Slide 1

Administrivia

- Homework for chapter 2 on the Web, in two parts: Homework 2 due next Wednesday, Homework 3 due next Friday (tentative).

Slide 2

Sidebar: MIPS Simulator, Revisited

- Installed on the lab machines (under Linux at least) is a MIPS simulator. Allows you to try out your programs. (“Programming is not a spectator sport.”) Can also install on your own machine (or so they say!) versions for Linux and Windows. Linux version can be compiled for Mac OS X (they say).
- Graphical version is `xspim`, command-line version `spim`.
- Documentation on using them on companion CD (contents available at Web site for book).
- Supports some simple “system calls” for doing input/output. Examples of using them — sample programs `echo.s` and `echoint.s`.

Parallel Execution and Synchronization — Recap

- A lot of commodity hardware these days features multiple processing units ("cores") sharing access to memory. One reason for this is that in theory we can make individual applications faster by splitting computation up among processing elements.

  (Shameless self-promotion: We plan to try again to offer the parallel-programming elective in the fall.)

- Having processing elements share memory makes parallel programming easier in some ways but has risks ("race conditions"). Avoiding the risks requires some way to control access to shared variables (e.g., to implement notion of "lock").

Parallel Execution and Synchronization, Continued

- Most texts on operating systems discuss synchronization issues and present several solutions ("synchronization mechanisms"), some rather high-level and others not.

  (Why is this in o/s textbooks?)

- The most primitive can (with some simplifying assumptions) be implemented with no hardware support. But hardware support is very useful.

Slide 5

Instructions for Synchronization

- Key goal in designing hardware support for synchronization is to provide "atomic" (indivisible) load-and-store. This allows writing a low-level implementation of "lock" idea.
- Many architectures do this with a single instruction (e.g., "test and set" or "compare and swap"). Requires two accesses to memory so may be difficult to implement efficiently.
- MIPS approach — same idea, but using a pair of instructions, `ll` ("load linked") and `sc` ("store conditional"). Example of use in textbook.

Slide 6

Minute Essay

- We talked today about instructions to support multithreaded programming. What exposure, if any, do you have to this kind of programming?