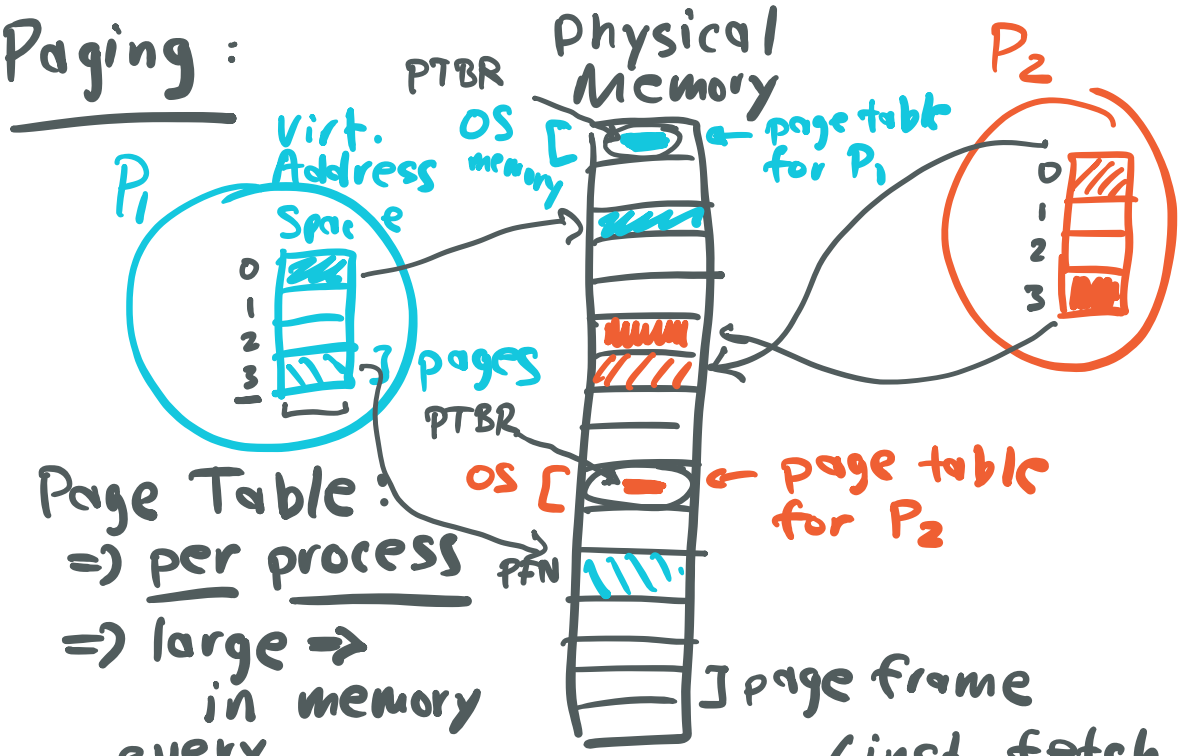
Paging:

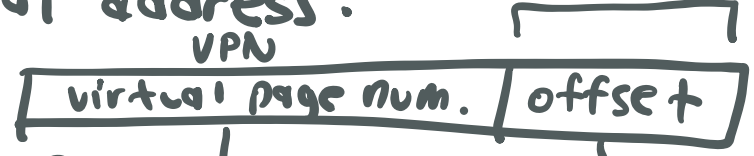


Page Table:
- => per process
- => large → in memory

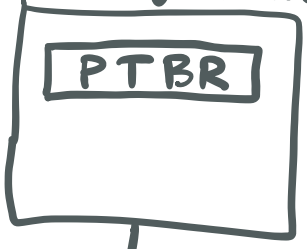
Upon every memory reference: (inst. fetch, load/store)

address translation

virtual address:



what does h/w need to know?
→ location of page table



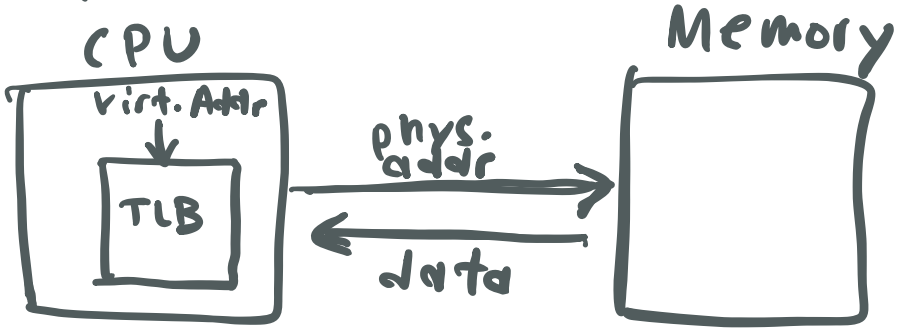
(not translated)

physical frame number (PFN)

PTBR : page table base register
points to start of page table
of currently running process

Paging: Too Slow
to do translation: fetch (PTE) Page Table Entry
(extra memory reference)

Solution: Hardware (TLB)
translation lookaside buffer
(address translation cache)
⇒ in CPU hardware that
holds some number of
"popular" translations



Method:
virtual address
↓

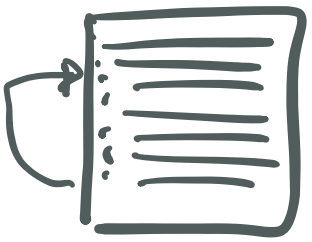
TLB

1) if VPN is in the TLB, "hit"
use PFN in TLB, form phys. addr., perform mem access

