SpecShield: Shielding Speculative Data from Microarchitectural Covert Channels

Kristin Barber, Anys Bacha*, Li Zhou, Yinqian Zhang, Radu Teodorescu

Department of Computer Science and Engineering

The Ohio State University

http://arch.cse.ohio-state.edu

*The University of Michigan

THE OHIO STATE UNIVERSITY

UNIVERSITY OF MICHIGAN

COMPUTER ARCHITECTURE RESEARCH LAB

Motivation

Motivation

- In Jan. 2018, a new class of vulnerabilities was unveiled

Motivation

- In Jan. 2018, a new class of vulnerabilities was unveiled
  - Ability to bypass isolation boundaries and violate control-flow integrity

Motivation

- In Jan. 2018, a new class of vulnerabilities was unveiled
    - Ability to bypass isolation boundaries and violate control-flow integrity
    - Manipulation of speculative mechanisms to perform computation of interest to attacker

Motivation

- In Jan. 2018, a new class of vulnerabilities was unveiled
  - Ability to bypass isolation boundaries and violate control-flow integrity
  - Manipulation of speculative mechanisms to perform computation of interest to attacker
- Meltdown demonstrated reading entire kernel memory from user process!

Motivation

- In Jan. 2018, a new class of vulnerabilities was unveiled
  - Ability to bypass isolation boundaries and violate control-flow integrity
  - Manipulation of speculative mechanisms to perform computation of interest to attacker
- Meltdown demonstrated reading entire kernel memory from user process!
  - Indicates how powerful these attacks can be

Motivation

- In Jan. 2018, a new class of vulnerabilities was unveiled
  - Ability to bypass isolation boundaries and violate control-flow integrity
  - Manipulation of speculative mechanisms to perform computation of interest to attacker
- Meltdown demonstrated reading entire kernel memory from user process!
  - Indicates how powerful these attacks can be
  - Process isolation, sandboxed environments, virtualized environments are all susceptible

Motivation

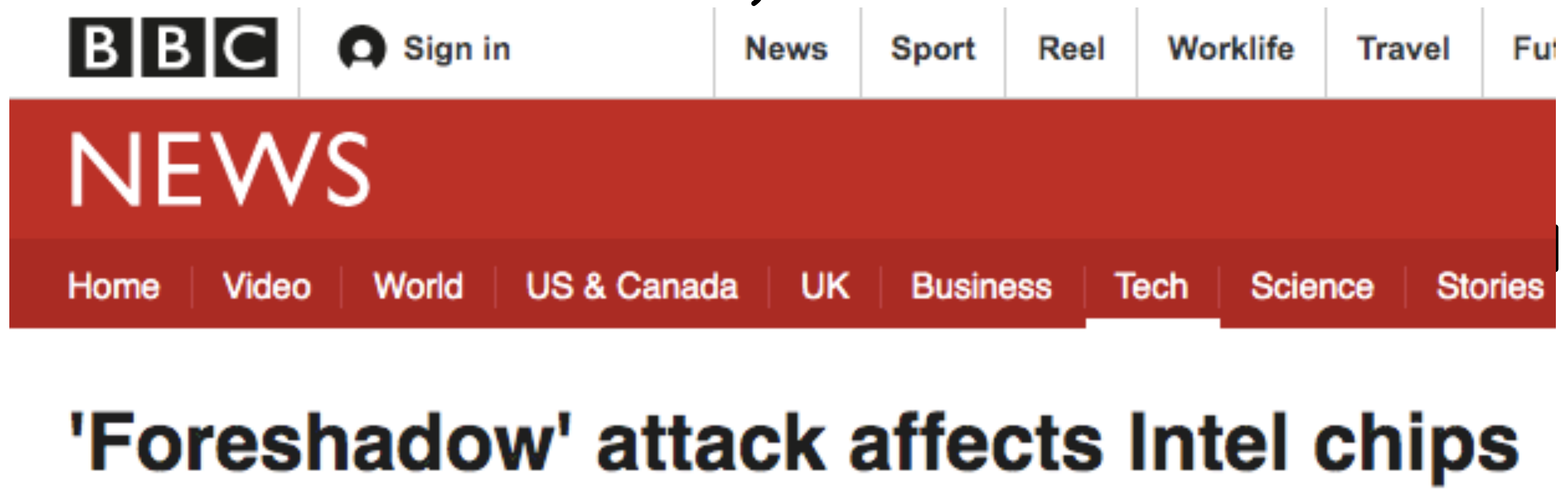
- In Jan. 2018, a new class of vulnerabilities was unveiled
  - Ability to bypass isolation boundaries and violate control-flow integrity
  - Manipulation of speculative mechanisms to perform computation of interest to attacker
- Meltdown demonstrated reading entire kernel memory from user process!
  - Indicates how powerful these attacks can be
  - Process isolation, sandboxed environments, virtualized environments are all susceptible
  - Don’t rely on an implementation “bug”, use features correctly as intended

Motivation

- In Jan. 2018, a new class of vulnerabilities was unveiled
  - Ability to bypass isolation boundaries and violate control-flow integrity
  - Manipulation of speculative mechanisms to perform computation of interest to attacker
- Meltdown demonstrated reading entire kernel memory from user process!
  - Indicates how powerful these attacks can be
  - Process isolation, sandboxed environments, virtualized environments are all susceptible
  - Don’t rely on an implementation “bug”, use features correctly as intended
- The past 1.5 years has seen a wide range of attacks with variations on this theme

Motivation

- In Jan. 2018, a new class of vulnerabilities was unveiled



'Foreshadow' attack affects Intel chips

ndaries and violate control-flow integrity

  - Manipulation of speculative mechanisms to perform computation of interest to attacker

- Meltdown demonstrated reading entire kernel memory from user process!
  - Indicates how powerful these attacks can be
  - Process isolation, sandboxed environments, virtualized environments are all susceptible
  - Don’t rely on an implementation “bug”, use features correctly as intended
- The past 1.5 years has seen a wide range of attacks with variations on this theme

Motivation

- In Jan. 2018, a new class of vulne

BBC Sign in News Sport Reel Worklife Travel Fu

NEWS

Home Video World US & Canada UK Business Tech Science Stories

'Foreshadow' attack affects Intel chips

c|net BEST PRODUCTS REVIEWS NEWS VIDEO HOW TO SMART HOME CARS DEALS DOWN

SECURITY | LEER EN ESPAÑOL

Spectre and Meltdown: Deta on those big chip fl

nda

- Manipulation of speculative mech

- Meltdown demonstrated reading entire kernel memory from user process!
  - Indicates how powerful these attacks can be
  - Process isolation, sandboxed environments, virtualized environments are all susceptible
  - Don’t rely on an implementation “bug”, use features correctly as intended
- The past 1.5 years has seen a wide range of attacks with variations on this theme

Motivation

- In Jan. 2018, a new class of vulne

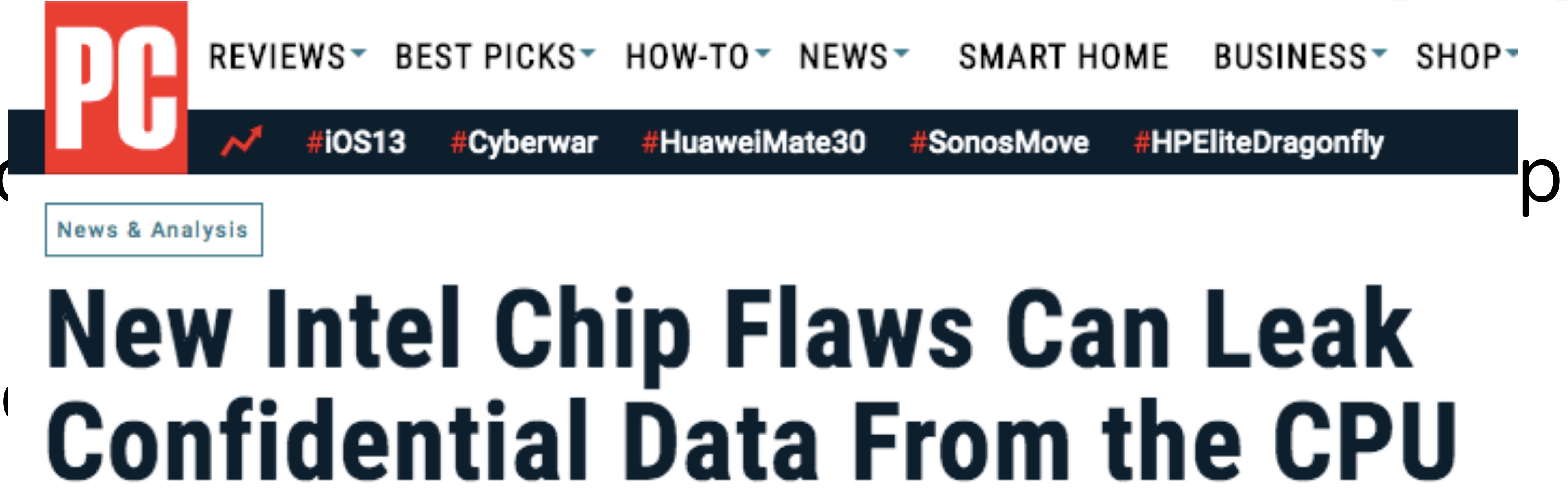
nda

'Foreshadow' attack affects Intel chips

Spectre and Meltdown: Deta
on those big chip fl

  - Manipulation of speculative mech

er



New Intel Chip Flaws Can Leak Confidential Data From the CPU

- Meltdown demonstrated process!
  - Indicates how pow
  - Process isolation, sandboxed environments, virtualized environments are all susceptible
  - Don’t rely on an implementation “bug”, use features correctly as intended
- The past 1.5 years has seen a wide range of attacks with variations on this theme

Motivation

- In Jan. 2018, a new class of vulne

nda

  - Manipulation of speculative mech

process!

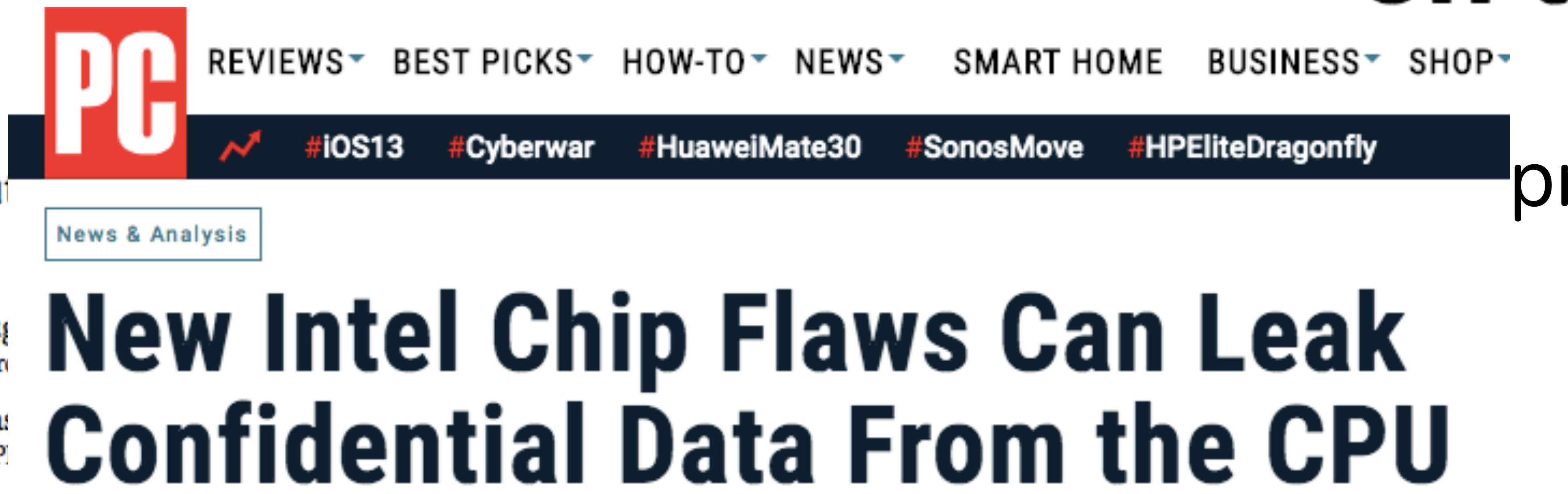
oxed environments, virtualized environments are all susceptible

  - Don't rely on an implementation "bug", use features correctly as intended

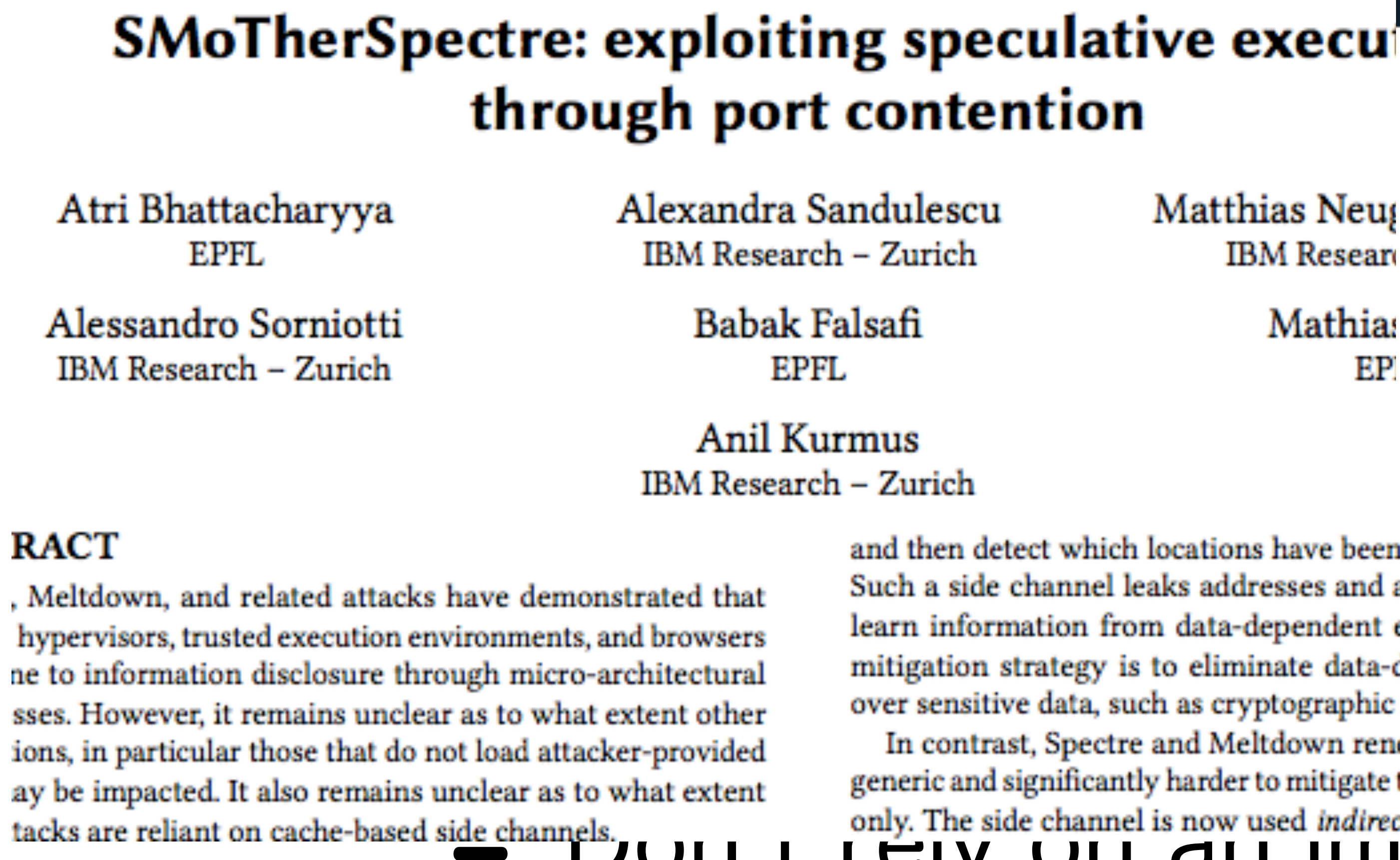
- The past 1.5 years has seen a wide range of attacks with variations on this theme

'Foreshadow' attack affects Intel chips

Spectre and Meltdown: Deta
on those big chip fl



New Intel Chip Flaws Can Leak Confidential Data From the CPU



SMoTherSpectre: exploiting speculative execu
through port contention

Atri Bhattacharyya
EPFL

Alexandra Sandulescu
IBM Research – Zurich

Matthias Neu
IBM Resear

Alessandro Sorniotti
IBM Research – Zurich

Babak Falsafi
EPFL

Mathia
EP

Anil Kurmus
IBM Research – Zurich

RACT

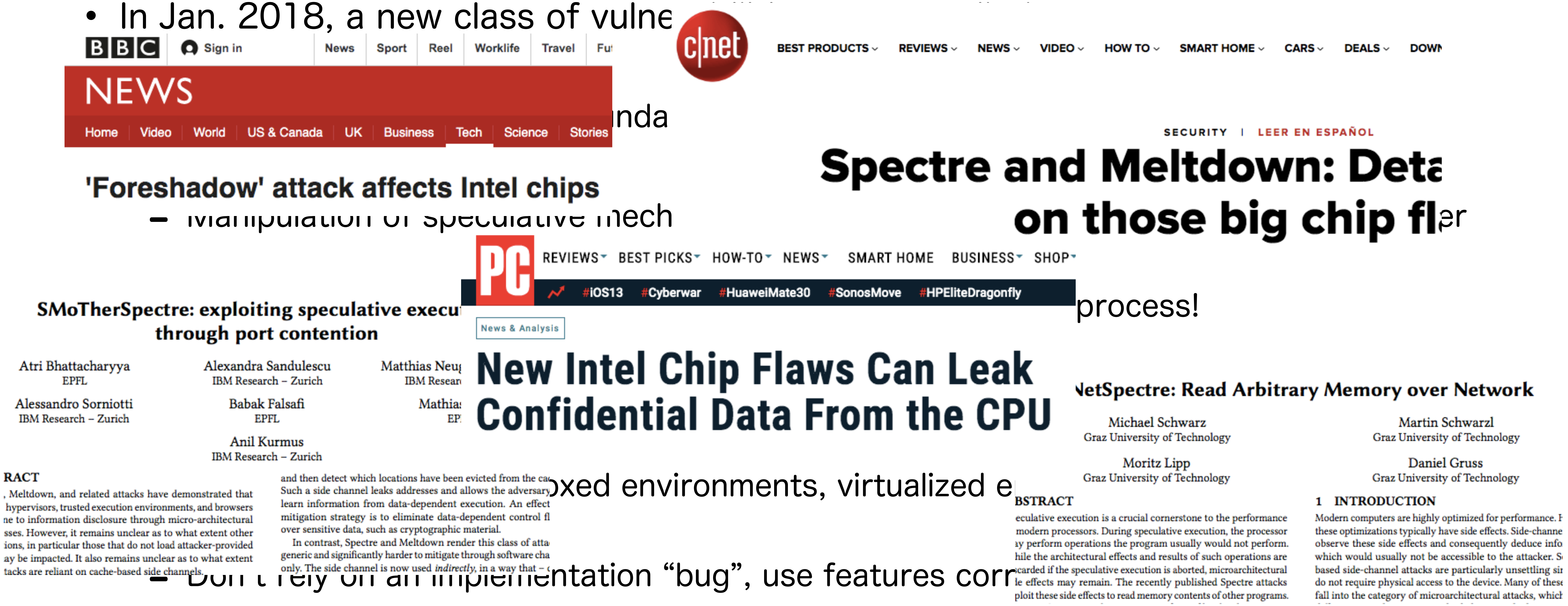
, Meltdown, and related attacks have demonstrated that hypervisors, trusted execution environments, and browsers ne to information disclosure through micro-architectural sses. However, it remains unclear as to what extent other ions, in particular those that do not load attacker-provided ay be impacted. It also remains unclear as to what extent tacks are reliant on cache-based side channels.

and then detect which locations have been evicted from the ca Such a side channel leaks addresses and allows the adversary learn information from data-dependent execution. An effect mitigation strategy is to eliminate data-dependent control fl over sensitive data, such as cryptographic material.

In contrast, Spectre and Meltdown render this class of atta generic and significantly harder to mitigate through software cha only. The side channel is now used *indirectly*, in a way that –

Motivation

- In Jan. 2018, a new class of vulne
  nda
  - Manipulation of speculative mech
  process!
  xed environments, virtualized e
  - Don't rely on an implementation "bug", use features corr



Sign in News Sport Reel Worklife Travel Fu

NEWS

Home Video World US & Canada UK Business Tech Science Stories

'Foreshadow' attack affects Intel chips

BEST PRODUCTS REVIEWS NEWS VIDEO HOW TO SMART HOME CARS DEALS DOWN

SECURITY | LEER EN ESPAÑOL

Spectre and Meltdown: Deta on those big chip fl

REVIEWS BEST PICKS HOW-TO NEWS SMART HOME BUSINESS SHOP

#iOS13 #Cyberwar #HuaweiMate30 #SonosMove #HPEliteDragonfly

News & Analysis

New Intel Chip Flaws Can Leak Confidential Data From the CPU

SMoTherSpectre: exploiting speculative execu through port contention

Atri Bhattacharyya, EPFL; Alexandra Sandulescu, IBM Research – Zurich; Matthias Neu, IBM Resear; Alessandro Sorniotti, IBM Research – Zurich; Babak Falsafi, EPFL; Mathia, EP; Anil Kurmus, IBM Research – Zurich

RACT

, Meltdown, and related attacks have demonstrated that hypervisors, trusted execution environments, and browsers ne to information disclosure through micro-architectural sses. However, it remains unclear as to what extent other ions, in particular those that do not load attacker-provided ay be impacted. It also remains unclear as to what extent tacks are reliant on cache-based side channels.

and then detect which locations have been evicted from the ca Such a side channel leaks addresses and allows the adversary learn information from data-dependent execution. An effect mitigation strategy is to eliminate data-dependent control fl over sensitive data, such as cryptographic material.

In contrast, Spectre and Meltdown render this class of atta generic and significantly harder to mitigate through software cha only. The side channel is now used *indirectly*, in a way that –

NetSpectre: Read Arbitrary Memory over Network

Michael Schwarz, Graz University of Technology; Martin Schwarzl, Graz University of Technology; Moritz Lipp, Graz University of Technology; Daniel Gruss, Graz University of Technology

BSTRACT

eculative execution is a crucial cornerstone to the performance modern processors. During speculative execution, the processor ay perform operations the program usually would not perform. hile the architectural effects and results of such operations are carded if the speculative execution is aborted, microarchitectural le effects may remain. The recently published Spectre attacks ploit these side effects to read memory contents of other programs.

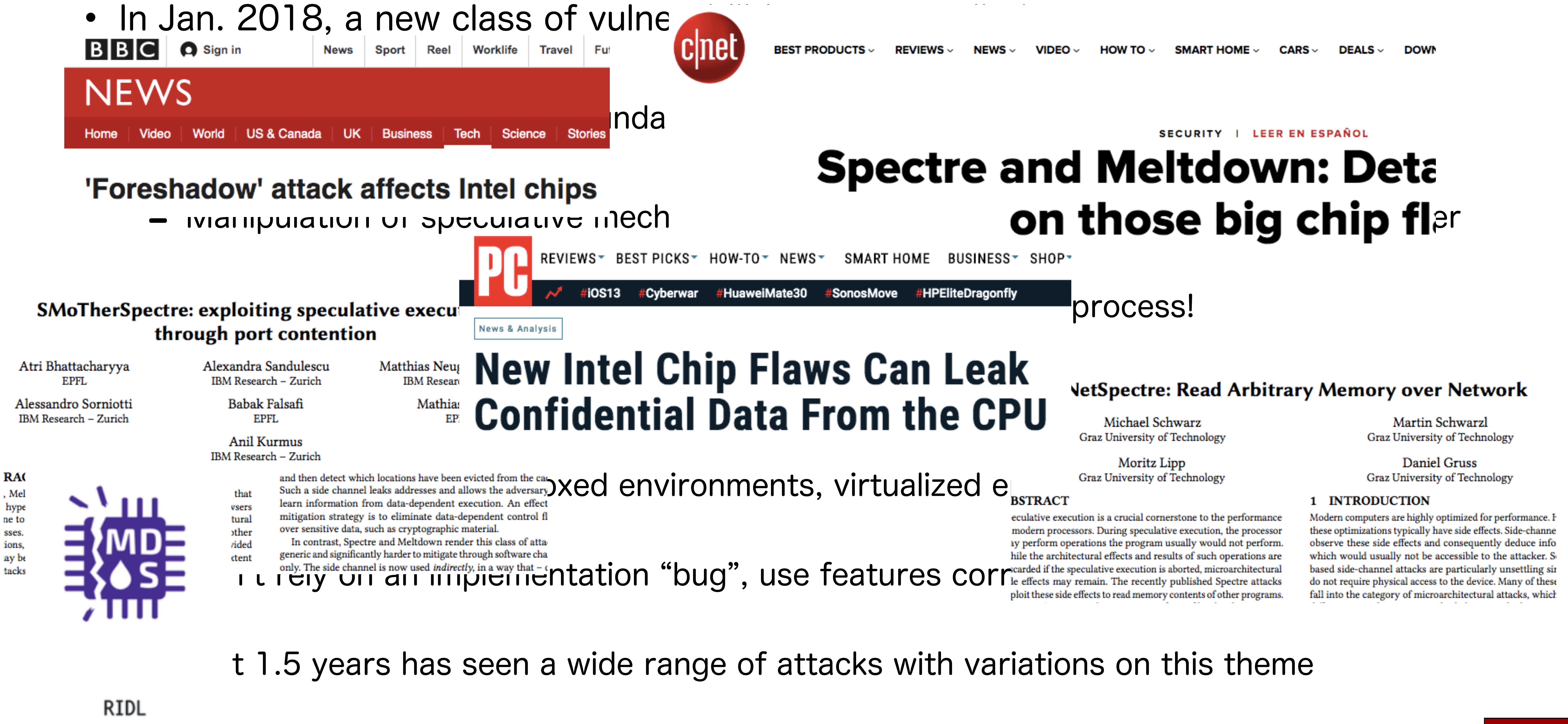
1 INTRODUCTION

Modern computers are highly optimized for performance. F these optimizations typically have side effects. Side-channe observe these side effects and consequently deduce info which would usually not be accessible to the attacker. S based side-channel attacks are particularly unsettling sir do not require physical access to the device. Many of these fall into the category of microarchitectural attacks, which

- The past 1.5 years has seen a wide range of attacks with variations on this theme

Motivation

- In Jan. 2018, a new class of vulne
  nda
  - Manipulation of speculative mech
  process!
  xed environments, virtualized e
  t rely on an implementation “bug”, use features corr
  t 1.5 years has seen a wide range of attacks with variations on this theme



BBC Sign in News Sport Reel Worklife Travel Fu

NEWS

Home Video World US & Canada UK Business Tech Science Stories

'Foreshadow' attack affects Intel chips

c|net BEST PRODUCTS REVIEWS NEWS VIDEO HOW TO SMART HOME CARS DEALS DOWN

SECURITY | LEER EN ESPAÑOL

Spectre and Meltdown: Deta
on those big chip fl

PC REVIEWS BEST PICKS HOW-TO NEWS SMART HOME BUSINESS SHOP

#iOS13 #Cyberwar #HuaweiMate30 #SonosMove #HPEliteDragonfly

News & Analysis

New Intel Chip Flaws Can Leak Confidential Data From the CPU

SMoTherSpectre: exploiting speculative execu
through port contention

Atri Bhattacharyya, EPFL; Alexandra Sandulescu, IBM Research – Zurich; Matthias Neu, IBM Resear; Alessandro Sorniotti, IBM Research – Zurich; Babak Falsafi, EPFL; Mathias, EP; Anil Kurmus, IBM Research – Zurich

and then detect which locations have been evicted from the ca
Such a side channel leaks addresses and allows the adversary
learn information from data-dependent execution. An effect
mitigation strategy is to eliminate data-dependent control fl
over sensitive data, such as cryptographic material.
In contrast, Spectre and Meltdown render this class of atta
generic and significantly harder to mitigate through software cha
only. The side channel is now used *indirectly*, in a way that –

MDS

RIDL

NetSpectre: Read Arbitrary Memory over Network

Michael Schwarz, Graz University of Technology; Martin Schwarzl, Graz University of Technology; Moritz Lipp, Graz University of Technology; Daniel Gruss, Graz University of Technology

BSTRACT

eculative execution is a crucial cornerstone to the performance
modern processors. During speculative execution, the processor
ay perform operations the program usually would not perform.
hile the architectural effects and results of such operations are
carded if the speculative execution is aborted, microarchitectural
le effects may remain. The recently published Spectre attacks
ploit these side effects to read memory contents of other programs.

1 INTRODUCTION

Modern computers are highly optimized for performance. F
these optimizations typically have side effects. Side-channe
observe these side effects and consequently deduce info
which would usually not be accessible to the attacker. S
based side-channel attacks are particularly unsettling sin
do not require physical access to the device. Many of these
fall into the category of microarchitectural attacks, which

Motivation

- In Jan. 2018, a new class of vulne

nda

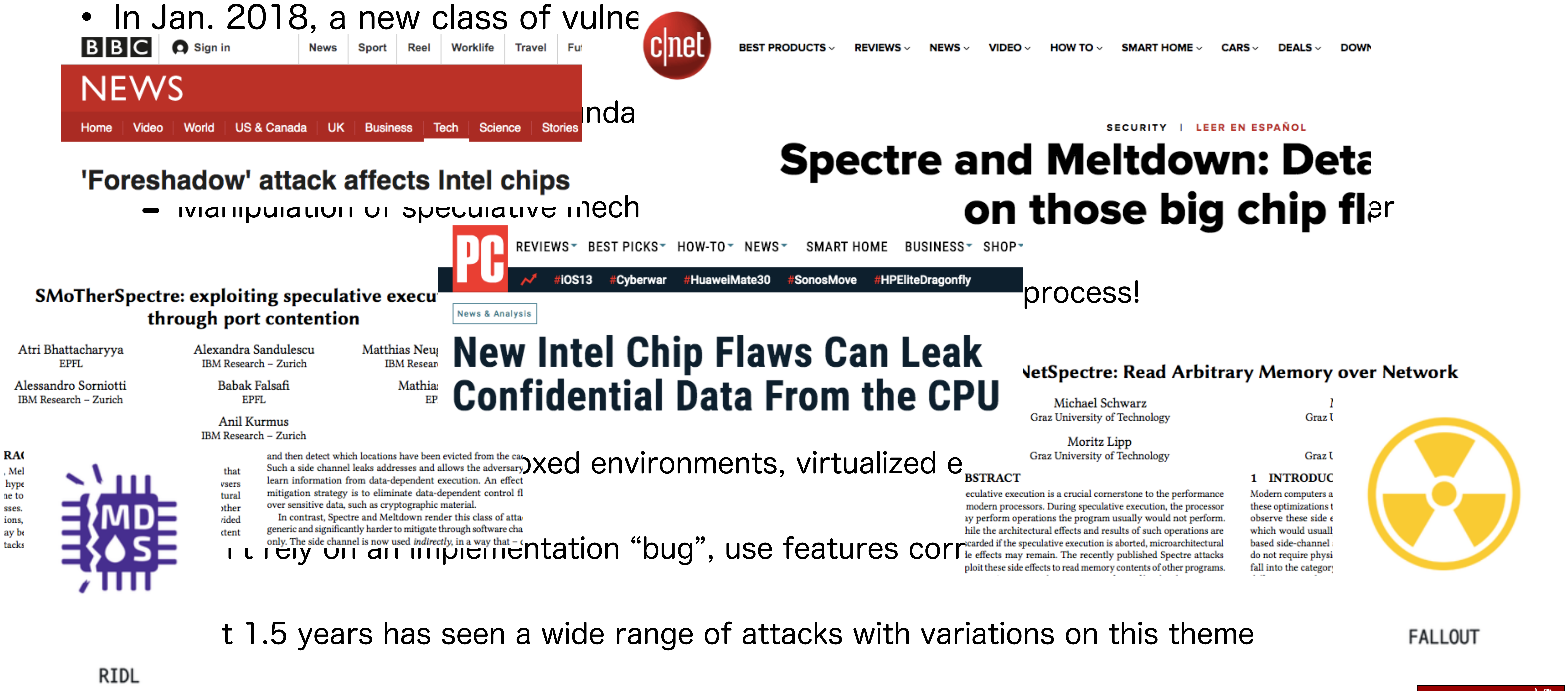
– Manipulation of speculative mech

process!

xed environments, virtualized e

t rely on an implementation "bug", use features corr

t 1.5 years has seen a wide range of attacks with variations on this theme



BBC Sign in News Sport Reel Worklife Travel Fu

NEWS

Home Video World US & Canada UK Business Tech Science Stories

'Foreshadow' attack affects Intel chips

BEST PRODUCTS REVIEWS NEWS VIDEO HOW TO SMART HOME CARS DEALS DOWN

SECURITY | LEER EN ESPAÑOL

Spectre and Meltdown: Deta on those big chip fl

REVIEWS BEST PICKS HOW-TO NEWS SMART HOME BUSINESS SHOP

#iOS13 #Cyberwar #HuaweiMate30 #SonosMove #HPEliteDragonfly

News & Analysis

New Intel Chip Flaws Can Leak Confidential Data From the CPU

SMoTherSpectre: exploiting speculative execu through port contention

Atri Bhattacharyya, EPFL · Alexandra Sandulescu, IBM Research – Zurich · Matthias Neu, IBM Resear · Alessandro Sorniotti, IBM Research – Zurich · Babak Falsafi, EPFL · Mathias, EP · Anil Kurmus, IBM Research – Zurich

and then detect which locations have been evicted from the ca Such a side channel leaks addresses and allows the adversary learn information from data-dependent execution. An effect mitigation strategy is to eliminate data-dependent control fl over sensitive data, such as cryptographic material.

In contrast, Spectre and Meltdown render this class of atta generic and significantly harder to mitigate through software cha only. The side channel is now used *indirectly*, in a way that –

NetSpectre: Read Arbitrary Memory over Network

Michael Schwarz, Graz University of Technology · Moritz Lipp, Graz University of Technology

BSTRACT

eculative execution is a crucial cornerstone to the performance modern processors. During speculative execution, the processor ay perform operations the program usually would not perform. hile the architectural effects and results of such operations are carded if the speculative execution is aborted, microarchitectural le effects may remain. The recently published Spectre attacks ploit these side effects to read memory contents of other programs.

1 INTRODUC

Modern computers a these optimizations t observe these side e which would usuall based side-channel do not require physi fall into the categor

Motivation

- In Jan. 2018, a new class of vulne

BBC Sign in News Sport Reel Worklife Travel Fu

NEWS

Home Video World US & Canada UK Business Tech Science Stories

'Foresha

nda

cnet BEST PRODUCTS REVIEWS NEWS VIDEO HOW TO SMART HOME CARS DEALS DOWN

SECURITY | LEER EN ESPAÑOL

Deta

p fl er

Transient Execution Attacks

SMoTherSpectre: thro

Atri Bhattacharyya
EPFL

Alessandro Sorniotti
IBM Research – Zurich

ory over Network

Graz U

Moritz Lipp
Graz University of Technology

Graz U

oxed environments, virtualized e

and then detect which locations have been evicted from the ca
Such a side channel leaks addresses and allows the adversary
learn information from data-dependent execution. An effect
mitigation strategy is to eliminate data-dependent control fl
over sensitive data, such as cryptographic material.
In contrast, Spectre and Meltdown render this class of atta
generic and significantly harder to mitigate through software cha
only. The side channel is now used *indirectly*, in a way that –

t rely on an implementation "bug", use features corr

BSTRACT

eculative execution is a crucial cornerstone to the performance modern processors. During speculative execution, the processor ay perform operations the program usually would not perform. hile the architectural effects and results of such operations are scarded if the speculative execution is aborted, microarchitectural le effects may remain. The recently published Spectre attacks ploit these side effects to read memory contents of other programs.

1 INTRODUC

Modern computers a these optimizations t observe these side e which would usuall based side-channel do not require physi fall into the categor

t 1.5 years has seen a wide range of attacks with variations on this theme

Outline

- Transient Execution Attacks
- Deep Dive Example
- SpecShield Defense
- Evaluation
- Conclusion

Transient Execution Attacks

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions
- Under the right set of conditions, allows for retrieval of restricted data

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions
- Under the right set of conditions, allows for retrieval of restricted data
- But speculative results might be discarded?

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions
- Under the right set of conditions, allows for retrieval of restricted data
- But speculative results might be discarded?

Sources of Speculation

conditional branches

exceptions

speculative store bypass

value speculation

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions
- Under the right set of conditions, allows for retrieval of restricted data
- But speculative results might be discarded?
  - Exploit µarch side-effects of transient execution

Sources of Speculation

conditional branches

exceptions

speculative store bypass

value speculation

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions
- Under the right set of conditions, allows for retrieval of restricted data
- But speculative results might be discarded?
  - Exploit μarch side-effects of transient execution
  - Architectural changes are discarded, μarch changes persist

Sources of Speculation

conditional branches

exceptions

speculative store bypass

value speculation

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions
- Under the right set of conditions, allows for retrieval of restricted data
- But speculative results might be discarded?
  - Exploit μarch side-effects of transient execution
  - Architectural changes are discarded, μarch changes persist
  - Attacker encodes data into μarch state —> covert channel

Sources of Speculation

conditional branches

exceptions

speculative store bypass

value speculation

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions
- Under the right set of conditions, allows for retrieval of restricted data
- But speculative results might be discarded?
  - Exploit µarch side-effects of transient execution
  - Architectural changes are discarded, µarch changes persist
  - Attacker encodes data into µarch state —> covert channel
    - Ex., traditional cache timing attack techniques, infer memory access patterns of victim

Sources of Speculation

conditional branches

exceptions

speculative store bypass

value speculation

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions
- Under the right set of conditions, allows for retrieval of restricted data
- But speculative results might be discarded?
  - Exploit μarch side-effects of transient execution
  - Architectural changes are discarded, μarch changes persist
  - Attacker encodes data into μarch state —> covert channel
    - Ex., traditional cache timing attack techniques, infer memory access patterns of victim
    - Covert channel functions as medium for send/recv data outside the speculative window

Sources of Speculation

conditional branches

exceptions

speculative store bypass

value speculation

Transient Execution Attacks

- Speculation allows the execution of **incorrect** instructions
- Under the right set of conditions, allows for retrieval of restricted data
- But speculative results might be discarded?
  - Exploit µarch side-effects of transient execution
  - Architectural changes are discarded, µarch changes persist
  - Attacker encodes data into µarch state —> covert channel
    - Ex., traditional cache timing attack techniques, infer memory access patterns of victim
    - Covert channel functions as medium for send/recv data outside the speculative window

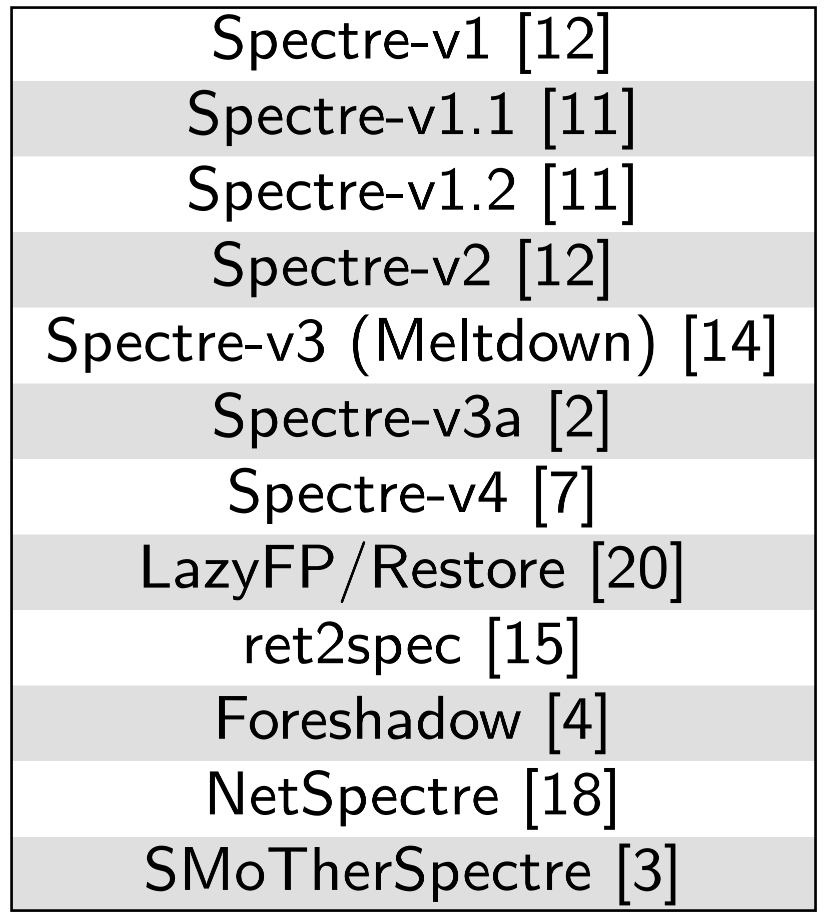
Sources of Speculation

conditional branches

exceptions

speculative store bypass

value speculation



Spectre-v1 [12]
Spectre-v1.1 [11]
Spectre-v1.2 [11]
Spectre-v2 [12]
Spectre-v3 (Meltdown) [14]
Spectre-v3a [2]
Spectre-v4 [7]
LazyFP/Restore [20]
ret2spec [15]
Foreshadow [4]
NetSpectre [18]
SMoTherSpectre [3]

Deep Dive Example: Spectre-v1

Deep Dive Example: Spectre-v1

- Illustrative example: Spectre-v1, bounds-check-bypass

Deep Dive Example: Spectre-v1

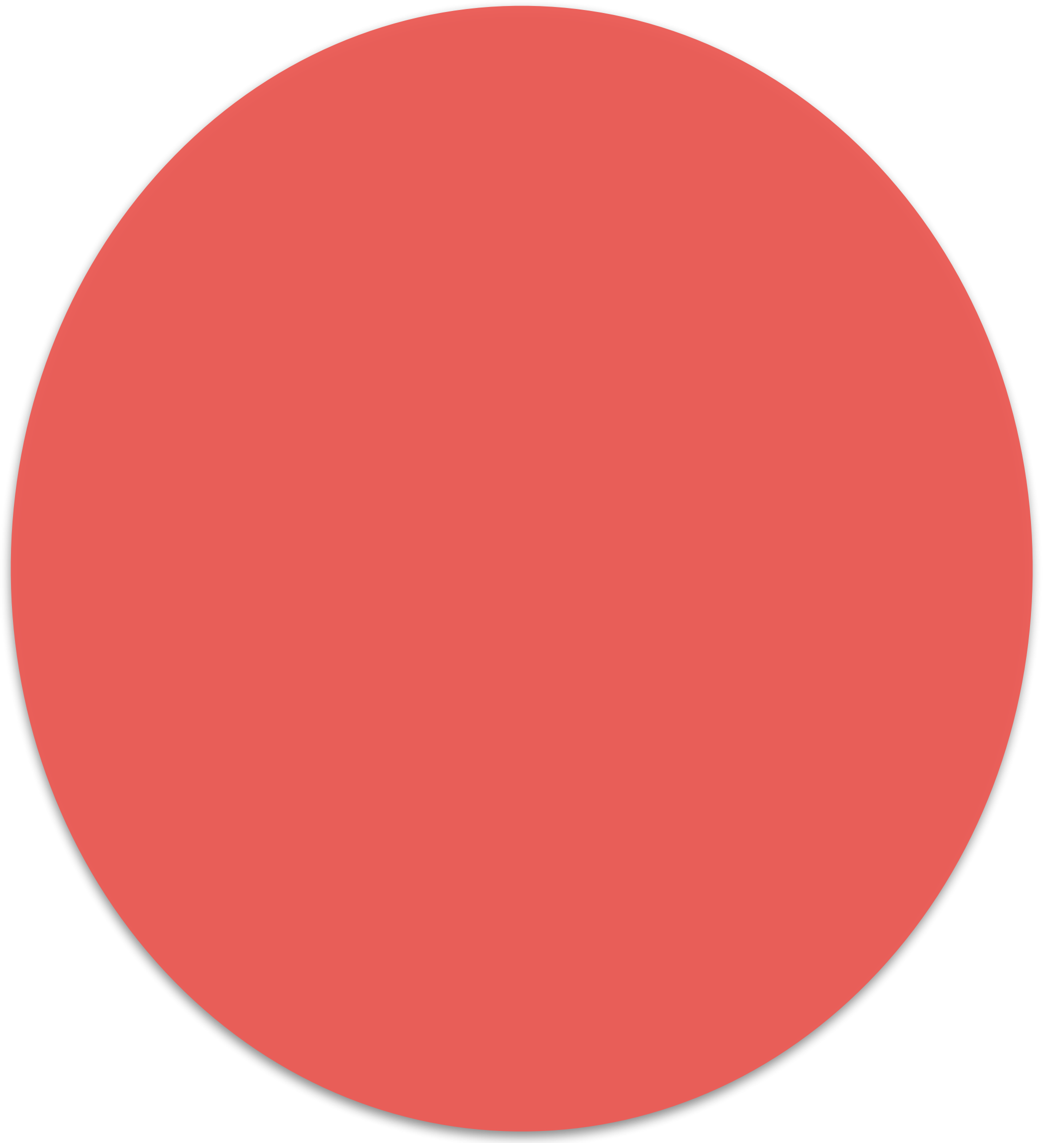
- Illustrative example: Spectre-v1, bounds-check-bypass

- Transient execution attacks require two phases:
  - (1) Speculation primitive allows access to restricted data
  - (2) Utilization of covert channel to disclose data outside of speculative window

