Структурная и функциональная организация ЭВМ (Computer Organization and Design)

БГУИР
кафедра ЭВМ

Доцент кафедры ЭВМ

Лекция 20-21
«Кэш-I»

2019

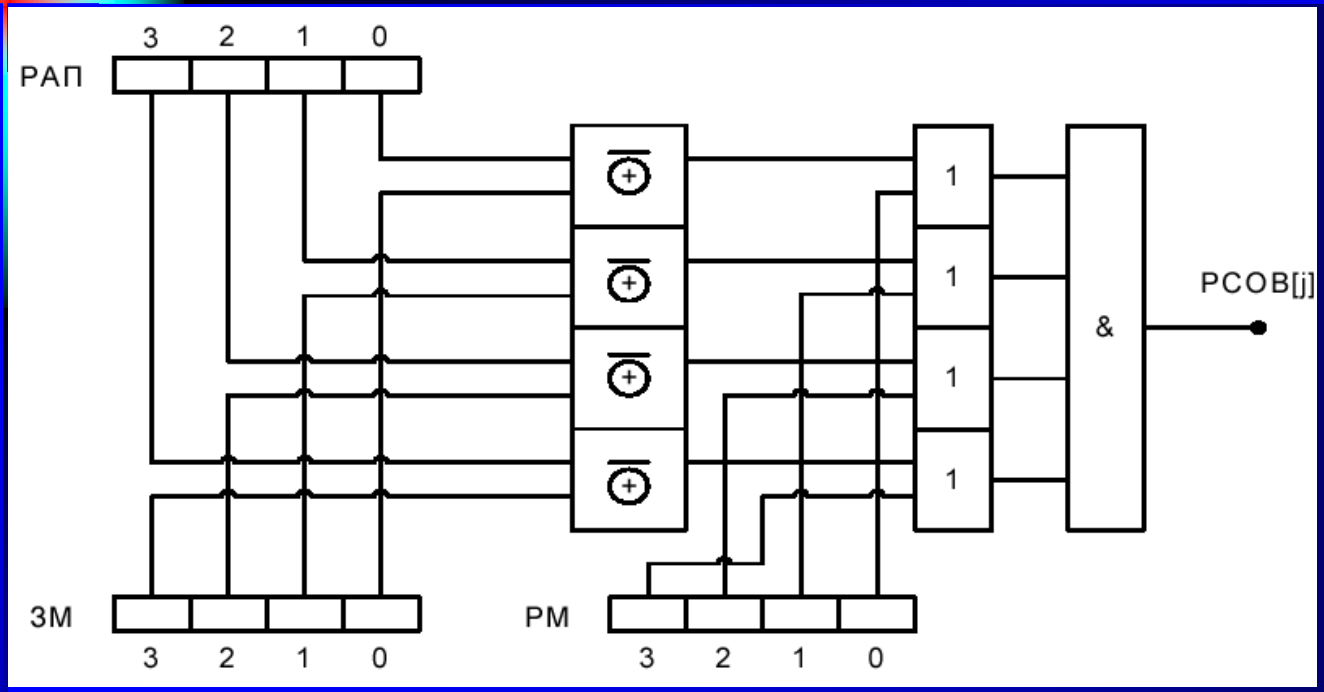
План лекции

1. Ассоциативные ЗУ
2. Организация Кэш-памяти
3. Полностью ассоциативное отображение
4. Прямое отображение
5. Множественно-ассоциативное отображение

Ассоциативная память

Структура ассоциативного ЗУ

Ассоциативная память



РАП – регистр ассоциативного признака;

РМ – регистр маски;

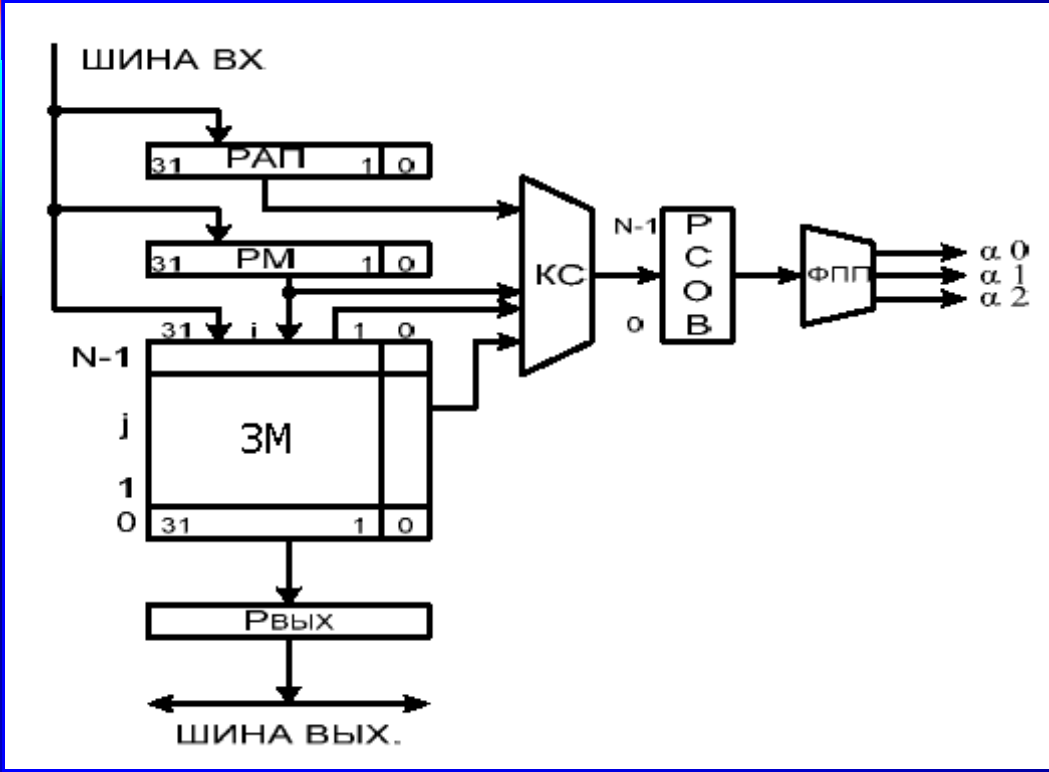
ЗМ – запоминающий массив;

РСОВ [j] – регистр совпадений;

ЗМ содержит N ячеек для указания занятости которых используется нулевой разряд (0-свободна, 1-занята) - необходим для поиска свободных ячеек при записи новой информации.

РМ - в ассоциативном поиске участвует только тот разряд, в котором $РМ_i = 1$, если $РМ_i = 0$, то i-ый разряд не участвует в поиске.

Ассоциативная память



РАП – регистр ассоциативного признака;

РМ – регистр маски;

ЗМ – запоминающий массив;

РСОВ [j] – регистр совпадений;

КС – комбинационная схема

ФПП – формирование признака поиска

а0- данные не найдены

а1- содержатся в одной яч-ке

а2- сод-ся в нескольких яч-х.

