**Внимание**

*На зачете Вам предстоит ответить на 2 билета: один из первой лекционной части и второй практической части*

Первая часть (лекции)

1. *Две формы представления информации. Способы представления дискретной информации. Системы счисления, используемые в вычислительной технике: 2-я, 8-я, 10-я, 16-я, двоично-десятичная.*
2. Аналоговая (непрерывная)

Аналоговая величина всегда непрерывна и ее поведение полностью копирует поведение соответствующей физической величины. Количество возможных значений бесконечно. С ней можно проводить вычисления – складывать, вычитать, дифференцировать. Плюсами такого представления информации является высокое быстродействие и малая погрешность вычислений, тк погрешность не меняется и зависит от пропускной способности ЭВМ. Минус – большие габариты машин.

1. Цифровая (прерывная, дискретная)

Набор напряжений трактуется ограниченным количеством заданных значений. Данные передаются «пакетами», которые в дальнейшем расшифровываются. Цифровые ЭВМ для определенных задач (к примеру, интегрирование и дифференцирование) обладают меньшей скоростью выполнения, однако такой способ представления информации значительно упростил конструкцию вычислительных машин.

Представление информации в привычной для нас десятичной сс значительно усложняет архитектуру, тк устройство в таком случае должно будет различать огромное количество состояний. В связи с этим было принято решение использовать другие сс.

Системы счисления, используемые в ВТ:

* *Двоичная*0 или 1 для каждого бита, либо есть сигнал, либо его нет, является основной для ЭВМ
* *Восьмеричная*

от 0 до 7. Широко использовалась в программировании и компьютерной документации, но позднее была почти полностью вытеснена шестнадцатеричной.

* *Десятичная*

“Не до конца” удобная для машины, однако понятная для человека, при написании кодов, мнемоник и вообще выводов(по типу калькулятора)

* *Шестнадцатеричная*  
  от 0 до F. Удобна, поскольку минимальной адресуемой единицей памяти является байт (8 бит), значения которого удобно записывать двумя 16-ричными цифрами.
* *Двоично-десятичная*

Используется для хранения десятичных чисел в памяти компьютера.  
Десятичная цифра задается 4 битами (от 0000 до 1001).  
+Удобно для вывода чисел на индикацию.   
+Для дробных чисел при переводе в десятичный формат не теряется точность.  
-Требует больше памяти.   
-Усложнены арифметические операции.

В свою очередь 8я и 16я более трудоемка для эвм, однако упрощает представление информации для понимания ее человеком.

1. *Представление чисел с фиксированной точкой. Прямой, обратный и дополнительный код. Формирование битовых признаков переноса, переполнения, отрицательного результата, нуля*

В ЭВМ в основном используются три вида представления чисел – числа с фиксированной точкой, плавающей и строка символов.

Особенностью представления чисел с фиксированной точкой является то, что они должны размещаться в разрядной сетке. В зависимости от положения точки определяется вес каждого символа в зависимости от заданной сс. С помощью фиксированной точки представляются как целые, так и вещественные числа.

Прямой код числа – обычное его представление в заданной сс, обратный – инвертированный прямой код. С помощью дополнительного кода представляются отрицательные числа, что, в свою очередь упрощает конструкцию АЛУ. Именно обратный код обычно используется в вычислительных машинах. Доп код положительного числа равен самому числу, для отрицательных – дополнение до максимального положительного числа +1. Простейший способ вычислить доп код числа – инвертировать его и прибавить 1.

По окончании выполнения операции с числами устанавливаются признаки результата. Признаки отрицательного результата и нуля понятны исходя из их названия: отрицательность устанавливается по старшему биту результата (1-отрицательное, 0-положительное), ноль – при всех битах сетки равных 0. Существуют такие явления как перенос и переполнение. Перенос – для беззнаковых чисел, является признаком выхода за размерную сетку. Переполнение – для знаковых. Формируется как сложение по модулю 2 поразрядных переносов в старший и из старшего разрядов.

1. *Представление символьных и строковых данных. Принципы построения кодовых таблиц ASCII, КОИ-8, ISO 8859-5, Windows-1251, UTF-8, UTF-16.*

В ЭВМ представление текстовой информации основывается на кодировании букв и символов при помощи кодовой таблицы, где каждому символу соответствует номер. Для работы с символьной информацией необходимо также сохранять визуальное представление того или иного символа в шрифтах. Они, шрифты, бывают растровые – состоят из точек и не могут масштабироваться, те это конечное представление символа и векторные – которые чертятся последовательностью линий в пределах заданной площади.

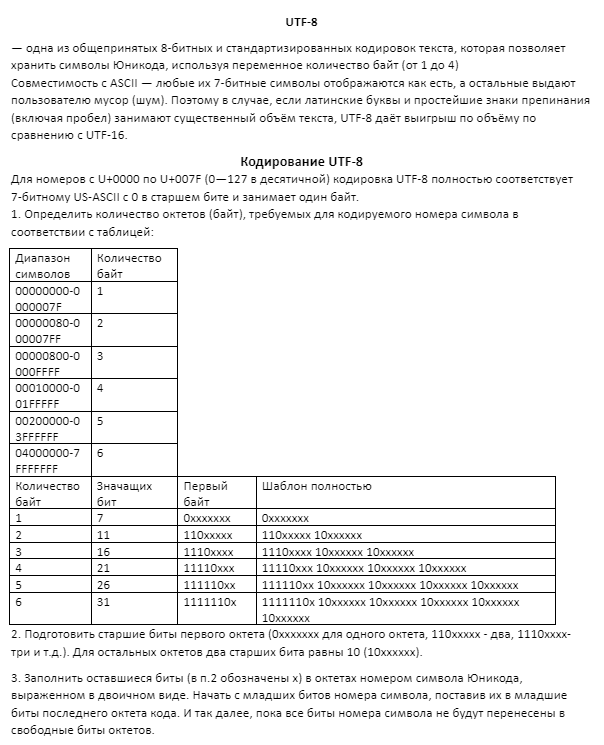
ASCII – American Standard Code for Information Interchange. Она состоит в верхней части из служебных символов, которые отвечали за контроль начертания (перемещение каретки, переход на строки, пробелы), символов и знаков препинания, цифр и заглавных и маленьких букв латинского алфавита, которые, в свою очередь, были расположены одна под другой в соответствии большая-маленькая. Символы кодировались 7ю разрядами, а старший, 8ой, использовался как бит четности для контроля правильности переданных данных.

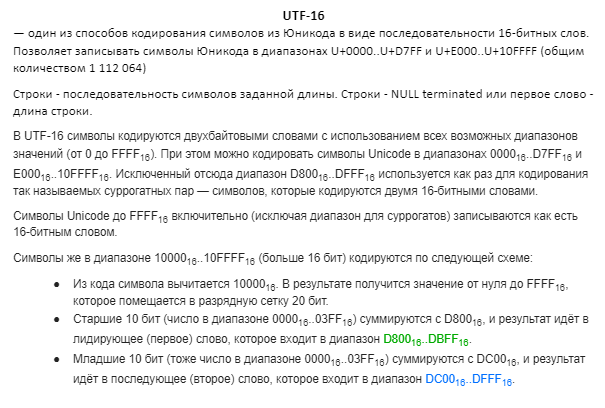
В дальнейшем появилась интерпретация данной кодировки с русским алфавитом КОИ-7, после чего для удобства перестали использовать 8 бит для проверки корректности данных и появилась кодировка КОИ-8. Она представляла собой кодировку ASCII с дополнительными символами псевдографики и полным русским алфавитом заглавных и строчных букв. При этом их расположение определялось соответствием русских-английских, что позволяло при сбое кодировки прочитать сообщение, написанное транслитом. Однако такое расположение букв затрудняло упорядочивание информации по алфавиту и приходилось для данной задачи хранить отдельную таблицу, где коды букв располагались в привычном нам алфавитном порядке.

ISO8859-5 (ГОСТ-основная) – 8битная страница из семейства стандарта ISO-8859 для представления кириллицы. Числа под буквами соответствуют шестнадцатеричному коды буквы Юникода. Недостатком является отсутствие многих нужных символов как тире, кавычки-елочки, градус и другие.

Несмотря на уже существующие многообразие кодировок компанию Microsoft это не утроило, и она разработала свою кодировку WIN1251, которая использовалась в русифицированной версии ос Windows и состояла из символов пунктуации, русского алфавита и все символы для других славянских языков. Минусами является отсутствие символов псевдографики и совпадение кода буквы я со служебным символом в некоторых других кодировках.

Unicode – стандарт, определяющий таблицу символов в соответствие значение в виде U+число в 16сс.





Строки в ЭВМ могут представляться в 2 стандартных видах (хотя сейчас этих видов больше, к примеру язык программирования Java использует собственный).

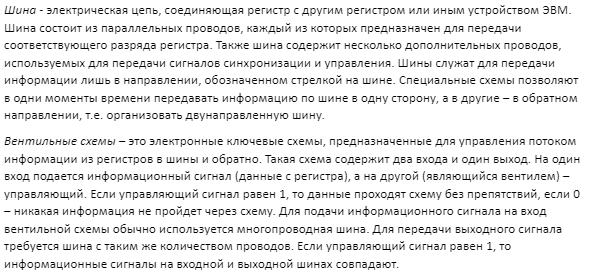
* 1. NUL-terminated – признаком окончания строки является специальный символ (обычно NUL)..
  2. Упаковка с длиной – первое слово строки для хранения длины строки.

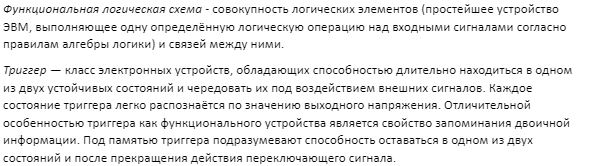
Запись обоих вариантов в память зависит от использования Big-endian(запись в память со старших байтов) и little-endian(с младших)

1. *Базовые элементы вычислительной техники: ячейки, шины, вентили, тактовые генераторы, логические схемы, триггеры, регистры, счетчики, сумматоры.*

Ячейка памяти – минимальный адресуемый элемент запоминающего устройства ЭВМ. Они имеют адрес, по которому к ним могут обращаться команды процессора. Память бывает статическая и динамическая. Динамическая состоит из одного транзистора и одного конденсатора. Из-за невозможности создания идеального конденсатора без потерь энергии требует перезарядки для поддержания функционала и хранения информации. Однако он маленький, простой в своей архитектура и дешевый, именно поэтому остается актуальным по сей день. Статическая представляет собой 6 транзисторов с обратными цепями связи, за счет чего происходит взаимная подпитка энергией и не требует постоянной дозарядки, однако является дорогой, но притом быстродействующей.

Тактовый генератор – «сердце» ЭВМ, задающее ритм исполнения задач. Все пересылки данных, арифметические или логические операции могут происходить только в строго заданное время, определяемое размером такта генератора. К концу такта все операции должны быть завершены.







Сумматор – электрическая схема, производящая суммирование поступивших на вход операндов. Имеет 3 входа (2 операнда и бит переноса из предыдущего разряда) и 2 выхода (результат сложения и бит переноса, который пойдет на сход в следующий сумматор).

Регистр – совокупность нескольких триггеров.

