### Вопросы к зачету по дисциплине ОПД (2023г.)

Оглавление

[1. Операционные системы 2](#__RefHeading___Toc14499_1991852699)

[2. История UNIX/Linux 2](#__RefHeading___Toc14497_1991852699)

[3. Современность 2](#__RefHeading___Toc14495_1991852699)

[4. Ядро \*NIX 2](#__RefHeading___Toc14493_1991852699)

[5. Файловая система 3](#__RefHeading___Toc14491_1991852699)

[6. Права доступа к файлам 3](#__RefHeading___Toc14489_1991852699)

[7. Способы задания прав 3](#__RefHeading___Toc14487_1991852699)

[8. Потоки stdin(0), stdout(1), stderr(2) 4](#__RefHeading___Toc14485_1991852699)

[9. Интерпретатор команд 4](#__RefHeading___Toc14667_1991852699)

[10. Перенаправление потоков stdin(0), stdout(1), stderr(2) 4](#__RefHeading___Toc14665_1991852699)

[11. Фильтры 4](#__RefHeading___Toc14663_1991852699)

[12. Регулярные выражения 4](#__RefHeading___Toc14641_1991852699)

[13. Команды 4](#__RefHeading___Toc14661_1991852699)

[14. Аналоговые ЭВМ 4](#__RefHeading___Toc14659_1991852699)

[15. Цифровые ЭВМ 4](#__RefHeading___Toc27312_1991852699)

[16. Функциональные элементы ЭВМ 4](#__RefHeading___Toc14655_1991852699)

[17. Первая ЭВМ: Калькулятор (1) 4](#__RefHeading___Toc14653_1991852699)

[18. Архитектура ЭВМ, Гарвардская, Фон Неймана. 4](#__RefHeading___Toc14651_1991852699)

[19. Структура БЭВМ-NG 5](#__RefHeading___Toc14649_1991852699)

[20. Устройство Управления 5](#__RefHeading___Toc14647_1991852699)

[21. Адресуемая память БЭВМ 5](#__RefHeading___Toc14645_1991852699)

[22. АЛУ, коммутатор, блок признаков результата 5](#__RefHeading___Toc14643_1991852699)

[23. Форматы команд 5](#__RefHeading___Toc14639_1991852699)

[24. Адресные команды 5](#__RefHeading___Toc14637_1991852699)

[25. Безадресные команды 5](#__RefHeading___Toc14635_1991852699)

[26. Представление чисел: фиксированная точка 5](#__RefHeading___Toc14633_1991852699)

[27. Представление беззнаковых целых чисел 5](#__RefHeading___Toc14631_1991852699)

[28. Представление знаковых целых чисел 5](#__RefHeading___Toc14629_1991852699)

[29. Представление знаковых чисел: дополнительный код 5](#__RefHeading___Toc14627_1991852699)

[30. Перенос, Переполнение 5](#__RefHeading___Toc14625_1991852699)

[31. БЭВМ: представление чисел 5](#__RefHeading___Toc14623_1991852699)

[32. Представление чисел с плавающей точкой 5](#__RefHeading___Toc14621_1991852699)

[33. Представление логической информации 5](#__RefHeading___Toc14619_1991852699)

[34. Представление символьной и текстовой информации 6](#__RefHeading___Toc14617_1991852699)

[35. Символы: ASCII 6](#__RefHeading___Toc14615_1991852699)

[36. Символы: ASCII (КОИ-7H0)КОИ-7H1 (РУС),КОИ-7H2 (Mix) 6](#__RefHeading___Toc14613_1991852699)

[37. Символы: КОИ-8 6](#__RefHeading___Toc14611_1991852699)

[38. Символы: ISO8859-5 (ГОСТ-основная) 6](#__RefHeading___Toc14609_1991852699)

[39. Символы: WIN1251 6](#__RefHeading___Toc14607_1991852699)

[40. Символы: UNICODE, UTF-8 7](#__RefHeading___Toc14605_1991852699)

[41. Big-endian и Little-endian 7](#__RefHeading___Toc14603_1991852699)

[42. Представление строк 7](#__RefHeading___Toc14601_1991852699)

[43. История развития ЭВМ 8](#__RefHeading___Toc14599_1991852699)

[44. История развития ЭВМ в СССР/России 8](#__RefHeading___Toc14597_1991852699)

[45. Канальная организация 9](#__RefHeading___Toc14595_1991852699)

[46. Раздельные шины 9](#__RefHeading___Toc14593_1991852699)

[47. Общие шины 9](#__RefHeading___Toc14591_1991852699)

[48. Мультиплексирование шин 9](#__RefHeading___Toc14589_1991852699)

[49. Мультипроцессорность:UMA - Uniform Memory Access 9](#__RefHeading___Toc14587_1991852699)

[50. Мультипроцессорность: Коммутатор 9](#__RefHeading___Toc14585_1991852699)

[51. Мультипроцессорность: NUMA – Non Uniform Memory Access 9](#__RefHeading___Toc14583_1991852699)

[52. Современные коммерческие процессоры 10](#__RefHeading___Toc14581_1991852699)

[53. CISC, RISC, VLIW 10](#__RefHeading___Toc14579_1991852699)

[54. Характеристики памяти 11](#__RefHeading___Toc14577_1991852699)

[55. Статическая vs Динамическая память 11](#__RefHeading___Toc14575_1991852699)

[56. Адресуемая память 11](#__RefHeading___Toc14573_1991852699)

[57. Адресуемая память с фиксацией строк и столбцов 11](#__RefHeading___Toc14571_1991852699)

[58. Синхронная память SDRAM 11](#__RefHeading___Toc14569_1991852699)

[59. Конструктивные особенности современной памяти 11](#__RefHeading___Toc14567_1991852699)

[60. Память, ориентированная на записи 11](#__RefHeading___Toc14565_1991852699)

[61. Память, с последовательным доступом\* 11](#__RefHeading___Toc14563_1991852699)

[62. Структура ассоциативного запоминающего устройства 11](#__RefHeading___Toc14561_1991852699)

[63. Кэш память 11](#__RefHeading___Toc14559_1991852699)

[64. Пирамида памяти 11](#__RefHeading___Toc14557_1991852699)

[65. Влияние промахов кэш-памяти 11](#__RefHeading___Toc14555_1991852699)

[66. Сегментно-страничная виртуальная память 11](#__RefHeading___Toc14553_1991852699)

[67. MMU и TLB 11](#__RefHeading___Toc14551_1991852699)

[68. История сети Internet 12](#__RefHeading___Toc14549_1991852699)

[69. Понятие сети ЭВМ 12](#__RefHeading___Toc14547_1991852699)

[70. Сообщение, пакет 12](#__RefHeading___Toc14545_1991852699)

[71. Модель взаимодействия открытых систем (OSI) 12](#__RefHeading___Toc14543_1991852699)

[72. Модель TCP/IP 12](#__RefHeading___Toc14525_1991852699)

[73. Уровень передающей среды 12](#__RefHeading___Toc14541_1991852699)

[74. Канальный уровень Ethernet 12](#__RefHeading___Toc14539_1991852699)

[75. Сетевой уровень IP 12](#__RefHeading___Toc14537_1991852699)

[76. Сетевой уровень IP: маршрутизация 12](#__RefHeading___Toc14535_1991852699)

[77. DHCP 12](#__RefHeading___Toc14533_1991852699)

[78. Cервис имен DNS и другие 13](#__RefHeading___Toc14531_1991852699)

[79. Транспортный уровень 13](#__RefHeading___Toc14529_1991852699)

[80. Прикладной уровень 13](#__RefHeading___Toc14527_1991852699)

1. Операционные системы

Операционная система — это комплекс программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

* Исторически предназначена для замены работы оператора компьютерной системы – это люди которые брали данные с пользователей, загружали их в ЭВМ, а затем печатали им результаты.
* Условно делятся на пользовательские (Linux, Windows и т.д.) – должны быть информативные, серверные (обработка большого количества задач) и встроенные ОС (там где её не ждут, к примеру на холодильнике)
* Windows, Linux/Unix, Android/IOS, VXWoks/EmbededLinux/DOS, Гипервизоры -  программное решение, которое позволяет запускать несколько виртуальных машин одновременно. Это ПО координирует и разделяет вычислительную мощность, память и другие ресурсы, выделяя нужные мощности каждой виртуальной машиной по мере необходимости.
* Обычно включают ядро с подсистемами управления памятью и процессами и драйверы устройств. Драйвер — это программа, которая помогает компьютеру распознать какое-нибудь подключенное устройство. Благодаря драйверам [операционная система](https://blog.skillfactory.ru/glossary/operaczionnaya-sistema/) компьютера видит мышь, клавиатуру, монитор и другие девайсы.

Человек -> оболочка -> ядро -> далее оно выдает ресурсы

Драйвер — это программное обеспечение, которое позволяет операционной системе взаимодействовать с аппаратным обеспечением компьютера. Драйверы необходимы для правильной работы различных устройств, таких как видеокарты, принтеры, сканеры, звуковые карты и другие периферийные устройства.

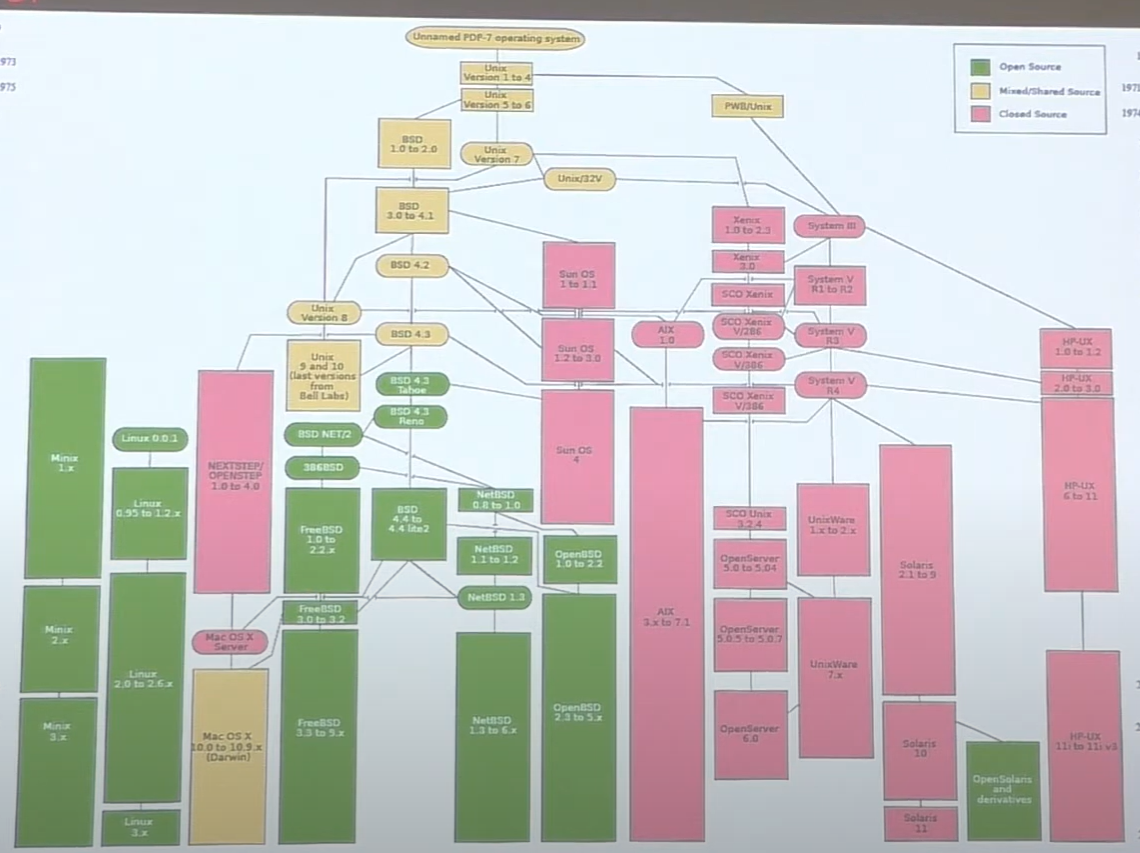
Основные функции драйвера включают:

1. Инициализация устройства: Драйвер запускает устройство и готовит его к работе.
2. Обработка запросов: Драйвер переводит команды операционной системы в команды, понятные устройству, и наоборот.
3. Обработка прерываний: Драйвер управляет прерываниями, которые устройство может посылать операционной системе.
4. Оптимизация работы: Драйвер может включать функции для оптимизации производительности устройства.

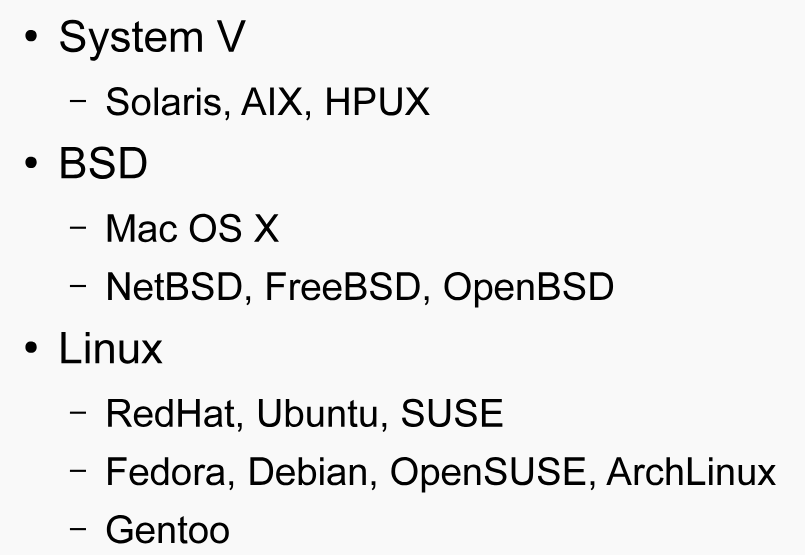
Драйверы могут быть встроены в операционную систему или устанавливаться отдельно. Важно использовать актуальные и совместимые драйверы для обеспечения стабильной и эффективной работы устройств.

2. История UNIX/Linux

UNIX — семейство переносимых, многозадачных и многопользовательских операционных систем. Идеи, заложенные в основу UNIX, оказали огромное влияние на развитие компьютерных операционных систем. В настоящее время UNIX-системы признаны одними из самых исторически важных ОС. Основное отличие UNIX-подобных систем от других операционных систем заключается в том, что это изначально многопользовательские многозадачные системы. То есть в один и тот же момент времени сразу множество людей может выполнять множество вычислительных задач (процессов). Даже популярную во всём мире систему Microsoft Windows нельзя назвать полноценной многопользовательской системой, так как кроме как на некоторых серверных версиях, в один и тот же момент за одним компьютером с Windows может работать только один человек. В Unix может работать сразу много людей, при этом каждый из них может выполнять множество различных вычислительных процессов, которые будут использовать ресурсы именно этого компьютера. Вторая колоссальная заслуга Unix в её мультиплатформенности.



3. Современность



System V — это одна из основных ветвей Unix, разработанная AT&T. Она была выпущена в 1983 году и стала основой для многих коммерческих версий Unix, таких как HP-UX, AIX и Solaris.

Основные характеристики System V:

* Коммерческая ориентация: System V была разработана для коммерческого использования и поддерживала множество функций, необходимых для корпоративных систем.
* Системные вызовы и библиотеки: System V имеет свои уникальные системные вызовы и библиотеки, которые отличаются от других версий Unix.
* Инициализация системы: System V использует систему инициализации (init) с уровнями запуска (runlevels), что позволяет гибко управлять запуском и остановкой системы.
* Файловая система: System V поддерживает различные файловые системы, включая UFS (Unix File System).

BSD — это другая ветвь Unix, разработанная в Калифорнийском университете в Беркли. Она была выпущена в 1977 году и стала основой для многих открытых и свободных операционных систем, таких как FreeBSD, OpenBSD и NetBSD.

Основные характеристики BSD:

* Академическая ориентация: BSD была разработана в академической среде и включала множество инноваций, таких как TCP/IP стек, который стал основой для интернета.
* Открытый исходный код: Многие версии BSD являются открытыми и свободными, что позволяет пользователям и разработчикам свободно изменять и распространять код.
* Сетевые возможности: BSD известна своими мощными сетевыми возможностями и безопасностью.
* Файловая система: BSD использует файловую систему UFS (Unix File System), но также поддерживает другие файловые системы, такие как ZFS.

4. Ядро \*NIX

Ядро системы написано таким образом, что его легко можно приспособить практически под любой микропроцессор. UNIX имеет и другие характерные особенности:

• использование простых текстовых файлов для настройки и управления системой;

• широкое применение утилит, запускаемых из командной строки;

• взаимодействие с пользователем посредством виртуального устройства — терминала;

• представление физических и виртуальных устройств и некоторых средств межпроцессового взаимодействия в виде файлов;

• использование конвейеров из нескольких программ, каждая из которых выполняет одну задачу.

Основные компоненты ядра

1. Менеджер процессов

Управляет созданием, выполнением и завершением процессов.

Планировщик процессов распределяет процессорное время между активными процессами.

2. Менеджер памяти

Управляет распределением и использованием памяти, обеспечивая виртуальную память и защиту памяти.

3. Менеджер файлов

Отвечает за доступ к файловой системе, организует чтение и запись данных на дисковые накопители.

4. Менеджер устройств

Обеспечивает взаимодействие с аппаратными устройствами через драйверы.

5. Менеджер сетей

Обрабатывает сетевые запросы и обеспечивает сетевую коммуникацию между устройствами.

Пример работы ядра

Когда пользователь запускает программу, например текстовый редактор, ядро выполняет следующие действия:

1. Создание процесса: Ядро создает новый процесс для текстового редактора.

2. Выделение памяти: Ядро выделяет необходимую память для работы программы.

3. Открытие файлов: Если текстовый редактор открывает файл, ядро предоставляет доступ к файлу, контролируя права доступа.

4. Обработка ввода-вывода: Когда пользователь вводит текст или сохраняет файл, ядро управляет взаимодействием с клавиатурой и диском.

5. Файловая система

Понятие файла является одним из наиболее важных для ОС UNIX. Все файлы, с которыми могут манипулировать пользователи, располагаются в файловой системе, представляющей собой дерево, промежуточные вершины которого соответствуют каталогам, и листья - файлам и пустым каталогам. Каждый каталог и файл файловой системы имеет уникальный полный путь. Коротким или относительным путем называется путь к файлу от текущего рабочего каталога. В каждом каталоге содержатся два специальных файла-ссылки, файл "." - ссылка на текущий каталог, и ссылка ".." на родительский каталог. inode - Index-node - описатель файла, его уникальный номер. Он содержит всю информацию о файле, за исключением имени файла, и собственно данных файла. В inod'е хранится: тип файла, права, время модификации/создания файла и другая служебная информация под общим названием «метаданные».

Файловая система Unix: В Unix файл — это хранилище двоичных и символьных данных, организованных как поток байтов. Файлы в Unix играют ключевую роль и расположены в файловой системе, которая представляет собой дерево. Вершины дерева — это каталоги, а листья — файлы и пустые каталоги.

Структура файловой системы:

**Корневой каталог** (/) — корень всей файловой системы. Каталог, являющийся корнем файловой системы (корневой каталог) имеет путь /.

**Уникальные пути**: Каждый файл и каталог имеют уникальный полный путь.

**Относительные пути**: Пути к файлам могут быть указаны относительно текущего рабочего каталога.

**Специальные ссылки**: В каждом каталоге есть два специальных файла-ссылки — «.» (текущий каталог) и «..» (родительский каталог).

**Иноды** (inode): Индексовый узел (inode) — это уникальный идентификатор файла в Unix. Он содержит всю информацию о файле, кроме имени и данных самого файла. В inode хранится:

* Тип файла (обычный файл, каталог). Права доступа к файлу.
* Время создания и модификации файла.
* Другие метаданные, такие как количество ссылок на файл, размер файла и указатели на блоки данных.

Дополнительные сведения об inode: Индексовые узлы являются ключевыми элементами в файловой системе Unix, так как они позволяют системе быстро находить и управлять файлами. Каждый файл имеет свой уникальный inode, который можно рассматривать как паспорт файла, содержащий всю необходимую информацию для его идентификации и управления.

Unix является мощной и гибкой операционной системой, которая благодаря своим уникальным характеристикам и структуре файловой системы остаётся востребованной и популярной во многих областях.

Директория – файл с названиями файлов и i-node ов к ним. Системная утилита не дает его прочитать. Не более того

6. Права доступа к файлам

Все категории. Не рассматривается как отдельная катеория.

Каждая из этих категорий может иметь любую комбинацию из прав:

Право на чтение файла/просмотр каталога: r (read)

Право на запись в файл/добавление или удаление каталога: w (write)

Право на исполнение файла/поиск и переход в каталог: x (execute)

Права представляют собой последовательность из 9 бит – по 3 бита на категорию: владелец,группа, прочие; в следующем порядке – чтение, запись, исполнение. В случае отсутствия какого-либо из прав у категории, ставится символ «-».

