Оглавление

1. Многоуровневая компьютерная организация. Современные многоуровневые машины. Организация компьютерных систем.

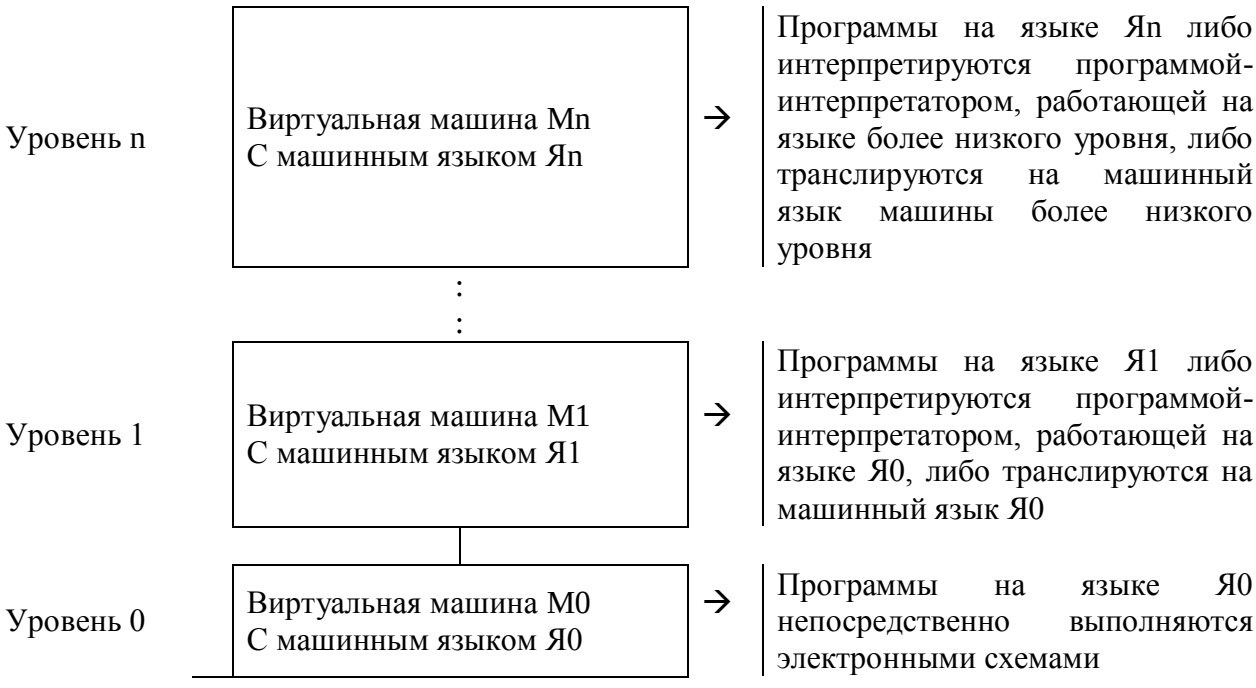
Семантический разрыв — различие принципов, лежащих в основе языков программирования высокого уровня и тех, которые определяют архитектуру ЭВМ.

Многоуровневая компьютерная реализация

Я1 – язык программирования, удобный для человека. Я0 - некоторый язык, понятный компьютеру.

Я1 -> Я0:

1. Трансляция — замена каждой команды, написанной на Я1 эквивалентным набором команд Я0. Можно отбросить программу на Я1.
2. Интерпретация — программа-интерпретатор (на Я0) рассматривает каждую команду Я1 по очереди и сразу выполняет эквивалентный набор команд Я0. Нельзя отбросить программу на Я1.



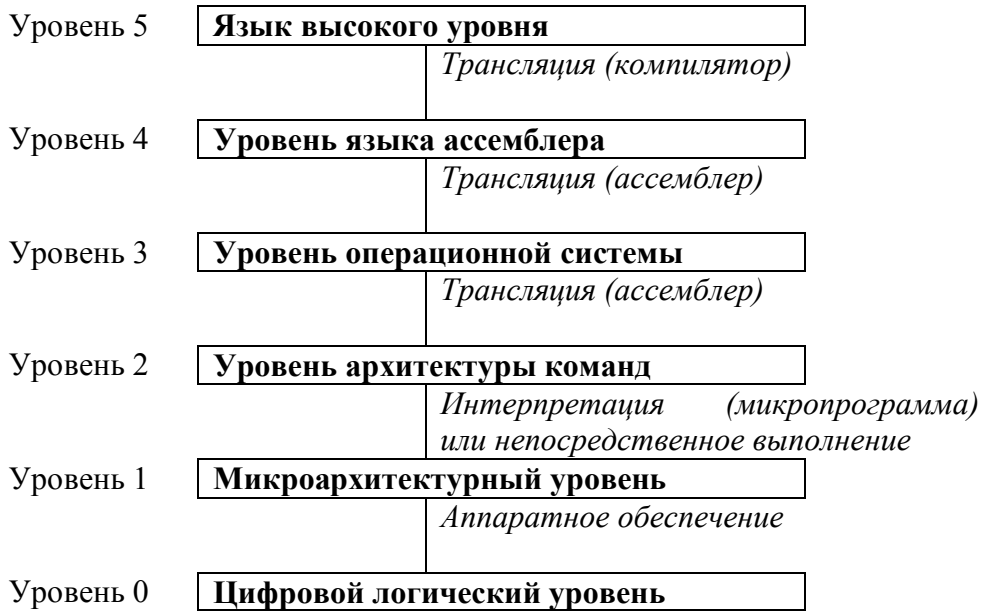
Для удобства считается, что ЯN выполняет виртуальная машина MN.

Целесообразность преобразования ЯN -> ЯN-1 — малое их различие. Для создания удобного человеку языка создаётся серия виртуальных машин, где MN выполняет поставленным требованиям.

Для написания программы необходимо знать лишь ЯN.

Современные многоуровневые машины

Выделяют 6 уровней представления:



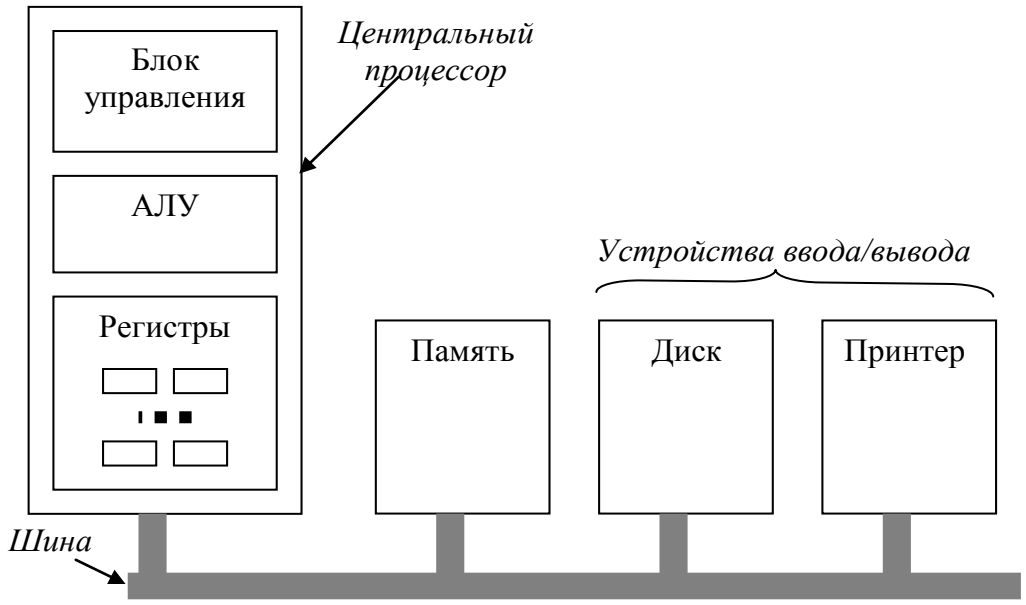
1. Цифровой логический уровень
   Объекты – вентили (бит памяти), описывающиеся с помощью двоичной логики
2. Микроархитектурный уровень (аппаратное обеспечение)
   Совокупности регистров, которые образуют локальную (или регистровую) память и АЛУ. Регистры вместе с АЛУ образуют тракт данных. Основная операция состоит в следующем: выбирается один или два регистра, АЛУ производит над ними некоторую операцию, результат помещается в один из регистров. Контролируется микропрограммой или аппаратно.
3. Уровень архитектуры команд (интерпретация (микропрограмма), либо непосредственное управление)
   Команды, выполняемые программой-интерпретатором.
4. Уровень операционной системы (трансляция (ассемблер))
   Особенности уровня: новые команды, иная организация памяти, способность выполнять две и более программ одновременно. Команды третьего уровня, идентичные командам второго, выполняются микропрограммой или аппаратными средствами, а не операционной системой.
5. Уровень языка ассемблера (трансляция (ассемблер))
6. Язык высокого уровня (трансляция (компилятор))
   Уровень физических устройств отсутствует, т.к. это электронные схемы и не является предметом рассмотрения курса.

Уровни 0-3 предназначены для системного программиста, цифровые (программы состоят из наборов цифр), 2-й и 3-й интерпретируются, 4+ - для прикладного программиста, содержат слова и сокращения, понятные человеку, могут как интерпретироваться, так и транслироваться.

Организация компьютерных систем.

Набор типов данных, операций и особенностей каждого уровня называется архитектурой. Архитектура связана с аспектами, которые видны программисту. Аспекты разработки, технологии и т.д. не являются частью архитектуры. Термины компьютерная архитектура и компьютерная организация в сущности означают одно и то же.

Цифровой компьютер состоит из связанных между собой процессора, памяти и устройств ввода-вывода.

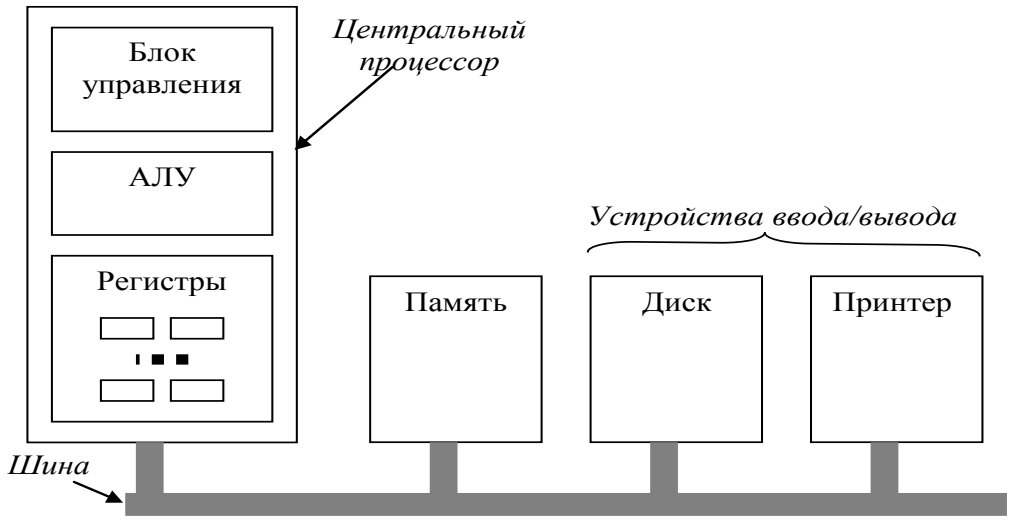


2. Процессоры. Устройство центрального процессора. RISC и CISC. Принципы разработки современных компьютеров.

Цифровой компьютер состоит из связанных между собой процессора, памяти и устройств ввода-вывода.

Процессор

Задача – выполнять программы, находящиеся в основной памяти.



Части:

1. Блок управления – вызов команд из памяти и определение их типа.
2. АЛУ – выполнение арифметических и логических операций.
3. Регистровая память – хранения промежуточных результатов и некоторых команд управления.

Разделяют регистры общего назначения и специальные. Количество регистров и их организация определяется архитектурой компьютера.

Счетчик команд (указатель команд) – адрес следущей команды.

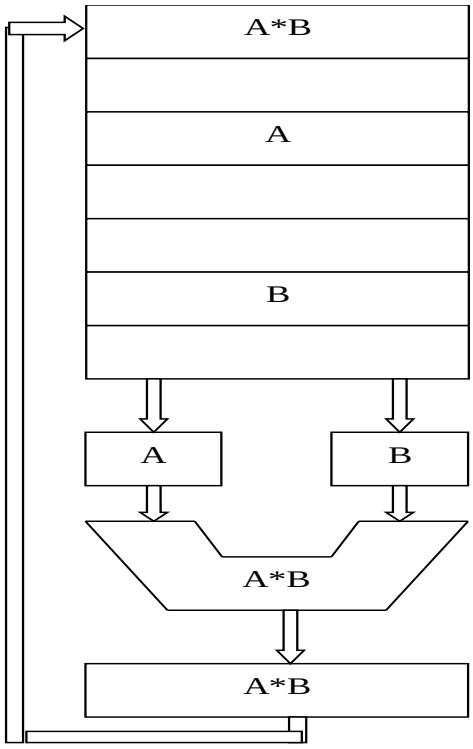
Регистр команд – команда, выполняемая в данный момент.

Устройство центрального процессора

Тракт данных состоит из регистров, АЛУ и нескольких соединяющих шин.

Центральный процессор выполняет каждую команду за несколько шагов (выборка – декодирование – исполнение) :

1. вызывает следующую команду из памяти и переносит ее в регистр команд;
2. меняет положение указателя команд, который теперь указывает на следующую команду;
3. определяет тип вызванной команды;
4. если команда использует слово из памяти, определяет местонахождение этого слова;
5. переносит слово (если это необходимо) в регистр центрального процессора;
6. выполняет команду;
7. переходит к шагу 1, чтобы начать выполнение следующей команды.



Описание работы центрального процессора можно представить в виде программы-интерпретатора.

RISC и CISC

Первые компьютеры -> малый набор простых команд. Усложнение компьютеров -> появление более сложных команд (выполняются быстрее, нагрузка на аппаратное обеспечение). Дальнейшее развитие -> появление сложных команд на дешёвых компьютерах.

50-е гг. - IBM - производство аппаратно-совместимых (архитектурно совместимых) компьютеров выгодно. Для переноса сложных команд на дешёвые компьютеры используется интерпретация.

Преимущества интерпретатора:

1. возможность фиксировать неправильное выполнение команды и восполнять недостатки аппаратного обеспечения;
2. возможность добавлять новые команды при минимальных затратах;
3. структурированная организация.

Со временем интерпретаторы стали применяться практически во всех компьютерах.

1980 год – процессор без использования интерпретатора – RISC (Reduced Instruction Set Computer) – ограниченный набор команд, выбор и декодирование команд осуществляется быстро.

RISC противопоставляется CISC (Complex Instruction Set Computer). Типичный CISC-компьютер – VAX.

Замещения не произошло из-за того, что :

1. RISC-компьютеры были несовместимы по программному обеспечению с другими моделями.
2. идеи RISC воплощались в компьютерах с CISC архитектурой.

Принципы разработки современных компьютеров

Принципы RISC:

1. Все команды непосредственно выполняются аппаратными средствами. Таким исключается этап интерпретации, что обеспечивает высокую скорость выполнения. Однако сложные и/или редко встречающиеся команды могут быть разбиты на несколько команд и выполняться как микропрограмма.
2. Компьютер должен начинать выполнение большого числа команд. Процессор MIPS-500 способен приступить к выполнению 500 млн ком/с. Этот принцип полагает, что параллелизм может играть главную роль в повышении производительности.
3. Команды должны легко декодироваться. Предел количества вызываемых команд в секунду зависит от процесса декодирования. Декодирование необходимо для того, чтобы определить, какие ресурсы необходимы для выполнения команды. Полезны любые средства для упрощения процедуры декодирования.
4. К памяти должны обращаться только команды записи/считывания.
5. Должно быть большое количество регистров.

3. Цифровой логический уровень. Основные цифровые логические схемы.

В самом низу иерархической схемы находится цифровой логический уровень, или аппаратное обеспечение компьютера.

Работа аппаратных средств компьютера основана на цифровых схемах. В цифровых схемах существуют лишь два логических значения. Эти логические значения создаются аналоговыми устройствами. Обычно сигнал напряжением от 0 до 1 В представляет одно значение (например, 0), а сигнал от 2 до 5 В - другое (например, 1). Напряжение за пределами этих величин недопустимо. Мельчайшие электронные устройства – вентили – могут вычислять различные функции от этих двузначных сигналов. Вентили представляют собой основу аппаратного обеспечения цифровых компьютеров. Отметим лишь, что вентили строятся с помощью транзисторов.

