- [x] 1. Last time
- [x] 2. Intro to virtual memory
- [x] 3. Paging
    - [x] Intro
    - [x] Key data structure: page table
    - [x] Multilevel page table
    - [ ] Alternatives/tradeoffs

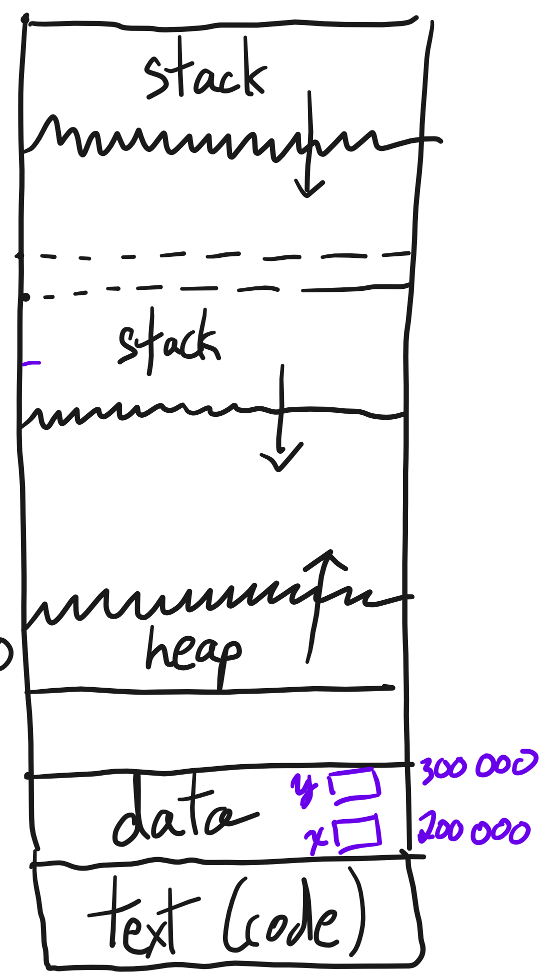
---

2. Intro to virtual memory

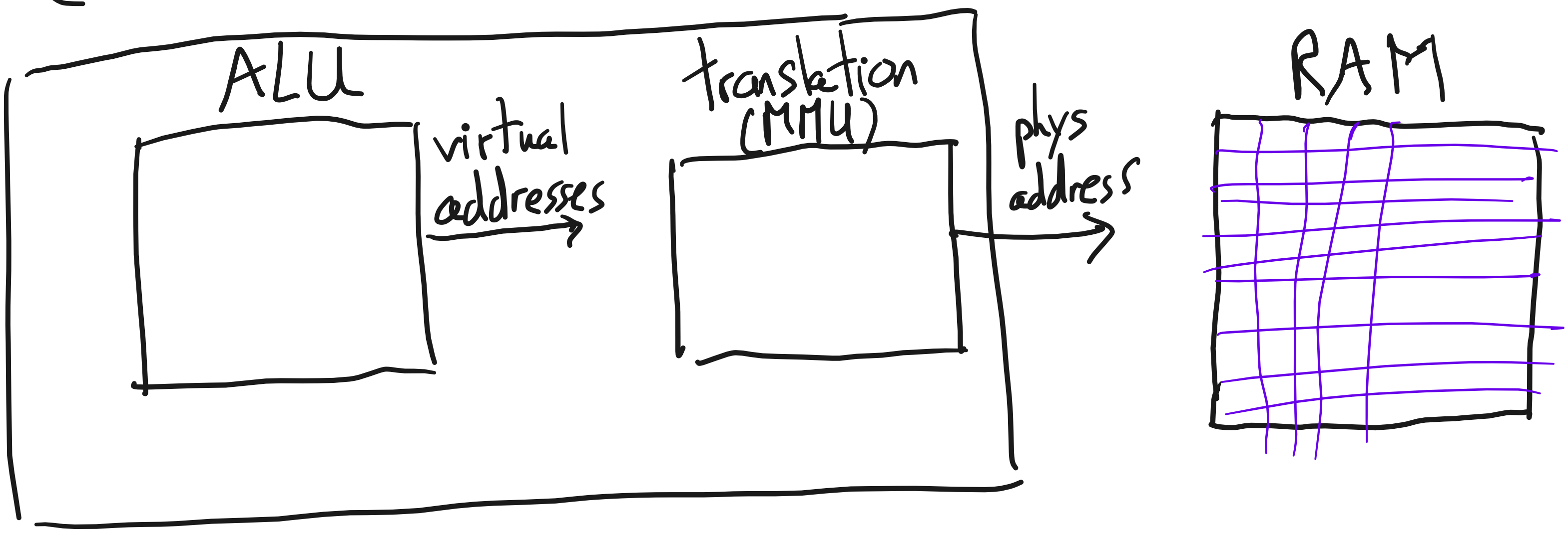
process "sees"

