Высокопроизводительные системы

Судаков А.А.

“Параллельные и распределенные вычисления” Лекция 9

План

- Типы высокопроизводительных систем
- Однопроцессорные системы
- SMP
- Векторные процессоры
- Массивно-параллельные системы
- NUMA системы
- Кластеры
- Метакомпьютеры
- Основные характеристики

Литература

- http://www.intel.com
- http://www.parallel.ru
- Когерентность кэша http://www.update.uu.se/~gus/Misc/Compositions/cache_coherence.html

Высокопроизводительные системы

- Основная цель – получение высокой скорости вычислений
  - Обработка больших объемов данных
  - Выполнение большого количества операций
- Суперкомпьютер – компьютер, который по своим возможностям значительно превосходит широко распространенные вычислительные системы

Пути повышения производительности

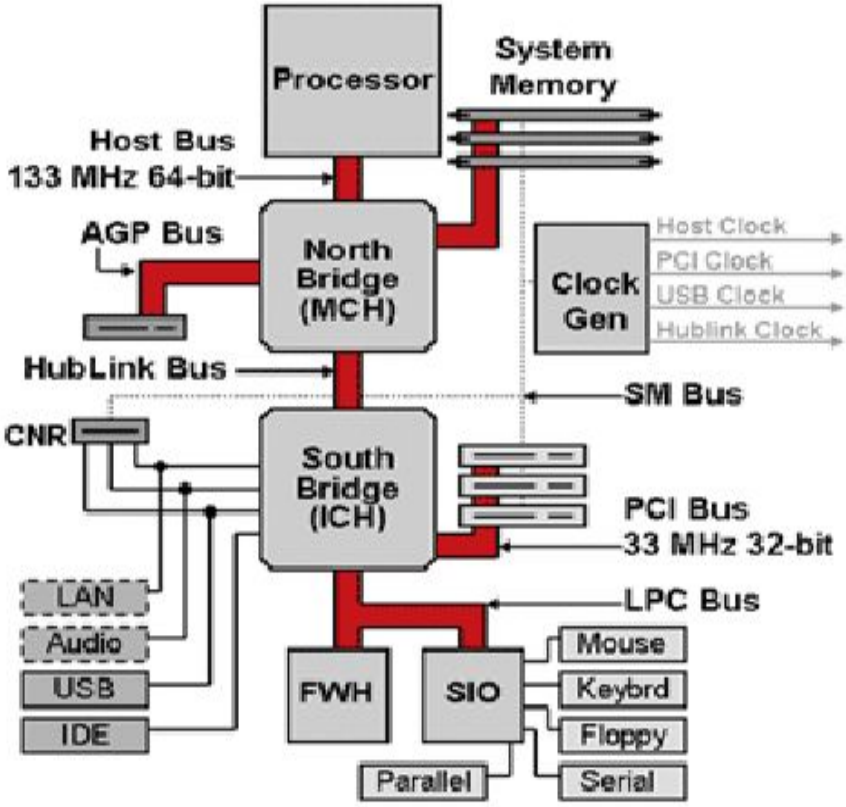
- Использование новых физических эффектов
- Увеличение тактовой частоты
- Параллельные вычисления
  - Использование большого количества вычислительных элементов на которых одновременно решаются различные части задачи

Типы высокопроизводительных систем

- Однопроцессорные системы (UP)
- Симметричные многопроцессорные системы (SMP)
- Векторно-конвейерные системы (PVP)
- NUMA системы
- Вычислительные кластеры
- Метакомпьютры

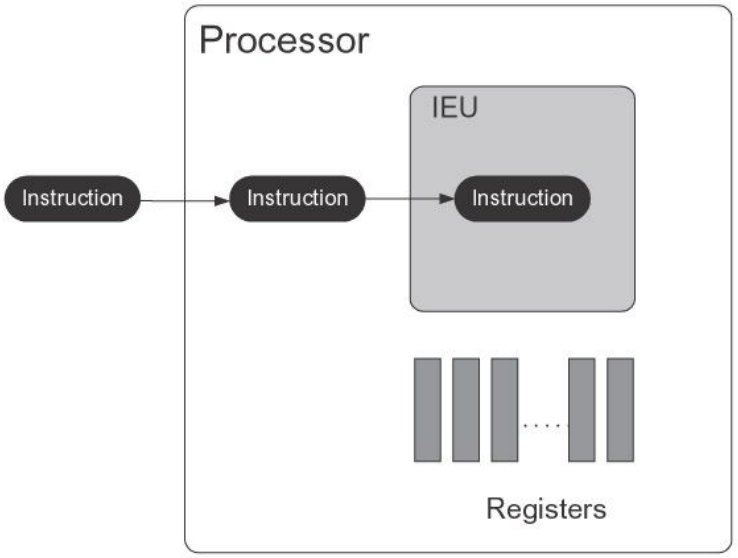
Однопроцессорные ЭВМ

- Один процессор
- Общая шина
- Параллелизм на уровне инструкций за счет особенностей устройства процессора



Скалярная обработка

- Один поток инструкций
- Одно функциональное устройство
- Все инструкции обрабатываются последовательно
- Каждая инструкция работает со скалярными данными
- Скалярные данные – не массив
- Intel 8086



Кэширование оперативной памяти

- Кэш – память с быстрым доступом (быстрее, чем из оперативной памяти)
  - Кэш инструкций – память для инструкций (8-64K)
  - Кэш данных – память (8-64K)
- Кэш первого, второго, третьего уровня
  - L1 - 8-64K
  - L2 128K-2M
  - L3 4M
- Процессор из кэша может считывать и записывать данные порциями (строка кэша) обычно 128 байт
- Процессор может выполнять действия с кэшем не обращаясь к шине
  - Возможна параллельная обработка данных процессором и передача по шине

Предварительная выборка

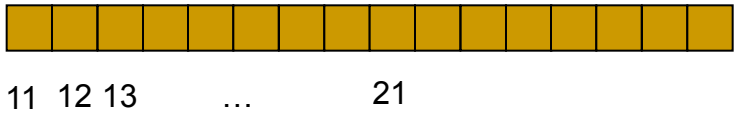
- Если данные и инструкции находятся в кэше, то операции с ними выполняются очень быстро
- В процессорах используется различные методы, чтобы увеличить вероятность того, что необходимые данные находятся в кэше – prefetch
  - Предварительная загрузка
  - Предсказание ветвлений
- Для максимального быстродействия
  - Данные/код должны считываться/записываться порциями кратными размеру сроки кэша
  - Данные/код должны помещаться в кэш и кэш должен быть загружен по максимому

