CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

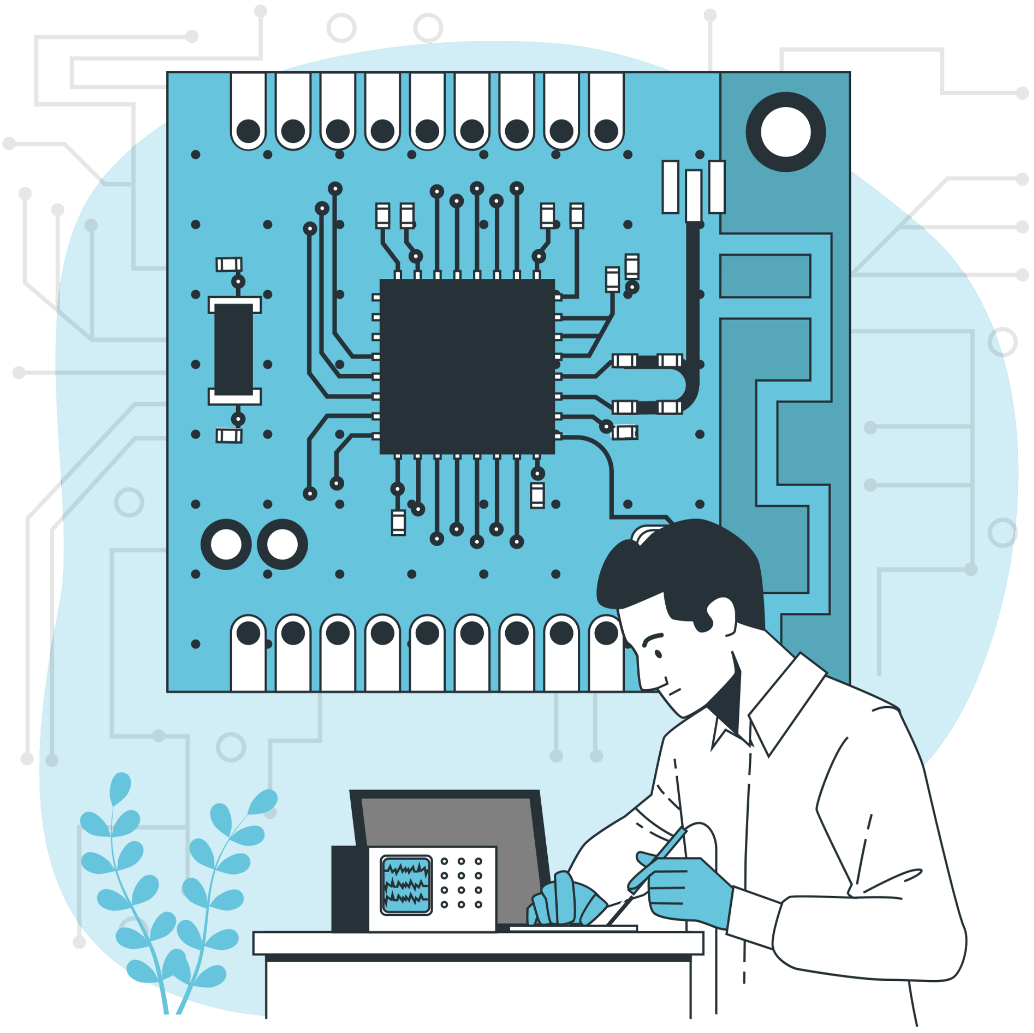
A Security RISC?

The State of Microarchitectural Attacks on RISC-V

Lukas Gerlach, **Daniel Weber**, Michael Schwarz | BlackHat EU 2023

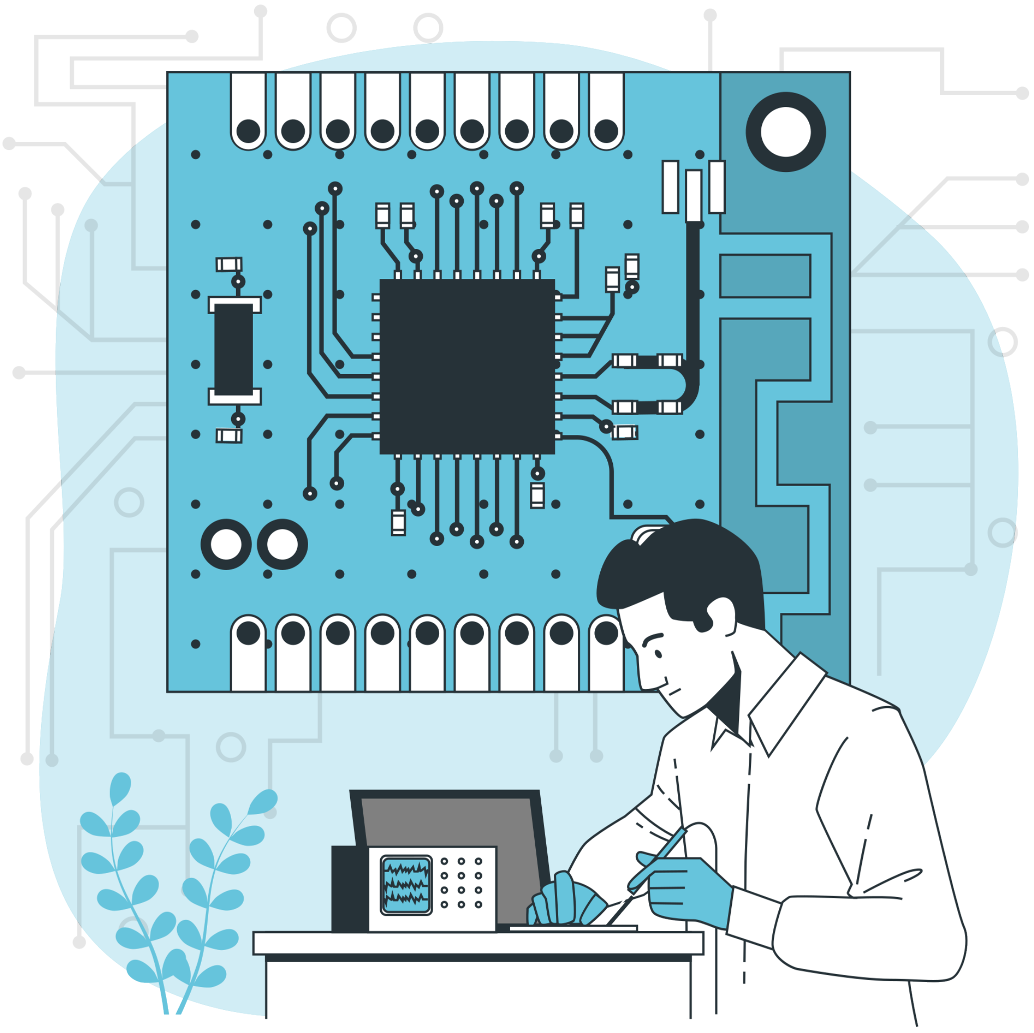
Agenda

Agenda



CPU Security Basics

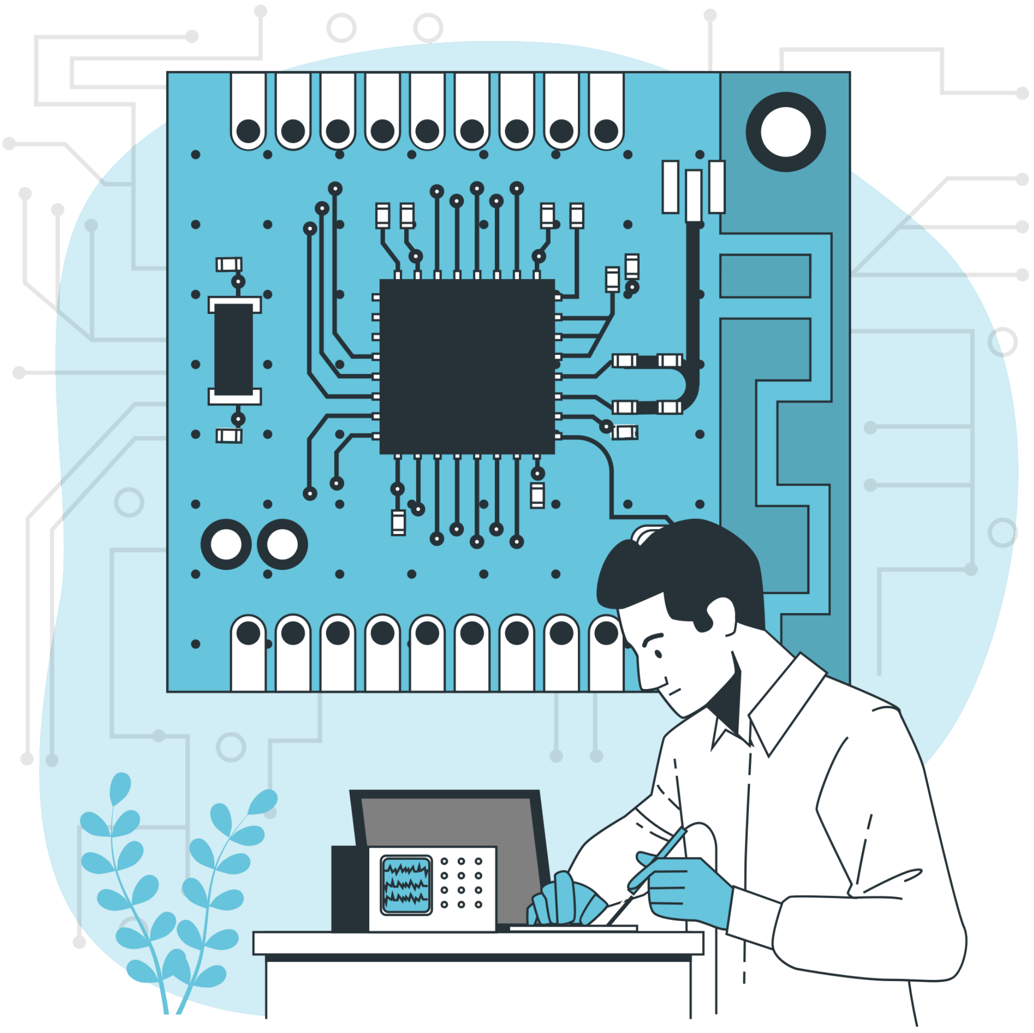
Agenda



CPU Security Basics

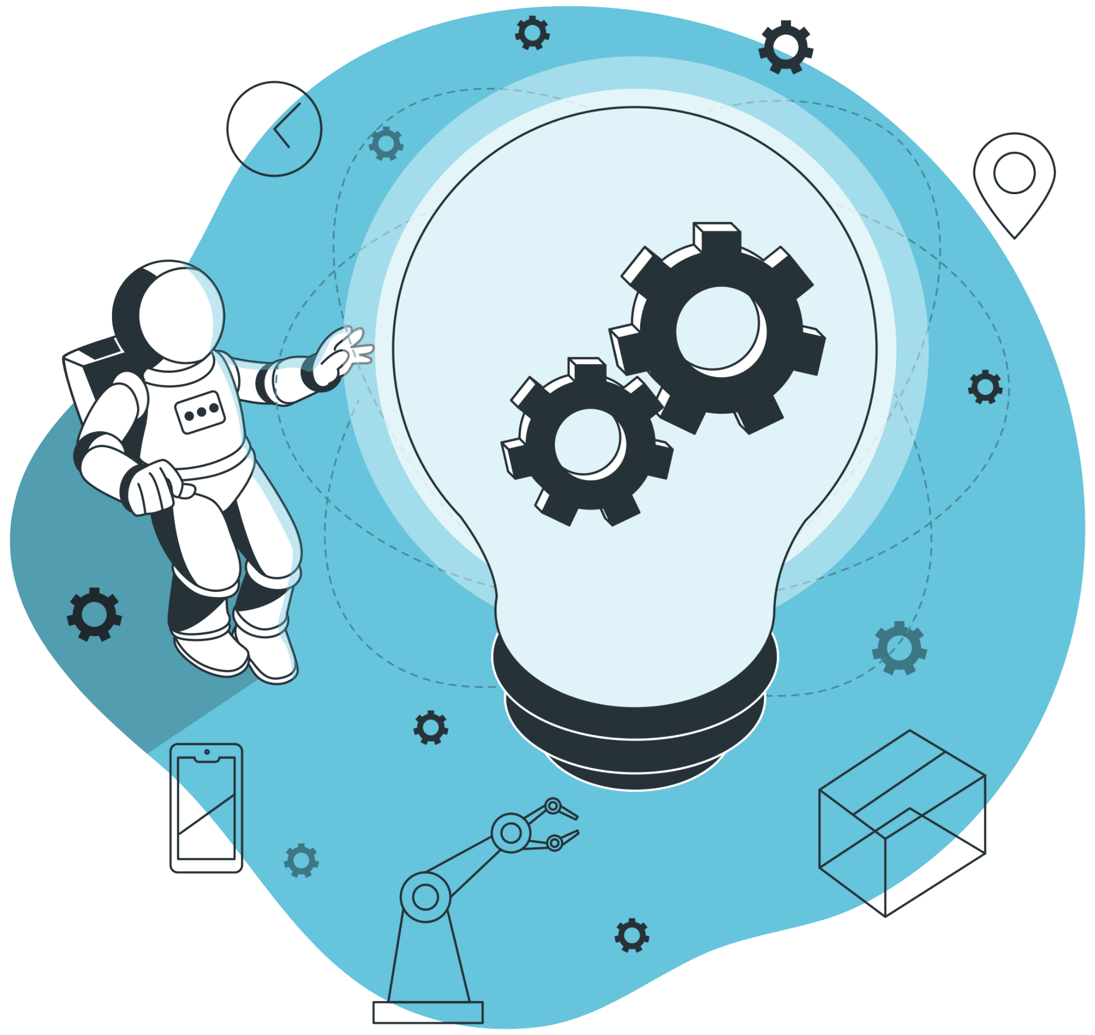
Learn about existing Attacks

Agenda



CPU Security Basics

Learn about existing Attacks



Investigate RISC-V Security

Who are we?

RootSec

Research Group

@

CISPA

Helmholtz Center for
Information Security

Who are we?

RootSec

Research Group

@

CISPA

Helmholtz Center for
Information Security

Lukas Gerlach

PhD Student

Who are we?

RootSec

Research Group

@

CISPA

Helmholtz Center for Information Security

Lukas Gerlach

PhD Student

Daniel Weber

PhD Student

Who are we?

RootSec

Research Group

@

CISPA

Helmholtz Center for
Information Security

Lukas Gerlach

PhD Student

Daniel Weber

PhD Student

Michael Schwarz

Faculty

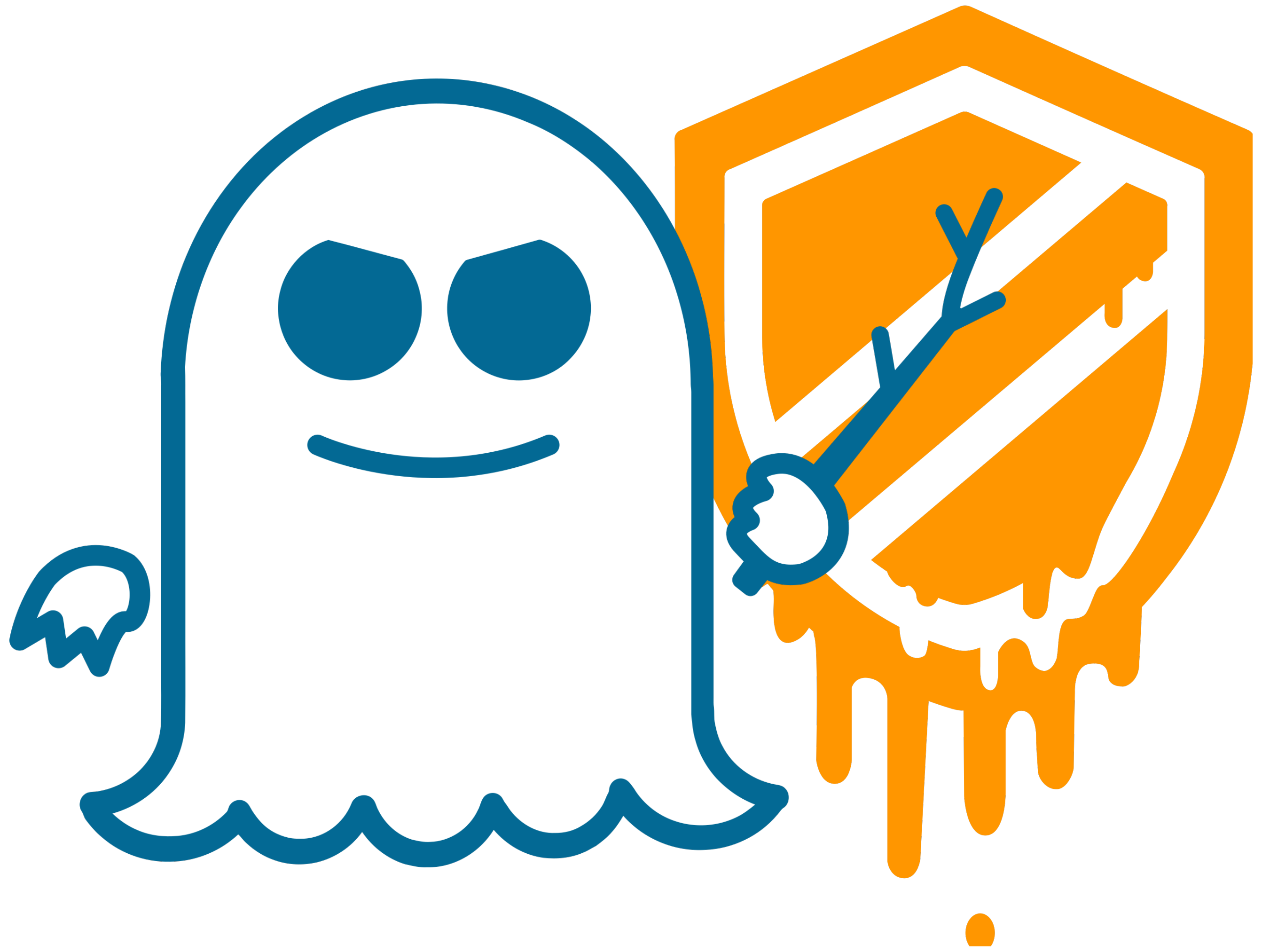
Why do we Care about CPU Security?

Live Demo

demo@lab24: ~/demos
⌥⌘1

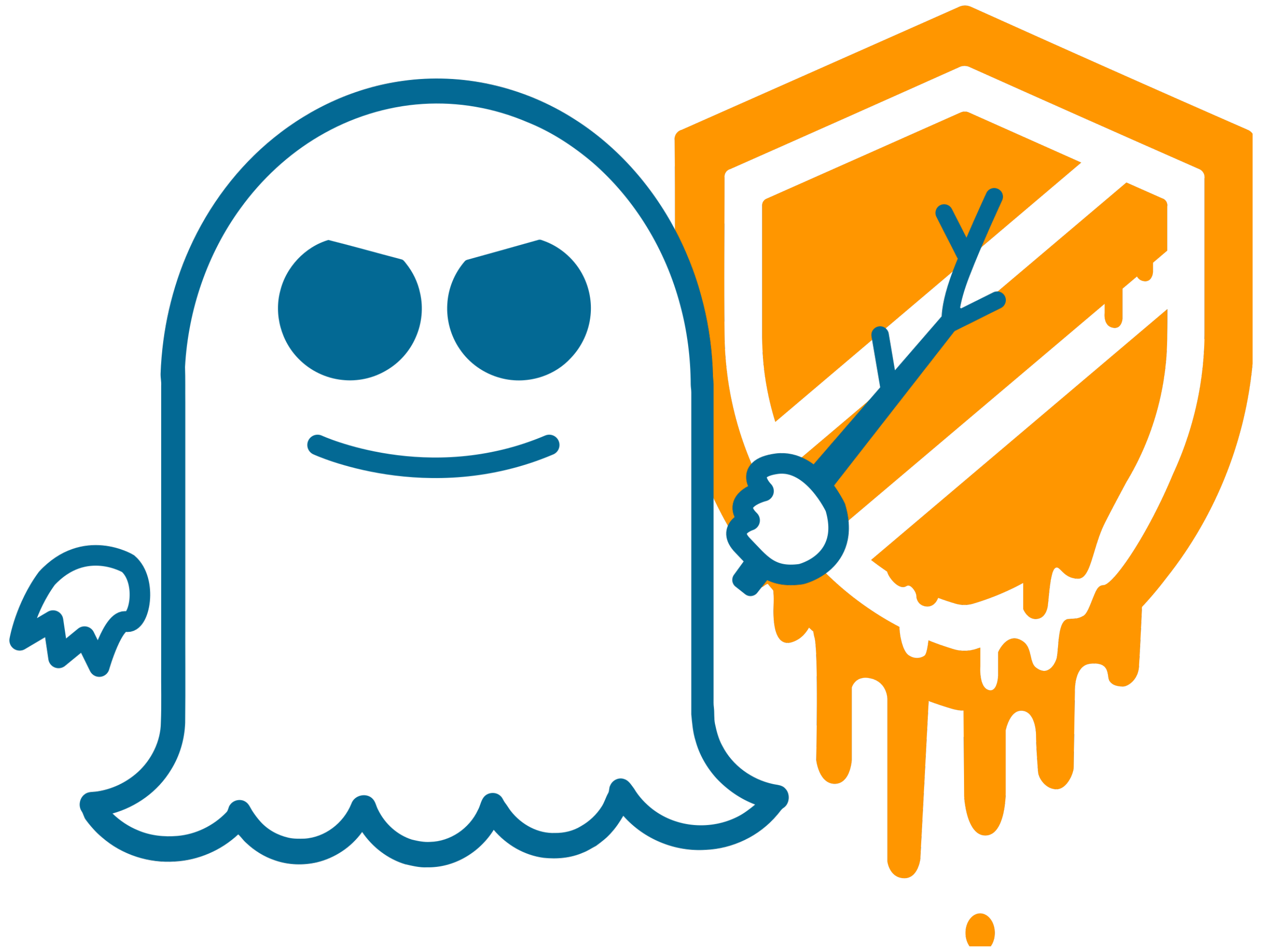
```
demo@lab24:~/demos$ ./leak
Calibrating Threshold
Cache hit timing: 5, Cache miss timing: 150
Threshold is: 101
```

Why do We Care about CPU Security?



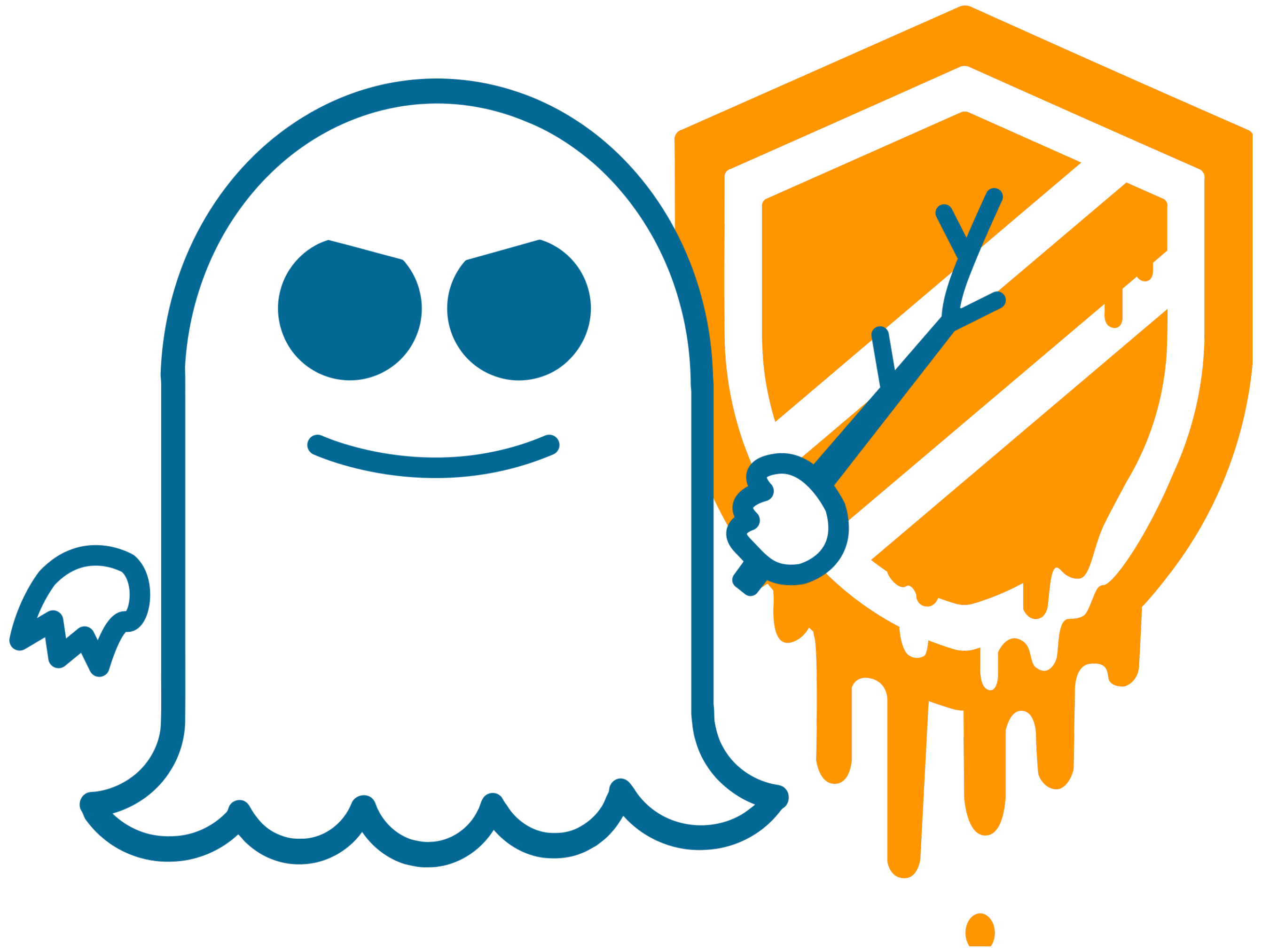
Why do We Care about CPU Security?

- **CPU vulnerabilities can leak or spy on...**



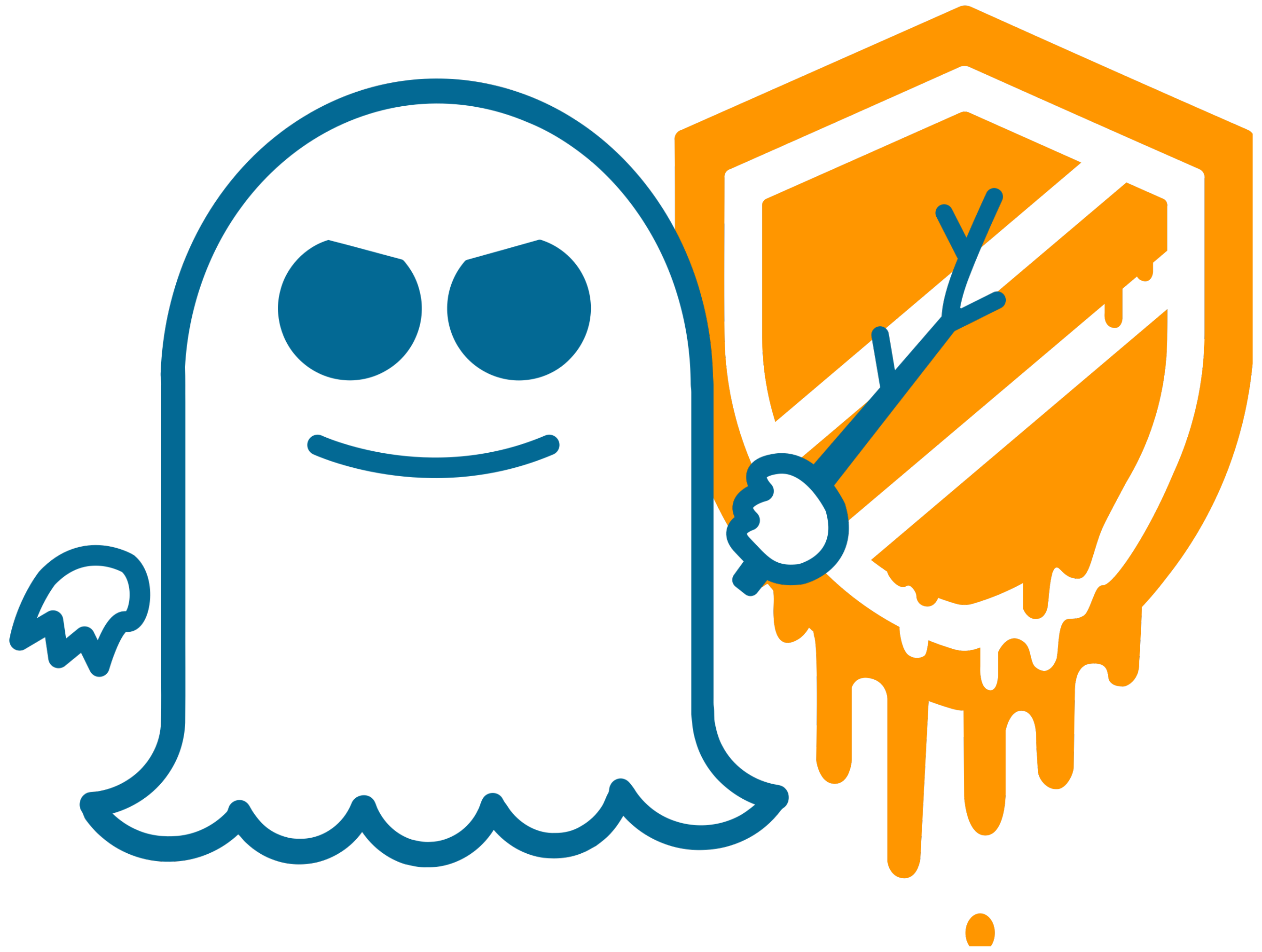
Why do We Care about CPU Security?

- **CPU vulnerabilities can leak or spy on...**
  - ... cryptographic keys.



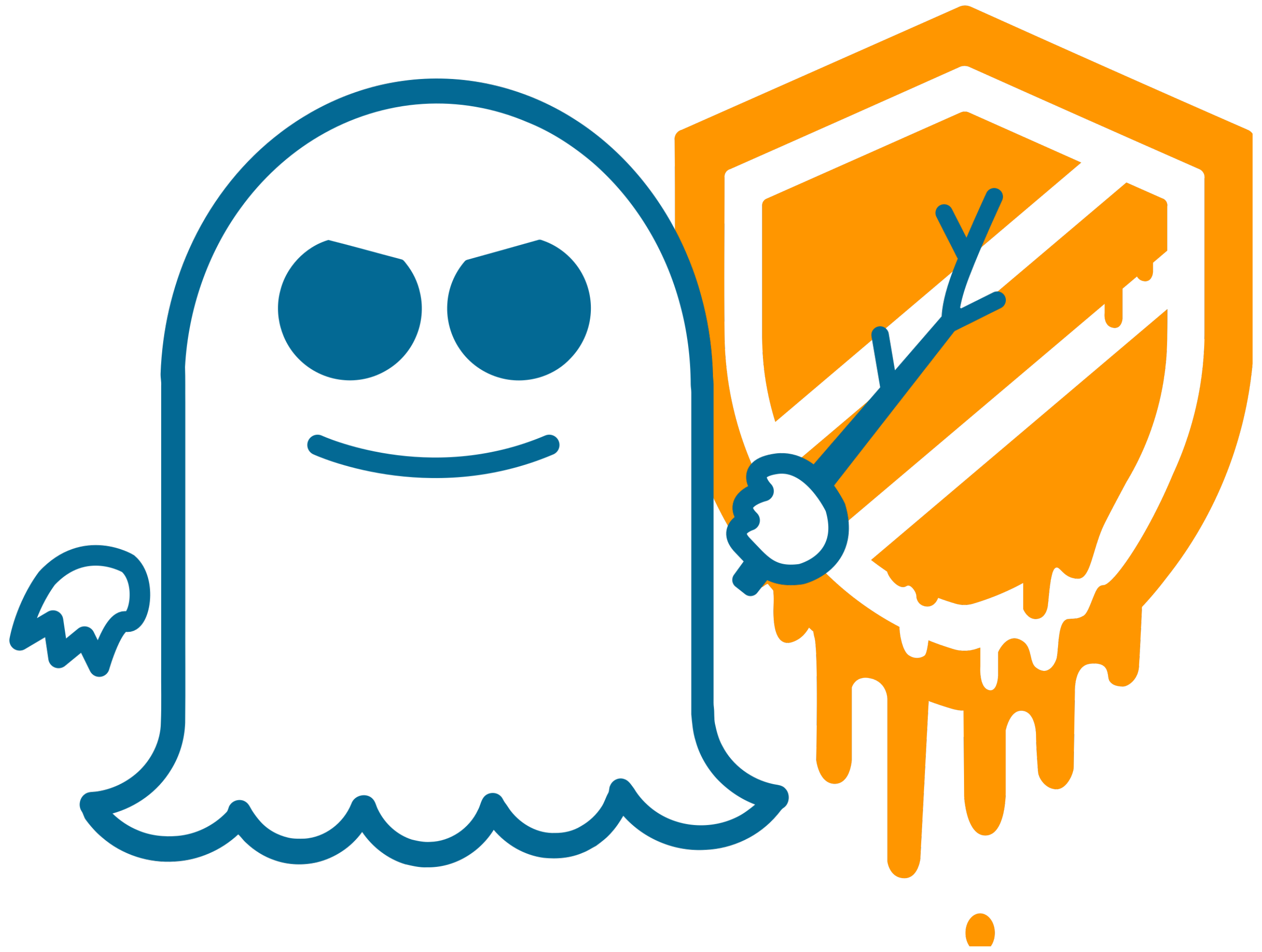
Why do We Care about CPU Security?

