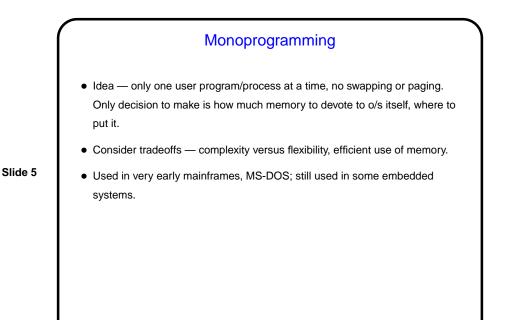
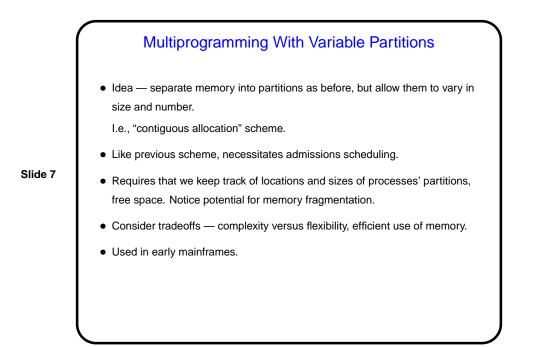


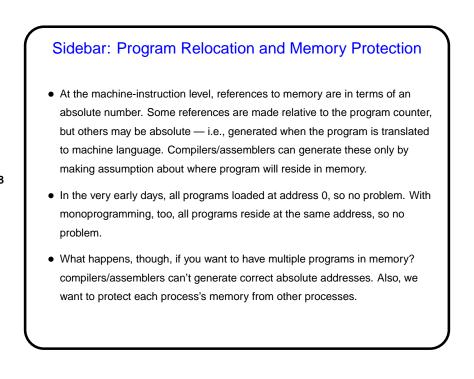
Simple Schemes — No Abstraction
Memory (a.k.a. "RAM") can be thought of as a very long list of numbered cells (usually bytes). (This is a somewhat simplified view but "good enough" for our purposes.)
Simplest schemes for managing it don't try to hide that view. (Names for these come from older edition of Tanenbaum's book.)

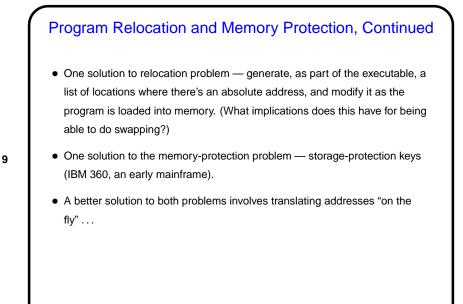
Slide 3



Multiprogramming With Fixed Partitions
Idea — partition memory into fixed-size "partitions" (maybe different sizes), one for each process. Possibly also add the ability to "swap" programs (write their memory to disk, read back in later).
Limits "degree of multiprogramming" (how many processes can run concurrently).
Probably necessitates admissions scheduling — either one input queue per partition, or one combined queue. If one combined queue, how to choose from it when a partition becomes available? first job that fits? largest job that fits? etc.
Consider tradeoffs — complexity versus flexibility, efficient use of memory.
Used in early mainframes.



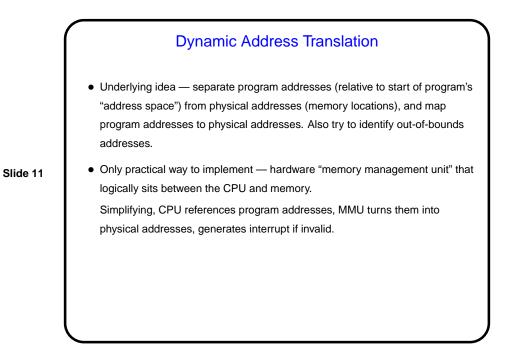


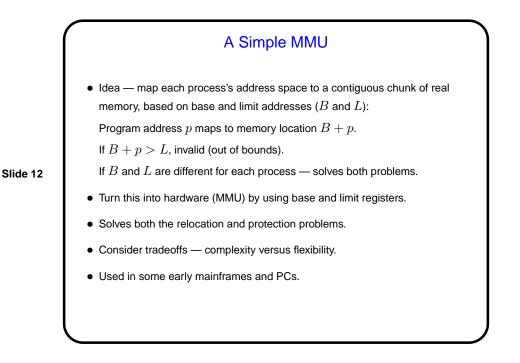


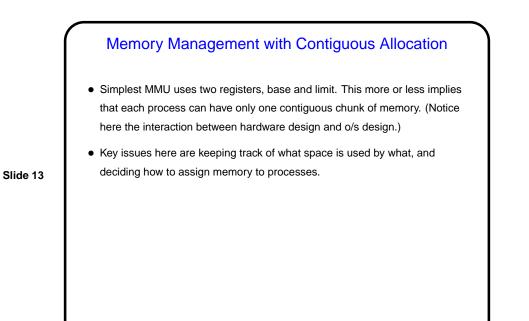
Slide 9

Sidebar: The "Address Space" Abstraction

- Basic idea is somewhat analogous to process abstraction, in which each process has its own simulated CPU. Here, each process has its own simulated memory.
- As with processes, implementing this abstraction is part of what an operating system can/should do.
- Usually, though, o/s needs help from hardware ...
- Slide 10







Multiprogramming With Variable Partitions, Continued
Another implementation issue — how to decide, when starting a process, which of the available free chunks to assign.
Several strategies possible:

First fit.
Next fit.
Best fit.
Worst fit.
Quick fit.