Вторым способом записи прав является запись этой последовательности в 8-ричной системе счисления, где праву на чтение (r) соответствует цифра 4, праву на запись (w) – цифра 2, а праву на исполнение (x) – цифра 1. Цифра 0 означает отсутствие прав. Для получения конечной цифры, нужные права суммируются. Таким образом, запись занимает всего 3 бита: по 1 биту на категорию.

**umask** (user file-creation mode mask) — это команда в Unix-подобных операционных системах, таких как Linux, которая используется для установки значений по умолчанию для прав доступа к файлам и каталогам, создаваемым пользователем.

7. Способы задания прав

Для выставления прав файлу (каталогу) используется команда chmod. Существует 3 способа задания прав доступа:

* **chmod [ugoa]{+-=}[rwx] файл** - добавляет, удаляет или устанавливает выбранную комбинацию прав для выбранной комбинации категорий.
* **chmod число файл** - устанавливает права на основе восьмеричной записи.
* **chmod категория1=категория2 файл** - копирует права одной категории и присваивает их другой.

8. Потоки stdin(0), stdout(1), stderr(2)

Для взаимодействия и обмена информацией с пользователем используются файлы, именуемые стандартными потоками ввода (для чтения из него) и вывода (для записи в него). Вывод на экран представляется тоже как запись в файл, а ввод – как чтение из файла. Кроме потоков ввода и вывода существует также стандартный поток ошибок, на который выводится вся служебная информация, которая не должна попадать в поток вывода (сообщения об ошибке или ходе работы программы). Стандартные потоки привязаны к файловым дескрипторам с номерами: 0 для ввода (stdin), 1 для вывода (stdout), 2 для ошибок (stderr).

Потоки по умолчанию связаны с терминалом (командной строкой\*), но их можно подключить к чему угодно – к файлам, программам или устройствам. В интерпретаторе такая операция называется перенаправлением. Таким образом, стандартные потоки можно перенаправлять не только в файлы, но и на вход других программ.

\*командная строка — текстовый интерфейс пользователя для взаимодействия с операционной системой компьютера и/или другим программным обеспечением с помощью команд, вводимых с клавиатуры.

Файловый дескриптор (File Descriptor, FD) в Unix и Unix-подобных операционных системах — это целочисленный идентификатор, который используется для доступа к файлу или другому ресурсу ввода-вывода (I/O). Файловые дескрипторы предоставляют абстракцию для работы с различными типами ресурсов, такими как файлы, сокеты, каналы (pipes) и устройства.

9. Интерпретатор команд

Командный интерпретатор – программа, предоставляющая пользователю интерфейс для общения с командной строкой; эта программа «переводит» введенные пользователем команды на понятный операционной системе язык. Интерпретатор более известен как оболочка (англ. shell). Наиболее распространенными оболочками являются sh, ksh, bash (стандарт в Unix), c shell. Пользователь может вводить команды как по отдельности, так и с помощью набора команд (скриптов).

Команды могут задаваться как напрямую в командной строке, так и поступать из стандартного ввода или указанного файла.

Пользователи могут вводить команды по одной или использовать скрипты — файлы, содержащие последовательности команд. Эти команды могут быть:

* Системные утилиты (например, команды для управления файлами).
* Прикладные утилиты (программы, написанные пользователем).
* Управляющие конструкции (условия, циклы и т.д.).

Кроме того, оболочка отвечает за перенаправление потоков ввода-вывода. В совокупности с набором утилит, она представляет собой операционную среду, язык программирования и средства решения как системных, так и прикладных задач, особенно по части автоматизации выполняемых последовательностей команд.

10. Перенаправление потоков stdin(0), stdout(1), stderr(2)

U nix использует три основных потока для взаимодействия с данными:

* Стандартный ввод (stdin) — поток, из которого программы читают данные. Обычно это клавиатура.
* Стандартный вывод (stdout) — поток, в который программы записывают результаты своей работы. Обычно это экран.
* Стандартный поток ошибок (stderr) — поток, в который записываются сообщения об ошибках и другая служебная информация.

Потоки по умолчанию связаны с терминалом (командной строкой), но их можно подключить к чему угодно – к файлам, программам или устройствам. В интерпретаторе такая операция называется перенаправлением. Таким образом, стандартные потоки можно перенаправлять не только в файлы, но и на вход других программ.

Для осуществления перенаправления используются следующие операции:

**Команда > файл (или >>)**

Выполняется команда, а вывод помещается в файл (или добавляется в конец).

**Команда < файл**

Файл используется в качестве источника ввода. При этом на каждый запрос ввода программы считывается 1 строка текста из файла:

**Команда1 | Команда2**

Вывод команды1 пойдет в качестве ввода на команду2 без использования промежуточных файлов. Такая возможность называется конвейером:

**Команда 2> файл**

Поток ошибок направляется в файл. По умолчанию этот поток выводится на стандартный вывод:

**Команда 2>&1 файл (или &> или >&)**

Такой синтаксис используется для объединения потоков вывода и потока ошибок для обработки их вместе.

Файл т.н. «пустое устройство» - /dev/null – перенаправление в него позволяет избавиться от ненужных сообщений об ошибке или игнорирования вывода. С помощью него также можно создавать пустые файлы, используя в качестве источника ввода. При записи в него может вместить любое количество информации, он работает в качестве «черной дыры».

11. Фильтры

Фильтры — это утилиты, которые обрабатывают данные на входе и выводят результат. Примеры фильтров:

wc — подсчет слов, строк и символов в файле.

grep — поиск строк, соответствующих шаблону.

sort — сортировка строк.

1. Подсчет строк в файле: wc -l file.txt

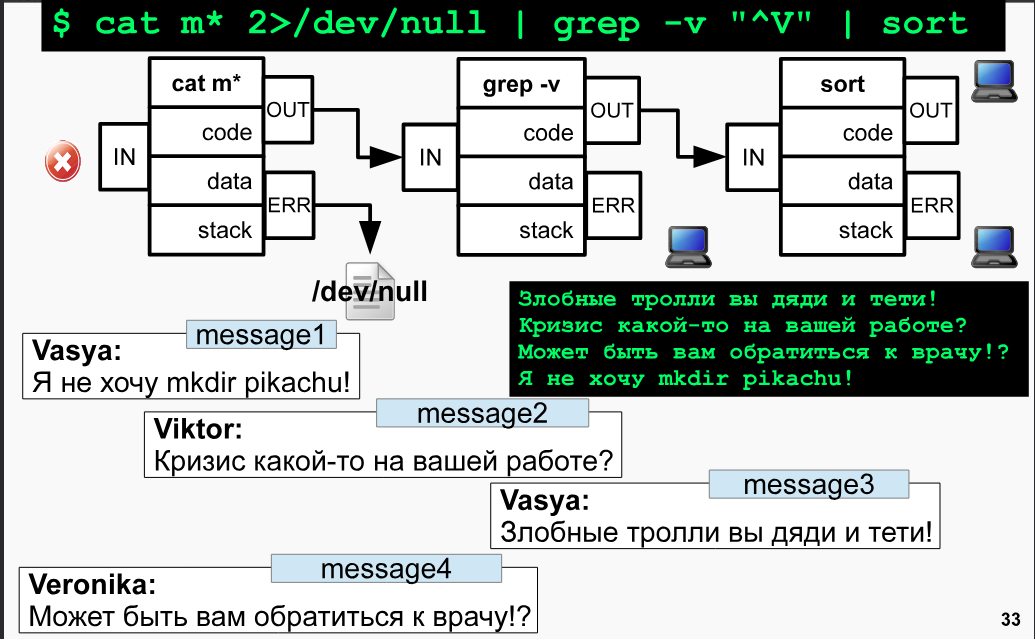
Эта команда выведет количество строк в файле file.txt.

2. Поиск строки, содержащей слово "error": grep "error" file.txt

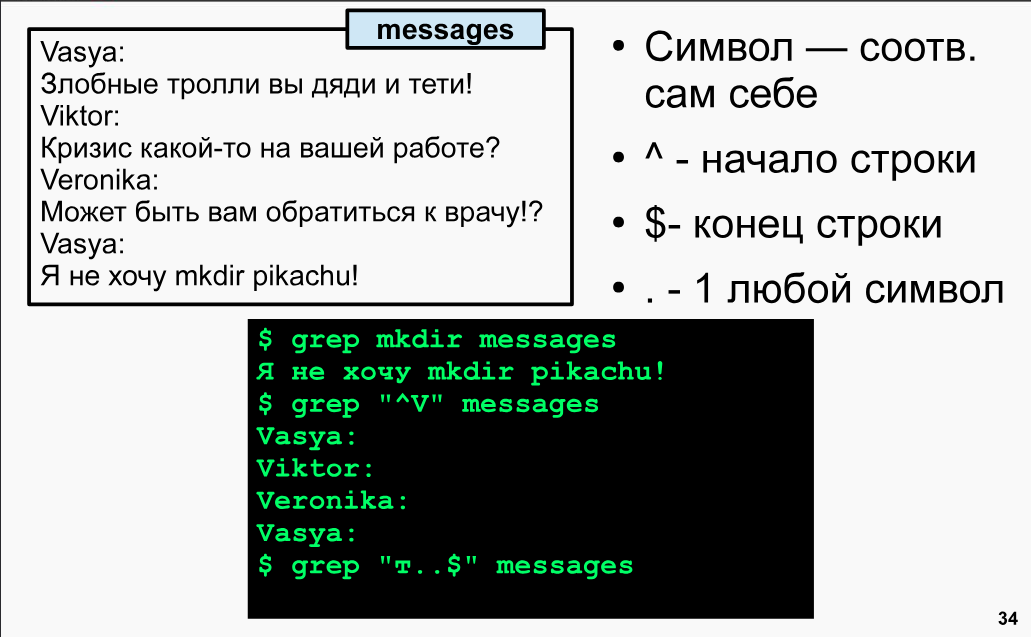
Эта команда выведет все строки из файла слово "error".

3. Сортировка строк в файле: sort file.txt, содержащие file.txt

Эта команда отсортирует строки в файле и выведет результат.



12. Регулярные выражения



13. Команды

Создает пустой файл, а если он уже есть – обновляет время последней модификации: **touch** файл

Создает пустой каталог: **mkdir** каталог

Удаляет файл: **rm** файл

**–r** - рекурсивно стирает каталоги. Если этого флага нет, файл не может быть каталогом.

Стирает только пустые каталоги: **rmdir** каталог

Выводит строку текста: **echo**

Выводит содержимое файла: **cat** файл

Выводит имя текущего каталога: **pwd**

Выводит список файлов в каталоге или информацию о файле, если это не каталог:

**ls** файл

**–l** - длинный формат. Выводится с подробной информацией о каждом файле.

**–a** - вывод вместе со скрытыми файлами.

**–F** - к имени файла добавляется его тип.

**–R** - рекурсивно выводит подкаталоги.

Переходит в каталог: **cd** каталог

Копирует файл в другой файл: **cp** файл1 файл2

Перемещает файл в каталог: **mv** файл каталог

Создает новую жесткую ссылку на файл. Жесткая ссылка может ссылаться только в пределах одного диска. Файл не будет удален, пока на него есть хоть одна жесткая ссылка: **ln** файл1 файл2

**–s** - создает символическую ссылку. Может ссылаться куда угодно. Если переместить/удалить файл, симв. ссылка будет недействительна.

Выводит первые/последние 4 строки файла: **head/tail** файл

**–n** - первые/последние n строк.

**–c** - первые/последние с байт.

Выводит количество строк, слов и байт в файле: **wc** файл

**–l** - только кол-во строк.

**–w** - только кол-во слов.

**–c** - только кол-во байт.

**–m** - кол-во символов.

Ищет файлы в иерархии каталогов по заданным параметрам: **find** выражение

Выводит справку по команде: **man** команда

14. Аналоговые ЭВМ

**Электронная вычислительная машина (ЭВМ)** — комплекс технических средств, где основные функциональные элементы (логические, запоминающие, индикационные и др.) выполнены на электронных элементах, предназначенных для автоматической обработки информации в процессе решения вычислительных и информационных задач.

**Аналоговый компьютер** или **аналоговая вычислительная машина** (**АВМ**) — вычислительная машина, которая представляет числовые данные при помощи аналоговых физических параметров (скорость, длина, напряжение, сила тока, давление), в чём и состоит его главное отличие от цифровой ЭВМ. Другим принципиальным отличием является отсутствие у АВМ хранимой программы, под управлением которой с помощью одной и той же вычислительной машины можно решать разнообразные задачи. Решаемая задача (класс задач) жёстко определяется внутренним устройством АВМ и выполненными настройками (соединениями, установленными модулями, клапанами и т. п.). Даже для универсальных АВМ для решения новой задачи требовалась перестройка внутренней структуры устройства.

**Аналоговая (непрерывная) форма** представления информации используется для описания величин, которые могут принимать бесконечное количество значений в определённом диапазоне. Например, температура может изменяться плавно от одного значения к другому без скачков и перерывов. Это свойство называется *непрерывностью*. В аналоговых вычислительных машинах информация представляется в виде одного сигнала, который пропорционален величине, которую он представляет.

Количество значений, которые может принимать величина, представленная в такой форме бесконечно велико, даже если величина изменяется в ограниченном диапазоне. Отсюда названия – *непрерывная величина* и *непрерывная информация*. Слово непрерывность выделяет основное свойство таких величин – отсутствие разрывов, промежутков между значениями, которые может принимать аналоговая величина. Величина представляется в виде одного сигнала, пропорционального этой величине. Эта форма представления используется в аналоговых вычислительных машинах.

15. Цифровые ЭВМ

**Цифровая (дискретная) форма** представления информации описывает величины, которые могут принимать только определённые значения, не все возможные. Эти величины называются дискретными. В цифровой форме информация представляется в виде набора сигналов, каждый из которых соответствует одной из цифр. Например, число 123 может быть представлено тремя сигналами, где каждый сигнал соответствует цифре: 1, 2 и 3.

Такие величины, принимающие не все возможные, а лишь вполне определённые значения, называются дискретными (прерывистыми). В отличие от непрерывной величины количество значений дискретной величины всегда будет конечным. Величина представляется в виде нескольких сигналов, каждый из которых соответствует одной из цифр заданной величины. Эта форма представления используется в электронных вычислительных машинах (ЭВМ). Примером аналогового представления графической информации может служить, например, живописное полотно, цвет которого изменяется непрерывно, а дискретного - изображение, напечатанное с помощью струйного принтера и состоящее из отдельных точек разного цвета. Примером аналогового хранения звуковой информации является виниловая пластинка (звуковая дорожка изменяет свою форму непрерывно), а дискретного - аудиокомпакт-диск (звуковая дорожка которого содержит участки с различной отражающей способностью).

16. Функциональные элементы ЭВМ

ЭВМ способна понимать и выполнять точно определенный набор команд. При составлении программы пользователь ограничен этими командами. В зависимости от того, к каким блокам базовой ЭВМ обращается команда или на какие блоки она ссылается, команды можно разделить на четыре группы: адресные команды,

безадресные команды, команды ввода-вывода, команды ветвления.

Логические элементы

Элементы хранения (DRAM/SRAM)

### DRAM (Dynamic Random Access Memory)

**DRAM** — это тип оперативной памяти, который используется для временного хранения данных и инструкций, которые активно используются процессором. DRAM является наиболее распространённым типом памяти в современных компьютерах и используется в основной оперативной памяти (RAM).

#### Основные характеристики DRAM:

1. **Динамическое хранение**:
   * DRAM требует периодического обновления (рефреша) для сохранения данных, так как данные хранятся в виде зарядов на конденсаторах, которые со временем разряжаются.
2. **Высокая плотность**:
   * DRAM имеет высокую плотность хранения данных, что позволяет упаковывать больше данных в меньшем пространстве.
3. **Низкая стоимость**:
   * DRAM дешевле в производстве по сравнению с SRAM, что делает его предпочтительным выбором для основной оперативной памяти.
4. **Высокая скорость доступа**:
   * DRAM обеспечивает высокую скорость доступа к данным, хотя и не такую высокую, как SRAM.

#### Примеры использования DRAM:

* Основная оперативная память (RAM) в компьютерах.
* Видеопамять (VRAM) в графических картах.

### SRAM (Static Random Access Memory)

**SRAM** — это тип оперативной памяти, который используется для временного хранения данных и инструкций, которые требуют очень высокой скорости доступа. SRAM используется в кэш-памяти процессоров и других приложениях, где требуется высокая производительность.

#### Основные характеристики SRAM:

1. **Статическое хранение**:
   * SRAM не требует периодического обновления для сохранения данных, так как данные хранятся в виде состояния транзисторов.
2. **Высокая скорость доступа**:
   * SRAM обеспечивает очень высокую скорость доступа к данным, что делает его идеальным для использования в кэш-памяти.
3. **Низкая плотность**:
   * SRAM имеет низкую плотность хранения данных по сравнению с DRAM, что делает его более дорогим в производстве.
4. **Низкое энергопотребление**:
   * SRAM потребляет меньше энергии по сравнению с DRAM, так как не требует периодического обновления.

#### Примеры использования SRAM:

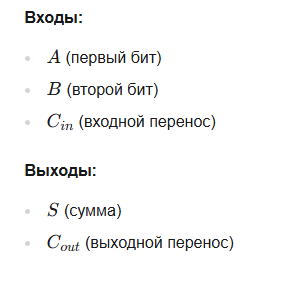
* Кэш-память процессоров (L1, L2, L3 кэш).
* Встроенные системы, где требуется высокая скорость доступа и низкое энергопотребление.

### Сравнение DRAM и SRAM

| **Характеристика** | **DRAM** | **SRAM** |
| --- | --- | --- |
| **Тип хранения** | Динамическое | Статическое |
| **Плотность** | Высокая | Низкая |
| **Стоимость** | Низкая | Высокая |
| **Скорость доступа** | Высокая | Очень высокая |
| **Энергопотребление** | Высокое (требует рефреша) | Низкое (не требует рефреша) |
| **Примеры использования** | Основная оперативная память, видеопамять | Кэш-память процессоров, встроенные системы |

* Логические элементы
* Элементы хранения (DRAM/SRAM)
* Триггеры (хранит 1 бит информации 1 или 0)
* Регистры
* Провода, шины (совокупность проводов)
* Вентили – управление передачи информации (как кран, можно открыть и закрыть)
* Сумматоры (у сумматора 3 входа и 2 выхода)
* Тактовые генераторы

Полный сумматор имеет три входа и два выхода, чтобы учитывать перенос из предыдущего разряда при сложении многоразрядных бинарных чисел. Это позволяет полному сумматору корректно обрабатывать сложение битов, включая перенос, что делает его более универсальным и пригодным для использования в многоразрядных сумматорах.



Тактовые генераторы, также известные как генераторы тактовых сигналов или тактовые схемы, — это электронные устройства, которые генерируют периодические сигналы, используемые для синхронизации работы цифровых схем и процессоров. Эти сигналы, называемые тактовыми сигналами или тактовыми импульсами, обеспечивают временную координацию выполнения операций в цифровых системах.

*Адресные команды* выполняют различные действия, ссылаясь на ячейку памяти в

зависимости от типа адресации. Исключением является прямая загрузка, в ней не происходит обращения к ячейке, поэтому OF пустой, а в CR(0, 7) -> BR, BR -> DR.

*Безадресные команды* выполняют различные действия без ссылок на ячейку памяти.

*Команды ввода-вывода* осуществляют обмен данными между процессором и внешними устройствами ЭВМ. Всего устройств ввода-вывода 2^8 = 256.

*Команды ветвления* осуществляют работу с регистром данных(DR), являются командами условного/безусловного перехода, реализуют смещение относительно IP.

Для реализации одной команды требуется выполнить определенное количество микрокоманд, каждая из которых инициируется одним тактовым импульсом. Общее число тактовых импульсов, требуемых для выполнения команды, определяет время ее выполнения, называемое **циклом команды**. Цикл команды обычно включает один или несколько **машинных циклов**. Устройство управления базовой ЭВМ может находиться в **5 возможных состояниях**: выборки команды (Instruction fetch), выборки адреса(Address fetch), выборки операнды(Operand fetch), исполнения(Execution), прерывания (Interrupt).

23

**Устройство управления** в БЭВМ реализовано как микропрограммное устройство управления (MCU). Это своего рода простой компьютер, который выполняет программы, состоящие из микроопераций.

Микрооперации управляют изменениями значений вентилей, что задаёт атомарные операции: вычисления в ALU, пересылка данных между регистрами и проверки.

**Исполнение** машинной команды **в MCU** делится на пять циклов:

1. Цикл выборки команды - загрузка команды в регистр команды и её частичное декодирование.

2. Цикл выборки адреса - обработка адресных команд и выборка адреса операнда с учётом режимов адресации.

3. Цикл выборки операнда - размещение второго операнда команды в буфере. Первый операнд — аккумулятор.

4. Цикл исполнения - непосредственное выполнение команды.

5. Цикл прерывания - выполняется, если разрешены прерывания и устройство ввода- вывода готово к обмену данными.

Для обеспечения работы оператора предусмотрена микропрограммная реализация циклов **пультовых операций**:

Ввод адреса: Адрес из клавишного регистра вводится в счётчик команд.

Чтение: Чтение информации по адресу из памяти в буферный регистр, увеличение счётчика команд.