Deep Dive Example: Spectre-v1

Deep Dive Example: Spectre-v1

Attacker

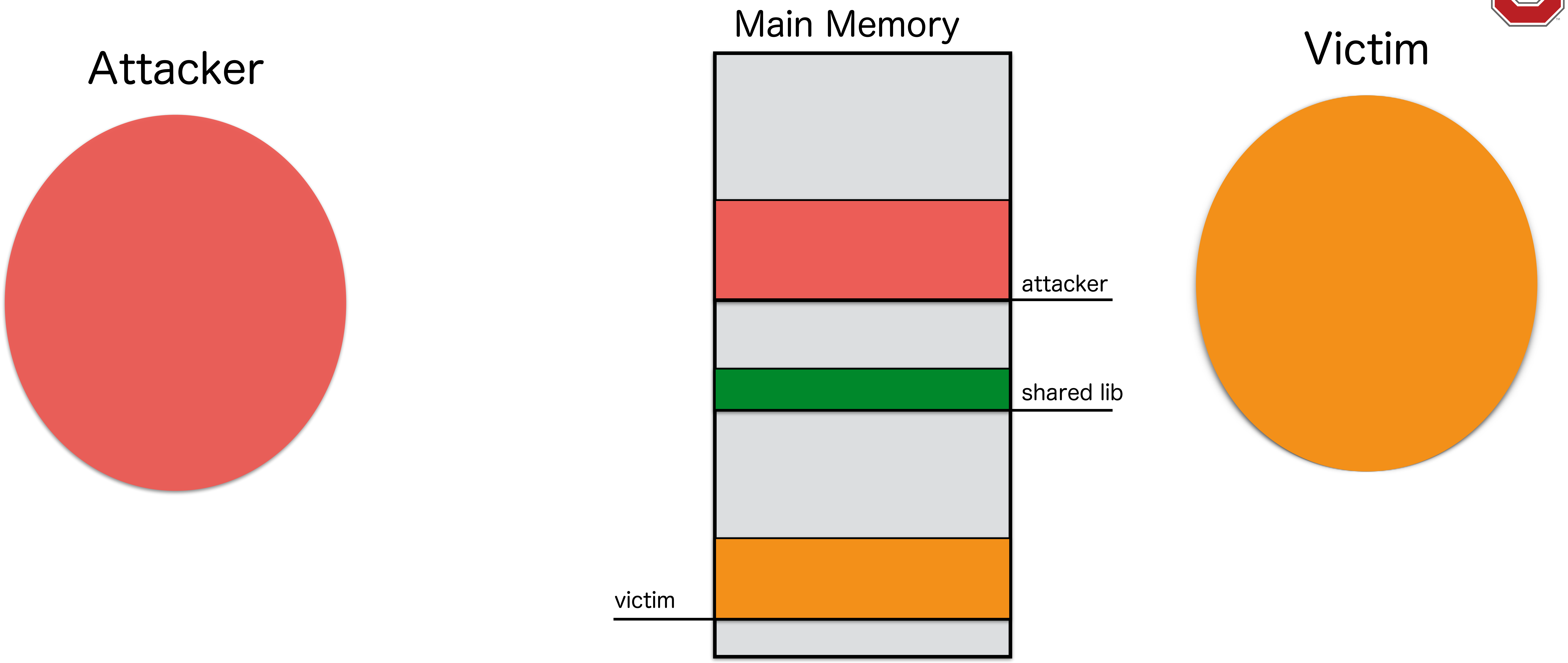


Deep Dive Example: Spectre-v1

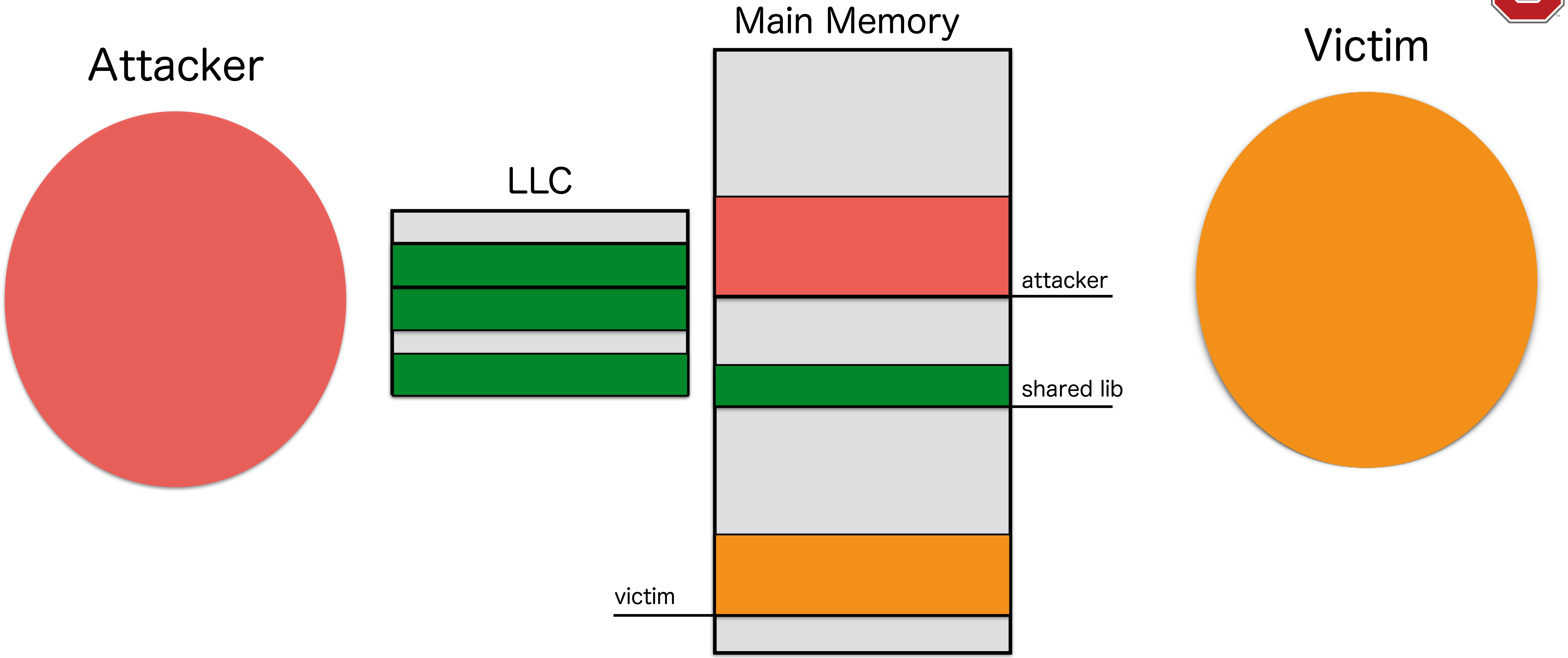
Attacker

Victim

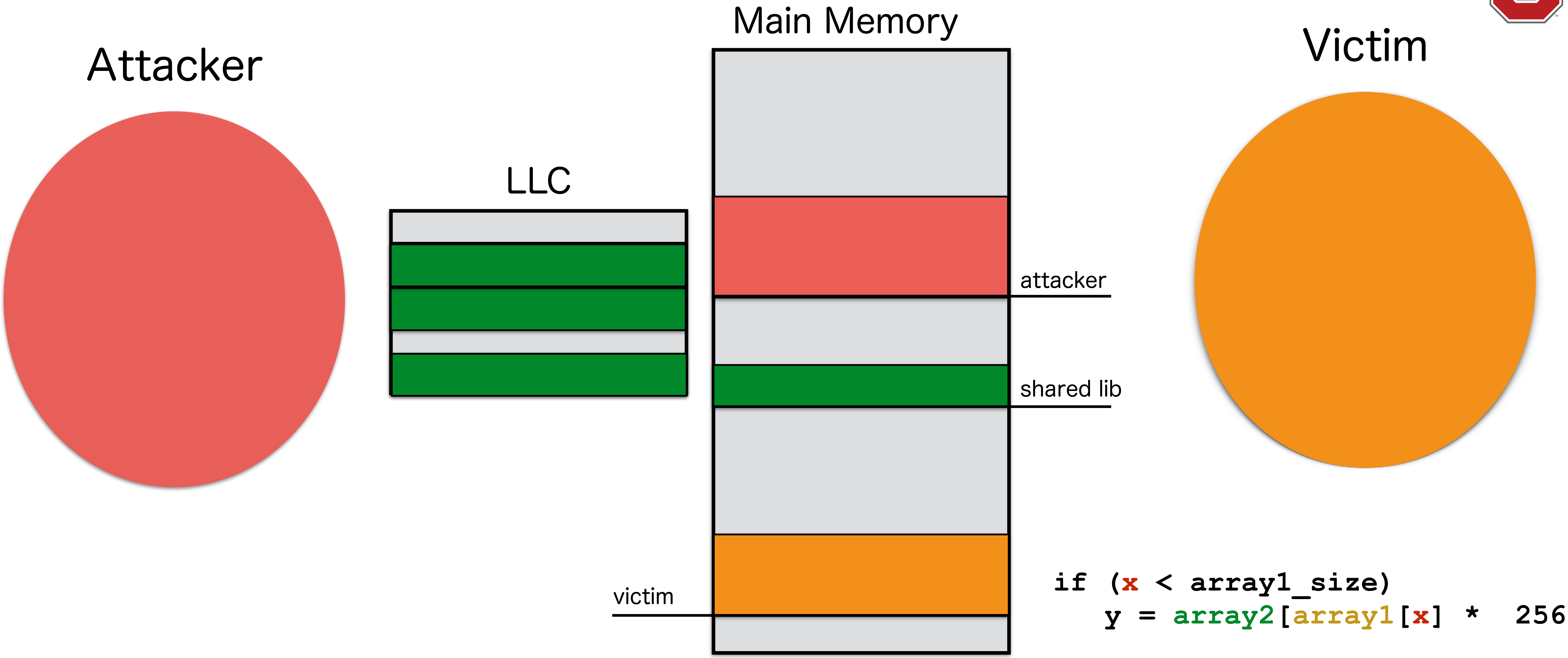
Deep Dive Example: Spectre-v1



Deep Dive Example: Spectre-v1

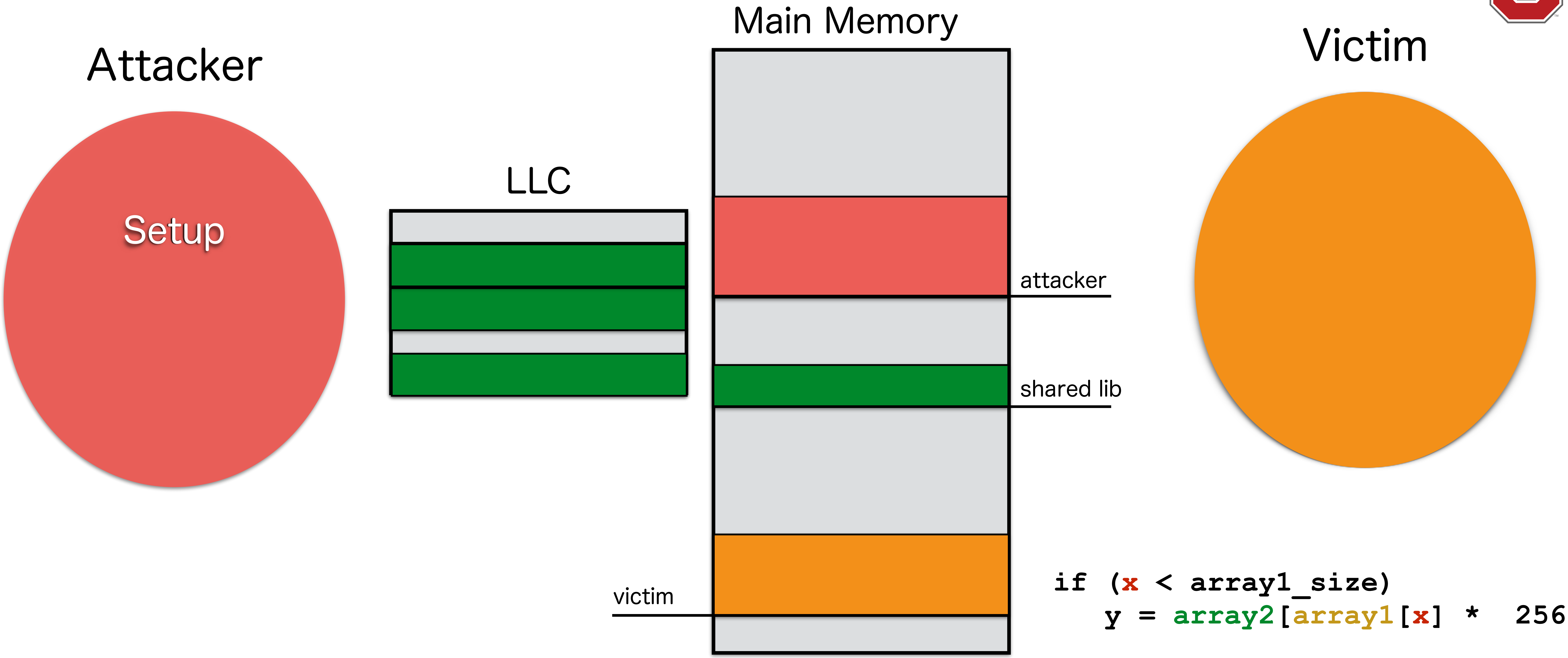


Deep Dive Example: Spectre-v1



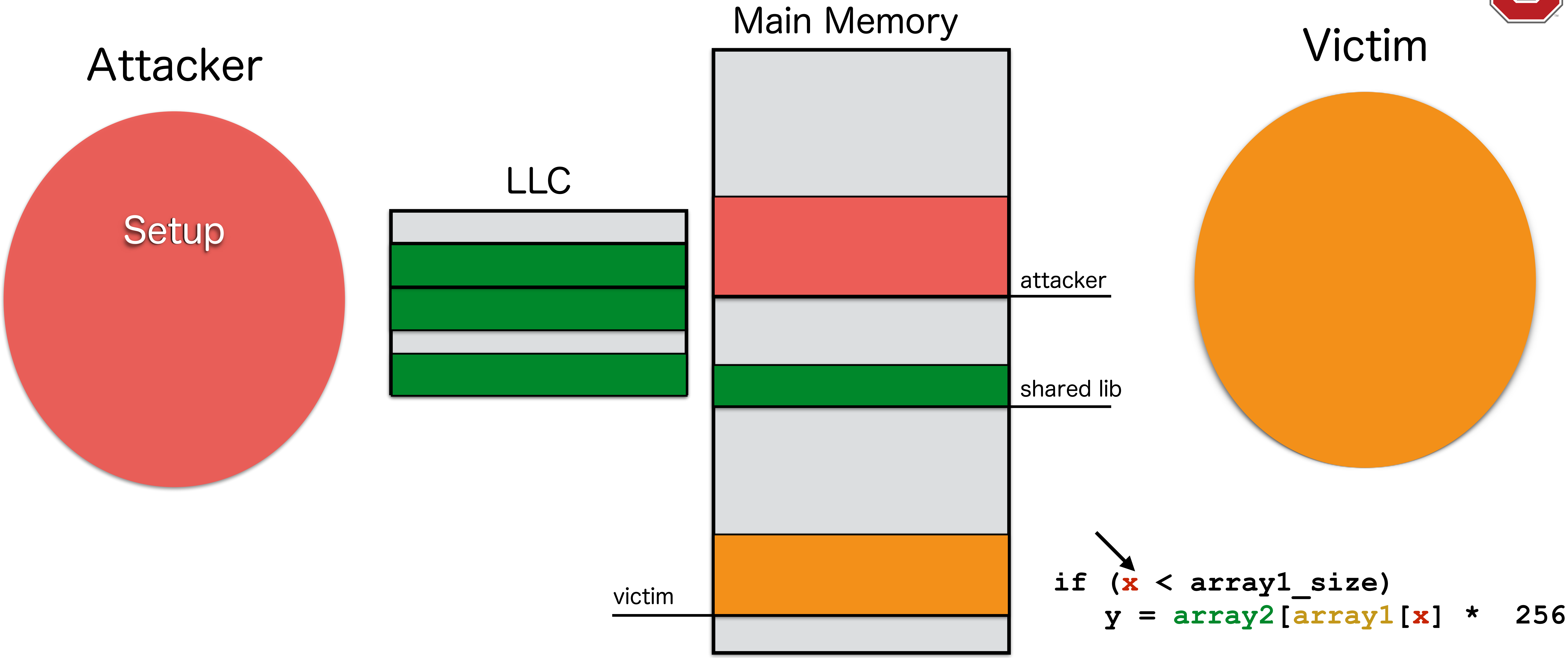
```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

Deep Dive Example: Spectre-v1



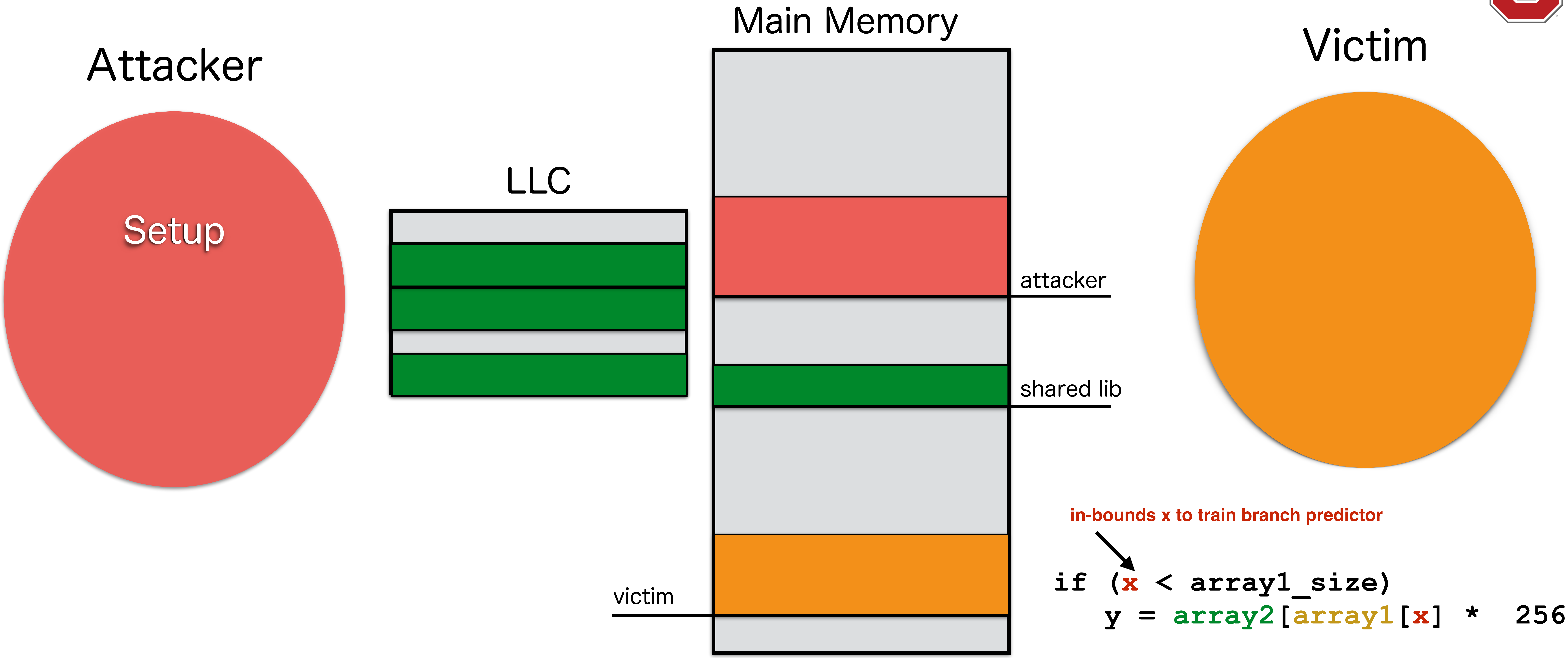
```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

Deep Dive Example: Spectre-v1



```
if (x < array1_size)
    y = array2[array1[x] *  256];
```

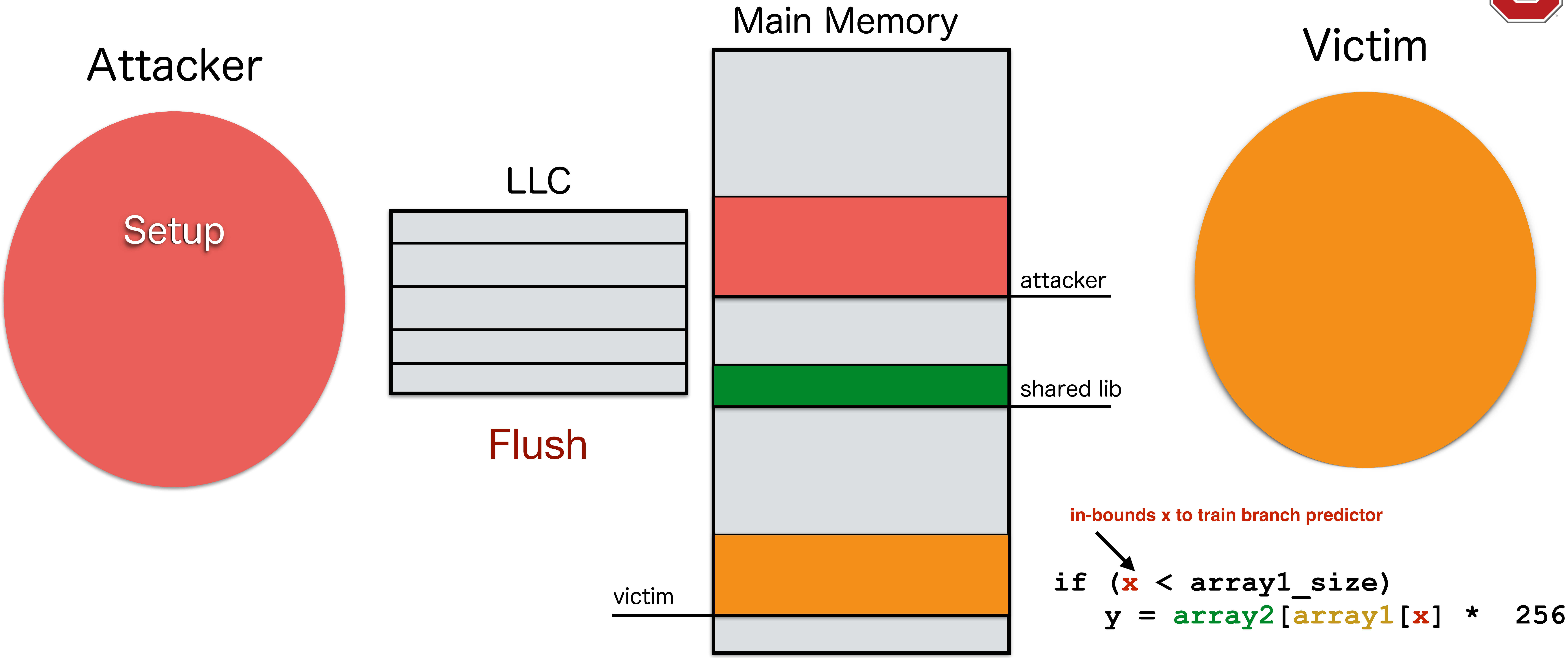
Deep Dive Example: Spectre-v1



in-bounds x to train branch predictor

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

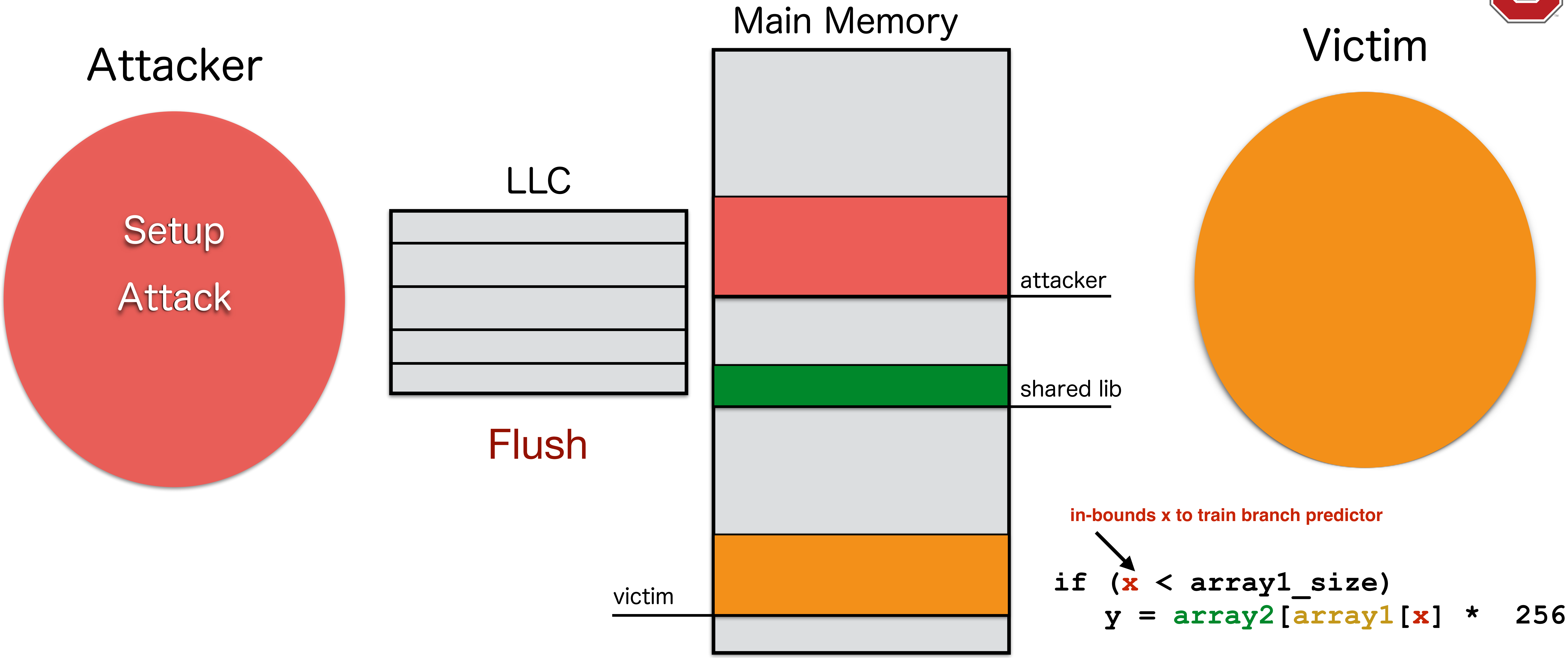
Deep Dive Example: Spectre-v1



in-bounds x to train branch predictor

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

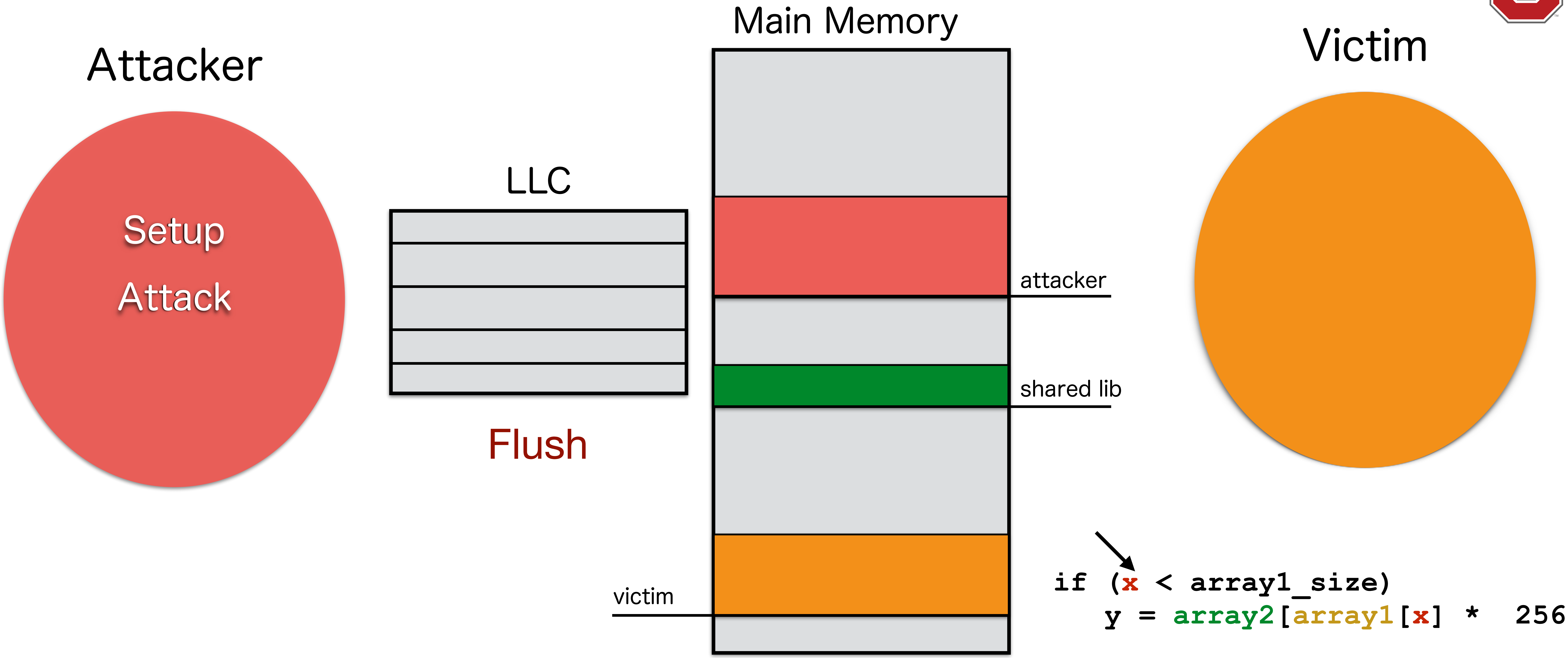
Deep Dive Example: Spectre-v1



in-bounds x to train branch predictor

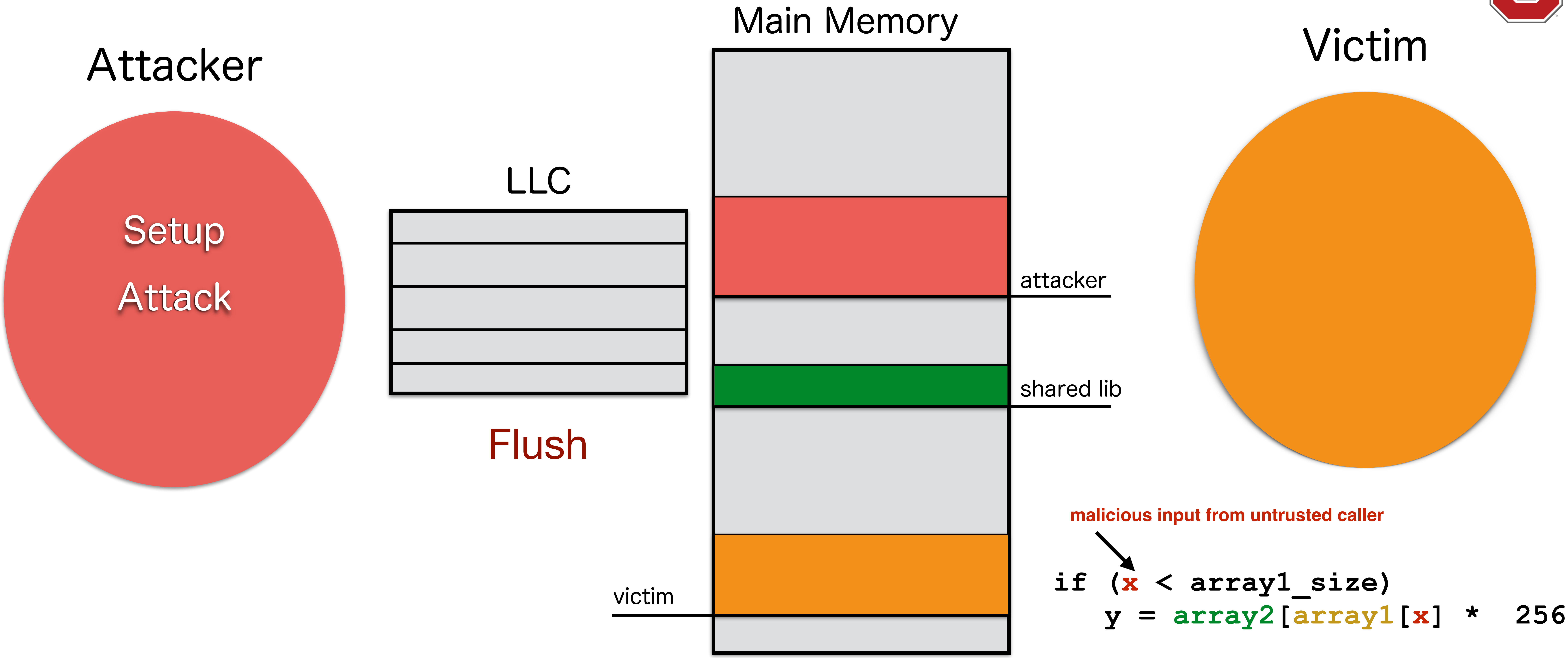
```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

Deep Dive Example: Spectre-v1



```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

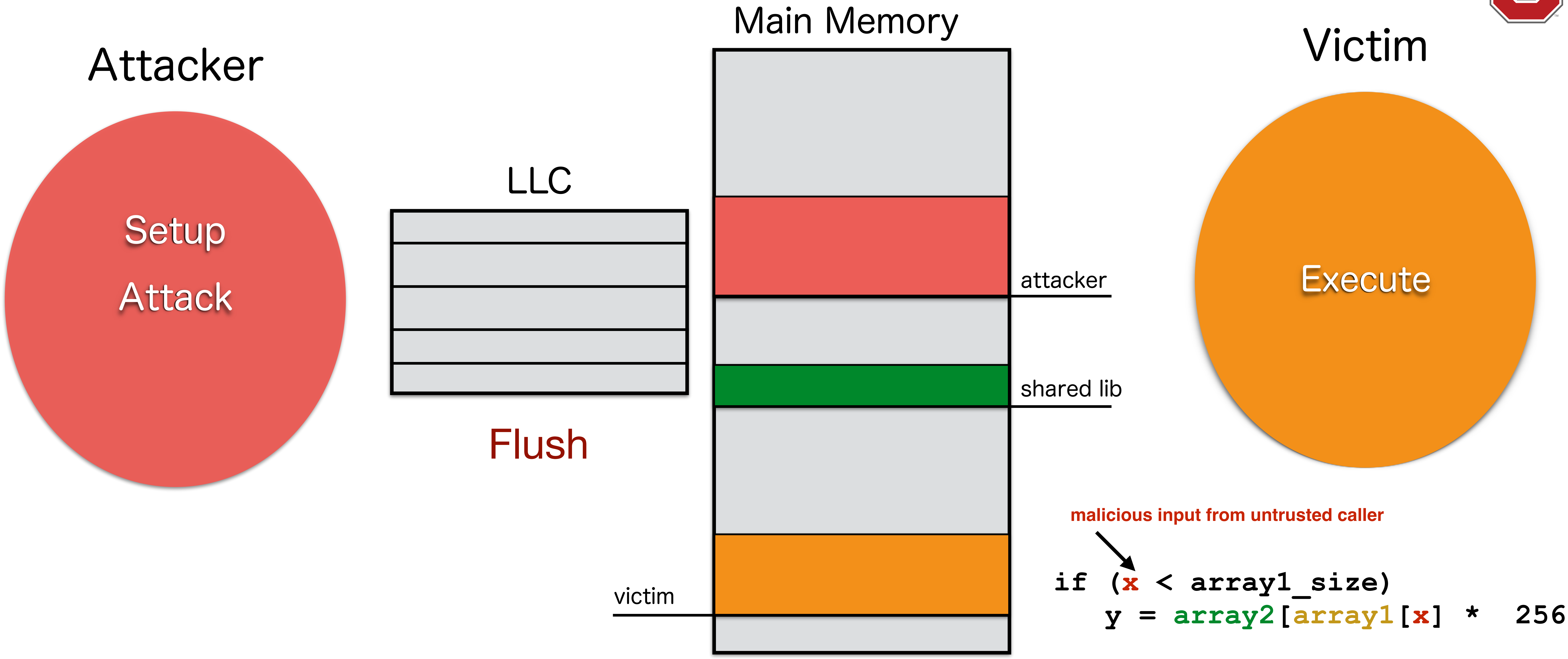
Deep Dive Example: Spectre-v1



malicious input from untrusted caller

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

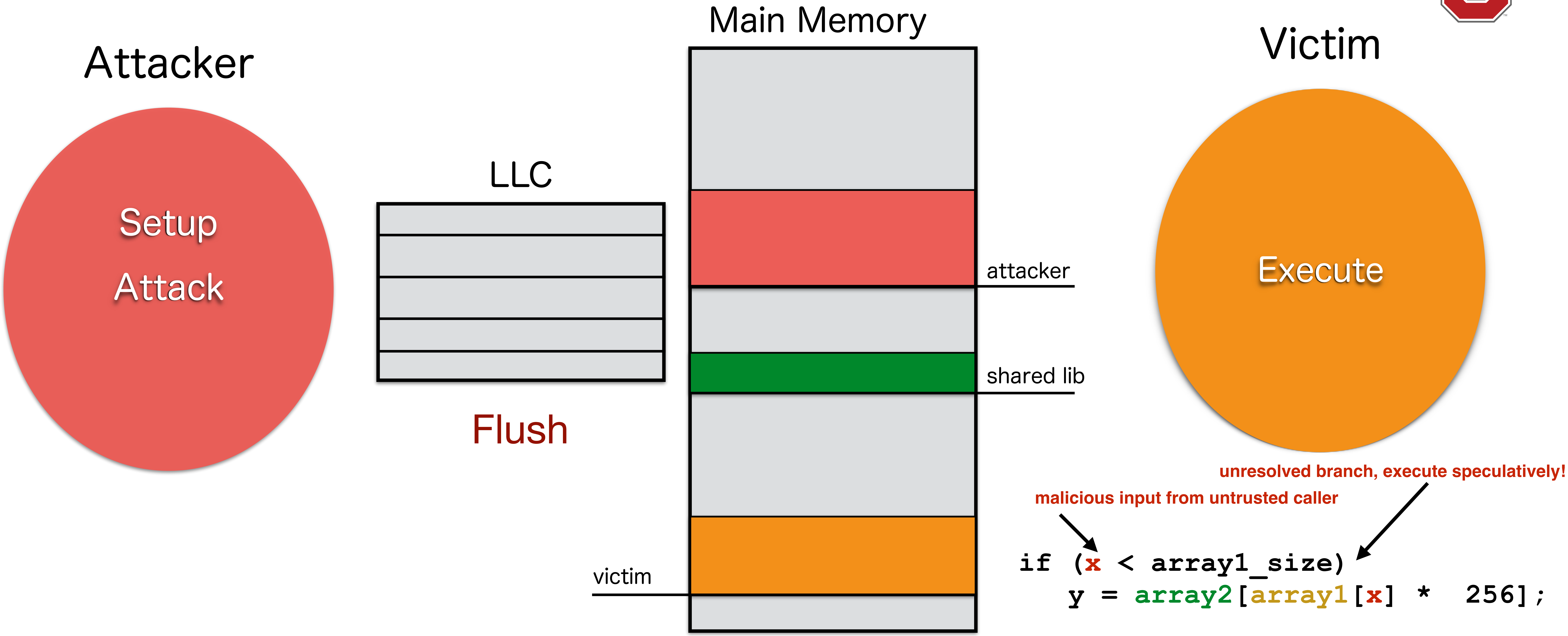
Deep Dive Example: Spectre-v1



malicious input from untrusted caller

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

Deep Dive Example: Spectre-v1

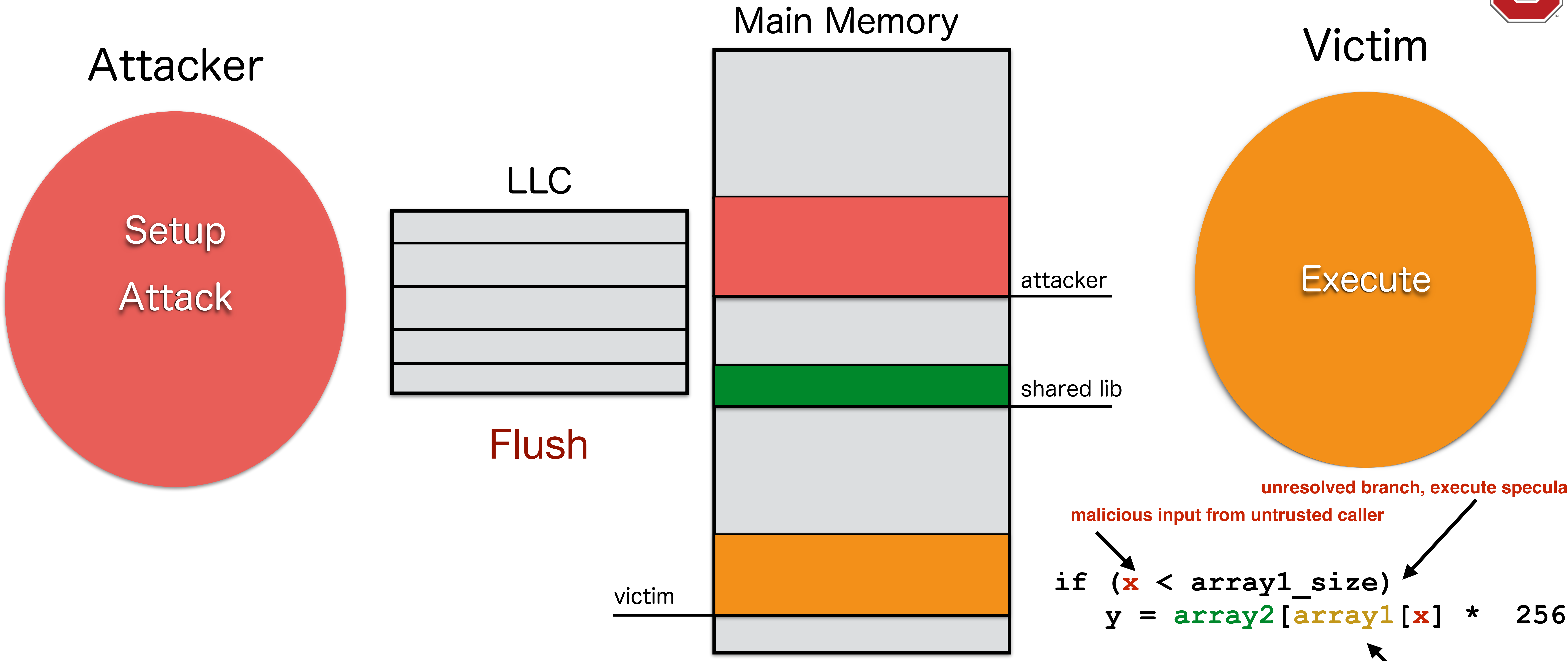


unresolved branch, execute speculatively!

malicious input from untrusted caller

```
if (x < array1_size)
    y = array2[array1[x] *  256];
```

Deep Dive Example: Spectre-v1

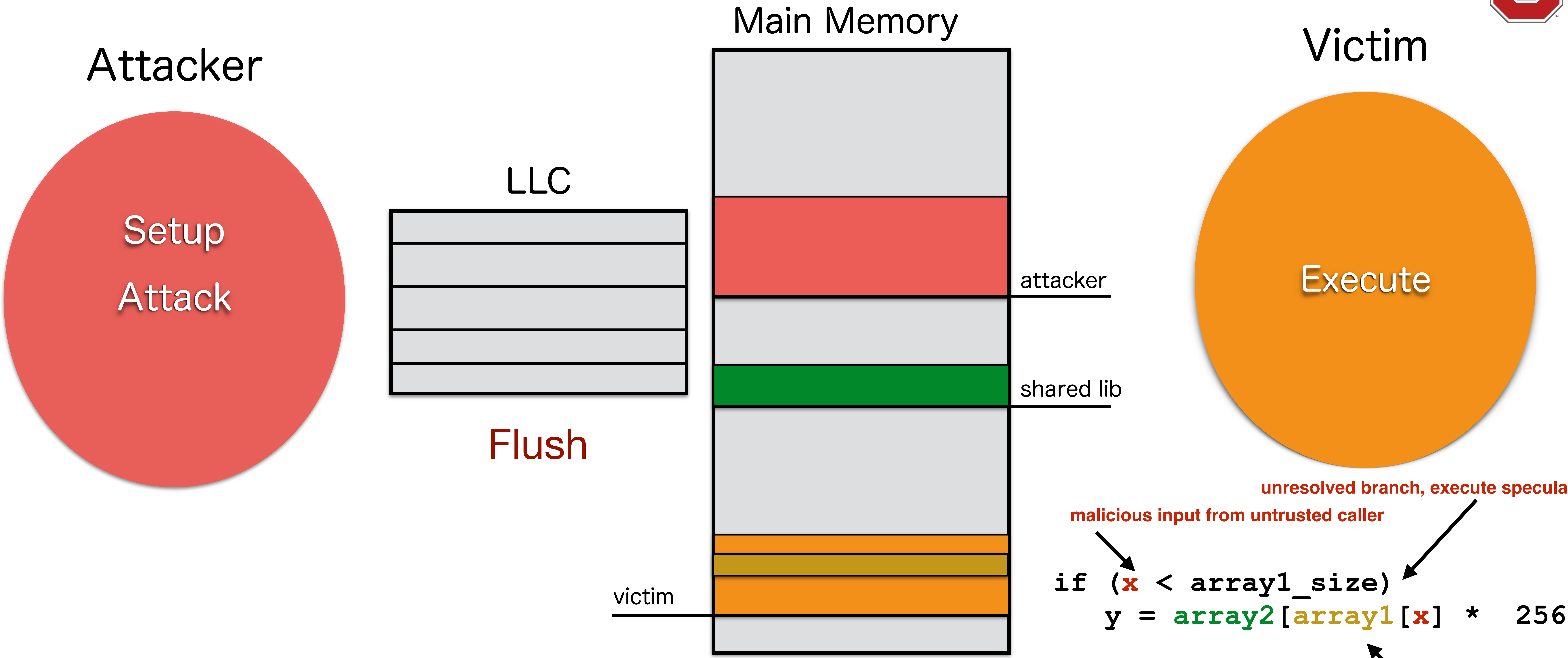


unresolved branch, execute speculatively!

malicious input from untrusted caller

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

Deep Dive Example: Spectre-v1

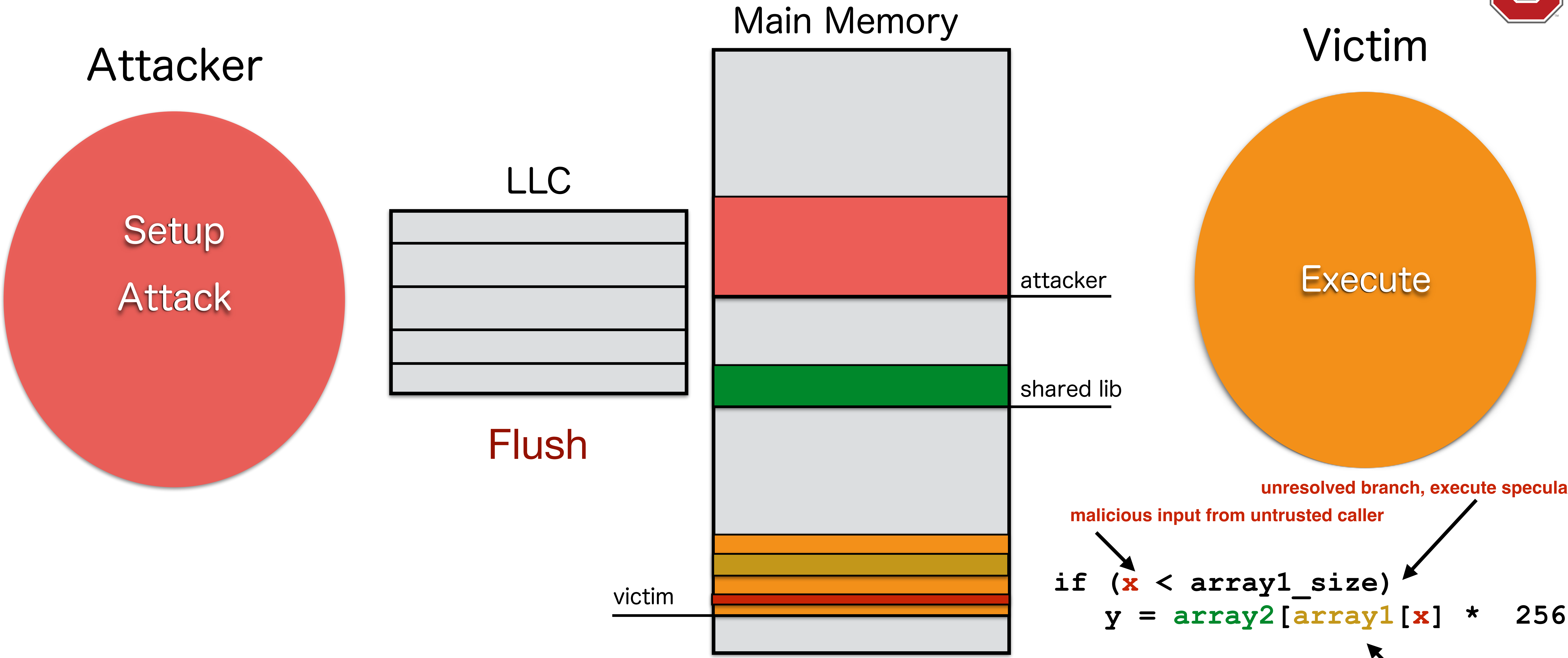


unresolved branch, execute speculatively!

malicious input from untrusted caller

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

Deep Dive Example: Spectre-v1



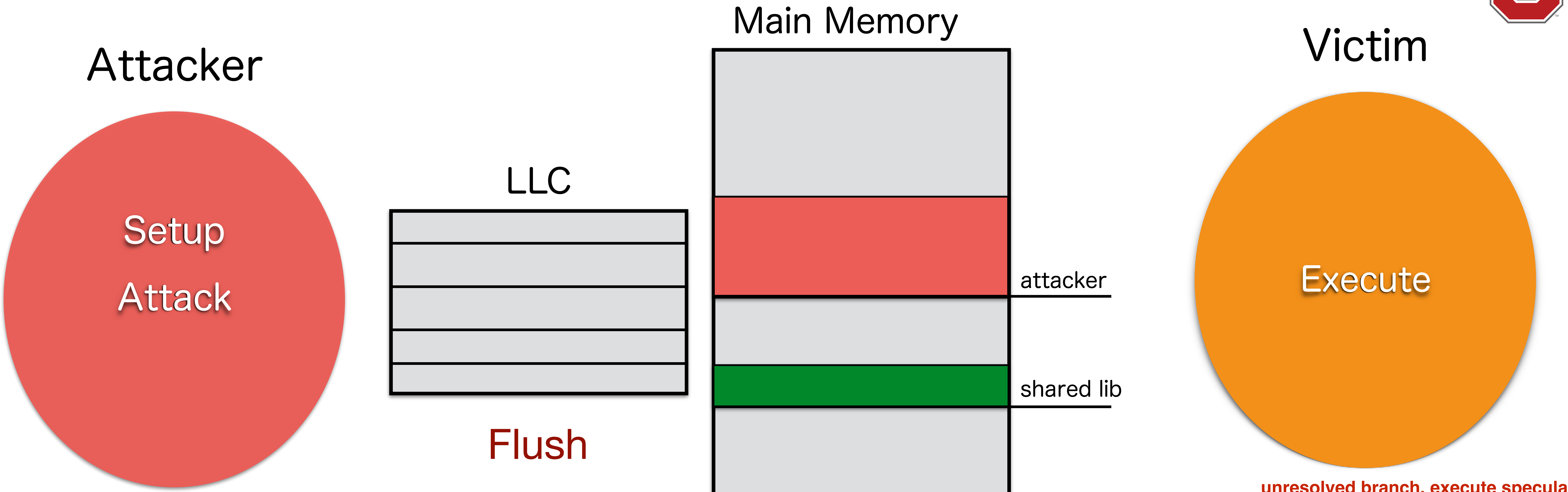
unresolved branch, execute speculatively!

malicious input from untrusted caller

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

offset to restricted data, known a priori

Deep Dive Example: Spectre-v1



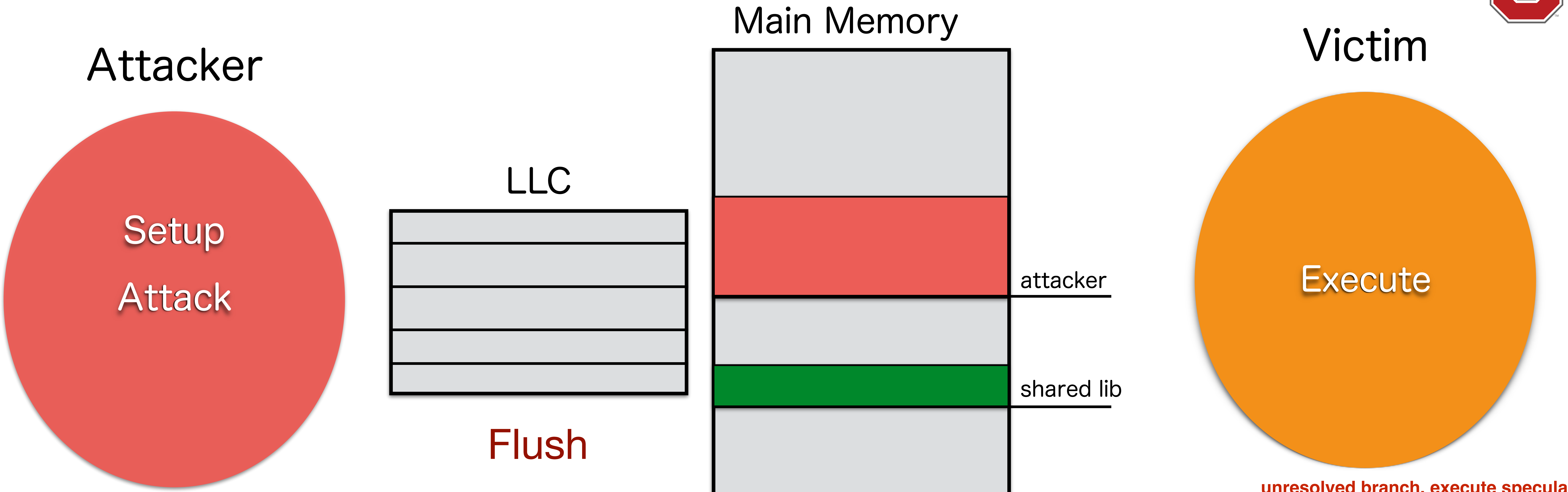
unresolved branch, execute speculatively!

malicious input from untrusted caller

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

offset to restricted data, known a priori

Deep Dive Example: Spectre-v1



unresolved branch, execute speculatively!

