Slide 1

Administrivia

- Reminder: Homework 4 due today.
- Quiz 4 next Friday.

Slide 2

Memory Management — Recap

- The problem we're solving — partition physical memory among processes.
- Two related issues — program relocation and memory protection. Whether program relocation is potentially a problem depends on the processor's instruction set and on the program — are there instructions that use absolute addresses, and does the program use them? (For MIPS, some forms of jump and load/store instructions could.)
- Both nicely solved by defining "address space" abstraction and implementing with help from hardware (MMU). Also makes it easier to move processes around in memory. (Why would you want to?)
- We looked briefly at several schemes in which each process's memory is contiguous. Good fit for simple MMU but not very flexible. Can we do better? Yes . . .

Slide 3

Paging

- Idea — divide both address spaces and memory into fixed-size blocks ("pages" and "page frames"), allow non-contiguous allocation.
- Consider tradeoffs yet again — complexity versus flexibility, efficient use of memory.

Slide 4

Paging — Mapping Program to Physical Addresses

- One consequence — mapping from program addresses to physical addresses is much more complicated.
- How? "page table" for each process maps pages of address space to page frames; use this to calculate physical address from program address.
  (Are there page sizes for which this is easier?)
- As with base/limit scheme, makes more sense to implement this in MMU. (Notice again interaction between hardware design and o/s design.)
- Could let page table size vary, but easier to make them all the same (i.e., each process has the same size address space), have a bit to indicate valid/invalid for each entry. Attempt to access page with invalid bit — "page fault".

Slide 5

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.
- (Compare to swapping. Details later.)

Slide 6

Paging and Memory Protection

- This scheme also provides memory protection. (How?)
- We could also use it to allow processes to share memory. (How?)

Slide 7

Minute Essay

- To do its job the MMU must have access to the current process's page table. The textbook mentions two simple schemes for doing this:
  - Keep the entire table in (processor) registers.
  - Keep the table in memory and have a particular processor register point to its starting location.
- What advantages/disadvantages can you think of for each of these? (Think about context switching between processes and also about how quickly the MMU will be able to translate each address.)

Slide 8

Minute Essay Answer

- The first scheme almost surely makes for faster translations, but for a large page table it will require a lot of registers, which would make context switches slow. The second scheme won't slow down context switches, but as stated it isn't going to make for fast translation.