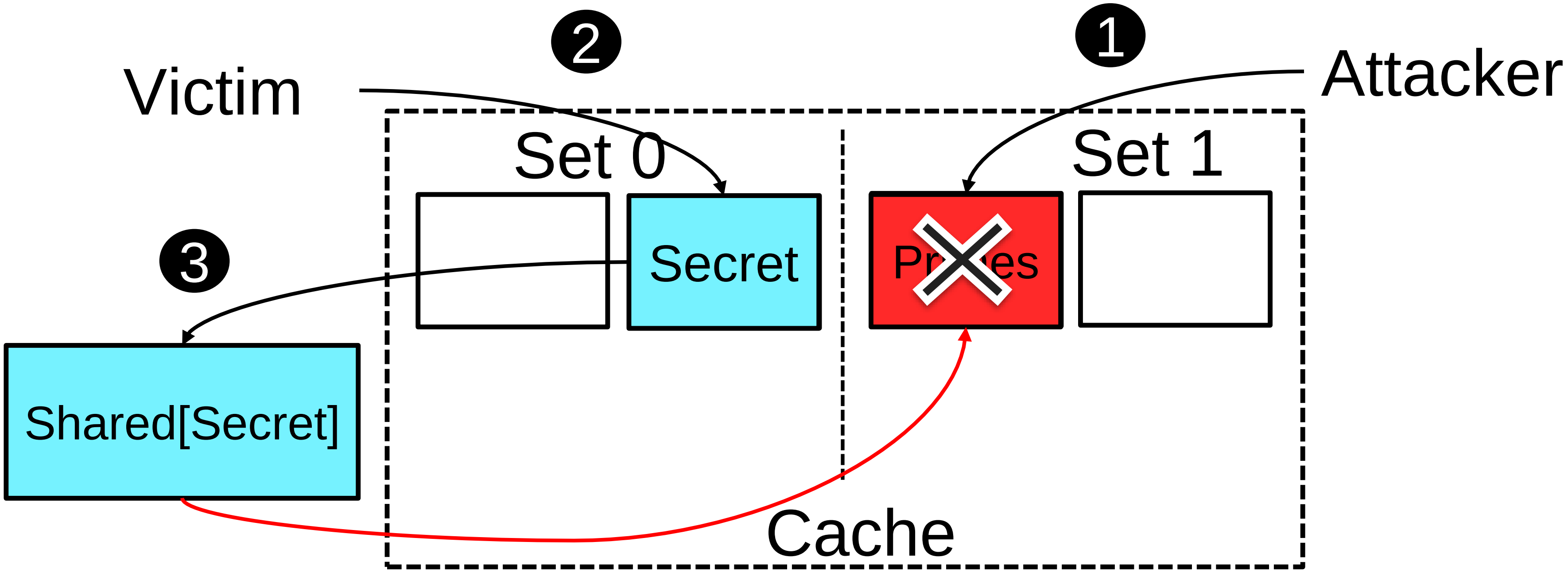
GhostMinion: A Strictness-Ordered Cache System for Spectre Mitigation[1]

Sumon Nath
sumon@cse.iitb.ac.in

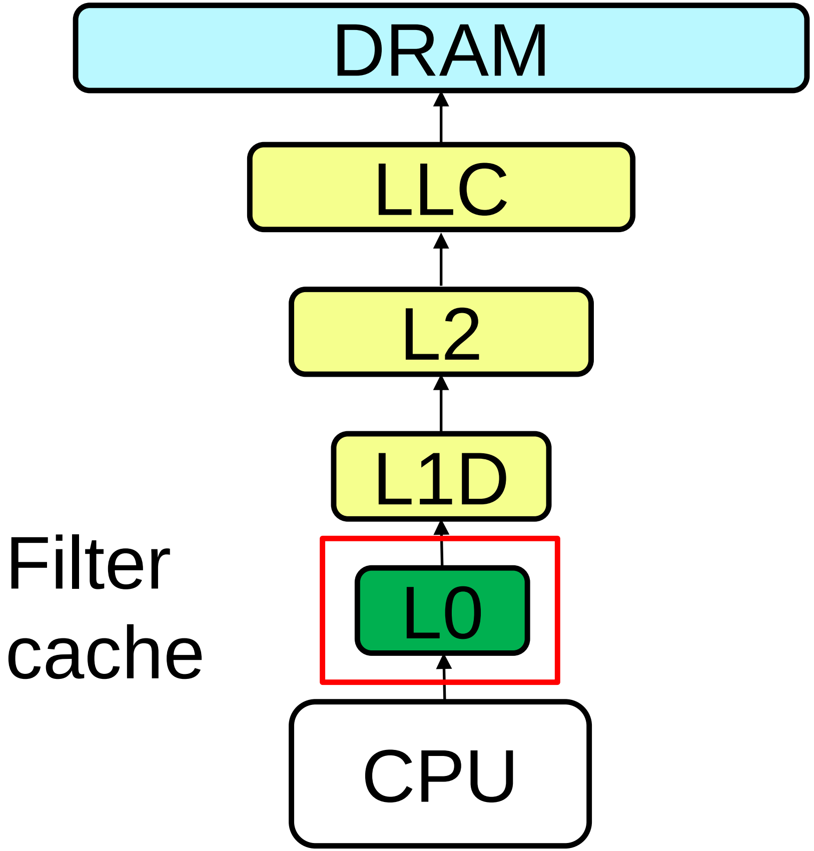
[1] Ainsworth, Sam. "GhostMinion: A strictness-ordered cache system for Spectre mitigation." In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 592-606. 2021.

