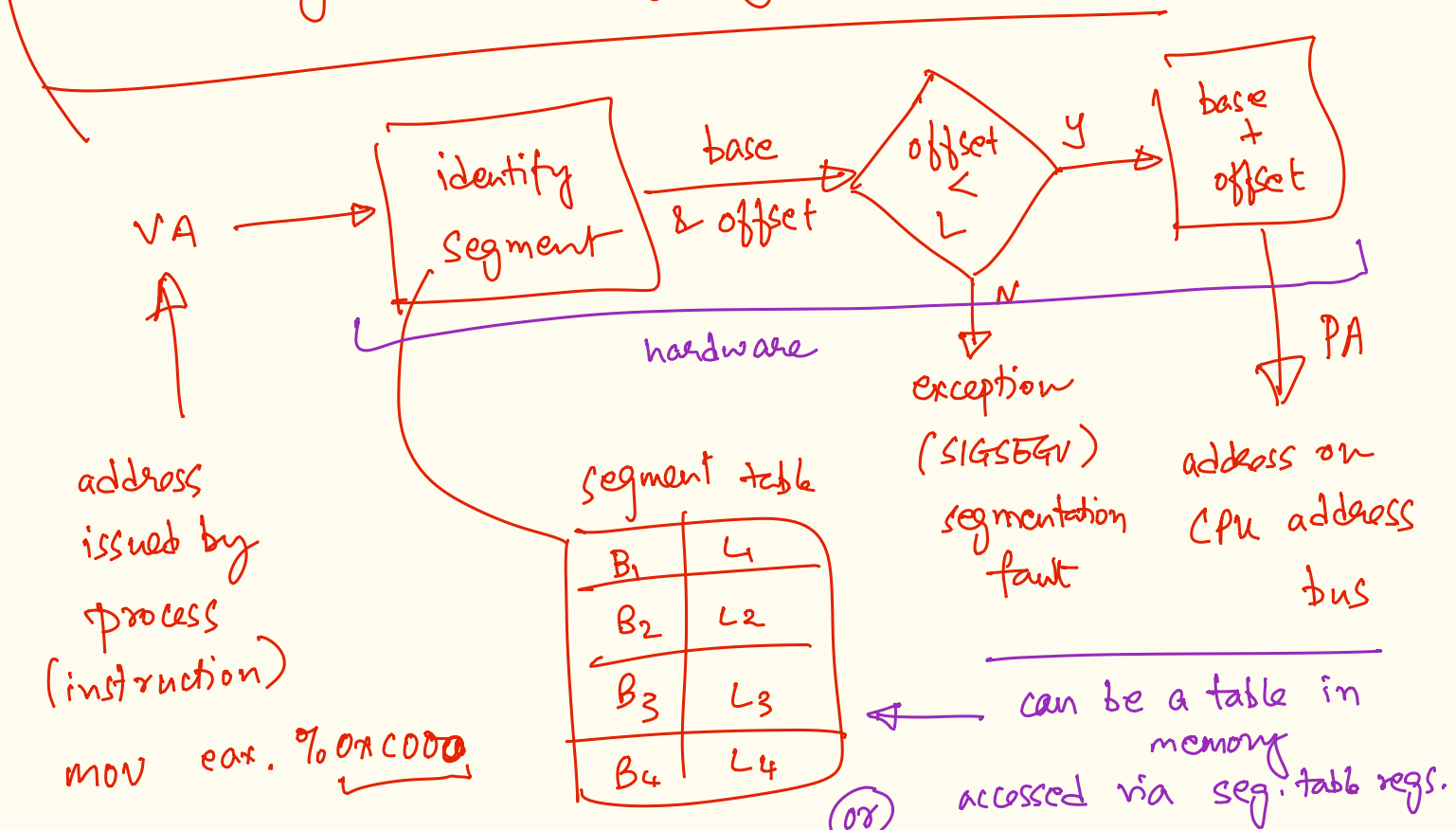
CS 219 — Lecture 14

* memory virtualization
  - allocation
  - multiplexing
  - access.

- isolation
- zero-stating
- full address range
- transparent
- efficiency!
  - fragmentation
    - internal
    - external

- design 0 (save & restore)
- design 1 (base & bound)
- design 2 (segmentation)
- design 3 (paging)



(*) how large should each segment be?

↳ internal & external fragmentation.

⇒ segments are contiguous in virtual & physical memory (address spaces)

paging.

