

## 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 операционных систем, 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 неразрешённого взаимодействия между собой, драйверы аппаратного обеспечения, подсистемы обобщённого ввода и вывода информации.

## 2) История UNIX/Linux

Название Unix произошло от акронима UNICS (UNiplex Information and Computing System). Разработчики Кен Томпсон, Денис Ритчи 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, SunOS, 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 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, состоят 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 функционирование 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 её работы данные, сегмент `stack`, для хранения адресов возврата и передачи аргументов к подпрограммам. 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. Верхним 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 каталог `/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 за свою тройку и высчитывается так  $rw\bar{x} = 111_2 = 7_8$ ;  $r\bar{x} = 101_2 = 5_8$ ;  $\bar{r}\bar{x} = 001_2 = 1_8$ . Таким образом, рассчитывается десятичная цифра, соответствующая определенным правам для каждой тройки типов прав. Команда `chmod 644 <имя файла>` установит права `rw-r--`.

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 функциями, 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 - страничная 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 величинами, или просто аналогами. В современной вычислительной технике, в качестве аналоговых величин используются значения напряжения. При этом высоте, например, может ставиться в соответствие напряжение от датчика высоты. Аналоговая величина всегда непрерывна, и её 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 производятся 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 ёмкости, которая находится в цепи зарядов транзистора. Ячейка памяти состоит из одного транзистора и одного конденсатора. Недостаток такой памяти: ёмкость постепенно разряжается, соответственно информация теряется. Для того, чтобы поддерживать заряд на ёмкости, предусмотрено устройство регенерации памяти: компьютер раз в определённое время (64 мс) сканирует всю имеющуюся оперативную память и считывает каждую ячейку, возобновляя заряд на конденсаторе. Достоинством DRAM является её 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.

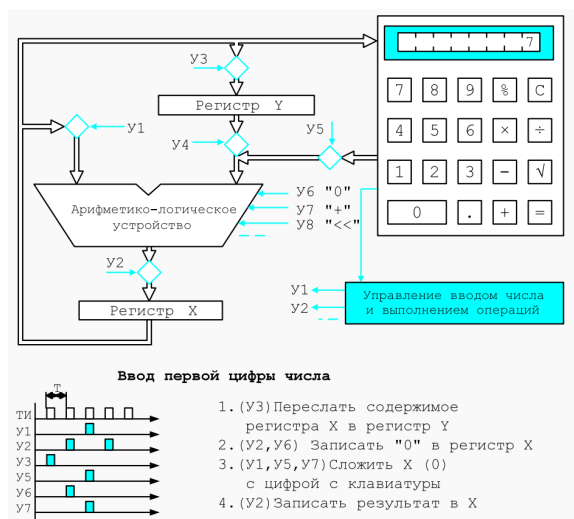
Передачу информации обычно нужно осуществлять не непрерывно, а в строго заданное (тактовым генератором) время. Для этого предусмотрено логическое устройство, называемое вентилем, которое функционирует по принципу хорошо известного нам крана. Вентиль может быть в открытом состоянии, и тогда информация с выхода одного регистра 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 2-х 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 к входу следующего сумматора).

Тактовые генераторы являются «сердцем» ЭВМ (в ЭВМ он обычно один, 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 вентилях, 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 U5, 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 Y. Для этого должен быть открыт вентиль, управляющий записью в регистр Y. Он открывается управляющим сигналом U3. Далее необходимо обнулить регистр X, подготовив его для новой цифры числа, которая была введена с клавиатуры. Для этого должны быть закрыты все вентили, кроме U2 - который осуществляет передачу данных из АЛУ в регистр X и U6 — 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 (U1), цифра с клавиатуры на правый вход АЛУ (U5), и выбирается операция сложения (U7). В конце цикла необходимо передать результат сложения в регистр X (U2), 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 операции «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 вентилях 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 команд и частичное её декодирование. Выполняется для каждой исполняемой команды.

Цикл выборки адреса. Предназначен для обработки адресных команд и выборки адреса операнда с учётом 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 счётчик команд.

Чтение — информация по адресу в IP читается из памяти в DR, IP увеличивается на единицу.

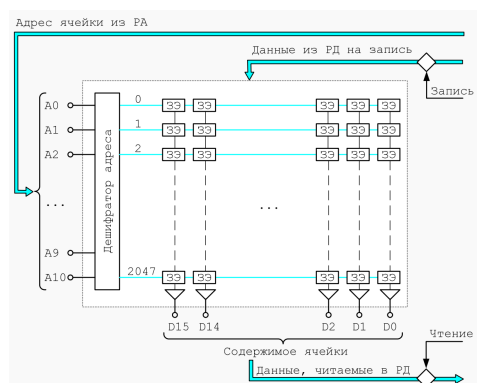
Запись - информация из IR записывается в память по адресу в IP, IP увеличивается на единицу. Используется для ввода программы и данных в режиме оператора.

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

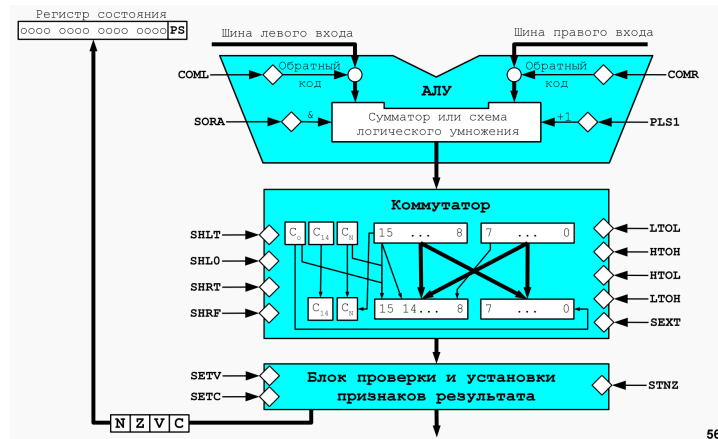


Дешифратор — это устройство, на вход которого подаётся код числа, а на выходе выбирается только одна выходная линия, номер которой соответствует коду на входе дешифратора.

