Topic V31

Block Size and Associativity

Reading: (Section 5.3)

Real Processor Caches

Intel Core duo (Yonah) has a 32 KB L1 cache that is 8-way set associative with 256-bit blocks
(http://www.xbitlabs.com/articles/mobile/display/core2duo_2.html)

Assuming a 64-bit address, what is the address mapping (bits used for tag, index and offset) for this cache?

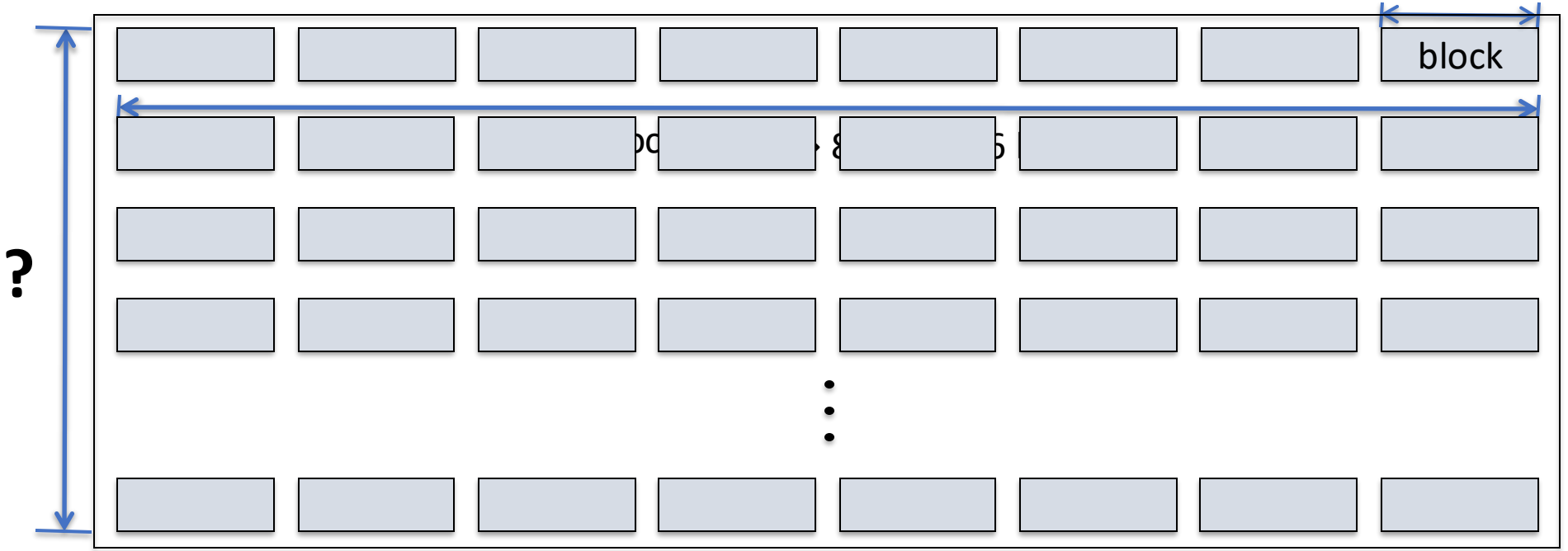
- We need to answer these questions:
  - How many bytes per cache block (offset bits)
  - How many sets are there in the cache (index bits)
    - First find how many bytes per set

Intel Core duo (Yonah) has a 32 KB L1 cache that is 8-way set associative with 256-bit blocks and 64-bit address

