CS 219 Lecture # 20.

(#) mov ebx, (% 0x abcd0023)

- in real mode: — physical address
- in protected mode — virtual address

w/ paging — mmu ~~of~~ on CPU performs page table walk.

look pgdir ← this is PA

- use offset from VA on pgdir to check pte.

if pte valid — move to next level
else page fault.

check pte flags for validity. — V, P etc.

& recurse on page table levels.

(#) on valid/lazy page fault
or on enthusiastic mapping of v2p

- find free physical page for mapping.
- if no free page, evict + swap!

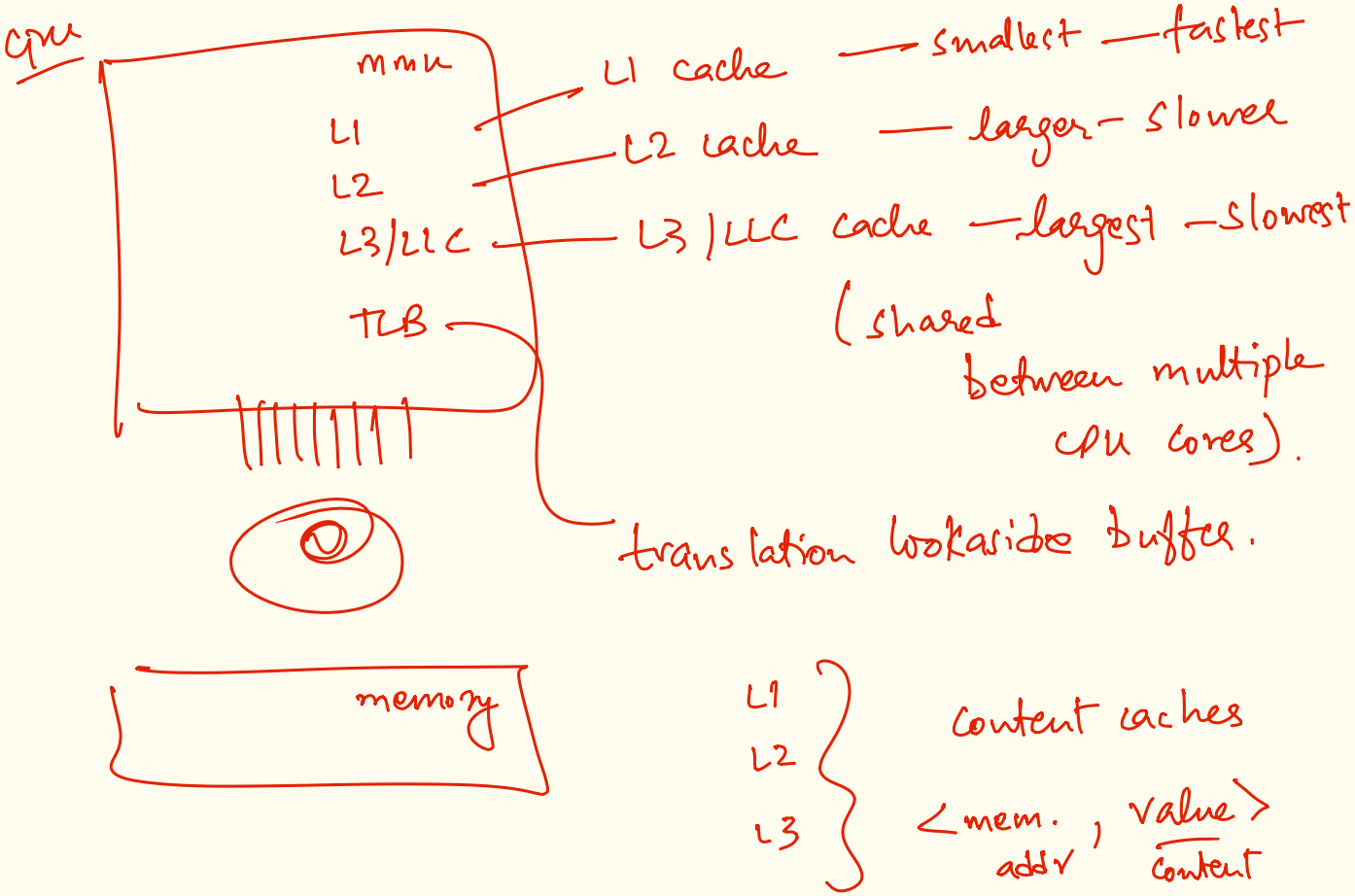
(#) OS is overhead

paging is an overhead!

