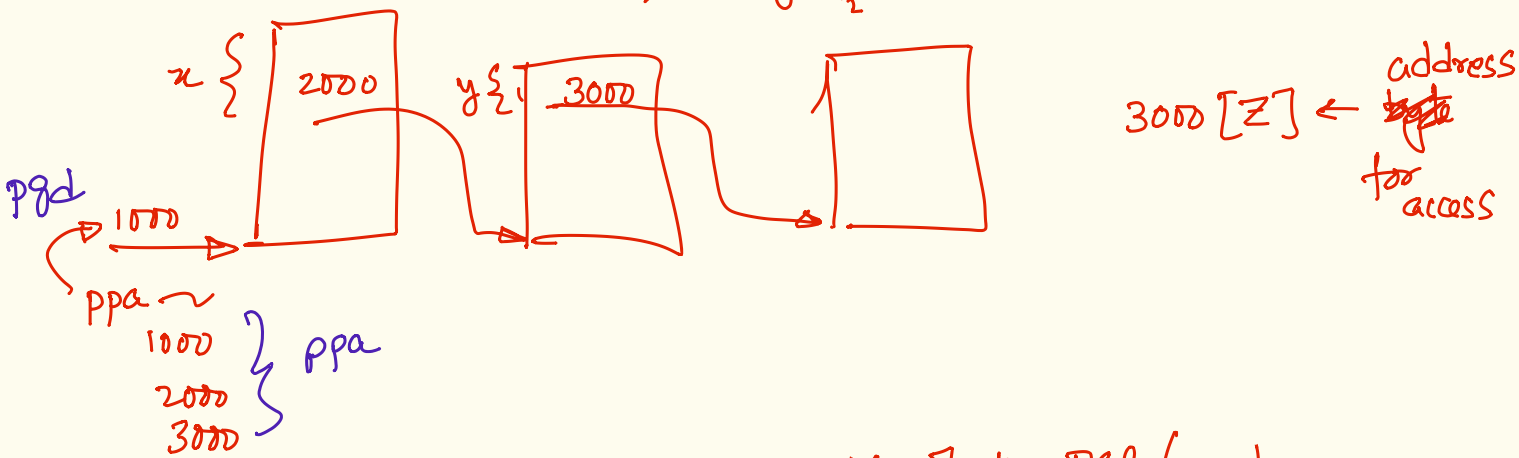
Lecture 6 CS695

- memory virtualization for VMs.

- assignment #1 due soon!
  25th Jan.
  11.59 pm.

* example
  - starting point is a user VA (gVA)



  - store 1000 in pgd variable of the PCB and load CR3 with pgd 1000
    - in the guest mark 1000 as used/active
    - in the vmm mpa(1000) → a1000 is marked as active/used (update p2m)
    - mark 2000 as active
    - ⇒ Kva(1000)[x] ← 2000 — p2m → a1000 (mpa); move 3000 to active list/set
    - Kva(2000)[Y] ← 3000 — p2m → a2000 (mpa)

(v) (m)

Kva(1000) → a1000

Kva(2000) → a2000

④ PV vs full virtualization for memory

- (i) explicit vs implicit vmm invocation.

