ZombieLoad

Cross-Privilege-Boundary Data Sampling

Michael Schwarz, Moritz Lipp, **Daniel Moghimi**, Jo Van Bulck, Julian Stecklina, Thomas Prescher, Daniel Gruss

whoami

- Daniel Moghimi (@danielmgmi)
- Computer Security Enthusiast *since 2010*
  - Reverse Engineering
  - Application Security
- PhD Student *since 2017*
  - Microarchitectural Security
  - Side Channels
  - Breaking Cryptographic Implementations

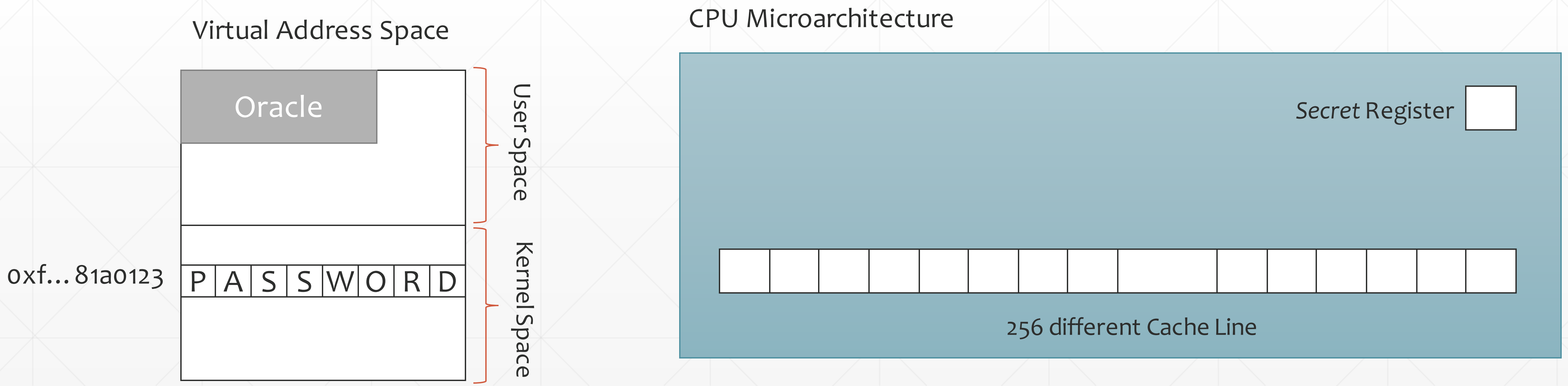
2018: Meltdown Attack?

```
char secret = *(char *) 0xffffffff81a0123;
printf("%c\n", secret);
```

- Segmentation Fault! Inaccessible Address.

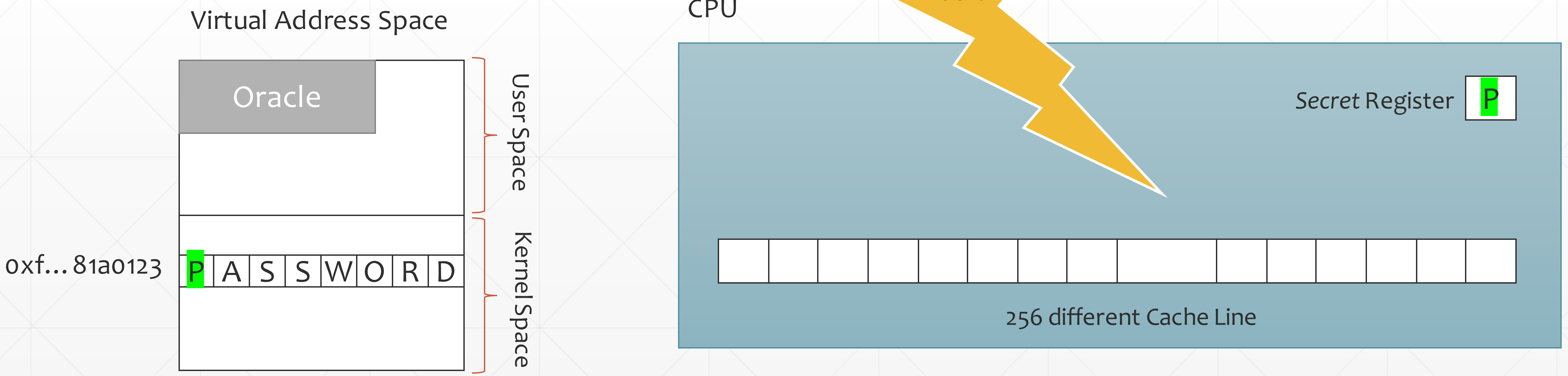
2018: Meltdown Attack?

```
char oracle[4096 * 256];
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];

for(char secret = 0; i < 256; i++){
    if(flush_reload(oracle[i * 4096]) < threshold)
        printf("%c\n", i);
}
```



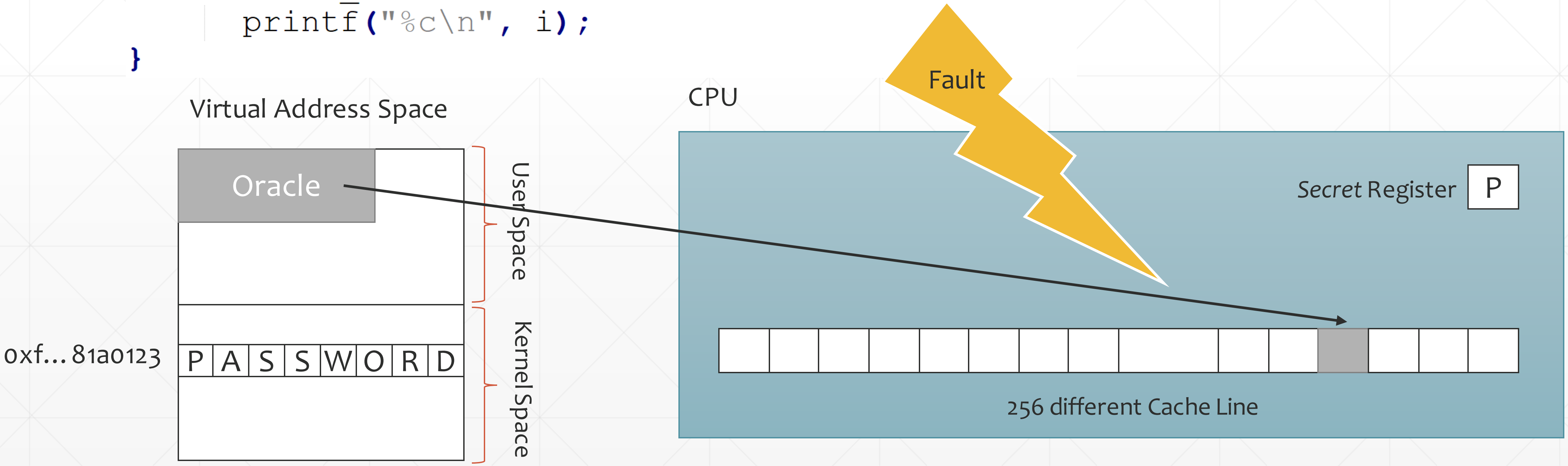
2018: Meltdown Attack?

```
char oracle[4096 * 256];
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];

for(char secret = 0; i < 256; i++){
    if(flush_reload(oracle[i * 4096]) < threshold)
        printf("%c\n", i);
}
```



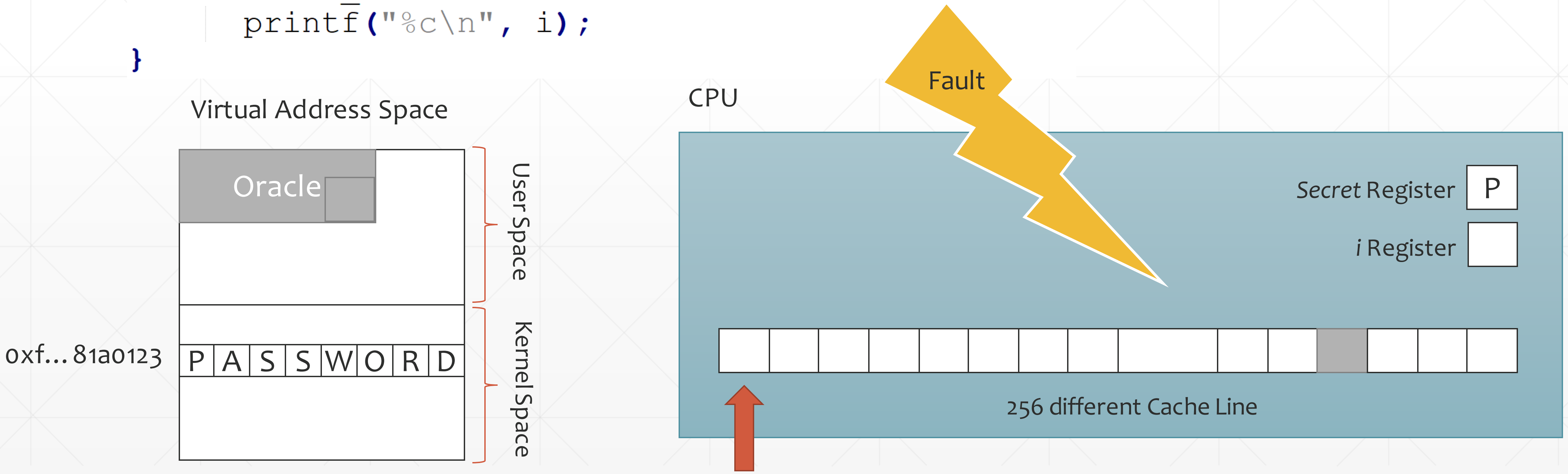
2018: Meltdown Attack?

```
char oracle[4096 * 256];
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];

for(char secret = 0; i < 256; i++){
    if(flush_reload(oracle[i * 4096]) < threshold)
        printf("%c\n", i);
}
```



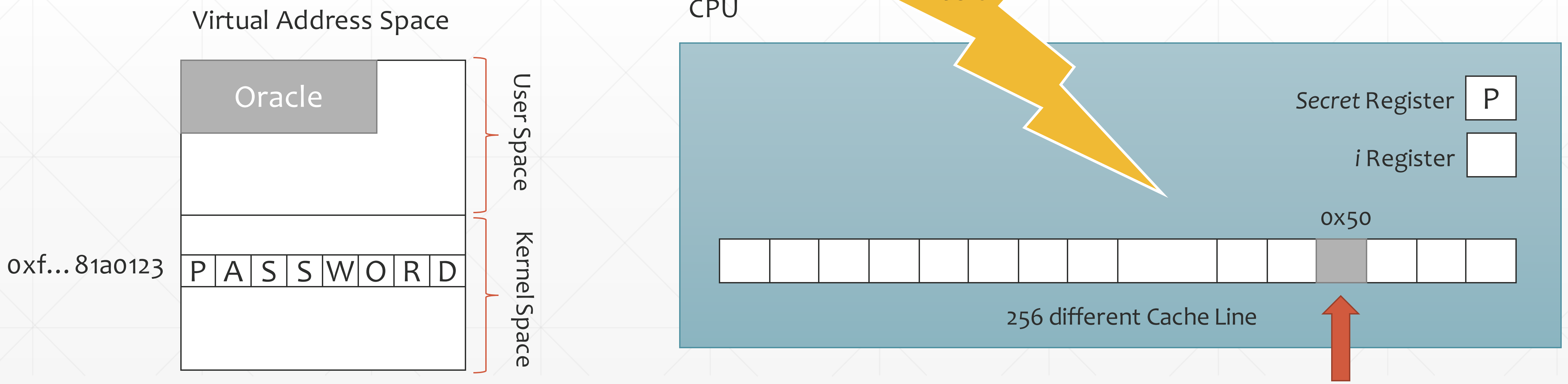
2018: Meltdown Attack?

```
char oracle[4096 * 256];
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];

for(char secret = 0; i < 256; i++){
    if(flush_reload(oracle[i * 4096]) < threshold)
        printf("%c\n", i);
}
```



2018: Meltdown Attack?

```
char oracle[4096 * 256];
char secret = *(char *) 0xffffffff81a0123;
char x = oracle[secret * 4096];

for(char secret = 0; i < 256; i++){
    if(flush_reload(oracle[i * 4096]) < threshold)
        printf("%c\n", i);
}
```



Meltdown-style Attacks !!!