Whole cache stores 32 KB = 2^{15} bytes

5 offset bits

32 bytes
256 bits

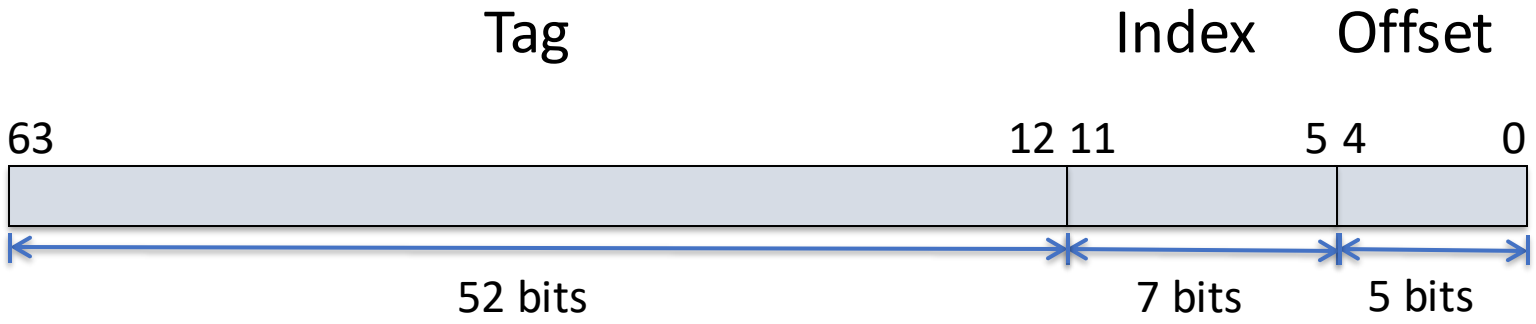


$$\frac{32\ \text{KB}}{256\ \frac{\text{bytes}}{\text{set}}} = \frac{32 \times 1024\ \text{bytes}}{256\ \frac{\text{bytes}}{\text{set}}} = 32 \times 4\ \text{sets} = 128\ \text{sets} \Rightarrow 7\ \text{index bits}$$

64 address bits – 7 index bits – 5 offset bits = 52 tag bits

Intel Core duo (Yonah) has a 32 KB L1 cache that is 8-way set associative with 256-bit blocks and 64-bit address

offset bits: 5
index bits: 7
tag bits: 52

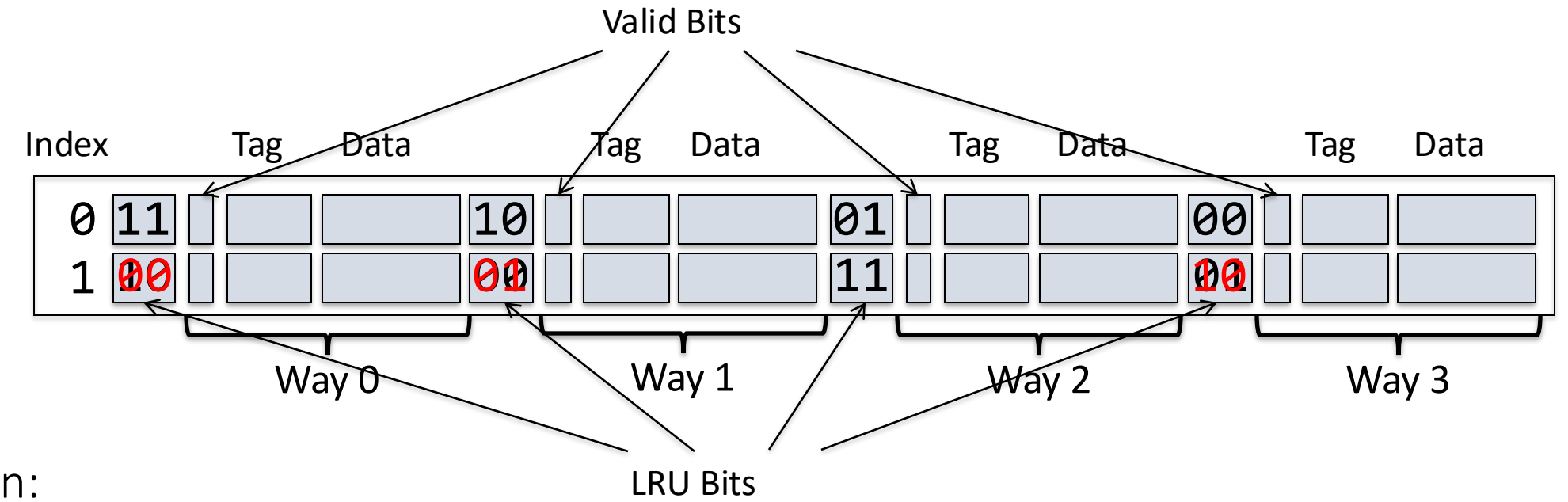


LRU for Higher Associativity

How can LRU work for higher associativity?