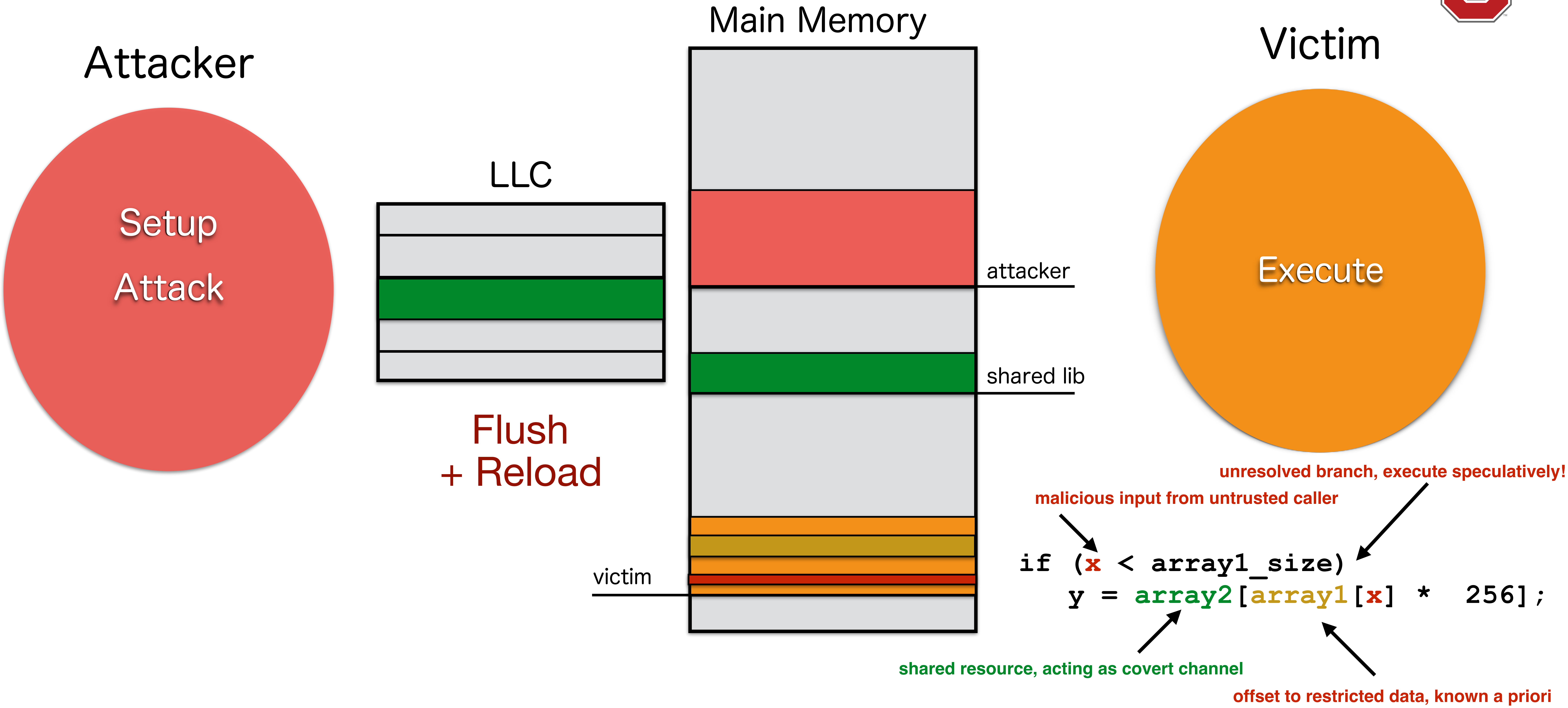
malicious input from untrusted caller

```
if (x < array1_size)
    y = array2[array1[x] *  256];
```

shared resource, acting as covert channel

offset to restricted data, known a priori

Deep Dive Example: Spectre-v1



unresolved branch, execute speculatively!

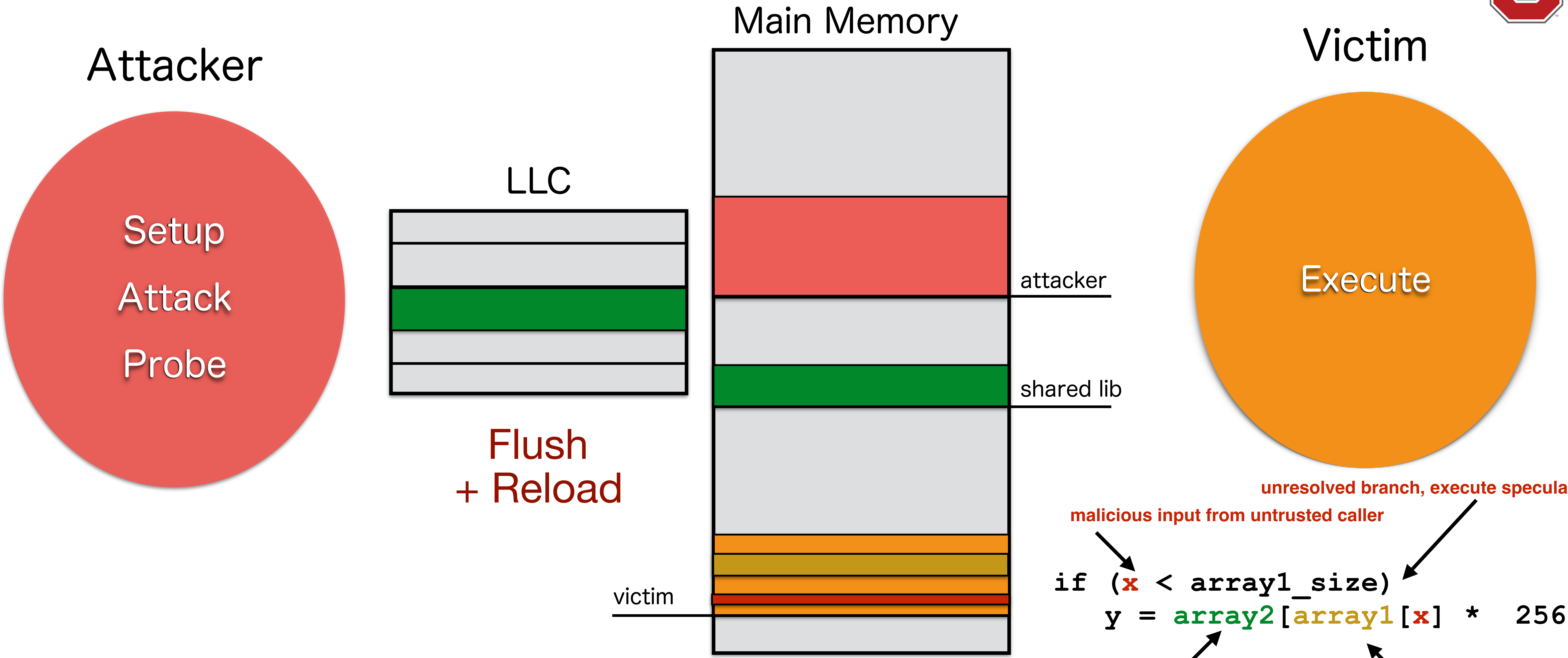
malicious input from untrusted caller

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

shared resource, acting as covert channel

offset to restricted data, known a priori

Deep Dive Example: Spectre-v1



unresolved branch, execute speculatively!

malicious input from untrusted caller

```
if (x < array1_size)
    y = array2[array1[x] *  256];
```

shared resource, acting as covert channel

offset to restricted data, known a priori

Existing Solutions

Existing Solutions

- Software-only mitigation solutions

Existing Solutions

- Software-only mitigation solutions
  - Generally very high overhead for good coverage

Existing Solutions

- Software-only mitigation solutions
  - Generally very high overhead for good coverage
    - Attack exploits traces left in µarch, opaque to programmer by design

Existing Solutions

- Software-only mitigation solutions
  - Generally very high overhead for good coverage
    - Attack exploits traces left in μarch, opaque to programmer by design
    - Coarse-grain, rely on serializing instructions

Existing Solutions

- Software-only mitigation solutions
  - Generally very high overhead for good coverage
    - Attack exploits traces left in μarch, opaque to programmer by design
    - Coarse-grain, rely on serializing instructions
  - Ad-hoc and specific to exploits, rely on manual insertion, static analysis shown to miss corner cases

Existing Solutions

- Software-only mitigation solutions
  - Generally very high overhead for good coverage
    - Attack exploits traces left in µarch, opaque to programmer by design
    - Coarse-grain, rely on serializing instructions
  - Ad-hoc and specific to exploits, rely on manual insertion, static analysis shown to miss corner cases
- Existing hardware solutions

Existing Solutions

- Software-only mitigation solutions
  - Generally very high overhead for good coverage
    - Attack exploits traces left in μarch, opaque to programmer by design
    - Coarse-grain, rely on serializing instructions
  - Ad-hoc and specific to exploits, rely on manual insertion, static analysis shown to miss corner cases
- Existing hardware solutions
  - Lower-overhead, better coverage

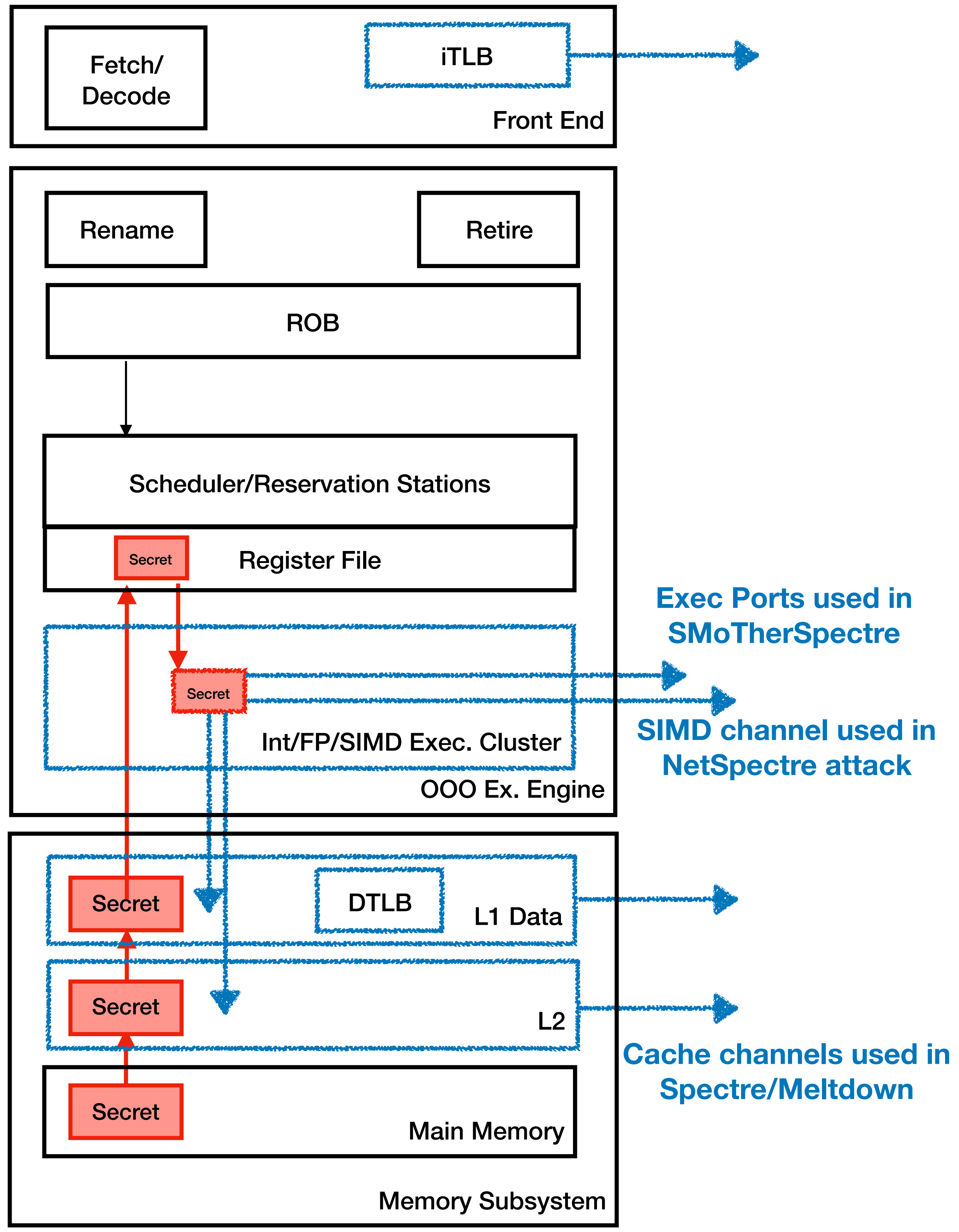
Existing Solutions

- Software-only mitigation solutions
  - Generally very high overhead for good coverage
    - Attack exploits traces left in µarch, opaque to programmer by design
    - Coarse-grain, rely on serializing instructions
  - Ad-hoc and specific to exploits, rely on manual insertion, static analysis shown to miss corner cases
- Existing hardware solutions
  - Lower-overhead, better coverage
  - Mostly focused on closing specific covert channels

Existing Solutions

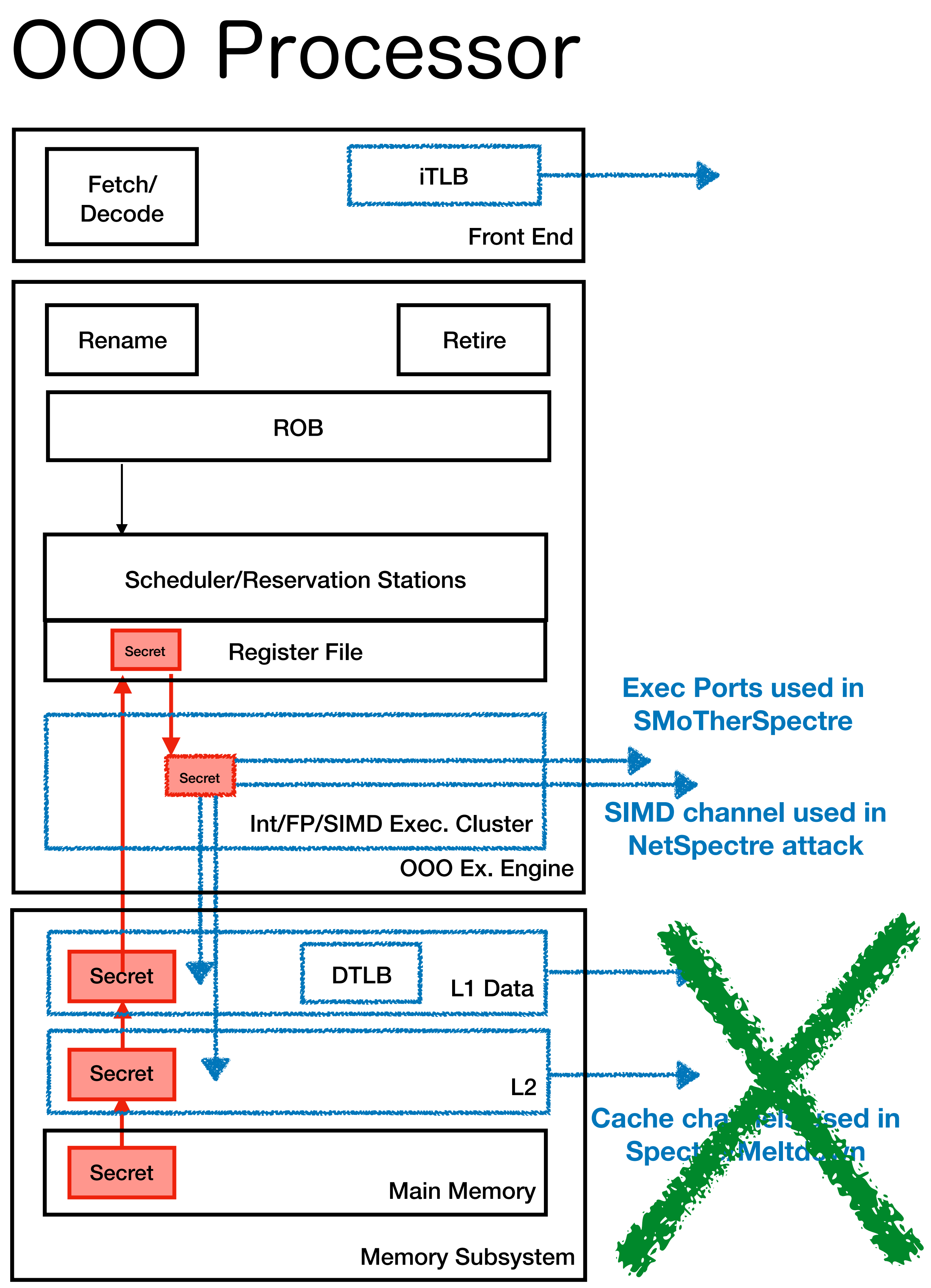
- Software-only mitigation solutions
  - Generally very high overhead for good coverage
    - Attack exploits traces left in µarch, opaque to programmer by design
    - Coarse-grain, rely on serializing instructions
  - Ad-hoc and specific to exploits, rely on manual insertion, static analysis shown to miss corner cases
- Existing hardware solutions
  - Lower-overhead, better coverage
  - Mostly focused on closing specific covert channels

OOO Processor


Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Cache channels used in Spectre/Meltdown

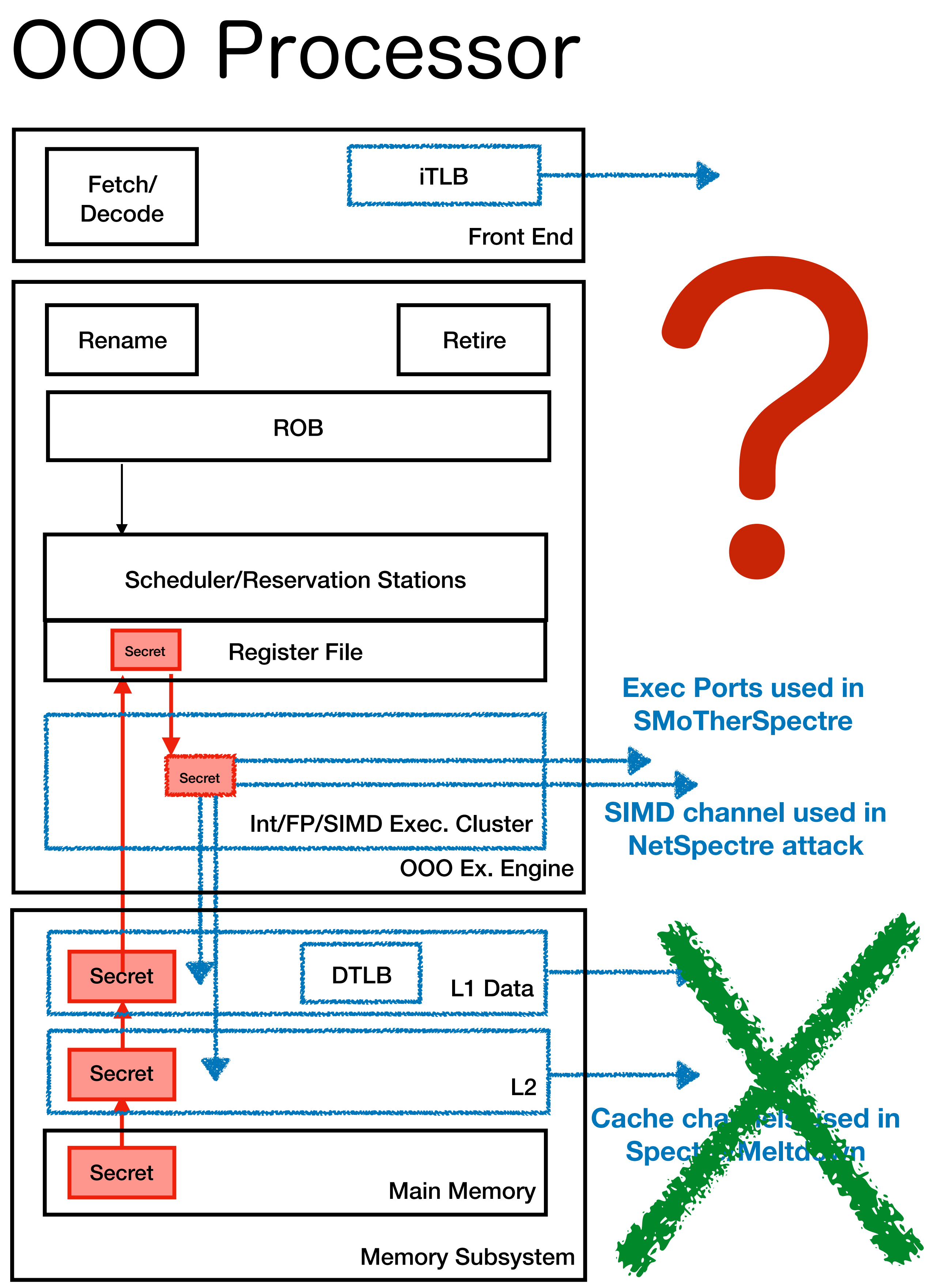
Existing Solutions

- Software-only mitigation solutions
  - Generally very high overhead for good coverage
    - Attack exploits traces left in μarch, opaque to programmer by design
    - Coarse-grain, rely on serializing instructions
  - Ad-hoc and specific to exploits, rely on manual insertion, static analysis shown to miss corner cases
- Existing hardware solutions
  - Lower-overhead, better coverage
  - Mostly focused on closing specific covert channels

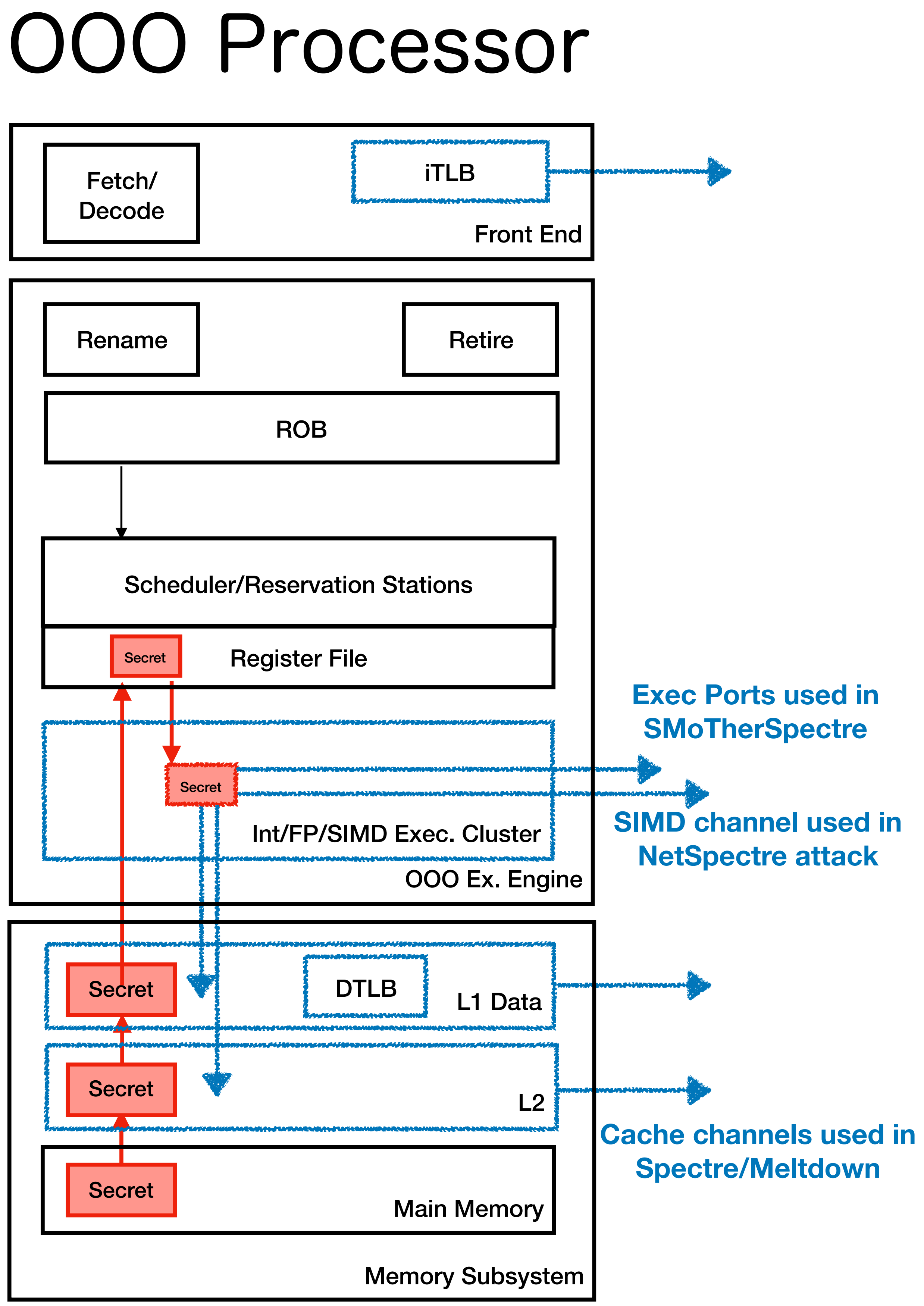

OOO Processor
Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

Existing Solutions

- Software-only mitigation solutions
  - Generally very high overhead for good coverage
    - Attack exploits traces left in μarch, opaque to programmer by design
    - Coarse-grain, rely on serializing instructions
  - Ad-hoc and specific to exploits, rely on manual insertion, static analysis shown to miss corner cases
- Existing hardware solutions
  - Lower-overhead, better coverage
  - Mostly focused on closing specific covert channels


OOO Processor
Fetch/
Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Cache channels used in Spectre/Meltdown
?

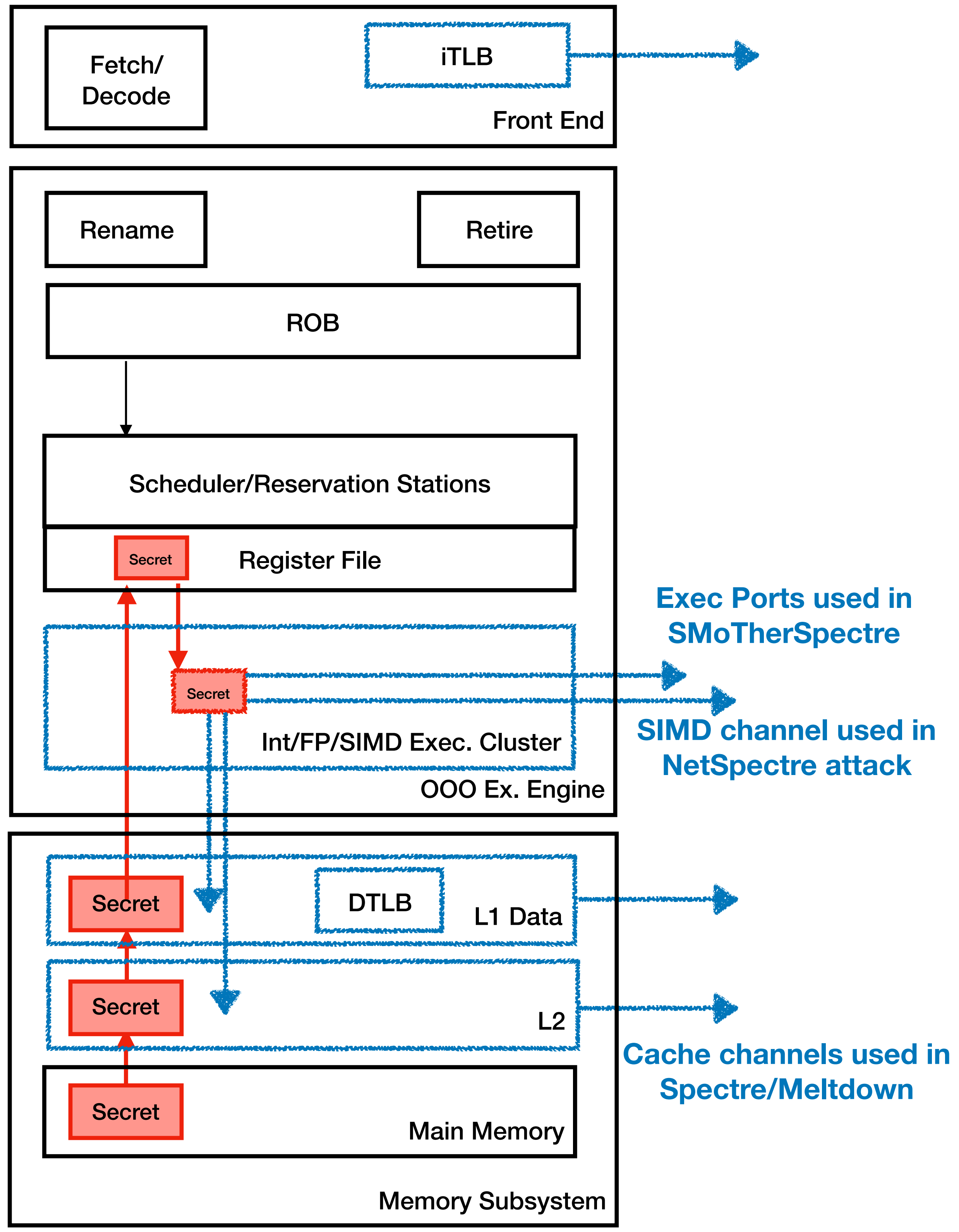
SpecShield: Our Proposed Solution


OOO Processor
Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

SpecShield: Our Proposed Solution

SpecShield is a family of uarch mitigation solutions with different isolation properties for trade-offs with performance

OOO Processor



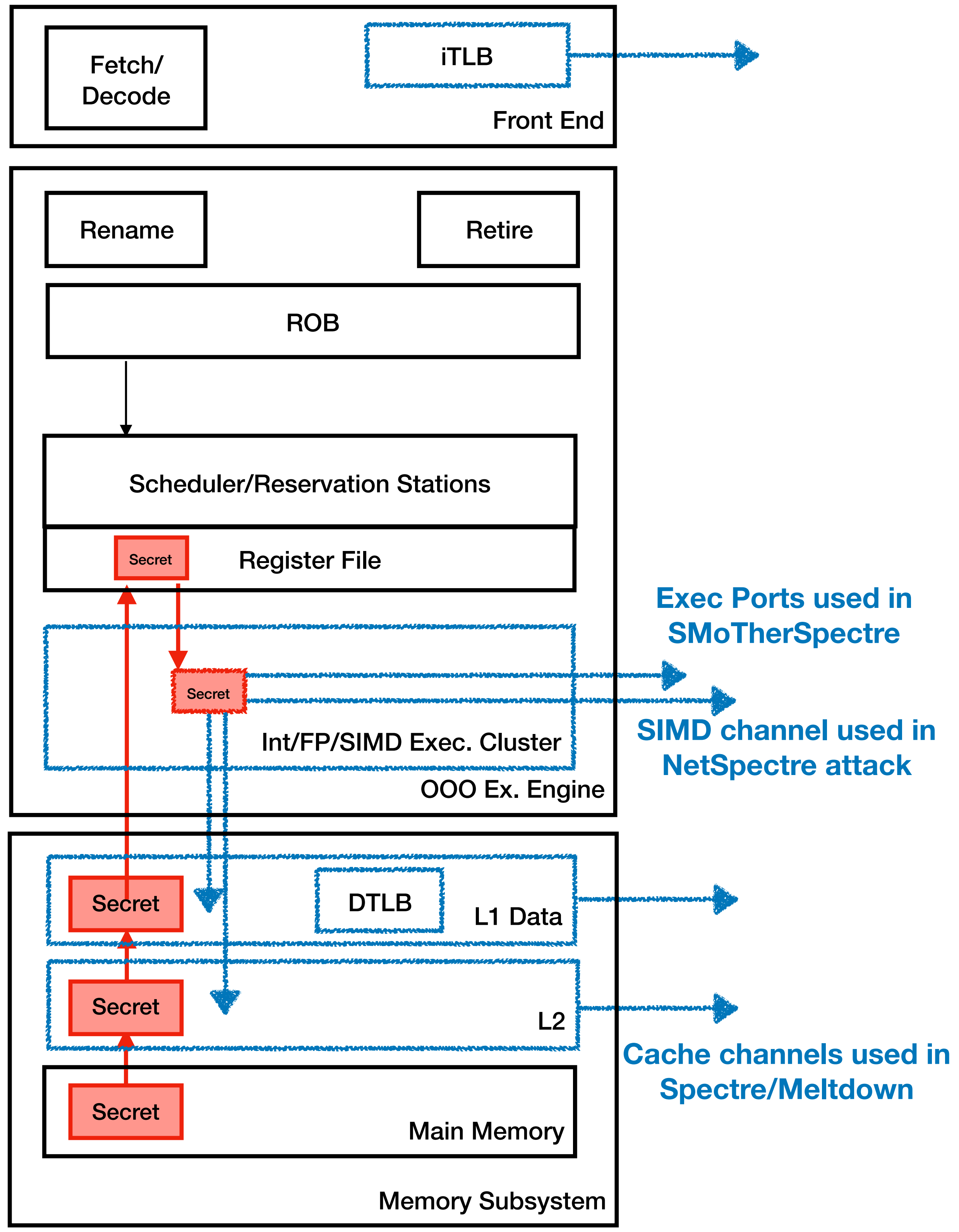
Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Cache channels used in Spectre/Meltdown

SpecShield: Our Proposed Solution

SpecShield is a family of uarch mitigation solutions with different isolation properties for trade-offs with performance

Goal: Isolate transient data from covert channel transmission

OOO Processor



Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

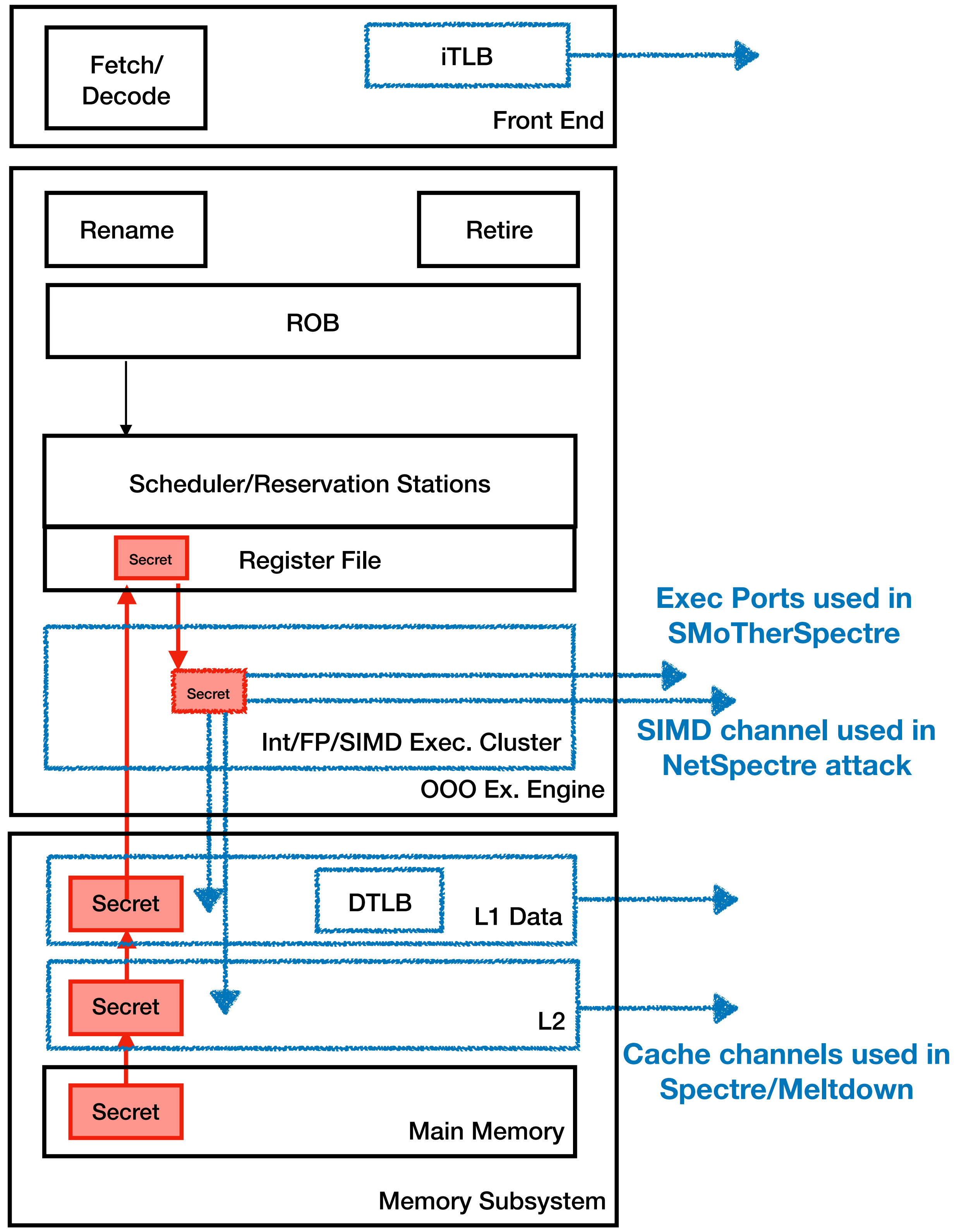
SpecShield: Our Proposed Solution

SpecShield is a family of uarch mitigation solutions with different isolation properties for trade-offs with performance

Goal: Isolate transient data from covert channel transmission

Threat Model

OOO Processor


Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Cache channels used in Spectre/Meltdown

SpecShield: Our Proposed Solution

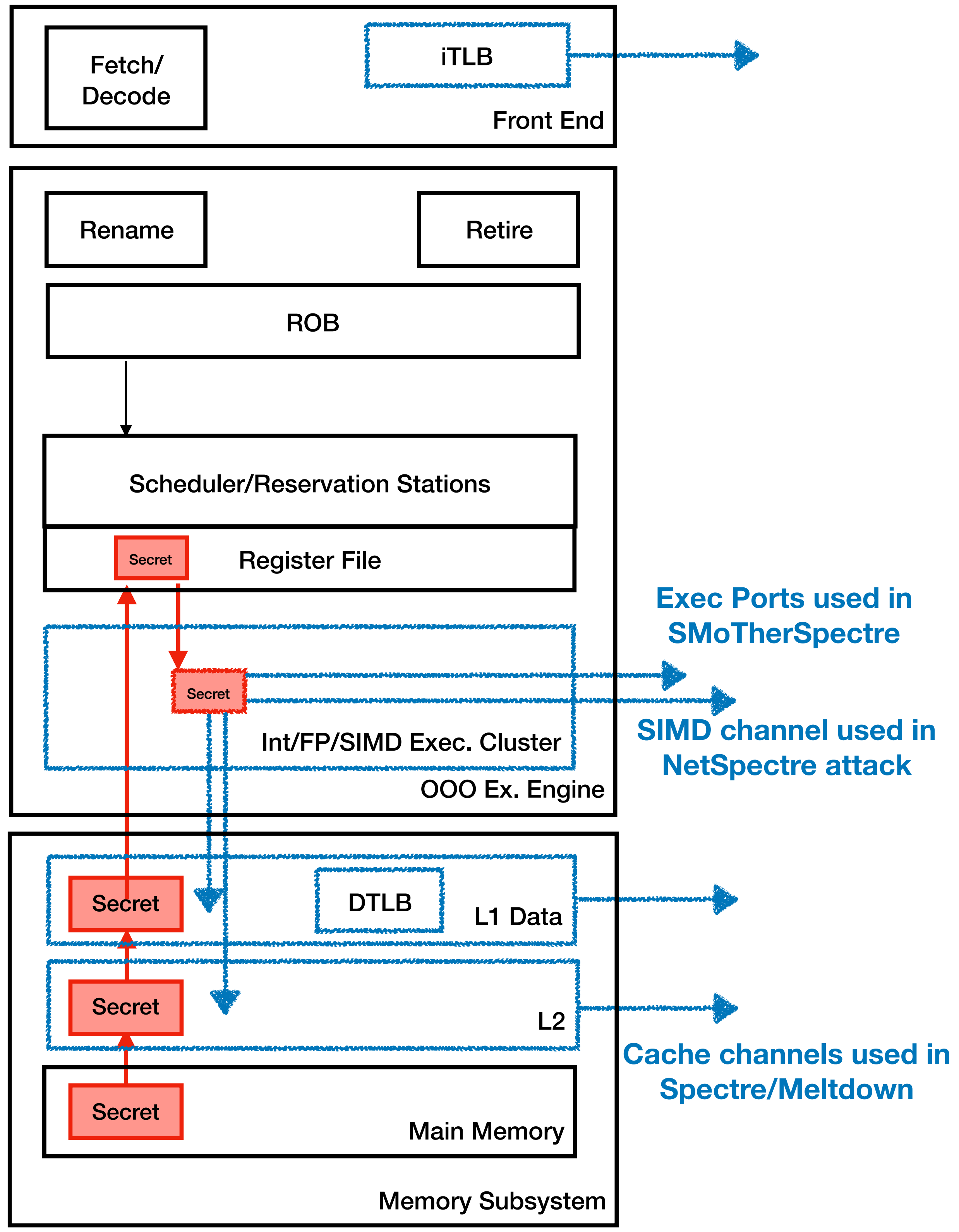
SpecShield is a family of uarch mitigation solutions with different isolation properties for trade-offs with performance

Goal: Isolate transient data from covert channel transmission

Threat Model

- Sensitive data resides anywhere in the memory hierarchy

OOO Processor


Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

SpecShield: Our Proposed Solution

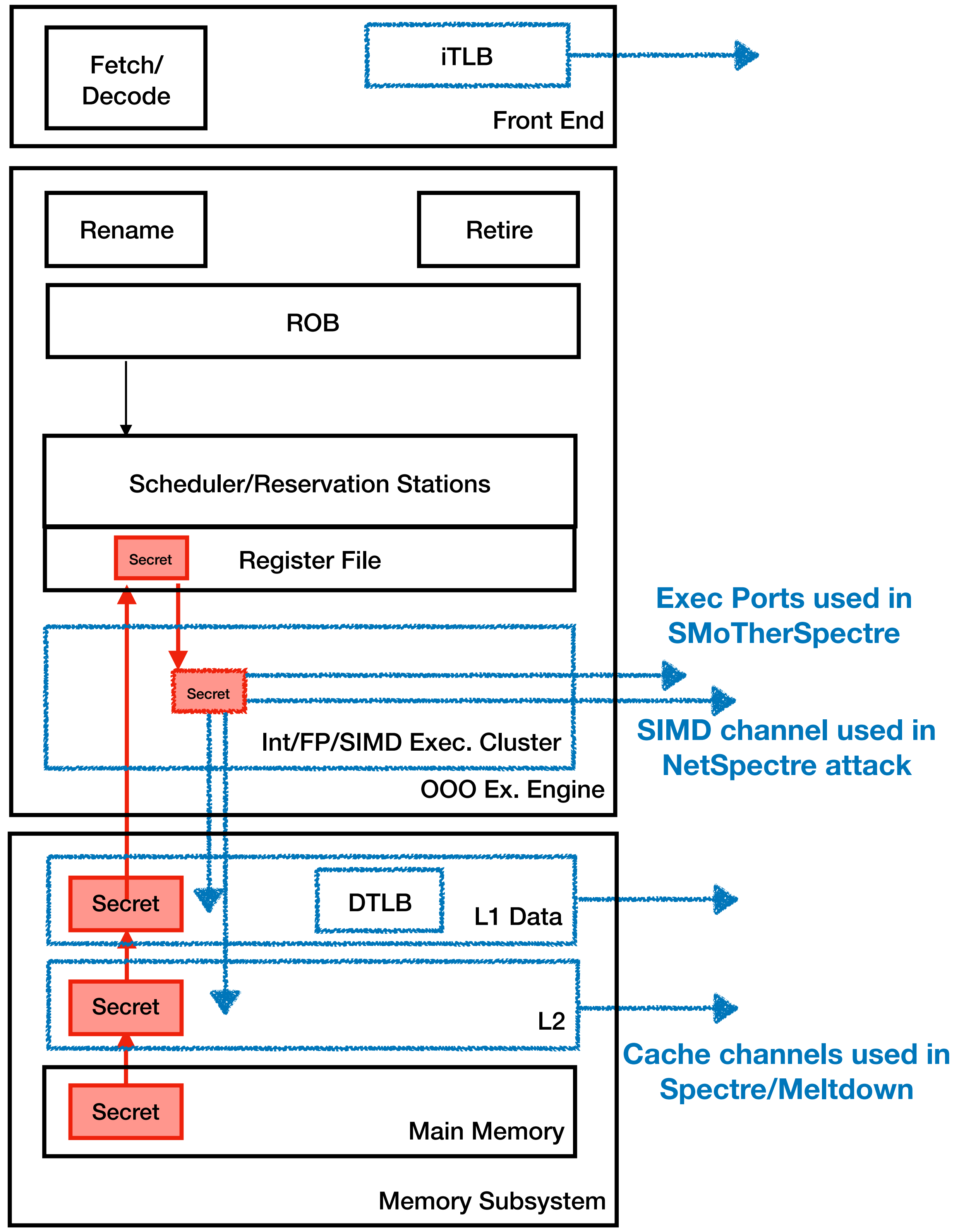
SpecShield is a family of uarch mitigation solutions with different isolation properties for trade-offs with performance

Goal: Isolate transient data from covert channel transmission

Threat Model

- Sensitive data resides anywhere in the memory hierarchy
  - Accessed through a transient misspeculated instruction

OOO Processor



Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Cache channels used in Spectre/Meltdown

SpecShield: Our Proposed Solution

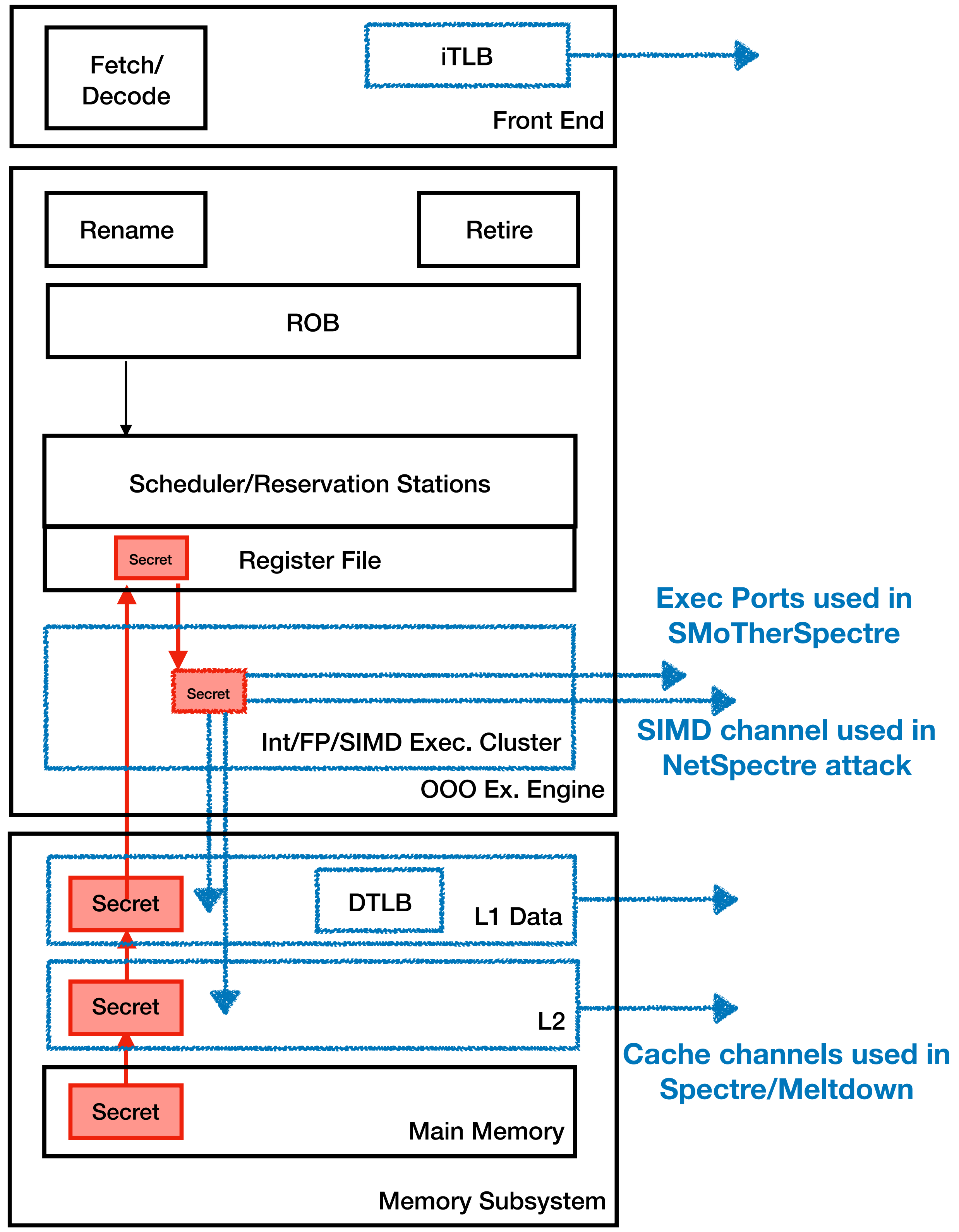
SpecShield is a family of uarch mitigation solutions with different isolation properties for trade-offs with performance

Goal: Isolate transient data from covert channel transmission

Threat Model

- Sensitive data resides anywhere in the memory hierarchy
  - Accessed through a transient misspeculated instruction
- Any covert channel can be used to exfiltrate secret data
  - Caches, SIMD units, TLBs, etc.

