[1 Принципы программного управления Фон Неймана. Отличия Неймановской модели ВС от Гарвардской. 4](#_Toc12360562)

[2 Функциональные блоки компьютера, их назначение и взаимосвязь. 6](#_Toc12360563)

[3. Операционные устройства компьютера, их назначение, типы и характеристики. 9](#_Toc12360564)

[4 Оперативные запоминающие устройства компьютера, их типы и характеристики. 10](#_Toc12360565)

[5 Внешние запоминающие устройства. Пpинципы хpанения инфоpмации на внешних запоминающих устpойствах. 11](#_Toc12360566)

[6 Устройства ввода/вывода. Классы устройств, принципы их функционирования. 13](#_Toc12360567)

[7 Блочный принцип построения аппаратных средств компьютера. Понятие интерфейса. 15](#_Toc12360568)

[8 Структурная организация компьютеров. Варианты структурных подходов к построению компьютеров, их достоинства и недостатки. 16](#_Toc12360569)

[9. Организация ввода/вывода компьютерных систем. Способы организации пространства портов ввода/вывода. Работа по поллингу и по прерываниям. 18](#_Toc12360570)

[10. Двоичная и двоично-кодированные системы, способы перевода чисел. 20](#_Toc12360571)

[11 Внутреннее представление данных. Данные с фиксированной точкой, данные с плавающей точкой. Данные в упакованном и pаспакованном десятичном фоpмате. 21](#_Toc12360572)

[12 Понятие команды, операции, операндов, программы. 22](#_Toc12360573)

[13 Логическая схема функционирования компьютера. Логический состав процессора и назначение его компонентов. 23](#_Toc12360574)

[14 Процесс функционирования компьютера при выполнении программы. 25](#_Toc12360575)

[16 Классификация машинных команд по функциональному назначению. Группа арифметико-логических команд. Примеры. 27](#_Toc12360576)

[17 Классификация машинных команд по функциональному назначению. Группа команд переходов и управления состоянием процессора. Примеры. 29](#_Toc12360577)

[18 Методы адресации памяти. Индексная, косвенная, индексно - относительная адресация, организация стековой памяти. Примеры. 31](#_Toc12360578)

[19 Методы адресации памяти. Абсолютная, непосредственная, относительная, регистровая адресация. Примеры. 33](#_Toc12360579)

[20 Подпрограммы. Предпосылки использования, механизмы передачи и возврата управления. Примеры. 34](#_Toc12360580)

[21 Организация подпрограмм, механизмы передачи параметров. Примеры. 36](#_Toc12360581)

[22 Прерывания, механизмы обработки прерываний, передача параметров программным прерываниям. 38](#_Toc12360582)

[23. Классификация программного обеспечения. 41](#_Toc12360583)

[24. Операционная система. Назначение и функции. 42](#_Toc12360584)

[25. Типы операционных систем. Режимы работы операционных систем. 43](#_Toc12360585)

[26. Системы программирования. Классификация языков программирования. 45](#_Toc12360586)

[27. Специальное (проблемное) программное обеспечение. 47](#_Toc12360587)

[28. Базовая структура процессоров i80x86.Регистры процессоров i80x86. Назначение регистров. 48](#_Toc12360588)

[29. Распределение адресного пространства памяти системы. Использование блока основной (conventional) памяти в реальном режиме. 50](#_Toc12360589)

[30. Распределение блоков верхней памяти, свыше 640 К. Обеспечение свойства самоконфигурируемости векторов прерываний системы. 52](#_Toc12360590)

[31. Адресация памяти более 1 Мб в реальном режиме. Отображаемая память, область памяти высших адресов. 53](#_Toc12360591)

[32. Организации видеопамяти для текстовых и графических режимов. 55](#_Toc12360592)

[33. Базовая система адресации процессоров I80x86. Адресация в реальном режиме. Методы адресации. 58](#_Toc12360593)

[34. Организация системы прерываний семейства IBM PC. 60](#_Toc12360594)

[35. Краткая хаpактеpистика(ПЕРЕДЕЛАТЬ СО СЛАЙДОВ) 63](#_Toc12360595)

[микpопpоцессоpов семейства I80x86 и их клонов. 63](#_Toc12360596)

[36. Расширения процессоров 80386 и выше для 32-х и 64-х разрядного режима, регистры процессоров, их назначение. 65](#_Toc12360597)

[37. Защищенный режим процессоров 80х86. Организация памяти в защищенном режиме. 66](#_Toc12360598)

[38. Понятие об организации виртуальной памяти и страничной адресации. Реализация страничной памяти в микропроцессорах семейства 80х86. 68](#_Toc12360599)

[39. Ключевые аспекты развития вычислительных систем. Основные исторические моменты развития, примеры ранних вычислительных систем и их влияние на историю развития ВС в целом. 70](#_Toc12360600)

[40 Архитектура ARM. Модель программирования, режимы работы процессора. Расширения архитектуры ARM. Особенности реализации системы команд. 72](#_Toc12360601)

1 Принципы программного управления Фон Неймана. Отличия Неймановской модели ВС от Гарвардской.

**1.Принцип двоичного кодирования.**

Для представления данных и команд используется двоичная система счисления.

**2.Принцип однородности памяти.**

Команды и данные хранятся в одной и той же памяти (и кодируются в одной и той же системе счисления). Над командами можно выполнять такие же действия, как и над данными.

**3.Принцип адресуемости памяти.**

Структурно основная память состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка.

**4.Принцип последовательного программного управления.**

Все команды располагаются в памяти и выполняются последовательно, одна после завершения другой.

**5.Принцип условного перехода.**

Команды из программы не всегда выполняются одна за другой. Возможно присутствие в программе команд условного перехода, которые изменяют последовательность выполнения команд в зависимости от значений данных.

**6.Принцип хранимой программы.**

Реализуемая компьютером функция целиком определяется хранимой в памяти программой; замена программы полностью изменяет функцию компьютера.

**Фон Неймановская архитектура** предполагает, что программа и данные находятся в общей памяти, доступ к которой производится по одной шине данных и команд. Основным преимуществом - гибкость, так как для изменения программы достаточно просто загрузить новый код в соответствующую область памяти. Состоит из центрального процессора памяти и общей шины, по которой в обоих направлениях пересылаются данные. ЦПУ также должен взаимодействовать и с окружающим миром. При этом данные к/от соответствующих интерфейсных портов передаются по одной общей шине данных. Однако использование общей шины означает, что в любой момент времени может выполняться только одна операция. Соответственно, пересылка данных между ЦПУ и памятью данных не может осуществляться одновременно с выборкой команды.

**Гарвардская архитектура** соответствует структуре с разделенными устройствами памяти команд и данных и отдельными шинами команд и данных. Программа считывалась с бумажной перфоленты. Такая концепция была более эффективной, чем фон-неймановская архитектура, поскольку код программы мог считываться из памяти программ одновременно с обменом между ЦПУ и памятью данных или с операциями ввода/вывода. Однако такие машины были намного сложнее и дороже в изготовлении.

2 Функциональные блоки компьютера, их назначение и взаимосвязь.

Функциональный состав компьютера, включающий устройства трех основных классов:

1. операционные, предназначенные для выполнения операций над информацией;
2. запоминающие, предназначенные для хранения информации;
3. устройства ввода-вывода, предназначенные для связи вычислительной системы с внешней средой.

Важнейшими устройствами, без которых невозможна реализация программного управления, является цп и оп или ОЗУ. Эти устройства часто называются центральными. Внешние запоминающие устройства и устройства ввода/вывода называются периферийными устройствами.

Центральный процессор служит для выполнения арифметических и логических операций над данными, также для управления всеми компонентами вычислительной системы. Одной из важнейших характеристик процессора является его архитектура.

Выделяют следующие архитектуры процессоров:

1)CISC – процессоры;

2)RISС – процессоры;

3)VLIW – процессоры;

**CISС** - это процессоры со сложной системой команд. Система команд такого процессора имеет инструкции, различающиеся внутренним форматом, методами адресации, длиной и временем выполнения. Система команд процессора достаточно громоздкая, но гибкая.

**RISC** - процессоры с сокращенным набором команд имеют инструкции унифицированного формата, совпадающие по длине, методам адресации операндов и времени выполнения. Это позволяет упростить схемотехнику процессоров и реализовать возможность автоматического распараллеливания операций в отдельных фрагментах программы.

**VLIW**– процессоры с очень длинной инструкцией – являются компромиссом между CISC и RISC системами, имеют длинные инструкции, различающиеся форматами, длиной и временем выполнения. И эти длинные инструкции при обработке разбиваются на совокупность RISC инструкций, интерпретируемых внутренним RISC-ядром процессора.

**Оперативная память RAM** - устройство, способное работать в темпе процессора. Служит для хранения оперативной информации и выдачи блокам машины этой информации по их требованию. ОП состоит из ячеек - байтов. Каждый байт состоит из восьми двоичных разрядов.

В состав центральных устройств компьютера практически всегда входят блоки постоянных запоминающего устройства (ПЗУ). Память данного типа предназначена только для считывания хранимой в ней информации. В такой памяти обычно хранятся наиболее часто используемые программы и данные, например, программы обслуживания устройств ввода/вывода и внешних запоминающих устройств, а также программы, необходимые для начальной загрузки вычислительной системы. Объем подобной памяти значительно меньше, чем основного ОЗУ. Ее быстродействие также соизмеримо с быстродействием процессора, однако несколько ниже, чем у обычной оперативной памяти.

**Кэш-память** - обладает наиболее высоким быстродействием и располагается между центральным процессором и оперативной памятью. Предназначена для хранения данных, чаще всего используемых в ходе работы той или иной программы. Имеет ассоциативную или частично ассоциативную организацию. В ходе работы процессор для выборки данных в первую очередь организует обращение в кэш-память, затем, если данные в кэш-памяти не найдены, осуществляется обращение в обычную оперативную память. *Ассоциативная организация* кэш-памяти наряду с ее повышенным быстродействием позволяет повысить скорость выборки данных, что, в свою очередь, существенно повышает быстродействие вычислительной системы в целом.

**Внешние запоминающие устройства** предназначены для длительного хранения больших объемов информации.

**Устройства ввода/вывода** - для связи компьютера с окружающей средой. Любой современный компьютер имеет модульную структуру, т.е. строится из набора модулей, каждый из которых реализует законченную функцию и является относительно независимым от других модулей. Данный принцип обеспечивает унификацию узлов, а также позволяет изменять конфигурацию оборудования компьютера даже в процессе его функционирования - подключать дополнительные модули, заменять одни модули на другие, менять конфигурацию модулей, а это, в свою очередь, существенно повышает гибкость и универсальность вычислительной системы. Связь между отдельными модулями осуществляется посредством интерфейса. Интерфейсом называется совокупность аппаратных средств, системы сигналов и набора алгоритмов, определяющих порядок обмена информацией между устройствами.

3. Операционные устройства компьютера, их назначение, типы и характеристики.

Центральный процессор – устройство, служащее для выполнения арифметических и логических операций над данными, а также управляющим всеми компонентами вычислительной системы.

В машине фон Неймана в качестве операционного устройства применяется АЛУ.

Учитывая разнообразие выполняемых операций и обрабатываемых данных, можно говорить не об одном устройстве, а о целом комплексе специализированных операционных устройств, каждое из которых выполняет свое подмножество операций.

**АЛУ** — блок процессора, который под управлением устройства управления служит для выполнения арифметических и логических преобразований над данными, называемыми в этом случае операндами. Разрядность операндов обычно называют размером или длиной машинного слова.

Следует выделить операционные устройства:

* целочисленной арифметики.
* логических операций.
* десятичной арифметики.
* чисел с плавающей запятой.

Кроме указанных устройств процессоры в зависимости от их назначения могут иметь и другие операционные устройства: управления потреблением, графических операций, упаковки/распаковки изображений и др.

В минимальном варианте АЛУ должно содержать аппаратуру для реализации лишь основных логических операций, сдвигов, инвертирования, а также сложения чисел в формате с фиксированной запятой. Опираясь на этот набор, можно программным способом обеспечить выполнение остальных арифметических и логических операций как для чисел с фиксированной запятой, так и для других форм представления информации. Следует отметить, что подобный вариант не позволяет добиться высокой скорости вычислений, поэтому по мере расширения технологических возможностей доля аппаратных средств в составе АЛУ постоянно возрастает.

*Набор элементов, на основе которых строятся структуры различных операционных устройств, называется структурным базисом.* **Структурный базис** операционных устройств включает в себя:

* регистры, обеспечивающие хранение слов данных;
* шины, связывающие регистры и предназначенные для передачи слов данных;
* комбинационные схемы, реализующие вычисления по управляющим сигналам от устройства управления. Выполнение команды может быть сведено к нескольким операциям пересылки из регистра в регистр. Возможно с промежуточным преобразованием пересылаемых двоичных слов на комбинационных логических схемах.

4 Оперативные запоминающие устройства компьютера, их типы и характеристики.

**Оперативная память (RAM)** или **оперативное запоминающее устройство (ОЗУ)** — энергозависимая часть системы компьютерной памяти, в которой во время работы компьютера хранится выполняемый машинный код, входные, выходные и промежуточные данные, обрабатываемые процессором.

Обмен данными между процессором и оперативной памятью производится: непосредственно или через сверхбыструю память 0-го уровня — регистры в АЛУ, либо при наличии аппаратного кэша процессора — через кэш. Кэш находится между ЦП и ОП. Процессор проверяет, если копия данных присутствует в кэше, то от берёт ее оттуда, чтобы не нагружать оп. Если кэш заполняется полностью, то из нее удаляются самые редкоиспольз. данные.

**Постоянная память** используется для хранения системной информации.

**Дополнительная память** – внешние устройства.

Наибольшее распространение получила два вида ОЗУ:

* Статическая память (SRAM), в виде массива триггеров.
* Динамическая память (DRAM), в виде массива конденсаторов.

**SRAM** – этот вид памяти является более дорогим в расчете на хранение 1 бита, но как правило имеет наименьшее время доступа и меньшее энергопотребление, чем DRAM.

**DRAM** - однобитовая ячейка памяти содержит конденсатор и транзистор. Так как хранение 1 бита информации в этом виде памяти дешевле и такой вид занимает меньше места в кристалле, DRAM преобладает в компьютерах 3-поколения. Однако DRAM медленнее работает, конденсаторы со временем разряжаются.

5 Внешние запоминающие устройства. Пpинципы хpанения инфоpмации на внешних запоминающих устpойствах.

Внешние запоминающие устройства предназначены для длительного хранения больших объемов информации.

К ВЗУ относятся накопители на магнитных дисках различных типов, накопители на магнитных лентах, накопители на оптических и магнитооптических дисках.

ВЗУ по физическому принципу:  
**I С магнитной записью:**

**Магнитные диски:**

* + - **Гибкий магнитный диск** - носитель информации, используемый для многократной записи и хранения данных.

Плюсы: огромная плотность записи, низкое энергопотребление, высокая надежность работы.

Минусы: недолговечность.

* + - **Жесткий магнитный диск** - произвольного доступа, основанное на принципе магнитной записи.

Плюсы: большой объем, относительно быстрая скорость записи/чтения.

Минусы: риск потери информации.

**II Оптические:**

* + - **CD** - оптический носитель информации в виде пластикового диска с отверстием в центре, процесс записи и считывания, информации которого осуществляется при помощи лазера.

Плюсы: удобство обмена информацией.

Минусы: малая надежность, низкая скорость чтения/ записи, малая емкость отдельного носителя.

* + - **DVD** - оптический носитель информации, для хранения различной информации в цифровом виде.

Плюсы: объем данных, поддержка различных экранных форматов, многоканальный звук.

Минусы: проблема совместимости с приводом.

* + - **Blue-ray** - формат оптического носителя, используемый для записи с повышенной плотностью хранения цифровых данных, включая видео высокой чёткости.

Плюсы: высокое качество изображения и звука.

Минусы: малое число blue-ray медиа-файлов и т.д.

**III Магнитно-оптические** - носитель информации, сочетающий свойства оптических и магнитных накопителей. Для чтения информации используется магнитная система, для записи — одновременно оптическая и магнитная.

* Плюсы: невысокая стоимость, гарантия качества записи, синхронный вывод, длительное время хранения
* Минусы: низкая скорость записи, высокое энергопотребление, малая распространенность.

**IV Твердотельные:**

* + - **USB - Stick**- запоминающее устройство, использующее в качестве носителя флеш - память.

Плюсы: малый вес, бесшумность, портативность, универсальность, низкое энергопотребление.

Минусы: скорость записи и чтения ограничены, ограниченное число циклов записи/стирания.

* + - **SSD** - немеханическое запоминающее устройство на основе микросхем памяти, содержащий управляющий контроллер. Создает на базе драм памяти, снабженной аккумулятором.

Плюсы: бесшумность, высокая скорость чтения/записи, малые габариты и вес.

Минусы: ограниченное количество циклов перезаписи, высокая себестоимость и т.д.

* + - **SD/MMC.**
    - **CF.**
    - **MS.**
    - **xD-Picture.**

6 Устройства ввода/вывода. Классы устройств, принципы их функционирования.

УВВ – предназначены для связи компьютера с окружающей средой, в том числе и с человеком.

Устройства ввода/вывода обмениваются информацией с магистралью по тем же принципам, что и память. Модуль памяти имеет в адресном пространстве системы много адресов, а устройство ввода/вывода обычно имеет немного адресов, а иногда и всего один адрес.

