CMSC 411
Computer Architecture

Lecture 21

Cache Performance

Lecture's Overview

- ***Previous Lecture:***
  - Memory hierarchy
    - ➔ Principles of locality
    - ➔ Types of memory
  - The basic of cache memory
    - ➔ Direct-mapped cache
    - ➔ Handling of cache misses
    - ➔ Consistency between cache and main memory

- ***This Lecture:***
  - Organization of main memory
  - Measuring and improving cache performance

Memory Hierarchy

Capacity
Access Time

Upper Level

Staging
Transfer Unit

faster

CPU Registers
100s Bytes
<10s ns

Prog./compiler
1-8 bytes

Cache
K Bytes
10-40 ns

cache cntl
8-128 bytes

Main Memory
M Bytes
70ns-1us

OS
512-4K bytes

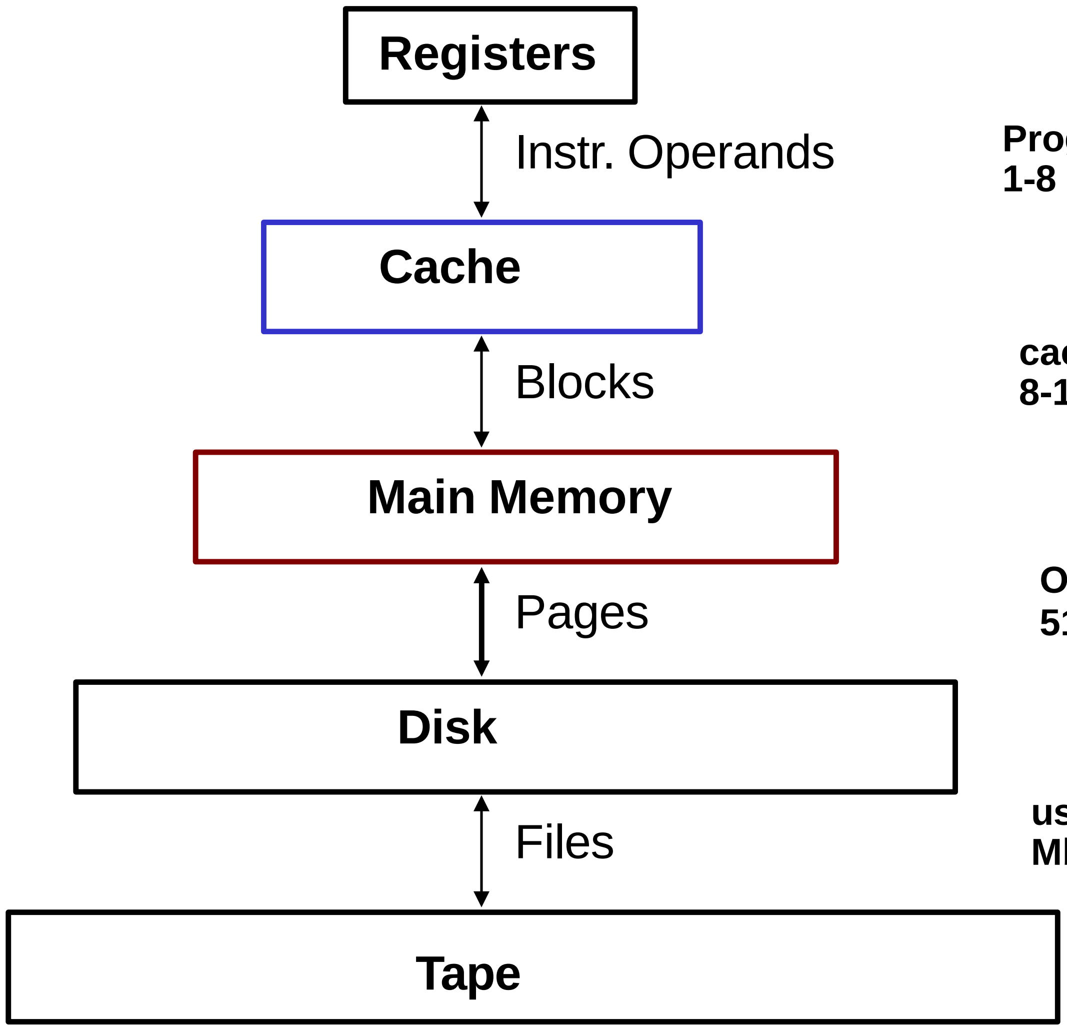
Disk
G Bytes
ms

user/operator
Mbytes

Tape
infinite
sec-min

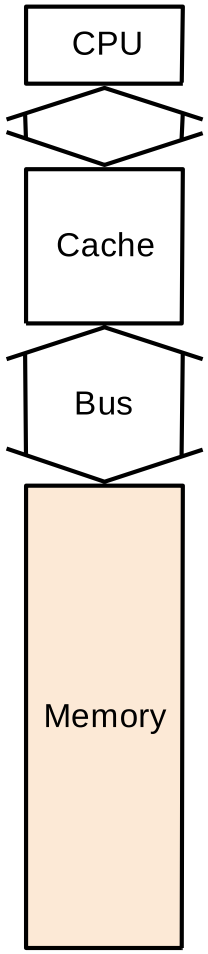
Larger

Lower Level

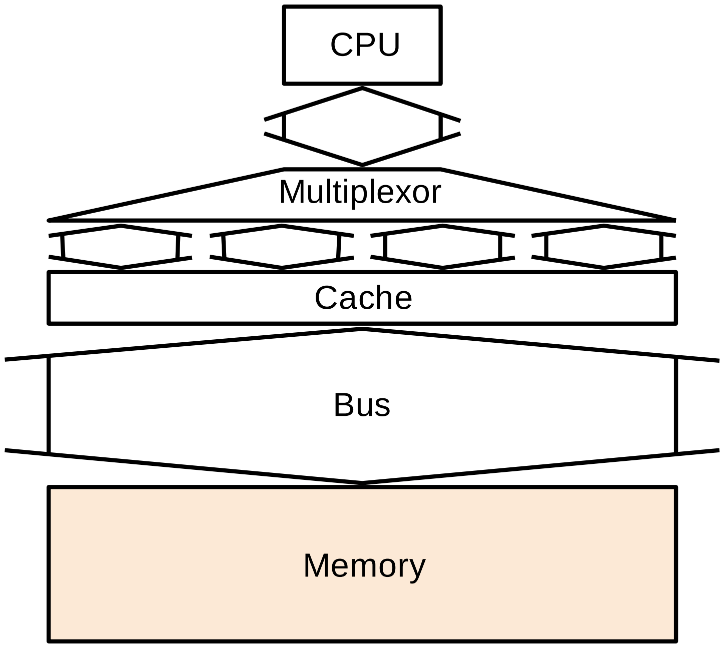


* Slide is courtesy of Dave Patterson

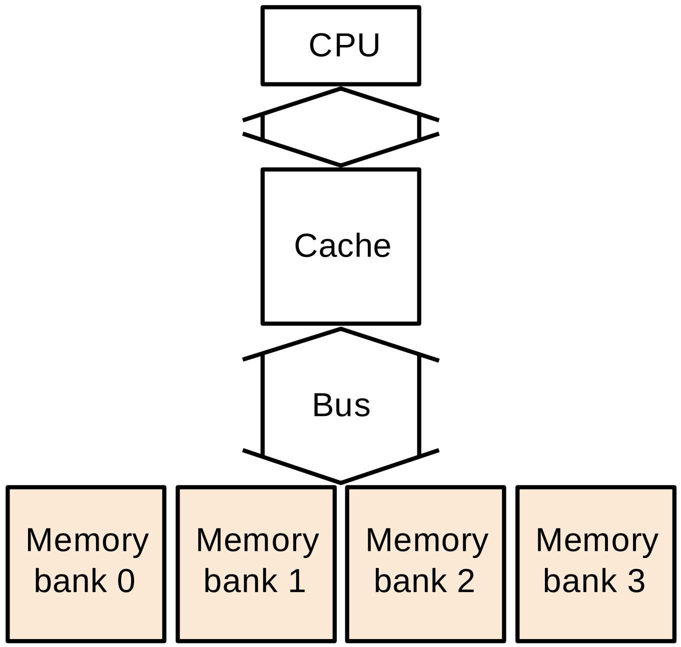
Memory Organization



a. One-word-wide
memory organization



b. Wide memory organization



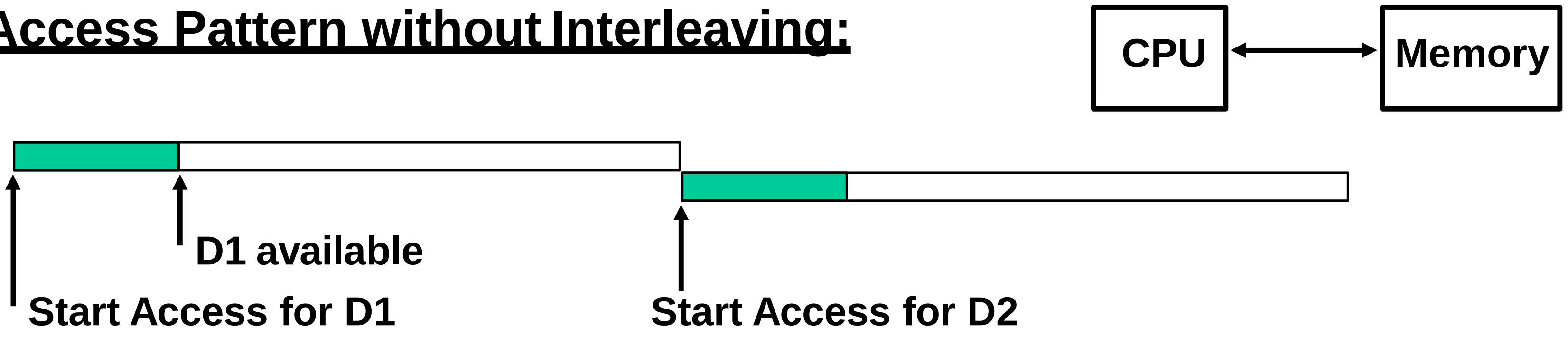
c. Interleaved memory organization

- *Simple*: CPU, Cache, Bus, Memory same width (32 bits)
- *Wide*: CPU/Mux 1 word; Mux/Cache, Bus, Memory N words
- *Interleaved*: CPU, Cache, Bus 1 word: Memory N Modules (4 Modules); example is *word interleaved*

