CSE 112: Computer Organization

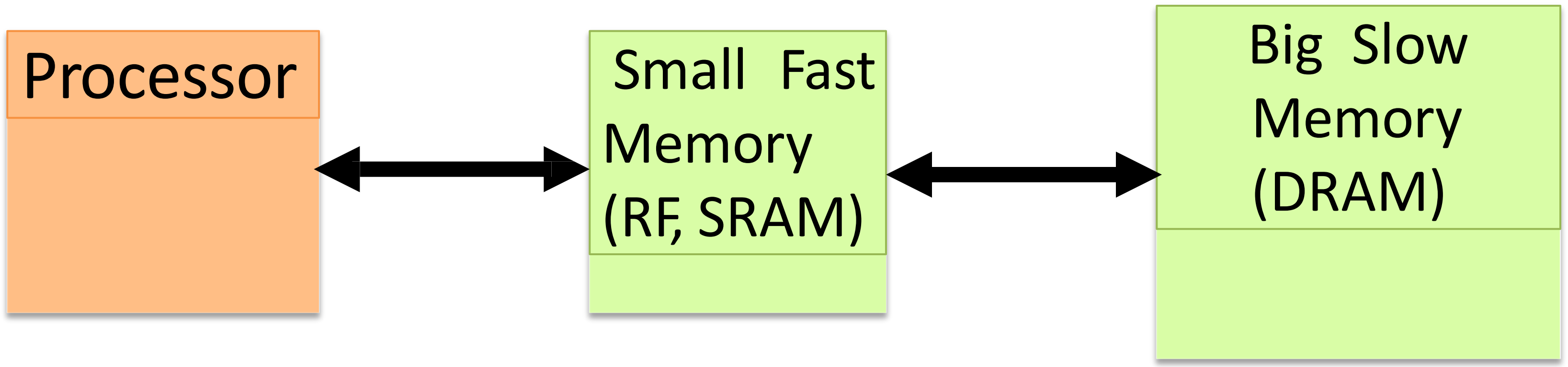
Instructor: Sujay Deb

Lecture 23

INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY DELHI

AMS Lab @IIITD

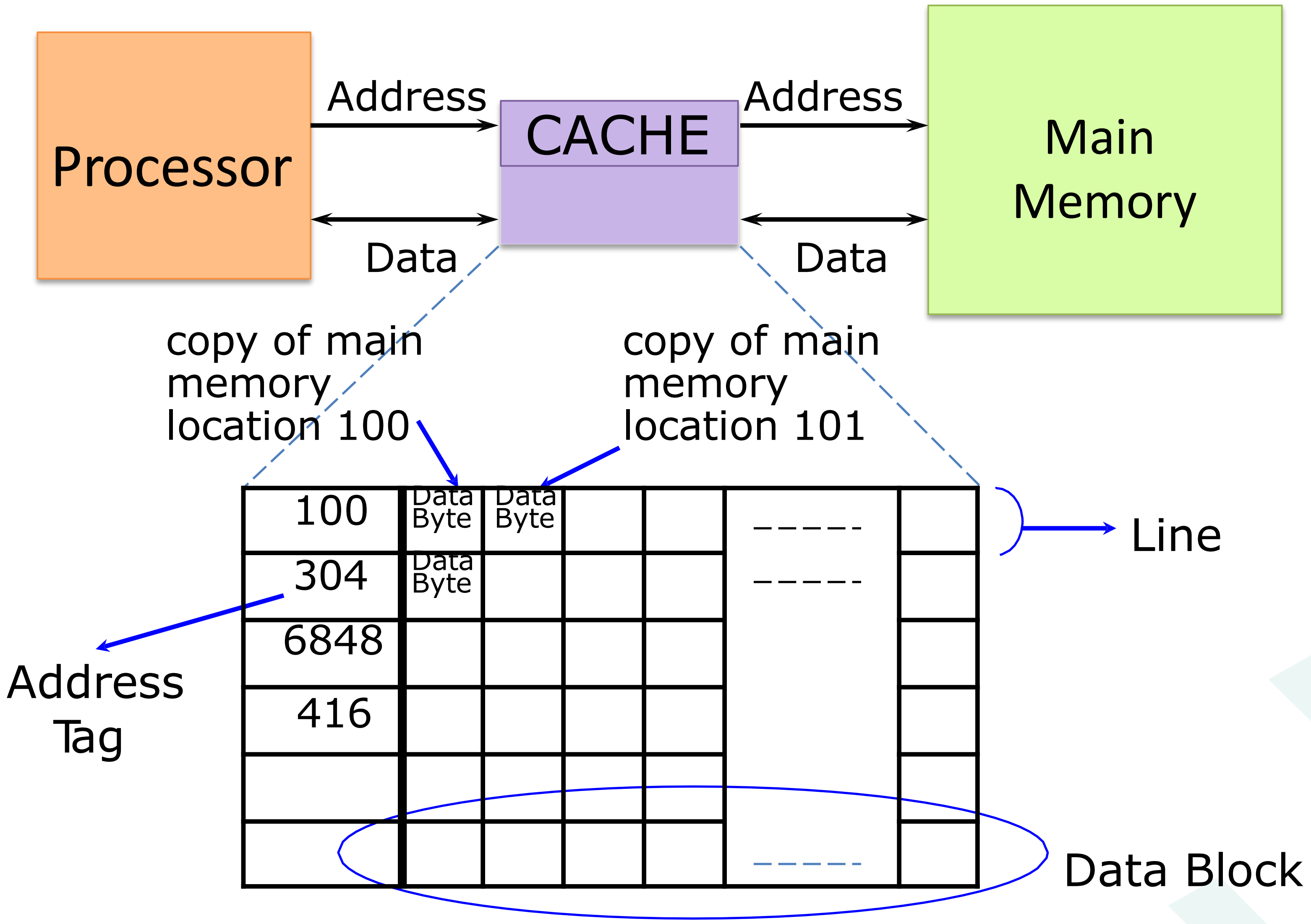
Caches Exploit Both Types of Locality



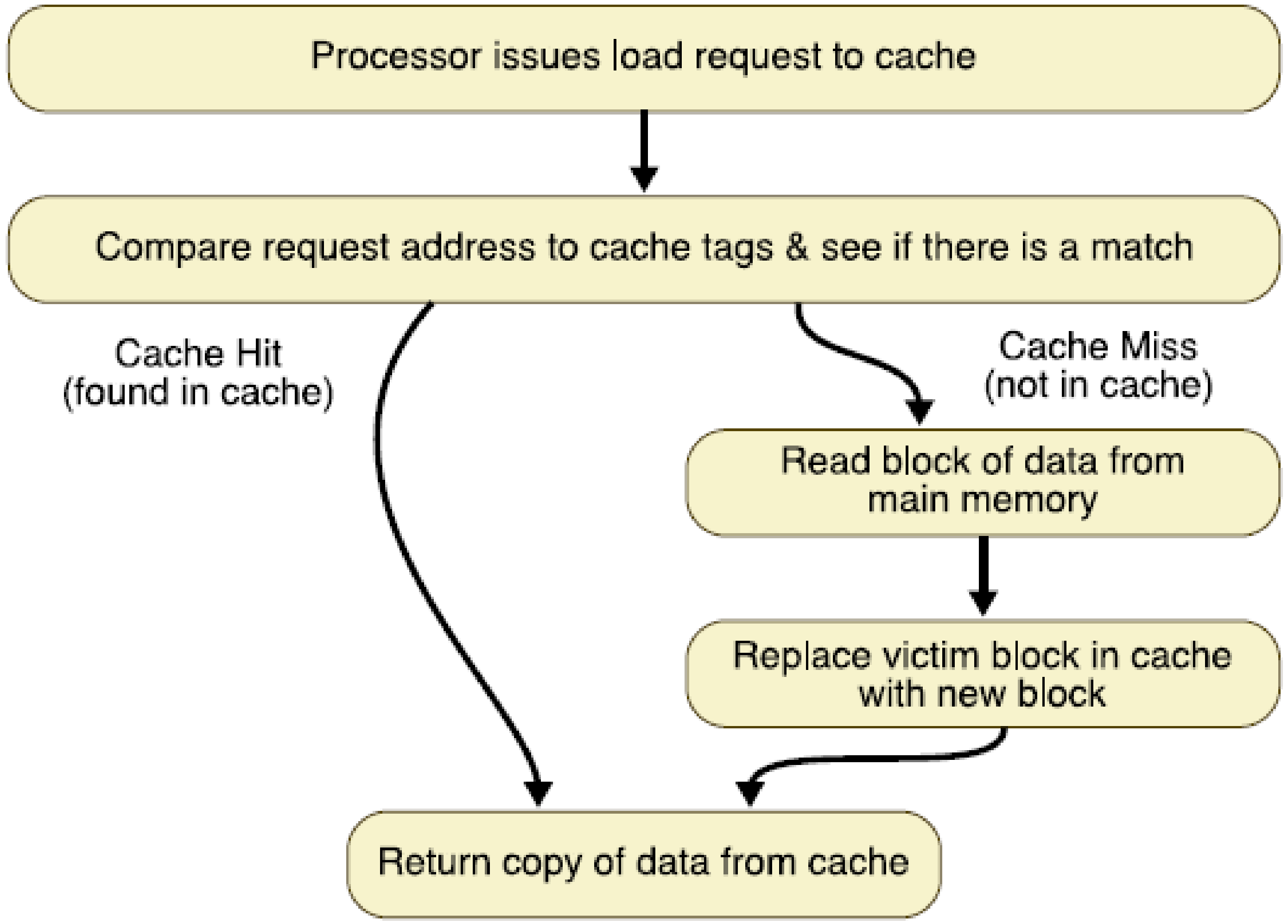
- Exploit temporal locality by remembering the contents of recently accessed locations
- Exploit spatial locality by fetching blocks of data around recently accessed locations

