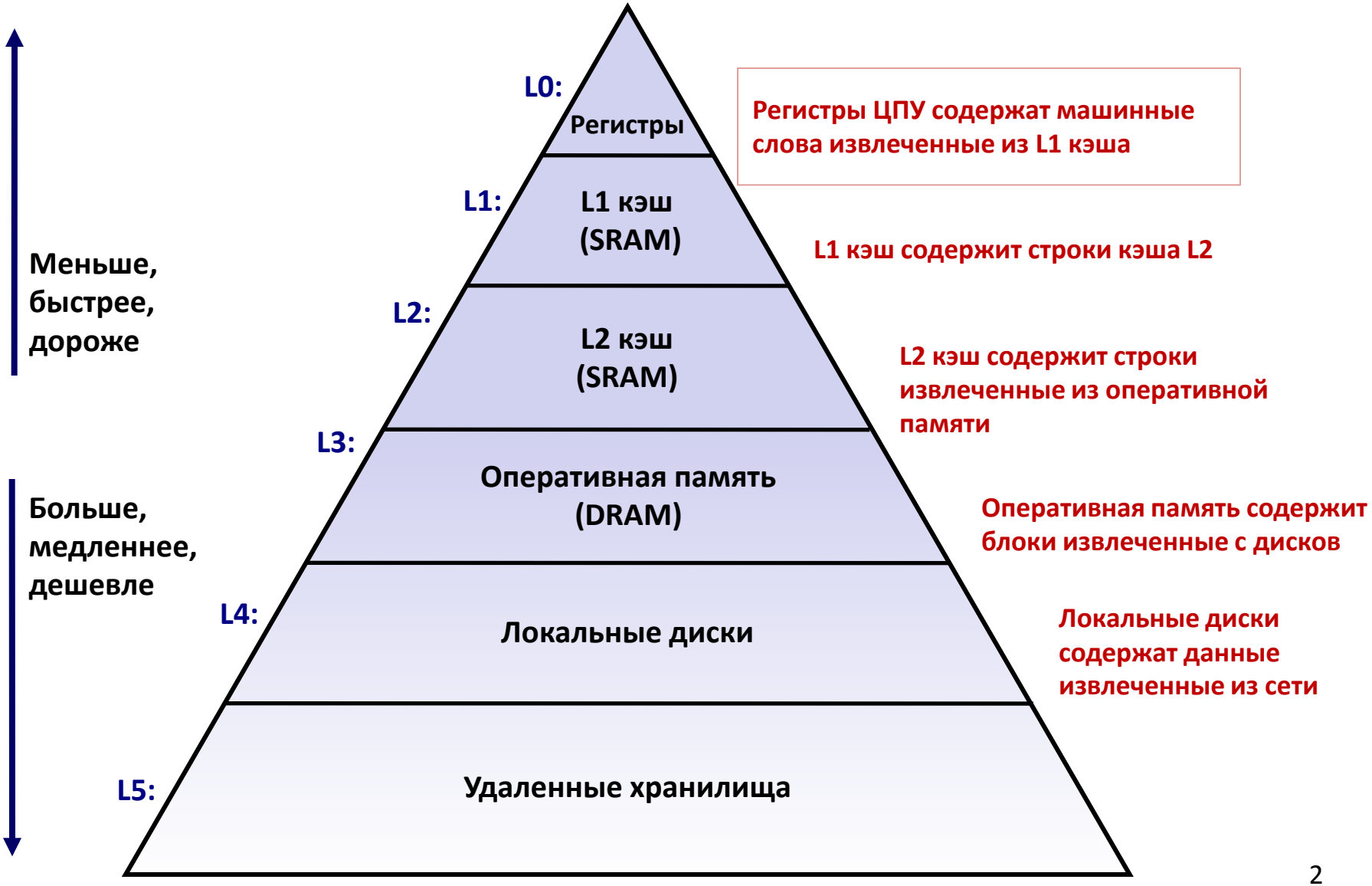
© 2018 МГУ/ВМК/СП

Лекция 0x17

30 апреля

© 2018 МГУ/ВМК/СП

Иерархия памяти



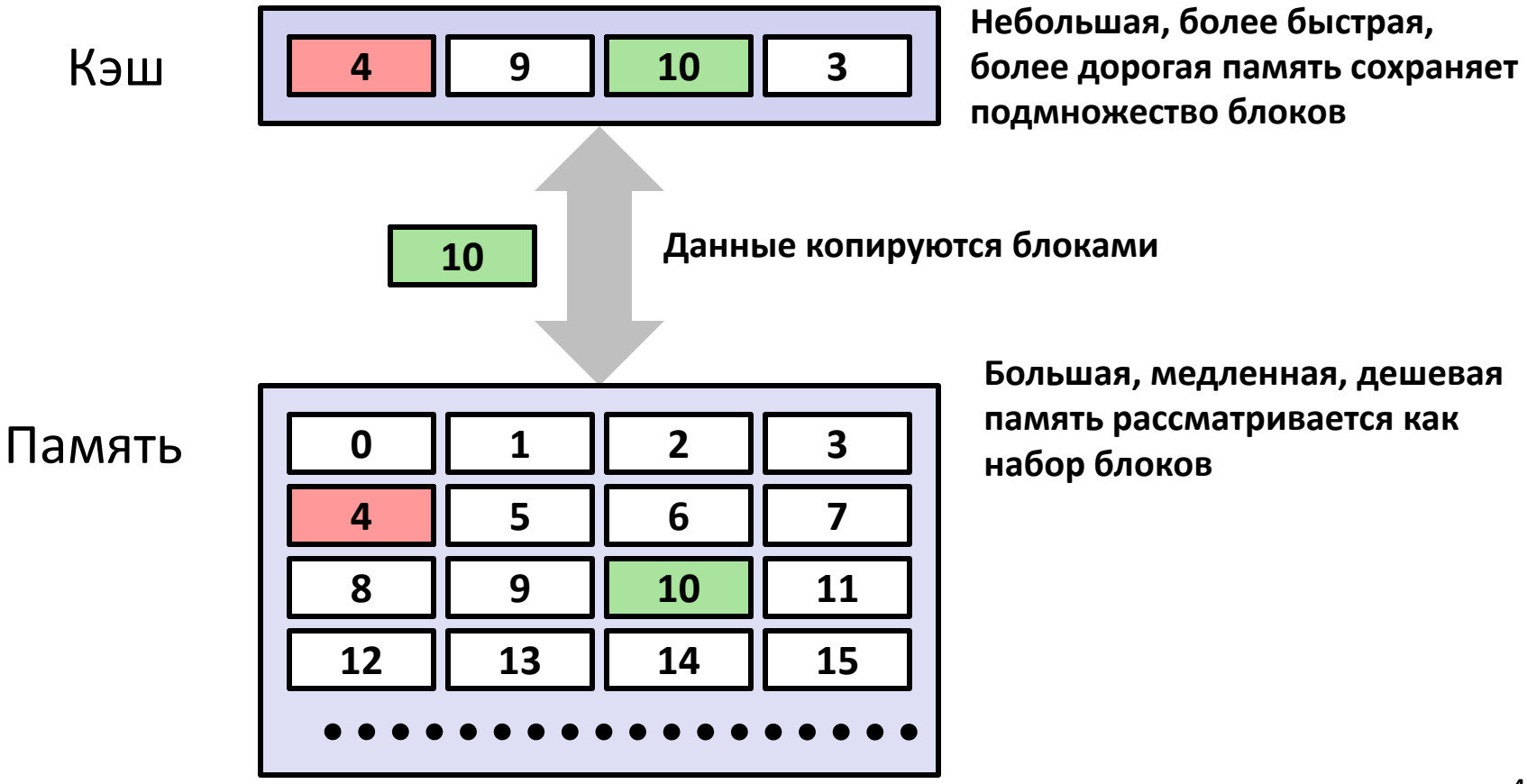
© 2018 МГУ/ВМК/СП

Кэш

- *Кэш:* меньшее объемом, но более быстрое устройство хранения, выступает в роли промежуточного хранилища для большего, но более медленного устройства.
- Основная идея иерархии памяти:
  - Для каждого k, более быстрое, меньшее устройство на уровне k выступает как кэш для большего, но более медленного устройства на уровне k+1.
- Почему это работает?
  - Из-за локальности программа обращается к данным на уровне k гораздо чаше чем к данным на уровне k+1.
  - Устройство уровня k+1 может быть более медленным → большего размера, более дешевым.
- *Результат:* Иерархическая память – большой объем данных, стоит сопоставимо с самой дешевой компонентой, работает с максимальной скоростью.

