Slide 1

Administrivia

- Reminder: Homework 8 due this week.
- Not-accepted-past deadline for all homeworks May 12.

  Also, if you missed an online class, you can get the attendance point(s) for it by watching the recorded class and sending me answers to the minute essay — if you do so by that same deadline.
- I can post some extra-credit problems if there is interest. (I'll ask in the minute essay.) Can only help your grade; try as many as you like. Note that for these you'd be asked to work individually.

  Also keep in mind that you can do Homework 5 (entirely optional) and the optional problems for Homework 6 for extra credit. Homework 5 is (I think!) tedious but not difficult.

Slide 2

More Administrivia

- I'm planning to do virtual office hours a few times during reading days and finals. Times TBA. I'll ask about preferences in the minute essay and let you know my plans by e-mail.
- Also keep in mind that while even Zoom meetings can be a bit problematical for me, I'm generally very willing to try to help by e-mail!

Slide 3

More Administrivia

- Sample solutions to Homeworks 1 through 7 also in shared folder, and there will be one for Homework 8 early next week.
- Quiz solutios linked from "Lecture topics etc." page.
- Exam 1 graded and graded papers scanned in and uploaded to Google Drive (at last!), in your individual folders. Sample solution in shared folder.
- Grade summaries also uploaded. As of 4/27, includes quizzes up through Quiz 5 but nothing else after the exam. I will do this again as I get more graded.

  You might want to check that I've graded everything you think you've turned in!

Slide 4

More Administrivia

- I'm still not entirely sure what to do about late penalties, given the situation. My idea is to apply them to work that was due before spring break, but greatly reduce or even eliminate them for work due after that. So:
- *If you turn in work late,* and you feel that your situation is such that you couldn't reasonably turn it in on time, please explain when you turn it in, and I will take that into consideation.
- *If you can't meet the final deadline of May 12 but think you could turn in work that week,* please let me know. I don't want to just extend that deadline for everyone since I do have a deadline for turning in grades, but I can make a few exceptions.
- Questions?

Slide 5

Digression — Drawing Figures Programmatically

- As I was preparing a sample solution for Homework 6 in a previous year, I got interested in whether there wasn't some nice tool to do this programmatically — rather than me drawing a bunch of gates with a drawing program and connecting them, well, it just seemed like something a computer could help a lot with, and similarly with the state machines.
- Being a LaTeX fanatic, I looked for LaTeX-based approaches, and found . . .

Slide 6

Digression — Drawing Figures Programmatically

- . . . something called TikZ (short for German for "TikZ is not a drawing program"). There's quite a learning curve, but the results can be really nice. Examples on "sample programs" page.

  (I got carried away and spent part of that summer drawing some of the figures in Chapter 4 with it! And I *think* it really is easier for me now to produce nice-looking diagrams like the ones in Appendix B.)
- Take-home message, maybe: LaTeX is really good in general at converting "logical markup" into something more graphical. That this can apply to turning a logical(?) representation of a figure into something graphical — maybe surprising, maybe not? Other tools could work the same way (and maybe some do)?

Slide 7

Parallel Computing — Recap/Review

- Two classic hardware/software models:

  Shared memory, programmed using threads and shared variables.

  Distributed memory, programming using processes and message-passing.

- GPGPU doesn't fit neatly into either one; may be a third model.

Slide 8

Shared-Memory Hardware, Revisited

- Figure 6.7 sketches basic idea: Multiple processing elements (processors, cores, whatever) connected to a single memory.

- Synchronization (locking) *can* be done with no hardware support, but it's tricky. Simple approach is something such as:

  ```
  while (lock != 0) {};
  lock = 1;
  ```

  which doesn't work because test and set are separate instructions!

- Somewhat-tricky algorithms exist for solving this problem in software, but . . .

Slide 9

Shared-Memory Hardware — Locking

- Locking is much easier if ISA provides some support, in the form of an instruction that allows . . . Well, essentially allows both read and write access to a location *as a single atomic operation*.
- Some architectures implement this directly, via a "compare and swap" or "test and set" instruction. But for MIPS that might be challenging (why?).
- So MIPS defines two instructions, "load linked" (`ll`) and "store conditional" (`sc`). Tricky, but textbook has an example (p. 122), which we looked at earlier.

