CSE 330: Operating Systems Class: 17 Date: 10/18

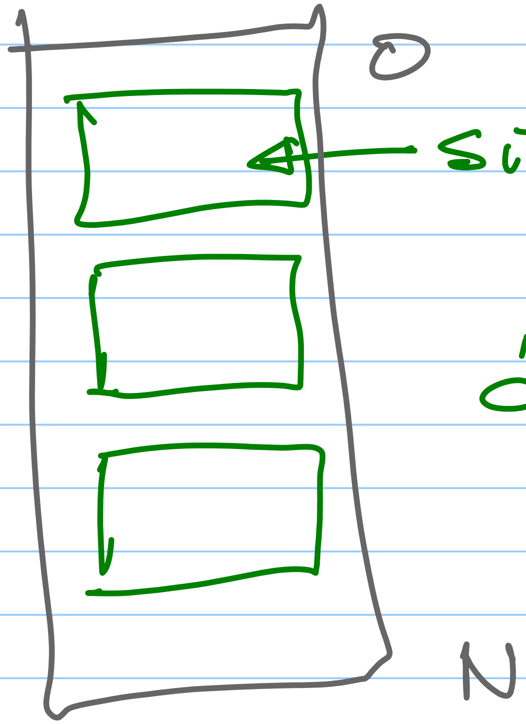
Fall 2016

Note Title

Topics

- Intro / History etc
- Protection, System Calls
- Scheduling - CPU
- Processes / Threads / Race Cond
- Semaphores — Program
  — Implement
- Monitors, pthreads → mem mgt

Physical Memory mngmnt



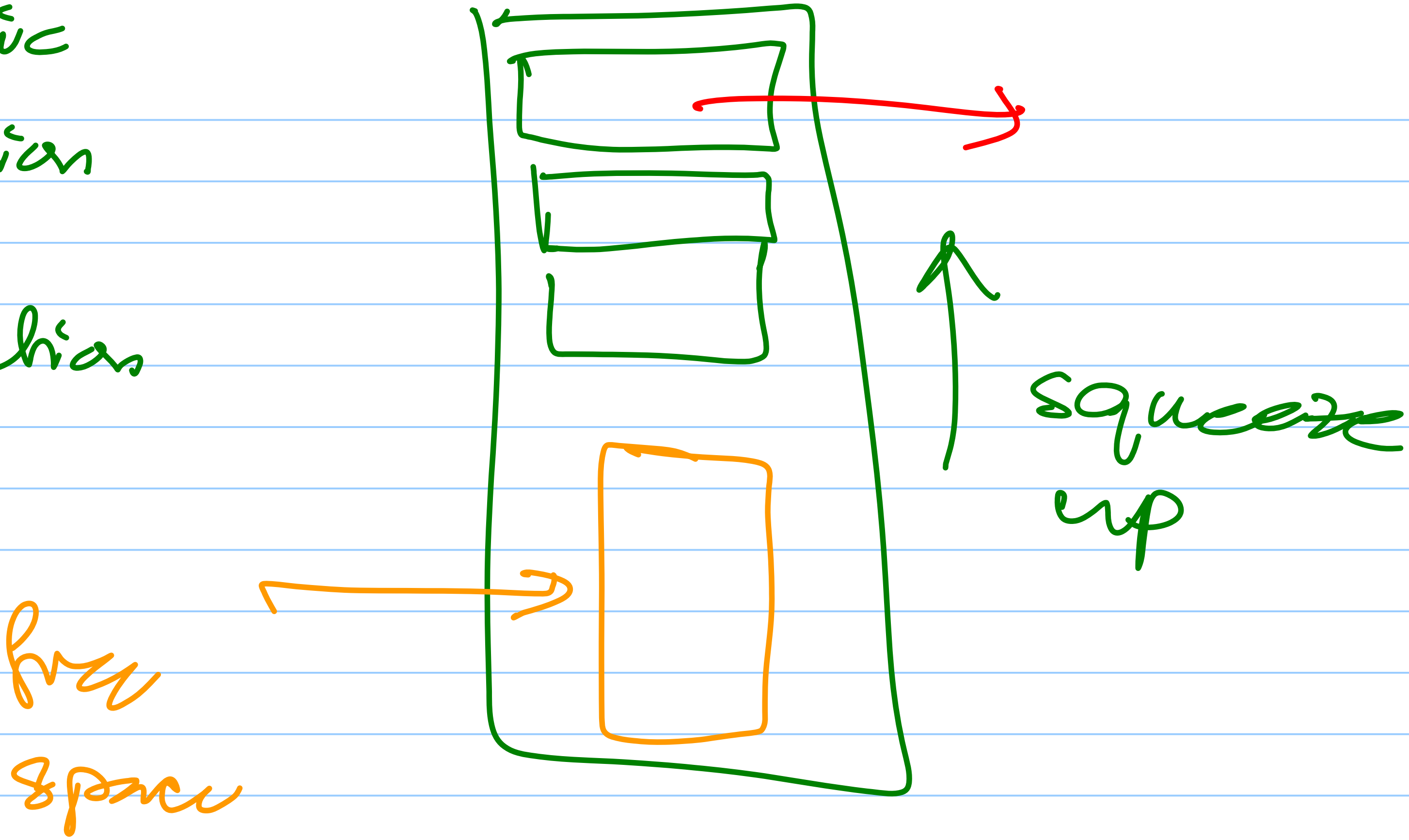
size varies
(small to large)

