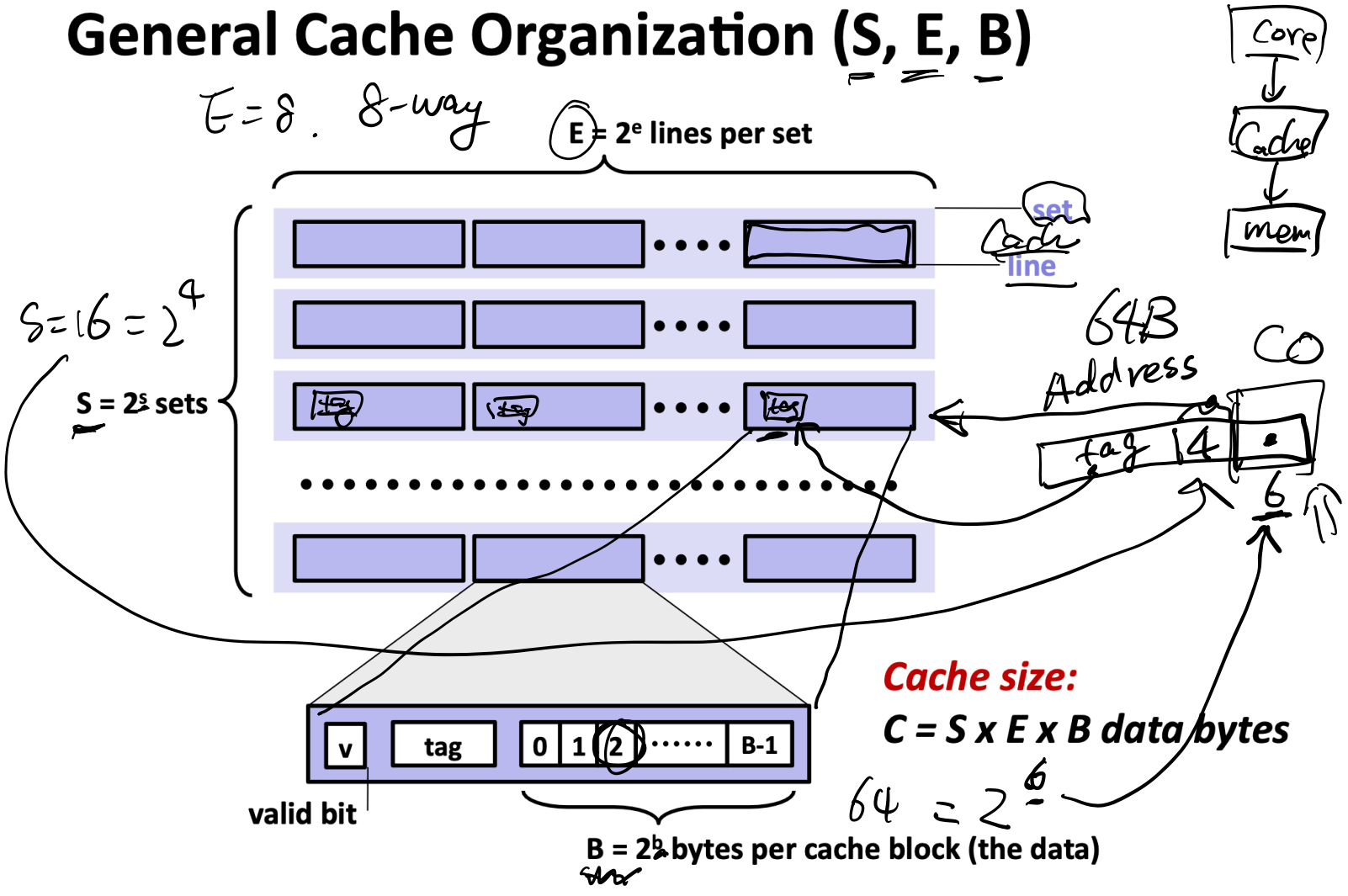