program excerpt: "y = x + 1"

code address	instruction
0x1500	movq 0x200000, %rax
0x1508	incq 1, %rax
0x1510	movq %rax, 0x300000



CPU



Benefits of virtual memory:

- programmability
  - (a) program <u>thinks</u> it has lots of memory
  - (b) programs can use "easy" addresses: compiler and linker don't have to worry about where program lives in physical memory
  - (c) multiple instances of a program can be loaded and not collide

- protection
  - processes cannot read/write each other's memory
  - enables isolation (which is essential)

- effective use of resources "ps"

- sharing

- — — — —

How is translation implemented?

- hardware does it, in MMU

OS sets up data structures that the hardware "sees".

These data structures are per-process.

3. Paging

A. Intro

- Divide memory (virtual + physical) into fixed-size chunks
- These chunks are called PAGES
- PAGE SIZE
- x86-64: 4096 B = 4 KB = 2^{12} bytes

8 bits = 1 byte

Aside:

2^{10}: kilo, ~1000

2^{20}: mega, ~1 million

2^{30}: giga, ~1 billion

2^{40}: tera, ~1 trillion

2^{50}: peta, ~1 quadrillion

2 : peta,

How many pages are there in a 32-bit (byte-addressed) architecture?

$$2^{32} \text{ bytes} \times \frac{1 \text{ page}}{2^{12} \text{ bytes}} = 2^{20} \text{ pages}$$

What if 48 bits are used to address memory?

$$2^{48} \text{ bytes} \times \frac{1 \text{ page}}{2^{12} \text{ bytes}} = 2^{36} \text{ pages}$$

$$= 2^{30} \times \underbrace{2^{6}}_{64}$$

Page 0: [0, 4095]

Page 1: [4096, 8191]

⋮

VPN PPN

⋮

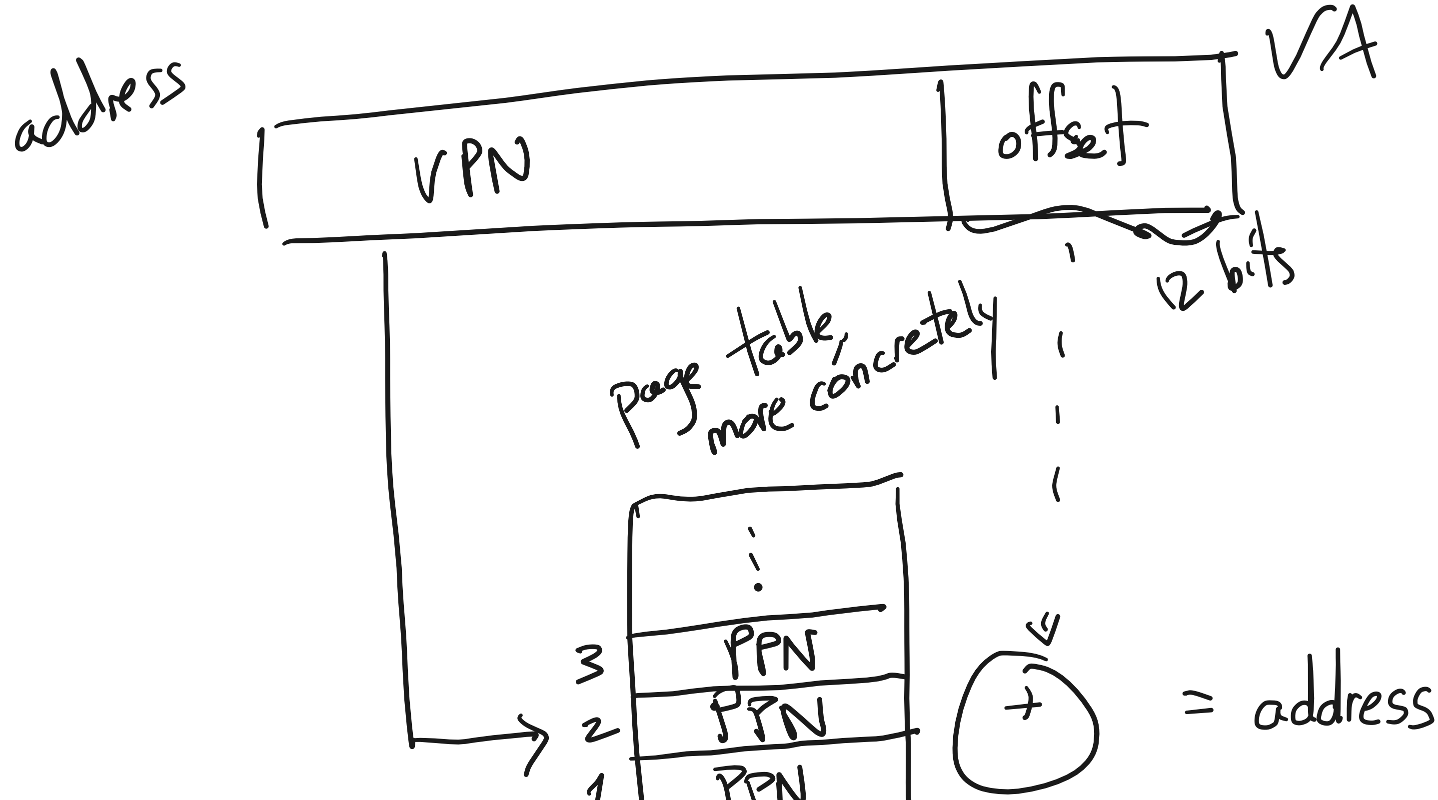
Page $2^{20}-1$:

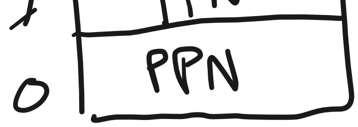
$[\ldots\ldots, 2^{32}-1]$

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

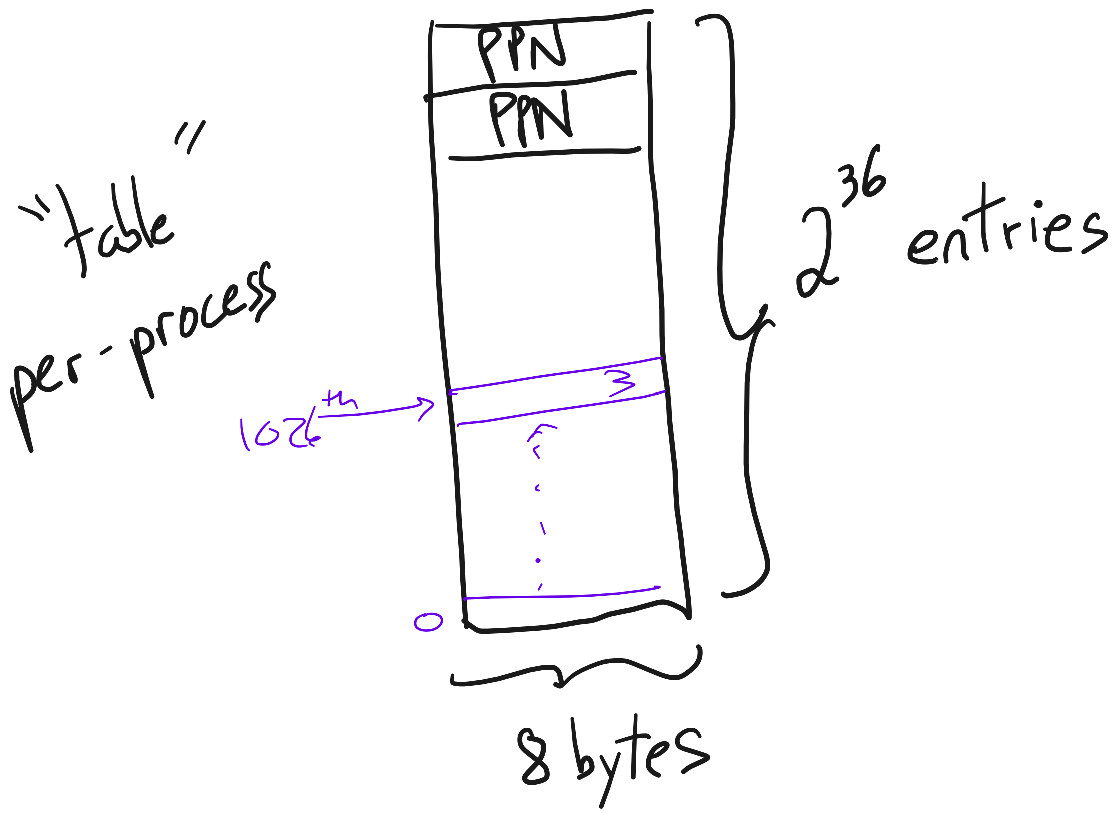
conceptually: a map from