Recap- Spectre



Problem: Access to speculative data across **domain boundaries**

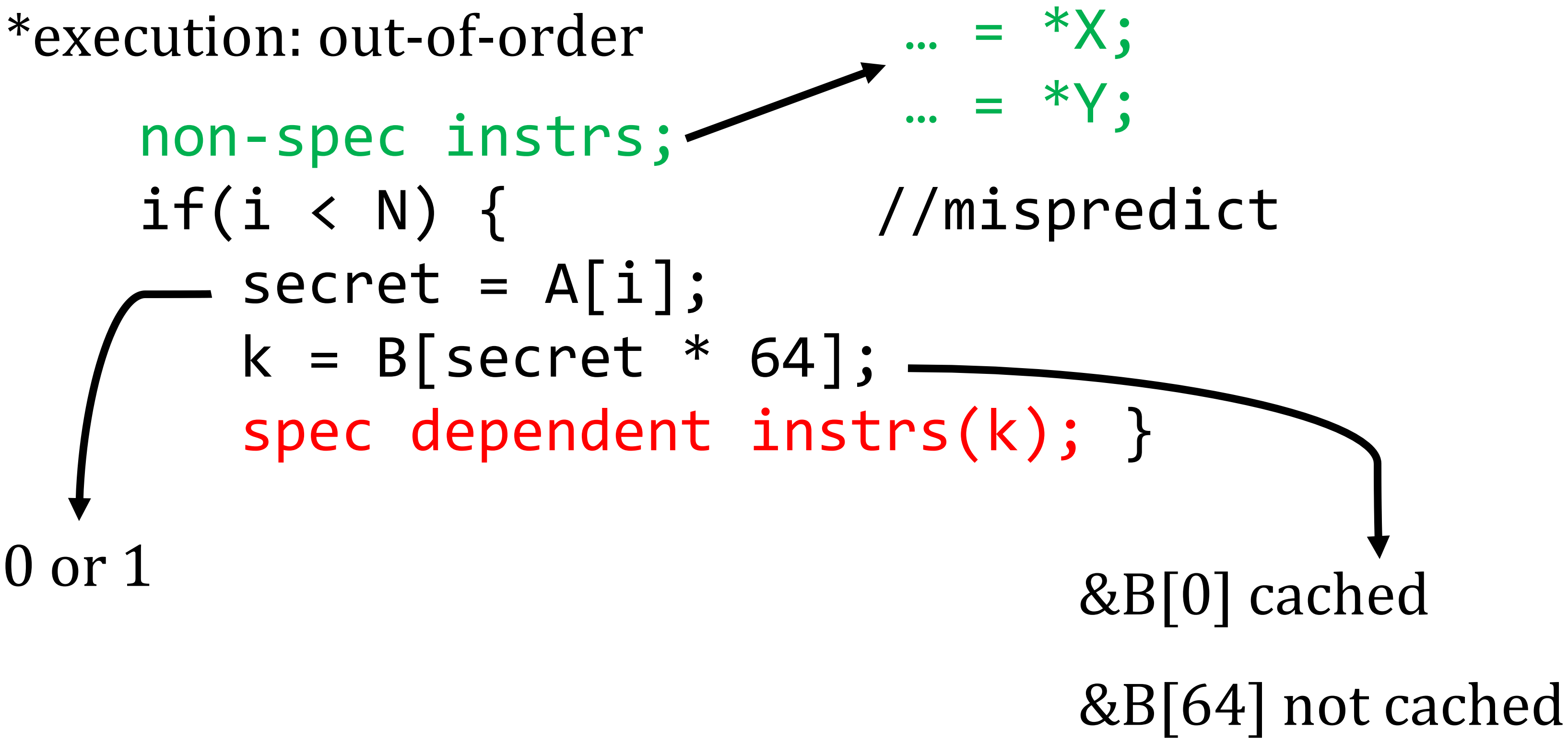
Recap- MuonTrap



- Stores all speculative data
- Wiped on context switch
- Non-inclusive/non-exclusive
- Instruction cache, TLBs

Does this work always?

Attack scenario



*execution: out-of-order

```
non-spec instrs;                 … = *X;
                                 … = *Y;
if(i < N) {                      //mispredict
  secret = A[i];
  k = B[secret * 64];
  spec dependent instrs(k); }
```

