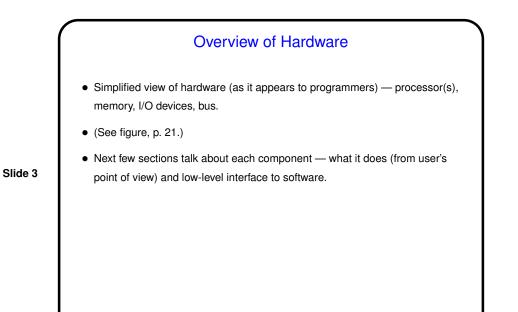
A reminder: *Please do not* reboot the machines in HAS 340! People depend on these machines to do background processing.
If a previous user has left a machine in the "locked by screensaver" state, you can bail out by pressing control-alt-backspace to restart X (the graphical subsystem) without disturbing background processes.
If you log out from the "System" menu, it might be easy to shut down by mistake. Can put an icon on the task bar for logout to avoid this.
Prox card access should be enabled now, so you should be able to get into the labs after hours.

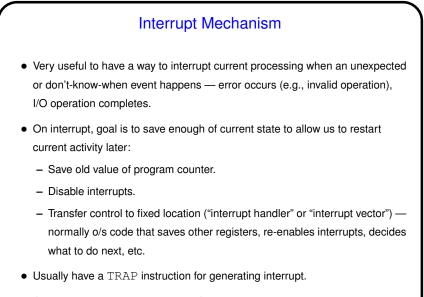
Slide 1

Operating System Functionality Provide a "virtual machine": Filesystem abstraction — files, directories, ownership, access rights, etc. Process abstraction — "process" is a name for one of a collection of "things happening at the same time" (in effect if not in fact), including: In batch systems, user "jobs", plus input/output spooling. In timesharing system, interactive users. In PC o/s, concurrently-executing tasks. Here too, idea of ownership / access rights. Manage resources (probably on behalf of multiple users/applications): Memory. CPU cycles (one or more CPUs).

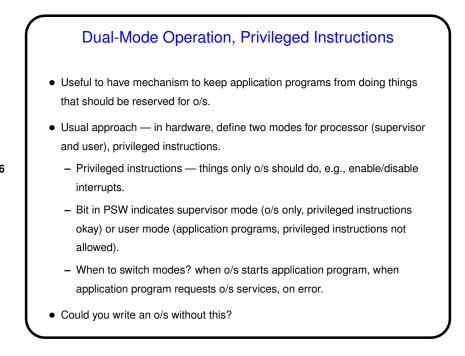


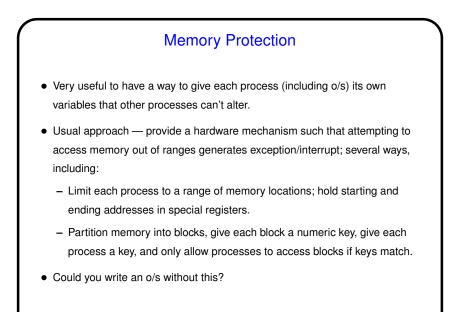
Processors "Instruction set" of primitive operations — load/store, arithmetic/logical operations, control flow. Basic CPU cycle — fetch instruction, decode, execute. Registers — "local memory" for processor; general-purpose registers for arithmetic and other operations, special registers (program counter, stack pointer, program status word (PSW)). Now consider what additional features would make it easier to write an operating system ...

Slide 5



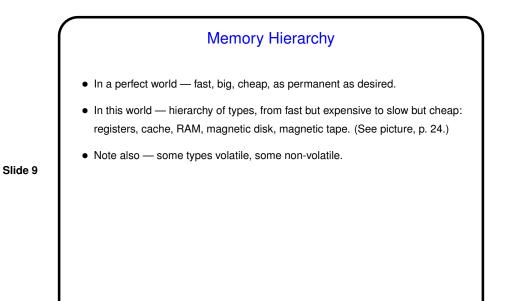
• Could you write an o/s without this?



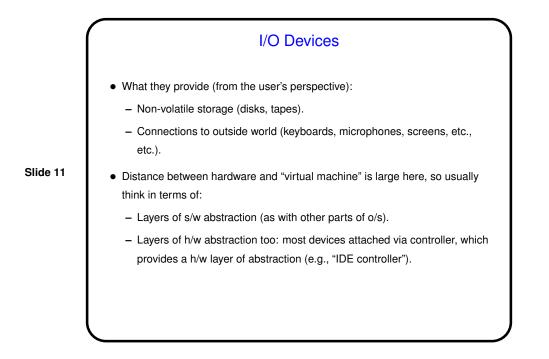


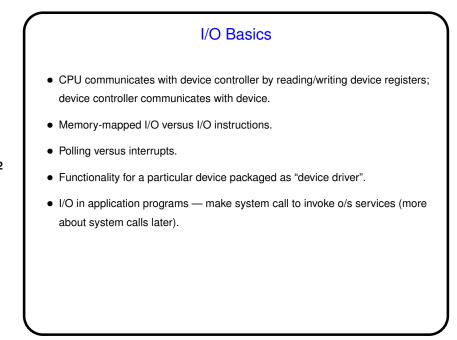
Useful to have a way to set a timer / "alarm clock" — e.g., to get control back if application program enters infinite loop.
Usual approach — hardware features that tracks real time and can be set to interrupt CPU.
Could you write an o/s without this?

Slide 7



Program Relocation
At the machine-instruction level, references to memory are in terms of an absolute number. Compilers/assemblers can generate these only by making assumption about where program will reside in memory.
In the very early days, programs started at 0, so no problem. Now they hardly ever do, so we need a way to relocate programs — when loaded, or "on the fly".
"On the fly" relocation uses MMU (memory management unit) — which can provide both program relocation and memory protection.
Logically between CPU and memory, physically usually part of CPU.
A simple scheme — base and limit registers (described in text). When do values in them need to change?





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?

Slide 13

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.)