Virtual memory

□ 1. Virtual memory intro

□ 2. Paging

□ 3. Page table

□ 4. Today's virtual memory

------

1. Virtual memory introduction

Virtual memory benefits:

(a) programmability

① huge contiguous memory address space

② multiplexing memory addresses 0x800000

(b) protection:

① separate addr space (isolation)

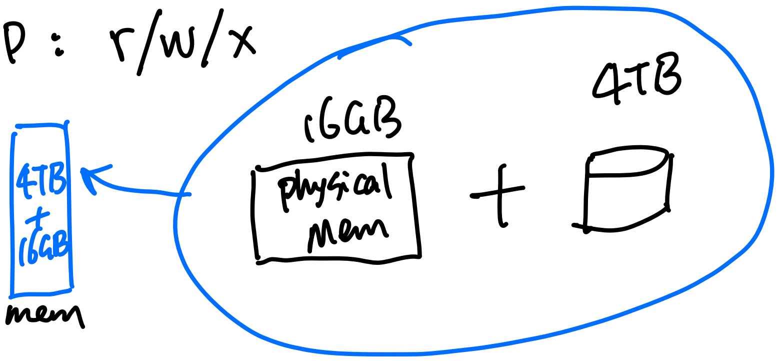
② access control → priv-level: M/S/U

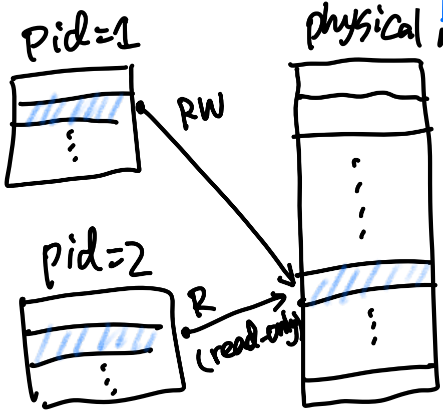
→ op: r/w/x

(c) effective use of resources:

① overcommitting memory

② secure sharing memory



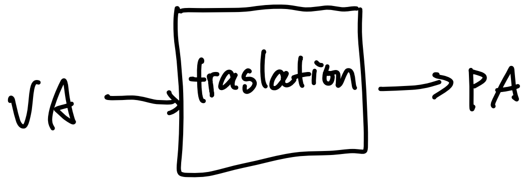


Virtual memory → ① translation (*)

