

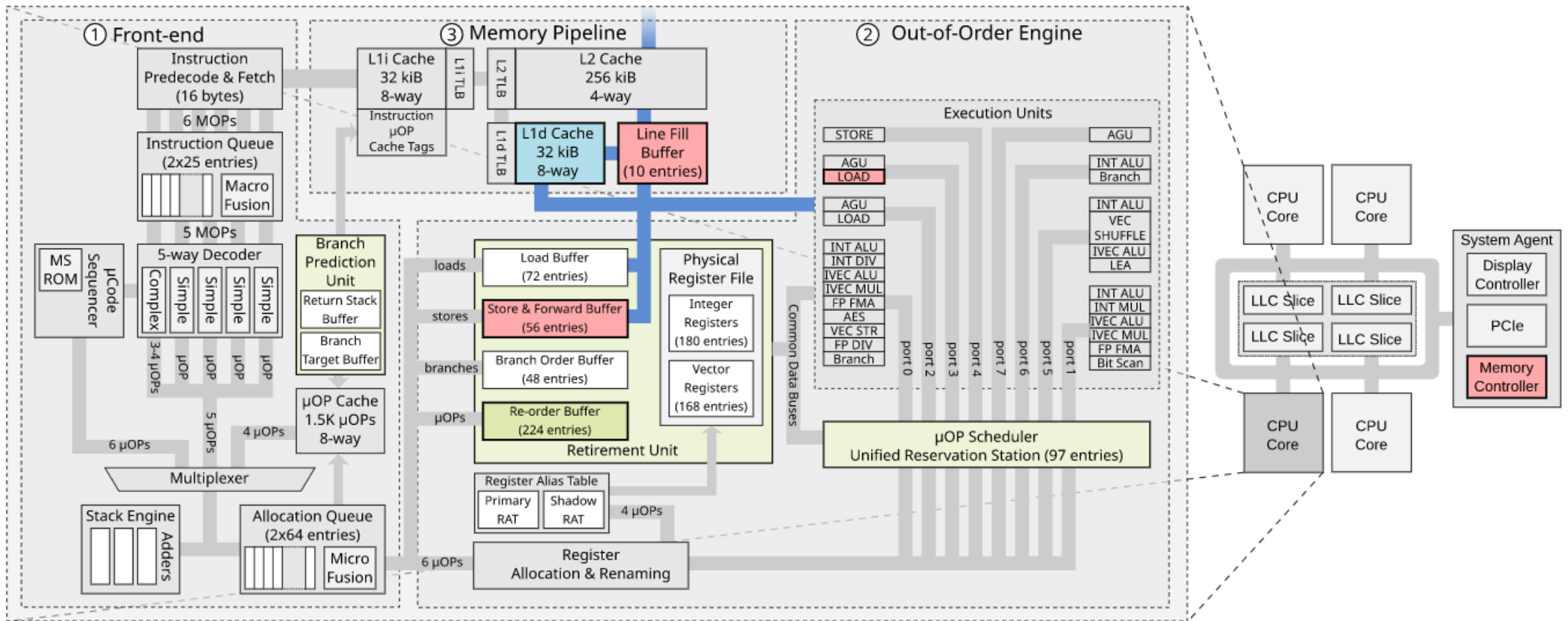
Meltdown & Spectre

MonKey Lee

Ruilin Wen

2012-04

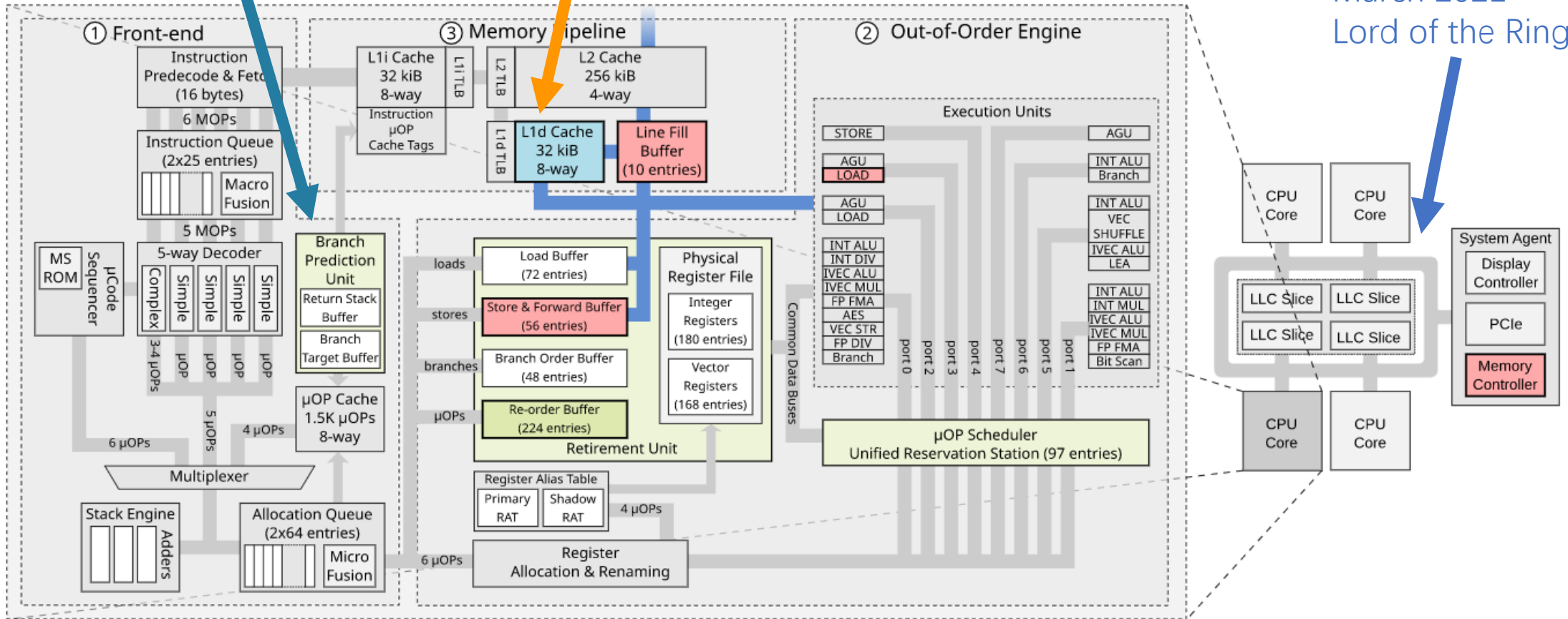
Skylake Microarchitecture (from mdsattacks.com)



Skylake Microarchitecture (from mdsattacks.com)



March 2021
Lord of the Ring(s)



Out-of-order Processor & Speculations (1)

- Out-of-order Execution
 - Tomasulo (CPU) & Scoreboarding (GPU)
 - Maximizing the utilization of all execution units of a CPU core as exhaustive as possible

