

Cache Memory

CPE186 Computer Hardware Design

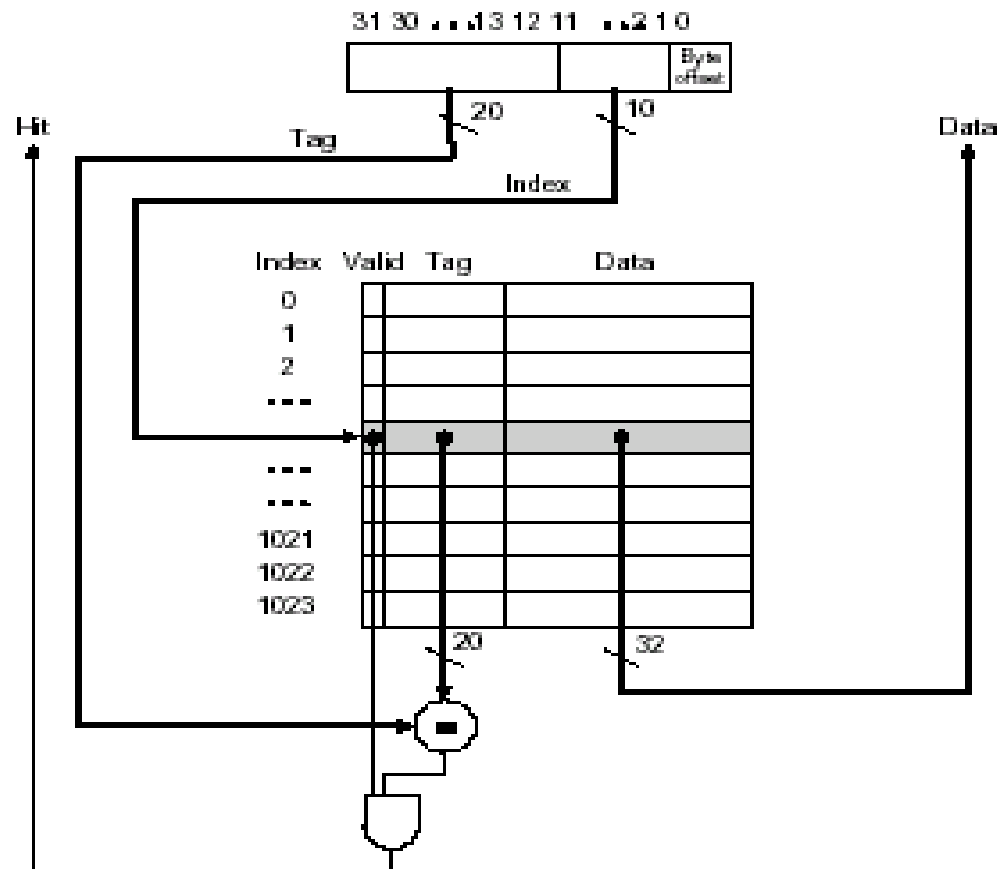
Revisiting Memory Hierarchy

- Facts
 - Big is slow
 - Fast is small
- Increase performance by having “hierarchy” of memory subsystems
- “Temporal Locality” and “Spatial Locality” are big ideas

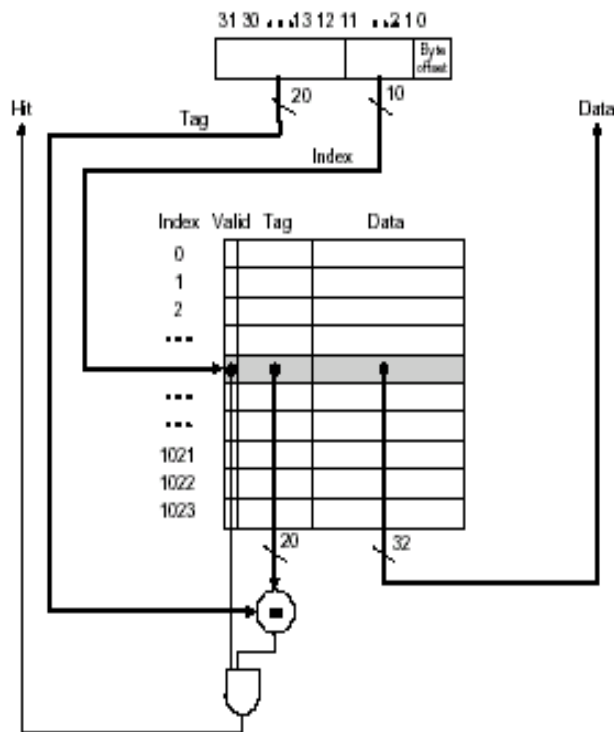
Revisiting Memory Hierarchy

- Terms
 - Cache Miss
 - Cache Hit
 - Hit Rate
 - Miss Rate
 - Index, Offset and Tag

Direct Mapped Cache



Direct Mapped Cache [contd...]



- What is the size of cache ?