- What if we dereference addresses that causes other faults/assists?
- What if the Microarchitecture is in a specific state during the fault?
- Where does this data leak from?!

ZombieLoad !?!

ZombieLoad !?!

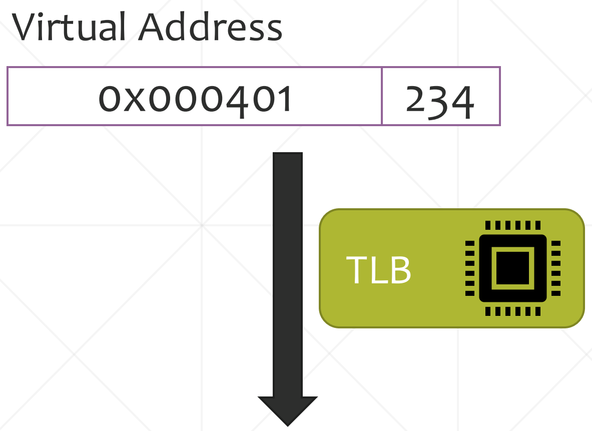
```
mov 0x401234, %rsi

mov (%rsi), %rax
```

Virtual Address

0x000401	234

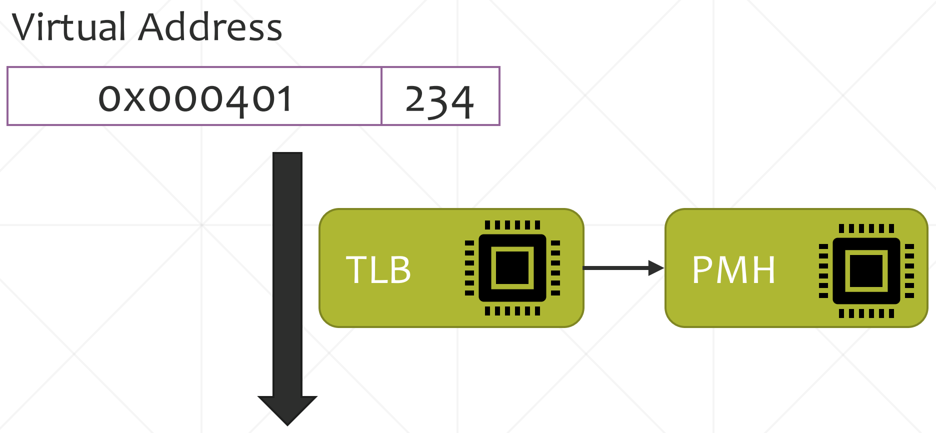
ZombieLoad !?!

```
mov 0x401234, %rsi

mov (%rsi), %rax
```



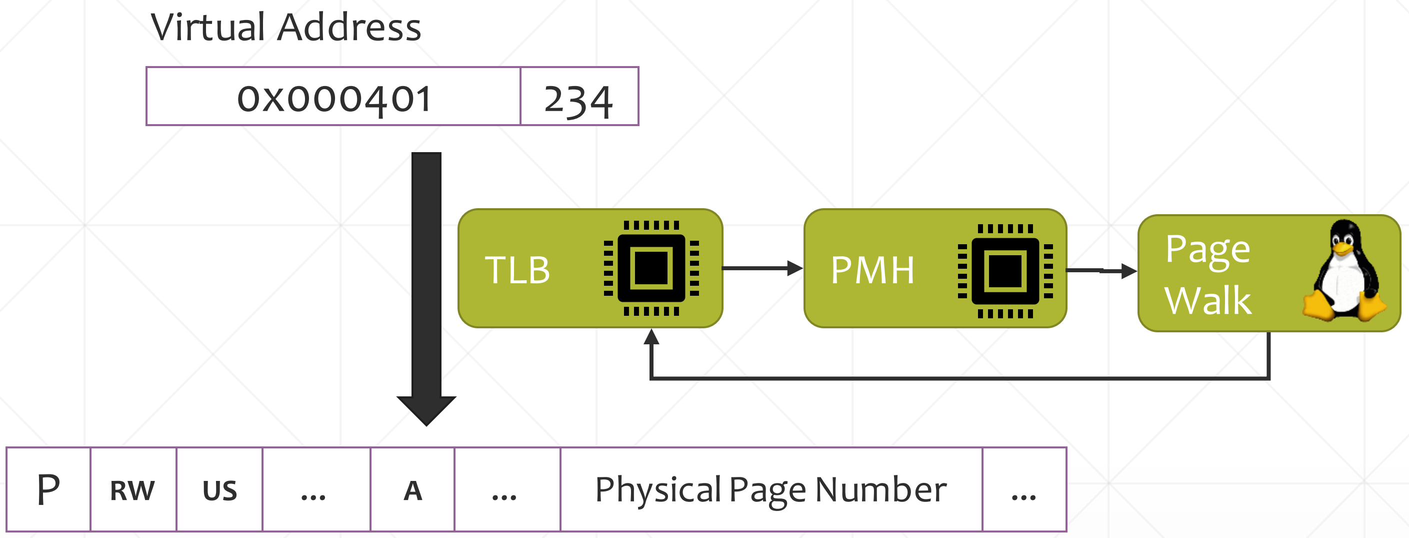
ZombieLoad !?!

```
mov 0x401234, %rsi

mov (%rsi), %rax
```



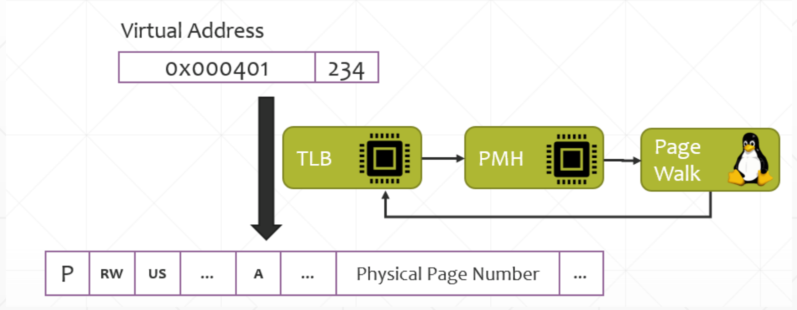
ZombieLoad !?!

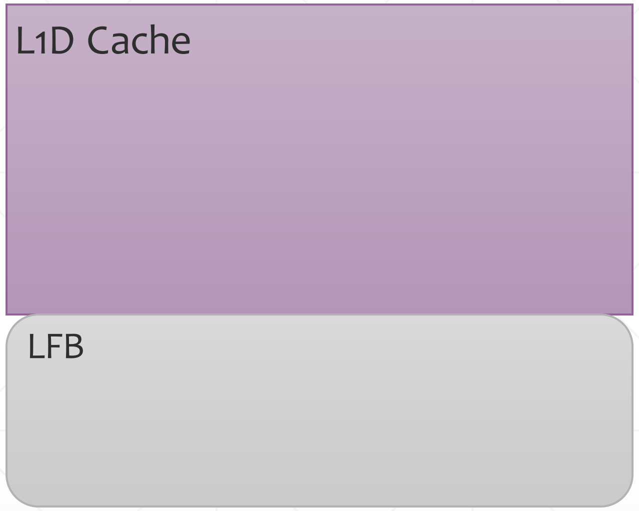
```
mov 0x401234, %rsi

mov (%rsi), %rax
```



ZombieLoad !?!

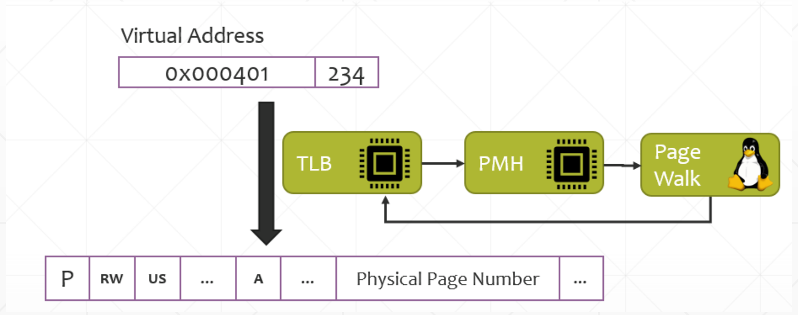
```
mov 0x401234, %rsi

mov (%rsi), %rax
```

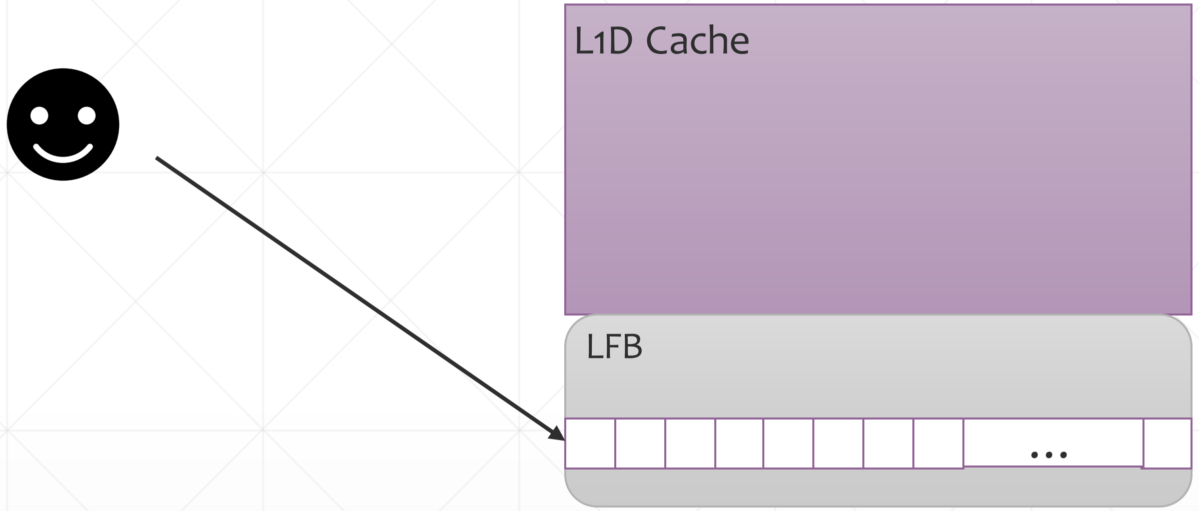




ZombieLoad !?!