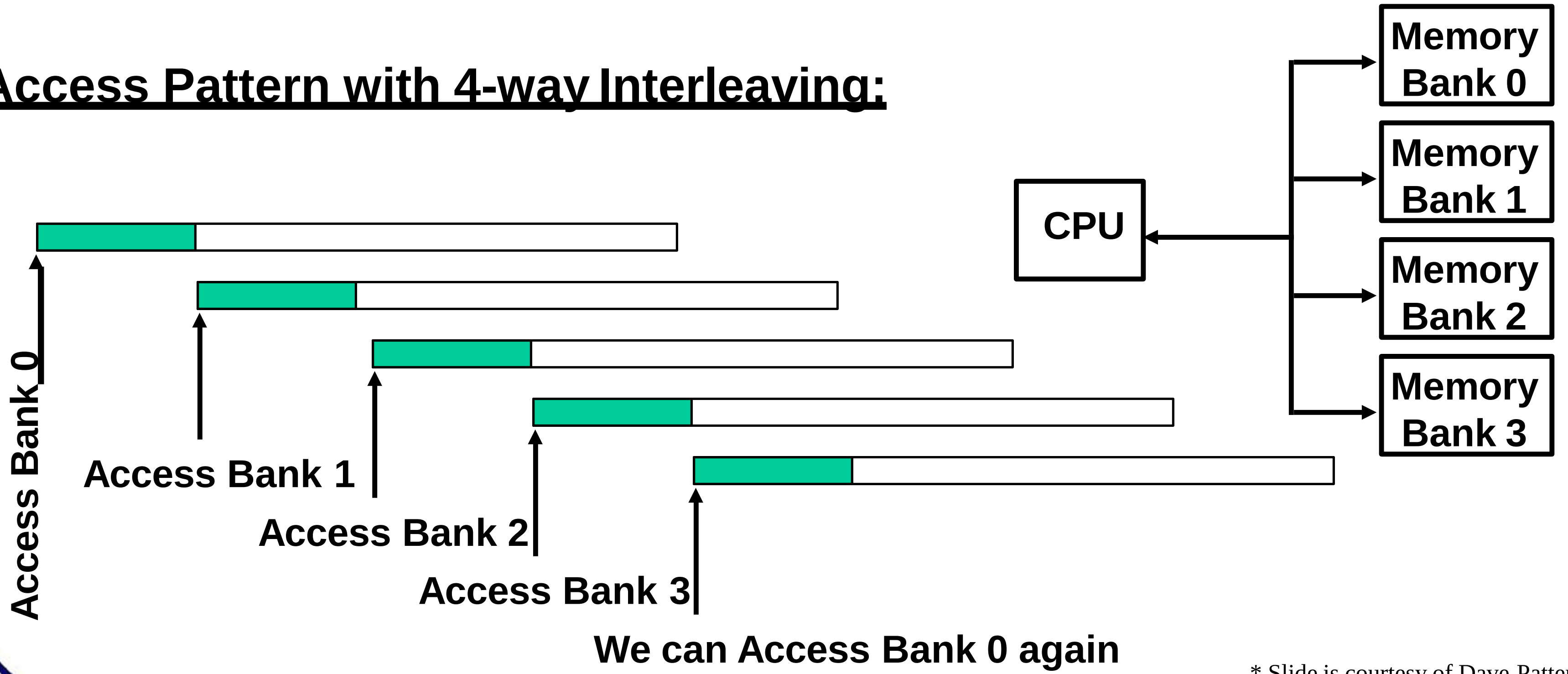
Memory organization would have significant effect on bandwidth

Memory Interleaving

- **Access Pattern without Interleaving:**



- **Access Pattern with 4-way Interleaving:**



* Slide is courtesy of Dave Patterson

Measuring Cache Performance

To enhance cache performance, one can:

- ❑ reduce the miss rate by diminishing blocks collision probability
- ❑ reduce the miss penalty by adding multi-level caching

$$\text{CPU time} = (\text{CPU execution clock cycles} + \text{Memory-stall clock cycles}) \times \text{Clock cycle time}$$

Where:

$$\text{Memory-stall clock cycles} = \text{Read-stall cycles} + \text{Write-stall cycles}$$

$$\text{Read-stall cycles} = \frac{\text{Read}}{\text{Program}} \times \text{Read miss rate} \times \text{Read miss penalty}$$

For write-through scheme:

Hard to control, assume enough buffer size

$$\text{Write-stall cycles} = \left(\frac{\text{Write}}{\text{Program}} \times \text{Write miss rate} \times \text{Write miss penalty}\right) + \text{Write buffer stalls}$$

