COMPUTER ORGANIZATION

Lecture 12
Memory Hierarchy (2)

2024 Spring

Recap

- Lec11 Memory1
  - Storage technology
    - Storage hierarchy
      - upper level: fast, small, expensive
      - lower level: slow, big, cheap
    - Principle of Locality
      - temporal
      - spatial
  - Cache
    - basics
      - block placement
      - block finding
      - block replacement
      - write policy
    - hit/miss
      - hit rate, miss rate
      - read hit
      - read miss
        - CPU stalls
      - write hit
        - write through/write back
      - write miss
        - write allocation/no write allocation
    - direct mapped cache
      - address subvision
        - tag, index, offset
      - cache implementation
        - cache line: valid bit, tag, data

Outline

- **Measuring cache performance**
- Improving performance – Associative cache
  - Fully associative
  - n-ways Set associative
- Improving performance – Multilevel Caches

Measuring Cache Performance

- Components of CPU time
  - Program execution cycles
    - Includes cache hit time
  - Memory stall cycles
    - Mainly from cache misses
- With simplifying assumptions:

$$
\begin{aligned}
&\text{Memory stall cycles} \\
&= \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty} \\
&= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty}
\end{aligned}
$$

Cache Performance Example

- Given
  - I-cache miss rate = 2%
  - D-cache miss rate = 4%
  - Miss penalty = 100 cycles
  - Base CPI (ideal cache) = 2
  - Load & stores are 36% of instructions
- Miss cycles per instruction
  - I-cache: 2% × 100 = 2
  - D-cache: 36% × 4% × 100 = 1.44
- Effective CPI = 2 + 2 + 1.44 = 5.44
  - Ideal CPU is 5.44/2 =2.72 times faster

Average Access Time

- Hit time is also important for performance
- Average memory access time (**AMAT**)

$$\boxed{\text{AMAT} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}}$$

- Example
  - CPU with 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, I-cache miss rate = 5%
  - AMAT = 1 + 0.05 $\times$ 20 = 2cycles = 2ns
    - 2 cycles per instruction

Performance Summary

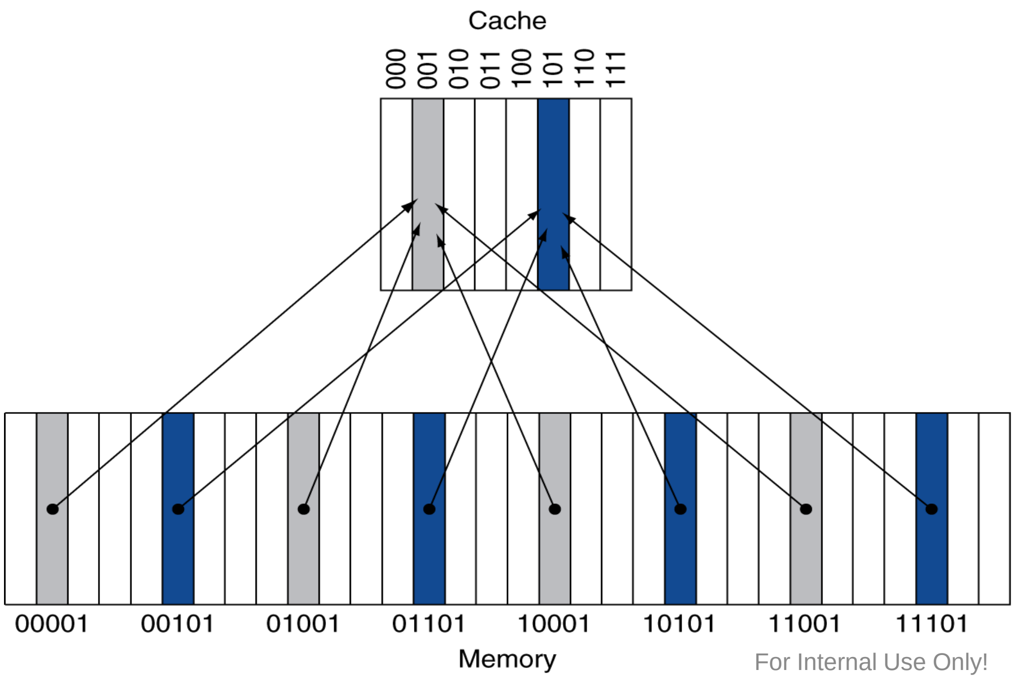
- When CPU performance increased
  - Miss penalty becomes more significant
- Decreasing base CPI
  - Greater proportion of time spent on memory stalls
- Increasing clock rate
  - Memory stalls account for more CPU cycles
- Can’t neglect cache behavior when evaluating system performance