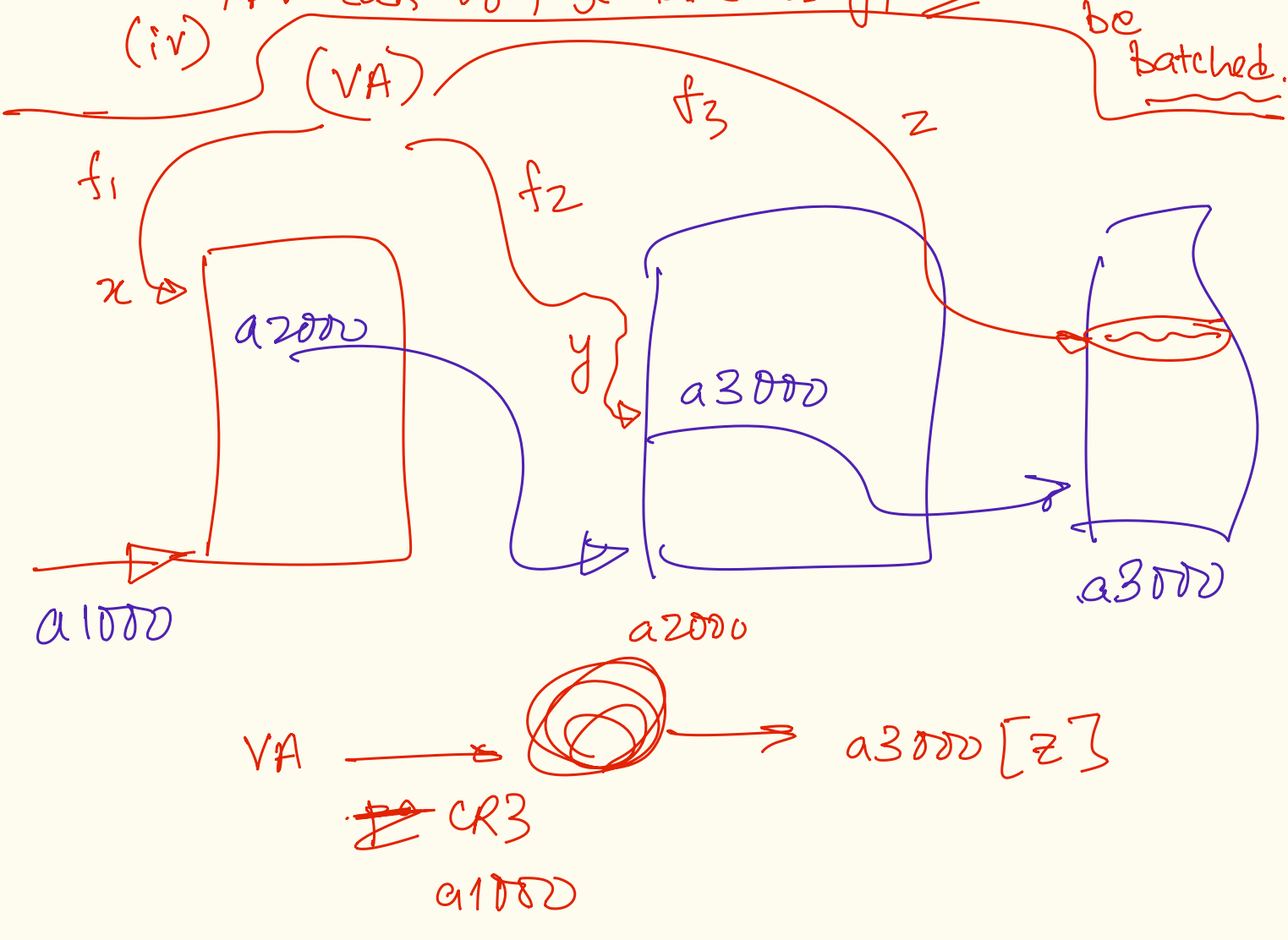
(ii) both techniques mark page table pages read-only.

(iii) w/ full virtualization shadow page table has to mimic a full page table along w/ PTE flags.

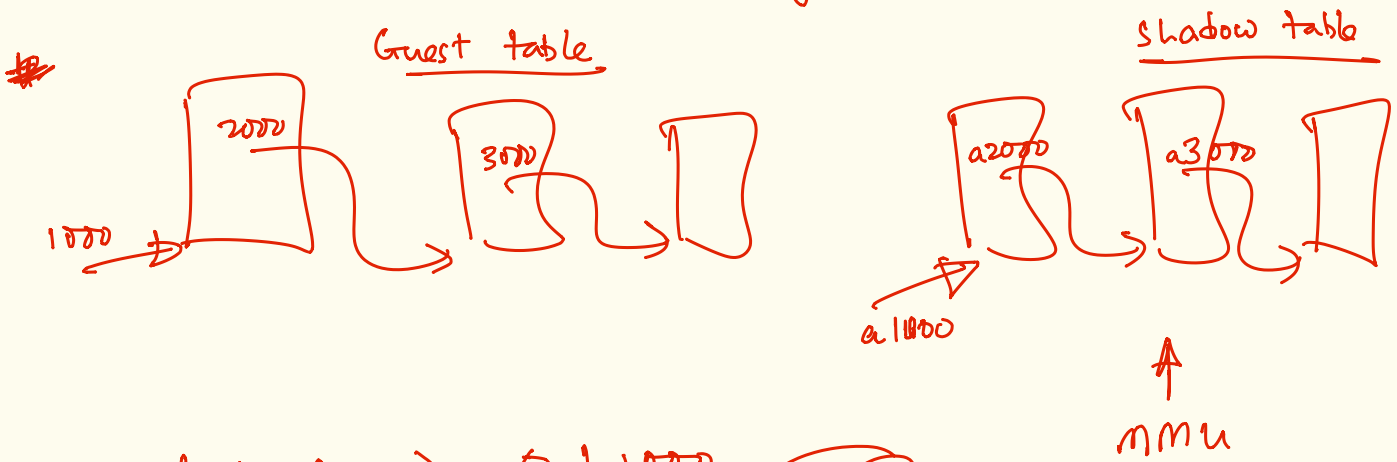
- update to PTE flags - A/D/R/W/S have to be synchronized.

w/ PV reads of page table costly, but writes can be batched.

(iv)



Full virtualization Shadow Page table.