Example

Assume an instruction cache miss rate for gcc of 2% and a data cache miss rate of 4%. If a machine has a CPI of 2 without any memory stalls and the miss penalty is 40 cycles for all misses, determine how much faster a machine would run with a perfect cache that never missed. Assume 36% combined frequencies for load and store instructions

Answer:

Assume number of instructions = I

The number of memory miss cycles = $I \times 2\% \times 40 = 0.8 \times I$

Data miss cycles = $I \times 36\% \times 4\% \times 40 = 0.56 \times I$

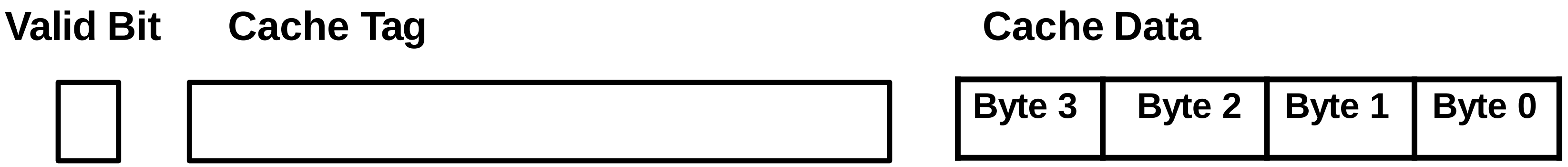
Total number of memory-stall cycles = $0.8\,I + 0.56\,I = 1.36\,I$

The CPI with memory stalls = $2 + 1.36 = 3.36$

$$\frac{\text{CPU time with stalls}}{\text{CPU time with perfect cache}} = \frac{I \times CPI_{stall} \times \text{Clock cycle}}{I \times CPI_{perfect} \times \text{Clock cycle}} = \frac{CPI_{stall}}{CPI_{perfect}} = \frac{3.36}{2}$$

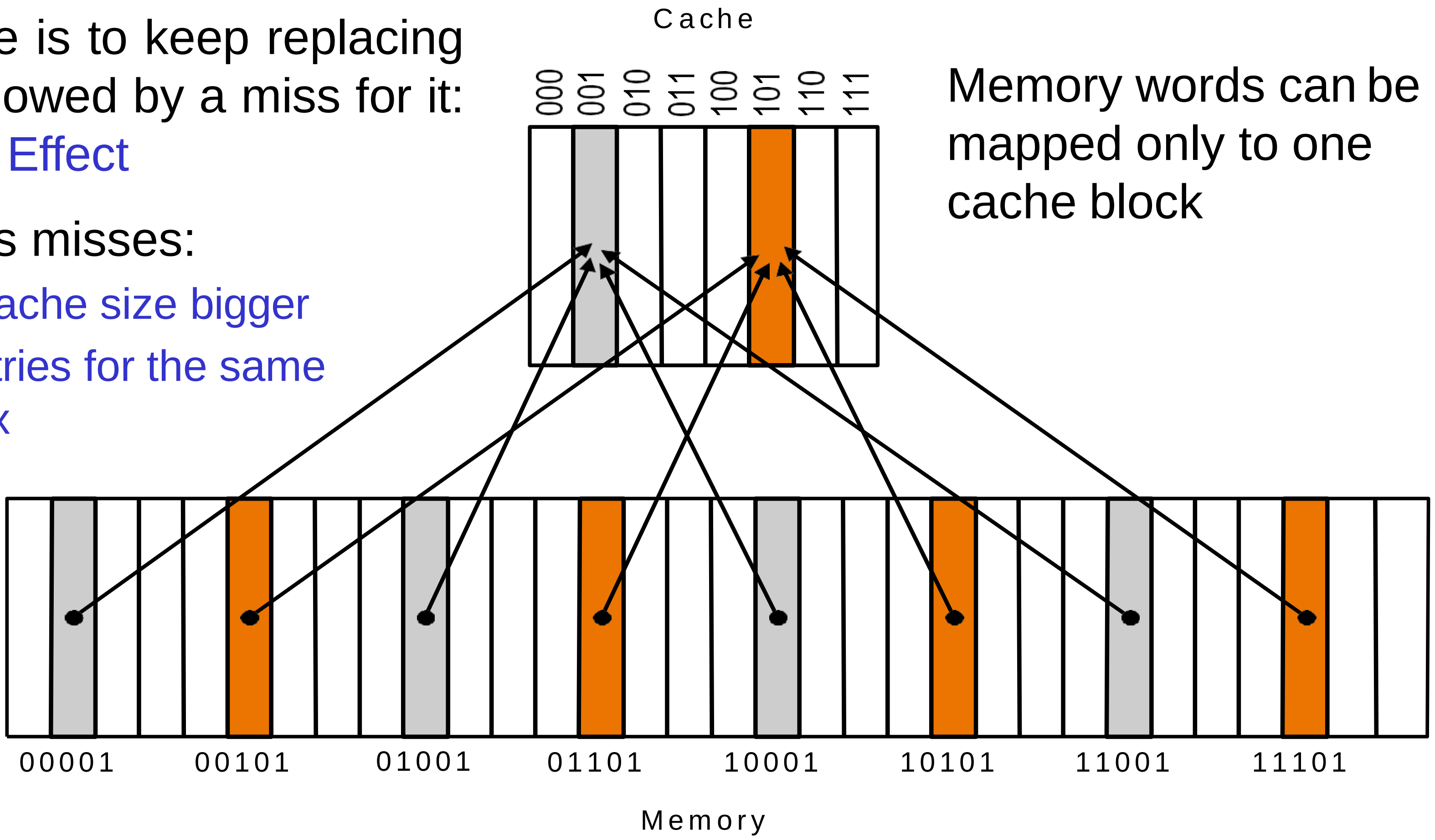
What happen if CPU gets faster?