4K

- If I read

0000 0000 0000 0000 0000 0000 1000 0001

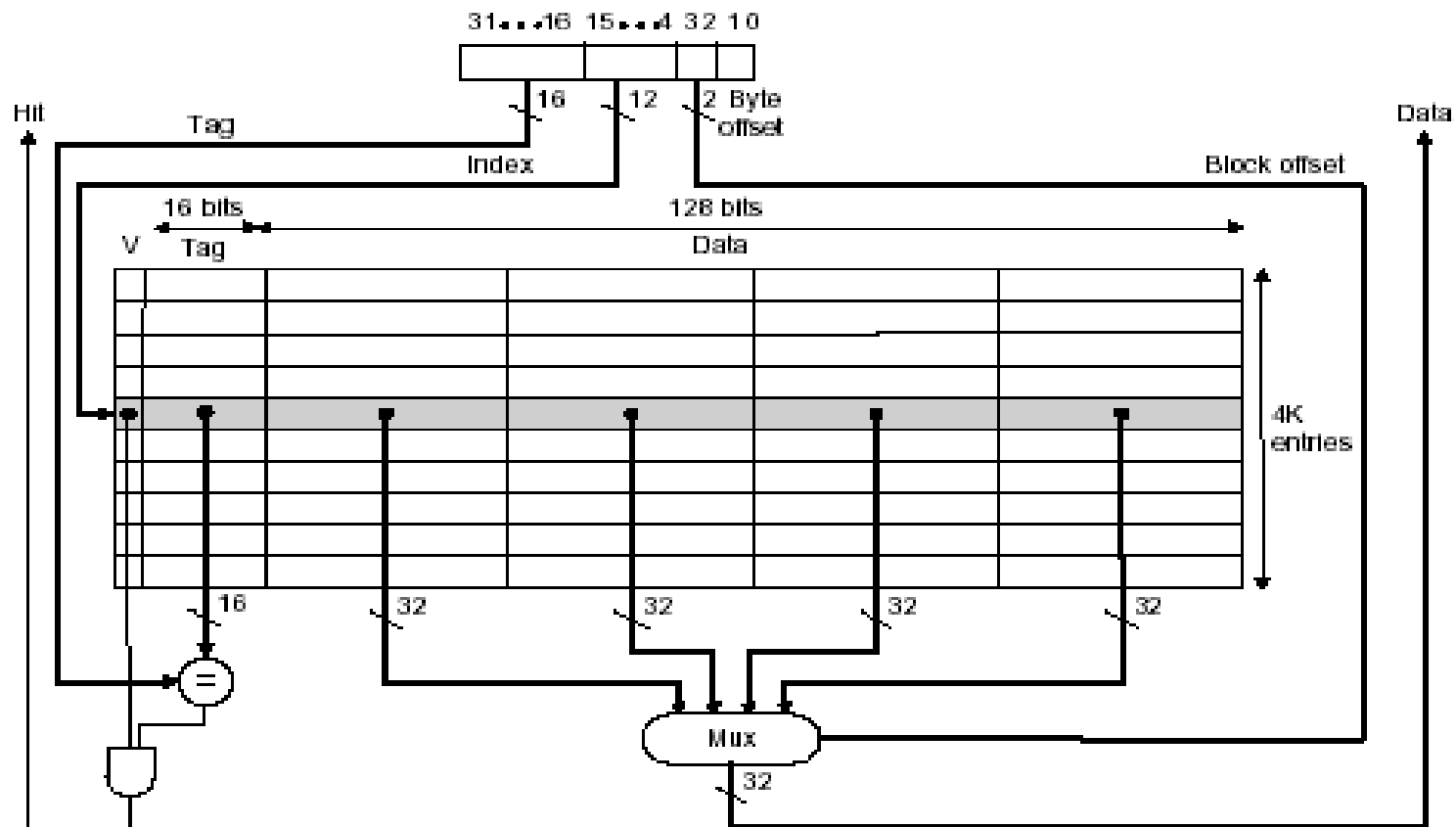
- What is the index number checked ?

32

- If the number was found, what are the inputs to comparator ?

Direct Mapped Cache [contd...]

