CS 744 Lecture 12

* Lab 3 is active

⊛ memory virtualization, the address space abstraction

every process has a 0-starting linear contiguous till max addr. } memory region.

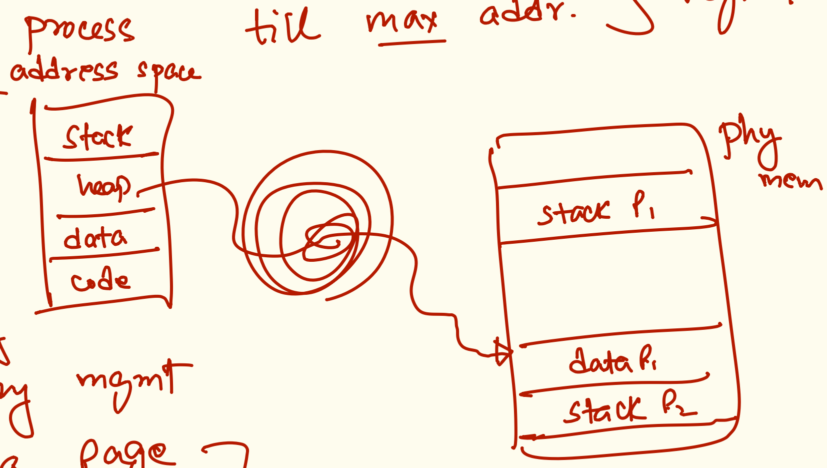
base & bounds based mapping.

↓ segmentation

↓ paging

use changes the granularity of all memory mgmt to be that of a page

page — seq. of contiguous finite sized addresses



segmentation

VA: 32-bits

segment identifiers | offset in the segment

$2^2 \times 2^{30} \sim 2^{32}$

segment table — 4 entries

paging

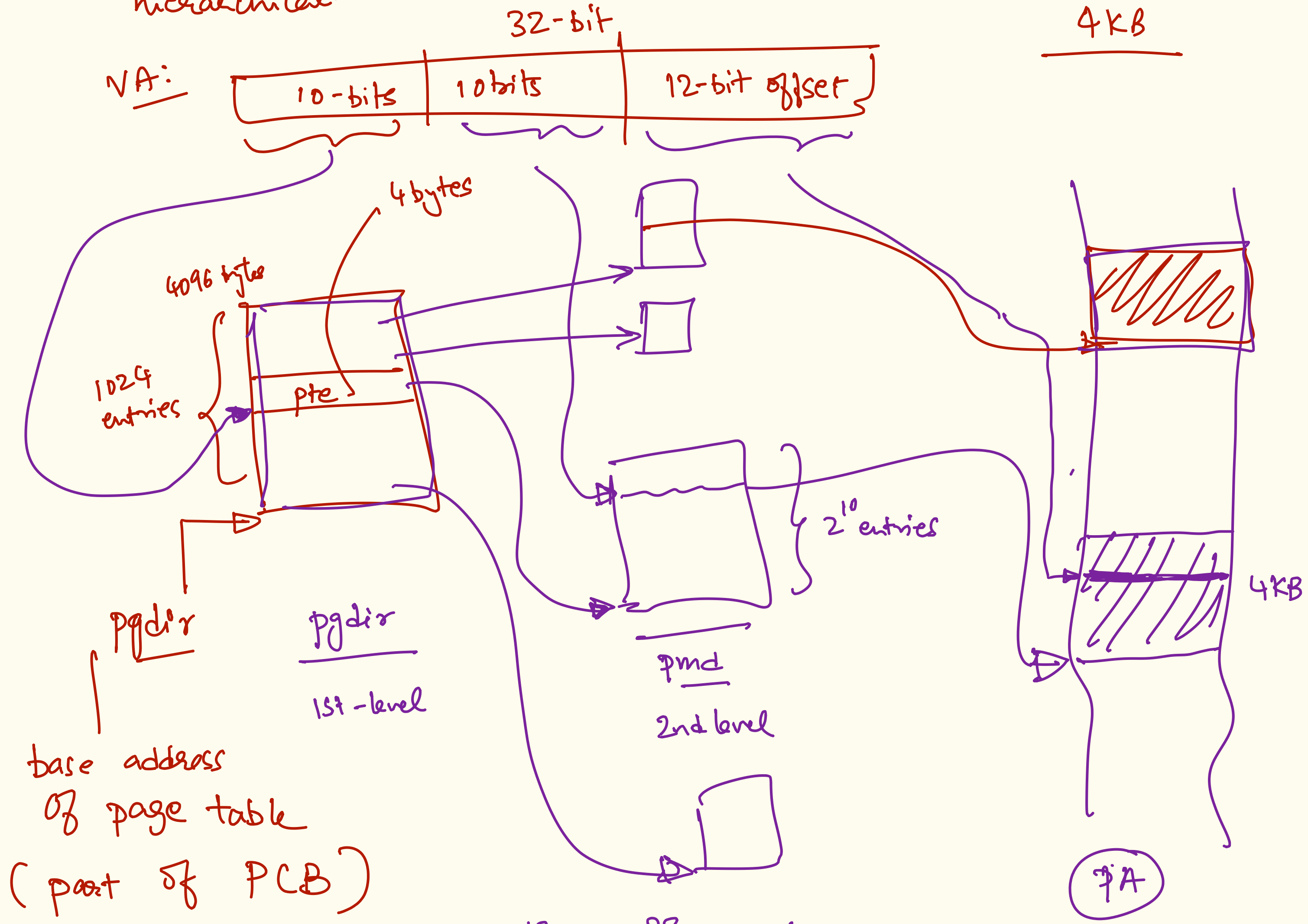
VA: 20-bits | 12 bits offset

index ↳ page table — needs 2^{20} entries (contiguous).

⊛ multi-level page tables

hierarchical.

page size 4KB



$2^{10} \times 2^{10} \times 2^{12} = \underline{2^{32}}$ bytes

HW

redo above w/ pte of size 8bytes.

⊛ H/w assistance of paging

(i) CPU mode to enable / disable paging — x86 ~ CR0

(ii) register to store pgdir — CR3 ~ phy. address of the pagetable directory

(iii) MMU which can lookup the page table for mapping / translation

(*) p = malloc (sizeof(int) * 4096); — a call indicating to the OS about processes <u>intent</u> to use addresses / region of it's address space.

— returns a virtual address to process (start region)

*p = 23;

(i) VA to PA mapping exists?

- mmu can lookup mapping & access the PA for the VA

(ii) no mapping in page table

- page fault (~~sig~~ interrupt) implicit s/w

lazy page alloc.

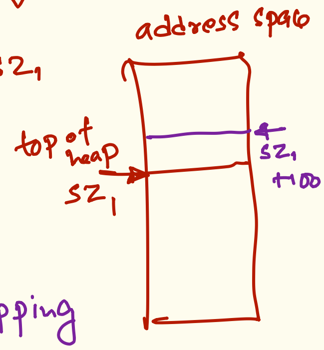
- page fault handler — faulting address (VA) — CR2
  - check is VA is valid
- find (a) free physical page
- add mapping to page table
- return from handler
- re access the VA.

malloc (100) → sbrk — sys call

Kennel.

address space mgmt.
- return val. sz_1
- top = sz_1 + 100

set up page table mapping



(*) pte (what is ?)

4 ~~bytes~~ bytes

20 bits	12 bits
start/base addr. of a page	pte (flags)

info about the page table entry.

R - read
w - written
u - user
P - present
v - valid
e - execute

- accessed & manipulated by MMU & the OS.

(*) demand paging

swapping — P = 0, V = 1