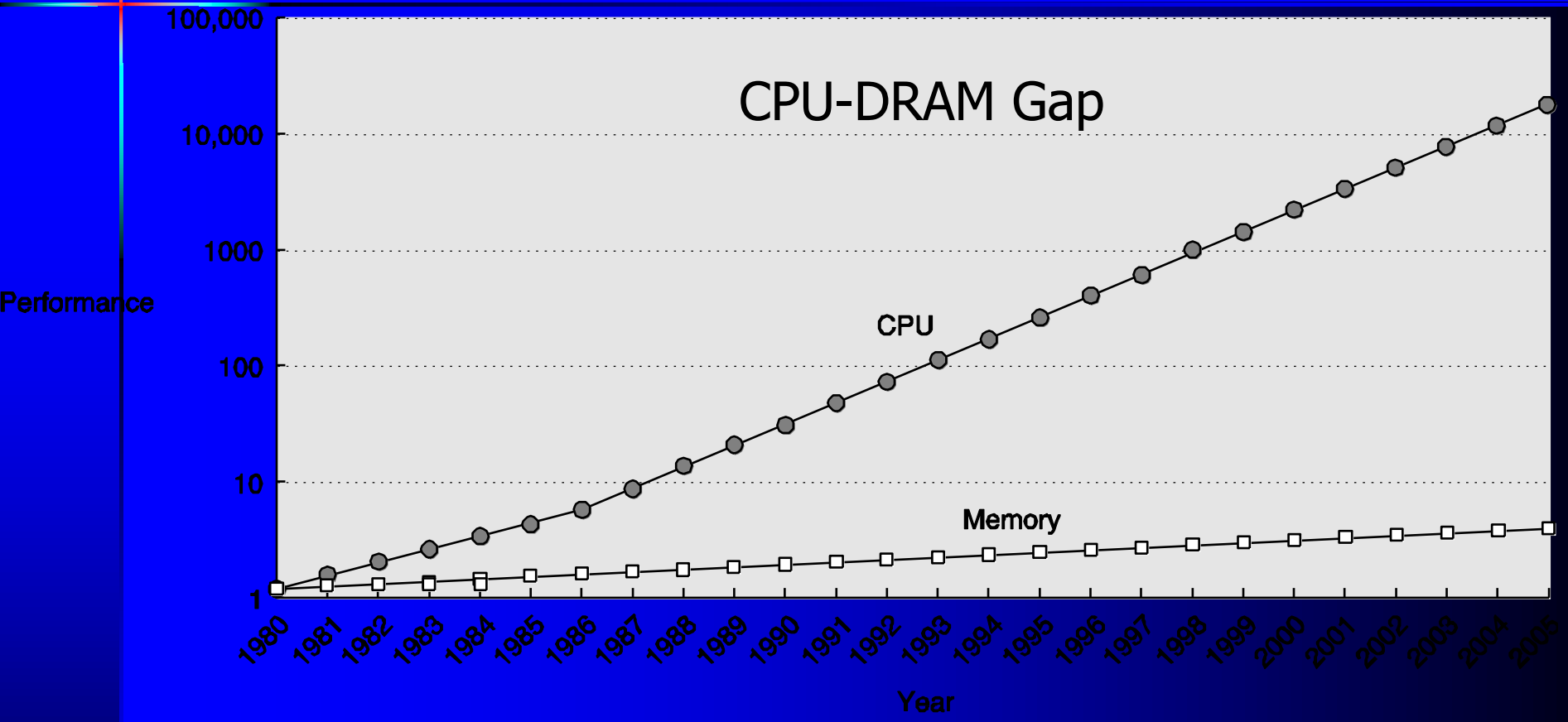
При считывании сначала – поиск по ассоциативному признаку в РАП и по РМ, далее анализируются признаки поиска.
При α0 = 1 считывание отменяется из-за отсутствия искомой информации. При α1 = 1 считывается единственное найденное слово. При α2 = 1 – обычно считывается слово из ячейки, имеющей наименьший номер и отмеченной единицей в РСОВ.

Ассоциативная память

При записи сначала отыскивается свободная ячейка в ЗМ. Для ее поиска выполняется операция контроля ассоциаций для РАП равного 11..110 и РМ = 00…001, так как у свободных ячеек последний младший разряд равен нулю. Все найденные свободные ячейки отмечаются единицами в РСОВ. Для записи выбирается свободная ячейка с наименьшим номером, в нее записывается слово с шины входных данных. Если α0 = 0, то есть, свободных ячеек нет, то производится удаление из ЗМ “устаревшей” информации. По некоторому устанавливаемому признаку в РАП производится контроль ассоциации и операция удаления, то есть сброс в нуль нулевого разряда ячеек, отмеченных при контроле.

Для ассоциативной памяти не может использоваться динамическое ЗУ, так как считывание производится одновременно во всем ЗМ. Поэтому используются элементы памяти, не допускающие разрушения информации при считывании.

Зачем нужна иерархия памяти?



Простой кэш

- Полностью ассоциативный: любой блок данных ОЗУ может быть помещён в любое место кэш
- LRU (least recently used) стратегия замещения: при недостатке места выкидывается наиболее давно требовавшиеся данные.

Очень маленький кэш:
4 записи, каждая по 4-х байтному слову, любая запись может содержать любое слово.

Вспомогательное поле, облегчающее определение кандидатов на вылет

Тэг (tag) определяет адреса данных кэш

Пример работы простого кэша

Последовательность ячеек: 24, 20, 04, 12, 20, 44, 04, 24, 44

tag	data	time since last reference
24 - 27	- data -	3
20 - 23	- more data -	2
04 - 07	- etc-	1
12 - 15	- etc -	0

Первые четыре ссылки – «промахи». Кэш был пустым.

