

CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Caches Part 2

Instructors:

Krste Asanovic & Vladimir Stojanovic

<http://inst.eecs.berkeley.edu/~cs61c/>

You Are Here!

Software

Hardware

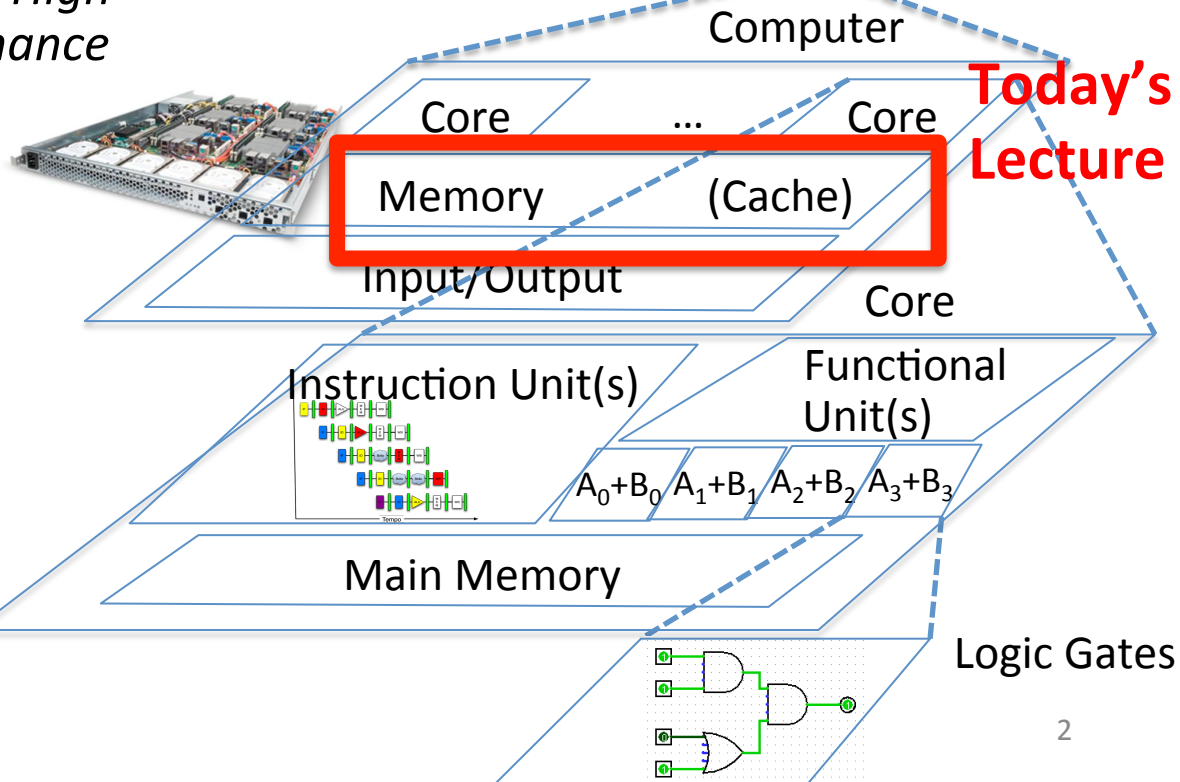
Warehouse
Scale
Computer



Smart
Phone



*Harness
Parallelism &
Achieve High
Performance*

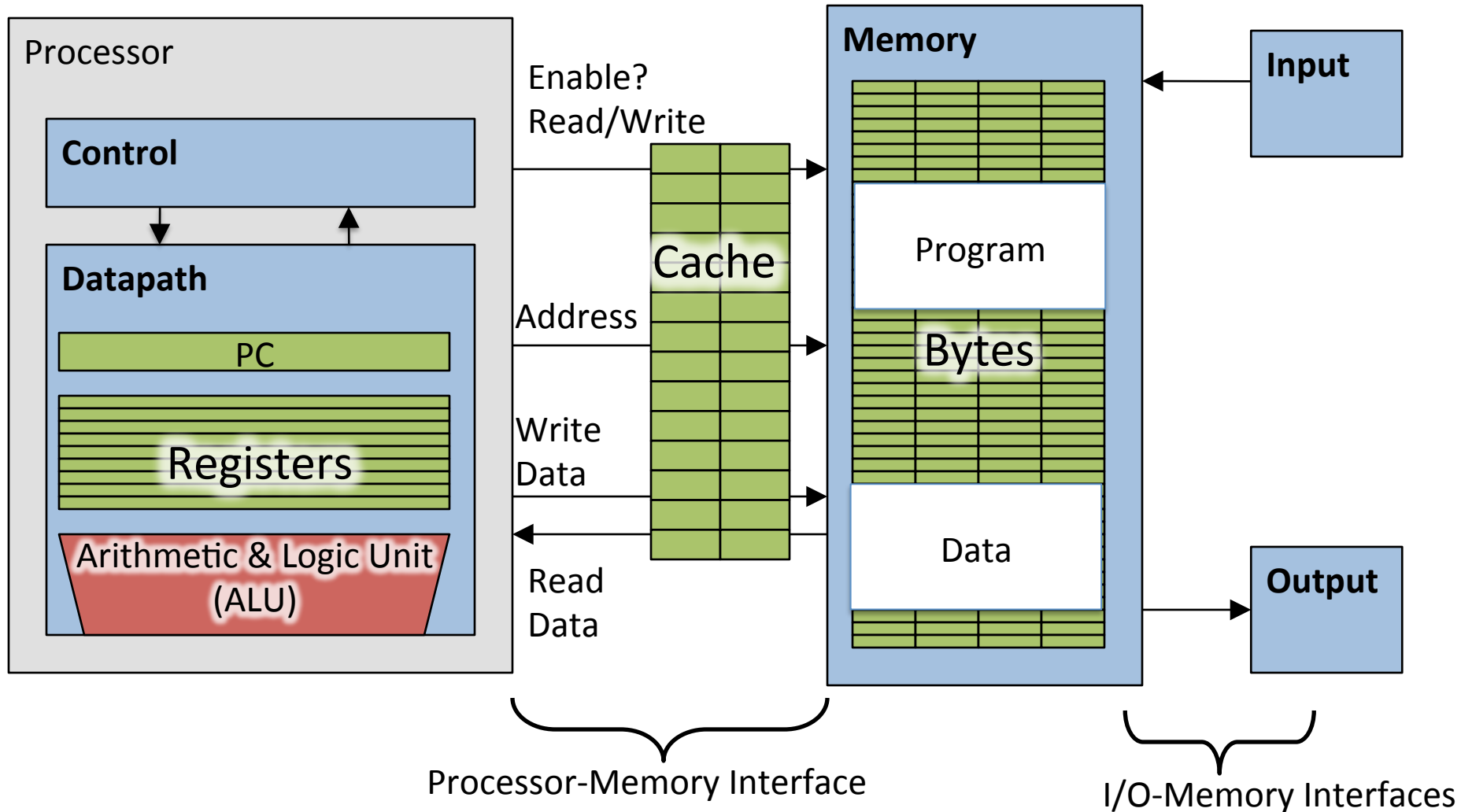


- Parallel Requests
Assigned to computer
e.g., Search “Katz”
- Parallel Threads
Assigned to core
e.g., Lookup, Ads
- Parallel Instructions
>1 instruction @ one time
e.g., 5 pipelined instructions
- Parallel Data
>1 data item @ one time
e.g., Add of 4 pairs of words
- Hardware descriptions
All gates @ one time