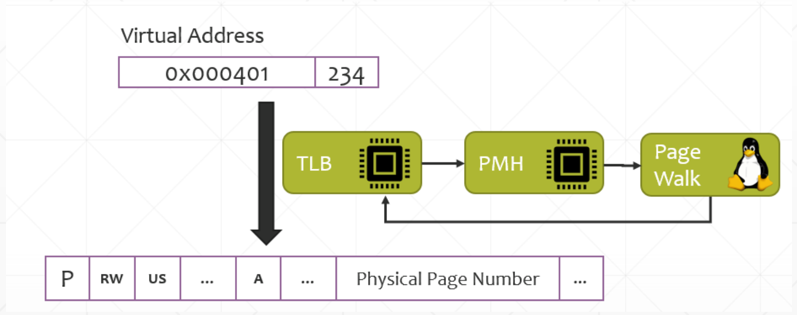
```
mov 0x401234, %rsi

mov (%rsi), %rax
```

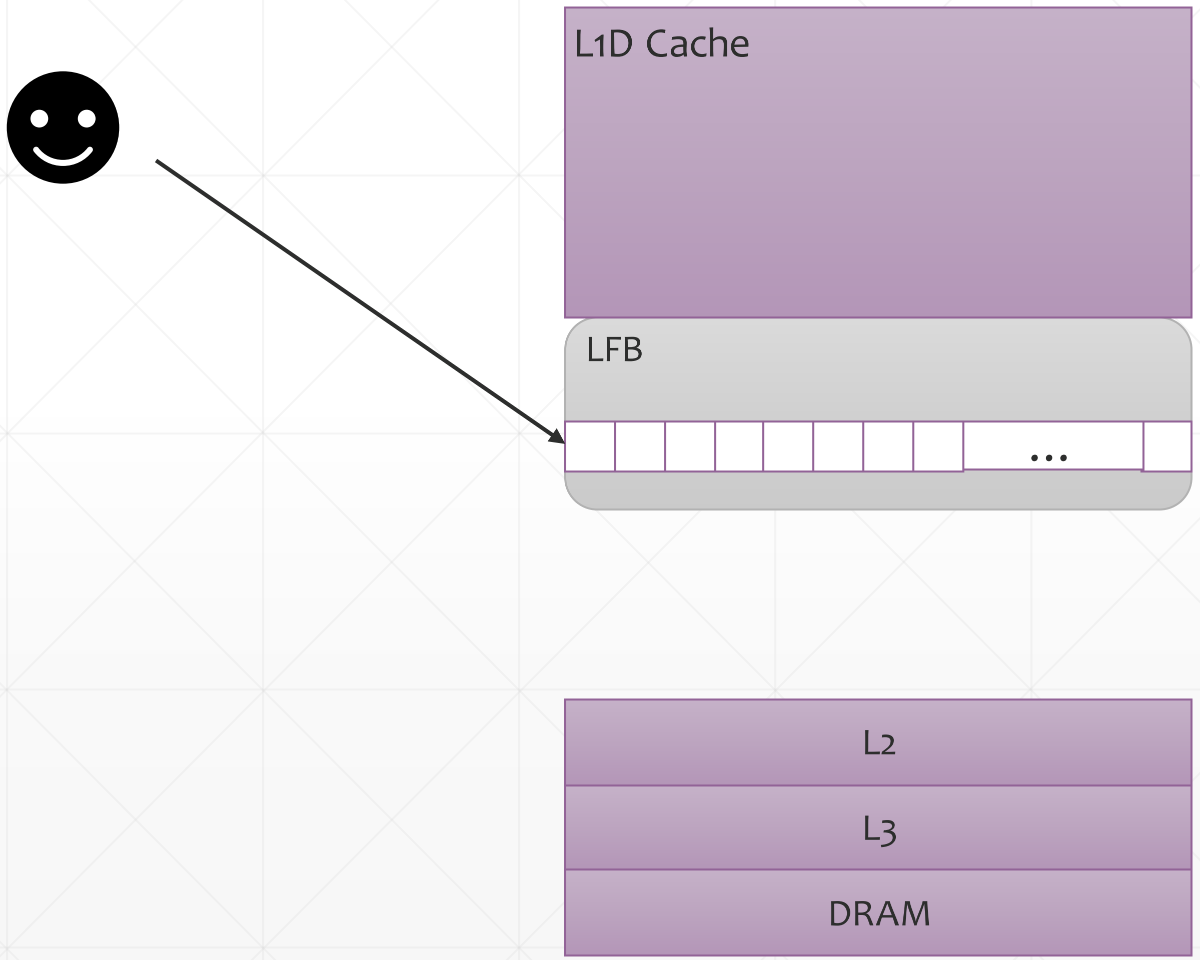




ZombieLoad !?!

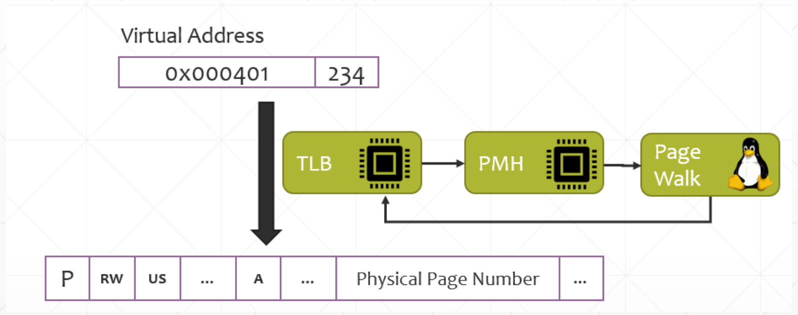
```
mov 0x401234, %rsi

mov (%rsi), %rax
```

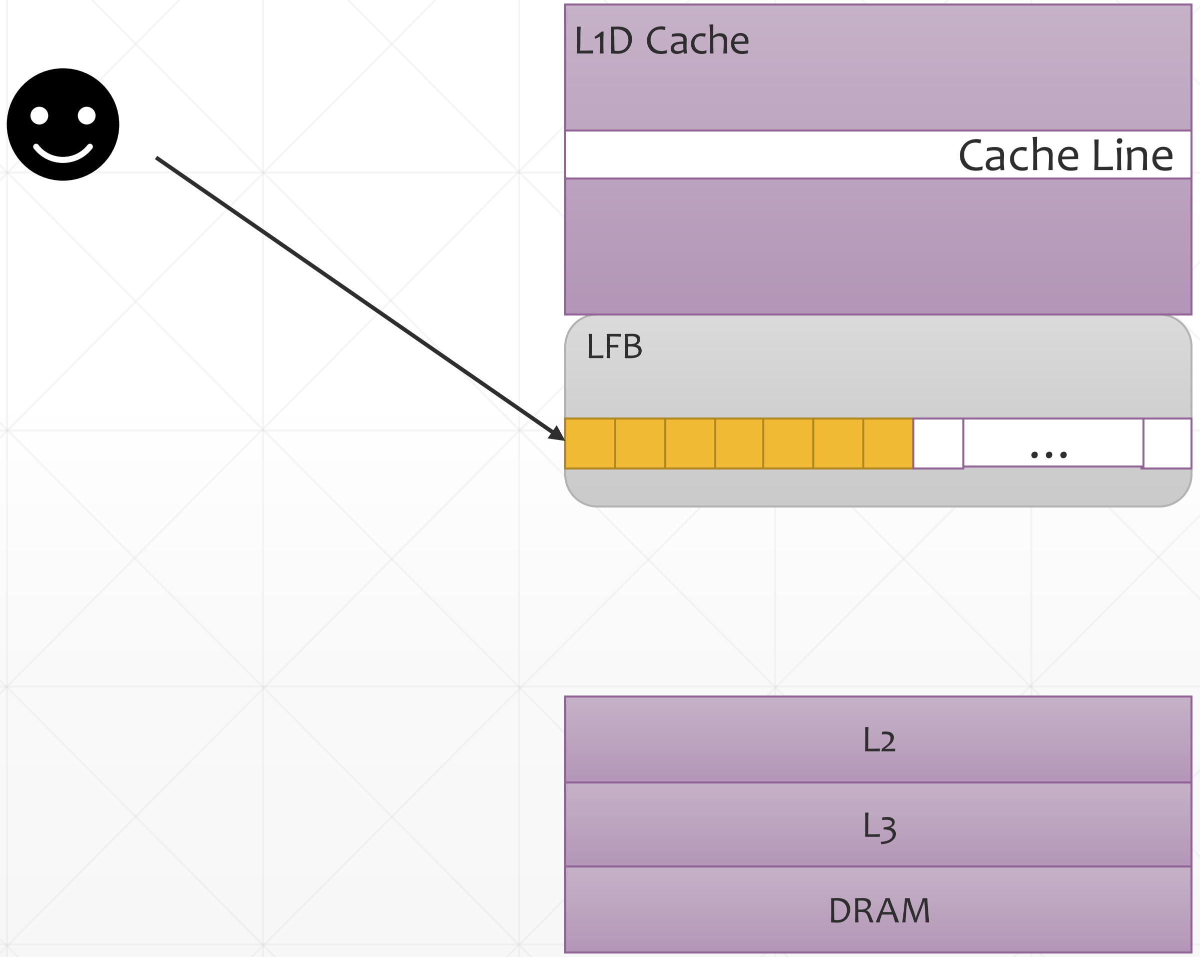




ZombieLoad !?!

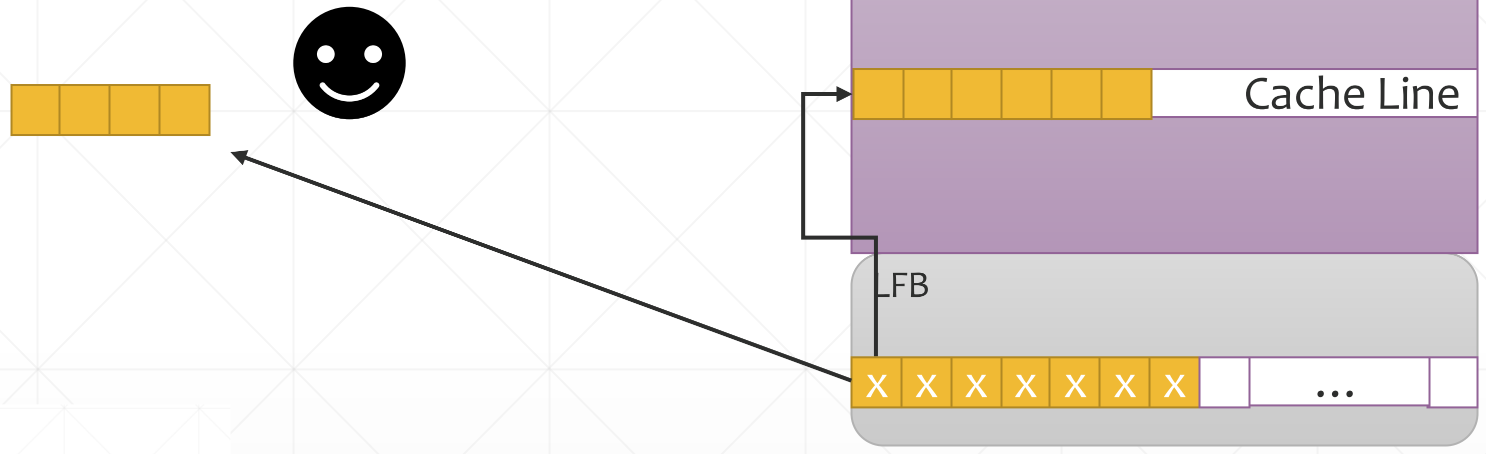
```
mov 0x401234, %rsi

mov (%rsi), %rax
```

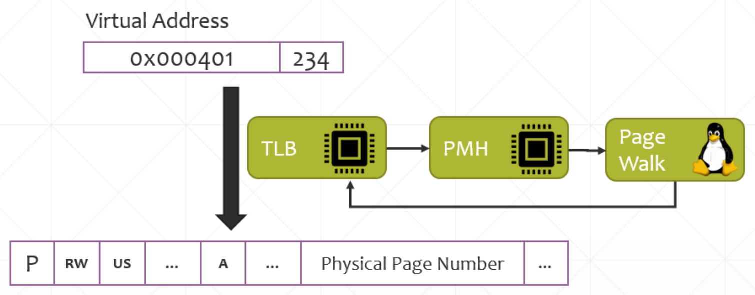


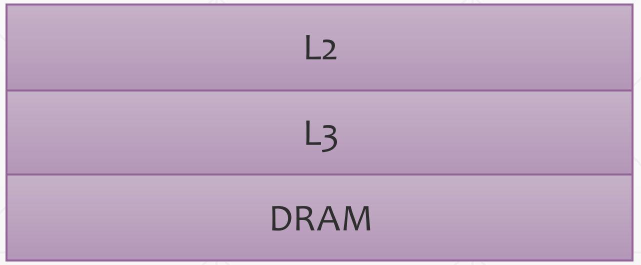


ZombieLoad !?!

