**Архитектура ЭВМ и систем.**

**28. Поколения архитектур ЭВМ. Понятие архитектура ЭМВ в узком и широком смысле. Основные характеристики ЭВМ**

Идея делить машины на поколения вызвана стремительной эволюцией ЭВМ как в смысле элементной базы, так и в смысле изменения её структуры, появления новых возможностей, расширения областей применения и характеристика использования.

Поколения архитектур ЭВМ:

К 1-му поколению относятся машины, созданные на рубеже 50-х годов. Элементная база- электронные лампы и реле; оперативная память выполнялась на триггерах, позднее на ферритовых сердечниках. Надежность - невысокая, требовалась система охлаждения; ЭВМ имели значительные габариты. Быстродействие: 5 - 30 тыс. арифметических оп/с; Программирование: в кодах ЭВМ (машинный код), позднее появились автокоды и ассемблеры. Программированием занимался узкий круг математиков, физиков, инженеров - электронщиков. ЭВМ первого поколения использовались в основном для научно-технических расчетов. Отечественные машины этого поколения: Урал, м-20.

2-ое поколение (1955-1965) характеризуется использованием как ламповых так и транзисторных логических элементов. Оперативная память построена на магнитных сердечниках. Появились высокопроизводительные устройства для работы с магнитными лентами, магнитные барабаны и первые магнитные диски. Быстродействие достигло сотен тысяч операций, ёмкость памяти - десятков тысяч слов. Появились языки высокого уровня. Для перевода их на машинный язык были созданы особые программы, которые получили название трансляторы. Появился широкий набор библиотечных программ, которые в дальнейшем привели к созданию оперативных систем. Недостаток машин 2ого поколения - это программная несовместимость.

3-е поколение. Совместимость машин программной архитектуры. Имеют разные ОС, обладают возможностями мультипрограммирования. Быстродействие до млн оп/с. Ёмкость ОП несколько сотен тысяч слов. Примеры: IBM 360, 370 и единая система ЭВМ.

4-ое поколение (после 1970г.). Эти машины проектировались в расчёте на эффективное использование современных высокоуровневых языков и упрощение процесса программирования для конечного пользователя. Быстродействие несколько десятков млн слов. ОП 512Мб. Для них характерно: применение ПК, телекоммуникационная обработка данных, компьютерные сети, широкое применение СУБД, элементы интеллектуального поведения СУБД и устройств.

5-ое поколение. В них должен произойти качественный переход от обработки данных к обработке знаний. Архитектура таких компьютеров будет содержать 2 основных блока: традиционный компьютер + интеллектуальный интерфейс.

В узком смысле под архитектурой понимается архитектура набора команд. Архитектура набора команд служит границей между аппаратурой и программным обеспечением и представляет ту часть системы, которая видна программисту или разработчику компиляторов. В широком смысле архитектура охватывает понятие организации системы, включающее такие высокоуровневые аспекты разработки компьютера как систему памяти, структуру системной шины, организацию ввода/вывода и т.п.

Характеристики ЭВМ:

1. Число разрядов в машинном слове.

2. Скорость выполнения основных видов команд.

3. Ёмкость ОП.

4. Макс скорость передачи информации между ядром ЭВМ и периферийным оборудованием.

5. Эксплуатационная надёжность машины.

**29. Принципы (архитектура) фон Неймана. Состав ЭВМ, основные устройства**

**Принципы Фон Неймана:**

**Принцип однородности памяти**. Команды и данные хранятся в одной и той же памяти и внешне в памяти неразличимы. Распознать их можно только по способу использования; то есть одно и то же значение в ячейке памяти может использоваться и как данные, и как команда, и как адрес в зависимости лишь от способа обращения к нему. Это позволяет производить над командами те же операции, что и над числами, и, соответственно, открывает ряд возможностей. Так, циклически изменяя адресную часть команды, можно обеспечить обращение к последовательным элементам массива данных. Такой прием носит название **модификации команд** и с позиций современного программирования не приветствуется. Более полезным является другое следствие принципа однородности, когда команды одной программы могут быть получены как результат исполнения другой программы. Эта возможность лежит в основе **трансляции** — перевода текста программы с языка высокого уровня на язык конкретной вычислительной машины.

**Принцип адресности**. Структурно основная память состоит из пронумерованных ячеек, причем процессору в произвольный момент доступна любая ячейка. Двоичные коды команд и данных разделяются на единицы информации, называемые **словами**, и хранятся в ячейках памяти, а для доступа к ним используются номера соответствующих ячеек — **адреса**.

**Принцип программного управления**. Все вычисления, предусмотренные алгоритмом решения задачи, должны быть представлены в виде программы, состоящей из последовательности управляющих слов — команд. Каждая команда предписывает некоторую операцию из набора операций, реализуемых вычислительной машиной. Команды программы хранятся в последовательных ячейках памяти вычислительной машины и выполняются в естественной последовательности, то есть в порядке их положения в программе. При необходимости, с помощью специальных команд, эта последовательность может быть изменена. Решение об изменении порядка выполнения команд программы принимается либо на основании анализа результатов предшествующих вычислений, либо безусловно.

**Принцип двоичного кодирования**. Согласно этому принципу, вся информация, как данные, так и команды, кодируются двоичными цифрами 0 и 1. Каждый тип информации представляется двоичной последовательностью и имеет свой формат. Последовательность битов в формате, имеющая определенный смысл, называется полем. В числовой информации обычно выделяют поле знака и поле значащих разрядов. В формате команды можно выделить два поля: поле кода операции и поле адресов.

