CS773-2022-Autumn: Computer Architecture for Performance and Security

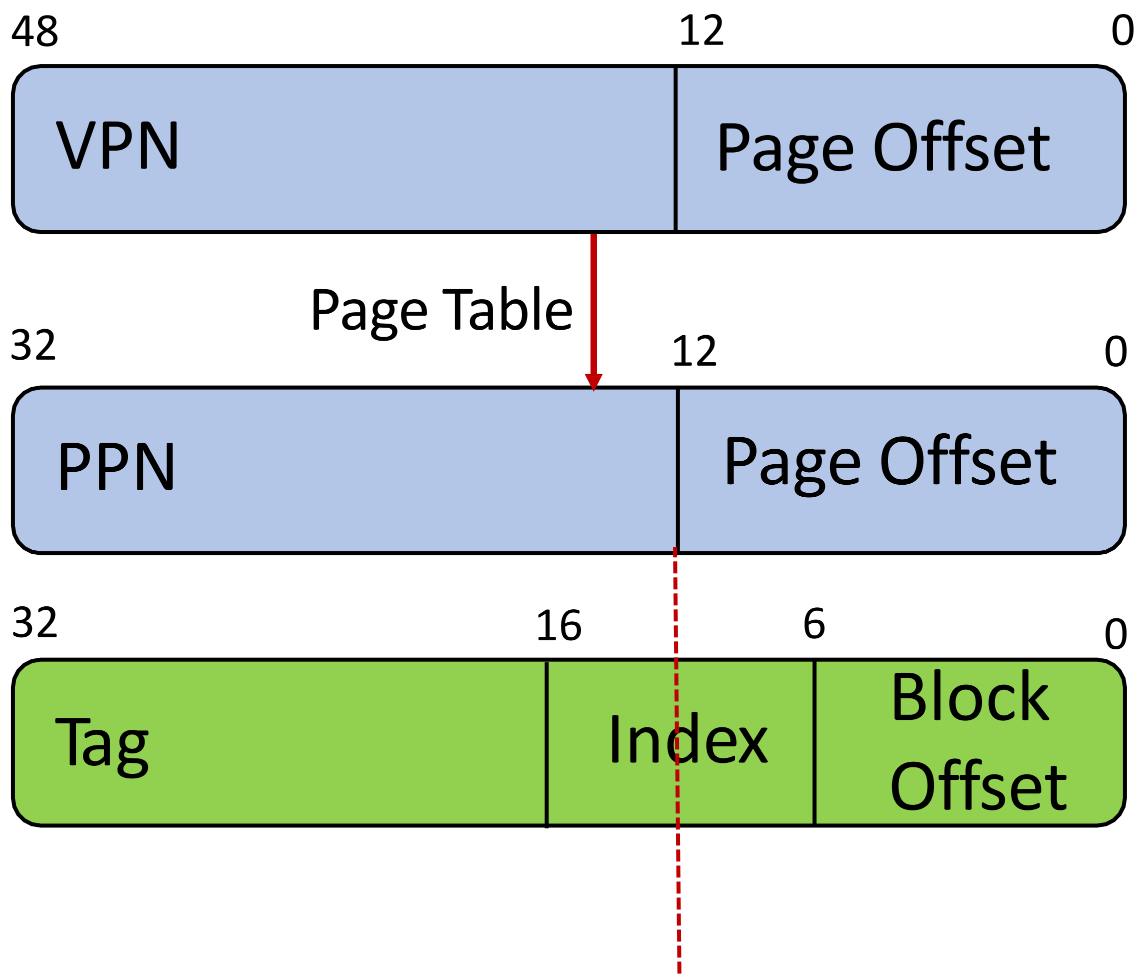
Lecture 11-12: Transient attacks

https://www.cse.iitb.ac.in/~biswa/

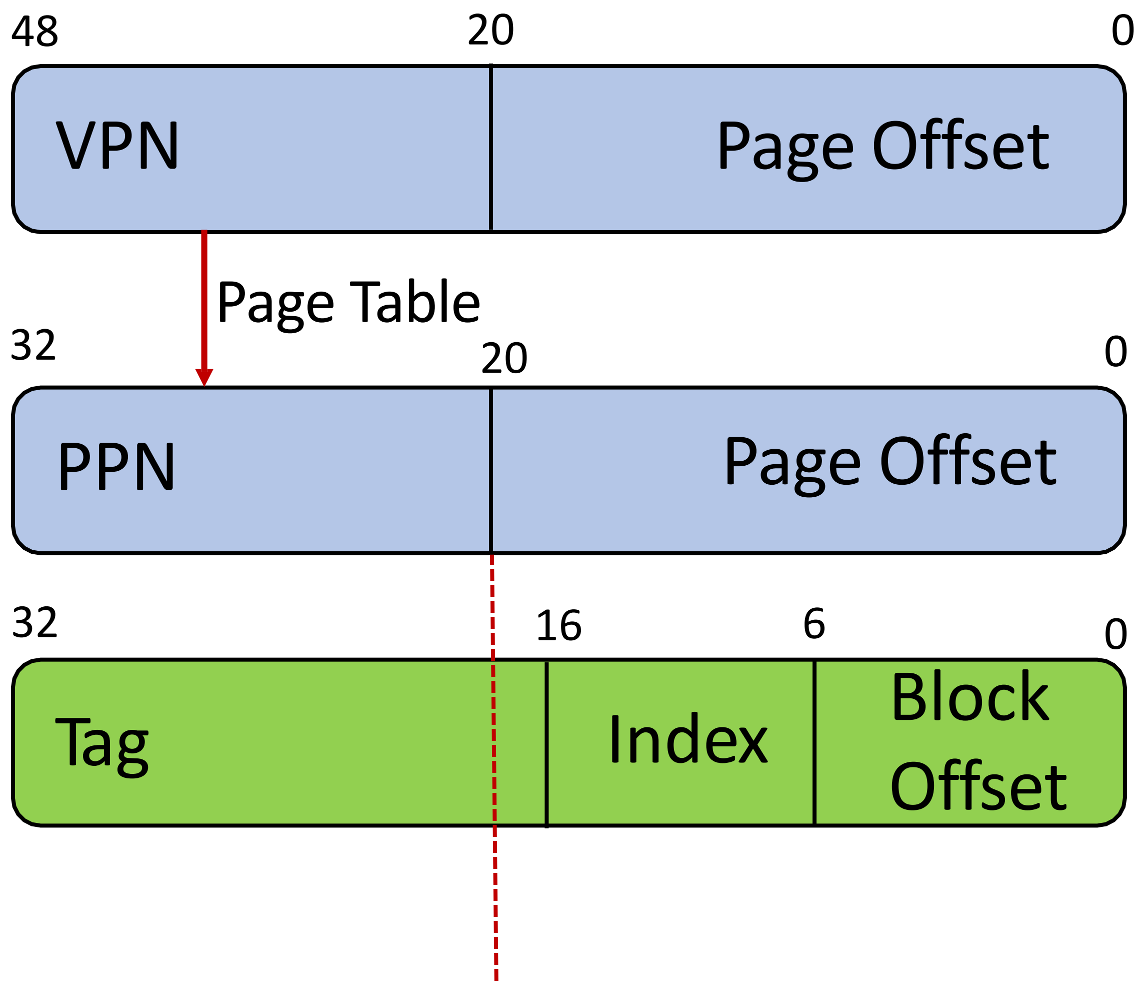
Thrashing Entire LLC: Questions of interest

- Extremely Slow pre-attack step: Think about an 8MB/16MB LLC
- Why not thrash a group of addresses that are mapped to the same set?
- Is there an algorithm to find out the same? Eviction set algorithm?
- But what about virtual to physical address translation? LLC will have the physical address.
- How to trigger requests that will go to the same set bypassing L1 and L2?