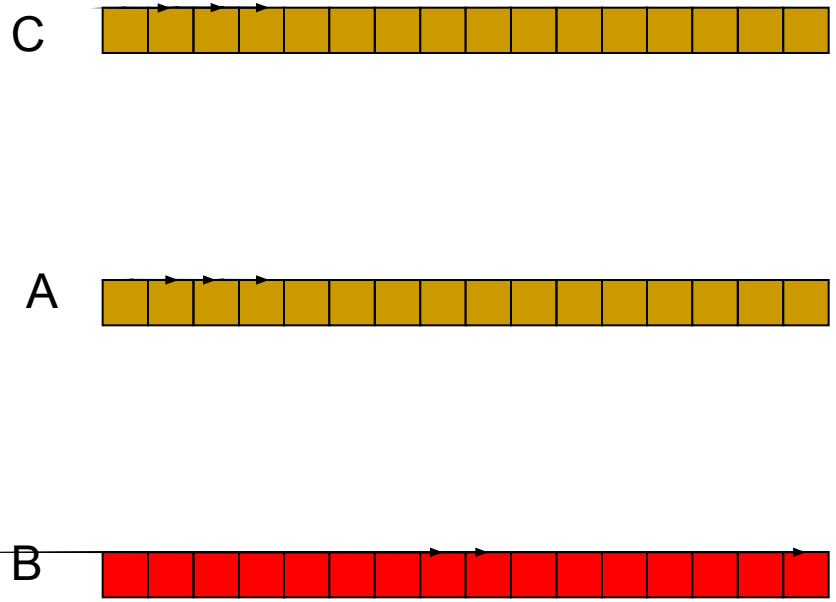
Пример замедления за счет не попадания в кэш

- Умножение двух матриц
- C=AB
- $C_{ij}=A_{ik}B_{kj}$

```
for(i=0; i<n; i++)
    for(j=0; j<n; j++)
      for(k=0; k<n; k++)
            c[i][j]+=a[i][k]*b[k][j]
```

- Данные в памяти хранятся построчно
- Соседние элементы сроки находятся в соседних адресах памяти
- Соседние элементы столбца находятся «далеко» друг от друга
- Необходимые элементы матриц A и C будут в кэше
- Необходимый элемент матрицы B там будут отсутствовать



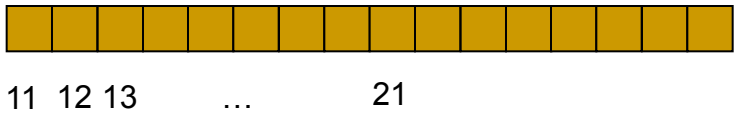


Пример ускорения за счет кэширования

- Умножение двух матриц
- C=AB
- $C_{ij}=A_{ik}B_{kj}$

```
for(i=0; i<n; i++)
    for(k=0; k<n; k++)
      for(j=0; j<n; j++)
            c[i][j]+=a[i][k]*b[k][j]
```

- Данные в памяти хранятся построчно
- Соседние элементы сроки находятся в соседних адресах памяти
- Если k и j поменять местами, то все необходимые элементы будут последовательно загружаться в кэш за счет prefetch

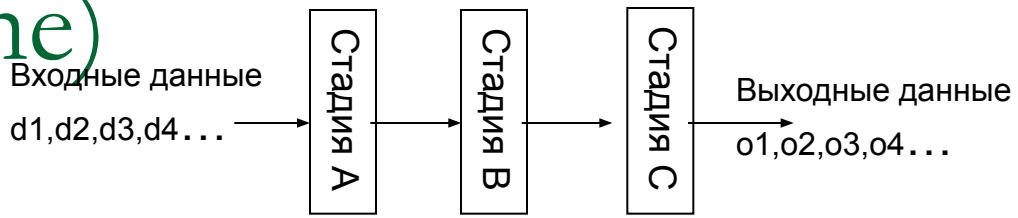


C

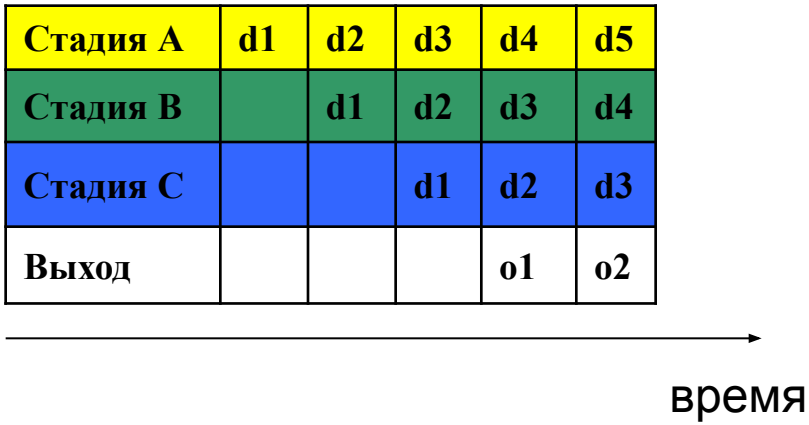
A

B

Конвейер (pipeline)

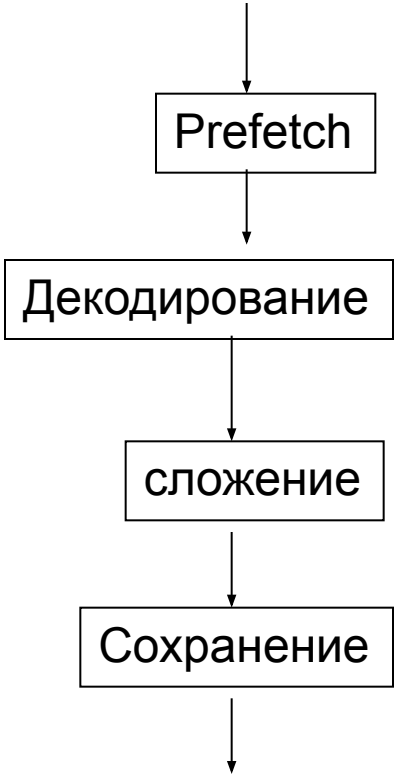


- Одну сложную операцию можно разбить на несколько простых
- Если сложную операцию необходимо выполнить с большим количеством входных данных, то возможна параллельная обработка
- Первый результат получается за время сложной операции, а все остальные за время самой медленной стадии



Пример конвейерной обработки

- В платформе P5
  - 4 стадии конвейера
- IEEE сложение 6 стадий
  - Сравнение порядков
  - Нормализация
  - Сложение мантисс
  - Нормализация результата
  - Проверка на исключительную ситуацию
  - Округление
- При большом количестве входных данных можно получить ускорение в 4-10 раз



Пример ускорения за счет конвейерной обработки loop unrolling

- Цикл - разные операции

```
for (i=0; i<n; i++)
    s+=a[i];
```

  - Проверка i
  - Получение a[i]
  - Сложение s+=a[i]
  - Увеличение i
- Не очень эффективно