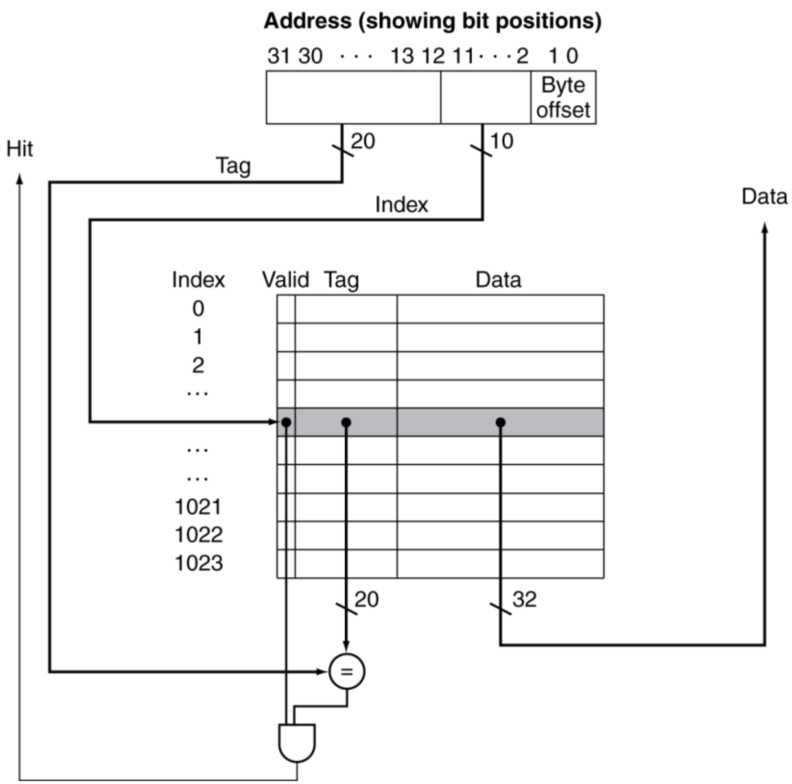
Outline

- Measuring cache performance
- **Improving performance – Associative cache**
  - Fully associative
  - n-ways Set associative
- Improving performance – Multilevel Caches

Recall: Direct Mapped Cache

- Direct mapped cache:
  - Location determined by address
  - One data in memory is mapped to **only one location** in cache
  - Capacity of cache is not fully exploited
  - Miss rate is high





Recall: Direct Mapped Cache

16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

Index	V	Tag	Data
000	**Y**	**10**	**Mem[10000]**
001	N		
010	Y	11	Mem[11010]
011	**Y**	**00**	**Mem[00011]**
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

18	10 010	Miss	010
16	10 000	Hit	000

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	**Y**	**10**	**Mem[10010]**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Associative Caches

- Fully associative
  - Allow a given block to go in any cache entry
  - Requires all entries to be searched at once
  - Comparator per entry (expensive)
- n-way set associative
  - Each set contains *n* entries
  - Block number determines which set
    - (Block number) modulo (#Sets in cache)
  - Search all entries in a given set at once
  - n comparators (less expensive)