24 - 27	- data -	3
20 - 23	- more data -	0
04 - 07	- etc-	2
12 - 15	- etc -	1

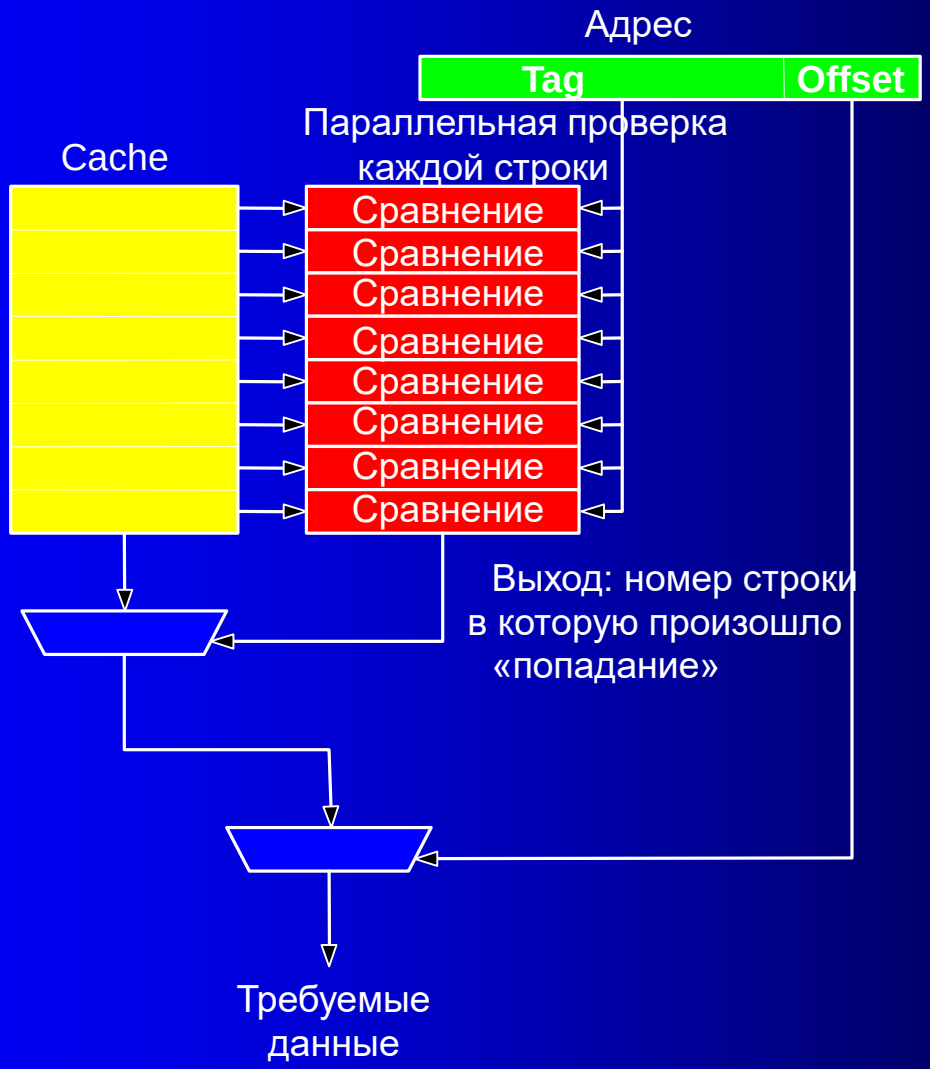
Следующая ссылка (“20”) - попадание. Обновление времени

44 - 47	- new data -	0
20 - 23	- more data -	1
04 - 07	- etc-	3
12 - 15	- etc -	2

“44” – промах, наистарейшие данные (24-27) замещаются.

44 - 47	- new data -	1
20 - 23	- more data -	2
04 - 07	- etc-	0
12 - 15	- etc -	3

Полностью ассоциативное отображение (Fully-Associative Cache)



Прямое отображение

В кэш с прямым отображением (direct mapped cache), каждому блоку памяти поставлена в соответствие отдельная строка кэш-памяти.

i = j mod m, где m – общее число строк в кэш-памяти.

- Обычно* используются несколько бит адреса для «задания» отображения блоков памяти на кэш
- Например, биты 2 и 3 (если считать LSB = "0") адреса – будут индексом.

**В некоторых машинах используется псевдо-случайная хэш-таблица адресов*

Direct mapped cache in action

Последовательность ячеек: 24, 20, 04, 12, 20, 44, 04, 24, 44

index	tag	data
00	-	-
01	20 - 23	data
10	24 - 27	data
11	-	-

24 = 011000_2 ; индекс - 10.

20 = 010100_2 ; индекс - 01.

(* index is bits 2-3 of address)

00	-	-
01	04 - 08	data
10	24 - 27	data
11	-	-

04 = 000100_2 ; индекс - 01.

(выбрасывает 20-23 из кэш)

00	-	-
01	04 - 08	data
10	24 - 27	data
11	12 - 15	data

12 = 001100_2 ; индекс - 11.

your turn ...