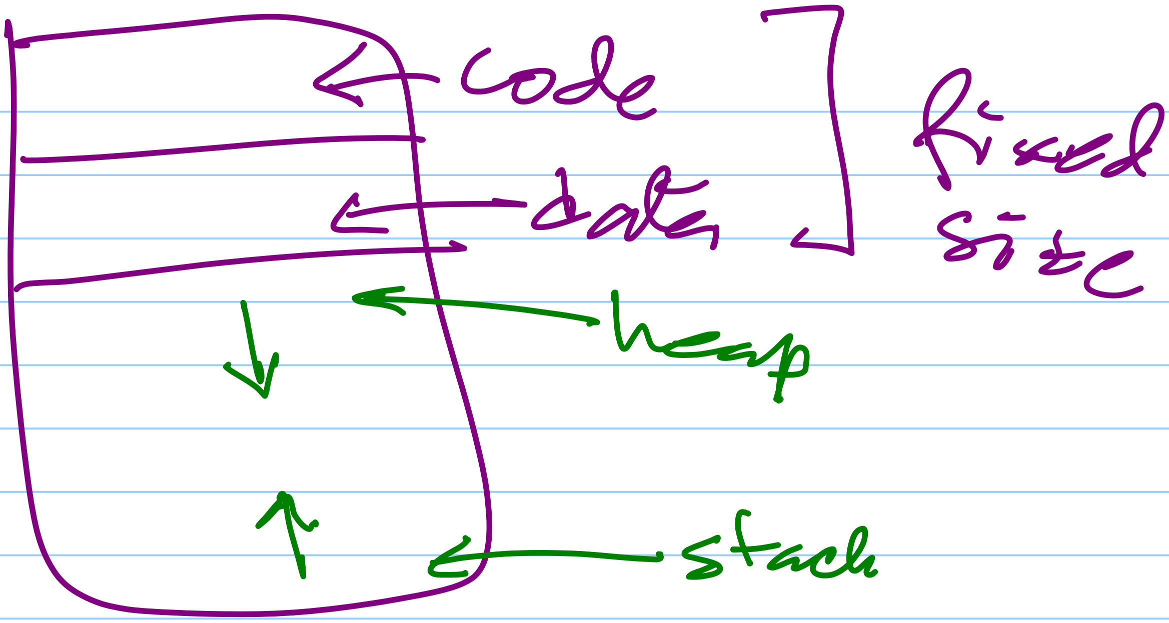
dynamically
loaded &
unloaded

dynamic relocation

+

compaction





heap

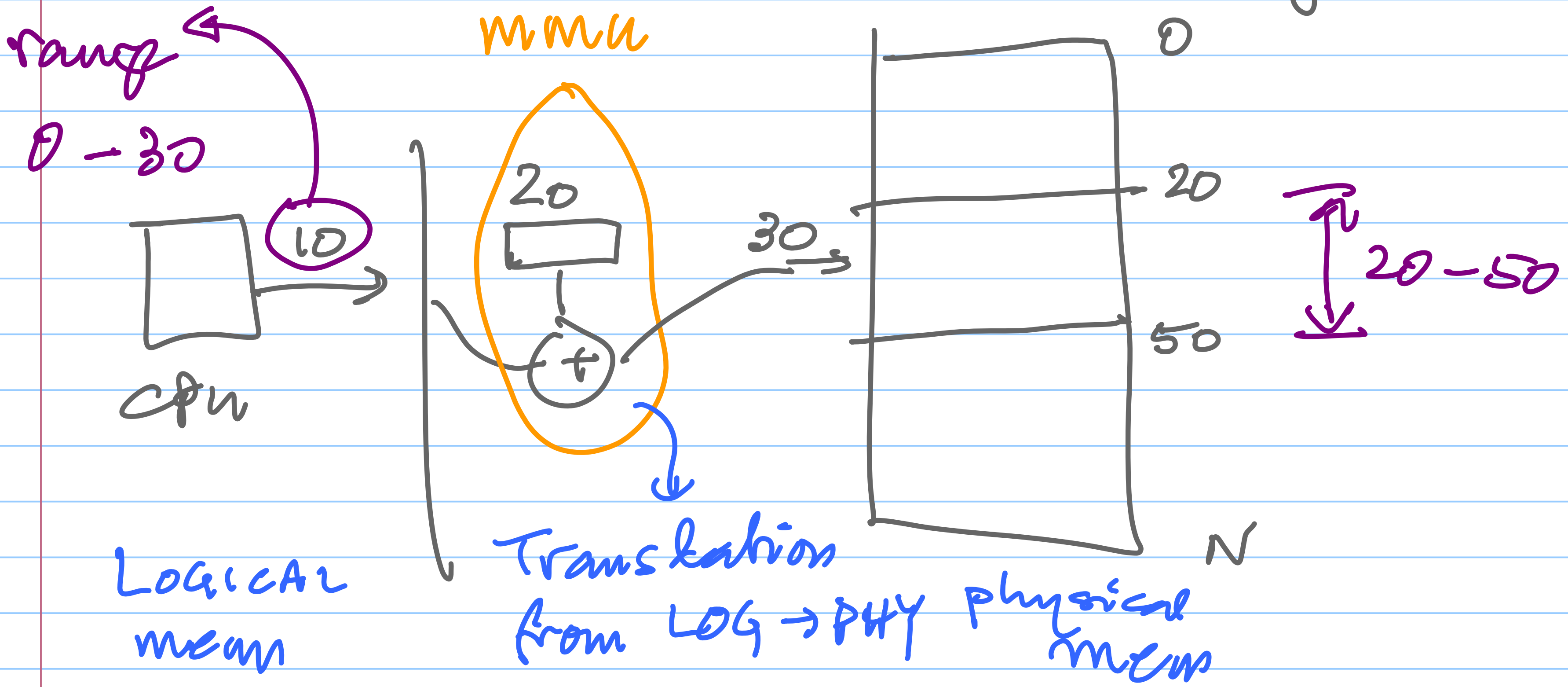
stack

grows & shrinks

Cannot do

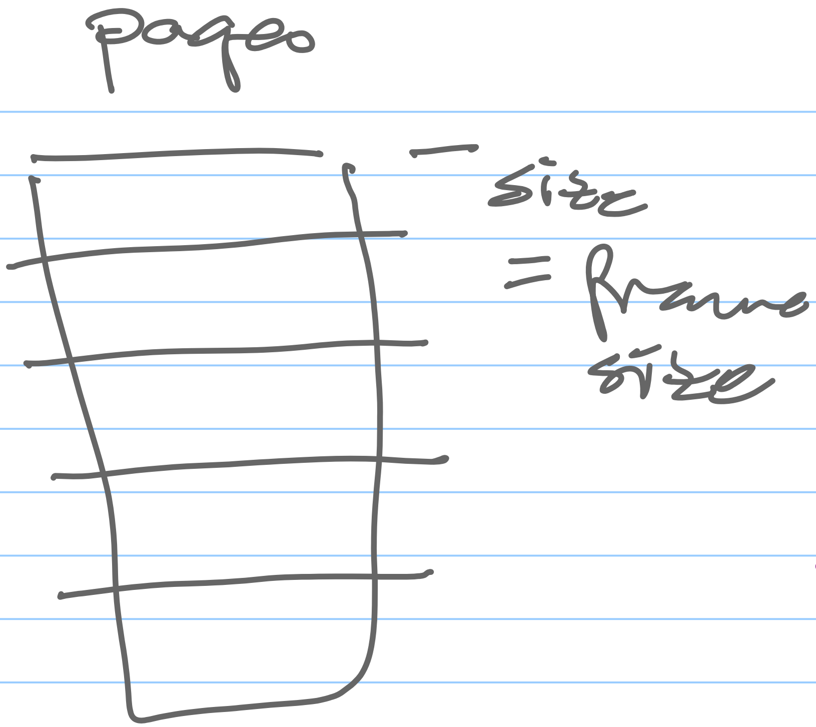
- run programs larger than RAM
- selectively protect memory areas
- share code/data between processes