↳ allocate all (addresses) memory in fixed sizes (page)

↳ no external fragmentation

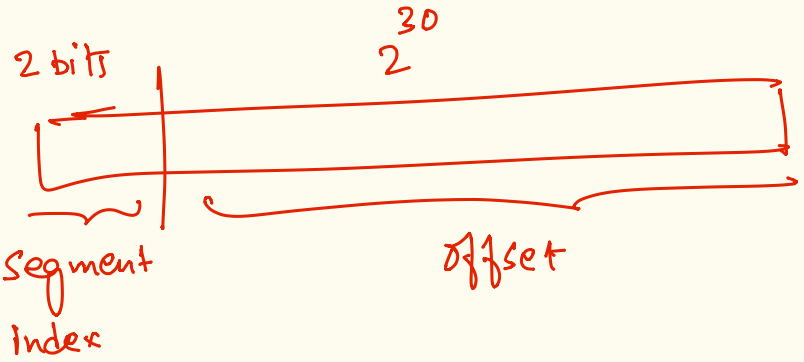
↳ internal fragmentation is bounded.

32-bit address

segmentation

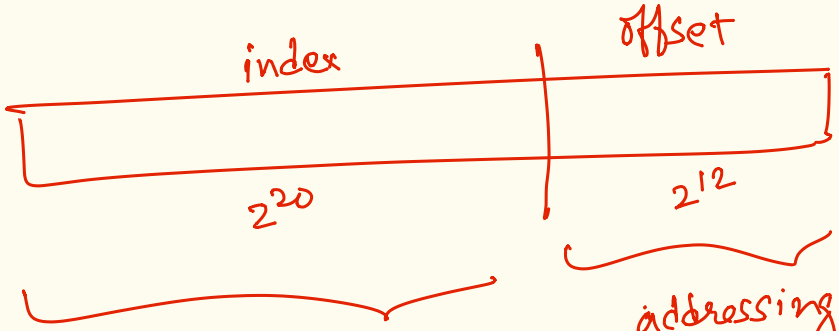
4 segments

VA:



paging

VA:



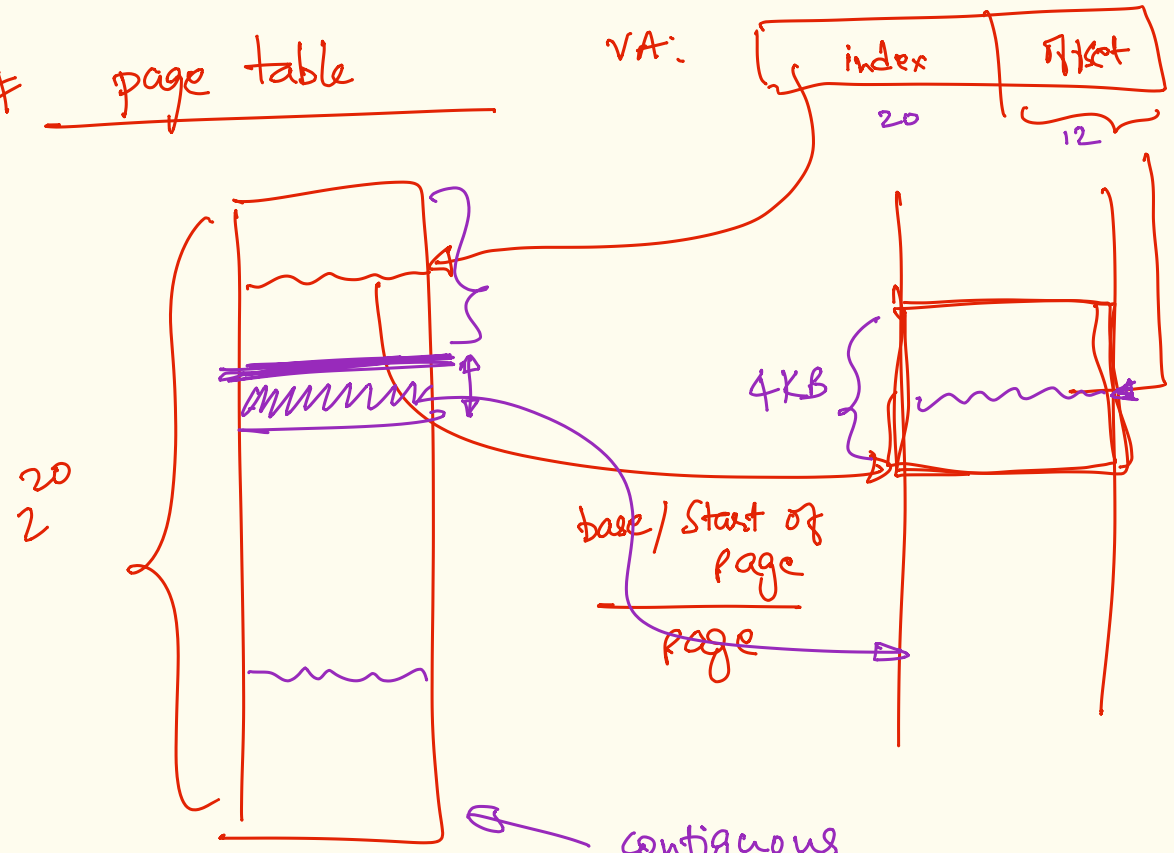
2^{20} index items

addressing for 4 KB of data

$2^{12} \Rightarrow 4096$

2^{20} pages $\times$ 2^{12} address/page $= 2^{32}$

page table

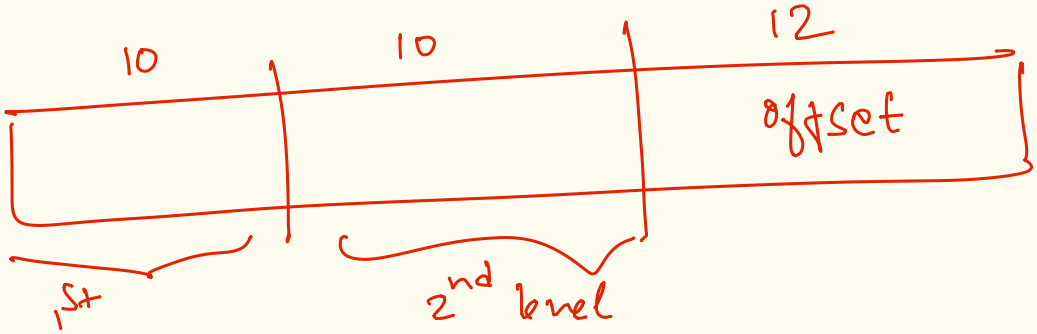


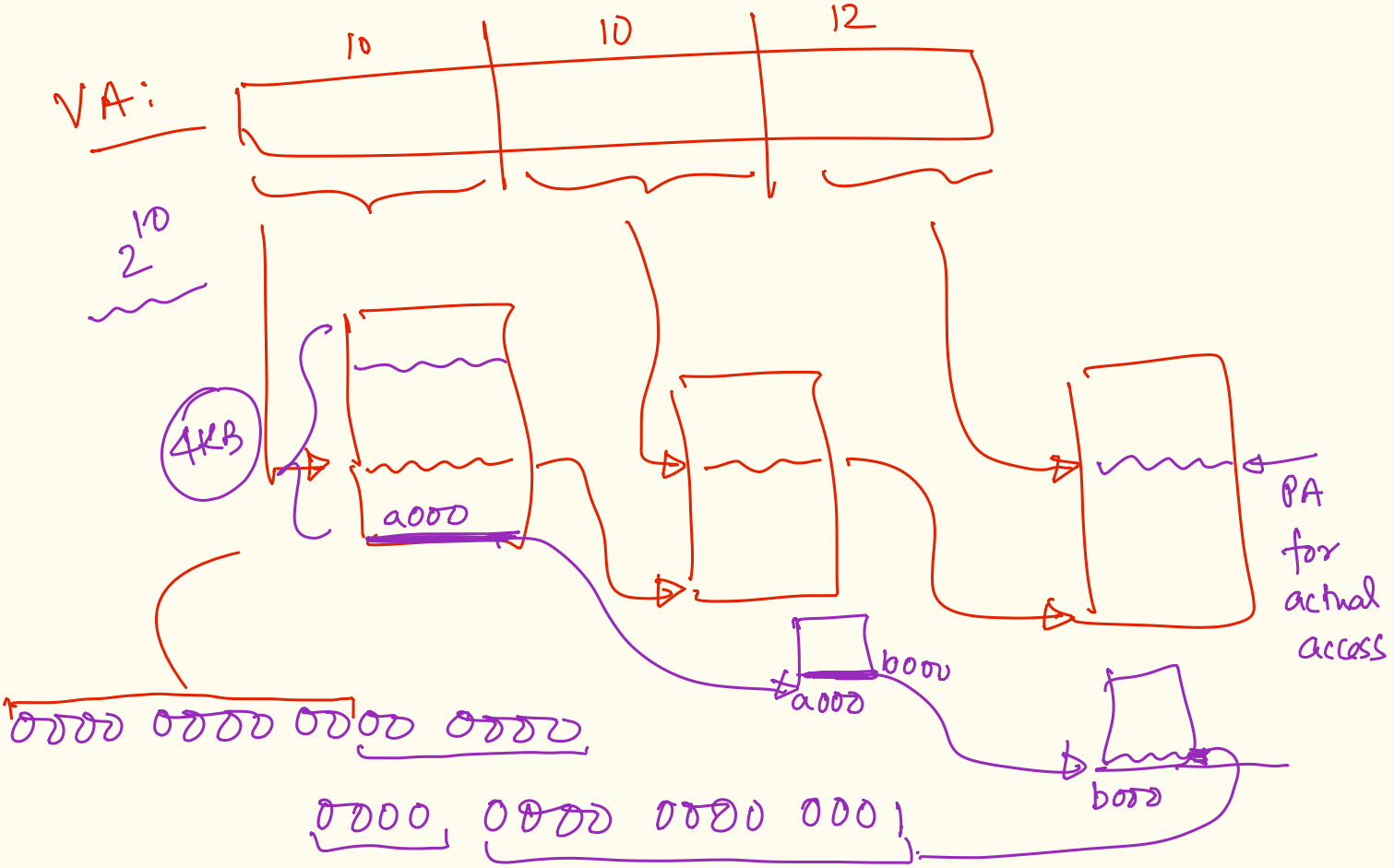
- contiguous
- not all mappings required / valid at all times
  ∴ process may not use all memory
- still entire memory for full table has to be reserved.