- Tomasulo

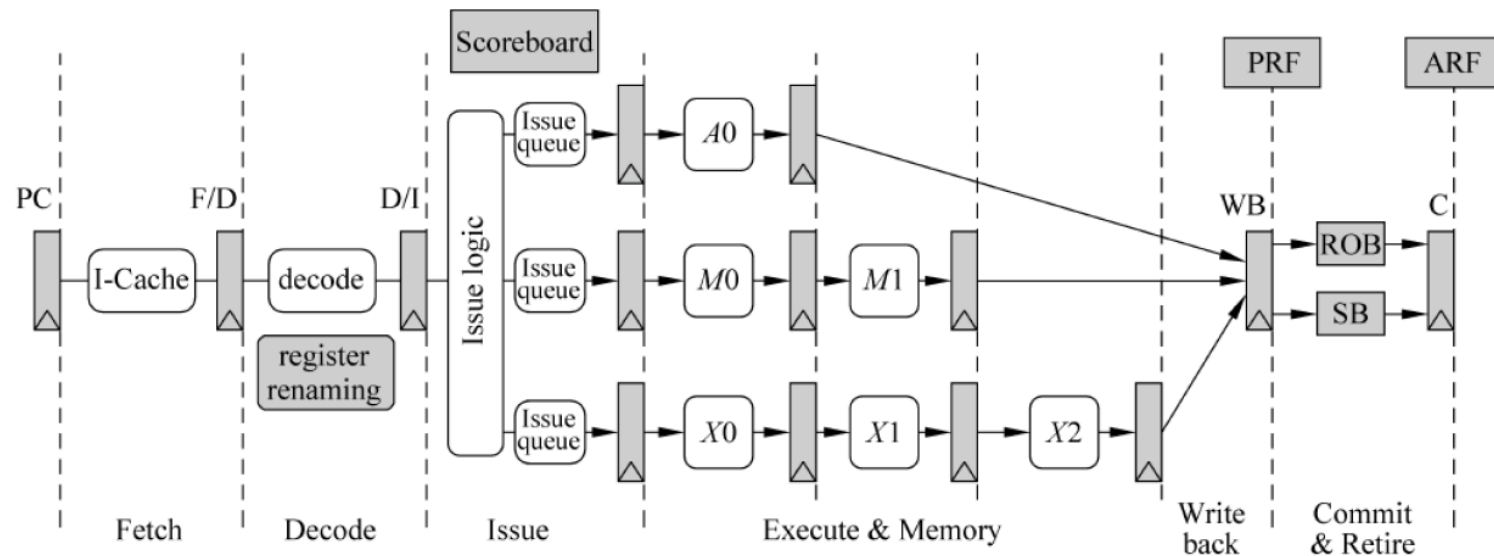


图 1.14 乱序执行的超标量处理器的流水线

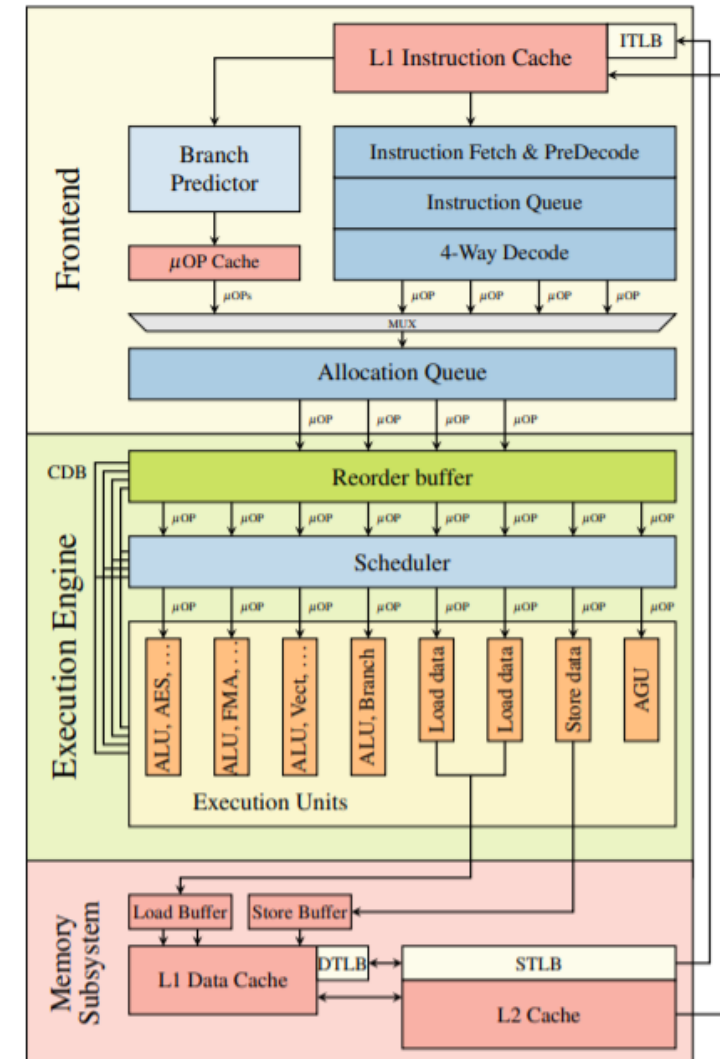
Out-of-order Processor & Speculations (2)

Instructions are

- fetched and decoded in the **front-end**
- dispatched to the **backend**
- processed by **individual execution units**

Instructions

- are executed **out-of-order**
- wait until their **dependencies are ready**
 - Later instructions might execute prior earlier instructions
- **retire in-order**
 - State becomes architecturally visible
- **Exceptions** are checked during retirement
 - Flush pipeline and recover state



Out-of-order Processor & Speculations (3)

- To keep the execution part busy, we need:

- Speculations

- Control-flow speculation

- Branch Taken/Not-taken

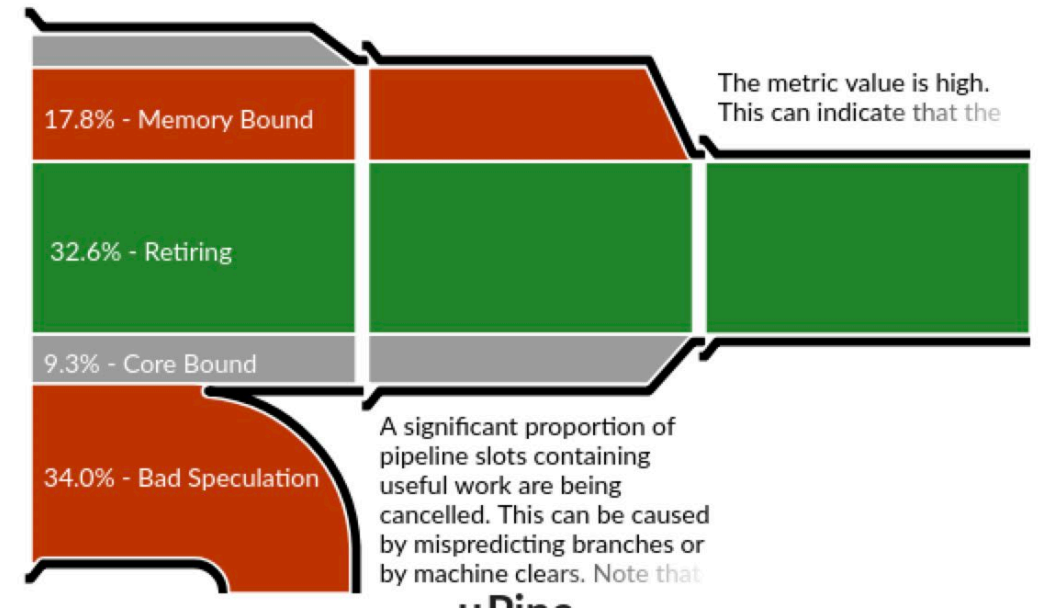
- Branch Target

- Delayed Exception Handling

- Data-flow speculation

- Predict a value for the program to carry on

- (AMD Predictive Store Forwarding)