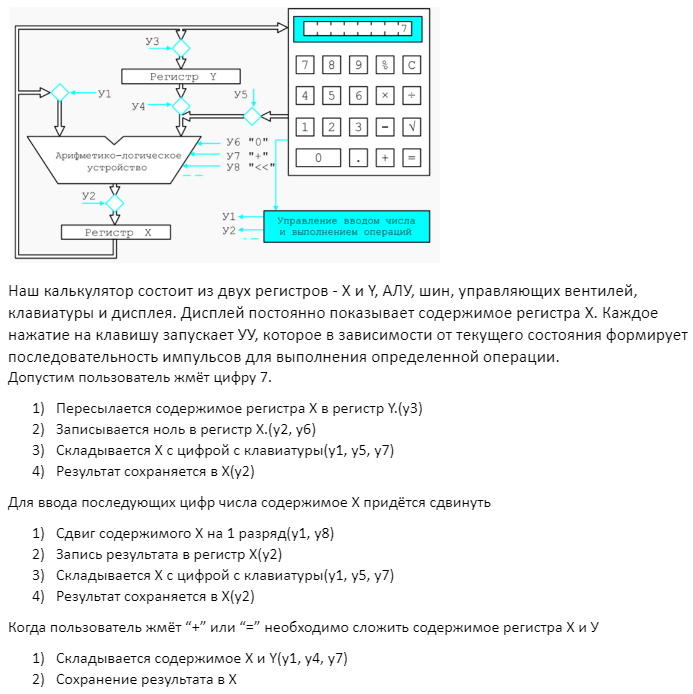
1. *Структура и принцип функционирования ЭВМ. Порядок функционирования простого процессора на примере калькулятора.*

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

Устройство ввода обеспечивает считывание и ее представление в, воспринимаемой другими устройствами ЭВМ. Устройства вывода представляют результаты обработки информации в форме, удобной для визуального восприятия.

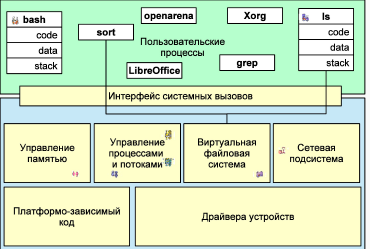
Память ЭВМ включает устройство, обеспечивающее хранение команд и данных. Это устройство состоит из ячеек памяти, предназначенных для хранения одного слова информации. Ячейка памяти состоит из элементов памяти, состояние каждого из которых соответствует одной двоичной цифре. В микро ЭВМ используются безадресные, адресные команды, команды ввода-вывода и ветвления.

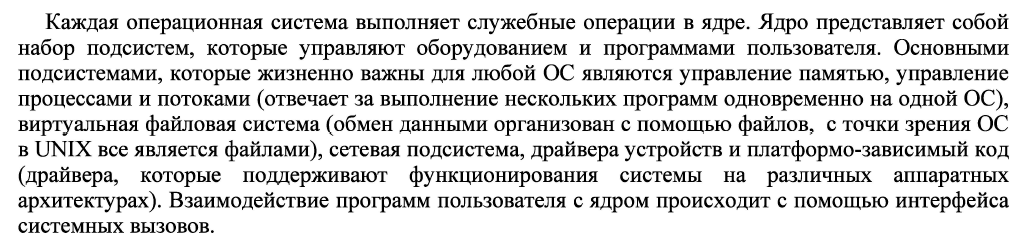
В процессе работы ЭВМ последовательно выполняет набор достаточно простых операций: выборку команды, выборка адреса, выборка операнда, выполнение, прерывание и ввод-вывод.



1. *Операционная система Unix — ядро ОС и файловая система*

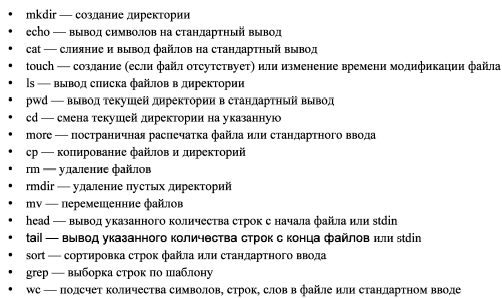
UNIX – семейство ос. Одна из основных ее заслуг – мультиплатформенность. Ядро можно адаптировать почти под любой микропроцессор.





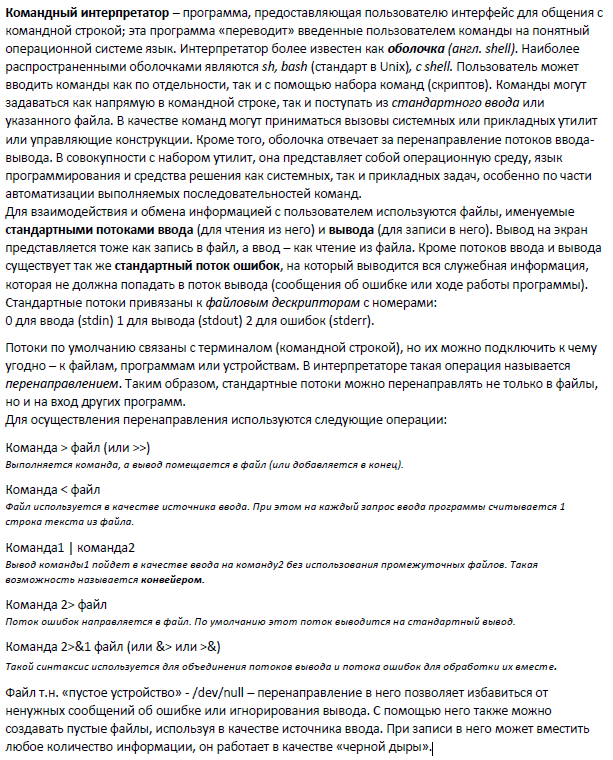
Файловая система UNIX представляет собой совокупность файлов, напоминающих дерево. Все в файловой системе является файлом, в том числе директории и каталоги. При этом каждый файл имеет свой собственный уникальный номер inode, а также путь. Путь может быть абсолютным – содержащим полный путь от корневой директории фс и относительным – начиная от текущего файла рабочего каталога. В каждом каталоге содержатся по умолчанию 2 файла: «.» - ссылка на самого себя и «..» - ссылка на родительский каталог. Ссылки в свою очередь бывают жесткими – они обладают тем же inode, что и исходный файл, существуют вне зависимости от его положения и содержат всю информацию о файле за исключением имени файла и его данных. Символически ссылки имеют свой собственный код и содержат только путь к заданному файлу.

1. *Операционная система Unix — основные команды, права файлов и способы их задания.*



Права определяют доступ к файлам для разных типов участников. С помощью команды ls -la выводится расширенная информация о файлах в том числе скрытых. Она содержит информацию о типах файла, количестве жестких ссылок, владельце, группе владельце, объеме, дате модификации и имени файла. Права делятся на права владельца, группы и остальных, каждый из которых состоит из чтения, записи и исполнения. Исполнение для файла изначает исполнение содержимого, если оно является скриптом, для каталога – переход в него. Права задаются с помощью команды chmod и по умолчанию все файлы имеют права rw-r—r--, если маска задания прав не была изменена. их можно задавать числовым значением (перевод из 8 в 10 сс) или прибавлением(отниманием)/присвоением данному участнику какого либо из прав.

1. *Операционная система Unix — интерпретаторы, стандартные потоки ввода вывода, фильтры.*

**

Вместе с ос Unix поставляются различные утилиты в том числе которые могут осуществлять фильтрацию данных, поступающих на стандартный ввод, результат – на стандартный вывод. Командами-фильтрами являются grep – осуществляет выборку строк совпадающих с шаблоном регулярного выражения и sort – осуществляет сортировку строк по заданному условию.

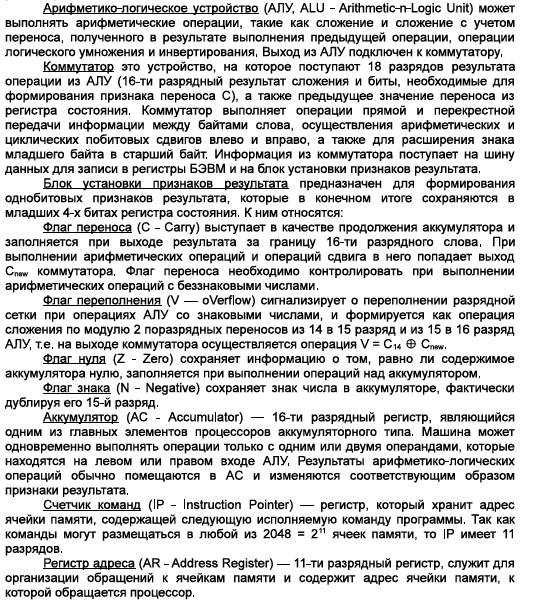
1. *Состав и структура БЭВМ. Адресные пространства БЭВМ. Система команд БЭВМ, форматы команд. Машинные циклы.*

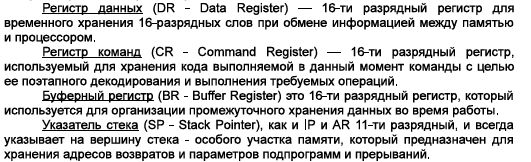
БЭВМ – одноадресная ЭВМ аккумуляторного типа, выполняющая простейшие операци с 16-разраядными машинными словами. Состоит из нескольких функциональных блоков и регистров.

Память – 2048 16-разрядных ячеек с адресами от 000 до 7FF.

Процессор – состоит из ряда регистров, алу с коммутатором и блоком установки признаков результата а также устройства управления

Устройство управления (микропрограммное устройство) – выполняет команды процессора с помощью элементарных микроопераций (открытие вентелей и проверка битов заданного регистра)

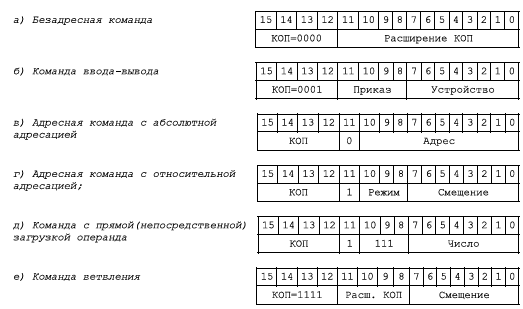








Адресация в эвм делится на 4 типа: адресные команды, безадресные, ввода-вывода и ветвления. Выбор одного из этих типов осуществляется МПУ с помощью анализа старших 4 битов команды (кода операции).

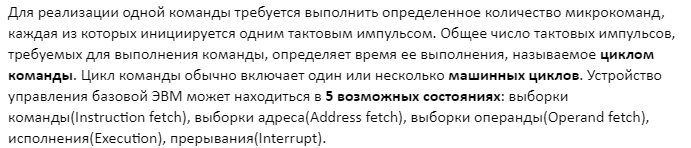


Адресные команды выполняют различные действия ссылаясь на ячейку памяти в зависимости от типа адресации. Исключение – прямая загрузка операнда, при которой обращения к ячейке памяти не производится.

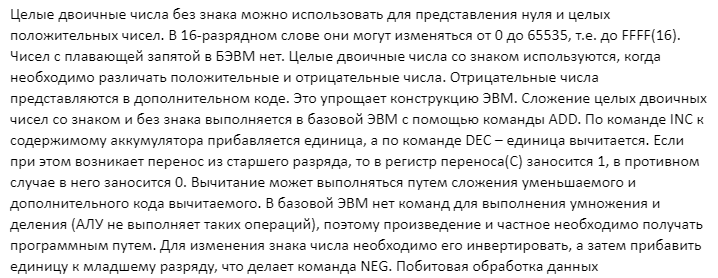
Безадресные команды выполняют действия без ссылок на ячейку памяти.

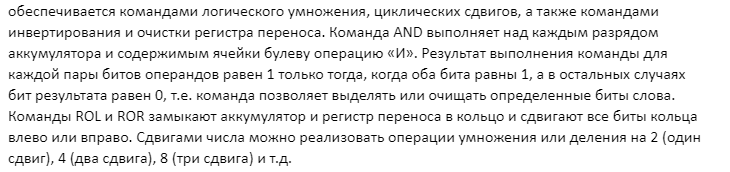
Команды ввода-вывода управляют обменом данными между процессором и внешними устройствами эвм.

Команды ветвления позволяют продолжать вычислительны процесс с дургого адреса программы в зависимости от состояния признаков результата.



