Overview of system security - focusing on Meltdown

01 Sys-protection

02 MeltDown

03 HyperSafe

04 Conclusion

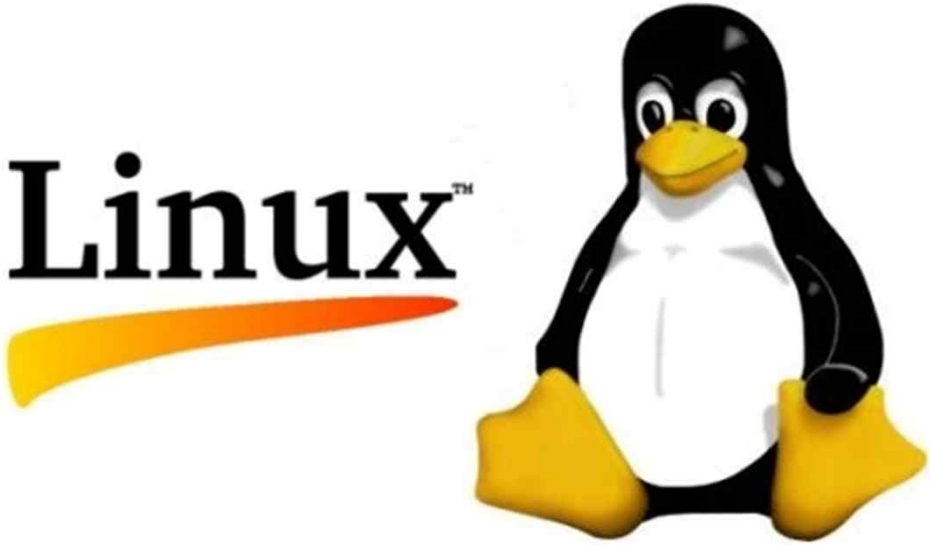
01 What is System Protection?

01 What is system protection?

System-level programming

There are some examples..
It usually means OS and HW design.

Intel’s CPU design

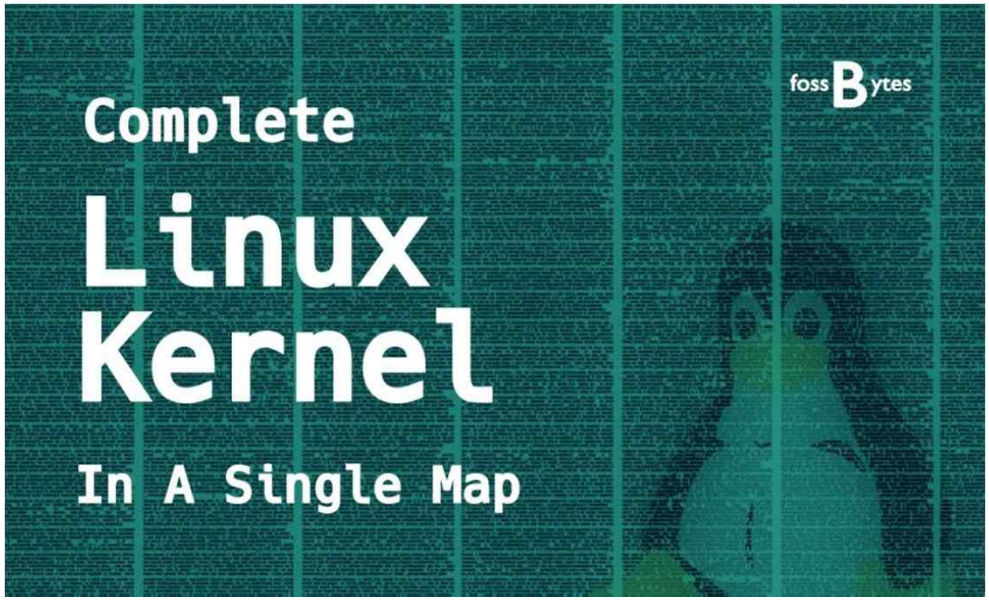


Linux: Open source OS

01 What is system protection?

This is not natural science!