- Ideally, mis-predicted are withdrawn later

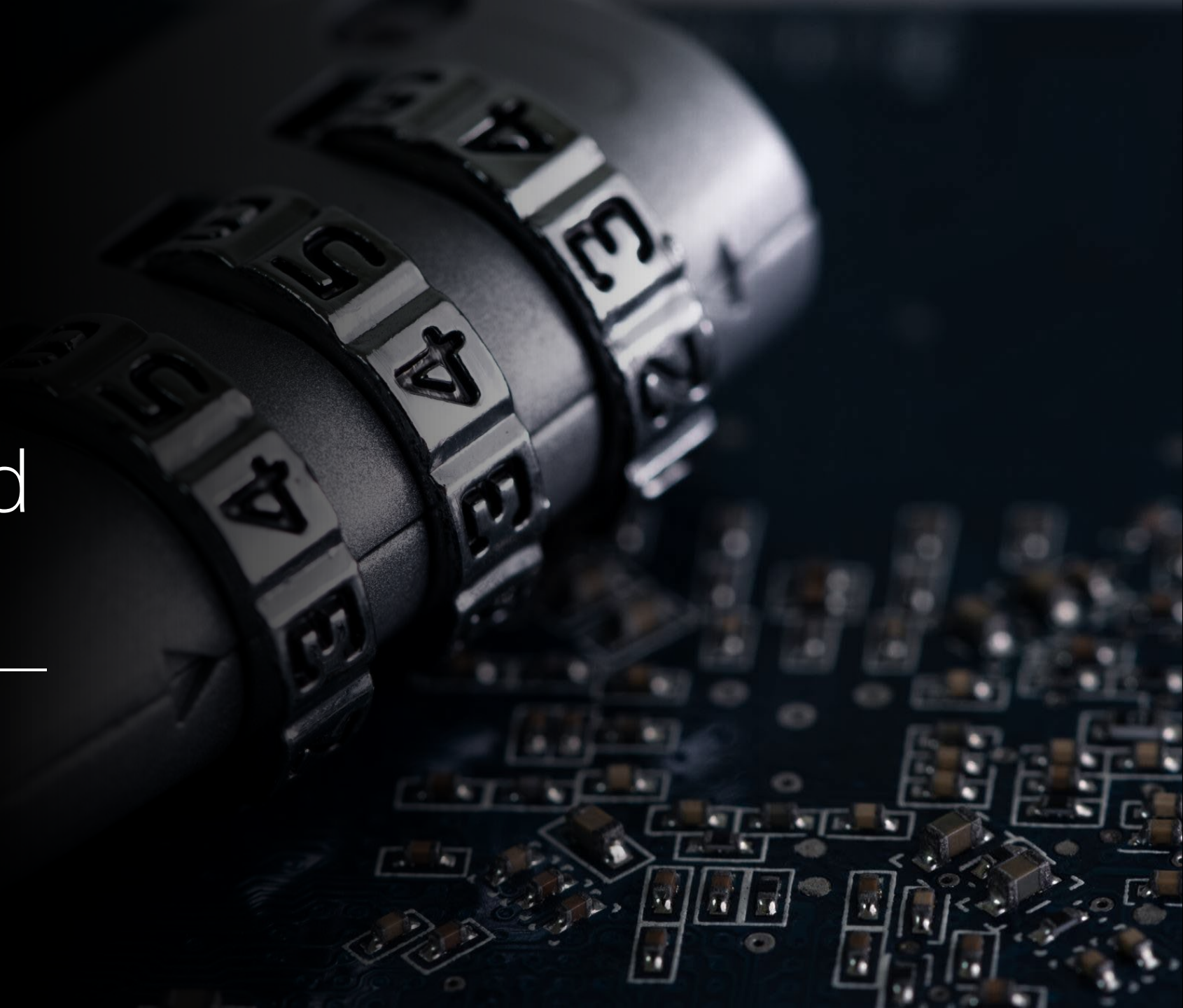
- But it has FLAWS.

Spectre and Meltdown Variants





Flush + Reload Attack



Side-channel Attack

- Safe software infrastructure does not mean safe execution
- Information leaks because of the **underlying hardware**
- Exploit **unintentional information leakage by side-effects**



Power
consumption

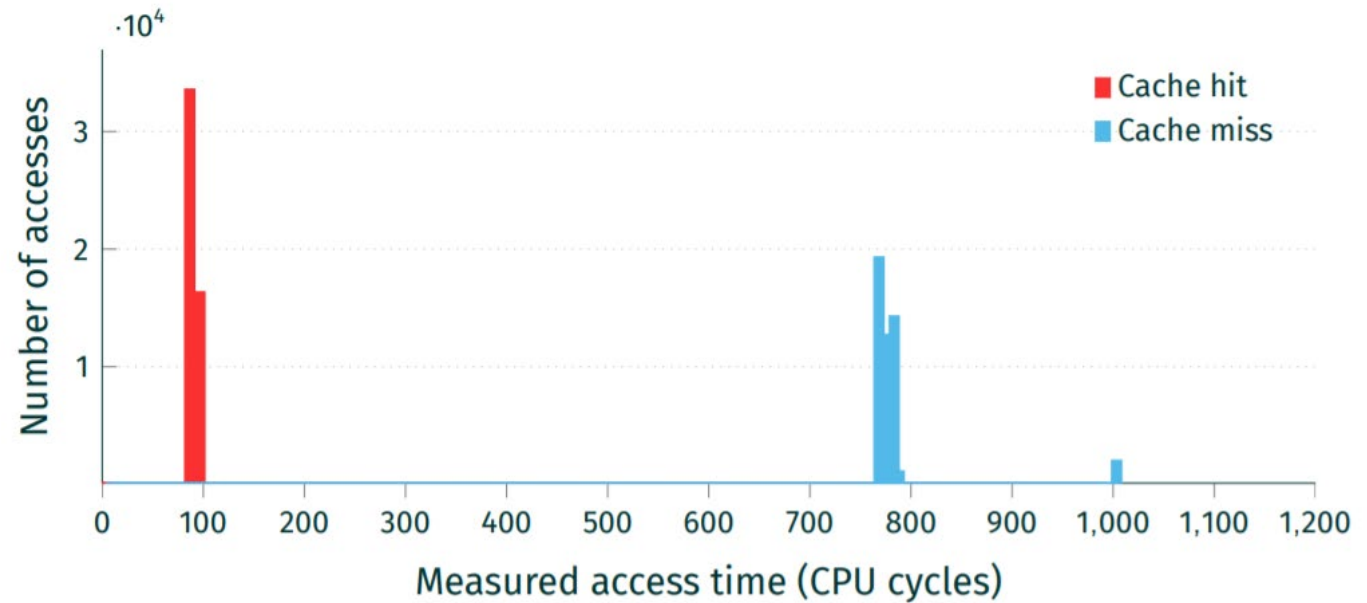
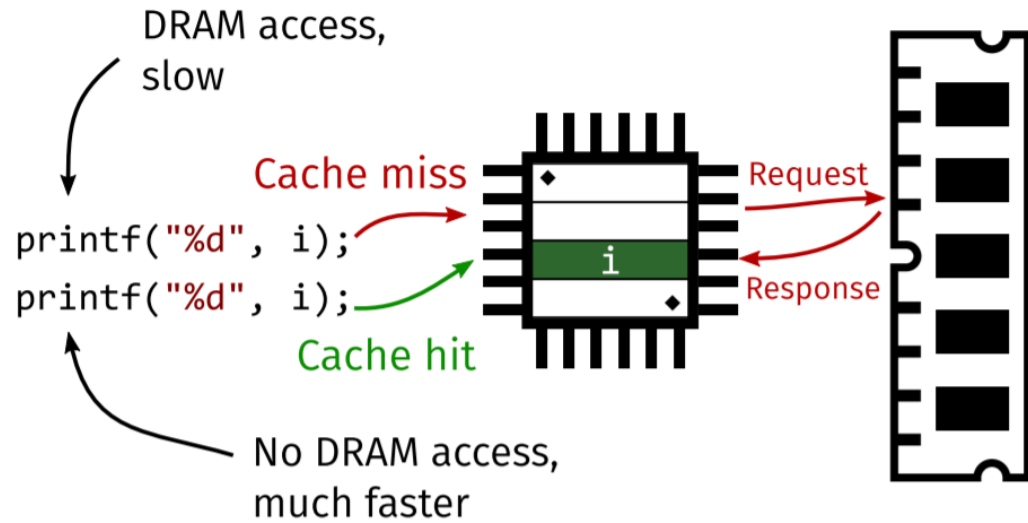