Основные цифровые логические схемы

- Комбинационные схемы – цифровые устройства без памяти, т.е. комбинация входных сигналов определяет состояние выхода схемы. Комбинационные элементы:
    1. Мультиплексоры – устройство с 2^n входами данных, одним выходом и n управляющими входами, позволяющими выбрать один из входов данных;
    2. Декодер – из позиционного кода формирует унитарный;
    3. Компараторы – производят сравнение двух слов;
    4. Программируемые логические матрицы;
- Арифметические схемы (используются для выполнения арифметических операций):
    1. Схемы сдвига;
    2. Сумматоры;
    3. Арифметико-логические устройства;
    4. Тактовые генераторы;
- Память
    1. Защелки и триггера—это один бит памяти;
    2. Регистры;
    3. Микросхемы памяти.
    4. ОЗУ и ПЗУ.

4. Микросхемы процессоров. Шины. Примеры центральных процессоров

Микросхемы процессоров

Каждая микросхема процессора содержит набор выводов, через которые происходит обмен информацией с внешним миром. Выводы микросхем можно подразделить на три типа: адресные, информационные, управляющие.

Число адресных и информационных выводов – два ключевых параметра, определяющих производительность процессора. Обычно процессор имеет 16, 20, 32 или 64 – разрядный адрес и 8, 16, 32, 36 или 64 – разрядный информационный код.

Также имеются выводы управления, питания, земли и синхронизирующего сигнала (меандр).

Состав управляющих сигналов у различных процессоров может различаться, но их можно разделить на следующие основные категории:

1. Управление шиной;
2. Прерывание;
3. Арбитраж шины;
4. Состояние;
5. Разное.

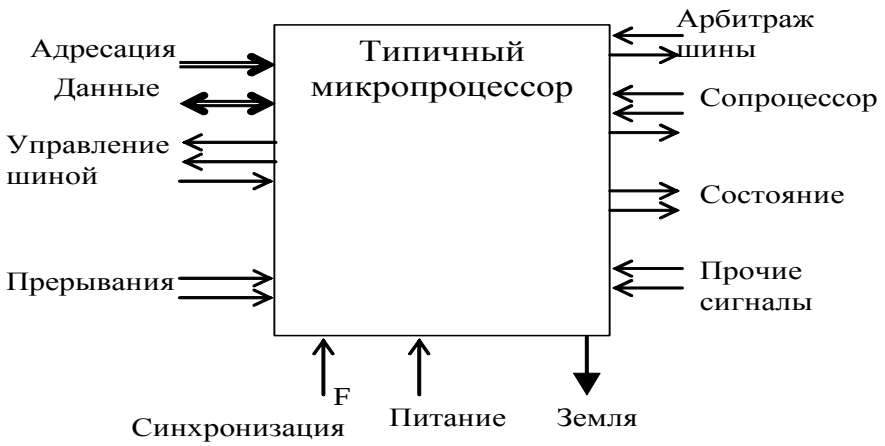


Рис. 2.1

Шины

Шина – это группа проводников, соединяющих различные устройства.

Первые персональные компьютеры имели одну внешнюю шину, которая называлась *системной.* Современные компьютеры содержат обычно содержат специальную шину между центральным процессором и памятью и по крайней мере еще одну шину для устройств ввода вывода. Существуют четкие правила о том, как работает шина, и все устройства, связанные с шиной подчиняются этим правилам. Эти правила называются *протоколом шины.* Существуют ряд широко используемых шин.

Устройство, подключённое к шине:

1. Активное (задающее) – может инициировать обращение к шине
2. Пассивное (подчинённое) – ждёт запроса от шины

Устройство может быть активным и подчинённым одновременно. Память – всегда подчиненное устройство.

Параметры шины:

- Ширина шины
  Чем больше ширина шины, тем она лучше и тем она дороже. Пропускную способность шины можно увеличить двумя способами: увеличением ширины шины и сокращением времени цикла. Увеличение скорости работы шины порождает проблему совместимости ее со старыми устройствами.
- Синхронизация шины
  Типы шин:

1. Синхронная – содержит линию, которая запускается кварцевым генератором и любое действие шины занимает целое число циклов шины.
2. Асинхронная – не содержит кварцевого генератора и циклы шины могут быть любой длины.

- Арбитраж шины:
  1. Централизованный – арбитр производит последовательный опрос устройств. Если устройство начинает передачу, опрос прекращается. Приоритет определяется физической близостью устройства к арбитру. Для создания нескольких уровней приоритета производится увеличение опросных линий.
  2. Децентрализованный – проверка устройством состояния. Свободна – монополизирует; занята – сбрасывает сигнал запроса и спустя некоторое время повторяет попытку воспользоваться шиной.

Примеры центральных процессоров

Pentium 4

Программно – 32bit, аппаратно – схож с 64b, т.к. может передавать данные блоками по 64 бита (для программиста процесс не виден). Реализация новой микроархитектуры NetBurst (два АЛУ, каждое работает в два раза быстрее тактовой частоты). Поддерживает гиперпоточность (наличие двух наборов регистров и других внутренних ресурсов, быстрое переключение программ). Работает как суперскалярная машина.

Двух-/трёх- (в зависимости от модели) уровневая кэш-память. L1 – SRAM 8 Кбайт, хранит декодированные команды для выполнения на RISC-ядре (до 12 000 декодированных микроопераций). L2 – [256 – 1024] Кбайт, хранение не декодированных данных. Pentium 4 Extreme Edition - L2 = 2 Мбайт. Механизм мультипроцессорной синхронизации для устранения несогласованности кэш-памяти.

Две внешние шины, обе синхронные. Шина памяти – доступ к ОЗУ; шина PCI – общение с устройствами ввода-вывода.

Расположен в квадратном корпусе со стороной 35 мм, имеет 478 выводов. (85-питание, 180 – «Земля»). Одна из проблем – отвод тепла, особенно актуальна для ноутбуков. 5 состояний от полной активности до глубокого сна.

Поскольку процессор работает быстрее, чем ОЗУ, для повышения производительности системы шина памяти процессора работает в конвейерном режиме, при этом в шине одновременно 8 операций.

Обращение процессора к памяти называются транзакциями и имеют 6 стадий:

1. Фаза арбитража шины - определяется, какое из задающих устройств будет следующим.
2. Фаза запроса - на шину передается адрес.
3. Фаза сообщения об ошибке - подчиненное устройство передает сигнал об ошибке четности в адресе или каких-либо других неполадках.
4. Слежение - фаза проверки на наличие нужного слова на другом процессоре – проверяется, нет ли конфликта с другим процессором.
5. Фаза ответа - задающее устройство узнает, где взять необходимые данные.
6. Фаза передачи данных - собственно передача данных.

Наличие всех фаз необязательно.

UltraSPARC III

UltraSPARC – это серия 64-разрядных процессоров SPARC с архитектурой RISC. В каждую микросхему UltraSPARC III включены связующие элементы, необходимые для построения мультипроцессорных систем до 4 процессоров.

Микросхема содержит 29 млн. транзисторов, 1368 выводов. Тактовая частота 600 МГц (2000 г.), 900 МГц (2001), 1,2 ГГц (2002).

Два кэша L1: 32 Кбайт для команд и 64 Кбайт для данных + блок предвыборки 2 Кбайт и кэш записи на 2 Кбайт. Кэш-память L2 не упакована в один картридж с процессором и может быть заменена.

В процессоре UltraSPARC III соединение с памятью используется механизм UPA (Ultra Port Architecture – высокоскоростной пакетный коммутатор). Реализация интерфейса может быть в виде шины, коммутатора.

8051

Микроконтроллер поставляется в стандартном корпусе с 40 выводами. Имеется 16 адресных вывода (адресуемое пространство 64 Кбайт). Ширина данных 1 байт. Отличительная особенность – 32 линии ввода/вывода в четырех группах (по 8 в каждой). Объем внутренней памяти составляет 4 Кбайт (8 для 8052). Если памяти недостаточно, то можно добавить дополнительный модуль до 64 Кбайт.

5. Микроархитектурный уровень. Тракт данных

Микроархитектурный уровень

Задача – интерпретация уровня команд в управляющие сигналы для цифровых устройств. Строение зависит от уровня архитектуры команд, от стоимости и назначения компьютера.

В настоящее время уровень архитектуры команд часто содержит простые команды, которые выполняются за один цикл (системы RISC). В других системах (например Pentium II) на этом уровне имеются более сложные команды, выполняемые за несколько циклов.

Для того, чтобы выполнить команду небходимо

- Найти операнды в памяти;
- Считать операнды;
- Выполнить операцию;
- Записать полученный результат в память.

Микроархитектура содержит микропрограмму, располагаемую в ПЗУ. Микропрограмма должна вызывать, декодировать и выполнять команды. Поскольку аппаратное обеспечение состоит из вентилей и регистров, то микропрограммы должна выдавать команды, управляющие этими вентилями и регистрами.

Микропрограмма содержит набор переменных, к которым имеют доступ все функции. Эти переменные называются состоянием компьютера. Каждая функция изменяет некоторые переменные, формируя новое состояние компьютера.

Каждая команда состоит из нескольких полей, каждое из которых выполняет определенную функцию. Первое поле называется кодом операции. Это поле определяет действие, которое необходимо выполнить. Другие поля указывают на операнды, типы и т.д. Чем сложнее команда, тем больше полей требуется для ее описания.

Тракт данных

Часть центрального процессора, состоящая из АЛУ, его входов и выходов. Содержит ряд регистров и АЛУ. Регистры и АЛУ связаны шинами. Под действием управляющих сигналов содержимое регистров передается на вход АЛУ, а результат записывается в требуемый регистр.

Функционирование АЛУ зависит от управляющих сигналов, имеет два входа. Схема сдвига имеет свое управление.

За один цикл АЛУ можно считать и записать один и тот же регистр. Процессы считывания и записи происходят в разных частях цикла.

Содержимое одного из регистров выдается на шину С в начале цикла и сохраняются там на протяжении всего цикла. Затем АЛУ выполняет операцию, результат которой через схему сдвига поступает на шину С. После стабилизации сигналов на шине С ее содержимое передается в один или несколько регистров. Одним из нескольких регистров может быть тот, от которого поступил сигнал на шину В. Для реализации этих действий должна быть выполнена синхронизация тракта данных.

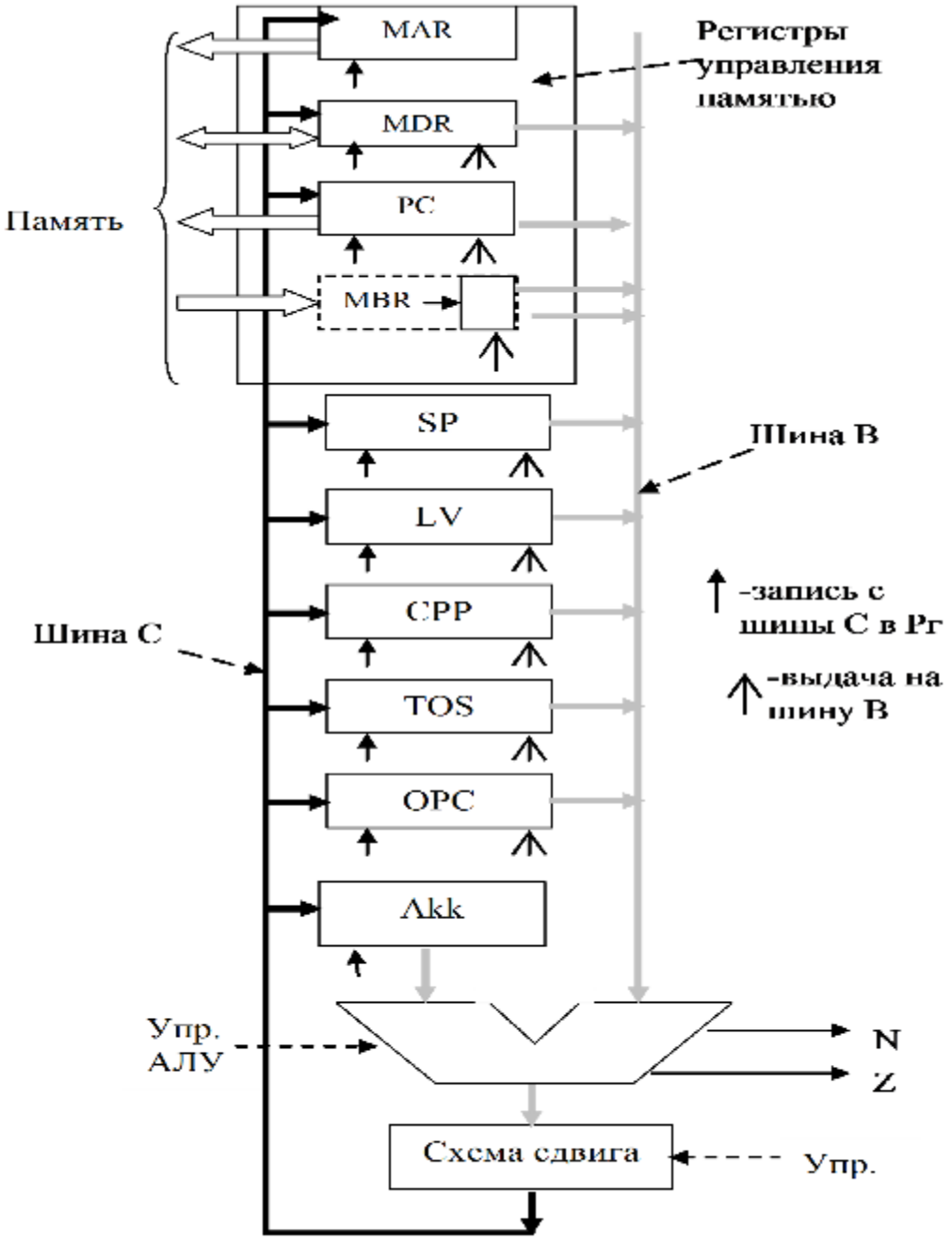


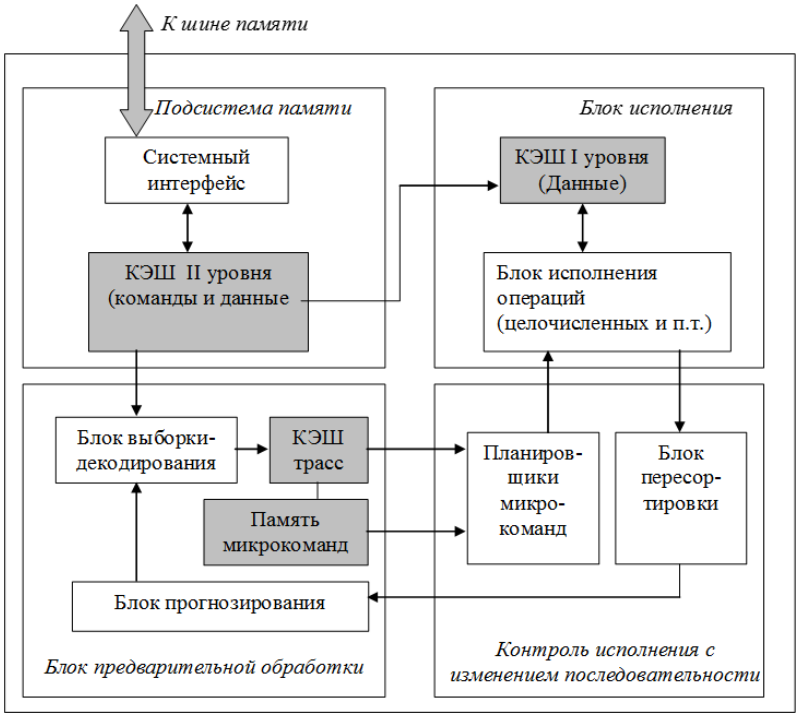
Рис. 3.1

6. Примеры реализации микроархитектур процессоров

0. Pentium 4

Поддерживает 32-битные операнды и арифметику, 64-битные операции с плавающей точкой, а также 8-ми и 16-ти битные операции, унаследованные от предыдущих моделей. Процессор может адресовать до 64 Гбайт памяти и считывать слова по 64 бита за раз.

NetBurst:



Подсистема памяти: кэш L2, логика доступа к ОЗУ. Кэш от 256 Кбайт до 1 Мбайт. Длина строки 128 байт.

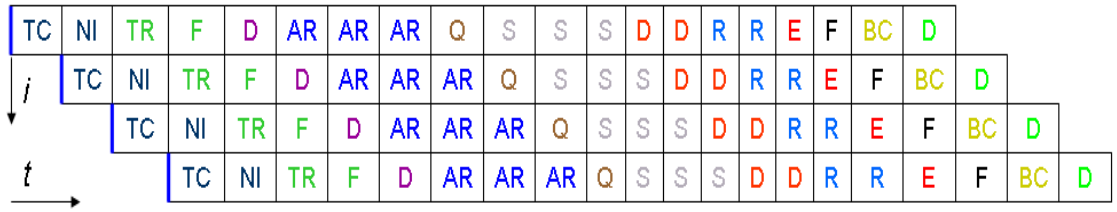
Блок предварительной обработки выбирает команды из L2 и декодирует их в порядке выполнения команд в программе. Каждая команда разбивается на последовательность RISC-операций.

