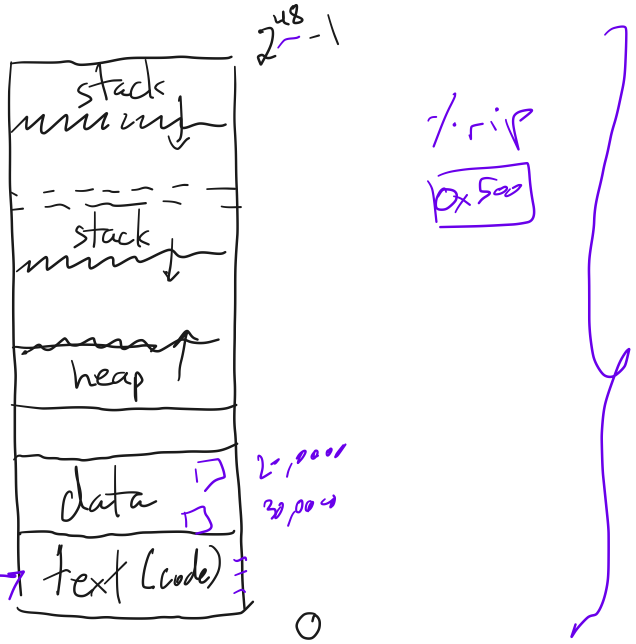
- [x] 1. Last time
- [x] 2. Intro to virtual memory
- [ ] 3. Paging
  - [x] - Intro
  - [x] - Key data structure: page table
  - [x] - Multilevel page table — large vs small pages; many levels of mapping vs. few
  - [ ] - Alternatives / tradeoffs

---

2. Intro to virtual memory

process "sees":

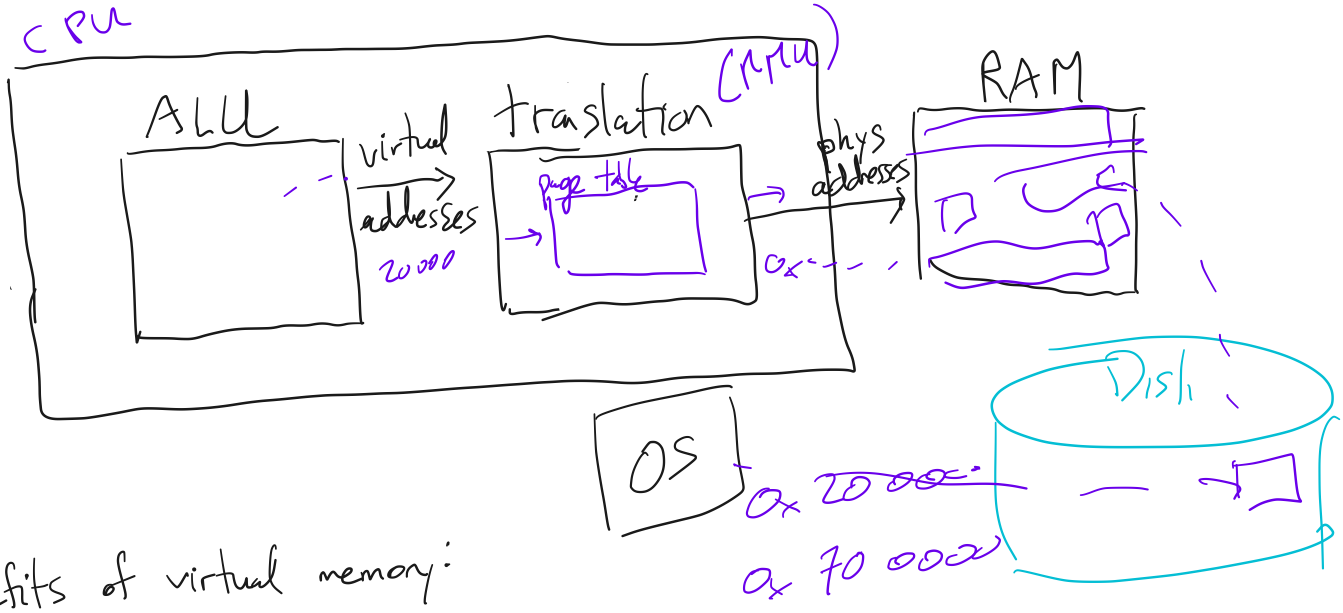


program excerpt:

code address	instruction
0x500	movq 0x200000, %rax
0x508	incq %rax, 1
0x510	movq %rax, 0x300000

load

store



Benefits of virtual memory:

- programmability
  - (a) program thinks it has lots of memory
  - (b) programs can use "easy" addresses like 0x20000

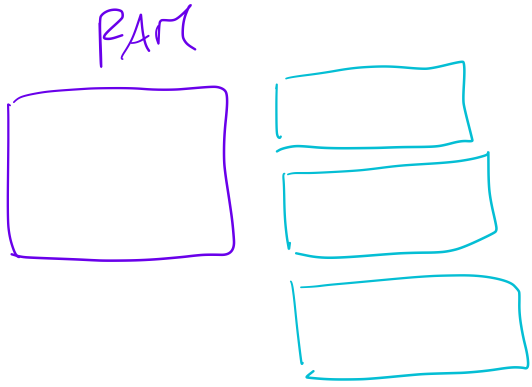
0x20000

(c) multiple instances of program can be loaded, and not collide

0x 500
0x 508
0x 510

0x 30000

- protection
  - processes cannot read/write each other's memory
  - leads to isolation

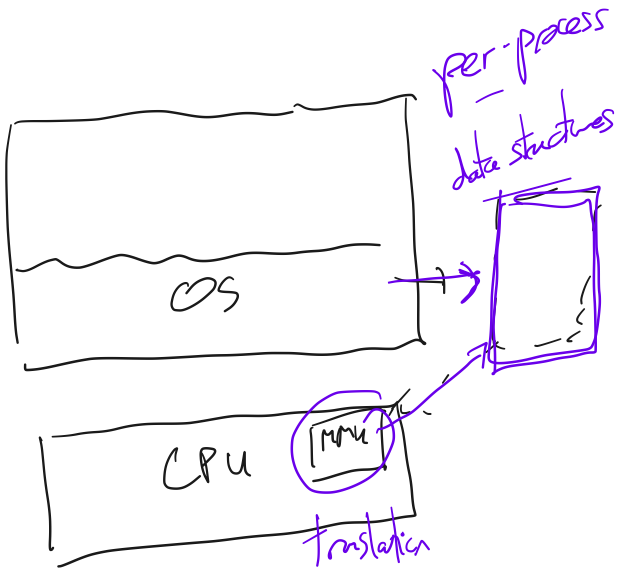


- effective use of resources

- sharing

how is the translation implemented?

- hardware does it (MMU)



---

3. Paging

A. Intro

- Divide mem (virtual and physical) into chunks.
- Called PAGES

- PAGE SIZE

↓

x86(-64): 4096B = 4KB = 2^{12} bytes

Aside: 1024

2^{10} : kilo, ~1000

2^{20} : mega, ~1 million

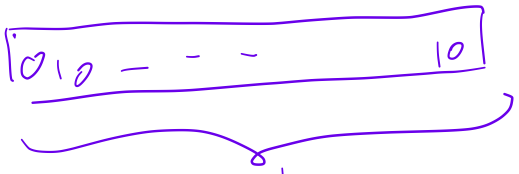
2^{30} : giga, ~1 billion

2^{40} : tera, ~1 trillion

2^{50} : peta, ~1 quadrillion

Warm-up

how pages are there on a 32-bit arch? (48-bit)

addr 

addr. 2^{32} bytes

2^{32} bytes / 2^{12} bytes/page = 2^{20} pages

2^{48} bytes / 2^{12} bytes/page = 2^{36} pages ~ 64 billion

$2^{30} \cdot 2^{6}$ = giga × 64

2^{26}

2^0

$2^6 = 64$

$2^7 = 128$

$2^8 = 256$

$2^9 = 512$

$2^{10} = 1024$

page 0: [0, 4095]

page 1: [4096, 8191]

⋮

page $2^{20}-1$ [..., $2^{32}-1$]

