Meltdown, Spectre, ZombieLoad

Daniel Gruss, Moritz Lipp, Michael Schwarz

October 1, 2019

Graz University of Technology

amazon.com
Prime+Probe

ROWHAMMER.JS

ANOTHER FLIP IN THE ROW

FANTASTIC TIMERS

AND WHERE TO FIND THEM

HIGH-RESOLUTION MICROARCHITECTURAL ATTACKS IN JAVASCRIPT

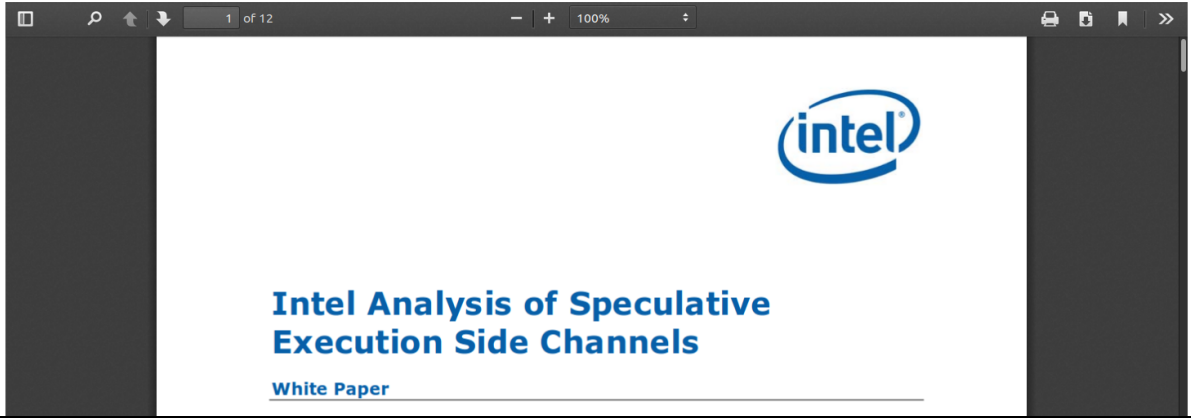
JavaScript zero

REAL JavaScript AND ZERO SIDE-CHANNEL ATTACKS


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

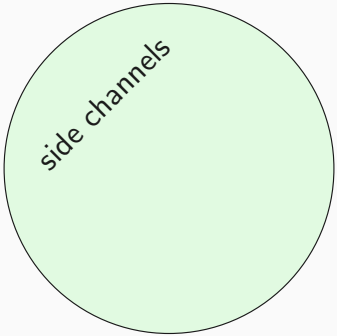
Intel Analysis of Speculative Execution Side Channels

Download PDF



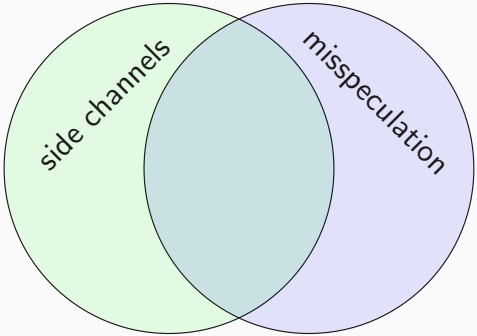
Speculative Side-Channel Attacks?

Speculative Side-Channel Attacks?



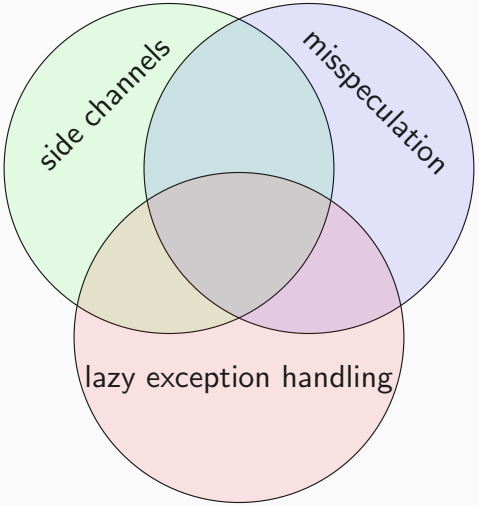
- traditional cache attacks (crypto, keys, etc)

Speculative Side-Channel Attacks?



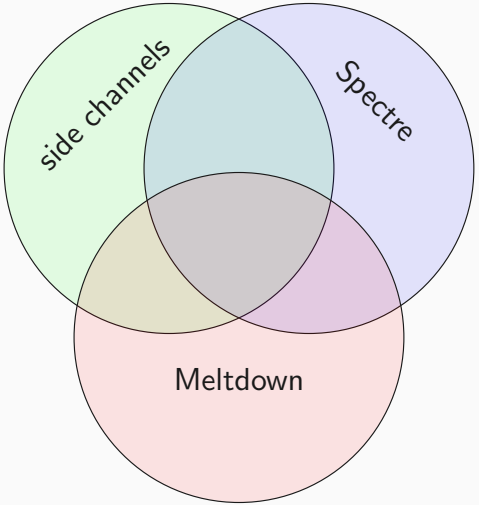
- traditional cache attacks (crypto, keys, etc)
- actual misspeculation (e.g., branch misprediction)

Speculative Side-Channel Attacks?



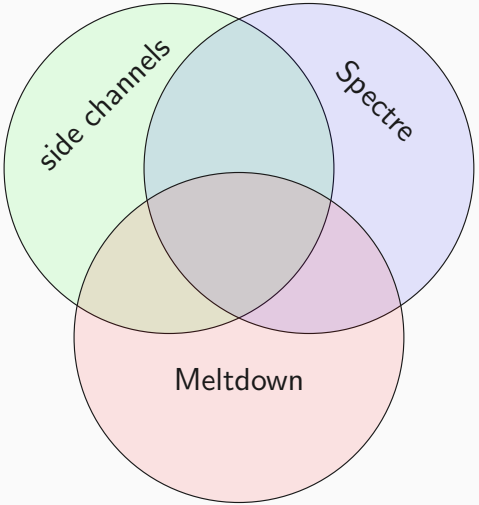
- traditional cache attacks (crypto, keys, etc)
- actual misspeculation (e.g., branch misprediction)
- Meltdown, Foreshadow, ZombieLoad, etc

Speculative Side-Channel Attacks?



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

Speculative Side-Channel Attacks?



- 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.

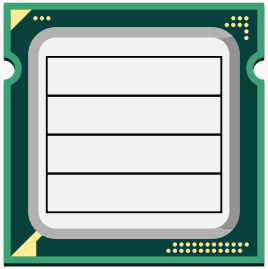
Can store **ALL** kinds of food.

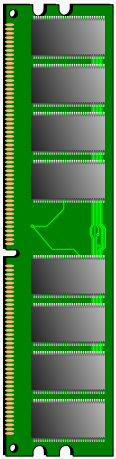
ONLY TODAY INSTEAD OF ~~$1,300~~

$1,299

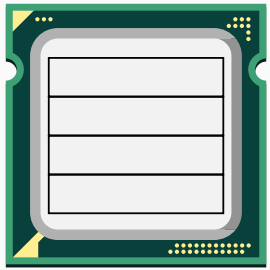
ORDER VIA PHONE: +555 12345

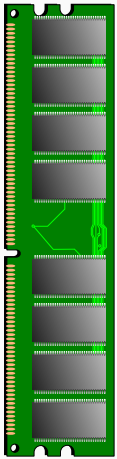
```
printf("%d", i);
printf("%d", i);
```



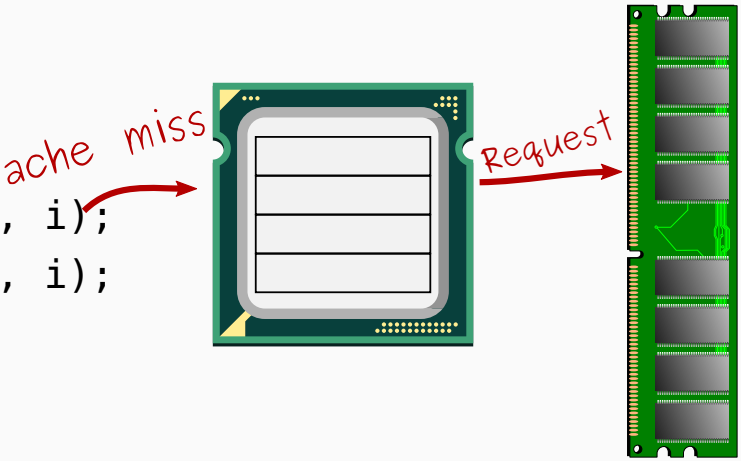


```
printf("%d", i);
printf("%d", i);
```

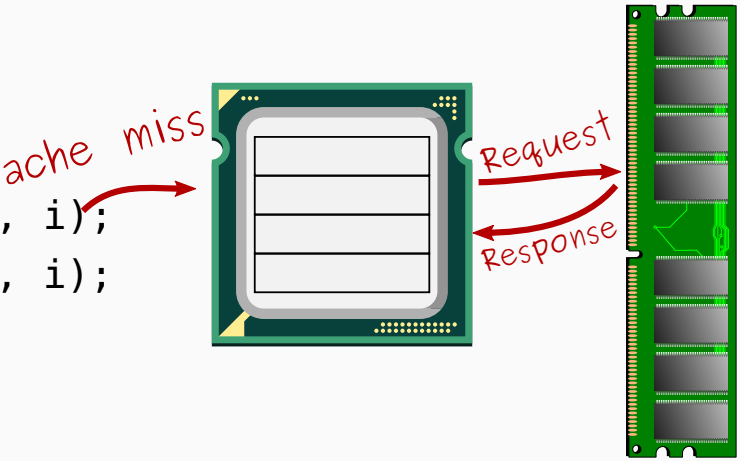




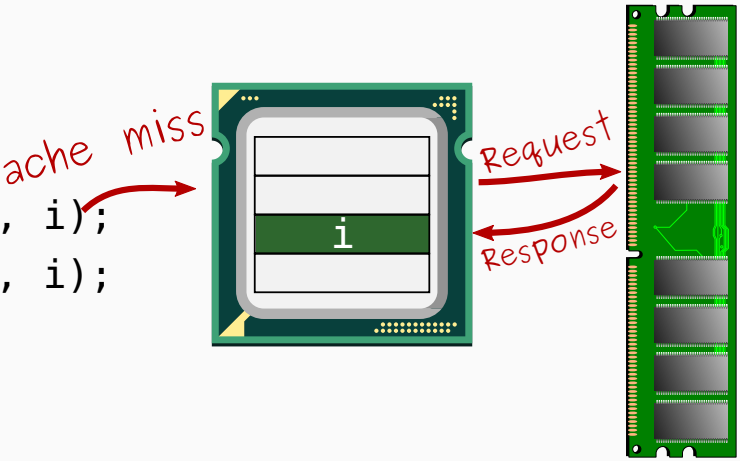
```
printf("%d", i);
printf("%d", i);
```

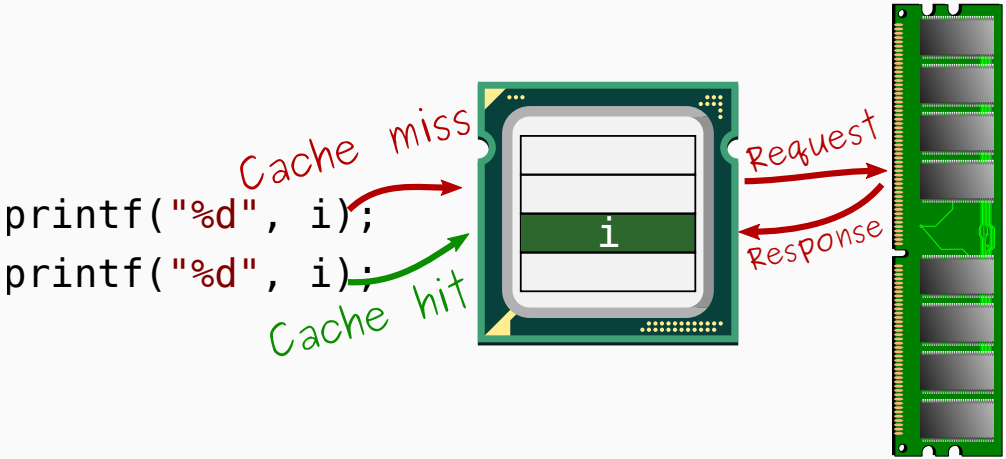


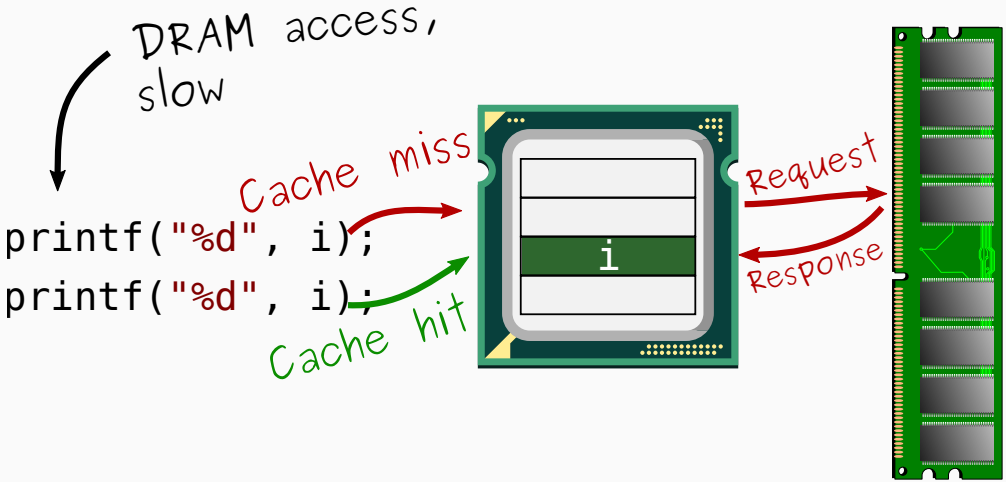
```
printf("%d", i);
printf("%d", i);
```

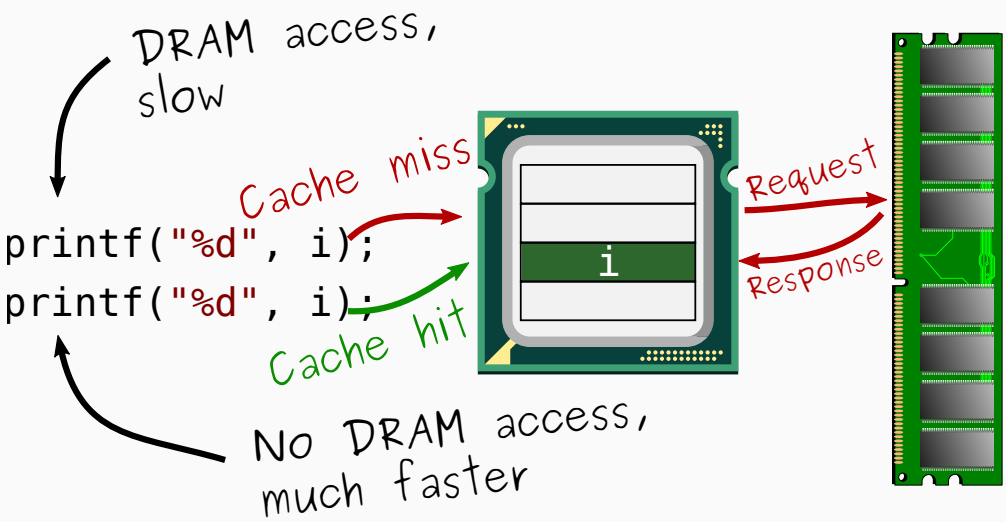


```
printf("%d", i);
printf("%d", i);
```




printf("%d", i);
printf("%d", i);
Cache miss
Cache hit
i
Request
Response


DRAM access, slow
printf("%d", i);
printf("%d", i);
Cache miss
Cache hit
i
Request
Response

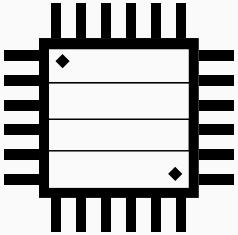

DRAM access,
slow
printf("%d", i);
printf("%d", i);
Cache miss
Cache hit
i
Request
Response
No DRAM access,
much faster

