

Side Channels and Transient Execution Attacks

Daniel Gruss

December 5, 2019

Graz University of Technology





90
0

0

15



side channel
= obtaining meta-data and
deriving secrets from it

CHANGE MY MIND



- Profiling cache utilization with performance counters?



- Profiling cache utilization with performance counters? → No





- Profiling cache utilization with performance counters? → No
- Observing cache utilization with performance counters and using it to infer a crypto key?



- Profiling cache utilization with performance counters? → No
- Observing cache utilization with performance counters and using it to infer a crypto key? → Yes



- Profiling cache utilization with performance counters? → No
- Observing cache utilization with performance counters and using it to infer a crypto key? → Yes
- Measuring memory access latency with Flush+Reload?



- Profiling cache utilization with performance counters? → No
- Observing cache utilization with performance counters and using it to infer a crypto key? → Yes
- Measuring memory access latency with Flush+Reload? → No



- Profiling cache utilization with performance counters? → No
- Observing cache utilization with performance counters and using it to infer a crypto key? → Yes
- Measuring memory access latency with Flush+Reload? → No
- Measuring memory access latency with Flush+Reload and using it to infer keystroke timings?



- Profiling cache utilization with performance counters? → No
- Observing cache utilization with performance counters and using it to infer a crypto key? → Yes
- Measuring memory access latency with Flush+Reload? → No
- Measuring memory access latency with Flush+Reload and using it to infer keystroke timings? → Yes

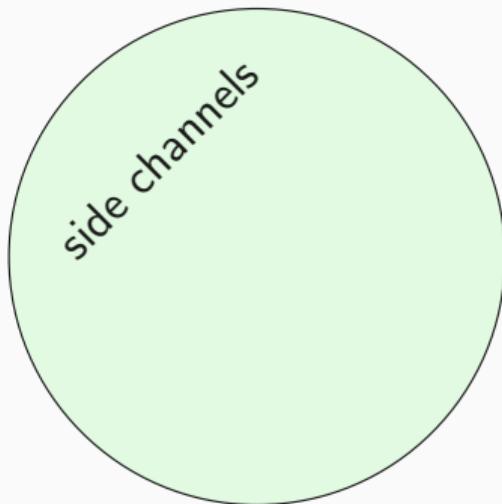
Intel Analysis of Speculative Execution Side Channels

[Download PDF](#)

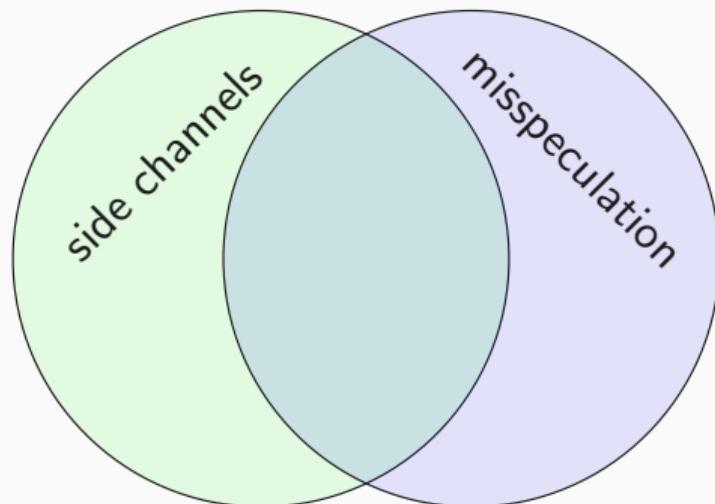


Intel Analysis of Speculative Execution Side Channels

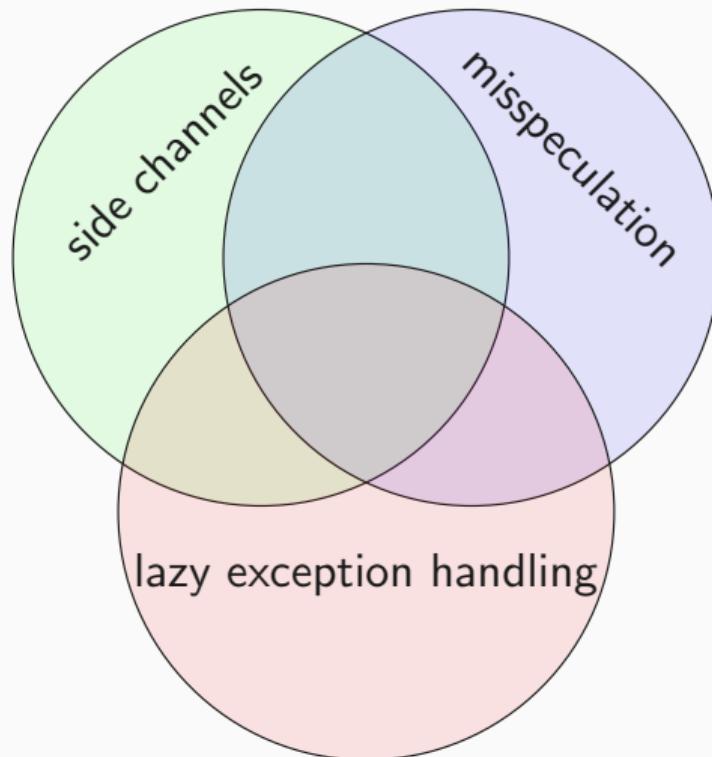
[White Paper](#)



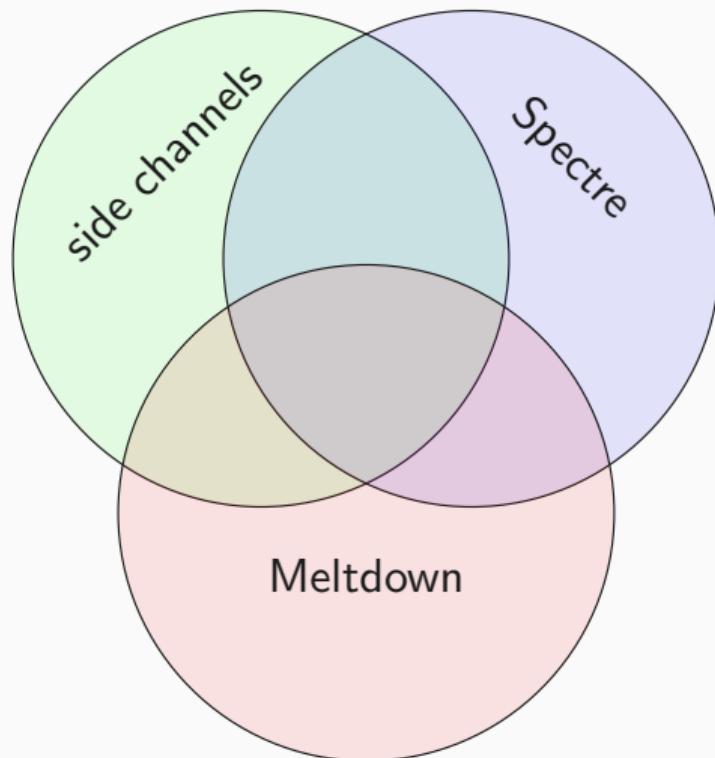
- traditional cache attacks (crypto, keys, etc)



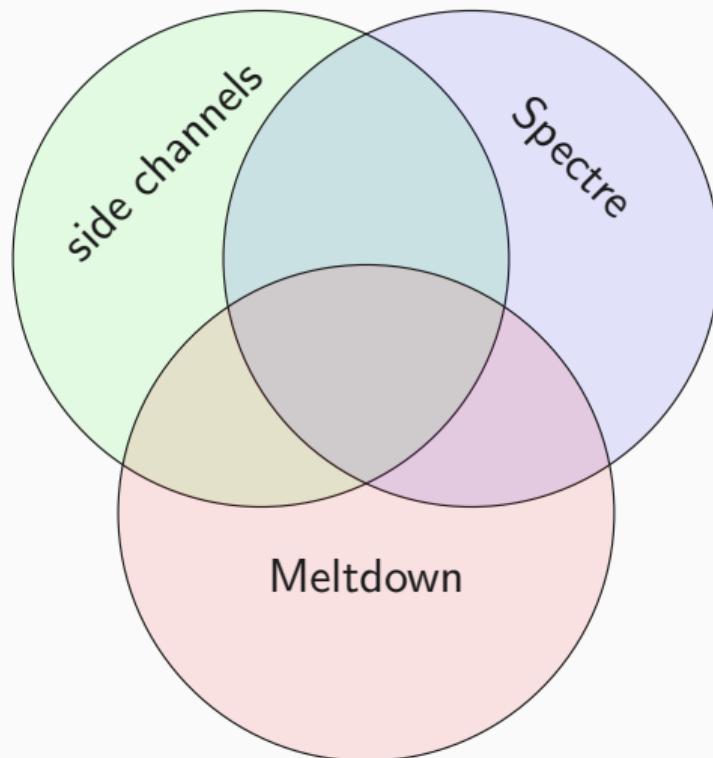
- traditional cache attacks (crypto, keys, etc)
- actual misspeculation (e.g., branch misprediction)



- traditional cache attacks (crypto, keys, etc)
- actual misspeculation (e.g., branch misprediction)
- Meltdown, Foreshadow, ZombieLoad, etc



- traditional cache attacks (crypto, keys, etc)
- actual misspeculation (e.g., branch misprediction)
- Meltdown, Foreshadow, ZombieLoad, etc



- traditional cache attacks (crypto, keys, etc)
- actual misspeculation (e.g., branch misprediction)
- Meltdown, Foreshadow, ZombieLoad, etc
- **Let's avoid the term Speculative Side-Channel Attacks**









1337 4242

FOOD CACHE

Revolutionary concept!

Store your food at home,
never go to the grocery store
during cooking.

Can store **ALL** kinds of food.

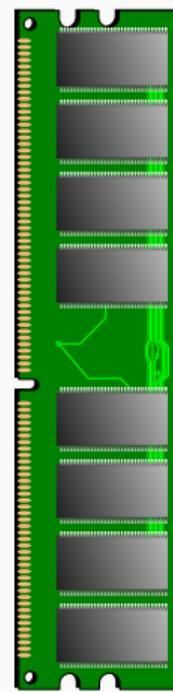
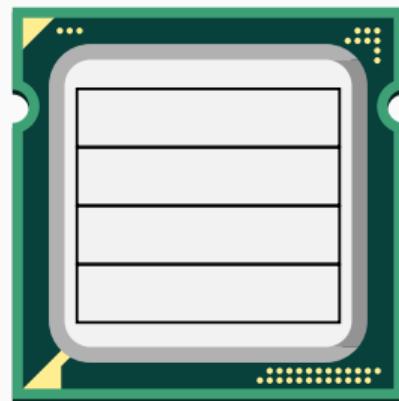
ONLY TODAY INSTEAD OF ~~\$1,300~~

\$1,299

ORDER VIA PHONE: +555 12345

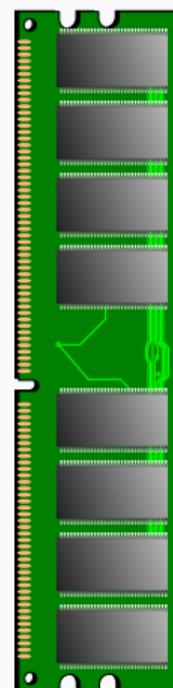
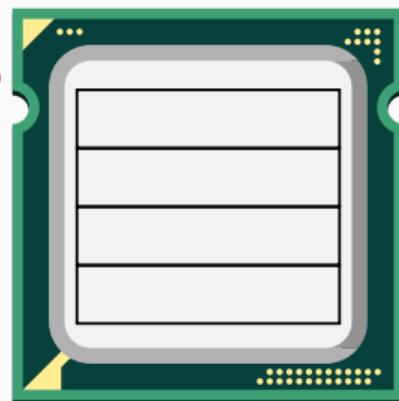


```
printf("%d", i);  
printf("%d", i);
```



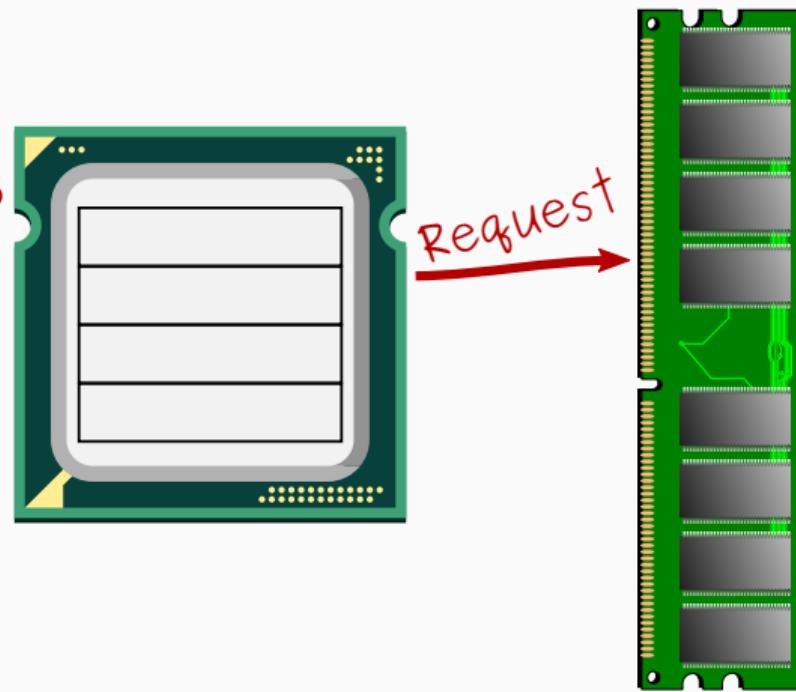
```
printf("%d", i);  
printf("%d", i);
```

Cache miss

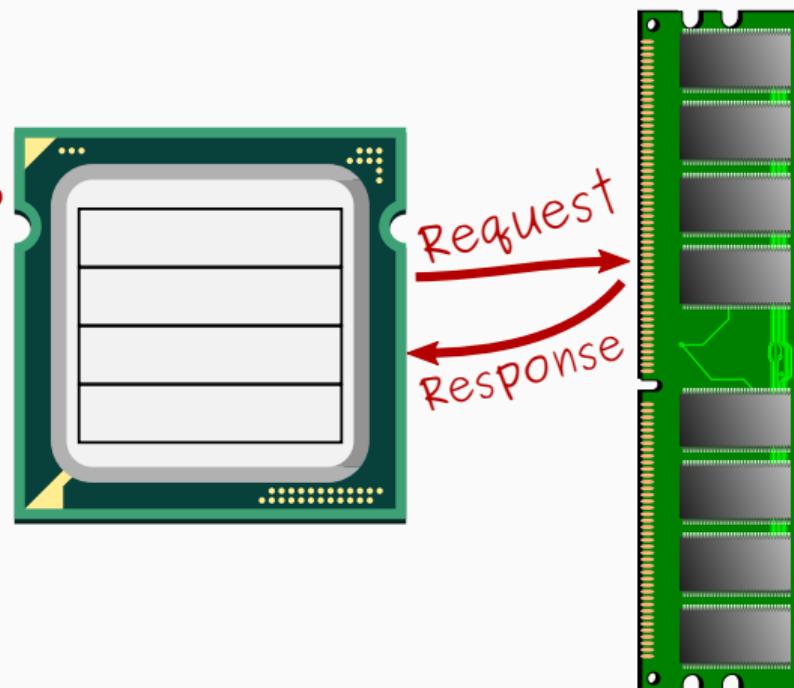


```
printf("%d", i);  
printf("%d", i);
```

Cache miss

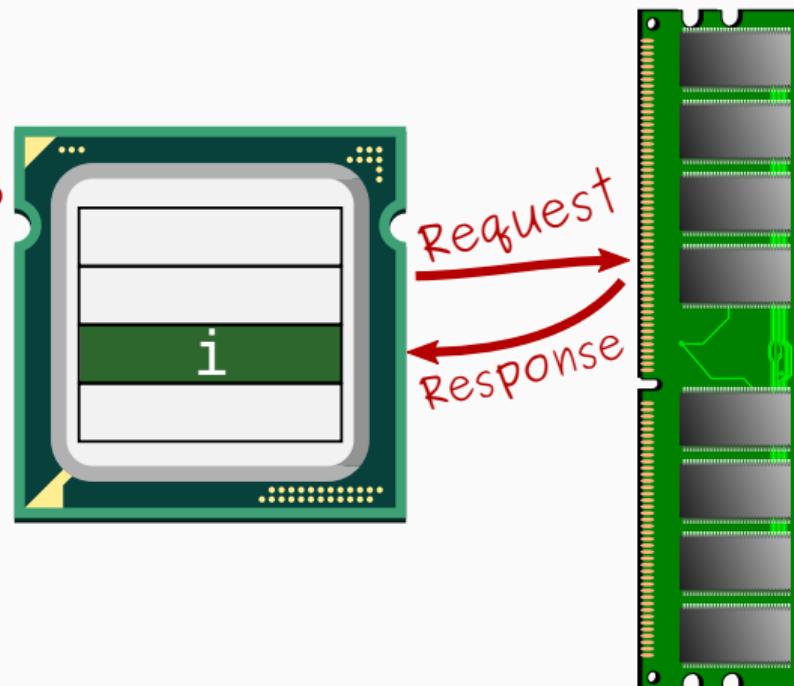


```
Cache miss  
printf("%d", i);  
printf("%d", i);
```



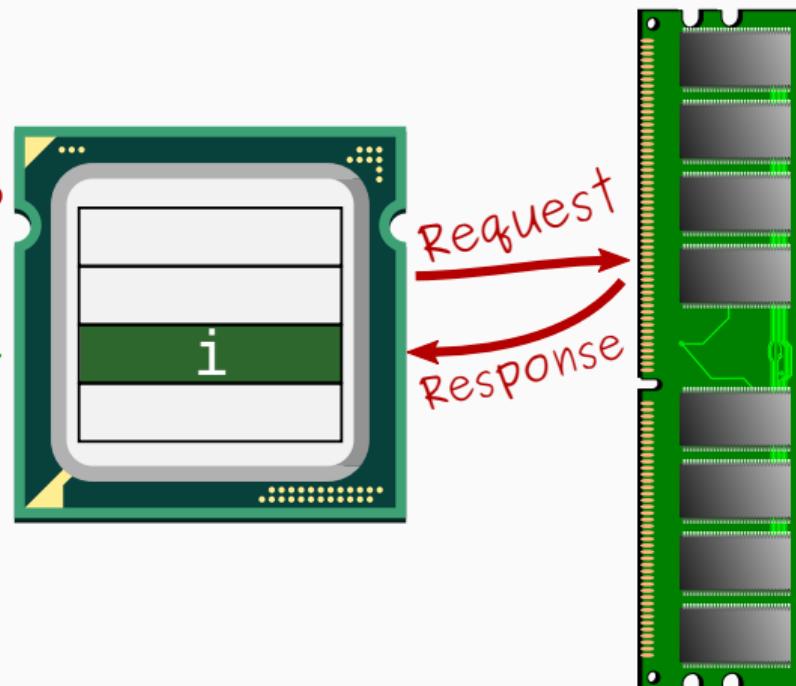
```
printf("%d", i);  
printf("%d", i);
```

Cache miss



```
printf("%d", i);  
printf("%d", i);
```

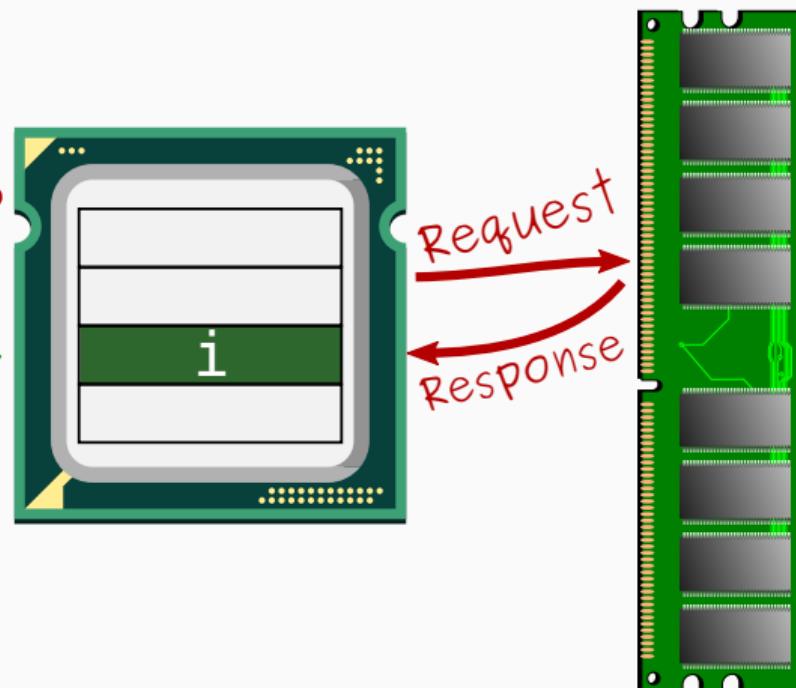
Cache miss
Cache hit

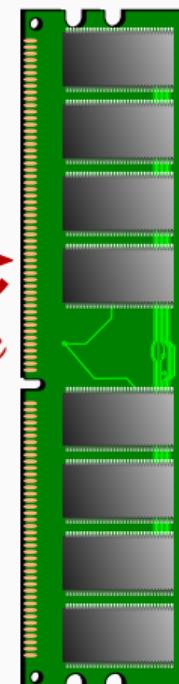
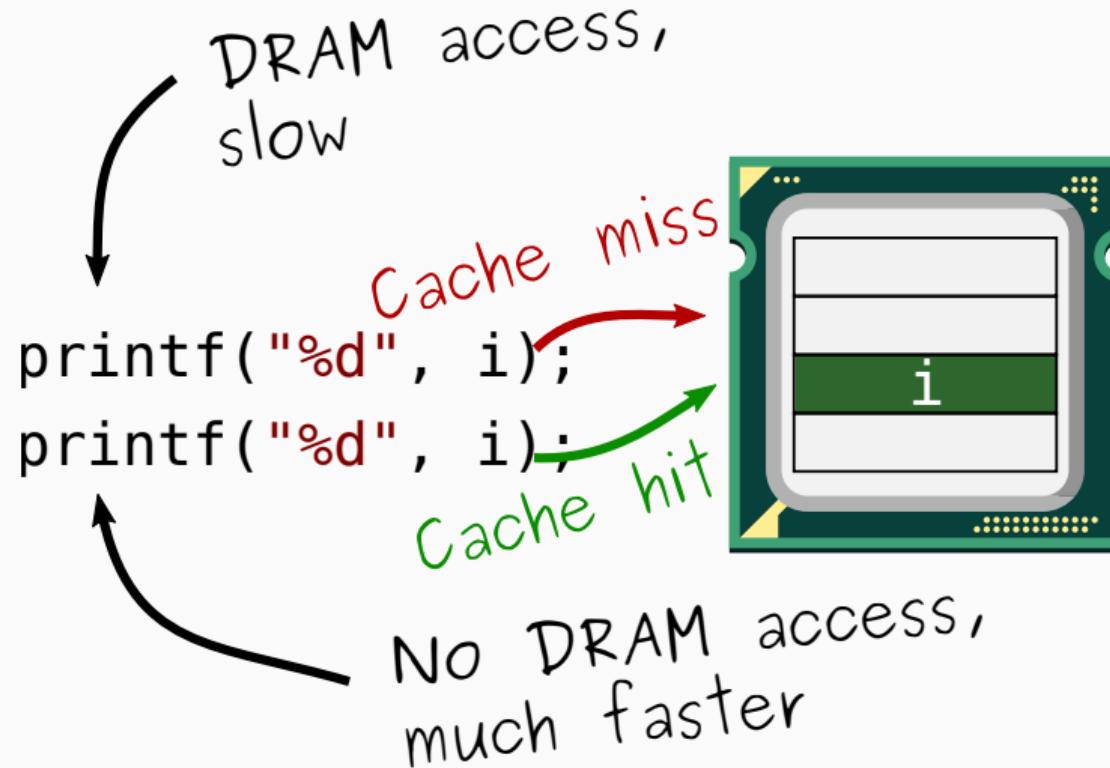


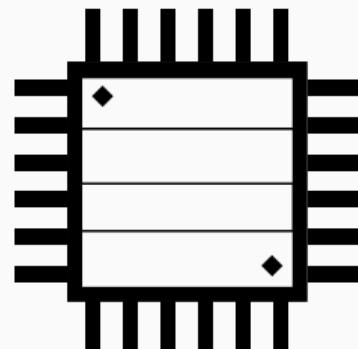
DRAM access,
slow

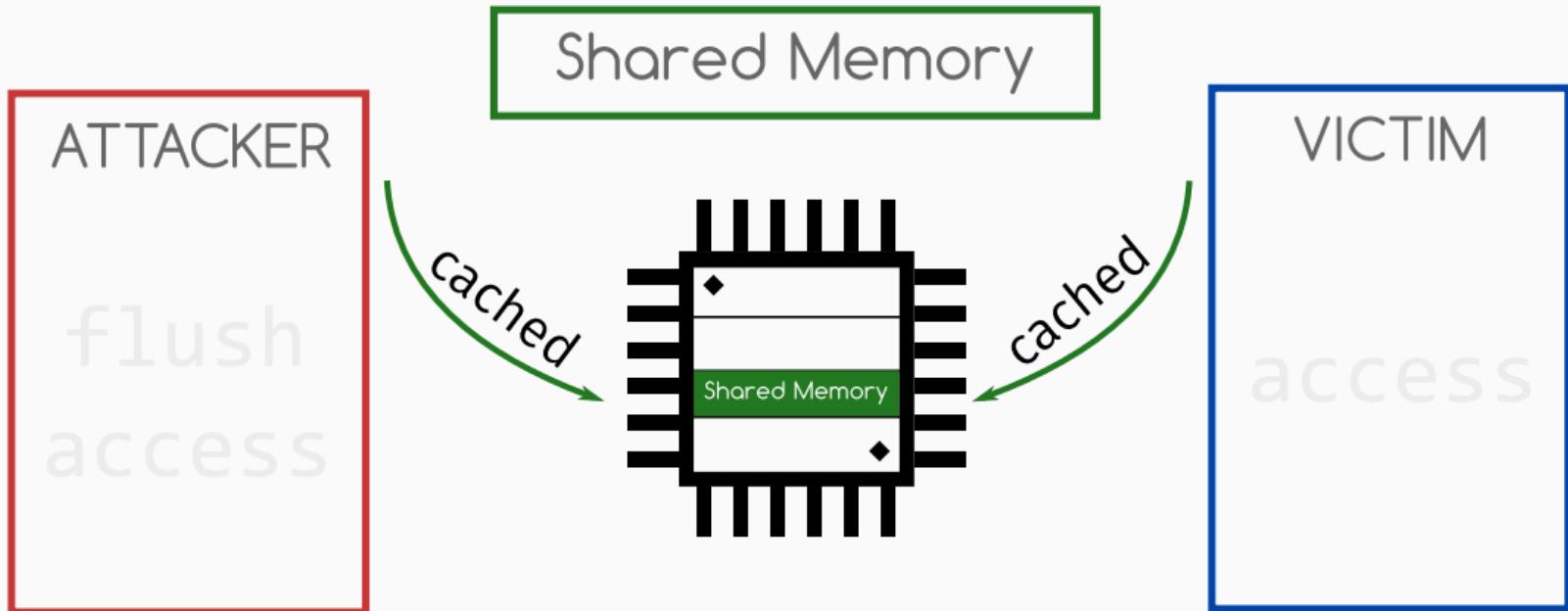
printf("%d", i);
printf("%d", i);

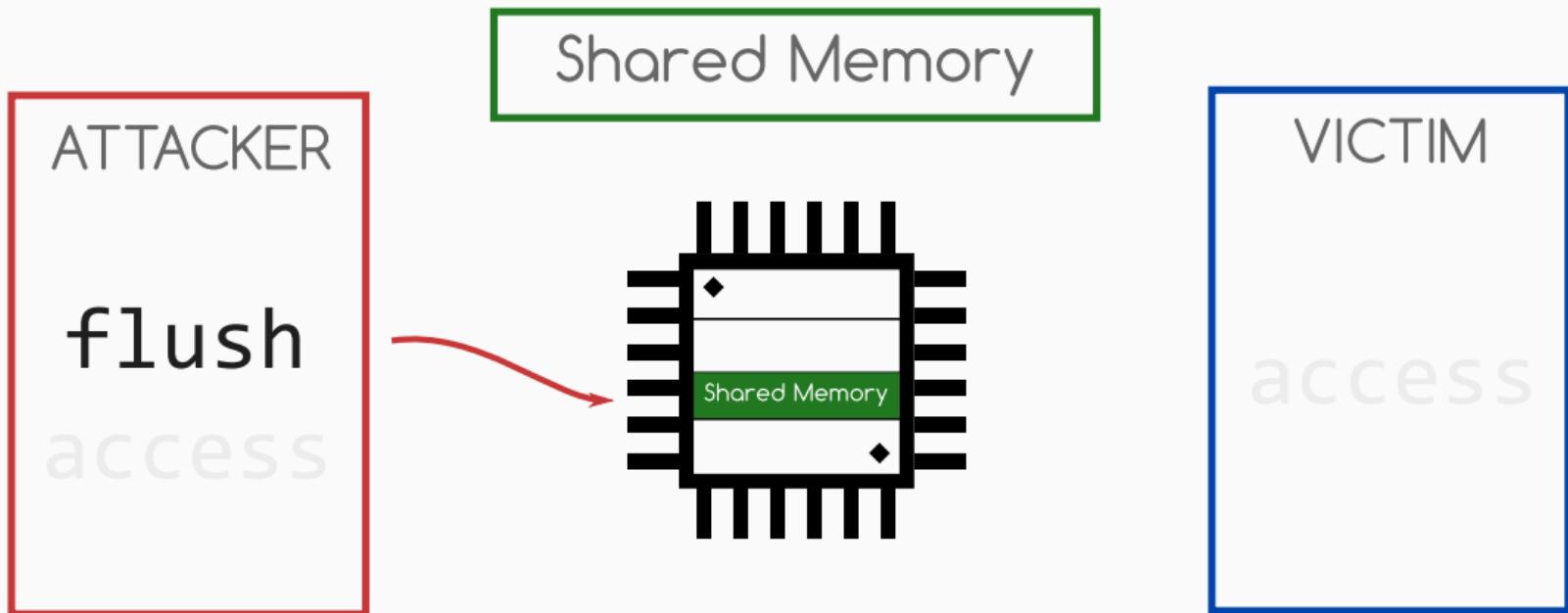
Cache miss
Cache hit

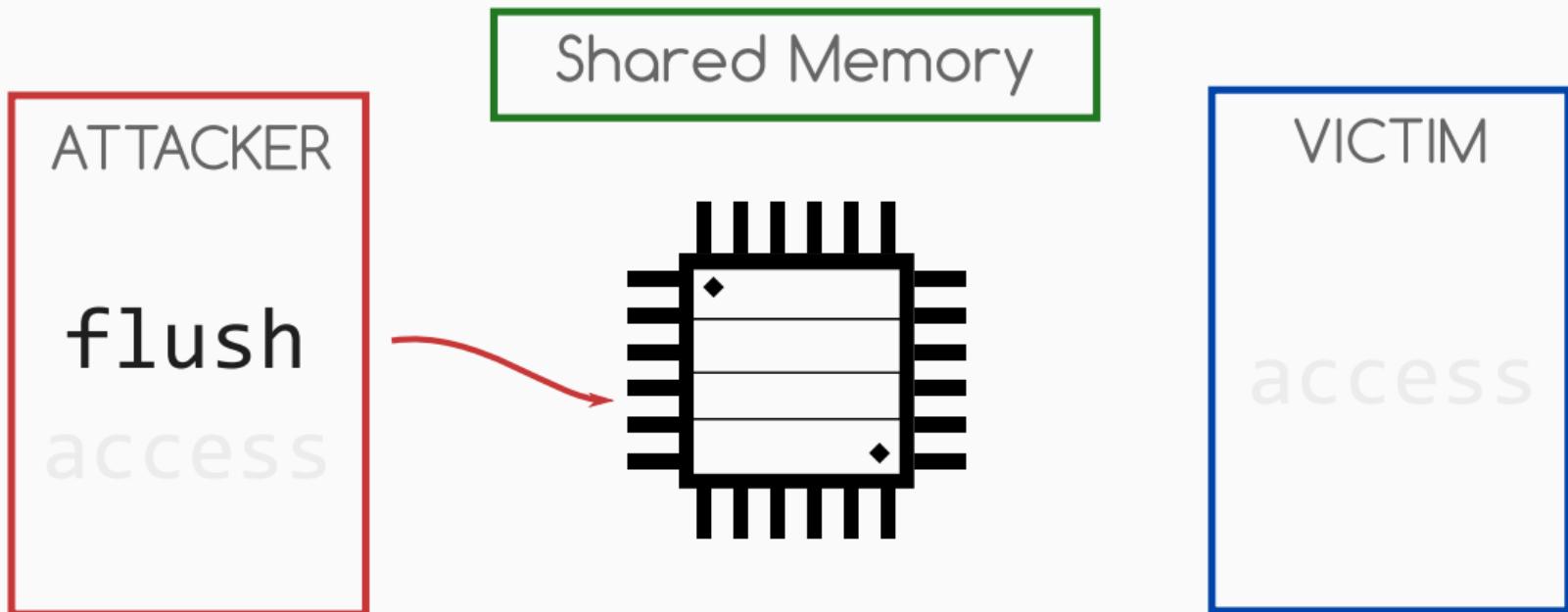


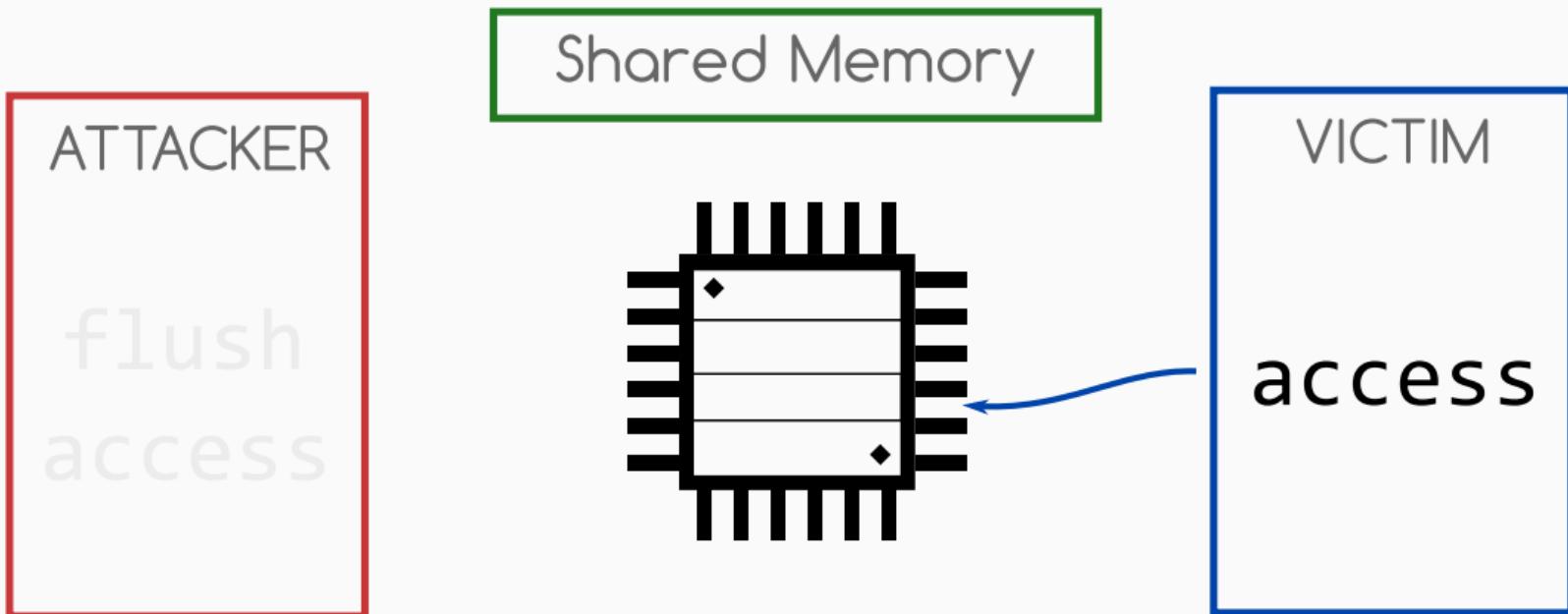


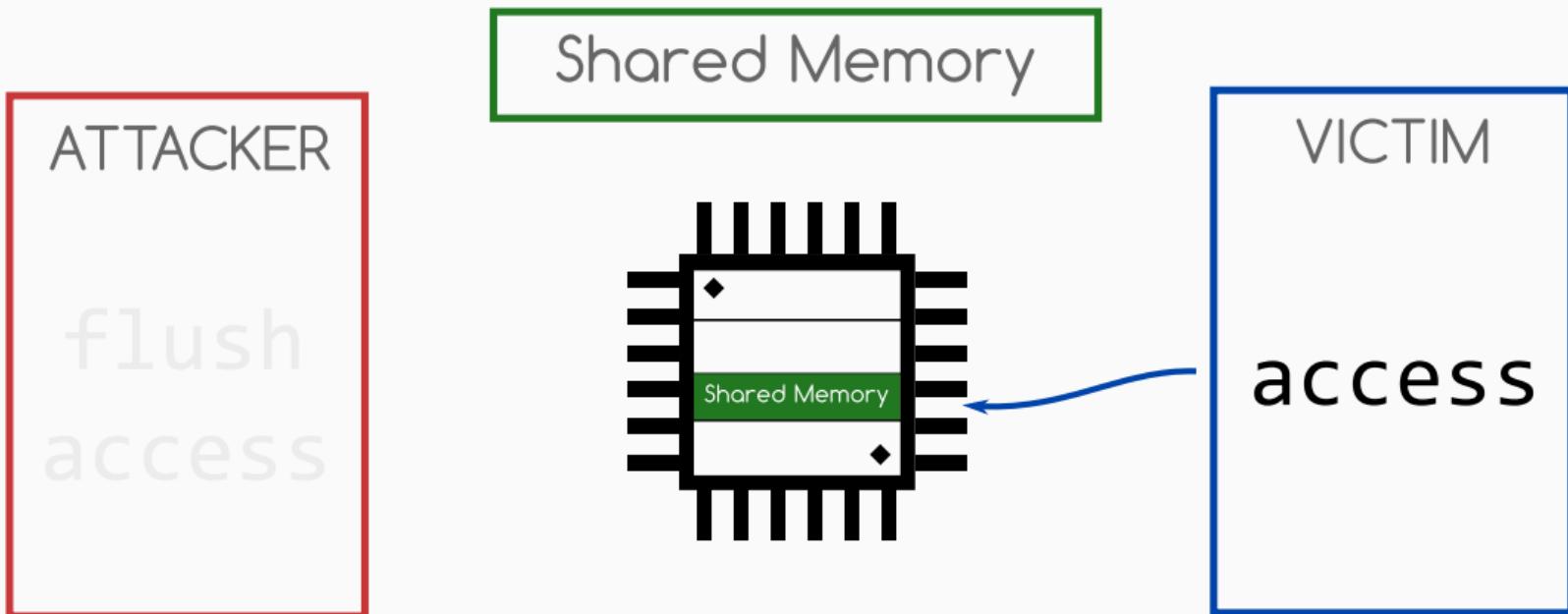


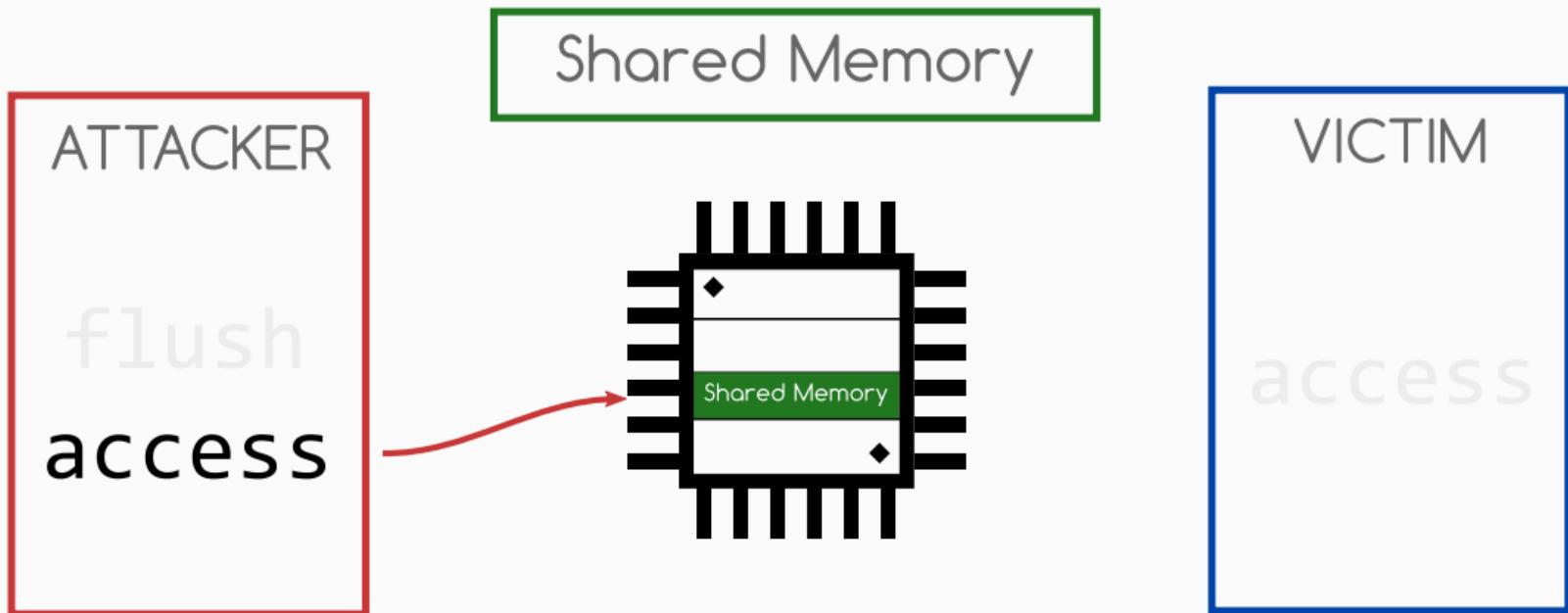


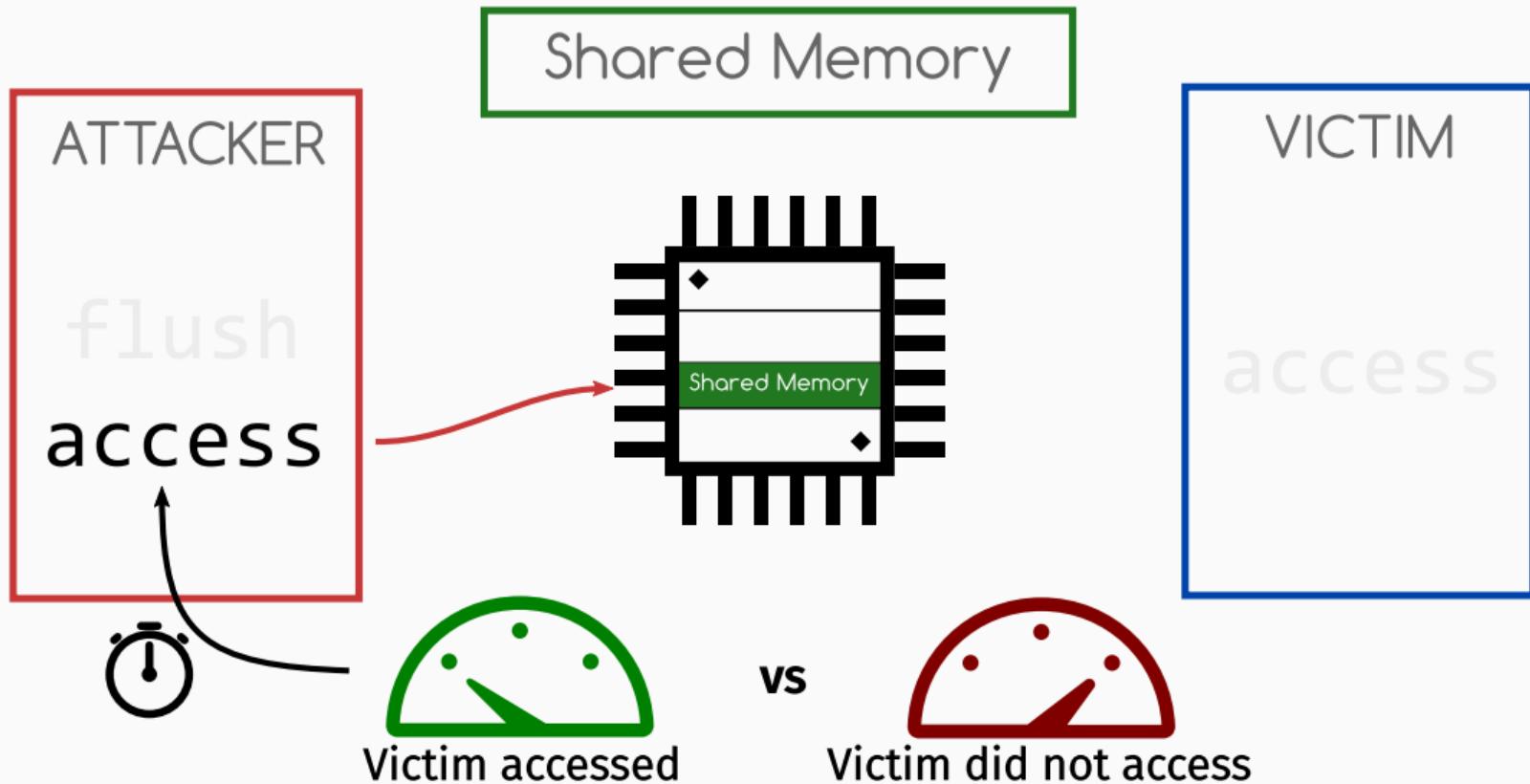












- use pseudo-serializing instruction `rdtscp` (recent CPUs)

