

**Метод буферированной сквозной записи**. Информация сначала записывается в кэшпамять и в специальный буфер, работающий по схеме FIFO. Запись в основную память производится уже из буфера, а процессор, не дожидаясь ее окончания, может сразу же продолжать свою работу.

**Метод флаговой обратной записи.** Когда в какой-то строке кэша производится изменение, устанавливается связанный с этой строкой бит изменения (флажок). При замещении строка из кэш- памяти переписывается в ОП только тогда, когда ее флажок установлен в 1.

### 33.Виды внешней памяти.

Наиболее распространенные ВИДЫ таких ЗУ — это магнитные и оптические диски и магнитоленточные устройства.

* 1. **Магнитные и оптические диски**

Информация в ЗУ на магнитных дисках (МД) хранится на плоских металлических или пластиковых пластинах (дисках), покрытых магнитным материалом. Данные записываются и считываются с диска с помощью электромагнитной катушки, называемой головкой считывания/записи.

Данные на диске организованы в виде набора концентрических окружностей, называемых дорожками. Каждая из них имеет ту же ширину, что и головка. Соседние дорожки разделены промежутками.

Обмен информацией с МД осуществляется блоками. Размер блока обычно меньше емкости дорожки, и данные на дорожке хранятся в виде последовательных областей — секторов, разделенных между собой промежутками. Размер сектора равен минимальному размеру блока.

* 1. **Дисковая кэш-память**

Дисковая кэш-память представляет собой память с произвольным доступом, «размещенную» между дисками и ОП. Емкость такой памяти обычно достаточно велика — от 8 Мбайт и более. Пересылка информации между дисками и основной памятью организуется контроллером дисковой кэш-памяти. Изготавливается дисковая кэш-память на базе таких же полупроводниковых ЗУ что и основная память.

В дисковой кэш-памяти хранятся блоки информации, которые с большой вероятностью будут востребованы в ближайшем будущем.

В качестве единицы пересылки может выступать сектор, несколько секторов, а так же одна или несколько дорожек диска.

* 1. **Массивы магнитных дисков с избыточностью**

### 34.Понятие виртуальной памяти.

Виртуальная память — это подход к управлению памятью компьютером, который скрывает физическую память (в различных формах, таких как: оперативная память, ПЗУ или жесткие диски) за единым интерфейсом, позволяя создавать программы, которые работают с ними как с единым непрерывным массивом памяти с произвольным доступом.

Виды адресов памяти: · физический - адрес аппаратной ячейки памяти · логический - виртуальный адрес, которым оперирует приложение

### 35.Страничная и сегментно-страничная организация памяти

**Страничная организация памяти**

Целям преобразования виртуальных адресов в физические служит страничная организация памяти. Ее идея состоит в разбиении программы на части равной величины, называемые страницами. Размер страницы обычно выбирают в предела-4-8 Кбайт, но так, чтобы он был кратен емкости одного сектора магнитного диска. Виртуальное и физическое адресные пространства разбиваются на блоки размером в страницу. Блок основной памяти, соответствующий странице, часто называю страничным кадром или фреймом (page frame). Страницам виртуальной и физической памяти присваивают номера. Процесс доступа к данным по их виртуальному адресу иллюстрирует рис. 17.2.1.



Центральный процессор обращается к ячейке, указав ее виртуальный адрес , состоящий из номера виртуальной страницы и смещения относительно ее начала. Этот адрес поступает в систему преобразования адресов , с целью получения из него физического адреса ячейки в основной памяти . Поскольку смещение в виртуальном и физическом адресе одинаковое, преобразованию подвергается лишь номер страницы. Если преобразователь обнаруживает, что нужная физическая страница отсутствует в основной памяти (произошел промах или страничный сбой), то нужная страница считывается из внешней памяти и заносится в ОП .

**Сегментно-страничная организация памяти**

При страничной организации предполагается, что виртуальная память — это непрерывный массив со сквозной нумерацией слов, что не всегда можно признать оптимальным. Обычно программа состоит из нескольких частей — кодовой, информационной и стековой. Так как заранее неизвестны длины этих составляющих, то удобно, чтобы при программировании каждая из них имела собственную нумерацию слов, отсчитываемых с нуля. Для этого организуют систему сегментированной памяти, выделяя в виртуальном пространстве независимые линейные пространства переменной длины, называемые сегментами.

