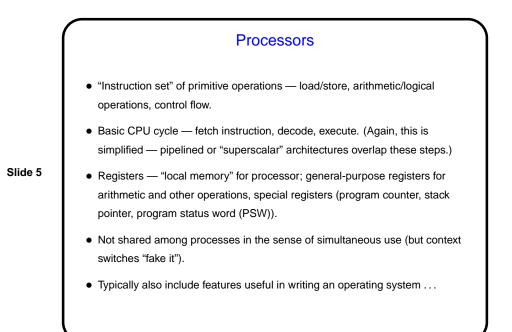
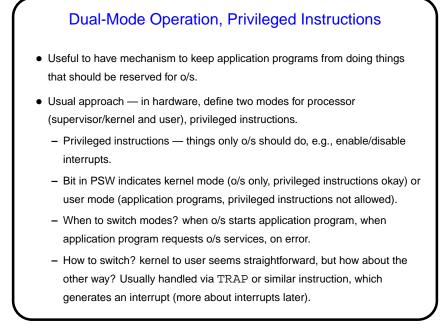


## Overview of Hardware Simplified view of hardware (as it appears to programmers) — processor(s), memory, I/O devices, bus. Figure on p. 19 shows simplified view of overall organization — components connected to a single bus. (Actual processors may have more than one bus.)



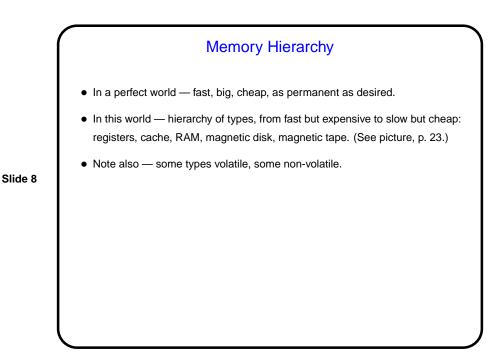


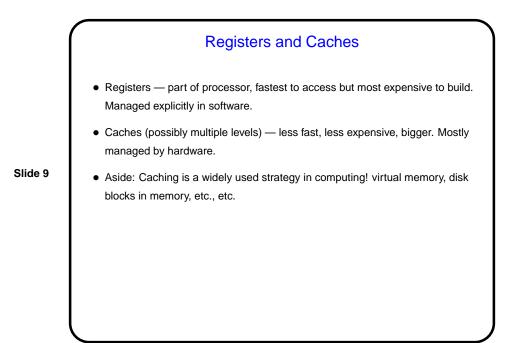
## Multithreaded and Multicore Chips

 For many years (at least 20, to my knowledge) advocates of parallel programming have been saying that eventually hardware designers would run out of ways to make single processors faster — and finally it seems to be happening.

Slide 7

 Basic idea — number of transistors one can put on a chip is still increasing, but how to use them to make single processors faster isn't clear. So, instead, hardware designers have chosen to provide (more) hardware support for parallelism. Various approaches, including "hyperthreading" (fast switching among threads), "multicore" (multiple independent CPUs, possibly sharing cache), "GPGPU" (use of graphic card's many processors for computation).



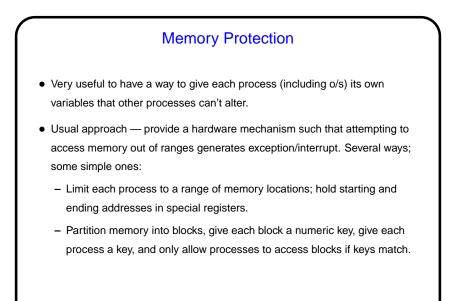


 Main Memory (RAM)

 • Still less fast, less expensive, bigger.

 • Shared among processes — which presents some interesting challenges …

 Slide 10



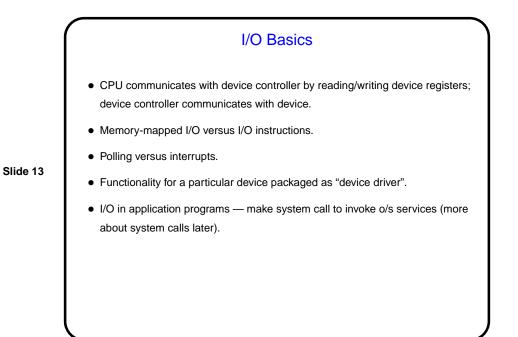
I/O Devices
What they provide (from the user's perspective):

Non-volatile storage (disks, tapes).
Connections to outside world (keyboards, microphones, screens, etc., etc.).

Slide 12

Distance between hardware and "virtual machine" is large here, so usually think in terms of:

Layers of s/w abstraction (as with other parts of o/s).
Layers of h/w abstraction too: most devices attached via controller, which provides a h/w layer of abstraction (e.g., "IDE controller").



Minute Essay
I once had a learning experience about "how DOS is different from a real o/s". Summary version: A program using pointers (possibly uninitialized) caused the whole machine to lock up, so thoroughly that the only recovery was to power-cycle.
What do you think went wrong?

## Minute Essay Answer

The program changed memory at the addresses pointed to by the uninitialized pointers — and this memory was being used by the o/s, possibly to store something related to interrupt handling. A "real" o/s wouldn't allow this!
 (Then again, the version of MS-DOS in question was supposedly written to run on hardware that didn't provide memory protection, so maybe it's not DOS's fault.)