- use pseudo-serializing instruction `rdtscp` (recent CPUs)
- and/or use serializing instructions like `cpuid`

- use pseudo-serializing instruction `rdtscp` (recent CPUs)
- and/or use serializing instructions like `cpuid`
- and/or use fences like `mfence`

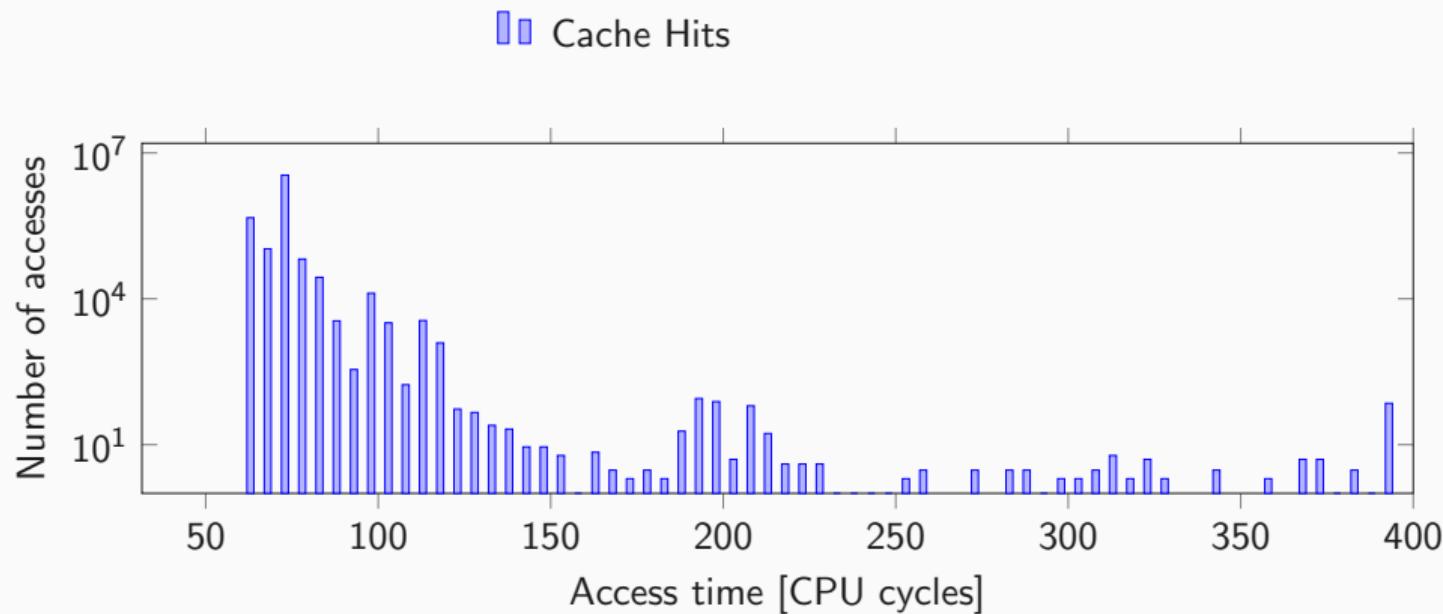
- use pseudo-serializing instruction `rdtscp` (recent CPUs)
- and/or use serializing instructions like `cpuid`
- and/or use fences like `mfence`

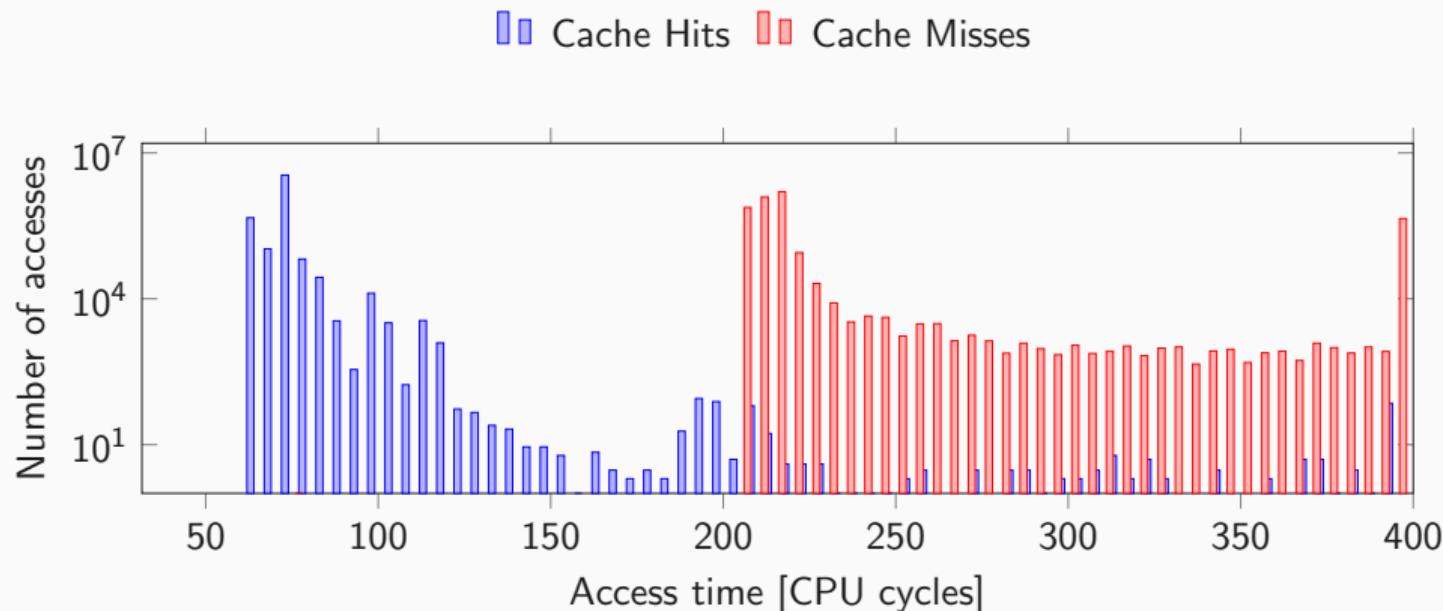
Intel, *How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures White Paper*, December 2010.

AUGUST 22, 2018 BY BRUCE

Intel Publishes Microcode Security Patches, No Benchmarking Or Comparison Allowed!

UPDATE: **Intel has resolved their microcode licensing issue which I complained about in this blog post.** The new license text is [here](#).







- We can build our **own timer** [Lip+16; Sch+17]



- We can build our **own timer** [Lip+16; Sch+17]
- Start a thread that continuously increments a global variable



- We can build our **own timer** [Lip+16; Sch+17]
- Start a thread that continuously increments a global variable
- The global variable is our **timestamp**





**ARE YOU REALLY EXPECTING TO
OUTPERFORM THE HARDWARE COUNTER?**

CPU cycles one increment takes

```
rdtsc [REDACTED] 3           1 timestamp = rdtsc();
```

CPU cycles one increment takes

rdtsc [REDACTED] 3

C

```
1 while(1) {  
2     timestamp++;  
3 }
```

CPU cycles one increment takes

rdtsc  3

C  4.7

```
1 while(1) {  
2     timestamp++;  
3 }
```

CPU cycles one increment takes

rdtsc  3

C  4.7

```
1 while(1) {  
2     timestamp++;  
3 }
```

CPU cycles one increment takes

rdtsc  3

C  4.7

```
1 mov &timestamp, %rcx
2 1: incl (%rcx)
3 jmp 1b
```

Assembly

CPU cycles one increment takes

rdtsc  3

```
1 mov &timestamp, %rcx  
2 1: incl (%rcx)  
3 jmp 1b
```

C  4.7

Assembly  4.67

CPU cycles one increment takes

rdtsc  3

```
1 mov &timestamp, %rcx  
2 1: incl (%rcx)  
3 jmp 1b
```

C  4.7

Assembly  4.67

CPU cycles one increment takes

rdtsc  3

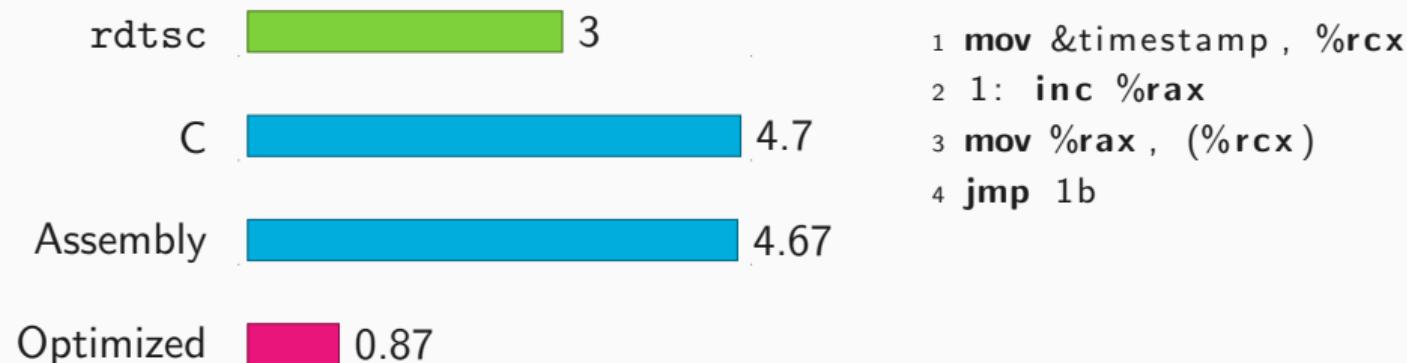
C  4.7

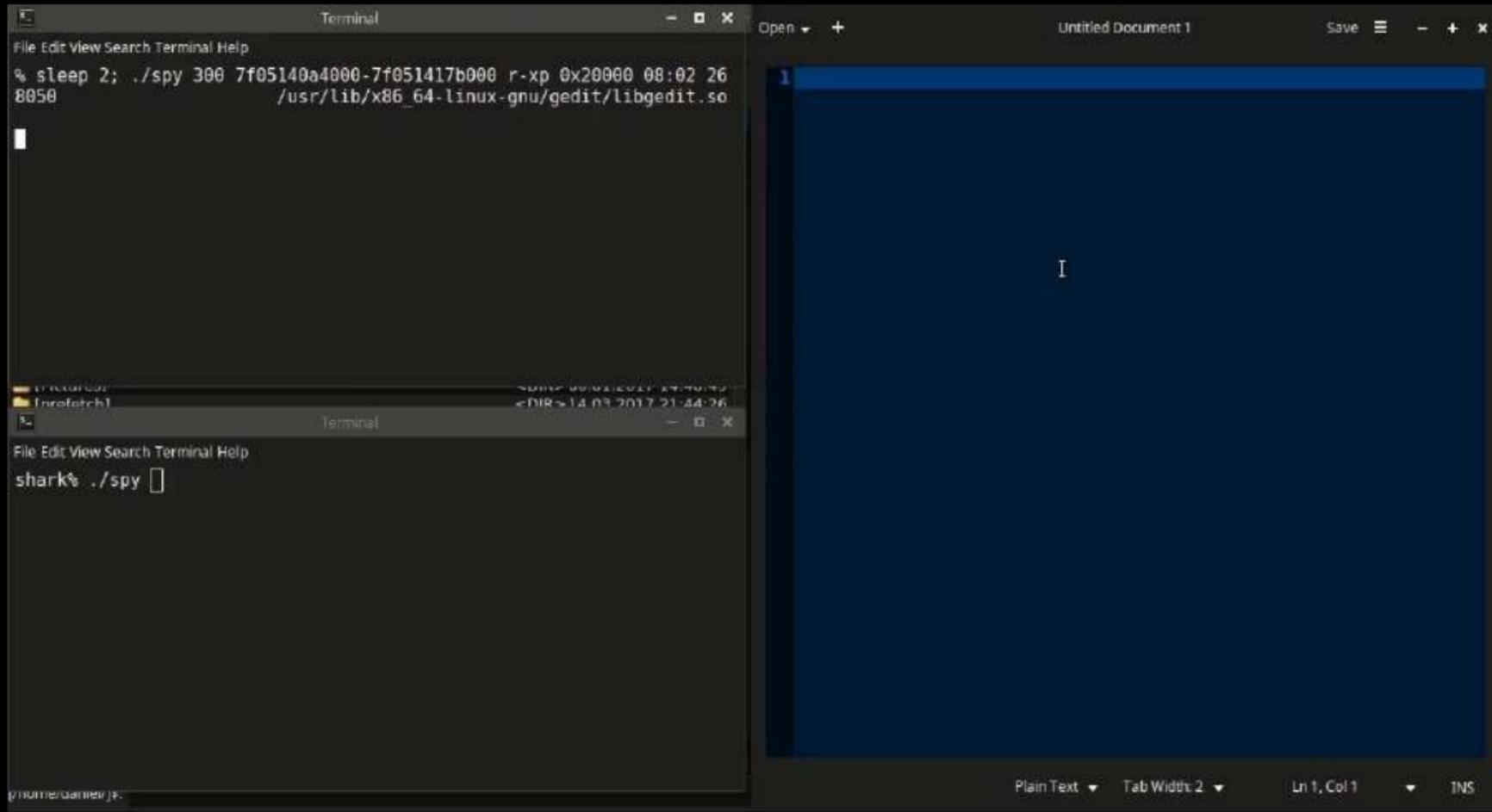
Assembly  4.67

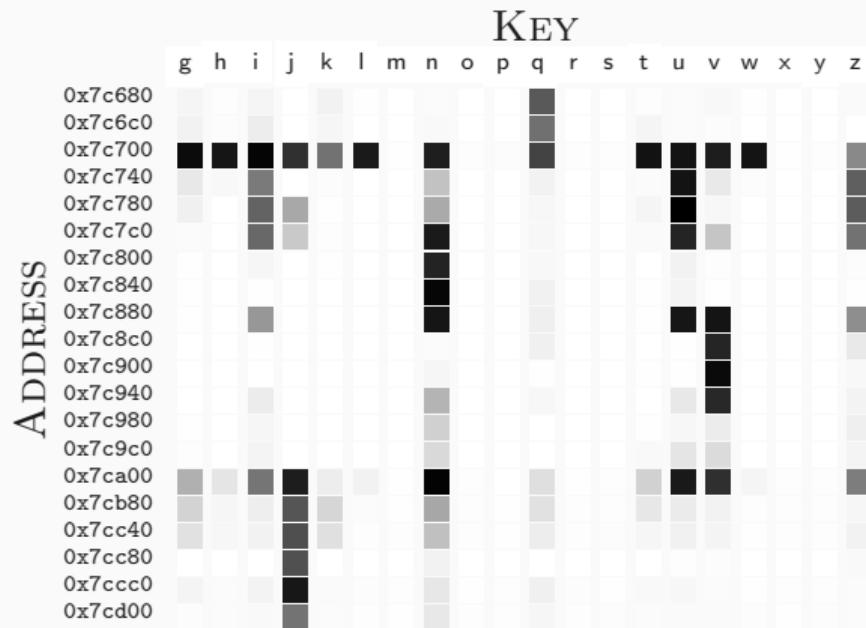
Optimized

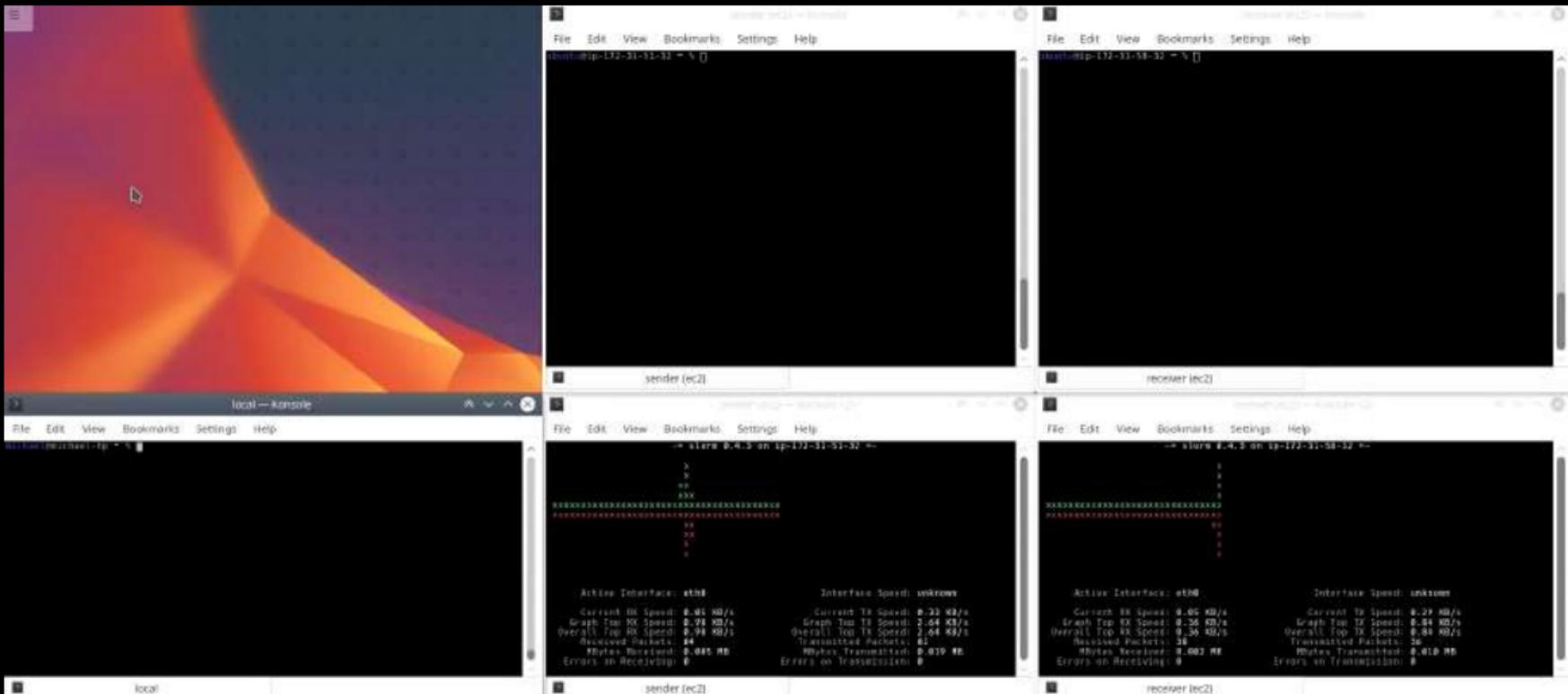
```
1 mov &timestamp, %rcx
2 1: inc %rax
3 mov %rax, (%rcx)
4 jmp 1b
```

CPU cycles one increment takes









HELLO FROM THE OTHER SIDE (DEMO):
VIDEO STREAMING OVER CACHE COVERT CHANNEL

Protection from Side-Channel Attacks

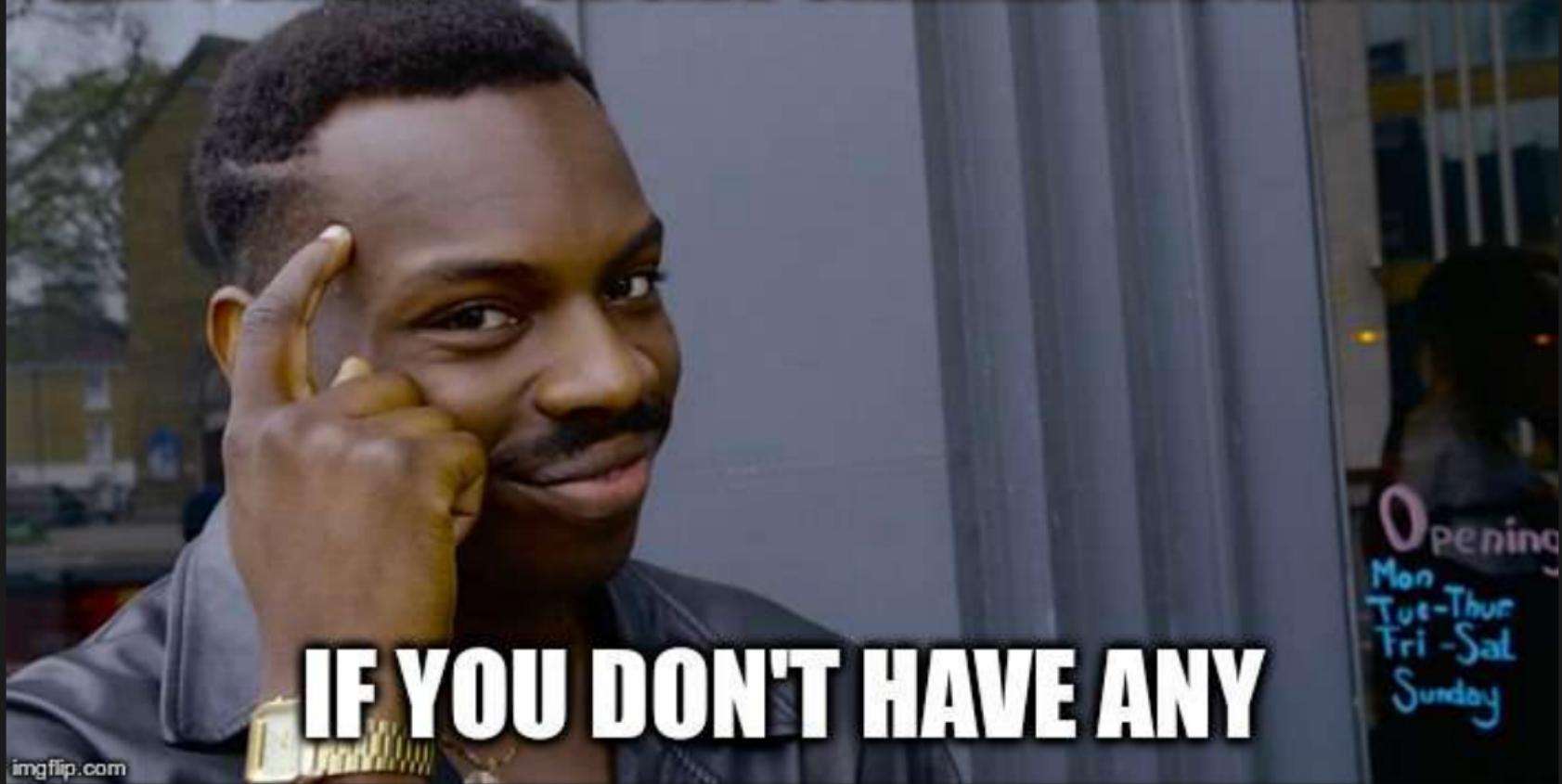
Protection from Side-Channel Attacks

Intel SGX does not provide explicit protection from side-channel attacks.

Protection from Side-Channel Attacks

Intel SGX does not provide explicit protection from side-channel attacks. It is the enclave developer's responsibility to address side-channel attack concerns.

CAN'T BREAK YOUR SIDE-CHANNEL PROTECTIONS



IF YOU DON'T HAVE ANY

- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX





- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX

Teechain

[...] We assume the TEE guarantees to hold



- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX

Teechain

[...] We assume the TEE guarantees to hold and do not consider side-channel attacks [5, 35, 46] on the TEE.

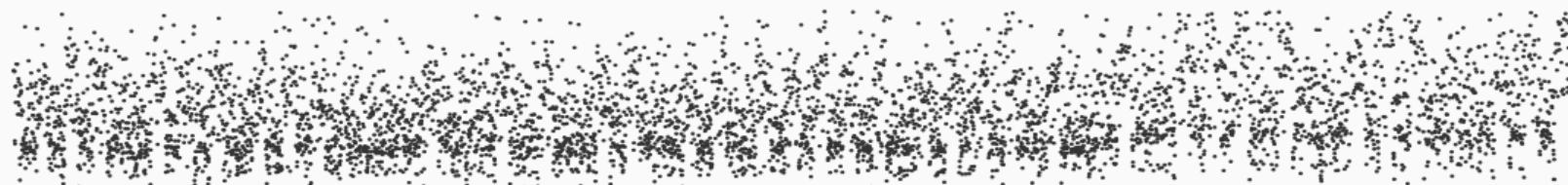


- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX

Teechain

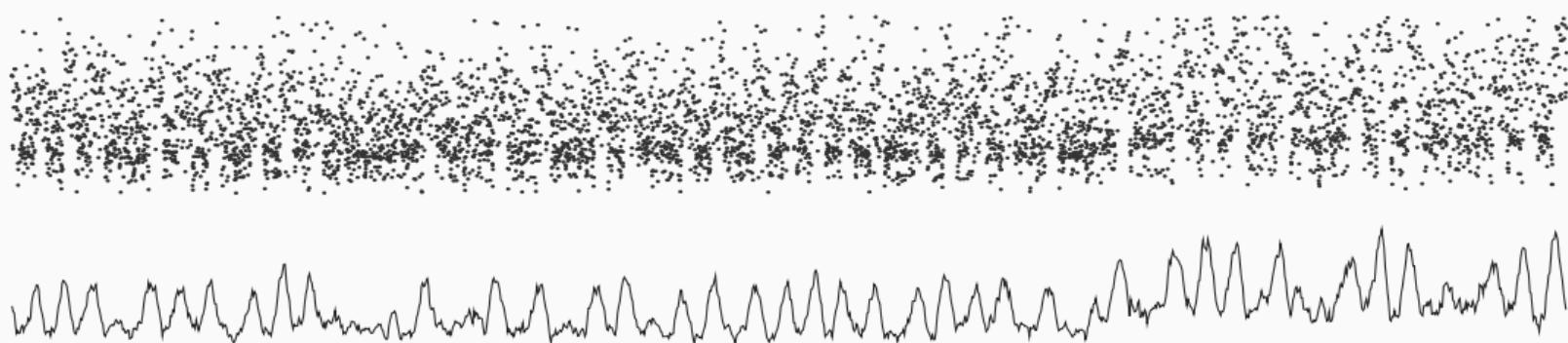
[...] We assume the TEE guarantees to hold and do not consider side-channel attacks [5, 35, 46] on the TEE. Such attacks and their mitigations [36, 43] are outside the scope of this work. [...]

Raw Prime+Probe trace...¹



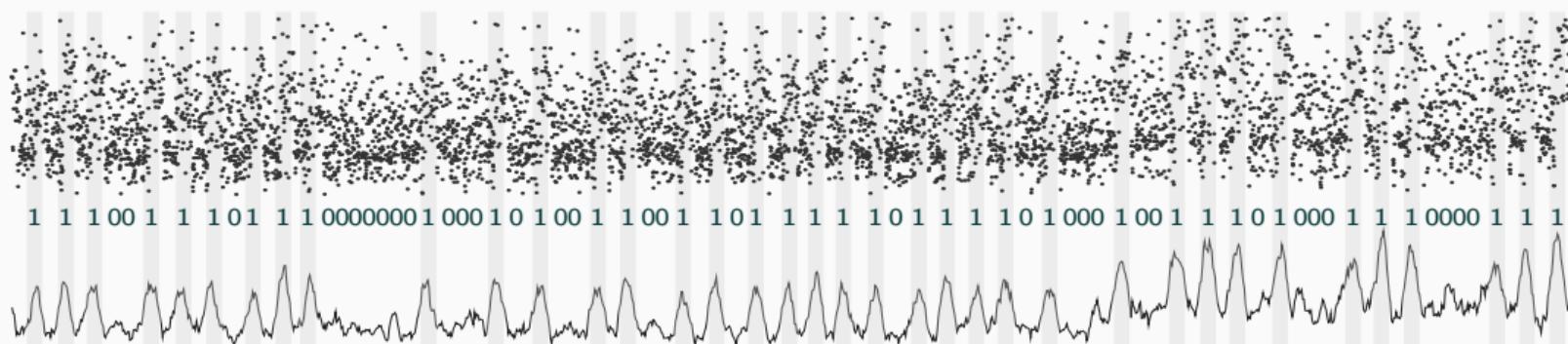
¹ Michael Schwarz et al. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA. 2017.

...processed with a simple moving average...²



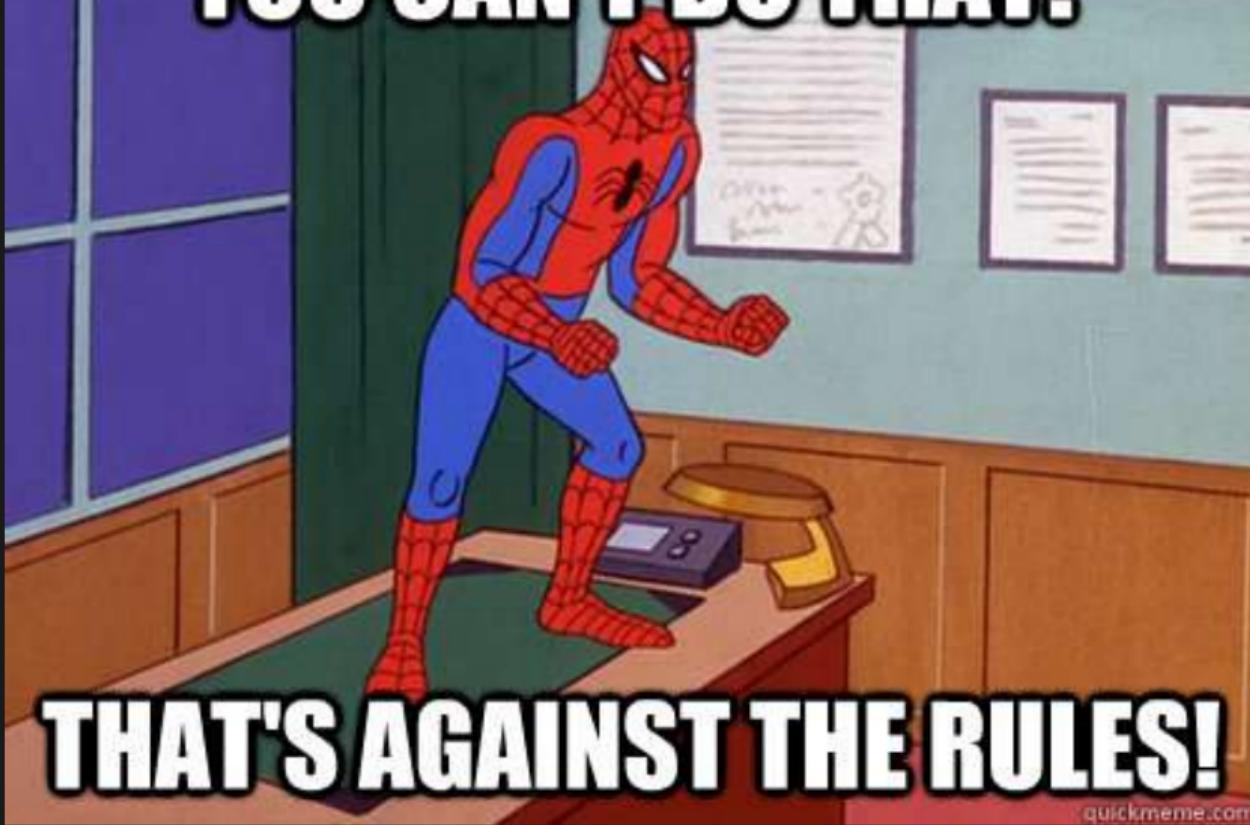
²Michael Schwarz et al. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA. 2017.

...allows to clearly see the bits of the exponent³



³Michael Schwarz et al. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA. 2017.

YOU CAN'T DO THAT!



THAT'S AGAINST THE RULES!



Back to Work

*7. Serve with cooked
and peeled potatoes*



Wait for an hour





Wait for an hour

LATENCY

*1. Wash and cut
vegetables*

*2. Pick the basil leaves
and set aside*

*3. Heat 2 tablespoons of
oil in a pan*

*4. Fry vegetables until
golden and softened*



Dependency

1. Wash and cut vegetables

2. Pick the basil leaves and set aside

3. Heat 2 tablespoons of oil in a pan

4. Fry vegetables until golden and softened



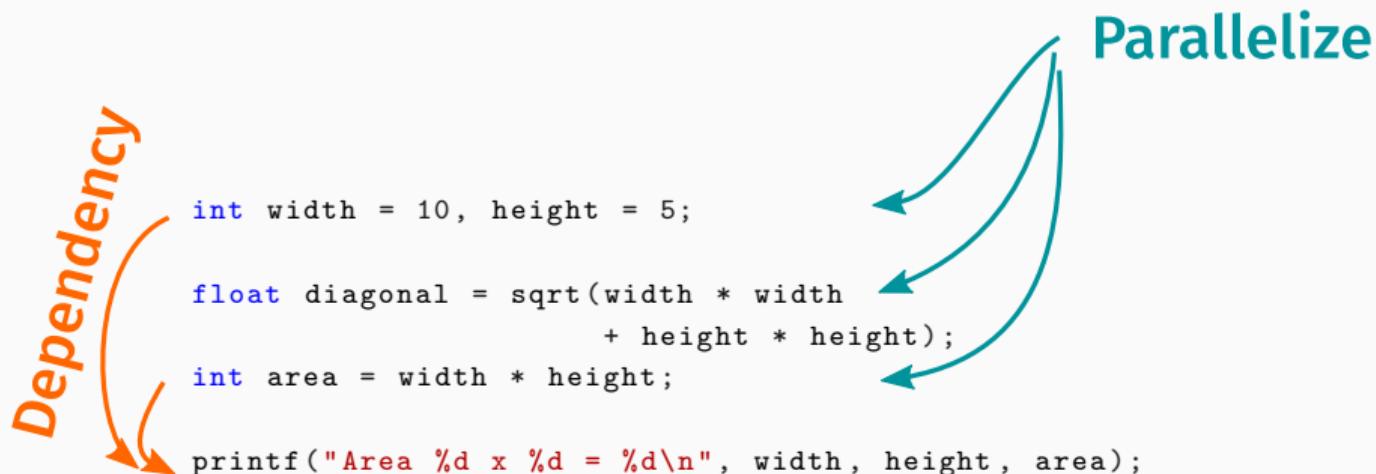
Parallelize



```
int width = 10, height = 5;

float diagonal = sqrt(width * width
                      + height * height);
int area = width * height;

printf("Area %d x %d = %d\n", width, height, area);
```

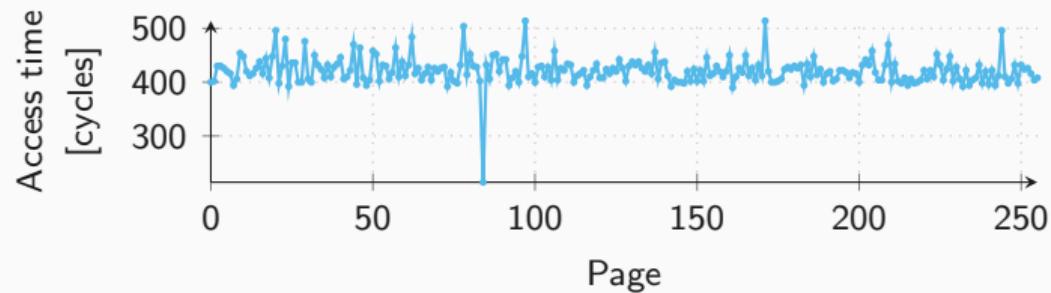




```
*(volatile char*) 0;  
array [84 * 4096] = 0;
```

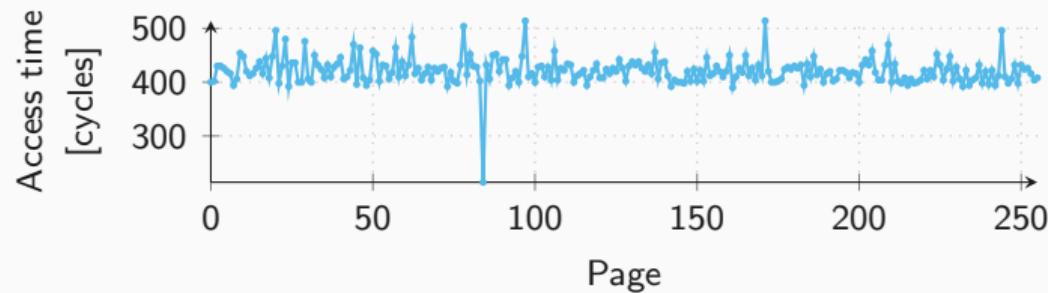


- Flush+Reload over all pages of the array





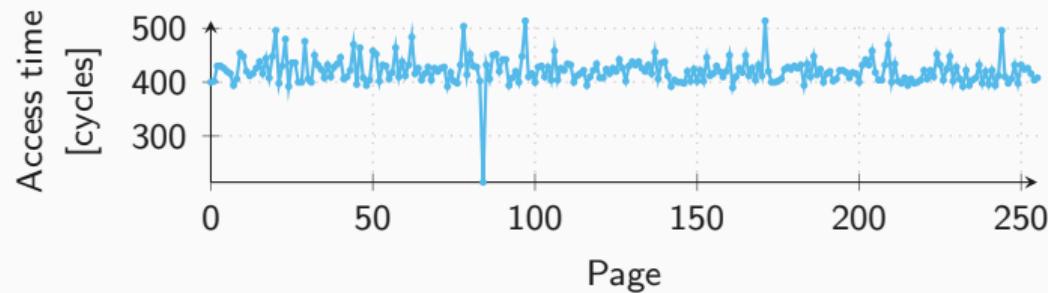
- Flush+Reload over all pages of the array



- “Unreachable” code line was **actually executed**



- Flush+Reload over all pages of the array



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



- Out-of-order instructions leave microarchitectural traces



- Out-of-order instructions **leave microarchitectural traces**
 - We can see them for example through the cache



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



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



- Add another **layer of indirection** to test

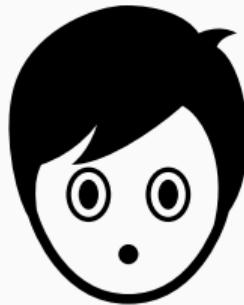
```
char data = *(char*) 0xffffffff81a000e0;  
array[data * 4096] = 0;
```



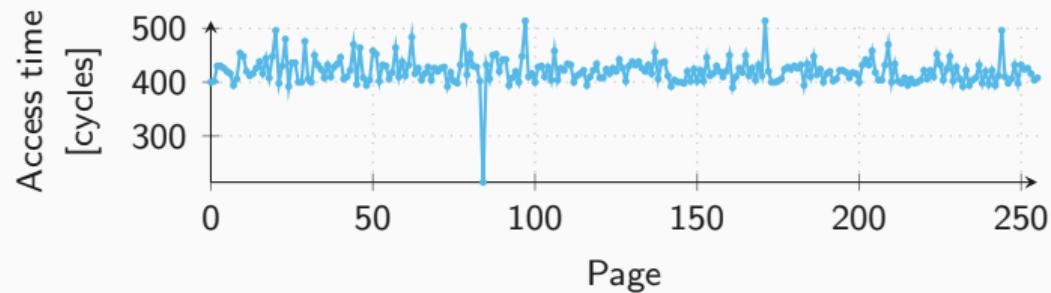
- Add another **layer of indirection** to test

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

- Then check whether any part of array is **cached**



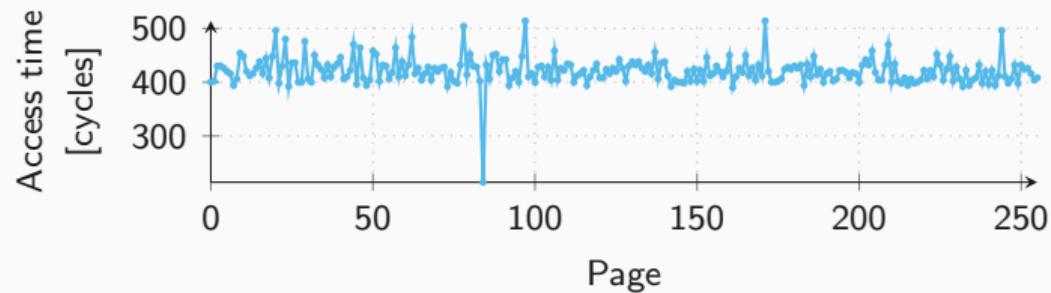
- Flush+Reload over all pages of the array



