

Глава 8

Цифровая схемотехника и архитектура компьютера, второе издание

Дэвид Мани Харрис и Сара Л.
Харрис

Цифровая схемотехника и архитектура компьютера

Эти слайды предназначены для преподавателей, которые читают лекции на основе учебника «Цифровая схемотехника и архитектура компьютера» авторов Дэвида Харриса и Сары Харрис. Бесплатный русский перевод второго издания этого учебника можно загрузить с сайта компании Imagination Technologies:

<https://community.imgtec.com/downloads/digital-design-and-computer-architecture-russian-edition-second-edition>

Процедура регистрации на сайте компании Imagination Technologies описана на странице:

<http://www.silicon-russia.com/2016/08/04/harris-and-harris-2/>



Благодарности

Перевод данных слайдов на русский язык был выполнен командой сотрудников университетов и компаний из России, Украины, США в составе:

- Александр Барабанов - доцент кафедры компьютерной инженерии факультета радиофизики, электроники и компьютерных систем Киевского национального университета имени Тараса Шевченко, кандидат физ.-мат. наук, Киев, Украина;
- Антон Брюзгин - начальник отдела АО «Вибро-прибор», Санкт-Петербург, Россия.
- Евгений Короткий - доцент кафедры конструирования электронно-вычислительной аппаратуры факультета электроники Национального технического университета Украины «Киевский Политехнический Институт», руководитель открытой лаборатории электроники Lamp, кандидат технических наук, Киев, Украина;
- Евгения Литвинова – заместитель декана факультета компьютерной инженерии и управления, доктор технических наук, профессор кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, Харьков, Украина;
- Юрий Панчул - старший инженер по разработке и верификации блоков микропроцессорного ядра в команде MIPS I6400, Imagination Technologies, отделение в Санта-Кларе, Калифорния, США;
- Дмитрий Рожко - инженер-программист АО «Вибро-прибор», магистр Санкт-Петербургского государственного автономного университета аэрокосмического приборостроения (ГУАП), Санкт-Петербург, Россия;
- Владимир Хаханов – декан факультета компьютерной инженерии и управления, проректор по научной работе, доктор технических наук, профессор кафедры автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, Харьков, Украина;
- Светлана Чумаченко – заведующая кафедрой автоматизации проектирования вычислительной техники Харьковского национального университета радиоэлектроники, доктор технических наук, профессор, Харьков, Украина.



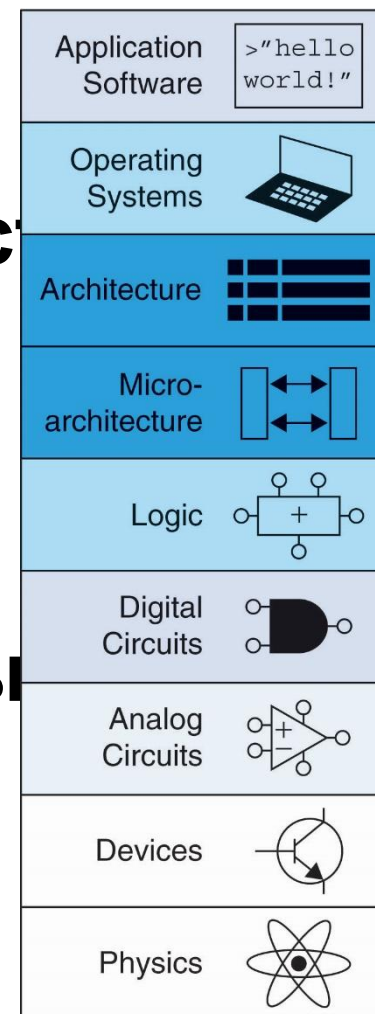
Глава 8 :: Темы

ИЕРАРХИЯ ПАМЯТИ И

ПОДСИСТЕМА

ВВОДА-ВЫВОДА

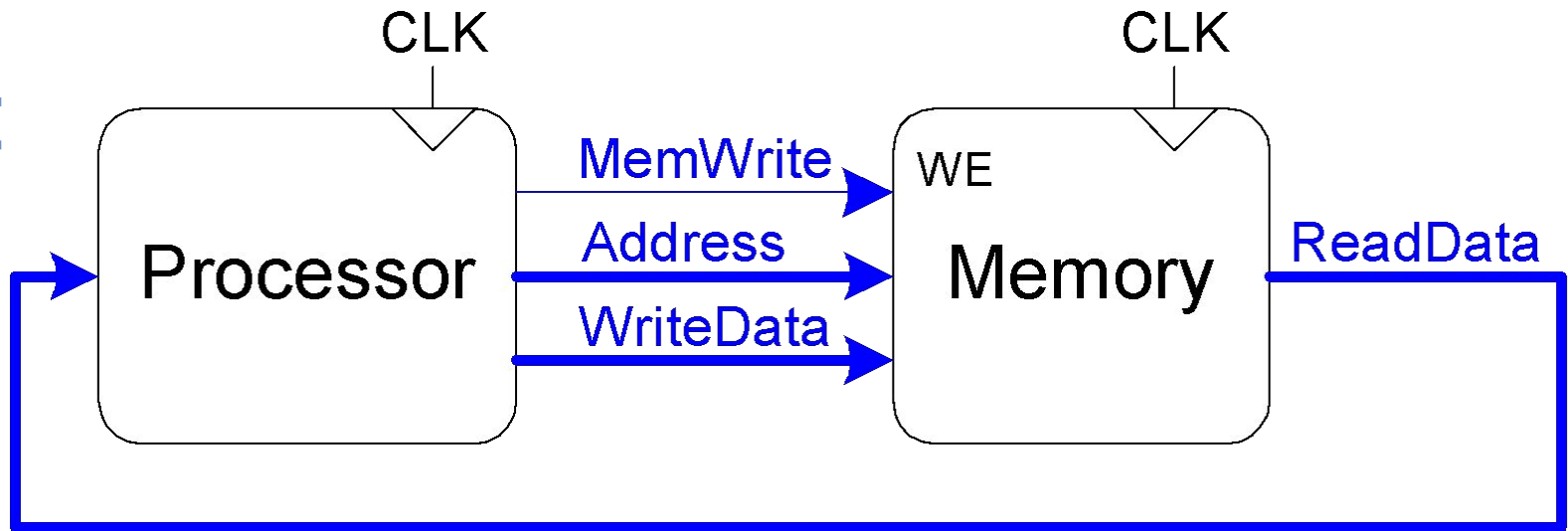
- Введение
- Анализ производительности систем памяти
- Кэш-память
- Виртуальная память
- Ввод-вывод, отображённый в память
- Резюме



Введение

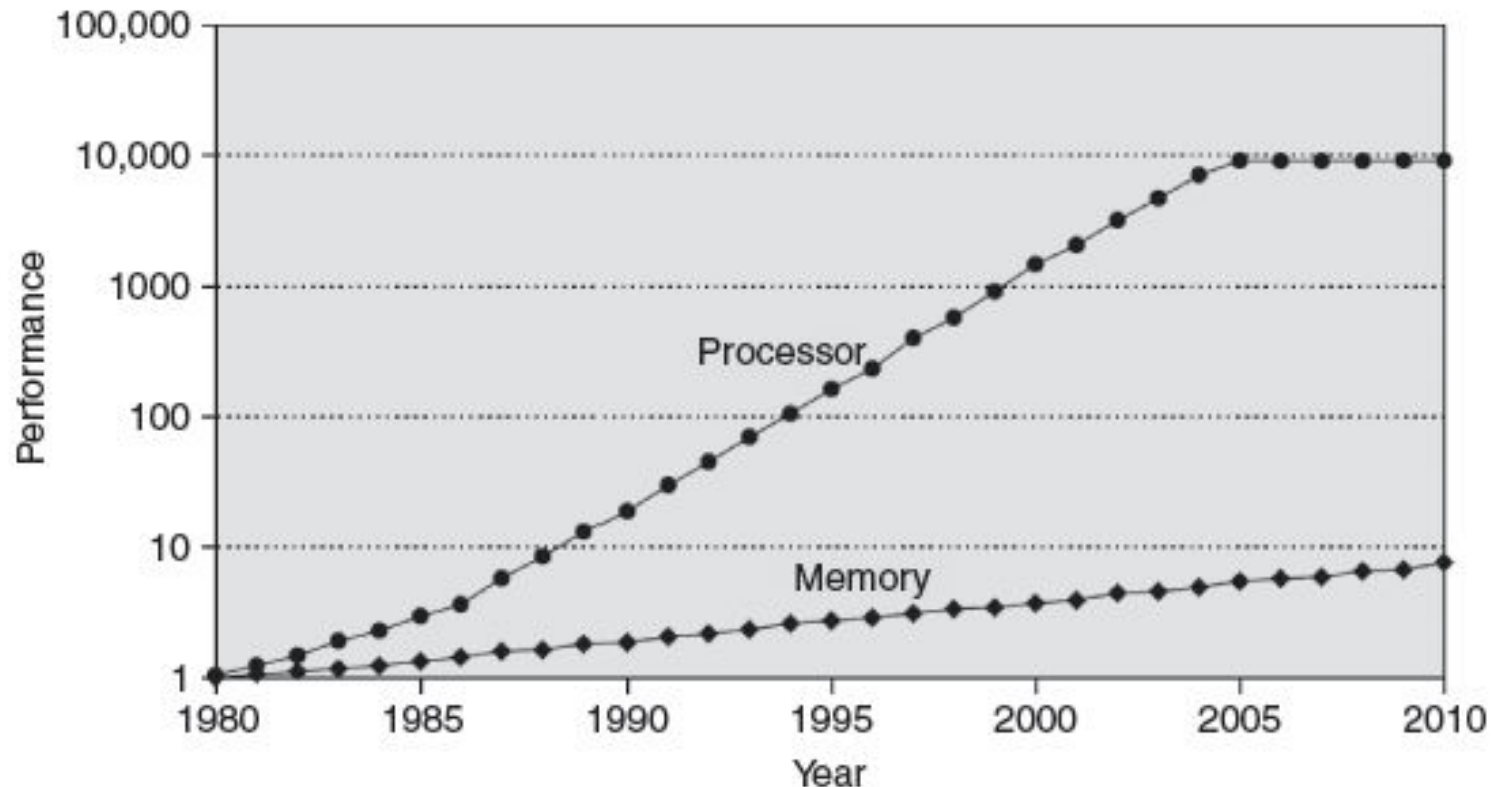
- Производительность компьютера зависит от:
 - Производительности процессора
 - Производительности подсистемы памяти

Интерфейс памяти



Разрыв между процессором и

В предыдущих главах, предполагалось, что доступ к памяти осуществляется за 1 такт, но это не было верным уже с 1980-х годов



Проблема подсистемы

ИЕРАРХИЯ ПАМЯТИ И
ПОДСИСТЕМА
ВВОДА-ВЫВОДА

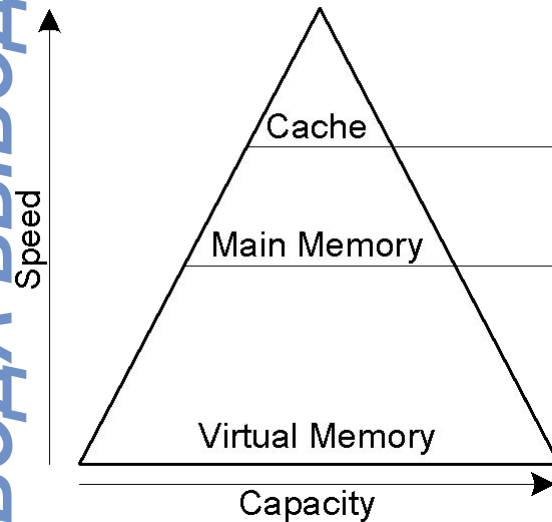
- Сделать подсистему памяти такой же быстрой, как процессор
- Использовать иерархию памяти
- Идеальная память:
 - Быстрая
 - Дешёвая (недорогая)
 - Большая (ёмкая)

Можно выбрать только два!



Иерархия памяти

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА



Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	\$10,000	1	25+
DRAM	\$10	10 - 50	10
SSD	\$1	100,000	0.5
HDD	\$0.1	10,000,000	0.1

Локальность

Используйте локальность для того, чтобы сделать доступ к памяти более быстрым

Временная локальность:

- Локальность во времени
- Если данные использовались недавно, то вероятно они скоро понадобятся снова
- **Как это использовать:** держать недавно использованные данные на более высоких уровнях иерархии памяти

Пространственная локальность:

- Локальность в пространстве
- Если данные использовались недавно, то вероятно скоро понадобятся данные поблизости
- **Как это использовать:** при доступе к данным переносить также близлежащие данные на более высокие уровни иерархии памяти



Производительность памяти

- **Попадания:** данные найдены на этом уровне иерархии памяти
- **Прوماхи:** данные не найдены на этом уровне иерархии памяти (нужно перейти на следующий уровень)

Процент попаданий = количество попаданий / количество доступов к памяти = 1 – процент промахов

Процент промахов = количество промахов / количество доступов к памяти = 1 – процент попаданий

- **Среднее время доступа (англ. Average memory access time, AMAT):** среднее время, которое процессор тратит на доступ к памяти

$$\text{AMAT} = t_{\text{cache}} + MR_{\text{cache}}[t_{MM} + MR_{MM}(t_{VM})]$$

Пример производительности

памяти 1

ИЕРАРХИЯ ПАМЯТИ И
ПОДСИСТЕМА
ВВОДА-ВЫВОДА

- Программа имеет 2000 операций загрузки и сохранения
- 1250 из них нашли данные в кэш-памяти
- Остальные данные находятся на других уровнях иерархии памяти
- **Чему равен процент промахов и попаданий в кэш-память?**

Пример производительности

памяти 1

ИЕРАРХИЯ ПАМЯТИ И
ПОДСИСТЕМА
ВВОДА-ВЫВОДА

- Программа имеет 2000 операций загрузки и сохранения
- 1250 из них нашли данные в кэш-памяти
- Остальные данные находятся на других уровнях иерархии памяти
- **Чему равен процент промахов и попаданий в кэш-память?**

Процент попаданий = $1250/2000 = 0.625$

Процент промахов = $750/2000 = 0.375 = 1 -$
процент попаданий

Пример производительности

памяти 2

ИЕРАРХИЯ ПАМЯТИ И
ПОДСИСТЕМА
ВВОДА-ВЫВОДА

- Предположим, что процессор имеет 2 уровня иерархии: кэш-память и оперативную память

$$t_{\text{cache}} = 1 \text{ цикл}, t_{MM} = 100 \text{ циклов}$$

- **Чему равно среднее время доступа для программы из примера 1?**

Пример производительности

памяти 2

- Предположим, что процессор имеет 2 уровня иерархии: кэш-память и оперативную память

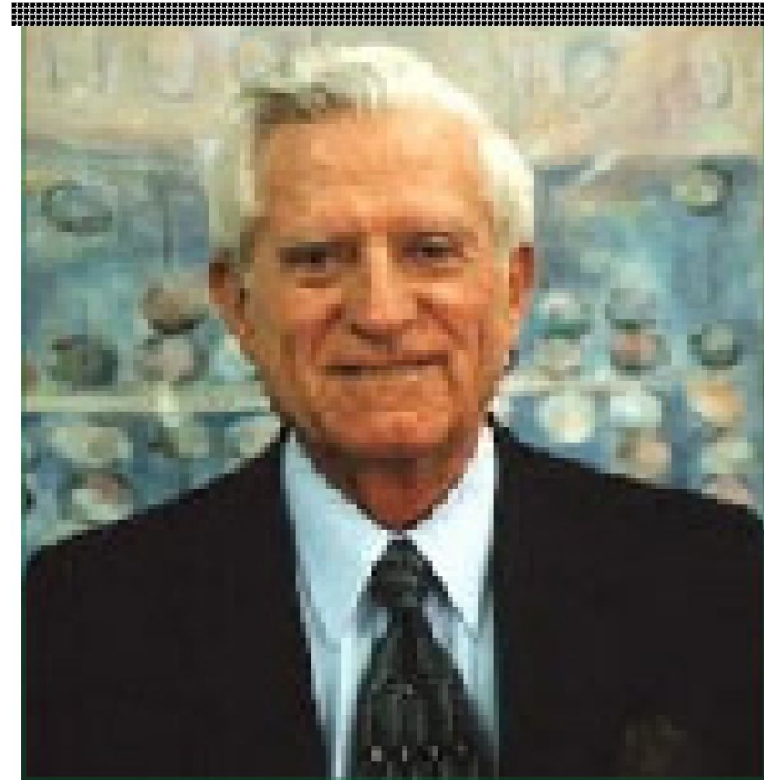
$$t_{\text{cache}} = 1 \text{ цикл}, t_{MM} = 100 \text{ циклов}$$

- **Чему равно среднее время доступа для программы из примера 1?**

$$\begin{aligned} \text{AMAT} &= t_{\text{cache}} + MR_{\text{cache}}(t_{MM}) \\ &= [1 + 0.375(100)] \text{ циклов} \\ &= \mathbf{38.5 \text{ циклов}} \end{aligned}$$

Джин Амдал, 1922-2015

- **Закон Амдала:** усилия, потраченные на улучшение производительности подсистемы, оправдываются только тогда, когда она оказывает значительное влияние на общую производительность системы
- Основал 3 компании, одну из которых назвал Amdahl Corporation в 1970 году



Кэш-память

- Наивысший уровень в иерархии памяти
- Быстрая (обычно время доступа ≈ 1 такт)
- В идеале предоставляет бóльшую часть данных процессору
- Обычно содержит последние использованные данные

Вопросы проектирования кэш-

- Какие данные хранятся в кэш-памяти?
- Как найти данные?
- Какие данные заместить?