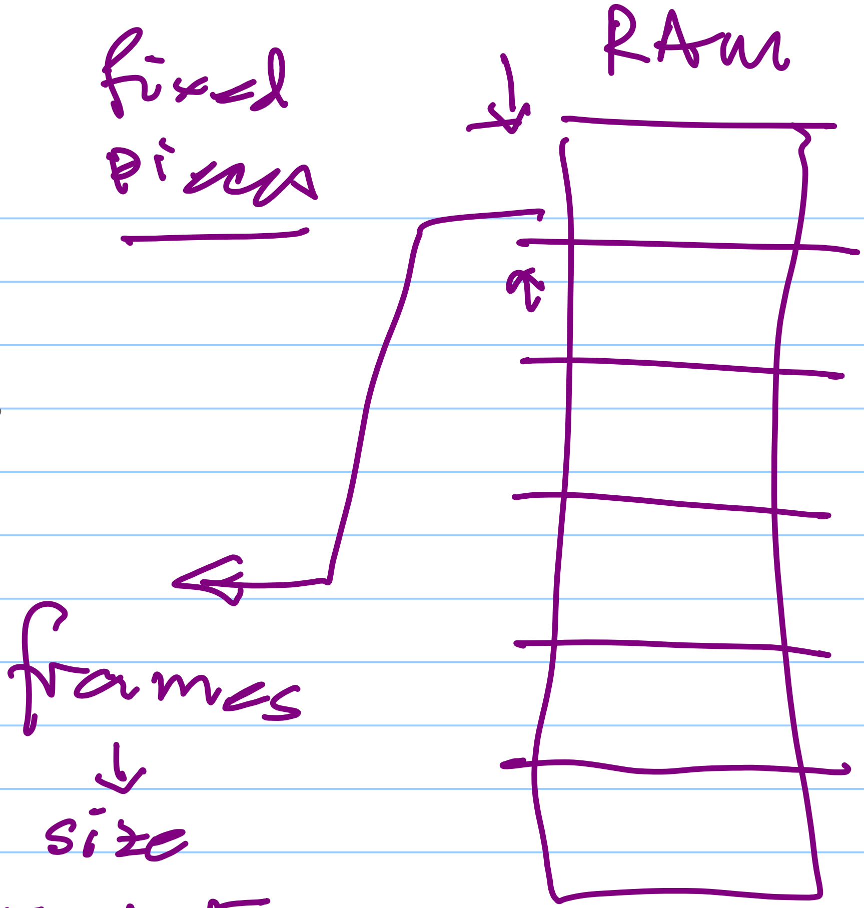
logical memory vs physical memory



physical memory problems

- → need for contigous chunks (variable size)
- → " " moving memory



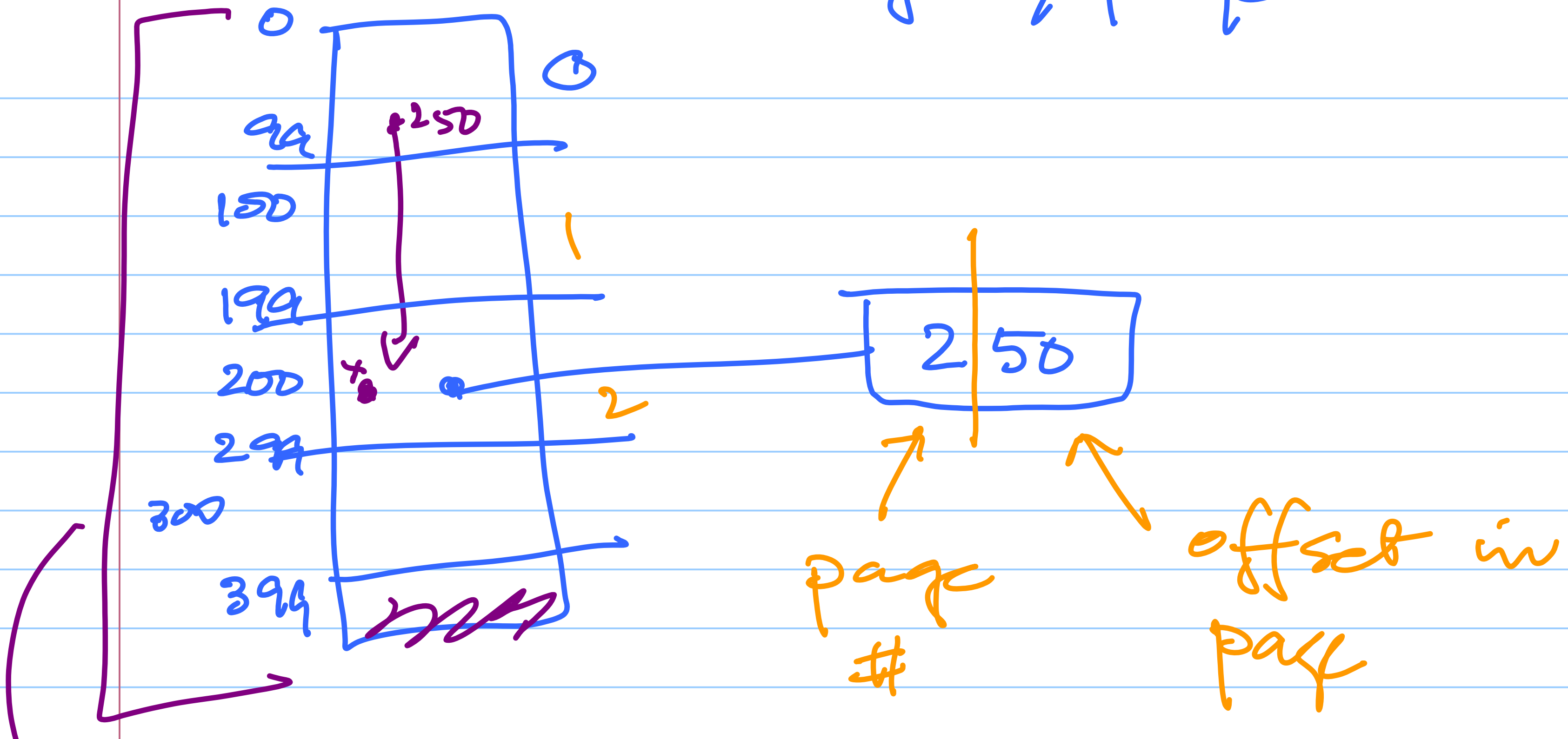


512 bytes
4k "