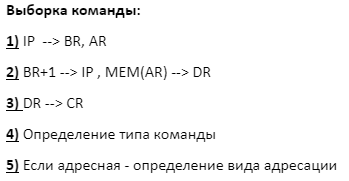
1. *Организация вычислений в БЭВМ. Сдвиги, арифметические и логические операции. Цикл выборки команды.*



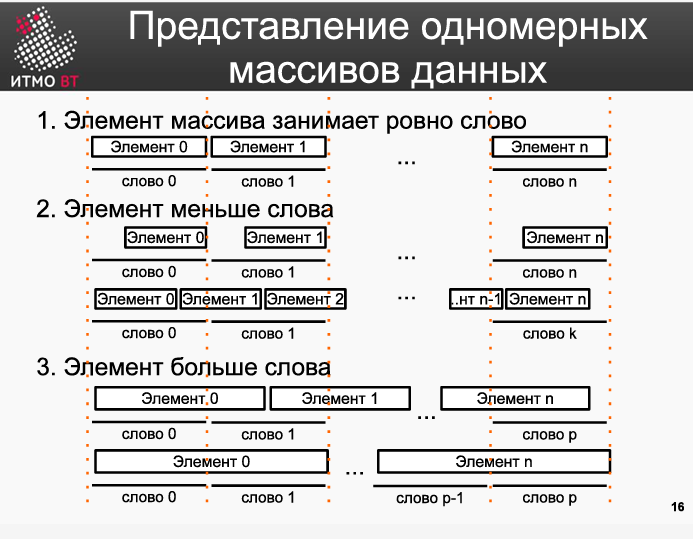


Арифметический сдвиг влево смещает на один бит влево и заносит 0 в младший бит, а высвободившийся бит старшего разряда в признак результата С.

Сдвиг влево : АС0 -> C, AC15->AC14.

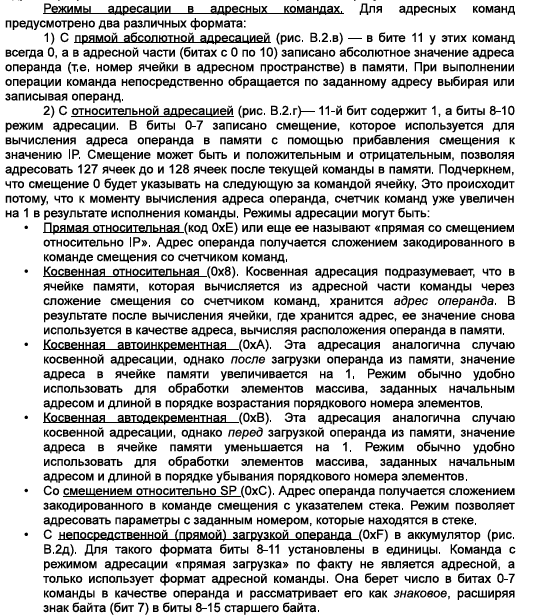


1. *Организация массивов данных. Режимы адресации. Цикл выборки адреса и операнда БЭВМ.*



В БЭВМ работа с массивами организована с помощью режимов адресации. Обычно хранится ячейка, содержащая начало массива данных, указатель на следующий элемент массива, размер массива и счетчик цикла LOOP.



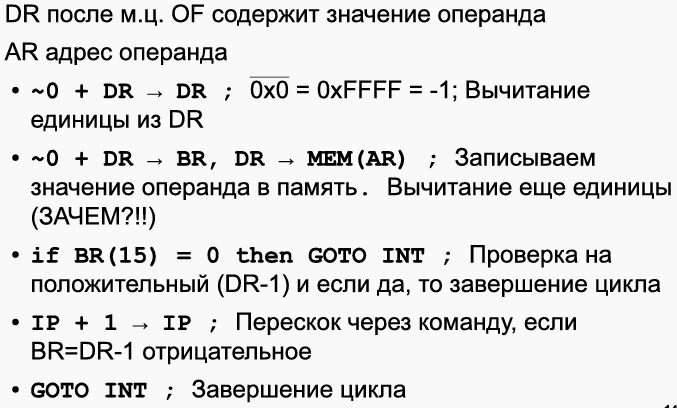
**

1. *Управление вычислительным процессом в БЭВМ. Команды ветвлений, цикл исполнения команды LOOP*

Управление вычислительным процессом в БЭВМ организовано с помощью команд перехода. Они имеют код F0XX-F9XX, CEXX – безусловный переход, где ХХ – смещение перехода относительно IP. Они работают с помощью анализа флагов операции. Для сравнения двух чисел перед командами перехода используется команда CMP, которая сравнивает число А из AC и B из аргумента CMP. Она вычитает из А В и выставляет флаги по результату, не изменяя содержимое аккумулятора. Примером команд перехода являются BEQ (Z=1, числа равны), BMI (N=1, В>A) и многие другие.

С помощью команды LOOP организуется цикл. Она декрементирует значение ячейки, содержащейся в аргументе, сравнивает ее с 0, и, если ее значение <=0, перепрыгивает следующую за LOOP команду.

Цикл исполнения LOOP:



1. *Подпрограммы в БЭВМ. Цикл исполнения команд перехода и возврата из подпрограммы. Стек, передача параметров. Позиционно-независимый код. Загрузчик и библиотеки.*

Зачастую в процессе исполнения кода необходимо производить одинаковые действия, так называемую функцию, которая хорошо выделяется в кусок кода, который впоследствии можно вызывать из основной программы. Такой кусок кода в Ассемблере называется подпрограммой. Выгодно использовать подпрограмму, когда количество команд подпрограммы вместе со служебными командами вызова и возврата будeт меньше, чем если бы код подпрограммы мы инлайнили. Осуществление вызова подпрограмм вызывается с помощью команды CALL, а возврата – RET.

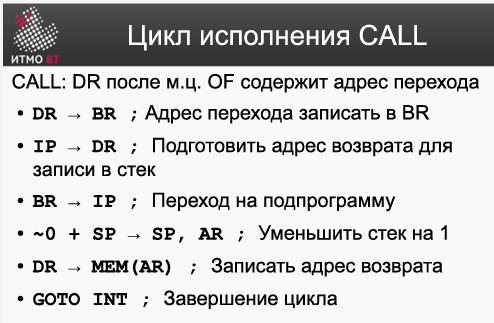
При вызове подпрограммы в структуру данных под названием стек записывается адрес возврата, по которому после исполнения подпрограммы мы вернемся в основную. Стек обычно ассоциируют со стопкой книг или блинов, где мы можем либо положить сверху, либо взять верхнее. В БЭВМ стек находится также в основной памяти, по дефолту указывает на ячейку 000, и растет вниз (декрементом), те следующая ячейка после 000 будет 7FF и так далее.

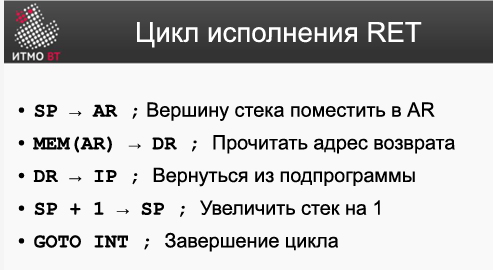
Простейшим способом передачи данных в подпрограмму и сохранение/использование результата является через аккумулятор, однако минусом такого способа является ограничение количества параметров, так как количество регистров общего назначения ограниченное количество. Также можно использовать ячейки памяти, но в таком случае нужно эту логику организовывать. Существует передача параметров с помощью регистровых окон (кольцо, разбитое на сегменты входных, выходных и локальных параметров). Намного более частым и удобным способом передачи параметров является использование стека. Стоит отметить, что адрес возврата всегда должен лежать сверху, поэтому загрузка параметров должна производиться перед вызовом подпрограммы с помощью команды PUSH. Перемещение содержимого ячеек в аккумулятор происходит с помощью команды POP (из стека удаляется, перемещается в АС).

Позиционно-независимый код – код, который не изменяет своей работы при перемещении ее в разных областях памяти. Относительная адресация должна использоваться внутри программы, все внешние обращения – абсолютные.

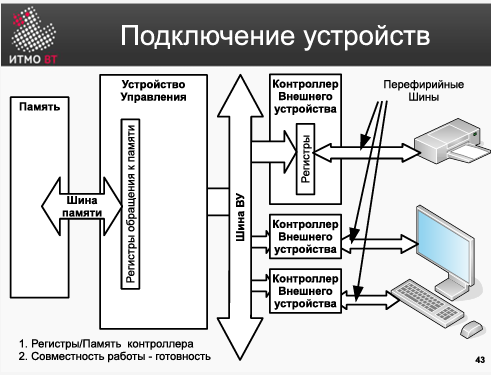
В реальности подпрограммы объединяются в библиотеки. Существует специальная часть операционной системы под названием загрузчик и динамический линковщик программ. В программе, когда потребовалась какая-то функция, динамический линковщик грузит в память библиотеку с необходимой функцией, связывает программу и библиотечную функцию, устанавливая адреса переходов.

Библиотеки делятся на разделяемые (динамически линкуемые, по требованию загружается код библиотеки, в процессе исполнения) и архивные (статически линкуемые, автоматически линкуется при компиляции, в явном виде загружается)





1. *Организация ввода-вывода в вычислительных системах. Инициализация обмена, передачи информации и завершение обмена. Драйверы.*



К шине периферийных устройств любого процессора подключаются контроллеры, которые управляют внешними устройствами, которые к ним подключены. Нельзя напрямую подключить ВУ к процессору. Технически возможно, но из-за разницы скоростей при управлении программным путем, большая часть времени процессор будет ожидать готовности подключенного к нему ВУ и не сможет делать ничего другого, те ресурсы будут тратиться нецелесообразно. Эта проблема решена контроллером (что-то типа локального процессора, который управляет обменом с ВУ, хранит данные, статусную информацию в своих регистрах).

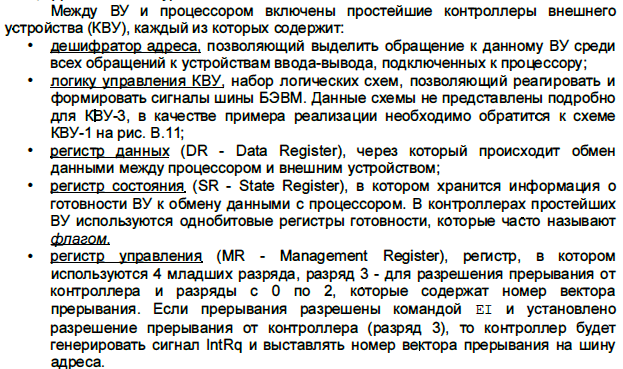
Внутри ОС помимо контроллера нужен драйвер, который организует данные, которые поступают с ВУ, и преобразует в вид, понятный ОС. Они организуют совместную работу с устройством, поскольку «знают» о принципах устройства и работы как ОС, так и контроллера ВУ.

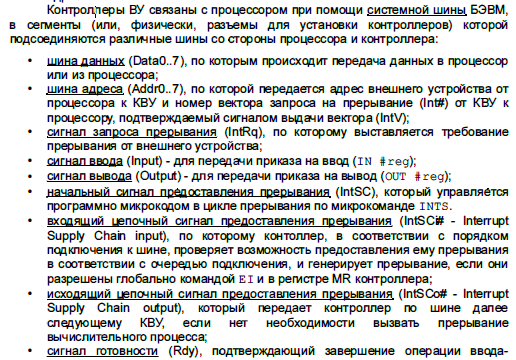
Ввод-вывод разделяется на программно-управляемый (обмен под управлением программы на процессоре через его регистры) и управляемый аппаратурой (прямой доступ к памяти, управляющее устройство – контроллер, берет на себя функцию инициализации обмена, минуя процессор, взаимодействует с памятью).

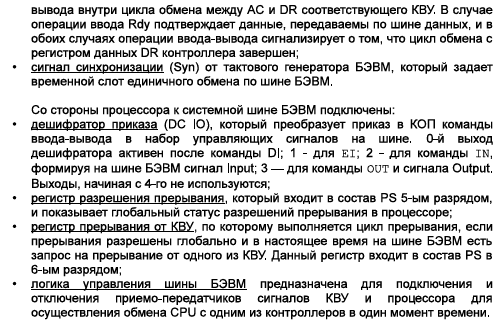
При программно-управляемом обмене инициализация обмена бывает синхронной (в заданное время), асинхронной (проверка состояния готовности) и по прерыванию (при инициализации обмена приостанавливается выполнение текущей программы, производится ввод-вывод и возвращается к выполнению программы, которая исполнялась до этого). Обмен данными (прием и передача) может также быть организован синхронно, когда наличие данных на шине подтверждается специальным сигналом синхронизации с постоянной частотой, и асинхронно, с использованием сигналов готовности и/или подтверждения приема-передачи данных. Завершение обмена – синхронное (в ответ на системный вызов приходит сигнал окончания обмена синхронно с ним) и асинхронное (после системного вызова через какое-то время приходит сигнал окончания обмена).