Сосредоточьтесь на загрузке данных, а сохранение производите по тем же принципам

Какие данные хранятся в кэш-

- В идеале, процессор предугадывает какие данные потребуются и помещает их в кэш
- Но невозможно предсказать будущее
- Используйте прошлое, чтобы предсказать будущее – временную и пространственную локальность
 - **Временная локальность:** копировать часто используемые данные в кэш-память
 - **Пространственная локальность:** копировать также рядом лежащие данные в кэш-память

Терминология кэш-памяти

- **Ёмкость (C):**

- количество байт данных, которое может поместиться в кэш-памяти

- Размер строк (b):**

- количество байт данных, заносимое в кэш-память одновременно

- Количество строк ($B = C/b$):**

- количество строк в кэш-памяти: $B = C/b$

- Степень ассоциативности (N):**

- количество строк в наборе

- Количество наборов ($S = B/N$):**

- каждый адрес памяти отображается только в один набор кэша

Как данные найти?

- Кэш-память состоит из S наборов
- Каждый адрес памяти отображается только в один набор кэша
- По количеству строк в наборе кэш делится на:
 - **Прямого отображения:** 1 строка в наборе
 - **Наборно-ассоциативный кэш с N секциями:** N строк в наборе
 - **Полностью ассоциативный:** все строки кэш-памяти в одном наборе

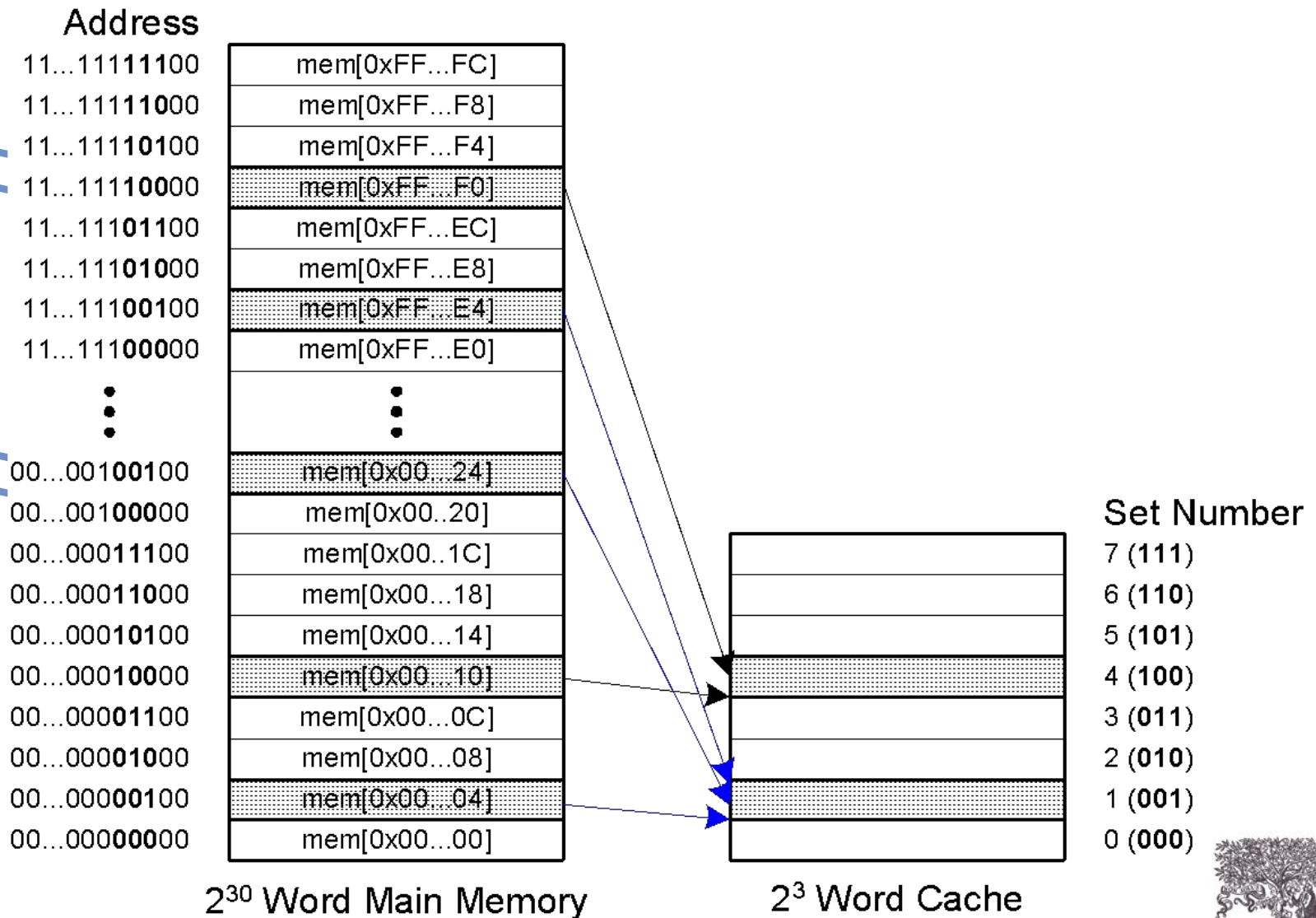
Пример параметров кэш-памяти

- $C = 8$ слов (ёмкость)
- $b = 1$ слово (размер строки)
- Тогда, $B = 8$ (количество строк)

Нелепо небольшой, но иллюстрирует организацию

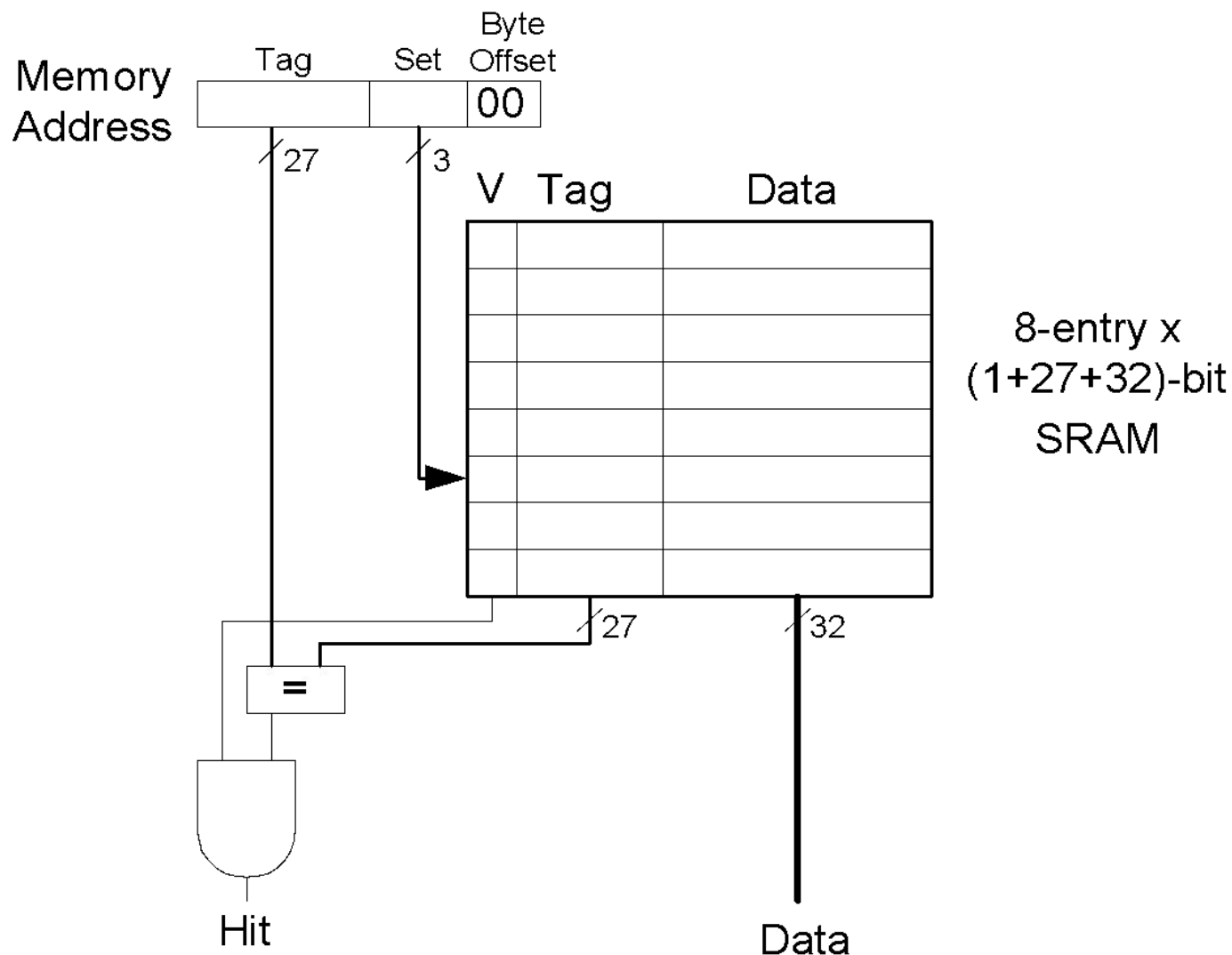
Кэш прямого отображения

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА



Аппаратная реализация кэша прямого отображения

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА



Производительность кэша прямого отображения

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА

MIPS код

```
addi $t0, $0, 5
loop: beq $t0, $0, done
      lw  $t1, 0x4($0)
      lw  $t2, 0xC($0)
      lw  $t3, 0x8($0)
      addi $t0, $t0, -1
      j   loop
done:
```

Memory Address	Tag	Set	Byte Offset
	00...00	001	00

3

V	Tag	Data	
0			Set 7 (111)
0			Set 6 (110)
0			Set 5 (101)
0			Set 4 (100)
1	00...00	mem[0x00...0C]	Set 3 (011)
1	00...00	mem[0x00...08]	Set 2 (010)
1	00...00	mem[0x00...04]	Set 1 (001)
0			Set 0 (000)

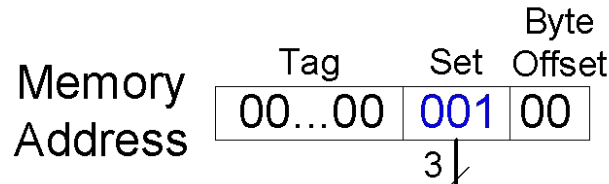
Процент промахов = ?

Производительность кэша прямого отображения

MIPS код

```

addi $t0, $0, 5
loop: beq  $t0, $0, done
      lw   $t1, 0x4($0)
      lw   $t2, 0xC($0)
      lw   $t3, 0x8($0)
      addi $t0, $t0, -1
      j    loop
done:
    
```



V	Tag	Data	
0			Set 7 (111)
0			Set 6 (110)
0			Set 5 (101)
0			Set 4 (100)
1	00...00	mem[0x00...0C]	Set 3 (011)
1	00...00	mem[0x00...08]	Set 2 (010)
1	00...00	mem[0x00...04]	Set 1 (001)
0			Set 0 (000)

**Процент промахов = 3/15
= 20%**

**Временная локальность
Обязательные промахи**



Кэш прямого отображения:

Конфликты

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА

MIPS код

```

addi $t0, $0, 5
loop: beq $t0, $0, done
      lw  $t1, 0x4($0)
      lw  $t2, 0x24($0)
      addi $t0, $t0, -1
      j   loop
done:
    
```

Memory Address	Tag	Set	Byte Offset
	00...01	001	00

3

V Tag Data

0		
0		
0		
0		
0		
0		
0		
1	00...00	mem[0x00...04] mem[0x00...24]
0		

Set 7 (111)
Set 6 (110)
Set 5 (101)
Set 4 (100)
Set 3 (011)
Set 2 (010)
Set 1 (001)
Set 0 (000)

Процент промахов = ?

