SpecDoctor: Differential Fuzz Testing to Find Transient Execution Vulnerabilities

Jaewon Hur, Suhwan Song, Sunwoo Kim*, Byoungyoung Lee

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

*Samsung Research

hurjaewon@snu.ac.kr

https://compsec.snu.ac.kr/people/jaewonhur/

MELTDOWN

SPECTRE

FORESHADOW

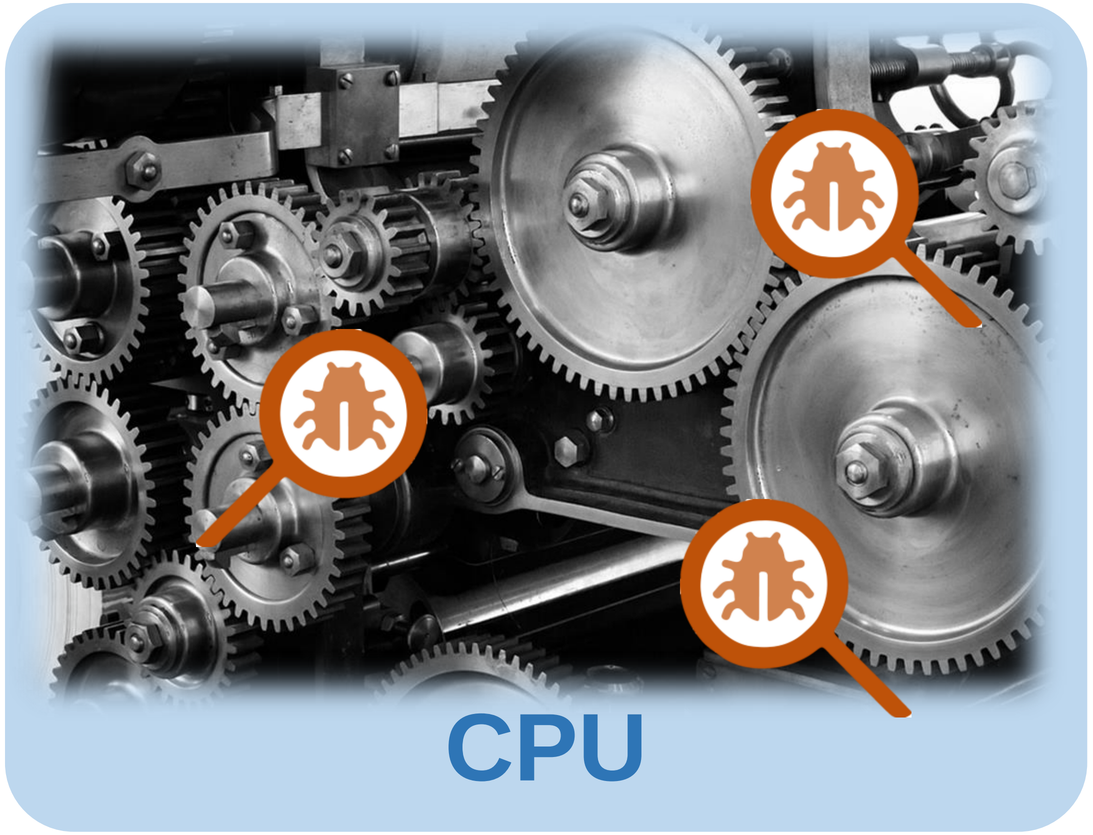
ZOMBIELOAD ATTACK

CacheOut

Leaking Data on Intel CPUs via Cache Evictions

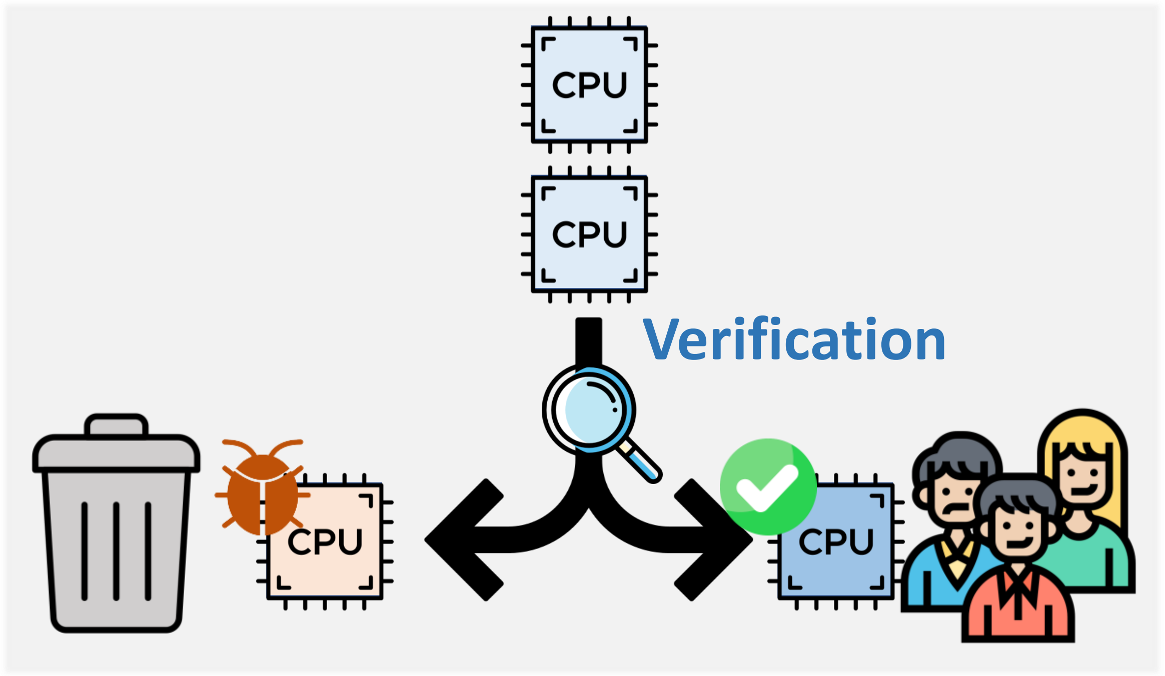
MDS

Importance of CPU Verification



CPUs cannot be fixed after they are released

Importance of CPU Verification



We should find bugs before releasing the chip

SpecDoctor: Differential Fuzz Testing to Find Transient Execution Vulnerabilities

SpecDoctor Found Real-world Vulnerabilities

10s of bugs in RISC-V Boom and NutShell

RISC-V

The Berkeley Out-of-Order Machine (**BOOM!**):
Computer Architecture Research Using an
Industry-Competitive, Synth
RISC-V Pro

Christopher Celio, K
David Patt
2015 Ju
celio@eecs.be

ASPIRE UC Berkeley

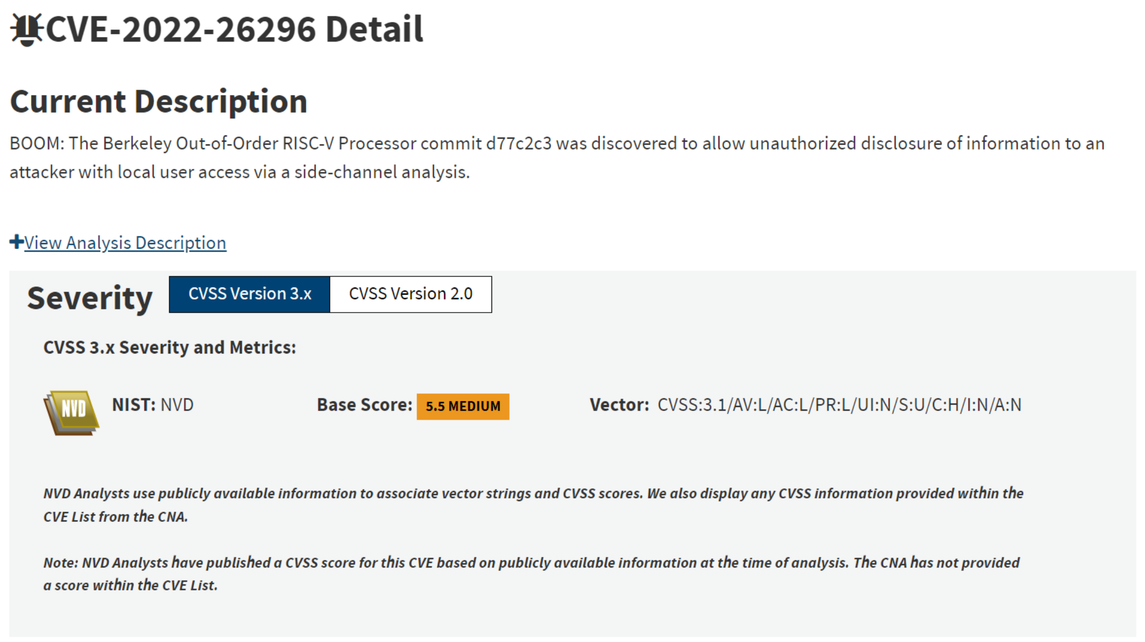
Tuesday, June 30, 15

RISC-V Global Forum

September 3, 2020

Nutshell: A Linux-Compatible RISC-V Processor Designed by Undergraduates

Huaqiang Wang, University of Chinese Academy of Sciences


CVE-2022-26296 Detail

Current Description

BOOM: The Berkeley Out-of-Order RISC-V Processor commit d77c2c3 was discovered to allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis.

+View Analysis Description

Severity CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 5.5 MEDIUM **Vector:** CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

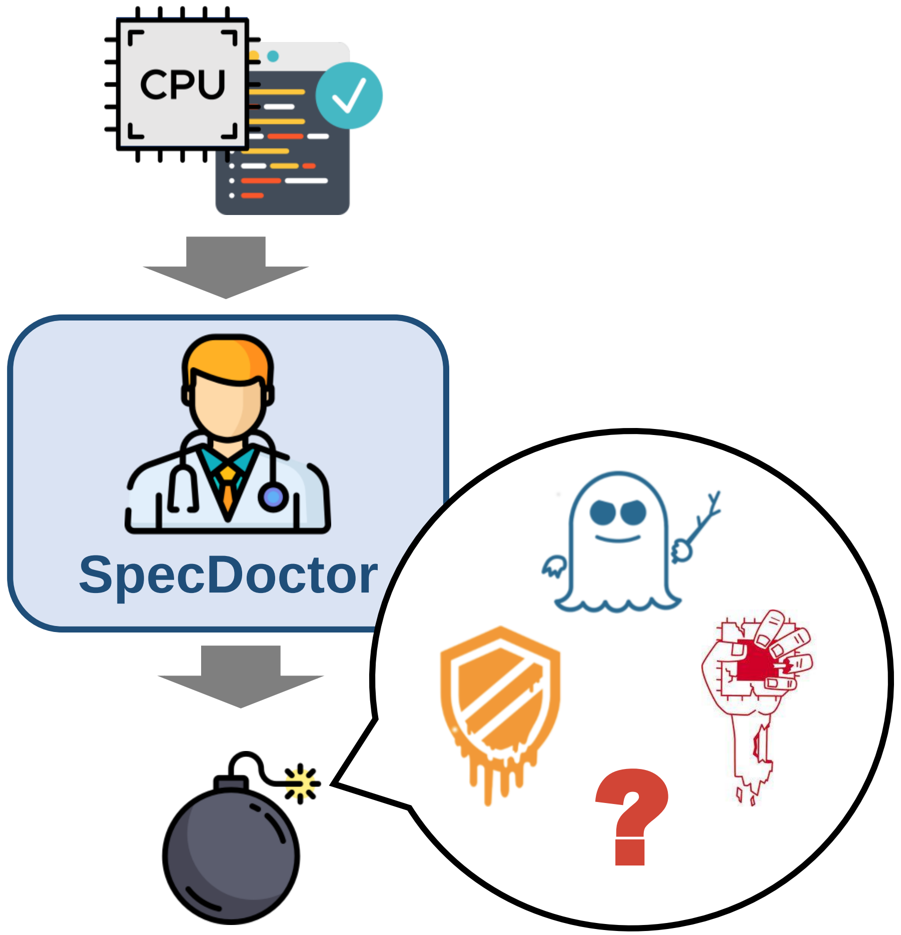
Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

What Does SpecDoctor Do?

