**1) Направл развития ЭВМ. История механических и электромеханических приборов для вычислений. ЭВМ Фон Неймана** – 1. Последов обработка данных; 2. увелич производ за счет улучшения тех характ. **Параллельные вычисл системы** – 1. Парал-ая обработка инфы; 2. Повыш производ за счет увелич кол вычислителей и улучшения тех характер вычислителя. **История вычисл техники:** Абак (счетные доски), арифметич машина, станки с перфокартами, мех мозг, арифмометр, машины Z1,Z2,Z3. Электрон приборы: термоэлектронная эмиссия, 1-ая электрон лампа, триод (управляющий электрод), триггер, полупроводниковый диод, полевой транзистор, биполярный тран. **Механич приборы: 1)** 1-ая мех счетная машина (Шиккард 1623) - операции + и -, а \* и / с элементами механизации. **2)** "паскалина", + и -, изобрел в 1646г. Блез Паскаль. **3)** В 1671 Лейбниц изобрел машину, \* и /. **4)** Жаккар - машина, управляемой введением в нее инфой в 1802 г., кот облегчила процесс производства тканей со сложным узором. **5)**1822 г. Бэббидж счетное ус-во. Для выполн мат опер в машине исп цифровые колеса с зубьями. **6)** Конец XIX в. Холлерит ус-во, в кот впервые была употреб идея перфокарт и расчеты велись с помощью электрич тока. Исп при переписи США. **Электромеханич приборы: 1)** английский COLOSSUS-1 в 1943, расшифровка секрет кода, кот исп Германия. **2)** В1942г. Атанасов Джон Винсент. Особенность компа - разделение обрабатыв и запомин ус-тв. Блок памяти - набора конде-ров с автоматич восстановл заряда. Инфа вводилась с перфокарт. При вычисл исп двоичные числа. Блок управл был собран на электронных лампах и позволял осущ + и - чисел. **3)** “Марк-1” - 1 в мире автоматич вычисл комп (1944 г.) созд профессором Айкнем. В нём использ сочетание электрич сигналов и механич приводов.

**2) Поколения ЭВМ.** *4 основные поколения ЭВМ:* **I**. 1) *Годы применения*: 1945-1955, 2) *Основной элемент*: Эл. Лампа, 3) *Количество ЭВМ в мире*: Сотни, 4) *Размеры ЭВМ*: Большие, 5) *Быстродействие* (усл): 1, 6) *Носитель информации*: Перфокарта, Перфолента. У каждой машины свой язык программир. (“БЭСМ”, ”Стрела”). **II.** 1) 1955-1965, 2) Транзистор, 3) Тысячи, 4) Знач меньше, 5) 10, 6) Магнитная лента. В 1960 г. в ЭВМ были транзисторы, кот были более надёжны, долговечны, с большой операт памятью. 1 транзистор = ~40 эл. ламп и раб с большей скоростью. (“Минск-2”,”Урал-14).