They all designed by human!
Most of them are developed by some geniuses.
However, they can not care all thing.



There are 240,000 lines of li
Source code..

01 What is system protection?

It’s not always perfect..

There are many vulnerability..
Ex) Dirty cow, rootkit, careless of users…

But, these are minor things and not that important.

We have to block them from our private information.
Information is stored in memory.

02 MeltDown

02 MeltDown

Privilege is important

Usually, there are two mode to protect invalid access to Private memory.

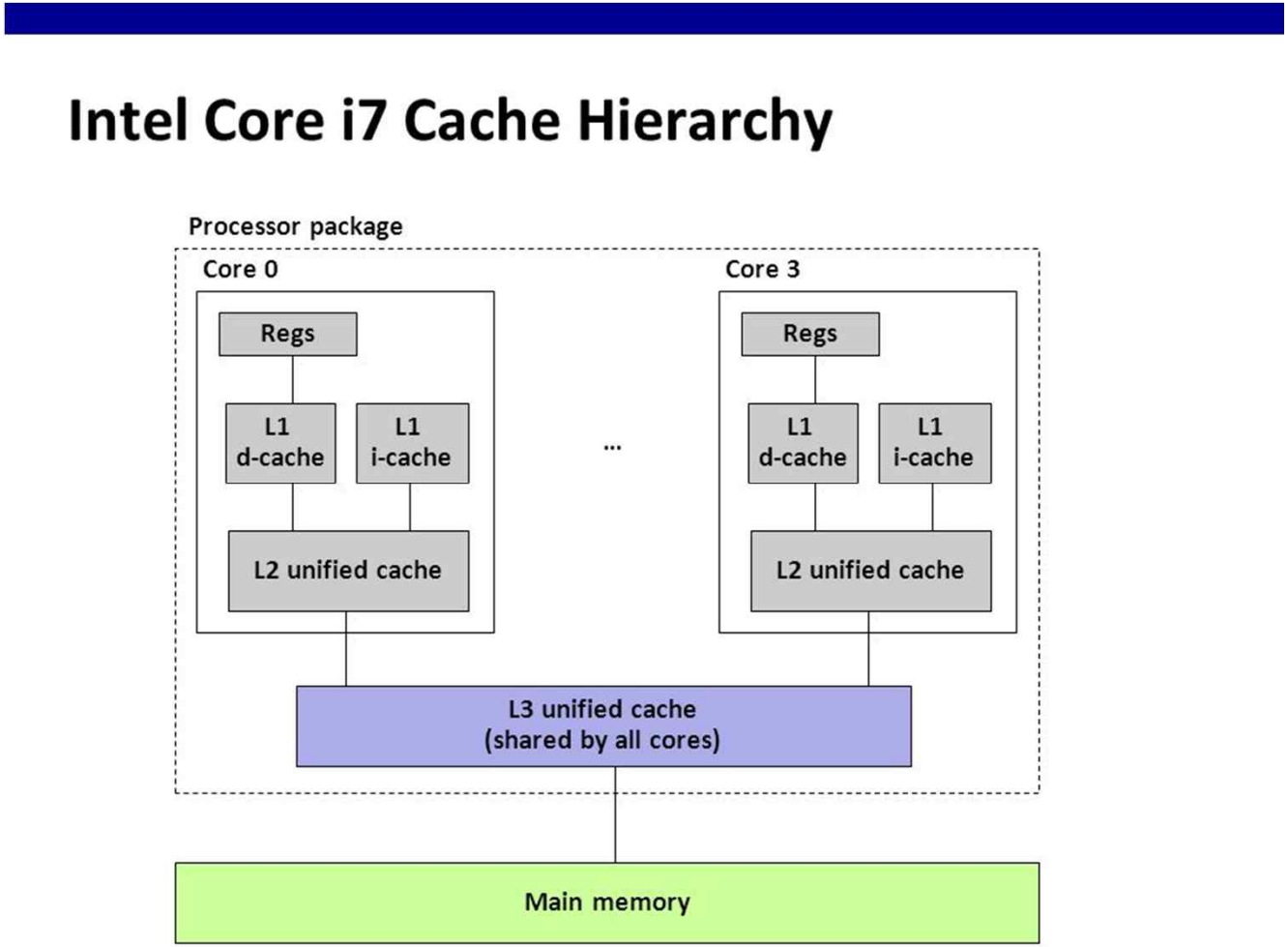
Kernel mode – can access to any kind of instruction, dat
User mode – can not access to OS's memory. (User mus
believe OS or HW. And all of their permission must be passed..