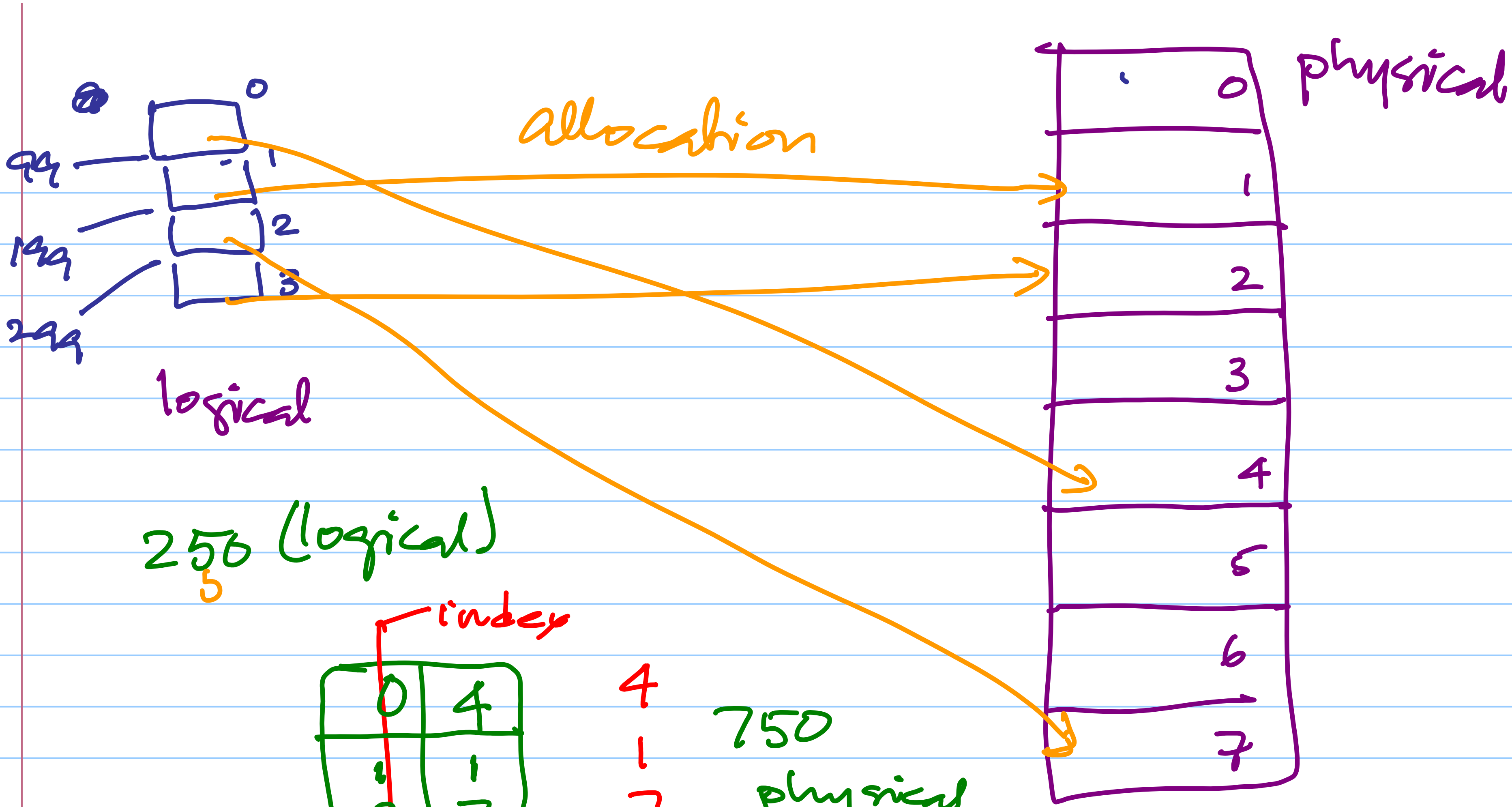
Any page can be put into any frame

PAGING

100 bytes / page



Logical address → 0 to 399



250 (logical)