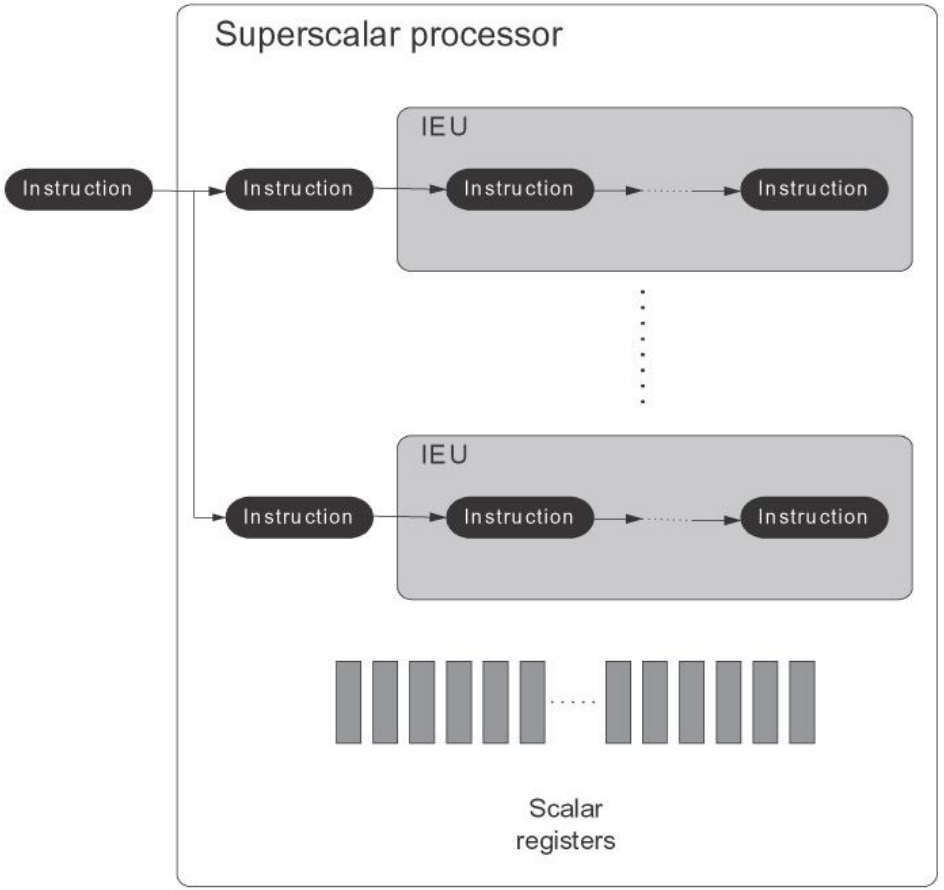
- Развертывание циклов (loop unrolling)
- Вместо цикла выполняем последовательность

```
s+=a[1];
s+=a[2];
s+=a[3];
s+=a[4];
s+=a[5];
…
s+=a[n];
```

- Получаем значительное ускорение за счет конвейерной обработки

Суперскалярная обработка

- Один поток инструкций
- Несколько функциональных устройств
- Одновременно считывается несколько инструкций
- Независимые инструкции обрабатываются одновременно на нескольких функциональных устройствах
- Все функциональные устройства работают со скалярными данными
- Если инструкции зависимы друг от друга, то одно из функциональных устройств простаивает

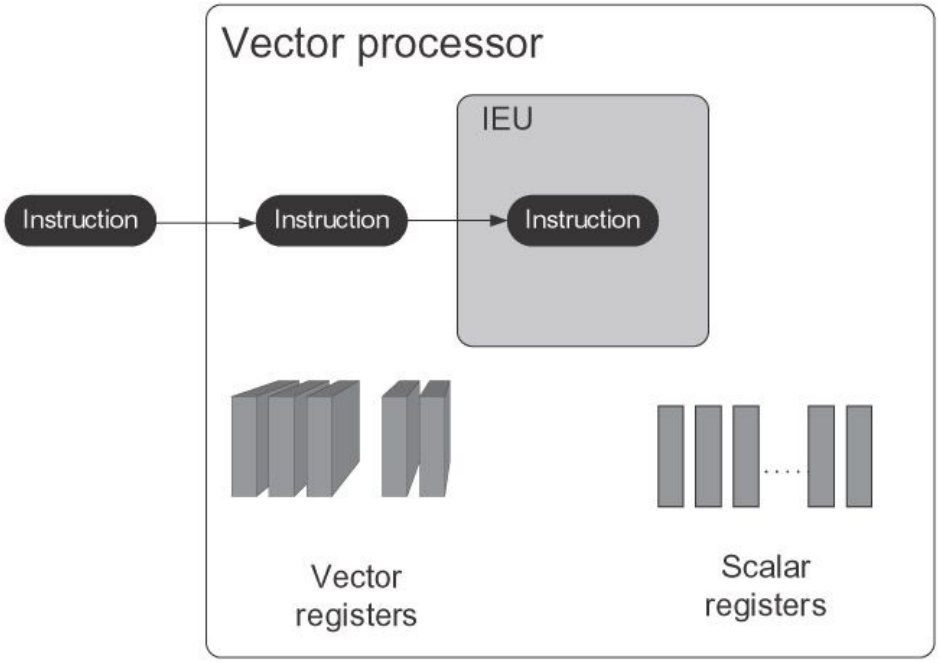


Пример суперскалярной обработки

- (a+jb)*(c+jd)=ac-bd+j(bc+ad)
- Операции (10 последовательных операций)
  - Загрузка a в регистр
  - Загрузка b в регистр
  - Загрузка c в регистр
  - Загрузка d в регистр
  - ac
  - bd
  - ac-bd
  - bc
  - ad
  - bc+ad
- Параллелизм на уровне инструкций
- Суперскалярная обработка на 2-х функциональных устройствах (5 последовательных операций)
  - Загрузка a, загрузка b
  - Загрузка c, загрузка d
  - ac одновременно с bd
  - ad одновременно с bc
  - ac-bd одновременно с bc+ad

Векторная обработка

- Функциональное устройство обрабатывает последовательно по одной инструкции
- Инструкции могут быть векторными – одна инструкция сразу обрабатывает массив данных
- Массивы хранятся в векторных регистрах
- Операции с двумя векторными регистрами можно выполнить с помощью одной команды



