Slide 1

Administrivia

- Reminder: Quiz 6 Wednesday.
- Homework 6 on the Web; due Friday. No more required homeworks, though there will likely be some optional extra-credit problems.

Slide 2

Minute Essay From Last Lecture

- (Several answers having to do with privileged/non-privileged users. Important points made, but not really the point?)
- The point I was trying to make: If the processor has two modes of execution, one privileged and one not, how do you get from one to another? Going from privileged to non-privileged seems harmless, but the other way?

Slide 3

Exceptions — Review/Recap

- Several situations (e.g., errors and external interrupts) in which it would be useful to interrupt normal flow of control and — do something.
- MIPS architecture, like many, has a notion of "exceptions" (a.k.a. "interrupts") that cause the processor to transfer control to a fixed location, with a saved PC value and some indication of the cause of the exception.
- Figure 4.66 shows what has to be added to the pipelined design to make this work. Pipelining makes implementing this idea more complicated in some ways (must consider what to do about all instructions in the pipeline), possibly simpler in others (there is already support for dealing with "control hazards", similar in some ways).

Slide 4

Parallel Computing — Overview

- (In the time we have left — a little about parallel computing and hardware to support it.)
- Support for "things happening at the same time" goes back to early mainframe days, in the sense of having more than one program loaded into memory and available to be worked on. If only one processor, "at the same time" actually means "interleaved in some way that's a good fake". (Why? To "hide latency".)
- Support for actual parallelism goes back almost as far, though mostly of interest to those needing maximum performance for large problems. Somewhat controversial, and for many years "wait for Moore's law to provide a faster processor" worked well enough. Now, however . . .

Slide 5

Parallel Computing Overview, Continued

- Improvements in "processing elements" (processors, cores, etc.) seem to have stalled. Instead hardware designers are coming up with ways to provide more processing elements.
- One result is that multiple applications can execute really at the same time.
- Another result is that individual applications *could* run faster by using multiple processing elements.

  Non-technical analogy: If the job is too big for one person, you hire a team. But making this effective involves some challenges (how to split up the work, how to coordinate).
- In a perfect world, maybe compilers could be made smart enough to convert programs written for a single processing element to ones that can take advantage of multiple PEs. Some progress has been made, but goal is elusive.

Slide 6

Parallel Computing — Hardware Platforms

- Clusters — multiple processor/memory systems connected by some sort of interconnection (could be ordinary network or fast special-purpose hardware). Examples go back many years.
- Multiprocessor systems — single system with multiple processors sharing access to a single memory. Examples also go back many years.
- Multicore processors — single "processor" with multiple independent PEs sharing access to a single memory. Relatively new. Hardware multithreading is maybe a special case?
- "SIMD" platforms — hardware that executes a single stream of instructions but operates on multiple pieces of data at the same time. Popular early on (vector processors, early Connection Machines) and now being revived (GPUs used for general-purpose computing).

Slide 7

Minute Essay

- What experience, if any, do you have with programs that make use of multiple threads, processes, etc.?