CSE 330: Operating Systems

Class: 17 **Date:** 3/15

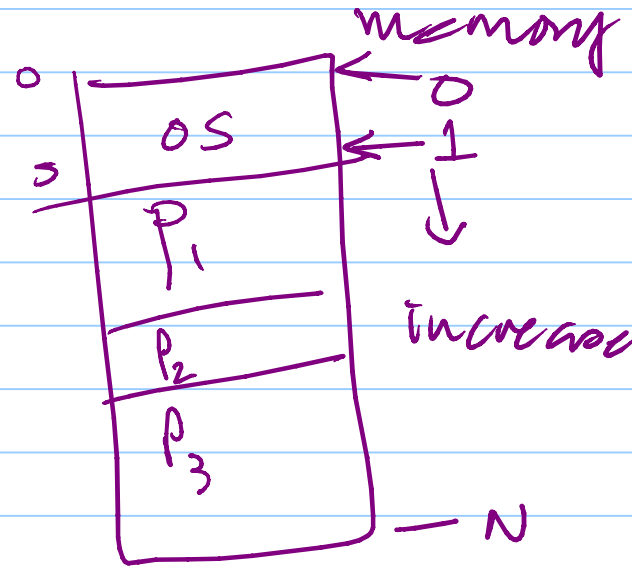
Spring 2018

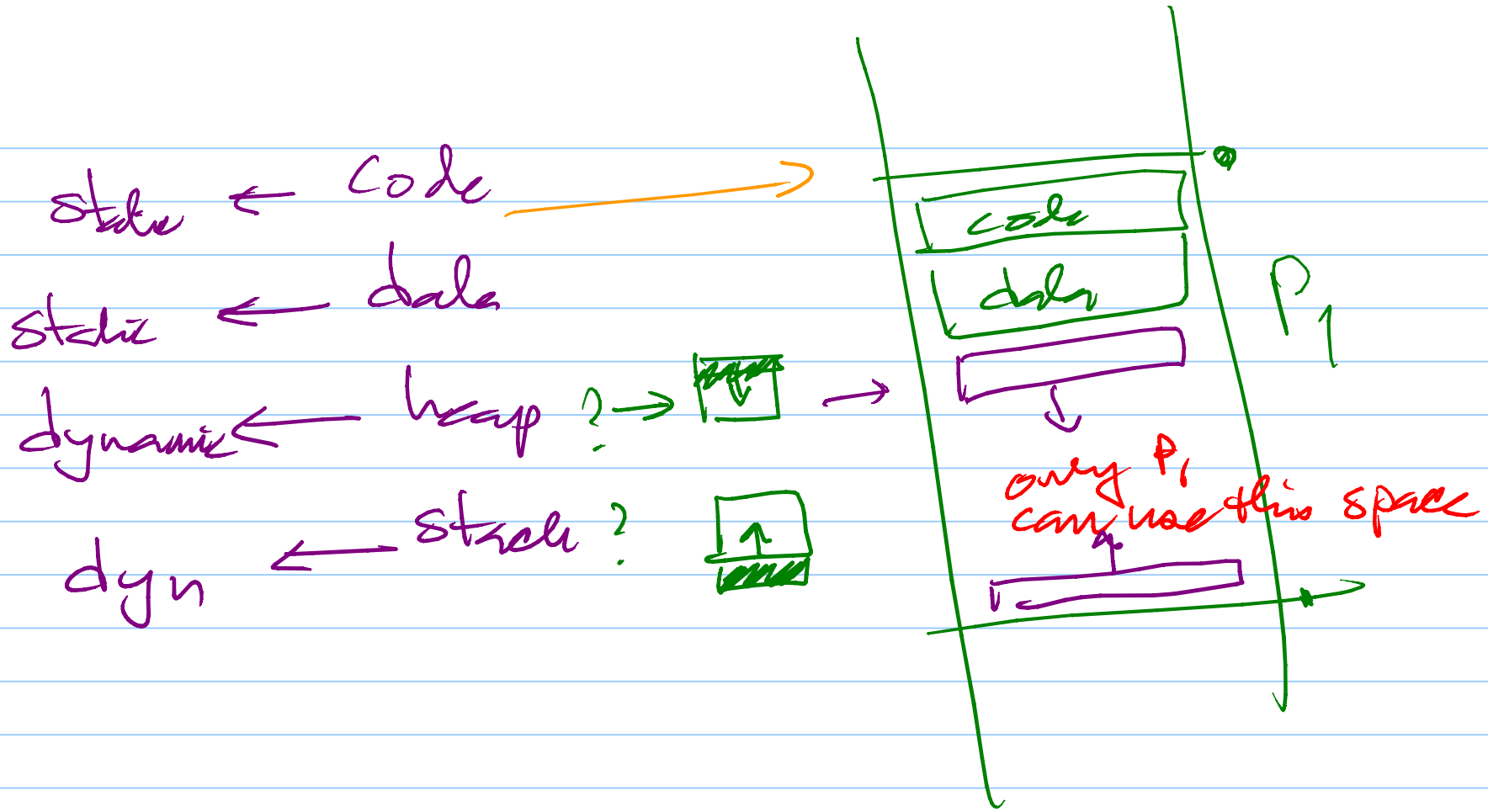
Note Title

Memory Management

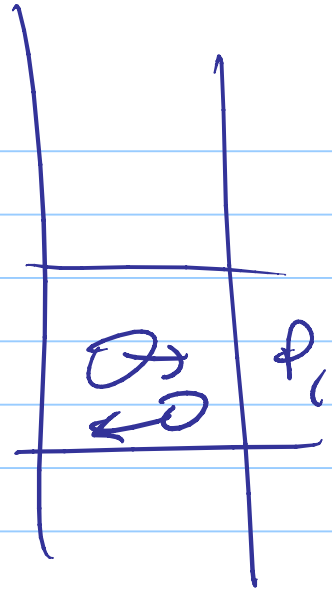
- physical memory systems

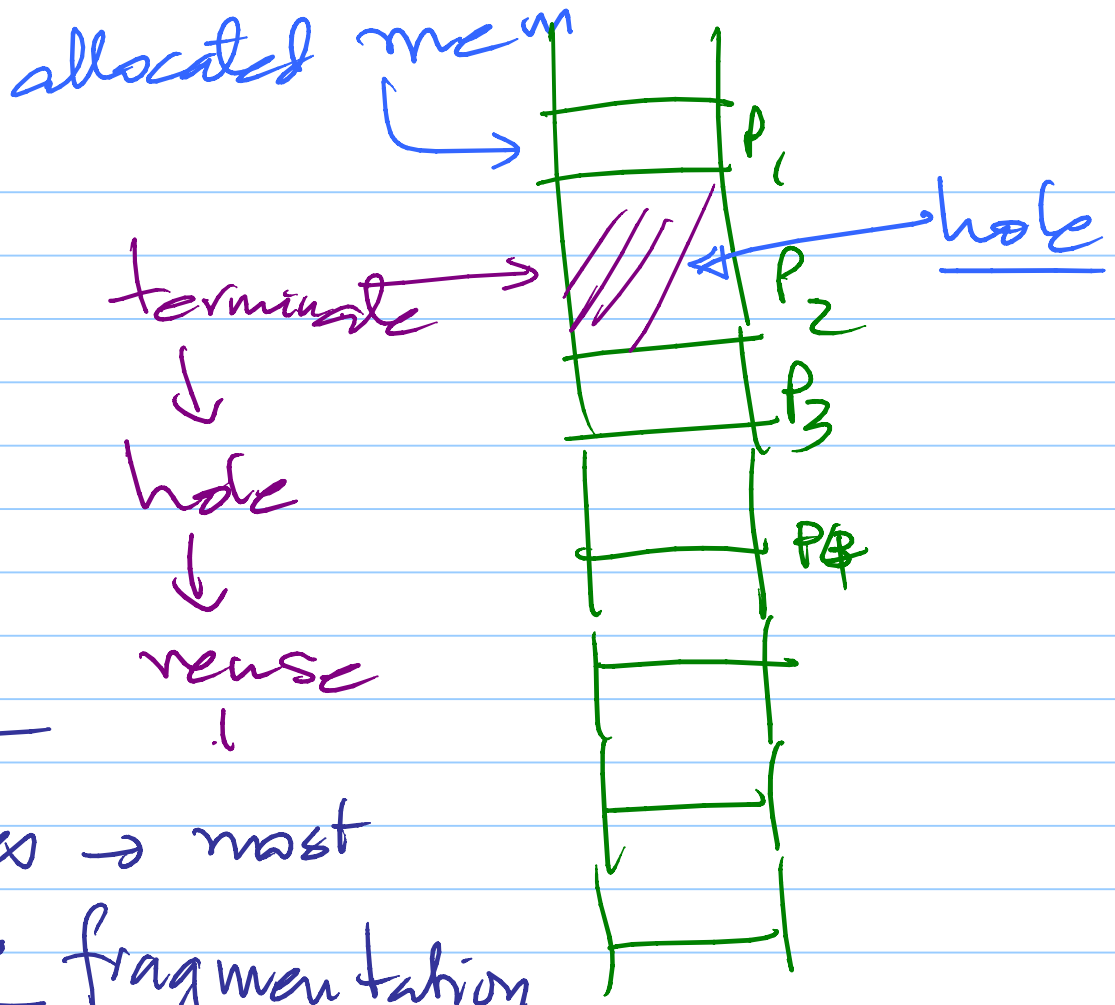
bytes
words
} addresses




static ← code
static ← data
dynamic ← heap ?
dyn ← stack ?
code
data
P1
every P1 can use this space

- program / code / data etc size
- dynamic loading (DLL)
- stack / heap
- pre-allocate? → wastes memory
  → if mem too small process cannot continue
  ↓
  causes internal fragmentation





new process → put in which hole?

terminate → hole → reuse

too many holes → most

too small → fragmentation

fragmentation

- → external (holes that are small)
- → internal (allocated to process but not used)

Wasted memory

① Compaction → [move or squish processes together]

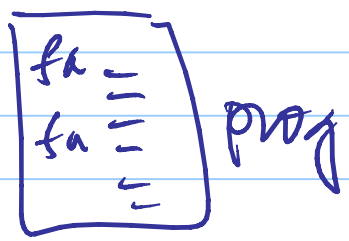
↳ impossible in absolute address code

or bit slow in indirect addressing (relative)