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 находятся 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 чтение 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. Отдельный вентиль предназначен для 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 вентиль SORA (Sum OR And). Если на его входе 1, то будет выполняться поразрядное логическое умножение левого и правого входа АЛУ, иначе - сложение 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 разрядов результат операции плюс сформированные 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 0xE 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 адреса 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-и разрядный регистр БЭВМ SP (Stack Pointer). Этот регистр всегда указывает на вершину стека. При пултовой операции пуск данный регистр обнуляется вместе со всеми остальными регистрами. Когда будет выполнена первая команда PUSH, вызвана подпрограмма или произойдет прерывание, регистр сперва декрементируется, а затем по адресу, записанному в этом регистре, запишется значение, которое зависит от операции.

## 26) Представление точек: фиксированная точка

- **Целые: двоичная точка фиксирована за разрядом с номером 0, веса положительные**

1) номер бита и степень веса разряда соответствуют

№	3	2	1	0
Вес	8	4	2	1
Бит	1	0	1	1

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$$

2) степень веса разряда ~ номеру бита + 1

№	3	2	1	0
Вес	16	8	4	2
Бит	1	0	1	1

$$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 = 16 + 0 + 4 + 2 = 22$$

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 невозможно представить нечетные числа. Двоичную точку можно переместить еще дальше, в зависимости от задачи, при этом будут получаться числа, которые кратны 4-м, 8-и 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 разрядами. Применяя формулу, получим М (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		
<b>-32768</b>	<b>67232</b>	<b>8000</b>	<b>1000 0000 0000 0000</b>
-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		

**K=+3**

$M = b^n - K$   
 $2^4 - 3 = 13$

1 0 0 0 0  
 - 0 0 1 1 1  
 ---  
 1 1 0 1 1

**M = 1 1 0 1**

ИНВ.  
 0 0 1 1 1  
 + 1 1 0 0 0  
 ---  
 1 1 0 1 1

**M = 1 1 0 1**

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 потери значения, и её необходимо обработать отдельно.

В знаковых числах точка возникновения ошибки расположена между представлением чисел -8 и 7. При вычитании из -8 единицы или прибавлении 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

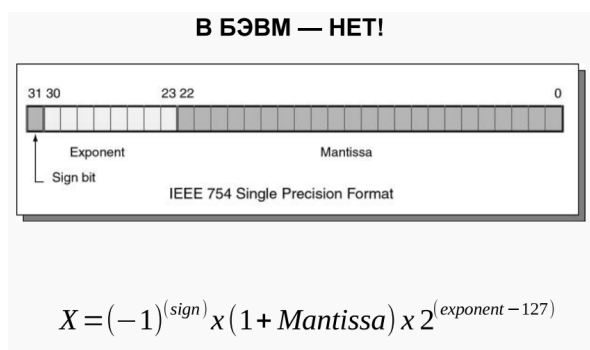
Представление в разрядной сетке	Беззнаковые числа	Знаковые числа
0000 0000 0000 0000	0	0
0000 0000 0000 0001	1	1
...		
0111 1111 1111 1110	32766	32766
0111 1111 1111 1111	32767	32767
1000 0000 0000 0000	32768	-32768
1000 0000 0000 0001	32769	-32767
...		
1111 1111 1111 1110	65534	-2
1111 1111 1111 1111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

**ОДЗ:**

$0 \leq X \leq 2^{16} - 1$

$-2^{15} \leq X \leq 2^{15} - 1$

## 32) Представление чисел с плавающей точкой



Мантисса — это дробная часть числа. Внутри она 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: .100101000000000000000000

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:

$$312.3125_{10} = 100111000.0101_2$$

$$312.3125_{10} = 1.001110000101_2 \cdot 2^8$$

$$E_{\text{biased}} = 8 + 127 = 135 \text{ или в двоичной форме: } 10000111_2$$

$$M = 001110000101000000000000$$

Знак (S): 0

Порядок (E): 10000111

Мантисса (M): 001110000101000000000000

Итоговая запись в формате IEEE 754 (одинарная точность): 0 10000111 001110000101000000000000

### 33) Представление логической информации

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

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 (всего 128) расположены управляющие символы, которые используются в качестве служебных, например, для перемещения курсора на терминале и организации текстов. Организация текста была придумана исходя из 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.

Проблема UTF-8 состоит в том, что один символ может быть закодирован 1-4 байтами. Наиболее часто используемая операция со строками - определение длины. В UTF-8 её 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 UTF-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 называли 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, 53 кВт (потребление)

Второе поколение - транзисторы (обособленно от 1-го поколения)

5Э926, 1964, самодиагностика, горячая замена блоков, 500000 оп/с

БЭСМ-6, 1965 год, +конвейерная обработка (выполнение следующей команды начинается до полного окончания выполнения предыдущей команды), удалённое управление по телефонным линиям

Третье поколение - интегральные схемы (взаимствование с Запада)