(i) page table walk!

(ii) pages tables need memory!

(i) ISA/h/w has a set of caches to reduce mmu translation latency potentially



TLB - cache of VPN → PPN ~ physical page

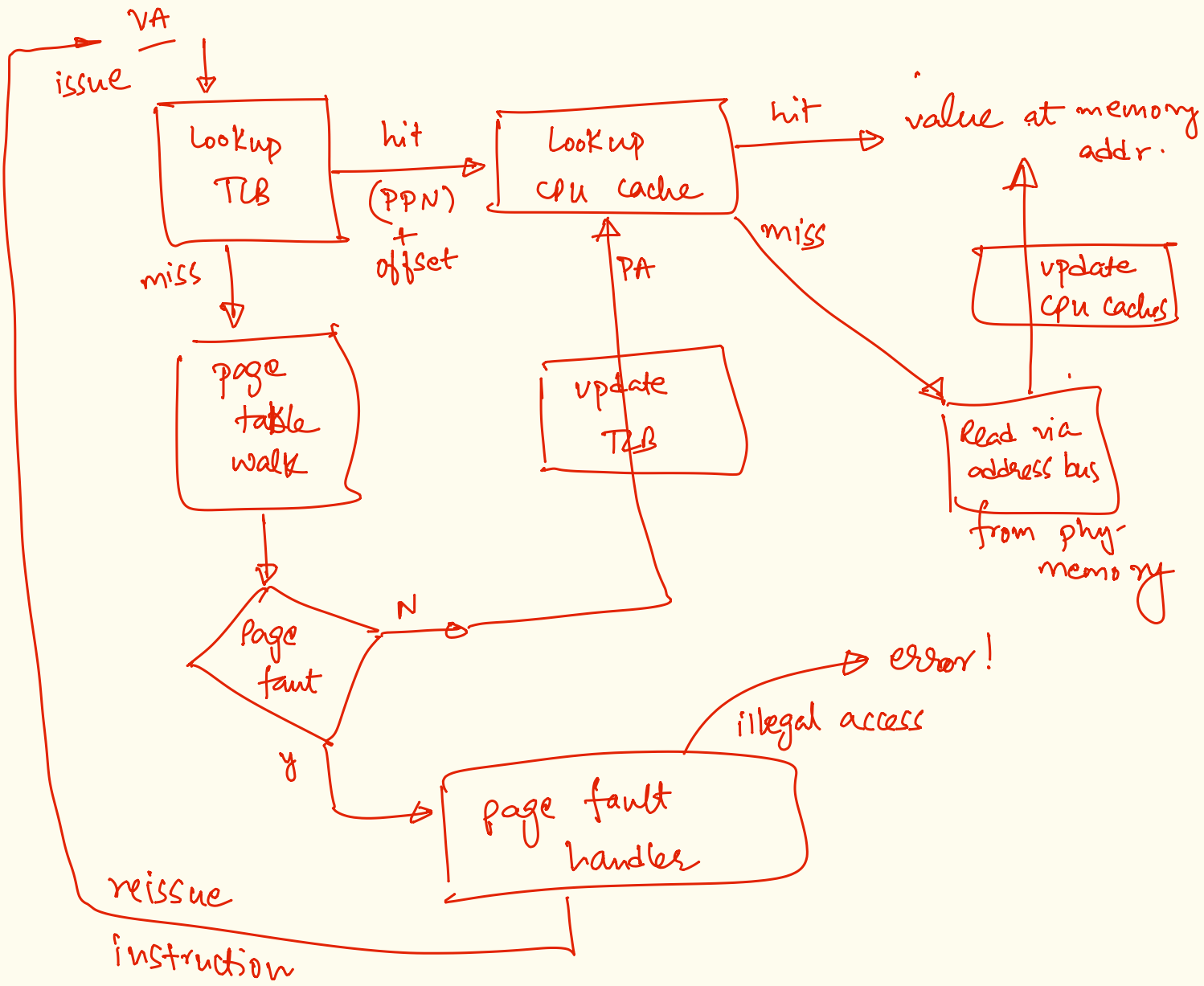
virtual page

(VA) → VPN ~→ PPN

translation cache

~ TLB's hold VPN → PPN

- need to flushed on process switch!
- to avoid stale / incorrect page table mappings.