True LRU is expensive, lets see an example

LRU for a 4-way Set Associative Cache (example)



Assumption:

11: Least recently used

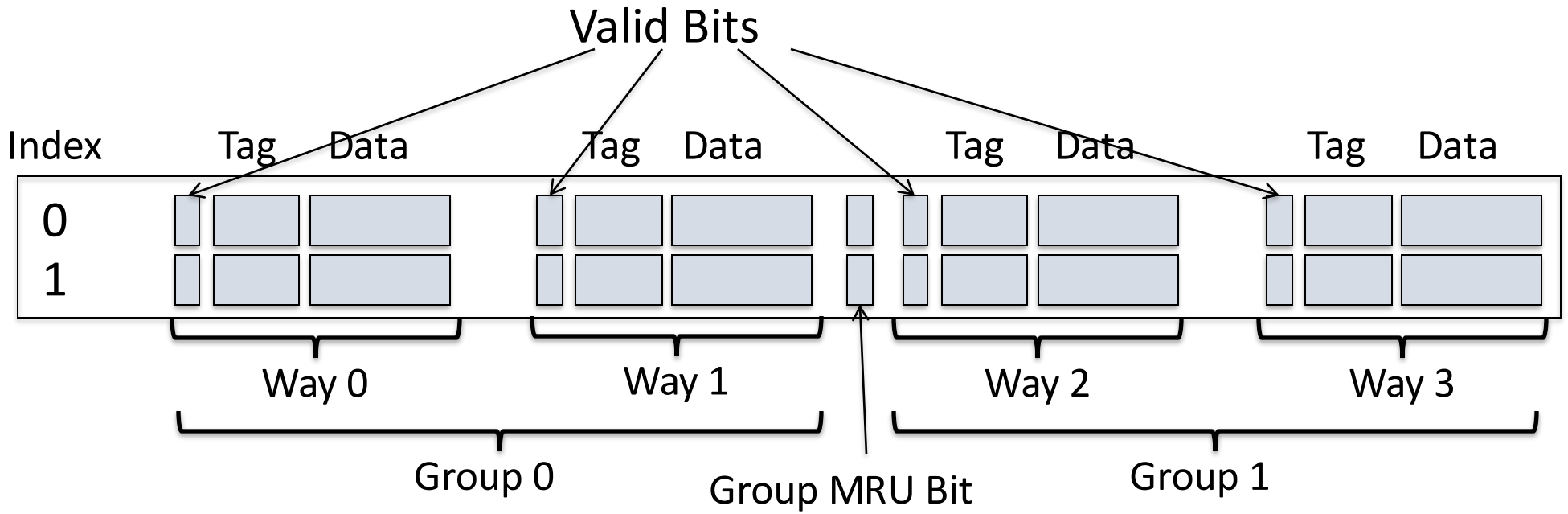
10: Second least recently used

01: Third least recently used

00: Most recently used.

For true LRU, if a reference with index 1 that matches on way 0 occurs, what must happen?

Alternative to True LRU: Not Most Recently Utilized



Ensures that replacement is not in the MRU group

Way selection of the non-MRU group is random

Tree-Based Algorithm (LRU approximation)

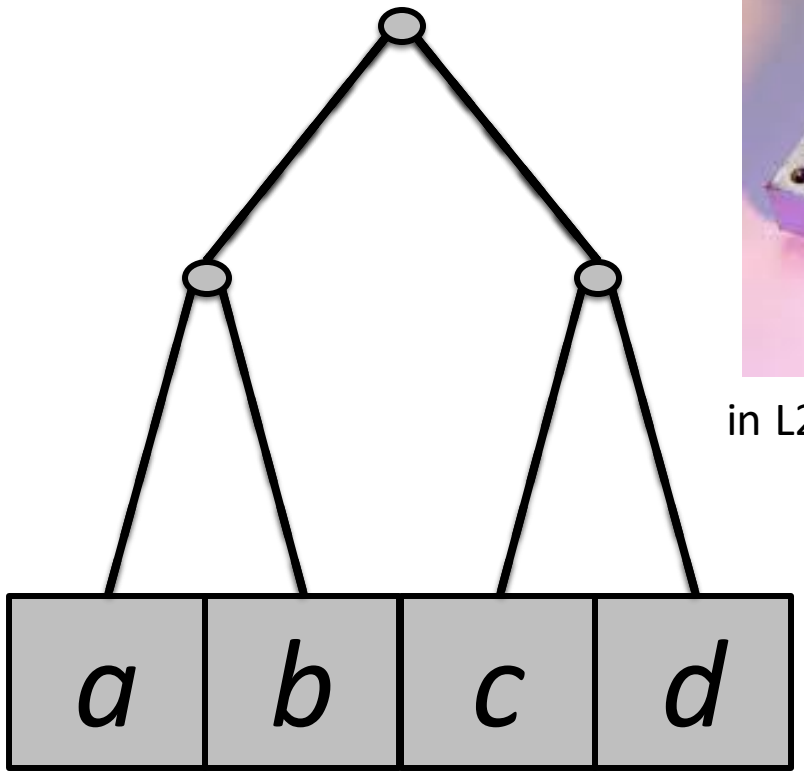
- Example: 4-way set associative cache.