**III.** 1) 1965-1975, 2) ИС, 3) Десятки тысяч, 4) Мини — ЭВМ, 5) 1000, 6) Диск. В 1964 г. появ 1-ые интегральные схемы (ИС — это кристалл, площадь кот 10 мм2). 1 ИС заменит 1000 транзисторов. Появ возмож обрабат параллельно неск прог. **IV.** 1) 1975-Н.В., 2) БИС, 3) Миллионы, 4) микро — ЭВМ, 5) 10000, 6) Жесткий диск. **Б**ольшие (число элементов на кристалле микросхемы) **ИС**, по мощности = 1000 ИС. Это привело к снижению стоимости производства компов. В 1980 г. ЦП небольшой ЭВМ оказалось возможным разместить на кристалле площадью 1/4 дюйма. (“Иллиак”, ”Эльбрус”). были **СБИС** (С – сверх). **3) Типы архитектур вычислительных систем (ВС).**1. *Классич арх (ЭВМ фон Неймана)* — одно АЛУ. Особенности: - последов выполн команд; - блочная организация ЭВМ; - принцип однородности памяти; - принцип адресности; - принцип программного управления. *Классификация Флинна* базируется на понятии потока, т.е. последов элементов, команд или данных, обрабатыв процом. На основе числа потоков команд и потоков данных Флинн выделяет 4 класса архитектур: **SISD** — одиночный **П**оток **К**оманд, одиночный **П**оток **Д**анных, **MISD** — множественный **П К**, одиночный **П Д**, **SIMD** — одиночный **П К**, множеств **П Д**, **MIMD** — множеств **П К**, множеств **П Д**. *Классификация Хокни*: класс MIMD чрезвычайно широк, объединяет много различ типов архитектур. Хокни, пытаясь систематизир архитектуры внутри этого класса, получил иерархическую структуру. Основная идея: множеств поток команд может быть обработан 2 способами: либо одним конвейерным уст-вом обработки, работ в режиме раздел времени для отдел потоков, либо каждый поток обрабатыв своим собстве ус-вом. **Также есть классификации**: *Фенга,* *Шора*, *Хендлера*, *Шнайдера*, *Джонсона*, *Базу*, *Кришнамарфи, Скилликорна, Дазгупты, Дункана*. **4) Шинная организация ЭВМ.** Шина – совокуп линий, идущих параллельно и имеющих одинак функцион назначение. *Шина данных* нужна для пересылки кодов обрабатыв данных, а также машинных кодов команд между уст-вами ЭВМ. *Шина адреса* несет адрес ячейки памяти или порта ввода-вывода, кот взаимод с микропроцом. *Шина управления* несет сигналы управления, обеспеч правильное взаимод блоков ЭВМ друг с другом и с внеш средой. **Архитектуры**: 3шинные, 2-ые (шина адреса + шина данных), 1-ые. *По способам передачи:* **параллельные** (данные переносятся потактово словами: каждый бит — отдельным проводником), **последовательные** (биты данных переносятся поочередно по каналу, например, паре проводников). **5) Иерархия шин. Арбитраж шин.** **Иерархия шин** - если к шине подключ много уст-тв, ее пропус-ая способность падает, т.к. слишком частая передача прав управл шиной от 1 уст-ва к другому приводит к задержкам. Поэтому предпочтение отдается использ неск шин, образ определен иерархию. 3 вида ВМ: **с 1 шиной** (одна системная шина, обеспеч обмен инфой между процом и памятью, а также между УВВ, с 1 стороны, и процом либо памятью - с другой), **с 2 видами шин** (УВВ подключаются к шинам ввода/вывода, которые берут на себя основной трафик, не связанный с выходом на процессор или память. Такая схема снижает нагрузку на скоростную шину «процессор-память» и повышает общую производ ВМ), **с 3 видами шин** (добав высокоскоростная шина расширения. Шины ввода/вывода подключ к шине расшир, а уже с нее через адаптер к шине «проц-память». Схема еще более снижает нагрузку на шину «проц-память»). **Арбитраж шин -** это предостав шины 1 ус-ву, т.е. когда неск ус-тв пытаются использ шину одноврем, это может привести к поврежд данных и нестаб системы**.** Арбитраж осущ по схеме **С**мены **П**риоритетов: **1)** простая циклич С П; **2)** циклич С П с учетом последн запроса; **3)** С П по случ закону; **4)** схема равных П; **5)** алгоритм наиболее давнего использ; **6)** алгоритм очереди (1ым пришел — 1ым обслуж); **7)** алгоритм фиксир кванта времени. **6) Характеристики памяти. Классификация запоминающих устройств. *Основные характ:*** *емкость* (Бит, Байт), *разрядность*, *быстродействие* (tсч = tп + tч + tр, tзп = tп + tз), *массо-габарит показатели*, *энергозависимость*, *удельная стоимость* (руб./байт). ***Классиф механизмов доступа:*** *последоват* (поиск файлов и записей в файлах выполн в той последов, в какой они располож на физ носителе), *прямой* (ус-ва пишут в память и читают из памяти без участия ЦП), *произвольный* (позвол единовреме получить доступ к любой ячейке по её адресу на чтение или запись), *ассоциативный* (адресация осущ на основе содерж данных, а не их местополож, чем обеспеч ускорение поиска нужных записей). ***Запомин ус-ва:*** *Оперативные*: 1) **DRAM** (динам (активная) память, использ конденсаторы и транзисторы), 2) **SRAM** ((КЭШ) стат память, раб быстрее DRAM, использ транзисторы и защелки). *Постоянные*: 1) **EPROM** (стир программир ПЗУ, стир с использ ультрафиол лучей), 2) **EEPROM** (электрически стир программир ПЗУ, стир с использ электрич сигналов), 3) **флэш-память** (читается сколько угодно раз (в пределах срока хран данных, 10-100 лет), но писать в такую память можно около 1 млн циклов).