Given the **CPU RTL** (Blueprint of the CPU),

SpecDoctor outputs **PoCs**

triggering transient execution vulnerabilities



Challenges of SpecDoctor

Transient Execution Vulnerability has countless attack vectors

In order to launch a Transient Execution Attack,

1. trigger a transient execution

Many ways to trigger a transient execution

BPU, BTB, RAS, TLB, Store Buffer, Line Fill Buffer, etc.

2. leak secret data in the transient execution

Many ways to leak secret data

I/D-Cache, BPU, TLB, FPU, Ex. Port, Replace logic, etc.

Approaches of SpecDoctor

SpecDoctor catches them **ALL AT ONCE**

1. Find instructions **triggering transient executions**
2. Find instructions **leaking secret data**

1. Detecting Transient Executions

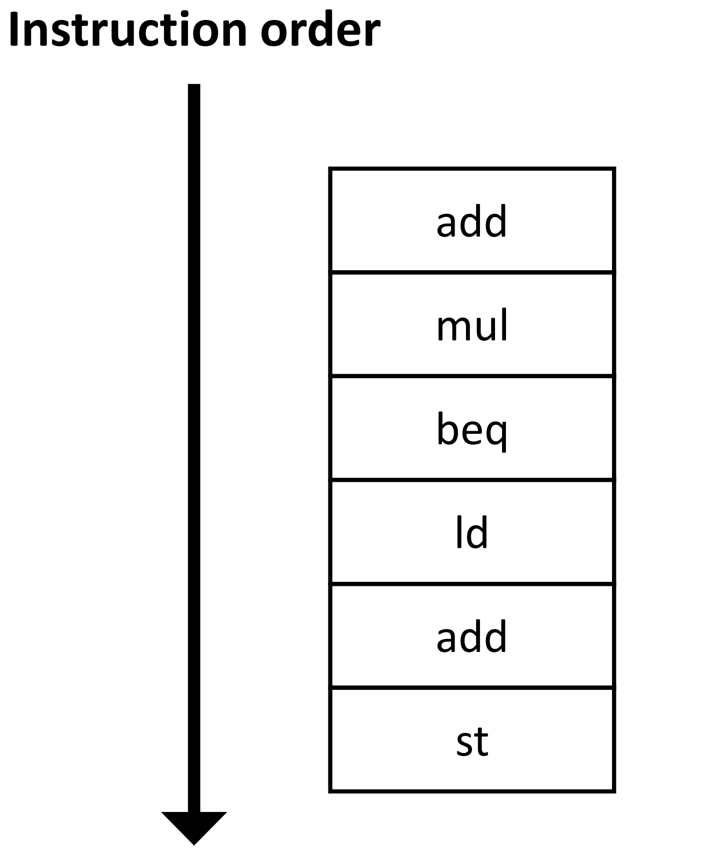
Transient Execution

Mispredicted execution inside a CPU,
which should be rollbacked later

1. Detecting Transient Executions

Transient Execution

Mispredicted execution inside a CPU, which should be rollbacked later



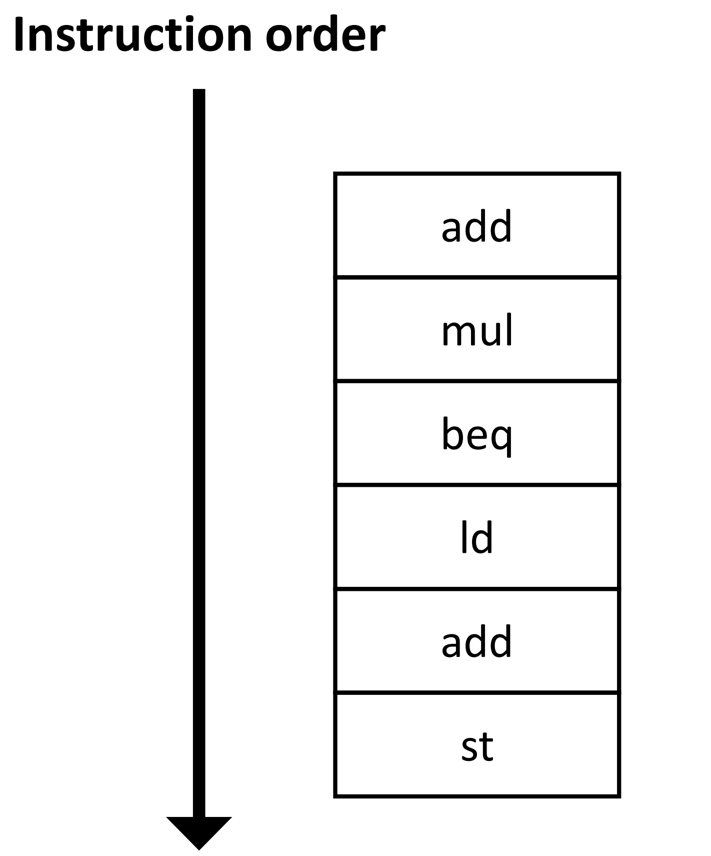
Out-of-order CPU

e.g.) Branch misprediction

1. Detecting Transient Executions

Transient Execution

Mispredicted execution inside a CPU, which should be rollbacked later