Direct-Mapped Cache



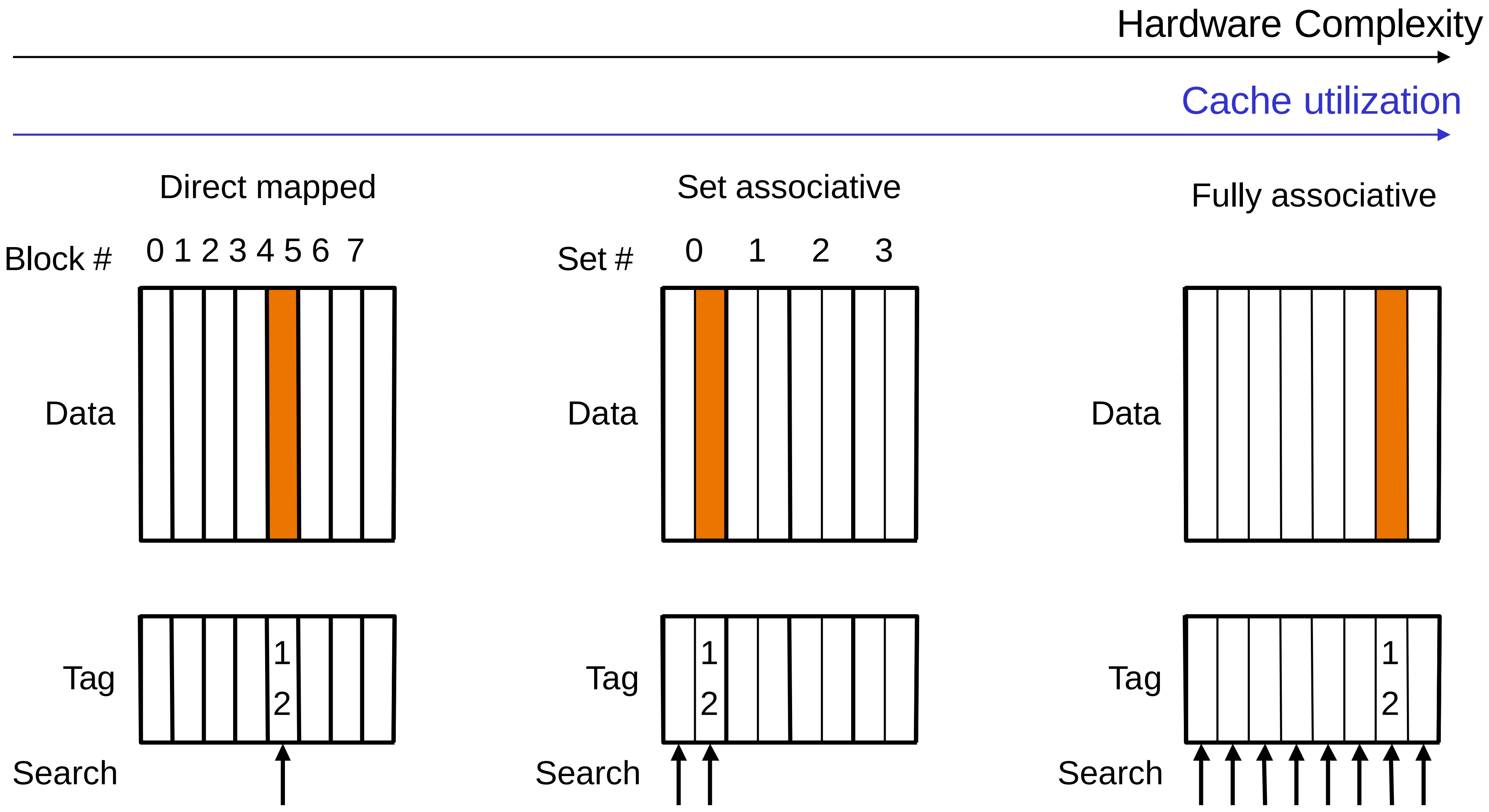
- ❑Worst case is to keep replacing a block followed by a miss for it: Ping Pong Effect
- ❑ To reduces misses:
  - ❶ make the cache size bigger
  - ❷ multiple entries for the same Cache Index

Memory words can be mapped only to one cache block



Cache block address = (Block address) modulo (Number of cache blocks)