**7) RAID-массивы.** *RAID* - *Это набор физ дисковых ЗУ, управляемых ОС*, кот объединяет неск дисков в логический элемент для повышения производ. Версии построения массивов: **RAID 0** (чередование дисков) - разделение массива данных на единич блоки и распределяет по неск ус-ам хранения, «HDD» или «SSD», объедин в совокупную группу. *Расслоение данных, полосы распределены по всем дискам массива ЗУ, данные записываются по циклической схеме*; **RAID 1** (зеркальное отображение дисков) - *Запись 1ой инфы на пару жестких дисков. Скорость счит большая, т.к. можно счит с 2 дисков одноврем. При потери данных на 1ом диске можно его восстан при помощи 2ого диска*; **RAID 2** (чередование дисков), выделены отдельные диски, хранящие инфу о проверке и исправлении ошибок («**ECC**»). *Выс степень защиты инфы с помощью кода Хэмминга, можно восстан 1 из дисков. 3 резервных диска;* **RAID 3** - чередов и выделяет 1 диск для хран инфы о четности, которая провер целостность посредством определения, были ли данные потеряны или перезаписаны при перемещении из одного места хранилища в другое, также использ «**ECC**». *Запись на 4 диска. Формируется полосы паритета. 1 резервный диск. Хорошо работает на большом объеме данных;* **RAID 4** -*Как RAID 3, только размер полос паритета больше. Используется техника независимого доступа. Хорошо работает на данных малого размера*; **RAID 5** -*Не содержит отдельного диска для хранения полос паритета, они разносятся по всем дискам, поэтому раб в случае сбоя 1 диска и может восстан данные*; **RAID 6** – (с двойным контролем четности), как «RAID 5», *только на каждом диске хран 2 полосы паритета (одна вертик срез, вторая горизонт). При отказе 2 дисков сможем востан оба.* Повыш уровень защищенности сниж производ массива, кот ниже чем в «RAID 5».

**8) Способы ввода-вывода информации.** *Есть 4 способа*: 1. ***Непосредств*** (обмен данными с УВВ выполн процом); 2. ***по опросу готовности*** (обмен инфой между ЭВМ и внешн уст-ми осущ по инициативе и под управл выполн ЭВМ проги); 3. ***по прерываниям*** (обмен инфой инициир сами перифер уст-ва, генерируя сигнал прерывания. При получ сигнала прерывания микропроц приостанав выполн текущей программы, временно запоминает ее состояние, идентифицирует прерывающее уст-во и осущ обмен инфой); 4. ***прямой доступ к памяти*** (проц предоставляет модулю ввода-вывода права на чтение или запись в память без его участия. Модуль сам управляет обменом данными между основной памятью и устройством ввода-вывода. ЦП участвует только в начале и конце передачи данных и прерывается только после передачи всего блока). Клавиатура, монитор, видеоадаптер (вывод картинки на экран), терминалы (единое устройство, соединённое с ЭВМ (или каким-то другим устройством) через кабельные или телефонные каналы).