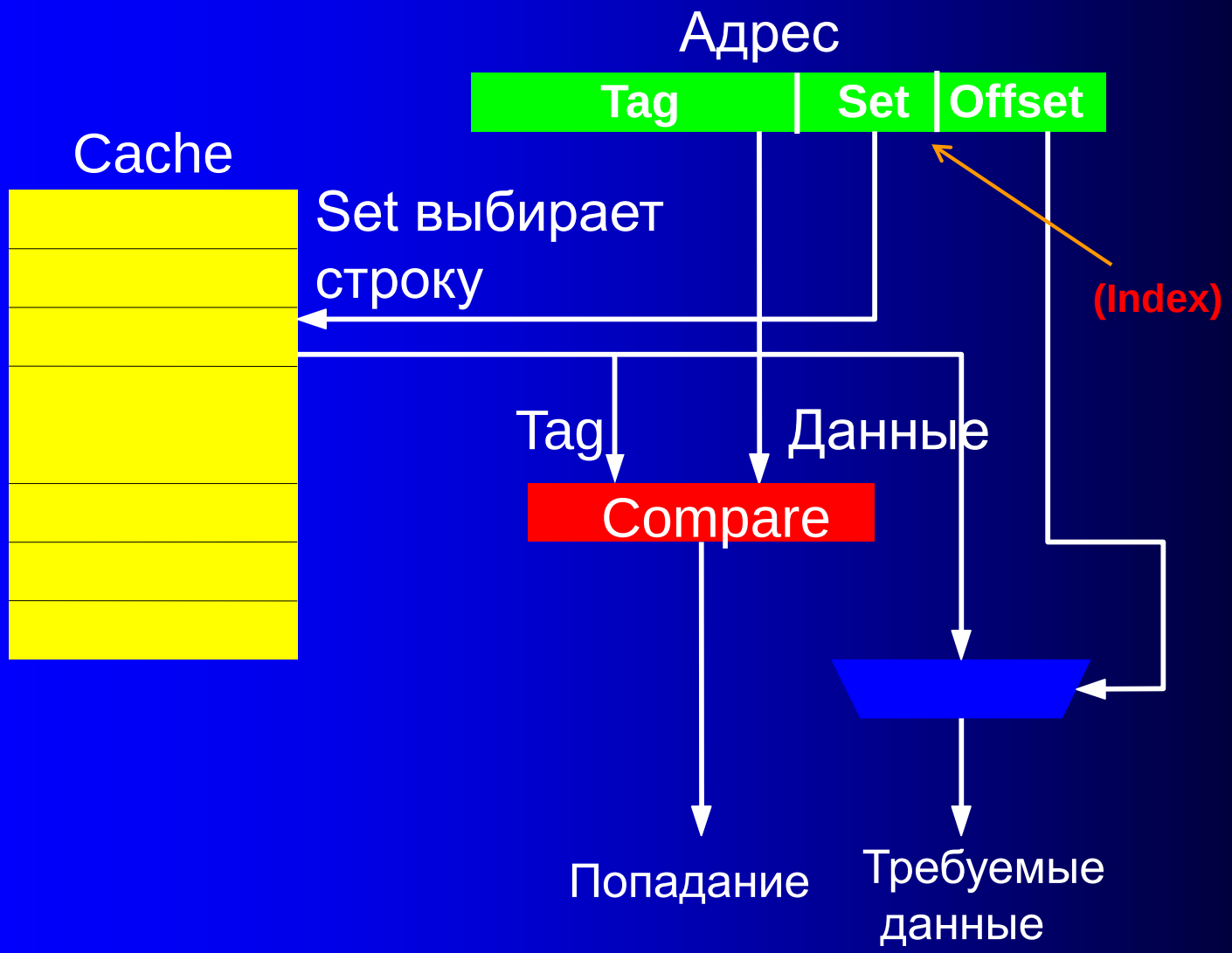
20 = 010100_2

44 = 101100_2

04 = 000100_2

00	-	-
01	04 - 08	data
10	24 - 27	data
11	12 - 15	data

Direct-Mapped Cache

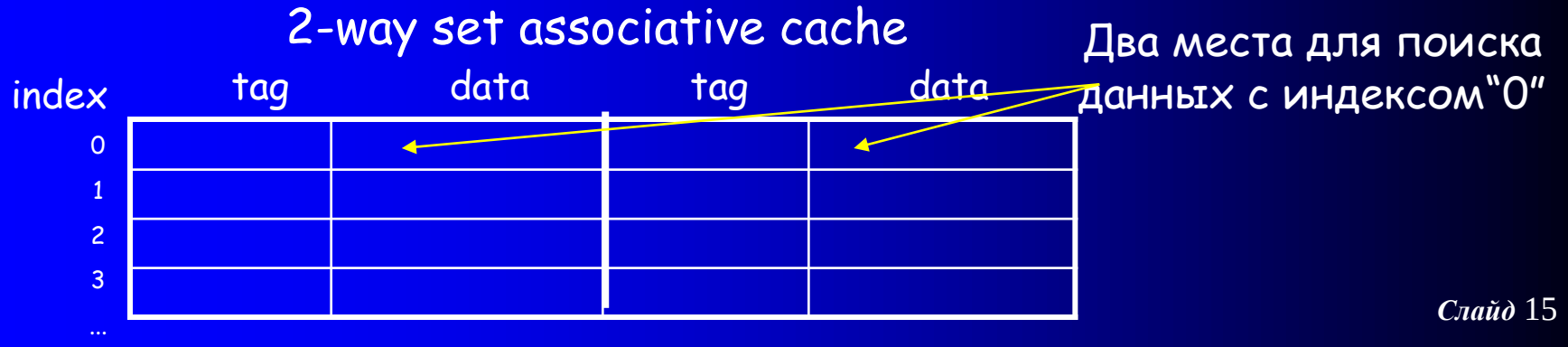


Direct-Mapped vs. Fully-Associative

- Прямое отображение (Direct-Mapped)
  - Требует меньше места
    - Только один компаратор
    - Требуется меньше битов для тэгов
  - Быстрый: может возвращать данные процессору параллельно с определением попадания или промаха
  - Поочерёдное обращение к словам из разных блоков, размещаемых в одной строке кэш (Conflict misses) – снижение вероятности попадания
- Полностью ассоциативное отображение (Fully-Associative)
  - Нет конфликтов (conflict misses) –> общая высокая вероятность попадания
  - Требуется дорогая ассоциативная память – отдельный компаратор для каждой строки кэш

Множественно-ассоциативное отображение (k-way set associative cache)

- Direct mapped кэш проще:
  - Меньше аппаратуры; более быстр в потенциале
- Fully associative - меньше промахов и конфликтов.
- Кэш, совмещающий преимущества обоих способов:
  - Индекс вычисляется из адреса
  - В "k-way set associative cache" индекс определяет множество k модулей кэш, где могут храниться данные (т.е. k вариантов размещения для каждого блока данных ОЗУ).
    - k=1 – прямое отображение (direct mapped).
    - k= размеру кэша (в строках) – полностью ассоциативное.
  - Используется замещение LRU (или другое) для всего набора.



2-way set associative cache in action

Последовательность ячеек: 24, 20, 04, 12, 20, 44, 04, 24, 44

index	tag	data	tag	data
0	24 - 27	data	-	-
1	20 – 23	data	-	-

24 = $011\underline{0}00_2$; индекс- 0.
20 = $010\underline{1}00_2$; индекс- 1.
(index is bit 2 of address)

0	24 - 27	data	-	-
1	20 – 23	data	04 – 07	data

04 = $000\underline{1}00_2$; индекс- 1.
(-во 2ой slot of "01" set)

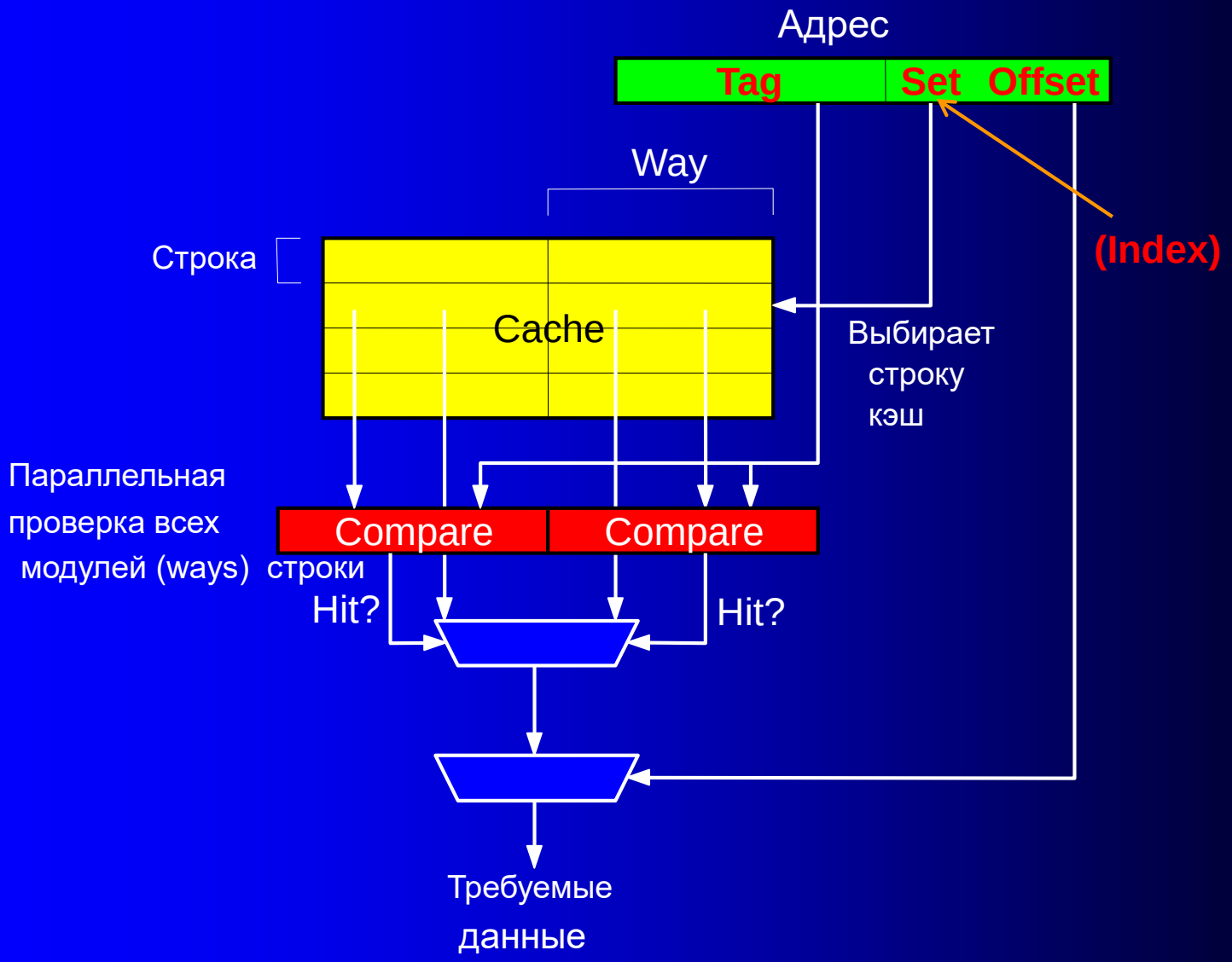
0	24 - 27	data	-	-
1	12 – 15	data	04 – 07	data