- Programming Languages

Caches Review

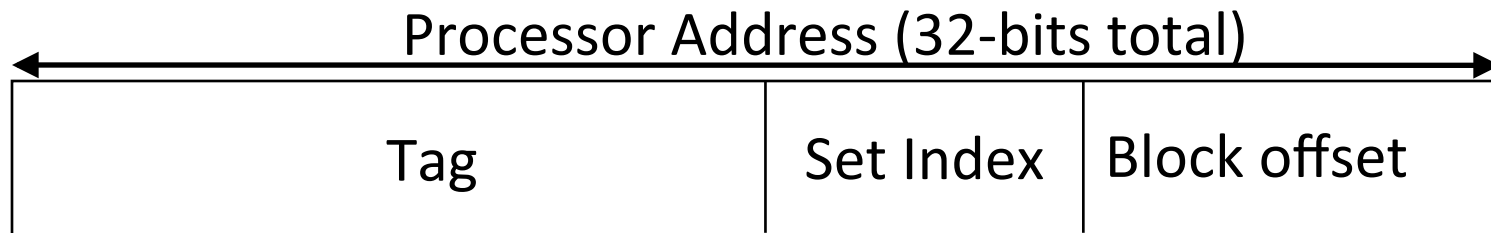
- Principle of Locality
 - Temporal Locality and Spatial Locality
- Hierarchy of Memories (speed/size/cost per bit) to Exploit Locality
- Cache – copy of data in lower level of memory hierarchy
- Direct Mapped to find block in cache using Tag field and Valid bit for Hit
- Cache design choice:
 - Write-Through vs. Write-Back

Review: Adding Cache to Computer



Processor Address Fields used by Cache Controller

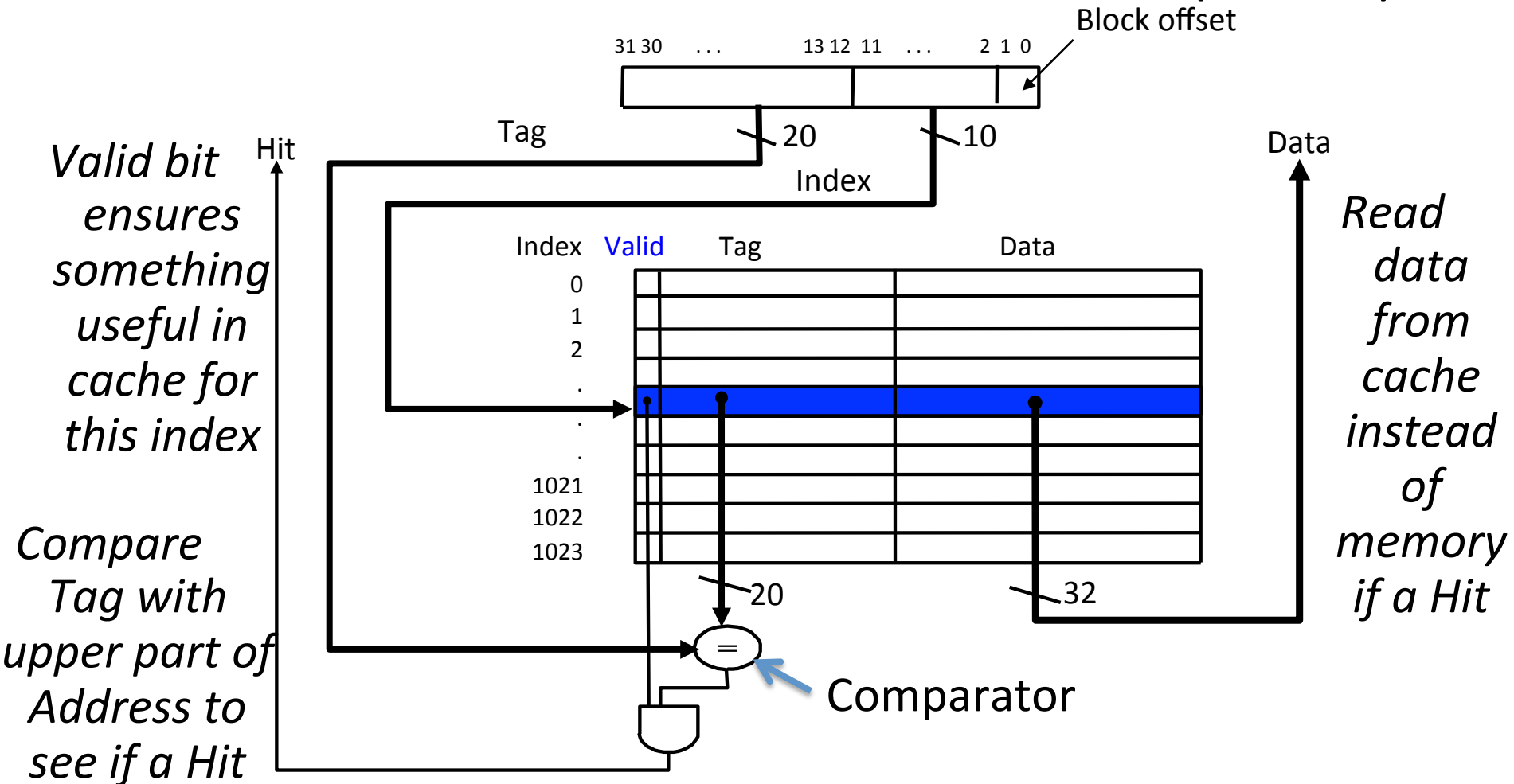
- **Block Offset**: Byte address within block
- **Set Index**: Selects which set
- **Tag**: Remaining portion of processor address