**9) Машинная команда, структура. Маш команда** - код, определ операцию ВС, и данные, участв в операции. Команда содерж в явной или неявной форме инфу об адресе рез операции, и об адресе след команды. **Маш операция** – это действия машины по преобраз инфы, выполн под воздействием 1 команды. **Программа** – последов команд, отображ все действия, необход для решения задачи по алгоритму. **Маш такт** – период тактовой частоты работы проца. **Маши цикл** – кол машинных тактов, требуемых для выполн 1 команды. **Группы команд:** **1.** арифм операции над числами с фиксир или плав точкой; **2.** команды двоично-десятичной арифметики; **3.** логические (поразрядные) операции; **4.** пересылка операндов; **5.** операции ввода-вывода; **6.** передача управления; **7.** управление работой ЦП. *В курсовой: в качестве приемника результата и одного из операндов используется аккумулятор (одноадресная).* **Структура**: Маш команда состоит из операционной и адресной частей **(код операции и адрес)**. Эти части могут состоять из неск полей. У маш команды след структура: Операц часть содержит код, задающий вид операции (+, \*, передача и т.д.). Адресная часть содержит инфу об адресах операндов, результата операции и след команды. **Форматы маш команд:** Фиксиров – все команды заним одинак кол бит, как и каждое поле (код операции и адресная часть). Плавающий -противопол фикс.

**10) Показатели производительности. *Производ ЭВМ*** – потенц возмож по обработке инфы (а не реальная). В процессе обработки инфы в ЭВМ реализ те или иные операции из ее набора (или системы) операций. Состав набора операций характеризует архитектуру ЭВМ и определяет ее производ. **Показатели**: **1.** *тактовая частота* (кол команд (тактов), кот вып проц за 1 сек, *Гц, MГц, ГГц*); **2.** *номинальное быстродействие:* (кол операций, выполняемых ус-ами компьютера за сек, для чисел с фиксир запятой: *MIPS, MOPS*); **3.** *быстродействие по Гибсону* (можно сравн быстродей разл вычисл ср-тв при реш ими науч-технич задач)*:*  (для чисел с фиксир запятой: MIPS, MOPS); **4.** *оценка по независ тестам* (1 FLOPS = 1 опер./с, 1 MegaFLOPS = млн опер./с, 1 GigaFLOPS = млрд опер./с, 1 TeraFLOPS = трлд опер./с, 1 PetaFLOPS = квадр оп./с)

**11) Процессор. Типы архитектур.** Проц — это мозг компьютера. Он читает инструкции (числа, кот интерпрет спец образом) из памяти, кот указ, что делать компу. Микропроцы (CPU) выполн простые операции: Intel и AMD, использ *арх набора команд x86*. А микропроц, A12, A13, A14, M1 от Apple, исп набор команд *ARM*. **Арх набора команд сильно влияет на арх проца.** *Арх CISC* — ***расширенный набор команд***: большой набор команд, использ переменные диапазоны форматов от 16-64 бит на инструкцию; много способов адресации от 12 до 24; арх использ 8-24 регистров общего назна (мало); механизм памяти к памяти для выполн операций; унифициров кэш для данных и инструкций; имеет микрокодирование и исп управляющую память (ПЗУ). *Архитектура RISC* - ***сокрашенный набор команд***: малый набор команд, фиксир формат маш команды (32 бита), исп один такт на команду и режим огранич адресации (т.е. 3-5), число регистров общего назнач колеблется от 32 до 192 (большое), основные операции в проце выполняются над содержимым регистров, имеет разделенный дизайн данных и кэш инструкций, большая часть управления процом является аппаратной, имеет большую производ чем CISC. Но под него мало ПО. **Отличие форматов команд CISC и RISC:**

У RISC фиксир длина команд. *Гибридные архитектуры:* Идея — обеспечить программную совместимость с CISC процами, но добиться эффективности RISC процов. Микропроц имеет интерфейс CISC, а команды выполняет как RISC, поддерживается конвейеризация (способ организации вычислений).