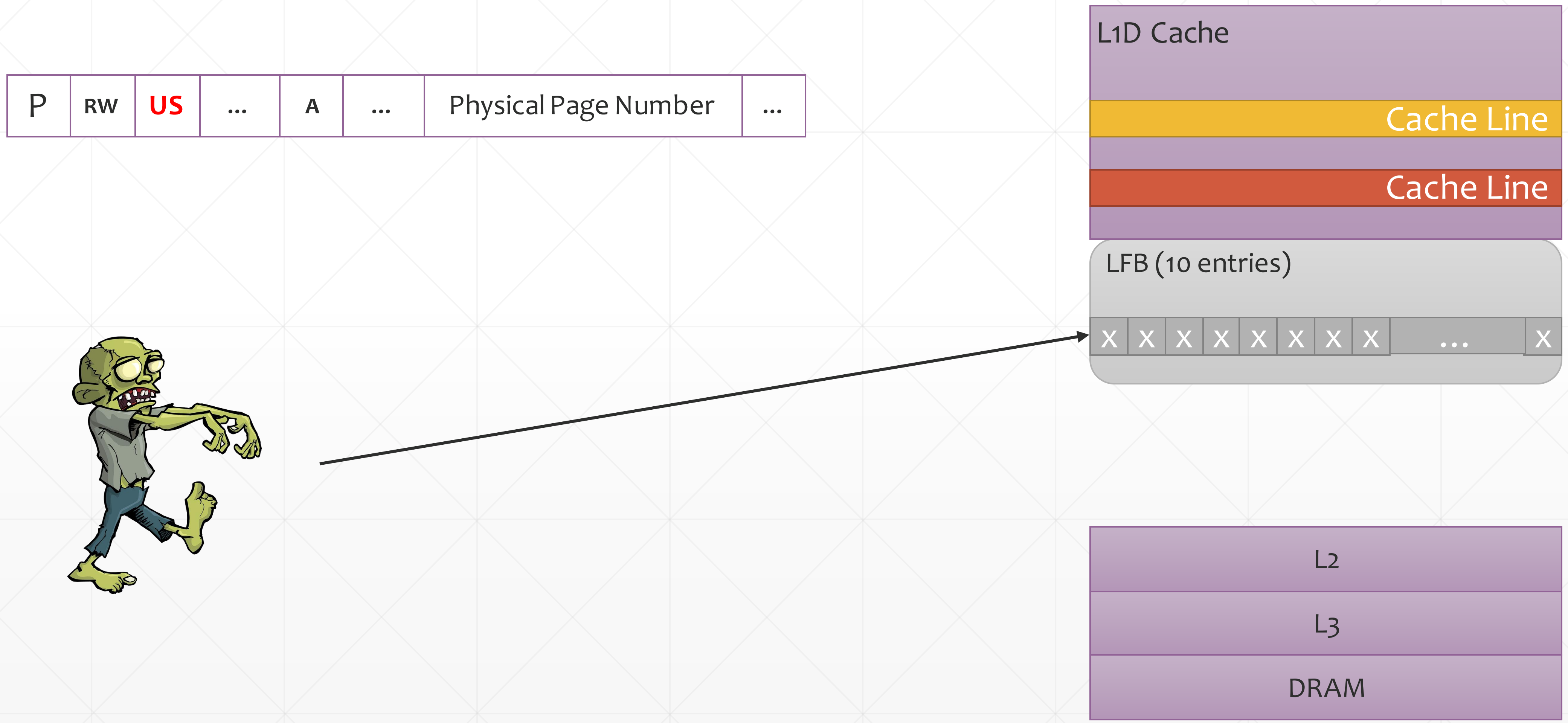
```
mov 0x401234, %rsi

mov (%rsi), %rax
```





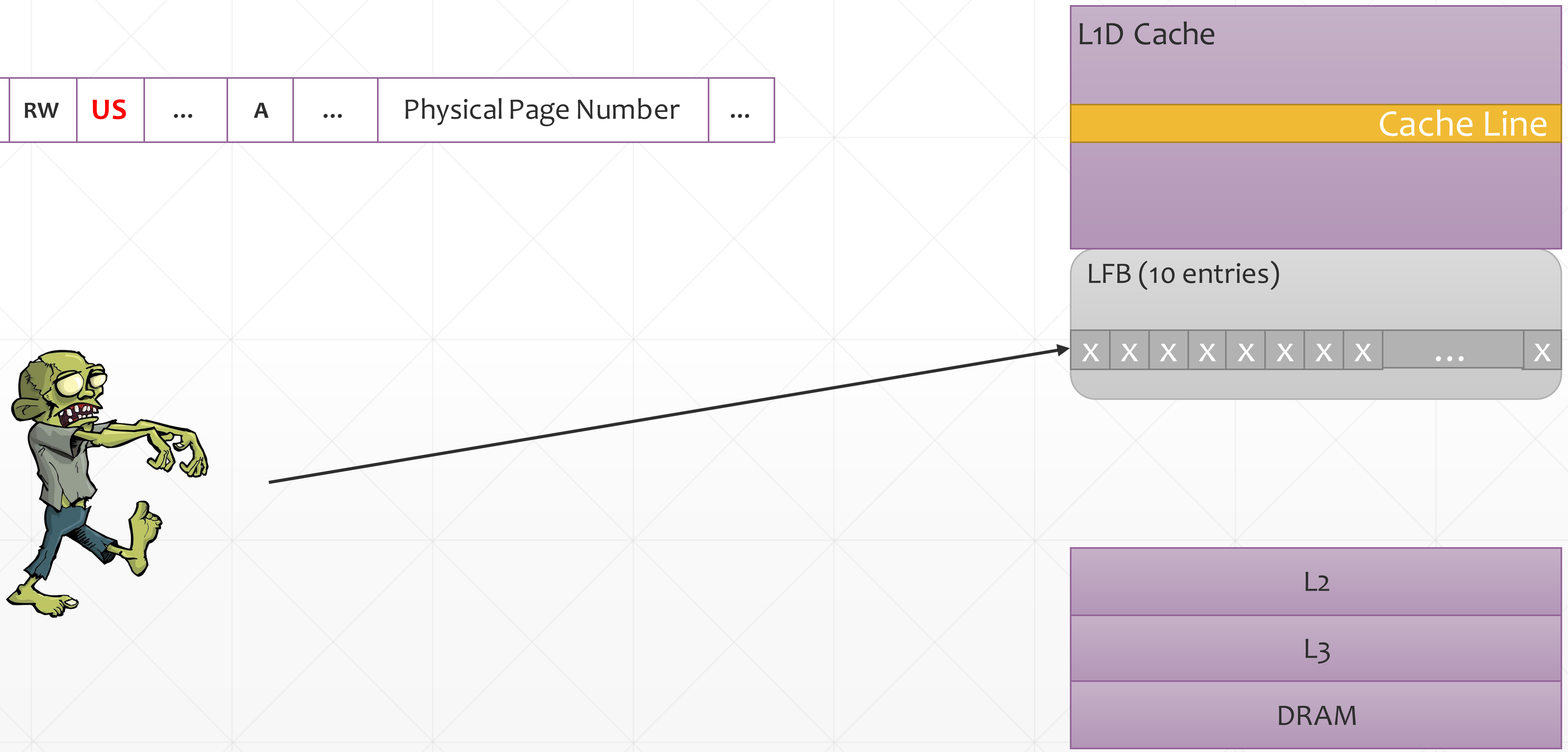


ZombieLoad Attack !?!



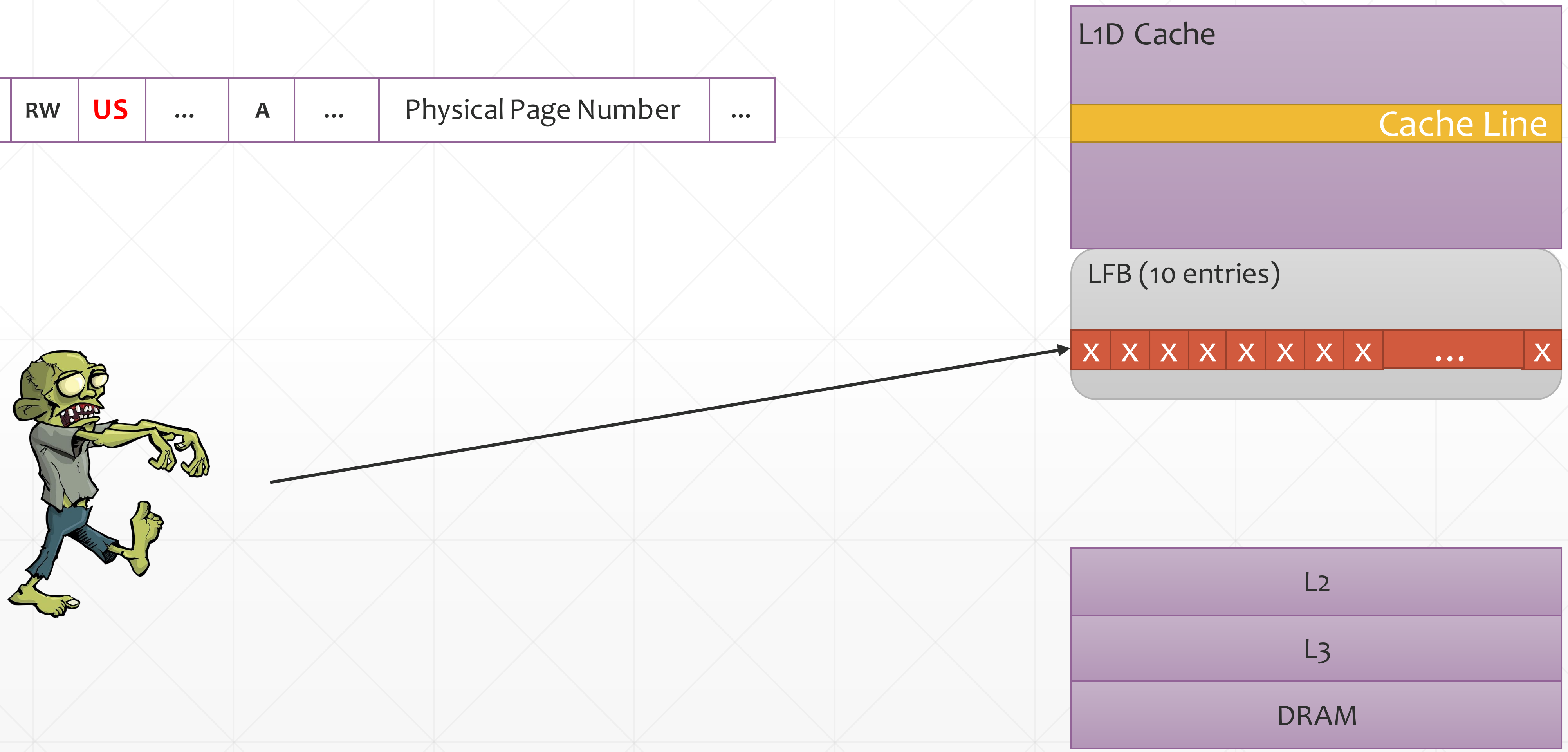
ZombieLoad Attack !?!