Кэш прямого отображения:

Конфликты

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА

MIPS код

```
addi $t0, $0, 5
loop: beq $t0, $0, done
      lw  $t1, 0x4($0)
      lw  $t2, 0x24($0)
      addi $t0, $t0, -1
      j   loop
done:
```

Memory Address	Tag	Set	Byte Offset
	00...01	001	00

3

V Tag Data

V	Tag	Data
0		
0		
0		
0		
0		
0		
0		
1	00...00	mem[0x00...04] mem[0x00...24]
0		

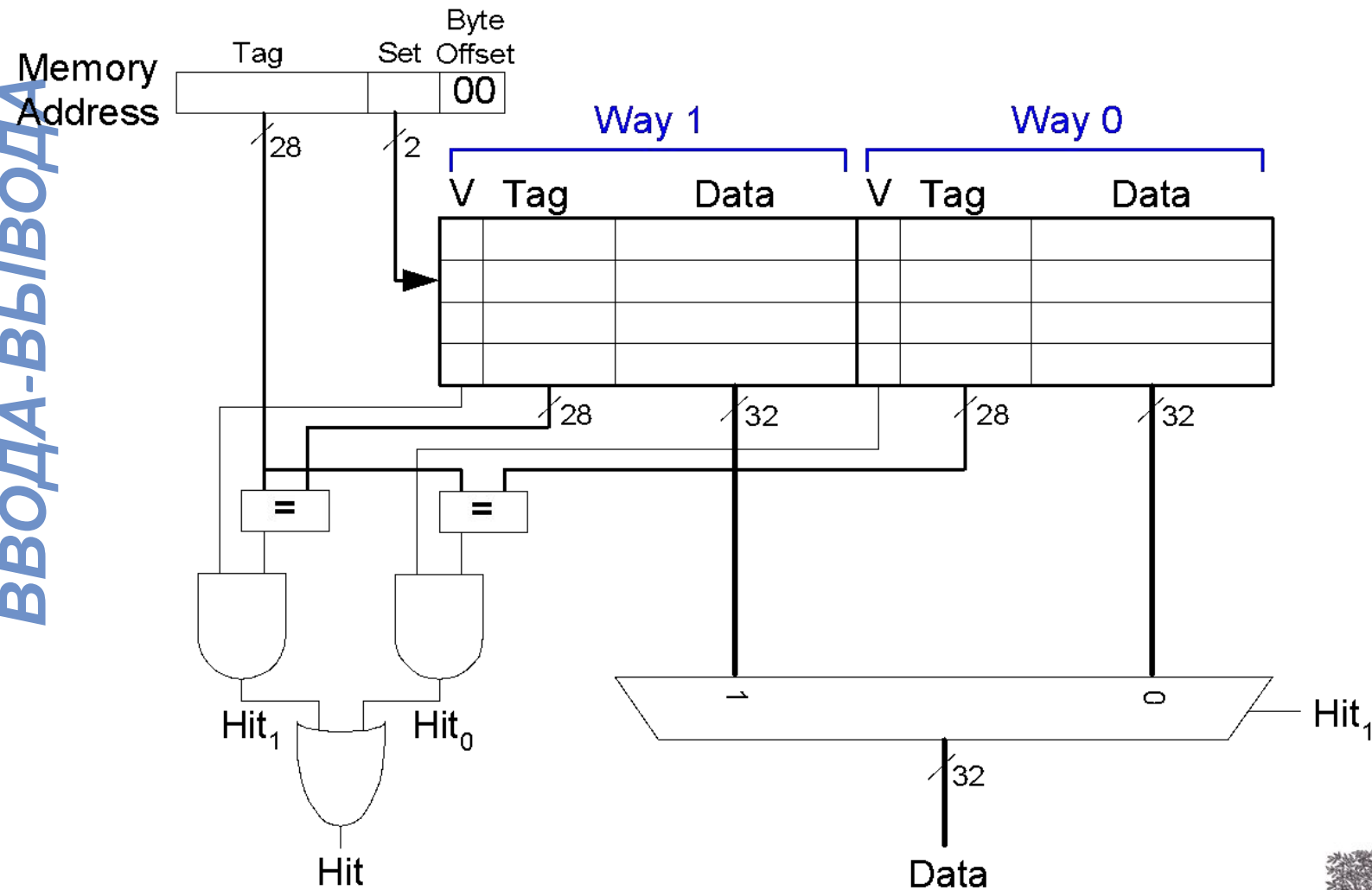
Set 7 (111)
Set 6 (110)
Set 5 (101)
Set 4 (100)
Set 3 (011)
Set 2 (010)
Set 1 (001)
Set 0 (000)

Процент промахов = 10/10
= 100%

Прوماхи из-за конфликтов



Наборно-ассоциативный кэш с N



Производительность наборно-ассоциативного кэша с N секциями

MIPS код

```
addi $t0, $0, 5
loop: beq $t0, $0, done
      lw  $t1, 0x4($0)
      lw  $t2, 0x24($0)
      addi $t0, $t0, -1
      j   loop
done:
```

Процент промахов = ?

Way 1			Way 0			
V	Tag	Data	V	Tag	Data	
0			0			Set 3
0			0			Set 2
0			0			Set 1
0			0			Set 0

Производительность наборно-ассоциативного кэша с N секциями

MIPS код

```
addi $t0, $0, 5
loop: beq $t0, $0, done
      lw  $t1, 0x4($0)
      lw  $t2, 0x24($0)
      addi $t0, $t0, -1
      j   loop
done:
```

Процент промахов = $2/10$
= 20%

Ассоциативность уменьшает количество промахов из-за конфликтов

Way 1			Way 0		
V	Tag	Data	V	Tag	Data
0			0		
0			0		
1	00...10	mem[0x00...24]	1	00...00	mem[0x00...04]
0			0		

Set 3
Set 2
Set 1
Set 0



Полностью ассоциативный

ИЕРАРХИЯ ПАМЯТИ И
ПОДСИСТЕМА
ВВОДА-ВЫВОДА

V	Tag	Data	V	Tag	Data	V	Tag	Data	V	Tag	Data	V	Tag	Data	V	Tag	Data	V	Tag	Data

Уменьшает количество конфликтов из-за промахов
Построение крайне затратное

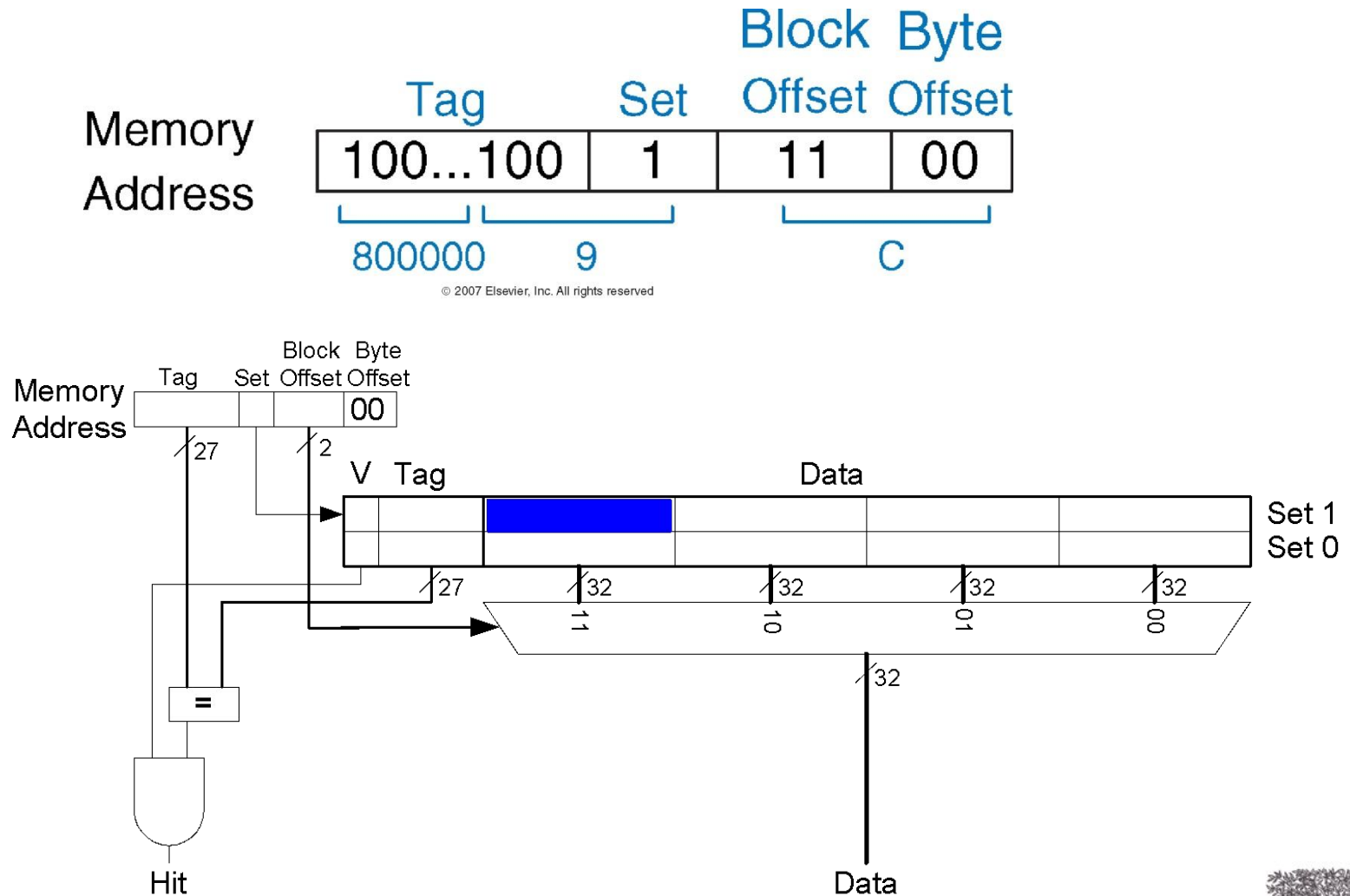
ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА

- # ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА



Кэш с бóльшим размером строки

ИЕРАРХИЯ ПАМЯТИ И
ПОДСИСТЕМА
ВВОДА-ВЫВОДА



Производительность кэша прямого отображения

```
    addi $t0, $0, 5
loop: beq  $t0, $0, done
      lw   $t1, 0x4($0)
      lw   $t2, 0xC($0)
      lw   $t3, 0x8($0)
      addi $t0, $t0, -1
      j    loop
done:
```

Процент промахов = ?

Производительность кэша прямого отображения

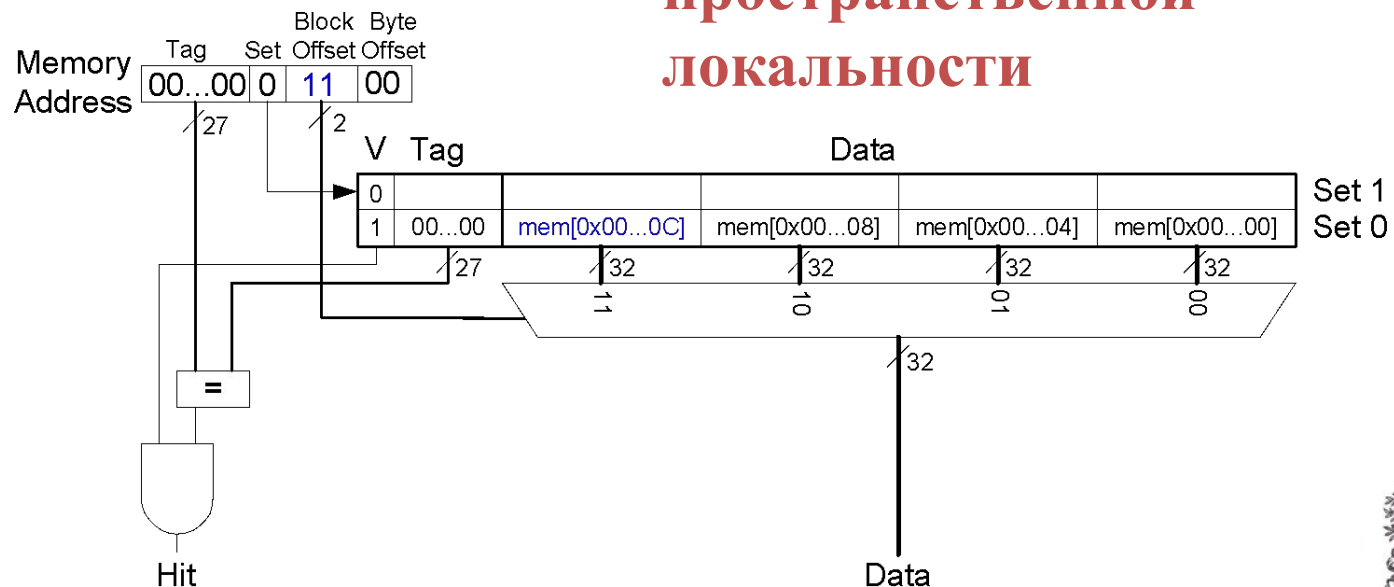
```

addi $t0, $0, 5
loop: beq $t0, $0, done
      lw  $t1, 0x4($0)
      lw  $t2, 0xC($0)
      lw  $t3, 0x8($0)
      addi $t0, $t0, -1
      j   loop
done:

```

**Процент промахов = 1/15
= 6.67%**

**Строки с бóльшим размером
уменьшают обязательные
промахи с помощью
пространственной
локальности**



Резюме организации кэш-

- Ёмкость: C
- Размер строки: b
- Количество строк в кэш-памяти: $B = C/b$
- Количество строк в наборе: N
- Количество наборов: $S = B/N$

Способ организации	Количество секций (N)	Количество наборов ($S = B/N$)
Прямого отображения	1	B
Наборно-ассоциативный	$1 < N < B$	B / N
Полностью ассоциативный	B	1

Промахи из-за недостаточной

- Кэш слишком мал, чтобы вместить сразу все данные, представляющие интерес
 - Если кэш заполнен: программа получает доступ к данным X и вытесняет данные Y
- Промахи из-за недостаточной ёмкости** возникают, когда снова будут необходимы данные Y
- Как выбрать такие данные Y, чтобы свести к минимуму вероятность необходимости в них снова?
- Замена редко используемых данных (англ. Least recently used, LRU):** вытеснение той строки, которая дольше всего не использовалась

Типы промахов

- **Неизбежные:** при первом доступе к данным
- **Из-за недостаточной ёмкости:** кэш слишком мал, чтобы вместить сразу все данные, представляющие интерес
- **Из-за конфликтов:** данные отображаются в один и тот же набор кэша
- **Цена промахов:** время, необходимое для извлечения строки из более низкого уровня иерархии

Замещение LRU

MIPS код

```
lw $t0, 0x04($0)
lw $t1, 0x24($0)
lw $t2, 0x54($0)
```

Way 1				Way 0			
V	U	Tag	Data	V	Tag	Data	
0	0			0			Set 3 (11)
0	0			0			Set 2 (10)
0	0			0			Set 1 (01)
0	0			0			Set 0 (00)

Замещение LRU

MIPS код

```
lw $t0, 0x04($0)
lw $t1, 0x24($0)
lw $t2, 0x54($0)
```

Way 1				Way 0				
V	U	Tag	Data	V	Tag	Data		
0	0			0				Set 3 (11)
0	0			0				Set 2 (10)
1	0	00...010	mem[0x00...24]	1	00...000	mem[0x00...04]		Set 1 (01)
0	0			0				Set 0 (00)

(a)

Way 1				Way 0				
V	U	Tag	Data	V	Tag	Data		
0	0			0				Set 3 (11)
0	0			0				Set 2 (10)
1	1	00...010	mem[0x00...24]	1	00...101	mem[0x00...54]		Set 1 (01)
0	0			0				Set 0 (00)

(b)

Резюме кэш-памяти

• Какие данные хранить в кэш-памяти?