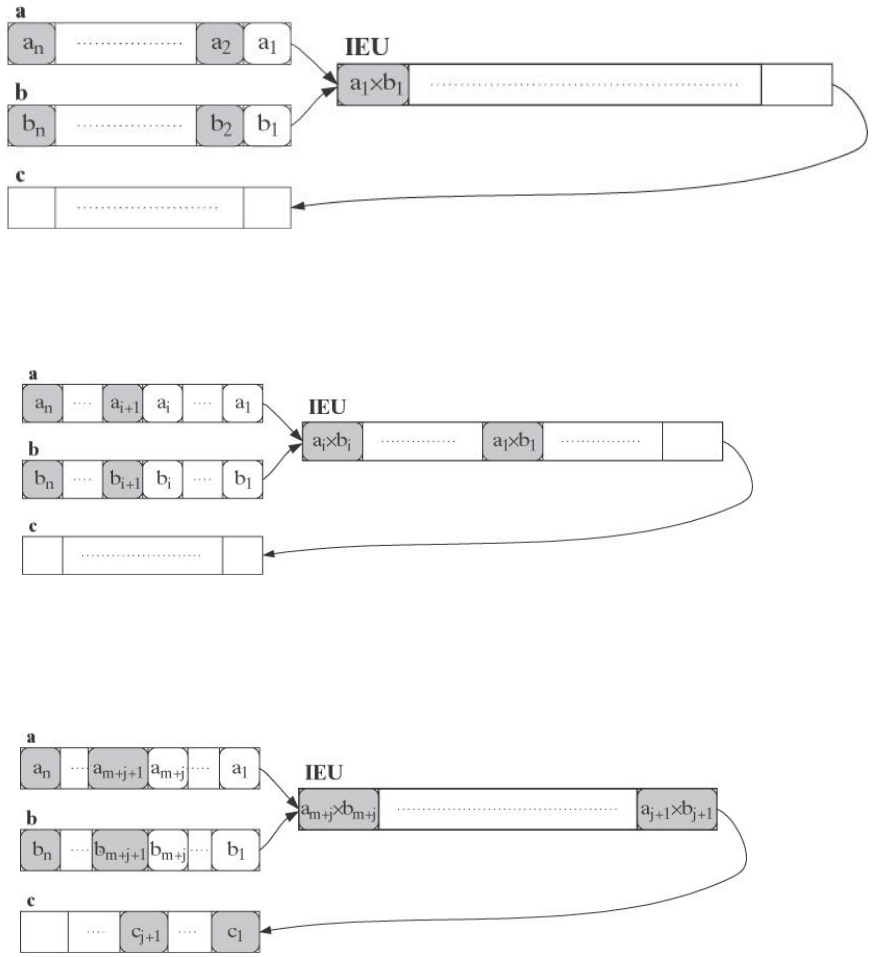
Пример векторной обработки

- Сложение двух массивов
  - a[i]+b[i]=c[i]
- Последовательный вариант
  - c[1]=a[1]+b[1]
  - c[2]=a[2]+b[2]
  - ….
  - c[N]=a[N]+b[N]
- Порядка 4N операций
  - Загрузка 2N
  - Сложение N
  - Присвоение N

- Векторная обработка
  - Загрузка a[] в векторный регистр 1
  - Загрузка b[] в векторный регистр 2
  - Сложение a[]+b[] одной машинной инструкцией
  - Присвоение c[]=a[]+b[] одной машинной инструкцией
- Порядка 2N+2 операции
  - Загрузка 2N
  - Сложение 1
  - Присвоение 1

Векторно-конвейерная обработка

- Конвейер очень эффективен для векторных операций
- c[]=a[]+b[]
- Процессор всегда работает эффективно для длинных последовательностей одинаковых операций

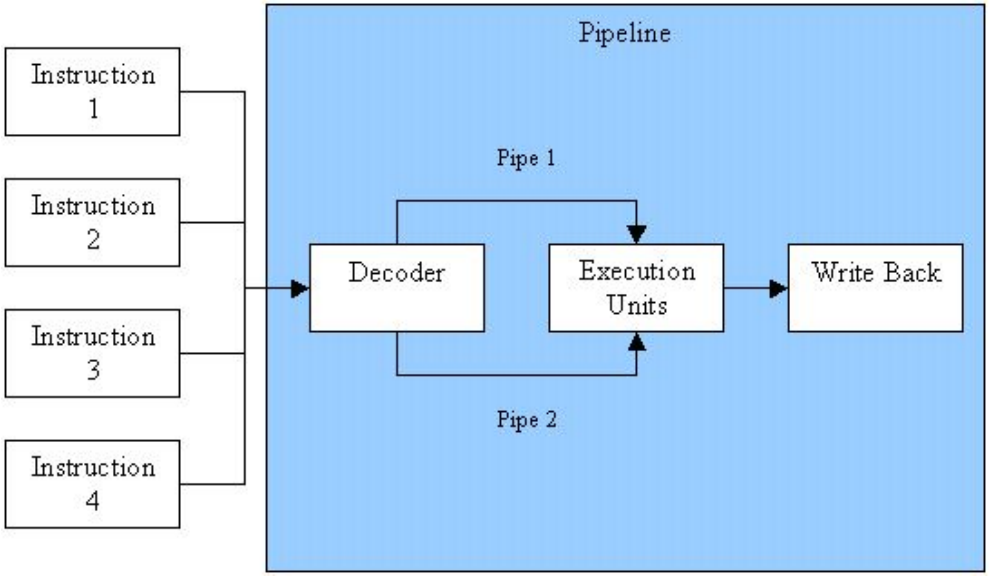


Современные микропроцессоры

- Используются все рассмотренные подходы в комбинации
- Каждый тип процессора имеет свои особенности и требует своего подхода при программировании для получения максимальной производительности

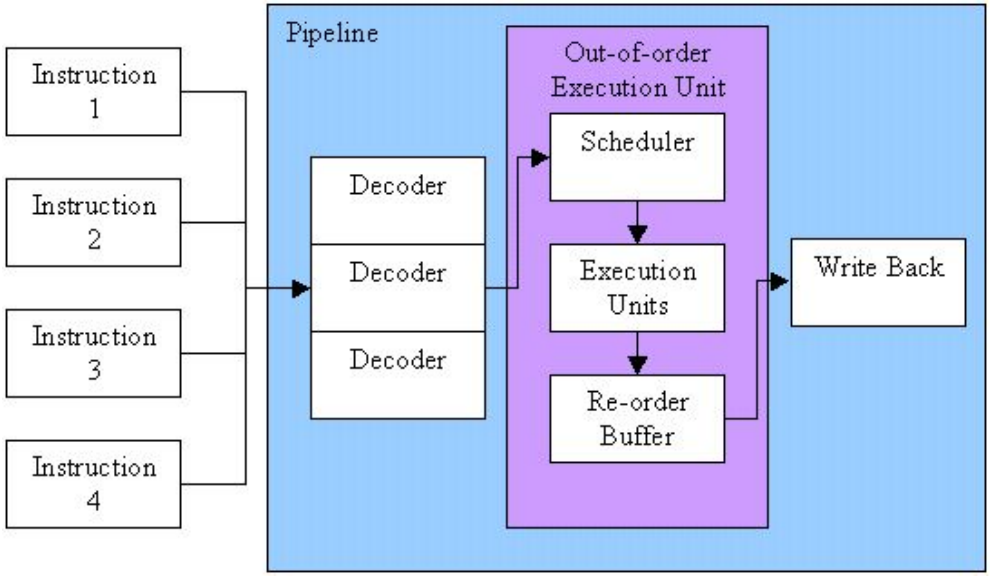
Intel Pentium

- Суперскаляр
  - 2 целочисленных конвейера
  - 1 с плавающей точкой
- Конвейер
  - 5 стадий
    - pre-fetch (PF), Decode stage 1(D1), Decode stage 2 (D2), Execute (E), and Write back (WB).
- 2 целочисленных операции за такт