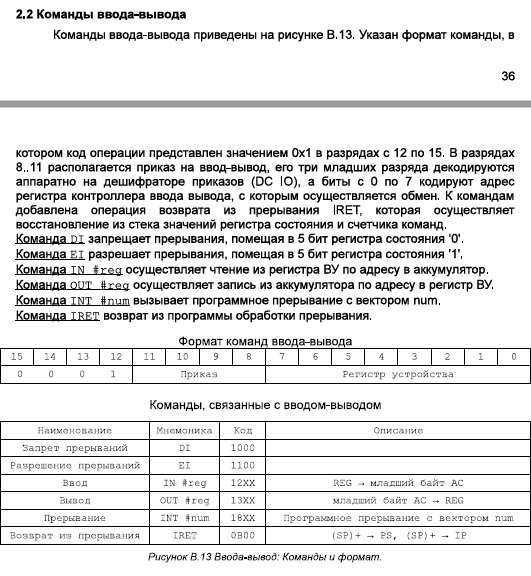
1. *Организация ввода-вывода в БЭВМ. Устройства ввода-вывода, команды*

В БЭВМ реализована только программно-управляемая передача данных, т.е. только под управлением БЭВМ. Существует 10 ВУ: таймер, устройство вывода, устройство ввода, 2 устройства ввода-вывода, текстовый принтер, бегущая строка, 7-ми сегментный индикатор, клавиатура, цифровая клавиатура.





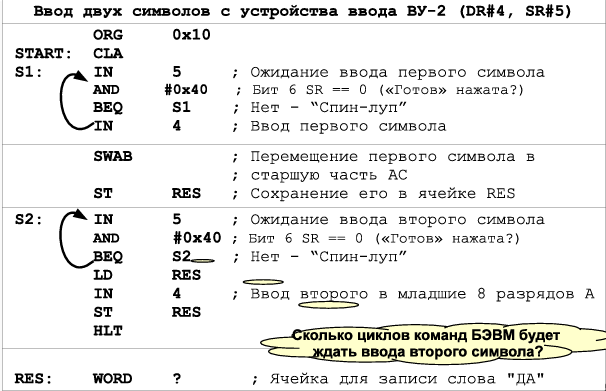


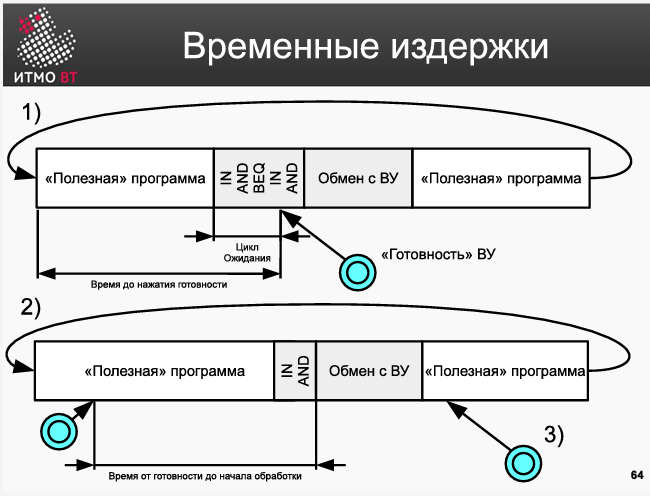




1. *Организация асинхронного обмена в БЭВМ. Пример программы. Временные издержки*

При асинхронном обмене: происходит спин-луп с ожиданием готовности ВУ (6 бит регистра состояния ВУ), как только ВУ готово, дается команда на обмен, после выполнения которого готовность обнуляется (бит готовности). Во время ожидания готовности загрузка ЦПУ 100%, тк происходит постоянная проверка и ожидание, хотя полезной нагрузки 0.

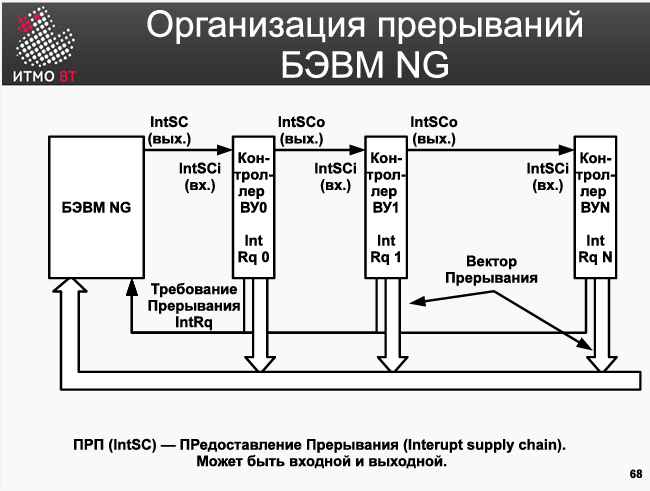




1. *Организация прерываний в БЭВМ. Вектора прерываний, контроллер прерывания.*

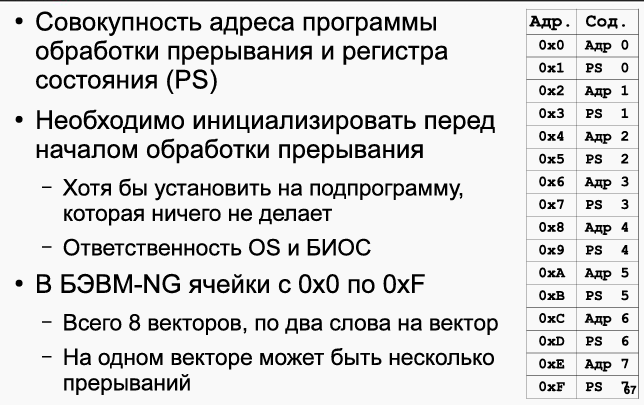
В процессе исполнения основной программы готовность ВУ вызывает программно-аппаратный процесс, который после цикла исполнения команды переходит на цикл прерывания, останавливает текущую программу и переходит на специальную подпрограмму обработки прерываний, произойдет ввод-вывод и возврат на исполняемую до этого полезную программу.

Подпрограмма обработки прерываний запрещает прерывания, сохраняет текущее состояние процессора (в БЭВМ минимально АС и регистр состояния PS). После обмена по этому сохраненному состоянию будет произведено восстановление состояния, разрешение прерываний и продолжение исполнения программы с момента на котором мы остановились до инициализации прерывания.



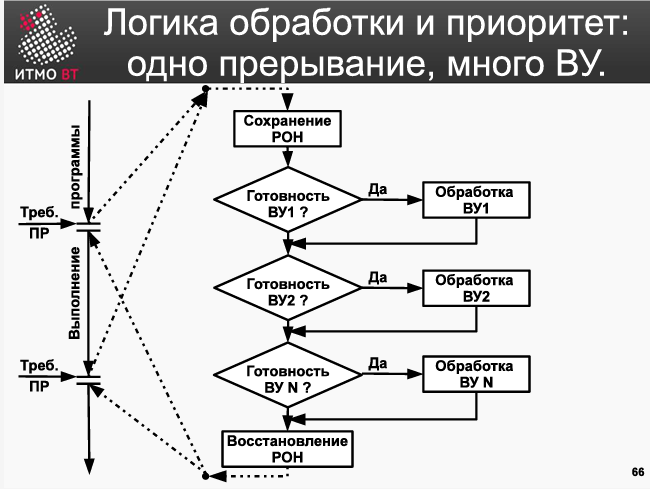
Более детально: ВУ выставляет на свою шину сигнал требования прерывания, который сохраняется на процессоре в регистре состояния, пока не начнется цикл обработки прерывания. В цикле обработки прерываний проверяются некоторые входные параметры и как раз есть ли сигнал IntRq (бит в PS), после чего БЭВМ формирует управляющий сигнал предоставления прерываний (IntSC). Он проходит цепочкой насквозь всех ВУ, между входом и выходом проверяет был ли запрос прерывания у каждого ВУ (если был маскирует сигнал, не пропуская дальше и выставляет на шину адреса номер вектора своего прерывания). Будет происходить до тех пор, пока не будут обработаны все прерывания, которые установлены в контроллерах ВУ.

Вектор прерывания – совокупность адреса программы обработки прерывания и его регистра состояния, т.е. 2 ячейки. В БЭВМ максимально 8 типов прерываний, то есть 8 векторов, лежит от 000 до 00F включительно. Чтобы правильно инициализировать вектора прерываний, все адреса в ячейках обработки прерываний всех векторов необходимо установить на подпрограмму, которая хотя бы ничего не делает. На одном векторе может быть несколько прерываний.

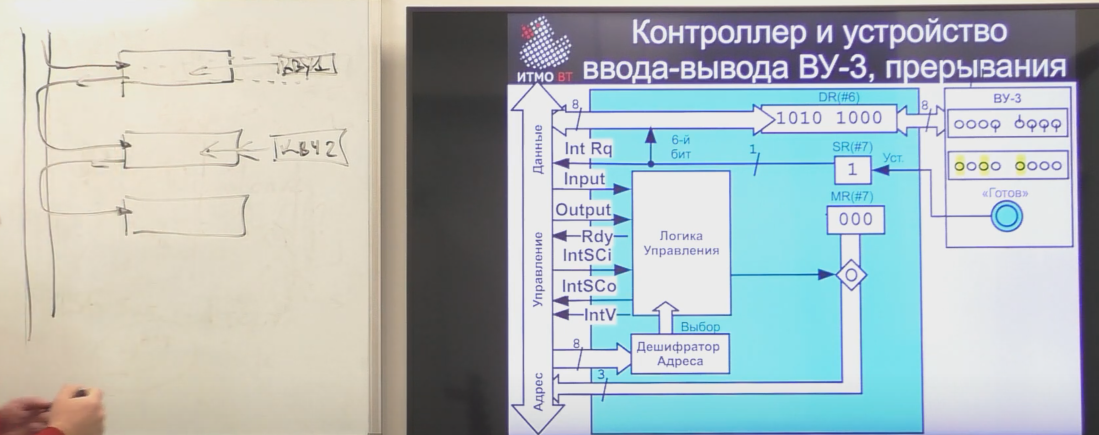


Контроллер прерываний контролирует вложенность прерываний и возможность их исполнения, то есть проверяет является ли новое прерывание более приоритетным, чем то, которое обрабатывается сейчас и вложенность, так как она ограничена.

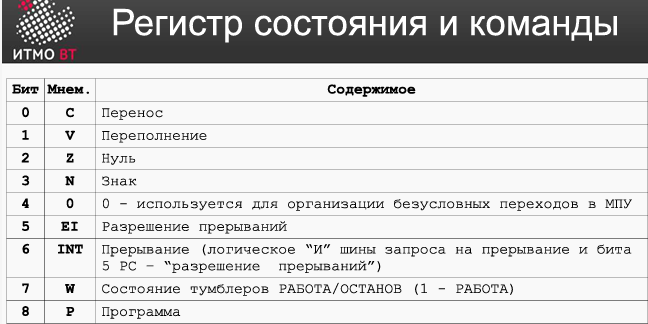
1. *Организация обмена по прерыванию программы в БЭВМ. Пример программы. Цикл прерывания.*



В ВУ наряду с сигналами обмена есть сигналы запроса прерываний (из ВУ), предоставления прерываний (входящий сигнал в логику управления ВУ, исходящий – из для дальнейшего его пути к другому ВУ, если текущий не выставлял запрос прерывания, который определяется как 1 в регистре состояни ВУ SR).

