1. Intro to Virtual memory
2. bit manipulation
3. Paging
    --intro
    --page table
    --multilevel page table
    --alternatives & tradeoffs

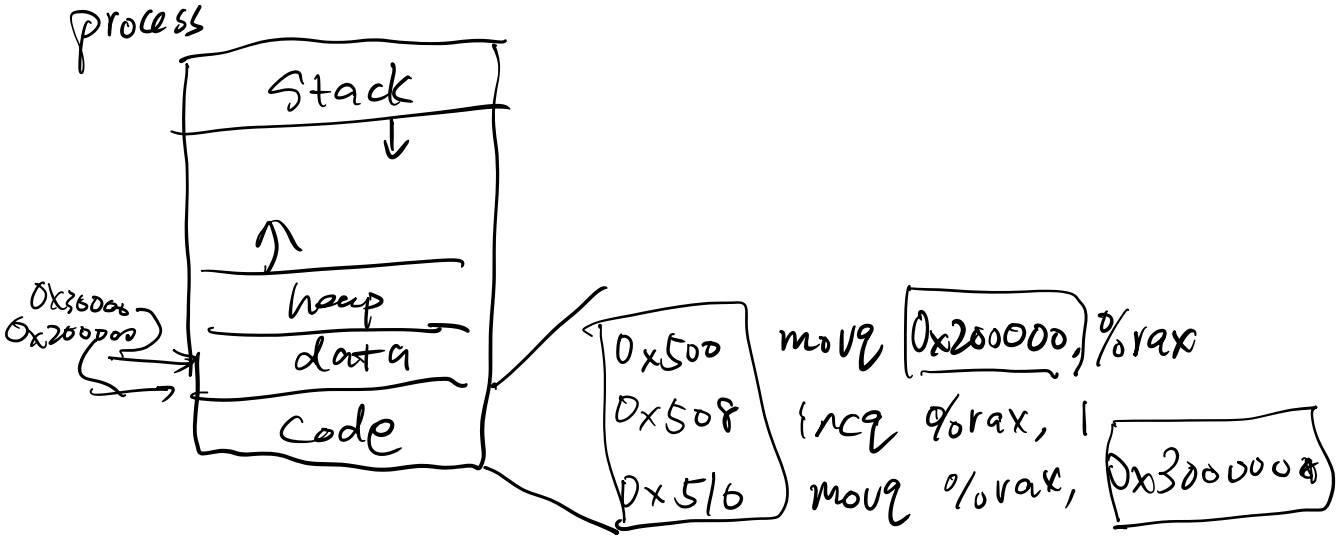
---

① private : 3 questions

② public

Virtual Memory.

VA ⟶ PA



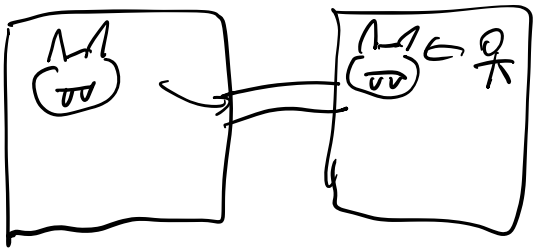
Q:

4. 6. 2. 5

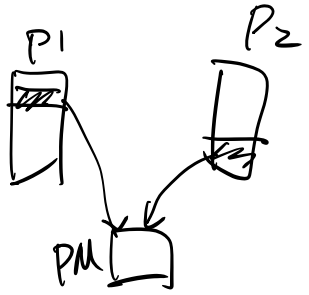
① Programmability (transparency)

② protection.

if you cannot name sth,
you cannot use it.



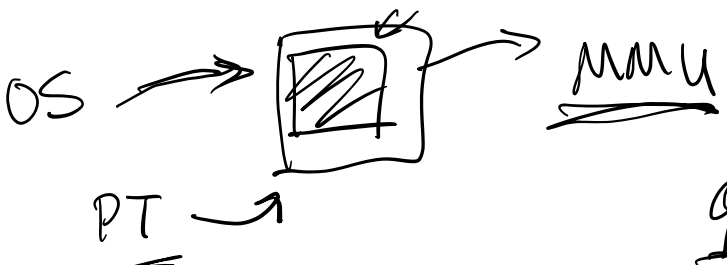
file descriptor



③ effective use of resources

④ sharing

VA → PA



movq 0x200000, %rax

%rip

qemu

bit manipulation.

1 bit < 0
1

hex. 0x 1234…9 a b c d e f
10 11 15

0000 = 0x0

1111 = 0xf

Ternary Computer

1 trit < 0
1
2

binary: 3×2=6
XXX $2^3=8$
→ 2 states

Trinary: 2×3=6
XX → 3 $3^2=9$

32-bit CPU vs 64-bit CPU

2^{32} → Byte addressable → $2^{32}B = 4GB$

1B = 8bits

64-bit X86-64 → 48-bit VA

$2^{48} = 2^{40} \times 2^8 = 512TB$

- n bit ⟺ large memory.