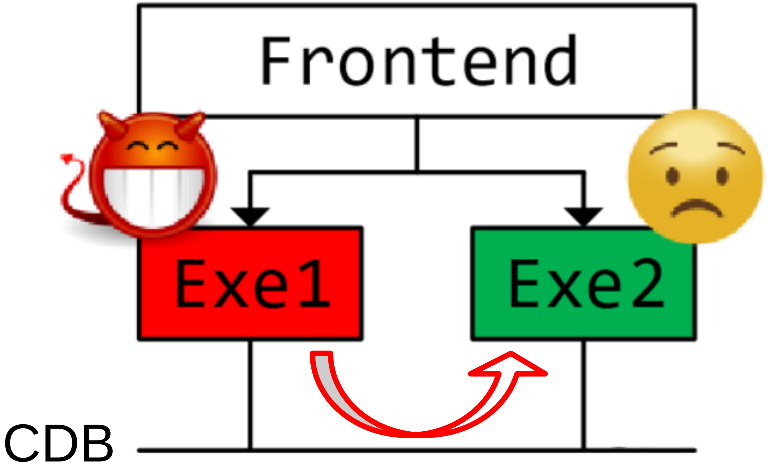
0 or 1

&B[0] cached

&B[64] not cached

Execution unit contention

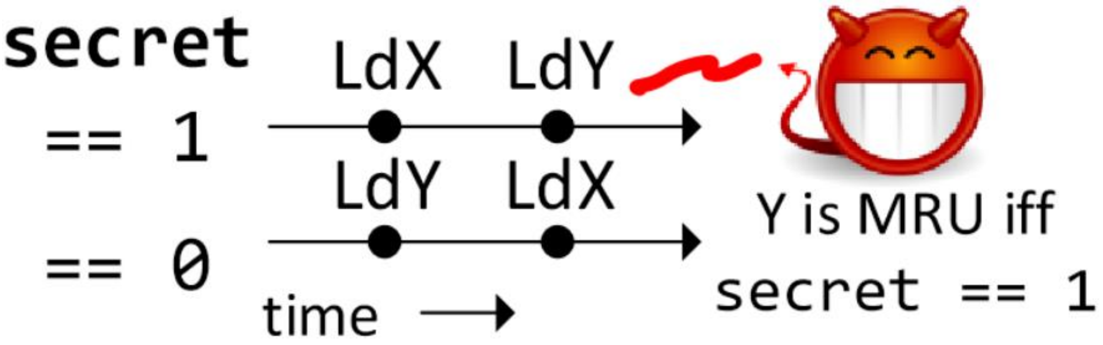
```
non-spec instrs;
if(i < N) {                  //mispredict
   secret = A[i];
   k = B[secret * 64];
   spec dependent instrs(k); }
```



Attack scenario

```
non-spec instrs;
if(i < N) {           //mispredict
   secret = A[i];
   k = B[secret * 64];
   spec dependent instrs(k); }
```

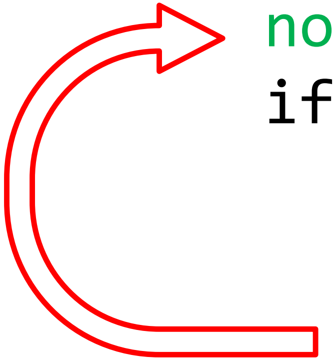
Speculative Interference[2]



Y is MRU iff secret == 1

[1] Behnia, Mohammad, Prateek Sahu, Riccardo Paccagnella, Jiyong Yu, Zirui Neil Zhao, Xiang Zou, Thomas Unterluggauer et al. "Speculative interference attacks: Breaking invisible speculation schemes." In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 1046-1060. 2021.

Does MuonTrap work?



```
non-spec instrs;
if(i < N) {                //mispredict
   secret = A[i];
   k = B[secret * 64];
   spec dependent instrs(k); }
```

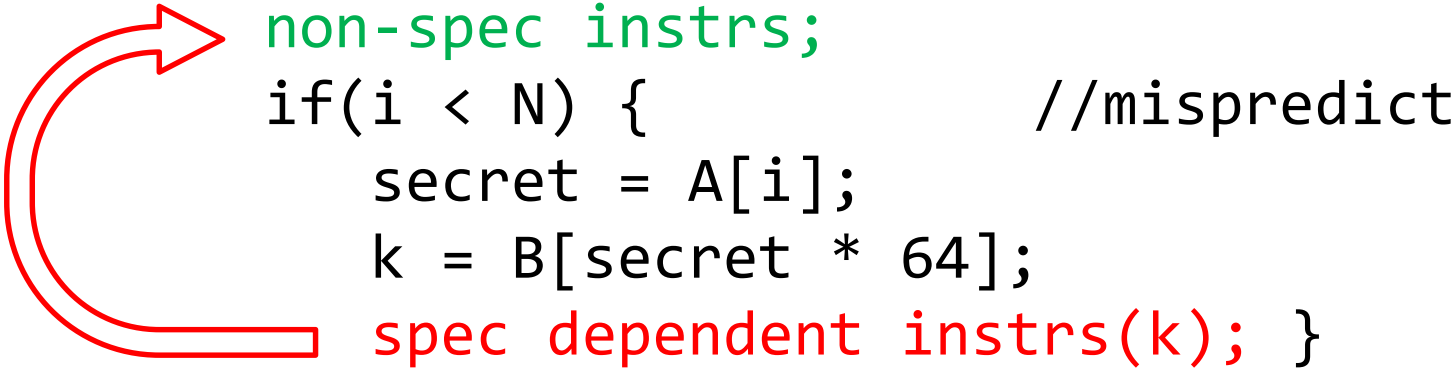
Does not take care of
backwards-in-time channels

No!