Attacker cannot control: LLC with 1024 sets



What if we have huge pages

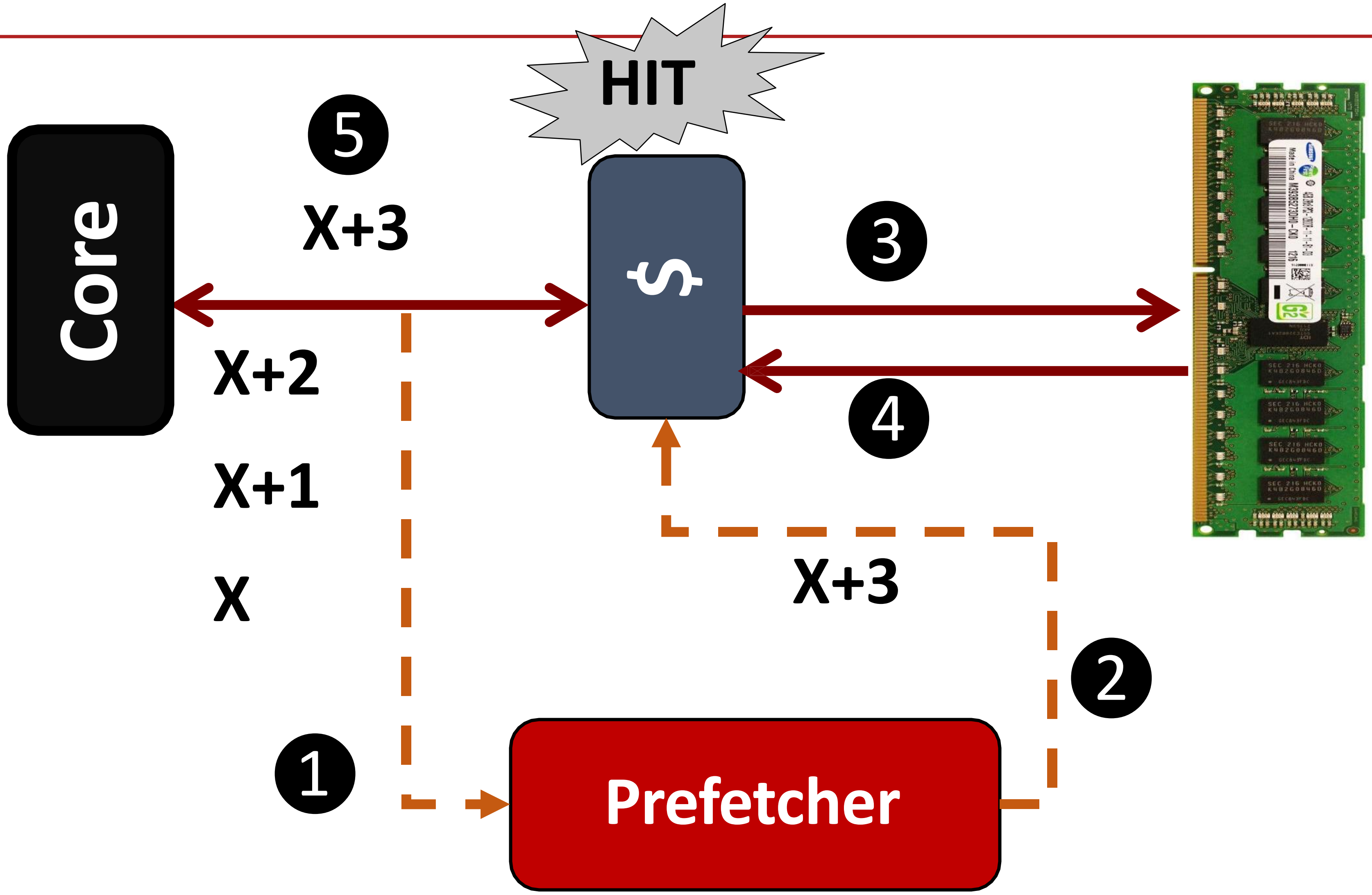


Awesome. Now attacker can control all the accesses to a particular set.

What About?

- Effect of cache replacement policy at the LLC?