Shared Memory

ATTACKER

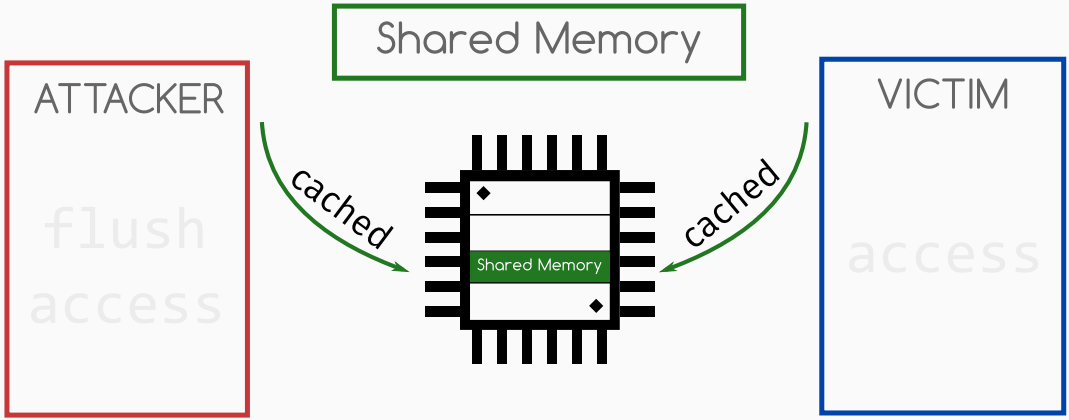
flush

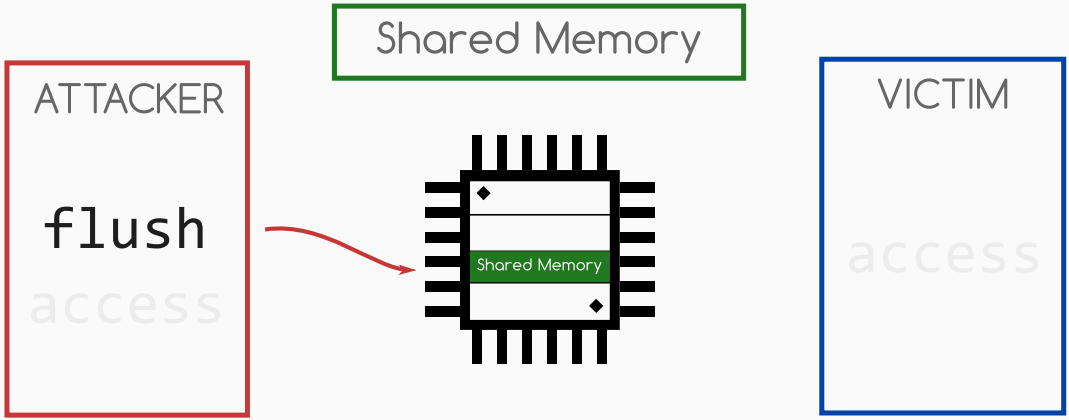
access

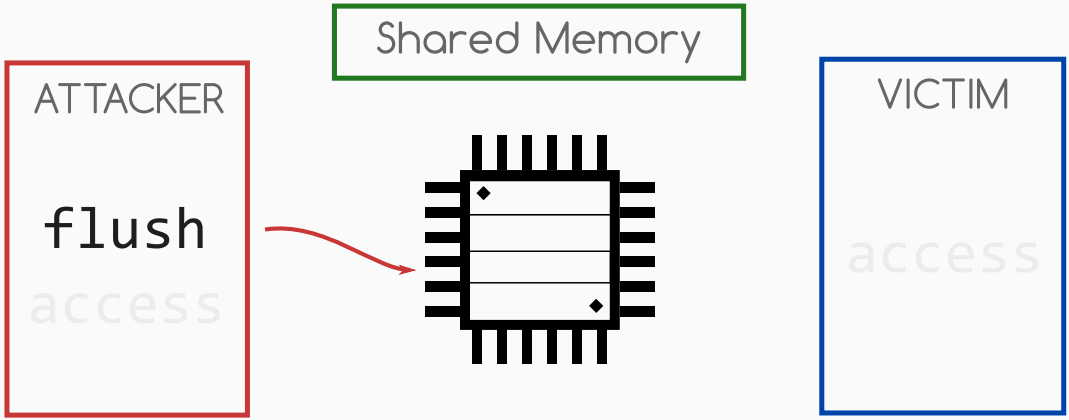


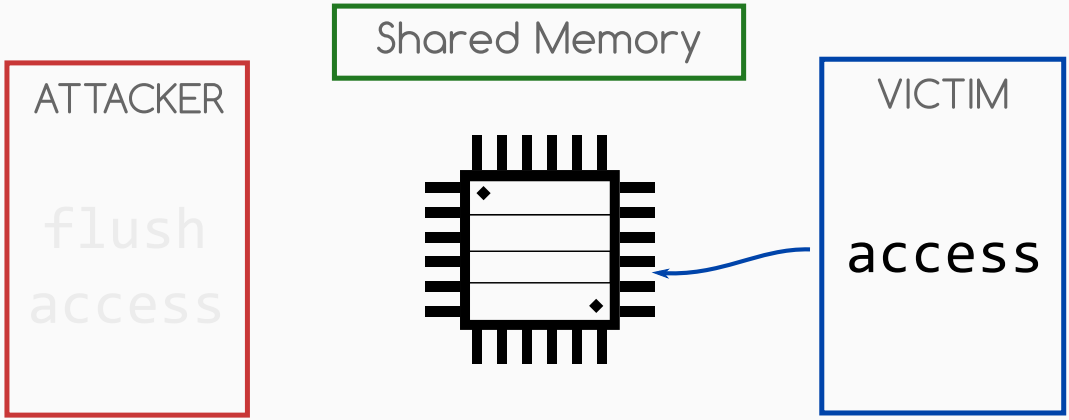
VICTIM

access


Shared Memory
ATTACKER
flush
access
cached
Shared Memory
cached
VICTIM
access


Shared Memory
ATTACKER
flush
access
Shared Memory
VICTIM
access


Shared Memory
ATTACKER
flush
access
VICTIM
access

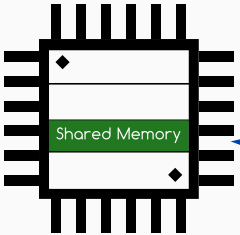

Shared Memory
ATTACKER
flush
access
VICTIM
access

Shared Memory

ATTACKER

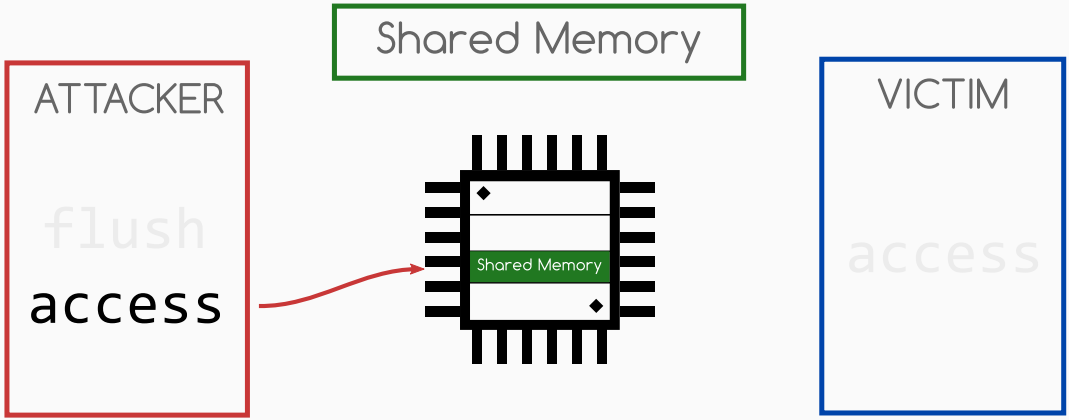
flush

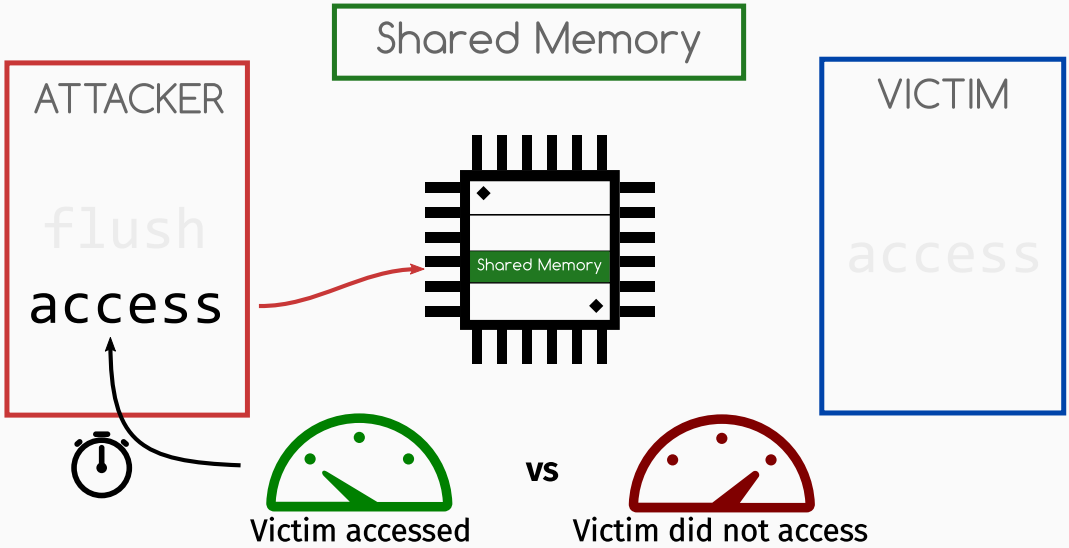
access



VICTIM

access


Shared Memory
ATTACKER
flush
access
Shared Memory
VICTIM
access


Shared Memory
ATTACKER
flush
access
Shared Memory
VICTIM
access
vs
Victim accessed
Victim did not access

- 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`

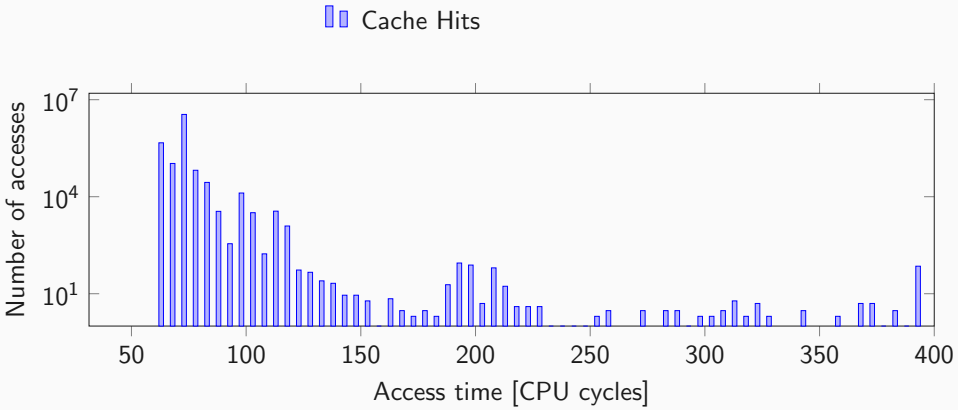
- use pseudo-serializing instruction `rdtscp` (recent CPUs)
- and/or use serializing instructions like `cpuid`
- and/or use fences like `mfence`

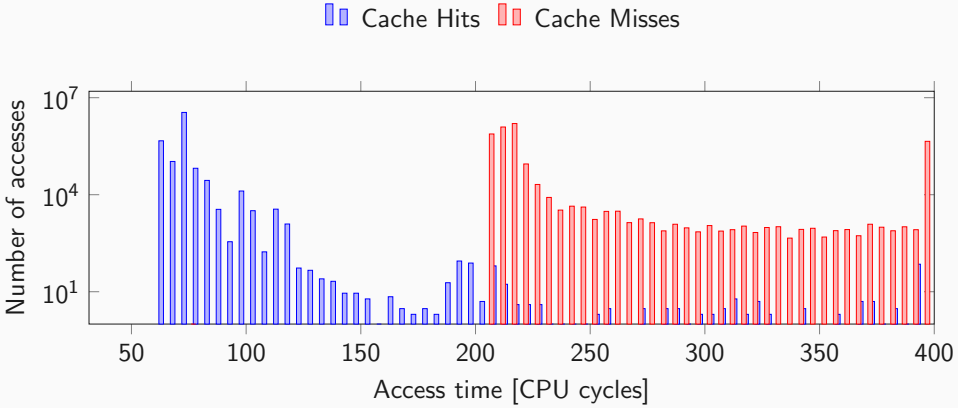
Intel, *How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures White Paper*, December 2010.

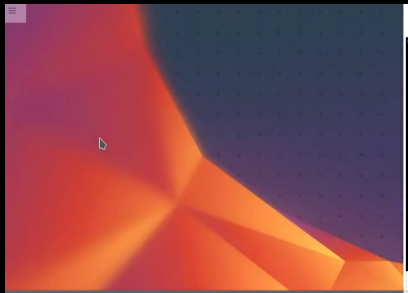
AUGUST 22, 2018 BY BRUCE

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.


Cache Hits
Number of accesses
10^7
10^4
10^1
50
100
150
200
250
300
350
400
Access time [CPU cycles]


Cache Hits
Cache Misses
Number of accesses
10^7
10^4
10^1
50
100
150
200
250
300
350
400
Access time [CPU cycles]



local — Konsole

File Edit View Bookmarks Settings Help

michael@michael-hp ~ %

local

sender (ec2) — Konsole

File Edit View Bookmarks Settings Help

ubuntu@ip-172-31-51-32 ~ %

sender (ec2)

receiver (ec2) — Konsole

File Edit View Bookmarks Settings Help

ubuntu@ip-172-31-58-32 ~ %

receiver (ec2)

sender (ec2) — Konsole <2>

File Edit View Bookmarks Settings Help

-= slurm 0.4.3 on ip-172-31-51-32 =-

```
                x
                x
               xx
              xxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                xx
               xx
               x
               x
```

Active Interface: eth0 Interface Speed: unknown

Current RX Speed: 0.05 KB/s	Current TX Speed: 0.33 KB/s
Graph Top RX Speed: 0.98 KB/s	Graph Top TX Speed: 2.64 KB/s
Overall Top RX Speed: 0.98 KB/s	Overall Top TX Speed: 2.64 KB/s
Received Packets: 84	Transmitted Packets: 82
MBytes Received: 0.005 MB	MBytes Transmitted: 0.019 MB
Errors on Receiving: 0	Errors on Transmission: 0

sender (ec2)

receiver (ec2) — Konsole <2>

File Edit View Bookmarks Settings Help

-= slurm 0.4.3 on ip-172-31-58-32 =-

```
                x
                x
                x
                x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                xx
                x
                x
                x
```

Active Interface: eth0 Interface Speed: unknown

Current RX Speed: 0.05 KB/s	Current TX Speed: 0.29 KB/s
Graph Top RX Speed: 0.36 KB/s	Graph Top TX Speed: 0.84 KB/s
Overall Top RX Speed: 0.36 KB/s	Overall Top TX Speed: 0.84 KB/s
Received Packets: 38	Transmitted Packets: 36
MBytes Received: 0.002 MB	MBytes Transmitted: 0.010 MB
Errors on Receiving: 0	Errors on Transmission: 0

receiver (ec2)

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

Line 17, Column 1 INSERT Soft Tabs: 4 (8) UTF-8 Normal

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

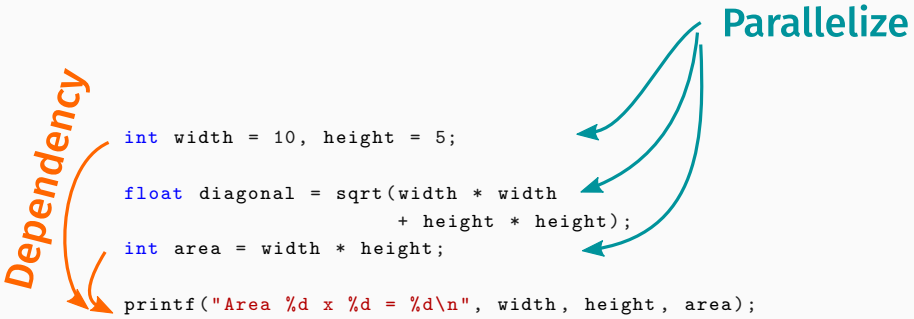
4. Fry vegetables until golden and softened

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);
```


Parallelize
Dependency
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

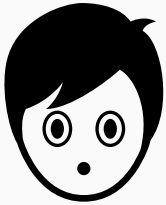
- Add another layer of indirection to test

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

- Add another layer of indirection to test

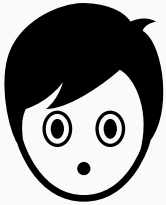
```
char data = *(char*) 0xffffffff81a000e0;
array[data * 4096] = 0;
```

- 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

Terminal

File Edit View Search Terminal Help