Директива «Ряд», 1968 год, клонирование S/360, 1971 год - ЕС ЭВМ

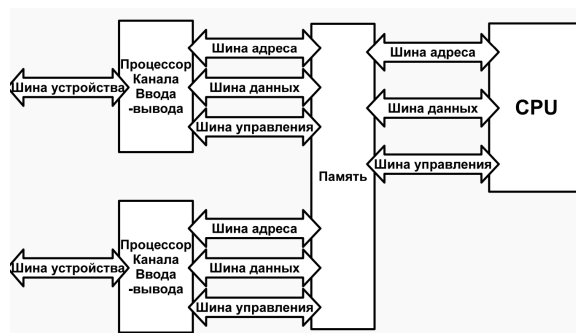
Клоны PDP-11

Четвертое поколение - сверхбольшие интегральные схемы

Эльбрус (процессор) — разработка по настоящее время

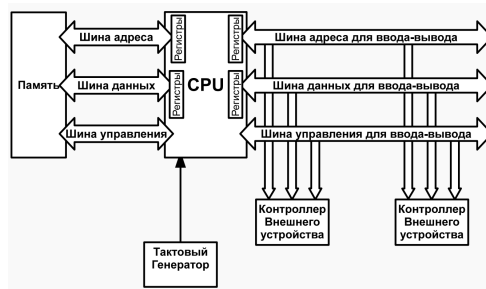
#### 45) Канальная организация

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



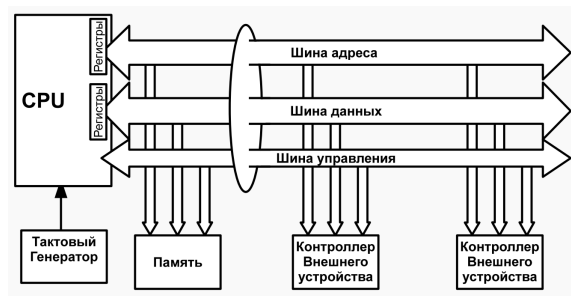
#### 46) Раздельные шины

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



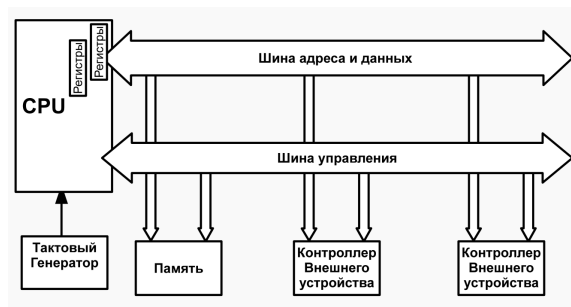
#### 47) Общие шины

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).

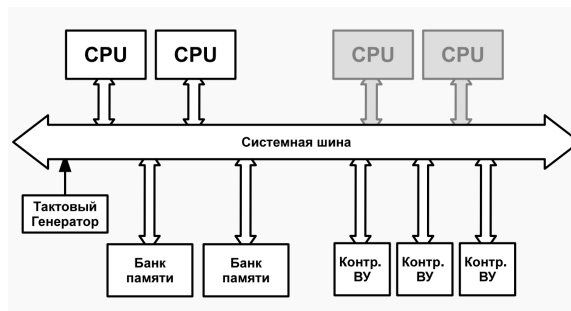


#### 48) Мультиплексирование шин

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-го поколения. Те в один импульс тактового генератора передавали адрес, в другой - данные.

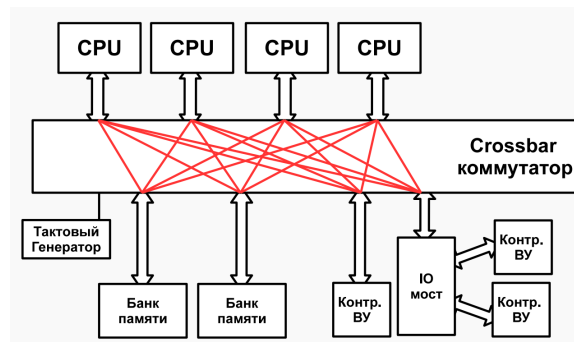


#### 49) Мультипроцессорность: UMA - Uniform Memory Access



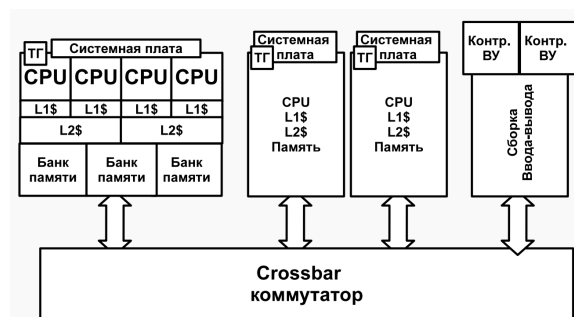
Самая быстрая составляющая - процессор, где-то в 100 раз медленнее память, где-то в  $10^6$  медленнее - контроллеры внешних устройств. Если несколько процессоров (невозможно дальше увеличивать частоту ТГ из-за скорости света), то возникают проблемы с системной шиной: потеря времени, отсутствие синхронизации и т.д. Если  $\geq 3$  процессора, то производительность падает (когда 2 процессора, 1 работает с подпрограммами, другой - с шиной), ибо шины уже не хватает.

## 50) Мультипроцессорность: Коммутатор



Каждый порт соединен с каждым процессором, шину убрали. Каждый процессор в один момент времени может обратиться к одному из устройств. В результате производительность многопроцессорных машин сильно возросла. В явном виде также разделили устройства на быстродействующие и медленные (через мосты подключаются).

## 51) Мультипроцессорность: NUMA – Non-Uniform Memory Access



Предназначена для того, чтобы еще больше увеличивать кол-во процессоров (до сотен). У каждого процессора локальный кэш, локальная память и тактовый генератор. Все это мезиво реализовано в виде одной платы, которая соединена с коммутатором. Память общая для процессоров, но каждый процессор обращается к общей памяти с разной скоростью. Теоретически каждый процессор может работать на своей частоте, также можно на горячую менять элементы системы.

## 52) Современные коммерческие процессоры

Разрядность адреса и данных 16/32/64 бита

Тактовые частоты 500 МГц - 5 ГГц.