- **CPU vulnerabilities can leak or spy on...**
  - ... cryptographic keys.
  - ... browser cookies.



Why do We Care about CPU Security?

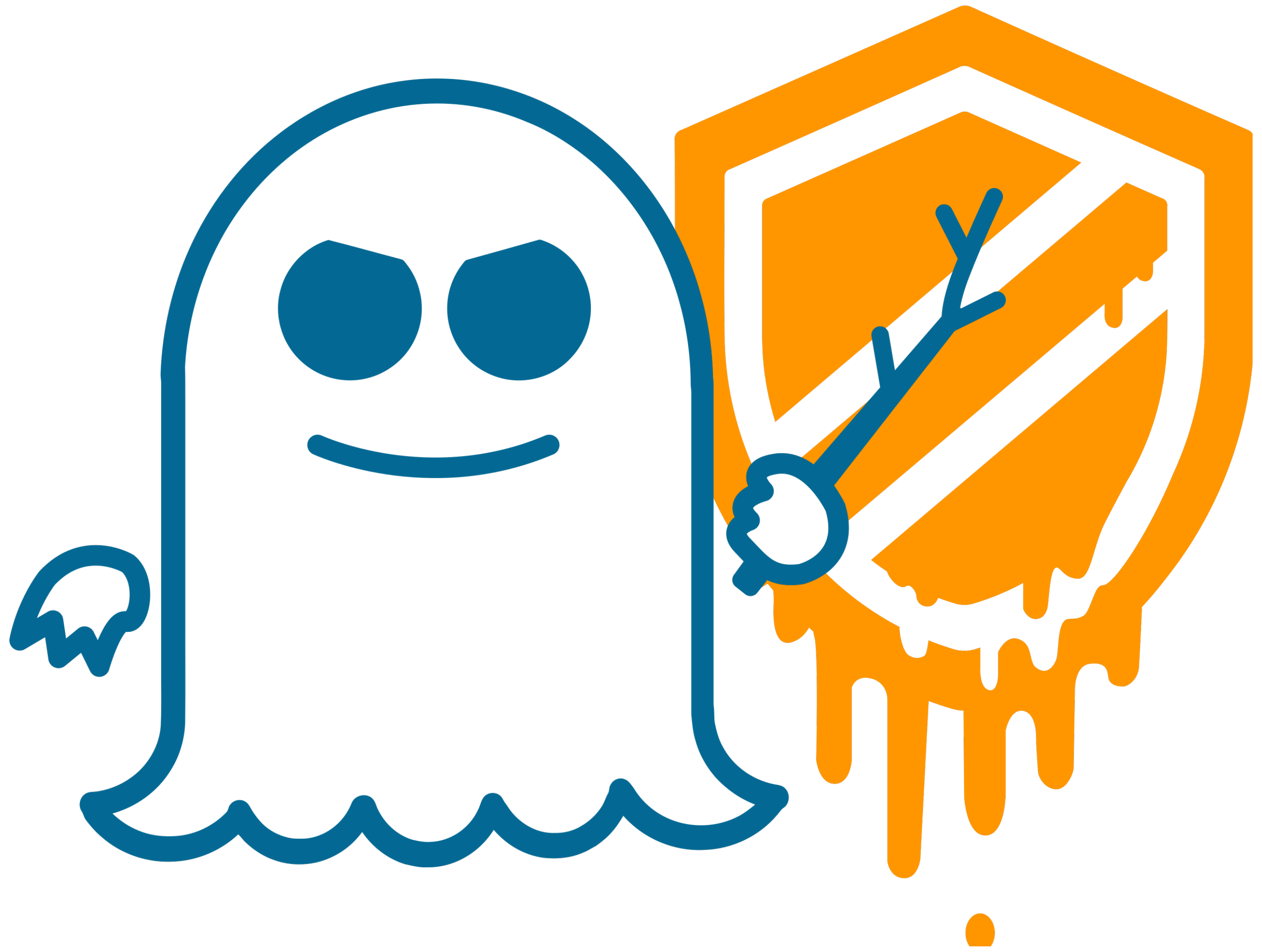
- **CPU vulnerabilities can leak or spy on...**
  - ... cryptographic keys.
  - ... browser cookies.
  - ... (almost) arbitrary memory.



Why do We Care about CPU Security?

- **CPU vulnerabilities can leak or spy on...**
  - ... cryptographic keys.
  - ... browser cookies.
  - ... (almost) arbitrary memory.

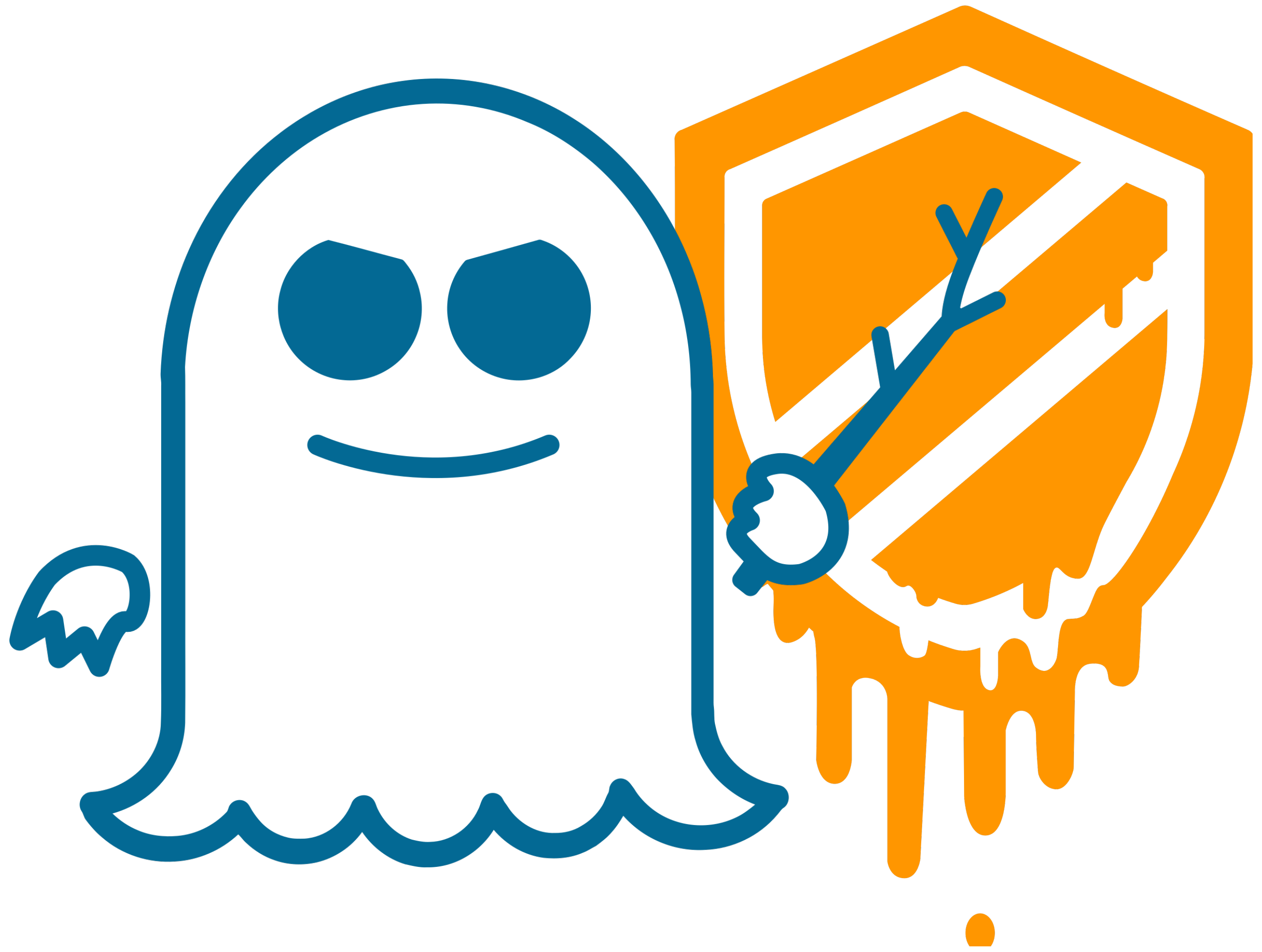
- **CPU vulnerabilities can attack...**



Why do We Care about CPU Security?

- **CPU vulnerabilities can leak or spy on...**
  - ... cryptographic keys.
  - ... browser cookies.
  - ... (almost) arbitrary memory.

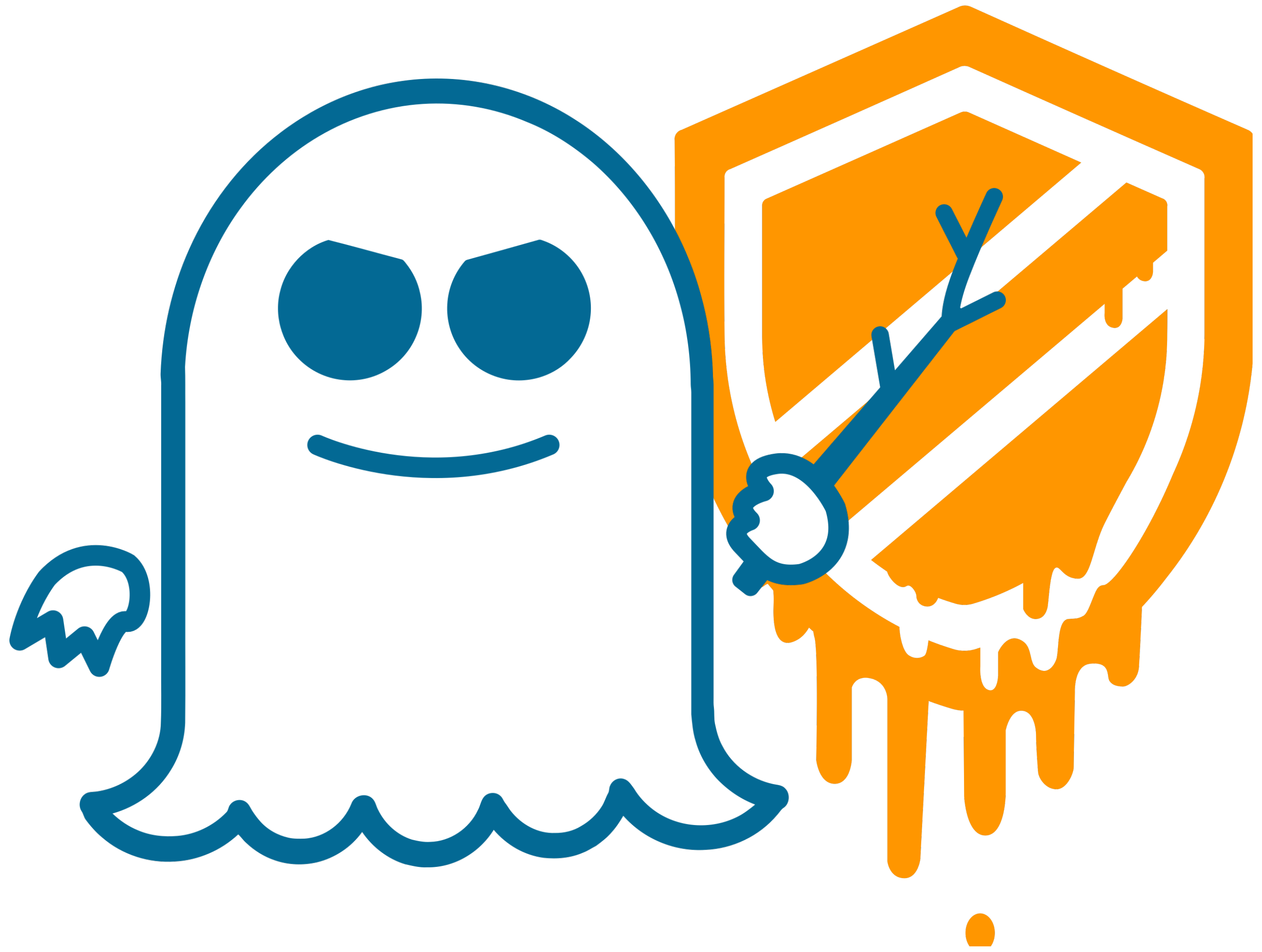
- **CPU vulnerabilities can attack...**
  - ... the kernel.



Why do We Care about CPU Security?

- **CPU vulnerabilities can leak or spy on...**
  - ... cryptographic keys.
  - ... browser cookies.
  - ... (almost) arbitrary memory.

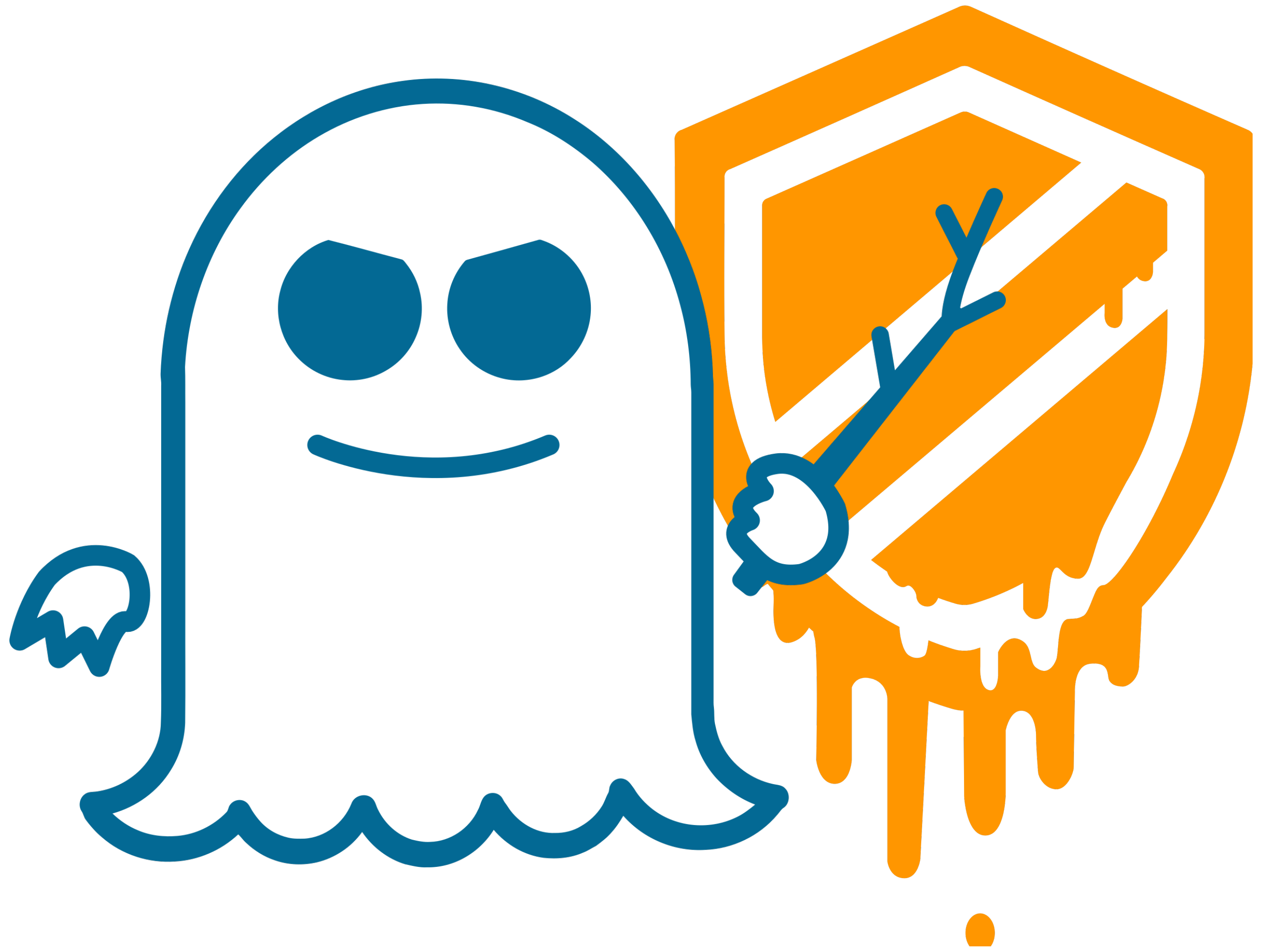
- **CPU vulnerabilities can attack...**
  - ... the kernel.
  - ... the browser (from within JavaScript).



Why do We Care about CPU Security?

- **CPU vulnerabilities can leak or spy on...**
  - ... cryptographic keys.
  - ... browser cookies.
  - ... (almost) arbitrary memory.

- **CPU vulnerabilities can attack...**
  - ... the kernel.
  - ... the browser (from within JavaScript).
  - ... virtual machines.



Why do We Care about CPU security?

Why do We Care about CPU security?

Why do We Care about CPU security?

Why do We Care about CPU security?

Why do We Care about CPU security?

MDS
PLUNDER
VOLT

Why do We Care about CPU security?

MDS
PLUNDER
VOLT

Why do We Care about CPU security?

Why do We Care about CPU security?

Why do We Care about CPU security?

MD
S
PLUNDER
VOLT
ÆPIC
LEAK
iLeakage

Why do We Care about CPU security?

MDS
PLUNDER
VOLT
ÆPIC
LEAK
iLeakage

Why do We Care about CPU security?

MDS
PLUNDER VOLT
ÆPIC LEAK
iLeakage
CACHE WARP

Different Vulnerabilities, Different Attacks

Different Vulnerabilities, Different Attacks

- **Leaking Secrets:**

Different Vulnerabilities, Different Attacks

- **Leaking Secrets:**
  - Spectre

Different Vulnerabilities, Different Attacks

- **Leaking Secrets:**
  - Spectre
  - Meltdown

Different Vulnerabilities, Different Attacks

- **Leaking Secrets:**
  - Spectre
  - Meltdown
  - MDS
  - ...

Different Vulnerabilities, Different Attacks

- **Leaking Secrets:**
  - Spectre
  - Meltdown
  - MDS
  - ...

- **Tampering with Data:**

Different Vulnerabilities, Different Attacks

- **Leaking Secrets:**
  - Spectre
  - Meltdown
  - MDS
  - ...

- **Tampering with Data:**
  - PlunderVolt

Different Vulnerabilities, Different Attacks

- **Leaking Secrets:**
  - Spectre
  - Meltdown
  - MDS
  - ...
- **Tampering with Data:**
  - PlunderVolt
  - CacheWarp
  - ...

Only Intel and AMD CPUs?

Only Intel and AMD CPUs?
What about the Others?

Other Architectures? ARM?

Other Architectures? ARM?

- **Demonstrated attacks** include:

Other Architectures? ARM?

- **Demonstrated attacks** include:
  - **Cache** attacks

Other Architectures? ARM?

- **Demonstrated attacks** include:
  - **Cache** attacks
  - **Spectre**

Other Architectures? ARM?

- **Demonstrated attacks** include:
  - **Cache** attacks
  - **Spectre**
  - Some **Meltdown** Variants

Other Architectures? ARM?

- **Demonstrated attacks** include:
  - **Cache** attacks
  - **Spectre**
  - Some **Meltdown** Variants
  - ...

Other Architectures? ARM?

- **Demonstrated attacks** include:
  - **Cache** attacks
  - **Spectre**
  - Some **Meltdown** Variants
  - ...

→ **ARM is also vulnerable**

There is another!

RISC-V: The New Star on the Horizon

RISC-V: The New Star on the Horizon

- New **Instruction Set Architecture** (ISA)

RISC-V: The New Star on the Horizon

- New **Instruction Set Architecture** (ISA)
- **Open-Source** Standard

RISC-V: The New Star on the Horizon

- New **Instruction Set Architecture** (ISA)

- **Open-Source** Standard
  - White-box bug hunting

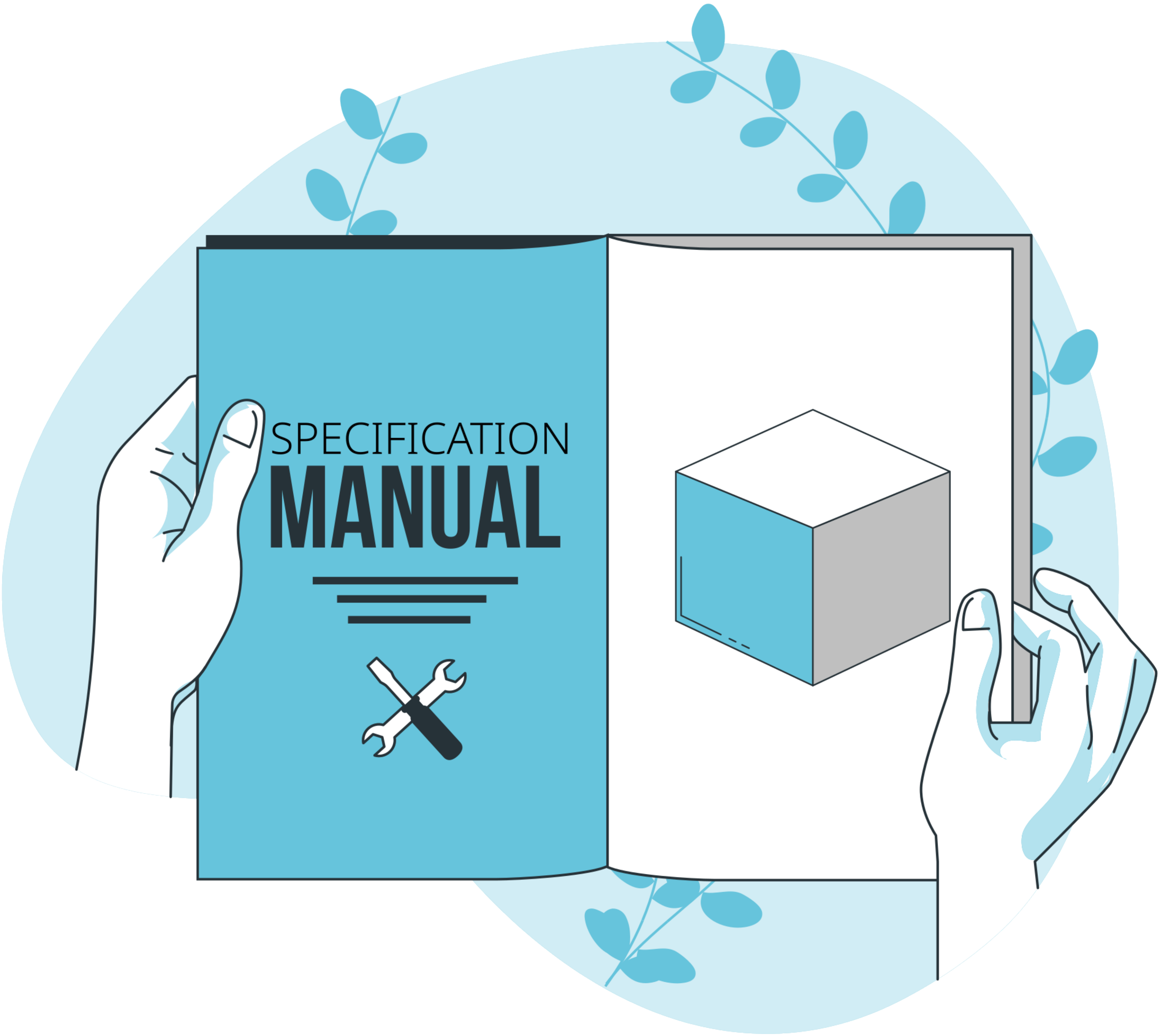
RISC-V: The New Star on the Horizon

- New **Instruction Set Architecture** (ISA)

- **Open-Source** Standard
  - White-box bug hunting
  - Testing of hardware mitigations

RISC-V: The New Star on the Horizon

- New **Instruction Set Architecture** (ISA)
- **Open-Source** Standard
  - White-box bug hunting
  - Testing of hardware mitigations
- Lots of **academic research**



Is RISC-V just another Academia Thingy?

You Can Buy RISC-V Cores!

You Can Buy RISC-V Cores!

RISC-V®

Now available in Hardware

You Can Buy RISC-V Cores!

Allwinner D1 (C906)

Now available in Hardware

You Can Buy RISC-V Cores!

Allwinner D1 (C906)

Now available in Hardware

SiFive U74

RISC-V is Coming

RISC-V is Coming

***62.4 billion** RISC-V cores predicted to be running*
by 2025

RISC-V is Coming

62.4 billion *RISC-V cores predicted to be running* ***by 2025***

*RISE Project by major vendors (**Google**, **Qualcomm**, **Samsung**, …)*

But Security?

But Security?

Security?

But Security?

Security?

What is the ***status quo*** *on*

hardware RISC-V processors*?*

Let’s Investigate RISC-V...

Let’s Investigate RISC-V...

- **Did we learn from the past**?

Let’s Investigate RISC-V…

- **Did we learn from the past**?
  - **Are these CPUs hardened** against known attacks?

Let's Investigate RISC-V...

- **Did we learn from the past**?
  - **Are these CPUs hardened** against known attacks?
  - **Do academic mitigations make it** into production CPUs?

Let’s Investigate RISC-V...

- **Did we learn from the past**?
  - **Are these CPUs hardened** against known attacks?
  - **Do academic mitigations make it** into production CPUs?

- Does **open-source imply better or worse** security?

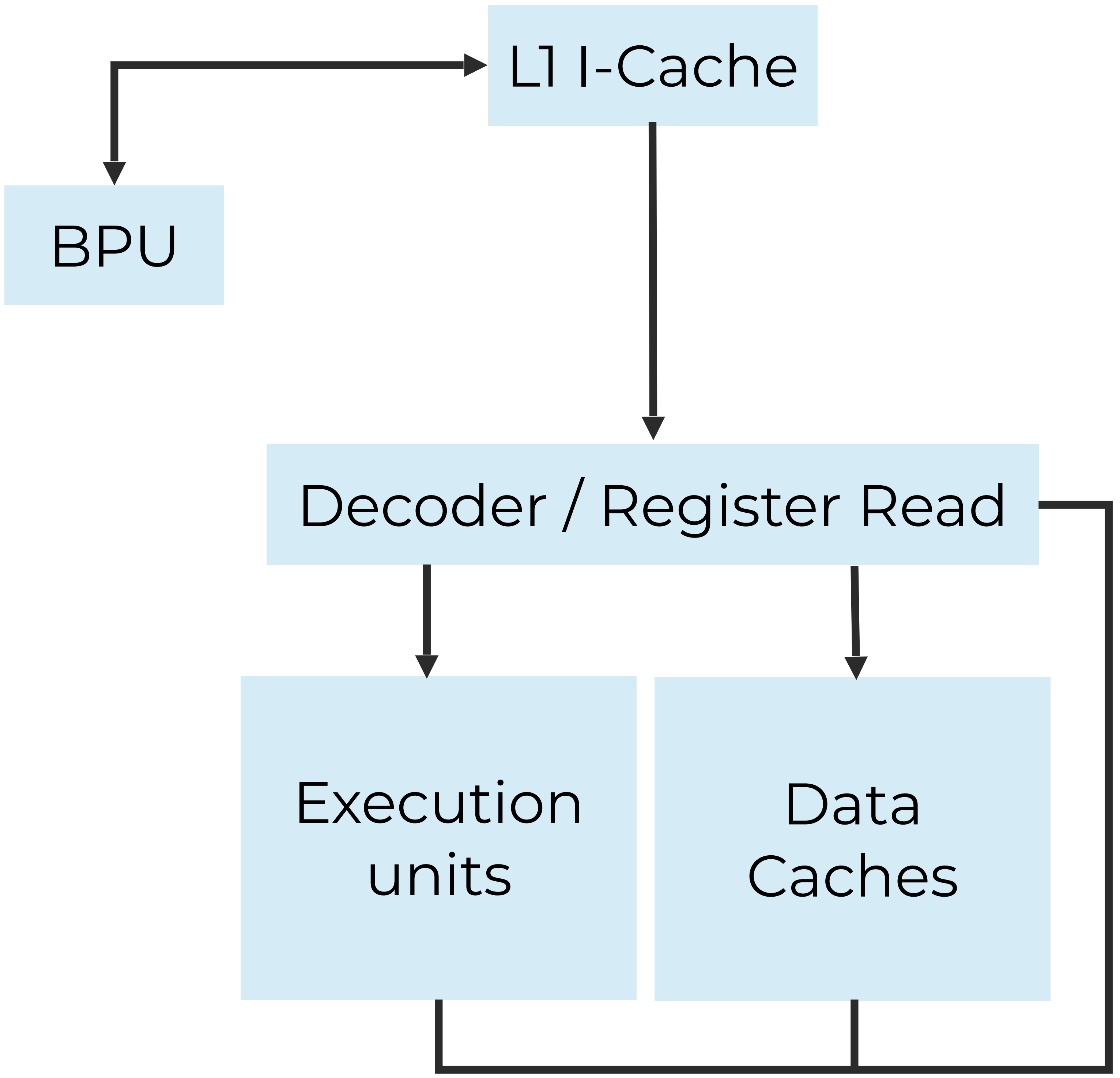
Let’s Investigate RISC-V...

- **Did we learn from the past**?
  - **Are these CPUs hardened** against known attacks?
  - **Do academic mitigations make it** into production CPUs?

- Does **open-source imply better or worse** security?