Execution
time

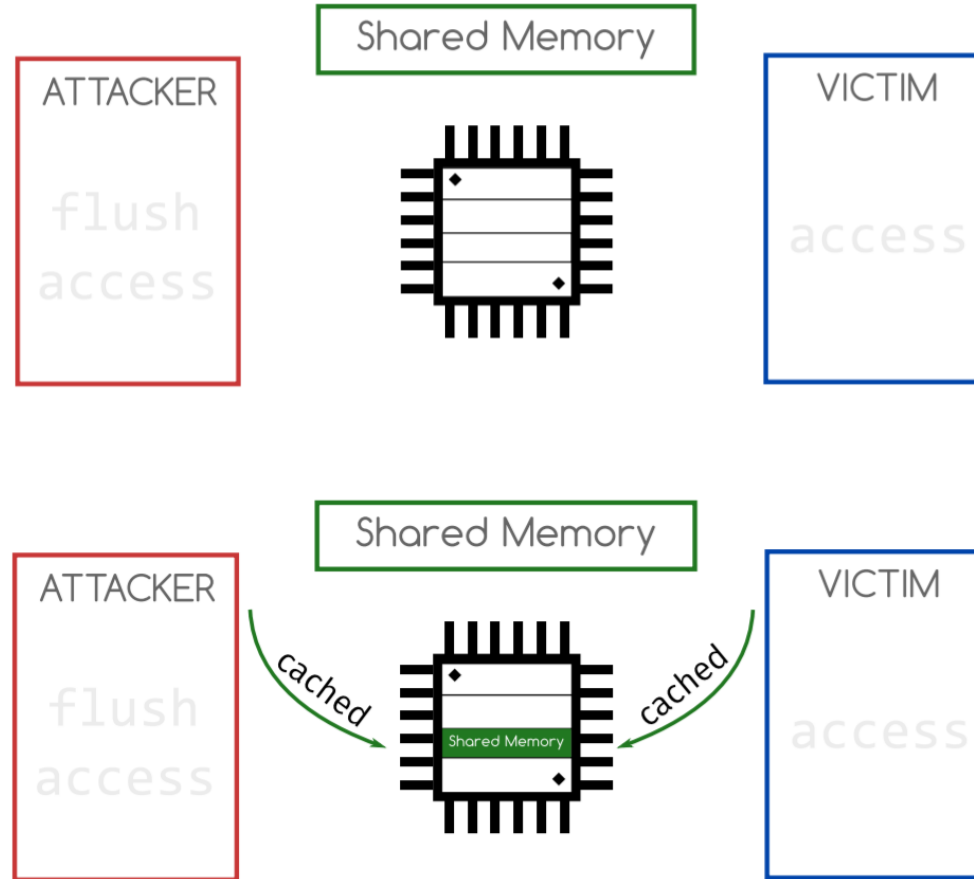


CPU caches

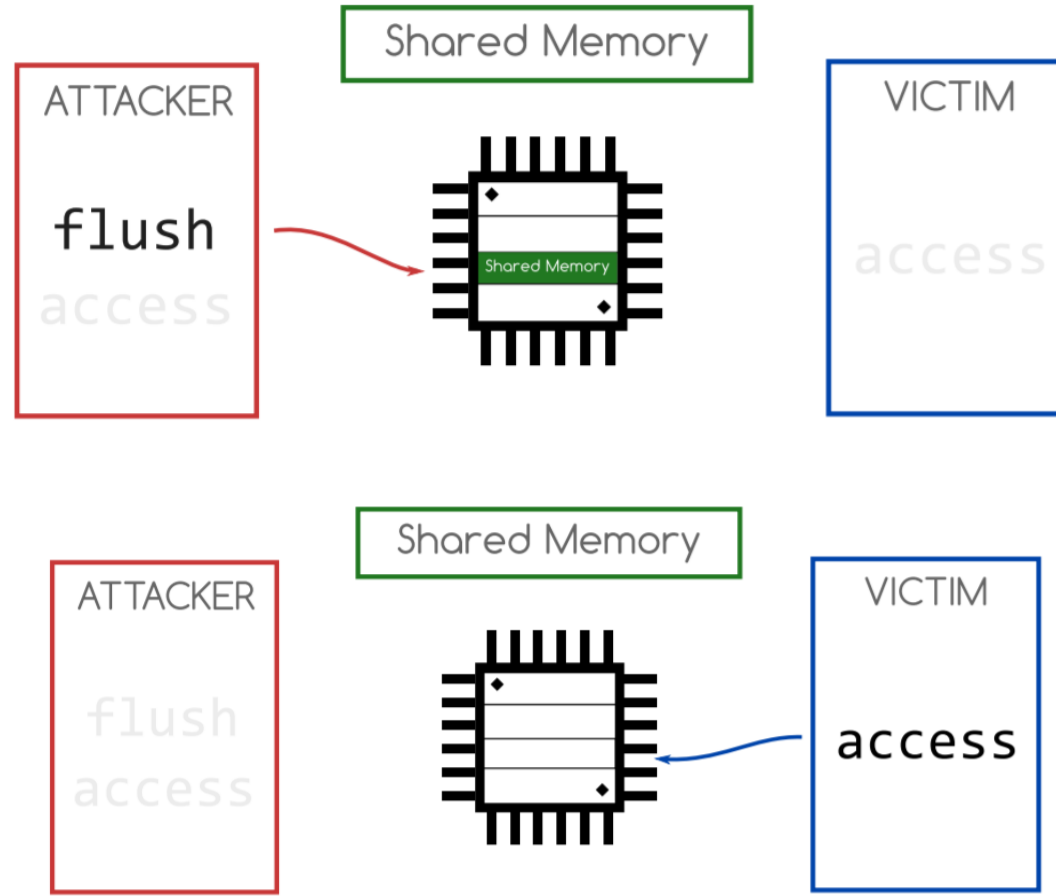
Side-effects of Cache



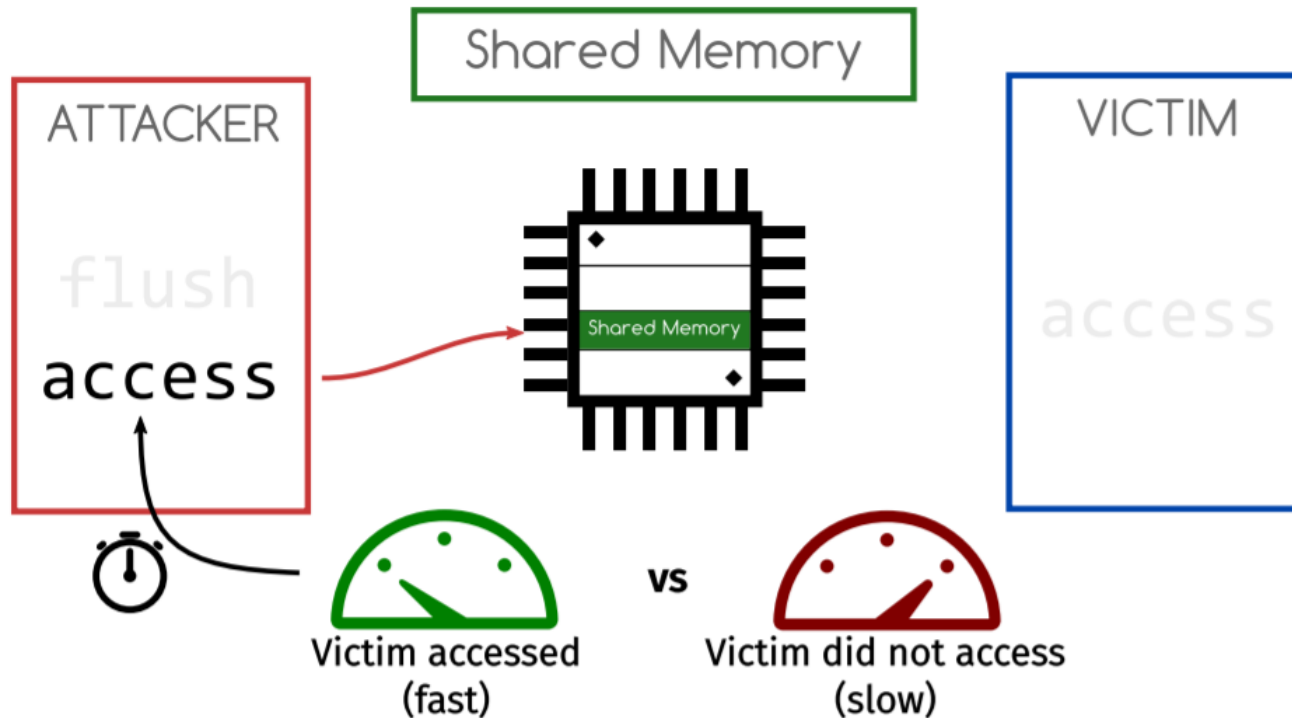
Side-effects of Cache




Side-effects of Cache



Side-effects of Cache



- It's possible to explicitly flush a cache line
 - CLFLUSH
- Same for measure the access latency
 - RDTSC / RDTSCP
-  Non-privileged operations

Code Example

Why 4096 here?

```
void victim(char buf*) {  
    // assume secret = 84  
    load &buf[secret * 4096];  
}
```

```
int attack() {  
    char *buf = malloc(1024 * 4096);  
    for (int i = 0; i < 1024 * 4096; i += 64)  
        clflush(buf + i);
```

```
victim(buf);
```

```
for (int i = 0; i < 1024 * 4096; i += 4096) {  
    int64_t start = rdtsc();  
    load &buf[i];  
    int64_t end = rdtsc();  
    if (end - start < THRESHOLD)  
        return i; // i is the secret  
}
```

- Flush+Reload over all pages of the array



Summary of Hardware Side-channel

- Stateful Attack (e.g. Cache)
 - (the Attacker) restore shared resource to a known state
 - (the Victim) execute and change the state
 - (the Attacker) checks the state of the shared resource again to learn secrets about the victim's execution
