Cache Misses

- On cache hit, CPU proceeds normally
- On cache miss
  - Stall the CPU pipeline
  - Fetch block from next level of hierarchy
  - Instruction cache miss
    - Restart instruction fetch
  - Data cache miss
    - Complete data access

Measuring Cache Performance

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

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

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
  - Actual CPI = base CPI + I-cache cycles + D-cache cycles
- Miss cycles per instruction
  - I-cache: 0.02 x 100 = 2
  - D-cache: 0.36 x 0.04 x 100 = 1.44
- Actual CPI = 2 + 2 + 1.44
  - Ideal CPU is 5.44/ 2 times faster

Average Access Time

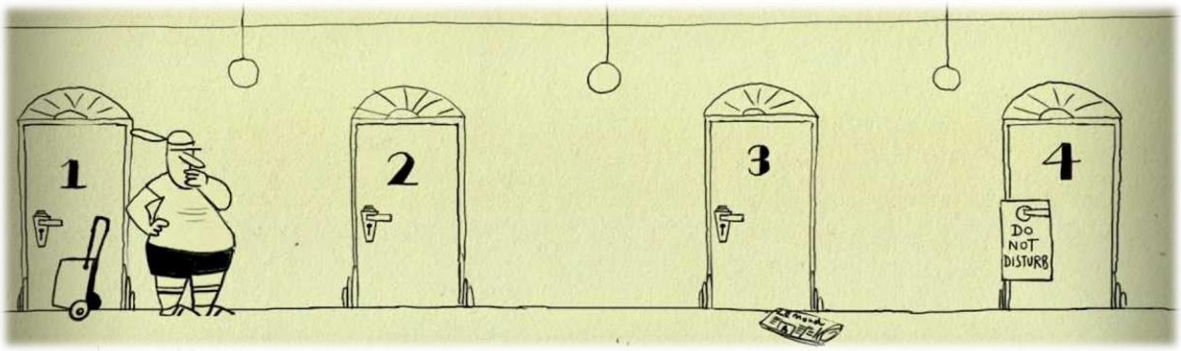
- Average memory access time (AMAT)
  - AMAT = Hit time + Miss rate × Miss penalty
  - At some point, the increase in hit time for a larger cache will overcome the improvement in hit rate leading to a decrease in performance
- Example
  - CPU with 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, I-cache miss rate = 5%
  - AMAT = 1 + 0.05 x 20 = 2

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

CSULB

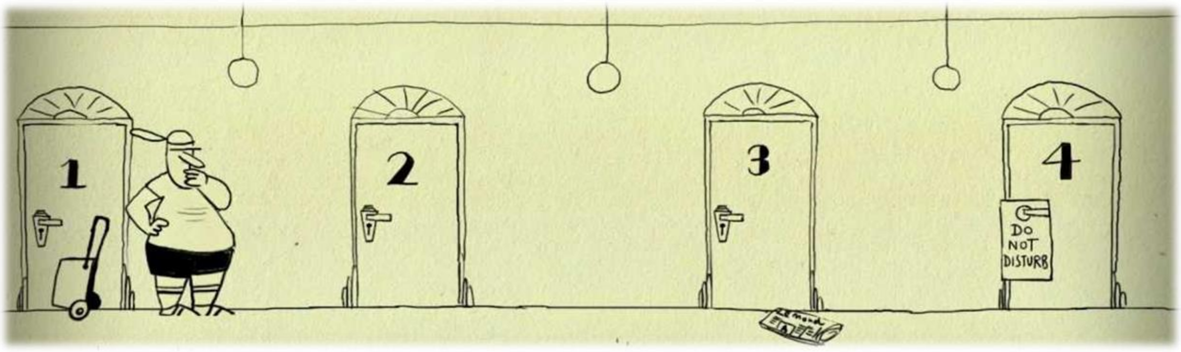
Hotel California



Agustin Avitia Buendia Coleman	Cuellar Denholm Espinoza Han Huang	Minaya Moreno Nguyen Pham	Phung Tecson Tran Vu Yip