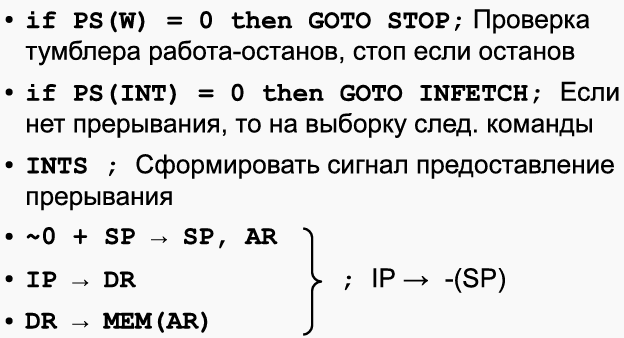


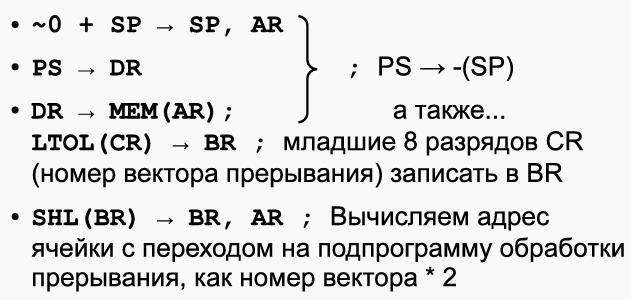
IntV – сигнал, который подтверждает запись вектора прерываний. Номер вектора прерывания из ВУ будет приходить в 3 младших разряда регистра команд по шине адреса (из регистра управления контроллера MR).

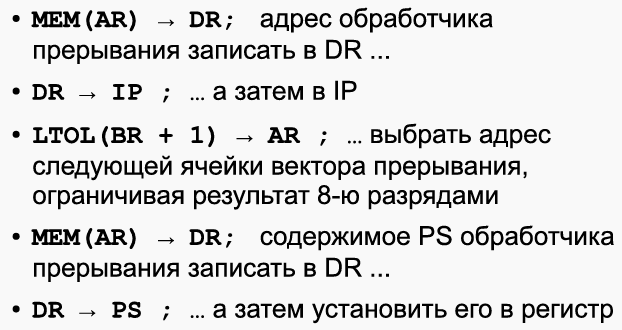


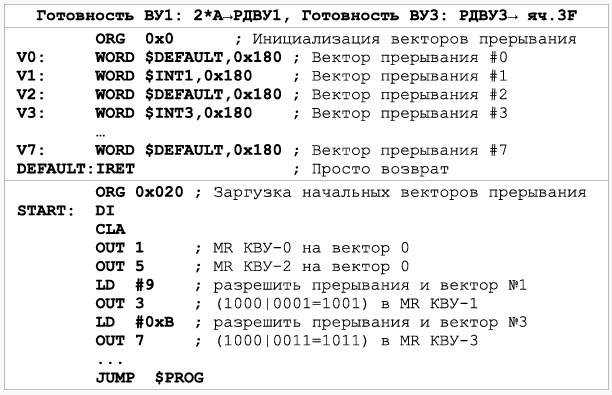
Запуск процесса прерывания реализуется как раз анализом 6 бита в PS. То есть если у нас разрешены прерывания и есть запрос прерывания от ВУ, происходит переход на цикл прерывания.

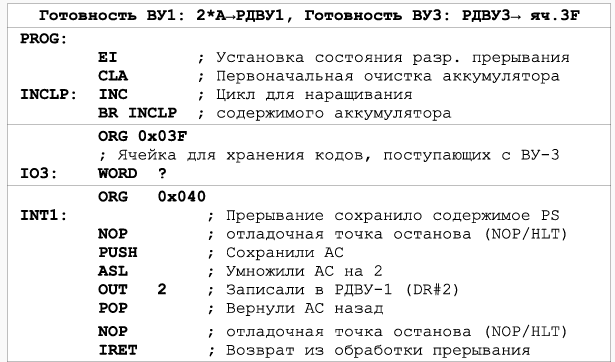
Цикл прерывания:

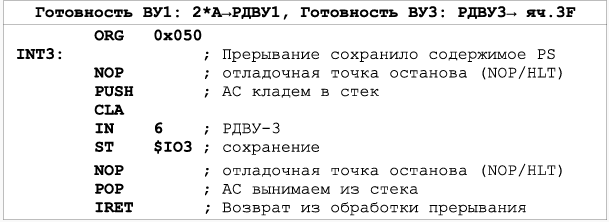




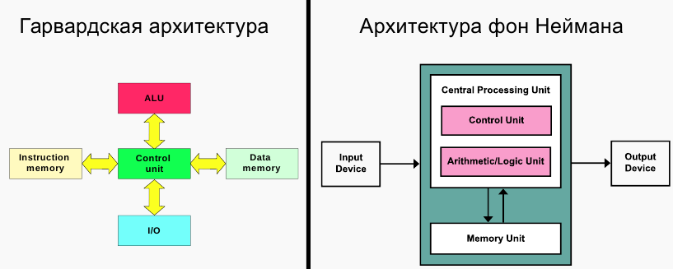


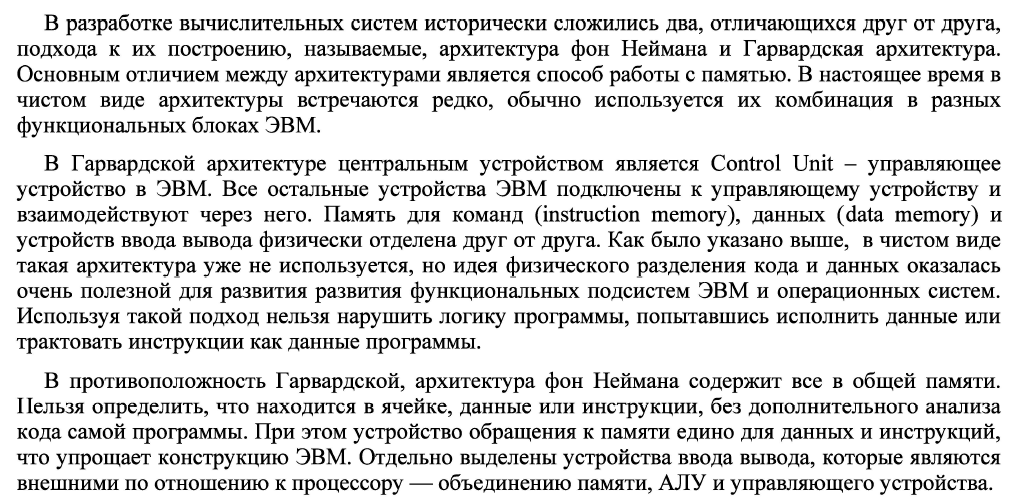
Пример программы:  






1. *Понятие многоуровневой ЭВМ. Понятие и пример программы на разных уровнях.*
2. *Многопрограммный уровень БЭВМ. Структура МПУ. Форматы микрокоманд.*
3. *Архитектура ЭВМ. Гарвардская и фон-Неймановская архитектура. Организация обмена архитектуры ЭВМ с использованием шин.*

**

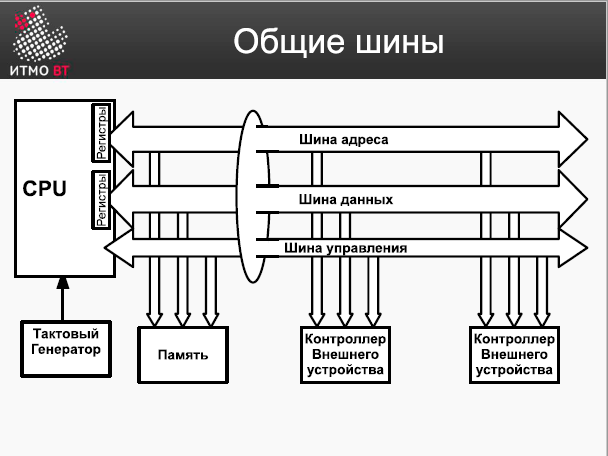




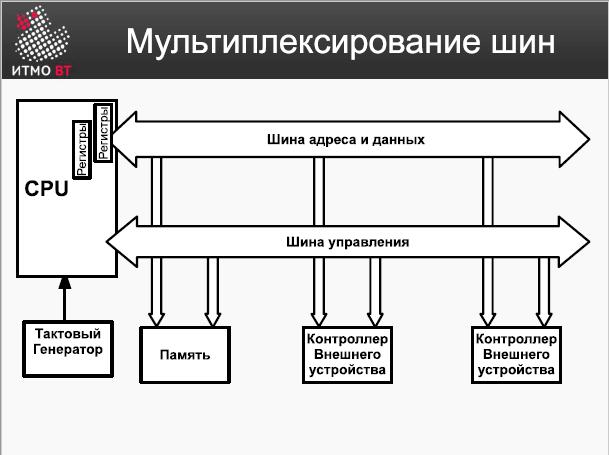
Архитектура строилась вокруг памяти, она была центральным звеном. Скорость всех блоков вычислительной машины была примерно одинакова. С внешними устройствами работали отдельные процессоры. Блоки связаны шинами, работают независимо друг от друга.



Через какое-то время производительность процессора стала больше чем памяти, поэтому процессор занял центральную позицию. Он управлял памятью под тремя различными шинами. Скорость процессора и памяти повысилась, а контроллер внешних устройств остался примерно таким же.



Отличием между 1,2 и третьим поколением является задача увеличения количества элементов и производительности на единицу объема. Для упрощения конструкции на одной шине стали располагаться все устройства кроме процессора.



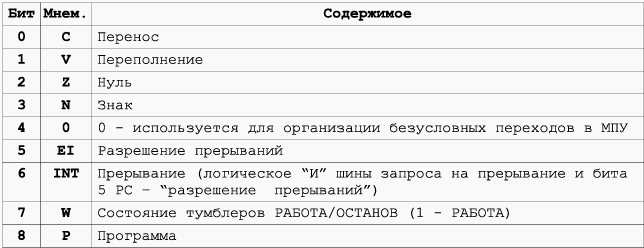
Для сокращения количества проводов, которые соединяли блоки вычислительной машины, адрес и данные стали передавать по одним и тем же физическим проводам. В один импульс тактового генератора передавался адрес, в другой – данные.

1. *Микропрограммное управление вентильными схемами. Схема управления. Интерпретатор в БЭВМ.*
2. *Структура и принципы работы арифметико-логического устройства и коммутатора. Регистр состояния БЭВМ*

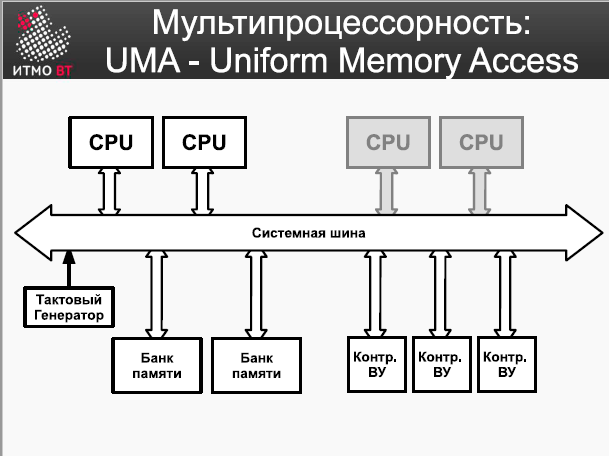
У АЛУ есть два входа: левый и правый. К правому подключены регистры SP, IP, CR, DR, к левому -– AC, BR, SP, клавишный регистры. На каждом входе находятся инверторы, внутри есть схема сумматора и схема логического И, реализована операция инкремента и обратного кода.

Из АЛУ на коммутатор поступают 18 разрядов результата (16 бит результата и биты, необходимые для формирования признака переноса С), а также предыдущее значение переноса из регистра состояния. Коммутатор, в свою очередь, выполняет сдвиги, операции расширение знака, а также операции прямой и перекрестной передачи информации между байтами. Информация из коммутатора поступает на шину данных записи в регистры ЮЭВМ и на блок установки признаков результата.