Classifying Caches

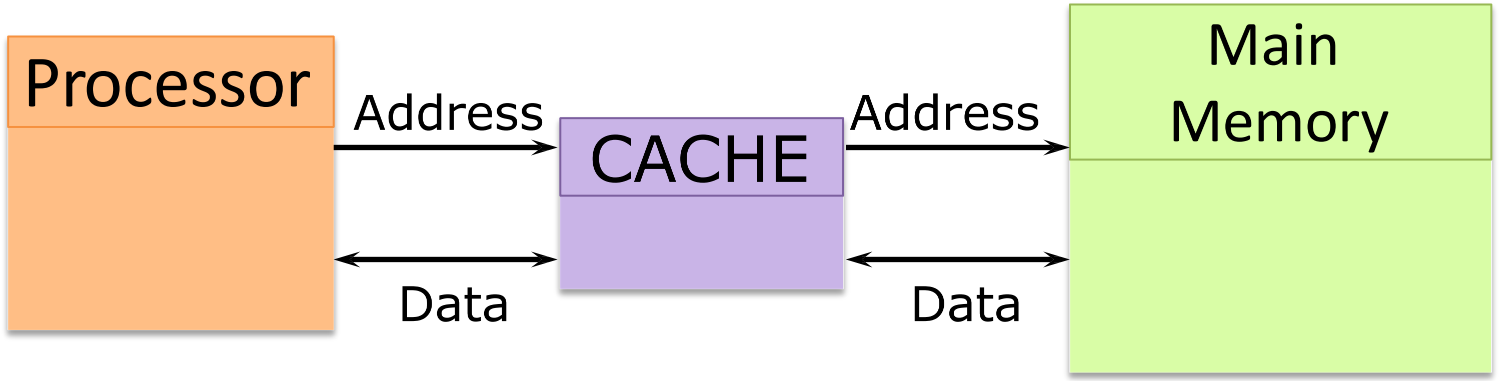
Inside a Cache



Basic Cache Algorithm for a Load

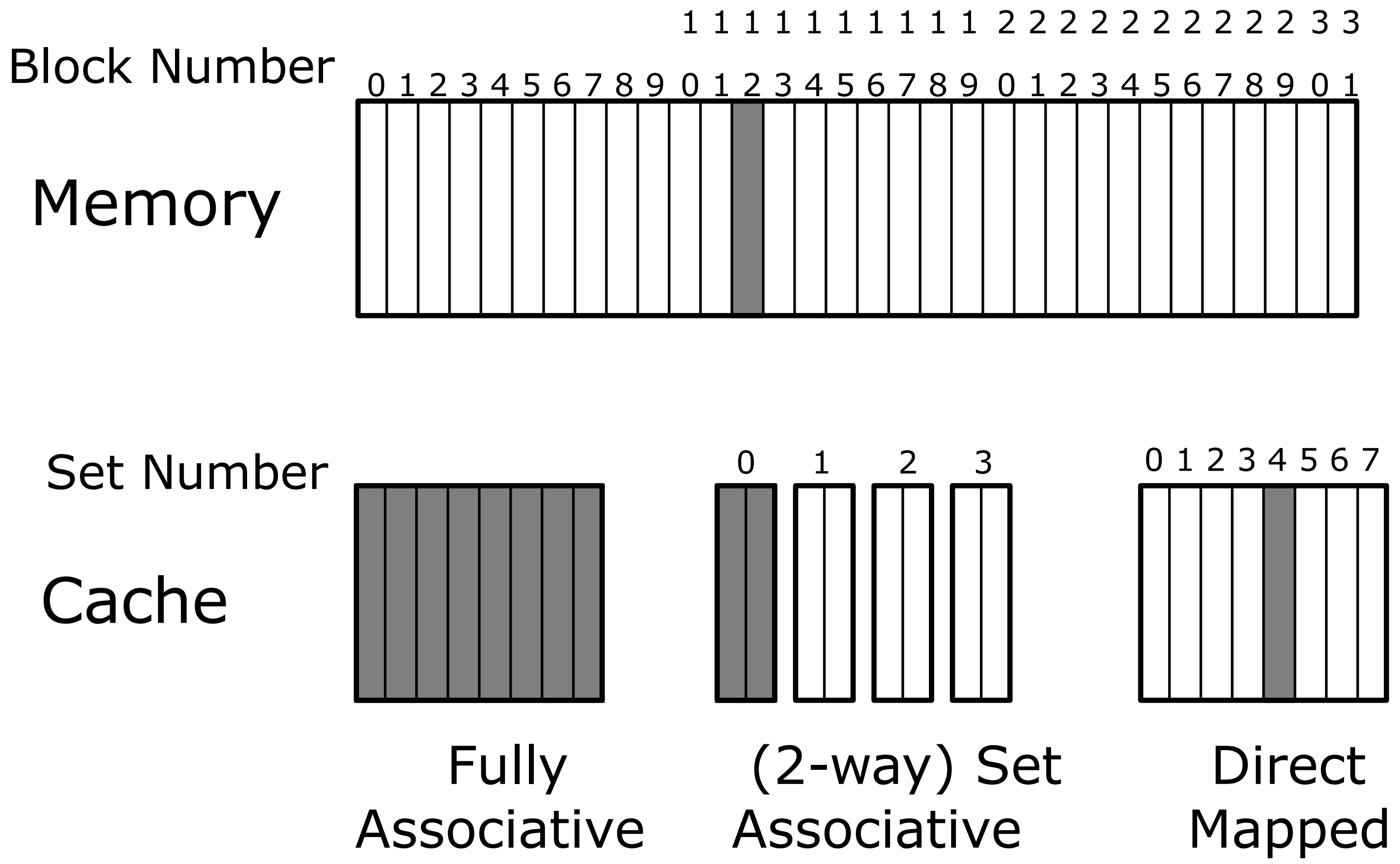

Processor issues load request to cache
Compare request address to cache tags & see if there is a match
Cache Hit (found in cache)
Cache Miss (not in cache)
Read block of data from main memory