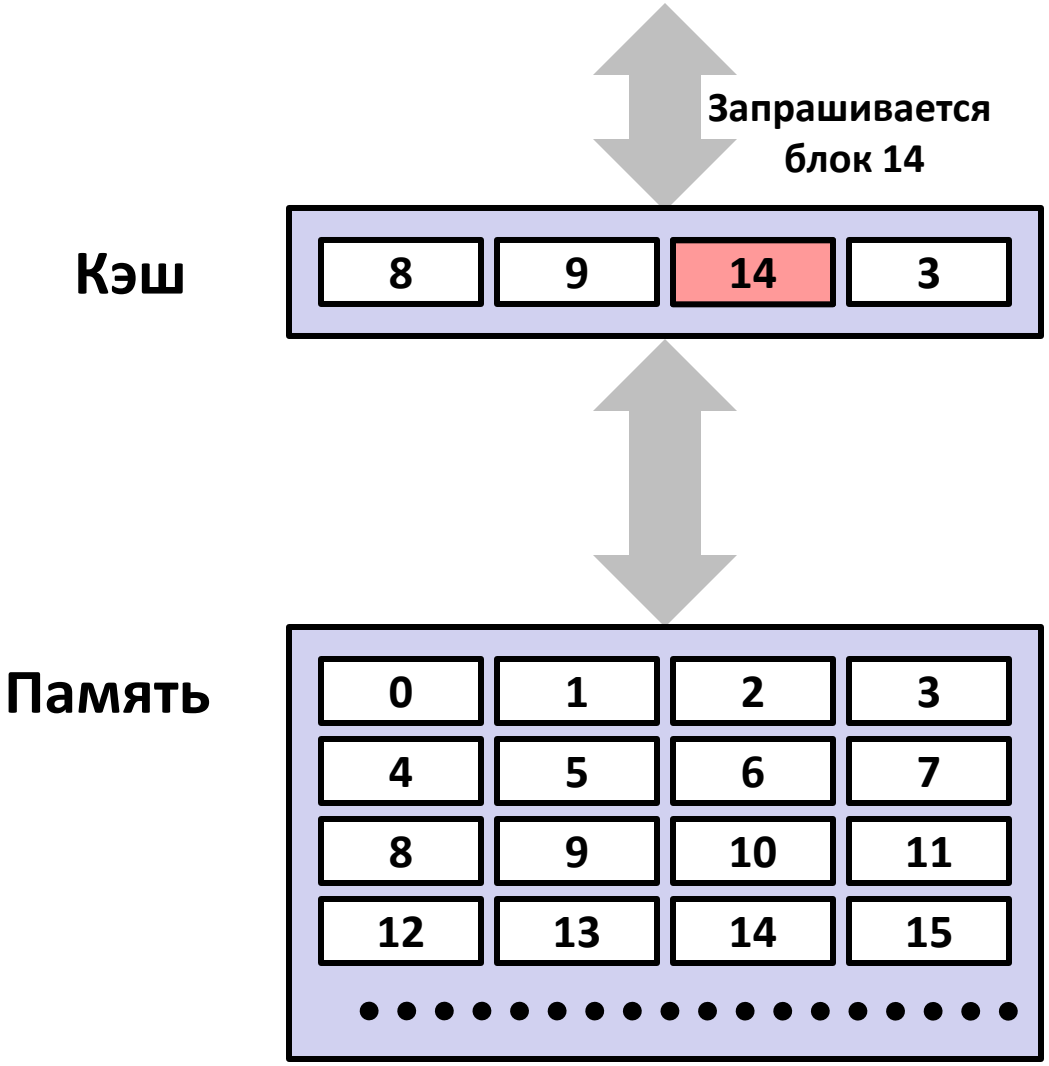
© 2018 МГУ/ВМК/СП

Основная идея кэша



© 2018 МГУ/ВМК/СП

Попадание в кэш



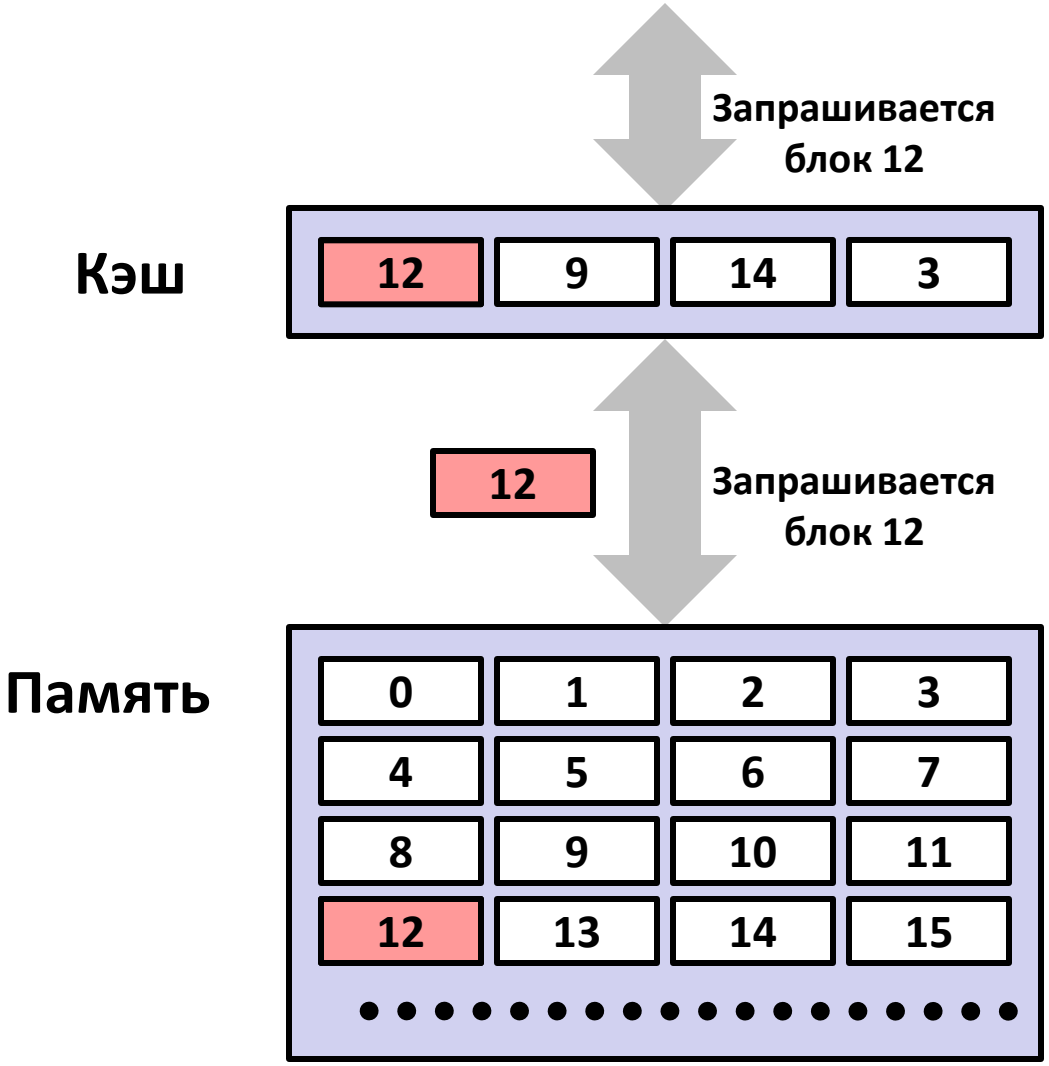
Запрашиваются данные из блока b

Блок b находится в кэше:

Попадание!

© 2018 МГУ/ВМК/СП

Промахи



Запрашиваются данные из блока b

Блока b нет в кэше: Промах!

Блок b извлекается из памяти

Блок b размещается в кэше

- Правила размещения определяют где будет находится блок b
- Правила замещения определяют какой блок будет исключен из кэша

© 2018 МГУ/ВМК/СП

Различные типы промахов

- Холодные (вынужденные) промахи
  - Причина – пустой кэш.
- Промахи из-за конфликтов
  - Количество мест размещения ограничено (может быть единственным)
    - Пример: блок i уровня k+1 размещается в блоке (i mod 4) на уровне k.
  - Промахи из-за конфликтов возникают когда несколько блоков размещаются на одном и том же месте.
    - Пример: запрос блоков 0, 8, 0, 8, 0, 8, ... Будет постоянно вызывать промахи.
- Промахи из-за нехватки емкости
  - Причина – используемых блоков (рабочее множество) больше, чем кэш может в себе вместить.