- Size of Index = \log_2 (number of sets)
- Size of Tag = Address size – Size of Index – \log_2 (number of bytes/block)

Review: Direct-Mapped Cache

- One word blocks, cache size = 1K words (or 4KB)



Cache Terms

- **Hit rate**: fraction of accesses that hit in the cache
- **Miss rate**: $1 - \text{Hit rate}$
- **Miss penalty**: time to replace a block from lower level in memory hierarchy to cache
- **Hit time**: time to access cache memory (including tag comparison)
- Abbreviation: “\$” = cache (A Berkeley innovation!)

Average Memory Access Time (AMAT)

- Average Memory Access Time (AMAT) is the average to access memory considering both hits and misses in the cache

$$\text{AMAT} = \text{Time for a hit} + \text{Miss rate} \times \text{Miss penalty}$$

Clickers/Peer instruction

$$\text{AMAT} = \text{Time for a hit} + \text{Miss rate} \times \text{Miss penalty}$$

Given a 200 psec clock, a miss penalty of 50 clock cycles, a miss rate of 0.02 misses per instruction and a cache hit time of 1 clock cycle, what is AMAT?

- ☐ A: ≤ 200 psec
- ☐ B: 400 psec
- ☐ C: 600 psec
- ☐ D: ≥ 800 psec

Example: Direct-Mapped \$ with 4 Single-Word Lines, Worst-Case Reference String

- Consider the main memory address reference string

Start with an empty cache - all blocks
initially marked as not valid 0 4 0 4 0 4 0 4

0

4

0

4

0

4

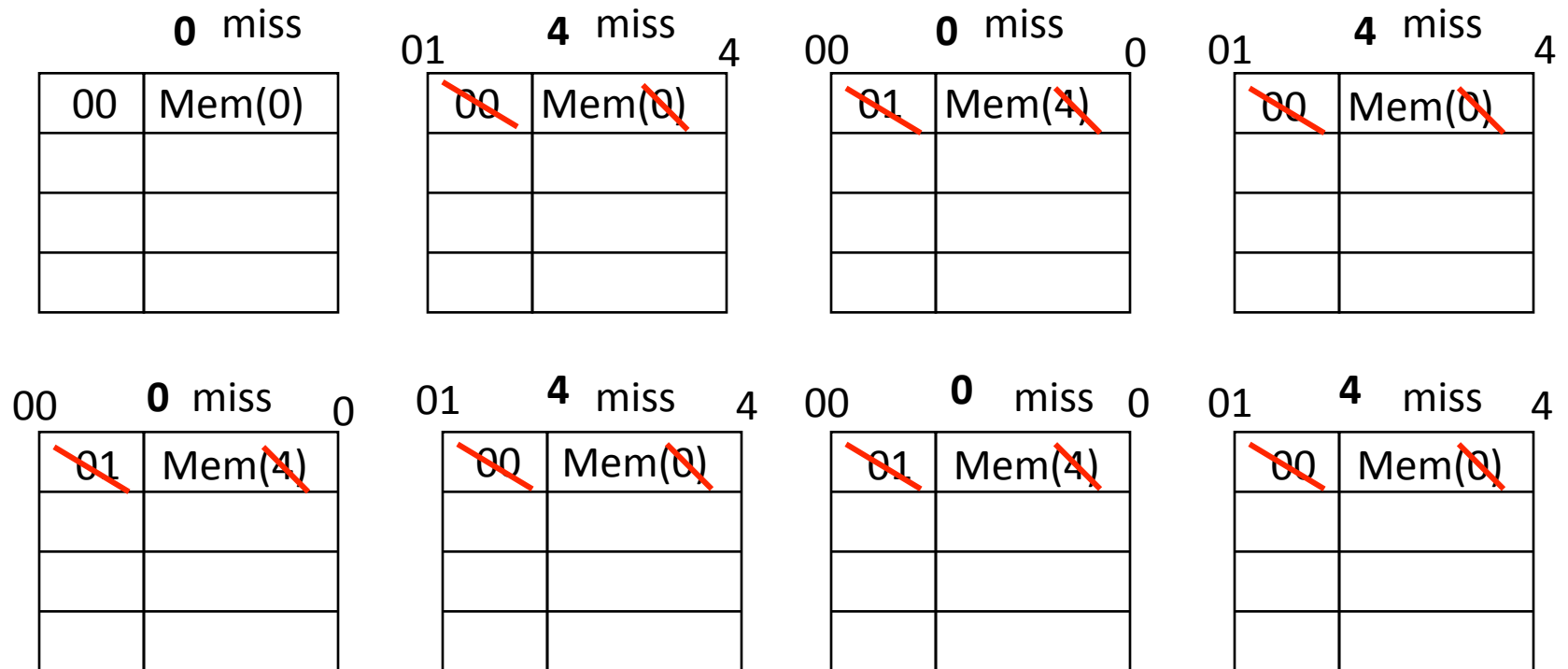
0

4

Example: Direct-Mapped \$ with 4 Single-Word Lines, Worst-Case Reference String

- Consider the main memory address reference string

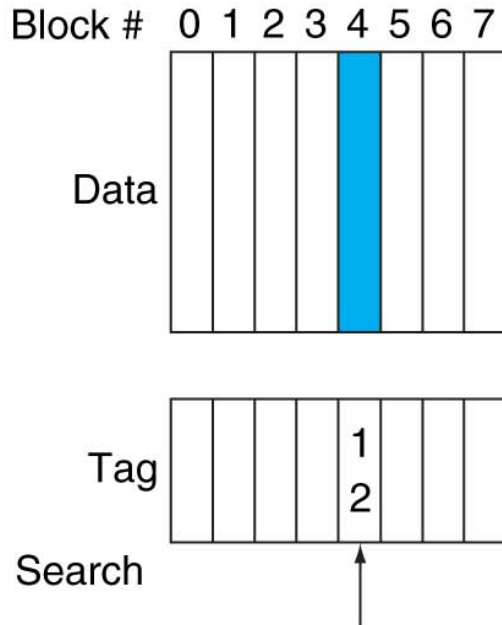
Start with an empty cache - all blocks
initially marked as not valid 0 4 0 4 0 4 0 4