Replace victim block in cache with new block
Return copy of data from cache

Classifying Caches



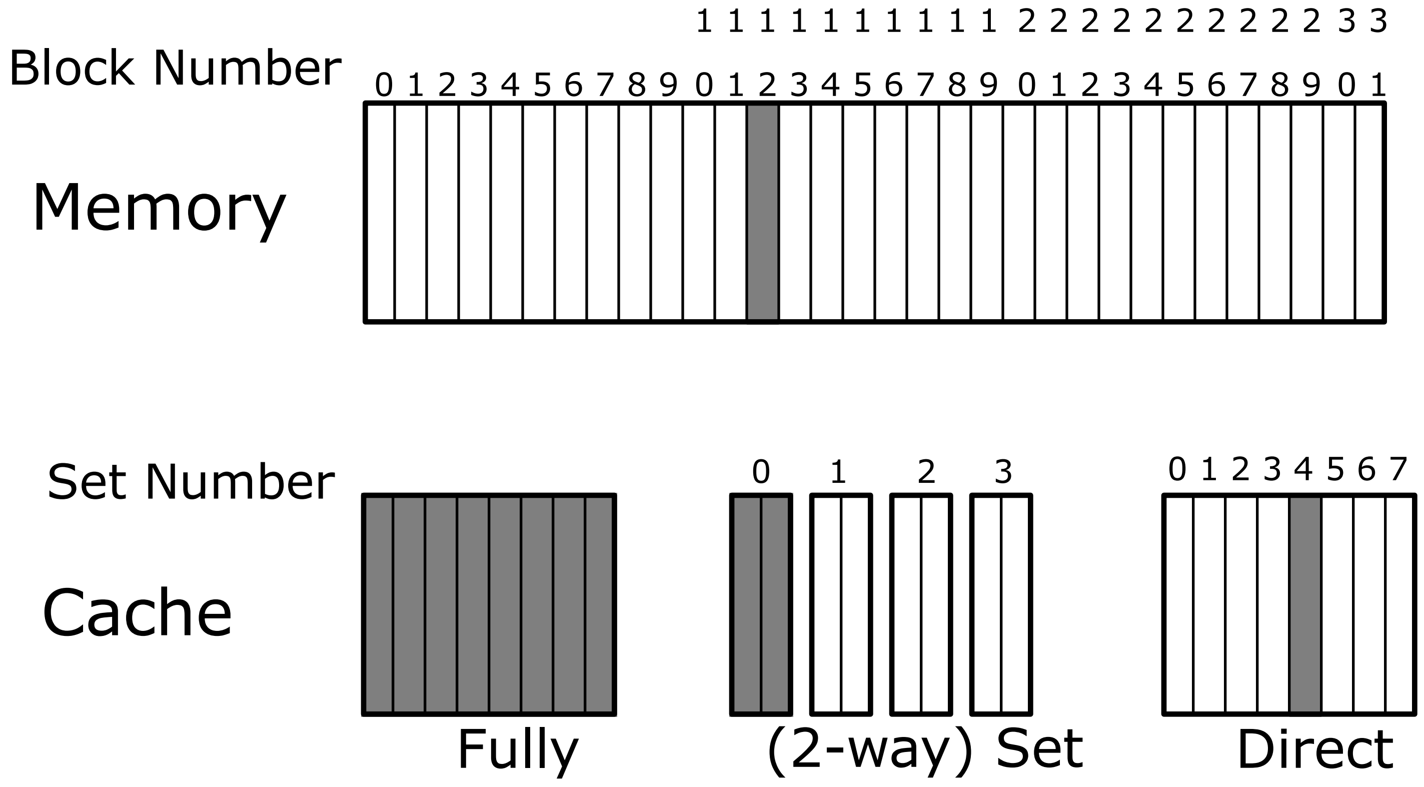
- **Block Placement**: Where can a block be placed in the cache?
- **Block Replacement**: Which block should be replaced on a miss?
- **Block Identification**: How a block is found if it is in the cache?
- **Write Strategy**: What happens on a write?

