

CSCI 210: Computer Architecture

Lecture 36: Caches V

Stephen Checkoway

Slides from Cynthia Taylor

Oberlin College

CS History: IBM System/360 Model 85



- First computer to have cached memory
- Shipped in December of 1969
- Had either 16 KB or 32 KB cache
 - Modern systems have around 64 MB
- IBM only built about 30 of them

Unidentified US government agent, Public domain, via Wikimedia Commons

CACHE SIMULATOR PROJECT

Cache Simulator

- Take in a datatrace of load/stores from a real program
- Simulate running the program on a given cache
- Calculate how well a given cache would perform for that trace

Cache Parameters

- Always: Write-allocate, write-back, LRU replacement
- Change:
 - Cache size
 - Block size
 - Associativity
 - Miss penalty

Address Trace

Load/Store Address InstructionCount

```
# 0 7fffed80 1
# 0 10010000 10
# 0 10010060 3
# 1 10010030 4
# 0 10010004 6
# 0 10010064 3
# 1 10010034 4
```

L/S: 0 for load, 1 for store

Simulation Results

Simulation results:

execution time	52268708	cycles
instructions	5136716	
memory accesses	1957764	
overall miss rate	0.79	
load miss rate	0.88	
CPI	10.18	
average memory access time	24.07	cycles
dirty evictions	225876	
load_misses	1525974	
store_misses	30034	
load_hits	205909	
store_hits	195847	

What do you need to do?

- Create data structures that emulate a cache
- For each load/store instruction, find where the data would go in the cache, check if it's already there
- Calculate number of miss penalty cycles, load misses, store misses, instructions, etc.

Implementation hints

- Create a data structure that contains all of the data you need to keep for one cache entry (valid bit, dirty bit, tag) to keep this data in one place; don't keep a vector of valid bits, a vector of dirty bits, and a vector of tags for example
- Create a data structure that keeps track of a set of cache entries and the LRU metadata for the set
- Keep track of counts of events (instructions, misses, hits, dirty evictions, etc.) not statistics; e.g., don't have a CPI field
- Compute the statistics from the counts

Questions on Cache Final Project?

We have an 8 byte block size, direct-mapped, 16 kB cache. For a given byte address, to find the offset

- A. Mod by 8
- B. Mod by 16
- C. Divide by 8
- D. Divide by 16
- E. None of the above

If we are simulating a cache and not actually accessing data, do we need to use the offset?

A. Yes

B. No

We have an 8 byte block size, direct-mapped, 16 kB cache. How many rows will this cache have?

A. 16

*Recall a kB is 1024 bytes

B. 2×1024

C. 8×1024

D. 16×1024

We have an 8 byte block size, direct-mapped, 16 kB cache. How can we find the index from an address?

- A. Divide by 8
- B. Divide by $(16 * 1024)$
- C. Mod by $16 * 1024$
- D. Divide by 8, then mod by $(2 * 1024)$
- E. None of the above

We have an 8 byte block size, 2-way set associative, 16 KB cache. How can we find the index from an address?

- A. Divide by 8, then mod by 1024
- B. Divide by 8, then mod by $(2 * 1024)$
- C. Divide by 8, then mod by $(2 * 2 * 1024)$
- D. Divide by 16, then mod by $(2 * 1024)$
- E. None of the above

We have an 8 byte block size, direct mapped, 16 KB cache. How can we find the tag of an address?

A. Divide by 8

B. Divide by $8 * 1024$

C. Divide by $16 * 1024$

D. None of the above

Questions?

Reading