Многопроцессорные 1-100+ CPU

Многоядерные 1-16 ядер

От 1 ГБ до терабайтов ОЗУ

Используют кэш-память разных уровней

Суперскалярные (процессор может выполнять более 1 команды за 1 такт)

CISC, RISC, VLIW

## 53) CISC, RISC, VLIW

CISC (Complete Instruction Set Computer) – полный набор команд микропроцессора:

нефиксированное значение длины команды;

арифметические действия кодируются в одной команде;

небольшое число регистров, каждый из которых выполняет строго определённую функцию.

Недостатки CISC архитектуры: высокая стоимость аппаратной части; сложности с распараллеливанием вычислений.

Состав и назначение их регистров существенно неоднородны, широкий набор команд усложняет декодирование инструкций, на что расходуются аппаратные ресурсы. Возрастает число тактов, необходимое для выполнения инструкций. К процессорам с полным набором инструкций относится семейство x86. Лидером в разработке CISC-процессоров считается компания Intel со своей серией x86 и Pentium.

Для CISC-процессоров характерно:

сравнительно небольшое число регистров общего назначения;

большое количество машинных команд, некоторые из которых нагружены семантически аналогично операторам высокоуровневых языков программирования и выполняются за много тактов;

большое количество методов адресации;

большое количество форматов команд различной разрядности.

RISC - Reduced Instruction Set Computer – архитектура компьютера с сокращенным набором команд. RISC-архитектура предполагает реализацию в ЭВМ сокращенного набора простейших, но часто употребляемых команд, что позволяет упростить аппаратные средства процессора и, благодаря этому, получить возможность повысить его быстродействие. Современные процессоры типа RISC характеризуются следующими особенностями:

упрощенный набор команд, имеющих одинаковую длину;

отсутствуют макрокоманды, усложняющие структуру процессора и уменьшающую скорость его работы;

взаимодействие с оперативной памятью ограничивается операциями пересылки данных;

уменьшено число способов адресации.

Новый подход к архитектуре команд процессора значительно сократил площадь, требуемую для него на кристалле интегральной микросхемы. Это позволило резко увеличить число регистров. В результате процессор стал на 20-30% реже обращаться к оперативной памяти. Особенностью RISC архитектуры является механизм перекрывающихся окон, предназначенный для уменьшения числа обращений к оперативной памяти и межрегистровых передач, что способствует увеличению производительности ЭВМ.

Наиболее широко используемые в настольных компьютерах процессоры архитектуры x86 ранее являлись CISC-процессорами, однако новые процессоры, начиная с Intel 486DX, являются CISC-процессорами с RISC-ядром. Они непосредственно перед исполнением преобразуют CISC-инструкции x86-процессоров в более простой набор внутренних инструкций RISC.

VLIW – появилась в России. Она не попадает под принципы фон Неймана (нарушает принцип программного управления, т. е. последовательного выполнения команд). Архитектура ЭВМ с длинным командным словом (VLIW – Very Long Instruction Word) позволяет сократить объем оборудования, требуемого для реализации параллельной выдачи нескольких команд. Вместо того чтобы выдавать на эти устройства последовательные команды, операции упаковываются в одну очень длинную команду. Ответственность за вывод параллельно выдаваемых для выполнения команд полностью ложится на компилятор.

Для машин с VLIW-архитектурой был разработан новый метод планирования выдачи команд – трассировочное планирование. Во время планирования генерируется длинное командное слово. Процесс упаковки команд последовательной программы в длинные командные слова продолжается до тех пор, пока не будет оптимизирована вся программа.

Примеры процессоров на этой архитектуре – Intel Itanium (архитектура IA-64, Merced), микропроцессоры серии «Эльбрус» («Эльбрус 2000», «Эльбрус S»).

#### 54) Характеристики памяти

Месторасположение (чем ближе к процессору, тем дороже):

процессорные, внутренние (внутри вычислительной машины), внешние

Емкость:

В метрических (Кило-) и двоичных (Киб-) множителях (минус в том, что может быть путаница)

Единица пересылки:

Слово (16 разрядов, за 1 такт), строка кэша, блок на диске

Метод доступа:

Произвольный (адресный), ориентированный на записи (прямой), последовательный, ассоциативный

Быстродействие и временные соотношения:

Время доступа  $T_d$

Длительность цикла памяти (время обращения)  $T_{\Sigma}$

Время чтения и время записи (они могут быть сильно разными)

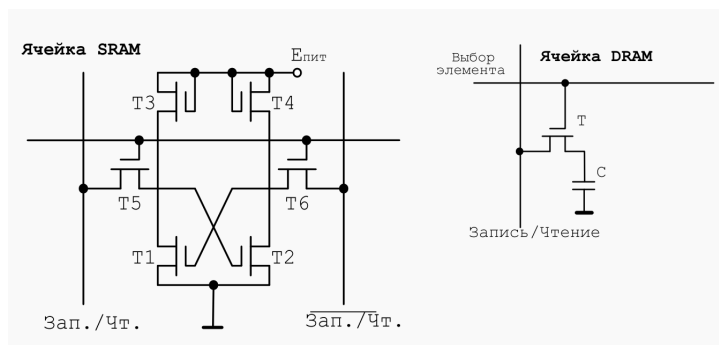
Время восстановления  $T_v$  (восстановление системы в исходное состояние, например, магнитная лента)

Скорость передачи информации

Физический тип и особенности;

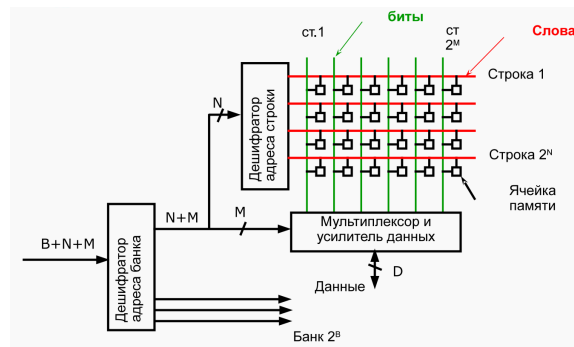
Стоимость (обычно стоимость одного байта).

