**Архитектура ЭВМ и систем.**

**24. 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 называние 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.

**25. 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.

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

**38. 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.

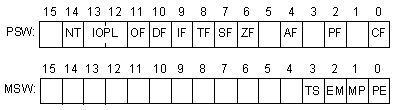
1. **Регистры АЛУ микропроцессора. РОН, сегментные регистры, регистр флагов.**

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

Регистры общего назначения. Они 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 некот объекта в памяти. Ecx– регистр – счетчик, применяется в командах, произв-х нек-е повтор-ся действие. Esi– индекс источника, этот регистр в цепочных операциях содержит текущий адрес эл-та цепочки устр-ва. Edi– индекс приемника, в цепочных операциях сод-т регистр. Esp – регистр – указатель стека, содержит указатель вершины стека в текущем эл-те. Ebp – регистр базы кадра стека, предназначен для орг-ии произв-го доступа к данным внутри стека. Стек – структура данных, кот поддерживает на программно-аппаратном уровне в арх микропр-ра.

Сегментные регистры*.* Микропр-р аппаратно поддерживает структ орг-ю программы в виде 3-х частей, кот наз-ют сегментами, соот-но память наз-ся сегментная. МП поддерж-ет след. Типы сегментов: 1)Cs-сегм регистр кода(содержит команды программы). 2) Ds - сегм регистр данных(сод-т обрабатываемые данные). 3) Ss-сегм регистр стека. (область памяти, кот наз-ют стеком), 4) Es, gs, fc- доп сегмент данных. *Регистры состояния и назначения.* В микропр-р вкл нескоко регистров, кот постоянно содержат информ о состоянии самого микропр-ра, программы, команды, кот в данный момент производится работа.Делятся на регистр флагов и регистр указателя команды. Регистр флагов – определяет и позволяет изменять состояние процессора. Каждый его бит называется «*флагом»* и может быть установлен или сброшен в зависимости от результата выполнения очередной команды. Также некоторые команды могут выполнять какие-то действия только в том случае, если установлен определенный флаг.

**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 было ли перенос разрядов. Соответсвенно если было, то определенный флаг(СF) установится в 1, если нет, то останется 0. Дальше, анализируя флаг с помощью команд условного перехода можно к примеру обработать перенос.

mov al, 240d|mov ah, 16d|add al, ah|jc \_label

al= 11110000|ah=10000|

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 рез0та выполнения логических или арифметических команд. Это даёт возможность анализировать состояние вычисл процесса и реагировать на него спомощью команд условных переходов и вызова попрограмм. 2) флаг управления DF, нах-ся в 10-ом виде р-ра и исп-ся цепочными командами. 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 сущ-ют спец-ые 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 проц-ра.

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

памятью ЭВМ наз-ся сов-ть устройств, кот служат для запоминания, хран-я, и выдачи инф-ии.

отдельные уср-ва, входящие в эту сов-ть наз-ся запоминающие устр-ва.

Класс-ция ЗУ:

1)по типу ЗУ: - полупроводниковые, - магнитные, - конденсаторные, - оптоэлектронные, - голографические, - креагенные

2) по функц. назначения: - ОЗУ, - СОЗУ(свероперативные), - внешние зу, - промежуточные ЗУ

3) по способу орг-ции обр-я: - с последовательным поиском, - с прямым доступом, - с непосредственным доступом. – ассоциативные, - стековые, - магазинные

4) по хар-ру считывания: - с разрушением инф-ции, - без разрушения инф-ции

5) по способу хран-я: - статические, - динамические

6) по способу орг-ции: - однокоординатные, - двух-коорд, - 3-коорд, - 2-3-коорд

Хар-ки памяти:

Емкость памяти опред-ся максимальным кол-вом данных, кот могут иметь границы

Удельная емкость – отношение емкости ЗУ к её физич объему

Быстродействие памяти опр-ся продолжительностью операции обращения, т е временем, затрачиваемым на поиск нужной единицы инф-ции в памяти на её считывание или на поиск места в памяти, предназнач.для хран-я данный ед-цы инф-и и на её запись в память.