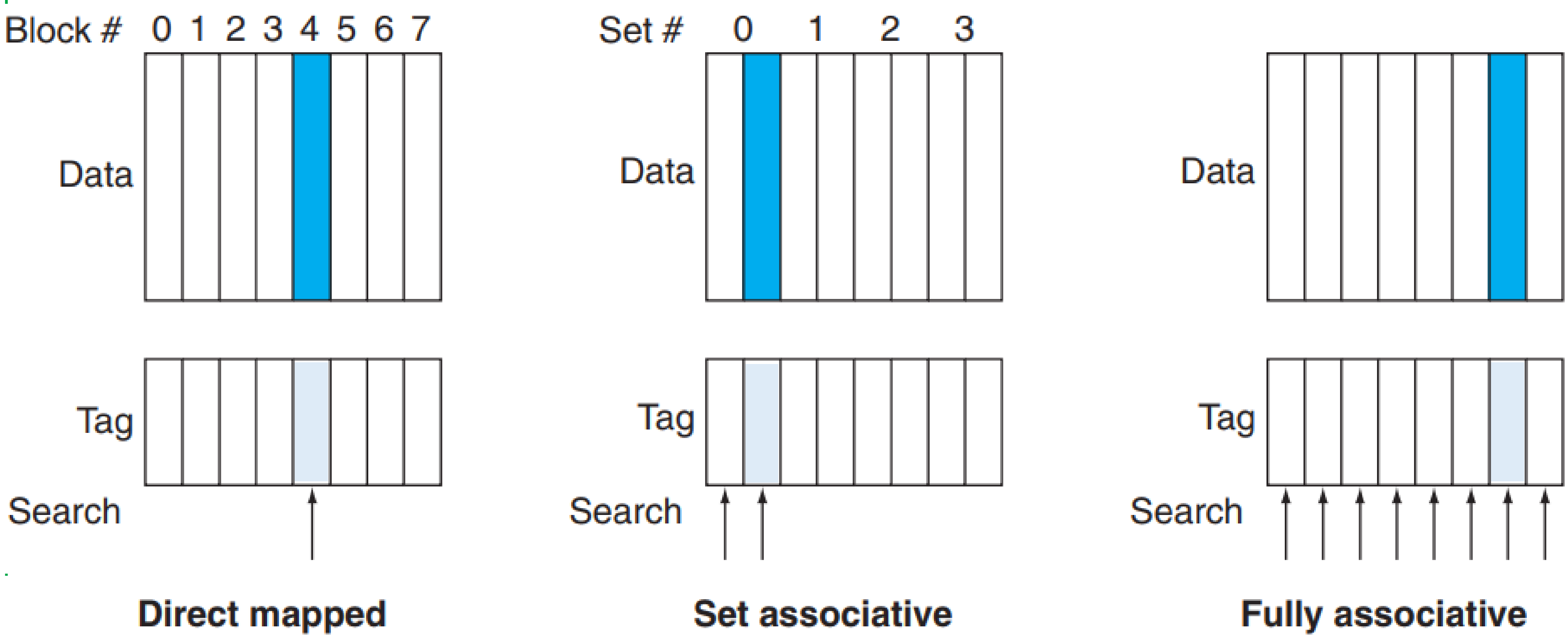
Associative Cache Example

- Example: Placement of a block whose address is 12:

12 mod 8 = block 4

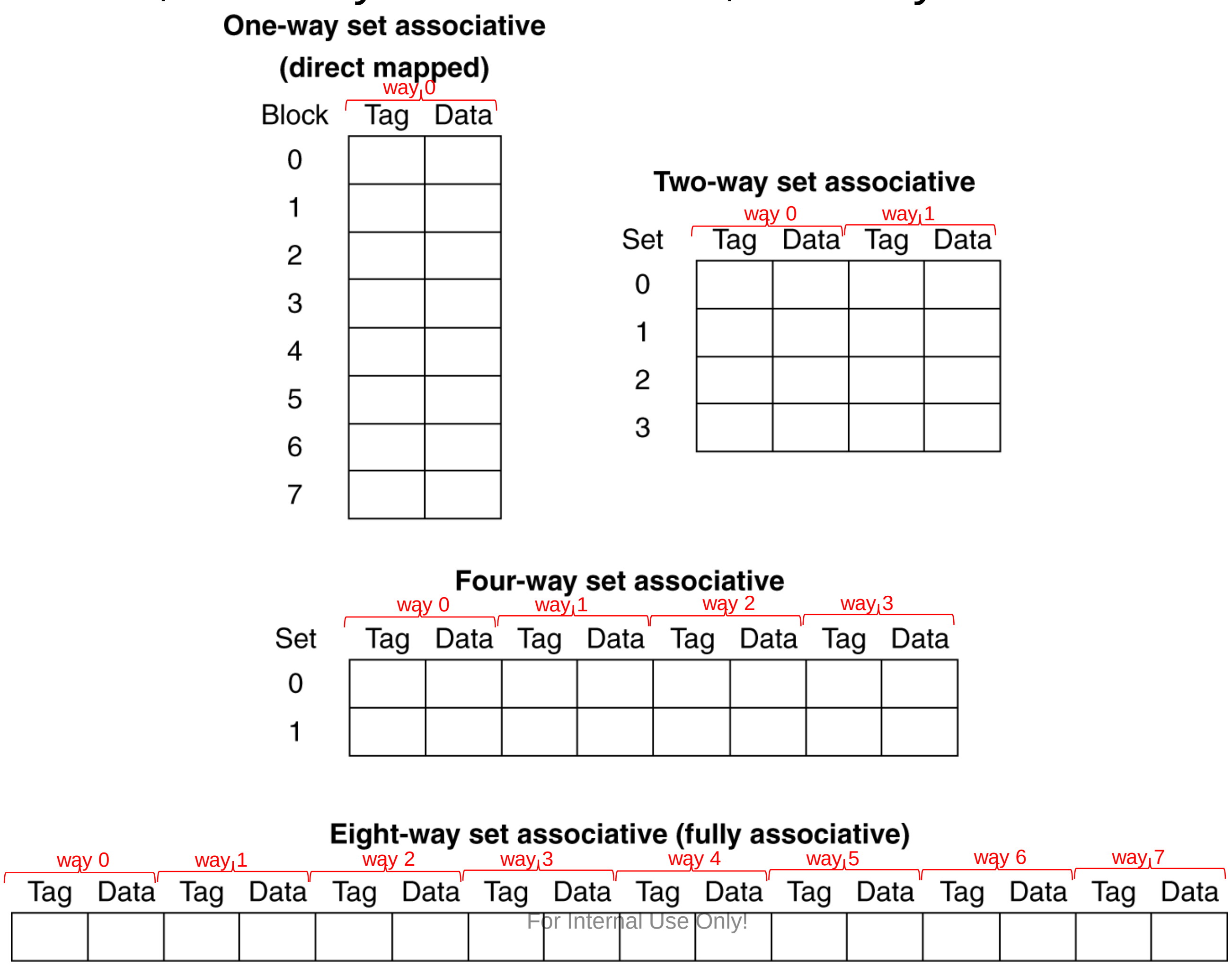
12 mod 4 = set 0, can be in either way of the set

can appear in any of the eight cache blocks



Spectrum of Associativity

- An eight-block cache configured as direct mapped, two-way set associative, four-way set associative, and fully associative.



Associativity Example

- 4-block caches, 1byte/block
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- Direct mapped
  - index = (Block address) mod (**#Blocks**)

needs 2 bit for index

Block address	Cache index	Hit/miss	Cache content after access			
			way 0 block 0	way 0 block 1	way 1 block 2	way 1 block 3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0	miss	**Mem[8]**		Mem[6]	

Associativity Example (cont.)

Block access sequence: 0, 8, 0, 6, 8

- 2-way set associative
  - index = (Block address) mod (**#Sets**)

needs 1 bit for index

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
			way 0	way 1	way 0	way 1
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

- Fully associative (no index)

needs 0 bit for index

Block address		Hit/miss	Cache content after access			
			way 0	way 1	way 2	way 3
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

