Measuring Cache Improvements

4-11-2003

Opening Discussion

- What did we talk about last class?
- Have you seen anything interesting in the news?
- Memory optimizations for processors. Last time I asked you to explain why memory system optimizations are considered valid reasons for chip upgrades. Would anyone like to explain this?

General Idea

- We want to be able to measure how much time the processor spends waiting on memory and how the impacts the CPI.
- If we use write-back or a good write buffer with write-through, then reads and writes have about the same penalty on misses and we can consider general cache misses, not special types. We might have lower miss rates for I-cache than D-cache.

Analysis of Cache Impact on CPI

- We can calculate the CPI as the normal CPI plus the average number of clock cycles we wait for memory reads.
- Every instruction comes from memory plus what we do in loads and stores.
  - Assume there is no delay for cache access while memory access takes 50 clock cycles.
  - For a realistic cache, assume it has a 5% miss rate. Let's look at no cache, realistic cache, and perfect cache scenarios.

Increasing Processor Speed

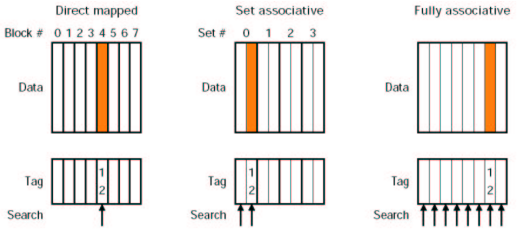
- We can mimic this simply by reducing the CPI in the earlier examples. In reality we increase the number of cycles in a miss penalty. Since the cycles are shorter we still get a speed boost, but by a smaller factor than the clock rate speed boost.
- Let's repeat out "realistic cache" scenario, but now with a 100 clock cycle memory delay.

Flexible Block Placement

- We can potentially decrease the miss rate if we are more flexibly in where we allow blocks to be placed instead of using the direct mapped method.
- Fully associative caches allow a block to be placed anywhere.
- Set associative is between these extremes. Here a block can go into some set of cache blocks. If it can go in *n* places it is called an *n*-way set associative cache. An address maps to a set and the set is searched for a match.

Example of Options

- Comparison of mapping options and where we have to search.

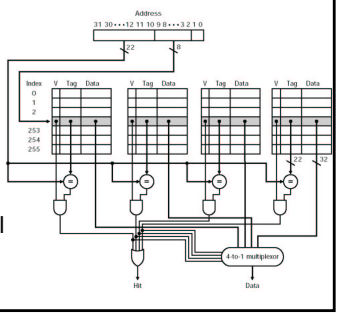


Cache Associatively

- Higher associatively generally results in larger hit times. We'd like to keep those small so that hits don't have to stall the pipeline.
- Since we have a choice of which element in a set to replace when we resolve a miss, we typically pick the block that was used least recently. This works with temporal locality.

Locating Blocks

- We still break an address into a tag, and index, and a block offset. The index tells us what set to look in. We then compare the tag to all the tags in the set in parallel to find the right one.



Multilevel Caches

- The first (L1) cache is on the chip with the processor and goes very fast, but can't be too large. To help reduce the cost of misses many computers have an L2 cache (now often on chip too) that takes longer than L1 but is faster than main memory.
- Even if the L2 only prevents a few percent of the pulls from main memory it can be a very significant improvement.

Minute Essay

- What did we talk about today?
- Remember to turn in assignment #6 to me.