- 8 requests, 8 misses
- Ping-pong effect due to conflict misses - two memory locations that map into the same cache block

Alternative Block Placement Schemes

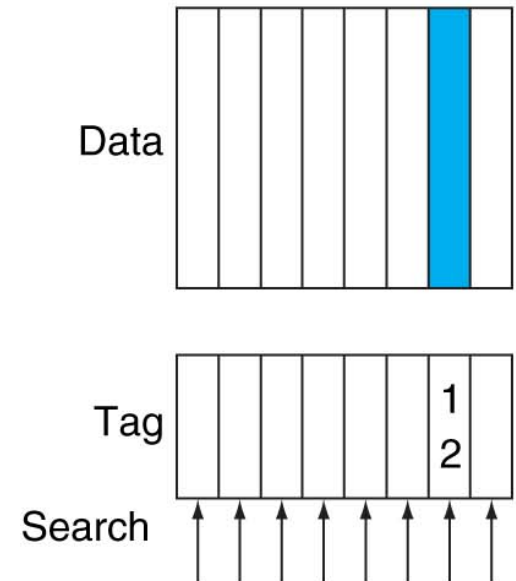
Direct mapped



Set associative



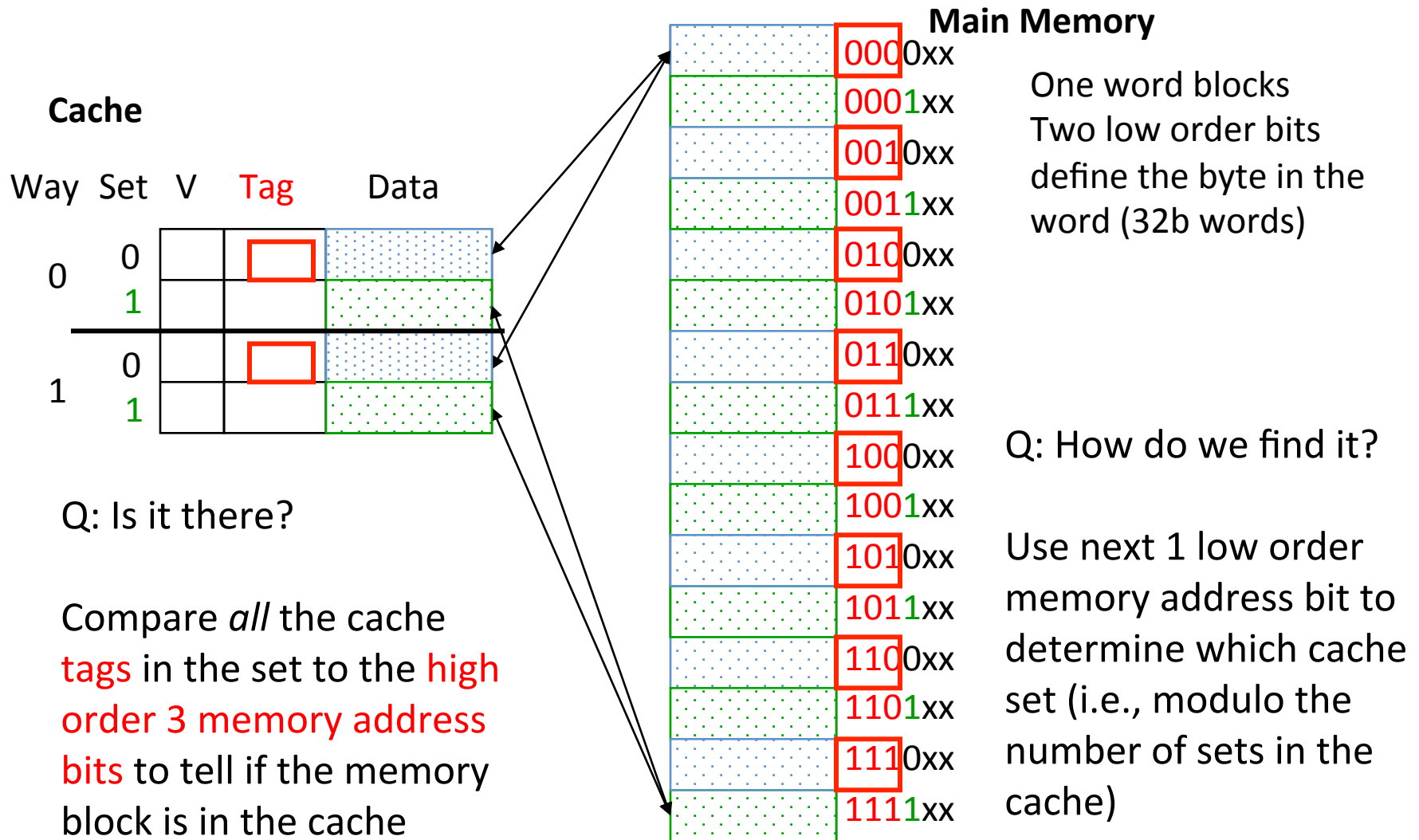
Fully associative



- DM placement: mem block 12 in 8 block cache: only one cache block where mem block 12 can be found— $(12 \bmod 8) = 4$
- SA placement: four sets x 2-ways (8 cache blocks), memory block 12 in set $(12 \bmod 4) = 0$; either element of the set
- FA placement: mem block 12 can appear in any cache blocks