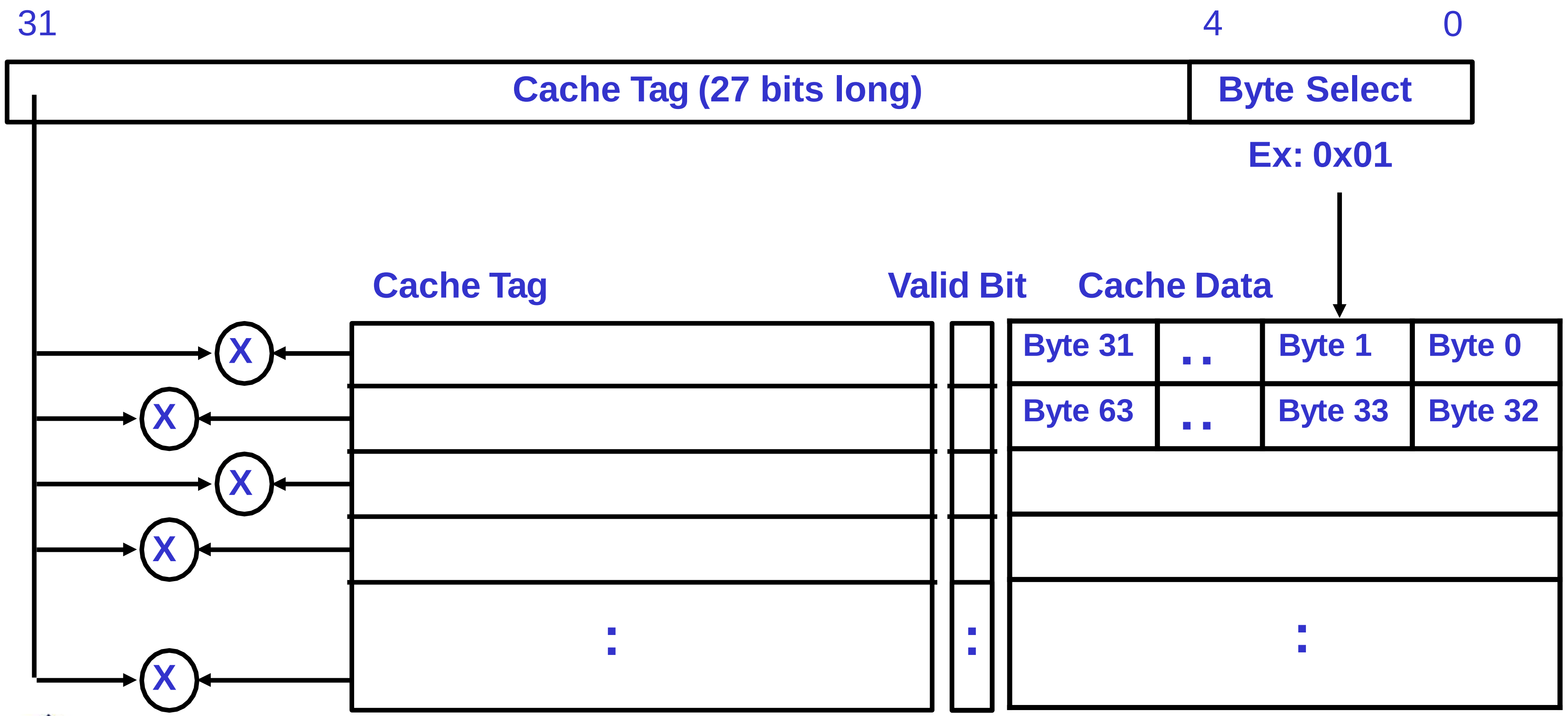
Block Placement



- Set number = (Block number) modulo (Number of sets in the cache)
- Increased flexibility of block placement reduces probability of cache misses

