1) Операционные системы

Оператор в компьютерных системах первых поколений занимался тем, что принимал от программистов программу и исходные данные, вводил и запускал её на исполнение. При этом дисплей и клавиатура появились у вычислительных машин не сразу, вместо них использовался пульт управления вычислительной машиной. С развитием вычислительной техники появилась естественная идея заменить оператора специальной программой, которая будет заменять его работу. Такие программы получили название операционных систем.

Операционная система загружает программы в память, контролирует их исполнение, предоставляет результаты работы в понятном для человека виде, обеспечивает возможность одновременного исполнения нескольких программ. В современном мире операционные системы можно условно разделить на пользовательские — где основной целью является предоставление конечному пользователю удобства работы с вычислительной системой, серверные, предназначенные для обработки большого количества запросов от других систем, и встроенные — целью которых является поддержка функционирования аппаратного обеспечения вычислительных систем, датчиков, преобразователей сигнала и другого оборудования.

Современные операционные системы: Windows, Unix, Linux, Android, IOS, DOS. Отдельно следует упомянуть гипервизоры операционные системы, которые предназначены для управления другими операционными системами.

Операционная система содержит ядро и окружение (библиотеки, файлы) пользовательских программ. В ядро обычно включаются подсистемы, позволяющие управлять и эффективно разграничивать данные и программы пользователя для исключения их неразрешенного взаимодействия между собой, драйверы аппаратного обеспечения, подсистемы обобщенного ввода и вывода информации.

2) История UNIX/Linux

Название Unix произошло от акронима UNICS (UNiplex Information and Computing System). Разработчики Кен Томпсон, Денис Риччи и Руд Канадей создали однопользовательскую (позднее двухпользовательскую), написанную на ассемблере операционную систему для популярной в то время вычислительной системы PDP-7. Позднее имя было заменено на UNIX. Позднее система была переписана на язык C и портирована на PDP-11.

Рождение UNIX (1969–1979):

Начало с Unnamed PDP-7 OS, перешедшей в UNIX версии 1–7.

BSD (Berkeley Software Distribution) появляется как первая ветка UNIX (версии 1.0–4.1).

Развитие BSD и коммерческих UNIX-систем (1980-е):

BSD становится основой для будущих открытых UNIX-систем.

Параллельно развиваются коммерческие закрытые варианты UNIX (Solaris, Sun OS, AIX).

Открытые системы и переход к Linux (1990-е):

BSD-форки, такие как FreeBSD, NetBSD, OpenBSD, продолжают развитие в открытой среде.

Появляется Linux (1991), создавая новую открытую альтернативу UNIX, которая станет доминирующей в будущем.

Консолидация коммерческих UNIX-систем (1990-е–2000-е):

Solaris (Sun Microsystems) становится важной коммерческой платформой.

UNIXWare и OpenServer продолжают развиваться, но постепенно теряют позиции.

Доминирование открытых систем (2000-е):

Linux активно развивается и доминирует в серверной среде.

Mac OS X (на базе Darwin) объединяет элементы BSD и NeXTSTEP.

3) Современность

Базовой версией коммерческих операционных систем UNIX послужило ядро ОС UNIX System V. Изначально разработанное для поддержки масштабируемых вычислительных систем оно представляет собой ядро с уникальными для всего семейства UNIX функциями программной поддержки горячей замены аппаратных компонентов (включая память и процессоры) без перезагрузки ядра, высокой масштабируемостью (одновременно могут работать от 1 до сотен процессоров, поддерживаются терабайты ОЗУ), развитой архитектурой, предоставляемой пользовательским процессам. Наиболее известными, используемыми на предприятиях, в настоящее время являются Solaris 2.x+ (Sun/Oracle) и AIX (IBM).

BSD и ее производные часто используются для организации сетевой инфраструктуры, поскольку поддержка большого количества протоколов и средств, позволяющих управлять сетевым взаимодействием, в них широко развита. Особняком стоит MacOS прямой наследник ОС NextStep - предоставляющая удобную оболочку для пользователя и развитый пользовательский интерфейс.

Linux представлен богатым выбором операционных систем. Отличия в версиях, которые тем не менее базируются на одном программном ядре, состоит в управлении программным обеспечением (пакетные менеджеры) и способах распространения и поддержки ОС.

4) Ядро *NIX

Каждая операционная система выполняет служебные операции в ядре. Ядро представляет собой набор подсистем, которые управляют оборудованием и программами пользователя. Основными подсистемами, которые жизненно важны для любой ОС являются управление памятью, управление процессами и потоками (отвечает за выполнение нескольких программ одновременно на одной ОС), виртуальная файловая система (обмен данными организован с помощью файлов, с точки зрения ОС в UNIX все является файлами), сетевая подсистема, драйвера устройств и платформозависимый код (драйвера, которые поддерживают функционирования системы на различных аппаратных архитектурах). Взаимодействие программ пользователя с ядром происходит с помощью интерфейса системных вызовов.

С точки зрения пользователя любая исполняемая программа представляет собой адресное пространство, которое разбито логически и физически на несколько сегментов. Например, сегмент code, где выполняется собственно сама программа, сегмент data, где находятся необходимые для ее работы данные, сегмент stack, для хранения адресов возврата и передачи аргументов к подпрограммам. Кроме этих основных сегментов еще используются heap или куча, сегменты разделяемых библиотек, memory-maped файлы, т. е. файлы непосредственно отображенные в адресное пространство процесса и другие. Все эти сегменты вместе составляют образ процесса, который работает внутри операционной системы и взаимодействует с ней. Кроме этого, программы могут взаимодействовать между собой, используя ядро и интерфейс системных вызовов (и только через него).

5) Файловая система
ФС представлена в виде дерева. Верхнем элементом иерархии файловой системы является директория root (корень иерархии). Логические разделы дисковых накопителей могут подключаться в файловую систему в точках монтирования созданных администраторами директорий, в которых при подключении их файловая иерархия заменяется на расположенную на разделе диска. Каждая директория содержит два обязательных файла. Файл «.» - это ссылка на саму себя и файл «..» - это ссылка на директорию выше по файловой иерархии. В файловой иерархии есть каталоги, которые использует сама система. К ним, например, относится каталоги /usr/bin, где расположены основные утилиты и программы, поставляемые вместе с ОС.
В файловой системе можно адресоваться при помощи абсолютных и относительных путей. Абсолютный путь — это путь от корня ФС. Относительный путь – это путь от текущей директории, откуда была запущена программа пользователя.
В файловой системе есть специальные виды файлов, которые могут помочь организовать необходимую для пользователя или программ иерархию файлов ссылки на другие файлы. Ссылки бывают двух типов: жесткие и символические ссылки. В ОС UNIX каждому файлу внутри файловой системы для каждого накопителя присваивается свой уникальный номер inode. Директория это просто файл, где указаны имена файлов и их inode. Если два имени файла имеют один и тот же inode (внутри одной точки монтирования) значит это один и тот же файл с таким же содержимым. Или можно сказать, что один файл является жесткой ссылкой на другой. Файл существует в файловой системе до тех пор, пока не удалена последняя жесткая ссылка. Символическая ссылка – файл содержащий абсолютный или относительный путь до файла.