Декодированные операции -> кэш трас - кэш команд первого уровня. Кэшируются декодированные команды - исключается повторное декодирование. +прогнозирование ветвлений.

Контроль исполнения с изменением последовательности - изменение последовательности команд для эффективного выполнения команд.

Блок исполнения - блоки исполнения (параллельные). Данные получают из кэша L1.

Конвейер NetBurst

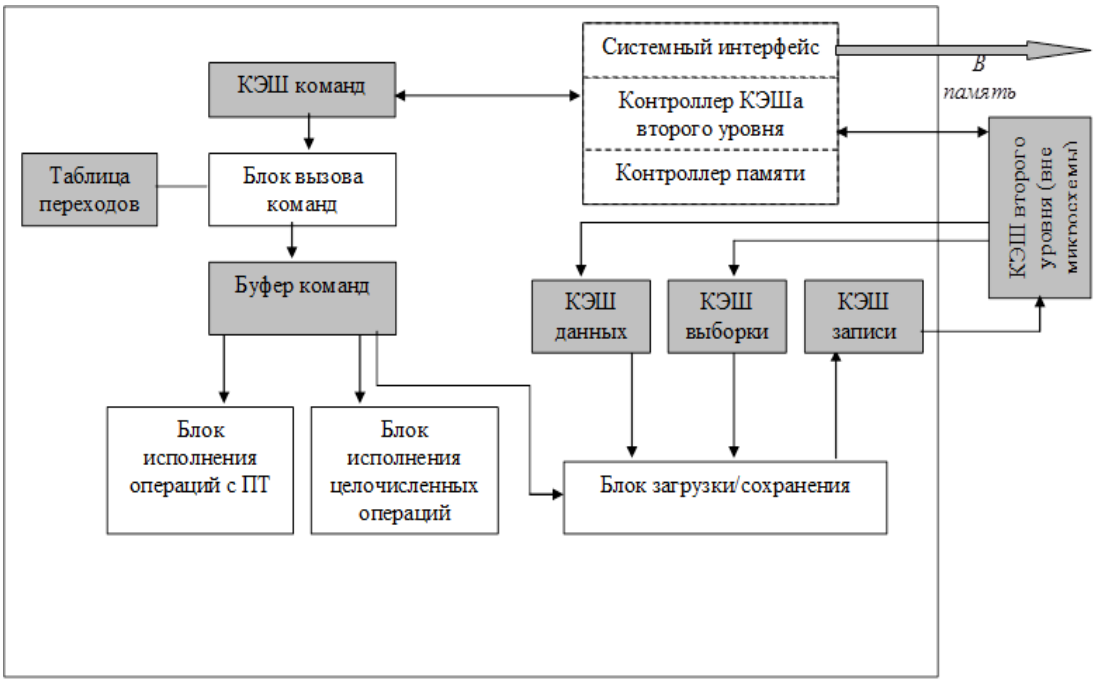


Конвейер состоит из 20 стадий:

1. TC, NI (1, 2) — поиск микроопераций, на которые указывает последняя выполненная инструкция.
2. TR, F (3, 4) — выборка микроопераций.
3. D (5) — перемещение микроопераций.
4. AR (6—8) — резервирование ресурсов процессора, переименование регистров.
5. Q (9) — постановка микроопераций в очереди.
6. S (10—12) — изменение порядка исполнения.
7. D (13—14) — подготовка к исполнению, выборка операндов.
8. R (15—16) — чтение операндов из регистрового файла.
9. E (17) — исполнение.
10. F (18) — вычисление флагов.
11. BC, D (19, 20) — проверка корректности результата.

7. UltraSPARC III

64-разрядная машина (по данным и адресам), но в целях совмещения с предыдущими версиями, она воспринимает 32- разрядные операнды, адреса и программы.



Кэш команд имеет объем в 32 Кбайт (8 000 команд), длина строки – 32 байт.

Блок вызова команд - до 4 команд за цикл. Промах -> количество вызываемых команд уменьшается. Условный переход -> обращение к таблице переходов (16 000 записей). Подготовленные команды -> 16-командный буфер.

Блок исполнения целочисленных операций состоит из двух АЛУ и короткого конвейера для обработки команд перехода. Кроме того здесь имеются регистры.

Блок исполнения операций с ПТ -> 32 регистра, 3 АЛУ, графические операции.

Блок загрузки/сохранения -> команды записи и считывания.

Доступ к памяти:

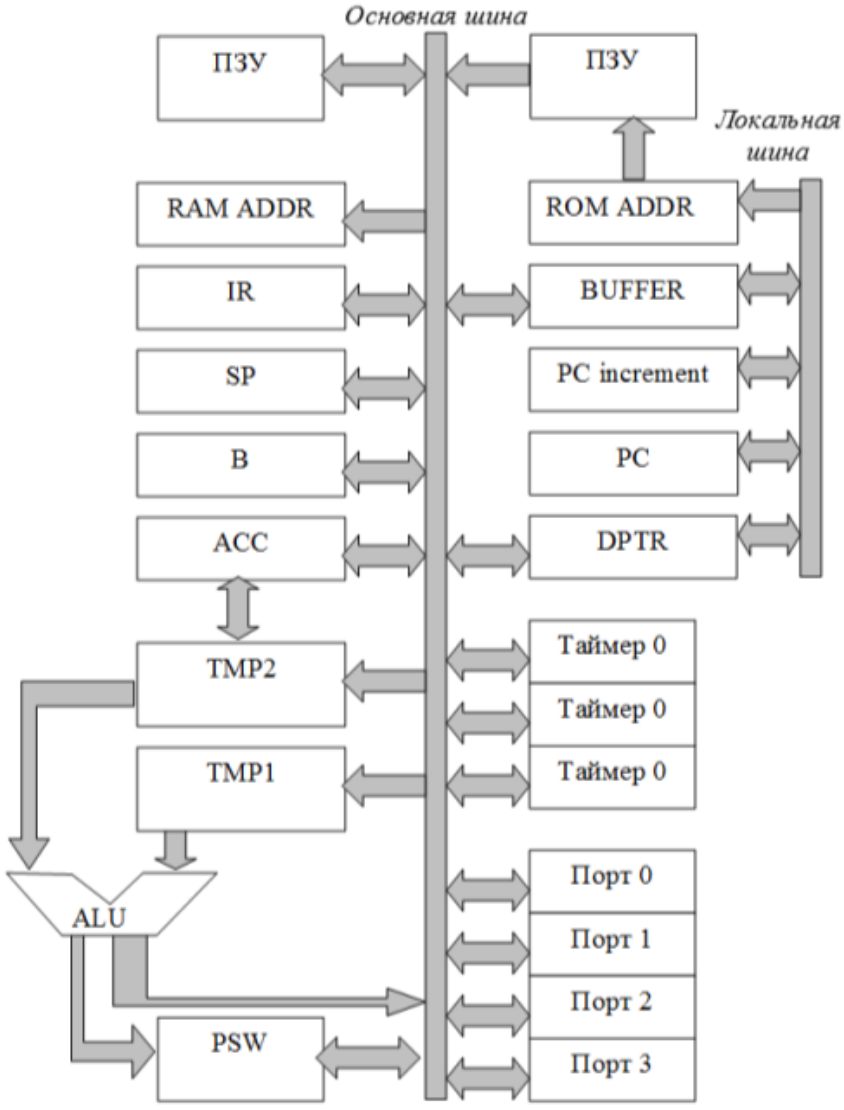
1. Системный интерфейс обеспечивает взаимодействие с памятью по 128-разрядной шине
2. Контроллер кэша второго уровня сопряжен с объединенным кэшем второго уровня, расположенный вне микросхемы
3. Контроллер памяти преобразует 64-разрядные виртуальные адреса в 43-разрядные физические

Конвейер UltraSPARC III (14 стадий):

1. Стадия A (генерация адреса). На этом этапе определяется адрес следующей команды.
2. Стадия P (предварительная выборка) вызывает команды из КЭШа команд первого уровня.
3. Стадия F (выборка) завершается выборка команд и их передача в КЭШ команд.
4. Стадия B (обнаружение объекта перехода) происходит декодирование выбранных команд.
5. Стадия I (группировка команд) – команды сортируются по группам, в зависимости от того, к каким функциональным блокам они обращаются.
6. Стадия J (извлечение команды из очереди) – команды подготавливаются к отправке в блок выполнения во время следующего цикла.
7. Стадия R (регистр) – производится поиск регистров, необходимых для обработки команд целочисленных операций.
8. Стадия E (выполнение) предназначена для выполнения целочисленных команд.
9. Стадия C (кэш) – завершается доступ к кэш-памяти первого уровня.

10. Стадия M (промах) – производится обработка слов, запрошенных, но не найденных в кэш-памяти первого уровня.
11. Стадия W (запись) – результаты записываются в регистровый файл рабочего регистра.
12. Стадия X (продленное выполнение) – завершается большинство команд с ПТ и графических команд.
13. Стадия T (перехват) – перехватываются исключения, связанные с целочисленными командами и командами с ПТ.
14. Стадия D – состояние целочисленных регистров и регистров с ПТ фиксируется в соответствующих архитектурных регистровых файлах.

3. 8051



Независимые модули памяти для данных и кода. Емкость ОЗУ для данных составляет 128 (8051) или 256 (8052) байт.

Регистры:

1. ACC - основной арифметический регистр.
2. B - операции умножения и деления, временный регистр.

3. SP - указатель стека.
4. IR - команды, выполняемые в данный момент.
5. TMP1 и TMP2 – защелки, в которые помещаются операнды.
6. RAM ADDR - адресует память.
7. DPRT (двойной указатель) – временный регистр, предназначенный для управления и сборки 16-разрядных адресов.
8. PC - счетчик команд.
9. Регистр PC incrementer – специальный аппаратный модуль, выполняющий роль псевдорегистра.
10. BUFFER – временный регистр.

Ни к PC, ни к PC incrementer нельзя обратиться через основную шину.

Результаты работы АЛУ - любой регистр через основную шину.

Коды состояний - PSW (слово состояния программы).

Таймеры 16-разрядные, позволяют управлять выполнением приложений в масштабе реального времени.

Предусмотрены четыре 8-разрядных порта, которые могут управлять 32 внешними кнопками, индикаторами, датчиками и т.п.

Процессор относится к синхронным процессорам, большинство операций завершается за один цикл.

Цикл:

1. Следующая команда вызывается из ПЗУ и отправляется в регистр IR.
2. Проводится декодирование команды, значение PC увеличивается на единицу.
3. Подготавливаются операнды.
4. Один из операндов по основной шине передается и размещается в TMP1. + Копирование содержимого ACC в регистр TMP2, после чего два операнда готовы к выполнению.
5. Выполнение команды.
6. Передача результата операции в регистры.

7. Общий обзор уровня архитектуры команд. Свойства уровня команд. Модели памяти.

Уровень архитектуры команд имеет особое значение: он является связующим звеном между аппаратным и программным обеспечением.

Свойства уровня команд

Уровень команд – это то, каким представляется компьютер программисту машинного языка. Программа уровня архитектуры команд – это то, что выдает компилятор.

Для одних архитектур уровень команд определяется формальным документом, который выпускается промышленным консорциумом, для других – нет. Цель такого официального документа – дать другим производителям выпускать машины данного типа, чтобы эти машины могли выполнять одни и те же программы и получать одни и те же результаты. В документе говорится, какая модель памяти используется, какие имеются регистры, какие действия выполняют команды, но не описывается микроархитектура.

Два режима уровня команд:

- В привилегированном режиме запускается операционная система. Этот режим позволяет выполнять все команды.
- Пользовательский режим предназначен для запуска прикладных программ. Он не позволяет запускать некоторые потенциально опасные программы.

Модели памяти

Во всех компьютерах память разделена на ячейки, которые имеют последовательные адреса. В настоящее время наиболее распространены ячейки в 8 байт (ASCII-символ, 7 разрядов и бит четности). Байты обычно группируются в слова по 4 или 8 байт. Многие архитектуры требуют, чтобы слова были выровнены.

Большинство машин имеет единое адресное пространство. В некоторых машинах содержатся отдельные адресные пространства для команд и данных. Такая система гораздо сложнее, чем единое адресное пространство, но зато она имеет два преимущества:

1. Появляется возможность иметь 2^{32} байтов для программы и 2^{32} байтов для данных при размере регистра адреса в 32 разряда;
2. Исключается сама возможность интерпретировать код как данные и наоборот.

Один из аспектов модели памяти – семантика памяти. Семантика памяти – поведение при записи и чтении значений из памяти. Естественно ожидать, что если в программе идет запись по некоторому адресу, а затем считывание из этого же адреса, то мы получим только что сохраненное значение. Но в некоторых машинах микрокоманды переупорядочиваются. Возникает опасность, что память не будет действовать так, как ожидалось.

В некоторых случаях запросы в память делают упорядоченными, в других – возлагают на разработчиков компиляторов заботу о правильном функционировании памяти.

8. Общий обзор уровня команд машины Pentium 4.

3 операционных режима, в 2 работает как 8086:

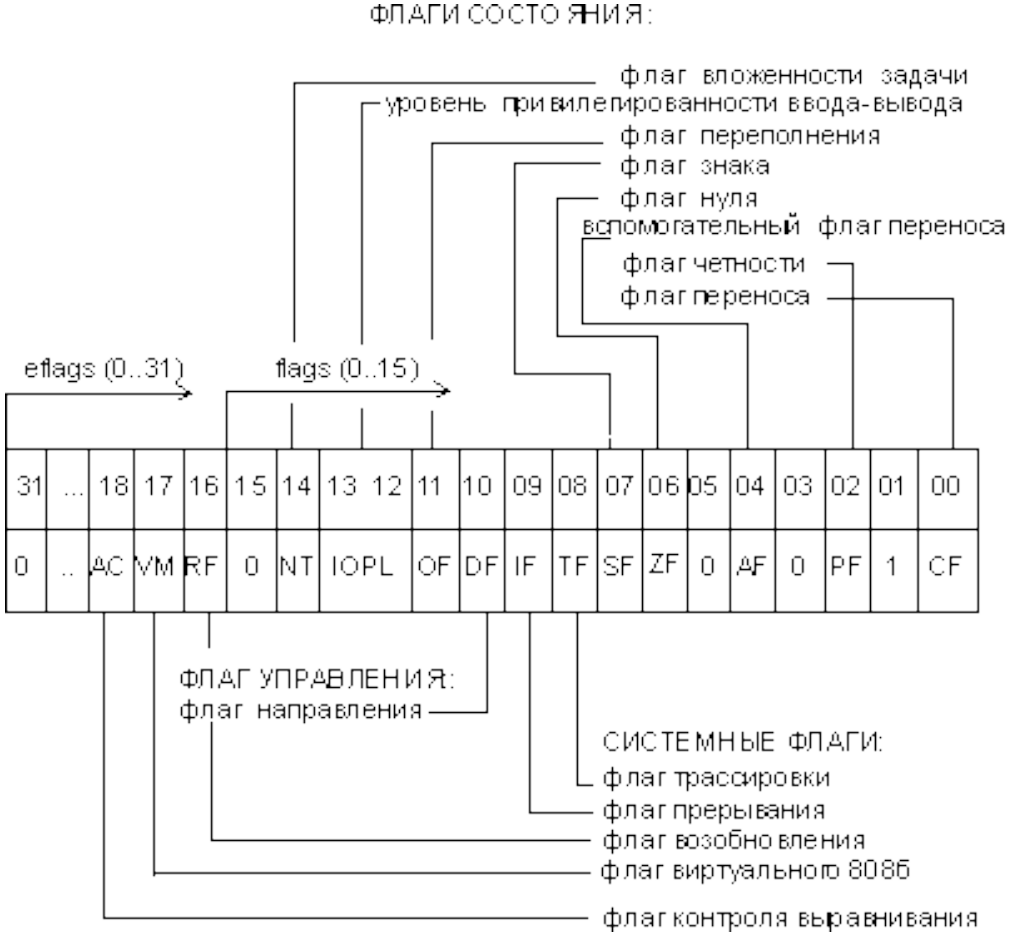
- Real Address Mode – режим реальной адресации, полностью совместим с 8086. До 1 Мб физической памяти.
- Virtual 8086 Mode – режим виртуального процессора 8086. Процессор функционирует как 8086, но могут параллельно выполняться несколько задач с изолированными друг от друга ресурсами. Попытки выполнения недопустимых команд, выхода за пределы отведенного пространства памяти контролируются системой защиты.
- Protected Virtual Address Mode – защищенный режим виртуальной адресации. До 4 Гб физической памяти. Доступны 4 уровня привилегий.

