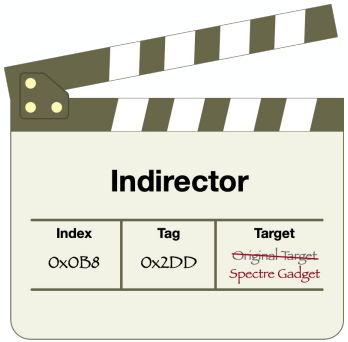


Indirector: High-Precision Target Injection Attacks Exploiting the Indirect Branch Predictor

Luyi Li*, Hosein Yavarzadeh*, Dean Tullsen

* Equal Contribution

UC San Diego

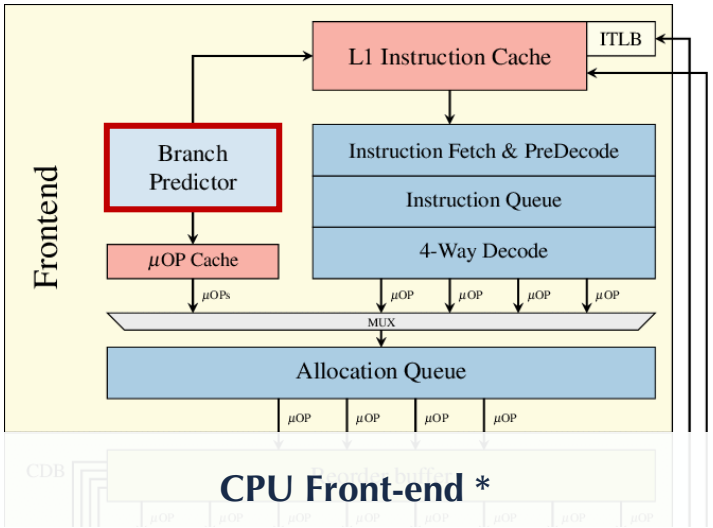
August 15, 2024

Indirector Website

Outline

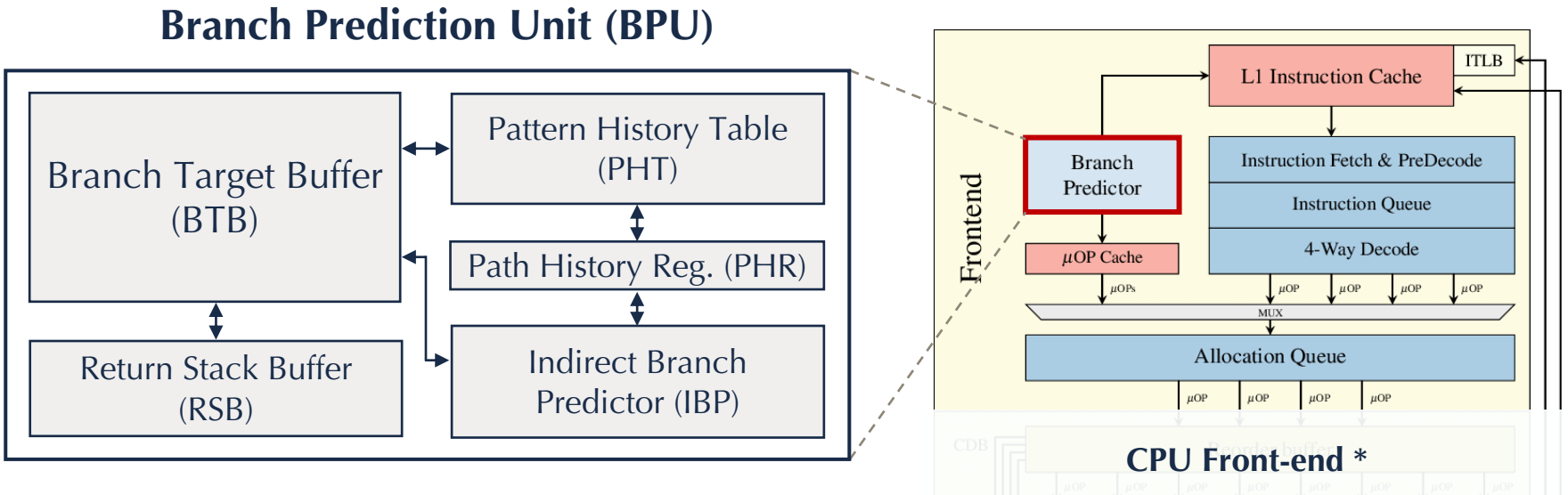
- **Background**
  - Branch Prediction
  - Branch Target Injection (BTI) Attack
- Motivation
- Indirect Branch Prediction
- Intel BTI Defense Analysis
- High-Precision Injection Attack

Branch Prediction Unit (BPU)

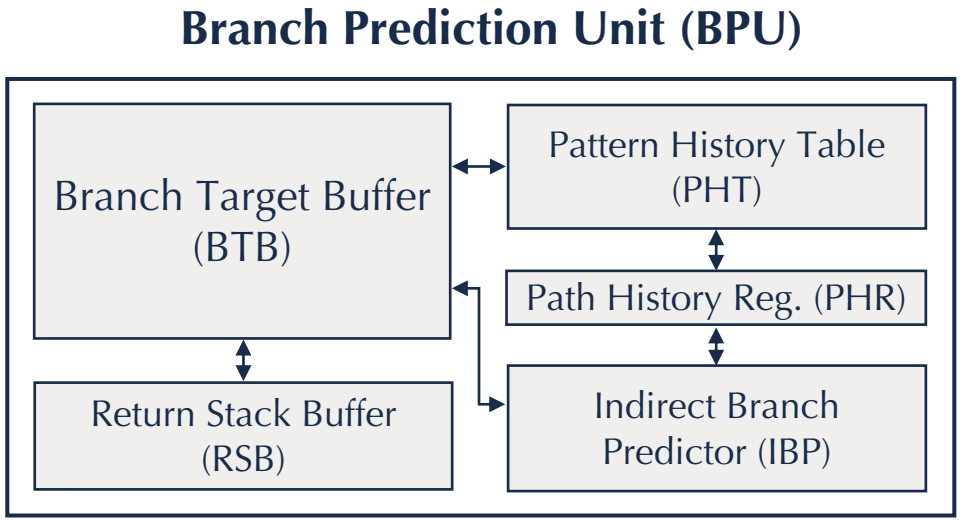


* Figure from the Meltdown paper. (Lipp, Moritz, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. "Meltdown." *arXiv preprint arXiv:1801.01207* (2018) .)

Branch Prediction Unit (BPU)



Branch Prediction Unit (BPU)




```
jmp Target
…
Target:
```

Direct Branch

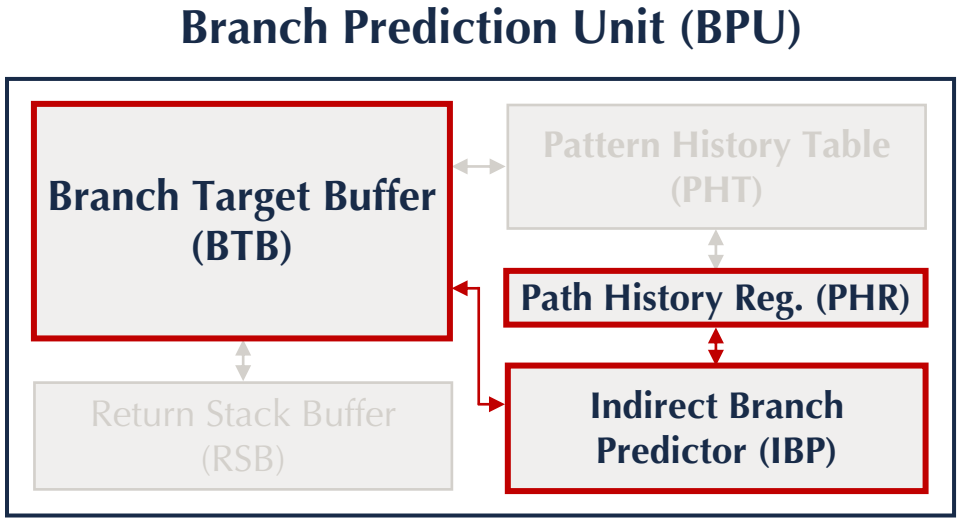
```
je Target
…
Target:
```

Conditional Branch

```
jmp rax/[mem]
…
Target_0: …
Target_i: …
```

Indirect Branch

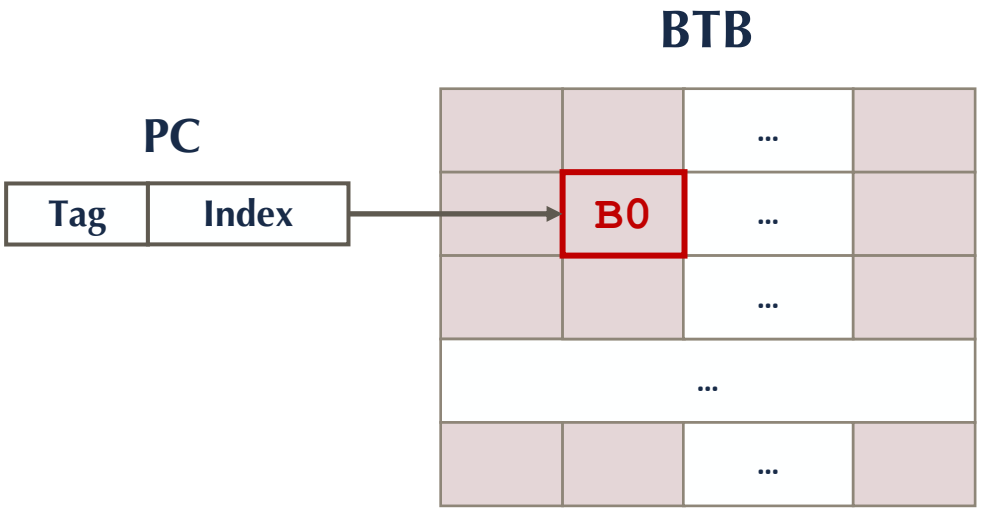
Today's Focus: Indirect Branch Prediction




```
  jmp rax/[mem]
  …
Target_0: …
Target_i: …
```

Indirect Branch

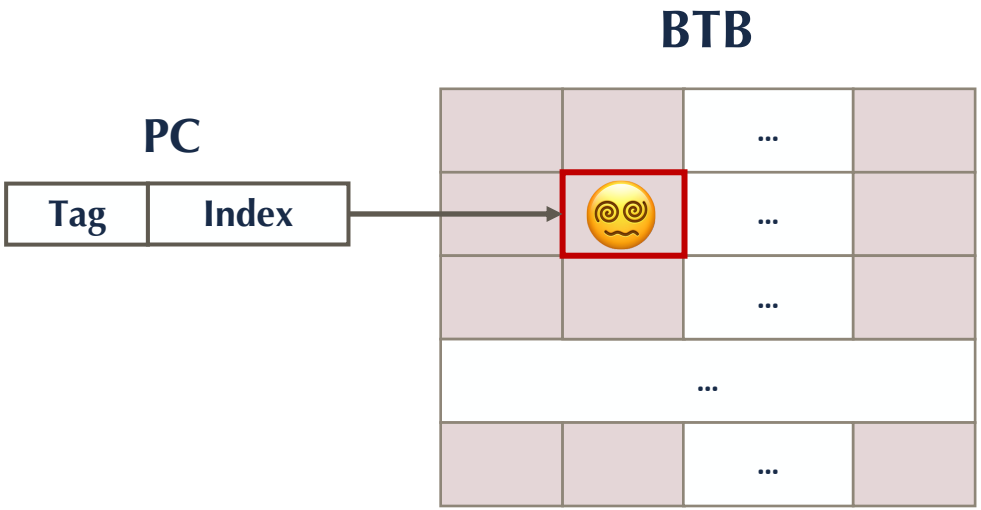
Step 1: Look Up in BTB




```
B0:jmp rax/[mem]
   ...
   Target_0: ...
   Target_i: ...
```

Indirect Branch

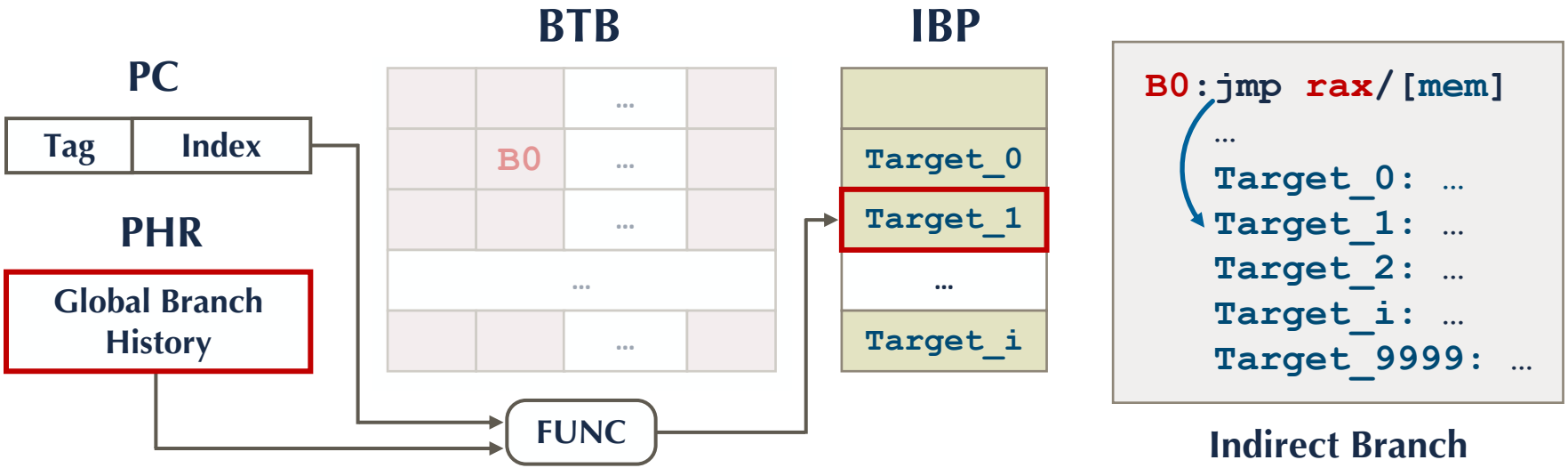
Step 1: Look Up in BTB



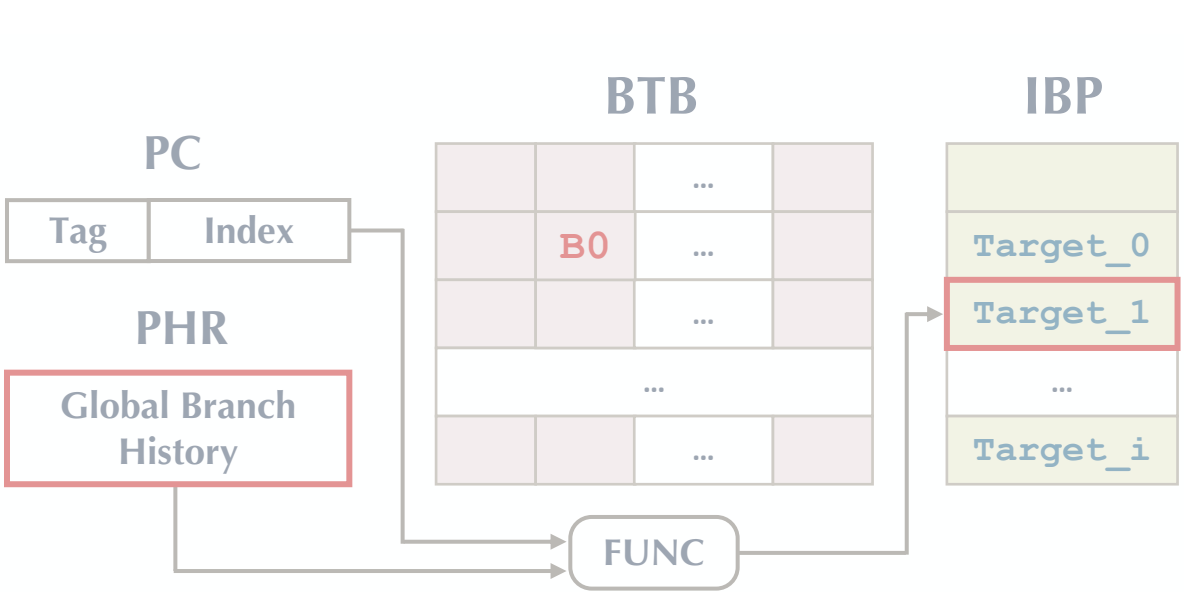

```
B0:jmp rax/[mem]
    ...
    Target_0: ...
    Target_1: ...
    Target_2: ...
    Target_i: ...
    Target_9999: ...
```