- Stateless Attack (not included in this talk, see LOTR paper)
 - passively monitors the latency to access the shared resource and uses variations in this latency to infer secrets about the victim's execution
 - e.g. in LOTR:
 - Ringbus contention stands for "1", and no contention for "0"
 - Author claims: 4Mbps leak speed, possible to leak private key from crypto routine

Meltdown

CVE-2017-5754



Overview

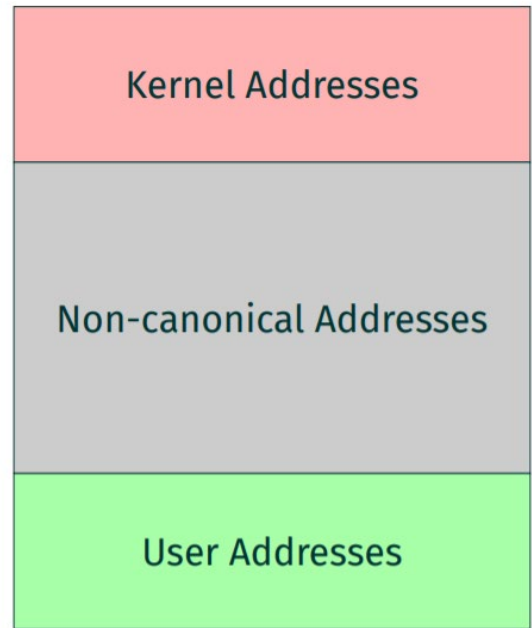
the authors claim...

- Meltdown exploits side effects of out-of-order execution on modern processors.
- Meltdown breaks all security guarantees provided by address space isolation.
- Meltdown can ...
 - **read arbitrary kernel-memory** locations including personal data and passwords.
 - **read memory of other processes or virtual machines** in the cloud without any permissions or privileges.

Outline

- Memory Layout & Isolation of an OS
- Side-channel of Cache
 - Flush + Reload Attack
- Out-of-order Processor & Speculations
- Building the Attack

Memory Isolation of an OS (1)

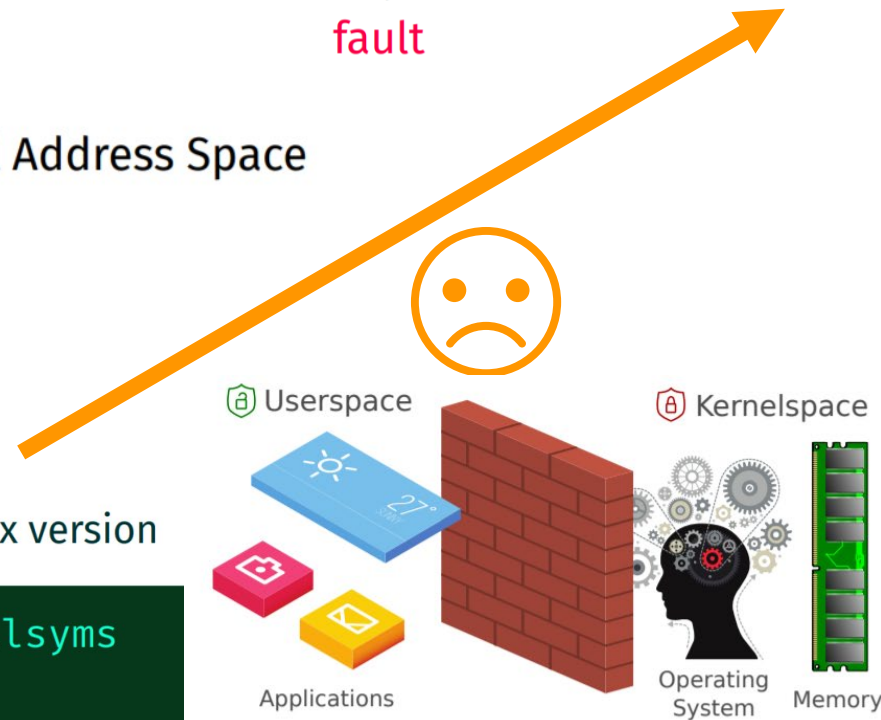


```
char data = *(char*) 0xffffffff81a000e0;  
printf("%c\n", data);
```