- Недавно использованные данные (временная локальность)
- Рядом лежащие данные (пространственная локальность)

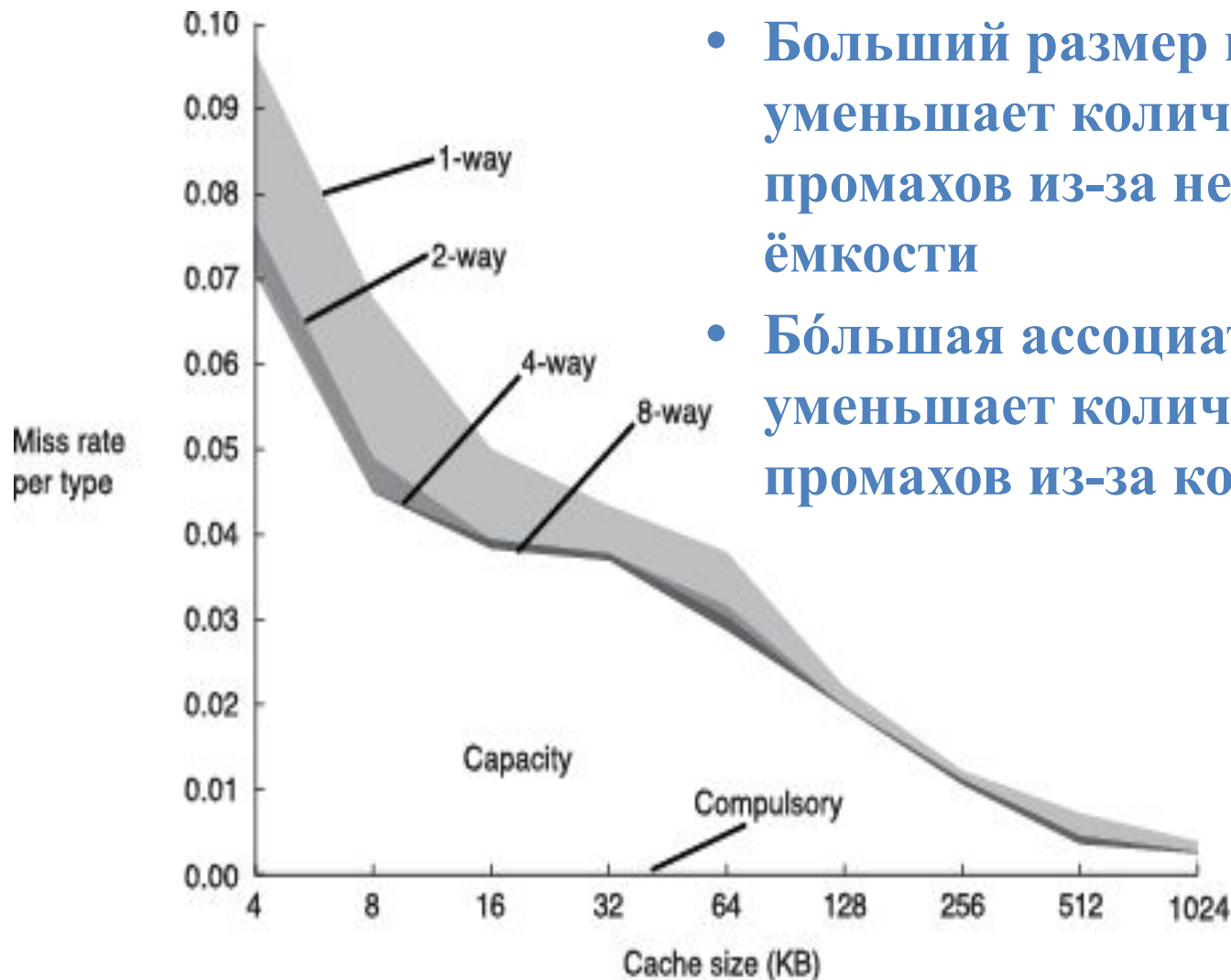
Как найти данные?

- Набор определяется адресом данных
- Слово внутри строки также определяется адресом
- В ассоциативном кэше данные могут находиться в одной из нескольких секций

Какие данные заместить?

- Замещать те секции данных в наборе, которые дольше не использовались

Динамика процента промахов

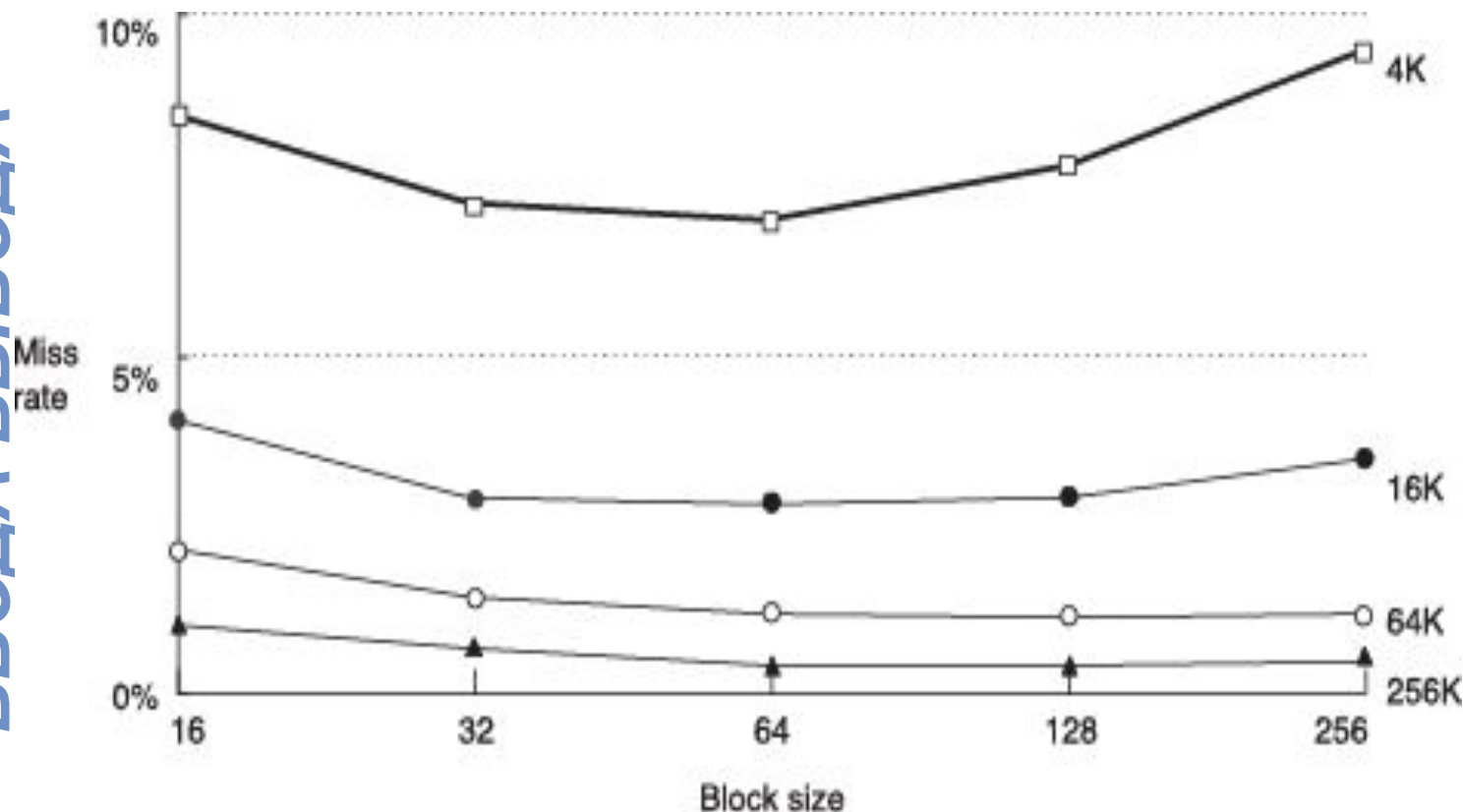


- Большой размер кэша уменьшает количество промахов из-за недостаточной ёмкости
- Бóльшая ассоциативность уменьшает количество промахов из-за конфликтов

Adapted from Patterson & Hennessy, *Computer Architecture: A Quantitative Approach*, 2011

Динамика процента промахов

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА



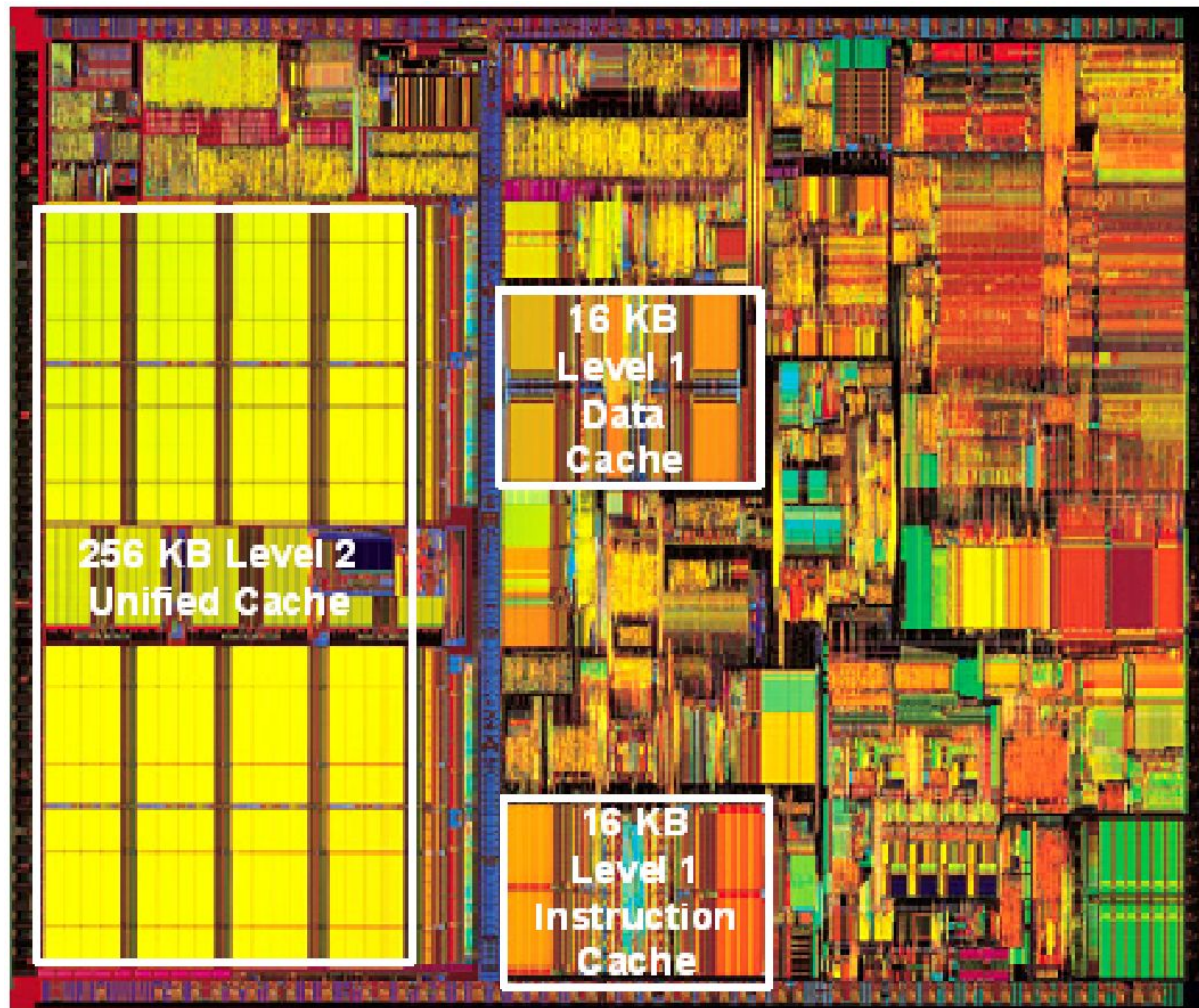
- Большие размеры строк уменьшают количество обязательных промахов
- Большие размеры строк увеличивают количество промахов из-за конфликтов

Многоуровневые кэши

- Кэши большего размера имеют меньший процент промахов, но более длительное время доступа
- Спроецируйте идею иерархии памяти на несколько уровней кэшей
- Уровень 1 (L1): маленький и быстрый (например 16 КВ, 1 такт)
- Уровень 2 (L2): больший и медленный (например 256 КВ, 2-6 циклов)
- Большинство современных компьютеров имеют кэши L1, L2 и L3

Intel Pentium III

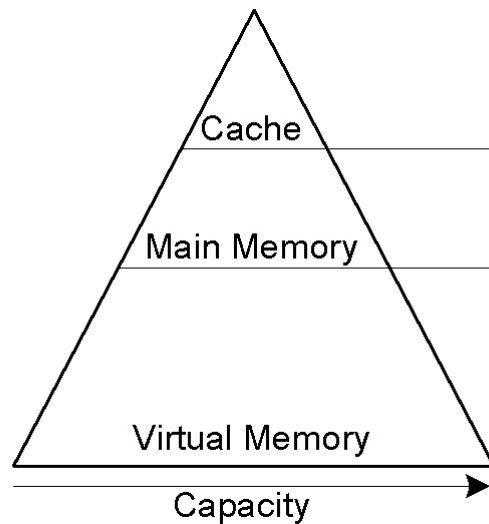
ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА



Виртуальная память

- Даёт иллюзию большего размера памяти
- Оперативная память (DRAM) выступает в качестве кэша для жесткого диска

Иерархия памяти

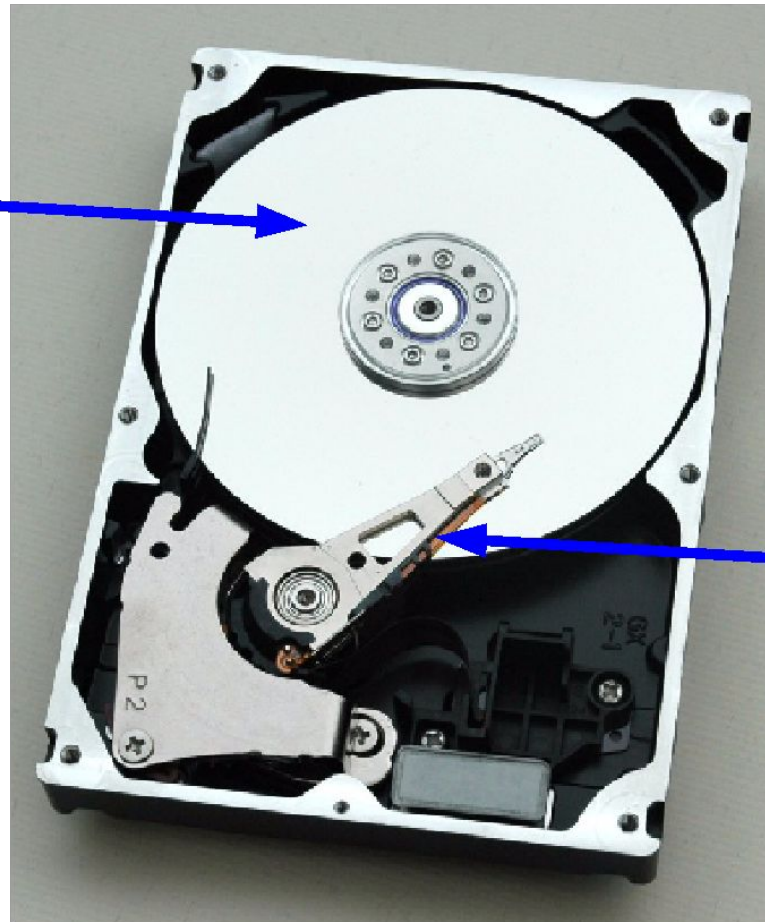


Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	\$10,000	1	25+
DRAM	\$10	10 - 50	10
SSD	\$1	100,000	0.5
HDD	\$0.1	10,000,000	0.1

- **Физическая память:** DRAM (оперативная память)
- **Виртуальная память:** жёсткий диск
 - медленная, большая, дешёвая

Жёсткий диск

Magnetic
Disks



Read/Write
Head

Поиск правильного положения занимает миллисекунды

Виртуальная память

- **Виртуальные адреса**

- Программы используют виртуальные адреса
- Всё виртуальное адресное пространство хранится на жёстком диске
- Подмножество виртуальных адресов данных хранится в DRAM
- ЦП транслирует виртуальные адреса в **физические адреса** (DRAM адреса)
- Данные, не помещающиеся в DRAM, выгружаются на жёсткий диск

- **Защита памяти**

- Каждая программа имеет своё виртуальное адресное пространство, отображаемое в физическое
- Две программы могут использовать тот же виртуальный адрес для различных данных
- Программы не должны знать, как работают другие программы
- Одна программа (или вирус) не может повредить память, используемую другой программой