Slide 10

Shared-Memory Hardware — Memory

- Access to RAM can be reasonably straightforward — only one processor at a time. Caches complicate things (next slide).
- "Single memory" may actually be multiple memories, with each processing element having access to all memory, but faster access to one section ("NUMA" (Non-Uniform Memory Access)). Making good use of this can affect performance — and may be non-trivial to accomplish, especially if programming environment doesn't give you appropriate tools. (As best I can tell, most don't, sadly.)

Slide 11

Shared-Memory Hardware — Caches

- As noted, even if access to RAM is one-processor-at-a-time, if each processing element has its own cache, things may get tricky. Typically hardware provides some way to keep them all in synch (the “cache coherency” problem discussed in Chapter 5).
- Further, application programs may have to deal with “false sharing” — multiple threads access distinct data in the same “cache line”. Cache coherency guarantees correctness of result, but performance may well be affected. (Example — multithreaded program where each thread computes a partial sum. Having the partial sums as “thread-local” variables can be much faster than having a shared array of partial sums.)

Slide 12

Distributed-Memory Hardware, Revisited

- Figure 6.13 sketches basic idea: Multiple systems (processor(s) plus memory) communicating over a network.
- No special hardware required, though really high-end systems may provide a fast special-purpose network.

Slide 13

SIMD Hardware

- Various ways to implement this idea in hardware.
- One approach: Multiple processing elements sharing access to memory and all executing the same instruction stream,

  This is more or less how GPUs work. A complication: They often have a separate memory, so data must be copied to/from RAM. Potential performance problem, may be cumbersome for programmers.
- Another approach: "Vector processing units" that stream/pipeline operation on data elements to get the data-parallelism effect.

Slide 14

Other Hardware Support for Parallelism

- Instruction-level parallelism (discussed in not-assigned section(s) of Chapter 4) allows executing instructions from a single instruction stream at the same time, if it's safe to do so. Requires hardware and compiler to cooperate, and (sometimes?) involves duplicating parts of hardware (functional units).
- Hardware multithreading (discussed in Chapter 6) includes several strategies for speeding up execution of multiple threads by duplicating parts of processing element (as opposed to duplicating full PE, as happens with "cores").

Slide 15

Exam Review

- (Topic by topic through review sheet.)

Slide 16

Course Recap — Topics

- A little about performance. (It's not simple!)
- MIPS assembler language; translating C to MIPS assembler language.
- Compiling, assembling, and linking.
- Binary representation of instructions.
- Binary representation of data (integers, ASCII, floating-point numbers); basics of computer arithmetic.

Slide 17

Course Recap — Topics, Continued

- Gate-level logic design.
- Design of a processor — ALU, datapath, control; a little about pipelining.
- A little about caches, and a very little about virtual machines and support for parallelism.
- Other schools spread this material over two or even three courses (though they presumably cover more in all). So, we have done a lot?

Slide 18

Course Recap, Continued

- Some topics — representation of data, computer arithmetic, maybe finite state machines — are review, or small extension of what you know.
- Others, though — assembler language, gate-level logic, designing a processor in terms of AND/OR/NOT and how it works — are not familiar to most, and involve a new perspective, or mindset, or “mental model”. My observation — some students take to it, others struggle.
- I do hope, however, that all of you have come away with more understanding of how things work “under the hood” than you had!

Slide 19

Minute Essay

- What times of day work best for you for office hours? Any particular days?
- Would you be interested in extra-credit problems?
- I'd be interested in one more round of comments on (1) how this remote-learning thing has been for you, and (2) how you've all been coping in general.
- Course evaluations:

  The online equivalent of those little slips of paper is on Google Drive. Each of you should have a file `course-eval.png`. (A "thank you" to the folks in the ASO for helping make this happen!)

  I prefer that you do these now but I can't insist.
- Best of luck finishing the semester, and best wishes for a pleasant summer break!