Agenda for today:

1. Fork, $\boxed{\text{exec,}}$ $\boxed{\text{wait}}$
2. Limited direct execution
3. Virtual memory. [Real mode, protected mode, etc]

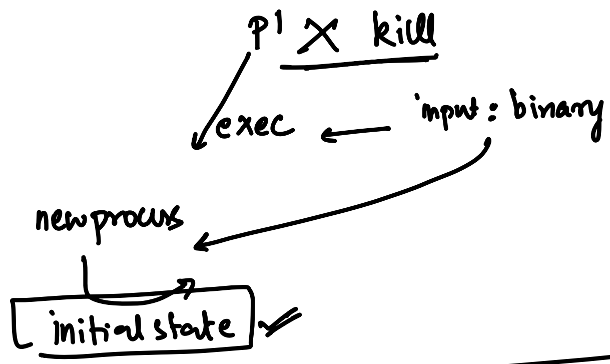
⇒ Init process / swapper process

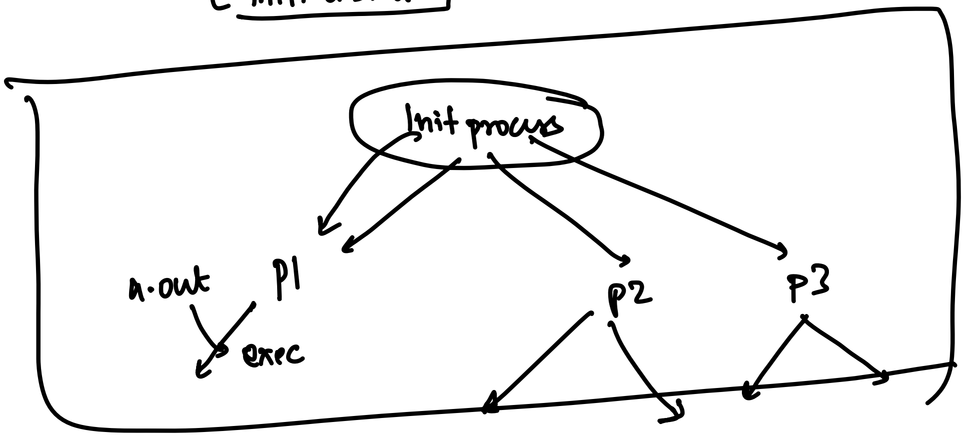
↳ pid: 0

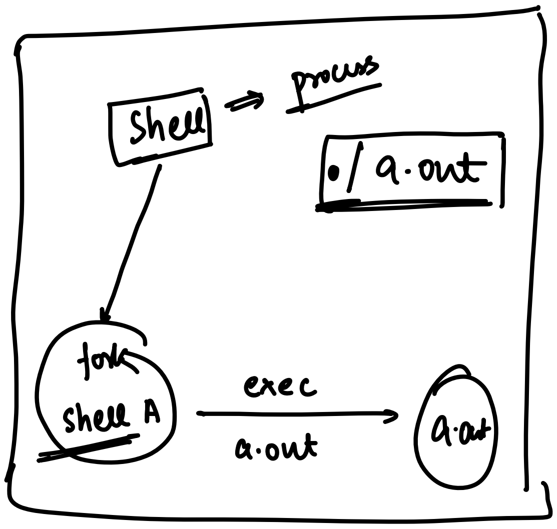
$\boxed{\text{process id}}$

$$\begin{tikzcd}
 & p1 \arrow[dl, "(fork)"'] \arrow[dr, "\boxed{\text{clone}}"] & \text{state A} \arrow[l] \\
\text{Child process} & & \\
p2 & \text{State A} \arrow[l] &
\end{tikzcd}$$

State:
- → Instruction pointer
- → memory state
- → file system (open, → reading)