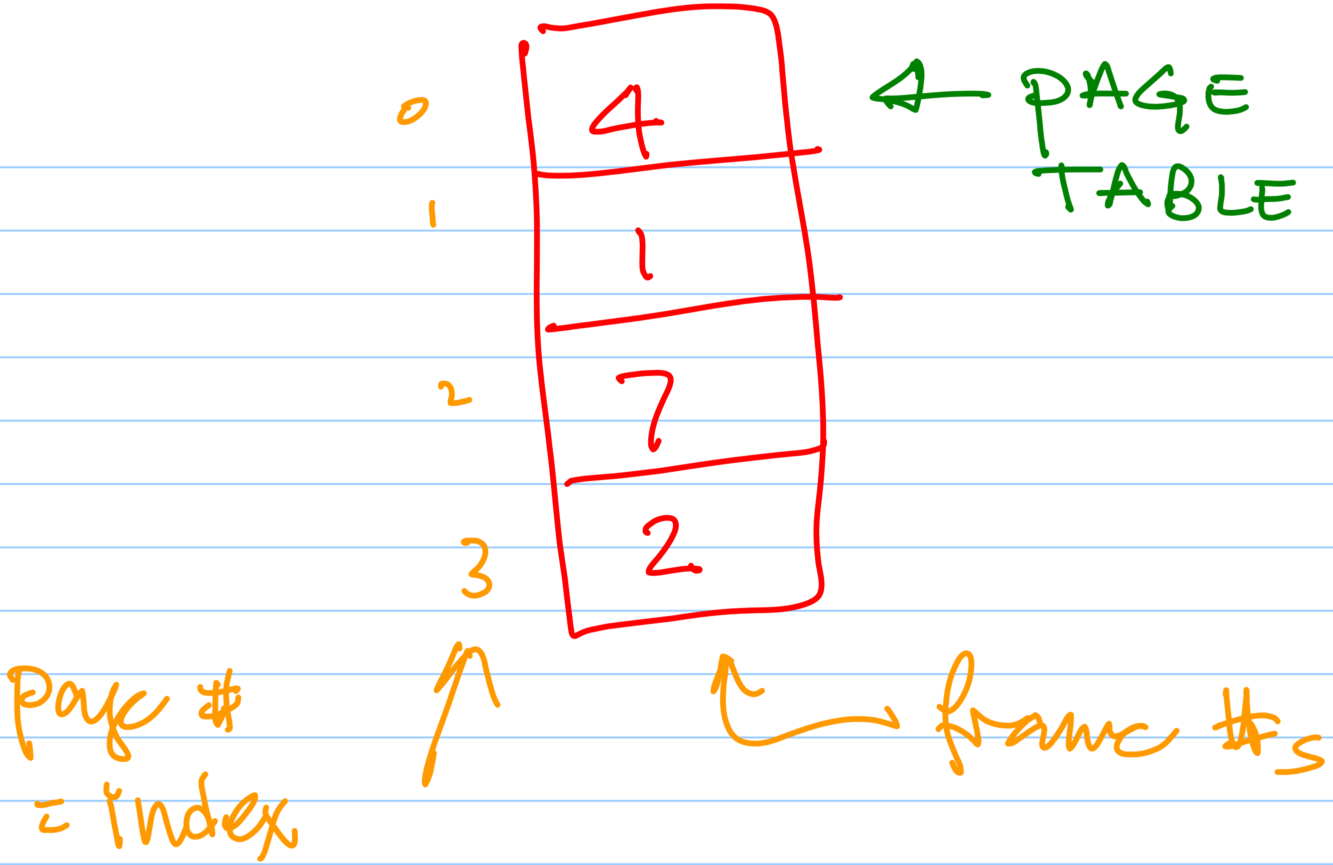
5

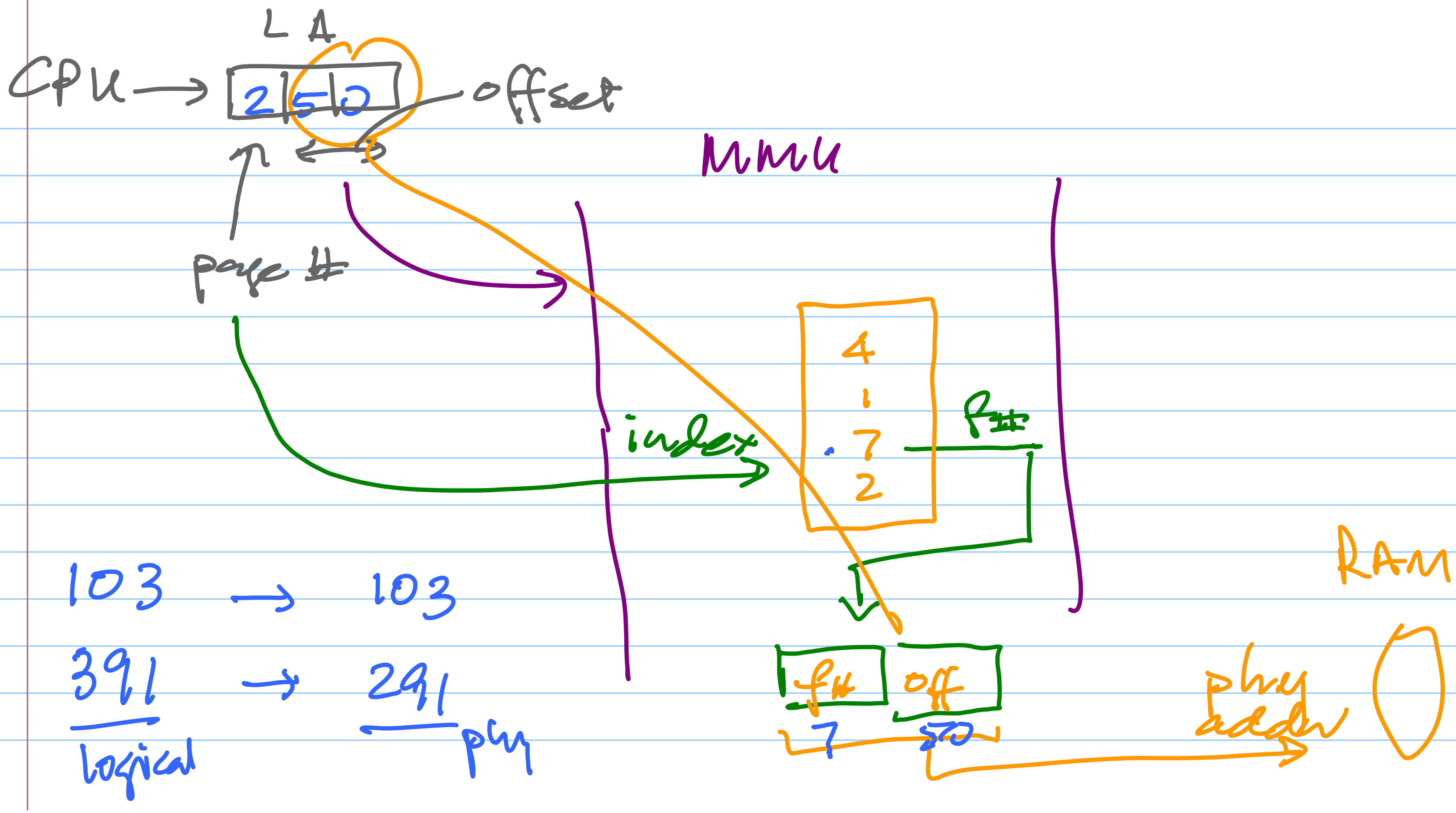
index

0	4
1	1
2	7
3	2

4
1
7
2

750 physical

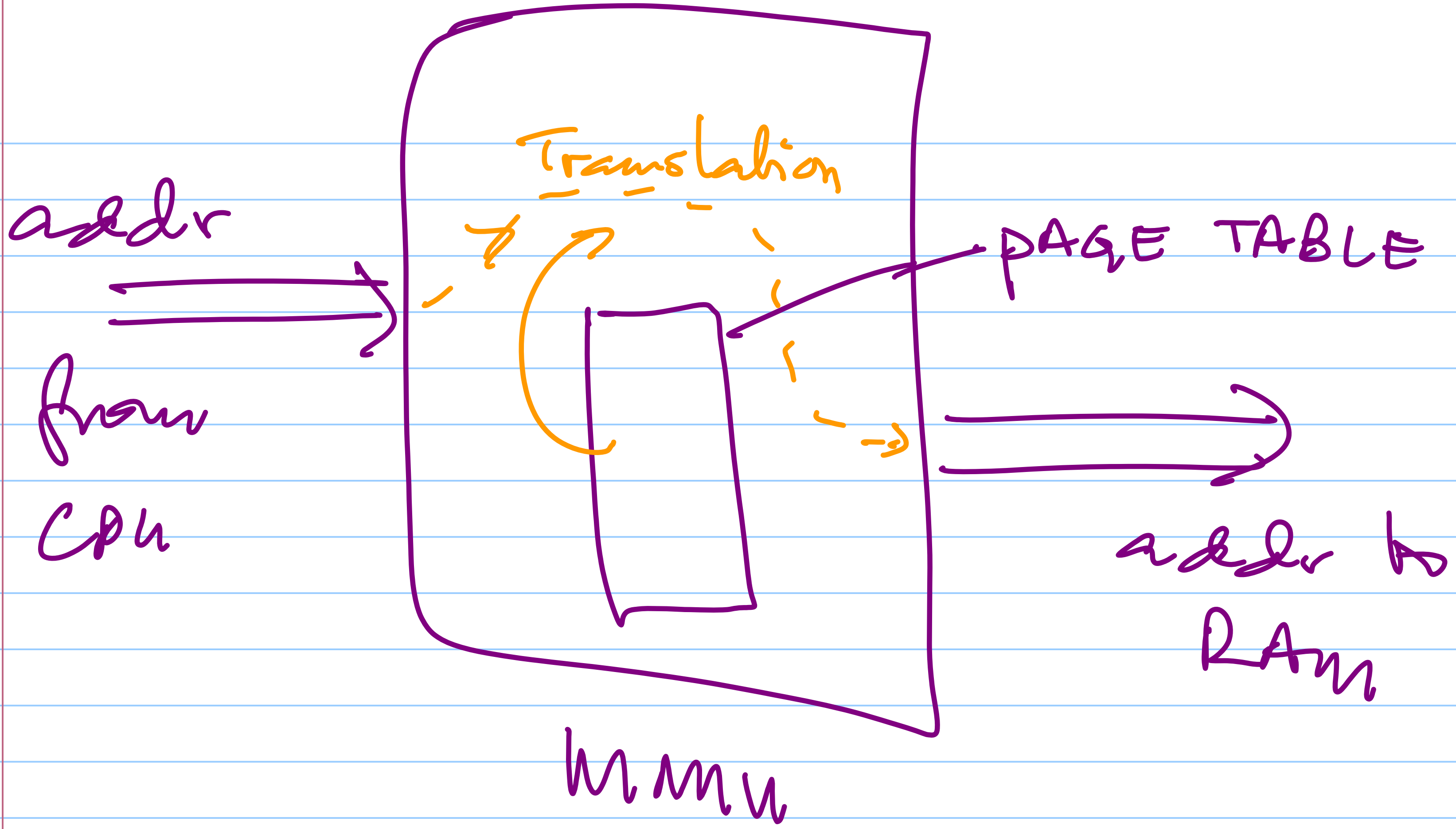

PAGE TABLE
0
1
2
3
4
1
7
2
Page #
= index
frame #s



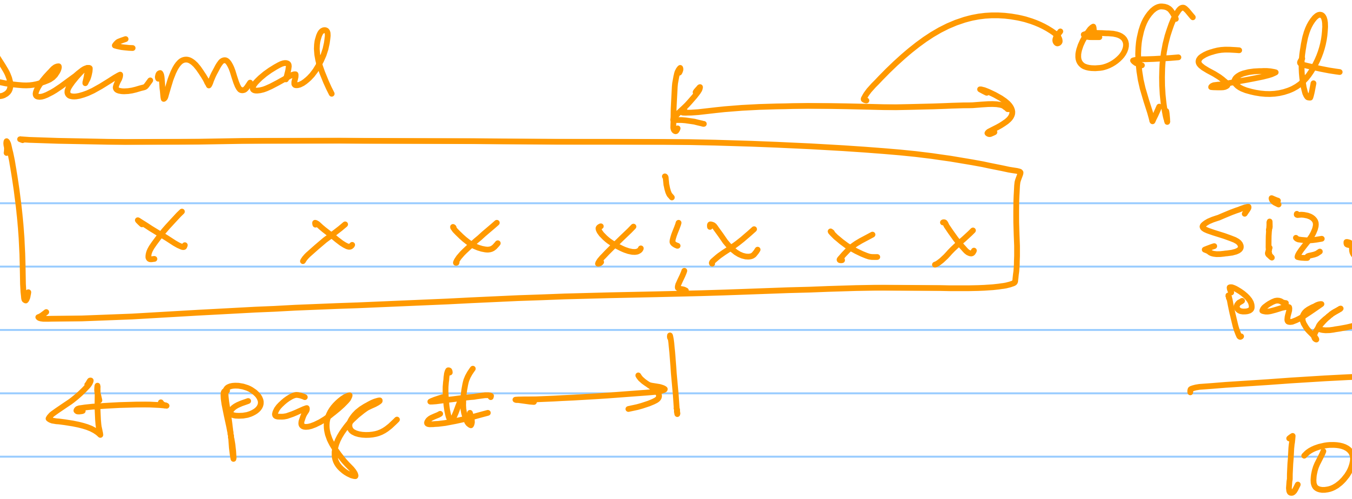