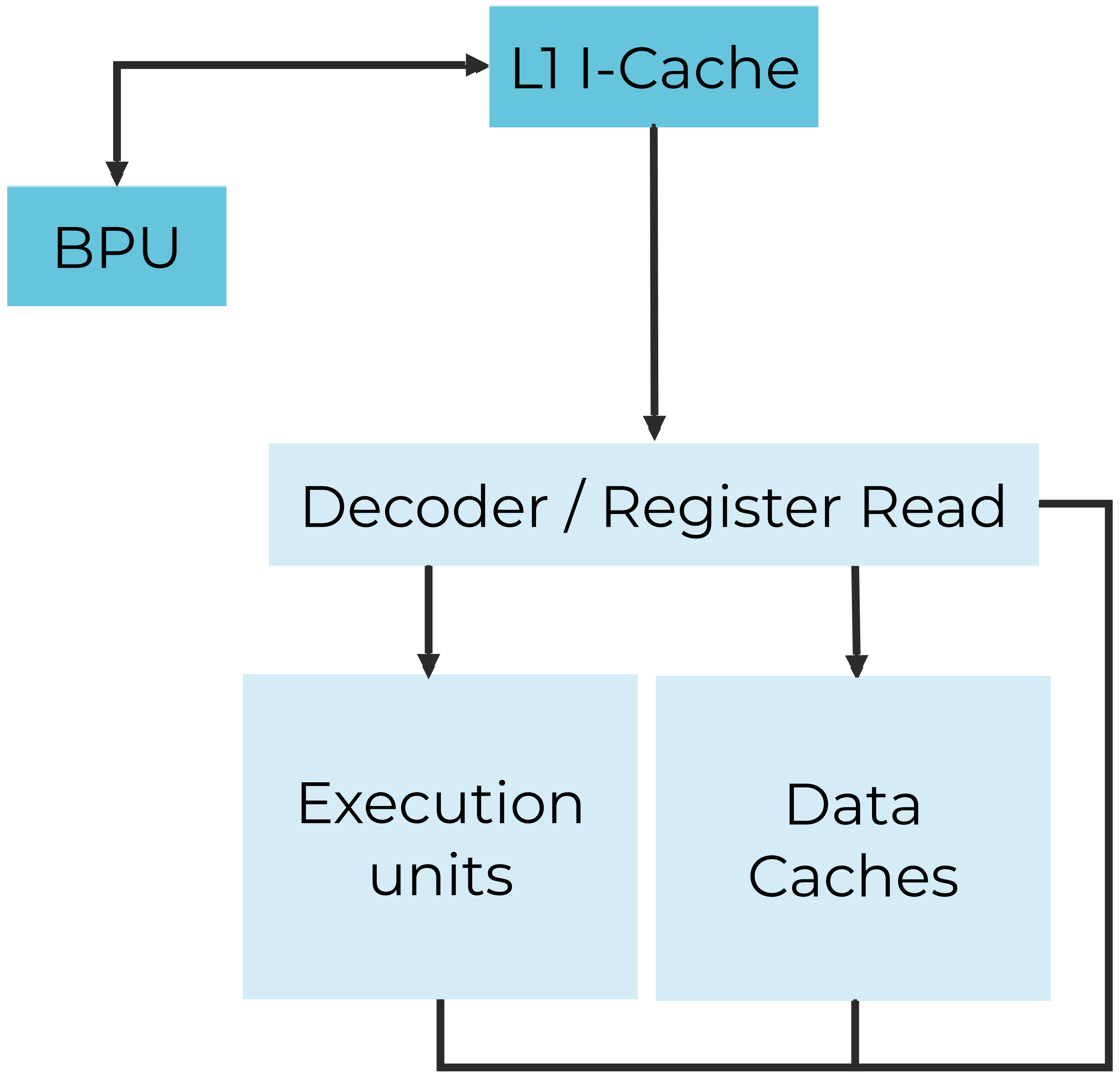
→ **Let’s analyze the C906 and U74!**

How Does a CPU Work?



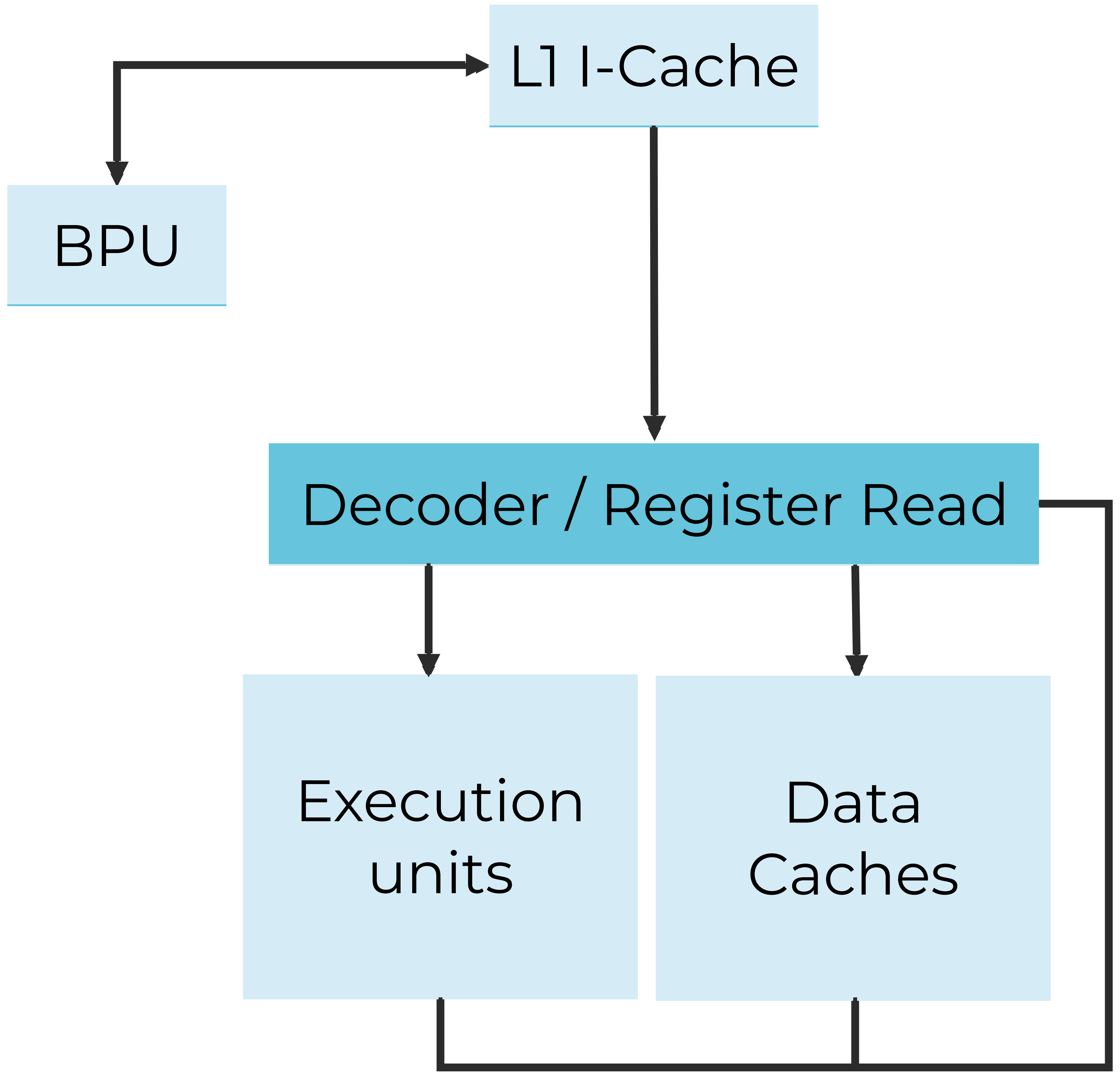
How Does a CPU Work?

1. Fetch instruction from memory



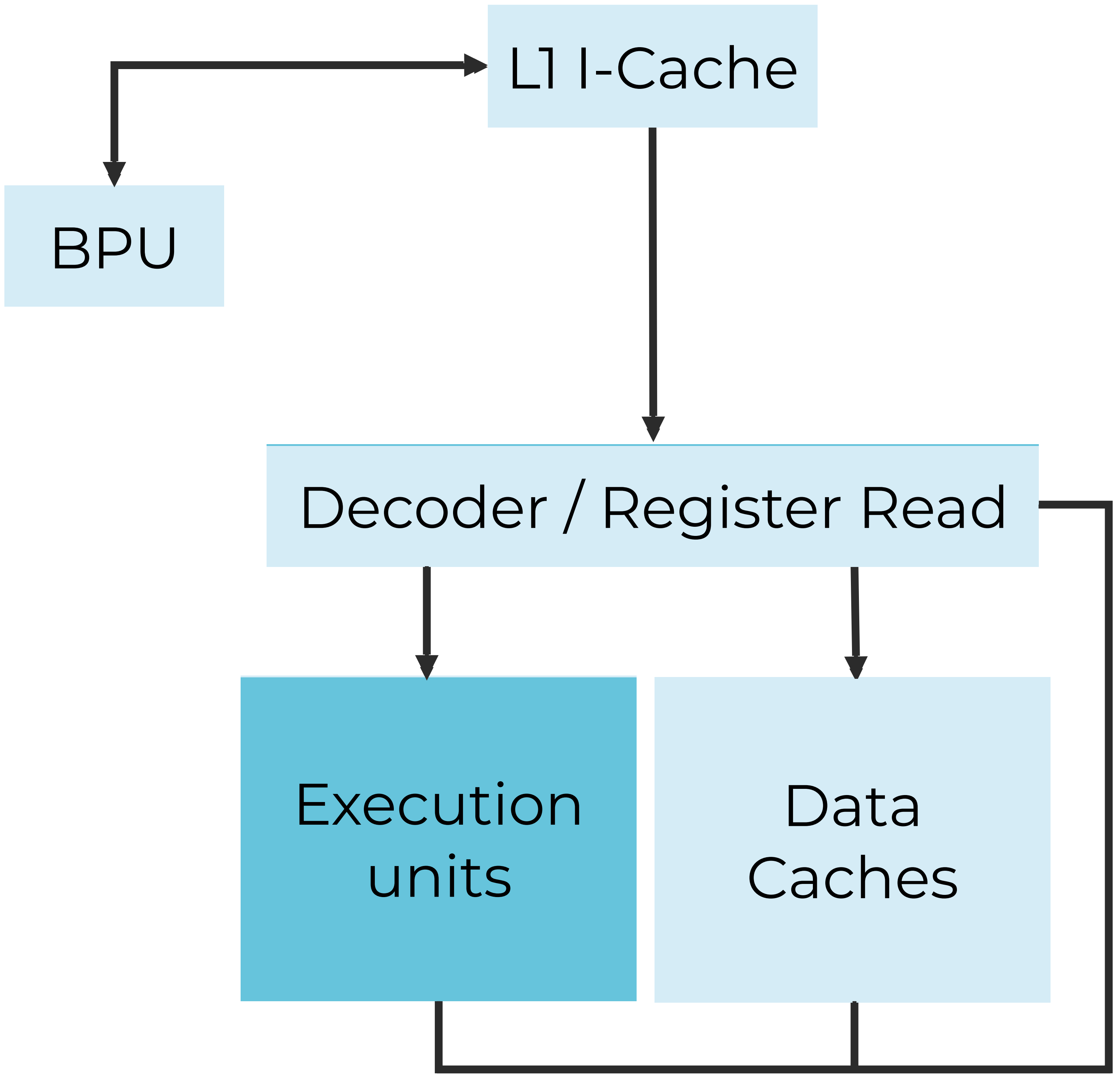
How Does a CPU Work?

1. Fetch instruction from memory
2. Decode instruction and decide what to do



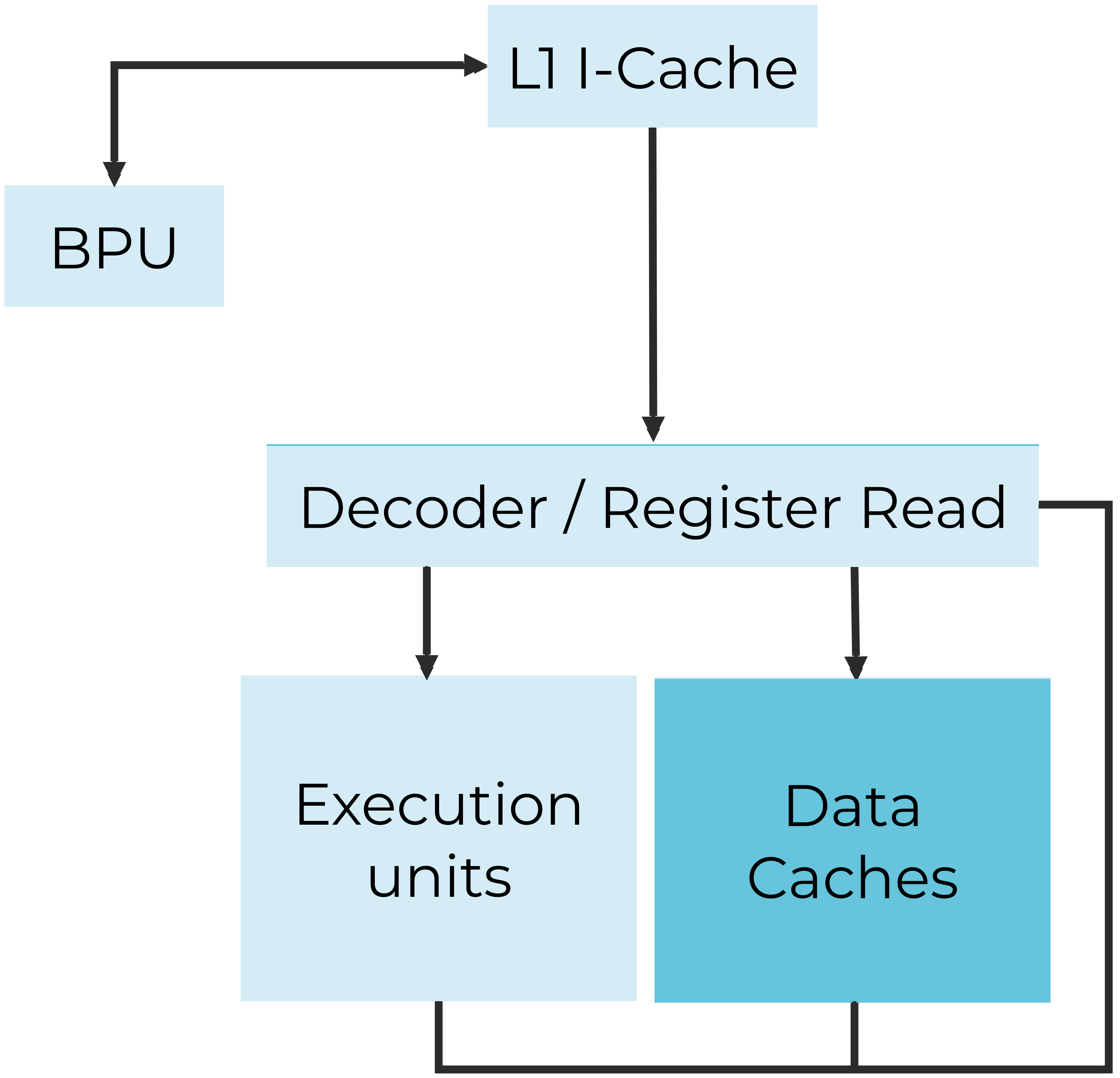
How Does a CPU Work?

1. Fetch instruction from memory
2. Decode instruction and decide what to do
3. Execute the instruction

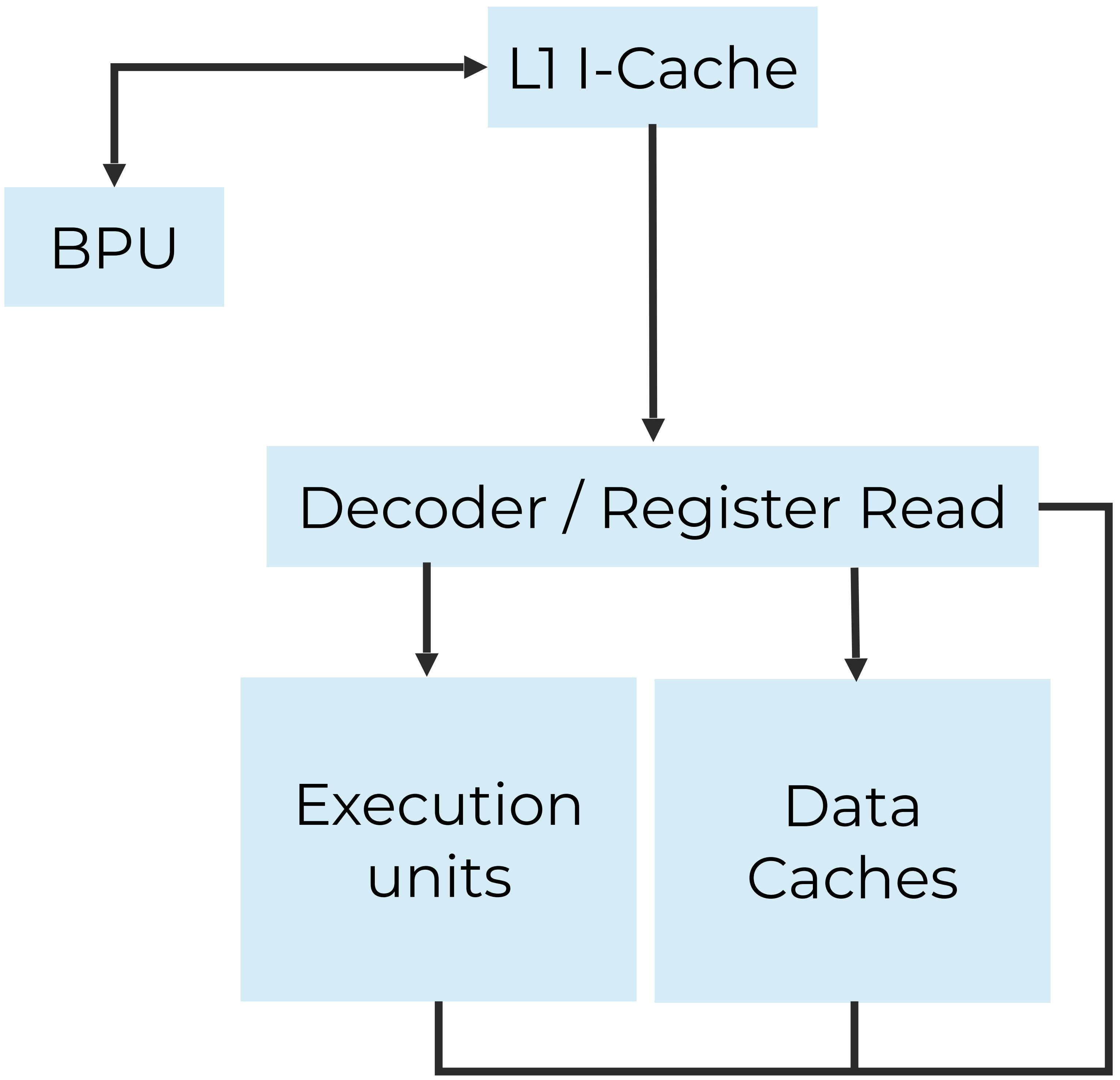


How Does a CPU Work?

1. Fetch instruction from memory
2. Decode instruction and decide what to do
3. Execute the instruction
4. Write back the results to memory

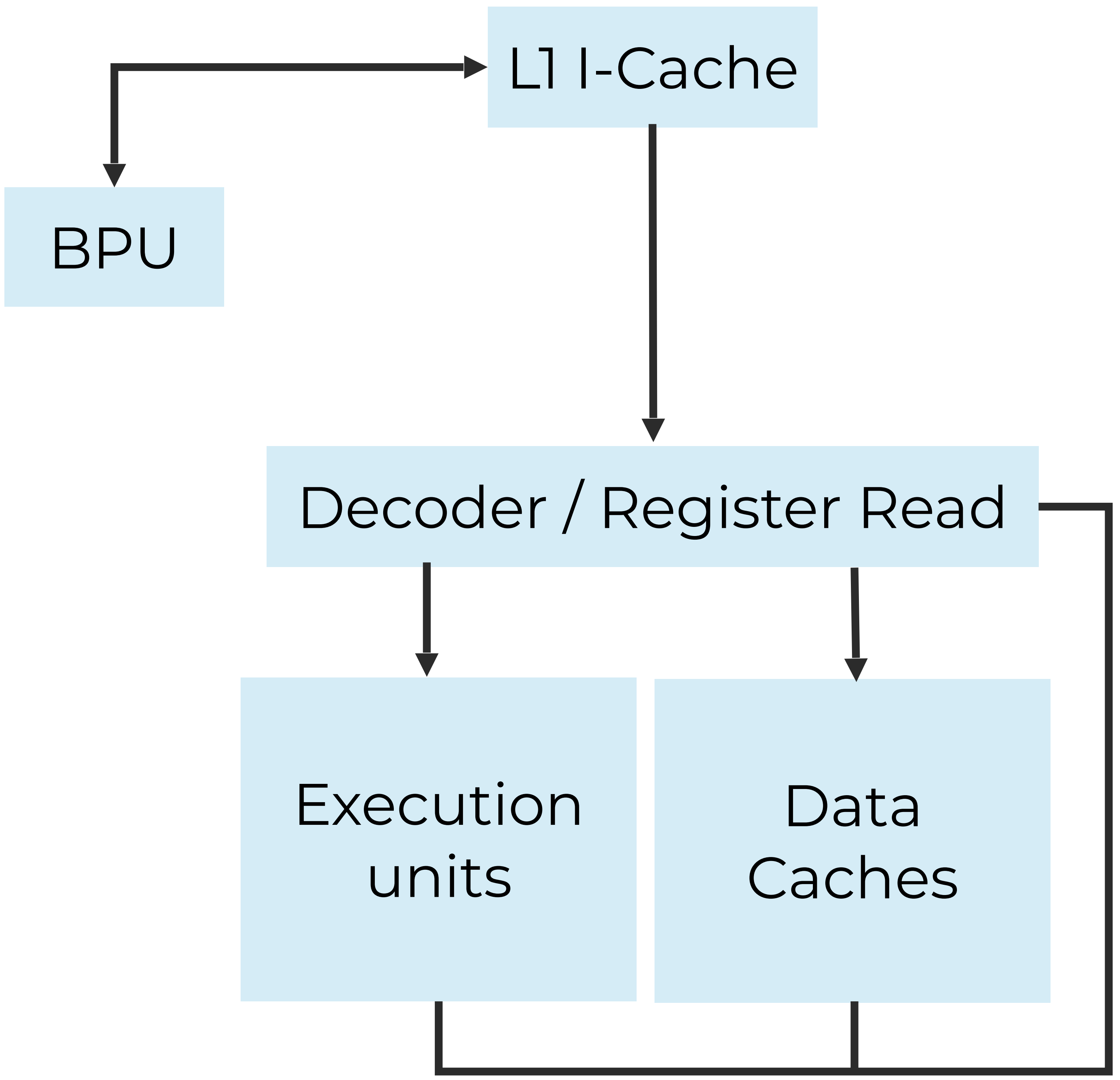


What Type of Attacks are There?



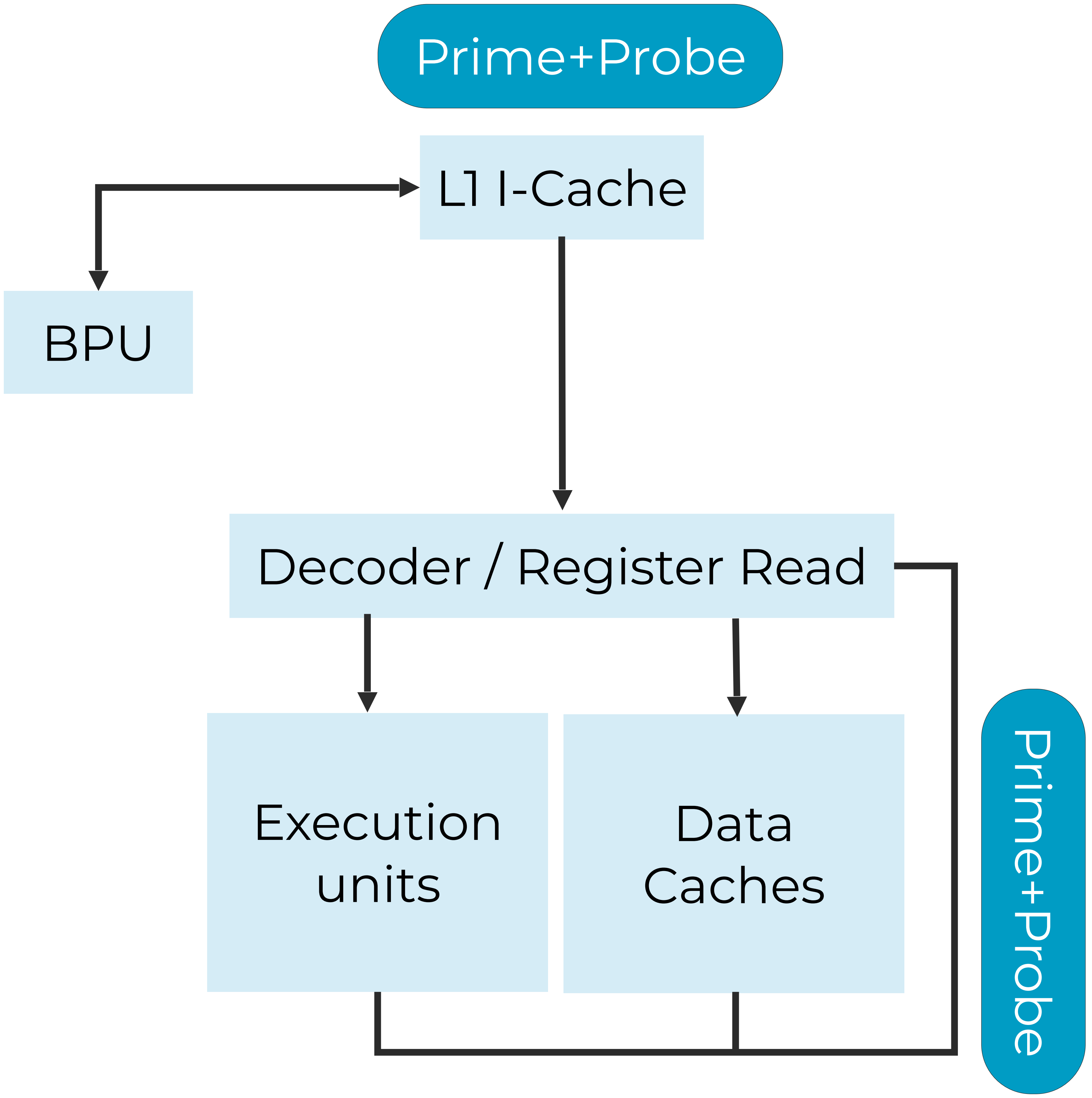
What Type of Attacks are There?

- Attack **timing differences** in caches and predictors



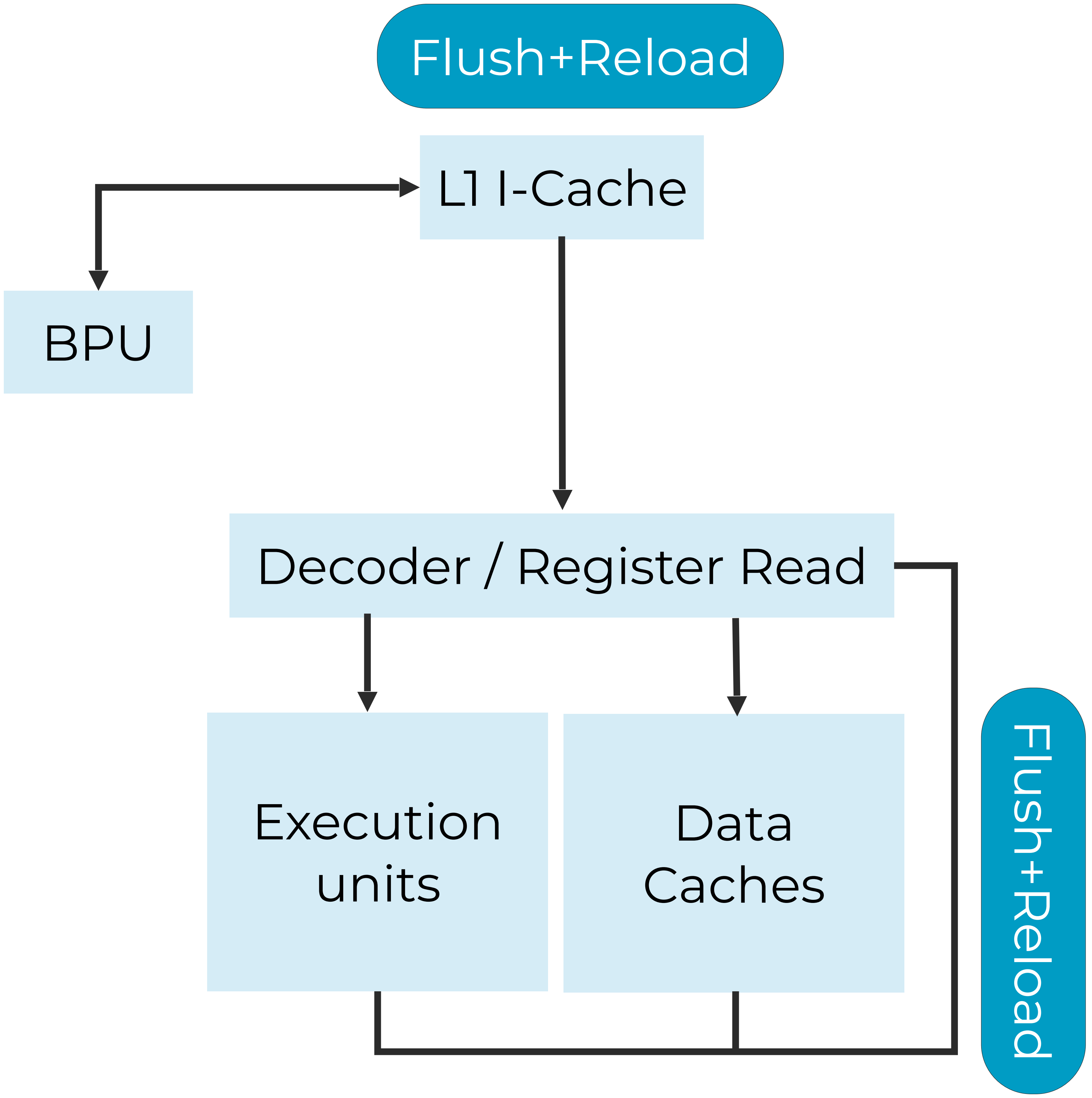
What Type of Attacks are There?

- Attack **timing differences** in caches and predictors
  - Flush+Reload, Prime+Probe



What Type of Attacks are There?

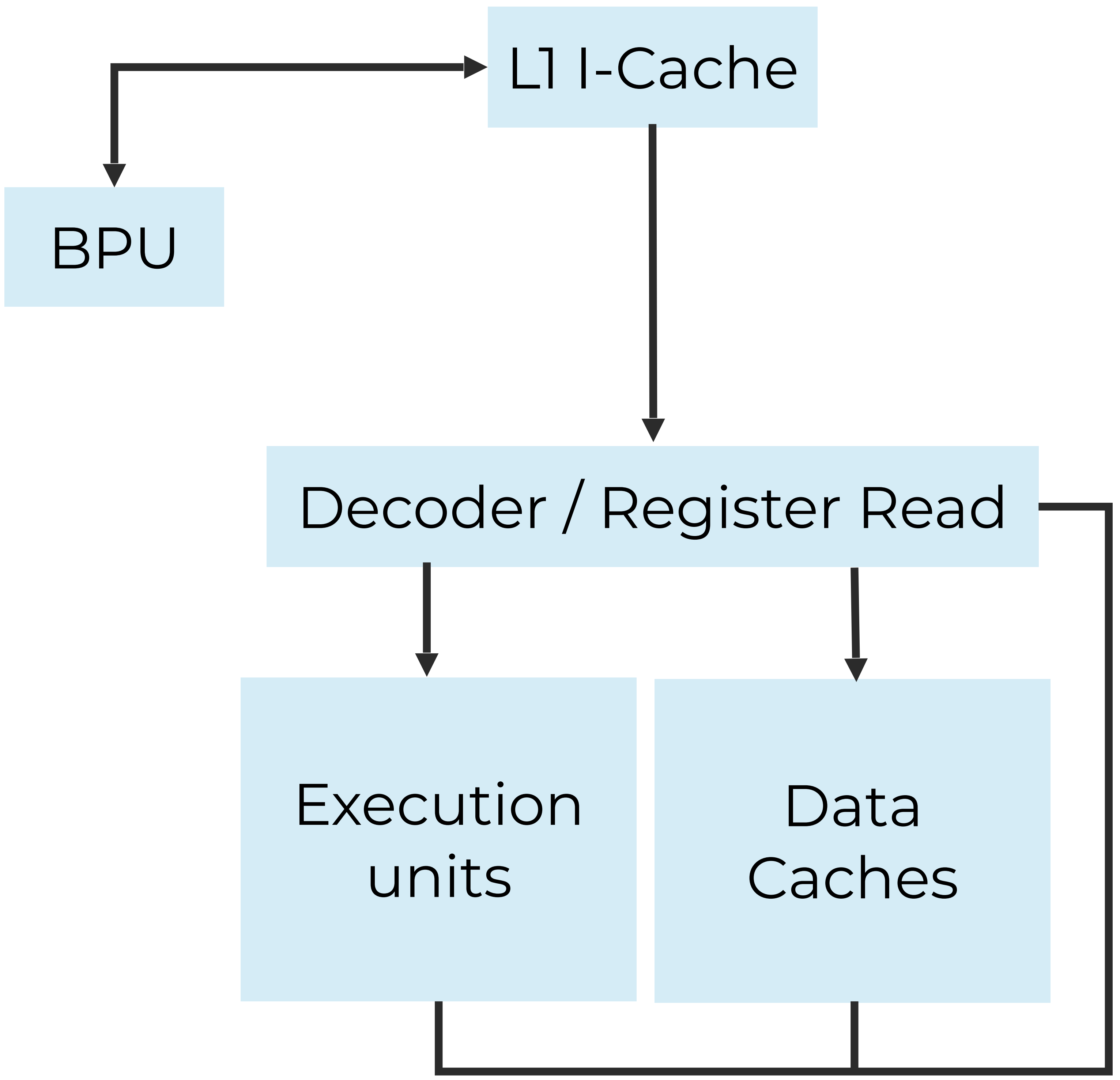
- Attack **timing differences** in caches and predictors
  - Flush+Reload, Prime+Probe



What Type of Attacks are There?