В Pentium имеется 31 регистр; они подразделяются на 16 регистров прикладного программиста (пользовательские регистры) и 15 регистров системного программиста (системные регистры). Пользовательские регистры:

1. Регистры общего назначения:
    1. eax/ax/ah/al - аккумулятор. Применяется для хранения промежуточных данных. В некоторых командах использование этого регистра обязательно;
    2. ebx/bx/bh/bl - базовый регистр. Часто применяется для хранения базового адреса некоторого объекта в памяти;
    3. edx/dx/dh/dl - регистр данных. Хранит промежуточные данные. В некоторых командах его использование обязательно; для некоторых команд это происходит неявно.
    4. ecx/cx/ch/cl - регистр-счетчик. Применяется в командах, производящих некоторые повторяющиеся действия.
    5. esp/sp - регистр указателя стека. Содержит указатель вершины стека в текущем сегменте стека.
    6. ebp/bp - регистр указателя базы кадра стека. Предназначен для организации произвольного доступа к данным внутри стека. Часто хранит адрес начала локальных переменных текущей подпрограммы.

7. esi/si - индекс источника;
8. edi/di - индекс приемника (получателя).

2. Сегментные регистры
   1. cs - сегмент кода. Содержит команды программы.
   2. ds - сегмент данных. Содержит обрабатываемые программой данные.
   3. ss - сегмент стека. Этот сегмент представляет собой область памяти, называемую стеком.
   4. es, gs, fs - дополнительные сегменты данных. Используются, если программе не хватает ds.
3. Регистры состояния и управления
   1. Регистр флагов eflags/flags;
   2. Регистр указателя команды eip/ip.



Системные регистры:

3. Управления - CRO, CR1, CR2, CR3 хранят признаки состояния процессора, общие для всех задач.
4. Системные адресные регистры предназначены для ссылок на сегменты и таблицы в защищенном режиме

- GDTR (Global Descriptor Table Register). Указывает на глобальную таблицу дескрипторов.
- IDTR (Interrupt Descriptor Table Register). Указывает на таблицу дескрипторов прерываний.
- TR (Task Register). Содержит селектор дескриптора сегмента состояния задачи.
- LDTR (Local Descriptor Table register). Содержит селектор локальной таблицы дескрипторов.

5. Регистры отладки предназначены для задания и управления отладочными точками останова.
   1. DRO...DR3 (Linear Breakpoint Address 0...3) хранят 32-битные линейные адреса точек останова;
   2. DR4, DR5 в 386-м и 486-м не используются, обращение к ним эквивалентно обращению к регистрам DR6, DR7.
   3. DR6 (Breakpoint Status) отражает состояние контрольной точки;
   4. DR7 (Breakpoint Control) управляет установкой контрольных точек.

6. Тестовые регистры

Состав:

1. TR3 - регистр данных внутреннего кэша,
2. TR4 — тестовый регистр состояния кэша,
3. TR5 — управляющий регистр тестирования кэша,
4. TR6 (Test Control) — управляющий регистр для теста кэширования страниц,
5. TR? (Test Status) — регистр данных для теста кэширования страниц.

Регистры сопроцессора

Сопроцессор (FPU) предназначен для выполнения операций над вещественными числами. С программной точки зрения сопроцессор содержит блок регистров данных, регистр управления и группу регистров состояния и указателей. Восемь регистров данных разрядностью 80 бит организованы в стек. Номер регистра, являющегося текущей вершиной стека, хранится в специальном поле регистра состояния (указателе вершины стека). Операция push уменьшает значение указателя на 1 и помещает в стек данные в регистр, являющийся новой вершиной стека. Операция pop записывает данные с вершины стека в память или регистр и увеличивает указатель на 1. Инструкции адресуют регистры либо явно, либо неявно. Неявная адресация подразумевает операнд, находящийся на

вершине стека. Явная адресация подразумевает указание смещения регистра относительно вершины стека - st(i).

Устройство FPU определяет численные данные в 7-ми внешних форматах, образуя при этом три класса:

7. двоичные числа – целые числа (16, 32, 64 разряда);
8. упакованные десятичные целые числа;
9. двоичные вещественные числа – 32, 64, 80 разрядов.

Регистры данных FPU (арифметический стек)

Физические номера	80 бит: 1 бит Знак	15 бит Порядок	64 бит Мантисса	Относительные номера
0			mm0	ST(5)
1			mm1	ST(6)
2			mm2	ST(7)
3			mm3	ST(0)
4			mm4	ST(1)
5			mm5	ST(2)
6			mm6	ST(3)
7			mm7	ST(4)

Регистры MMX / 3DNow!

Расширение MMX

MMX было первым расширением, реализующим технологию SIMD (Single Instruction - Multiple Data). Основная идея SIMD заключается в одновременной обработке нескольких элементов данных одной операцией. Расширение MMX использует новые типы упакованных 64-битные целочисленных данных:

10. 8 упакованных байт (Packed byte);
11. 4 упакованных слова (Packed word);
12. 2 упакованных двойных слова (Packed double word);
13. 1 учетверенное слово (Quad word);

Эти типы данных могут специальным образом обрабатываться в 64-битных регистрах MM0-MM7, представляющих собой младшие биты стека 80-битных регистров FPU. Каждая инструкция MMX выполняет действие сразу над всем комплектом операндов (8, 4, 2 или 1), размещенных в адресуемых регистрах. Как и регистры FPU, эти регистры не могут использоваться для адресации памяти. Совпадение регистров MMX и FPU накладывает ограничение на чередование кодов FPU и MMX. В отличие от стека FPU регистры MMX адресуются не с помощью стека, а физически (по своим физическим номерам).

Расширение 3DNow!

Технология 3DNow!, разработанная AMD, расширяет возможности MMX. Она позволяет оперировать с новым типом данных - парой упакованных вещественных чисел одинарной точности. Эти числа занимают по двойному слову (32 бита) в 64-битных регистрах MMX.

Расширение 3DNow! работает с упакованными данными в FP-формате с одинарной точностью, а также упакованными 8 байт, 4 слова, 2 двойных слова и 64-битными целыми числами, размещая их в младших 64 битах регистров FPU/MMX.

Блок XMM

Начиная с Pentium III, Intel использует в своих процессорах новое потоковое расширение SSE (Streaming SIMD Extension). Оно реализуется дополнительным независимым блоком, имеющим восемь 128-битных регистров, названных XMM0-XMM7, и регистр состояния/управления MXCSR. В каждый из регистров XMM помещаются четыре числа в формате с плавающей точкой одинарной точности. Блок позволяет выполнять векторные (пакетные) и скалярные инструкции. Векторные инструкции реализуют операции сразу над четырьмя комплектами операндов. Скалярные инструкции работают только с одним комплектом операндов - младшим 32-битным словом. При выполнении инструкций XMM традиционное оборудование FPU/MMX не используется, что позволяет эффективно смешивать инструкцииMMX с инструкциями с плавающей точкой.

Кроме инструкций с новым блоком XMM в расширение SSE входят и дополнительные целочисленные инструкции с регистрами MMX, а также инструкции управления кэшированием.

В процессоре Pentium4 набор инструкций получил новое расширение - SSE2, в основном касающееся добавления новых типов 128-битных типов данных для блока XMM:

14. Упакованная пара вещественных чисел двойной точности;
15. Упакованные целые числа: 16 байт, 8 слов, 4 двойных слова или пара учетверенных слов.

В процессор введены новые функции целочисленной арифметики, 128-разрядные для регистров XMM и такие же 64-разрядные для регистров MMX; ряд старых инструкций MMX распространили на XMM (в 128-битном варианте); добавлены инструкции преобразования для новых форматов данных, а также расширены возможности "перемешивания" данных в блоке XMM. Кроме того, расширена поддержка управления кэшированием и порядком исполнения операций с памятью.

9. Общий обзор архитектуры уровня команд системы UltraSPARC III.

Архитектура SPARC была впервые введена в 1987 г. и была первой архитектурой промышленного назначения типа RISC. Изначально она была 32-разрядной, но UltraSPARC III – это 64-разрядная машина.

Структура памяти проста: память представляет собой линейный массив из 2^{64} байтов. Память настолько велика, что в настоящее время она не реализуема. Современные реализации имеют размер адресного пространства 2^{44} (UltraSPARC III), но в будущем память будет увеличиваться.

Одна из проблем разработки архитектуры заключается в том, что архитектура команд ограничивает размер адресуемой памяти.

Архитектура команд SPARC достаточно проста, хотя организация регистров усложнена для повышения эффективности вызова процедур.

В системе UltraSPARC имеется две группы регистров: 32 64-битных регистров общего назначения и 32 регистра с плавающей точкой.

Регистр	Вариант названия	Функция
R0	GO	Связан с 0. Все, что сохраняется в этом регистре, просто игнорируется
R1- R7	G1—G7	Содержит глобальные переменные
R8 – R13	O0 – O5	Содержит параметры вызываемой процедуры
R14	SP	Указатель стека
R15	O7	Временный регистр
R16 – R23	L0 – L7	Содержит локальные переменные для текущей процедуры
R24 – R29	I0 – I5	Содержит входные параметры
R30	FP	Указатель на основу текущего стекового фрейма
R31	I7	Адрес возврата для текущей процедуры

В действительности процессор UltraSPARC имеет более 32 регистров общего назначения, но видимы для программиста только 32 в любой момент времени. Эта особенность, называемая «регистровыми окнами»,

предназначена для повышения эффективности вызова процедур. Идея состоит в том, что имеется несколько наборов регистров, точно также, как существует несколько фреймов в стеке. Ровно 32 регистра видны в текущий момент; регистр CWP (Current Window Pointer – указатель текущего окна) следит за тем, какой набор регистров используется в данный момент.

Команда вызова процедуры скрывает старый набор регистров и предоставляет новый набор, который может использовать вызванная процедура. Однако некоторые регистры из старого набора переносятся в новый. Система эффективна, если нет многократных вложений.

В системе UltraSPARC III также есть 32 разряда с плавающей точкой, которые могут содержать либо 32-битные (одинарная точность), либо 64-битные (двойная точность) значения. Предусмотрена возможность использования пары регистров для поддержания 128-битных значений.

Архитектура UltraSPARC III – это архитектура загрузки/хранения. Это значит, что единственные операции, которые обращаются в память – это операции записи и чтения, служащие для передачи данных между регистрами и памятью. Все операнды для команд арифметических действий должны браться из регистров или предоставляться самой командой, а все результаты должны сохраняться в регистрах.

10. Типы данных. Типы данных процессора Pentium 4. Типы данных FPU, MMX, SIMD.

Типы данных

С точки зрения архитектуры ключевым вопросом является вопрос о том, осуществляется ли аппаратная поддержка конкретного типа данных. Под аппаратной поддержкой подразумевается, что одна или несколько команд ожидают данные в определенном формате и пользователь не может брать другой формат.

Типы данных процессора Pentium 4

Тип	1 бит	8 бит.	16 бит.	32 бита	64 бита	128 бита
Бит						
Целые числа со знаком		*	*	*		
Целые числа без знака		*	*	*		
Двоично-десятичные целые		*				

числа						
Числа с плавающей точкой				*	*	

Pentium 4 также может манипулировать 8-ми разрядными символами ASCII: существуют специальные команды для копирования и поиска цепочки символов. Эти команды используются и для цепочек, длина которых известна заранее, и для цепочек, в конце которых стоит специальный маркер. Эти операции часто используются в библиотеках операций над строками.

Операнды не обязательно должны быть выровнены в памяти, но если адреса слов кратны 4 (выравнивание по словам), то производительность повышается.

Типы данных FPU

Устройство FPU определяет численные данные в 7-ми внешних форматах, образуя при этом три класса:

- двоичные числа – целые числа (16, 32, 64 разряда);
- упакованные десятичные целые числа;
- двоичные вещественные числа – 32, 64, 80 разрядов.

Целые числа представляются в дополнительном коде и могут занимать 16 бит (целое слово), 32 бита (короткое целое), 64 бита (длинное целое). Эти форматы существуют только в памяти, внутри FPU они автоматически преобразуются в 86-разрядный расширяемый вещественный формат.

Упакованные десятичные целые. Они представляются в прямом коде и упакованном формате, т.е. каждый байт содержит две десятичные цифры. Старший бит старшего разряда байта отведен для знака числа, остальные биты этого байта игнорируются. Длина этого формата –10 байт (80 бит). Опять же этот формат существует только в памяти.

Вещественные числа. Имеется три формата с плавающей точкой (одинарной точности ОТ, двойной точности ДТ, расширенной точности РТ). Значащие числа находятся в поле мантиссы, поле порядка показывает фактическое положение десятичной точки в разрядах мантиссы, а бит знака определяет знак числа.

Специальные значения. Форматы чисел допускают представление специальных объектов и значений. Для их представления зарезервированы значения смещенного порядка 000 … 0 и 111 … 1.

Денормализованные вещественные числа – это число, которое меньше минимального нормализированного числа для каждого вещественного формата. Такие числа имеют минимальный смещенный порядок 000 … 0 и

ненулевую мантиссу, бит (F0) явный или неявный считается нулевым. Введены потому, что иногда промежуточные результаты вычислений могут принимать очень малые значения.

Истинный нуль. Нуль кодируется нулевым смещенным порядком и мантиссой. Нуль является знаковым (положительные или отрицательные нули).

Бесконечность. Поддерживается знаковое представление бесконечности. Это значение кодируется со смещенным порядком из всех единиц и мантиссой 1.000… . Знаки бесконечностей учитываются и сравнение возможно. Бесконечности интерпретируются так:

$-\infty < X < +\infty$.

Нечисла. Обозначаются NaN (Not-a-Number). Оно имеет смещенный порядок из всех единиц, любой знак и любую мантиссу, за исключением 1.000… 0. В свою очередь могут делиться на типы. Возникают иногда как результат недействительных операций. Могут использоваться для отладки программ.

Неподдерживаемые форматы. Формат PT имеет ряд двоичных наборов, не попадающих в рассмотренные классы. Они представляют устаревшие форматы, использовавшиеся в сопроцессоре 8087.

Типы данных MMX

Расширение MMX использует новые типы упакованных 64-битные целочисленных данных:

- 8 упакованных байт (Packed byte);
- 4 упакованных слова (Packed word);
- 2 упакованных двойных слова (Packed double word);
- 1 учетверенное слово (Quad word);

Типы данных XMM

Добавлены следующие типы данных

- Упакованная пара вещественных чисел двойной точности;
- Упакованные целые числа: 16 байт, 8 слов, 4 двойных слова или пара учетверенных слов.

11. Типы данных машины UltraSPARC III. Типы данных 8051

Типы данных машины UltraSPARC III

Тип	1	8	16	32	64	128

	бит	бит.	бит.	бита	бита	бита
Бит						
Целые числа со знаком		*	*	*	*	
Целые числа без знака		*	*	*	*	
Двоично-десятичные целые числа						
Числа с плавающей точкой				*	*	*

Все операнды должны быть выровнены в памяти.

UltraSPARC представляет собой регистровую структуру, и почти все команды оперируют 64-разрядными регистрами. Символьные и строковые типы данных специальными командами аппаратного обеспечения не поддерживаются.

Типы данных 8051

Количество типов данных строго ограничено. Поддерживается тип данных, не используемый в арифметических операциях – бит. Для работы с битами имеются специальные операции.

Тип	1 бит	8 бит.	16 бит.	32 бита	64 бита	128 бита
Бит	*					
Целые числа со знаком		*				
Целые числа без знака						
Двоично-десятичные целые числа						
Числа с плавающей точкой						

12. Форматы команд. Критерии разработки для форматов команд

Форматы команд

Команда – код, определяющий операцию вычислительной машины и данные, участвующие в операции. Содержит в явной или неявной форме информацию об адресе, по которому помещается результат операции и адрес следующей команды.

Команда:

операционная часть — адресная часть

1. Операционная часть - код операции - вид выполняемой операции
2. Адресная часть - информация об адресах операндов и результата операции, иногда об адресе следующей команды.

Структура команды определяется составом, назначением и расположением полей в команде. *Форматом команды* называют ее структуру с разметкой номеров разрядов (бит), определяющих границы отдельных полей команды, или с указанием числа бит в определенных полях.

Классические структуры команд:

- Принудительная - вся необходимая информация указывается в команде (код операции, операнд 1, операнд 2, результат, адрес следующей команды). Неэффективен и в настоящее время не применяется.

КОП	А1	А2	А3	А4

- Трёхадресная - обычно адрес следующей команды можно вывести из адреса текущей команды. В команде хранятся адреса результата и операндов (код операции, операнд 1, операнд 2, результат)

КОП	А1	А2	А3

- Двухадресная - результат помещается в один из операндов (код операции, операнд 1, операнд 2)

КОП	А1	А2

- Одноадресная - адрес второго операнда подразумевается (код операции, операнд)

КОП	А1

- Безадресная - подразумеваются адреса обоих операндов и результата.