```
mschwarz@lab06:~/Documents$
```

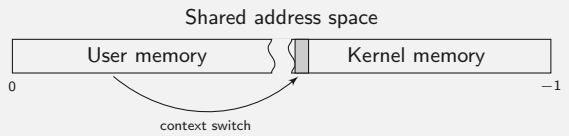
Kernel **A**ddress **I**solation to have **S**ide channels **E**fficiently **R**emoved

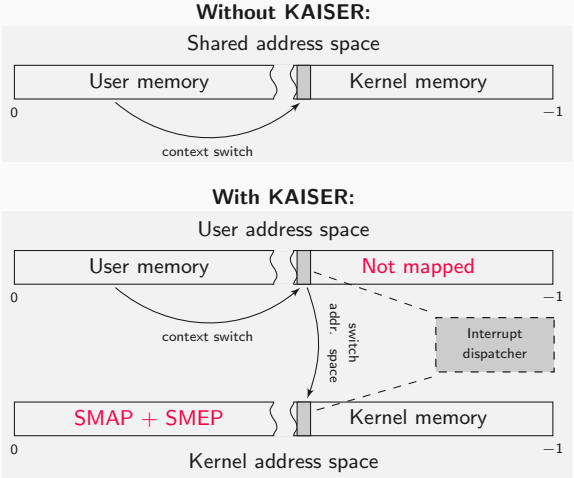
KAISER /ˈkʌɪzə/

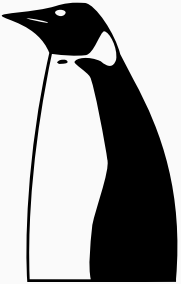
1. [german] Emperor, ruler of an empire
2. largest penguin, emperor penguin

Kernel **A**ddress **I**solation to have **S**ide channels **E**fficiently **R**emoved

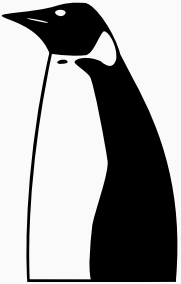
Without KAISER:



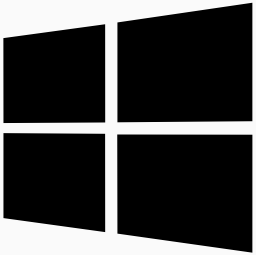

Without KAISER:
Shared address space
User memory
Kernel memory
0
−1
context switch
With KAISER:
User address space
User memory
Not mapped
0
−1
context switch
addr. space switch
Interrupt dispatcher
SMAP + SMEP
Kernel memory
0
−1
Kernel address space



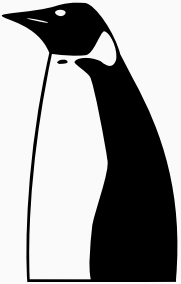
- Our patch
- Adopted in Linux



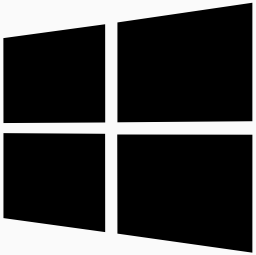
- Our patch
- Adopted in Linux



- Adopted in Windows

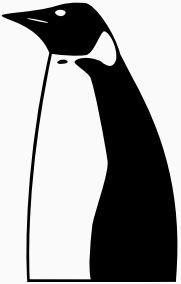


- Our patch
- Adopted in Linux

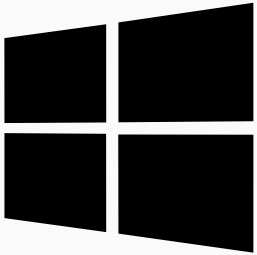


- Adopted in Windows

- Adopted in OSX/iOS



- Our patch
- Adopted in Linux

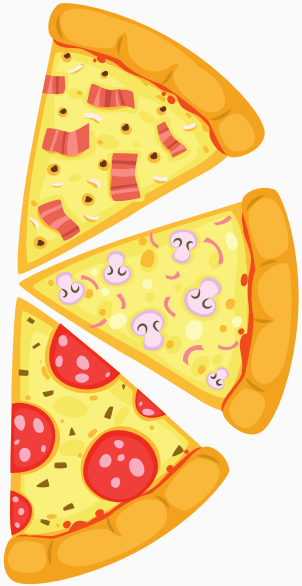


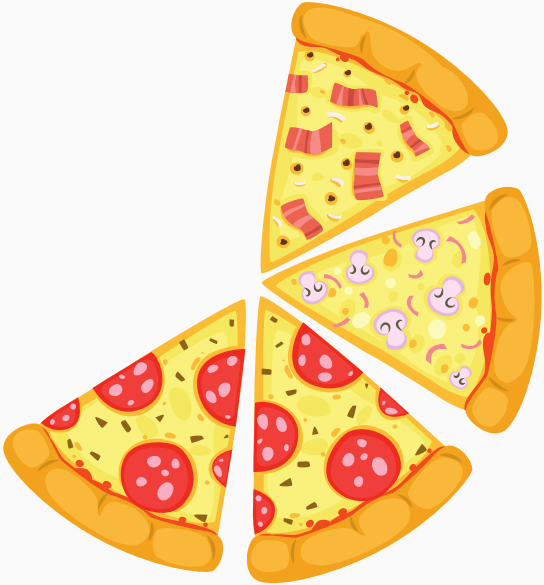
- Adopted in Windows

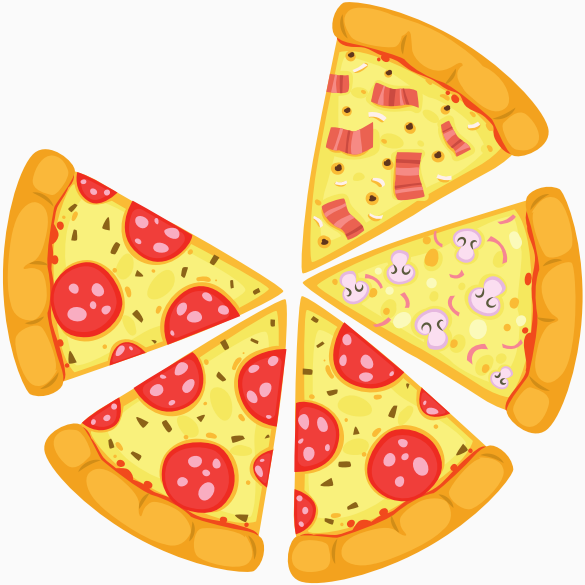
- Adopted in OSX/iOS

$\rightarrow$ **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

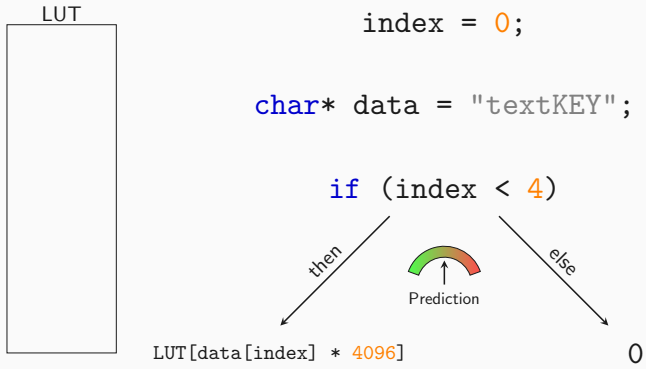

Pizza

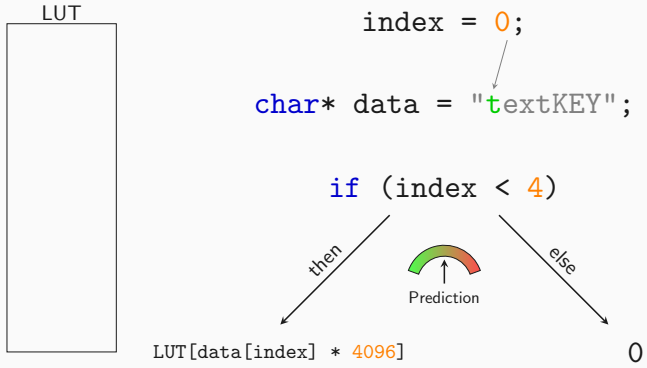

PIZZA
SPECIAL RECIPES

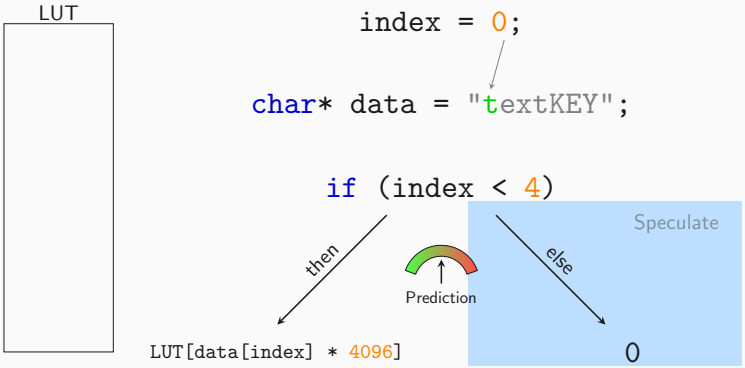

Pizza

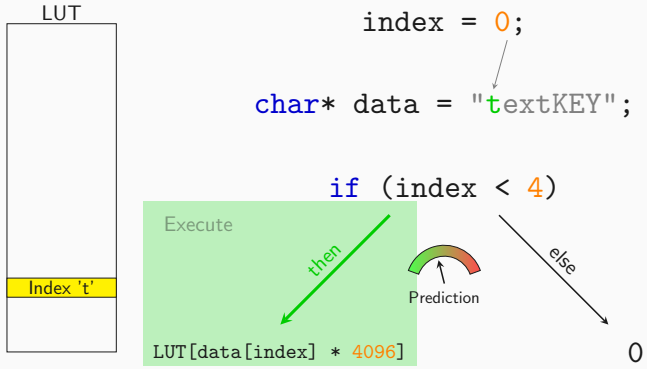

PIZZA
SPECIAL RECIPES

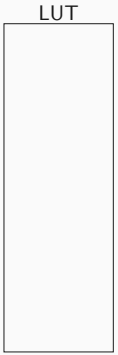

Pizza


LUT
index = 0;
char* data = "textKEY";
if (index < 4)
then
Prediction
else
LUT[data[index] * 4096]
0


LUT
index = 0;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
index = 0;
char* data = "textKEY";
if (index < 4)
then
else
Speculate
Prediction
LUT[data[index] * 4096]
0


LUT
Index 't'
index = 0;
char* data = "textKEY";
if (index < 4)
Execute
then
Prediction
else
LUT[data[index] * 4096]
0




```
index = 1;

char* data = "textKEY";

if (index < 4)
```

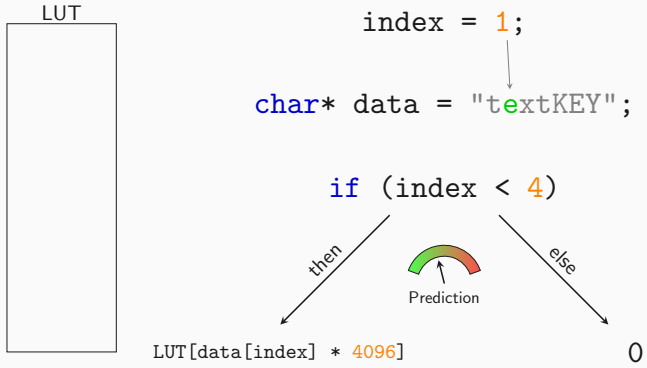
then

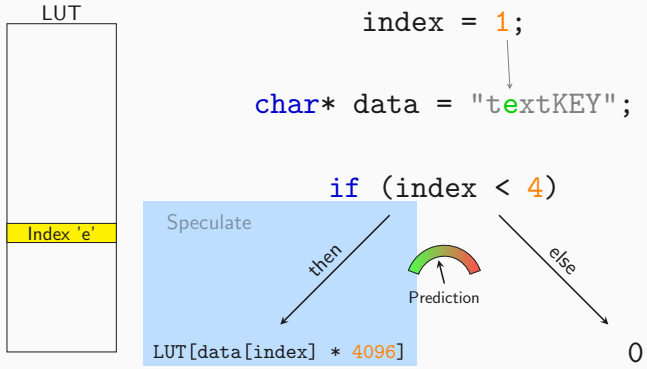
Prediction

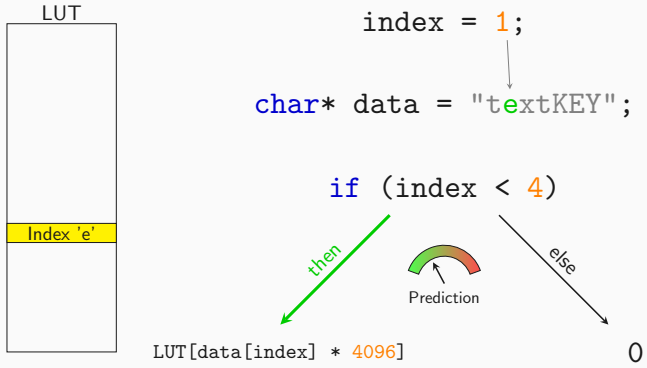
else

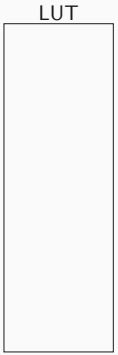
`LUT[data[index] * 4096]`

0


LUT
index = 1;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 'e'
index = 1;
char* data = "textKEY";
if (index < 4)
Speculate
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 'e'
index = 1;
char* data = "textKEY";
if (index < 4)
then
Prediction
else
LUT[data[index] * 4096]
0




```
index = 2;

char* data = "textKEY";

if (index < 4)
```

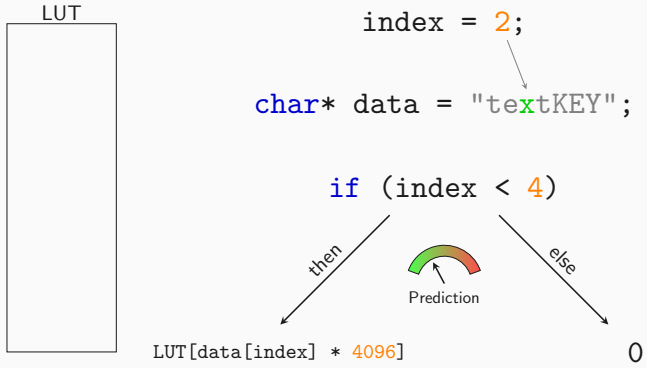
then

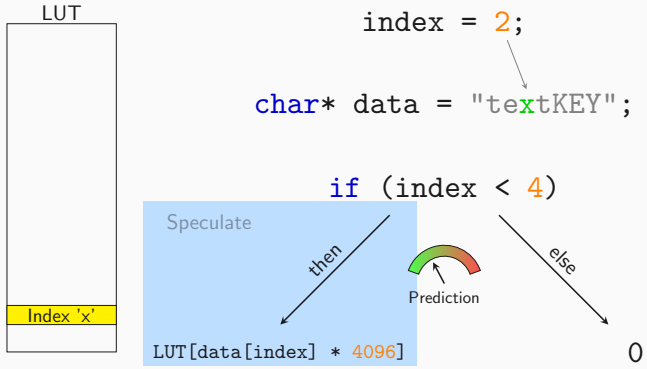
Prediction

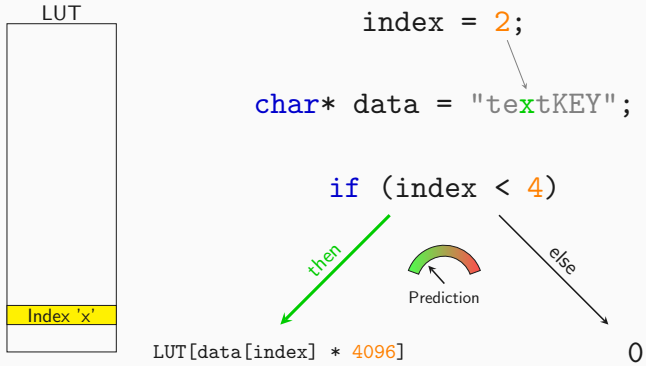
else

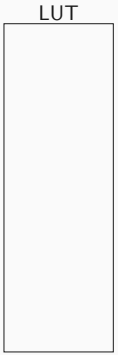
`LUT[data[index] * 4096]`

0


LUT
index = 2;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 'x'
index = 2;
char* data = "textKEY";
if (index < 4)
Speculate
then
Prediction
else
LUT[data[index] * 4096]
0


LUT
Index 'x'
index = 2;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
LUT[data[index] * 4096]
0