6) Права доступа к файлам
Вывод команды ls -l:
1) Тип файла и права доступа к нему
2) Кол-во жестких ссылок на файл
3) Имя владельца
4) Имя группы владельца
5) Длина файла в байтах
6) Время. По умолчанию показывается mtime, в зависимости от ключей может быть отображено atime (ls -u) или ctime (ls -c).
Права доступа разбиты на несколько групп и определяют права для разных категорий пользователей по отношению к текущему владельцу файла и группе владельца:
| Тип файла | Права для владельца | Права для группы | Права для остальных |
Все типы прав содержат три значения: право на чтение/право на запись/право на исполнение. Право на чтение (r) директории позволяет пользователю просматривать список файлов и поддиректорий, находящихся внутри этой директории. Право на запись (w) в директорию даёт возможность добавлять, удалять и переименовывать файлы в этой директории. Право на исполнение (x) директории даёт возможность "входить" в директорию, т. е. открывать её и перемещаться по её содержимому. Оно не связано с запуском программ, как для файлов, но означает, что пользователь может получить доступ к самой директории и выполнять действия над её содержимым, зная его точные имена.
Символ + (. - нет расширенных прав) в выводе команды ls -l указывает на то, что для файла или директории заданы расширенные списки контроля доступа (ACL, Access Control List). ACL позволяют задавать более детальные права на уровне пользователя и группы, которые выходят за рамки стандартной системы прав.

7) Способы задания прав
Права (для владельца, группы и остальных) принято обозначать в виде трех восьмеричных цифр. Каждая цифра отвечает свою тройку и высчитывается так rwx = $111_2 = 7_8$; r-x = $101_2 = 3_8$; - -x = $001_2 = 1_8$. Таким образом, рассчитывается десятичная цифра, соответствующая определенным правам для каждой тройки типов прав. Команда chmod 644 <имя файла> установит права rw - r- - r- - -.
Еще один способ изменять права это добавлять их или убирать с помощью буквенных обозначений. Сначала указывается, какие права будут изменены. u — права пользователя, g - права группы, o – права для остальных. Далее можно использовать знаки +, - или =.

8) Потоки stdin(0), stdout(1), stderr(2)
Файловый дескриптор — это целое число, которое операционная система использует для идентификации открытого файла или ресурса. Оно выступает как ссылка на структуру данных, содержащую информацию о текущем состоянии взаимодействия с файлом. Дескрипторы уникальны только внутри одного процесса. В каждом процессе в Unix автоматически открываются три стандартных файловых дескриптора (номера зарезервированы):

0 — Стандартный ввод (stdin)
1 — Стандартный вывод (stdout)
2 — Стандартный вывод ошибок (stderr).

9) Интерпретатор команд
Всеми командами управляет интерпретатор команд - shell. Эта программная оболочка позволяет организовывать выполнение команд, задавать им переменные окружения, связывать потоки ввода-вывода у различных команд между собой, а также анализировать коды возврата у команд и использовать простейший язык для написания новых исполняемых команд - скриптов.
sh, bash и ksh — это разные типы оболочек shell, которые предоставляют интерфейс для взаимодействия с операционной системой через командную строку. Каждый из них имеет свои особенности, синтаксис, функции и возможности.
sh - это одна из первых оболочек, созданных в Unix в конце 1970-х. sh имеет простой синтаксис, базовый набор команд и операций управления потоком (циклы, условия и т.д.). Она довольно ограничена по функционалу по сравнению с современными оболочками, но все еще используется для написания скриптов из-за высокой совместимости с другими Unix-оболочками.
bash — это расширенная версия sh. Она является стандартной оболочкой в большинстве современных дистрибутивов Linux. bash добавляет множество новых возможностей, таких как автодополнение команд, история команд, поддержка работы с массивами и функций, расширенные конструкции для работы с циклами и условиями. bash обеспечивает обратную совместимость с sh, так что большинство скриптов, написанных для sh, работают в bash без изменений.
ksh (KornShell) направлен на работу с бОльшим количеством пользователей.

10) Перенаправление потоков stdin(0), stdout(1), stderr(2)
$>$ (перенаправление стандартного вывода в файл): используется для записи стандартного вывода команды в файл (если файл уже существует, его содержимое будет перезаписано).
$>>$ (добавление стандартного вывода в файл): используется для добавления (аппендирования) стандартного вывода команды в файл (если файл уже существует, содержимое сохраняется, а новые данные добавляются в конец файла).
$<$ (перенаправление стандартного ввода из файла): используется для передачи содержимого файла как стандартного ввода в команду.
$<<$ (here-document, встроенный ввод): позволяет передать текст напрямую в команду через стандартный ввод, используя специальный маркер конца ввода EOF (в начале и в конце передаваемого текста на отдельной строке пишем EOF).
$<<<$ - аналогично $<<$, но работает только для одной строки (не требует EOF).
&>file или >&file - направить стандартный поток вывода и стандартный поток ошибок в файл. Другая форма записи: >file 2>&1.
Конвейер (|) — это механизм, позволяющий передавать вывод одной команды на вход другой. С помощью конвейера можно комбинировать несколько команд, чтобы выполнять обработку данных поэтапно.

11) Фильтры
Это команды (или программы), которые воспринимают входной поток данных, производят над ним некоторые преобразования и выдают результат на стандартный вывод (откуда его можно перенаправить куда-то еще по желанию пользователя). Фильтр-команды: cat, grep, sort, tail, head, wc, sed.

12) Регулярные выражения
Регулярное выражение определяет шаблон, которому должна соответствовать строка в тексте, в котором проводится поиск на совпадение. Простейший случай регулярного выражения - это последовательность символов, которые соответствуют сами себе.
. — любой одиночный символ.
* — ноль или более вхождений предыдущего символа.
^ — начало строки.
$ — конец строки.
[] — диапазон символов. Например, [a-z] — любой символ от a до z.
Экранирование — это способ использования специальных символов так, чтобы они воспринимались как обычные символы и не имели особого значения в строках или командах, выполняется с помощью символа обратного слэша (\).

13) Команды
mkdir - создание директории
echo - вывод символов на стандартный вывод
cat - конкатенация и вывод файлов на стандартный вывод
touch - создание (если файл отсутствует) или изменение mtime и atime файла
ls - вывод списка файлов в директории
pwd - вывод текущей директории в стандартный вывод
cd - смена текущей директории на указанную
more - постраничная распечатка файла или стандартного ввода
cp — копирование файлов и директорий
rm - удаление файлов

rmdir — удаление пустых директорий
mv - перемещение файлов
head - вывод указанного количества строк с начала файла или stdin
tail - вывод указанного количества строк с конца файлов или stdin
sort - сортировка строк файла или стандартного ввода
grep - выборка строк по шаблону
wc - подсчет количества символов, строк, слов в файле или стандартном вводе.