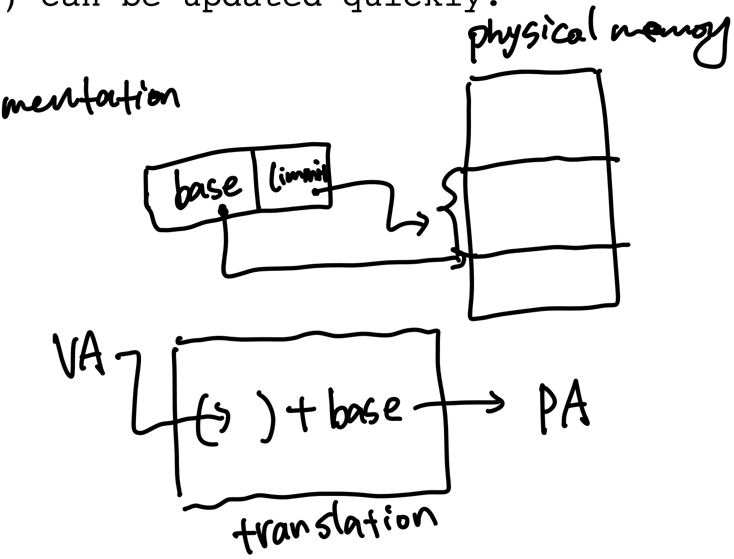
→ ② protection

2. paging

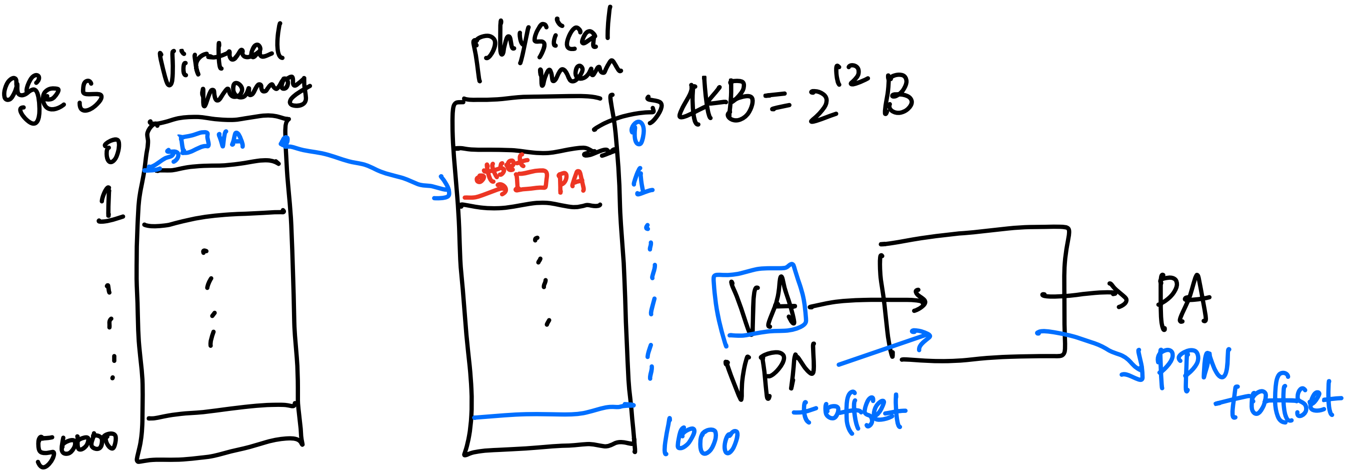
the translation problem:
VA => PA
and hope this translation
(i) runs fast,
(ii) has small memory overhead,
(iii) can be updated quickly.



- segmentation



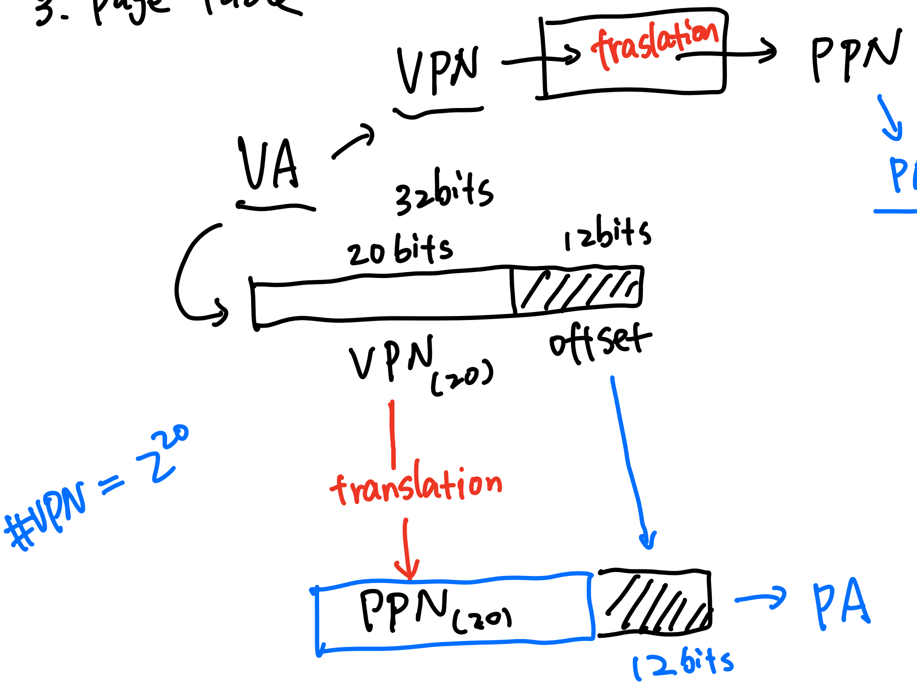
- pages

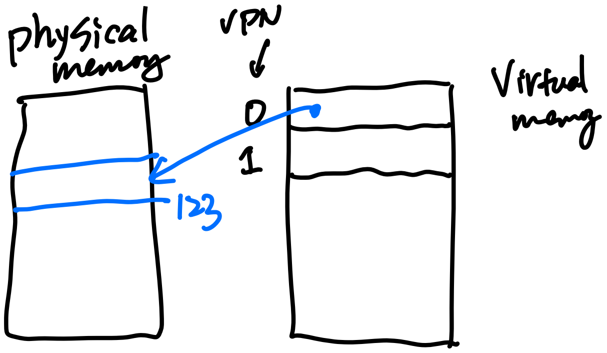


- who owns the translation?