OOO Processor



Fetch/ Decode

iTLB

Front End

Rename

Retire

ROB

Scheduler/Reservation Stations

Secret

Register File

Secret

Int/FP/SIMD Exec. Cluster

OOO Ex. Engine

Secret

DTLB

L1 Data

Secret

L2

Secret

Main Memory

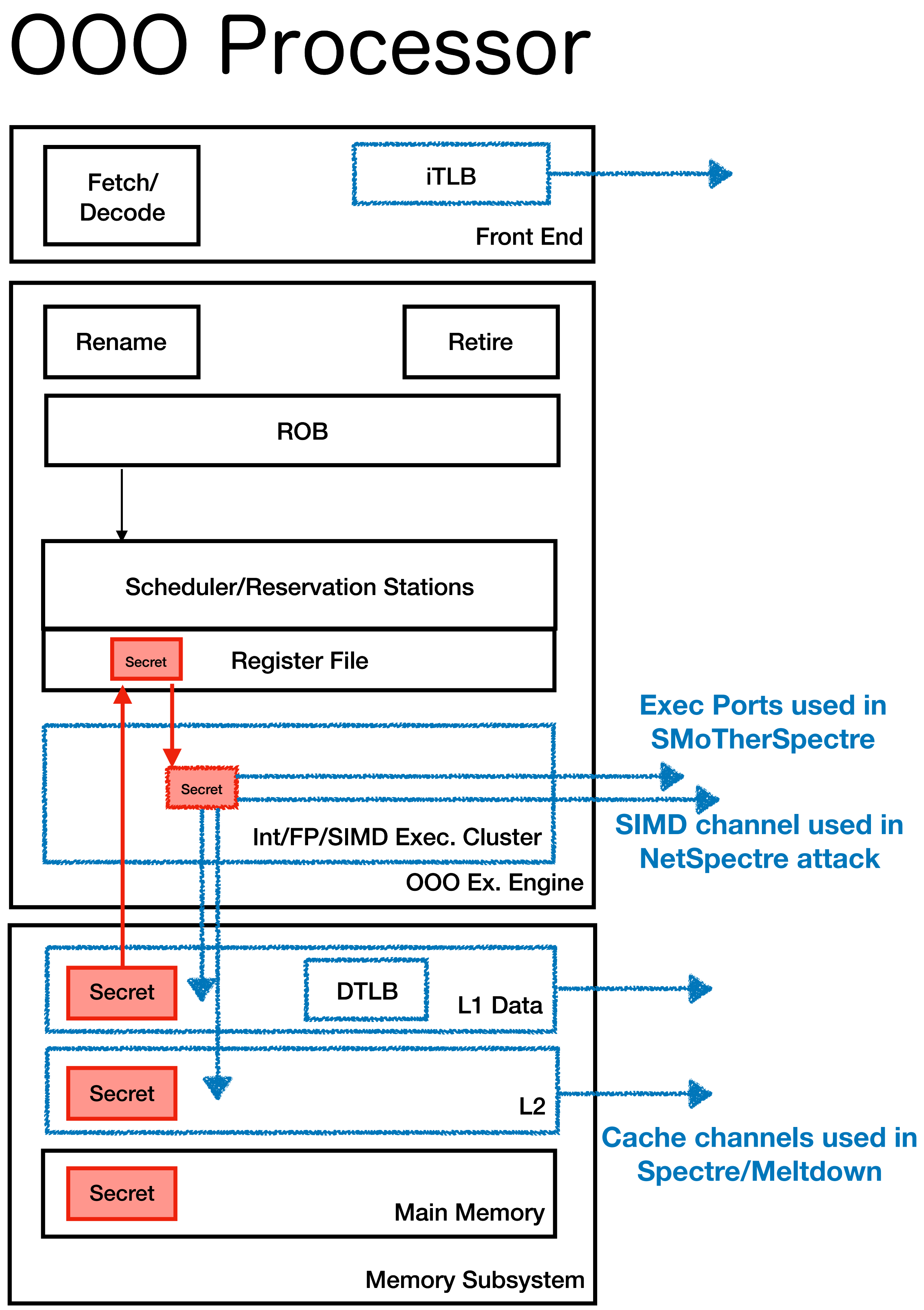
Memory Subsystem

Exec Ports used in SMoTherSpectre

SIMD channel used in NetSpectre attack

Cache channels used in Spectre/Meltdown

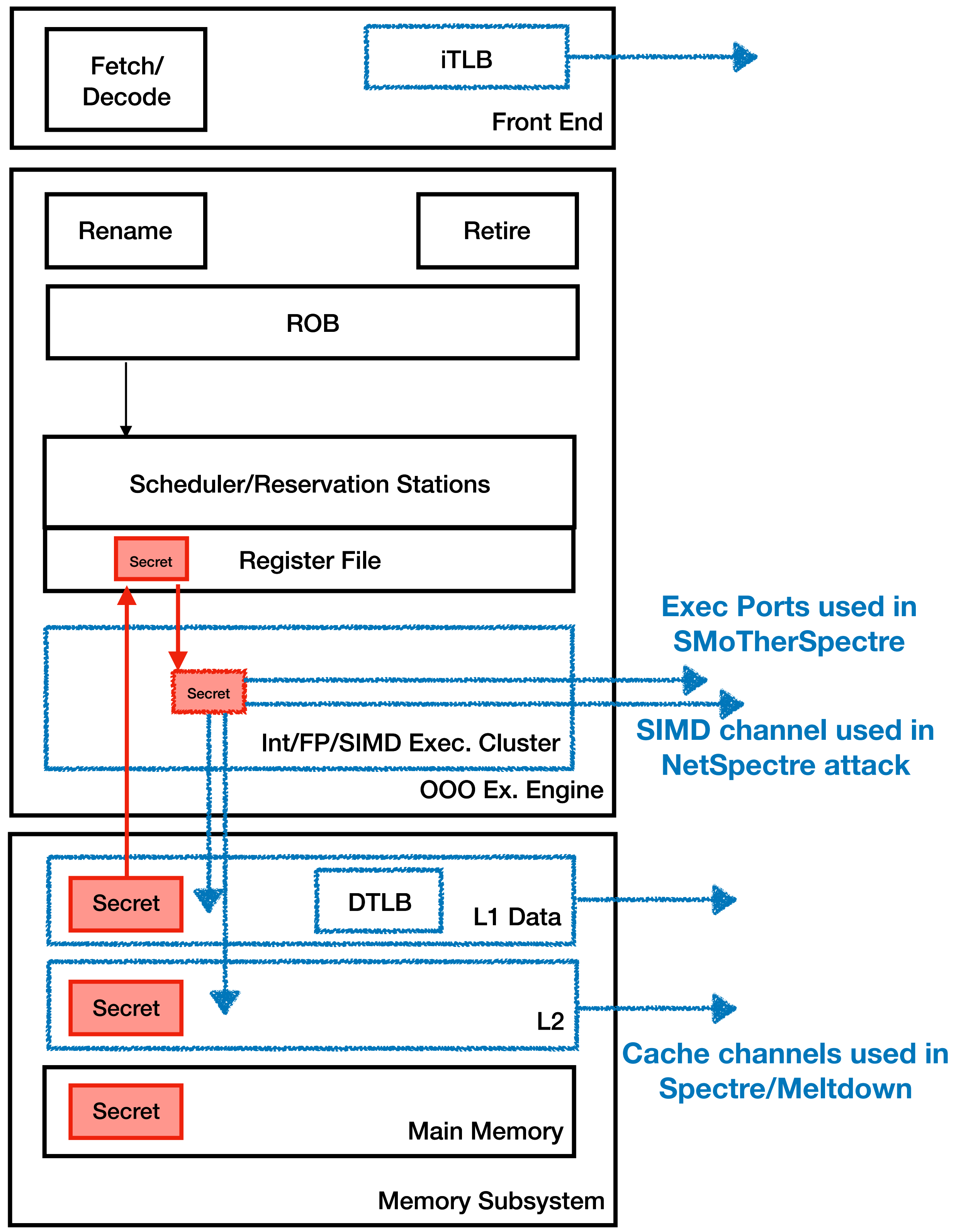
SpecShield: Main Idea


OOO Processor
Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

OOO Processor

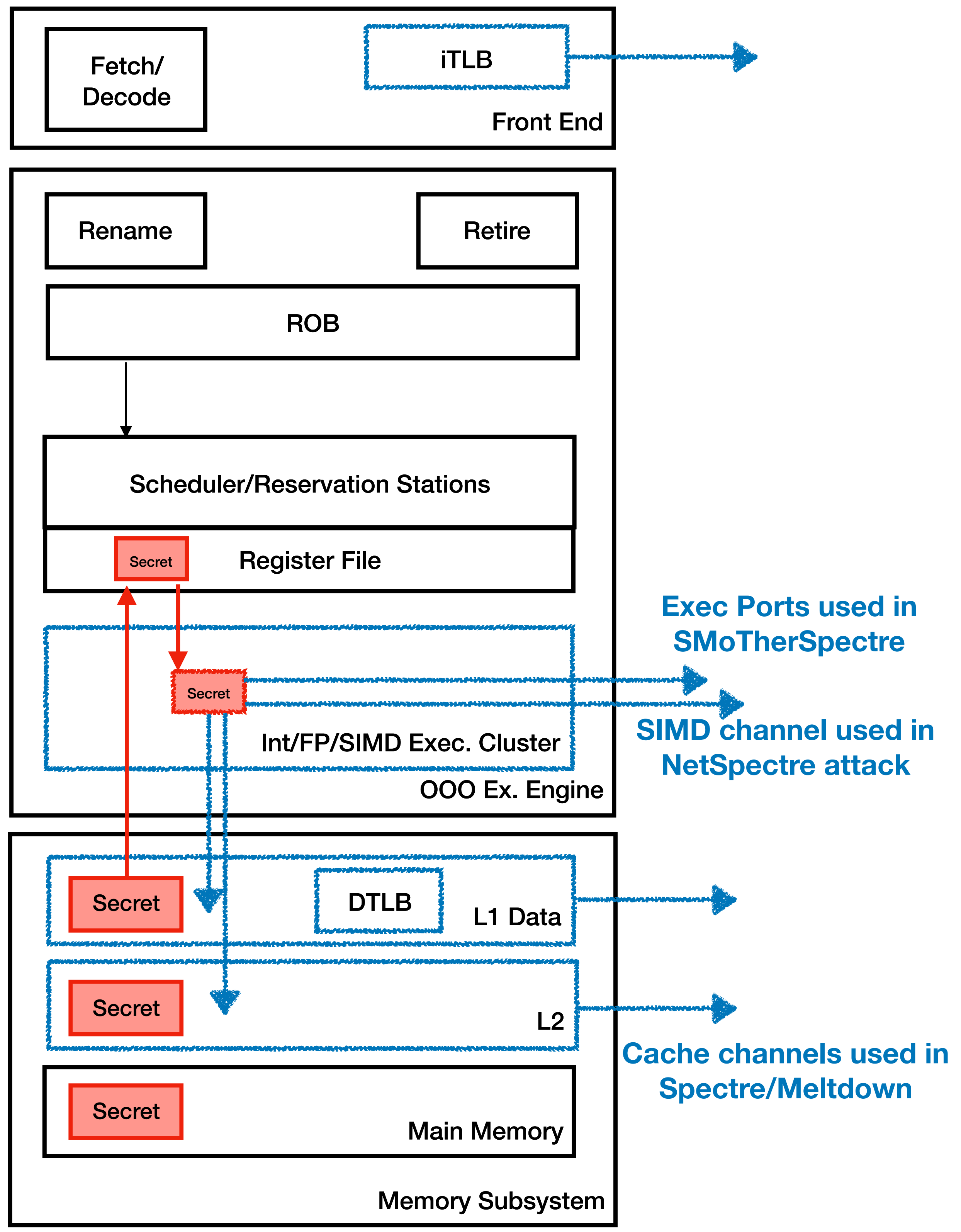

Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

Key Observation: Leakage source by definition has dependence on the secret data

OOO Processor

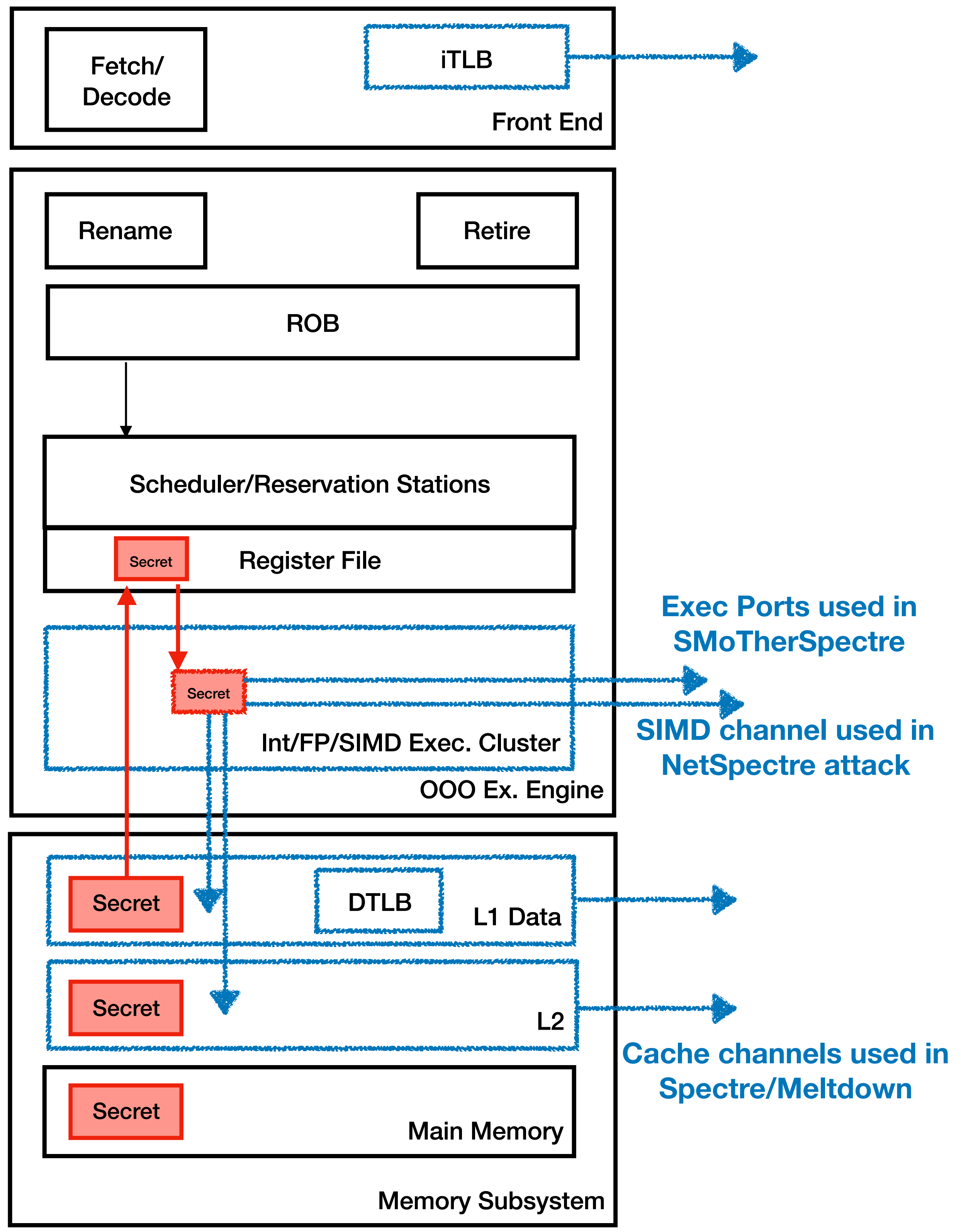

Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

Key Observation: Leakage source by definition has dependence on the secret data

OOO Processor


Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

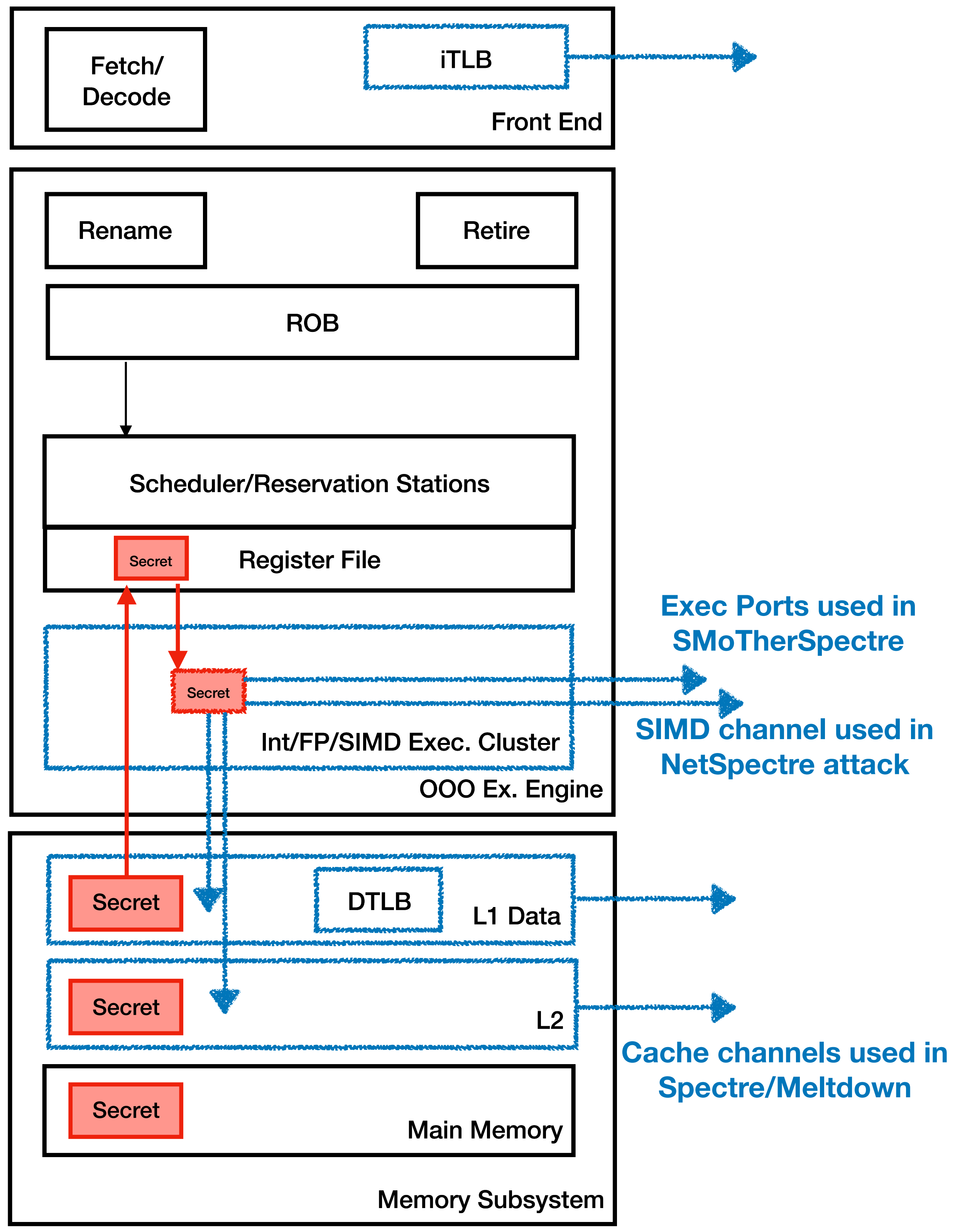
SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

Key Observation: Leakage source by definition has dependence on the secret data

- Prevent covert channel formation by policing speculative data use by dependent instructions

OOO Processor


Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

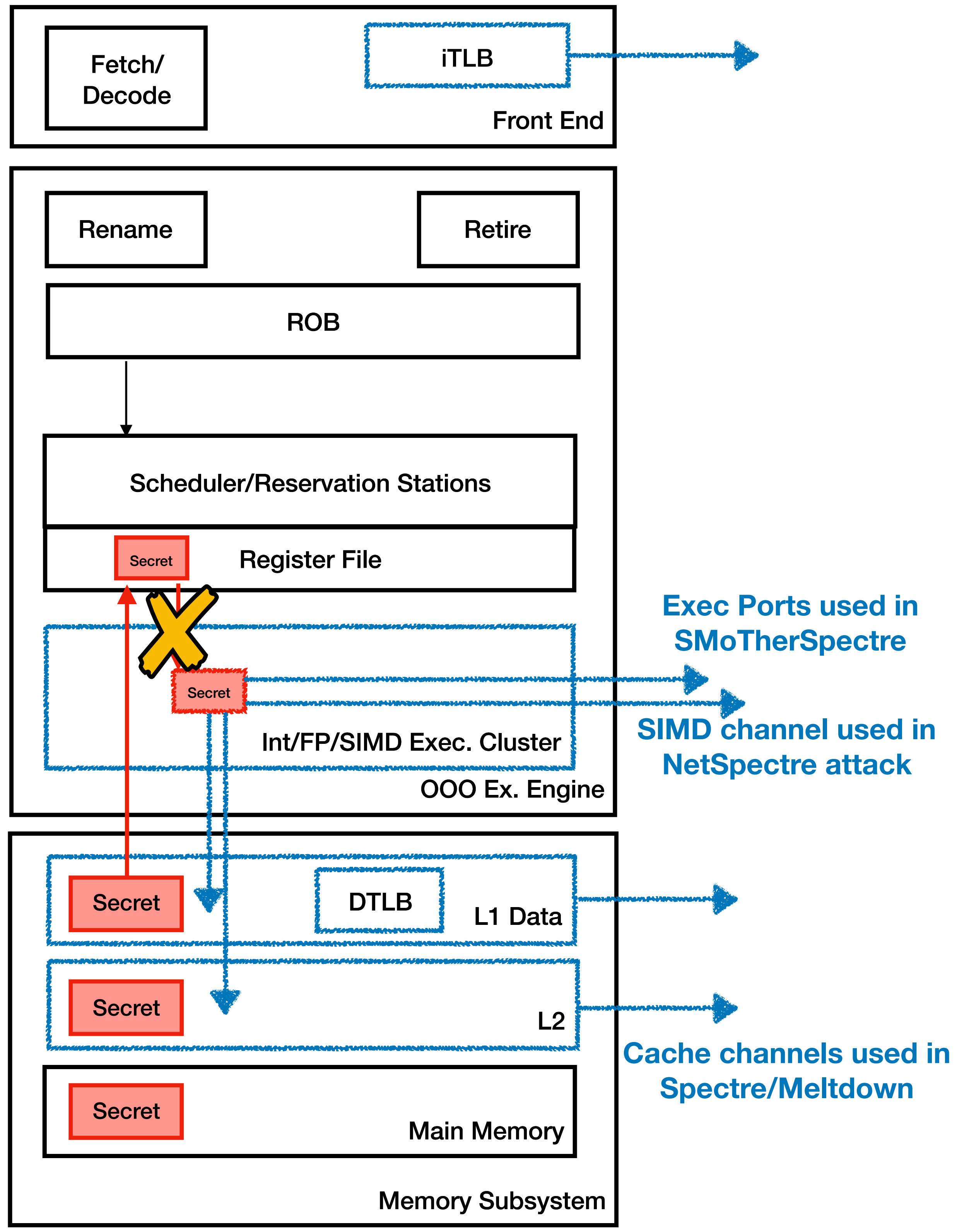
SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

Key Observation: Leakage source by definition has dependence on the secret data

- Prevent covert channel formation by policing speculative data use by dependent instructions

OOO Processor



Fetch/ Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Exec Ports used in SMoTherSpectre
SIMD channel used in NetSpectre attack
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
Cache channels used in Spectre/Meltdown

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

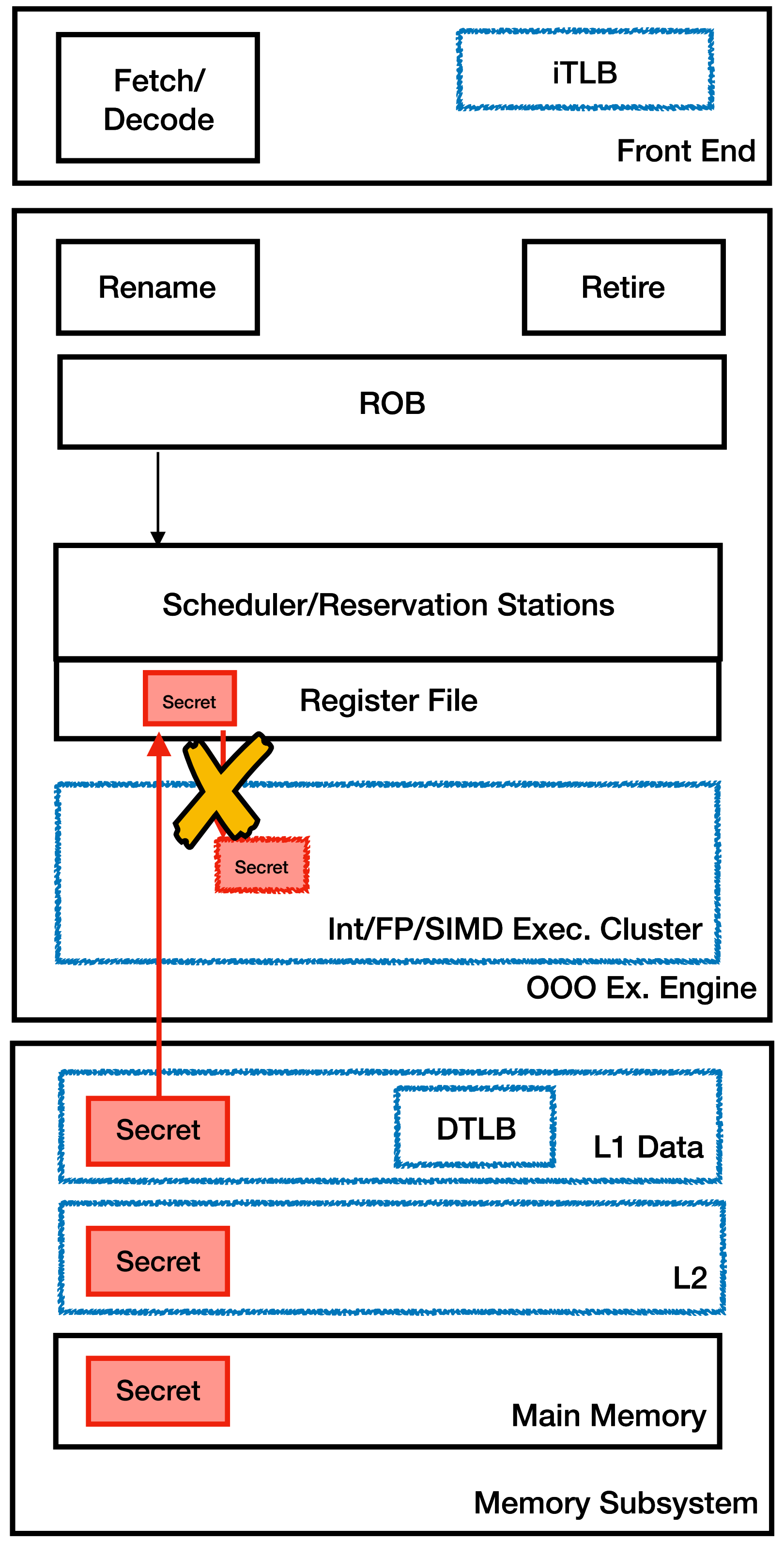
SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

Key Observation: Leakage source by definition has dependence on the secret data

- Prevent covert channel formation by policing speculative data use by dependent instructions

OOO Processor



```
if (x < array1_size)
   y = array2[array1[x] * 256];
```

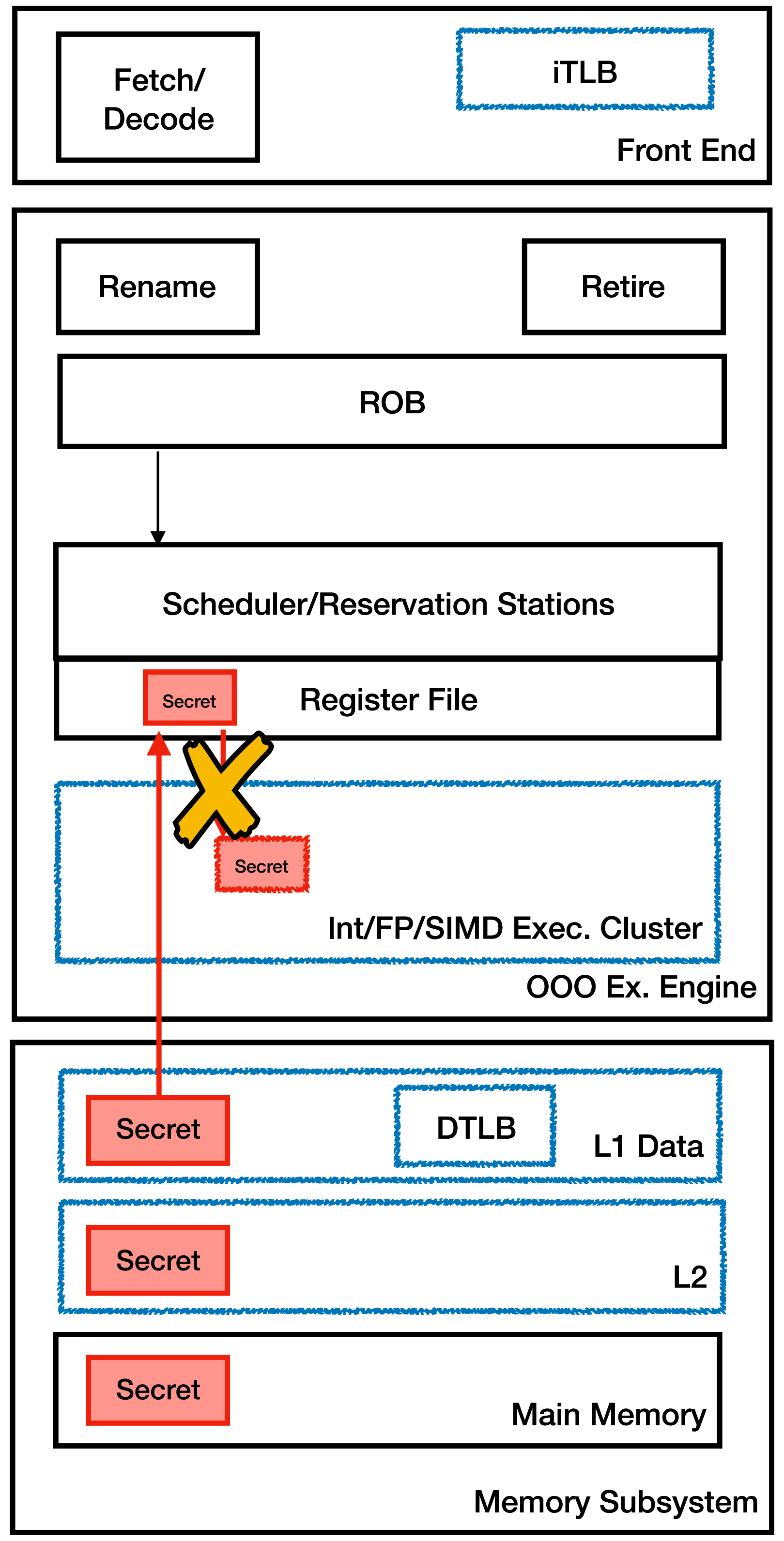
SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

Key Observation: Leakage source by definition has dependence on the secret data

- Prevent covert channel formation by policing speculative data use by dependent instructions
  - Speculative status determined by producing instruction

OOO Processor



```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

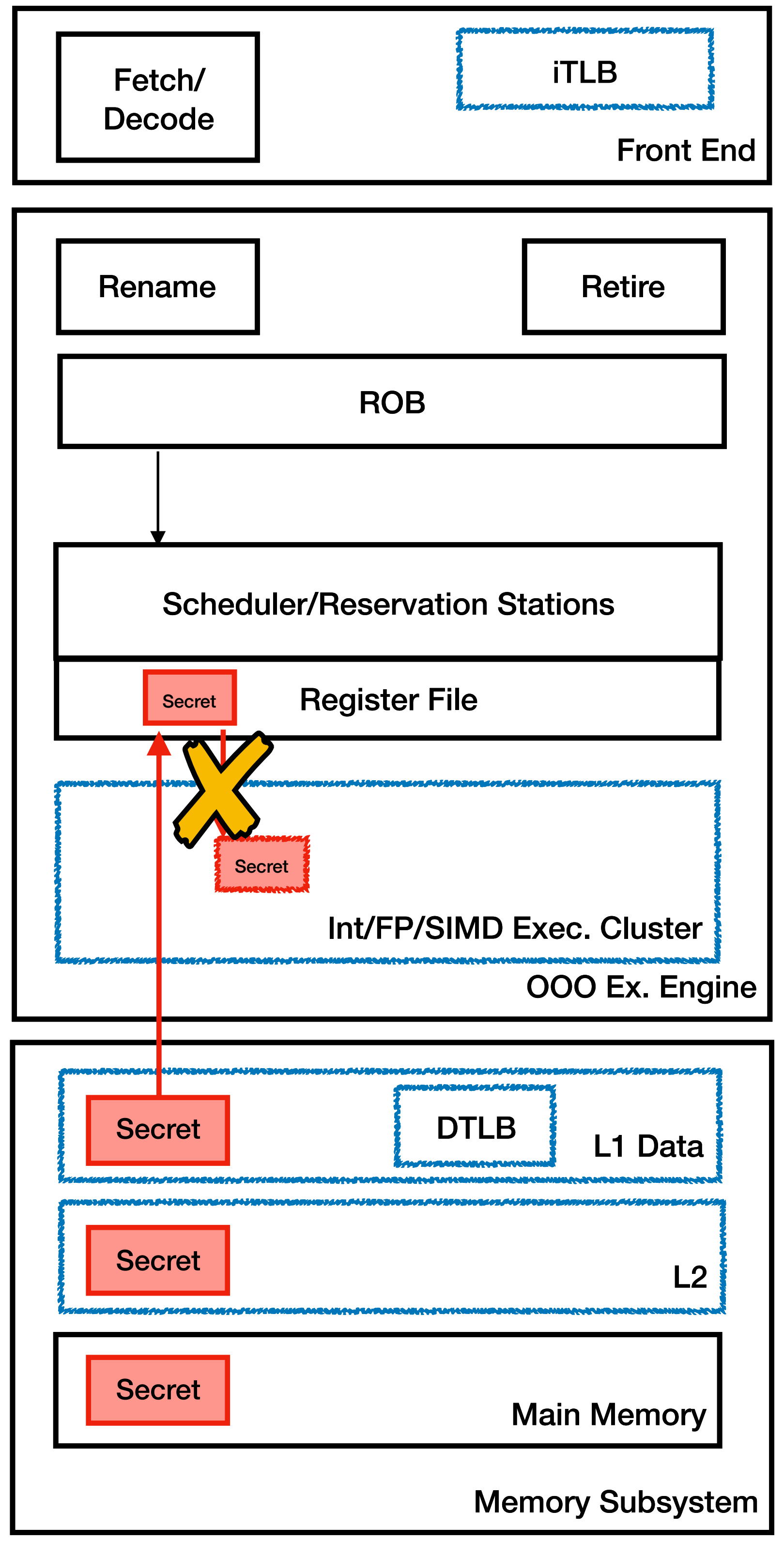
SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

Key Observation: Leakage source by definition has dependence on the secret data

- Prevent covert channel formation by policing speculative data use by dependent instructions
  - Speculative status determined by producing instruction
  - Monitor speculative status of loads

OOO Processor



```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

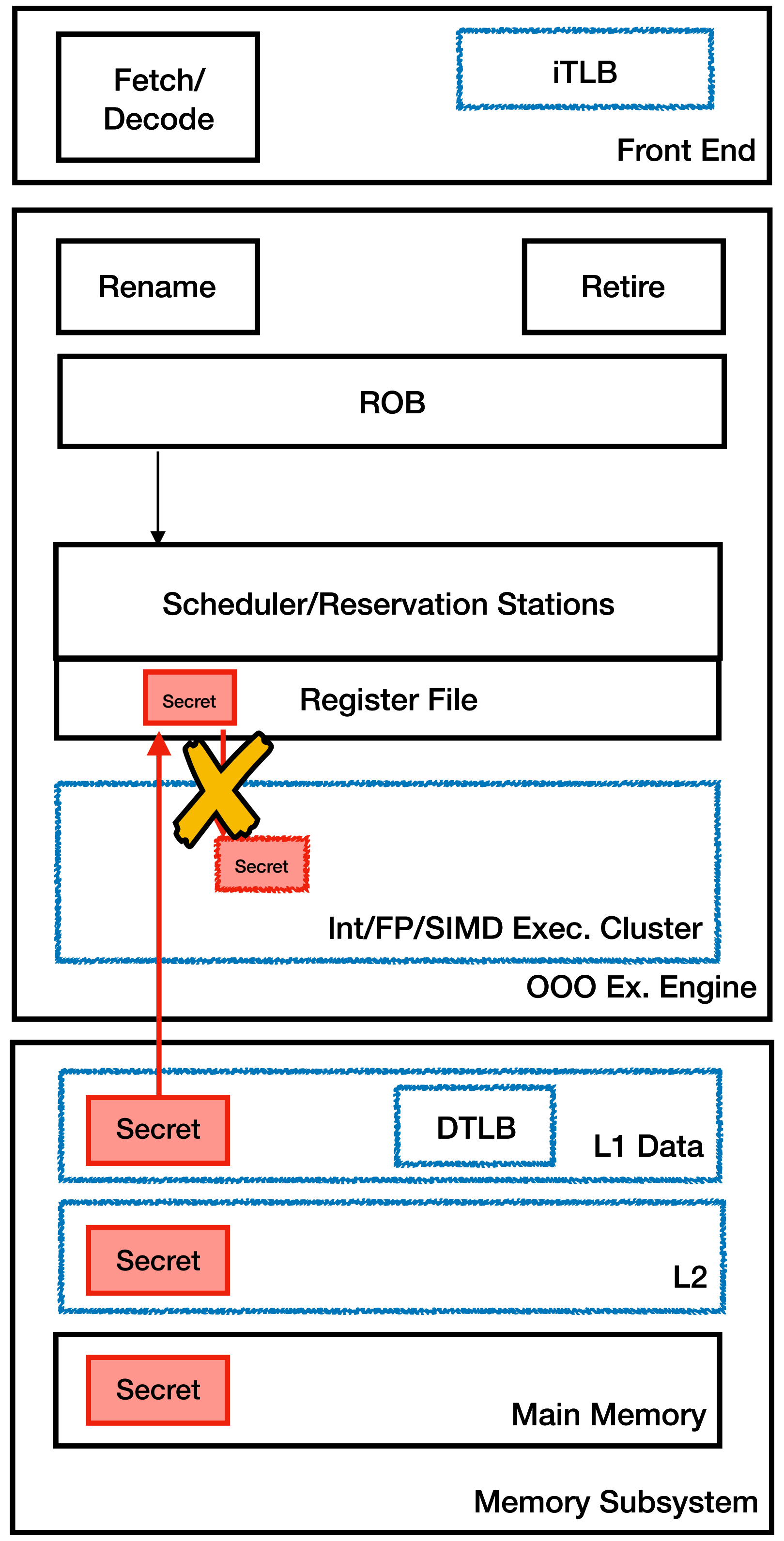
SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

Key Observation: Leakage source by definition has dependence on the secret data

- Prevent covert channel formation by policing speculative data use by dependent instructions
  - Speculative status determined by producing instruction
  - Monitor speculative status of loads
- Delay forwarding until window of speculation is closed

OOO Processor



```
if (x < array1_size)
    y = array2[array1[x] *  256];
```

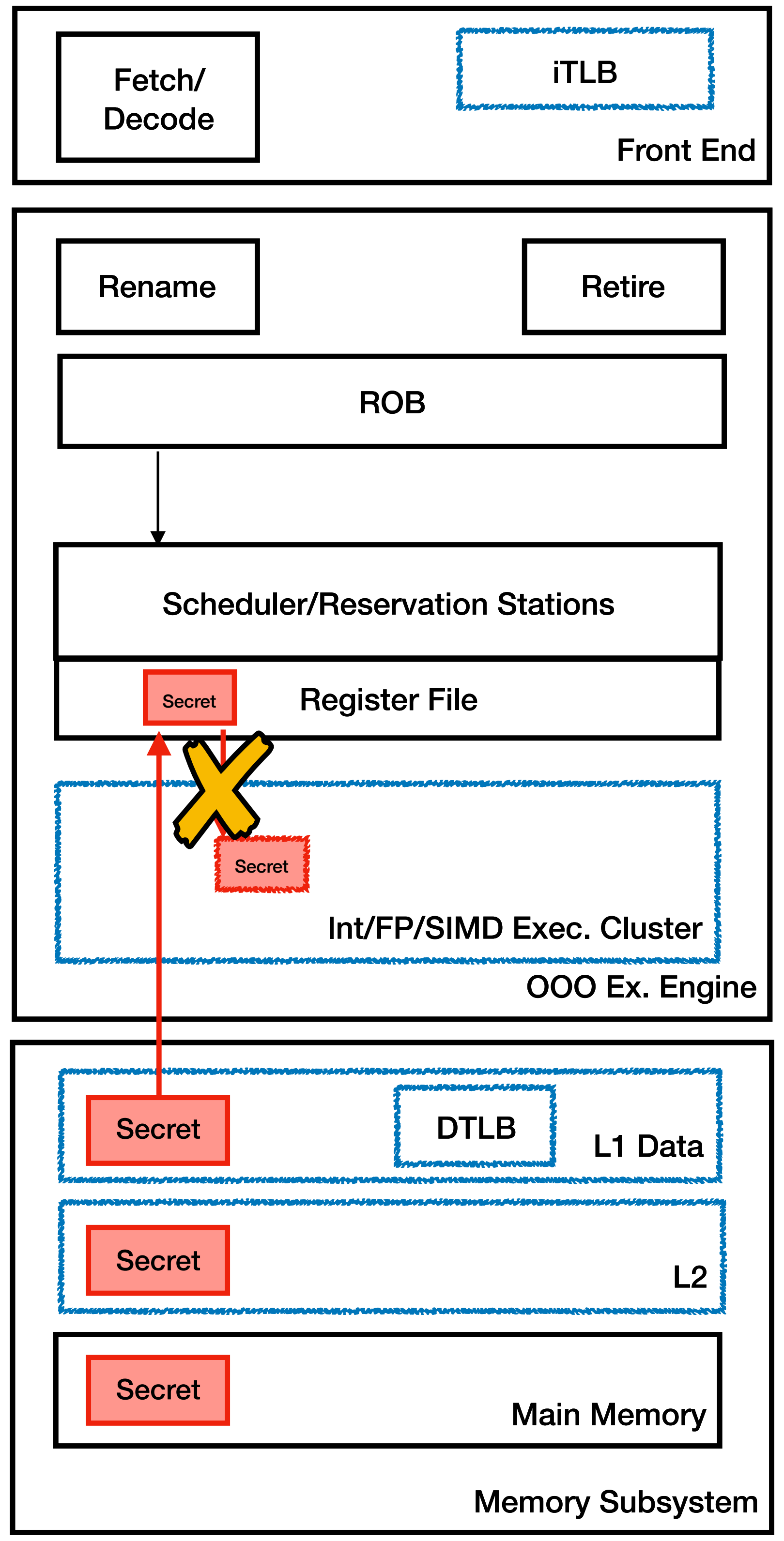
SpecShield: Main Idea

A more **general** solution that prevents covert channel formation

Key Observation: Leakage source by definition has dependence on the secret data

- Prevent covert channel formation by policing speculative data use by dependent instructions
  - Speculative status determined by producing instruction
  - Monitor speculative status of loads
- Delay forwarding until window of speculation is closed
- Traditionally, instructions considered non-speculative when reaching ROB head