- Any invalid access throws an exception → **segmentation fault**

- Find something human readable, e.g., the Linux version

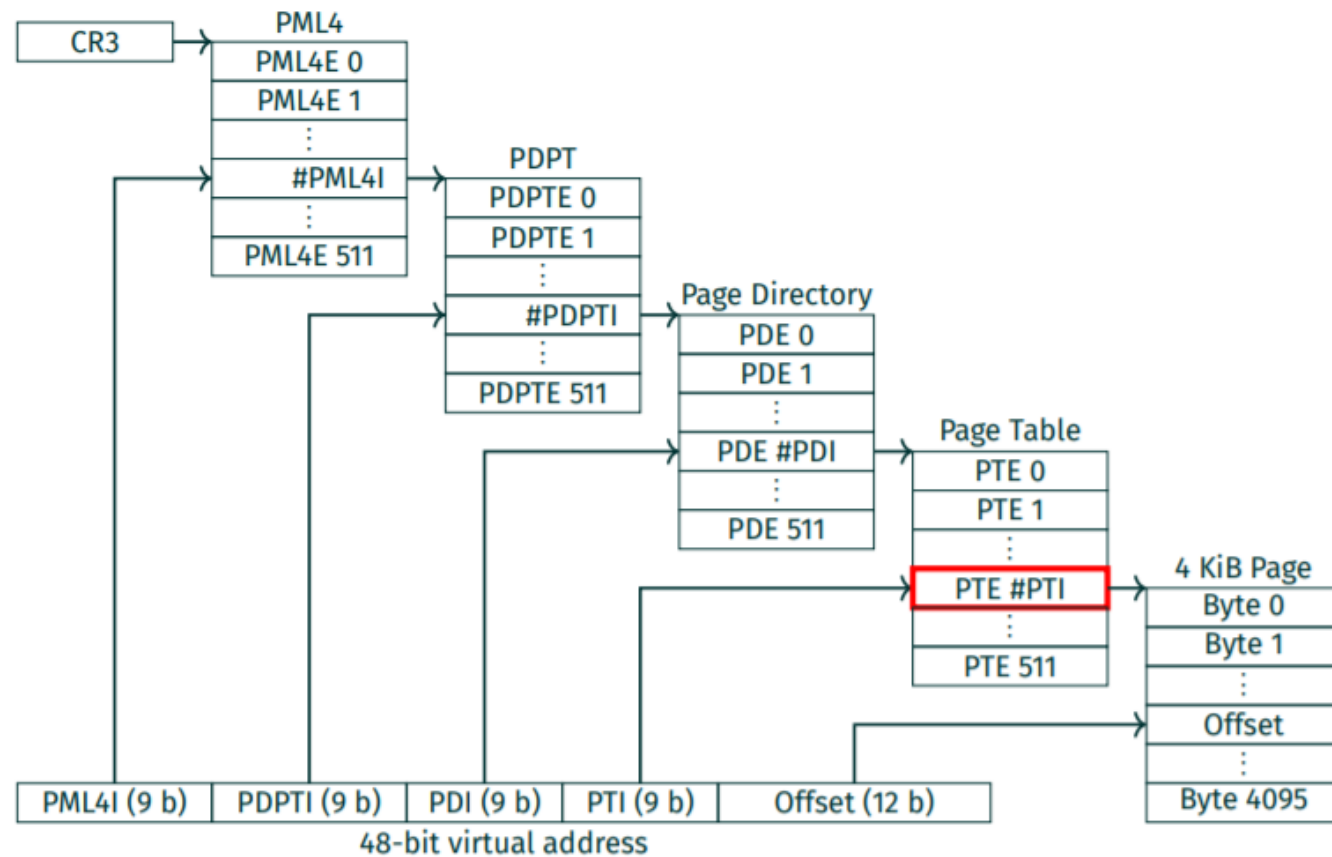
```
# sudo grep linux_banner /proc/kallsyms  
ffffffff81a000e0 R linux_banner
```



- Kernel is isolated from user space
- This **isolation** is a combination of hardware and software
- User applications cannot access anything from the kernel

Memory Isolation of an OS (2)

- Page Table



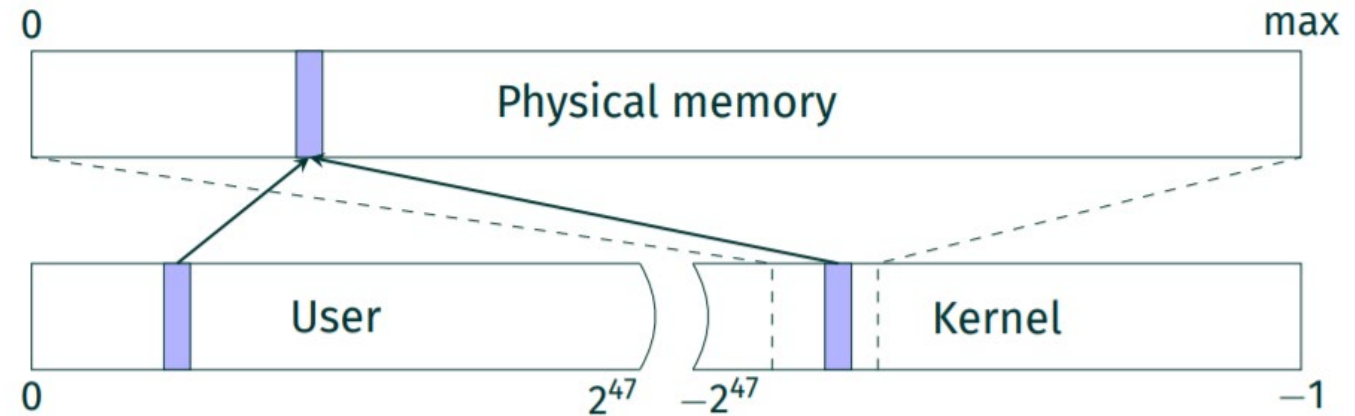
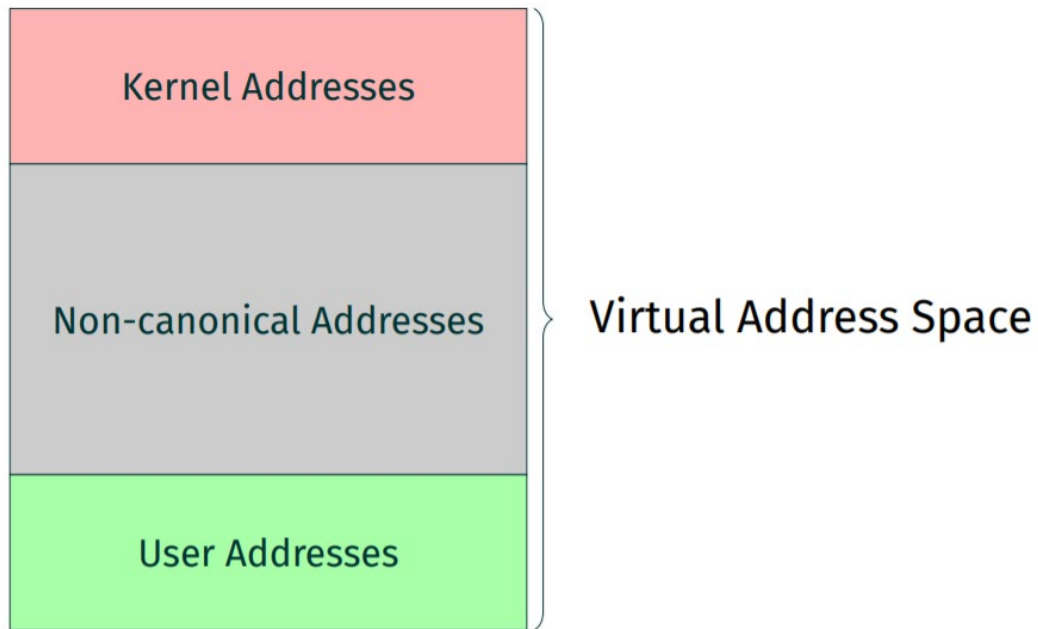
Memory Isolation of an OS (3)

- CPU support **virtual address spaces** to isolate processes
- Physical memory is organized in **page frames**
- Virtual memory pages are **mapped** to page frames **using page tables**

P	RW	US	WT	UC	R	D	S	G	Ignored	
Physical Page Number										
Ignored										X

User/Supervisor bit defines in which **privilege level** the page can be accessed

Memory Layout of an OS



- Kernel is typically **mapped** into every address space

- In the past...
- For performance reason, memory reserved for kernel is mapped into each processes' virtual space.

