Slide 1

Administrivia

- (None.)

Slide 2

Paging — Review

- Recall basic ideas of paging:
  - Divide address spaces into pages, memory into page frames; allocate memory page (frame) by page (frame).
  - Use page tables (one per process) to keep track of things.
  - Use MMU to translate program (virtual) addresses into memory locations — using page table for current process. Generate “page fault” interrupt if impossible.
- Notice that we get memory protection for free; can also get memory sharing.

Slide 3

Page Table Entries

- Exactly what's in a page table entry depends partly on hardware.
- Required(?) fields — page frame number, present/absent bit.
- Optional but useful fields — bits that can be used to track usage ("referenced/modified"), bits indicating what access is allowed (e.g., read-only), etc.

Slide 4

Page Sizes and Other Details

- How big to make pages? compare extreme cases (really big, really small).
- If you know how big addresses are, what does that tell you about (maximum) sizes of physical/virtual memory?
- How big are page tables . . .

Slide 5

Page Table Size — Example

- Given a page size of 64K (2^{16}), 64-bit addresses, and 4G (2^{32}) of main memory, at least how much space is required for a page table? Assume that you want to allow each process to have the maximum address space possible with 64-bit addresses, i.e., 2^{64} bytes.
- (Hints: How many entries? How much space for each one? and no, this is not a very realistic system.)

Slide 6

Performance / Large Address Spaces

- Even with good choice of page size, serious performance implications — page table can still be big, and every memory reference involves page-table access — how to make this feasible/fast?
- Consider several options — compare access time, cost, context-switch time:
  - Keep page table for current process in registers.
  - Keep whole page table in main memory, pointed to by special register.
  - Use multilevel page tables. (More about this later.)
  - Use inverted page tables (one entry per page frame). (More about this later.)
- If page tables are in memory, performance improves with “translation lookaside buffer” (TLB) — special-purpose cache.

Slide 7

Large Address Spaces

- Clearly page tables can be big. How to make this feasible?
- One approach — multilevel page tables.
- Another approach — inverted page tables (one entry per page frame).

Slide 8

Paging and Virtual Memory

- Idea — if we don't have room for all pages of all processes in main memory, keep some on disk ("pretend we have more memory than we really do").
- Or a simpler view: All address spaces live in secondary memory / swap space / backing store, and we "page in" as needed (demand paging).
- Making this work requires help from both hardware (MMU) and software (operating system). To be continued . . .

Slide 9

Minute Essay

- None — sign in.