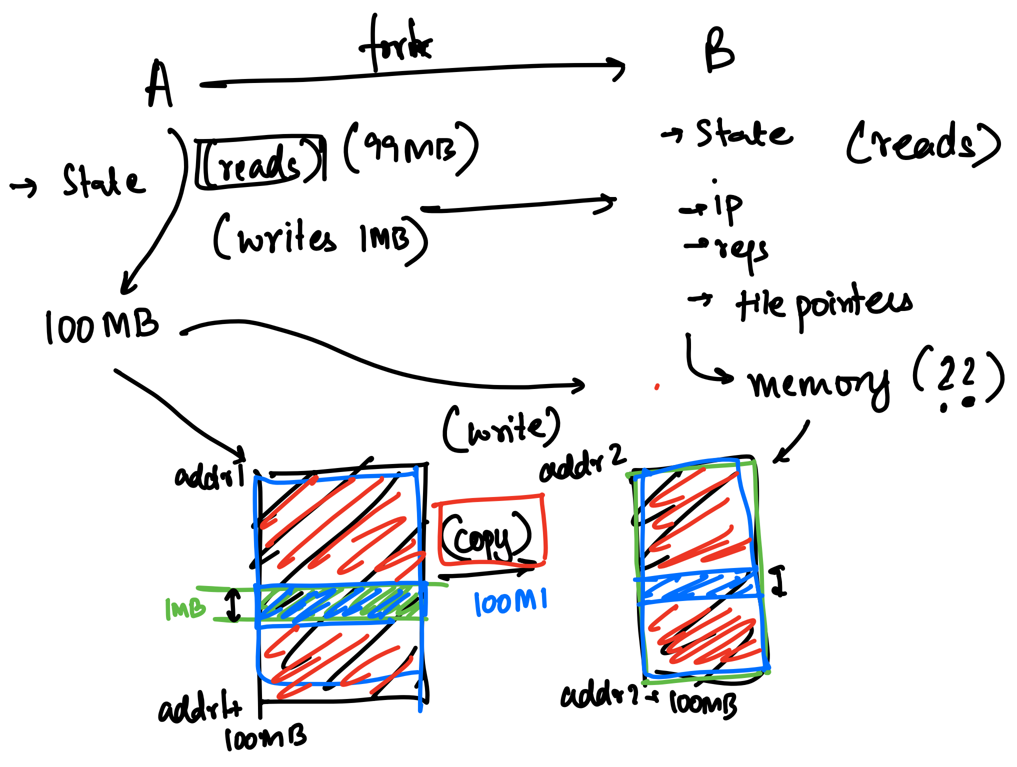




System calls: ⇒ function calls used to communicate with the OS

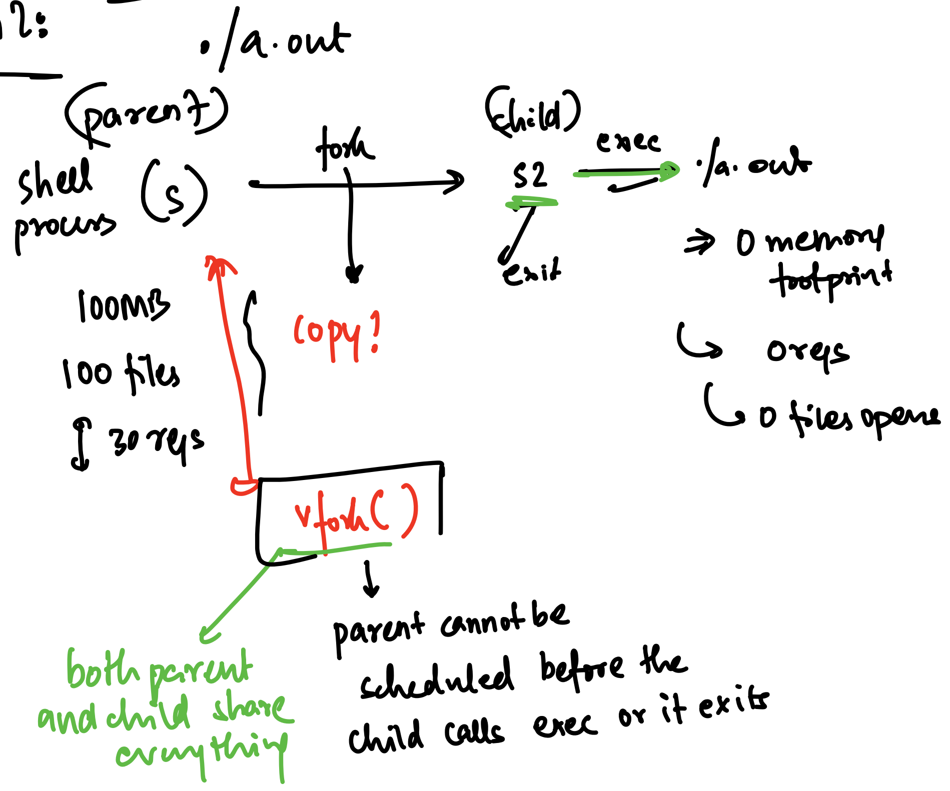
- fork (glibc wrapper of syscall)
- exec ← kill, →
- malloc ←
- open
- close
- read

Optimizations: fork and exec

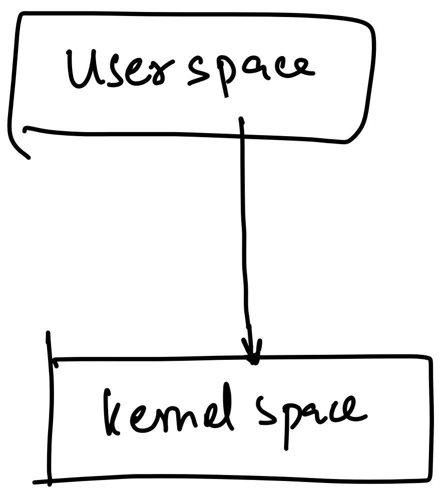


optimization 1: $\boxed{\text{CoW (Copy on write)}}$

optimization 2: vfork()

