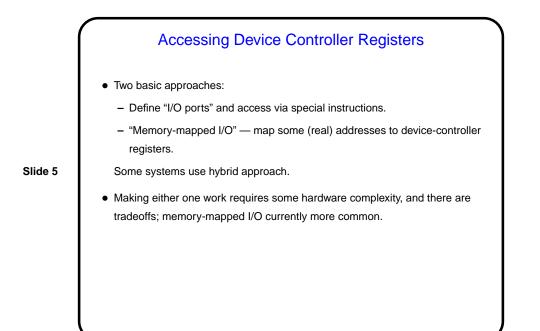
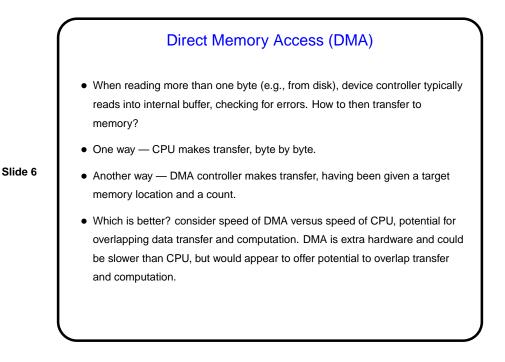
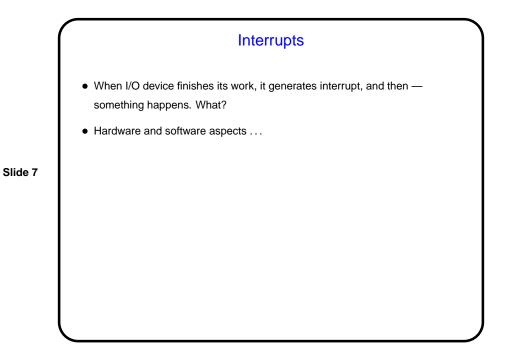
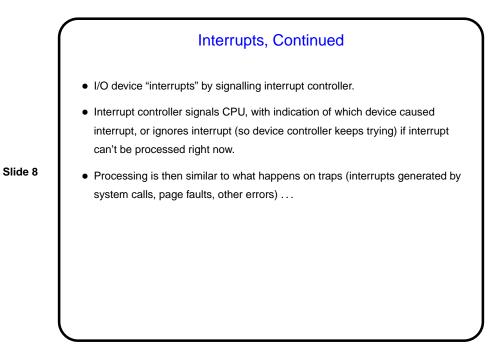


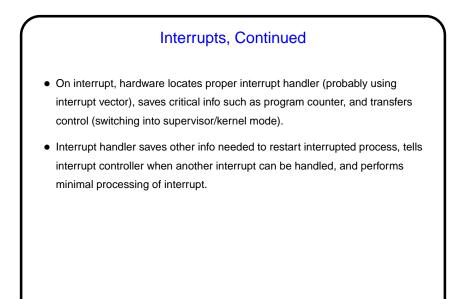
	I/O Hardware, Revisited
•	<ul> <li>First, a review of I/O hardware — simplified and somewhat abstract view, mostly focusing on how low-level programs communicate with it.</li> </ul>
•	<ul> <li>Many, many kinds of I/O devices — disks, tapes, mice, screens, etc., etc. Car be useful to try to classify as "block devices" versus "character devices".</li> </ul>
	<ul> <li>Many/most devices are connected to CPU via a "device controller" that manages low-level details — so o/s talks to controller, not directly to device.</li> </ul>
	<ul> <li>Interaction between CPU and controllers is via registers in controller (write to tell controller to do something, read to inquire about status), plus (sometimes data buffer.</li> </ul>
	Example — parallel port (connected to printers, etc.) has control register (example bit — linefeed), status register (example bit — busy), data register (one byte of data). These map onto the wires connecting the device to the CPU.











## Interrupts, Continued

- Worth noting that pipelining (very common in current processors) complicates interrupt handling when an interrupt happens, there could be multiple instructions in various stages of execution. What to do?
- "Precise interrupts" are those that happen logically between instructions. Can try to build hardware so that this happens always, or sometimes.
- "Imprecise interrupts" are the other kind. Hardware that generates these may provide some way for software to find out status of instructions that are partially complete. Tanenbaum says this complicates o/s writers' jobs.

