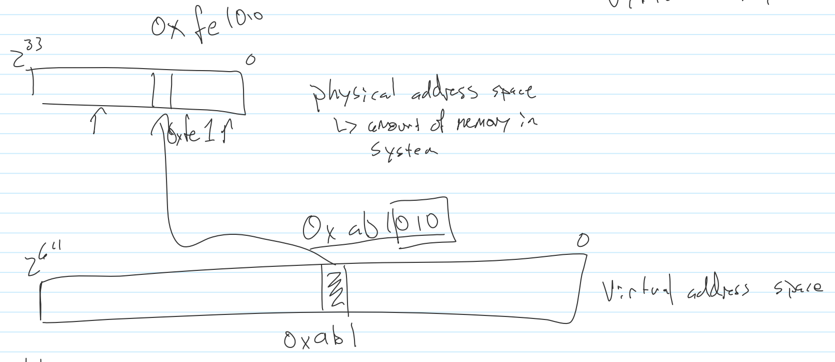
Lecture 18: Virtual memory

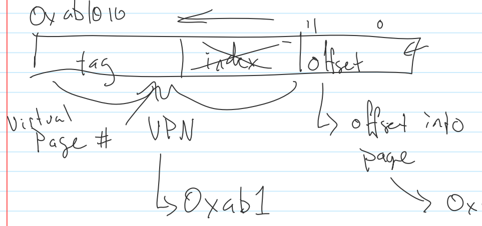
Thursday, March 8, 2018 9:29 AM

Outline

- Virtual memory
  - How to translate virtual to physical addresses?
  - Multi-level page tables and x86 translation
  - TLBs and MMUs
  - Finish memory
- Parallel machine types

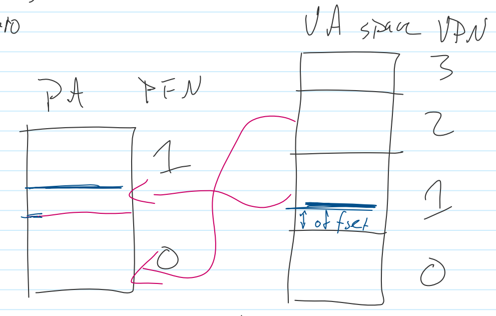
Virtual to physical mapping?





in caches we have blocks (64B)

in virtual memory it is broken up into pages (4kB)



Page table

$2 \rightarrow 0$

$1 \rightarrow 1$

page are 256B

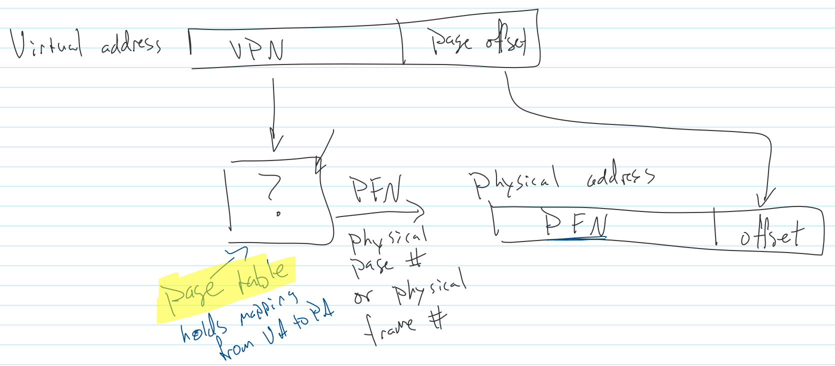
↳ 8 bits offset

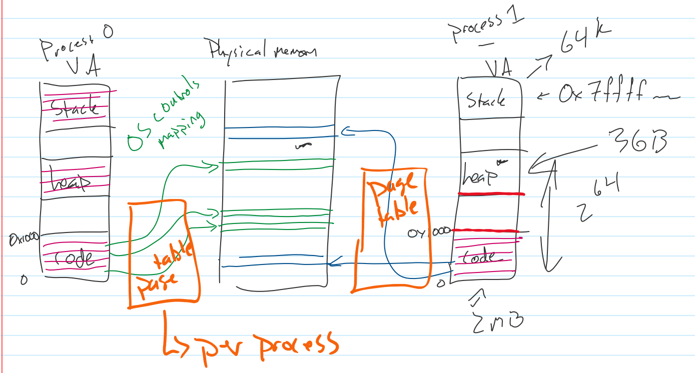
total VA size 2bits for VPN

↳ 8+2 → 10bits

VA 0x214 → VPN(2) offset (0x14)