Taking advantage of spatial locality, we read 4 bytes at a time

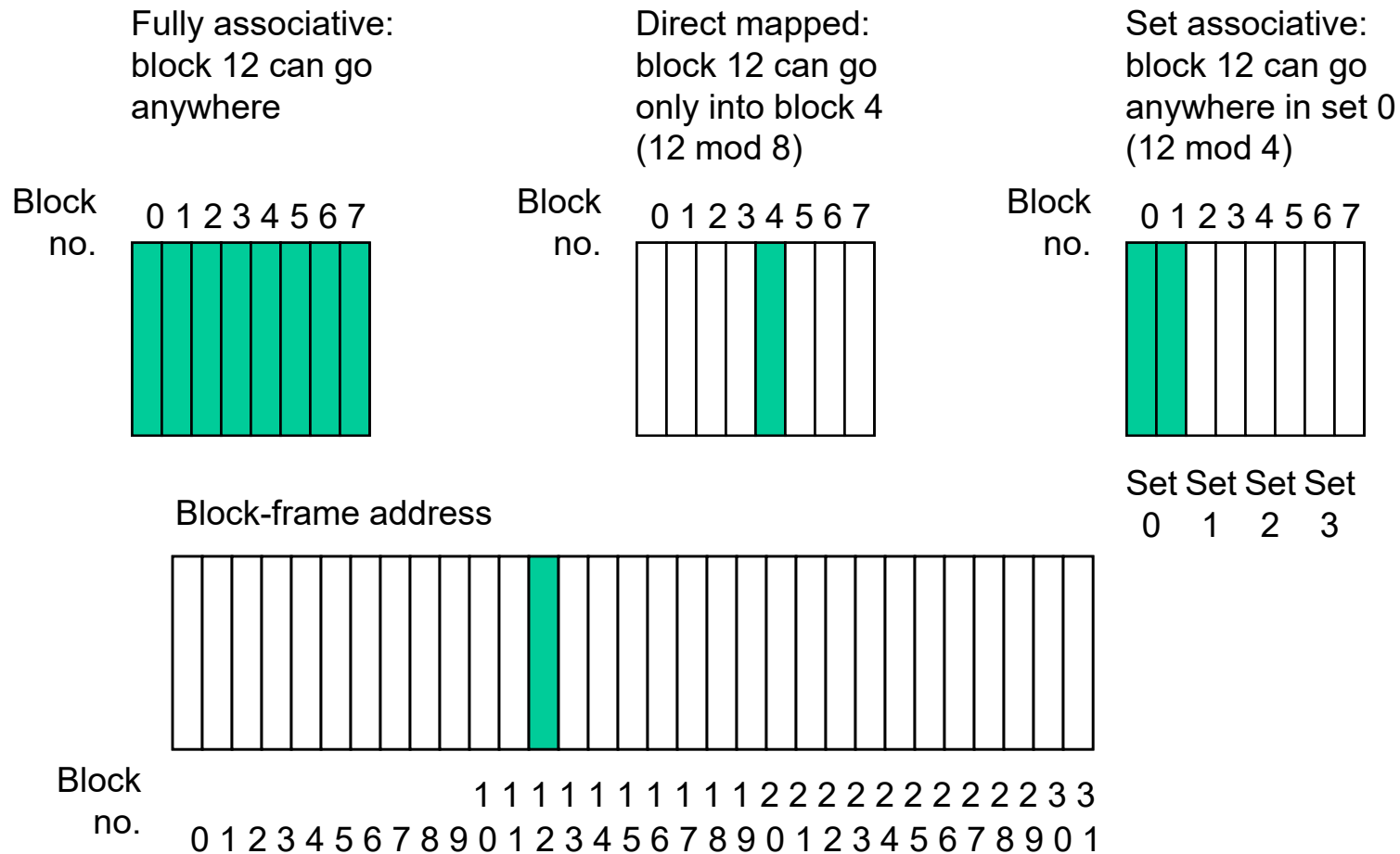


Direct Mapped Cache [contd...]

- Advantage
 - Simple
 - Fast
- Disadvantage
 - Mapping is fixed !!!

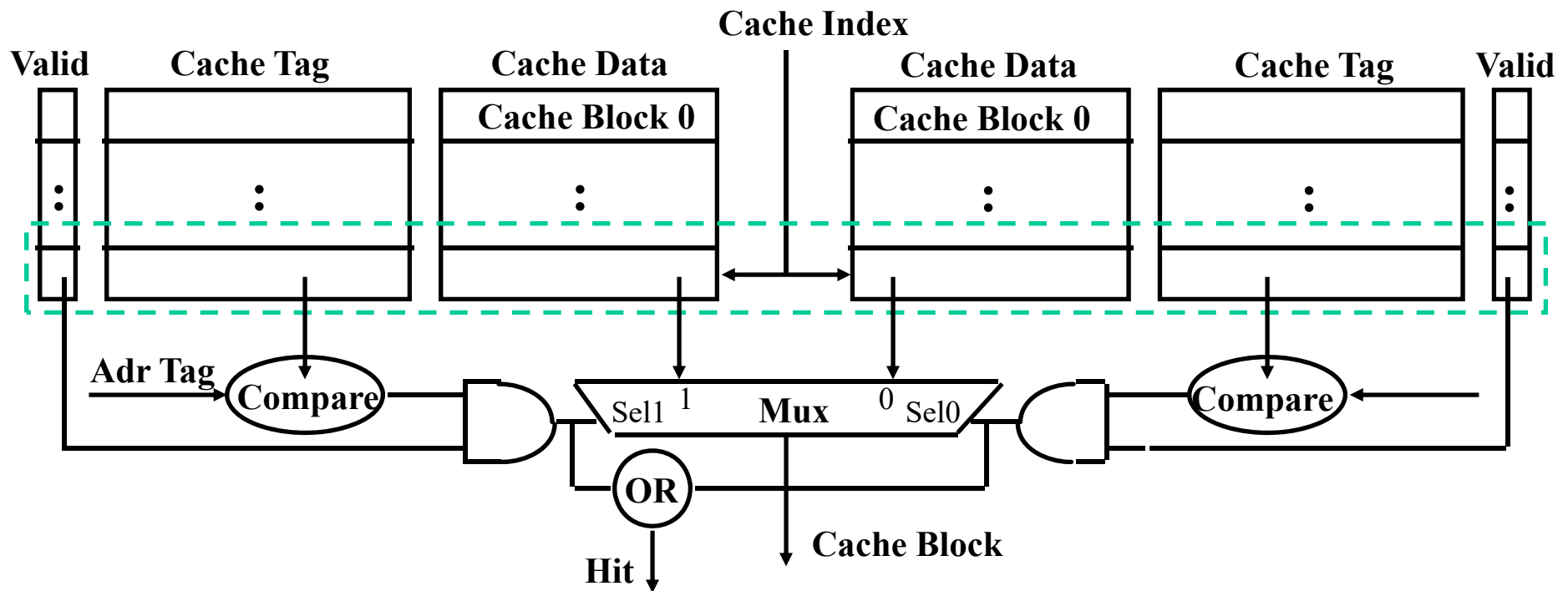
Associative Caches

- Block 12 placed in 8 block cache:
 - Fully associative, direct mapped, 2-way set associative
 - S.A. Mapping = Block Number Modulo Number Sets