- What if it is adaptive?
- What if attacker’s access pattern is predictable?
- A hardware prefetcher can affect the eviction set creation process?

Hardware Prefetching



What About?

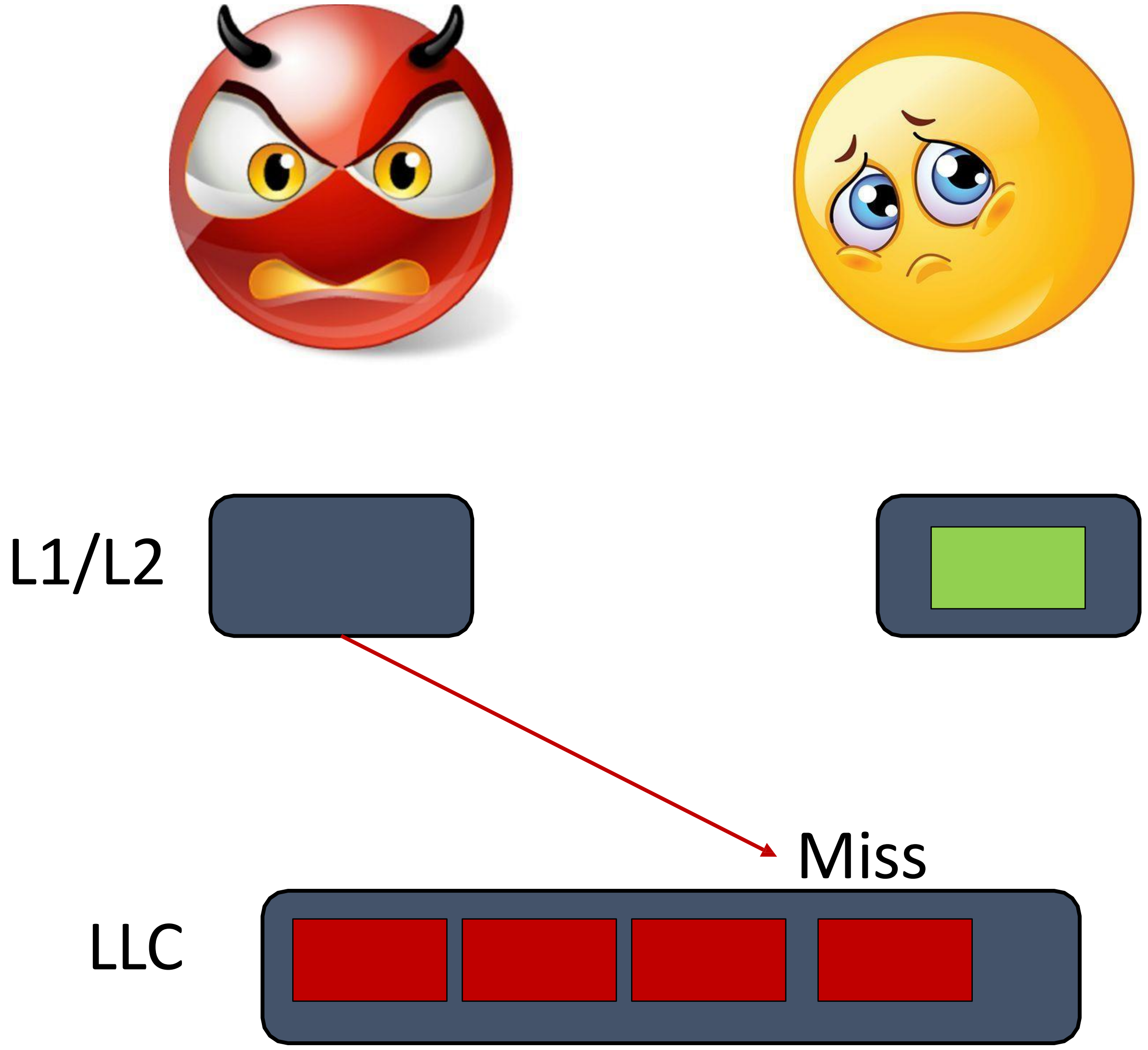
Effect of cache replacement policy at the LLC?