© 2018 МГУ/ВМК/СП

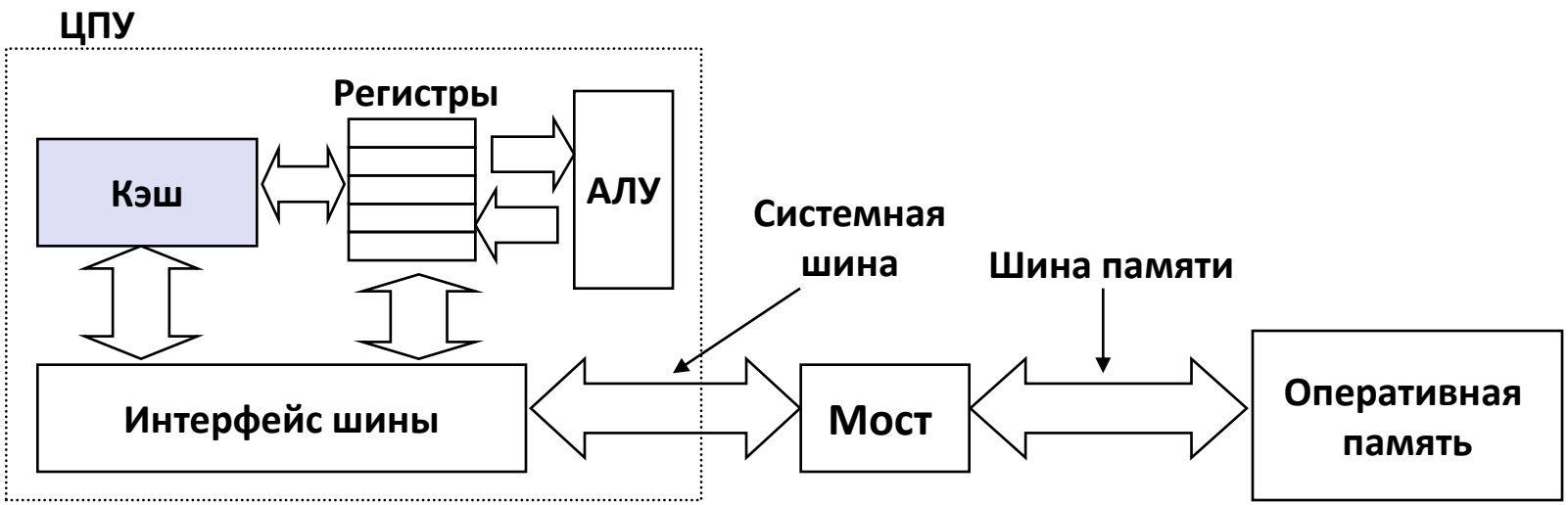
Примеры кэшей

Тип кэша	Что кэшируется?	Где кэшировано?	Задержка (такты)	Управление
Регистры	Слова по 4-8 байт	Ядро ЦПУ	0	Компилятор
L1 кэш	Блок 64 байта	На кристалле, L1	1	Аппаратура
L2 кэш	Блок 64 байта	На/вне кристалла, L2	10	Аппаратура
Буфер ввода/вывода	Части файлов	Оперативная память	100	ОС
Кэш диска	Сектора диска	Контроллер диска	100,000	Прошивка диска
Сетевой кэш	Части файлов	Локальный диск	10,000,000	AFS/NFS клиент
Кэш браузера	Веб-страницы	Локальный диск	10,000,000	Веб браузер
Веб-кэш	Веб-страницы	Диск на удаленном сервере	1,000,000,000	Веб-прокси сервер

© 2018 МГУ/ВМК/СП

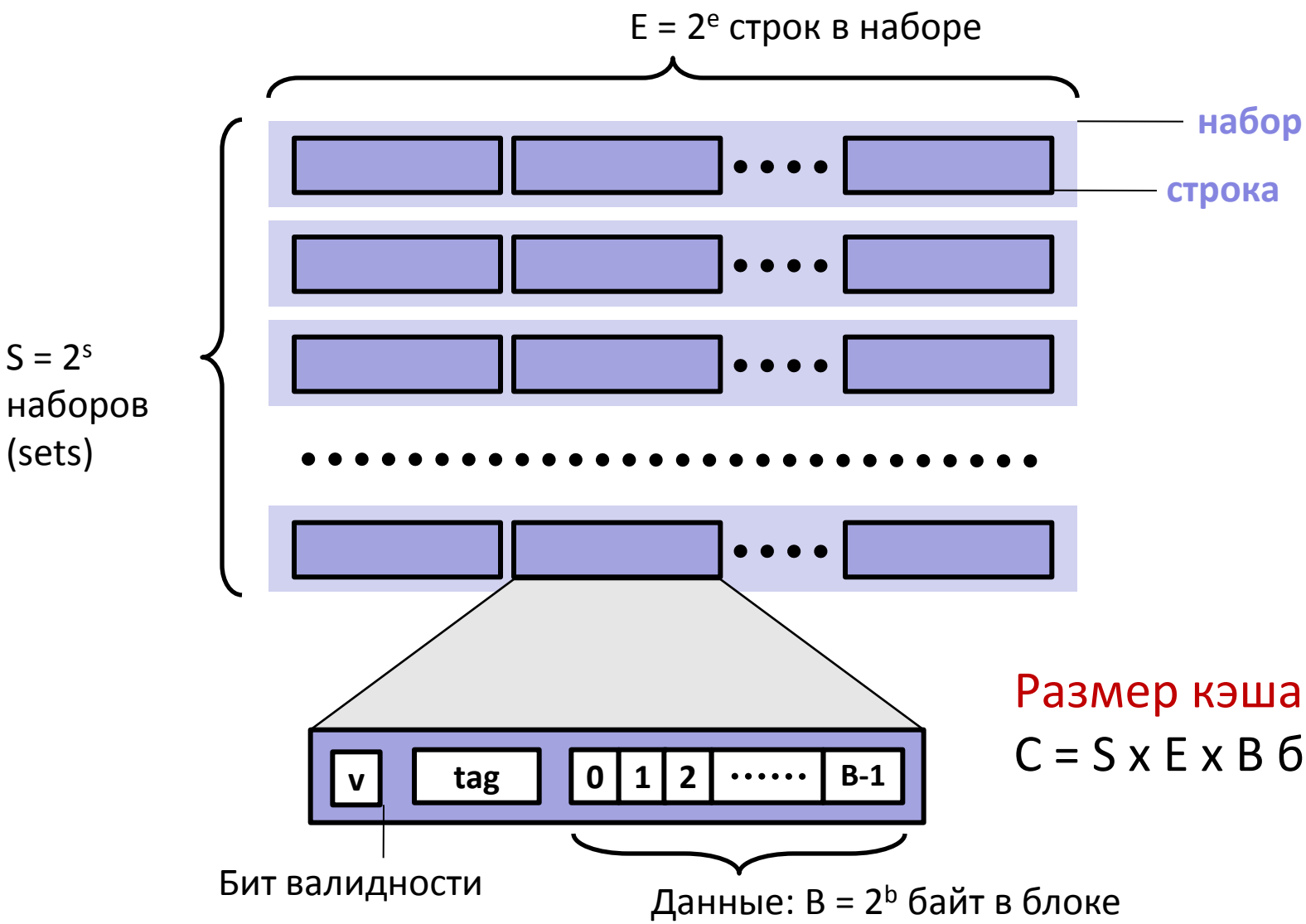
Кэш памяти

- 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.



© 2018 МГУ/ВМК/СП

Организация кэша (S, E, B)



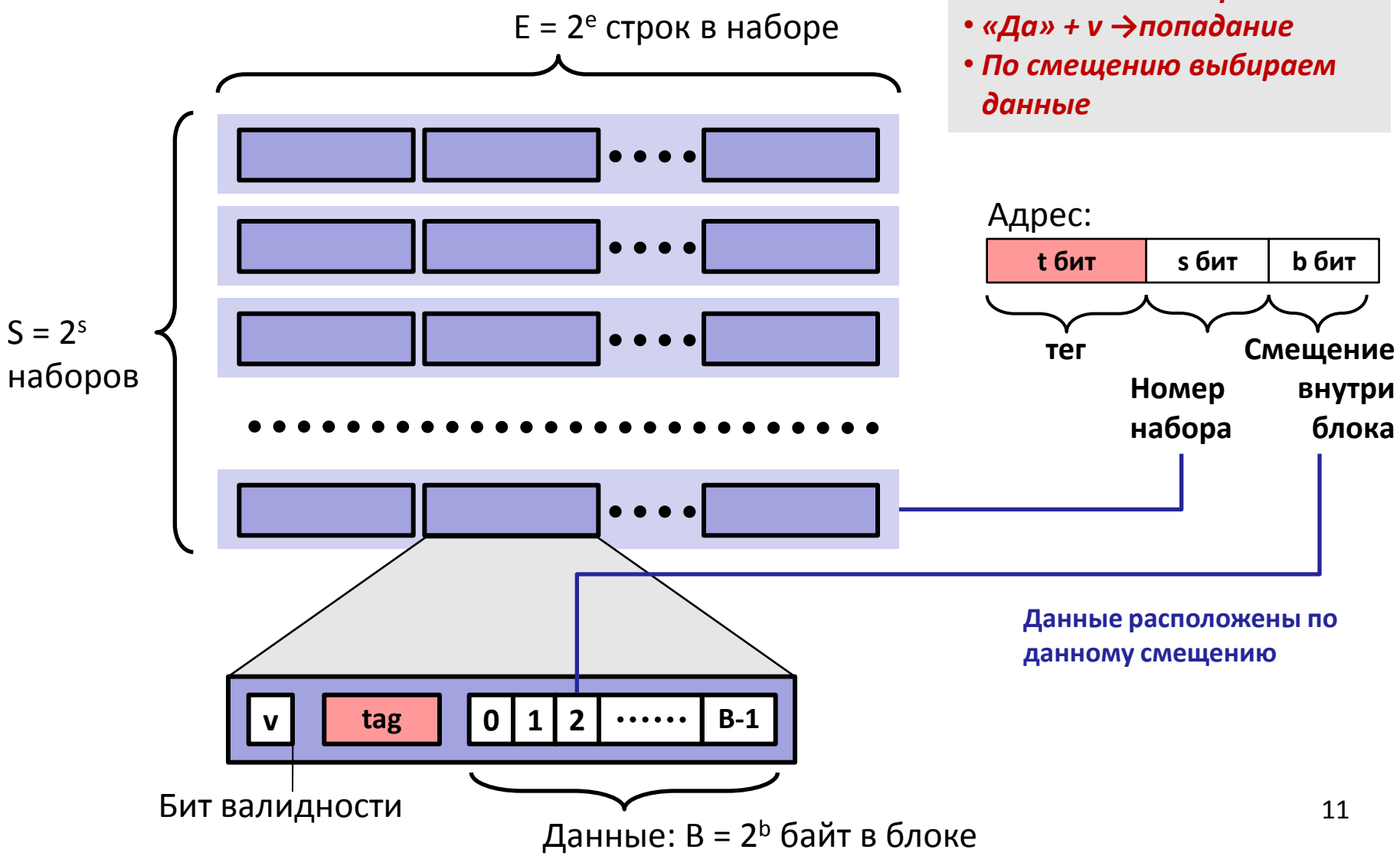
Размер кэша:

C = S x E x B байт

© 2018 МГУ/ВМК/СП

Чтение данных из кэша

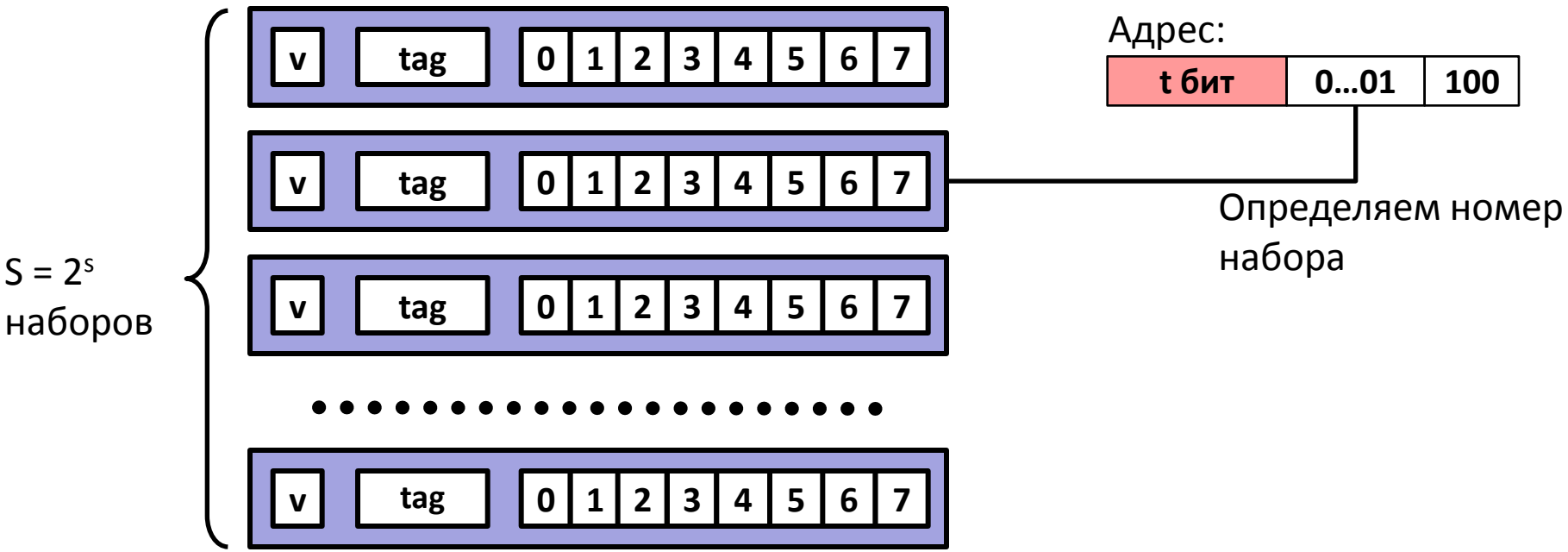
- ***Определяем номер набора***
- ***Проверяем на совпадение тега по всем строкам***
- ***«Да» + v →попадание***
- ***По смещению выбираем данные***



© 2018 МГУ/ВМК/СП

Пример: Кэш прямого отображения (E = 1)

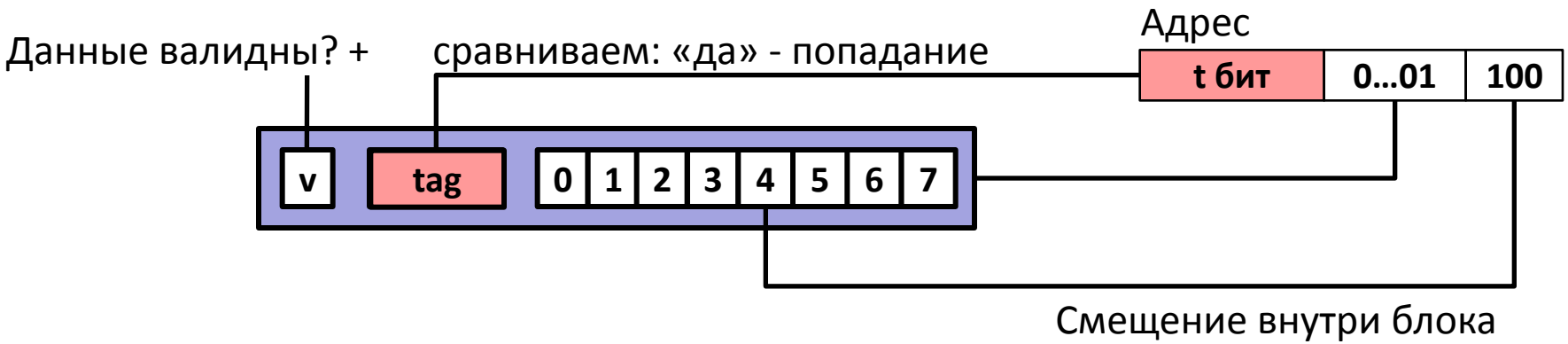
Прямое отображение: одна строка в наборе
Для данного примера: размер блока 8 байт



© 2018 МГУ/ВМК/СП

Пример: Кэш прямого отображения (E = 1)

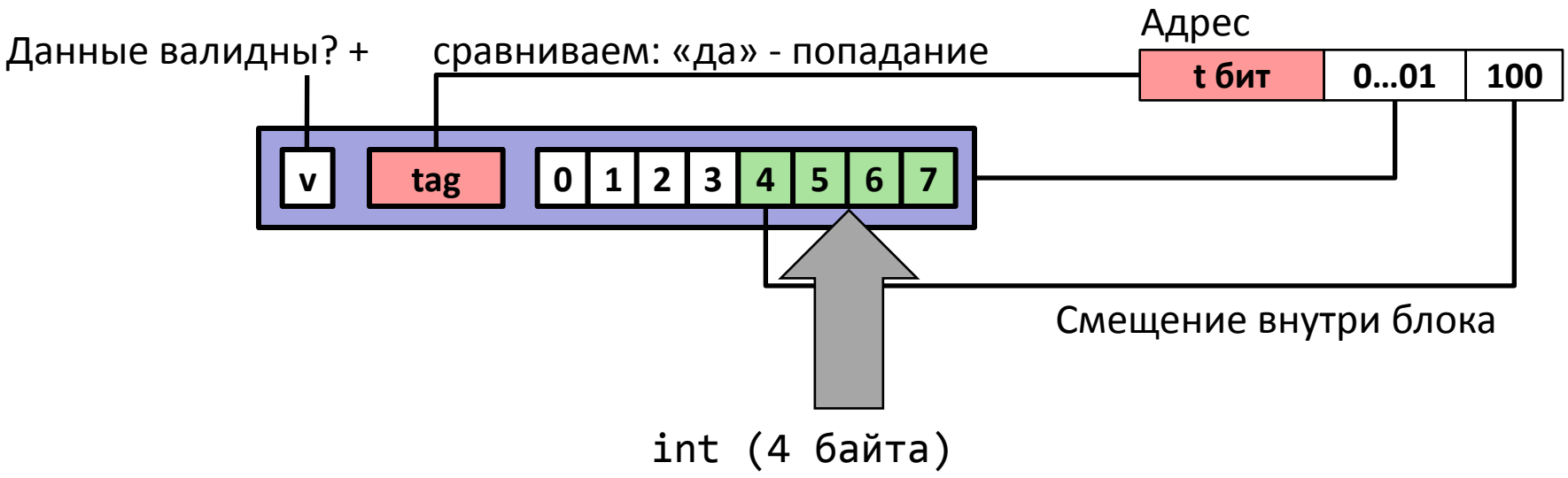
Прямое отображение: одна строка в наборе
Для данного примера: размер блока 8 байт



