CSE 351
Virtual Memory
Why virtual memory?

• Provides a layer of indirection that has several benefits
  • Shared memory spaces
  • Memory isolation
  • Memory R/W/E protection
  • Illusion of large address space despite limited physical memory
Imagine virtual memory as...

- An enormous, contiguous region for storage
  - This region is broken up into fixed-size “pages”
  - When a page is used, it is stored in memory (or disk if memory is full)
- A cache for disk
  - Imagine that virtual memory is all stored on disk
  - Recently used pages of this virtual memory are cached in physical memory
Address translation

• We need a way to convert virtual addresses into physical addresses
• The common solution is page tables
  • A page table is an array of page table entries (PTEs)
  • An entry exists for every virtual page number (VPN)
  • Each entry stores a physical page number (PPN)
  • Each process gets its own page table
Page Faults

• When a page table entry points to disk, a process will generate a page fault
  • Transfers control to the OS
  • The OS will copy the page of data from disk into memory

• I like to think of this transaction as a coat check
  • OS organizes and manages the pages of virtual memory, just like a coat attendant
    stores and manages coats
Other benefits of virtual memory

• Shared memory is easy
  • In each process’s page table, simply point to the same physical page

• Memory protection
  • In addition to valid bits, store R/W/E bits
  • If a process attempts to use a page incorrectly, a segmentation fault will occur
Speeding up virtual memory

• How have we made things faster in the past?
  • Caching!

• Translation Lookaside Buffer (TLB)
  • Caches recently-translated addresses
  • This prevents the MMU from querying DRAM for every address translation