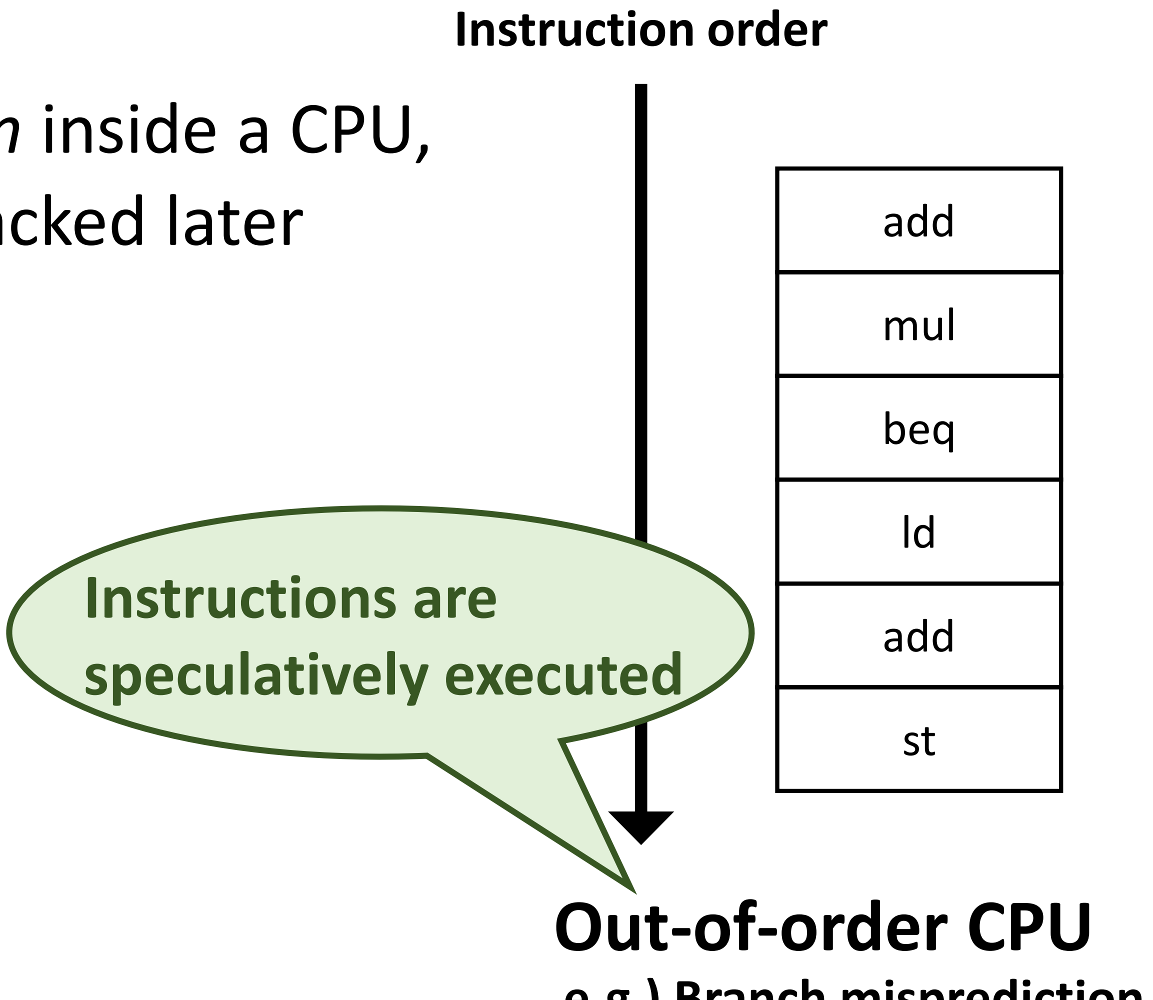
Out-of-order CPU

e.g.) Branch misprediction

1. Detecting Transient Executions

Transient Execution

Mispredicted execution inside a CPU, which should be rollbacked later

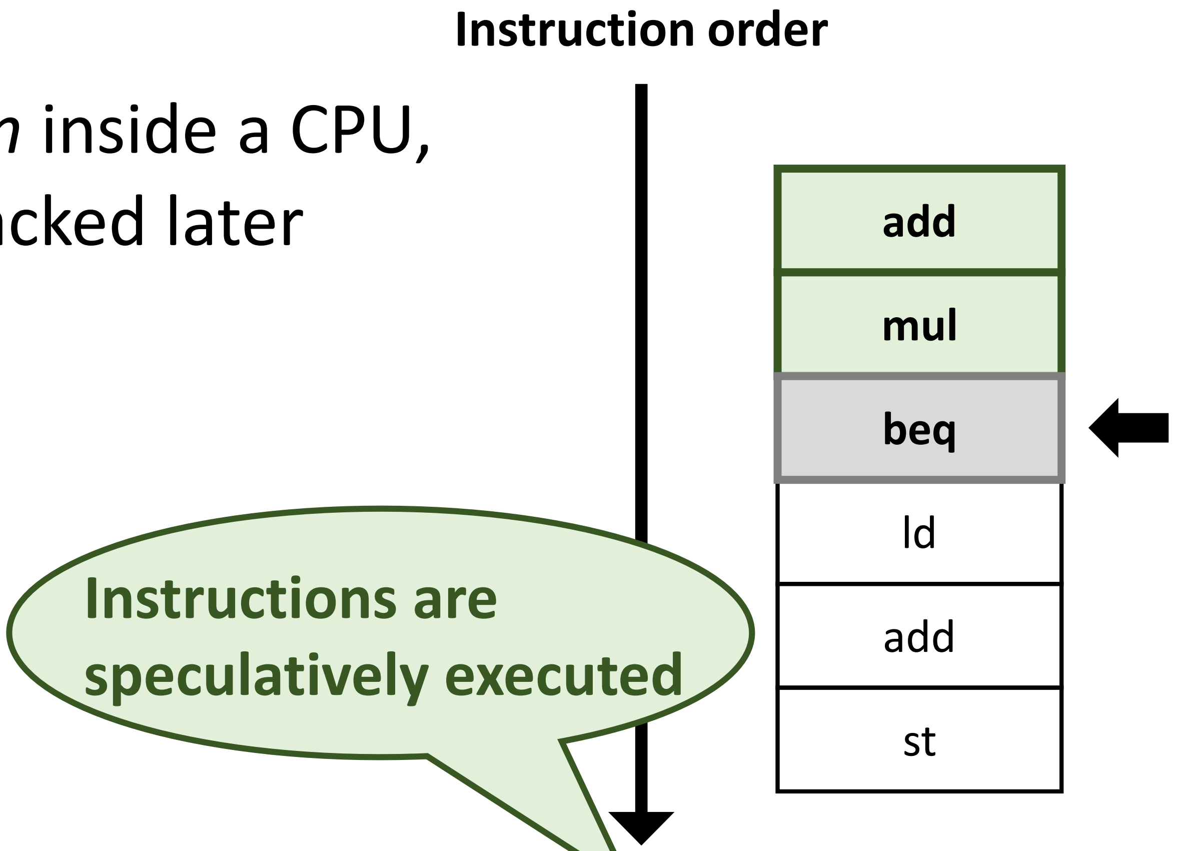


e.g.) Branch misprediction

1. Detecting Transient Executions

Transient Execution

Mispredicted execution inside a CPU, which should be rollbacked later



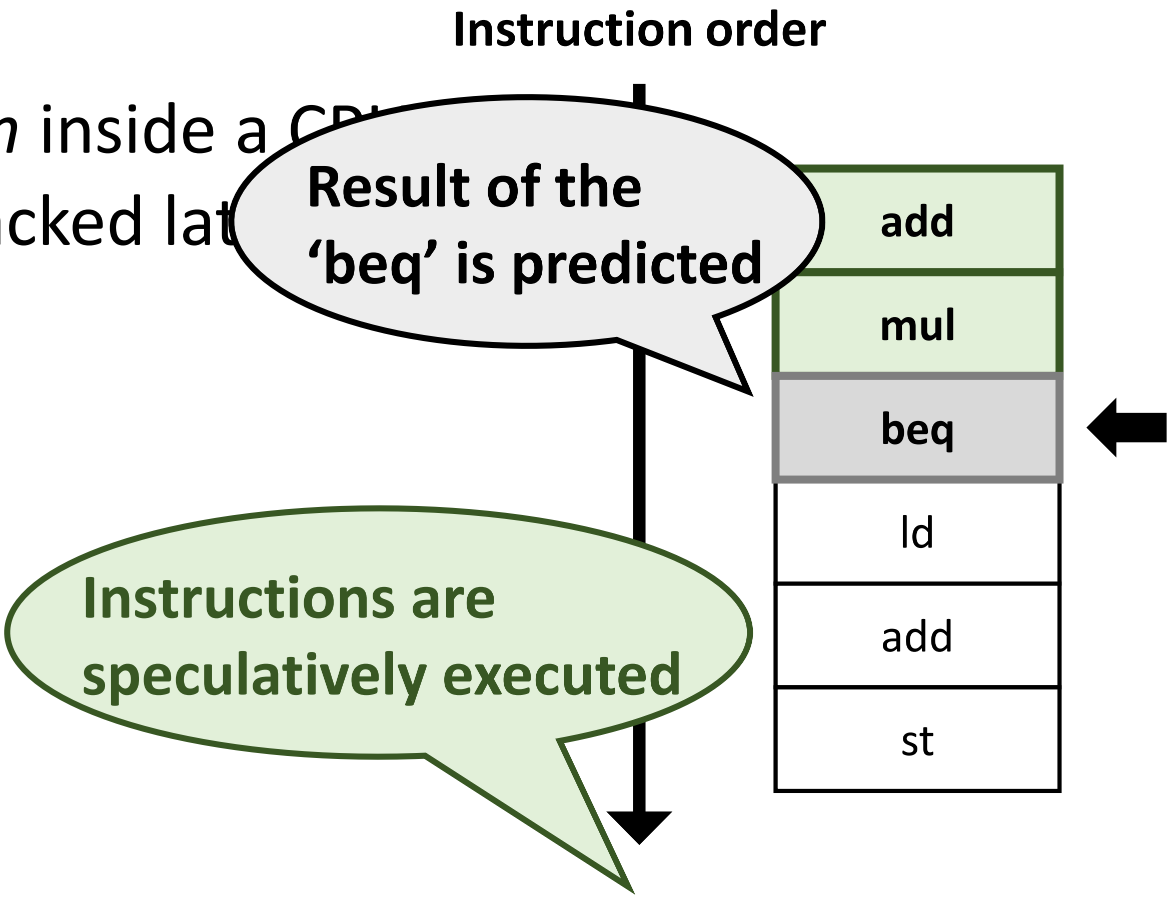
Out-of-order CPU

e.g.) Branch misprediction

1. Detecting Transient Executions

Transient Execution

Mispredicted execution inside a CP
which should be rollbacked lat


Instruction order
Result of the 'beq' is predicted
add
mul
beq
ld
add
st
Instructions are speculatively executed

Out-of-order CPU

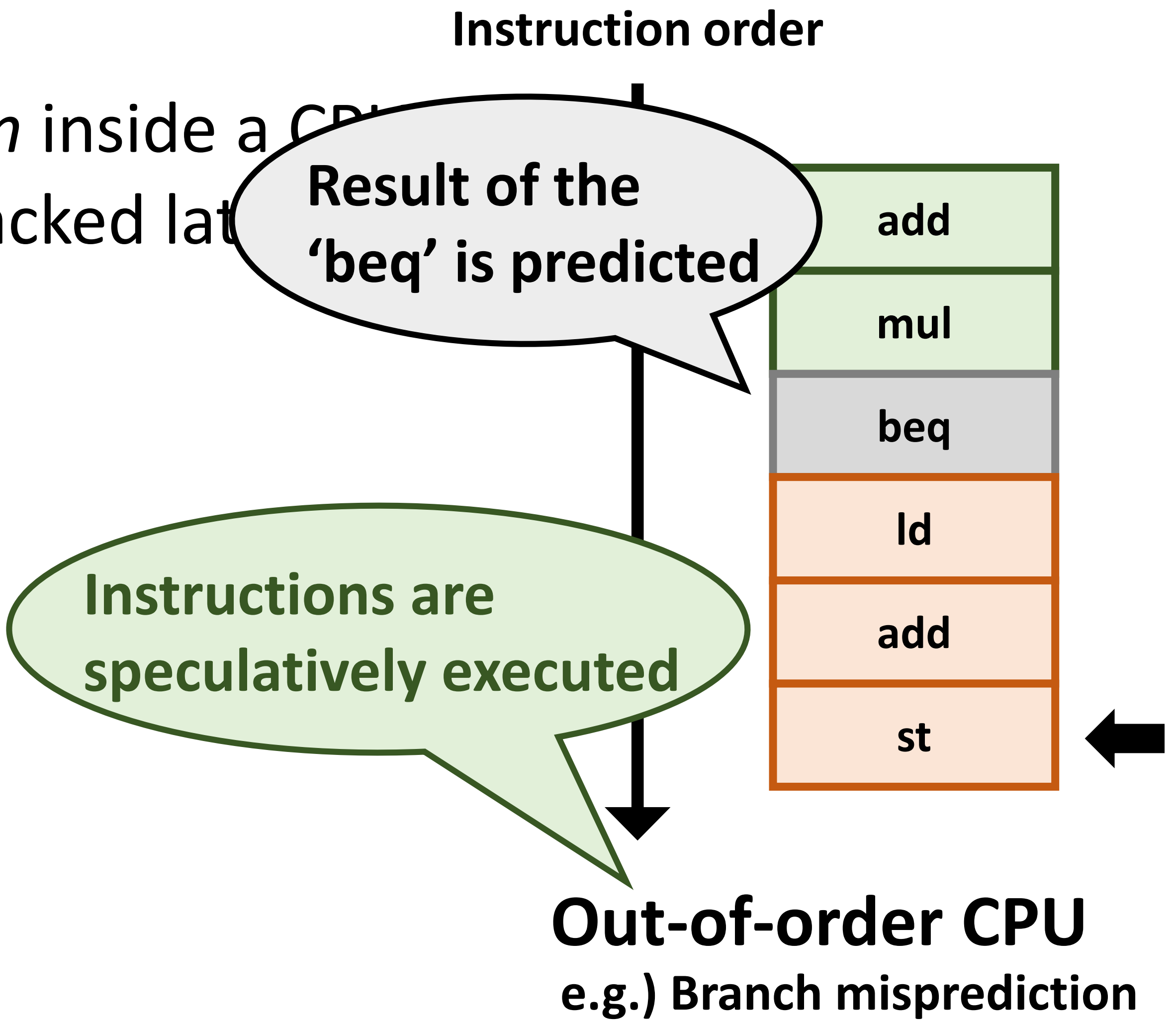
e.g.) Branch misprediction

1. Detecting Transient Executions

Transient Execution

Mispredicted execution inside a CP

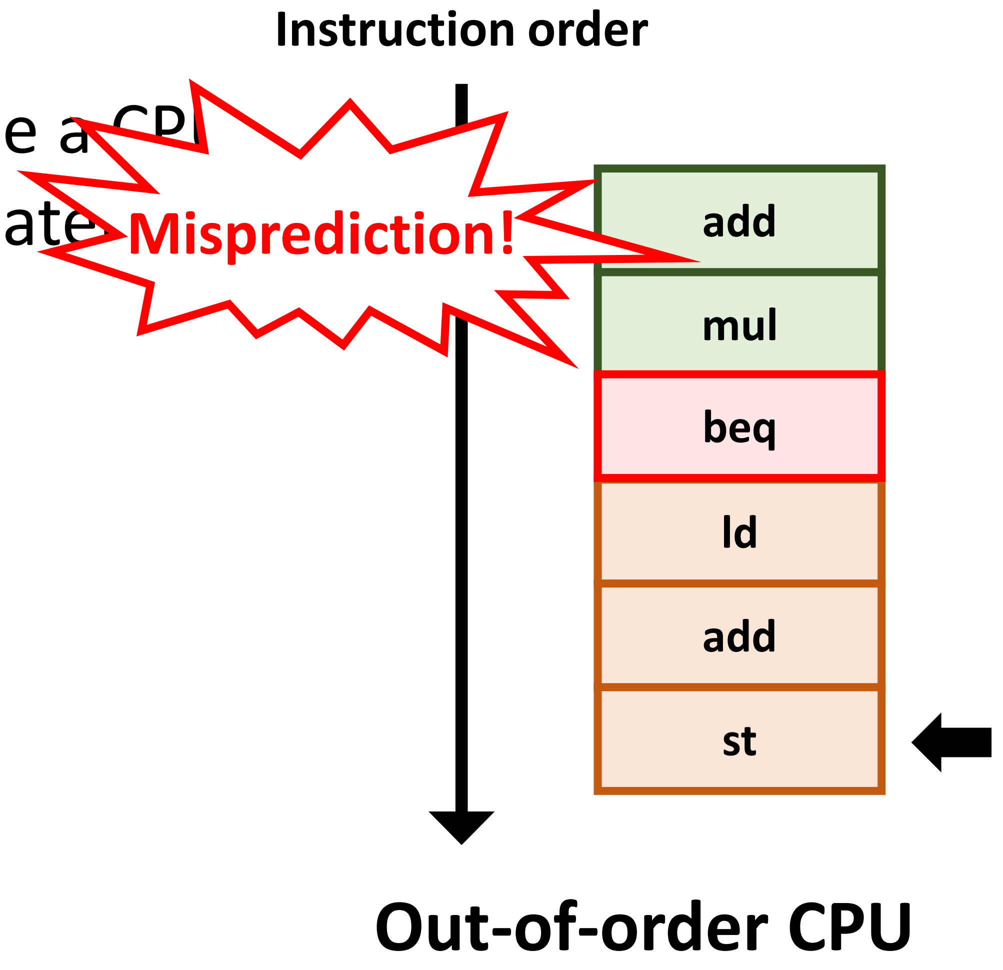
which should be rollbacked lat


Instruction order
Result of the 'beq' is predicted
add
mul
beq
ld
add
st
Instructions are speculatively executed
Out-of-order CPU
e.g.) Branch misprediction