② Swapping → move things in & out of disk to make space

Code, data, stack, heap

↳ have to be contiguous



paging systems & virtual memory systems

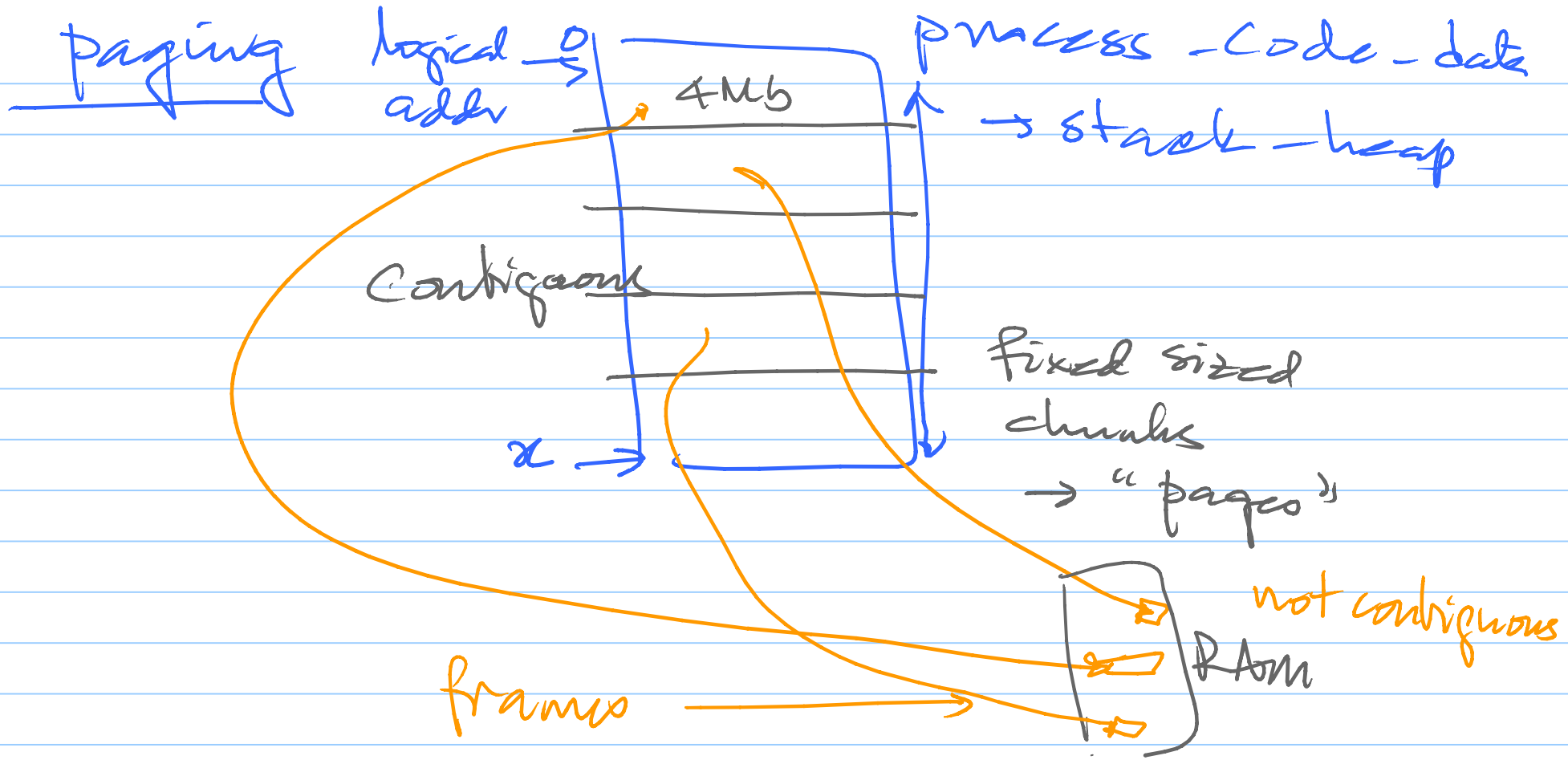
→ simple paging systems

→ real paging systems (the VAX architecture)

↳ applications are allocated larger memory than possible

→ use of logical & physical addresses

CPU RAM


paging
logical addr
0
4MB
process - Code - data
→ stack - heap
Contiguous
fixed sized
chunks
→ "pages"
x
not contiguous
RAM
frames

