© 2015 МГУ/ВМК/СП

Лекция 0x16

22 апреля

© 2015 МГУ/ВМК/СП

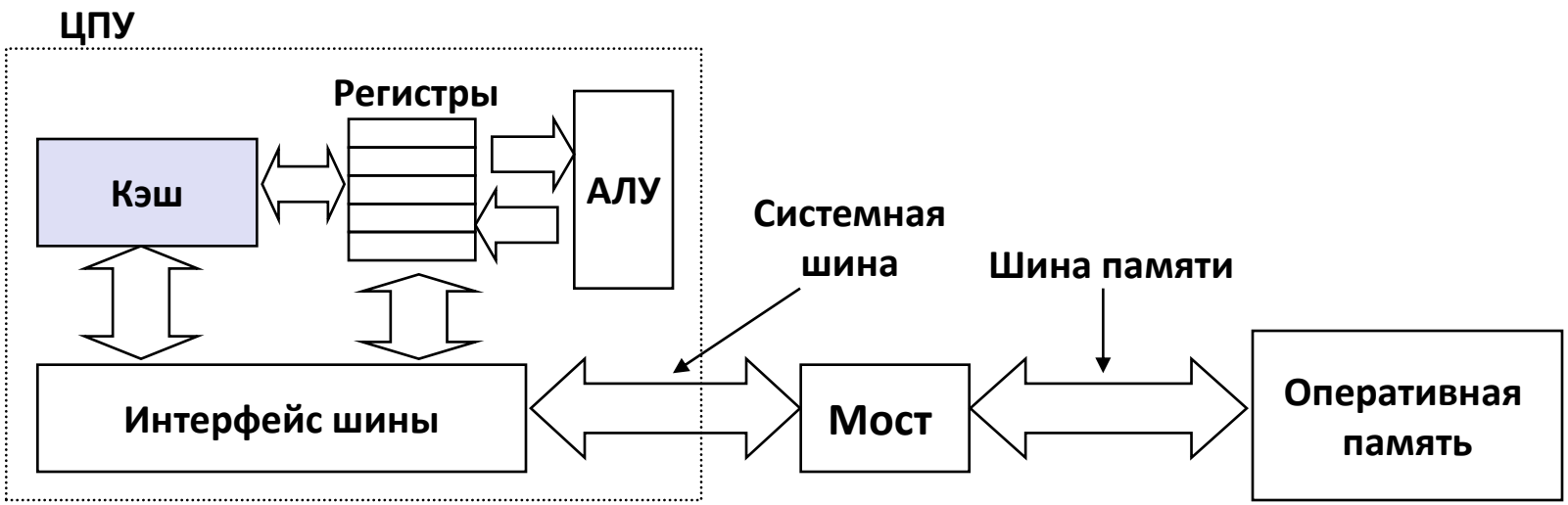
Иерархическая память: промежуточные итоги

- Разрыв в скорости работы между ЦПУ и оперативной памятью продолжает увеличиваться.
- Хорошо написанные программы имеют хорошую локальность.
- Иерархическая организация памяти, основанная на кэшировании, позволяет эффективно бороться с разрывом в скорости с помощью локальности.

© 2015 МГУ/ВМК/СП

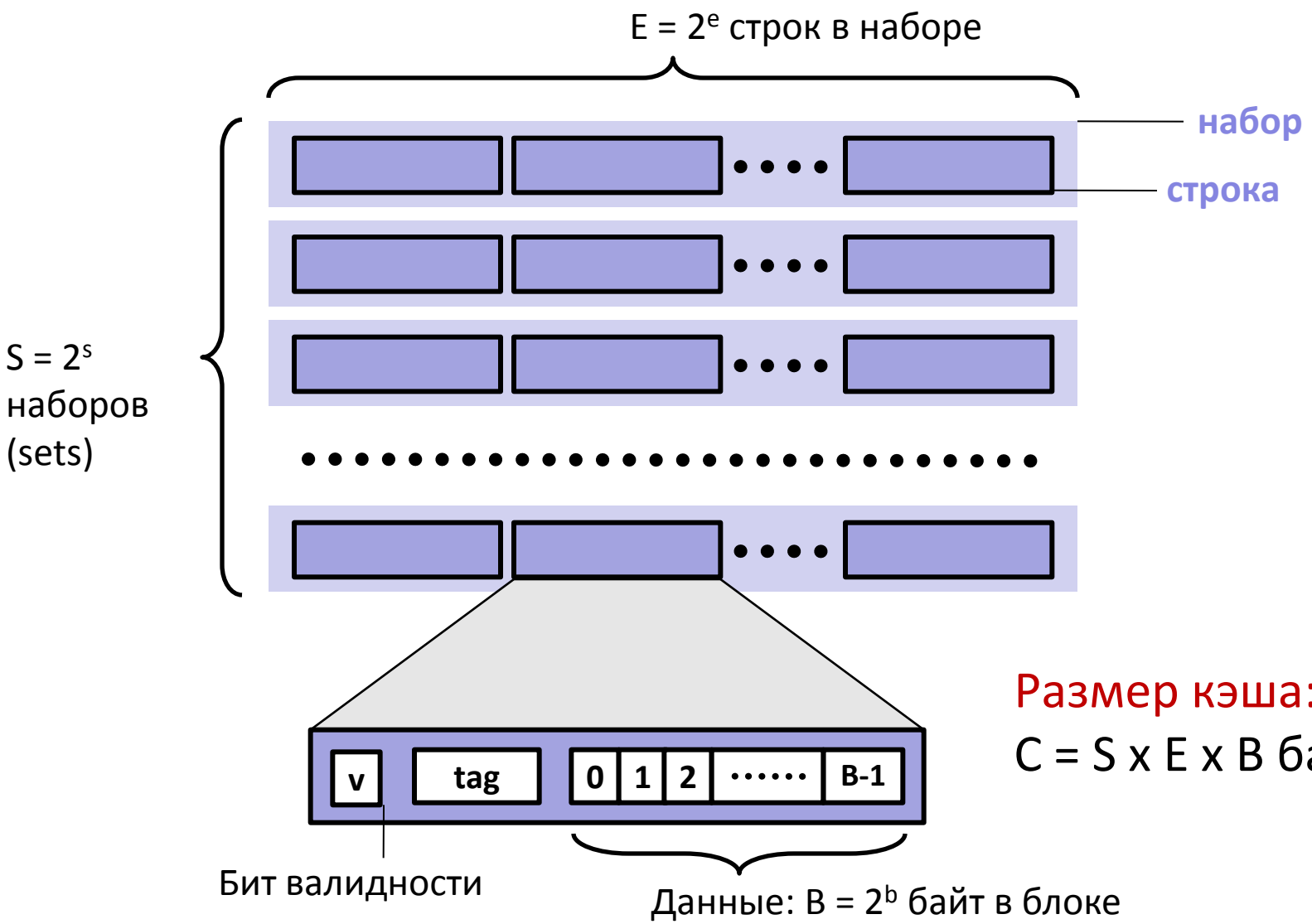
Кэш памяти

- Кэш оперативной памяти – небольшая, быстрая память (SRAM). Управление кэшем аппаратное.
  - Хранит в себе часто используемые блоки оперативной памяти
- ЦПУ сперва ищет требуемые данные в кэше (L1, L2 и L3), и только потом в оперативной памяти.