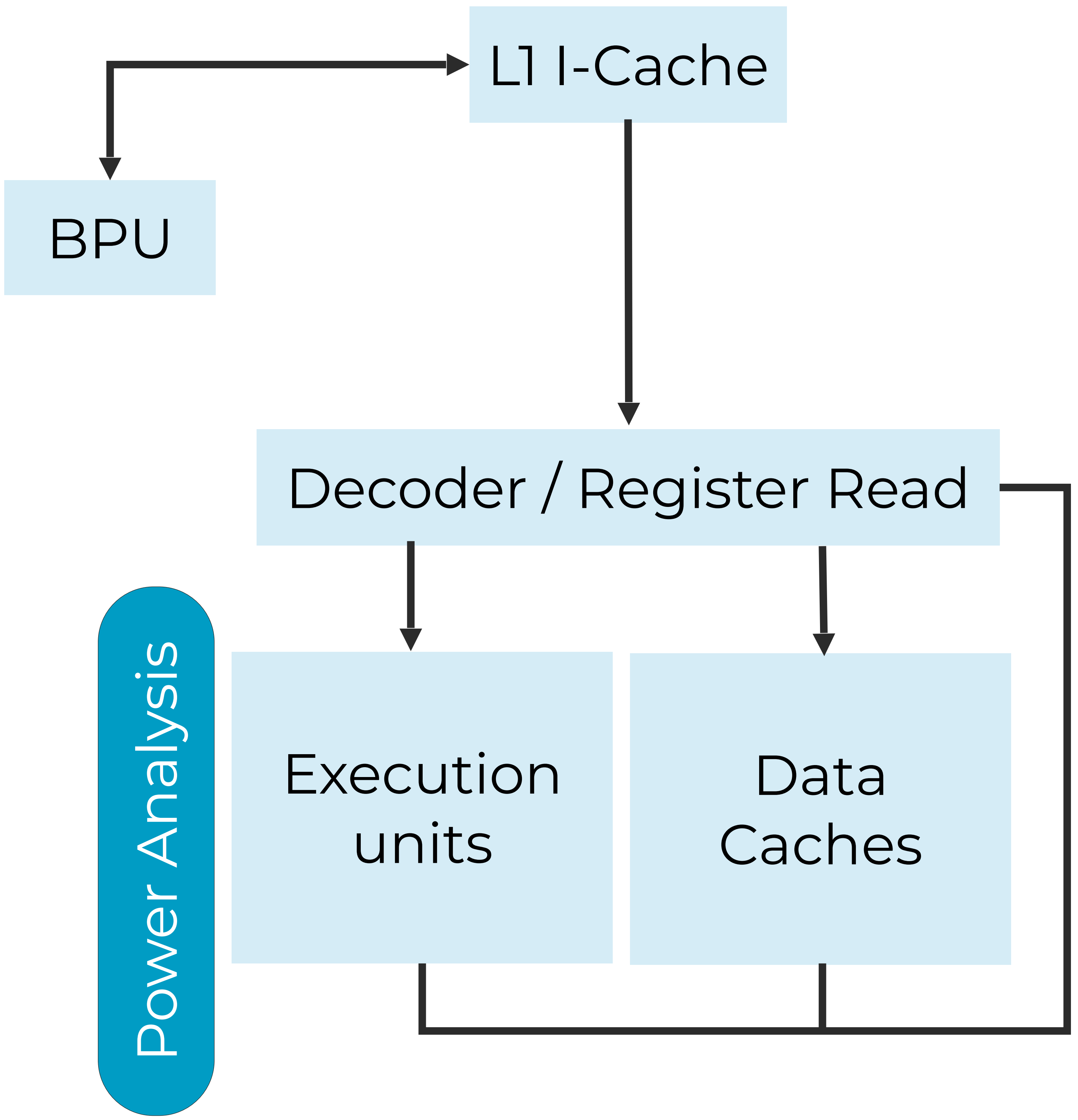
- Attack **timing differences** in caches and predictors
  - Flush+Reload, Prime+Probe

- Exploiting **implementation bugs**



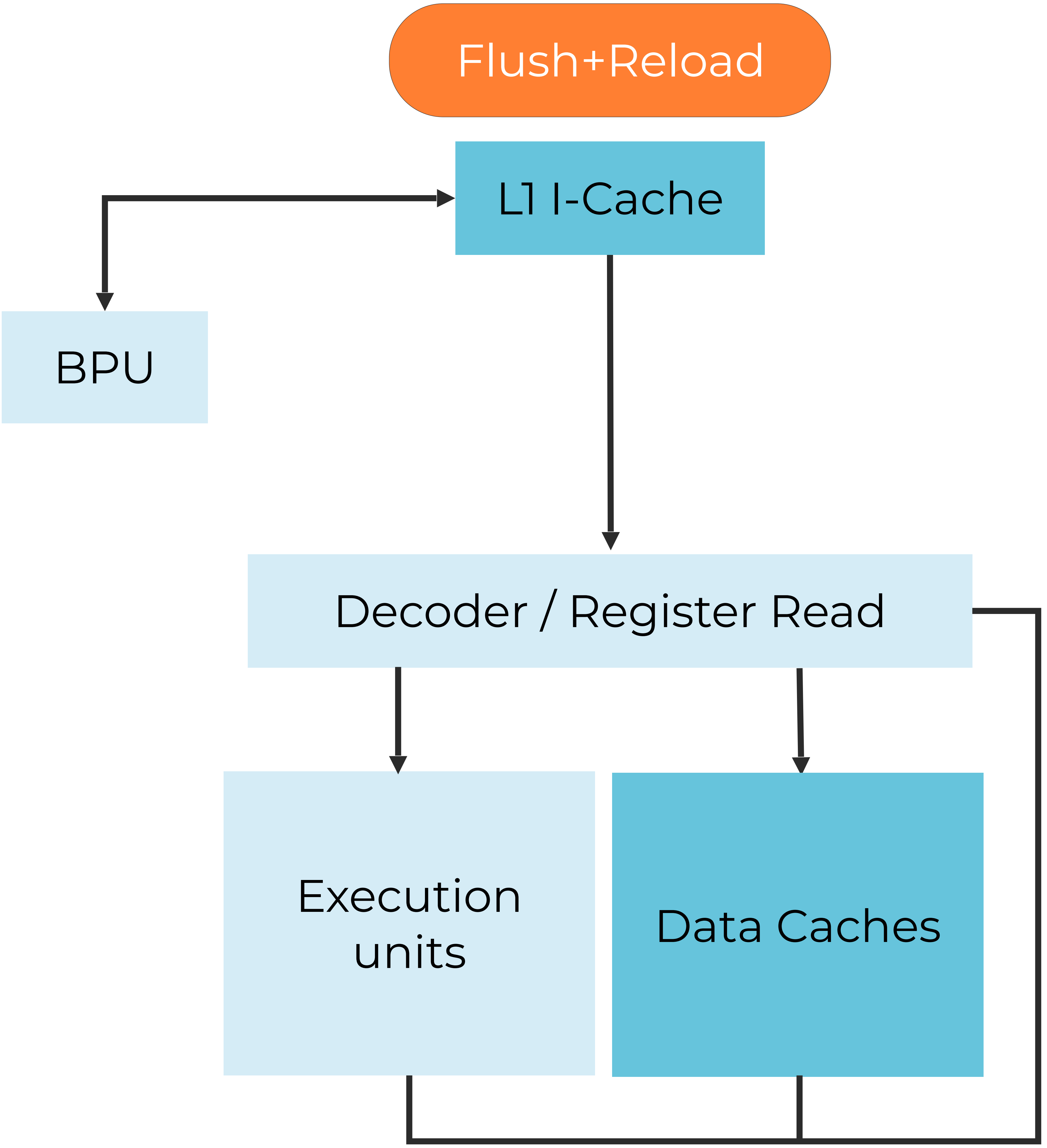
What Type of Attacks are There?

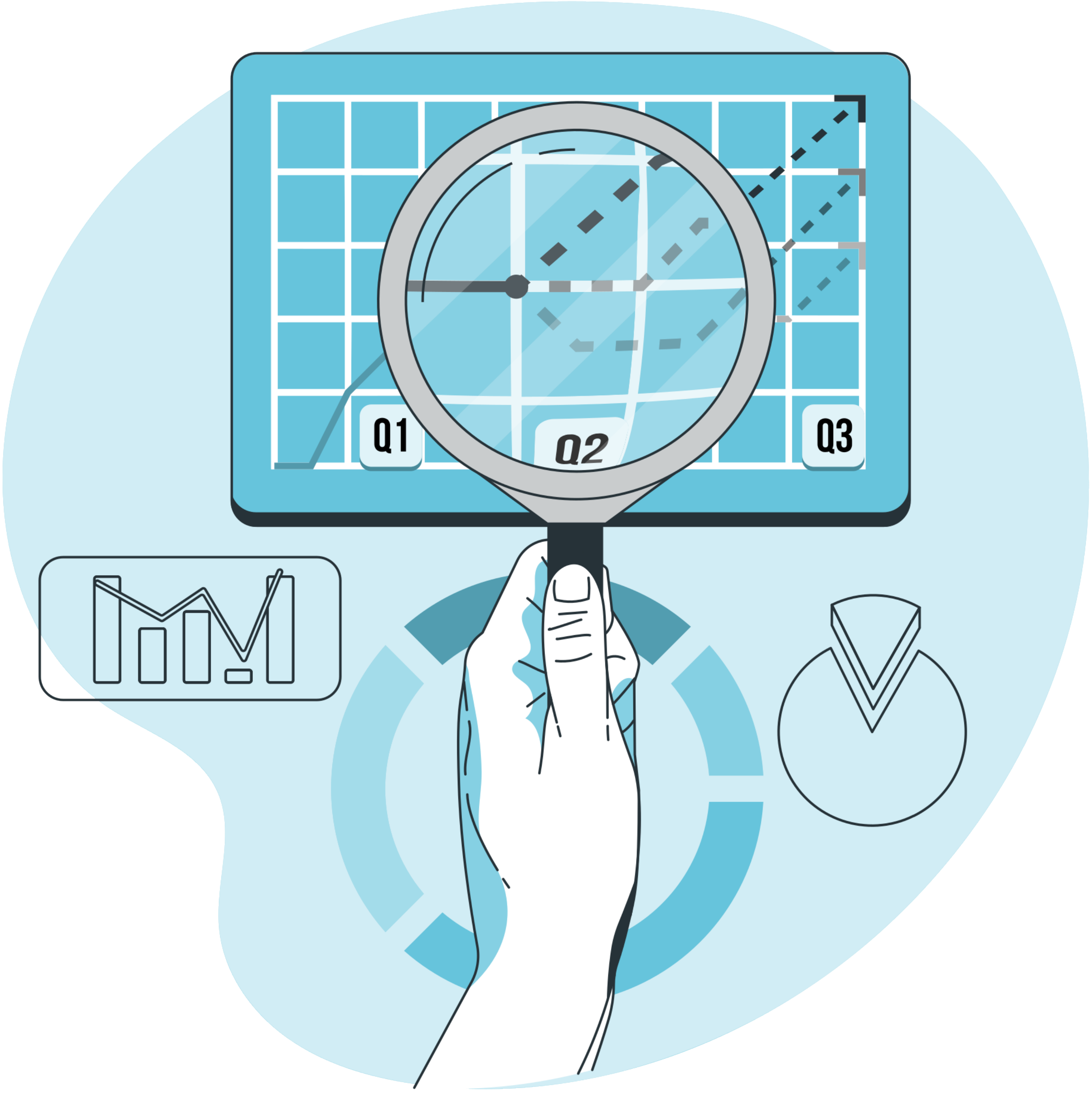
- Attack **timing differences** in caches and predictors
  - Flush+Reload, Prime+Probe
- Exploiting **implementation bugs**
- Abusing **physical properties**
  - Rowhammer
  - Power Analysis



Let’s start with our first attack!

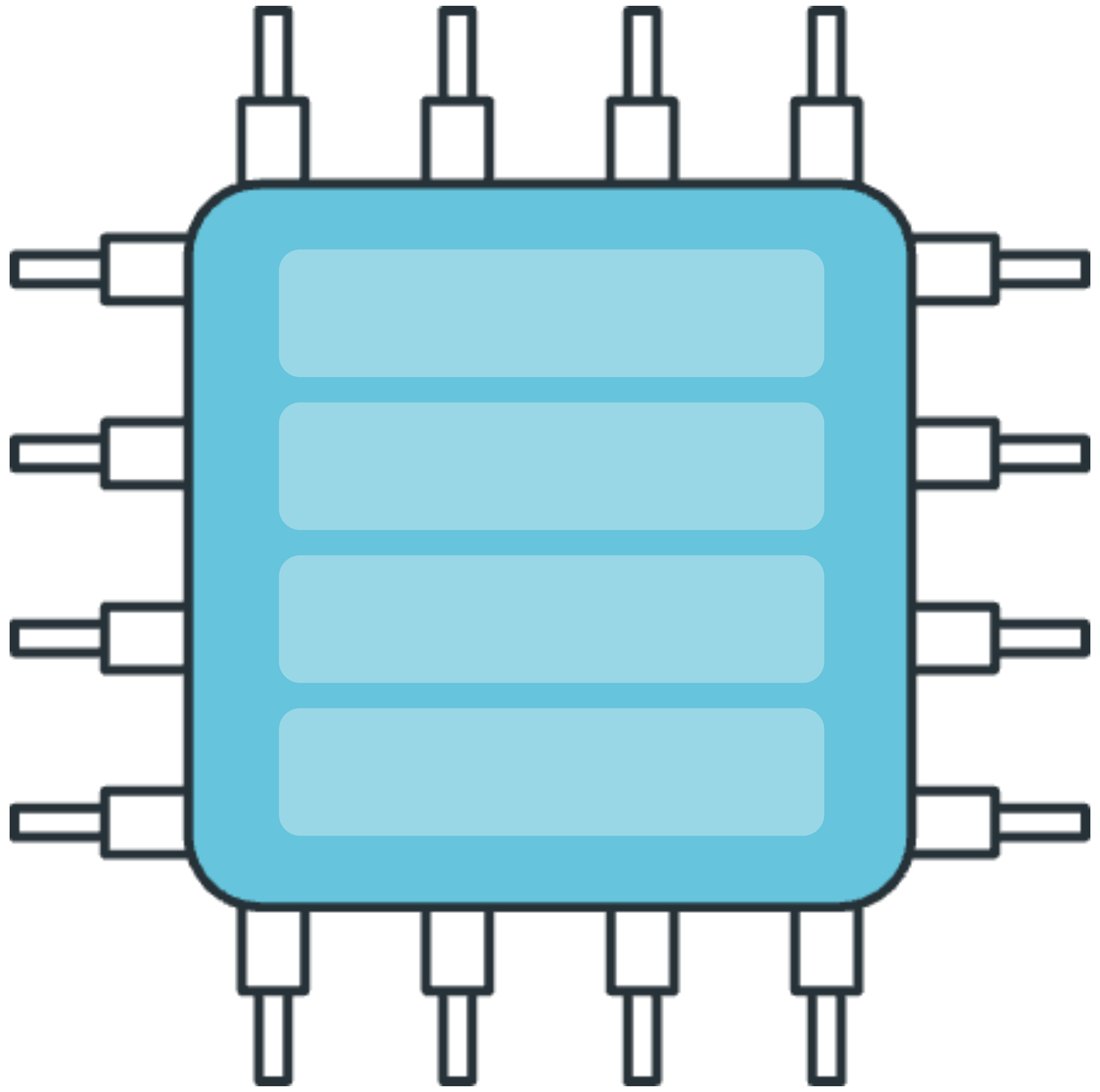
CPU Design: Flush+Reload



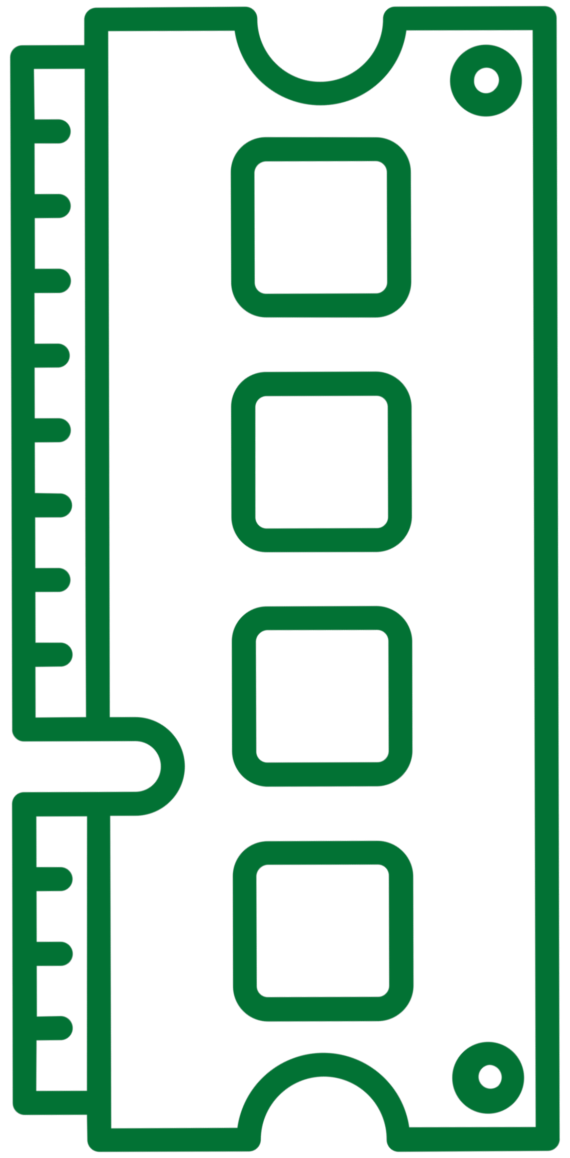


CPU Optimization: The Cache

```
access(array[0]);

access(array[0]);
```



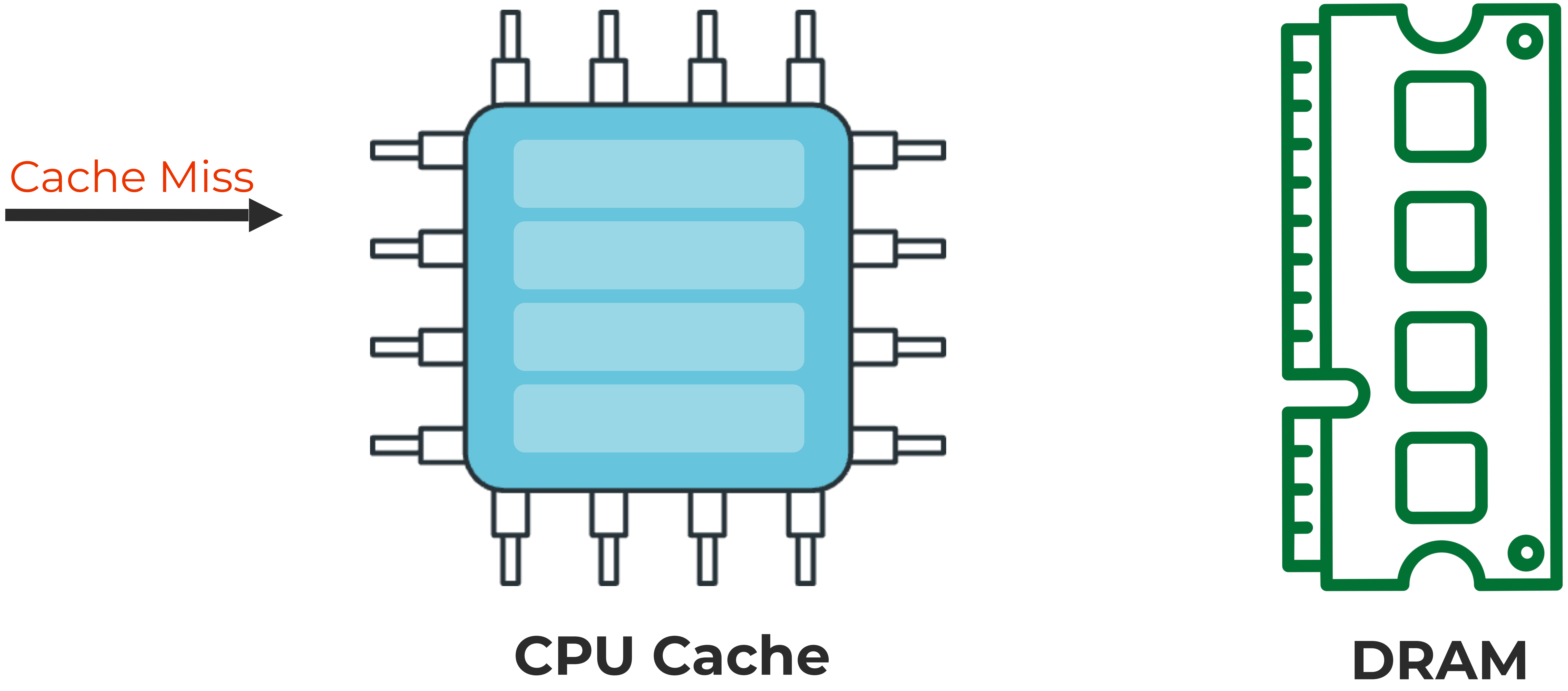
CPU Cache



DRAM

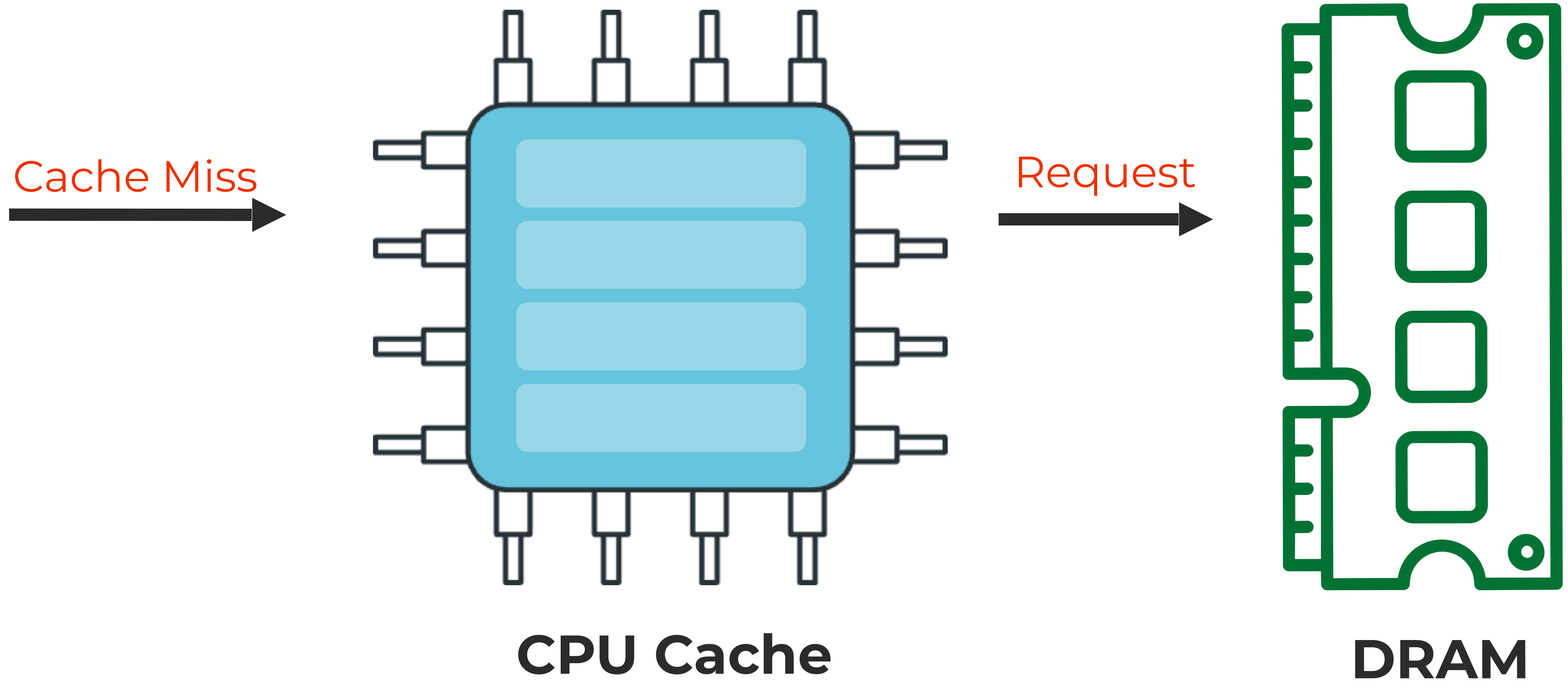
CPU Optimization: The Cache

```
access(array[0]);

access(array[0]);
```



CPU Optimization: The Cache

```
access(array[0]);

access(array[0]);
```



Cache Miss

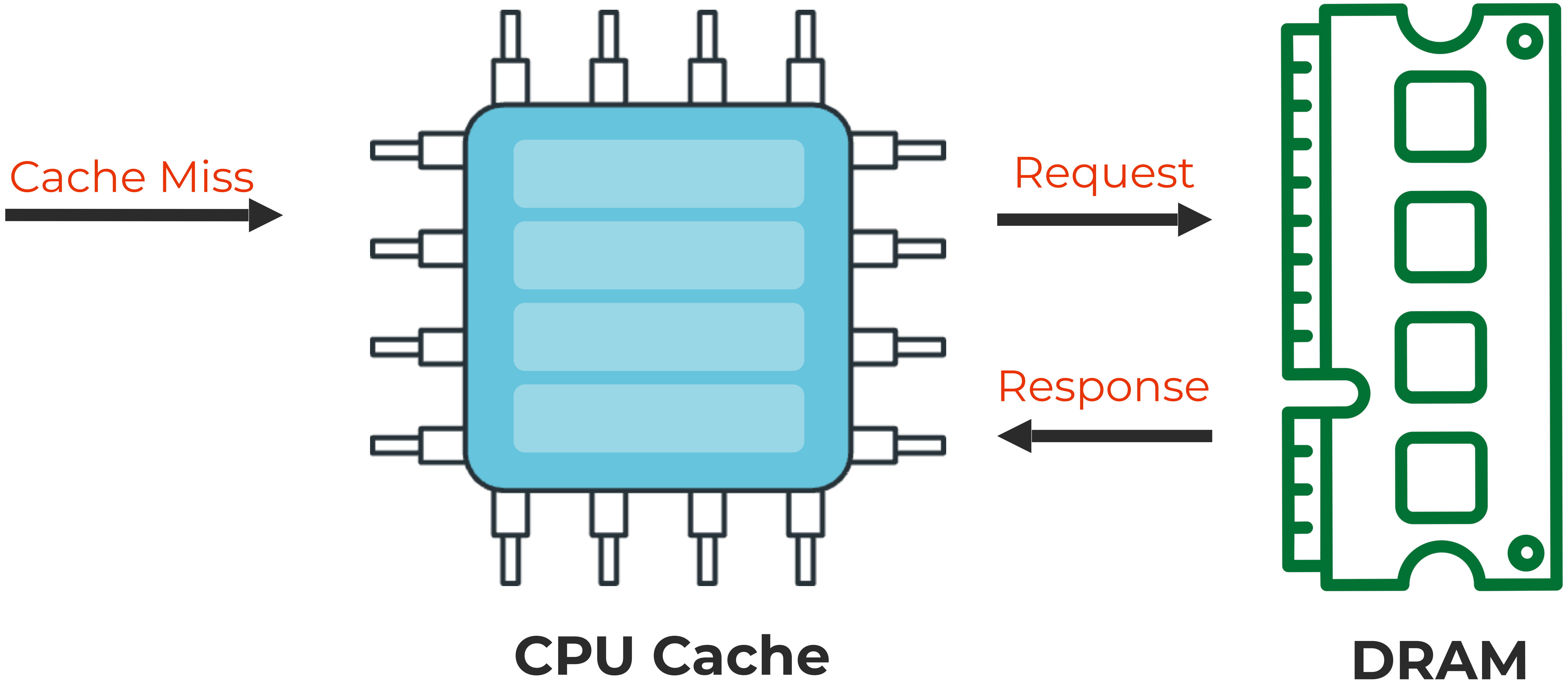
Request

CPU Cache

DRAM

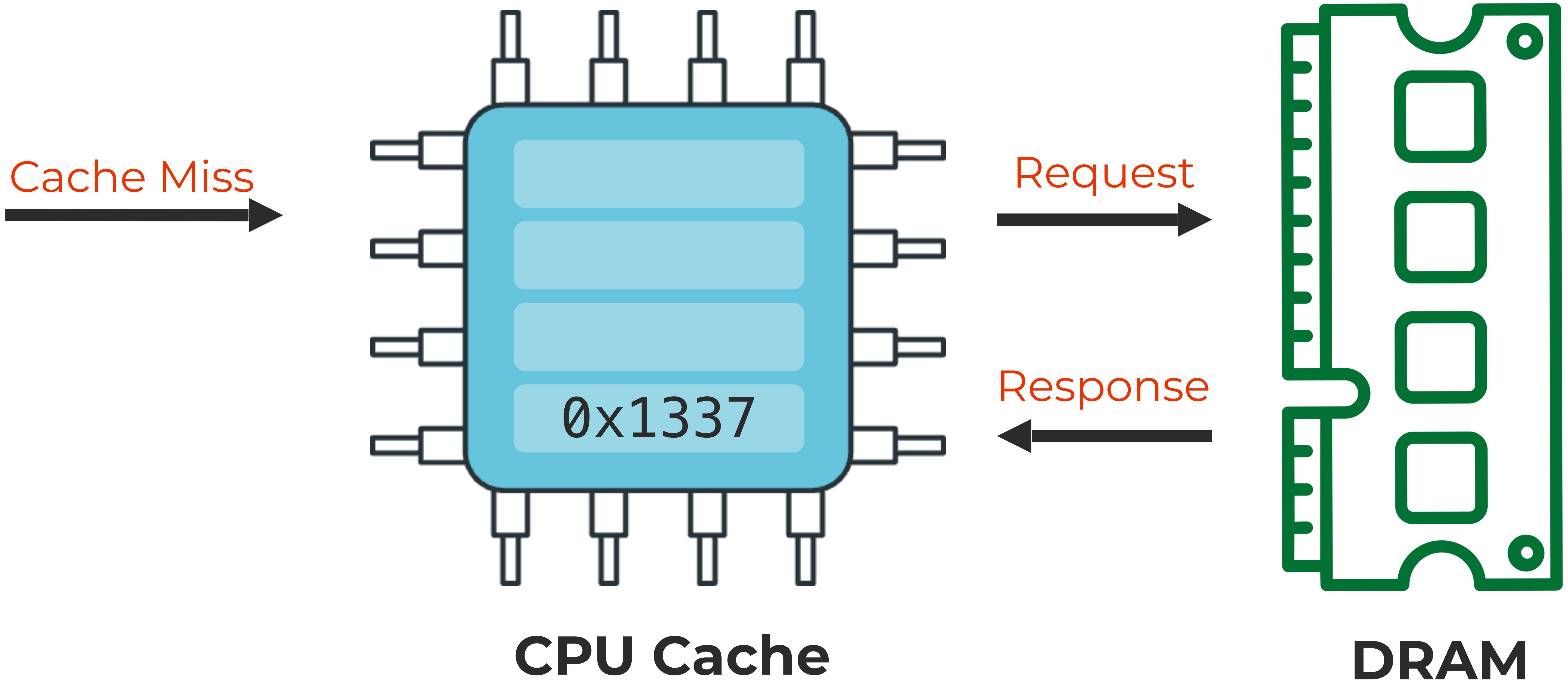
CPU Optimization: The Cache

```
access(array[0]);

access(array[0]);
```