Fully Associative Cache

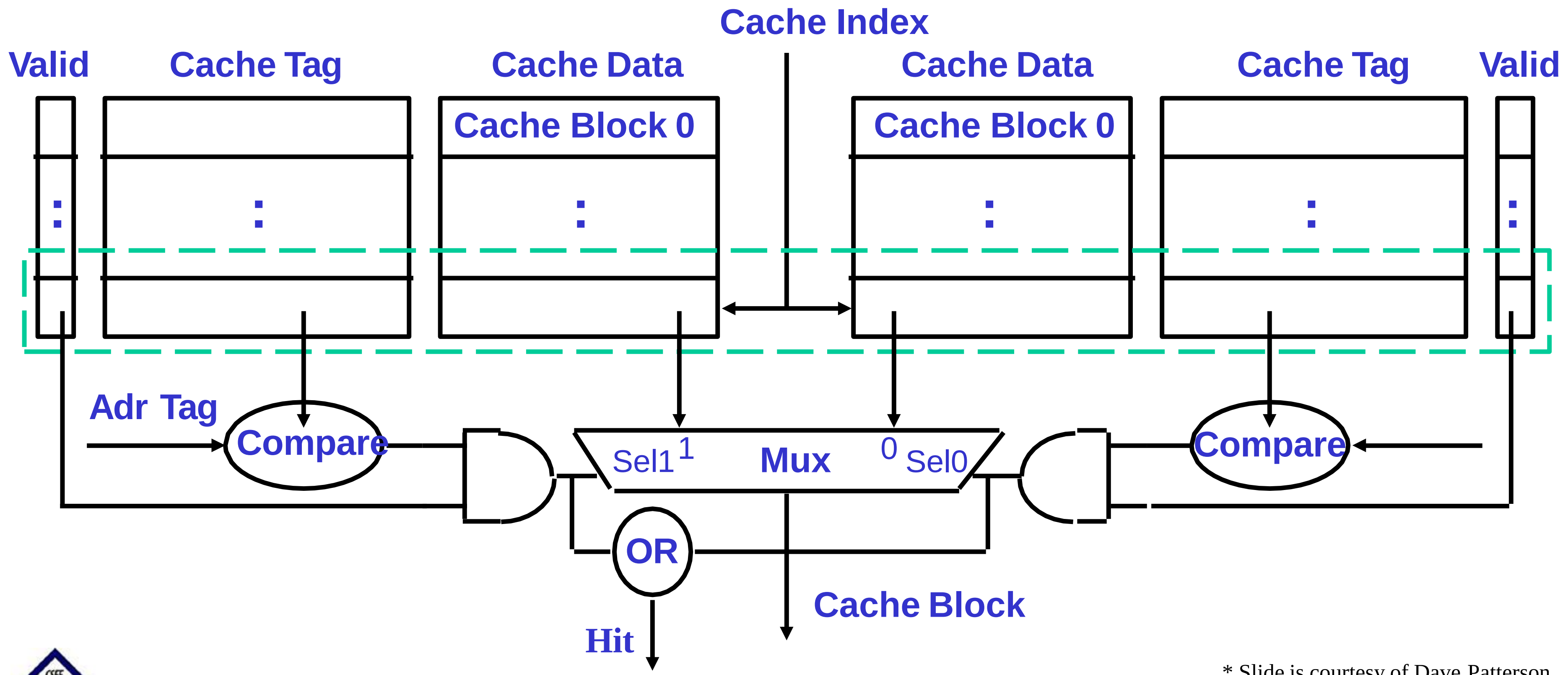
- Forget about the Cache Index
- Compare the Cache Tags of all cache entries in parallel
- Example: Block Size = 32 Bytes, we need N 27-bit comparators
- By definition: Conflict Miss = 0 for a fully associative cache