© 2015 МГУ/ВМК/СП

Организация кэша (S, E, B)



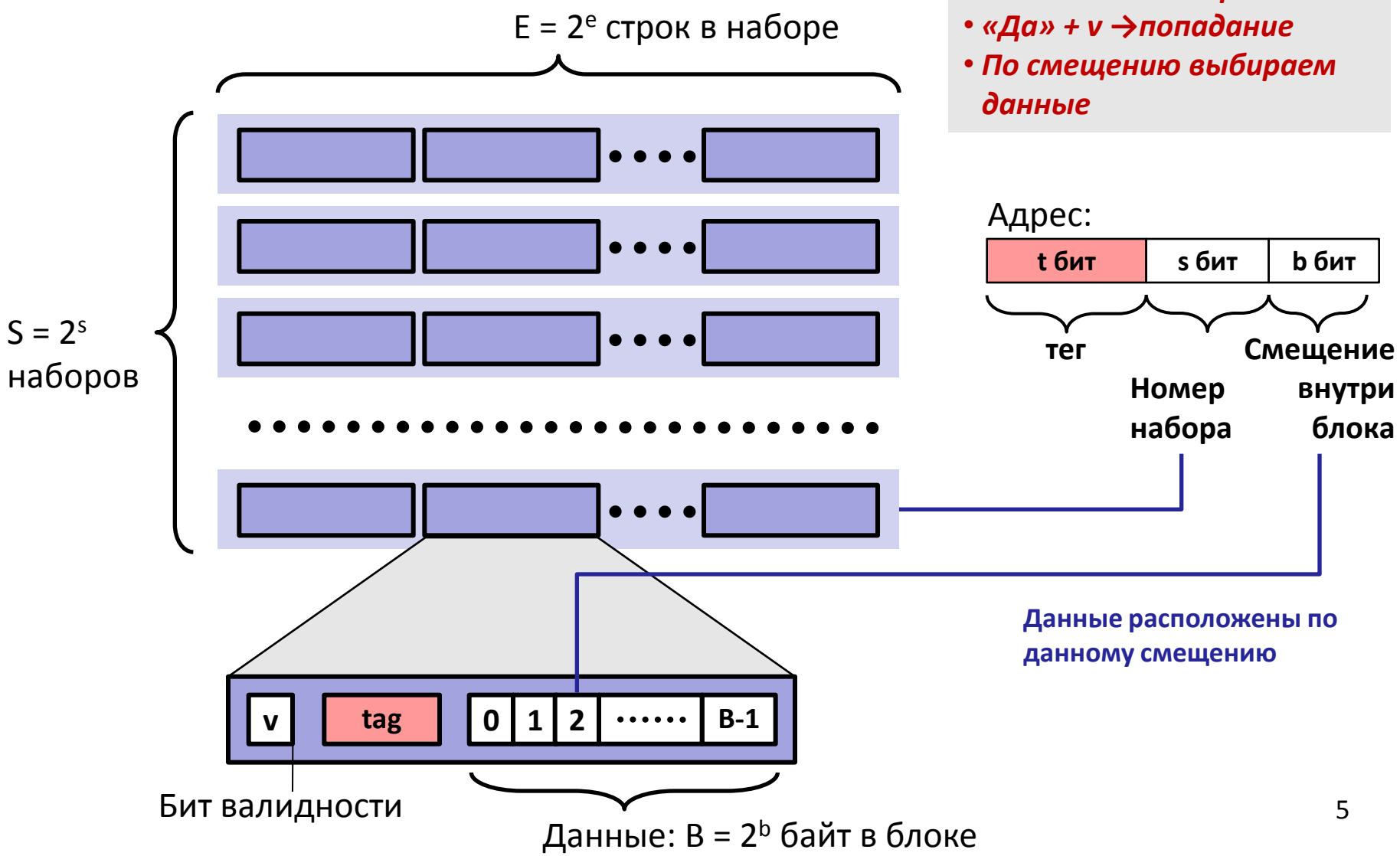
Размер кэша:

C = S x E x B байт

© 2015 МГУ/ВМК/СП

Чтение данных из кэша

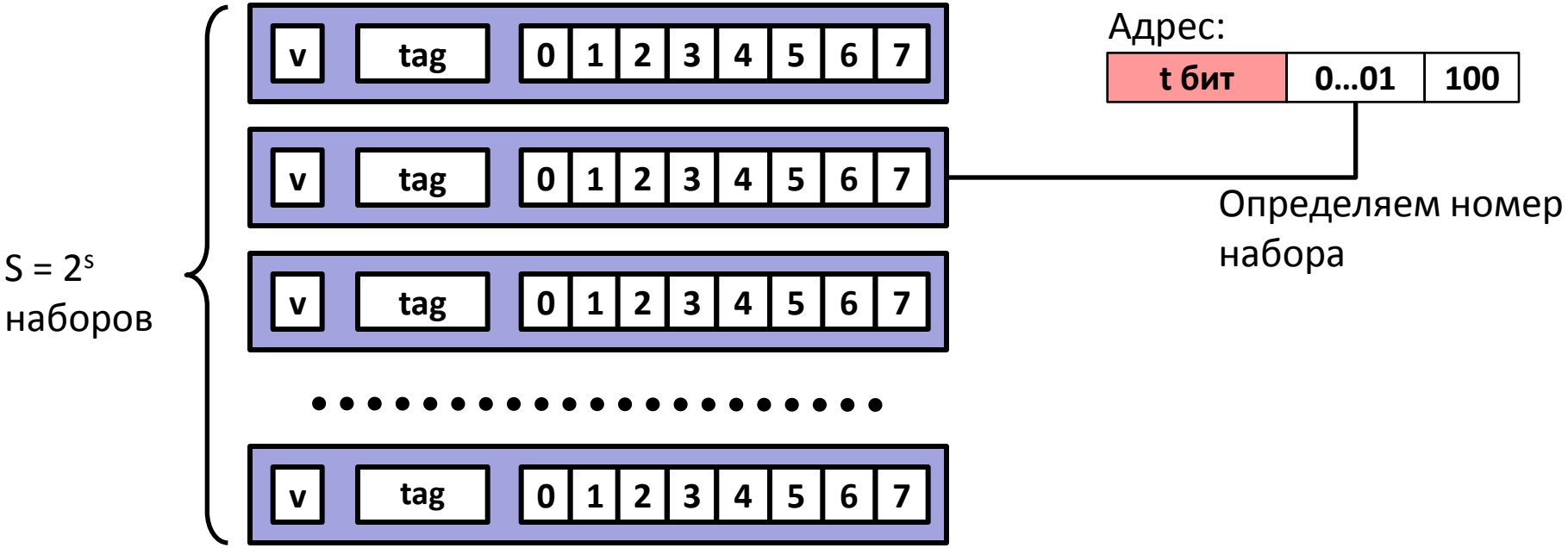
- ***Определяем номер набора***
- ***Проверяем на совпадение тега по всем строкам***
- ***«Да» + v →попадание***
- ***По смещению выбираем данные***



© 2015 МГУ/ВМК/СП

Пример: Кэш прямого отображения (E = 1)

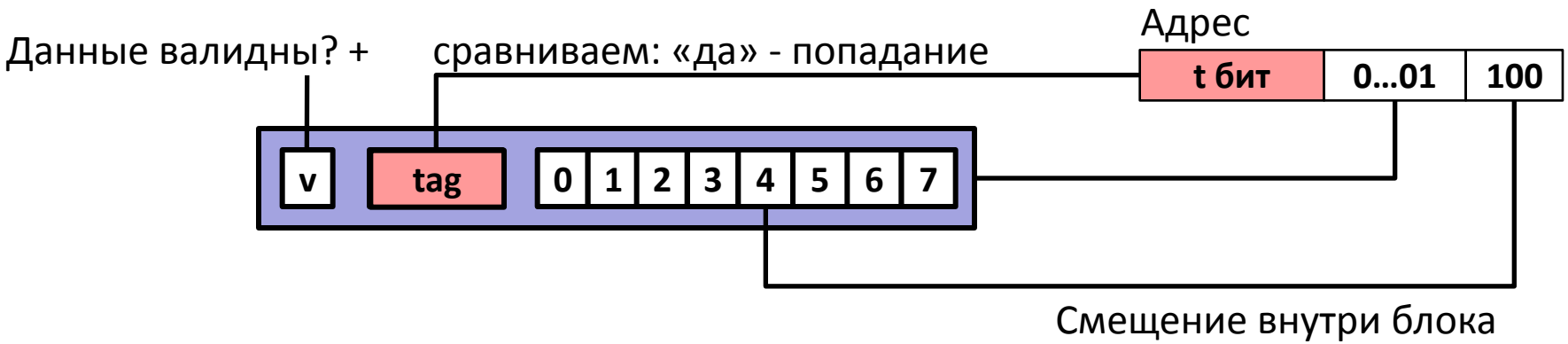
Прямое отображение: одна строка в наборе
Для данного примера: размер блока 8 байт



© 2015 МГУ/ВМК/СП

Пример: Кэш прямого отображения (E = 1)