Block Placement: Where to Place Block in Cache?



block 12
can be placed

Block Placement: Where to Place Block in Cache?



Fully | (2-way) Set | Direct

Associative | Associative | Mapped

block 12 can be placed

anywhere

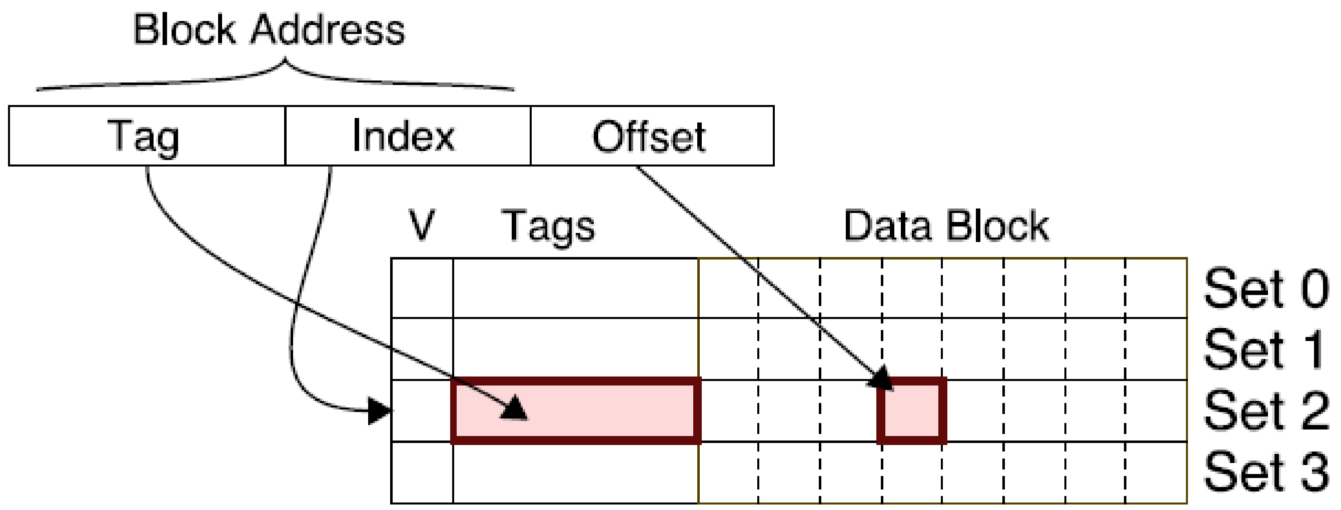
anywhere in set 0 *(12 mod 4)*

only into block 4 *(12 mod 8)*

Block Replacement: Which block to replace?

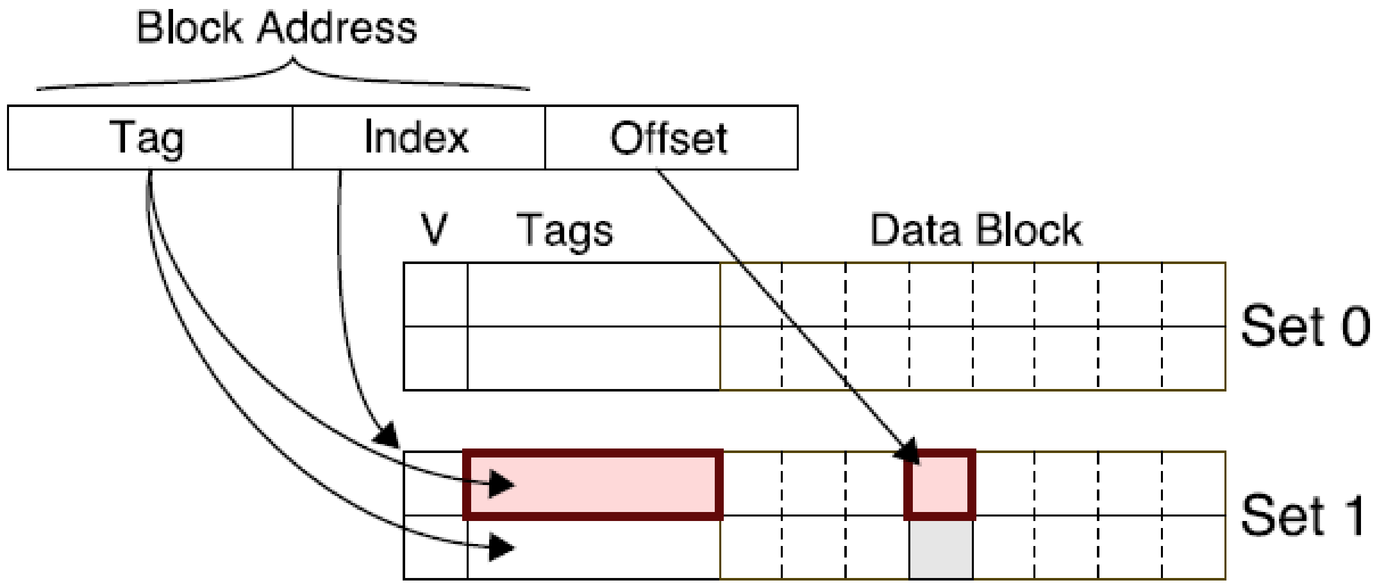
- No choice in a direct mapped cache
- In an associative cache, which block from set should be evicted when the set becomes full?
- **Random**
- **Least Recently Used (LRU)**
  - LRU cache state must be updated on every access
  - True implementation only feasible for small sets (2-way)
  - Pseudo-LRU binary tree often used for 4-8 way
- **First In, First Out (FIFO) aka Round-Robin**
  - Used in highly associative caches
- **Not Most Recently Used (NMRU)**
  - FIFO with exception for most recently used block(s)