```
index = 3;

char* data = "textKEY";

if (index < 4)
```

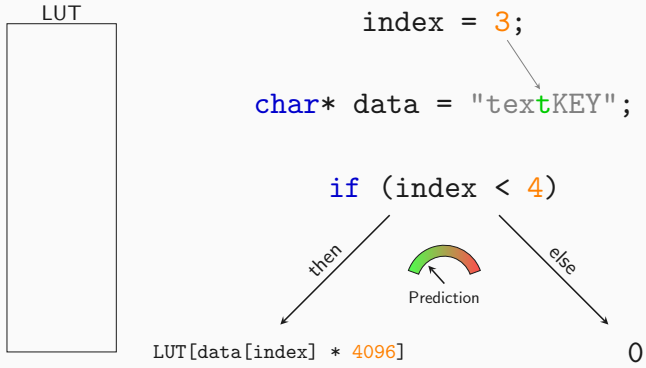
then

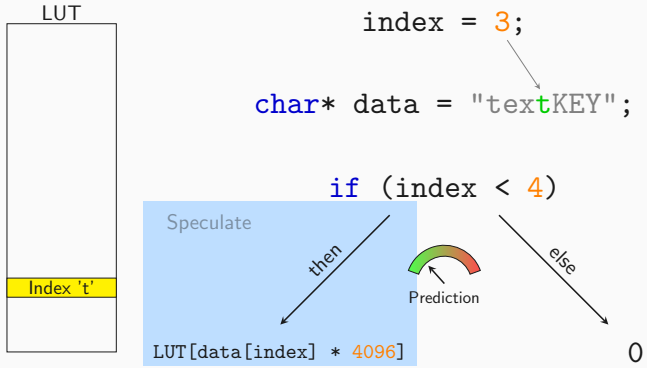
Prediction

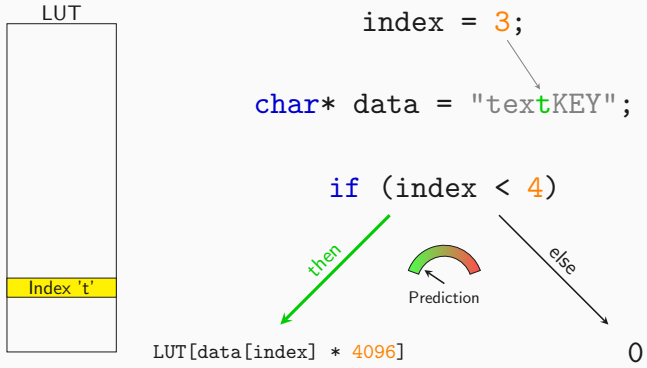
else

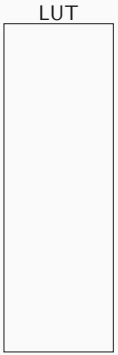
`LUT[data[index] * 4096]`

0


LUT
index = 3;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 't'
index = 3;
char* data = "textKEY";
if (index < 4)
Speculate
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 't'
index = 3;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
LUT[data[index] * 4096]
0




```
index = 4;

char* data = "textKEY";

if (index < 4)
```

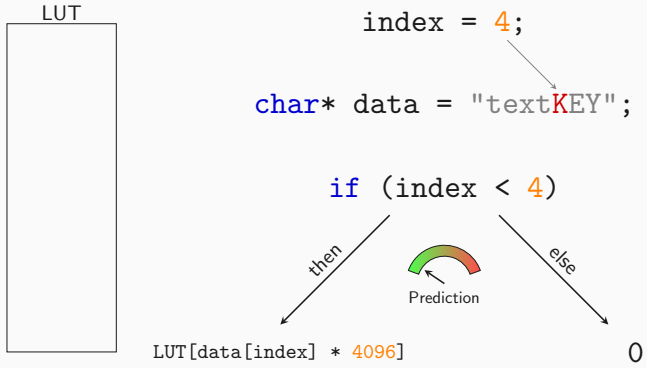
then

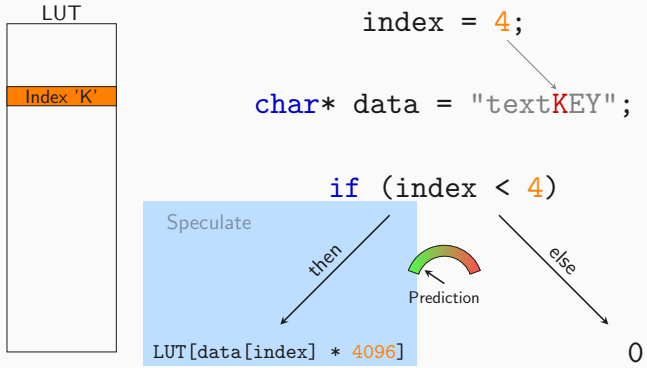
Prediction

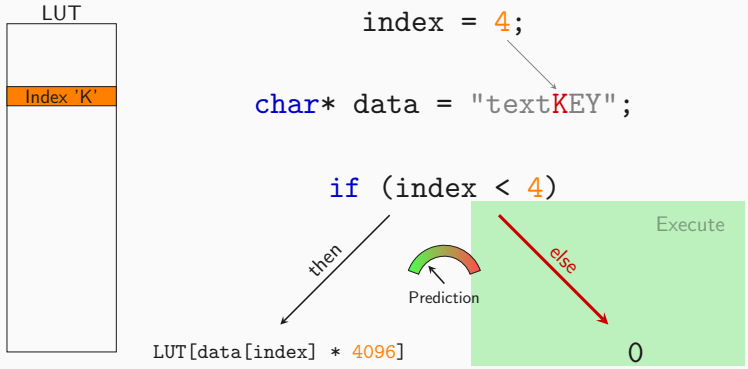
else

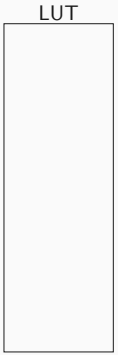
`LUT[data[index] * 4096]`

0


LUT
index = 4;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 'K'
index = 4;
char* data = "textKEY";
if (index < 4)
Speculate
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 'K'
index = 4;
char* data = "textKEY";
if (index < 4)
then
Prediction
else
Execute
LUT[data[index] * 4096]
0




```
index = 5;

char* data = "textKEY";