Запись: Запись информации из буферного регистра в память, увеличение счётчика команд.

Пуск: Сброс состояния БЭВМ и переход к выполнению программы.

На панели оператора расположены:

Переключатель «Работа/Останов»: Останавливает программу после каждой команды.

Переключатель «Такт»: Выполняет микрокод по одному такту.



17. Первая ЭВМ: Калькулятор (1)

Типичная ЭВМ состоит из процессора, памяти и устройств ввода-вывода.

«Сердцем» ЭВМ является процессор, в состав которого входят устройство управления выборкой команд из памяти и их выполнением, арифметико-логическое устройство, производящее операции над данными, регистры, осуществляющие временное хранение данных и состояний процессора, схемы для управления и связи с подсистемами памяти и ввода-вывода.

*Устройство ввода* обеспечивает считывание информации с определенных носителей информации и ее представление в форме электрических сигналов, воспринимаемых другими устройствами ЭВМ.

*Устройства вывода* представляют результаты обработки информации в форме, удобной для визуального восприятия. Память ЭВМ включает устройство, обеспечивающее хранение команд и данных. Это устройство состоит из блоков одинакового размера – ячеек памяти, предназначенных для хранения одного слова информации.

Ячейка памяти состоит из элементов памяти, состояние каждого из которых соответствует одной двоичной цифре. Совокупность нулей и единиц, хранящихся в элементах одной ячейки, представляет собой содержимое этой ячейки памяти. В микро ЭВМ используются *безадресные*, *одноадресные* и реже *двухадресные* команды. В одноадресных командах один из операндов выбирается из специального регистра – аккумулятора. В него же заносится и результат операции. Безадресные команды или задают какое-либо действие с устройствами ЭВМ, или используются для работы с операндами, имеющими фиксированное расположение (чаще всего с аккумулятором). В процессе работы ЭВМ последовательно выполняет набор достаточно простых операций: выборку команды, определение ее типа, исполнение команды и определение адреса следующей команды.

Арифметико-логическое устройство (АЛУ) — это ключевой компонент центрального процессора (CPU) или микроконтроллера, который выполняет арифметические и логические операции. АЛУ принимает два операнда и выполняет над ними заданную операцию, такую как сложение, вычитание, умножение, деление, логическое И (AND), логическое ИЛИ (OR), логическое НЕ (NOT) и другие. Результат операции затем передается обратно в регистры или другие компоненты процессора.

**Принципы функционирования простейшей ЭВМ калькулятора.**

Он состоит из двух регистров Х и Y, хранящих результаты ввода пользователя и промежуточных вычислений, АЛУ, которая может выполнять простейшие арифметические и логические операции, шин и управляющих вентилей, осуществляющих передачу данных между функциональными блоками калькулятора, *устройства управления* (УУ), клавиатуры и дисплея.

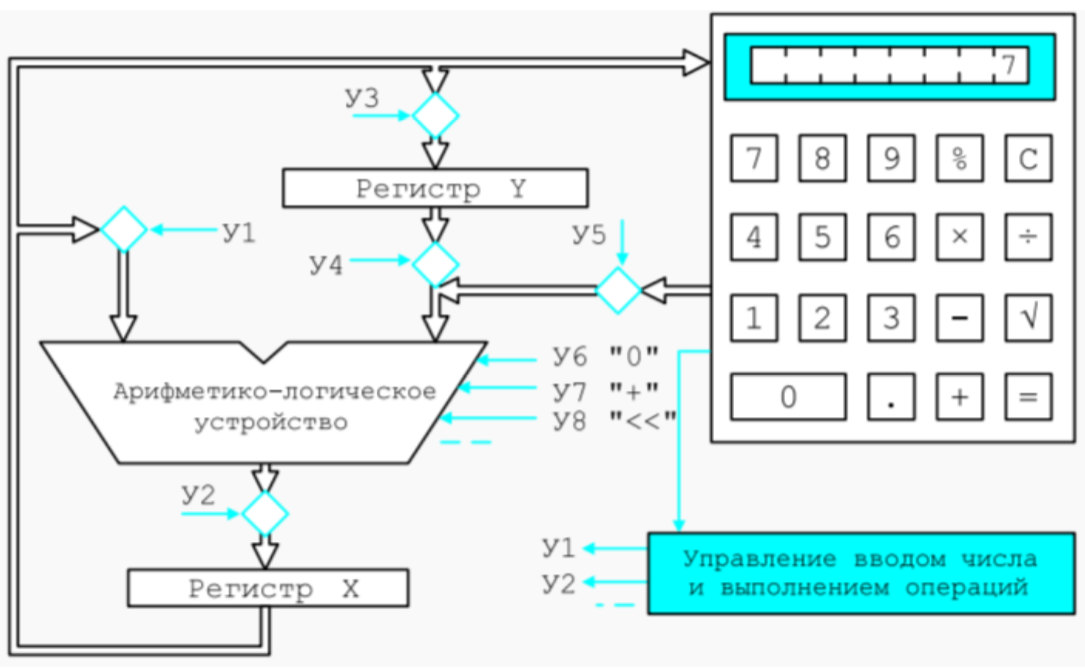
Дисплей постоянно отображает содержимое регистра X. Клавиатура передает значение нажатой клавиши на вентиль У5, каждое нажатие на клавишу запускает УУ, которое в зависимости от текущего состояния ЭВМ формирует последовательность импульсов для выполнения требуемой операции, которая называется циклом импульсов. Каждая группа импульсов выдается последовательно, в моменты, совпадающие с импульсами тактового генератора.

Пример: пользователь вводит первую цифру необходимого ему числа (7). Так как это новая операция, УУ, после своей активации нажатием кнопки 7, выдаст последовательность управляющих импульсов для первой цифры числа. В первую очередь необходимо сохранить предыдущее значение регистра Х в регистре У. Для этого должен быть открыт вентиль, управляющий записью в регистр Ү. Он открывается управляющим сигналом УЗ. После этого необходимо обнулить регистр Х, подготовив его для новой цифры числа, которая была введена с клавиатуры. Для этого должны быть закрыты все вентили, кроме У2 - который осуществляет передачу данных из АЛУ в регистр Y и У6 — который сформирует в АЛУ значение 0. Далее необходимо сложить значение 0 с цифрой с клавиатуры. Для этого содержимое регистра Х поступает на правый вход АЛУ (У1), цифра с клавиатуры на правый вход АЛУ (У5), и выбирается операция сложения (У7). В конце цикла необходимо передать результат сложения в регистр Х (У2), отобразив его, при этом, на дисплее.

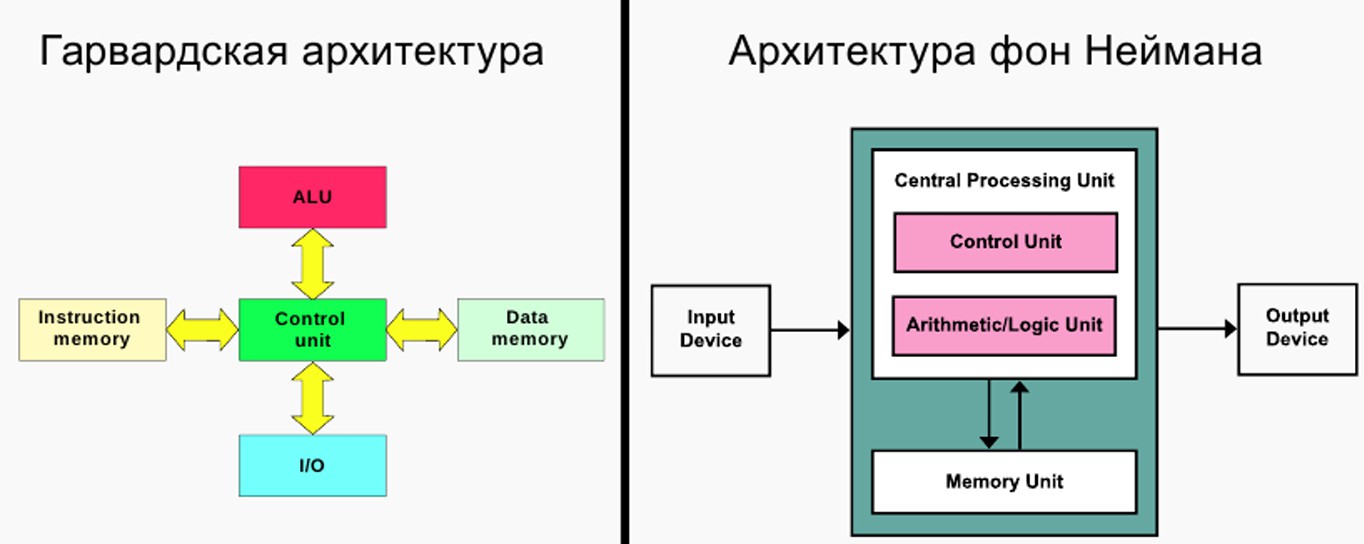
**Процессор** является центральным элементом ЭВМ. Он состоит из нескольких частей:

* Устройство управления – выбирает команды из памяти и выполняет их.
* Арифметико-логическое устройство (АЛУ) – выполняет арифметические и логические операции.
* Регистры – временные хранилища данных и состояний процессора.
* Схемы управления и связи – обеспечивают взаимодействие процессора с памятью и устройствами ввода-вывода.

**Память** включает устройство для хранения команд и данных. Память состоит из ячеек, каждая из которых хранит одно слово информации. Ячейки имеют уникальные адреса, что позволяет процессору обращаться к ним.



18.Архитектура ЭВМ, Гарвардская, Фон Неймана.



**Гарвардская архитектура**

Гарвардская архитектура предполагает разделение памяти данных и памяти инструкций. Это означает, что для доступа к инструкциям и данным используются отдельные шины и устройства ввода-вывода.

Плюсы:

1. Высокая производительность: Параллельный доступ к памяти данных и памяти инструкций позволяет выполнять операции более эффективно.

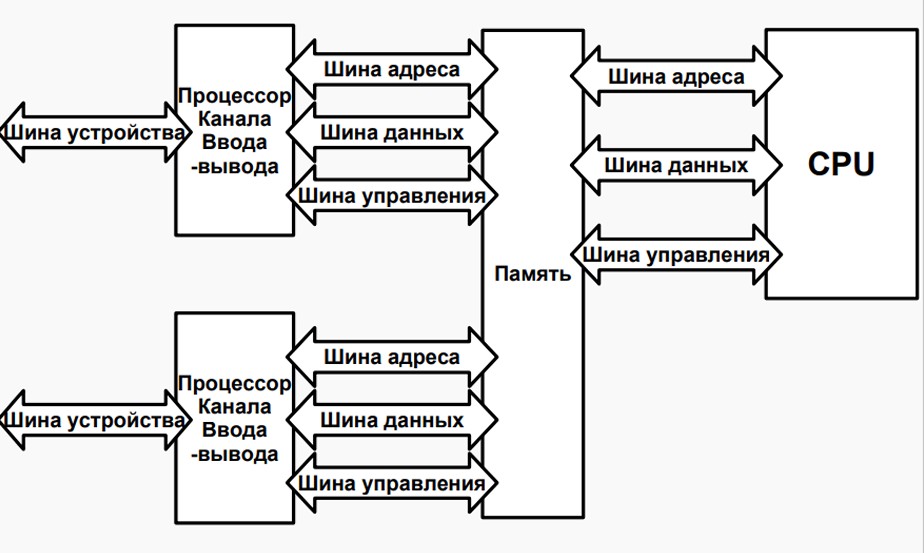
2. Отсутствие конфликтов: Разделение памяти исключает конфликт доступа к памяти между инструкциями и данными.

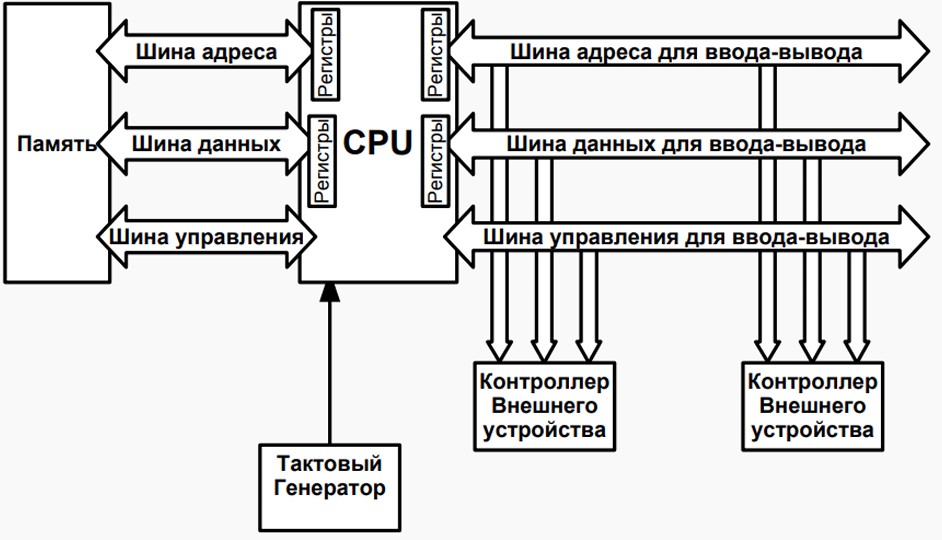
3. Увеличение скорости: Возможность одновременного чтения инструкций и данных увеличивает общую скорость выполнения программы.

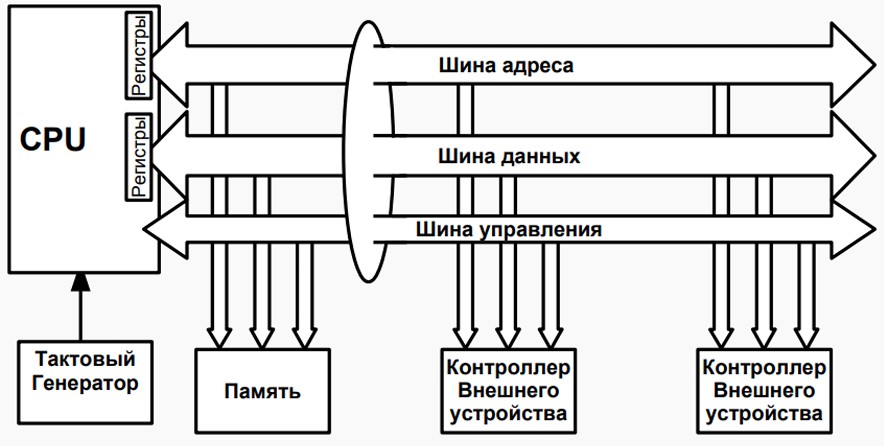
Минусы:

1. Сложность реализации: Наличие двух отдельных шин и памяти усложняет конструкцию и увеличивает стоимость.

2. Гибкость: Менее гибкая в плане изменения программного кода и данных, так как каждая часть имеет свое фиксированное место.







**Архитектура фон Неймана**

Архитектура фон Неймана использует одну общую память для инструкций и данных. Все компоненты компьютера подключены к одной шине.

Плюсы:

1. Простота реализации: Общая шина и память упрощают конструкцию и снижают стоимость производства.

2. Гибкость: Легкость в изменении программного кода и данных, так как они находятся в одной памяти.

Минусы:

1. Конфликт доступа: Процессор не может одновременно выполнять чтение инструкций и данных, что снижает производительность.

2. Узкое место: Чем больше шина, тем больше и ответственность. Шина является узким местом, так как все данные и инструкции проходят через нее, что ограничивает скорость.

Организация обмена данными в архитектурах ЭВМ осуществляется через шины, которые могут быть общими для всех компонентов (фон Нейман) или отдельными для инструкций и данных (Гарвард).

1. Шина данных: Используется для передачи данных между памятью, процессором и внешними устройствами.

2. Шина адреса: Передает адреса ячеек памяти или устройств ввода- вывода.

3. Шина управления: Передает управляющие сигналы, координируя работу всех компонентов.

19. Структура БЭВМ-NG

**БЭВМ** (базовая электронная вычислительная машина) состоит из нескольких функциональных блоков и регистров, обеспечивающих её работу.

Основные компоненты БЭВМ:

1. Память

* Состоит из 2048 ячеек, каждая из которых имеет 16 разрядов.
* Регистр адреса (AR): 11-разрядный регистр, в который помещается адрес памяти.
* Регистр данных (DR): 16-разрядный регистр для чтения или записи данных в/из ячеек памяти.

2. Счётчик команд (IP)

* 11-разрядный регистр, хранящий адрес следующей команды, которая будет выполнена.

3. Арифметико-логическое устройство (ALU)

* Выполняет арифметические и логические операции: сложение, логическое умножение, инверсия и добавление единицы.
* Обрабатывает биты переполнения и переноса при сложении.

4. Буферный регистр (BR) - 16-разрядный регистр для промежуточного хранения данных.

5. Регистр команд (CR) - хранит код команды и декодирует операции во время работы.

6. Аккумулятор (AC) - основной регистр для выполнения всех вычислений с данными.

7. Указатель стека (SP) - 11-разрядный регистр, указывающий на вершину стека, используемого для хранения адресов возвратов и параметров подпрограмм.

8. Клавишный регистр (IR) - 16-разрядный регистр, предназначенный для ввода данных с пульта оператора, включая адреса программ, коды программ и данные.

9. Регистр состояния (PS) - 16-разрядный регистр, хранящий биты, управляющие работой БЭВМ, и признаки результата выполнения программ.

20. Устройство Управления

Устройство управления в БЭВМ реализовано как **микропрограммное устройство управления (MCU)**. Это своего рода простой компьютер, который выполняет программы, состоящие из микроопераций.

Микрооперации управляют изменениями значений вентилей, что задаёт атомарные операции: вычисления в ALU, пересылка данных между регистрами и проверки.

Исполнение машинной команды в MCU делится на пять циклов:

**Циклы команды**

1. Цикл выборки команды: загрузка команды в регистр команды и её частичное декодирование.

2. Цикл выборки адреса : обработка адресных команд и выборка адреса операнда с учётом режимов адресации.

3. Цикл выборки операнда: размещение второго операнда команды в буфере. Первый операнд — аккумулятор.

4. Цикл исполнения: непосредственное выполнение команды.

5. Цикл прерывания: выполняется, если разрешены прерывания и устройство ввода- вывода готово к обмену данными.

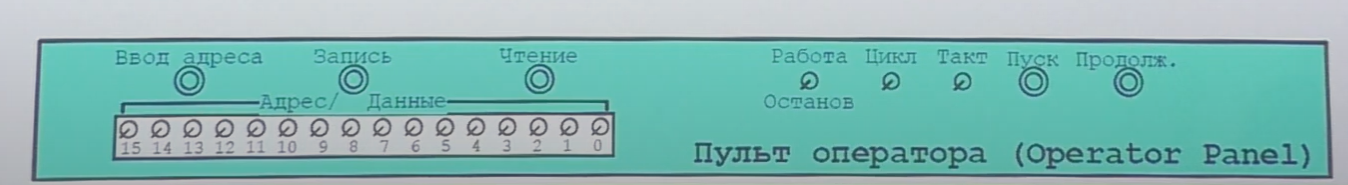
**Циклы пультовы****х операций**

Ввод адреса

Чтение

Запись

Пуск



Микропрограммное устройство управления (Microprogrammed Control Unit, MCU) — это компонент центрального процессора (CPU) в электронных вычислительных машинах (ЭВМ), который использует микропрограммы для управления выполнением инструкций. Микропрограммы представляют собой набор микроопераций, которые определяют, как процессор должен выполнять каждую инструкцию. Этот подход позволяет гибко и эффективно управлять выполнением инструкций, обеспечивая высокую производительность и масштабируемость.

### Основные компоненты микропрограммного устройства управления

1. **Микропрограммное хранилище (Control Store)**:
   * Это память, в которой хранятся микропрограммы. Микропрограммное хранилище обычно реализуется с использованием быстрой памяти, такой как ROM (Read-Only Memory) или PROM (Programmable Read-Only Memory).
2. **Микропрограммный счетчик (Microprogram Counter)**:
   * Это счетчик, который указывает на текущую микрооперацию в микропрограмме. Он используется для последовательного выполнения микроопераций.
3. **Микропрограммный регистр (Microinstruction Register)**:
   * Это регистр, который хранит текущую микрооперацию, извлеченную из микропрограммного хранилища. Микрооперация содержит управляющие сигналы, которые управляют различными частями процессора.
4. **Декодер микроопераций (Microinstruction Decoder)**:
   * Это компонент, который декодирует микрооперации и генерирует управляющие сигналы для различных частей процессора, таких как арифметико-логическое устройство (ALU), регистры и шины данных.

### Принцип работы микропрограммного устройства управления

1. **Извлечение инструкции**:
   * Процессор извлекает инструкцию из основной памяти и помещает ее в регистр инструкций (Instruction Register).
2. **Декодирование инструкции**:
   * Инструкция декодируется, чтобы определить, какая микропрограмма должна быть выполнена. Адрес начальной микрооперации микропрограммы определяется на основе декодированной инструкции.
3. **Выполнение микропрограммы**:
   * Микропрограммный счетчик устанавливается на адрес начальной микрооперации. Микрооперации последовательно извлекаются из микропрограммного хранилища и помещаются в микропрограммный регистр.
4. **Генерация управляющих сигналов**:
   * Декодер микроопераций декодирует микрооперации и генерирует управляющие сигналы, которые управляют различными частями процессора для выполнения инструкции.
5. **Повторение процесса**:
   * Процесс повторяется для каждой микрооперации в микропрограмме, пока инструкция не будет полностью выполнена.