**12) Способы адресации. Способ адресации** - алгоритм получ физ адреса по командному. В адресном поле команды содерж командный (исполнит) адрес. Команда выбир по физ адресу. **Явные** - в адресном поле имеется командный адрес: **1.** *Непосредств адресация*: операнд (байт или слово) размещ в коде команды. **2.** *Прямая*: физ адрес совпадает с командным. **3.** *Косвенная*: командный адрес содерж адрес ячейки памяти или регистра где нах операнд. **4.** *Автоинкрементная*: после выполн операции с данными адрес операнда увелич на 1, если операнд имеет формат "байт", или на 2, если операнд имеет формат «слово». **5.** *Базовая*: физ адрес явл суммой базового и командного адресов. **6.** *Базово-индексная*: физ адрес явл суммой базового, индексного и командного адресов. **7.** *Укороченная*: адресный код содержит часть физ адреса, другая часть подразумев. **8.** *Регистровая*: адресный код содержит номер регистра. **9.** *Стековая*: указ вершина стека. **Неявные** - отсутствует адрес. 1. Подразумев операнд (i++). 2. Подразумев адрес операнда или приемника рез-та. *В курсовой: адрес 2ого операнда подразумев аккумулятор, также приемник рез-та явл аккумулятор. Адрес след команды +1) (исп прямая адре-ия).*

**13) Устройство управления. Жесткое и микропрограммное управление.** *УУ формир управл сигналы для выполн след функций:* **1.** выборка из основной памяти (ОЗУ или ПЗУ) кода команды; **2.** расшифровка кода операции и признаков выбранной команды; **3.** формир исполнител адреса операнда; **4.** выборка операндов и выполн маш опер; **5.** анализ запросов на прерывание исполн проги; **6.** формир адреса след команды.**Принципы построения управляющих устройств:** 1) ***Управляющие устройства с жесткой логикой*** (закон функционирования определяется способом соединения логических элементов). Минус: невозможно внести правки в связи. Плюс: работает очень быстро. ***УУ с жесткой логикой***– этосовокупность лог схем, вырабатыв распределенные во времени функциональные управляющие сигналы. Изменить логику работы у таких управляющих автоматов можно, только переделав схему. Множество состояний таких управляющих автоматов задается набором запоминающих элементов (ячеек памяти), а функции переходов и выходов – реализ набором лог элементов. 2) ***Микропрограммные устройства управления*** (закон функционирования задается программой, хранимой в ячейках памяти, называемых управляющей памятью). Плюс: можно вносить правки в связи. ***УУ с микропрограмной логикой***каждой выполняемой маш операции ставится в соответствие совокупность хранимых в памяти слов – микрокоманд. Последовательность микрокоманд образует микропрограмму, хранимую в спец памяти микропрограмм. Каждая микропрограмма состоит из определен последовательности микрокоманд, которые после выборки из памяти преобраз в набор управляющих сигналов. **БУО** (блок управления операциями (АЛУ)) выполняет функцию блока хранения и выборки кодов микрокоманд. В машинном такте производится выборка требуемой микрокоманды, выдача сигналов микроопераций, формирование адреса след микрокоманды.

**14) Периферийные устройства ввода ЭВМ.** Периферийные уст-ва входят в состав внеш оборудов ЭВМ, обеспеч ввод/вывод данных, промежуточное и длит хран данных, передачу инфы, но не определяют архитектуру и принципы функционир ЭВМ. Осуществ связь ЭВМ с различными «поставщиками» и «потребителями» инфы. *Ус-ва ввода инф:* Клавиатуры (проводные и беспров); Мыши (проводные и беспров); Сканеры; Графич планшеты; Пенмаусы (аналог шарик ручки); Световые перья; Цифр видеокамеры и фотики, вебкамеры; Джойстики; Сенсорные панели (тачпады); Средства речевого ввода.