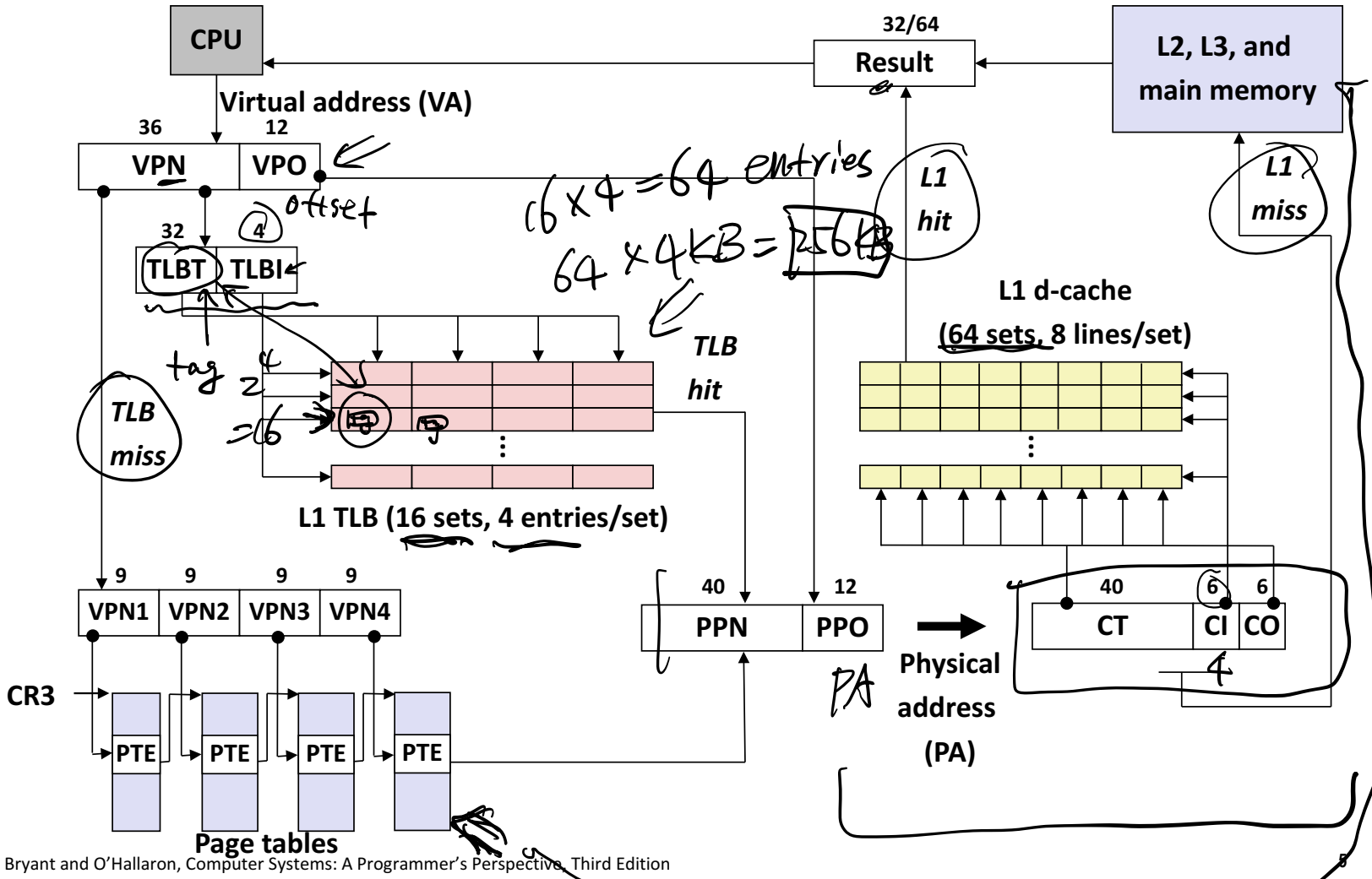
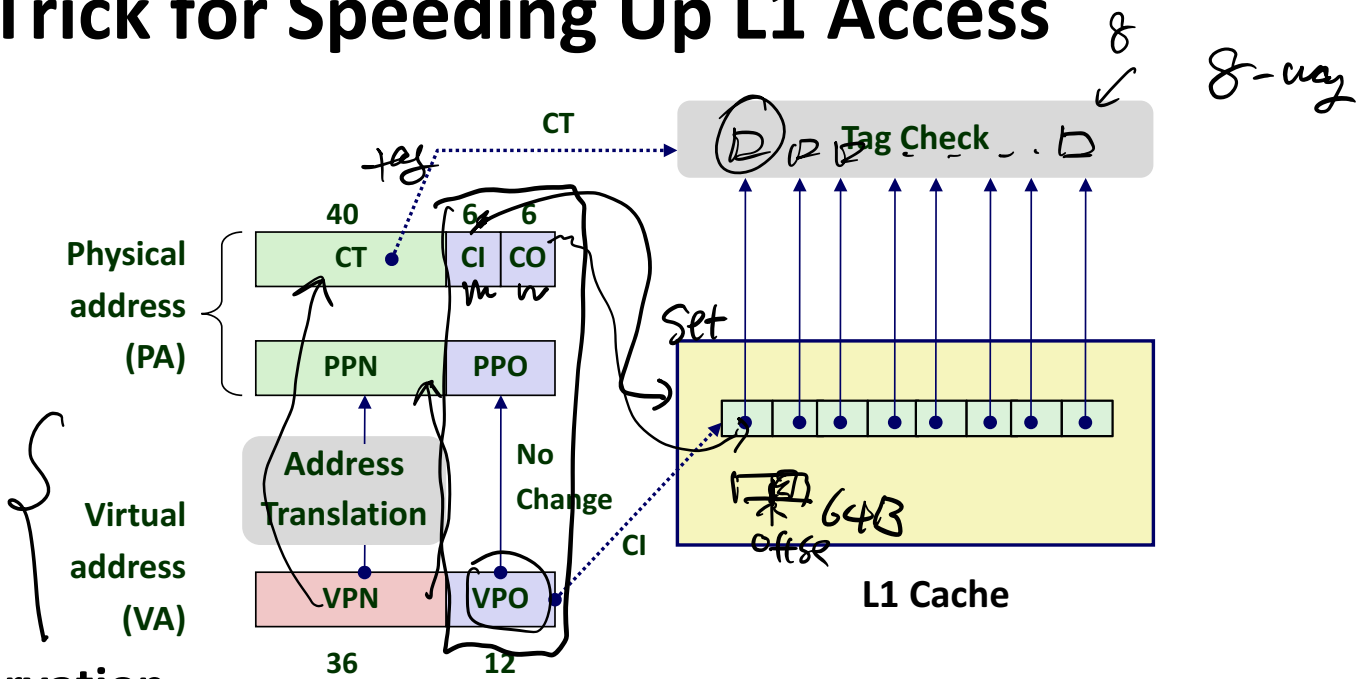
General Cache Organization (S, E, B)