VPN → PPN





assume: 48-bit addresses, and 4KB pages (2^{12} bytes)



Ex: OS wants a process to use address

VA: 0x00402000 to refer to

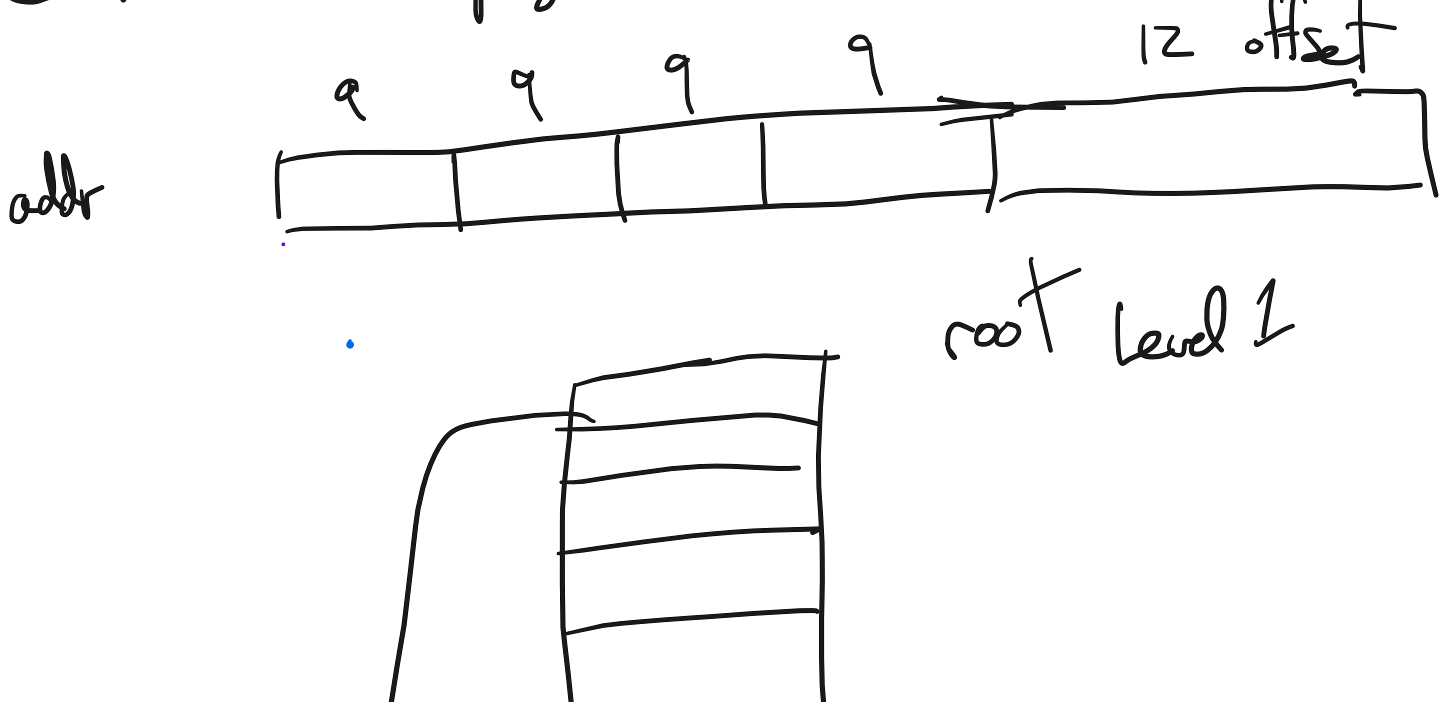
(36 | 12 bits)