**15) Периферийные устройства вывода ЭВМ.** *Ус-ва вывода инф: Мониторы* (ЭЛТ, ЖК, LED, OLED, qLED); *Сенсорные экраны* (резистривные, емкостные*); Синтезаторы звука* (спикер, бытовые, профессиональные); *Проекторы* (LCD, DLP, LCoS, CRT); *Принтеры* (ударные, струйные, электромагнито-графич, электростатические, термические); *3D-принтеры* (FDM, печать пластиком), SLA (фотополимерная печать), *SLS* (печать полиамидом – термопласт, исп в качестве клея), *SLM* (печать металлом); *Плоттеры* (векторного, растрового типов, планшетные, барабанные, перьевые, струйные, карандашные).

**16) Внутренняя структура процессора i8086.** Первый 16-битный процессор с 16-битным ALU, 16-битными регистрами, внутренней шиной данных и 16-битной внешней шиной данных, что привело к более быстрой обработке. **Особенности i8086:** CISC архитектура, 20 адресных линий, может адресовать 2Мб оперативной памяти, поддерживает прямой доступ к памяти, обмен по запросу на прерывание, внутрен регистры 16-разрядные, поддерживает конвейер команд, тактируется от внешнего генератора. **Регистр флагов:** состоит из 16 бит, из них используются 9. **CF** — содержит перенос из старшего бита после арифметических операций и сдвигах, **PF** — показывает четность младших восьмибитовых данных (1 – четное, 0 – нечетное), **AF** — признак межтетрадного переноса (из 3 бита), **ZF** — флаг нулевого результата, **SF** — флаг знака (совпадает со старшим битом результата, 0 – плюс, 1 - минус), **TF** — флаг пошагового режима (используется при отладке), **IF** — флаг разрешения аппаратных прерываний, **DF** — флаг направления при строковых операциях. Обознач левое или правое направление, **OF** -указ на переполнение старш бита. **Включает в себя три основных устройства: УОД** - устройство обработки данных (предназначено для выполнения команд); **УСМ** - устройство связи с магистралью (обеспечивает формирование 20-разрядного физического адреса памяти и 16-разрядного адреса ВУ, выбор команд из памяти, обмен данными с ЗУ, ВУ, другими процессорами по магистрали; **УУС** - устройство управления и синхронизации (обеспечивает синхронизацию работы устройств микропроц, выработку управляющих сигналов и сигналов состояния для обмена с другими устройствами, анализ и соответствующую реакцию на сигналы других устройств МПС.) Микропроц может работать в одном из двух режимов - min (увеличен адресное пространство, более высокое быстродействие и значительно расширен системой команд) и max (наличие в системе нескольких МП и специального блока арбитра магистрали).

**17) Назначение входов процессора i8086. CLK** (синхронизация)- вход тактовых импульсов, задающих частоту проца; **NMI** (управление) - немаскируемые запросы на прерывания; **INTR** (упр) - вход маскируемых запросов на прерывания; **RESET** (упр) - аппаратный сброс процессора; **READY** (упр) - готовность внешнего устройства или памяти к обмену данными; **TEST** (упр) - входной сигнал проверки, использ в сочетании с сопроцами; **MN/MX** (упр) - задает режим работы процессора; **HOLD** **(RO/GT0)** (упр) - вход запроса на прямой доступ к памяти; **V** - питание (+5В); **GND** - заземление (отриц контакт питания).

**18) Назначение выходов процессора i8086. AD0-AD15** - совмещ шина адреса и данных; **AD16-AD19** —старшие разряды шины адреса; **BHE/S7** (упр) - для разреш подключения старшего банка памяти (16 или 8); **RD** (упр) - чтение из памяти или внешнего ус-ва; **WR (LOCK)** (упр) - запись в порт внеш ус-ва или в память; **DT/R (S1)** (упр) — 1 - передача данных, 0 -прием; **INTA (QS1)** (упр) —для подтвержд обработки запроса на прерывание; **HLDA (RO/GT1)** (упр) —выход подтверж запроса на прямой доступ к памяти; **DEN (S0)** (упр) —по совмещ шине передаются данные; **ALE (QS0)** (упр) - при выставлении адреса на шину формир посылающий импульс; **M/IO (S2)** (упр) —раздел адресных пространств памяти и внеш ус-тв.