Критерии разработки для форматов команд

- *Минимизация размера команды*: плюсы – уменьшение размера программы в памяти; минусы – усложнение декодирования. Если

кэш-память – t бит/с, средняя длина команды – r бит -> способность кэш-памяти передать максимум t/r команд/с (чем меньше команда, тем больше команд сможем получить) -> увеличение скорости работы процессора.

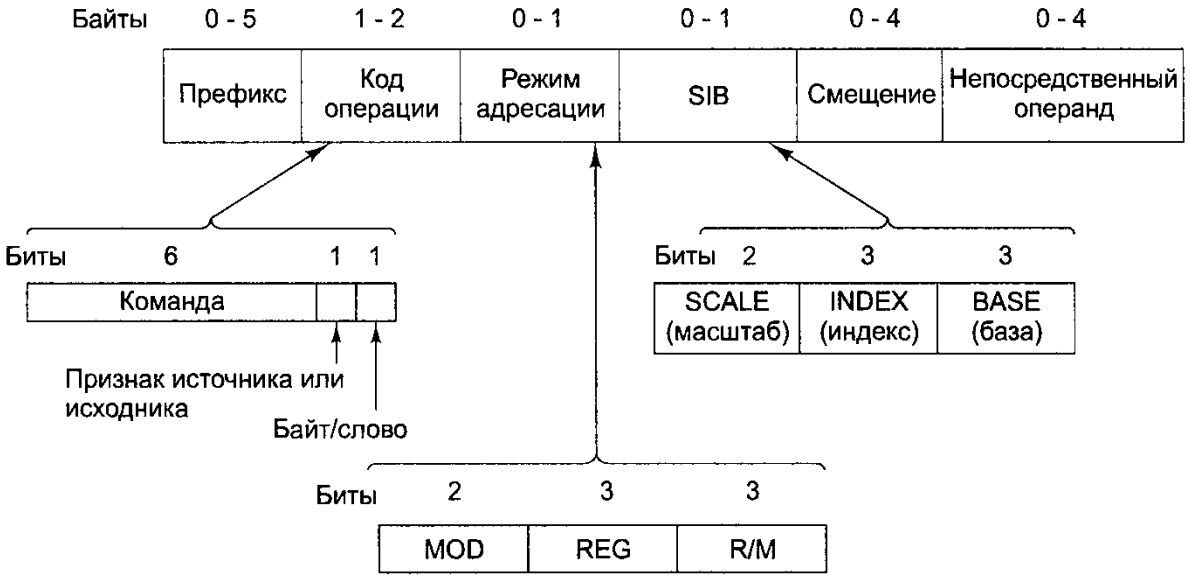
- *Формат команд*.
  - Достаточно большой объем пространства в формате команд для выражения всех требуемых команд.
    Размер поля команд $n_{КОП} = \log_2 M$, где M - число команд. Необходимо оставлять место для будущих добавлений команд.
  - Число битов в адресном поле. Для адресации S ячеек памяти адресное поле должно иметь разрядность $n_A = \log_2 S$. Для адресации памяти требуется более длинные адреса. Одна крайность – это организация памяти, при которой адресуется каждый бит. Другая крайность – это память, состоящая из очень длинных слов.

13. Форматы команд процессора Pentium 4

Форматы команд процессора Pentium 4 очень сложны и нерегулярны. Они содержат до шести полей разной длины, пять из которых не обязательны. Такая ситуация сложилась из-за того, что архитектура развивалась на протяжении нескольких поколений, и ранее в нее были включены не очень удачные варианты команд. Из-за требования обратной совместимости позднее их нельзя было изменить.

Поля:

1. префиксы команды (необязательны), могут следовать в произвольном порядке;
2. одного или двух байт (главного) кода операции;
3. спецификатора адреса, представленного битами mod r/m и sib;
4. смещения в команде (displacement) – если необходимо;
5. поле непосредственного значения (если такое значение присутствует).



В первых архитектурах Intel все коды операций были размером 1 байт, хотя для изменения некоторых команд часто использовался так называемый префиксный байт. Префиксный байт — это дополнительный код операции, который ставится перед командой, чтобы изменить ее действие.

Отдельные биты в кодах операций процессора Pentium 4 дают довольно мало информации о команде. Единственной структурой такого рода в поле кода операции является младший бит в некоторых командах, который указывает, что именно вызывается — слово или байт, а также соседний бит, который указывает, является ли адрес памяти (если он вообще есть) источником или приемником. Таким образом, в большинстве случаев код операции нужно полностью декодировать, чтобы установить, к какому классу относится выполняемая операция и, следовательно, какова длина команды. Это значительно снижает производительность, поскольку необходимо производить декодирование еще до того, как определено, где начинается следующая команда.

В большинстве команд вслед за байтом кода операции, который указывает местонахождение операнда в памяти, следует второй байт, несущий всю информацию об операнде. Эти 8 бит распределены в 2-разрядном поле MOD и двух 3-разрядных регистровых полях REG и R/M. Иногда первые 3 бита этого байта используются в качестве расширения для кода операции, давая в сумме 11 бит для кода операции. Тем не менее 2-разрядное поле означает, что существуют только 4 способа обращения к операндам и один из операндов всегда должен быть регистром. Логически должен идентифицироваться любой из регистров: EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, но правила кодирования команд запрещают некоторые комбинации, которые отводятся для особых случаев. В некоторых типах

команд требуется дополнительный байт, называемый SIB (Scale, Index, Base — масштаб, индекс, база), который дает дополнительную спецификацию.

Префикс – байт со специальным кодированием, модифицирует операцию находящейся за ним команды.

Префиксы:

1. префиксы команды (условный повтор операции и блокировка доступа к шине);
2. замены сегмента (определяет сегментный регистр для данной команды);
3. размера операнда (переключает 16– и 32–битные операнды);
4. размера адреса (коммутирует формирование 16– и 32–битных адресов);

Байт режима адресации сообщает всю информацию об операндах (2-битное поле mod и в два 3-битных регистровых поля reg и r/m).

Байт SIB может присутствовать только в командах с 32-разрядной адресацией, когда байт mod r/m содержит r/m = 100 и значение в поле mod≠11.

14. Форматы команд процессора UltraSPARC III.

Команды 32-битные, выровненные в памяти. Каждая команда определяет только одно действие. Типичная команда указывает два регистра, из которых берутся операнды и выходной регистр. Вместо одного регистра допускается использование константы со знаком. При выполнении команды считывания два регистра (или один регистр и одна константа) складываются вместе и определяют адрес памяти, по которому производится считывание; результат записывается в указанный регистр.

Изначально SPARC имела ограниченное число форматов . Со временем добавлялись дополнительные форматы и сейчас их 31.

Первые два бита команды помогают определить формат команды и сообщают программному обеспечению, где найти оставшуюся часть кода операции, если она есть.

В формате 1a оба источника расположены в регистрах. В формате 1b один операнд – константа. Бит 13 (0 или 1) определяет один из этих форматов.

Поскольку все команды 32-битные, то включить в команду 32-битную константу невозможно. Команда SETHI устанавливает 22 бита, оставляя пространство для другой команды, чтобы установить оставшиеся 10 битов. Это единственная команда, использующая данный формат.

Для непрогнозируемых условных переходов используется формат 3, в котором поле условие определяет, какое условие надо проверить. Бит A нужен для того, чтобы избегать пустых операций при определенных условиях. Прогнозируемые переходы используют тот же формат, но только с 19-битным смещением.

Последний формат используется для вызова процедур. Требуемый адрес – это целевой адрес, разделенный на четыре.

15. Адресация. Способы адресации.

Адресация.

Для адресации S ячеек памяти адресное поле должно иметь разрядность $n_A \geq \log_2 S$.

Это требование находится в противоречии с желанием иметь малую разрядность команды и возможностью использовать большое адресное

пространство. Существуют два основных подхода к решению этой проблемы.

1. Использовать регистры общего назначения. Это существенно повышает быстродействие, но имеет эффект только в том случае, если операнд используется многократно. Кроме того, существует ограничение на количество регистров общего назначения.
2. Второй подход подразумевает определение одного или нескольких операндов неявным образом. Это означает использование не трехадресной команды, а двух- одно- и безадресных команд.

Способы адресации

1. Подразумеваемый операнд. Операнд подразумевается и фактически задается кодом операции команды.
2. Подразумеваемый адрес. В команде не содержится явных указаний об адресе участвующих в операции операндов, или адреса, по которому помещается результат операции, но этот адрес подразумевается.
3. Непосредственная адресация. В команде содержится не адрес операнда, а непосредственно операнд.
4. Прямая адресация. Исполнительный адрес совпадает с адресной частью кода команды.
5. Регистровая адресация. В качестве операнда используется содержимое регистров процессора.
6. Косвенная адресация. Адресный код команды указывает адрес ячейки памяти, в которой находится адрес операнда или команды.
7. Относительная адресация или базирование. Исполнительный адрес определяется суммой адресного кода команды Ак и некоторого числа Аб, называемого базовым адресом: Аи = Аб + Ак. Позволяет при меньшей длине адресного кода команды обеспечить доступ к любой ячейке памяти. Обеспечивает перемещаемость программ - возможность перемещения программ в памяти без изменений внутри самой программы.
8. Индексная адресация. Формирование исполнительного адреса осуществляется путём добавления к базовому адресу содержимого индексного регистра.
9. Стековая адресация. Адресация посредством регистра — указателя стека. Стек - группа последовательно пронумерованных регистров (аппаратный стек) или ячеек памяти, снабженных указателем стека (обычно регистром SP), в котором автоматически при записи и считывании указывается номер (адрес) последней занятой ячейки

стека (вершины стека). При выполнении операции записи в стек слово помещается в следующую ячейку стека, а при считывании из стека последнее поступившее в него слово. Таким образом, в стеке реализуется дисциплина обслуживания «последний пришел – первый ушел» (LIFO).

Способы адресации команд перехода

Способы:

1. Прямая адресация. Целевой адрес полностью включается в команду.
2. Косвенная и регистровая адресация. Вычисление целевого адреса, помещение его в регистр, а затем переход по этому адресу.
3. Индексная адресация. Известно смещение от регистра.
4. Относительная адресация по счетчику команд. Смещение (со знаком) прибавляется к программному счетчику.

16. Способы адресации процессора Pentium 4.

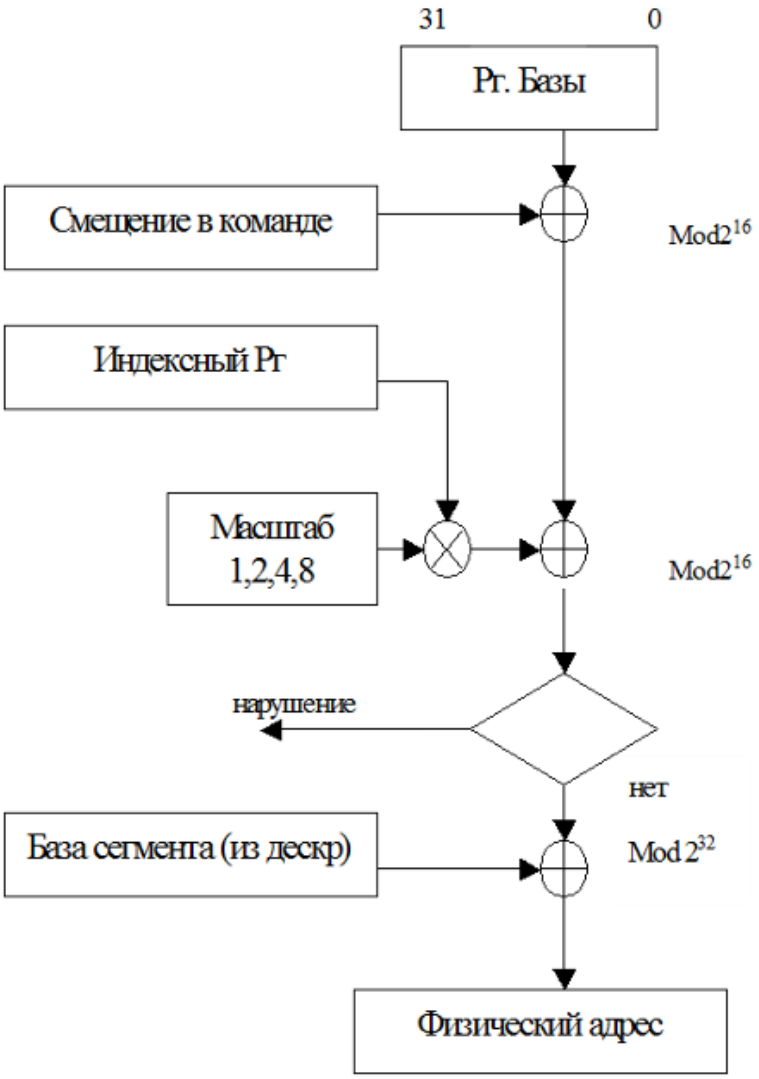
Способы адресации процессора Pentium II нерегулярны и зависят от того, в каком формате находятся команды: 16- или 32-битном.

Способы адресации:

Непосредственная	Mov	eax,	12345678h
Регистровая	Mov	eax,	ecx
Прямая (абсолютная)	Mov	eax,	[3456789h]
Регистровая косвенная	Mov	eax,	[ecx]
Базовая (индексная) со смещением	Mov	eax,	[ecx]+1200h
Базовая индексная со смещением	Mov	eax,	[ecx][edx]+40h
Индексная с масштабированием и смещением	Mov	eax,	[eci*4]+400h
Базовая индексация с масштабированием	Mov	eax,	[edx][ecx*8]
Базовая индексация с масштабированием и смещением	Mov	eax,	[ebx][edi*2]+20h

Не все способы адресации применимы ко всем командам и не все регистры могут использоваться при всех способах адресации. Это существенно усложняет работу компилятора.

Схема формирования адреса в 32-разрядной адресации:



17. Способы адресации процессора UltraSPARC III. Способы адресации машины 8051.

UltraSPARC III

Арифметические и логические операции:

1. Регистровый. 5 битов просто сообщают, какой регистр нужно использовать.
2. Непосредственная адресация. 13-битная константа со знаком обеспечивает данные адреса.

Операции считывания (Load), записи (Store) и синхронизации памяти:

1. Вычисляется сумма двух регистров, а затем через полученное значение производится косвенная адресация.
2. Индексирование с 13-битным смещением со знаком.

8051

Режимы:

1. Неявная адресация. Первый операнд находится в сумматоре, второй в памяти или регистрах.
2. Регистровая адресация. Регистры могут быть как входными, так и выходными.
3. Прямая адресация. Адрес операнд указан в команде.
4. Косвенная регистровая адресация, подразумевающая размещение в регистре указателя на операнд.
5. Непосредственная адресация. Операнд является частью команды.

За взаимодействие с внешней памятью программ отвечают команды LJMP и LCALL. Для взаимодействия с внешней памятью данных используется 16-разрядный регистр DPTR, в котором размещаются 16-разрядные адреса памяти.

18. INTEL IA-64 и процессор Itanium 2

Со временем увеличивать скорость работы IA-32 становилось все сложнее и решением проблемы стало разработка совершенно новой архитектуры команд. Новая архитектура, которая разрабатывалась совместно компаниями Intel и Hewlett Packard получила название **IA-64.** Это полностью 64-битная машина. Самым первым процессором этого типа был процессор **Itanium,** обладающий высокой производительностью.

Проблемы архитектуры IA-32:

1. Ориентация на двухадресные команды. Сейчас стандартом является обращение в память только командами загрузки-сохранения, а вся работа производится на регистрах.
2. Небольшой и нерегулярный набор регистров. Постоянные обращения в память.
3. Переупорядочивание команд для оптимального использования ресурсов и последующее упорядочивание результатов для соответствия логике программы.
4. Спекулятивное выполнение для компенсации возможности неправильного предсказания ветвлений.
5. Преобразование CISC-инструкций в RISC-инструкции.

IA-64

Основной принцип архитектуры - перенести нагрузку периода выполнения на период компиляции

Компилятор:

1. генерирует программу, в которой нет конфликтов между регистрами.
2. следит за загрузкой функциональных блоков.