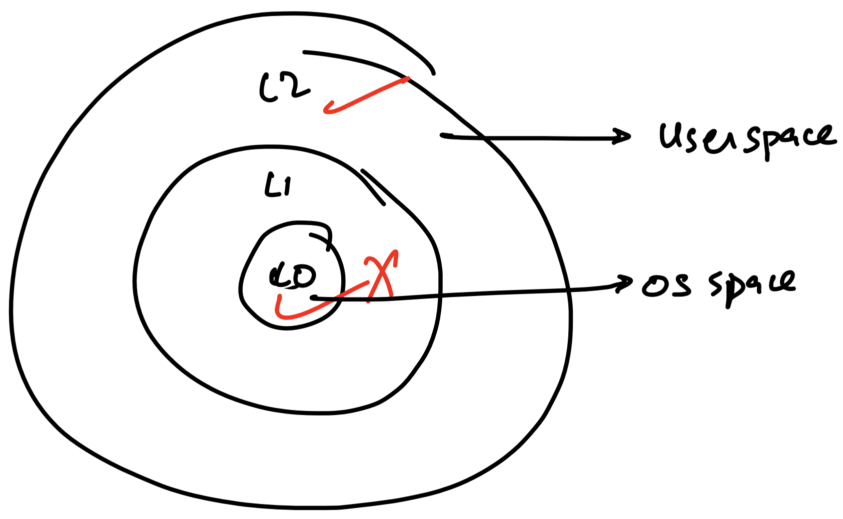


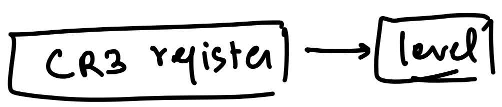
Limited Direct Execution:



you need help of hardware

x86





Go into O.S. space from user space:

⇒ System calls

⇒ ~~Illegal~~ access (memory) ??

⇒ interrupts, faults, exceptions → page fault

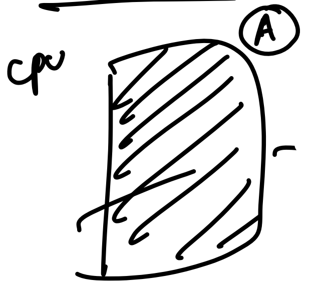
⇒ timer interrupt

keyboard, mouse,
network packet, . . .

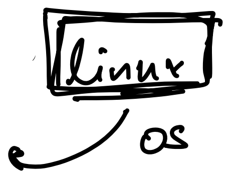
design choice?
- late handling
  - slow
  - scheduling
- bohot kam
  - overhead

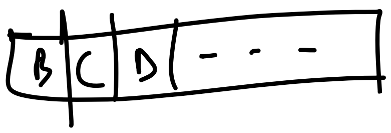
context-switch ?