P	RW	US	...	A	...	Physical Page Number	...

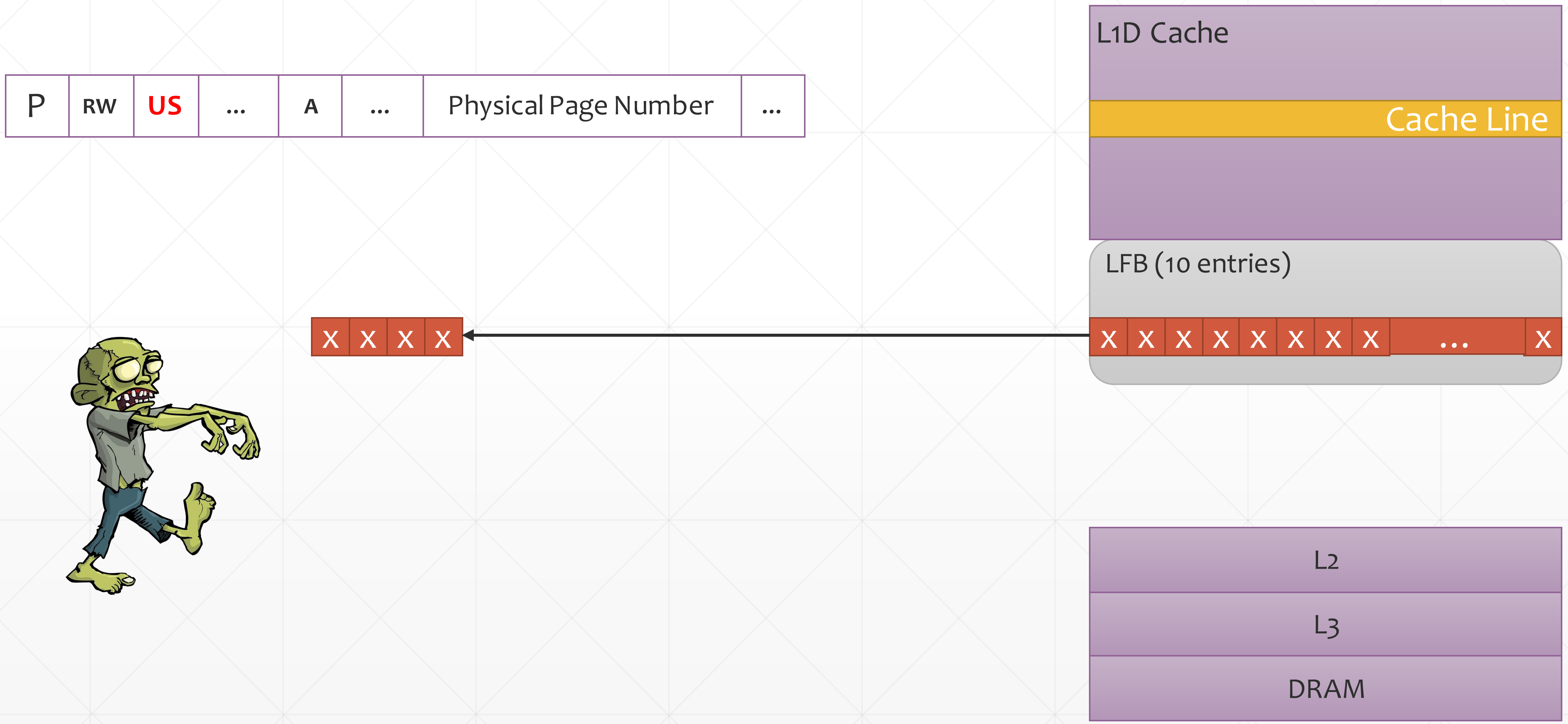


ZombieLoad Attack !?!

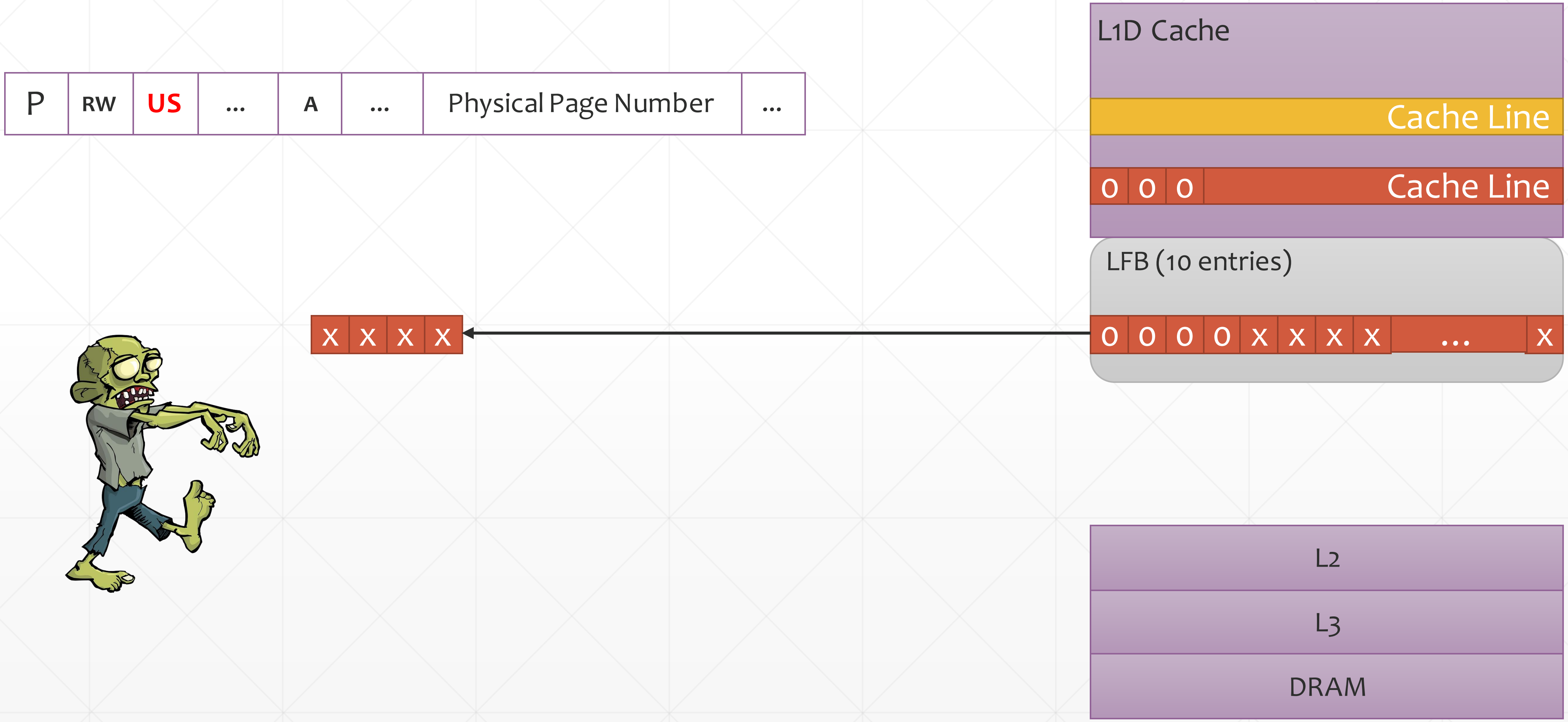
P	RW	US	...	A	...	Physical Page Number	...



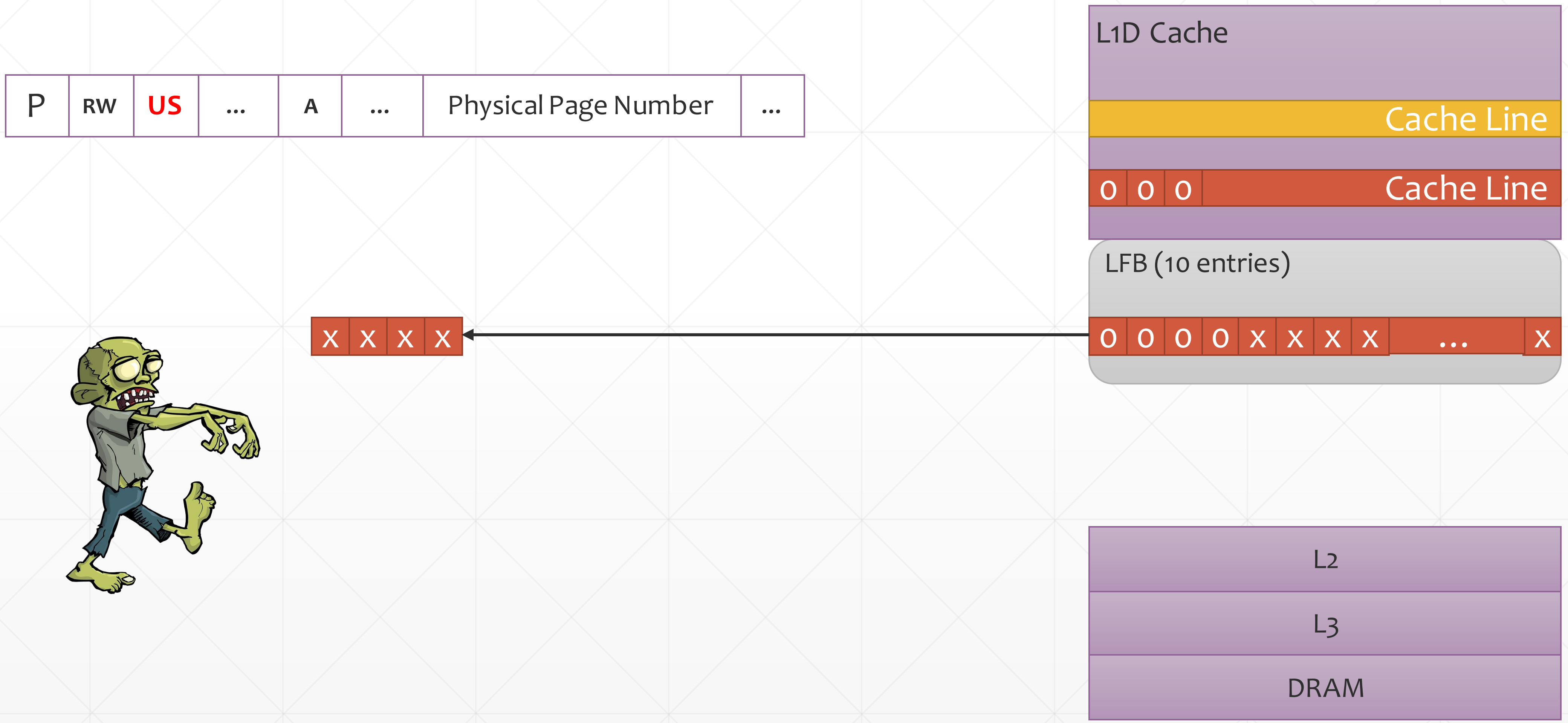
ZombieLoad Attack !?!