Особенности IA-64, заметно повышающие производительность:

1. Сокращение числа обращений в память
   Увеличение числа регистров:
   128 64-разрядных регистров общего назначения. Первые 32 статические, остальные группируются в стек регистров.
   128 регистров с плавающей точкой, организованных по стандарту IEEE 745, но не группируемых в стек.
   64-разрядных предикатных регистра, 8 регистров ветвления и 128 специализированных прикладных регистра, которые могут использоваться для обмена параметрами между прикладными программами и операционной системой.
2. Планирование команд
   Функции планирования возложены на компилятор.
   Команды поступают группами по три штуки. Такая группа называется пучком. Каждый 128-битный пучок содержит три 40-битных команды фиксированного формата и 8-битный шаблон. Пучки могут быть связаны между собой, поэтому в пучке может быть более трех команд. Формат содержит информацию о том, какие команды могут выполняться параллельно.
3. Сокращение числа условных переходов
   Механизм предикации. Команды содержат условия, которые сообщают, в каком случае надо выполнять команду, а в каком нет.
4. Спекулятивная загрузка
   Компилятор помещает команды считывания из памяти раньше, чем потребуется результат. Перед операцией со значением вставляется команда CHECK. Если запрошенное значение присутствует, операция выполняется, иначе простой операции.

19. Организации памяти. Проблемы защиты памяти.

Организация памяти

Нехватка памяти -> попытки использовать внешние накопители.

Способы:

1. Оверлеи - программист разбивает программу на несколько частей и загружает нужный оверлей. Контроль за разбиением, хранением и загрузкой оверлеев лежал полностью на программисте.
2. Виртуальная память - автоматизированный процесс динамического разбиения памяти. Предотвращение несанкционированного воздействия одних программ на другие достигается за счёт механизма защиты памяти.

Защита памяти

Защита от разрушения данных другими программами - запрет записи этими программами в адресное пространство защищаемой программы.

Защита от копирования - защита от записи и чтения из области памяти программы.

Необходимо выявлять и предотвращать интерпретацию данных как кода и кода как данных.

Варианты защиты:

- задается отношение к областям памяти чужой программы, определяющее, относится защита только к операции записи или к любому обращению к памяти;
- задается одно из следующих отношений к области памяти собственной программы:
  1. разрешается полный доступ к данному блоку памяти;
  2. разрешается только считывание;
  3. разрешается обращение только через счетчик команд;
  4. разрешается обращение за исключением счетчика команд.

Нарушение защиты памяти - запрос прерывания по нарушению защиты памяти.

Виды реализаций защиты памяти:

1. Защита отдельных ячеек памяти. Каждая ячейка -> информация о защите доступа.
2. Защита области памяти. Память делится на блоки. Каждый блок -> ключ защиты. Каждая программа -> ключ программы. Доступ программы к блоку -> анализ ключа программы и ключа защиты.

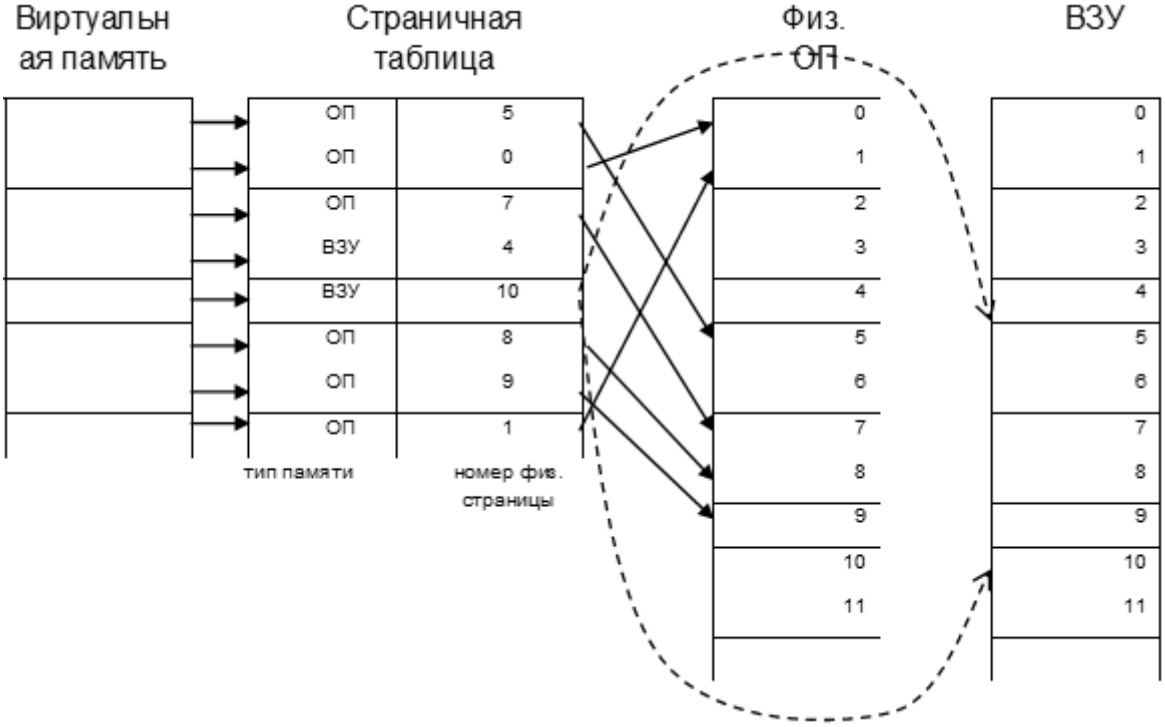
20. Принципы организации виртуальной памяти.

Программист работает не с физической памятью, имеющей некоторую фиксированную емкость, а с виртуальной одноуровневой памятью, объем

которой равен всему адресному пространству. Программа преобразует виртуальные адреса в физические только при своём исполнении.

Физическая и виртуальная память разбивается на страницы, содержащие одинаковое число байт. Страницам присваиваются номера. Каждая физическая страница способна хранить одну из виртуальных страниц. Порядок расположения байт в виртуальной и физической страницах сохраняется одним и тем же.

Соответствие между виртуальной и физической памятью устанавливается так называемой страничной таблицей.



Физические страницы могут содержаться как в оперативной, так и во внешней памяти. Страничная таблица формируется ОС.

Для каждой из программ, обрабатываемых в мультипрограммном режиме, организуется своя виртуальная память и создается своя страничная таблица, при этом все программы делят между собой общую физическую память.

Работа со страницами, находящимися на внешнем запоминающем устройстве:

1. Вызов страниц по требованию
   Обращение к адресу страницы, отсутствующей в ОЗУ -> ошибка отсутствия страницы. ОС считывает нужную страницу из памяти, вводит новый адрес в таблицу страниц и повторяет команду.
   Страницы вызываются по мере необходимости, а не заранее.
2. Рабочее множество

Обычно большинство обращений команд относится к небольшому числу страниц. В каждый момент времени t существует набор страниц, которые использовались за последние k обращений.

3. Политика замещения страниц
   При возникновении нехватки памяти ОЗУ происходит вытеснение страниц из рабочего множества на внешнее ЗУ.

   Алгоритмы определения вытесняемой страницы:

1. LRU. По этому алгоритму удаляется та страница, которая использовалась наиболее давно.
2. FIFO. Удаляет первую поступившую в память страницу.Возможно ошибочное удаление наиболее «активной» страницы.

   Если в страницу не производилась запись, её можно просто удалить.

21. Сегментация. Дескрипторные таблицы. Формирование адреса.

Сегментация

Сегмент - линейная последовательность адресов от 0 до некоторого максимума. Длина может быть любой (в допустимых пределах) и может меняться в процессе выполнения программы. Указатель: номер сегмента и адрес внутри сегмента.

Преимущества:

1. Если каждая процедура занимает свой сегмент, то связывание раздельно компилирующихся процедур сильно упрощается. Для обращения к I-му слову n-й процедуры используем адрес (n,i).
2. Изменение процедуры в отдельном сегменте без затрагивания остальных сегментов.
3. Облегчает разделение общих процедур и данных между несколькими программами.
4. Разные сегменты могут иметь разные виды защиты.

Сравнение страничной организации памяти и сегментации.

Свойство	Страничная организация памяти	Сегментация
Необходимость учета программистом	Нет	Да
Количество линейных адресных	1	много

пространств		
Возможность увеличения памяти для виртуального адресного пространства	Да	Да
Простота управления структурами переменного размера	Нет	да
Назначение технологии	Имитация памяти большого размера	Предоставление нескольких адресных пространств

Способы реализации:

1. Подкачка. В памяти находится некотрый набор сегментов. Обращение к сегменту на внешнем запоминающем устройстве - перенос в ОЗУ. Если памяти ОЗУ недостаточно - запись нескольких сегментов на ВЗУ.
2. Разбиение на страницы. Разделение каждого сегмента на страницы фиксированного размера и вызов по требованию. Необходимо иметь отдельную таблицу страниц для каждого сегмента.

Дескрипторные таблицы

Дескрипторная таблица - массив из 8–байтных элементов – дескрипторов. Порядок размещения дескрипторов не играет роли. Максимальное число дескрипторов - 8192. Максимальный размер таблицы 64 Кбайт.

Типы дескрипторных таблиц:

1. Глобальная дескрипторная таблица (GDT). Главная общесистемная таблица дескрипторов. Её могут использовать все программы. Местонахождение - регистр GDT - 32-разрядное поле линейного базового адреса и 16-разрядное поле предела L=8*N-1, где N – число дескрипторов.
2. Дескрипторная таблица прерываний (IDT). Общесистемная. Содержит дескрипторы шлюзов (gate), определяющих точки входов в процедуры обработки прерываний и особых случаев. Расположение - регистр IDTR - аналогично по структуре с GDTR.
3. Локальная дескрипторная таблица (LDT). Локальна для задачи и определяет сегменты, доступные только для неё. Расположение - регистр LDTR - селектор сегмента, содержащего LDT.

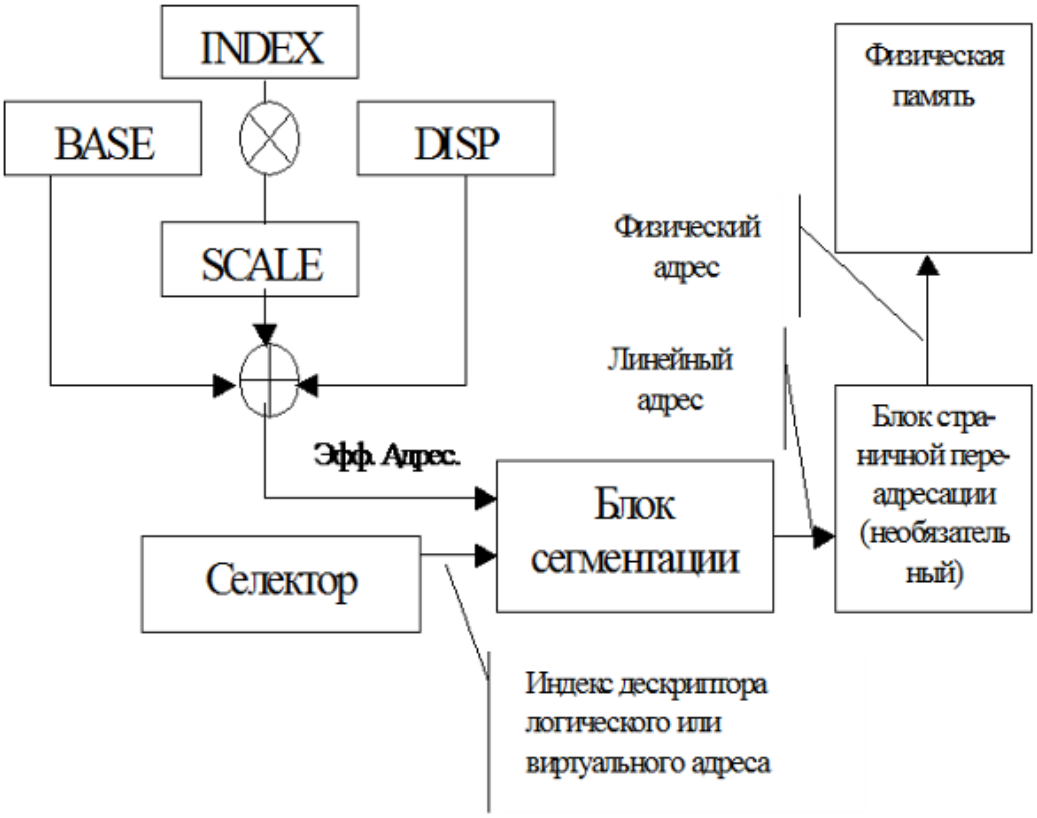
Формирование адреса.

Виды адресов:

1. Логический адрес. Формируется из 16-битного селектора сегмента и 32(или 16 в Real Address Mode)– разрядного смещения относительно начала сегмента. Существует только внутри программного обеспечения.
2. Линейный адрес. Формируется из логического. Предназначен для обращения к линейному (непрерывному и несегментированному) пространству объемом 2^{32} байт.
3. Физический адрес. Передается на внешнюю шину для обращения к ячейкам памяти. Адресуется 2^{32} байт в обычном режиме и до 2^{36} байт в случае поддержки механизма расширения физического адреса.

Адресация в различных режимах работы:

1. Real Address Mode. Физический адрес = Линейный адрес = База (селектор) сегмента + смещение относительно начала сегмента. До 1 Мб памяти.
2. Protected Virtual Address Mode – защищенный режим виртуальной адресации. До 4 Гб физической памяти. Действия механизма образования физического адреса основано на использовании дескрипторных таблиц. Схема формирования адреса :



3. Virtual 8086 Mode – режим виртуального процессора 8086. Особое состояние задачи защищенного режима, в котором процессор функционирует как 8086. Параллельно могут выполняться несколько задач с изолированными друг от друга ресурсами. Использование

физического адресного пространства памяти управляется механизмами сегментации и трансляции страниц.

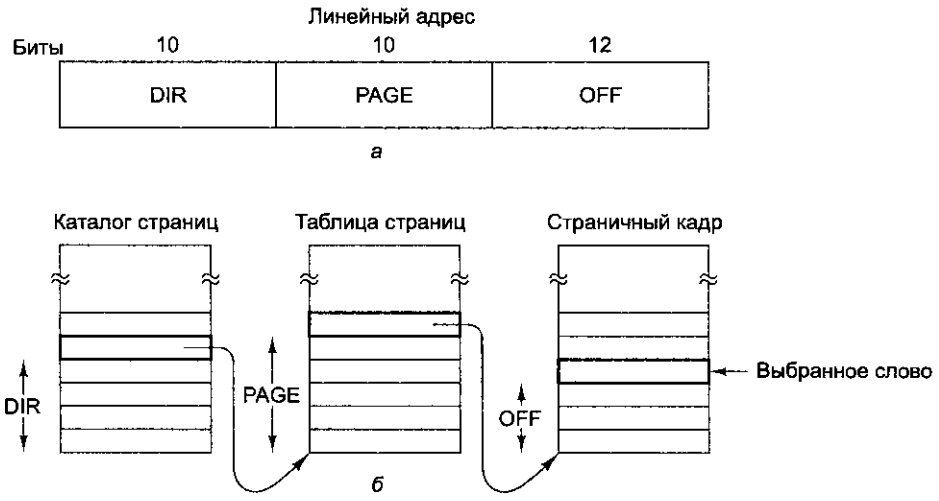
4. System Management Mode (SMM). Изолированное от остальных режимов пространство памяти. Используется в служебных и отладочных целях.

22. Виртуальная память со страничной организацией в Pentium IV

Линейное и физическое адресное пространство делится на 1 Мб страниц по 4 Кб. Границы сегментов и границы страниц не зависят друг от друга и не обязаны быть выровнены. Старшие 20 бит 32-разрядного линейного адреса через страничную таблицу замещаются номером физической страницы. Младшие 12 бит остаются неизменными.

Таблица страниц в 1 Мб элементов - слишком велика (для каждой задачи нужна своя таблица) -> двухуровневое преобразование.

Корневая таблица называется таблицей страниц первого уровня или просто каталог страниц, содержит 1024 32-разрядных дескриптора, называемых элементами каталога страниц PDE. Каждый из них адресует подчиненную таблицу страниц (таблицу страниц второго уровня). Каждая подчиненная таблица содержит 1024 32-разрядных дескриптора, называемых элементами таблицы страниц (PTE), каждый из которых адресует страницу в адресной памяти.



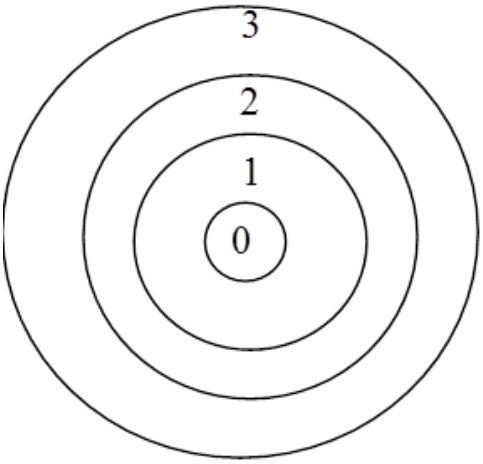
Преобразование линейного адреса в физический состоит из следующих действий:

1. старшие 10 бит (31–22) линейного адреса служат индексом в каталоге страниц, выбирая один из 1024 элементов PDE, который выбирает таблицу страниц;
2. средние 10 бит (21–12) индексируют таблицу страниц, выбирая из нее элемент PTE, который содержит 20–разрядный базовый физический адрес в памяти;
3. базовый адрес из PTE объединяется с 12 младшими битами линейного адреса, после чего получается физический адрес.