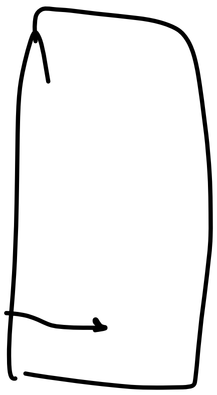
Ⓐ cpu hog



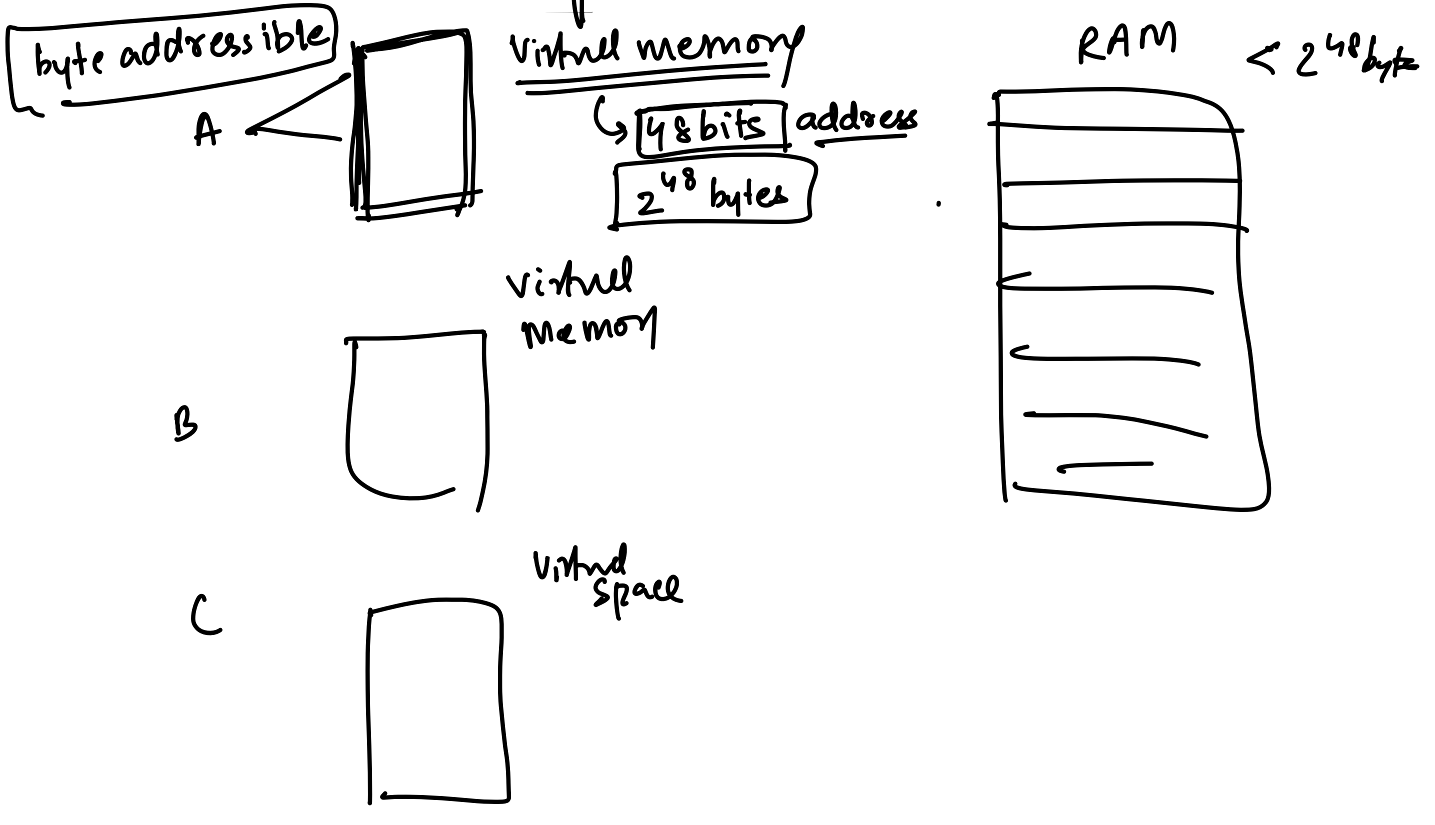
1 core





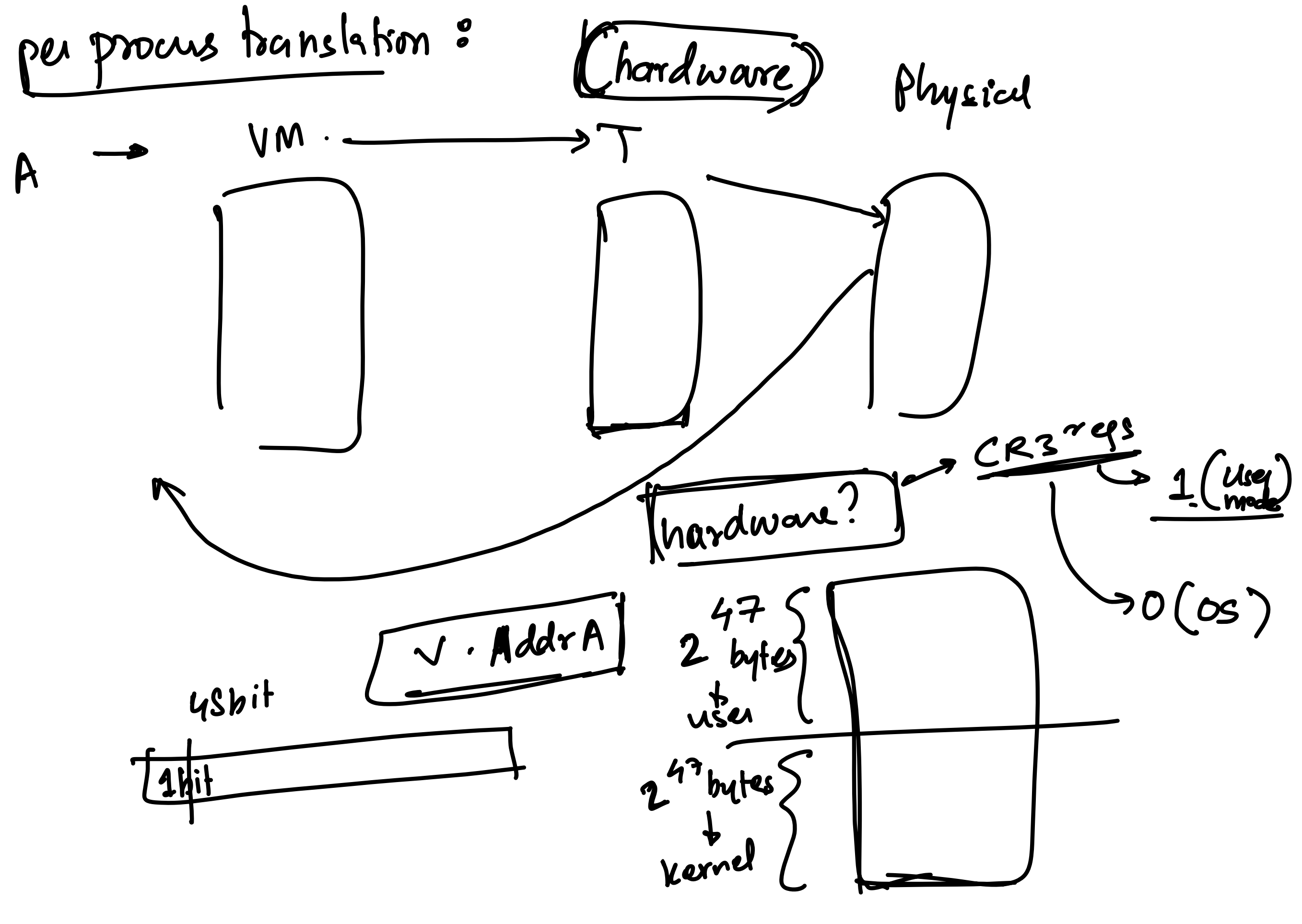


Virtual Memory: Virtualizes your physical memory



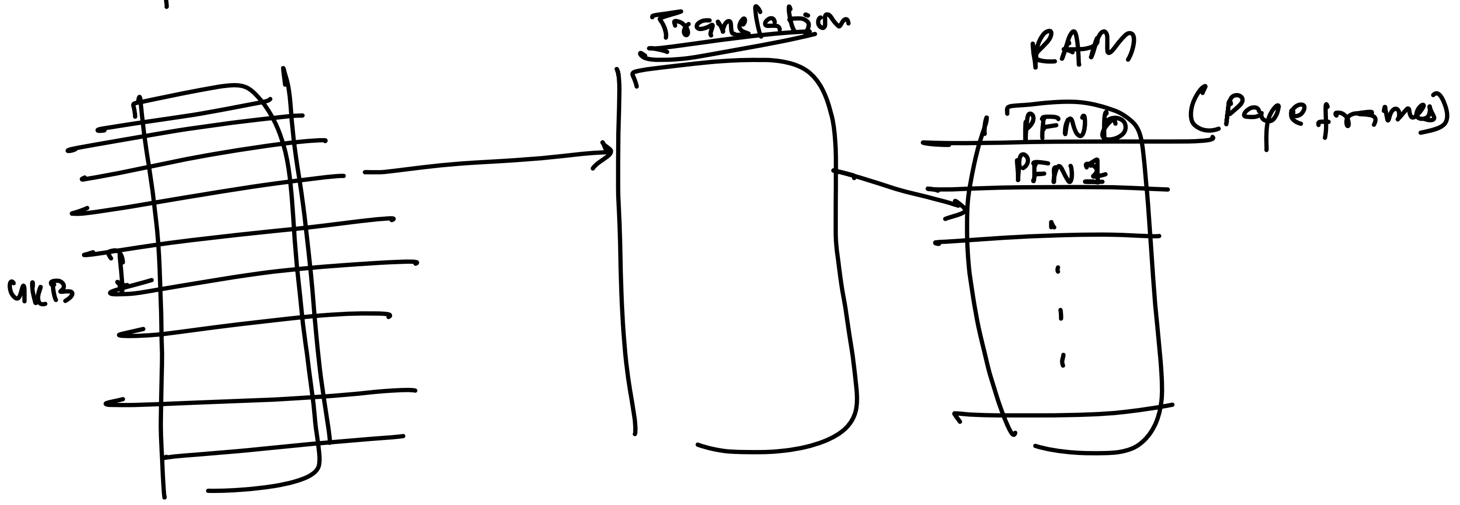
illegal → kernel