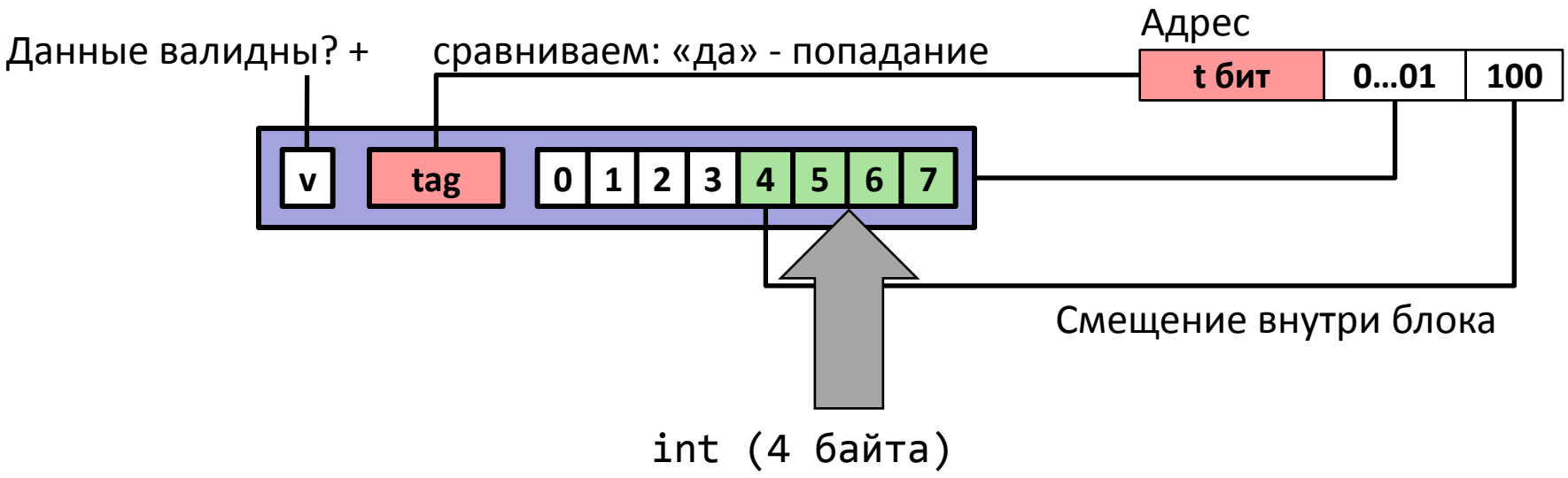
Прямое отображение: одна строка в наборе
Для данного примера: размер блока 8 байт



© 2015 МГУ/ВМК/СП

Пример: Кэш прямого отображения (E = 1)

Прямое отображение: одна строка в наборе
Для данного примера: размер блока 8 байт



Тег не совпал: строка вытесняется из кэша

© 2015 МГУ/ВМК/СП

Моделируем кэш прямого отображения

t=1	s=2	b=1
x	xx	x

M=16 адресуемых байтов, B=2 байта в блоке,
S=4 наборов, E=1 блок в наборе

Последовательность (трасса)
запрашиваемых адресов (чтение одного байта):

0	**[0$\underline{00}$0$_2$],**	промах
1	**[0$\underline{00}$1$_2$],**	попадание
7	**[0$\underline{11}$1$_2$],**	промах
8	**[1$\underline{00}$0$_2$],**	промах
0	**[0$\underline{00}$0$_2$]**	промах

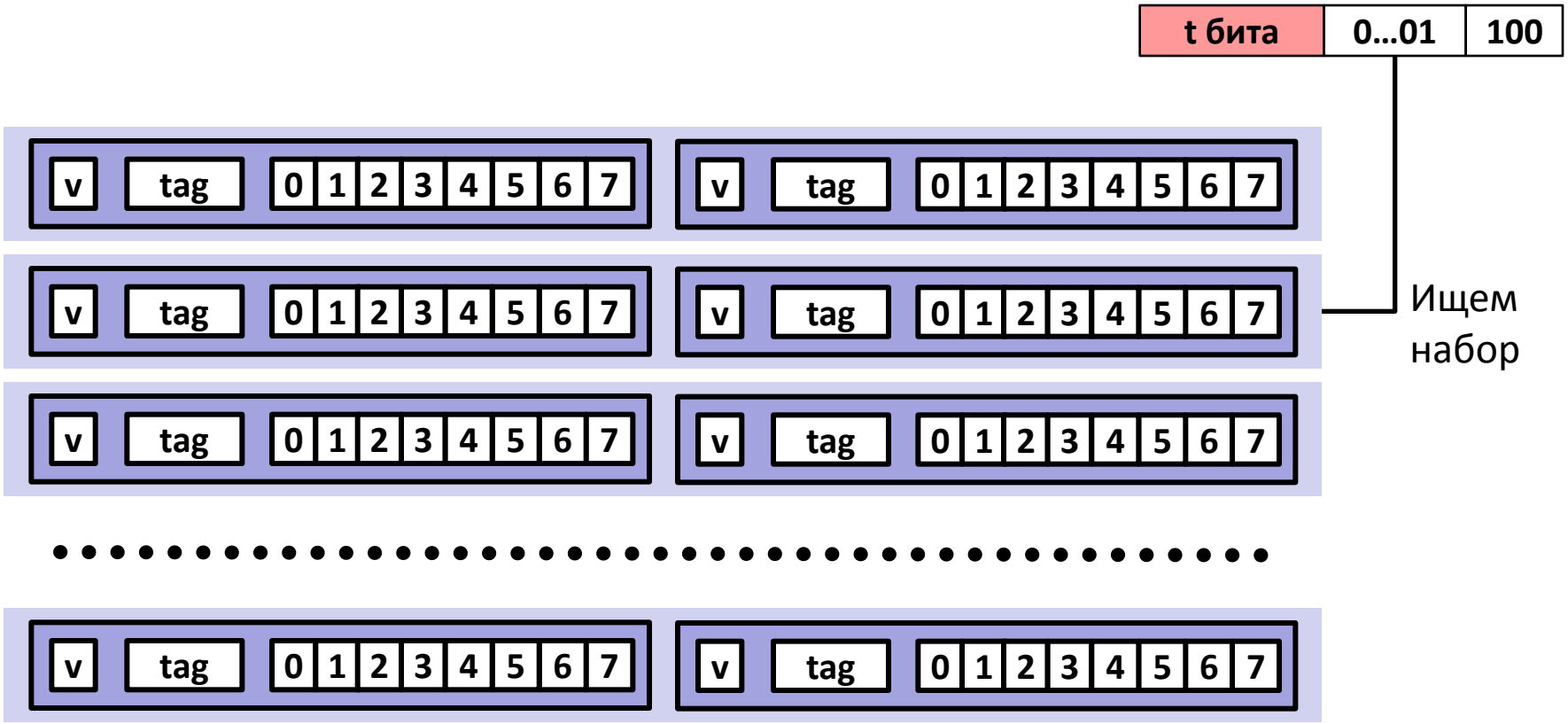
	v	Тег	Блок
Набор 0	1	0	M[0-1]
Набор 1	0	?	?
Набор 2	0	?	?
Набор 3	1	0	M[6-7]

© 2015 МГУ/ВМК/СП

N-канальный ассоциативный кэш (N = 2)