if (index < 4)
```

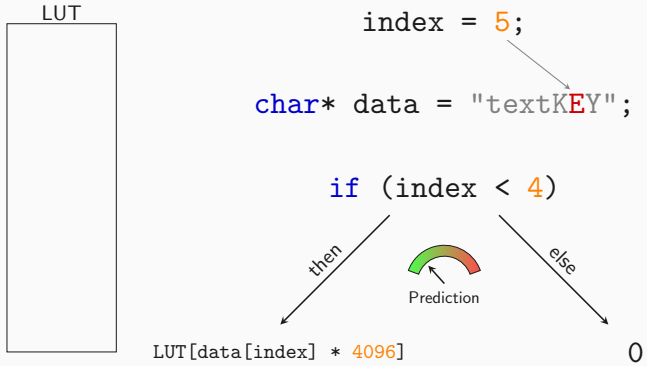
then

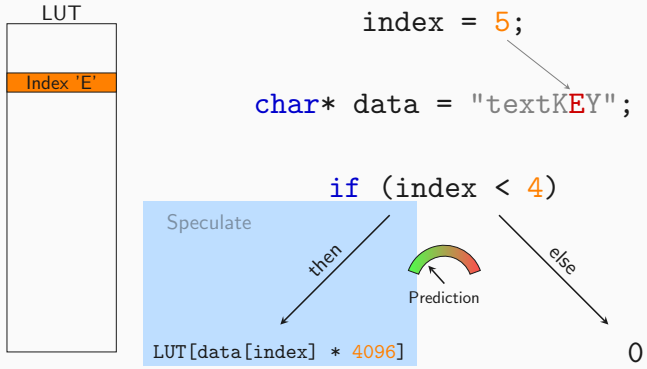
Prediction

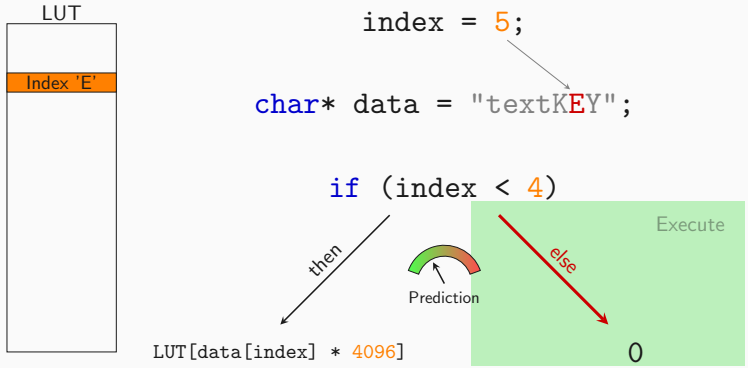
else

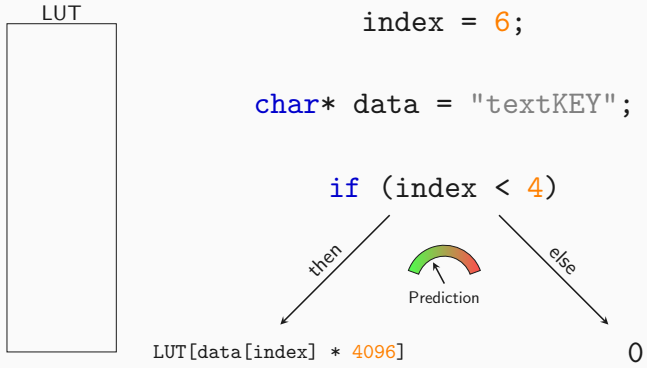
`LUT[data[index] * 4096]`

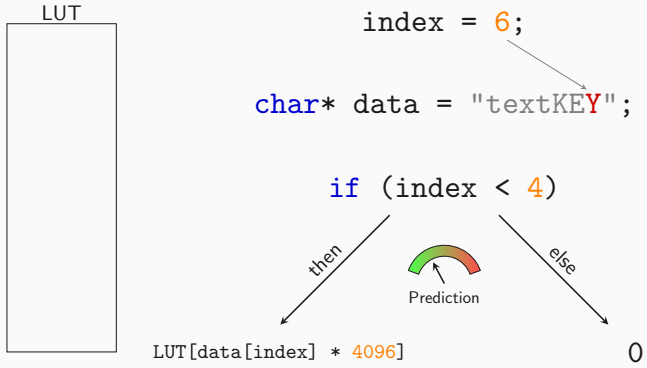
0

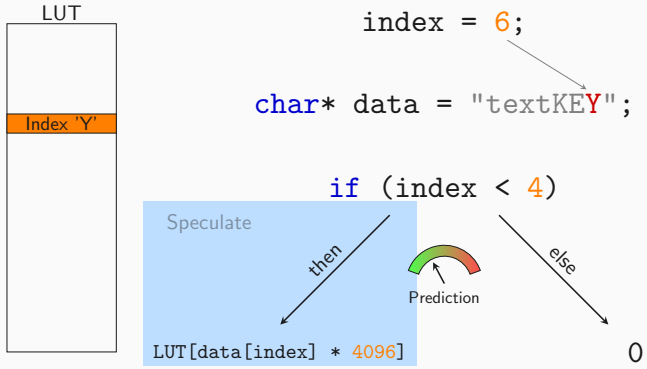

LUT
index = 5;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
LUT[data[index] * 4096]
0

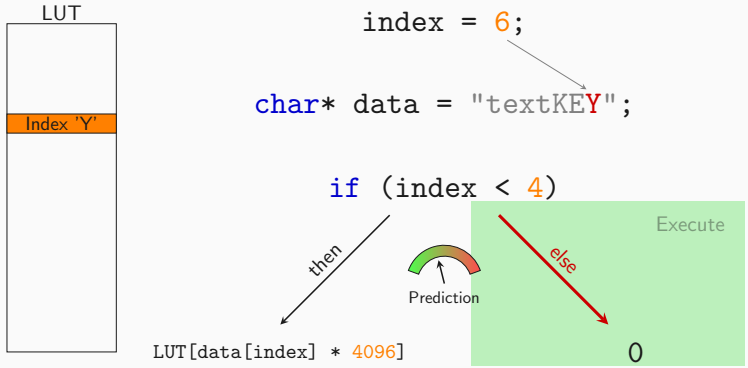

LUT
Index 'E'
index = 5;
char* data = "textKEY";
if (index < 4)
Speculate
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 'E'
index = 5;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
Execute
LUT[data[index] * 4096]
0

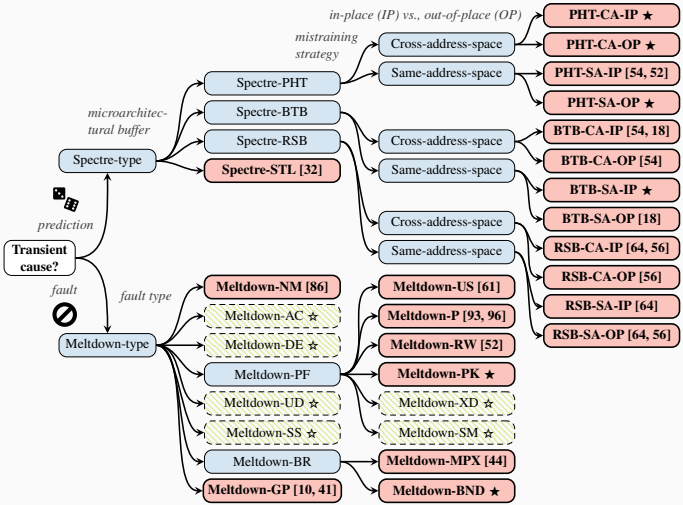

LUT
index = 6;
char* data = "textKEY";
if (index < 4)
then
Prediction
else
LUT[data[index] * 4096]
0


LUT
index = 6;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 'Y'
index = 6;
char* data = "textKEY";
if (index < 4)
Speculate
then
else
Prediction
LUT[data[index] * 4096]
0


LUT
Index 'Y'
index = 6;
char* data = "textKEY";
if (index < 4)
then
else
Prediction
Execute
LUT[data[index] * 4096]
0

Classification Tree



Mitigations?


BLOCKCHAIN
HD
HISTORY.COM

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

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

Attack		InvisiSpec	SafeSpec	DAWG	RSB Stuffing	Repoline	Poison Value	Index Masking	Site Isolation	YSNB	IBRS	STIPB	IBPB	Serialization	Taint Tracking	Timer Reduction	Sloth	SSBD/SSBB
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
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%	SYSmark®2014 SE & SPEC integer
KAISER/KPTI	0–2.6%	system call rates
L1TF mitigations	-3–31%	various SPEC


FINALLY THE RIGHT SPEED FOR ME
imgflip.com

How to find the next big thing ;)

they become the target of the doll-maker's possessed creation, Annabelle.

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, Tallahasse, 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–)

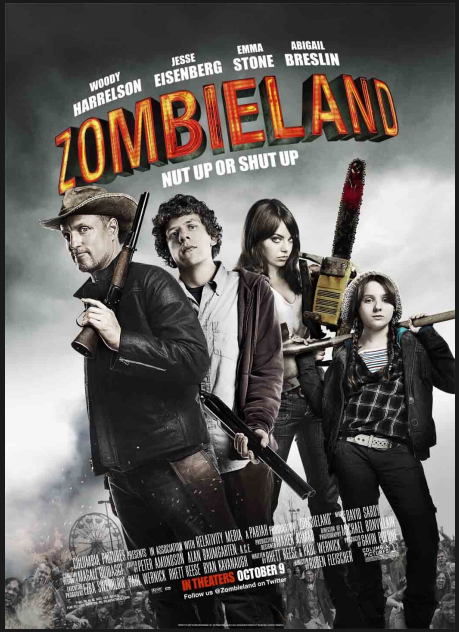
TV-14 | 42 min | Comedy, Crime, Drama

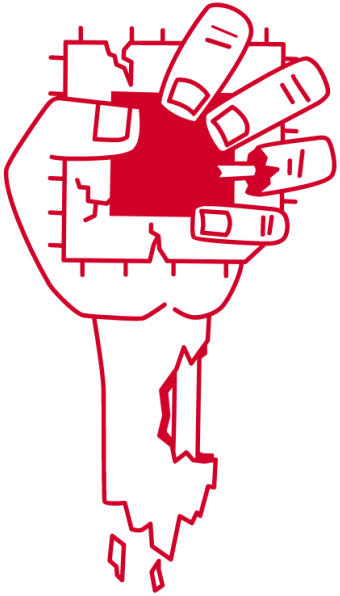
7.9 Rate this

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

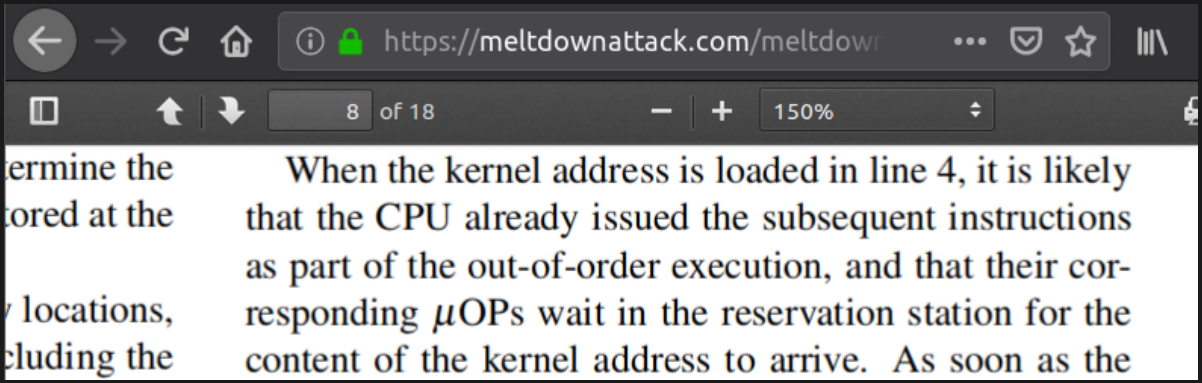
Stars: Rose McIver, Malcolm Goodwin, Rahul Kohli, Robert Buckley

Votes: 54,215


WOODY HARRELSON
JESSE EISENBERG
EMMA STONE
ABIGAIL BRESLIN
ZOMBIELAND
NUT UP OR SHUT UP
IN THEATERS OCTOBER 9
Follow us @Zombieland on Twitter



ZombieLoad Attack


https://meltdownattack.com/meltdowr
8 of 18
150%
ermine the
tored at the
locations,
cluding the
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 μOPs 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

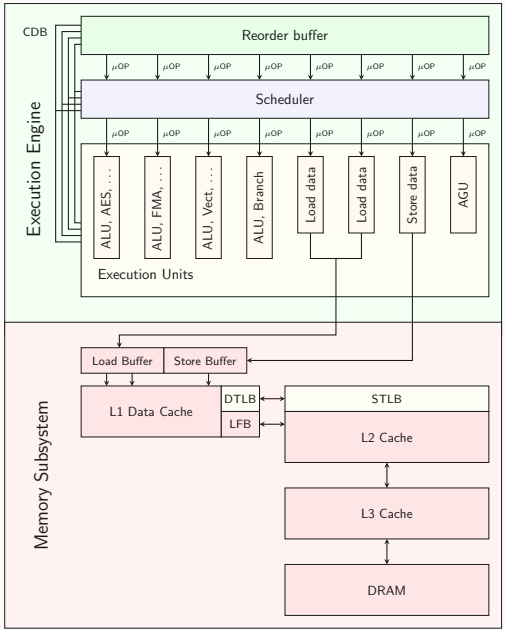
19.05.2007 - by CaptBilly1Eye · 12 years ago In reply to Toshiba Boot **Error** ... partition on the floppy disk, hard drive or a CD ROM to **load** the **operating** system. ... prior to this situation starting to **occur**, or if you find that the boot sequence already has the ... Leave the notebook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

