Slide 1

Administrivia

- Reminder: Homework 8 due today.
- Reminder: Exam 2 next Wednesday. Review sheet posted on course Web site.

  Solutions to quizzes linked from “lecture topics” etc. page.

  Solution to homeworks to be shared with you via Google Drive. I’m trusting you not to look at sample solutions if you haven’t yet turned in an assignment and plan to sometime!
- I’m finishing up grading of Homework 6 and hope to finish before I leave today. I’ll put them in my mailbox for you to pick up?

  I’ll try hard to grade by Monday anything turned in to me before Friday.

Slide 2

Exceptions — Review/Recap

- Useful as a way of dealing with errors.
- Some might be recoverable (e.g., arithmetic overflows).
- Others may not be (e.g., attempt to execute something that’s not an instruction).

Interrupts — Review/Recap

Slide 3

- Useful for interacting with I/O devices, where typically operations are slow compared to processor speeds, so it makes sense to start one and then let device interrupt whe it's done.
- Also useful for sharing processor among concurrently-executing threads or processes, where it often makes sense to "time-slice", i.e., allow each one to run for limited time before giving another a turn.

Exceptions and Interruptions — Recap/Review

Slide 4

- In both situations, what seems to make sense: Transfer control to operating system, which can decide what to do.
- Mechanism for doing this: Hardware saves PC of next instruction to execute and possibly something indicating type of exception/interrupt, then transfers control to fixed location. Can be the same location for all types, or different locations for different types.

Slide 5

Exceptions/Interruptions and Operating System Services — Background

- In a general-purpose system able to execute more than one program at a time, there are things these application programs should not be allowed to do for themselves. Instead there should be a central authority — an operating system.

  Examples include communicating with I/O devices, requesting memory, etc.
- To really make this work reliably, need to make sure *only* operating system can do these things. How? . . .

Slide 6

Operating System Services — Dual-Mode Operation

- Many processors have notion of two modes of operation: “privileged” one for doing O/S stuff, “unprivileged” one for regular applications.
- Special-purpose register (akin to PC) says which mode currently in effect.
- Attempts to do privileged operations while in unprivileged mode generate exceptions. Obviously(?) can’t just let application programs set this bit! How then can application programs request O/S services?

Slide 7

Requesting Operating System Services

- Typical solution:
- Have application program execute instruction that generates exception/interrupt, with something indicating which service requested. (In MIPS, `syscall`.)
- Have hardware set “privileged” bit when it transfer control to O/S. O/S performs service, clears “privileged” bit, returns to application.
- Same idea works for all exceptions/interrupts.

Slide 8

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 9

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 10

Minute Essay

- None — quiz.