Каждый сегмент представляет собой отдельную логическую единицу информации, содержащую совокупность данных или программный код и расположенную в адресном пространстве пользователя. В каждом сегменте устанавливается своя собственная нумерация слов, начиная с нуля. Виртуальная память также разбивается на сегменты, с независимой адресацией слов внутри сегмента. Каждой составляющей программы выделяется сегмент памяти. Виртуальный адрес определяется номером сегмента и адресом внутри сегмента. Для преобразования виртуального адреса в физический используется специальная сегментная таблица.

Недостатком такого подхода является то, что неодинаковый размер сегментов приводит к неэффективному использованию ОП. Так, если ОП заполнена, то при замещении одного из сегментов требуется вытеснить такой, размер которого равен или больше размера нового. При многократном повторе подобных действии в ОП остается множество свободных участков, недостаточных по размеру для загрузки полного сегмента. Решением проблемы служит сегментно-страничная организация памяти. В ней размер сегмента выбирается не произвольно, а задается кратным размеру страницы. Сегмент может содержать то или иное, но обязательно целое число страниц, даже если одна из страниц заполнена частично. Возникает определенная иерархия в организации доступа к данным, состоящая из трех ступеней: сегмент > страница > слово. Этой структуре соответствует иерархия таблица служащих для перевода виртуальных адресов в физические. В сегментной таблице программы перечисляются все сегменты данной программы с - 6 - указанием начальных адресов СТ, относящихся к каждому сегменту. Количество страничных таблиц равно числу сегментов и любая из них определяет расположение каждой из страниц сегмента в памяти, которые могут располагаться не подряд — часть страниц может находиться в ОП, остальные — во внешней памяти. Структуру виртуального адреса и процесс преобразования его в физический адрес иллюстрирует рис. 17.3.1.



### 36.Массивы дисков с избыточностью (RAID-массивы).

RAID (англ. Redundant Array of Independent Disks — избыточный массив независимых дисков) — технология виртуализации данных, которая объединяет несколько дисков в логический элемент для повышения производительности. Соответственно, минимальное количество требуемых дисков — 2, но может потребоваться и больше. Всё зависит от того, какой именно массив вам нужен и для чего.

RAID 0

Принцип работы — striping (чередование). Массив при котором информация разбивается на одинаковые по длине блоки, а затем записывается поочерёдно на каждый диск в структуре. Основное предназначение такой системы — фактическое увеличение производительности в 2 раза, при этом вам будет доступен полный объем всех дисков.

Преимущества

Высокая производительность

Недостатки

Низкая надежность

Сложность подбора дисков с одинаковыми характеристиками

RAID 1

Принцип работы — mirroring («зеркалирование»). Самая простая система RAID-массивов из всех возможных. Представляет собой параллельную запись информации с основного диска на другие — дублирующие. Производительность при этом никак не изменяется. Имеет широкое применение в серверном обслуживании, потому что в случае выхода из строя одного из накопителей, все продублированные данные остаются на других носителях. При этом вам будет доступен объем лишь одного винчестера.

RAID 10  
Все остальные виды массивов являются различными вариациями первых двух. RAID 10 — совмещает в себе всё самое лучшее из RAID 1 и RAID 0. Вам потребуется минимум 4 носителя, и их количество всегда должно быть четным. В данном массиве вы получаете высокую производительность и высокую надежность. Однако, как в случае и с RAID 1, вам будет доступна лишь половина от общего объема всей системы.

RAID 5  
Сильно схож по своему принципу работы с RAID 1. Только вам теперь потребуется минимум 3 накопителя, на одном из которых будет храниться продублированная информация. В этом случае вам будет доступен практически весь объем в системе, кроме одного диска с данными под восстановление. Кроме того, увеличится и производительность, но не в несколько раз, как в случае с RAID 0. Основное отличие RAID 5 от RAID 10 — это уровень надежности и доступный объем. Данный массив предназначен для более специфических задач, когда вместе собрано огромное количество дисков.