End-to-end Core i7 Address Translation

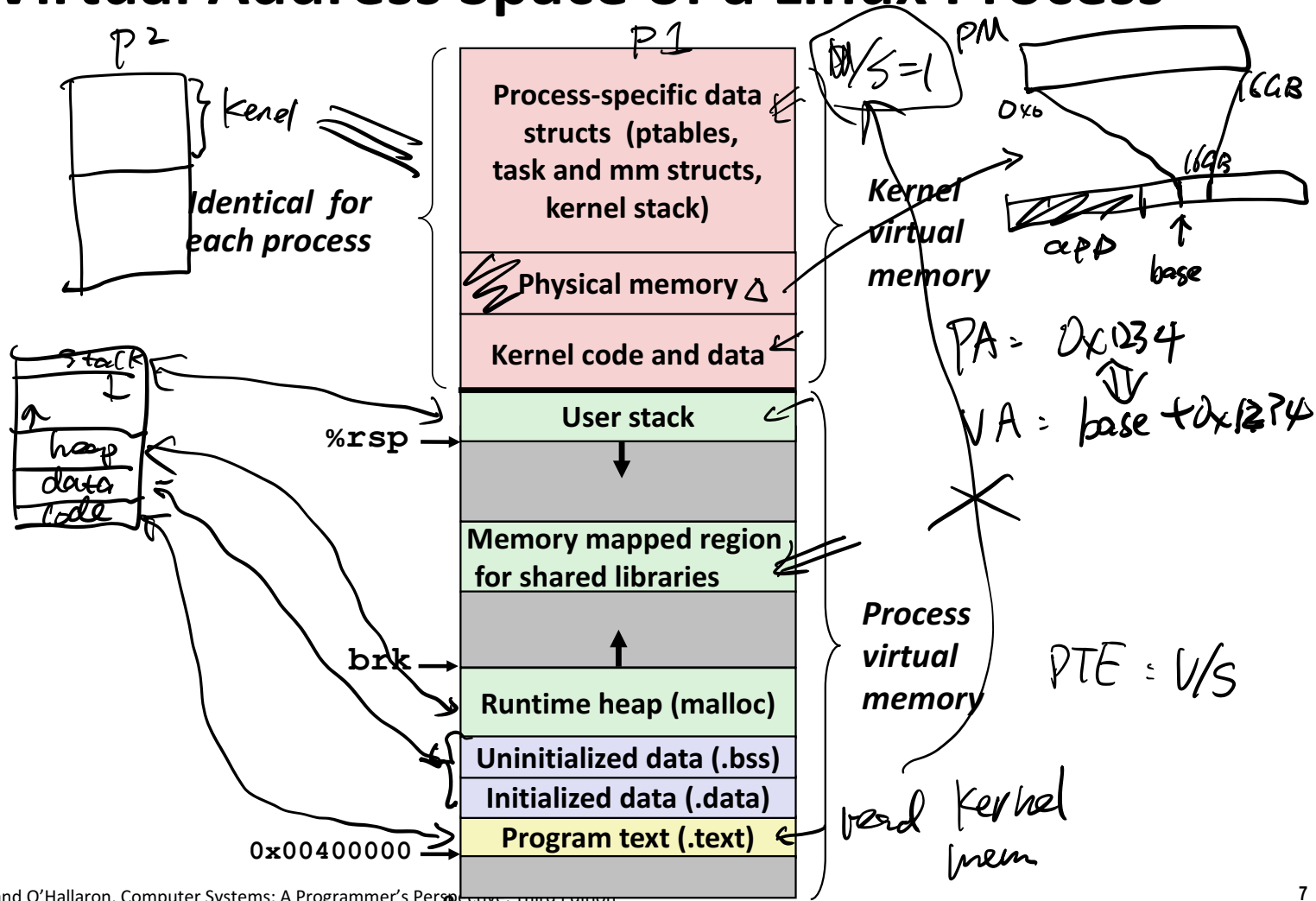


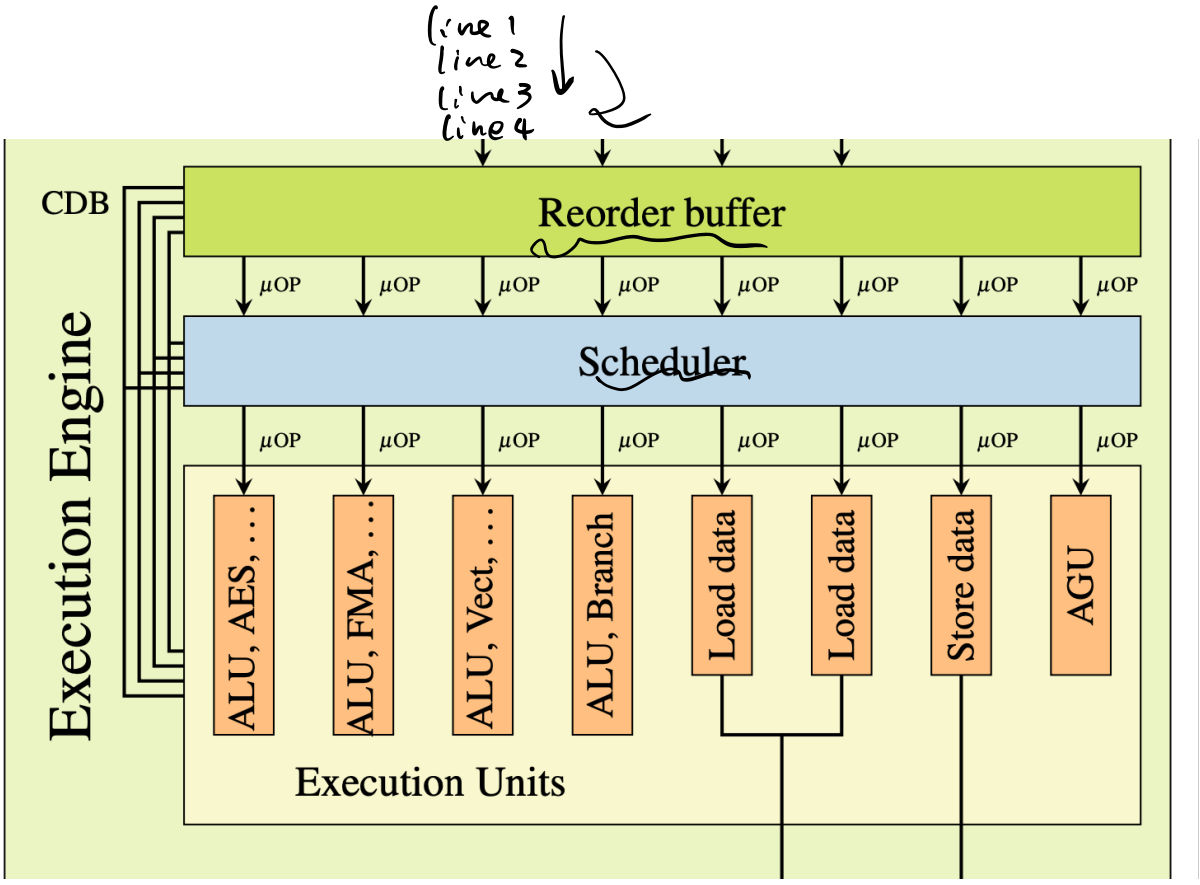
Cute Trick for Speeding Up L1 Access



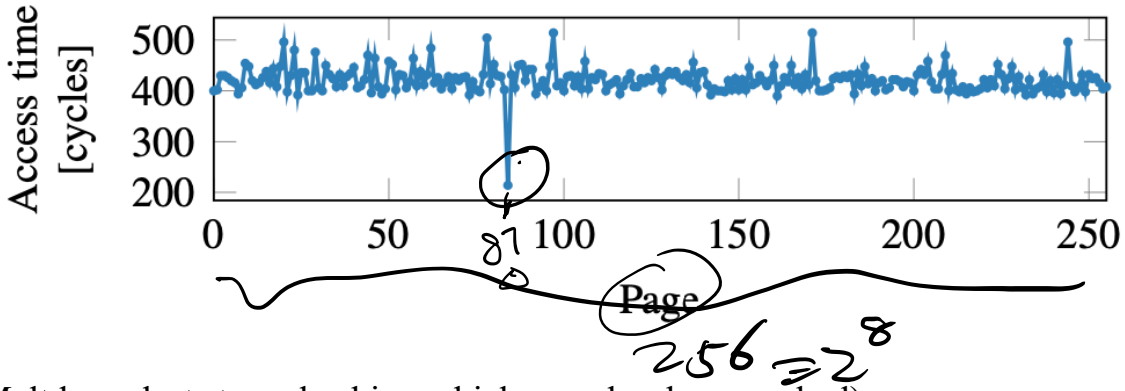
- **Observation**
  - Bits that determine CI identical in virtual and physical address
  - Can index into cache while address translation taking place
  - Cache carefully sized to make this possible: 64 sets, 64-byte cache blocks
  - Means 6 bits for cache index, 6 for *cache* offset
  - That's 12 bits; matches *VPO, PPO* → One reason pages are 2^{12} bits = 4 KB

Virtual Address Space of a Linux Process





(Partial view of CPU internals: execution engine)



(Meltdown last step: checking which page has been cached)

Figures borrowed from Meltdown paper.