Аналогия между виртуальной памятью и кэшем

ИЕРАРХИЯ ПАМЯТИ И
ПОДСИСТЕМА
ВВОДА-ВЫВОДА

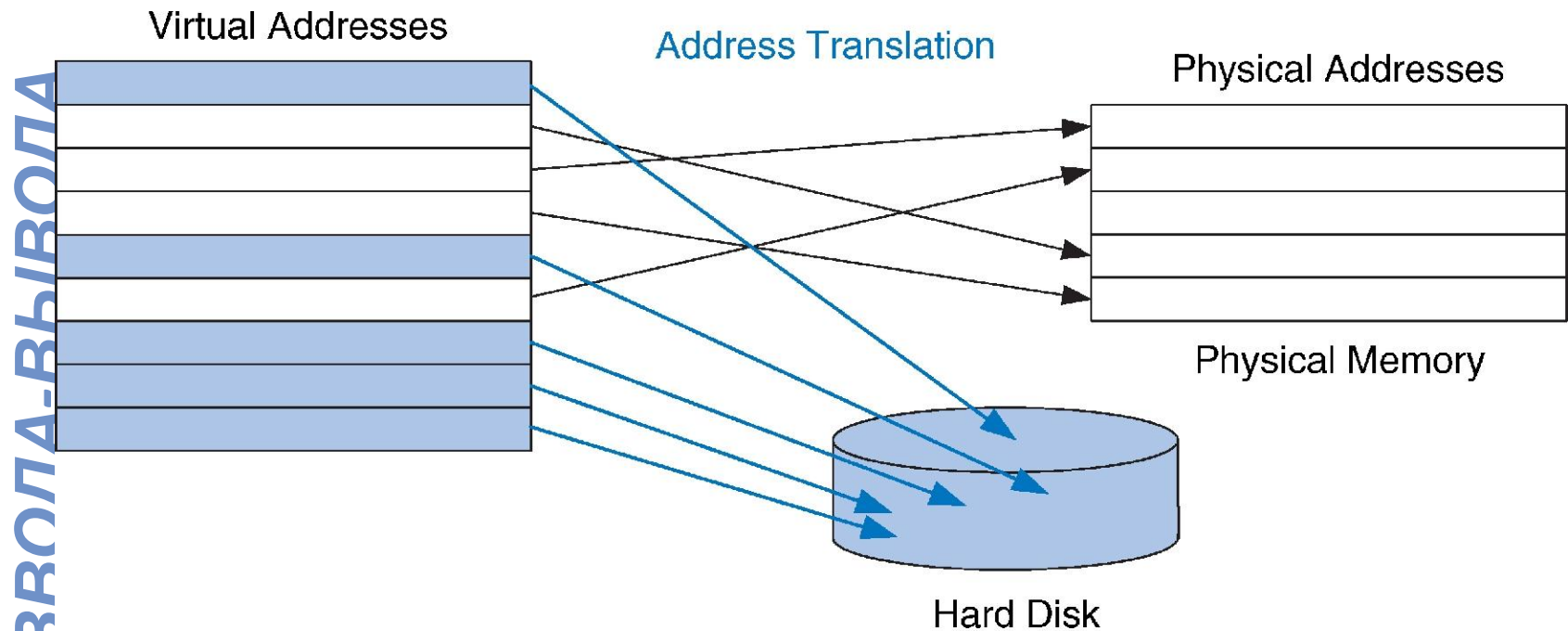
Кэш	Виртуальная память
Строка	Страница
Размер строки	Размер страницы
Смещение относительно начала строки	Смещение относительно начала страницы
Промах	Страничная ошибка
Тег	Номер виртуальной страницы

Физическая память выступает в качестве кэша виртуальной памяти

Терминология виртуальной

- **Размер страницы:** количество памяти, переносимое с жесткого диска в DRAM одновременно
- **Трансляция адреса:** определение физического адреса по виртуальному
- **Таблица страниц:** таблица поиска, используемая для трансляции виртуальных адресов в физические

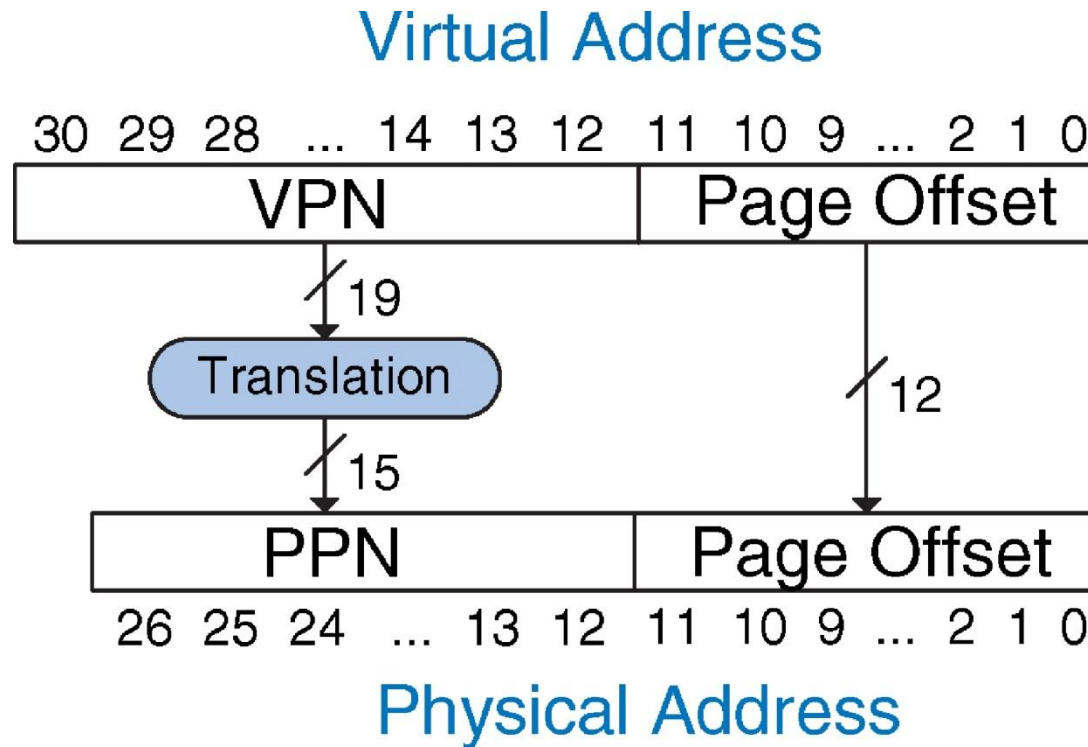
Виртуальные и физические



© 2007 Elsevier, Inc. All rights reserved

Большинство доступов осуществляется в физическую память
Но программы имеют большую ёмкость виртуальной памяти

Трансляция адреса



© 2007 Elsevier, Inc. All rights reserved

Пример виртуальной памяти

Система:

- Размер виртуальной памяти: 2 ГБ = 2^{31} байт
- Размер физической памяти: 128 МБ = 2^{27} байт
- Размер страницы: 4 КБ = 2^{12} байт

Пример виртуальной памяти

- Система:

- Размер виртуальной памяти: 2 ГБ = 2^{31} байт
- Размер физической памяти: 128 МБ = 2^{27} байт
- Размер страницы: 4 КБ = 2^{12} байт

Организация:

- Виртуальный адрес: **31** бит
- Физический адрес: **27** бит
- Смещение относительно начала страницы: **12** бит
- Номеров виртуальных страниц (англ. virtual page number, VPN) = $2^{31}/2^{12} = 2^{19}$ (VPN = 19 бит)
- Номеров физических страниц (англ. physical page number, PPN) = $2^{27}/2^{12} = 2^{15}$ (PPN = 15 бит)

Пример виртуальной памяти

- 19-битный номер виртуальной страницы
- 15-битный номер физической страницы

Physical Page Number
7FFF
7FFE
⋮
0001
0000

Physical Addresses
0x7FFF000 - 0x7FFFFF
0x7FFE000 - 0x7FFEFFF
⋮
0x0001000 - 0x0001FFF
0x0000000 - 0x0000FFF

Physical Memory

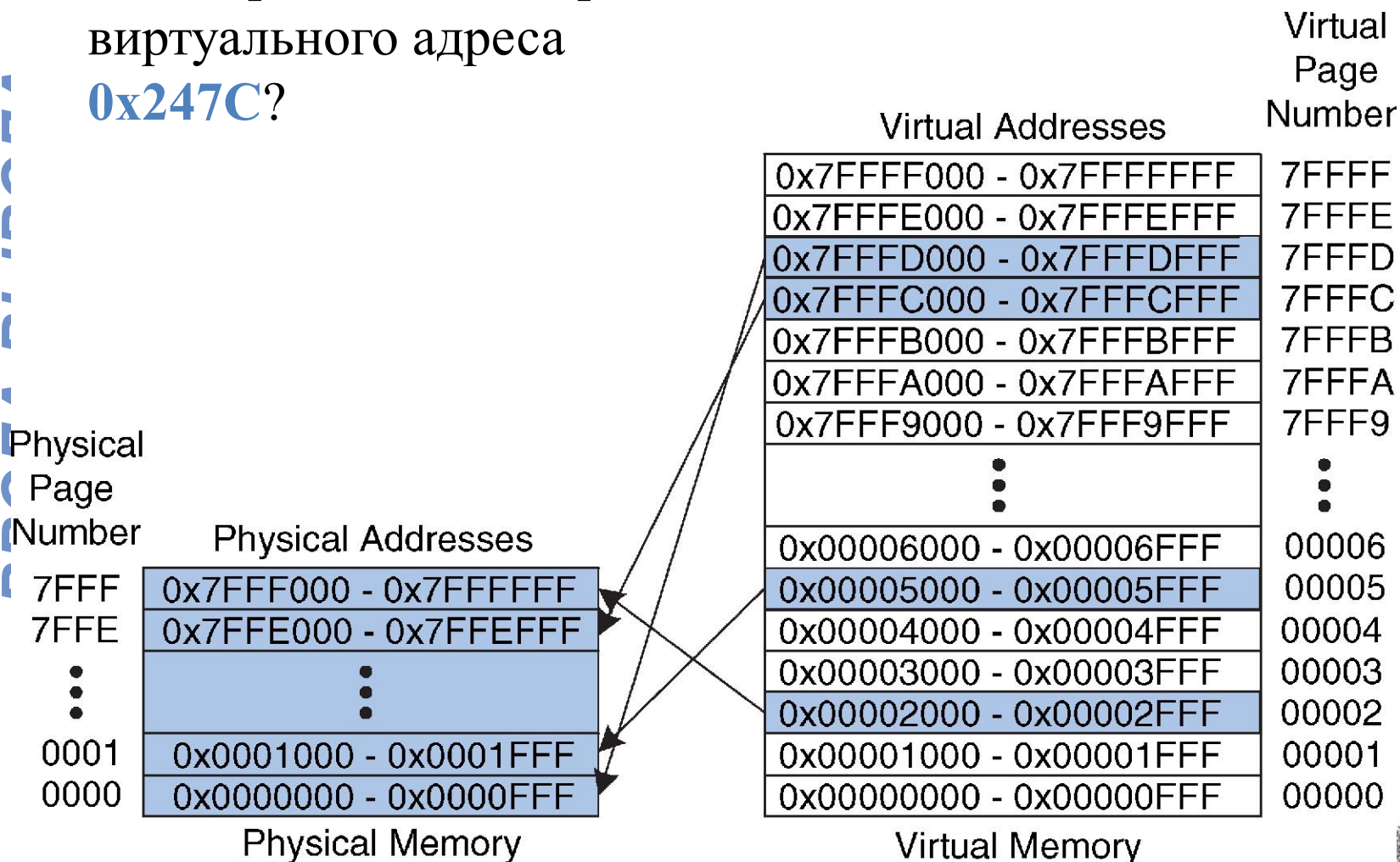
Virtual Addresses	Virtual Page Number
0x7FFFF000 - 0x7FFFFFFF	7FFFF
0x7FFFE000 - 0x7FFFEFFF	7FFFE
0x7FFFD000 - 0x7FFFDFFF	7FFFD
0x7FFFC000 - 0x7FFFCFFF	7FFFC
0x7FFFB000 - 0x7FFFBFFF	7FFFB
0x7FFFA000 - 0x7FFFAFFF	7FFFA
0x7FFF9000 - 0x7FFF9FFF	7FFF9
⋮	⋮
0x00006000 - 0x00006FFF	00006
0x00005000 - 0x00005FFF	00005
0x00004000 - 0x00004FFF	00004
0x00003000 - 0x00003FFF	00003
0x00002000 - 0x00002FFF	00002
0x00001000 - 0x00001FFF	00001
0x00000000 - 0x00000FFF	00000

Virtual Memory

Пример виртуальной памяти

Каков физический адрес
виртуального адреса
0x247C?

ИЕРАРХИЯ ПАМЯТИ И
ПОДСИСТЕМА



© 2007 Elsevier, Inc. All rights reserved

Пример виртуальной памяти

Каков физический адрес
виртуального адреса

0x247C?

- VPN = **0x2**
- VPN 0x2 отображается в PPN **0x7FFF**
- 12-битное смещение от начала страницы: **0x47C**
- Физический адрес = **0x7FFF47C**

Physical
Page
Number

7FFF
7FFE
⋮
0001
0000

Physical Addresses

0x7FFF000 - 0x7FFFFF
0x7FFE000 - 0x7FFEFFF
⋮
0x0001000 - 0x0001FFF
0x0000000 - 0x0000FFF

Physical Memory

Virtual Addresses

0x7FFFF000 - 0x7FFFFFFF
0x7FFFE000 - 0x7FFFEFFF
0x7FFFD000 - 0x7FFFDFFF
0x7FFFC000 - 0x7FFFCFFF
0x7FFFB000 - 0x7FFFBFFF
0x7FFFA000 - 0x7FFFAFFF
0x7FFF9000 - 0x7FFF9FFF
⋮
0x00006000 - 0x00006FFF
0x00005000 - 0x00005FFF
0x00004000 - 0x00004FFF
0x00003000 - 0x00003FFF
0x00002000 - 0x00002FFF
0x00001000 - 0x00001FFF
0x00000000 - 0x00000FFF

Virtual Memory

Virtual
Page
Number

7FFFF
7FFFE
7FFFD
7FFFC
7FFFB
7FFFA
7FFF9
⋮
00006
00005
00004
00003
00002
00001
00000

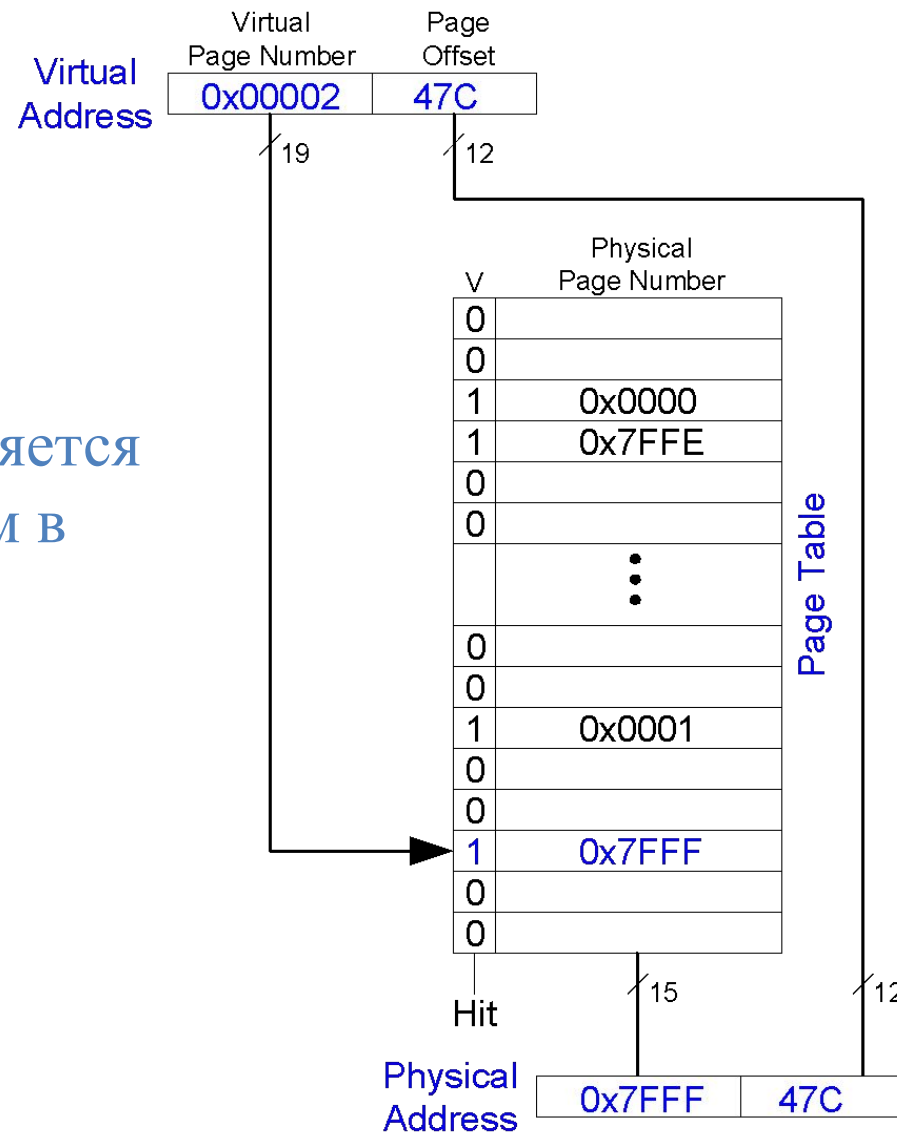
Как провести трансляцию?