### 37.Организация и виды шин данных.

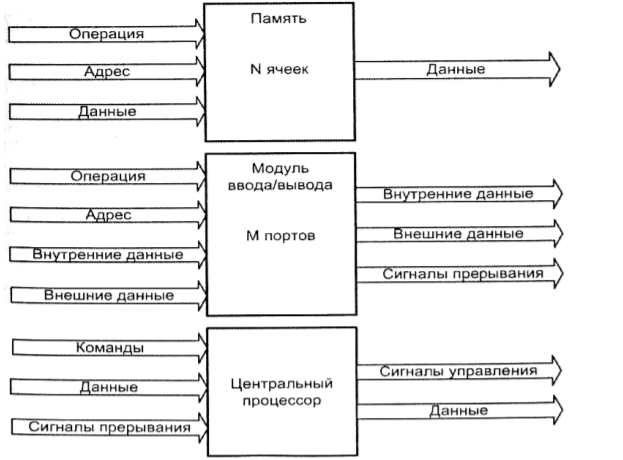
Совокупность трактов, объединяющих между собой основные устройства ВМ (центральный процессор, память и модули ввода/вывода), образует структуру взаимосвязей вычислительной машины. Структура взаимосвязей должна обеспечивать обмен информацией между:

- центральным процессором и памятью;

- центральным процессором и модулями ввода/вывода;

- памятью и модулями ввода/вывода.

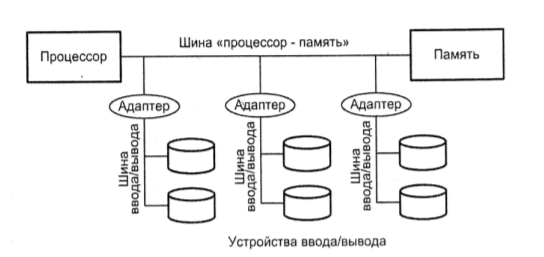
Информационные потоки, характерные для основных устройств ВМ, показаны на рис. 12.1.1. Рис. 12.1.1

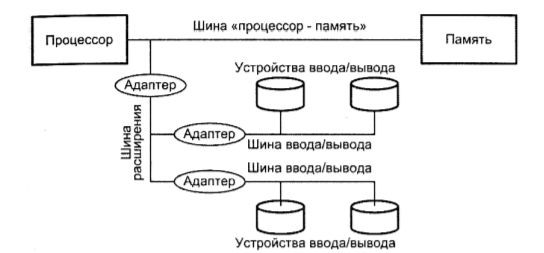
  
Информационные потоки в вычислительной машине С развитием вычислительной техники менялась и структура взаимосвязей устройств ВМ (рис. 12.1.2).

Виды:

1. Системная шина
2. Шина ввода/вывода
3. Шина процессор-память

### 38.Иерархия шин данных.

Если к шине подключено большое число устройств, ее пропускная способность падает, поскольку слишком частая передача прав управления шиной от одного устройства к другому приводит к ощутимым задержкам. По этой причине во многих ВМ предпочтение отдается использованию нескольких шин, образующих определенную иерархию. Сначала рассмотрим ВМ с одной шиной.   
  
Вычислительная машина с одной шиной   
  
В структурах взаимосвязей с одной шиной имеется одна системная шина, обеспечивающая обмен информацией между процессором и памятью, а также между УВВ с одной стороны, и процессором либо памятью — с другой (рис. 12.5.1). Рис. 12.5.1 Структура взаимосвязей с одной шиной Для такого подхода характерны простота и низкая стоимость. Однако одношинная организация не в состоянии обеспечить высокие интенсивность и скорость транзакций, причем «узким местом» становится именно шина.  
  
 Вычислительная машина с двумя видами шин Хотя контроллеры устройств ввода/вывода (УВВ) могут быть подсоединены непосредственно к системной шине, больший эффект достигается применением одной или нескольких шин ввода/вывода (рис. 12.5.2).  
  
 УВВ подключаются к шинам ввода/вывода, которые берут на себя основной трафик, не связанный с выходом на процессор или память. Адаптеры шин обеспечивают буферизацию данных при их пересылке между системной шиной и контроллерами УВВ. Это позволяет ВМ поддерживать работу множества устройств ввода/вывода и одновременно «развязать» обмен информацией по тракту процессор-память и обмен информацией с УВВ. - 6 - Рис. 12.5.2. Структура взаимосвязей с двумя видами шин Подобная схема существенно снижает нагрузку на скоростную шину «процесс память» и способствует повышению общей производительности ВМ.