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

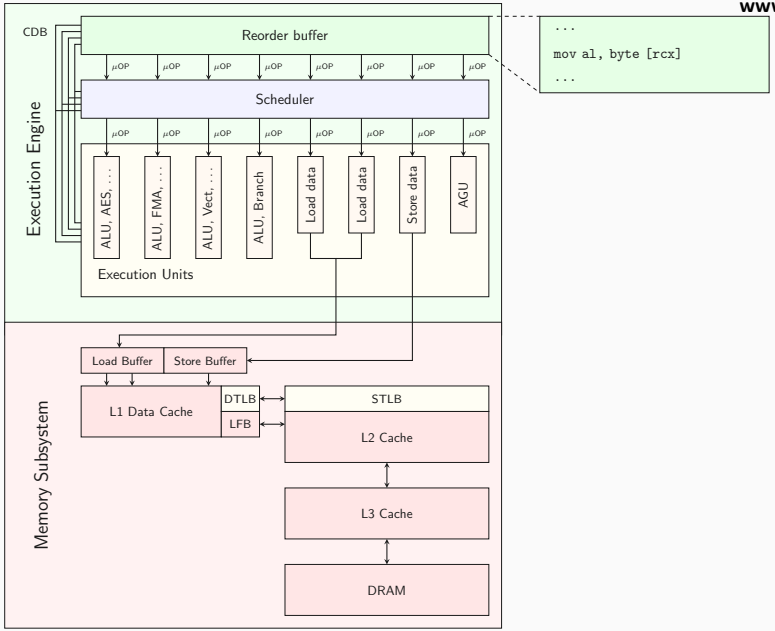
www.google.com/patents/US5751983 - Diese Seite übersetzen

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**.

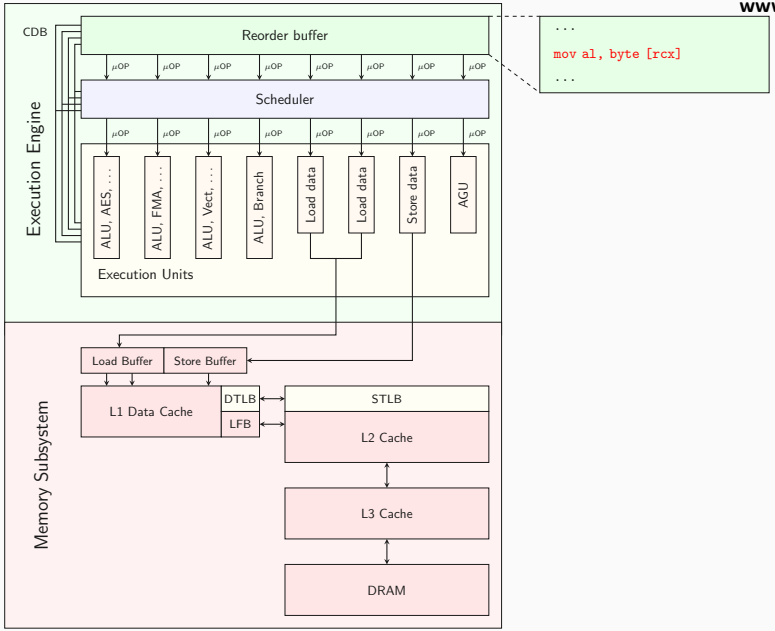
Meltdown


CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

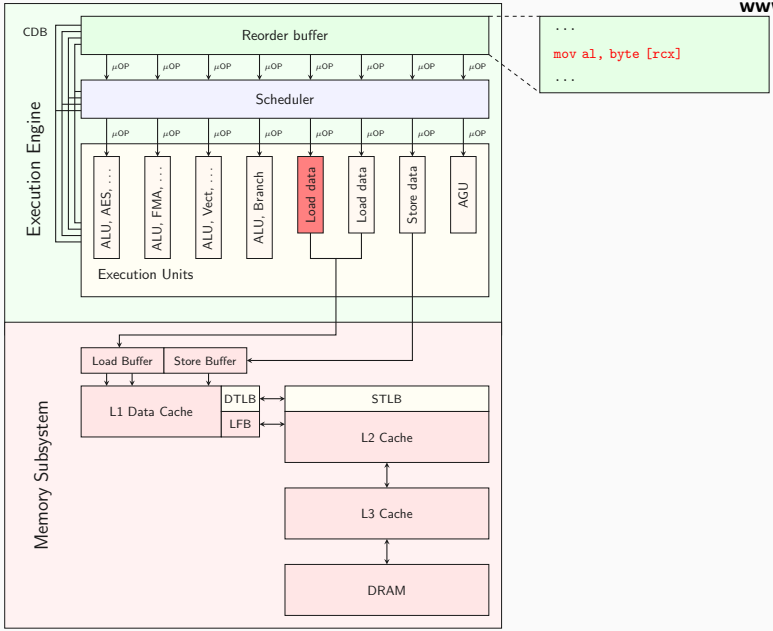
Meltdown


...
mov al, byte [rcx]
...
CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

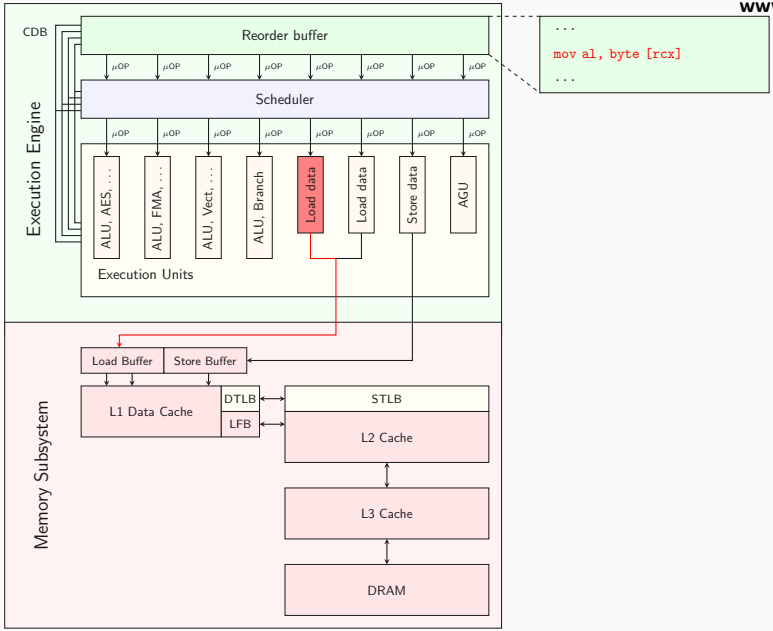
Meltdown


...
mov al, byte [rcx]
...
CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

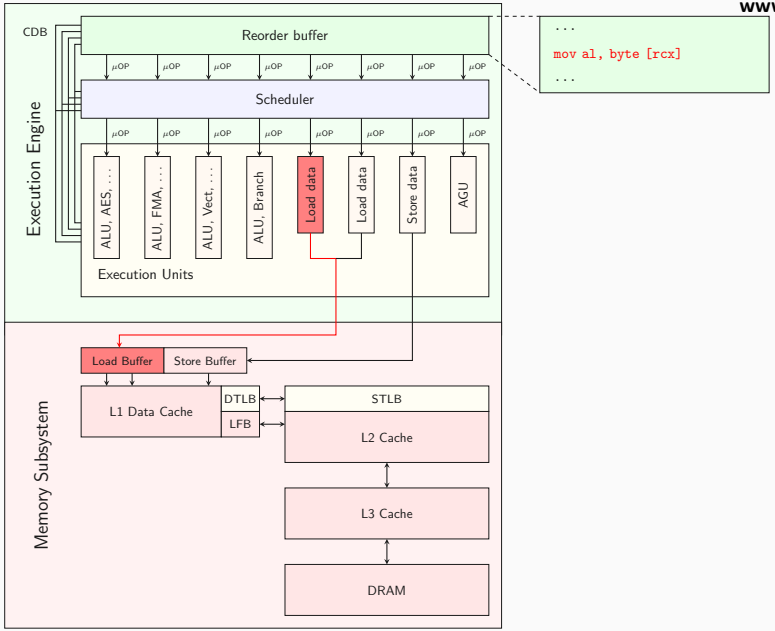
Meltdown


...
mov al, byte [rcx]
...
CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

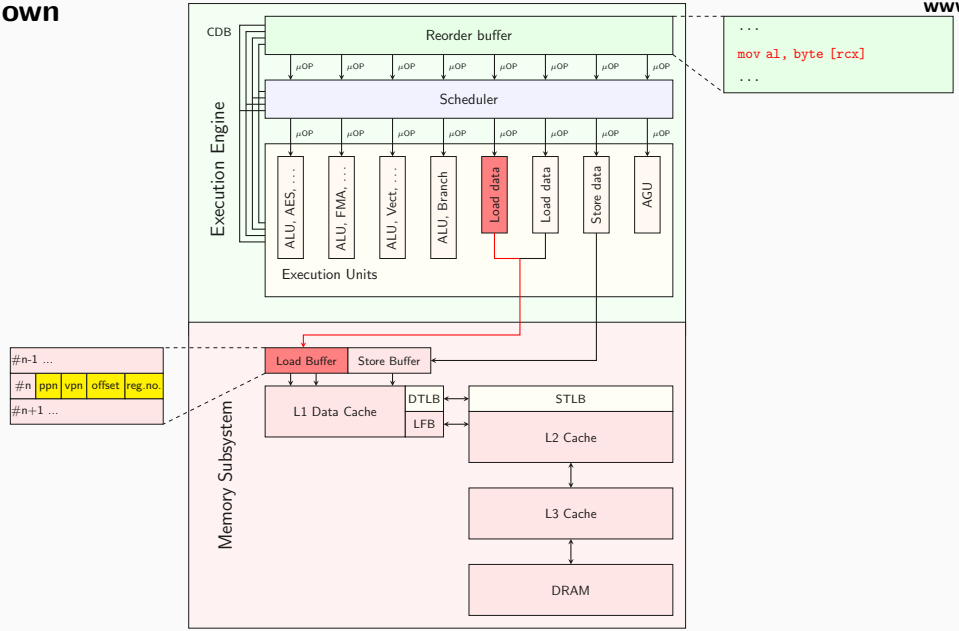
Meltdown


...
mov al, byte [rcx]
...
CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

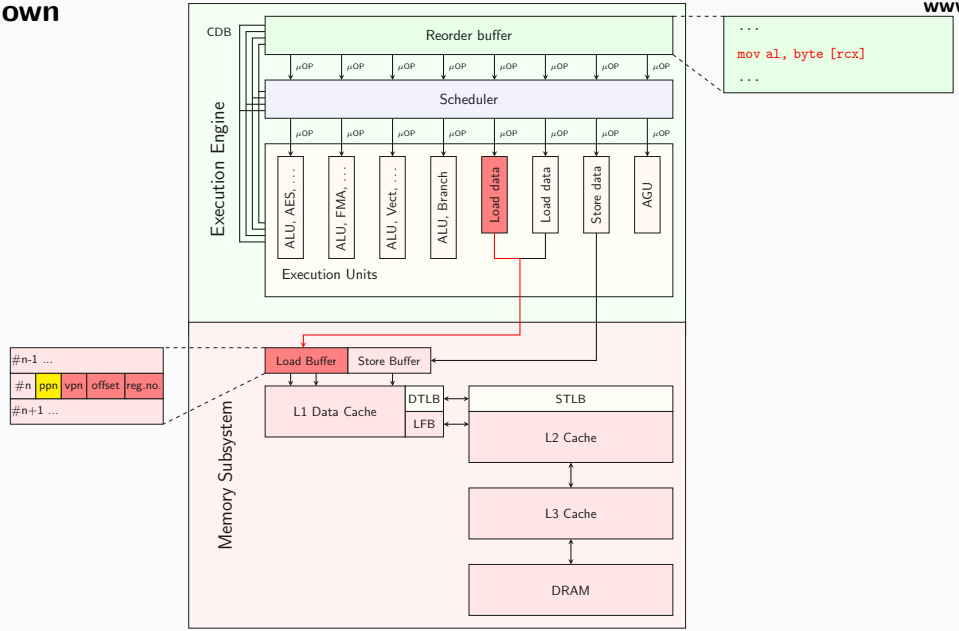
Meltdown


...
mov al, byte [rcx]
...
CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

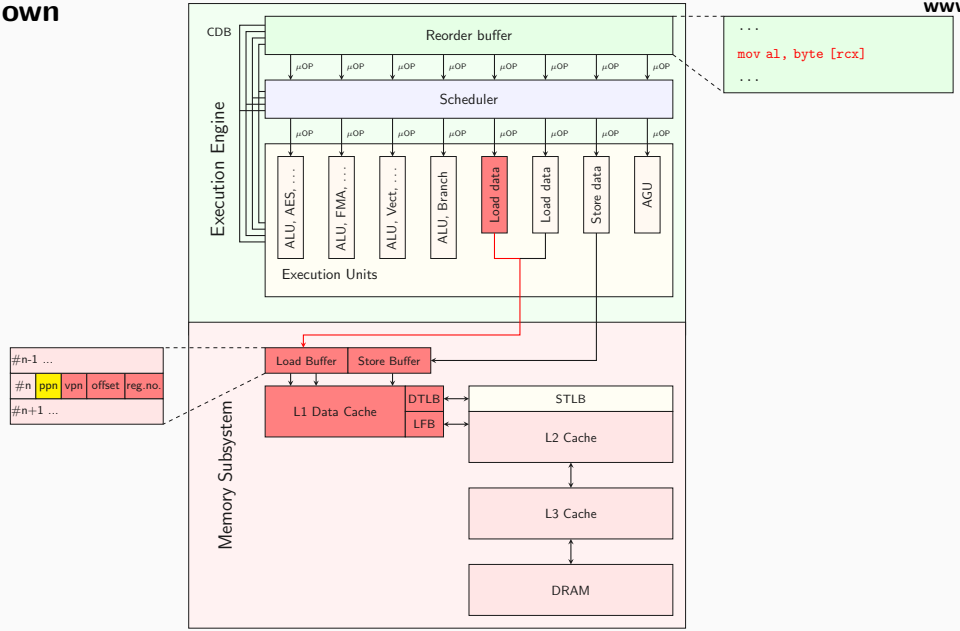
Meltdown


CDB
Reorder buffer
...
mov al, byte [rcx]
...
µOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
#n-1 ...
#n
ppn
vpn
offset
reg. no.
#n+1 ...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

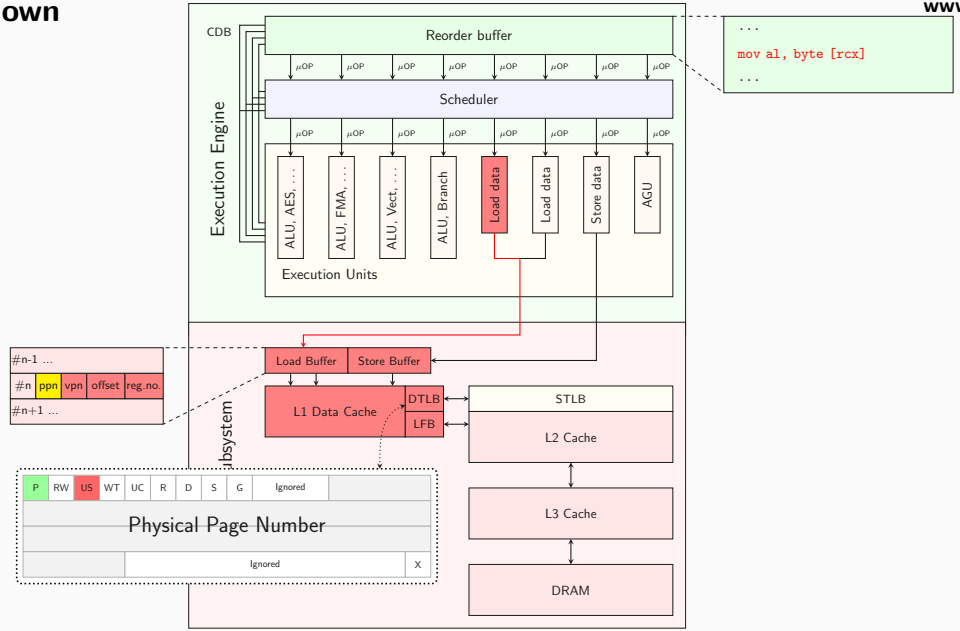
Meltdown


CDB
Reorder buffer
...
mov al, byte [rcx]
...
µOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
#n-1 ...
#n
ppn
vpn
offset
reg.no.
#n+1 ...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

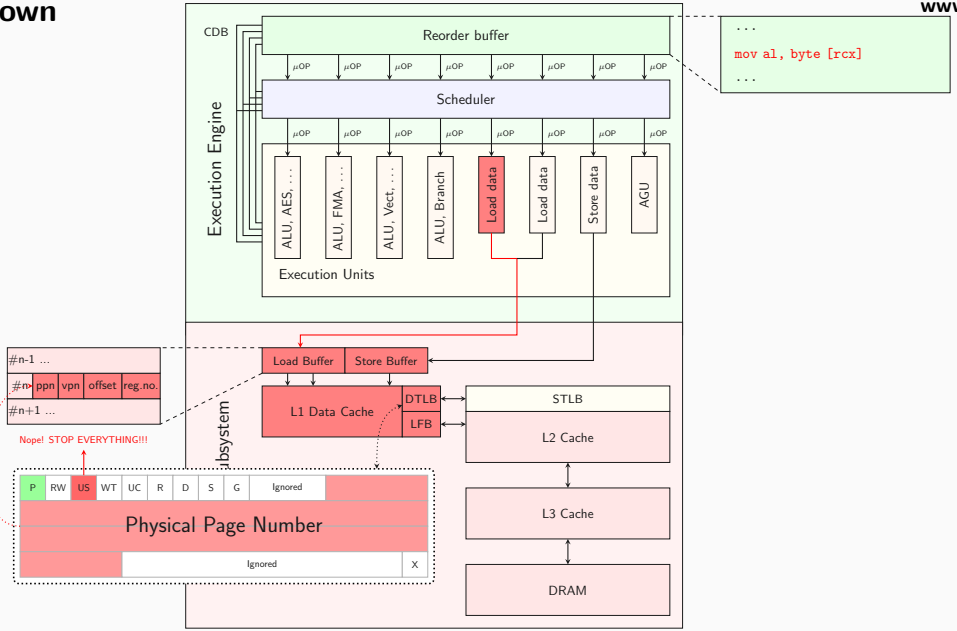
Meltdown


...
mov al, byte [rcx]
...
CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
#n-1 ...
#n
ppn
vpn
offset
reg.no.
#n+1 ...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

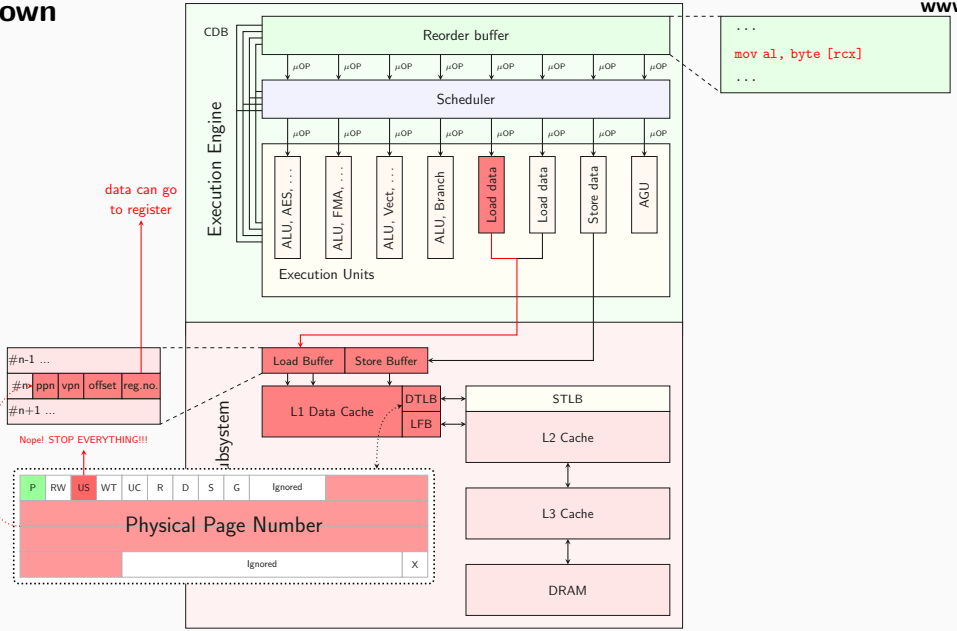
Meltdown


CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
...
mov al, byte [rcx]
...
#n-1 ...
#n ppn vpn offset reg.no.
#n+1 ...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
ubsystem
P RW US WT UC R D S G Ignored
Physical Page Number
Ignored X

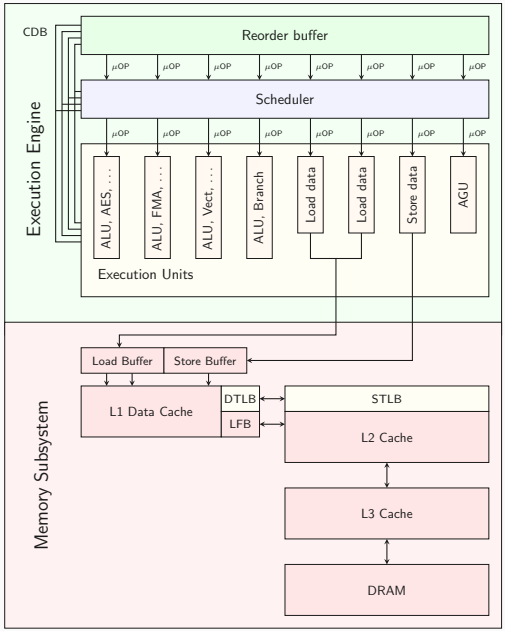
Meltdown


CDB
Reorder buffer
µOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
...
mov al, byte [rcx]
...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
#n-1 ...
#n ppn vpn offset reg.no.
#n+1 ...
Nope! STOP EVERYTHING!!!
P RW US WT UC R D S G Ignored
Physical Page Number
Ignored X

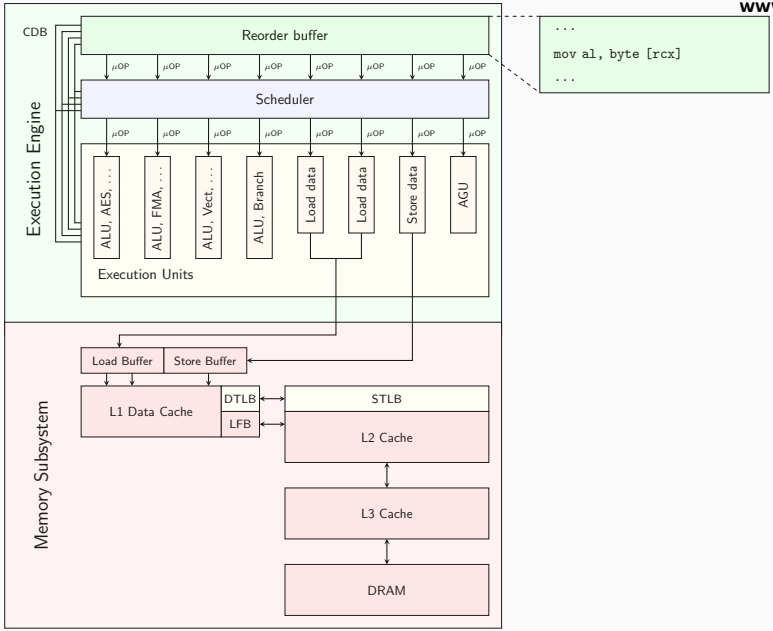
Meltdown


...
mov al, byte [rcx]
...
CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
data can go to register
#n-1 ...
#n ppn vpn offset reg.no.
#n+1 ...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Nope! STOP EVERYTHING!!!
P RW US WT UC R D S G Ignored
Physical Page Number
Ignored X

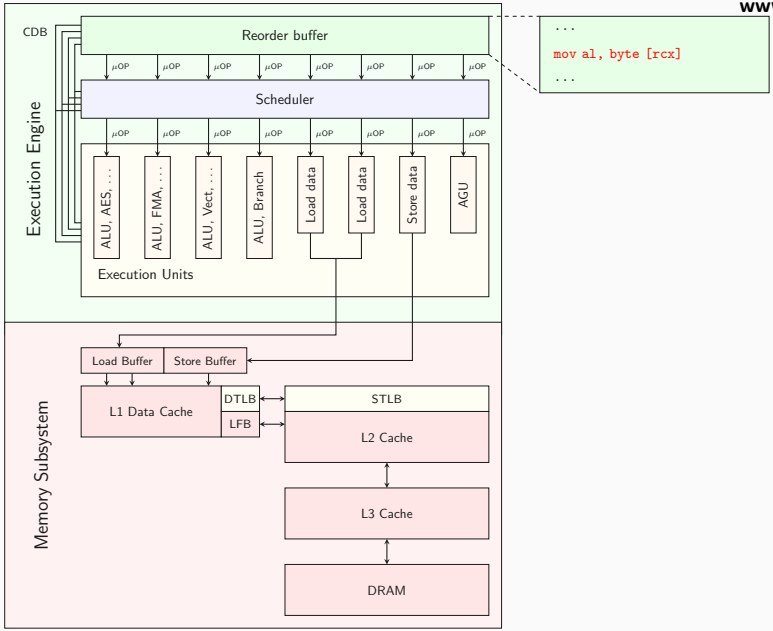
Foreshadow-VMM



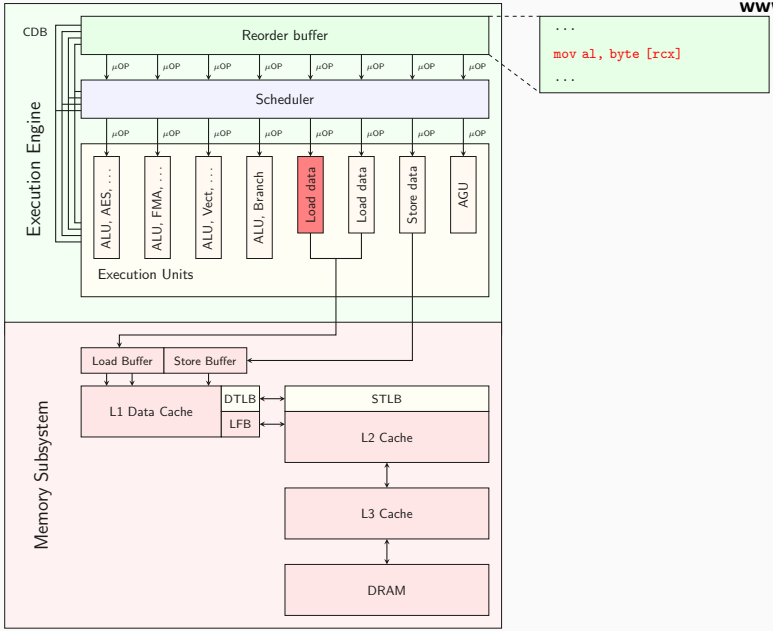
Foreshadow-VMM



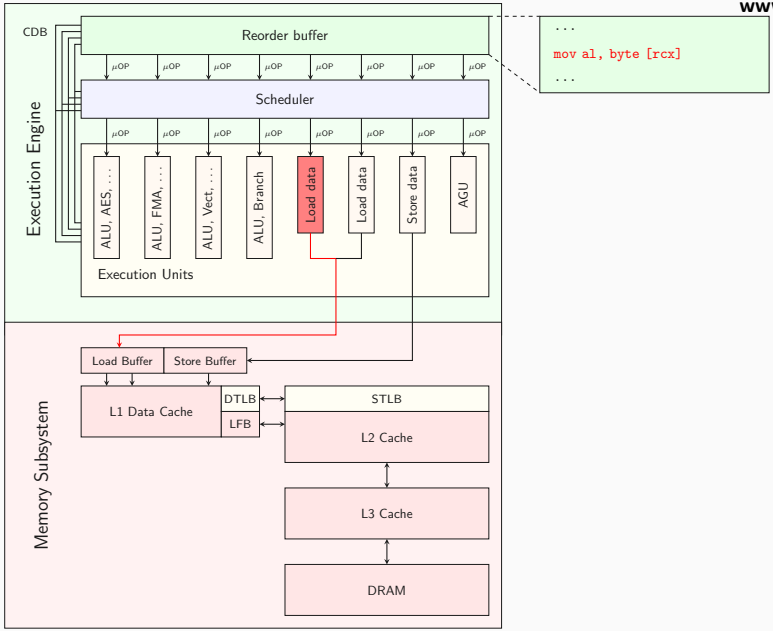
Foreshadow-VMM



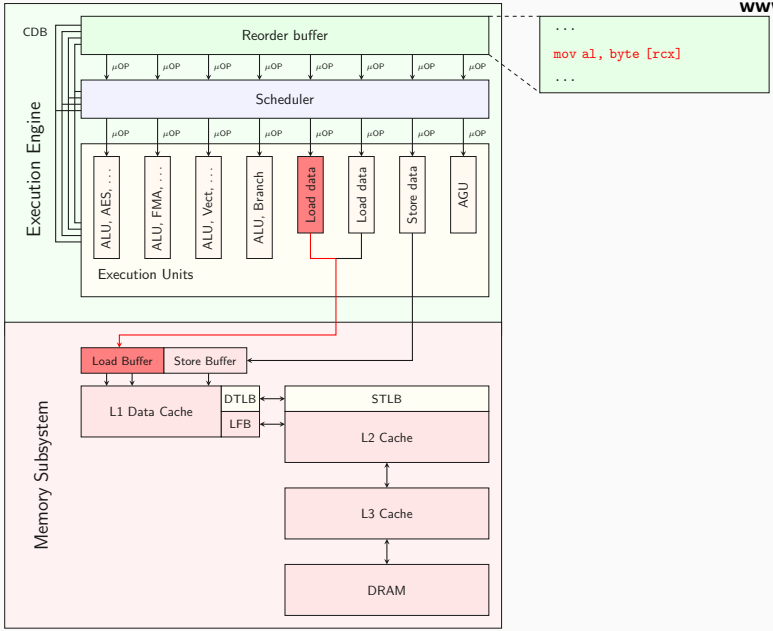
Foreshadow-VMM



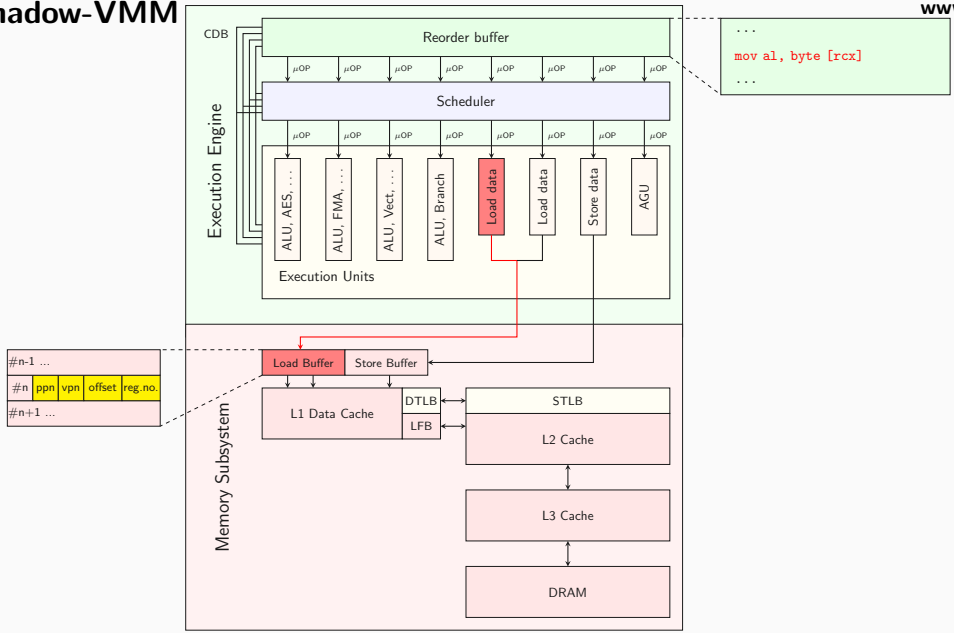
Foreshadow-VMM