© 2018 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

© 2018 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

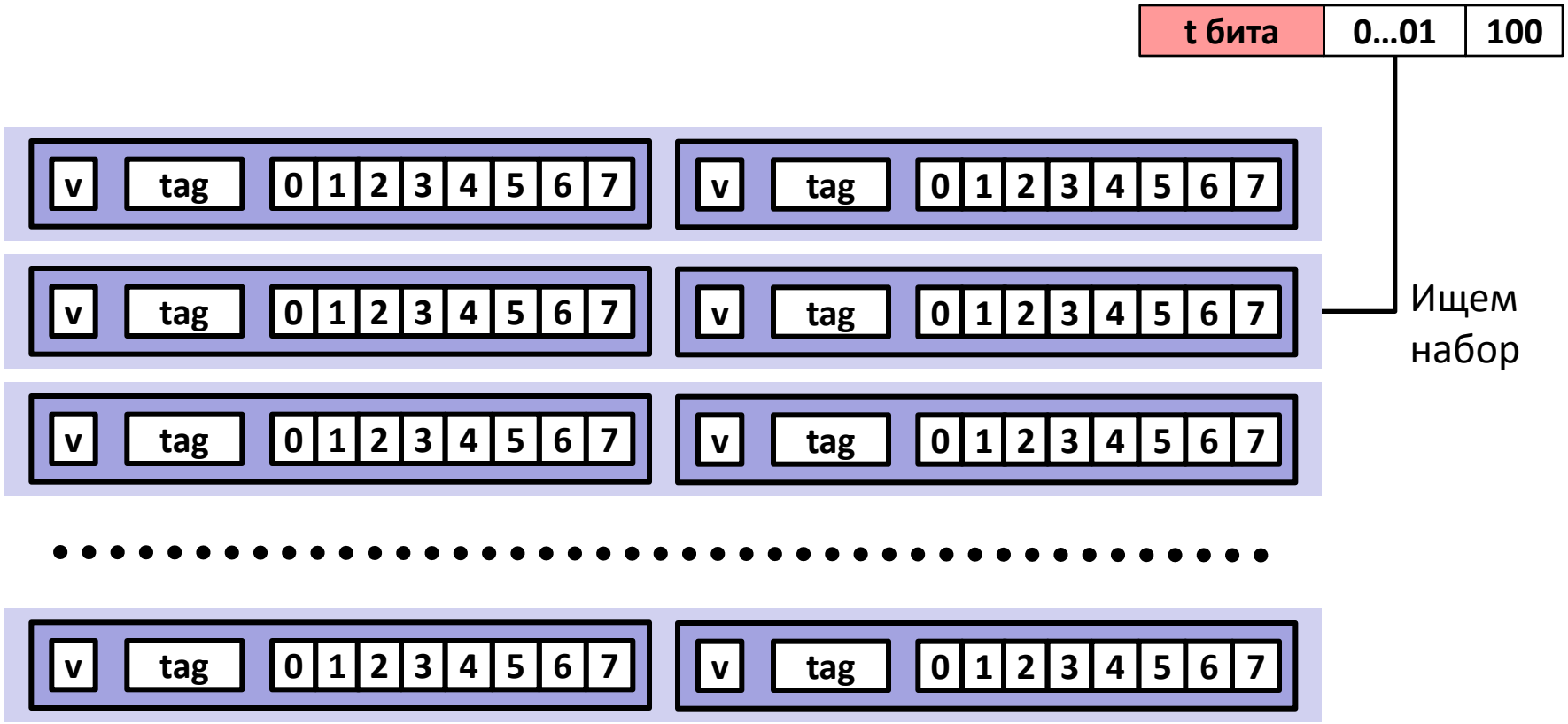
	v	Тег	Блок
Набор 0	1	0	M[0-1]
Набор 1	0	?	?
Набор 2	0	?	?
Набор 3	1	0	M[6-7]

© 2018 МГУ/ВМК/СП

N-канальный ассоциативный кэш (N = 2)

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

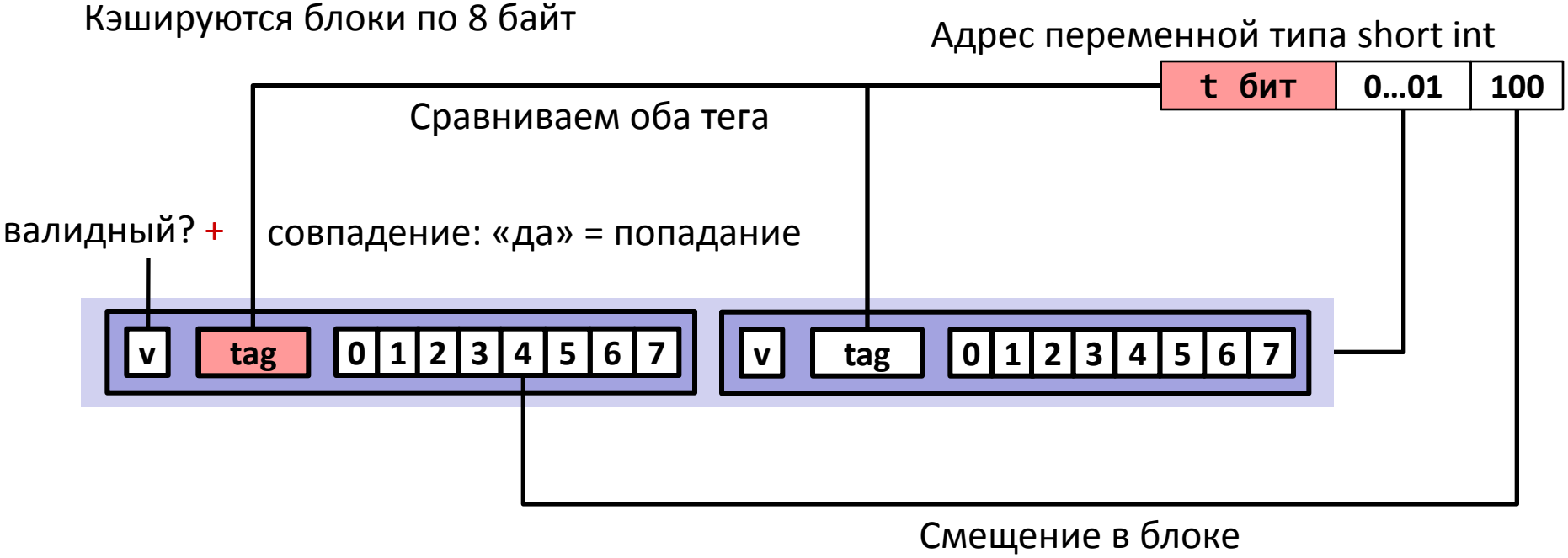
Адрес переменной типа short int



© 2018 МГУ/ВМК/СП

N-канальный ассоциативный кэш (N = 2)

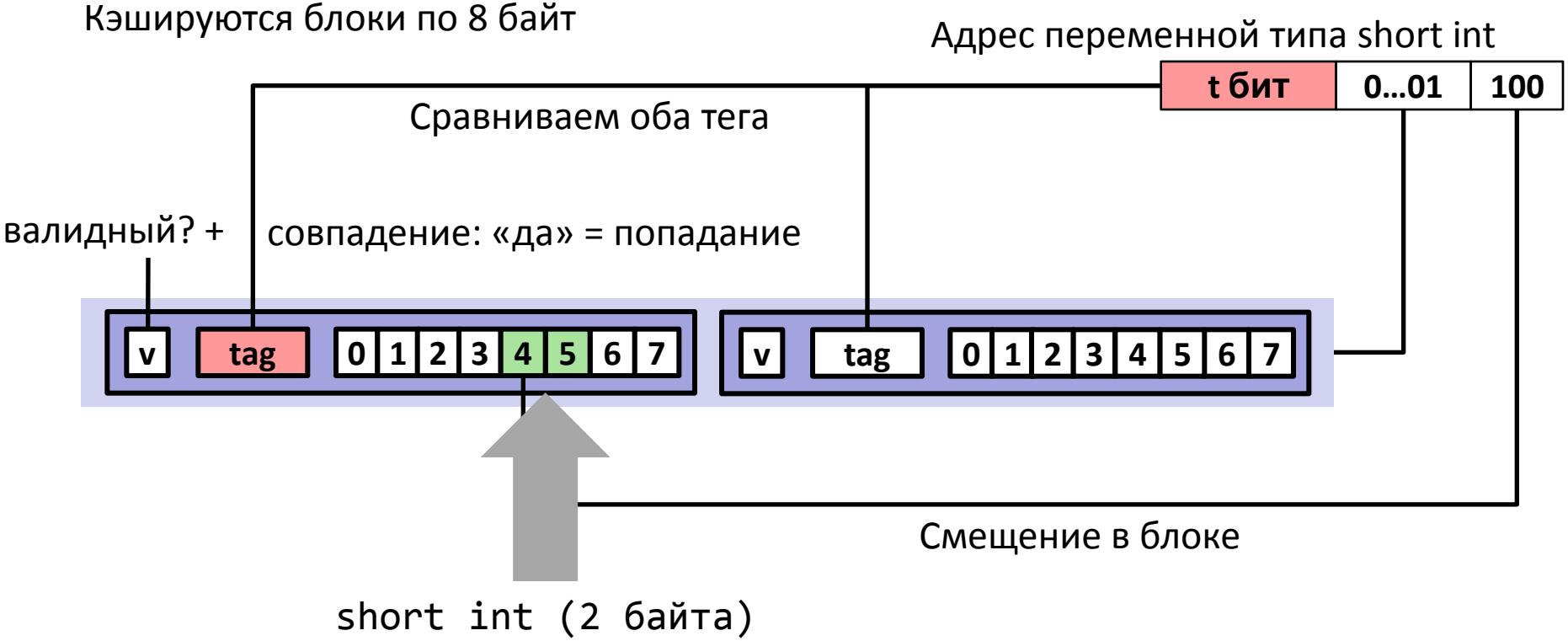
N = 2: Две строки в наборе
Кэшируются блоки по 8 байт



© 2018 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), …

© 2018 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	?	?