**Состав ЭВМ:**

**Центральное устройство**: состоит из центрального процессора и оперативной памяти. ЦП реализует операции обработки информации и управления вычислительным процессом, осуществляет выборку машинных команд и данных из ОП и запись в ОП. Включение и отключение внешних устройств. Процессор состоит: 1.устройство управления с интерфейсом процессора (системы сопряжения связей процессора с другими узлами машины). 2. АЛУ. 3. Процессорная память (КЭШ).

Центральное устройство описывается следующими характеристиками: 1. длина машинного слова (разрядность, адресность), 2. система команд, 3. объём ОП, 4. Быстродействие (тактовая частота процессора, цикл записи/считывания ОП).

Внешние устройства: обеспечивают взаимодействие компьютера с окружающей средой – пользователями, объектами управления, другими вычислительными машинами. Внешние устройства делятся на след группы: 1. Устройства ввода-вывода. 2. Устройства хранения (массовые накопители). 3. Устройство массового ввода/вывода информации.

**Классическая архитектура** (Фон Неймана) – это однопроцессорный компьютер, который включает в себя следующие типы устройств: 1. Центральный процессор (АЛУ, устройство управления). 2. Запоминающее устройство (ОП и внешнее ЗУ). 3. Устройства ввода-вывода.

**30.** **Регистры АЛУ микропроцессора. РОН, сегментные регистры, регистр флагов**

Данные, с которыми работает процессор, должны находиться в *регистрах*. **Регистры** – это устройства, предназначенные для временного хранения данных ограниченного размера. Регистр состоит из разрядов, которые можно записывать, запоминать и считывать слово, команду, двоичное число и т.д. Регистр, обладающий способностью перемещать содержимое своих разрядов, называется сдвиговым. В этом регистре за 1 такт хранимое слово сдвигается на 1 разряд. Такие регистры используются для кодирования и декодирования. Пользовательские регистры: такие регистры программист может использовать для написания программ. К ним относятся РОН (8), регистры сегментов (6), регистры состояния и управления (2).

Регистры общего назначения. Физически расположены внутри АЛУ процессора. Все регистры общего назначения могут использоваться для адресных вычислений и для получения результатов большинства арифметических и логических операций. Тем не менее, некоторые команды используют фиксированные регистры для хранения операндов. Например, команды обработки строк используют в качестве операндов содержимое регистров ECX, ESI и EDI. Использование фиксированных регистров для некоторых операций позволяет более компактно кодировать набор команд. Следующие команды используют фиксированные регистры: умножение и деление, ввод/вывод, обработка строк, перекодирование, цикл, сдвиговые операции, операции со стеком. Eax – аккумулятор, применяется для хранения промежуточных данных. Ebx – базовый регистр, применяется для хранения базового адреса некоторого объекта в памяти. Ecx – регистр – счетчик, применяется в командах, производящих некоторые повторяющиеся действие. Esi – индекс источника, этот регистр в цепочных операциях содержит текущий адрес элемента цепочки устройства. Edi – индекс приемника, в цепочных операциях содержит регистр. Esp – регистр – указатель стека, содержит указатель вершины стека в текущем элементе. Ebp – регистр базы кадра стека, предназначен для организации произвольного доступа к данным внутри стека. **Стек** – структура данных, которая поддерживает на программно-аппаратном уровне в архитектуру микропроцессора.

Сегментные регистры*.* Микропроцессор аппаратно поддерживает структурную организацию программы в виде 3-х частей, которую называют сегментами, соответственно память называется сегментная. Микропроцессор поддерживает следующие типы сегментов: 1) Cs - сегментный регистр кода (содержит команды программы). 2) Ds - сегментный регистр данных (содержит обрабатываемые данные). 3) Ss - сегментный регистр стека. (область памяти, которую называют стеком), 4) Es, gs, fc - дополнительный сегмент данных.

Регистр флагов. Этот регистр, называется eflags или 16-битная его часть flags. Этот регистр содержит информацию, которая используется побитно, а не в качестве числа. Каждый бит этого регистра называется флагом и имеет определенное значение. Программа пользуется этими флагами для управления своим выполнением. Так же есть управляющие флаги, которые задают работу программе. По выполнению каждой команды, анализируя определенные флаги можно судить о выполнении команды. Например, команда add (сложение двух чисел). По ее выполнению можно определить был ли перенос разрядов. Соответственно если был, то определенный флаг (СF) установится в 1, если нет, то останется 0. Дальше, анализируя флаг с помощью команд условного перехода можно обработать перенос.

Дополнительные биты слова состояния процессора PSW имеют следующее назначение: CF - флаг переноса, PF - флаг четного, AF - флаг дополнительного переноса, ZF - флаг нуля, SF - флаг знака, OF - флаг переполнения, IOPL - два бита, определяющие уровень привилегий ввода/вывода, NT - флаг вложенной задачи.

3 группы флагов: 1) 8 флагов состояния. Эти флаги изменяются после выполнения машинных команд. Отражают особенности результата выполнения логических или арифметических команд. Это даёт возможность анализировать состояние вычислительного процесса и реагировать на него с помощью команд условных переходов и вызова подпрограмм. 2) флаг управления DF. Значение DF определяет направления по элементной обработке в этой операции: от начала строки к концу (DF = 0) или от конца к началу (DF = 1). Для работы с DF существуют специальные команды: cld – снять флаг DF и std – установить DF. Их использование позволяет увеличить или уменьшить значение счётчика при выполнении операций со строками. 3) 5 системных флагов. Управляют вводом, выводом, прерыванием, отладкой, переключением между режимами процессора.