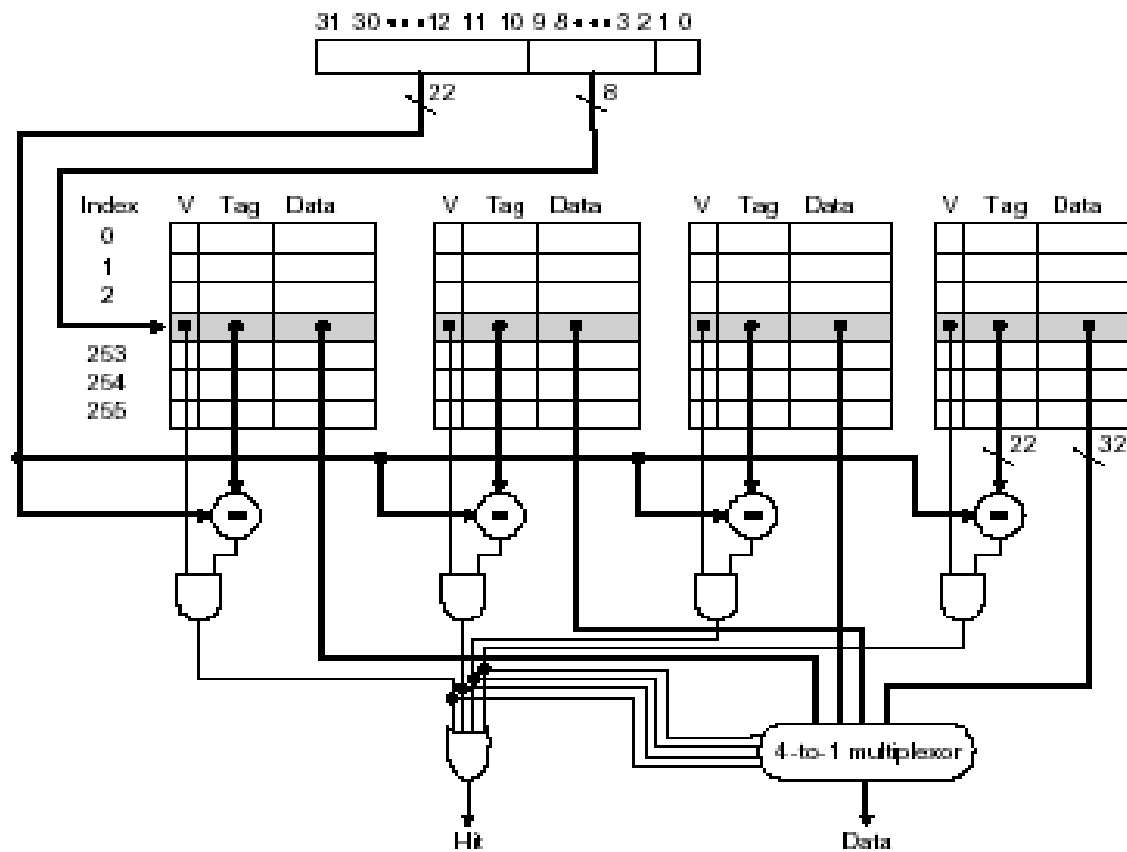


Set Associative Cache

- **N-way set associative**: N entries for each Cache Index
 - N direct mapped caches operates in parallel
- Example: Two-way set associative cache
 - Cache Index selects a “set” from the cache
 - The two tags in the set are compared to the input in parallel
 - Data is selected based on the tag result