12 = $001\underline{1}00_2$; индекс- 1.
(kicks out older item in "01" set)

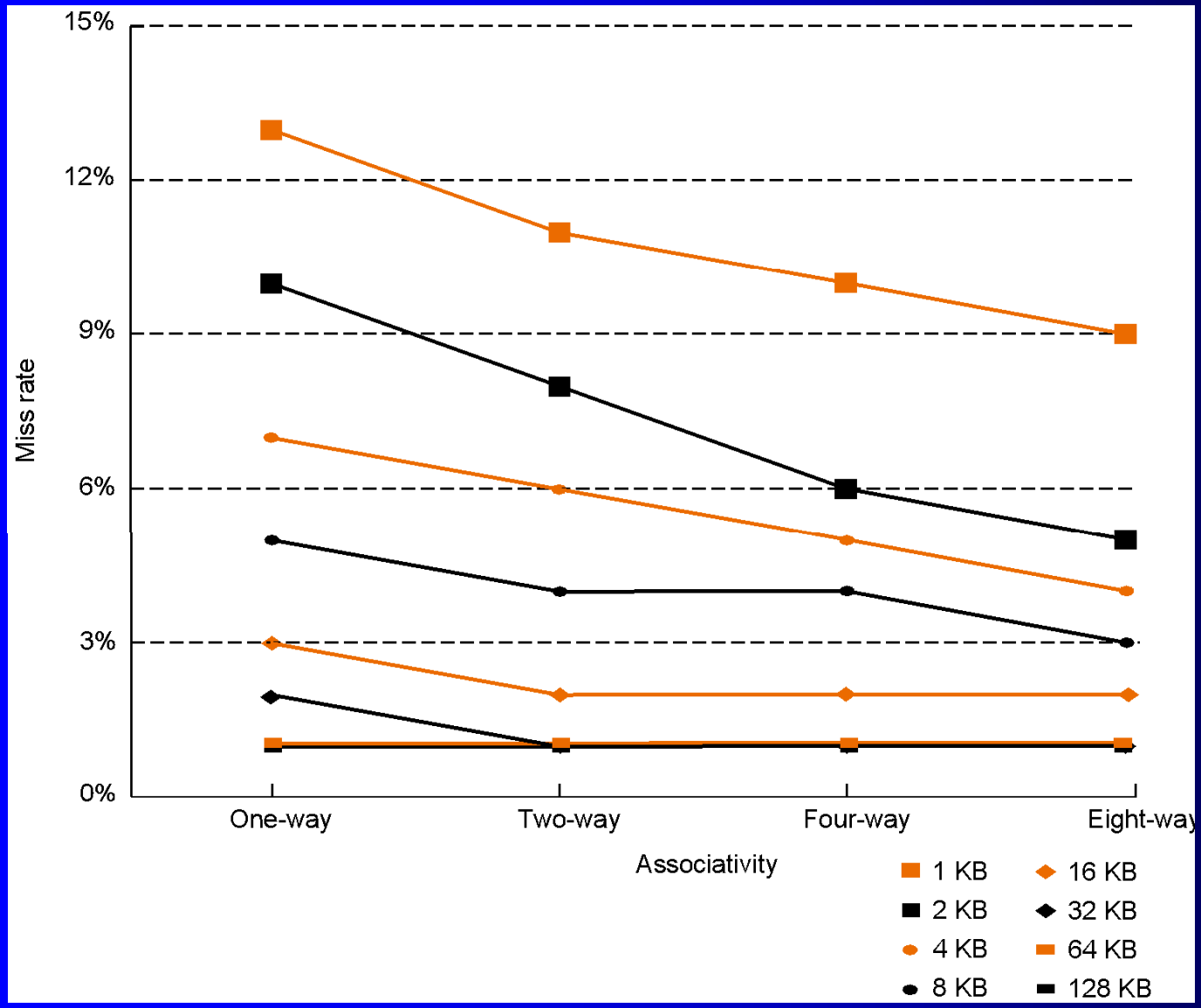
your turn ...
20 = $010\underline{1}00_2$
44 = $101\underline{1}00_2$
04 = $000\underline{1}00_2$

0	24 - 27	data	-	-
1	12 – 15	data	04 – 07	data

Set-Associative Caches

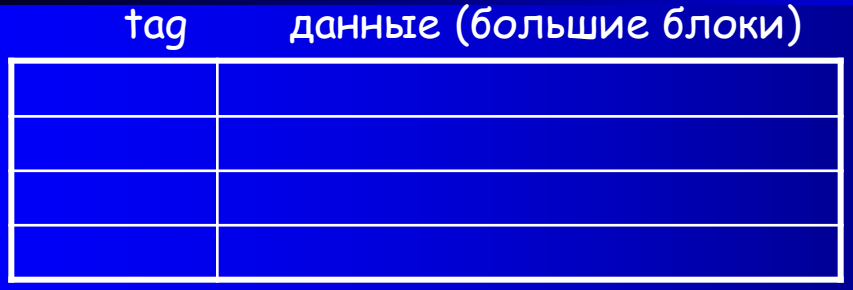


Эффективность множественно-ассоциативного отображения (cache associativity)



4-way cache – почти идентичная вероятность попадания как и у direct-mapped cache двойного размера

Размеры блоков данных в кэш? (Cache Blocks)



Плюсы:

- Большие блоки имеют преимущество в пространственной локальности.
- Меньше места требуется для тэгов (при заданном объёме кэш)

Минусы:

- Слишком большие блоки – неразумное использование пространства кэш (больше вероятность промаха).
- Большие блоки – больше времени для пересылки данных.

Пример использования больших блоков в кэш

Последовательность ячеек: 24, 20, 04, 12, 20, 44, 04, 24, 44…

index	tag	8 Bytes of data
0	-	-
1	24 – 31	data

0	16 - 23	data
1	24 – 31	data

0	16 - 23	data
1	24 – 31	data

$24 = 011\underline{1}000_2$; индекс- 1.
(index is bit 3 of address)

$20 = 010\underline{1}00_2$; индекс- 0.
(линия - Bytes 16-23 –
адреса линии кратны 8 байтам)

$28 = 011\underline{1}00_2$; индекс- 1.
Попадание - даже не смотря, что до этого 28 не было!

