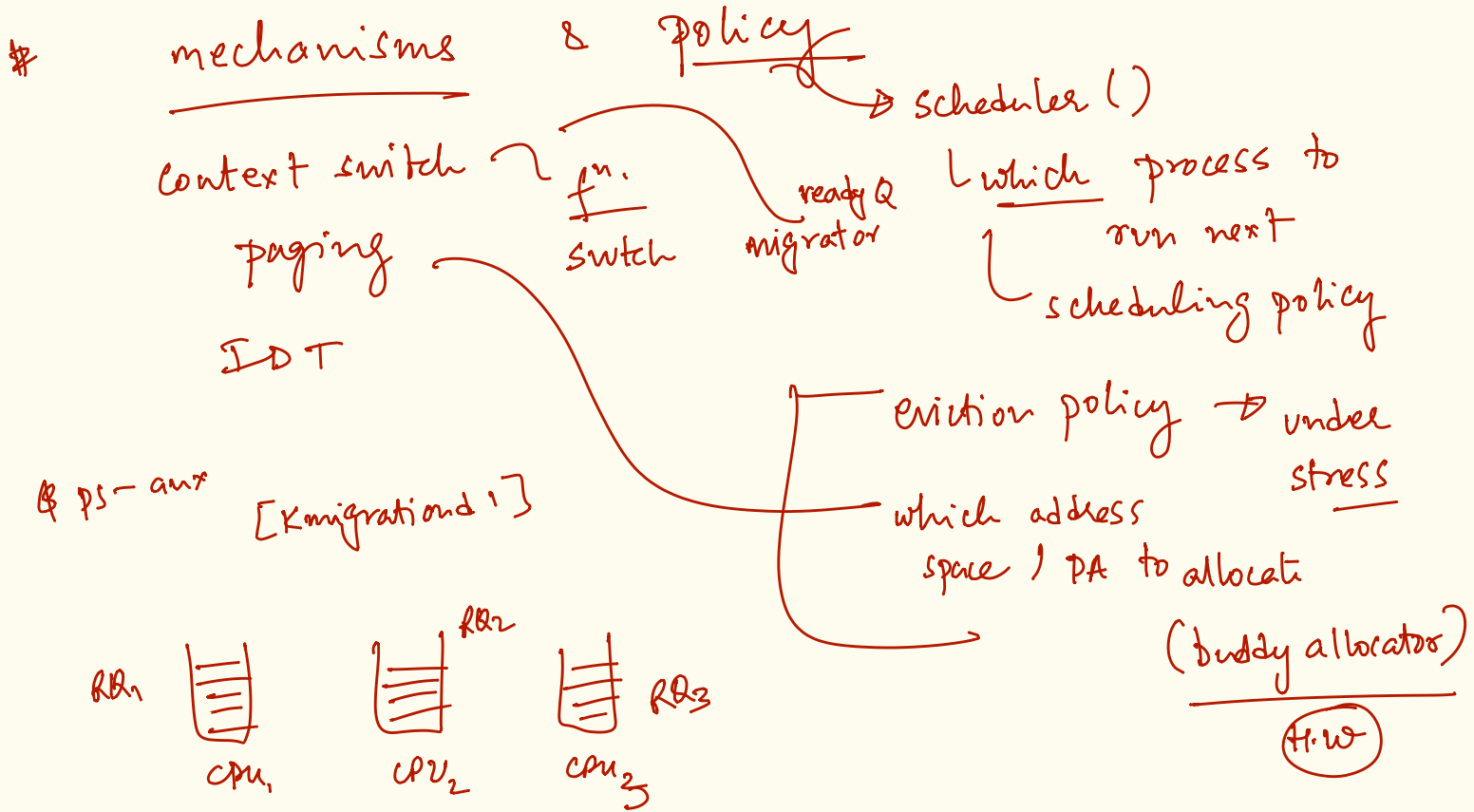


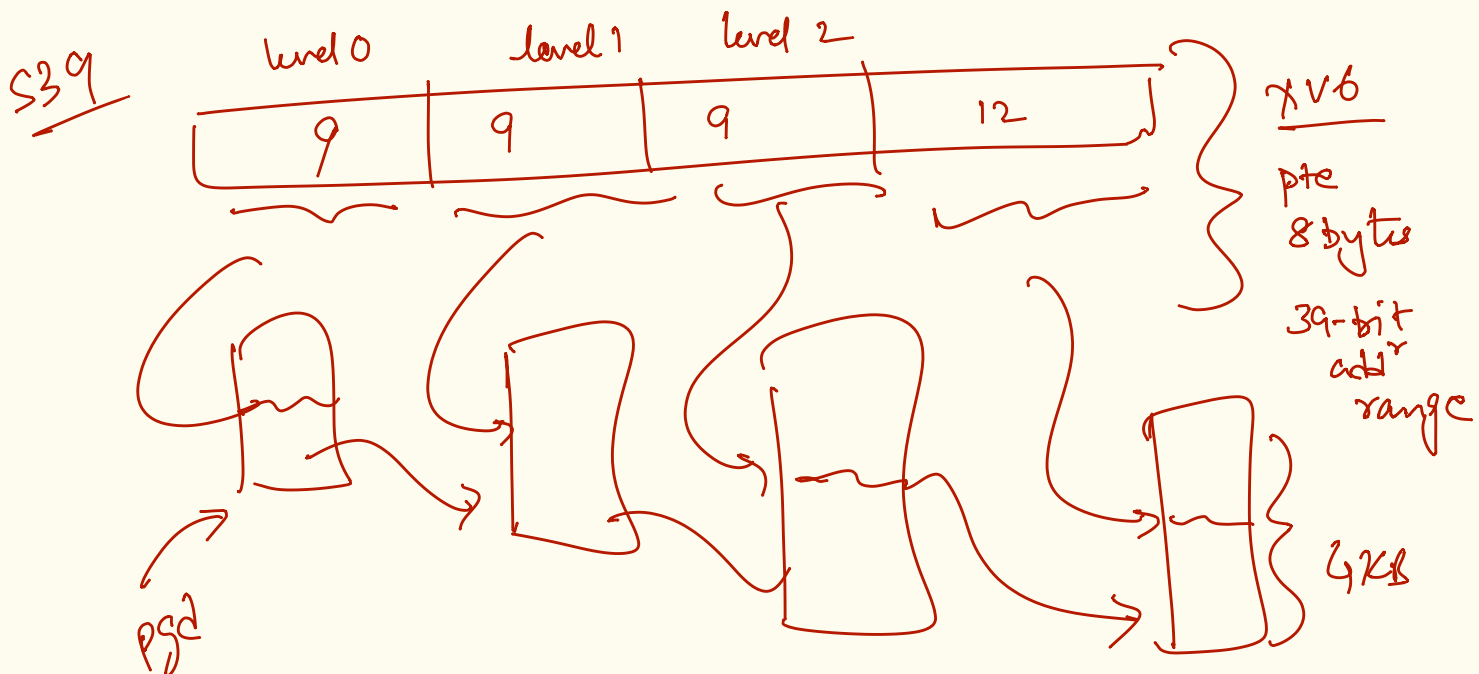


huge pages

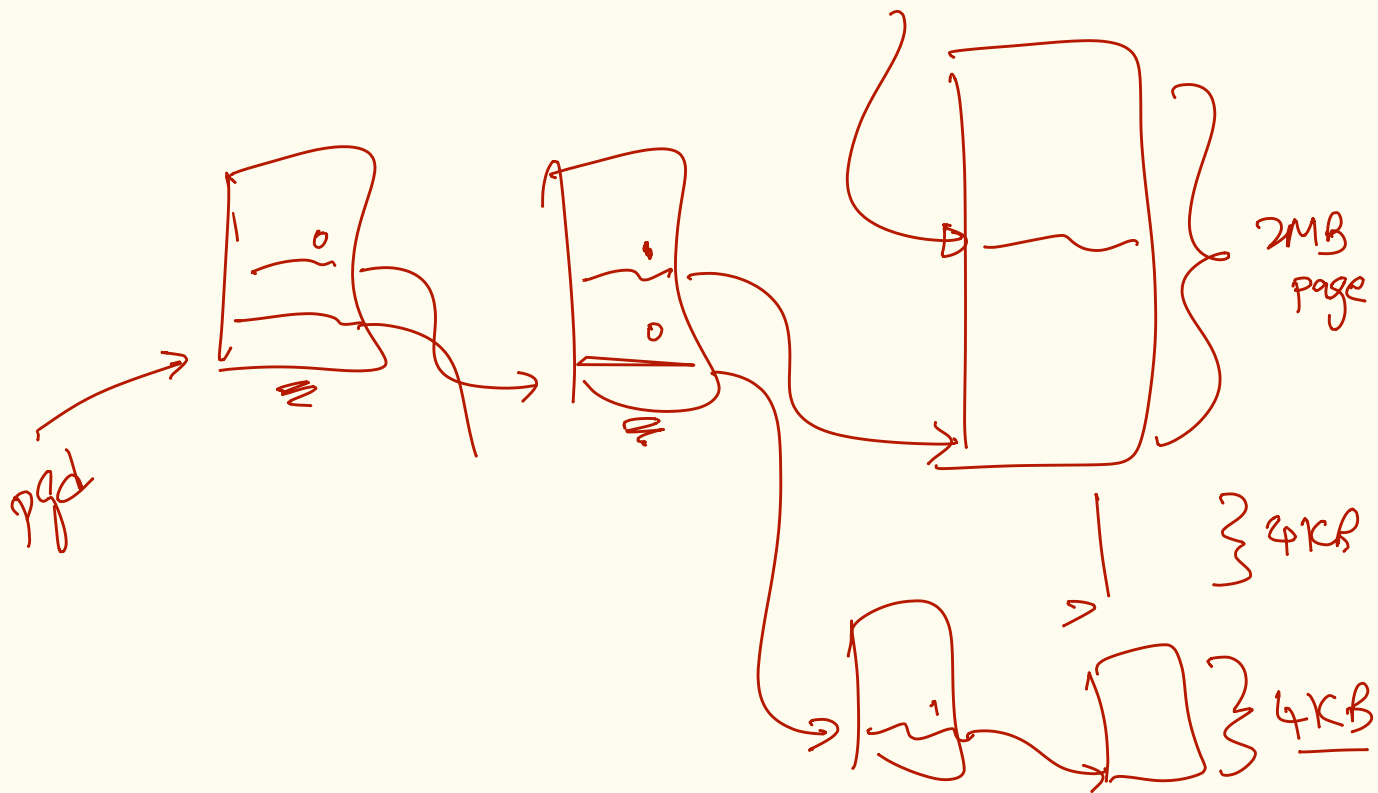