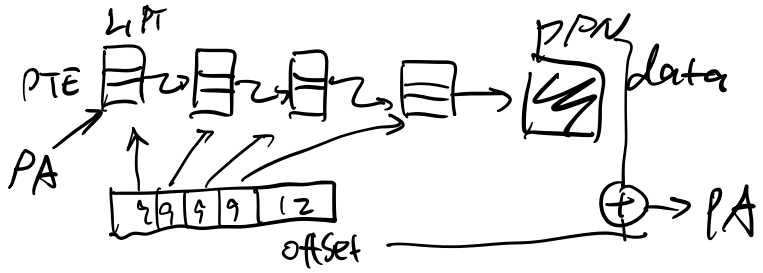
1. TLBs
2. Where does the OS live?
3. Meltdown and Spectre
4. page faults

WUH 332

- Last time

VA ⟶ PA

(48 bits) (52 bits)



Q: root?

%cr3.

PA

Q:

0x1234: movq (0xbebeebee), %rax

VA

2, 8, 5, 1,

- VA → PA for 0x1234 : 4 + 1 = 5

VA → PA for 0xbebeebee : 4 + 1 = 5 + 5 = 10

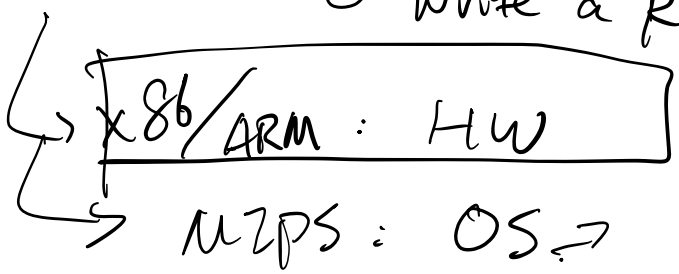
"Cache" ⇒ TLB <VA→PA>, U/S, R/W ...

TLB miss v.s. Page fault (PTE P=0)

Q.: TLB miss $\overset{?}{\Longrightarrow}$ PF (NO)

Q: PF $\overset{?}{\Longrightarrow}$ TLB miss (NO)

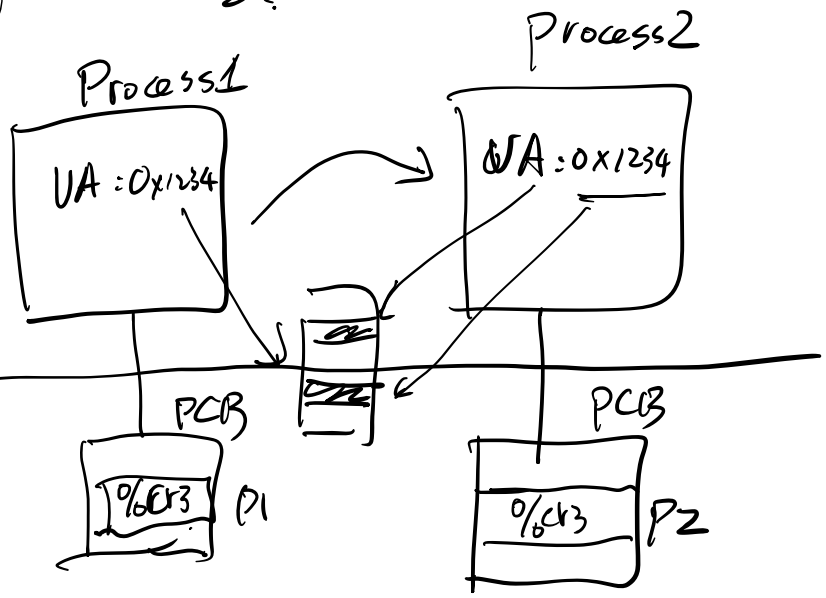
- Who fill TLB? ① VA → PA (PTE P=0) ② write a RO page



Q: TLB:

OS flush all TLB?