21. Адресуемая память БЭВМ

У ячейки памяти всегда есть адрес. 2048 16разрядных ячеек (свободноадресуемых)

**Адресное пространство** — это набор адресов, который может быть использован процессором для обращения к памяти. Каждой ячейке памяти соответствует определённый адрес.

**Мультиплексор** - устройство, соединяющее несколько входов и один выход.

Оперативная память разбивается на чётное количество банков, которая состоит из страниц. Страница состоит из матриц, которые состоят из ячеек с 1 битом

памяти. Доступ к ячейкам осуществляется с помощью указания адреса строки (RAS - Row Address Strobe) и адреса столбца (CAS - Column Address Strobe).

* На адресную шину подается номер строки, активируется сигнал выборки строки - RАЅ
* Содержимое строки переносится в буферный уселитель данных
* На адресную шину подается номера ячеек, сигнал - CAS, данные поступают на шину данных
* Деактивируются сигналы RAS, CA S

22. АЛУ, коммутатор, блок признаков результата

**Арифметико-логическое устройство** (АЛУ, ALU - Arithmetic-n-Logic Unit) может выполнять арифметические операции, такие как сложение и сложение с учетом переноса, полученного в результате выполнения предыдущей операции, операции логического умножения и инвертирования. Выход из АЛУ подключен к коммутатору.

**Принципы работы АЛУ:**

* Входные данные поступают в АЛУ из регистров или памяти.
* Команда управления определяет, какую операцию нужно выполнить.
* Результат операции записывается в выходной регистр или обратно в память.
* Флаги состояния (перенос, нуль, отрицательное число и переполнение) обновляются на основе результата операции.

**Коммутатор** — это устройство, на которое поступают 18 разрядов результата операции из АЛУ (16-ти разрядный результат сложения и биты, необходимые для формирования признака переноса С), а также предыдущее значение переноса из регистра состояния. Коммутатор выполняет операции прямой и перекрестной передачи информации между байтами слова, осуществления арифметических и циклических побитовых сдвигов влево и вправо, а также для расширения знака младшего байта в старший байт. Информация из коммутатора поступает на шину данных для записи в регистры БЭВМ и на блок установки признаков результата.

1. Структура коммутатора:

* Порты ввода-вывода: обеспечивают соединение с другими устройствами и компонентами.
* Кроссбарная матрица: обеспечивает прямое соединение между любыми двумя портами.
* Контроллер управления: обрабатывает команды по перенаправлению данных.

2. Принципы работы коммутатора:

* Получение данных через один из входных портов.
* Определение направления: контроллер решает, куда перенаправить данные на основе управляющих сигналов.
* Перенаправление данных через кроссбарную матрицу к нужному выходному порту.

3. Сигналы управления коммутатором:

* SHLT, SHLO, SHRT, SHRF: Сигналы сдвига данных влево или вправо.
* LTOL, LTOH, HTOH, HTOL, SEXT: Сигналы для перенаправления данных между разными частями регистра (например, из старшего байта в младший).

23. Форматы команд

Прямая абсолютная адресация

Относительная адресация

Непосредственная загрузка операнда

1. Вентиль УМК (Управляющий Микрокомандами) и Вентиль ОМК (Основной Микрокомандами ):

* Определяют, какой тип микрокоманды будет выполняться.
* Эти вентили выбирают нужную микрокоманду для выполнения.

2. Тактовый генератор:

* Обеспечивает синхронизацию выполнения микрокоманд.
* Генерирует тактовые импульсы, которые используются для управления процессом выполнения микрокоманд.

3. Сравнение бит в УМК и регистре:

* Сравниваются биты микрокоманды и регистра для принятия решений о дальнейшем выполнении.
* Выбор бита для сравнения осуществляется с помощью мультиплексора.

4. Интерпретатор БЭВМ управляет выполнением команд процессора с помощью микрокоманд. Основные этапы работы:

* 256 ячеек для хранения микрокоманд, включая резервные.
* Содержит горизонтальные микрокоманды: включают множество управляющих сигналов, которые могут выполняться параллельно.

**Циклы выполнения команд**:

1. Цикл выборки команд: извлечение следующей команды из памяти.

2. Цикл выборки адреса операнда и обработки режимов адресации: определение адреса операнда с учетом режимов адресации.

3. Цикл выборки операнда: извлечение операнда из памяти или регистра.

4. Цикл исполнения: декодирование и исполнение различных типов команд:

* + Адресные команды: Команды, требующие доступа к памяти.
  + Команды ветвлений: Команды, изменяющие поток выполнения программы.
  + Безадресные команды: Команды, не требующие указания адреса (например, команды работы с аккумулятором).

5. Декодирование и исполнение команд ввода-вывода: управление операциями ввода-вывода с использованием специальных микрокоманд.

6. Цикл прерывания: обработка прерываний, поступающих от внешних устройств или программ.

7. Пультовые операции: управление работой процессора с помощью панели оператора (ввод, вывод, диагностика).

8. Свободные ячейки для: хранение микрокоманд, необходимых для выполнения специфических операций:

* + Арифметическая команда
  + Команда перехода
  + Безадресная команда

24. Адресные команды

**Адресные команды** предписывают машине производить действия с ячейкой памяти, адрес которой определяется исходя из адресной части команды, состоящей из 12 бит (биты 0..11). КОП (биты 12..15) принимает значения от 0х2 до 0хE и задает операцию.

Режимы адресации в адресных командах. Для адресных команд предусмотрено два различных формата:

1) С *прямой абсолютной адресацией* - в бите 11 у этих команд всегда 0, а в адресной части (битах с 0 по 10) записано абсолютное значение адреса операнда (т.е. номер ячейки в адресном пространстве) в памяти. При выполнении операции команда непосредственно обращается по заданному адресу выбирая или записывая операнд.

2) С *относительной адресацией* — 11-й бит содержит 1, а биты 8-10 режим адресации. В биты 0-7 записано смещение, которое используется для вычисления адреса операнда в памяти с помощью прибавления смещения к значению ІР. Смещение может быть и положительным и отрицательным, позволяя адресовать 127 ячеек до и 128 ячеек после текущей команды в памяти. Подчеркнем, что смещение 0 будет указывать на следующую за командой ячейку. Это происходит потому, что к моменту вычисления адреса операнда, счетчик команд уже увеличен на 1 в результате исполнения команды.

25. Безадресные команды

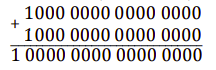
**Безадресные команды** выполняют различные действия без ссылок на ячейку памяти. Например, команда CLA предписывает ЭВМ очистить аккумулятор (записать в АС код нуля). Это команда обработки операнда, расположенного в конкретном месте, "известном" машине. Другой пример безадресной команды - команда HLT. Формат команды состоит из значения 0 в КОП и расширения кода операции (биты 0-11), которое задает необходимую операцию без использования явного указания ячейки памяти. Следует отметить, что безадресные операции могут использовать ячейки памяти неявно, например, команды POP или RET.

26. Представление чисел: фиксированная точка

Целые двоичные числа без знака можно использовать для представления нуля и целых положительных чисел. При размещении таких чисел в одном 16-разрядном слове они могут изменяться от (0000 0000 0000 0000)2 = (000 0)16 = 0 до (1111 1111 1111 1111)2 = (FFFF)16 = 2^16 - 1 = 65535. Такая запись называется **прямым кодом** числа. Подобные числа (так же как и рассмотренные ниже двоичные числа со знаком) относятся к числам с **фиксированной запятой**, разделяющей целую и дробную части числа. В числах, используемых в базовой ЭВМ, положение запятой строго фиксировано после младшего бита слова. Целые двоичные числа со знаком используются тогда, когда необходимо различать положительные и отрицатеx`льные числа. В современных ЭВМ для представления целых чисел со знаком используется дополнительный код, в котором старший бит формата определяет знак числа: 0 - для положительных чисел и 1 - для отрицательных чисел. При этом дополнительный код положительного числа совпадает с его прямым кодом.

А для представления отрицательного числа в дополнительном коде производится инвертирование прямого кода модуля числа (получение обратного кода числа) и добавление к результату единицы. Такая же операция используется при изменении знака числа, представленного в дополнительном коде. Использование дополнительного кода упрощает конструкцию ЭВМ, так как при сложении двух таких чисел, имеющих разные знаки, не требуется переходить к операциям вычитания меньшего (по модулю) числа из большего и присвоения результату знака большего числа.

Кроме того, одной и той же схемой сумматора можно воспользоваться для выполнения операций над знаковым и беззнаковым представлением числа. Признаком выхода за границы разрядной сетки для беззнакового представления числа является перенос в старший разряд (бит C - Carry). Например, при сложении:



27. Представление беззнаковых целых чисел

Если мы зафиксируем двоичную точку числа непосредственно за 0-вым разрядом и используем все 16 разрядов машинного слова для хранения значения числа, задав тем область представления, то **беззнаковое** представление можно использовать для представления нуля и натуральных чисел, не превышающих 65535. Подобные числа (так же как и рассмотренные ниже двоичные числа со знаком) называются числами с фиксированной точкой, разделяющей целую и дробную части числа. При размещении таких чисел в одном 16-разрядном слове они могут изменяться от (0000 0000 0000 0000)2 = (0000)16 = 0 до (1111 1111 1111 1111)2 = (FFFF)16 = 216 - 1 = 65535. Такая запись называется прямым кодом числа.

28. Представление знаковых целых чисел

Целочисленное **знаковое** представление числа используются тогда, когда необходимо различать положительные и отрицательные числа. В современных ЭВМ для представления целых чисел со знаком используется дополнительный код, в котором старший бит формата определяет знак числа: 0 - для положительных чисел и 1 - для отрицательных чисел. При этом дополнительный код положительного числа совпадает с его прямым кодом. А для представления отрицательного числа в дополнительном коде производится инвертирование прямого кода модуля числа (получение обратного кода числа) и добавление к результату единицы. Такая же операция используется при изменении знака числа, представленного B дополнительном коде. Дополнительный код определен для любой системы счисления, включая, например, десятичную.

29. Представление знаковых чисел: дополнительный код

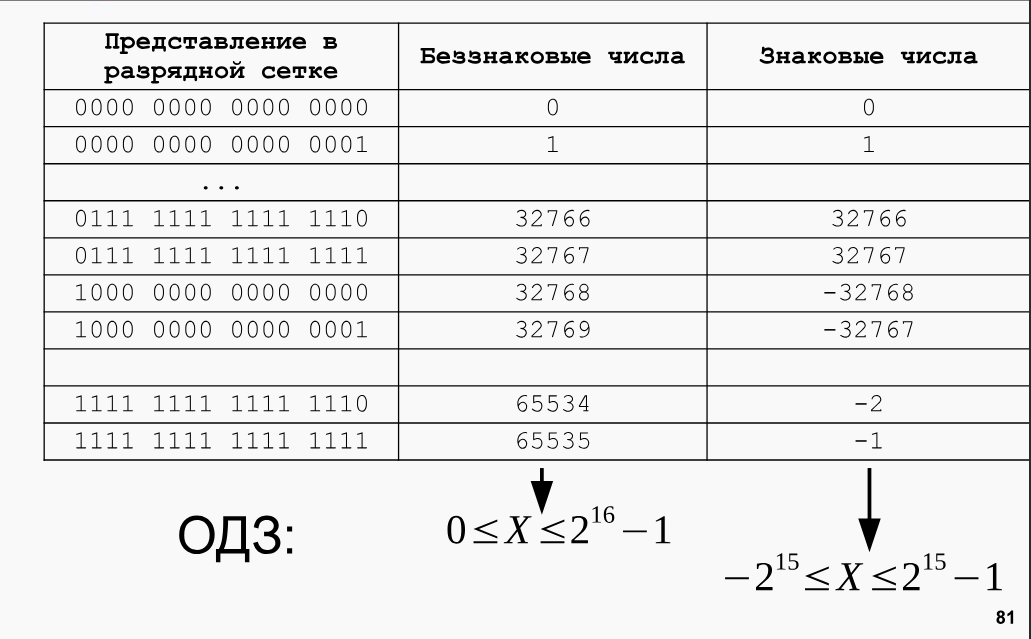
Использование **дополнительного кода** упрощает конструкцию ЭВМ, так как при сложении двух таких чисел, имеющих разные знаки, не требуется переходить к операциям вычитания меньшего (по модулю) числа из большего и присвоения результату знака большего числа. Кроме того, одной и той же схемой сумматора можно воспользоваться для выполнения операций над знаковым и беззнаковым представлением числа. Признаком выхода за границы разрядной сетки для беззнакового представления числа является перенос в старший разряд (бит С - Carry). Признаком переполнения разрядной сетки для знакового представления является бит переполнения (OVerflow). Рассмотрим возникновение этих ситуаций на примере представления чисел в четырехразрядной сетке (рис. В.4).

30. Перенос, Переполнение

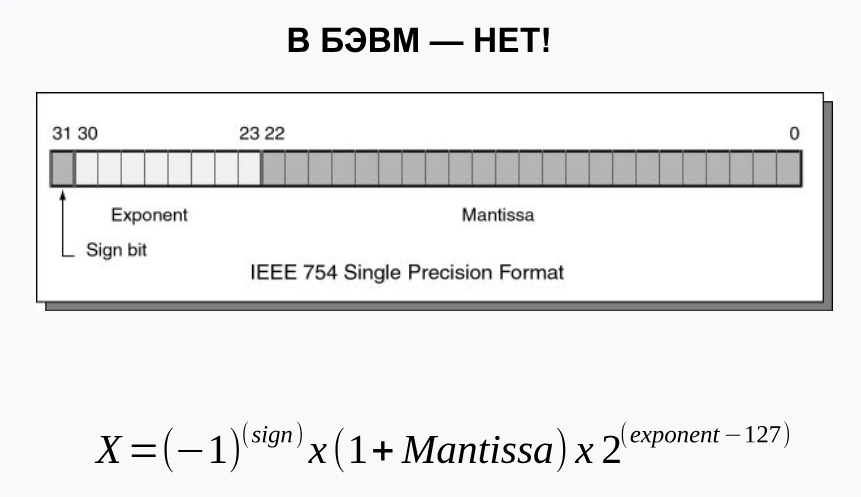
**Флаг переноса** (C - Carry) выступает в качестве продолжения аккумулятора и заполняется при выходе результата за границу 16-ти разрядного слова. При выполнении арифметических операций и операций сдвига в него попадает выход Сnew коммутатора. Флаг переноса необходимо контролировать при выполнении арифметических операций с беззнаковыми числами.

**Флаг переполнения** (V oVerflow) сигнализирует о переполнении разрядной сетки при операциях АЛУ со знаковыми числами, и формируется как операция сложения по модулю 2 поразрядных переносов из 14 в 15 разряд и из 15 в 16 разряд АЛУ, т.е. на выходе коммутатора осуществляется операция V = С14 + Cnew.

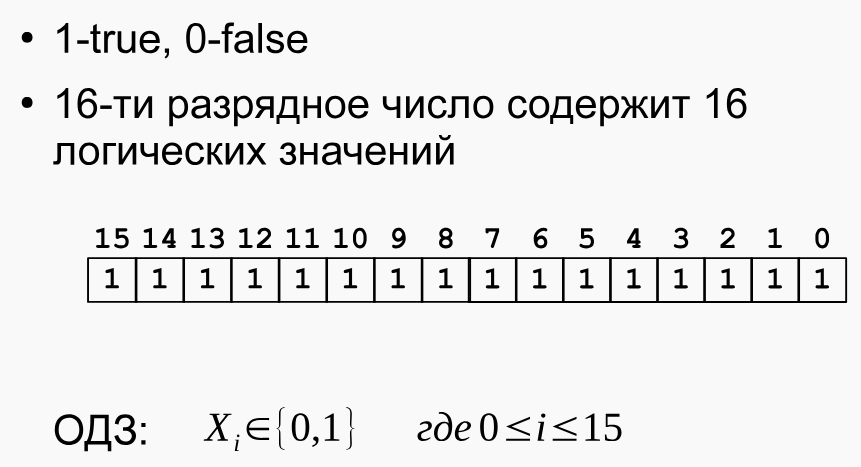
31. БЭВМ: представление чисел



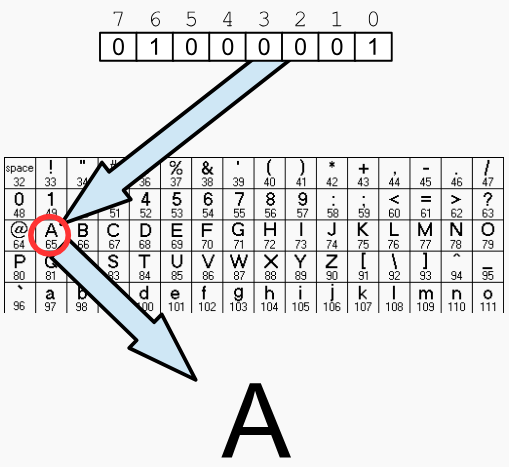
32. Представление чисел с плавающей точкой



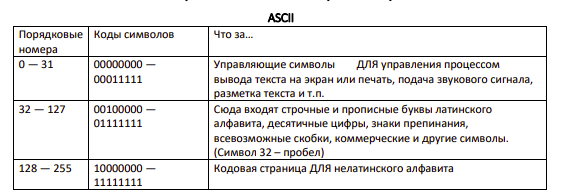
33. Представление логической информации



34. Представление символьной и текстовой информации



35. Символы: ASCII

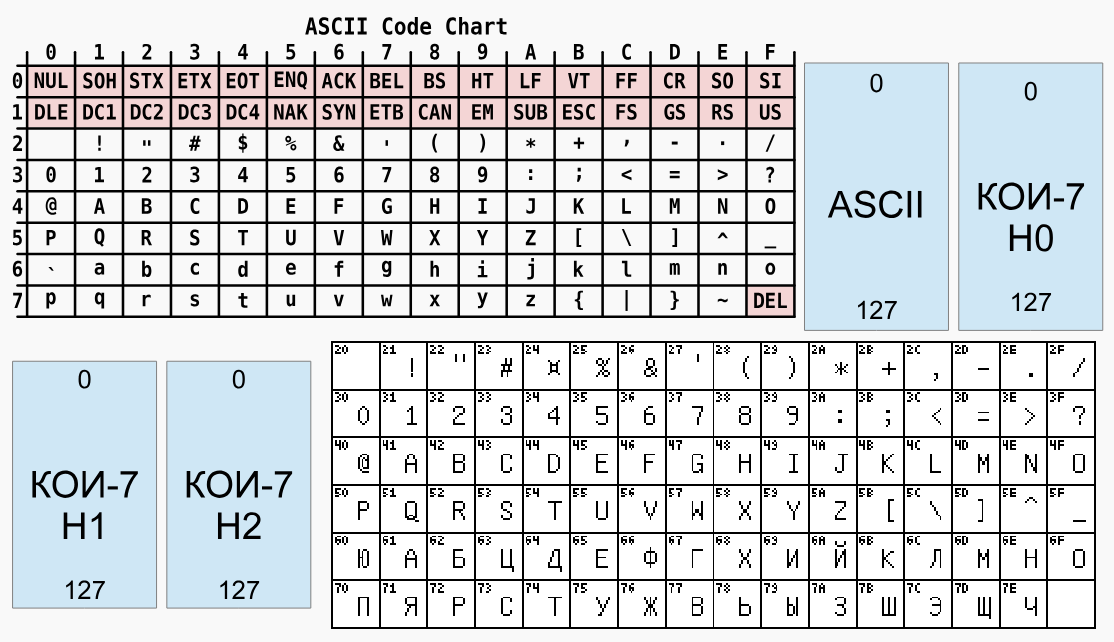


ASCII — таблицы кодировок, в которых содержатся основные символы (английский алфавит, цифры, знаки препинания, символы национальных алфавитов(свои для каждого региона), служебные символы) и длина кода каждого символа n = 8 бит.

7 бит:

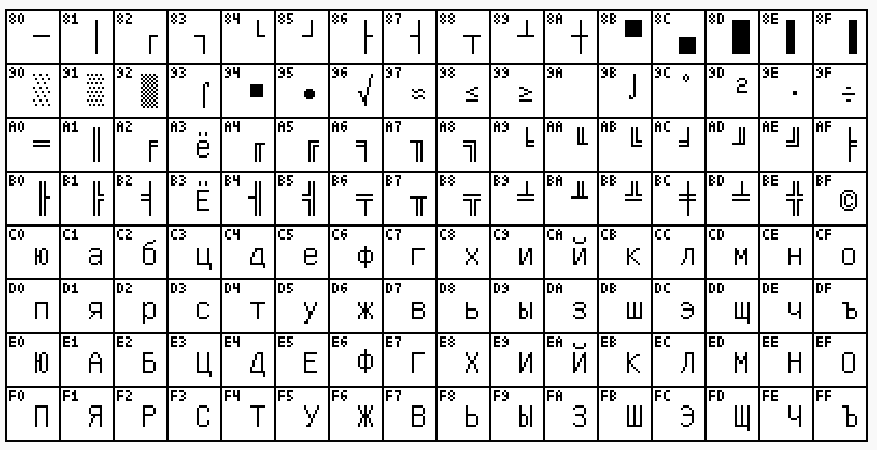
ASCII — первая кодировка, пригодная для работы с текстом. Помимо маленьких букв английского алфавита и служебных символов, содержит большие буквы английского языка, цифры, знаки препинания и другие символы (при этом старший бит использовался для контроля четности битов передаваемого по каналом связи символа).