---

two level page table

eg: 32-bit addresses & 4KB pages





---

page size = 4KB ~ 4096 ~ 2^{12}

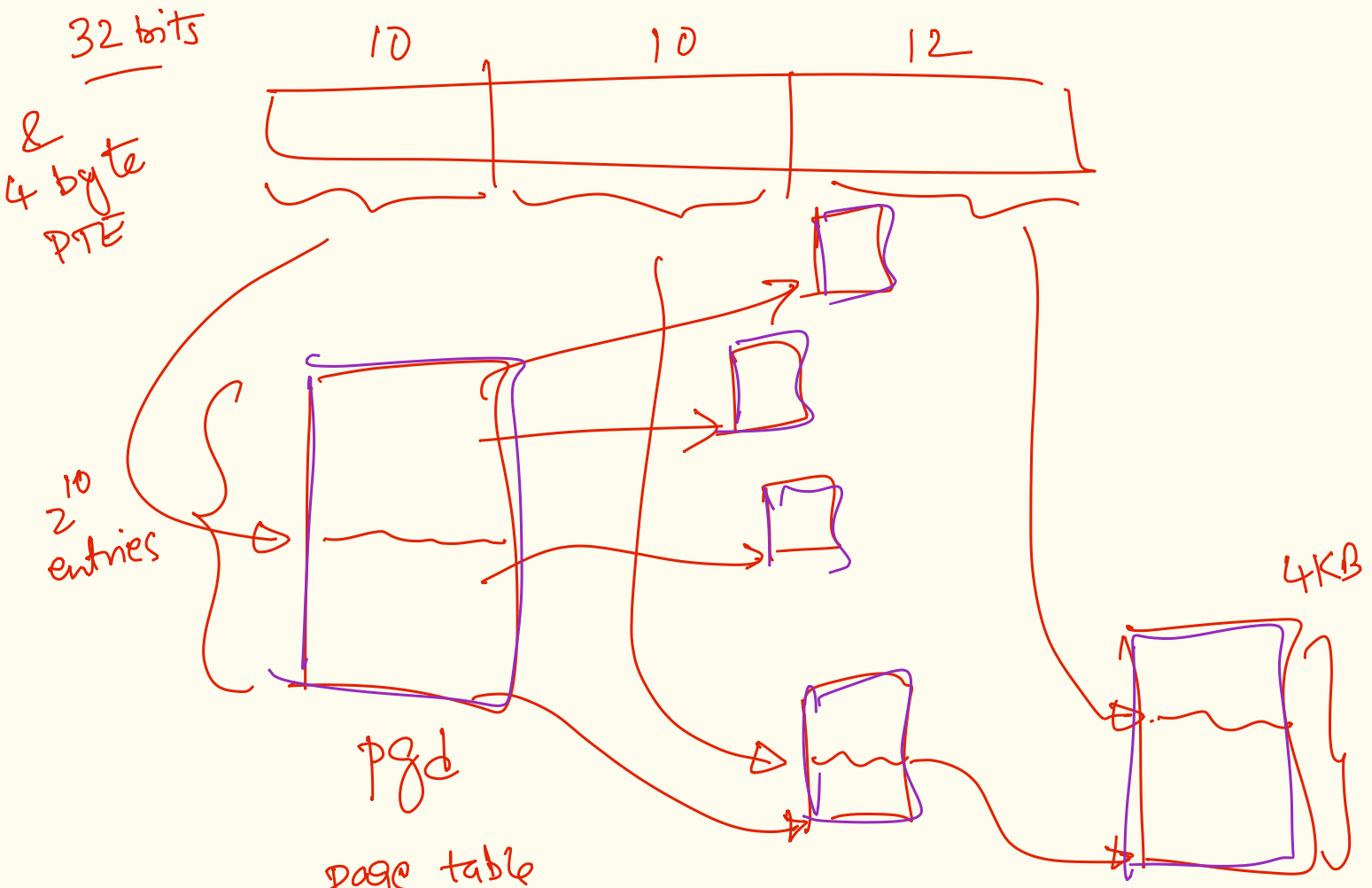
index / entries per page = $\frac{\text{pg. size}}{\text{size of pte}}$

