CSE 120
Day 14 Notes

Elijah Hantman

DM Cache and Cache Writes

Handling a Cache Miss

- If the cache reports a hit
  Continue as usual
- If the cache reports a miss
  Cache miss has to be done in collaboration with processor control unit that initiates the memory access and refills the cache.
- Cache misses create stalls while waiting for memory.
- Cache misses increase CPI.

Handling Cache Writes

- Writing causes additional issues
- Assume the address is already loaded in cache
- If we write a new value to that address we can store the new data in the cache and avoid a memory access.

Problems with Cache Writing?

- Cache and memory are desynced.
- How can we ensure subsequent loads return the right value?

Write-Through Caches

- Solves by forcing all writes to update both cache and main memory.
- Forces all writes to access memory
  No cache advantage.

Write Buffers

- Buffer allows devices to work at average data rate rather than peak.
- Write buffers stores data before being committed to memory, amortizes writes.

Write Back Caches

- Memory is not updated until a block needs to be replaced
- The highest location of a block has the correct data.
- Data only moves down the memory hierarchy when a block is evicted.
- Makes cache misses more expensive

Discussion

- Requries dirty bit to keep track of writes
- Write back is more popular
- penalty is only applied to next memory access
- Automatically handles collating writes

Write Miss

- Second scenario is we try to write to an address not contained in the cache
- When we update main memory, should we copy into cache?

Write Around

- Writes go directly into memory
- goes around cache.

- Good if data is used once.

Allocate on Write

- Loads new data into cache
- If the data is needed again it will be avalible.

Unified vs Split Cache

- Split Cache
  - One cache for instructions
  - one cache for data
  - Allows for accessing both instructions and data at the same time
- Unified Cache
  - Single cache with both program and code
  - More flexible
  - Lower miss rate
  - Allows for more entries and does not enforce as strict a structure.

Measuring Cache Performance

- If assume reasonable buffer depth and significantly faster buffer to lower memory, stalls are negligible
- Most write through implementations read and write stalls into the same miss rate and penalty.
- Memory stall cycles = Accesses * miss Rate * Miss Penalty

Average Memory Access Times

- Large caches have large hit times
- Average memory access times accounts for hit time.
- AMAT = Hit Time + Miss rate * Miss Penalty

Associative Caches

- Fully Associative
  - Allow a given block to map to any entry
  - Requires all entries to be searched at once
  - Comparator per entry is expensive
- N-way set associative
  - Each set contains n entries
  - Block number determines set
  - Search all entries in a set at once
  - n comparators is less expensive (scales with n)
- Side notes:
  - 1-way associative cache is the same as direct mapped cache.
  - If n is equal to the number of entries an n-way associative cache is equivalent to a fully associative cache.

How much Associativeity

- Increased associativity increases hit rate but has diminishing returns

Replacement Policy

- Direct mapped has no options
- Set associative

  - prefer non-valid entry
  - if all entries are valid choose using some policy
- Least Recently Used Policy (LRU)
  Chose the least used value
  Probably some linear feedback shift register