Fool the replacement policy too.

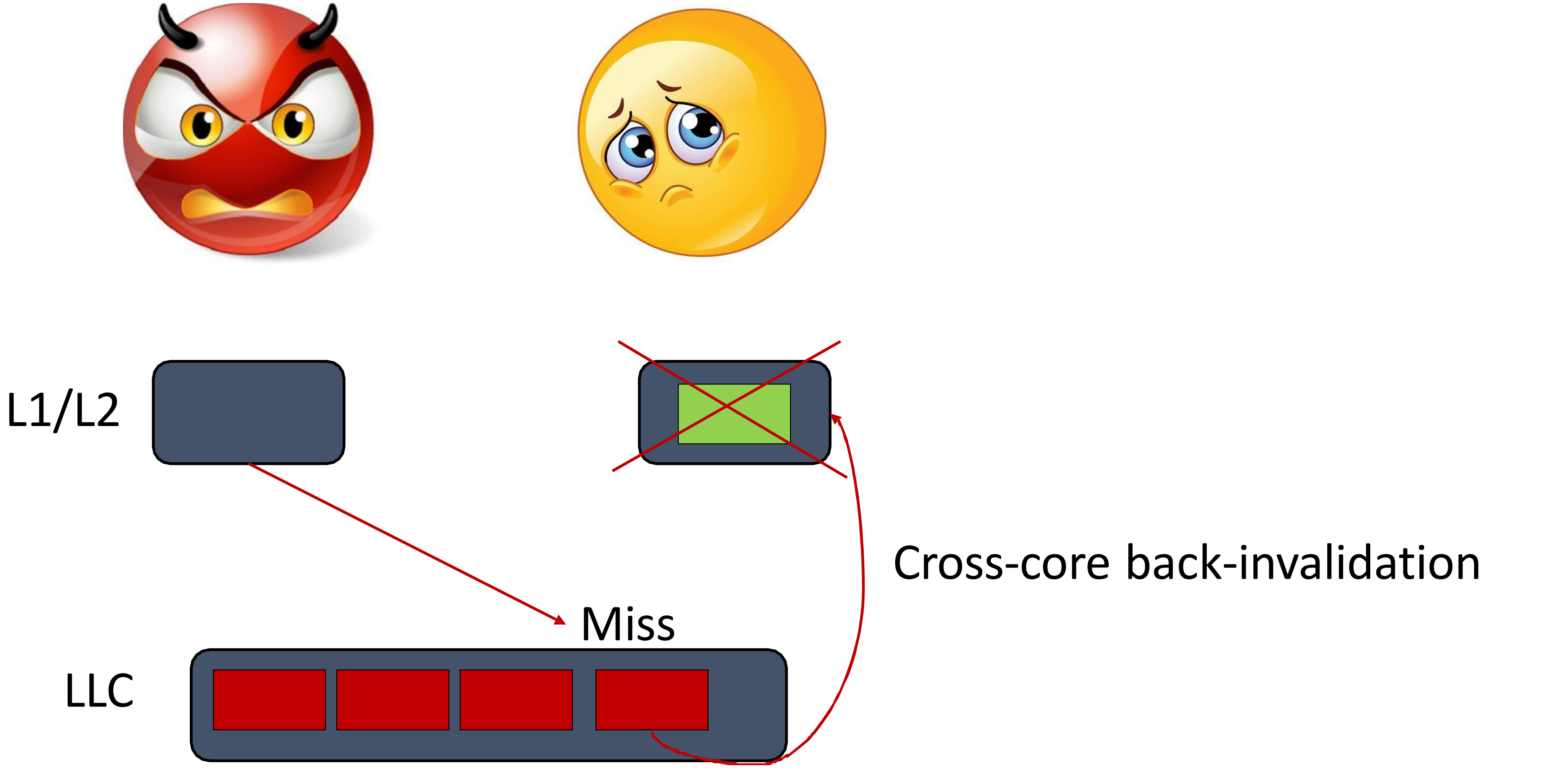
What if attacker’s access pattern is predictable?

Fool the prefetcher too.