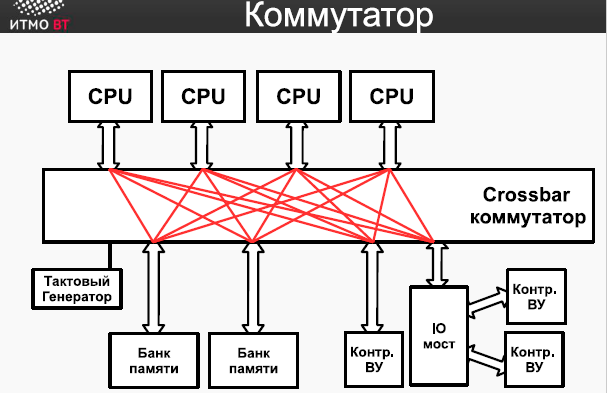




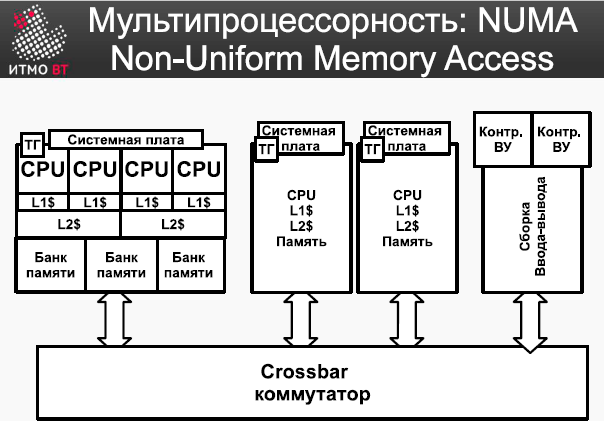
1. *Архитектура многопроцессорных ЭВМ. Системный коммутатор. Архитектуры UMA и NUMA.*



UMA (Uniform Memory Access) – архитектура современных машин. Технически UMА-системы предполагают наличие узла, соединяющего каждый из n процессоров с каждым из m модулей памяти. Простейший путь построения таких ВС - объединение нескольких процессоров с единой памятью посредством общей шины. В этом случае, однако, в каждый момент времени обмен по шине может вести только один из процессоров, то есть процессоры должны соперничать за доступ к шине. При использовании системной шины с несколькими процессорами возникают проблемы взаимодействия каждого из них через шину с устройствами (возникла нехватка пропускной способности). Само использование нескольких процессоров обеспечивает возможность параллельной организации многих потоков команд и данных.

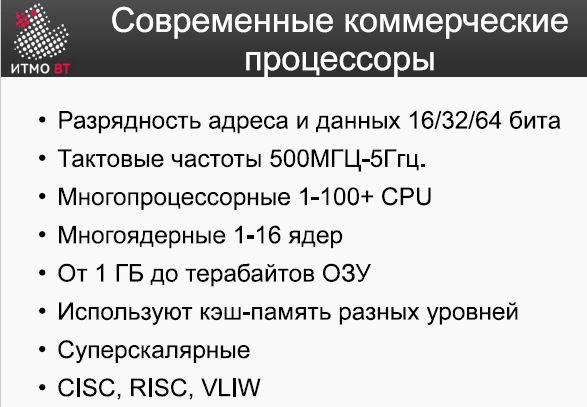


Эту проблему решили с помощью коммутатора. Он представляет собой несколько системных шин, каждая из которых подключена к отдельному процессору и остальным устройствам. Коммутатор в состоянии параллельно обслуживать несколько запросов. Каждый процессор может быть соединен со своим модулем памяти и иметь доступ к нему на максимально допустимой скорости. Соперничество между процессорами может возникнуть при попытке одновременного доступа к одному и тому же банку памяти. В этом случае доступ получает только один процессор, а прочие - блокируются.



NUMA (Non-Uniform Memory Access) – предназначена для повышения количества процессоров еще больше (даже сотни). Каждый процессор организован с локальной памятью, локальным кэшем, своим тактовым генератором в виде единой сборки, платы. Все платы связаны друг с другом через коммутатор. Все процессоры работают под одной операционной системой, и вся память всех плат является общей, те при необходимости процессор может использовать память и других системных плат, однако скорость доступа к другим участкам памяти в отличие от локальной будет меньше. ОС заботится о том, чтобы данные сохраняли принцип локальности. Плюсами такой архитектуры является возможность работы процессоров на разной тактовой частоте, тк каждый процессор имеет свой тактовый генератор, а также возможность на горячую менять компоненты вычислительной системы.

1. *Структура современных процессоров. Окружение процессора. CISC, RISC, VLIW.*

**

Суперскалярность – исполнение нескольких команд процессором за один такт генератора

Архитектура процессора в зависимости от способа набора команд (способа его задания)

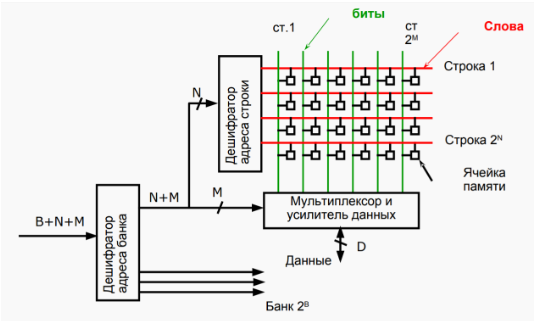


CISC обладал огромным количеством программ, одни отмирали и были не нужны, другие хранились с целью совместимости, в общей массе было невероятное количество хранимых команд.

RISC команды загрузки данных и сохранения в памяти + остальные команды выполнялись между регистрами.

VLIW позволяет одной сложной командой выполнять несколько операций. Процессор за один такт выполняет команду.

1. *Адресуемая память, организация и временные диаграммы. Конструктивные особенности современной памяти.*



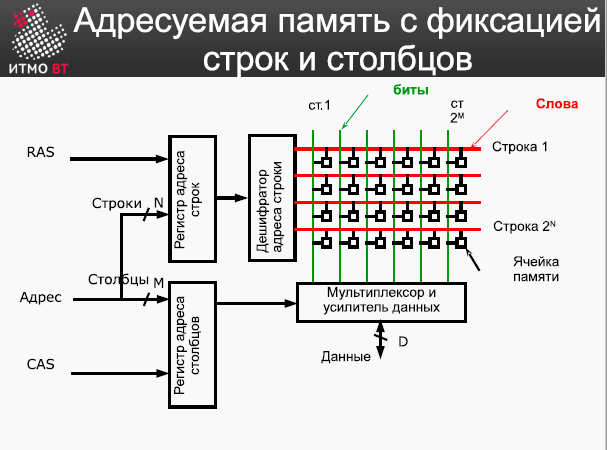
Организация адресуемой памяти в реальности отличается от той, которая представлена в бэвм (дешифратор, который обращается к заданной линии). В реальности существует мультиплексор – устройство, которое осуществляет передачу сигнала с входа неизменным на выход. Входов много, а выходов один. Память разбита на банки, а не является единым пространством. Дешифратор выбирает адрес банка, а каждый банк состоит как раз из дешифратора адреса строки и мультиплексора. Такая структура существенно проще по сравнению с использованием огромного единого дешифратора.



Цикл считывания и записи разные, разные управляющие сигналы, которые подаются в разные моменты времени.

Цикл считывания: при поступлении сигнала для чтения до момента получения данных из памяти пройдет какое-то время (время доступа), за которое произойдет обращение сначала к дешифратору банка, потом адреса и мультиплексора, и только после этого данные будут считаны. Одновременно с данными обычно поступает сигнал готовности. После этого процессор читает данные и снимает с шин адрес для чтения и сигнал для чтения, вследствие чего считываемое слово пропадет с шины данных. Наступает время восстановления.

Цикл записи: выставляется на шину адрес записи, на шину управления сигнал записи, на шину данных выставляется записываемое слово. Срабатывают дешифраторы, внутри памяти формируется импульс записи (запись в выбранную ячейку). Формируется сигнал готовности, с шин снимаются сигналы и данные, наступает время восстановления.

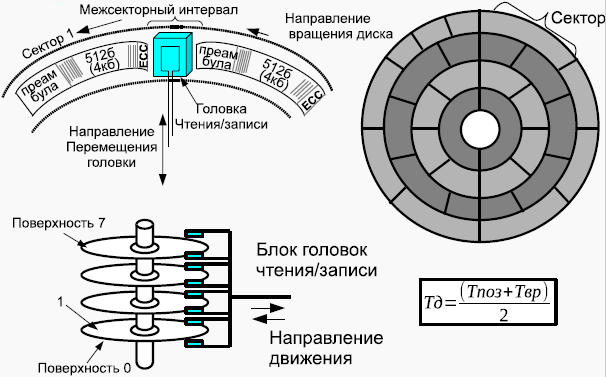
**

Структура, представленная ранее, оказалась сложной для физической реализации. Было решено разбить адрес на условно адрес строки и адрес столбца и передавать их последовательно по шине. Схема с банком не изменилась, добавились 2 доп регистра и 2 сигнала: RAS – выбор строки и CAS – выбор столбца.

Конструктивные особенности современной памяти

* Burst mode – после первой подачи адреса каждое следующее слово выдается последовательно из памяти. Данные выдаются пакетами соотвествующими строке кэша.
* Double Data Rate – передача данных по фронту и спаду импульса
* SPD – чип, содержащий идентификационную информацию о памяти. По нему процессор понимает совместимость и проводит подключение к памяти.
* Interleaving – расслоение памяти, повышает производительность. Несколько димов памяти располагаются не параллельно, а последовательно, засчет чего увеличивается возможное машинное слово
* DDR4-2133 8192MB **PC4-17000** – индекс производительности памяти

1. *Память, ориентированная на записи (блочная память). Организация дисковой памяти и памяти на магнитных лентах*

**

Жесткие диски. Состоит из блинов – поверхность, на которой записана информация и головок – читают информацию с намагниченной поверхности. Когда она идет по диску (диск разбит на секторы, содержащие преамбулы говорящие о начале сектора, информацию (512 байт/сектор) и поле для контроля четности).

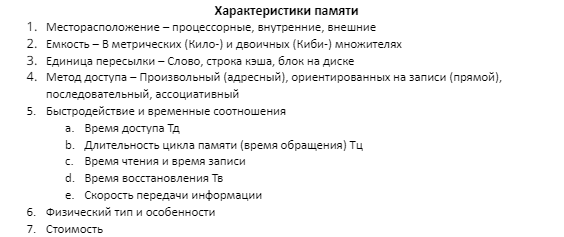


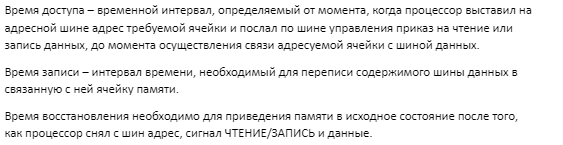
Ленточные накопители обладают дешевой стоимостью хранения информации. За счет поворота головки данные записываются на ленту под углом, что увеличивает скорость передачи данных. Затруднена работа посекторно, поэтому информацию располагают сначала, последовательно и без промежутков.



.