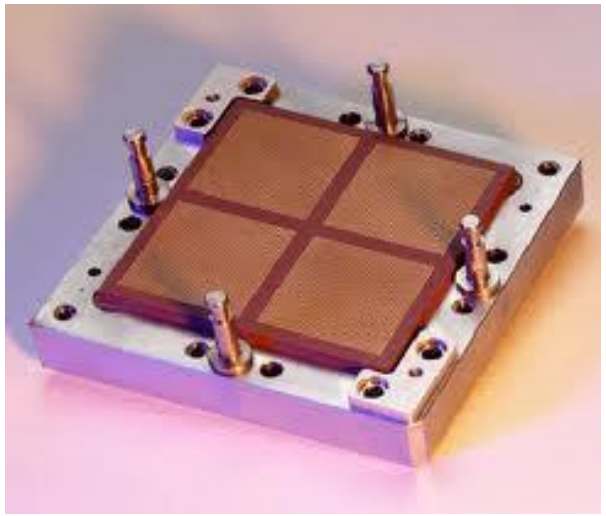
References:

b c a d

MRU Path

Entry to Replace





in L2 cache of IBM Power4

Requires n-1 bits per set for a n-way set-associative cache.

Baer, p. 235

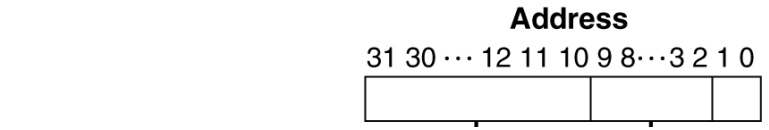
How Much Associativity

Increased associativity decreases miss rate

- But with diminishing returns

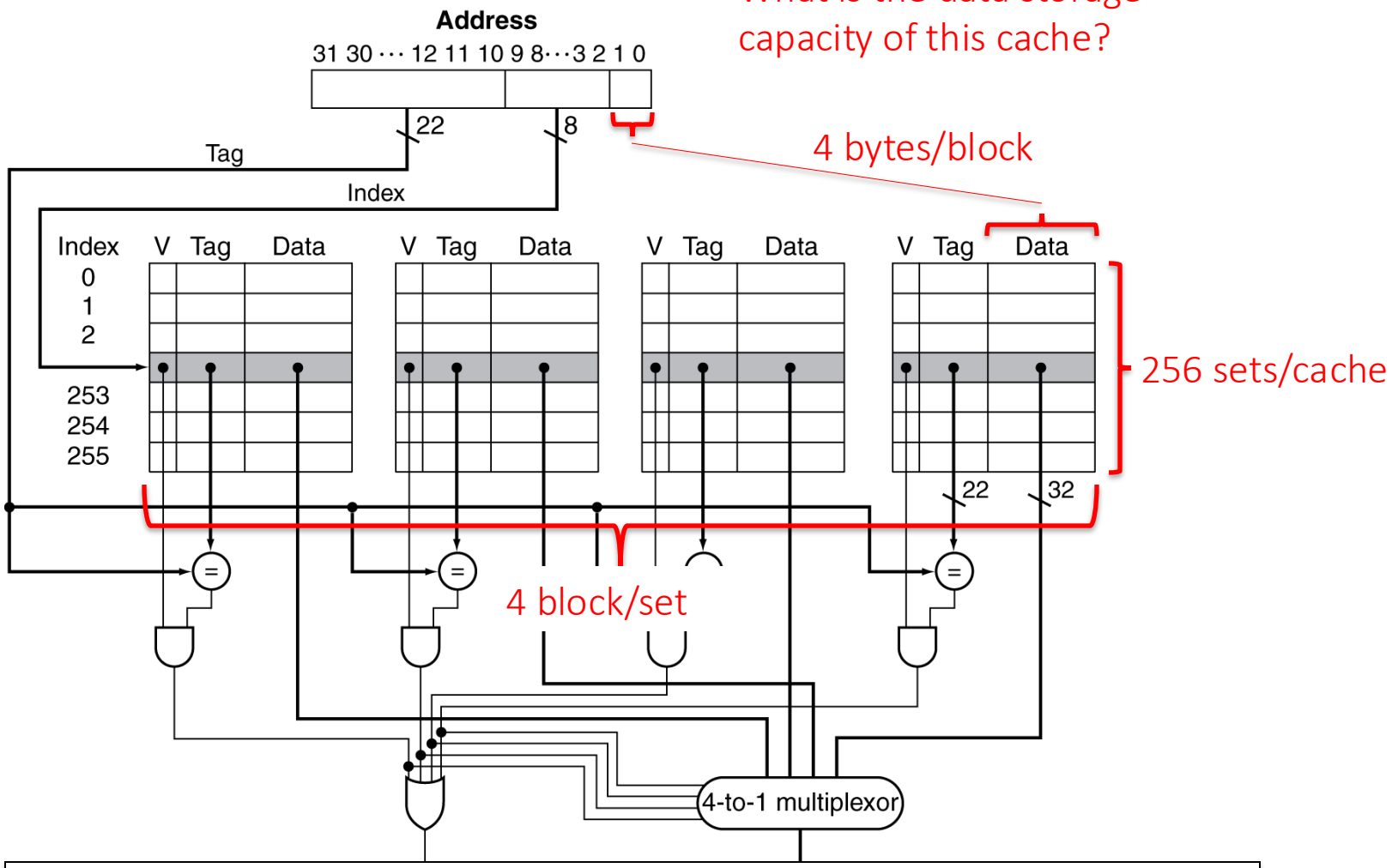
- Simulation of a system with 64KB D-cache, 16-word blocks, SPEC2000
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%