1. Detecting Transient Executions

Transient Execution

Mispredicted execution inside a CPU
which should be rollbacked late

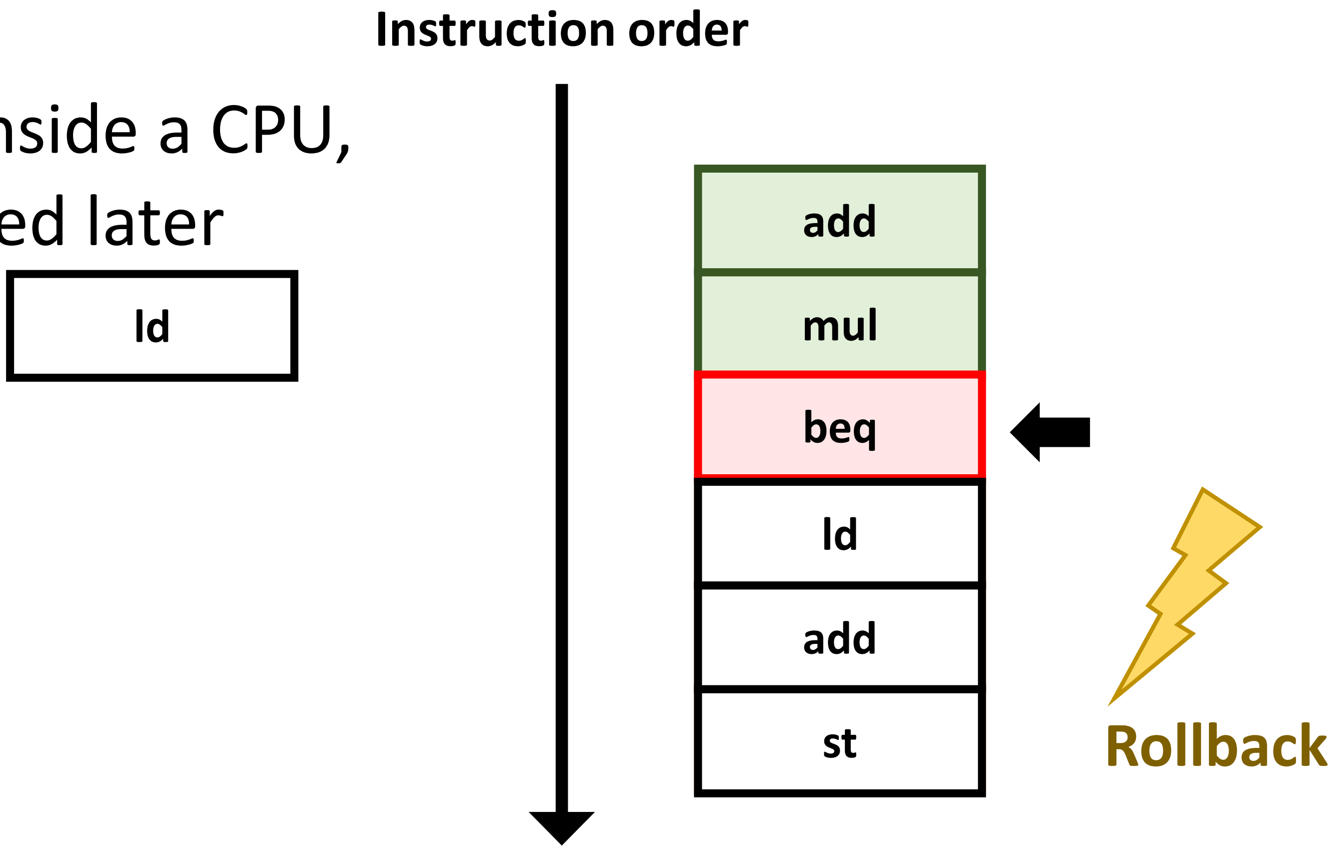


e.g.) Branch misprediction

1. Detecting Transient Executions

Transient Execution

Mispredicted execution inside a CPU, which should be rollbacked later



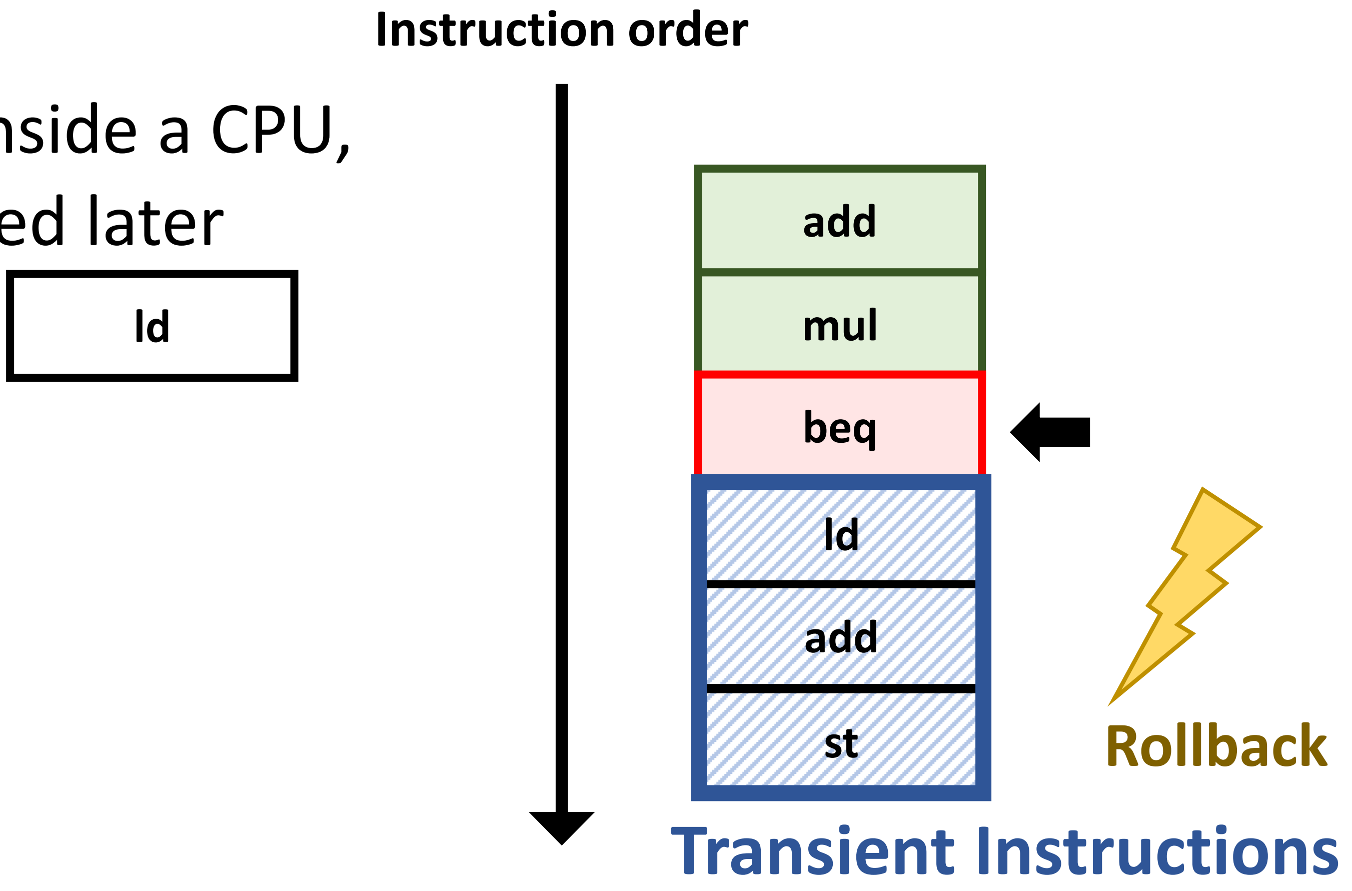
Out-of-order CPU

e.g.) Branch misprediction

1. Detecting Transient Executions

Transient Execution

Mispredicted execution inside a CPU, which should be rollbacked later



Out-of-order CPU

e.g.) Branch misprediction

1. Detecting Transient Executions

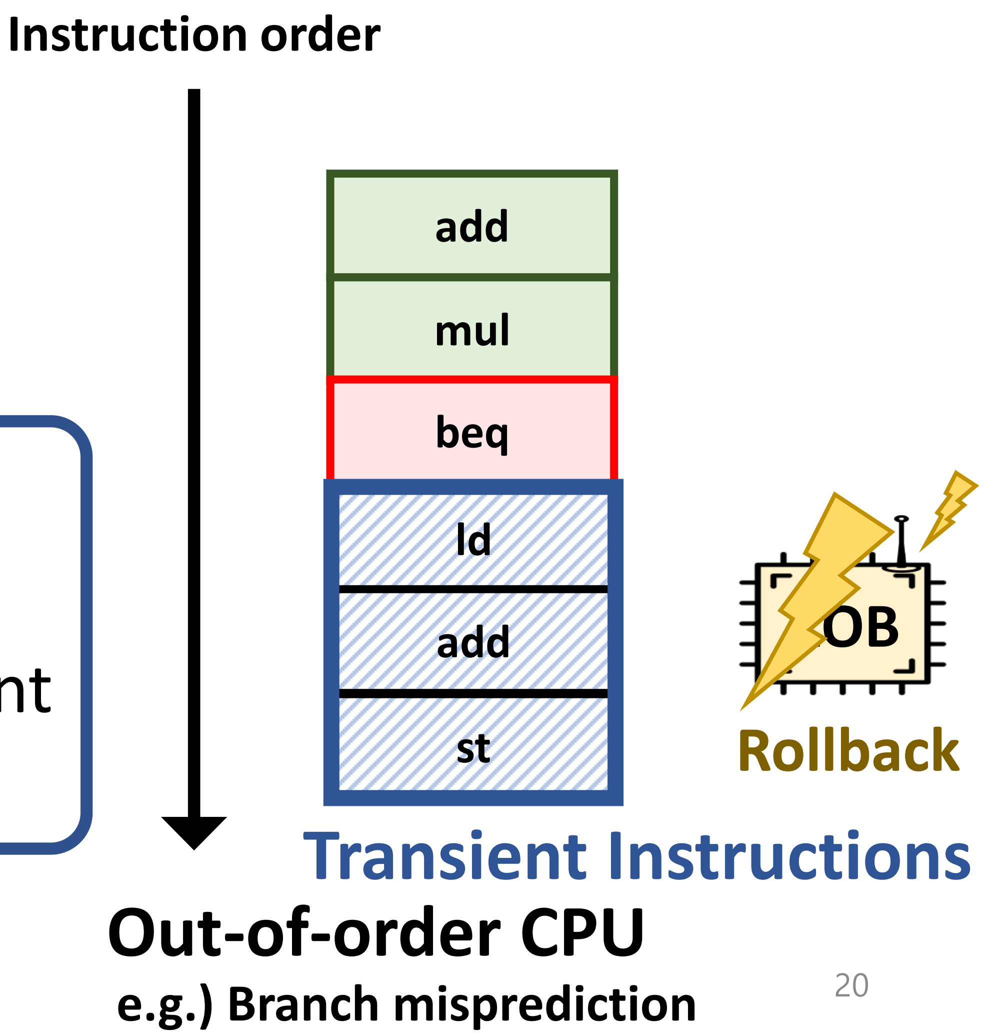
Transient Execution

Mispredicted execution inside a CPU, which should be rollbacked later

Observation

All transient execution should be **rollbacked**
(e.g., branch prediction, load-store bypass, TLB check, MDS)