**19) Схема демультипликации шин адреса и данных.** От проца подается мультиплексированная шина адреса и данных и передается на 2 устройства. Первое устройство **буферный регистр** - ячейка памяти, способная что-то хранить. При появления единичного сигнала на входе запоминает что на входе, удерживает и выставляет на своем выходе, а именно на шину адреса. Второе **шинный формирователь** – его задача обеспечить определенное согласование. При наличии на входе den 1 шинный формирователь отключается и все что находится в ШАД дальше не выходит. Когда на входе den 0, то шинный формирователь подключается к внешним шинам и обеспеч передачу в обоих направлениях.

**20) Организация памяти i8086.** (+смотри 21) ЭВМ выполняет вычисления в соответствии с прогой, кот располагается в памяти ЭВМ. Любая прога включает в себя команды (операторы) и данные (операнды). Прога выполн с целью получения результирующих данных на основе преобразования исходных, с возможными промежуточ данными. Команды и данные располагаются в единой памяти и представлены в двоичных кодах. Память представляет собой набор ячеек, каждая из кот имеет свой уникальный номер – адрес. Т.к. команды и данные на уровне кодов неотличимы друг от друга, то для различия команд и данных используется их размещение в различных областях памяти – сегментах. (В курсовой: для отличия используется специальный бит).

**21) Сегментация памяти i8086. Сегмент** - это прямоуг обл памяти, характериз начальным адресом и длиной. *Начальный адрес* – это номер (адрес) ячейки памяти, с кот начин сегмент. *Длина сегмента* –кол входящ в него ячеек памяти. Сегменты могут иметь различ длину. Все ячейки, расположен внутри сегмента, перенумеров-ся, начиная с 0. Адресация ячеек внутри сегмента ведется относит начала сегмента; адрес ячейки в сегменте наз смещением или эффективным адресом. **Переполнения при формировании адреса:** Перенос из старшего бита, кот может возникнуть при суммир, игнорируется. Это приводит к кольцевой организации памяти, при кот за ячейкой с макс адресом FFFFF следует ячейка с нулевым адресом. Аналогич кольцевую организ имеет и каждый сегмент. **Сегментные регистры:** 16-битная x86-архитектура, наличие 4х сегментных регистров позволяет одноврем иметь доступ к 4 сегментам памяти. *Назначение сегментных регистров*: **DS** - сегмент данных; **CS** - сегмент кода; **SS** - сегмент стека; **ES** - доп сегмент. **Лог адрес записывают так**: смещение, где сегмент и смещение задаются в 16 системе счисл. В реальном режиме для вычисл физ адреса байта памяти происх сдвиг влево на 4 разряда значения соответств сегментного регистра, а затем добавл смещение. Лог адрес 7522:F139 дает 20-битный физ адрес: 75220 + F139 = 84359. **Особен сегментации памяти в микропроце i8086:** **1.** Сегменты памяти определ только сегментными регистрами; **2.** Начальный адрес сегмента кратен 16; **3.** Никаких средств проверки правильности использ сегментов нет; **4.** Размещение сегментов в памяти достаточно произвольно; **5.** Сегменты могут частично или полностью перекрываться, или не иметь общих частей; **6.** Прога может обращаться к любому сегменту как для считывания, так и для записи данных и команд; **7.** Для защиты памяти от несанкциониров доступа других прог требуются спец "внешние" схемы; **8.** Система не делает различий между сегментами данных, кода и стека; **9.** Нет никаких препятствий для обращения к физически не существ памяти; **10.** При обращении к несуществ памяти рез непредсказуем (все зависит от разработчика аппаратного обеспечения компа) **Защищенный режим:** Появился в проце 80286, сохранив понятия сегмента и смещения, и изменил механизм сегментации, предложив идею виртуального адреса. В защищ режиме работы проца содержимое сегментных регистров использ не как слагаемое при вычислении адреса, а как индекс или селектор, выбирающий дескриптор сегмента в спец структуре, наз *дескрипторной таблицей* и опис св-ва каждого сегмента: базовый адрес, размер и атрибуты, связанные с управлением привилегиями и защитой памяти.

