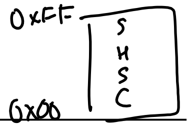
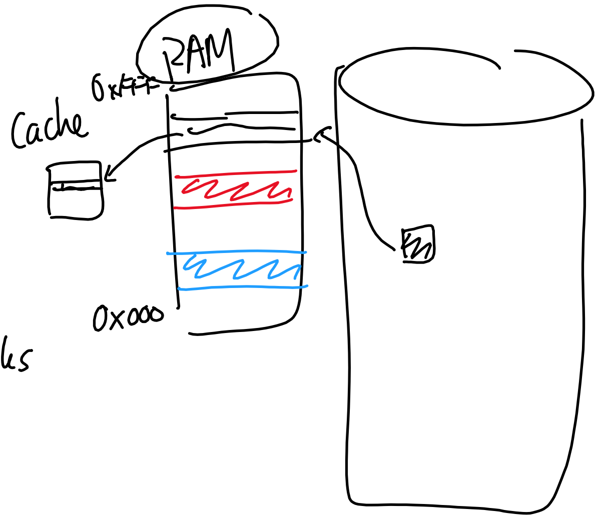
Jon 101 Dis 13

Wednesday, November 28, 2018 2:05 PM





CS 61C Fall 2018

Virtual Memory

Discussion 13: November 26, 2018

1 Addressing

Virtual Address (VA) What your program uses - process sees this

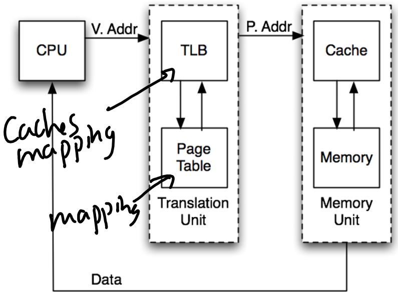
Virtual Page Number (VPN)	Page Offset

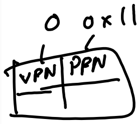
≠ =

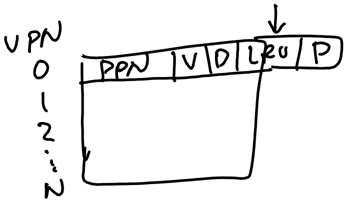
Physical Address (PA) What actually determines where in memory to go - OS handles this

Physical Page Number (PPN)	Page Offset

With 4 KiB pages and byte addresses, $2^{\text{page offset bits}} = 4096$, so there are 12 page offset bits. Translate virtual addresses (VA) to physical addresses (PA) using the translation lookaside buffer (TLB) and page table. Then, use the physical address to access memory as the program intended.







Pages

A chunk of memory or disk with a set size. Addresses in the same virtual page map to addresses in the same physical page. The page table determines the mapping.

Valid	Dirty	Permission Bits	PPN
— Page entry (VPN: 0) —			
— Page entry (VPN: 1) —			

Each stored row of the page table is called a **page table entry**. The page table is stored in memory: the OS sets a register telling the hardware the address of the first entry of the page table. The processor updates the "dirty" bit in the page table which lets the OS to know whether updating a page on disk is necessary. Each process gets its own page table.

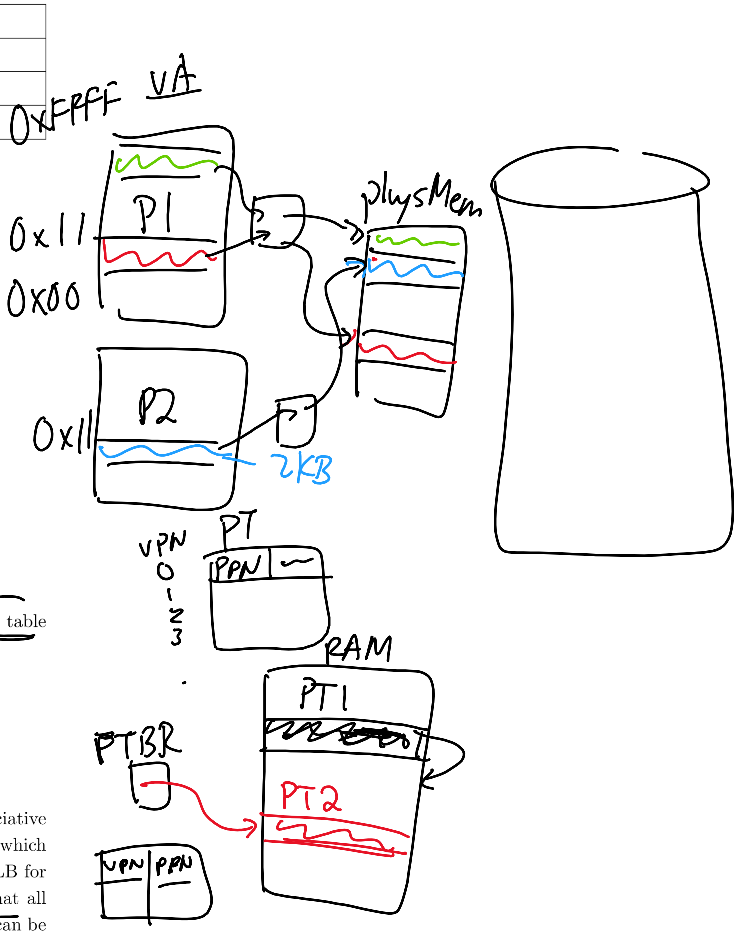
Protection Fault The page table entry for a virtual page has permission bits that prohibit the requested operation.