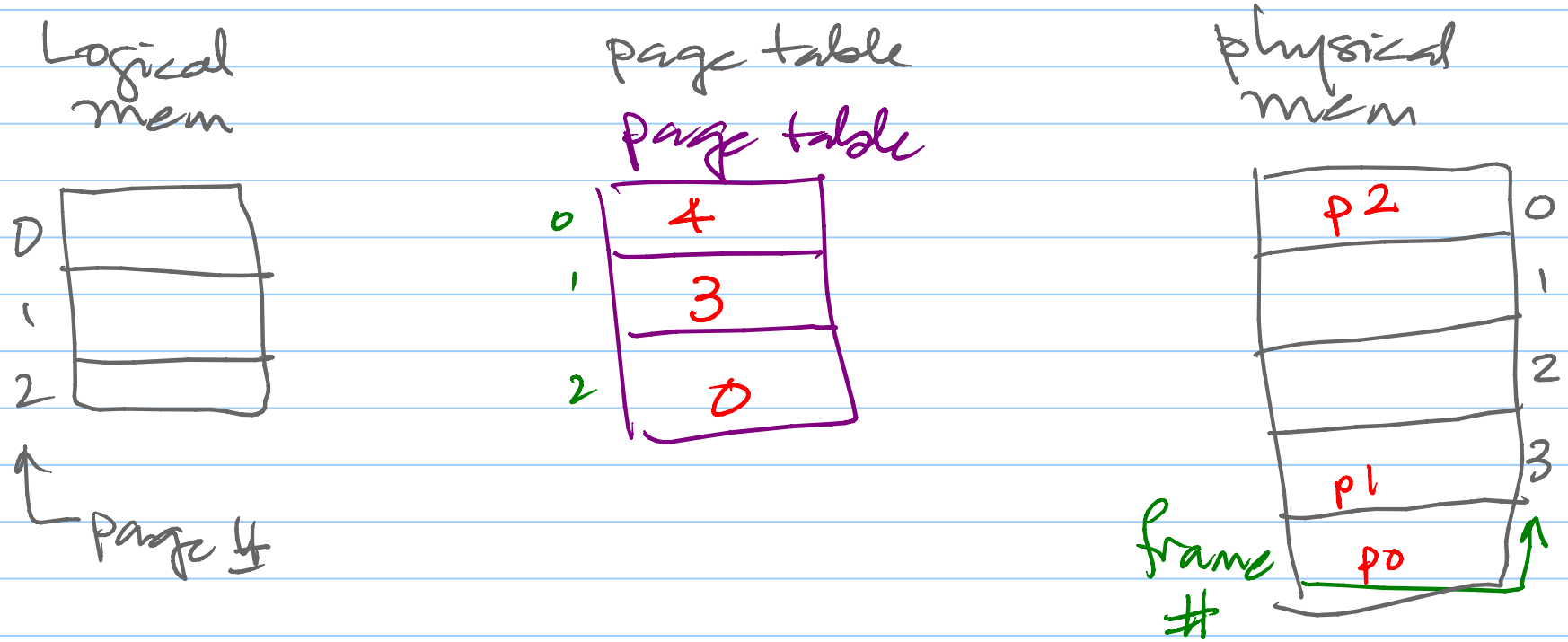
Paging systems

→ no external fragmentation

→ minimal internal fragmentation

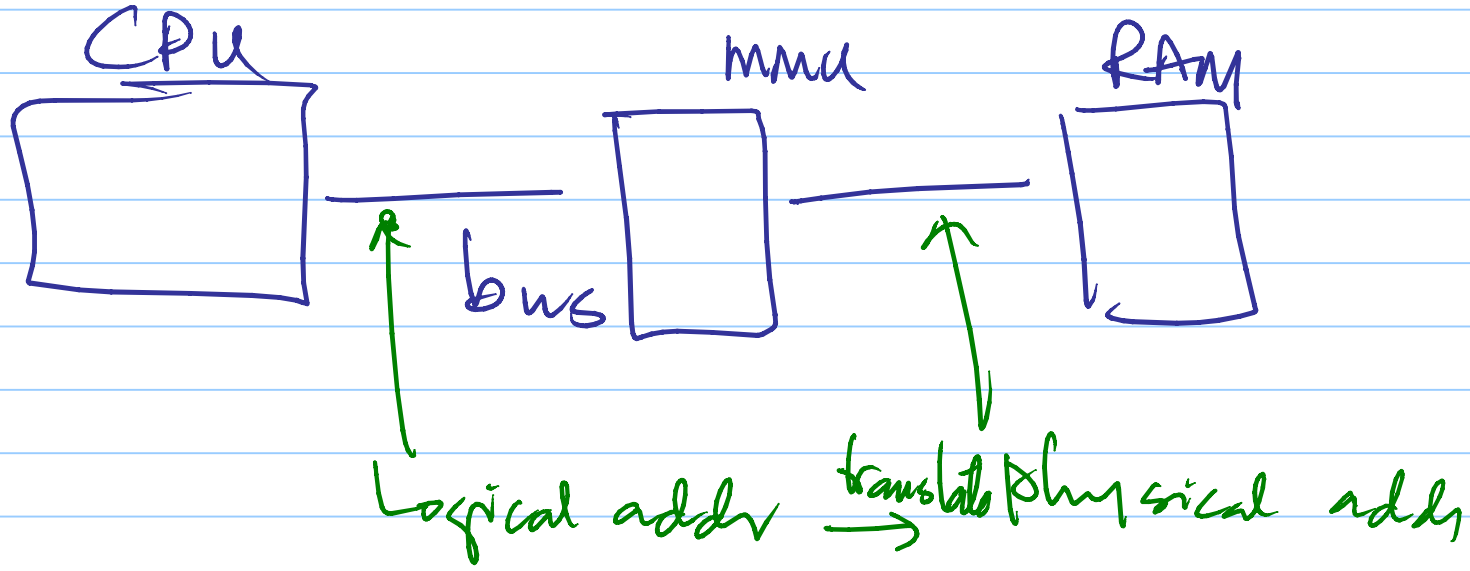
↳ ½ page per process

When a process is started a "page table" is created by the OS