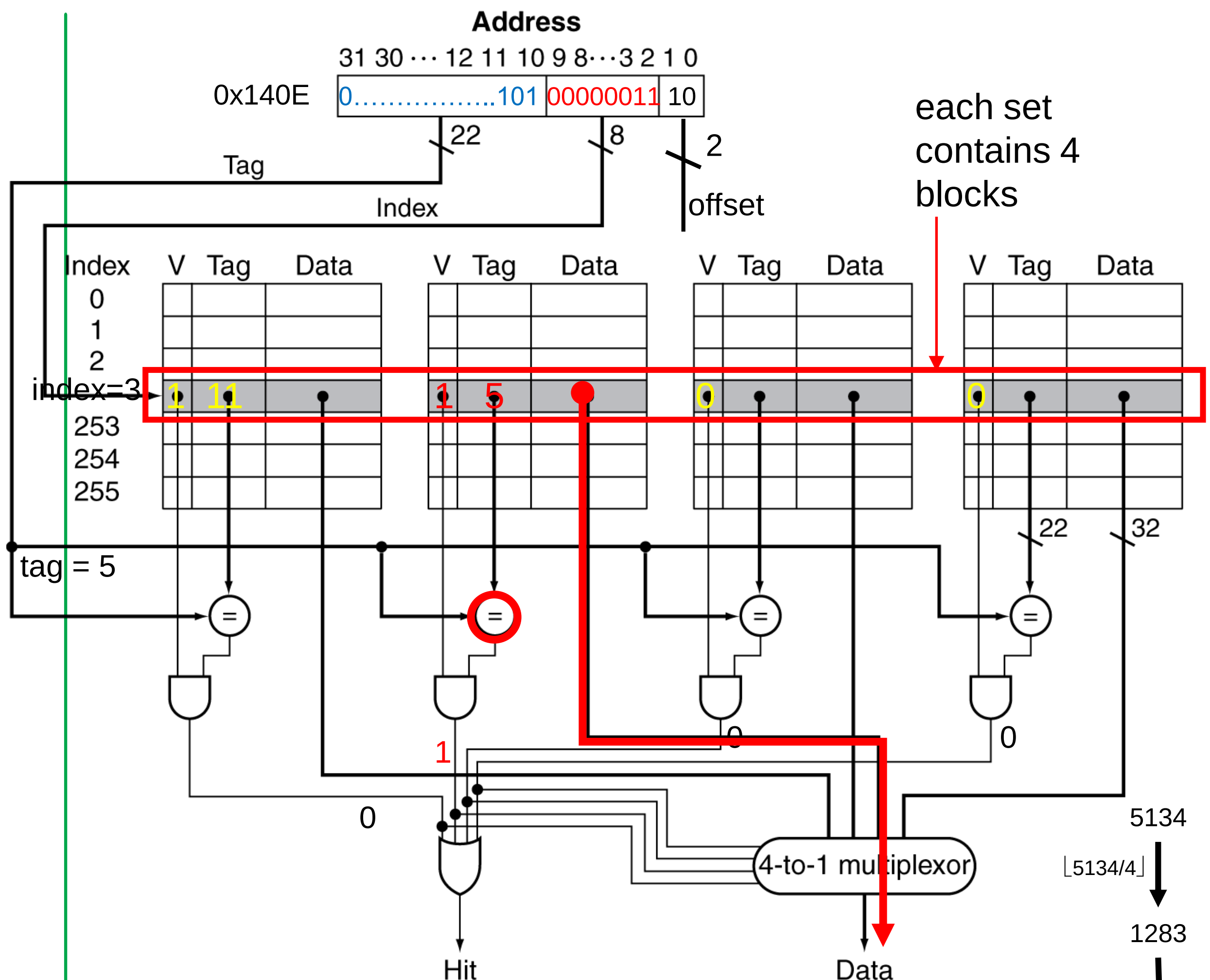
How Much Associativity

- Increased associativity decreases miss rate
  - But with diminishing returns
- Miss rate simulation of a system with 64KB D-cache, 16-word blocks, SPEC2000
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%

Address Subdivision

- 4K blocks cache, 4-word/block, 32-bit address
- offset = $\log_2 16$ = 4bits → 28bits for index + tag for all types
- direct mapped (1-way set associative)
  - same number of sets as blocks →4096 blocks(sets)
  - index = $\log_2 4096$ = 12 bits
  - tag = 32 – 4 – 12 = 16 bits
- 2-way set associative
  - 2 blocks/set → 2048 sets
  - index = $\log_2 2048$= 11 bits
  - tag = 32 – 4 – 11 = 17 bits
- 4-way set associative
  - 4 blocks/set → 1024 sets
  - index = $\log_2 1024$= 10 bits
  - tag = 32 – 4 – 10 = 18 bits
- fully associative
  - just 1 set → no index
  - tag = 32 – 4 = 28