PA 0x014 → PPN(0) offset (0x14)





int *a = &b;

↗ 64 bits

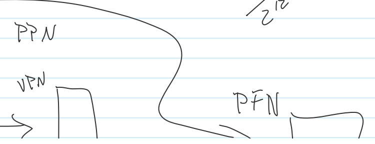
every process has its own Virtual address space

Page table

- hash table
  key is VPN data is PPN
  "map" C++
- flat array →

# of pages in 64 VA? 4K

$2^{64}/2^{12} = 2^{52}$ for flat array → 2^{55} bytes



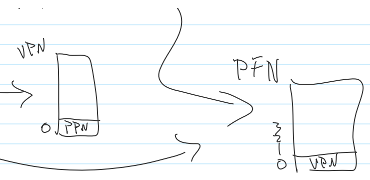
key is VPN

"map" c++

- flat array →

↳

-> inverted page table



- Tree or multi-level page table

Intel software developer manual Volume 3A section 4.5.

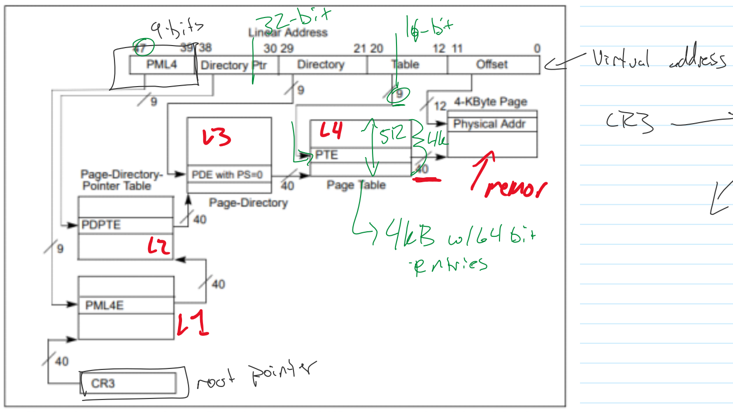