продолжит-ть обращения к памяти при считывании опр-ся по формуле:

, где - время доступа, определяющееся промежутком времени м/ду моментом начала операции обращения при счит-нии до мом-та когда стан-ся возможным доступ к данной ед-це инф-ии, - продолж-ть самого физич.процесса обнаружения и фиксации состояния соответствующих запом.эл-в.

В некот устр-вах считывание инф-ии сопров-ся её разрушением. В этом сл-е:, где – время доступа при записи, - время подготовки, расходуемое на поведение в исх.сост-е и запом.эл-в им участка пов-ти носителя инф-ии для записи опред.ед-цы инф-ии, -время занесения инф-ии.

**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.

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

По способу обращения ЗУ делятся:

- с последов.поиском

- с прямым доступом

- с несосредств.доступом или адресные

- ассоциативные

- стековые

- магазинные

Организация опер.памяти

В адресном ЗУ каждый эл-т памяти имеет адрес, соответствующий его пространств.расположению в запоминающей среде. Поэтому обращение к соответств.эл-ту производится в соответствии с кодом его адреса. В ОЗУ после приема кода осущ-ся его дешифрация, после чего следует выборка из эл-та конкретной группы битов или слов.

*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.

Стек – область памяти, предназначенная для временного хран-я произвольных данных. Стек традиционно исп-ся для сохранения сожержимого регистров, использ=х пр-мой, перед вызовом подпрограммы. Др. распростр-й прием – прием-передача продпрограмме требуемых ею пар-ров через стек.

Отличит-й особ-тью явл. Своеобразный порядок в выборке содержащихс в нем данных. В любой момент времени в стеке доступен только верхний эл-т , т е эл-т загруженный в стек последним. Выгрузка из стека ерхнего эл-та делает доступным след. Эл-т. Эл-ты стека располагаются в обл.памяти отведенной под стек со дна стека, т е с его максимального адреса. Адрес верхнего доступного эл-та хранится в регистре указателя стека SP. Как любая др.область памяти стек должен входить в какой-то сегмент или образовывать отдельный сегмент. Сегментный адрес этого сегмента помещается в сегм.регистр стека SS. Пара регистров SS:SP описывает адрес доступной ячейки стека.

Кэш память представляет собой буферное ЗУ, работающее со скоростью обеспечивающей функц-е 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 наз-ся соот-но попаданием или промахом. Частота попадания и промаха явл-ся одной из важнейших хар-к 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 наз-ся полностью ассоц-ой, если некот блок осн памяти может располагаться в любом месте кэш памяти. И наз-ся частично ассоц-ной, если блок памяти может располагаться на ограниченном кол-ве мест. Кэш память наз-ся памятью с прямым отображением. При исп-ии кэш-памяти с прямым отображением адрес пред­ставляется как набор трех компонент, составляющих группы старших, средних и младших разрядов адреса, соответственно тега, номера строки, смещения. Ассоц-я кэш-память использует 2-хкомпонентное представле­ние адреса: группа старших разрядов трактуется как тег, а группа млад­ших разрядов - как смещение в строке. Нахождение строки в кэше определяется совпадением тега-строки со зна­чением тега адреса. По­этому при определении нахождения требуемой строки в кэш-памяти необ­ходимо сравнение тега адреса с тегами всех строк кэша. При отсутствии необходимой строки в кэш-памяти одна из его строк должна быть заменена на требуемую. Кэш-память состоит из набора ассоц-х блоков кэш-памяти. Средняя компонента адреса задает в отличие от прямо адресуе­мой кэш-памяти не номер строки, а номер одного из ассоциативных бло­ков. При поиске данных ассоциативное сравнение тегов выполняется толь­ко для набора блоков, номер, которого совпадает со средней компонентой адреса. По количеству *n* строк в наборе кэш-память называется *n*-входовой. Соответствие между данными в оперативной памяти и кэш-памяти обес­печивается внесением изменений в те области оперативной памяти, для которых данные в кэш-памяти подверглись модификации. Суще­ствует несколько способов такой модификации: 1)внесение изменений в оперативную память сразу после изменения данных в кэше. При этом процессор простаивает в ожидании завершения записи в основную память. В основной памяти под­держивается правильная копия данных кэша, и при замене строк не требу­ется никаких дополнительных действий. Кэш-память, работающая в та­ком режиме, называется памятью со сквозной записью. 2)Предполагает отображение изменений в основной па­мяти только в момент вытеснения строки данных из кэша. Если данные по адресу памяти, в кот необходимо произвести запись, находятся в кэш-памяти, то идет запись только в кэш-память. При отсутствии данных в кэш-памяти производится запись в основную память. Такой режим рабо­ты кэша получил название обратной записи. Существуют также промежуточные варианты, при которых запросы на изменение в основной памяти буферизуются и не за­держивают процессор на время операции записи в память. Эта запись вы­полняется по мере возможности доступа контроллера кэш-памяти к ос­новной памяти.