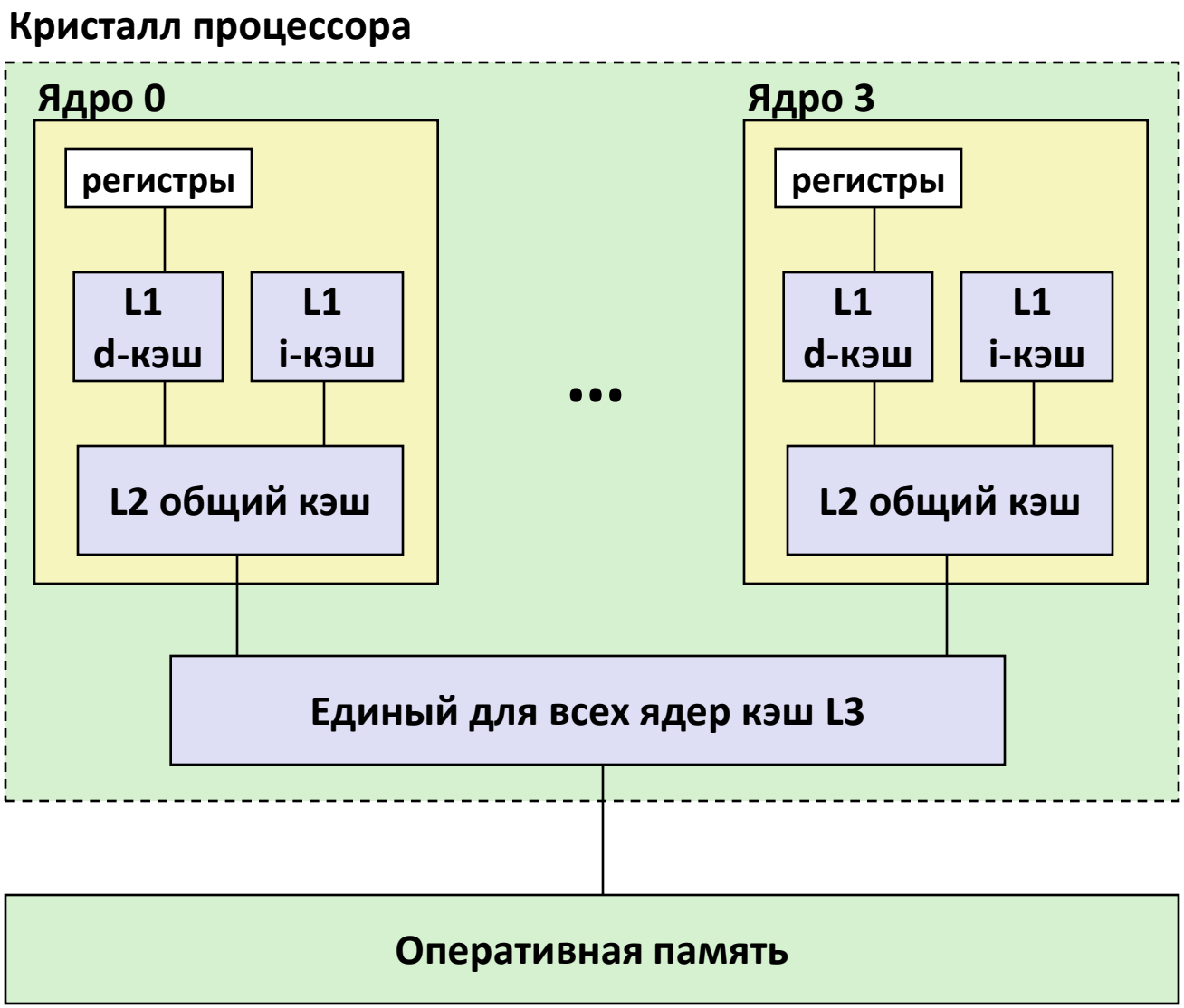
© 2018 МГУ/ВМК/СП

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

© 2018 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.

© 2018 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

© 2018 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”

© 2018 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)

© 2018 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

© 2018 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
   ...
```

© 2018 МГУ/ВМК/СП

Оценка производительности памяти: синтетический бенчмарк (контрольная задача)

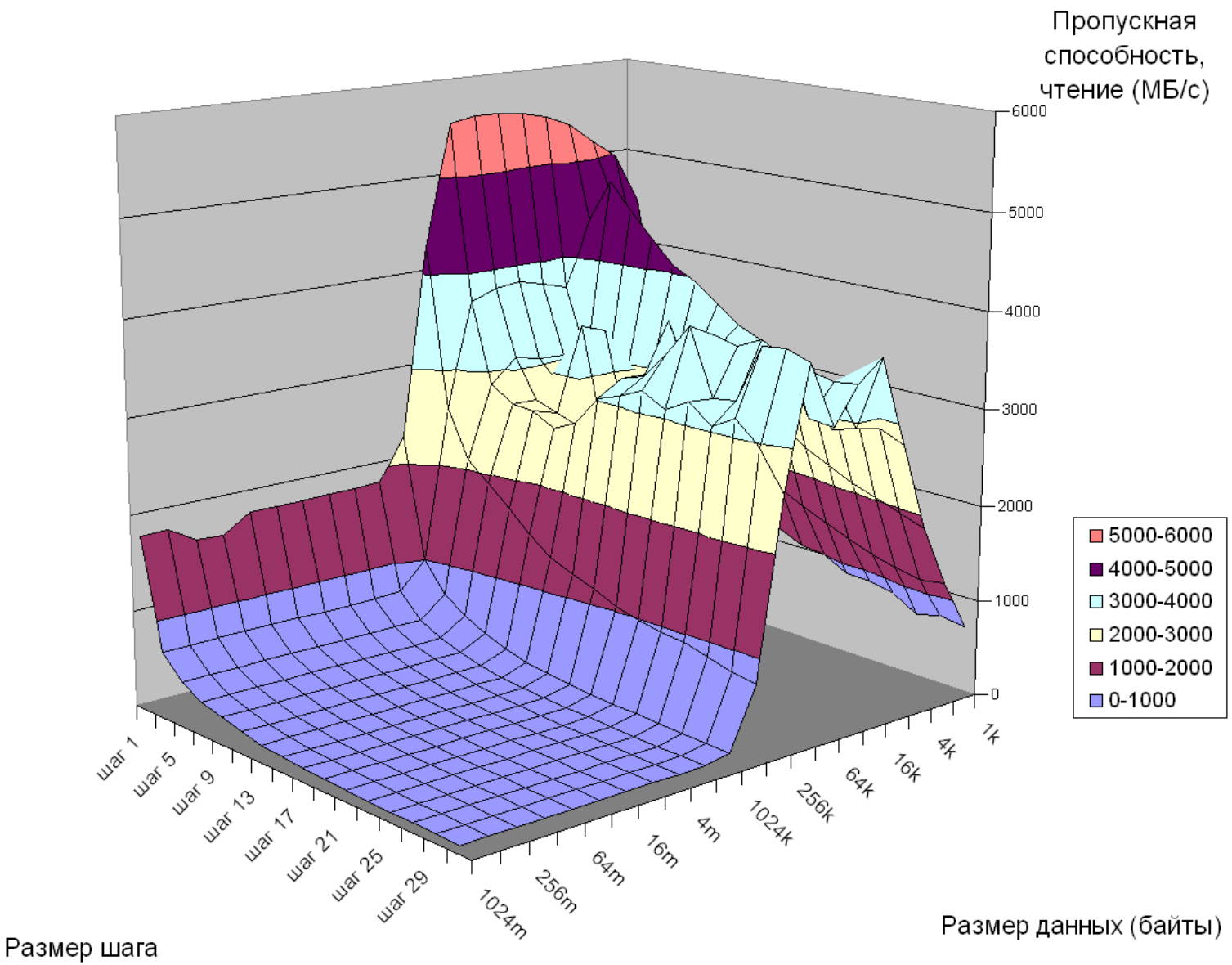
```
/* Оценочная функция */
void test(int elems, int stride) {
    int i, result = 0;
    volatile int sink;