Reorder Buffer (ROB) is a single handling point of **all rollbacks**



1. Detecting Transient Executions

Transient Execution

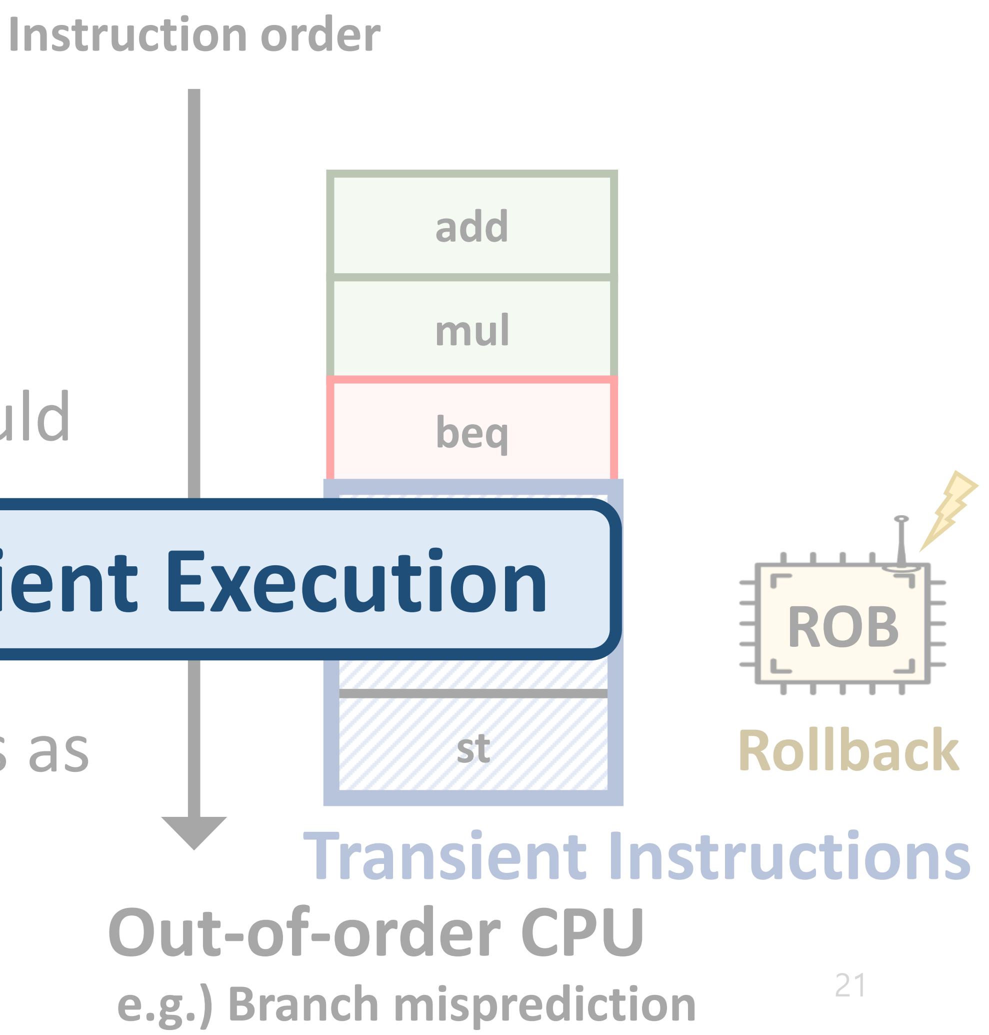
Mispredicted execution inside a CPU, which should be rollbacked later

Idea

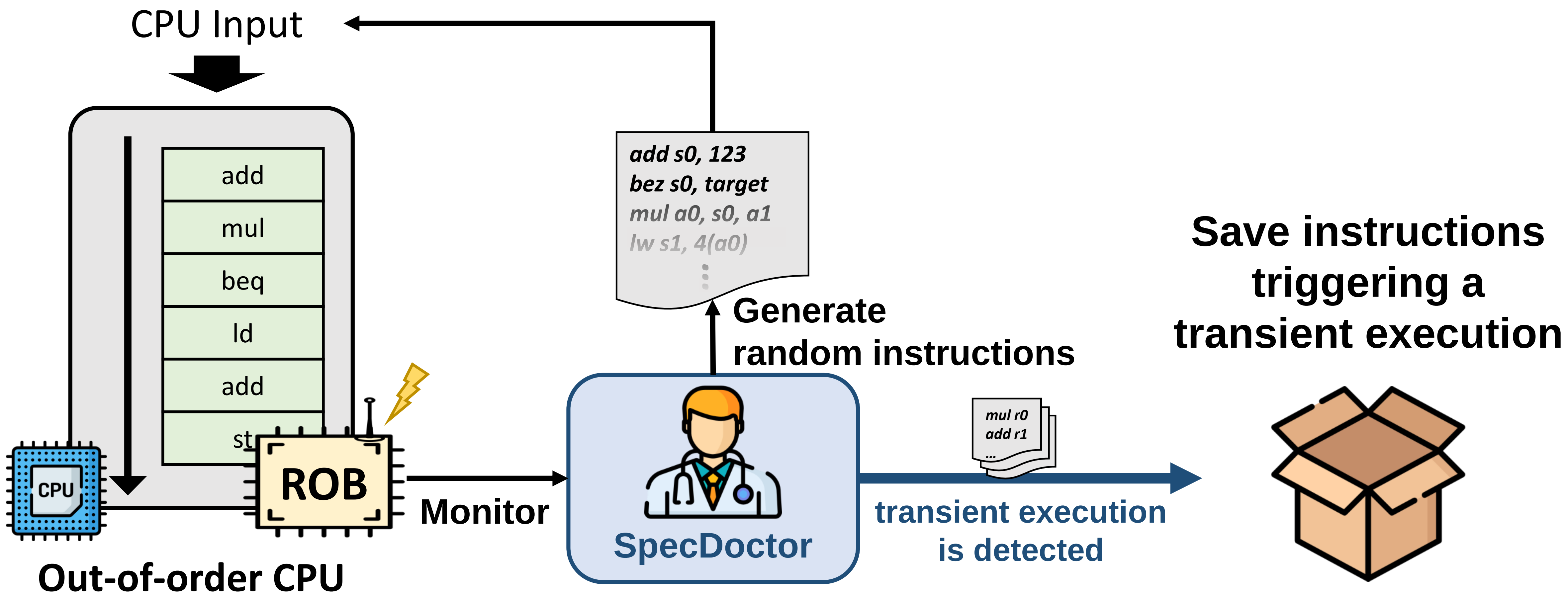
ation 1. All transient execution should

Monitoring RoB to Detect Transient Execution

Observation 2. Reorder Buffer (ROB) works as a single point for handling **all the rollbacks**