* Slide is courtesy of Dave Patterson

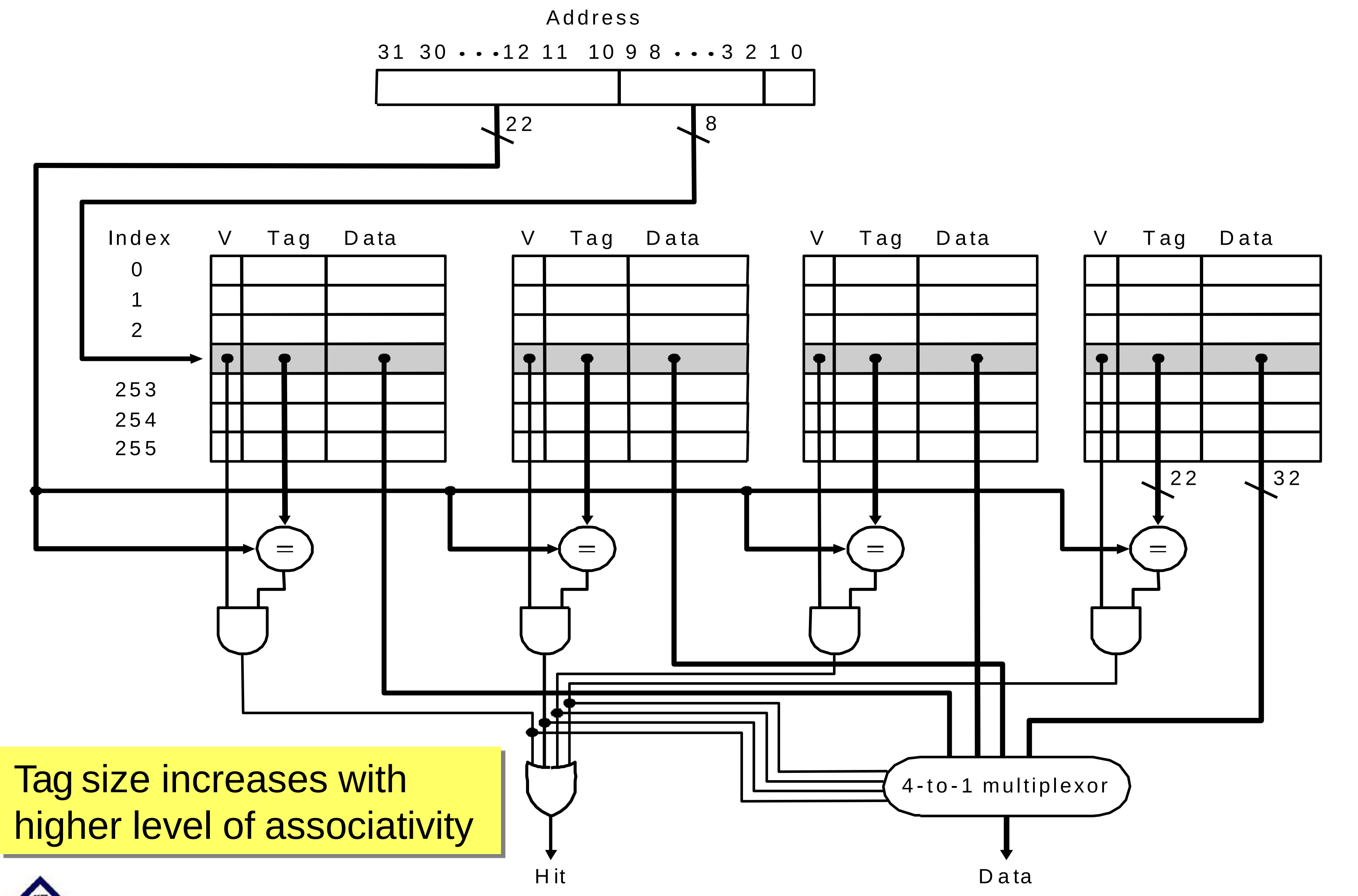
N-way Set Associative Cache

- ❑ N entries for each Cache Index ⇒ N direct mapped caches operate in parallel
- ❑ Example: Two-way set associative cache
  - ➔ Cache Index selects a "set" from the cache
  - ➔ The two tags in the set are compared in parallel
  - ➔ Data is selected based on the tag result



* Slide is courtesy of Dave Patterson

Locating a Block in Cache



Tag size increases with higher level of associativity

Block Replacement Strategy

- ❑ Straight forward for Direct Mapped since every block has only one location
- ❑ Set Associative or Fully Associative:
  - ➔ Random: pick any block
  - ➔ LRU (Least Recently Used)
    - requires tracking block reference
    - for two-way set associative cache a reference bit is attached to every block
    - more complex hardware is needed for higher level of cache associativity

Associativity	2-way		4-way		8-way	
Size	LRU	Random	LRU	Random	LRU	Random
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

- ❑ Empirical results indicates less significance of replacement strategy with increased cache sizes

* Slide is courtesy of Dave Patterson

Example

There are three caches, each consisting of four one-word blocks. One cache is direct-mapped, the second is two-way set associative and the third is fully associative. Find the number of misses for each organization given the following sequence of block addresses: 0, 8, 0, 6, 8.

Answer:

Direct-mapped cache: *5 misses*

Block address	Cache block
0	(0 modulo 4) = 0
6	(6 modulo 4) = 2
8	(8 modulo 4) = 0

Address of memory block accessed	Hit or Miss	Contents of cache block after reference			
		0	1	2	3
0	Miss	Memory[0]			
8	Miss	Memory[8]			
0	Miss	Memory[0]			
6	Miss	Memory[0]		Memory[6]	
8	Miss	Memory[8]		Memory[6]	

Example

2-way associative cache:

4 misses

Block address	Cache set
0	(0 modulo 2) = 0
6	(6 modulo 2) = 0
8	(8 modulo 2) = 0

Address of memory block accessed	Hit or Miss	Contents of cache block after reference			
		Set 0	Set 0	Set 1	Set 1
0	Miss	Memory[0]			
8	Miss	Memory[0]	Memory[8]		
0	Hit	Memory[0]	Memory[8]		
6	Miss	Memory[0]	Memory[6]		
8	Miss	Memory[8]	Memory[6]		

Fully associative cache: *3 misses*

Address of memory block accessed	Hit or Miss	Contents of cache block after reference			
		Block 0	Block 1	Block 2	Block 3
0	Miss	Memory[0]			
8	Miss	Memory[0]	Memory[8]		
0	Hit	Memory[0]	Memory[8]		
6	Miss	Memory[0]	Memory[8]	Memory[6]	
8	Hit	Memory[0]	Memory[8]	Memory[6]	

Performance of Multi-level Cache

Suppose we have a 500 MHz processor with a base CPI of 1.0 with no cache misses. Assume memory access time is 200 ns and average cache miss rate is 5%. Compare performance after adding a second level cache, with access time 20 ns, that reduces miss rate to main memory to 2%.

Answer:

The miss penalty to main memory = 200/cycle time

= $200 \times 500/1000$ = 100 clock cycles

Effective CPI = Base CPI + memory-stall cycles/instr. = $1 + 5\% \times 100 =$ 6.0

With two level caches:

The miss penalty for accessing 2nd cache = $20 \times 500/1000$ = 10 clock cycles

Total CPI = Base CPI + main memory-stall cycles/instruction +

secondary cache stall cycles/instruction

$= 1 + 2\% \times 100 + 5\% \times 10 =$ 3.5

Conclusion

- ❑ **Summary**
  - ➔ Memory organization
    - Main memory performance issues
    - Memory interleaving
  - ➔ Measuring and improving cache performance
    - Relationship between computer performance and cache
    - Factors that affect cache performance
    - Optimization techniques for cache performance
  - ➔ Multi-level caches
    - N-way set associative cache design
    - Performance set of associate cache memory
- ❑ **Next Lecture**
  - ➔ Virtual Memory

Read section 5.3 of the textbook