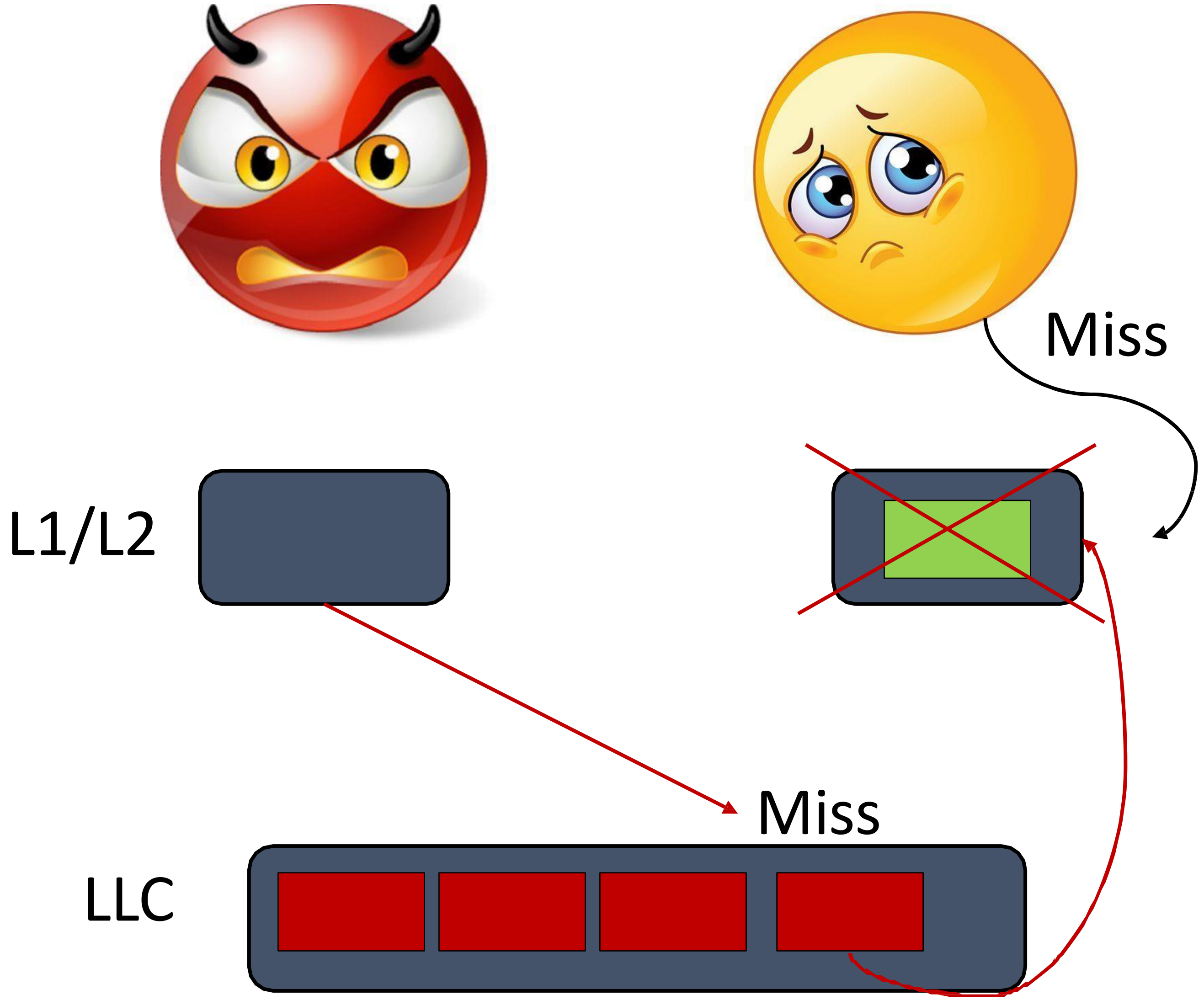
Inclusiveness helps



Inclusiveness



Inclusiveness



Attacker knows whether victim has accessed a set or not

Questions of interest

- HOW GOOD IS THE ATTACKER?
- ASSUMPTIONS
- AGILITY (BANDWIDTH)
- ADAPTIVE
- ACCURACY
- STEALTHY (DETECTOR CANNOT DETECT)

Pause

How Practical?

Future is uncertain, if we do not take care of present attacks, future may be worse ☹

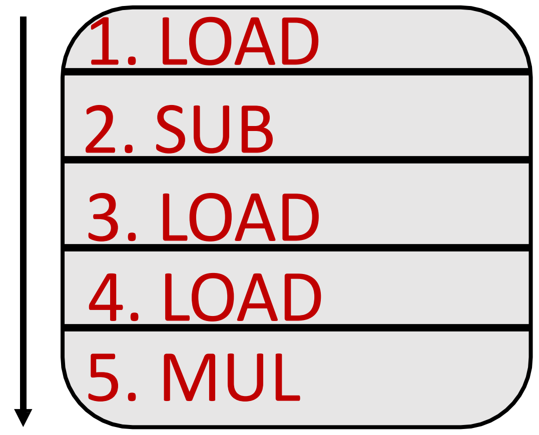
Transient Execution Attack

A speculative instruction may squash: Affects microarchitecture state

A transient instruction will squash (will not get committed)

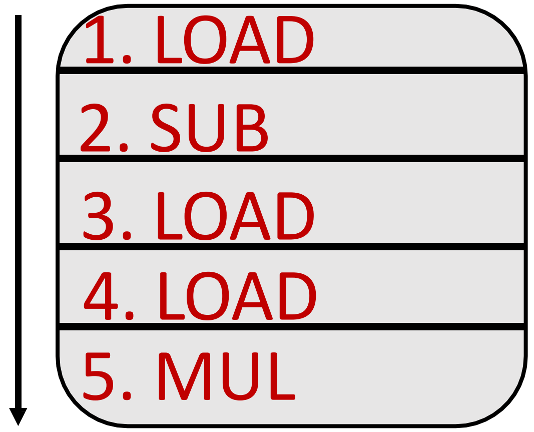
A non-transient instruction will not squash (eventually get committed/retired)

Modern Processors: In-order fetch

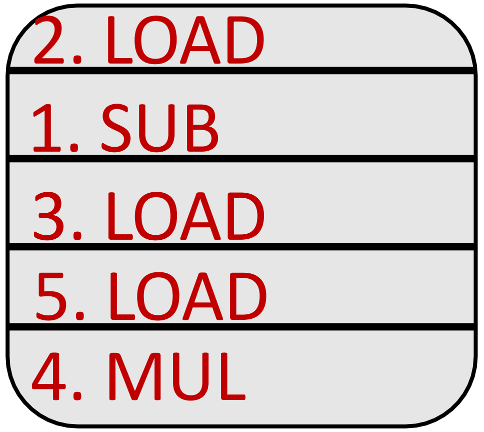


In-order Instruction Fetch
(Multiple fetch in one cycle)