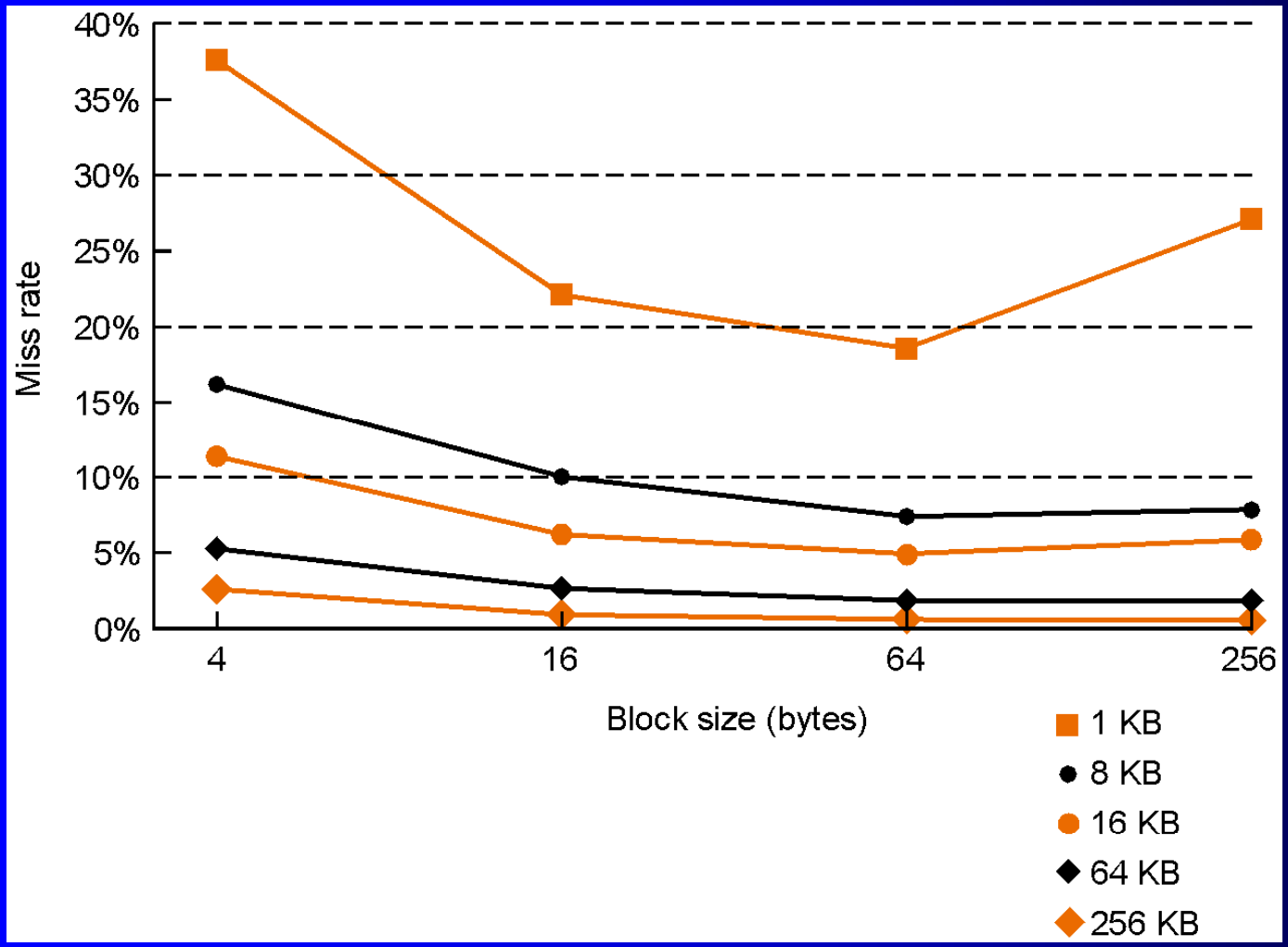
your turn …

$12 = 00\underline{1}100_2$

$08 = 00\underline{1}000_2$

$44 = 10\underline{1}100_2$

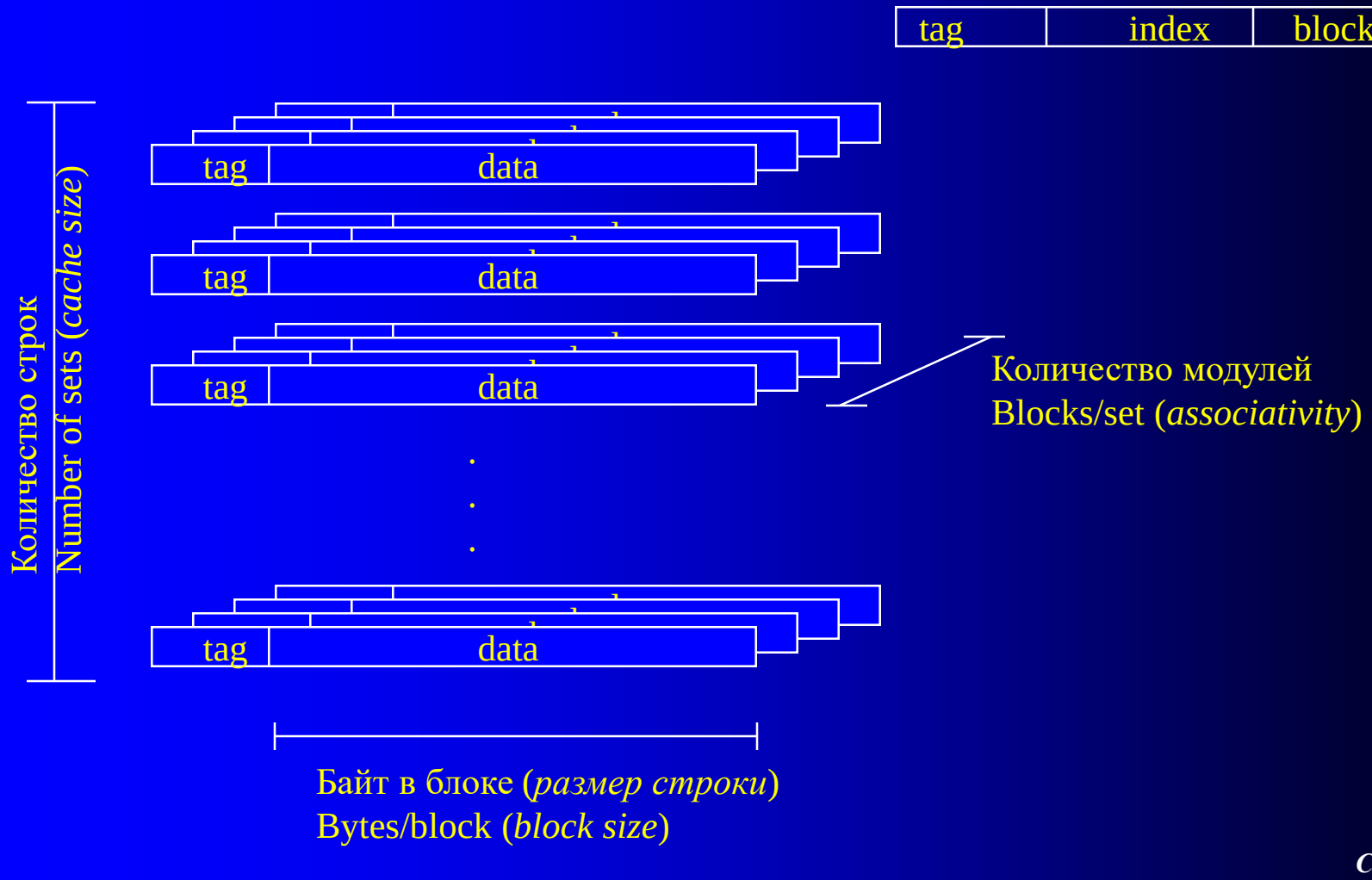
Block Size and Miss Rate



Rule of thumb: block size should be less than square root of cache size.