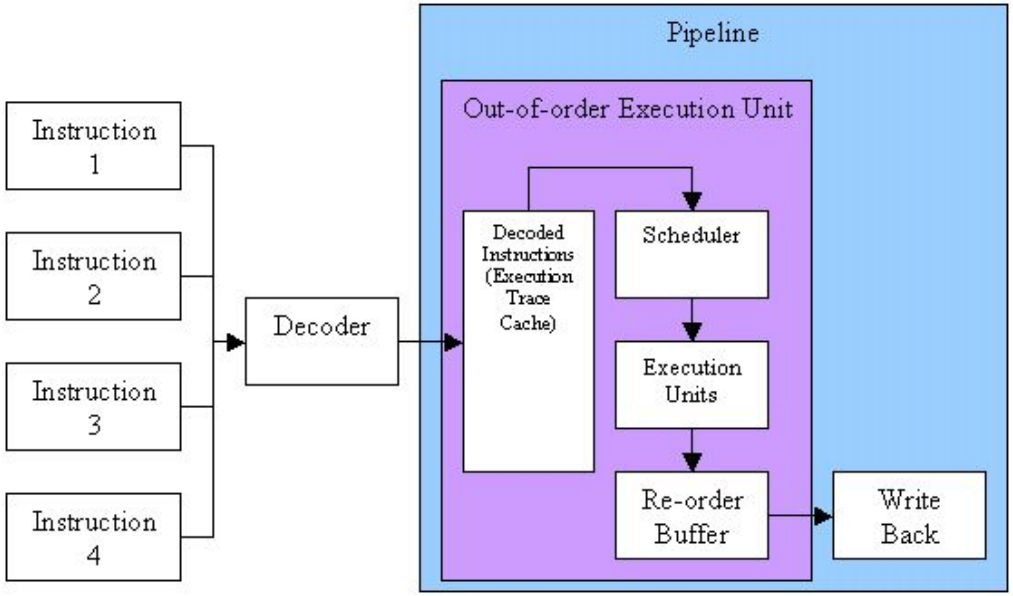
Intel Pentium III

- Reordering – перестановка порядка выполнения
- SSE – streaming SIMD extension
- 1-2 FPU операции



Intel Pentium 4

- Кэш декодированных инструкций
- Увеличение количества стадий конвейера да 20
- SSE2

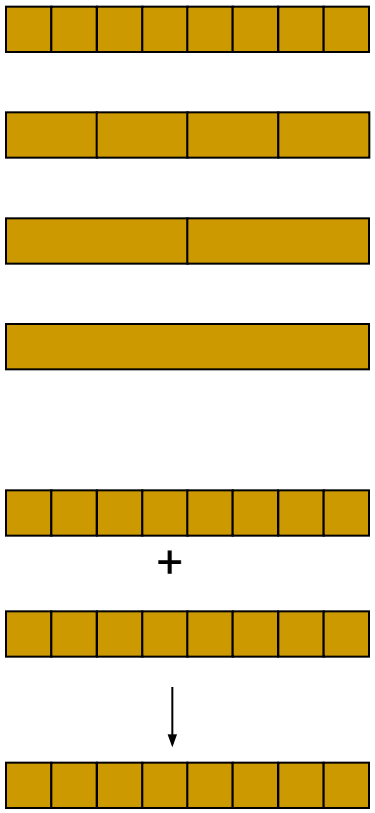


MMX, SSE, SSE2, SSE3

- MMX – multimedia extension
  - 8 64-х битных векторных регистра (4 элемента вектора) для операций с целыми числами
  - Нельзя использовать совместно с FPU
- SSE – streaming SIMD extension
  - 8 128-и битных векторных регистра (4 элемента вектора) для операций с плавающей точкой одинарной точности
  - Можно использовать совместно с MMX
- SSE2
  - Добавлены инструкции для интерпретации каждого регистра как 2 значения типа double
- SSE3
  - Добавлены операции с комплексными числами и др.

Как работают SIMD расширения

- Данные записываются в векторные регистры
- Вызывается одна инструкция для выполнения операции с этими регистрами
- В зависимости от команды данные интерпретируются как байты, слова, float, double
- Операция выполняет с помощью векторно-конвейерной обработки



Упаковка байтов

Упаковка слов

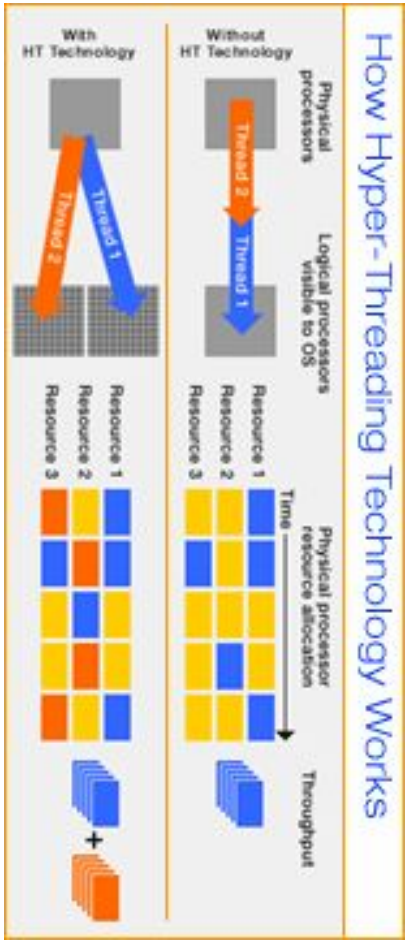
Упаковка float

Упаковка double

Параллельное Сложение одной инструкцией

Hyperthreading

- В суперскалярных процессорах с большим количеством стадий конвейера (Pentium 4, Xeon) на практике используется только 15% процентов функциональных элементов, остальные простаивают
- Для того, чтобы загрузить неиспользуемые стадии конвейеров было предложено на тех же функциональных элементах выполнять два потока команд
- Эта технология называется Hyperthreading или Jackson Technology
- Такие процессоры в системе видятся как два



Когда Hyperthreading эффективен?