OOO Processor



```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

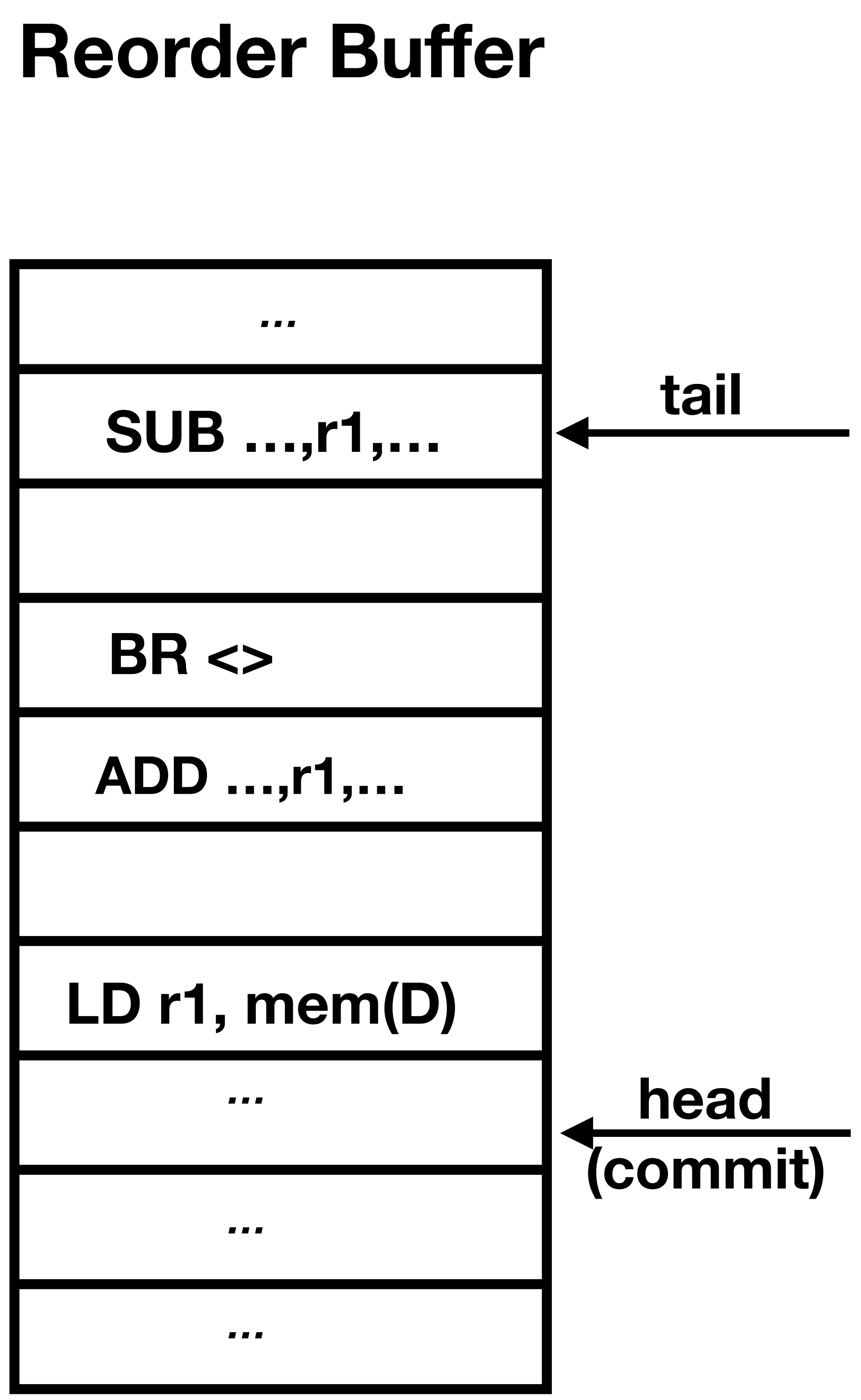
Conservative Design: SpecShield STL

Conservative Design: SpecShield STL

- Wait until load reaches ROB head before forwarding to dependent instruction

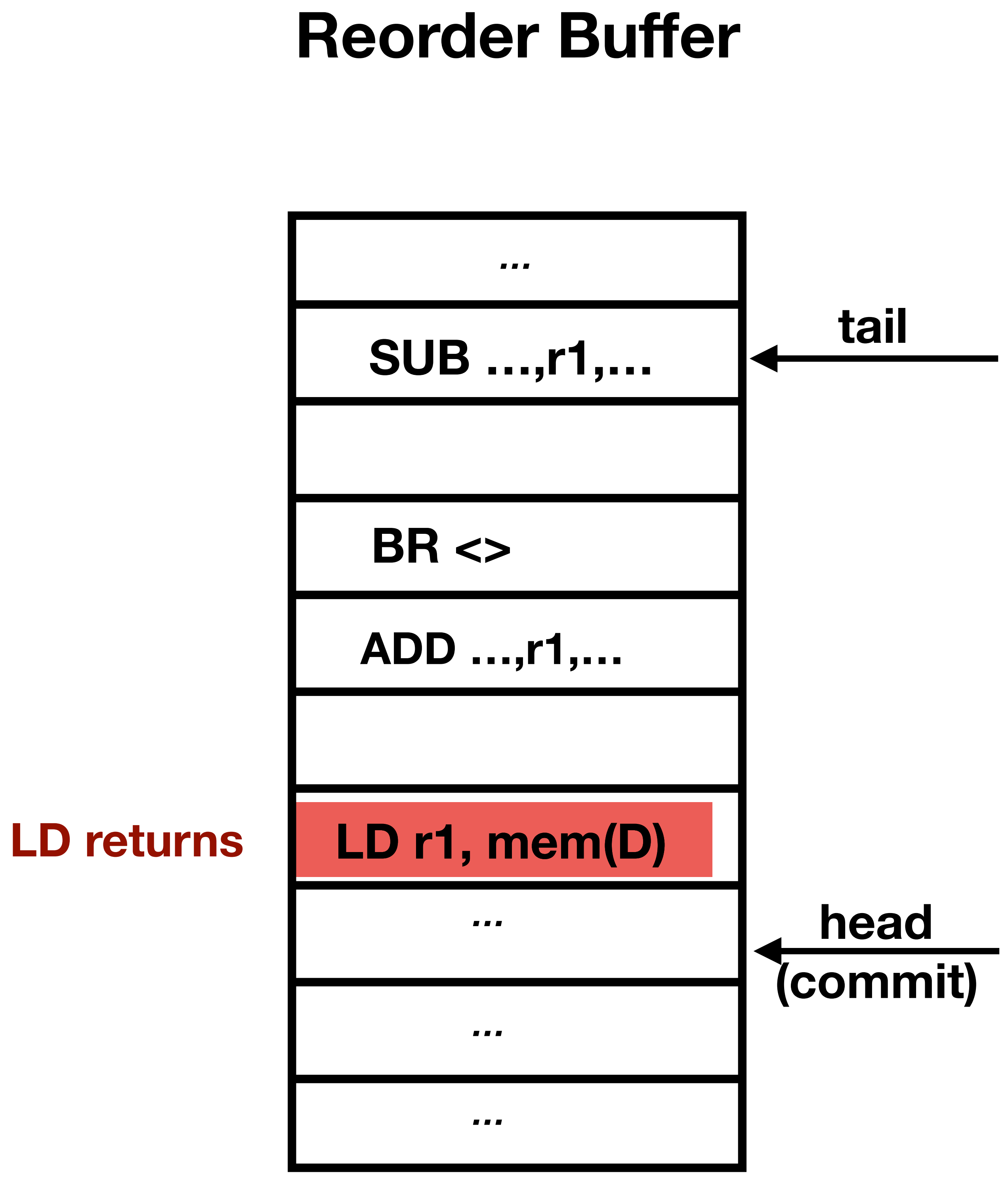
Conservative Design: SpecShield STL

- Wait until load reaches ROB head before forwarding to dependent instruction



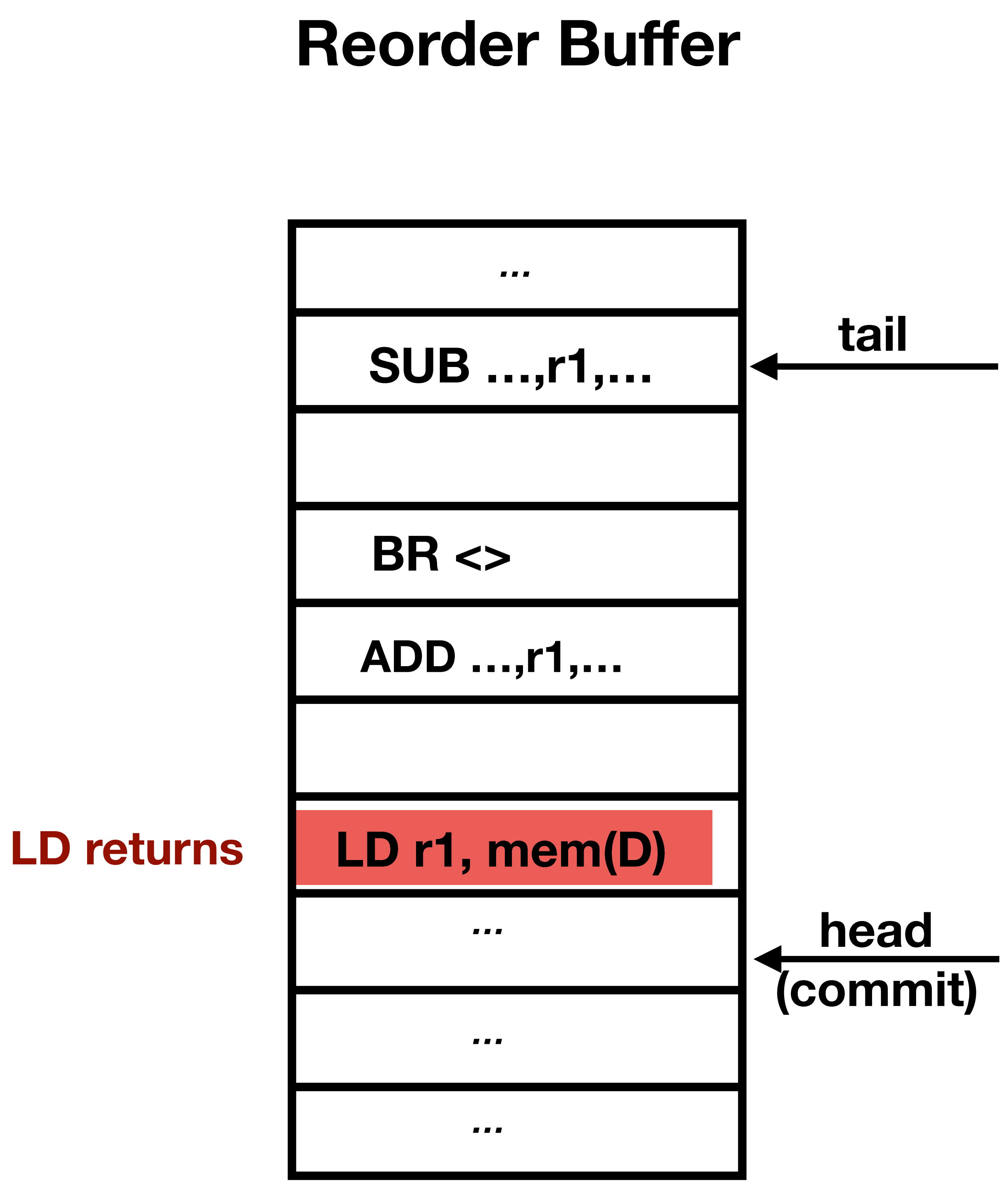
Conservative Design: SpecShield STL

- Wait until load reaches ROB head before forwarding to dependent instruction
- When data returns from memory (cache)



Conservative Design: SpecShield STL

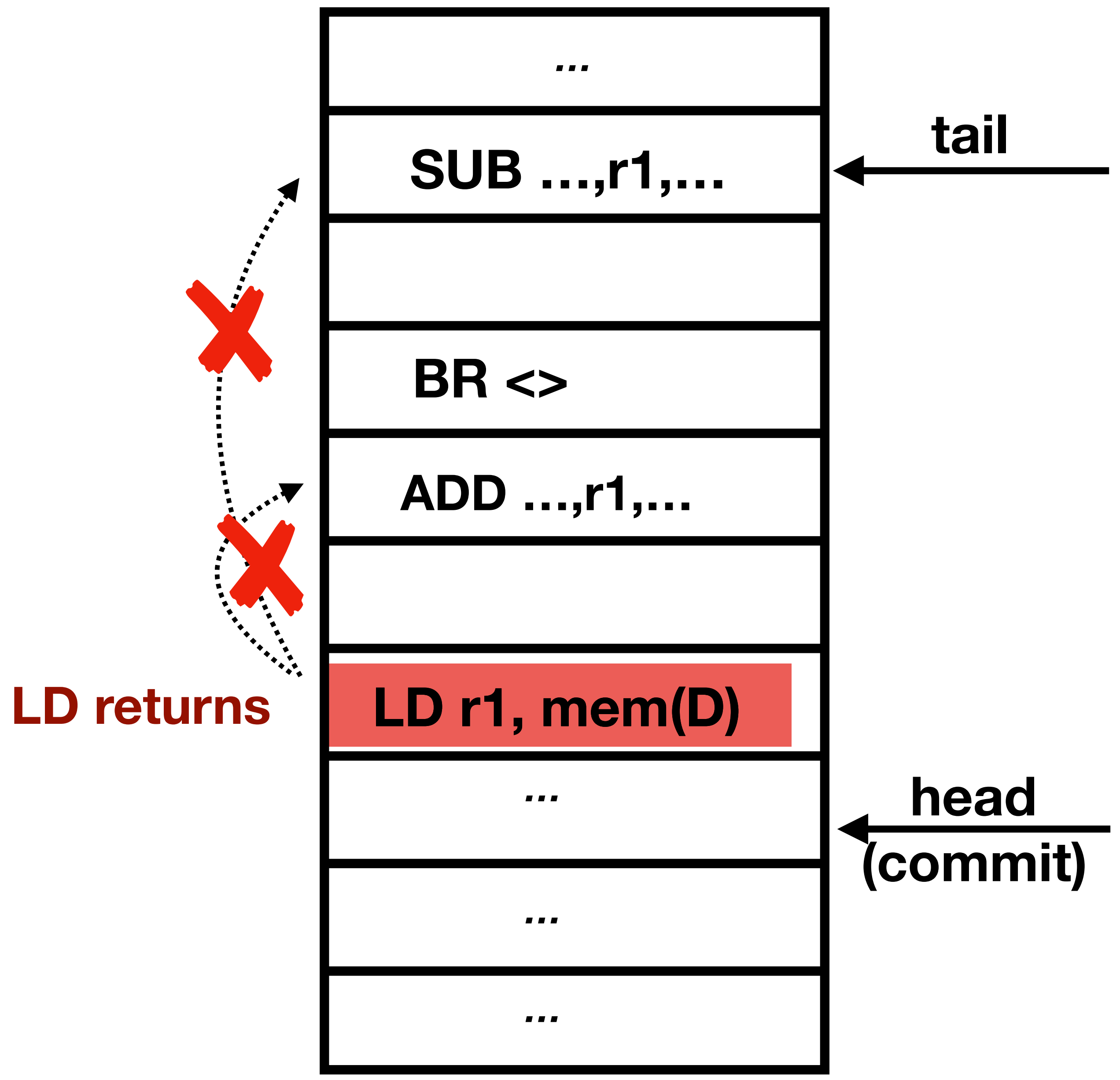
- Wait until load reaches ROB head before forwarding to dependent instruction
- When data returns from memory (cache)
  - Register file is updated



Conservative Design: SpecShield STL

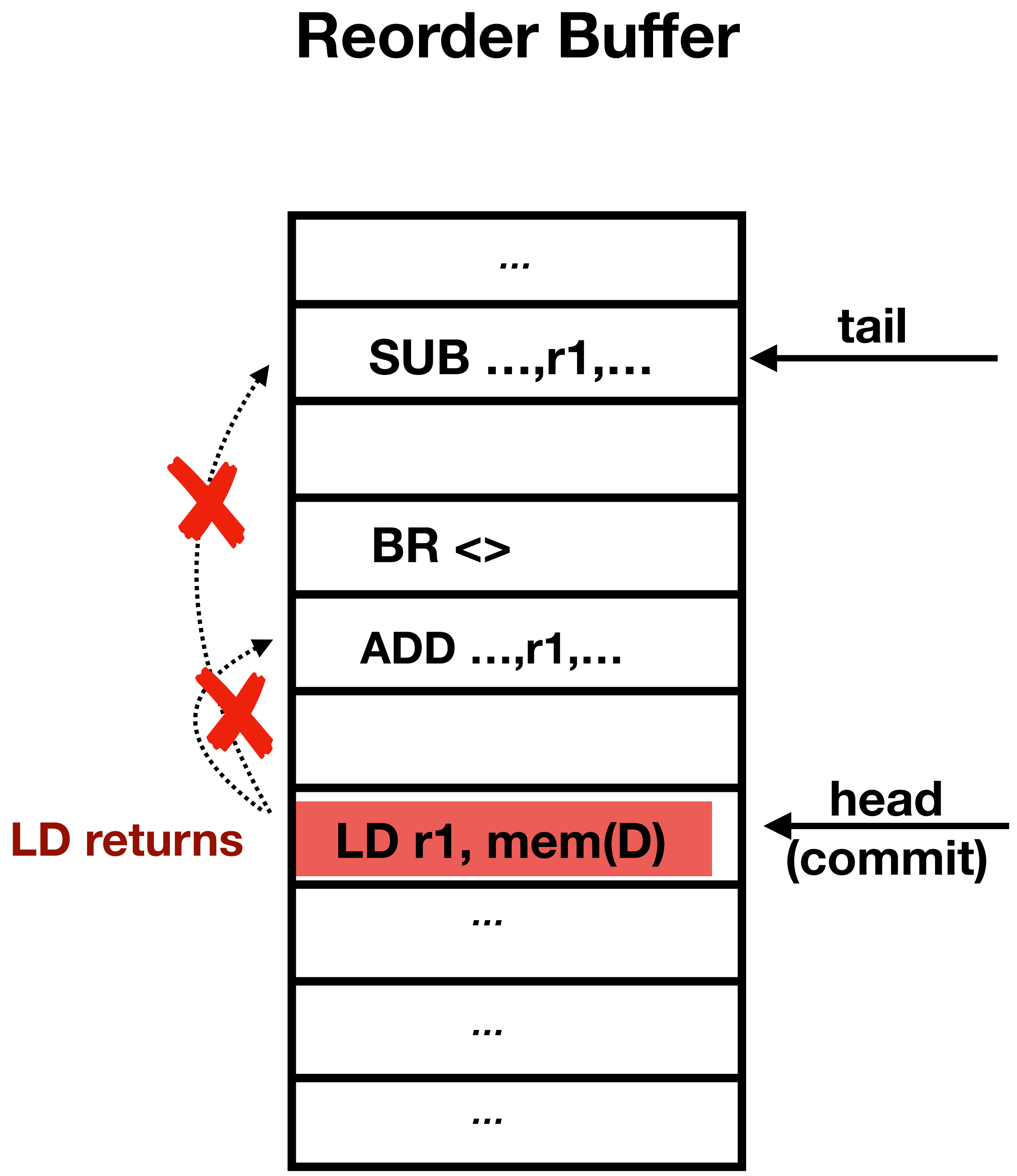
- Wait until load reaches ROB head before forwarding to dependent instruction
- When data returns from memory (cache)
  - Register file is updated
  - Delay forwarding data to dependent instructions

Reorder Buffer



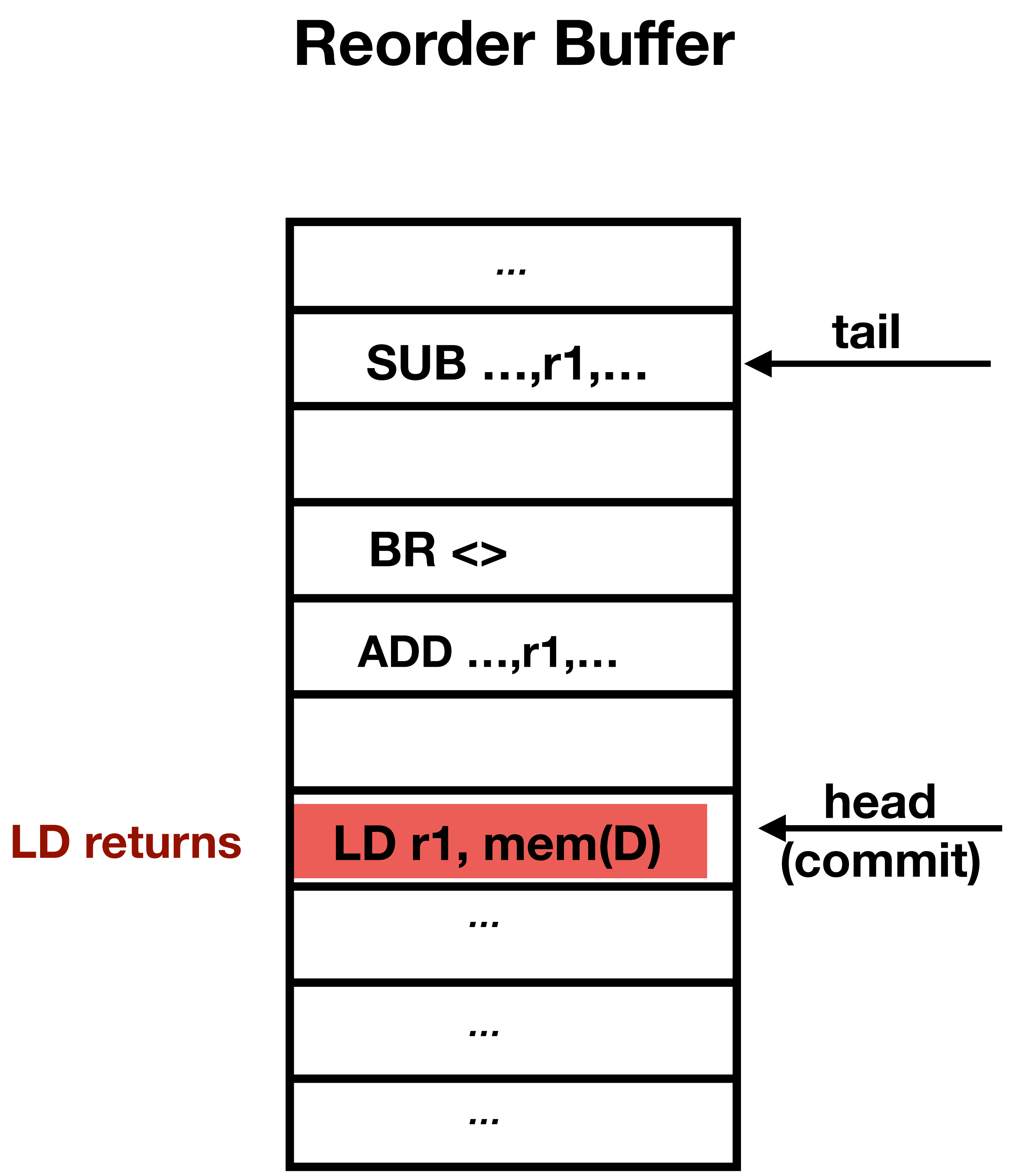
Conservative Design: SpecShield STL

- Wait until load reaches ROB head before forwarding to dependent instruction
- When data returns from memory (cache)
  - Register file is updated
  - Delay forwarding data to dependent instructions



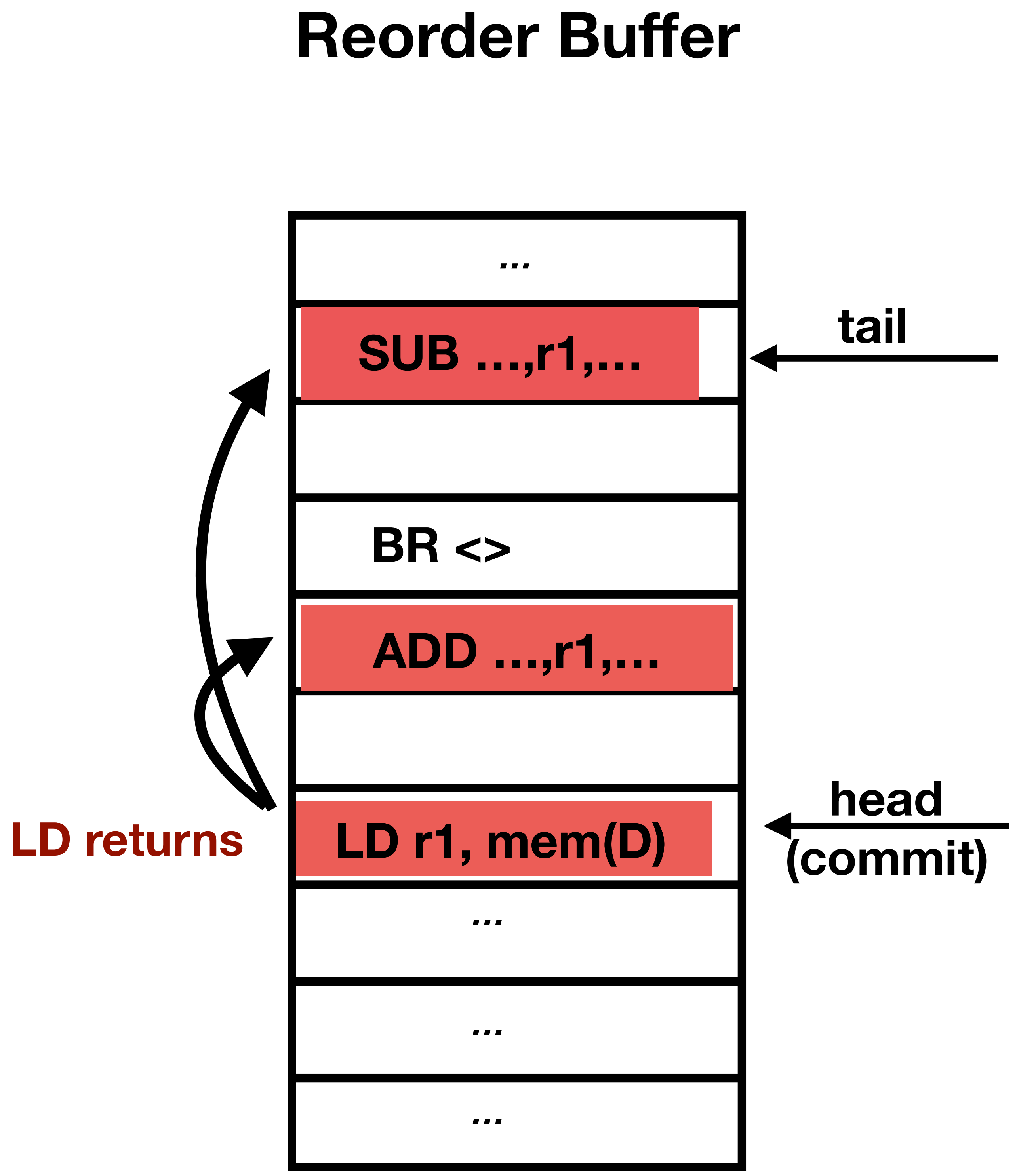
Conservative Design: SpecShield STL

- Wait until load reaches ROB head before forwarding to dependent instruction
- When data returns from memory (cache)
  - Register file is updated
  - Delay forwarding data to dependent instructions



Conservative Design: SpecShield STL

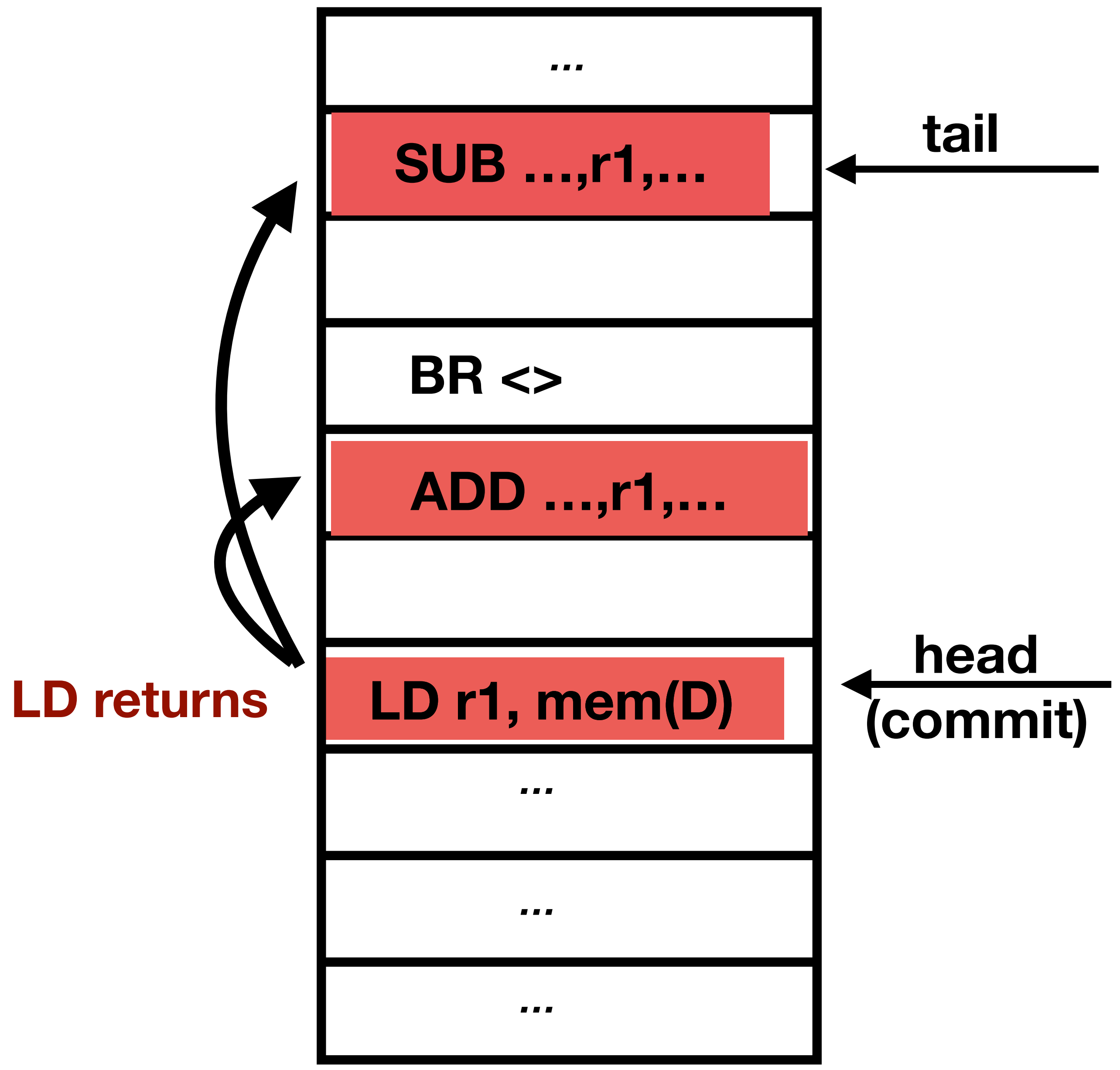
- Wait until load reaches ROB head before forwarding to dependent instruction
- When data returns from memory (cache)
  - Register file is updated
  - Delay forwarding data to dependent instructions



Conservative Design: SpecShield STL

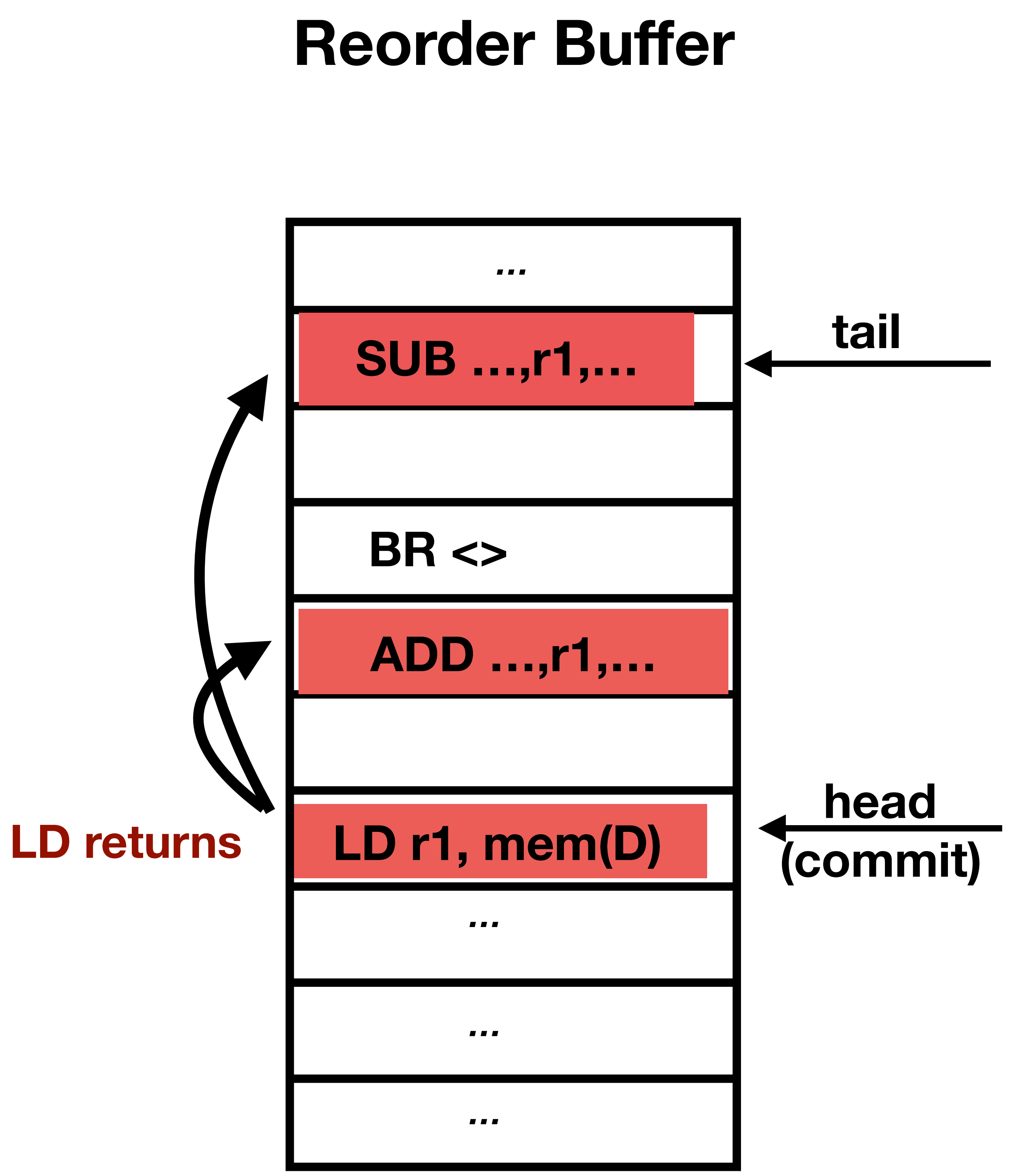
- Wait until load reaches ROB head before forwarding to dependent instruction
- When data returns from memory (cache)
  - Register file is updated
  - Delay forwarding data to dependent instructions
- All data guaranteed to be non-speculative before use

Reorder Buffer



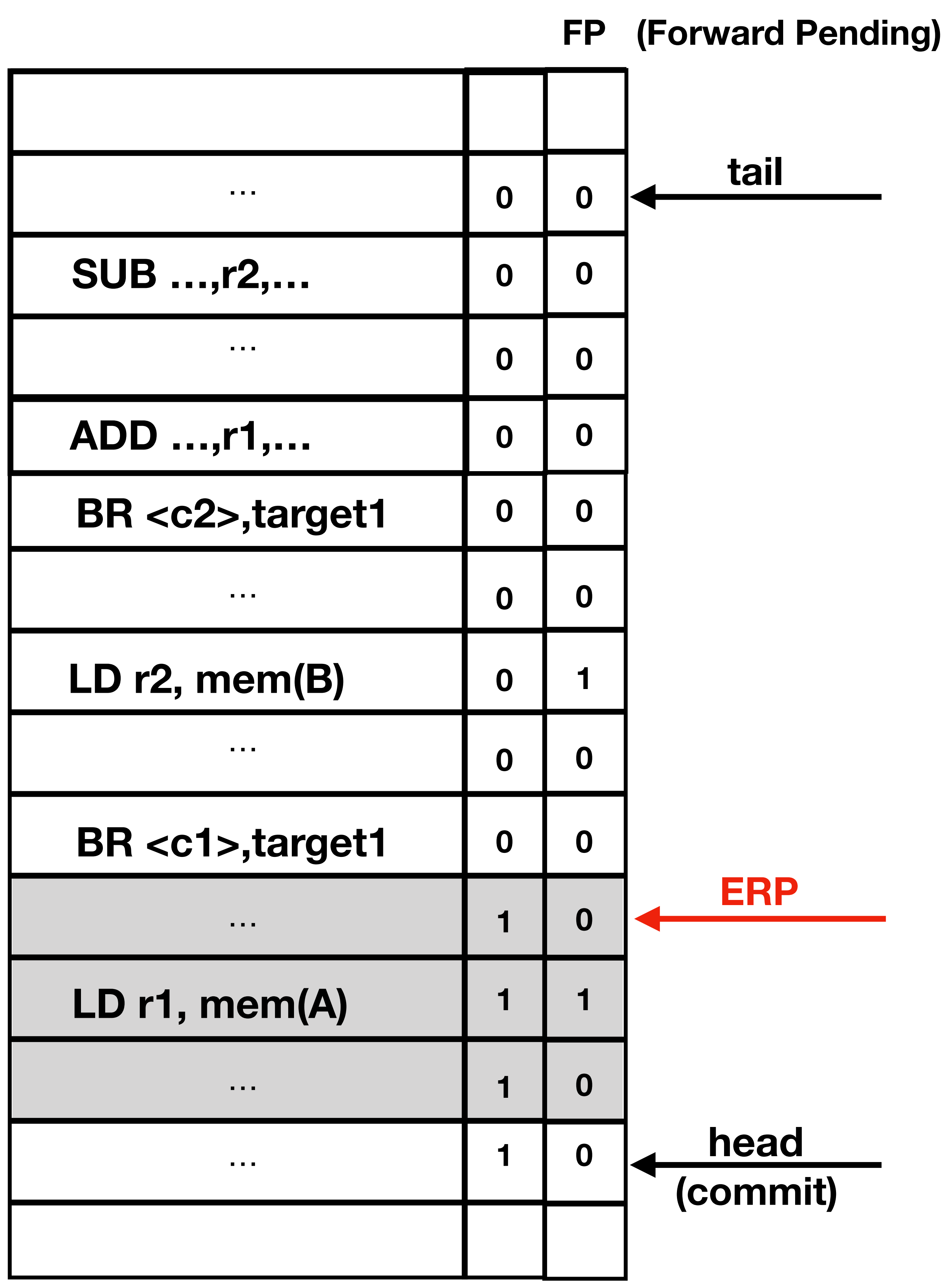
Conservative Design: SpecShield STL

- Wait until load reaches ROB head before forwarding to dependent instruction
- When data returns from memory (cache)
  - Register file is updated
  - Delay forwarding data to dependent instructions
- All data guaranteed to be non-speculative before use
- Downside: relatively large performance impact



SpecShield ERP

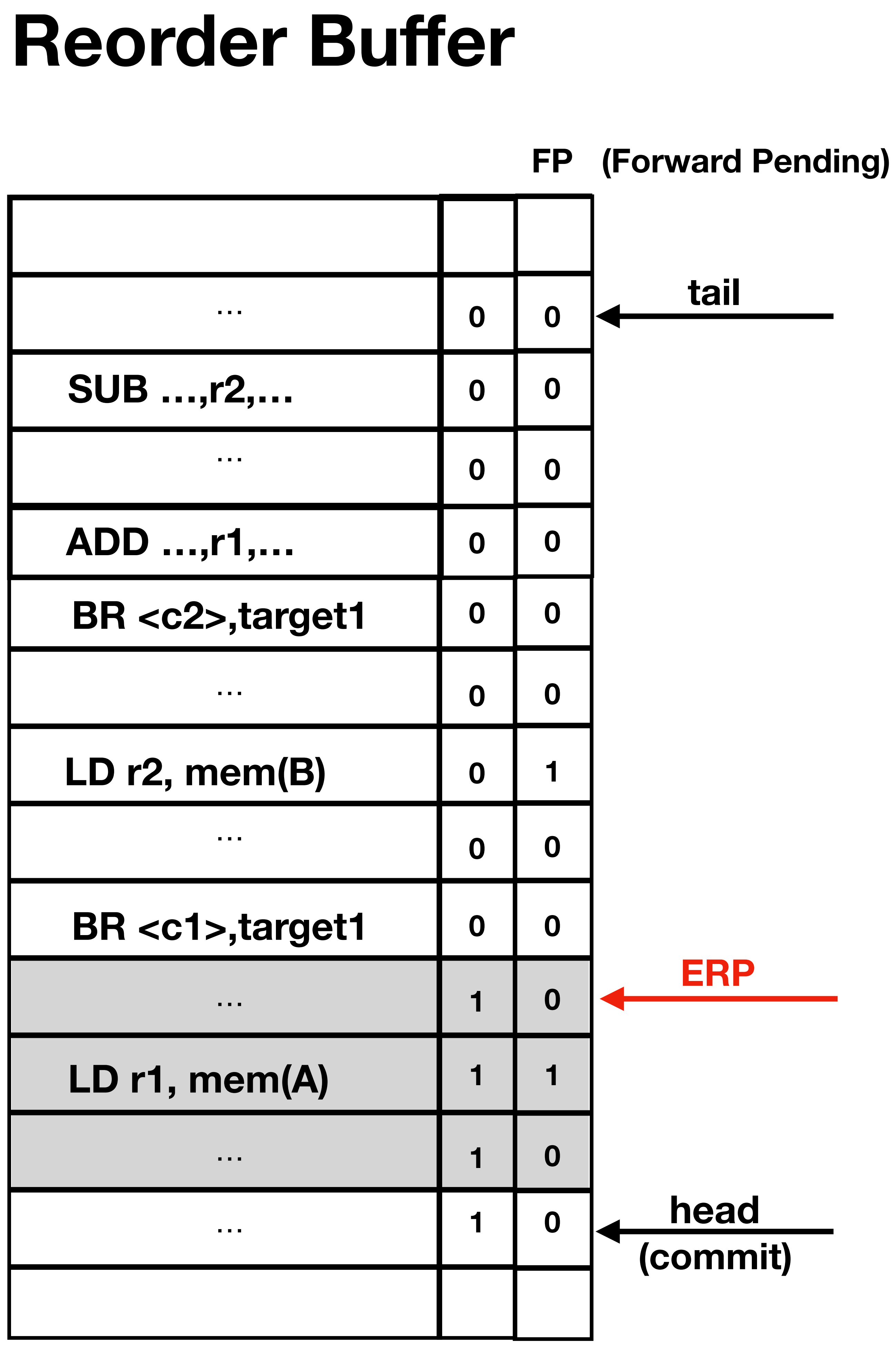
Reorder Buffer



FP (Forward Pending)

…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP



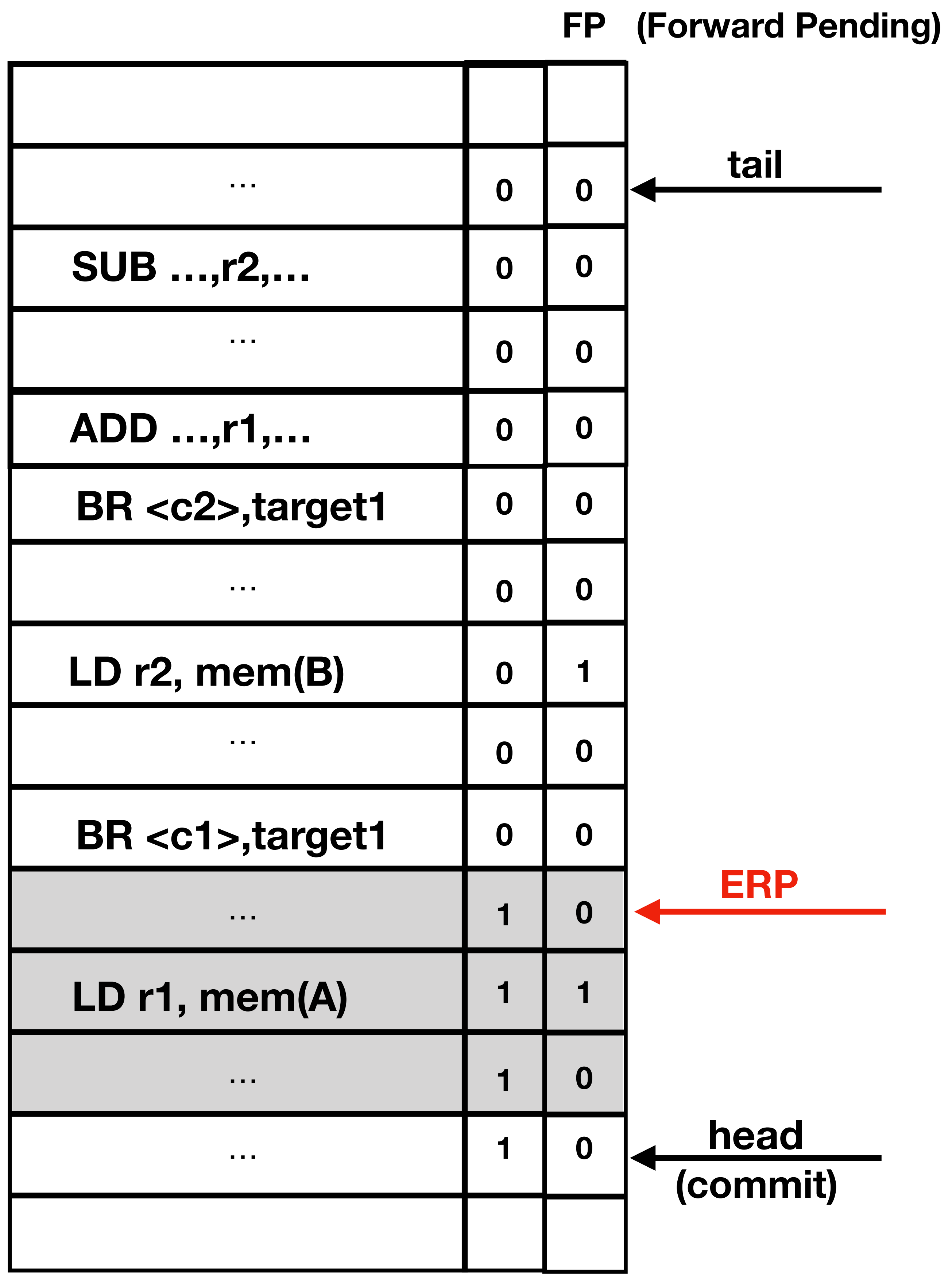
Reorder Buffer

		FP	(Forward Pending)
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding

Reorder Buffer

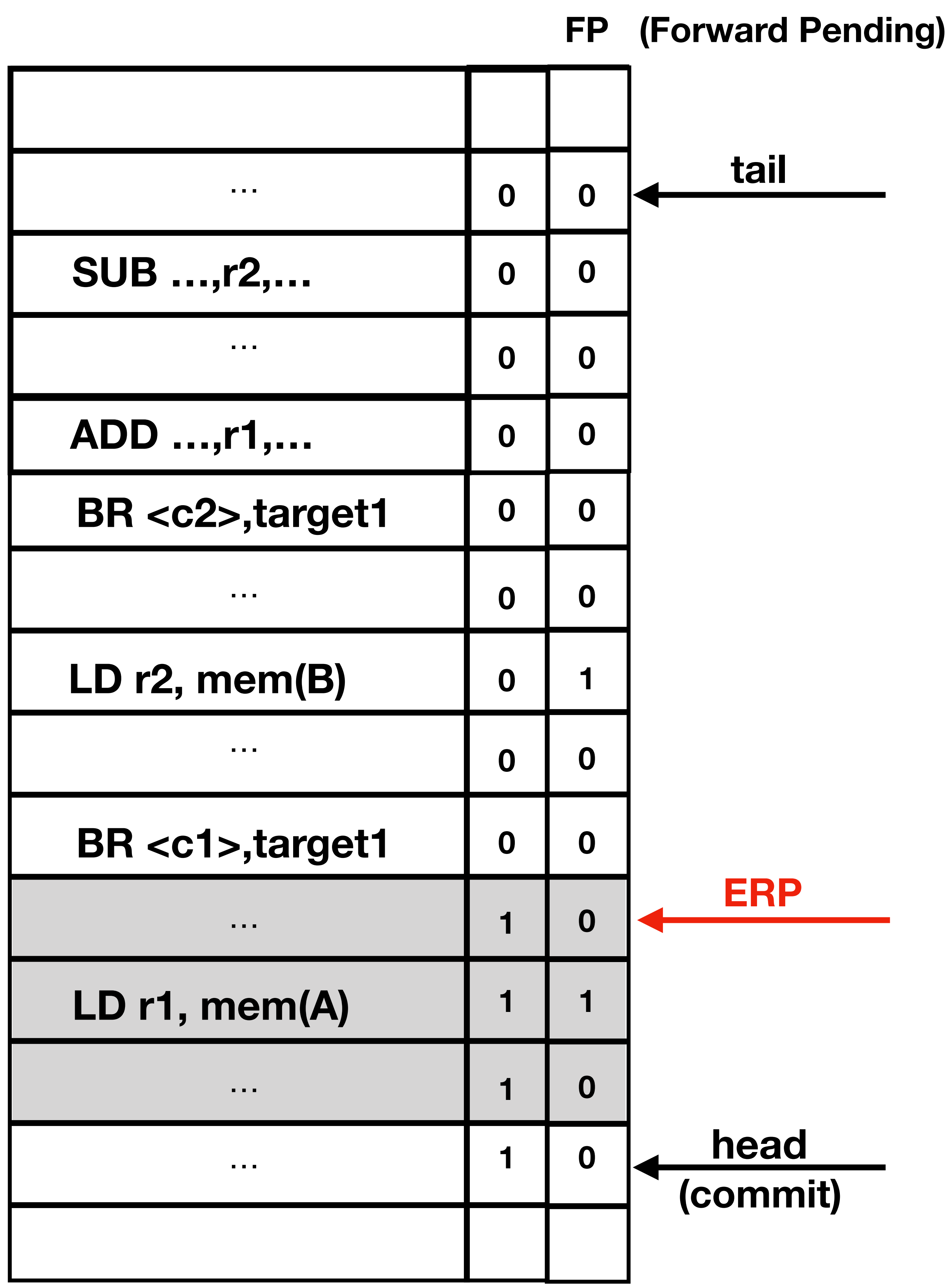


		FP (Forward Pending)	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement

Reorder Buffer

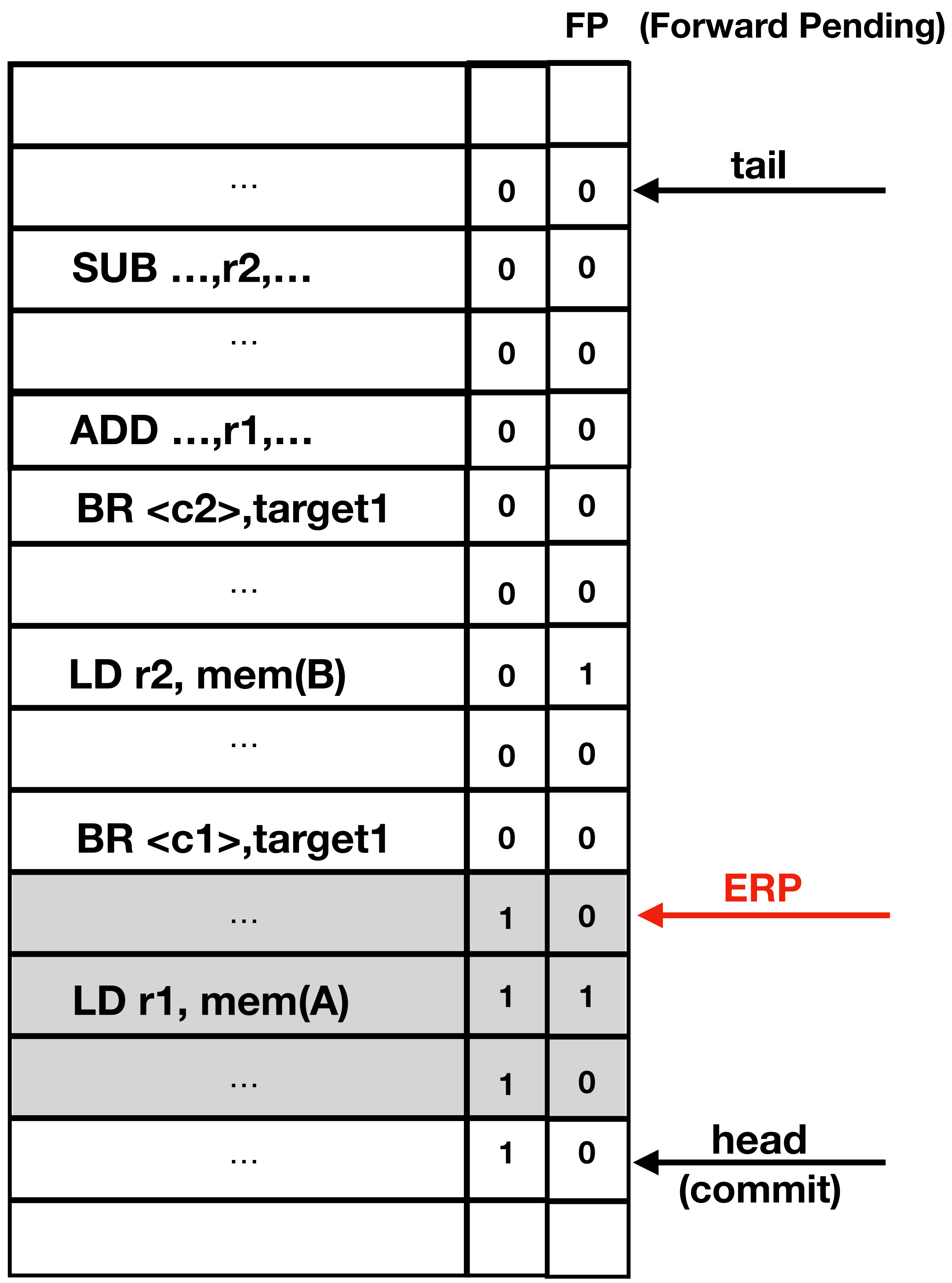


		FP (Forward Pending)	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:

Reorder Buffer

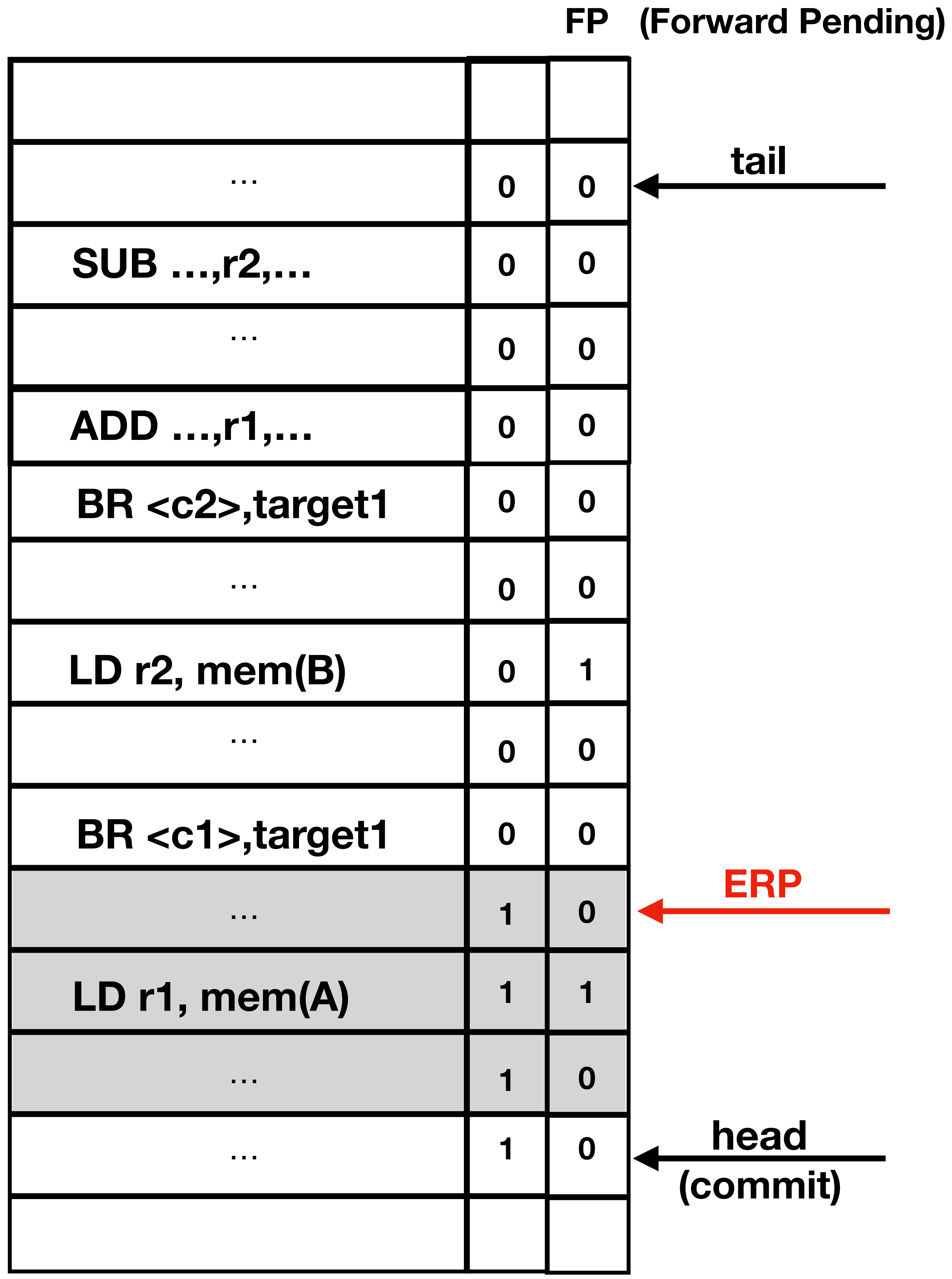


		FP	(Forward Pending)
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
    - All older branch instructions have resolved

Reorder Buffer

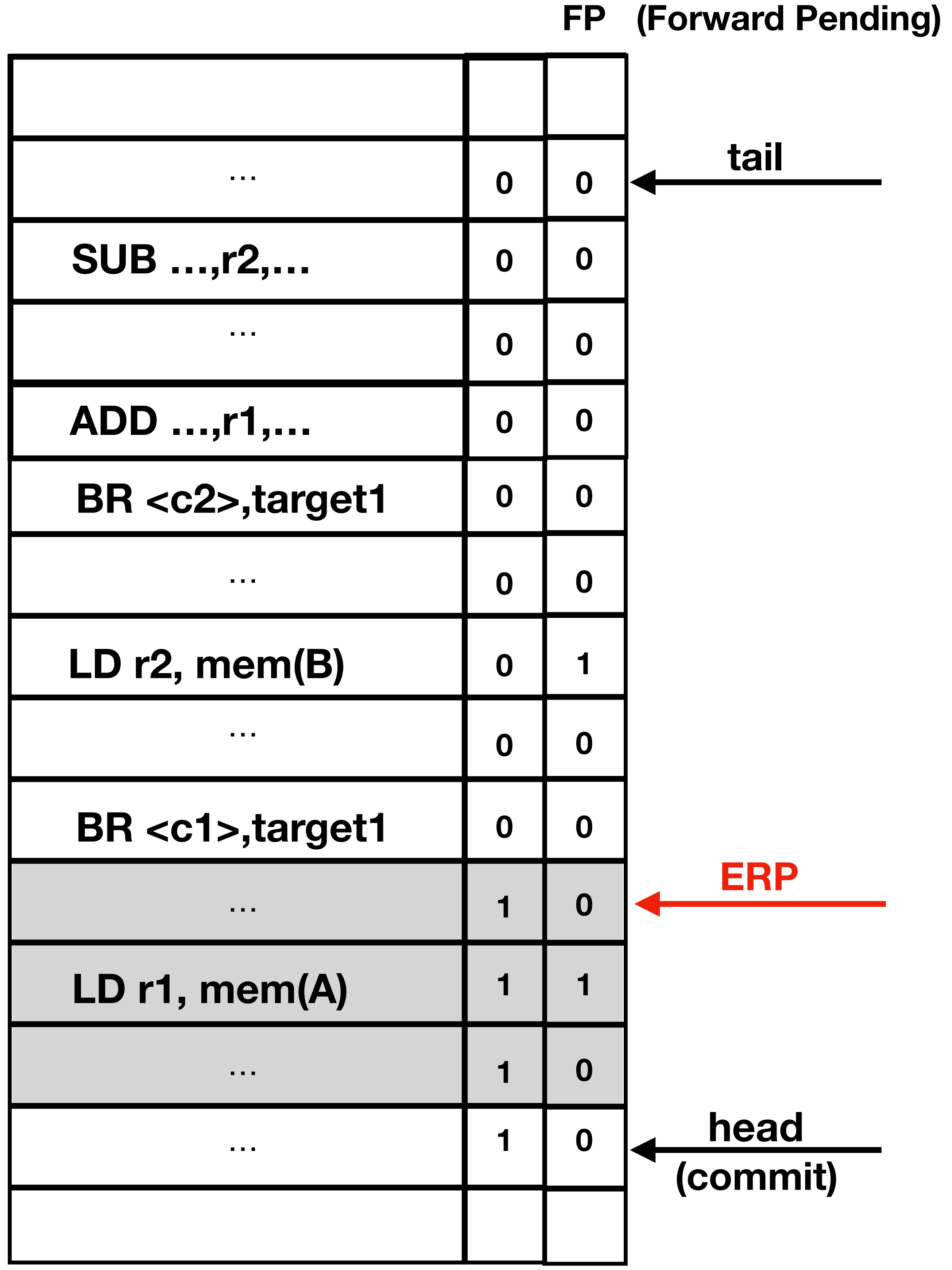


		FP	(Forward Pending)
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed

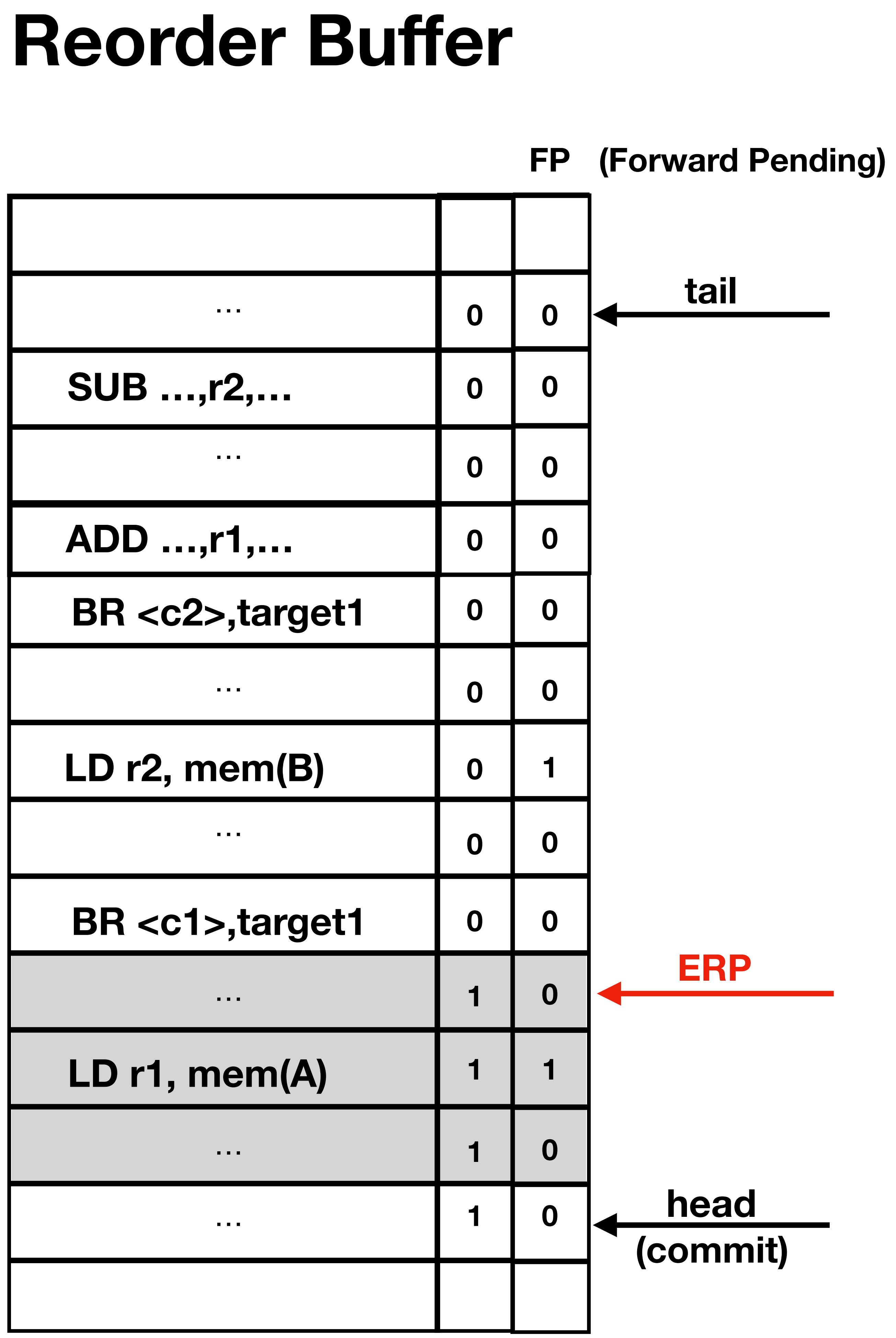
Reorder Buffer



		FP (Forward Pending)	
...	0	0	tail
SUB ...,r2,...	0	0	
...	0	0	
ADD ...,r1,...	0	0	