Самые **простейшие устройства ввода/вывода** выдают на внешнее устройство код данных в параллельном формате и принимают из внешнего устройства код данных в параллельном формате. Такие устройства ввода/вывода часто называют ***параллельными портами***ввода/вывода. Они наиболее универсальны, поэтому их часто вводят в состав микропроцессорной системы в качестве стандартных устройств. Параллельные порты обычно имеются в составе микроконтроллеров.

**Входной порт** в простейшем случае представляет собой параллельный регистр, в который процессор может записывать информацию.

**Выходной порт** обычно представляет собой просто однонаправленный буфер, через который процессор может читать информацию от внешнего устройства. Порт может быть и двунаправленным (входным/выходным). В этом случае процессор пишет информацию во внешнее устройство и читает информацию из внешнего устройства по одному и тому же адресу в адресном пространстве системы.

**Более сложные устройства ввода/вывода** имеют в своем составе внутреннюю буферную оперативную память и даже могут иметь микроконтроллер, на который возложено выполнение функций обмена с внешним устройством.

Каждому устройству ввода/вывода отводится свой адрес в адресном пространстве микропроцессорной системы.

Устройства ввода/вывода, помимо программного обмена, могут поддерживать режим обмена по прерываниям.

К устройствам *ввода для интерфейса пользователя* относятся контроллеры клавиатуры, тумблеров, отдельных кнопок, мыши, трекбола, джойстика и т.д.

К устройствам ***вывода для интерфейса*** *пользователя* относятся контроллеры светодиодных индикаторов, табло, жидкокристаллических, плазменных и электронно-лучевых экранов и т.д.

**У*стройства ввода/вывода для длительного хранения информации***обеспечивают сопряжение микропроцессорной системы с дисководами (компакт-диски или магнитные диски), а также с накопителями на магнитной ленте. Применение таких устройств существенно увеличивает возможности микропроцессорной системы в отношении хранения выполняемых программ и накопления массивов данных. В простейших контроллерах эти устройства отсутствуют.

**Т*аймерные устройства*** отличаются тем, что они могут не иметь внешних выводов для подключения к внешним устройствам. Предназначены для того, чтобы микропроцессорная система могла выдерживать заданные временные интервалы, следить за реальным временем, считать импульсы и т.д.

7 Блочный принцип построения аппаратных средств компьютера. Понятие интерфейса.

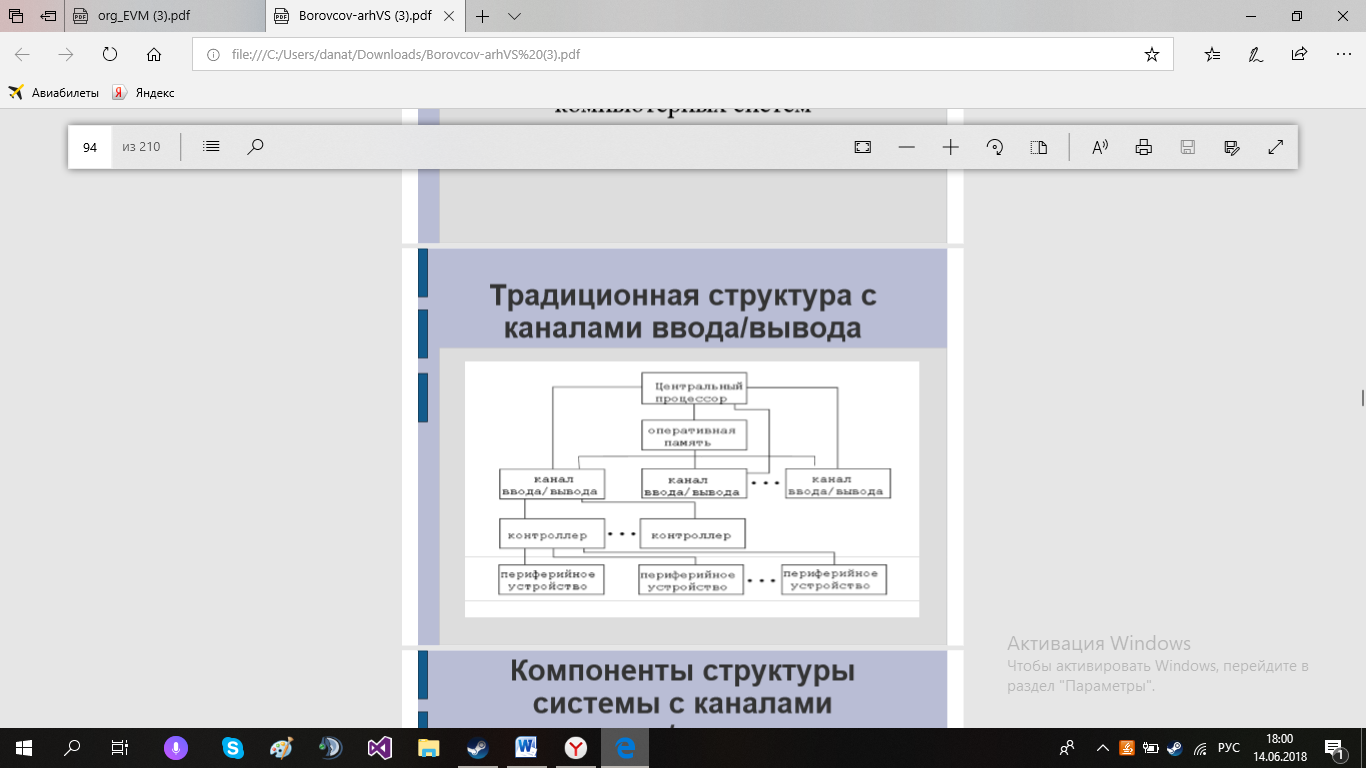
Любой современный компьютер имеет модульную структуру, т.е строится из набора модулей, каждый из которых реализует законченную функцию и является относительно независимым от других модулей.

Данный принцип обеспечивает унификацию узлов, а также позволяет изменять конфигурацию оборудования компьютера даже в процессе его функционирования ( процесс называется HotPlug ) – подключать дополнительные модули, заменять одни модули на другие, менять конфигурацию модулей, а это, в свою очередь, существенно повышает гибкость и универсальность ВС. Связь между отдельными модулями осуществляется посредством интерфейса. Интерфейсом называется совокупность аппаратных средств, системы сигналов и набора алгоритмов, определяющих порядок обмена информацией между устройствами.

8 Структурная организация компьютеров. Варианты структурных подходов к построению компьютеров, их достоинства и недостатки.

Рассмотрим два варианта варианты объединения устройств компьютера в единую  
вычислительную систему, являющиеся базовыми.

**1.** Структура высокопроизводительного компьютера с канальной архитектурой:



Каналы ввода-вывода представляют собой специализированные устройства, обеспечивающие обмен информацией между ОП и ПУ.

Каналы могут быть **селекторные** и **мультиплексные**.

* **Селекторные** каналы предназначены для обмена с быстродействующими ВУ.
* **Мультиплексные** каналы обеспечивают обмен с группой низкоскоростных устройств.

Основное свойство канала в том, что он работает по специальной канальной программе и освобождает ЦП от выполнения трудоемких операций ввода-вывода.

**Контроллер ПУ** - устройство управления ПУ, выполняет следующие **функции**:

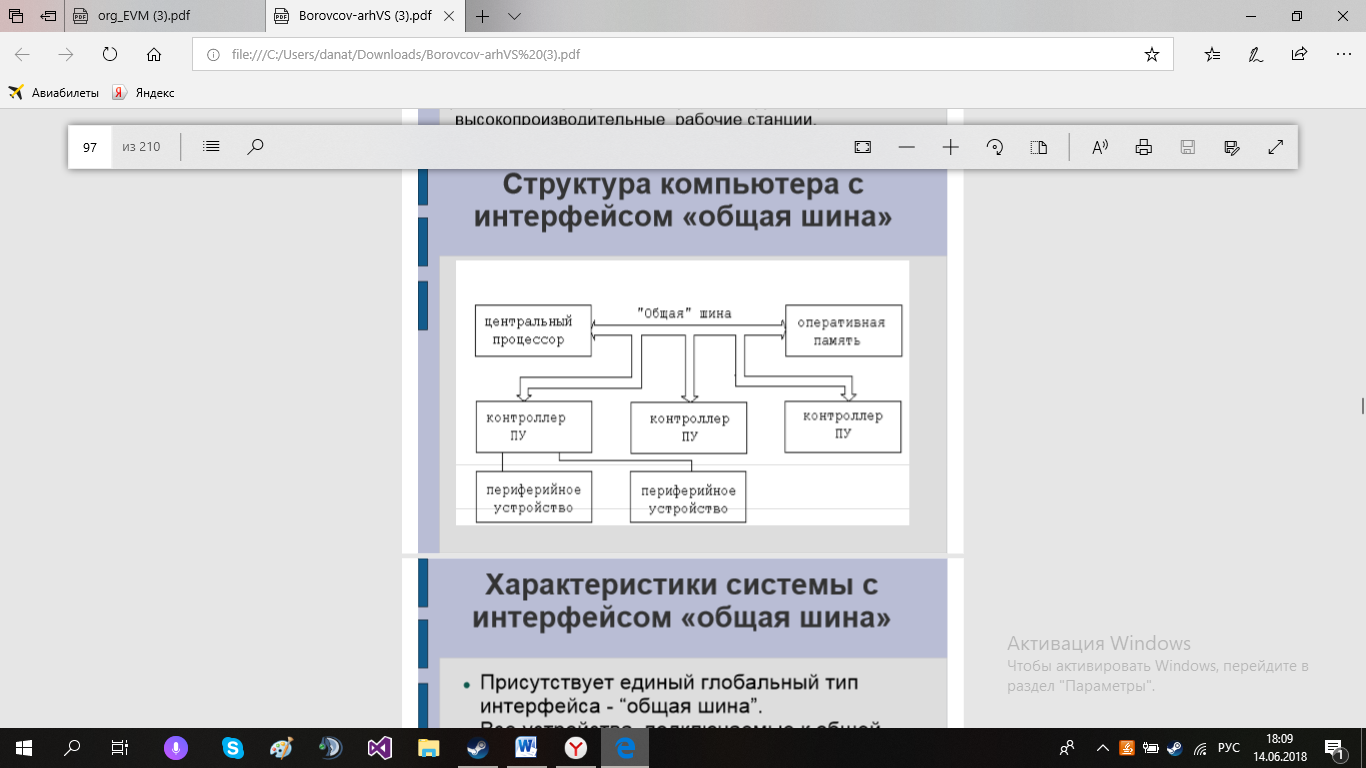
* преобразование последовательности сигналов интерфейса в последовательность сигналов, управляющих работой ПУ;
* обеспечивает синхронизацию работы ПУ с другими устройствами;
* обеспечивает буферизацию информации.

В данной структуре имеется **четыре типа интерфейса**:

1. интерфейс оперативной памяти;
2. интерфейс каналов;
3. интерфейс контроллеров;
4. интерфейс периферийного устройства.

С использованием данного структурного решения строятся мощные высокопроизводительные системы - суперкомпьютеры, узловые компьютеры сетей, файловые сервера, высокопроизводительные графические рабочие станции.

**2.** Второй вариант структурного решения - с использованием интерфейса **«общая шина»** - используется в мини-, микрокомпьютерах и в большинстве персональных систем.



Все устройства, подключаемые к общей шине, являются равноправными. В каждый момент времени по общей шине может обмениваться только пара устройств, причем одним устройством из этой пары является процессор, который полностью берет на себя функции управления обменом.

Достоинством данного варианта являются:

* простота аппаратной реализаций ведущая к резкому снижению себестоимости конструкции;
* единый интерфейс для оперативной памяти, процессора и контроллеров всех типов ПУ.

Однако, с другой стороны, отвлечение процессора на выполнение операций обмена значительно снижав общую производительность системы и повышает простои процессора.

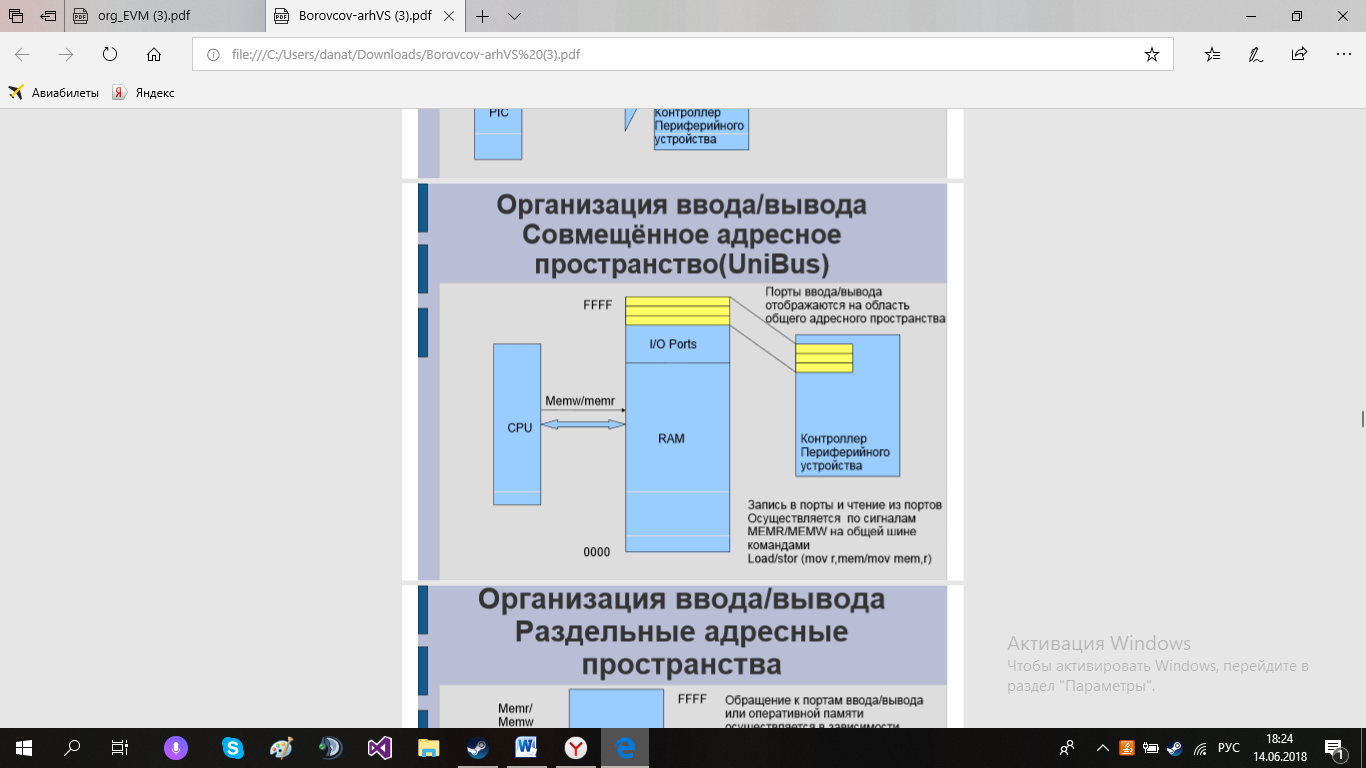
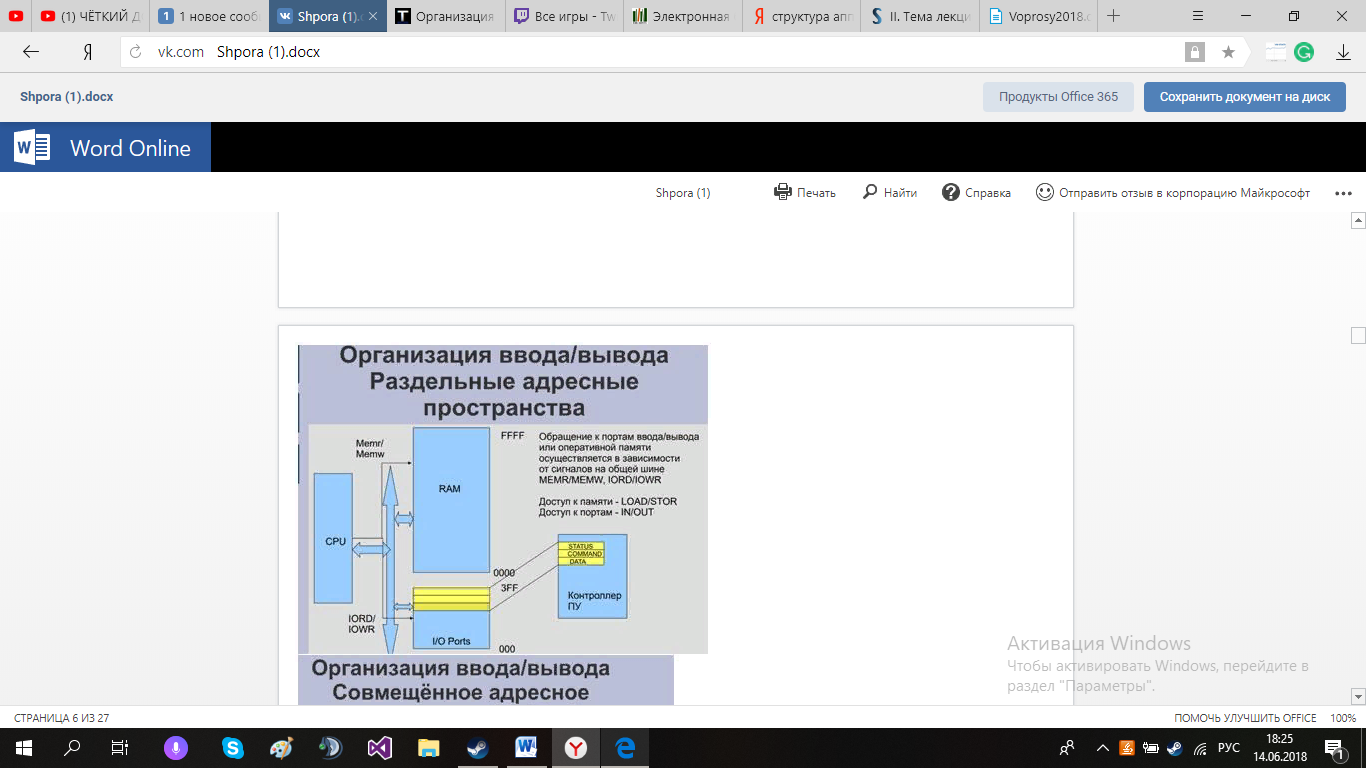
9. Организация ввода/вывода компьютерных систем. Способы организации пространства портов ввода/вывода. Работа по поллингу и по прерываниям.