Block Identification: How to find block in cache?



- Cache uses index and offset to find potential match, then checks tag
- Tag check only includes higher order bits
- In this example (Direct-mapped, 8B block, 4 line cache)

Block Identification: How to find block in cache?



- Cache checks all potential blocks with parallel tag check
- In this example (2-way associative, 8B block, 4 line cache)

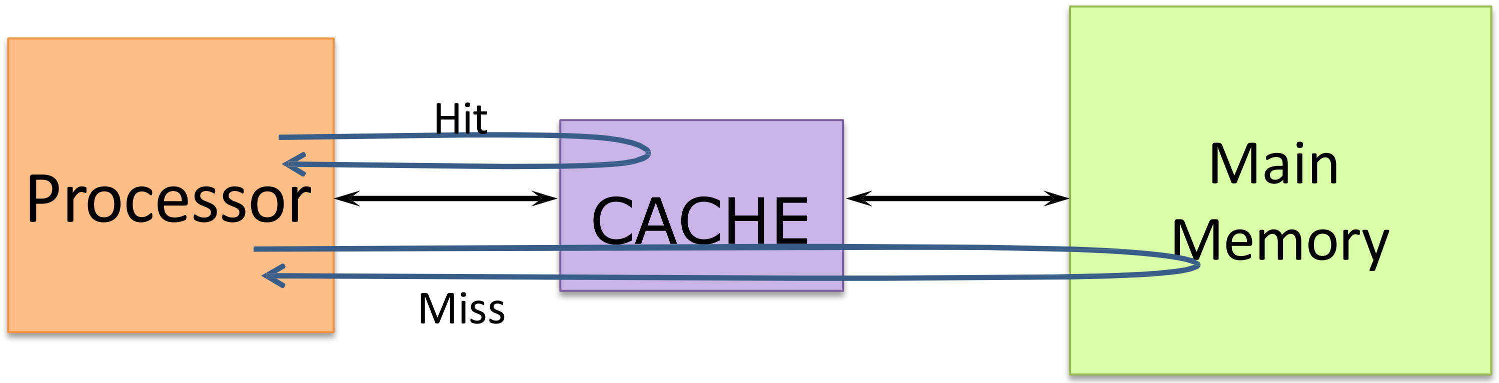
Write Strategy: How are writes handled?

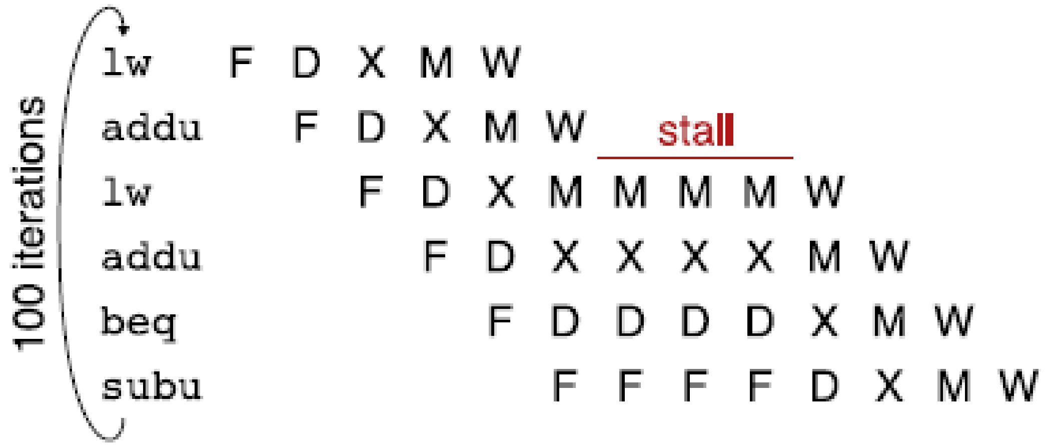
- Cache Hit
  - **Write Through –** write both cache and memory, generally higher traffic but simpler to design
  - **Write Back –** write cache only, memory is written when evicted, dirty bit per block avoids unnecessary write backs, more complicated
- Cache Miss
  - **No Write Allocate –** only write to main memory
  - **Write Allocate –** fetch block into cache, then write
- Common Combinations
- Write Through & No Write Allocate
- Write Back & Write Allocate