- Index of cache hit reveals **data**



- Flush+Reload over all pages of the array

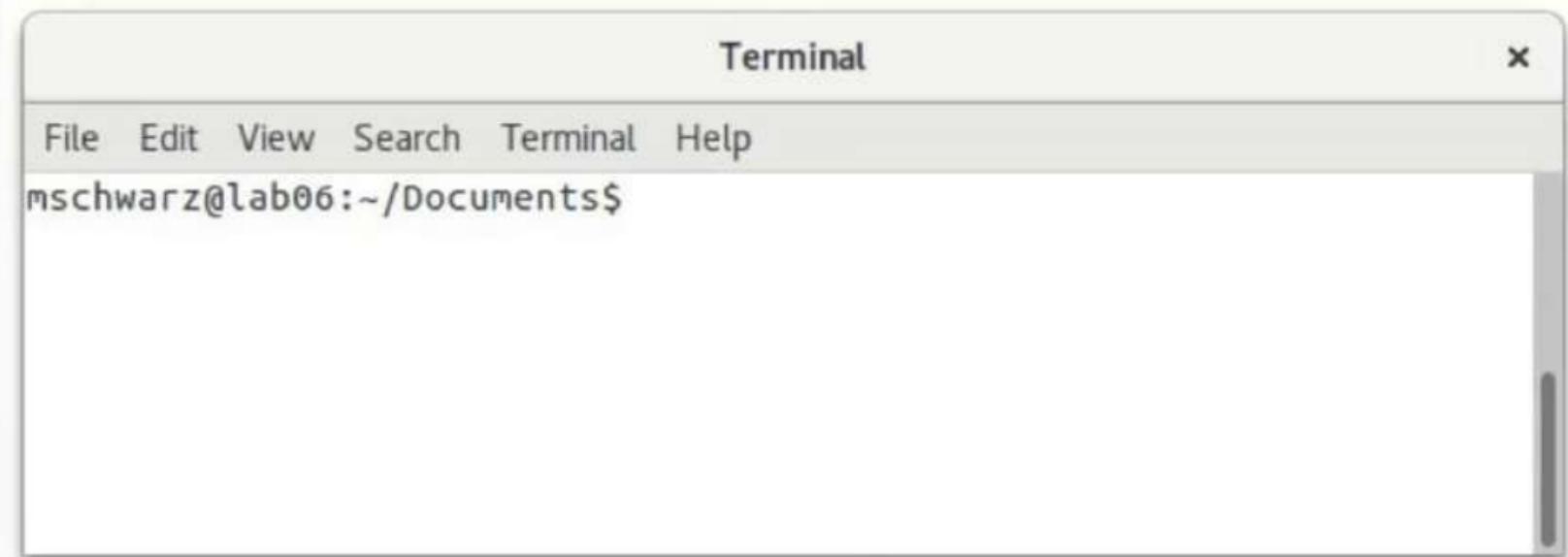
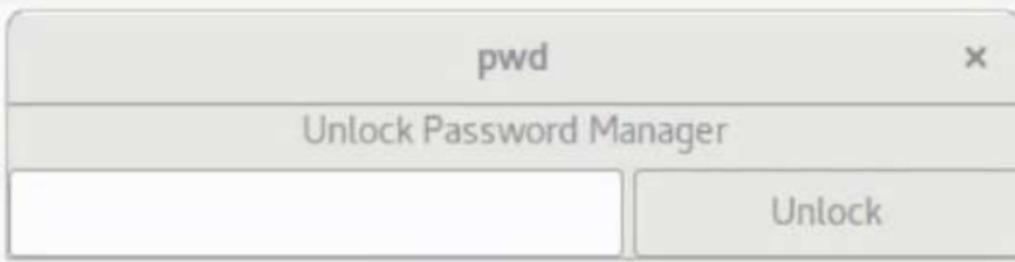


- Index of cache hit reveals **data**
- Permission check is in some cases **not fast enough**

I SHIT YOU NOT

**THERE WAS KERNEL MEMORY ALL
OVER THE TERMINAL**





File Edit View Search Terminal Help

attacker@meltdown ~/exploit %

File Edit View Search Terminal Help

victim@meltdown ~ %





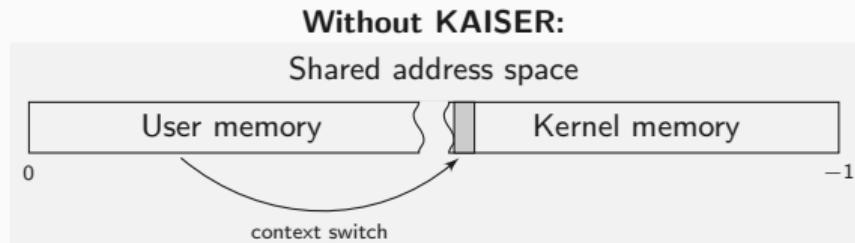
K_{er}n_el A_{dd}re_ss I_{sol}ation to have S_{ide} chann_{el}s E_{ff}icien_{tly} R_{em}oved

KAISER /'kʌɪzə/

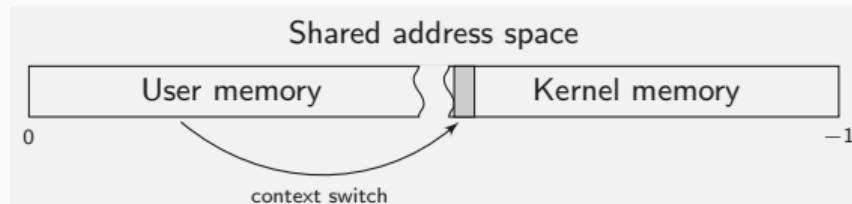
1. [german] Emperor, ruler of an empire
2. largest penguin, emperor penguin



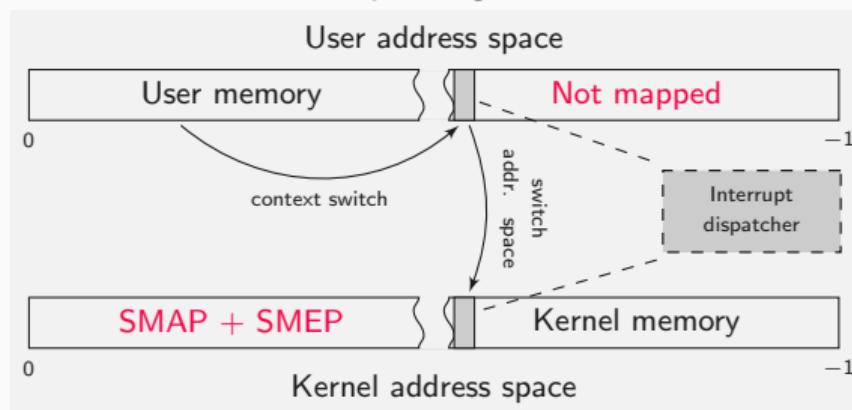
K_{er}nel A_{dd}ress I_{sol}ation to have S_{ide} channels E_{fficiently} R_{emoved}



Without KAISER:



With KAISER:



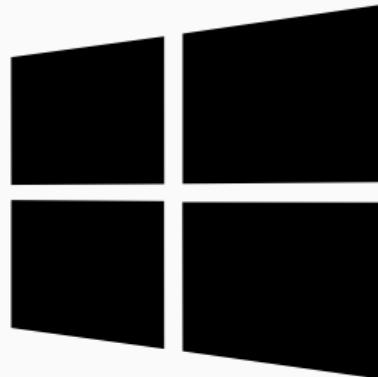




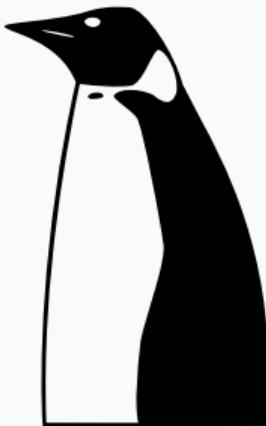
- Our patch
- Adopted in
Linux



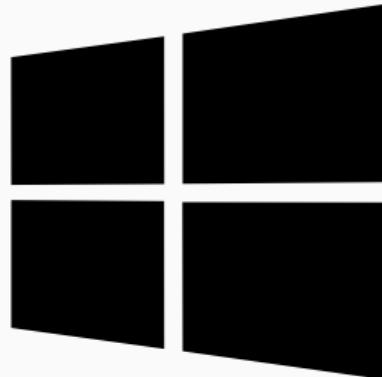
- Our patch
- Adopted in Linux



- Adopted in Windows



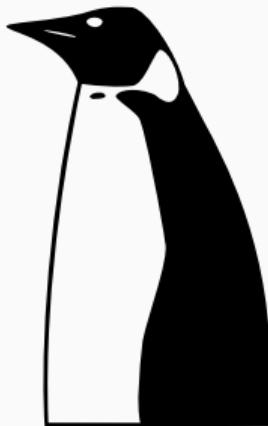
- Our patch
- Adopted in Linux



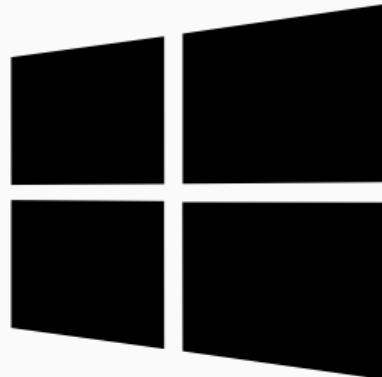
- Adopted in Windows



- Adopted in OSX/iOS



- Our patch
- Adopted in Linux



- Adopted in Windows



- Adopted in OSX/iOS

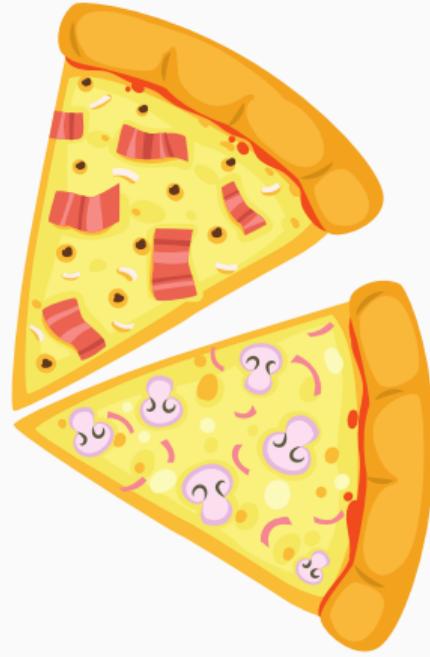
→ now in every computer

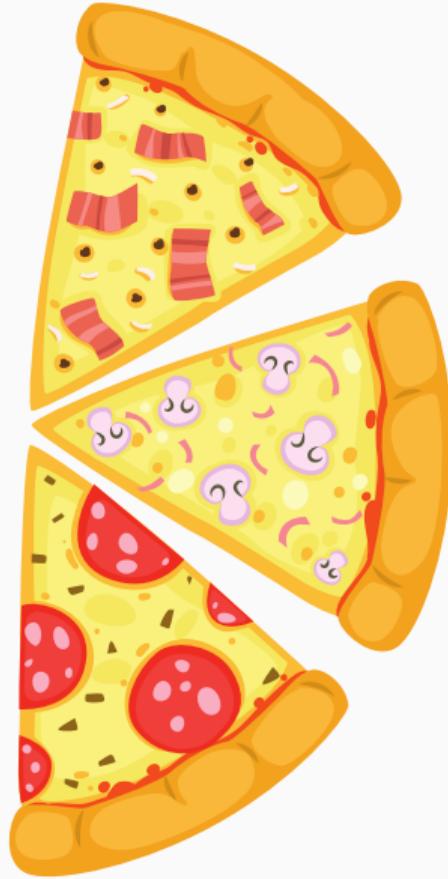


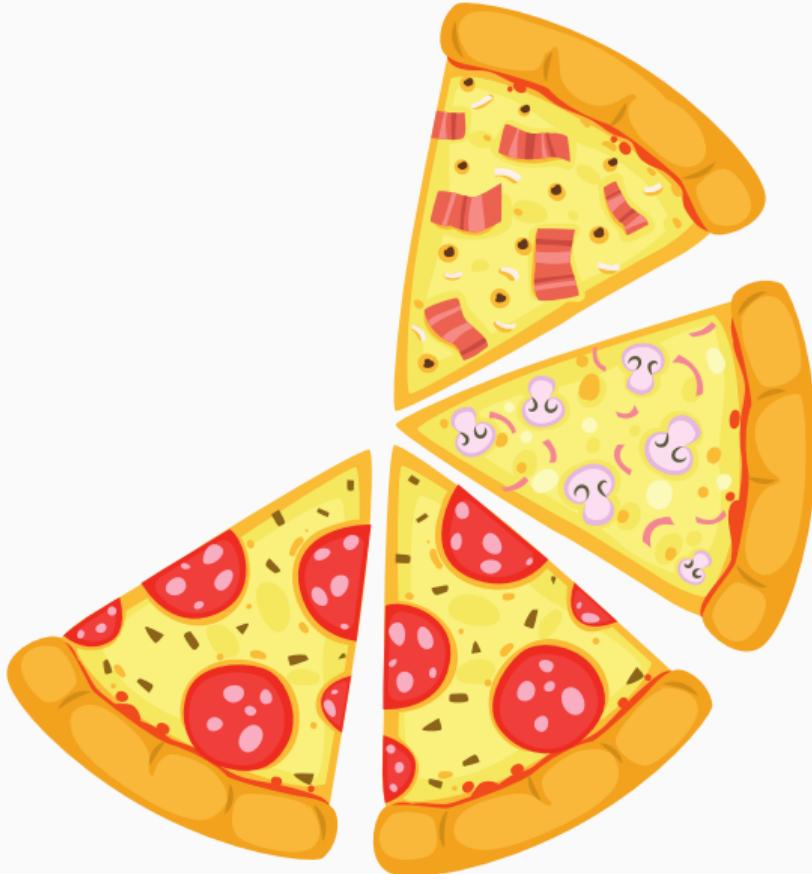
PIZZA

SPECIAL RECIPES

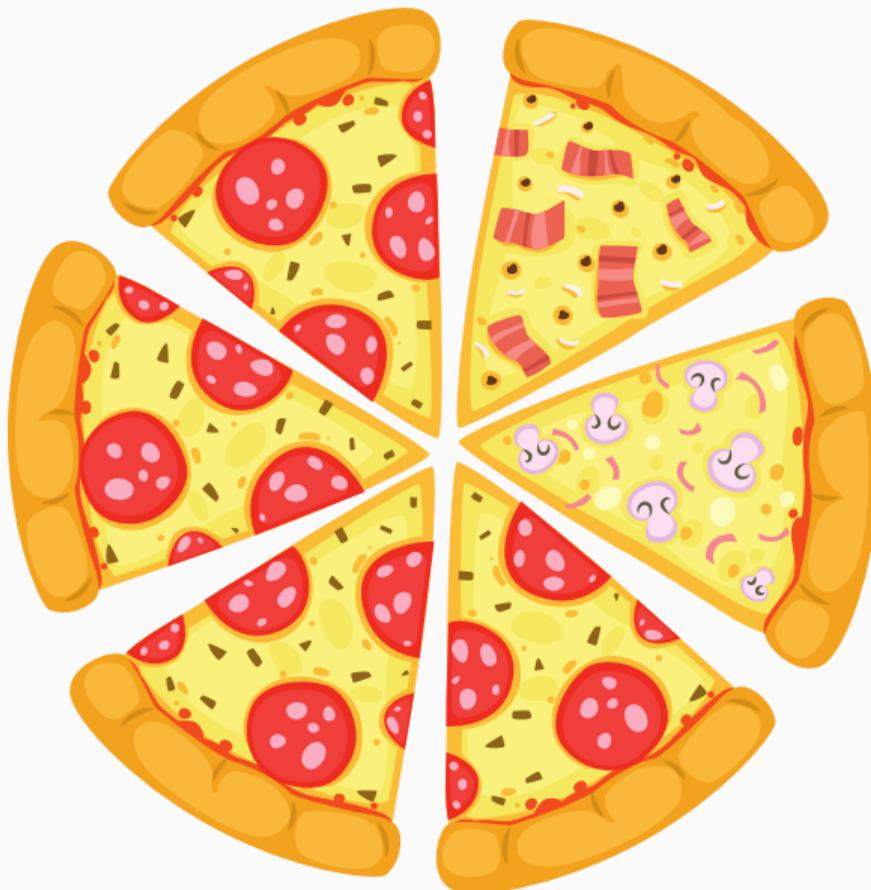














›A table for 6 please‹



Speculative Cooking





A table for 6 please





PIZZA

SPECIAL RECIPES



PIZZA

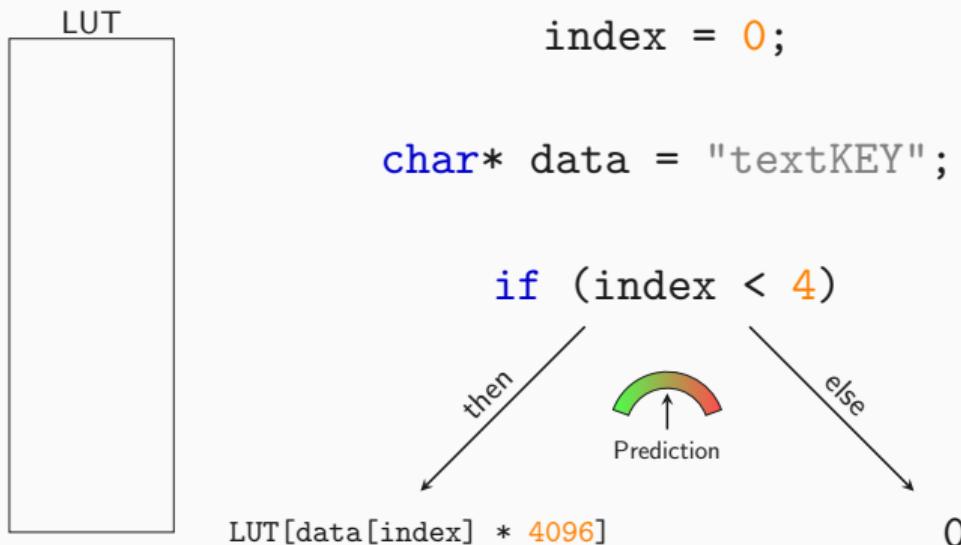
SPECIAL RECIPES

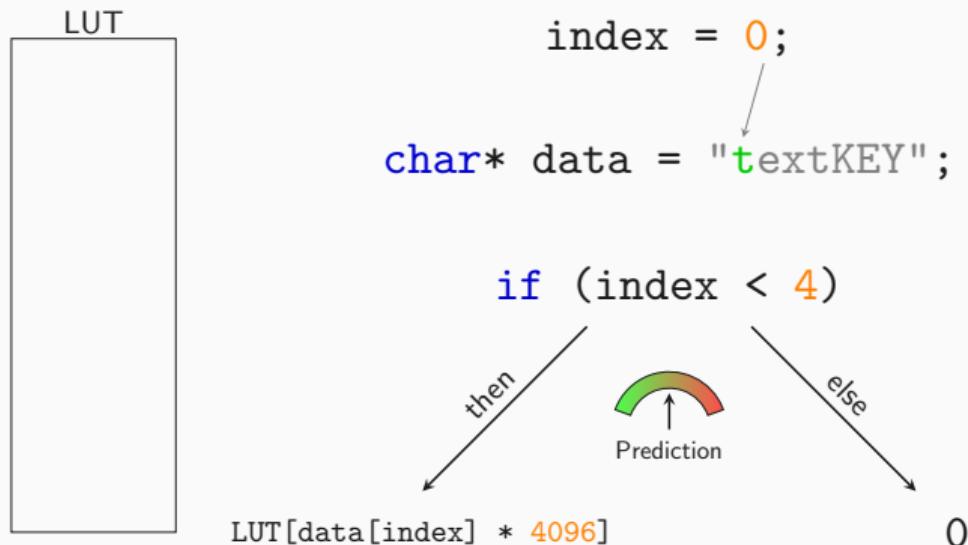
PIZZA

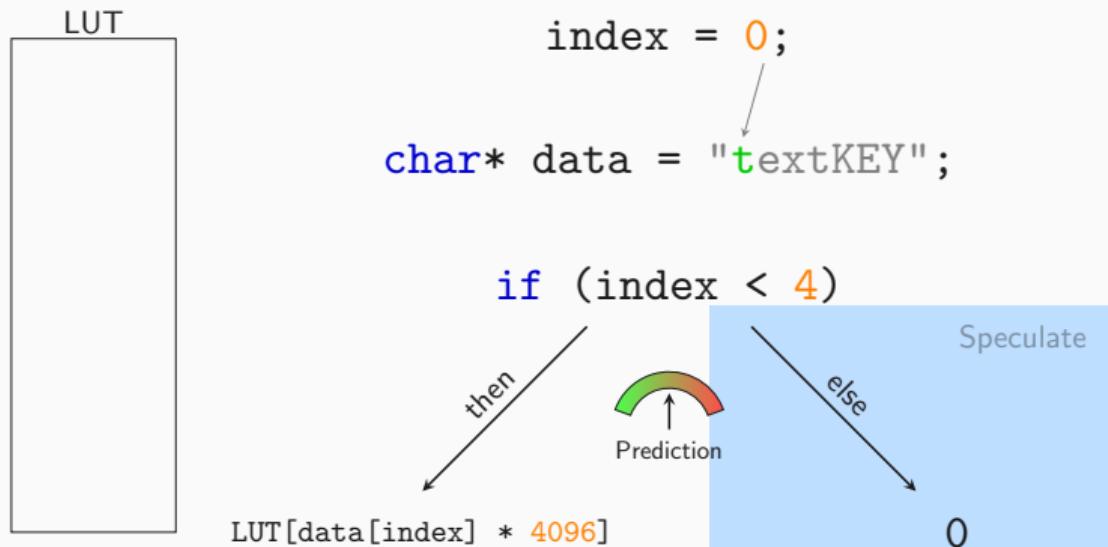


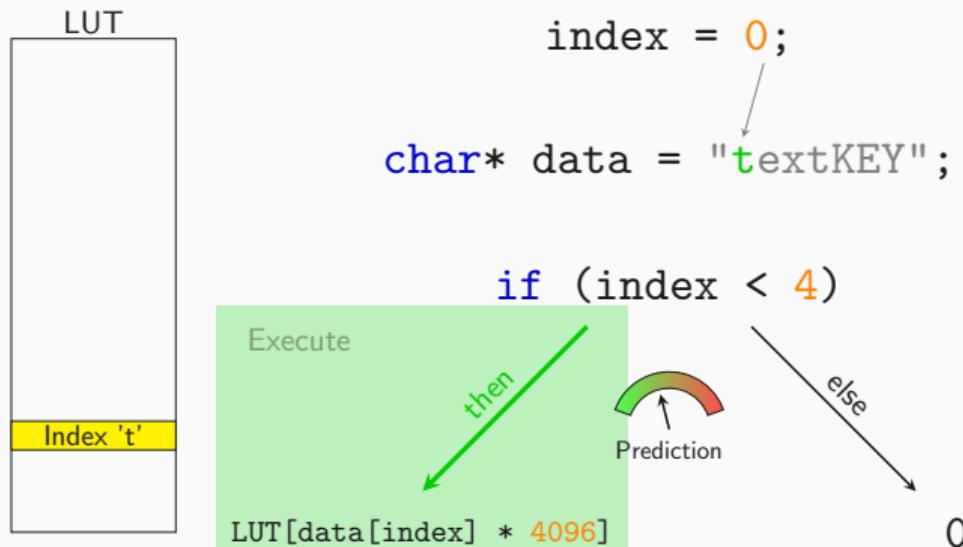


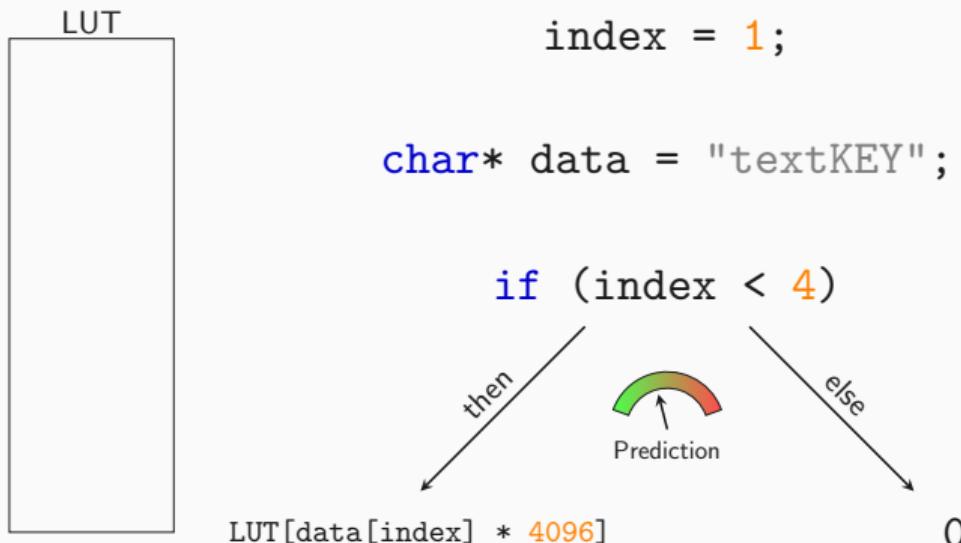


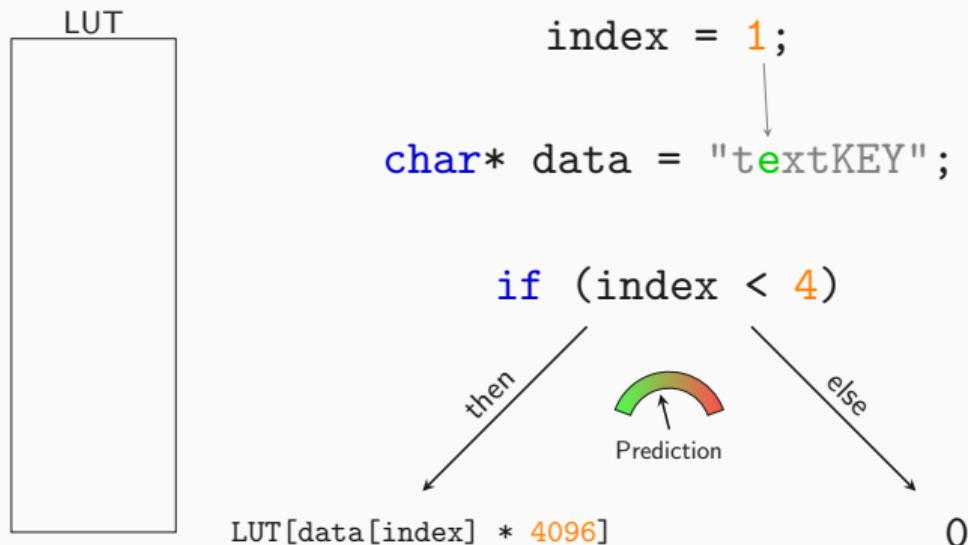


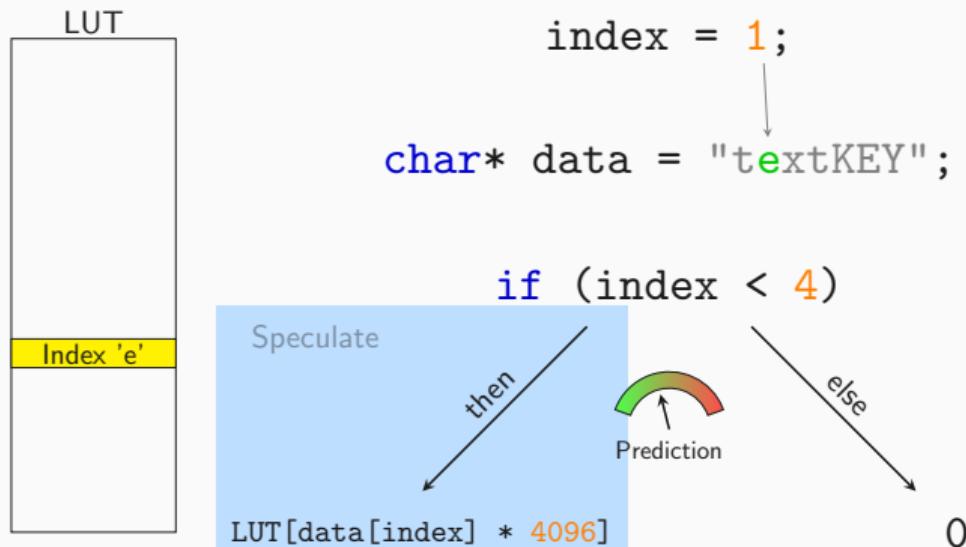


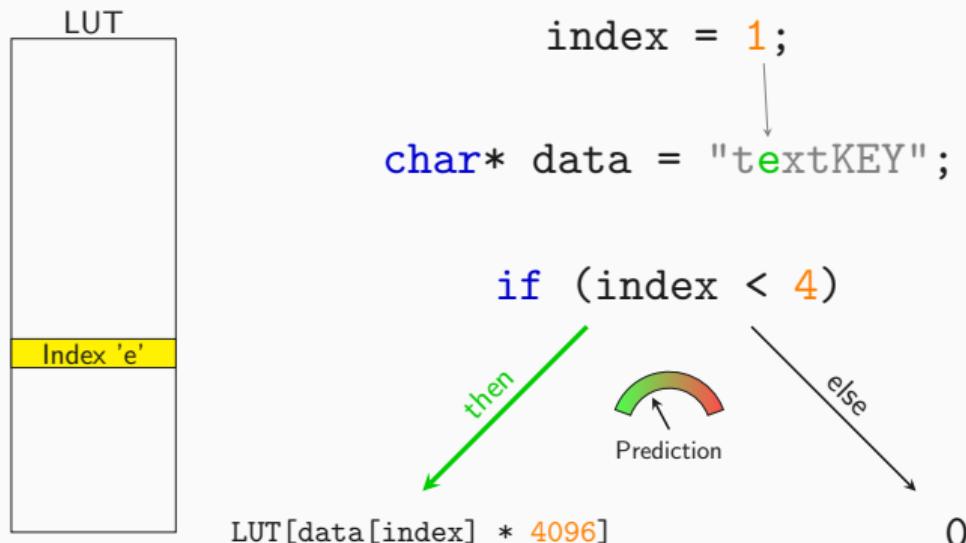


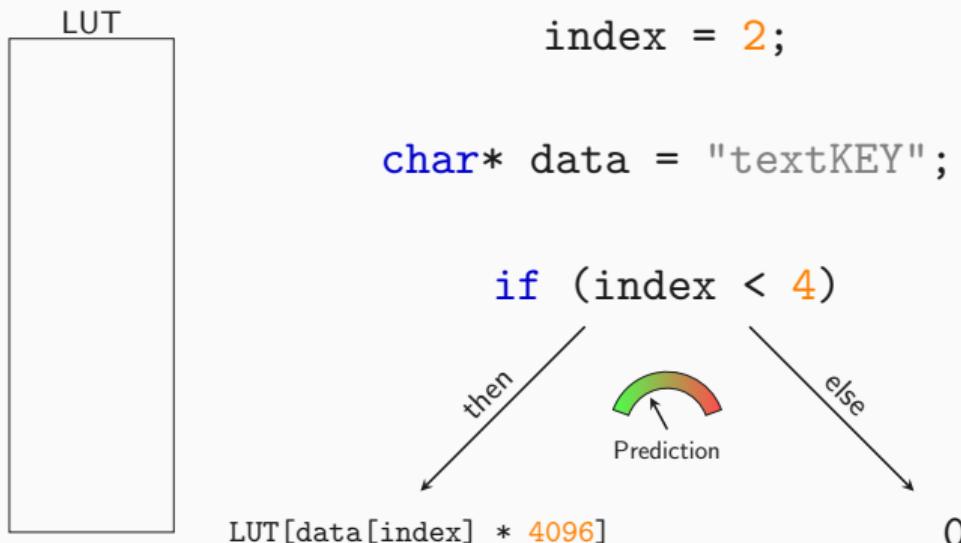


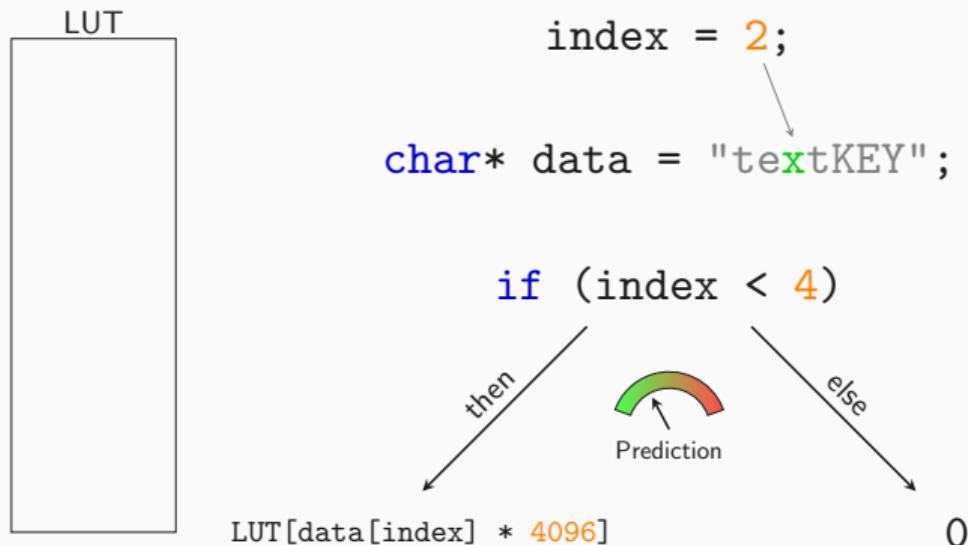


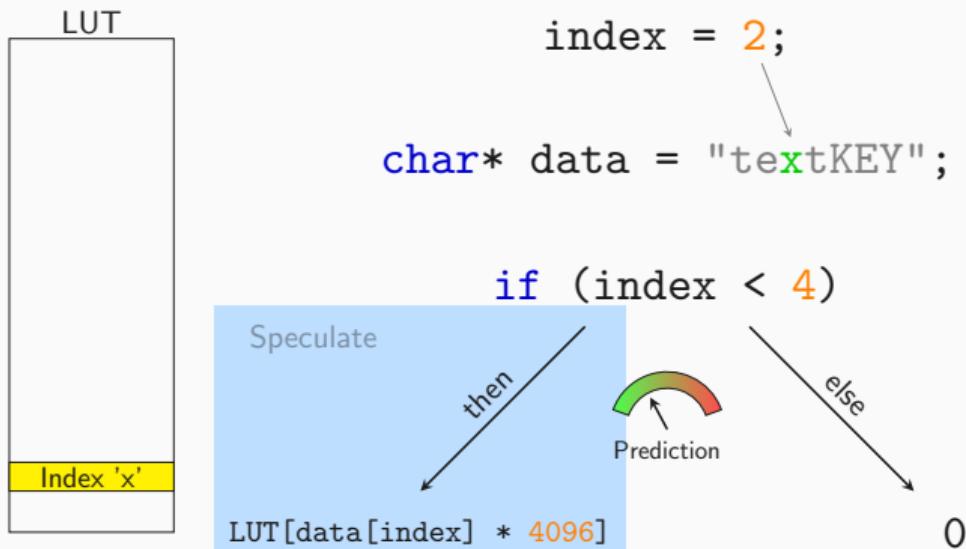


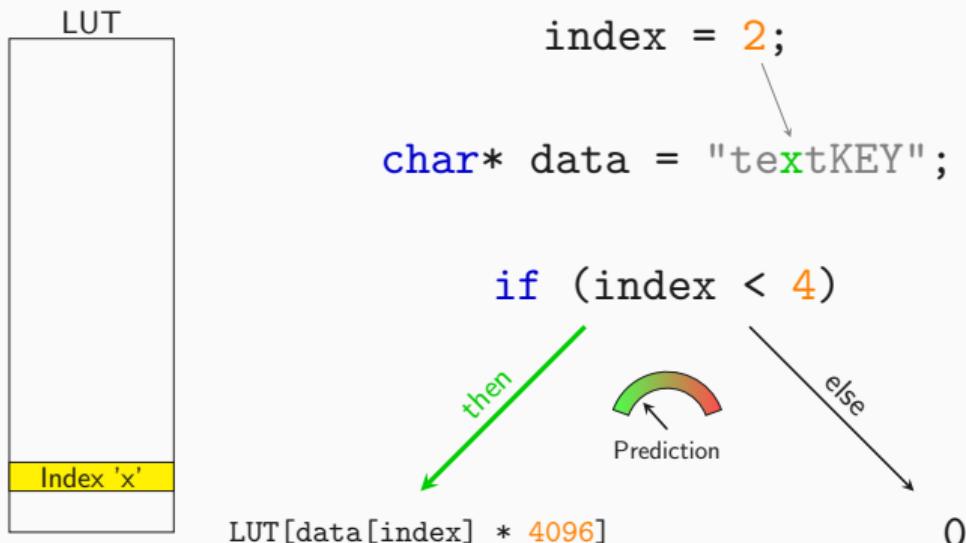


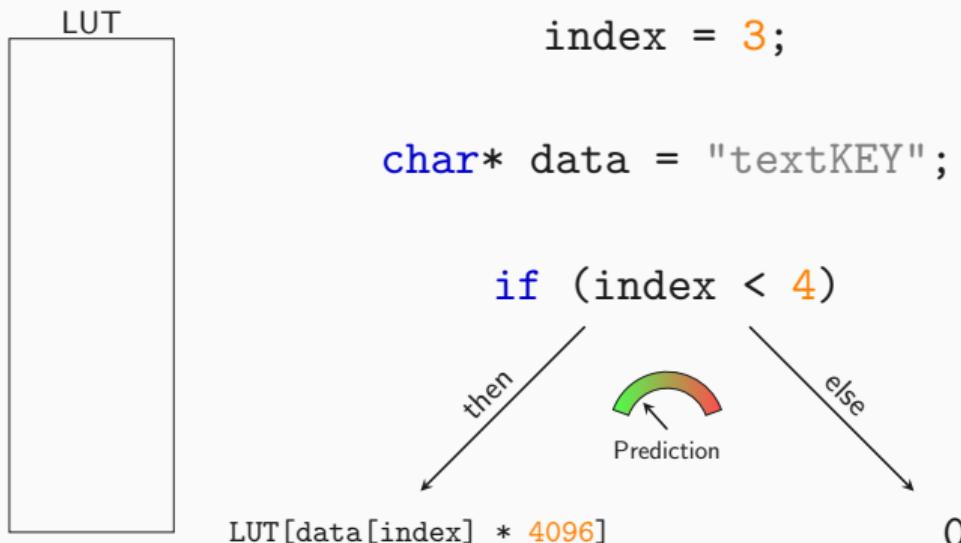


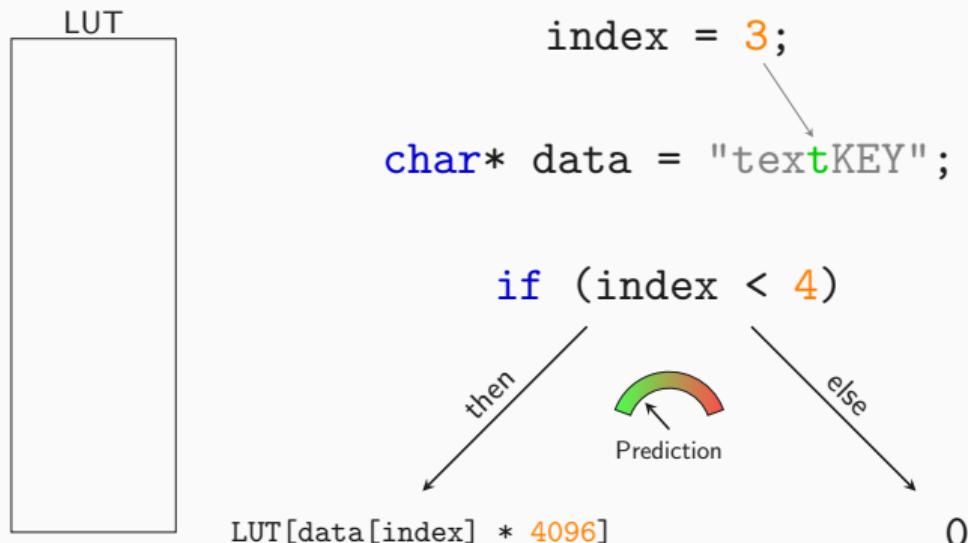


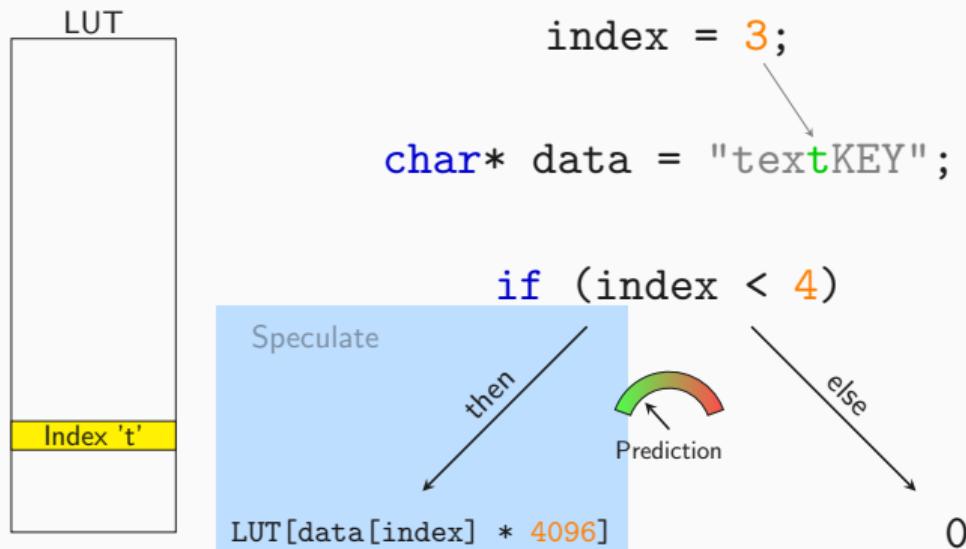


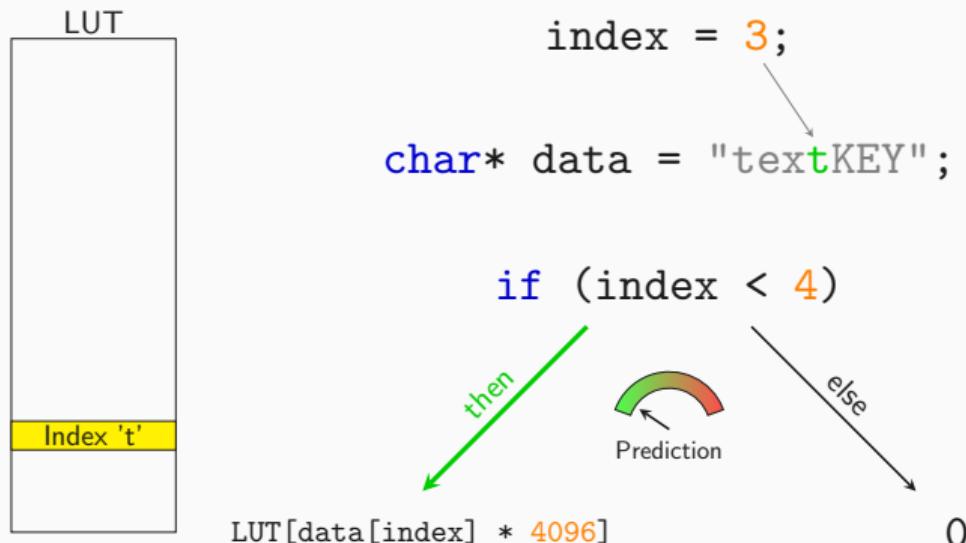


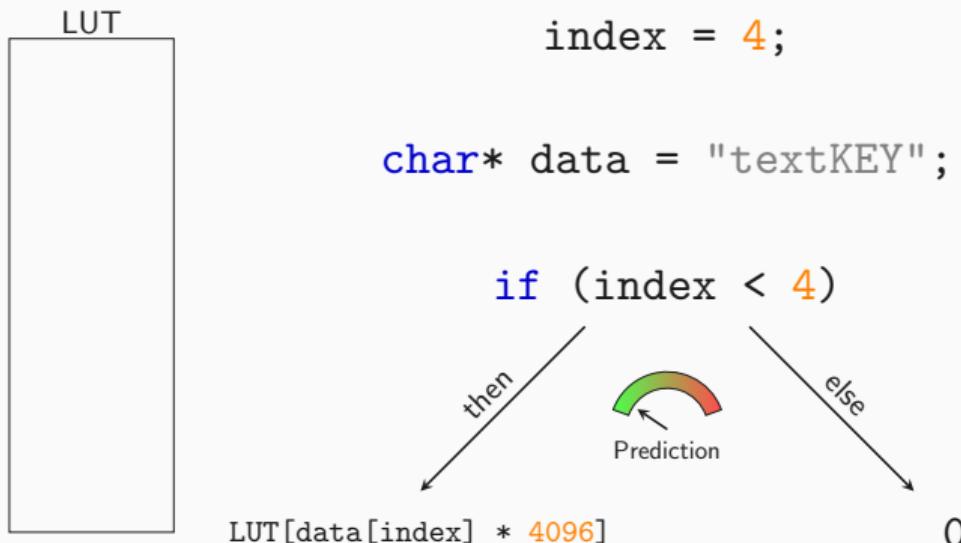


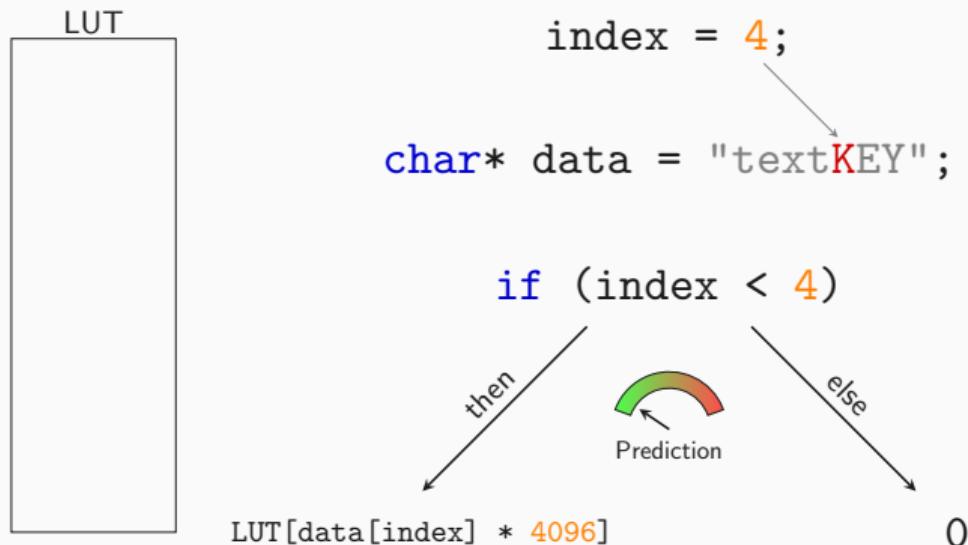


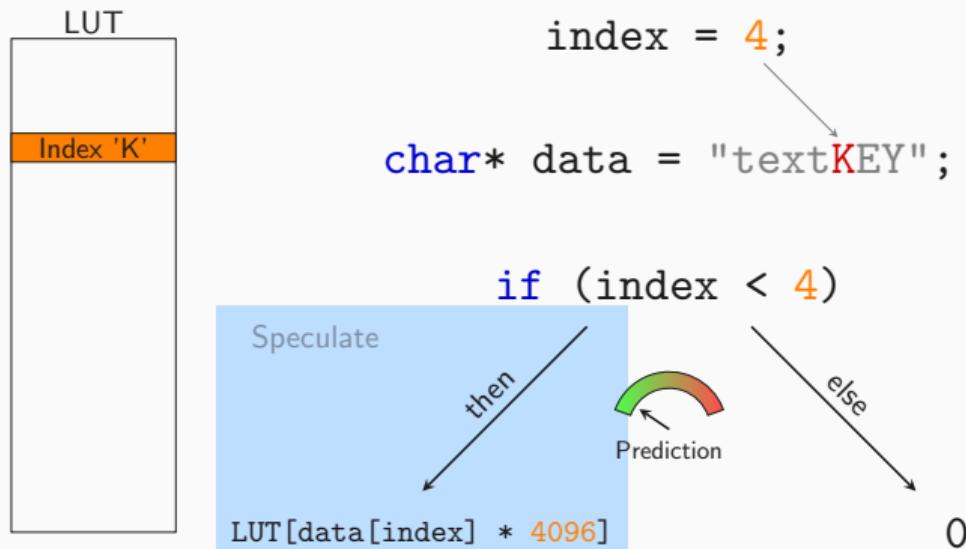


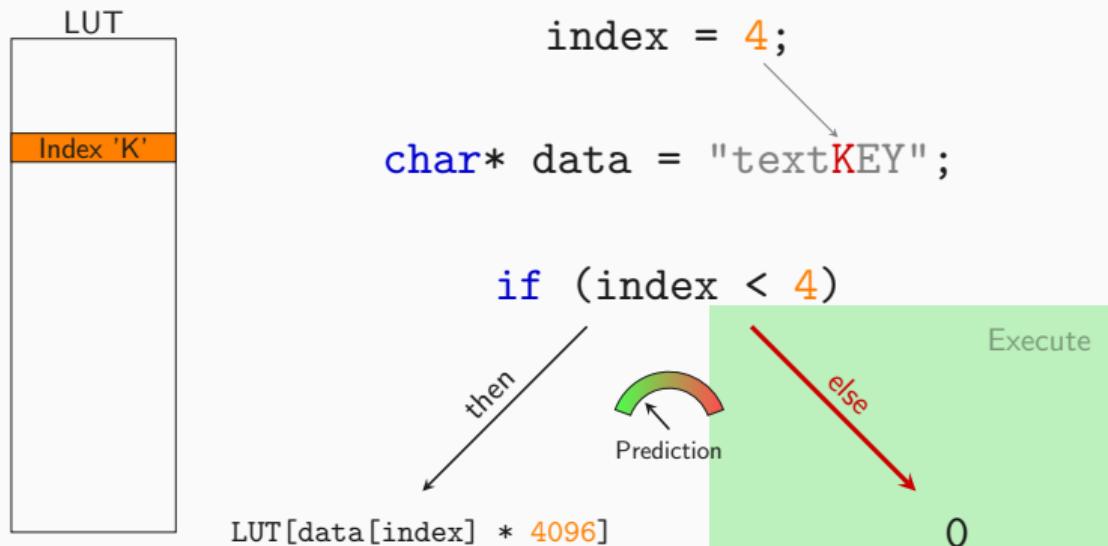


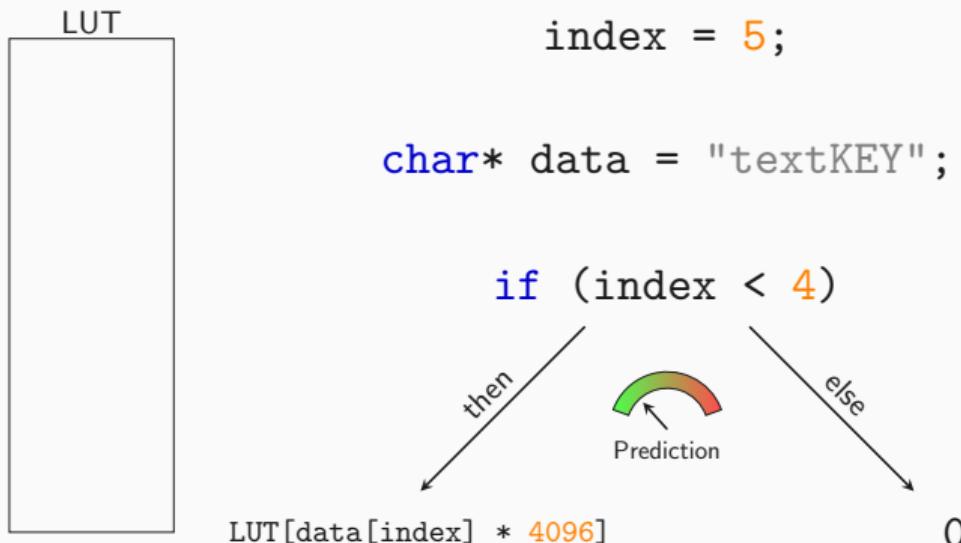


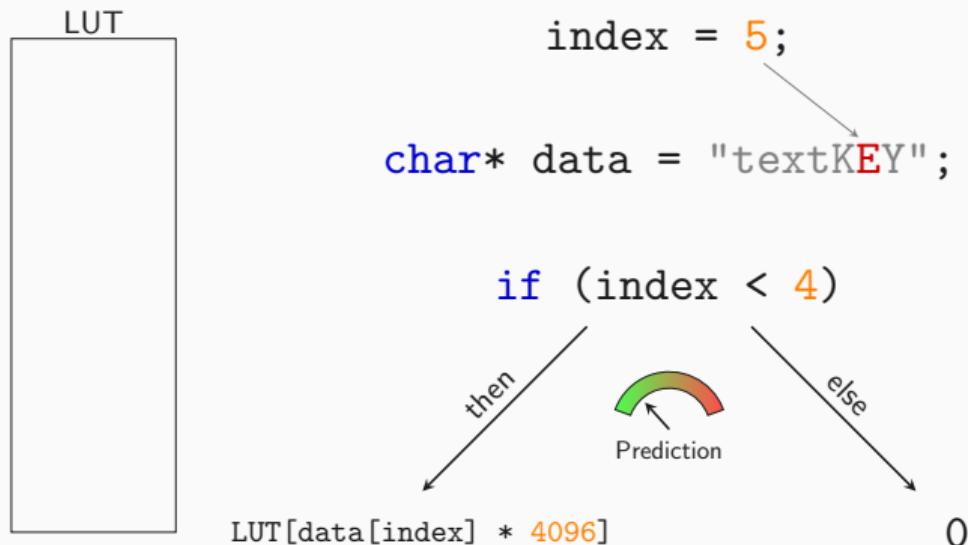


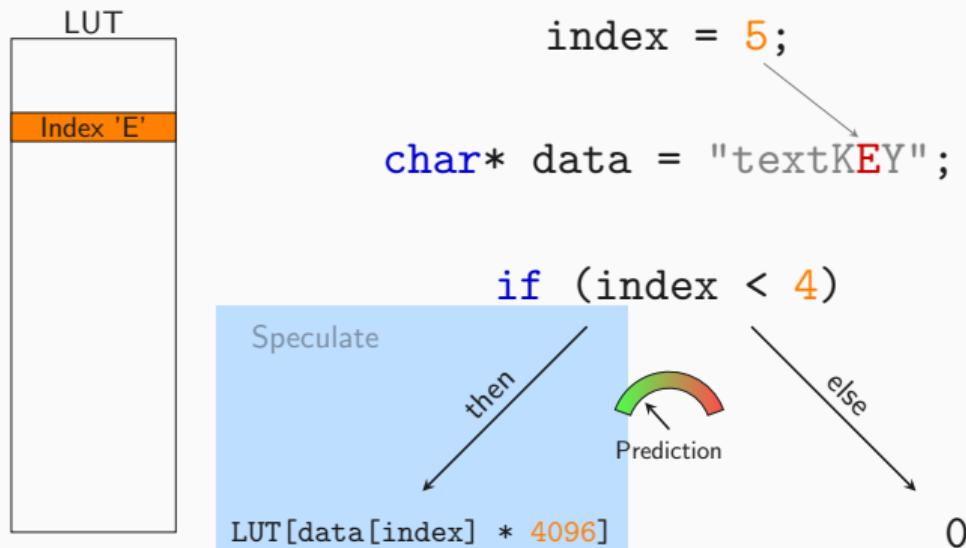


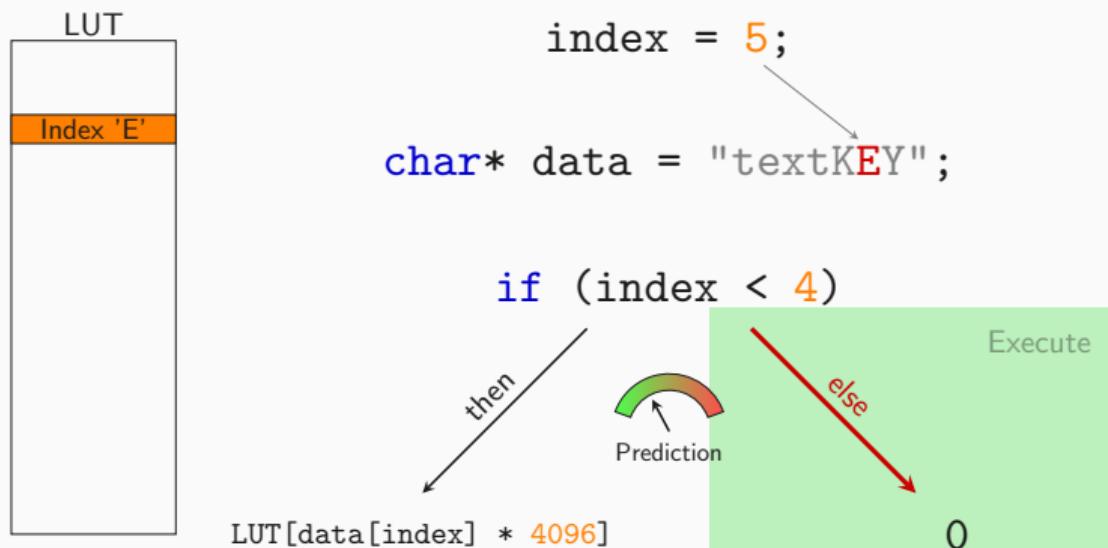


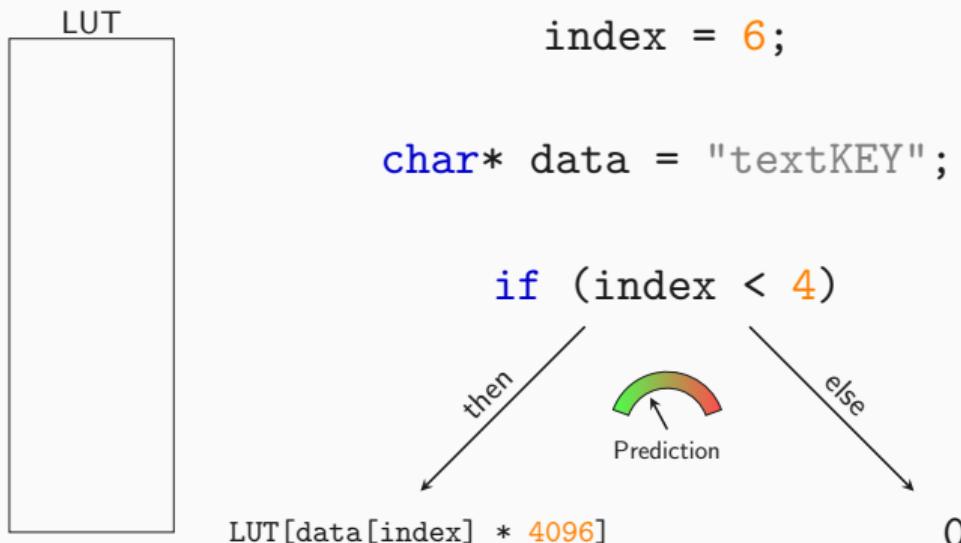


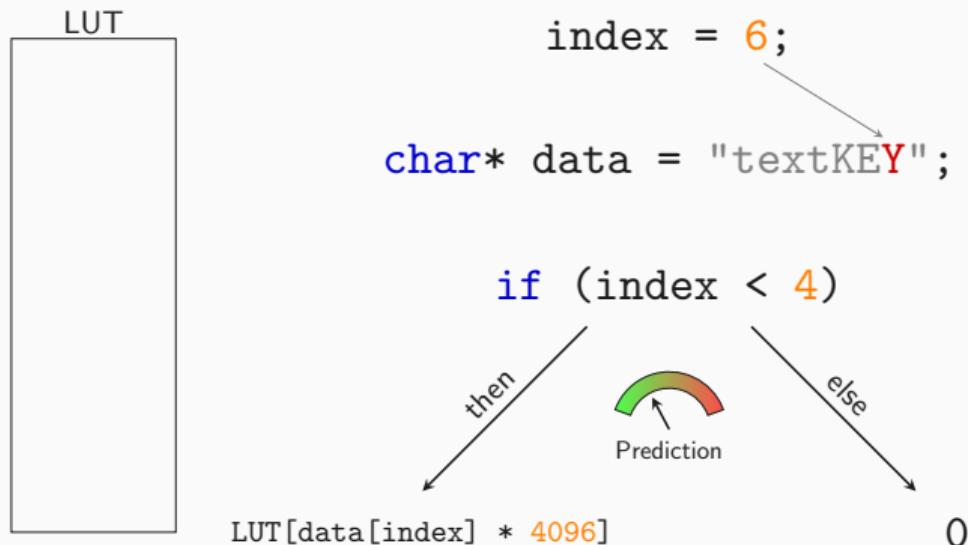


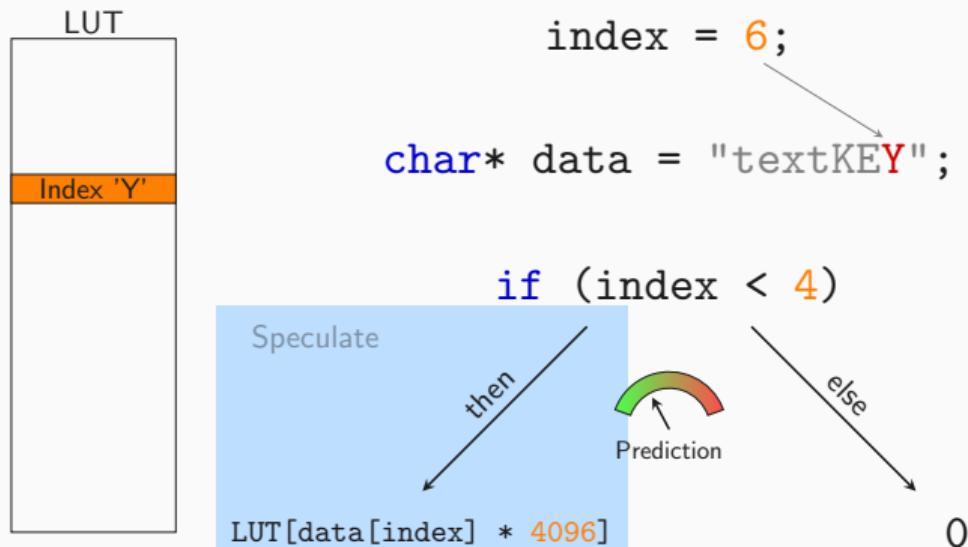


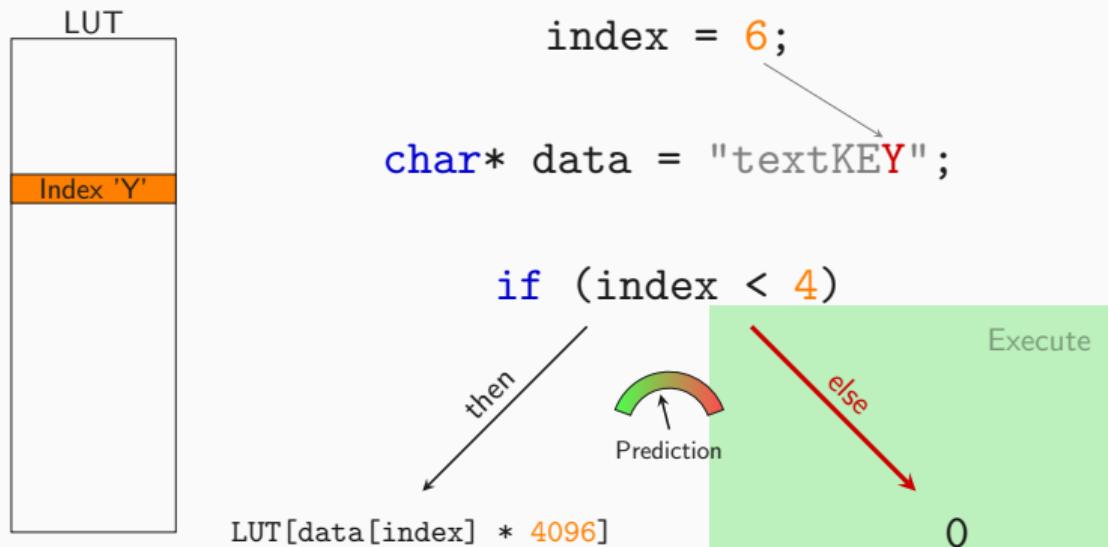




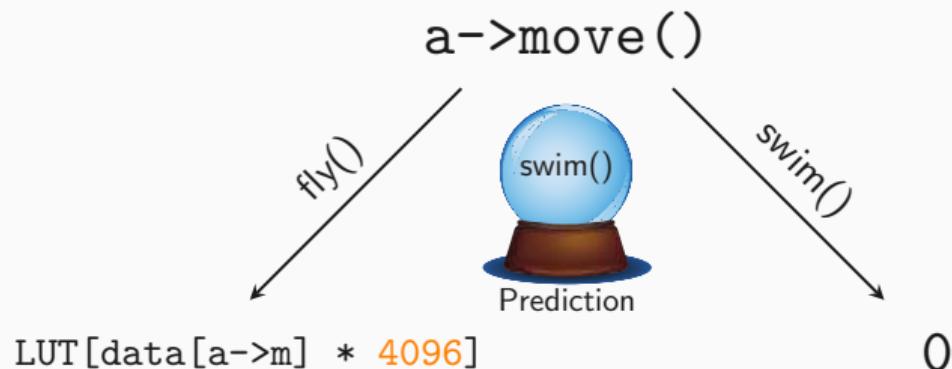




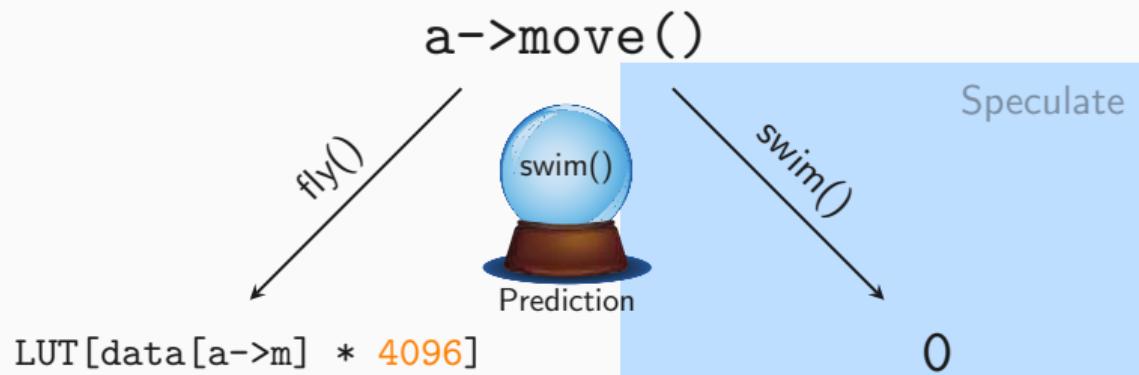




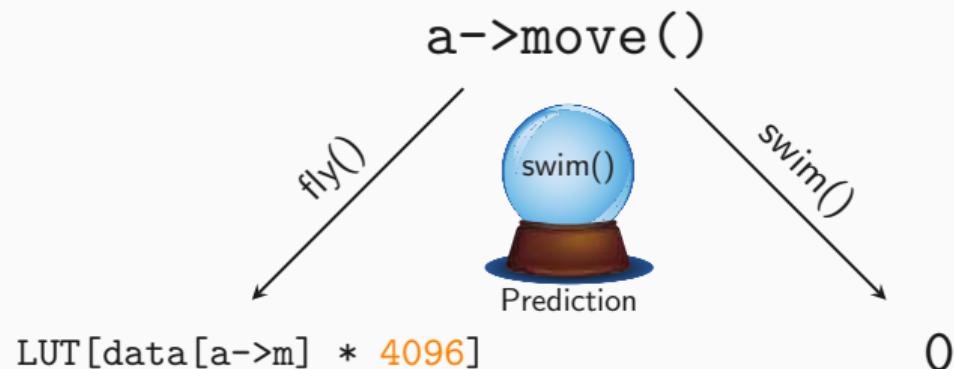
```
Animal* a = bird;
```



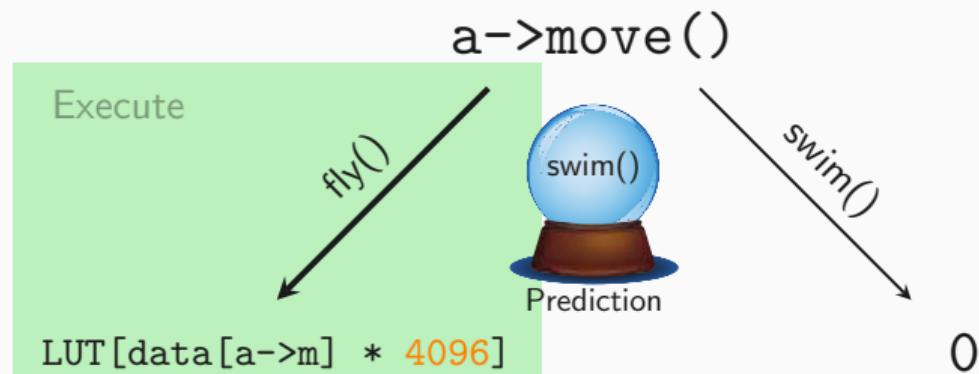
```
Animal* a = bird;
```



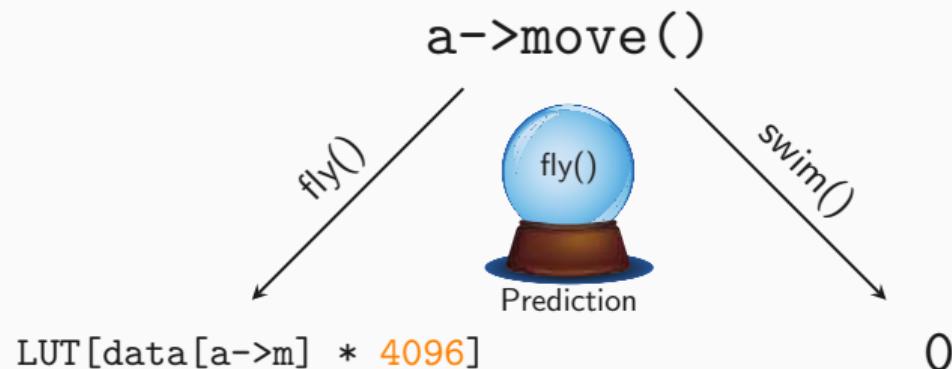
```
Animal* a = bird;
```



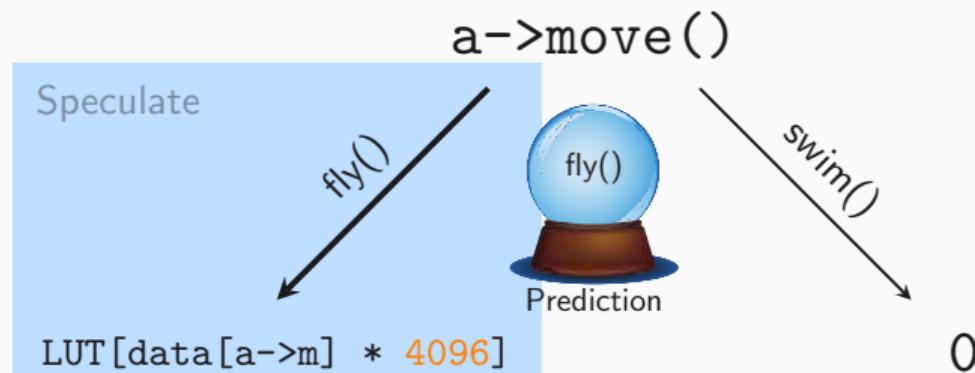
```
Animal* a = bird;
```



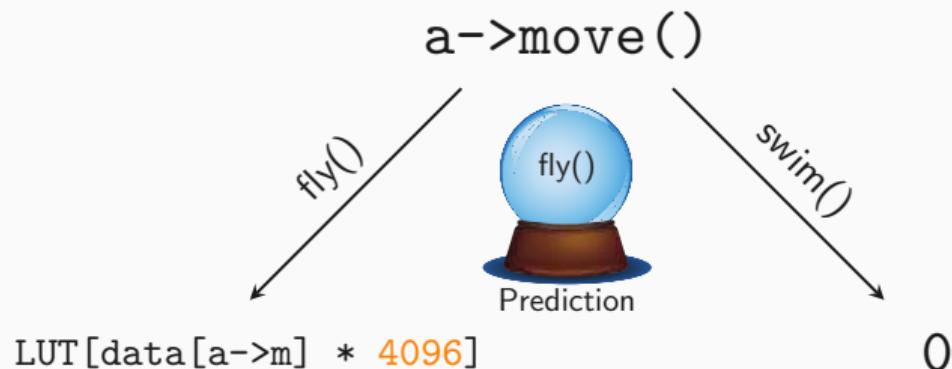
```
Animal* a = bird;
```



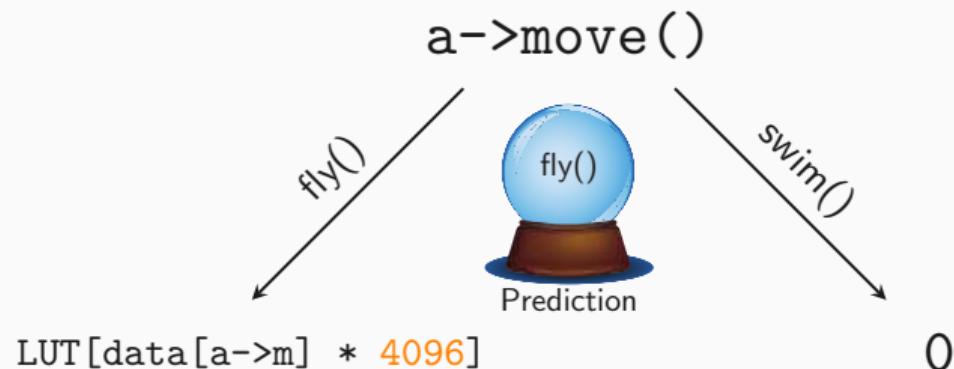
```
Animal* a = bird;
```



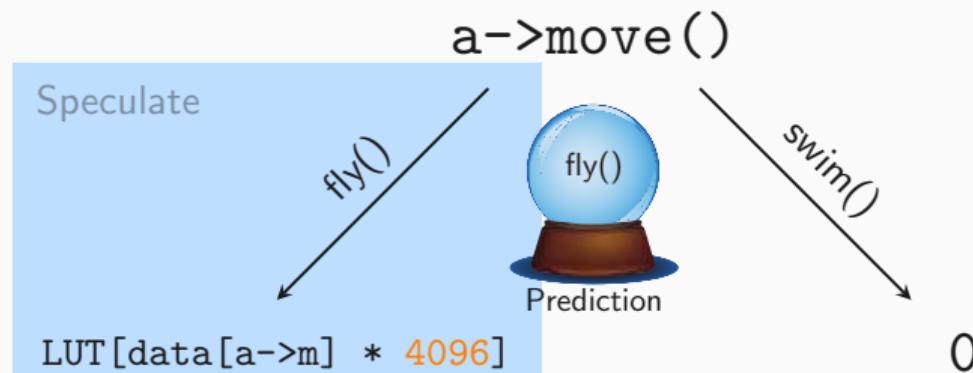
```
Animal* a = bird;
```



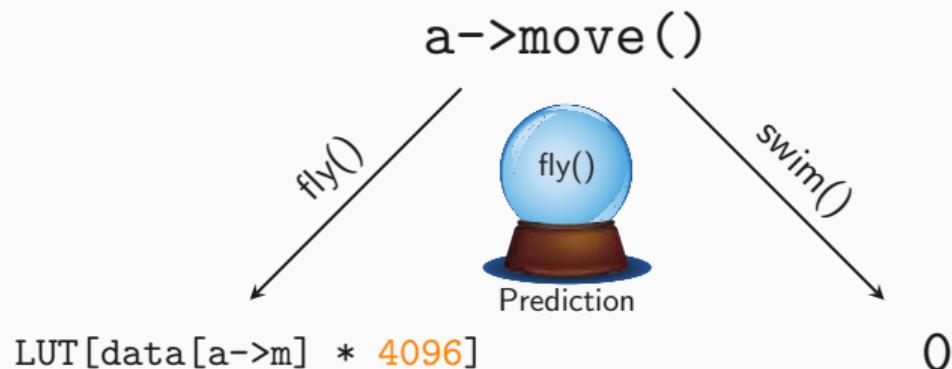
```
Animal* a = fish;
```



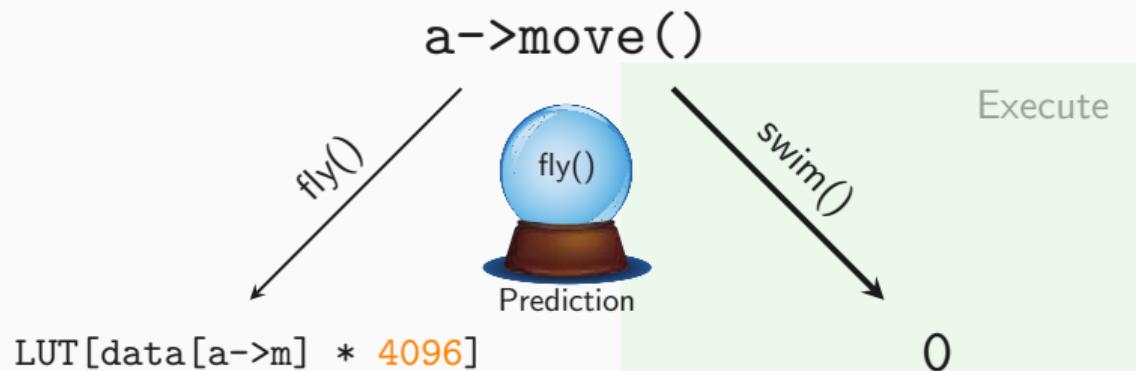
```
Animal* a = fish;
```



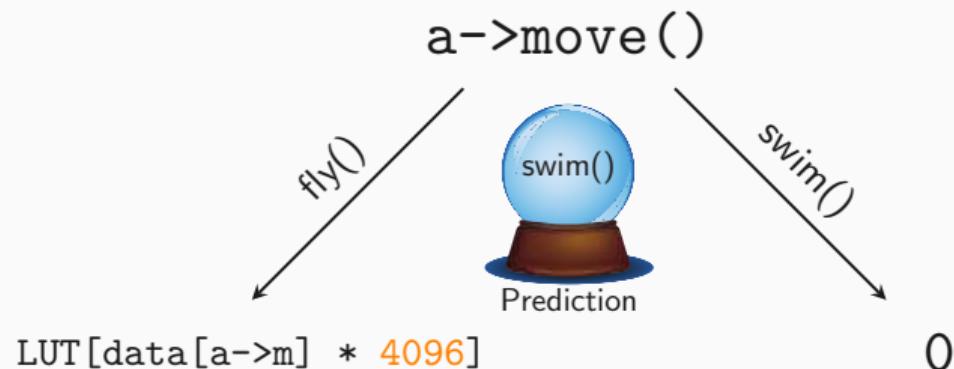
```
Animal* a = fish;
```

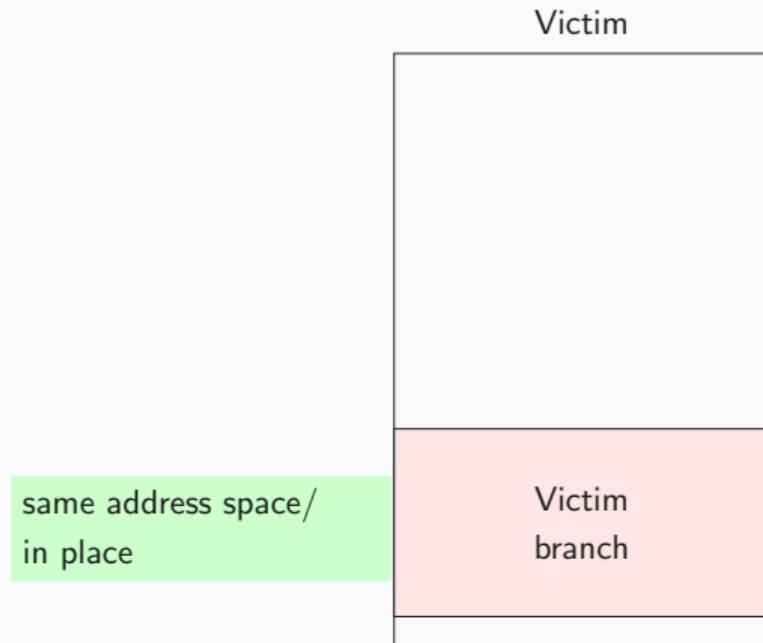


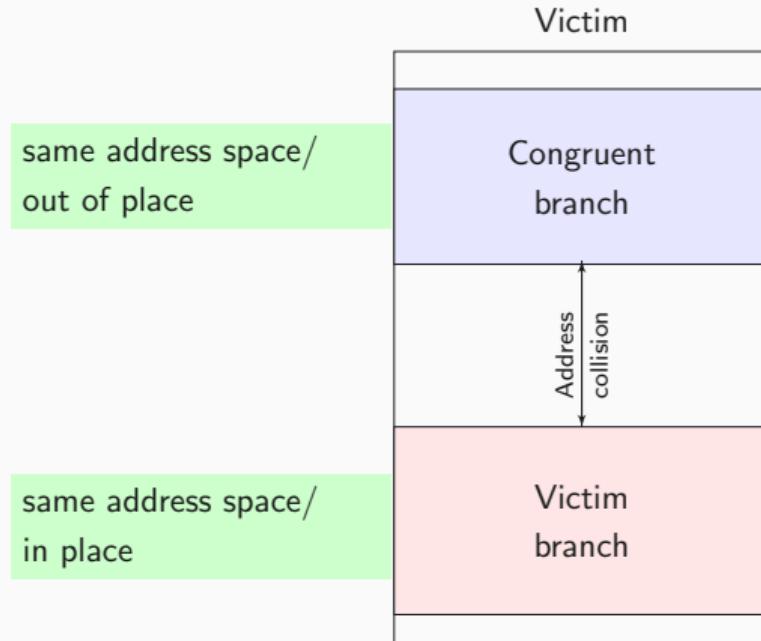
```
Animal* a = fish;
```

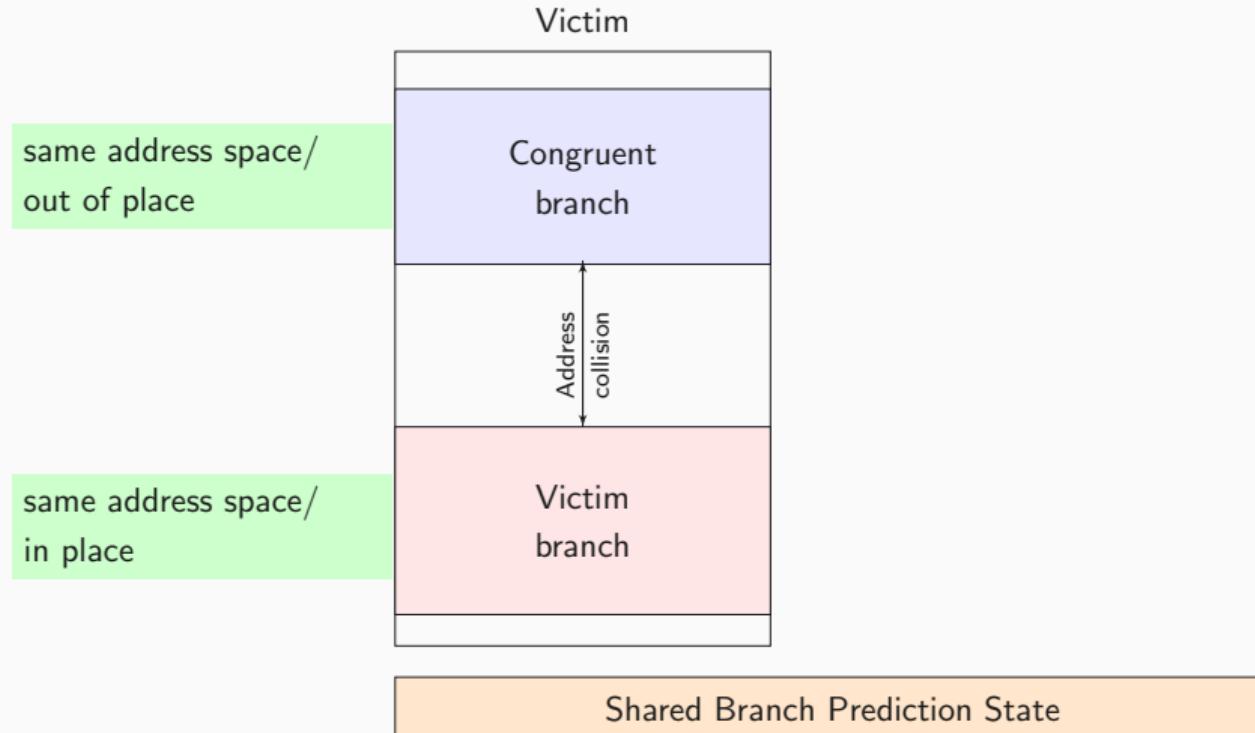


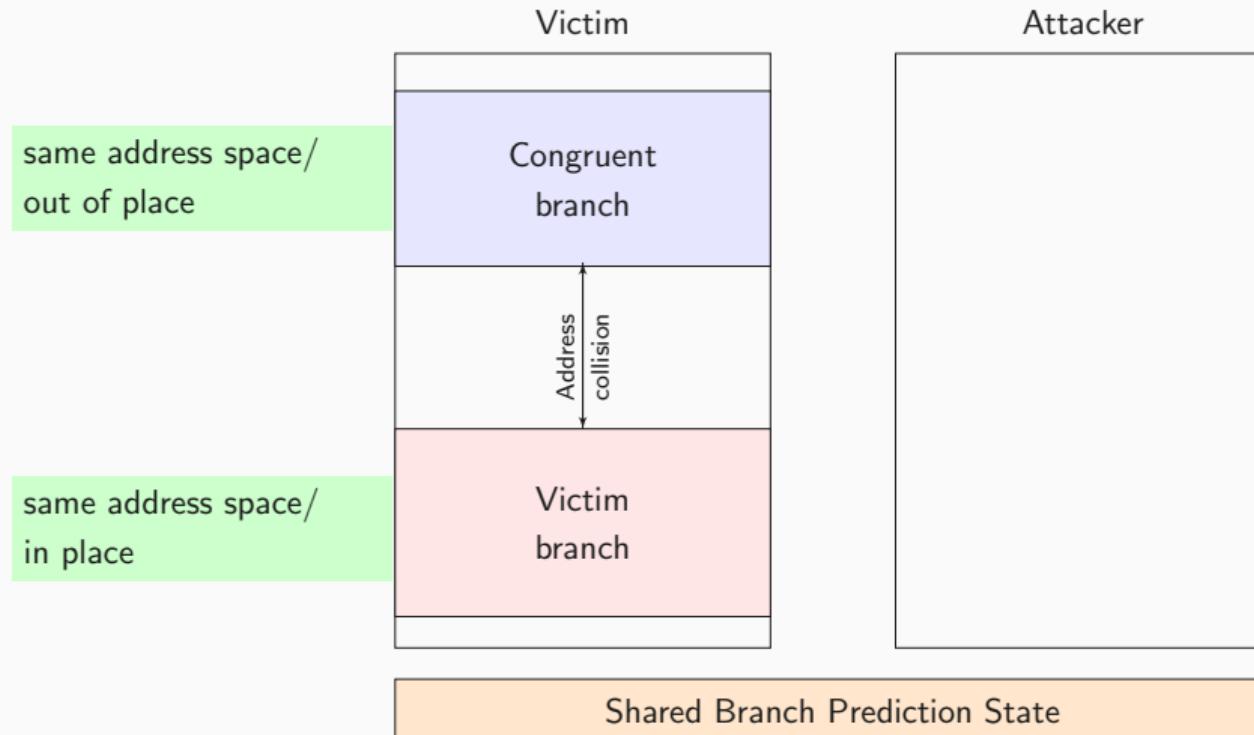
```
Animal* a = fish;
```

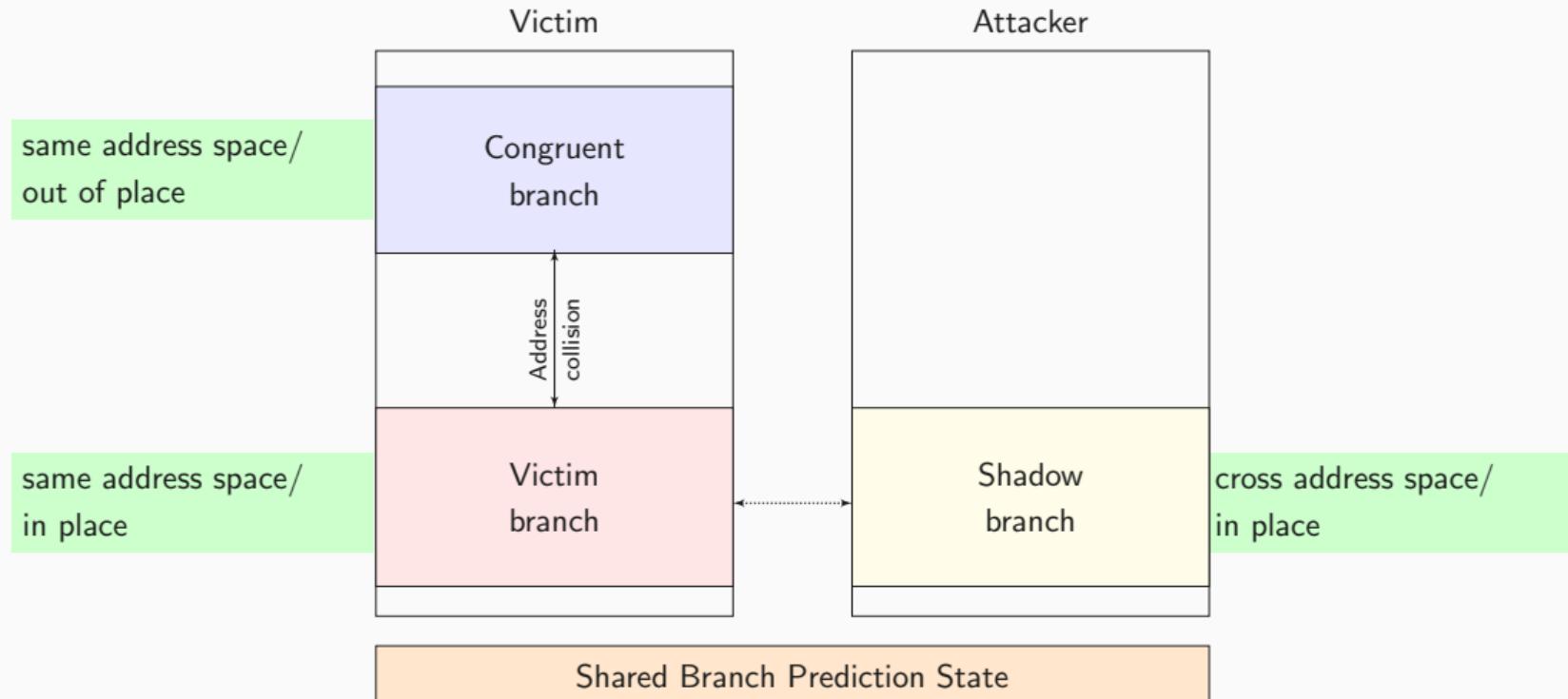


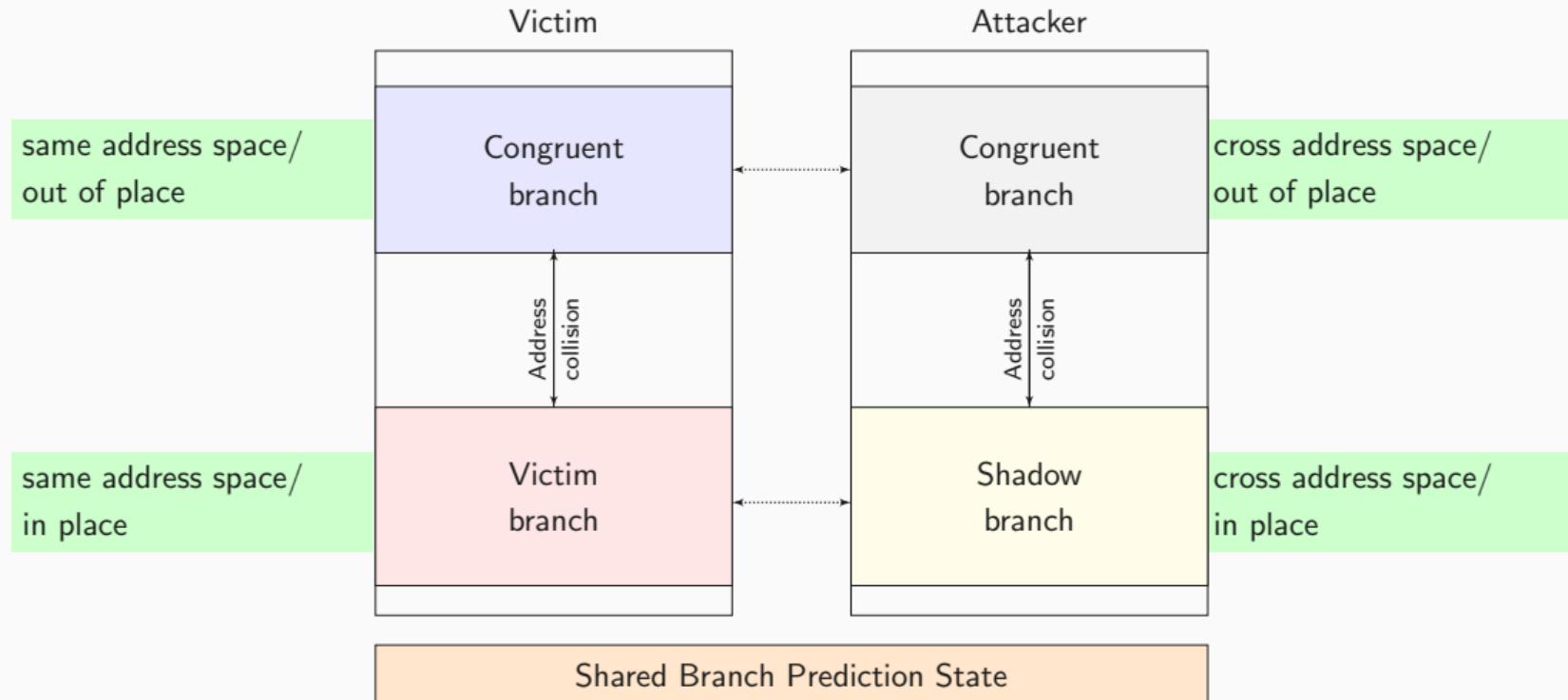




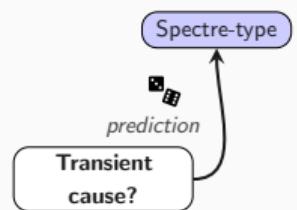


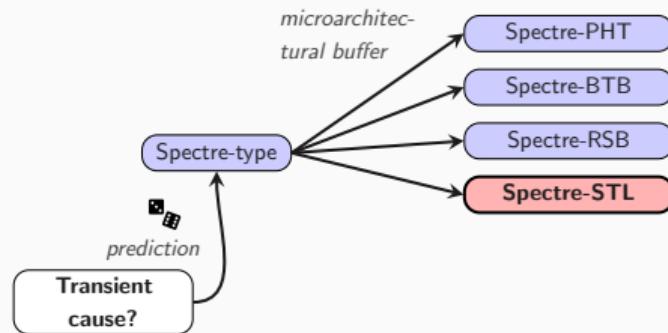


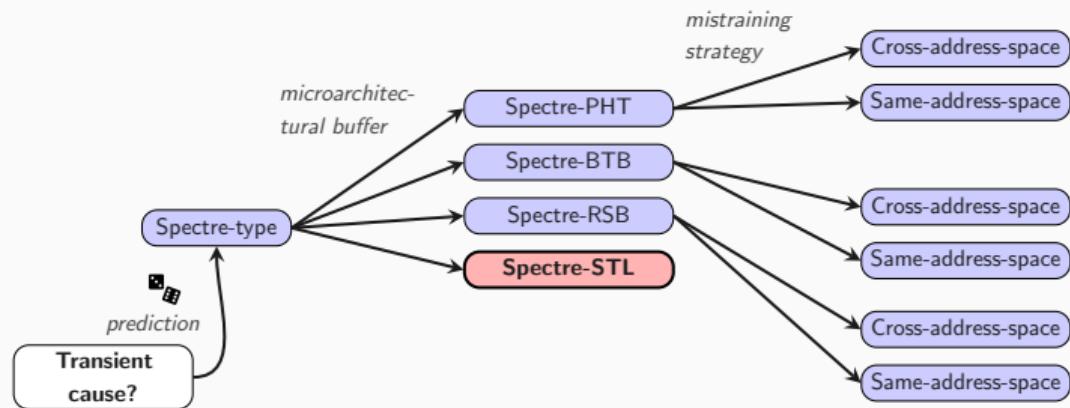


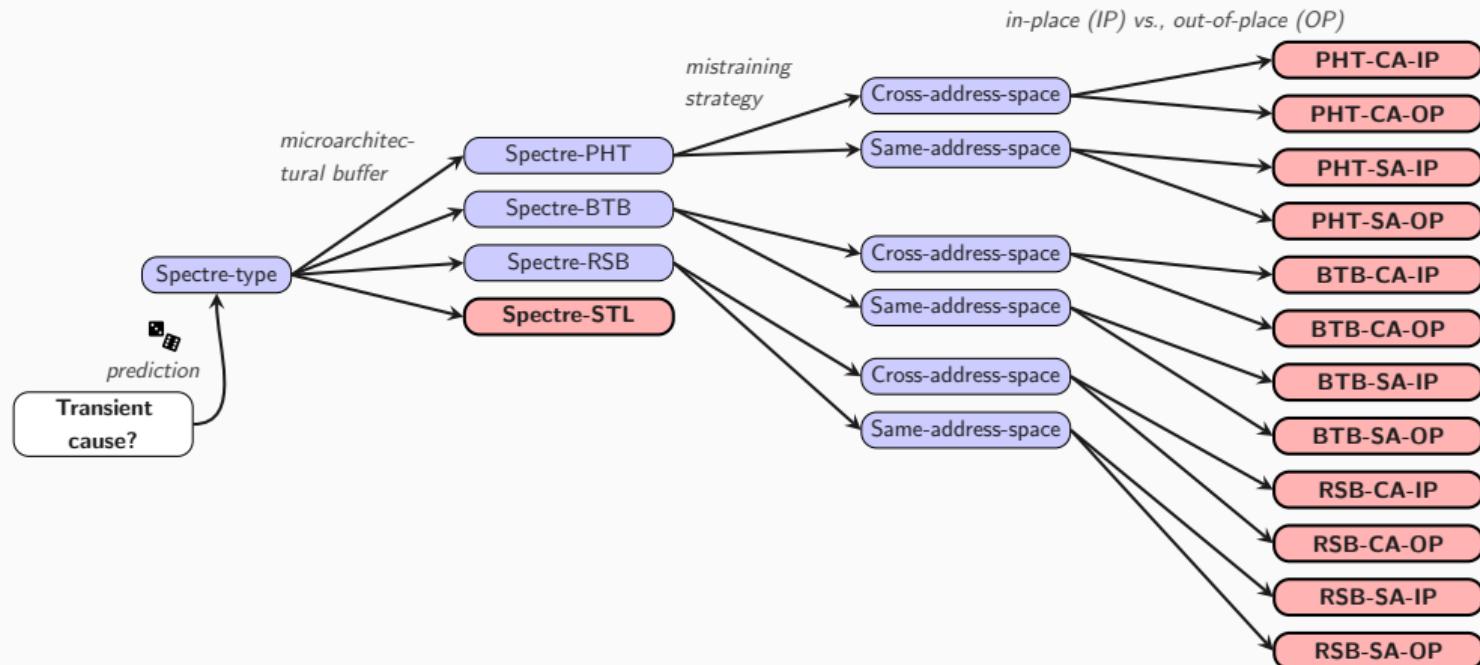


Transient
cause?





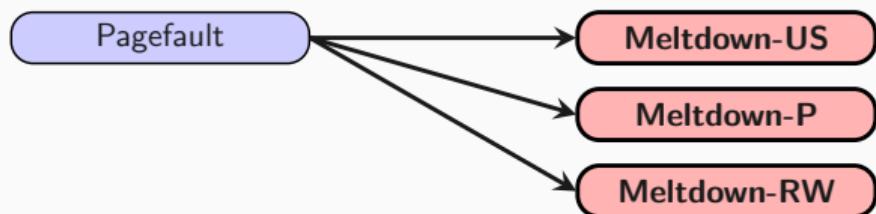


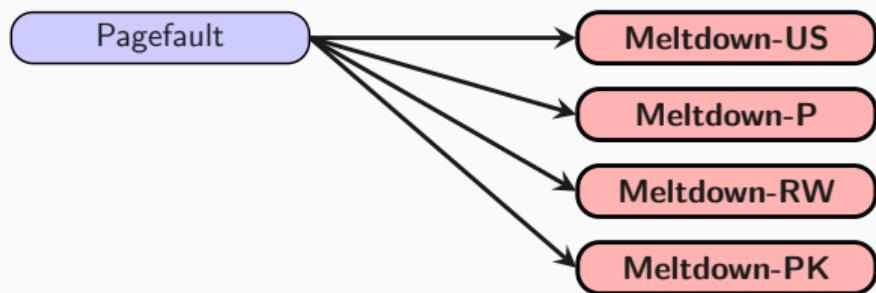


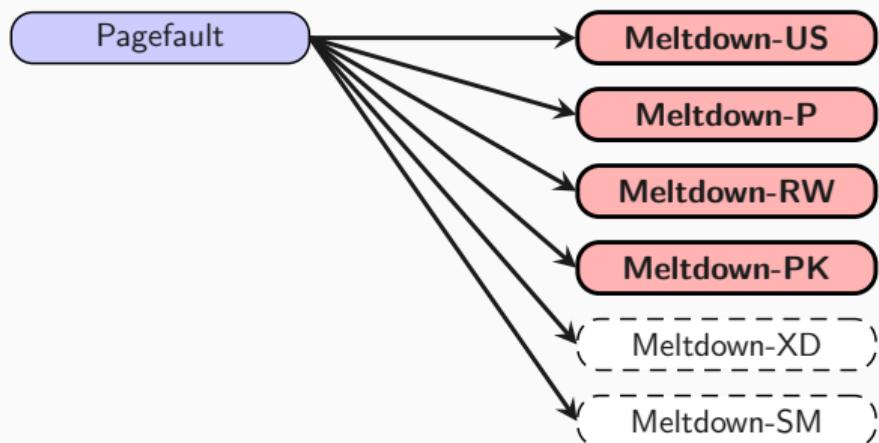
Pagefault



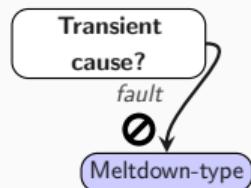


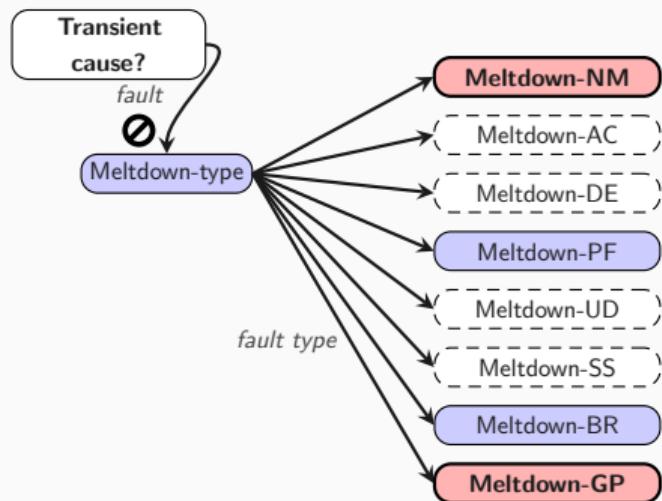


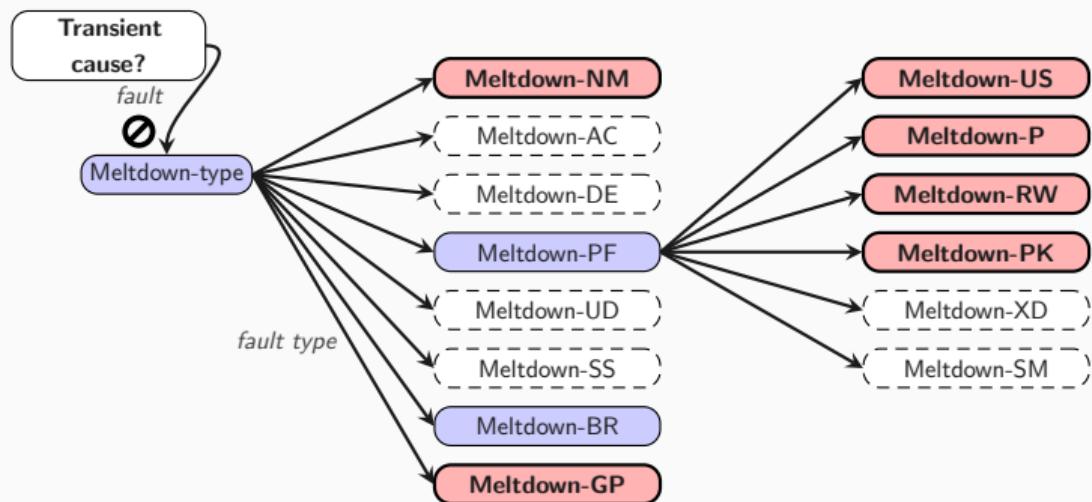


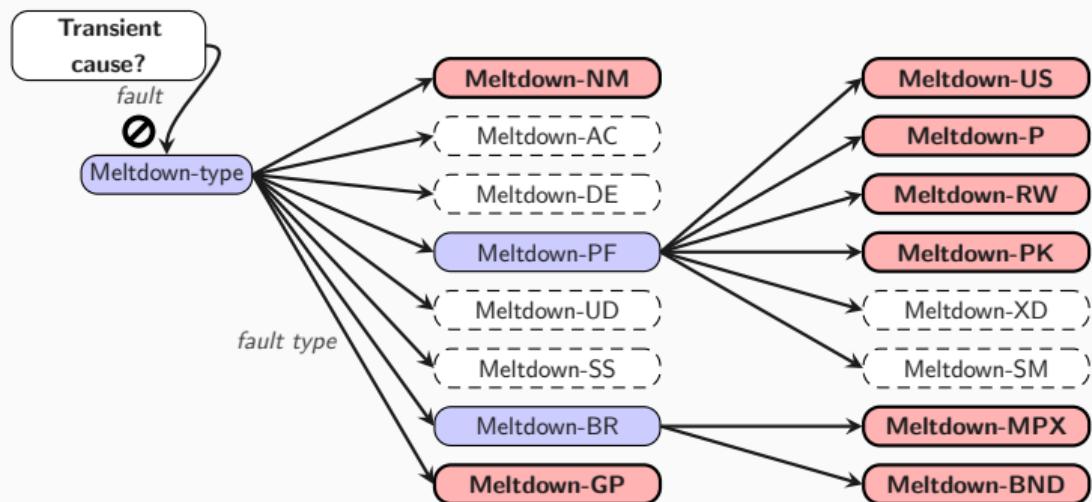


Transient
cause?









Mitigations?





Computer Architecture Today

Informing the broad computing community about current activities, advances and future directions in computer architecture.

Let's Keep it to Ourselves: Don't Disclose Vulnerabilities

by Gus Uht on Jan 31, 2019 | Tags: Opinion, Security



CONTRIBUTE

Editor: Alvin R. Lebeck

Associate Editor: Vijay Janapa Reddi

[Contribute to Computer
Architecture Today](#)

Table 1: Spectre-type defenses and what they mitigate.

	Defense	InvisISpec	SafeSpec	DAM/G	RSB Stuffing	Retpoline	Poison Value	Index Masking	Site Isolation	SLH	YSNB	IBRS	STPB	IBPB	Serialization	Taint Tracking	Slosh Reduction	SSBD/SSBB