1. *Характеристики запоминающих устройств. Пирамида памяти.*

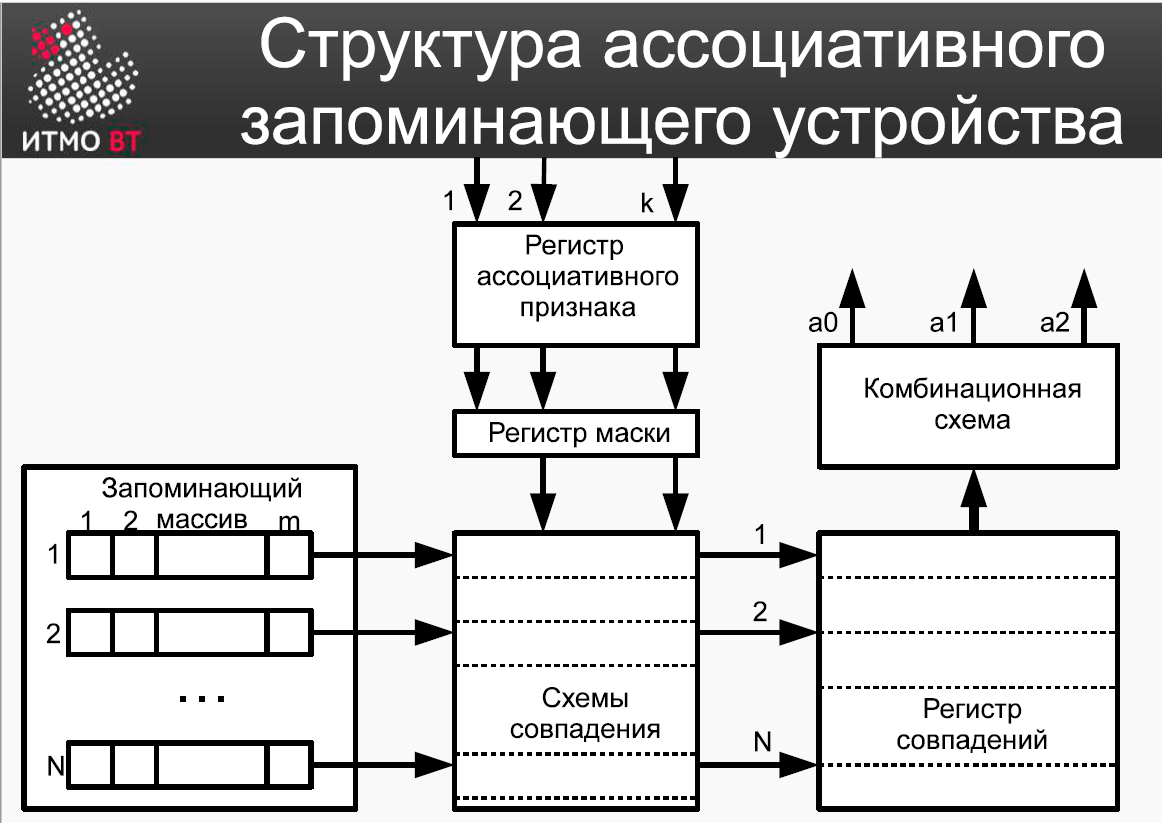








1. *Ассоциативная память, Кэш-память. Влияние промахов кэш-памяти на производительность*



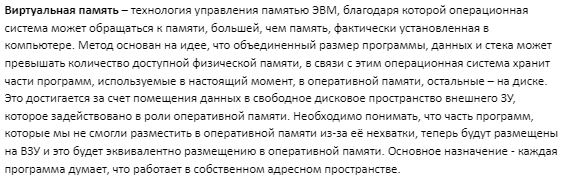
Выбирает не информацию по адресу, а по заданному признаку. Каждая ячейка внутри имеет схему сравнения. Компаратор – сравнивающее устройство. Помимо привычных регистров содержит регистры ассоциативного признака, маски, совпадений.

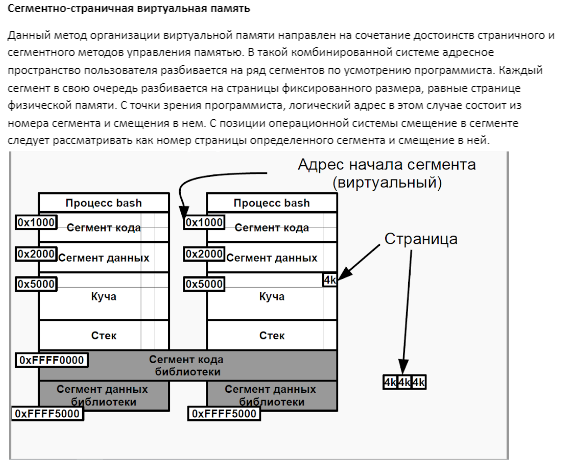
**

Адрес разбивается на несколько частей: тег и адрес. По тегу выбирается содержимое из памяти.

Кэш-промах – когда приложение запрашивает данные, которых в кэше нет, так что для помещения их в кэш нужно обратиться к диску. Чем меньше кэш промахов, тем лучше функционирование системы и производительности. К примеру, при 98% попадания в кэш производительность падает до 60% по сравнению со 100% попаданий.

1. *Предназначение и организация виртуальной памяти. Сегментно-страничная организация. Устройство управления памятью (MMU), буфер трансляции (TLB).*



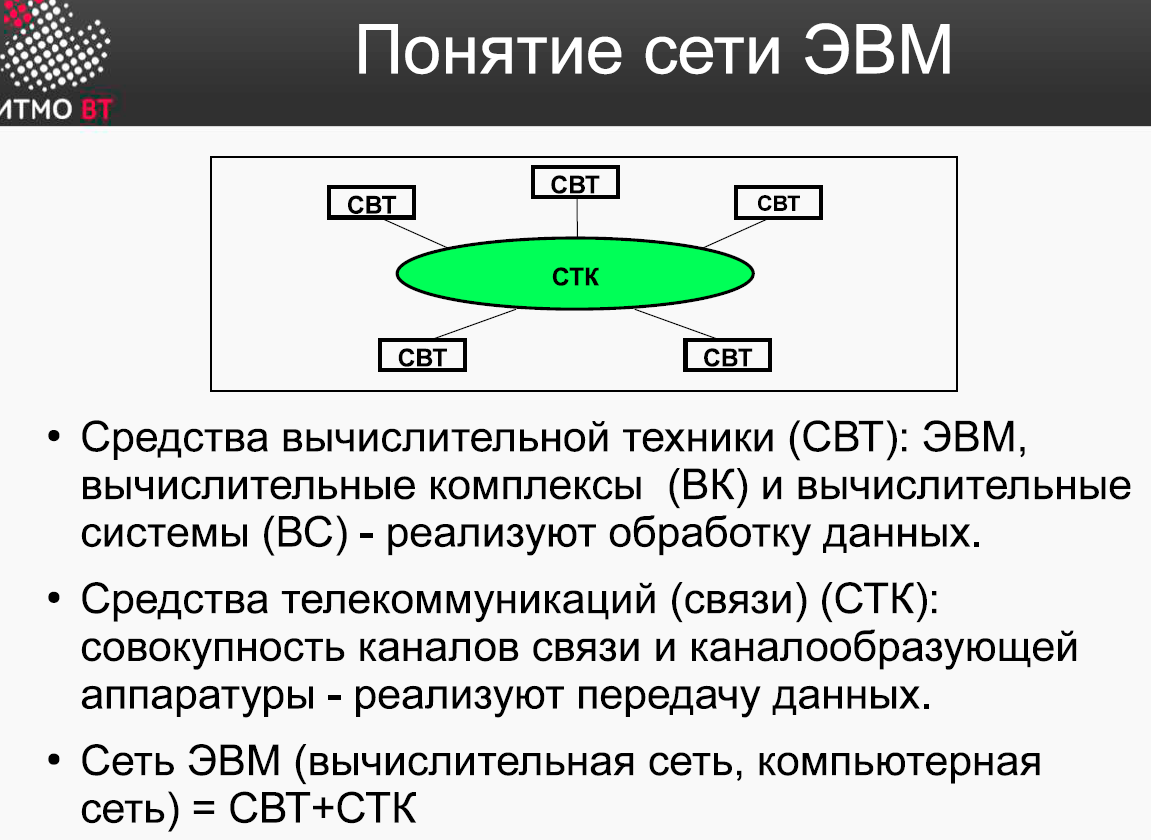


MMU – устройство, на вход поступает виртуальный адрес, есть специальная таблица преобразований, которая говорит есть ли данная страница в виртуальной памяти и какой у нее адрес физической памяти. Манипулирование таблицей осуществляет ОС.

TLB – устройство. Туда записываются виртуальные адреса и соответствующие им физические, те кэширует готовый мапинг для часто используемых программ. Организован в виде ассоциативной памяти.

1. [*Сетевые технологии, Понятие сети ЭВМ, классификация компьютерных сетей. Сообщение и пакет. Модель взаимодействия открытых систем.*](https://docs.google.com/document/d/16KdfSF0s1K1T25zAv2ozuWFhWl1p_-vdHZjAxsL59qI/edit#heading=h.7psfen6e7np3)

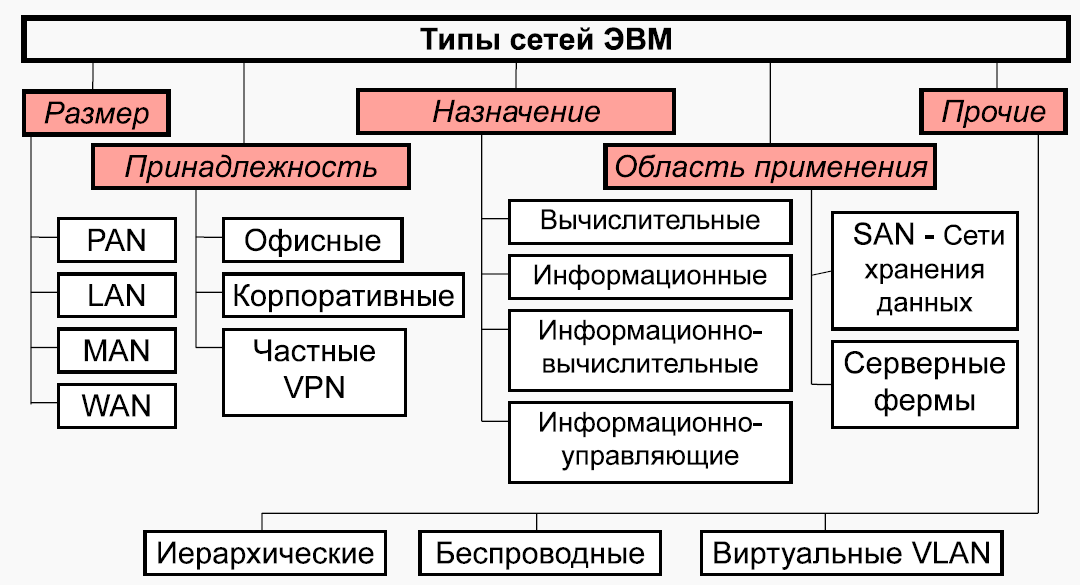
Сетевая технология – согласованный набор стандартных протоколов и реализующих их программно-аппаратных средств, достаточных для построения сетей эвм.



Вычислительный комплекс – несколько ЭВМ, объединенных одной задачей.

Вычислительные системы – вычислительный комплекс + прикладное программное обеспечение.

Канал связи – физический провод (к примеру между зданиями)  
Каналообразующая аппаратура – модемы, вызывающие передачу информации.

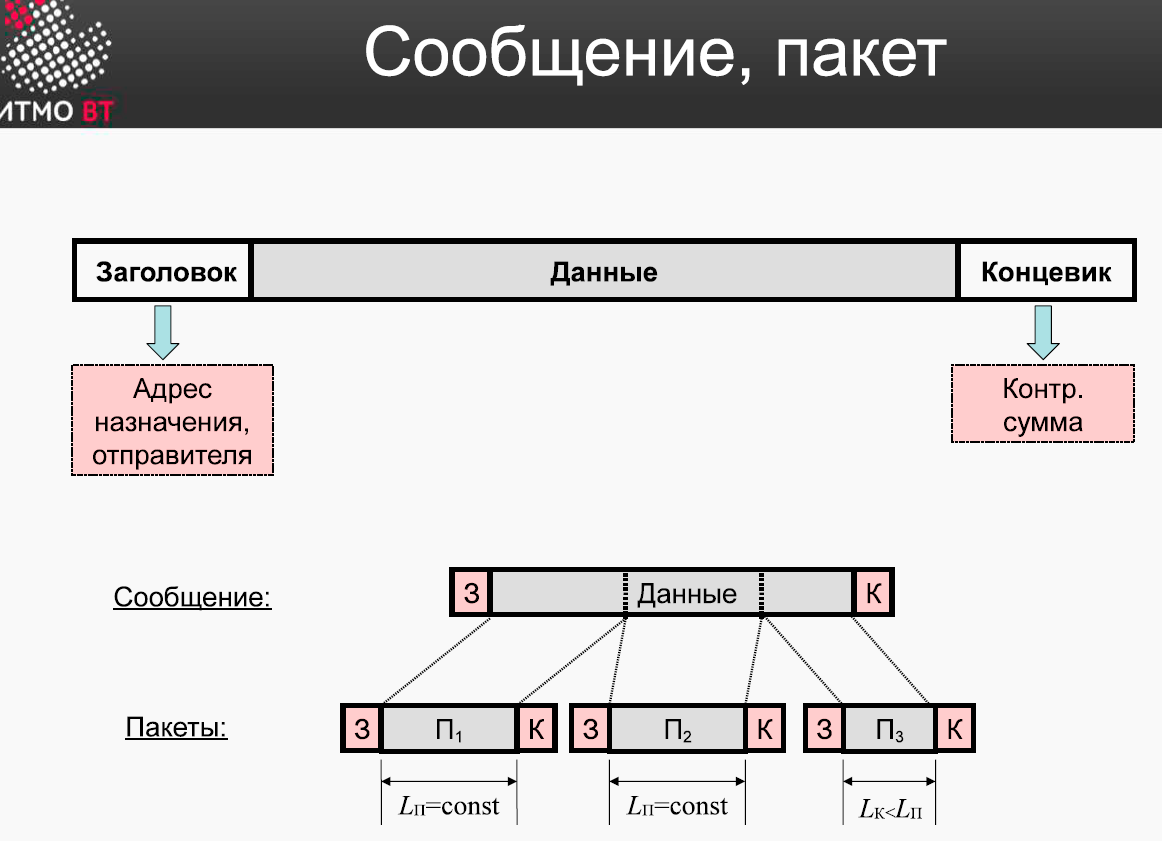
**

PAN – персональная (блюпуп)