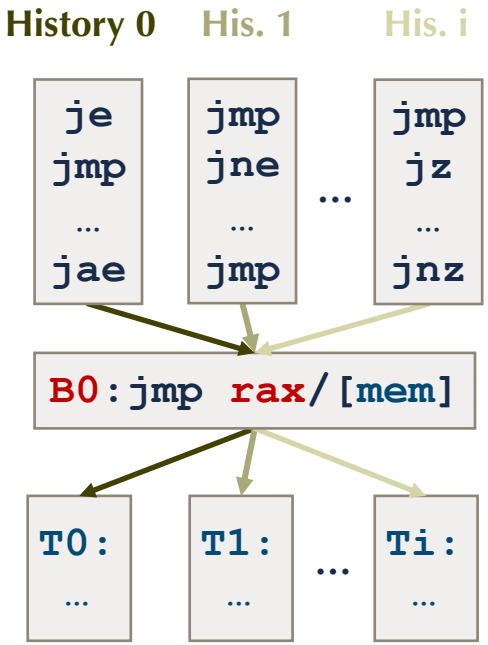
Indirect Branch

Step 2: Get Targets from IBP



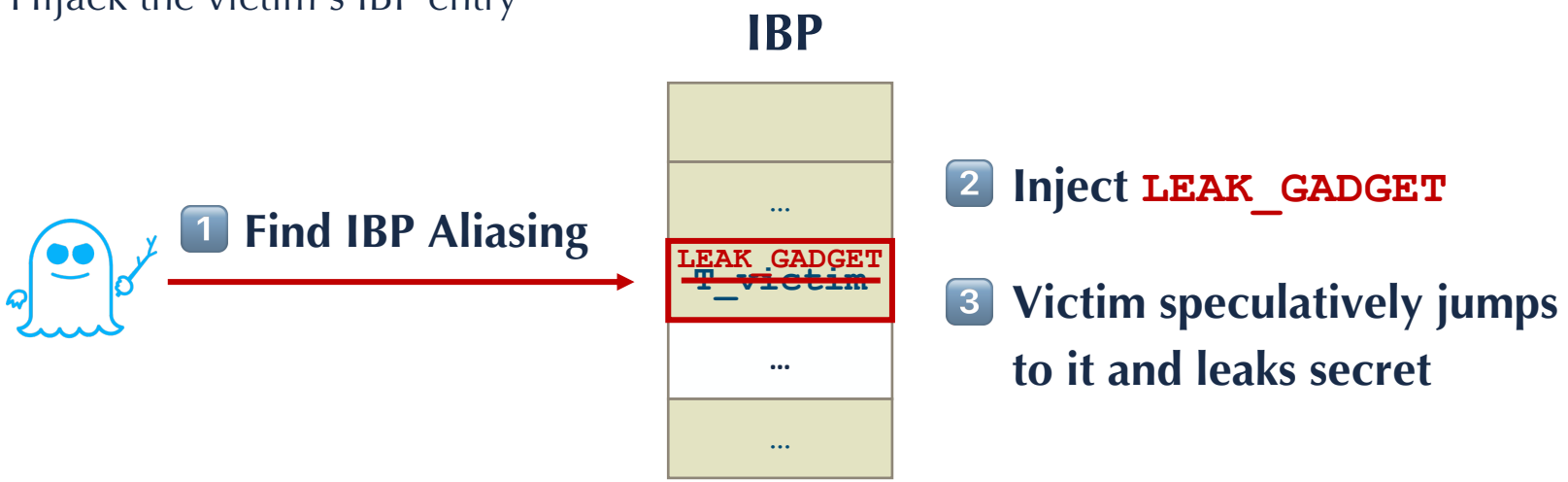
Step 2: Get Targets from IBP





Branch Target Injection Attack

- **Branch Target Injection** (BTI) (a.k.a. Spectre v2) (CVE-2017-5715)
  - Attack indirect branches
  - Hijack the victim's IBP entry

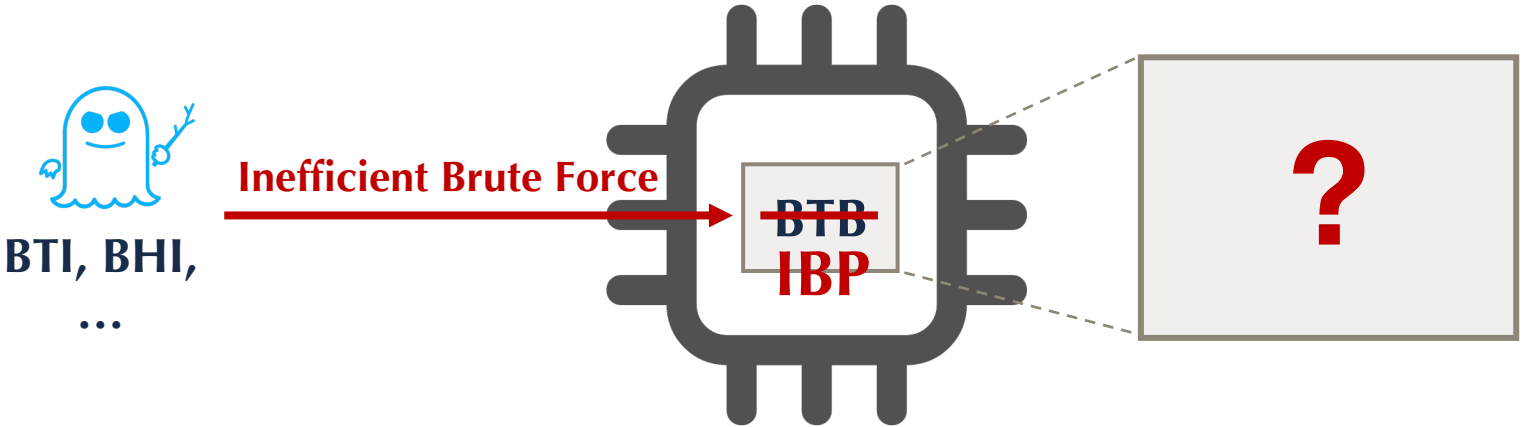


Outline

- Background
- **Motivation**
- Indirect Branch Prediction
- Intel BTI Defense Analysis
- High-Precision Injection Attack

Finding IBP Aliasing is Non-Trivial

- IBP designs in modern CPUs are **complex** and **undocumented**
- IBP structure is **overlooked** by previous works:



Understanding IBP is Important

From an Attacker's Perspective:

- **Understand** existing injection attacks better
- **Launch** injection attacks with greater efficiency
- **Discover** new injection surfaces inside BPU

Understanding IBP is Important

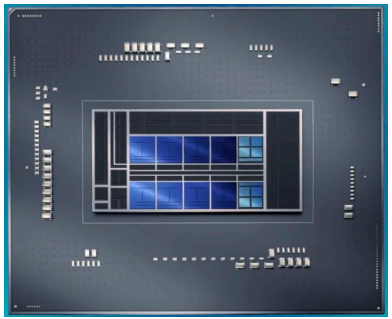
From a Defender's Perspective:

- **Deconstruct** Intel BTI defenses for the first time
- **Increase** the frequency of current defenses
  - Faster attack -> More frequent defense -> **Higher overhead!**
- **Inspire** future defenses and secure BPU designs

Outline

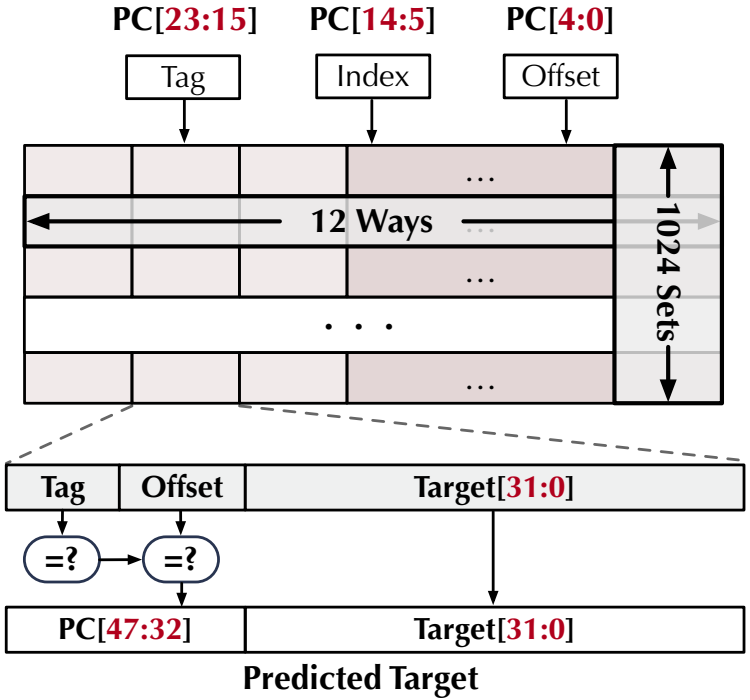
- Background
- Motivation
- **Indirect Branch Prediction**
  - Branch Target Buffer (BTB)
  - Path History Register (PHR)
  - Indirect Branch Predictor (IBP)
- Intel BTI Defense Analysis
- High-Precision Injection Attacks

Reverse Engineering Intel CPU



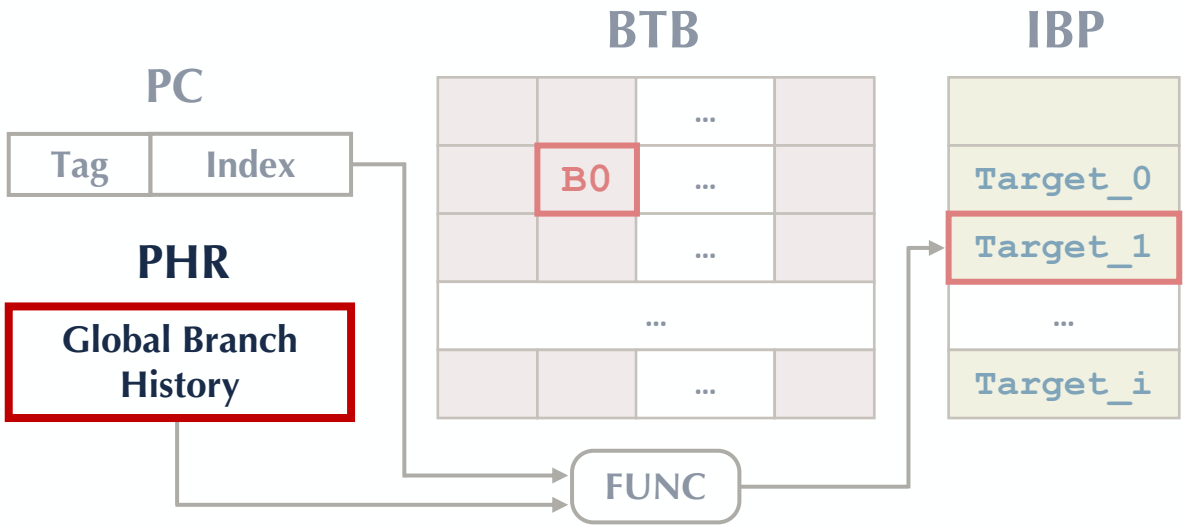
Golden/Raptor Cove (2021/2022)

Reverse Engineering BTB



- **Fundamental BTB details:**
  - **1024** sets, **12** ways
  - **PC[23:0]** used as input
    - One entry for one branch
  - **32-bit** absolute target

Quick Recall - PHR

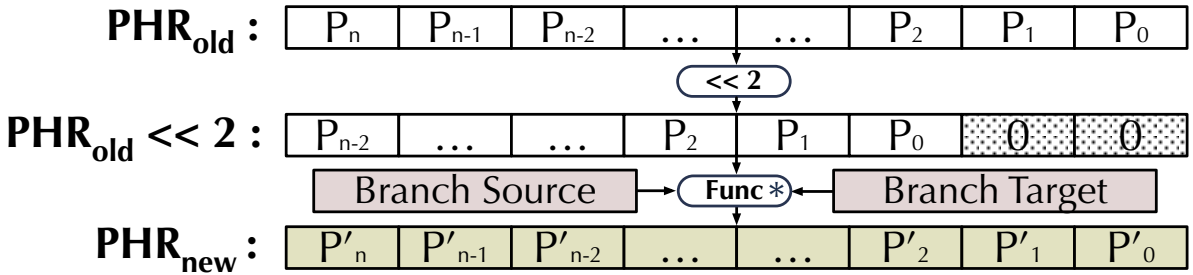


Reverse Engineering PHR

- **Previous works have revealed PHR for PHT (Conditional Branch Prediction)**
  - PHR for IBP is constructed in the same way
  - All types of **taken** branches will update PHR:

Half & Half S&P'23

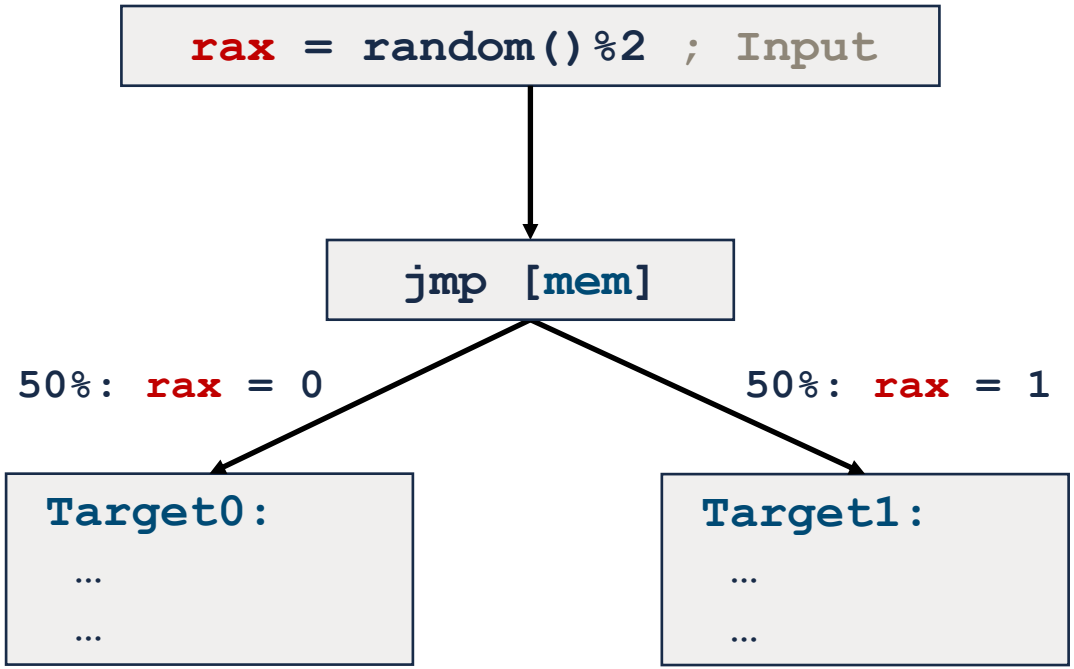
PATH FINDER ASPLOS'24



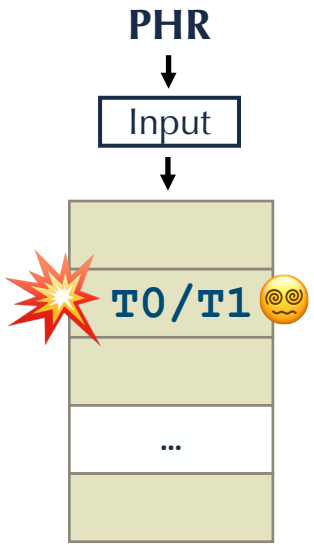
- **We can set PHR to any desired value!**

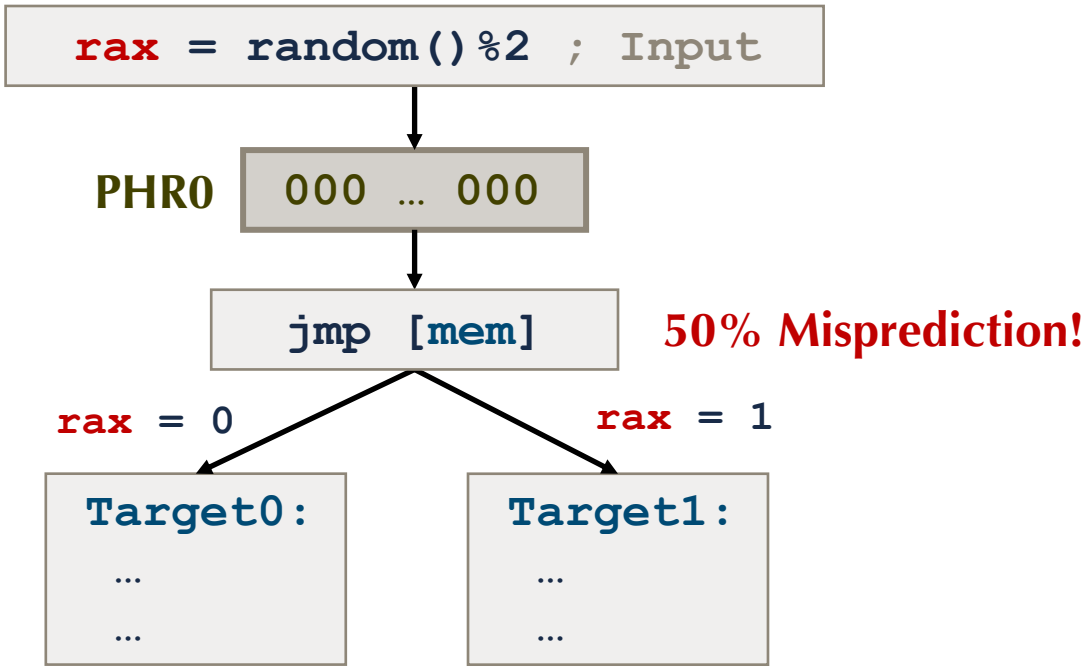
*The detailed PHR update function is in Appendix A of the paper.

How Does PHR Help IBP Predict?