Attack \ Defense																		
Intel	Spectre-PHT	□	□	□	◊	◊	●	○	●	○	◊	◊	●	■	●	□	◊	
	Spectre-BTB	□	□	□	◊	●	◊	◊	○	◊	◊	●	●	●	◊	■	●	◊
	Spectre-RSB	□	□	□	◊	●	◊	◊	◊	○	◊	◊	◊	◊	◊	■	●	◊
	Spectre-STL	□	□	□	◊	◊	◊	◊	●	◊	◊	◊	◊	◊	◊	■	●	●
ARM	Spectre-PHT	□	□	□	◊	◊	●	○	●	○	◊	◊	◊	●	■	●	□	◊
	Spectre-BTB	□	□	□	◊	●	◊	◊	○	◊	◊	◊	◊	◊	◊	■	●	◊
	Spectre-RSB	□	□	□	◊	◊	◊	◊	●	◊	◊	◊	◊	◊	◊	■	●	◊
	Spectre-STL	□	□	□	◊	◊	◊	◊	●	◊	◊	◊	◊	◊	◊	■	●	●
AMD	Spectre-PHT	□	□	□	◊	◊	●	○	●	○	◊	◊	◊	●	■	●	□	◊
	Spectre-BTB	□	□	□	◊	●	◊	◊	○	◊	◊	■	■	■	◊	■	●	◊
	Spectre-RSB	□	□	□	◊	◊	◊	◊	●	◊	◊	◊	◊	◊	◊	■	●	◊
	Spectre-STL	□	□	□	◊	◊	◊	◊	●	◊	◊	◊	◊	◊	◊	■	●	●

Symbols show if an attack is mitigated (●), partially mitigated (○), not mitigated (○), theoretically mitigated (■), theoretically impeded (□), not theoretically impeded (□), or out of scope (◊).

Table 2: Reported performance impacts of countermeasures

Defense \ Impact	Performance Loss	Benchmark
Defense		
InvisiSpec	22%	SPEC
SafeSpec	3% (improvement)	SPEC2017 on MARSSx86
DAWG	2–12%, 1–15%	PARSEC, GAPBS
RSB Stuffing	no reports	
Retpoline	5–10%	real-world workload servers
Site Isolation	only memory overhead	
SLH	36.4%, 29%	Google microbenchmark suite
YSNB	60%	Phoenix
IBRS	20–30%	two sysbench 1.0.11 benchmarks
STIPB	30– 50%	Rodinia OpenMP, DaCapo
IBPB	no individual reports	
Serialization	62%, 74.8%	Google microbenchmark suite
SSBD/SSBB	2–8%	SYStmark®2014 SE & SPEC integer
KAISER/KPTI	0–2.6%	system call rates
L1TF mitigations	-3–31%	various SPEC



FINALLY THE RIGHT SPEED FOR ME

How to find the next big thing ;)

they become the target of one anti-masker's possessed creation, Arnaudine.

Director: David F. Sandberg | Stars: Anthony LaPaglia, Samara Lee, Miranda Otto, Brad Greenquist

Votes: 92,806 | Gross: \$102.09M



29. **Zombieland: Double Tap** (2019)



Action, Comedy, Horror | Post-production

Columbus, Tallahassee, Wichita, and Little Rock move to the American heartland as they face off against evolved zombies, fellow survivors, and the growing pains of the snarky makeshift family.

Director: Ruben Fleischer | Stars: Emma Stone, Zoey Deutch, Woody Harrelson, Abigail Breslin



30. **Love, Death & Robots** (2019-)



TV-MA | 15 min | Animation, Short, Comedy

8.7 Rate this

A collection of animated short stories that span various genres including science fiction, fantasy, horror and comedy.

Stars: Scott Whyte, Nolan North, Matthew Yang King, Michael Benyaer

Votes: 58,780



31. **iZombie** (2015-)



TV-14 | 42 min | Comedy, Crime, Drama

7.9 Rate this

A medical resident finds that being a zombie has its perks, which she uses to assist the police.

Stars: Rose McIver, Malcolm Goodwin, Rahul Kohli, Robert Buckley

Votes: 54,215

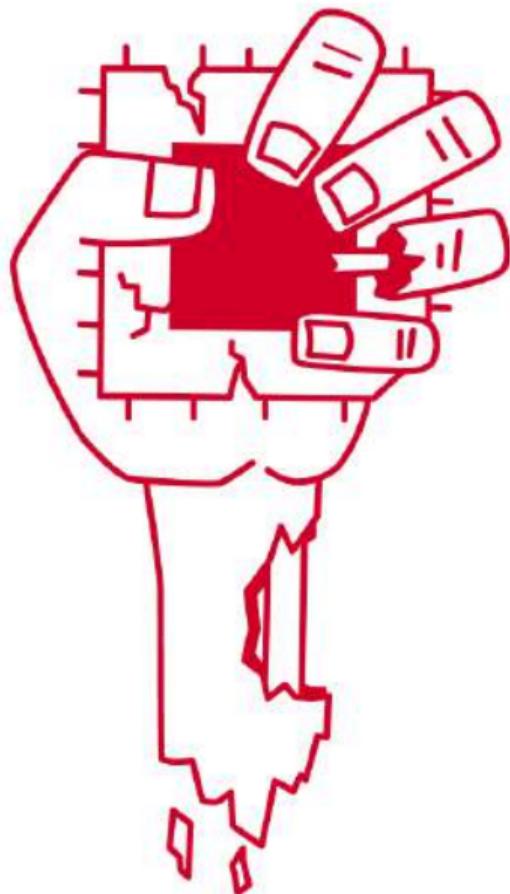
WOODY
HARRELSON JESSE
EISENBERG EMMA
STONE BRIGITTE
BRESLIN

ZOMBIELAND

NUT UP OR SHUT UP



IN THEATERS OCTOBER 9
Follow us @Zombieland on Twitter



ZOMBIELOAD ATTACK

The screenshot shows a web browser window with the URL <https://meltdownattack.com/meltdown>. The page content discusses the Meltdown attack, specifically focusing on the CPU's handling of kernel addresses during out-of-order execution. The browser interface includes standard controls like back, forward, and search, along with a zoom level of 150%.

When the kernel address is loaded in line 4, it is likely that the CPU already issued the subsequent instructions as part of the out-of-order execution, and that their corresponding μ OPs wait in the reservation station for the content of the kernel address to arrive. As soon as the



fault occurs load operation completed? "intel corp"



Alle

News

Bilder

Shopping

Videos

Mehr

Einstellungen

Tools

Ungefähr 111 000 Ergebnisse (0.42 Sekunden)

Toshiba Boot Error - TechRepublic

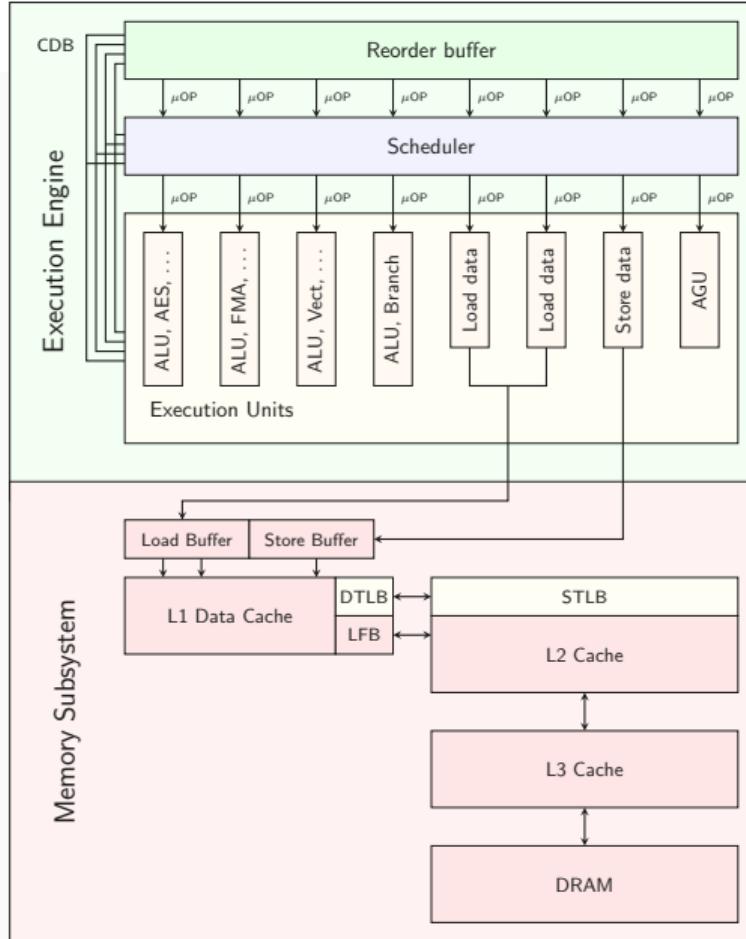
<https://www.techrepublic.com/.../toshiba-boot-error/> ▾ Diese Seite übersetzen

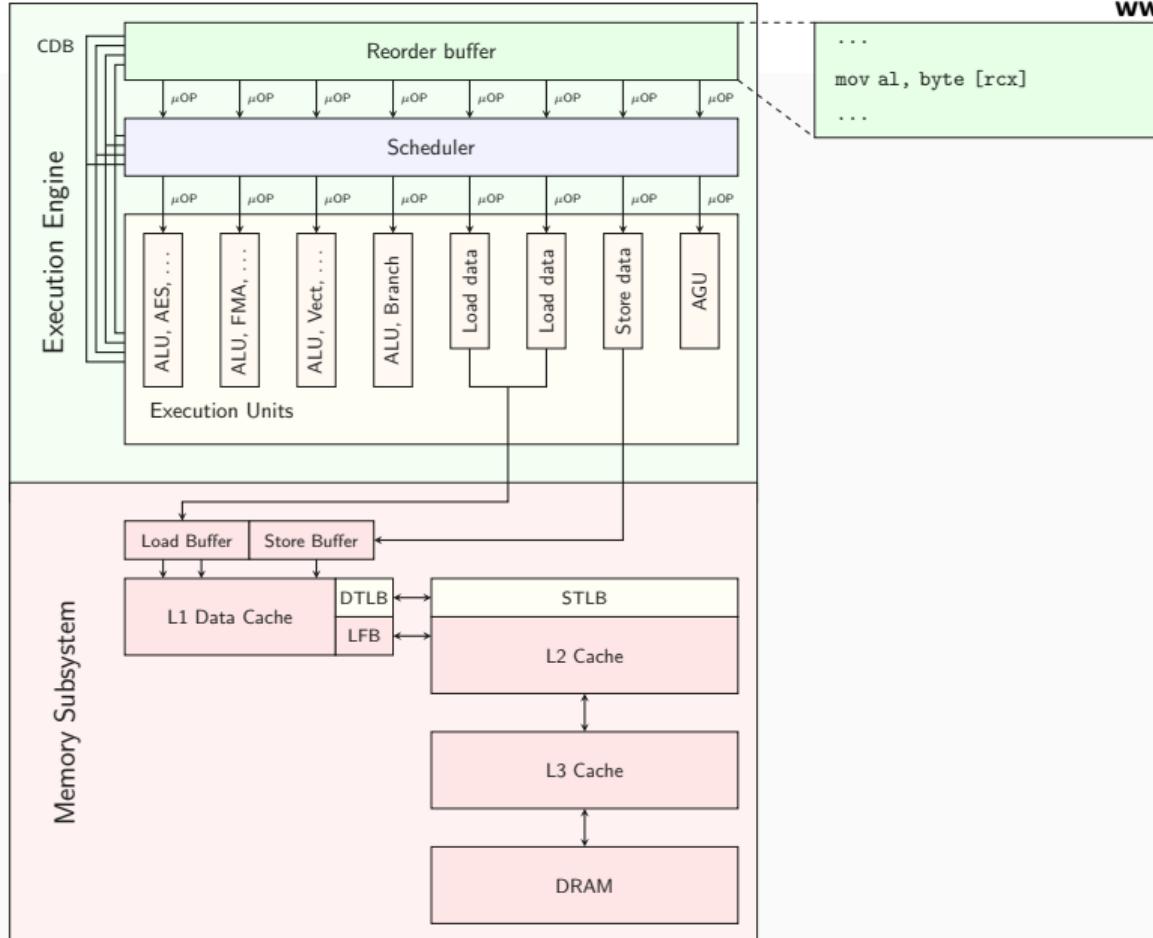
19.05.2007 - by CaptBilly1Eye · 12 years ago In reply to Toshiba Boot Error ... partition on the floppy disk, hard drive or a CD ROM to load the operating system. ... prior to this situation starting to occur, or if you find that the boot sequence already has the ... Leave the notebook plugged in and undisturbed until completed.

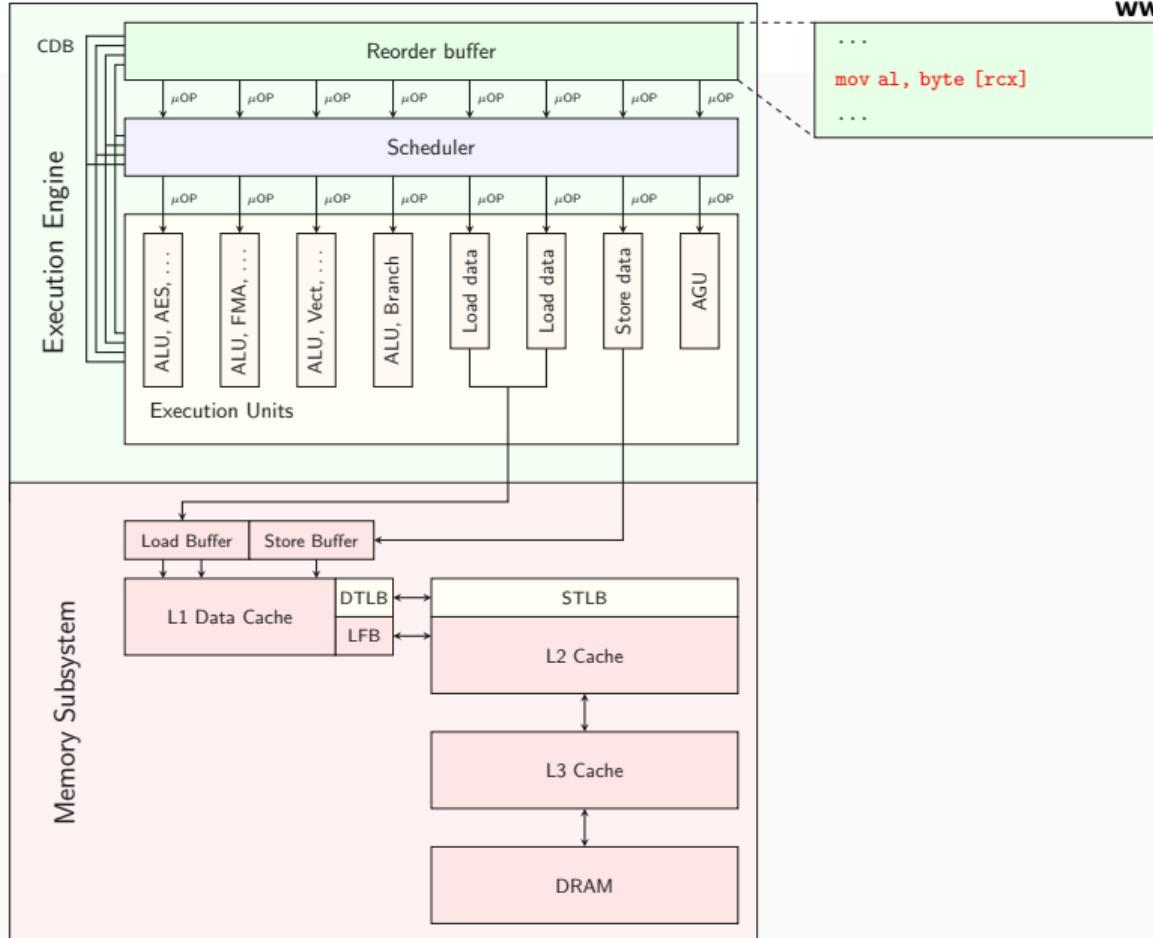
US5751983A - Out-of-order processor with a memory ...