**31. Основные характеристики памяти. Критерии классификации запоминающих устройств. Иерархия памяти современного ПК.**

Памятью ЭВМ называется совокупность устройств, которая служит для запоминания, хранения, и выдачи информации. Отдельные устройства, входящие в эту совокупность называются **запоминающие устройства**.

Классификация ЗУ:

1) по типу ЗУ: полупроводниковые, магнитные, конденсаторные, оптоэлектронные, голографические, криогенные.

2) по функциональному назначения: ОЗУ, СОЗУ(сверхоперативные), внешние ЗУ, промежуточные ЗУ.

3) по способу организации обращения: с последовательным поиском, с прямым доступом, с непосредственным доступом, ассоциативные, стековые, магазинные.

4) по характеру считывания: с разрушением информации, без разрушения информации.

5) по способу хранения: статические, динамические.

6) по способу организации: однокоординатные, двух-координатные, - трех- координатные, - двух-трех- координатные.

Характеристики памяти:

**Емкость** памяти определяется максимальным количеством данных, которые могут в ней храниться.

**Удельная емкость** – отношение емкости ЗУ к её физическому объему.

**Быстродействие памяти** определяется продолжительностью операции обращения, т.е. временем, затрачиваемым на поиск нужной единицы информации в памяти на её считывание или на поиск места в памяти, предназначенного для хранения данной единицы информации и на её запись в память.

Продолжительность обращения к памяти при считывании определяется по формуле:

, где - время доступа, определяющееся промежутком времени между моментом начала операции обращения при считывании до момента когда становится возможным доступ к данной единице информации, - продолжительность самого физического процесса обнаружения и фиксации состояния соответствующих запоминающих элементов.

Иерархическая структура памяти.

Идеальная память должна обеспечивать процессор командами и данными так, чтобы не вызывать простоев процессора. При этом память должна иметь большую емкость. В современных условиях уменьшение времени доступа достигается введе­нием многоуровневой иерархии памяти. Время доступа зависит от объема и типа используемой памяти.

Типовая современная иерархия памяти имеет следующую структуру:

* регистры 64 - 256 слов с временем доступа 1 такт процес­сора;
* кэш 1 уровня - 8к слов с временем доступа 2 такта;
* кэш 2 уровня - 256к слов с временем доступа 3-5 тактов;
* основная память - до 4 Гигаслов с временем доступа 12-55 тактов.

Используя помимо основной памяти небольшую и более быструю бу­ферную память, можно значительно сократить количество обращений к основной памяти, за счет аккумуляции текущего фрагмента программного кода в буферной памяти. Создание иерархической многоуровневой памяти, пересылающей блоки программ и данных между уровнями памяти за вре­мя, пока предшествующие блоки обрабатываются процессором, позволяет существенно сократить простои процессора в ожидании данных. При этом эффект уменьшения времени доступа в память будет тем больше, чем боль­ше время обработки данных в буферной памяти по сравнению с временем пересылки между буферной и основной памятью. Это достигается при ло­кальности обрабатываемых данных, когда процессор многократно исполь­зует одни и те же данные для выработки некоторого результата. Например, такая ситуация имеет место при решении систем уравнений в научных и инженерных расчетах, когда короткие участки программного кода с боль­шим количеством вложенных и зацепленных друг с другом циклов обраба­тывают поочередно, переходя от точки к точке, небольшие порции данных, многократно используя одни и те же данные и внутренние результаты.

В связи с тем, что локально обрабатываемые данные могут возникать в динамике вычислений и не обязательно сконцентрированы в одной об­ласти при статическом размещении в основной памяти, буферную память организуют как ассоциативную, в которой данные содержатся в совокуп­ности с их адресом в основной памяти. Такая буферная память получила название кэш-памяти. Кэш-память позволяет гибко согласовывать струк­туры данных, требуемые в динамике вычислений, со статическими струк­турами данных основной памяти.

**32. Адресная, ассоциативная и стековая организация памяти. Кэш – память.**

ЗУ с произвольным обращением обычно содержит множество одинаковых запоминающих элементов, образующих запоминающий массив (ЗМ). ЗМ разделен на отдельные ячейки, число разрядов в которых равно ширине выборки памяти. Способ организации памяти зависит от метода размещения и поиска информации в ЗУ. По этому признаку различают адресную, ассоциативную и стековую память.

Адресная память. В памяти с адресной организацией размещение и поиск информации в запоминающем массиве (ЗМ) основаны на использовании адреса хранения слова (числа, команды и т.п.). Адресом служит номер ячейки ЗМ, в которой это слово размещается.

При записи (или считывании) слова в ЗМ инициирующая эту операцию команда должна указывать адрес (номер ячейки), по которому производится запись (считывание).

Ассоциативная память. В памяти этого типа поиск нужной информации производится не по адресу, а по ее содержанию (по ассоциативному признаку). При этом поиск по ассоциативному признаку (или последовательно по отдельным разрядам этого признака) происходит параллельно во времени для всех ячеек запоминающего массива. Во многих случаях ассоциативный поиск позволяет существенно упростить и ускорить обработку данных. Это достигается за счет того, что в памяти этого типа операция считывания информации совмещена с выполнением ряда логических операций.

Для ассоциативной памяти необходимы запоминающие элементы, допускающие считывание без разрушения записанной в них информации. Это связано с тем, что при ассоциативном поиске считывание производится по всему ЗМ для всех незамаскированных разрядов и негде сохранять временно разрушаемую считыванием информацию.