23. Защита по привилегиям в архитектуре x86.

Термин «привилегия» - права и возможности, которые обычно не разрешаются.

Процессоры x86 поддерживают 4 уровня привилегий: 0, 1, 2, 3. Чем меньше номер уровня, тем он больше привилегирован. Уровни привилегий обычно изображаются в виде колец защиты.



При выполнении почти каждой машинной команды осуществляется проверка защиты по привилегиям. Процессор в защищенном режиме постоянно контролирует, что текущая программа достаточно привилегированна, чтобы:

1. выполнять некоторые команды;
2. обращаться к данным других программ;
   программам не разрешается доступ к данным, которые имеют более высокий уровень привилегий.
3. передавать управление внешнему (по отношению к самой программе) коду командами передачи управления типа FAR.
   Передача управления (FAR JMP, FAR CALL) не допускается за пределы своего поля защиты.

С каждым сегментом кода, данных или стека ассоциируется уровень привилегий и все, что находится внутри этого сегмента имеет этот уровень

привилегий. Уровень привилегий выполняющегося в данный момент сегмента кода называется текущим уровнем привилегий (CPL) и он задается полем RPL селектора в регистре CS. При передаче управления сегменту кода с другим уровнем привилегий процессор будет работать на новом уровне привилегий. Часто говорят, например, «программа выполняется в кольце 0», подразумевая, что CPL процессора равен 0.

Правила защиты:

Команды, воздействующие на механизм сегментации и защиты могут выполняться только в нулевом кольце (PL0-программы). Вторую группу образуют команды, которые изменяют состояние флажка прерываний IF и проводят ввод–вывод.

Защита доступа к данным.

Процессор не разрешает обращаться к данным, которые более привилегированны, чем выполняемая программа. Основное правило защиты имеет вид: CPL ≤ DPL. Когда программа пытается осуществить обращение с нарушением этого правила, процессор отказывается произвести его и сообщает о нарушении общей защиты.

Контроль реализуется двумя способами.

1. Контроль осуществляется при загрузке селектора в один из регистров DS, ES, FS, GS.
2. После успешной загрузки селектора при использовании его для фактического обращения к памяти процессор контролировать, чтобы запрашиваемая операция была разрешена.

Защита сегментов кода.

Запрещает передачу управления сегменту кода, находящемуся на другом уровне привилегий, тем самым предотвращая произвольное изменение уровня привилегий. Если бы CPL можно было бы легко изменять, все остальные средства защиты оставались бы бессмысленными.

При загрузке CS(code segment) процессор предпринимает следующие проверки:

- проверяется, что новый дескриптор является сегментом кода;
- проверяется DPL = CPL, и бит присутствия P.

Если все проверки прошли, то дескриптор используется.

Передача управления между уровнями привилегий

Способы передачи управления более высоким уровням привилегий:

1. Подчиненные сегменты кода. С подчиненными сегментами кода не ассоциируется конкретный уровень привилегий, т.к. они подчиняются уровню привилегий того кода, который передает или управление с

помощью команд CALL или JMP. Если, например, подчиненный сегмент кода вызывает программа из кольца 3, то он работает с CPL = 3, если из кольца 0, то с CPL = 0.

Применительно к подчиненному сегменту кода действует одно ограничение. Оно заключается в том, что значение DPL дескриптора подчиненного сегмента кода всегда должно быть меньше или равно текущему значению CPL. Другими словами, передача управления разрешается только во внутренне более защищенные кольца.

2. Шлюзы вызова. Для реализации фактического изменения уровней привилегий привлекаются особые системные объекты, называемые шлюзами вызова. Дескриптор шлюза вызова действует как посредник между сегментами кода, находящимися на различных уровнях привилегий. Шлюзы вызова идентифицируют разрешенные точки в более привилегированном коде, которым может быть передано управление, и являются единственным средством смены уровня привилегий.

Реализованный в x86 такой косвенный вызов привилегированных процедур имеет несколько преимуществ:

1. Привилегированный код сильно защищен и вызывающие программы не могут его разрушить.
2. Шлюзы вызова делают код процедуры невидимым для программ на внешних уровнях привилегий.
3. Так как вызывающая программа прямо адресует только шлюз, реализуемые процедурой функции можно изменять или перемещать в адресном пространстве, не затрагивая интерфейс со шлюзом.

24. Виртуальная память со страничной организацией в UltraSPARC III.

Виртуальная память:

1. Страничная организация памяти
2. 64-битные виртуальные адреса
3. Невозможность использования всего 64-битного пространства (физические и программные препятствия)
4. Доступные области памяти - 2^{43} байт в начале и конце адресного пространства
5. Четыре размера страниц: 8, 64, 512 Кбайт и 4 Мбайта.

6. Максимальная физическая память - 2^{41} байт (2200 Гбайт)
7. Большое адресное пространство -> неэффективность обычной таблицы страниц -> таблица TLB (Translation Lookaside Buffer - буфер быстрого преобразования адреса) содержит номера только последних используемых виртуальных страниц. Страницы команд и данных рассматриваются отдельно. Обращение к виртуальной странице -> поиск в TLB: попадание - объединение номера страничного кадра со смещением из виртуального адреса в 41-битный физический адрес, промах (может случиться даже при присутствии страницы в ОЗУ)- вызов ловушки в операционной системе.
8. Наиболее часто используемые элементы TLB операционная система сохраняет в TSB (translation storage buffer - буфер хранения преобразований) - кэш-память прямого отображения виртуальных страниц.
9. Страницы нет в TLB - используется таблица трансляции. Аппаратное обеспечение не используется -> ОС может использовать любой формат данной таблицы.
10. Таблицы нет в ОЗУ - стандартная ошибка.

Виртуальная память и кэширование

Сходство виртуальной и кэш-памяти:

Параметр	Виртуальная память	Кэш-память
Хранение	На диске	В ОЗУ
Разбиение	Страницы фиксированного размера	Блоки фиксированного размера
Нахождение блоков в основной памяти (кэш/виртуальная)	Некоторое подмножество страниц	Некоторое подмножество блоков
Условие эффективной работы	Нахождение данных главным образом в ОЗУ	Нахождение данных главным образом в кэш-памяти

Из этого следует, что виртуальная и кэш-память просто работают на разных уровнях иерархии памяти.

Различия в виртуальной и кэш-памяти:

1. Промахи кэш-памяти - аппаратное обеспечение, ошибки из-за отсутствия страниц - операционная система.

2. Блоки кэш-памяти обычно гораздо меньше страниц (64 байта или 8 Кбайт).
3. Таблицы страниц индексируются по старшим битам виртуального адреса, кэш-память - по младшим битам адреса памяти.

25. Архитектуры компьютеров параллельного действия.

Многое можно сказать о программном обеспечении для параллельных компьютеров, но сейчас мы должны вернуться к основной теме данной главы — архитектуре компьютеров параллельного действия. Было предложено и построено множество различных видов параллельных компьютеров, поэтому хотелось бы узнать, можно ли их как-либо категоризировать. Это пытались сделать многие исследователи. К сожалению, хорошей классификации компьютеров параллельного действия до сих пор не существует. Чаще всего используется классификация Флинна (Flynn), но даже она является очень грубым приближением.

Классификация Флинна:

3. SISD (одна инструкция, один поток данных). Классический фон Неймановский компьютер.
4. SIMD (одна инструкция, множественные данные). Один блок управления, выдающий по одной команде, но при этом имеется несколько АЛУ, которые могут обрабатывать несколько наборов данных одновременно.
5. MISD (множественные инструкции, один поток данных). Несколько команд оперируют над одними данными. Возможно конвейер.
6. MIMD (множественные инструкции, множественные данные). Несколько независимых процессов работают как часть большой системы. Мультикомпьютеры и мультипроцессоры.

Виды параллелизма:

1. На уровне процессора. Ковейеризация и суперскалярная архитектура с несколькими функциональными блоками. Одновременная обработка нескольких программных потоков. Несколько процессорных элементов. Повышение производительности максимум в 10 раз.
2. Внедрение внешних плат ЦП с улучшенными вычислительными возможностями. Реализуются специальные возможности (обработка сетевых пакетов, цифровая обработка сигналов, криптография).

Производительность специализированных приложений возрастает в 5-10 раз.

3. Мультипроцессоры и мультикомпьютеры. Объединение многих процессоров или компьютеров в единую систему. Увеличение производительности зависит от типа объединения.
4. Интеграция через Интернет. Слабо связанные распределенные вычислительные сетки или решетки.

26. Вопросы разработки компьютеров параллельного действия.

Скорость работы компьютеров становится все выше, но и требования к ним постоянно возрастает и задача повышения производительности компьютерных систем всегда будет актуальной.

Хотя тактовая частота постоянно растет, скорость коммуникации нельзя увеличивать постоянно. Нельзя бесконечно уменьшать размеры транзисторов, поскольку существуют законы квантовой механики (принцип неопределенности Гейзенберга).

Для повышения производительности систем необходимо использовать архитектурные решения, а именно разрабатывать вычислительные системы параллельного действия (параллельные системы).

Параллелизм можно вводить на разных уровнях.

На самом низком уровне он может быть реализован в процессоре за счет ковейеризации и суперскалярой архитектуры с несколькими функциональными блоками. Можно организовать одновременную обработку нескольких программных потоков. Можно установить несколько процессорных элементов. Однако все эти приемы способны повысить производительность максимум в 10 раз по сравнению с классическими последовательными решениями.

На следующем уровне возможно внедрение внешних плат ЦП с улучшенными вычислительными возможностями. Как правило в подключенных модулях реализуются некоторые специальные возможности, например сетевых пакетов, цифровая обработка сигналов, криптографии и т.д. Производительность специализированных приложений возрастает в 5-10 раз.

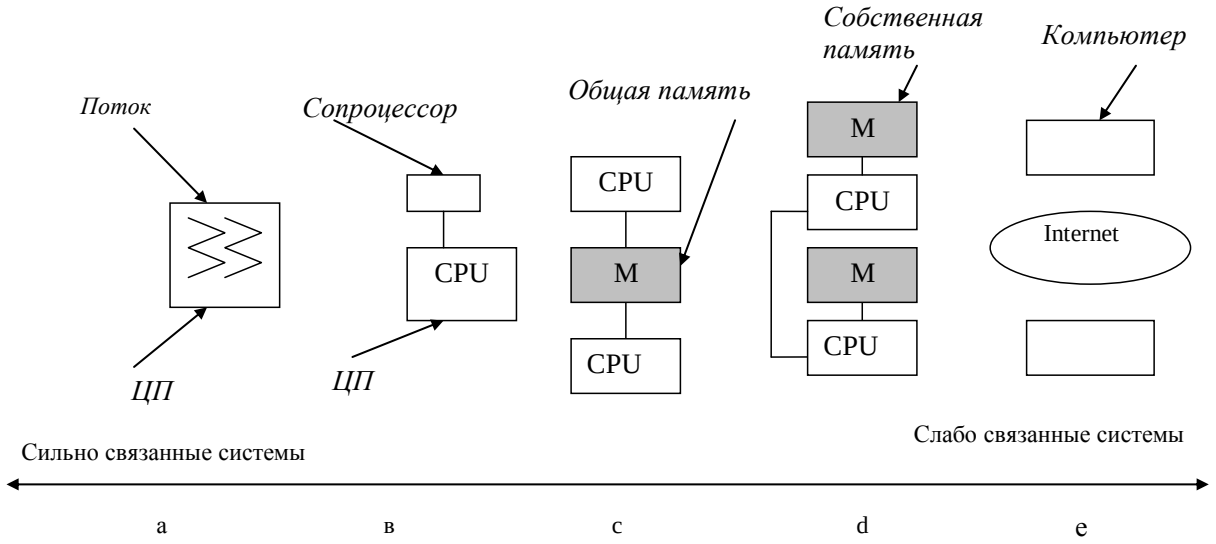
Для увеличения производительности в 100, 1000, 1000000 раз необходимо свести воедино многочисленные процессоры и обеспечить их эффективное взаимодействие. Этот принцип реализуется в виде больших

мультипроцессорных и мультикомпьютерных систем. Объединение тысяч процессоров в единую систему порождает новые проблемы, некоторые из которых мы будем рассматривать.

В последнее время появилась возможность интеграции через Интернет. В результате появляются слабо связанные распределенные вычислительные сетки или решетки.

Когда два процессора (или обрабатывающие элементы) находятся рядом и обмениваются большими объемами данных с небольшими задержками, они называются сильно связанными (tightly couplet). Соответственно, если два процессора располагаются далеко друг от друга и обмениваются небольшими объемами информации с большими задержками, они называются слабо связанными (loosely couplet).

Рассматриваемые системы параллельной обработки иллюстрируются рисунком 6.1.

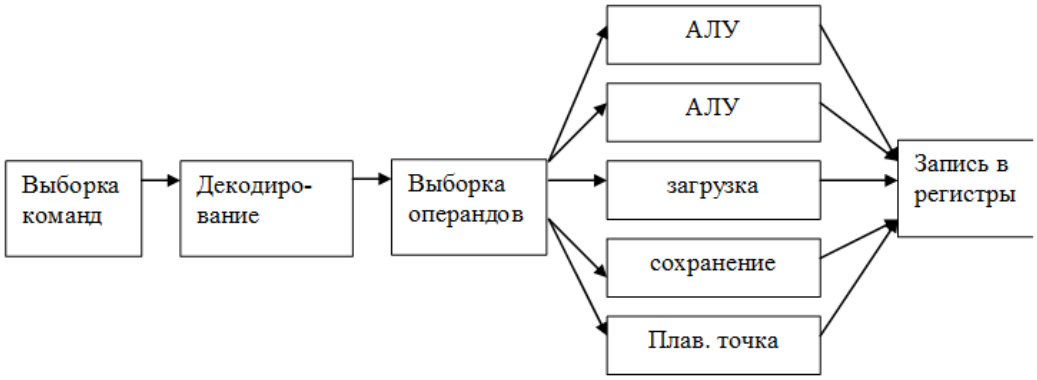


27. Параллелизм на уровне команд. VLIW-процессор. Внутрипроцессорная многопоточность.

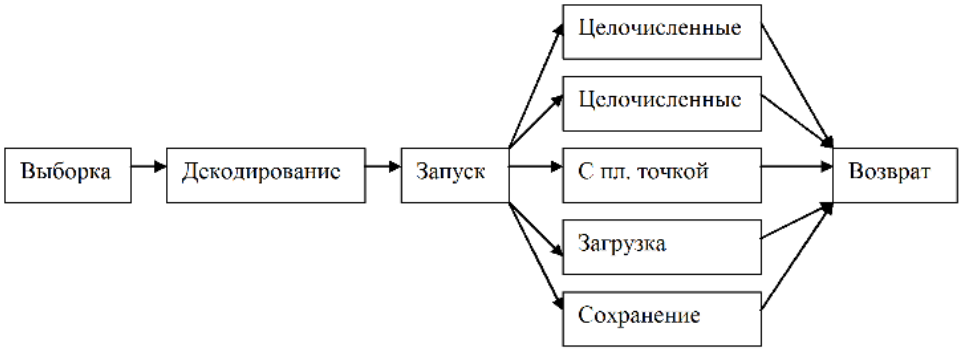
Параллелизм на уровне команд

Процессоры, которые вызывают несколько команд за один цикл, делятся на:

1. Суперскалярные. Вызывают несколько команд за один тактовый цикл, что определяется как аппаратной реализацией, так и последовательностью команд.



2. VLIW (Very Long Instruction Word – процессоры со сверхдлинным словом). В одной команде содержится пять кодов операций и пять пар операндов. Таким образом, код операции -- 6 бит, регистр – 5 бит, память – 32 бита, а общая длина команды 134 бита. Такое решение было признано неудачным, т.к. появлялось большое количество бессмысленных операций.



В современных VLIW-процессорах задача по подготовке и завершению связок выполняется компилятором.

Преимущества:

1. Упрощается аппаратное обеспечение;
2. Поскольку на стадии компиляции нет жестких временных ограничений, то поиск и составление связок может быть выполнен более качественно.

VLIW-процессор TriMedia

Встроенный процессор для устройств обработки изображений, видео- и аудиоданных.

RISC-процессор с расширенной функциональностью. Проверка операций на совместимость на этапе исполнения не производится и возлагается на компилятор.

Команда - до 5 операций. Оптимальные условия - запуск одной команды и выбор пяти операций за цикл.