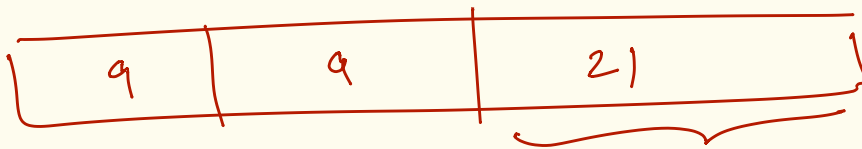
① instead of 4KB pages use larger sized pages

2MB

1GB



hugepage
2MB



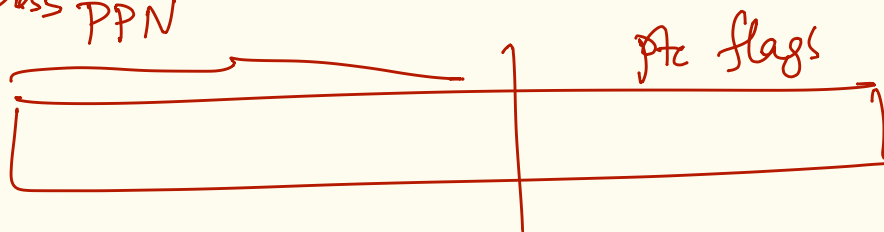
pte flags ~ encodes if
the next
level is a
page table
page or
a mapped page.