Стековая память является безадресной. Стековую память можно рассматривать как совокупность ячеек, образующих одномерный массив, в котором соседние ячейки связаны друг с другом разрядными цепями передачи слов.

Запись нового слова производится в верхнюю ячейку (ячейку 0), при этом все ранее записанные слова (включая слово, находившееся в ячейке 0), сдвигаются вниз, в соседние ячейки с большими на 1 номерами. Считывание возможно только из верхней (нулевой) ячейки памяти, при этом, если производится считывание с удалением, все остальные слова в памяти сдвигаются вверх, в соседние ячейки с меньшими номерами. В этой памяти порядок считывания слов соответствует правилу: последним поступил - первым обслуживается. В ряде устройств рассматриваемого типа предусматривается также операция простого считывания слова из нулевой ячейки (без его удаления и сдвига слова в памяти). Иногда стековая память снабжается счетчиком стека (СчСт), показывающим количество занесенных в память слов. Сигнал СчСт=0 соответствует пустому стеку, СчСт=N-1 - заполненному стеку.

Обычно стековую память организуют, используя адресную память. В этом случае счетчик стека, как правило, отсутствует, так как количество слов в памяти можно выявить по указателю стека. Широкое применение стековая память находит при обработке вложенных структур данных.

Кэш память представляет собой буферное ЗУ, работающее со скоростью, обеспечивающей функционирование центрального процесса без режима ожидания. Создание иерархической многоуровневой памяти, пересылающей блоки программ и данных между уровнями памяти за вре­мя, пока предшествующие блоки обрабатываются процессором, позволяет существенно сократить простои процессора в ожидании данных. Успешные или неуспешные обращения к более высокому уровню памяти называются соответственно **попаданием или промахом**. Частота попадания и промаха являются одной из важнейших характеристик памяти: время обращения и время доступа. Кэш имеет совокупность строк, каждая из которых состо­ит из фиксированного количества адресуемых единиц памяти (байтов, слов) с последовательными адресами. Типичный размер строки: 16, 64, 128, 256 байтов. Наиболее часто используются три способа организации кэш-памяти: 1) кэш-память с прямым отображением; 2) частично ассоциативная кэш-память; 3) ассоциатив­ная кэш-память. Кэш память называется **полностью ассоциативной**, если некоторый блок основной памяти может располагаться в любом месте кэш памяти. И называется **частично ассоциативной**, если блок памяти может располагаться на ограниченном количестве мест. Кэш память с прямым отображением - адрес пред­ставляется как набор трех компонент, составляющих группы старших, средних и младших разрядов адреса, соответственно тега, номера строки, смещения.

**33. Назначение и структура процессора. Система команд. Динамическая и статическая память.**

**34. Реальный режим работы процессора типа Intel-8086 и старше. Адресация памяти в реальном режиме. Виртуальный режим.**

**35. Защищенный режим работы процессора типа Intel-80286 и старше.**

Процессор Intel 80286 может работать в новом защищенном режиме (ЗР) и соответственно выполнять ряд задач. Адресная шина этого процессора была увеличена с 20 до 24 разрядов. Это привело к расширению адресного пространства с 1 Мб до 16Мб. Новый метод адресации позволил изолировать адресные пространства отдельных задач друг от друга. При этом прикладная программа, работающая в среде ОС, использующей защищенный режим не может случайно или намерено разрушить целостность самой ОС. Основным защищаемым ресурсом является память, в которой хранятся коды, данные, различные системные таблицы.

Защита памяти основывается на сегментации. Защищать требуется и совместно используемую аппаратуру, обращение к которой обычно происходит через операции ввода\вывода и прерывания. В ЗР процессор аппаратно реализует многие функции защиты, необходимые для построения многозадачной ОС, в том числе и механизма виртуальной памяти. В ЗР программа может записывать данные только в те области памяти, которые выделяет ей ОС. Изолирование адресных пространств задач исключает ситуацию, когда после запуска одним пользователем на недостаточно отлаженной программе приходится перезапускать всю систему.

Следующие модели процессоров Intel -386, -486, -586 (Pentium) были 32х разрядными. Адресное пространство было увеличено до 4Гб и в них была реализована концепция строчной виртуальной памяти, возможной только в ЗР.

Механизм строчной виртуальной памяти позволяет разместить часть ОП на диске, при этом размер виртуальной памяти, предоставляемой программам ограничивается размером свободного пространства на диске.

Основные преимущества:

1. Возможность непосредственной адресации памяти за пределами первого Мб.
2. Реализован механизм страничной виртуальной памяти, размер которой больше физической ОП установленной на компьютере.
3. Аппаратная поддержка мультизадачности.

**36. Таблица дескрипторов.**

Содержимое каждого регистра сегмента не соответствует непосредственно какому-либо участку ОП. Регистр сегмента, которого в защищенном режиме называют селектором становится индексом, указывающим на элемент таблицы, называемой таблицей дескрипторов. Каждый элемент этой таблицы характеризует 1 сегмент команд или 1 сегмент данных. Сегмент стека относится при этом к сегментам данных, т.к. он не содержит исполняемых команд.

Дескриптор содержит физический адрес начала сегмента памяти, его размер, а также другую информацию. Базовый адрес размещается в 24 битах, поэтому сегменты теперь не обязательно должны быть выровнены на адрес кратный 16.

Размер сегмента указывается в 16 битах и может принимать любые значения от 64 кб и меньше. В этом состоит фундаментальное отличие от реального режима, где каждый сегмент по умолчанию имеет размер 64 кб. После определения базового адреса к нему прибавляется значение смещения, размещаемого в 24 битах и процессор обращается к соответствующей ячейке памяти.