36. Символы: ASCII (КОИ-7H0)КОИ-7H1 (РУС),КОИ-7H2 (Mix)

КОИ8-R — первая русская кодировка. Символы кириллицы расположены не в алфавитном порядке. Их разместили в верхнюю половину таблицы так, чтобы позиции кириллических символов соответствовали их фонетическим аналогам в английском алфавите. Это значит, что даже при потере старшего бита каждого символа, например, при проходе через устаревший семибитный модем, текст остается "читаемым" (отсюда и появилось понятие транслита).

37. Символы: КОИ-8

— восьмибитовая кодовая страница, совместимая с ASCII Разработчики КОИ-8 поместили символы русского алфавита в верхней части кодовой таблицы таким образом, что позиции символов кириллицы соответствуют их фонетическим аналогам в английском алфавите из нижней части таблицы. Это означает, что если в тексте, написанном в КОИ-8, убрать восьмой бит каждого символа, то получится «читаемый» текст, подобный транслиту. Например, слова «Русский Текст» превратятся в «rUSSKIJ tEKST». Из‑за этого символы кириллицы расположены не в алфавитном порядке.



38. Символы: ISO8859-5 (ГОСТ-основная)

— 8-битная кодовая страница из семейства кодовых страниц стандарта ISO-8859 для представления кириллицы. Нижняя часть таблицы кодировки полностью соответствует кодировке ASCII(за такое можно и бан словить, давно ASCII стала 8ми битной? совпадает начало, потому что ASCII 7 ми битная). Основной недостаток - отсутствие некоторых символов, такие как тире (—), кавычки-ёлочки («»), градус (°), поэтому в России почти не использовалась.

ISO 8859 — первая кодировка, в которой стало возможно использовать символы национальных алфавитов.

39. Символы: WIN1251

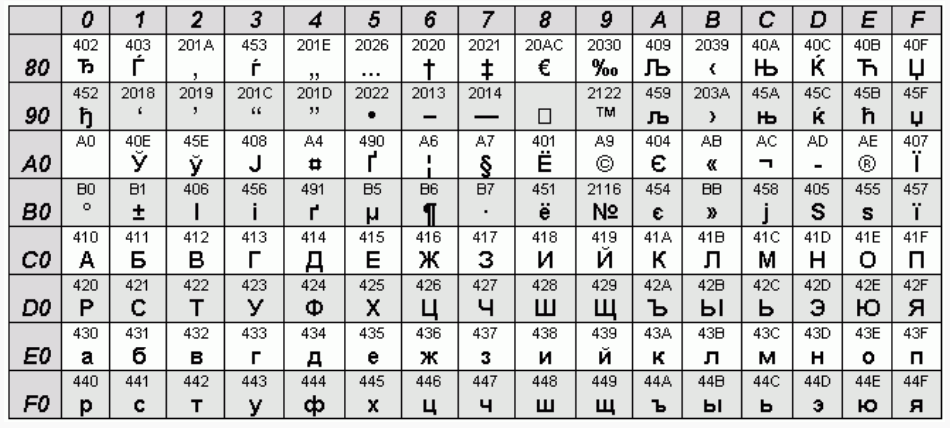
— набор символов и кодировка, являющаяся стандартной 8-битной кодировкой для русских версий Microsoft Windows до 10-й версии. Windows-1251 как и KOI8-R выгодно отличается от других 8‑битных кириллических кодировок (таких как CP866 и ISO 8859-5) наличием практически всех символов, использующихся в русской типографике для обычного текста Минусы:

1. Строчная буква «я» имеет код 0xFF (255 в десятичной системе), что совпадает со служебным символом в некоторых других кодировках, например в СР437 это «неразрывный пробел»

2. Отсутствуют символы псевдографики, имеющиеся в CP866 и KOI8

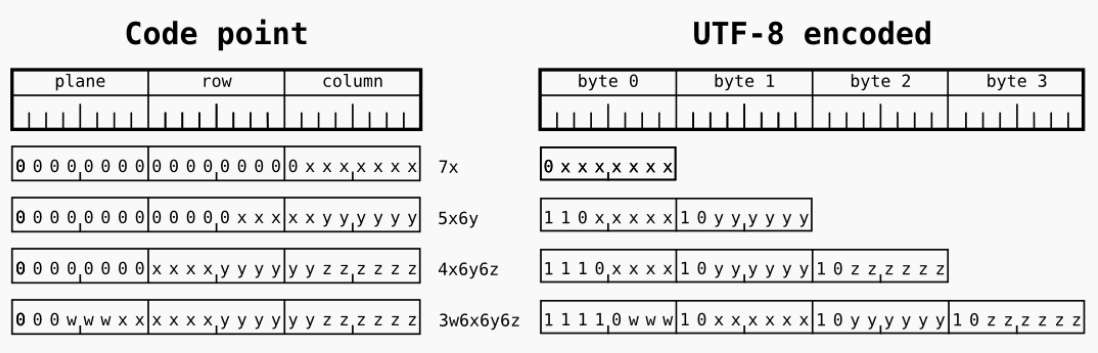
Windows-1251 — русская кодировка, использовавшаяся в русскоязычных версиях операционной системы Windows в начале 90-х годов.

Кириллические символы идут в алфавитном порядке. Содержит все символы, встречающиеся в типографике обычного текста (кроме знака ударения).



40. Символы: UNICODE, UTF-8

— одна из общепринятых 8-битных и стандартизированных кодировок текста, которая позволяет хранить символы Юникода, используя переменное количество байт (от 1 до 6) Совместимость с ASCII — любые их 7-битные символы отображаются как есть, а остальные выдают пользователю мусор (шум). Поэтому в случае, если латинские буквы и простейшие знаки препинания (включая пробел) занимают существенный объём текста, UTF-8 даёт выигрыш по объёму по сравнению с UTF-16.



Кодирование UTF-8

1. Если размер символа в кодировке UTF-8 = байт Код имеет вид (0aaa aaaa), где «0» — просто ноль, остальные биты «a» — это код символа в кодировке ASCII; 2. Если размер символа в кодировке в UTF-8 байт (то есть от до ):

2.1 Первый байт содержит количество байт символа, закодированное в единичной системе счисления;

2 — 11

3 — 111

4 — 1111

5 — 1111 1

6 — 1111 11

2.2 «0» — бит терминатор, означающий завершение кода размера

2.3 далее идут значащие байты кода, которые имеют вид (10xx xxxx), где «10» — биты признака продолжения, а «x» — значащие биты. В общем случае варианты представления одного символа в кодировке UTF-8 выглядят так:

(1 байт) 0aaa aaaa

(2 байта) 110x xxxx 10xx xxxx

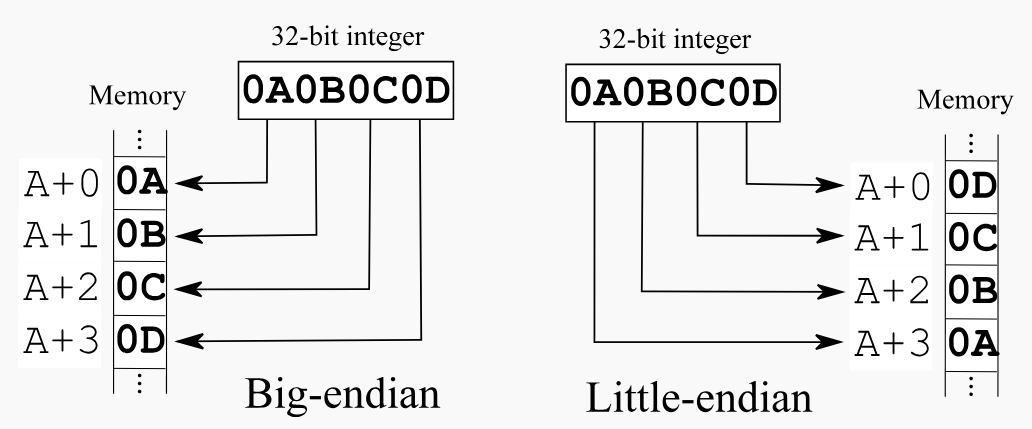
(3 байта) 1110 xxxx 10xx xxxx 10xx xxxx

(4 байта) 1111 0xxx 10xx xxxx 10xx xxxx 10xx xxxx

(5 байт) 1111 10xx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx

(6 байт) 1111 110x 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx

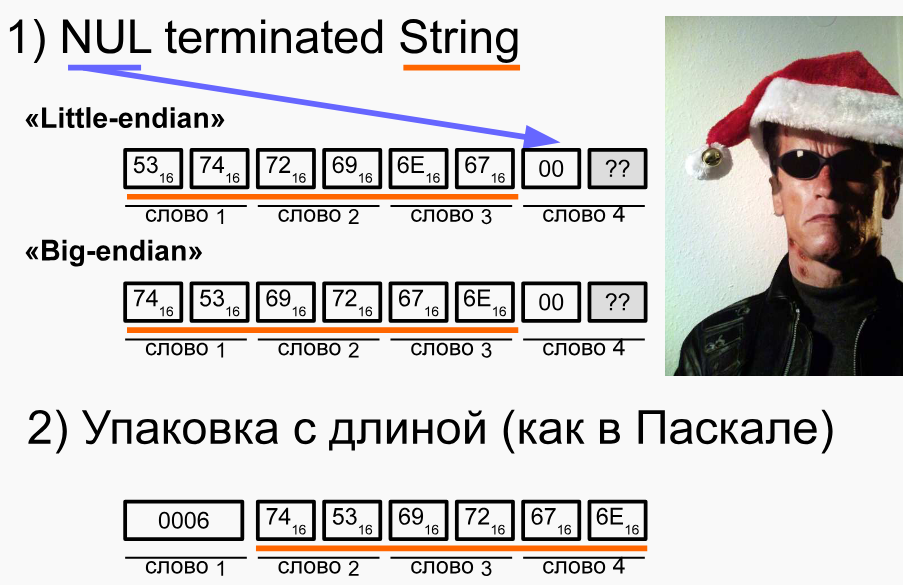
41. Big-endian и Little-endian (архитектура процессора)



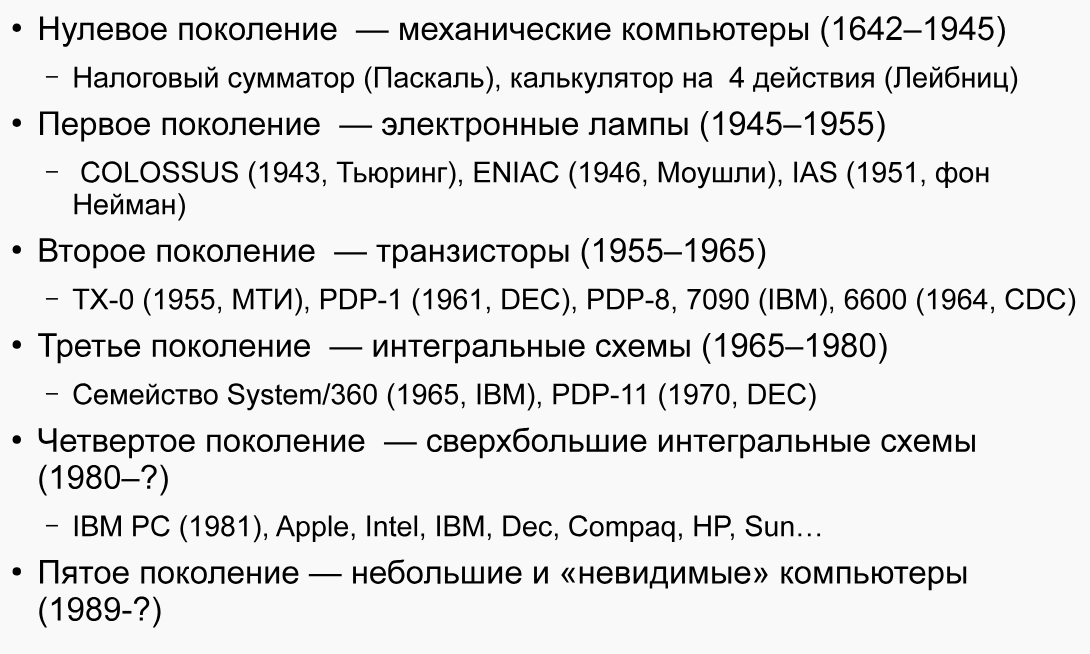
Big-endian и Little-endian — это два способа представления многобайтовых данных в памяти компьютера. Они определяют порядок байтов в многобайтовых числах, таких как 16-битные, 32-битные или 64-битные числа. Есть еще мидл – ендн.

Big-endian и Little-endian — это два способа представления многобайтовых данных в памяти компьютера. Они определяют порядок байтов в представлении чисел и других данных. Различие между ними заключается в том, какой байт (старший или младший) хранится в памяти первым.

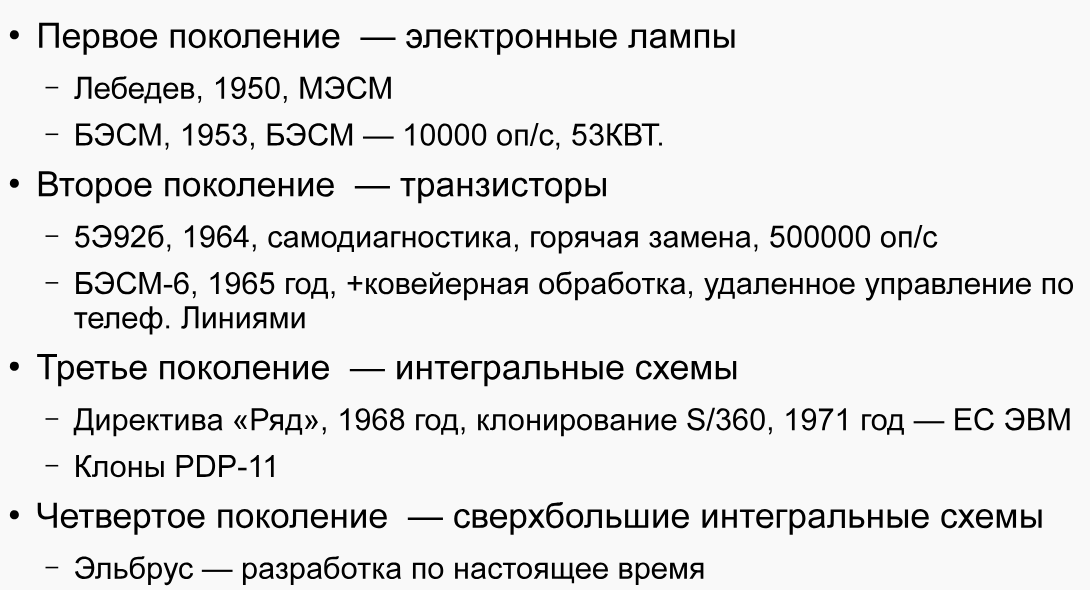
42. Представление строк



43. История развития ЭВМ



44. История развития ЭВМ в СССР/России



45. Канальная организация



В ЭВМ с канальной организацией процессор освобожден от организации ввода-вывода. Канал представляет собой специализированный процессор, который выполняет канальную программу, состоящую из канальных команд, загружаемых операционной системой и хранящихся в оперативной памяти. Управление контроллерами внешних устройств и обмен данными осуществляется каналом. Наличие нескольких трактов передачи данных повышает скорость обмена. Все это дает возможность производить обмен данными с внешними устройствами параллельно с основной вычислительной работой центрального процессора. Одной из первых машин с каналами была ЭВМ второго поколения IBM-704, а также машины семейства IBM-360/370.

### 4.2.1 Каналы

Помимо уже знакомого набора устройств (центральный процессор, память, устройства ввода-вывода) в состав ЭВМ с канальной организацией входят устройства, называемые каналами. *Канал* — это специализированный процессор, осуществляющий всю работу по управлению контроллерами внешних устройств и обмену данными между основной памятью и внешними устройствами. Устройства группируются по характерной скорости и подключаются к соответствующим каналам. «Быстрые» устройства (например, накопители на магнитных дисках) подсоединяются к *селекторным каналам*. Такое устройство получает селекторный канал в монопольное использование на все время выполнения операции обмена данными.

46. Раздельные шины

**Шина** - электрическая цепь, соединяющая регистр с другим регистром или иным устройством ЭВМ. Шина состоит из параллельных проводов, каждый из которых предназначен для передачи соответствующего регистра. Также шина содержит несколько дополнительных проводов, используемых для передачи сигналов синхронизации и управления. Шины служат для передачи информации лишь в направлении, обозначенном стрелкой на шине. Специальные схемы позволяют в одни моменты времени передавать информацию по шине в одну сторону, а в другие – в обратном направлении, т.е. организовать двунаправленную шину.

Раздельные шины в электронных вычислительных машинах (ЭВМ) представляют собой архитектуру, в которой используются отдельные шины для передачи данных, адресов и управляющих сигналов. Это позволяет улучшить производительность и гибкость системы по сравнению с архитектурами, использующими общую шину для всех типов передачи.

### Основные компоненты раздельных шин

1. **Шина данных**:
   * Предназначена для передачи данных между различными компонентами системы, такими как процессор, память и устройства ввода-вывода.
   * Ширина шины данных определяет количество бит, которые могут быть переданы одновременно. Например, 32-битная шина данных может передавать 32 бита за один такт.
2. **Шина адресов**:
   * Используется для передачи адресов, которые указывают на конкретные ячейки памяти или устройства ввода-вывода.
   * Ширина шины адресов определяет максимальный объем адресуемой памяти. Например, 32-битная шина адресов может адресовать до 4 ГБ памяти.
3. **Шина управления**:
   * Передает управляющие сигналы, такие как сигналы чтения/записи, сигналы прерываний и другие сигналы, необходимые для координации работы различных компонентов системы.
   * Управляющие сигналы обеспечивают синхронизацию и управление передачей данных и адресов.

47. Общие шины

Общая шина — это шина, которая используется для передачи различных типов сигналов, таких как данные, адреса и управляющие сигналы, по одной и той же физической линии. Это означает, что в любой момент времени по шине может передаваться только один тип сигнала.

#### Преимущества общих шин:

1. **Простота реализации**:
   * Общие шины проще в реализации, так как требуют меньше проводов и логики для управления передачей данных.
2. **Экономия ресурсов**:
   * Общие шины экономят пространство на печатной плате и уменьшают количество необходимых контактов, что может быть важно в компактных устройствах.

#### Недостатки общих шин:

1. **Ограниченная производительность**:
   * Поскольку по шине может передаваться только один тип сигнала в любой момент времени, это может ограничивать общую производительность системы.
2. **Конфликты и задержки**:
   * Возможны конфликты и задержки при передаче данных, особенно в системах с высокой нагрузкой на шину.

### Мультиплексирование шин

Передавать адрес и данные по одинаковым проводам

Мультиплексирование шин — это метод, при котором несколько сигналов передаются по одной и той же физической шине путем разделения времени или частоты. Это позволяет использовать одну и ту же шину для передачи различных типов данных, адресов и управляющих сигналов.

#### Преимущества мультиплексирования шин:

1. **Экономия ресурсов**:
   * Мультиплексирование позволяет использовать меньшее количество проводов для передачи различных типов сигналов, что экономит пространство и ресурсы.
2. **Гибкость**:
   * Мультиплексирование позволяет легко добавлять новые устройства и компоненты в систему, так как одна шина может обслуживать несколько устройств.

#### Недостатки мультиплексирования шин:

1. **Сложность реализации**:
   * Мультиплексирование требует дополнительной логики для управления передачей сигналов и разделения времени или частоты, что увеличивает сложность системы.
2. **Задержки**:
   * Мультиплексирование может вводить дополнительные задержки, особенно если количество устройств или объем передаваемых данных велик.

### Примеры использования

1. **Общие шины**:
   * Общие шины часто используются в простых микроконтроллерах и встраиваемых системах, где важна экономия ресурсов и простота реализации.
2. **Мультиплексирование шин**:
   * Мультиплексирование шин используется в более сложных системах, таких как современные процессоры и системы на кристалле (SoC), где важна гибкость и экономия пространства.

48. Мультиплексирование шин

49. Мультипроцессорность:UMA - Uniform Memory Access

Система UMA представляет собой архитектуру многопроцессорных систем с общей памятью. Все процессоры обращаются к единой памяти через межсоединительную сеть. Это позволяет каждому процессору иметь одинаковое время доступа к памяти и скорость доступа.

1. Структура:

Единая память: Все процессоры обращаются к одной и той же памяти.

