CS 354 - Machine Organization & Programming
Thursday, October 24, 2019

Project p3 (6%): DUE at 10 pm on Monday, October 28th

Homework hw4 (1.5%): DUE at 10 pm on Thursday, October 31st

Last Time

Caching Basic Idea
Designing a Cache - Blocks
Designing a Cache - Sets and Tags
Basic Cache Lines
Basic Cache Operation
Basic Cache Practice
Direct Mapped Cache

Today

Set Associative Cache (from last time)
Replacement Policies
Fully Associative Cache
Writing to Caches
Cache Performance Metrics
Cache Parameters and Performance

Next Time

Finish Caching
Assembly Language
Read: B&O 3 Intro, 3.1 - 3.4

Replacement Policies

Assume the following sequence of memory blocks

are fetched into the same set of a 4-way associative cache that is initially empty:

`b1, b2, b3, b1, b3, b4, b4, b7, b1, b8, b4, b9, b1, b9, b9, b2, b8, b1`

1. *Random Replacement*

→ Which of the following four outcomes is possible after the sequence finishes? Assume the initial placement is random.

	L0	L1	L2	L3	
1.	b9	b1	b8	b2	✓
2.	b1	b2	--	b8	No
3.	b1	b4	b7	b3	OK, No.
4.	b1	b2	b8	b1	No → no duplicate.

empty line used before replace.

1. L0 L1 L2 L3 → L0 L1 L2 L3
   b3 b1 b4 b2 → b1 b4 b2 b3
   b7 b8 b7
   b4 b8
   b9 b9

2. *Least Recently Used* **(LRU)**

need to track when a line is last used.
use a LRU queue. - when used move it to front.

→ What is the outcome after the sequence finishes? Assume the initial placement is in ascending line order (left to right below).

L0 L1 L2 L3
b1 b2 b3 b4
b7 b8 b8
b9 b2

LRU Q

Front
b1, b8, b2, b9, b7, b4, b3, b2
rear

Finish: b1, b9, b2, b8.

3. *Least Frequently Used* **(LFU)**

need to track how often a line is used.
each line has a counter.
- zeroed with line gets new block.
- incremented each time line is accessed.
- Ties choose any one of the tieing blocks.

→ Which blocks will remain in the cache after the sequence finishes?

b1'' b2' b3'' b4''
b7'
b8'

Final result: b1 b4 b8 b9.

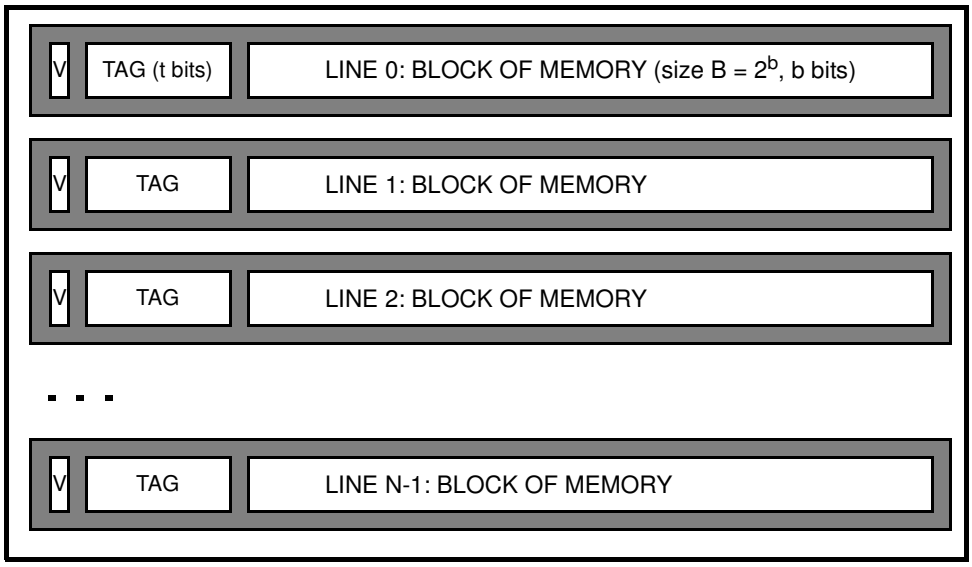
✵ *Exploiting replacement policies* to improve performance isn't easy so pregramers don't.

Fully Associative Cache

Fully Associative Cache

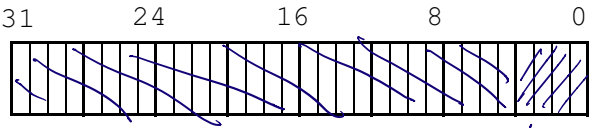
is a cache having only one set so that any mem block can be stored in any line.

- must search entire cache in worst case – a miss.
- complex circuitry to search all lines.



→ What is the address breakdown if blocks are 32 bytes?

32-bit Address Breakdown



t bits. b bits.

= block number.

→ How many lines should a fully associative cache have?

First decide cache size. and block size

Then E = C/B.

➢ Why isn't it possible for E > C/B?

✵ appropriate only for very small caches. used to be used for L1 but not today.

Writing to a Cache

- *Reading data copies of* a block of main memory into all cache levels.
- *Writing data requires that* these copies are kept consistent.

Write Hits

occur when writing to a block that is in this cache

→ When should a block be updated in lower memory levels?

1. Write Through: update next lower level immediately.
   + simply by pass this cache.
   - slower since must wait for lower level
   - more bus traffic, must pass write to lower level.
2. Write Back: update next lower level only when that changed line is evicted.
   must track if a line is changed.
   Dirty bit added to each line. 1 = changed, 0 = not.
   + Fast, no waiting.
   + less bus traffic.

Write Misses

occur when writing to a block that is not in this cache.

→ Should space be allocated in this cache for the block being changed?

1. No Write Allocate: write directly to next lower level.
   + less bus traffic. just passing the write.
2. Write Allocate: cache the block and then write it.
   - more bus traffic to copy entire block into this cache.

Typical Designs

1. Write Through paired with no write allocate.
2. Write Back paired with write allocate

→ Which best exploits locality? 2.

Cache Performance Metrics

Hit Rate # hits / # memory references

Higher is better.

Hit Time time to move a word cache to CPU.

= set selection + line matching.
+ word extraction.

Shorter is better.

Miss Penalty Additional time required to process a miss.

shorter is better.

L1 hit	4 cycles.
L1 miss served from L2	+6 cycles.
L1 miss served from L3	+25 cycles
L1 miss served from MM	+100 cycles to serve first 16 bytes. +45 cycles to finish block.

Cache Parameters and Performance

Larger Blocks (S and E unchanged)

hit rate better ↑ spatial

hit time same

miss penalty worse larger block are slower to transfer over same size bus.

THEREFORE block sizes are small.

More Sets (B and E unchanged)

hit rate better ↑ temporal.

hit time worse more sets slows set selection.

miss penalty same.

THEREFORE faster caches (L1) are smaller
slower caches (L3) are larger.

More Lines E per Set (B and S unchanged)

hit rate better ↑ temporal.

hit time worse more line slows line matching.

miss penalty worse more lines slows choosing a victim.

THEREFORE faster caches (L1) have fewer lines/sets.
slower caches (L3) have more lines/set.

Intel Quad Core i7 Cache (gen 7)

all: 64 byte block, use pseudo LRU, write back

L1: 32KB, 4-way Instruction & 32KB 8-way Data, no write allocate
L2: 256KB, 8-way, write allocate
L3: 8MB, 16-way (2MB/Core shared), write allocate