1CS61C – 11/17/2016

* OS multiplexes over many processes over available cores
  + Only 1 memory
    - But we cannot always store the contents in a context switch because stores take too much time.
* Virtual versus Physical Addresses
  + Virtual addresses come from the processor and comes out of the assembler
  + Physical memory addresses are the actual memory of the DRAM
    - Need a translator that translates between virtual addresses and physical memory addresses
  + Processes use virtual addresses, eg. 0 … 0xffff, fff
    - Many processes, all using same conflicting addresses
  + Memory uses physical addresses
  + Memory manager maps virtual to physical addresses
* Address Spaces
  + Set of addresses for all available memory locations
  + 2 kinds of memory addresses:
    - virtual address space
      * set of addresses that the user program knows about
    - physical address space
      * set of addresses that map to actual physical cell in memory
      * hidden from user applications
    - memory manager maps between these two address spaces
* Memory Manager
  + Maps VA to certain areas (blocks) of physical memory addresses
  + Makes it so that the VA don’t conflict to same PA
  + Responsibilities:
    - Map virtual to physical addresses
    - Protection:
      * Isolate memory between processes
      * Each process gets dedicated “private” memory
      * Errors in one program won’t corrupt memory of other programs
      * Prevent user programs from messing with OS’ memory
    - Swap memory to disk
      * Give illusion of larger memory by storing some content on disk
      * Disk is usually much larger & slower than DRAM
        + Using caching strategies
* Paged Memory (type of Memory Manager)
  + Physical memory (DRAM) is broken into pages
    - Typically : 4KiB+
  + Breaks Virtual Address into “page number(20 Bits)” and “offset (12 bits)”
  + Each process has a dedicated page table
* Paged Memory Address Translation
  + OS keeps track of which process is active
    - Chooses correct page table (one for each process)
  + Memory manager extracts page number from virtual address
  + Looks up page address in page table
  + Computes physical memory address from sum of page address and offset
* Protection
  + Assigning different pages in DRAM to processes also keeps them from accessing each other’s memory
    - Isolation
    - Page tables handled by OS (in supervisory mode)
  + Sharing is possible too
    - OS may assign same physical page to several processes
  + Write Protection
    - Has “Write protected” bit
    - You wouldn’t want to overwrite your program code of the program you are currently running
    - Protects from Malware so it cannot overwrite your software to do things that you wouldn’t want to occur
* Page Tables
  + Eg. 32-Bit virtual address, 4 – KiB pages
    - Single page table size:
      * 4\* 2^20 Bytes = 4-MiB
      * 0.1% of 4 – GiB memory
      * But much too large for a cache
    - Total size for 256 processes (each needs a page table)
      * 256\*4\*2^20 Bytes = 4-MiB
      * 1=GiB
      * 25% of 4 –GiB memory!!!
  + Store page tables in memory (DRAM)
    - Two (slow) memory accesses per lw/sw on cache miss
    - Solution:
      * Transfer blocks (not words) between DRAM and processor cache
        + Spatial locality
      * Cache is frequently used page table entries
  + Options
    - Increase page size
      * E.g. doubling page size cuts PT size in half
      * At the expense of potentially wasted memory
    - Hierarchical page tables
      * With decreasing page size
    - Most programs use only fraction of memory
      * Split PT in two (or more parts)
* Hierarchical Page Table
  + Exploits sparsity of virtual address space use
  + Virtual Address is divided up between L1 index and L2 index and offset
  + A two-level page table has 3 memory accesses so we need to make it faster some other way
    - Use cache
    - Cache some translations in Translation Lookaside Buffers (TLB)
      * If TLB hit = Single-Cycle Translation
      * If TLB miss = Page – Table Walk to refill
* TLB Designs
  + Typically 32-128 entries, usually fully associative
    - Each entry maps a large page, hence less spatial locality across pages = more likely that two entries conflict
    - Sometimes larger TLBS (256-512 entries) are 4-8 way set associative
* TLB Reach : Size of largest virtual address space that can be simultaneously mapped by TLB
* Execve
  + Code that loads program into memory for execu tion