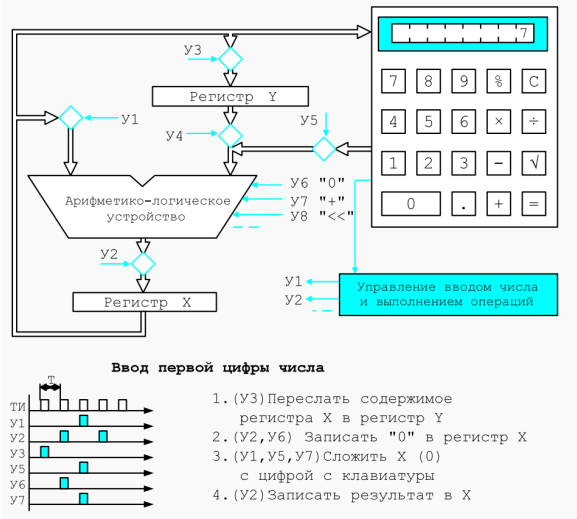
14) Аналоговые ЭВМ
С развитием техники и технологий людьми были сконструированы приборы, которые позволяли определять различные физические величины, такие как скорость, ускорение, расстояния. Например, если мы хотим измерить высоту полета самолета, то мы можем измерить давление атмосферы или послать электромагнитный импульс к земле и посчитать, когда придет отраженный сигнал. Но сами физические значения, такие как высота, сложно использовать в дальнейших вычислениях в их естественном виде, и люди придумали заменять их аналогичными величинам или просто аналогами. В современной вычислительной технике, в качестве аналоговых величин используются значения напряжения. При этом высоте, например, может ставится в соответствие напряжение от датчика высоты. Аналоговая величина всегда непрерывна, и ее поведение полностью копирует поведение соответствующей физической величины. Теперь с этой аналоговой величиной можно проводить вычисления - складывать, вычитать, дифференцировать. Например, при изменении высоты, можно рассчитать вертикальную скорость относительно поверхности. Аналоговые вычисления производится аналоговыми ЭВМ. К сожалению, у таких ЭВМ есть недостатки.
Основной из них - большие габариты таких вычислительных машин. Кроме того, не всегда просто преобразовать физическую величину в аналог и обратно. Аналоговые системы до сих пор широко используются в системах реального времени, в различных измерительных устройствах, в информационно-управляющих системах, потому что у них очень высокое быстродействие для определенного класса задач и удовлетворительная погрешность вычислений. Это происходит за счет того, что аналоговая величина непрерывна.

15) Цифровые ЭВМ
Вопрос о недостаточной точности вычисления аналоговых ЭВМ привел к созданию цифровых ЭВМ. Решили, что вместо сложного устройства, которое точно различает величину амплитуды напряжения в 0,001 вольта, проще использовать сигнал, у которого будет всего два состояния - либо 0, либо 1. Это существенно упростило конструкцию вычислительных машин, приведя к созданию цифровых вычислительных систем.
Отдельные импульсы в цифровых вычислительных машинах объединяются в группы, позволяя закодировать информацию кодовой посылкой в виде одного или нескольких чисел. Цифровое представление существенно упрощает необходимые элементы для построения вычислительной машины.

16) Функциональные элементы ЭВМ
На самом нижнем логическом уровне компьютеры состоят из логических элементов, которые выполняют логические функции. У каждого логического элемента есть один или несколько входов и один выход. На вход подаются значения (0 или 1), внутри элемента выполняется функция, результат которой попадает на выход. Простейшая микросхема объединяет несколько логических элементов и обычно содержит входы и выходы этих элементов, а также входы для подачи напряжения питания.
Для хранения информации и данных в ЭВМ используется оперативная память, информация в которой сохраняется, пока ЭВМ включена. В современных вычислительных системах существует два типа оперативной памяти: динамическая память (DRAM) и статическая память (SRAM).
В DRAM информация хранится на емкости, которая находится в цепи зарядов транзистора. Ячейка памяти состоит из одного транзистора и одного конденсатора. Недостаток такой памяти: емкость постепенно разряжается, соответственно информация теряется. Для того, что бы поддерживать заряд на емкости предусмотрено устройство регенерации памяти: компьютер раз в определенное время (64 мс) сканирует всю имеющуюся оперативную память и считывает каждую ячейку, возобновляя заряд на конденсаторе. Достоинством DRAM является ее дешевизна на единицу хранимой информации, т. к. для одного бита используется всего один транзистор.
Статическая память SRAM не требует регенерации за счет такой конструкции ячейки, где шесть транзисторов объединены в схему, которая самоподдерживает свое состояние за счет обратных цепей связи. Однако цена такой памяти будет больше, кроме того, ячейки данного типа являются более быстродействующими. На их основе строится память, которая обычно встроена в процессор. К такой памяти относятся, например, кэш память различных уровней.
С точки зрения схемы ЭВМ, построенной на логических элементах, в качестве хранения информации используются триггеры. Триггер = «защелка», с помощью которой можно хранить и управлять информацией.
Например, D-триггер представляет собой ячейку информации, хранящую один бит информации. У триггера есть 2 входа. Один из них D - информационный, на него подается значение, которое необходимо сохранить, и C синхронизации, который определяет, будет ли значение на информационном входе записано в триггер. Триггеры объединяются в регистр (регистр просто совокупность единичных триггеров с общим управляющим входом C). Имеется общий вход C, который для всех триггеров один. Когда на него подается 1, одновременно вся информация поместится в ячейки памяти. Таким образом, регистры образуют промежуточное хранение информации.
Передачу информации обычно нужно осуществлять не непрерывно, а в строго заданное (тактовым генератором) время. Для этого предусмотрено логическое устройство, называемое вентилем, которое функционирует по принципу хорошо известного нам крана. Вентиль может быть в открытом состоянии, и тогда информация с выхода одного ре-

гистра поступает на другой, или в закрытом, тогда передача информации блокирована.
У вентиля есть управляющий и информационный сигналы. На его выходе единица будет только в том случае, если информационный и управляющий сигналы будут равны единице (эквивалент логического И).
Вычислительная машина не только передает и хранит данные. Она должна еще выполнять с ними различные арифметические или логические операции. Например, одни из простейших операций сложение и вычитание, осуществляются с помощью сумматора (входит в АЛУ). В общем случае, для сложения 2х разрядов многоразрядного двоичного числа у логического элемента должны присутствовать 3 входа два для значений разрядов слагаемых и один, дополнительный, для входящего переноса из предыдущего разряда, и два выхода Для значения суммы и выходящего переноса. Перенос передается следующему разряду после суммирования предыдущих (один из выходов побитового сумматора подключен ко входу следующего сумматора).
Тактовые генераторы являются «сердцем» ЭВМ (в ЭВМ он один обычно, иначе будет сдвиг и ничего не будет работать). Подобно человеческому сердцу они задают ритм работы вычислительной машины в целом. Все пересылки данных, арифметические или логические операции могут происходить только в строго заданное время, определяемое размером такта генератора (время такта постоянно). Одни операции могут выполняться по фронту сигнала ТГ, другие по значению, некоторые по спаду сигнала. К концу такта все операции должны быть завершены.