Answer: Why this won't work? (ideally)

```
char data = *(char*) 0xffffffff81a000e0;  
printf("%c\n", data);
```

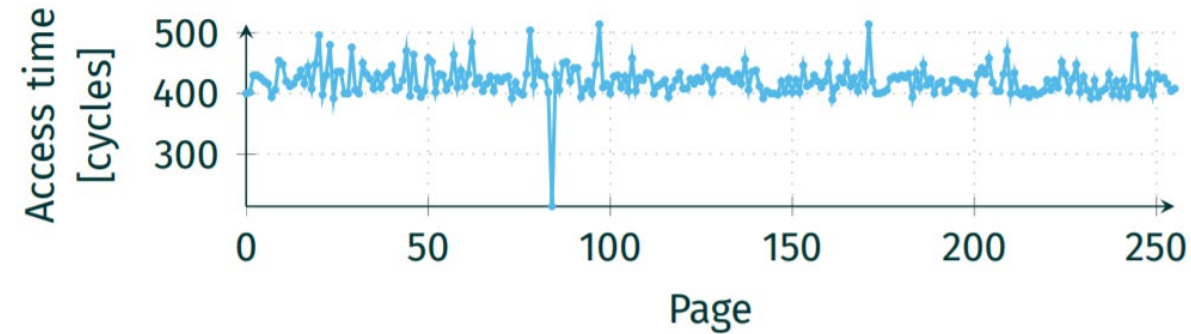
- We try to load an **inaccessible address**
- Permission is **checked**

Building the attack

Toy Example

```
*(volatile char*) 0; // raise_exception();  
array[84 * 4096] = 0;
```

- Flush+Reload over all pages of the array

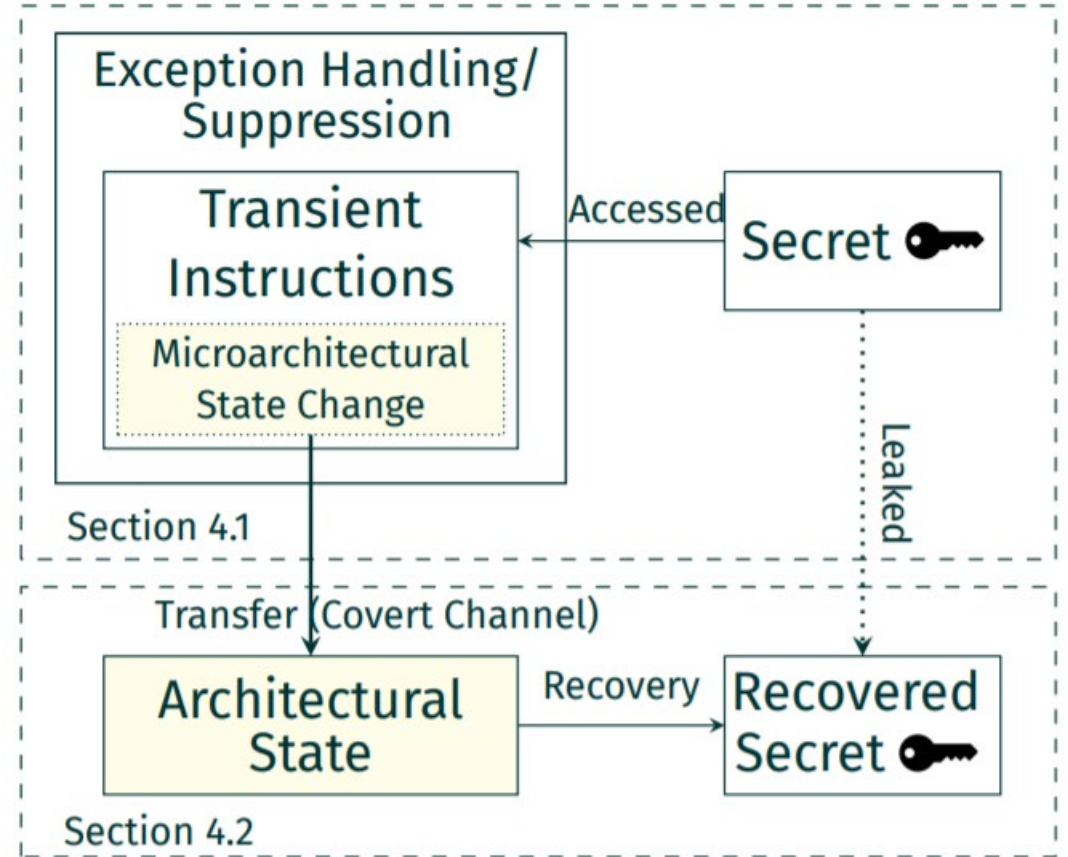


- “Unreachable” code line was **actually executed**
- Exception was only thrown **afterwards**

Building the Attack

The blueprint

- Out-of-order instructions **leave microarchitectural traces**
 - We can see them for example in the cache
- Give such instructions a name: **transient instructions**
- We can indirectly observe the **execution of transient instructions**



Building the Attack

Crash handling

- Transient instructions are executed all the time
- Loading inaccessible addresses leads to a crash (segfault)
- How to **prevent the crash?**



To sum up, Meltdown Attack is ...

```
char data = *(char*) 0xffffffff81a000e0;  
array[data * 4096] = 0;
```

1. Flush the **array**
2. User program read a kernel-side virtual address to a register **secret_to_be_leaked**
3. Before exception being actually handled by hardware, use a follow-up memory access to **array** to memorize the **secret_to_be_leaked**
4. Exception handled by hardware
 - Prevent crash by:
 - POSIX signal handler (SIGSEGV)
 - or Intel TSX (another advanced hardware feature)
5. Use Flush + Reload to recover **secret_to_be_leaked**



- **Index** of cache hit reveals **data**
- **Permission check** is in some cases **not fast enough**