Итого: организация кэш

- Обычный кэш – трёхмерная структура



Определение параметров кэш

Cache size = Number of sets * block size * associativity.

128 blocks, 32-byte blocks, direct mapped,
size = ? 4096 байт

128 KB cache, 64-byte blocks, 512 sets,
associativity = ? 4

Определение параметров кэш

- Какие биты и (сколько) должны быть выбраны для
  - тэга?
  - индекса?
  - смещения?

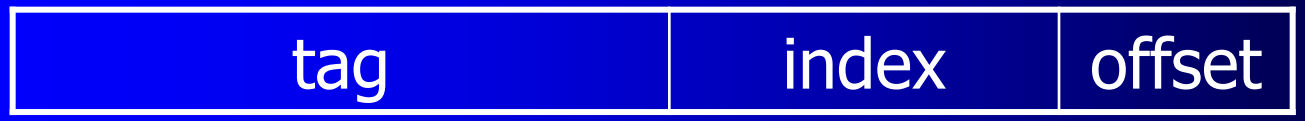
Биты для индекса

Если длина блока – n байт, младшие биты $\log_2 n$ адреса байта – дают смещение внутри блока данных.

Следующая группа битов index (или set) – определяет количество номер строки кэш памяти. Соответственно, в случае, если кэш содержит X строк в каждом модуле, то количество модулей при фиксированном размере кэш будет k/X, где k-общее количество строк.

Оставшиеся биты - тэги tag.

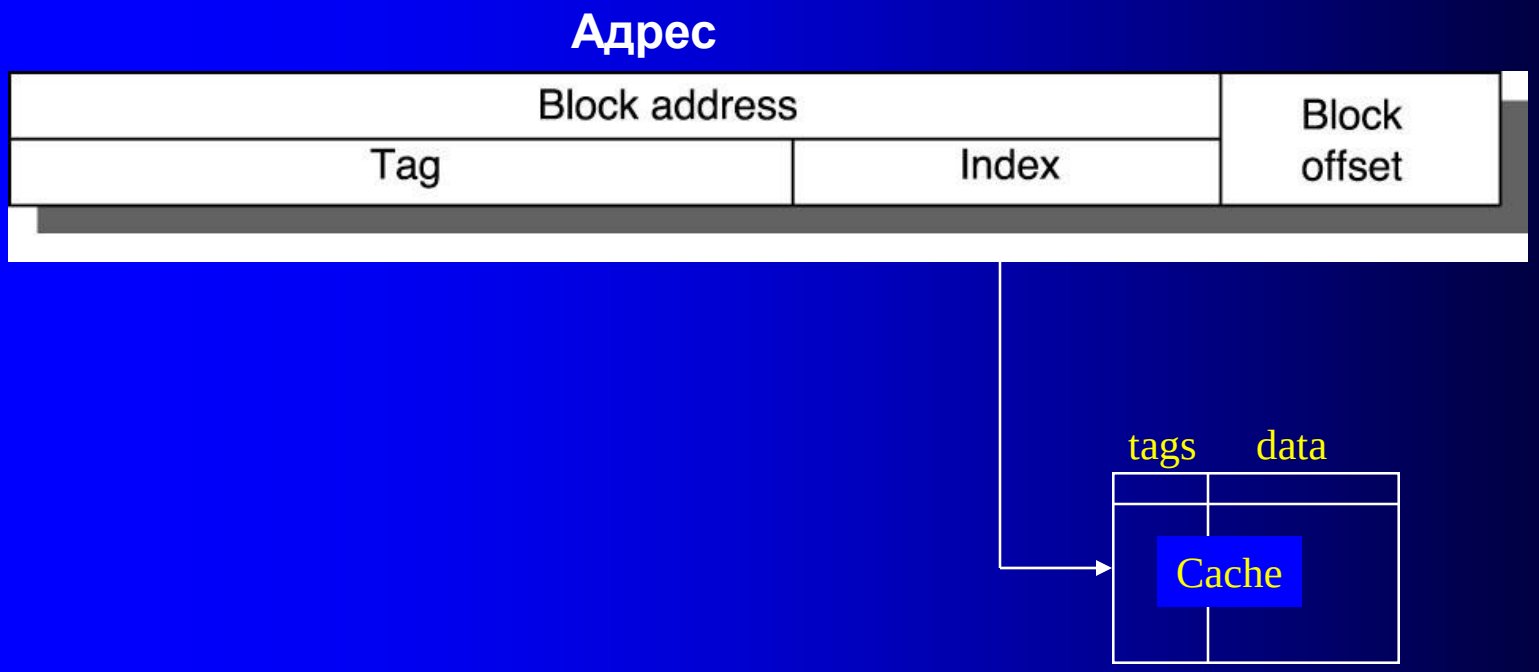
Анатомия адреса:



Условие размещения в кэш любого последовательного (n байт) блока памяти - каждый блок начинается с байта, адрес которого кратен размеру блока.

Как блок ищется в кэш?

- Тэг для каждого блока
  - Для поиска нет необходимости проверять индекс либо смещение внутри блока



Пример адресации

Ёмкость кэш - 32 KB, строки по 256 байт.

ША – 32 разряда -> (виртуальная память 2^{32} - 4Гб) сколько бит на tag, set, and offset для

