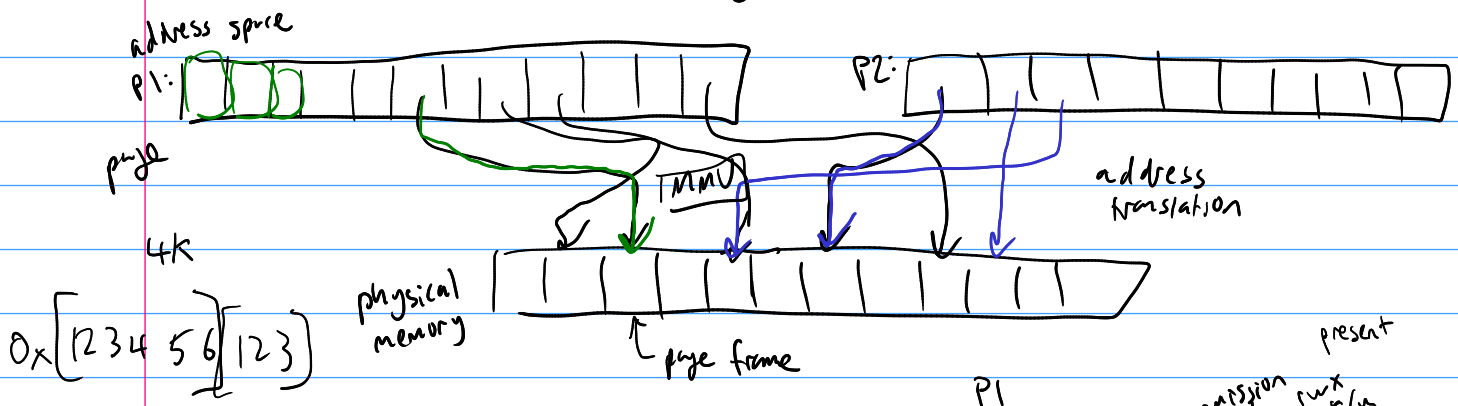
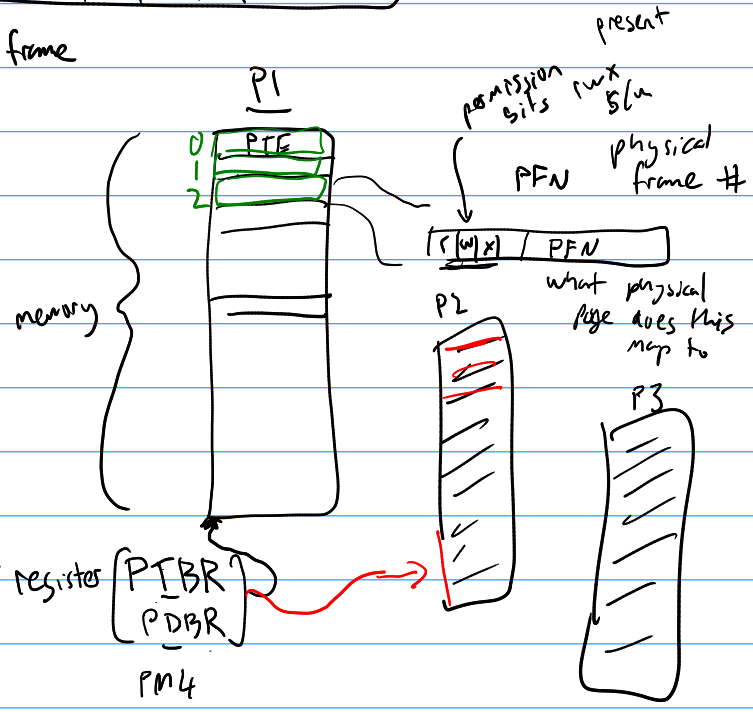
CS 3214 "virtual memory"



How it is done:

OS sets up page table

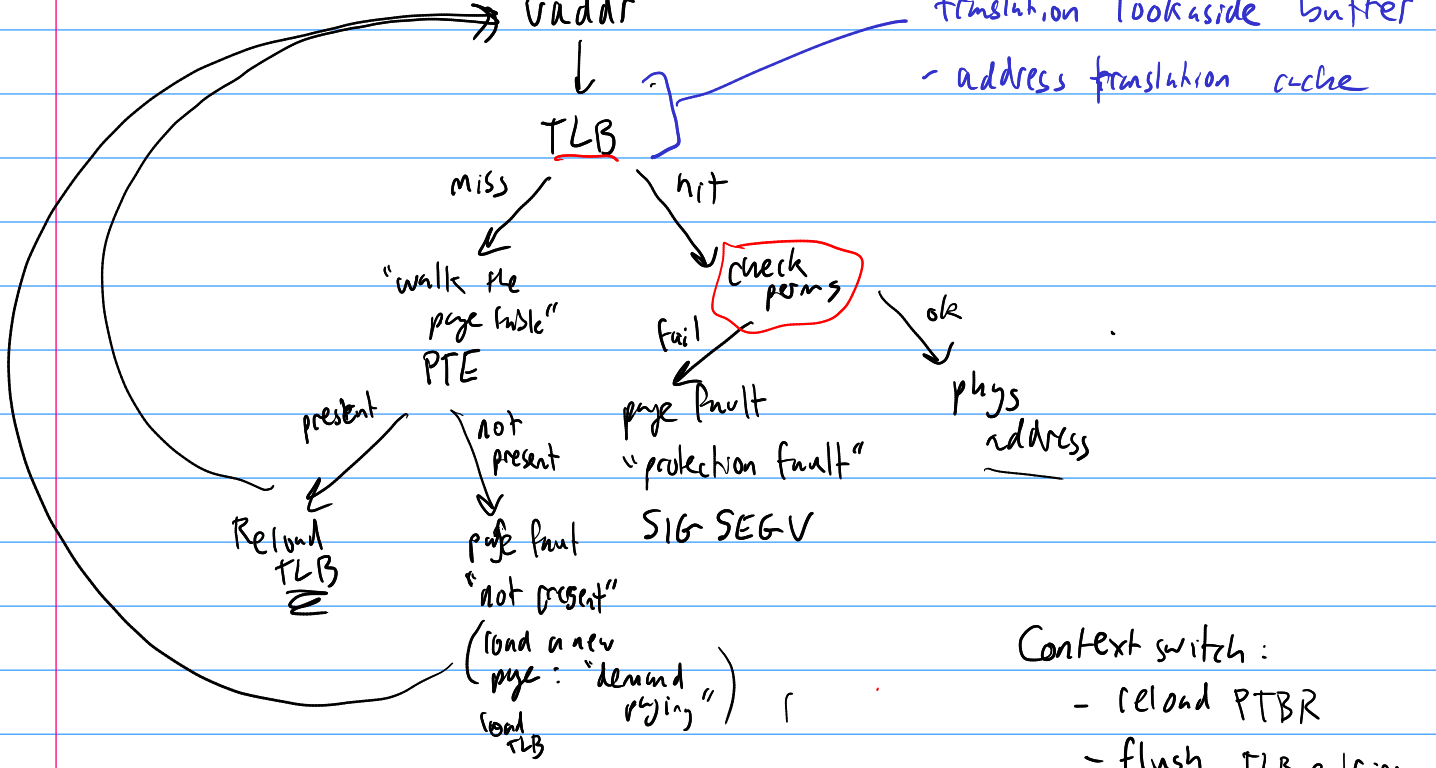
map virt → phys



to deal with large page tables

- use trees

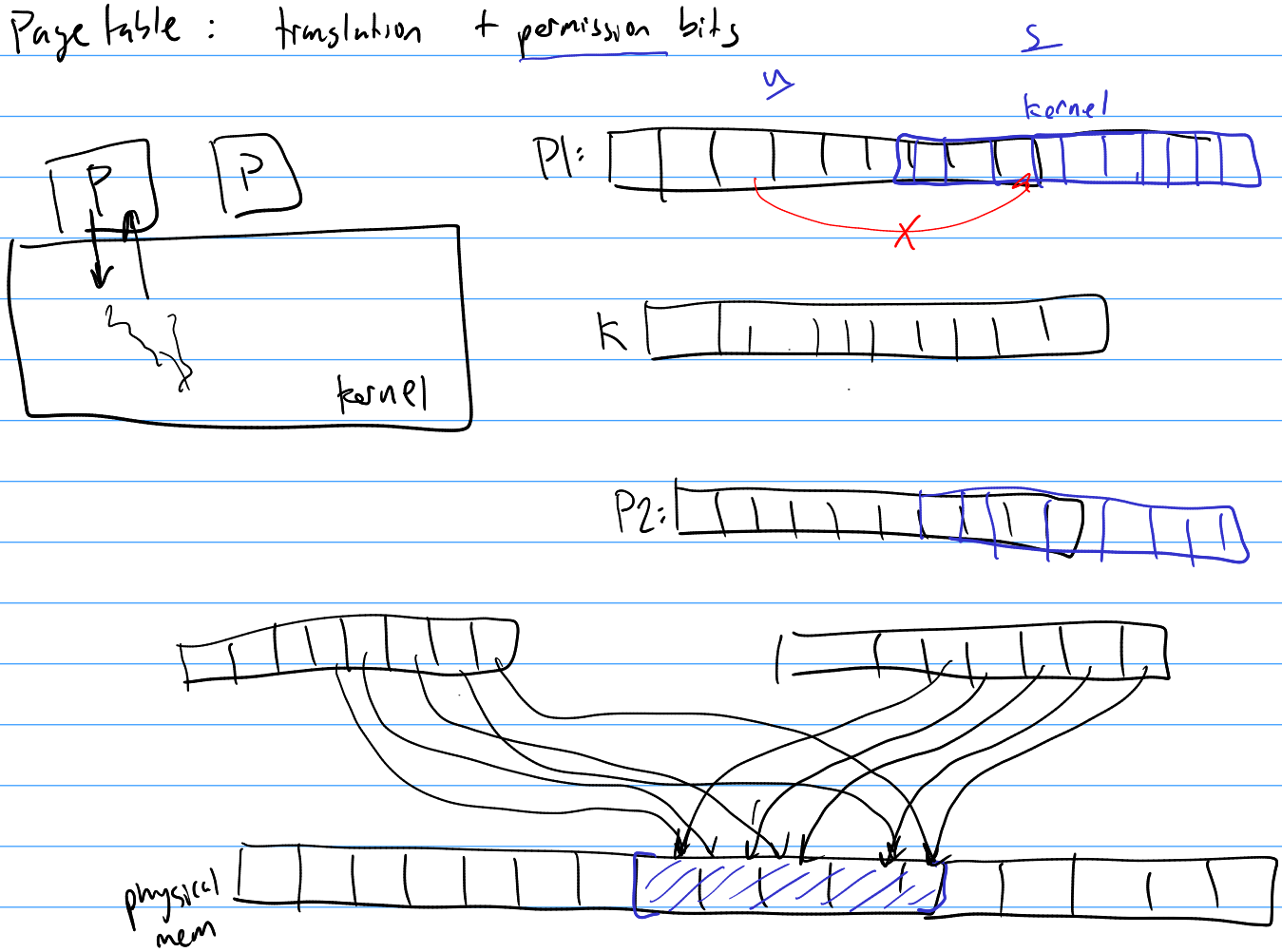
H/W does this



Context switch:

- reload PTBR
- flush TLB entries for outgoing process

Page table: translation + permission bits



kernel mapped to all addr. spaces

Meltdown: if user access data in kernel page

① processor fetch into cache ② processor checks perm bit ③ page fault SIGSEGV

transient execution attacks

Spectre

side effect: kernel data in cache

→ figure out where the kernel is mapped KASLR

→ figure out contents of kernel mem (cache collisions)