BR <c2>,target1	0	0	
...	0	0	
LD r2, mem(B)	0	1	
...	0	0	
BR <c1>,target1	0	0	
...	1	0	ERP
LD r1, mem(A)	1	1	
...	1	0	
...	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed
  - No branch mispredictions or memory-access exceptions



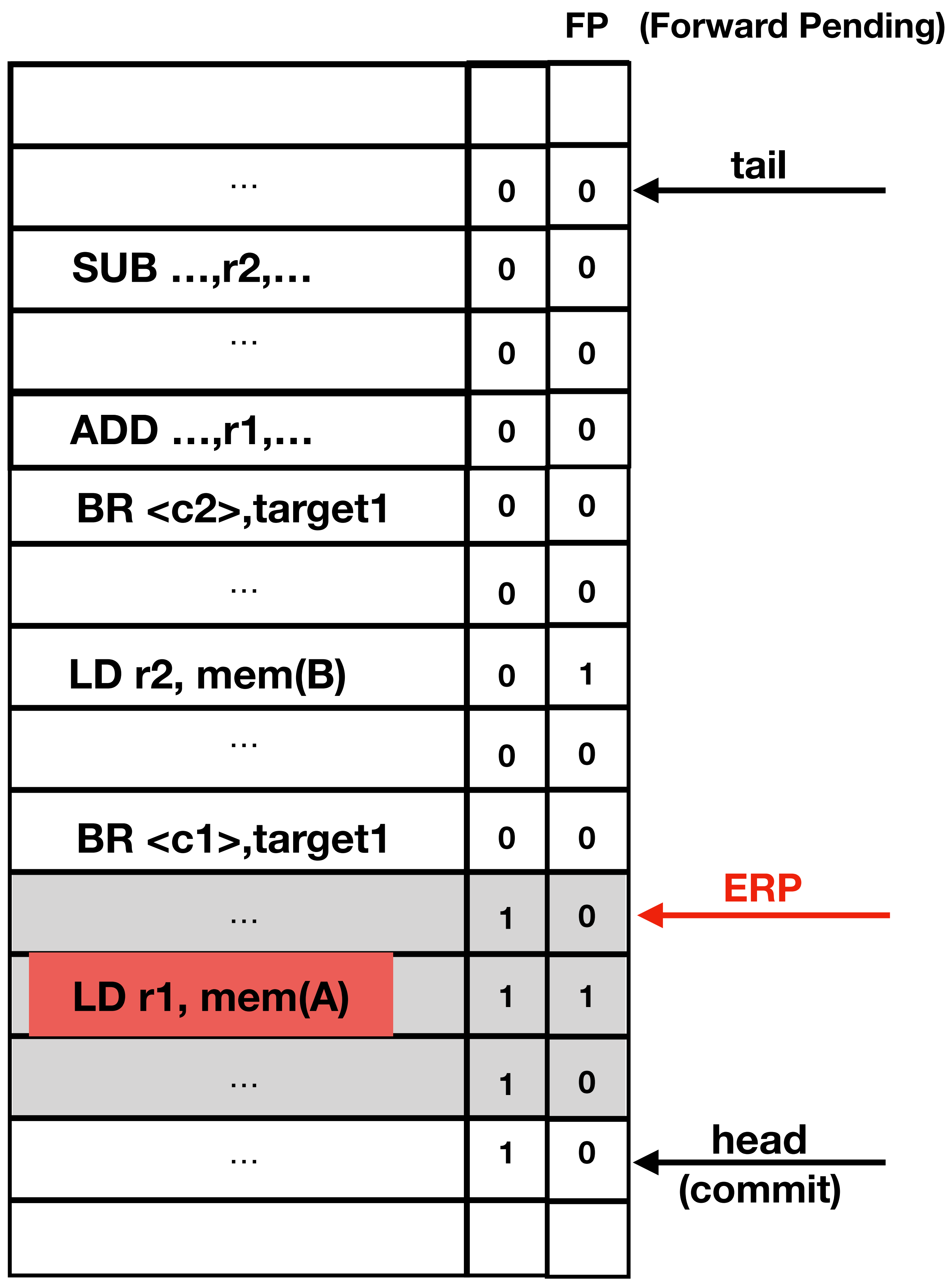
Reorder Buffer

		FP (Forward Pending)	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
    - All older branch instructions have resolved
    - All older loads and stores have had addresses computed
    - No branch mispredictions or memory-access exceptions

Reorder Buffer

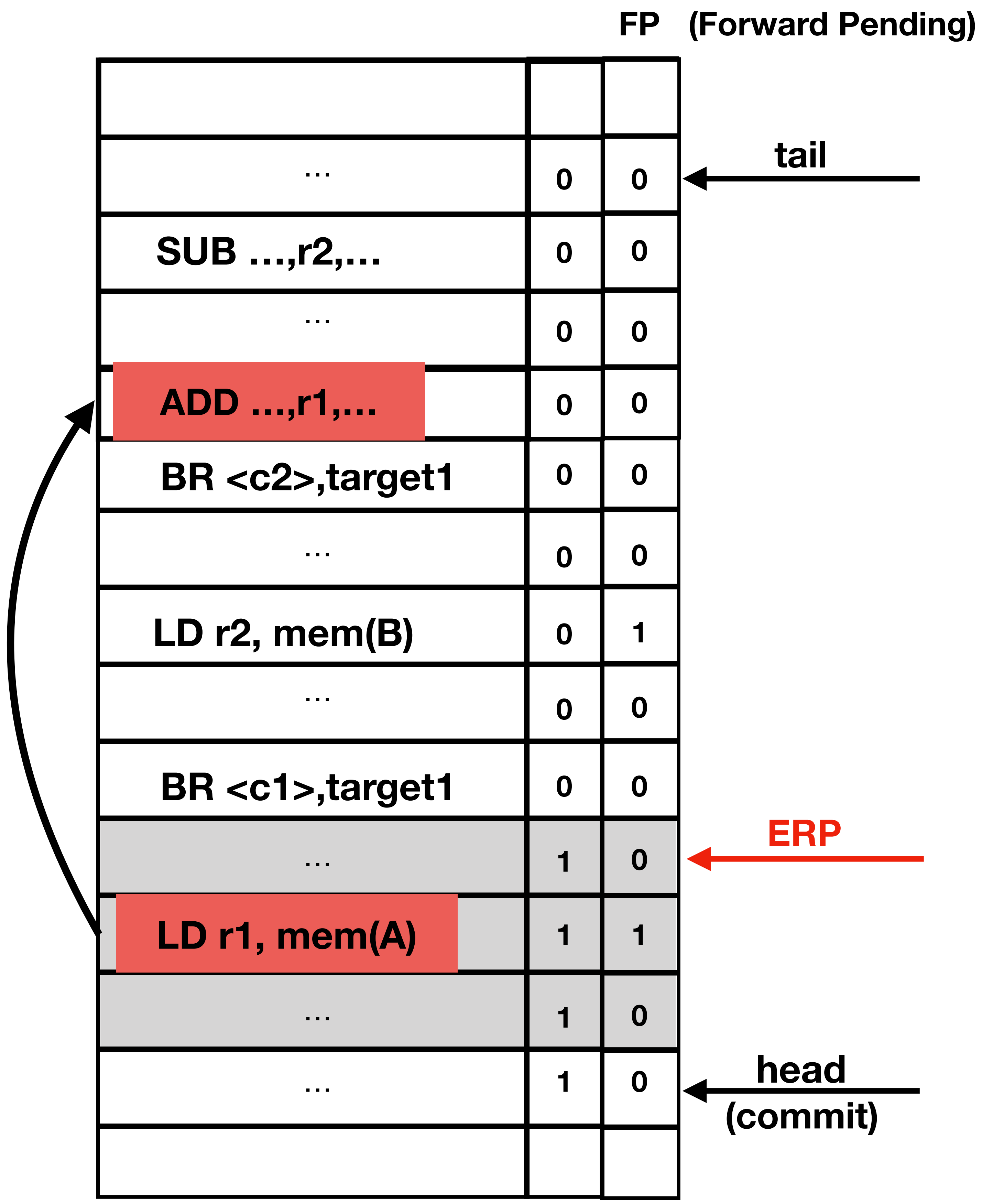


		FP (Forward Pending)
…	0	0
SUB …,r2,…	0	0
…	0	0
ADD …,r1,…	0	0
BR <c2>,target1	0	0
…	0	0
LD r2, mem(B)	0	1
…	0	0
BR <c1>,target1	0	0
…	1	0
LD r1, mem(A)	1	1
…	1	0
…	1	0

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
    - All older branch instructions have resolved
    - All older loads and stores have had addresses computed
    - No branch mispredictions or memory-access exceptions

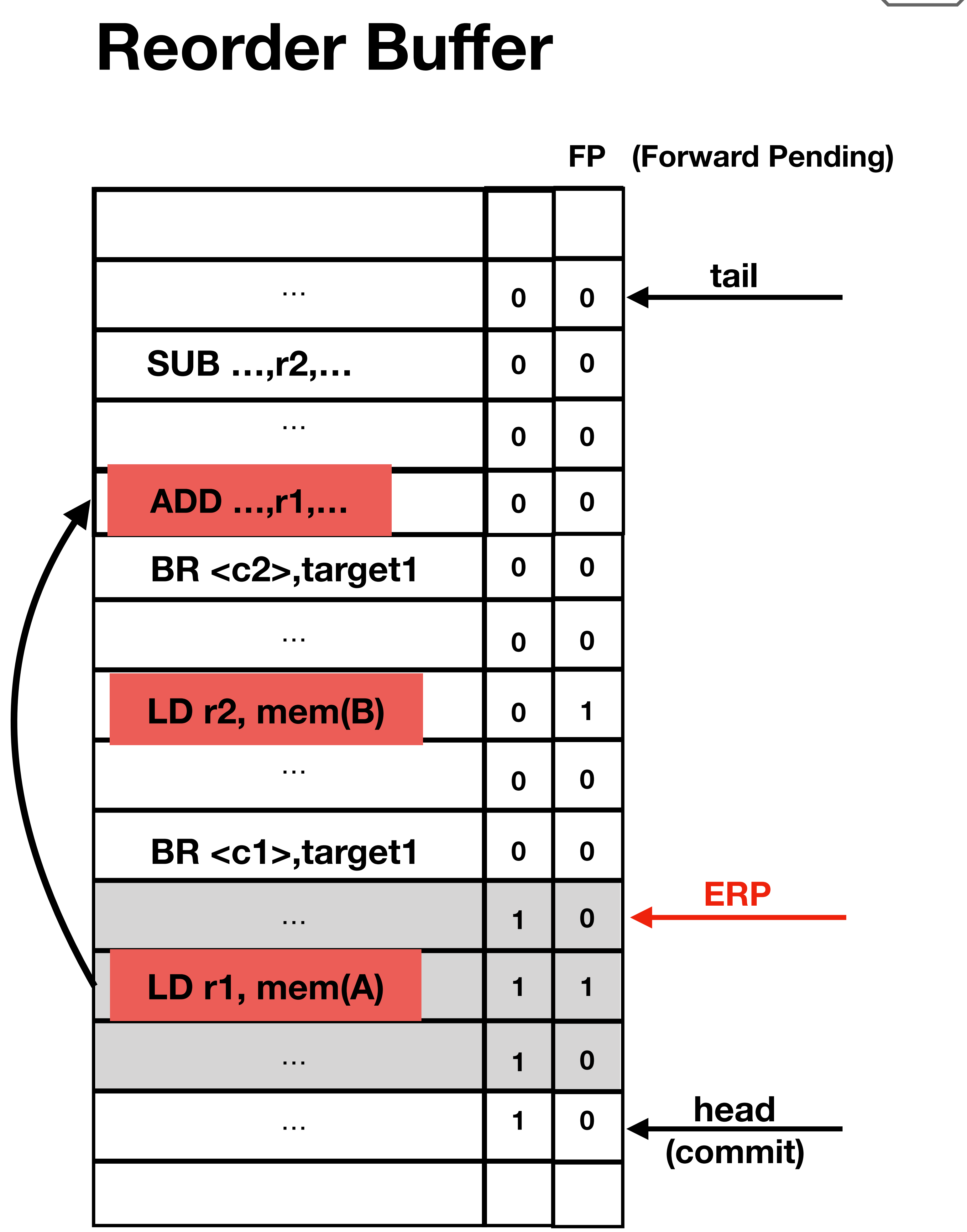
Reorder Buffer



		FP (Forward Pending)
…	0	0
SUB …,r2,…	0	0
…	0	0
ADD …,r1,…	0	0
BR <c2>,target1	0	0
…	0	0
LD r2, mem(B)	0	1
…	0	0
BR <c1>,target1	0	0
…	1	0
LD r1, mem(A)	1	1
…	1	0
…	1	0

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed
  - No branch mispredictions or memory-access exceptions



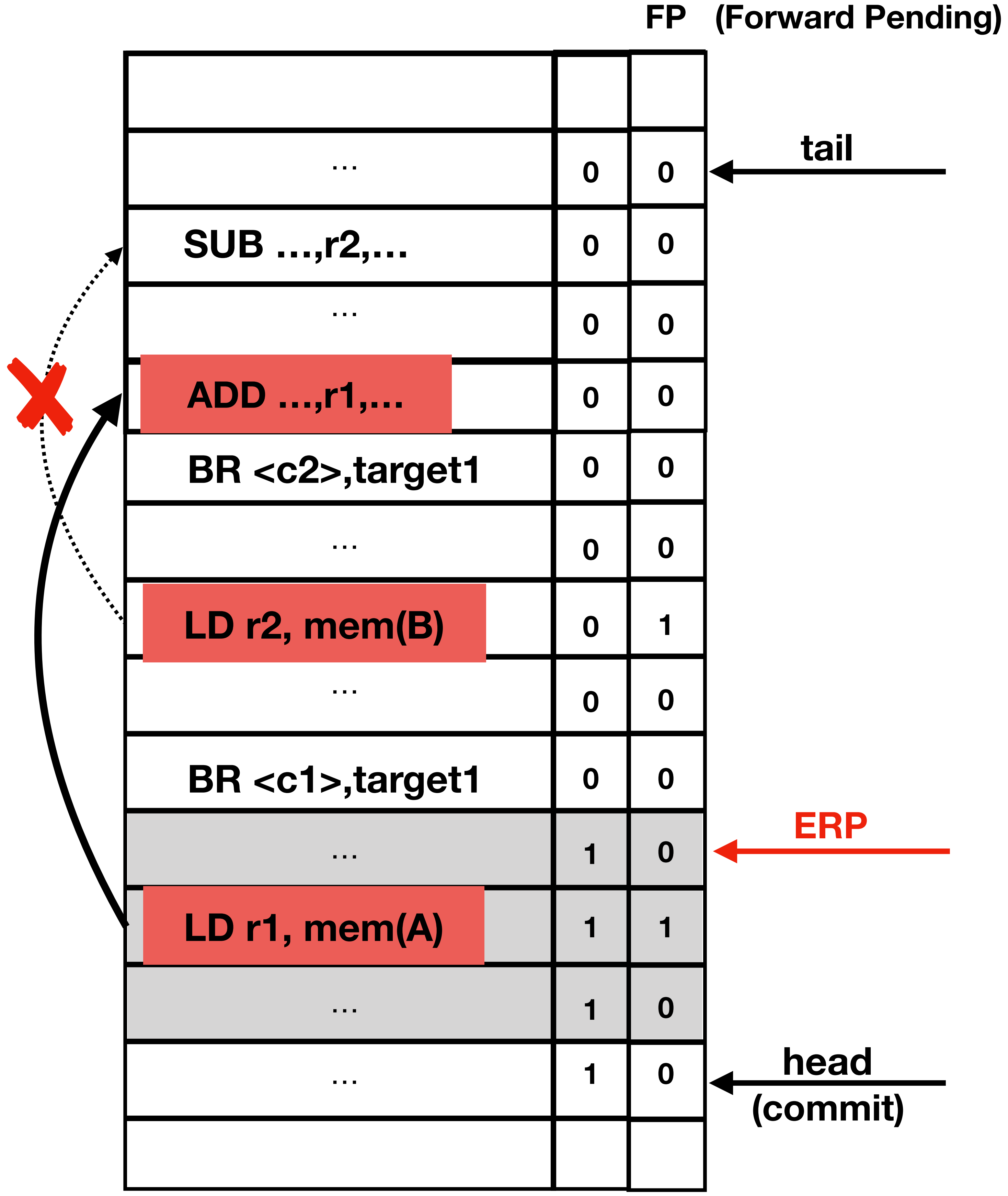
Reorder Buffer

		FP (Forward Pending)	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed
  - No branch mispredictions or memory-access exceptions

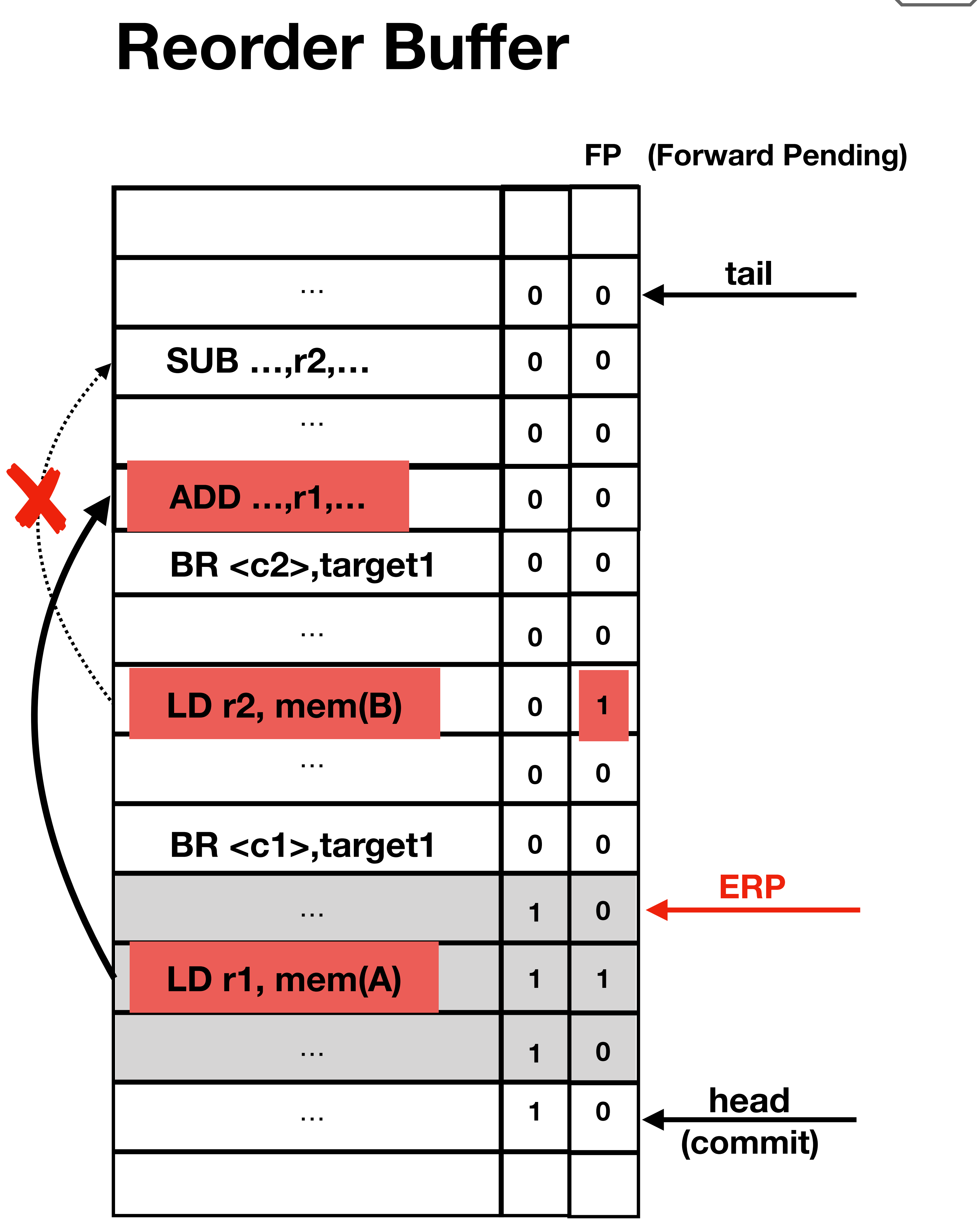
Reorder Buffer



		FP (Forward Pending)	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed
  - No branch mispredictions or memory-access exceptions



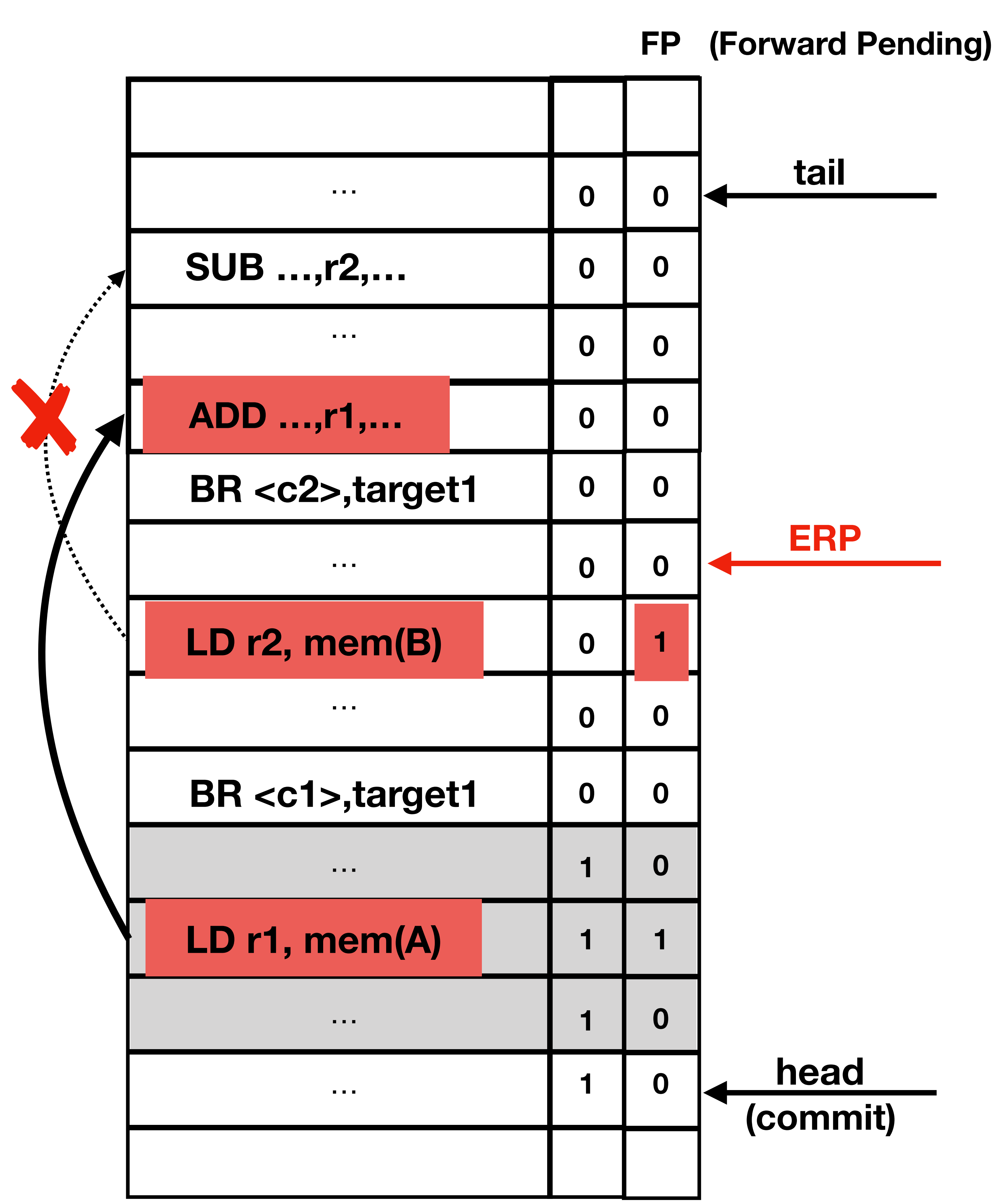
Reorder Buffer

		FP (Forward Pending)	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	ERP
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed
  - No branch mispredictions or memory-access exceptions

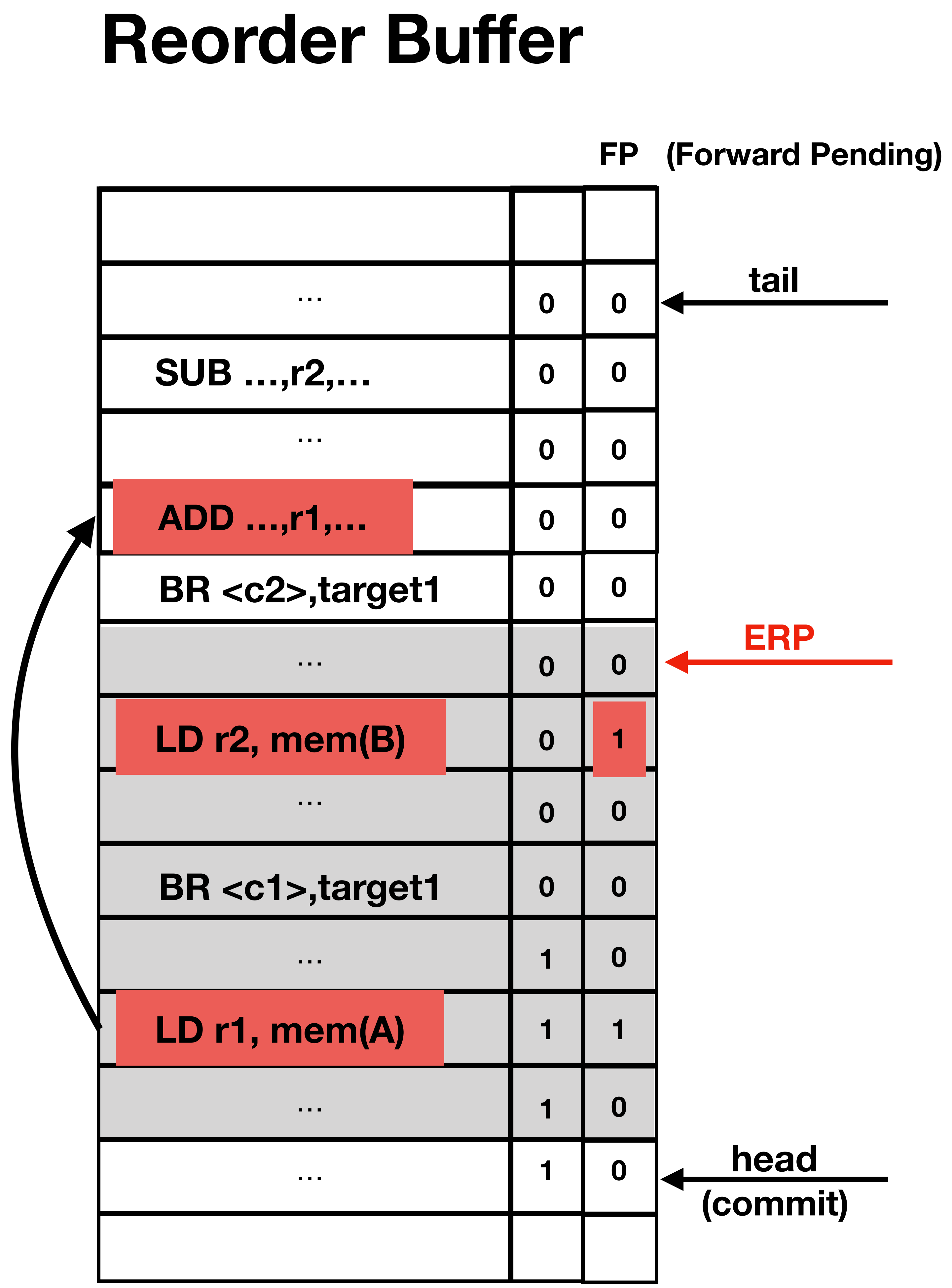
Reorder Buffer



		FP (Forward Pending)	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	ERP
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed
  - No branch mispredictions or memory-access exceptions



Reorder Buffer

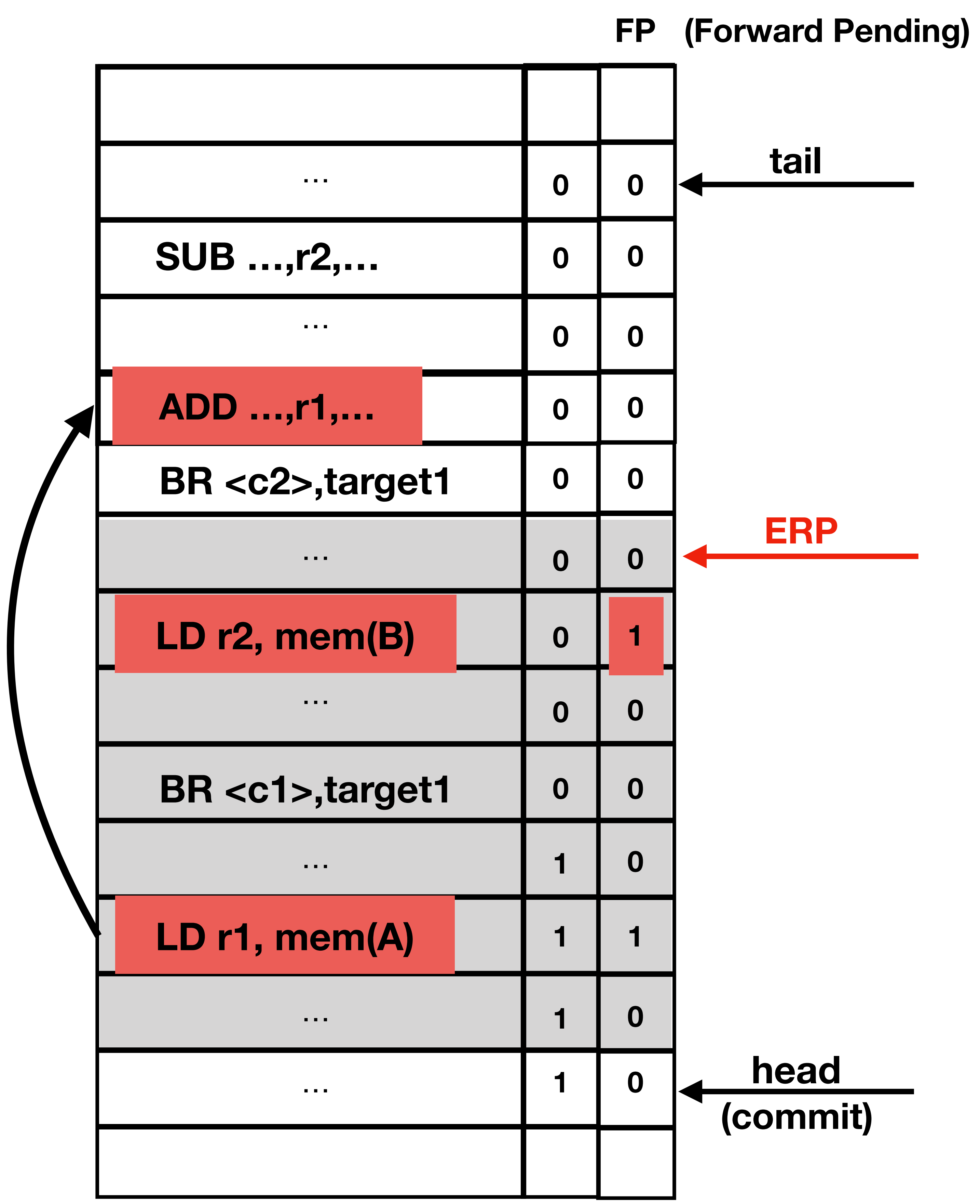
FP (Forward Pending)

		FP	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	ERP
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed
  - No branch mispredictions or memory-access exceptions
- Loads behind ERP can be considered safe and allowed to forward data

Reorder Buffer

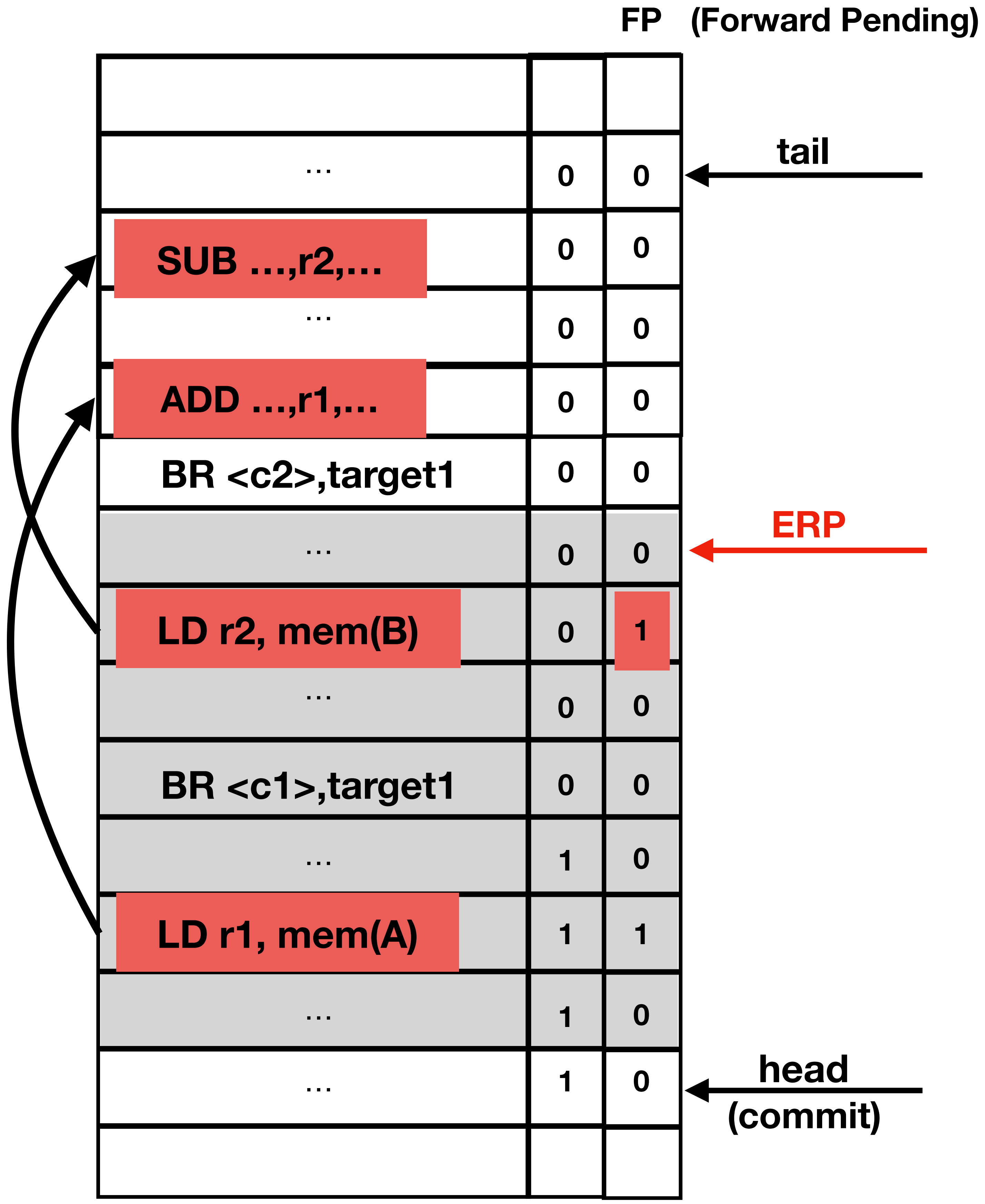


		FP	(Forward Pending)
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	ERP
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed
  - No branch mispredictions or memory-access exceptions
- Loads behind ERP can be considered safe and allowed to forward data

Reorder Buffer

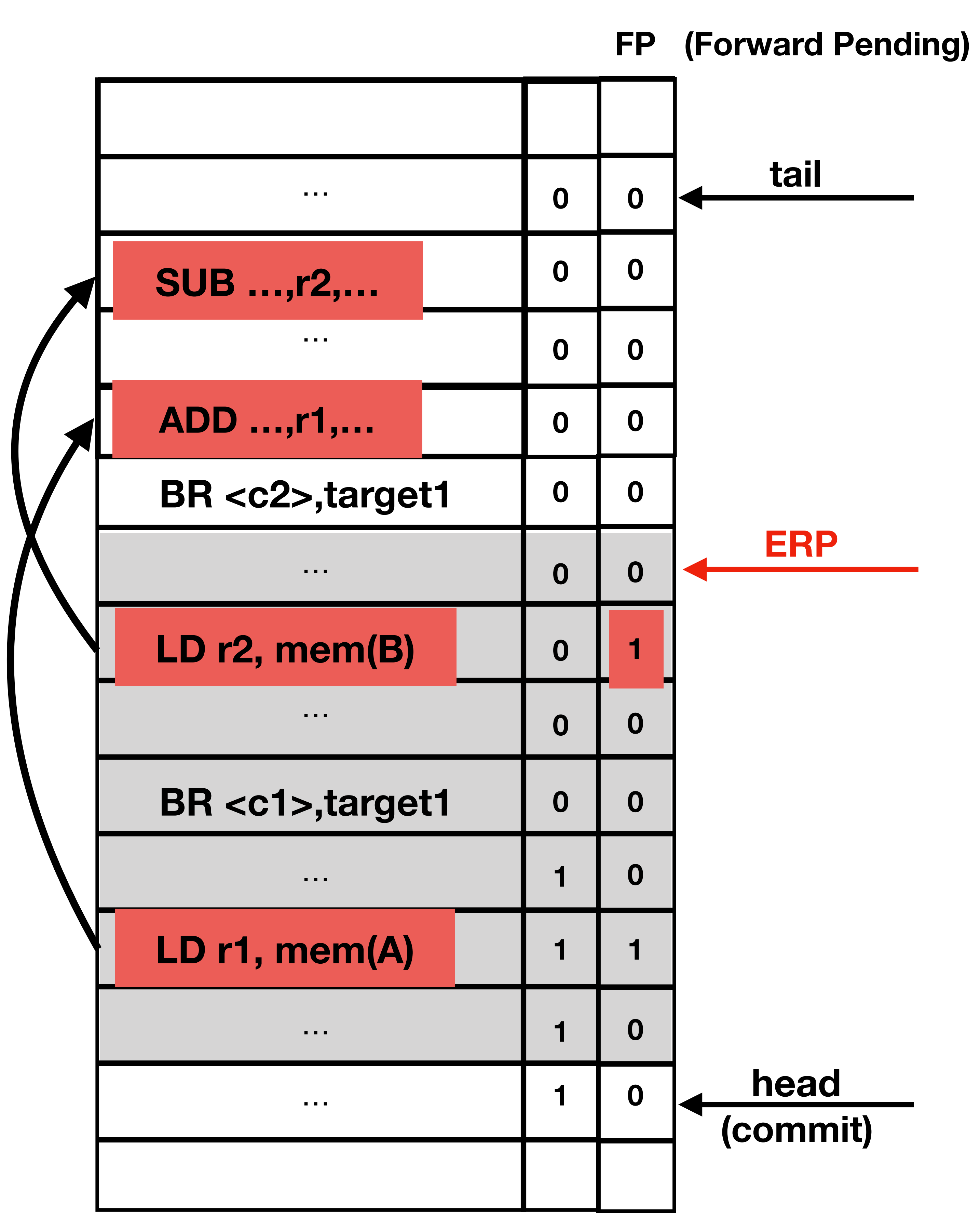


		FP (Forward Pending)	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	ERP
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP

- Goal: Relax constraints on allowable forwarding
- Observation: Most loads are safe earlier than retirement
- Define Early Resolution Point (ERP), instruction in the ROB where:
  - All older branch instructions have resolved
  - All older loads and stores have had addresses computed
  - No branch mispredictions or memory-access exceptions
- Loads behind ERP can be considered safe and allowed to forward data
- Much lower performance impact, equivalent security

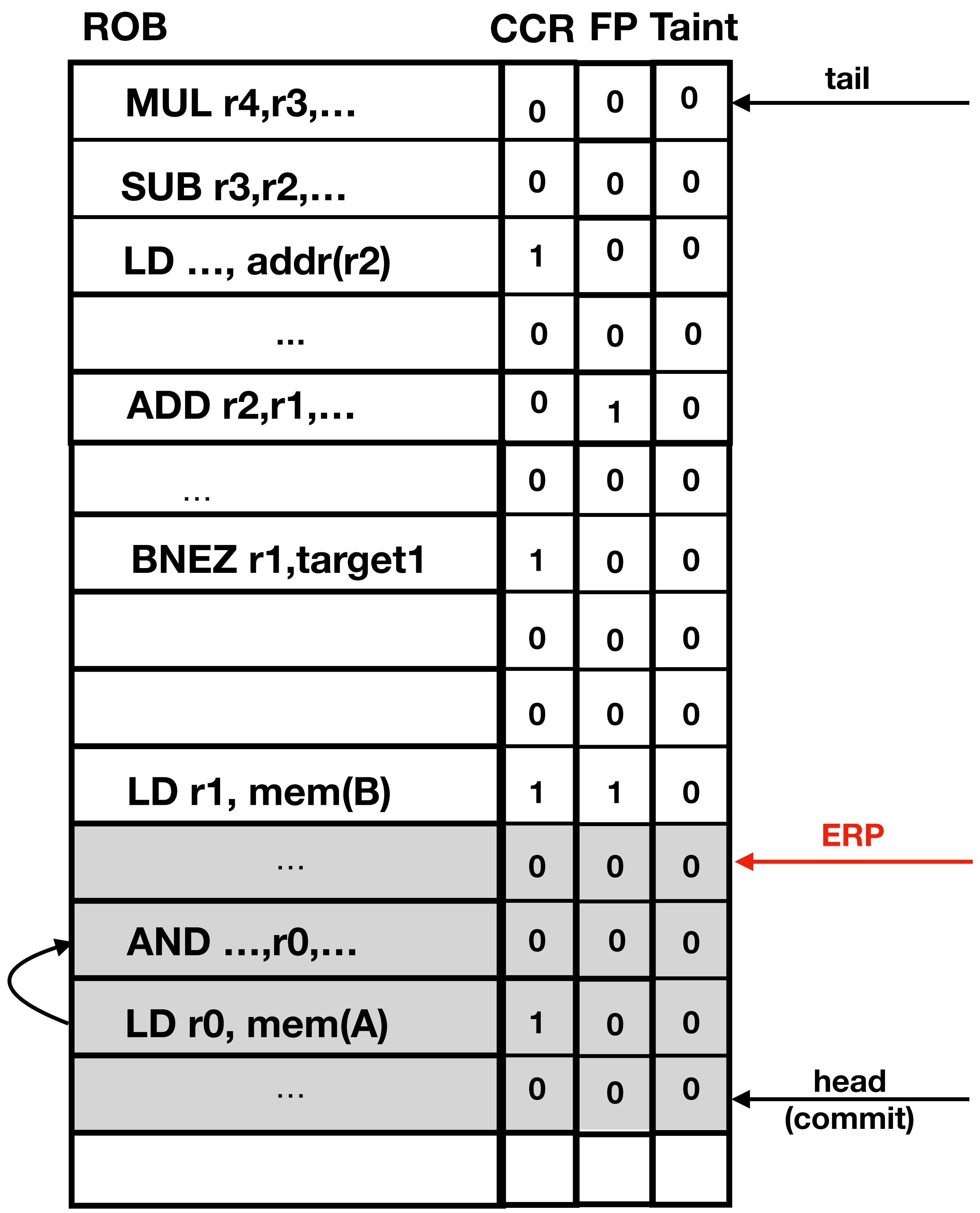
Reorder Buffer



		FP (Forward Pending)	
…	0	0	tail
SUB …,r2,…	0	0	
…	0	0	
ADD …,r1,…	0	0	
BR <c2>,target1	0	0	
…	0	0	ERP
LD r2, mem(B)	0	1	
…	0	0	
BR <c1>,target1	0	0	