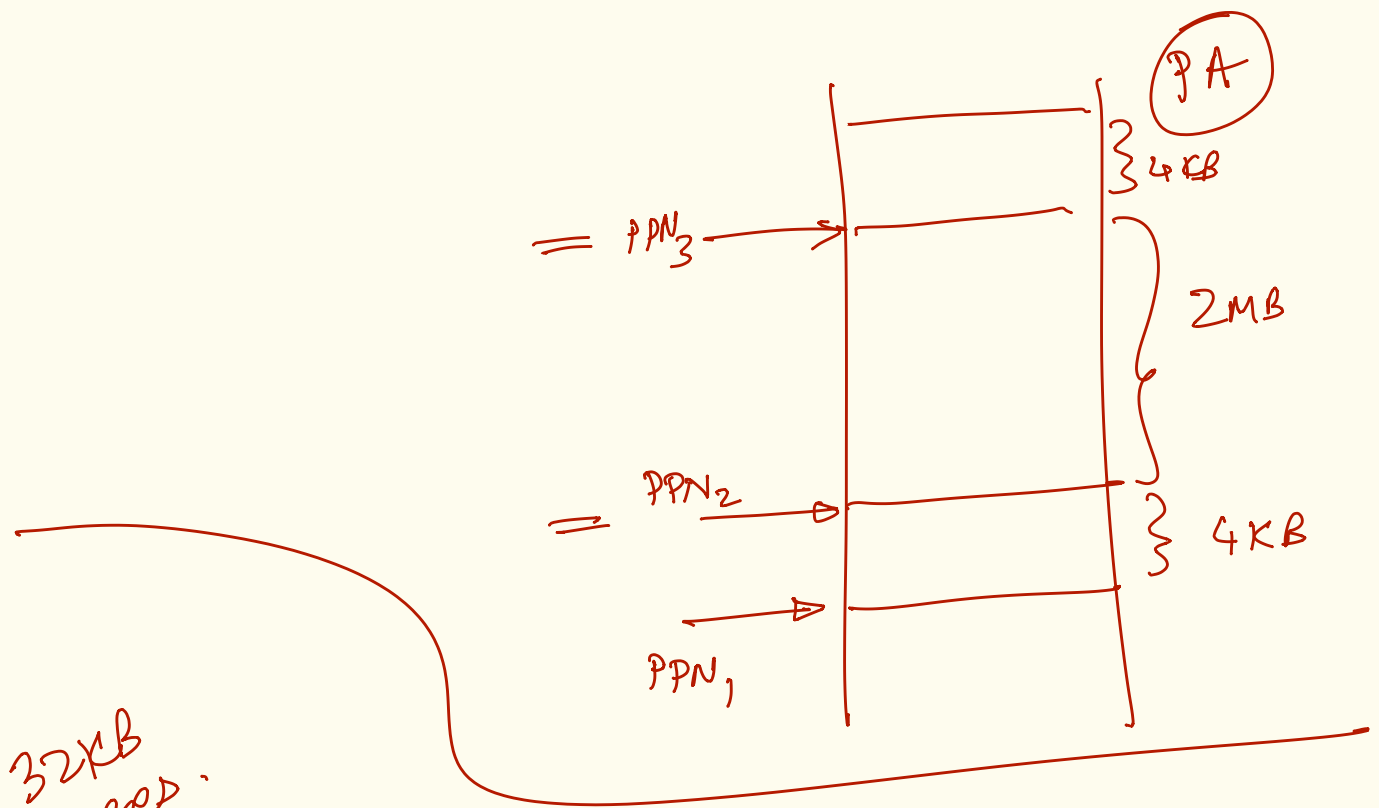
e.g: x86.

PS bit
0 — 4KB page
1 — mapped page

(start of page)
address PPN

pte:





32KB pages.

39-bit address space

8 byte PTEs

12	12	15
----	----	----

#ptes/page

$$= \frac{2^5}{2^3} = 2^{12} \text{ ptes/page}$$

64KB pages

11	12
12	11
13	10
10	13
16	

#ptes/page

$$= \frac{2^{16}}{2^3} = 2^{13} \text{ ptes/page}$$

why hugepage?

- ① smaller page tables
 - ↳ less mem ovhd.
 - ↳ quicker translations
- ② TLB efficiency - VPN → PPN