Очевидно, что данный метод сложнее чем для реального режима, но он остается «прозрачным» для программиста, т.к. механизм адресации в целом выглядит неизменно.

Таблица дескрипторов содержит описание блоков (сегментов) памяти: их расположение, права доступа и т.д.

Таблица дескрипторов бывает трех видов:

* Глобальная ТБ(GDT).
* Локальная ТД(LDT).
* Таблица дескрипторов прерываний(IDT).

Таблицы эти расположены в ОП. GDT существует в единственном экземпляре и обычно с ОС. LDT может быть много, это характерно для многозадачного режима, в котором каждой задаче соответствует своя LDT. LDT – это сегмент, имеющий свой дескриптор, который расположен в GDT.

Виртуальное адресное пространство делится на 2 равных части и при переключении задачи, GDT остается постоянной, а текущая LDT заменяется на LDT новой задачи.

**37. Принципы магнитной и оптико-магнитной записи информации. Системы ввода-вывода.**

**38. Обработка прерываний.**

**Прерывание** - это прекращение выполнения текущей команды или текущей последовательности команд для обработки некоторого события специальной программой - обработчиком прерывания, с последующим возвратом к выполнению прерванной программы. Событие может быть вызвано особой ситуацией, сложившейся при выполнении программы, или сигналом от внешнего устройства.

2 общих класса прерываний:

- внутренние (порождаются событиями внутри процессора, в основном самой программой точнее, ошибками в программе): деление на 0, переполнение

- внешние (инициируются сигналом, подаваемым от других компонентов системы): запрос на обслуживание со стороны какого-либо устройства.

Переход к процедуре прерывания осуществляется из любой программы, а после выполнения процедуры прерывания обязательно происходит процесс возврата в прерванную программу. Перед обращением к процедуре прерывания должно быть сохранено состояние всех регистров и флагов, используемых процедурой прерывания, а после окончания прерывания эти регистры должны быть восстановлены.

Прерывание вынуждает процедуру прекратить выполнение одной последовательности команд и начать выполнение другой. При этом адрес очередной команды, которая должна была выполняться запомнится. Адрес команды, которая должна была выполниться после возникновения прерывания выбирается из таблицы, хранящейся в начальной области памяти. Эта таблица называется **таблица векторов прерывания**. В ней записано 256 адресов.

Когда устройство вызывает прерывание процессора, оно сообщает ему какой адрес из таблицы следует выполнить для перехода к новой последовательности команд.

***Аппаратные***прерывания используются для организации взаимодействия с внешними устройствами. Запросы аппаратных прерываний поступают на специальные входы микропроцессора. Они бывают:

1. маскируемые, которые могут быть замаскированы программными средствами компьютера;
2. немаскируемые, запрос от которых таким образом замаскирован быть не может.

***Программные***прерывания вызываются следующими ситуациями:

1. особый случай, возникший при выполнении команды и препятствующий нормальному продолжению программы (переполнение, нарушение защиты памяти, отсутствие нужной страницы в оперативной памяти и т.п.);
2. наличие в программе специальной команды прерывания INT n, используемой обычно программистом при обращениях к специальным функциям операционной системы для ввода-вывода информации.

Каждому запросу прерывания в компьютере присваивается свой номер (*тип прерывания*), используемый для определения адреса обработчика прерывания.

При поступлении запроса прерывания компьютер выполняет следующую последовательность действий:

1. определение наиболее приоритетного незамаскированного запроса на прерывание (если одновременно поступило несколько запросов);
2. определение типа выбранного запроса;
3. сохранение текущего состояния счетчика команд и регистра флагов;
4. определение адреса обработчика прерывания по типу прерывания и передача управления первой команде этого обработчика;
5. выполнение программы - обработчика прерывания;
6. восстановление сохраненных значений счетчика команд и регистра флагов прерванной программы;
7. продолжение выполнения прерванной программы.

Этапы 1-4 выполняются аппаратными средствами ЭВМ автоматически при появлении запроса прерывания. Этап 6 также выполняется аппаратно по команде возврата из обработчика прерывания.

**39. Сравнительная характеристика микропроцессоров типа CISC и RISC.**

Архитектура набора команд служит границей между аппаратурой и программным обеспечением и представляет ту часть системы, которая видна программисту или разработчику компиляторов.

Двумя основными архитектурами набора команд, используемыми компьютерной промышленностью на современном этапе развития вычислительной техники являются архитектуры CISC и RISC. Основоположником CISC-архитектуры можно считать компанию IBM с ее базовой архитектурой /360, ядро которой используется с 1964 года и дошло до наших дней, например, в таких современных мейнфреймах как IBM ES/9000.

Лидером в разработке микропроцессоров c полным набором команд (CISC - Complete Instruction Set Computer) считается компания Intel со своей серией x86 и Pentium. Эта архитектура является практическим стандартом для рынка микрокомпьютеров. Для CISC-процессоров характерно: сравнительно небольшое число регистров общего назначения; большое количество машинных команд, некоторые из которых нагружены семантически аналогично операторам высокоуровневых языков программирования и выполняются за много тактов; большое количество методов адресации; большое количество форматов команд различной разрядности; преобладание двухадресного формата команд; наличие команд обработки типа регистр-память.