02 MeltDown

Privilege is important

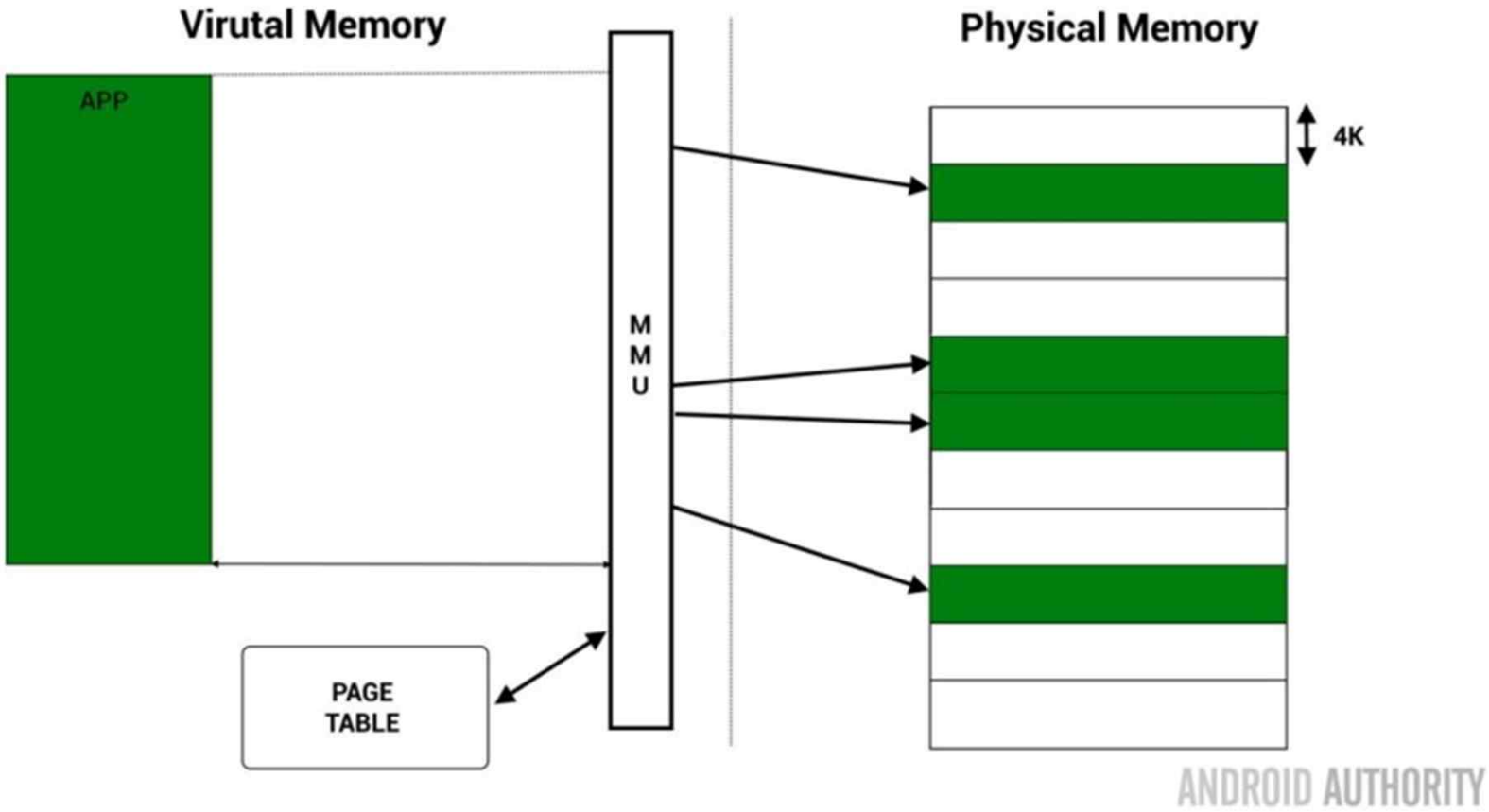
02 MeltDown

Computer basic - cache



02 MeltDown

Computer basic - VM



02 MeltDown

Computer basic – OoO superscalar CPU

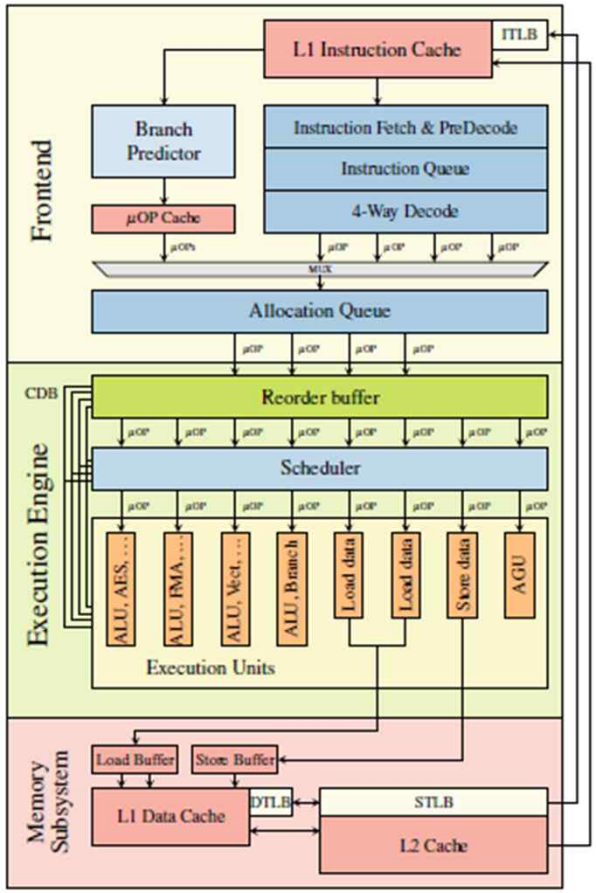


Figure 1: Simplified illustration of a single core of the Intel's Skylake microarchitecture. Instructions are decoded into μOPs and executed out-of-order in the execution engine by individual execution units.

Reference from https://meltdownattack.com - "Meltdown"