#### 55) Статическая vs Динамическая память





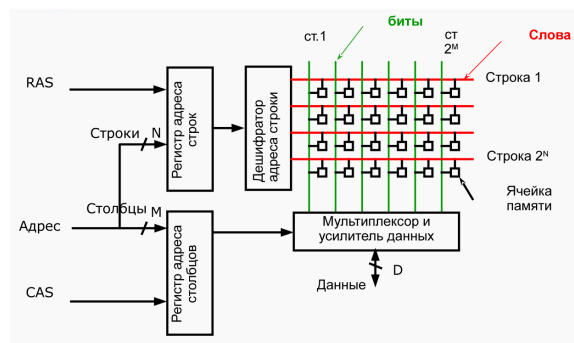
## 56) Адресуемая память



Мультиплексор - устройство, которое осуществляет передачу неизменным сигнала с входа на выход (входов много, а выход один).

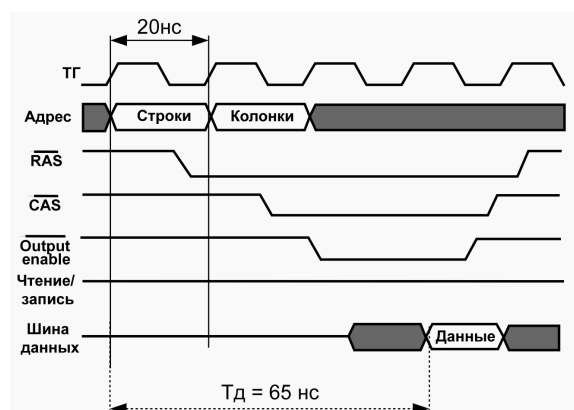
Имеем регистр, в котором первые биты выделены под M, следующие под N, оставшиеся - под Bank (банк - это табличка на слайде, их много в памяти).

## 57) Адресуемая память с фиксацией строк и столбцов



Люди решили передавать адрес не полностью, а разбить его на 2 части: адрес строки и адрес столбца. Половина адреса передаётся по шине памяти в один такт, другая половина - в другой (RAS - row address select, CAS - column). Теперь строка и столбец сохраняются в регистрах, и мы можем инкрементировать и декрементировать их, чтобы обращаться к следующим ячейкам памяти. То есть наша память будет работать поначалу медленнее в 2 раза, чем просто адресуемая память, но обращаться к ячейкам мы будем быстрее.

## 58) Синхронная память SDRAM



Память синхронна (появился собственный тактовый генератор, теперь она работает на своей частоте), сигналы инвентированы (так было удобнее производителям); output enable - подключение к выходу памяти. Также память синхронизируется с частотой процессора.

DDR SDRAM (double data rate SDRAM) отличается от обычной SDRAM тем, что на той же частоте имеет вдвое большую пропускную способность. По принципам работы она похожа на SDRAM, но, в отличие от неё, может принимать и передавать данные на обоих фронтах тактовых импульсов.

## 59) Конструктивные особенности современной памяти

Burst mode - пакетный режим (как раз быстрое обращение к ячейкам по RAS и CAS (они уже установлены), данные передаются пакетом - то есть много за раз)

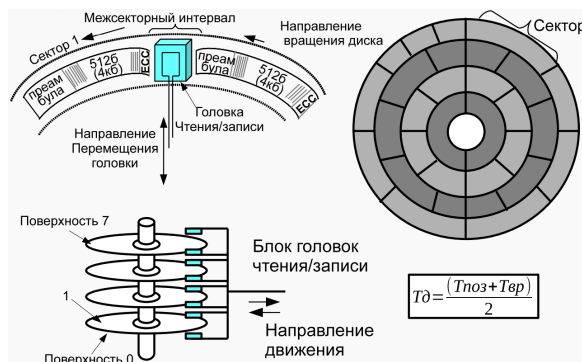
Double Data Rate - передача данных и по фронту, и по спаду

SPD - чип, содержащий идентификационную информацию (по нему процессор понимает, может ли он работать с памятью на одной частоте)

Interleaving - расслоение памяти, повышает производительность (после чтения 1 слова сразу прочесть другую ячейку памяти)

DDR4-2133 (версия памяти, поколение, частота ТГц) 8192MB (объём) PC4-17000 - индекс производительности

## 60) Память, ориентированная на записи



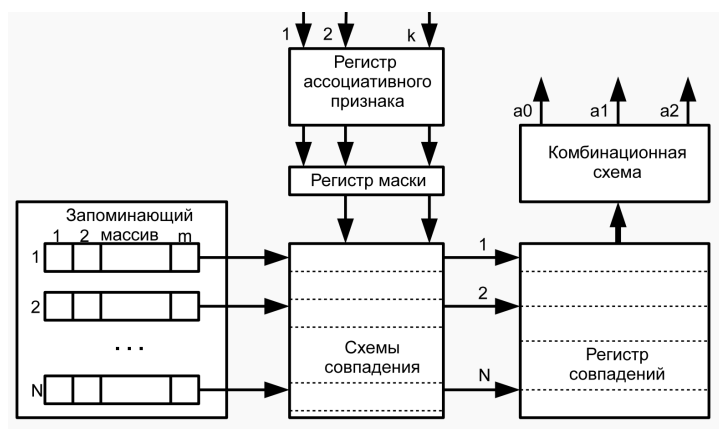
Блин - поверхность, на которой расположена информация. Головки читают информацию с намагниченной поверхности. Головка идет по диску, диск же разбит на секторы, у сектора есть преамбула (чтобы понять, что начался новый сектор), есть поле для контроля чётности дисков.