N = 2: Две строки в наборе
Кэшируются блоки по 8 байт

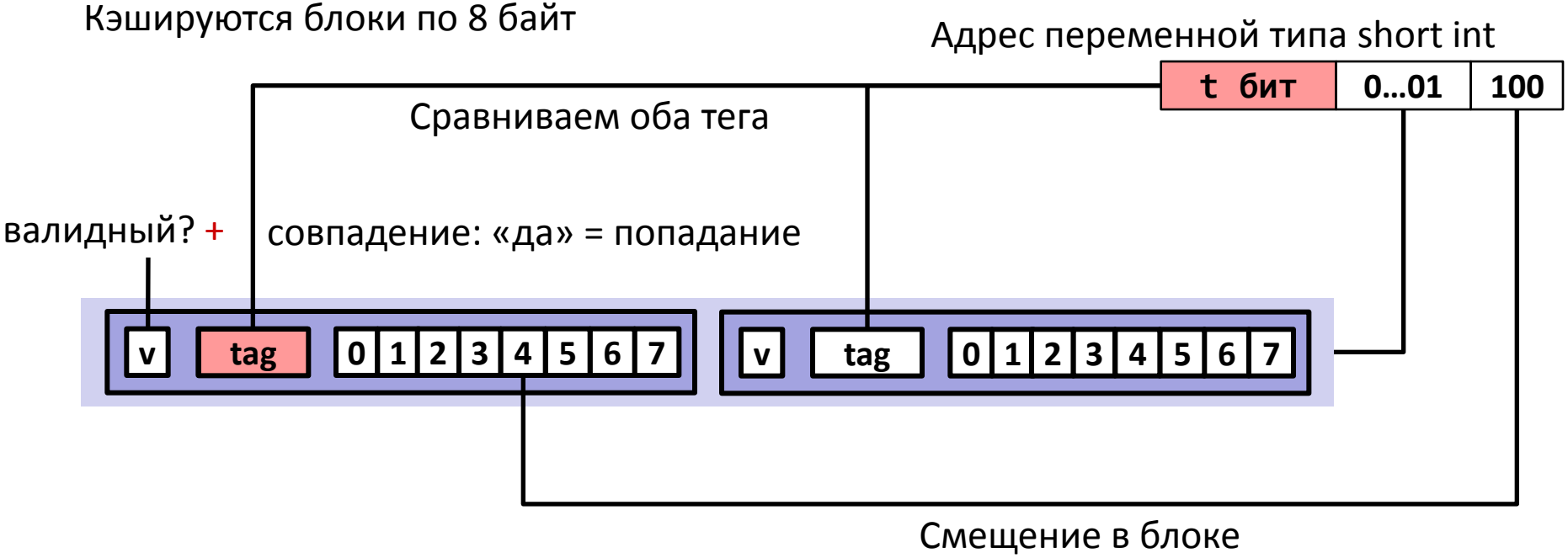
Адрес переменной типа short int



© 2015 МГУ/ВМК/СП

N-канальный ассоциативный кэш (N = 2)

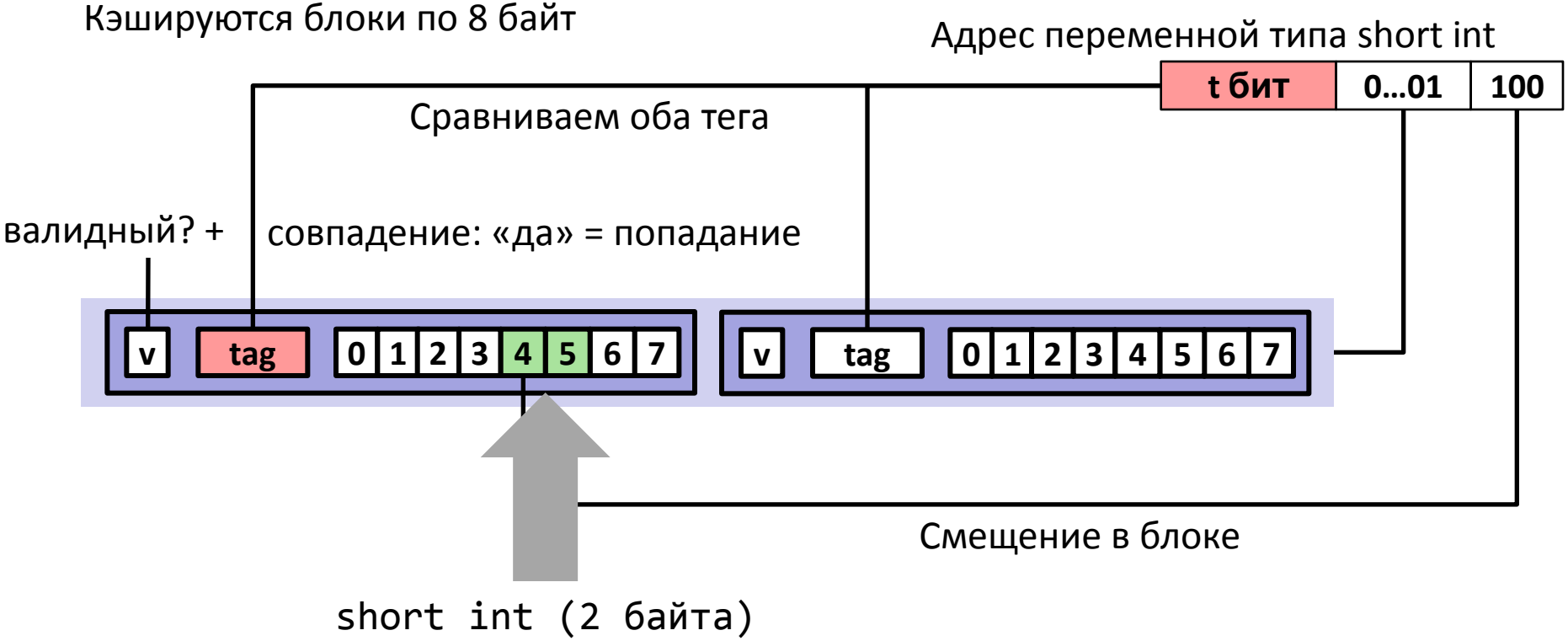
N = 2: Две строки в наборе
Кэшируются блоки по 8 байт



© 2015 МГУ/ВМК/СП

N-канальный ассоциативный кэш (N = 2)

N = 2: Две строки в наборе
Кэшируются блоки по 8 байт



Совпадений нет:

- Одна из строк будет вытеснена из кэша
- Стратегии замещения строк: произвольная, самая «старая» (LRU), …

© 2015 МГУ/ВМК/СП

Моделируем 2-канальный ассоциативный кэш

t=2	s=1	b=1
xx	x	x

M=16 адресуемых байт, B=2 байта в блоке,
S=2 набора, E=2 блока в наборе

Последовательность (трасса)
запрашиваемых адресов (чтение одного байта):

0	**[00$\underline{0}$0$_2$],**	промах
1	**[00$\underline{0}$1$_2$],**	попадание
7	**[01$\underline{1}$1$_2$],**	промах
8	**[10$\underline{0}$0$_2$],**	промах
0	**[00$\underline{0}$0$_2$]**	попадание

	v	**Тег**	**Блок**
Набор 0	1	00	M[0-1]
	1	10	M[8-9]
Набор 1	1	01	M[6-7]
	0	?	?