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

В больш-ве ЭВМ принципы Фон-Неймана реализованы в след.виде:

1)ОП организована как сов-ть машинных слов фиксир.доины, например первые ЭВМ имели разрядность 8, 16, 32, 64. Существовали 45-разрядные, 35 –разр.ЭВМ

2)ОП образует единое адресное простр-во. Адреса машин.слов возрастают от младших к старшим.

3)В ОП размещаются как данные, так и пр-мы

4) Команды выполняются в естественной посл-ти по возрастанию адресов в ОП, пока не встретится команда ветвления

5) Центр.процессор может произвольно обращаться к любым адресам в ОП для выборки или записи в машин.слова чисел и команд

Команда – описание операции, кот необх-мо выполнить.

Каждая команда начинается с кода операции(КОП), содержит необходимые адреса, хар-ся форматом, кот опр-ет структуру команды, её орг-цию, код, длину, метод расположения адресов. Длина различ.команд может быть одинаковой или разной. Команды подразделяются на арифметич., логич, ввода-выв., передачи данных.

Классы команд:

1. Класс обр-ки д-х:

1.1 короткие операции

1.1.1 логические: логич сложение, лог умножение, инверсия, лог сравление

1.1.2 арифм.: слодение операндов, вычитание, арифм сравнение

1.2 длинные операции: сложение, умн-е

2 операции управления: безусловный переход, условный пер-д

3 операции обращения к вне шустр-вам: требование на запись или чтение

типовая стр-ра 3-адресной команды: КОП-А1-А2-А3. А2 и А3 – адреса ячеек где расположены 1 и 2 числа, А1 – адрес ячейки регистра, куда необходимо поместить результат, КОП-код оп-ции

типовая стр-ра 2-адресной команды: А1 – ад-с ячейки, где хр-ся первое из чисел, участ-х в оп-ции и куда будет записан рез-т, А2-//- 2 число

типовая стр-ра 1-адресной команды: А1 – может обозн-ть адрес ячейки где хр-ся 1 из чисел, уч-х в оп-ции либо адрес ячейки для рез-та

безадресная команда сод-т КОП

Каждая команда сод-т эл-ты, определяющие:

1. Что делать(код)
2. Объекты, над кот надо что то сделать
3. Как делать(типы операндов)

Максим.длина команды 15байт

Префиксы(необязат поле – 1 байт)

- префикс замены сегмента

- префикс разрядности адреса

- префикс разрядности операнда

- пр-с повторения

Каждая команда вып-ся за 1 или неск тактов.

Послед-ть взаимосвяз-х команд именуется макрокомандой. Исп-е макрокоманд упрощает программир-е и обеспеч-ет механизм вставки добавления пр-мы.

Цикл пр-ра – это период времени, за кот осущ-ся вып-е команды исход.пр-мы в машинном виде.

Состоит из неск.тактов.

Такт работы пр-ра – период времени между соседними импульсами генер-ра тактовых импульсов.

Процедура, соответ. Такту реализуется определенной логич.цепью проц-ра и наз-ся микропрог-й.

Существуют две стратегии распределения оперативной памяти, как и любого ресурса: статическое и динамическое распределение.