Вычислительная машина с тремя видами шин  


Для подключения быстродействующих периферийных устройств в систему шин может быть добавлена высокоскоростная шина расширения (рис. 12.5.3). Рис. 12.5.3 Структура взаимосвязей с тремя видами шин Шины ввода/вывода подключаются к шине расширения, а уже с нее через адаптер к шине «процессор-память». Схема еще более снижает нагрузку на шину «процессор-память». Такую организацию шин называют архитектурой с «пристройкой» (mezzanine architecture).

### 39.Схемы приоритетов, арбитража и опроса шин данных.

В реальных системах на роль ведущего вправе одновременно претендовать сразу несколько из подключенных к шине устройств, однако управлять шиной в каждый момент времени может только одно из них. Чтобы исключить конфликты, шина должна предусматривать определенные механизмы арбитража запросов и правила предоставления шины одному из запросивших устройств. Решение обычно принимается на основе приоритетов.

Каждому потенциальному ведущему присваивается определенный уровень приоритета, который может оставаться неизменным (**статический или фиксированный приоритет**) либо изменяться по какому-либо алгоритму (**динамический приоритет**).

Основной недостаток статических приоритетов в том, что устройства, имеющие высокий приоритет, в состоянии полностью блокировать доступ к шине устройств с низким уровнем приоритета. Системы с динамическими приоритетом дают шанс каждому из запросивших устройств рано или поздно получить право на управление шиной, то есть в таких системах реализуется принцип равнодоступности.

Наибольшее распространение получили следующие алгоритмы динамические изменения приоритетов:

простая циклическая смена приоритетов; циклическая смена приоритетов с учетом последнего запроса; смена приоритетов по случайному закону; схема равных приоритетов; алгоритм наиболее давнего использования.

В алгоритме **простой циклической смены приоритетов** после каждого цикла арбитража все приоритеты понижаются на один уровень, при этом устройство, имевшее ранее низший уровень приоритета, получает наивысший приоритет.

В схеме **циклической смены, приоритетов с учетом последнего запроса** все возможные запросы упорядочиваются в виде циклического списка. После обработки очередного запроса обслуженному ведущему назначается низший уровень приоритета. Следующее в списке устройство получает наивысший приоритет, а остальным устройствам приоритеты назначаются в убывающем порядке, согласно их следованию в циклическом списке.

В обеих схемах циклической смены приоритетов каждому ведущему обеспечивается шанс получить шину в свое распоряжение, однако большее распространение получил второй алгоритм.

При смене **приоритетов по случайному закону** после очередного цикла арбитража с помощью генератора псевдослучайных чисел каждому ведущему присваивается случайное значение уровня приоритета.

В схеме **равных приоритетов** при поступлении к арбитру нескольких запросов каждый из них имеет равные шансы на обслуживание. Возможный конфликт разрешается арбитром. Такая схема принята в асинхронных системах.

В **алгоритме наиболее давнего использования** (LRU, Least Recently Used) после каждого цикла арбитража наивысший приоритет присваивается ведущему, который дольше чем другие не использовал шину.

### 40.Синхронный и асинхронный протоколы шин данных.

**Синхронный** **протокол**

В синхронных шинах имеется центральный генератор тактовых импульсов (ГТИ), к импульсам которого «привязаны» все события на шине. Тактовые импульсы (ТИ) распространяются по специальной сигнальной линии и представляют собой регулярную последовательность чередующихся единиц и нулей. Один период такой последовательности называется *тактовым периодом шины.* Именно он определяет минимальный квант времени на шине (временной слот). Все подключенные к шине устройства могут считывать состояние тактовой линии, и все события на шине отсчитываются от начала тактового периода. Изменение управляющих сигналов на шине обычно совпадает с передним или задним фронтом тактового импульса, иными словами, момент смены состояния на синхронной шине известен заранее и определяется тактовыми импульсами.

**Асинхронный** **протокол**

Синхронная передача быстра, но в ряде ситуаций не подходит для использования. В частности, в синхронном протоколе ведущий не знает, корректно ли ответил ведомый, — возможно, он был не в состоянии удовлетворить запрос на нужные данные. Более того, ведущий должен работать со скоростью самого медленного из участвующих в пересылке данных ведомых. Обе проблемы успешно решаются в асинхронном протоколе шины.