CSULB

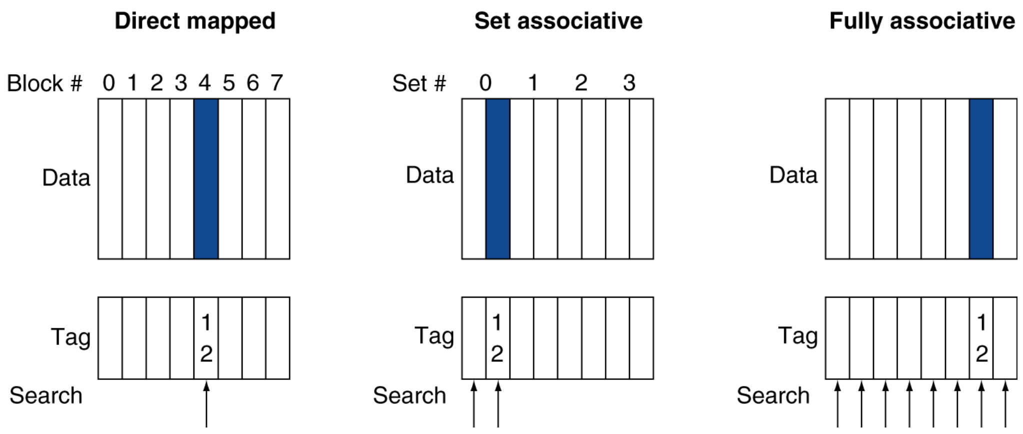
Hotel California



Agustin	Cuellar
Avitia	Denholm
Buendia	Espinoza
Coleman	Han
	Huang

Minaya	Phung
Moreno	Tecson
Nguyen	Tran
Pham	Vu
	Yip

Associative Cache Example



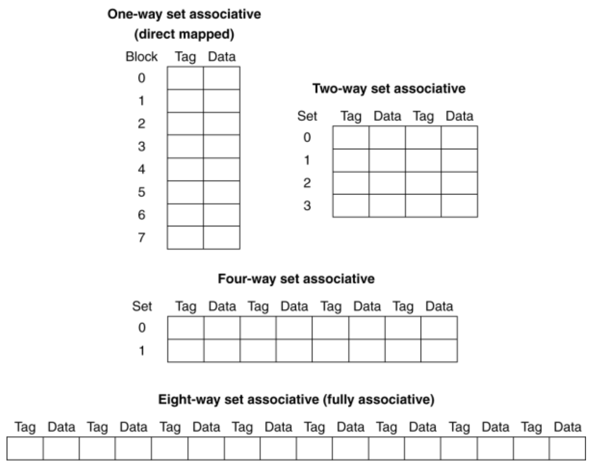
Associative Caches

- 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)

CSULB

Spectrum of Associativity

- For a cache with 8 entries


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

CSULB

Associativity Example

- Compare 4-block caches
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 6, 8
- Direct mapped

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

Associativity Example

- 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	M	Mem[0]			
8	0	M	Mem[0]	Mem[8]		
0	0	H	Mem[0]	Mem[8]		
6	0	M	Mem[6]	Mem[8]		
8	0	H	Mem[6]	Mem[8]		

Associativity Example

- Fully associative

Block address		Hit/miss	Cache content after access			
0		M	Mem[0]			
8		M	Mem[0]	Mem[8]		
0		H	"	"		
6		M	"	"	Mem[6]	
8		H	"	"	"	