Каждое устройство ввода/вывода вычислительной системы снабжено устройством управления – контроллером.

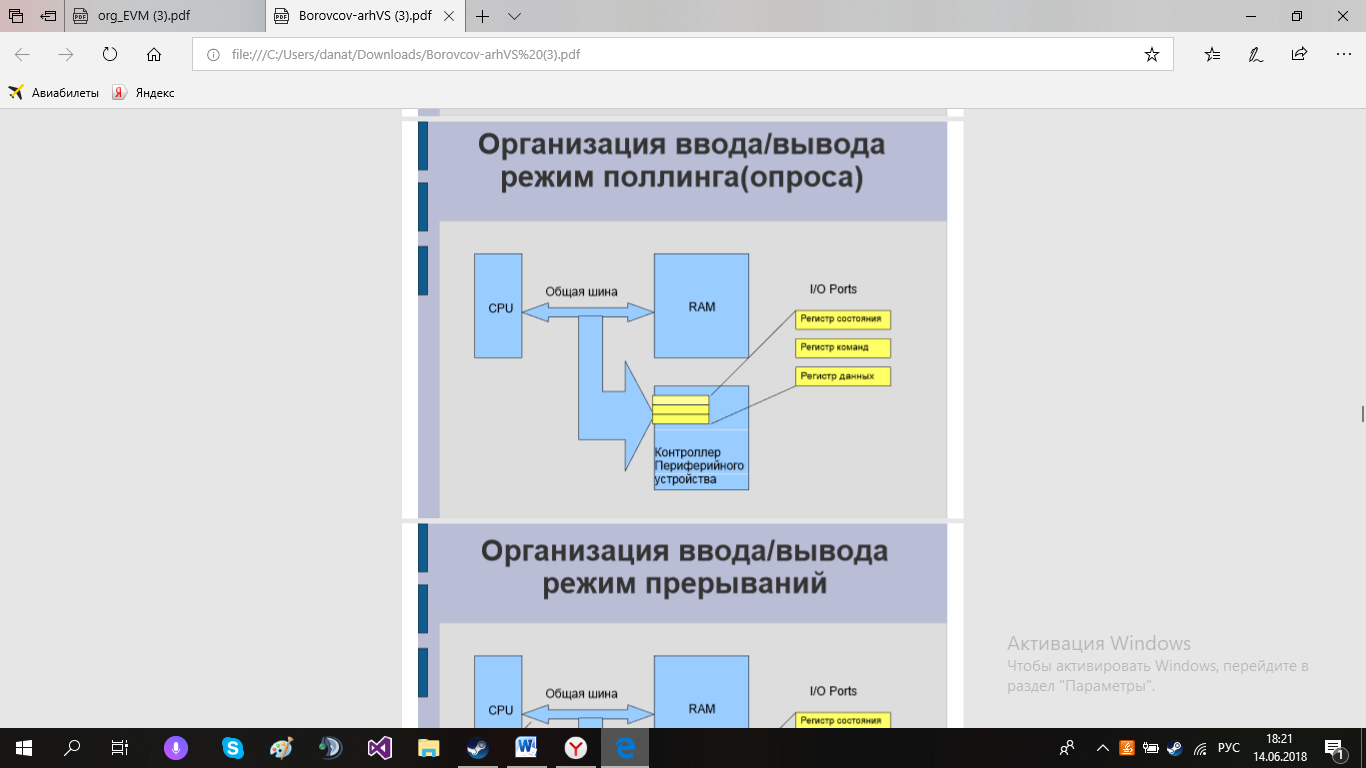
Контроллер взаимодействует с драйвером – системным программным модулем, предназначенным для управления данным устройством. Контроллер периодически принимает от драйвера выводимую на устройство информацию, а также команды управления, которые говорят о том, что с этой информацией нужно сделать.

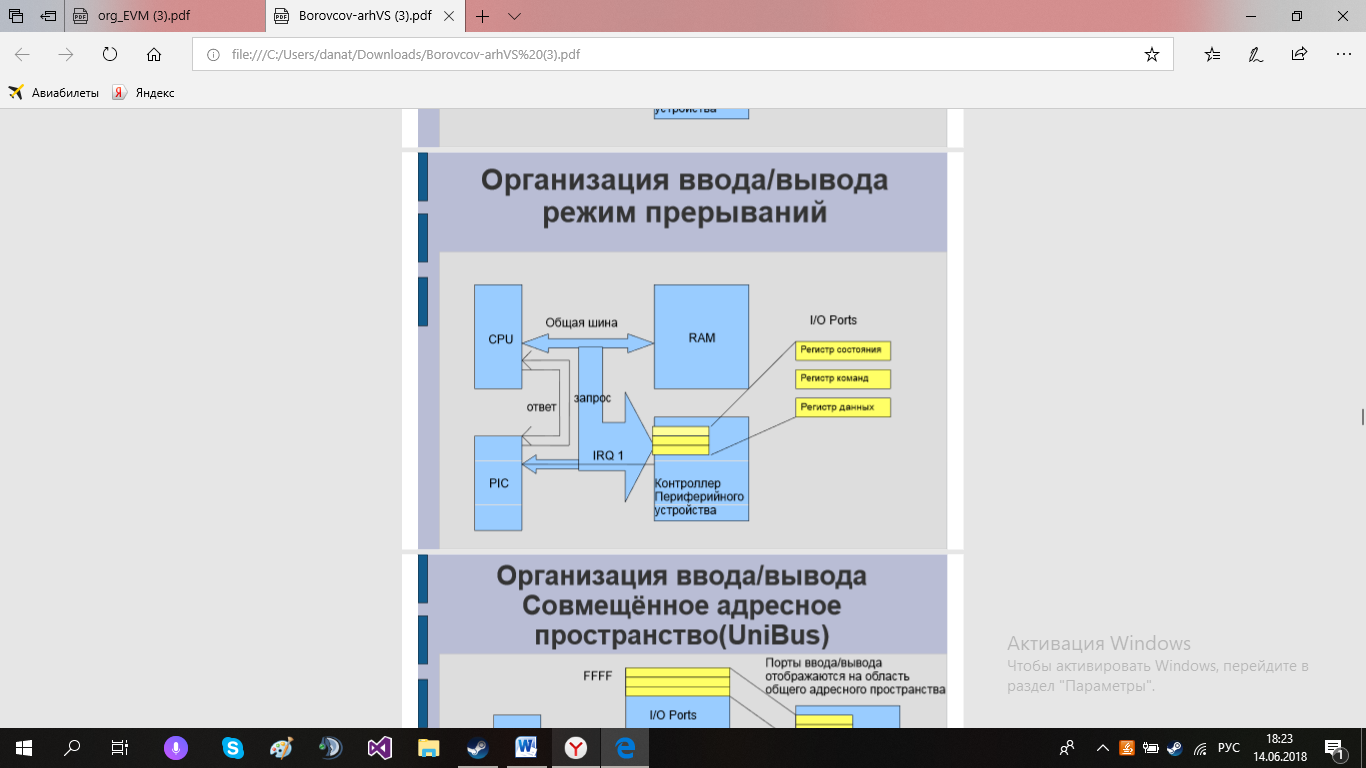
Адресное пространство портов ввода/вывода составляет 64 Kбайта. Каждое устройство в большинстве случаев представлено несколькими портами, занимающими смежные адреса адресного пространства портов ввода/вывода, при этом порт с самым младшим адресом принято называть базовым.

В большинстве случаев периферийные устройства имеют стандартные адреса портов, однако, следует иметь в виду, что в ряде случаев, с целью устранения конфликтов с уже установленным оборудованием, адаптер периферийного устройства может быть переконфигурирован для работы в другой области адресуемого пространства портов.

При поллинге процессор постоянно опрашивает внешнее устройство, дожидаясь его готовности к обмену или наступления другого события. Такой вариант является самым простым, так как требует только чтения портов ввода/вывода, определенные биты которых сигнализируют о наступлении ожидаемого события, однако он приводит к непродуктивным потерям времени процессором, даже если он выполняет свои запросы не постоянно, а через определенные интервалы времени. С другой стороны, при таком варианте, если длительность интервалов достаточно велика, процессор может и пропустить какое-то событие.



При обмене по прерываниям процессор не ждет наступления события, а реагирует на него по факту его наступления, когда формируется сигнал прерывания. В результате процессор сможет отреагировать на все события, без пропуска, и при этом освобождается от необходимости постоянно проводить опрос внешних устройств. Традиционно обмен по прерываниям предназначался в основном для «медленных» внешних устройств, но может использоваться и для быстрых, для инициирования начала обмена, либо – для отслеживания процессором внешних событий без постоянных запросов. 

10. Двоичная и двоично-кодированные системы, способы перевода чисел.

Двоичная система счисления является позиционной с/с с основанием 2. Для представления любого разряда двоичного числа используется всего две цифры – 0 и 1. Применение двоичной системы счисления в компе логически вытекает из того, что электронные схемы, из которых состоит аппаратура компа, способны различать только два состояния – наличие сигнала (1), или его отсутствие (0).

* + плюсы:
    - Простота выполнения арифметических и логических операций и, как следствие, простота устройств, реализующих эти операции.
    - Возможность использования аппарата алгебры логики для анализа и синтеза операционных устройств.

Для представления числа в двоичной системе счисления воспользуемся тем, что каждому разряду двоичного числа соответствует определенная степень двойки.

Для представления информации в компьютерах используют и другие системы счисления, позволяющие записывать числа в более компактной форме, а именно, восьмеричную и шестнадцатеричную системы счисления.

Эти системы счисления относятся к двоично-кодированным системам, у которых основание системы счисления кратно некоторой степени двойки:

* 23 - для восьмеричной
* 24 - для шестнадцатеричной систем счисления

Запись восьмеричного числа в три раза, а запись шестнадцатеричного числа в четыре раза короче, чем запись того же числа в двоичном виде. В то же время восьмеричное или шестнадцатеричное представление числа эквивалентно двоичному, поскольку каждая восьмеричная или шестнадцатеричная цифра получены просто заменой двоичных разрядов, сгруппированных по три или по четыре, соответствующей восьмеричной или шестнадцатеричной цифрой.

11 Внутреннее представление данных. Данные с фиксированной точкой, данные с плавающей точкой. Данные в упакованном и pаспакованном десятичном фоpмате.

Действительные числа в памяти компьютера представляются в формате двоичного с плавающей точкой (2спт) и занимают, как минимум четыре байта. В зависимости от типа машины и требуемой точности внутреннее представление числа в формате 2спт может быть различным.

В любом случае, во внутреннем представлении числа с плавающей точкой выделяются три поля:

* поле кода знака (“0” - плюс, “1” - минус),
* поле характеристики, которая является преобразованным значением порядка числа, и
* поле мантиссы.

Рассмотрим, каким образом кодируются эти поля

* Старший бит старшего байта - код знака числа;
* следующие семь битов старшего байта и один бит следующего за ним байта (всего восемь битов) - характеристика.
* следующие двадцать три двоичных разряда хранят мантиссу числа. Мантисса числа хранится в нормализованном виде, при этом точка располагается после второй значащей двоичной цифрой мантиссы. Поскольку первая значащая двоичная цифра всегда двоичная единица, то эта двоичная единица во внутреннем представлении числа не хранится (!) (так называемый “скрытый” бит).

В упакованном формате для каждой десятичной цифры отводится по 4 разряда, при этом знак числа кодируется в крайнем правом полубайте числа. Упакованный формат используется в ПК при выполнении операций сложения и вычитания. В распакованном формате для каждой десятичной цифры отводится 1 байт, при этом старшие полубайты каждого байта (кроме самого младшего) в ПК заполняется кодом 0011 (в соответствии с ASCII-кодом), а в младших полубайтах обычным образом кодируются десятичные цифры. Старший полубайт (зона) самого младшего (правого) байта используется для кодирования знака числа. Распакованный формат используется при вводе-выводе информации в компьютере, и также при выполнении операций умножения и деления двоично-десятичных чисел.

12 Понятие команды, операции, операндов, программы.

* Программа — последовательность команд с явным/неявным порядком выполнения
* Команда — содержит код операции и инфу об операндах
* Операнд — единица информации, над которыми выполняются операции
* Операция — действие, выполняемое на уровне аппаратуры процессора

13 Логическая схема функционирования компьютера. Логический состав процессора и назначение его компонентов.

В состав процессора компьютера входят следующие основные функциональные блоки:

* + - **Блок управляющий регистров.** 
      * Блок регистров включает в себя:
        + **Счетчик адреса команд**, который содержит адрес ячейки памяти, в которой размещается адрес очередной команды.
        + **Регистр команд**, содержащий код выполняемой в текущий момент времени команды.
        + **Регистр признаков**, отражающий результат выполнения операции, выполненной процессором, а также признаки некоторых особых ситуаций.
        + **Указатель стека** – специальный регистр, с помощью которого осуществляется доступ к стековой памяти.   
          **Стековая память** относится к специальным видам памяти. Для чтения и записи доступная только одна ячейка, называемой вершиной стека. Указатель стека содержит адрес ячейки стековой памяти доступной в настоящий момент для чтения или записи. После записи в стек, указатель меняется так, чтобы он указывал на следующую ячейку. Обычно стек располагается так, что он растет от старших адресов к младшим. В этом случае после выполнения записи в вершину стека, указатель стека автоматически уменьшается, а перед чтением из стека автоматически увеличивается.
      * Таким образом, блок управляющих регистров содержит управляющую информацию, необходимую для выполнения программы. Следует отметить, что состав блока управляющих регистров может различаться для различных типов машин, однако тип хранимой и обрабатываемой в нем информации практически не изменяется.
    - **АЛУ.** Арифметико-логическое устройство (АЛУ) процессора обеспечивает выполнение всех арифметических и логических операций над данными в ходе выполнения программы. АЛУ является достаточно сложным блоком и, помимо блока арифметики с фиксированной точкой, являющегося обязательным для всех машин, может включать в себя блоки арифметики с плавающей точкой и десятичной арифметики. Все арифметические операции в АЛУ выполняются сумматором - специальным блоком, обеспечивающим сложение кодов чисел. Для выполнения логических операций в составе АЛУ имеются и другие блоки.
    - **Регистры общего назначения.** Блок регистров общего назначения (РОН) предназначен для оперативного хранения данных, наиболее часто используемых в ходе выполнения программы Разрядность регистров общего назначения совпадает с разрядностью процессора. Доступ к данным, хранящимся в регистрах общего назначения, осуществляется значительно быстрее, чем к данным в оперативной памяти, однако емкость регистровой памяти очень мала.

14 Процесс функционирования компьютера при выполнении программы.

1. Из оперативной памяти по адресу, записанному в PC, выбирается команда и записывается в регистр команд.
2. Команда дешифруется, т.е. из нее выделяются поля операции и операндов, Одновременно модифицируется PC, так, чтобы он указывал на следующую команду (случае команд переменной длины алгоритм изменения PC может быть несколько иным).
3. Производится выборка операндов из оперативной памяти и/или регистров общего назначения; код операции и операнды подаются в АЛУ.
4. Выполняется операция, причем если операция, арифметическая или логическая, то результат получается в регистрах АЛУ
5. В зависимости от результата операции (положительный, отрицательный,  
   нулевой, переполнение разрядной сетки и др.) устанавливается признак результата в  
   регистре признаков.
6. Результат записывается в соответствующее место оперативной памяти или регистр общего назначения.
7. Происходит возврат к П. 1.

**15. Классификация машинных команд по функциональному назначению. Группа команд пересылки. Примеры.**

Процессоры различных типов компьютеров имеют индивидуальные наборы машинных команд, различающихся как внутренним форматом, так и количеством. Следует заметить, что классификация по различным признакам часто бывает взаимосвязанной, например, длина команды зависит как от методов адресации, так и от количества операндов; с другой стороны количество операндов может определяться, например, функциональным назначением команды и, опять же, методом адресации.

По функциональному назначению набор машинных команд процессора любого компьютера можно условно разбить на четыре основные группы:

* команды пересылки;
* арифметико-логические команды;
* команды переходов;
* команды управления состоянием процессора.

