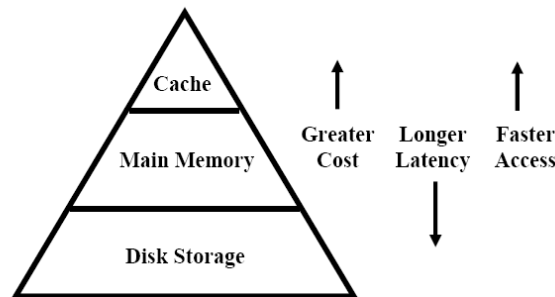


EECS 370

Discussion 8

Week of 3/20 – 3/26

- I) Cache basics
 - a. Memory hierarchy



- b. Cache holds data that we think is likely to be used (and used often!)
 - c. Architectural view of memory: program sees memory as a huge array of storage, NOT separate components of cache, main memory, etc.
 - d. ASIDE: Layers of abstractions
- II) Cache design
 - a. Cache memory copies data from main memory
 - b. Two parts:
 - i. Tag – holds memory address
 - ii. Block – holds memory data
 - c. When a program accesses the cache, and is referencing an address, it is compared with the tag
 - i. Match? Grab the data (cache HIT)
 - ii. Doesn't match? Go to main memory (cache MISS)
 - d. What if we write to memory? May write to cache, but need to update main memory
 - i. Write-through: all writes to caches propagate to main memory
 - ii. Write-back: only propagate to main memory before being evicted
 - 1. Dirty bit
 - 2. Advantages? Disadvantages?
- III) Cache allocation
 - a. Temporal locality: most recently accessed memory is more likely to be accessed again
 - b. Spatial locality: memory stored near recently accessed memory is more likely to be accessed than memory far away
 - c. Knowing this: **evict** least recently referenced data

- d. ASIDE: Thrashing
- IV) Average latency
 - a. Equal to cache latency x hit rate + memory latency x miss rate
 - b. Improve average latency by improving hit rate, or cache/memory latency
- V) Examples

Consider a program in which 20% of the instructions are memory loads/stores. Assume a main memory access time of 50 nsec, a clock rate of 500 MHz, and a CPI of 2 for a machine with a perfect cache (i.e. data and instructions are always found in first-level cache). What is the CPI if the instruction cache miss rate is 3% and the data cache miss rate is 10%?

50nsec * 500 MHz = 25 cycle stall for memory miss

$$CPI = 2 + (1 * 0.03 * 25) + (0.2 * 0.1 * 25) = 3.25$$

Given is a 4 byte cache with 2 byte block size and a 16 byte memory.

What is the size of the tag? 3 bits

What is the size of the block offset? 1 bit

Compute the cache state after executing the instruction sequence assuming a LRU replacement policy and a write-back policy for stores. Also indicate which loads cause a cache hit or cache miss.

- lw M[5] (miss)
- lw M[8] (miss)
- lw M[9] (hit)
- lw M[4] (hit)
- sw M[5] (hit) (mark dirty)
- lw M[10] (miss)
- lw M[14] (miss) (write-back M[4,5])
- sw M[11] (hit) (mark dirty)
- lw M[8] (miss)
- lw M[6] (miss) (write-back M[10,11])

In caching, "thrashing" is when two pieces of data repeatedly replace each other in the cache without ever successfully using it.

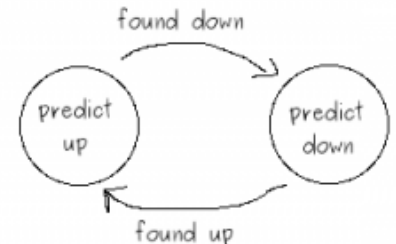


We should just predict that the seat should be in the position we found it in. That way only one person needs to change the seat.

```

addi $t0, t0, 5
loop: beq $t0, $0, done
lw $t1, 0x4($0)
lw $t2, 0x24($0)
addi $t0, $t0, -1
done:
  
```

Like a toilet seat. I always put it up and she always puts it down. We each change the seat every time but never find it the way we want it.



1-bit Branch Prediction