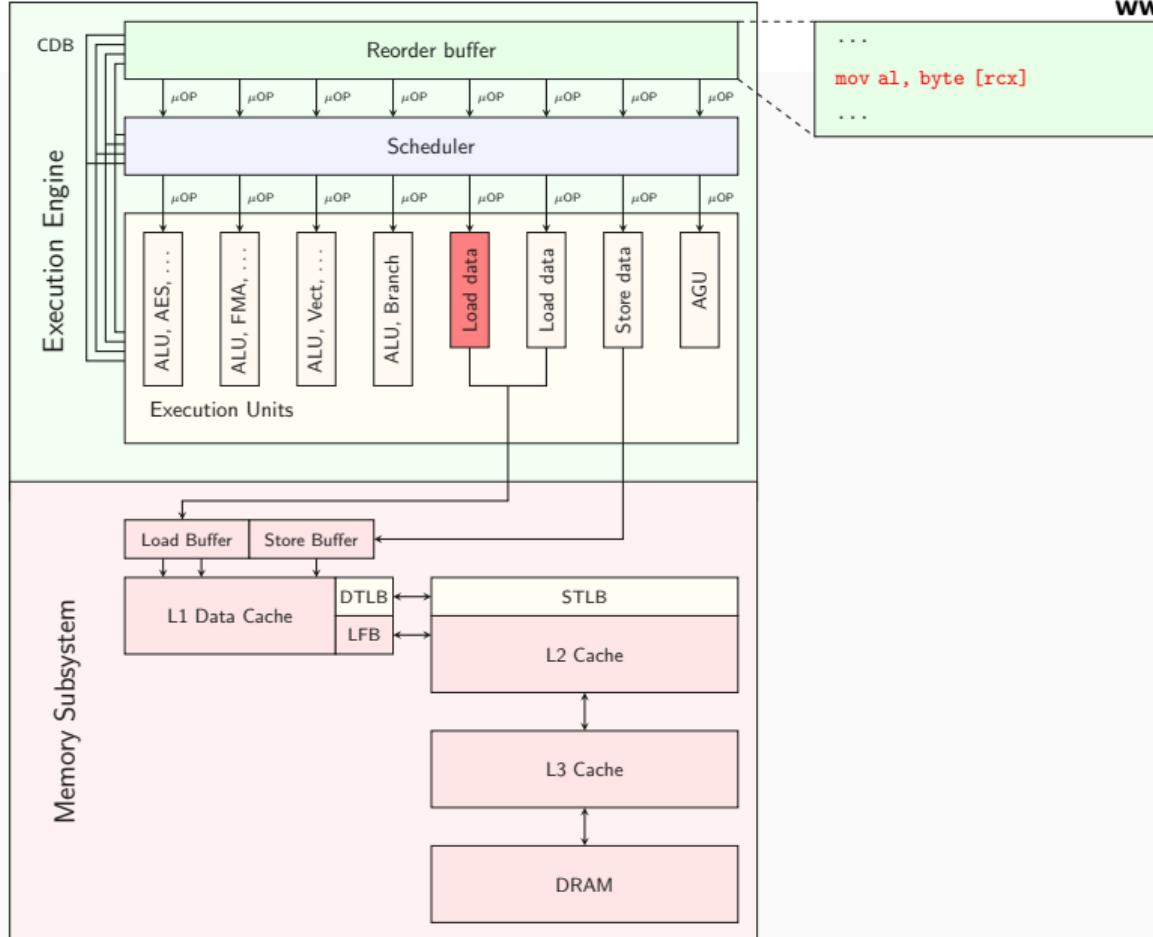
www.google.com/patents/US5751983 - Diese Seite übersetzen

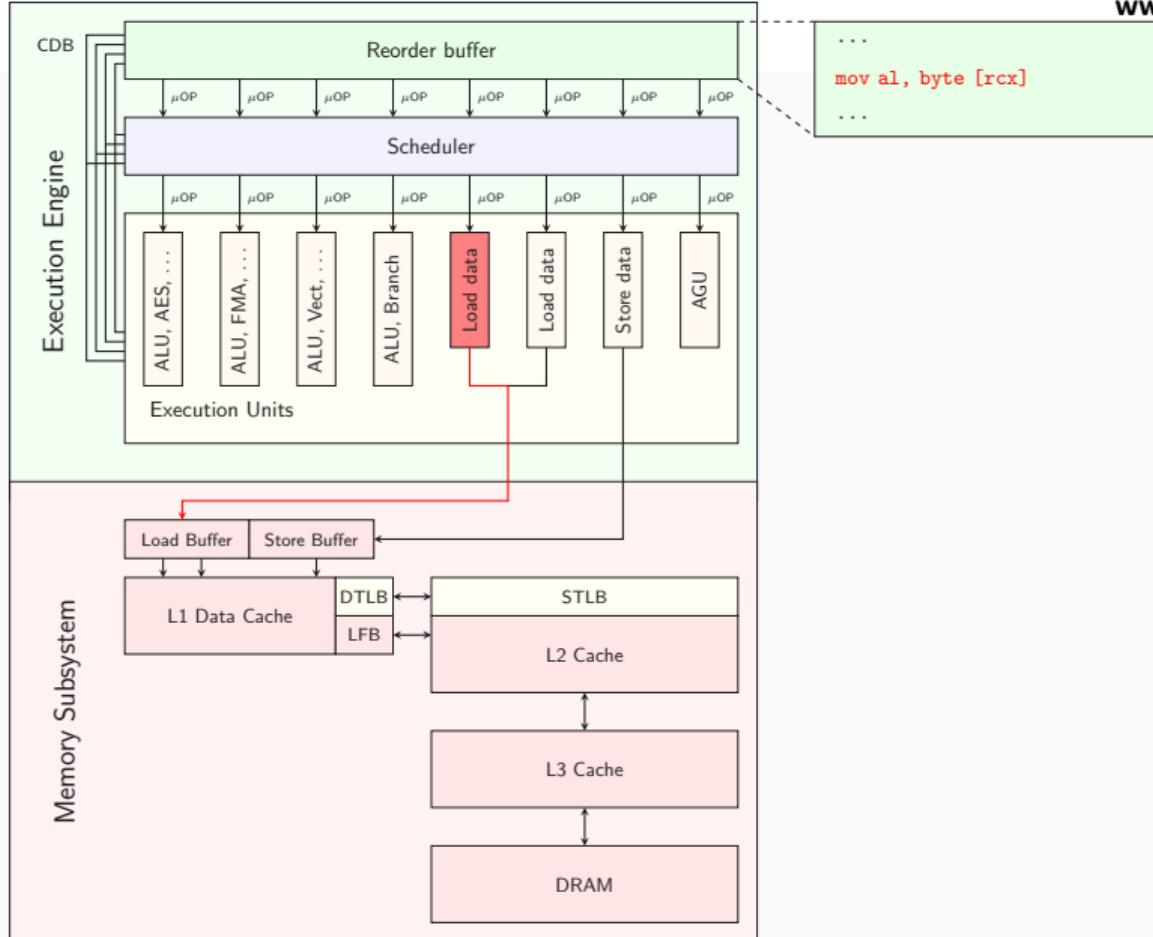
Application filed by Intel Corp ... Hence, a functional unit may often complete a first instruction (which logically precedes a second instruction in the If a fault occurs with respect to the LOAD operation, it is marked as valid and completed.

