

EECS 370

Discussion 11

Week of 4/10 – 4/16

- I) Virtual Memory
 - a. We want a program to see a full usable address space, even if our DRAM isn't big enough
 - b. We want multiple programs to EACH see a full usable address space
 - c. We want to be able to enforce different policies on different parts of memory (read-only, etc.)
 - d. Solution: take whatever address the programmer sees (on a 32-bit machine, this can be any 32-bit address) and **translate** it to an actual address in physical memory
 - e. Layers of abstraction: programmer does not see any of this
 - f. Huge operating system topic
 - g. Physical memory acts like a “cache”; data is brought in and evicted as necessary, interacting with the disk
- II) Terminology
 - a. One block of data, whether virtual or physical, is called a **page**
 - b. “Cache miss” is called a page fault, handled as exception
 - c. **Page table register** – special register that points to beginning of page table
 - d. Every virtual address has a page offset, analogous to block offset
 - e. Each address also has one or more virtual page numbers that get translated into physical numbers
- III) Multi-level page tables
 - a. More levels of indirection
 - b. Top level is called superpage table
 - c. Instead of translating to a physical address, it translates to a pointer to the next level of page tables
 - d. This continues until the last level contains the actual address
 - e. Advantage: much less space used on average!
- IV) Translation Look-aside Buffer (TLB)
 - a. Cache for most common address translations
 - b. Always go to TLB before page tables; saves accesses to main memory
- V) Caches
 - a. Should we use virtual or physical addresses to access caches?
 - b. Physical address
 - i. Go to TLB before accessing cache
 - ii. Slower, but cache can be retained after context switches (TLB flushed)
 - c. Virtual address
 - i. Perform TLB lookup and cache access in parallel
 - ii. Faster, but cache must be flushed after each context switch!

□ Given the following

- 4KB page size, physical memory of 16KB, page table stored in physical page 0 and can never be evicted, 20 bit, byte-addressable virtual address space.
- The page table initially has virtual page 0 in physical page 1, virtual page 1 in physical page 2 and no valid data in other physical pages.

Virt addr	Virt page	Page fault?	Phys addr
0x00F0C			
0x01F0C			
0x20F0C			
0x00100			
0x00200			
0x30000			
0x01FFF			
0x00200			

Class problem – multi-level VM

- Design the virtual memory of a byte addressable processor with 24-bits long addresses. No cache in the system. 256Kbytes of memory installed, and no additional memory can be added.
- Virtual memory page: 512 Bytes. Each page table entry must be an integer number of bytes, and must be the smallest size required to fit the physical page number + 1 bit to mark valid-entry
- **Design** a two-level virtual memory system. We want each second-level page table to fit exactly in one memory page, and superpage table entries are 3 bytes each (a memory address pointer to a 2L page table). Compute: Number of entries in each second-level page table; Number of virtual address bits used to index the second-level page table; Number of virtual address bits used to index the superpage table; Size of the superpage table.