В асинхронном протоколе начало очередного события на шине определяется не тактовым импульсом, а предшествующим событием и следует непосредственно за этим событием. Каждая совокупность сигналов, помещаемых на шину, сопровождается соответствующим синхронизирующим сигналом, называемым *стробом.* Синхросигналы, формируемые ведомым, часто называют *квитирующими синалами* (handshakes) или *подтверждениями сообщения* (acknowledges).

### 41.Системы ввода/вывода.

Система ввода/вывода призвана обеспечить обмен информацией между ядром ВМ и разнообразными внешними устройствами (ВУ). Технические и программные средства СВВ несут ответственность за физическое и логическое сопряжение ядра вычислительной машины и ВУ

Технически система ввода/вывода в рамках ВМ реализуется комплексом модулей ввода/вывода (МВВ). Модуль ввода/вывода выполняет сопряжение ВУ с ядром ВМ и различные коммуникационные операции между ними. Две основные функции МВВ: - обеспечение интерфейса с ЦП и памятью («большой» интерфейс); - обеспечение интерфейса с одним или несколькими периферийными устройствами («малый» интерфейс). Анализируя архитектуру известных ВМ, можно выделить три основных способа подключения СВВ к ядру процессора:

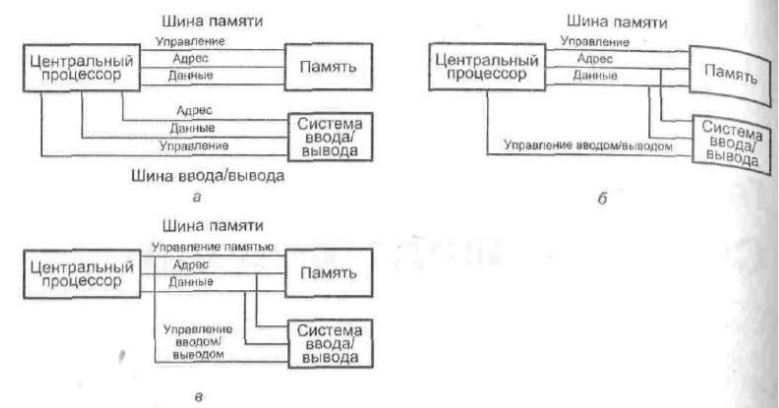


Рис. 13.1.1. Место системы ввода/вывода в архитектуре вычислительной машины: (а) - раздельными шинами памяти и ввода/вывода; (б) - с совместно используемыми линиями данных и адреса; (в) - подключение на общих правах с процессором и памятью

В варианте с раздельными шинами памяти и ввода/вывода (см. рис. 13.1.1, а) обмен информацией между ЦП и памятью физически отделен от ввода/вывода, поскольку обеспечивается полностью независимыми шинами. Это дает возможность осуществлять обращение к памяти одновременно с выполнением ввода/вывода.

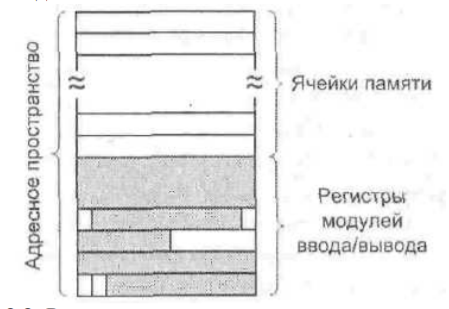
Второй вариант — с совместно используемыми линиями данных и адреса (см рис. 13.1.1, 6). Память и СВВ имеют общие для них линии адреса и линии данных, разделяя их во времени. В то же время управление памятью и СВВ, а также синхронизация их взаимодействия с процессором осуществляются независимо по раздельным линиям управления. Это позволяет учесть особенности процедур обращения к памяти и к модулям ввода/вывода и добиться наибольшей эффективности доступа к ячейкам памяти и внешним устройствам.

Последний тип архитектуры ВМ предполагает подключение СВВ к системном шине на общих правах с процессором и памятью (см. рис. 13.1.1, в).