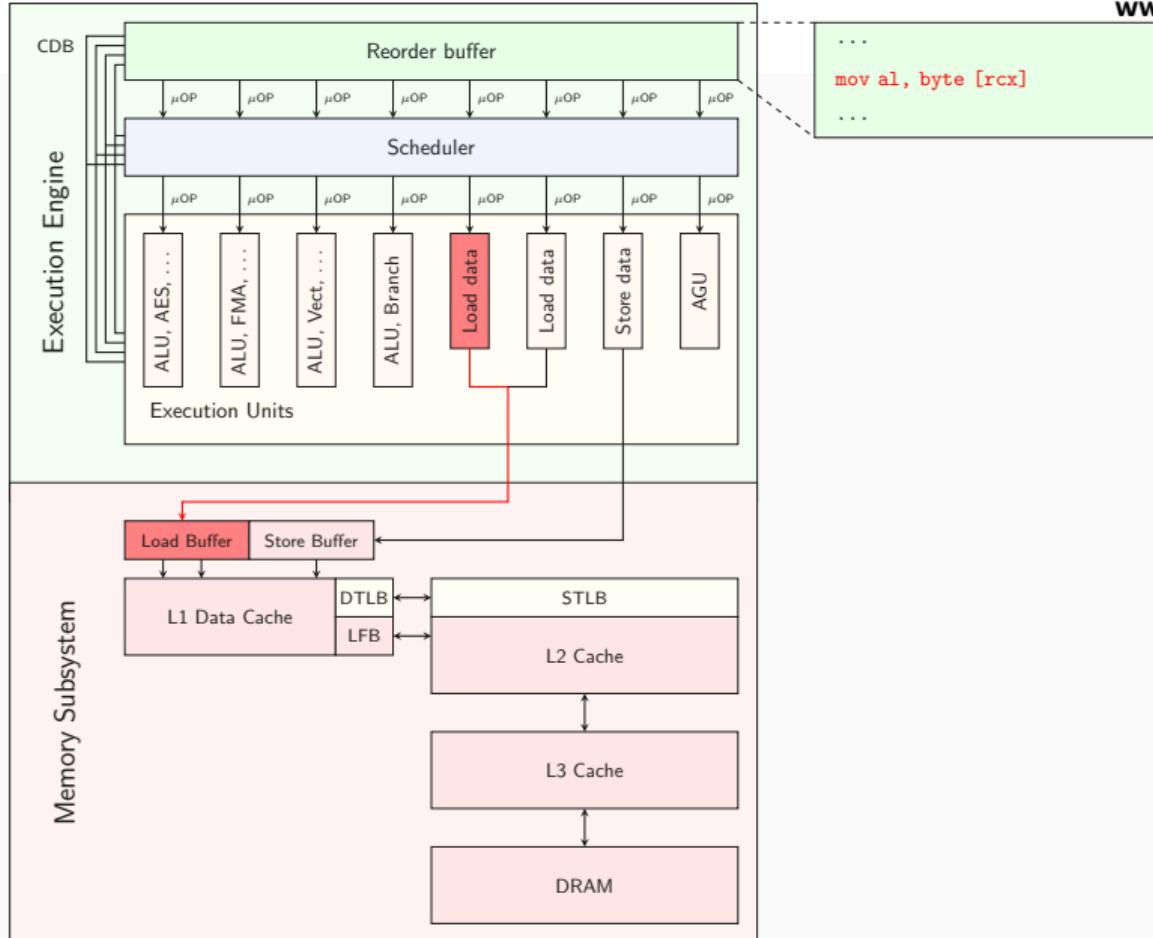


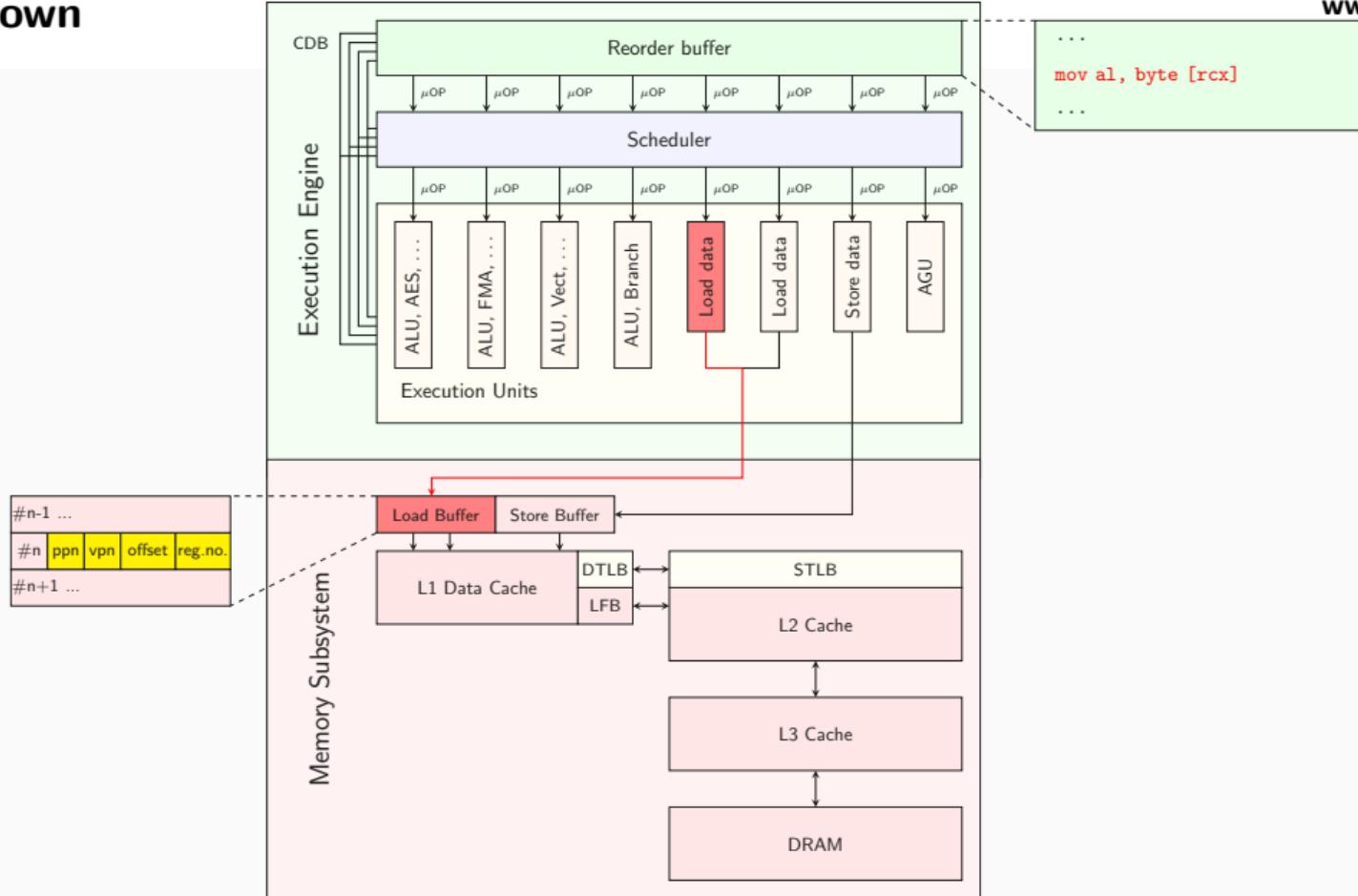


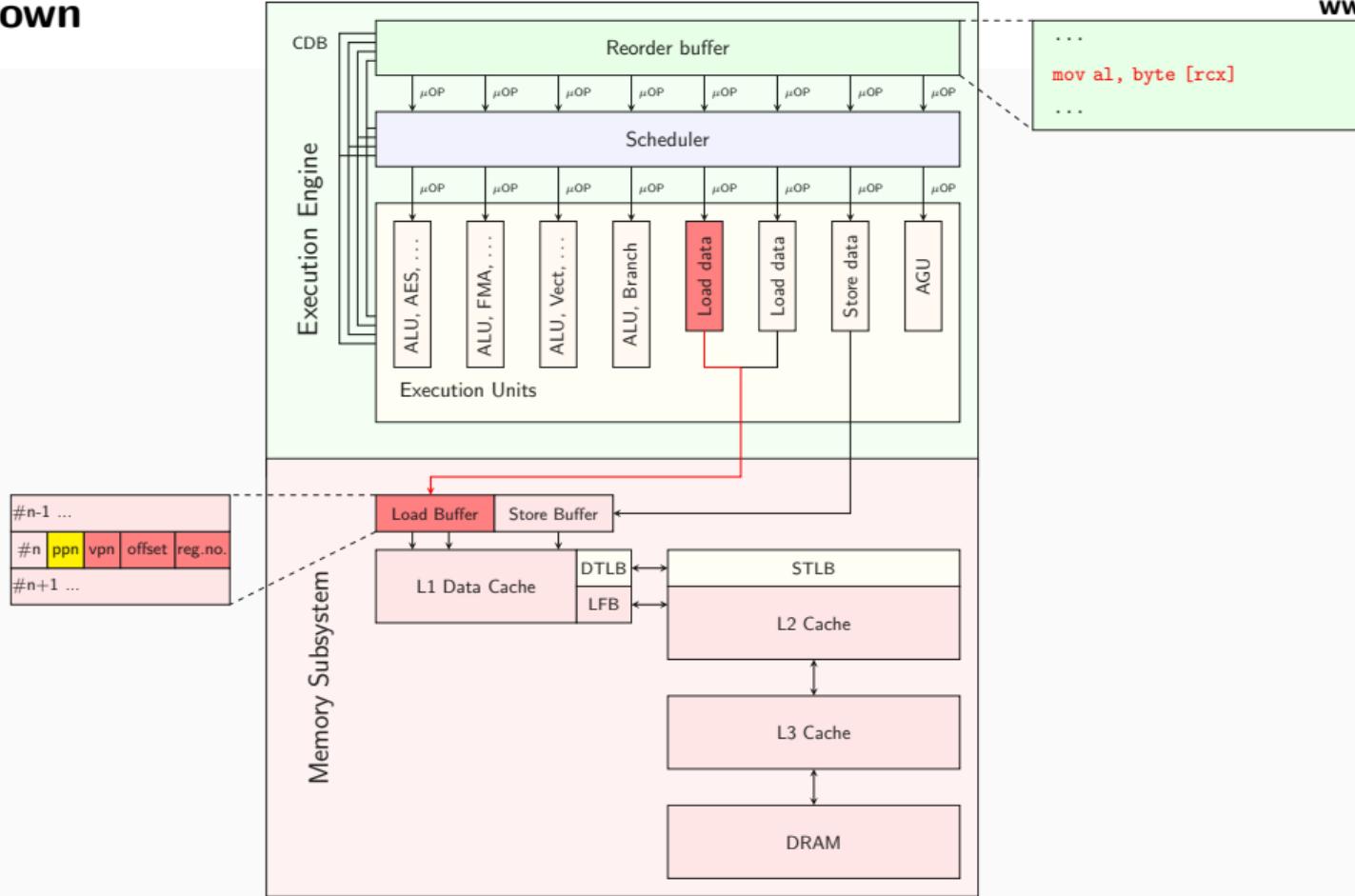


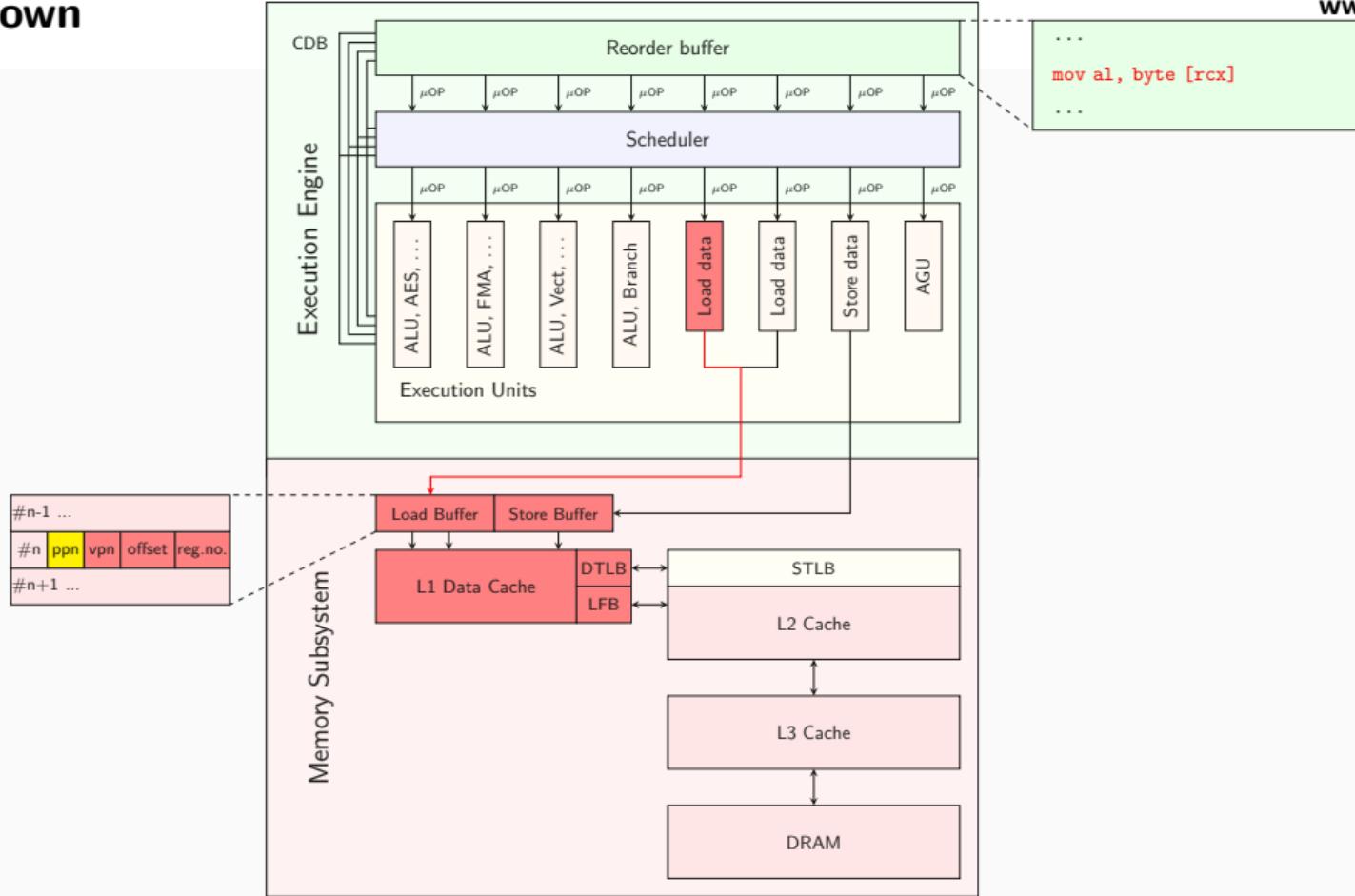


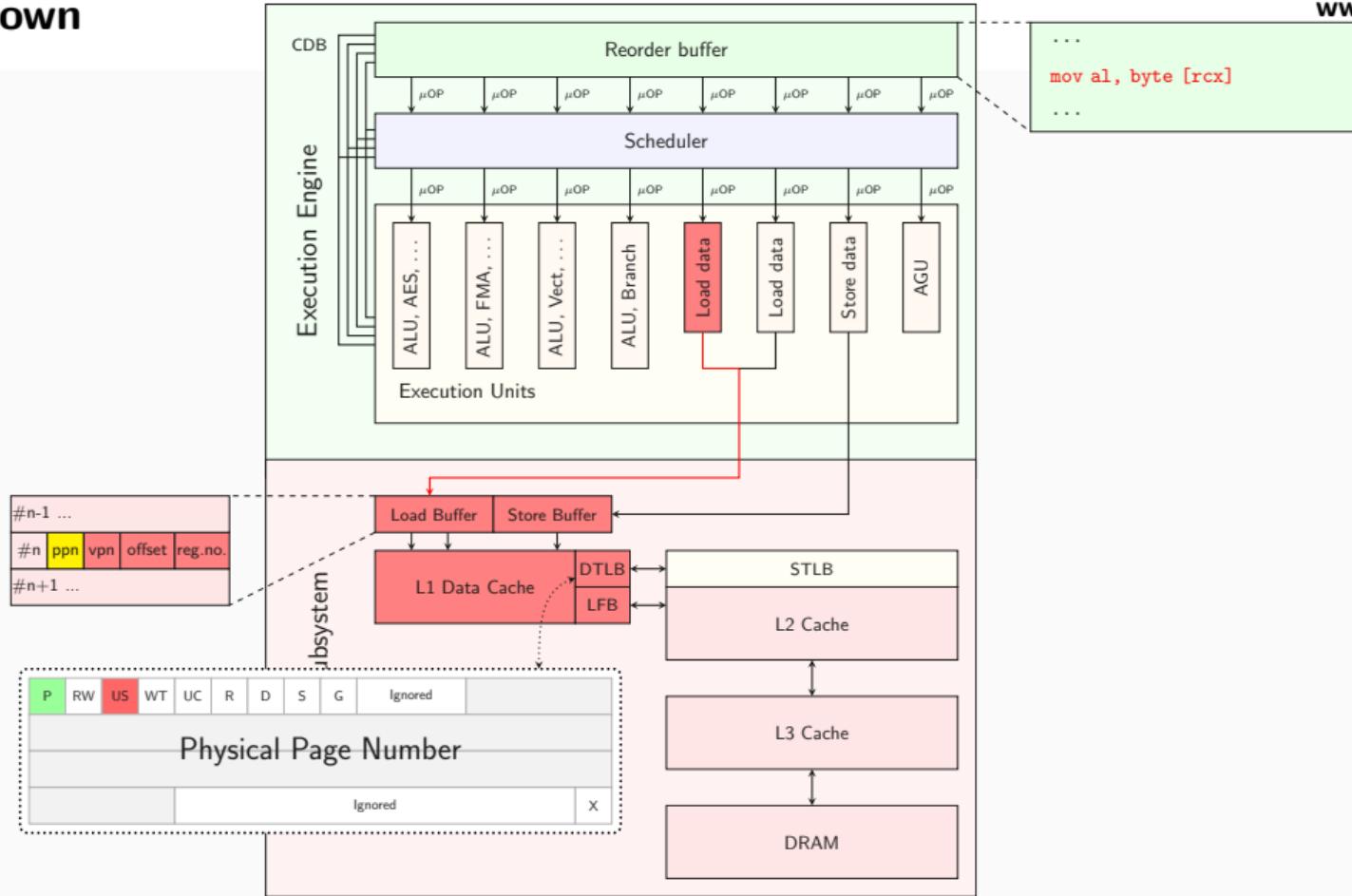


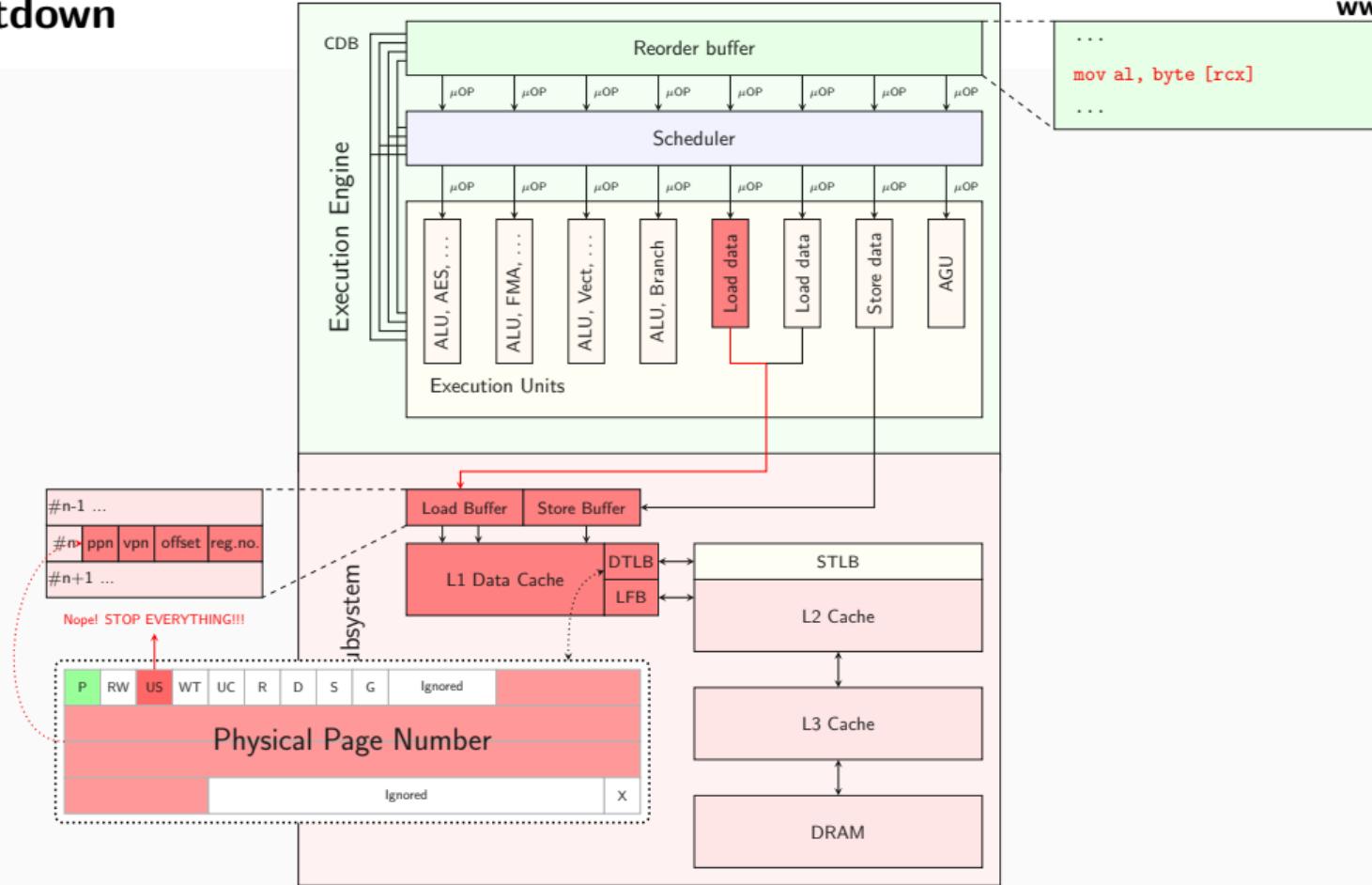


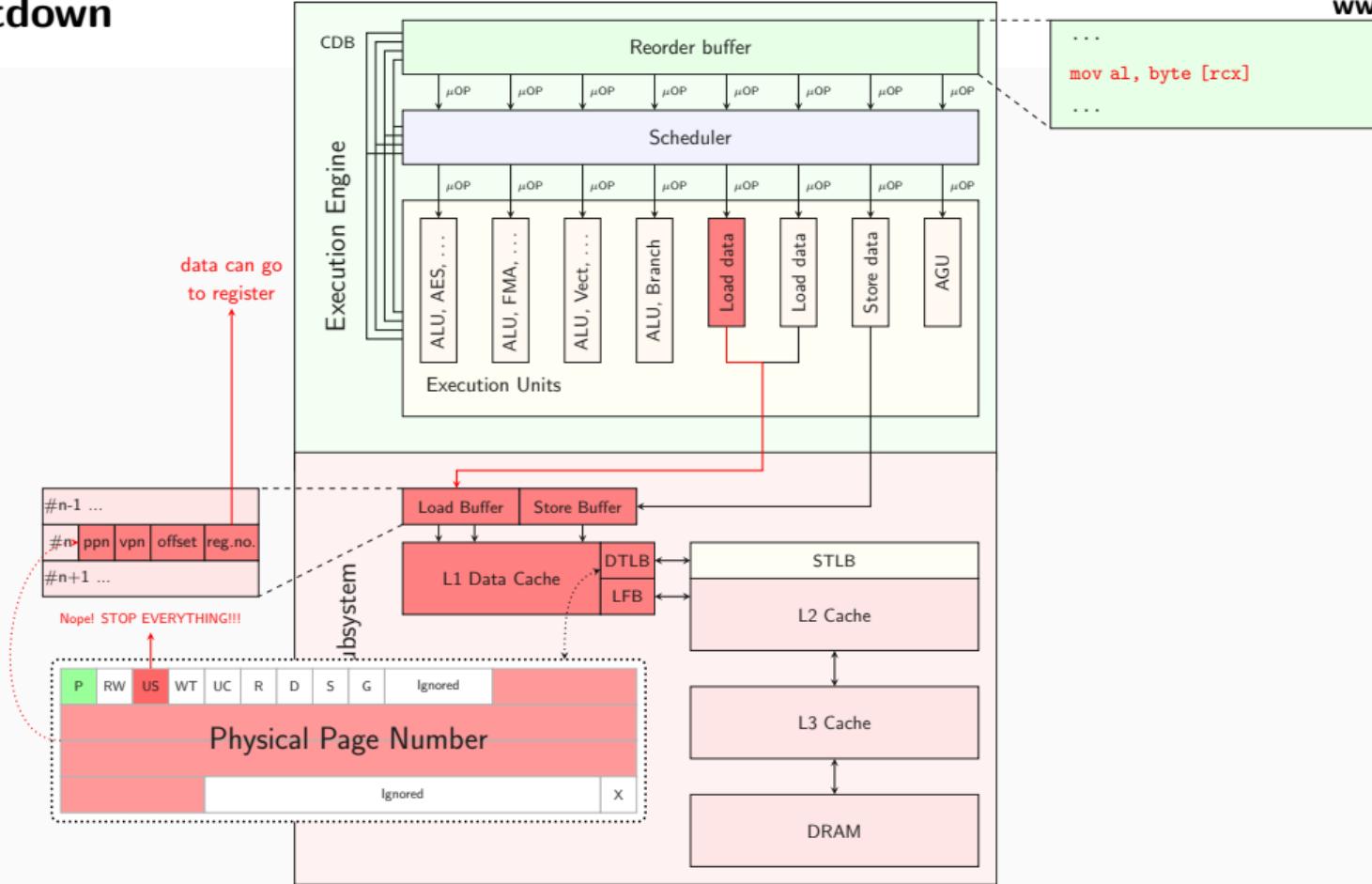


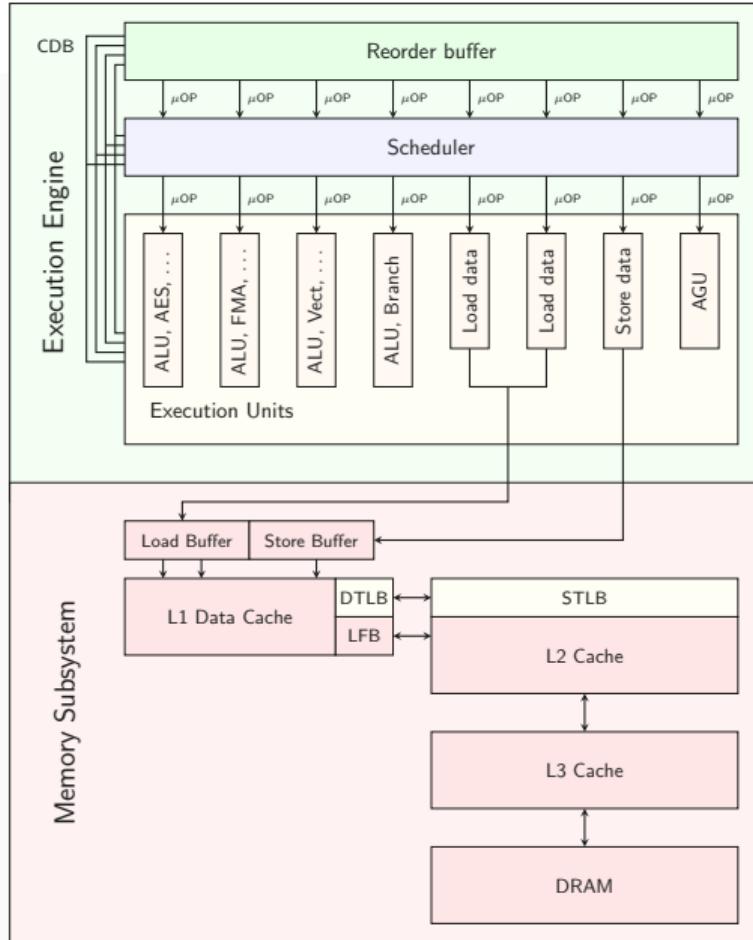


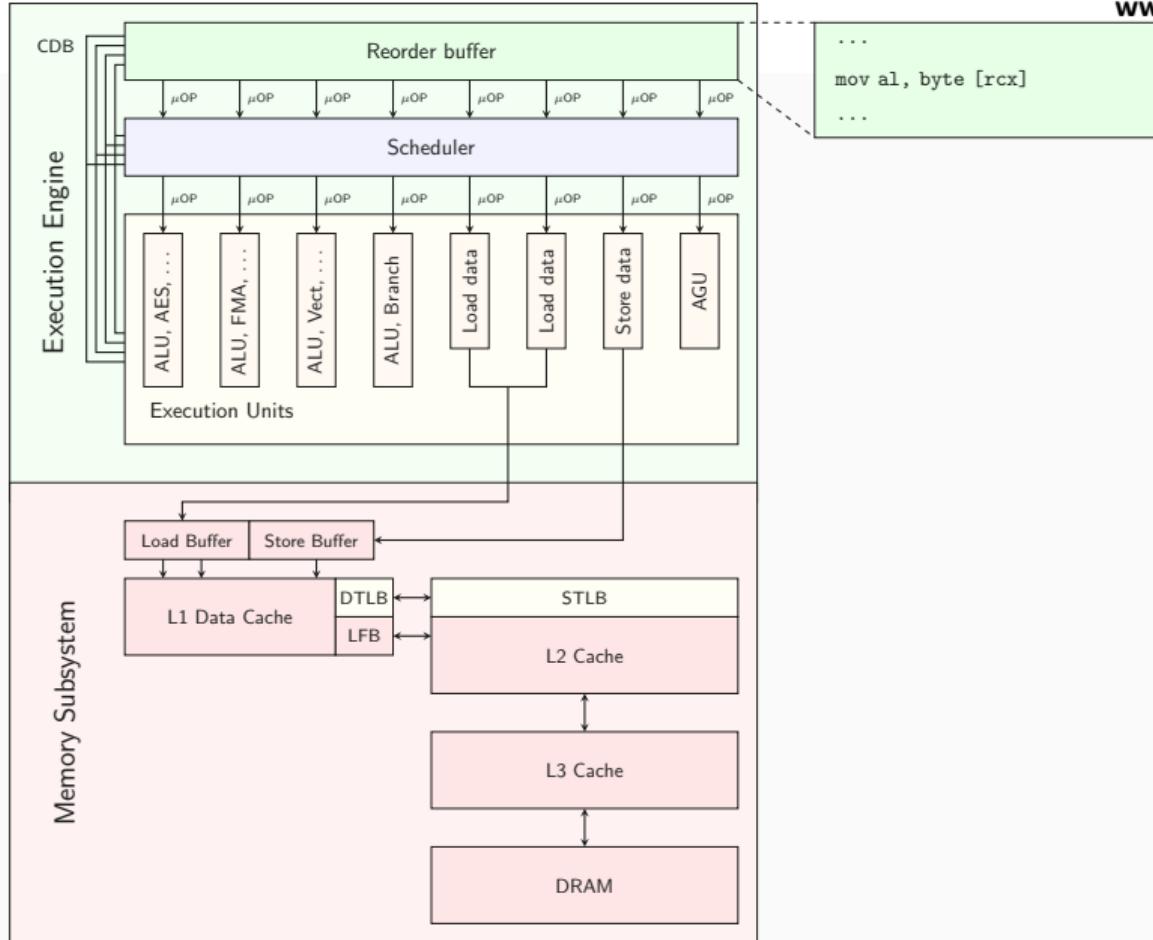


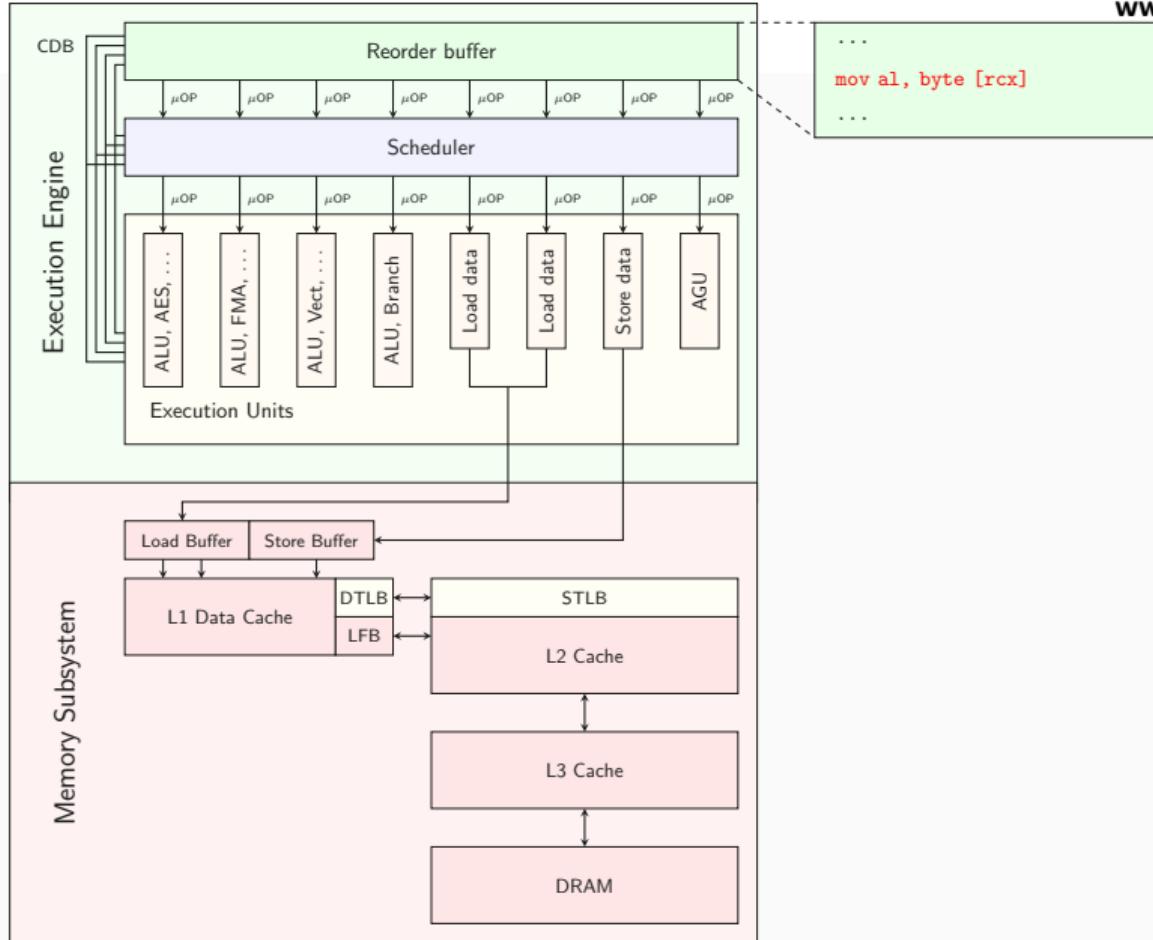


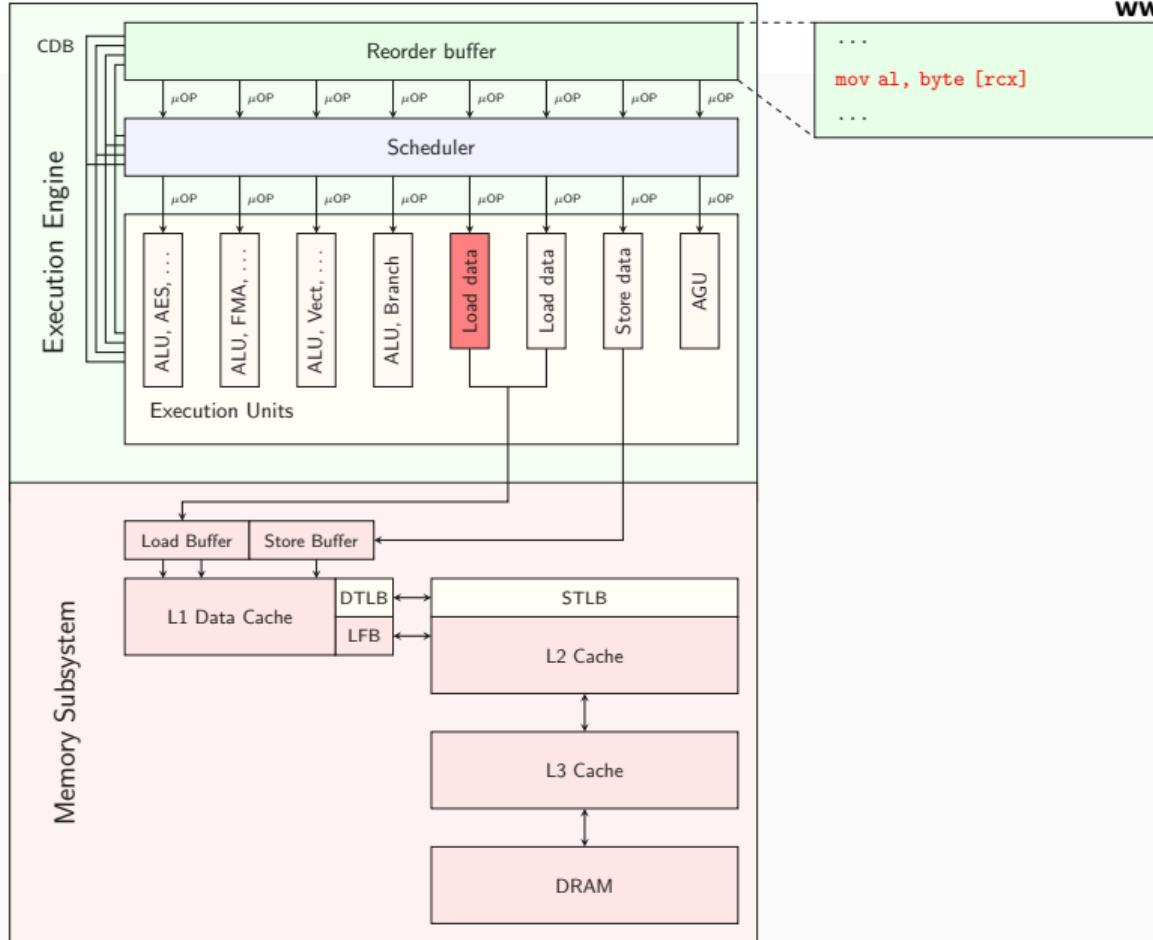


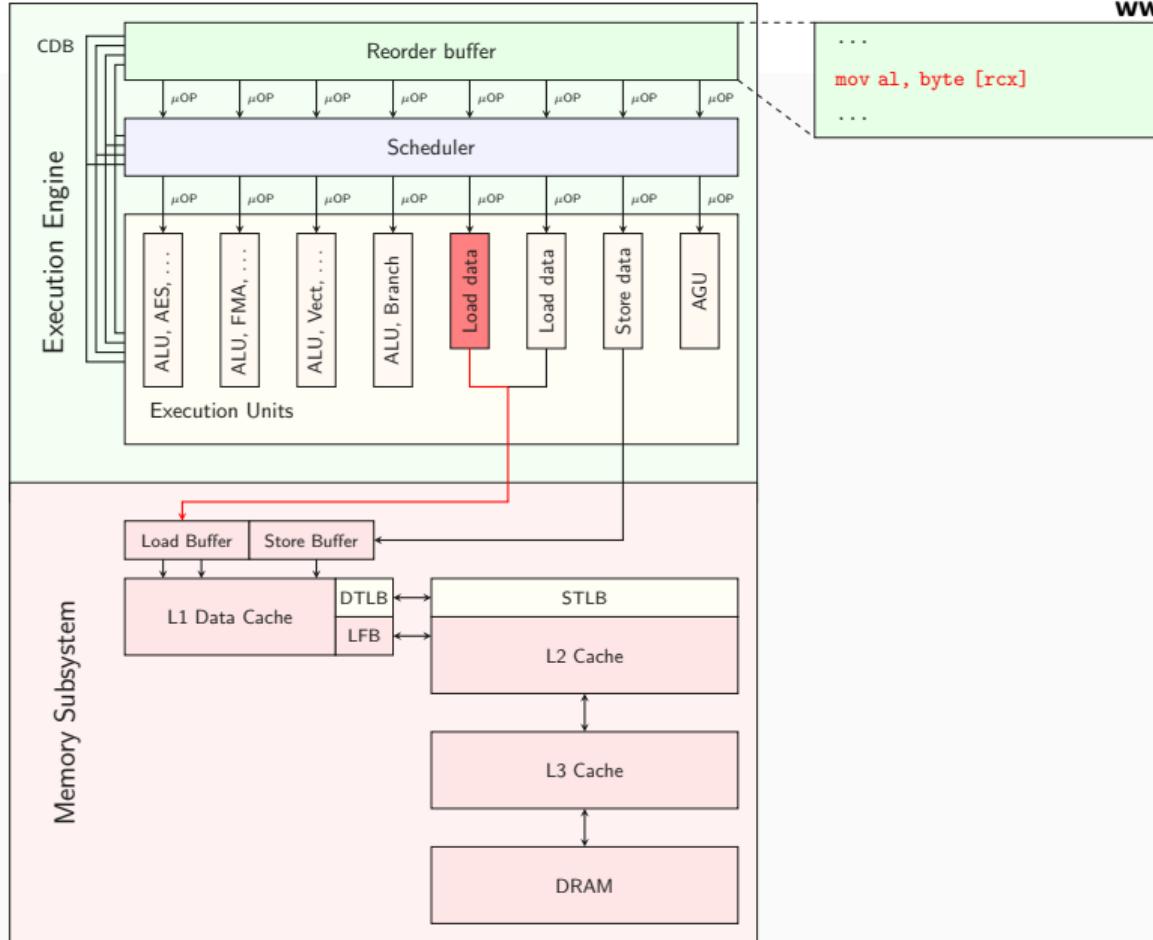


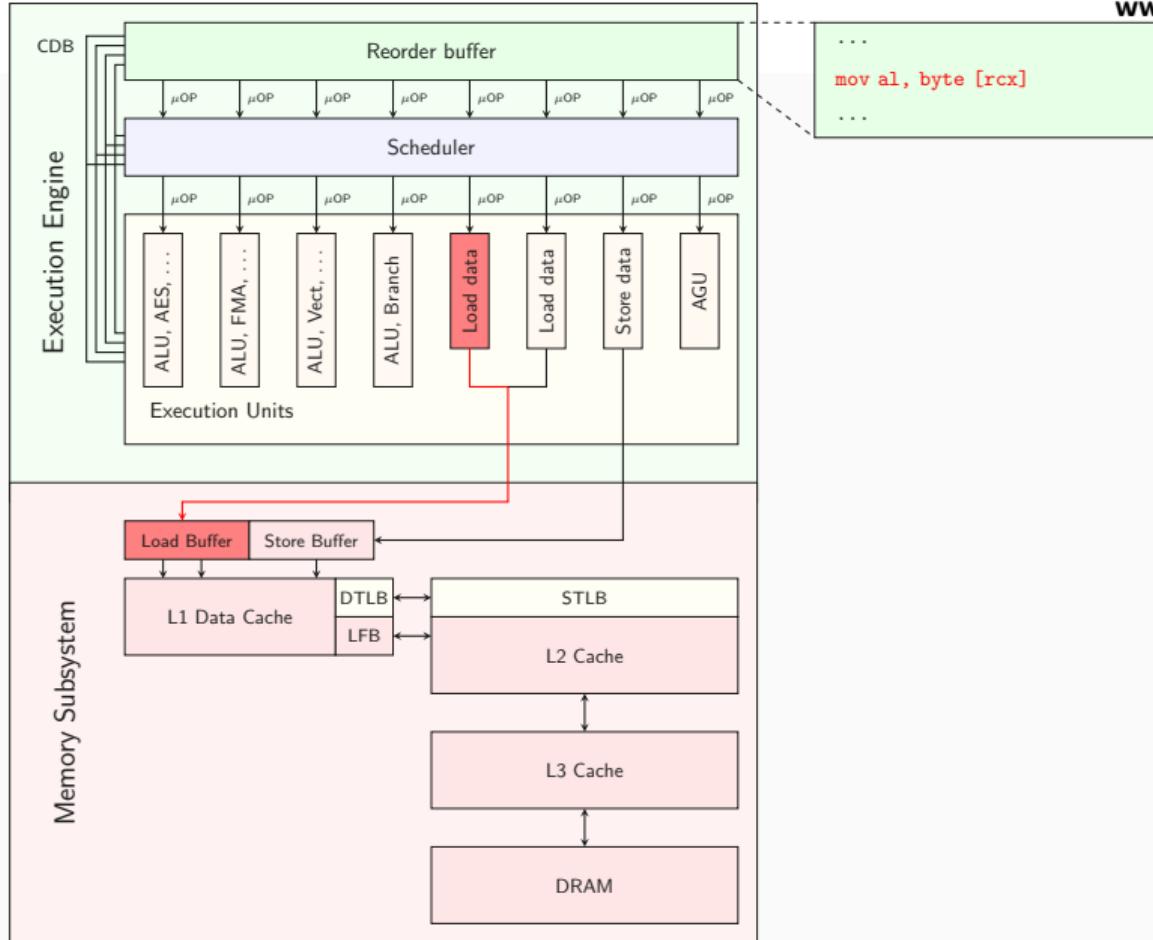


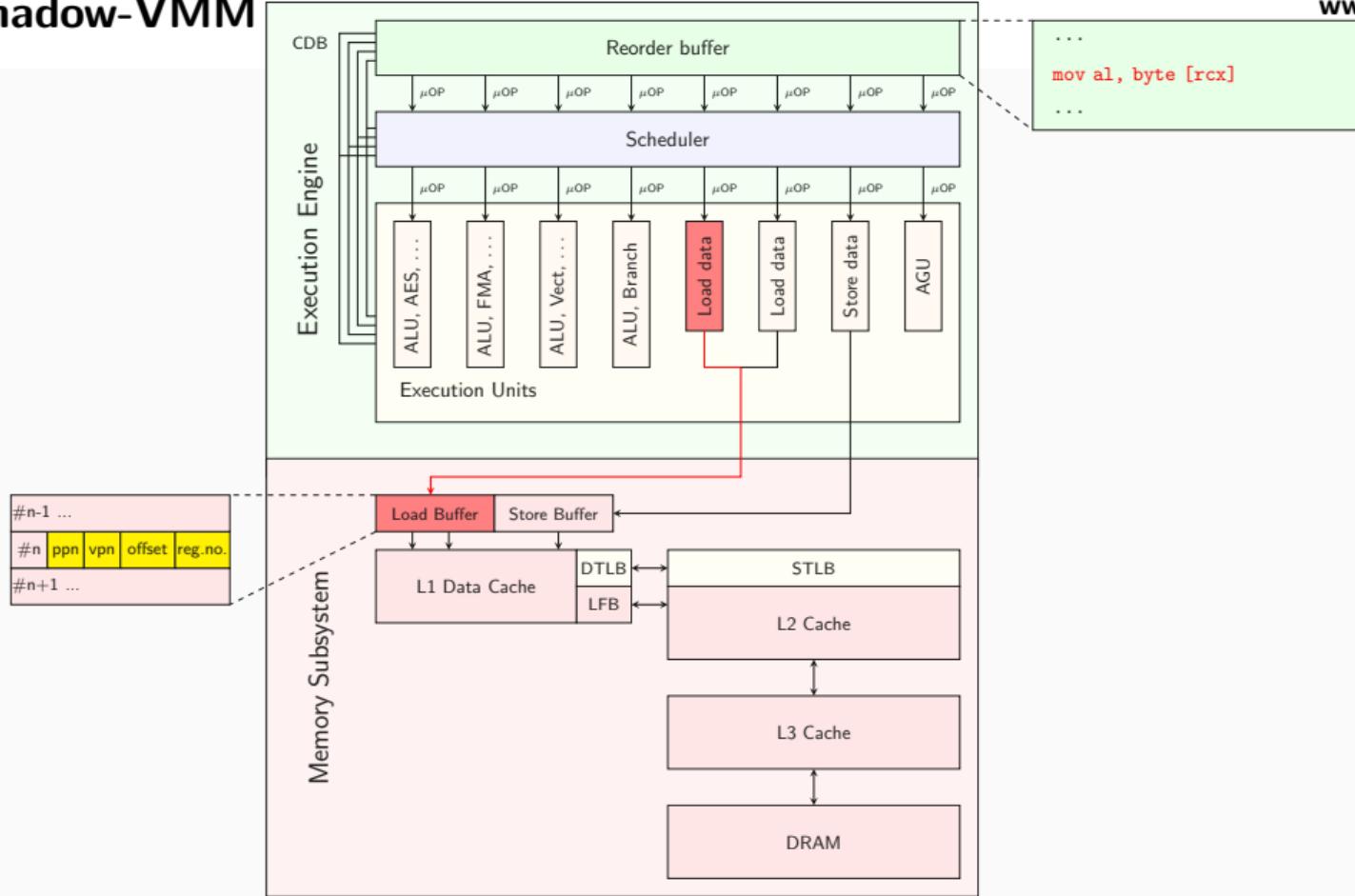


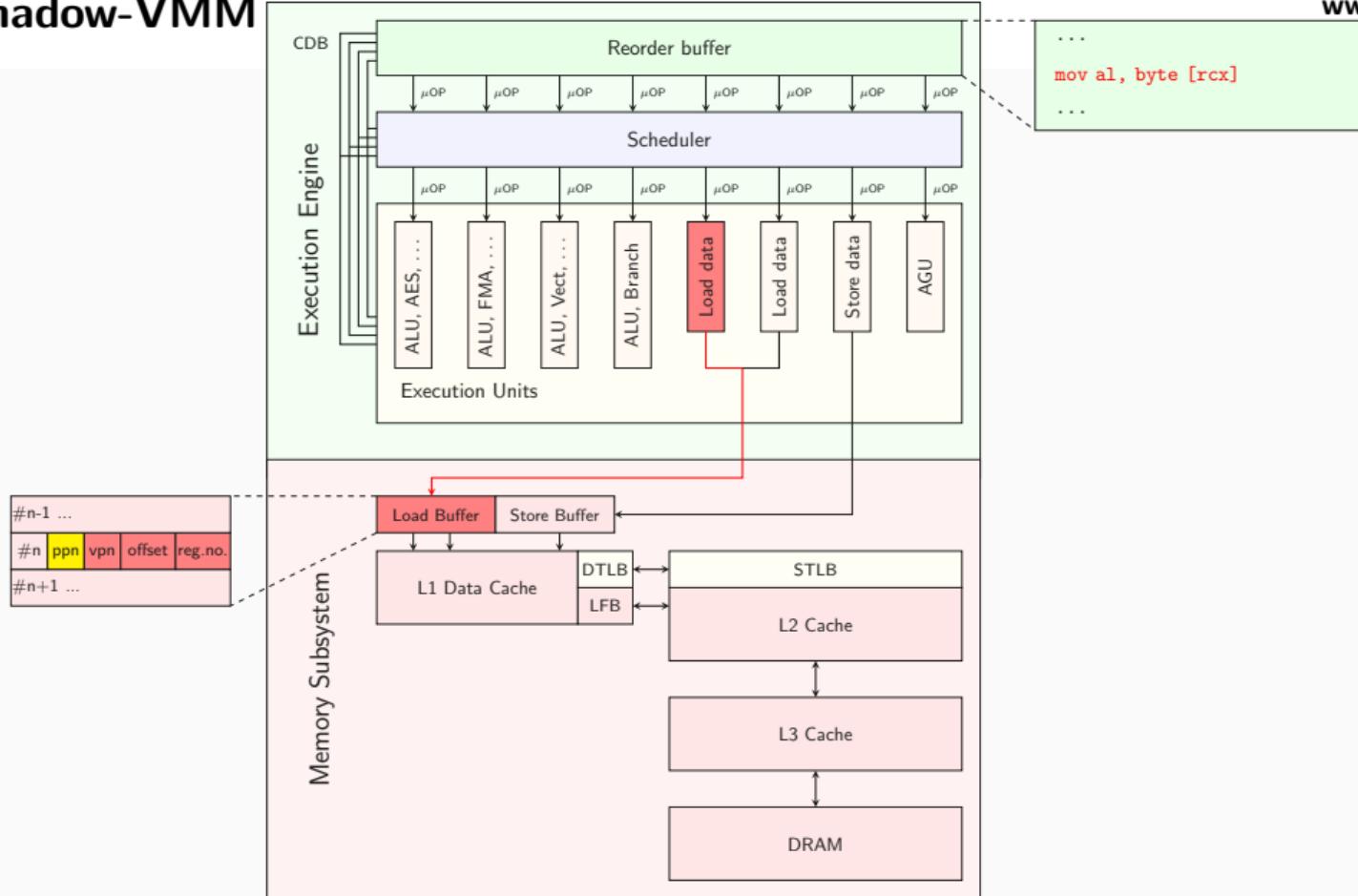


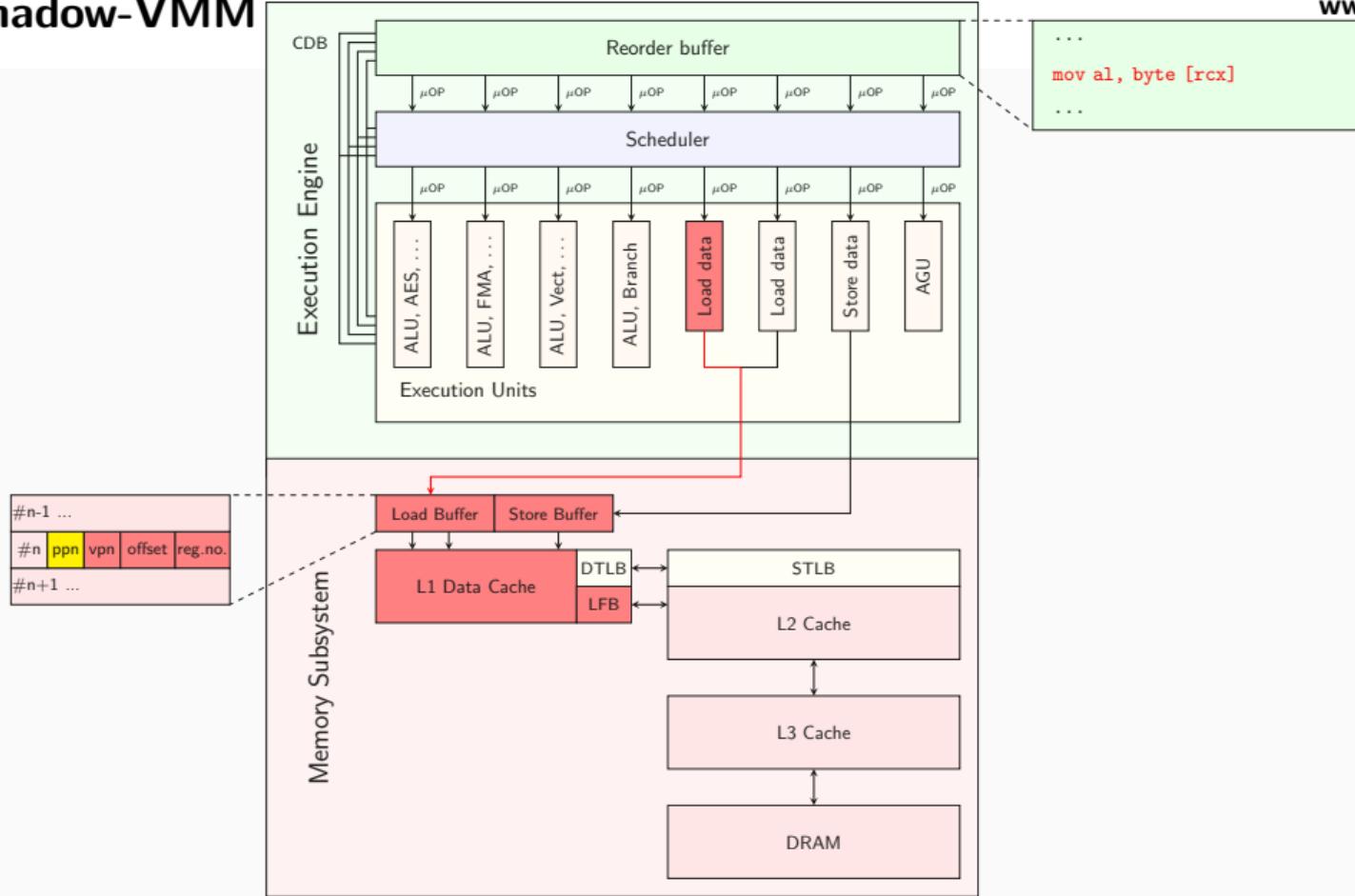


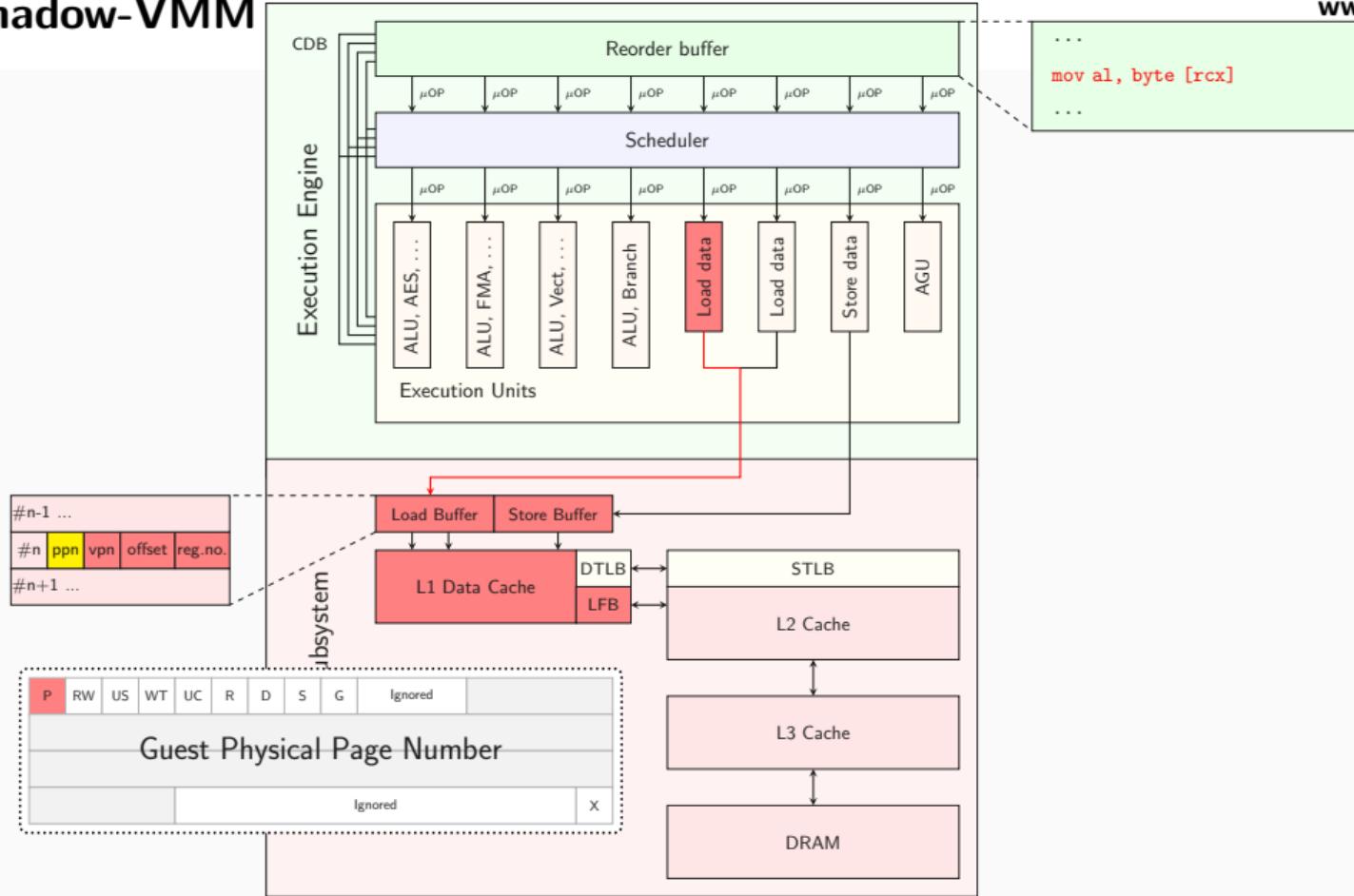


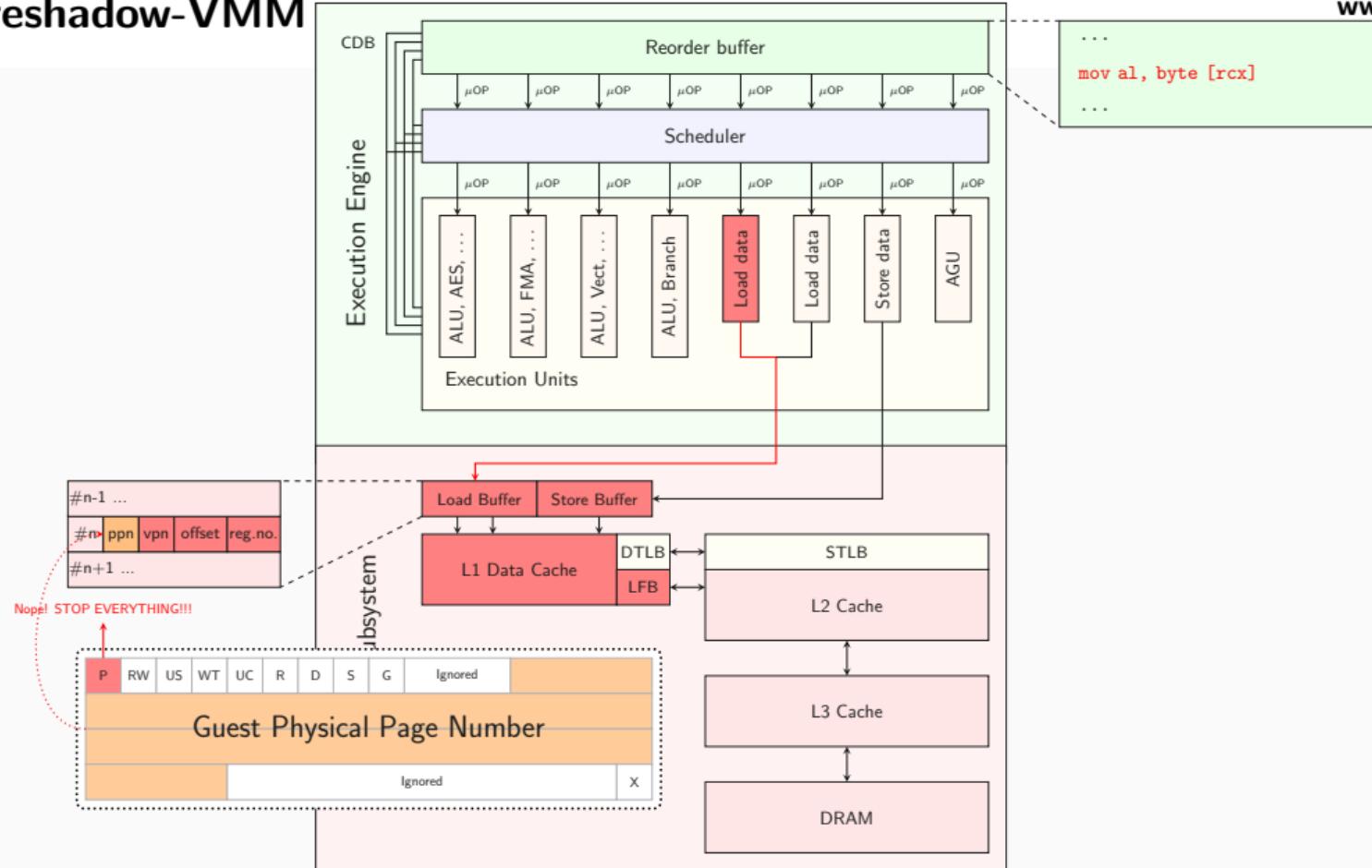


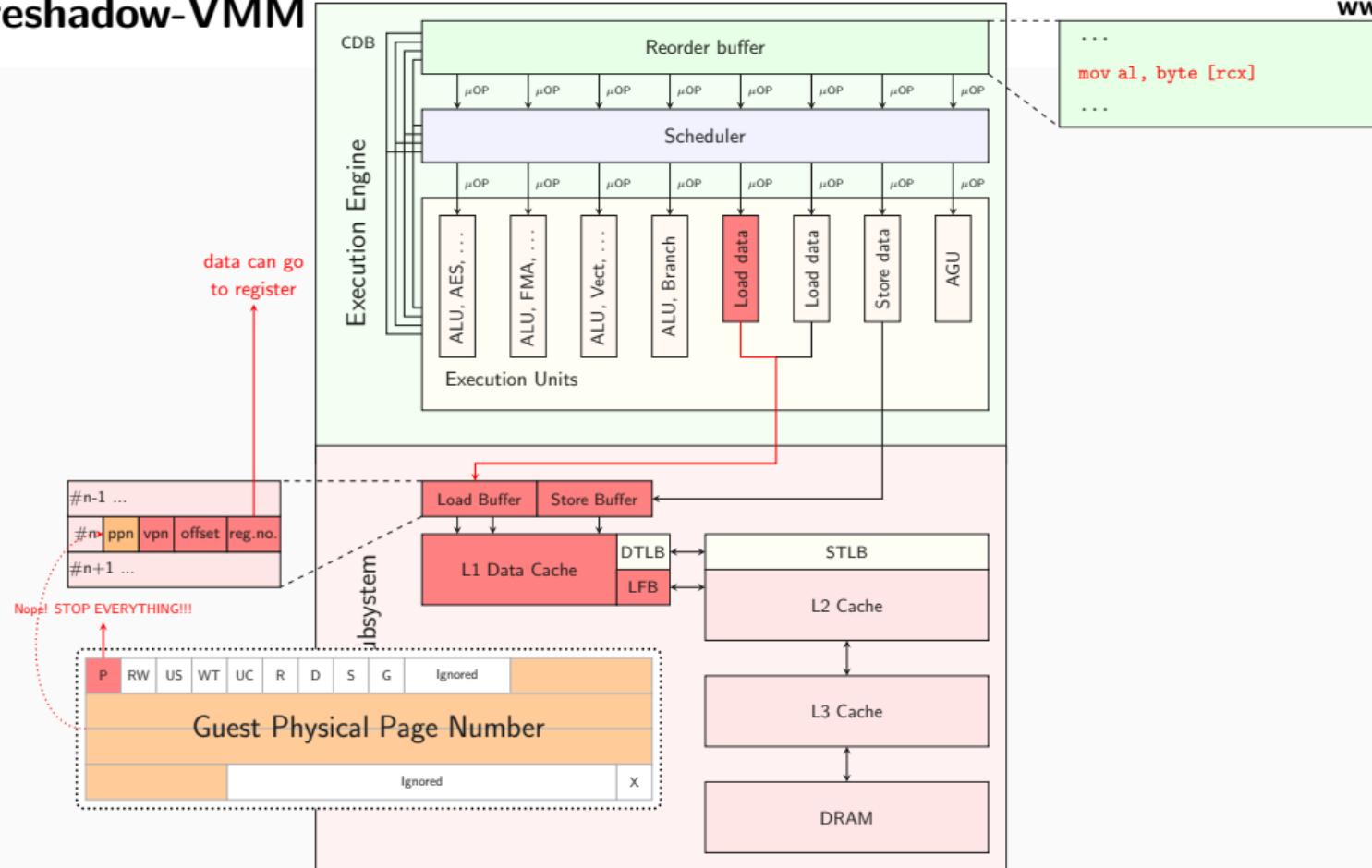


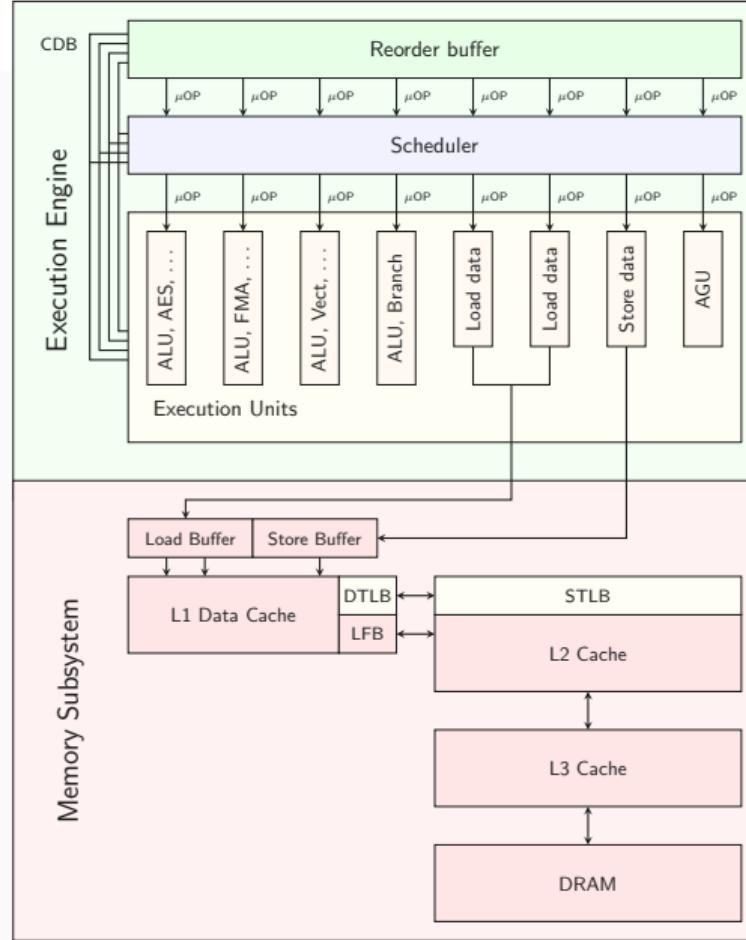


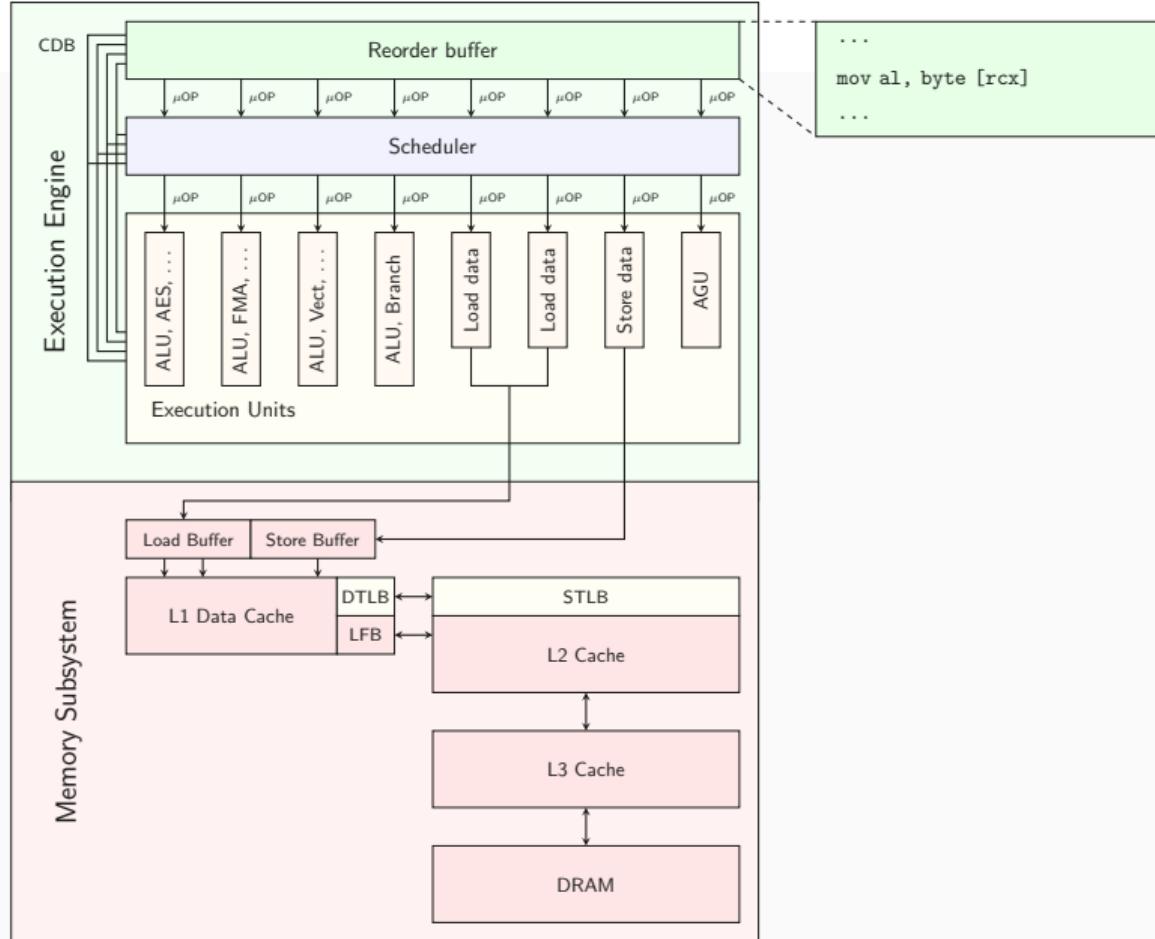


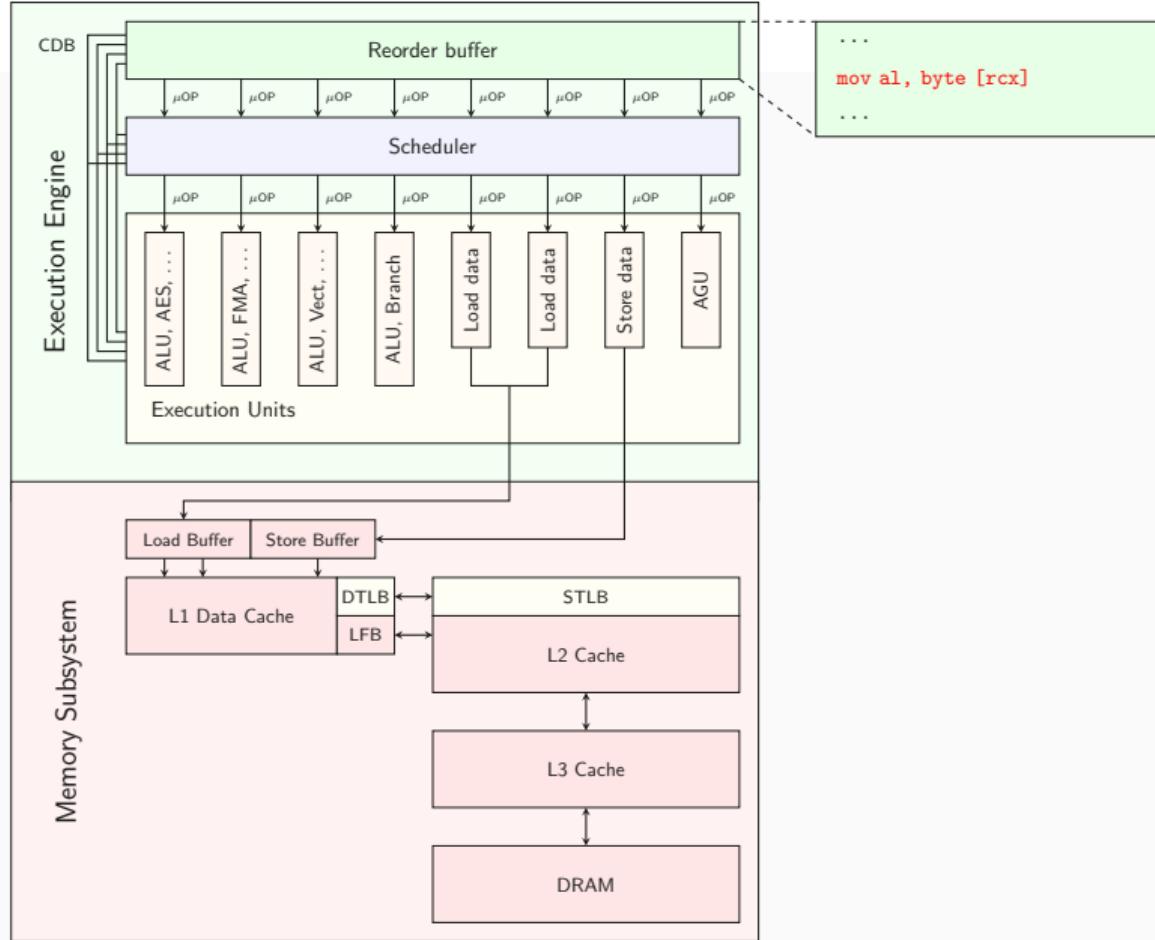


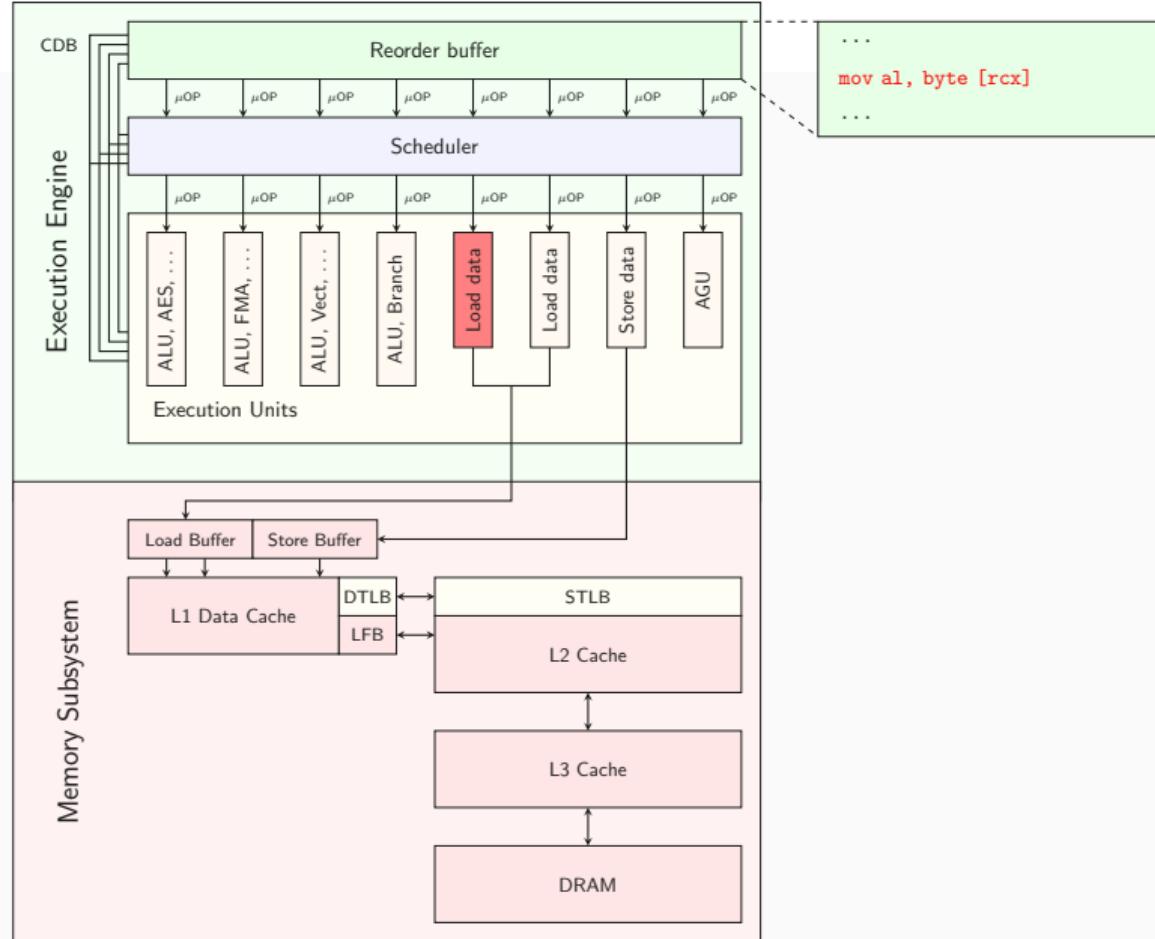


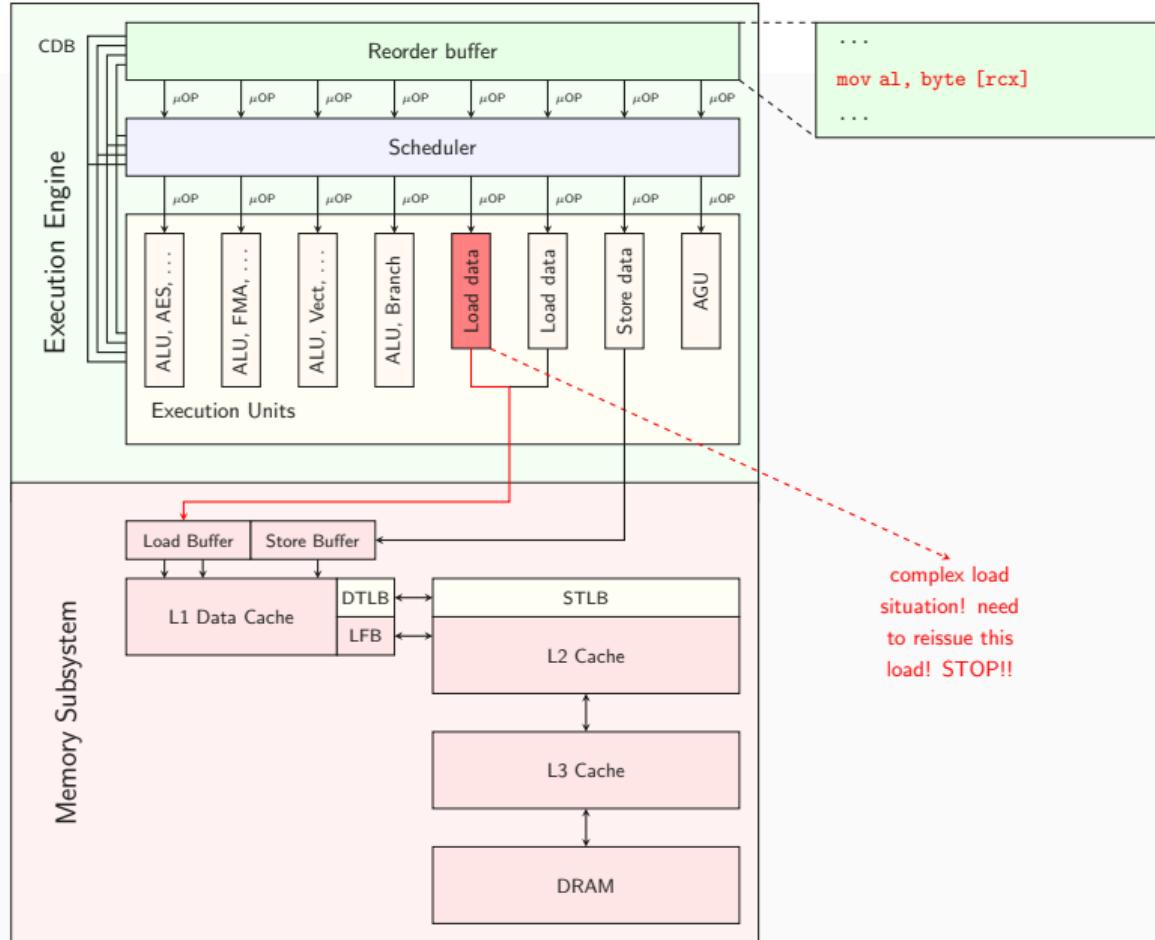


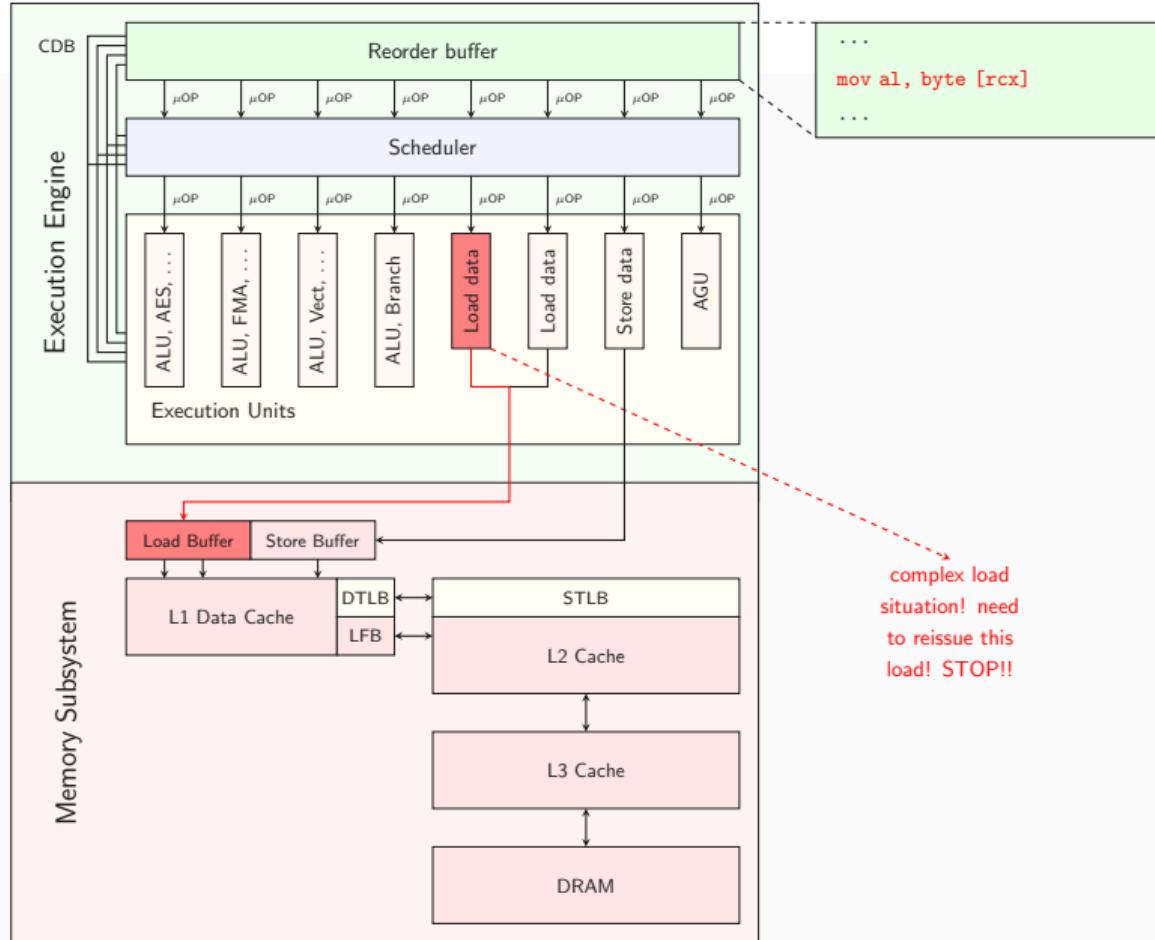


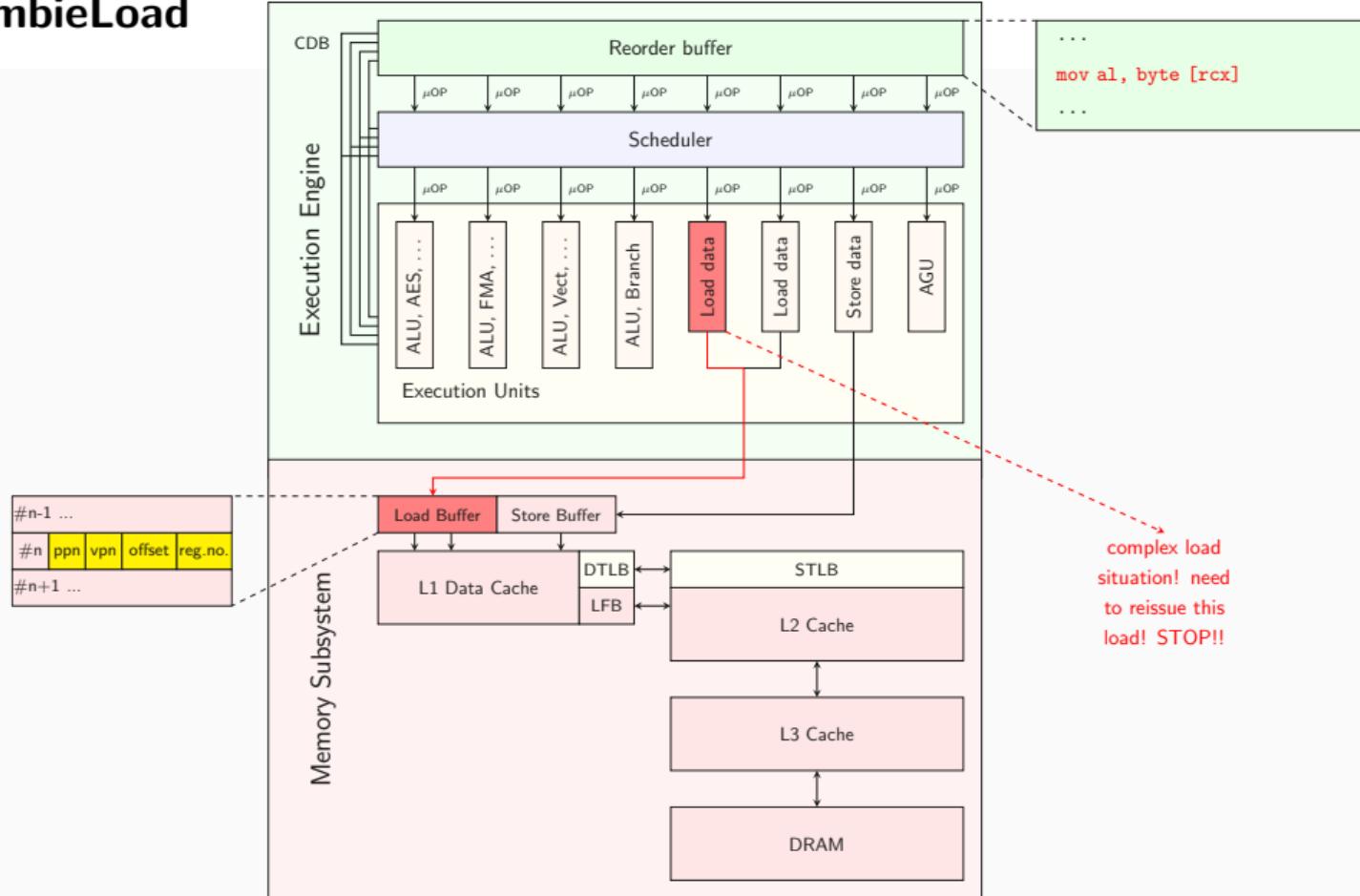


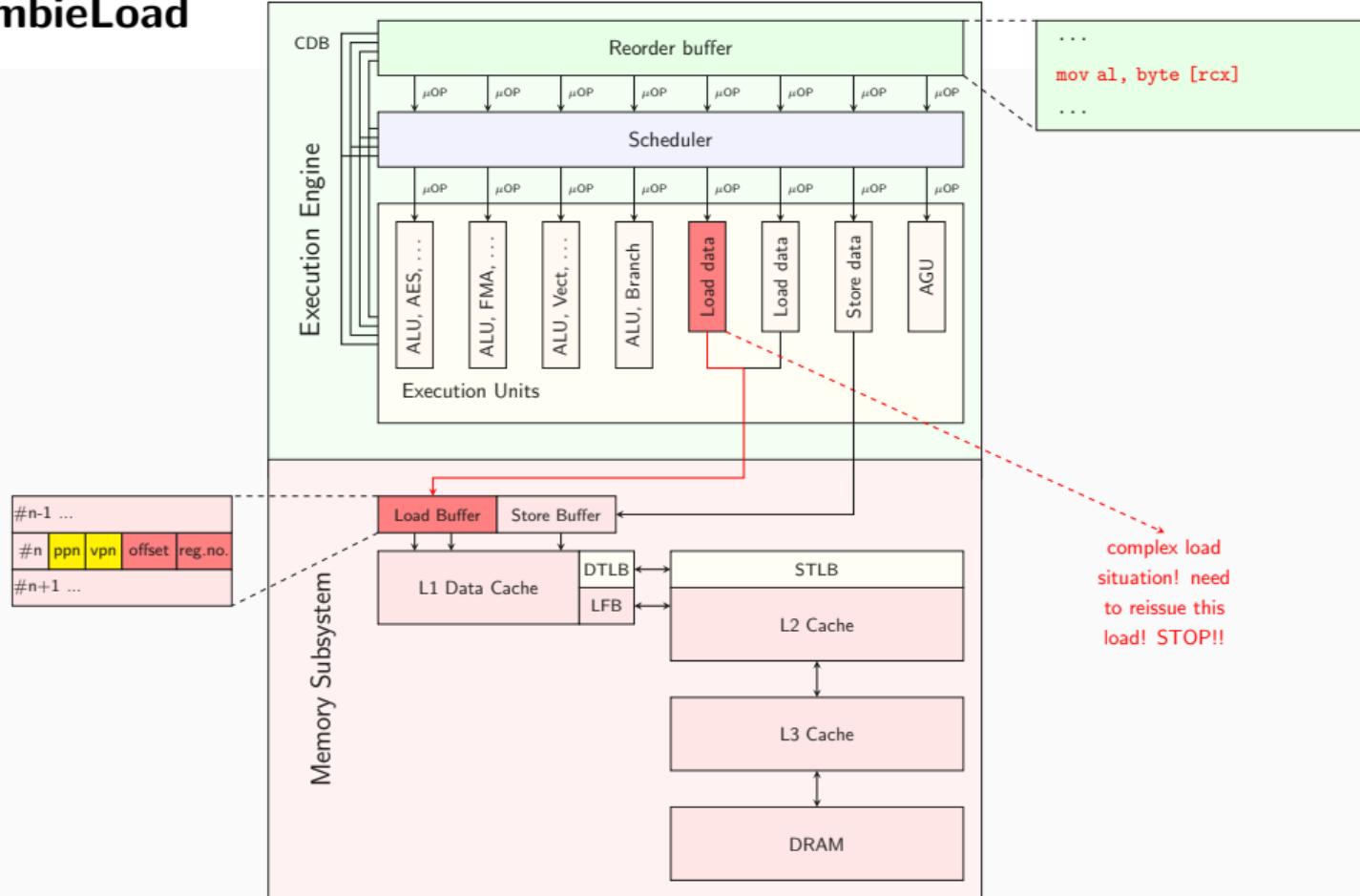


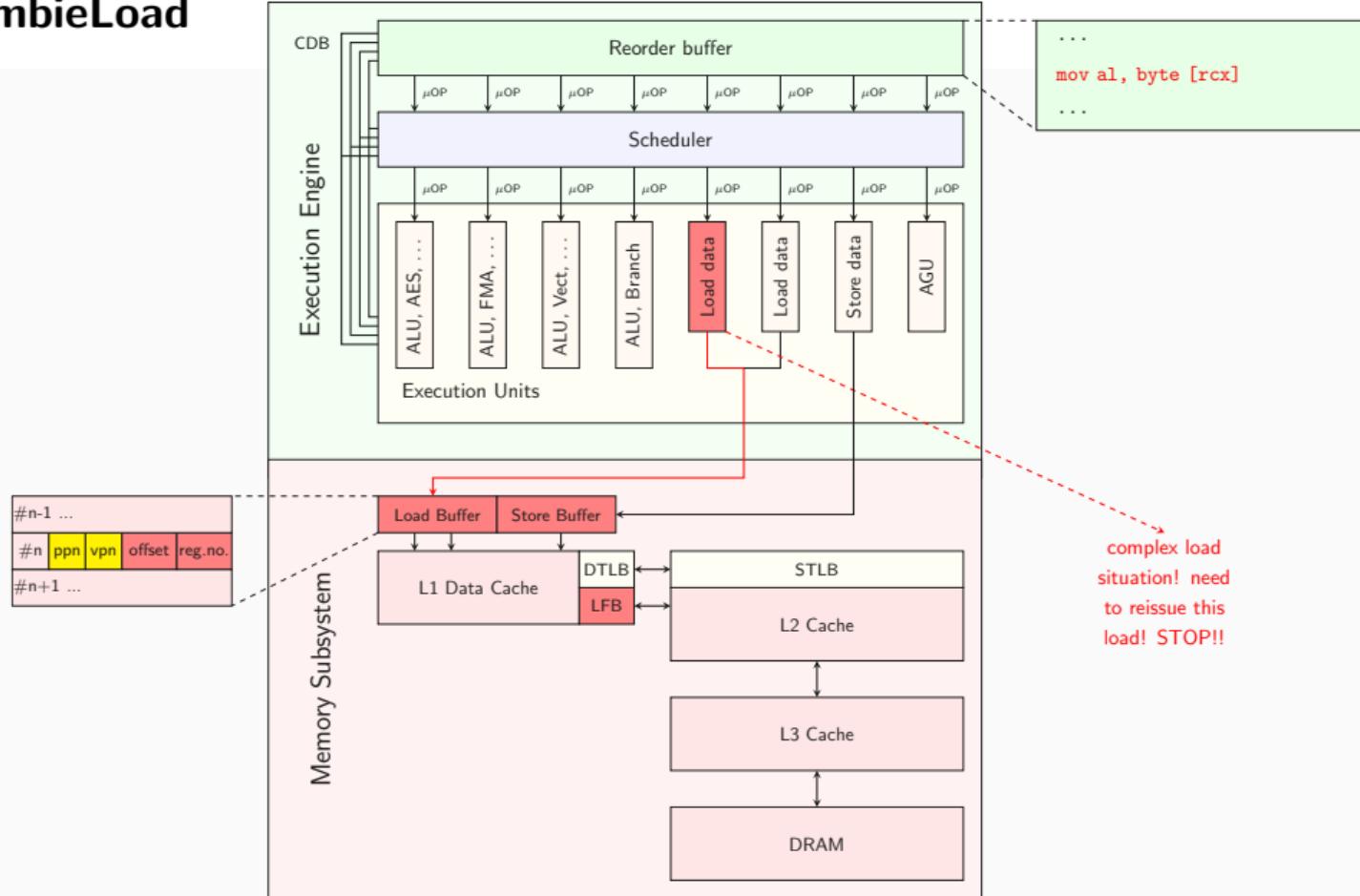


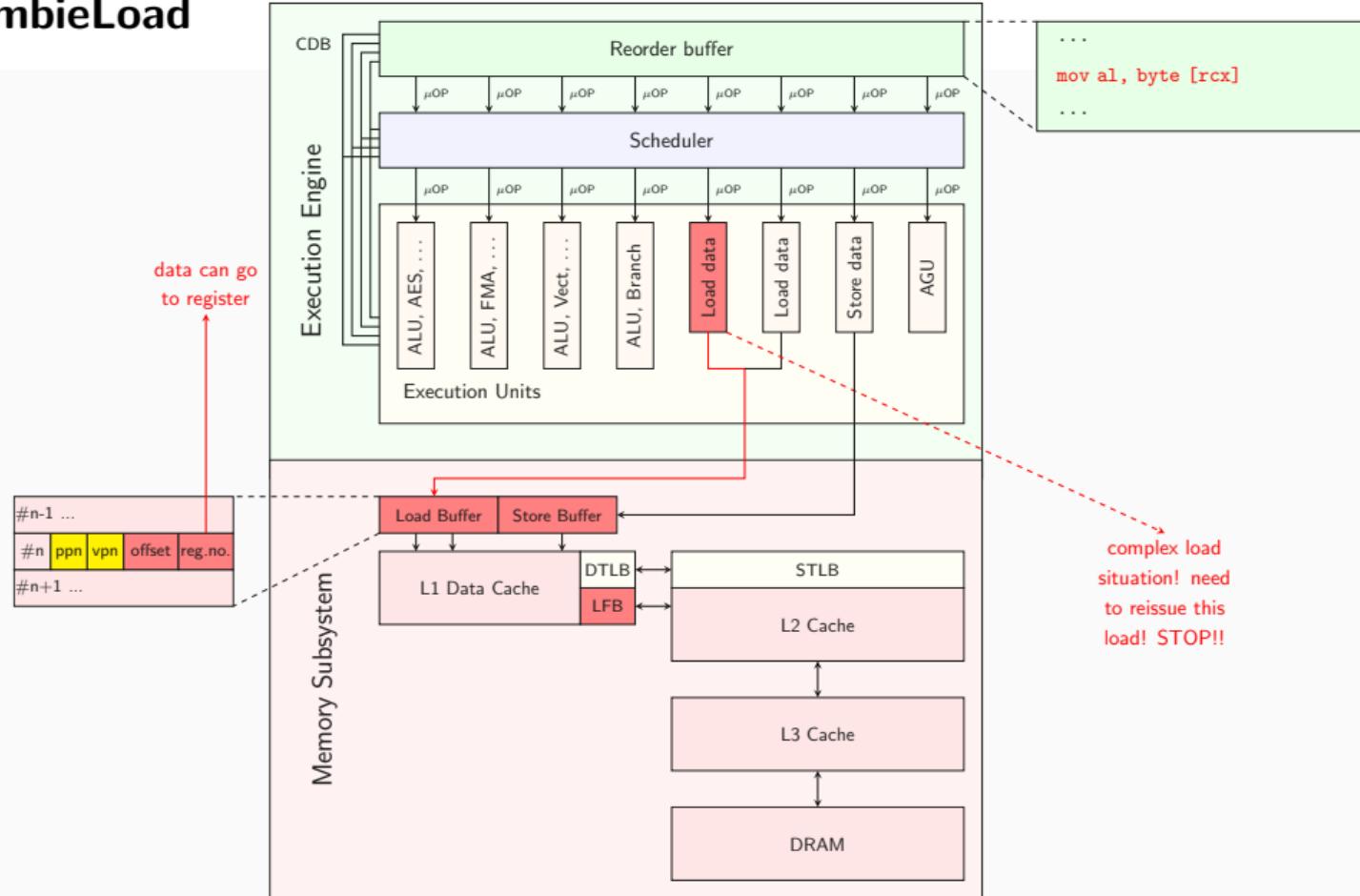












So how did we find it?

So how did we find it?

www.tugraz.at ■



So how did we find it?

www.tugraz.at ■





- our Meltdown PoC always worked on non-L1 memory (for us)



- our Meltdown PoC always worked on non-L1 memory (for us)
 - co-authors confirmed



- our Meltdown PoC always worked on non-L1 memory (for us)
 - co-authors confirmed
 - PoCs/reports → Intel - December 2017



- our Meltdown PoC always worked on non-L1 memory (for us)
 - co-authors confirmed
- PoCs/reports → Intel - December 2017
 - “can't reproduce”



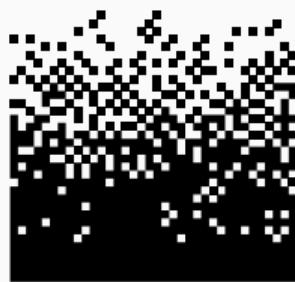
- our Meltdown PoC always worked on non-L1 memory (for us)
 - co-authors confirmed
- PoCs/reports → Intel - December 2017
 - “can't reproduce”
- works with uncacheable



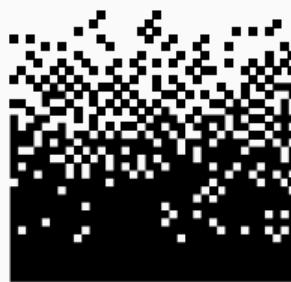
- our Meltdown PoC always worked on non-L1 memory (for us)
 - co-authors confirmed
- PoCs/reports → Intel - December 2017
 - “can't reproduce”
- works with uncacheable
 - PoC → Intel - March 2018



- our Meltdown PoC always worked on non-L1 memory (for us)
 - co-authors confirmed
- PoCs/reports → Intel - December 2017
 - “can't reproduce”
- works with uncacheable
 - PoC → Intel - March 2018
- “It's the LFB” → Intel - May 2018



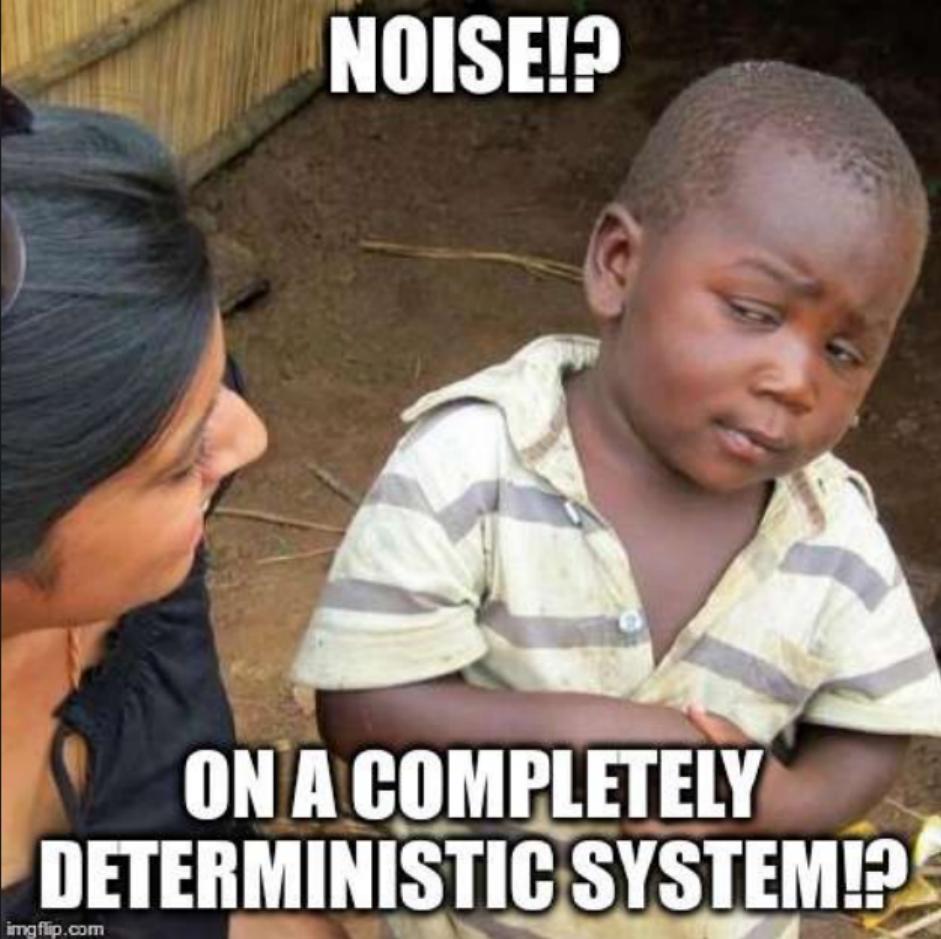




- Meltdown has noise

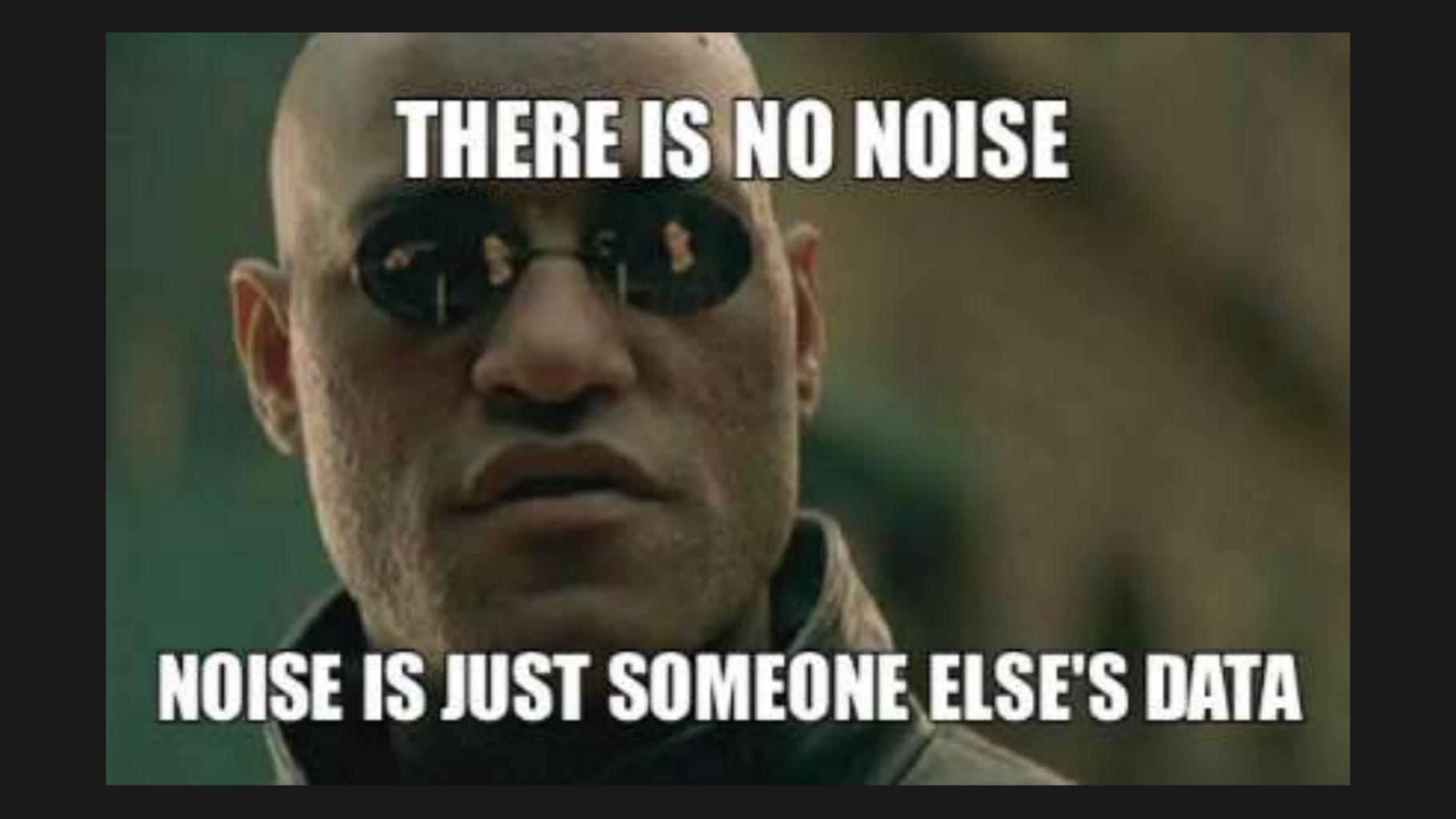


- Meltdown has noise
- Uncacheable → lower signal to noise ratio

A photograph of a young African boy with a shaved head, wearing a yellow and white striped shirt, sitting on the ground with his arms crossed. He is looking off-camera with a confused expression. In the foreground, the profile of a woman's face is visible, looking down at the boy. The background shows a dirt floor and some wooden structures.

NOISE!?

**ON A COMPLETELY
DETERMINISTIC SYSTEM!?**



THERE IS NO NOISE

NOISE IS JUST SOMEONE ELSE'S DATA



zombieLoad : zsh — Konsole <2>

File Edit View Bookmarks Settings Help

michael@hp /tmp/zombieLoad %



zombieLoad : zsh



zombieload:zsh — Konsole <2>

File Edit View Bookmarks Settings Help

michael@hp /tmp/zombieload %



zombieload:zsh