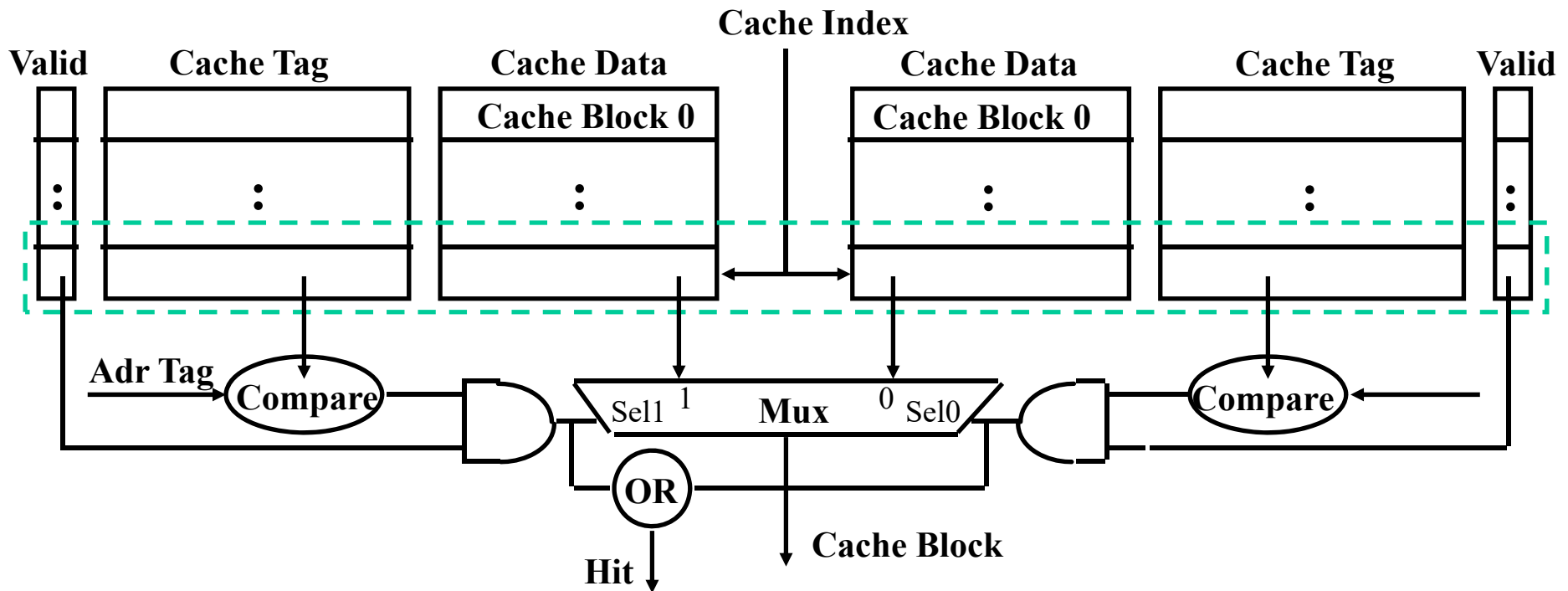
Example: 4-way set associative Cache



What is the cache size in this case ?

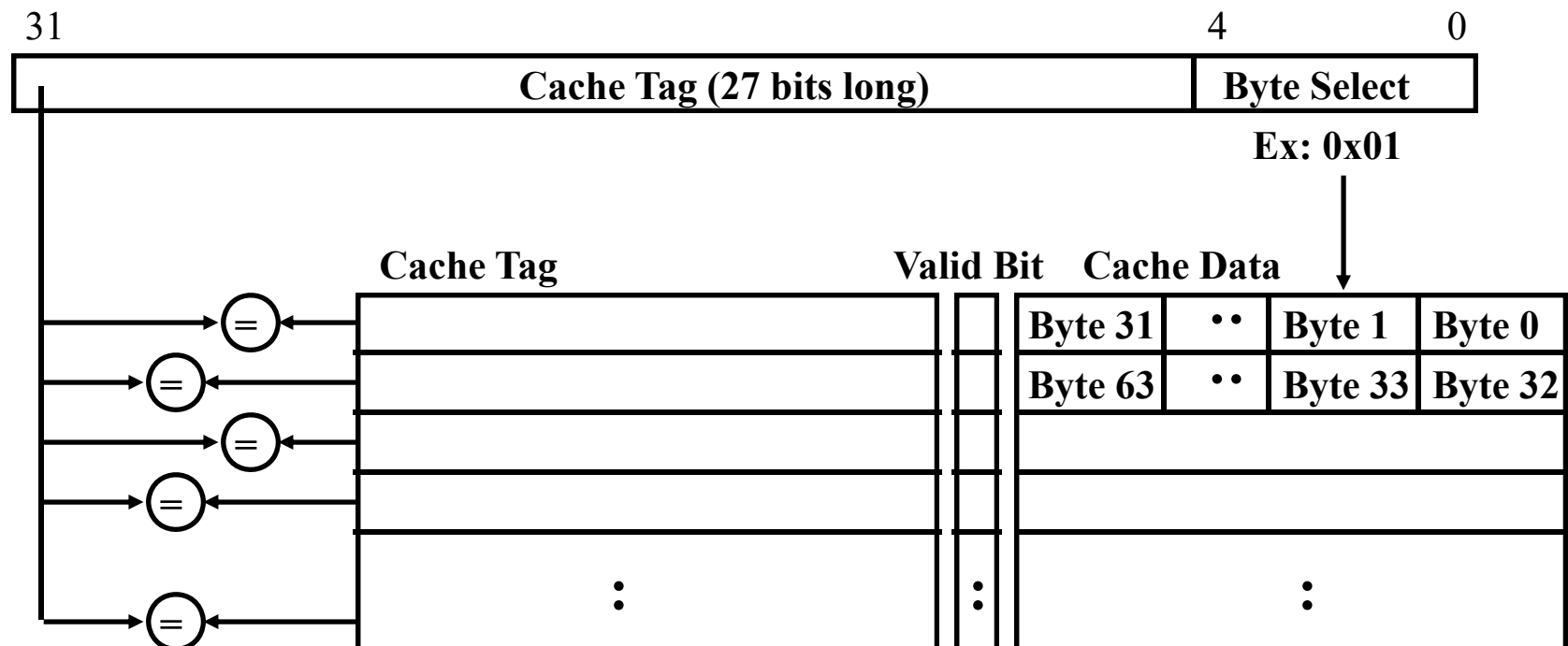
Disadvantages of Set Associative Cache

- N-way Set Associative Cache versus Direct Mapped Cache:
 - N comparators vs. 1
 - Extra MUX delay for the data
 - Data comes **AFTER** Hit/Miss decision and set selection
- In a direct mapped cache, Cache Block is available **BEFORE** Hit/Miss:



Fully Associative Cache

- Fully Associative Cache
 - Forget about the Cache Index
 - Compare the Cache Tags of all cache entries in parallel
 - Example: Block Size = 32 B blocks, we need N 27-bit comparators
- By definition: Conflict Miss = 0 for a fully associative cache