Слот 1	Слот 2	Слот 3	Слот 4	Слот 5

Сложение	сдвиг	мультимедиа	загрузка	сохранение

Память с байтовой организацией. Полуслова (16 бит), слова (32 бита) выровнены. Порядок следования байтов выставляется операционной системой. КЭШ-память – разделенная, 8- входовая, ассоциативная, длина строки 64 байт. КЭШ команд – 64 Кбайт, данных – 16 Кбайт.

Регистры:

1. 128 универсальных 32-разрядных
2. R0 и R1 равны машинному нулю и единице
3. 4 специальных регистра:
    1. счетчик команд
    2. слово состояния программы
    3. два регистра, связанных с прерываниями

64 разрядный счетчик, подсчитывающий число циклов процессора с момента последнего сброса.

Функциональные блоки:

1. операции с константами
2. АЛУ целочисленных операций
3. Сдвиги
4. Загрузка и сохранение
5. Умножение (целых и с плавающей точкой)
6. АЛУ с пл. точкой
7. Сравнение с ПТ
8. Извлечение корня и деления с ПТ
9. Ветвления
10. АЛУ ЦОС
11. Умножитель для ЦОС

Внутрипроцессорная многопоточность

Позволяет процессору одновременно управлять несколькими программными потоками и маскировать простои. Общая идея: если программный поток 1 блокируется, процессор может обеспечить полную загрузку аппаратуры, запустив программный поток 2. Существуют различные способы реализации этой идеи.

1. Мелкомодульная многопоточность.Простой маскируется путем исполнения потоков «по кругу», т.е. в смежных циклах запускаются по циклу. При простое в n циклов необходимо иметь n+1 поток.

2. Крупномодульная многопоточность. Программный поток А выполняется до тех пор, пока не возникнет простой. При малом числе потоков крупномодульная многопоточность оптимальна.
3. Синхронная многопоточность. Каждый программный поток может запускать по две команды за такт. В случае простоя с целью обеспечения полной загрузки процессора запускаются команды следующего потока. В случае невозможности запуска команды из-за занятости функционального блока выбирается команда из другого потока.

Поток 1	A1	A2			A3	A4	A5			A6	A7	A8
Поток 2	B1			B2			B3	B4	B5	B6	B7	B8
Поток 3	C1	C2	C3	C4			C5	C6			C7	C8
Мелкомодульная	A1	B1	C1	A2	B2	C2	A3	B3	C3	A4	B4	C4
Крупномодульная	A1	A2		B1		C1	C2	C3	C4		A3	A4

28. Многопоточность в Pentium IV.

Многопоточность позволяет использовать аппаратные ресурсы, которые в противном случае простаивали бы. Intel называет реализацию многопоточности hypertreading.

Основной принцип – выполнение двух программных потоков. Операционная система рассматривает гиперпоточный процессор как двухпроцессорный комплекс с общими КЭШами и основной памятью. Планирование для каждого потока выполняется отдельно.

Средства координации потоков:

7. Дублирование ресурсов.

   Дублируются:

   8. Счетчик команд
   9. Контроллер прерываний
   10. Таблица отображения архитектурных регистров (EAX, EBX и т.д.) на физические регистры.

2. Разделение ресурсов
   1. Жесткое. Разделение слотов между потоками. Легко реализуется, обеспечивает полную независимость потоков, процессор фактически превращается в два. Недостатки: может сложиться

ситуация, когда один из потоков не использует все ресурсы, а другому не хватает.

2. Пороговое. Каждый поток получает требуемые ресурсы, но в допустимых пределах (например, 75%).
3. Полное. Доступ к ресурсам может получить любой программный поток, а обслуживание осуществляется в порядке поступления запросов. Решает проблему неоптимального использования ресурсов, но создает проблему дисбаланса их использования.

29. Классификация параллельных компьютерных систем

Классификация Флинна:

1. SISD (одна инструкция, один поток данных). Классический фон Неймановский компьютер.
2. SIMD (одна инструкция, множественные данные). Один блок управления, выдающий по одной команде, но при этом имеется несколько АЛУ, которые могут обрабатывать несколько наборов данных одновременно.
3. MISD (множественные инструкции, один поток данных). Несколько команд оперируют над одними данными. Возможно конвейер.
4. MIMD (множественные инструкции, множественные данные). Несколько независимых процессов работают как часть большой системы. Мультикомпьютеры и мультипроцессоры.

Расширенная классификация Флинна:

1. SISD (фон Неймановский компьютер)
2. SIMD
    1. Векторный процессор
    2. Матричный процессор
3. MISD - возможно конвейер.
4. MIMD
    1. Мультипроцессоры - совместно используемая память
        1. UMA (Uniform Memory Access) – архитектура с однородным доступом к памяти.
            1. С шинной организацией
            2. Коммутация

2. COMA (Cache Only Memory Access) – архитектура с доступом только к кэш-памяти.
3. NUMA (Non Uniform Memory Access) – архитектура с необнородным доступом к памяти.
    1. CC-NUMA - с поддержанием когерентности (идентичности) кэш-памяти
    2. NC-NUMA - без поддержания когерентности (идентичности) кэш-памяти

2. Мультикомпьютеры - передача сообщений
    1. MPP процессоры с массовым параллелизмом, дорогостоящие компьютеры, которые состоят из большого количества процессоров, связанных высокоскоростной коммуникационной сетью.
        1. Решётка
        2. Гиперкуб
    2. COW. Сети рабочих станций и кластеры рабочих станций, связанные при помощи уже имеющихся соединений.

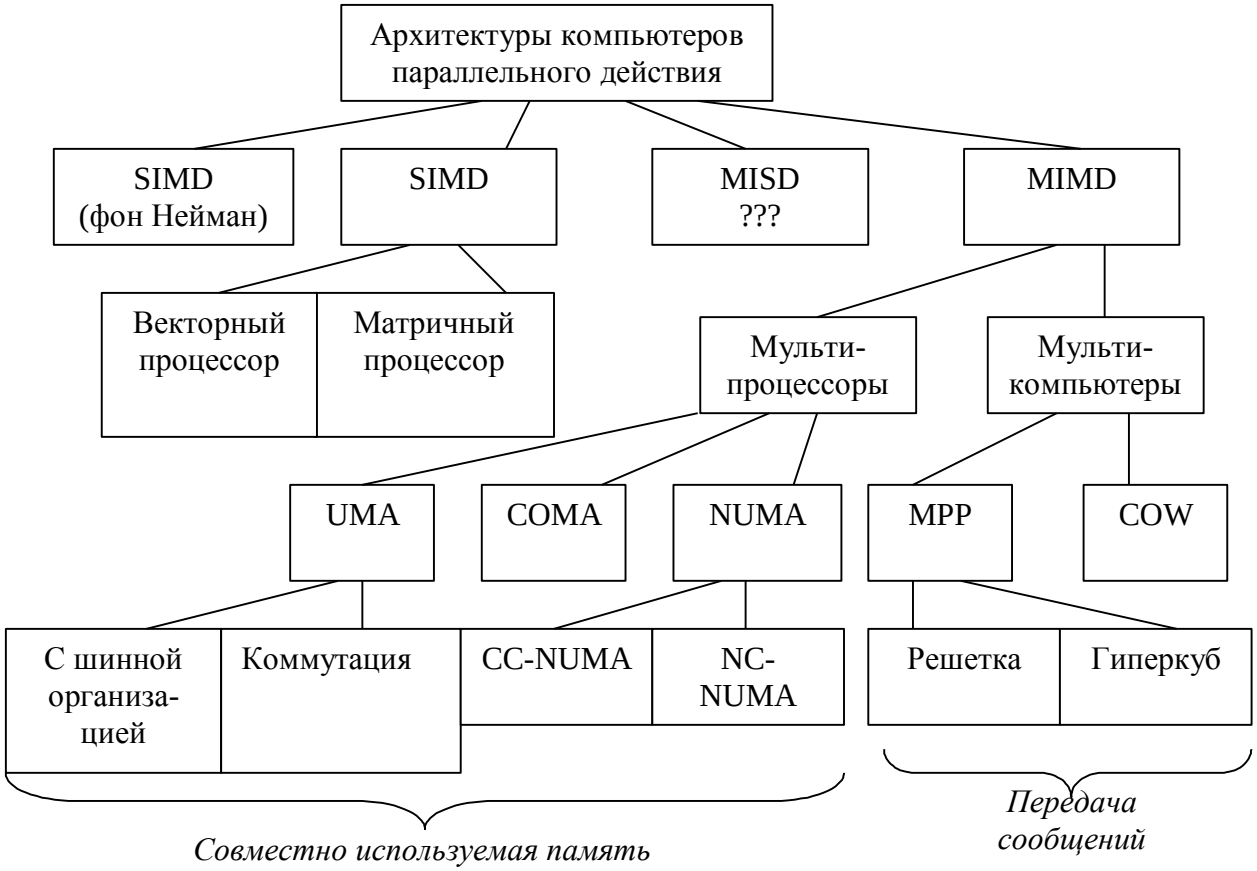


Рис. 6.14

30. Семантика памяти. Отслеживание изменений данных в кэш-памяти. Протокол MESI

Семантика памяти - контракт между программным обеспечением и аппаратным обеспечением памяти. Эти правила называются моделями согласованности (моделями состоятельности).

Виды согласованности:

- Строгая. При любом считывании из адреса X всегда возвращается результат самой последней записи в X. Неэффективна, так как может быть реализована только следующим образом: один модуль памяти, без кэш-памяти, без дублирования данных.
- По последовательности. При наличии нескольких запросов на чтение и запись аппаратное обеспечение определяет порядок всех запросов, но все процессоры наблюдают одну и ту последовательность запросов.
- Процессорная. Реализуется проще, чем согласованность по последовательности.

  Свойства:

  1. Все процессоры воспринимают записи любого процессора в том порядке, в котором они начинаются;
  2. Все процессоры видят записи в слово памяти в том порядке, в котором они происходят.
- Слабая. Записи, произведенные одним процессором, воспринимаются по порядку.

  Для упораядочивания производится синхронизация. Когда она выполняется, все незаконченные записи завершаются, а новые не могут начаться пока не будут завершены все начатые и не будет проведена синхронизация. Операции синхронизации согласованы по последовательности. Время разделяется на последовательные периоды, разграниченные моментами синхронизации. Внутри периодов в последовательность не упорядочена, но в момент синхронизации происходит "выравнивание".
- Свободная. Более эффективна, чем слабая, так как не требует завершения записи при синхронизации. Если процесс выходит за пределы критической области, требуется чтобы все записи были завершены до того, как процесс снова войдет в эту критическую область.

Синхронизация разделена на две операции:

1. Acquire. Считывание или запись общей переменной - получение монопольного доступа. Производится проверка, все ли начатые

операции release завершены. Если нет, то операция acquire будут задержана.

2. Release. Освобождение переменной, не требует завершения незаконченных записей, однако не может быть завершена, пока они не закончатся. Новые операции памяти могут быть начаты сразу же.

Отслеживание изменений данных в кэш-памяти

Непротиворечивость кэшей - соблюдение синхронизации кэшей при изменении одним из процессоров общего значения в кэше. Алгоритм, разрешающий эту проблему - протокол когерентности кэширования. Они различаются деталями, но все не допускают одновременного появления разных вариантов одной и той же строки в разных блоках кэш-памяти.

Во всех решениях контроллер кэш-памяти разрабатывается таким образом, чтобы кэш-память могла перехватить запросы на шине от других процессоров и кэш-памятей и предпринимала определенные действия в необходимых случаях. Такие устройства называются кэш-памятью с отслеживанием.

Протокол MESI

Популярный протокол когерентности кэширования.

Каждый элемент кэш-памяти находится в одном из четырех состояний:

1. Invalid – элемент кэш-памяти содержит недействительные данные.
2. Shared – несколько кэшей могут содержать данную строку, основная память обновлена.
3. Exclusive – никакой другой кэш не содержит эту строку, основная память обновлена.
4. Modified – элемент действителен, основная память недействительна, копий элемента не существует.

Работа алгоритма:

1. Загрузка процессора. Все элементы - состояние Invalid.
2. Первое считывание из основной памяти. Строка - состояние Exclusive. Использование строки.
3. Вызов другим процессором. Обе копии - состояние Shared. Чтение - использование строки.
4. Запись в строку с состоянием Shared. Сигнал о недействительности строки, строка у остальных процессоров - состояние Invalid. Строка - состояние Modified, не записывается в основную память.

5. Считывание Modified строки. Сигнал о задержке чтения и запись строки в основную память. Завершение записи - считывание строки другим процессором. Обе копии - состояние Shared.
6. Коллизия записи в Modified строку. Сигнал о задержке записи и запись строки в основную память. Завершение записи - строка в состояние Invalid.

31. Мультикомпьютеры. Коммуникационные сети.

Мультикомпьютер.

0. Процессор - собственная независимая память.
1. Общение программ - механизм передачи сообщений.
2. Узел:
    0. Один или несколько процессоров
    1. ОЗУ (общие для процессоров данного узла)
    2. Диск и другие узлы ввода/вывода
    3. Процессор передачи данных.
3. Процессоры передачи данных связаны с помощью высокоскоростной коммуникационной сети.

Коммуникационные сети

Топология сети межсоединений - расположение каналов связи и коммутаторов.

Изображается в виде графа:

1. Дуги - каналы связи
2. Вершины (узлы) - коммутаторы
3. Степень вершины - коэффициент разветвления
4. Расстояние между вершинами – число ребер (дуг) которые нужно пройти, чтобы попасть из одной в другую
5. Диаметр графа – максимальное расстояние между двумя вершинами. Определяет самую большую задержку при передаче пакетов
6. Пропускная способность сети - количество данных, которое она может передать в секунду
7. Бисекционная пропускная способность – минимальная из всех возможных (минимальный разрез сети). Самая важная характеристика сети

Сети межсоединений можно характеризовать по их размерности. Размерность определяется по числу возможных вариантов перехода:

1. 1 - нульмерная
2. 2 - одномерная
3. 4 - двухмерная

   И так далее.

32. Метрика аппаратного обеспечения.

Параметры межсоединений:

- Время ожидания

  Полное время ожидания. время от отправки процессором пакета до получиления ответа.

  1. Сеть с коммутацией каналов. Время установки соединения + время передачи. Установка соединения - резервация канала и получение подтверждения. Передача происходит за время $T = T_s + p/b$, где T_s - общее время установки, p - количество бит, b - пропускная способность. Если схема дуплексная то передача произойдёт за $T = T_s + 2p/b$.
  2. Пакетная коммутация. Открытие канала вещания и посылка пакетов. Время передачи до первого коммутатора $\mathrm{T} = \mathrm{T_a} + \mathrm{p/b}$. Прохождение через коммутатор - задержка $\mathrm{T_d}$ (время обработки и задержки в очереди). Полное время прохождения пакета $\mathrm{T} = \mathrm{T_a} + \mathrm{n(p/b + T_d)} + \mathrm{p/b}$, где n - число коммутаторов, последнее слагаемое - копирование в пункт назначения.
  3. Коммутация без буферизации пакетов приближается к $\mathrm{T} = \mathrm{T_a} + \mathrm{p/b}$ (нет пробных пакетов и нет задержки).

- Пропускная способность.
  1. Бисекционная. Минимальная из всех возможных.
  2. Суммарная. Сумма пропускных способностей всех каналов связи. Максимальное число бит, которые можно передать сразу.
  3. Средняя. Суммарная делённая на количество процессоров.

Теоретическая пропускная способность никогда не достигается. При разработке сетей сначала добиваются малого времени ожидания ответа, так как ширину канала можно увеличить позднее.

33. Метрика программного обеспечения.

Для пользователя компьютера параллельного действия ключевым является вопрос о коэффициенте ускорения, который показывает,

насколько быстрее работает программа в n-процессорной системе по сравнению с однопроцессорной. Идеальное повышение скорости – использование n процессоров заставляет программу работать в n раз быстрее. На практике все на так.

Пусть T_1 – время решения задачи на однопроцессорной системе, Tn – время решения той же задачи на n-процессорной системе.

Пусть W = Wck + Wpr, где W – общее число операций, Wpr – число операций, которые можно выполнить параллельно, Wск – число скалярных (не распараллеливаемых операций). Обозначим через t время выполнения одной операции; тогда получаем выражения для закона Амдала:

$$r = \frac{Wt}{\left(W_{ck} + \frac{W_{pr}}{n}\right)t} = \frac{1}{a + \frac{1-a}{n}} \xrightarrow[n\to\infty]{} \frac{1}{a}, \qquad W = W_{ck} + W_{pr},$$ где

W - общее число операций,

W_{pr} - число параллельных операций,

W_{ck} - число скалярных операций,

t - время выполнения одной операции,

$a = W_{ck}/W$ - удельный вес скалярных операций.

Выводы:

1. ускорение вычислений зависит как от потенциального параллелизма задачи (величина 1-a), так и от параметров аппаратуры (число процессоров n);
2. предельное ускорение определяется свойствами задачи (величина a).