17) Первая ЭВМ: Калькулятор (1)



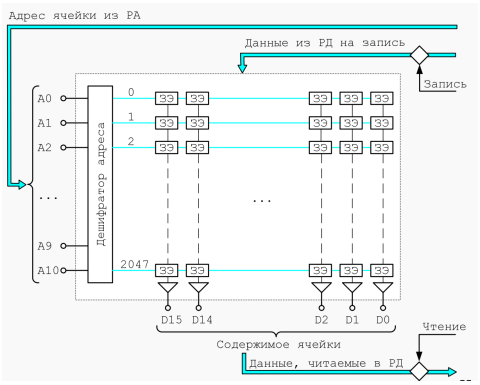
ЭВМ калькулятора состоит из двух регистров X и Y, хранящих результаты ввода пользователя и промежуточных вычислений, АЛУ, которая может выполнять простейшие арифметические (считает в двоично-десятичной системе) и логические операции, шин и управляющих вентилей, осуществляющих передачу данных между функциональными 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.
Пусть 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.
Во-первых необходимо сохранить предыдущее значение регистра X в регистре У. Для этого должен быть открыт вентиль, управляющий записью в регистр Y. Он открывается управляющим сигналом У3. Далее необходимо обнулить регистр X, 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 X 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 X (У2), отобразив его на дисплее.

18) Архитектура ЭВМ, Гарвардская, Фон Неймана
Основным отличием между архитектурами является способ работы с памятью. В настоящее время в чистом виде архитектуры встречаются редко, обычно используется их комбинация в разных функциональных блоках ЭВМ.
В Гарвардской архитектуре центральным устройством является Control Unit - управляющее устройство в ЭВМ. Все остальные устройства ЭВМ подключены к управляющему устройству и взаимодействуют через него. Память для команд (instruction memory), данных (data memory) и устройств ввода-вывода физически отделена друг от друга.
Архитектура фон Неймана содержит все в общей памяти. Нельзя определить, что находится в ячейке, данные или инструкции, без дополнительного анализа кода самой программы. При этом устройство обращения к памяти едино для данных и инструкций, что упрощает конструкцию ЭВМ. Отдельно выделены устройства ввода-вывода, которые являются внешними по отношению к процессору - объединению памяти, АЛУ и управляющего устройства.

19) Структура БЭВМ-NG
Память состоит из 2048 ячеек. Каждая ячейка занимает 16 разрядов. Для обращения к памяти существует два регистра: 11-разрядный регистр адреса (AR - Address Register), в который нужно поместить адрес прежде чем обратиться к памяти; 16-разрядный регистр данных (DR - Data Register), который предназначен для чтения или записи данных в/из ячеек памяти. Чтение данных и запись данных реализуется по шинам, которые подключаются к ячейке памяти.
11-разрядный счетчик команд (IP - Instruction Pointer). Хранит в себе адрес следующей исполняемой команды.
Арифметико-логическое устройство или АЛУ (ALU - Arithmetic-n-Logic Unit) может выполнять несколько операций: сложение, логическое умножение, инверсия и прибавление единицы. При операций «сложение» возможен выход за пределы разрядной сетки и формирование битов переполнения и переноса. Выход из АЛУ через коммутатор подключается к шине, по которой информация может быть передана в любой другой регистр БЭВМ.
Буферный регистр (BR - Buffer Register) это 16-разрядный регистр, который используется для организации промежуточного хранения данных во время работы.
Регистр команд (CR Command Register) используется для хранения кода команды и декодирования операций, происходящих во время работы.
Аккумулятор (AC - Accumulator). БЭВМ относится к ЭВМ, которые называются ЭВМ аккумуляторного типа, где все вычисления с данными производятся через этот регистр.
Указатель стека (SP - Stack Pointer), как и IP и AR 11-ти разрядный, и всегда указывает на вершину стека (тип памяти, который используется для промежуточного хранения данных).
16-разрядный клавишный регистр (IR Input Register) находится в составе пульта оператора ЭВМ и предназначен для ввода адреса программы, кодов программы и данных, запуска программы на выполнение и управления режимами работы БЭВМ.
16-ти разрядный регистр состояния (PS - Program State) хранит биты управляющие работой БЭВМ (работа, прерывание и пр.) и признаки результата.

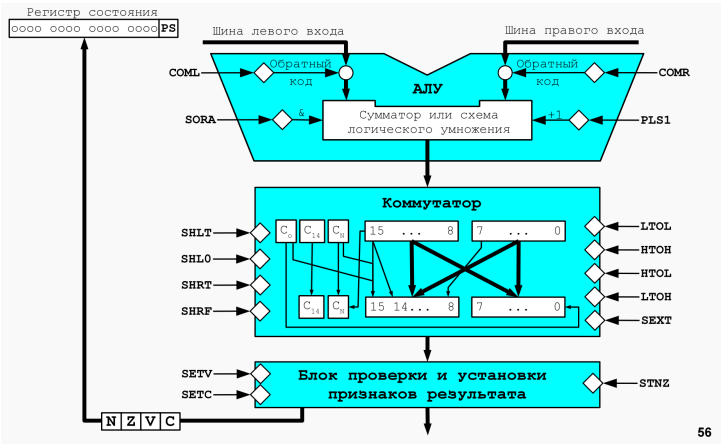
20) Устройство Управления
Устройство управления разработано в виде микропрограммного устройства управления (МПУ, Microprogram Control Unit) - простейшего компьютера, программа которого непосредственно состоит из микроопераций - т. е. по-тактного изменения значений вентилей БЭВМ, которые задают атомарные операции: вычисления в АЛУ, пересылки данных между регистрами и простейшие проверки. Код программы для МПУ называется микрокодом. МПУ выполняет все машинные команды БЭВМ. Исполнение в МПУ машинной команды называется циклом команды. Цикл команды логически разбит на пять циклов:
Цикл выборки команды. Осуществляет загрузку исполняемой команды в регистр команды и частичное ее декодирование. Выполнятся для каждой исполняемой команды.
Цикл выборки адреса. Предназначен для обработки адресных команд и выборки адреса операнда с учетом режимов адресации.
Цикл выборки операнда. Для тех команд, где это необходимо, размещает в DR второй операнд команды.
Цикл исполнения. Производится исполнение команды.
Цикл прерывания. Цикл выполняется в том случае, если разрешены прерывания.
Для обеспечения работы оператора БЭВМ в ней предусмотрена микропрограммная реализация циклов пультовых операций:
Ввод адреса - адрес из клавишного регистра вводится в счетчик команд.
Чтение — информация по адресу в IP читается из памяти в DR, IP увеличивается на единицу.
Запись - информация из IR записывается в память по адресу в IP, IP увеличивается на единицу. Используется для ввода программы и данных в режиме оператора.
Пуск - осуществляет сброс состояния БЭВМ (сброс всех регистров кроме IP) и переход к выполнению программы.
Продолж - выполнение программы покомандно.
На панели оператора, кроме того, расположены другие органы управления: переключатель «Работа/Останов», который вызывает останов программы после каждой команды; переключатель «Такт», который может выполнить микрокод по одному такту, кнопка «Продолжение» возобновляющая работу остановленной БЭВМ.