Example: 2-Way Set Associative \$

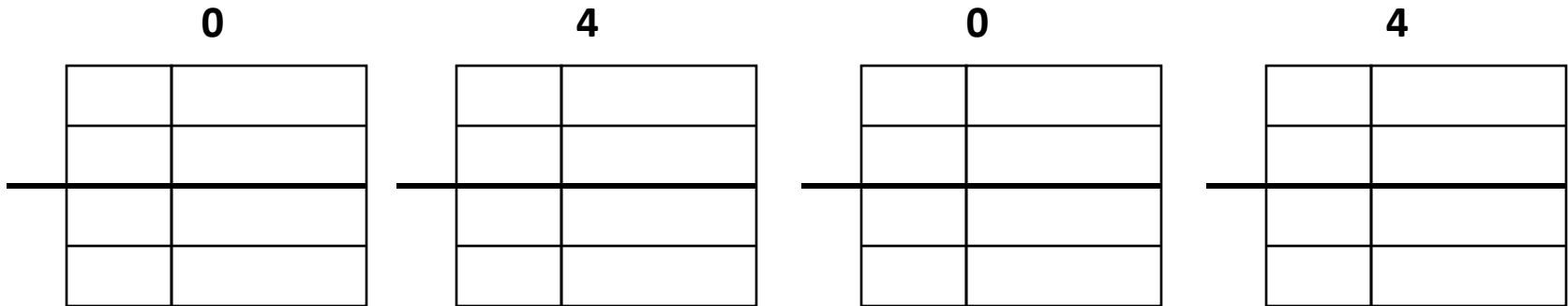
(4 words = 2 sets x 2 ways per set)



Example: 4 Word 2-Way SA \$ Same Reference String

- Consider the main memory word reference string

Start with an empty cache - all blocks
initially marked as not valid 0 4 0 4 0 4 0 4



Example: 4-Word 2-Way SA \$ Same Reference String

- Consider the main memory address reference string

Start with an empty cache - all blocks
initially marked as not valid 0 4 0 4 0 4 0 4

0 miss

000	Mem(0)

4 miss

000	Mem(0)
010	Mem(4)

0 hit

000	Mem(0)
010	Mem(4)

4 hit

000	Mem(0)
010	Mem(4)

- 8 requests, 2 misses
- Solves the ping-pong effect in a direct-mapped cache due to conflict misses since now two memory locations that map into the same cache set can co-exist!

Different Organizations of an Eight-Block Cache

One-way set associative (direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set 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								

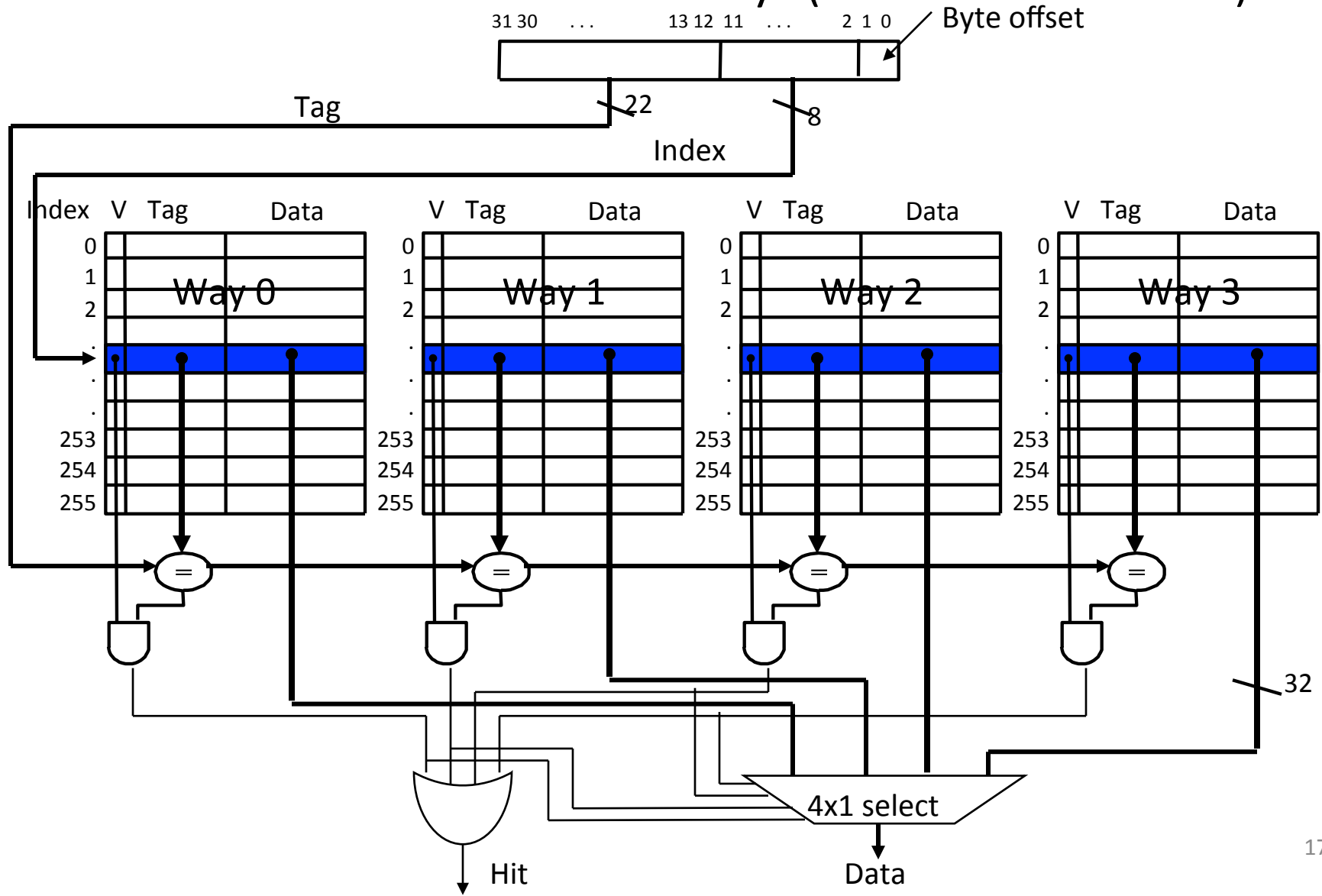
Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Total size of \$ in blocks is equal to *number of sets x associativity*. For fixed \$ size, increasing associativity decreases number of sets while increasing number of elements per set. With eight blocks, an 8-way set-associative \$ is same as a fully associative \$.