Межсоединительная сеть: Может использовать одну шину, несколько шин или коммутатор для соединения процессоров с памятью.

2. Преимущества:

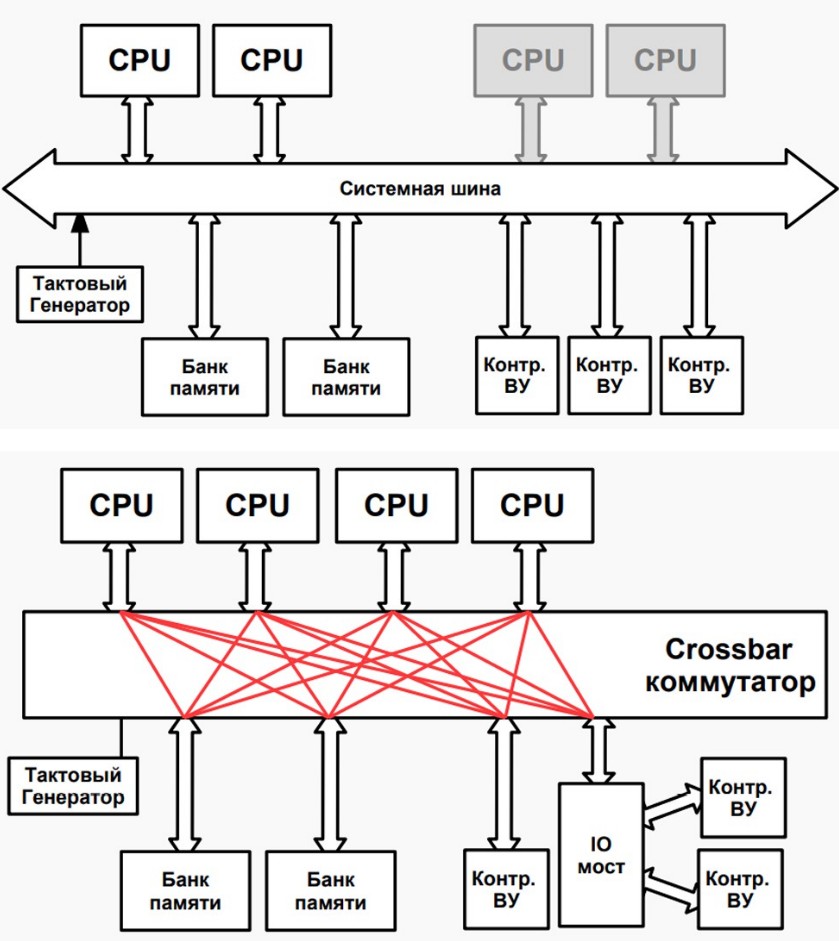
Равный доступ: Одинаковое время доступа к памяти для всех процессоров.

Простота программирования: Легче разрабатывать приложения, так как не нужно учитывать различия в доступе к памяти.

3. Недостатки:

Ограниченная пропускная способность: Ограничена пропускной способностью шины или сети.

Медленная работа: Меньшая скорость по сравнению с NUMA, особенно в параллельных системах.



50. Мультипроцессорность: Коммутатор

Коммутатор управляет потоком данных между различными частями процессора.

1. Структура коммутатора:

Коммутатор включает в себя множество мультиплексоров, которые позволяют направлять данные между различными

частями процессора.

2. Сигналы управления коммутатором:

SHLT, SHLO, SHRT, SHRF: Сигналы сдвига данных влево или вправо.

LTOL, LTOH, HTOH, HTOL, SEXT: Сигналы для перенаправления данных между разными частями регистра (например, из старшего байта в младший).

51. Мультипроцессорность: NUMA – Non Uniform Memory Access

NUMA также является архитектурой многопроцессорных систем, но каждый процессор имеет свою выделенную память. Эти небольшие части памяти объединяются в единое адресное пространство, что позволяет получить доступ к любой ячейке памяти, используя физический адрес.

1. Структура:

Выделенная память: Каждый процессор связан с определенной частью памяти.

Единое адресное пространство: Все части памяти объединяются в одно адресное пространство.

2. Преимущества:

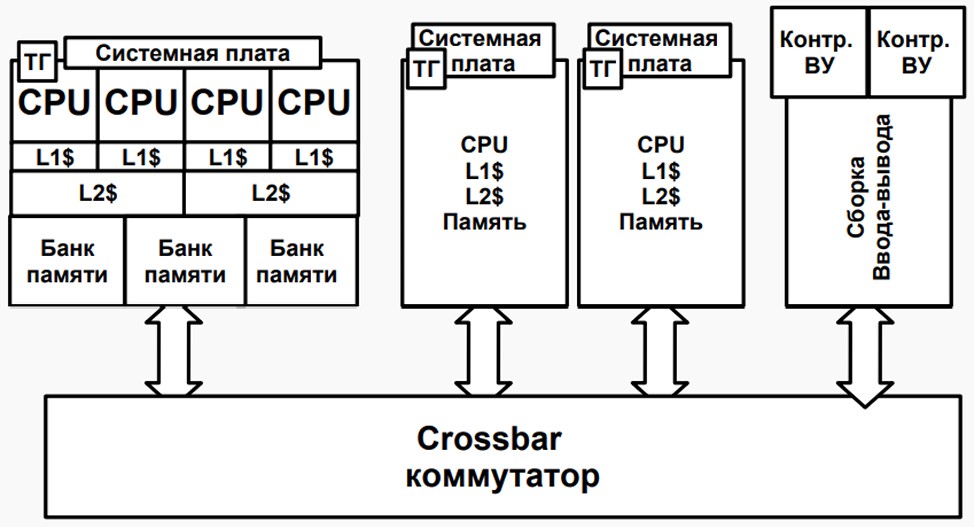
Высокая пропускная способность: Более высокая пропускная способность по сравнению с UMA.

Быстрый доступ: Уменьшенное время доступа к памяти, близкой к процессору.

3. Недостатки:

Сложность программирования: Необходимость учитывать различия в доступе к памяти.

Изменяющееся время доступа: Время доступа к памяти зависит от расстояния до памяти.



52. Современные коммерческие процессоры

Современные процессоры имеют сложную и многоуровневую структуру. Рассмотрим основные характеристики и виды процессоров:

Разрядность адреса и данных

16/32/64 бита: Процессоры могут работать с данными и адресами различной длины (16, 32 или 64 бита), что влияет на их производительность и возможности.

Тактовые частоты

500 МГц - 5 ГГц: Современные процессоры работают на тактовых частотах от 500 мегагерц до 5 гигагерц, что определяет скорость выполнения инструкций.

Многопроцессорные и многоядерные системы

Многопроцессорные: Системы могут иметь от 1 до 100 и более процессоров.

Многоядерные: Один процессор может содержать от 1 до 16 ядер, каждое из которых способно выполнять задачи независимо.

Оперативная память

От 1 ГБ до терабайтов ОЗУ: Современные компьютеры могут иметь от одного гигабайта до нескольких терабайтов оперативной памяти, что позволяет обрабатывать большие объемы данных.

53. CISC, RISC, VLIW

**CISC** - Концепция проектирования процессоров, которая характеризуется следующим набором свойств: · большим числом различных по формату и длине команд; · введением большого числа различных режимов адресации; 40 · обладает сложной кодировкой инструкции. Процессору с архитектурой CISC приходится иметь дело с более сложными инструкциями неодинаковой длины. Выполнение одиночной CISC-инструкции может происходить быстрее, однако обрабатывать несколько таких инструкций параллельно сложнее. Облегчение отладки программ на ассемблере влечет за собой загромождение узлами микропроцессорного блока. Для повышения быстродействия следует увеличить тактовую частоту и степень интеграции, что вызывает необходимость совершенствования технологии и, как следствие, более дорогого производства. Для CISC-архитектуры типичны: • наличие в процессоре сравнительно небольшого числа регистров общего назначения; • большое количество машинных команд, некоторые из них аппаратно реализуют сложные операторы ЯВУ; • разнообразие способов адресации операндов; • множество форматов команд различной разрядности; • наличие команд, где обработка совмещается С обращением к памяти.

**RISC**

Процессор с сокращенным набором команд. Система команд имеет упрощенный вид. Все команды одинакового формата с простой кодировкой. Обращение к памяти происходит посредством команд загрузки и записи, остальные команды типа регистр-регистр. Команда, поступающая в CPU, уже разделена по полям и не требует дополнительной дешифрации. Команда меньше загромождает ОЗУ, CPU дешевле. Программной совместимостью указанные архитектуры не обладают. Отладка программ на RISC более сложна. Поскольку RISC-инструкции просты, для их выполнения нужно меньше логических элементов, что в конечном итоге снижает стоимость процессора. Но большая часть программного обеспечения сегодня написана и откомпилирована специально для CISC-процессоров фирмы Intel. Для использования архитектуры RISC нынешние программы должны быть перекомпилированы, а иногда и переписаны заново.

**VLIW** - архитектура процессоров с несколькими вычислительными устройствами. Характеризуется тем, что одна инструкция процессора содержит несколько операций, которые должны выполняться параллельно, базируется на RISC-архитектуре. – Несколько инструкций, упакованных в одну команду – Упаковка операций в инструкцию ложится на компилятор. Идея VLIW базируется на том, что задача эффективного планирования параллельного выполнения нескольких команд возлагается на «разумный» компилятор. Такой компилятор вначале исследует исходную программу с целью обнаружить все команды, которые могут быть выполнены одновременно, причем так, чтобы это не приводило к возникновению конфликтов. В процессе анализа компилятор может даже частично имитировать выполнение рассматриваемой программы. На следующем этапе компилятор пытается объединить такие команды в пакеты, каждый из которых рассматривается так одна сверхдлинная команда.



54. Характеристики памяти

Месторасположение

Процессорные: Встроенные в процессор (например, регистры).

Внутренние: Память, находящаяся внутри системного блока (например, ОЗУ).

Внешние: Внешние устройства хранения (например, жесткие диски, SSD, магнитные ленты).

Емкость

Емкость памяти измеряется в метрических множителях (килобайты, мегабайты) или в двоичных множителях (кибибайты, мебибайты).

Единица пересылки

Слово: Минимальная единица данных, обрабатываемая процессором.

Строка кэша: Блок данных, используемый в кэш-памяти.

Блок на диске: Минимальная единица данных, хранимая на диске.

Метод доступа

Произвольный (адресный): Доступ к любой ячейке памяти возможен напрямую по её адресу.

Ориентированный на записи (прямой): Доступ осуществляется к блокам данных (например, SSD).

Последовательный: Доступ к данным происходит последовательно (например, магнитные ленты).

Ассоциативный: Доступ к данным по их содержимому (например, кэш-память).

Характеристики памяти

1. Быстродействие и временные соотношения

Время доступа (Тд): Время от момента подачи адреса до момента получения данных.

Длительность цикла памяти (Tц): Время между двумя последовательными операциями доступа к памяти.

Время чтения и время записи: Интервалы времени, необходимые для чтения и записи данных.

Время восстановления (Тв): Время, необходимое для восстановления содержимого ячеек памяти.

Скорость передачи информации: Количество данных, передаваемых за единицу времени.

2. Физический тип и особенности

Различные типы памяти имеют свои физические особенности, такие как энергозависимость, плотность записи и т.д.

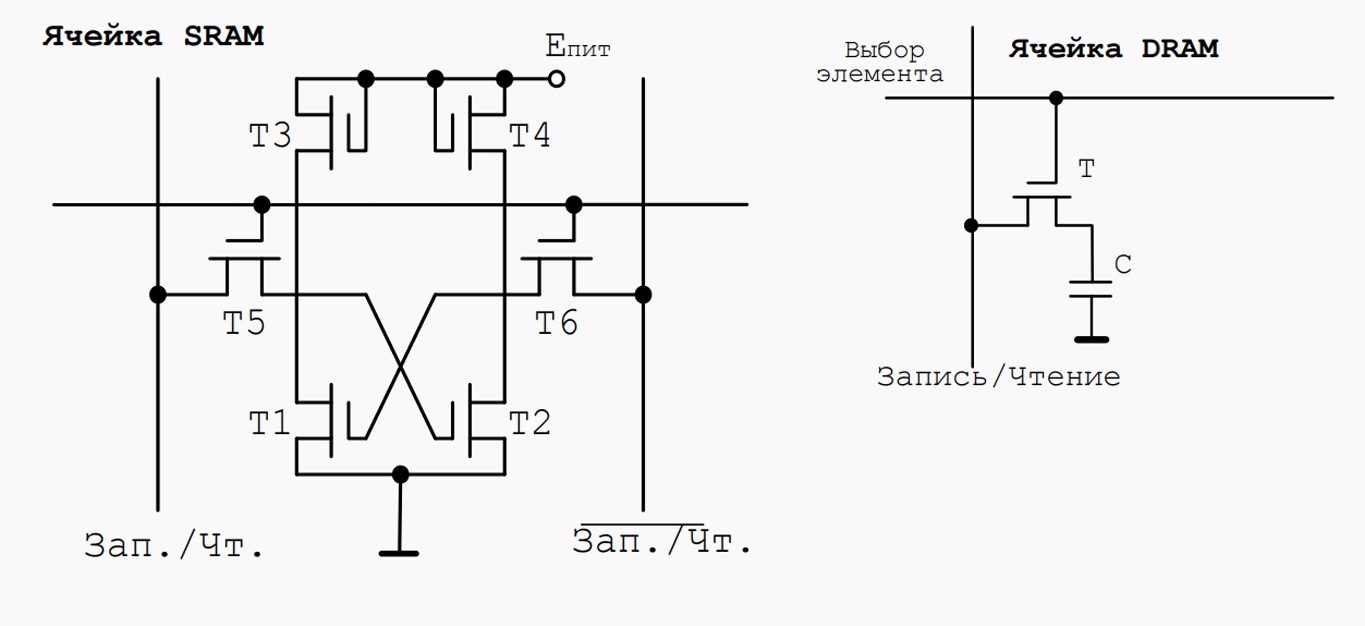
3. Стоимость

Стоимость памяти зависит от её типа, ёмкости и быстродействия. (стоимость байта )

55. Статическая vs Динамическая память

**SRAM** (статическая память с произвольным доступом): Она построена на транзисторах и не требует постоянного обновления данных. Данные в такой памяти хранятся, пока есть питание. Схема слева показывает, что SRAM состоит из шести транзисторов (T1-T6).

**DRAM** (динамическая память с произвольным доступом): Здесь данные хранятся в конденсаторе, который нужно периодически обновлять, чтобы сохранить информацию. Схема справа показывает, что DRAM состоит из одного транзистора и одного конденсатора (Т и С).



**Чтение информации** – процесс извлечения данных из ячейки памяти без изменения её содержимого.

**Шина** – это набор проводов, соединяющих различные части компьютера, позволяя им обмениваться данными. Шины бывают однонаправленными (данные идут в одну сторону) и двунаправленными (данные могут идти в обе стороны).

**Регистр процессора** – это небольшая, но быстрая память внутри процессора, используемая для хранения данных и промежуточных результатов вычислений.

**Двунаправленная шина** – шина, по которой данные могут передаваться в обе стороны, в зависимости от управляющих сигналов.

**Вентильная схема** – электронная схема, управляющая потоком данных. Если управляющий сигнал равен 1, данные проходят, если 0 – блокируются.

**Тактовый генератор** – устройство, которое генерирует регулярные электрические импульсы, синхронизирующие работу всех компонентов компьютера.

**Функциональная логическая схема** – комбинация логических элементов, выполняющих определённые логические операции над входными сигналами.

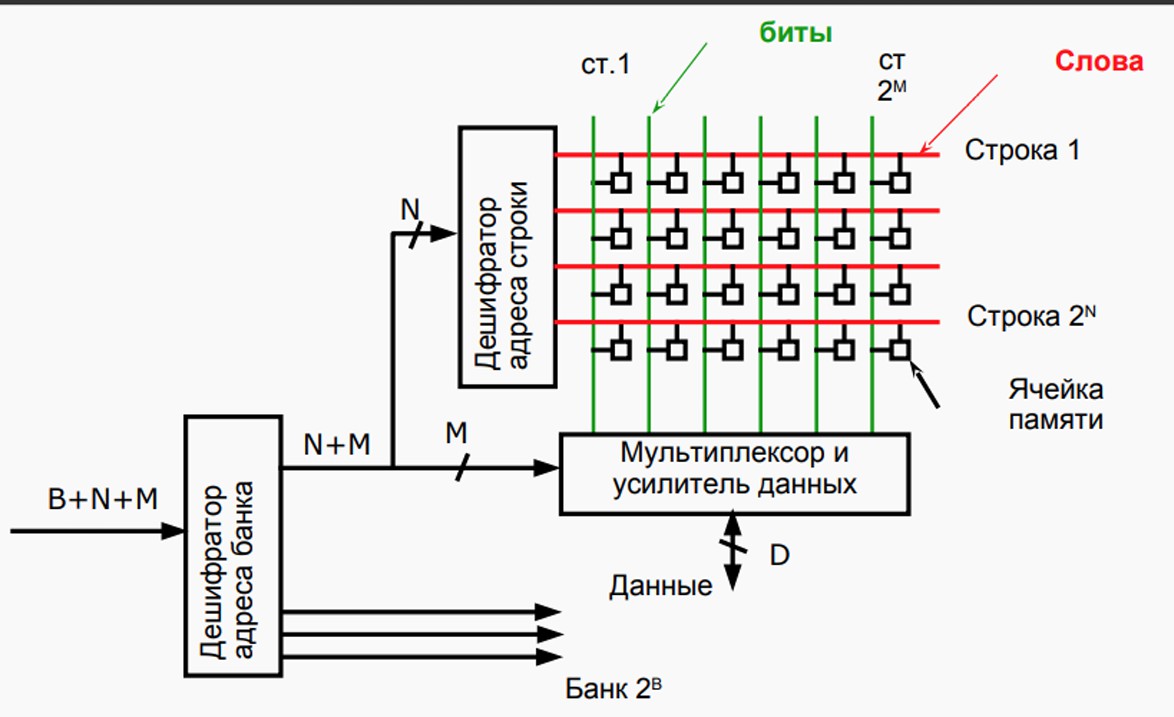
**Триггер** – электронное устройство, которое может находиться в одном из двух устойчивых состояний (0 или 1) и переключаться между ними под воздействием внешнего сигнала. Триггер используется для хранения одного бита информации. Отличительной особенностью триггера как функционального устройства является свойство запоминания двоичной информации. Под памятью триггера подразумевают способность оставаться в одном из двух состояний и после прекращения действия переключающего сигнала.

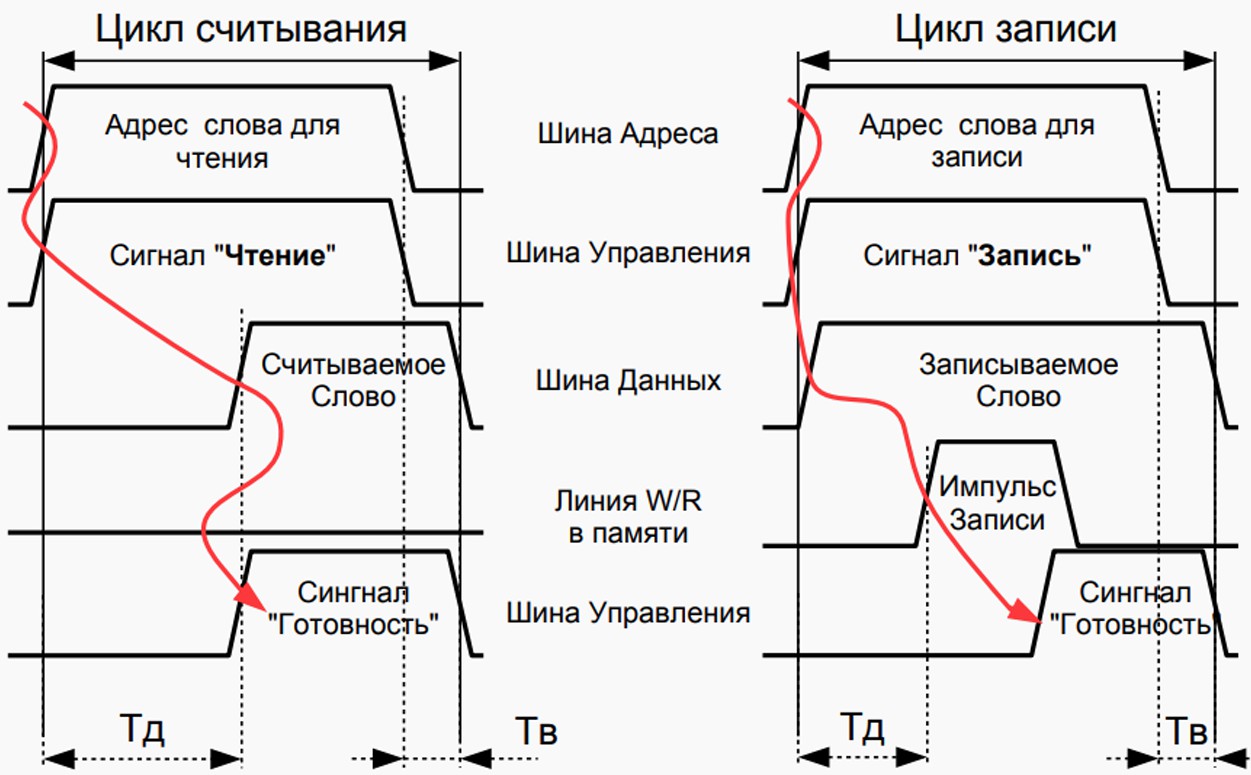
**Счетчик числа импульсов** – устройство, которое считает количество поступивших на него электрических импульсов и отображает их в виде двоичного или десятичного кода.