21) Адресуемая память БЭВМ



Дешифратор — это устройство, на ввод которого подается код числа, а на выходе выбирается только одна выходная линия, номер которой соответствует коду на входе дешифратора.
Память Базовой ЭВМ является адресной (RAM) памятью. Адресуемая память выбирает одну из ячеек, которая соответствует коду адреса, и операция с памятью (чтение или запись) производится с этой выбранной ячейкой. Т.е. для того, чтобы прочитать или записать ячейку памяти необходимо знать адрес ячейки.
По верхней шине адреса поступает адрес из регистра адреса и попадает на дешифратор адреса. На выходе дешифратора активируется одна из линий, и на этой линии находится несколько запоминающих элементов (D-триггеры), количество элементов соответствует разрядности памяти. После этого, приходит сигнал о необходимой операции, активируется та часть схемы, которая отвечает за запись данных в память или чтения из памяти в регистр данных.

22) АЛУ, коммутатор, блок признаков результата



АЛУ имеет два входа левый и правый. На каждом входе АЛУ расположен инвертор схема побитной инверсии поступающих на вход сигналов. Эти схемы позволяют получить обратный код двоичного числа поступающих на заданный вход АЛУ. В обратном коде происходит побитовая замена единиц на нули и наоборот.
Отдельный вентиль предназначен для выполнения в АЛУ операции увеличение на 1 или инкремента. Это сделано путем подачи единичного сигнала на вход входного переноса сумматора (+1 на слайде). Соответственно, инкремент может только осуществляться во время операции сложения в АЛУ.
Основная часть АЛУ состоит из схемы сумматора или логического умножения. Выбор операции производится при помощи вентиля SORA (Sum OR And). Если на его входе 1, то будет выполняться поразрядное логическое умножение левого и правого входа АЛУ, в противоположном операндов с учетом входного переноса и формирование выходного переноса. Схемы сумматора или логического умножения непосредственно подключены к коммутатору.
В Коммутатор поступают 18 разрядов с АЛУ (16-ть разрядов результат операции плюс сформированный биты нового переноса и переноса из 14 разряда сумматора в 15-й), а также предыдущее значение переноса из регистра состояния. В коммутаторе осуществляются прямая передача данных, обмен байтов слова, расширение знака младшего байта в старший, а также арифметические и логические сдвиги. Выход коммутатора поступает на блок формирования однобитовых признаков результата, которые характеризуют результат операции, проведенной в АЛУ и коммутаторе:
Бит признака отрицательного числа (N - Negative)
Признак того, что буферный регистр содержит 0 (Z — Zero)
Бит переполнения V знаковых чисел (V — oVerflow)
Бит переноса C беззнаковых чисел (Carry)

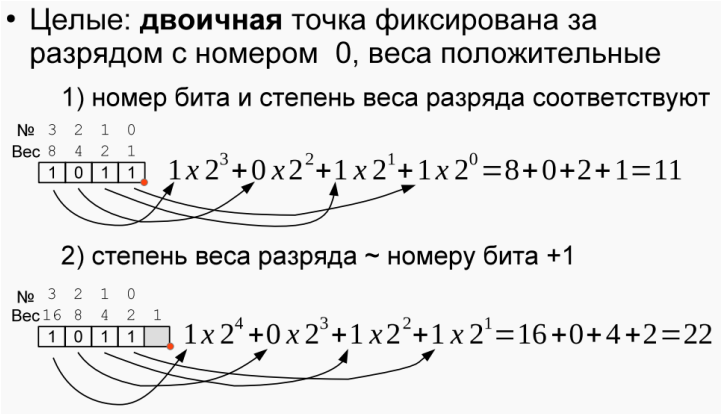
23) Форматы команд
Выбор одного из четырех типов команд осуществляется МПУ при помощи анализа четырех старших бит кода команды (КОП).
Безадресные команды выполняют различные действия без ссылок на ячейку памяти (КОП = 0000).
Команды ввода-вывода управляют обменом данными между процессором и внешними устройствами ЭВМ (КОП = 0001).
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 0x2 до OxE и задает операцию.
Команды ветвления позволяют продолжить вычислительный процесс с другого адреса программы в зависимости от состояния признаков результата NZVC (КОП = 1111).
Список всех команд есть в методе на странице 21.

24) Адресные команды
С прямой абсолютной адресацией в бите 11 у этих команд всегда 0, а в адресной части (битах с 0 по 10) записано значение адрес операнда в памяти. При выполнении операции команда непосредственно обращается по данному адресу выбирая или записывая операнд.
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 IP.

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. Режимы адресации могут быть:
прямая относительная (0xE) (прямая со смещением относительно IP);
косвенная относительная (0x8): подразумевает, что в ячейке памяти, которая вычисляется из адресной части команды через сложение смещения со счетчиком команд, хранится адрес операнда;
косвенная автоинкрементная (0xA): аналогична случаю косвенной адресации, однако после загрузки операнда из памяти, значение адреса в ячейке памяти увеличивается на 1;
косвенная автодекрементная (0xB): аналогична случаю косвенной адресации, однако перед загрузкой операнда из памяти, значение адреса в ячейке памяти уменьшается на 1;
со смещением относительно SP (0xC): адрес операнда получается сложением закодированного в команде смещения с указателем стека;
с непосредственной (прямой) загрузкой операнда в аккумулятор (0xF): адресная команда использует число в битах 0-7 команды в качестве операнда.

25) Безадресные команды
Команды работы со стеком используют специальный 11-ти разрядный регистр БЭВМ SP (Stack Pointer). Этот регистр всегда указывает на вершину стека. При пультовой операции пуск данный регистр обнуляется вместе со всеми остальными регистрами. Когда будет выполнена первая команда PUSH, вызвана подпрограмма или произойдет прерывание регистр сперва декрементируется (если там был 0 то станет 7FF, потому, что подчеркнем еще раз, регистр SP 11-ти разрядный), а затем по адресу, записанному в этом регистре запишется значение, которое зависит от операции.