- Эффективен
  - Более эффективная обработка прерываний
  - Более эффективный ввод/вывод (дисковый, сетевой)
    - Подкачка страниц (работа в условиях нехватки памяти)
    - Запуск приложений
    - Отображение на память
  - Более эффективное одновременное выполнение различных задач
  - MIMD, MISD многопоточность
  - Более эффективное использование кэша
- Не эффективен
  - Одновременное выполнение одинаковых задач
  - SIMD многопоточность (математика)
  - При малом объеме кэша

RISC процессоры

- RISC – reduced instruction set
  - Уменьшение количества сложных инструкций
  - За счет этого увеличение производительности
  - Особенности
    - Увеличено количество регистров
    - Увеличение количества конвейеров и простых стадий конвейеров
    - Операции с памятью только чтение/запись
    - Вся ответственность за оптимизацию ложится на компилятор
    - Сложные операции выполняются медленно, но их мало
- CISC – complex instruction set
  - Увеличение количества сложных инструкций
  - Особенности
    - Увеличение количества сложных функциональных устройств
    - Больше инструкций для работы с памятью
    - Меньше регистров
    - Повышение «интеллектуальности» процессора
    - Компилятор должен учитывать особенности процессора
    - Сложные операции выполняются быстрее, но их меньше

Характеристики UP процессоров

- Тактовая частота
- Количество операций за один такт
  - Обычно 1-3
- Производительность – количество операций в секунду
  - Пиковая: теоретически максимально-возможная
  - Измеренная: измеренная определенным тестом

Единицы производительности

- MIPS - миллион инструкций в секунду
  - Характеристика для сравнения производительности разных процессоров
- FLOPS – количество операций с плавающей точкой в секунду
  - Характеристика производительность при решении прикладных задач
  - MFLOPS, GFLOPS, TFLOPS, PFLOPS

Пиковая производительность

- Максимально теоретически возможная
- Наиболее точная характеристика процессора

$$R = \sum_{i=1}^{p} f_i n_i ,$$

Где R — теоретическая пиковая производительность, p — количество процессоров в системе, i — нумерует все процессоры в системе, f_i — тактовая частота процессора с номером i, n_i — количество операций с плавающей точкой, которые процессор с номером i может выполнить за один машинный такт.

Пример расчета пиковой производительности

- Intel Xeon
  - 2 операции с числами с плавающей точкой двойной точности (double) за одни такт
  - Тактовая частота 2.4 ГГц
  - R=2*2.4 = 4.8 GFLOPS
- Пиковой производительности достичь тяжело

Измеренная производительность

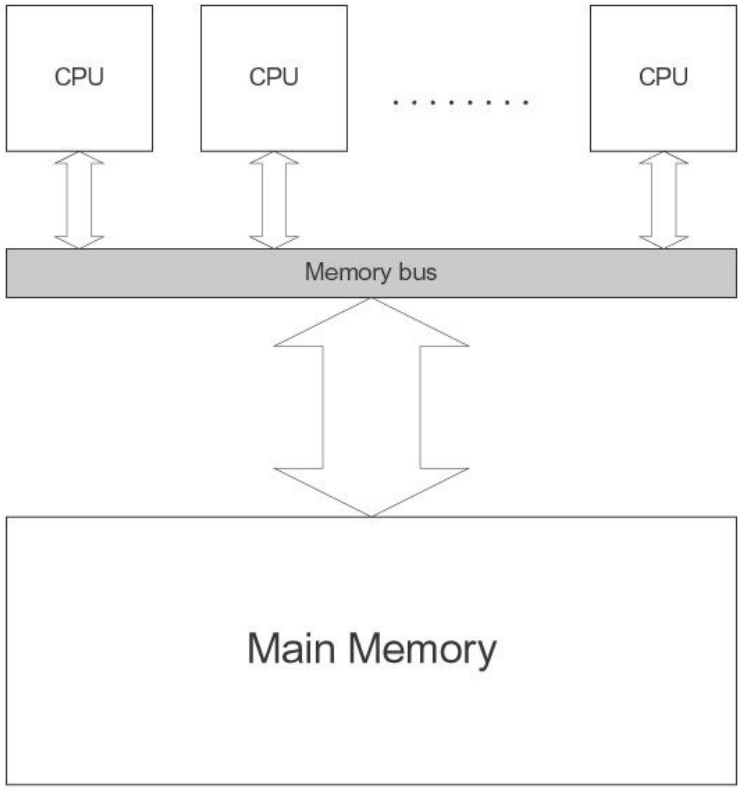
- Измеренная производительность зависит от задачи
  - Процессор эффективен только для определенных последовательностей команд
  - Для некоторых задач будет эффективнее одна система, а для некоторых другая
- Стандартные тесты
  - hpl
  - NAS

Как использовать возможности параллельной обработки UP систем

- Машинные инструкции необходимо выполнять в оптимальной последовательности
- Компилятор должен учитывать особенности процессора

Симметричные многопроцессорные системы (SMP)

- Несколько полностью эквивалентных процессоров, которые работают с общей памятью и подключены к общей шине

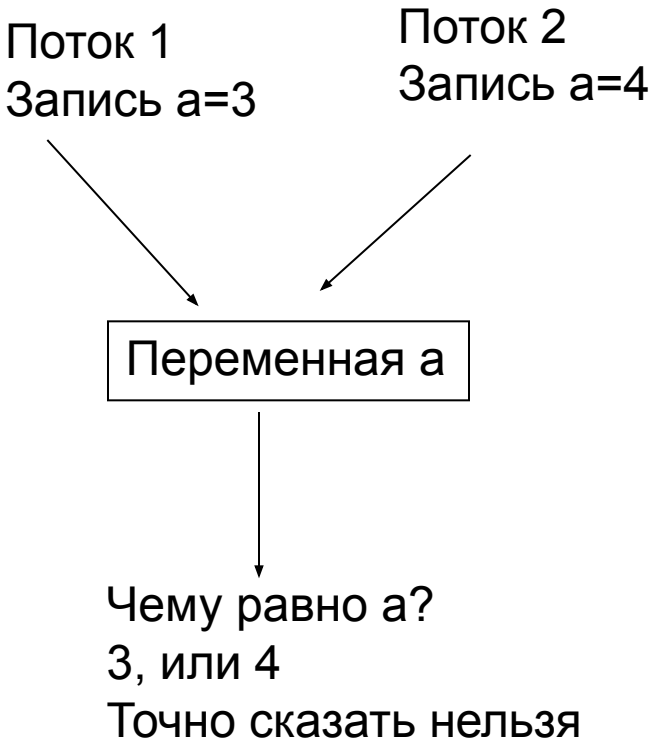


Особенности

- Преимущества
  - Параллелизм на уровне процедур
  - Многопоточные программы
  - Быстрая обработка прерываний
  - Повышение производительности ввода/вывода
- Сложности
  - Обеспечение синхронизации при обращениях к общей памяти
  - Обеспечение когерентности кэшей
  - Перестановка операций разными процессорами
  - Увеличение количества существенно последовательных операций, таких как передача по общей шине
  - Снижение эффективности при увеличении количества процессоров