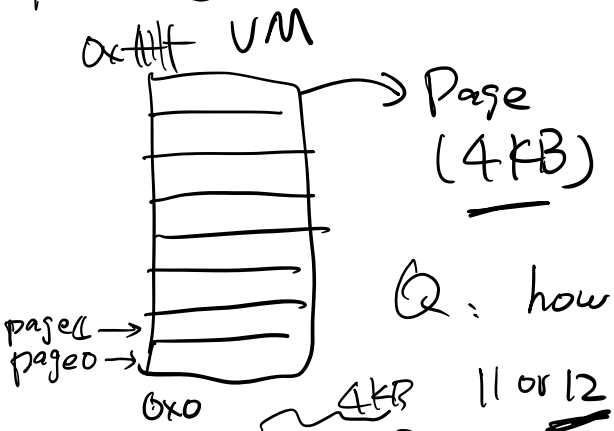
  5 bit ⇒ 2^5 B

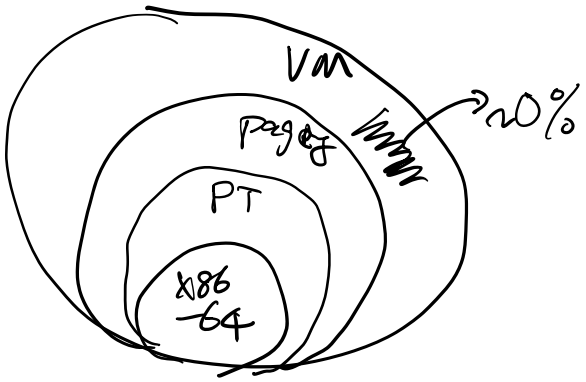
  10 bit ⇒ $2^{10} = 1024$ B

- 2^{10} Kilo 1KB
  2^{20} Mega 1MB ← L3 Cache
  2^{30} Giga 1GB ← Memory
  2^{40} Tera 1TB ← disk

2^{50} Peta 1PB ← big size of DC.

Paging





Q: how many bits? 11 or 12

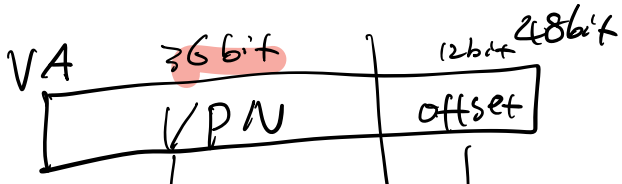
$2^{12} = 2^{10} \cdot 2^{2}$

1KB 4

page0 ≡ [0, 4095]

VPN —PT→ PPN.

page number



VA 12bit PA

0x402000 ⇒ 0x003000 12bit

PageTable [VPN] ⇒ PPN 36-bit

PT

Concat

PPN + offset

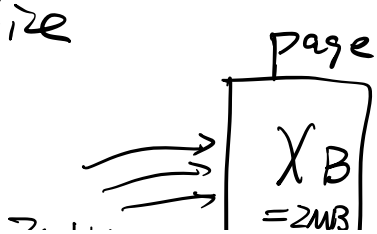
PA

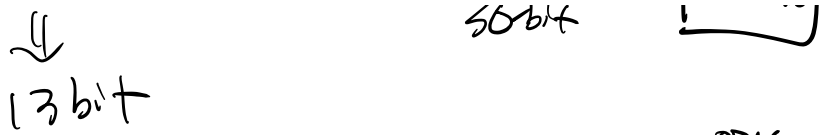
Example:

pageTable [0x402] ⇒ 0x3 = 0x003

Q: offset vs. Page Size

$2^{12bit} = 4KB$





Q: VA 0x 123456 ⟹ PA. 0xabcdef6.

VPN offset PPN offset

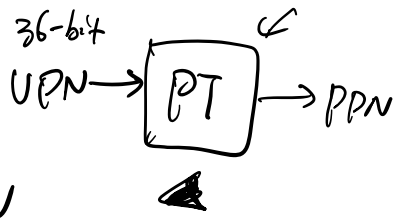
is it possible in x86-64?
(Page size = 4kB)

- possible
NOT

Q: how can we make ↘ happen?

page size ≤ 2^4 = 16B.

- How to implement PT?



. array PT[VPN] ⟹ PPN (36-bit)

⟹ #item = 2^{36}

. linked list

SLOW.