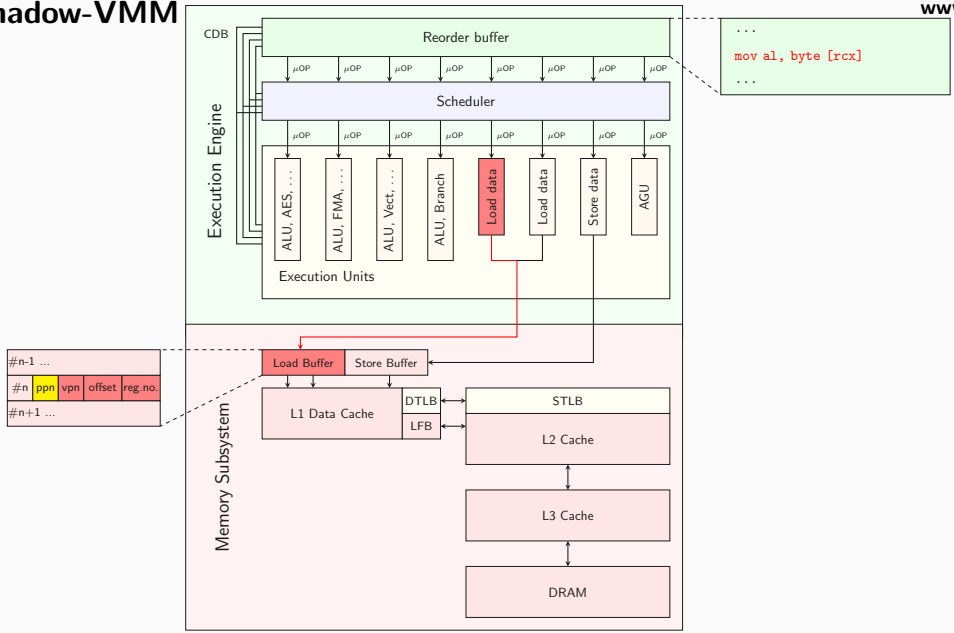
Foreshadow-VMM



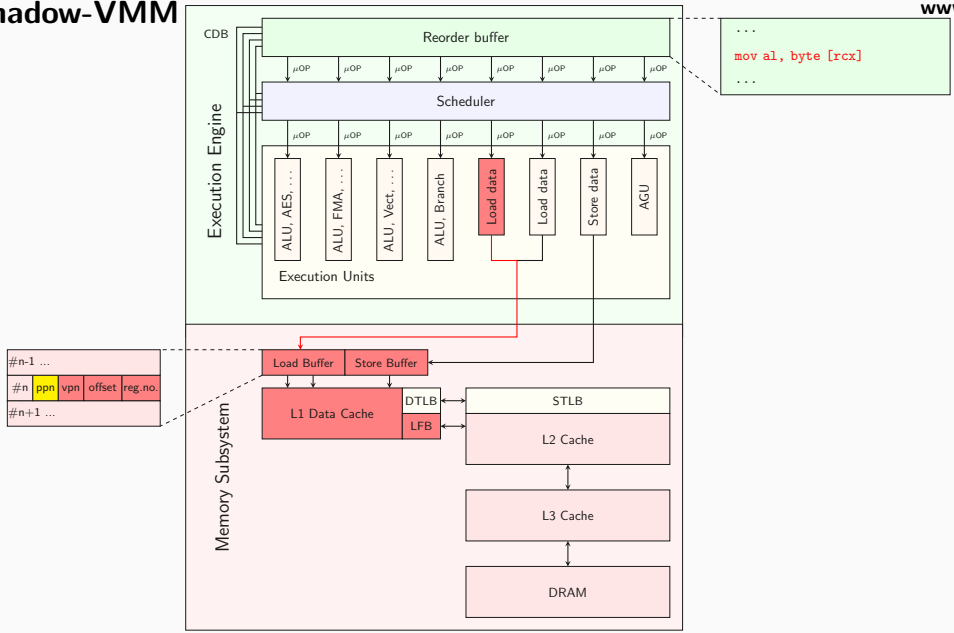
Foreshadow-VMM



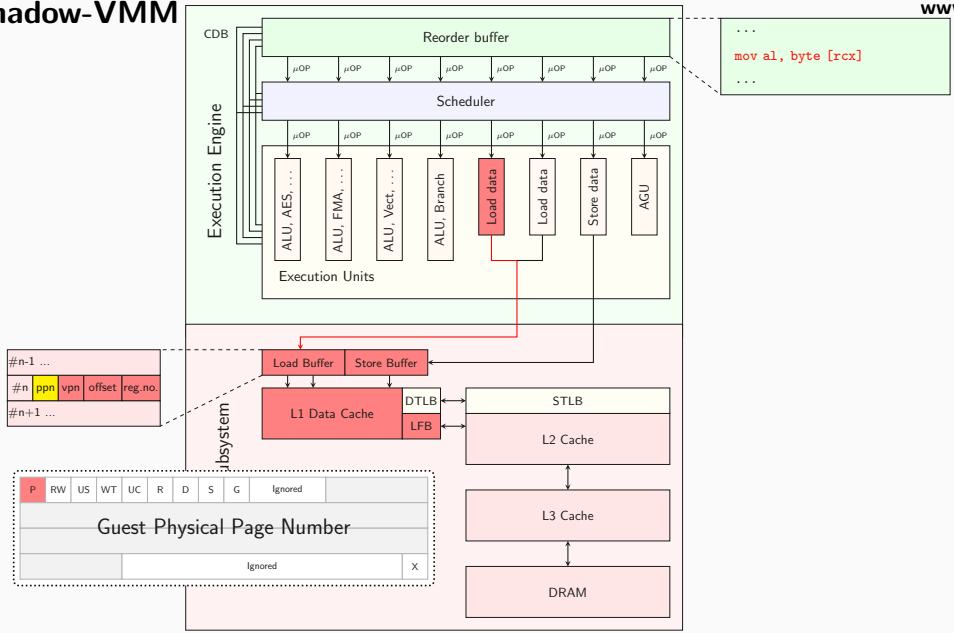
Foreshadow-VMM



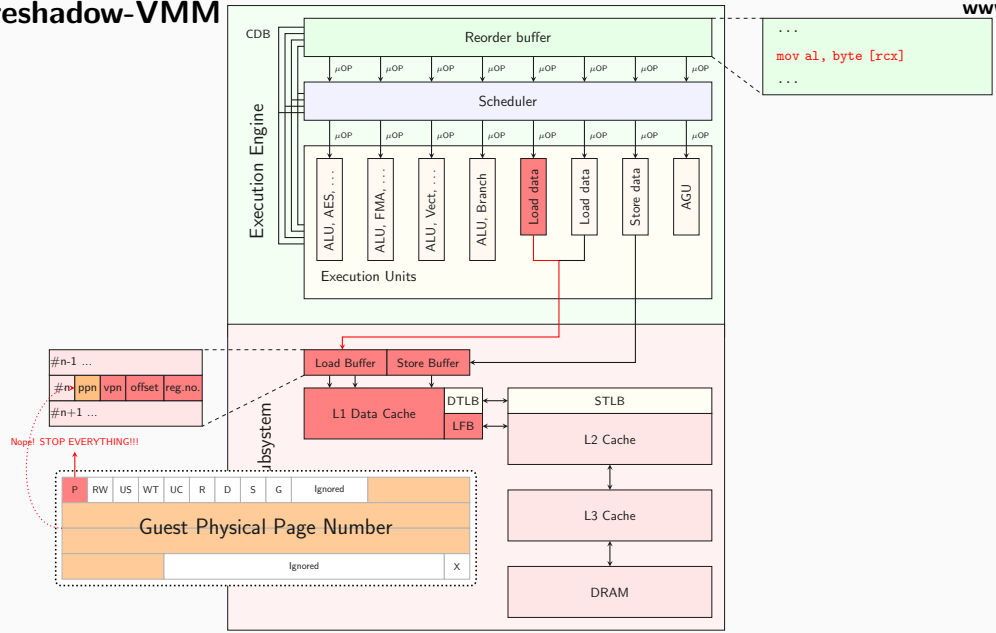
Foreshadow-VMM



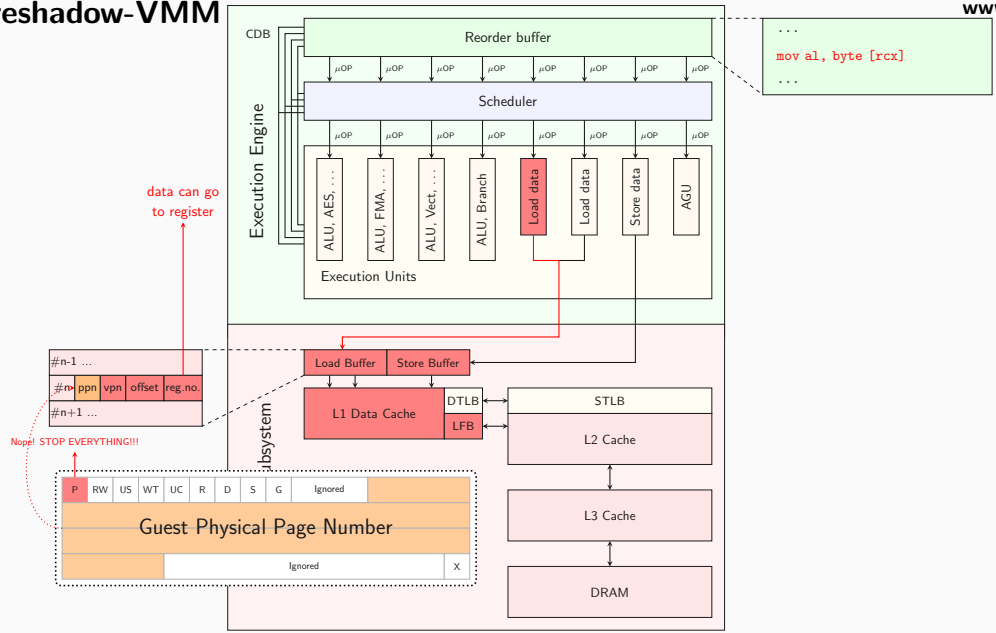
Foreshadow-VMM

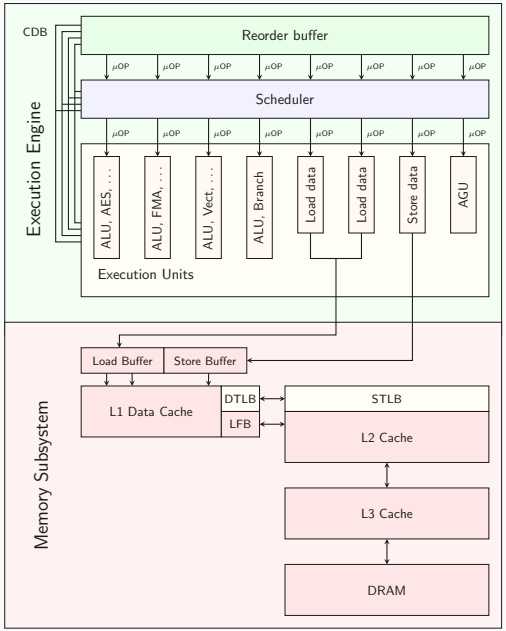


Foreshadow-VMM

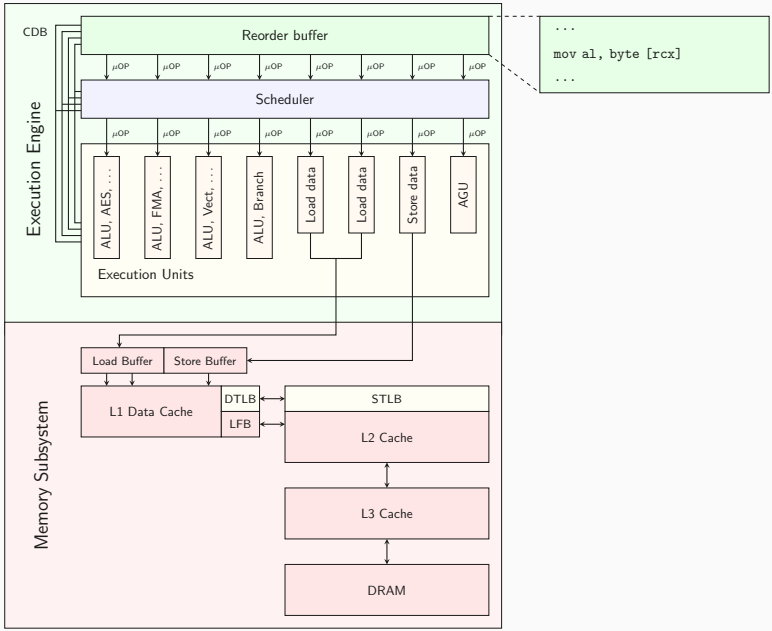


Foreshadow-VMM

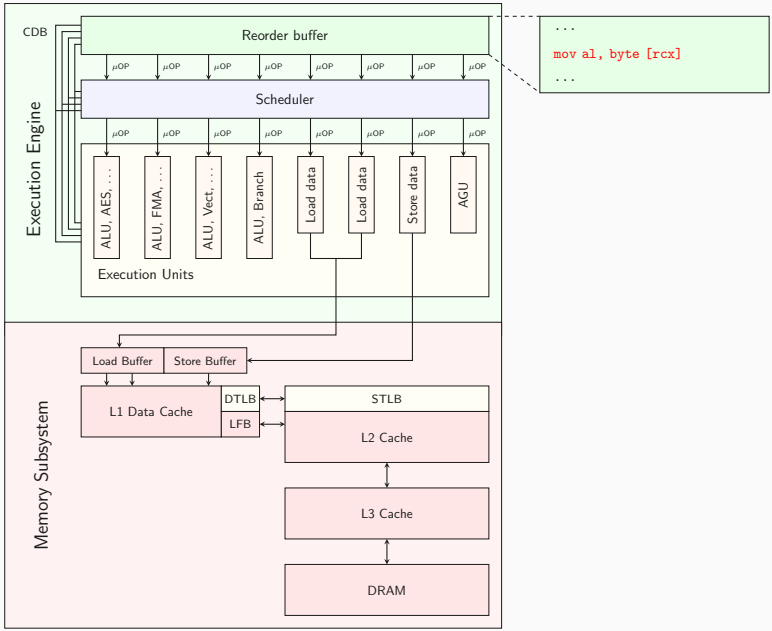



CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

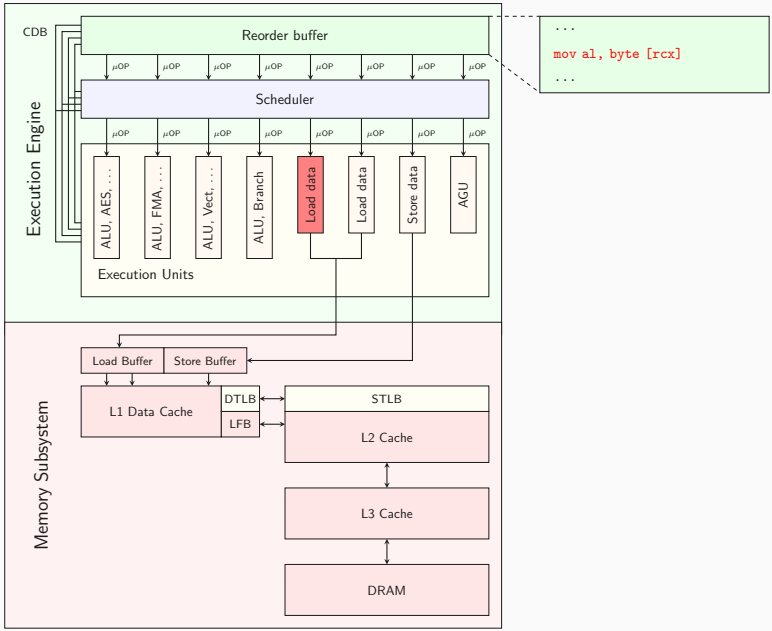
ZombieLoad


CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
...
mov al, byte [rcx]
...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

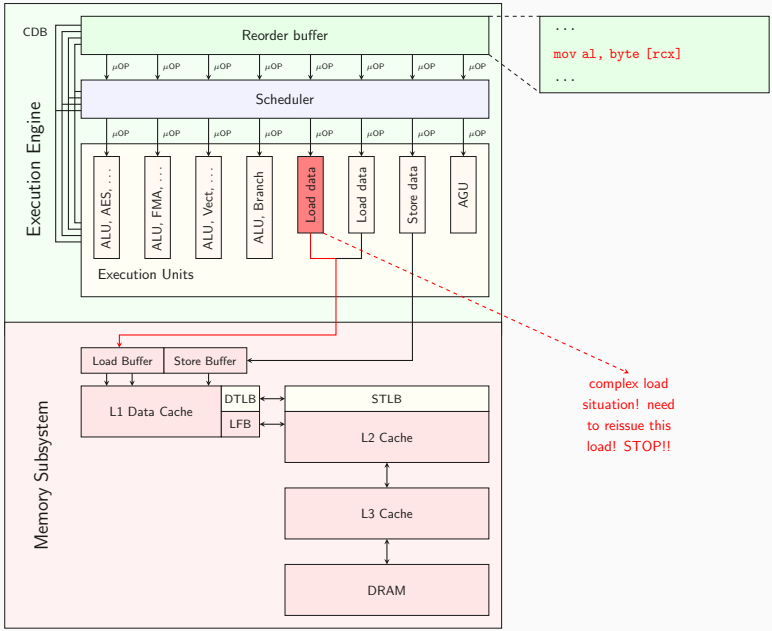
ZombieLoad


CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
...
mov al, byte [rcx]
...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

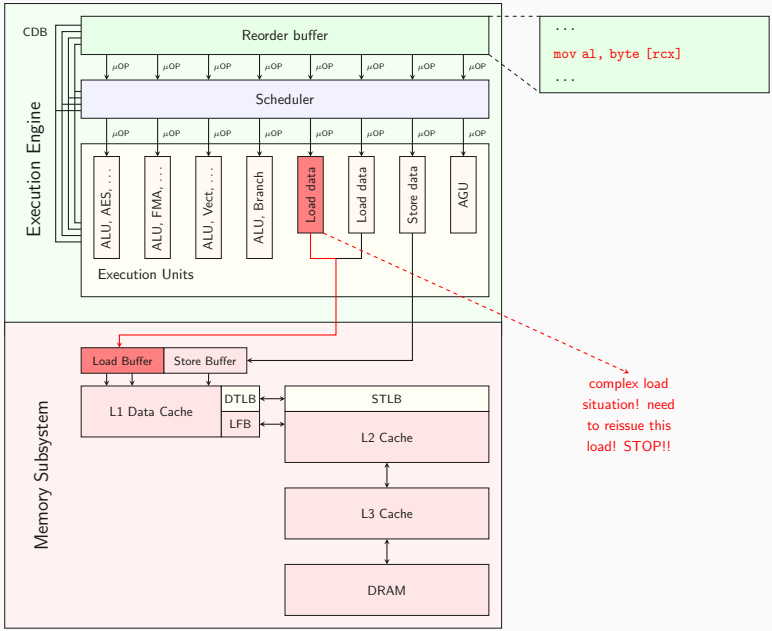
ZombieLoad


CDB
Reorder buffer
...
mov al, byte [rcx]
...
µOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem

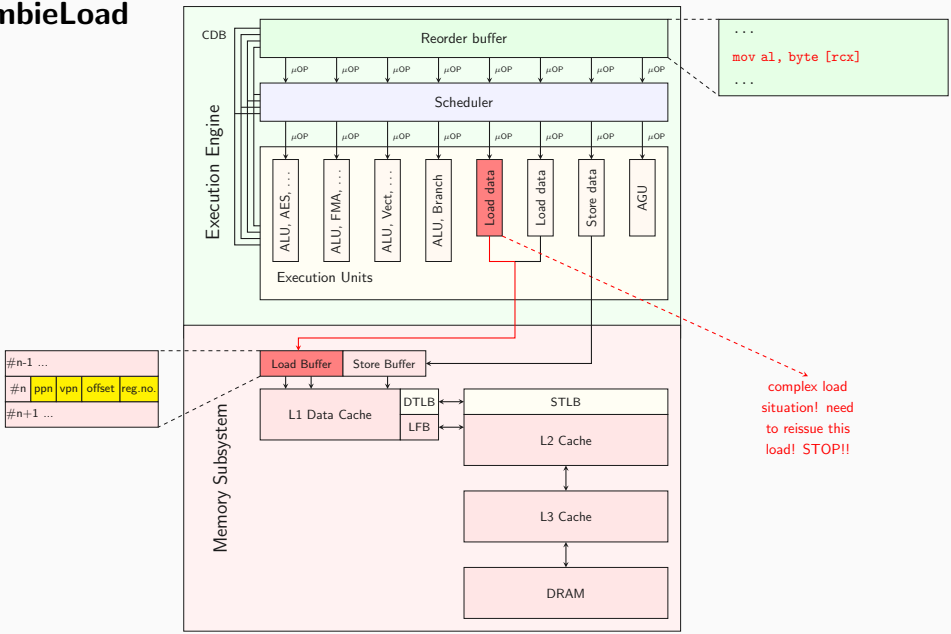
ZombieLoad


CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
...
mov al, byte [rcx]
...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem
complex load situation! need to reissue this load! STOP!!

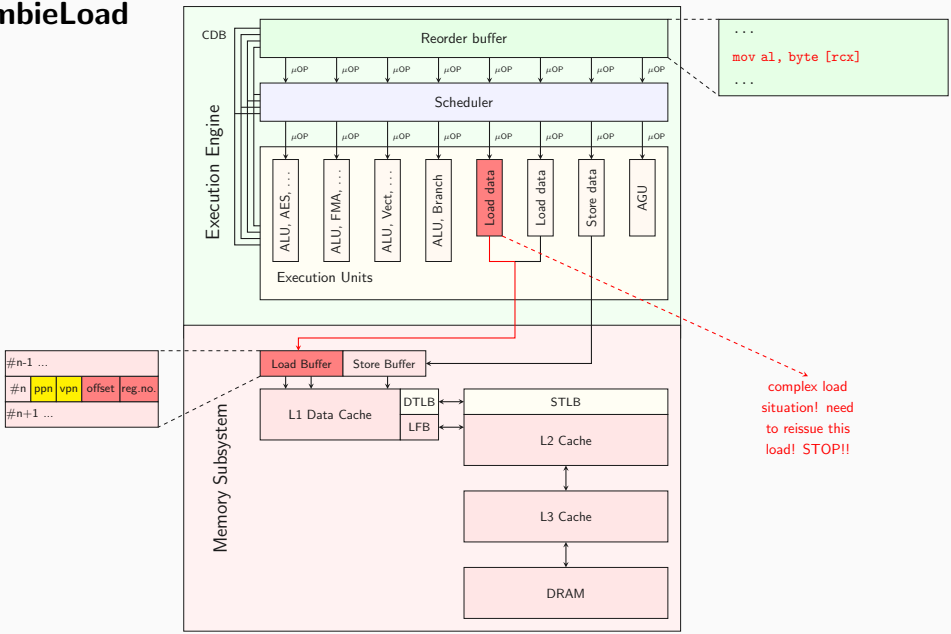
ZombieLoad


CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
...
mov al, byte [rcx]
...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem
complex load situation! need to reissue this load! STOP!!

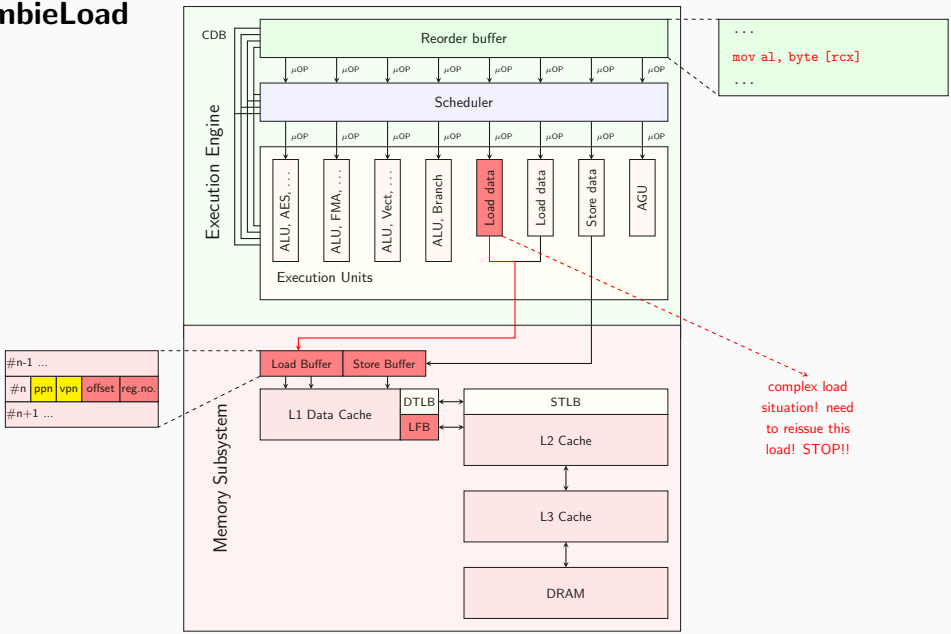
ZombieLoad


CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
...
mov al, byte [rcx]
...
#n-1 ...
#n
ppn
vpn
offset
reg.no.
#n+1 ...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem
complex load situation! need to reissue this load! STOP!!

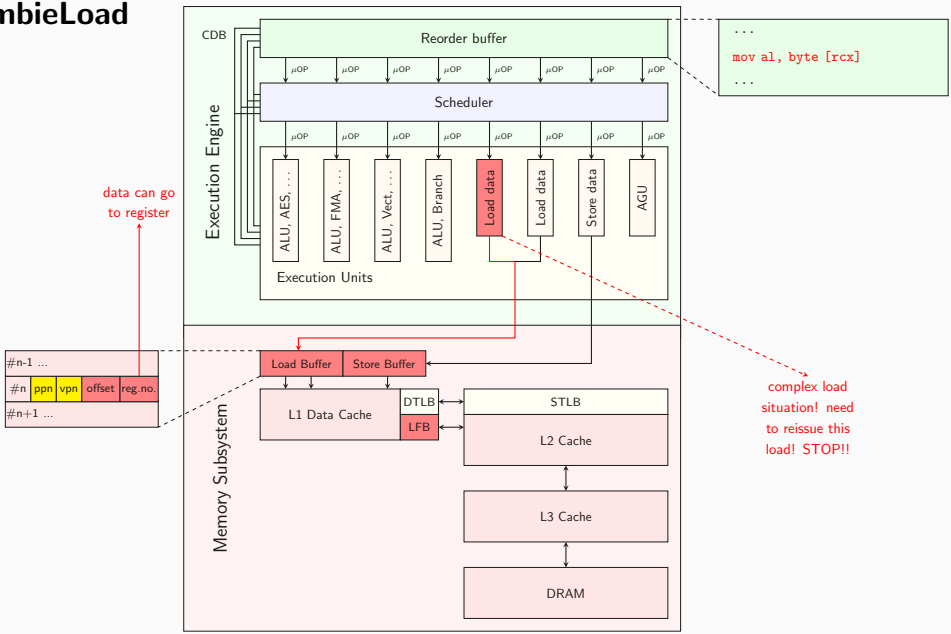
ZombieLoad


CDB
Reorder buffer
...
mov al, byte [rcx]
...
µOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
#n-1 ...
#n
ppn
vpn
offset
reg.no.
#n+1 ...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem
complex load situation! need to reissue this load! STOP!!

ZombieLoad


...
mov al, byte [rcx]
...
CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
#n-1 ...
#n
ppn
vpn
offset
reg.no.
#n+1 ...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem
complex load situation! need to reissue this load! STOP!!

ZombieLoad


...
mov al, byte [rcx]
...
CDB
Reorder buffer
μOP
Scheduler
Execution Engine
ALU, AES, ...
ALU, FMA, ...
ALU, Vect, ...
ALU, Branch
Load data
Load data
Store data
AGU
Execution Units
data can go to register
#n-1 ...
#n
ppn
vpn
offset
reg.no.
#n+1 ...
Load Buffer
Store Buffer
L1 Data Cache
DTLB
LFB
STLB
L2 Cache
L3 Cache
DRAM
Memory Subsystem
complex load situation! need to reissue this load! STOP!!

So how did we find it?

So how did we find it?

So how did we find it?

So how did we find it?

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

So how did we find it?

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