При ***статическом распределении*** вся необходимая оперативная память выделяется процессу в момент его порождения. При этом память выделяется единым блоком необходимой длины, начало которого определяется базовым адресом. Программа пишется в адресах относительно начала блока, а физический адрес команды или операнда при выполнении программы формируется как сумма базового адреса блока и относительного адреса в блоке. Значение базового адреса устанавливается при загрузке программы в оперативную память. Так как в разных программах используются блоки разной длины, то при таком подходе возникает проблема фрагментации памяти, то есть возникают свободные участки памяти, которые невозможно без предварительного преобразования использовать для вычислительного процесса.

В кч-ве запоминающ эл-та исп-ся статич триггер, состояий из 4-6 транзисторов

«+» обладает большим быстродействием

«-»Плотность упаковки таких микросхем меньше пл-ти уп-ки динамич памяти, стоимость выше

При ***динамическом распределении памяти*** каждой программе в начальный момент выделяется лишь часть от всей необходимой ей памяти, а остальная часть выделяется по мере возникновения реальной потребности в ней.Такой подход базируется на следующих предпосылках.

В кач запомин эл-та исп-ся простейшая сборка, состоящая из 1 транзистора и 1 конденсатора

«+»высокая плотность интеграции(кол-во числа запом.эл-в на ед-цу пов-ти), малое потр-е энергии

«-»для того, чтобы сохранить записанную инф-ю запом.эл-т должен постоянно регенерироваться

Во-первых, при каждом конкретном исполнении в зависимости от исходных данных некоторые части программы (до 25% ее длины) вообще не используются. Следует стремиться к тому, чтобы эти фрагменты кода не загружались в ОП.

Во-вторых, исполнение программы характеризуется так называемым принципом локальности ссылок. Он подразумевает, что при исполнении программы в течение некоторого относительно малого интервала времени происходит обращение к памяти в пределах ограниченного диапазона адресов (как по коду программы, так и по данным). Следовательно, на протяжении этого времени нет необходимости хранить в ОП другие блоки программы.

При этом системные средства должны отслеживать возникновение требований на обращение к тем частям программы, которые в данный момент отсутствуют в ОЗУ, выделять этой программе необходимый блок памяти и помещать туда из внешнего ЗУ требуемую часть программы. Для этого может потребоваться предварительное перемещение некоторых блоков информации из ОЗУ во внешнюю память. Данные перемещения должны быть скрыты от пользователя и в наименьшей степени замедлять работу его программы.

Перемещение блоков информации из ОЗУ во внешнюю память с целью освобождения места для новой информации происходит обычно по одному из следующих алгоритмов:

LRU (least recently used) - наиболее давно не использовавшийся;

FIFO - самый давний по пребыванию в ОЗУ;

Random - случайным образом.

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

**Реальный режим**

Первоночально IBM PC исп-ся пр-р 8086, кот имел 16-разрядные команды и мог адресовать один Мб памяти используя 20 разрядов для адреса. ПО – DOS, 1-е варианты Windows. Более поздние процессоры – Intel 80286 также могли выполнять те же самые 16-разр команды, но намного быстрее. 16-разр режим, в котором выполнялись команды пр-ра Intel 8086 и Intel 80286 был назван реальным режимом. Для ПО этого типа обычно используется однозадачный режим, т.е. выполняется только 1 программа, нет никакой встроенной защиты для предотвращения перезаписи ячеек памяти одной программы или даже ОС. Это означает что при выполнении в реальном режиме нескольких программ вполне м.б. испорчены данные или код одной из них, соот-но это может привести к краху систему или остановку.

Реальный режим процессора Intel 8086

П0 при работе в этом режиме можно разбить на лог блоки по 64 Кб (сегменты), причем каждый сегмент может и начинаться с адреса кратного 16 байтам т.о. 1-сегмент имеет нач адрес 0, 2-й адрес 16 и т.д. Это удобно при организации совместного доступа к командам и данным разл программ. Доступ к каждой ячейке памяти происходит путем указания знач регистра сегмента, кот определяет лог блок размером 64 Кб и положение этого адреса внутри лог блока(смещение). Микропроцессор исп-ет 4 сегментных регистра. Каждый регистр при этом имеет размер, равный 1-му слову(16 разр)

1-регистр сегмента команд cs code segment, указ-ий на сегмент созд-ий текущую исп-мую пр-му