KPTI kernel page table isolation

don't map kernel into user address spaces

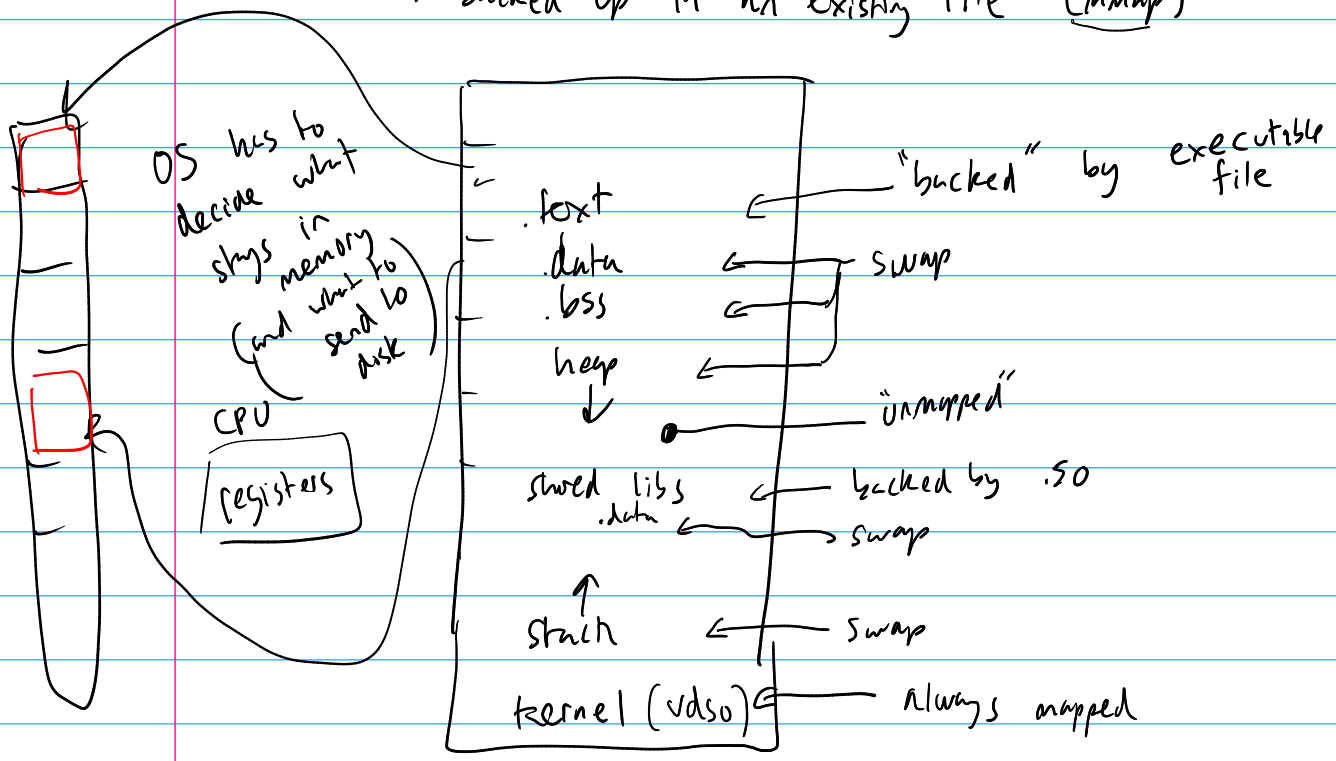
Virtual memory can exceed Physical memory

idea: only need to hold what process needs NOW

otherwise: disk

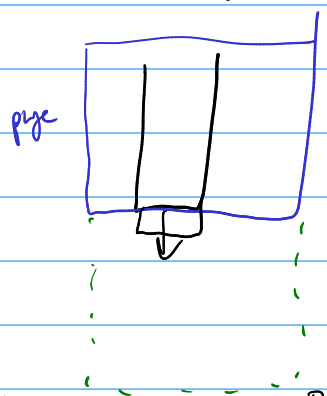
→ swap file

→ backed up in an existing file (mmap)



page faults:

not always fatal

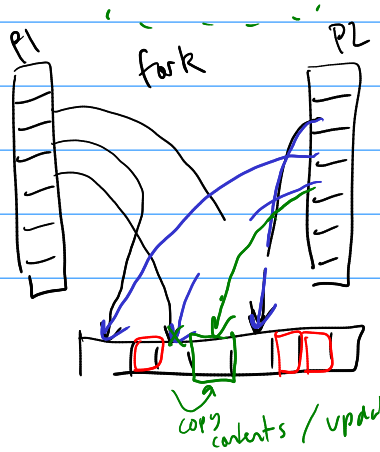


page fault

OS: where was it?

alloc new page / install new mapping in PT

retry instruction



OS: copy page table

set all entries to read-only

on fault (write):

alloc new page

update PT r/w

retried

exec : remove all PT entries & start over