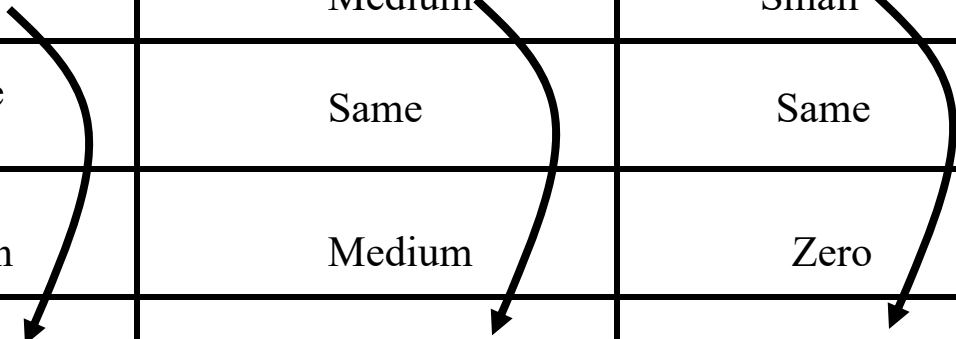


Cache Misses

- **Compulsory** (cold start or process migration, first reference): first access to a block
 - “Cold” fact of life: not a whole lot you can do about it
 - Note: If you are going to run “billions” of instruction, Compulsory Misses are insignificant
- **Capacity**:
 - Cache cannot contain all blocks access by the program
 - Solution: increase cache size
- **Conflict** (collision):
 - Multiple memory locations mapped to the same cache location
 - Solution 1: increase cache size
 - Solution 2: increase associativity
- **Coherence** (Invalidation): other process (e.g., I/O) updates memory

Design Options at Constant Cost

	Direct Mapped	N-way Set Associative	Fully Associative
Cache Size	Big	Medium	Small
Compulsory Miss	Same	Same	Same
Conflict Miss	High	Medium	Zero
Capacity Miss	Low	Medium	High
Coherence Miss	Same	Same	Same



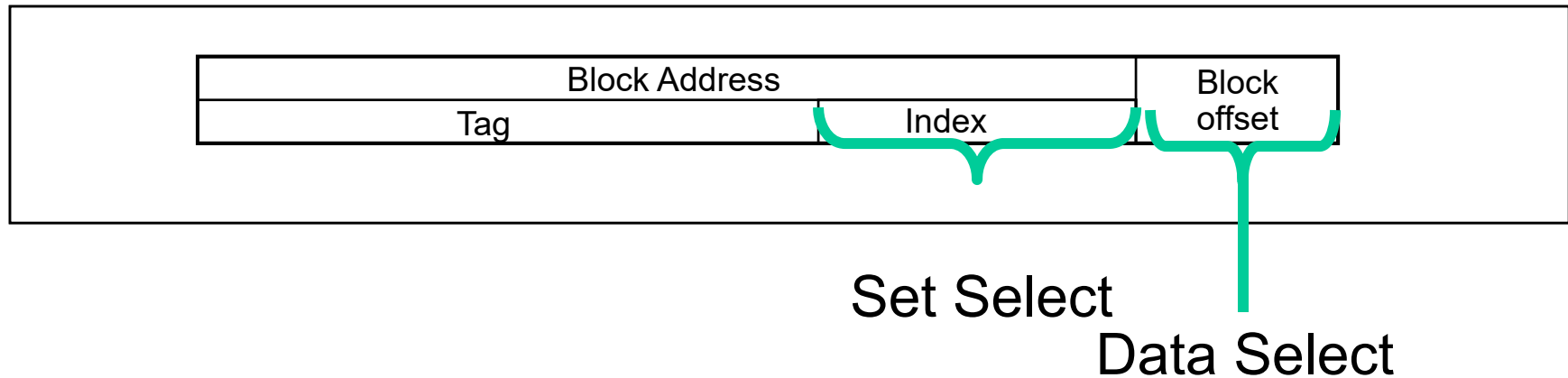
- Direct Mapped
- Set Associative
- Fully Associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative								
Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

[illegible]

How is a block found if it is in the upper level?



- Direct indexing (using index and block offset), tag compares, or combination
- Increasing associativity shrinks index, expands tag