Синхронизация при обращениях к общей памяти

- При обращениях к общей данным возможны конфликты чтения-записи
  - Результат двух одновременных операций чтения и записи или двух одновременных операций записи не определен
  - Две операции чтения могут выполняться одновременно



Синхронизация

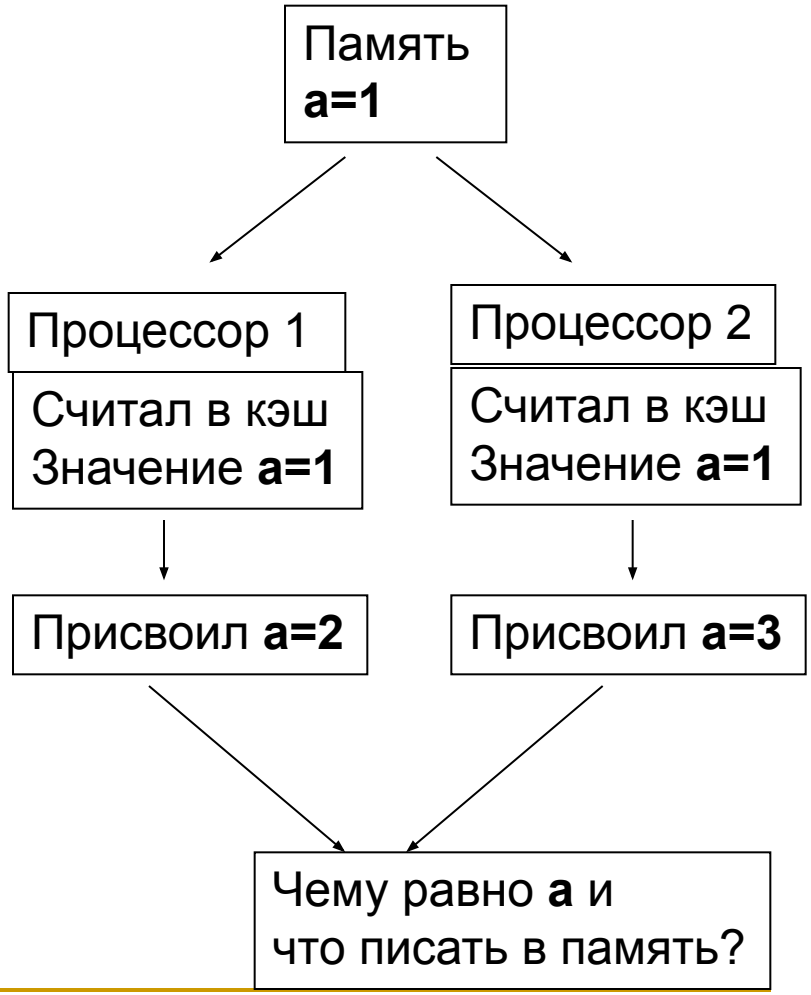
- Атомарные операции
  - Неделимые операции – если выполняется атомарная операция с некоторыми данными, то другая атомарная операция с этими данными одновременно с первой выполняться не может
  - Осуществляется специальными машинными инструкциями или блокировками
    - incl i
- Спин-блокировка
  - Программынй код, который периодически в цикле проверяет условие доступности общей переменной
  - Спин-блокировка самый быстрый тип блокировки, но тратит процессорное время

Использование блокировок

- Процессор 1
  - Захват блокировки, ждем если уже захвачена
  - Критический участок: обращение к совместно используемым данным
  - Освобождение блокировки
- Процессор 2
  - Захват блокировки, ждем если уже захвачена
  - Критический участок: обращение к совместно используемым данным
  - Освобождение блокировки

Проблема когерентности кэшей

- Кэш каждого процессора хранит копию данных из общей памяти
- Если эти данные хранятся к кэшах разных процессоров, то каждый процессор изменяет данные кэша
- Какое значение является правильным?
- Необходимо обеспечить непротиворечивость данных в кэшах и памяти (когерентность)