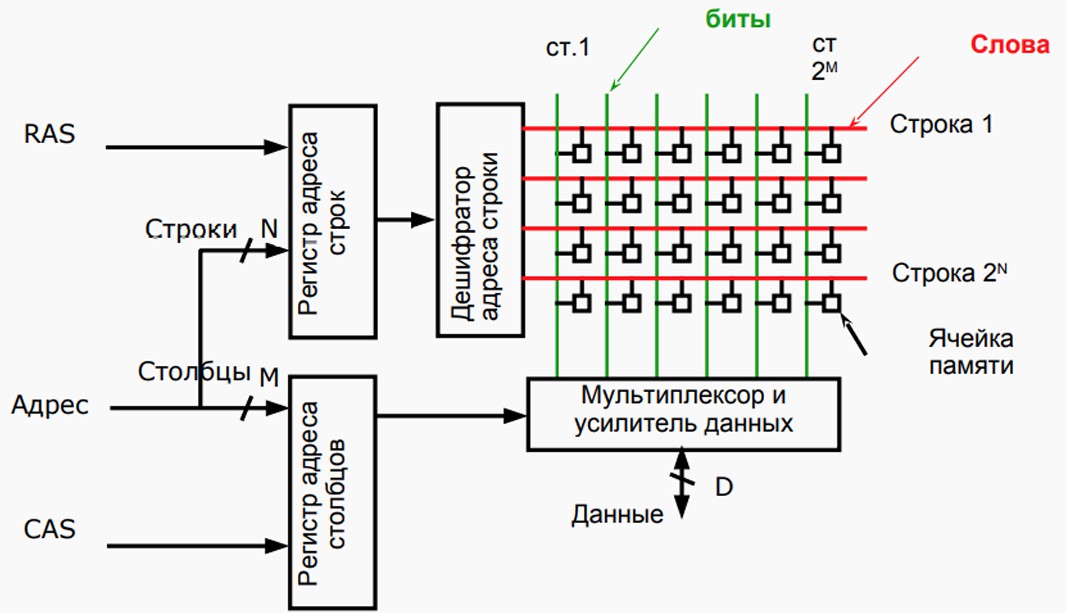
**Сумматор** – устройство, которое складывает два или более входных сигнала и выдает результат в виде одного выходного сигнала.

Эти компоненты и схемы являются основой работы любой вычислительной системы, помогая обрабатывать, хранить и передавать данные внутри компьютера.

56. Адресуемая память







Разбор схем

Первая схема: Адресация и организация памяти

1. Дешифратор адреса банка: Преобразует адрес банка памяти, позволяя процессору выбрать нужный банк для чтения или записи данных.

2. Дешифратор адреса строки: Определяет конкретную строку в выбранном банке памяти.

3. Мультиплексор и усилитель данных: Считывает или записывает данные в ячейки памяти.

Временные диаграммы

1. Цикл считывания:

Адрес слова для чтения: Процессор указывает адрес слова, которое нужно считать.

Сигнал "Чтение": Активируется сигнал для чтения данных.

Считываемое слово: Данные считываются и передаются процессору.

Сигнал "Готовность": Указывает, что данные готовы для использования.

2. Цикл записи:

Адрес слова для записи: Процессор указывает адрес слова, которое нужно записать.

Сигнал "Запись": Активируется сигнал для записи данных.

Записываемое слово: Данные передаются в память для записи.

Импульс записи: Записывает данные в ячейку памяти.

Сигнал "Готовность": Указывает, что запись завершена.

Вторая схема: Дешифрация и усиление данных

1. Регистры адресов строк и столбцов: Хранят адреса строк и столбцов, которые определяют ячейку памяти.

2. Дешифратор адреса строки: Преобразует адрес строки в сигнал для выборки конкретной строки.

3. Мультиплексор и усилитель данных: Управляет считыванием или записью данных в ячейки памяти.

Третья схема: Временные диаграммы DRAM

1. Сигналы RAS и CAS: Определяют строки и столбцы для доступа к данным.

2. Сигнал Output Enable: Активирует выходные данные.

3. Сигнал Чтение/Запись: Определяет режим работы (чтение или запись).

4. Тайминг: Общее время доступа к данным (Тд) составляет 65 нс, что показывает задержку при доступе к данным.

57. Адресуемая память с фиксацией строк и столбцов

58. Синхронная память SDRAM

59. Конструктивные особенности современной памяти

Современная память имеет ряд особенностей, которые улучшают её производительность и функциональность:

1. Burst mode — пакетный режим

Процессор запрашивает данные не по одному байту, а сразу целыми пакетами, состоящими из 32 или 64 бит.

Это позволяет считывать 2 или 4 машинных слова за один такт, что значительно ускоряет работу.

2. Double Data Rate (DDR) — передача данных и по фронту, и по спаду

Передача данных осуществляется как по восходящему, так и по нисходящему фронтам тактового сигнала.

Это удваивает скорость передачи данных без увеличения частоты тактового сигнала.

3. SPD (Serial Presence Detect) — чип с идентификационной информацией

Чип содержит информацию о параметрах модуля памяти, что позволяет автоматически настраивать систему для оптимальной работы с данной памятью.

4. Interleaving — расслоение памяти

Логически связанные байты располагаются в разных микросхемах памяти.

Пока одна микросхема восстанавливается, процессор может работать с другой, что повышает производительность.

5. DDR4-2133 8192MB PC4-17000 — индекс производительности

Обозначение DDR4-2133 указывает на тип памяти и её тактовую частоту (2133 МГц).

8192MB — объём памяти (8 ГБ).

PC4-17000 — показатель производительности (17 000 МБ/с).

Другие особенности

1. Включение динамической памяти в статическую память

Некоторое количество статической памяти (SRAM) включается в микросхемы динамической памяти (DRAM), что позволяет ускорить доступ к часто используемым данным.

2. Синхронная работа памяти и ЦП

Использование внутренней конвейерной архитектуры и чередование адресов позволяет памяти работать синхронно с процессором, повышая общую производительность системы.

3. Чередование памяти (Interleaving Mode)

Данные размещаются в разных микросхемах памяти, чтобы во время регенерации одной микросхемы процессор мог работать с другой.

4. Разбиение памяти на страницы (Paging Mode)

Ячейки с одинаковым адресом строки группируются в страницы. Это экономит время на поиск и считывание данных, так как адрес строки остаётся неизменным.

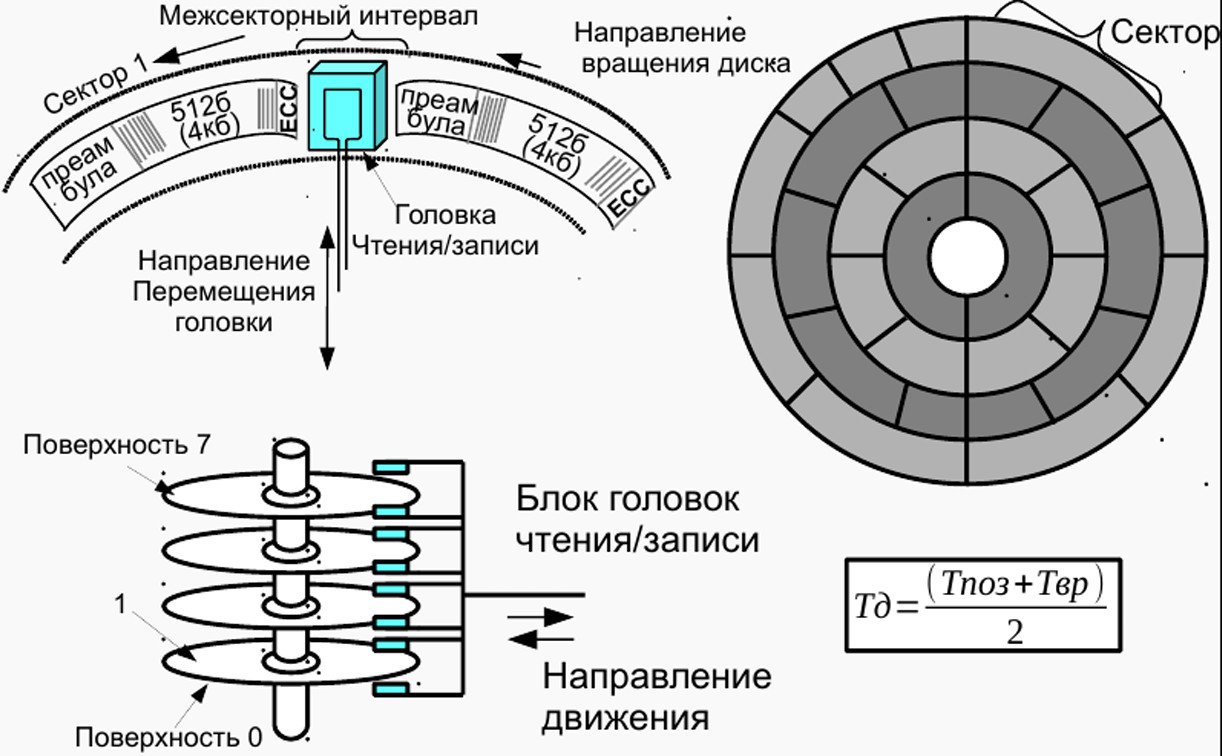
Обычно страницы имеют размер 512 байт и более.

5. Кэширование памяти

Между процессором и оперативной памятью (RAM) размещается кэш-память объёмом до 2 МБ.

Кэш-память работает на частоте процессора, что позволяет избежать циклов ожидания при обращении к часто используемым данным.

60. Память, ориентированная на записи



Блочная память, такая как дисковая память и память на магнитных лентах, организована в виде блоков или секторов, что позволяет эффективно управлять данными и обеспечивать их долговременное хранение.

Организация дисковой памяти

Дисковая память использует магнитные диски для хранения данных. Она состоит из нескольких ключевых элементов и организована особым образом:

1. Диск и его структура:

Секторы: Диск разделён на сектора, каждый из которых содержит определённое количество данных (обычно 512 байт или 4 КБ).

Треки: Сектора располагаются вдоль концентрических окружностей, называемых треками.

Цилиндры: Совокупность треков одинакового радиуса на всех поверхностях диска образует цилиндр.

2. Головка чтения/записи:

Блок головок чтения/записи: Каждый диск имеет отдельные головки для чтения и записи данных, которые перемещаются над поверхностью диска.

Направление движения: Головка перемещается радиально для доступа к различным трекам и секторам.

Межсекторный интервал: Пространство между секторами, позволяющее головке перемещаться между ними.

3. Процесс чтения/записи:

Адресация: Адрес каждого блока данных включает номер цилиндра, номер трека и номер сектора.

Чтение: Процесс чтения данных начинается с позиционирования головки над нужным цилиндром и треком, после чего считывается сектор данных.

Запись: Процесс записи данных аналогичен чтению, но включает изменение магнитного состояния диска для записи новых данных.

4. Среднее время доступа (Tд):

Среднее время доступа к данным на диске определяется как среднее арифметическое между временем позиционирования головки (Тпоз) и временем вращения диска (Твр): Tд= (Тпоз+Твр)2Tд = \frac{(Тпоз + Твр)}{2}Tд=2(Тпоз+Твр)

61. Память, с последовательным доступом\*

62. Структура ассоциативного запоминающего устройства

Процессоры всегда работают быстрее, чем память, поэтому разница в скорости приводит к задержкам в работе. Инженеры знают, как построить быструю память, но она дорого стоит и ограничена в размерах. Для решения этой проблемы используется кэш-память, которая находится ближе к процессору и хранит наиболее часто используемые данные.

Ассоциативная память (АМ)

Ассоциативная память, или память, адресуемая по содержимому, используется для быстрого поиска данных. В ней данные выбираются не по адресу, а по содержимому. Пример ассоциативной памяти — это кэш- память.

Принципы работы ассоциативной памяти:

1. Минимальная адресуемая единица (МАЕ):

В ассоциативной памяти МАЕ называется строкой ассоциативной памяти (СтрАП).

Каждая СтрАП содержит два поля: поле тега (метки) и поле данных.

2. Запрос на чтение:

Запрос выглядит как: "выбрать строку, у которой тег равен заданному значению".

Возможны три результата: одна строка с заданным тегом, несколько строк с заданным тегом, ни одной строки с заданным тегом.

Схема ассоциативной памяти:

Регистр ассоциативного признака: Хранит тег.

Регистр маски: Указывает, какие биты тега важны для поиска.

Схемы совпадения: Проверяют соответствие тегов строк в памяти с запрашиваемым тегом.

Регистр совпадений: Содержит результаты проверки.

63. Кэш память

Кэш-память используется для уменьшения времени доступа к данным. Она хранит копии часто используемых данных из основной памяти.

Структура кэш-памяти:

Кэш-контроллер: Управляет работой кэш-памяти, загружает данные из оперативной памяти и возвращает модифицированные данные обратно.

Процессор: Запрашивает данные у кэш-контроллера.

Оперативная память: Хранит все данные, к которым обращается процессор.

Принципы работы кэш-памяти:

1. Кэш-попадание (cache hit):

Процессор запрашивает данные, которые находятся в кэш- памяти.

Данные извлекаются быстро.

2. Кэш-промах (cache miss):

Запрашиваемые данные отсутствуют в кэш-памяти.

Кэш-контроллер запрашивает данные из оперативной памяти.

64. Пирамида памяти



Пирамида памяти показывает различные уровни памяти в компьютере, их объём и скорость доступа:

1. CPU:

Объём: 100-1000 байт

Время доступа (Тд): <1 нс Тип: Регистры

Управление: Компилятор

2. L1 Cache:

Объём: 32-128 КБ

Время доступа (Тд): 1-4 нс Тип: Ассоциативная память Управление: Аппаратура

3. L2-L3 Cache:

Объём: 0.5-32 МБ

Время доступа (Тд): 8-20 нс Тип: Ассоциативная память Управление: Аппаратура

4. Основная память (ОЗУ): Объём: 0.5 ГБ - 4 ТБ

Время доступа (Тд): 60-200 нс Тип: Адресная память

Управление: Программное

5. SSD:

Объём: 128 ГБ - 1 ТБ на диск

Время доступа (Тд): 25-250 мкс Тип: Блочная память

Управление: Программное

6. Жесткие диски:

Объём: 0.5 ТБ - 4 ТБ на диск Время доступа (Тд): 5-20 мс Тип: Блочная память

Управление: Программное

7. Магнитные ленты:

Объём: 1-6 ТБ на кассету

Время доступа (Тд): 1-240 секунд Тип: Последовательная память

Управление: Программное

Детали

Время доступа: Временной интервал от подачи адреса до получения данных.

Время записи: Интервал времени для записи данных в память.

65. Влияние промахов кэш-памяти

Влияние кэш-промахов на производительность:

Чем чаще кэш-промахи, тем ниже производительность.

График зависимости производительности от попаданий в кэш:

При 100% попаданий в кэш производительность максимальна.

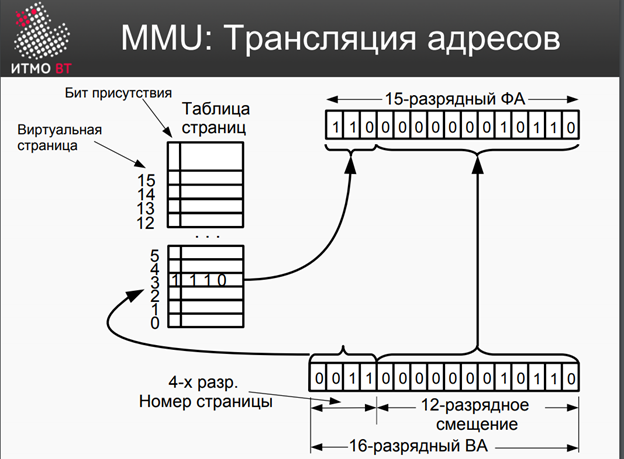
Снижение попаданий в кэш приводит к значительному падению производительности.

66. Сегментно-страничная виртуальная память



Каждый процесс в современной вычислительной машине разделен на сегменты и каждый сегмент находится в адресуемой памяти на своем месте, причем важно обратить внимание на то, что в виртуальном адресном пространстве все адреса одинаковые, что видно на слайде. Важно то, что внутри процессов адреса одинаковые, но они ссылаются на разные физические адреса памяти

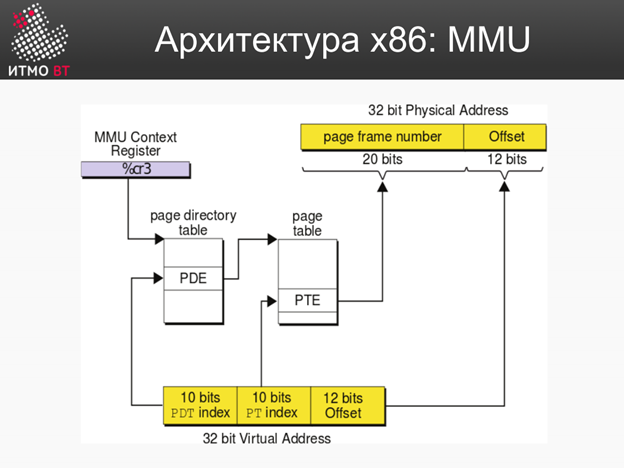
67. MMU и TLB

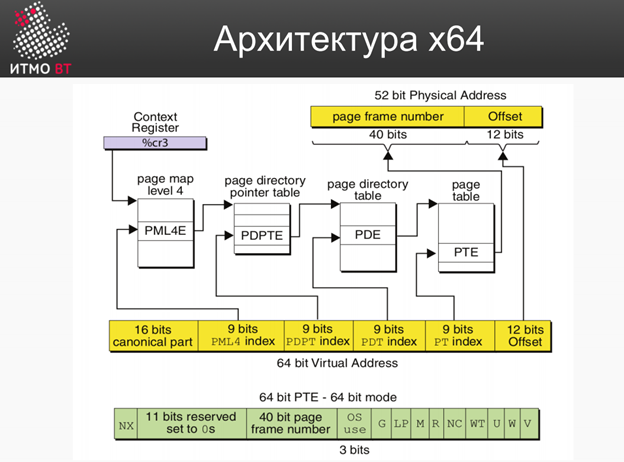


MMU – Memory Management Unit

У него на вход поступает виртуальный адрес, который с помощью таблицы преобразование проверяет наличие страницы в виртуальной памяти и какой у нее физический адрес

Этим занимает операционная система





MMU отвечает за преобразование виртуальных адресов в физические:

1. Процесс преобразования:

Процессор подаёт виртуальный адрес на вход MMU.

MMU трансформирует виртуальный адрес в физический, заменяя номер виртуальной страницы на номер физической.

2. Порядок работы:

Если MMU выключено или адрес попал в нетранслируемую область, физический адрес равен виртуальному.

Если MMU включено и адрес в транслируемой области, происходит трансляция адреса:

Если нужный номер страницы есть в TLB, используется он.

Если нет, идёт поиск в таблицах страниц ОС, после чего запись добавляется в TLB.

TLB ускоряет процесс преобразования виртуальных адресов:

1. Функции:

Кэширует часто используемые преобразования адресов.

Разделён для адресов и данных.

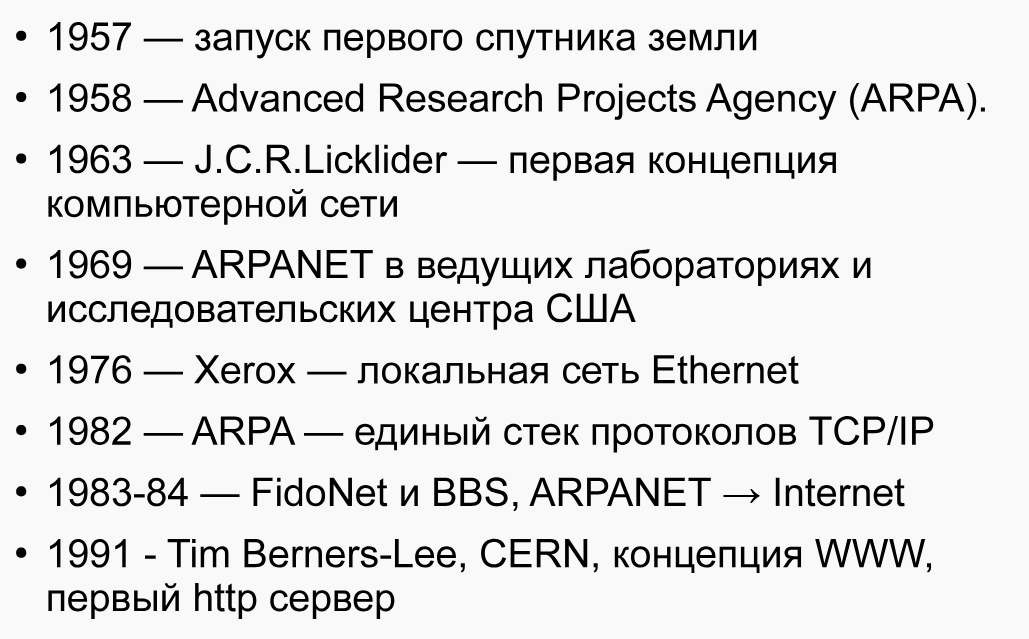
2. Организация:

Организован как ассоциативная память.