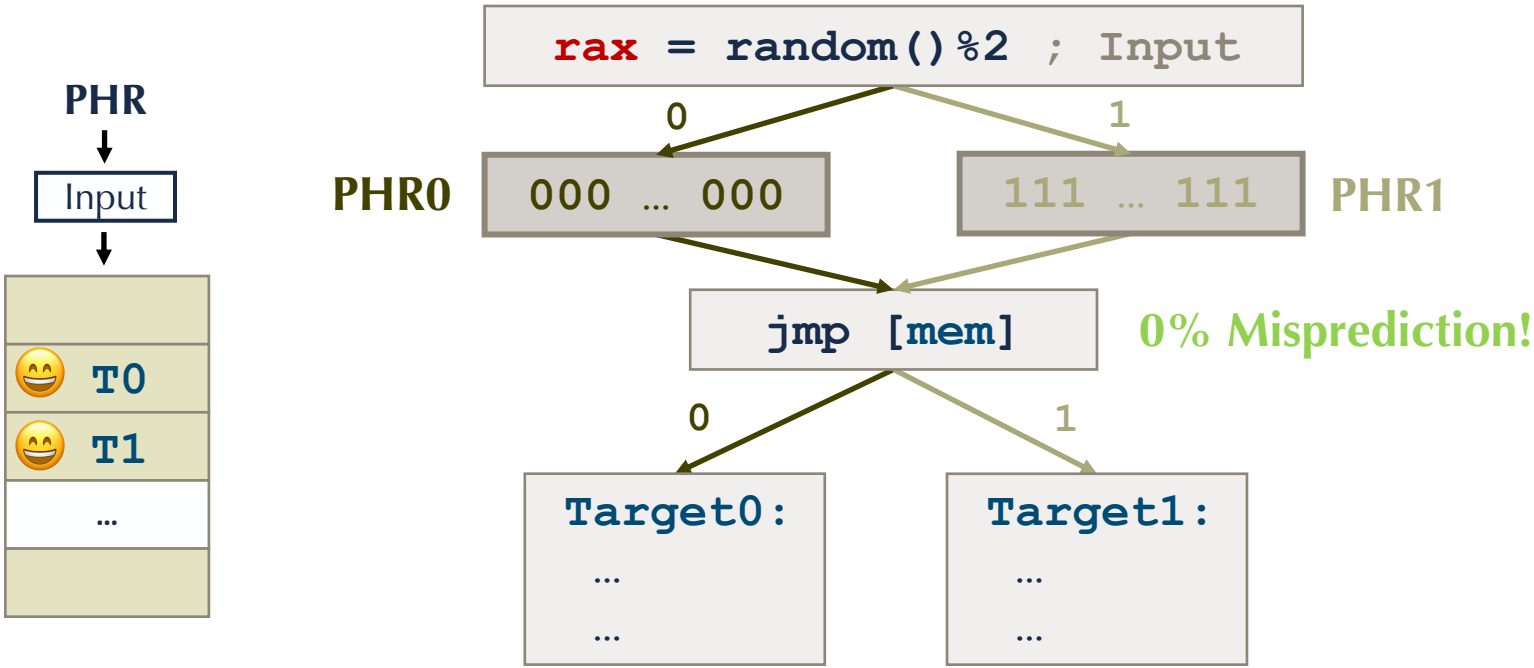


How Does PHR Help IBP Predict?

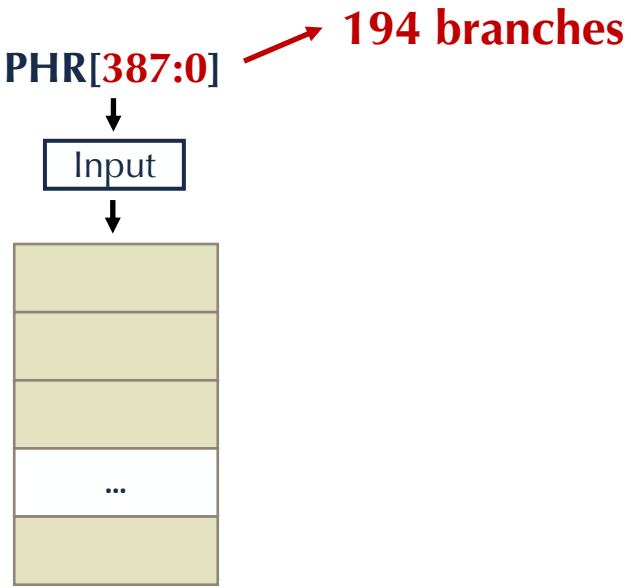




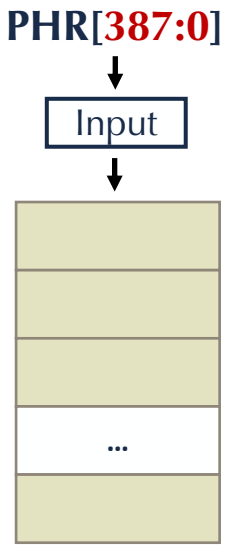
How Does PHR Help IBP Predict?

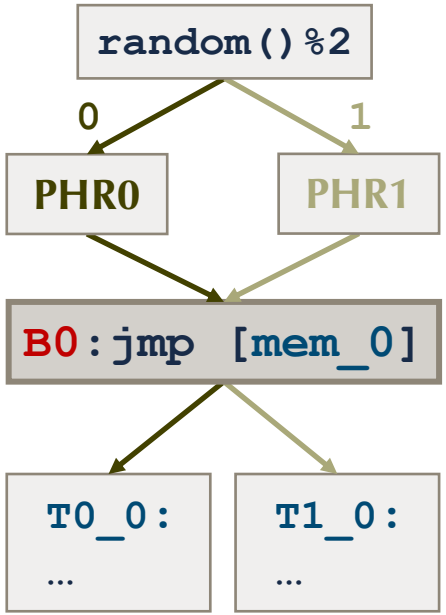


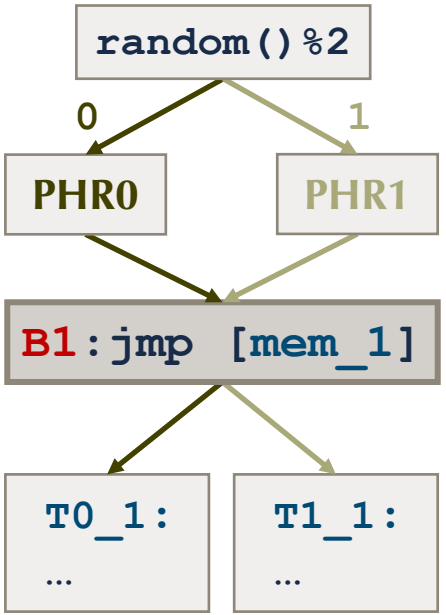
Reverse Engineering IBP - PHR Input



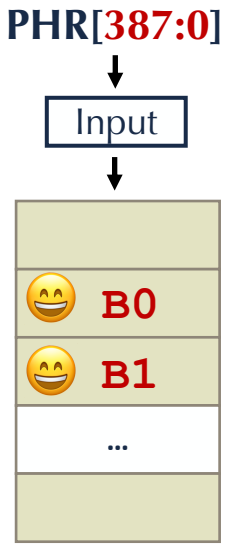
Reverse Engineering IBP - PC Input

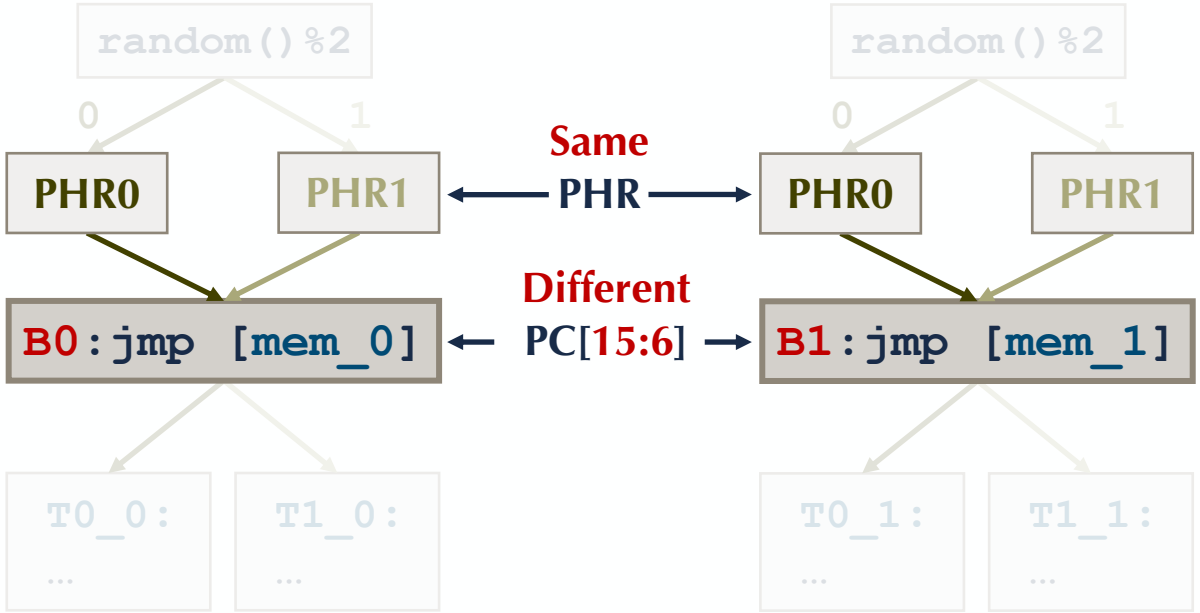




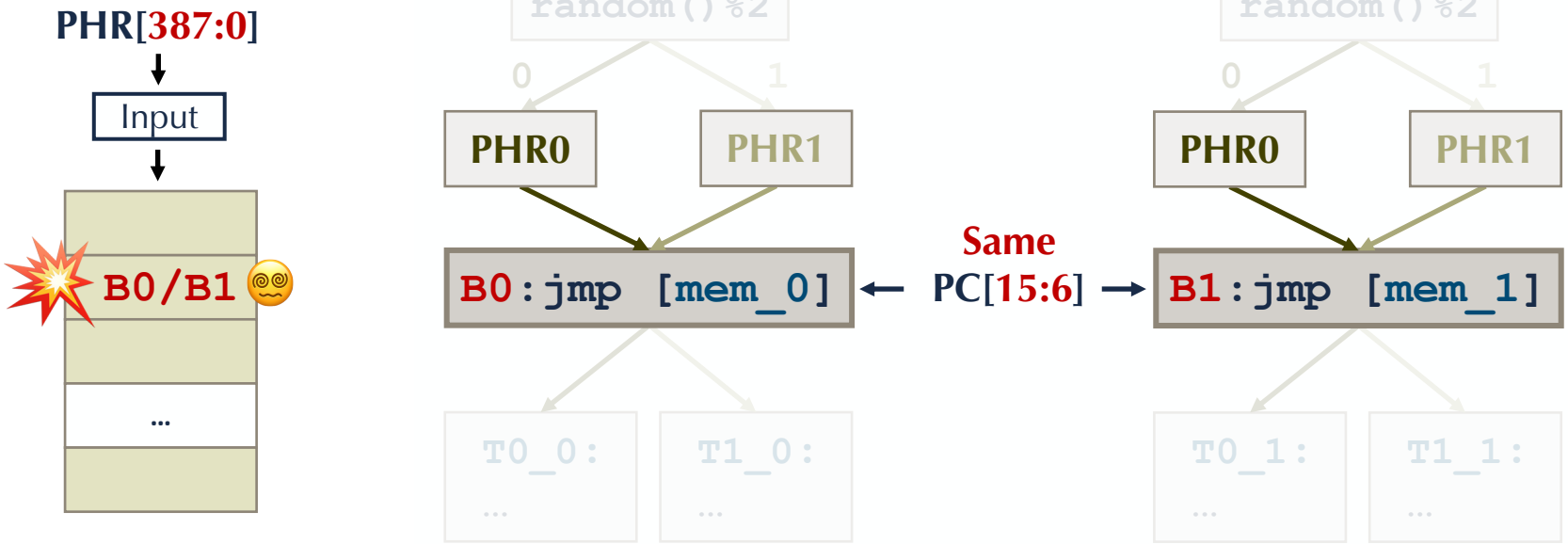


Reverse Engineering IBP - PC Input

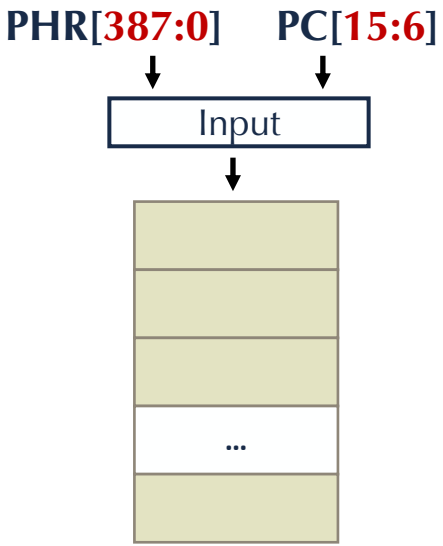




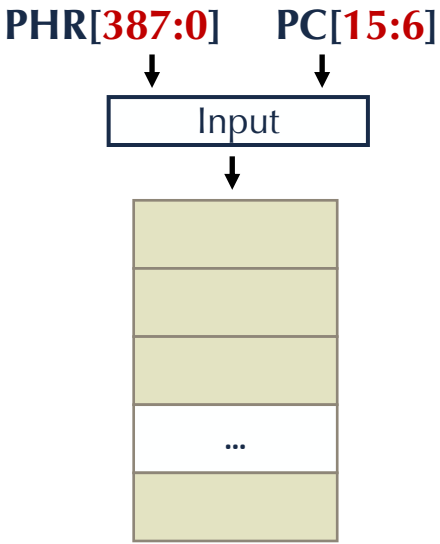
Reverse Engineering IBP - PC Input



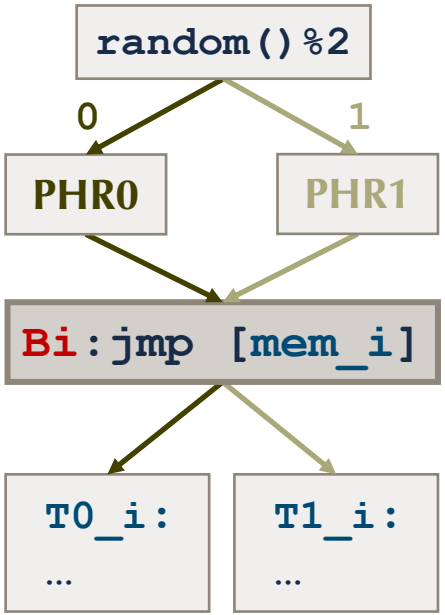
Reverse Engineering IBP - PC Input



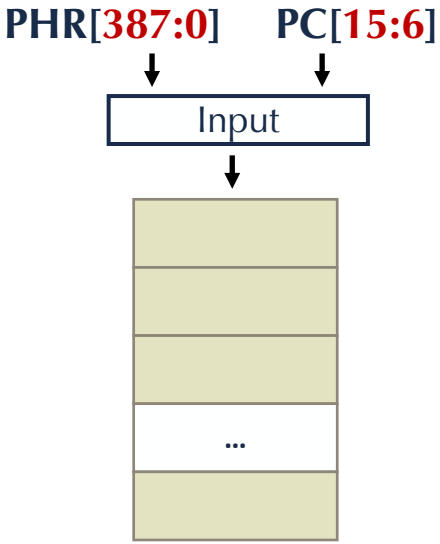
Reverse Engineering IBP - Associativity



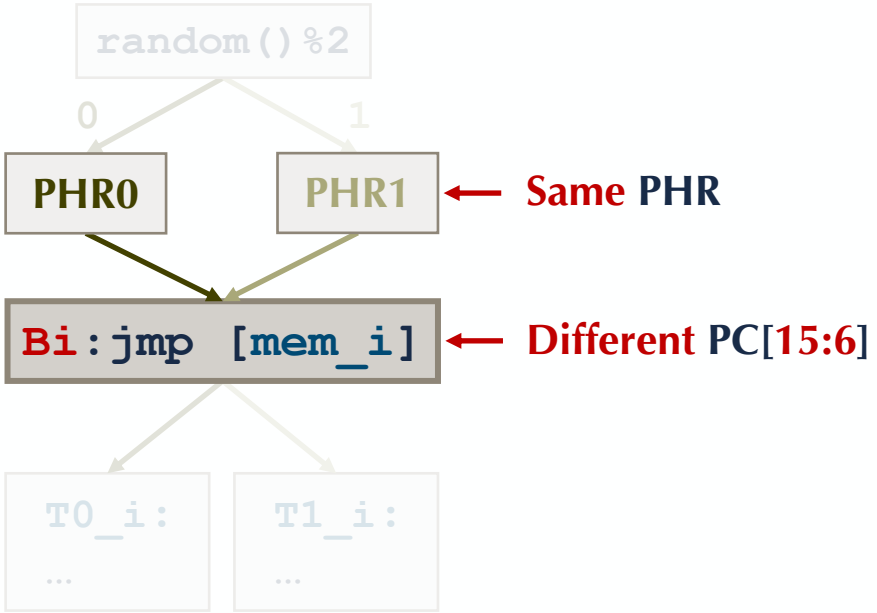
N Branches (`i = 1,2,…,N`)