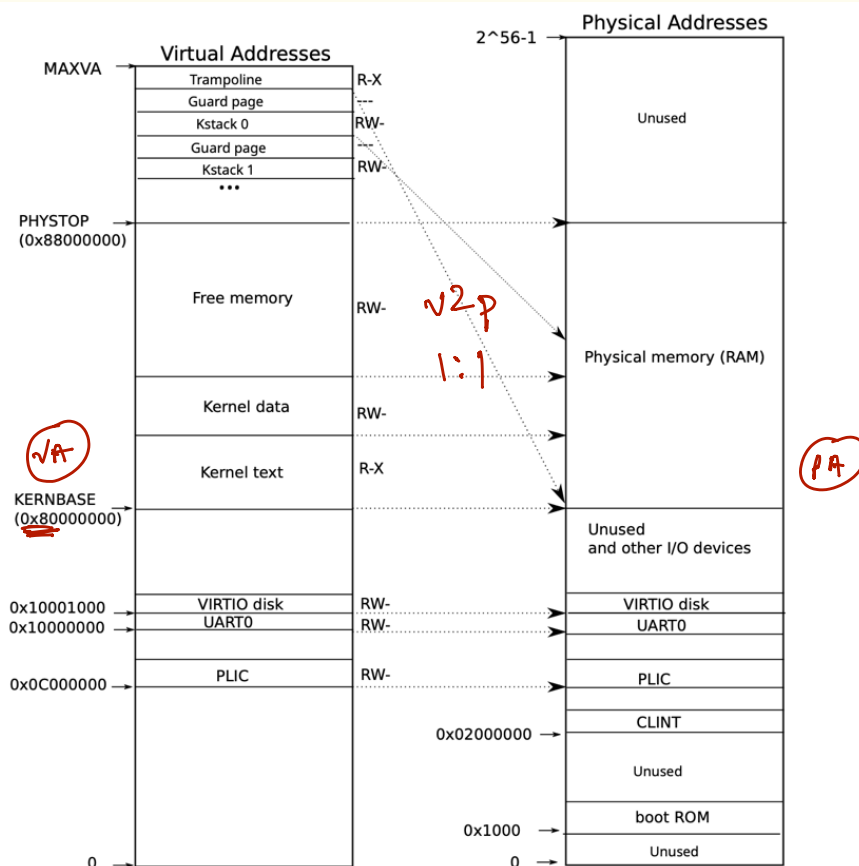


Figure 3.3: On the left, xv6's kernel virtual address space. RWX refer to PTE read, write, and execute permissions. On the right, the RISC-V physical address space that xv6 expects to see.

{ ~~flw~~ state at this and get jumpy!

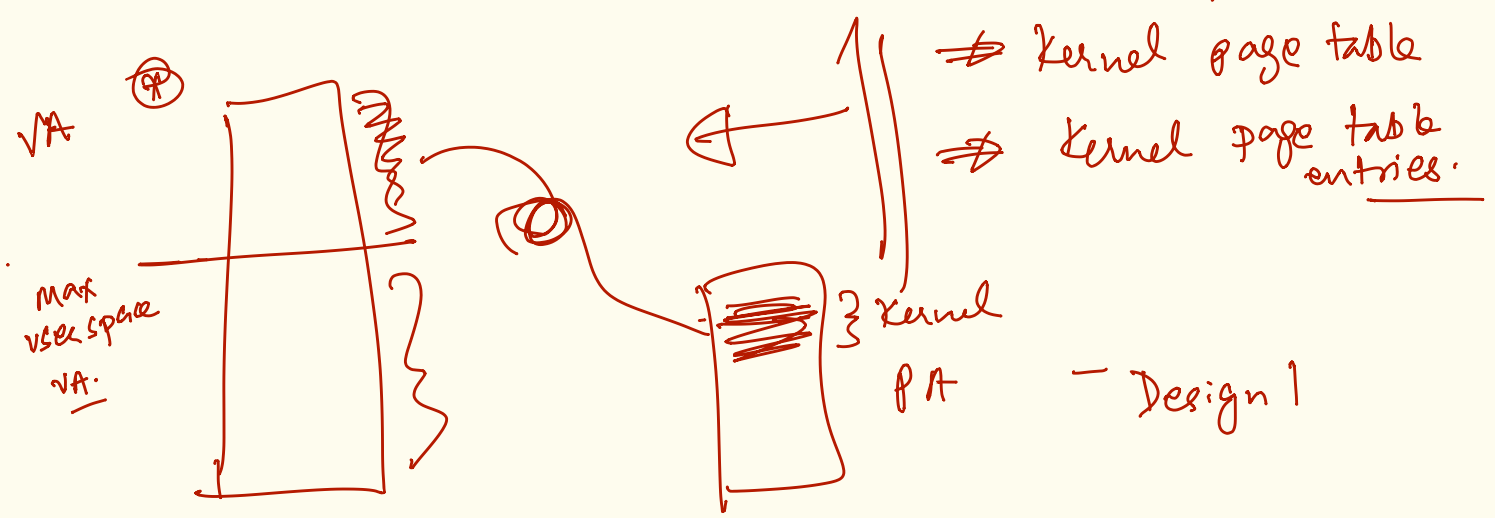
⊗ when bit 31 of CR0 is set on x86

PAGING is ON!

↓

every address issued by a CPU instruction is a virtual address.