OR

2) if not, "miss"
hardware: use PTBR to access page table, update TLB w/ necessary info., try again => TLB hit

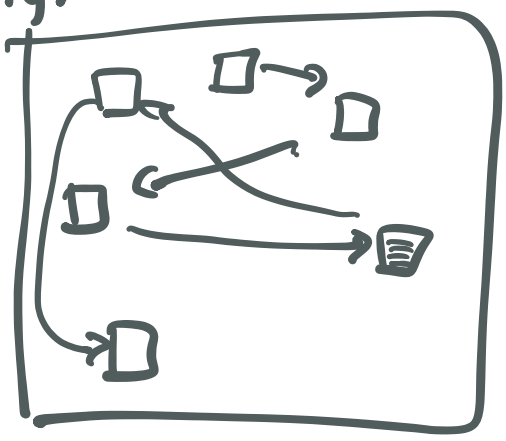
e.g. code locality
one virtual page



=> same VPN

data locality?

e.g. linked list



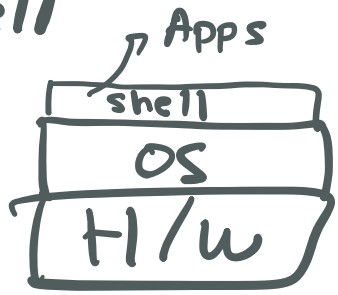
assumption: locality exists

what happens if TLB is full?
=> replacement
[kick out something to put new thing in]

TLB: 4 VPN -> PFN

Admin:

=> P2a available : shell



=> Midterm next week

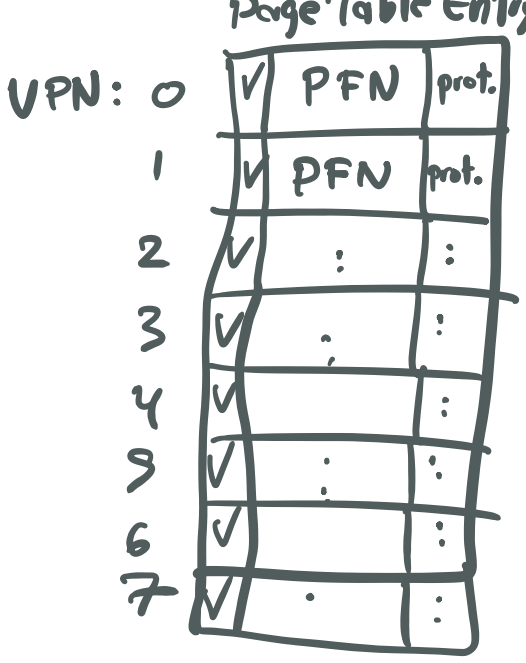
material: [through Tuesday 10/5]

closed book,

1-page helper sheet, both sides (8.5" x 11")

Page Table Structure:

Linear Page Table: Array



V: valid

V=1 prog can legally access the page

V=0 => not

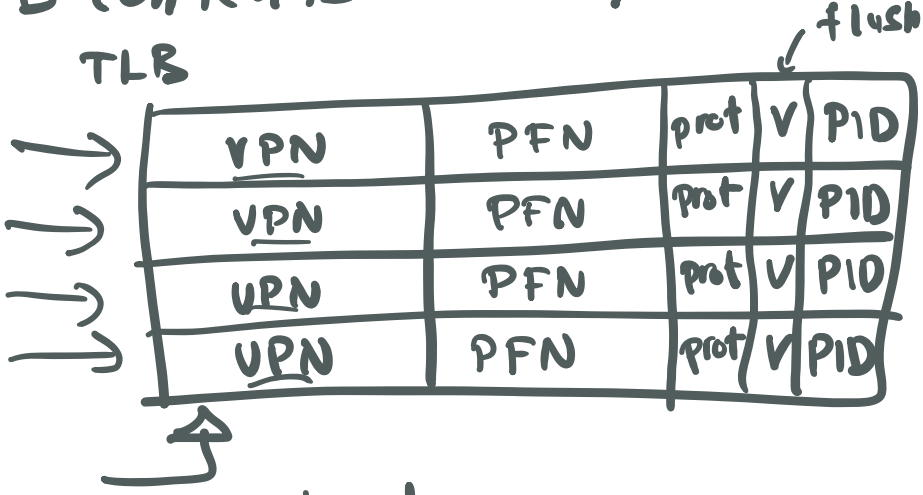
PFN: if V=1, Phys page (frame) that holds the page

protection:

allows for sharing ← [code: read only, exec.
heap: r/w

protection: read only, read/write, execute

TLB contents: similar, but different



H/W is going to search these in parallel to see if there is a hit

V: valid: in TLB means that this entry has valid translation

when is TLB valid bit needed?

e.g. P_1 is running, TLB fills w/ P_1 VPN → PFN translations

timer interrupt

scheduler => P_2 (context switch)

can we run P_2? NO

(because valid translations for P_1 in TLB)

solutions:

=> flush the TLB (set valid for all entries to 0)

OR

=> mark each entry w/ Process ID

H/w managed TLB:

TLB miss => hardware accesses page table + updates TLB

S/w managed : OS