**Группа команд пересылки** включает команды, обеспечивающие пересылку данных из одной ячейки памяти в другую, из ячейки памяти в регистры и обратно, а также из регистра в регистр. Кроме того, в эту группу входят команды обмена с внешними устройствами и команды работы со стеком. Выполнение команд данной группы с точки зрения процессора сводится к пересылке информации из одного места в другое, т.е. отсутствует обращение к сумматору и этап пересылки значения из регистра сумматора по адресу результата. В некоторых процессорах в зависимости от пересылаемого значения может устанавливаться признак результата (**не во всех! - это скорее исключение**).

Примеры:

* mov r1, r2;
* stor r1, mem;
* push r,
* pop r,
* load r1, mem

16 Классификация машинных команд по функциональному назначению. Группа арифметико-логических команд.   
Примеры.

Процессоры различных типов компьютеров имеют индивидуальные наборы машинных команд, различающихся как внутренним форматом, так и количеством. Следует заметить, что классификация по различным признакам часто бывает взаимосвязанной, например, длина команды зависит как от методов адресации, так и от количества операндов; с другой стороны количество операндов может определяться, например, функциональным назначением команды и, опять же, методом адресации.

По функциональному назначению набор машинных команд процессора любого компьютера можно условно разбить на четыре основные группы:

* команды пересылки;
* арифметико-логические команды;
* команды переходов;
* команды управления состоянием процессора.

**Группа команд арифметических и логических операций** включает команды, обеспечивающих выполнение над данными арифметических и логических операций (логическое И, ИЛИ, НЕ, исключающее ИЛИ), а так же операций арифметических и логических сдвигов.

* **Подгруппа команд арифметических операций** реализует бинарные операции (т.е. операции над двумя операндами) “+”, “-“,”\*”, “/” (“сложить”, “вычесть”, “перемножить”, “разделить”).
  + Если с операциями сложения и вычитания все достаточно просто, то реализация операций умножения и деления требует дополнительных пояснений.
    - Большинство процессоров вычислительных систем реализуют **операцию умножения** таким образом, что результат операции имеет двойную длину, то есть, если выполняется операция перемножения данных в формате слова, то результат будет иметь длину двойного слова. Например, если мы перемножаем содержимое регистра и ячейки памяти, то результат будет помещаться в паре регистров - старшая часть результата в одном регистре, младшая - в другом. В какой паре и каким образом - определяется архитектурой конкретного процессора.
    - Аналогичная ситуация наблюдается и в случае **операции деления** - делимое обычно размещается в паре регистров, результат так же занимает два регистра - в одном размещается частное, в другом - остаток от деления.
  + Объясняется это особенностями аппаратной реализации мультипликативных операций в процессорах.
* **Подгруппа логических операций** реализует бинарные операции И(AND), ИЛИ(OR), ИСКЛЮЧАЮЩЕЕ ИЛИ(XOR), а также унарную операцию НЕ(NOT). Операции реализуются побитно, то есть выполняются попарно над соответствующими битами операндов.
* **Подгруппа команд сдвигов** обеспечивает побитные сдвиги содержимого операнда влево или вправо на один или несколько разрядов. В зависимости от метода реализации различается сдвиг влево, сдвиг вправо арифметический или логический, сдвиг влево или вправо циклический с учетом или без учета содержимого флага переноса.
  + **Сдвиг влево** сдвигает биты операнда влево. Старший бит сдвигается во флаг переноса, а младший очищается.эквивалентен умножению на 2.
  + **Сдвиг вправо арифметический** сдвигает биты операнда вправо. Старший бит рассматривается, как знаковый, и остается неизменным, младший бит сдвигается во флаг переноса.эквивалентен делению на 2 числа со знаком.
  + **Сдвиг вправо логический** так же сдвигает биты операнда вправо. Но при этом старший бит сбрасывается в 0 , младший бит сдвигается во флаг переноса.эквивалентен делению на 2 числа без знака.
  + **Циклические сдвиги** (называемые иногда вращениями) отличаются от вышеописанных тем, что при сдвиге влево на один разряд выдвигающийся старший бит помещается в младший, а при сдвиге вправо – наоборот.

Все команды данной группы воздействуют на установку признака результата в регистре признаков.

* + Примеры:
    - * Idiv;
      * Imul;
      * Add;

17 Классификация машинных команд по функциональному назначению. Группа команд переходов и управления состоянием процессора. Примеры.

Процессоры различных типов компьютеров имеют индивидуальные наборы машинных команд, различающихся как внутренним форматом, так и количеством. Следует заметить, что классификация по различным признакам часто бывает взаимосвязанной, например, длина команды зависит как от методов адресации, так и от количества операндов; с другой стороны количество операндов может определяться, например, функциональным назначением команды и, опять же, методом адресации.

По функциональному назначению набор машинных команд процессора любого компьютера можно условно разбить на четыре основные группы:

* команды пересылки;
* арифметико-логические команды;
* команды переходов;
* команды управления состоянием процессора.

**Группа команд переходов** условно поделена на несколько подгрупп :

1. подгруппа команд безусловных переходов;
2. подгруппа команд условных переходов;
3. подгруппа команд организации циклов;
4. подгруппа команд организации подпрограмм;
5. подгруппа команд работы с прерываниями.

**Подгруппа команд безусловных переходов** может состоять из 1 однооперандной команды. В качестве операнды — адрес перехода, который записывается в счетчик адреса команды.

**Подгруппа команд условных переходов** обеспечивает ветвления в программе в зависимости от состояния регистра флагов.

В **подгруппе команд организации циклов** в качестве операндов указывается один из регистров общего назначения и адрес перехода. Команда организации цикла по счетчику эквивалентна двум командам: “вычитание из содержимого регистра 1” и “условный переход по ненулевому флагу результата”.

Выполнение **команды перехода к подпрограмме** вызывает, во-первых, запоминание текущего значения РС либо в специально отведенных ячейках памяти, либо в одном из регистров общего назначения, либо в верхушке стека, и, во-вторых, засылку в РС адреса первой выполняемой команды подпрограммы. Команда возврата из подпрограммы извлекает запомненный командой перехода к подпрограмме адрес возврата и записывает его в РС.

Аналогичным образом действуют **команды организации программного прерывания и возврата из прерывания**; существенное их отличие от команд организации подпрограмм состоит в том, что дополнительно запоминается состояние регистра признаков, которое при возврате из прерывания восстанавливается.

**Группа команд управления** состоянием процессора включает команды

* останова процессора,
* перевода процессора в состояние ожидания внешнего прерывания,
* разрешения и запрещения прерываний,
* сброса и установки битов регистра признаков.

Обычно команды этой группы не влияют на содержимое ячеек памяти и регистров общего назначения процессора, модифицируя только биты регистра признаков и внутренние флаги процессора.

18 Методы адресации памяти. Индексная, косвенная, индексно - относительная адресация, организация стековой памяти. Примеры.

Индексная адресация применяется в работе с элементами массивов. При индексной адресации адрес операнда вычисляется как сумма адреса начала массива и индексного смещения, определяющего смещение элемента массива относительно первого элемента, т.е. А = AN + (X), где (X) - содержимое индексного регистра. Индексное смещение хранится в специальном индексном регистре (могут использоваться РОН или специальные индексные регистры).

Для доступа к элементам массива индексное смещение рассчитывается на основе индекса (индексов) элемента массива по следующим формулам:

1. одномерный массив:
   * 1. **Х = (i - 1) \* Lэл**, где i - индекс элемента массива, а Lэл - длина элемента в байтах.
2. двумерный массив:
   * 1. **Х = ((i - 1) \* Jmax + j - 1) \* Lэл,**   
        где i - номер строки, j - номер столбца, Jmax - максимальное количество столбцов; при этом предполагается, что двумерный массив хранится в памяти по строкам, то есть так, что сначала последовательно располагаются элементы первой строки, затем второй и т.д.
3. трехмерный массив:
   * 1. **Х = ((i - 1) \* Jmax \* Kmax + (j - 1) \* Kmax + k-1) \* Lэл,**   
        где i, j, k - индексы элемента.

В реальных условиях часто используется **комбинированная индексно- относительная адресация**, при которой адрес элемента массива вычисляется как сумма трех величин А = (В) + (Х) + D.

В случае **косвенной адресации** в поле операндов команды указывается не адрес операнда, а адрес указателя на операнд, т.е. адрес регистра или ячейки памяти, содержащей адрес операнда. В цепочке указателей может быть более одного указателя. Количество указателей в цепочке - кратностью косвенной адресации, а адрес операнда, хранимый в указателе, называется косвенным адресом. В ходе выполнения программы адрес указателя может оставаться постоянным, а косвенный адрес может изменяться командами самой программы. Таким образом обеспечивается, например, обработка динамических данных сложной структуры, т.е. данных, память под которые запрашивается динамически, в процессе выполнения программы. Команды косвенной адресации выполняются дольше за счет дополнительного обращения к памяти по цепочке указателей, но иногда это позволяет резко повысить эффективность программы.

**Автоинкрементная и автодекрементная формы адресации** — развитие косвенной адресации. В автоинкрементной адресации после выполнения команды указатель автоматически увеличивается на определенную величину (слово или байт), а в автодекрементной - уменьшается перед выполнением операции. В этом смысле адресацию к верхушке стека с использованием регистра-указателя стека можно рассматривать, как частный случай **автодекрементной постфиксной**(при записи в стек) и **автоинкрементной префиксной**(при чтении из стека) адресации.

19 Методы адресации памяти. Абсолютная, непосредственная, относительная, регистровая адресация. Примеры.

**Прямая (абсолютная) адресация** предполагает, что в поле операндов в команде задан абсолютный (физический) адрес ячейки памяти, в которой расположен операнд.

Данный способ адресации является самым простым и достаточно универсальным, но есть несколько сложностей:

* сложность, а иногда и невозможность перемещения программы в памяти;
* низкая эффективность обработки массивов данных и данных сложной структуры.

Частный случай прямой адресации - **регистровая адресация**, поскольку в этом случае в поле операндов команды указывается абсолютный адрес регистра в блоке регистров общего назначения.

**Относительная адресация** обеспечивает перемещение программного модуля в памяти компьютера, т.е. позволяет создавать программы, способные работать в любых адресах памяти. В поле операндов команды указывается не абсолютный адрес операнда, а его смещение относительно начала или некоторой точки программы, называемой базовой точкой. Адрес этой точки программы загружается в специальный базовый регистр (его роль может выполнять один изРОН), адрес этого регистра также указывается в поле операндов команды. На этапе выполнения абсолютный адрес операнда вычисляется как сумма содержимого базового регистра В и смещения D, т.е. А = (В) + D.

В случае **непосредственной адресации** операнд записывается прямо в команде вместо адреса операнда. Подобный способ адресации значительно ускоряет выполнение команды, поскольку не требуется вычисление адреса операнда и его выборка из оперативной памяти.

20 Подпрограммы. Предпосылки использования, механизмы передачи и возврата управления. Примеры.

При разработке программ принято придерживаться концепции модульного программирования, позволяющее:

* -выделить в программе одинаковые фрагменты и оформить их в виде отдельного модуля и обращаться к нему, когда это необходимо, что позволяет значительно сократить объем программы.
* -гораздо легче спроектировать и отладить десяток небольших модулей, реализующих достаточно простые функции, а затем связать их между собой с помощью опять же небольшой управляющей программы, чем проектировать и отлаживать большую программу, созданную единым модулем, часто со сложной, сетевой структурой.
* -появляется возможность коллективной разработки программ коллективами программистов, при этом каждый из них выполняет отдельный, функционально автономный и самостоятельный модуль в соответствии с полученными спецификациями.

Модули подразделяются на главную (вызывающую) программу и вызываемую программу (подпрограмму). **Главная программа**, когда ей требуется произвести те или иные действия, обращается к подпрограмме. **Подпрограмма** выполняет требуемые действия, после чего продолжает выполняться главная программа, причем выполнение ее продолжается с точки, следующей за точкой обращения к подпрограмме.

В составе процессоров всех компьютеров имеются специальные команды, реализующие загрузку в программный счетчик адреса точки входа и запоминание адреса возврата. Реализация этих команд определяется архитектурой процессора и принятыми в данной архитектуре соглашениями о связях. Например, в System360/370 команда перехода к подпрограмме запоминает адрес точки возврата в одном из регистров общего назначения процессора, указанном в качестве операнда команды. В команде возврата из подпрограммы в качестве операнда должен быть указан тот же самый регистр. Например, архитектура процессоров семейства PDP для передачи управления подпрограмме предусматривает использование команды JSR R, ADR, которая обеспечивает выполнение следующих действий: R → (SP) – регистр связки помещается в стек РС → R – содержимое программного счетчика(в момент выполнения команды содержит адрес возврата) помещается в регистр связи ADR → PC – адрес точки входа в подпрограмму, указанный в команде, помещается в программный счетчик. При возврате управления подпрограммой с помощью команды RTS R выполняется обратная последовательность действий: R→PC - содержимое регистра связи, содержащее адрес возврата, помещается в программный счетчик; (SP)↑R - содержимое вершины стека извлекается в регистр связи. Аналогично этот процесс осуществляется в процессорах Intel. Для передачи управления используется команда CALL ADDR , которая обеспечивает, во-первых, запись текущего значения программного счетчика (в момент выполнения команды он содержит адрес возврата) в стек - PC↓(SP), во-вторых, запись адреса точки входа в подпрограмму (ADDR ) в программный счетчик - ADDR→PC. Для возврата управления в точку возврата используется команда RET, обеспечивающая извлечение из вершины стека адреса возврата и его запись в программный счетчик - (SP)↑PC.

21 Организация подпрограмм, механизмы передачи параметров. Примеры.

При передаче управления подпрограмме, в подавляющем большинстве случаев, ей передаются некоторые параметры, т.е. конкретные значения, с которыми оперирует алгоритм подпрограммы.

Параметры можно подразделить на входные параметры, и выходные. В подпрограмму можно передавать как конкретные значения некоторых переменных, так и адреса ячеек памяти, в которых эти значения расположены.

В первом случае мы имеем дело с передачей параметров по значению, а во втором - по ссылке. При передаче парaметров по значению, изменения их в подпрограмме не приводит к их изменению в основной программе. При рассмотрении внутренних механизмов передачи параметров можно выделить несколько способов передачи, а именно: \* передача параметров с использованием общей области памяти;\* передача параметров через регистры процессора;\* передача параметров через стек;\* передача параметров комбинированными способами, в том числе через использование таблицы адресов параметров. Передача параметров с использованием общей области памяти предполагает, что в оперативной памяти сформирована некоторая область памяти, доступная как главной программе, так и подпрограмме. Структура этой области жестко фиксирована и также известна программе и подпрограмме. Перед передачей управления подпрограмме главная программа помещает значения параметров в общую область, оттуда же извлекаются результаты работы подпрограммы. Подпрограмма, в свою очередь, извлекает из общей области переданные ей значения параметров и помещает в общую область результаты работы. Передача параметров с использованием общей области памяти предполагает, что в оперативной памяти сформирована некоторая область памяти, доступная как главной программе, так и подпрограмме. Структура этой области жестко фиксирована и также известна программе и подпрограмме. Перед передачей управления подпрограмме главная программа помещает значения параметров в общую область, оттуда же извлекаются результаты работы подпрограммы. Подпрограмма, в свою очередь, извлекает из общей области переданные ей значения параметров и помещает в общую область результаты работы. Передача параметров подпрограмме через общую область является достаточно простым и понятным способом, однако лищает ее универсальности - подпрограмма оказывается жестко привязанной к адресам и структуре общей области, что часто бывает нежелательным.Передача параметров через регистры процессора предполагает, что значения параметров (или их адресов) перед вызовом подпрограммы помещаются в регистры общего назначения процессора, выходные параметры подпрограмма так же передает через регистры, помещая в них значения перед возвратом управления. Передача параметров через регистры является стандартным способом в случае подпрограмм-функций, имеющих единственный входной и единственный выходной параметры. Таким образом реализуются, например, стандартные встроенные функции систем программирования PASCAL, C и ряда других. Это такие функции, как Sin(x), Cos(x), Abs(x), ln(x) и другие.Передача параметров через стек предполагает, что вызывающая программа перед обращением к подпрограмме помещает параметры в стек, при этом резервируется место для выходных параметров. После возврата управления подпрограммой главная программа обеспечивает очистку стека, извлекая из него значения входных и выходных параметров. Подпрограмма оперирует со значениями параметров либо непосредственно в стеке, используя адресацию относительно вершины стека, либо выталкивает значения параметров в регистры процессора, а затем использует. Если через стек передаются сами значения параметров, то реализуется передача по значению, в случае, если в стек записываются адреса параметров, реализуется механизм передачи параметров по ссылке.Способ передачи параметров с использованием таблицы адресов параметров позволяет передать множество параметров через единственный объект (регистр или ячейку стека). С помощью этого способа достаточно просто реализуется механизм передачи переменного числа параметров подпрограмме. В памяти организуется таблица (одномерный массив) элементами которой являются адреса параметров. Адрес этой таблицы перед передачей управления подпрограмме помещается в один из регистров процессора или в стек. Подпрограмма может получить доступ к параметрам, используя косвенно-регистровую адресацию кратности 2. В случае использования переменного числа параметров перед вызовом подпрограммы в первый элемент таблицы заносится количество параметров, использующееся в данном обращении к подпрограмме.

22 Прерывания, механизмы обработки прерываний, передача параметров программным прерываниям.