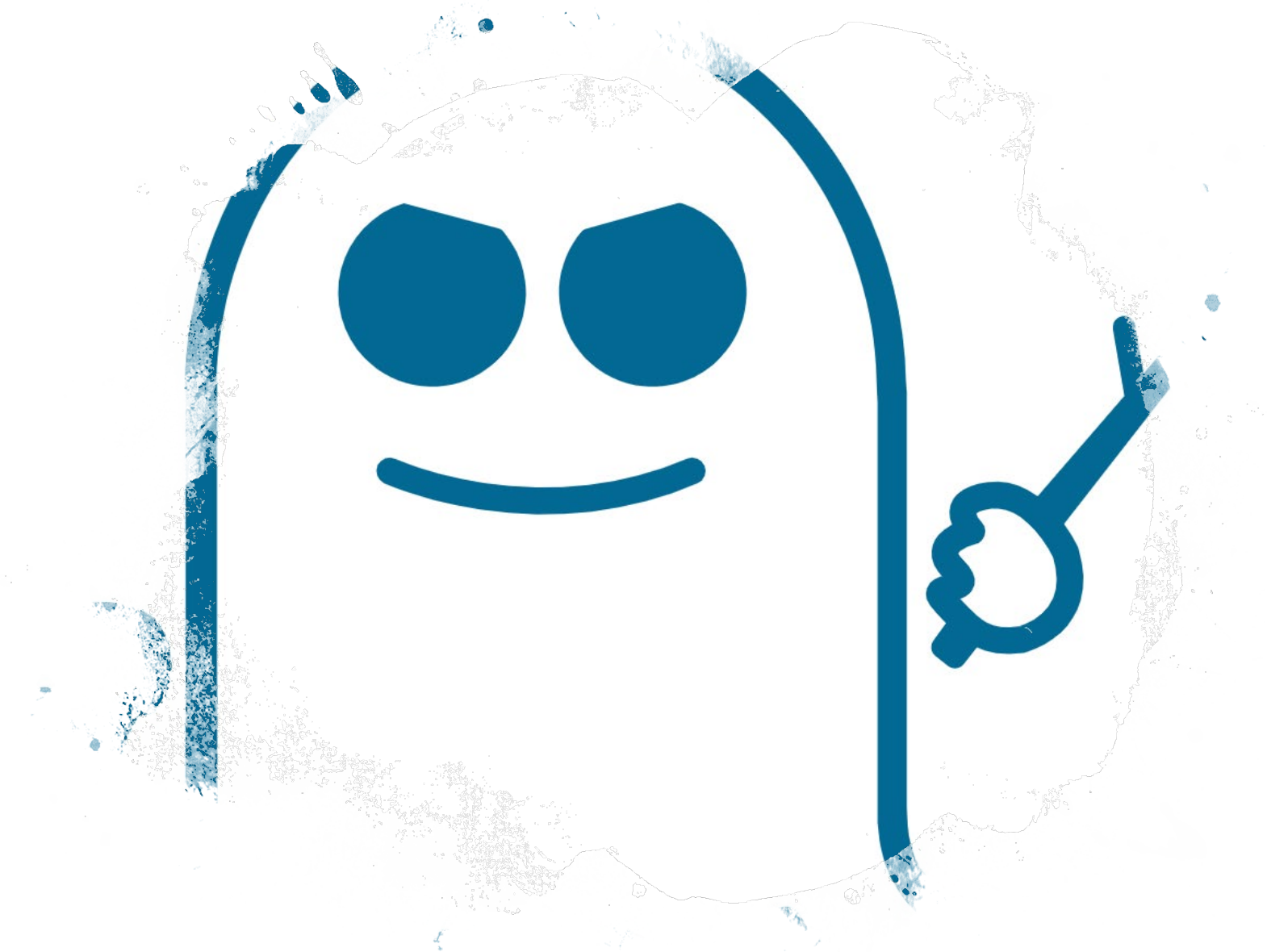
Q&A



Spectre

CVE-2017-5753

CVE-2017-5715



To Recap: Skylake Microarchitecture (from mdsattacks.com)

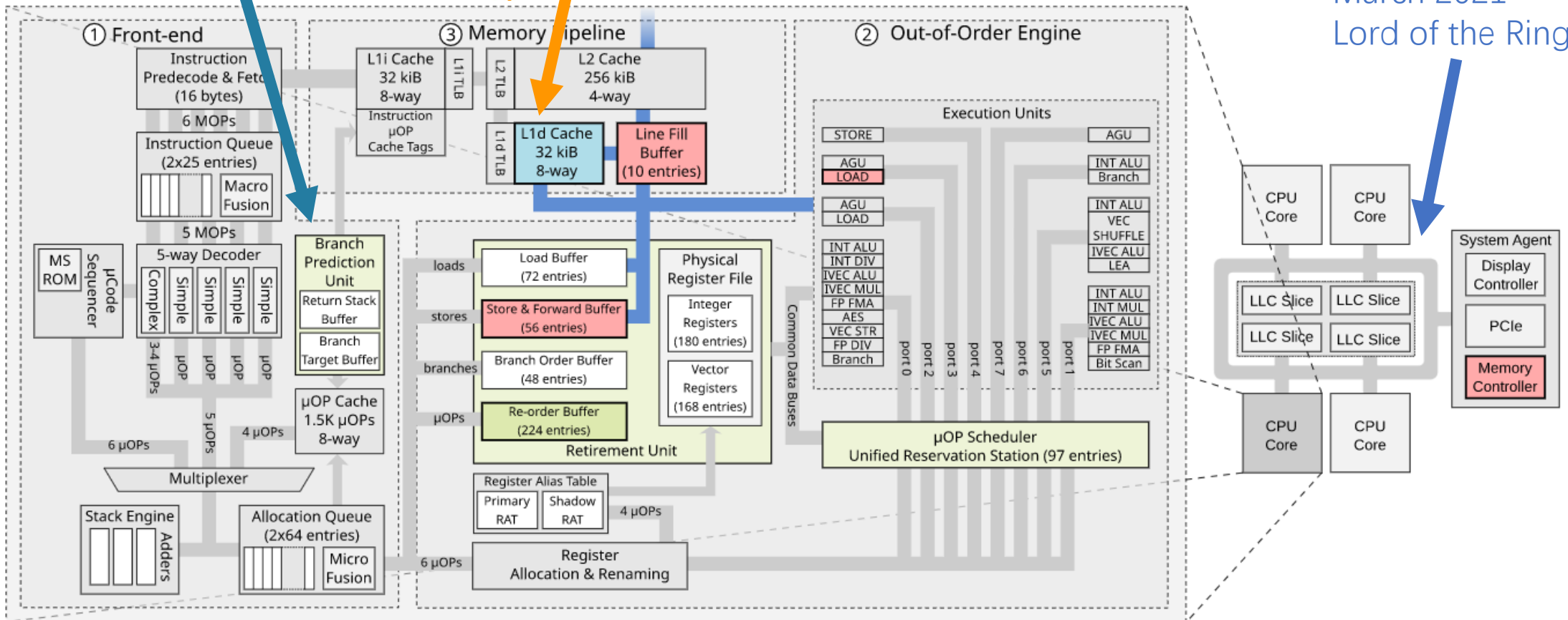


Spectre



Meltdown

March 2021
Lord of the Ring(s)



Variants

- ※ Variant 1: Bounds Checking Bypass
- Variant 2: Branch Target Injection

(Open with right-click)



Spectre Attacks: Exploiting Speculative Execution

IEEE Security & Privacy (May 20, 2019)

Paul Kocher¹, Jann Horn², Anders Fogh³, Daniel Genkin⁴, Daniel Gruss⁵,
Werner Haas⁶, Mike Hamburg⁷, Mortiz Lipp⁵, Stefan Mangard⁵,
Thomas Prescher⁶, Michael Schwartz⁵, Yuval Yarom⁸

¹ Independent, ² Google Project Zero, ³ G DATA Advanced Analytics, ⁴ University of Pennsylvania and University of Maryland,

⁵ Graz University of Technology, ⁶ Cyberus Technology, ⁷ Rambus, Cryptography Research Division, ⁸ University of Adelaide & Data61

All trademarks are the property of their respective owners. This presentation is provided without any guarantee or warranty whatsoever.