103 → 103

391 → 291

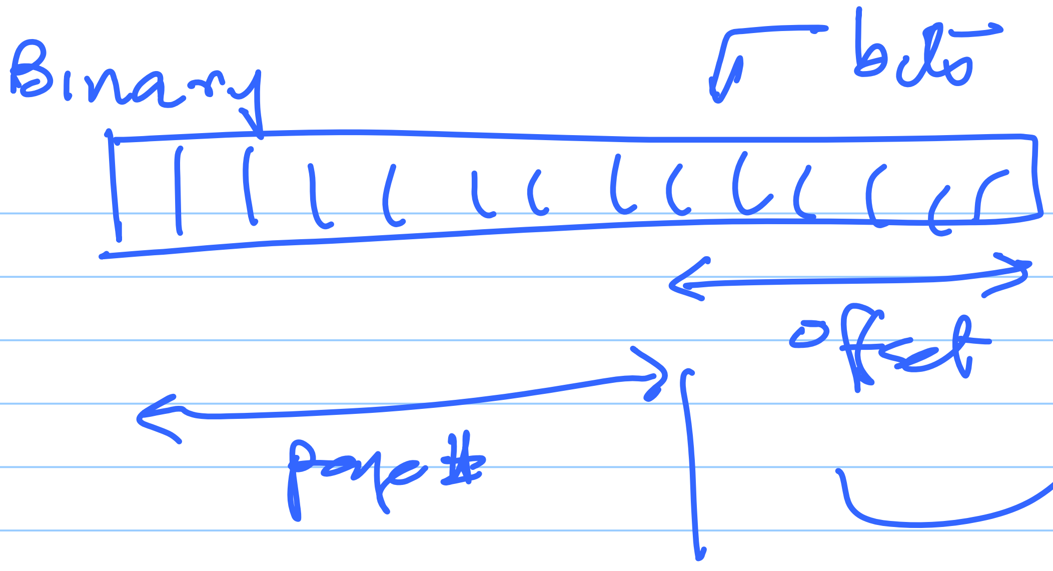
logical phy


addr
from
CPU
Translation
PAGE TABLE
addr to
RAM
MMU

Decimal



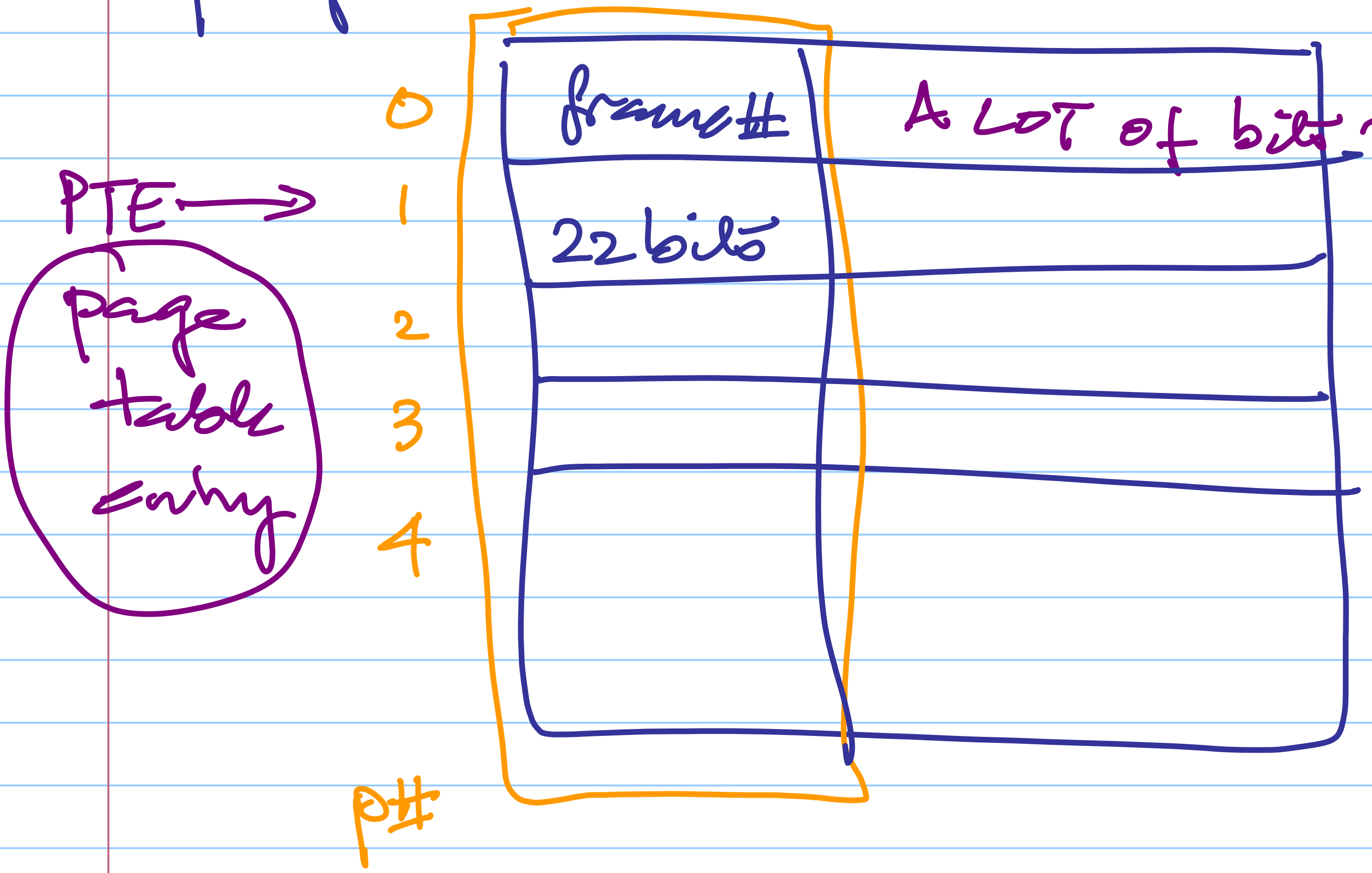
size page	# of digits
10	1
100	2
1000	3
10000	4



Size of page	# of bits in offset
64	6
128	7
256	8
512	9
1024	10

page table design

32 bits addr



VALID
MOD
REF
R-perm
W-perm
X- "
counters
disk addr

LA to PA translation

- get p# & offset from LA
- look up page table entry with p# as offset
- get f#
- concat f# & offset → PA