02 MeltDown

Key exploit

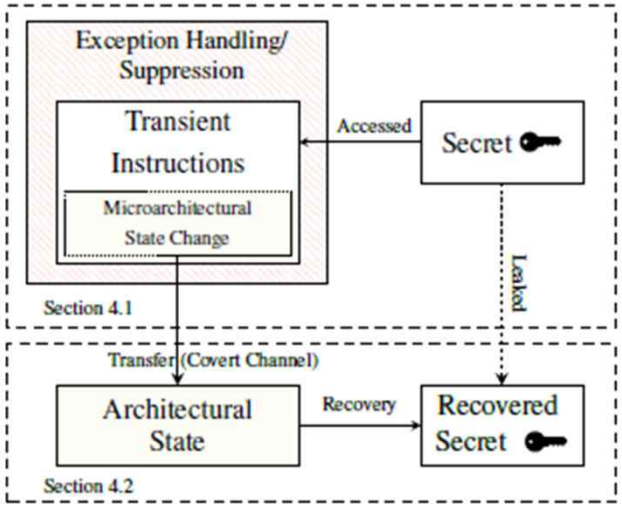


Figure 5: The Meltdown attack uses exception handling or suppression, e.g., TSX, to run a series of transient instructions. These transient instructions obtain a (persistent) secret value and change the microarchitectural state of the processor based on this secret value. This forms the sending part of a microarchitectural covert channel. The receiving side reads the microarchitectural state, making it architectural and recovering the secret value.