per process translation :



Linux Virtual memory

Segments ⇒ 4KB ⇒ page



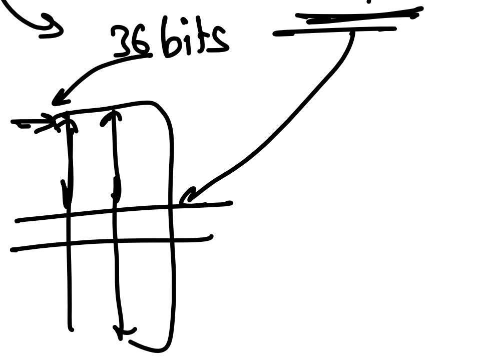
Virtual memory: 48 bits 4KB page

how many pages are there?

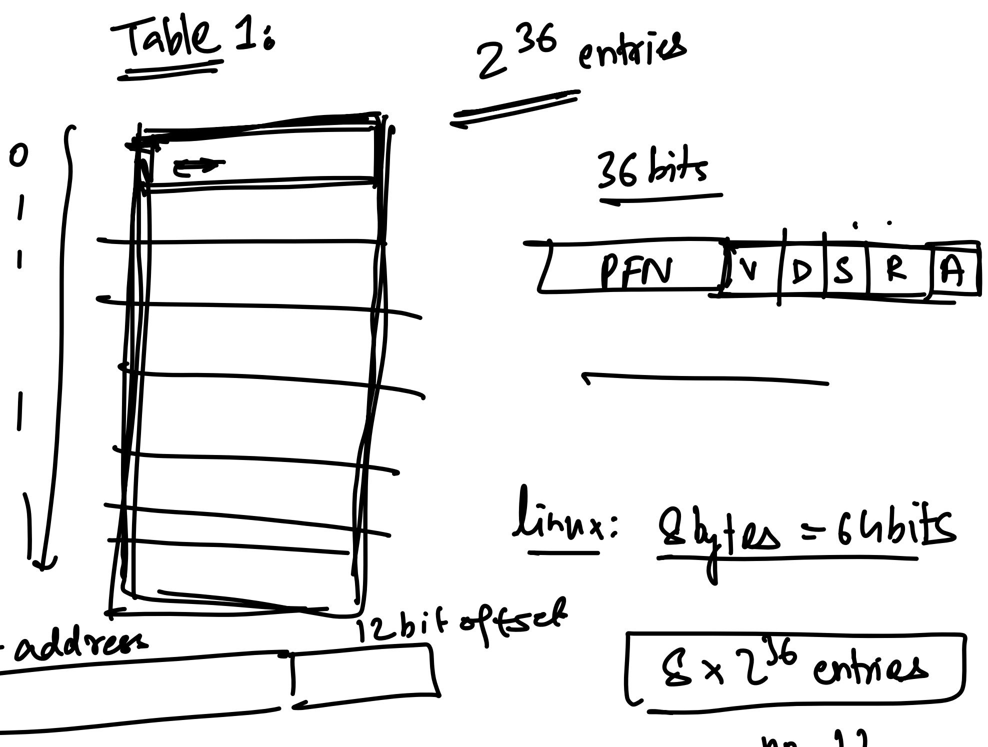
$$\frac{2^{48}}{2^{12}} \text{ pages} = 2^{36} \text{ pages}$$