Set Associative Cache Organization

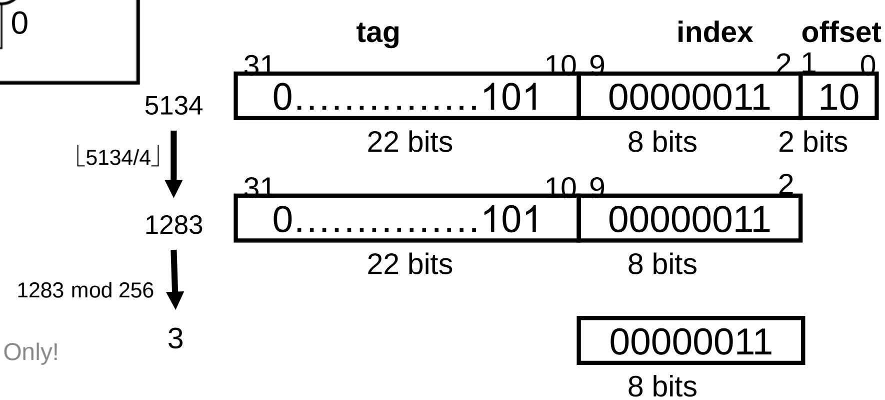


Increasing associativity shrinks index, expands tag

4KB cache, 1word/block
To what **set** number does address 0x140E map?

offset: $\log_2(4\text{byte}) = 2$bits
#blocks: 4KB/4B = 1K blocks
#sets: 1K blocks/4 way = 256 sets
index: $\log_2(256\text{sets}) = 8$bits

$0x140E_{hex} = \underbrace{101}_{\text{tag}}\underbrace{00000011}_{\text{index}}\underbrace{10}_{\text{offset}}{}_{two}$



Replacement Policy

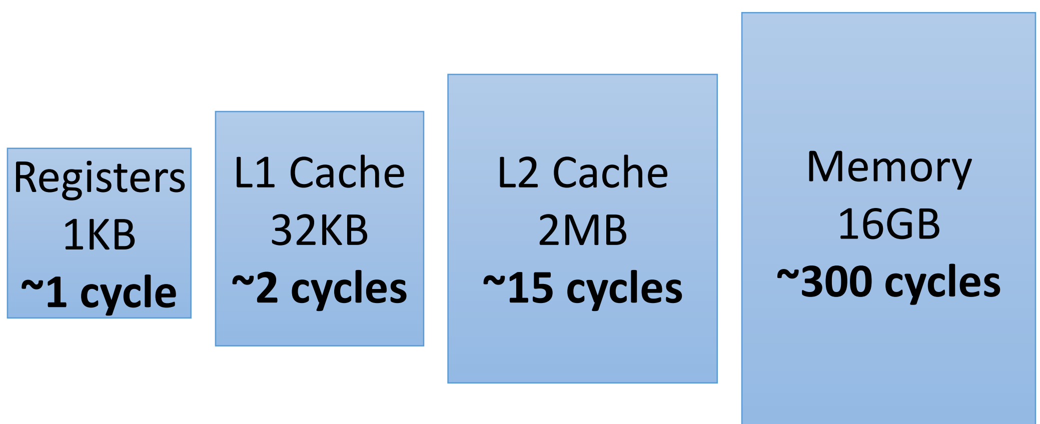
- Direct mapped: no choice
- Set associative
  - Prefer non-valid entry, if there is one
  - Otherwise, choose among entries in the set
- Least-recently used (LRU)
  - Choose the one unused for the longest time
    - Simple for 2-way, manageable for 4-way, too hard beyond that
- Random
  - Gives approximately the same performance as LRU for high associativity

Outline

- Measuring cache performance
- Improving performance – Associative cache
  - Fully associative
  - n-ways Set associative
- **Improving performance – Multilevel Caches**