- **Таблица страниц**
 - Содержит запись для каждой виртуальной страницы
 - Запись содержит:
 - **Бит достоверности:** 1 если страница находится в физической памяти
 - **Номер физической страницы:** расположение страницы

Пример таблицы страниц

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА

VPN является
индексом в
таблице
страниц



Первый пример таблицы

Каков физический
адрес виртуального
адреса **0x5F20**?

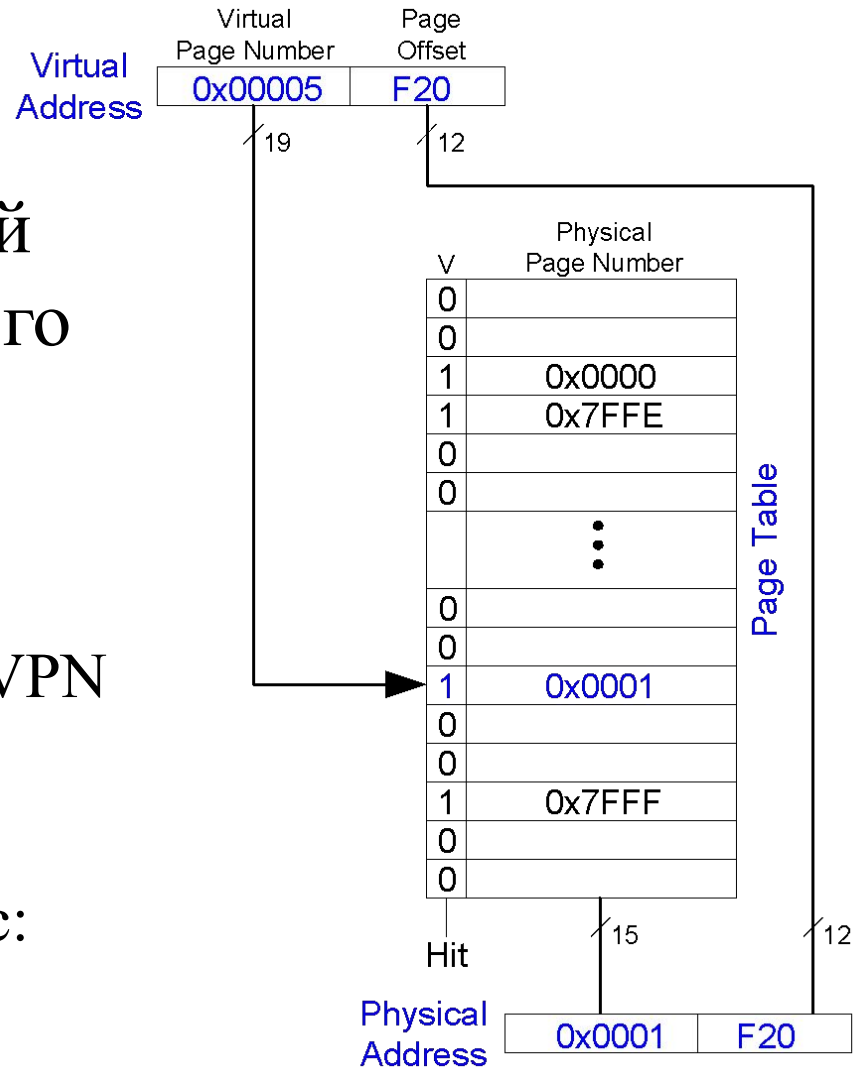
V	Physical Page Number
0	
0	
1	0x0000
1	0x7FFE
0	
0	
	⋮
0	
0	
1	0x0001
0	
0	
1	0x7FFF
0	
0	

Page Table

Первый пример таблицы

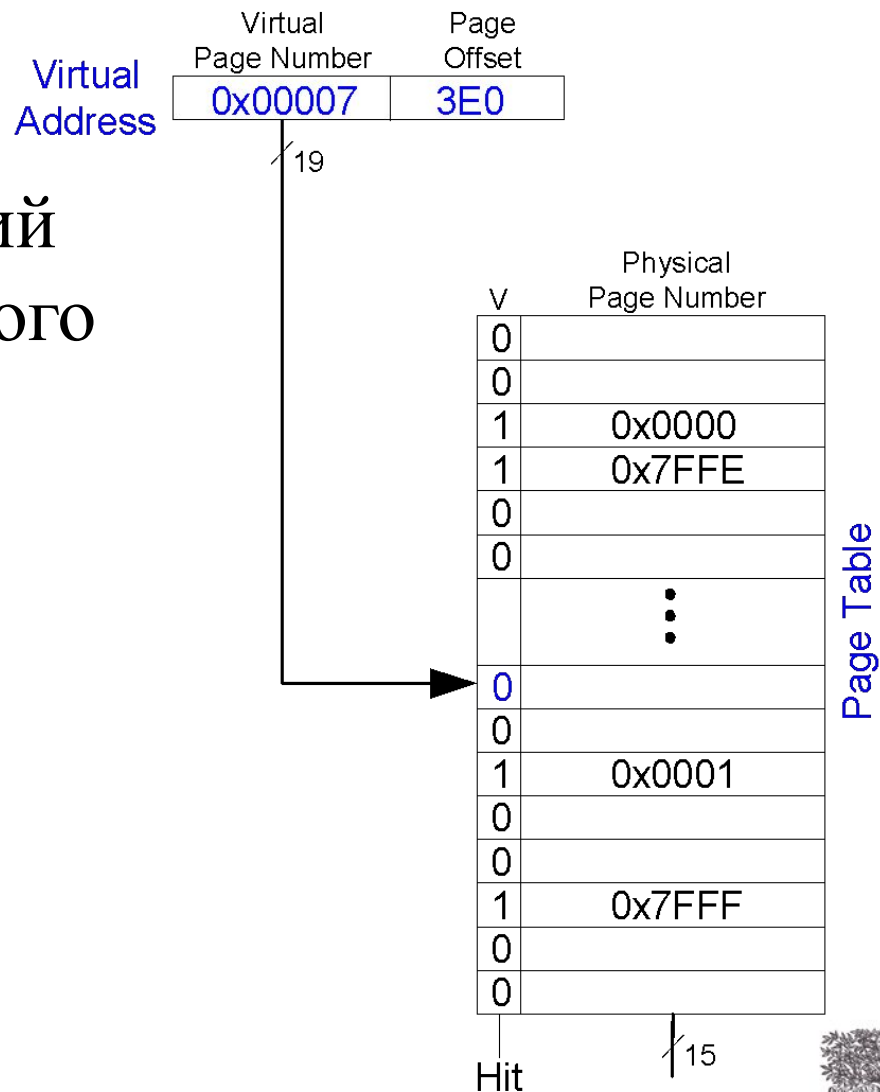
Каков физический
адрес виртуального
адреса **0x5F20**?

- VPN = 5
- Запись с № 5 в
таблице страниц VPN
5 => физическая
страница **1**
- Физический адрес:
0x1F20



Второй пример таблицы страниц

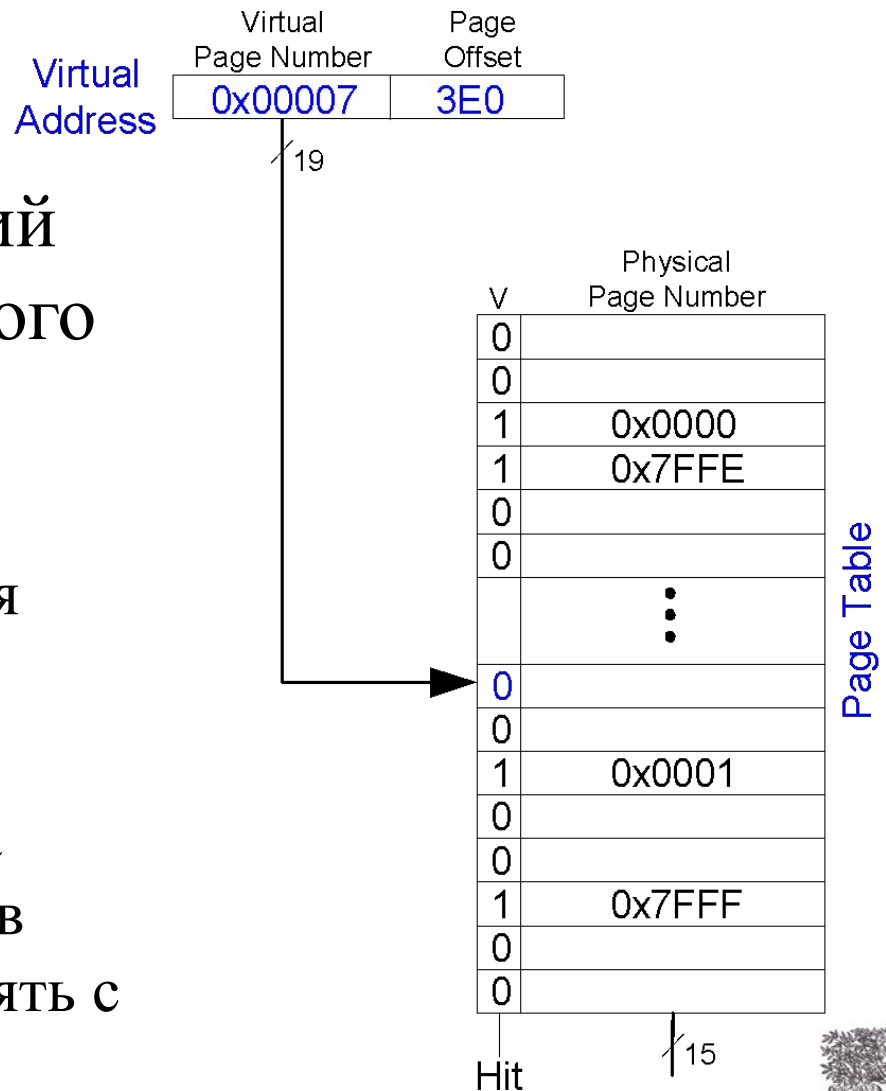
Каков физический
адрес виртуального
адреса **0x73E0**?



Второй пример таблицы страниц

Каков физический
адрес виртуального
адреса **0x73E0**?

- VPN = 7
- Запись 7 является
недостоверной
- Виртуальная
страница должна
быть *загружена* в
физическую память с
диска



Проблемы таблицы страниц

- **Таблица страниц большая**
 - как правило, находится в физической памяти
- **Загрузка/сохранение требуют два доступа к оперативной памяти:**
 - Один для трансляции (чтение из таблицы страниц)
 - Один для доступа к данным (после трансляции)
- **Уменьшает производительность памяти в 2 раза**
 - *Если мы не станем умнее...*

Буфер ассоциативной трансляции

(TLB)

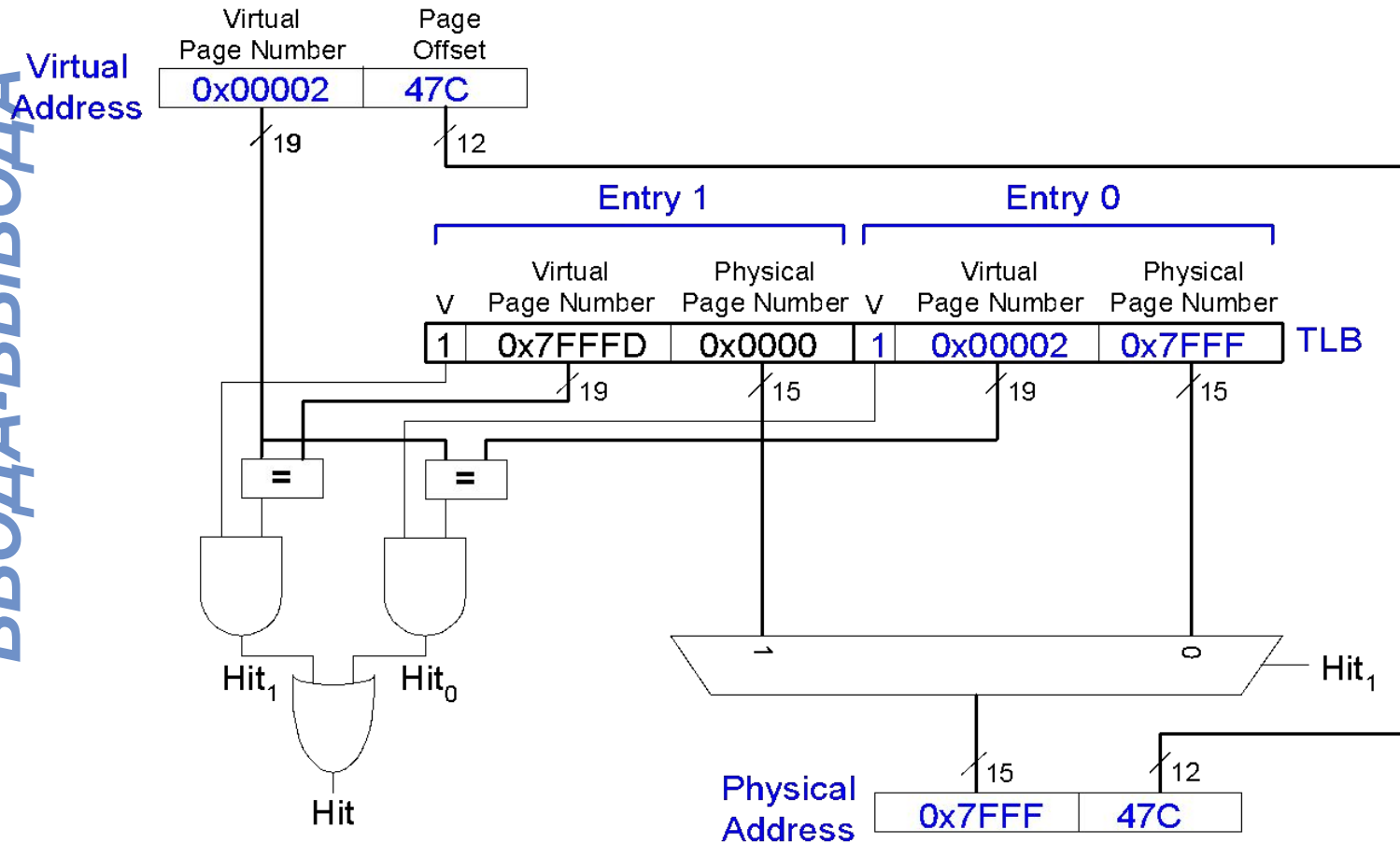
- Небольшой кэш самых последних трансляций
- Снижение количества доступов к памяти для большинства загрузок/сохранений с 2 до 1