PAGING

→ all pages in mem

→ MMU does translation

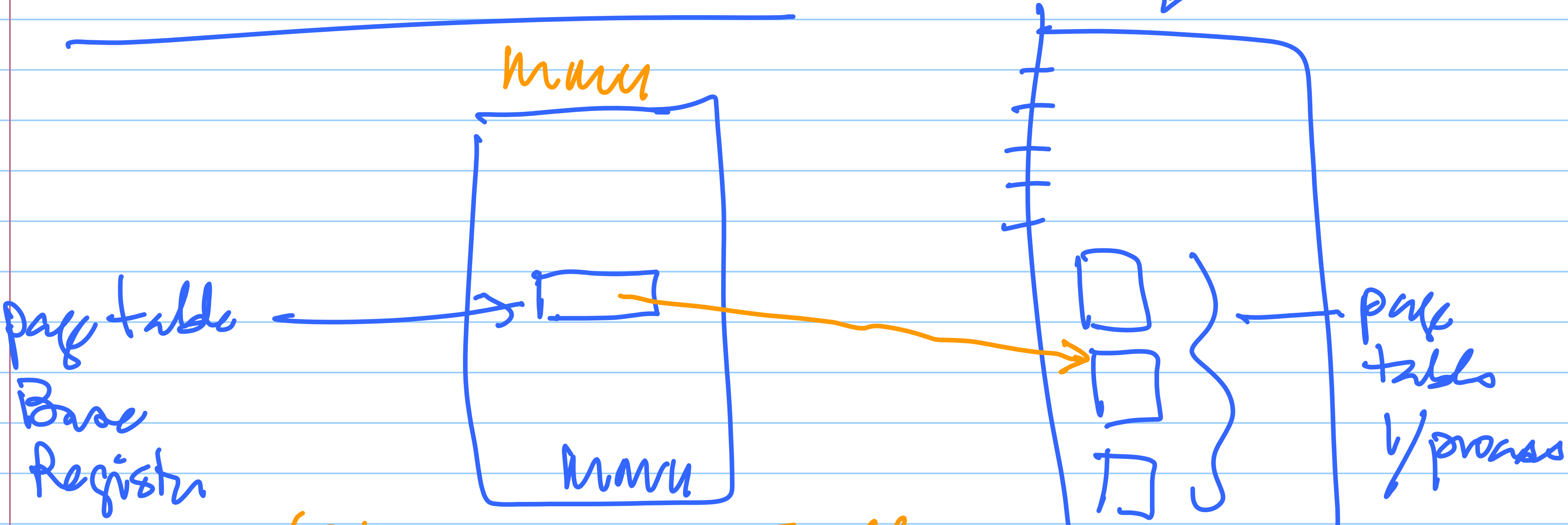
→ where is the page table (?)

→ every process has unique page table (private)

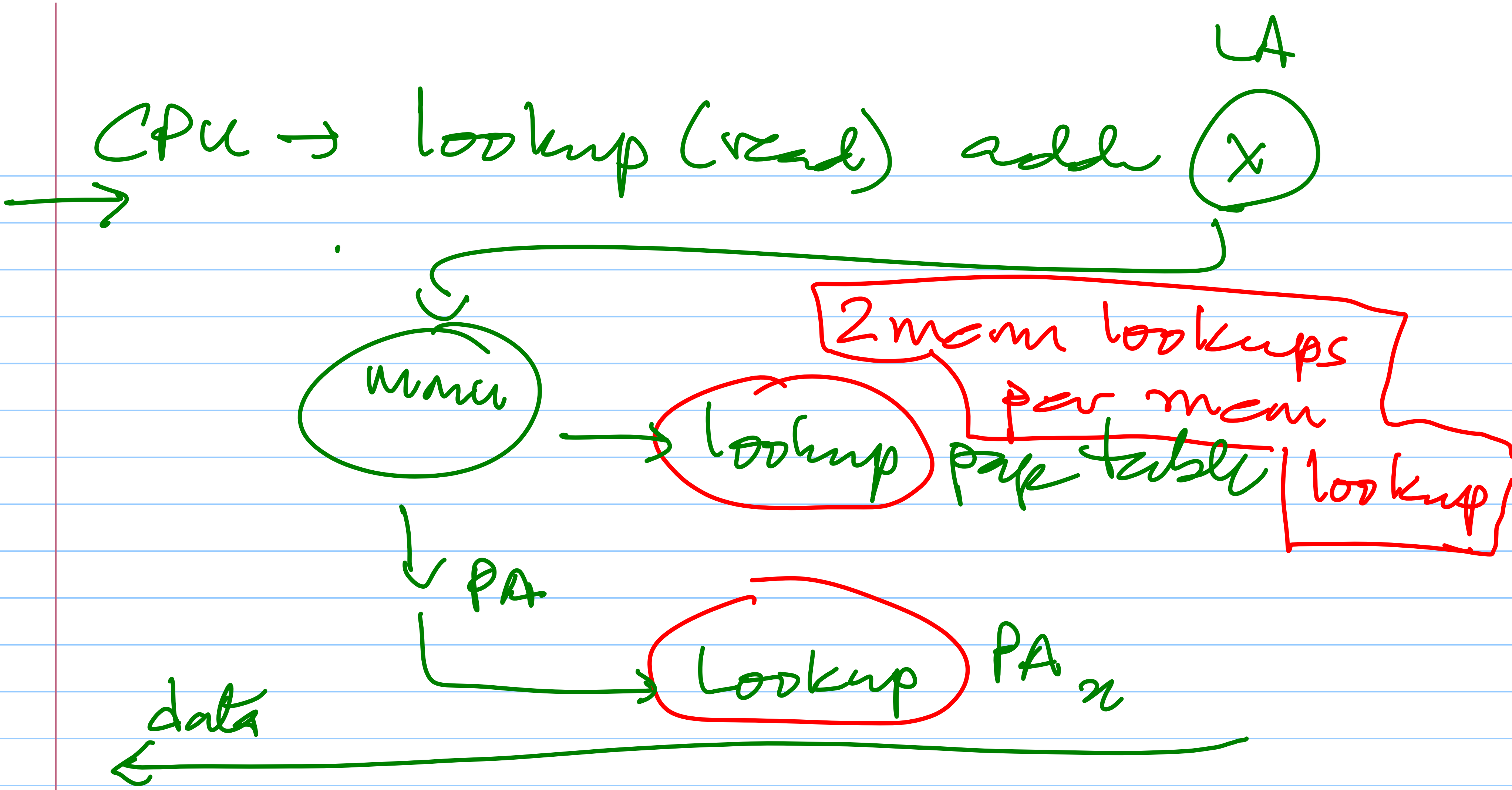
→ how big ? → max 2^{22} → 4M x size

=> TOO BIG

PAGE Tables are stored in Kernel Memory



(P# + PTBR) → PTE addr → LOOKup.


CPU → lookup (read) addr x
LA
MMU
lookup page table
lookup
PA
Lookup
PA x
data
2 mem lookups per mem
lookup!