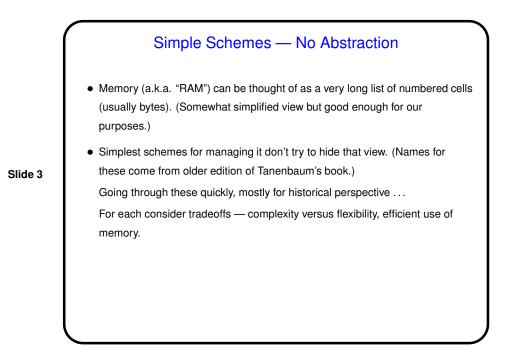
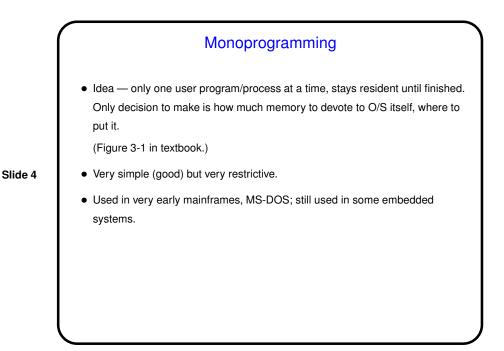


Memory Management — Overview

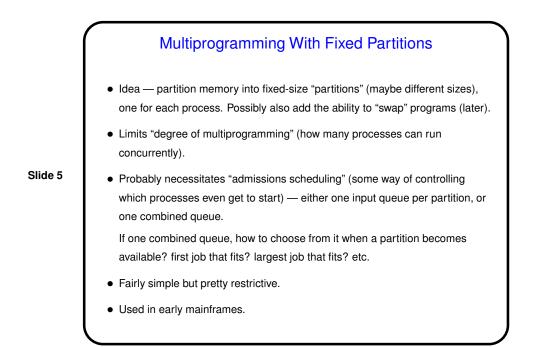
- One job of operating system is to "manage memory" assign sections of main memory to processes, keep track of who has what, protect processes' memory from other processes.
- As with CPU scheduling, we'll look at several schemes, starting with the very simple. For each scheme, think about how well it solves the problem, how it compares to others.
- As with processes, tradeoff between simplicity and providing a nice abstraction to user programs.







 $\mathbf{2}$





• Idea — separate memory into partitions as before, but allow them to vary in size and number. (Figure 3-4 in textbook, sort of.)

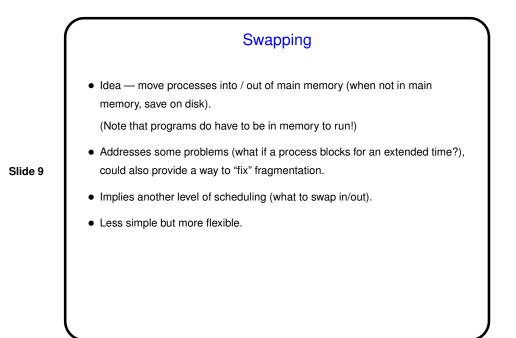
I.e., "contiguous allocation" scheme.

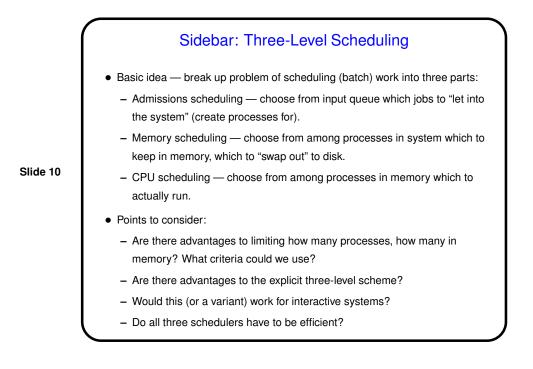
- Like previous scheme, necessitates admissions scheduling.
- Requires that we keep track of locations and sizes of processes' partitions, free space. Note potential for memory fragmentation.
- Also fairly simple but restrictive.
- Used in early mainframes.



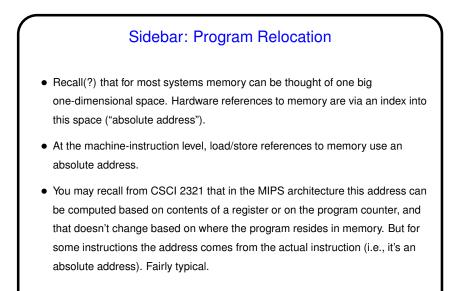
Multiprogramming with Fixed/Variable Partitions — Recap

- Comparing the two schemes:
 - Similar admission scheduling issues.
 - Both pretty simple and pretty restrictive, though variable partitions are less so. Neither makes great use of memory.
- Either could be adequate for a simple batch system, maybe with the addition of swapping.