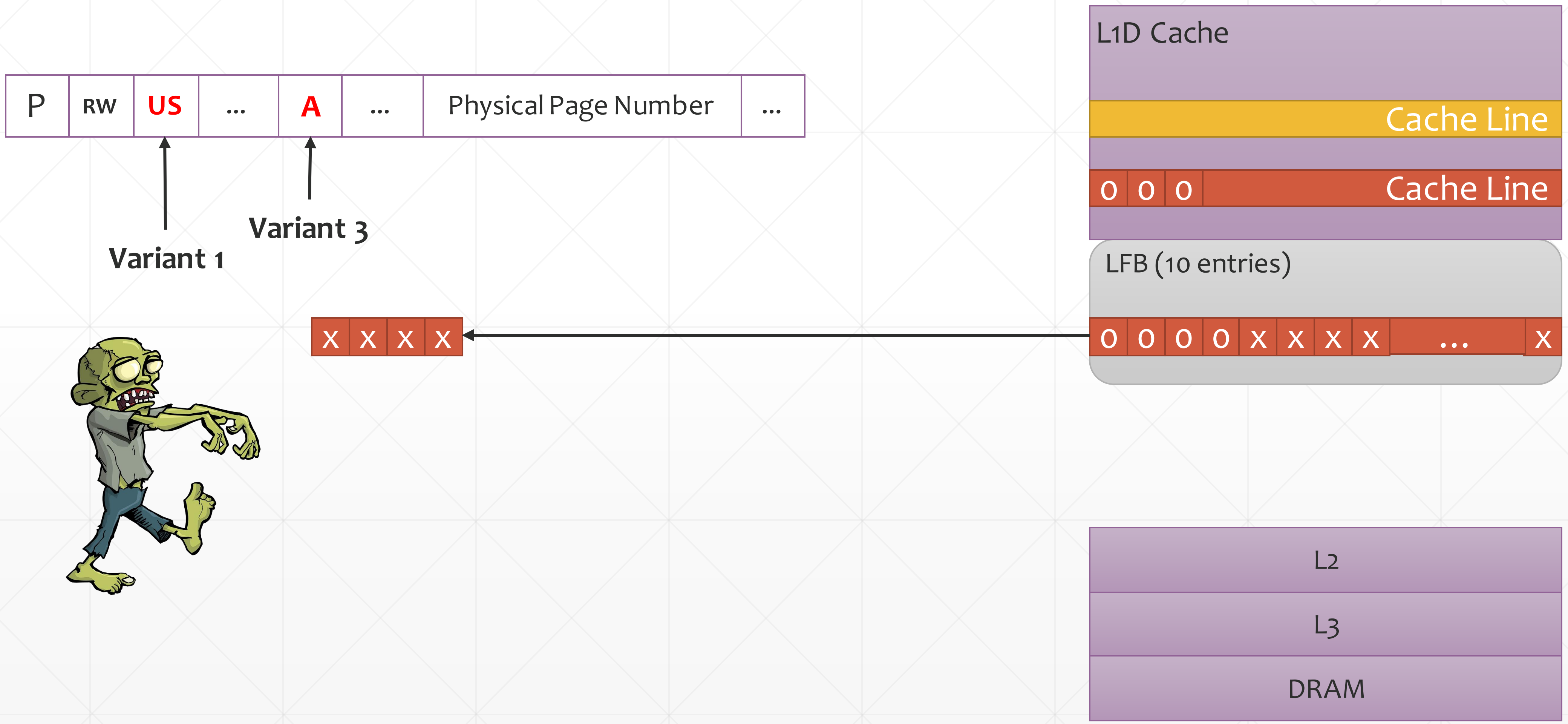
ZombieLoad Attack !?!



ZombieLoad Attack !?!



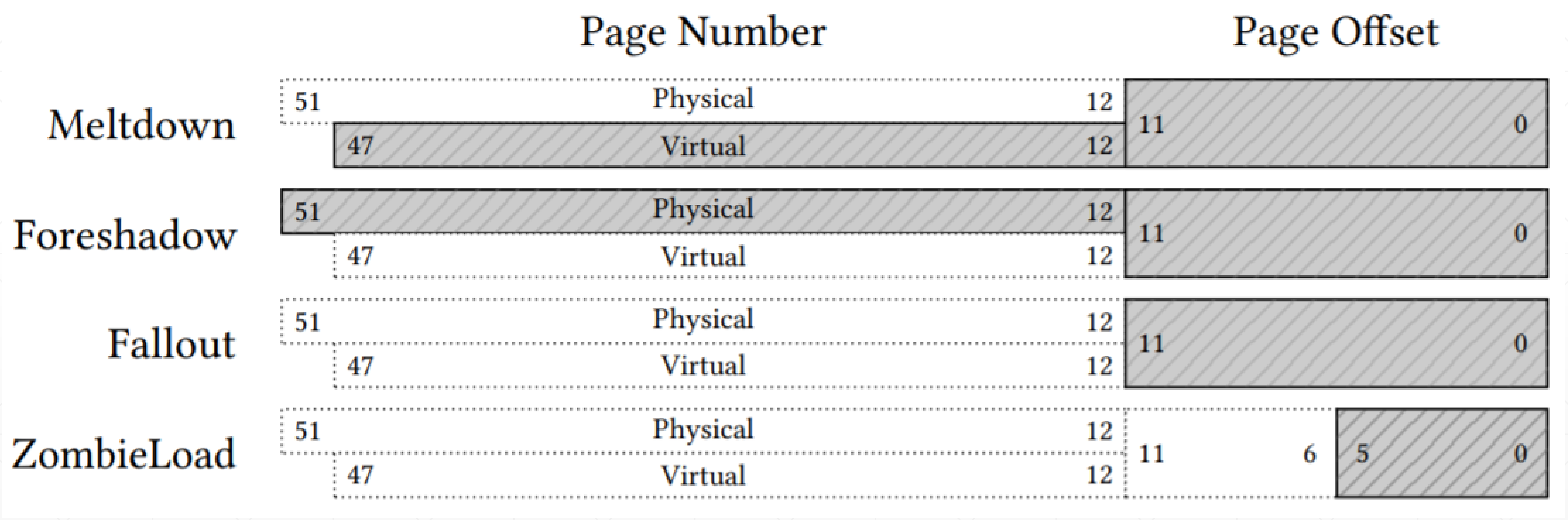
ZombieLoad Attack !?!



Microcode Assist on ‘A’ Bit

- The CPU tells the OS if a page has been accessed or not by setting the Access Bit
- The OS can clear the bit and causes a microcode assist
- The microcode assist flushes the pipeline while setting the access bit

ZombieLoad VS. other Meltdown-Style Attacks



What can we do with this data leakage?

- Attack across Process Context Switches
- Attack across Simultaneous Multithreading (SMT) AKA. Intel Hyperthreading

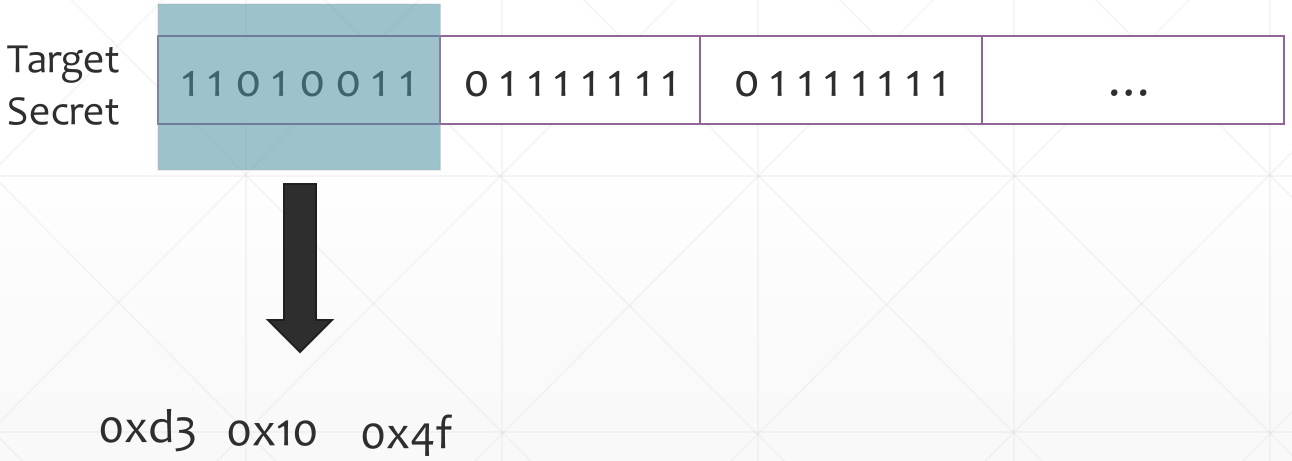
Data Sampling - Domino Attack

- We may leak bytes of data from other unimportant fill buffer entries
- Leak domino bytes to perform error correction

Target Secret	1 1 0 1 0 0 1 1	0 1 1 1 1 1 1 1	0 1 1 1 1 1 1 1	…

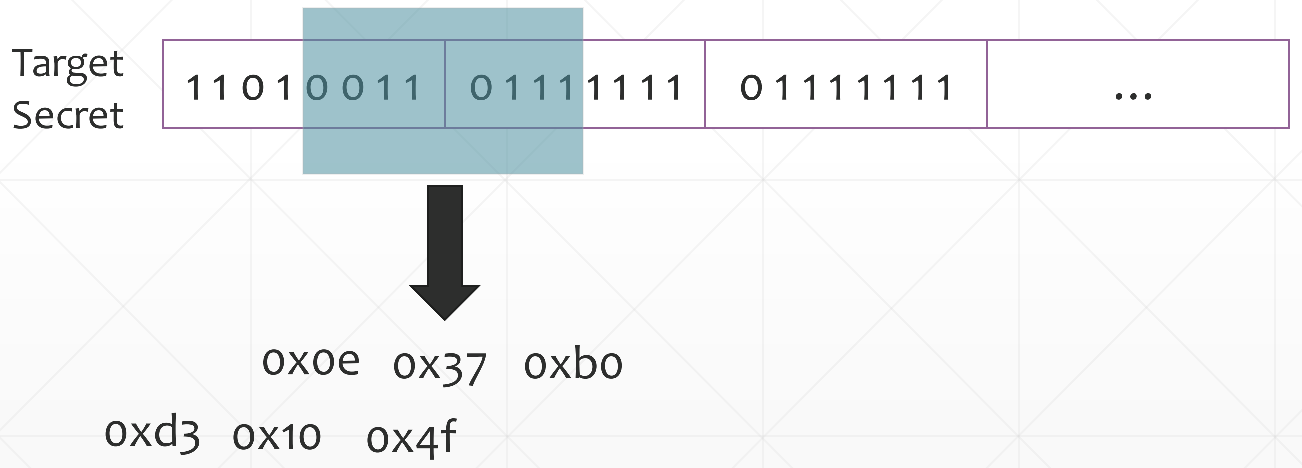
Data Sampling - Domino Attack

- We may leak bytes of data from other unimportant fill buffer entries
- Leak domino bytes to perform error correction