(page table entry)

e.g: pte = 4 bytes

$= 2^2$

entries = $\frac{2^{12}}{2^2} = 2^{10}$



page table directory

$1 \times 2^{10} \times 2^{10} \times 2^{12} \sim 2^{32}$

full address range

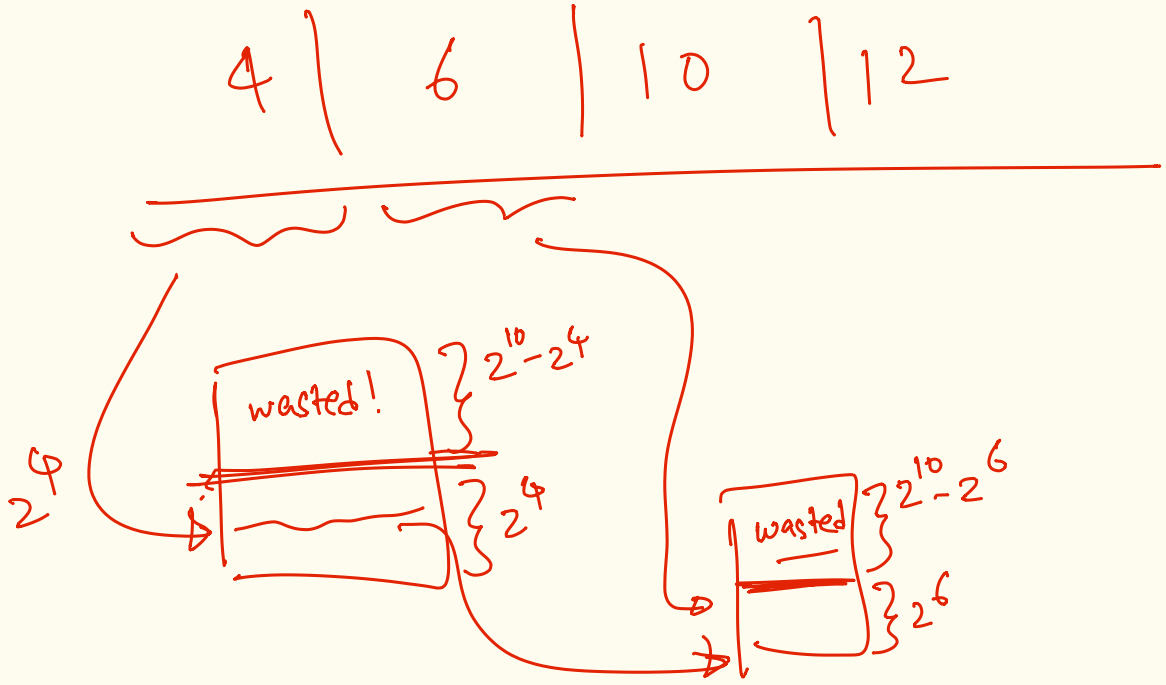
$(1+2^{10})$ pages

$(1+2^{10}) \times 4KB$

$((1+2^{10})\, 2^2 = 2^{20} + 2^2$

all mem accessed paging

2^{20} seg.



pte: 4 bytes

└ is sufficient / more than sufficient to store address of page (start of page)

└ all page-aligned (4KB addresses have 12 LSB bits zero.

⇒ 20 (MSB) bits are sufficient for pte address storage