- Доступ к таблице страниц: большая пространственная локальность
 - Большой размер страницы: идущие друг за другом загрузки/сохранения имеют большую вероятность доступа к одной и той же странице

TLV

- Маленький: доступ < 1 такта
- Обычно содержит 16 – 512 записей
- Полностью ассоциативный
- Обычно процент попадания $> 99 \%$
- Снижение количества доступов к памяти для большинства загрузок/сохранений с 2 до 1

Пример 2 – запись TLB



Защита памяти

- Множество процессов (программ) работают одновременно
- Каждый процесс имеет свою собственную таблицу страниц
- Каждый процесс может использовать всё виртуальное адресное пространство
- Процесс может получить доступ только к физической странице, отображённой в его таблице страниц

Резюме виртуальной памяти

- Виртуальная память увеличивает **пропускную способность**
- Подмножество виртуальных страниц хранится в физической памяти
- **Таблица страниц** отображает виртуальные страницы в физические – трансляция адресов
- **TLB** повышает скорость трансляции адресов
- Наличие различных таблиц страниц для различных программ обеспечивает **защиту памяти**

Ввод-вывод, отображённый в

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА

- Процессор получает доступ к устройствам ввода-вывода так же, как и к памяти (например к клавиатурам, мониторам, принтерам)
- Каждому устройству ввода-вывода присваивается один или более адресов
- Когда этот адрес обнаруживается, то данные считываются/записываются в устройство ввода-вывода, а не в память
- Часть адресного пространства отводится устройствам ввода-вывода

Дешифратор адреса:

- Смотрит на адрес для того, чтобы определить – какое устройство или память связывается с процессором

Регистры ввода-вывода:

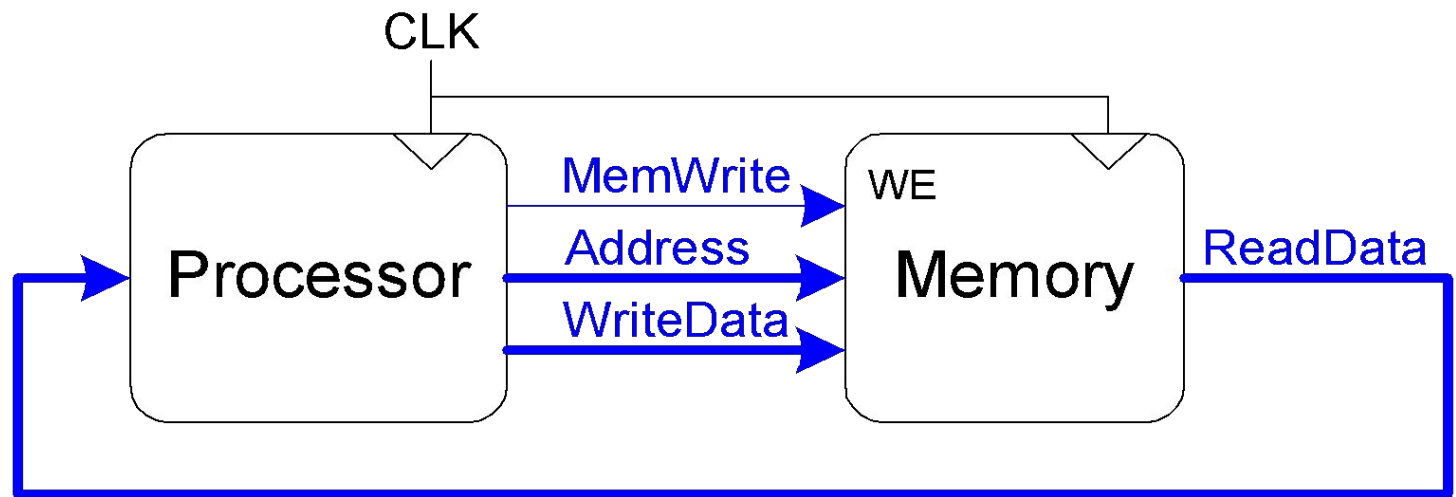
- Содержат значения, записываемые в устройство ввода-вывода

Мультиплексор чтения данных:

- Осуществляет выбор между памятью или устройствами ввода-вывода и устанавливает их в качестве источника данных, передаваемых процессору

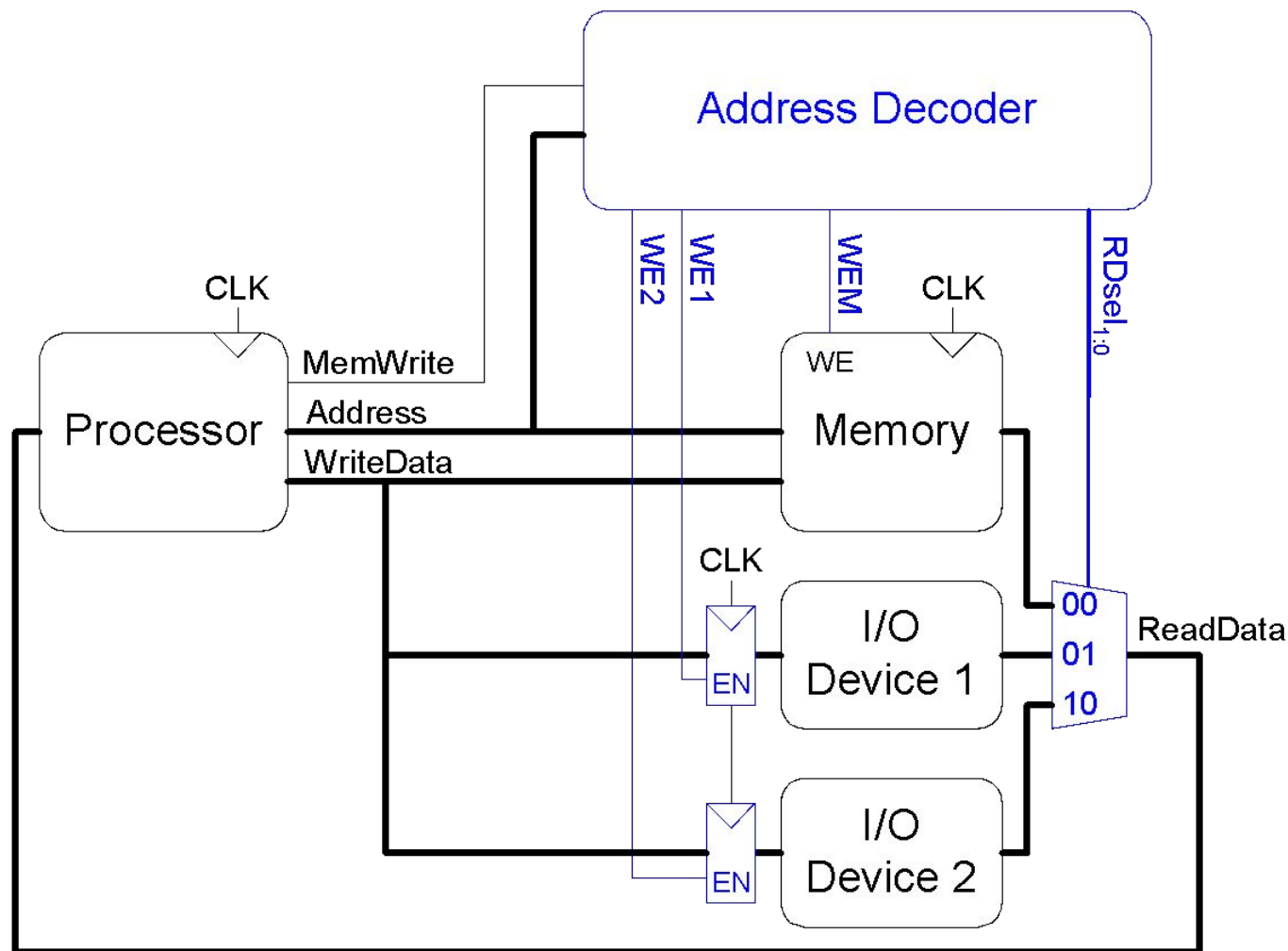
Интерфейс памяти

ИЕРАРХИЯ ПАМЯТИ И
ПОДСИСТЕМА
ВВОДА-ВЫВОДА



Аппаратная реализация ввода-вывода, отображённого в память

ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА



Код ввода-вывода, отображённого в памяти

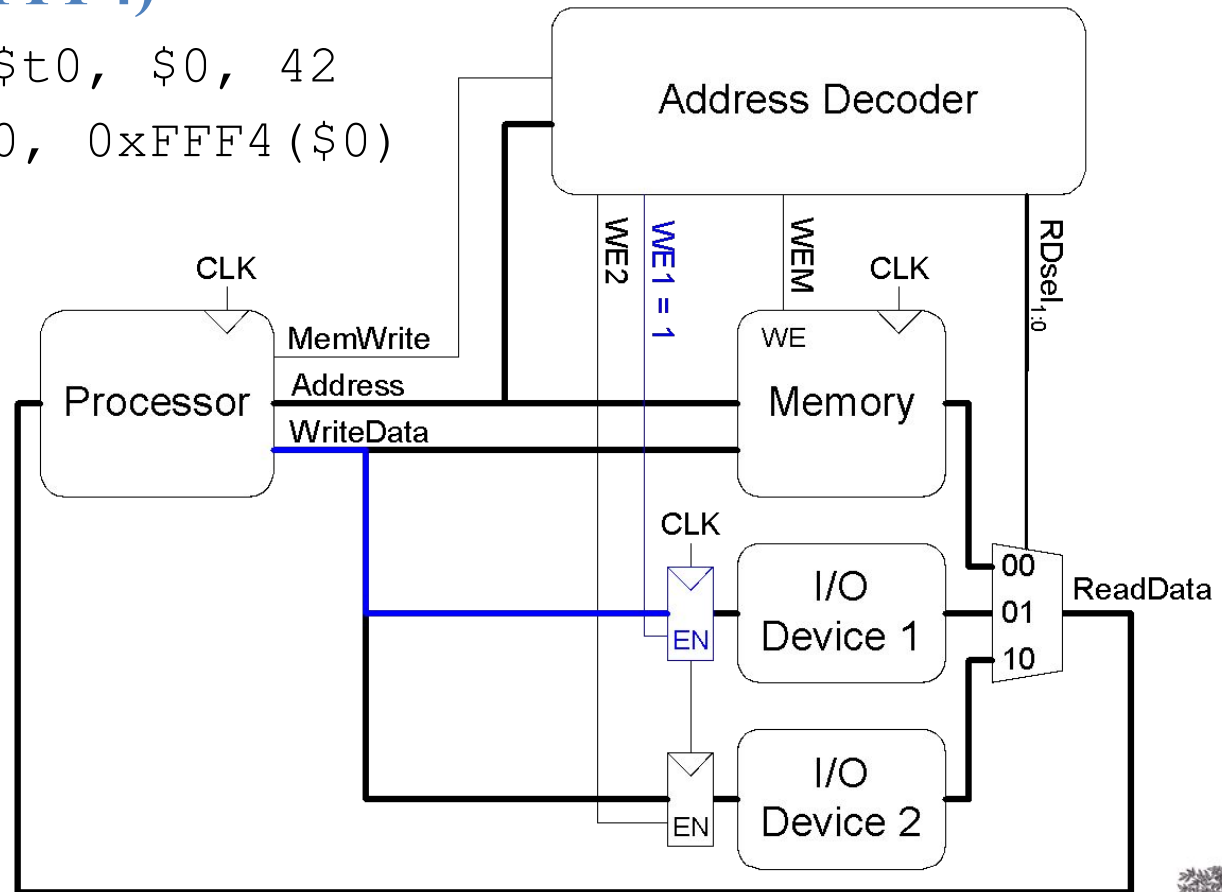
ИЕРАРХИЯ ПАМЯТИ И ПОДСИСТЕМА ВВОДА-ВЫВОДА

- Предположим, что устройству ввода-вывода 1 присваивается адрес $0xFFFFFFFF4$
 - Запишите значение 42 в устройство ввода-вывода 1
 - Прочтите значение из устройства ввода-вывода 1 и поместите его в $\$t3$

Код ввода-вывода, отображённого в ПАМЯТИ

- Запишите значение 42 в устройство ввода-вывода 1 (0xFFFFFFF4)

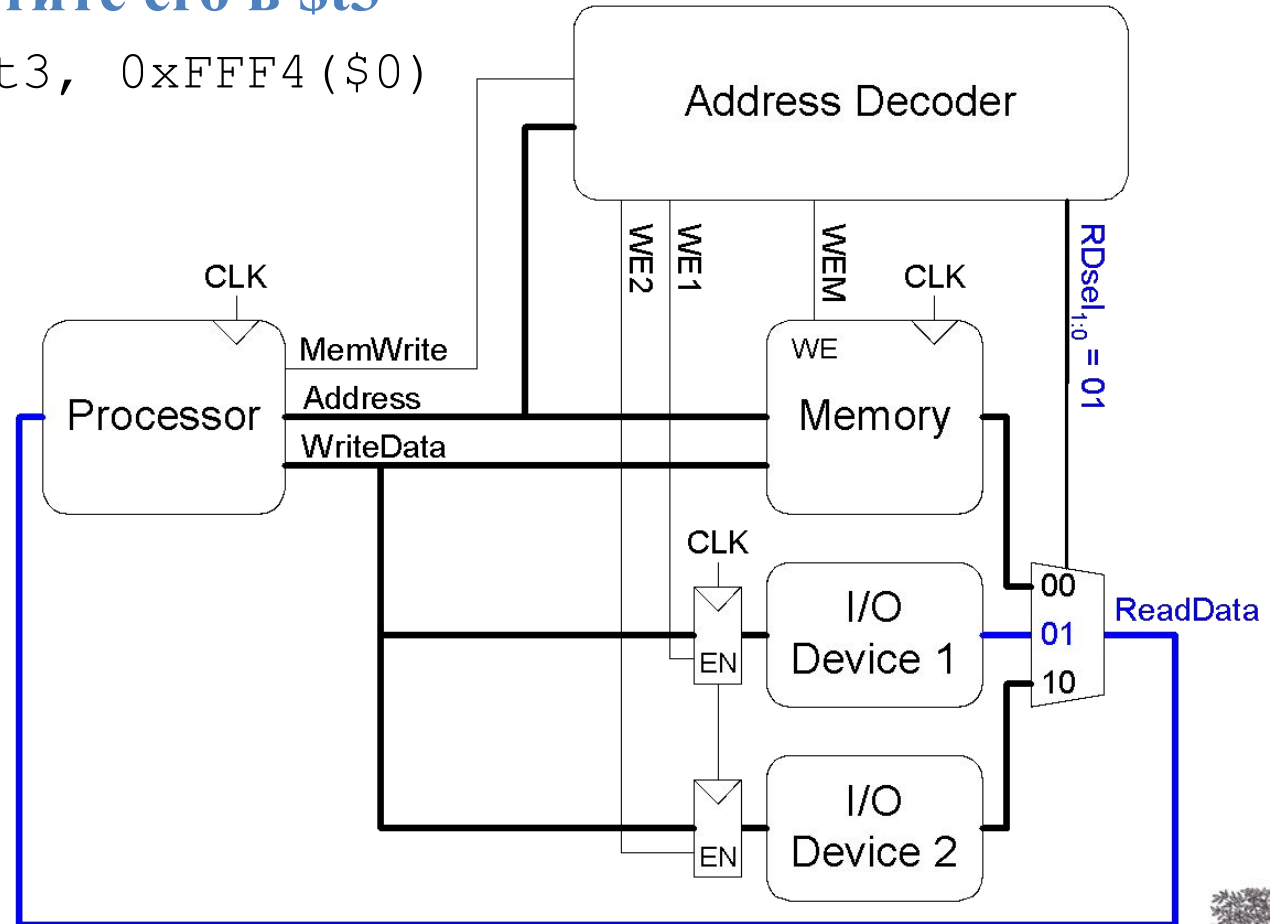
```
addi $t0, $0, 42  
sw $t0, 0xFFFF4($0)
```