Reverse Engineering IBP - Associativity

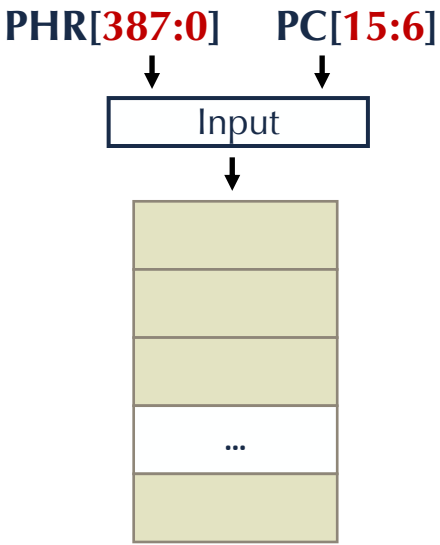


N Branches (`i = 1,2,…,N`)

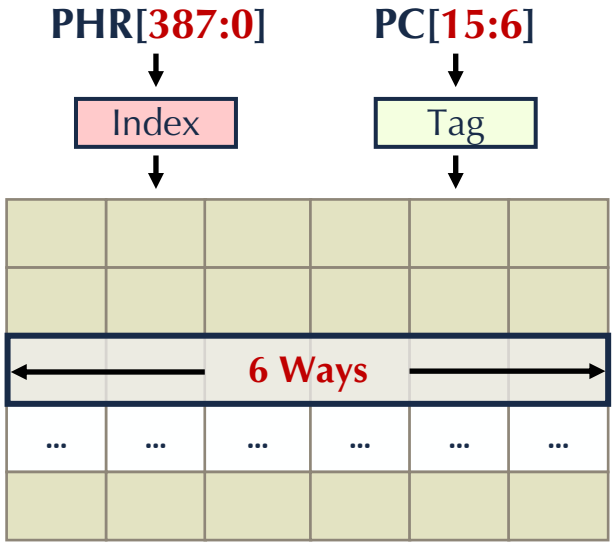


Reverse Engineering IBP - Associativity

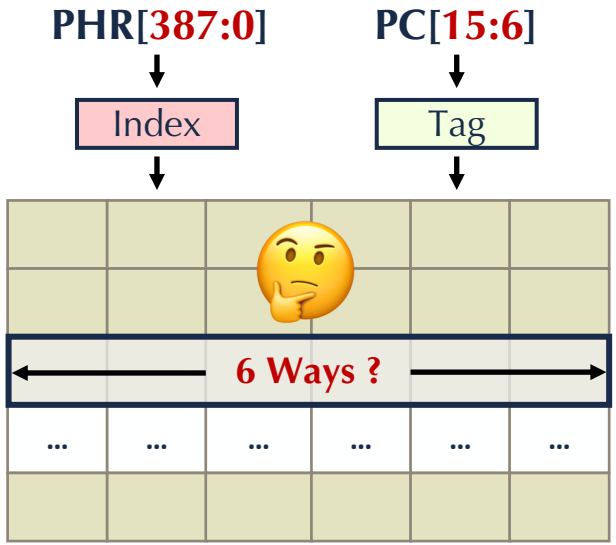
- Under the same PHR setup, misprediction rises when N > 6 !



Reverse Engineering IBP - Associativity

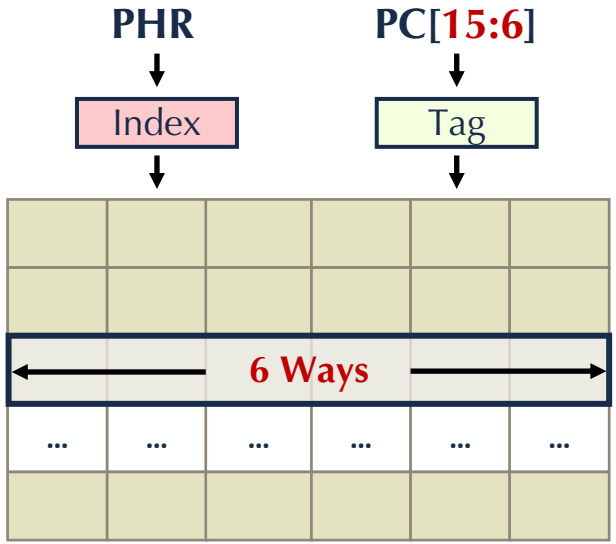


Reverse Engineering IBP - Associativity



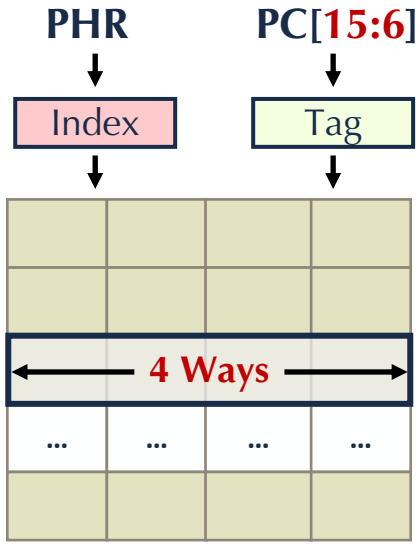
Reverse Engineering IBP - Associativity

- Under different PHR values, IBP **associativity varies**:



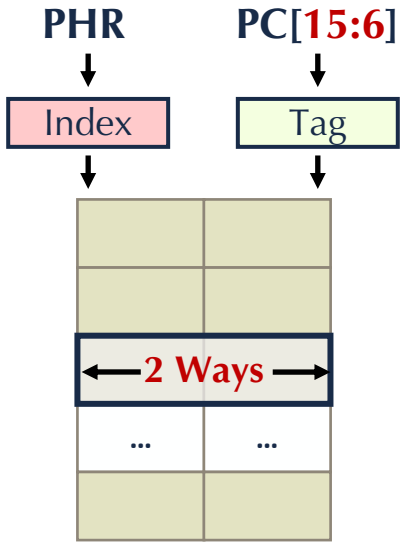
Reverse Engineering IBP - Associativity

- Under different PHR values, IBP **associativity varies**:

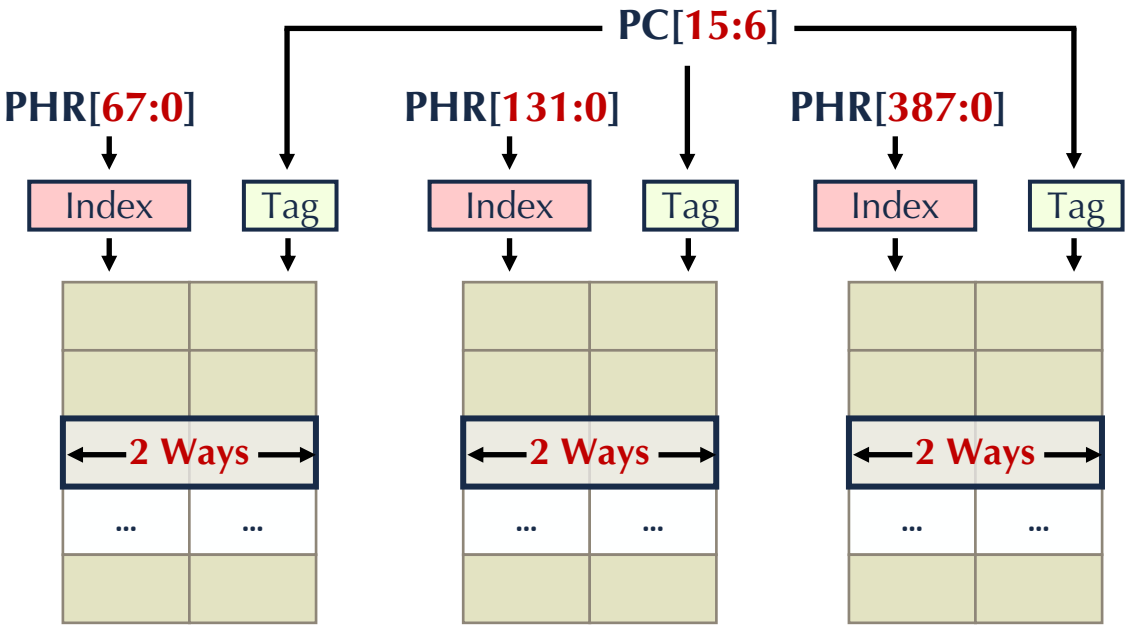


Reverse Engineering IBP - Associativity

- Under different PHR values, IBP **associativity varies**:

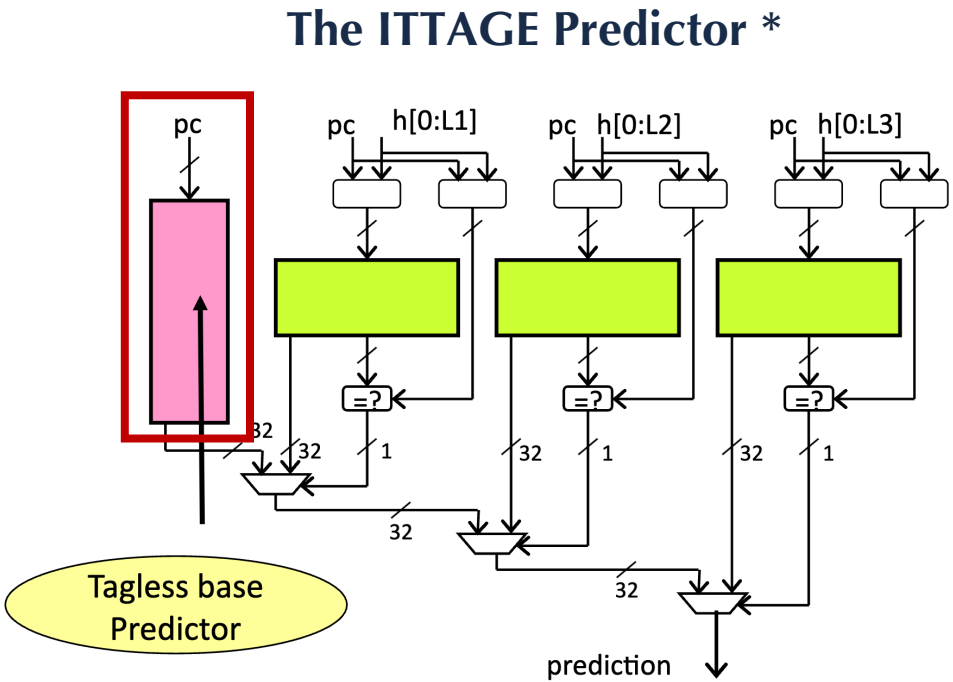


Reverse Engineering IBP - Associativity



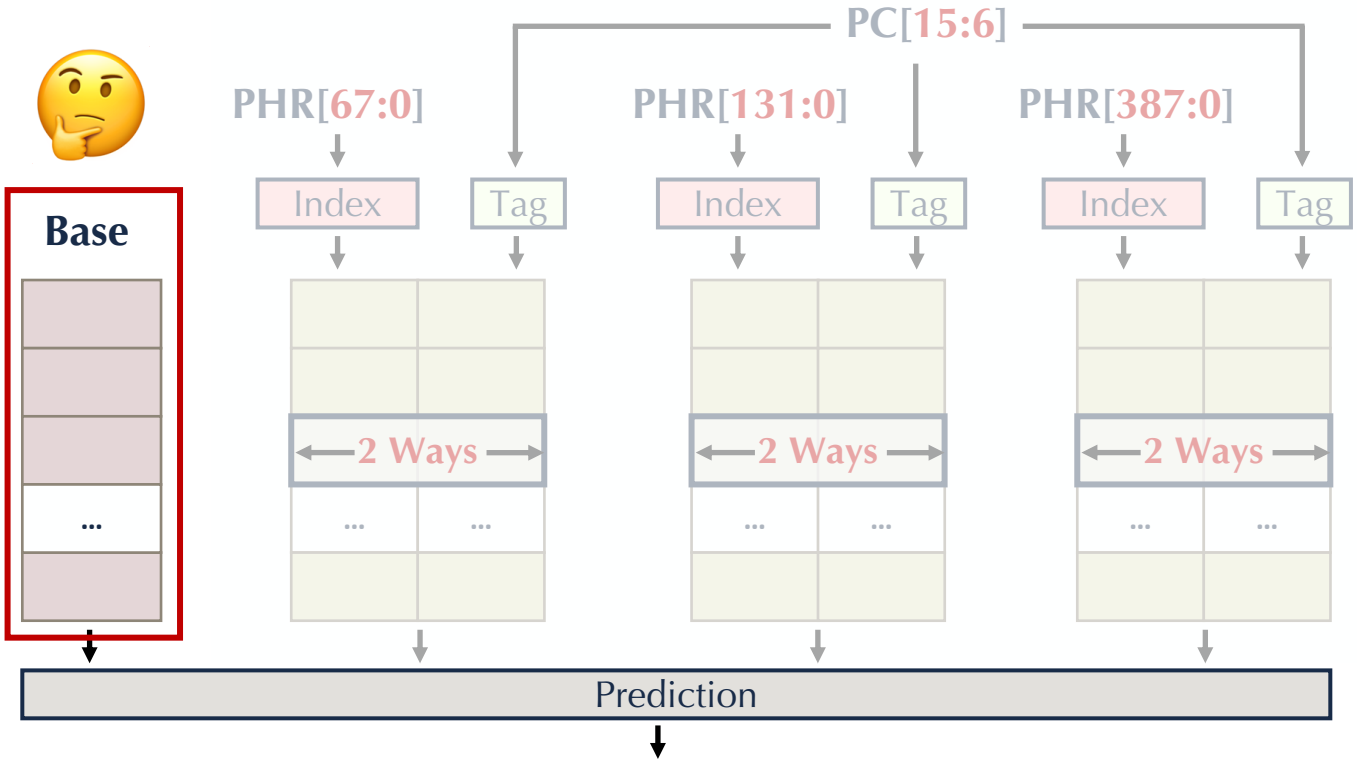
*The detailed associativity analysis is Section 3.2.2 of the paper.

This Reminds Us of …

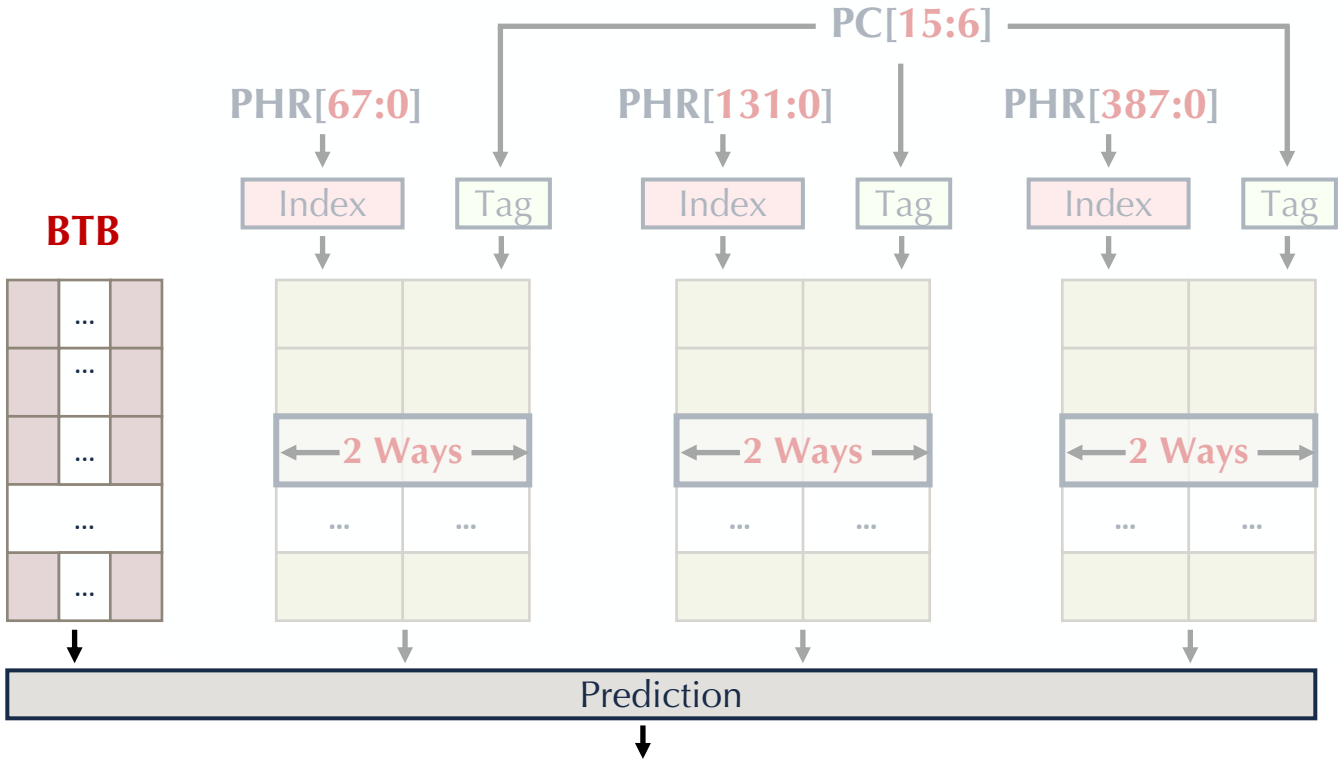


*Seznec, André. "A 64-Kbytes ITTAGE indirect branch predictor." In *JWAC-2: Championship Branch Prediction*. 2011.

What is the Base Predictor?