Step 1. Finding Instructions Triggering Transient Execution



2. Detecting Secret Leakage

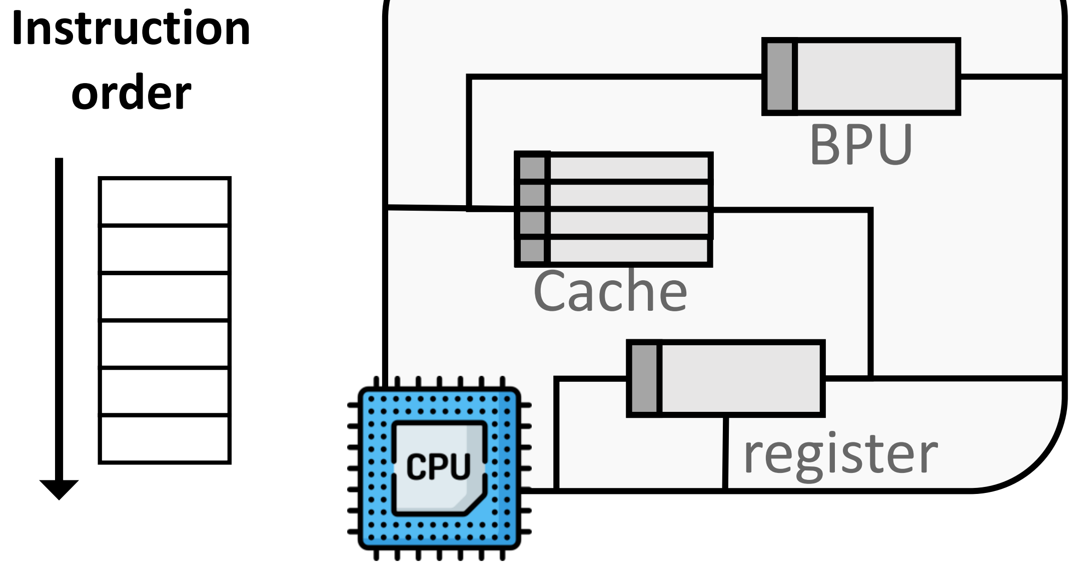
Micro-architectural Side Channel

Traces of transient execution in the CPU containing secret data

2. Detecting Secret Leakage

Micro-architectural Side Channel

Traces of transient execution in the CPU containing secret data

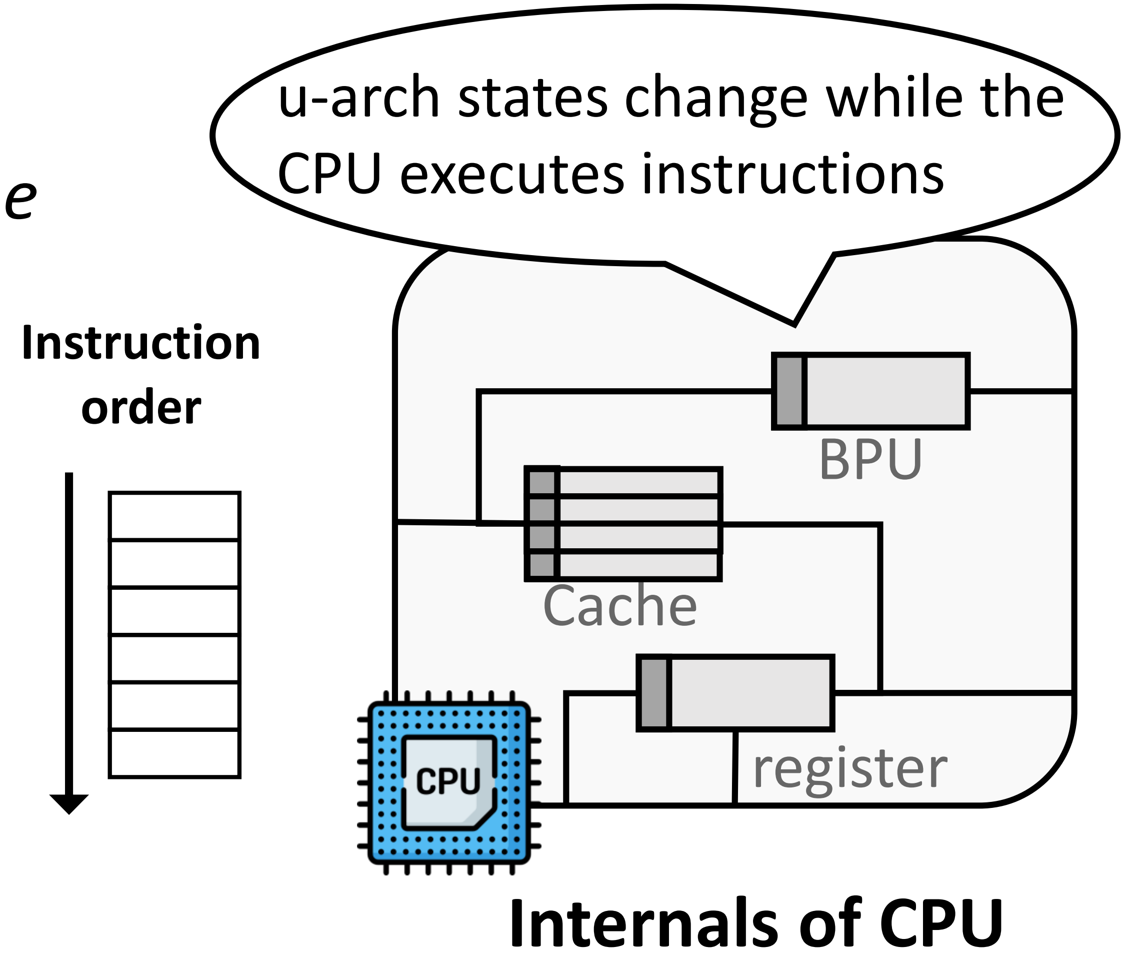


Internals of CPU

2. Detecting Secret Leakage

Micro-architectural Side Channel

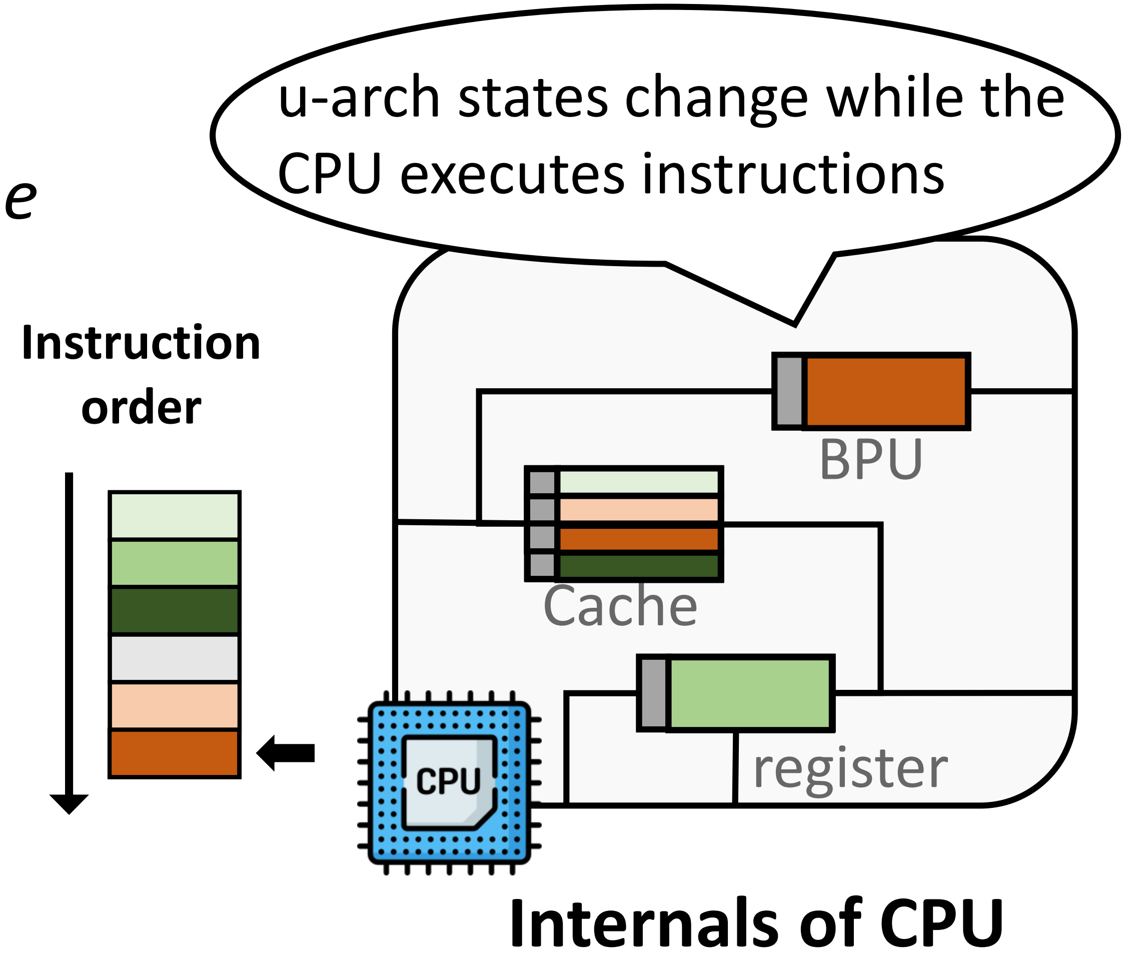
Traces of transient execution in the CPU containing secret data



2. Detecting Secret Leakage

Micro-architectural Side Channel

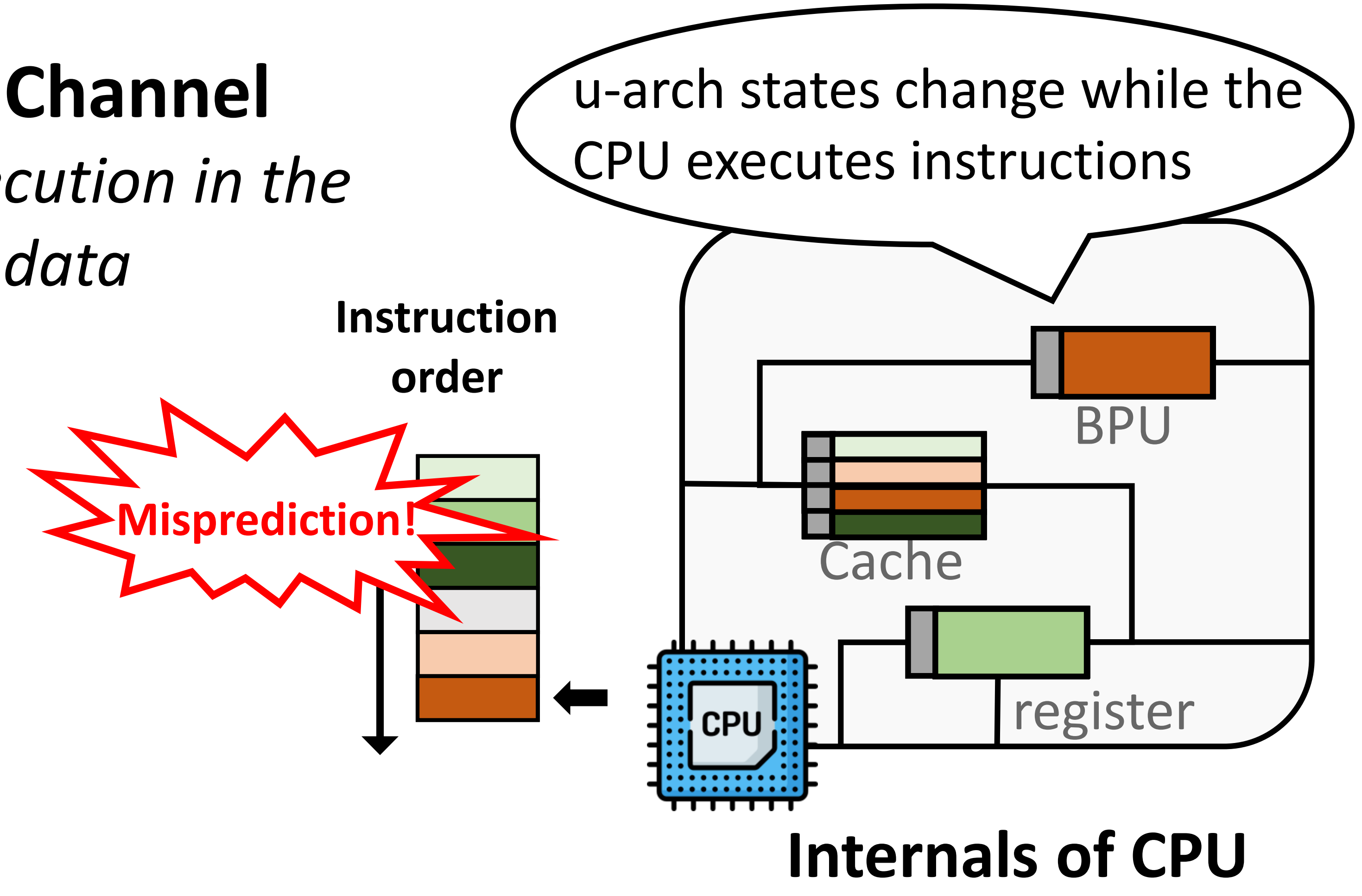
Traces of transient execution in the CPU containing secret data



2. Detecting Secret Leakage

Micro-architectural Side Channel

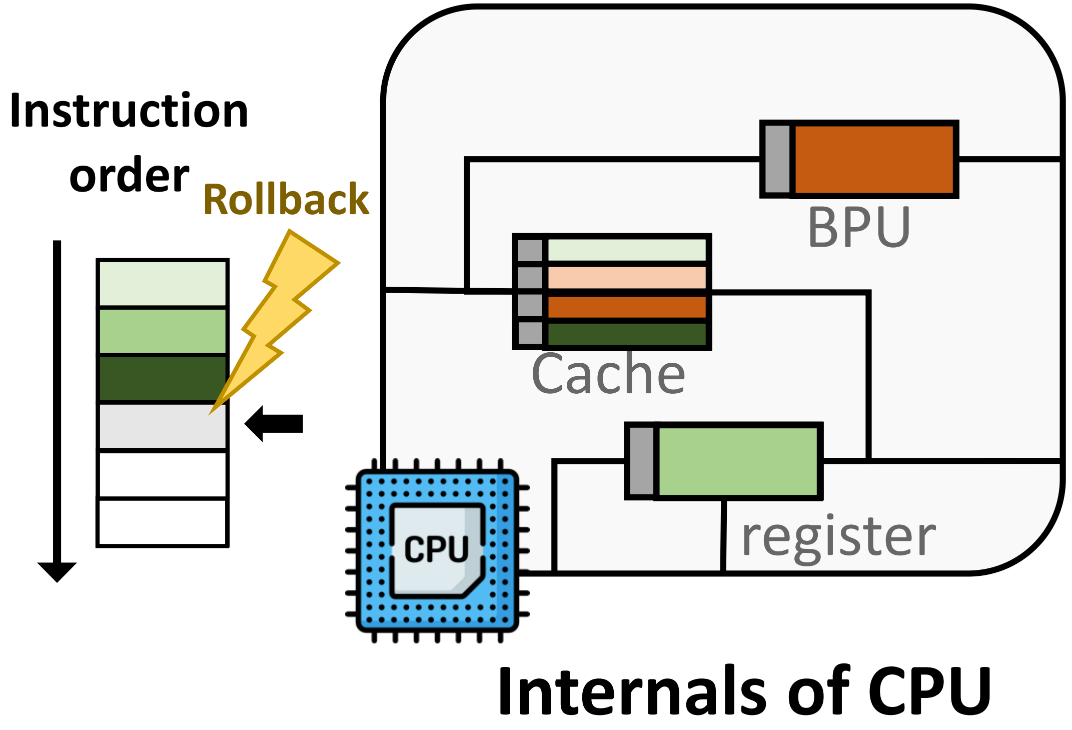
Traces of transient execution in the CPU containing secret data



2. Detecting Secret Leakage

Micro-architectural Side Channel

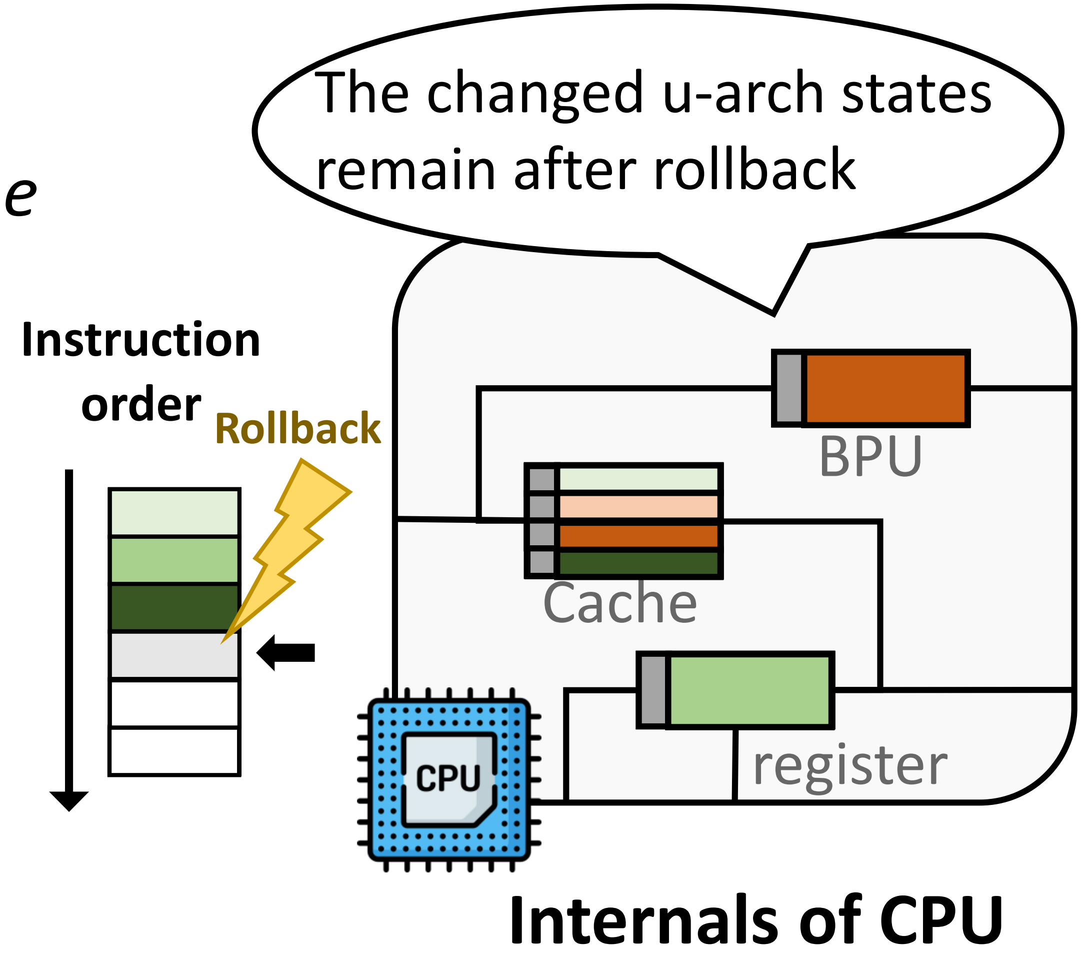
Traces of transient execution in the CPU containing secret data