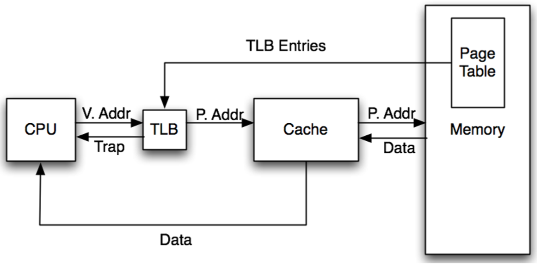
Page Fault The page table entry for a virtual page has its valid bit set to false. The entry is not in memory. - Not in RAM ⇒ go to disk

Translation Lookaside Buffer

A cache for the page table. Each block is a single page table entry. If an entry is not in the TLB, it's a TLB miss. Assuming fully associative:

TLB Valid	Tag (VPN)	Page Table Entry		
		Page Dirty	Permission Bits	PPN
— TLB entry —				
— TLB entry —				





1.1 What are three specific benefits of using virtual memory?

1) ∞ memory (illusion)

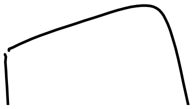
2) consisent addr.

3) security & Isolation

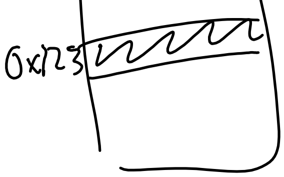
1.2 What should happen to the TLB when a new value is loaded into the page table address register? ✓

flushing the TLB
(set all valid bits to 0)

1.3 A processor has 16-bit addresses, 256 byte pages, and an 8-entry fully associative TLB with LRU replacement (the LRU field is 3 bits and encodes the order in which pages were accessed, 0 being the most recent). At some time instant, the TLB for the current process is the initial state given in the table below. Assume that all current page table entries are in the initial TLB. Assume also that all pages can be read from and written to. Fill in the final state of the TLB according to the access pattern below.



TLB size: hardware
PT size: #rows = tot #pages



8:0 8: VPN

16-8=8

2^8 B sized pages → 8 offset bits

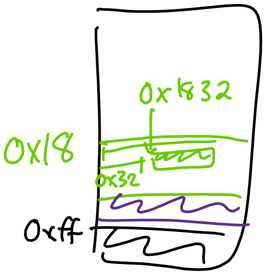
Free Physical Pages ~~0x17~~, ~~0x18~~, ~~0x19~~

Access Pattern

Virtual Addr

VPN 0

1. 0x11f0 (**Read**) hit
2. 0x1301 (**Write**) page fault
3. 0x20ae (**Write**) hit
4. 0x2332 (**Write**) page fault
5. 0x20ff (**Read**) hit
6. 0x3415 (**Write**)



Initial TLB

VPN	PPN	Valid	Dirty	LRU						
0x01	0x11	1	1	0	1	2	3	4	4	5
~~0x00~~ 0x13	~~0x00~~ 0x17	~~0~~ 1	~~0~~ 1	7	7	0	1	2	2	3
0x10	0x13	1	1	1	2	3	4	5	5	6
0x20	0x12	1	~~0~~ 1	5	5	6	0	1	0	1
~~0x00~~ 23	~~0x00~~ 18	~~0~~ 1	~~0~~ 1	7	7	7	7	0	1	2
0x11	0x14	1	0	4	0	1	2	3	3	4
0xac	0x15	1	1	2	3	4	5	6	6	7
~~0xff~~ 34	~~0xff~~ 19	1	~~0~~ 1	3	4	5	6	7	7	0

Final TLB

VPN	PPN	Valid	Dirty	LRU