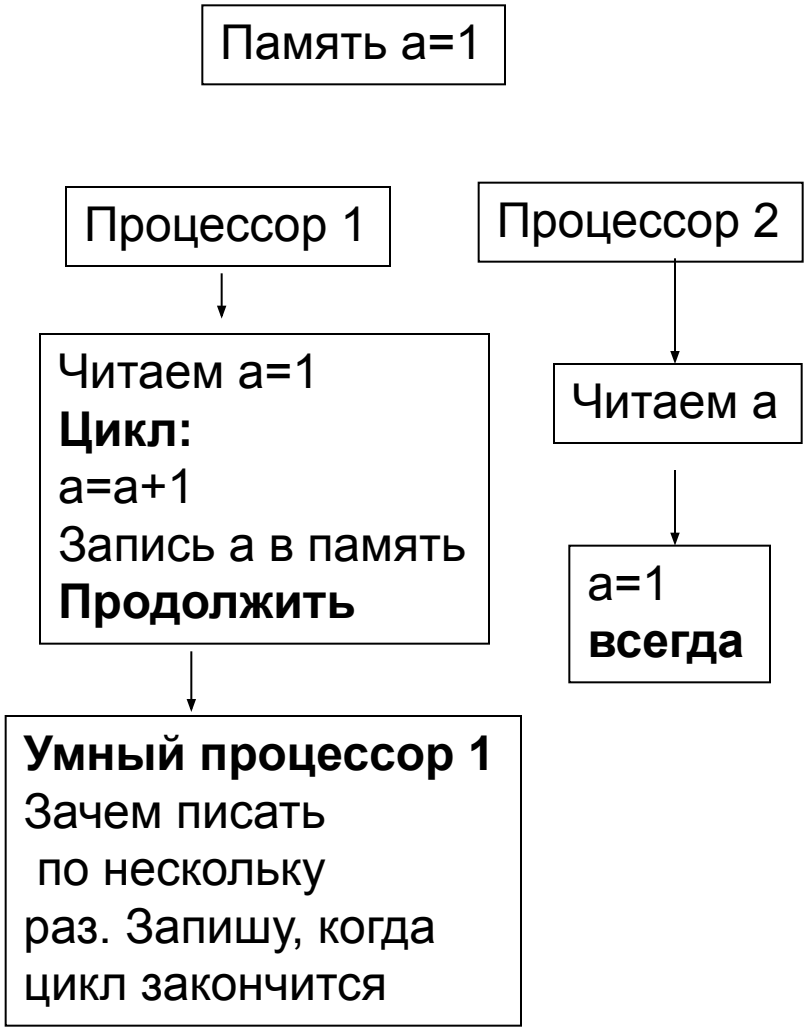


Обеспечение когерентности кэшей

- Общий кэш
  - Несколько процессоров имеют общий кэш и все данные в кэше синхронизированы
- Перехват (мониторинг шины, bus snooping)
  - При любых обращениях к данным по шине система мониторинга определяет какой процессор прочитал какие данные и нет ли соответствующих данных в кэше другого процессора, если есть, то возвращает их
- Каталог кэшей (cache directory)
  - Обращения к памяти выполняются через каталог, в котором хранится информация о том, какие данные хранятся в кэше какого процессора
  - Через каталог выполняется передача информации

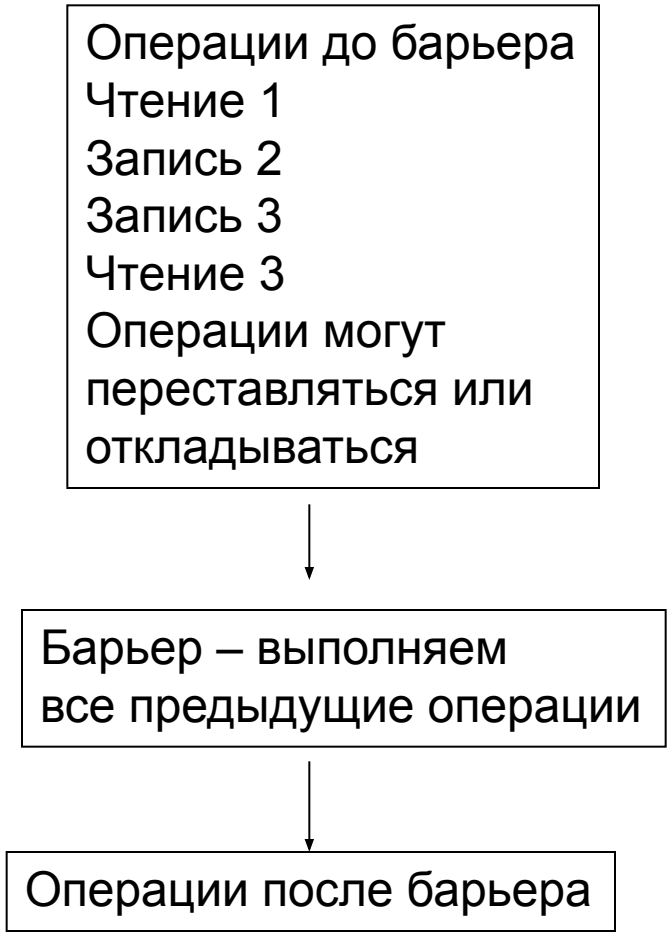
Перестановка порядка выполнения ввода-вывода

- Перестановка порядка
  - Современные процессы могут переставлять местами операции чтения записи или
  - Компилятор может сохранять копии данных в регистрах и тоже переставлять порядок
- Перестановки порядка выполнения разными процессорами может привести к записи в память неправильных данных
- Отсрочка операции записи может привести к неправильной работе программ на разных процессорах
- На однопроцессорной системе такого произойти не может



Барьеры

- Барьер – вид блокировки, который гарантирует, что все операции, которые идут до барьера гарантированно выполнятся после перехода через барьер
- Барьер памяти
  - Барьер чтения
  - Барьер записи
- Обычно используется специальная машинная инструкция



Обеспечение совместимости за счет компилятора

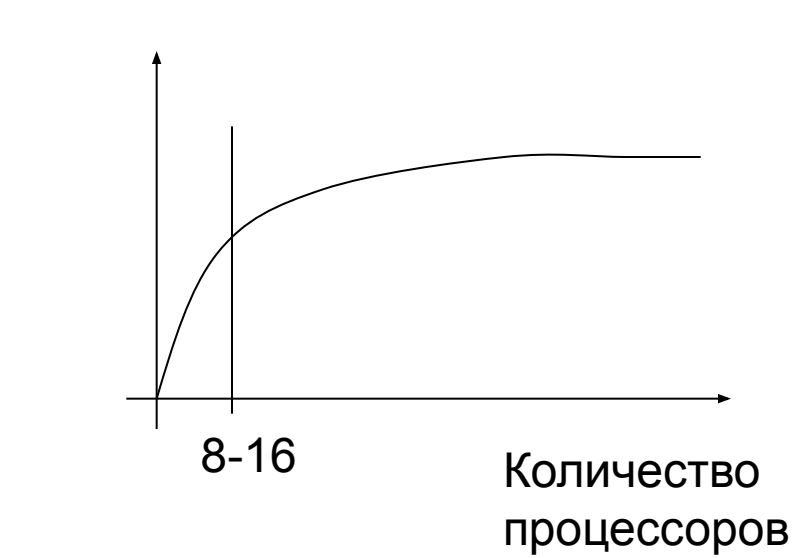
- Компилятору можно указывать инструкции в каких случаях можно переставлять данные, а в каких – нет
- С помощью специально скомпилированного кода можно обеспечить также когерентность кэшей

Работа с общей шиной

- Данные между памятью и процессорами передаются по одной общей шине
  - Захват шины
  - Передача/прием данных
  - Отпускание шины
- Передача по шине – существенно последовательная операция и снижает эффективность параллелизма
- При увеличении количества процессоров увеличивается количество захватов и вероятность того, что шина будет захвачена, когда она нужна – процессоры простаивают

Уменьшение эффективности SMP при увеличении количества процессоров

- Больше процессоров
  - Больше одновременно работающий потоков
  - Более интенсивное взаимодействие между процессорами
  - Больше конфликтов при захвате шины
  - Больше операций для обеспечения когерентности кэша
- Эффективность SMP системы снижается при увеличении количества процессоров



Практические SMP системы

- Обычно до 8 процессоров
- Alpha до 64 процессоров
- Наиболее эффективные 2-4 процессора

Оптимизация

- Существуют компиляторы с автораспараллеливанием
- OpneMP
- pthread
- SysV SHM, SEM

RISC процессоры

- Могут отсутствовать следующие функции
  - Атомарные операции
  - Обеспечение когерентности кэша
  - Барьеры
- Вся (или почти вся) ответственность за синхронизацию ложится на программиста

Вопросы ?