Reverse Engineering - BTB as Base BP

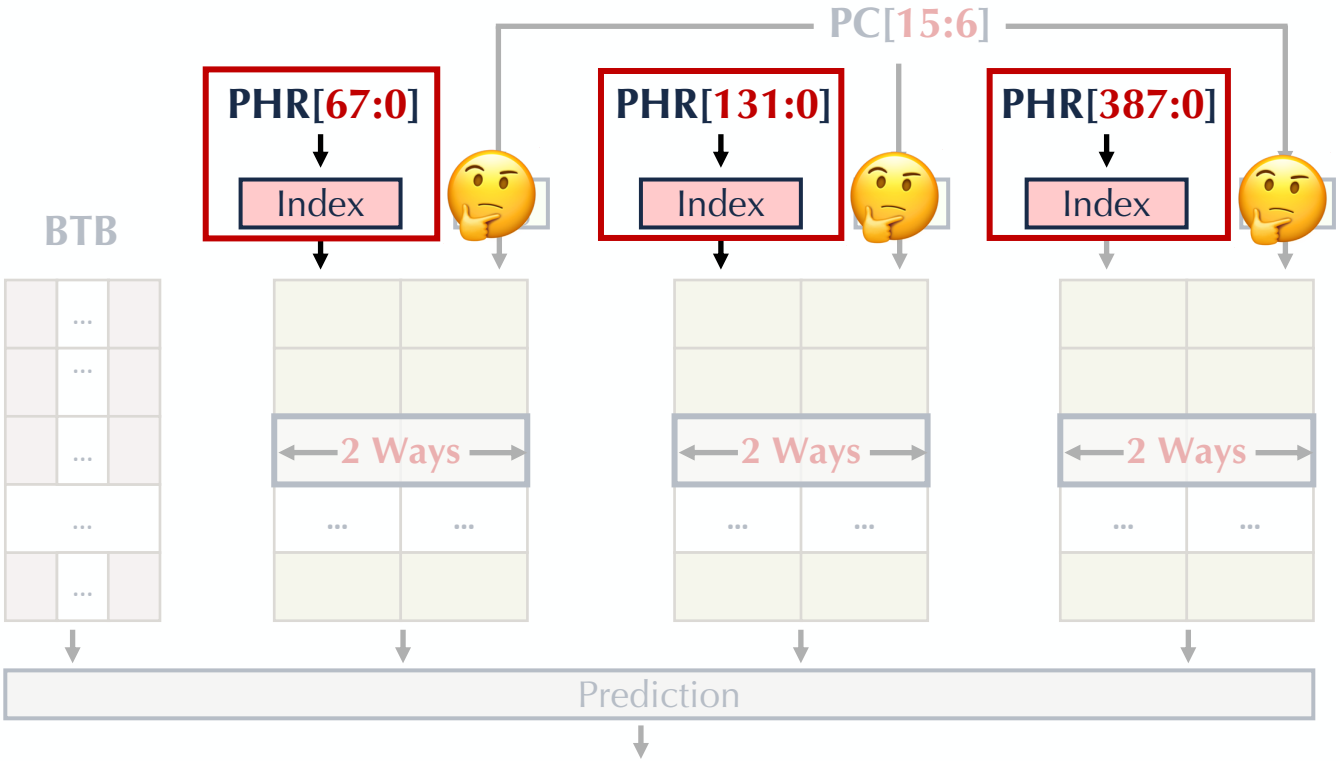


Reverse Engineering - BTB + IBP

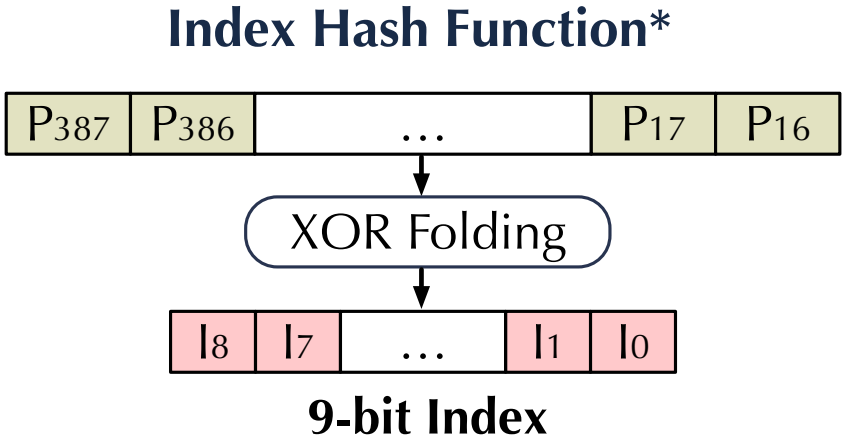
- **BPU predicts indirect branches only under BTB Hit**
  - No prediction under BTB Miss

- **Provide us opportunities to inject into both IBP and BTB**

How is IBP Index Calculated?



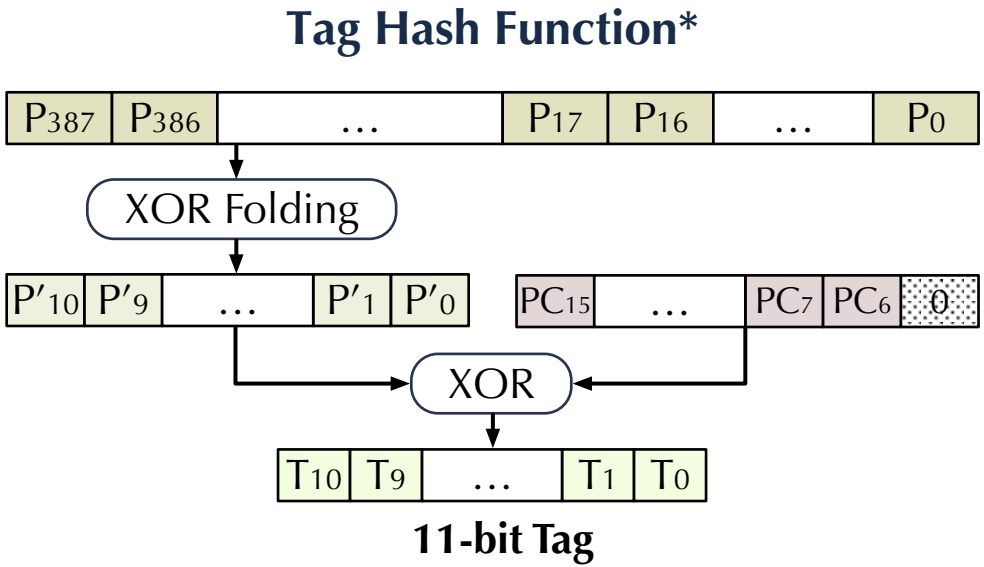
Reverse Engineering IBP - Index



*The detailed index hash function is shown in Figure 8 of the paper.

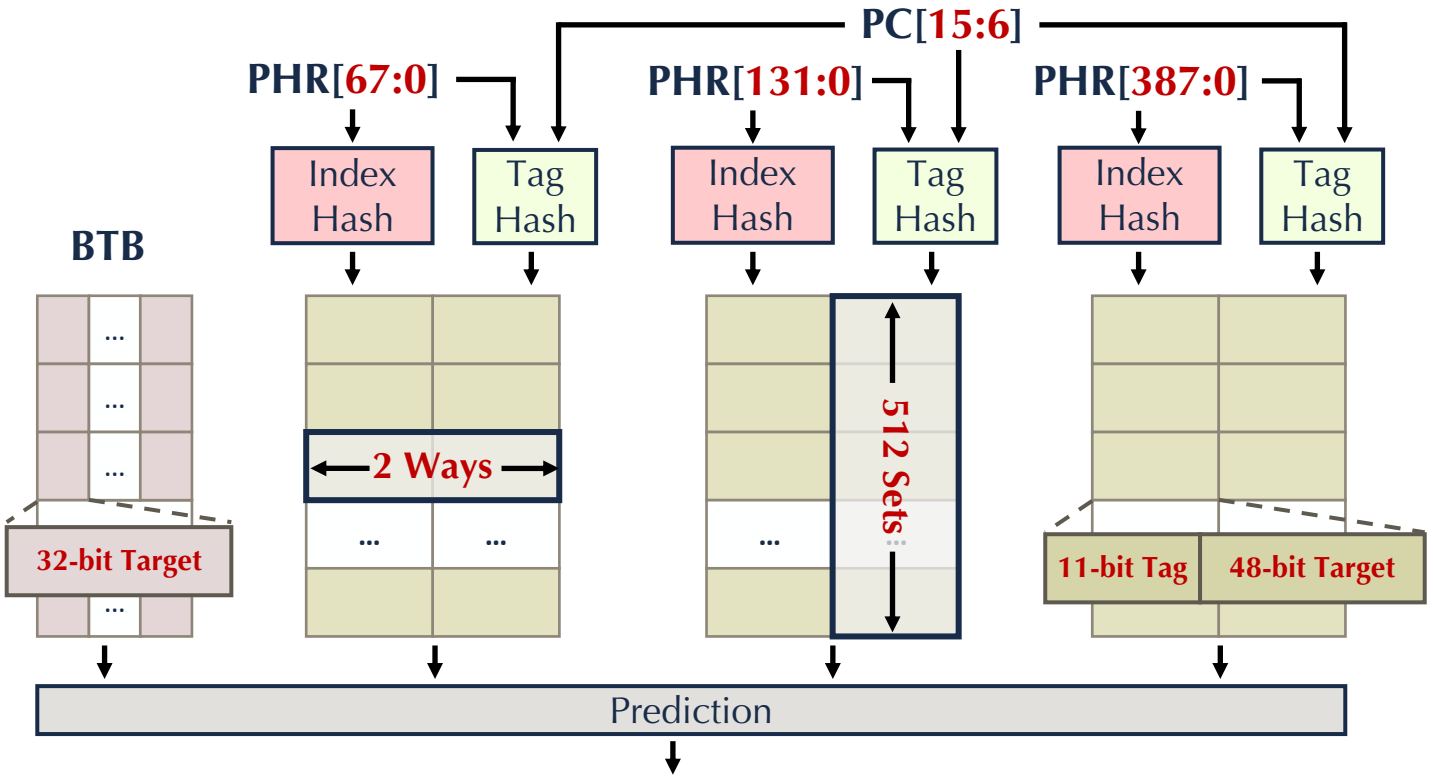
Reverse Engineering IBP - Tag

- **PHR is also used in calculating IBP tag, xored with PC:**



*The detailed tag hash function is shown in Figure 9 of the paper.

ITTAGE Predictor on Intel



Outline

- Background
- Motivation
- Indirect Branch Prediction
- **Intel BTI Defense Analysis**
  - IBRS
  - STIBP
  - IBPB
- High-Precision Injection Attack

Intel BTI Defenses

- **IBRS** - Indirect Branch Restricted Speculation
  - Prevent **Cross-Privilege** target injection
- **STIBP** - Single Thread Indirect Branch Predictors
  - Prevent **Cross-SMT Core** target injection

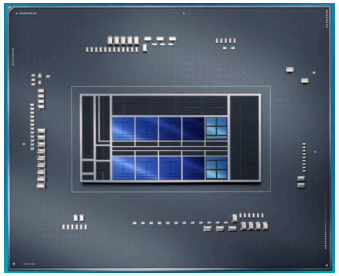
- **IBPB** - Indirect Branch Predictor Barrier
  - A **Fence**: prevent prior software from attacking indirect branches after it

How do these defenses impact BTB and IBP prediction?

Timeline

Skylake

BTI (Spectre v2)

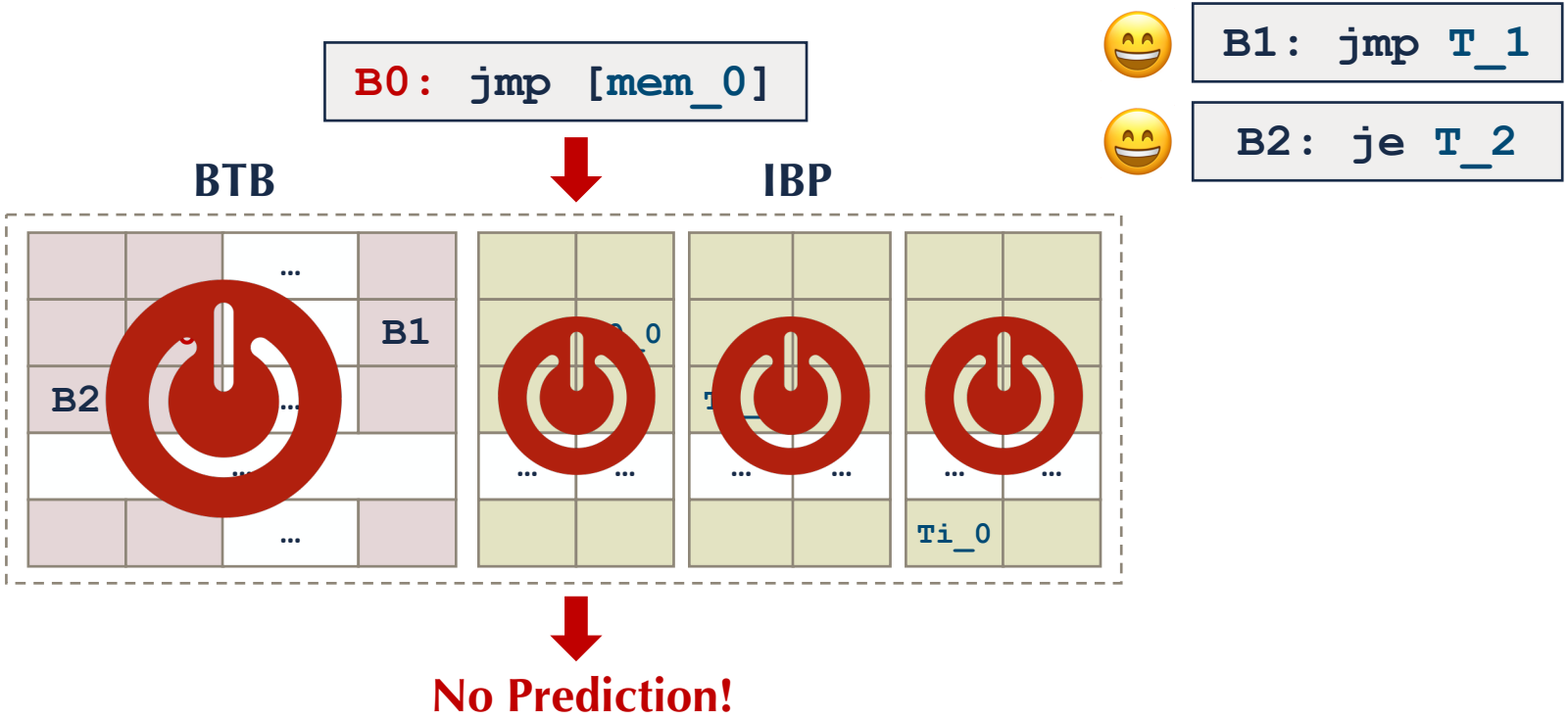


Golden Cove Raptor Cove

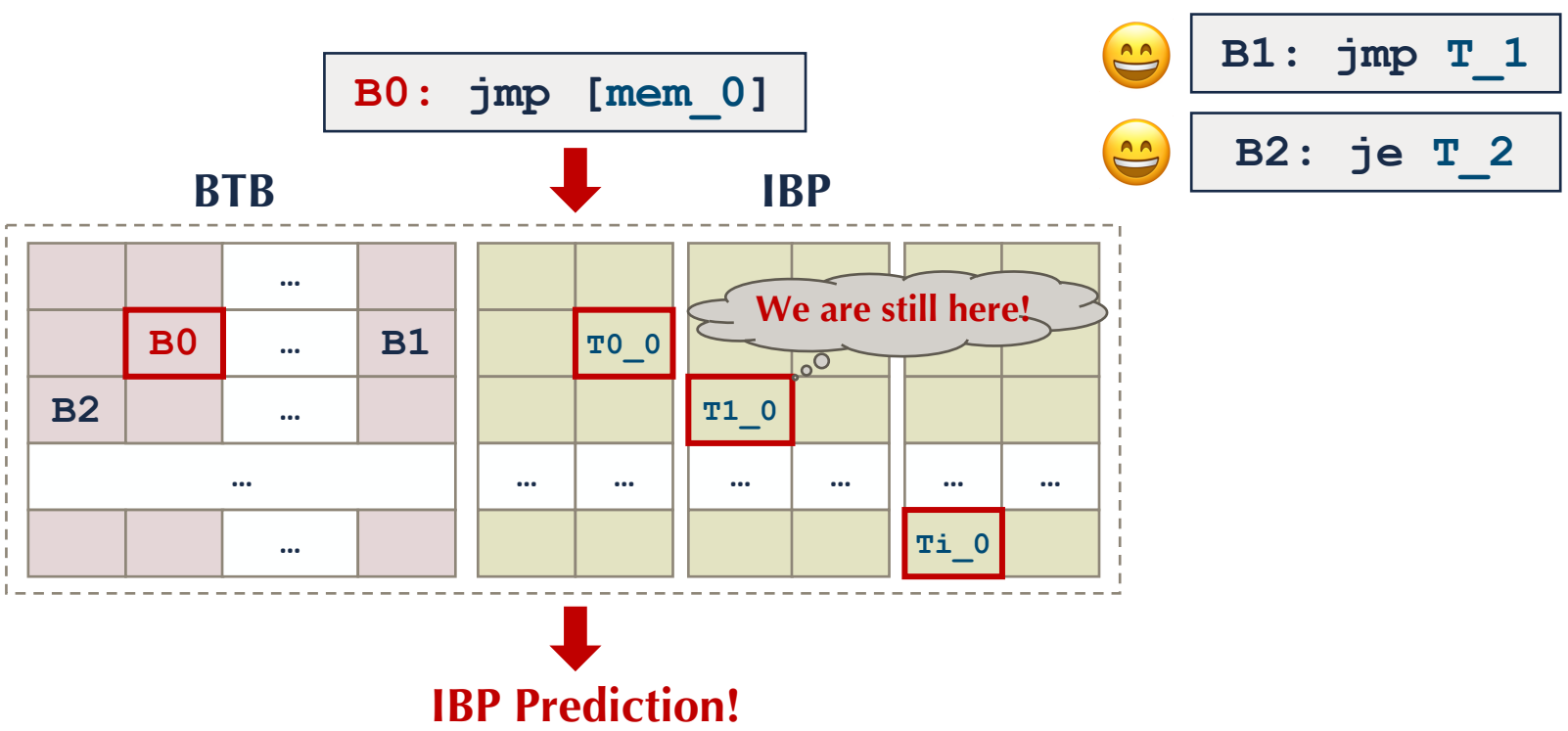
2015 2017 2021 2022

How does Intel defend without hardware modifications?