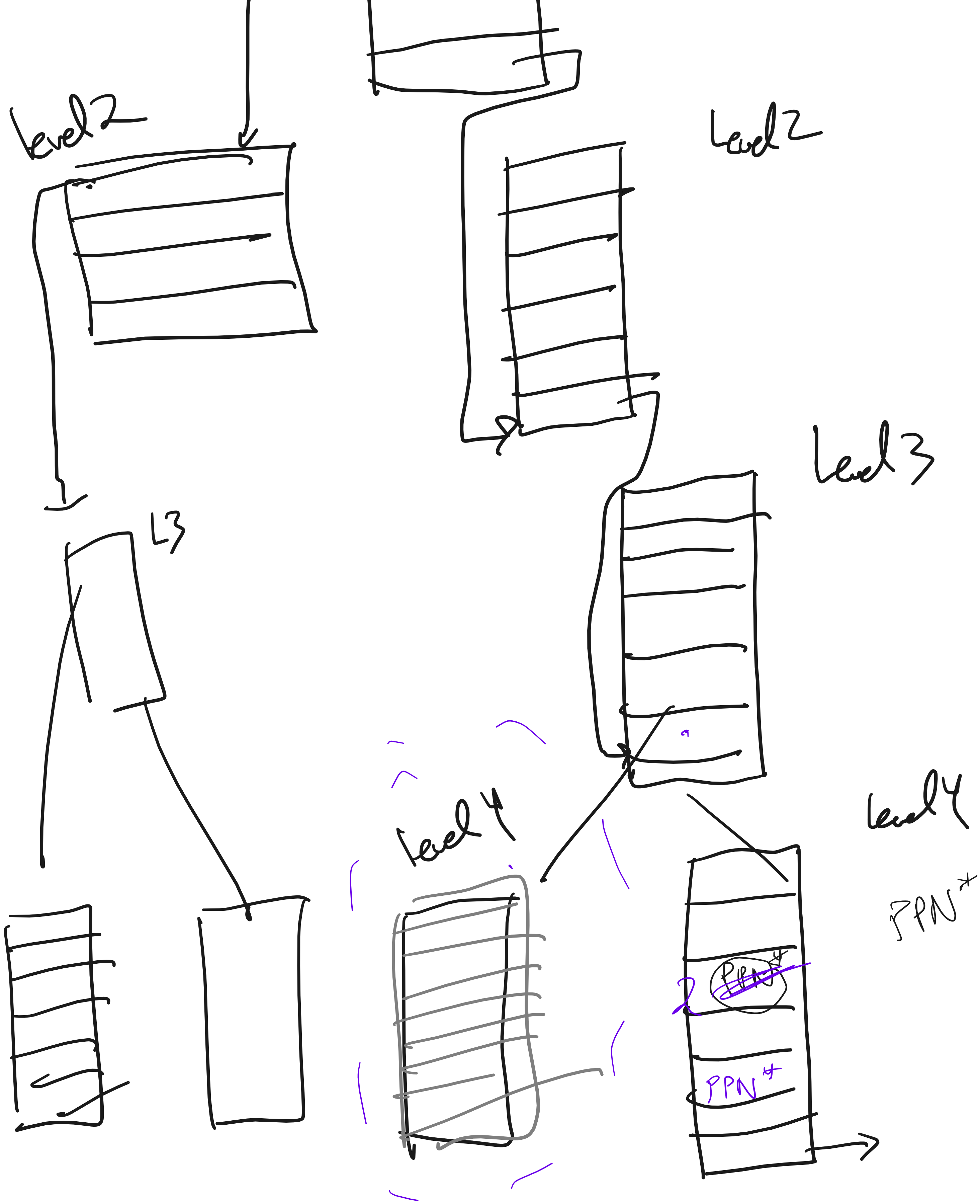
PA: 0x00003000

table[0x00402] = 0x00003

What's the issue?

C. Multilevel page tables





Ex: we want the physical address range [4MB, 6MB - 1] to appear at virtual address

48 ... 2^48 ...]

range $[2^{[illegible]} - 2MB, 2^{[illegible]} - 1]$