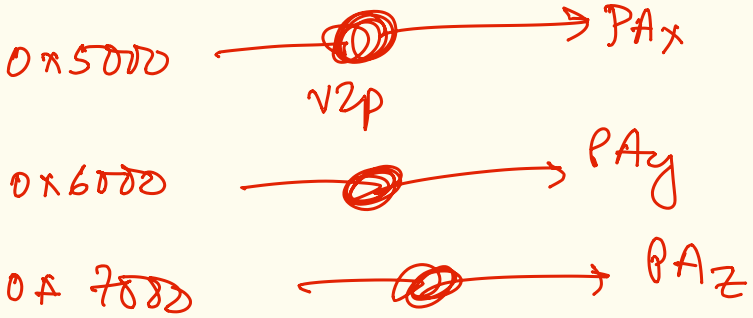


(*) Why is TLB not updated by the page fault handler?

(i) ~ pg. fault handler is slow — code
unless CPU/ISA exposes an instruction for TLB update — this cannot happen.

(ii) handler can do many things!

VA
0x5000 — pg. fault → lazy alloc.
valid VA
free PA page — PAx



(*) Why the re-issue?

note story starts with a VA access!

- pg. fault handler knows the PPN/PA

Q how to access the PA?

└ needs of VA.

└ on reissue use the original VA that was issued.

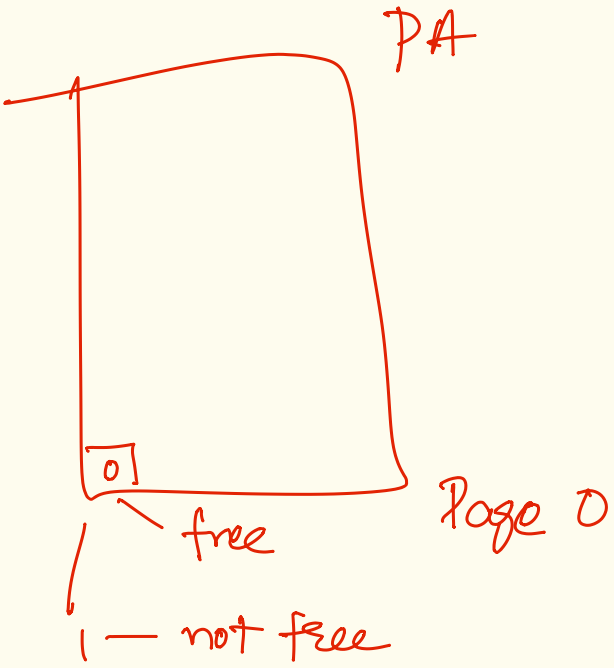
never used (OR) — a kva that maps to this address!

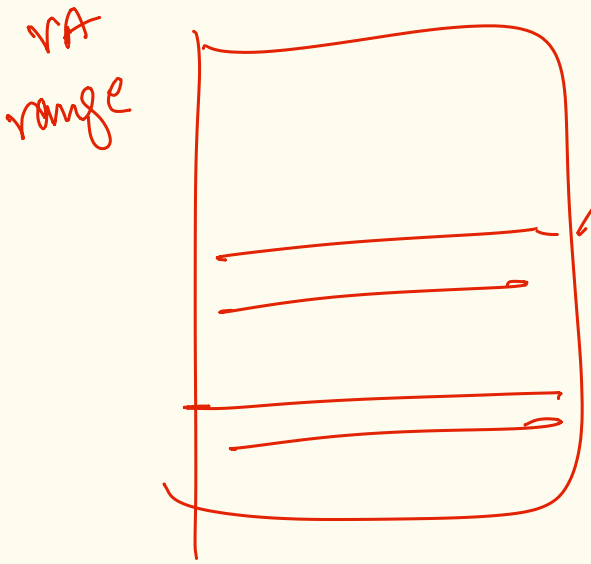
issue

reads

flag in

PA page





mange VA space!

malloc — which VA range address to return?