Записываем туда виртуальные адреса и соответствующие им физические, тем самым избегаем частого обращения к памяти

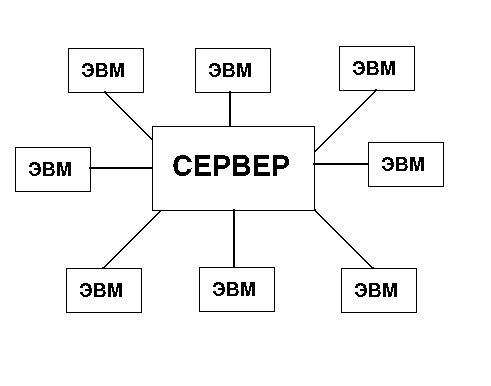
Когда виртуальный адрес необходимо преобразовать в физический адрес, то вначале просматривается TLB. Если совпадение найдено (называется «TLB hit») то возвращается физический адрес, и доступ к памяти может продолжаться. Однако, если нет совпадения (называемого «TLB miss») то либо модуль управления памятью либо обработчик пропусков TLB операционной системы ищет сопоставление адресов в таблице страниц, чтобы узнать, существует ли сопоставление (называется «page walk»). Если сопоставление существует — то оно записывается обратно в TLB (это необходимо сделать, поскольку аппаратное обеспечение обращается к памяти через TLB в системе виртуальной памяти), и текущая команда перезапускается (что также может происходить параллельно).

68. История сети Internet



69. Понятие сети ЭВМ

Под сетью ЭВМ понимают соединение двух и более ЭВМ с целью совместного использования их ресурсов (процессоров, устройств памяти, устройств ввода/вывода, данных). По степени охвата территории различают сети:

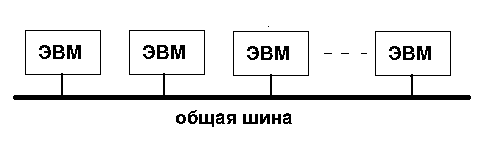
* локальные (местные) - в пределах одного учреждения, помещения (или при максимальном удалении ЭВМ не более 1км.)
* региональные - внутри населенного пункта, района
* национальные - внутри государства
* глобальные

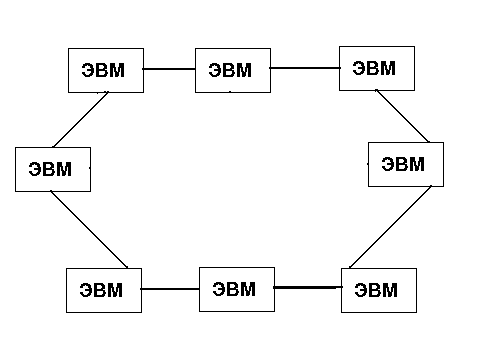
По степени доступности различают корпоративные и общедоступные сети.

По топологии:

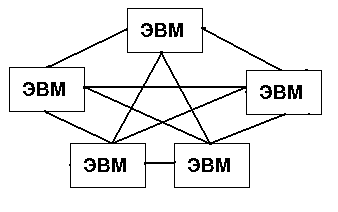
* звездообразная топология

При таком способе обмен данными между ЭВМ осуществляется через более мощную ЭВМ - сервер. Недостатком такого соединения является низкая живучесть сети - выход из строя сервера означает прекращение функционирование сети. Однако, простота и дешевизна реализации сделала эту структуру популярной в локальных сетях.

* топология с общей шиной

При этом способе обмен данными происходит через общую шину, которую используя механизм прерывания может "захватывать" тот или иной компьютер. Характерной особенностью здесь является отсутствие сервера. Очень часто используется в локальных сетях, а уж в "домашних" повсеместно.

* кольцевая топология

В этой структуре каждая ЭВМ используя механизм пм рерывания работает в качестве ретранслятора. Обратите внимание, живучесть сети повышена - при одиночном обрыве связи между соседними ЭВМ сеть продолжает функционировать

* полная топология

Соединение ЭВМ "каждая с каждой" позволяют получить сеть самую дорогую, но и обладающую максимальной живучестью.

### Примеры сетей ЭВМ

1. **Локальная сеть (LAN)**:
   * Лан — это сеть, которая охватывает небольшую географическую область, такую как офис, здание или кампус. LAN позволяет пользователям в этой области делиться ресурсами и обмениваться данными.

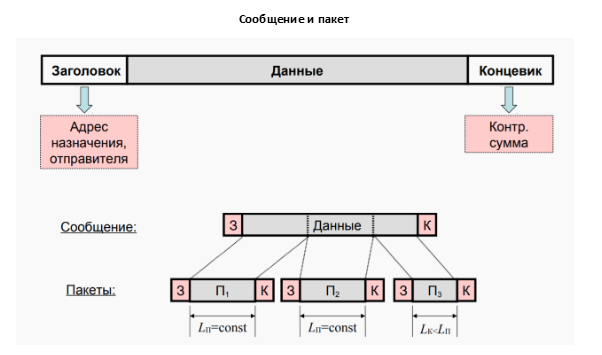
70. Сообщение, пакет

Сообщение – цифровые данные определенного формата, предназначенные для передачи.

Пакет — это определенным образом оформленный блок данных, передаваемый по сети. Часто состоит из заголовка и полезной нагрузки

передаче сообщений создается логический канал, по которому информация передается от одного узла к другому.

Пакет — это часть сообщения, передаваемая по сети. Сообщения разбиваются на пакеты, каждый из которых содержит заголовок с адресной информацией и номер пакета, необходимый для сборки сообщения на узле назначения.



71. Модель взаимодействия открытых систем (OSI)

OSI (Open Systems Interconnection) — это концептуальная модель, разработанная Международной организацией по стандартизации (ISO) для стандартизации и описания функций сетевого взаимодействия. Модель OSI описывает, как данные передаются по сети, и разделяет процесс передачи данных на семь уровней, каждый из которых выполняет определенные функции. Эти уровни помогают структурировать и стандартизировать сетевые протоколы и взаимодействие между различными сетевыми устройствами.

Модель OSI состоит из 7 уровней, каждый из которых выполняет определенные функции:

1. Физический уровень: Определяет физические средства передачи данных, такие как электрические сигналы или световые импульсы.

2. Канальный уровень: Обеспечивает адресацию в пределах локальной сети и обнаружение ошибок. Используются MAC-адреса и протокол Ethernet.

3. Сетевой уровень: Отвечает за маршрутизацию данных между различными сегментами сети. Используются протоколы IPv4 и IPv6.

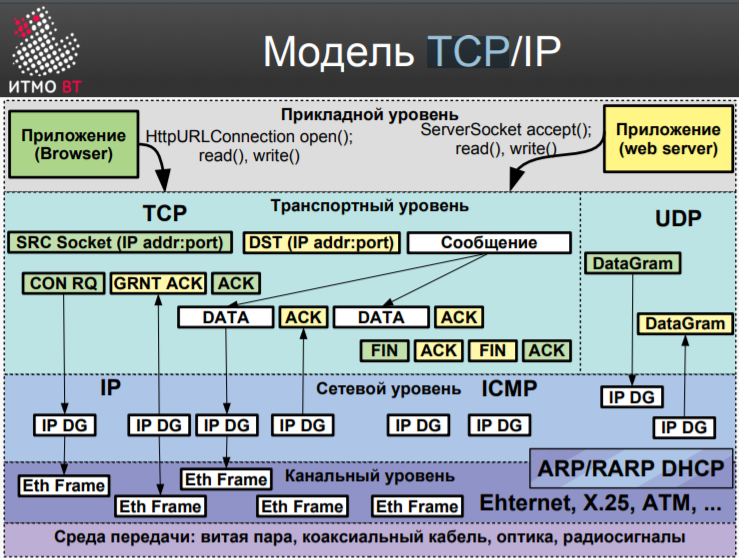
4. Транспортный уровень: Обеспечивает передачу данных между узлами сети. Протоколы TCP и UDP работают на этом уровне.

5. Сеансовый уровень: Управляет установлением, поддержанием и завершением сеансов связи.

6. Уровень представления: Преобразует данные в формат, пригодный для прикладного уровня, такие как кодировки ASCII или форматы изображений JPEG.

7. Прикладной уровень: Предоставляет интерфейс для приложений, таких как браузеры и почтовые клиенты, работающие по протоколу HTTP, FTP и др.

72. Модель TCP/IP



**TCP/IP** — сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю. В модели предполагается прохождение информации через четыре уровня, каждый из которых описывается правилом (протоколом передачи). Наборы правил, решающих задачу по передаче данных, составляют стек протоколов передачи данных, на которых базируется Интернет. Название TCP/IP происходит из двух важнейших протоколов семейства — Transmission Control Protocol (TCP) и Internet Protocol (IP), которые были первыми разработаны и описаны в данном стандарте.

73. Уровень передающей среды

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Уровень (layer) | | Тип данных (PDU) | Функции | Примеры |
| Host  layers | 7. Прикладной | Данные | Доступ к сетевым службам | HTTP, FTP, POP3, WebSocket |
| 6. Представления | Представление и шифрование данных | ASCII, EBCDIC |
| 5. Сеансовый | Управление сеансом связи | RPC, PAP, L2TP |
| 4. Транспортный | Сегменты/Дейтаграммы | Прямая связь между конечными пунктами и надёжность | TCP, UDP, SCTP, PORTS |
| Media  layers | 3. Сетевой | Пакеты | Определение маршрута и логическая адресация | IPv4, IPv6, IPsec, AppleTalk |
| 2. Канальный | Биты/Кадры (frame) | Физическая адресация | PPP, IEEE 802.22, Ethernet, DSL, ARP, сетевая карта. |
| 1. Физический | Биты | Работа со средой передачи, сигналами и двоичными данными | кабель (USB, «витая пара», коаксиальный, оптоволоконный), радиоканал |

74. Канальный уровень Ethernet

+ данные отправит и получат и исправ ошибки

Канальный уровень обеспечивает перенос данных по физической среде, обмен происходит кадрами. Уровень работает с применяемым в Ethernet физическими адресами, которые «вшиты» в сетевые адаптеры их производителями. Пример канального уровня: проверка доступности канала связи, если он общий для нескольких абонентов(Wireless).

* Управление доступом к среде (физическому уровню) передачи,
* Физическая адресация узлов (MAC)
* Обеспечение сервиса для протоколов более высокого уровня (Service Access Point)
* Упорядочивание кадров (фреймов), буферизация
* Учет топологии сети
* Управление потоком данных
* Определение способа взаимодействия источника и приемника (связь с установлением соединения или нет)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 байт | 1 байт | 6 байт | 6 байт | 2 байта | 46 – 1500 байт | 4 байта |
| Преамбула (101010…) | SFD (10101011) | Адрес получателя | Адрес источника | Тип | данные | Контрольная сумма |

**Преамбула** (Preamble) не несет полезной информации. Генерируемый при её передаче в физической среде сигнал извещает принимающие устройства о необходимости быть готовым к приему.

**SFD** (Start of Frame Delimiter) разделитель начала кадра тоже не несет полезной информации (в первых версиях разделитель считался частью преамбулы). Он позволяет приемнику точно определить момент начала передачи полезных данных.

Эти два элемента кадра явным образом предназначены для обеспечения правильной работы нижнего – физического уровня.

**Адрес получателя** DA (Destination Address) содержит уникальный MAC – адрес устройства, которому адресован данный кадр или специальный адрес для широковещательной рассылки пакетов.

MAC-адреса

Этот адрес всегда состоит из 6 байт или 48 бит. Для его записи обычно используется шестнадцатеричная форма ХХ:ХХ:ХХ:ХХ:ХХ:ХХ.

Значения первых двух бит в первом байте - признак уникального адреса, остальные байты задают адрес конкретного сетевого адаптера. Уникальность адресации адаптеров обеспечивается специ­альным соглашением, по которому каждому производителю аппаратуры выделяется свое значение (одно или несколько) кода OUI (Organizationally Unique Identifier – уникальный идентификатор организации) — 22 бита из байтов 1-3. Байты 4-6 заполняются изготовителем — на нем лежит ответственность за их уникальность (эта информация может рассматриваться как серийный номер платы).

75. Сетевой уровень IP

В отличии от канального уровня, имеющего дело с физическими адресами, сетевой уровень работает с логическими адресами. Он предназначается для определения пути передачи данных. Отвечает за трансляцию логических адресов и имён в физические, определение кратчайших маршрутов, коммутацию и маршрутизацию, отслеживание неполадок и заторов в сети. Одна из основных функций сетевого уровня – маршрутизация, объединение нескольких разнородных локальных сетей в одну сеть.

К протоколам сетевого уровня относится IP и ICMP (Internet Control Massage Protocol).

**IP-адрес** — уникальный сетевой адрес узла в компьютерной сети, построенной на основе стека протоколов TCP/IP

На сетевом уровне сам термин *сеть* наделяют специфическим значением. В дан­ном случае под сетью понимается совокупность компьютеров, соединенных между собой в соответствии с одной из стандартных типовых топологий и использующих для передачи данных один из протоколов канального уровня, определенный для этой топологии.

Внутри сети доставка данных обеспечивается соответствующим канальным уров­нем, а вот доставкой данных между сетями занимается сетевой уровень, который и поддерживает возможность правильного выбора маршрута передачи сообщения даже в том случае, когда структура связей между составляющими сетями имеет характер, отличный от принятого в протоколах канального уровня.

Сети соединяются между собой специальными устройствами, называемыми маршрутизаторами. *Маршрутизатор —* это устройство, которое собирает инфор­мацию о топологии межсетевых соединений и на ее основании пересылает пакеты сетевого уровня в сеть назначения. Чтобы передать сообщение от отправителя, находящегося в одной сети, получателю, находящемуся в другой сети, нужно со­вершить некоторое количество *транзитных передач между сетями, waaxonoe* (от *hop —* прыжок), каждый раз выбирая подходящий маршрут. Таким образом, марш­рут представляет собой последовательность маршрутизаторов, через которые про­ходит пакет.

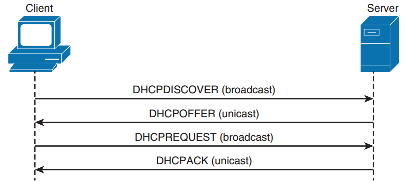
76. Сетевой уровень IP: маршрутизация

**Маршрутизация пакетов** проводится с помощью IP адресов. Маршрутизация осуществляется в посылающих TCP/IP конечных устройствах и в маршрутезаторах. В обеих случаях на интернет-уровне конечные устройства и в маршрутезаторы длжны принимать решение куда отсылать пакет. Для принятия такого решения интернет-уровень получает информацию из специальных таблиц маршрутеризации. Записи в таблице создаются автоматически при инициализации этого TCP/IP, хотя это может быть сделано и вручную. При доставке IP пакетов возможны два варианта доставки:

1. **Прямая доставка** (Direct delivery), в случае которой одно IP конечное устройство посылает пакеты другому устройству, принадлежащему одному и тому же сектору, используя MAC адрес принимающего устройства.

2. **Косвенная доставка** (Indirect delivery) производится через промежуточные устойства или маршрутезаторы к цели, которая не относится к тому же сектору локальной сети. В этом случае посылающий компьютер адресует пакет данных, используя MAC адрес маршрутизатора.

77. DHCP

TCP/IP сеть позволяет пользовательским ЭВМ при подключении к сети в автоматическом режиме получить IP-адрес, маску, шлюз и стандартный DNS-сервер сети посредством DHCP сервера. Согласно протоколу DHCP, клиент отправляет широковещательный(его получат все узлы сети) запрос DHCPDISCOVER, чтобы найти DHCP сервер. Сервер, в ответ на данный запрос, отвечает предложением(DHCPOFFER) выдать адрес для клиента. Если клиент хочет воспользоваться этим DHCP сервером, то в ответ на DHCPOFFER он посылает DHCPREQUEST, который тоже является широковещательным, чтобы уведомить все остальные DHCP сервера, что их услуги не требуются. После получения DHCPREQUEST, сервер резервирует за клиентом IP адрес и подтверждает право его использования пакетом DHCPACK.

DHCP (Dynamic Host Configuration Protocol) — это протокол, который автоматически назначает IP-адреса устройствам в сети. Это позволяет избежать ручной настройки IP-адресов, особенно в больших сетях, где это может быть трудозатратно и подвержено ошибкам.

Принцип работы DHCP

Получение IP-адреса через DHCP происходит в четыре шага, известные как процесс DORA:

1. Discovery (Поиск): Клиент, который еще не имеет IP-адреса, отправляет широковещательное сообщение DHCPDISCOVER, чтобы найти DHCP-серверы в сети.

2. Offer (Предложение): DHCP-серверы отвечают на запрос сообщением DHCPOFFER, предлагая IP-адрес из своего пула адресов.

3. Request (Запрос): Клиент принимает предложение и отправляет DHCPREQUEST серверу, подтверждая, что он хочет использовать предложенный IP-адрес.

4. Acknowledgement (Подтверждение): Сервер подтверждает запрос сообщением DHCPACK, завершая процесс назначения IP-адреса и указывая срок его действия.

78. Cервис имен DNS и другие

Для удобства пользования сетью модель TCP/IP предоставляет возможность создавать текстовые алиасы, называющиеся доменными именами, для хостов посредством сервиса DNS(Domain name system). Клиент, когда ему требуется получить IP-адрес, соответствующий имеющемуся доменному имени, отправляет запрос к DNS-серверу, и если у сервера есть информация о соответствии, то сервер отвечает на запрос и дает клиенту IP-адрес хоста. Если у сервера нет информации о соответствии, то он обращается к вышестоящему DNS-серверу и, кроме того, чтобы отправить полученную информацию клиенту, кэширует её и в следующий раз даст клиенту информацию не обращаясь к вышестоящим серверам. Существуют DNS-сервера самого высокого уровня, которые называются корневыми.

DNS (Domain Name System) — это система, которая преобразует удобные для человека доменные имена (например, www.example.com) в IP-адреса (например, 192.0.2.1), используемые для маршрутизации трафика в сети.

Принцип работы DNS

1. Запрос имени: Когда пользователь вводит доменное имя в браузере, запрос отправляется на локальный DNS-сервер.

2. Поиск: Если локальный DNS-сервер не знает IP-адрес, он пересылает запрос к корневым DNS-серверам.

3. Иерархия: Корневые серверы направляют запрос к серверу домена верхнего уровня (например, для .com), который затем направляет запрос к серверу конкретного домена (example.com).

4. Ответ: Когда DNS-сервер находит соответствие имени и IP-адреса, он отправляет ответ обратно пользователю.

DNS также поддерживает обратное сопоставление, когда по IP-адресу можно получить доменное имя. Для этого используется домен in-addr.arpa.

79. Транспорт уровень

Транспортный уровень модели OSI (равно как и стека протоколов TCP/IP) обеспечивает следующие возможности:

* Отслеживание индивидуальных сеансов общения между приложениями на передающем и принимающем устройствах.  
  Эта функция относится только к протоколу TCP. Перед началом передачи полезных данных узел, инициирующий соединение открывает сессию с узлом получателем, чтобы убедиться, что получатель существует и готов принимать данные. Далее все полезные данные передаются в рамках установленной сессии и после завершения такой передачи сессия закрывается.
* Сегментация данных (разбиение больших порций данных на сегменты для индивидуальной отсылки по сети, сборка этих сегментов после получения)
* Идентификация приложений, передающих и принимающих данные.  
  Для идентификации приложений используются номера портов: каждое приложение, желающее работать с сетью сообщает операционной системе о своих планах и регистрирует за собой какой-то номер порта. Впоследствии, когда на компьютер придут данные, транспортный уровень заглянет в поле «номер порта получателя» и передаст эти данные соответствующему приложению.

80. Прикладной уровень

Прикладной уровень (application layer) — это в действительности просто набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам, таким как файлы, принтеры или гипертекстовые веб-страницы, а также организуют свою совместную работу, например, по протоколу электронной почты. Единица данных, которой оперирует прикладной уровень, обычно называется сообщением.

Прикладной уровень включает протоколы, используемые приложениями для связи по сети. Примеры таких протоколов:

HTTP (Hypertext Transfer Protocol): Используется для передачи веб- страниц.

FTP (File Transfer Protocol): Используется для передачи файлов.

SMTP (Simple Mail Transfer Protocol): Используется для передачи электронной почты.

DNS (Domain Name System): Используется для преобразования доменных имен в IP-адреса.

DHCP (Dynamic Host Configuration Protocol): Используется для автоматического назначения IP-адресов.