Основой архитектуры современных рабочих станций и серверов является архитектура компьютера с сокращенным набором команд (RISC - Reduced Instruction Set Computer). Зачатки этой архитектуры уходят своими корнями к компьютерам CDC6600, разработчики которых (Торнтон, Крэй и др.) осознали важность упрощения набора команд для построения быстрых вычислительных машин. Эту традицию упрощения архитектуры С. Крэй с успехом применил при создании широко известной серии суперкомпьютеров компании Cray Research. Все они придерживались архитектуры, отделяющей команды обработки от команд работы с памятью, и делали упор на эффективную конвейерную обработку. Система команд разрабатывалась таким образом, чтобы выполнение любой команды занимало небольшое количество машинных тактов (предпочтительно один машинный такт). Сама логика выполнения команд с целью повышения производительности ориентировалась на аппаратную, а не на микропрограммную реализацию. Чтобы упростить логику декодирования команд использовались команды фиксированной длины и фиксированного формата.

Среди других особенностей RISC-архитектур следует отметить наличие достаточно большого регистрового файла (в типовых RISC-процессорах реализуются 32 или большее число регистров по сравнению с 8 - 16 регистрами в CISC-архитектурах), что позволяет большему объему данных храниться в регистрах на процессорном кристалле большее время и упрощает работу компилятора по распределению регистров под переменные. Для обработки, как правило, используются трехадресные команды, что помимо упрощения дешифрации дает возможность сохранять большее число переменных в регистрах без их последующей перезагрузки.

Развитие архитектуры RISC в значительной степени определялось прогрессом в области создания оптимизирующих компиляторов. Именно современная техника компиляции позволяет эффективно использовать преимущества большего регистрового файла, конвейерной организации и большей скорости выполнения команд. Современные компиляторы используют также преимущества другой оптимизационной техники для повышения производительности, обычно применяемой в процессорах RISC: реализацию задержанных переходов и суперскалярной обработки, позволяющей в один и тот же момент времени выдавать на выполнение несколько команд.

**40. Особенности RISC архитектуры.**

**RISC** – Reduced Instruction Set Computer – архитектура компьютера с сокращенным набором команд. В ходе анализа частоты выполнения тех или иных команд выяснено, что 40% команд используются крайне редко. RISC-архитектура предполагает реализацию в ЭВМ сокра­щенного набора простейших, но часто употребляемых команд, что позволяет упростить аппаратурные средства процессора и благодаря этому получить возможность повысить его быстро­действие.

Современные процессоры типа RISC характеризуются следующими особенностями:

1. упрощенный набор команд, имеющих одинаковую длину. Например, выполнение типичной команды можно разделить на этапы:
   * выборка команды
   * декодирование команды
   * выполнение операции
   * обращение к памяти
   * запоминание результата

Все команды в RISC имеют одну структуру, количество команд – обычно не более 50-100.

1. Большинство команд выполняется за 4-5 тактов процессора.
2. Отсутствуют макрокоманды, усложняющие структуру процессора и уменьшающие скорость его работы.
3. Взаимодействие с оперативной памятью ограничивается операциями пересылки данных.
4. Уменьшено число способов адресации (не используется косвенная, обычно 2-3 простых способов)
5. Используется конвейер команд.
6. Применяется высокоскоростная память.

В ЭВМ с RISC машинным циклом называют время, в течение которого производится выборка двух операндов из регистров, выполнение операции в АЛУ и запоминание результата в регист­ре. Большинство команд в RISC являются быстрыми командами типа «регистр - регистр» и выполняются без обращений к ОП. Обращения к ОП сохраняются лишь в командах загрузки регистров из памяти и запоминания в ОП. Чтобы это было возможным, про­цессор должен содержать достаточно большое число общих регистров.

Новый подход к архитектуре команд процессора значительно сократил площадь, требуемой для него на кристалле интегральной микросхемы. Это позволило резко увеличить число регистров (в типовых RISC-процессорах реализуются 32 или большее число регистров по сравнению с 8 - 16 регистрами в CISC-архитектурах). В результате процессор стал на 20-30% реже обращаться к оперативной памяти. Упростилась топология процессора, сократились сроки ее разработки, она стала дешевле.

Особенностью RISC архитектуры является механизм перекрывающихся окон, предназначенный для уменьшения числа обращений к оперативной памяти и межрегистровых передач, что способствует увеличению производительности ЭВМ. Процедурам динамически выделяются небольшие группы регистров фиксированной длины (регистровые окна). Окна последовательно выполняемых процедур перекрываются, благодаря чему возможна передача параметров из одной процедуры в другую. При этом не возникает необходимость передачи содержимого регистра в память.

Окно состоит из трех подгрупп регистров:

1. первая подгруппа содержит параметры, переданные данной процедуре от процедуры, вызвавшей её, и результаты для вызывающей процедуры при возврате в неё.
2. вторая подгруппа содержит локальные переменные процедуры.
3. третья является буфером для двухстороннего обмена.

В 1989 фирме Intel удалось на основе RISC-архитектуры создать однокри­стальный микропроцессор 80860, который практически представ­ляет собой кремниевый эквивалент суперЭВМ Cray-1.

В настоящее время многие архитектуры процессоров являются RISC-подобными, к примеру, ARM, DEC Alpha, SPARC, AVR, MIPS, POWER и PowerPC. Наиболее широко используемые в настольных компьютерах процессоры архитектуры x86 ранее являлись CISC-процессорами, однако новые процессоры, начиная с Intel 486DX, являются CISC-процессорами с RISC-ядром. Они непосредственно перед исполнением преобразуют CISC-инструкции x86-процессоров в более простой набор внутренних инструкций RISC.