**Прерывание** - это временное прекращение выполнения процессором последовательности команд одной программы с целью выполнения другой, имеющей в данный момент времени более высокий приоритет.

**Внешние аппаратные прерывания** вызываются внешними по отношению к процессору устройствами компьютера и сигнализируют процессору о том, что он должен прервать выполнение текущей программы и заняться обслуживанием внешнего устройства. Это избавляет процессор от необходимости периодически опрашивать состояние внешних устройств, непроизводительно затрачивая на это время.

**Аппаратные прерывания** не координируются с работой программного обеспечения. Когда процессор распознает прерывание, он заканчивает выполнение текущей команды и переходит к процессу обработки прерывания, заключающемуся в выполнении специальной программы - обработчика прерывания. После того, как прерывание обработано, процессор продолжает выполнение прерванной программы с той точки, в которой она была прервана.

Для реализации механизма обработки прерываний в составе компьютера имеется специальный блок - **контроллер прерываний**. Контроллер прерываний обеспечивает прием запросов на прерывание от отдельных устройств (каждое устройство или однотипная группа устройств использует отдельную линию запроса прерывания), оповещение процессора о возникшем запросе на прерывание и формирование для него кода прерывания, в случае готовности процессора обработать прерывание. На основании сформированного кода прерывания, полученного от контроллера прерываний, процессор определяет адрес программы - обработчика прерываний и передает ей управление - начинается процесс обработки прерывания. Последней командой обработчика прерывания является команда возврата из прерывания, обеспечивающая возврат управления в прерванную программу.

Используется **приоритетная схема прерываний**: каждой линии прерывания присваивается определенный приоритет. В первую очередь осуществляется обработка запросов на прерывание от наиболее приоритетных устройств. Работа менее приоритетного обработчика может быть прервана c целью выполнения действий по обслуживанию более приоритетного устройства.

**Внутренние аппаратные прерывания**, в отличие от внешних, генерируются аппаратурой процессора и его окружением и свидетельствуют о нарушении нормального хода вычислительного процесса и возникновении нестандартной ситуации. Подобными ситуациями являются, например, деление на ноль, переполнение, несуществующий код операции, переполнении стека, попытках нарушения защиты памяти и ряд других.

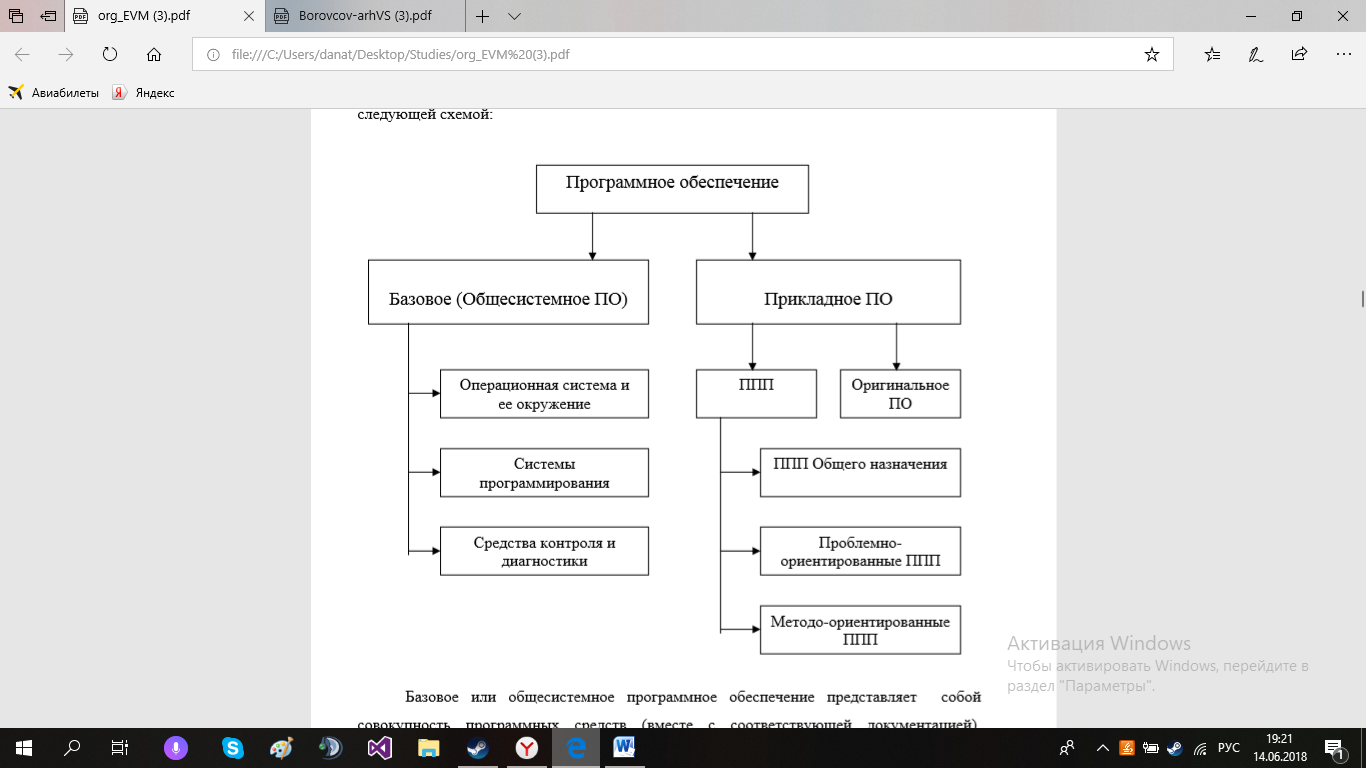
**Адреса программ - обработчиков прерываний** располагаются в защищенной системой области памяти, называемой областью векторов прерываний. Каждому типу прерывания соответствует свой обработчик прерывания. Сам обработчик прерывания может располагаться либо в ПЗУ (системном или периферийного контроллера), либо в ОЗУ.

Обработка прерываний может быть запрещена. **Запрещение прерываний** осуществляется на двух уровнях - на уровне контроллера прерываний и на уровне процессора. Запрещение прерываний, осуществляемое на уровне процессора, запрещает обработку прерываний от всех внешних устройств, при этом процессор просто игнорирует запрос на обработку прерывания, поступающий от контроллера прерываний и прерывания “теряются”. Запрещение прерываний, выполняемое на уровне контроллера, позволяет запретить обработку прерываний от отдельных внешних устройств путем “маскирования” отдельных линий запросов на прерывание. Естественно, что операция запрещения прерываний и маскирования прерываний имеет отношение только к внешним аппаратным прерываниям, и не оказывает никакого влияния на обработку внутренних.

**Программные прерывания** ничего не прерывают. Обращение к ним почти такое же, как и к подпрограммам. Они обеспечивают возможность работы с системными процедурами, адреса которых могут варьироваться в зависимости от версии системного программного обеспечения и конфигурации оборудования (например, различные типы видеоадаптеров могут иметь различный набор функций и реализующие их подпрограммы размещены по различным адресам). При модификации системного программного обеспечения и изменении конфигурации оборудования в этом случае нет необходимости изменять другие программы, использующие системные запросы, при условии, если не изменилось расположение векторов прерываний. Таким образом, механизм программных прерываний обеспечивает мобильность программного обеспечения, то есть его независимость от аппаратуры и расположения в памяти подпрограмм, реализующих системные функции.

**Немаскируемые прерывания** — аппаратные прерывания, немаскируемые обычным способом, через регистр маски контроллера прерываний. Обрабатываются всегда, вне зависимости от состояния запрета обработки прерываний. Служат для сигнализации неисправимых аппаратных сбоев или для целей отладки.

23. Классификация программного обеспечения.

Под **программным обеспечением** (ПО) понимается совокупность программ вычислительной системы, процедур и правил вместе со всей связанной с этими компонентами документацией, позволяющая использовать компьютер для решения различных задач. Состав и взаимосвязь различных компонентов ПО может быть представлена следующей схемой:

Базовое или общесистемное программное обеспечение представляет собой совокупность программных средств (вместе с соответствующей документацией), обеспечивающих автоматизацию трудоемких технологических этапов разработки программного обеспечения, а также для организации и контроля вычислительного процесса на компьютере в процессе его функционирования.Прикладное или специальное ПО – это совокупность программных средств, обеспечивающих решение той или иной задачи, либо круга задач из конкретной предметной области.

24. Операционная система. Назначение и функции.

Операционная система (ОС), по определению Г. Дейтела, специалиста в области операционных систем - это набор программ, обеспечивающих возможность использования аппаратуры компьютера. При этом аппаратные средства ЭВМ предоставляют «сырую» вычислительную мощность, а задача операционной системы заключается в том, чтобы сделать аппаратуру доступной, и по возможности удобной для пользователя.

Основное назначение операционной системы состоит в эффективном управлении ресурсами системы, основными из которых являются:

* процессор;
* память;
* устройства ввода/вывода.

Операционная система реализует множество функций, в том числе:

* обеспечивает взаимодействие между пользователем и вычислительной системой;
* обеспечивает разделение аппаратных средств между пользователями;
* планирует доступ пользователей к общим ресурсам;
* обеспечивает эффективное планирование и выполнение операций ввода/вывода;
* осуществляет восстановление информации и вычислительного процесса в случае сбоев и ошибок.

Иными словами, ОС представляет собой как бы «программную прослойку» между пользователями и аппаратурой ЭВМ.

25. Типы операционных систем. Режимы работы операционных систем.

ОС можно классифицировать по различным признакам, основными из которых являются:

* **1)** **режим работы** (диалоговый или пакетный);
  + - **Управление работой диалоговой** (интерактивной) ОС осуществляется с помощью команд, вводимых пользователем либо с клавиатуры терминала, либо с помощью других устройств ввода (мышь, тактильный экран, устройства речевого ввода и др.).   
      * + **Примерами диалоговых ОС** являются такие системы, как MS DOS, Windows, Mac OS, Unix и его разновидности, операционные системы мобильных систем Palm OS, Symbian OS, Linux DA и др.
    - **Управление работой пакетной ОС** осуществляется с помощью специального языка управления заданиями. Задание является внешней единицей работы такой ОС и представляет собой совокупность операторов языка управления заданиями, программы и исходных данных для программы.
    - Совокупность нескольких заданий образует **пакет (поток) заданий**. Пакет заданий поступает в систему и помещается во входные очереди. Из входных очередей задания для выполнения выбираются либо последовательно, либо на основе приоритетов. Задание может содержать один или несколько пунктов. При выборке из входной очереди очередного пункта задания и распределении ресурсов, необходимых для его функционирования, образуется задача, являющаяся внутренней единицей работы ОС. Результаты выполнения задания помещаются в выходные очереди ОС, откуда могут быть введены на периферийные устройства.
* **2)** **количество одновременно решаемых задач и обслуживаемых пользователей**.
  + - По количеству одновременно выполняемых задач и одновременно обслуживаемых пользователей можно выделить следующие разновидности\_ОС:  
      * + однопользовательские однозадачные;
        + однопользовательские мультизадачные;
        + многопользовательские\_мультизадачные.
    - В **однопользовательской однозадачной** операционной среде в каждый момент времени может работать только один пользователь, причем все ресурсы принадлежат только этому пользователю. Средства разделения и защиты ресурсов в такой среде отсутствуют, поэтому в системе в каждый момент времени может работать только одна\_задача.   
      * + **Примерами** таких операционных систем являются CP/M, MSDOS.
    - В **однопользовательской мультизадачной** операционной среде имеются средства диспетчеризации задач, механизмы планирования и управления доступом к разделяемым ресурсам, механизмы защиты. В каждый момент времени с такой системой может работать только один пользователь, однако у него есть возможность параллельного выполнения нескольких задач.   
      * + К подобным системам относятся, **например**, Windows 98/ME, Windows NT/2000 Workstation, Palm OS, Windows CE.
    - **Многопользовательские мультизадачные** операционные системы дополнительно снабжены средствами многотерминальной поддержки, то есть возможностью подключения множества физических или виртуальных терминалов для работы множества пользователей. Таким образом, в такой среде одновременно может работать несколько пользователей, причем каждый пользователь имеет возможность параллельного выполнения нескольких задач.   
      * + К этому классу относится семейство операционных систем UNIX, операционные системы Windows NT Server, Windows NT 2000 Server, QNX и ряд других.
    - **При реализации мультизадачного режима работы** все ресурсы вычислительной системы разделяются между множеством задачам, при этом механизмы диспетчеризации программ обеспечивают одновременное их выполнение (хотя на самом деле это далеко не так в случае однопроцессорной вычислительной системы). В общем случае в такой системе в определенные моменты времени системные ресурсы предоставляются то одной задаче, то другой и т.д.

26. Системы программирования. Классификация языков программирования.

Системы программирования обеспечивают полный цикл разработки, отладки и тестирования программ на различных языках программирования.

В общем случае системы программирования могут включать в себя следующий набор компонентов:

1. **язык программирования**;
2. **лингвистический процессор**, обеспечивающий преобразование конструкций языка программирования в машинный код;
3. библиотеки стандартных функций;
4. **средства компоновки** программ из различных модулей (редактор связей, Linker);
5. **средства отладки и тестирования программ** (отладчики, профилировщики);
6. **интегрированную среду разработки** (IDE).

В зависимости от изобразительных средств языки программирования традиционно классифицируются следующим образом:

1. **Машинные языки** (машинные коды) или языки уровня 0.
2. **Ассемблеры** - языки, в которых коды операций процессора заменены мнемоническими обозначениями, для ссылки на данные можно использовать символические адреса (имена), имеются способы описания распределения памяти. Лингвистические процессоры для таких языков часто также называют ассемблеры. Перевод в машинный язык осуществляется по принципу “один в один” - один оператор языка в одну команду машины.
3. **Языки высокого уровня**. Языки высокого уровня являются машинно-независимыми. В основе конструкций языка высокого уровня лежит ограниченный естественный или специализированный язык (например, язык математических обозначений и символов). Трансляция осуществляется по принципу “несколько - во множество”, то есть нескольким элементам языковой конструкции соответствует множество машинных команд. Транслятор с языка высокого уровня часто называют компилятором.
4. **Языки сверхвысокого уровня или генераторы программ**. В отличие от алгоритмических языков высокого уровня, описывающих процесс задачи в терминах алгоритма, описание задачи на языке сверхвысокого уровня формулирует цель решения задачи. По описанию задачи генератор формирует алгоритм решения и готовую рабочую программу.

27. Специальное (проблемное) программное обеспечение.

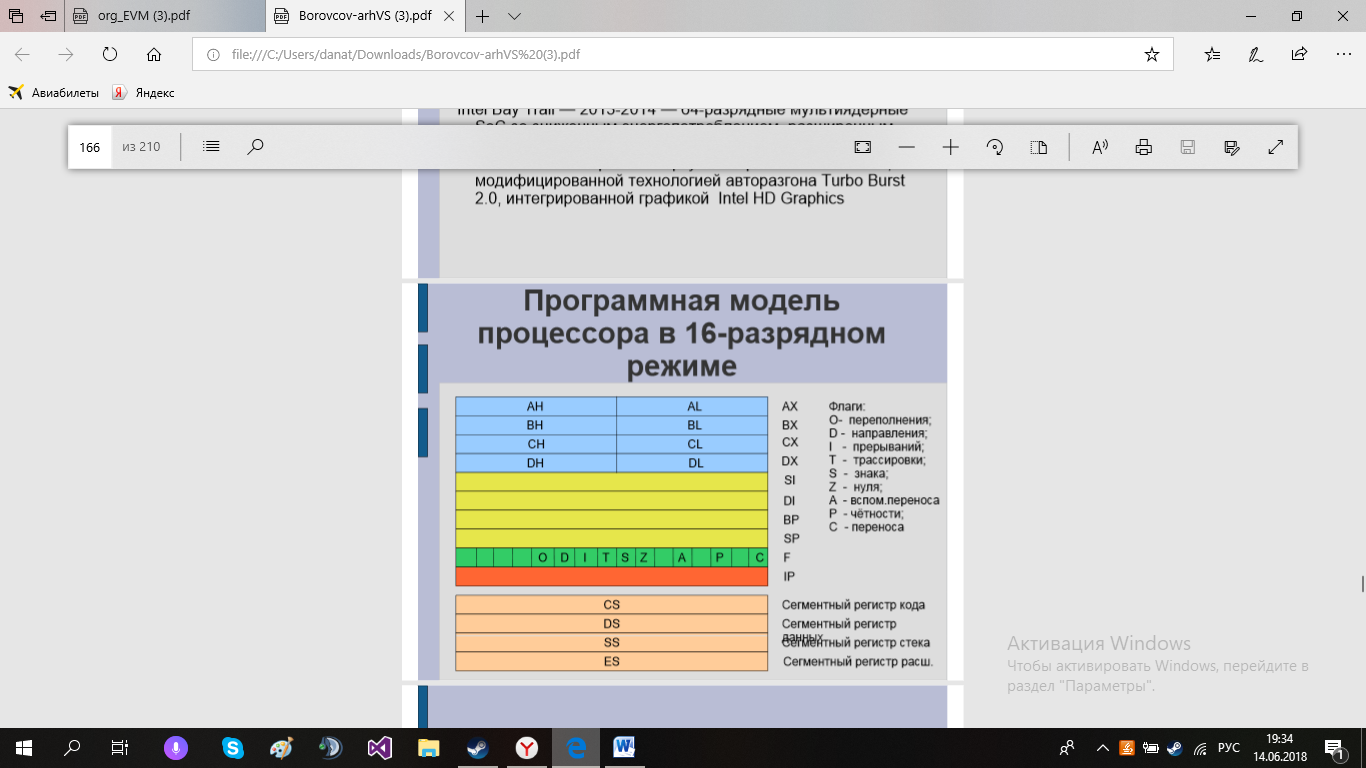
Проблемно-ориентированные пакеты используются для решения определенных задач в конкретной предметной области.