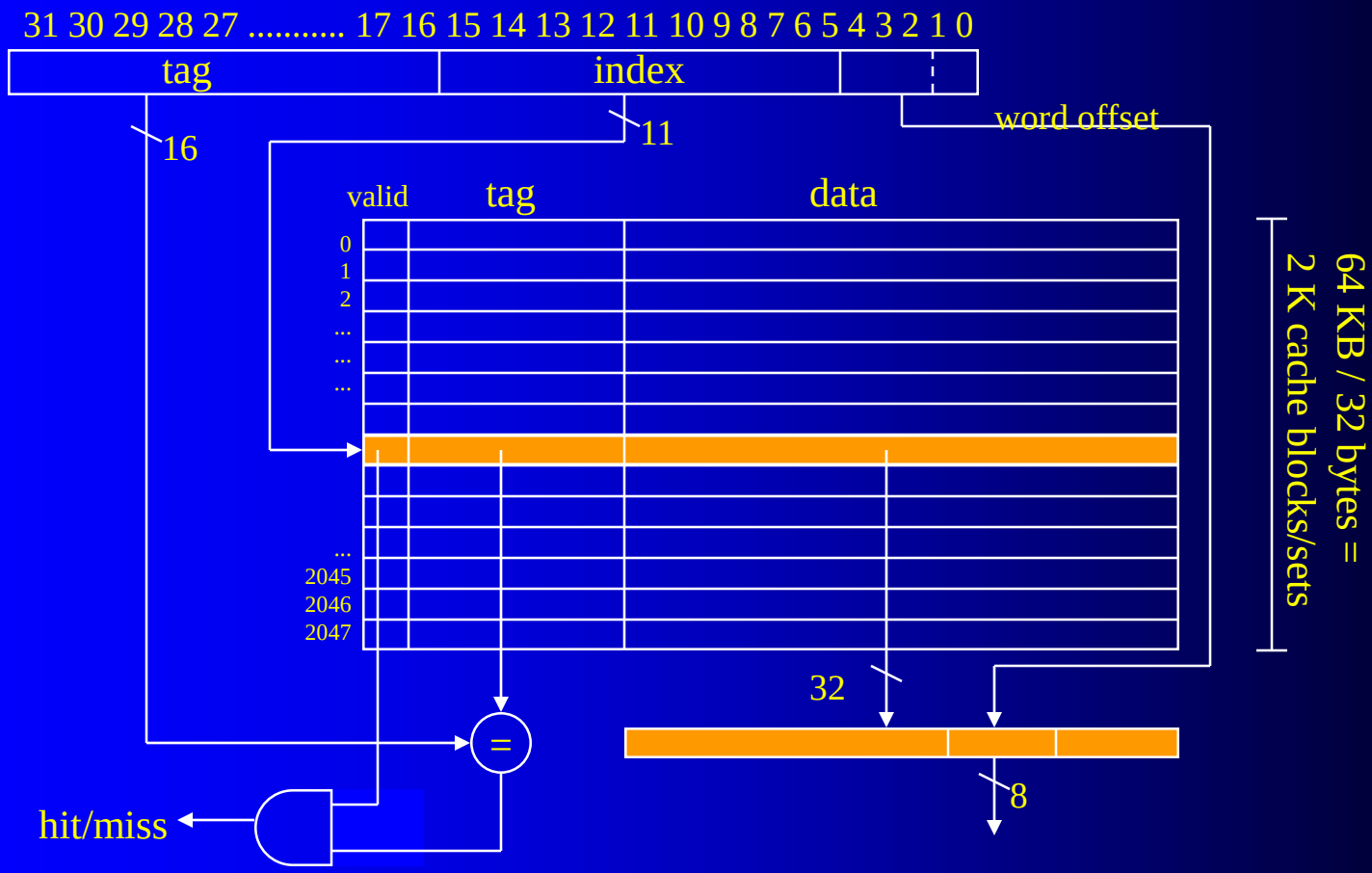
- Прямого отображения (direct-mapped implementation)?
- 4-way set-associative implementation?
- Полностью ассоциативного отображения (fully-associative implementation)?

Пример

- Смещение - 8 бит для всех версий кэш
- Все версии по 128 строк
- Прямое отображение - 128 sets -> по 7 bits для адресации:
  - Offset - 8 bits, set - 7 bits, tag =32 – (8 + 7) = 17 bits
- 4-way set-associative - 128/4 = 32 sets -> 5 bits для выбора модуля
  - Offset - 8 bits, set - 5 bits, tag = 32 – (8 + 5) = 19 bits
- Полностью ассоциативная память - 1 set, соответственно – поле set =0
  - Offset - 8 bits, set - 0 bits, tag - 32 – 8 = 24 bits

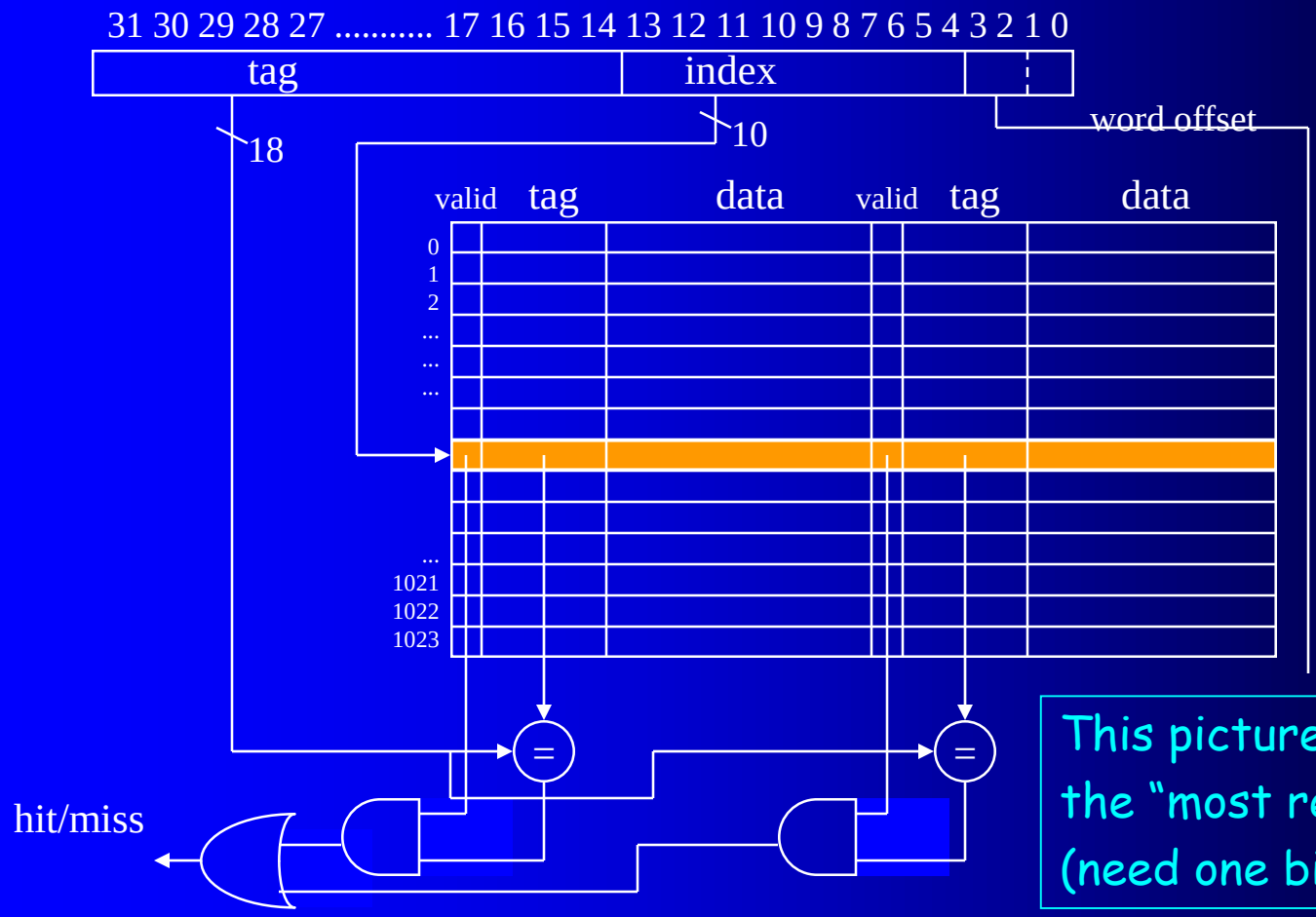
Итого: (Putting it all together)

64 KB cache, direct-mapped, 32-byte cache block



Множественно-ассоциативный кэш (a set associative cache)

32 KB cache, 2-way set-associative, 16-byte blocks



This picture doesn't show the "most recent" bit (need one bit per set)

Thought Experiment

- 16 KB, 4-way set-associative cache, 32-bit address, byte-addressable memory, 32-byte cache blocks/lines
- Сколько бит на tag?
- Где будет размещено слово с адресом 0x200356A4?