2-регистр сегмента данных ds data segment, указ-ий на данные

3-регистр доп сегмента es extra segment, указ-ий на доп данные

4-регистр сегмента стека ss steck segment, указ-ий на стэк

Содержание каждого из этих регистров однозначно связано с местом в памяти соот-го сегмента. Его адрес получается приписыванием справа 0000 (4-х двоичных 0) знач-го сегмента, что соот-ет умножению на 16. Полученное 20 битовое значение, представляет собой адрес начала(базовый адрес) сегмента физ памяти. Для определения реального адреса команды или данных процессор добавляет значение смещения к базовому адресу. В реальном режиме не существует никакого механизма защиты, поэтому любая прог-ма может обратиться к любой ячейке в пределах 1 Мб, включая область экрана и область расположения ОС

Виртуальный реальный режим

В проц-х Intel 386 и старше реал-н так наз-емый режим вирт.проц-ра Intel 8086. Режим реал-ся в рамках защищенного режима, т е проц-р может переключаться в вирт режим только из защищенного. В вирт режиме пр-р способен выполнять программы составленные для проц-ра 8086, находясь в защищенном режиме и используя аппарат ср-ва защищ.режима: мультизадачность, изолирование адресных простр-в отдельных задач др от друга, страничную вирт память.

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

Процессор Intel 80286 может работать в новом защищенном режиме 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.

Осн преимущества:

1. Возм-ть непосредственной адресациипамяти за пределами первого Мб
2. Реализован механизм страничной вирт памяти, размер кот больше физич.ОП установленной на компе
3. Аппаратная подд-ка мультизадачности
4. **Таблица дескрипторов.**

Содержимое каждого регистра сегмента не соот-ет непосредственно какому либо участку ОП. Регистр сегмента, кот в защищ режиме наз-ют селектором становится индексом указывающим на эл-т таблицы, называемой таблицей дескрипторов. Каждый эл-т этой т-цы хар-ет 1 сегмент команд или 1 сегмент д-х. Сегмент стека отн-ся при этом к сегментам д-х, т к он не содержит исполняемых команд.

Дескриптор содержит физич адрес начала сегмента памяти, его размер, а т же др инф-цию. Базовый адрес размещается в 24 битах, поэтому сегменты теперь не обязательно должны быть выровнены на адрес кратный 16.