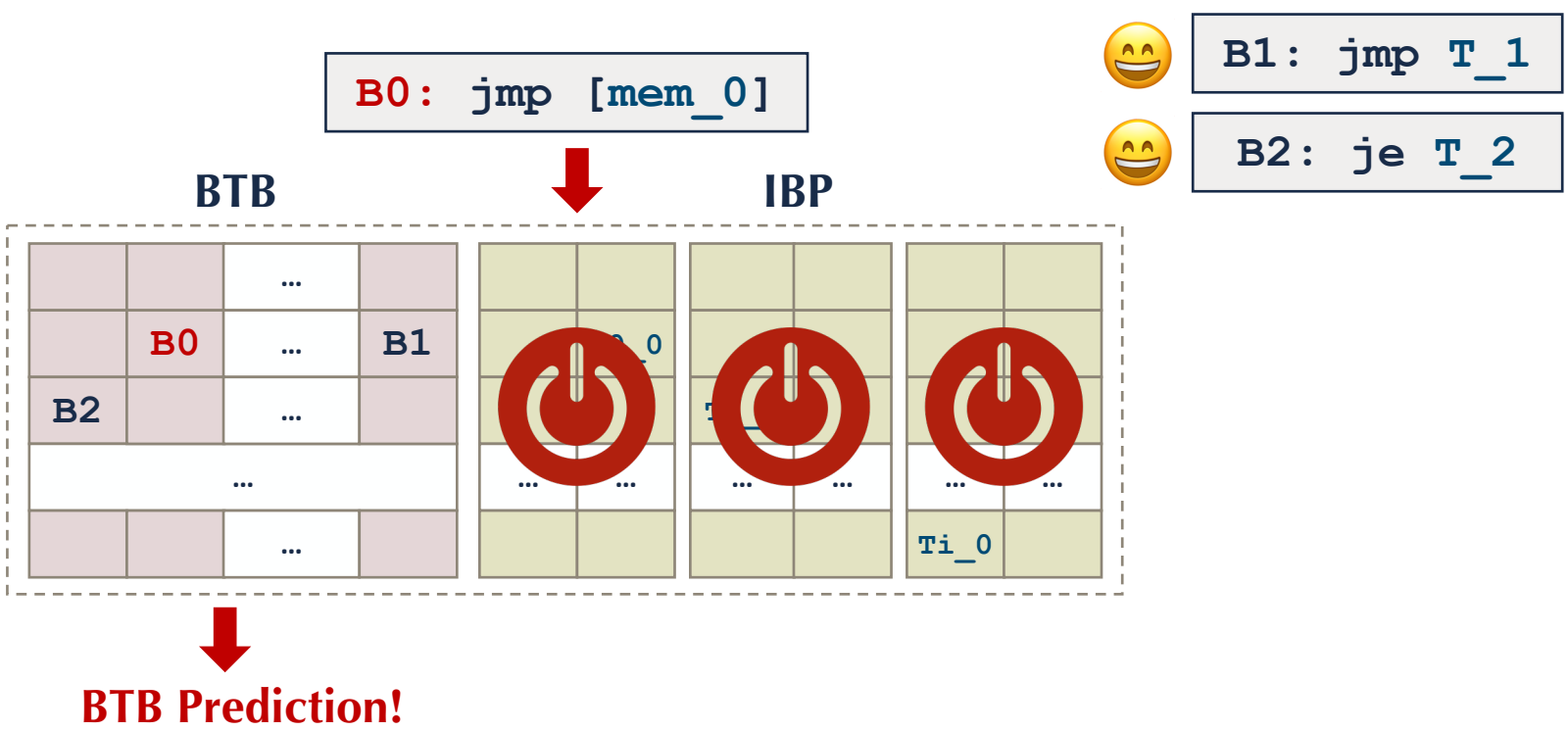
Pre-Spectre - IBRS



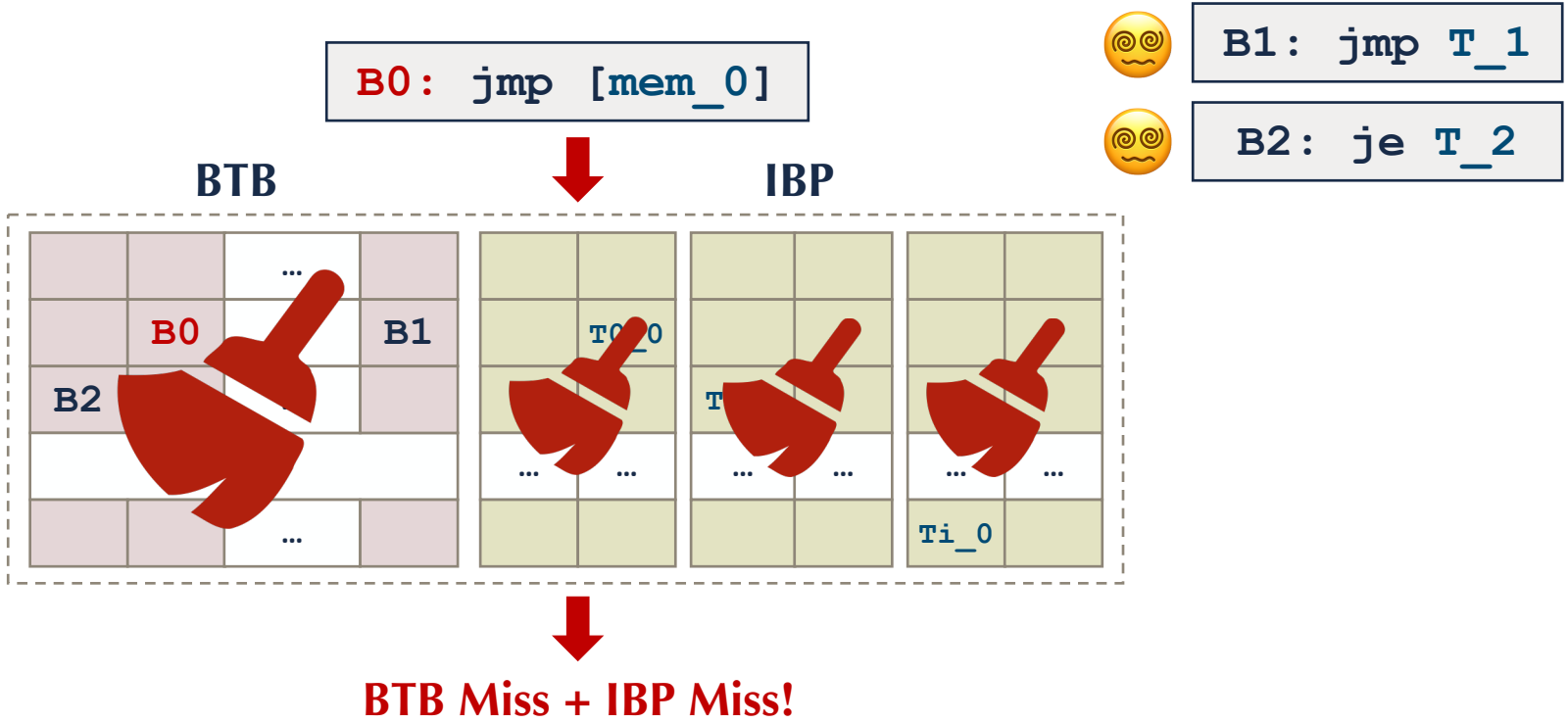
Pre-Spectre - After Resetting



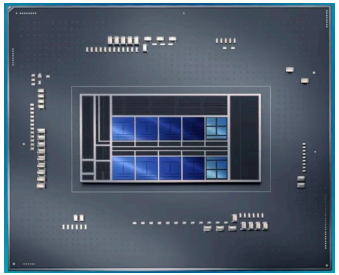
Pre-Spectre - STIBP



Pre-Spectre - IBPB



Timeline

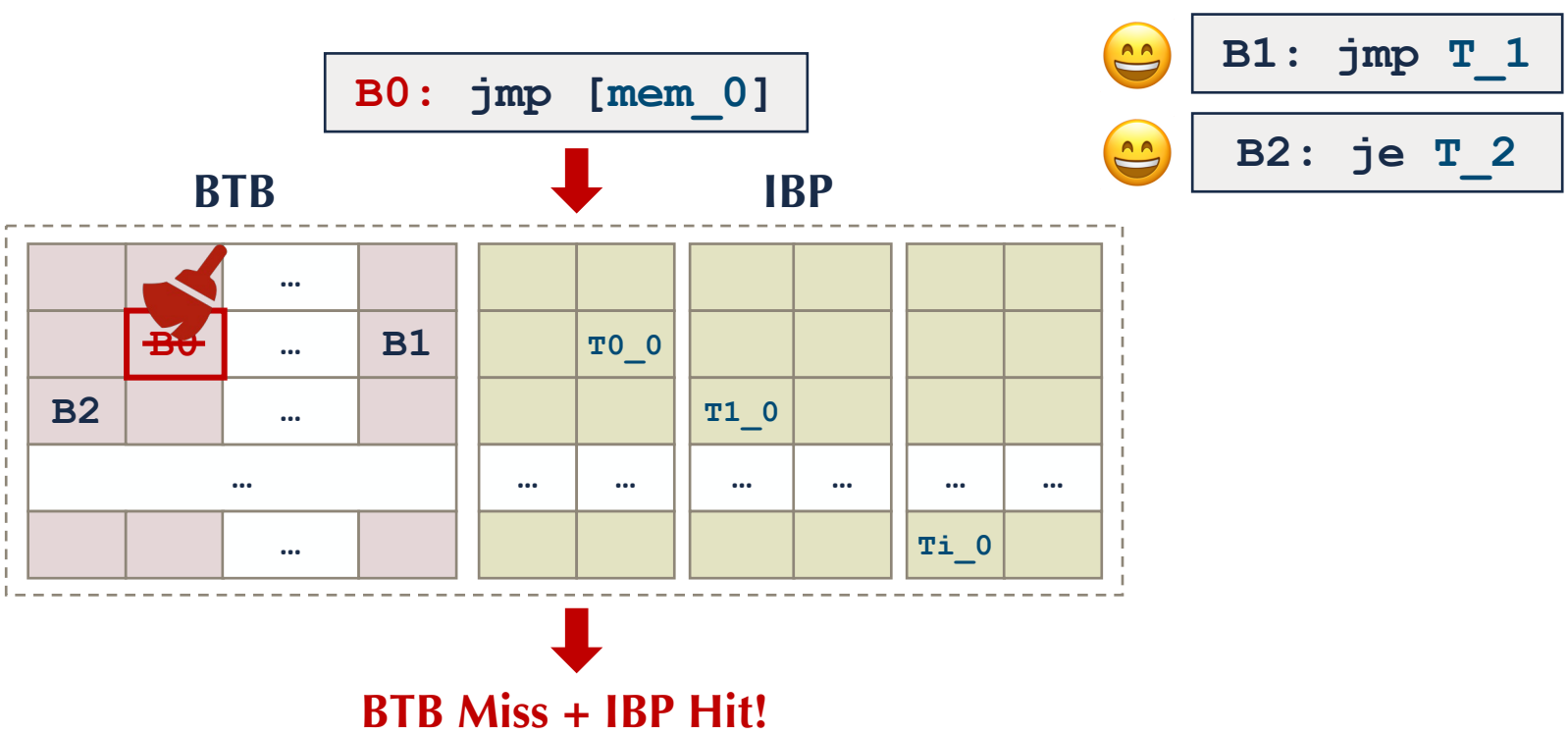


Skylake BTI (Spectre v2) Golden Cove Raptor Cove

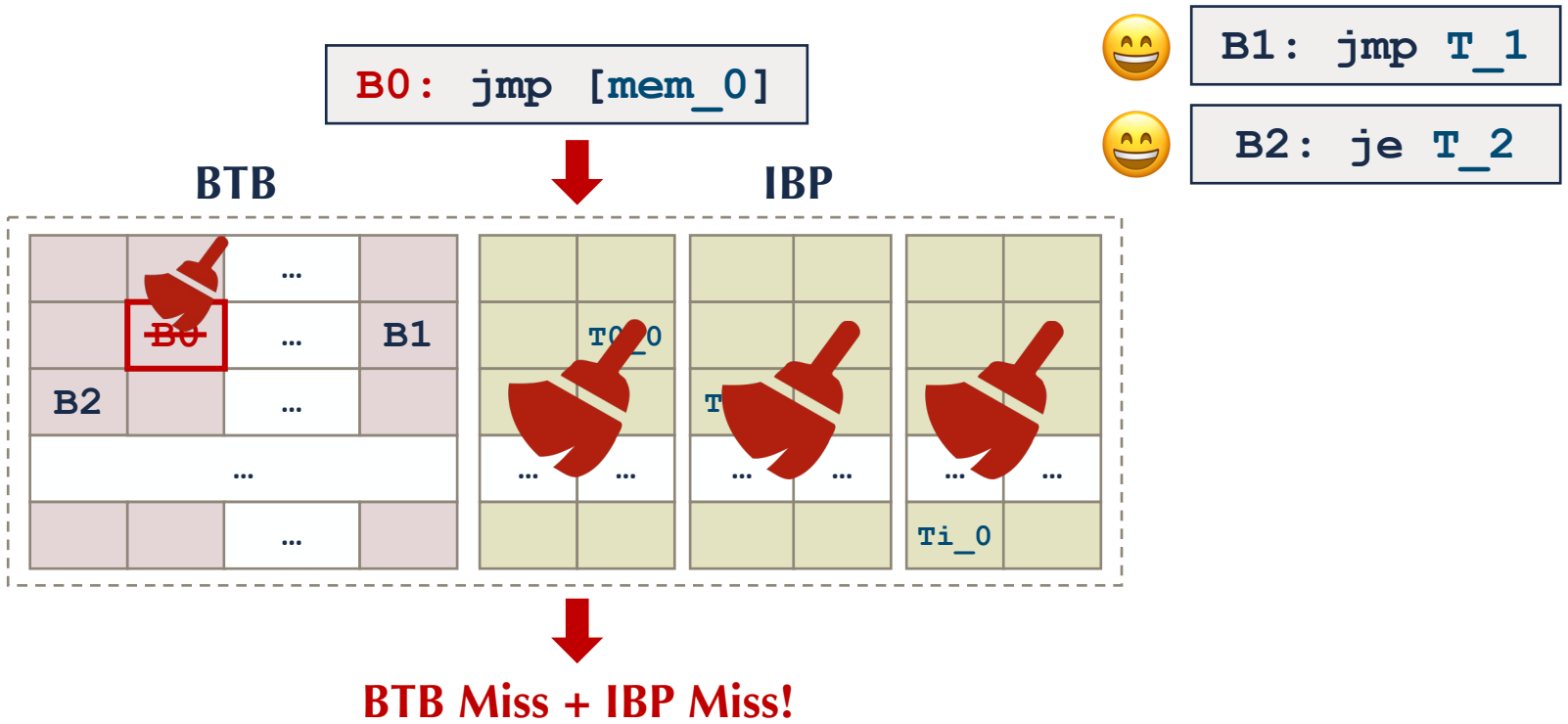
2015 2017 2021 2022

🤔 **Has Intel implemented new hardware-level defenses?**

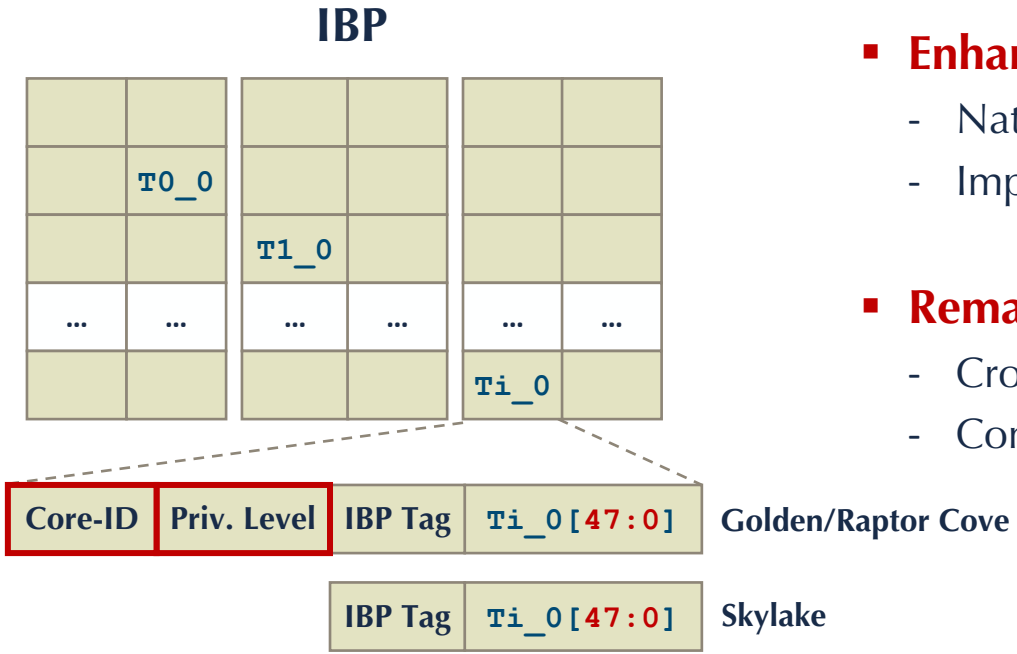
Post-Spectre - IBRS & STIBP



Post-Spectre - IBPB



Reverse Engineering - New Metadata



- **Enhanced IBRS and STIBP:**
  - Natural isolation
  - Improved overhead
- **Remaining attack surfaces:**
  - Cross-thread, Cross-process
  - Contention-based attack