VPN virtual page #

PPN phy page #

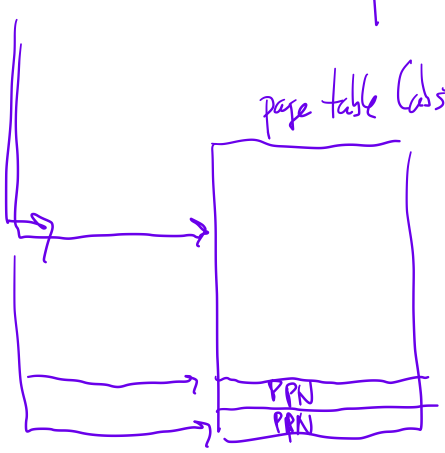
↳ PFN (textbook)

B. Key data structure: page table (per-process)

implement mapping from:

VPN → PPN , per process

Virt addr: [VPN | offset]



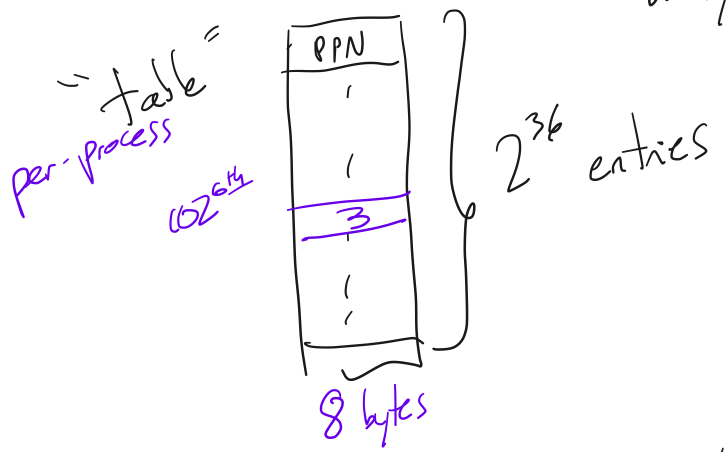
expresses a mapping about 2^{12} contiguous bytes addresses

PA : PPN + offset

assume: 48-bit addresses, and 4KB pages

2^{12} bytes

4KB = 2^{12} bytes = page size



Ex:

OS wants a process to use address

VA: 0x00402000 refer to PA: 0x00003000

36 bits 36 bits 36 bits

12 bits

table[0x00402] = 0x00003

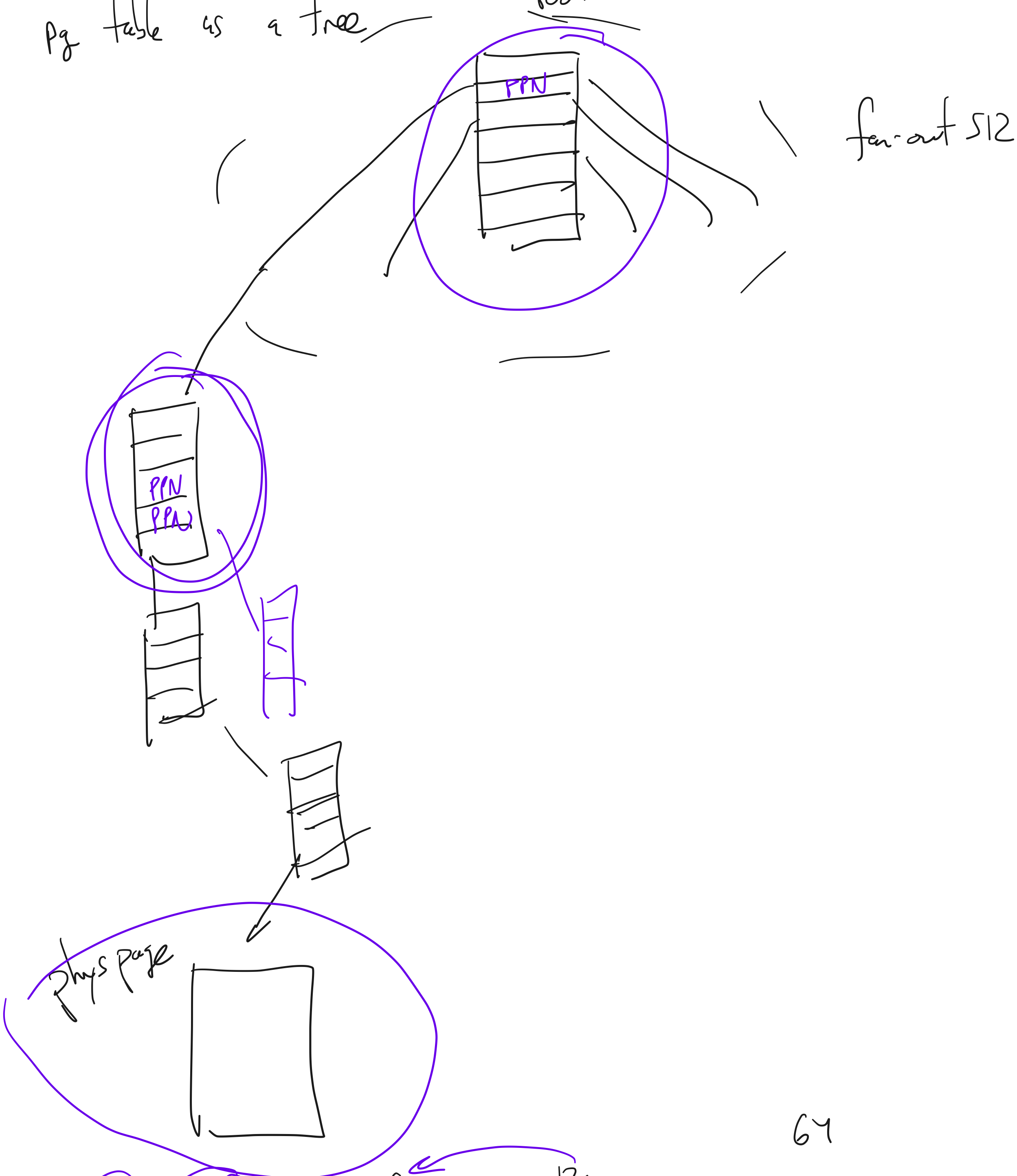
1026

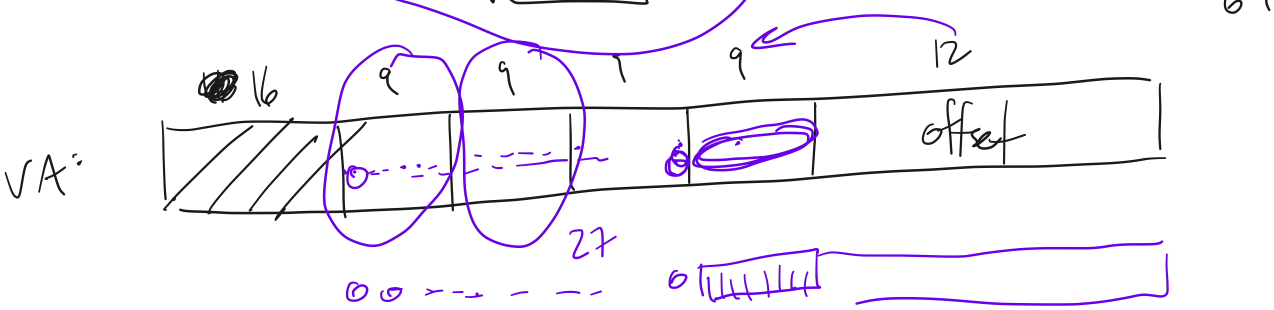
Issue: too much memory: 2^{36} entries × 8 bytes/entry =