⇒ page table from translation



⊗ Kernel & user page tables are diff.

slow from user mode to kernel — slow, kernel page table to

switch

↓

Kernel has its own VA space.

— Design2

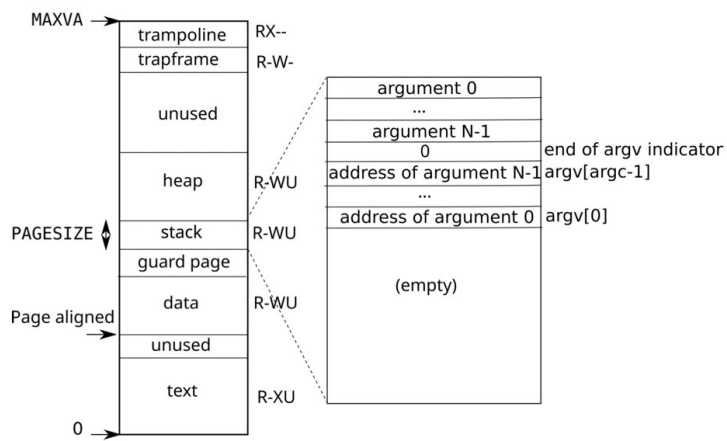
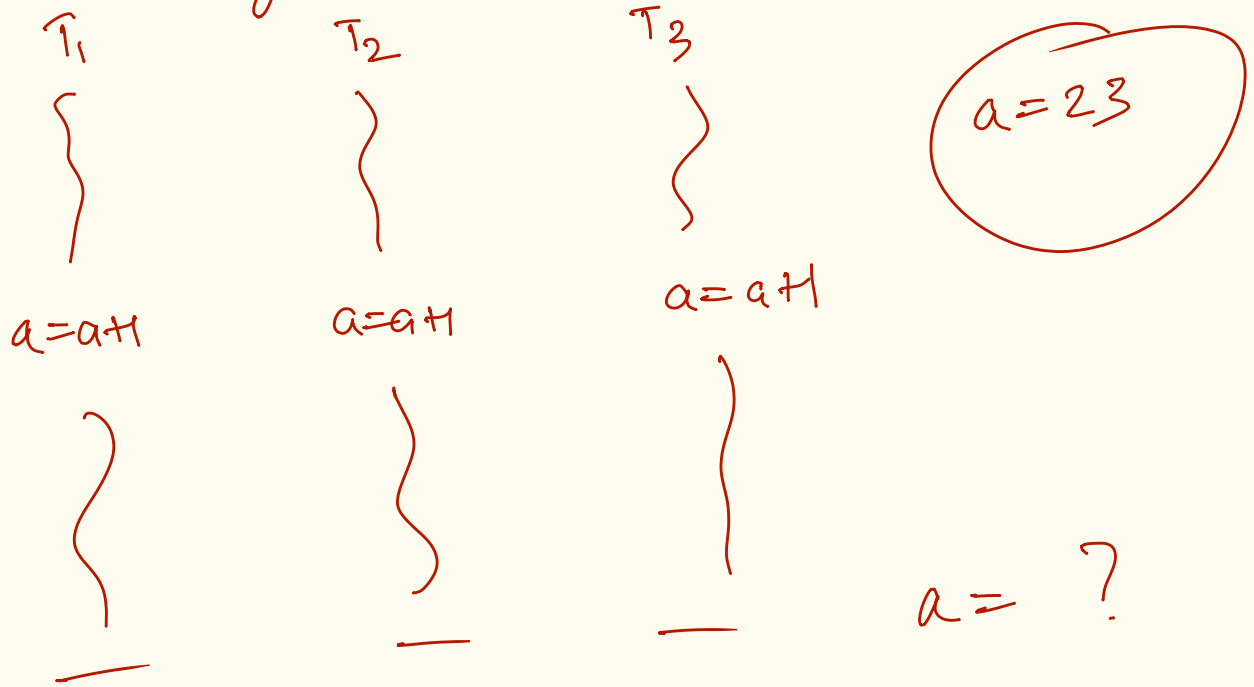


Figure 3.4: A process's user address space, with its initial stack.

Concurrency & synchronization



23, 24, 25, 26, 27, ...
| ✓ ✓ ✓ ✗
X ?

T_1 — thread!

thread of execution
instance

multiple process
multiple threads of same "process".
multiple instances of the OS (program!)

reqs
① concurrency (multiple instances)

② shared memory!