26) Представление точек: фиксированная точка



Особенностью представления чисел с фиксированной/плавающей точкой является то, что они ограничены разрядной сеткой ЭВМ. Рассмотрим ситуацию, когда вес разряда соответствует номеру разряда с положительным смещением. Это позволяет нам кодировать числа большего размера в ту же самую разрядную сетку, но при этом шаг между допустимыми числами будет больше. Во втором примере на слайде двоичная фиксированная точка находится в позиции, соответствующей номеру разряда + 1. В данном случае не возможно представить нечетные числа. Двоичную точку можно переместить еще дальше, в зависимости от задачи, при этом буду получаться числа, которые кратны 4-м, 8-ми и т.д. Алгоритм перевода двоичного числа в десятичное такой же, что и в первом примере.

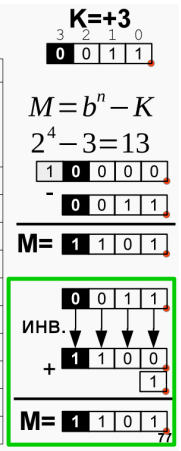
27) Представление беззнаковых целых чисел
На примере четырехразрядной сетки видно, что минимальным беззнаковым числом может быть 0, а максимальным 15. В БЭВМ минимальное значение беззнакового числа составляет 0, а максимальное - 65535.

28) Представление знаковых целых чисел
Прямое кодирование: знак можно хранить, записав его в отдельный, в самый старший бит. Знак кодируется 0, если число положительное и 1 если отрицательное. Для такого способа кодирования значение нуля представлено двумя кодами положительным и отрицательным (двойной нуль). Прямой код в вычислительных устройствах напрямую не используется, т.к. необходима дополнительная логика при определении знаков и больших по модулю чисел при выполнении операций. Конструкция АЛУ при использовании прямого кода существенно усложняется.
Для выполнения вычислений обычно используется дополнительный код числа, который представляет отрицательные числа в виде дополнения до максимально возможного положительного числа + 1 в заданной разрядной сетке. Дополнительный код можно использовать для любой позиционной системы исчисления.
Например, мы хотим представить -1 в десятичной системе счисления, ограниченной 5-ю разрядам. Применяя формулу, получим M (число в дополнительном коде, соответствующее -1) = $10^5 + (-1) = 99999$. Теперь наше АЛУ может выполнять корректные арифметические действия, не прибегая к сравнению знаков и модулей чисел. Пусть нам нужно к -1 прибавить 1, должен получиться 0. Соответственно, 99999 + 1 = старший разряд выходит за заданные пять разрядов и отбрасывается 99999 + 1 = 0.

29) Представление знаковых целых чисел: дополнительный код

$$M = b^n - K = ((b^n - 1) - K) + 1$$

Прямой код 5-ти разр. дес. чисел	Дополнительный код		
	5-ти разр. дес. чисел	4-х разр. шестн. чисел	16-ти разрядных двоичных чисел
-50000	50000		
-49999	50001		
-32768	**67232**	**8000**	**1000 0000 0000 0000**
-32767	67233	8001	1000 0000 0000 0001
-2	99998	FFFE	1111 1111 1111 1110
-1	99999	FFFF	1111 1111 1111 1111
0	**00000**	**0000**	**0000 0000 0000 0000**
1	00001	0001	0000 0000 0000 0001
32767	32767	7FFF	0111 1111 1111 1111
49999	49999		



Смысл использования модифицированной формулы состоит в упрощении перевода числа в отрицательное. $b^n - 1$ представляет собой максимальное положительное беззнаковое число в выбранной разрядной сетке. В двоичной системе получение поразрядного дополнения совпадает с простой инверсией битов числа. То есть, часть формулы = ((b^n – 1) – K) просто заменяется инверсией разрядов числа K. В результате отрицательное число получается в 2 действия: первое инверсия числа, второе прибавление 1 к инверсии.

30) Перенос. Переполнение

В беззнаковых числах, если прибавить 1 к 15 или вычесть из 0 единицу произойдет потеря значения числа, которая сопровождается возникновением переноса. Этот перенос в конструкции ЭВМ учитывается в бите C (Carry), который служит сигналом программисту, что произошла ситуация потеря значения, и ее необходимо обработать отдельно.

В знаковых числах точка возникновения ошибки расположена между представлением чисел -8 и 7. При вычитании из -8 единицы или прибавления единицы к 7 возникает переполнение (OVerflow), которое в ЭВМ контролируется битом переполнения. Напомним, что данный бит обозначается в БЭВМ - V, а в разных современных архитектурах процессоров O, V или OV. АЛУ определяет переполнение по следующему правилу: если поразрядные переносы в знаковом и старшем разряде одновременно отсутствуют или присутствуют - значит переполнения нет, если присутствует только В одном - значит переполнение знаковой разрядной сетки есть.

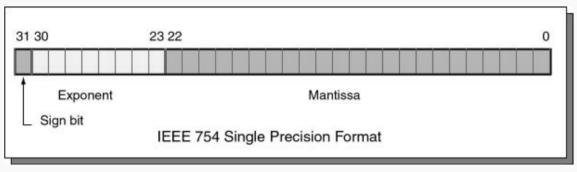
31) БЭВМ: представление чисел

Представление в разрядной сетке	Беззнаковые числа	Знаковые числа
0000 0000 0000 0000	0	0
0000 0000 0000 0001	1	1
...		
0111 1111 1111 1110	32766	32766
0111 1111 1111 1111	32767	32767
1000 0000 0000 0000	32768	-32768
1000 0000 0000 0001	32769	-32767
1111 1111 1111 1110	65534	-2
1111 1111 1111 1111	65535	-1

ОДЗ: $0 \le X \le 2^{16} - 1$ $-2^{15} \le X \le 2^{15} - 1$

32) Представление чисел с плавающей точкой

В БЭВМ — НЕТ!

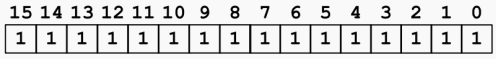


$$X = (-1)^{(sign)} x (1 + Mantissa) \, x \, 2^{(exponent - 127)}$$

Мантисса — это дробная часть числа. Она всегда внутри представлена нормализованной, т.е. подразумевается, что она всегда начинается с единицы. Так как мантисса представлена в виде двоичного числа, то перед тем, как число упаковать, его сдвигают так, чтобы всегда число было нормализовано. Для нашего примера это будет 0.100101. Так как незначащий ноль всегда присутствует в представлении, то его можно убрать. Поэтому после нормализации, упакованная в разрядную сетку согласно формату IEEE 754 Single Precision Format, мантисса числа будет выглядеть так: .10010100000000000000000
Для хранения порядка числа, который указывает, на сколько разрядов была сдвинута запятая, используются отдельные разряды. Старший разряд используется для хранения знака числа. Знак порядка образуется путем вычитания константы из значения порядка.
Перевод числа 312.3125 в формат IEEE 754:
$312.3125_{10} = 100111000.0101_2$
$312.3125_{10} = 1.001110000101_2 \cdot 2^8$
$E_{\text{biased}} = 8 + 127 = 135$ или в двоичной форме: 10000111_2
M = 00111000010100000000000
Знак (S): 0
Порядок (E): 10000111
Мантисса (M): 00111000010100000000000
Итоговая запись в формате IEEE 754 (одинарная точность): 0 10000111 00111000010100000000000

33) Представление логической информации

- 1-true, 0-false
- 16-ти разрядное число содержит 16 логических значений



ОДЗ: $X_i \in \{0,1\}$ $\quad$ где $0 \le i \le 15$

Когда применяется команда AND (поразрядное И) в БЭВМ, то происходит логическое умножение между одной ячейкой памяти и аккумулятором. Соответственно, будет произведено 16 логических операций одновременно. Полученный результат является результатом логической операции, и в большинстве случаев его нельзя трактовать, как математический (т. е. знаковый или беззнаковый, это просто 16 отдельных логических значений.