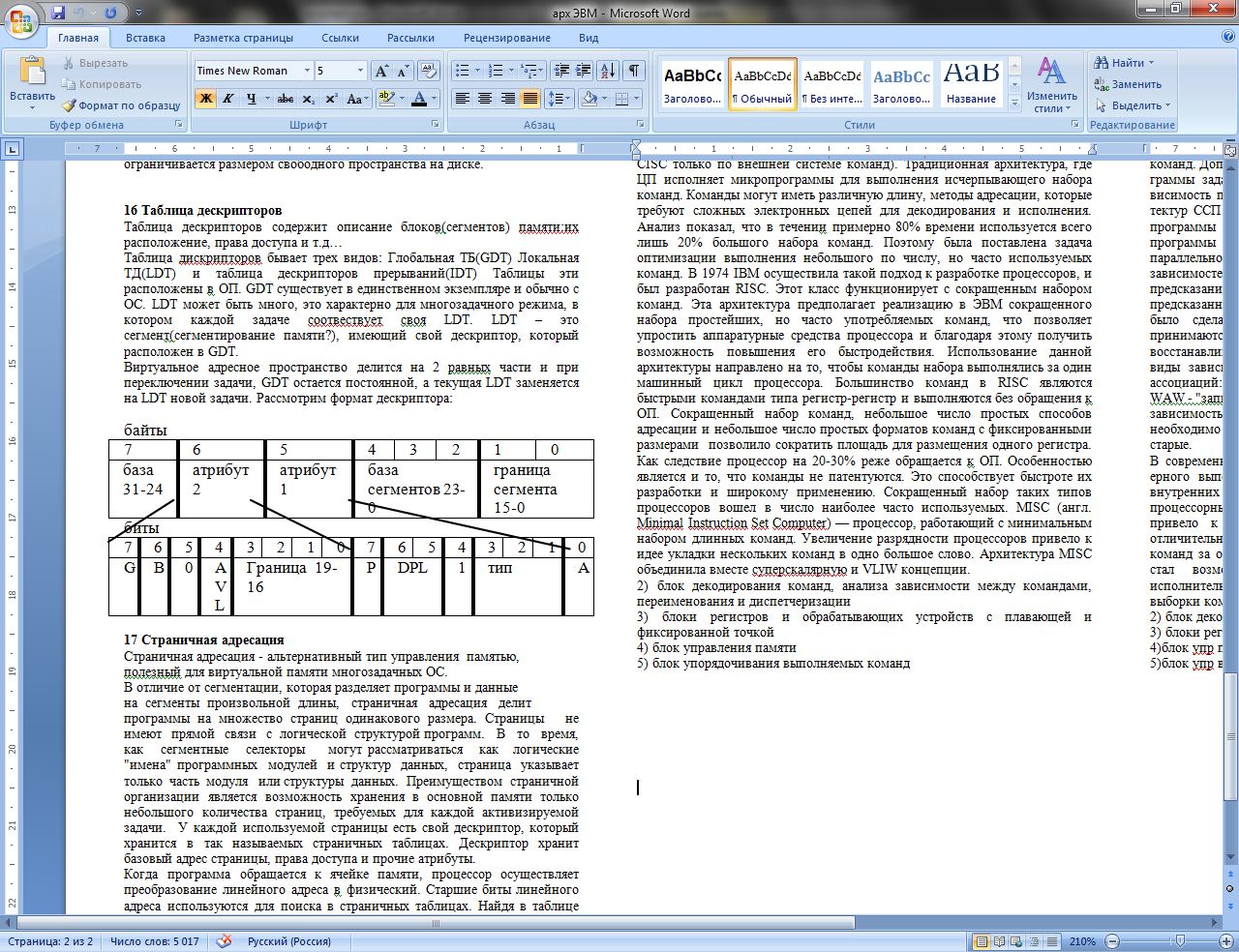
Размер сегмента указ-ся в 16 битах и может принимать любые значения от 64 кб и меньше. В этом состоит фундаментальное отличие от реального режима, где каждый сегмент по умолчанию имеет размер 64 кб. После опред-я базового адреса к нему прибавляется значение смещения, размещаемого в 24 битах и проц-р обращ-ся к соответствующей ячейке памяти.

Очевидно, что данный метод сложнее чем для реального режима, но он остается «прозрачным» для прогр-та, т к мех-м адресации в целом выглядит неизменно.

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

Таблица дискрипторов бывает трех видов: Глобальная ТБ(GDT) Локальная ТД(LDT) и 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 соотвествует своя LDT. LDT – это сегмент(сегментирование памяти?), 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. Рассмотрим формат дескриптора:



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

Одной из наиболее правильной оценок производительности системы является время ответа (время между момента ввода пользователем задания и получения им результата), который учитывает все накладные расходы, связанные с выполнением задания в системе, включая ввод-вывод.

**Система ввода-вывода** – комплекс средств обмена информации с внешними устройствами, который является важнейшей частью архитектуры процессора и ЭВМ в целом. К ней относятся:

* способы подключения к системной шине различного оборудования
* процедуры взаимодействия процессора с этим оборудованием
* команды процессора, предназначенные для обмена данными с внешними устройствами
* непосредственно устройства ввода-вывода.

Логическую схему современного компьютера можно представить в виде системной шины (магистрали) к которой подключен сам компьютер и все устройства компьютера.



*A – адресат, D – данные, M/IQ – один из сигналов управления (3 левые стрелки аналогичны всем остальным тройкам стрелок, ведущим к системной шине).*

Микропроцессор – арифметико-логическое устройство, кэш, управляющее устройство.

Системная шина представляет собой набор линий (проводов), к которым единообразно подключаются все устройства компьютера. В более широком плане понятие системной шины включает в себя электрические и логические характеристики сигналов, их назначение, а также правила взаимодействия этих сигналов при выполнении тех или иных операциях (протоколы обмена информации).