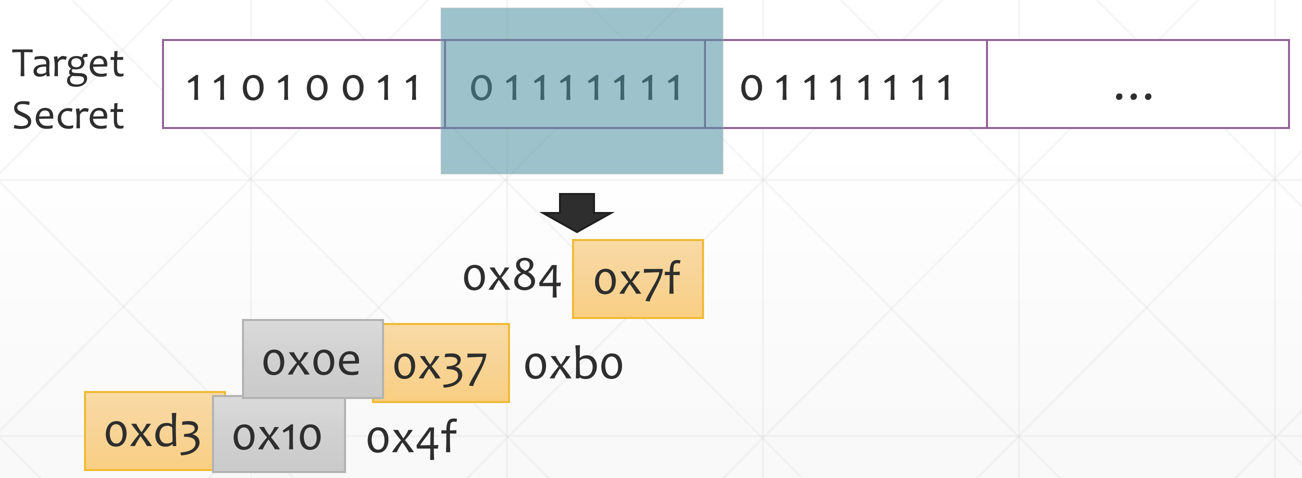
Data Sampling - Domino Attack

- We may leak bytes of data from other unimportant fill buffer entries
- Leak domino bytes to perform error correction



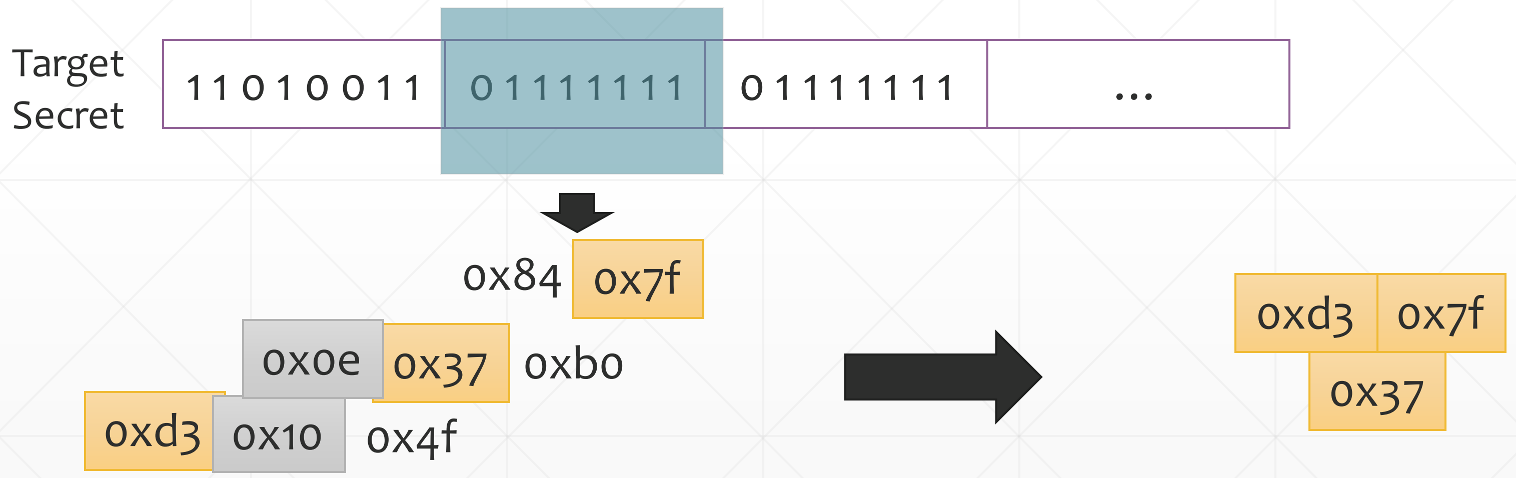
Data Sampling - Domino Attack

- We may leak bytes of data from other unimportant fill buffer entries
- Leak domino bytes to perform error correction



Data Sampling - Domino Attack

- We may leak bytes of data from other unimportant fill buffer entries
- Leak domino bytes to perform error correction

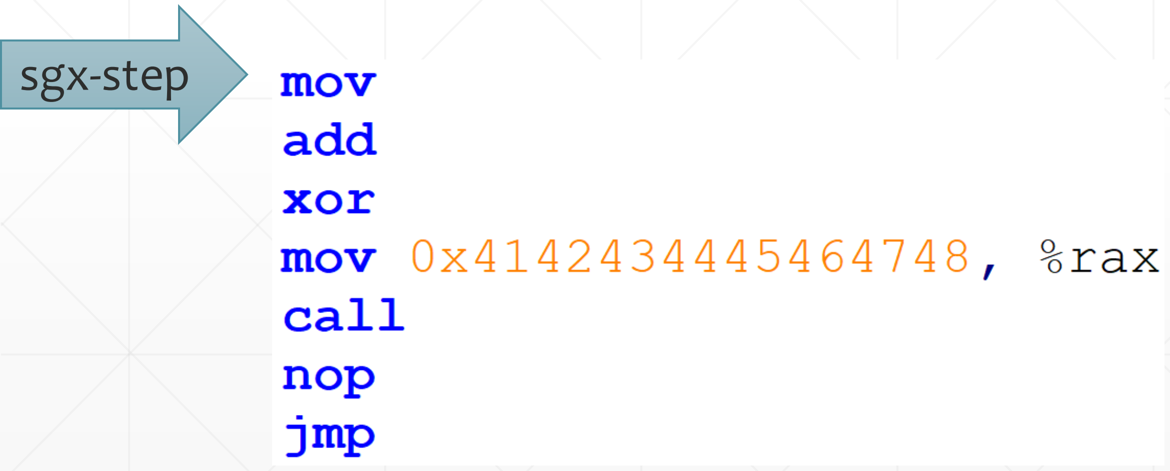


Recovering Intel SGX Sealing Key

- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS

Recovering Intel SGX Sealing Key

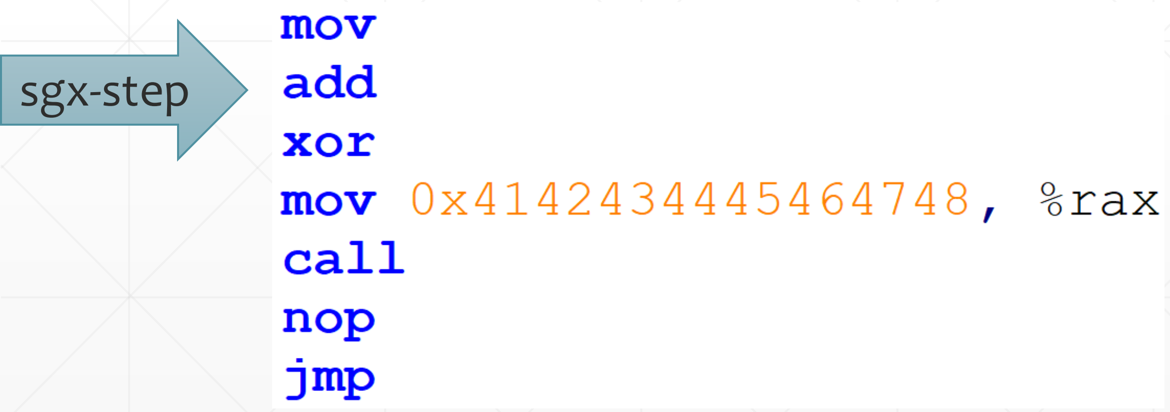
- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS



```
mov
add
xor
mov 0x4142434445464748, %rax
call
nop
jmp
```

Recovering Intel SGX Sealing Key

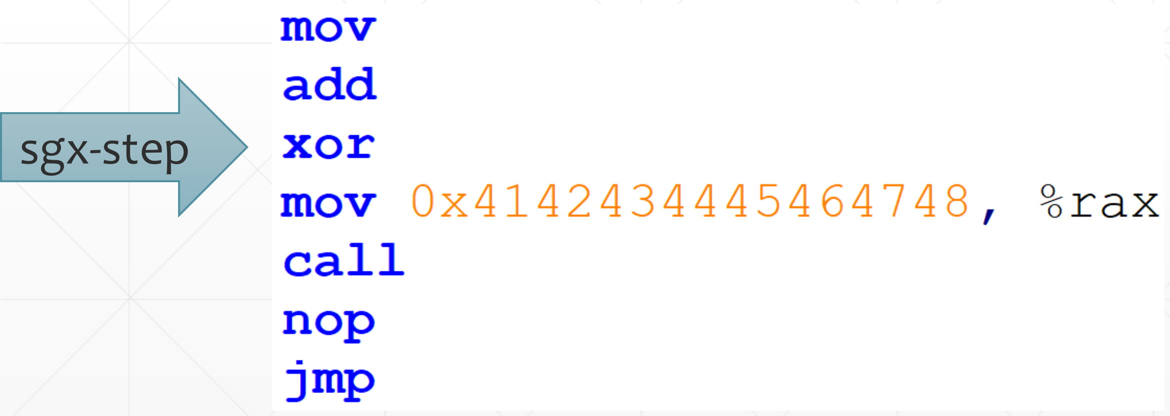
- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS



```
mov
add
xor
mov 0x4142434445464748, %rax
call
nop
jmp
```