Время доступа к диску = время позиционирования (головка идёт) + время ожидания поворота диска до нужного сектора пополам (берём среднее).

## 61) Память с последовательным доступом

Чтобы получить доступ к ячейке памяти  $n$ , нужно пролистать первые  $n - 1$  штук до неё. Это дорого, громоздко и супер медленно, но зато можно хранить огромные объёмы данных (поэтому цена за байт ничтожна). Примером является ленточный накопитель (ленту бесконечно разгонять нельзя, поэтому там вращается головка, дорожки записываются под наклоном). Плёнка энергоэффективна: если данные уже записаны, плёнке не требуется питания для их хранения. Плёнка надёжна: вероятность ошибок при записи или чтении на 4-5 порядков ниже, чем у жёстких дисков. Плёнка безопасна: в отличие от дисков, которые, как правило, подключены к компьютеру постоянно, картриджи с катушками могут храниться без подключения к устройствам, что защищает данные на плёнке от чтения или модификации злоумышленниками или от ошибок из-за человеческого фактора.

## 62) Структура ассоциативного запоминающего устройства



Информация выбирается не по адресу, а по нужному нам признаку. Каждая ячейка памяти содержит схему сравнения (то есть всё, что поступает на неё, сравнивается, есть ли такое содержимое, и если да, то те ячейки, что совпали, идут в комбинационную схему). Подобным образом устроен кэш.

### 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):

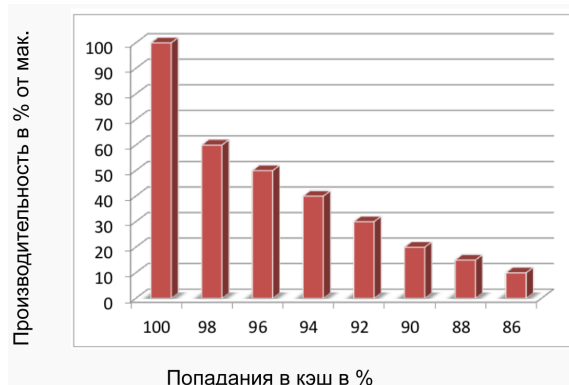
Запрашиваемые данные отсутствуют в кэш-памяти. Кэш-контроллер запрашивает данные из оперативной памяти.

### 64) Пирамида памяти

	Объем	Тд	*	Тип	Управл.
CPU	100-1000 б.	<1нс	1с	Регистр	компилятор
L1 Cache	32-128Кб	1-4нс	2с	Ассоц.	аппаратура
L2-L3 Cache	0.5-32Мб	8-20нс	19с	Ассоц.	аппаратура
Основная память	0.5Гб-4Тб	60-200нс	50-300с	Адресная	программно
SSD	128Гб-1Тб/drive	25-250мкс	5д	Блочн.	программно
Жесткие диски	0.5Тб-4Тб/drive	5-20мс	4м	Блочн.	программно
Магнитные ленты	1-6Тб/к	1-240с	200л	Последов.	программно

Чем выше - тем дороже, меньше объёма и быстрее. \* - время с точки зрения человека (порядки эквивалентны).

### 65) Влияние промахов кэш-памяти



### 66) Сегментно-страничная виртуальная память

При реализации страничной виртуальной памяти виртуальное адресное пространство делится на части одинакового фиксированного размера, называемые виртуальными страницами. Если виртуальное адресное пространство какого-то процесса не кратно размеру страницы, то последняя страница дополняется фиктивной областью. Физическая оперативная память также делится на части такого же размера, называемые физическими страницами. Размер страницы выбирается кратным степени двойки. Копия всех виртуальных страниц хранится на диске. Страничное распределение памяти предусматривает только механическое разбиение на страницы, оно не позволяет различным образом организовать работу со страницами, содержащими команды и данные.

При сегментном распределении памяти виртуальное адресное пространство делится на части, называемые сегментами, размер которых определяется смысловым значением содержащейся в них информации. Максимальный размер сегмента определяется разрядностью процессора, для 32-разрядного процессора он равен 4 Гбайт. Сегменты никак не упорядочиваются друг относительно друга. Виртуальный адрес задается парой чисел: номером сегмента и смещением внутри сегмента. При загрузке процесса в оперативную память помещается только часть его сегментов, полный образ виртуального адресного пространства находится на диске. Для каждого загружаемого сегмента подыскивается непрерывный участок свободной физической памяти достаточного размера.

## 67) MMU и TLB

В каждом компьютере есть устройство MMU (memory manager unit): на вход поступает виртуальный адрес, есть таблица преобразований, на выход поступает физический адрес. Табличкой управляет операционная система. Если ничего не размаплено, то время обращения к памяти увеличивается кратно.

Есть еще устройство TLB: туда записывается виртуальный адрес и соответствующий ему физический адрес (то есть готовый мапинг в виде кэша).