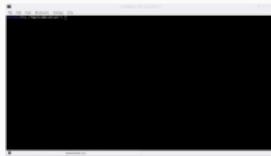
Unix - Frequently Asked Questions (3/7) [Frequent posting]

Section - How do I get rid of zombie processes
that persevere?

([Part1](#) - [Part2](#) - [Part3](#) - [Part4](#) - [Part5](#) - [Part6](#) - [Part7](#) - [Single Page](#))

[[Usenet FAQs](#) | [Web FAQs](#) | [Documents](#) | [RFC Index](#) | [Houses](#)]

- May 2019: 3 new **Meltdown-type** attacks
- Leakage from: line-fill buffer, store buffer, load ports



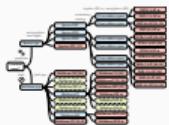
- May 2019: 3 new **Meltdown-type** attacks
- Leakage from: line-fill buffer, store buffer, load ports
- Key take-aways:
 1. Leakage from **intermediate buffers** (\supset L1D)
 2. Transient execution through **micrcode assists** (\supset exceptions)

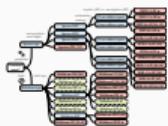


- 
- May 2019: 3 new **Meltdown-type** attacks
 - Leakage from: line-fill buffer, store buffer, load ports
 - Key take-aways:
 1. Leakage from **intermediate buffers** (\supset L1D)
 2. Transient execution through **microcode assists** (\supset exceptions)

⇒ **How to classify in our tree + lessons learned?**

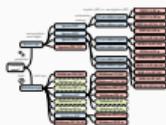
⇒ MD-faulttype-BUF naming scheme





⇒ **MD-faulttype-BUF** naming scheme

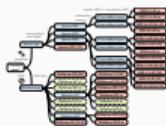
Update leaves – leakage source: REG, L1, LFB, SB, LP



⇒ **MD-faulttype-BUF** naming scheme

Update leaves – leakage source: REG, L1, LFB, SB, LP

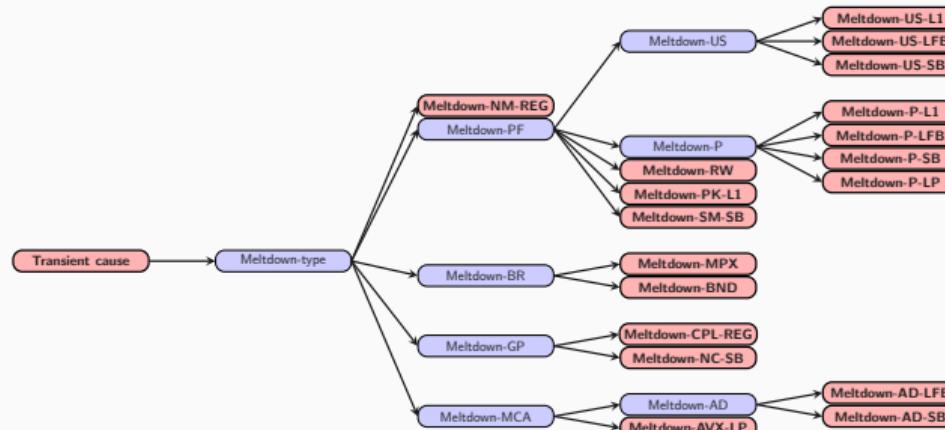
Add sub-branch – trigger Meltdown via μ -code assists

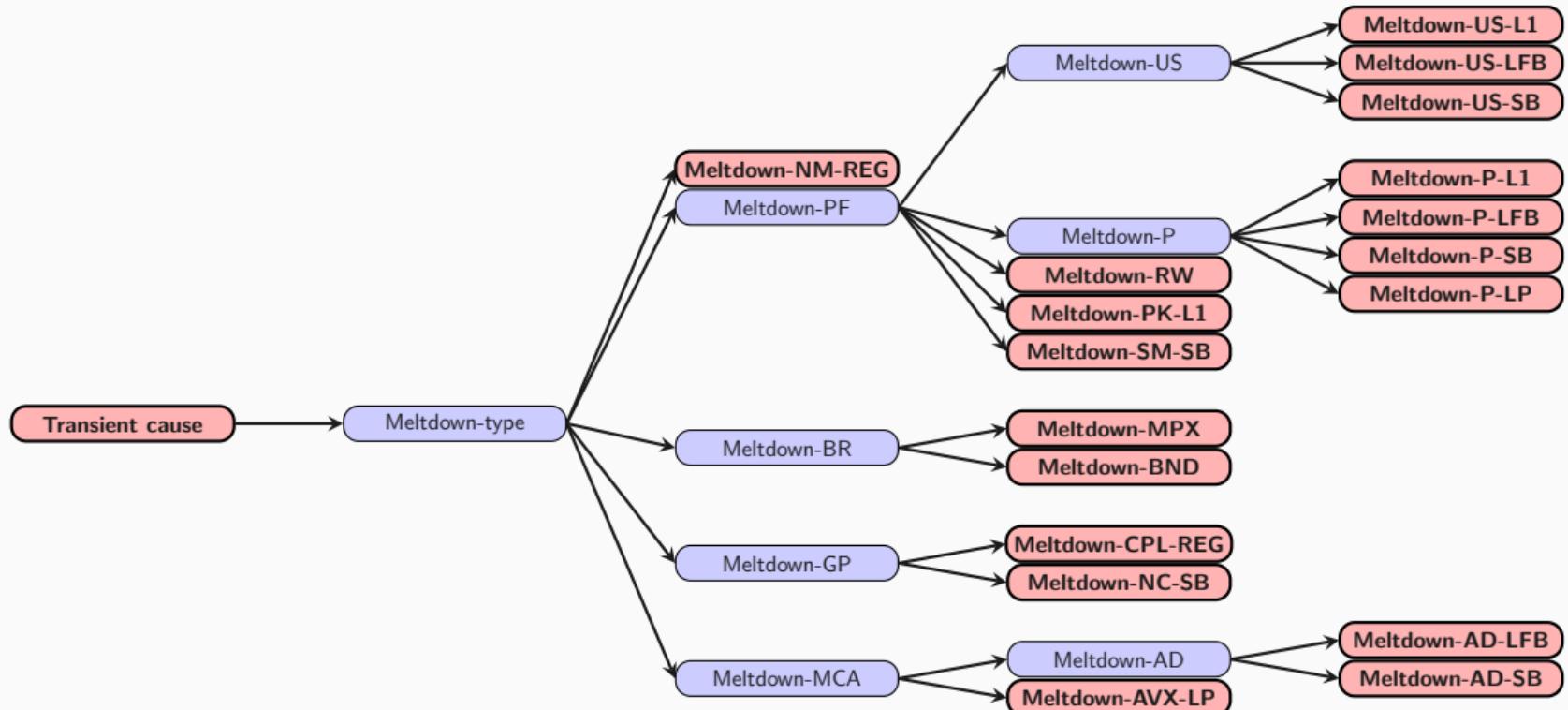


⇒ MD-faulttype-BUF naming scheme

Update leaves – leakage source: REG, L1, LFB, SB, LP

Add sub-branch – trigger Meltdown via μ -code assists

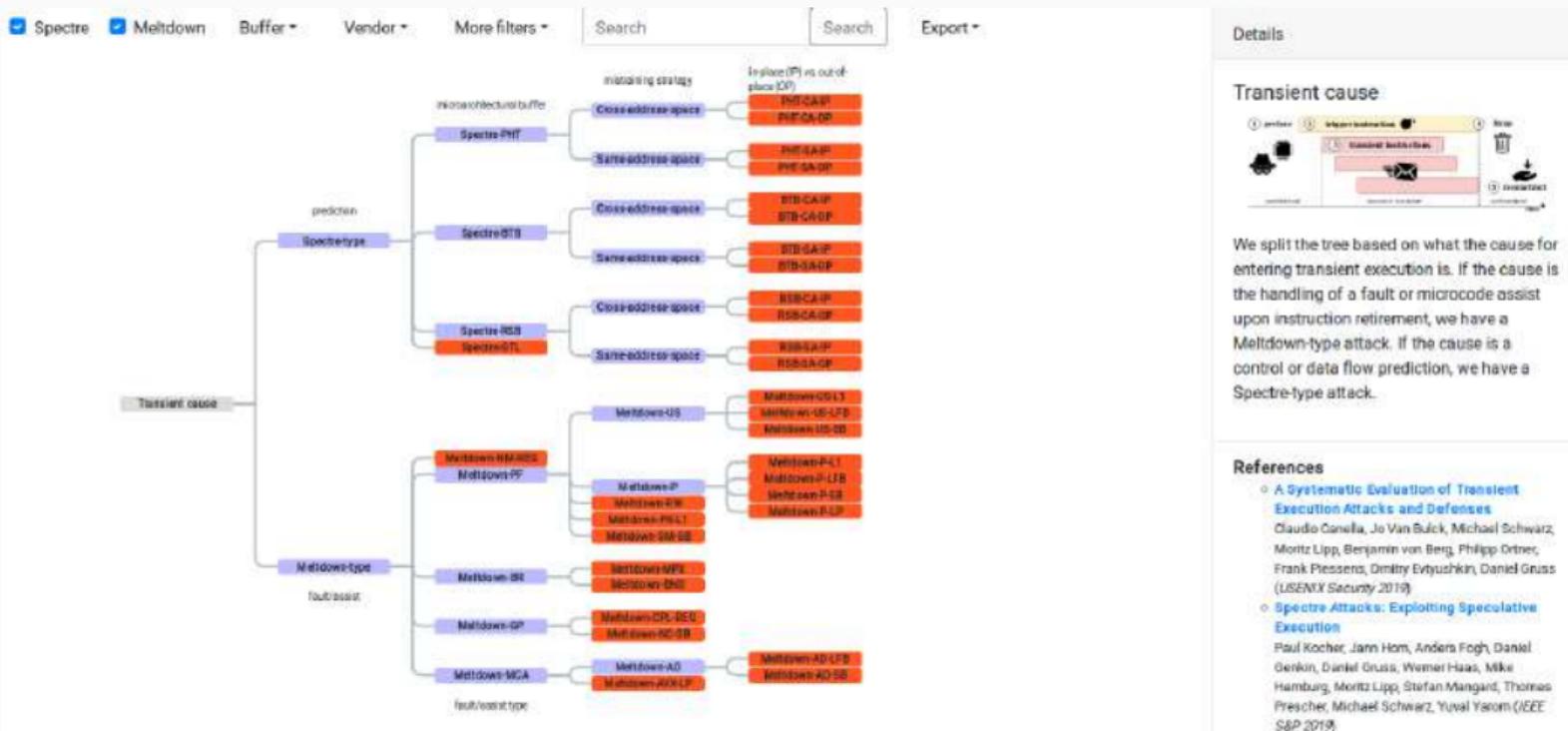








⇒ Our **systematic analysis** (tree search) revealed several overlooked variants
(see Canella et al. "A Systematic Evaluation of Transient Execution Attacks and Defenses", USENIX Security 2019).



Details

Transient cause



We split the tree based on what the cause for entering transient execution is. If the cause is the handling of a fault or microcode assist upon instruction retirement, we have a Meltdown-type attack. If the cause is a control or data flow prediction, we have a Spectre-type attack.

References

- A Systematic Evaluation of Transient Execution Attacks and Defenses
Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin von Berg, Philipp Dreher, Frank Piessens, Dmitry Evtyushkin, Daniel Gruss (USENIX Security 2019)
- Spectre Attacks: Exploiting Speculative Execution
Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, Yariv Yarom (IEEE S&P 2019)



- new class of software-based attacks



- new class of software-based attacks
- many problems to solve around microarchitectural attacks and especially transient execution attacks



- new class of software-based attacks
- many problems to solve around microarchitectural attacks and especially transient execution attacks
- systematically analyze attack space to discover new variants

Side Channels and Transient Execution Attacks

Daniel Gruss

December 5, 2019

Graz University of Technology

References

-  Moritz Lipp et al. ARMageddon: Cache Attacks on Mobile Devices. In: USENIX Security Symposium. 2016.
-  Michael Schwarz et al. Malware Guard Extension: Using SGX to Conceal Cache Attacks. In: DIMVA. 2017.