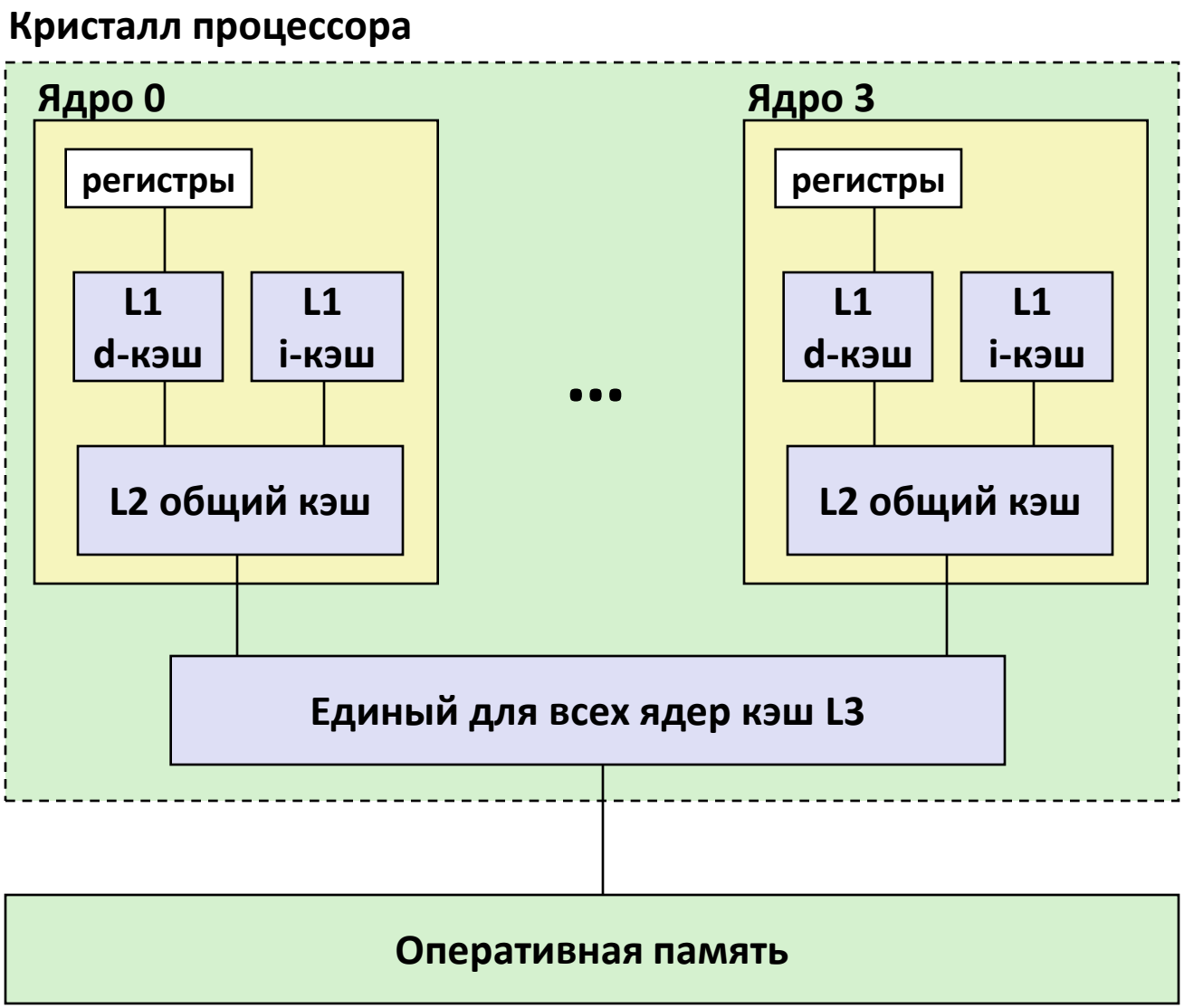
© 2015 МГУ/ВМК/СП

Запись данных в память

- Несколько копий данных:
  - L1, L2, оперативная память, диск
- Как поступать при попадании?
  - Сквозная запись (пишем в память незамедлительно)
  - Отложенная запись (откладываем до момента вытеснения строки)
    - Требуется дополнительный бит-признак, что данные отличаются
- Как поступать при промахе?
  - Запись с размещением в кэше
    - Эффективно когда выполняется несколько записей в последовательные адреса
  - Запись без размещения в кэше
- Типичные комбинации политик управления кэшем
  - Сквозная запись + Запись без размещения
  - **Отложенная запись** + Запись с размещением

© 2015 МГУ/ВМК/СП

Иерархия кэшей в Intel Core i7



L1 i-кэш и d-кэш:
32 KB, 8-канальный,
Время доступа:
4 такта

L2 общий кэш:
256 KB, 8-канальный,
Время доступа:
11 тактов

L3 общий кэш:
8 MB, 16-канальный,
Время доступа:
30-40 тактов

Размер блока:
64 байта у всех кэшей.

© 2015 МГУ/ВМК/СП

Метрики производительности кэша

- Коэффициент промахов
  - Отношение количества промахов к общему числу обращений (промахи / обращения) = 1 – коэффициент попаданий
  - Характерные показатели (в процентах):
    - 3-10% для L1
    - Может быть достаточно малым (< 1%) для L2, зависит от размера и т.д.
- Время попадания
  - Длительность извлечения данных из кэша
    - Включает время определения того, есть ли требуемые данные в кэше
  - Характерные показатели:
    - 1-2 тактов для L1
    - 5-20 тактов для L2
- Накладные расходы при промахе
  - Дополнительное время из-за промаха
    - Обычно 50-200 тактов для обращения к памяти

© 2015 МГУ/ВМК/СП

Что означают перечисленные показатели?

- Гигантская разница по времени промахов и попаданий
  - Два порядка, для L1 и оперативной памяти

- Пример: 99% попаданий вдвое более эффективно чем 97%
  - Характеристики:
    время попадания 1 такт
    накладные расходы при промахе 100 тактов

  - Среднее время доступа к элементу кэша:
  - 97% попаданий: 1 такт + 0.03 * 100 тактов = 4 такта
  - 99% попаданий: 1 такт + 0.01 * 100 тактов = 2 такта

- Поэтому “коэффициент неудач” используется вместо “коэффициента попаданий”

© 2015 МГУ/ВМК/СП

Дружественный к кэшу код

- В первую очередь улучшать часто работающий код
  - Тела вложенных циклов, часто вызывающиеся функции
- Минимизировать промахи при обращении к кэшу в теле вложенного цикла
  - Повторяющеюся обращения к одним и тем же переменным (временная локальность)
  - Проход по массиву с шагом 1 (пространственная локальность)

© 2015 МГУ/ВМК/СП

Оценка производительности

- Численная характеристика относительной производительности
  - Всего компьютера в целом
  - Отдельных компонент
  - Генерируемого компилятором кода
- Классификация
  - Синтетические / основанные на реальных приложениях
  - Микробенчмарк
- Индустриальные бенчмарки
  - SPEC
  - LINPACK
    - Решает систему линейных алгебраических уравнений Ax = b

© 2015 МГУ/ВМК/СП

Измерение времени

- В Pentium появился регистр 64-разрядный регистр TSC, подсчитывающий количество выполнившихся тактов
- Инструкция `rdtsc` считывает значение регистра TSC и заносит его в `EDX:EAX`
- `rdtsc` может быть недоступна пользователям на некоторых системах

```
section .rodata
 format db ‘0x%08X 0x%08X’, 10, 0

section .text
global CMAIN
CMAIN:
   ...
   rdtsc
   mov     dword [esp + 8], eax
   mov     dword [esp + 4], edx
   mov     dword [esp], format
   call    printf
   ...
```

© 2015 МГУ/ВМК/СП

Оценка производительности памяти: синтетический бенчмарк (контрольная задача)

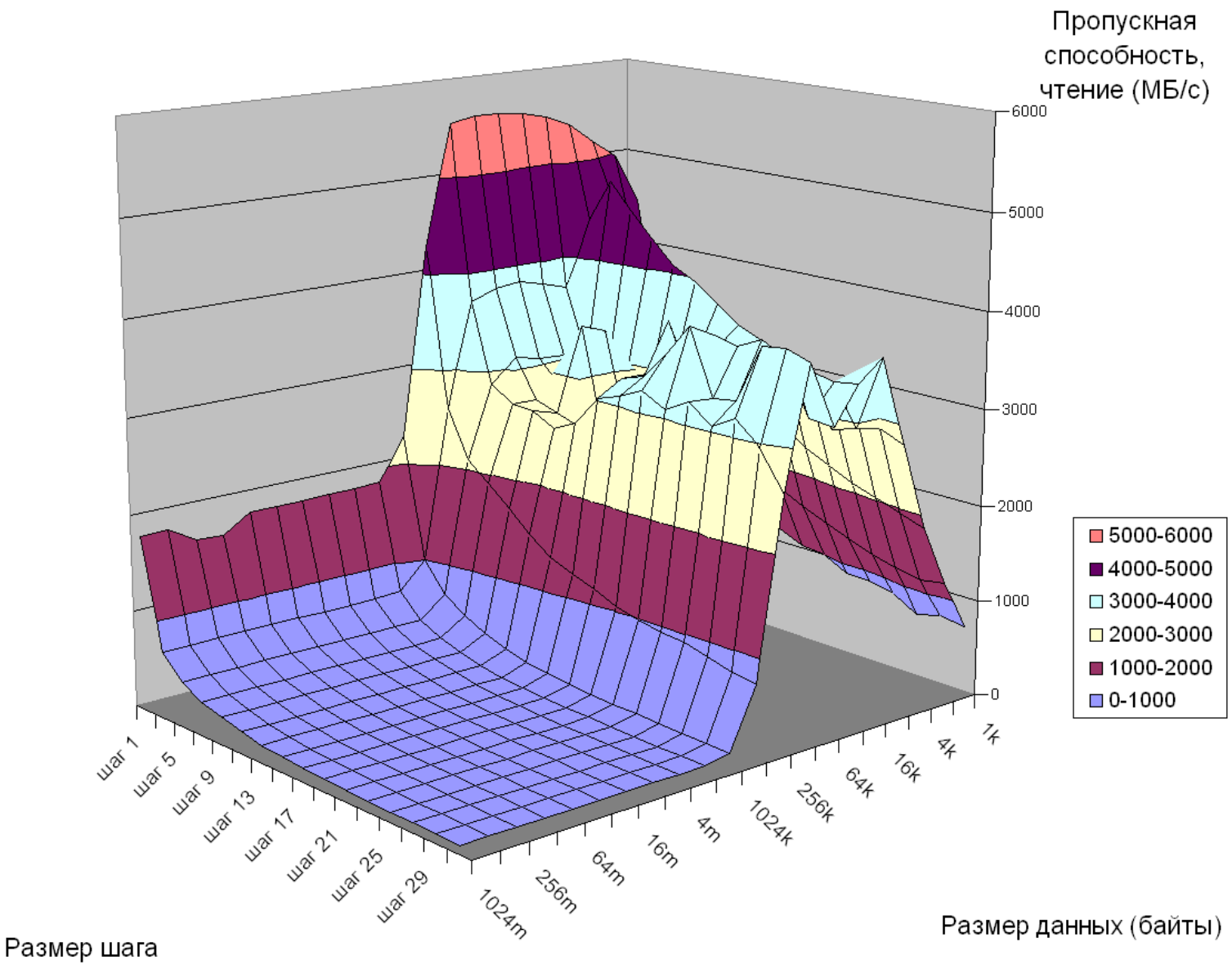
```
/* Оценочная функция */
void test(int elems, int stride) {
    int i, result = 0;
    volatile int sink;

    for (i = 0; i < elems; i += stride)
    result += data[i];
    sink = result;
}

/*
   Запуск test(elems, stride) и вычисление пропускной способности
   при чтении (МБ/с)
*/
double run(int size, int stride, double Mhz) {
    double cycles;
    int elems = size / sizeof(int);

    test(elems, stride);                      /* разогрев кэша */
    cycles = fcyc2(test, elems, stride, 0);  /* вызываем test(elems,stride) */
    return (size / stride) / (cycles / Mhz); /* переводим кол-во циклов в МБ/с
*/
}
```