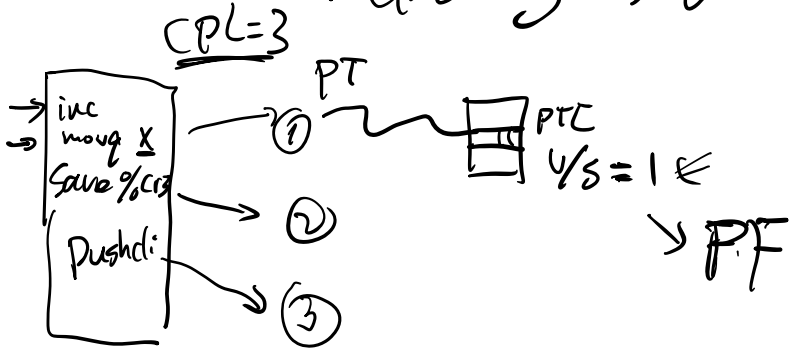
- yes ✓
- NO



Kernel v.s. app

CPU state.

CS → 2bits { CPL=0 ⇒ kernel mode
CPL=3 ⇒ user mode



Where is OS? 1% – 80% Security

⭐ ⇒ OPT 1: OS → Address space

OPT 2: OS ⇝ same address space with processes.

"""
Q: Am I affected by the vulnerability? ← meltdown spectre
A: Most certainly, yes.

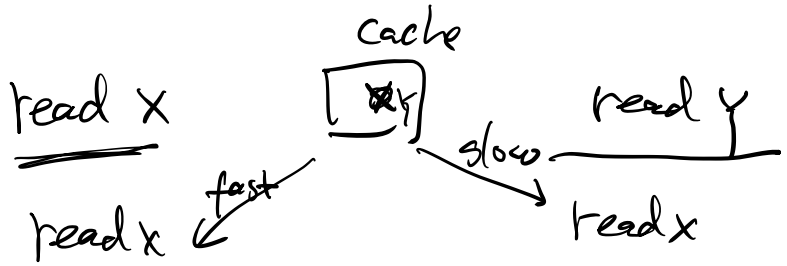
Q: Can I detect if someone has exploited Meltdown or Spectre against me?
A: Probably not. The exploitation does not leave any traces in traditional log files.

Q: What can be leaked?
A: If your system is affected, our proof-of-concept exploit can read the memory content of your computer. This may include passwords and sensitive data stored on the system.

Q: Has Meltdown or Spectre been abused in the wild?
A: We don't know.
"""

background

(1) side-channel attack (cache)

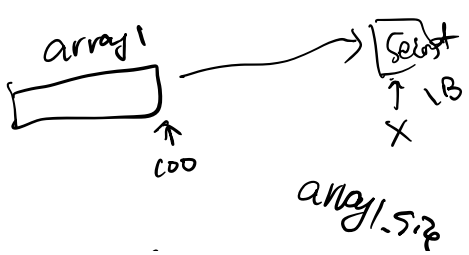


(2) speculative exec.

mem ← 500 cycles

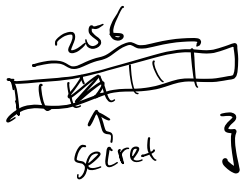
```
if ( read bool from ) {
    foo ();
}
```

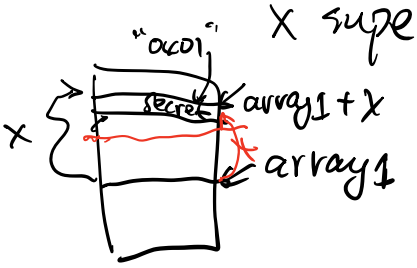
{ bool False: discard all state from foo()
  bool True: ☺



- spectre attack

```
if ( x < array1_size) { // if x >>>
    y = array2 [ array1[x] * 4096 ];
}
```





x supe large, s.t. array1+x points to secret.

0x01

toched array2[0x01 * 4096]

Recover secret:
```
for ( int i=0; i<256; i++) {
    test read latency of array2[i*4096];
}
```

i = 0x01

- meltdown.

000 Exec + side-channel