## 68) История сети Internet

1957 - запуск первого спутника Земли (нужно было что-то делать, чтобы догнать СССР)

1958 - Advanced Research Projects Agency (ARPA, агентство новых технологий для ВС США)

1963 — J.C.R.Licklider - первая концепция компьютерной сети

1969 — ARPANET (предок интернета) в ведущих лабораториях и исследовательских центрах США

1976-Хероx - локальная сеть Ethernet

1982 - ARPA - единый стек протоколов TCP/IP

1983-84 - FidoNet и BBS, ARPANET → Internet

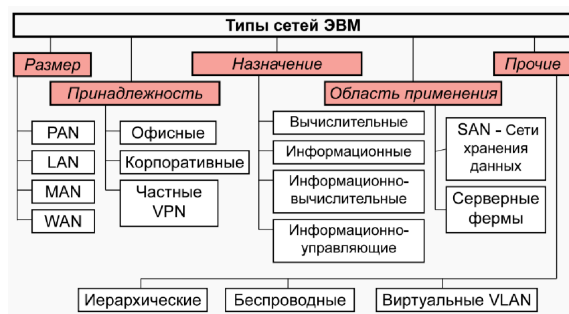
1991 - Tim Berners-Lee, CERN, концепция WWW, первый http-сервер (с появлением ПК)

## 69) Понятие сети ЭВМ

Средства вычислительной техники (СВТ): ЭВМ, вычислительные комплексы (ВК, то есть несколько ЭВМ с одной задачей) и вычислительные системы (ВС) - реализуют обработку данных.

Средства телекоммуникаций (связи) (СТК): совокупность каналов связи (провод) и каналобразующей аппаратуры (модели, которые организуют передачу данных) - реализуют передачу данных.

Сеть ЭВМ (вычислительная сеть, компьютерная сеть) = СВТ + СТК.



LAN - локальная сеть, WAN - глобальная (разница в модемах и протоколах, расстояние имеет значение (скорость света конечна)), MAN - гибрид LAN'a и WAN'a, PAN - суперлокальная система, которая крутится вокруг одного пользователя (Bluetooth).

## 70) Сообщение, пакет

Сообщение, которое мы передаём по сети, обычно очень большое, поэтому его принято разбивать на части - пакеты (около 1500 байт). В начале сообщения есть заголовок, который соответствует уровню передач (IP адреса приёмника, служебная информация). В конце - контрольная сумма (заголовок и КС добавляется к каждому пакету). Разбиением занимается один из верхних уровней.

## 71) Модель взаимодействия открытых систем (OSI)

Модель в реальности не используется (используется как референсная). Модель OSI делит протоколы на семь уровней, а не на четыре, как TCP/IP.

## 72) Модель TCP/IP

Набор интернет-протоколов (т. е. набор правил, которые описывают, как должен происходить обмен информацией) стандартизирует метод связи в интернете. Два основных протокола в наборе интернет-протоколов – это Transmission Control Protocol (TCP) и Internet Protocol (IP), известные под общим названием TCP/IP.

Сетевые протоколы работают в многоуровневой модели, и реализация такой модели называется сетевым стеком. Протоколы на самом нижнем уровне взаимодействуют с базовым сетевым оборудованием, в то время как приложения взаимодействуют с протоколами на верхних уровнях. На практике каждый уровень реализуется с помощью конкретных сетевых протоколов. Каждый сетевой уровень представляет собой область определённых обязанностей, и протоколы должны выполнять обязательства назначенного им уровня.

Четыре уровня модели TCP/IP, начиная снизу вверх, – это канальный, межсетевой, транспортный и прикладной уровень. Протоколы на каждом уровне взаимодействуют с протоколами на соседних уровнях. Исходящая передача от узла проходит вниз через сетевые уровни, входящая передача на узел проходит вверх по сетевым уровням в обратном порядке.

Протоколы, работающие на прикладном уровне, обеспечивают функциональность, специфичную для конкретного приложения, например, отправку электронной почты или поиск веб-страницы. Эти протоколы выполняют задачи, которые хотят решить конечные пользователи (или внутренние службы). Пример: HTTP, SSH.

Протоколы транспортного уровня обеспечивают канал связи для приложений для отправки и получения данных между узлами (хостами). Приложение структурирует данные в соответствии с протоколом прикладного уровня, а затем передает эти данные протоколу транспортного уровня для доставки на удаленный узел.

Протоколы межсетевого уровня обеспечивают механизм взаимодействия между сетями. Этот уровень отвечает за идентификацию узлов с помощью адресов и обеспечивает маршрутизацию данных из сети в сеть через интернет (например, IP).

Протоколы канального уровня обеспечивают взаимодействие в локальной сети. Протоколы этого уровня тесно связаны с типом сетевого оборудования в локальной сети, например, с Wi-Fi. Протоколы межсетевого уровня опираются на протоколы канального уровня для взаимодействия в локальной сети. Пример: Wi-Fi, Ethernet.

### 73) Уровень передающей среды

Коаксиальный кабель (устарел): тонкий (до 50 метров) и толстый (до 500).

Витая пара 10Base-T, 100Base-T: несколько категорий в зависимости от пропускной способности (до 100 ГБ/с на 15 метров без потерь).

Оптика: ST, SC, LC (до 80 км в режиме одного луча).

Wireless (беспроводной тип): Wi-Fi, 3G, 4G (до 15 ГБ/с).

### 74) Канальный уровень Ethernet

Физические и логические соединения между узлами называются сетевыми каналами. Протоколы канального уровня используются устройствами в одной сети для связи друг с другом. Каждое устройство на канале имеет сетевой адрес, который однозначно идентифицирует его. Для многих протоколов канального уровня этот адрес известен как media access control address (или MAC-адрес, он уникален в рамках одной локальной сети). Данные канального уровня разделены на небольшие блоки, известные как кадры. Заголовок кадра содержит MAC-адреса источника и назначения, описание типа данных, передаваемых в секции данных кадра.

Ethernet — это технология канального уровня локальных компьютерных сетей, обеспечивающая передачу данных на короткие расстояния. Ethernet определен стандартом IEEE 802.3. В Ethernet обычно используется кабель с парами медных проводов внутри, который заканчивается разъёмом, известным как RJ45.

Сетевой концентратор – это сетевое устройство, которое соединяет несколько устройств в локальной сети без какой-либо информации, касающейся отправляемых кадров. Простой концентратор может предоставлять несколько портов Ethernet для подключения устройств. Концентратор просто ретранслирует каждый кадр, полученный на одном физическом порту, на все остальные порты.