© 2015 МГУ/ВМК/СП

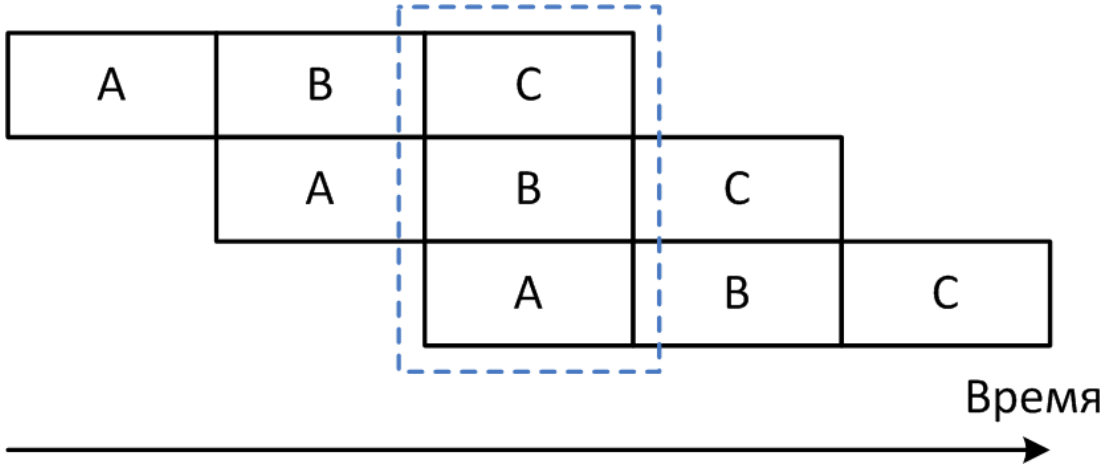


Intel(R) Xeon(TM) CPU 2.80GHz 2x2

© 2015 МГУ/ВМК/СП

Конвейер

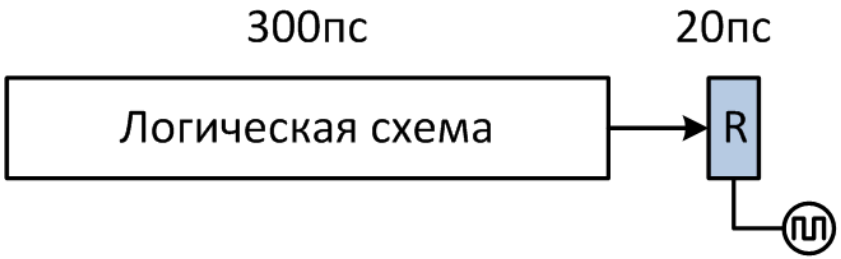
- Общая для различный предметных областей методика
- Длительность обработки неизменна или несколько увеличивается
- Увеличение пропускной способности

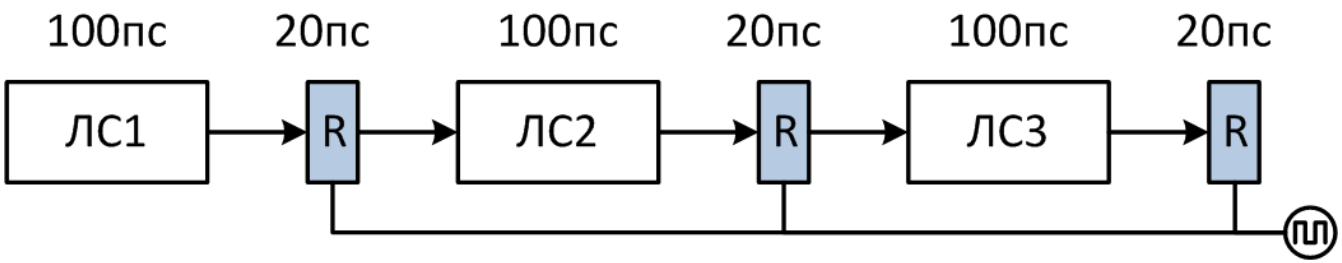


© 2015 МГУ/ВМК/СП

Организация конвейерных вычислений

Задержка 320пс
Пропускная способность 3.12 GIPS

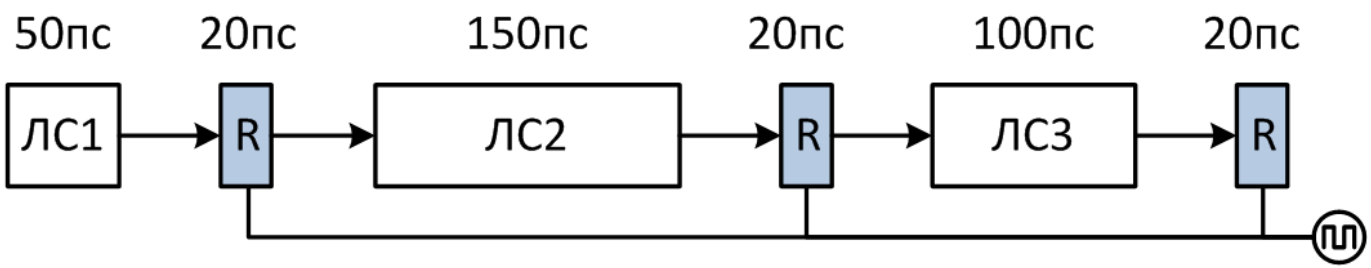




Задержка 360пс
Пропускная способность 8.33 GIPS

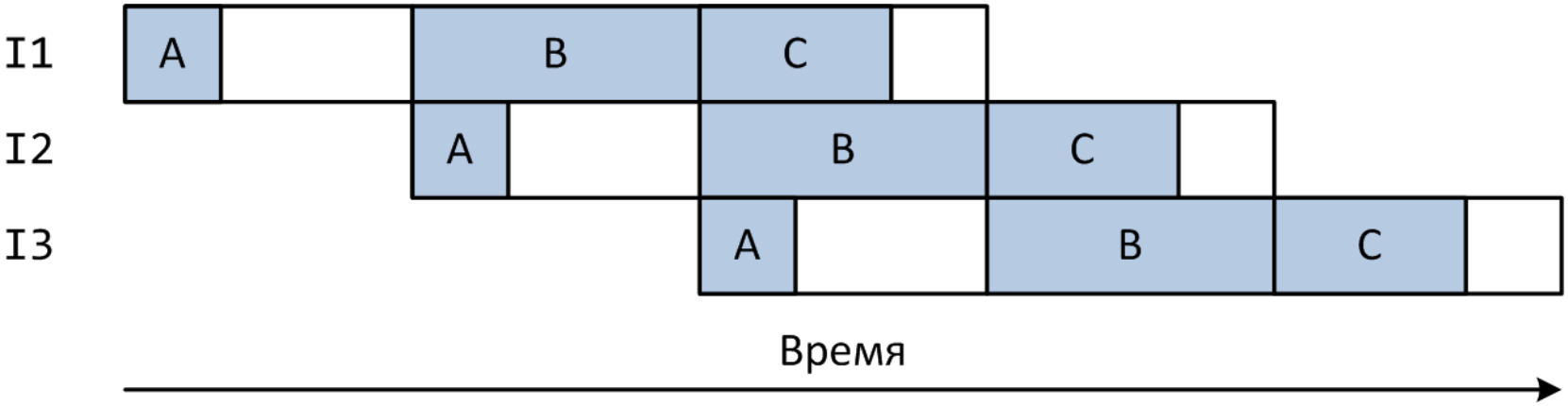
© 2015 МГУ/ВМК/СП

Неоднородность ступеней конвейера



Задержка 510пс

Пропускная способность 5.88 GIPS



© 2015 МГУ/ВМК/СП

Упрощенная схема работы конвейера

1. (F) Извлечение инструкции из памяти
2. (D) Декодирование, обновление EIP
3. (E) Извлечение данных
4. (O) Непосредственное выполнение операции
5. (W) Запись результата