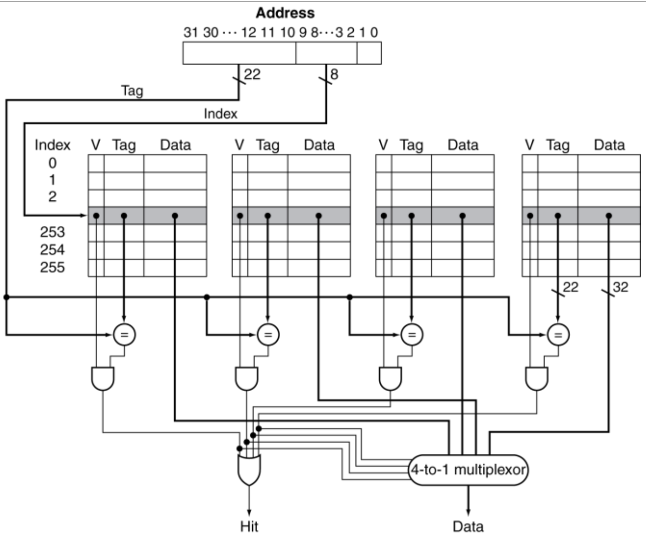
CSULB

How Much Associativity

- Increased associativity decreases miss rate
  - But with diminishing returns
- 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%

CSULB

Set Associative Cache Organization



CSULB

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

CSULB

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

CSULB

Multilevel Cache Example

- Given
  - CPU base CPI = 1, clock rate = 4GHz
  - Miss rate/instruction = 2%
  - Main memory access time = 100ns
- With just primary cache
  - Miss penalty = $\frac{100}{0.25} = \underline{400}$
  - Effective CPI = $1 + 0.02 \times 400 = 9$

CSULB

Example (cont.)

- Now add L-2 cache
  - Access time = 5ns
  - Global miss rate to main memory = 0.5%
- Primary miss with L-2 hit
  - Penalty = $\frac{5}{0.25} = \underline{20}$
- Primary miss with L-2 miss
- CPI = $1 + 0.02 \times 20 + 0.005 \times 400 = 3.4$
- Performance ratio = $9/3.4$

CSULB

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

CSULB

Virtual Machines

- Host computer emulates guest operating system and machine resources
  - Improved isolation of multiple guests
  - Avoids security and reliability problems
  - Aids sharing of resources
- Virtualization has some performance impact
  - Feasible with modern high-performance comptuers
- Examples
  - IBM VM/370 (1970s technology!)
  - VMWare
  - Microsoft Virtual PC

CSULB

Virtual Machine Monitor

- Maps virtual resources to physical resources
  - Memory, I/O devices, CPUs
- Guest code runs on native machine in user mode
  - Traps to VMM on privileged instructions and access to protected resources
- Guest OS may be different from host OS
- VMM handles real I/O devices
  - Emulates generic virtual I/O devices for guest

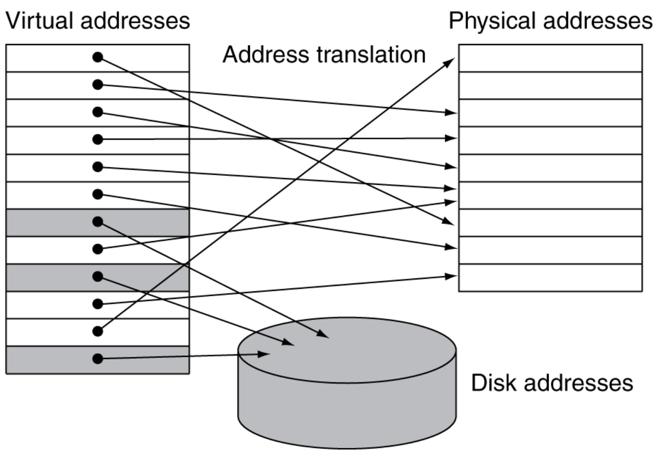
CSULB

Virtual Memory

- Use main memory as a "cache" for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Programs share main memory
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - VM "block" is called a page
  - VM translation "miss" is called a page fault

Address Translation

- Fixed-size pages (e.g., 4K)



Page Fault Penalty

- On page fault, the page must be fetched from disk
  - Takes millions of clock cycles
  - Handled by OS code
- Try to minimize page fault rate
  - Fully associative placement
  - Smart replacement algorithms

CSULB