Four-Way Set-Associative Cache

- $2^8 = 256$ sets each with four ways (each with one block)



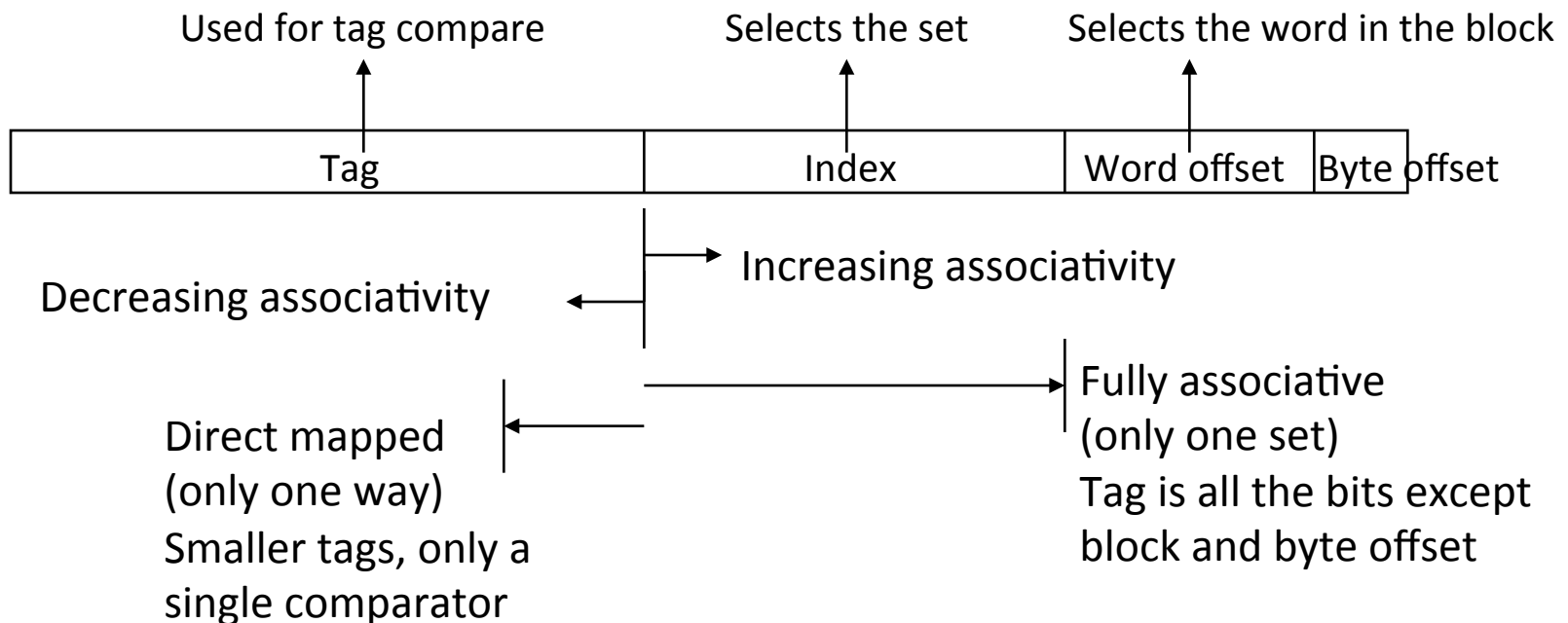
Range of Set-Associative Caches

- For a fixed-size cache, each increase by a factor of two in associativity doubles the number of blocks per set (i.e., the number or ways) and halves the number of sets – decreases the size of the index by 1 bit and increases the size of the tag by 1 bit

Tag	Index	Word offset	Byte offset
-----	-------	-------------	-------------

Range of Set-Associative Caches

- For a *fixed-size* cache, each increase by a factor of two in associativity doubles the number of blocks per set (i.e., the number or ways) and halves the number of sets – decreases the size of the index by 1 bit and increases the size of the tag by 1 bit

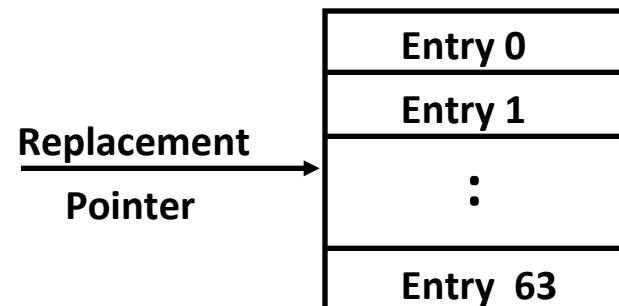


Costs of Set-Associative Caches

- N-way set-associative cache costs
 - N comparators (delay and area)
 - MUX delay (set selection) before data is available
 - Data available after set selection (and Hit/Miss decision).
DM \$: block is available before the Hit/Miss decision
 - In Set-Associative, not possible to just assume a hit and continue and recover later if it was a miss
- When miss occurs, which way's block selected for replacement?
 - **Least Recently Used** (LRU): one that has been unused the longest (principle of temporal locality)
 - Must track when each way's block was used relative to other blocks in the set
 - For 2-way SA \$, one bit per set → set to 1 when a block is referenced; reset the other way's bit (i.e., "last used")