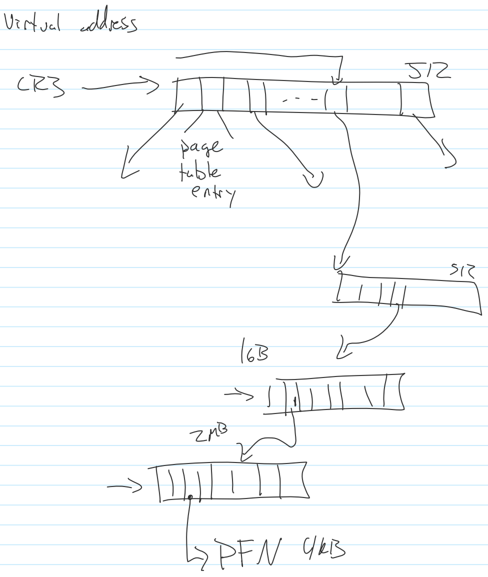
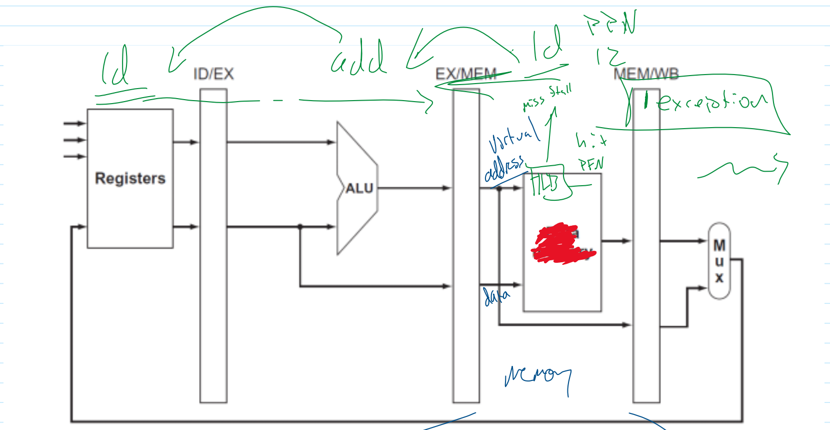
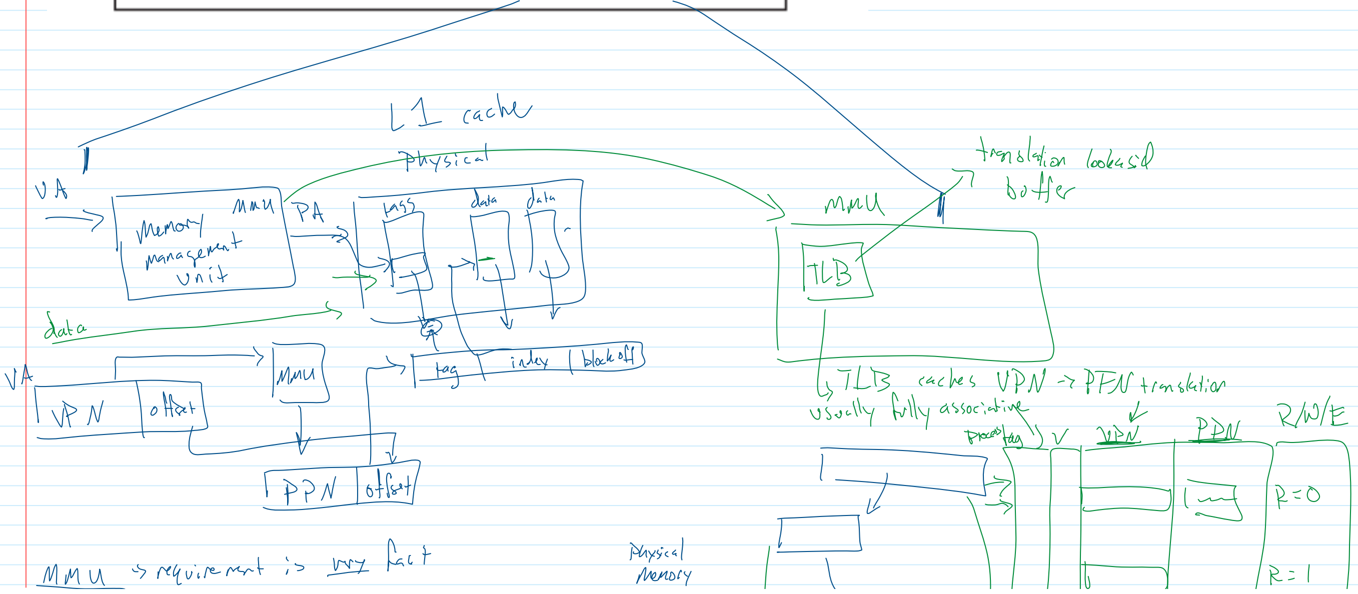


Figure 4-8. Linear-Address Translation to a 4-KByte Page using 4-Level Paging



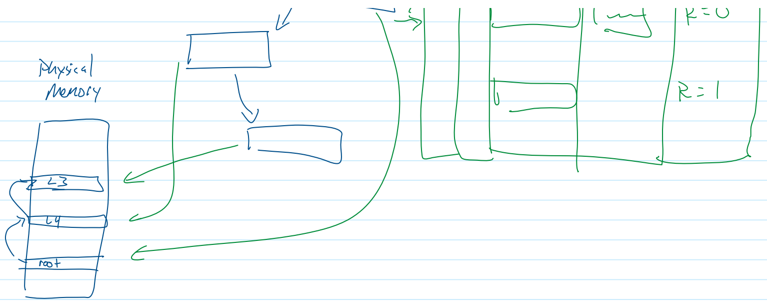
page table is "walked" by hardware





MMU → requirement is very fast

PPN [illegible]



MMU → requirement is very fast

for evry request translate from VA to PA

4 memory requests to get translation

Tree size

~ 1B per page

~ 2GB → $2^{31}/2^{12} = 2^{19}$ @ 8B per page → 2^{22}

4MB