So how did we find it?

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

So how did we find it?

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

So how did we find it?

- 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

So how did we find it?

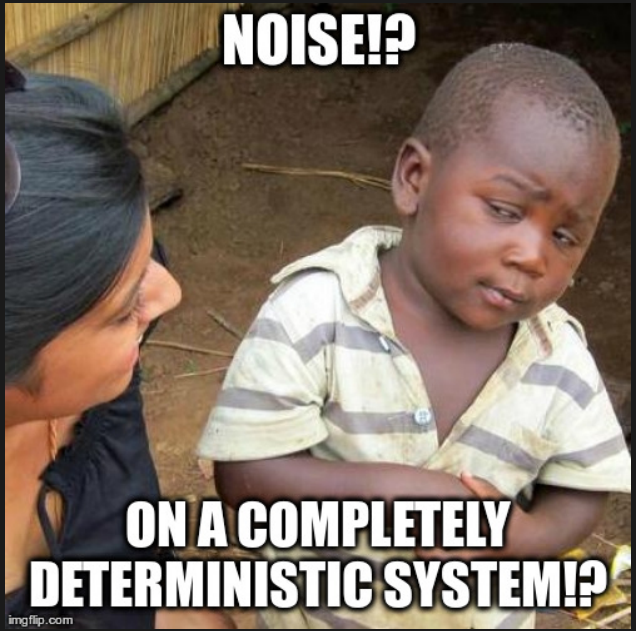
- our Meltdown PoC always worked on non-L1 memory (for us)
  - co-authors confirmed
- PoCs/reports → Intel - December 2017
  - "can't reproduce"
- works with uncacheable
  - PoC → Intel - March 2018

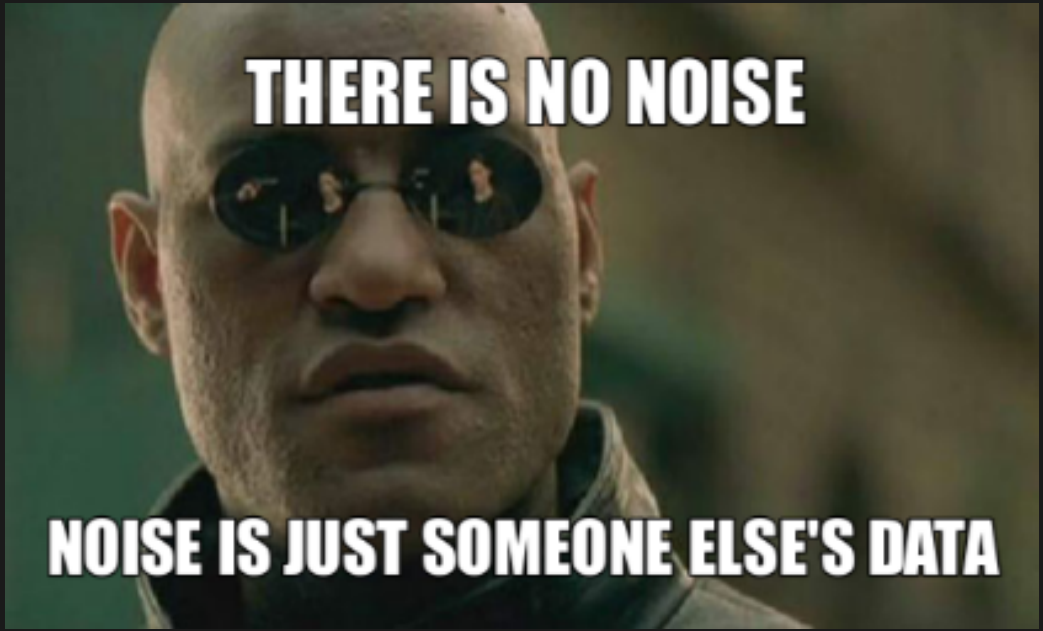
So how did we find it?

- 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 $\rightarrow$ lower signal to noise ratio


NOISE!?
ON A COMPLETELY DETERMINISTIC SYSTEM!?
imgflip.com


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]

Why attack research?

We have ignored microarchitectural attacks for many years:

We have ignored microarchitectural attacks for many years:

- attacks on crypto

We have ignored microarchitectural attacks for many years:

- attacks on crypto $\rightarrow$ "software should be fixed"

We have ignored microarchitectural attacks for many years:

- attacks on crypto → "software should be fixed"
- attacks on ASLR

We have ignored microarchitectural attacks for many years:

- attacks on crypto → "software should be fixed"
- attacks on ASLR → "ASLR is broken anyway"

We have ignored microarchitectural attacks for many years:

- attacks on crypto → "software should be fixed"
- attacks on ASLR → "ASLR is broken anyway"
- attacks on SGX and TrustZone

We have ignored microarchitectural attacks for many years:

- attacks on crypto → "software should be fixed"
- attacks on ASLR → "ASLR is broken anyway"
- attacks on SGX and TrustZone → "not part of the threat model"

We have ignored microarchitectural attacks for many years:

- attacks on crypto → "software should be fixed"
- attacks on ASLR → "ASLR is broken anyway"
- attacks on SGX and TrustZone → "not part of the threat model"
- Rowhammer

We have ignored microarchitectural attacks for many years:

- attacks on crypto → "software should be fixed"
- attacks on ASLR → "ASLR is broken anyway"
- attacks on SGX and TrustZone → "not part of the threat model"
- Rowhammer → "only affects cheap sub-standard modules"

We have ignored microarchitectural attacks for many years:

- attacks on crypto → "software should be fixed"
- attacks on ASLR → "ASLR is broken anyway"
- attacks on SGX and TrustZone → "not part of the threat model"
- Rowhammer → "only affects cheap sub-standard modules"

→ for years we solely optimized for performance

Conclusions

- new class of software-based attacks

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

Conclusions

- new class of software-based attacks
- many problems to solve around microarchitectural attacks and especially transient-execution attacks
- dedicate more time into identifying problems and not solely in mitigating known problems

Meltdown, Spectre, ZombieLoad

Daniel Gruss, Moritz Lipp, Michael Schwarz

October 1, 2019

Graz University of Technology