LAN – локальная сетка (в пределах комнаты, офиса, квартиры)

MAN (мегаполис) – гибрид между LAN и WAN, применяются обе технологии одновременно

WAN – глобальная (города, страны. Имеет меньшую скорость передачи данных, производится от точки к точке по каналам с большой пропускной способностью)

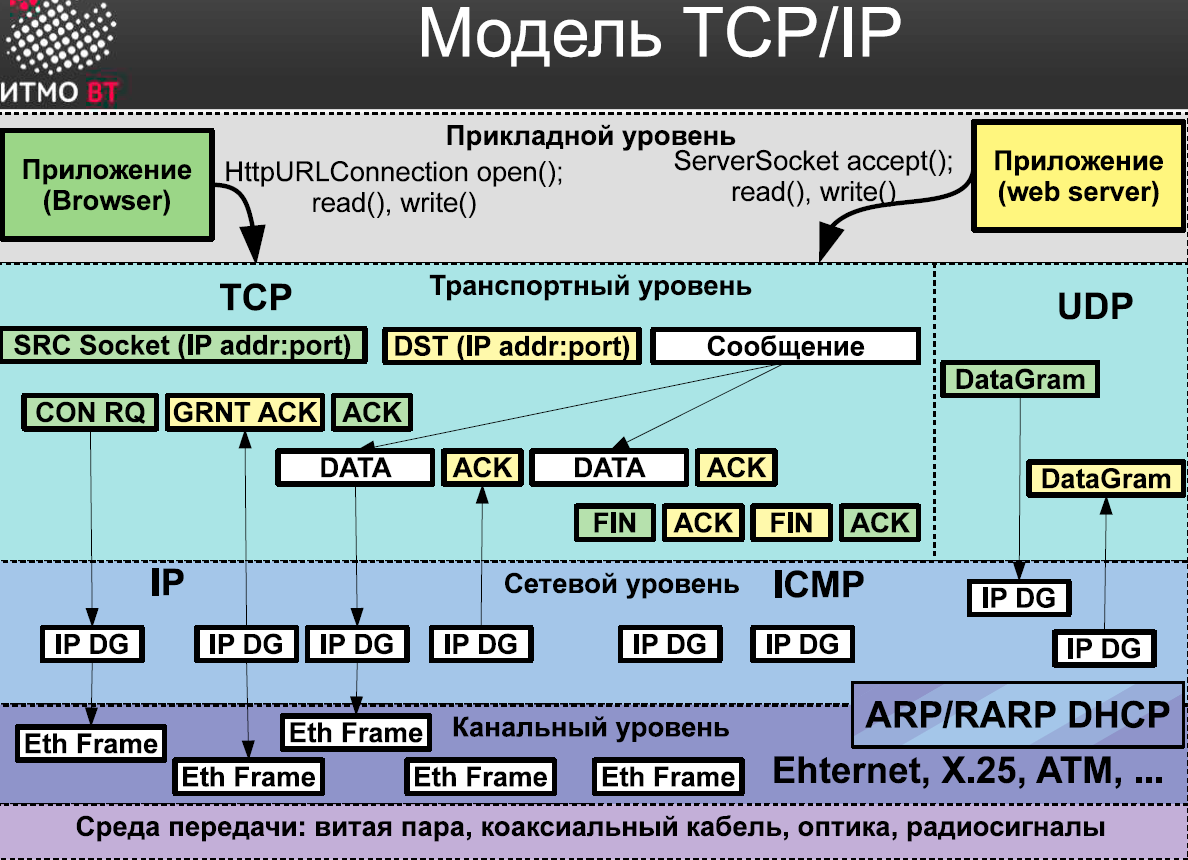


Длинные сообщения бьются на пакеты (обычно 1500 байт). Каждое сообщение и пакеты содержат заголовок (соответствует уровню передач, содержит IP, служебную информацию о портах, адреса отправления и назначения), данные и концевик (контрольная сумма по которой производится проверка полученных данных).



Для организации обмена по сети была придумана эталонная 7-уровневая модель взаимодействия систем. По мере понижения уровней сообщение обворачивается в дополнительный заголовок. На нижних уровнях присутствует заголовок всех предыдущих уровней. После передачи в другую систему в обратном порядке эти заголовки разворачиваются.

1. [*Модель TCP/IP: передающая среда, канальный и сетевой уровень. Адресация, передача и маршрутизация пакетов.*](https://docs.google.com/document/d/16KdfSF0s1K1T25zAv2ozuWFhWl1p_-vdHZjAxsL59qI/edit#heading=h.uuh9avfgmwpq)



Сетевой уровень (IP) – главная задача передавать пакеты между узлами сети. На сетевом уровне все наши служебные пакеты и пакеты с данными преобразуются в IP gatagramm: туда добавляются IP адреса, служебная информация и др. С помощью этого уровня проходит пересылка данных между несколькими маршрутизаторами. На сетевом уровне адреса задаются с помощью IP адресов, которые разбиты на 2 части: один определяет номер сети для групповой маршрутизации, другой – хоста внутри сети.

Маршрутизация – компьютер пытается выполнить коннект к заданному адресу. Для этого нужно сформировать пакет для отсылки по сети. Для этого у каждого компьютера есть таблица маршрутизации. В ней есть запись, соответствующая локальному сегменту сети, ее надо маршрутизировать локально те узнать МAC адрес для дальнейшей его пересылки. Если же передача данных осуществляется адресу, не входящему в локальную сеть, передача осуществляется роутеру по умолчанию в нашей локальной сети. Происходит запрос всем локальным адресам, с помощью которого машина получает нужный адрес из роутера. Пакет дооформляется IP адресами, уровень MAC добавляется нужными MAC-адресами. Пакет попадает на роутер, который разворачивает его MAC адреса и получает IP. По собственной таблице маршрутизации пакет оборачивается другими MAC адресами и отправляется на сервер и распаковывается, обрабатывается и по тому же пути отправляется на нашу локальную машину.

Канальный уровень – каждый пакет на уровне IP преобразуется в пакет канального уровня на уровне Ehternet. Состоит из преамбулы, адресов назначения и источника (MAС адреса, состоят из 6 байт информации, которые прошиты в устройстве и остаются неизменными), тип пакета и полезной нагрузки, содержащей протоколы высокого уровня (IP). Попадает в передающую среду (кабели и тд) и передается с одной физической машины на другую.

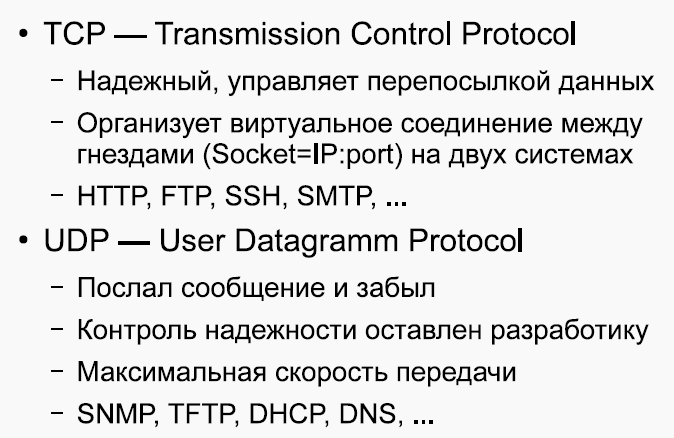
Передающая среда

* Коаксиальный кабель – антенна, которая приходит домой с телевизором, надежность оставляла желать лучшего, тк при разрыве происходило разделение сети на 2 части, которые не видели друг друга.
* Витая пара – появились свичи, коммутаторы, с помощью которых кабель был подключен к каждому компьютеру. При поломке одного кабеля ломалась не вся сеть, а конкретный комп. Преимущество в появлении индукции и распространении сигнала в обе стороны, за счет чего помехи взаимно уничтожаются.
* Оптика – при подключении из кабеля светится лазерный луч, излучающее устройство SFP, имеет возможность передачи на дальние расстояния.
* Wireless – привычный нам WIFI и мобильный интернет. Беспроводная передача информации.

1. [*Модель TCP/IP: выделение адресов (DHCP), сервисы имен, транспортный и прикладной уровни.*](https://docs.google.com/document/d/16KdfSF0s1K1T25zAv2ozuWFhWl1p_-vdHZjAxsL59qI/edit#heading=h.584zw9dliu6)

Прикладной уровень – состоит из приложения (к примеру, привычный нам веб-браузер) на локальной машине и приложения на другой вычислительной машине, к которому мы обращаемся (веб-сервер). Общение с удаленной системой имеет аналогию взаимодействия чтения из файла и записи в него.

С практической точки зрения любой веб-браузер обращается к серверу, устанавливает соединение и выкачивает необходимую нам информацию.



Транспортный уровень (TCP) – сообщение разбивается специальными драйверами на пакеты данных. Этот уровень организует виртуальный канал, для которого посылает огромное количество служебных пакетов. Сначала запрос на соединение, на которое сервер отвечает разрешением или отказом. В случае разрешения посылается на сервер еще одно подтверждение, что разрешение принято и начинается передача данных пакетами (передаем данные – подтверждаем и так несколько раз). Для закрытия соединения посылаются соответствующие пакеты, на которые сервер также отвечает, что этот запрос принят и соединение с его стороны прервано.

Кроме TCP протокола существует UDP. Разница состоит в том, что TCP медленный и требует установления соединения и его завершения, а UDP этого не требует и на практике просто послал пакет и забыл, быстрый.

DHCP – протокол, позволяющий выдавать IP адреса автоматически.

Для преобразования IP адресов во что-то более понятное для человека был придуман сервис имен. Простейший способ /etc/hosts. Концепция DNS упрощает эту задачу. Построена иерархическим путем: есть корневые серверы в сети интернет, которые содержат зоны, каждая из которых содержит свои name сервера, которые содержат информацию локально. При этом сервис имен не ограничивается DNS. Кроме имен машин нужно еще хранить имена пользователей, групп, номера телефонов. К таким серверам относится LDAP, предназначенный для хранения всего.

*34. Интерфейсы ввода-вывода. Контроллеры внешних устройств. Уровни стандартизации, сопряжения с*

*системной шиной, циклы обмена. Регистры контроллера.*

*35. Параллельная передача данных. Контроллеры параллельной передачи и приема.*

*36. Синхронные последовательные интерфейсы. Контроллеры последовательной передачи и приема.*

*37. Асинхронный обмен. Принципы деления частоты, формат кадра.*

*38. Контроллер передачи асинхронного последовательные интерфейса.*

*39. Контроллер приема асинхронного последовательные интерфейса.*

*40. Организация прямого доступа к памяти. Контроллер ПДП.*

<https://docs.google.com/document/d/12mt8a1ifyFH341VFZ4z_kT-_EDrm5X_GAZi1mlFPMGo/edit#heading=h.3f7cz3k0lv02>