Reference from https://meltdownattack.com
- “Meltdown”

02 MeltDown

Key exploit

```
; rcx = kernel address
; rbx = probe array
retry:
mov al, byte [rcx]
shl rax, 0xc
jz retry
mov rbx, qword [rbx + rax]
```

ng 2: The core instruction sequence of Meltdown. naccessible kernel address is moved to a register, ng an exception. The subsequent instructions are dy executed out of order before the exception is d, leaking the content of the kernel address through ndirect memory access.

Reference from https://meltdownattack.com - “Meltdown”

02 MeltDown

Key exploit

We can protect attack by software patch.

But, CPU slowed down on computers that require some higher throughput than PC.

Modern architecture faced big problem. SPEED vs SECURITY?

03 HyperSafe

03 HyferSafe

Hypervisor

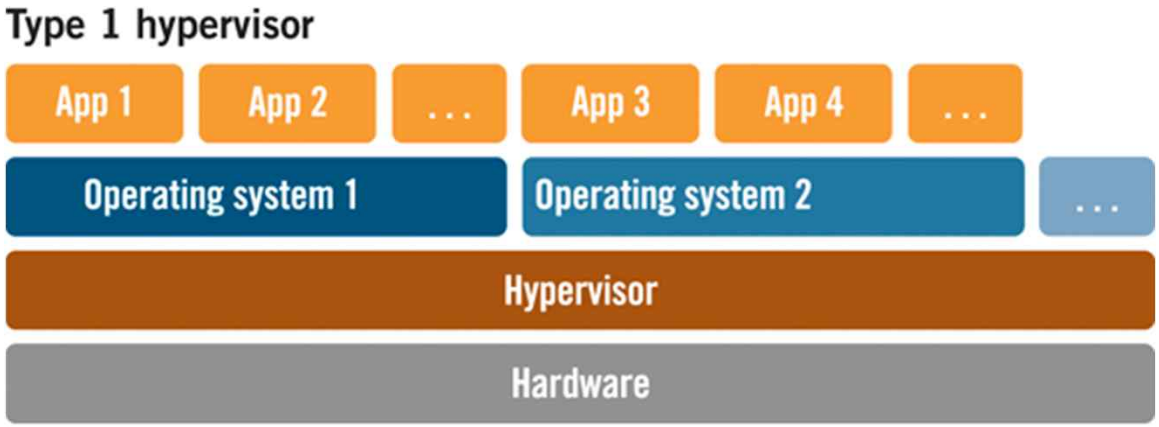


Figure 2. A Type 1 or bare-metal hypervisor sits directly on the host hardware.

What happens if attacker gains hypervisor's privilege?
How can we prevent that attacker cannot bread down our system? → principle of least privilege

03 HyferSafe

WP-bit

What is WP-bit?

- **Invented to make COW easy**
- **When WP-bit is on, memory is read-only.**
- **Privileged instruction. API is given to user by syscall.**