Multilevel Caches

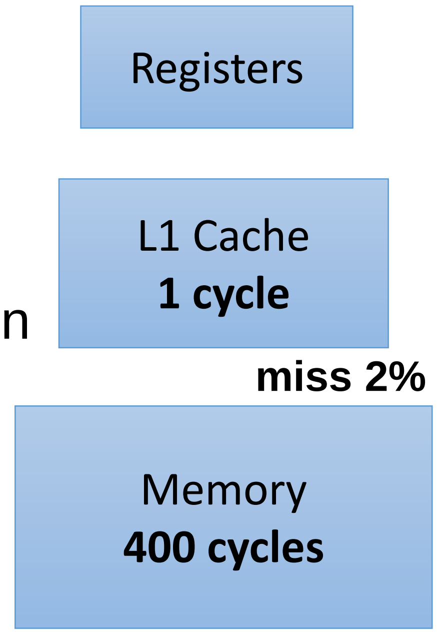
- Primary cache attached to CPU
  - Small, but fast
- Level-2 cache services misses from primary cache
  - Larger, slower, but still faster than main memory
- Main memory services L-2 cache misses
- Some high-end systems include L-3 cache



Multilevel Cache Example

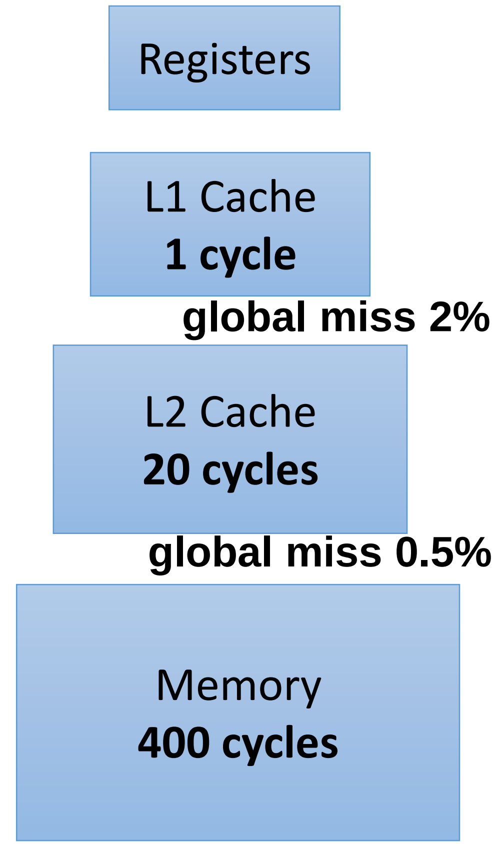
- Given
  - CPU base CPI = 1, clock rate = 4GHz
  - Primary cache Miss rate/instruction = 2%
  - Main memory access time = 100ns
- Solution: With just primary cache
  - Miss penalty = 100ns/0.25ns = 400 cycles
  - Effective CPI = Base CPI + miss penalty/instruction

    = 1 + 2% × 400 = 9



Example (cont.)

- Given: After adding L-2 cache
  - L2 cache Access time = 5ns
  - L2 global miss rate/instruction = 0.5%
- Solution 1, calculate based on Global miss rate (The fraction of references that miss in all levels):
  - L-1 miss (2%) first need to access the L2 cache
    - Penalty = 5ns/0.25ns = 20 cycles
  - L-1 miss with L-2 also miss (0.5%)
    - Extra penalty = 400 cycles
  - Effective CPI = $1 + 2\% \times 20 + 0.5\% \times 400 = 3.4$
  - Speedup = 9/3.4 = 2.6
- Solution 2, calculate based on Local miss rate (The fraction of references to one level of a cache that miss)
  - L2 local miss rate/instruction = 0.5%/2% = 25%
  - Effect CPI = $1 + 2\% \times (20 + 25\% \times 400) = 3.4$



Multilevel Cache Considerations

- Primary cache
  - Focus on minimal hit time
- L-2 cache
  - Focus on low miss rate to avoid main memory access
  - Hit time has less overall impact
- Results
  - L-1 cache usually smaller than a single cache
  - L-1 block size smaller than L-2 block size