Slide 1

Administrivia

- Reminder: Midterm in class Wednesday. See review sheet for format.
- Sample solutions to written problems distributed in hardcopy. Sample solutions to programming problems online (the last one is coming soon, today/tomorrow).
- Homework 3 graded. Good news is that most people came reasonably close on the last problem (not so in previous years). Not-so-good news is that this was not so on the first problem. A bit of review today.

Slide 2

Review — Processors and Code

- What processors do is repeatedly fetch an instruction from memory, figure it out, and execute it.
- Typically they also contain registers, some general-purpose (as a sort of on-chip "scratch pad") and some special-purpose (such as program counter).
- Ultimately, what processors run is machine language. Programs can be compiled into machine language directly, or they can be compiled into some other representation used as input by an interpreter or runtime system, or they can be interpreted directly.

Slide 3

Review — Process Abstraction

- Key point here is that a process represents a virtual processor — something that executes machine language and has whatever registers a real processor has.
- Associated with a process there might be other information too — something that identifies what memory the program has access to, a list of open files, etc.
- Distinction between (heavyweight) processes and threads is a little blurry, but in general threads implement only the “virtual CPU” part of the abstraction and are associated with an enclosing process, with other process-related resources being specific to the process and not to individual threads.

Slide 4

Review — Context Switches

- When a process is executing, its state (values in registers, etc.) is represented by the state of the processor.
- On interrupt, we want to be able to suspend execution and then later pick up where we left off. To make that happen, need to save/restore state of processor — “context switch”. The more such state has to be saved/restored, the longer context switches take.

Review — Synchronization

Slide 5

- Synchronization with shared memory only is possible but very tricky to get right (i.e., avoid race conditions). May involve processor-specific instructors such as TSL.
- Synchronization with a “synchronization mechanism” easier but requires one to be defined and implemented.

Review — Semaphores

Slide 6

- Semaphore abstraction is an ADT — a value and two operations (`up` and `down`).
- What constitutes an implementation? a description of needed variables plus (pseudo)code for the two operations. Notice that ensuring one-at-a-time access to shared variables may require use of some lower-level synchronization.

Review — Monitors

Slide 7

- Semaphores are all basically the same — instances of an ADT — but monitors are more like instances of object-oriented classes plus some synchronization.
- Describing a “monitor-based solution” for a problem is pretty much like defining a class in an OO language — say what its variables and methods (“procedures” in monitor-speak) are.
- Monitors often make use of condition variables — ADT with two operations (`wait` and `signal` and that’s all).

General Midterm Review

Slide 8

- (Topic by topic through review sheet.)

Minute Essay

- Anything noteworthy about Homework 3?