    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
*/
}
```

© 2018 МГУ/ВМК/СП



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

© 2018 МГУ/ВМК/СП

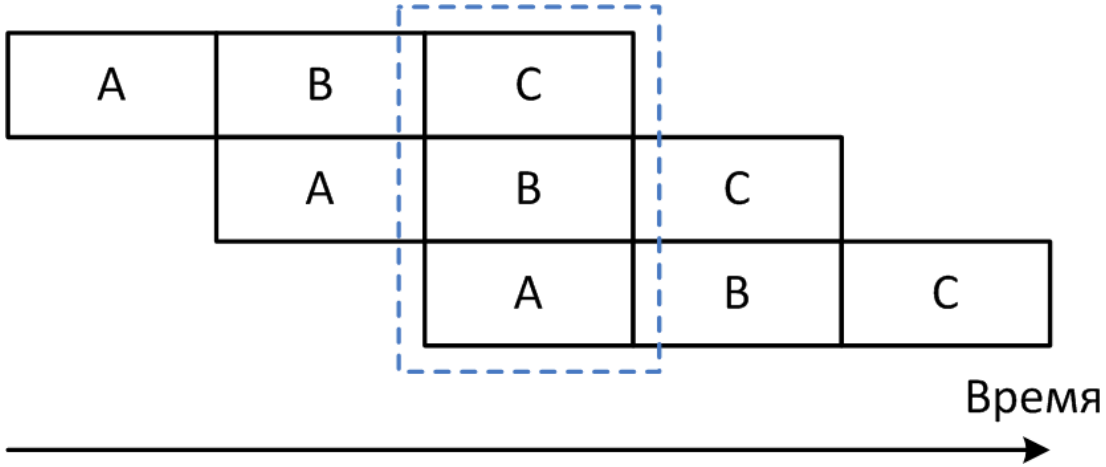
История развития x86

- 4004 – ноябрь 1971. 4-битный микропроцессор. Первый в мире коммерчески доступный однокристальный микропроцессор.
- 8008 – апрель 1972. 8080 – апрель 1974. 8-битные процессоры.
- 8086 – 1978. Размер слова – 16 бит, ширина адресной шины – 20 бит. Адреса вычисляются с использованием сегментных регистров.
- 80186 – 1982. Добавлено несколько новых инструкций.
- 80286 – 1982. Ширина адресной шины – 24 бита, добавлено устройство контроля памяти – MMU. Процессор мог переключаться между двумя режимами – реальным и защищенным.
- 80386 – октябрь 1985. Снят с производства в 2007. Размер слова – 32-разряда. Процессор мог переключаться между тремя режимами – реальным, защищенным, виртуальным. Адресуемая память – 4 ГБ.
- ...
- Intel Xeon E7-8870 – 2011, 10 ядер, до 8 ЦПУ в системе. 2.4 (2.8) ГГц; 2.2×10^9 транзисторов; системная шина QPI: 6.4 ГТ/с; обращение к памяти одного процессора: 32 ГБ/с (4x8ГБ/с), адресуемая память – 4 ТБ ; Кэш L1: 640КБ L2: 2.5МБ, L3: 30МБ.
- …
- Intel Xeon 8180M – 2017, 28 ядер, до 8 ЦПУ в системе. 2.5 (3.8) ГГц; транзисторов в аналоге от AMD 19×10^9; системная шина UPI: 10 ГТ/с; обращение к памяти одного процессора: 120 ГБ/с (6x20ГБ/с); адресуемая память – 1.5ТБ. Кэш L1: 896+896КБ, L2: 28МБ, L3: 38.5 МБ.
- …

© 2018 МГУ/ВМК/СП

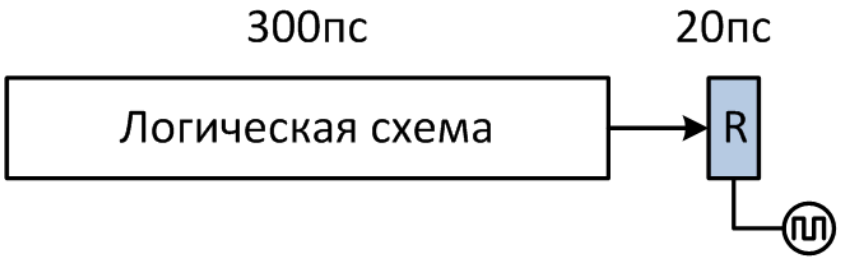
Конвейер – совмещение разных действий в один момент времени

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

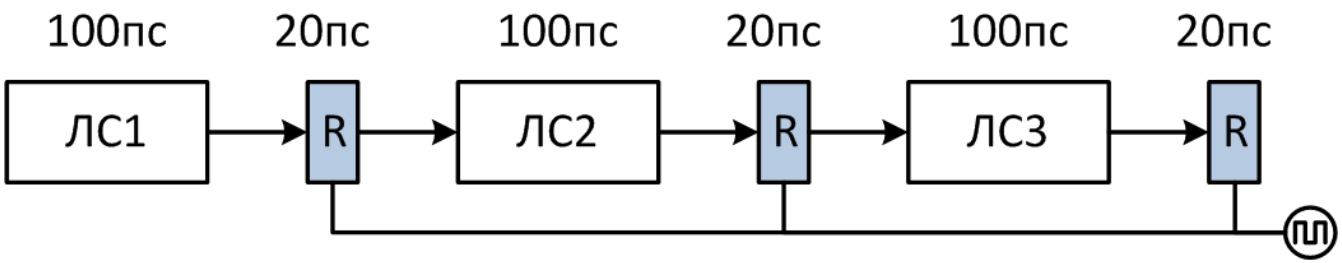


© 2018 МГУ/ВМК/СП

Организация конвейерных вычислений



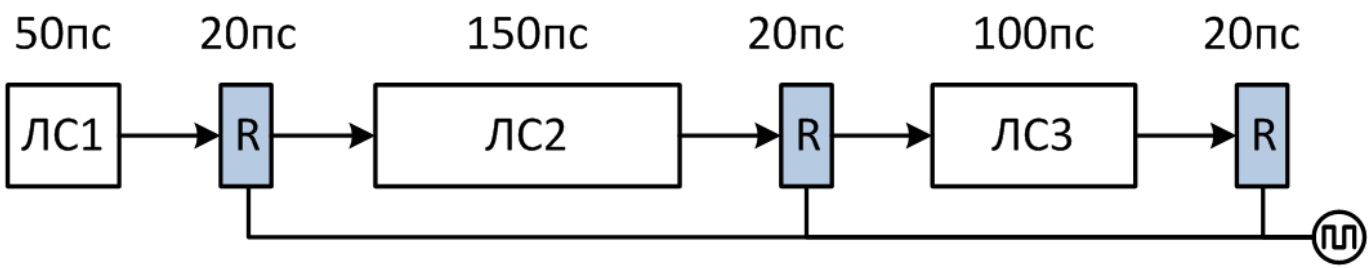
Задержка 320пс
Пропускная способность 3.12 GIPS



Задержка 360пс
Пропускная способность 8.33 GIPS

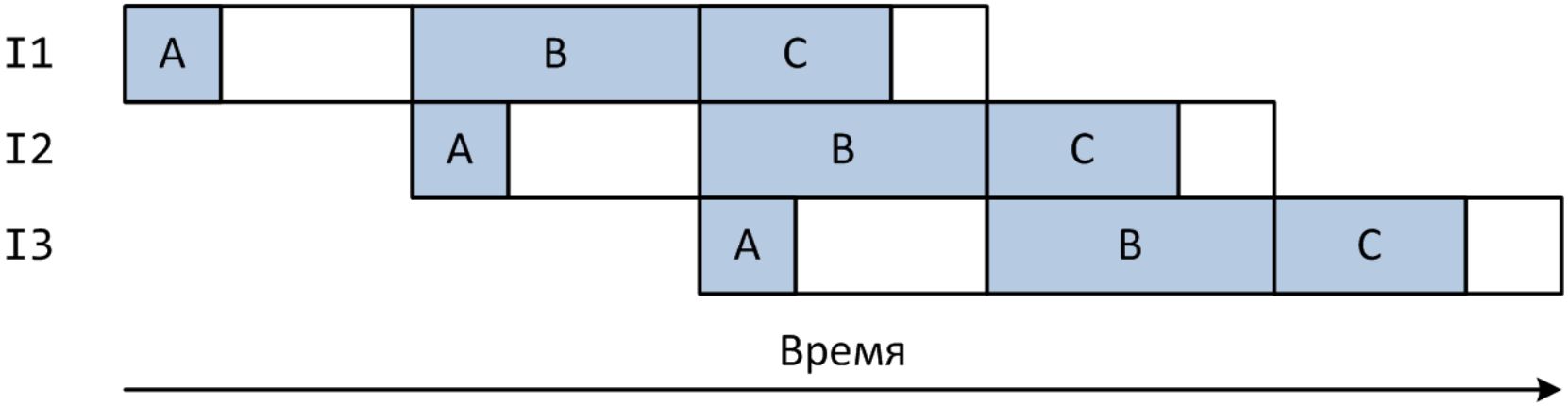
© 2018 МГУ/ВМК/СП

Неоднородность ступеней конвейера



Задержка 510пс

Пропускная способность 5.88 GIPS



© 2018 МГУ/ВМК/СП

Упрощенная схема работы конвейера x86

1. `(Fet)` Извлечение инструкции из памяти
2. `(Dec)` Декодирование, обновление EIP
3. `(D-F)` Извлечение данных, подготовка операндов к выполнению операции
4. `(Exe)` Непосредственное выполнение операции
5. `(Wrt)` Запись результата

- Для IA-32 такое модельное деление конвейера на этапы (стадии) обусловлено тем, что почти *каждая* команда способна работать с операндами различных типов: регистрами, константами и *даже* ячейками памяти.
- Определение исполнительного адреса операнда, размещенного в памяти, в общем случае требует получения значений двух регистров и двух констант, одного умножения и двух сложений.

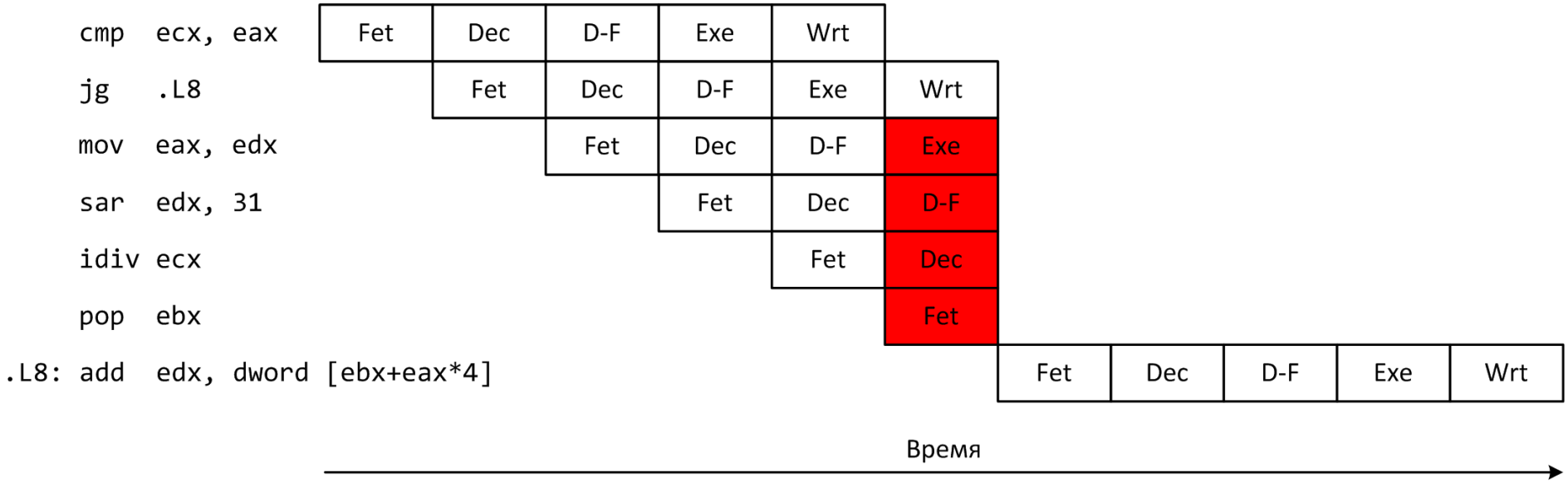
© 2018 МГУ/ВМК/СП

Проблемы конвейерной организации