Процессор, желая записать данные по некоторому адресу в памяти, выставляет в линии адресов требуемый адрес, а по линии данных - данные. Устройство управления памятью расшифровывает поступивший адрес, и принимает с линии данных поступившие данные и заносит их в соответствующую ячейку.

*Системные и локальные шины.*

Одним из простейших механизмов, позволяющих организовать взаимодействие простых подсистем, является единственная центральная шина. Имеет 2 основных преимущества: низкая стоимость, универсальность. Основной недостаток: ограничение максимальной пропускной способности.

Причины трудностей:

* физические факторы: длина шины и количество подсоединяемых устройств.
* ограничение фактической пропускной способности.

Традиционно шины делятся на шины, обеспечивающие связь процессора с памятью и шины ввода/вывода. С целью снижения стоимости некоторые компьютеры имеют единственную шину и она называется системной. Необходимость сохранения баланса производительности по мере роста производительности микропроцессоров, привела к двухуровневой организации шины ПК на основе локальной шины.

**Локальной шиной** называется шина, электрически выходящая непосредственно на контакты микропроцессора. Она обычно объединяет процессор, память, схемы буферизации для системной шины, и её контроллер, а также некоторые вспомогательные шины.

В настоящее время используется 2 типа шин, отличающихся способом коммутации: 1) шины с коммутацией цепей 2) шины с коммутацией пакетов. Второй тип шин обеспечивает значительно большую пропускную способность по сравнению с первой за счет разделения транзакций на две логические части: запроса шины и ответа.

Накопитель на магнитных дисках (**НМД**) представляет собой набор пластин, магнитных головок, кареток, линейных двигателей плюс воздухонепроницаемый корпус. Дисковым устройством называется НМД с относящимися к нему электронными схемами.

Принцип магнитной записи электрических сигналов на движущийся магнитный носитель основан на явлении остаточного намагничивания магнитных материалов. Запись и хранение информации на магнитном носителе производится путем преобразования электрических сигналов в соответствующие им изменения магнитного поля, воздействия его на магнитный носитель и сохранения следов этих воздействий в магнитном материале длительное время, благодаря явлению остаточного магнетизма. Воспроизведение электрических сигналов производится путем обратного преобразования  
При цифровой магнитной записи в магнитную головку поступает ток, при котором поле записи через определенные промежутки времени изменяет свое направление на противоположное. В результате под действием поля рассеяния магнитной головки происходят намагничивание или перемагничивание отдельных участков движущегося магнитного носителя.

Другим направлением развития систем хранения информации являются **магнитооптические диски**. Запись на них выполняется при взаимодействии лазера и магнитной головки. Луч лазера разогревает до точки Кюри (температуры потери материалом магнитных свойств) микроскопическую область записывающего слоя, которая при выходе из зоны действия лазера остывает, фиксируя магнитное поле, наведенное магнитной головкой. В результате данные, записанные на диск, не боятся сильных магнитных полей и колебаний температуры. МО-диски уступают обычным жестким магнитным дискам лишь по времени доступа к данным.

Для достижения повышенного уровня отказоустойчивости, приходится жертвовать пропускной способностью ввода\вывода или емкостью памяти. В этой ситуации используются дополнительные диски, которые содержат избыточную информацию, позволяющую остановить исходные данные при отказе диска (например, вместо 25 дорожек используется 20, но наличие дополнительных дорожек позволяет восстановить сигнал на других дорожках).

Существует несколько способов объединения дисков **RAID**.

**RAID 1** – зеркальные диски. В этом случае все диски дублируются. Один из самых дорогостоящих.

**RAID 2** – матрица с поразрядным расслоением. Вводятся избыточные контрольные разряды для исправления одиночных и обнаружения двойных ошибок. Один диск контроля четности позволяет обнаружить одиночную ошибку, но для ее исправления требуется больше дисков.

**RAID 3** – аппаратное обнаружение ошибок и четность. Большинство контроллеров в состоянии определить, когда диск отказал. По существу, если контроллер может определить положение ошибочного разряда, то для восстановления данных требуется лишь один бит четности. Уменьшение числа контрольных дисков до одного на группу снижает избыточность емкости до вполне разумных размеров.