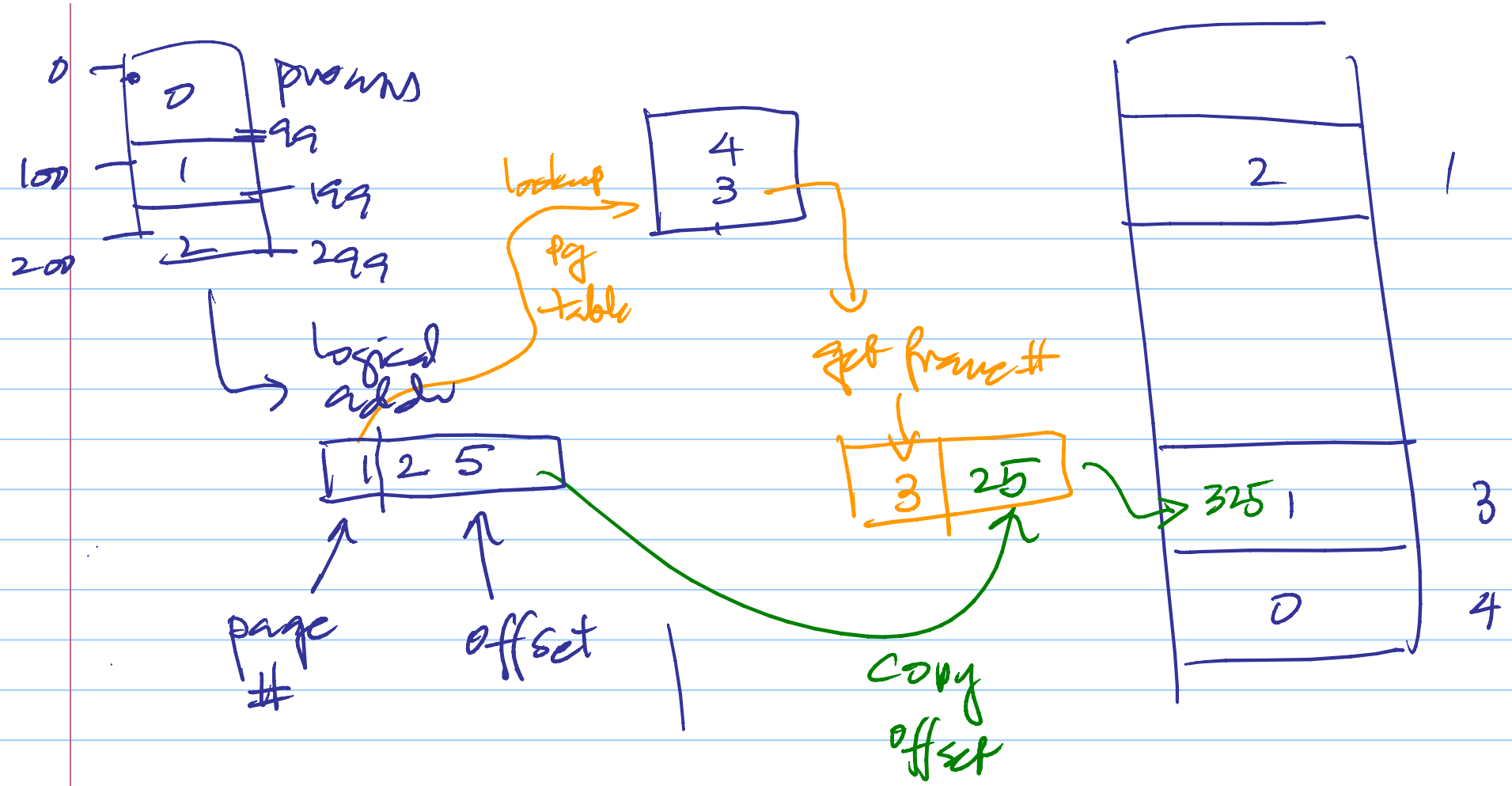
Address translation

- done by mmu → mem mngmt unit

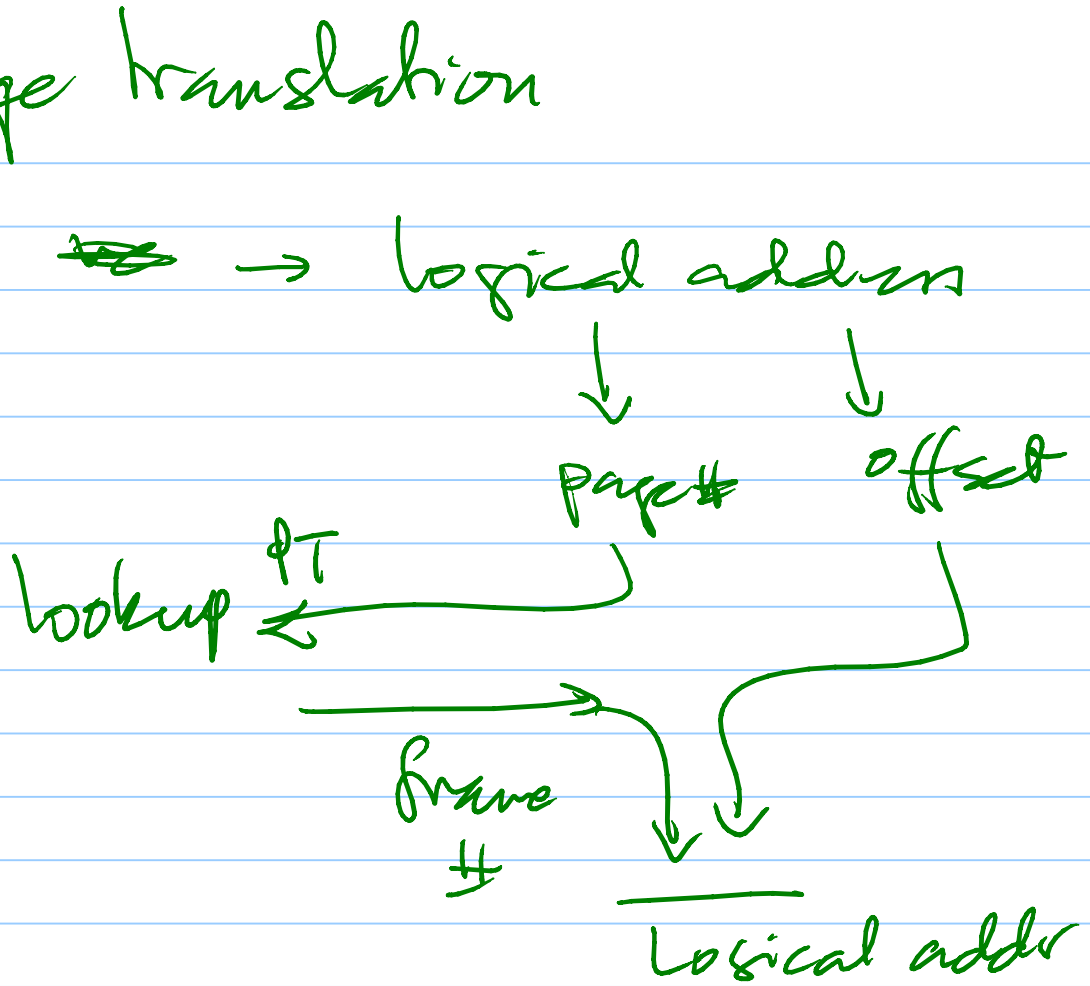


Architecture

- word addressible
- 100 words/page


0
100
200
0
1
2
99
199
299