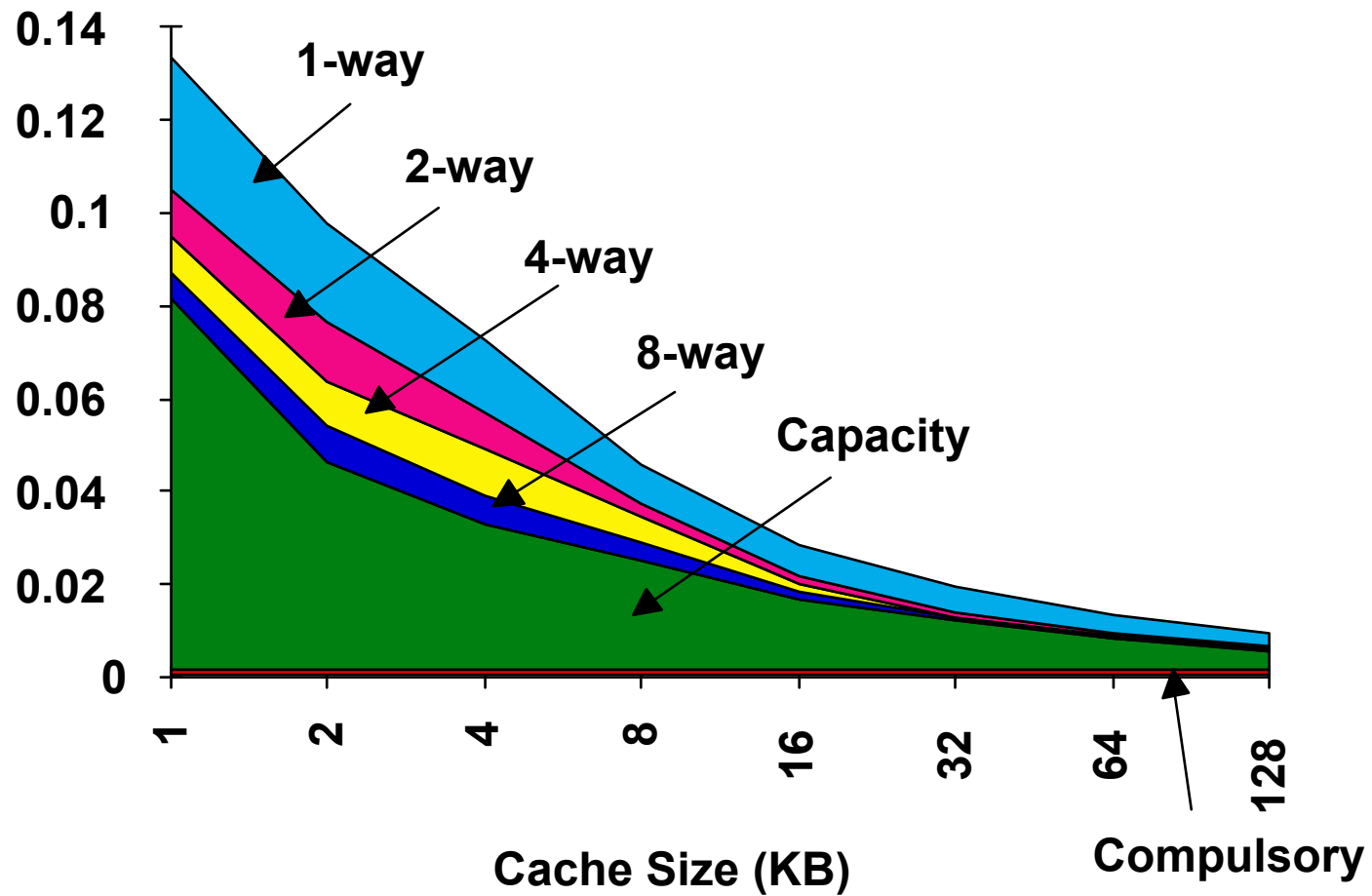
Which block should be replaced on a miss?

- Easy for Direct Mapped
- Set Associative or Fully Associative:
 - Random
 - LRU (Least Recently Used)

What happens on a write?

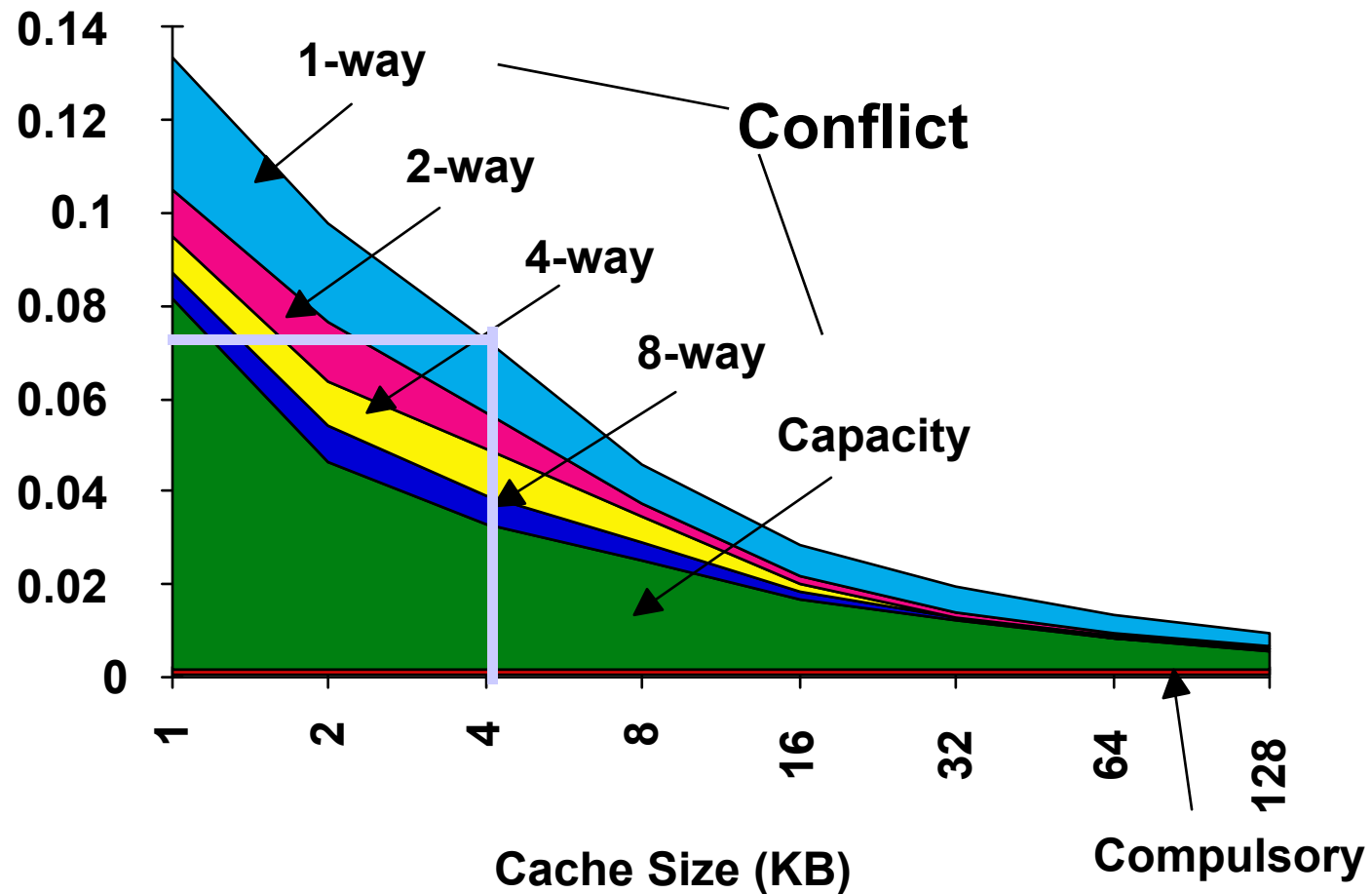
- Write through—The information is written to both the block in the cache and to the block in the lower-level memory.
- Write back—The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.
 - is block clean or dirty?
- Pros and Cons of each?
 - WT: read misses cannot result in writes
 - WB: no writes of repeated writes
- WT always combined with write buffers so that don't wait for lower level memory