* Это, например, системы автоматизированного проектирования (САПР), предназначенные для решения задач проектирования радиоэлектронных схем и их компонентов, строительных и машиностроительных конструкций, проектирования интерьеров зданий и сооружений.

**Специальное программное обеспечение** представляет собой совокупность программ, используемых для решения определенного класса задач. **Специальное программное обеспечение** - это часть разрабатываемого при создании конкретной АСУ(Автоматизированная система управления) программного обеспечения, включающая программы реализации управляющих, информационных и вспомогательных функций (обеспечение заданного функционирования технических средств системы, проверка правильности ввода информации, контроль за работой системы и т.п.); оно разрабатывается на базе и с использованием программ общего программного обеспечения. Специальное программное обеспечение разработано на проблемно-ориентированном языке высокого уровня.

**Проблемно-ориентированное программное обеспечение** — это программные продукты, предназначенные для решения какой-либо задачи в конкретной функциональной области. Преимущество проблемно-ориентированного ПО состоит в том, что для автоматизации решения одной конкретной задачи может быть привлечено значительное число профессиональных программистов и специалистов в предметной области. Это дает возможность решить задачи очень большого объема, например комплексной автоматизации деятельности банковского учреждения со всеми филиалами и удаленными пунктами.

28. Базовая структура процессоров i80x86.Регистры процессоров i80x86. Назначение регистров.

Базовая модель микропроцессоров Intel80x86 - микропроцессор I8086/8088 имеет следующий набор регистров:   


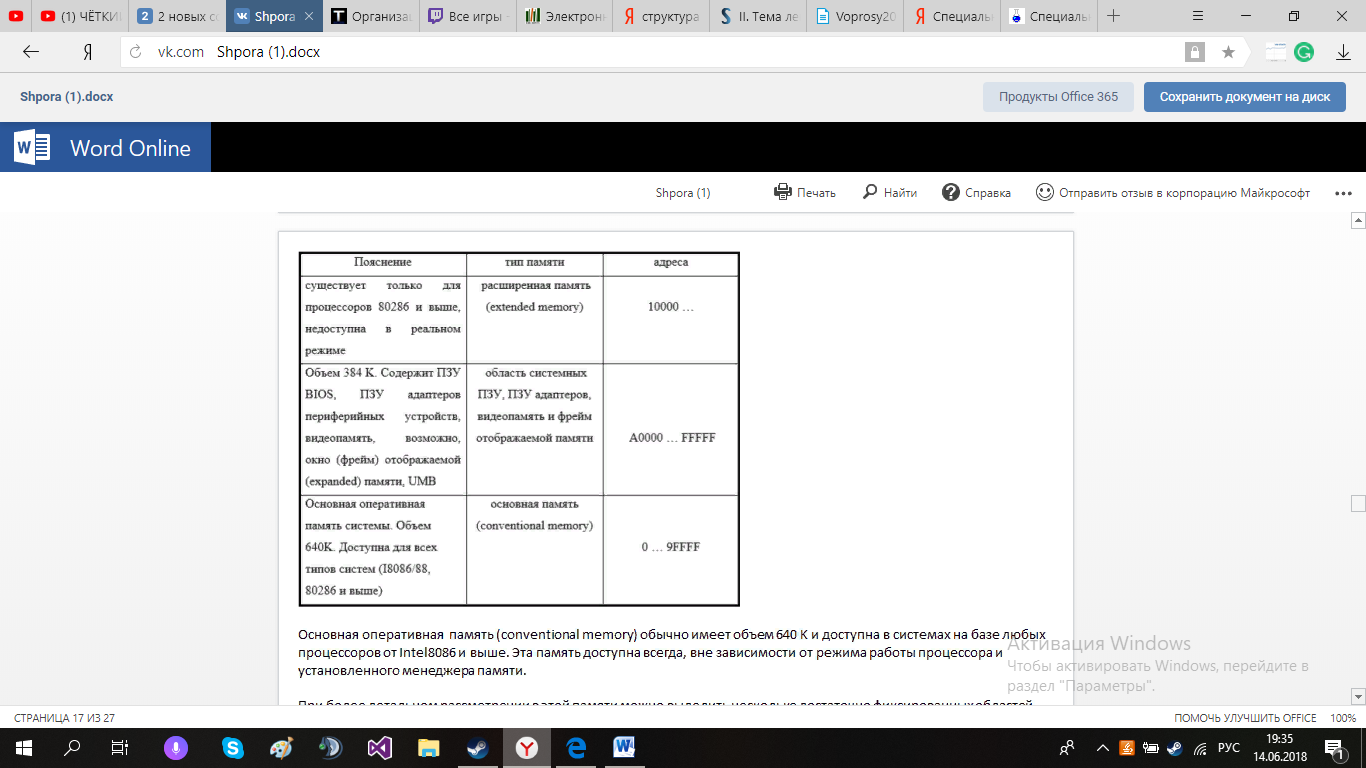
Все регистры являются шестнадцатиразрядными, однако каждый из регистров **AX, BX, CX, DX** может использоваться как два восьмиразрядных регистра.

Регистры **SP, BP, SI, DI** могут использоваться только в качестве шестнадцатиразрядных.

Регистры общего назначения процессора (первые восемь) могут использоваться в любых операциях с данными, кроме того, каждый из регистров имеет свои, строго определенные функции:

* **AX (AL)** выполняет роль, аналогичную аккумулятору процессоров I8080, I8085, Z80 - с использованием данного регистра выполняется умножение, деление, обмен с портами, преобразования, операции двоично-десятичной арифметики;
* **BX** выполняет роль базового регистра при использовании относительной и индексно-относительной адресации;
* **CX (CL)** выполняет роль счетчика в командах организации цикла и префиксах, используется при реализации сдвигов и вращений;
* **DX (DL)** используется в качестве расширителя регистра AX в операциях умножения и деления, а также при обмене данными с портами ввода/вывода в качестве регистра косвенной адресации;
* **SP** является шестнадцатиразрядным регистром указателя стека;
* BP используется в качестве базового при адресации относительно вершины стека, позволяя адресовать данные в окне стека (STACK FRAME);
* **SI, DI** используются в качестве индексных регистров и в строковых операциях.
* Шестнадцатиразрядный **регистр указателя инструкции IP** используется при выборке команд из оперативной памяти и является программно-недоступным, т.е. программист не может его изменить непосредственно, с помощью каких-либо команд.
* **Регистр флагов F** также является шестнадцатиразрядным, однако для хранения флагов, отражающих текущее состояние процессора и вычислительного процесса, используются только девять разрядов.

29. Распределение адресного пространства памяти системы. Использование блока основной (conventional) памяти в реальном режиме.

Распределение памяти компьютера IBM в случае работы под управлением операционной системы MS DOS выглядит следующим образом:  


Основная оперативная память (conventionalmemory) обычно имеет объем 640 K и доступна в системах на базе любых процессоров от Intel8086 и выше. Эта память доступна всегда, вне зависимости от режима работы процессора и установленного менеджера памяти.

При более детальном рассмотрении в этой памяти можно выделить несколько достаточно фиксированных областей (при работе в среде MS DOS):

- область векторов прерываний объемом 1K расположена в диапазоне адресов 00000 ... 003FF. Содержит адреса обработчиков прерываний. Адрес каждого обработчика представляется парой "сегмент:смещение" и занимает 4 байта. Таким образом, область векторов прерываний содержит 256 векторов прерываний;

- служебная область BIOS объемом 256 байт располагается с сегментного адреса 40 или с физического адреса 00400;

- служебная область DOS располагается с сегментного адреса 0050;

- ядро системы, содержащееся в файлах IBMBIO.COM (IO.SYS) и IBMDOS.COM (MSDOS.SYS) (размер зависит от версии системы. В случае, если произведена загрузка DOS в верхние адреса, эта область располагается в другом месте);

- область загружаемых драйверов устройств и системных буферов (также частично может быть перенесена в верхние адреса в системах с процессором 80286 и выше);

- резидентная часть командного процессора COMMAND.COM;- область пользователя (размер зависит от версии и конфигурации операционной системы);

- транзитная часть командного процессора COMMAND.COM.

30. Распределение блоков верхней памяти, свыше 640 К. Обеспечение свойства самоконфигурируемости векторов прерываний системы.

Область адресов памяти свыше 640 K имеет объем 384 K.В этой области также можно выделить несколько подобластей:

* + - - **область системного ПЗУ BIOS**; для PC/XT объем ПЗУ BIOS составляет 8 K. BIOS располагается в диапазоне адресов FE000 ... FFFFF. Для машин класса PC/AT обьем ПЗУ BIOS может достигать 64K и 128K, занимая при этом диапазон адресов F0000...FFFFF;
    - -**область видеопамяти**; в зависимости от дисплейного адаптера может располагаться в различных участках памяти. Для улучшенного графического адаптера EGA (EnhancedGraphicsAdapter) и адаптера VGA (VideoGraphicsArray) под видеопамять занимаются две области - для текстового режима - B8000...BDFFF (или B0000...B03FF в случае работы в монохромном текстовом режиме), для графического режима - A0000...AFFFF (64 K).Следует заметить, что в общем случае видеопамять адаптеров EGA и VGA может превышать объем 64 K, однако, поскольку эта память является страничной, то в общем адресном пространстве компьютера она все равно доступна через окно в 64 K;
    - - **ПЗУ дополнительных адаптеров** **периферийных устройств** в общем случае могут занимать несколько областей, например,
      * + ПЗУ BIOS EGA и VGA адаптеров обычно занимает адресное пространство в диапазоне C0000 ... C3FFF (EGA) или C0000 ... C7FFF (VGA);
        + ПЗУ BIOS жесткого диска может занимать адресное пространство C8000 ... CBFFF или D8000 ... DBFFF;
        + ПЗУ сетевых адаптеров располагается в диапазоне адресов DC000 ... DFFFF.
    - **Расширенная память** (ExtendedMemory) присутствует только в системах на базе процессора I80286 и выше. Обьем расширенной памяти для процессора 80286 может достигать 15 Mбайт, для процессоров 80386 и выше - 4 Гбайт. Память доступна только в защищенном режиме работы процессора за небольшим исключением 64 Кбайт с сегментного адреса FFFF. Этот блок памяти получил название "Область высших адресов" (HighMemoryArea или HMA). Эта область обычно занимается ядром операционной системы.
    - -**окно отображаемой памяти** (ExpandedMemory) обычно располагается в диапазоне адресов E0000 ... EFFFF, занимая область размером в 64 K (четыре страницы по 16Кбайт каждая). Отображаемая память имеет страничную организацию и позволяет через окно размером в 64K путем переключения страниц получить доступ к полю памяти, превосходящему размером 1M - т.е. то адресное пространство, которое способен адресовать процессор I8086/88.
    - - **блоки верхней памяти UMB** (UpperMemoryBlock) организуются на машинах с процессором 80286 и выше путем перемещения части Extended памяти в диапазон адресов до 1 Mбайта для увеличения объема памяти, доступной в реальном режиме. Эта память может быть использована для системных буферов, загружаемых драйверов устройств и резидентных программ.

31. Адресация памяти более 1 Мб в реальном режиме. Отображаемая память, область памяти высших адресов.

HighMemoryArea, HMA — начальный участок дополнительной памяти объёмом 65520 байт (64 килобайта минус 16 байт) с адресами от 10000016 до 10FFEF16 (сразу после UpperMemoryArea), доступный в реальном режиме через верхние сегменты адресного пространства.

Особенность процессоров 80286 и выше и построенных на них IBM PC/AT-совместимых компьютеров. Адресация памяти в процессорах 8086 и 8088 сегментная:

* полный (линейный) адрес памяти задаётся 16-битным номером сегмента («параграфа») и 16-битным смещением внутри этого сегмента (оба числа, как правило, записываются в шестнадцатеричной форме) и вычисляется по формуле: (сегмент) × 16 + (смещение).

C помощью сегментной адресации доступны линейные адреса от 0 (0000:0000) до 10FFEF16 (FFFF:FFFF, 1088 Кб). Однако адресная шина процессоров 808x 20-битная и может адресовать только 220 байт (1024 Кб или 1 Мб) памяти, то есть область с адресами от 0 до FFFFF16 (от 0000:0000 до FFFF:000F).

Адреса же от 10000016 до 10FFEF16 (от FFFF:0010 до FFFF:FFFF) — 21-битные, поэтому при их использовании возникает переполнение, 21-й бит теряется и, например, адрес 10000016 (FFFF:0010) указывает туда же, куда и адрес 0 (0000:0000).

В процессоре 80286 адресная шина 24-битная (возможна адресация 224 = 16 Мб памяти), поэтому в них переполнения не происходят. Компьютеры IBM PC/AT построены на процессоре 80286, но из соображений совместимости с IBM PC и IBM PC/XT, построенных на процессорах 808x, в них был введён логический элемент (вентиль), отключающий 21-й адресный провод (A20) от схемы управления памятью. Этот логический элемент (Gate A20 (англ. A20 line)) управляется через контроллер клавиатуры (микросхема Intel 8042) и по умолчанию он включён.

32. Организации видеопамяти для текстовых и графических режимов.

Распределение видеопамяти в графических режимах работы видеоадаптера отличается от распределения видеопамяти в текстовых режимах. Это вызвано тем, что в графических режимах необходимо хранить информацию о каждом пикселе изображения.

Часть адресного пространства старшей памяти отводится для адресации к графическому и текстовому видеобуферам графического адаптера. Графический адаптер представляет собой отдельную микросхему, в состав которой входит собственное запоминающее устройство (видеопамять).

Схемы управления видеопамятью настроены на диапазоны адресов A0000h...AFFFFh и B8000h...BFFFFh, входящих в общее с памятью адресное пространство процессора. Поэтому любая программа может обратиться по этим адресам и, например, записать данные в видеобуфер, что приведет к появлению на экране некоторого изображения. Если видеосистема находится в текстовом режиме, а запись осуществляется по адресам текстового видеобуфера, на экране появятся изображения тех или иных символов (букв, цифр, различных знаков). Если же перевести видеосистему в графический режим, и записывать данные в графический видеобуфер, то на экране появятся отдельные точки или линии. Можно также прочитать текущее содержимое ячеек видеобуфера. Для того чтобы изображение появилось на мониторе, оно должно быть записано в память видеоадаптера. В текстовом режиме для VGA-совместимых систем для видеопамяти отводится адресное пространство начинающееся с логического адреса B800h:0000h и заканчивающееся адресом BF00h:0FFFh.Данная область разбивается на 8 секторов по числу видеостраниц (4 Кбайта на страницу). Постраничное деление адресного пространства видеопамяти в текстовом режиме имеет следующий вид:

– B800h:0000h – страница 0, смещение в диапазоне 0000h – 0FFFh

– B900h:0000h – страница 1, смещение в диапазоне 0000h – 0FFFh

– ...........

– BF00h:0000h – страница 7, смещение в диапазоне 0000h – 0FFFh

Выделяют два вида режимов работы видеоадаптера, отличающиеся способом формирования видеосигнала: графический и текстовый. Для монитора оба режима одинаковы.

Графический режим: Для графического Gr (Graphics) или АРА (AllPointsAddressable – все точки адресуемы) режима видеоадаптера характерна возможность индивидуального управления каждым пикселом (точкой или триадой) выводимого на экран изображения. Каждому выводимому пикселу соответствует двоичный код, хранимый в ячейках видеопамяти. Кодированием задают количество цветов, градаций яркости или таких атрибутов, как мерцание, инверсия и др. Для кодирования пиксела используется п битов, которые определяют число возможных состояний пиксела. При выводе цветных изображений число п бит/пиксел называют глубиной цвета. При этом количество отображаемых цветов равно 2п, а размер кадрового буфера видеопамяти, необходимый для хранения цветного изображения с разрешением N х М и глубиной цвета п бит/пиксел, составит N×M×nбит.

В настоящее время используются два цветовых режима:

HighColor с глубиной цветов:

– 15 бит/пиксел (215 = 32 768 цветов) и распределением битов между базовыми цветами (красным, зеленым и синим) R : G : В = 5 : 5 : 5,

– 16 бит/пиксел (65 536 цветов) и распределением битов между базовыми цветами 5:6:5 или 6:6:4;

• TrueColor (истинный цвет) с глубиной цветов 24 бит/пиксел (16 777 216 цветов) и распределением битов между базовыми цветами 8:8:8.Обычно в видеопамяти хранится полный кадр образа изображения, подготовленного центральным процессором. Объем памяти для хранения одного кадра образа изображения называется страницей. При большом объеме видеопамяти она разбивается на страницы, т.е. может содержать несколько кадровых буферов.Ячейки памяти сканируются с частотой кадровой развертки, и за один цикл кадр выводится на экран. Процесс постоянного сканирования видеопамяти называется регенерацией изображения. Принцип образования цветовой палитры состоит в том, что путем включения/выключения лучей с одинаковым уровнем яркости при засветке триады основных цветов (красного, зеленого и синего) возможно создание палитры из 23 = 8 цветов. Если дополнительно использовать k уровней яркости для каждого цвета, то количество цветов палитры возрастает до 23к. Кодируемый уровень каждого базового цвета с помощью ЦАП переводится в аналоговый сигнал, который подается на модулятор ЭЛТ.