12 bytes → offset

36 bits



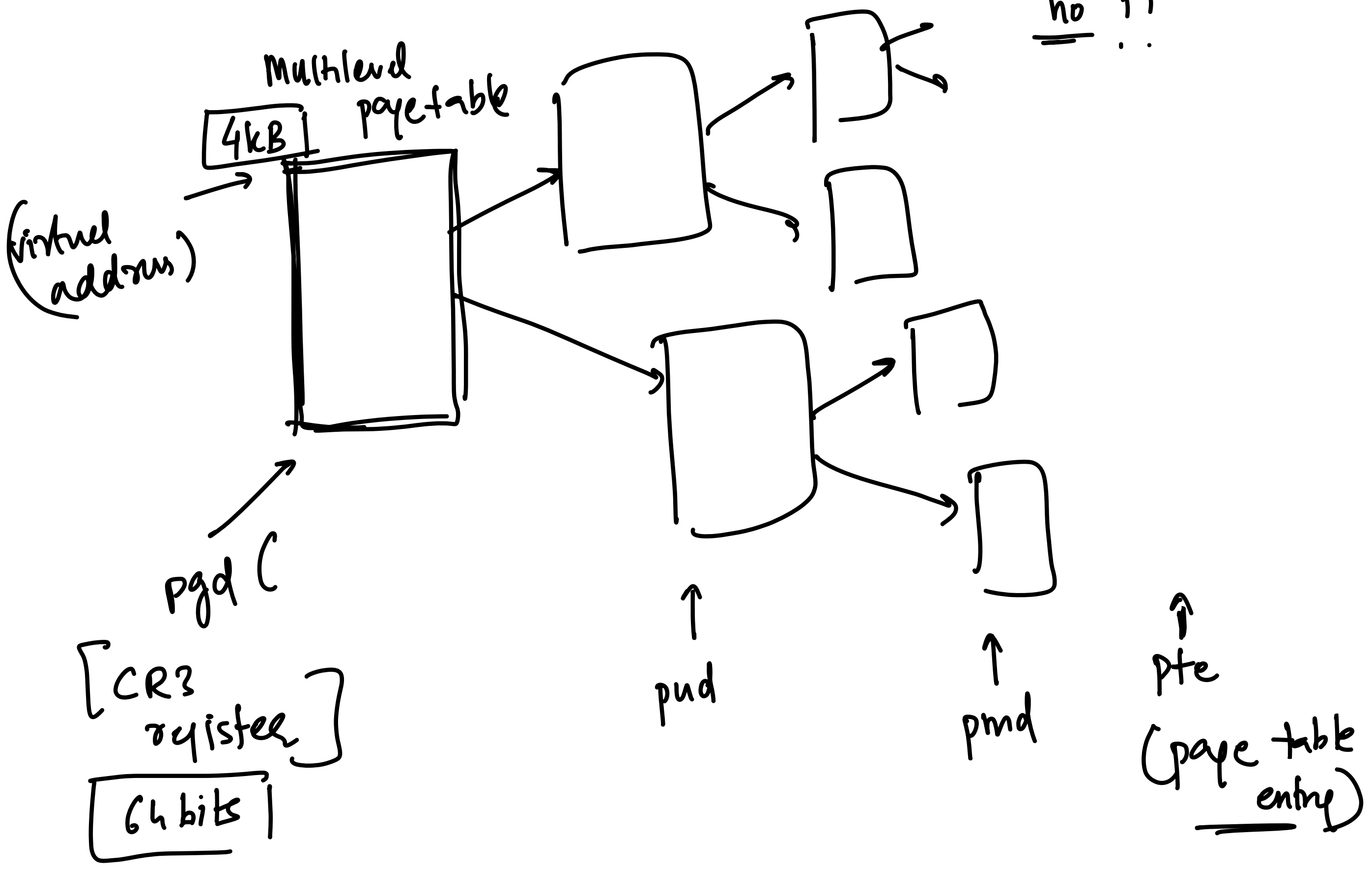
Scheme 1:



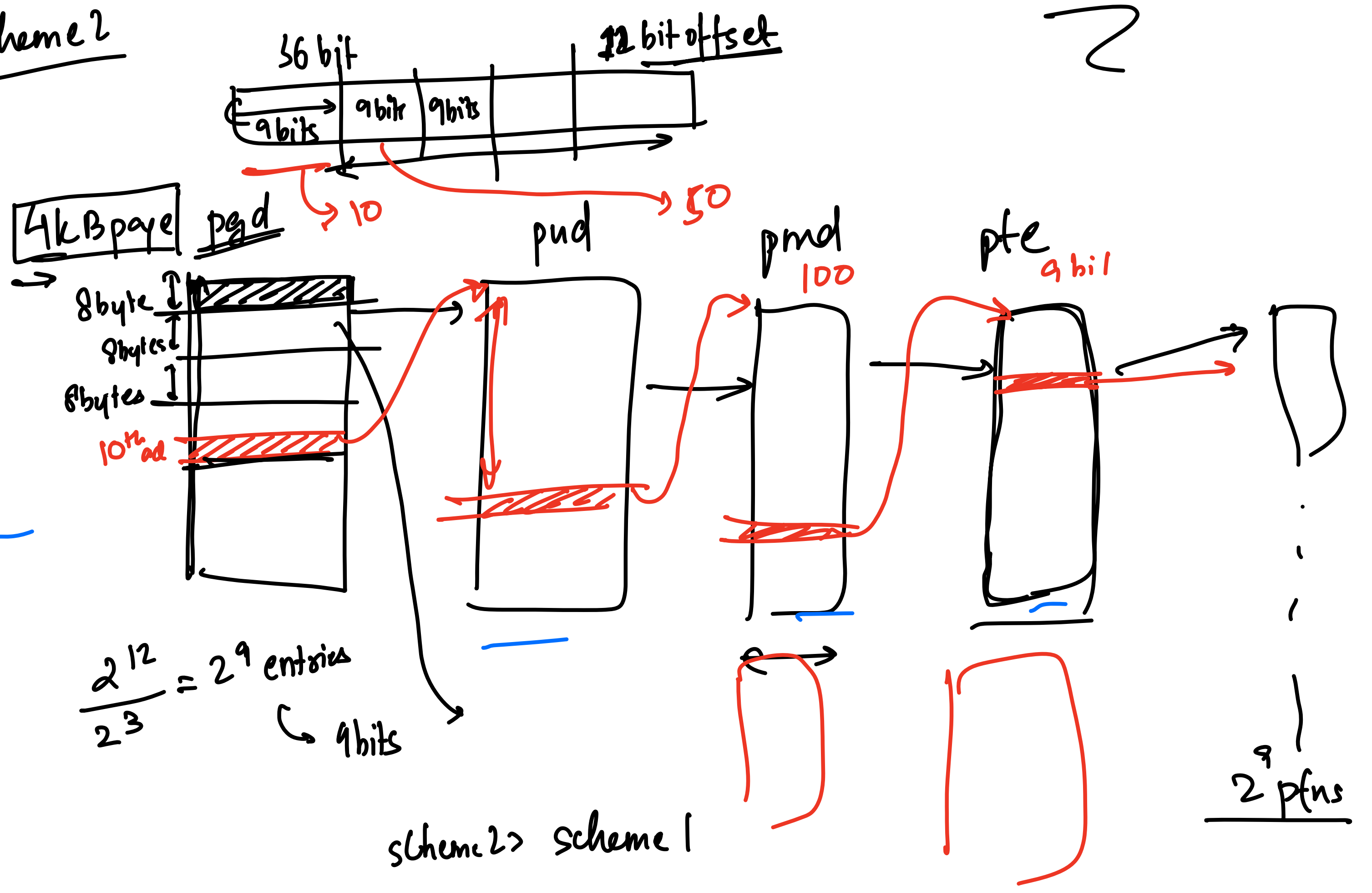
linux: 8 bytes = 64 bits

$\boxed{8 \times 2^{36} \text{ entries}}$

no !!



Scheme 2



$\frac{2^{12}}{2^3} = 2^9$ entries
↪ 9 bits

scheme 2 > scheme 1

Software walker: Virtual memory 'A'

pagetable to kernel address

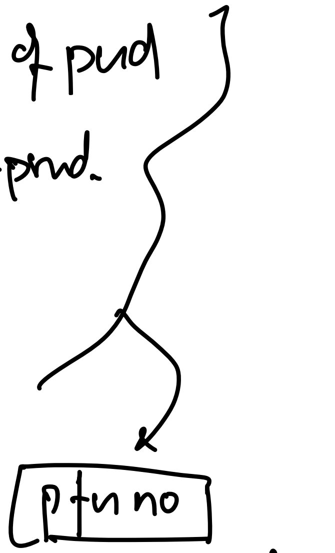
How many memory accesses do I require to get the page table entry?

1st memory access be: virtual address of pud

2nd memory access: virtual addr. of pmd.