Reduce Miss Rate

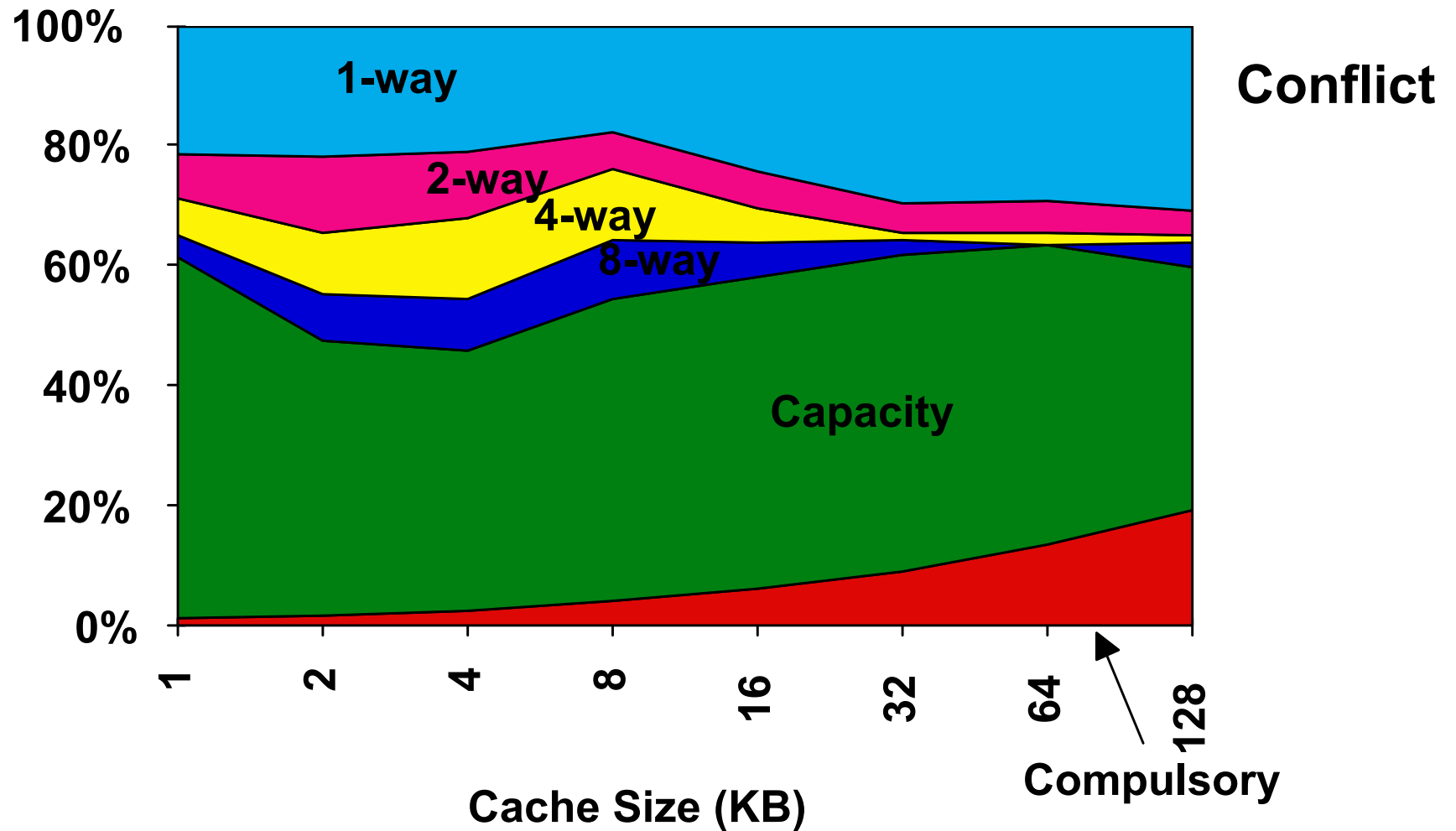


Reduce Miss Rate [contd...]

**miss rate 1-way associative cache size X
= miss rate 2-way associative cache size X/2**



3Cs Comparison



Reducing Miss Penalty

- Faster RAM memories
 - Driven by technology and cost !!!

Reduce Hit Time

- Lower Associativity
 - Add L1 and L2 caches

The possible R/W cases

- Read Hit
 - Good for CPU !!!
- Read Miss
 - Stall CPU, fetch, Resume
- Write Hit
 - Write Through
 - Write Back
- Write Miss
 - Write entire block into memory, Read into cache