Class Interaction # 26

Cache Performance

Average Memory Access Time

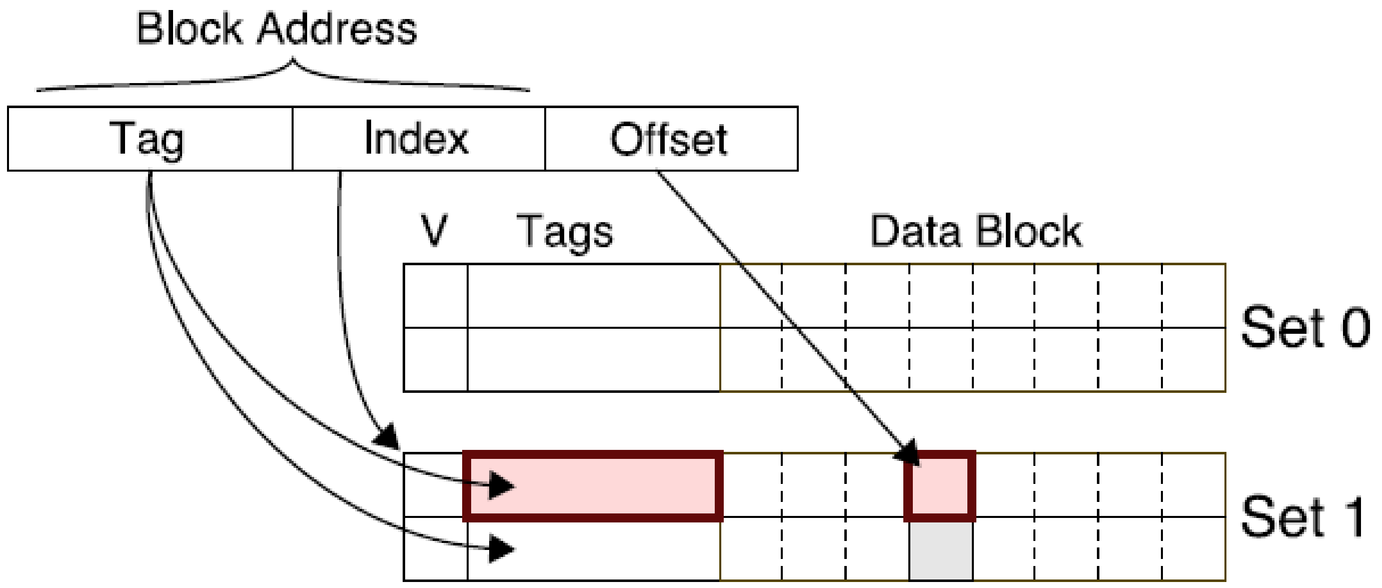



100 iterations

lw	F	D	X	M	W								
addu		F	D	X	M	W		stall					
lw			F	D	X	M	M	M	M	W			
addu				F	D	X	X	X	X	M	W		
beq					F	D	D	D	D	X	M	W	
subu						F	F	F	F	D	X	M	W

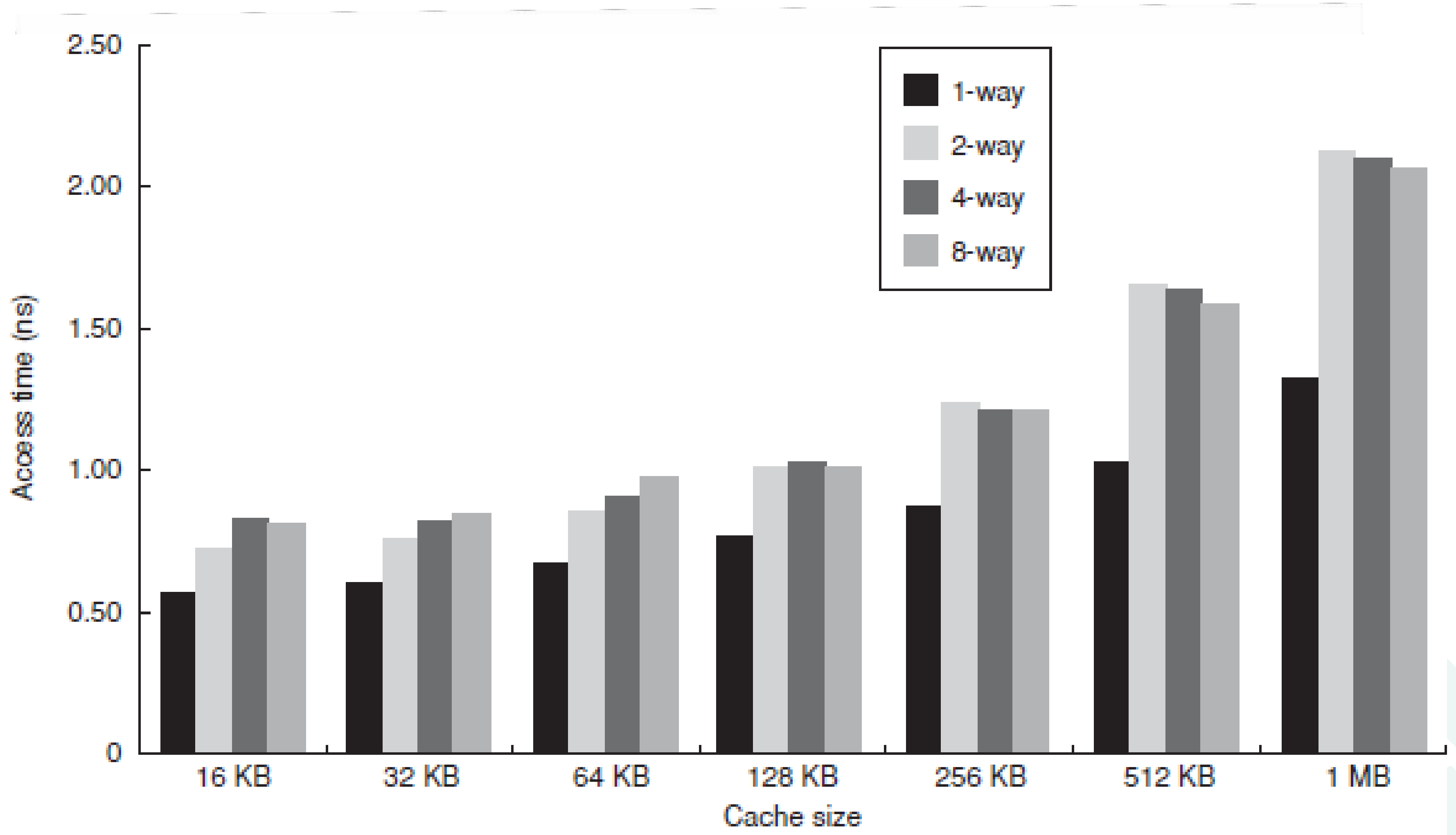
- Average Memory Access Time = Hit Time + (Miss Rate * Miss Penalty)