process
lookup
pg table
4
3
Logical address
1 2 5
page #
offset
get frame #
3 25
copy
offset
2
1
325
3
0
4

page translation



overhead of paging

① load page table on context switch ?

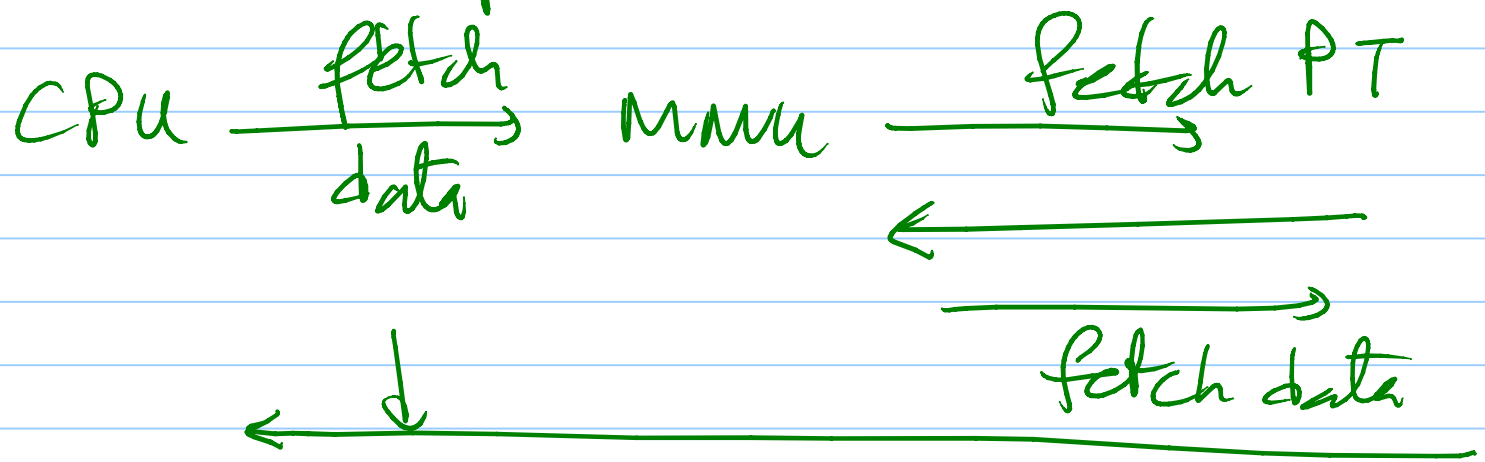
↳ size of Page table

page size = 4 K bytes

$\frac{100\,\text{MB}}{4\text{K}}$ $\frac{100\text{K}\,\text{B}}{4\text{K}}$

25k entries

Store the page table in memory & get the MMU fetch it from memory



TLB

Translation Lookaside buffer