

EN164: Design of Computing Systems

Lecture 28: Memory Systems 4

Professor Sherief Reda

<http://scale.engin.brown.edu>

Electrical Sciences and Computer Engineering
School of Engineering
Brown University
Spring 2011



[material from Patterson & Hennessy, 4th ed and Harris 1st ed]

Handling cache writes

- *How to handle cache writes?*
 - Write through
 - Write back
- *How to handle write misses?*

Write through

- On data-write hit, could just update the block in cache
 - But then cache and memory would be inconsistent
- Write through: also update memory
- But makes writes take longer
 - e.g., if base CPI = 1, 10% of instructions are stores, write to memory takes 100 cycles
 - Effective CPI = $1 + 0.1 \times 100 = 11$
- Solution: write buffer
 - Holds data waiting to be written to memory
 - CPU continues immediately
 - Only stalls on write if write buffer is already full

Write back

- Alternative: On data-write hit, just update the block in cache
 - Keep track of whether each block is dirty
- When a dirty block is replaced
 - Write it back to memory
 - Can use a write buffer to allow replacing block to be read first

Write allocation on write misses

- What should happen on a write miss?
- Alternatives for write-through
 - Allocate on miss: fetch the block
 - Write around: don't fetch the block
 - Since programs often write a whole block before reading it (e.g., initialization)
- For write-back
 - Usually fetch the block

Sources of misses

- **Compulsory misses** (aka cold start misses)
 - First access to a block
- **Capacity misses**
 - Due to finite cache size
 - A replaced block is later accessed again
- **Conflict misses** (aka collision misses)
 - In a non-fully associative cache
 - Due to competition for entries in a set
 - Would not occur in a fully associative cache of the same total size

Cache design trade-offs

Design change	Effect on miss rate	Negative performance effect
Increase cache size	Decrease capacity misses	May increase access time
Increase associativity	Decrease conflict misses	May increase access time
Increase block size	Decrease compulsory misses	Increases miss penalty.

Replacement choices

- Direct mapped: no choice
- Set associative
 - Prefer non-valid entry, if there is one
 - Otherwise, choose among entries in the set
- *What is the optimal entry to remove?*
- Random
- Least-recently used (LRU)
 - Choose the one unused for the longest time
 - Simple for 2-way, manageable for 4-way, too hard beyond that

LRU Replacement example

MIPS assembly

lw \$t0, 0x04(\$0)

lw \$t1, 0x24(\$0)

lw \$t2, 0x54(\$0)

(a)

V	U	Tag	Data	V	Tag	Data	Set Number
							3 (11)
							2 (10)
							1 (01)
							0 (00)

(b)

V	U	Tag	Data	V	Tag	Data	Set Number
							3 (11)
							2 (10)
							1 (01)
							0 (00)

LRU Replacement example

MIPS assembly

lw \$t0, 0x04(\$0)

lw \$t1, 0x24(\$0)

lw \$t2, 0x54(\$0)

Way 1				Way 0			
V	U	Tag	Data	V	Tag	Data	
0	0			0			Set 3 (11)
0	0			0			Set 2 (10)
1	0	00...010	mem[0x00...24]	1	00...000	mem[0x00...04]	Set 1 (01)
0	0			0			Set 0 (00)

(a)

Way 1				Way 0			
V	U	Tag	Data	V	Tag	Data	
0	0			0			Set 3 (11)
0	0			0			Set 2 (10)
1	1	00...010	mem[0x00...24]	1	00...101	mem[0x00...54]	Set 1 (01)
0	0			0			Set 0 (00)

(b)

Multi-level 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

Multi-level 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 = $100\text{ns}/0.25\text{ns} = 400$ cycles
- Effective CPI = $1 + 0.02 \times 400 = 9$

Multi-level cache example

- Now add L-2 cache
 - Access time = 5ns
 - Global miss rate to main memory = 0.5%
- Primary miss with L-2 hit
 - Penalty = $5\text{ns}/0.25\text{ns} = 20$ cycles
- Primary miss with L-2 miss
 - Extra penalty = 500 cycles
- $\text{CPI} = 1 + 0.02 \times 20 + 0.005 \times 400 = 3.4$
- Performance ratio = $9/3.4 = 2.6$

Interactions between cache and superscalar architecture

- Out-of-order CPUs can execute instructions during cache miss
 - Pending store stays in load/store unit
 - Dependent instructions wait in reservation stations
 - Independent instructions continue
- Effect of miss depends on program data flow
 - Much harder to analyze
 - Use system simulation

Cache summary

- Cache is on-chip memory built using 6T SRAM cells
- Cache organization:
 - Direct
 - Fully associative
 - N-way set associative
- Cache write techniques
- Cache block replacement techniques (for associative)
- Multi-level cache