Categorizing Misses: The Three C's



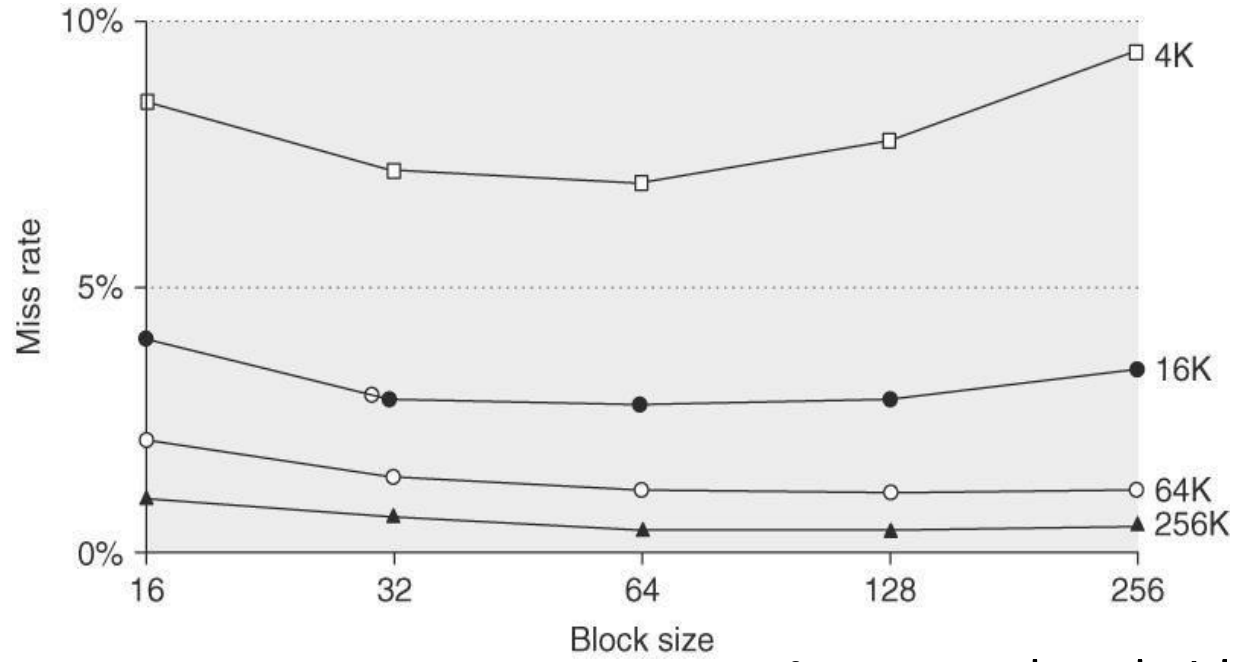
- **Compulsory** – first-reference to a block, occur even with infinite cache
- **Capacity** – cache is too small to hold all data needed by program, occur even under perfect replacement policy
- **Conflict** – misses that occur because of collisions due to less than full associativity

Reduce Hit Time: Small & Simple Caches



Plot from Hennessy and Patterson Ed. 4

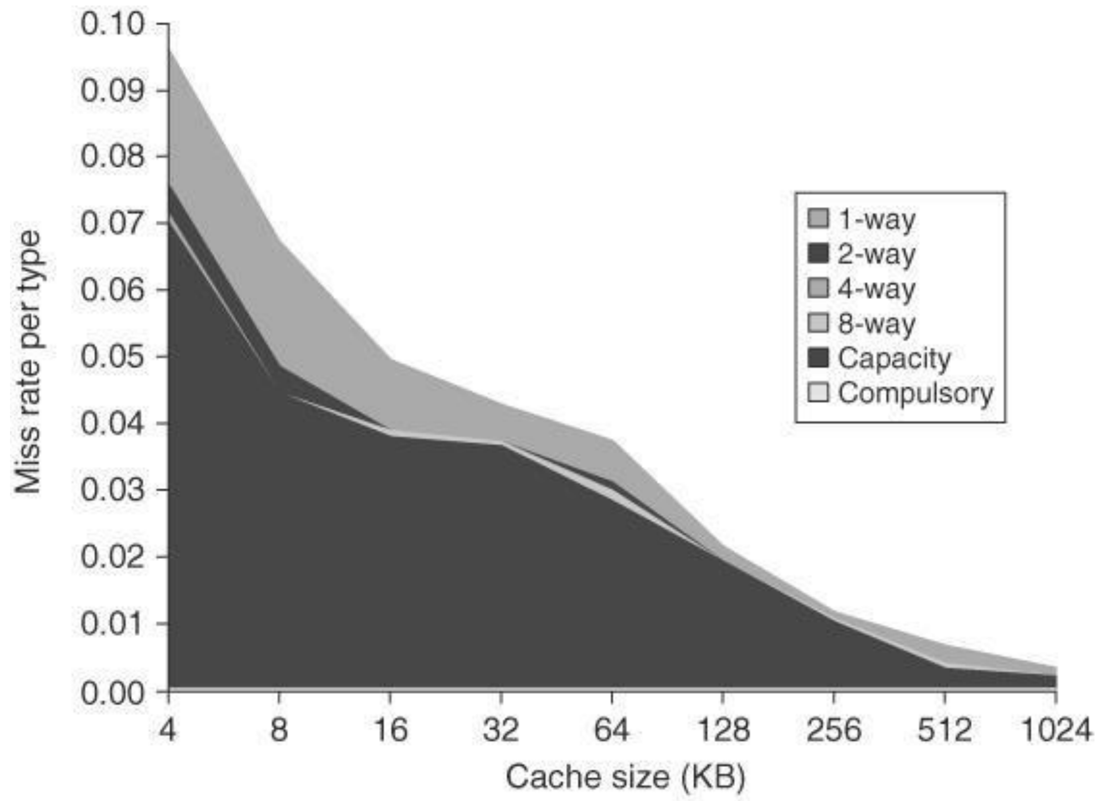
Reduce Miss Rate: Large Block Size



- Less tag overhead
- Exploit fast burst transfers from DRAM
- Exploit fast burst transfers over wide on-chip busses