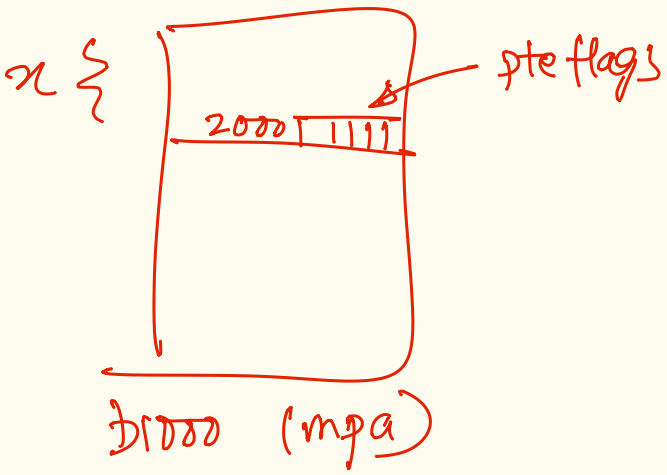


read Kva(1000) → b1000

vmm

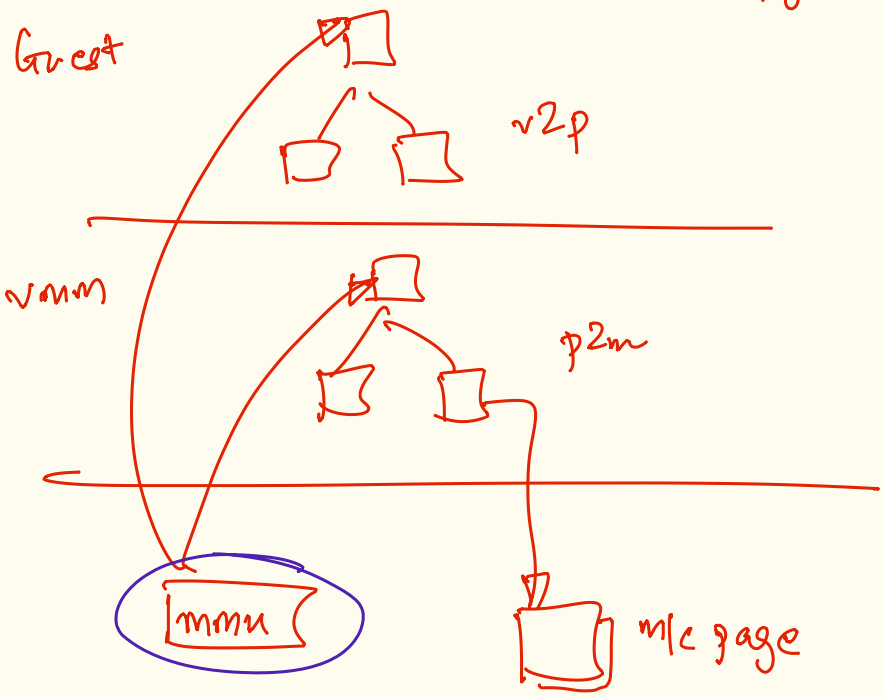
Kva(b1000) ← 2000

Kva (2000) → b2000 Kva (b2000) ← 3000

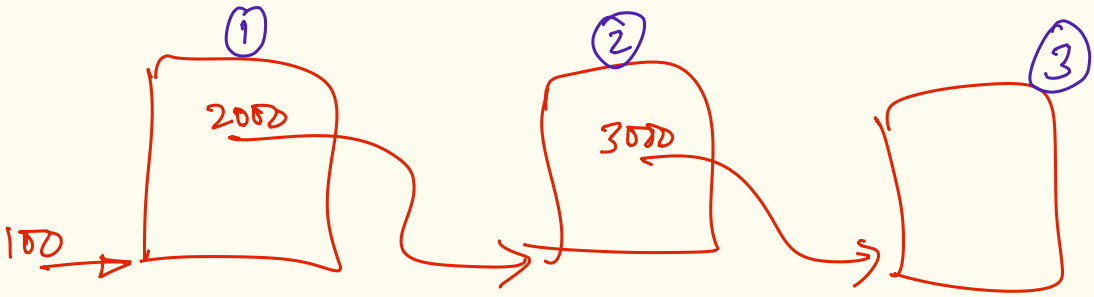


Design 3: H/w assisted EPT/NPT based memory virtualization

extended pg. tables
nested pg. tables



* mmu looks up p2m mapping for every p address of v2p.



Kva (1000) [x] ⇒ p 1000 to m a1000

ppa

pg. walk on the p2m table (2)

a1000 [x] — 2000 (1)

KVa (2000) [Y] ⇒ 2000 a2000

p2m walk (2)

a [2000] [Y] — 3000 (1)

Kva (3000) [Z] ← access (1)

3000 — a3000

p2m walk (2)

n-levels

* #pg. table page access = n^2+n

+1 (mapped) page