2. Detecting Secret Leakage

Micro-architectural Side Channel

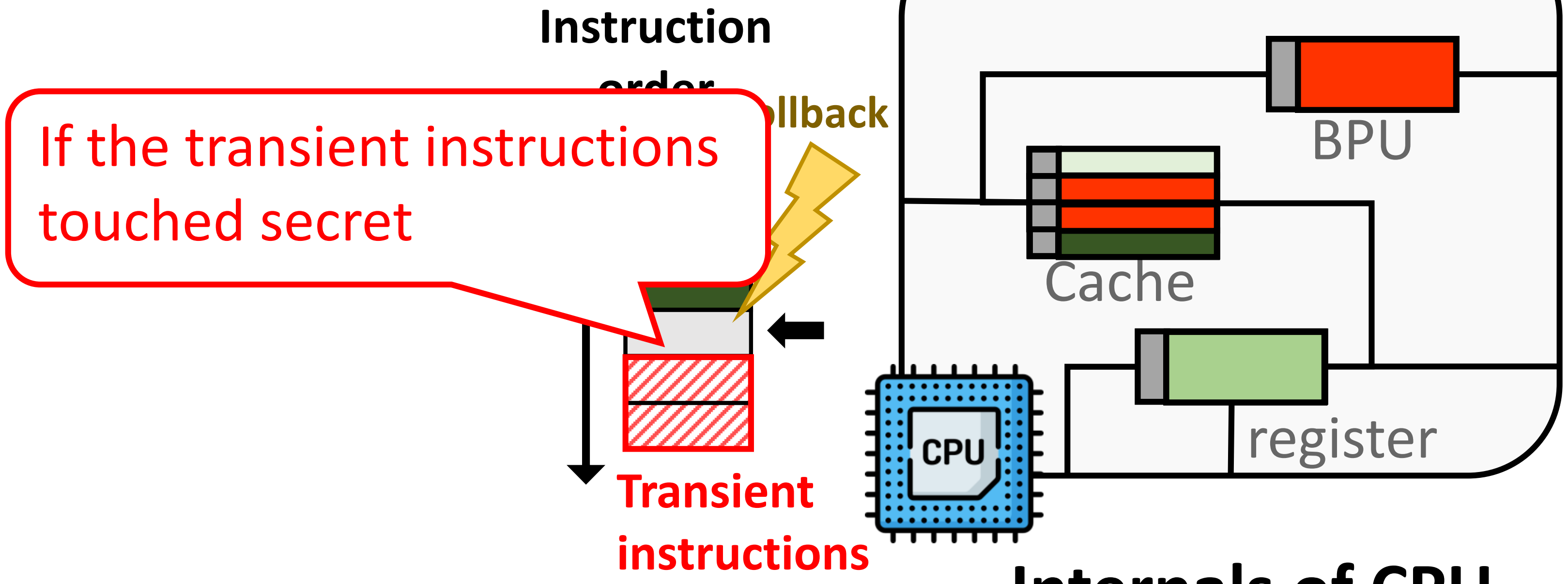
Traces of transient execution in the CPU containing secret data



2. Detecting Secret Leakage

Micro-architectural Side Channel

Traces of transient execution in the CPU containing secret data

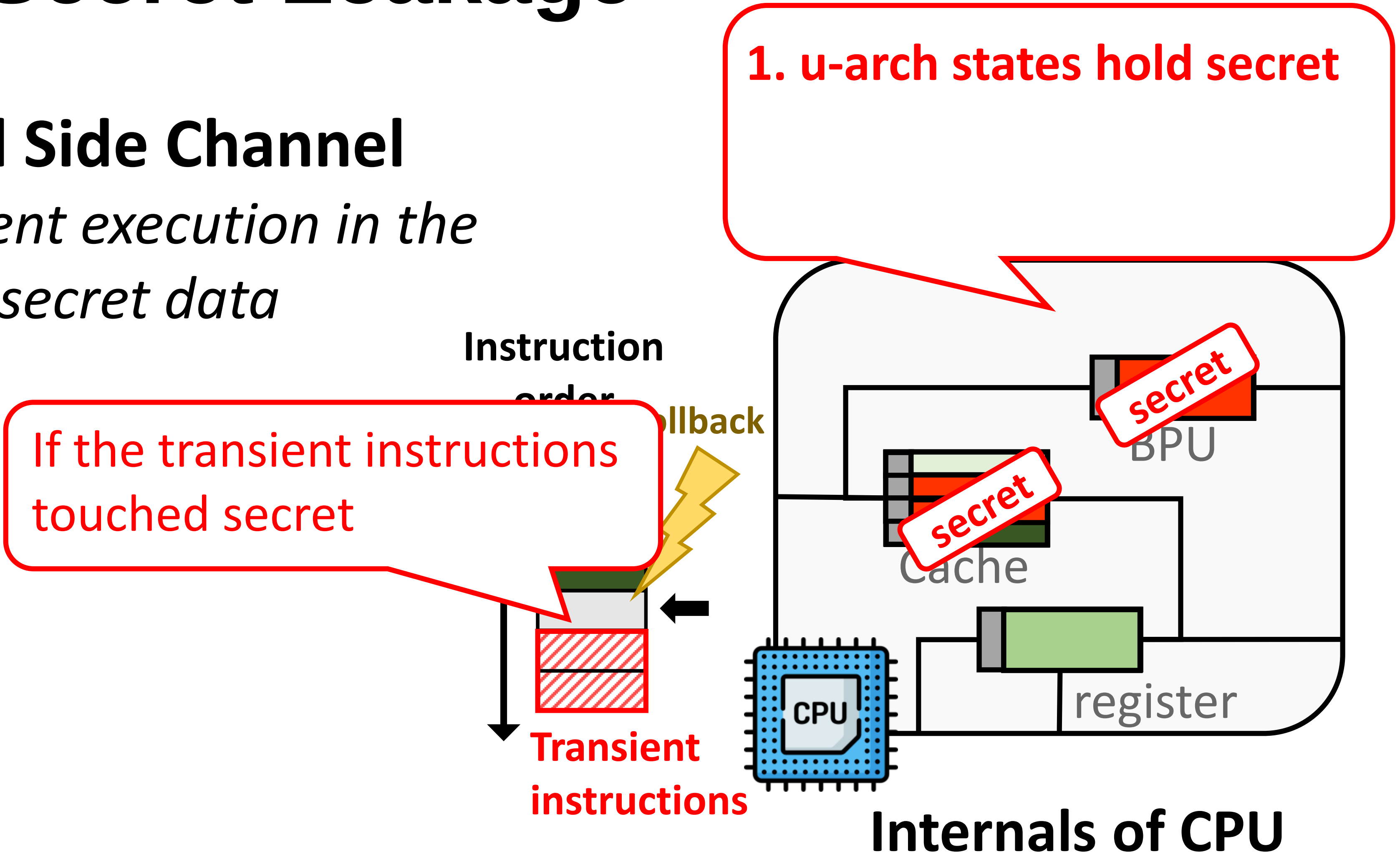


Internals of CPU

2. Detecting Secret Leakage

Micro-architectural Side Channel

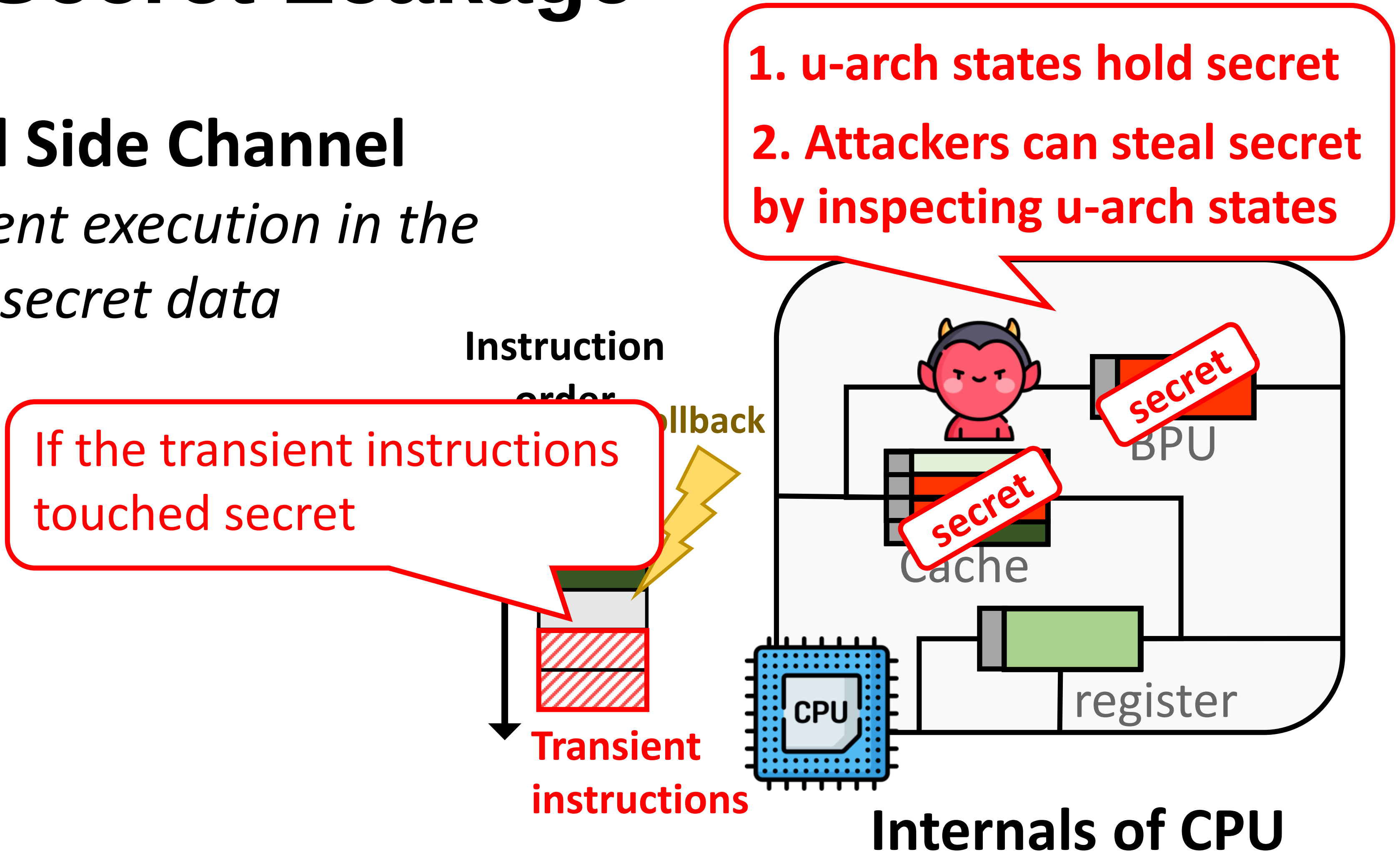
Traces of transient execution in the CPU containing secret data



2. Detecting Secret Leakage

Micro-architectural Side Channel

Traces of transient execution in the CPU containing secret data


1. u-arch states hold secret
2. Attackers can steal secret by inspecting u-arch states
Instruction order
If the transient instructions touched secret
llback
secret
BPU
secret
Cache
register
CPU
Transient instructions
Internals of CPU