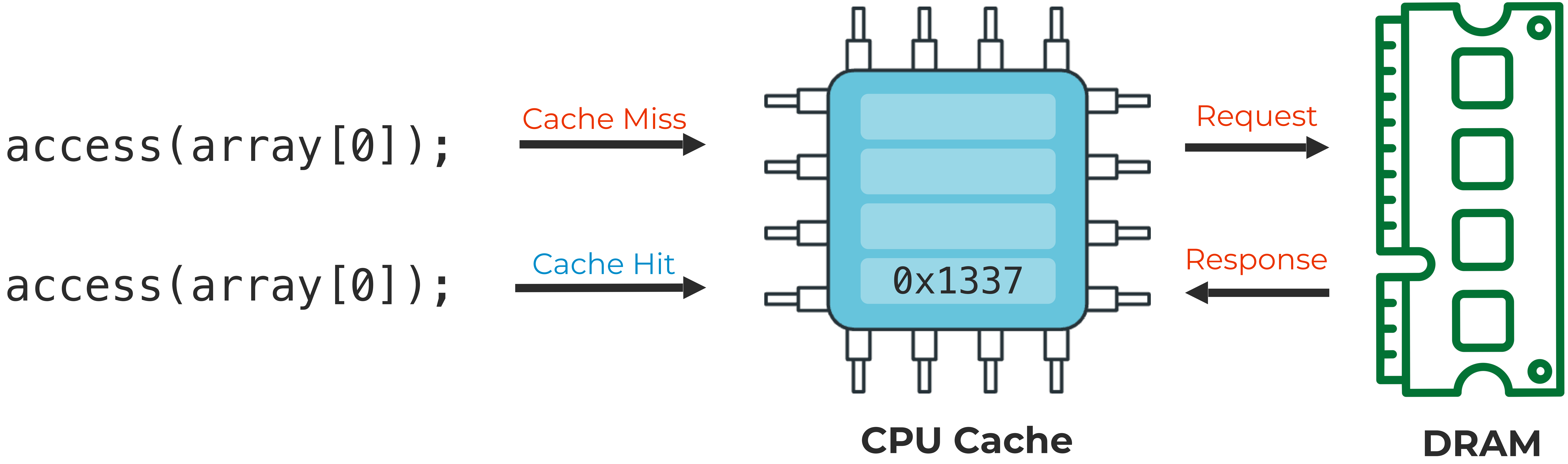


CPU Optimization: The Cache

```
access(array[0]);

access(array[0]);
```

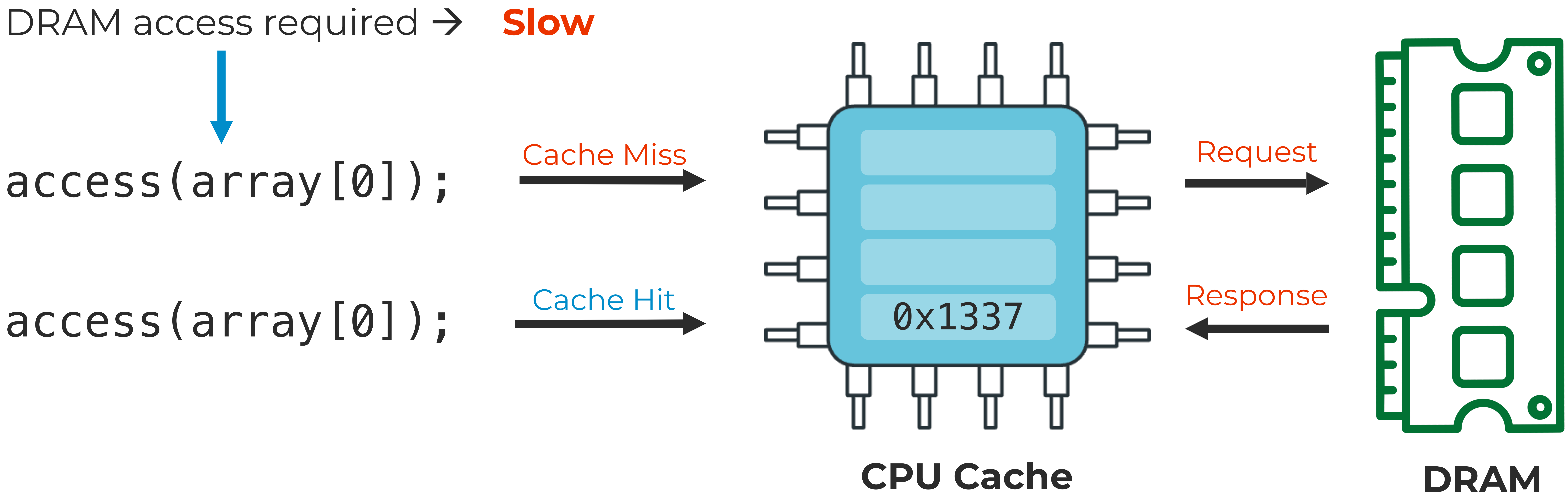


CPU Optimization: The Cache



```
access(array[0]);
access(array[0]);
```

CPU Optimization: The Cache

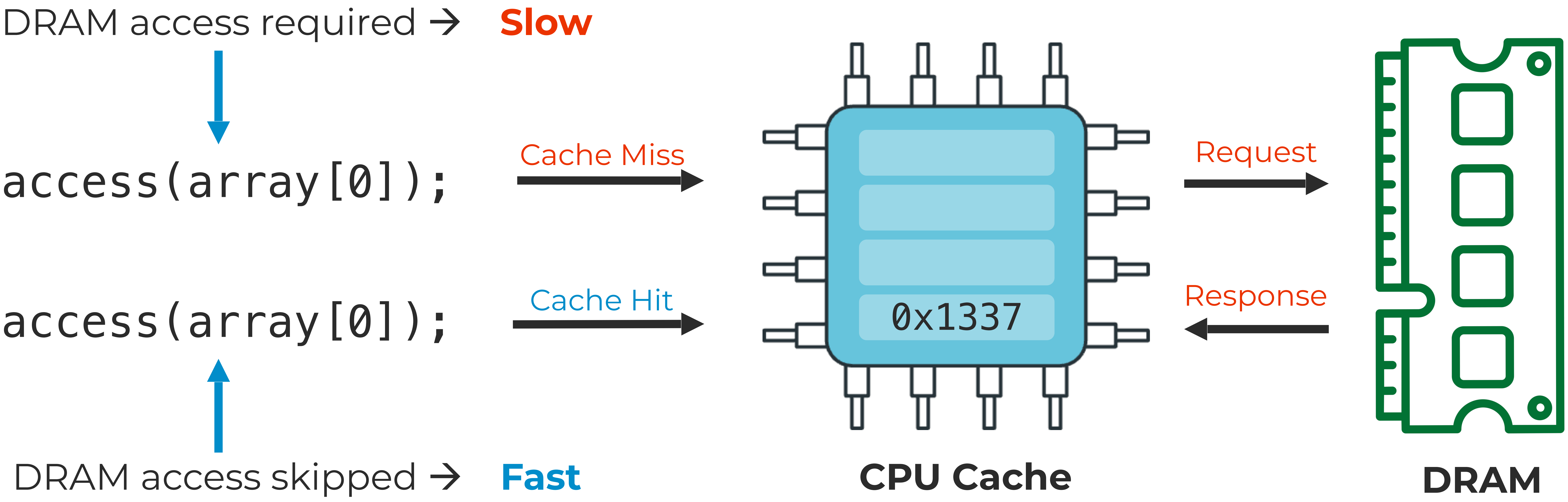


DRAM access required → **Slow**

```
access(array[0]);
```

```
access(array[0]);
```

CPU Optimization: The Cache



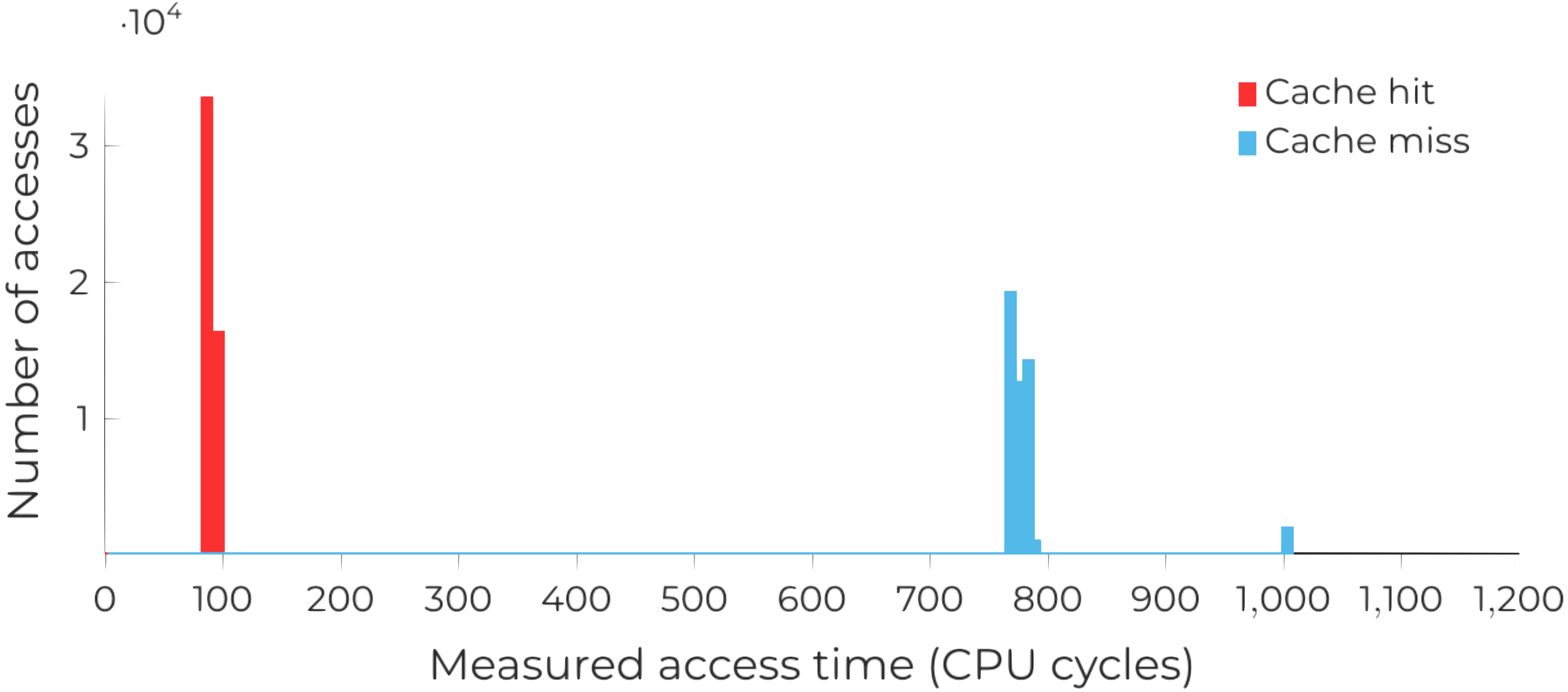
DRAM access required → **Slow**

```
access(array[0]);
```