Код ввода-вывода, отображённого в ПАМЯТИ

- Прочтите значение из устройства ввода-вывода 1 и поместите его в \$t3

```
lw $t3, 0xFFF4($0)
```



Подсистема ввода-вывода

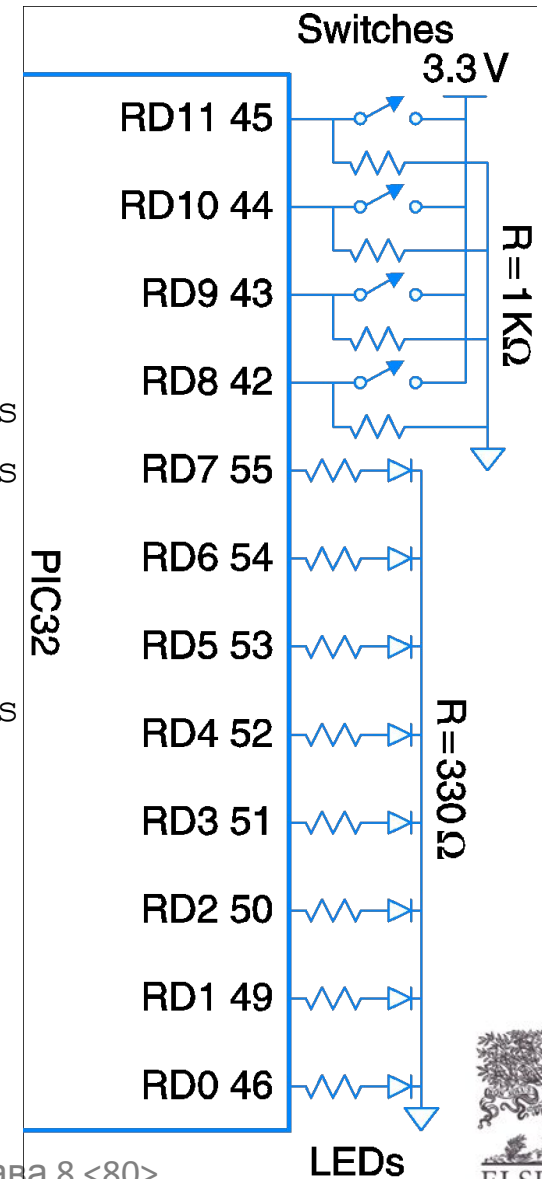
- Встроенные подсистемы ввода-вывода
 - Тостеры, светодиоды и т. д.
- Подсистемы ввода-вывода персональных компьютеров

- Пример микроконтроллера: PIC32
 - микроконтроллер
 - 32-битный MIPS процессор
 - низкоуровневая периферия включает:
 - последовательные порты
 - таймеры
 - аналого-цифровые преобразователи

Цифровой ввод-вывод

```
// С код
#include <p3xxxx.h>

int main(void) {
    int switches;
    TRISD = 0xFF00;           // RD[7:0] outputs
                              // RD[11:8] inputs
    while (1) {
        // read & mask switches, RD[11:8]
        switches = (PORTD >> 8) & 0xF;
        PORTD = switches;    // display on LEDs
    }
}
```

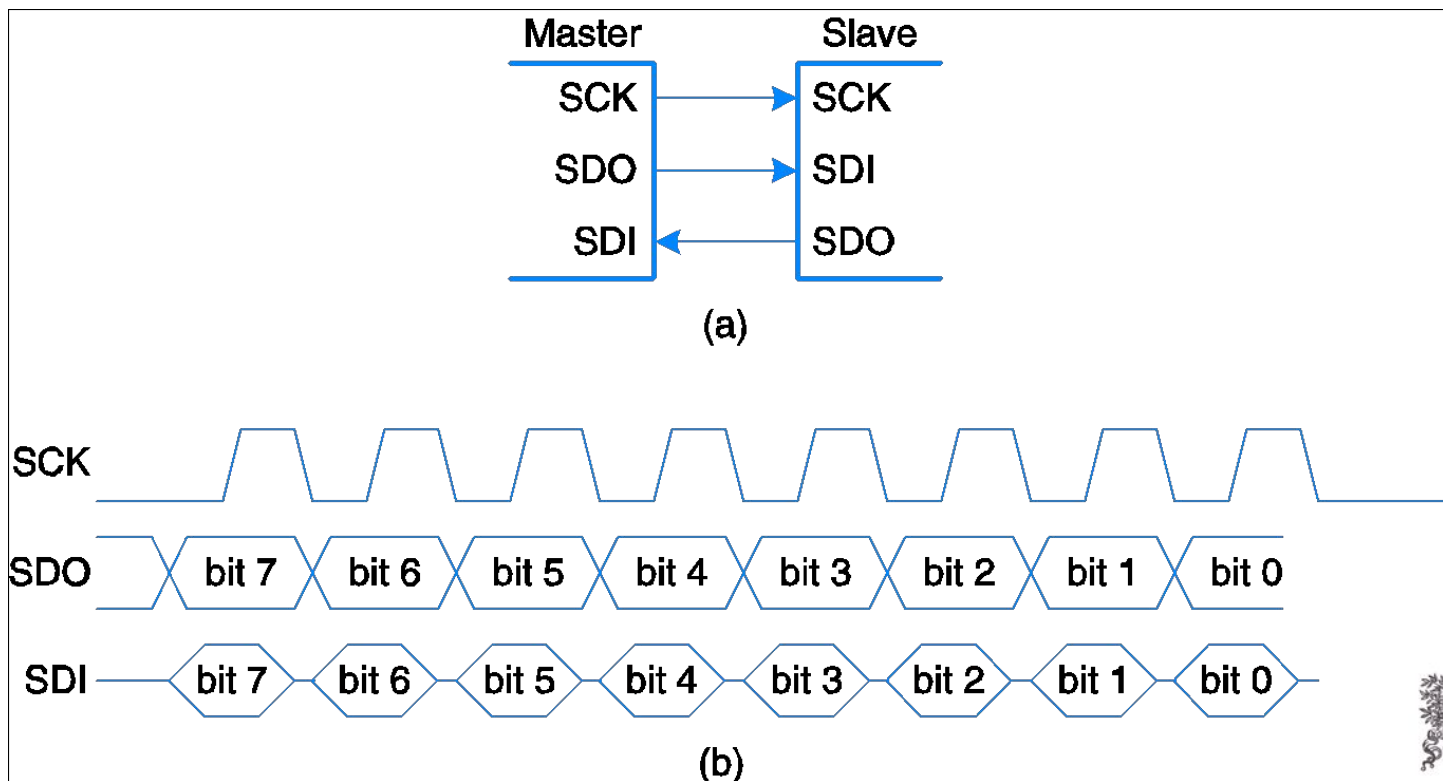


Последовательный ввод-

- Пример последовательных протоколов
 - последовательный периферийный интерфейс (англ. Serial Peripheral Interface, **SPI**)
 - универсальный асинхронный приемопередатчик (англ. Universal Asynchronous Receiver/Transmitter, **UART**)
 - а также: I²C, USB, Ethernet и т. д.

SPI: последовательный периферийный интерфейс

- Ведущее устройство (master) инициирует установку связи с ведомым устройством (slave) посредством генерации импульсов на пин SCK
- Ведущее устройство посылает данные на пин SDO (Serial Data Out – последовательный выход данных) ведомому устройству, начиная со старшего бита
- Ведомое устройство может послать данные (на пин SDI) ведущему устройству, начиная со старшего бита



UART: универсальный асинхронный приемопередатчик

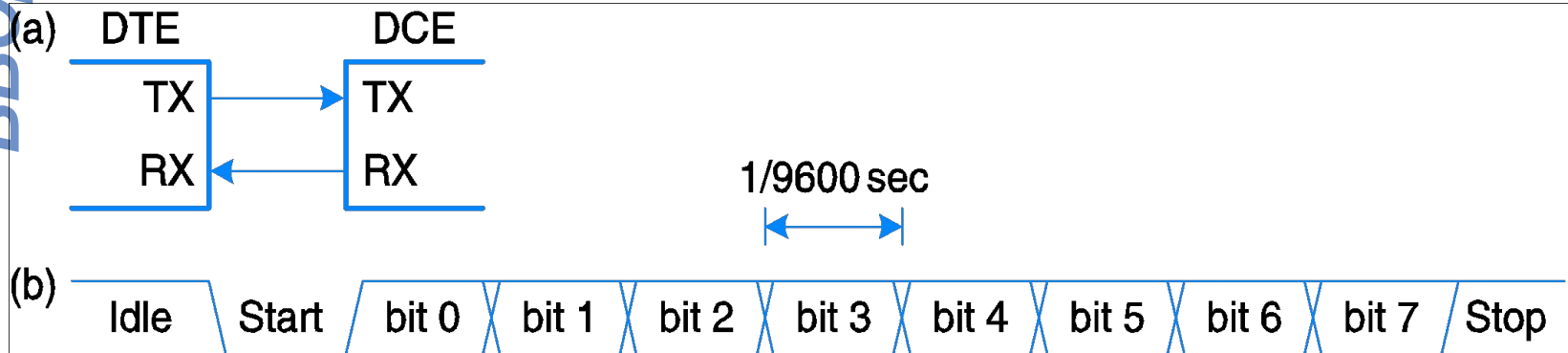
- Параметры:

- Стартовый бит (0), 7-8 бит данных, бит чётности (опционален), 1 и более стоповых битов (1)
- Скорость передачи данных: 300, 1200, 2400, 9600, ...115200 бод

- Линия простаивает при высоком логическом уровне (1)

Обычные параметры:

- 8 бит данных, без контроля чётности, 1 стоповый бит, 9600 бод



Таймеры

```
// Create specified ms/us of delay using built-in timer
#include <P32xxxx.h>

void delaymicros(int micros) {
    if (micros > 1000) {                // avoid timer overflow
        delaymicros(1000);
        delaymicros(micros-1000);
    }
    else if (micros > 6){
        TMR1 = 0;                      // reset timer to 0
        T1CONbits.ON = 1;              // turn timer on
        PR1 = (micros-6)*20;           // 20 clocks per microsecond
                                        // Function has overhead of ~6 us
        IFS0bits.T1IF = 0;             // clear overflow flag
        while (!IFS0bits.T1IF);        // wait until overflow flag set
    }
}

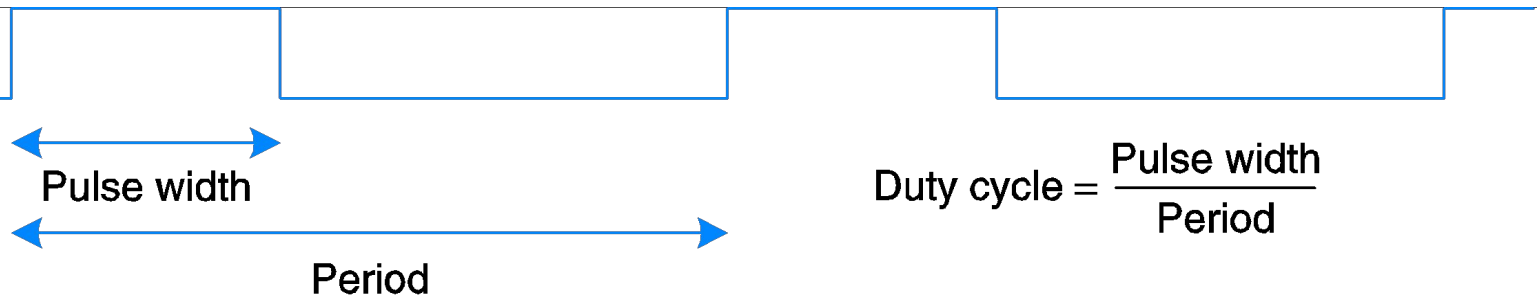
void delaymillis(int millis) {
    while (millis--) delaymicros(1000); // repeatedly delay 1 ms
                                        // until done
}
```

Аналоговый ввод-вывод

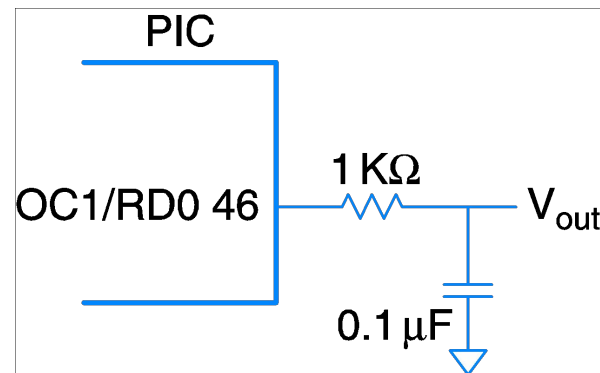
- Необходим для взаимодействия с внешним миром
- **Аналоговый ввод:** аналого-цифровое преобразование
 - Часто включено в микроконтроллер
 - N битовое: преобразует входной аналоговый сигнал от V_{ref-} до $0-2^{N-1}$
- **Аналоговый вывод:**
 - Цифро-аналоговое преобразование
 - Обычно требует внешний чип (например AD558 или LTC1257)
 - N -битовое: преобразует цифровой сигнал от $0-2^{N-1}$ до V_{ref-} до V_{ref+}
 - Широтно-импульсная модуляция

Широтно-импульсная модуляция (ШИМ)

- Среднее значение пропорционально коэффициенту заполнения



- Добавить фильтр верхних частот на выходе для установки среднего значения



• Примеры

- Символьный ЖК-дисплей
- VGA монитор
- Беспроводная связь Bluetooth
- Двигатели

- Универсальная последовательная шина (англ. Universal Serial Bus, USB)
 - USB 1.0 был выпущен в 1996 году
 - стандартизация кабелей/программного обеспечения для внешних устройств
- Шина связи периферийных устройств (англ. Peripheral Component Interconnect, PCI)/PCI Express (PCIe)
 - Разработана Intel, стала широко распространена с 1994 года
 - 32-битная параллельная шина
 - используется для карт расширения (например: звуковые карты, видеокарты и т. д.)
- Память с удвоенной скоростью передачи данных (англ. double-data rate memory, DDR)

- Протокол управления передачей (англ. Transmission Control Protocol, TCP) и межсетевой протокол (англ. Internet Protocol, IP)
 - Физическое соединение: Ethernet-кабель или Wi-Fi
- SATA – интерфейс жесткого диска
- Подключение к ПК (датчики, приводы, микроконтроллеры и т. д.)
 - Системы сбора данных (англ. Data Acquisition Systems, DAQs)
 - USB-подключение