- Опустошение конвейера: сброс промежуточных результатов уже выполняющихся в конвейере команд
  - Современные процессоры могут содержать конвейеры длины 15 и более (P4 Prescott – 31 стадия конвейера)
  - Изменение EIP сбрасывает промежуточные результаты выполнения следующих инструкций
  - Возникновение исключительной ситуации при обращении к памяти или при выполнении операции над данными
- Зависимости по данным между близко расположенными командами
- Остановки из-за обращения к памяти

© 2018 МГУ/ВМК/СП

Опустошение конвейера при передаче управления

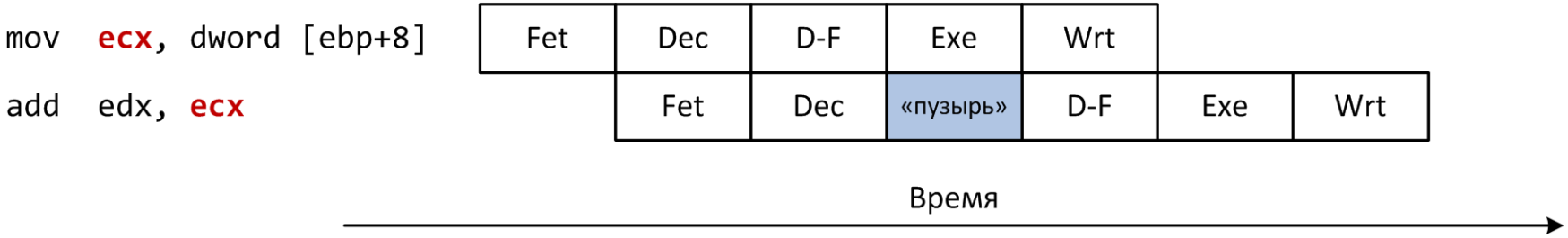


© 2018 МГУ/ВМК/СП

Приостановка конвейера

Некоторые этапы не удается выполнить за один такт.

Одновременное выполнение нескольких команд требует учета зависимостей по данным между командами: нельзя извлекать из регистра значение, если предыдущая команда его туда еще не записала.

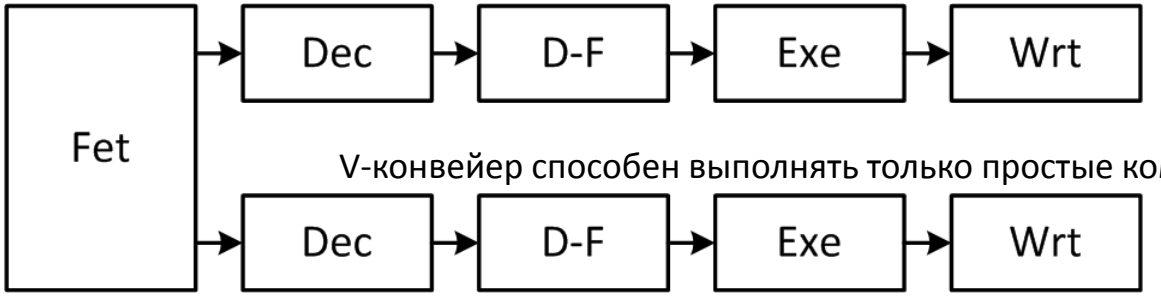


© 2018 МГУ/ВМК/СП

Двойной конвейер

- Двойной конвейер с общим этапом выборки команд
  - (Fet) Извлечение инструкции из памяти
  - (Dec) Декодирование, обновление EIP
  - (D-F) Извлечение данных
  - (Exe) Непосредственное выполнение операции
  - (Wrt) Запись результата

U-конвейер способен выполнять любые команды

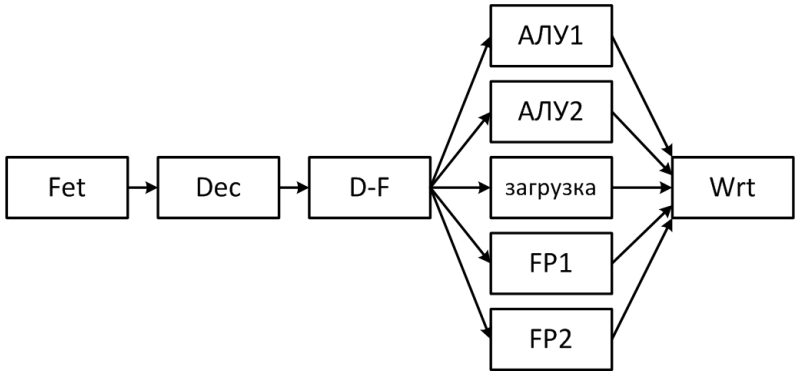


V-конвейер способен выполнять только простые команды, с «оглядкой» на состояние U-конвейера

- Впервые появился на Pentium: введены u- и v-конвейеры
- Возможности конвейеров неравноценны

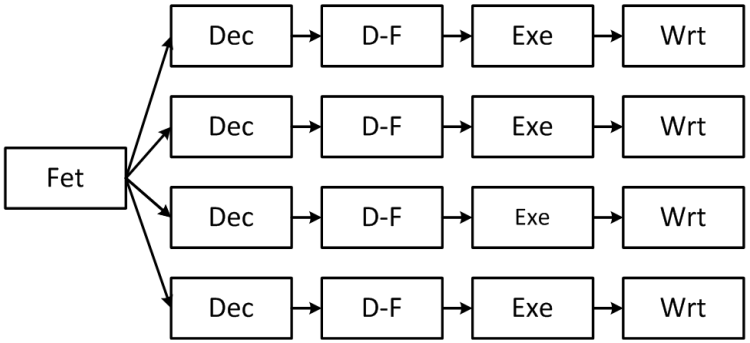
© 2018 МГУ/ВМК/СП

Суперскалярная и VLIW архитектуры



- Многие операции над данными не могут быть реализованы за один такт.
- В суперскалярной архитектуре выполняющиеся команды на стадии операции расходятся по нескольким *функциональным устройствам (ФУ)*.
- За распределение (планирование) потока команд по ФУ отвечает аппаратура процессора.

- Во VLIW-архитектуре (Very Long Instrictoin Word) каждая команда состоит из нескольких микрокоманд, независимо друг от друга, выполняющих операции над данными.
- За то, что бы команды действительно не конфликтовали между собой и могли выполняться параллельно, отвечает не процессор, а компилятор.



© 2018 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)

© 2018 МГУ/ВМК/СП

RISC vs CISC

(Reduced vs Complex Instruction Set Computer)

- Модель работы с памятью
  - Отдельные команды для загрузки/сохранения в память Альтернативное название RISC-архитектур – Load/Store-машина
  - Команды обработки данных работают только с регистрами
- Результат
  - Устройство ядра процессора становится проще
  - Частота процессора выше
  - По сравнению с CISC объем кода (число команд) при реализации того же алгоритма на RISC-машине будет больше
- Сложность оптимизаций перенесена из процессора в компилятор
  - Производительность сильно зависит от компилятора