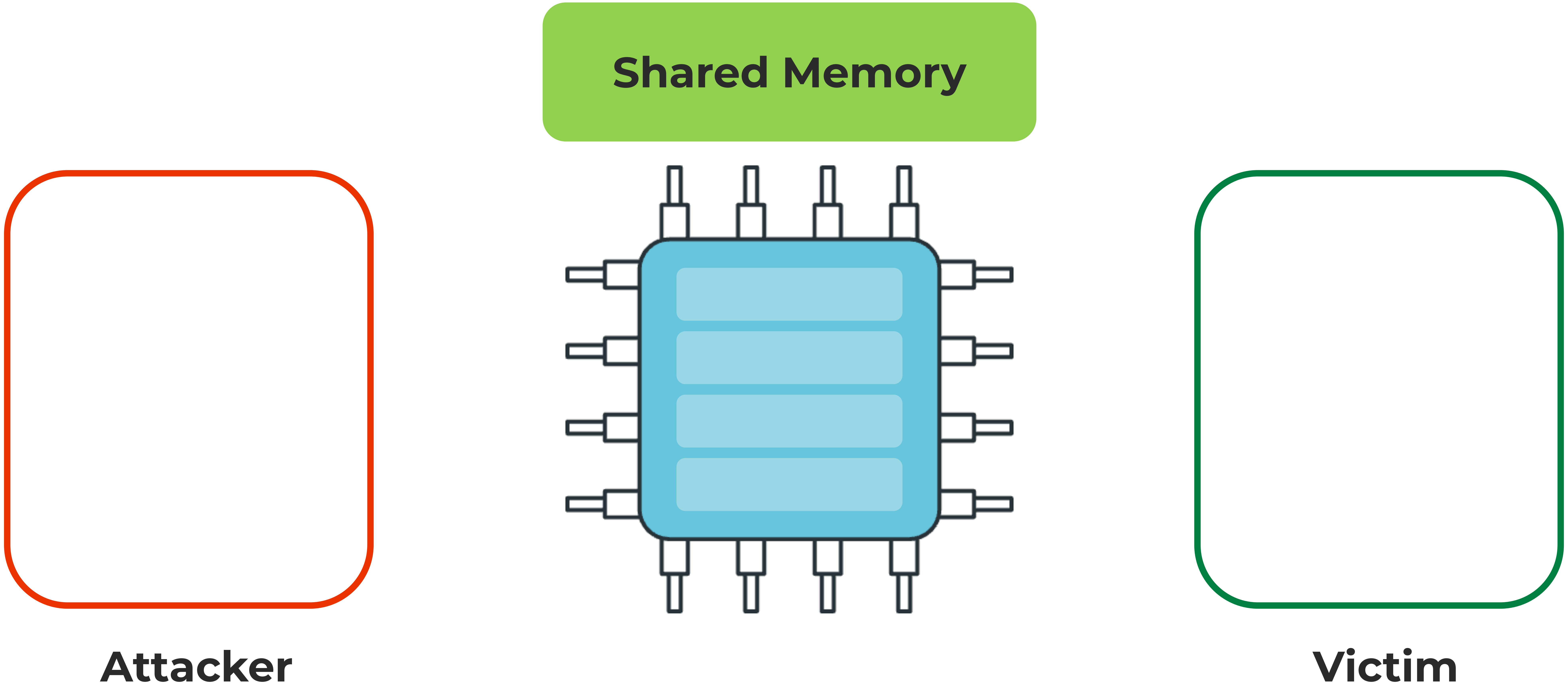
```
access(array[0]);
```

DRAM access skipped → **Fast**

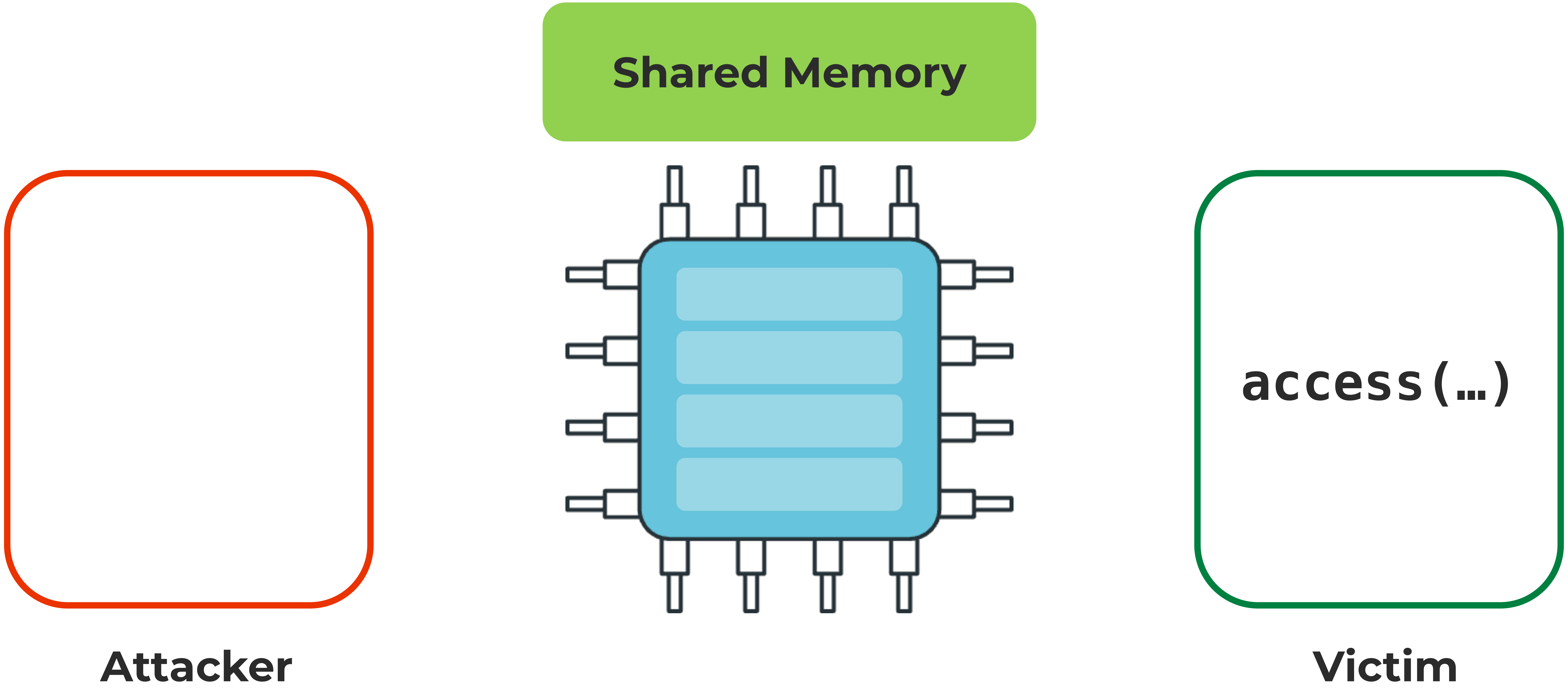
Measuring Cache Timings



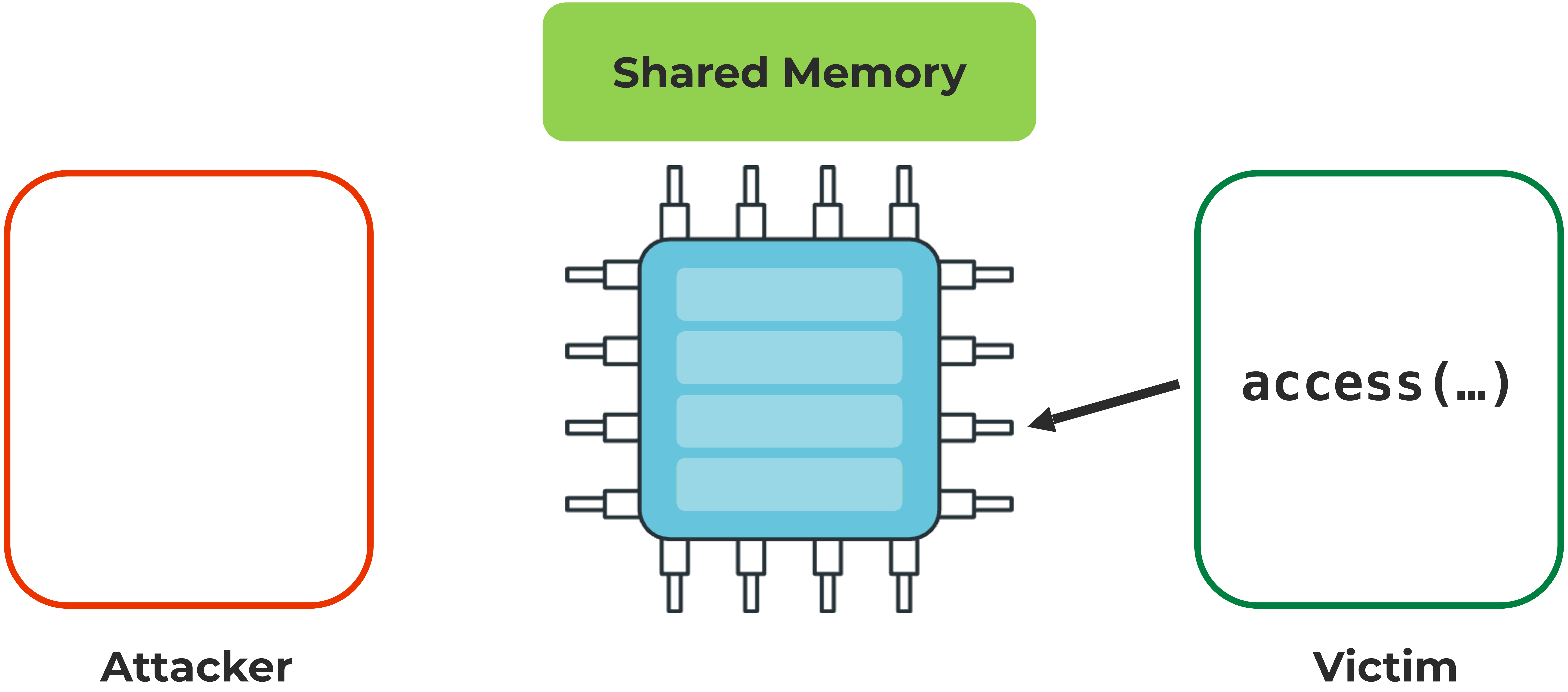
Flush+Reload: Shared Memory



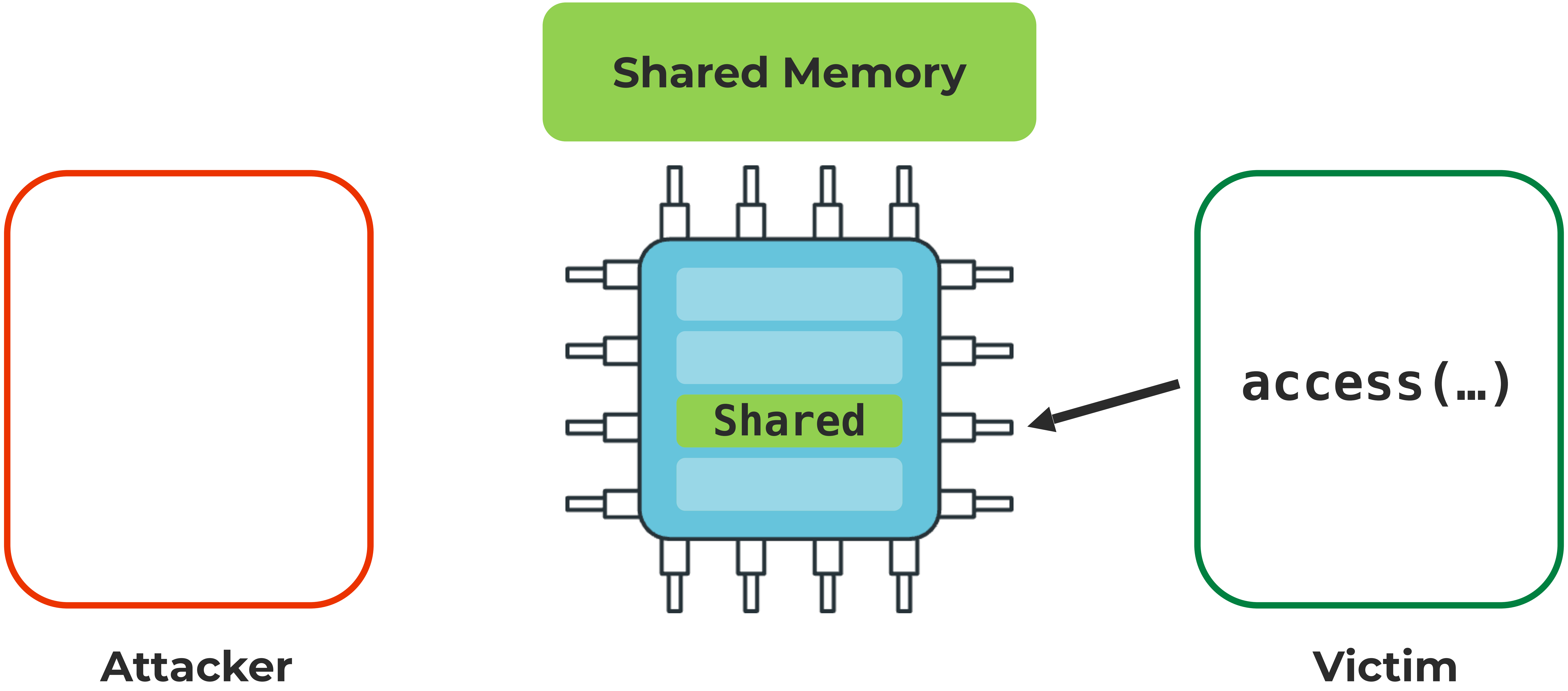
Flush+Reload: Shared Memory



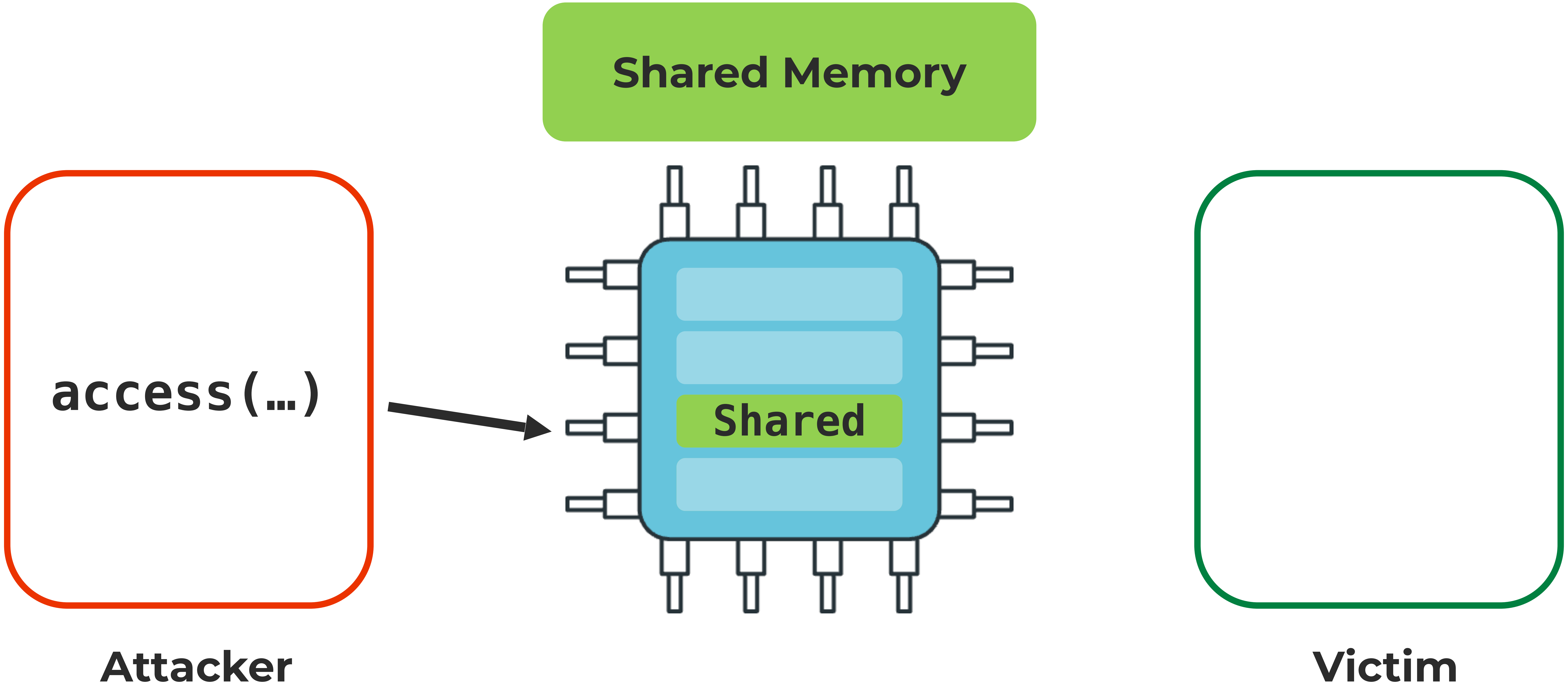
Flush+Reload: Shared Memory



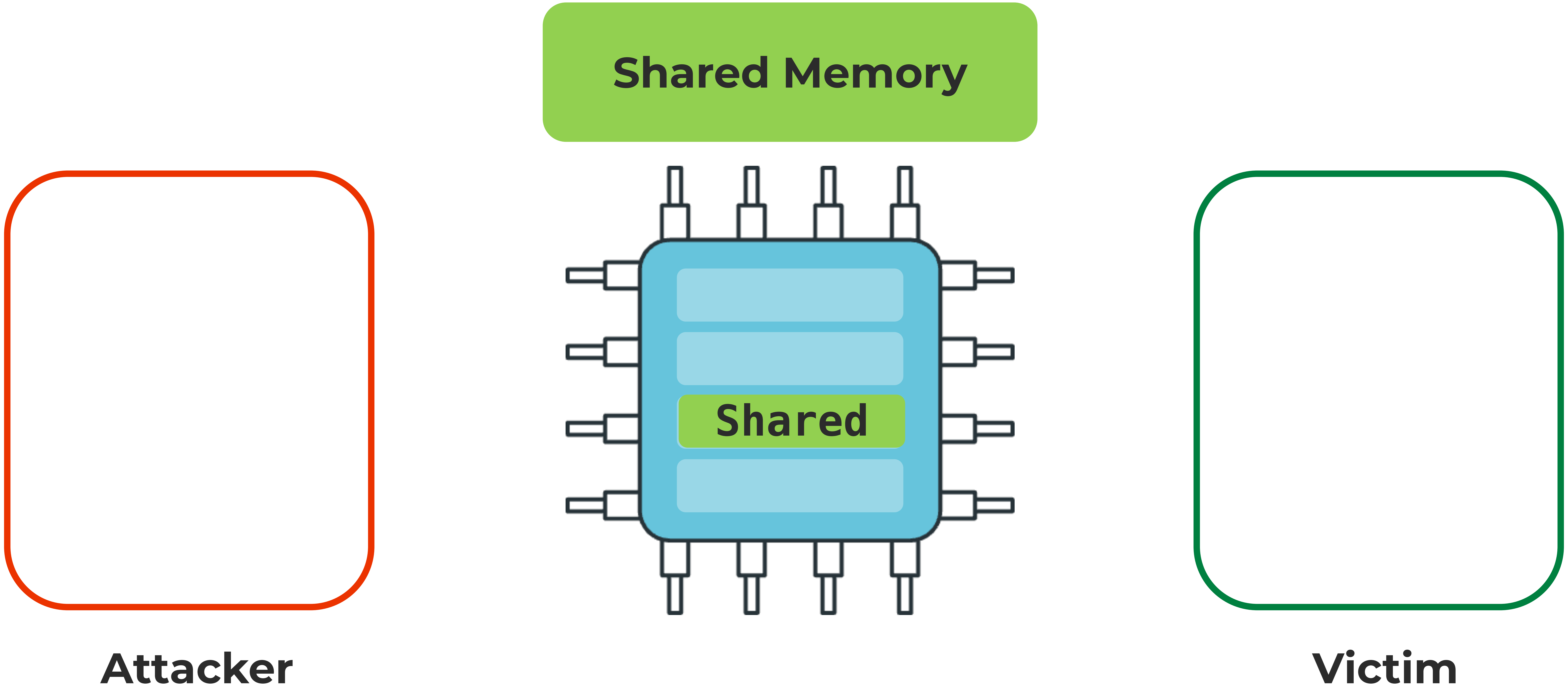
Flush+Reload: Shared Memory



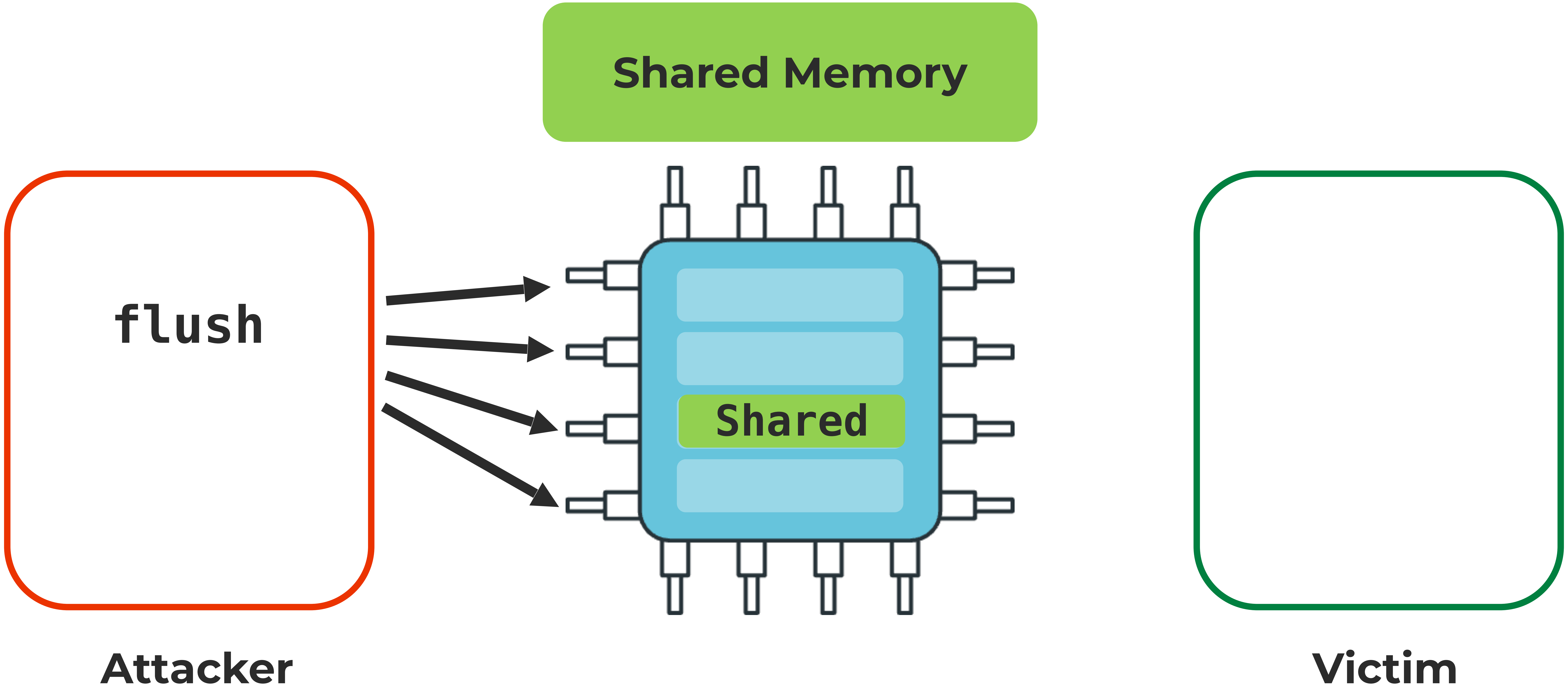
Flush+Reload: Shared Memory



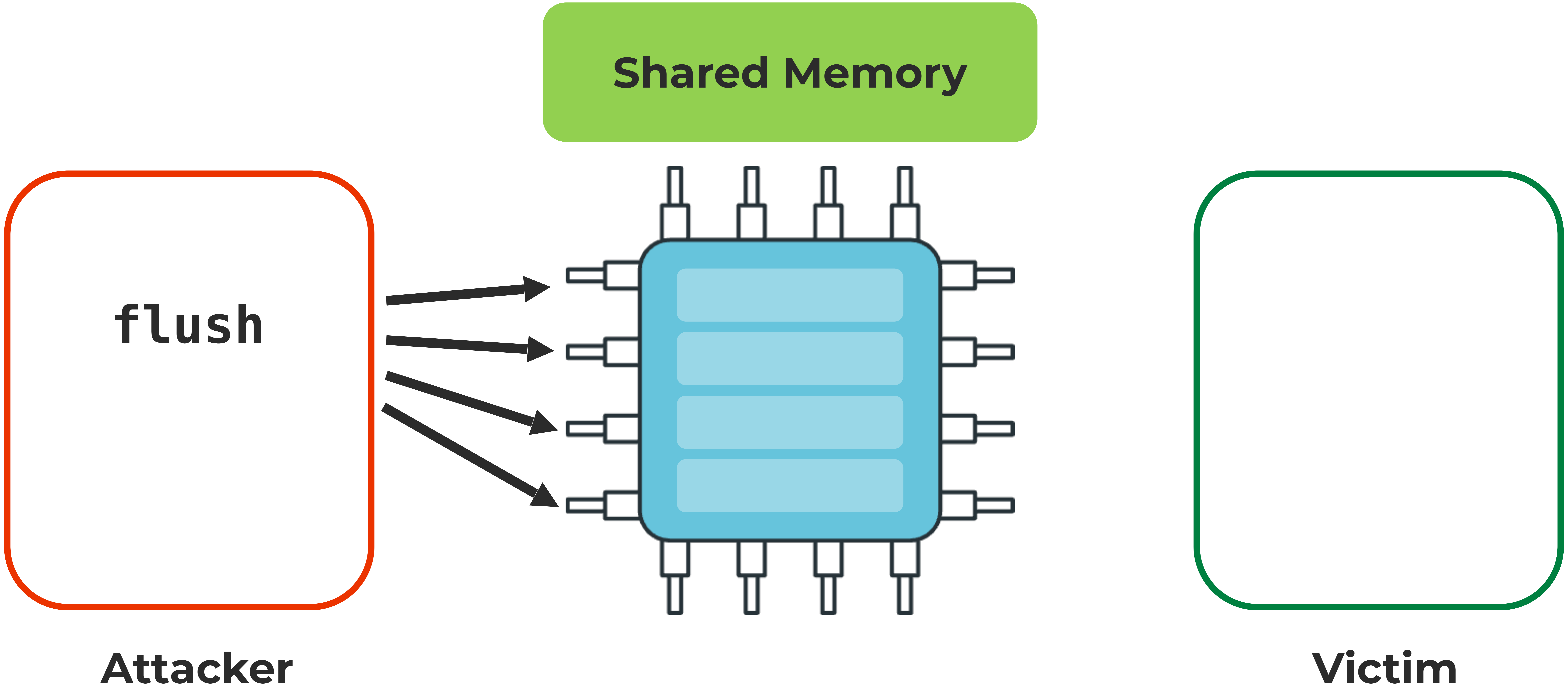
Flush+Reload



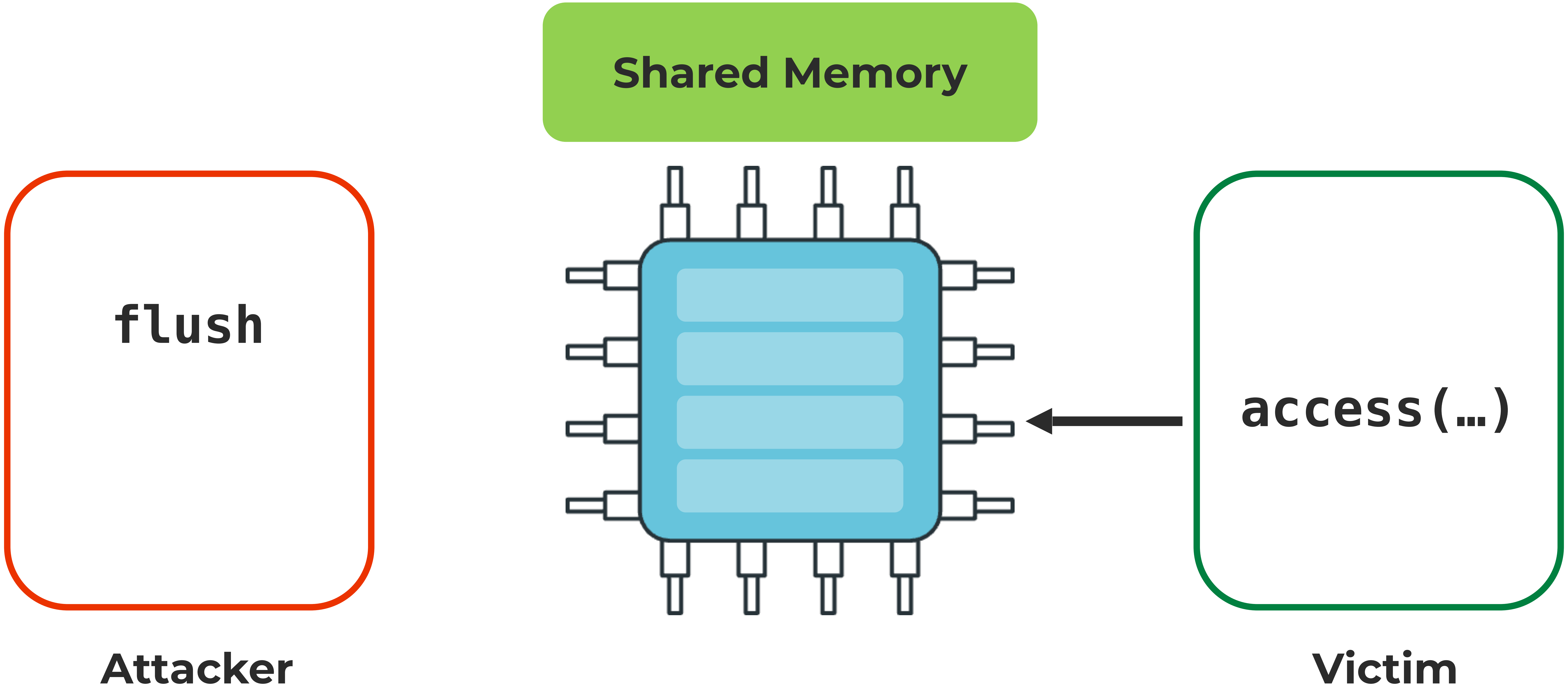
Flush+Reload



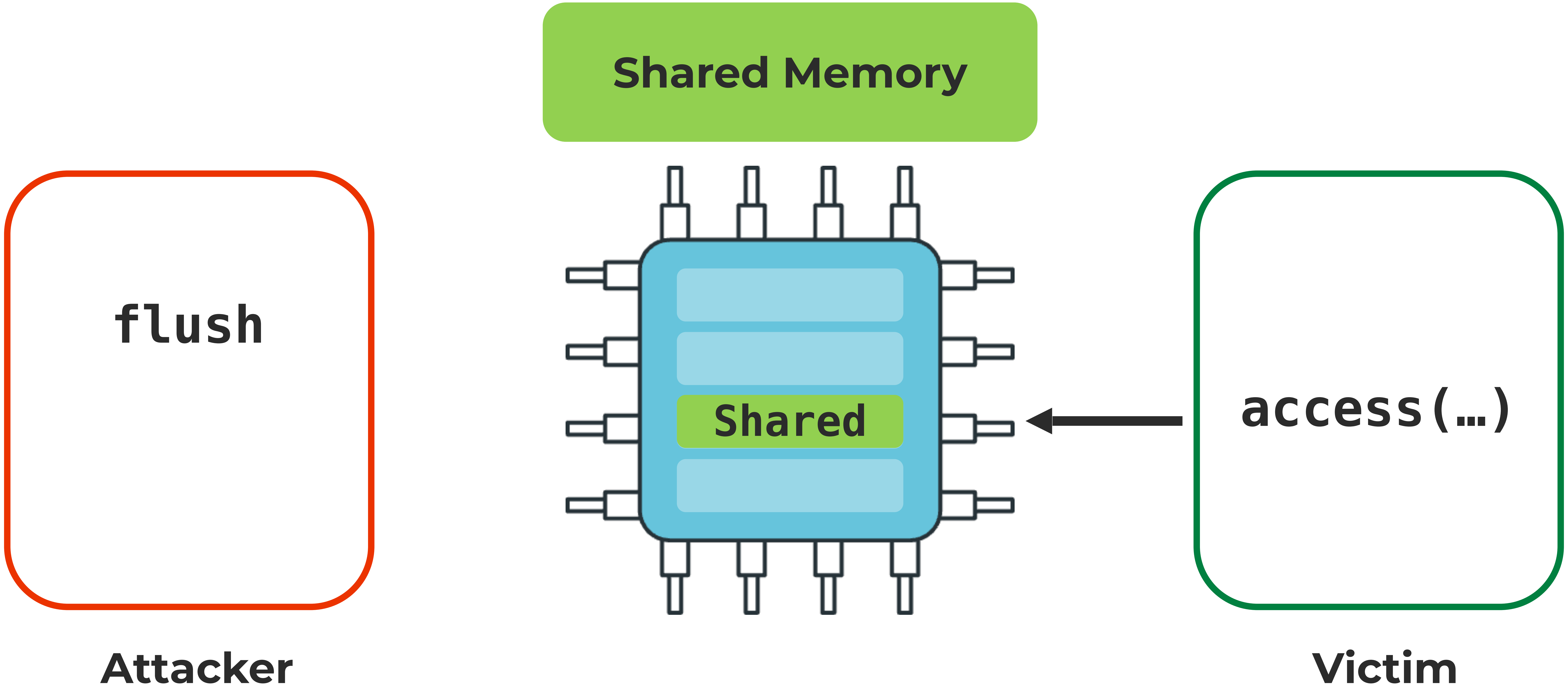
Flush+Reload



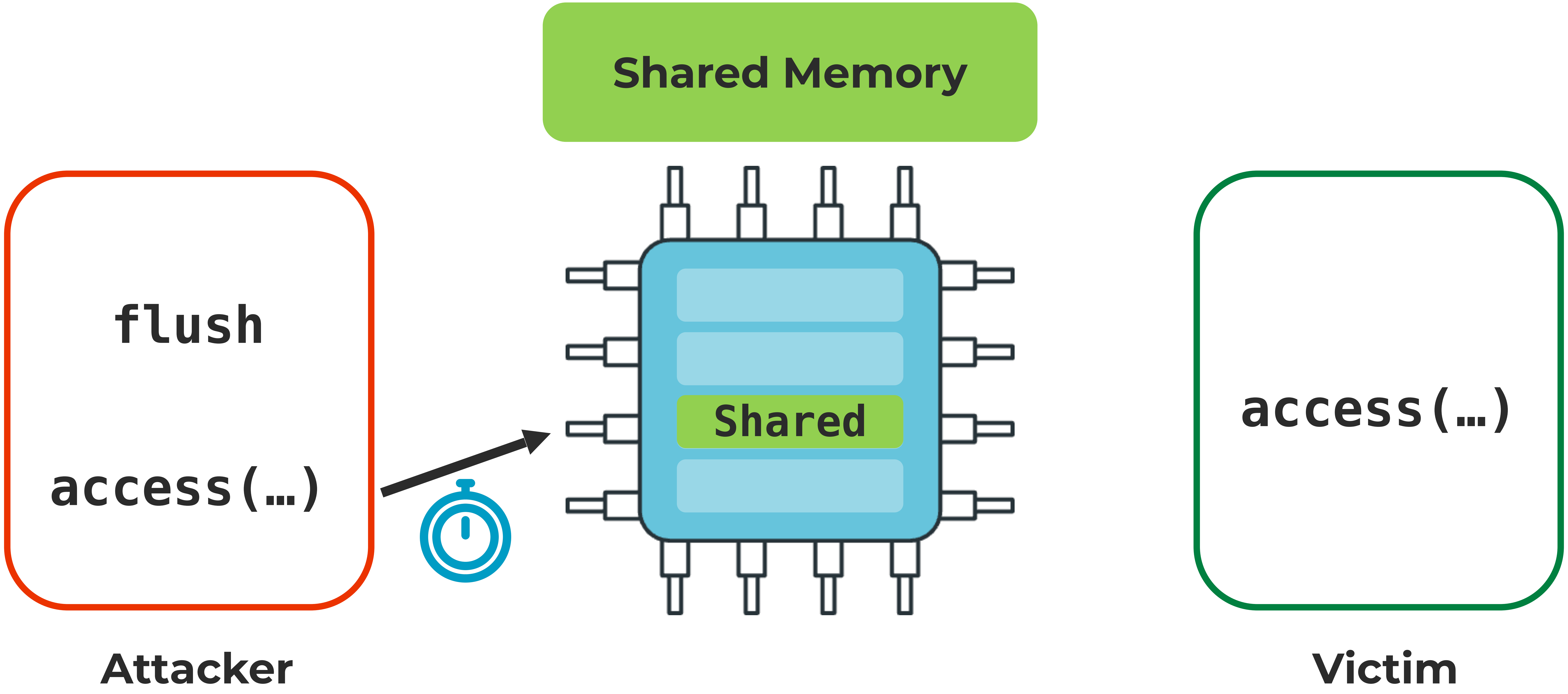
Flush+Reload



Flush+Reload



Flush+Reload



Flush+Reload is Mitigated on RISC-V

Flush+Reload is Mitigated on RISC-V

- Flush+Reload is typically used to **spy on control flow**

Flush+Reload is Mitigated on RISC-V

- Flush+Reload is typically used to **spy on control flow**
  - **Requires shared caches** for data and instructions

Flush+Reload is Mitigated on RISC-V

- Flush+Reload is typically used to **spy on control flow**
  - **Requires shared caches** for data and instructions

- The RISC-V cores have **split data and instruction caches**

Flush+Reload is Mitigated on RISC-V

- Flush+Reload is typically used to **spy on control flow**
  - **Requires shared caches** for data and instructions
- The RISC-V cores have **split data and instruction caches**
- Cache design **mitigates** Flush+Reload

Flush+Reload is Mitigated on RISC-V

- Flush+Reload is typically used to **spy on control flow**
  - **Requires shared caches** for data and instructions

- The RISC-V cores have **split data and instruction caches**

- Cache design **mitigates** Flush+Reload

→ **New Attack Variant:**

Flush+Fault

Flush+Fault

Attacker

```
if(secret){
    A();
} else {
    B();
}
```

Victim

Flush+Fault

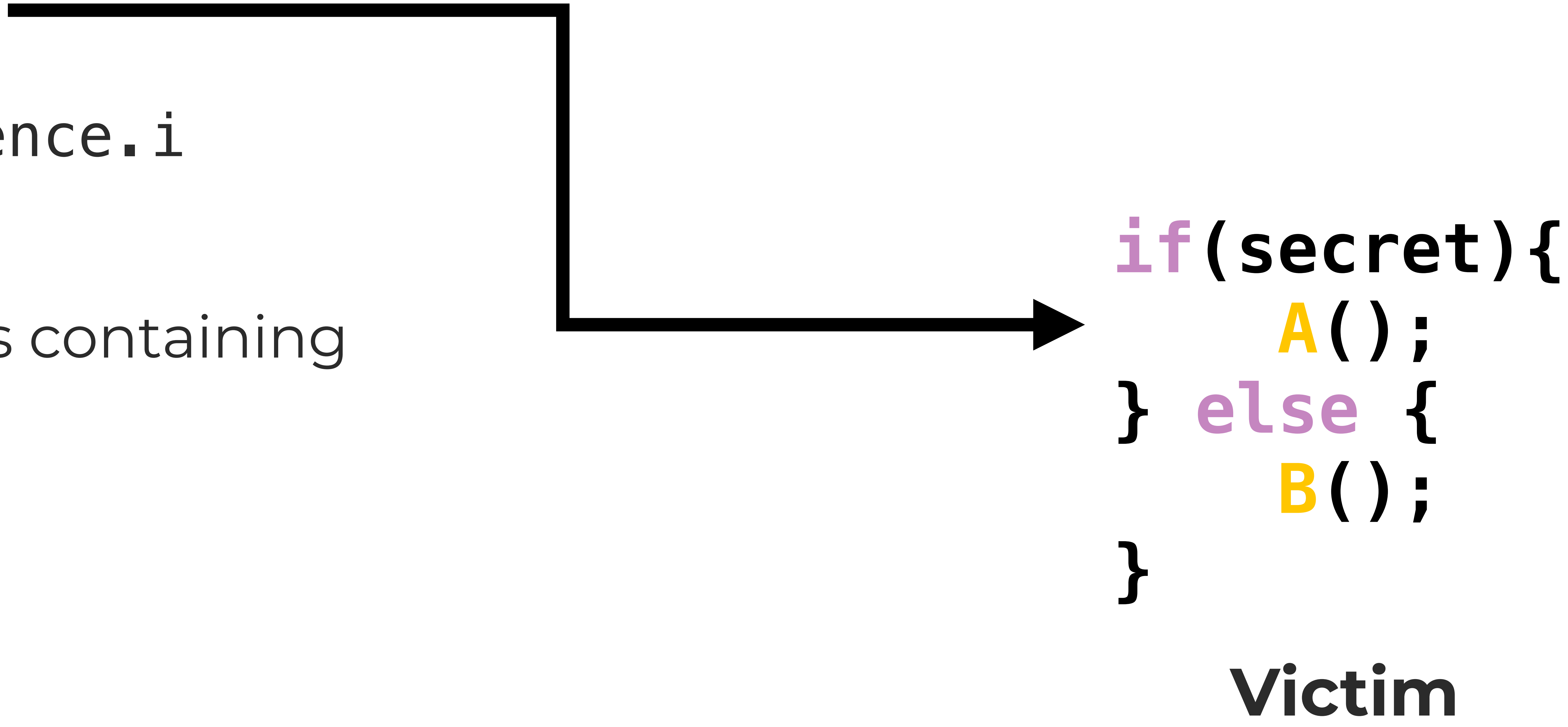
Attacker

I. Flush I-Cache with `fence.i`

```
if(secret){
    A();
} else {
    B();
}
```

Victim

Flush+Fault



Attacker

I. Flush I-Cache with `fence.i`

II. Time jump to address containing victim cache line

```
if(secret){
    A();
} else {
    B();
}
```

Victim

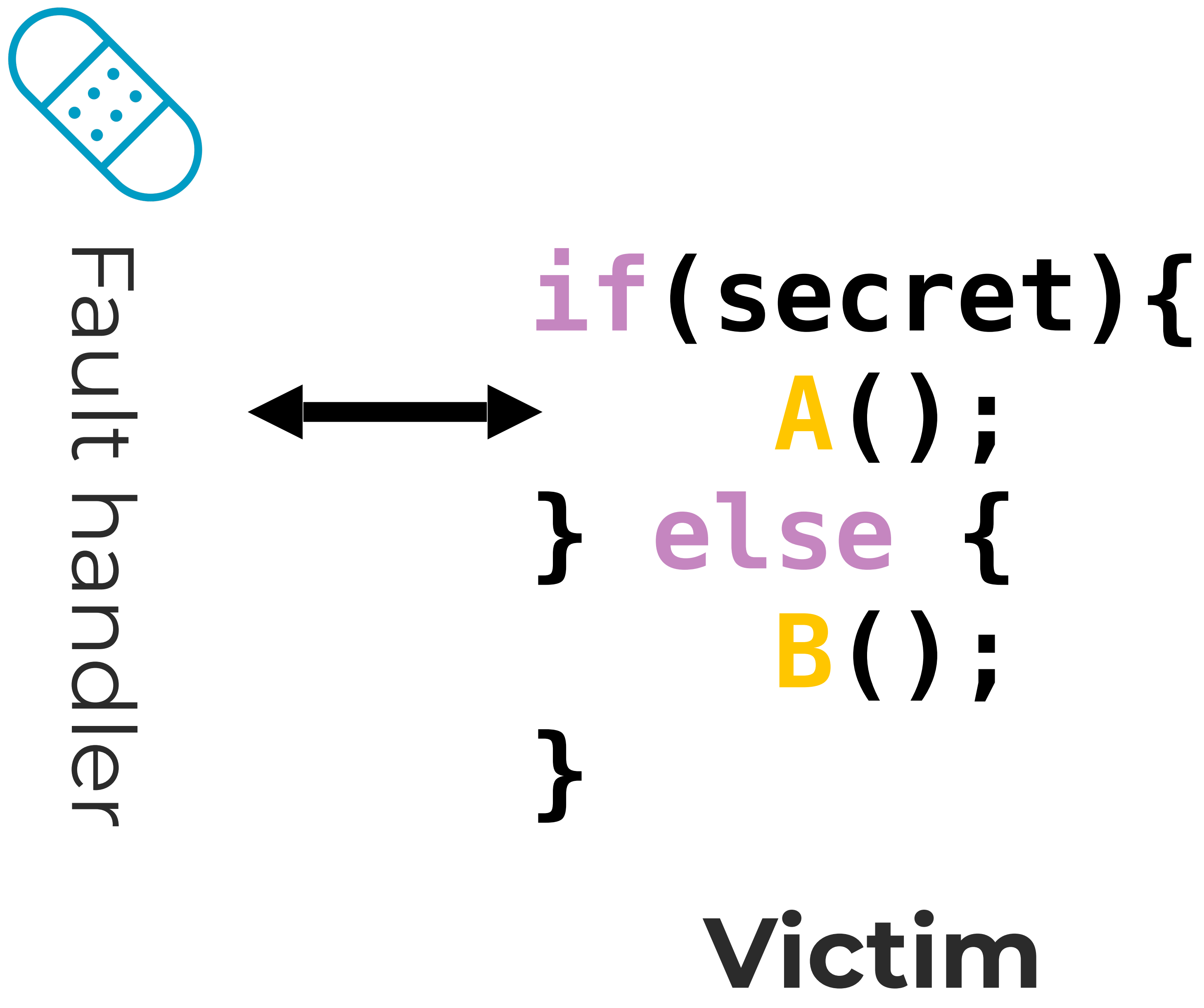
Flush+Fault

Attacker

I. Flush I-Cache with `fence.i`

II. Time jump to address containing victim cache line

III. Handle Fault



```
if(secret){
    A();
} else {
    B();
}
```

Flush+Fault

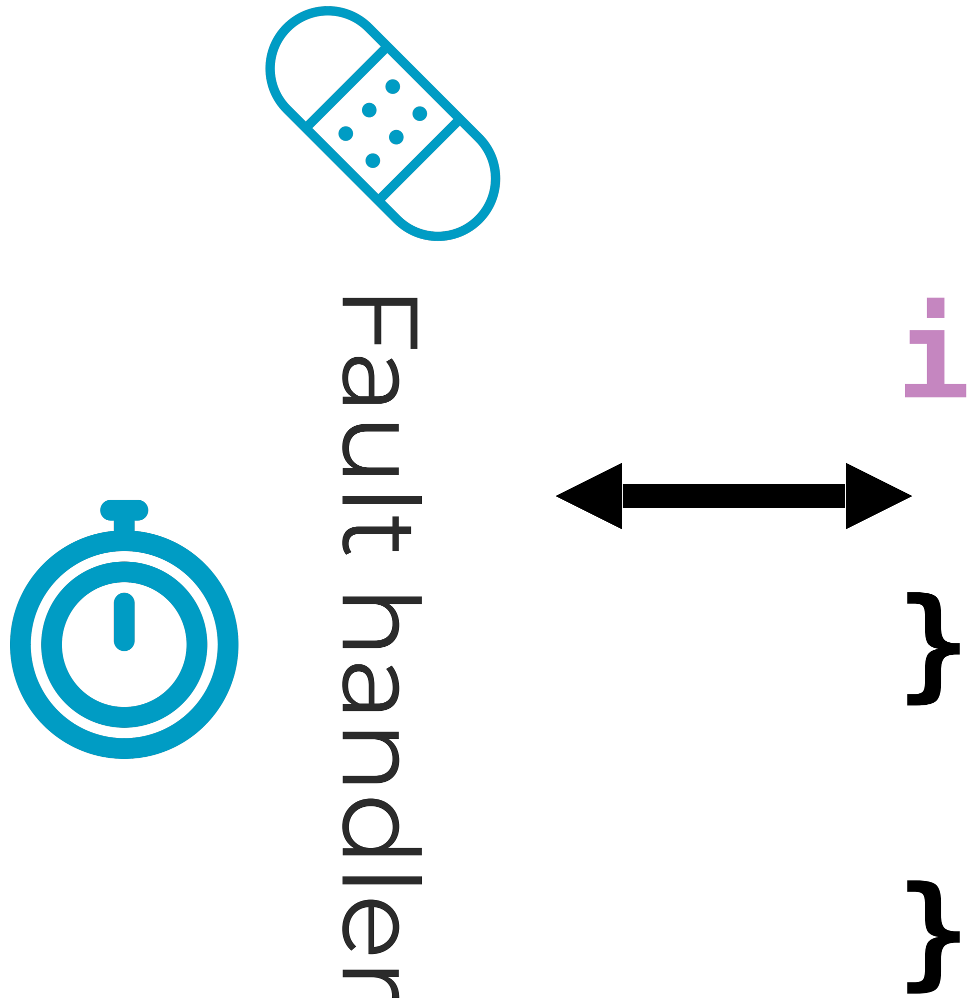
Attacker

I. Flush I-Cache with `fence.i`

II. Time jump to address containing victim cache line

III. Handle Fault

IV. Timing of fault handling leaks secret



```
if(secret){
    A();
} else {
    B();
}
```

Victim

Lesson Learned: Cache Attacks are Still Possible

Lesson Learned: Cache Attacks are Still Possible

- The cache design **mitigates well-known attacks**

Lesson Learned: Cache Attacks are Still Possible

- The cache design **mitigates well-known attacks**
  - e.g., Flush+Reload

Lesson Learned: Cache Attacks are Still Possible

- The cache design **mitigates well-known attacks**
  - e.g., Flush+Reload

- **Adaptions are still possible**

Lesson Learned: Cache Attacks are Still Possible

- The cache design **mitigates well-known attacks**
  - e.g., Flush+Reload

- **Adaptions are still possible**
  - Flush+Fault

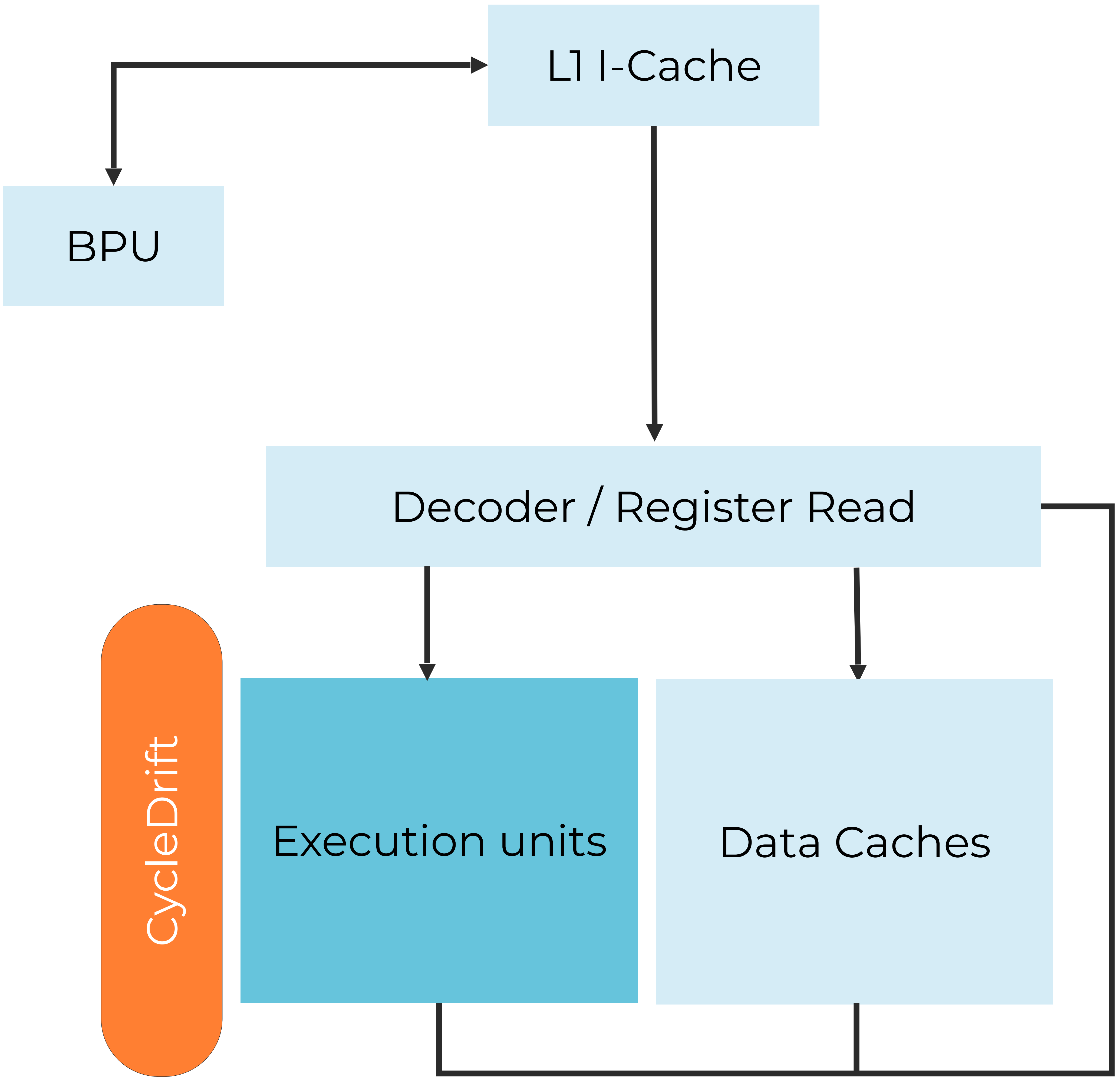
Lesson Learned: Cache Attacks are Still Possible

- The cache design **mitigates well-known attacks**
  - e.g., Flush+Reload

- **Adaptions are still possible**
  - Flush+Fault
  - Data-cache Attacks

Nice! But What About the Other Attacks?