2. Detecting Secret Leakage

Micro-architectural Side Channel

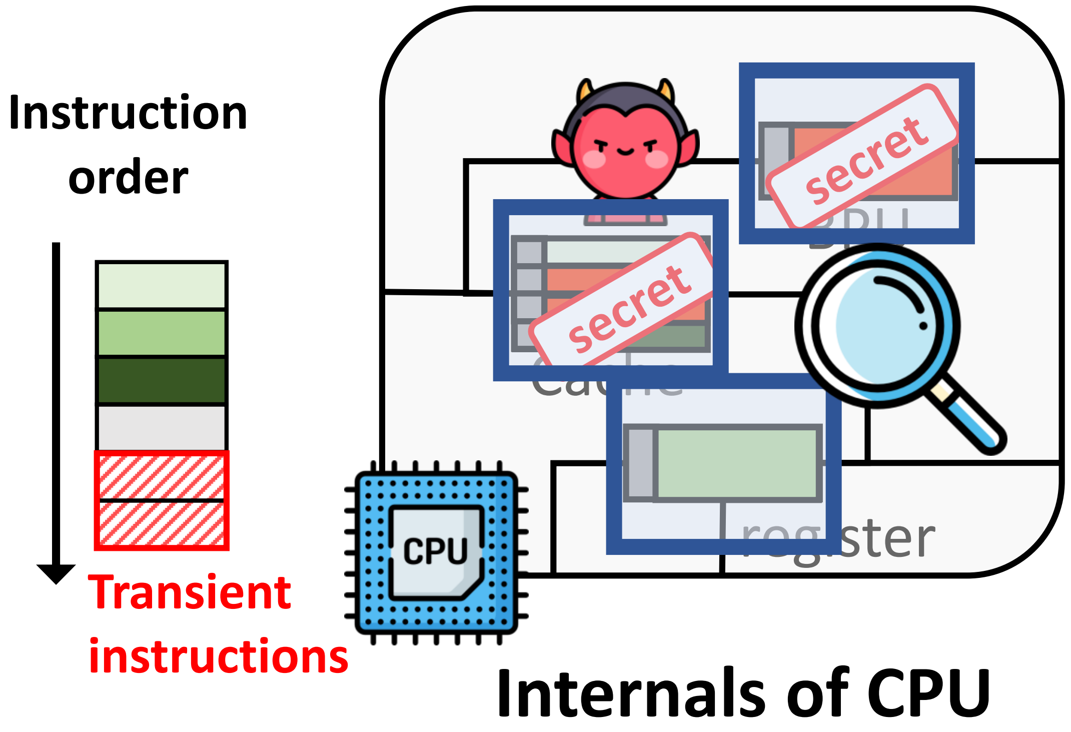
Traces of transient execution in the CPU containing secret data

Observation

All secret are transferred through **changed u-arch states**

(e.g., cache, BPU, TLB, FPU side channels)

u-arch states should be **different depending on the secret**



2. Detecting Secret Leakage

Micro-architectural Side Channel

Traces of transient execution in the CPU containing secret data

Idea

Differential Testing on u-arch states to find secret leakages

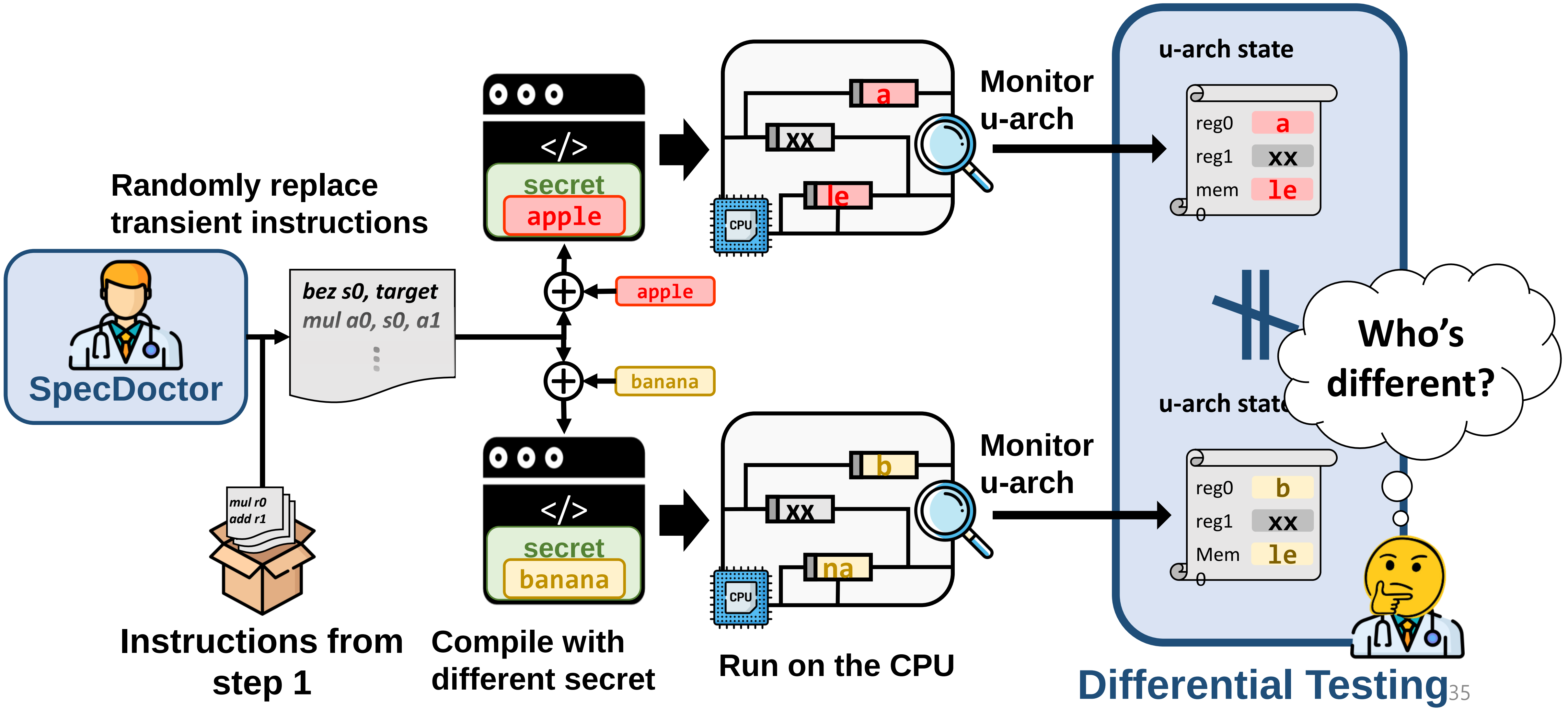
through **changed u-arch states**

(e.g., cache, BPU, TLB, FPU side channels)

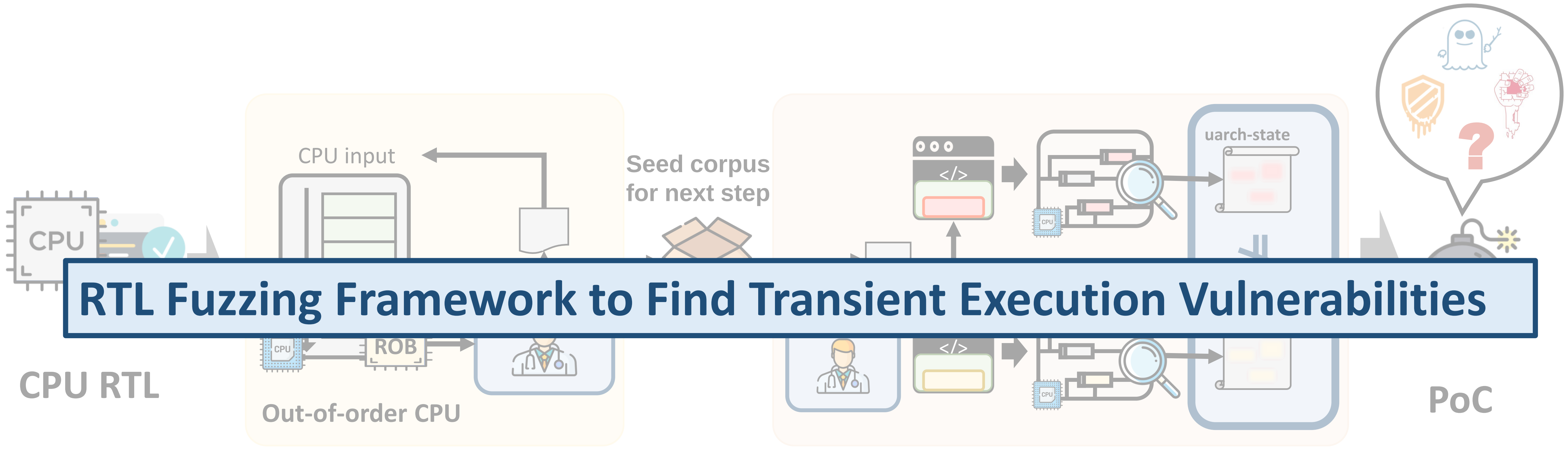
Observation 2. u-arch states should be **different depending on the secret**

Instruction
Transient instructions
CPU
register
Internals of CPU

Step 2. Finding Instructions Leaking Secret Data



Overall Framework of SpecDoctor

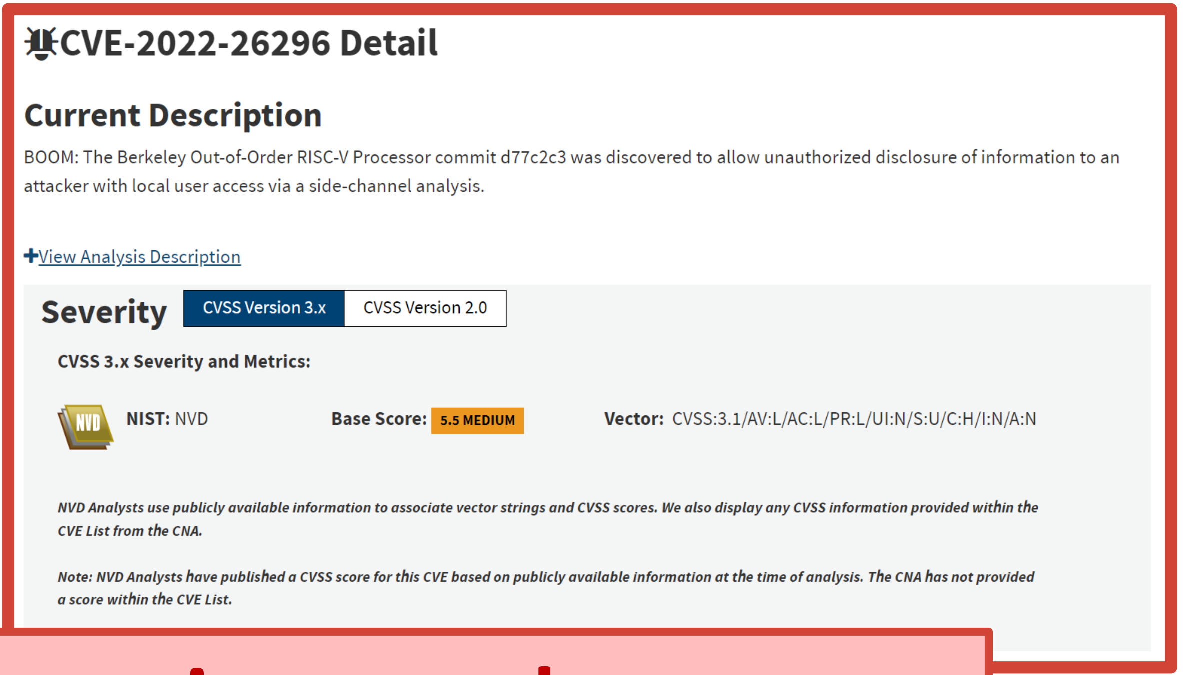


RTL Fuzzing Framework to Find Transient Execution Vulnerabilities

1. Find instructions triggering transient execution
2. Find instructions leaking secret data

Practical Impact of SpecDoctor

Project	Transient execution	Side channel
Boom	pmp/vm-fault	d-cache, bim, tlb,…
	bound check bypass	i/d-cache, ras, faubtb,…
	branch target corrupt	i/d-cache, btb, tlb,…
	load-store bypass	i/d-cache, bim, btb, …
NutShell	bound check bypass	i/d-cache, bim, tlb, …
	branch target corrupt	i/d-cache, ras, rs, …



CVE-2022-26296 Detail

Current Description

BOOM: The Berkeley Out-of-Order RISC-V Processor commit d77c2c3 was discovered to allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis.

+View Analysis Description

Severity CVSS Version 3.x CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 5.5 MEDIUM **Vector:** CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

First transient execution attack, exploiting the implementation bug in the CPU

Conclusion

- SpecDoctor is an RTL fuzzing framework to find transient execution vulnerabilities in CPU.

- https://github.com/compsec-snu/specdoctor.git

Thank you