Till now…

- Spectre
- MuonTrap
- Speculative Interference
- Why MuonTrap does not work?
- Next: GhostMinion

What can be done?



```
non-spec instrs;
if(i < N) {                   //mispredict
   secret = A[i];
   k = B[secret * 64];
   spec dependent instrs(k); }
```

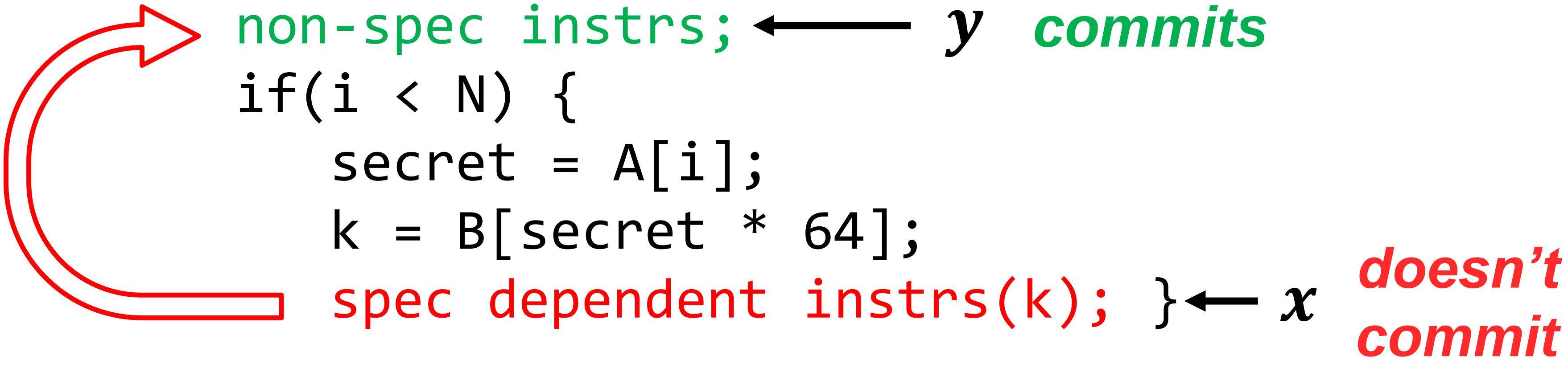
We need to restrict backward-in-time channels

How?

Strictness ordering

x **can impact timing of** y, iff

$$\mathbf{commit}(y) \cancel{\rightarrow} \mathbf{commit}(x)$$



```
non-spec instrs;
if(i < N) {
    secret = A[i];
    k = B[secret * 64];
    spec dependent instrs(k); }
```

y ***commits***

x ***doesn't commit***

Temporal ordering

*strictness ordering hard to implement

$\boldsymbol{x}$ **can impact timing of** $\boldsymbol{y}$**,** $\boldsymbol{iff}$

$$\mathbf{commit}(\boldsymbol{x}) \vee \mathbf{seq}(\boldsymbol{x}, \boldsymbol{y})$$

Does it ensure strictness ordering? 🤔

yes! $\mathbf{seq}(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \mathbf{commit}(\boldsymbol{y}) \rightarrow \mathbf{commit}(\boldsymbol{x})$

The picture can't be displayed.

Implementation

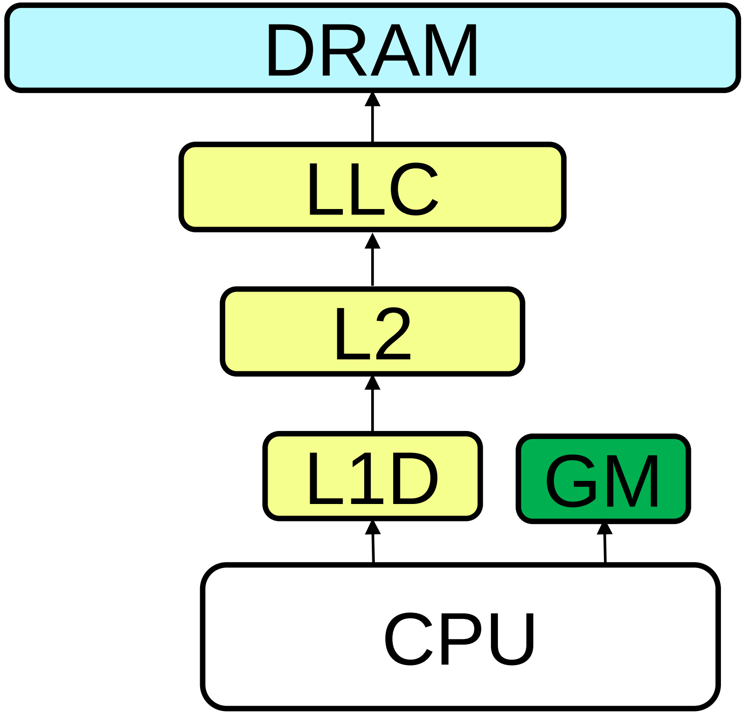
GhostMinion - cache system

Techniques:

- TimeGuarding
- Free-slotting
- LeapFrogging

*Applied to other microarchitecture structures

GhostMinion

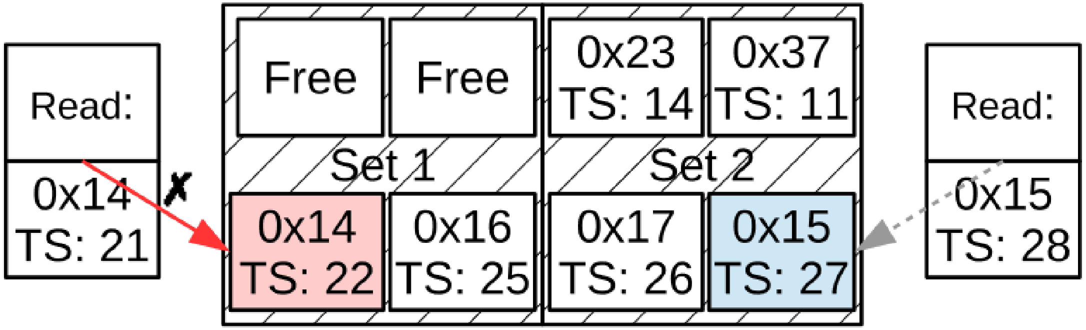


- Same as MuonTrap
- Accessed in parallel to L1D

TimeGuarding

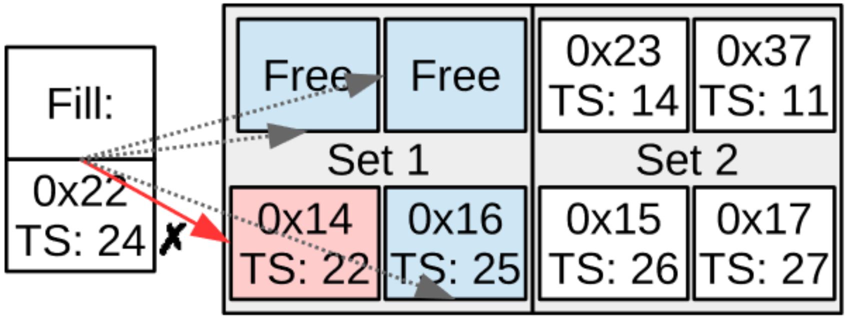
Ensures temporal ordering

Read



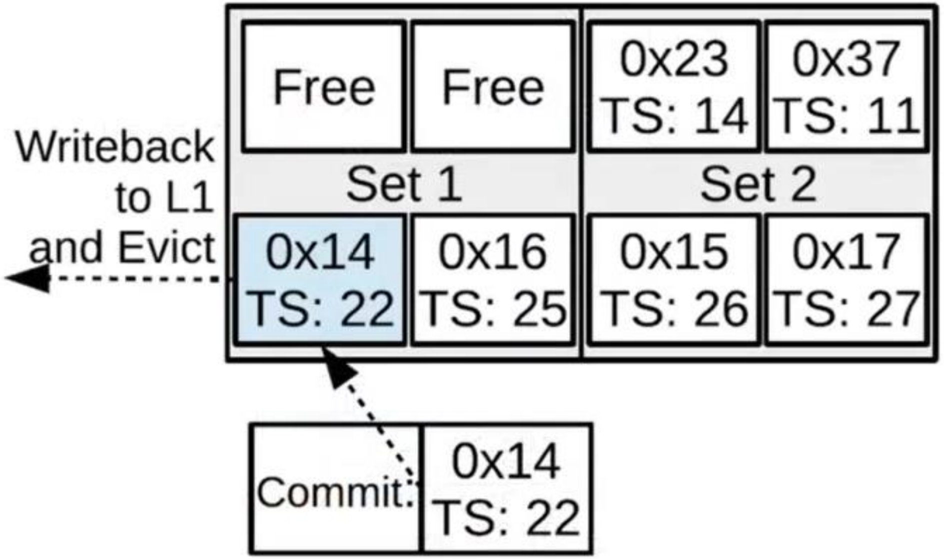
Read:
0x14
TS: 21
✗
Free
Free
0x23
TS: 14
0x37
TS: 11
Set 1
Set 2
0x14
TS: 22
0x16
TS: 25
0x17
TS: 26
0x15
TS: 27
Read:
0x15
TS: 28

Fill



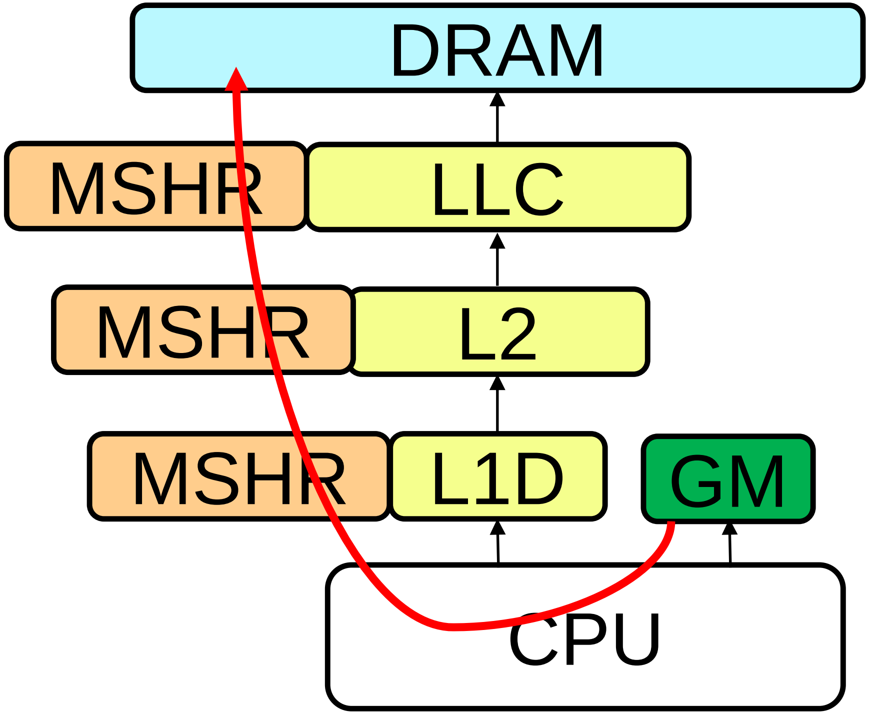
Fill:
0x22
TS: 24
✗
Free
Free
0x23
TS: 14
0x37
TS: 11
Set 1
Set 2
0x14
TS: 22
0x16
TS: 25
0x15
TS: 26
0x17
TS: 27

