Slide 1

Administrivia

- Reminder: Quiz 5 Tuesday. Topics from Chapter 4. Might be much-shorter versions of homework problems, or might be more conceptual.
- Homework 5 on the Web. Due last day of class.

Slide 2

Memory Hierarchy — Overview

- Significant overlap between Chapter 5 and material covered in operating-systems course (as I teach it anyway).
- So, executive-level summary now.
- A key idea (borrowed from one writer of o/s textbooks): In a perfect world, we could have as much memory as we wanted, and it would be very fast and very cheap. In the real world, there are tradeoffs (e.g., fast versus cheap, fast versus large).

Slide 3

"Principle of Locality"

- Basic underlying idea — most applications exhibit locality with regard to memory.
- "Temporal locality" — memory locations referenced in the near past are likely to be referenced again in the near future.
- "Spatial locality" — memory locations close together in space likely to be referenced close together in time.

Slide 4

Memory Hierachy and Caching

- To exploit temporal locality, can use "caching" — keep copies of frequently-used data in faster but smaller memory. Can do this on multiple levels.
- To exploit spatial locality, can move data between levels in blocks.
- Terminology — cache hit, cache miss, cache block/line.
- *Notice* that while impact of caching on performance can be significant, it should not affect results (which is why it makes some sense to just ignore it initially).

Slide 5

Caches (Between Processor and RAM) — Executive-Level Summary

- To make these work, we need:

  Some way to map memory address to cache location — can be simple ("direct map") or not.

  Some way to say, for each cache location, what memory address it's currently associated with, and whether the data is valid.

- Read "from memory" tries cache first, and then if not found there goes to RAM and updates cache.
- Write "to memory" is maybe more interesting — writes to cache but then must at some point write to RAM also — maybe right away (easier to get right but can be slow) or later.

Slide 6

Virtual Memory — Executive-Level Summary

- Basic idea here is to fake having more RAM than you really have, by keeping some data that would be in RAM on disk. In a sense, RAM is a cache for the "real" memory, on disk(!).
- Also provides a nice way to support multitasking — notion of "processes", each with its own "address space", with an operating system that maps this abstraction onto the hardware, by mapping "program addresses" (in a process's address space) to "physical addresses" (in RAM). *Lots* of details here, but the basic idea is fairly simple. One big advantage is more control over what data each process can access.

Slide 7

Cache Coherence — Executive-Level Summary

- Clearly(?) possible for cached data to be out of synch with data in memory.
- First step toward managing possible impact is to clearly define what we want, and then figure out how to either solve the possible problems or work around them.
- Clearly(?) much more difficult with multiple processing elements each of which has its own cache.

Slide 8

Caches and Applications Programming

- Mostly the memory hierarchy (including virtual memory) is managed transparently by a combination of hardware and (operating-system) software, so the first approximation presented in introductory courses (memory is essentially a really big array of bytes, with addresses as indices) is okay, especially if you just want right answers.
- However, effects on performance can be significant, so if you want right answers fast . . .

  For single-threaded programs, key idea is to maximize locality (temporal and spatial). Rearranging order in which data is accessed can have a big effect.

  For multi-threaded programs, also need to consider whether multiple threads need to share access to the same data (problem for correctness too!) or even nearby data (“false sharing” — no effect on correctness but can be slow).

Slide 9

Minute Essay

- How much of today's discussion was familiar?