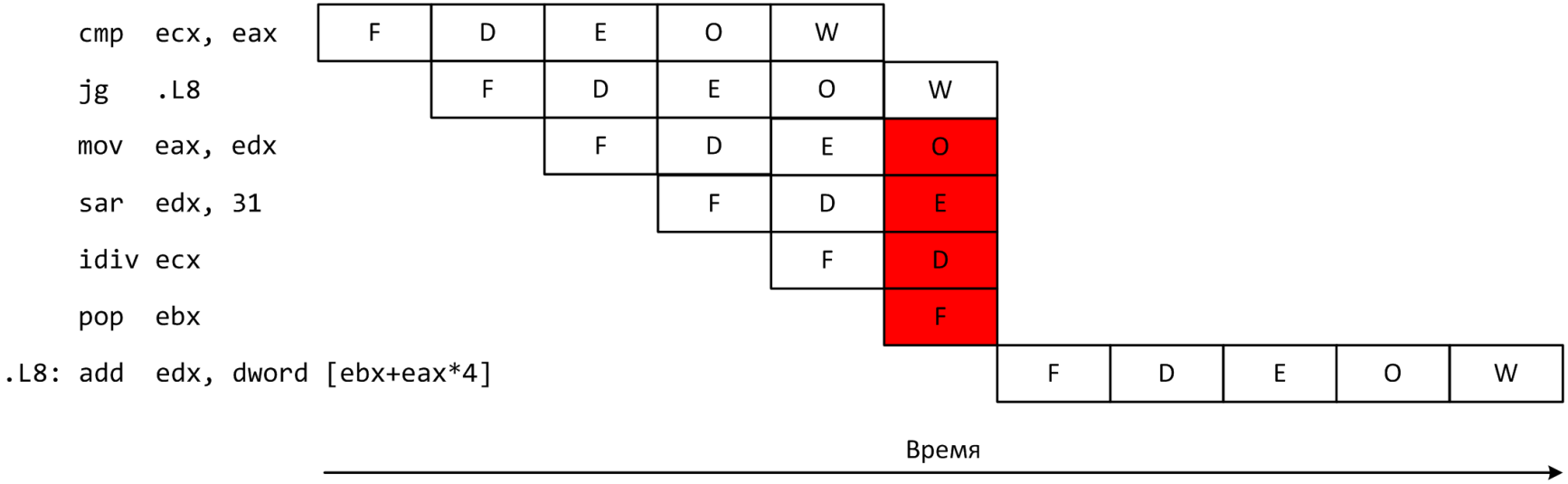
© 2015 МГУ/ВМК/СП

Недостатки конвейерной организации

- Опустошение конвейера
  - Современные процессоры могут содержать конвейеры длины 15 и более
  - Изменение EIP сбрасывает промежуточные результаты выполнения следующих инструкций
- Зависимости по чтению и записи регистров
- Остановки из-за обращения к памяти
- Сброс промежуточных результатов при возникновении исключительной ситуации