Set-associative Cache Organization



Set-associative Cache Organization

What is the data storage capacity of this cache?



$$Storage = 4\frac{bytes}{block} \times 4\frac{blocks}{set} \times 256\frac{sets}{cache} = 4096\frac{bytes}{cache}$$

Replacement Policy

Direct mapped: no choice

Set associative

- Prefer non-valid entry, if there is one
- Otherwise, choose among entries in the set

  - Least-recently used (LRU)
    - Choose the one unused for the longest time
      - Simple for 2-way, manageable for 4-way, too hard beyond that

  - Random
    - Gives approximately the same performance

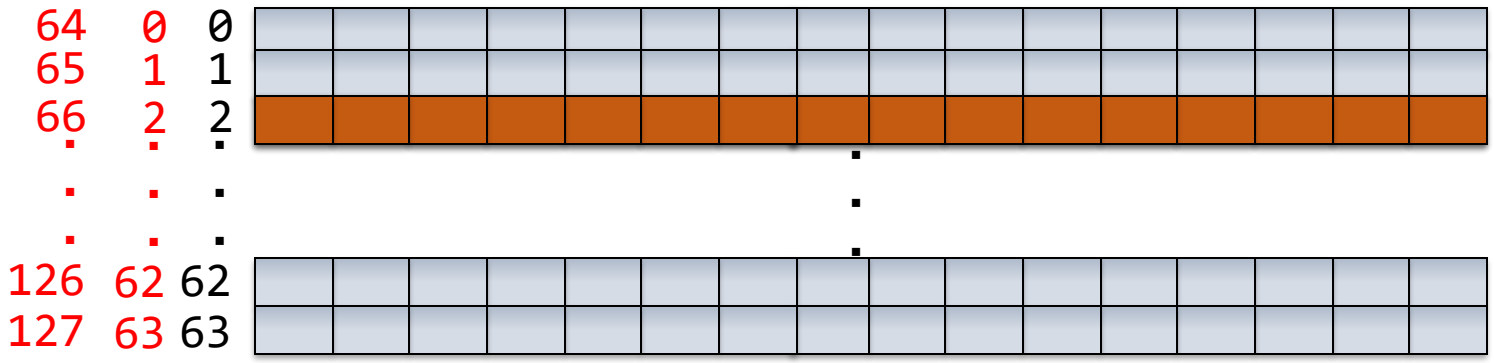
Larger Block Size (example)

Cache with 64 blocks, 16 bytes/block

Map address 1060_{10} to its cache block

$$\text{Block address} = \lfloor 1060/16 \rfloor = 66$$

$$\text{Block number} = 66 \text{ modulo } 64 = 2$$



Block Size Considerations

- Larger blocks should reduce miss rate
  - Due to spatial locality
- But in a fixed-sized cache
  - Larger blocks ⇒ fewer of them
    - More competition ⇒ increased miss rate
  - Larger blocks ⇒ pollution
- Larger miss penalty
  - Can override benefit of reduced miss rate