**41. Архитектура суперскалярных процессоров. Предварительная выборка и предсказание переходов.**

**Суперскалярная архитектура** – Способность выполнения нескольких машинных инструкций за один такт процессора. Появление этой технологии привело к существенному увеличению производительности. Основная идея, определяющая развитие суперскалярных микропроцес­соров, состоит в построении возможно большего количества парал­лельных структур при сохранении традиционных последовательных про­грамм. Это означает, что компиляторы и аппаратура микропроцессора сами, без вмешательства программиста, обеспечивают загрузку параллель­но работающих функциональных устройств микропроцессора. В соответствии с моделью последовательного программирования, про­граммы пишутся в предположении, что команды будут выполнены в том же порядке, в каком они представлены в программе. Однако с целью достиже­ния большей эффективности современные процессоры пытаются выполнять несколько команд одновременно и в некоторых случаях в порядке, отличном от их исходной последовательности в программе. Это переупорядочение мо­жет быть выполнено в трансляторе и (или) в аппаратных средствах во время выполнения. Суперскалярные и VLIW-процессоры принадлежат классу ар­хитектур, которые используют параллельность уровня команды (ILP). ILP-процессоры и компиляторы обычно преобразуют полностью упо­рядоченное множество команд исходной программы в частично упорядо­ченное множество, структурированное зависимостями по данным и управ­лению. Зависимости по управлению (которые проявляются как переходы по условию) представляют главное препятствие высокопараллельному выполнению потому, что эти зависимости должны быть установлены пре­жде, чем будут выполнены все последующие команды. Текст последовательной программы, представленной на языке высо­кого уровня, компилируется в машинный код, отражающий статическую структуру программы, т.е. упорядоченное множество команд (инструк­ций) в памяти компьютера. Процесс выполнения программы с конкрет­ными наборами входных данных может быть представлен динамической структурой программы, т.е. множеством последовательностей инструк­ций в порядке их исполнения. Повысить степень параллелизма программы можно изменяя соответст­вующим образом ее статическую или динамическую структуру. Поскольку статическая структура программы однозначно соответствует ее исходному тексту (в предположении неизменности компилятора), то изменение статиче­ской структуры сводится к изменению исходного кода, что, в общем случае, не всегда возможно.

Динамическая же структура программы может быть из­менена при неизменной статической структуре. И главной целью такого из­менения должно быть повышение степени параллельного исполнения команд. Допустимые границы преобразования динамической структуры про­граммы задают существующие на множестве инструкций отношения: за­висимость по управлению и зависимость по данным. При описании архи­тектур ССП часто используется модель окна ис­полнения. При исполнении программы микропроцессор как бы продви­гает по статической структуре программы окно исполнения. Команды в окне могут исполняться параллельно, если между ними нет зависимости. Для устранения зависимостей, вызванных командами переходов, исполь­зуется метод предсказания, позволяющий извлекать и условно исполнять команды предсказанного перехода. Если позднее обнаруживается, что пред­сказание было сделано, верно, то результаты условно исполненных команд принимаются. Если предсказание было ошибочным, состояние процессора восстанавливается на момент принятия решения о выполнении перехода. Все виды зависимостей по данным могут быть классифицированы по типу ассоциаций: RAR - "чтение после чтения", WAR - "запись после чте­ния" и WAW - "запись после записи", RAW - "чтение после записи". Действительной зави­симостью является только "чтение после записи" (RAW), так как необходи­мо прочитать предварительно записанные новые данные, а не старые.

**42. Архитектура ЭВМ с длинным командным словом.**

Архитектура машин с очень длинным командным словом (VLIW - Very Long Instruction Word) позволяет сократить объем оборудования, требуемого для реализации параллельной выдачи нескольких команд, и потенциально чем большее количество команд выдается параллельно, тем больше эта экономия. Например, суперскалярная машина, обеспечивающая параллельную выдачу двух команд, требует параллельного анализа двух кодов операций, шести полей номеров регистров, а также того, чтобы динамически анализировалась возможность выдачи одной или двух команд и выполнялось распределение этих команд по функциональным устройствам. Хотя требования по объему аппаратуры для параллельной выдачи двух команд остаются достаточно умеренными, и можно даже увеличить степень распараллеливания до четырех (что применяется в современных микропроцессорах), дальнейшее увеличение количества выдаваемых параллельно для выполнения команд приводит к нарастанию сложности реализации из-за необходимости определения порядка следования команд и существующих между ними зависимостей.

Архитектура VLIW базируется на множестве независимых функциональных устройств. Вместо того, чтобы пытаться параллельно выдавать в эти устройства независимые команды, в таких машинах несколько операций упаковываются в одну

очень длинную команду. При этом ответственность за выбор параллельно выдаваемых для выполнения операций полностью ложится на компилятор, а аппаратные средства, необходимые для реализации суперскалярной обработки, просто отсутствуют.

VLIW-команда может включать, например, две целочисленные операции, две операции с плавающей точкой, две операции обращения к памяти и операцию перехода. Такая команда будет иметь набор полей для каждого функционального устройства, возможно от 16 до 24 бит на устройство, что приводит к команде длиною от 112 до 168 бит.

**43. Основные классы современных параллельных компьютеров. MPP, SMP, NUMA, PVP.**

Основным параметром классификации параллельных компьютеров является наличие общей (SMP) или распределенной памяти (MPP). Нечто среднее между SMP и MPP представляют собой NUMA-архитектуры, где память физически распределена, но логически общедоступна. Кластерные системы являются более дешевым вариантом MPP. При поддержке команд обработки векторных данных говорят о векторно-конвейерных процессорах, которые, в свою очередь могут объединяться в PVP-системы с использованием общей или распределенной памяти.