### 42.Адресное пространство системы ввода/вывода.

Адресное пространство ввода/вывода может быть совмещено с адресным пространством памяти или быть выделенным.

При **совмещении адресного пространства** для адресации модулей ввода/вывода отводится определенная область адресов (рис. 13.2.2). Обычно - 3 - все операции с модулем ввода/вывода осуществляются с использованием входящих в него внутренних регистров: управления, состояния, данных. Фактически процедура ввода/вывода сводится к записи информации в одни регистры МВВ и считыванию ее из других регистров. Это позволяет рассматривать регистры МВВ как ячейки основной памяти и работать с ними с помощью обычных команд обращения к памяти, при этом в системе команд ВМ вообще могут отсутствовать специальные команды ввода и вывода.

****

**Достоинства совмещенного адресного пространства:**

- расширение набора команд для обращения к внешним устройствам, что позволяет сократить длину программы и повысить быстродействие;

- значительное увеличение количества подключаемых внешних устройств;

- возможность внепроцессорного обмена данными между внешними устройствами, если в системе команд есть команды пересылки между ячейками памяти;

- возможность обмена информацией не только с аккумулятором, но и с любым регистром центрального процессора.

**Недостатки совмещенного адресного пространства:**

- сокращение области адресного пространства памяти;

- усложнение декодирующих схем адресов в СВВ;

- трудности распознавания операций передачи информации при вводе/выводе среди других операций. Сложности в чтении и отладке программы, в которой простые команды вызывают выполнение сложных операций ввода/вывода;

- трудности при построении СВВ на простых модулях ввода/вывода: сигнал управления не смогут координировать сложную процедуру ввода/вывода. Поэтому МВВ часто должны генерировать дополнительные сигналы под управлением программы.

В случае **выделенного адресного пространства** для обращения к модулям ввода/вывода применяются специальные команды и отдельная система адресов. Эти позволяет разделить шины для работы с памятью и шины ввода/вывода, что дает возможность совмещать во времени обмен с памятью и ввод/вывод. В вычислительных машинах фирмы IBM и микроЭВМ на базе процессоров фирмы Intel система ввода/вывода, как правило, организуется в соответствии с концепцией выделенного адресного пространства.

**Достоинства выделенного адресного пространства:**

- адрес внешнего устройства в команде ввода/вывода может быть коротким. В большинстве СВВ количество внешних устройств намного меньше количества ячеек памяти. Короткий адрес ВУ подразумевает такие же короткие команды ввода/вывода и простые дешифраторы;

- программы становятся более наглядными, так как операции ввода/вывода выполняются с помощью специальных команд;

- разработка СВВ может проводиться отдельно от разработки памяти.

**Недостатки выделенного адресного пространства:**

- ввод/вывод производится только через аккумулятор центрального процессора; - перед обработкой содержимого ВУ это содержимое нужно переслать в ЦП.

### 43.Обобщенная структура внешних устройств.

****

Рис. 13.3.1 Структура внешнего устройства

Обобщенная структура ВУ показана на рис. 13.3.1. Интерфейс с МВВ реализуется в виде сигналов управления, состояния и данных

**Данные** представлены совокупностью битов, которые должны быть переданы в модуль ввода/вывода или получены из него.

**Сигналы управления** определяют функцию, которая должна быть выполнена внешним устройством. Это может быть стандартная для всех устройств функция — посылка данных в МВВ или получение данных из него, либо специфичная для данного типа ВУ функция, такая, например, как позиционирование головки магнитного диска или перемотка магнитной ленты. **Сигналы состояния** характеризуют текущее состояние устройства, в частности включено ли ВУ и готово ли оно к передаче данных. Логика управления - это схемы, координирующие работу ВУ в соответствии с направлением передачи данных.

Задачей **преобразователя** является трансформация информационных сигналов, имеющих самую различную физическую природу, в электрические сигналы, а также обратное преобразование.

Обычно совместно с преобразователем используется **буферная память**, обеспечивающая временное хранение данных, пересылаемых между МВВ и ВУ.

### 44.Методы управления вводом/выводом.

В ВМ находят применение три способа организации ввода/вывода (В/ВЫВ):

1. программно управляемый ввод/вывод;
2. ввод/вывод по прерываниям;
3. прямой доступ к памяти.