Modern Processors: Out-of-order Execute

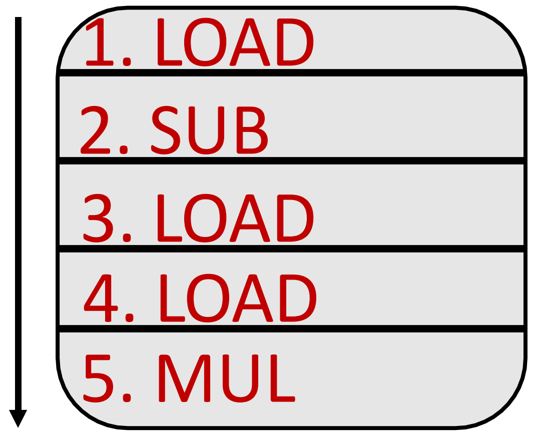


In-order Instruction Fetch
(Multiple fetch in one cycle)

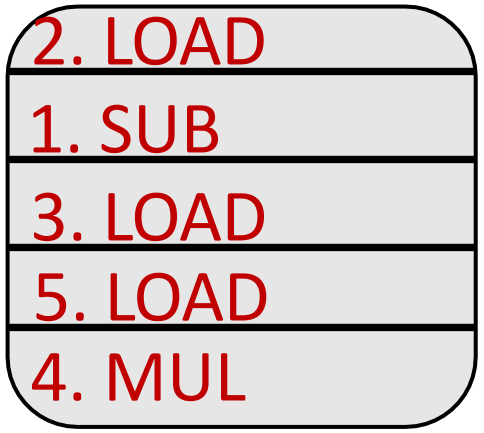


Out of order execute

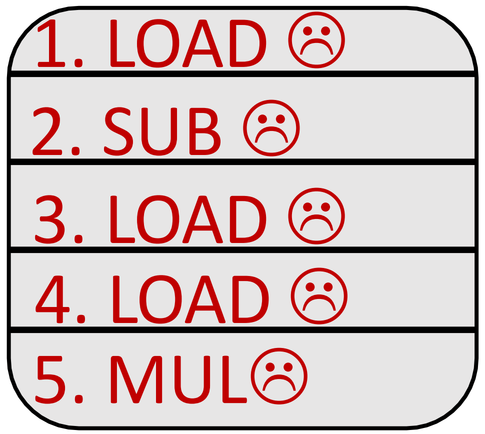
Modern Processors: In-order Commit



In-order Instruction Fetch
(Multiple fetch in one cycle)

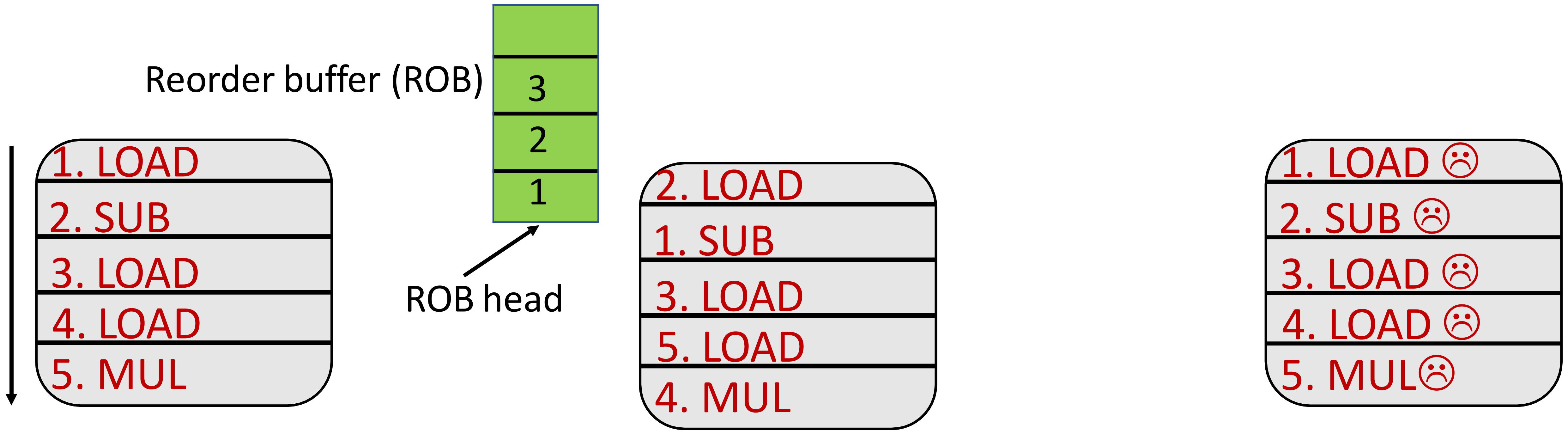


Out of order execute



In-order Completion
(commit)