В режимах HighColor и TrueColor используется линейная организация видеопамяти, при которой для отображения цвета каждого пиксела на экране задействовано 2 или 3 байта. Растровый формат хранения изображения, при котором пикселы образа изображения на экране отображаются на биты ячеек памяти, называется битовой картой (BitМар). Битовая карта формируется в видеопамяти графического адаптера под управлением программы, исполняемой центральным процессором. На первом этапе процессор готовит порцию данных, а на втором – передает их в видеопамять по каналу связи. Передача данных от процессора возможна только в те моменты, когда в монитор из видеопамяти не выводятся биты для регенерации изображения. В противном случае данные от процессора попадут на экран и создадут помехи в виде "снега".

Существует несколько путей решения этой проблемы: • расширение разрядности шин каналов связи процессор-видеопамять и видеопамять-монитор, что позволит увеличить количество передаваемых бит за один цикл;

• использование кэша, что позволит повысить скорость формирования битовой карты;

• использование второго кадрового буфера в видеопамяти, что позволит разделить операции записи и считывания данных;

• использование двухпортовой видеопамяти, что позволит одновременно записывать и считывать данные;

• использование в видеоадаптере собственного специализированного процессора, способного формировать в видеопамяти битовую карту по командам из центрального процессора.

Графический режим является основным режимом работы видеосистемы современного персонального компьютера, поскольку в нем на экран монитора можно вывести текст, рисунок, фотографию, анимацию или видеосюжет. Для эффективной работы в графическом режиме требуются значительный объем видеопамяти и высокопроизводительный компьютер.

33. Базовая система адресации процессоров I80x86. Адресация в реальном режиме. Методы адресации.

Для процессоров I8086/88 и реального режима процессоров I80286/386 и выше допустимыми являются следующие виды адресации:

* регистровая адресация;
* непосредственная адресация;
* прямая адресация;
* косвенная регистровая адресация;
* базовая адресация;
* индексная адресация;
* базовая индексная адресация;
* относительная адресация;
* неявная адресация.

**Регистровая адресация** предполагает, что операнд находится в одном из регистров общего назначения, причем регистр может быть как восьми -, так и шестнадцатиразрядный. Специфичных особенностей не имеется.

**Непосредственная адресация** предполагает задание в поле операнда команды восьми- или шестнадцатиразрядного непосредственного операнда. В случае двухоперандной команды обязательно соответствие длин операндов. Частным случаем данного вида адресации является длинная прямая адресация, при которой в поле операнда команды содержится четыре байта, указывающие абсолютный адрес памяти в формате 70 "сегмент: смещение", однако подобная форма адресации допустима только в командах межсегментных (далеких) переходов и вызовах подпрограмм. Другая форма прямой адресации допускается в командах обмена с портами, когда в качестве операнда указывается абсолютный адрес порта.

При **косвенно-регистровой адресации** адрес операнда может располагаться в одном из шестнадцатиразрядных регистров общего назначения, т.е. регистрах BX, BP, SI, DI. Разновидностью данного способа адресации является косвенная адресация портов ввода/вывода через регистр DX.

**Базовая адресация** предполагает, что в качестве операнда в команде указывается имя базового регистра и смещение. Исполнительный адрес в этом случае указывается как сумма базового регистра и смещения. В качестве базового регистра могут использоваться регистры BX и BP. Причем, если в качестве базового регистра указывается BX, выборка операнда осуществляется из сегмента данных, адресуемого сегментным регистром DS, в случае указания регистра BP выборка операнда осуществляется из стека.

**Индексная адресация** предполагает, что значение адреса вычисляется как сумма смещения и содержимого индексного регистра SI или DI, указанных в команде. Базовая индексная адресация является комбинацией чисто индексной и базовой адресации, адрес вычисляется соответствующим образом.

**Относительная** адресация реализуется только по отношению к регистру IP и предполагает, что адрес вычисляется как сумма содержимого регистра IP и смещения. Данная форма адресации используется в командах организации переходов, циклов и вызова подпрограмм, обеспечивая независимость соответствующих команд от места размещения в памяти.

**Неявная форма** адресации используется в командах умножения, деления, преобразования и перекодировки, в которых в качестве операнда всегда предполагается регистр AX (или пара DX:AX), однако в качестве операнда он не указывается. Кроме того, подобная форма адресации используется в строковых операциях, в которых для адресации применяются регистры SI и DI, также не указываемые явно в команде.

34. Организация системы прерываний семейства IBM PC.

**Внутренние аппаратные прерывания** (исключения) вызываются аппаратурой процессора, имеют номера в диапазоне 0 ... 31 и обеспечивают реакцию на следующие ситуации:

* **0** **-** **деление на ноль**; вызываются инструкциями деления DIV и IDIV в случае нулевого делителя;
* **1** **- пошаговый режим**; прерывание возникает после выполнения каждой команды при установленном флаге трассировки T;
* **2** **-** **немаскируемое прерывание**; генерируется процессором при поступлении сигнала на вход NMI;
* **3** **- прерывание по точке останова**; генерируется в случае, если код очередной команды - CC; используется отладчиками;
* **4 -** **переполнение**; генерируется в случае, если установлен флаг переполнения O и выполняется инструкция INTO;
* **5 -** **печать экрана** (PrintScreen); вообще говоря, является внешним, однако традиционно имеет фиксированное назначение и располагается в области внутренних прерываний;
* **6** (80286 и выше) - **неверный код операции**;
* **7** (80286 и выше) - **математический сопроцессор недоступен**; вызывается при попытке выполнить команду сопроцессора при его отсутствии; может использоваться для эмуляции сопроцессора;
* **8** (80286 и выше) - **обнаружена двойная исключительная ситуация**; вызывается в случае возникновения более чем одной исключительной ситуации при выполнении очередной команды; в защищенном режиме вызывается при попытке обратиться за установленный предел таблицы векторов прерываний;
* **9** (80286 и выше) - **попытка обращения к памяти за пределами границы сегмента;**
* **A** (80286 и выше) - **неверный сегмент состояния задачи**;
* **B** (80286 и выше) - **сегмент не найден**; возникает при отсутствии требуемого сегмента в памяти в защищенном режиме;
* **C** (80286 и выше) - **стек переполнен или неверное значение указателя стека**;
* **D** (80286 и выше) - **общее нарушение защиты памяти**;
* **E** (виртуальный режим 80386 и выше) - **используется для реализации виртуальной памяти;**
* **10** (80286 и выше) - **ошибка сопроцессора**.

Внешние аппаратные прерывания вызываются внешними по отношению к процессору устройствами.

Для управления аппаратными прерываниями во всех компьютерах IBM PC используется программируемый контроллер прерываний, обеспечивающий обработку прерываний с учетом их приоритетов. IBM PC/XT имеет 8 уровней прерываний, IBM PC/AT - 16, так как использует два контроллера прерываний, включенных каскадно. Запросы на аппаратные прерывания 0 - 7 соответствуют векторам прерываний от 8 до 0F, для IBM PC/AT запросам на аппаратные прерывания соответствуют вектора прерываний 70 ... 7716.

Приоритет аппаратных прерываний следующий:

* + - 0 - таймер;
    - 1 - клавиатура;
    - 2 - прерывания от платы расширения (PC,PC/XT);
    - управление вторым контроллером прерываний(AT);
      * + 8 - часы реального времени;
        + 9 - программно переводится в 2;
        + 10 - резерв;
        + 11 - резерв;
        + 12 – PS/2 Mouse;
        + 13 - математический сопроцессор;
        + 14 - контроллер жесткого диска EIDE/ATA I канал;
        + 15 – контроллер жесткого диска EIDE/ATA II канал;
    - 3 - прерывания от COM1 (COM2 для AT);
    - 4 - прерывания от COM2 (COM1 для AT);
    - 5 - жесткий диск (PC,PC/XT), принтер 2(LPT2)(AT) или звуковая карта (AT);
    - 6 - контроллер гибких дисков;
    - 7 - принтер (LPT1).

Высший приоритет имеет аппаратное прерывание 0 - прерывание от таймера, низший - 7 - прерывание от принтера.

**Программные прерывания.** Программные прерывания вызываются командой процессора INT n, где n - номер программного прерывания. Программные прерывания можно условно разбить на четыре группы:

\* прерывания BIOS;

\* прерывания DOS;

\* программные прерывания пользователя;

\* программные прерывания интерпретатора BASIC;

Прерывания BIOS устанавливаются при включении машины либо при полной перезагрузке (COLD BOOT) инициализирующей процедурой BIOS(сокращенный набор прерываний BIOS приведен в приложении 4). Прерывания DOS устанавливаются при загрузке ядра операционной системы и загружаемых драйверов устройств. Прерывания интерпретатора BASIC используются кассетным БЕЙСИКом и его дисковыми версиями при их функционировании. Прерывания пользователя могут использоваться программами по усмотрению программиста. В принципе, любое прерывание, как программное, так и аппаратное, может быть дополнено, модифицировано или подменено путем замены в таблице векторов прерываний соответствующего вектора и написания собственной процедуры обработки прерывания, на которую этот вектор указывает.

Прерывания BIOS

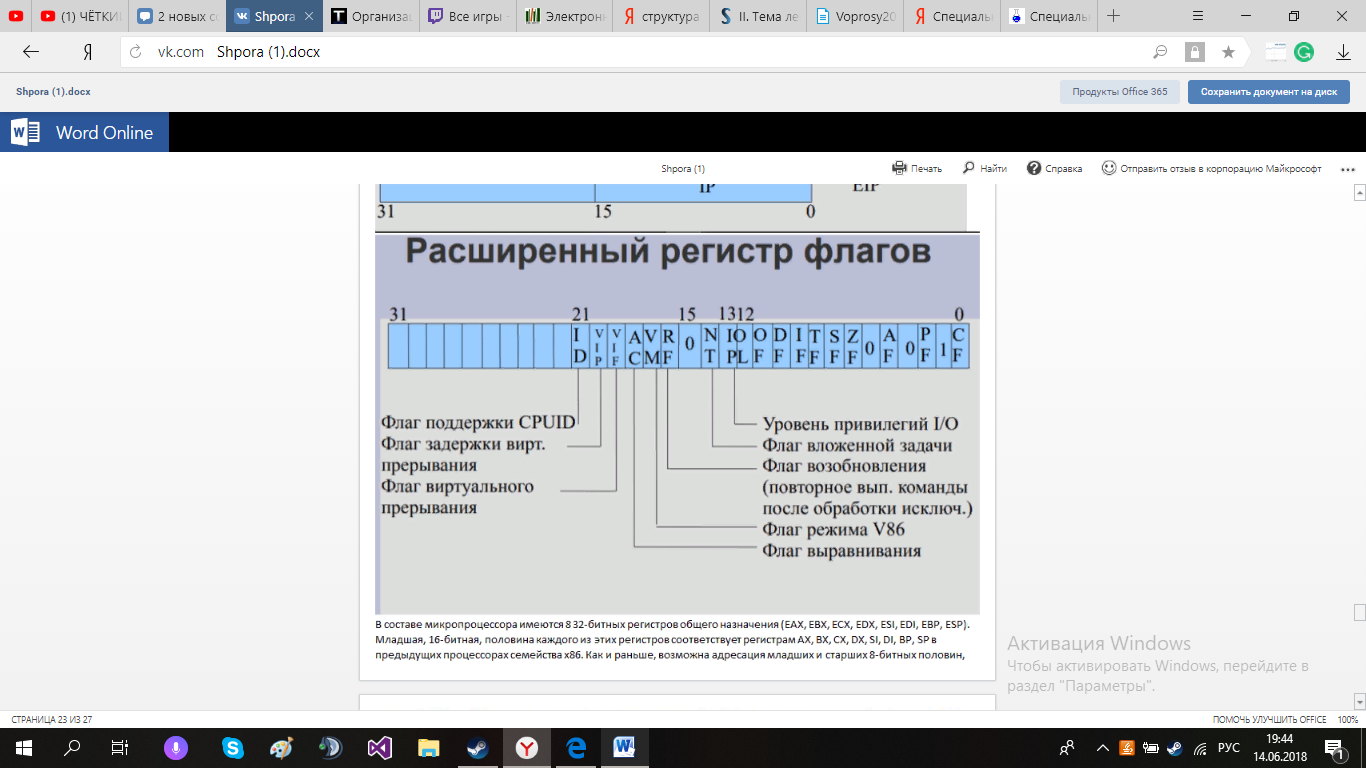
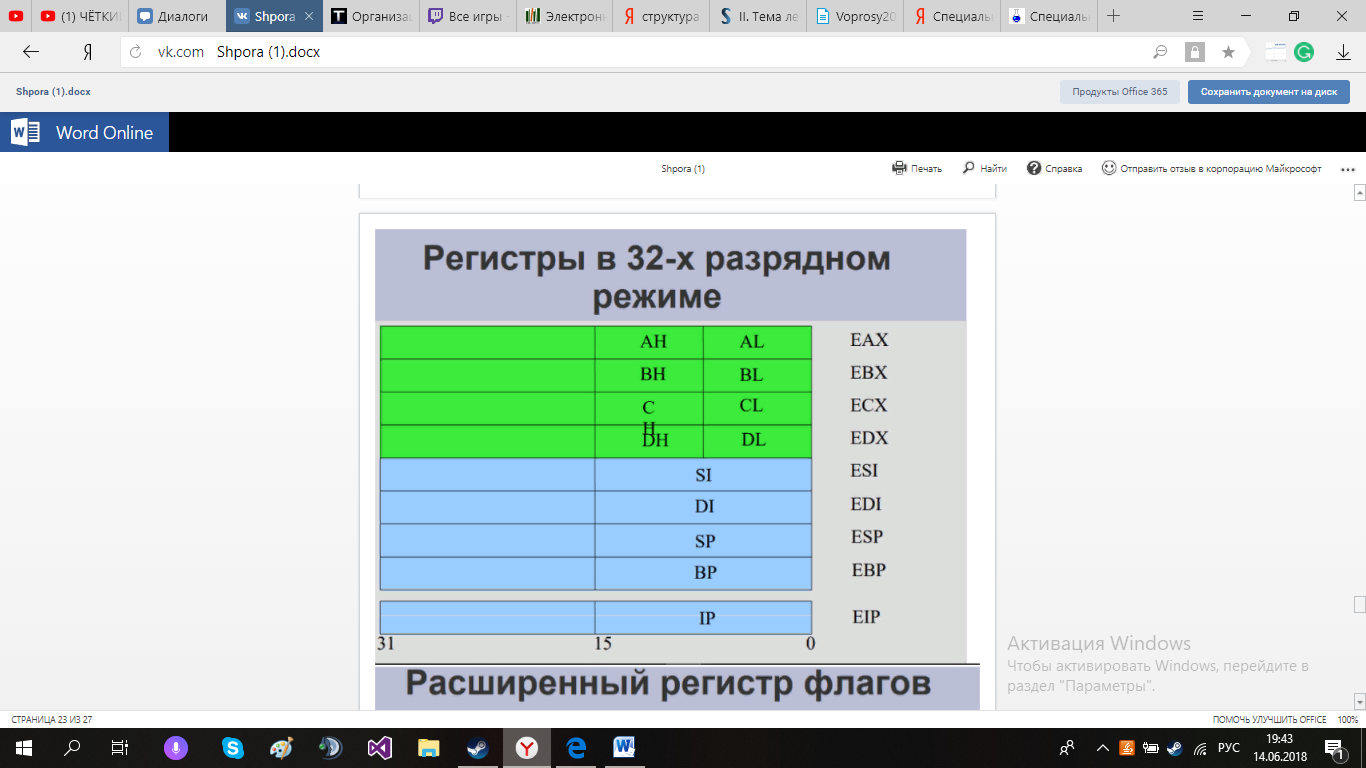
* 10 операции ввода/вывода на экран видеомонитора
* 11 определение конфигурации оборудования
* 12 определение объема оперативной памяти
* 13 обмен с дисками
* 14 организация обмена по последовательным портам
* 15 ввод/вывод для накопителя на магнитной ленте кассетного типа (устаревшая), работа с XMS памятью
* 16 ввод с клавиатуры
* 17 обслуживание принтера
* 18 инициализация системы кассетного БЕЙСИКа
* 19 загрузка операционной системы
* 1A функции даты и времени
* 1B обработчик прерывания Ctrl-Break
* 1C "заглушка" для модификации прерывания по таймеру
* 1D адрес таблицы видеопараметров
* 1E адрес таблица параметров дискеты
* 1F адрес второй половины таблицы знакогенератора для графического режима

35. Краткая хаpактеpистика(ПЕРЕДЕЛАТЬ СО СЛАЙДОВ)

микpопpоцессоpов семейства I80x86 и их клонов.