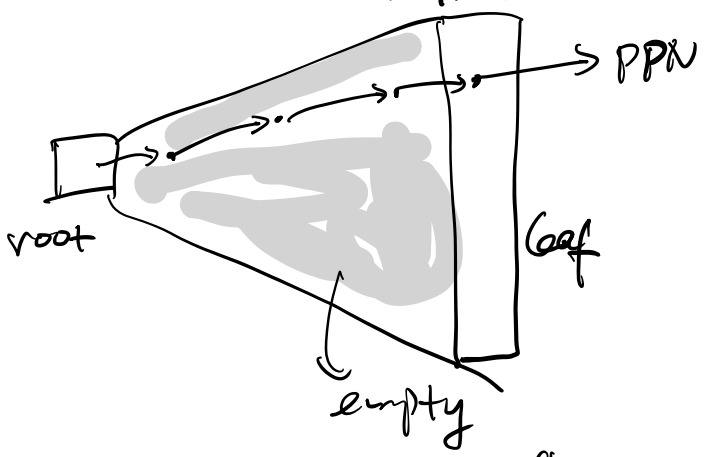
⟹ hash map/table

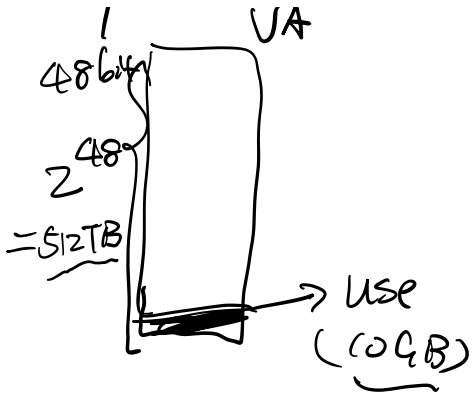
hash(VPN1) → PPN

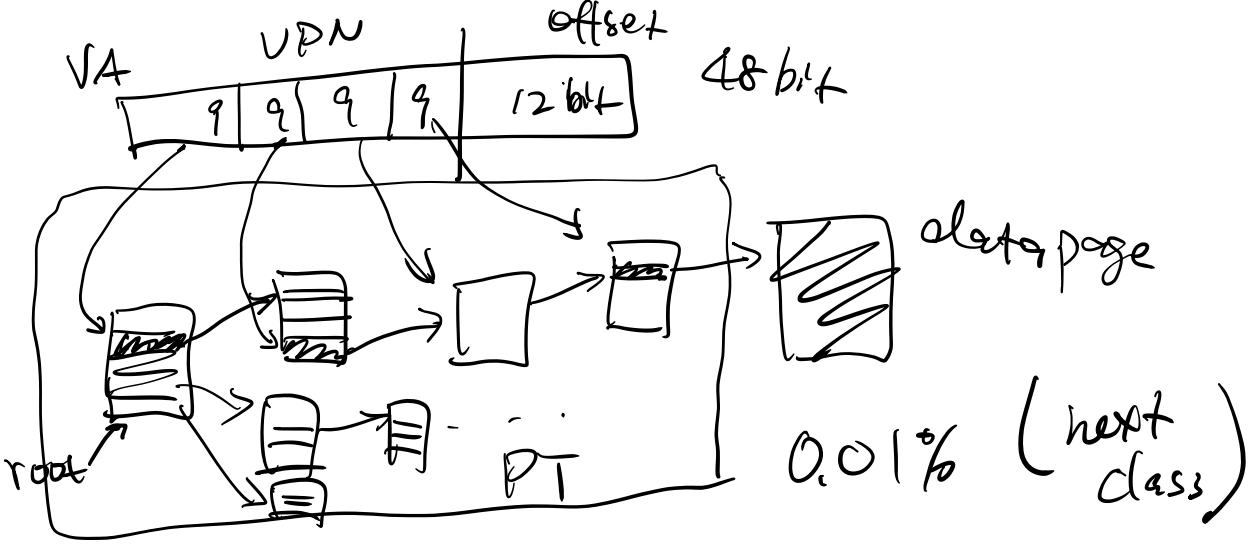
hash(VPN2) ↗

VA - 12bit = 36

- multi-level PT.

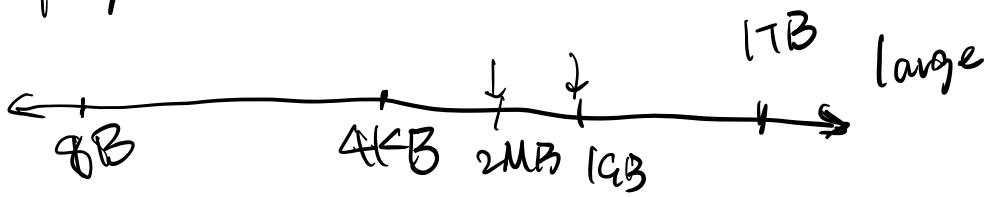






+ tradeoffs.

- page size.



- depth of PT.