CPU Design: Performance Counters



What are Performance Counters?

CPUs are complex and hard to benchmark

What are Performance Counters?

CPUs are complex and hard to benchmark

→ Performance Counters ease **benchmarking**

What are Performance Counters?

CPUs are complex and hard to benchmark

→ Performance Counters ease **benchmarking**

- Performance Counters **count/report events** such as...

What are Performance Counters?

CPUs are complex and hard to benchmark

→ Performance Counters ease **benchmarking**

- Performance Counters **count/report events** such as...
  - ... cache misses

What are Performance Counters?

CPUs are complex and hard to benchmark

→ Performance Counters ease **benchmarking**

- Performance Counters **count/report events** such as…
  - … cache misses
  - … instructions executed

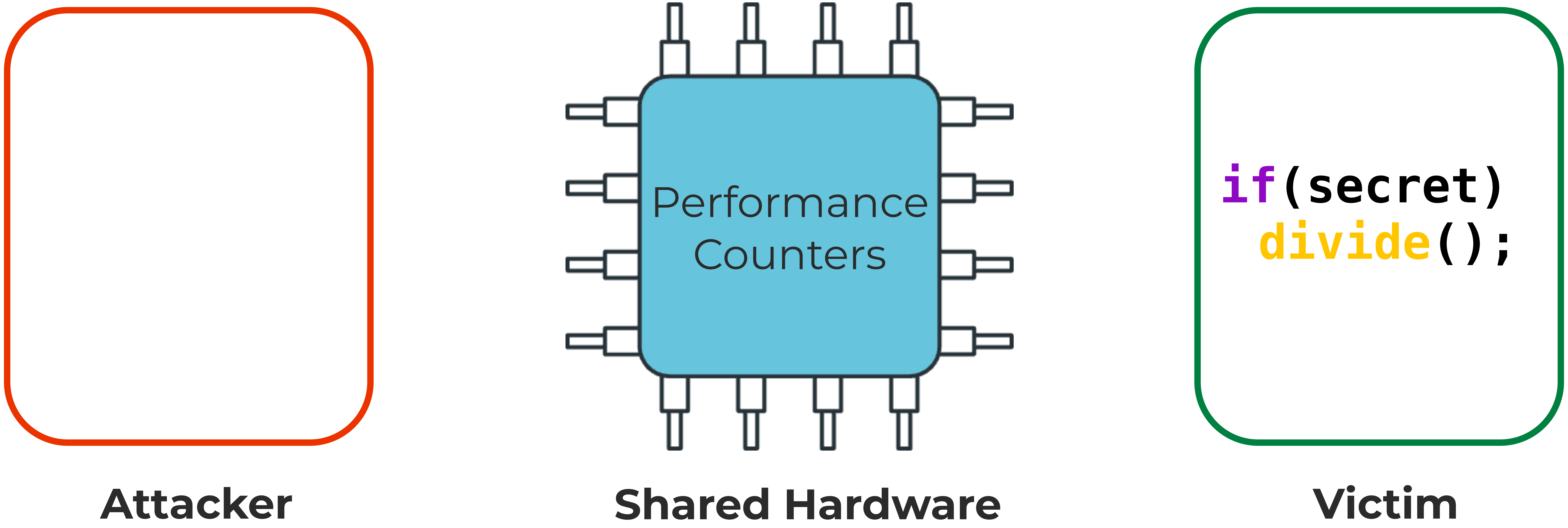
What are Performance Counters?

CPUs are complex and hard to benchmark

→ Performance Counters ease **benchmarking**

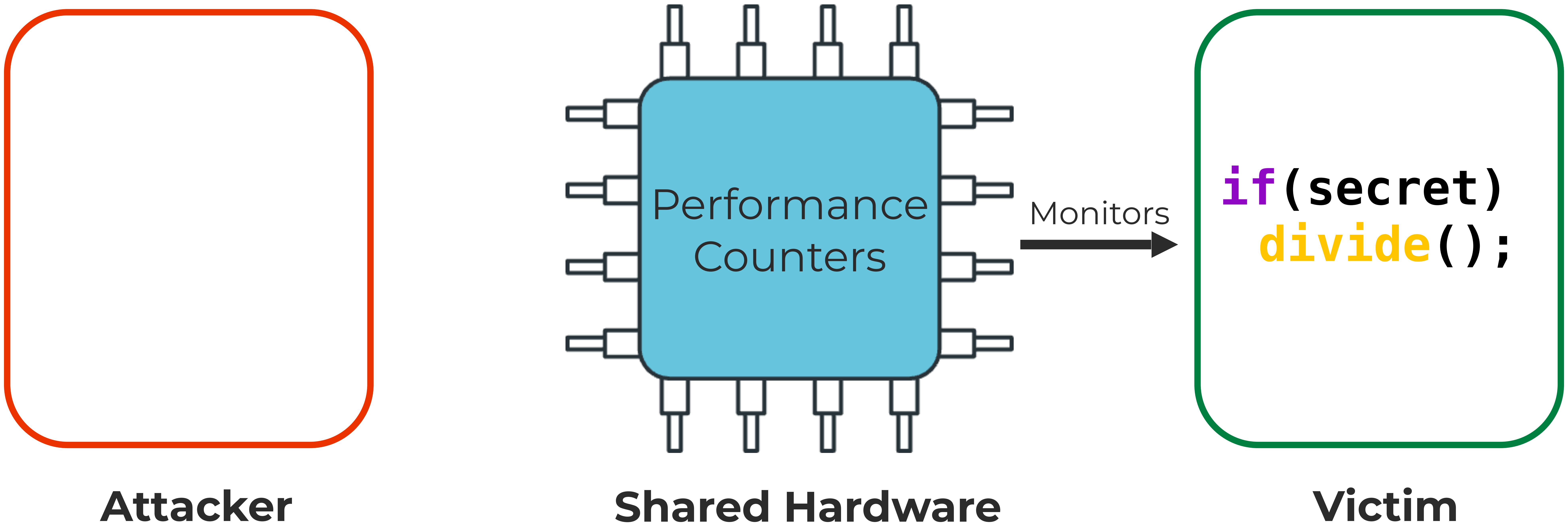
- Performance Counters **count/report events** such as...
  - ... cache misses
  - ... instructions executed
  - ... CPU Frequency

Why are Performance Counters Dangerous?



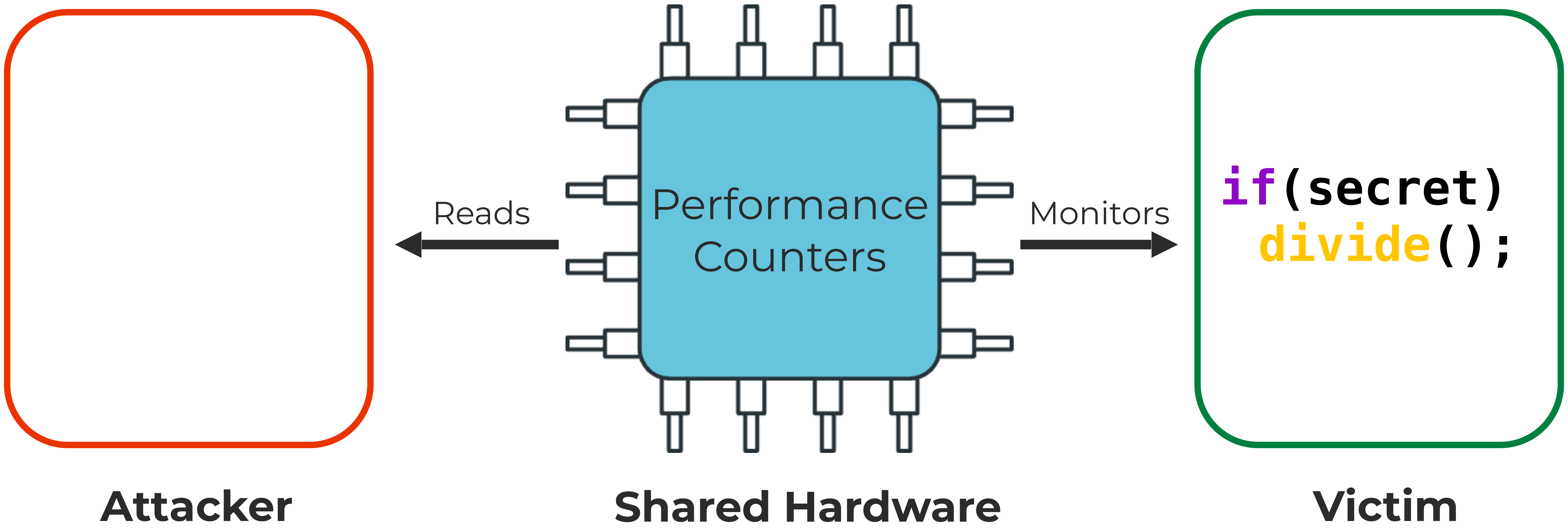
```
if(secret)
  divide();
```

Why are Performance Counters Dangerous?



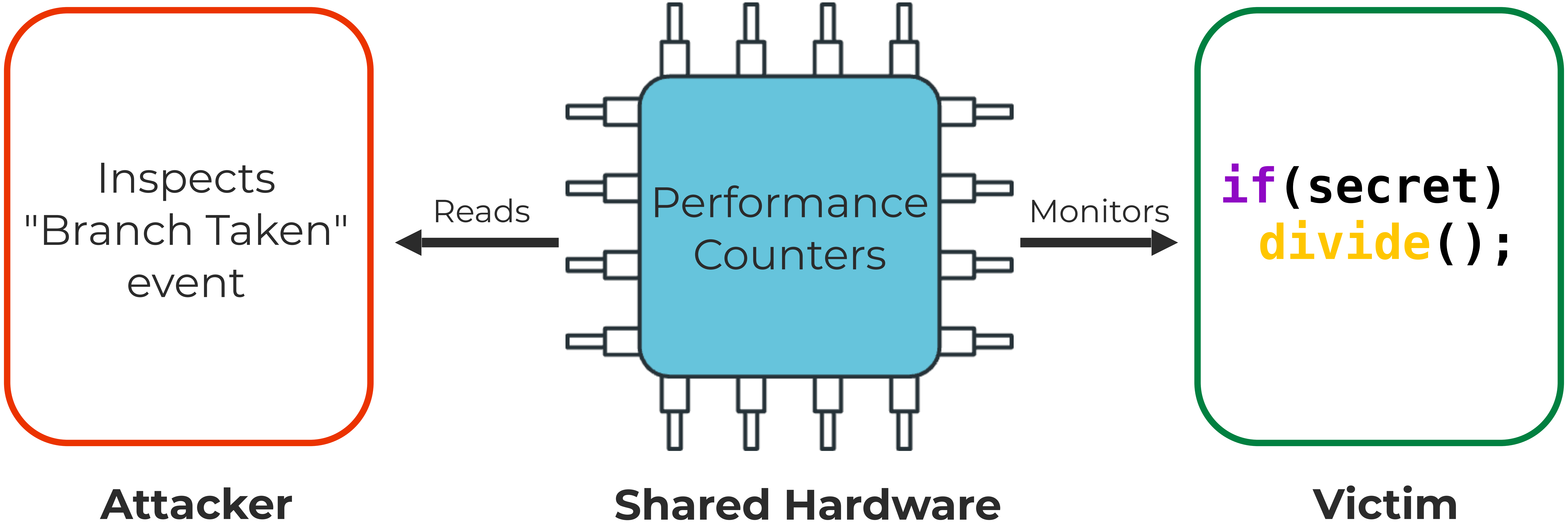
```
if(secret)
  divide();
```

Why are Performance Counters Dangerous?



```
if(secret)
  divide();
```

Why are Performance Counters Dangerous?



Inspects "Branch Taken" event

Reads

Performance Counters

Monitors

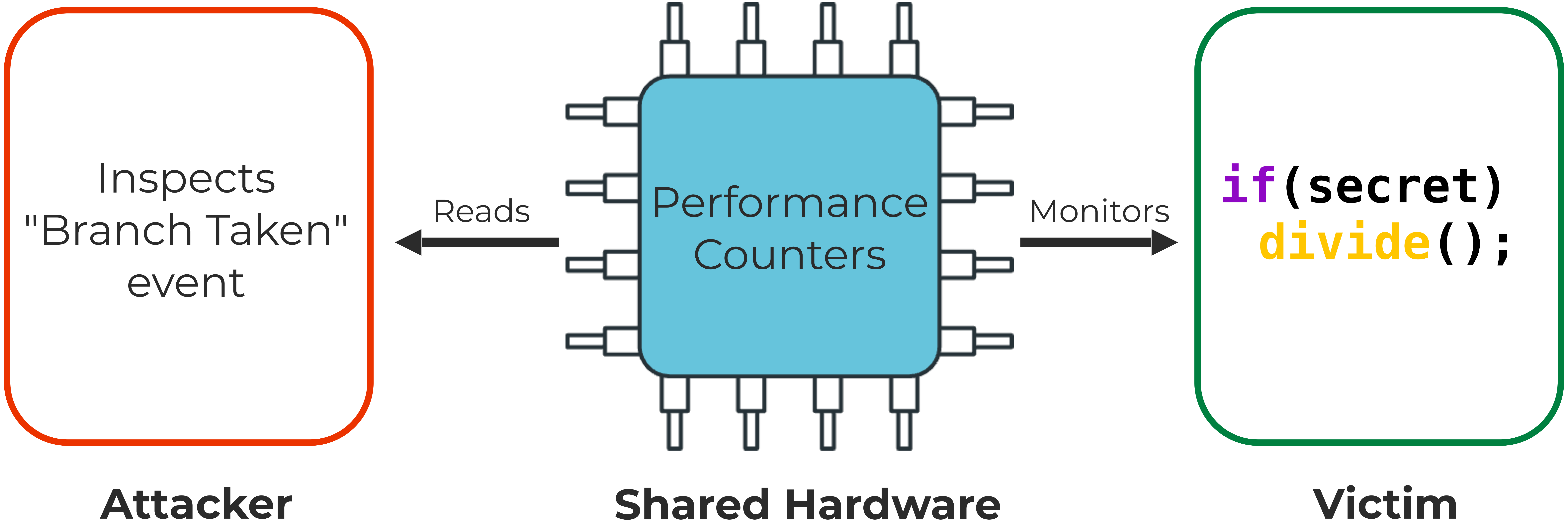
```
if(secret)
  divide();
```

Attacker

Shared Hardware

Victim

Why are Performance Counters Dangerous?



```
if(secret)
  divide();
```

Fix: Make the interface **privileged** (**root only**)!

What Happens on RISC-V?

What Happens on RISC-V?

- Some Performance Counter are still **unprivileged**!

What Happens on RISC-V?

- Some Performance Counter are still **unprivileged**!
  - Dictated by the RISC-V ISA standard

What Happens on RISC-V?

- Some Performance Counter are still **unprivileged**!
  - Dictated by the RISC-V ISA standard

- Only few **unprivileged** events

What Happens on RISC-V?

- Some Performance Counter are still **unprivileged**!
  - Dictated by the RISC-V ISA standard

- Only few **unprivileged** events
  - No "Branch Taken" event!

What Happens on RISC-V?

- Some Performance Counter are still **unprivileged**!
  - Dictated by the RISC-V ISA standard
- Only few **unprivileged** events
  - No "Branch Taken" event!
- Existing Events:

What Happens on RISC-V?

- Some Performance Counter are still **unprivileged**!
  - Dictated by the RISC-V ISA standard
- Only few **unprivileged** events
  - No "Branch Taken" event!
- Existing Events:
  - Number of **CPU cycles elapsed**

What Happens on RISC-V?

- Some Performance Counter are still **unprivileged**!
  - Dictated by the RISC-V ISA standard
- Only few **unprivileged** events
  - No "Branch Taken" event!
- Existing Events:
  - Number of **CPU cycles elapsed**
  - Number of **Instructions executed**

KASLR: Kernel Address Space Layout Randomization

KASLR: Kernel Address Space Layout Randomization

- Kernel **exploits require** knowledge about **addresses**

KASLR: Kernel Address Space Layout Randomization

- Kernel **exploits require** knowledge about **addresses**
- **KASLR randomizes** the kernel **addresses** on boot

KASLR: Kernel Address Space Layout Randomization

- Kernel **exploits require** knowledge about **addresses**

- **KASLR randomizes** the kernel **addresses** on boot

→ **Can we break that maybe?**

Live Demo: CycleDrift

demo@lab24: ~/demos
⌥⌘1

```
demo@lab24:~/demos$ ./kaslr-break
```

Lesson Learned: Perf Attacks are Still Possible

Lesson Learned: Perf Attacks are Still Possible

- RISC-V cores have **unprivileged** performance counters

Lesson Learned: Perf Attacks are Still Possible

- RISC-V cores have **unprivileged** performance counters

- Performance counter **attacks are again possible**!

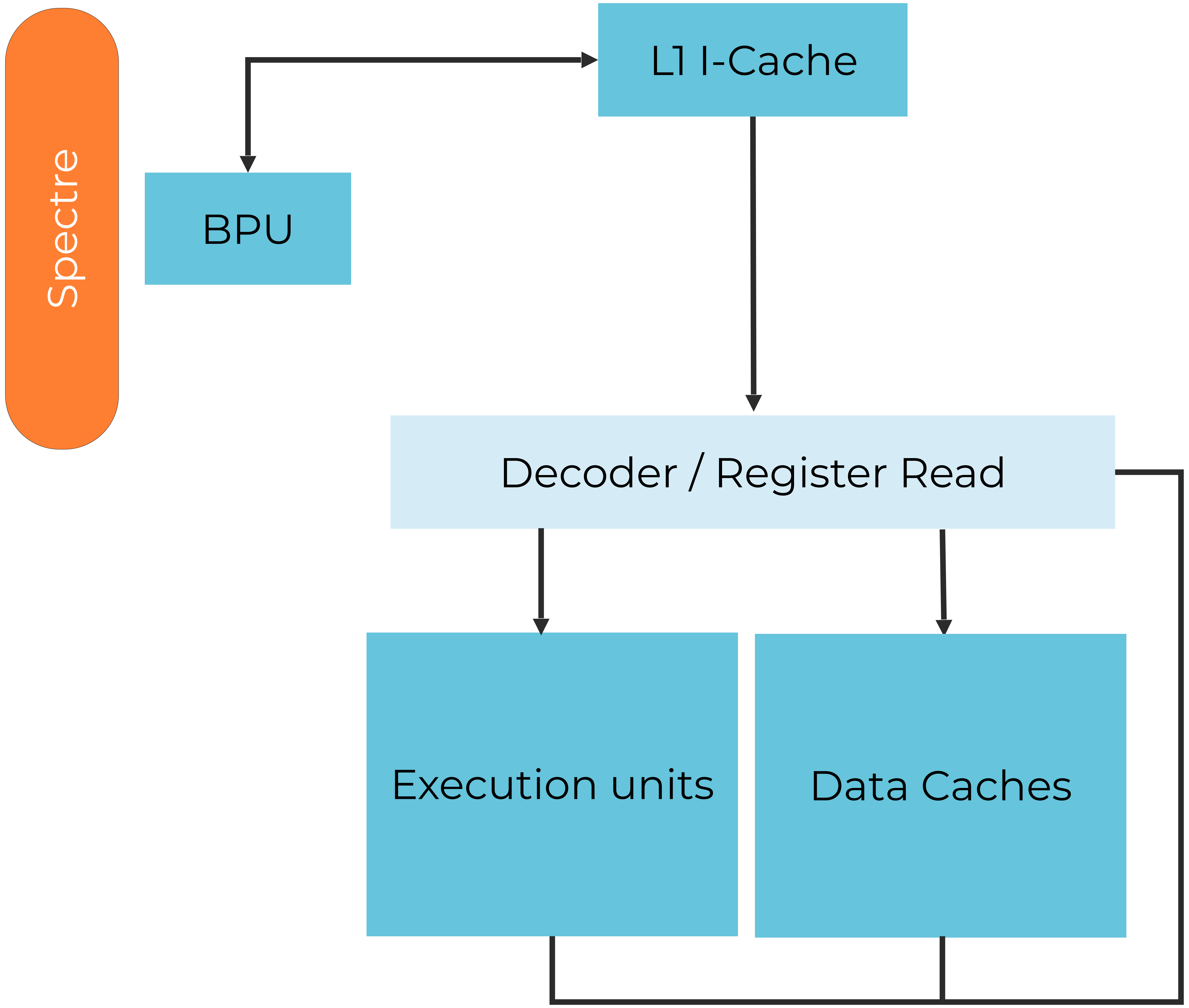
Lesson Learned: Perf Attacks are Still Possible

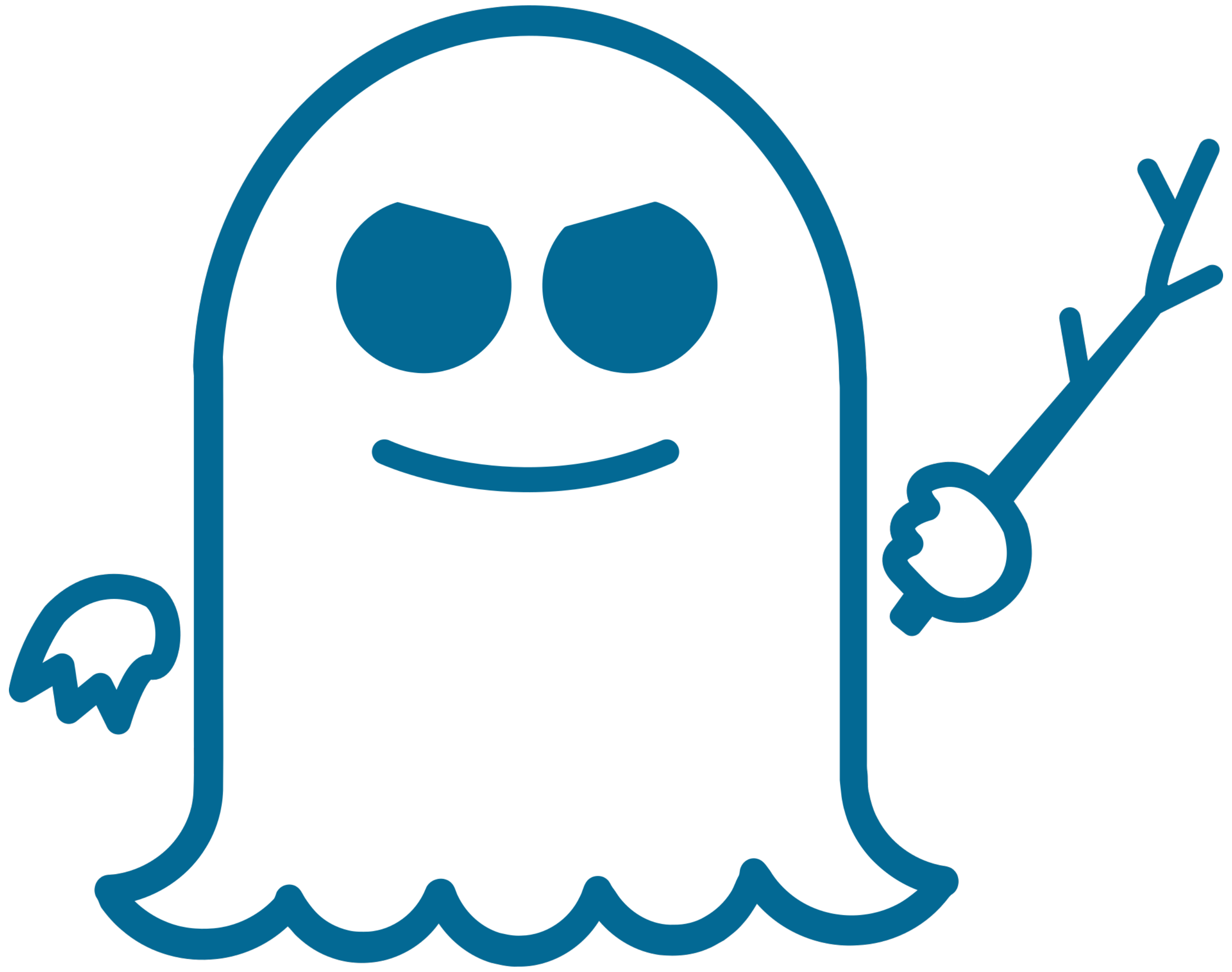
- RISC-V cores have **unprivileged** performance counters

- Performance counter **attacks are again possible**!

- More performance counters **will yield stronger attacks...**

CPU Design: Spectre





CPU Optimization: Branch-Prediction-Unit (BPU)

```
if(secret){
    A();
}else{
    B();
}
```

CPU Optimization: Branch-Prediction-Unit (BPU)

- **Branches** impact execution speed

```
if(secret){
    A();
}else{
    B();
}
```

CPU Optimization: Branch-Prediction-Unit (BPU)

- **Branches** impact execution speed
  - The CPU has to **wait for Branches**

```
if(secret){
    A();
}else{
    B();
}
```

CPU Optimization: Branch-Prediction-Unit (BPU)

- Branches impact execution speed
  - The CPU has to **wait for Branches**
  - Stalling slows down execution

```
if(secret){
    A();
}else{
    B();
}
```

CPU Optimization: Branch-Prediction-Unit (BPU)

- Branches impact execution speed
  - The CPU has to **wait for Branches**
  - Stalling slows down execution

→ **Optimize by Prediction**

```
if(secret){
    A();
}else{
    B();
}
```

CPU Optimization: Branch-Prediction-Unit (BPU)

- Branches impact execution speed
  - The CPU has to **wait for Branches**
  - Stalling slows down execution

Optimize by Prediction

- Look at history of last branches

```
if(secret){
    A();
}else{
    B();
}
```

CPU Optimization: Branch-Prediction-Unit (BPU)

- Branches impact execution speed
  - The CPU has to **wait for Branches**
  - Stalling slows down execution

→ **Optimize by Prediction**

- Look at history of last branches
- **Predict** which **branch direction** is taken next

```
if(secret){
    A();
}else{
    B();
}
```

CPU Optimization: Branch-Prediction-Unit (BPU)

- Branches impact execution speed
  - The CPU has to **wait for Branches**
  - Stalling slows down execution

Optimize by Prediction

- Look at history of last branches
- **Predict** which **branch direction** is taken next

```
if(secret){
    A();
}else{
    B();
}
```

CPU Optimization: Branch-Prediction-Unit (BPU)

- Branches impact execution speed
  - The CPU has to **wait for Branches**
  - Stalling slows down execution

Optimize by Prediction

- Look at history of last branches
- **Predict** which **branch direction** is taken next

```
if(secret){
    A();
}else{
    B();
}
```

CPU Optimization: Speculative Execution

CPU Optimization: Speculative Execution

- Why stop at predicting the branch?