per-process

3. page table

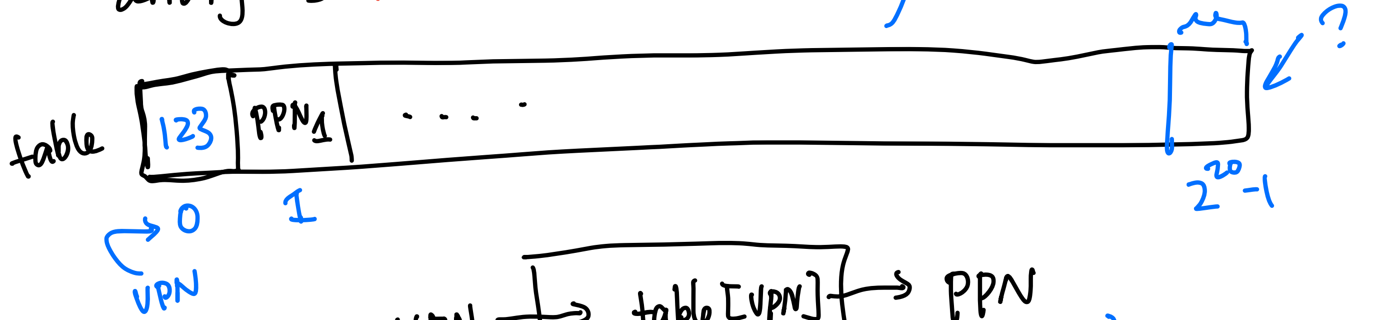


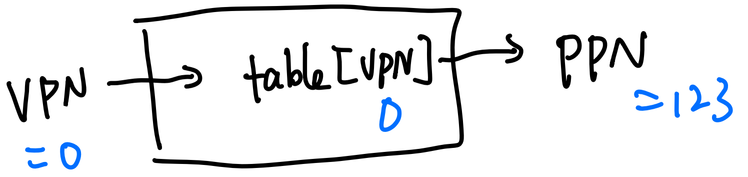


VPN=0 : [0, 4095]
VPN=1 : [4096, 8191]
⋮

size? $2^{20} * 4B = 4MB$

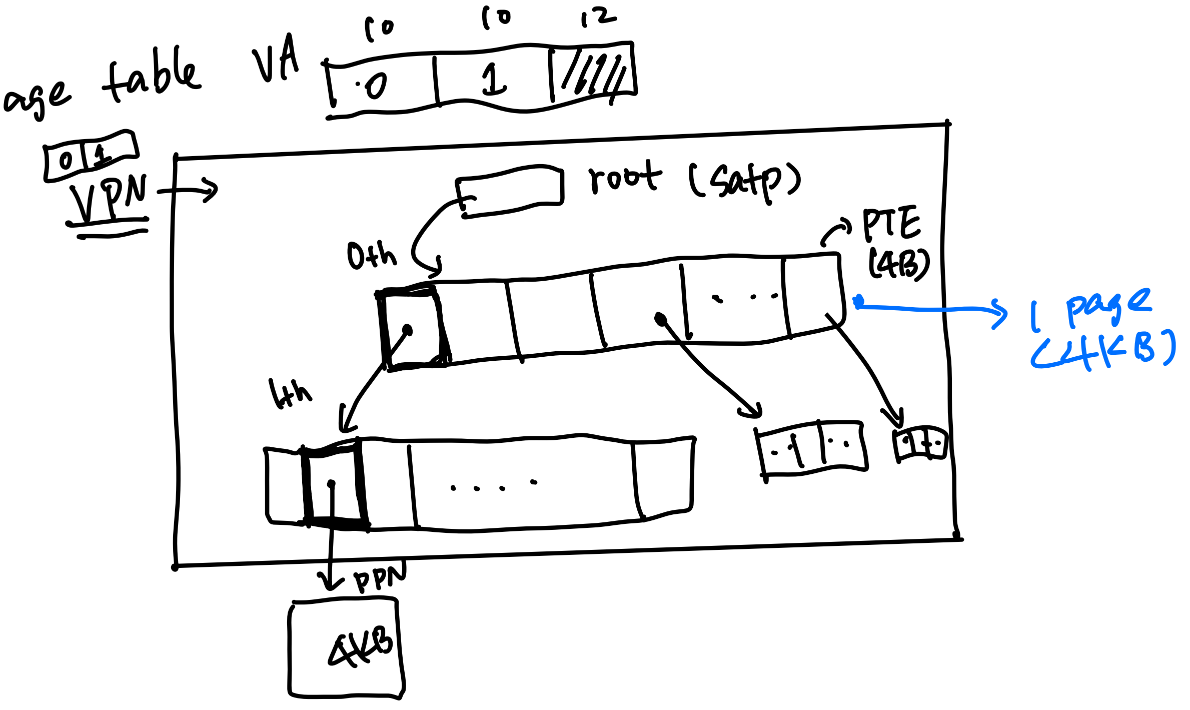
- array as translation



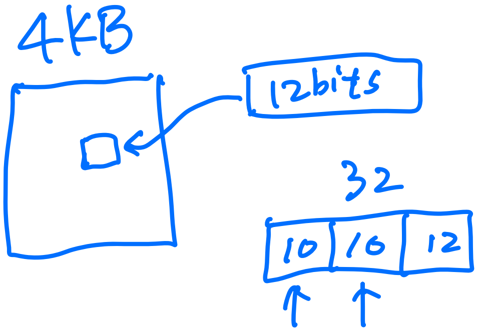


$2^{32} \to 4GB$

- page table



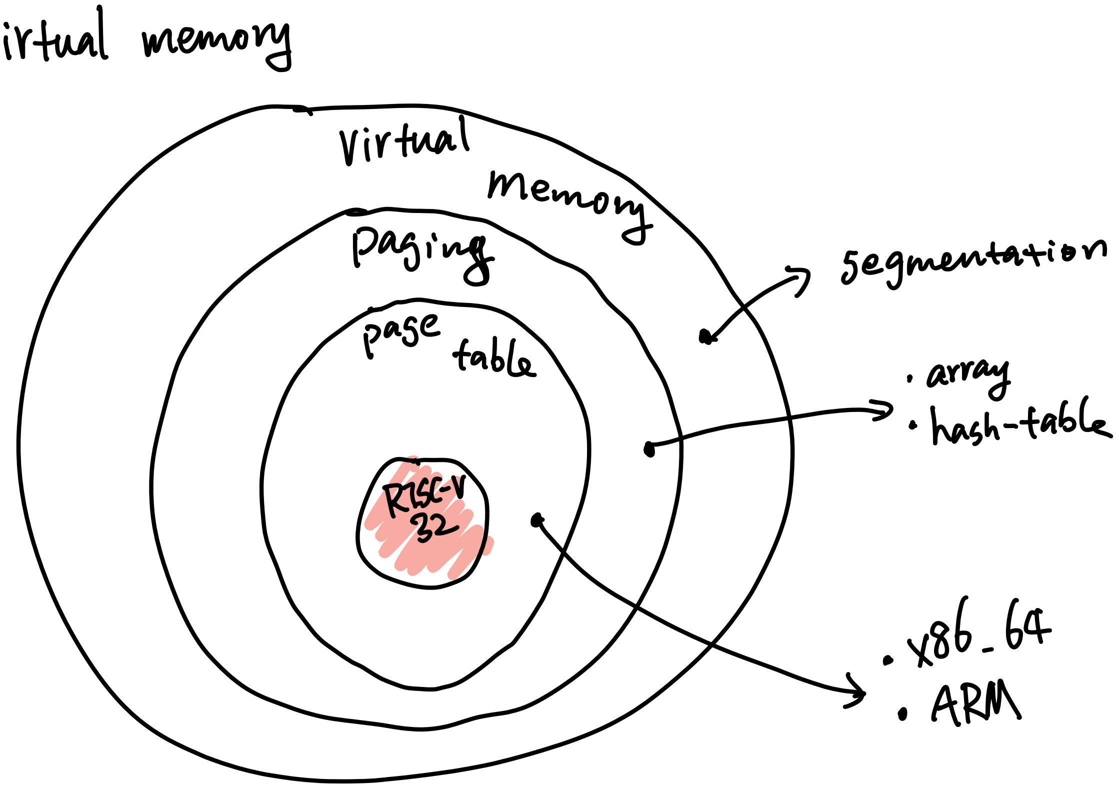
```
* PT design space:
  -- offset  (12bits)
  -- page size   (4KB)
  -- address length     (32bits)
  -- addressable memory unit  (1B)
  -- depth of the PT  (2 layer)
  -- PTE size  (4B)
```



$2^{10} \rightarrow 1024$

$1024 * 4B = 4KB$

4. today's virtual memory


Virtual memory
paging
page table
RISC-V 32
segmentation
• array
• hash-table
• x86_64
• ARM