* 1971 Intel 4004 – 4-x разрядный, 4к памяти программ, 640Б памяти данных
* 1972 Intel 8008 – 8-ми разрядный, 64к памяти
* 1978 Intel 8086 – 16 разрядный, 16 разрядная шина данных, 20 – разраядная шина адреса (1М RAM). Основоположник линейки 80х86
* 1979 – Intel 8088 – 16 разрядный, 8 ми разрядная мультиплексированная шина данных, 20 разрядная шина адреса (1М RAM)
* 1980 – Intel 8087 математический сопроцессор для 8086 и 8088
* 1982 Intel 80286 — 16-ти разрядный, 24- х разрядная шина адреса шина адреса, адресуемая адресуемая память - 16 Гб. Два режима работы — реальный и защищённый. В защищённом режиме аппаратная поддержка мультизадачного режима. Сопроцессор — 80287
* 1985 – Intel 80386 Dx-32-x разрядный. 32х разрядная шина данных и 32-х разрядная шина адреса. Адресное пространство 4Гб. Три режима работы: реальный режим, защищенный и режим виртуального 8086. В рамках защищенного режима. Сегментированная и плоская модели памяти. Улучшение мультизадачности. Защиты и виртуальной памяти
* 1994 – IntelPentium – 64-х раздраядная шина данных,, 32ъ раздрядная шина памяти, разделенный L1-кэш для инстурукции и данных (8к+8к), суперскалярная архитектура, позволяющая выполнять 2-е инструкции одновременно
* 1999 - IntelPentium ||| - увеличение кол-ва вычислительных блоков, появление SSE, позволяющее выполнять SIMD-операции над данными с плавающей точкой одинарной точности(32бит)
* 2002 – Intel Pentium 4 ( northwood) – MMX, SSE, SSE2, HyperTrading(HT)
* 2008 - — 64-х битный, двухядерный, поддержка наборовкоманд MMX,SSE,SSE2,SSE3,SSSE3,SSE4.1, EMT64, EIST, XD-bit, VT-x;
* 2011 Intel Core i3,i5,i7 Dual/Quad-Core (ядро Sandy Bridge) — 64- битный, интегрированное графическое ядро, интегрированный контроллер памяти, поддерживаемыенаборыинструкцийитехнологии:MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, AVX, Enhanced Intel SpeedStep Technology (EIST), Intel 64, XD bit (an NX bit implementation), TXT, Intel VT-x, Intel VT-d, Hyperthreading Turbo Boost AES threading, Turbo Boost, AES -NI Smart Cache

36. Расширения процессоров 80386 и выше для 32-х и 64-х разрядного режима, регистры процессоров, их назначение.



В составе микропроцессора имеются 8 32-битных регистров общего назначения (EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP). Младшая, 16-битная, половина каждого из этих регистров соответствует регистрам AX, BX, CX, DX, SI, DI, BP, SP в предыдущих процессорах семейства x86. Как и раньше, возможна адресация младших и старших 8-битных половин, младшей 16-битной половины 32-разрядных регистров данных (AL и AH, BL и BH, CL и CH, DL и DH). Регистр указателя команды и регистр флагов также стали 32-битными (EIP и EFLAGS соответственно), причём в регистре флагов добавлены новая группа флажков. К 4 16-битным сегментным регистрам (ES, CS, SS, DS) прибавились ещё 2 16-битных регистра (FS и GS), теперь стало 6 сегментных регистров. Добавлены несколько новых групп регистров (каждый регистр размером 32 бит): 3 регистра управления (CR0 (MSW), CR2, CR3), 8 регистров отладки (DR0, DR1, DR2, DR3, DR6, DR7), 2 тестовых регистра (TR6, TR7).

37. Защищенный режим процессоров 80х86. Организация памяти в защищенном режиме.

Суть защищённого режима в следующем: программист и разрабатываемые им программы используют логическое адресное пространство, размер которого может составлять 1 гигабайт.

Логический адрес преобразуется в физический адрес автоматически с помощью схемы управления памятью (MMU). При этом содержимое сегментного регистра не связано напрямую с физическим адресом, а является номером сегмента в соответствующей таблице.

Благодаря защищённому режиму, в памяти может храниться только та часть программы, которая необходима в данный момент, а остальная часть может храниться во внешней памяти (например, на жёстком диске). В случае обращения к той части программы, которой нет в памяти в данный момент, операционная система может приостановить программу, загрузить требуемую секцию кода из внешней памяти и возобновить выполнение программы. Следовательно, становятся допустимыми программы, размер которых больше объёма имеющейся памяти, и пользователю кажется, что он работает с большей памятью, чем на самом деле. Физический адрес формируется следующим образом.

В сегментных регистрах хранится селектор, содержащий индекс дескриптора в таблице дескрипторов (13 бит), 1 бит, определяющий к какой таблице дескрипторов будет производиться обращение (к локальной или к глобальной) и 2 бита запрашиваемого уровня привилегий. Далее происходит обращение к соответствующей таблице дескрипторов и соответствующему дескриптору, который содержит начальный 24-битный адрес сегмента, размер сегмента и права доступа, после чего вычисляется необходимый физический адрес путём сложения адреса сегмента со смещением из 16-разрядного регистра.

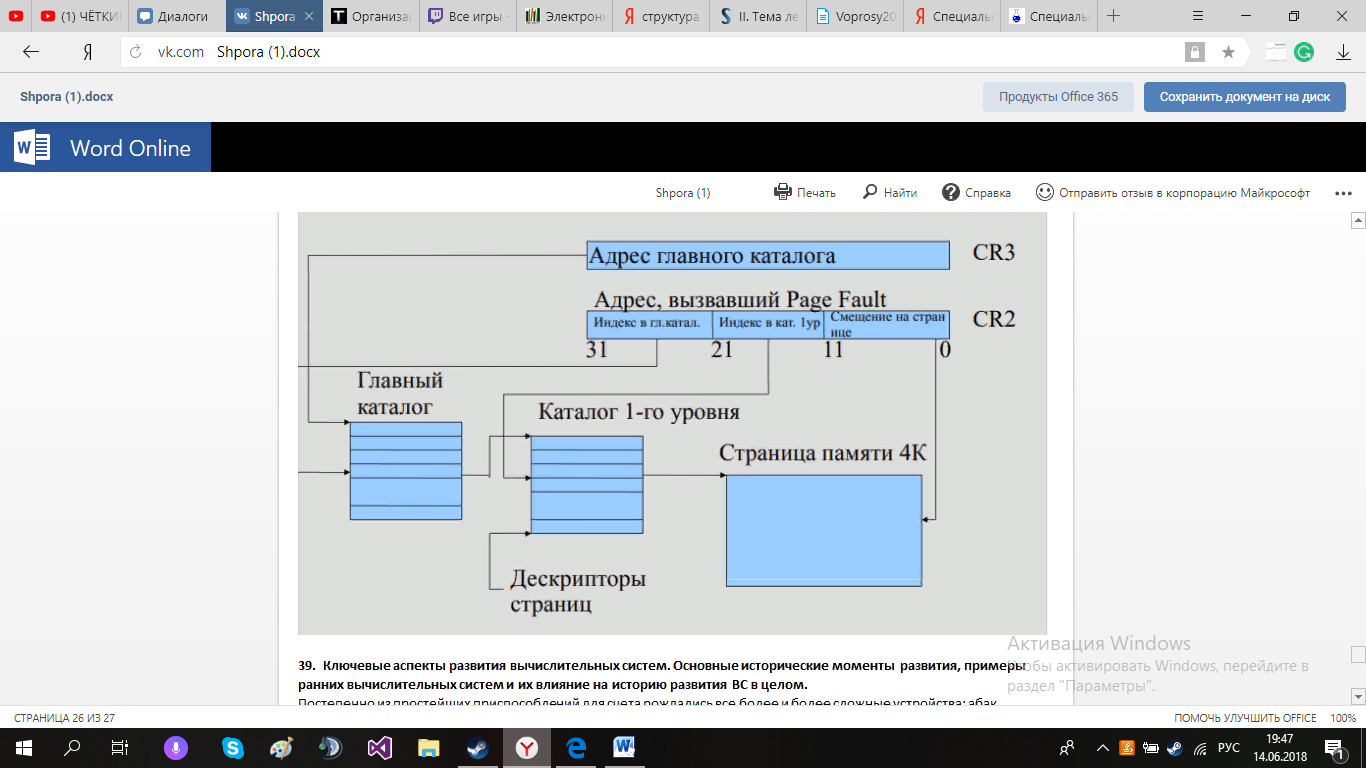
Вся физическая память делится на страницы фиксированного размера (4 КБайт, 2 МБ или 4 МБ, в x86\_64 также 1 ГБ). Каждая страница, независимо от размера, выравнена по границе 4 КБайт.

Таблица прерываний глобальна. Размещение в физической памяти определяется регистром IDTR. При возникновении прерывания (внешнего, аппаратного, или вызванного инструкцией Int): из IDT выбирается дескриптор шлюза, соответственно номеру прерывания; проверяются условия защиты (права доступа); при соблюдении условий защиты выполняется переход на подпрограмму-обработчик этого прерывания. В отличие от GDT, LDT может быть много (соответственно количеству задач (потоков), но не обязательно). Каждая задача может иметь свою. На расположение таблицы текущей задачи указывает регистр LDTR. Размер и расположение LDT в линейной памяти определяются дескриптором LDT из GDT (но это не означает, что размер LDT может быть больше 65536 байт). Первый дескриптор LDT (№ 0) использовать можно. Глобальная дескрипторная таблица является общей для всех процессов. Её размер и расположение в физической памяти определяются регистром GDTR. Размер таблицы не может превышать 8192 дескрипторов, поскольку один дескриптор занимает 8 байт, а лимит в регистре GDTR — двухбайтный и хранит размер таблицы минус один (максимальное значение лимита — 65535), а 8192 x 8 = 65536. Дескрипторы LDT и сегментов задач (TSS) могут находиться только здесь. Особенностью GDT является то, что у неё запрещён доступ к первому (с нулевым смещением относительно начала таблицы) дескриптору. Обращение к нему вызывает исключение #GP, что предотвращает обращение к памяти с использованием незагруженного сегментного регистра.

38. Понятие об организации виртуальной памяти и страничной адресации. Реализация страничной памяти в микропроцессорах семейства 80х86.

Виртуа́льная па́мять (англ. virtualmemory) — метод управления памятью компьютера, позволяющий выполнять программы, требующие больше оперативной памяти, чем имеется в компьютере, путём автоматического перемещения частей программы между основной памятью и вторичным хранилищем (например, жёстким диском)[1][2][3].

Для выполняющейся программы данный метод полностью прозрачен и не требует дополнительных усилий со стороны программиста, однако реализация этого метода требует как аппаратной поддержки, так и поддержки со стороны операционной системы. В системе с виртуальной памятью используемые программами адреса, называемые виртуальными адресами, транслируются в физические адреса в памяти компьютера. Трансляцию виртуальных адресов в физические выполняет аппаратное обеспечение, называемое блоком управления памятью. Для программы основная память выглядит как доступное и непрерывное адресное пространство либо как набор непрерывных сегментов, вне зависимости от наличия у компьютера соответствующего объёма оперативной памяти. Управление виртуальными адресными пространствами, соотнесение физической и виртуальной памяти, а также перемещение фрагментов памяти между основным и вторичным хранилищами выполняет операционная система Реализация страничной памяти:

Страничная память - способ организации виртуальной памяти, при котором единицей отображения виртуальных адресов на физические является регион постоянного размера (т. н. страница).Типичный размер страницы — 4096 байт, для некоторых архитектур — до 128 КБ  
Основная мысль сводится к формированию таблиц описания памяти, которые определяют состояние её отдельных сегментов/страниц и т. п. При нехватке памяти операционная система может выгрузить часть данных из оперативной памяти на диск, а в таблицу описаний внести указание на отсутствие этих данных в памяти. При попытке обращения к отсутствующим данным процессор сформирует исключение #PF (разновидность прерывания) и отдаст управление операционной системе, которая вернёт данные в память, а затем вернёт управление программе. Таким образом для программ процесс подкачки данных с дисков происходит незаметно. Огромным достоинством страничной виртуальной памяти по сравнению сегментной является отсутствие «ближних» и «дальних» указателей.  


39. Ключевые аспекты развития вычислительных систем. Основные исторические моменты развития, примеры ранних вычислительных систем и их влияние на историю развития ВС в целом.

Постепенно из простейших приспособлений для счета рождались все более и более сложные устройства: абак, логарифмическая линейка, механический арифмометр, электронный компьютер. Несмотря на простоту ранних вычислительных устройств, опытный счетовод может получить результат при помощи простых счет даже быстрее, чем нерасторопный владелец современного калькулятора. Естественно, сама по себе, производительность и скорость счета современных вычислительных устройств давно уже превосходят возможности самого выдающегося расчетчика-человека.

В наше время жизнь без компьютеров не представляется возможной. Внедрение вычислительной техники проникло почти во все жизненные аспекты, как личные, так и профессиональные. Развитие компьютеров было достаточно быстрым. Началом эволюционного развития компьютеров стал 1930 год, когда двоичная арифметика была разработана и стала основой компьютерных вычислений и языков программирования. В 1939 году были изобретены электронно-вычислительные машины, выполняющие вычисление в цифровом виде. Появление вычислительных устройств приходится на 1942 год, когда было изобретено устройство, которое могло механически добавлять числа. Вычисления производились с использованием электронных ламп.

Появившаяся в 1941 году модель Z3 Конрада Цузе в немецкой Лаборатории Авиации в Берлине была одним из наиболее значительных событий в развитии компьютеров, потому что эта машина поддерживала вычисления как с плавающей точкой, так и двоичную арифметику. Это устройство рассматривают как самый первый компьютер, который был полностью работоспособным. Язык программирования считают "Turing-complete", если он попадает в тот же самый вычислительный класс, как машина Тьюринга.

Первое поколение современных компьютеров появилось в 1943, когда были разработаны Марк I и машина Колосс. С финансовой поддержкой от IBM (InternationalBusinessMachinesCorporation) Марк был сконструирован и разработан в Гарвардском университете. Это был электромеханический программируемый компьютер общего назначения. Первое поколение компьютеров было построено с использованием соединенных проводов и электронных ламп (термоэлектронных ламп). Данные хранились на бумажных перфокартах. Колосс использовался во время Второй мировой войны, чтобы помочь расшифровать зашифрованные сообщения.

Чтобы выполнить его задачу расшифровки, Колосс сравнил два потока данных, прочитанных на высокой скорости с перфоленты. Колосс оценивал поток данных, считая каждое совпадение, которое было обнаружено, основываясь на программируемой Булевой функции. Для сравнения с другими данными был создан отдельный поток.

Другой компьютер общего назначения этой эры был ENIAC (Электронный Числовой Интегратор и Компьютер), который был построен в 1946. Он был первым компьютером, способным к перепрограммированию, чтобы решать полный спектр вычислительных проблем. ENIAC содержал 18 000 термоэлектронных ламп, он весил более чем 27 тонн, и потреблял электроэнергии 25 киловатт в час. ENIAC выполнял 100 000 вычислений в секунду. Изобретение транзистора означало, что неэффективные термоэлектронные лампы могли быть заменены более мелкими и надежными компонентами. Это было следующим главным шагом в истории вычислений.

40 Архитектура ARM. Модель программирования, режимы работы процессора. Расширения архитектуры ARM. Особенности реализации системы команд.

Области применения:

* встраиваемые системы, интеллектуальные контроллеры;
* мобильные системы (MID — MobileInternetDevice);
* cпециализированные системы(медиаплееры, медиацентры, системы хранения и др.);
* планшеты-трансформеры, нетбуки.

Базовая архитектура:

* традиционный RISC с дополнительными расширениями (MAC-
* конвейер,
* дополнительные режимы и соответствующие наборы
* команд (Thumb, Thumb-2),
* SIMD-расширение NEON,
* аппаратная поддержка JAVA - Jazelle)

Режимы обработки инструкций:

* + - **ARM**(32бит)
      * режим по умолчанию;
      * доступны все команды процессора, работа с сопроцессором;
    - **Thumb-1**(16бит)
      * высокая плотность кода, лучшее использование I-Cache;
      * на16-битной памяти команда передается за один такт;
      * ограниченный набор команд и регистров;
    - **Thumb-2**(смешанный16632бит)
      * объединяет преимущества ARM и Thumb-1;
      * размер кода на 25% меньше при сравнимой производительности;
    - **Jazelle**(8бит)
      * аппаратно реализовано свыше 60% команд Java-байт кода;
      * режим доступен только производителям оборудования

Состояния процессора:

* + - Процессор может находиться в одном из семи режимов работы:  
      * + **Режим пользователя**(USR) - обычное состояние ARM при выполнении программы, также используется для выполнения большинства прикладных программ.
        + **Режим быстрого прерывания** (FIQ) - поддерживает передачу данных или обработку
        + **Режим прерывания** (IRQ)- используется для обработки прерываний общего назначения.
        + **Супервизорный режим**(SVC)- является защищенным режимом для операционной системы.
        + **Аварийный режим**(ABT) - используется после аварийной выборки данных или инструкции.
        + **Системный режим**(SYS) - привилегированный режим пользователя для операционной системы.
        + **Неопределенный режим**(UND) - соответствует ситуации, когда обнаруживается попытка выполнения неопределенной инструкции.

Все режимы, кроме режима пользователя, совместно называются привилегированными режимами. Привилегированные режимы используются для обслуживания прерываний и исключительных ситуаций, а также для доступа к защищенным ресурсам.

Специальные регистры процессора:

* R13 — указатель стека (SP — StackPointer);
* R14 — регистр связи (LR — LinkRegister);
* R15 — программныйсчетчик (PC — Program Counter);
* CPSR — текущий регистр состояния (CurrentProgramStatus Register);
* SPSR — сохраненныйрегистрсостояния(Saved ProgramStatus Register);  
  