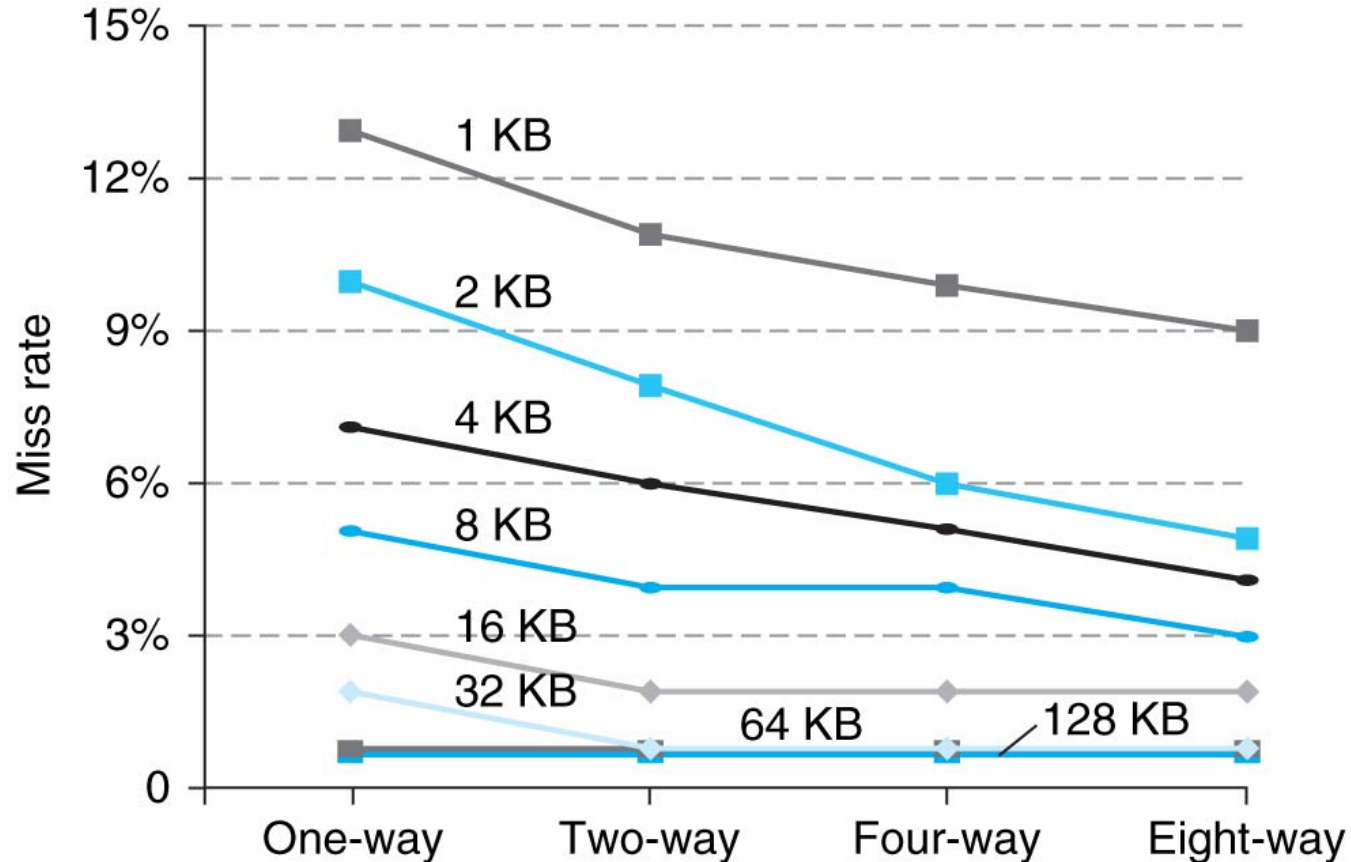
Cache Replacement Policies

- Random Replacement
 - Hardware randomly selects a cache evict
- Least-Recently Used
 - Hardware keeps track of access history
 - Replace the entry that has not been used for the longest time
 - For 2-way set-associative cache, need one bit for LRU replacement
- Example of a Simple “Pseudo” LRU Implementation
 - Assume 64 Fully Associative entries
 - Hardware replacement pointer points to one cache entry
 - Whenever access is made to the entry the pointer points to:
 - Move the pointer to the next entry
 - Otherwise: do not move the pointer
 - (example of “not-most-recently used” replacement policy)



Benefits of Set-Associative Caches

- Choice of DM \$ or SA \$ depends on the cost of a miss versus the cost of implementation



- Largest gains are in going from direct mapped to 2-way (20%+ reduction in miss rate)

Administrivia

- Project 2-1 due Sunday March 15th, 11:59PM
 - Use pinned Piazza threads!
 - We'll penalize those who ask, but don't search!
- Guerilla sections starting this weekend
 - Optional sections, focus on lecture/exam material, not projects
 - Vote for time slot on Piazza poll