Все большую популярность приобретают идеи комбинирования различных архитектур в одной системе и построения неоднородных систем.

При организациях распределенных вычислений в глобальных сетях (Интернет) говорят о мета-компьютерах, которые, строго говоря, не представляют из себя параллельных архитектур.

**Массивно-параллельные системы (MPP)**

*Архитектура*:

Система состоит из однородных *вычислительных узлов*, включающих:

1. один или несколько центральных процессоров (обычно RISC),
2. локальную память (прямой доступ к памяти других узлов невозможен),
3. коммуникационный процессор или сетевой адаптер
4. иногда - жесткие диски (как в SP) и/или другие устройства В/В

К системе могут быть добавлены специальные узлы ввода-вывода и управляющие узлы. Узлы связаны через некоторую коммуникационную среду (высокоскоростная сеть, коммутатор и т.п.)

*Примеры*: IBM RS/6000 SP2, Intel PARAGON/ASCI Red, SGI/CRAY T3E, Hitachi SR8000, транспьютерные системы Parsytec.

*Масштабируемость*: Общее число процессоров в реальных системах достигает нескольких тысяч (ASCI Red, Blue Mountain).

*Операционная система*: Существуют два основных варианта:

1. Полноценная ОС работает только на управляющей машине (front-end), на каждом узле работает сильно урезанный вариант ОС, обеспечивающие только работу расположенной в нем ветви параллельного приложения. Пример: Cray T3E.

2. На каждом узле работает полноценная UNIX-подобная ОС (вариант, близкий к кластерному подходу). Пример: IBM RS/6000 SP + ОС AIX, устанавливаемая отдельно на каждом узле.

*Модель программирования*: Программирование в рамках модели передачи сообщений ( MPI, PVM, BSPlib).

**Симметричные мультипроцессорные системы** (SMP)

*Архитектура:* Система состоит из нескольких однородных процессоров и массива общей памяти (обычно из нескольких независимых блоков). Все процессоры имеют доступ к любой точке памяти с одинаковой скоростью. Процессоры подключены к памяти либо с помощью общей шины (базовые 2-4 процессорные SMP-сервера), либо с помощью crossbar-коммутатора (HP 9000). Аппаратно поддерживается когерентность кэшей.

*Примеры:* HP 9000 V-class, N-class; SMP-cервера и рабочие станции на базе процессоров Intel (IBM, HP, Compaq, Dell, ALR, Unisys, DG, Fujitsu и др.).

*Масштабируемость:* Наличие общей памяти сильно упрощает взаимодействие процессоров между собой, однако накладывает сильные ограничения на их число - не более 32 в реальных системах. Для построения масштабируемых систем на базе SMP используются кластерные или NUMA-архитектуры.

*Операционная система:* Вся система работает под управлением единой ОС (обычно UNIX-подобной, но для Intel-платформ поддерживается Windows NT). ОС автоматически (в процессе работы) распределяет процессы/нити по процессорам (scheduling), но иногда возможна и явная привязка.

*Модель программирования:* Программирование в модели общей памяти. (POSIX threads, OpenMP). Для SMP-систем существуют сравнительно эффективные средства автоматического распараллеливания.

**Системы с неоднородным доступом к памяти** (NUMA)

*Архитектура:* Система состоит из однородных базовых модулей (плат), состоящих из небольшого числа процессоров и блока памяти. Модули объединены с помощью высокоскоростного коммутатора. Поддерживается единое адресное пространство, аппаратно поддерживается доступ к удаленной памяти, т.е. к памяти других модулей. При этом доступ к локальной памяти в несколько раз быстрее, чем к удаленной. В случае, если аппаратно поддерживается когерентность кэшей во всей системе (обычно это так), говорят об архитектуре cc-NUMA (cache-coherent NUMA).

*Примеры:* HP HP 9000 V-class в SCA-конфигурациях, SGI Origin2000, Sun HPC 10000, IBM/Sequent NUMA-Q 2000, SNI RM600.

*Масштабируемость:* Масштабируемость NUMA-систем ограничивается объемом адресного пространства, возможностями аппаратуры поддежки когерентности кэшей и возможностями операционной системы по управлению большим числом процессоров. На настоящий момент, максимальное число процессоров в NUMA-системах составляет 256 (Origin2000).

*Операционная система:* Обычно вся система работает под управлением единой ОС, как в SMP. Но возможны также варианты динамического "подразделения" системы, когда отдельные "разделы" системы работают под управлением разных ОС (например, Windows NT и UNIX в NUMA-Q 2000).

*Модель программирования:* Программирование в модели общей памяти. (POSIX threads, OpenMP). Для SMP-систем существуют сравнительно эффективные средства автоматического распараллеливания.

**Параллельные векторные системы** (PVP)

*Архитектура:* Основным признаком PVP-систем является наличие специальных векторно-конвейерных процессоров, в которых предусмотрены команды однотипной обработки векторов независимых данных, эффективно выполняющиеся на конвейерных функциональных устройствах. Как правило, несколько таких процессоров (1-16) работают одновременно над общей памятью (аналогично SMP) в рамках многопроцессорных конфигураций. Несколько таких узлов могут быть объединены с помощью коммутатора (аналогично MPP).

*Операционная система:* NEC SX-4/SX-5, линия векторно-конвейерных компьютеров CRAY: от CRAY-1, CRAY J90/T90, CRAY SV1, серия Fujitsu VPP.

*Модель программирования:* Эффективное программирование подразумевает векторизацию циклов (для достижения разумной производительности одного процессора) и их распараллеливание (для одновременной загрузки нескольких процессоров одним приложением).