$2^{36} \times 2^3 = 2^{39}$ bytes = 512 GB

3. Multilevel page tables

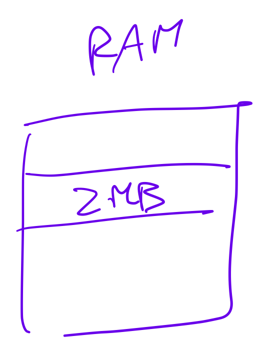
Rep. pg table as a tree





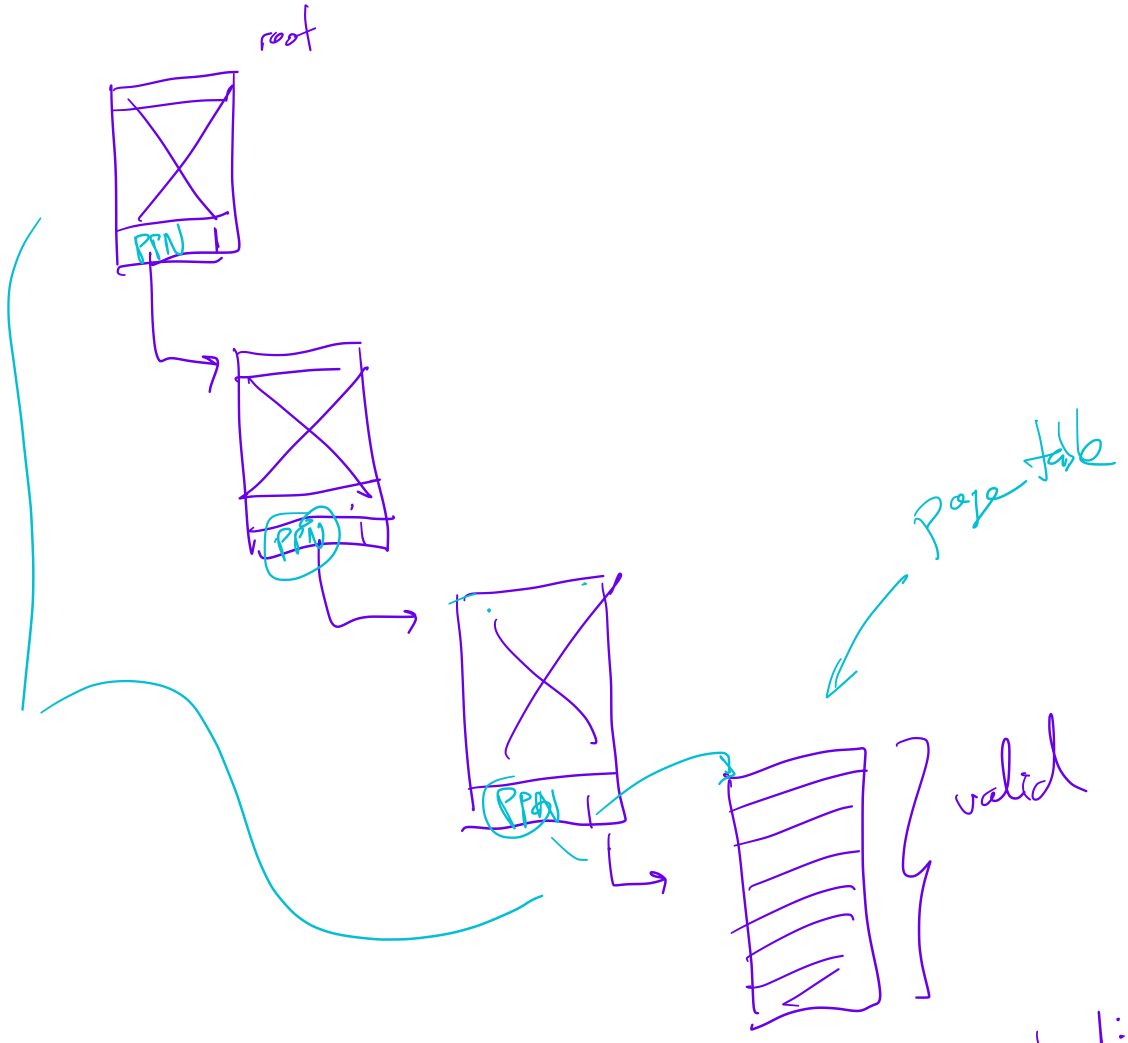
$2^9 = 512$

Ex: Map 2MB of phys memory at VM $0, \ldots, 2^{21} - 1$



[(2 2MB · (]

2MB/4KB = 512 entries



512 pages

4

+

516 pages

overhead:

4/512

2^{36} HUGE / 512