34) Представление символьной и текстовой информации
Представление текстовой информации в вычислительных машинах основано на кодировании букв алфавитов. Символ - это графическое изображение, которое используется человеком для создания слов, текстов и другой значимой информации.
В ЭВМ представление текстовой информации основывается на кодировании букв и символов при помощи кодовой таблицы. Для работы пользователя с символьной информацией недостаточно сохранять в памяти коды символов. Необходимо еще и отображать текстовую информацию на экране компьютера. Для хранения графических начертаний символов в ЭВМ существуют шрифты (fonts). Они хранят все графические изображения символов в заданном алфавите, и обычно устанавливаются вместе с операционной системой, драйвером принтера или микропрограммой BIOS. В шрифтах каждому коду символа соответствует свое начертание. Отдельно хранятся изображения строчных и заглавных букв, цифр, специальных символов и т. д.
Шрифты бывают векторные и растровые. В растровых шрифтах каждому коду символа соответствует изображение, определенного (в точках) размера. В векторных хранится принцип начертания символа в виде последовательности линий.

35) Символы: ASCII
ASCII расшифровывается как American Standard Code for Information Interchange. В первых 32 позициях кодировки расположены управляющие символы, которые используются в качестве служебных, например, для перемещения курсора на терминале и организации текстов. Организация текста была придумана исходя от принципа работы пишущей машинки. Поэтому, в кодовой таблице есть символы перевода строки (LF), возврата на один символ (BS), возврата каретки (CR) и другие. С помощью символа BS на принтере можно печатать один символ поверх другого (так, например, можно печатать ударение).
Отметим, что разные ОС по-разному кодируют конец строки. Это связано с особенностью работы печатных машинок и телетайпов. UNIX-системы используют один символ CR (при этом виртуальная печатная машинка как бы переводит печатную консоль в начало и автоматически на следующую строку), а Windows - последовательностью из двух: перенос строк и возврат каретки. Из-за этого текстовые файлы между этими ОС не полностью совместимы. При этом старший бит используется не для кодировки символов, а для контроля чётности.

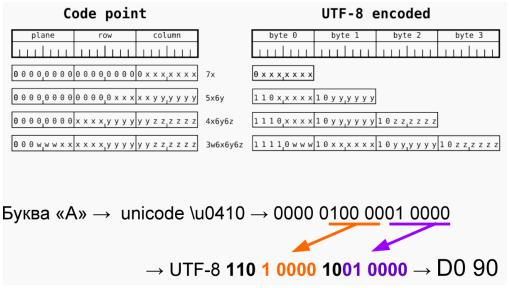
36) Символы: ASCII (КОИ-7H0)КОИ-7H1 (РУС),КОИ7H2 (Mix)
Для представления русских символов в СССР была предложен набор кодовых таблиц КОИ-7 (Код Обмена Информацией). КОИ-7 включает в себя 3 «набора» H0, H1, H2. H0 это просто US-ASCII (однако символ доллара $ заменён на символ валюты), в H1 все латинские буквы заменены на русские; в H2 заглавные латинские буквы оставлены, а строчные заменены на заглавные русские.
Кодировка является семибитной, как и ASCII оставляя старший бит для контроля четности. Другой ее особенностью было то, что русские символы, идентичные по начертанию латинским находились в тех-же местах, что и английские. В случае отсутствия в системе русских шрифтов текст, пусть и с неудобствами в восприятии всегда можно было прочитать с помощью английских шрифтов.
Минусом такого подхода была усложненная сортировка данных в алфавитном порядке. Для сортировки была необходима дополнительная таблица, которая содержала коды символов в алфавитном порядке.

37) Символы: КОИ-8
Когда появилась Extended ASCII, где для символов использовались все 8 разрядов байта, было решено для русских заглавных и строчных символов использовать позиции верхней части таблицы. Данная кодовая таблица получила название КОИ-8. В ней кроме символов русского языка были еще и символы псевдографики для рисования таблиц. Следует заметить, что если по каким-то причинам ПО не поддерживало 8-ми битные кодировки и обрезало старший бит (при контроле четности) то русские символы перемещались в младшую часть таблицы, и их можно было прочитать по сходным с русскими по начертанию английским символам. КОИ-8 до сих пор используется в качестве стандарта для обмена электронной почтой.

38) Символы: ISO8859-5 (ГОСТ-основная)
ISO 8859-5 — 8-битная кодовая страница из семейства кодовых страниц стандарта ISO-8859 для представления кириллицы. ISO 8859-5 была создана в 1988 году на базе «основной кодировки». ISO 8859-5 широко применяется в Сербии и иногда в Болгарии на юниксоподобных системах. В России эта кодировка в настоящее время почти не употребляется. Преимущества по сравнению с КОИ-7: русские буквы по алфавиту. Символ занимает 1 байт.
До широкого использования Unicode в российском сегменте вычислительных систем велись «войны кодировок». «Де факто» используется пять разных таблиц кодировок: с КОИ-8 использовались две кодировки ISO8859-5 (ГОСТ-основная) и cp866 (ГОСТ-альтернативная, в DOS), Win-1251, кириллическая кодировка для компьютеров Macintosh.

39) Символы: WIN1251
Несмотря на многообразие и стандартизацию кириллических символов, компания Microsoft в операционной системе Windows использовала не кодовую страницу CP86 из своей предыдущей операционной системы DOS, а новую, которая была создана на базе кодировок, использовавшихся в ранних «самопальных» русификаторах Windows в 1990—1991 гг. совместно представителями «Параграфа», «Диалога» и российского отделения Microsoft.