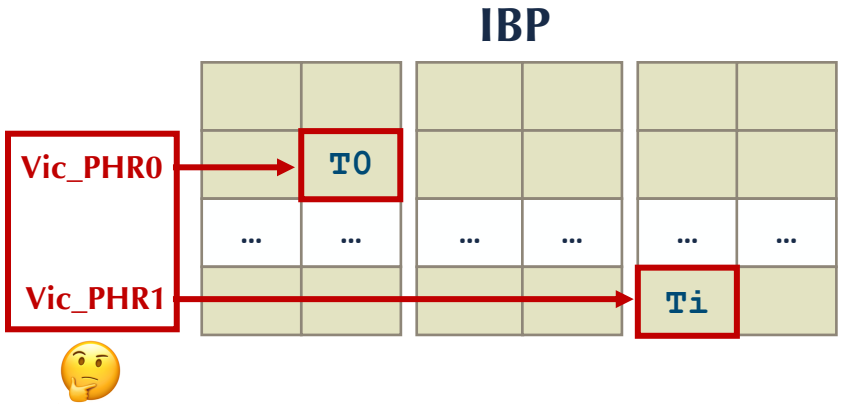
Outline

- Background
- Motivation
- Indirect Branch Prediction
- Intel BTI Defense Analysis
- **High-Precision Injection Attack**
  - iBranch Locator
  - IBP&BTB Injection Attack
  - Break ASLR inside BTB

Victim Indirect Branch

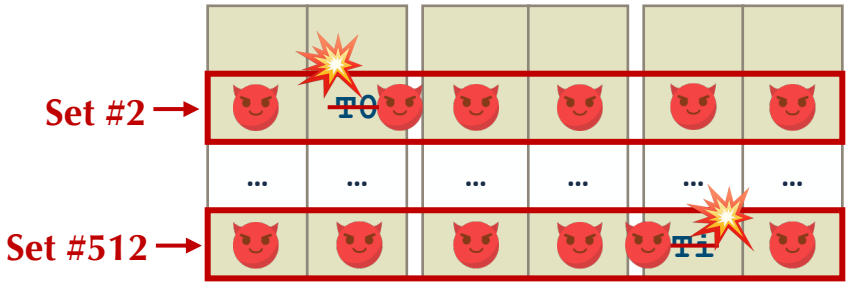

```
jmp [mem]
  …
T0:…
Ti:…
----------------------
LEAK_GADGET:
 mov rax, [secret]
 mov rbx, [2048+rax*128]
```

Victim Process



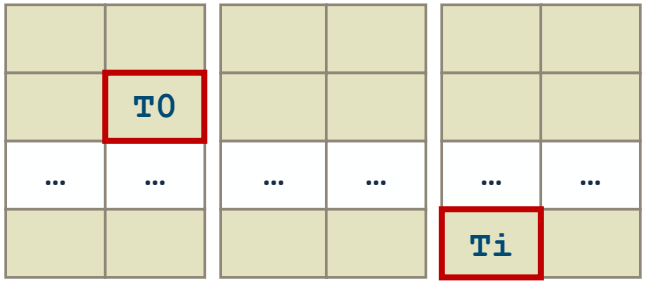
How to Find IBP Aliasing?

- **iBranch Locator - Search IBP aliasing precisely and efficiently**
  - **Step1:** To locate victim IBP **sets**
    - Use a **contention-based** technique to scan IBP



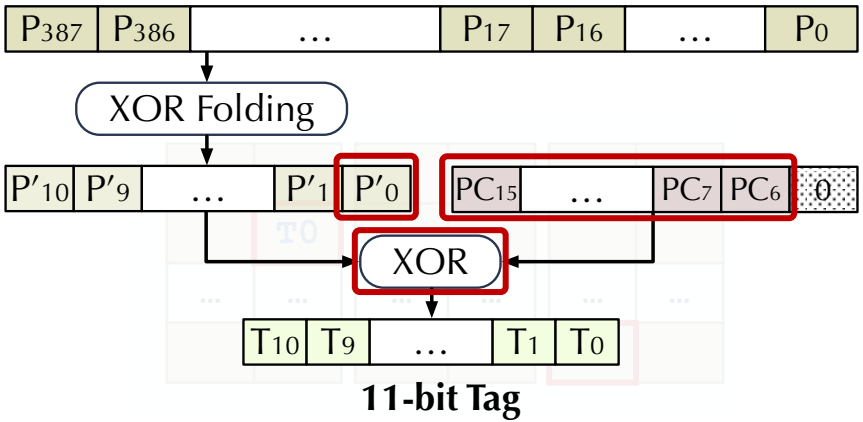
How to Find IBP Aliasing?

- **iBranch Locator - Search IBP aliasing precisely and efficiently**
  - **Step2:** To locate victim IBP **entries** (tag aliasing)



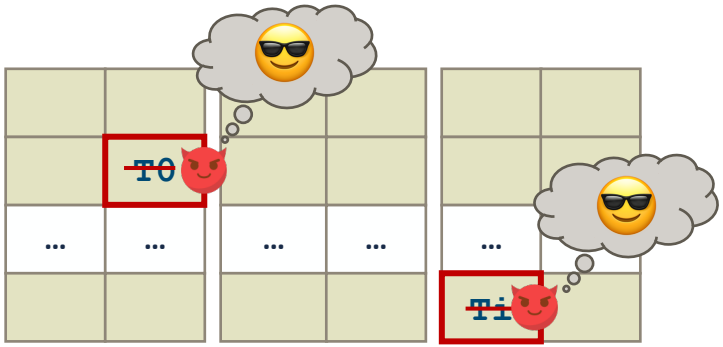
How to Find IBP Aliasing?

- **iBranch Locator - Search IBP aliasing precisely and efficiently**
  - **Step2:** To locate victim IBP **entries** (tag aliasing)



How to Find IBP Aliasing?

- **iBranch Locator - Search IBP aliasing precisely and efficiently**
  - **Step2:** To locate victim IBP **entries** (tag aliasing)
    - **Brute-force 10 PC bits and 1 PHR bit** to get all the tag permutations

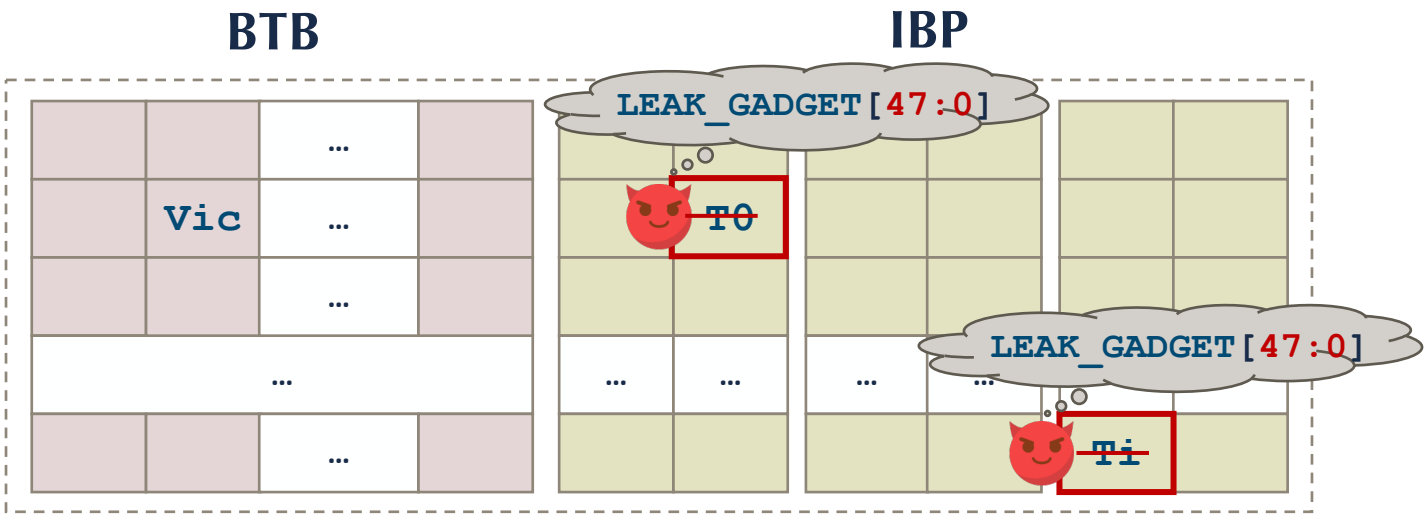


iBranch Locator

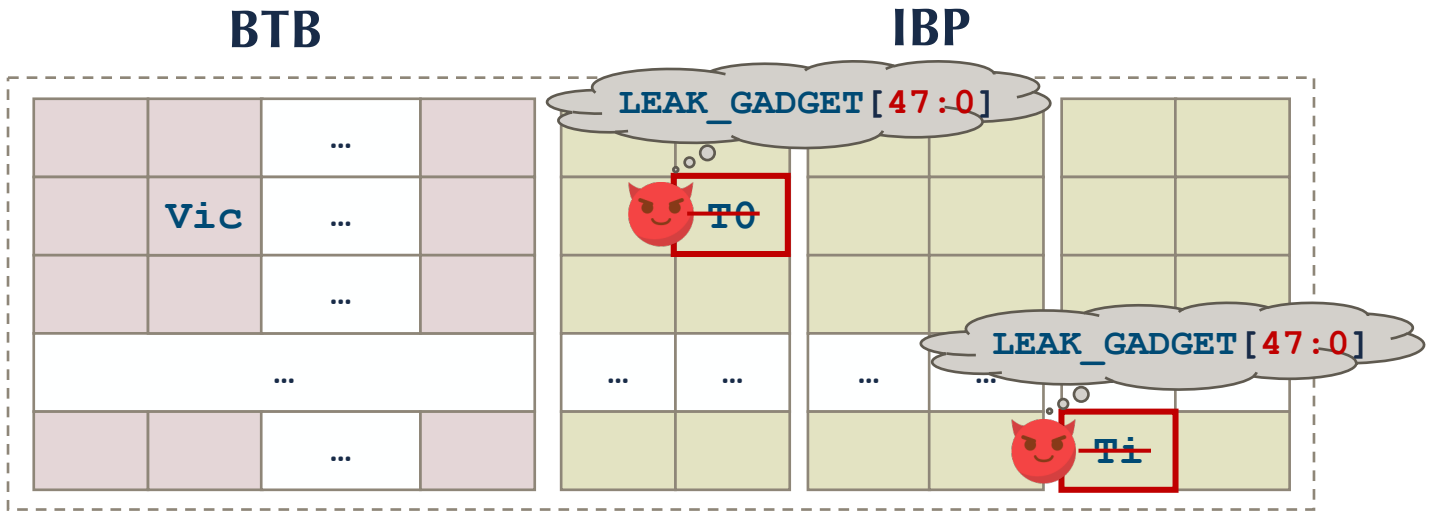
- **At most 512 (Index) + 2048 (Tag) attempts to find IBP aliasing**
  - **Index Locator:** cost ~ 2 s, success rate ~ 97%
  - **Tag Locator:** cost ~ 2.5 s, success rate 100%

- **Precisely search the exact IBP structure rather than the entire PHR**
  - Bypass the need for prior knowledge of victim branch history (vs. BTI)
  - Efficient and stable total search time (vs. BHI)

IBP Injection Attack

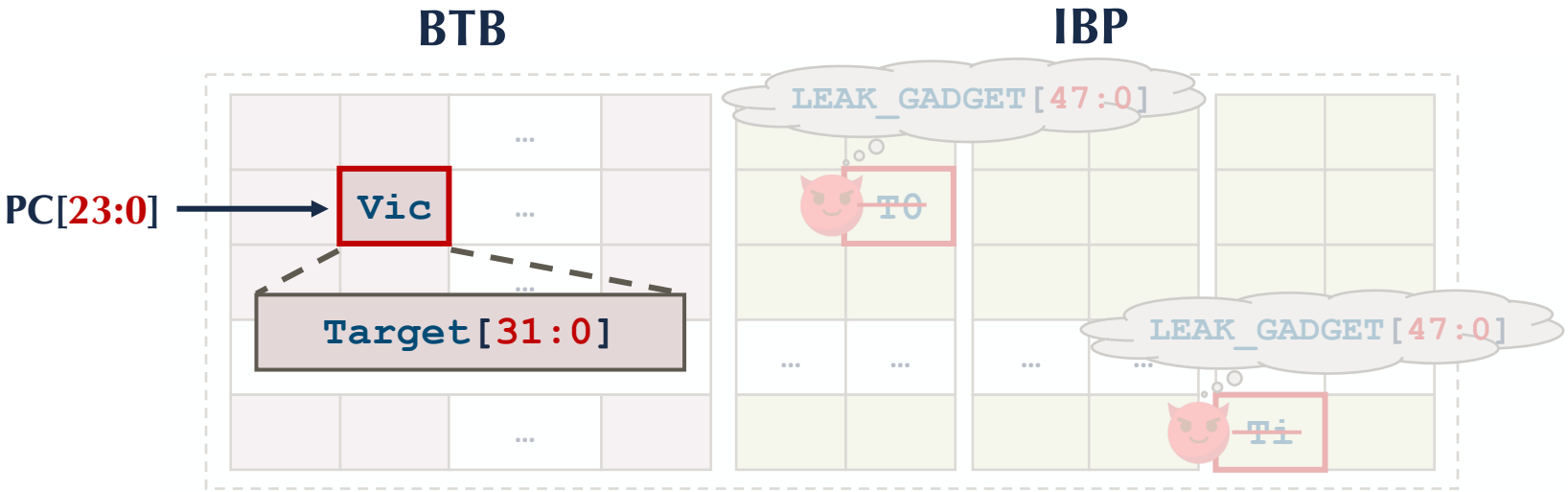


Injection Attack under full ASLR



Finding fully aliased virtual target is challenging under full ASLR!

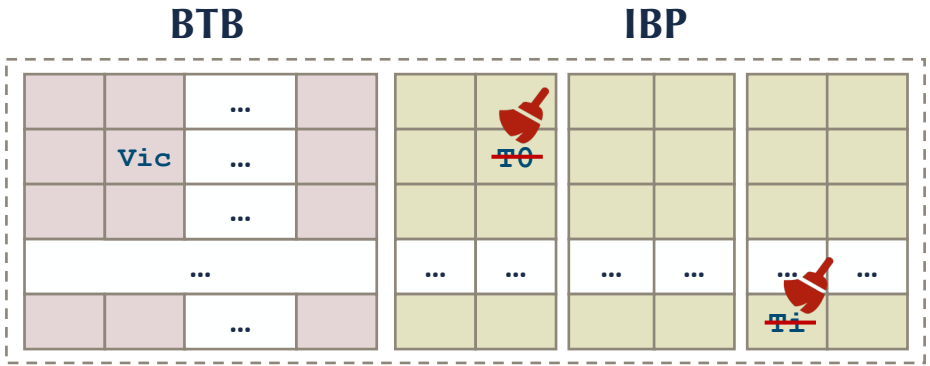
Injection Attack under full ASLR



Can we **inject into BTB** and mislead victim by **BTB prediction**?