Free-slotting



Avoids resource starvation

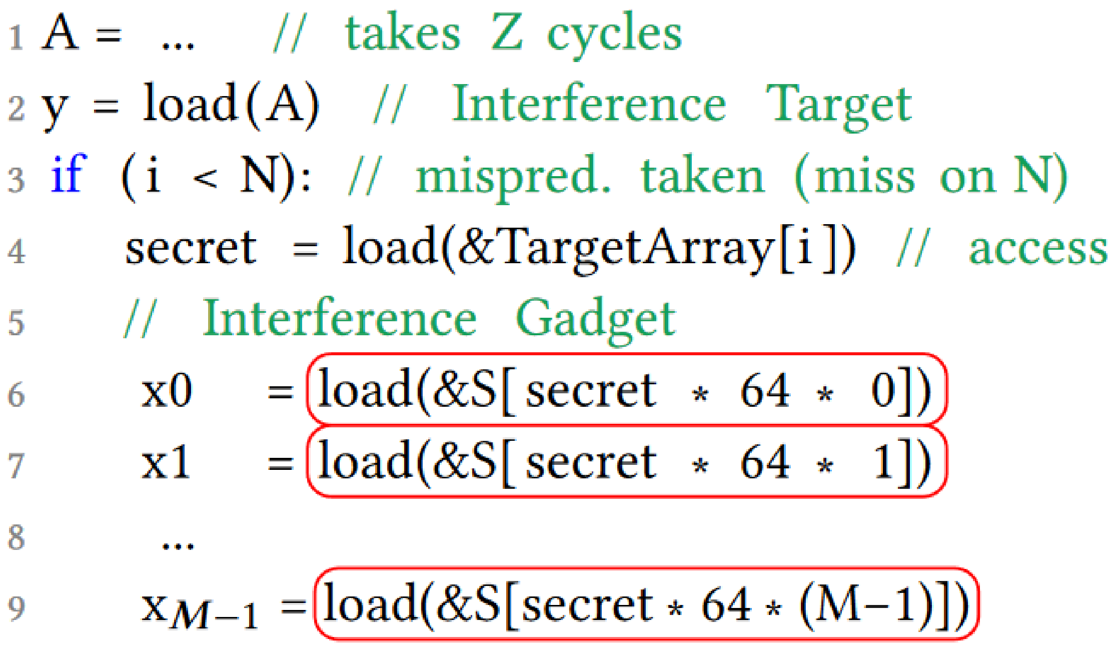
LeapFrogging



- MSHR:
  - stores miss status
  - enables parallel misses

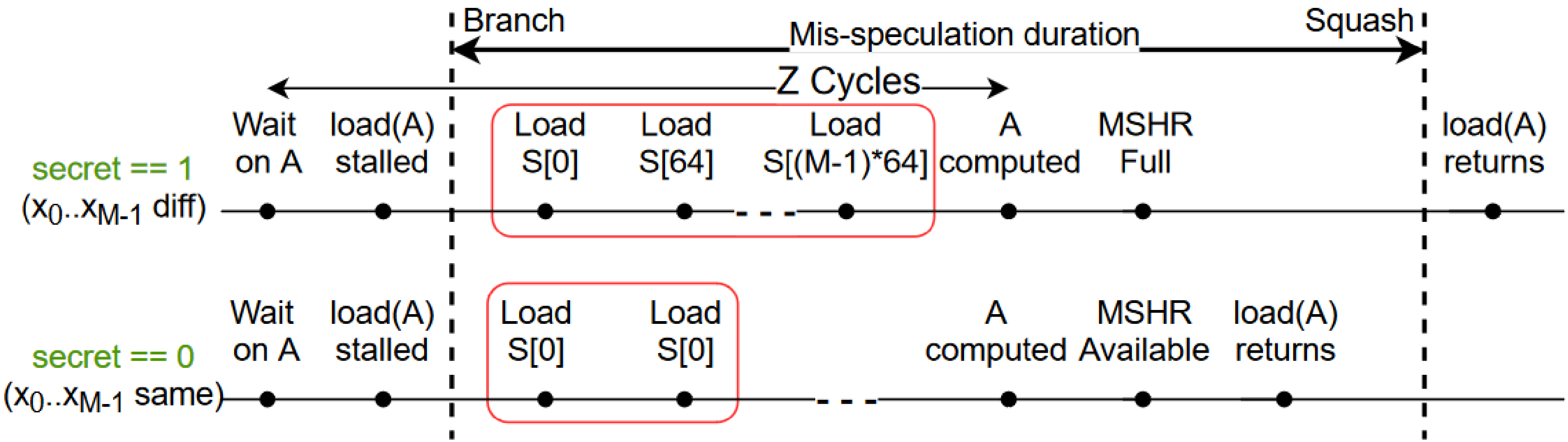
- Non-inclusive/ non-exclusive cache

Attack scenario



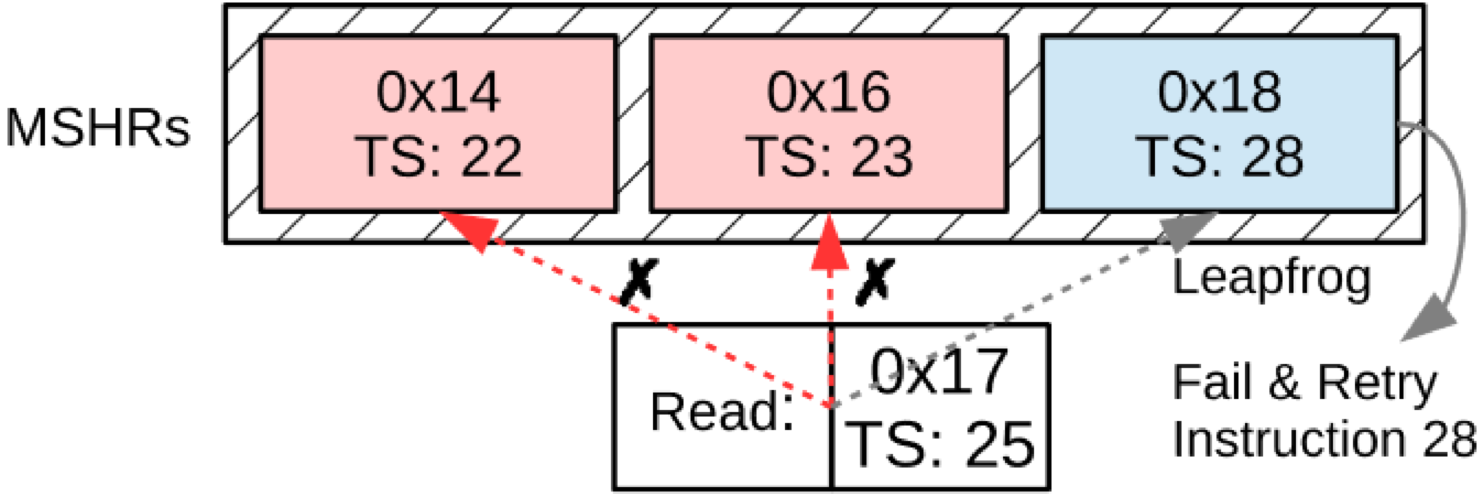
```
A =  ...     // takes Z cycles
y = load(A)   // Interference Target
if (i < N):  // mispred. taken (miss on N)
    secret = load(&TargetArray[i])  // access
    // Interference Gadget
     x0      = load(&S[secret * 64 * 0])
     x1      = load(&S[secret * 64 * 1])
     ...
     x_{M-1} = load(&S[secret * 64 * (M-1)])
```

