Improving Cache Performance

There are three basic approaches to improving cache performance.

- Reducing the miss penalty [H&P §5.4]
  - Hybrid-access caches (victim, MRU)
  - Multilevel caches
  - Write buffers
  - Early restart
  - Critical word first
  - Subblocking

- Reducing the miss ratio [H&P §5.5]
  - Block size, cache size, associativity
  - Prefetching: Hardware, Software
  - Data layout: Instructions, Data

- Reducing the hit time [H&P §5.7]
  - Avoiding address translation (TLB accesses in parallel)
  - Using simple caches, small caches
  - Pipelining writes

Topics listed in dark blue have already been covered in Lectures 5 and 6.

Miss-penalty reduction

Let’s take a look at some other techniques for reducing the miss penalty.

Write buffers

This technique is used with write-through or write-back.

The idea is not to make the CPU wait for the write to complete in memory.

Instead, data is written to a write buffer, and the processor can continue while it is being written to memory.
How must handling of a read miss change when a write buffer is in use?

Going one step further, we can use a merging write buffer. In this case, a buffer entry will be several words long, like a cache line.

Writes to multiple words in this line can share space in the same buffer entry, and be written to memory at the same time. (See H&P, p. 421 for details).

**Early restart and critical-word first**

*Early restart:* It’s not necessary to wait until the cache block is completely read before restarting the processor.

As soon as the requested word arrives, it can be forwarded to the CPU.

Miss penalty is now time to fetch the requested word, not the entire block.
Critical-word first: Early restart takes care of restarting the processor when the requested word arrives. But that may still be quite a long time. Why?

To handle this case better, we can start fetching the block with the required (critical) word, and fill in the rest later.

Example: Suppose a computer has a 128-byte cache block (containing sixteen 64-bit words). It takes 12 clock cycles to get the critical word (or the first word, if critical-word first is not in use), and then two clock cycles per word to fetch the rest of the block.

- How long does it take to read a whole block, without critical-word first?
- How long does it take to read a whole block, assuming critical-word first?

With critical word first, it takes ____ cycles to get the critical word.

Without critical word first,

To get the rest of the block,

- with critical-word first,
- without critical-word first,

Subblocking

Problem: Tags are overhead; they take up extra space.

Partial Solution: Large blocks reduce the amount of tag storage.

- Say we keep cache size fixed, but double block size
• Then the number of blocks is halved
• The number of tags is also halved (# tags == # blocks)
• So, we’ve reduced amount of tag overhead, conserving space on the chip.
• but large blocks increase the cache miss penalty

Complete Solution: Use large blocks, but also divide blocks into smaller “subblocks.”

• Fetch only 1 subblock on a miss
• Keep valid bits for subblocks
• Better if other subblocks are prefetched in the background (that is, combine subblocking with early restart and critical word first).

We have seen two examples of subblocking so far in this course. What are they?