### 75) Сетевой уровень IP

Межсетевой уровень позволяет данным выходить за пределы локальной сети. Основным протоколом, используемым на этом уровне, называется Internet Protocol (IP). Он обеспечивает маршрутизацию – процесс определения пути для данных, передаваемых между сетями. Каждому узлу в интернете присваивается IP-адрес, номер, который однозначно идентифицирует узел в глобальной сети Интернет. Данные, передаваемые через межсетевой уровень, называются пакетом, который заключён в кадр канального уровня.

32-битный IP-адрес (IPv4) обычно отображается в точечно-десятичном виде, т. е. 32 бита разделяются точками на четыре группы по 8 бит в каждой, 8-битные числа отображаются в десятичной системе счисления. Подсеть делит IP-адрес на две части: сетевой префикс, который разделяют все устройства одной подсети, и идентификатор узла, который уникален для каждого узла в этой подсети. Количество битов, входящих в сетевой префикс, зависит от конфигурации сети.

Компьютеры, подключенные к одной локальной сети, имеют IP-адреса, начинающиеся с одинаковых старших битов, и могут напрямую взаимодействовать друг с другом на канальном уровне. Компьютеры, находящиеся в разных подсетях, должны передавать свой трафик через маршрутизатор (роутер) – устройство, соединяющее подсети и работающее на межсетевом уровне.

Количество битов, зарезервированных для сетевого префикса, обычно выражается одним из двух способов. Метод бесклассовой адресации (Classless Inter-Domain Routing, CIDR) указывает IP-адрес, за которым следует косая черта (/), а затем количество битов, используемых для сетевого префикса. Другим распространённым способом представления количества битов префикса является маска подсети – 32-битное число, в котором двоичная 1 используется для каждого бита, являющегося частью сетевого префикса, а 0 – для каждого бита, являющегося частью номера узла.

Выполнение побитового логического И из IP-адреса и его маски подсети дает первый адрес в подсети. Этот первый адрес, где все биты номера узла равны 0, служит идентификатором для самой подсети. Его обычно называют сетевым идентификатором. Два компьютера, имеющих общий сетевой идентификатор, находятся в одной подсети.

### 76) Сетевой уровень IP: маршрутизация

Маршрутные таблицы: хранятся на каждом маршрутизаторе и определяют, куда отправлять пакеты для определённых сетей. Каждая запись состоит из: сети назначения, маски подсети, адреса следующего узла, интерфейса, через который отправлять пакеты, метрики (приоритет маршрута).

Когда устройство хочет отправить данные, оно определяет IP-адрес получателя. Устройство сравнивает IP-адрес назначения с собственной маской подсети: если адрес принадлежит той же сети, данные отправляются напрямую, иначе пакет передаётся маршрутизатору. Пакет попадает в маршрутизатор, где анализируется заголовок IP: IP-адрес назначения используется для поиска подходящего маршрута в таблице маршрутизации. Если в таблице маршрутизации есть подходящий маршрут, пакет пересылается на указанный интерфейс и next hop (следующий узел). Если маршрута нет:

пакет отправляется по "маршруту по умолчанию" (default route), если он настроен, если нет, то пакет отбрасывается, и отправителю отправляется сообщение о недостижимости цели. На каждом узле изменяется канальный уровень: новый MAC-адрес назначения (для следующего маршрутизатора или конечного устройства) добавляется в кадр, а IP-адрес остаётся неизменным.

#### 77) DHCP

Это протокол, который позволяет автоматически выдавать IP-адреса.

#### 78) Сервис имен DNS и другие

Людям неудобно использовать IP-адреса и хотелось их преобразовать во что-то человеческое. Был придуман сервис имён, который преобразует имя, понятное человеку, в IP. Простейший способ задать имя - прописать его в /etc/hosts DNS (Domain Name System) — это система имён, которая преобразует доменные имена (например, google.com) в IP-адреса.

Пользователь вводит доменное имя;

Запрос отправляется на локальный DNS-сервер (обычно провайдера);

Если локальный сервер не знает IP-адреса, он обращается к корневому серверу, затем к серверу домена верхнего уровня (например, .com), а затем к авторитетному серверу домена (содержит информацию о домене и отвечает на запросы о конкретном домене);

Авторитетный сервер возвращает IP-адрес домена;

Устройство подключается к серверу по полученному IP.

Есть ещё примеры сервисов имён для хранения много чего (пользователей, телефонов и так далее). Пример - Microsoft Active Directory - LDAP сервер (для хранения всего).

#### 79) Транспортный уровень

Транспортный уровень обеспечивает канал связи, который приложения могут использовать для отправки и получения данных. Существует два широко используемых протокола транспортного уровня: Transmission Control Protocol (TCP) и User Datagram Protocol (UDP). TCP обеспечивает надежное соединение между двумя узлами. Он гарантирует, что ошибки сведены к минимуму, данные поступают в порядке, потерянные данные отправляются повторно и т. д. Данные, передаваемые с помощью TCP, называются сегментами. С другой стороны, UDP – это протокол «негарантированной доставки», т. е. его доставка ненадежна. UDP предпочтительнее, когда скорость важнее надежности. Данные, передаваемые с помощью UDP, называются датаграммами.

#### 80) Прикладной уровень

В то время как три нижних уровня обеспечивают общую основу для коммуникации через интернет, протоколы прикладного уровня направлены на выполнение конкретной задачи. Например, веб-серверы используют протокол передачи гипертекста (HTTP) для получения и обновления веб-контента. Серверы электронной почты используют простой протокол передачи почты (SMTP) для отправки и получения сообщений электронной почты.