3rd : pud

4th : pte



page frame number

$\boxed{\text{4 memory accesses}}$ ⇒ for translating one entry

loop : 0 to 100

a[0]

constant

$\boxed{\text{VPN} \rightarrow \text{PFN}}$

$\boxed{4 \times 100}$

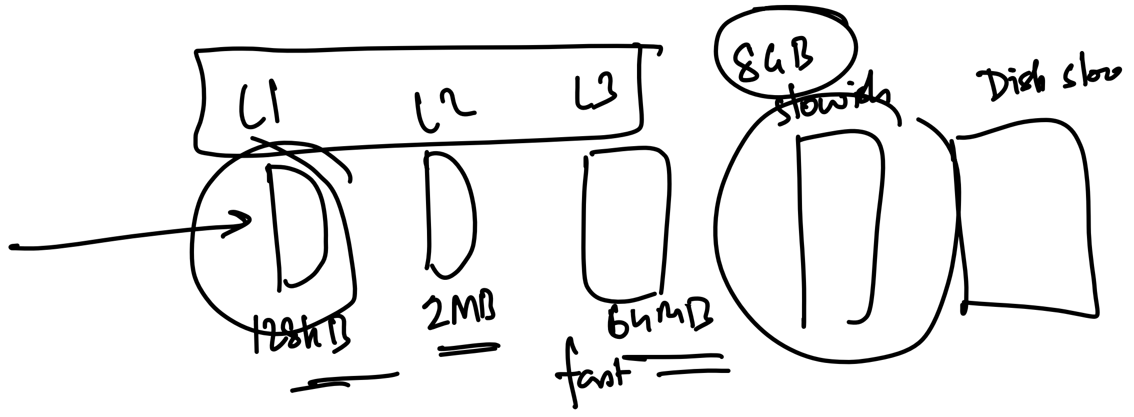
only 4 times

TLB: Translation lookaside buffer. ⇒ hardware cache

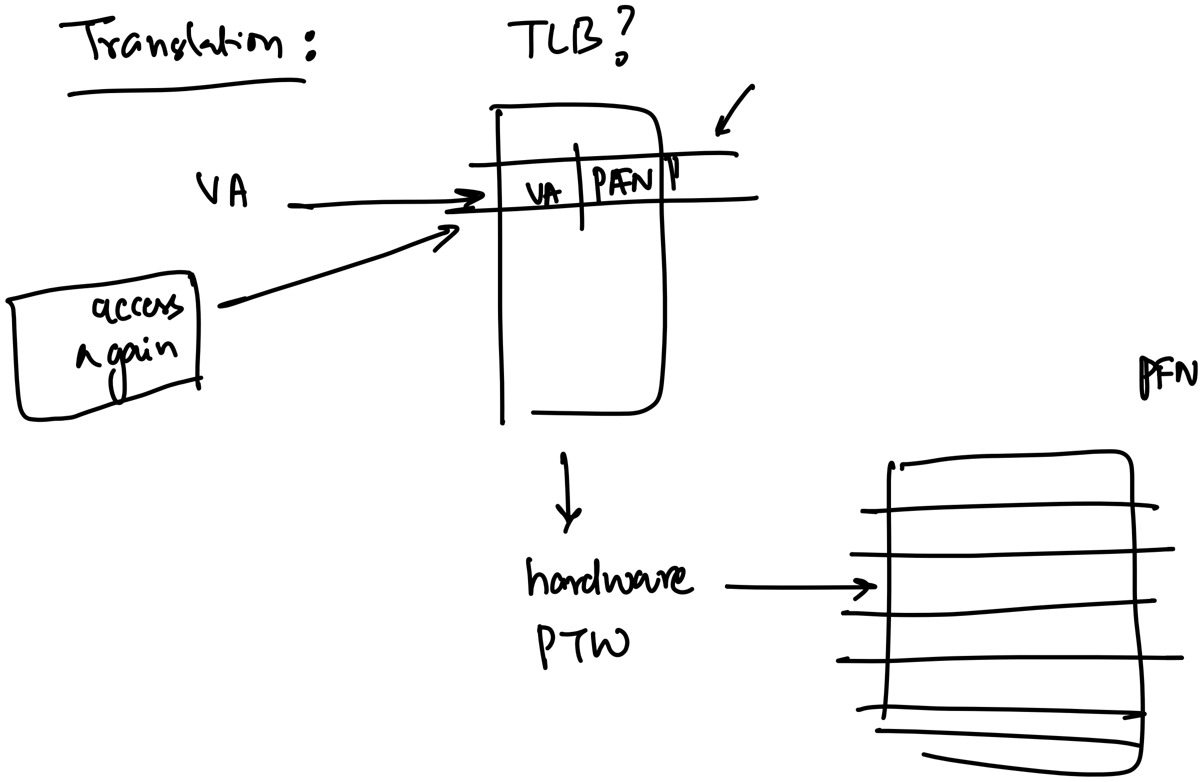
VPN 0	PFN 10
VPN1	PFN 20
VPN3	PFN 100

128 — 512 — 1024

entries



Translation:



→ Syscalls → how do they work?

→ cscope, ctags

→ Read about memory translation, paging,

→