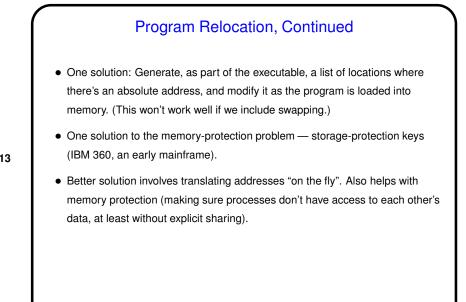




5

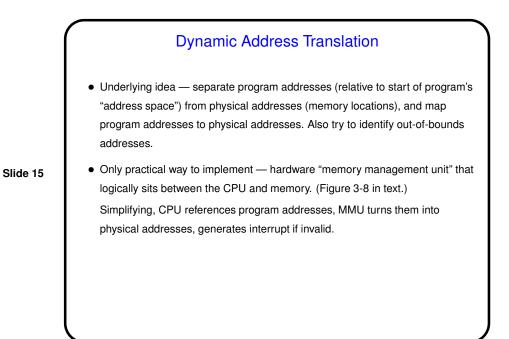


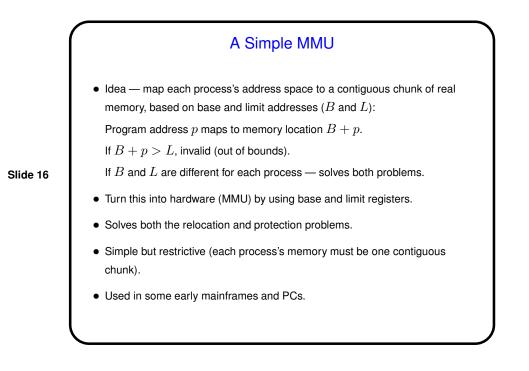
Program Relocation, Continued You may also recall from the discussion of assembling and linking that generating these absolute addresses is a bit complicated, since they can't be known at least until link time. But even then, they depend on where the program will reside in memory. Slide 12 In the very early days, all programs loaded at address 0, so no problem. With monoprogramming, too, all programs reside at the same address, so no problem. (SPIM works that way.) What happens, though, if you want to have multiple programs in memory? compilers/assemblers can't generate correct absolute addresses. This is the "relocation problem". What to do?

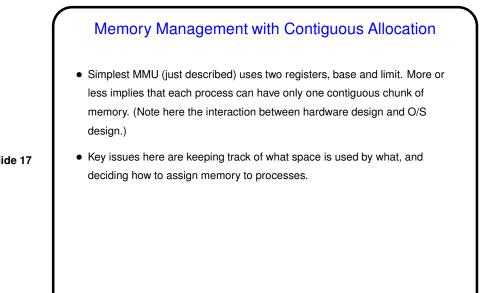


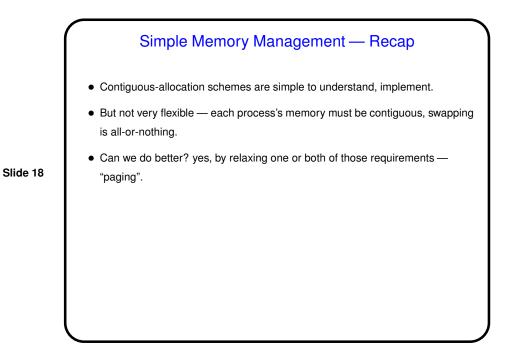
Sidebar: The "Address Space" Abstraction

- Basic idea 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 ...

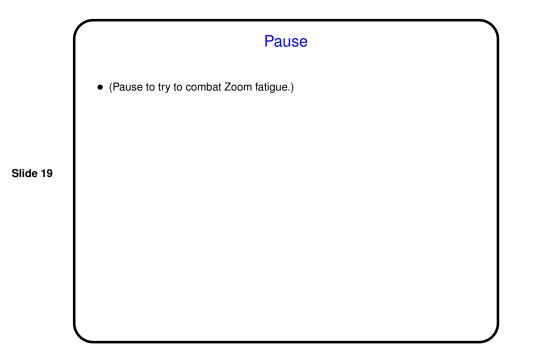




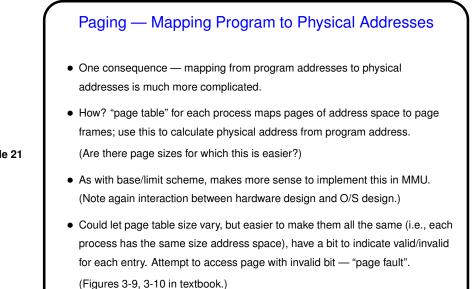




9

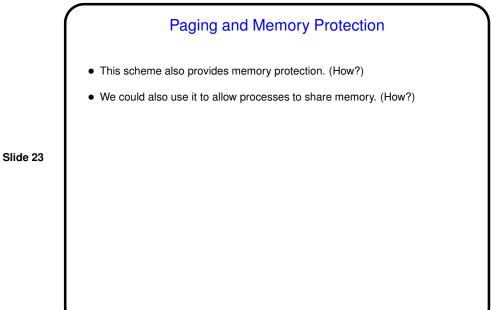


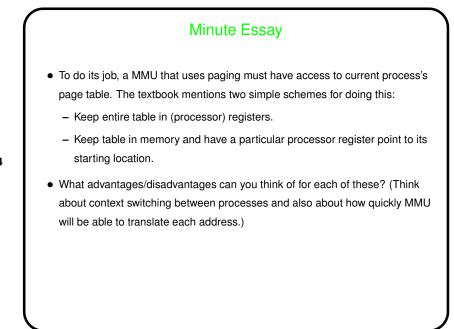
Paging — Overview
Idea — divide both address spaces and memory into fixed-size blocks ("pages" and "page frames"), allow non-contiguous allocation.
Seems like this would be more flexible and make better use of memory, but would be much more complex? Yes ...



Paging and Virtual Memory • Idea — extend this scheme to provide "virtual memory" — keep some pages on disk. Allows us to pretend we have more memory than we really do. (Not as important these days as previously, but still, sometimes it seems like however much you have of a resource it isn't always enough?) • (Compare to swapping.)

Slide 21





Minute Essay Answer

• First scheme almost surely makes for faster translations, but for a large page table it will require a lot of registers, which even if feasible would make context switches slow.

Second scheme won't slow down context switches, but as stated it isn't going to make for fast translation.