Attack scenario



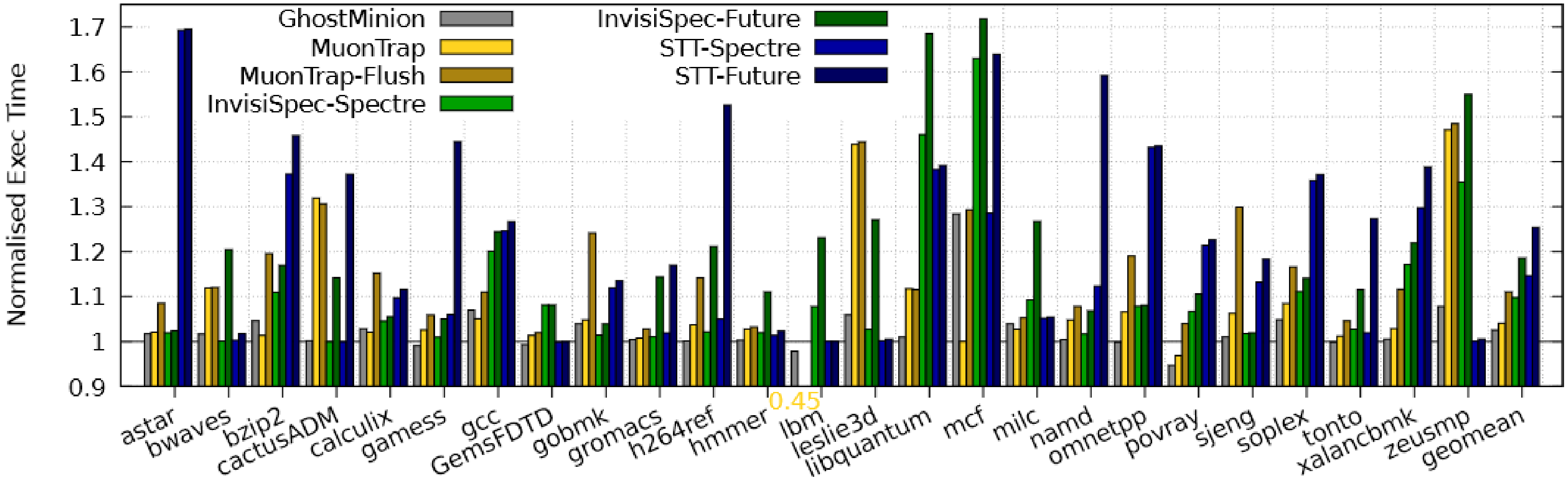
violates strictness ordering

LeapFrogging



Older instruction can kick out MSHR entry of newer instruction

Performance evaluation: SPEC 2006



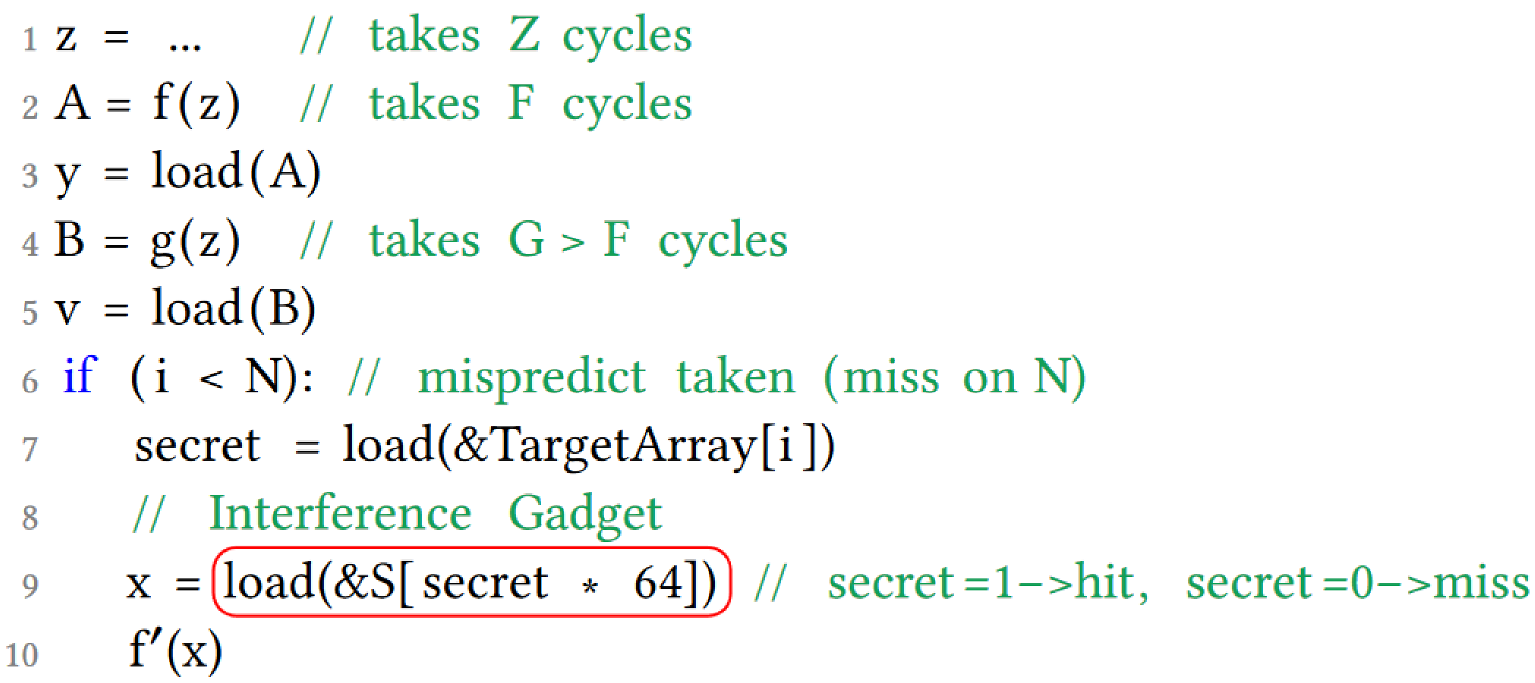
- GhostMinion 2.5% slowdown compared to 4% slowdown for MuonTrap

Conclusion

- MuonTrap was the first to solve Spectre with comparatively low performance overhead.
- GhostMinion proposes a precise framework to avoid Spectre and its different variants.

Thank You!

Speculative Interference



```
z =  ...       // takes Z cycles
A = f(z)    // takes F cycles
y = load(A)
B = g(z)    // takes G > F cycles
v = load(B)
if (i < N):  // mispredict taken (miss on N)
    secret = load(&TargetArray[i])
    // Interference Gadget
    x = load(&S[secret * 64]) // secret=1->hit, secret=0->miss
    f′(x)
```

contention on a non-pipelined EU. Instruction sequences `f` and `f′` use the same non- pipelined EU. Instruction sequence g uses a different EU