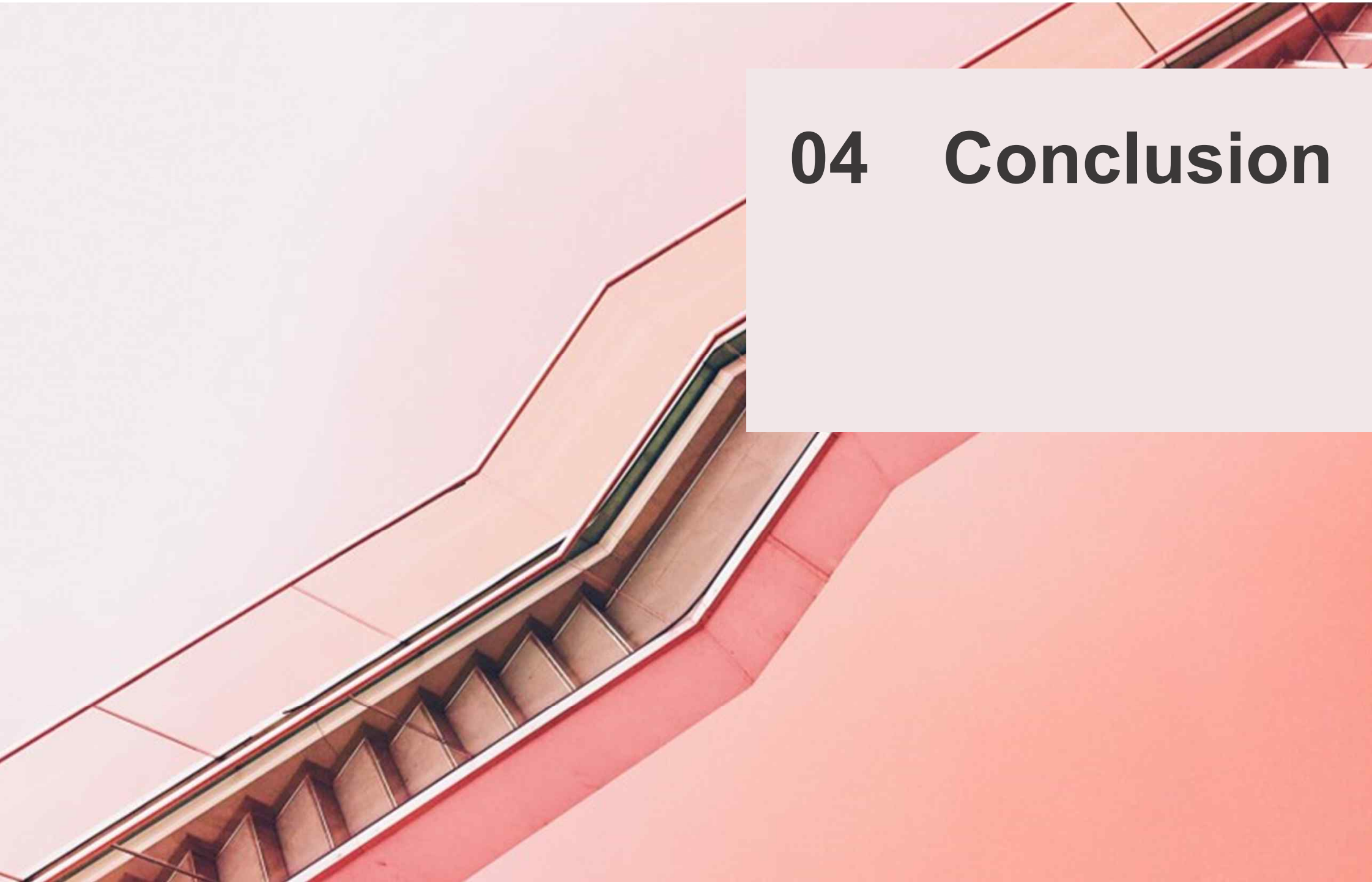
03 HyferSafe

WP-bit

We have to protect our pages from attackers changing to their intentions.

Locate all page table into physical memory's some section.
Turn on WP-bit on that section. And if invalid access to WP-bit occurs, block it.

I'm implementing it now, but it's tooooooooo hard.



04 Conclusion

04 Conclusion

System designers are not omnipotent. Let's be suspicious.

Most of hacking is caused by user's insult. We have to recognize user's non-expertise and let's consider!

But the things I mentioned earlier are fatal things that can break down all system's roots.
We have to consider security. Not just speed up.

04 reference

Meltdown and spectre - https://meltdownattack.com/

“HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity” by zhi wang, xuxian Jiang

“FLUSH+RELOAD: a High Resolution, Low Noise, L3 Cache Side-Channel Attack”

Questions?