Recovering Intel SGX Sealing Key

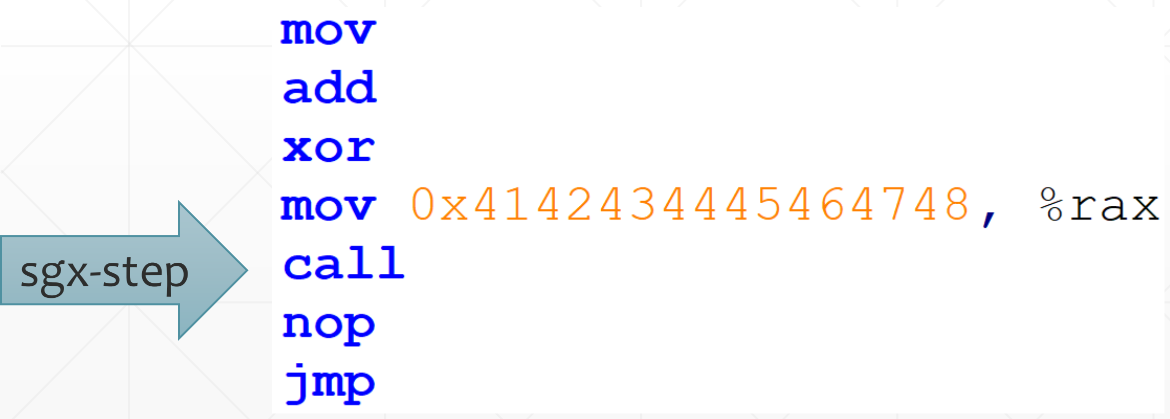
- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS



```
mov
add
xor
mov 0x4142434445464748, %rax
call
nop
jmp
```

Recovering Intel SGX Sealing Key

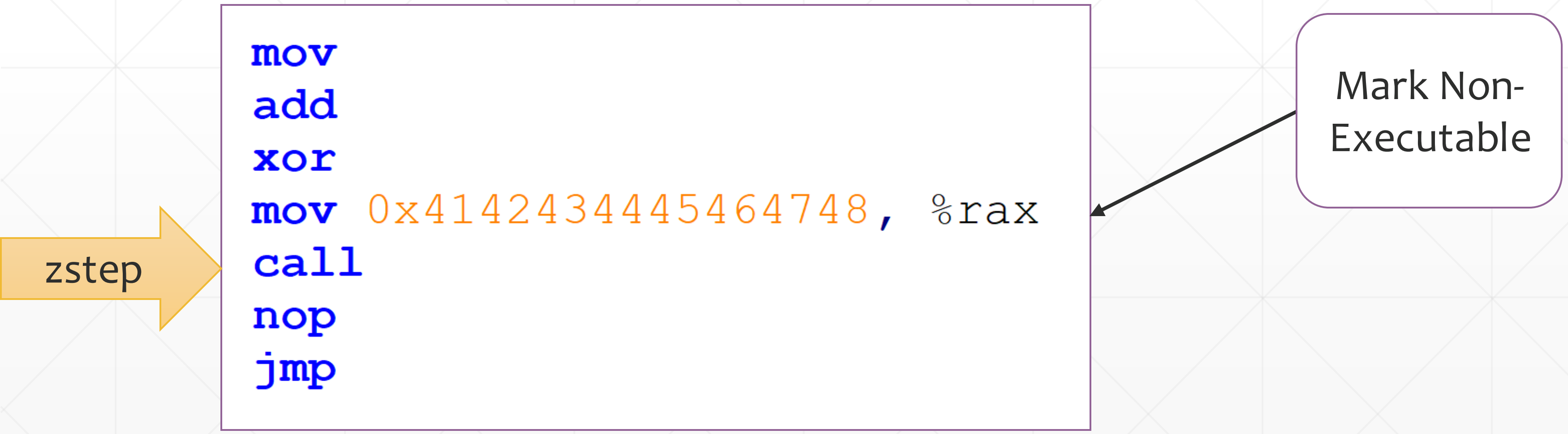
- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS



```
mov
add
xor
mov 0x4142434445464748, %rax
call
nop
jmp
```

Recovering Intel SGX Sealing Key

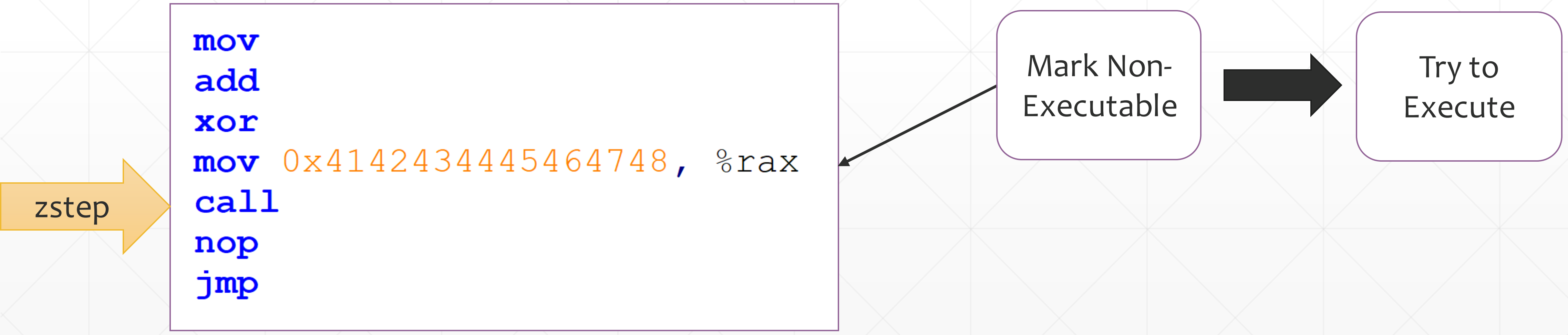
- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS



```
mov
add
xor
mov 0x4142434445464748, %rax
call
nop
jmp
```

Recovering Intel SGX Sealing Key

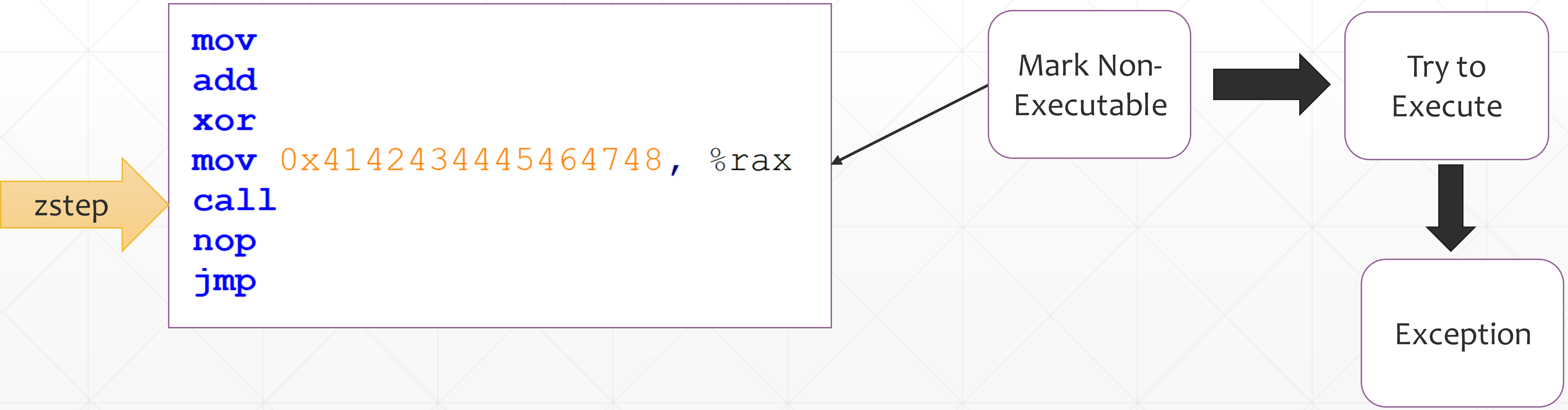
- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS



```
mov
add
xor
mov 0x4142434445464748, %rax
call
nop
jmp
```

Recovering Intel SGX Sealing Key

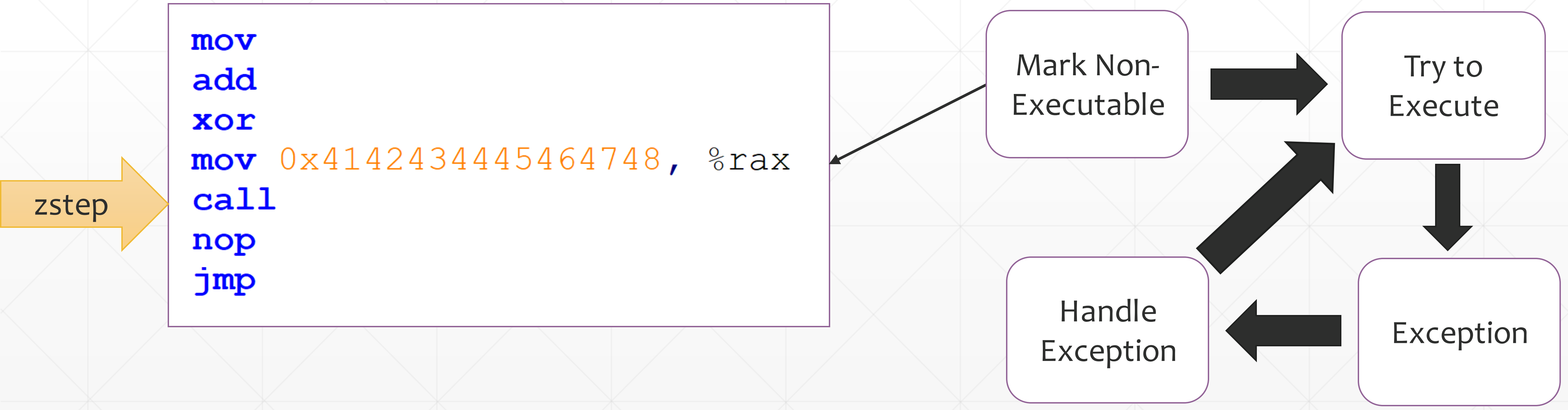
- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS



```
mov
add
xor
mov 0x4142434445464748, %rax
call
nop
jmp
```

Recovering Intel SGX Sealing Key

- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS



```
mov
add
xor
mov 0x4142434445464748, %rax
call
nop
jmp
```

Recovering Intel SGX Sealing Key

- Intel SGX allow developers to have hardware support for TEE
- Malicious OS is part of the threat model
- We can read register values of a trusted enclave with help of a malicious OS
- Repeated Context Switch in the transient domain w/ the same register values

ZombieLoad Demo! Recovering Linux Shadow

Is there any Mitigation?

- Intel suggested an instruction sequence to fill all the buffers across context switch
- Disable hyperthreading

Questions?!

MORITZ LIPP
MICHAEL SCHWARZ
DANIEL MOGHIMI
JO VAN BULCK
ZOMBIELOAD
@mlqxyz
@misc0110
@danielmgmi
@jovanbulck
GRAZ UNIVERSITY OF TECHNOLOGY PRESENTS IN COLLABORATION WITH
WORCESTER POLYTECHNIC INSTITUTE, KU LEUVEN, AND CYBERUS TECHNOLOGY
AN ACM CCS 2019 PAPER "ZOMBIELOAD: CROSS-PRIVILEGE-BOUNDARY DATA SAMPLING"
WRITTEN BY MICHAEL SCHWARZ, MORITZ LIPP, DANIEL MOGHIMI, JO VAN BULCK, JULIAN STECKLINA, THOMAS PRESCHER, DANIEL GRUSS

https://zombieloadattack.com/