Modern Processors: In-order Commit

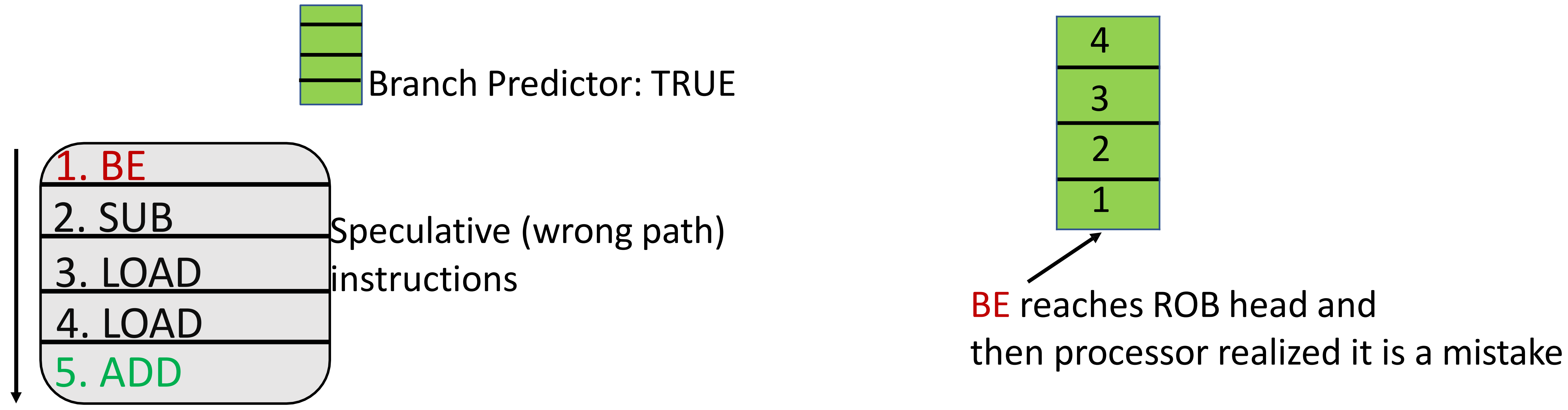


In-order Instruction Fetch
(Multiple fetch in one cycle)

Out of order execute

In-order Completion
(commit)

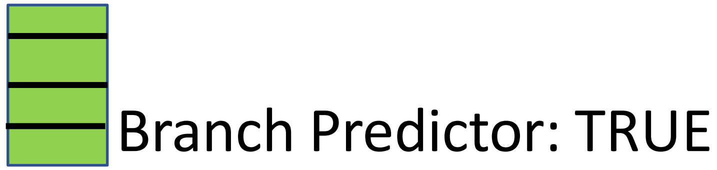
Modern Processors: Speculative Execution

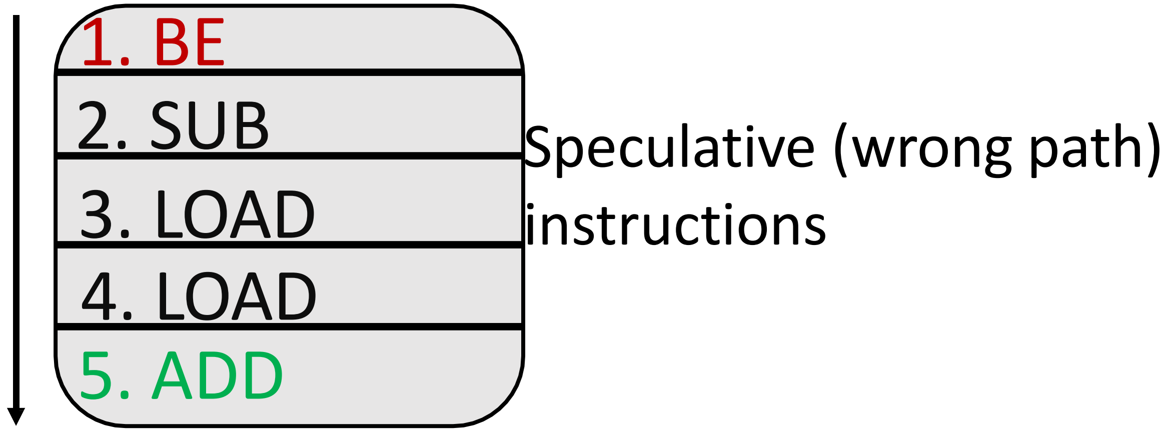

Branch Predictor: TRUE
1. BE
2. SUB
3. LOAD
4. LOAD
5. ADD
Speculative (wrong path) instructions
4
3
2
1
BE reaches ROB head and then processor realized it is a mistake

In-order Instruction Fetch
(Multiple fetch in one cycle)