**22) Организация прерываний на примере i8086.** Прерывание - это усл, кот временно останав МП для выполн другой задачи, а затем возвращ его к предыд задаче. Когда возникает прерывание, проц завершает текущую инструкцию и запускает обработчика прерываний ISR - это прога, кот сообщает проц, что делать при возникнов прерывания и после выполн ISR управление возвращается к основной подпрограмме, на которой оно было прервано. **В микропроце 8086 при возник прерывания выполн след задачи:** **1.** Значение регистра флага помещ в стек. **2.** Значение нач адреса памяти CS (сегмента кода) помещ в стек. **3.** Значение IP (указатель инструкции) помещается в стек. **4.** IP загружается из местоположения слова (тип прерывания) \* 04. **5.** CS загружается из следующего расположения слова. **6.** Флаги прерывания сбрасываются на 0. Таба векторов прерываний: в реал режиме работы в системе прерыв использ понятие вектора прерывания. Каждый вектор прерывания состоит из 4х байт, или двух слов: первые 2 содержат новое значение для регистра IP, а след 2 – новое значение для регистра CS. Таба векторов прерывания заним 1024 байт, в ней может быть задано 256 векторов прерываний. 2 вида прерываний: 1) **Внутр прерыв** возник в результате работы проца в ситуациях, которые нуждаются в специальном обслуживании, или при выполнении специальных команд (INT, INTO) (- прерывание при делении на ноль (номер прерывания 0); - прерывание по флагу TF (номер прерывания 1); - прерывания, возникающие при выполнении команд INT наз программными). В качестве операнда команды INT указ номер прерывания, кот нужно выполнить. 2) **Внеш прерыв** возникают по сигналу какого-нибудь внешнего уст-тва. Сущ 2 спец внешних сигнала среди входных сигналов процессора, при помощи кот можно прервать выполн текущей проги и тем самым переключить работу ЦП: 1. **NMI** (немаскируемое прерывание); 2. **INTR** (запрос на прерывание).***Немаскируемые прерывания*** *будут всегда обрабатыв процом. Маскируемые прерывания будут обрабатыв только если у нас стоит флаг.* ***Этапы обработки прерываний:*** 1. Контроллер прерываний получает заявку от определенного периферийного уст-ва и, соблюдая схему приоритетов, генерирует сигнал INTR (запрос на прерывание), кот явл входным для микропроца. 2. Микропроц проверяет флаг IF в регистре флагов. Если он установлен в 1, то переходим к шагу 3. В противном случае работа проца не прерывается. 3. Микропроц генерирует сигнал INTA (подтверждение прерывания). В ответ на этот сигнал контроллер прерываний посылает по шине данных номер прерывания. 4. В стек помещается регистр флагов. 5. Флаг включения-выключения прерываний IF и флаг трассировки TF, наход в регистре флагов, обнуляются для блокировки других маскируемых прерываний и исключения пошагового режима исполнения команд. 6. Значения регистров CS и IP сохраняются в стеке. 7. Вычисляется адрес вектора прерывания и из вектора, соответс номеру прерывания, загружаются новые значения IP и CS. 8. Выполнение подпрограммы обработчика прерывания. 9. Извлечение из стека IP и CS. 10. Извлечение из стека регистра флагов. 11. Проц продолжает работу с того места, где он был прерван. ***Использование прерываний:* 1)** для обмена информацией между процессором и внешним устройством (ВУ); **2)** в аварийных ситуациях, например, при понижении напряжения питания; **3)** при исключительных условиях, таких, как переполнение; **4)** для индикации аппаратных сбоев, приводящих к ошибкам при обработке данных; **5)** при программных сбоях; **6)** для координации работы в многопроцессорных системах; **7)** для профилактики, ремонта, тестирования и отладки системы.