**RAID 4** – внутригрупповой параллелизм. RAID уровня 4 повышает производительность передачи небольших объемов данных за счет параллелизма, давая возможность выполнять более одного обращения по вводу/выводу к группе в единицу времени. В системах уровня 4 для записи небольших массивов данных используются два диска, которые выполняют четыре выборки (чтение данных плюс четности, запись данных плюс четности).

**RAID 5** – четность вращения для распараллеливания записей Система уровня 5 улучшает возможности системы уровня 4 посредством распределения контрольной информации между всеми дисками группы.

**RAID 6** – двумерная четность для обеспечения большой надежности.

1. **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 прер-ий:

- внутренние (инициируются сост-ем центр пр-ра или команды): деление на 0, переполнение

- внешние (иниции-ся сигналом подаваемым от др компонентов системы): запрос на обслуживание со стороны какого либо устр-ва

Переход к проц-ре прер-я осущ-ся из любой пр-мы, а после вып-я процедуры пр-я обязательно происходит процесс возврата в прерванную пр-му. Перед обращением к проц-ре пр-я должно быть сохранено сос-е всех регистров и флагов, используемых проц-й прер-я, а после окончания пр-я эти регистры должны быть восстановлены.

Пр-е вынуждает проц-р прекратить вып-е одной послед-ти команд и начать вып-е другой. При этом адрес очередной команды, кот должна была выполняться запомниться. Адрес команды, кот должна была вып-ся после возникновения пр-я выбирается из т-цы, хранящейся в начальной области памяти. Эта т-ца наз-ся т-ца векторов прер-я. В ней записано 256 адресов.

Когда устр-во вызывает пр-е пр-ра, оно сообщает ему какой адрес из т-цы следует вып-ть для перехода к новой посл-ти 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.

1. **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 с1964 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.

1. **Особенности RISC архитектуры.**

RISC - ReducedInstructionSetComputer – 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 (более 100 по лекциям, а вообще 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 RISCархитектуры 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 суперЭВМCray-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.

1. **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.

1. **Архитектура ЭВМ с длинным командным словом.**
2. **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](computers.html#sp2), Intel PARAGON/ASCI Red, SGI/CRAY [T3E](computers.html#crayt3e), Hitachi [SR8000](/parallel/news/hitachi_sr8000f1.html), транспьютерные системы [Parsytec](vendors.html#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](#cluster) подходу). Пример: IBM RS/6000 SP + ОС AIX, устанавливаемая отдельно на каждом узле. |
| **Модель программирования** | Программирование в рамках модели **передачи сообщений** ( [MPI](/parallel/tech/tech_dev/mpi.html), [PVM](/parallel/tech/tech_dev/ifaces.html#pvm), [BSPlib](/parallel/tech/tech_dev/ifaces.html#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](computers.html#exemplar), 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](#clusters) или [NUMA](#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](/parallel/tech/tech_dev/openmp.html)). Для SMP-систем существуют сравнительно эффективные средства [автоматического распараллеливания](/parallel/tech/tech_dev/auto_par.html). |

**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](computers.html#exemplar) в SCA-конфигурациях, SGI [Origin2000](computers.html#origin2000), Sun [HPC 10000](computers.html#starfire), IBM/Sequent [NUMA-Q 2000](computers.html#numa-q), SNI [RM600](computers.html#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](#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** | Аналогично [SMP](#smp). |

**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](#smp)) в рамках многопроцессорных конфигураций. Несколько таких узлов могут быть объединены с помощью коммутатора (аналогично [MPP](#mpp)). |
| **Примеры** | NEC SX-4/[SX-5](computers.html#sx-5), линия векторно-конвейерных компьютеров CRAY: от CRAY-1, CRAY J90/[T90](computers.html#crayt90), [CRAY SV1](computers.html#sv1), серия Fujitsu [VPP](computers.html#vpp700). |
| **Модель программирования** | Эффективное программирование подразумевает векторизацию циклов (для достижения разумной производительности одного процессора) и их распараллеливание (для одновременной загрузки нескольких процессоров одним приложением). |