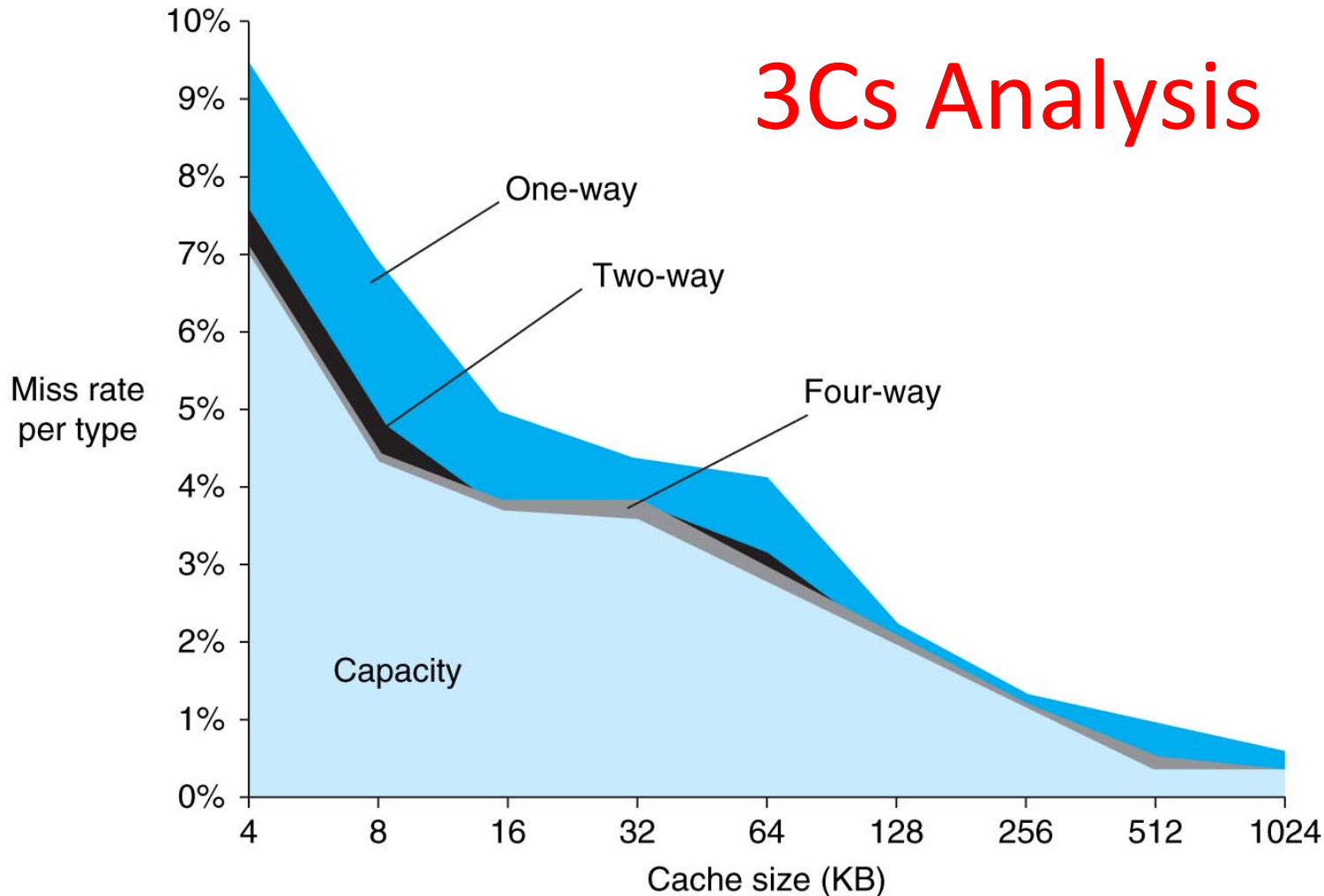
Understanding Cache Misses: The 3Cs

- **Compulsory** (cold start or process migration, 1st reference):
 - First access to block impossible to avoid; small effect for long running programs
 - Solution: increase block size (increases miss penalty; very large blocks could increase miss rate)
- **Capacity**:
 - Cache cannot contain all blocks accessed by the program
 - Solution: increase cache size (may increase access time)
- **Conflict** (*collision*):
 - *Multiple memory locations mapped to the same cache location*
 - *Solution 1: increase cache size*
 - *Solution 2: increase associativity (may increase access time)*

How to Calculate 3C's using Cache Simulator

1. *Compulsory*: set cache size to infinity and fully associative, and count number of misses
2. *Capacity*: Change cache size from infinity, usually in powers of 2, and count misses for each reduction in size
 - 16 MB, 8 MB, 4 MB, ... 128 KB, 64 KB, 16 KB
3. *Conflict*: Change from fully associative to n-way set associative while counting misses
 - Fully associative, 16-way, 8-way, 4-way, 2-way, 1-way

3Cs Analysis



- Three sources of misses (SPEC2000 integer and floating-point benchmarks)
 - Compulsory misses 0.006%; not visible
 - Capacity misses, function of cache size
 - Conflict portion depends on associativity and cache size

Improving Cache Performance

$AMAT = \text{Time for a hit} + \text{Miss rate} \times \text{Miss penalty}$

- Reduce the time to hit in the cache
 - E.g., Smaller cache
- Reduce the miss rate
 - E.g., Bigger cache
- Reduce the miss penalty
 - E.g., Use multiple cache levels

Impact of Larger Cache on AMAT?

- 1) Reduces misses (what kind(s)?)
- 2) Longer Access time (Hit time): smaller is faster
 - Increase in hit time will likely add another stage to the pipeline
- At some point, increase in hit time for a larger cache may overcome the improvement in hit rate, yielding a decrease in performance
- Computer architects expend considerable effort optimizing organization of cache hierarchy – big impact on performance and power!

Clickers: Impact of longer cache blocks on misses?

- For fixed total cache capacity and associativity, what is effect of longer blocks on each type of miss:
 - A: Decrease, B: Unchanged, C: Increase
- Compulsory?
- Capacity?
- Conflict?

Clickers: Impact of longer blocks on AMAT

- For fixed total cache capacity and associativity, what is effect of longer blocks on each component of AMAT:
 - A: Decrease, B: Unchanged, C: Increase
- Hit Time?
- Miss Rate?
- Miss Penalty?

Clickers/Peer Instruction:

For fixed capacity and fixed block size, how does increasing associativity effect AMAT?

A: Increases hit time, decreases miss rate

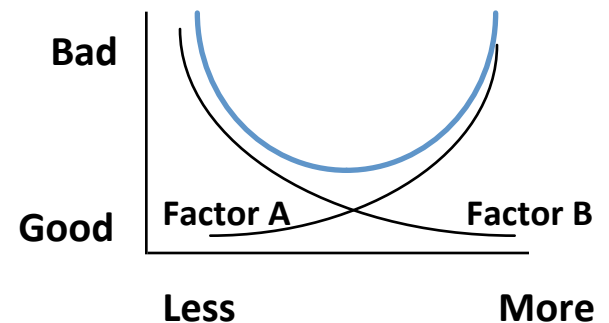
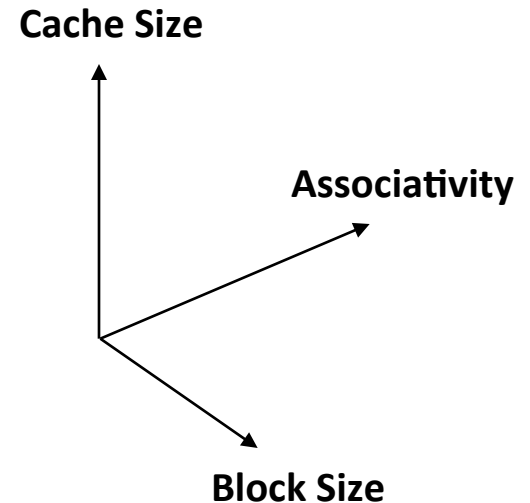
B: Decreases hit time, decreases miss rate

C: Increases hit time, increases miss rate

D: Decreases hit time, increases miss rate

Cache Design Space

- Several interacting dimensions
 - Cache size
 - Block size
 - Associativity
 - Replacement policy
 - Write-through vs. write-back
 - Write allocation
- Optimal choice is a compromise
 - Depends on access characteristics
 - Workload
 - Use (I-cache, D-cache)
 - Depends on technology / cost
- Simplicity often wins



And, In Conclusion ...

- Name of the Game: Reduce AMAT
 - Reduce Hit Time
 - Reduce Miss Rate
 - Reduce Miss Penalty
- Balance cache parameters (Capacity, associativity, block size)