При программно управляемом вводе/выводе все связанные с этим действия происходят по инициативе центрального процессора и под его полным контролем. ЦП выполняет программу, которая обеспечивает прямое управление процессом ввода/вывода, включая проверку состояния устройства, выдачу команд ввода или вывода. Выдав в МВВ команду, центральный процессор должен ожидать завершения ее выполнения, и, поскольку ЦП работает быстрее, чем МВВ, это приводит к потере времени.

Ввод/вывод по прерываниям во многом совпадает с программно управляемым методом. Отличие состоит в том, что после выдачи команды ввода/вывода ЦП должен циклически опрашивать МВВ для выяснения состояния устройства. Вместо этого процессор может продолжать выполнение других команд до тех пор, пока не получит запрос прерывания от МВВ, извещающим о завершении выполнения ранее выданной команды В/ВЫВ. Как и при программно управляемом В/ВЫВ, ЦП отвечает за извлечение данных из памяти (при выводе) и запись данных в память (при вводе).

Повышение как скорости В/ВЫВ, так и эффективности использования ЦП обеспечивает третий способ В/ВЫВ — прямой доступ к памяти (ПДП). В этом режиме основная память и модуль ввода/вывода обмениваются информацией напрямую, минуя процессор.

### 45. Уровни параллелизма.

Методы и средства реализации параллелизма зависят от того, на каком уровне он Должен обеспечиваться. Обычно различают следующие уровни параллелизма:

* **Уровень заданий**. Несколько независимых заданий одновременно выполняются на разных процессорах, практически не взаимодействуя друг с другом. Этот уровень реализуется на ВС с множеством процессоров в многозадачном режиме.
* **Уровень программ**. Части одной задачи выполняются на множестве процессоров. Данный уровень достигается на параллельных ВС.
* **Уровень команд**. Выполнение команды разделяется на фазы, а фазы нескольких последовательных команд могут быть перекрыты за счет конвейеризации. Уровень достижим на ВС с одним процессором.
* **Уровень битов** (арифметический уровень). Биты слова обрабатываются один за другим, это называется бит-последовательной операцией. Если биты слова обрабатываются одновременно, говорят о битпараллельной операции.

### 46.Метрики параллельных вычислений.

1. **Метрики** **параллельных** **вычислений -** это система показателей, позволяющая оценивать преимущества, получаемые при параллельном решении задачи на n процессорах, по сравнению с последовательным решением той же задачи на единственном процессоре. С другой стороны, они позволяют судить об обоснованности применения данного числа процессоров для решения конкретной задачи.

Базисом для определения метрик являются следующие характеристики вычислений:

n - количество процессоров, используемых для организации параллельных вычислений;

O(n) - объем вычислений, выраженный через количество операций, выполняемых n процессорами в ходе решения задачи;

Т(n) - общее время вычислений (решения задачи) с использованием n процессоров.

В однопроцессорной системе T(1)=O(1). В общем случае Т(n)<О(n), если в единицу времени n процессорами выполняется более чем одна команда, где n>2. Последнее соотношение формулирует утверждение: время вычислений можно сократить за счет распределения объема вычислений по нескольким процессорам.

Можно выделить четыре группы метрик.

Первая характеризует **скорость вычислений**:

Индекс параллелизма: PI(n)=O(n)/T(n).

Ускорение: S(n)=T(1)/T(n).

Вторую группу образуют метрики, дающие возможность судить об **эффективности привлечения к решению задачи дополнительных процессоров**.

Эффективность: E(n)=S(n)/n=T(1)/(nT(n)).

Утилизация: U(n)=R(n)×E(n)=O(n)/(n×T(n)).

Третья группа метрик характеризует **эффективность параллельных вычислений путем сравнения объема вычислений**, выполненного при параллельном и последовательном решении задачи.

Избыточность: R(n)=O(n)/O(1).

Сжатие: C(n)=O(1)/O(n).

Четвертую группу образует метрика **качество**:

Q(n)=S(n)E(n)C(n).

Поскольку эта метрика увязывает метрики ускорение, эффективность и сжатие, она является более объективным показателем улучшения производительности за счет параллельных вычислений.

# Вроде все Всем спокойной ночи и удачи!