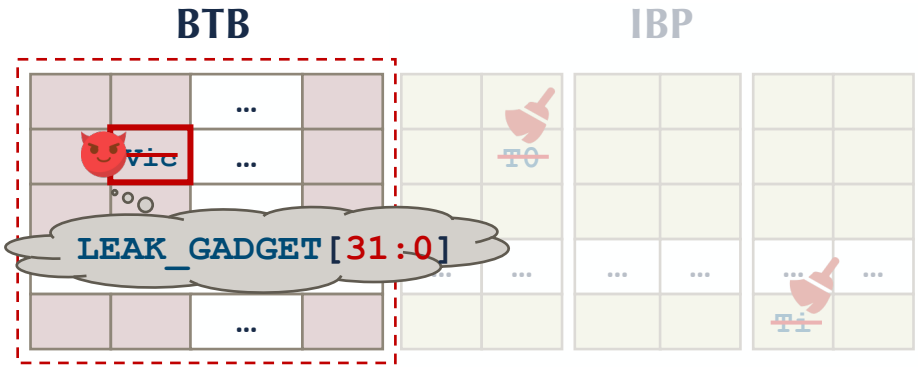
New Injection Surface - BTB

- **Step 1:** Use **Index** Locator to **EVICT** victim from **IBP**



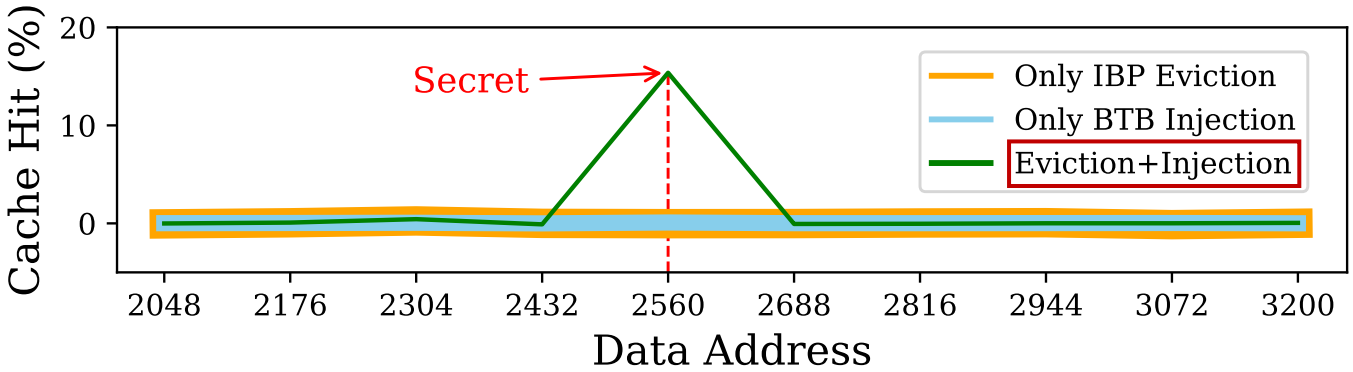
New Injection Surface - BTB

- **Step 1:** Use **Index** Locator to **EVICT** victim from **IBP**
  - Indirect branch prediction is **biased towards BTB** (BTB Hit + IBP Miss)
- **Step 2:** Use aliased **32-bit** target to **INJECT** the victim **BTB** entry
- **Step 3:** Use cache side channel (e.g., Flush+Reload*) to **EXTRACT** the secret



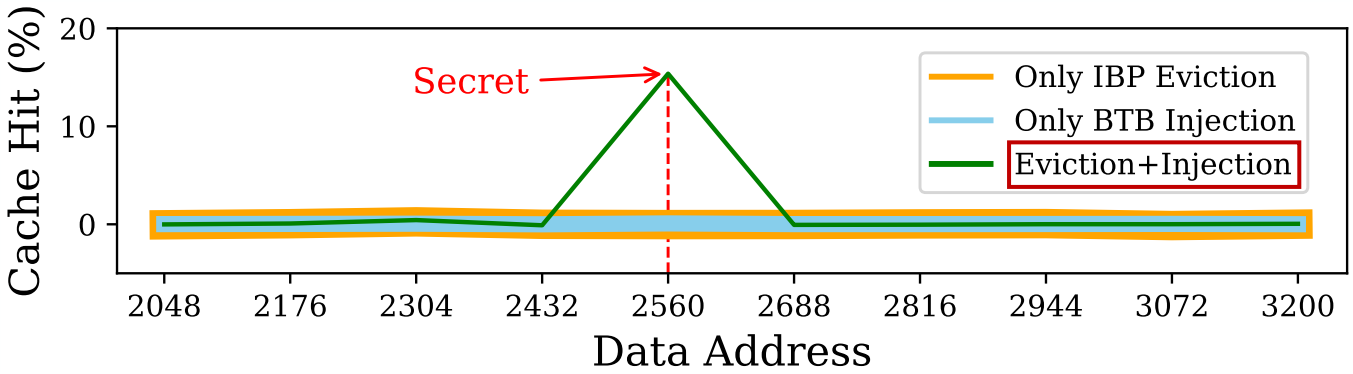
* Yarom, Yuval, and Katrina Falkner. "{FLUSH+ RELOAD}: A high resolution, low noise, l3 cache {Side-Channel} attack." In 23rd USENIX security symposium (USENIX security 14).

New Injection Surface - BTB



- **Efficient to Find Aliasing:** Only require Index Locator
- **Practical under IBRS, STIBP, IBPB:** Pre-inject after flush
- **Practical under Full ASLR:** Easier to find 32-bit aliased target

New Injection Surface - BTB

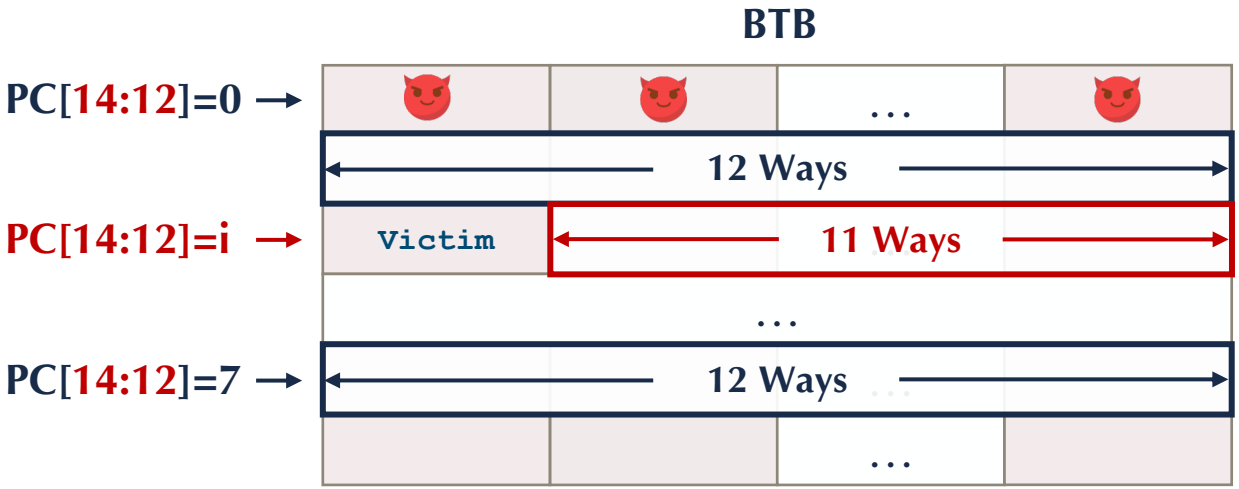


- Efficient to Find Aliasing: Only require Index Locator
- Practical under IBRS, STIBP, IBPB: Pre-inject after flush
- **Practical under Full ASLR: Easier to find 32-bit aliased target**

🤔 **Do we really need to exhaustively search `LEAK_GADGET`[31:12]?**

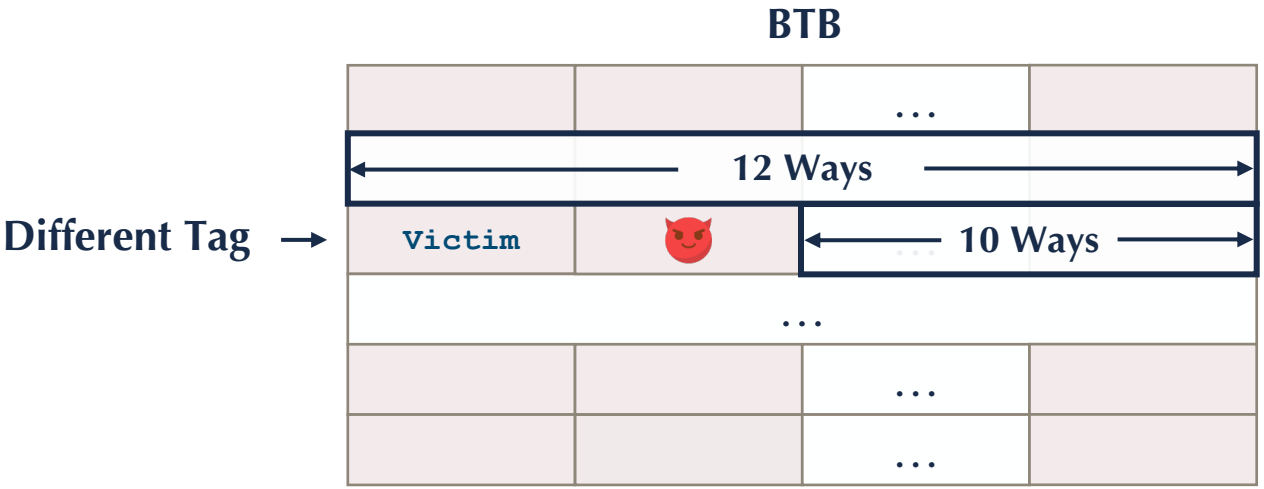
Breaking ASLR inside BTB

- **Step 1:** Break victim **PC[14:12]** (BTB index **PC[14:5]**) by **scanning 8 BTB sets**



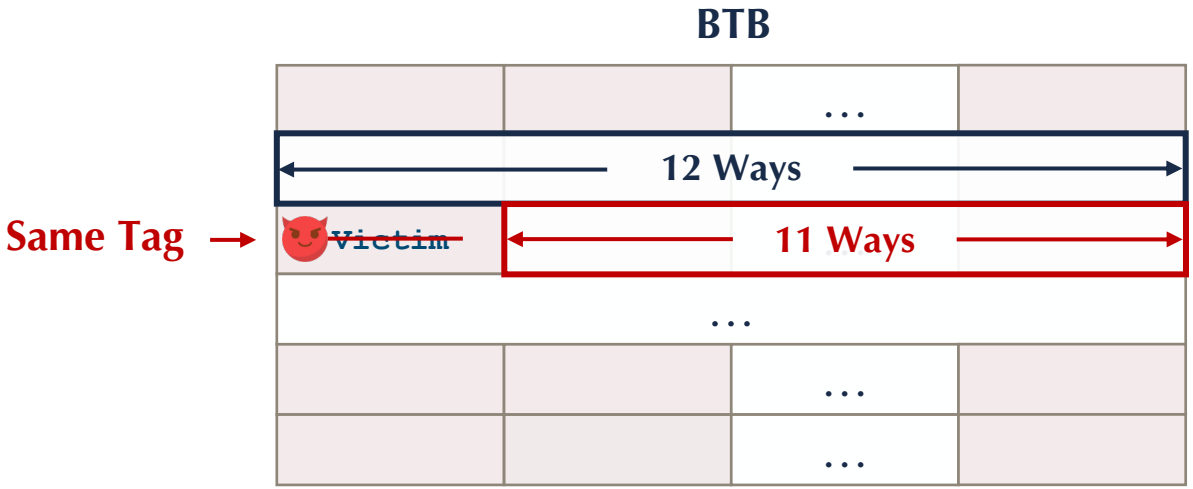
Breaking ASLR inside BTB

- **Step 2:** Break victim **PC[23:15]** (BTB Tag) by **detecting entry sharing**



Breaking ASLR inside BTB

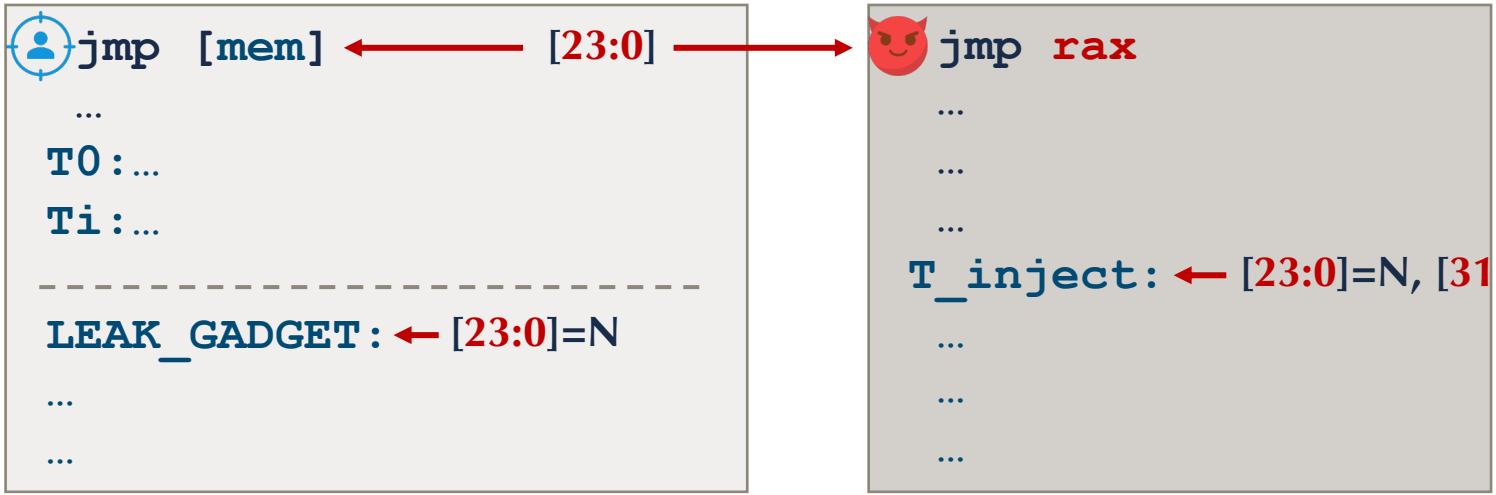
- **Step 2:** Break victim **PC[23:15]** (BTB Tag) by **detecting entry sharing**



- **Successfully locate victim BTB entry and extract `LEAK_GADGET`[23:12]**

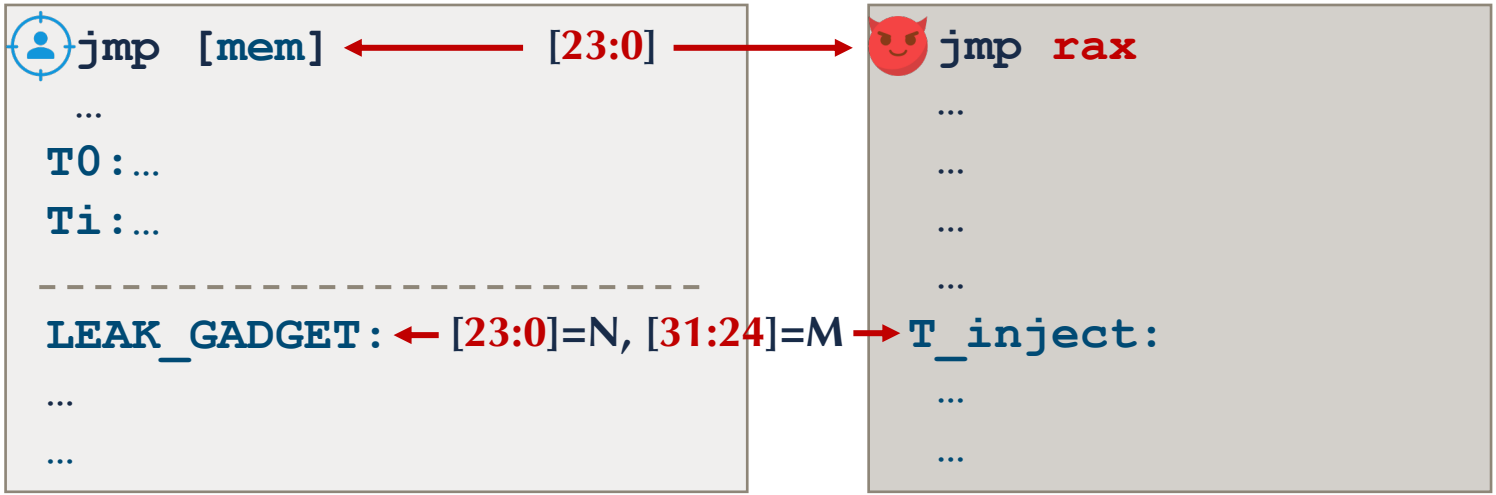
Breaking ASLR inside BTB

- **Step 3:** Break `LEAK_GADGET`[31:24] by **brute-forcing** `T_Inject`[31:24]



Breaking ASLR inside BTB

- **Step 3:** Break **`LEAK_GADGET`**[31:24] by **brute-forcing** **`T_Inject`**[31:24]
  - Once BTB injection is detected, **`LEAK_GADGET`**[31:12] is recovered



Breaking ASLR inside BTB

- **Step 3:** Break **`LEAK_GADGET`[31:24]** by **brute-forcing `T_Inject`[31:24]**
  - Once BTB injection is detected, **`LEAK_GADGET`[31:12]** is recovered

Reduce the number of attempts from 2^20* to < 800!
- < 4 seconds
- ~ 96.5% success rate

* Evtyushkin, Dmitry, Dmitry Ponomarev, and Nael Abu-Ghazaleh. "Jump over ASLR: Attacking branch predictors to bypass ASLR." In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

Mitigation

- **Mitigations from vendors:**
  - Trigger IBPB more frequently: Transitions between all security domains
  - However, **HUGE** overhead (up to 50%)

- **Future secure BPU designs:**
  - More complex tag designs to provide fine-grained isolation
  - Encryption or randomization in table mapping policies

Conclusion

- **Reverse Engineering:**
  - Precise IBP structure for the first time
  - Intel BTI defenses for the first time

- **Attacks:**
  - Two new high-precision target injection attacks
  - One new method of breaking ASLR inside BTB

Thank you!

Check our website for more details!