40) Символы: UNICODE, UTF-8



В настоящее время используется (предложён в 91 году) общемировая кодовая таблица Unicode. Применение этого стандарта позволяет закодировать очень большое число символов из разных систем письменности. Стандарт состоит из двух основных частей: универсального набора символов (Universal character set, UCS) и нескольких форм представления или кодировок (Unicode transformation format, UTF). Универсальный набор символов перечисляет допустимые по стандарту Юникод символы и присваивает каждому символу код в виде неотрицательного целого числа, записываемого обычно в шестнадцатеричной форме с префиксом U+ (или), например, U+040F.
Семейство кодировок определяет способы преобразования кодов символов для передачи в потоке или в файле. В настоящее время описаны более 136 тысяч символов. В своей начальной части Unicode полностью совпадает с ASCII. Если символ превышает 7 разрядов, то он представляется в виде двух байтов. При этом первые 5 битов предваряются преамбулой 110 и упаковываются в байт, следующий байт идет с преамбулой 10, и остальные 6 разрядов копируются во второй байт. Если символ занимает больше 2 байтов, то кодировка происходит как на картинке.
Проблема UTF-8 состоит в том, что один символ может быть закодирован 1-4 байтами. Наиболее часто используемая операция со строками - определение длины строки. В UTF-8 ее сложнее определить, чем для любой однобайтной кодировки, где длина строки равна количеству байтов. Подсчет количества символов для кодировки UTF-8 реализован системными библиотеками. Недостаток несоответствия длины строки количеству байтов можно исправить, используя кодировку UTF-16 или UFT-32, однако для представления строки в данных форматах используется больше памяти.

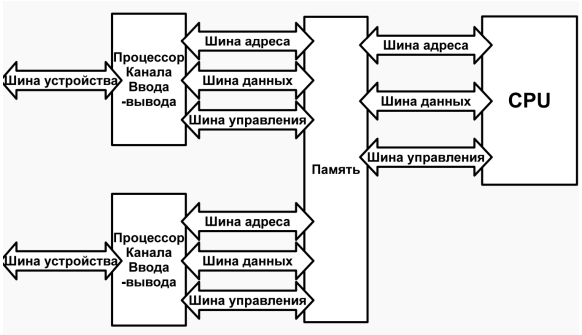
41) Big-endian и Little-endian
Представьте, что у вас в регистре лежит слово, а для пересылки слова в память, вы должны расположить байты этого слова в определенном порядке. Как удобней для ЭВМ выполнить это размещение - с младшего или со старшего байта? Этот вопрос породил большое количество споров. Приверженцев разных способов размещения и сам способ размещения назвали Big-endian и Little-endian (аналогия с "Путешествиями Гулливера").
В современных ЭВМ в основном используется Little-endian младшие разряды слова размещаются в начальных байтах памяти. В течение эволюции ЭВМ было выяснено, что никаких преимуществ один тип кодирования над другим не имеет. До сих пор между различными архитектурами, например Intel и SPARC, возникают проблемы совместимости.

42) Представление строк
Строка в ЭВМ (String) – конечная последовательность символов заданной длины. В БЭВМ не существует поддержки строк на уровне ЭВМ или библиотек, программист должен самостоятельно организовывать их в памяти. Так как длина строки является переменной величиной, то мы должны выделять для хранения строки разное количество байт, учитывая длину строки. В современных вычислительных системах и языках программирования размещение строк обычно организовано двумя разными способами.
В первом признаком конца строки является специальный символ. Обычно используется NUL, а строки называются NULL terminated String. Такой способ применяется в языке Си и подобных ему языках программирования.
Второй способ кодировки используется в Паскаль-подобных языках, где первое слово строки используется для хранения длины строки, и мы знаем, что следующие символы относятся к данной строке.

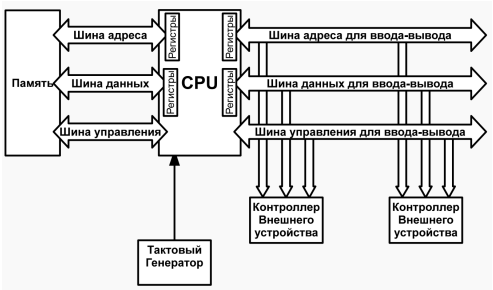
43) История развития ЭВМ
Смена технологической базы = новое поколение
Нулевое поколение - механические компьютеры (1642-1945)
Налоговый сумматор (Паскаль), калькулятор на 4 действия (Лейбниц)
Первое поколение — электронные лампы (1945-1955)
COLOSSUS (1943, Тьюринг), ENIAC (1946, Моушли), IAS (1951, фон Нейман)
Второе поколение — транзисторы (сильно сократился размер) (1955—1965)
TX-0 (1955, МТИ), PDP-1 (1961, DEC), PDP-8, 7090 (IBM), 6600 (1964, CDC)
Третье поколение — интегральные схемы (микросхемы) (1965—1980)
Семейство System/360 (1965, IBM), PDP-11 (1970, DEC) (сфера ЭВМ ушла в коммерцию, а не только в науку)
Четвертое поколение - сверхбольшие интегральные схемы (1980-?)
IBM PC (1981, первый ПК), Apple, Intel, IBM, Dec, Compaq, HP, Sun...
Пятое поколение (1989-?)
небольшие и «невидимые» компьютеры (небольшие кристаллы в бытовых вещах)

44) История развития ЭВМ в СССР/России
Первое поколение - электронные лампы
Лебедев, 1950, МЭСМ
БЭСМ, 1953, БЭСМ - 10000 оп/с, 53КВТ (потребление)
Второе поколение - транзисторы (обособленно от 1 поколения)
5Э92б, 1964, самодиагностика, горячая замена блоков, 500000 оп/с
БЭСМ-6, 1965 год, +конвейерная обработка (выполнение следующей команды начинается до полного окончания выполнения предыдущей команды), удаленное управление по телефонным линиям
Третье поколение - интегральные схемы (взаимствование с Запада)
Директива «Ряд», 1968 год, клонирование S/360, 1971 год - ЕС ЭВМ
Клоны PDP-11
Четвертое поколение - сверхбольшие интегральные схемы
Эльбрус (процессор) — разработка по настоящее время

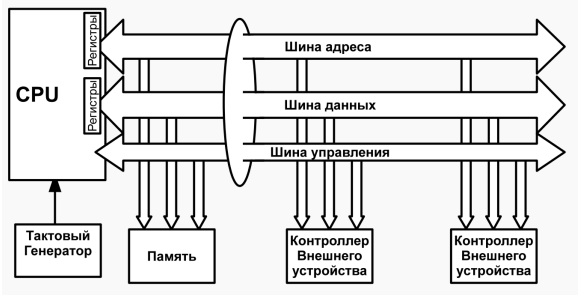
45) Канальная организация
Это первая организация вычислительных машин, она строится вокруг памяти. Скорость всех блоков примерно одинаковая. Для внешних устройств - отдельные процессоры. Все утройства независимы друг от друга.



46) Раздельные шины
Производительность процессора стала быстрее, чем производительность памяти и главным звеном архитектуры стал процессор. Скорость работы внешних контроллеров осталась прежней.



47) Общие шины
Между 2 и 3-4 поколениями пытались как можно сильнее повысить производительность, пришла идея все остальные устройства (кроме процессора) поместить на общую шину (чтобы упростить устройство процессора).



48) Мультиплексирование шин
Мультиплексирование шин (то есть для того, чтобы сократить количество проводов между блоками ВМ, адрес и данные решили передавать по одним и тем же проводам) было принято во время развития 3-его поколения. Те в один импульс тактового генератора передавали адрес, в другой - данные.