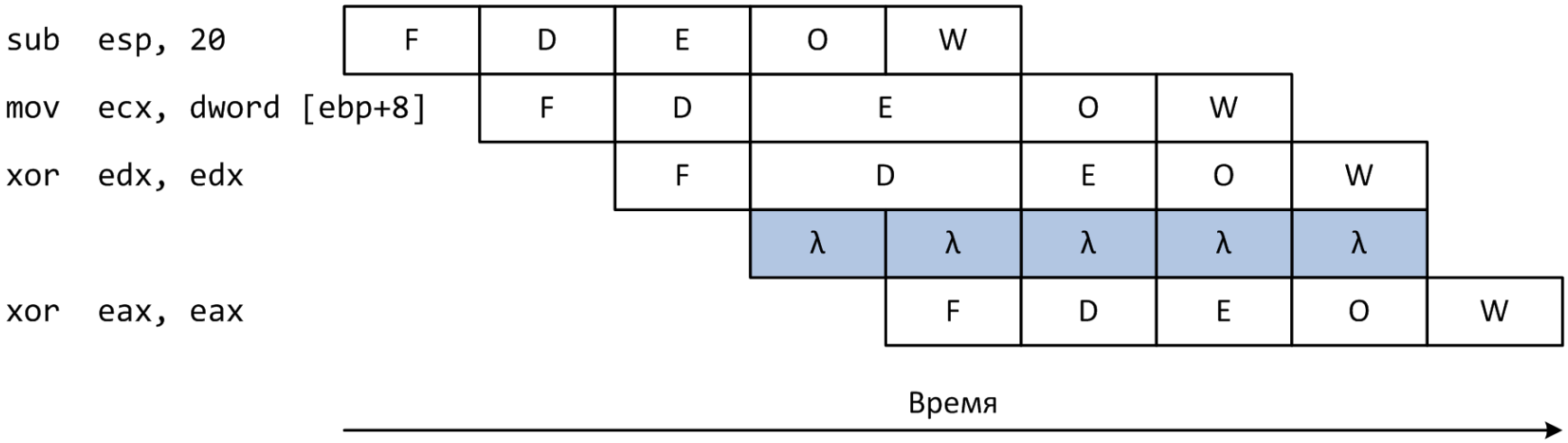
© 2015 МГУ/ВМК/СП

Опустошение конвейера



© 2015 МГУ/ВМК/СП

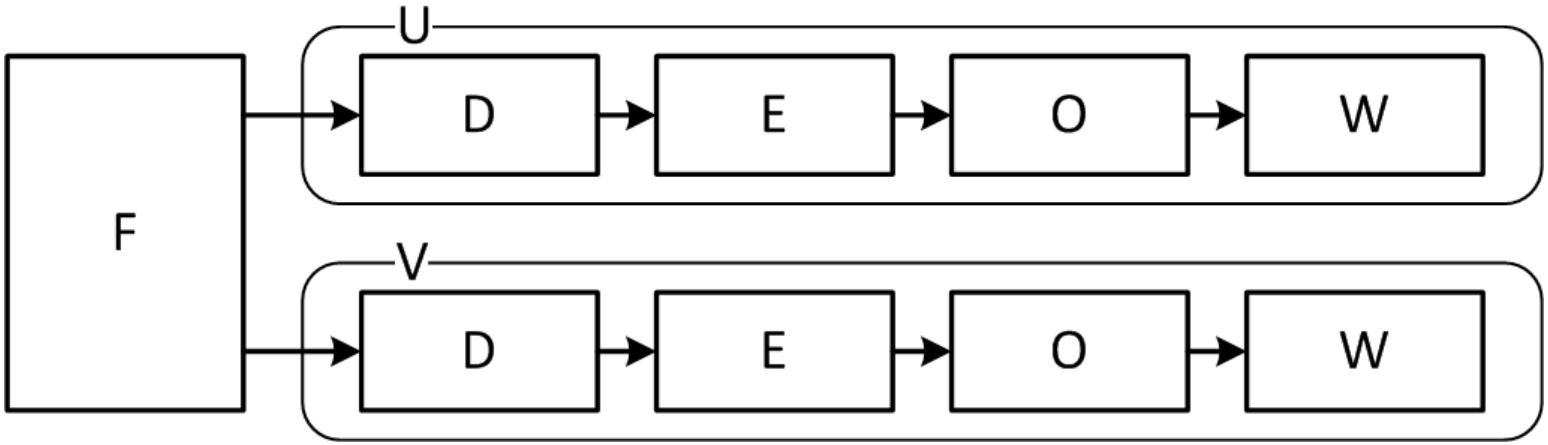
Приостановка конвейера



© 2015 МГУ/ВМК/СП

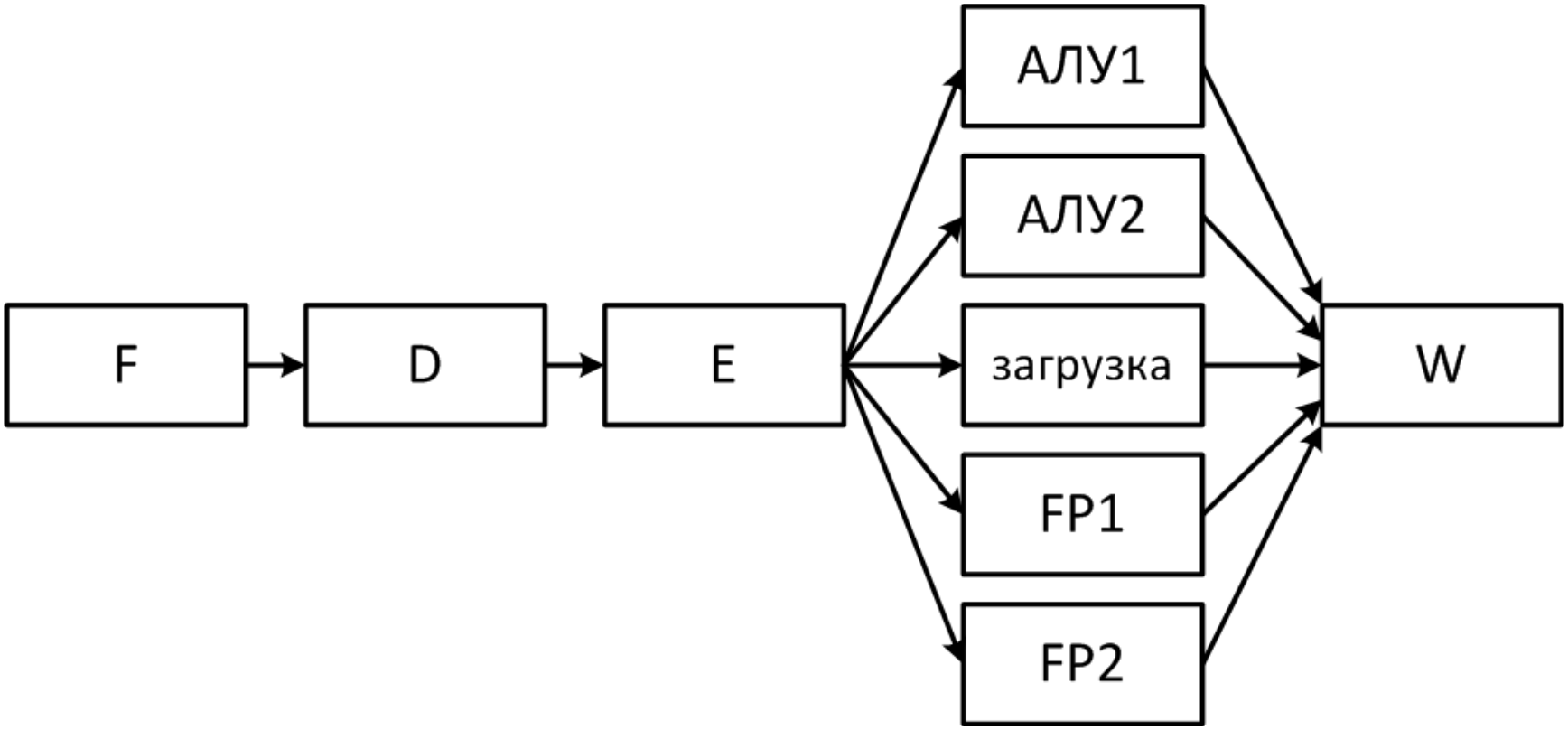
Двойной конвейер

- Двойной конвейер с общим этапом выборки команд
- Впервые появился на Pentium: введены u- и v-конвейеры
- Возможности конвейеров неравноценны



© 2015 МГУ/ВМК/СП

Суперскалярная архитектура



© 2015 МГУ/ВМК/СП

RISC vs CISC

(Reduced vs Complex Instruction Set Computer)

- Исторически CISC предшествовал RISC
  - PDP-11 → VAX / CISC, 1977 г.
  - MIPS, SPARC / RISC, конец 80-х
- Простые операции
  - Ограниченный набор простых команд (например, нет деления)
  - Команда выполняется за один такт
  - Фиксированная длина команды (простота декодирования)
- Конвейер
  - Каждая операция разбивается на однотипные простые этапы, которые выполняются параллельно
  - Каждый этап занимает 1 такт, в т.ч. декодирование
- Регистры
  - Много однотипных взаимозаменяемых регистров (могут использоваться и для данных, и для адресации)

© 2015 МГУ/ВМК/СП

RISC vs CISC

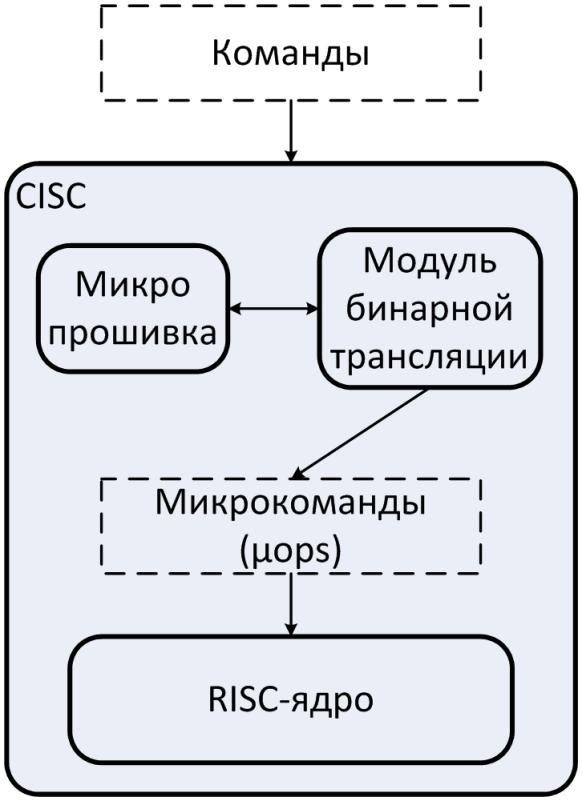
(Reduced vs Complex Instruction Set Computer)

- Модель работы с памятью
  - Отдельные команды для загрузки/сохранения в память
  - Команды обработки данных работают только с регистрами
- Сложность оптимизаций перенесена из процессора в компилятор
  - Производительность сильно зависит от компилятора
- Итого: более простое ядро, выше частота процессора
- В P6 реализован гибридный подход: CISC-процессор с RISC-ядром

© 2015 МГУ/ВМК/СП

Производительность: особенности современной архитектуры Intel64 и не только

- Многоядерность
- Многоуровневый кэш, раздельный кэш кода и данных
- Суперскалярная архитектура
- Векторные команды MMX, 3DNow!, SSE, ...
- Внеочередное выполнение команд
- Предсказание переходов
- ...



Наиболее показательный пример бинарной трансляции кода на уровне процессора: Transmeta Crusoe, 2000 г.

© 2015 МГУ/ВМК/СП

Промежуточные итоги
Борьба за производительность

- Программы
  - Оптимизирующий компилятор
    - машинно-независимые и машинно-зависимые оптимизации
  - Выравнивание данных
  - Улучшение локальности

- Аппаратура
  - Иерархическая организация памяти
    - многоуровневое кэширование данных
  - Расслоение памяти
  - Согласованность пропускной способности шин и устройств
  - Перекрытие операций ввода/вывода и вычислений
  - Конвейерное выполнение инструкций
    - Увеличение количества функциональных устройств
  - Упорядочивание архитектуры ЦПУ / RISC
    - Больше возможностей для разработки машинно-зависимых оптимизаций