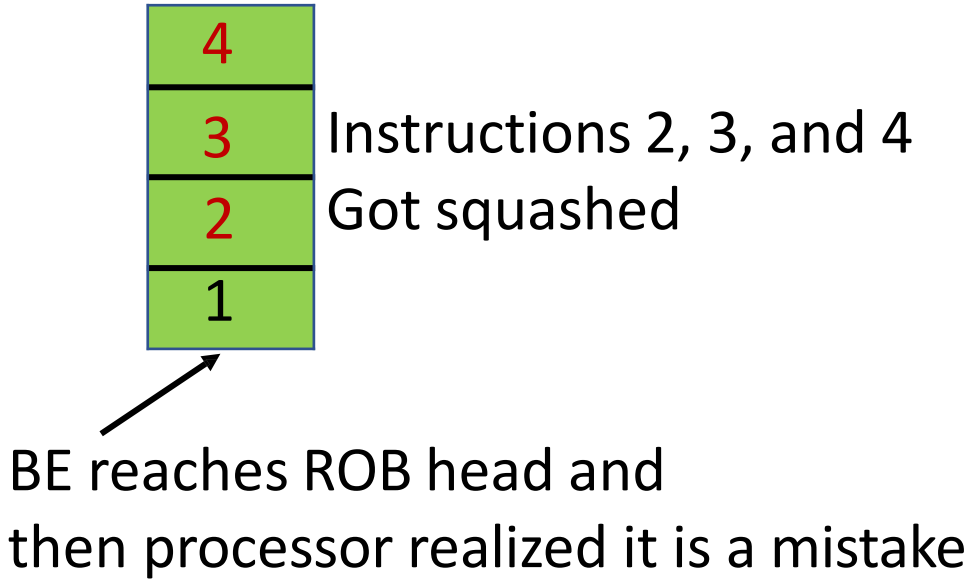
Recent Intel processors have 512-entry ROBs

Modern Processors: Speculative Execution





In-order Instruction Fetch
(Multiple fetch in one cycle)


4
3
2
1
Instructions 2, 3, and 4 Got squashed
BE reaches ROB head and then processor realized it is a mistake

Same happens in the case of a page fault, exception etc…

Spectre and Meltdown

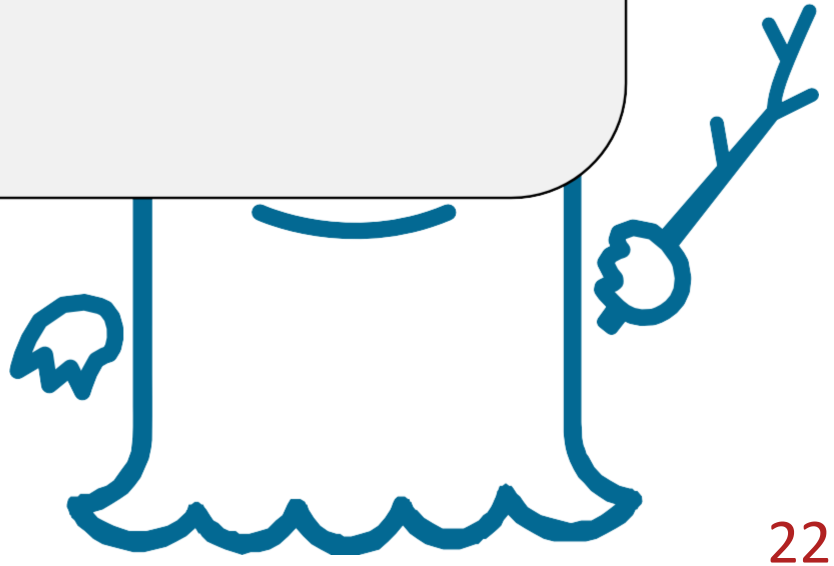
Spectre in Action: Fasten Your Seat Belts

```
int CS773Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS773Array))
        x = CS773Array[attacker]

y=MyArray[CS773Array[attacker]*512]
```

DRAM LOAD

DRAM LOAD



Branch Predictor and Speculative Execution

```
int CS773Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS773Array))
        y=MyArray[CS773Array[attacker]*512]
```

Branch predictor returns TRUE ☹

Branch Predictor and Speculative Execution

```
int CS773Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS773Array))
        y=MyArray[CS773Array[attacker]*512]
```

Branch predictor returns TRUE ☹

T T T T T T T T T T T

Attacker has mis-trained it ☹ ☹

How? By using values less than 3 always ☹ ☹

Branch Predictor and Speculative Execution

```
int CS773Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS773Array))
        y=MyArray[CS773Array[attacker]*512]
```

Branch predictor returns TRUE ☹

Attacker has mis-trained it ☹ ☹

Processor is on the wrong-path ☹ ☹ ☹

Branch Predictor and Speculative Execution

```
int CS773Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS773Array))
        y=MyArray[CS773Array[attacker]*512]
```

Branch predictor returns TRUE ☹

Attacker has mis-trained it ☹ ☹

Processor is on the wrong-path ☹ ☹ ☹

Branch resolution latency 200 cycles ☹ ☹ ☹ ☹

Within these 200 cycles ☺

```
int CS773Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS773Array))
        y=MyArray[CS773Array[attacker]*512]
```

CS773Array[4] is in L1/L2/L3 ☹

The address is in the cache ☹ ☹

Yes, you guessed it right: F+R, P+P cache attacks ☹ ☹ ☹

After say 200 cycles

Processor realized it was a mistake and *squashed* all wrong path instructions

But cache has the data ☹

y=MyArray[CS773Array[attacker]*512]

LOAD MyArray[0]   60 ns

LOAD MyArray[512] 60 ns

LOAD MyArray[1024] 5 ns Bingo !! *CS773Array[attacker] = 2*

Meltdown: The O3 Curse!!

```
1.  raise_exception();
2.  // line below is never reached
3.  secret=KernelArray[data*4096];
```

Kernel Trap

Out-of-order (O3) as it has no dependency

```
1.  secret=KernelArray[data*4096];
2. raise_exception();
```

What about page-fault?

Readings

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