CPU Optimization: Speculative Execution

- Why stop at predicting the branch?
- Instead **execute** the **prediction**

CPU Optimization: Speculative Execution

- Why stop at predicting the branch?
- Instead **execute** the **prediction**
- Two cases:

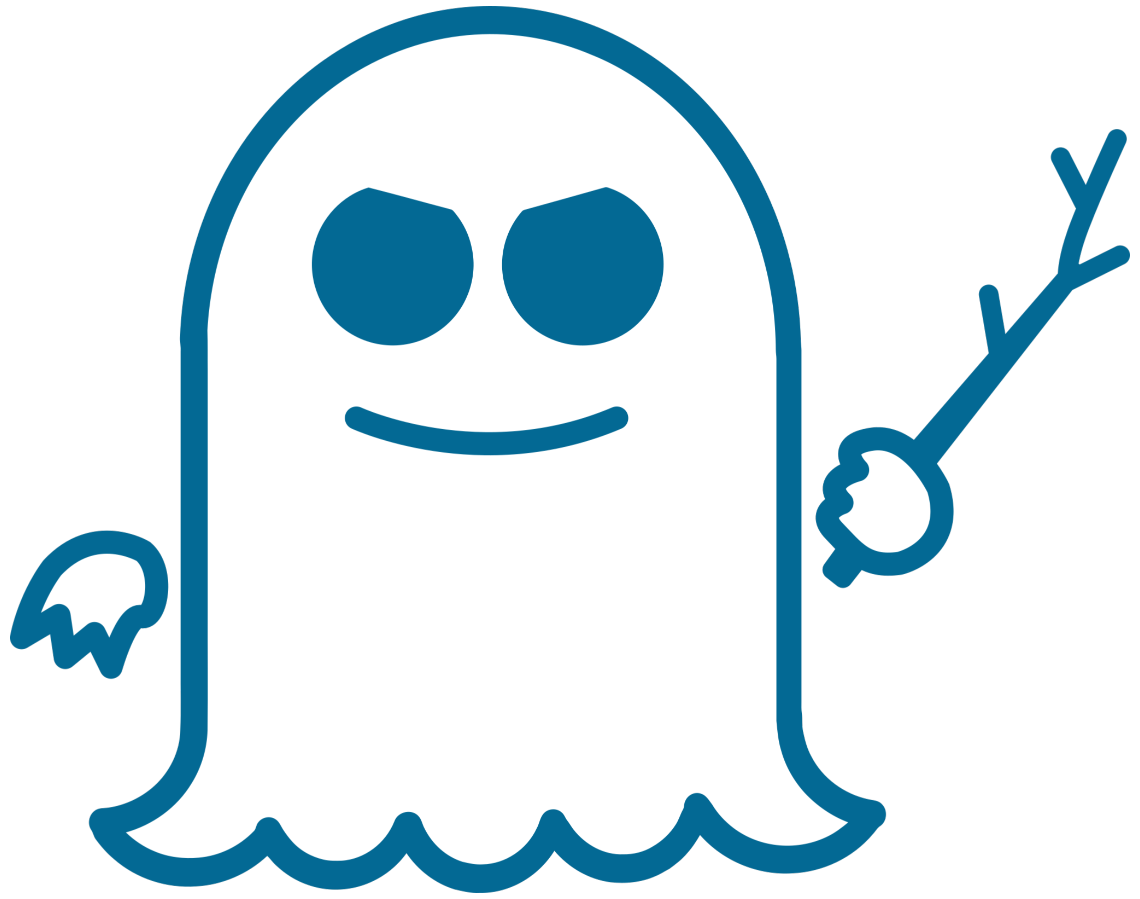
CPU Optimization: Speculative Execution

- Why stop at predicting the branch?
- Instead **execute** the **prediction**
- Two cases:
  - Correct prediction, we win

CPU Optimization: Speculative Execution

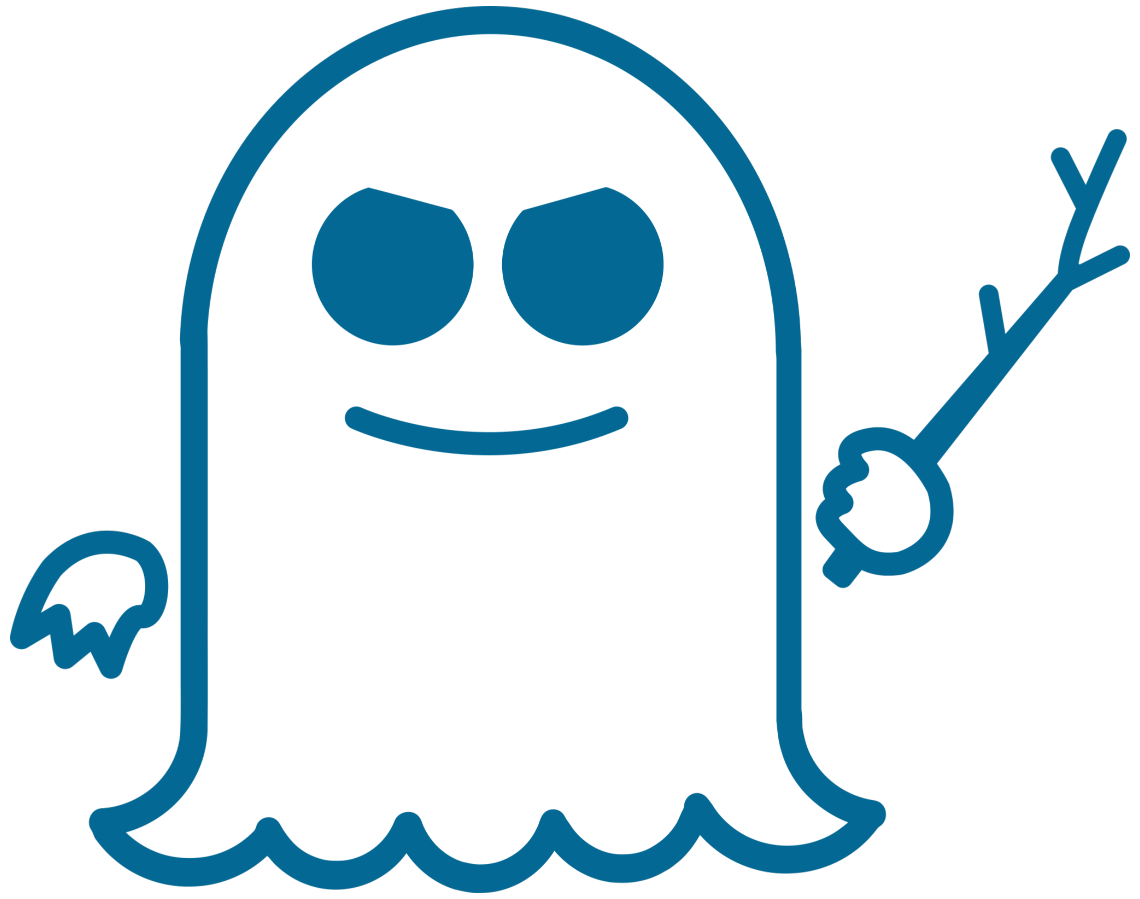
- Why stop at predicting the branch?
- Instead **execute** the **prediction**
- Two cases:
  - Correct prediction, we win
  - False prediction, **rollback** effects of branch

Spectre?



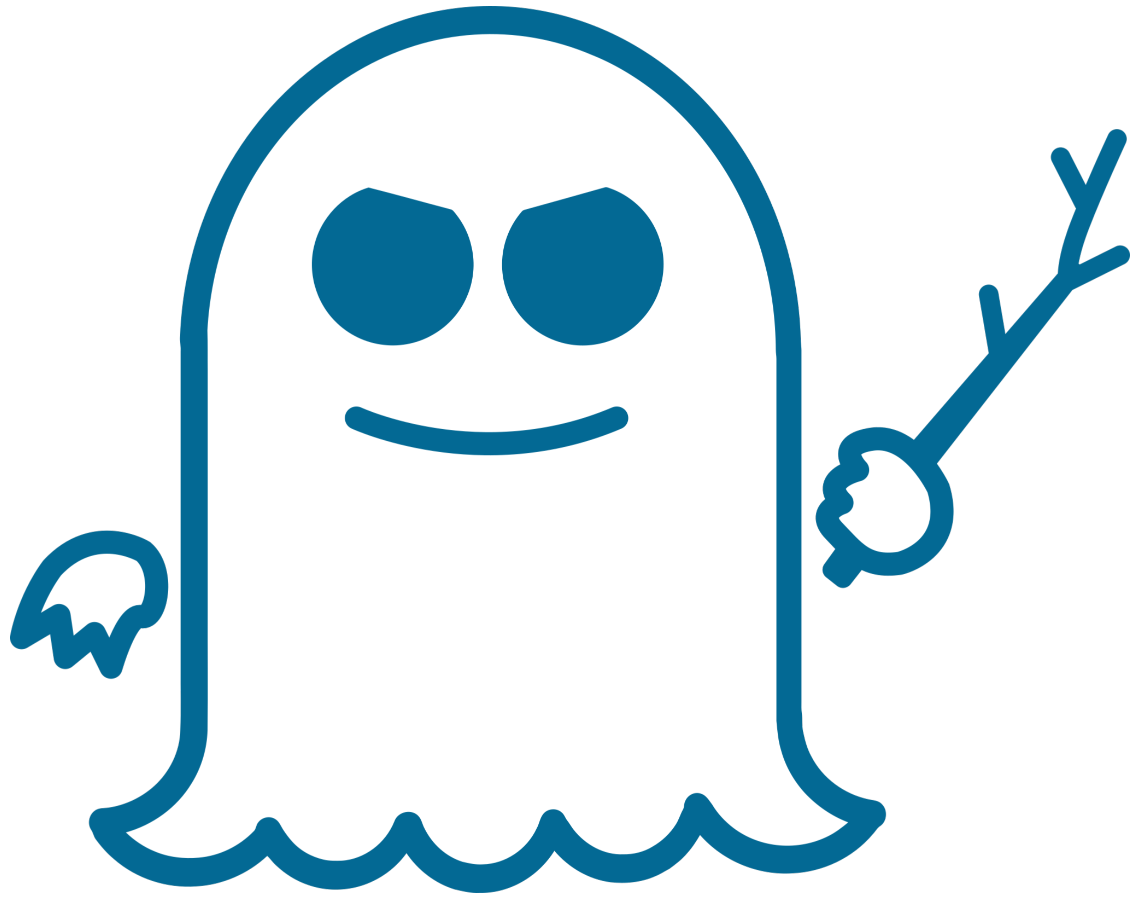
Spectre?

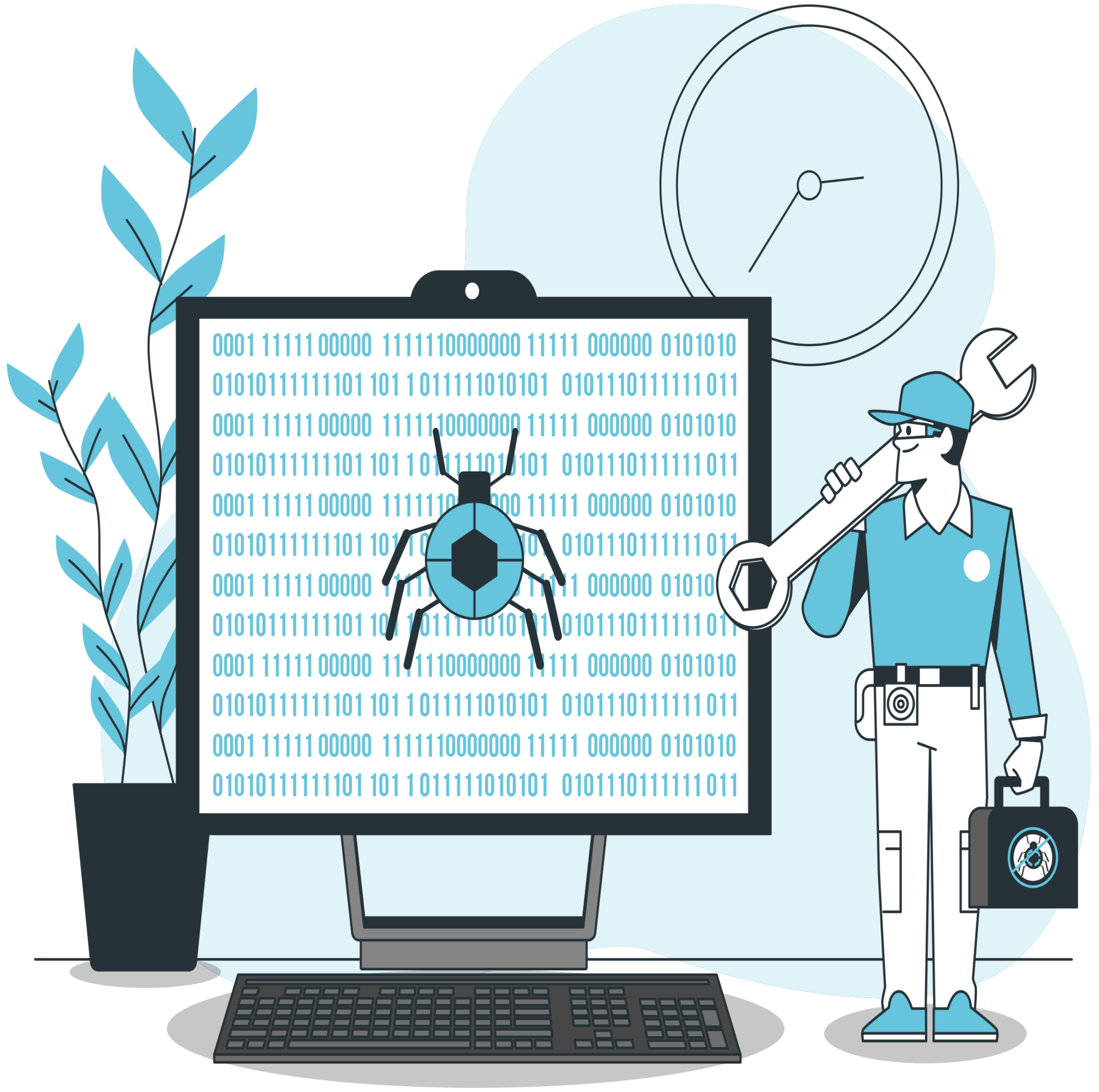
- Spectre **requires speculative execution**



Spectre?

- Spectre **requires speculative execution**
- **Our RISC-V CPUs:** No support for speculative execution **(yet...)**





But there is speculative prefetching!

Cache+Time

```
if(secret){
    A();
}else{
    B();
}
```

Victim

Cache+Time

- New **side channel** on the **instruction cache**

```
if(secret){
    A();
}else{
    B();
}
```

Victim

Cache+Time

- New **side channel** on the **instruction cache**
  - No shared memory

```
if(secret){
    A();
}else{
    B();
}
```

Victim

Cache+Time

- New **side channel** on the **instruction cache**
  - No shared memory
  - Cache-line granularity

```
if(secret){
    A();
}else{
    B();
}
```

Victim

Cache+Time

- New **side channel** on the **instruction cache**
  - No shared memory
  - Cache-line granularity

```
if(secret){
    A();
}else{
    B();
}
```

fence.i

Victim

Cache+Time

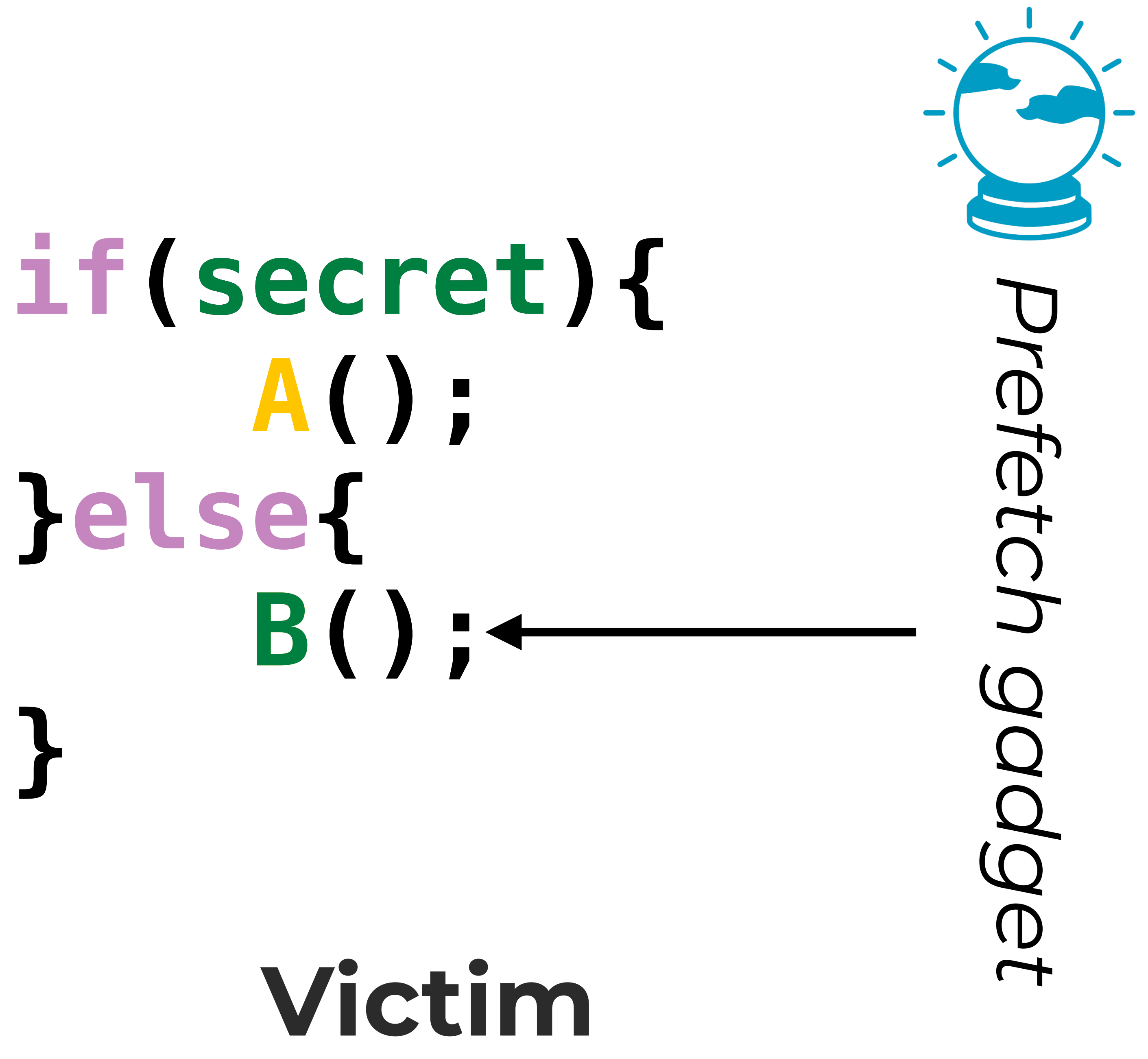
- New **side channel** on the **instruction cache**
  - No shared memory
  - Cache-line granularity

```
if(secret){
    A();
}else{
    B();
}
```

Victim

Cache+Time

- New **side channel** on the **instruction cache**
  - No shared memory
  - Cache-line granularity



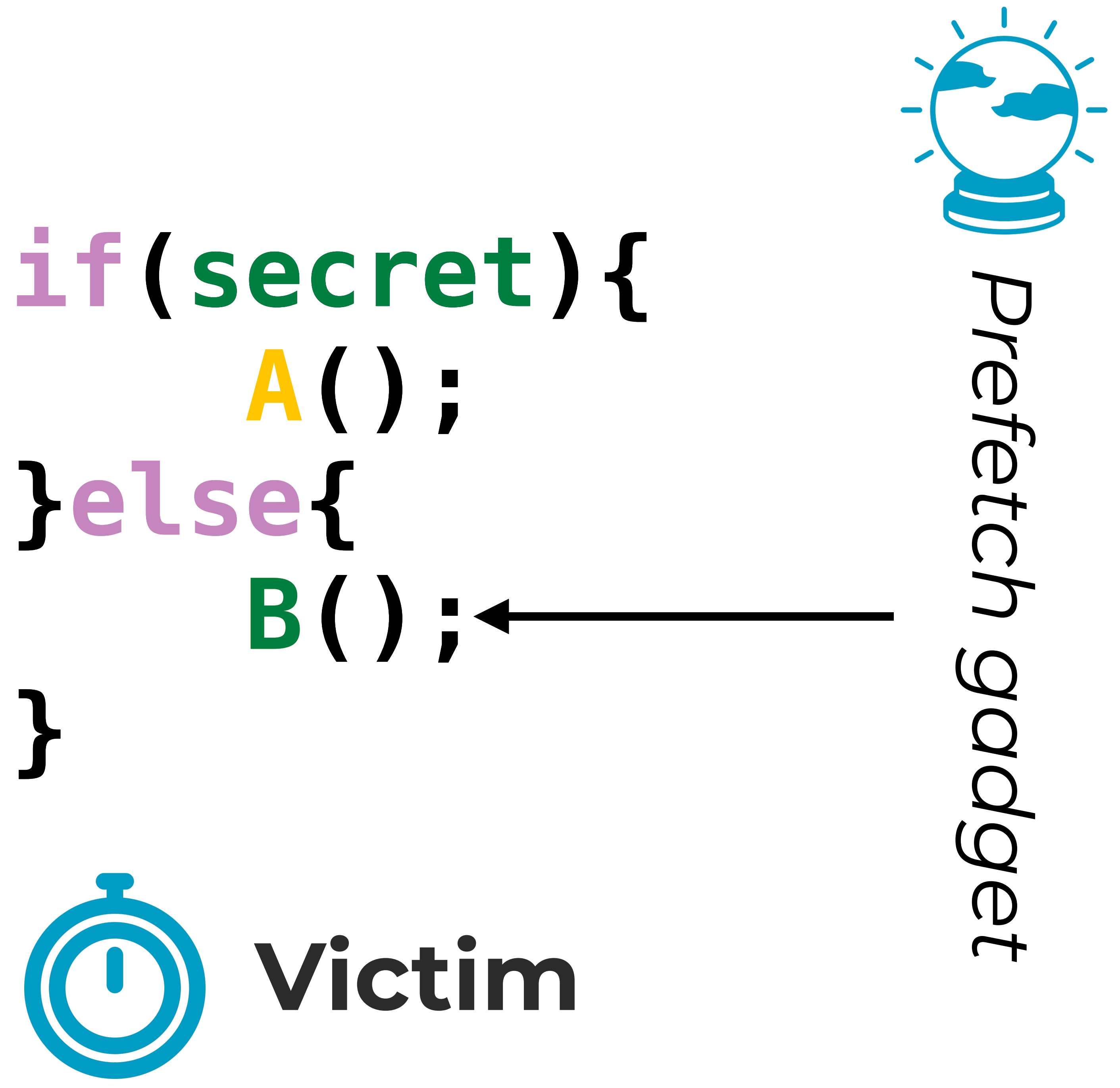
```
if(secret){
    A();
}else{
    B();
}
```

Victim

Prefetch gadget

Cache+Time

- New **side channel** on the **instruction cache**
  - No shared memory
  - Cache-line granularity

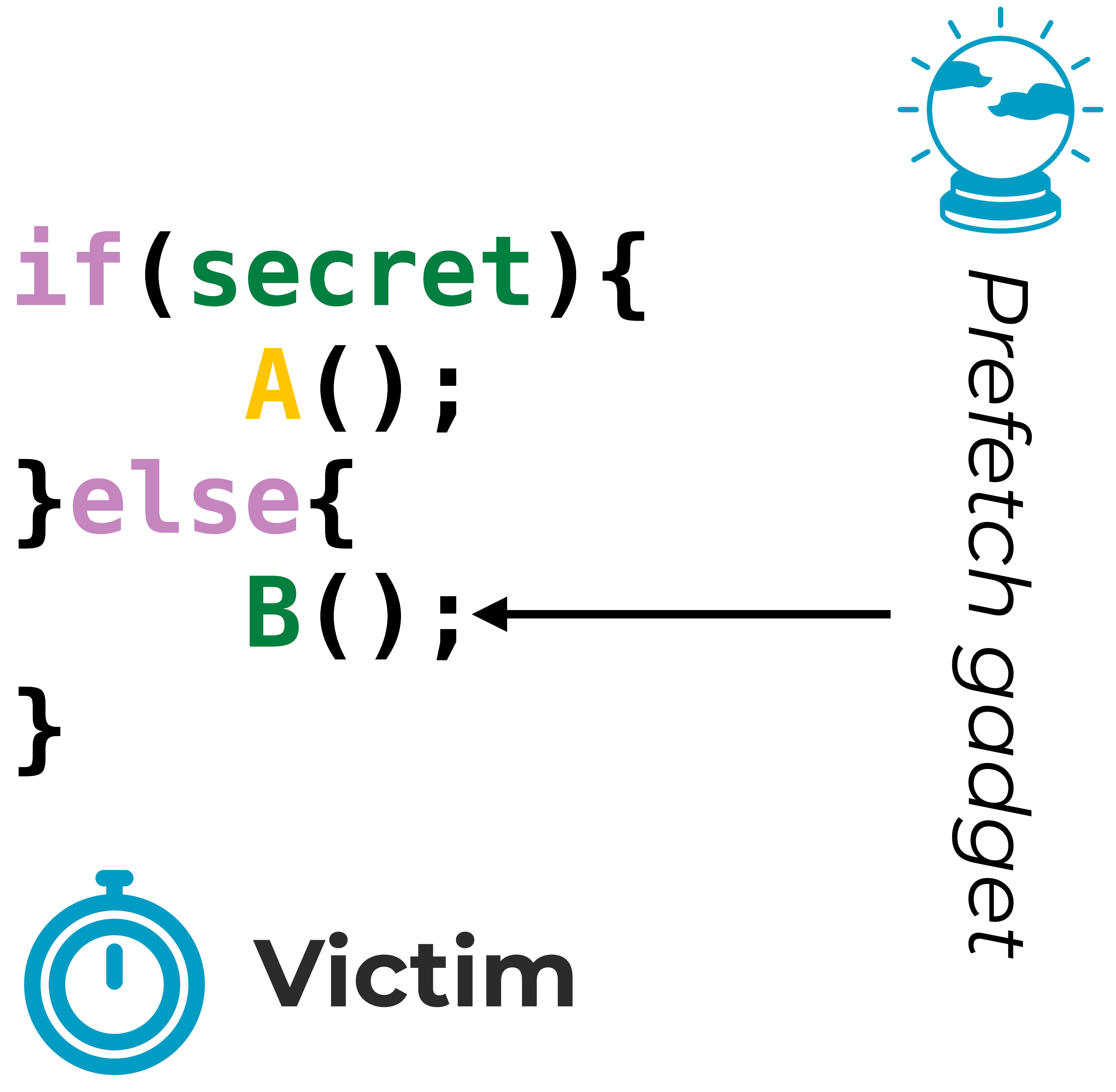


```
if(secret){
    A();
}else{
    B();
}
```

Cache+Time

- New **side channel** on the **instruction cache**
  - No shared memory
  - Cache-line granularity

➡ **Speculative prefetching is exploitable**



```
if(secret){
    A();
}else{
    B();
}
```

Surpise Demo: Spectre is fixed? Right?

beagle@lab46: ~/specre-riscv

⌥⌘1

```
beagle@lab46:~/specre-riscv$ .
```

Lesson Learned: BPU Attacks Possible. Spectre also.

Lesson Learned: BPU Attacks Possible. Spectre also.

- The limited speculation on C906 and U74 **mitigates well-known attacks**

Lesson Learned: BPU Attacks Possible. Spectre also.

- The limited speculation on C906 and U74 **mitigates well-known attacks**
  - e.g., Spectre

Lesson Learned: BPU Attacks Possible. Spectre also.

- The limited speculation on C906 and U74 **mitigates well-known attacks**
  - e.g., Spectre

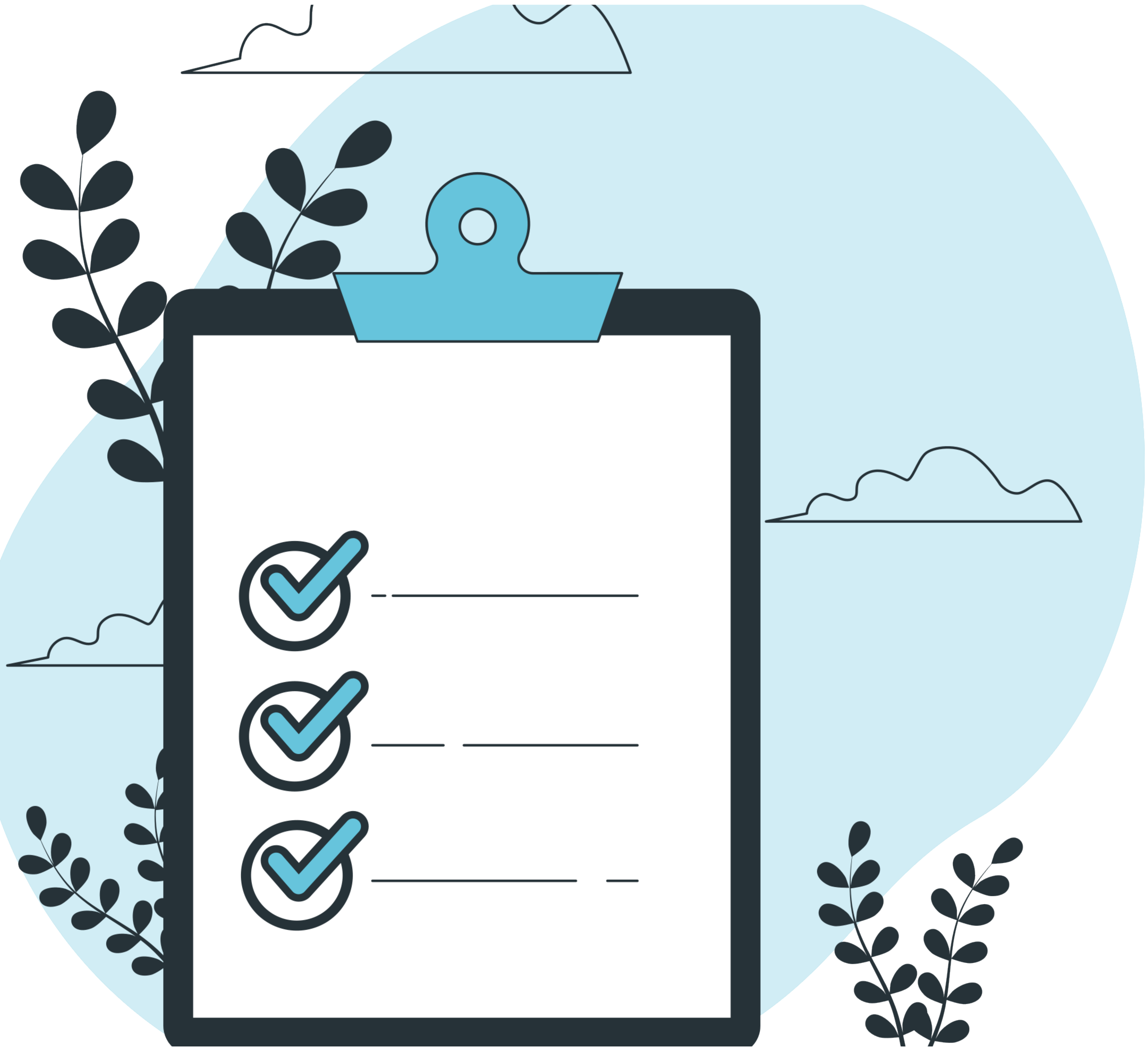
- Even **limited speculation** allows for **powerful attacks**

Lesson Learned: BPU Attacks Possible. Spectre also.

- The limited speculation on C906 and U74 **mitigates well-known attacks**
  - e.g., Spectre

- Even **limited speculation** allows for **powerful attacks**

➡ **More optimized cores (C910) are more vulnerable**



Conclusion

Lessons learned: Summary

Lessons learned: Summary

- **Open-source architectures are great!**

Lessons learned: Summary

- **Open-source architectures are great!**
  - Allow for **white-box bug hunting**

Lessons learned: Summary

- **Open-source architectures are great!**
  - Allow for **white-box bug hunting**
  - Eases proposing **and testing of defenses**

Lessons learned: Summary

- **Open-source architectures are great!**
  - Allow for **white-box bug hunting**
  - Eases proposing **and testing of defenses**

- RISC-V hardware comes with **similar vulnerabilities** as **other architectures**

Lessons learned: Summary

- **Open-source architectures are great!**
  - Allow for **white-box bug hunting**
  - Eases proposing **and testing of defenses**

- RISC-V hardware comes with **similar vulnerabilities** as **other architectures**
  - Cache attacks

Lessons learned: Summary

- **Open-source architectures are great!**
  - Allow for **white-box bug hunting**
  - Eases proposing **and testing of defenses**

- RISC-V hardware comes with **similar vulnerabilities** as **other architectures**
  - Cache attacks
  - Prediction-based attacks

Lessons learned: Summary

- **Open-source architectures are great!**
  - Allow for **white-box bug hunting**
  - Eases proposing **and testing of defenses**

- RISC-V hardware comes with **similar vulnerabilities** as **other architectures**
  - Cache attacks
  - Prediction-based attacks
  - Transient-execution attacks

Lessons learned: Summary

- **Open-source architectures are great!**
  - Allow for **white-box bug hunting**
  - Eases proposing **and testing of defenses**

- RISC-V hardware comes with **similar vulnerabilities** as **other architectures**
  - Cache attacks
  - Prediction-based attacks
  - Transient-execution attacks

Lessons learned: Summary

- **Open-source architectures are great!**
  - Allow for **white-box bug hunting**
  - Eases proposing **and testing of defenses**

- RISC-V hardware comes with **similar vulnerabilities** as **other architectures**
  - Cache attacks
  - Prediction-based attacks
  - Transient-execution attacks

- **“Surprising” design decision:**

Lessons learned: Summary

- **Open-source architectures are great!**
  - Allow for **white-box bug hunting**
  - Eases proposing **and testing of defenses**

- RISC-V hardware comes with **similar vulnerabilities** as **other architectures**
  - Cache attacks
  - Prediction-based attacks
  - Transient-execution attacks

- **“Surprising” design decision:**
  - Unprivileged performance counters

Want to Play Around with the Code?

https://github.com/cispa/Security-RISC

Takeaways

Takeaways

→ RISC-V has **a lot of potential...**

Takeaways

→ RISC-V has **a lot of potential...**

→ **Redesigned open-source** architectures do **not automagically solve security**

Takeaways

→ RISC-V has **a lot of potential...**

→ **Redesigned open-source** architectures do **not automagically solve security**

More (optimized) RISC-V cores on the way

@____salmon____ @weber_daniel

Takeaways

→ RISC-V has **a lot of potential...**

→ **Redesigned open-source** architectures do **not automagically solve security**

More (optimized) RISC-V cores on the way

↓

Security Research on RISC-V hardware **is essential!**

@____salmon____ @weber_daniel

CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

A Security RISC?

The State of Microarchitectural Attacks on RISC-V

Lukas Gerlach, **Daniel Weber**, Michael Schwarz | BlackHat EU 2023