- Can waste bandwidth if data is not used
- Fewer blocks -> more conflicts

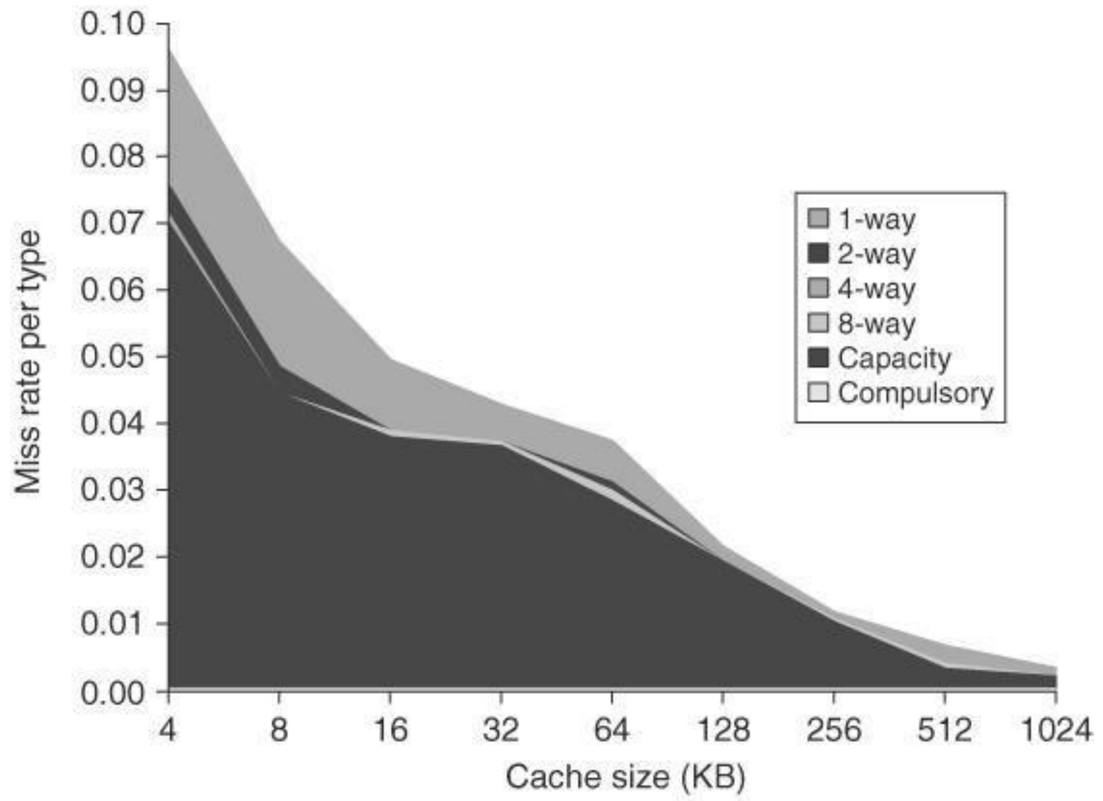
Reduce Miss Rate: Large Cache Size



Empirical Rule of Thumb:

If cache size is doubled, miss rate usually drops by about $\sqrt{2}$

Reduce Miss Rate: High Associativity

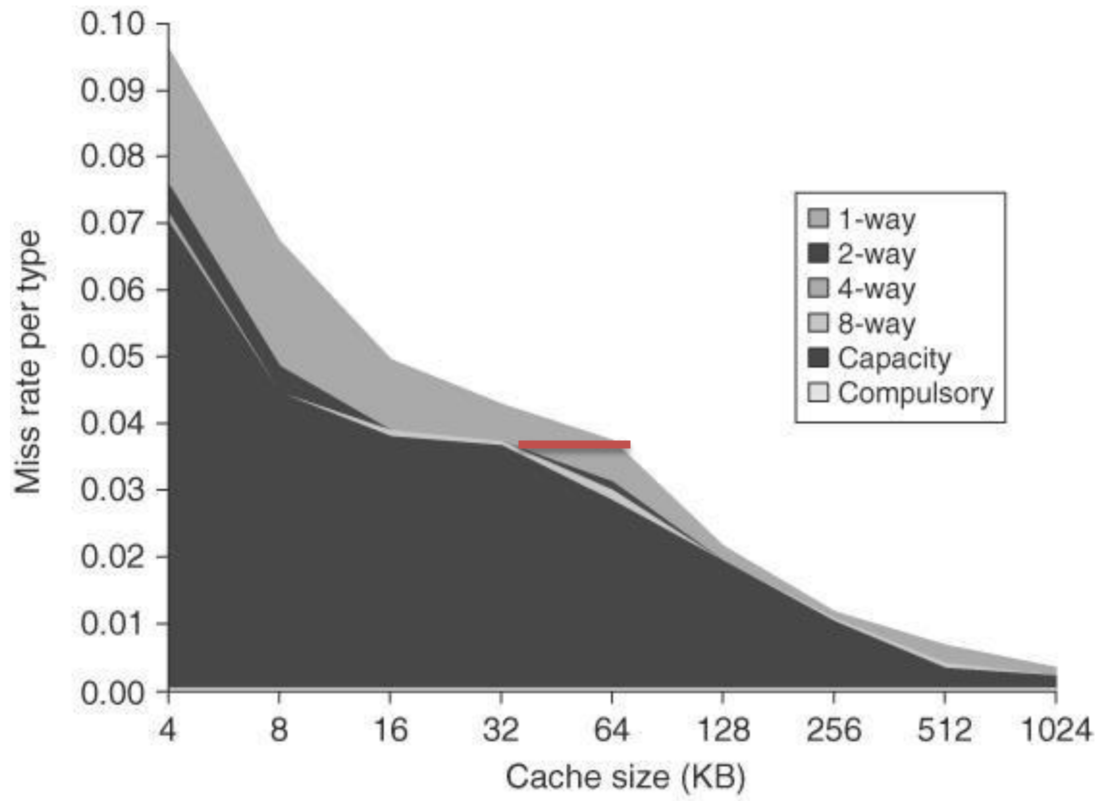


Empirical Rule of Thumb:

Direct-mapped cache of size N has about the same miss rate as a two-way set- associative cache of size N/2

Plot from Hennessy and Patterson Ed. 5 Image Copyright © 2011, Elsevier Inc. All rights Reserved.

Reduce Miss Rate: High Associativity



Empirical Rule of Thumb:

Direct-mapped cache of size N has about the same miss rate as a two-way set- associative cache of size N/2

- https://webriscv.dii.unisi.it/