…	1	0	
LD r1, mem(A)	1	1	
…	1	0	
…	1	0	head (commit)

SpecShield ERP+

Reorder Buffer

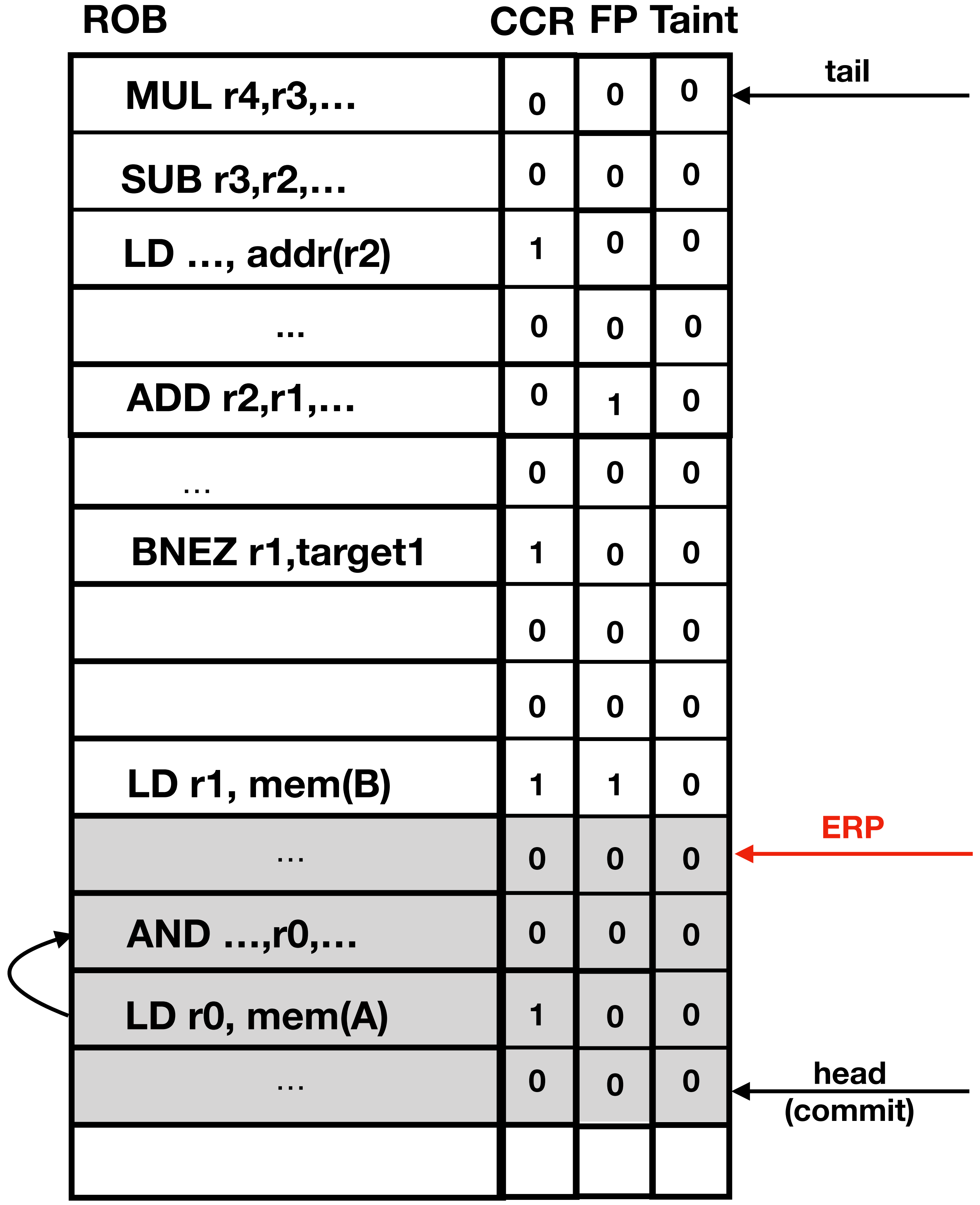


ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	0
SUB r3,r2,...	0	0	0
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	0
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

SpecShield ERP+

- A covert channel-specific optimization

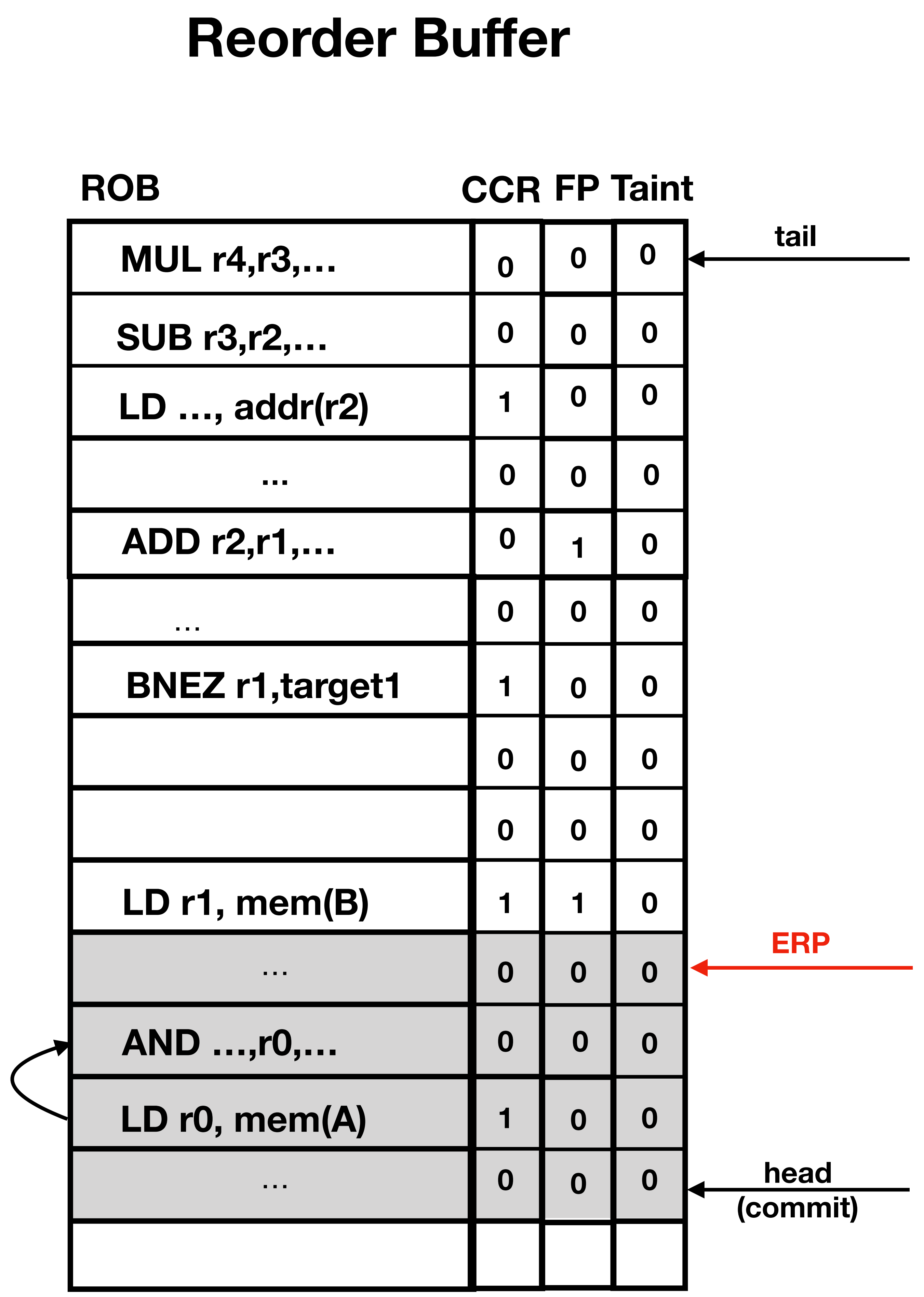
Reorder Buffer



ROB	CCR	FP	Taint
MUL r4,r3,…	0	0	0
SUB r3,r2,…	0	0	0
LD …, addr(r2)	1	0	0
…	0	0	0
ADD r2,r1,…	0	1	0
…	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
…	0	0	0
AND …,r0,…	0	0	0
LD r0, mem(A)	1	0	0
…	0	0	0

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative



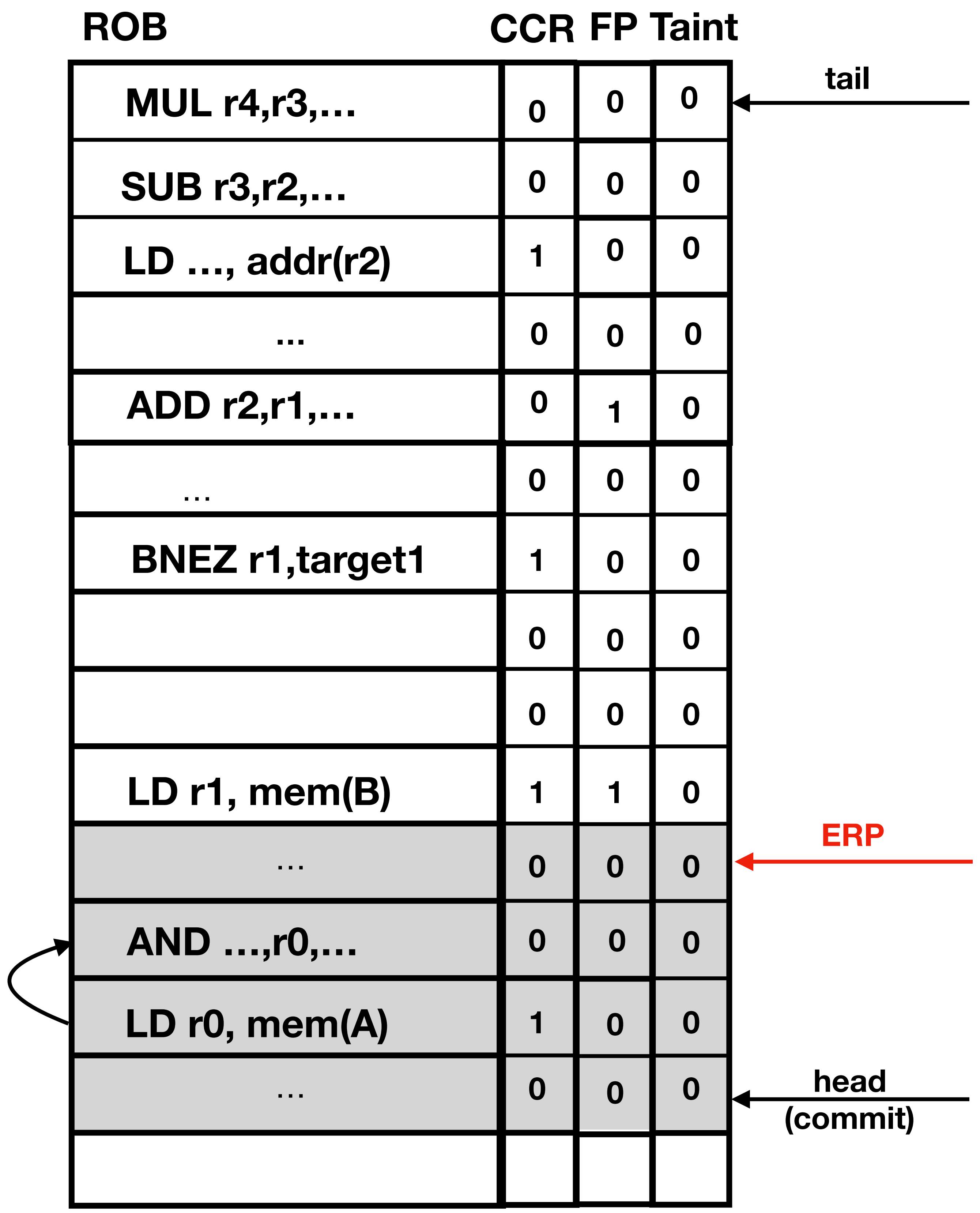
Reorder Buffer

ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	0
SUB r3,r2,...	0	0	0
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	0
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
    - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)

Reorder Buffer

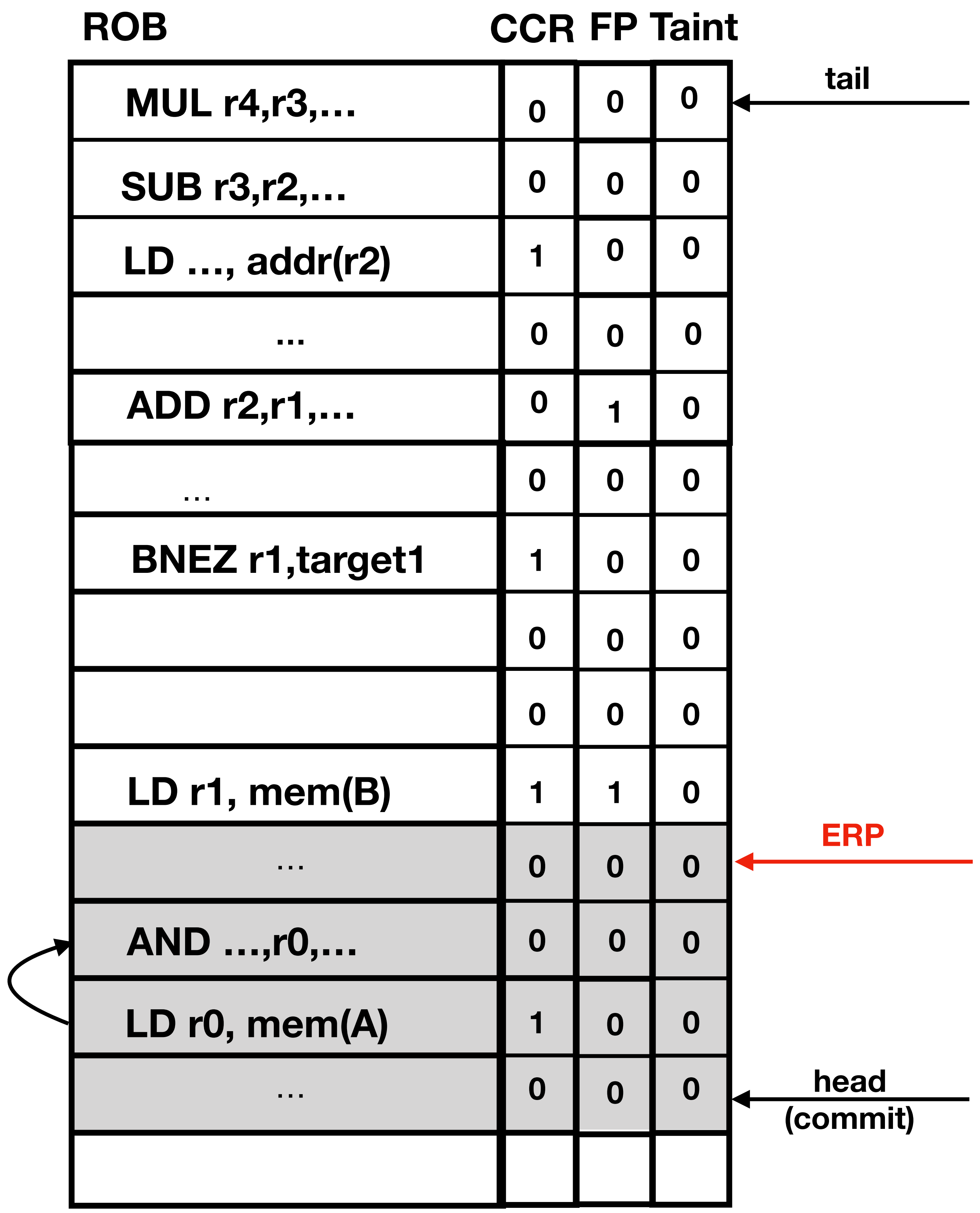


ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	0
SUB r3,r2,...	0	0	0
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	0
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk

Reorder Buffer

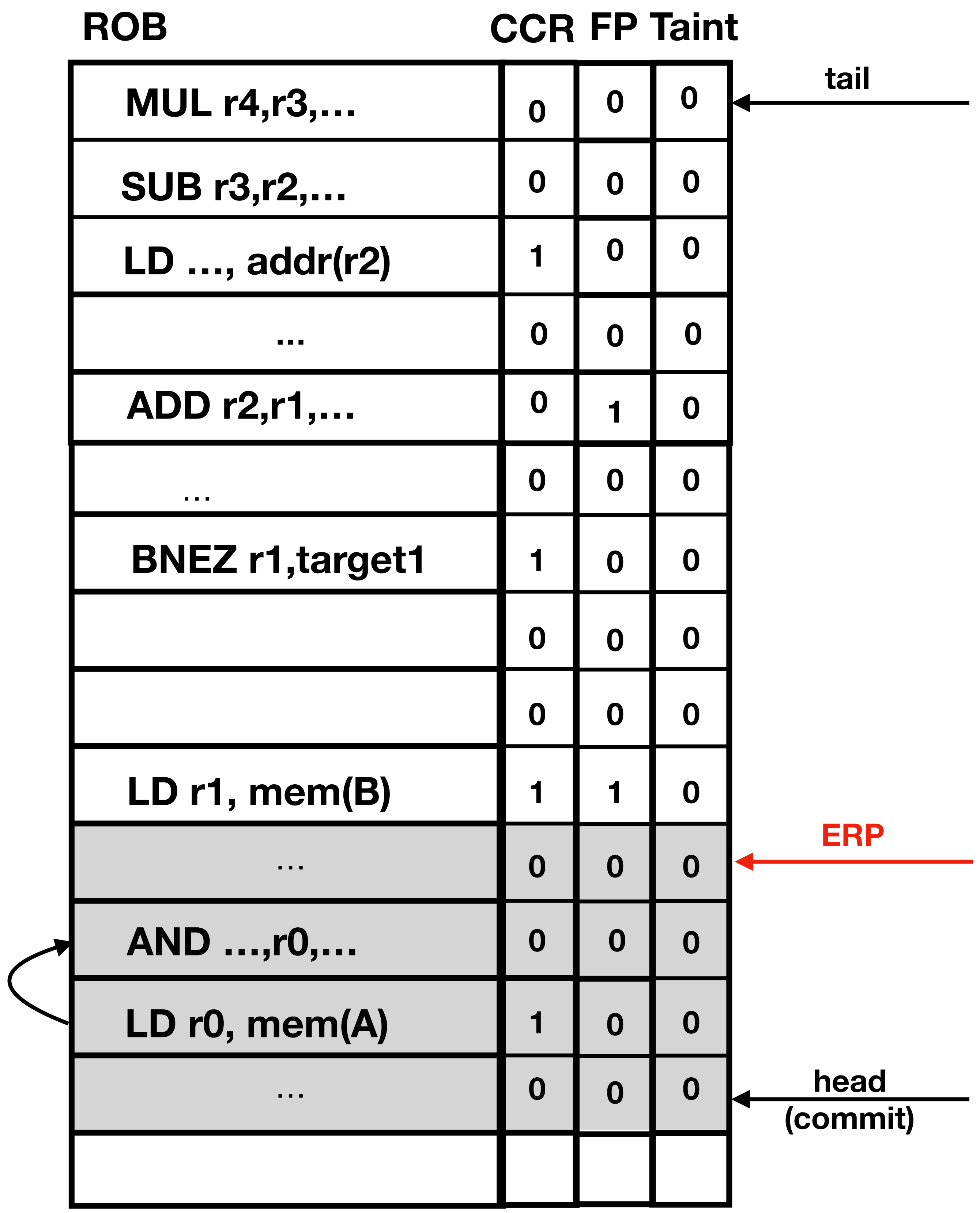


ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	0
SUB r3,r2,...	0	0	0
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	0
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk

Reorder Buffer



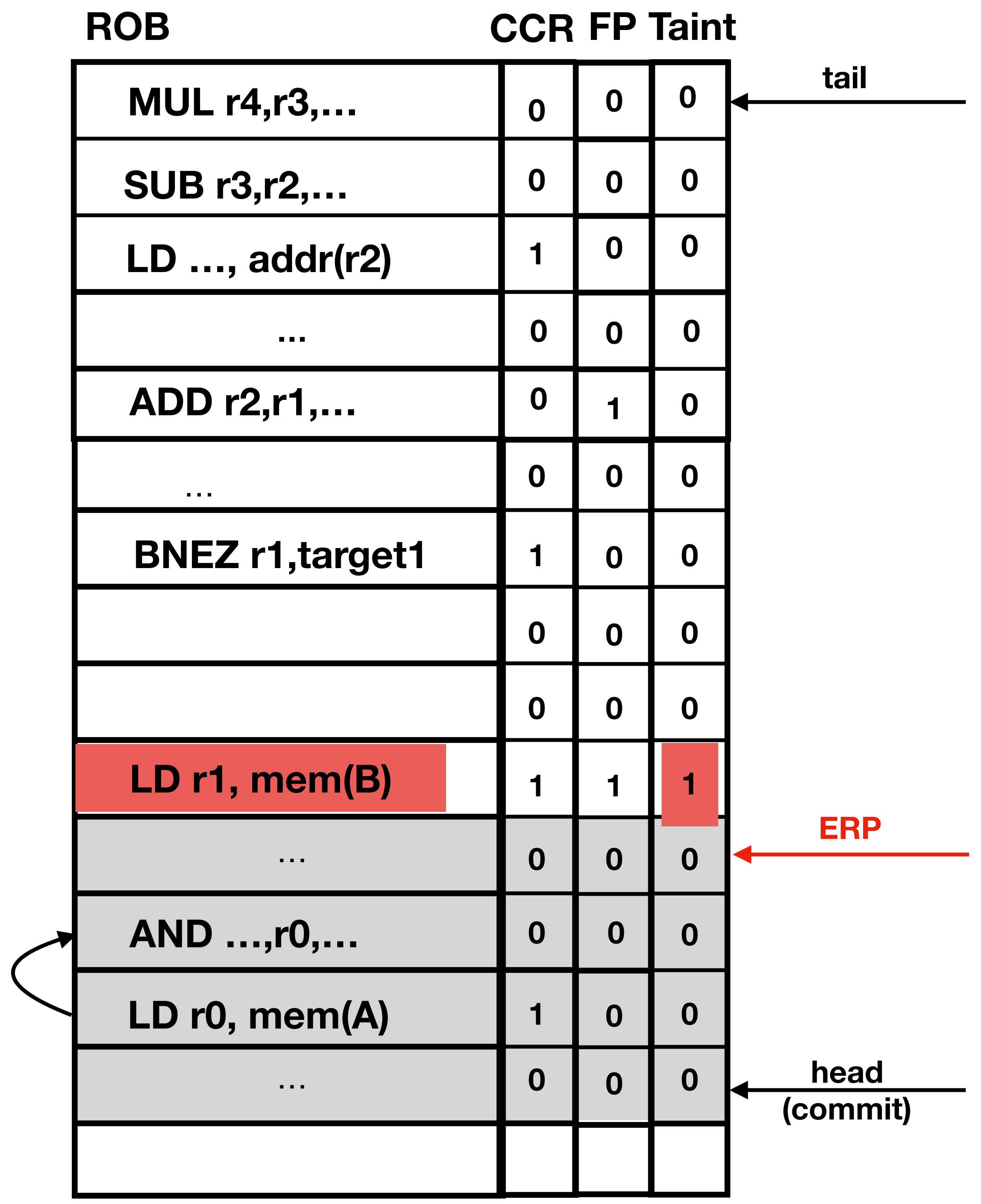
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	0
SUB r3,r2,...	0	0	0
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	0
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:

Reorder Buffer



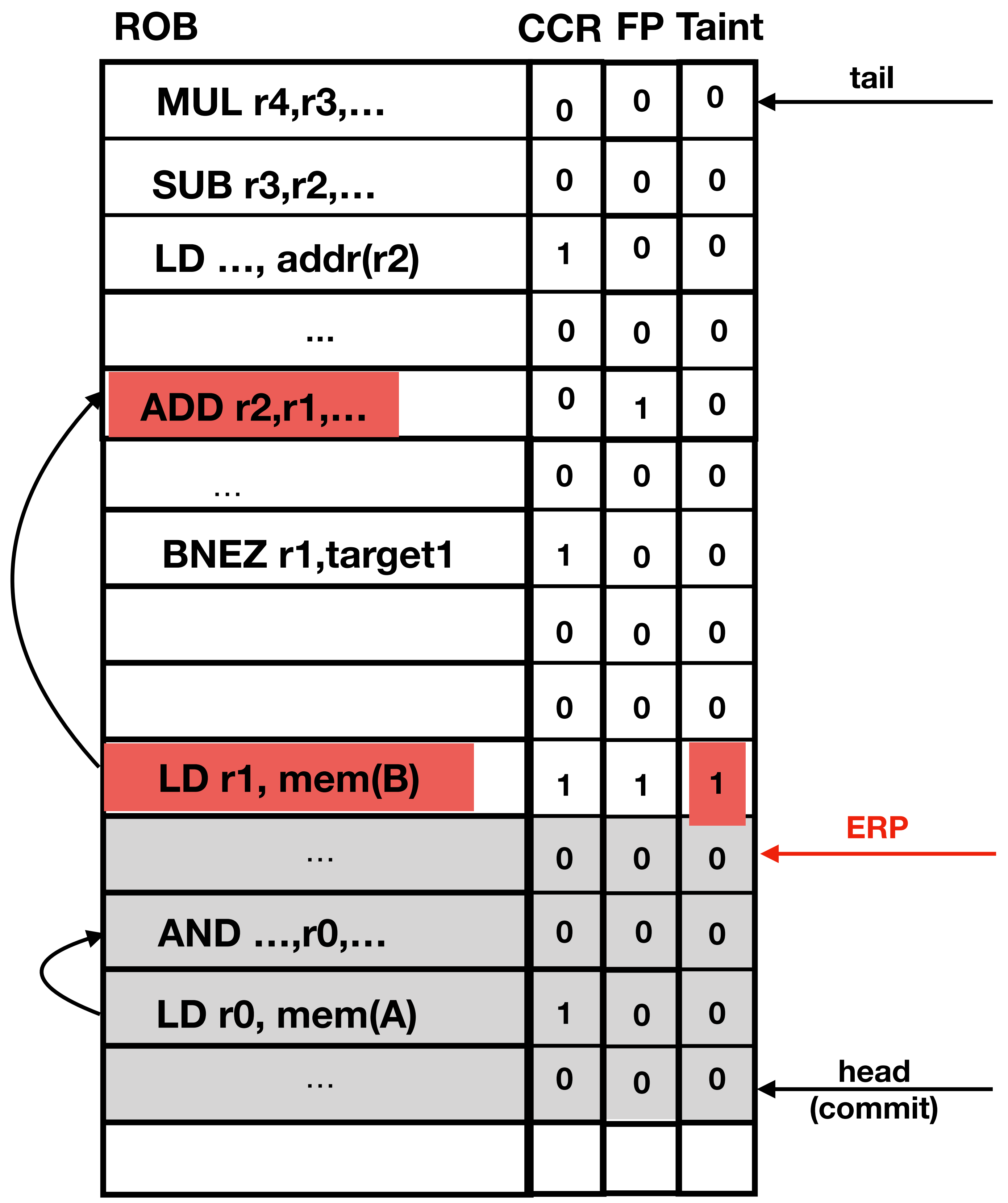
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	0
SUB r3,r2,...	0	0	0
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	0
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
    - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
    - Immediately to low leakage risk instructions

Reorder Buffer



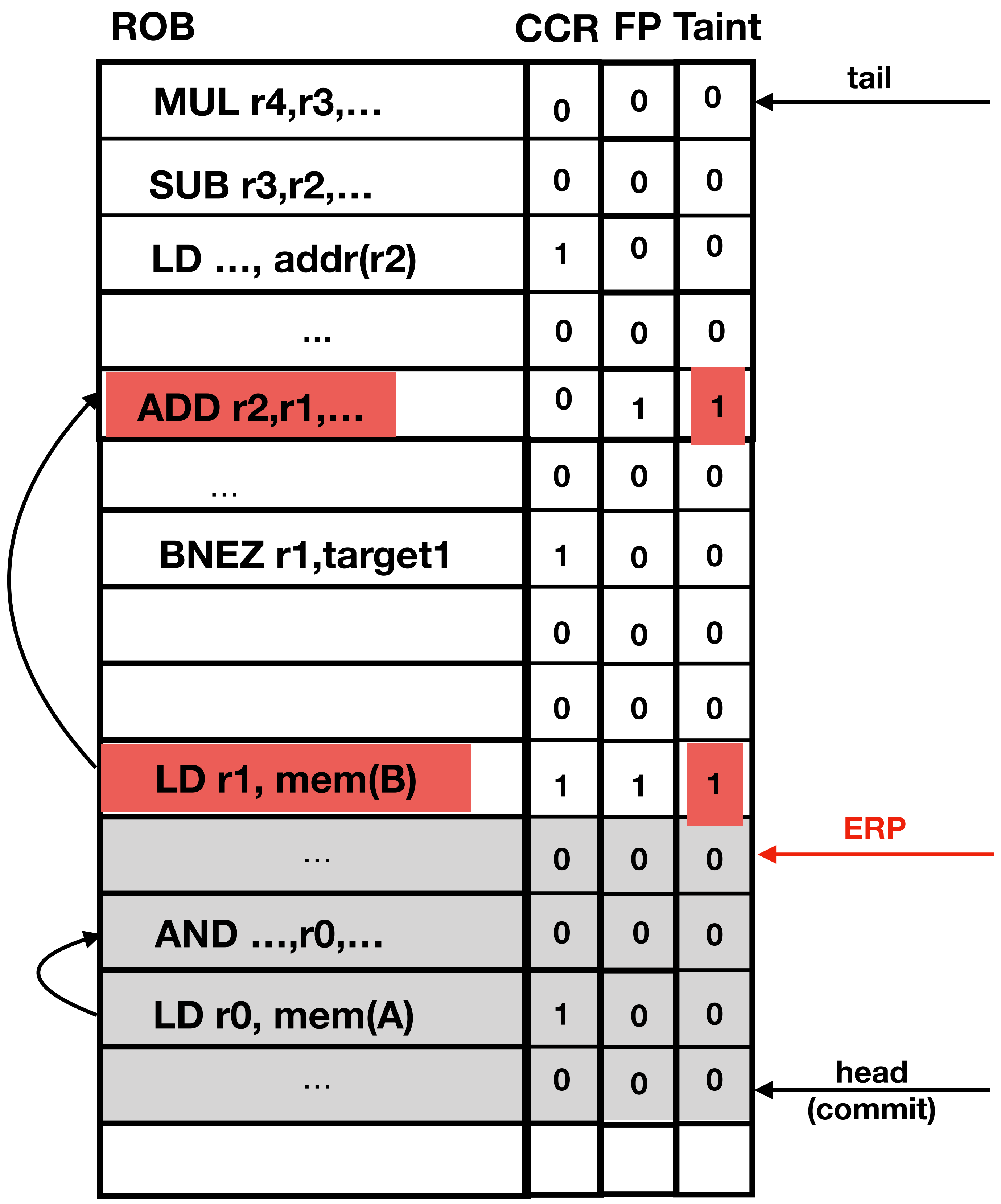
ROB	CCR	FP	Taint	
MUL r4,r3,...	0	0	0	tail
SUB r3,r2,...	0	0	0	
LD ..., addr(r2)	1	0	0	
...	0	0	0	
ADD r2,r1,...	0	1	0	
...	0	0	0	
BNEZ r1,target1	1	0	0	
	0	0	0	
	0	0	0	
LD r1, mem(B)	1	1	1	
...	0	0	0	ERP
AND ...,r0,...	0	0	0	
LD r0, mem(A)	1	0	0	
...	0	0	0	head (commit)

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data

Reorder Buffer



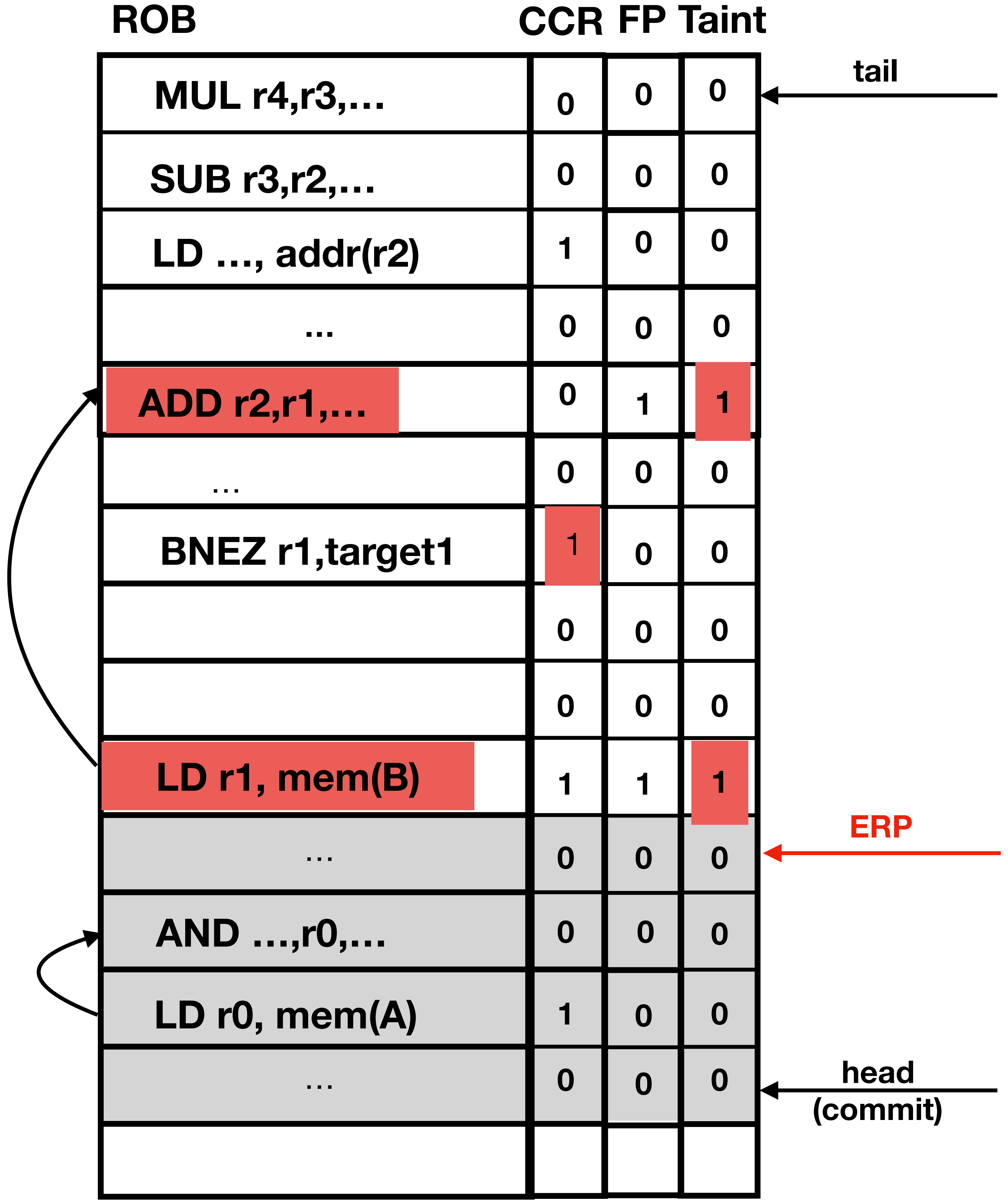
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	0
SUB r3,r2,...	0	0	0
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data

Reorder Buffer



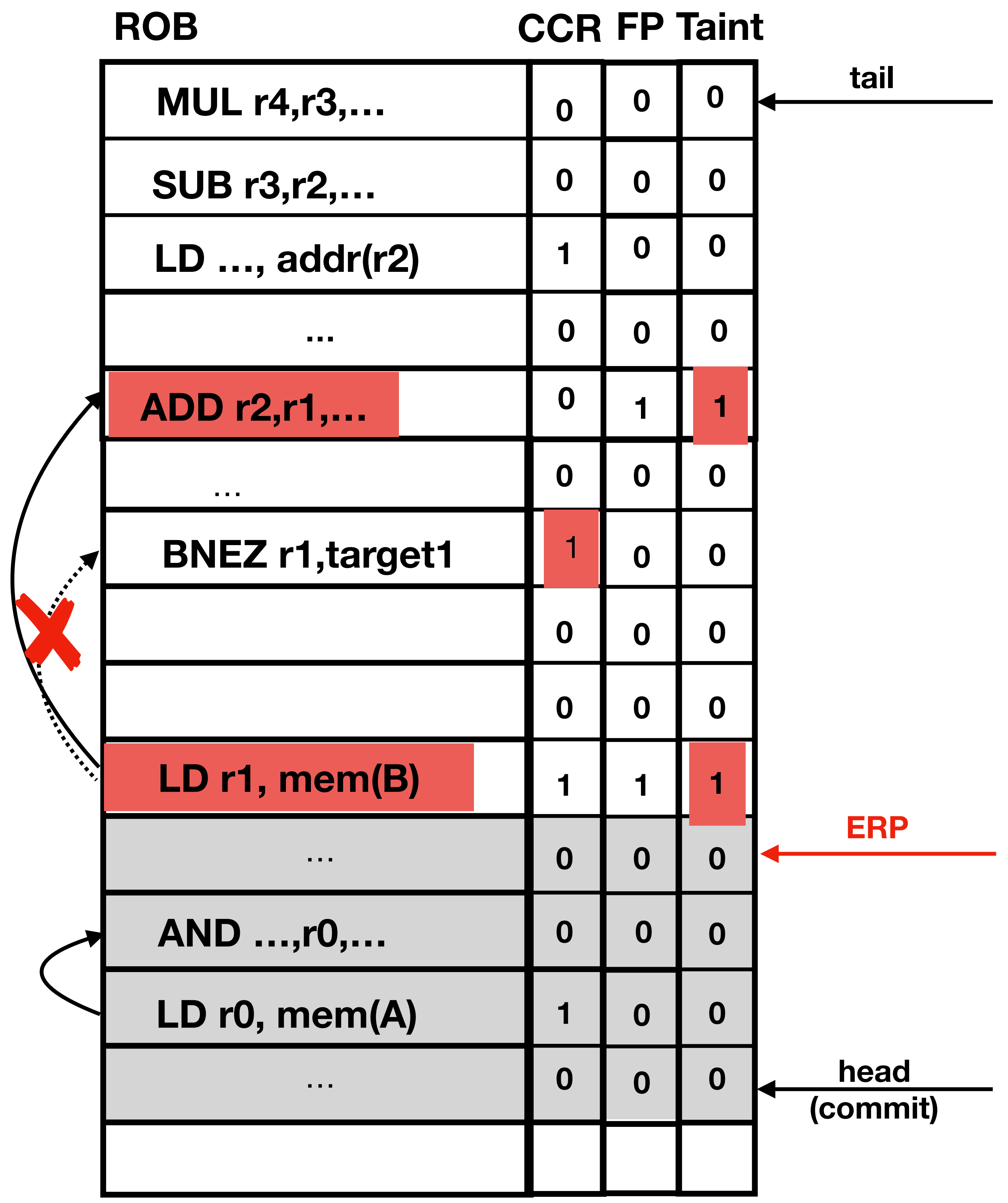
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	0
SUB r3,r2,...	0	0	0
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data

Reorder Buffer



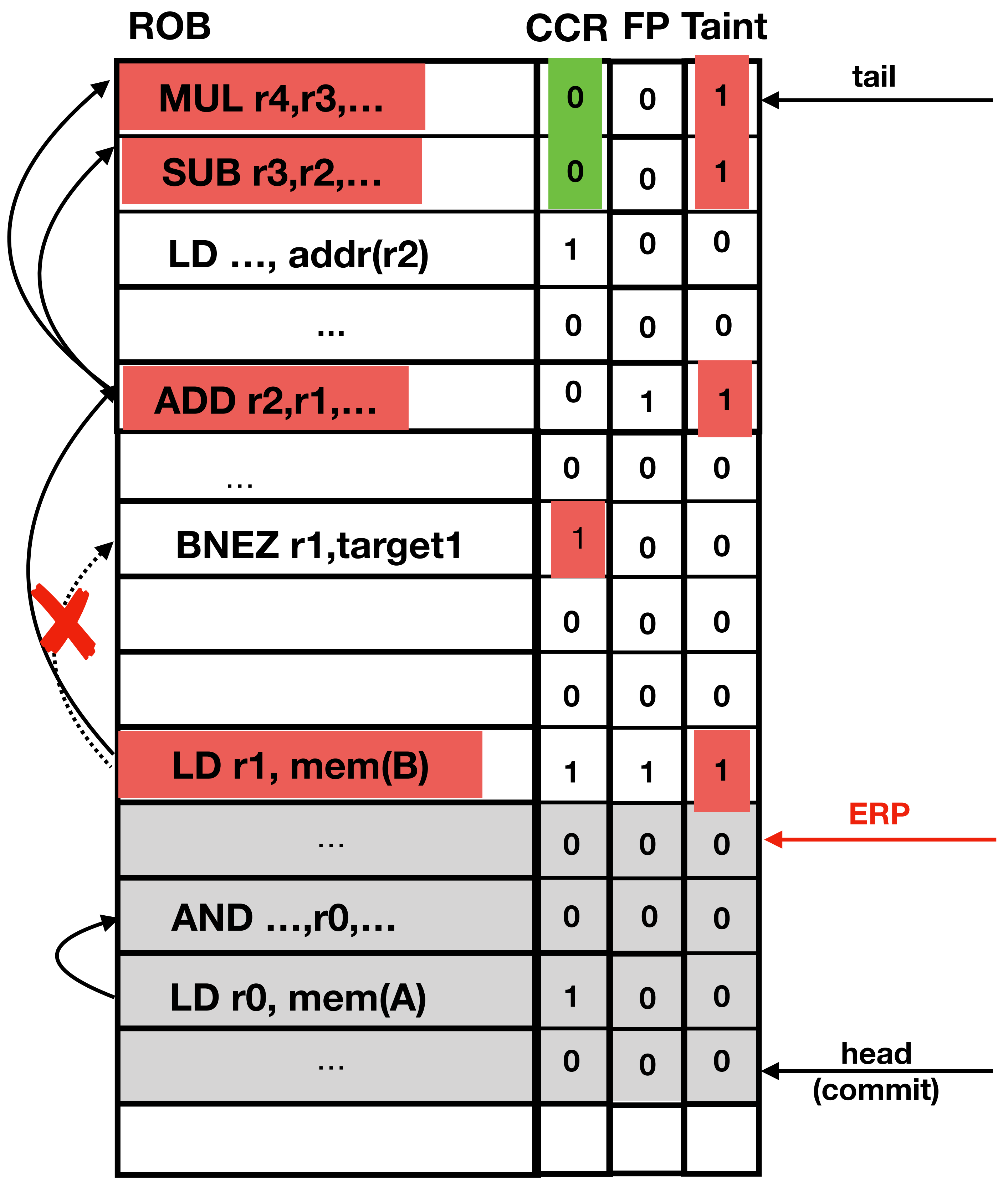
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	0
SUB r3,r2,...	0	0	0
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction

Reorder Buffer



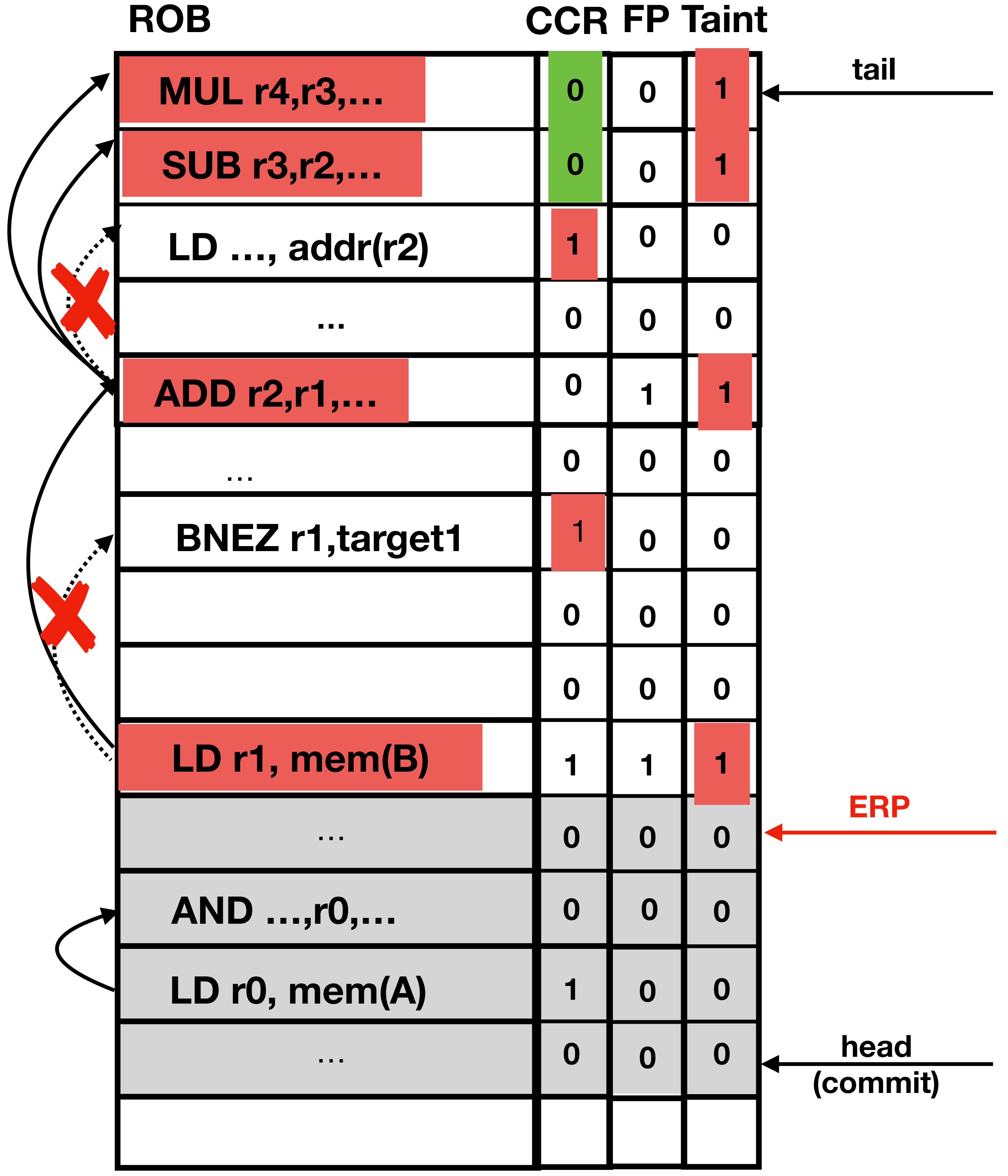
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction

Reorder Buffer



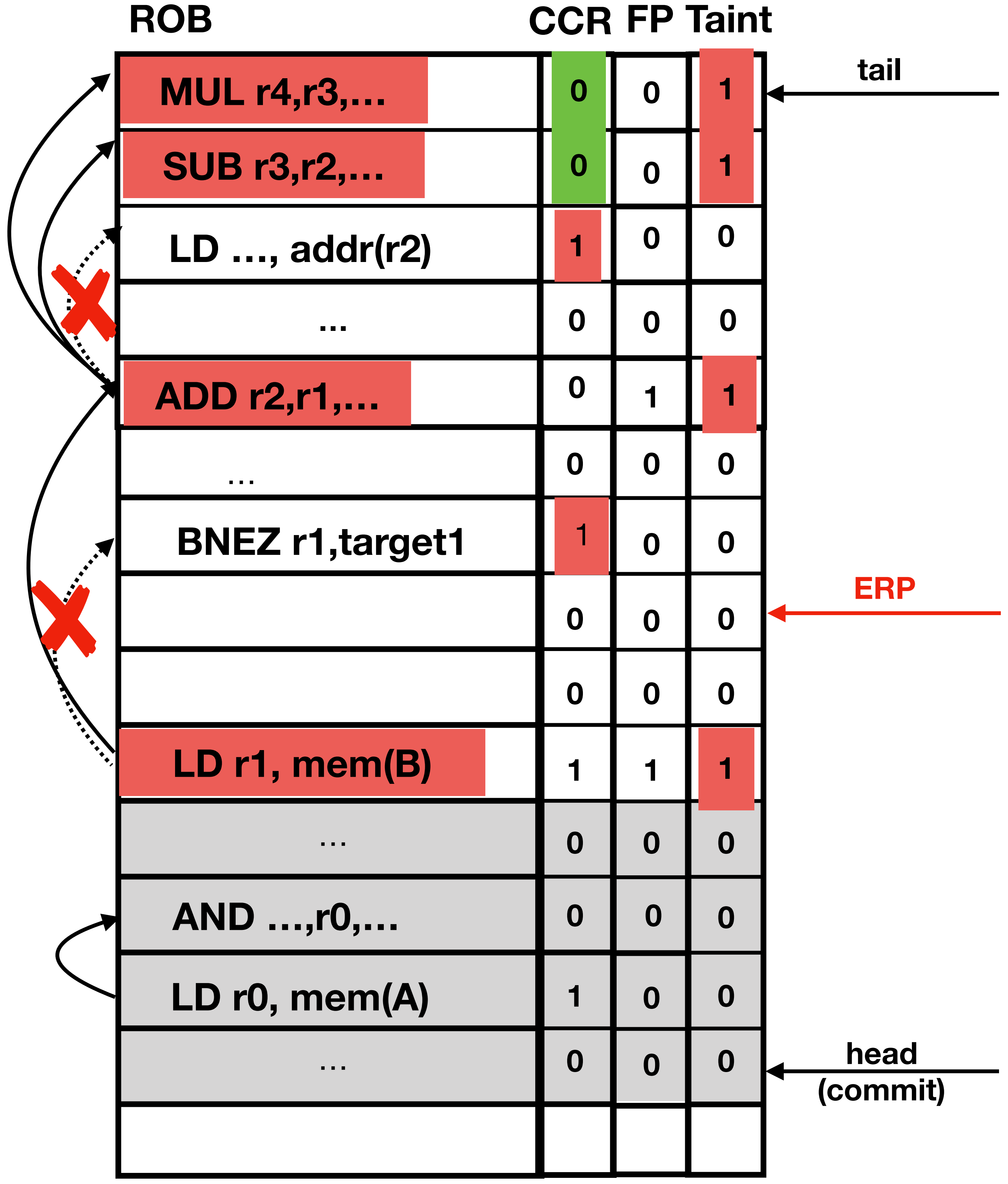
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction

Reorder Buffer



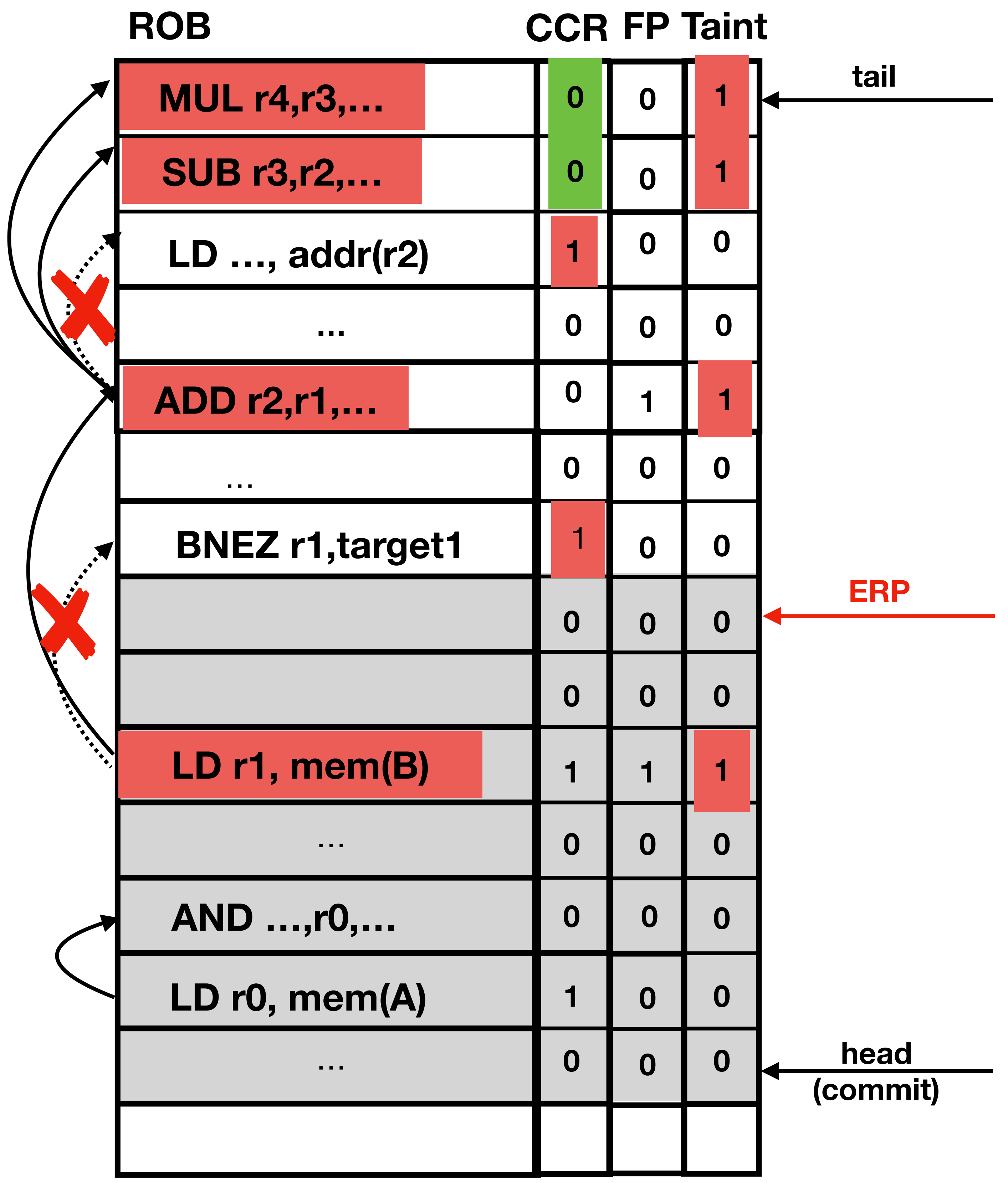
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction

Reorder Buffer



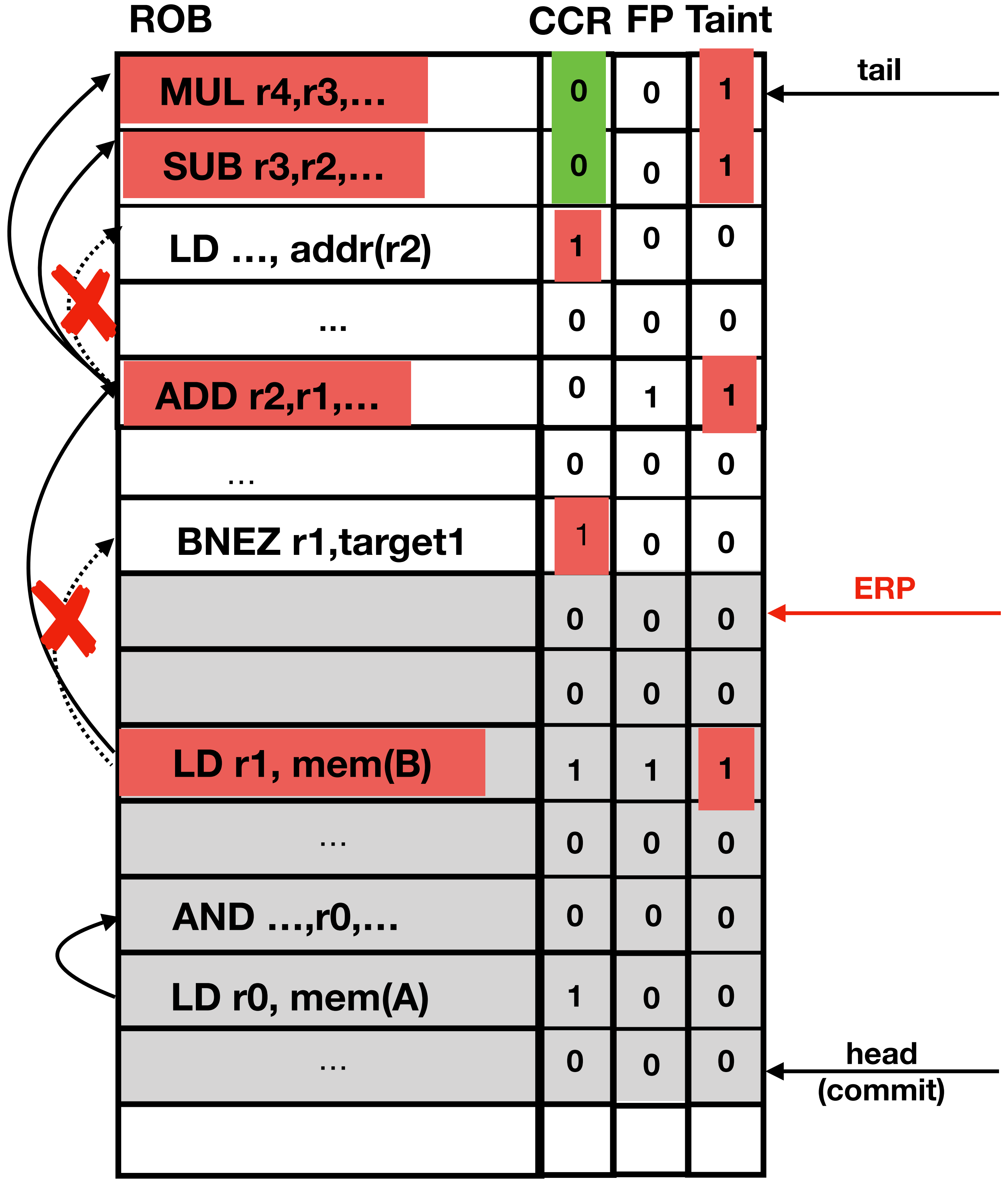
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction
  - At ERP for high leakage risk instructions, from tainted instructions

Reorder Buffer



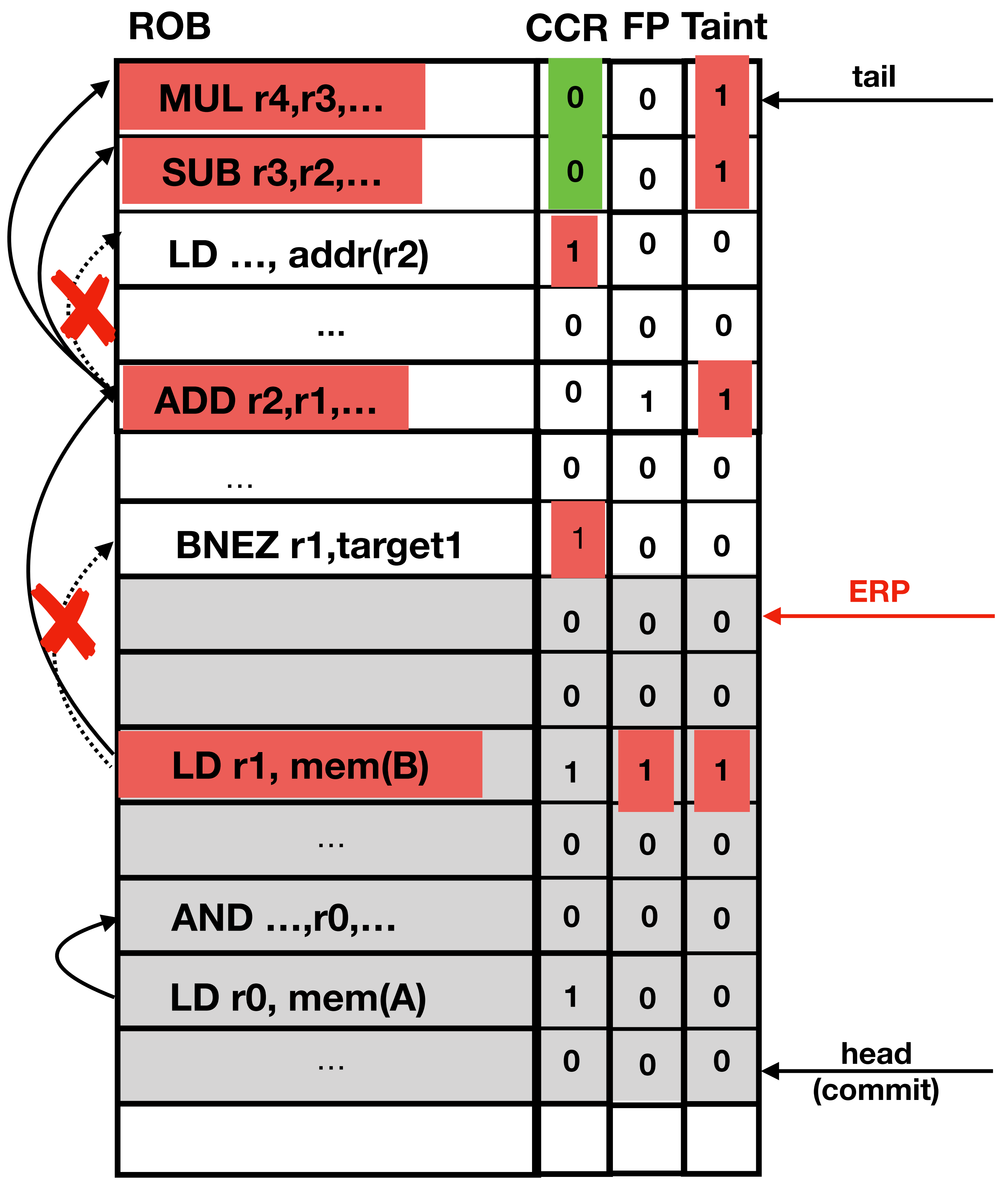
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction
  - At ERP for high leakage risk instructions, from tainted instructions

Reorder Buffer



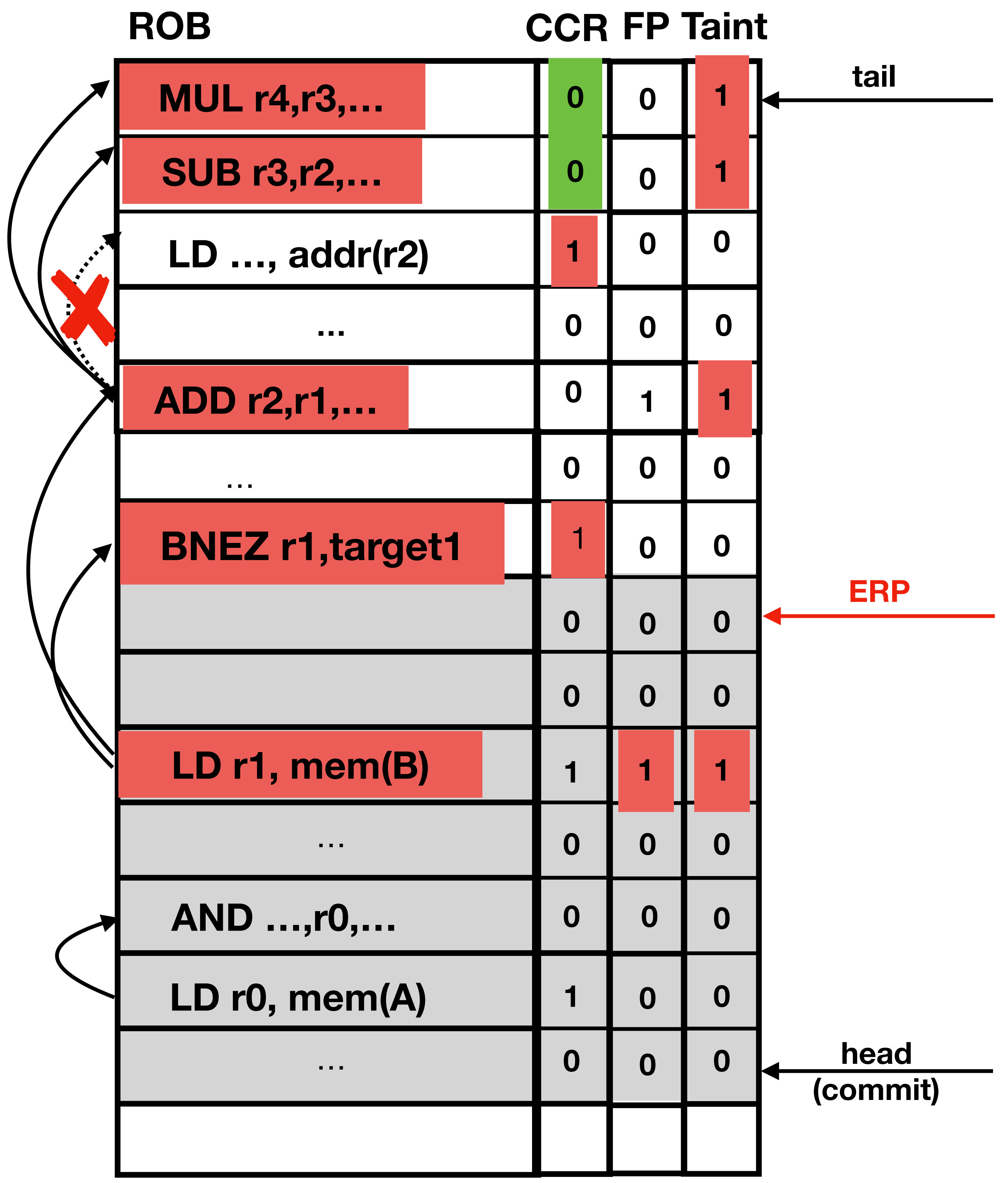
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction
  - At ERP for high leakage risk instructions, from tainted instructions

Reorder Buffer



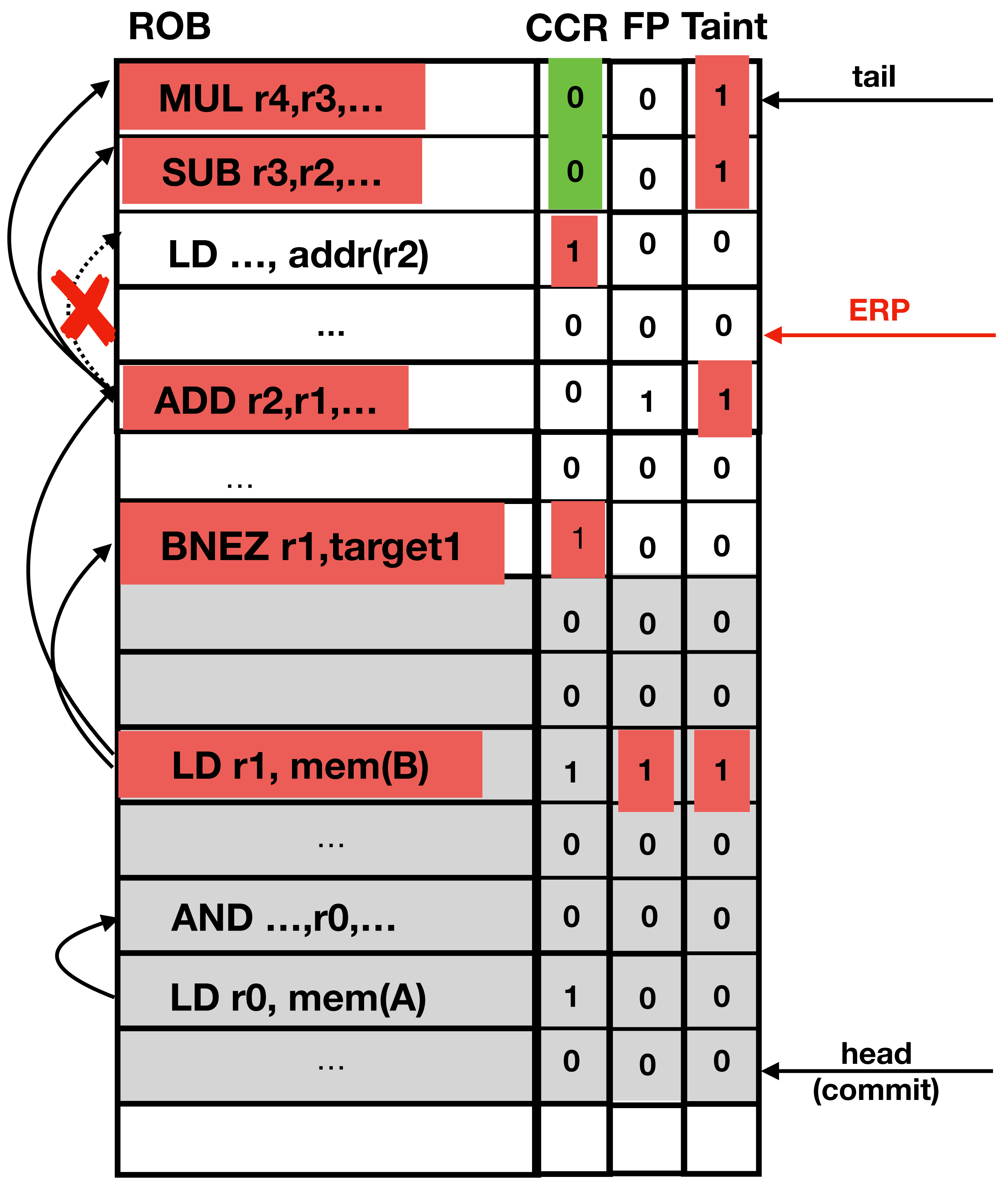
ROB	CCR	FP	Taint	
MUL r4,r3,...	0	0	1	tail
SUB r3,r2,...	0	0	1	
LD ..., addr(r2)	1	0	0	
...	0	0	0	
ADD r2,r1,...	0	1	1	
...	0	0	0	
BNEZ r1,target1	1	0	0	
	0	0	0	ERP
	0	0	0	
LD r1, mem(B)	1	1	1	
...	0	0	0	
AND ...,r0,...	0	0	0	
LD r0, mem(A)	1	0	0	
...	0	0	0	head (commit)

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction
  - At ERP for high leakage risk instructions, from tainted instructions

Reorder Buffer



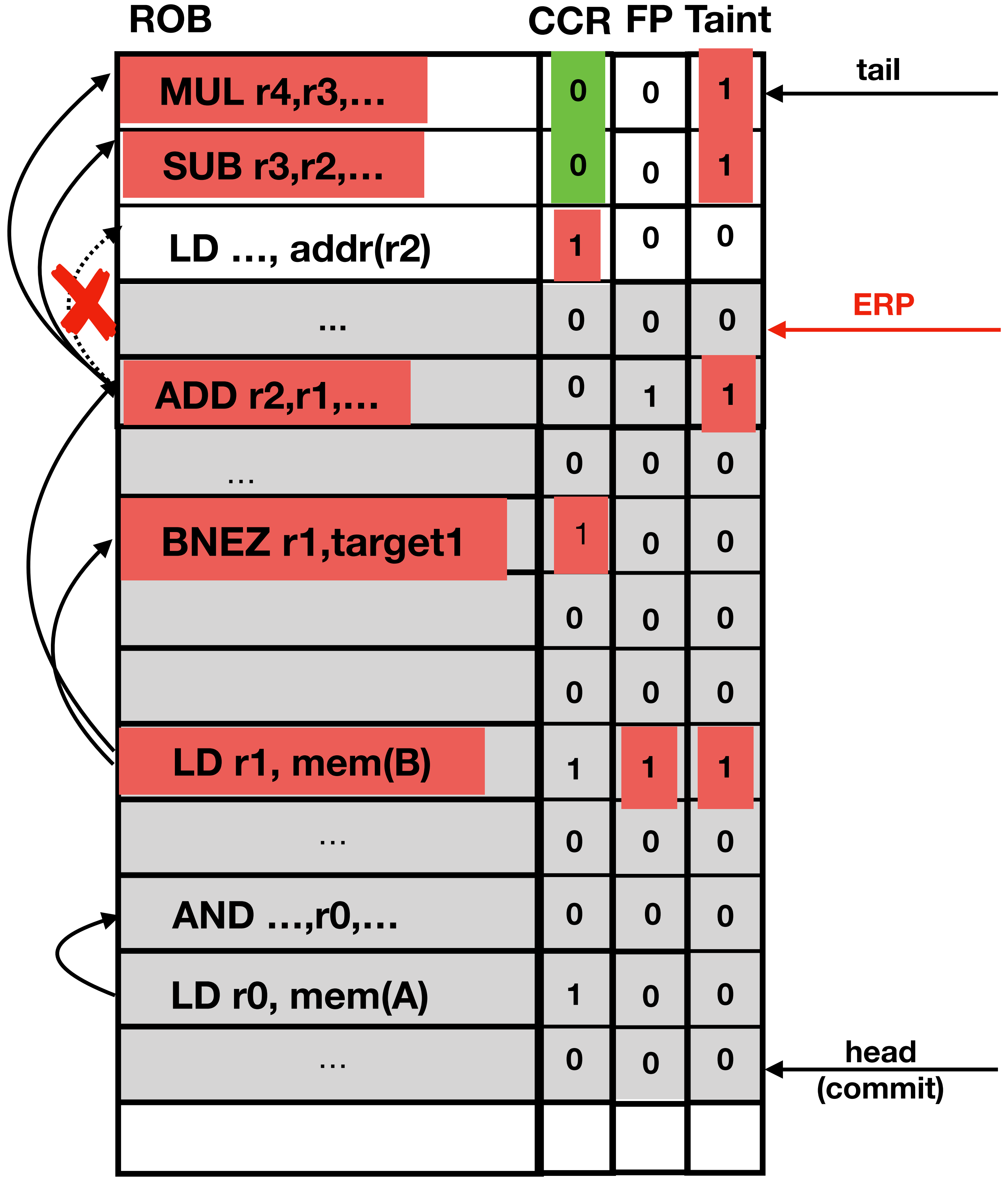
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction
  - At ERP for high leakage risk instructions, from tainted instructions

Reorder Buffer



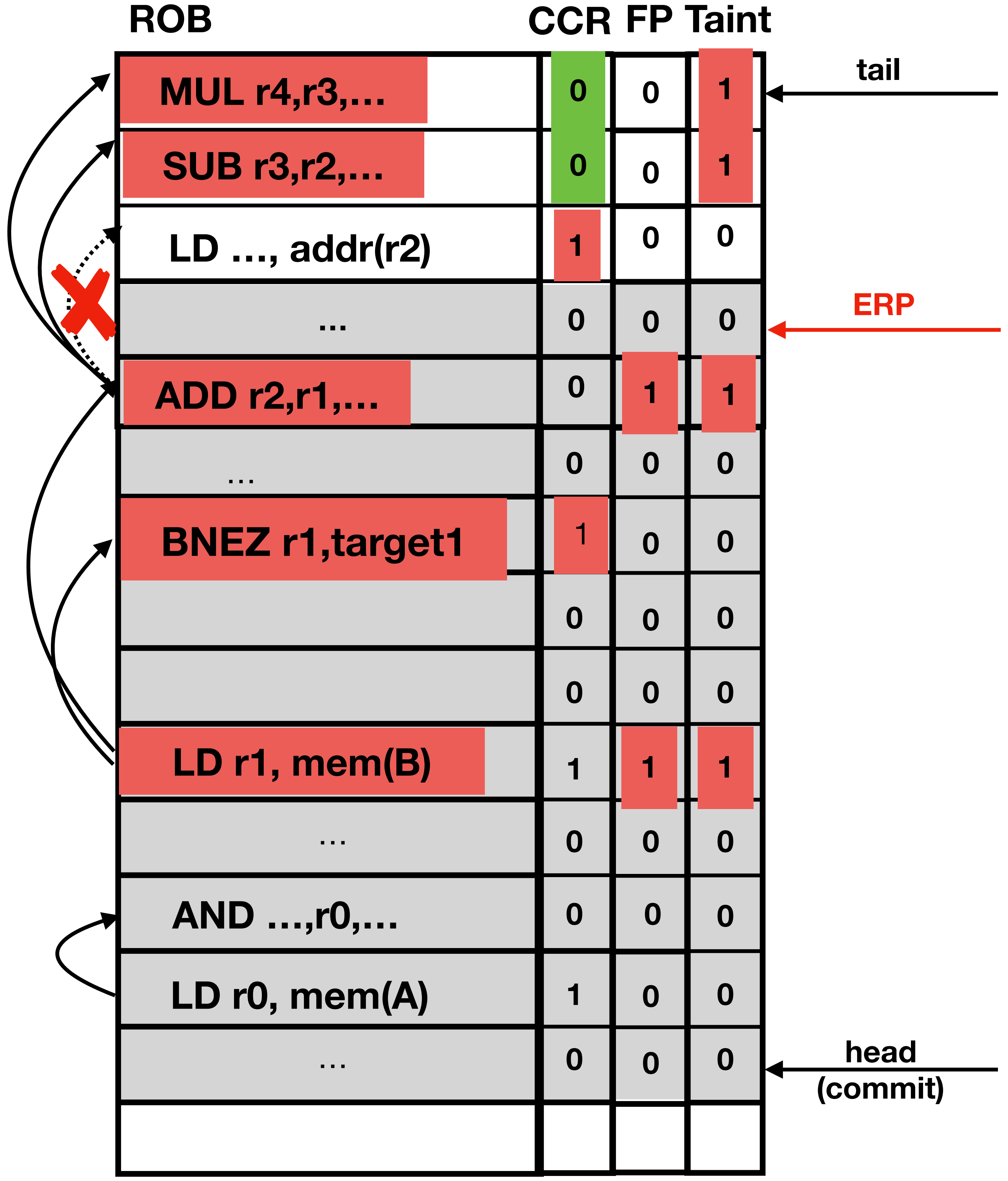
ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction
  - At ERP for high leakage risk instructions, from tainted instructions

Reorder Buffer

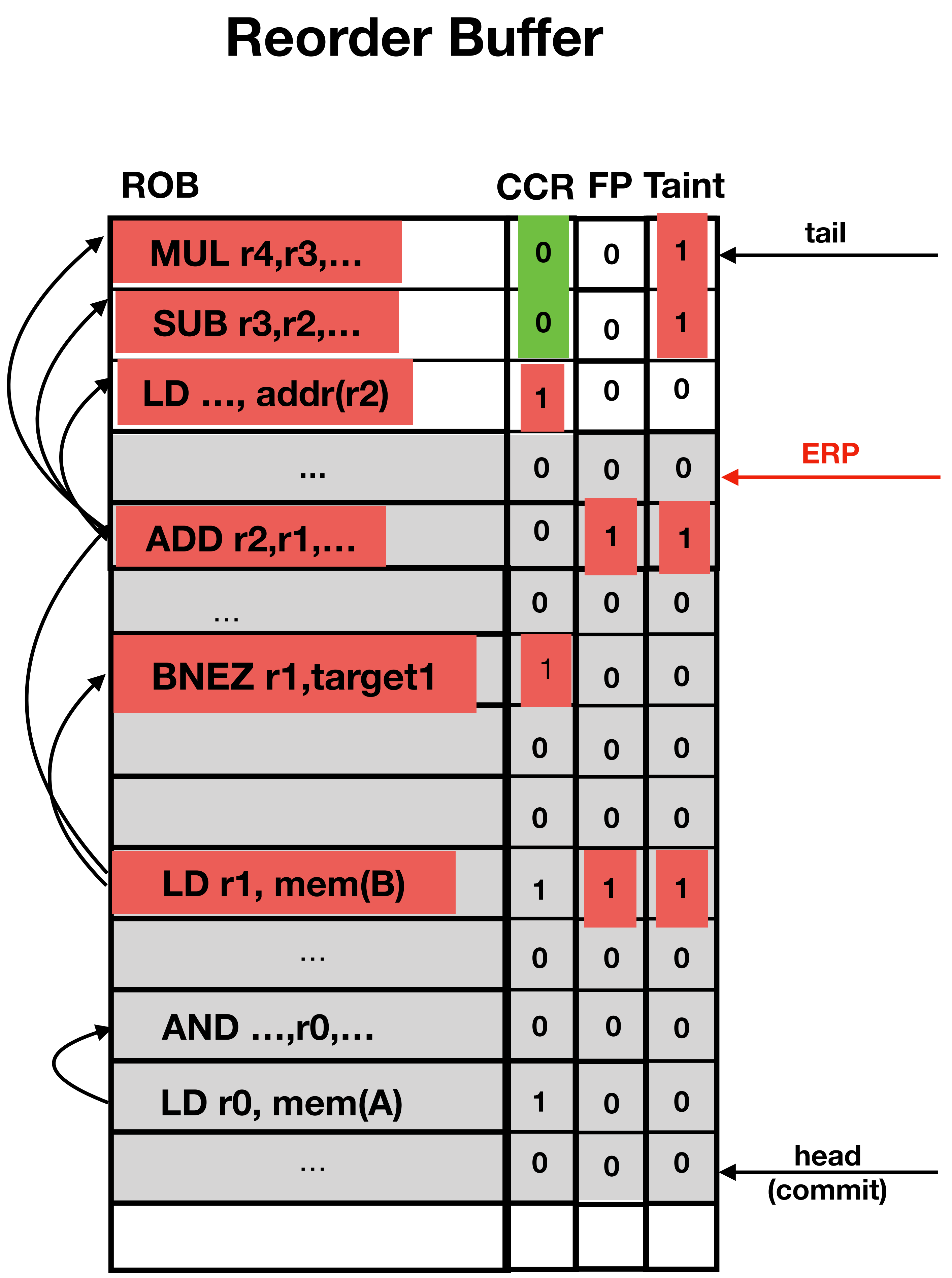


ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

SpecShield ERP+

- A covert channel-specific optimization
- Hypothesis: not all instructions form covert channels, loads delaying forwarding to all dependents is possibly still too conservative
  - Some classes of instructions may pose a low leakage risk (maybe arithmetic ops)
- Idea: classify instructions as high/low Covert Channel Risk
- Speculative data forwarded:
  - Immediately to low leakage risk instructions
    - Taint used to indicate if instruction computed with speculative data
    - Speculative data propagates along dependency chain until reaching high CCR instruction
  - At ERP for high leakage risk instructions, from tainted instructions

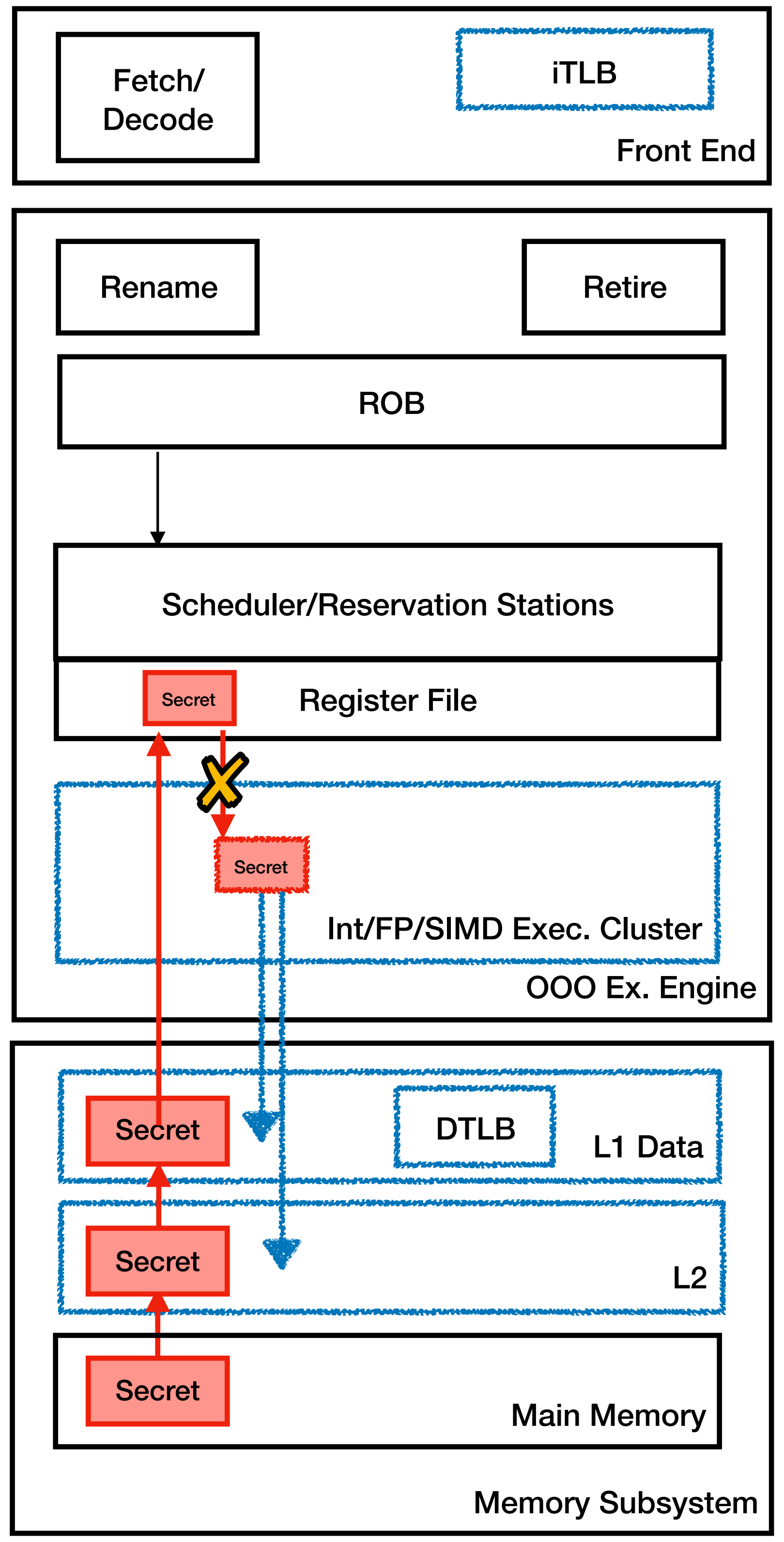


Reorder Buffer

ROB	CCR	FP	Taint
MUL r4,r3,...	0	0	1
SUB r3,r2,...	0	0	1
LD ..., addr(r2)	1	0	0
...	0	0	0
ADD r2,r1,...	0	1	1
...	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	1
...	0	0	0
AND ...,r0,...	0	0	0
LD r0, mem(A)	1	0	0
...	0	0	0

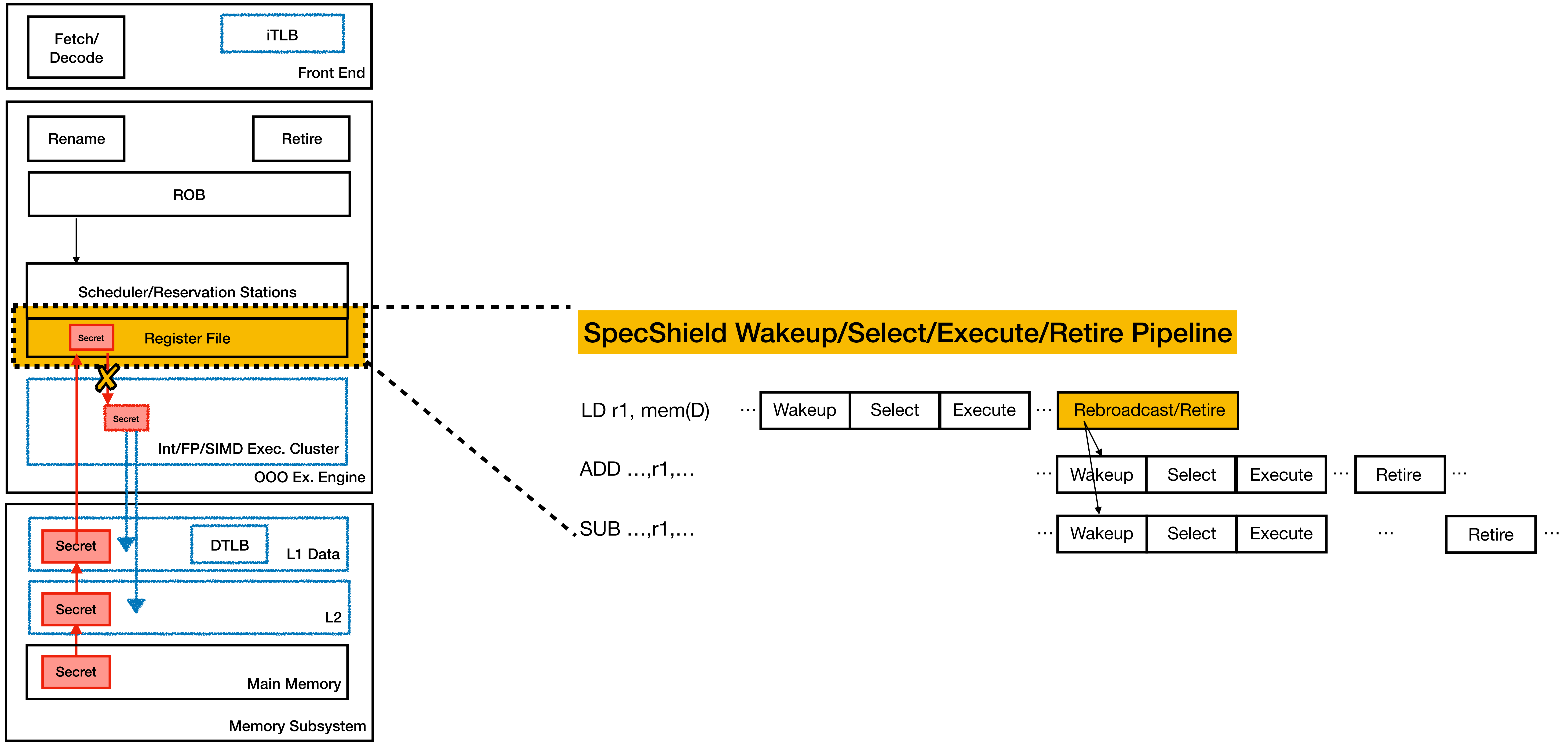
High CCR	LDs, Branches
Low CCR	Rest

SpecShield Hardware Support



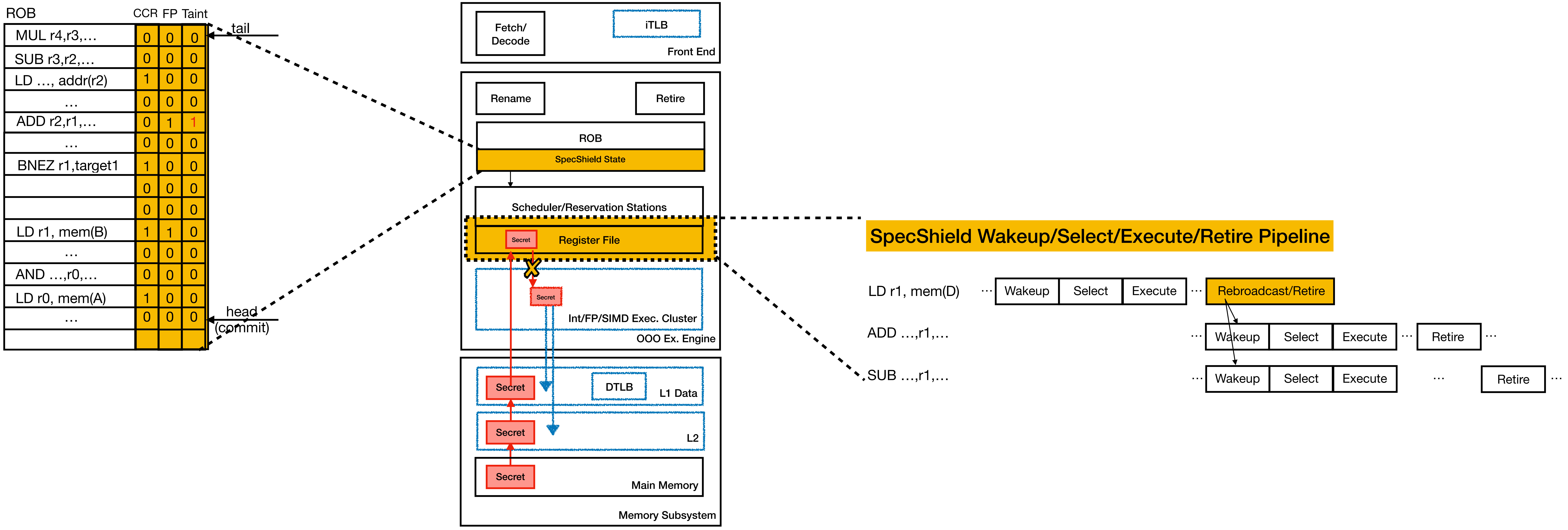
🟨 SpecShield Changes/Additions

SpecShield Hardware Support


Fetch/
Decode
iTLB
Front End
Rename
Retire
ROB
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
SpecShield Wakeup/Select/Execute/Retire Pipeline
LD r1, mem(D)
… Wakeup Select Execute … Rebroadcast/Retire
ADD …,r1,…
… Wakeup Select Execute … Retire …
SUB …,r1,…
… Wakeup Select Execute … Retire …

🟨 SpecShield Changes/Additions

SpecShield Hardware Support



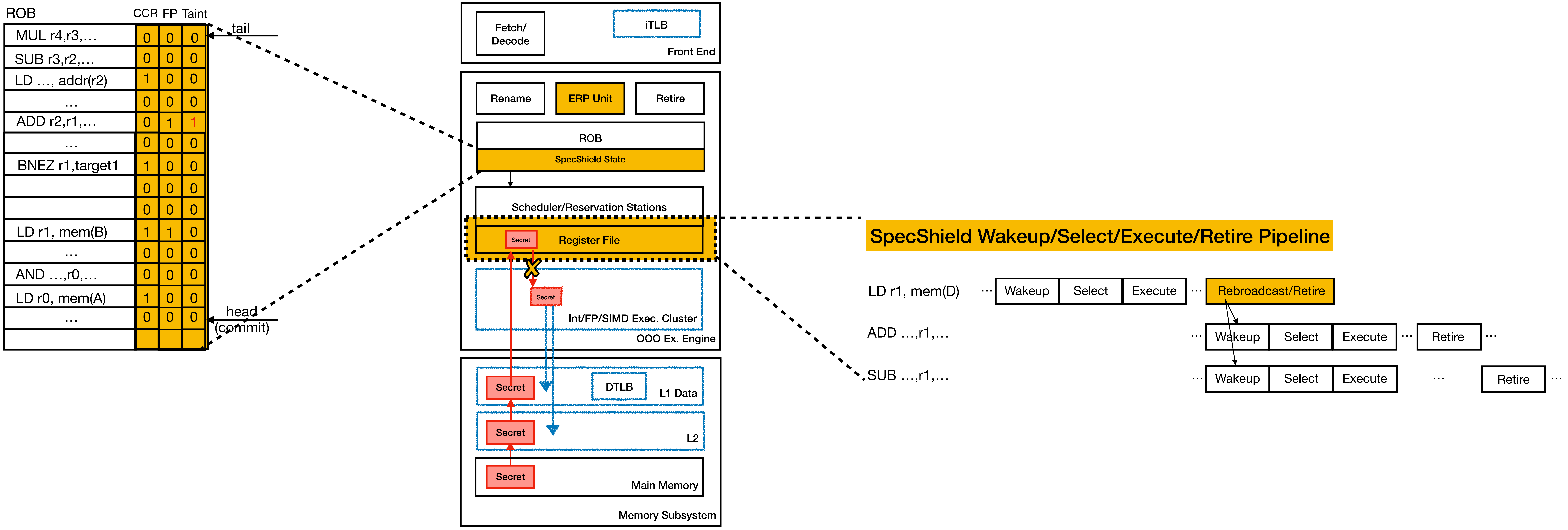
ROB

	CCR	FP	Taint
MUL r4,r3,…	0	0	0
SUB r3,r2,…	0	0	0
LD …, addr(r2)	1	0	0
…	0	0	0
ADD r2,r1,…	0	1	1
…	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
…	0	0	0
AND …,r0,…	0	0	0
LD r0, mem(A)	1	0	0
…	0	0	0

tail
head
(commit)
Fetch/
Decode
iTLB
Front End
Rename
Retire
ROB
SpecShield State
Scheduler/Reservation Stations
Secret
Register File
Secret
Int/FP/SIMD Exec. Cluster
OOO Ex. Engine
Secret
DTLB
L1 Data
Secret
L2
Secret
Main Memory
Memory Subsystem
SpecShield Wakeup/Select/Execute/Retire Pipeline
LD r1, mem(D)
… Wakeup Select Execute … Rebroadcast/Retire
ADD …,r1,…
… Wakeup Select Execute … Retire …
SUB …,r1,…
… Wakeup Select Execute … Retire …

🟨 SpecShield Changes/Additions

SpecShield Hardware Support



ROB

	CCR	FP	Taint
MUL r4,r3,…	0	0	0
SUB r3,r2,…	0	0	0
LD …, addr(r2)	1	0	0
…	0	0	0
ADD r2,r1,…	0	1	1
…	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
…	0	0	0
AND …,r0,…	0	0	0
LD r0, mem(A)	1	0	0
…	0	0	0

SpecShield Wakeup/Select/Execute/Retire Pipeline

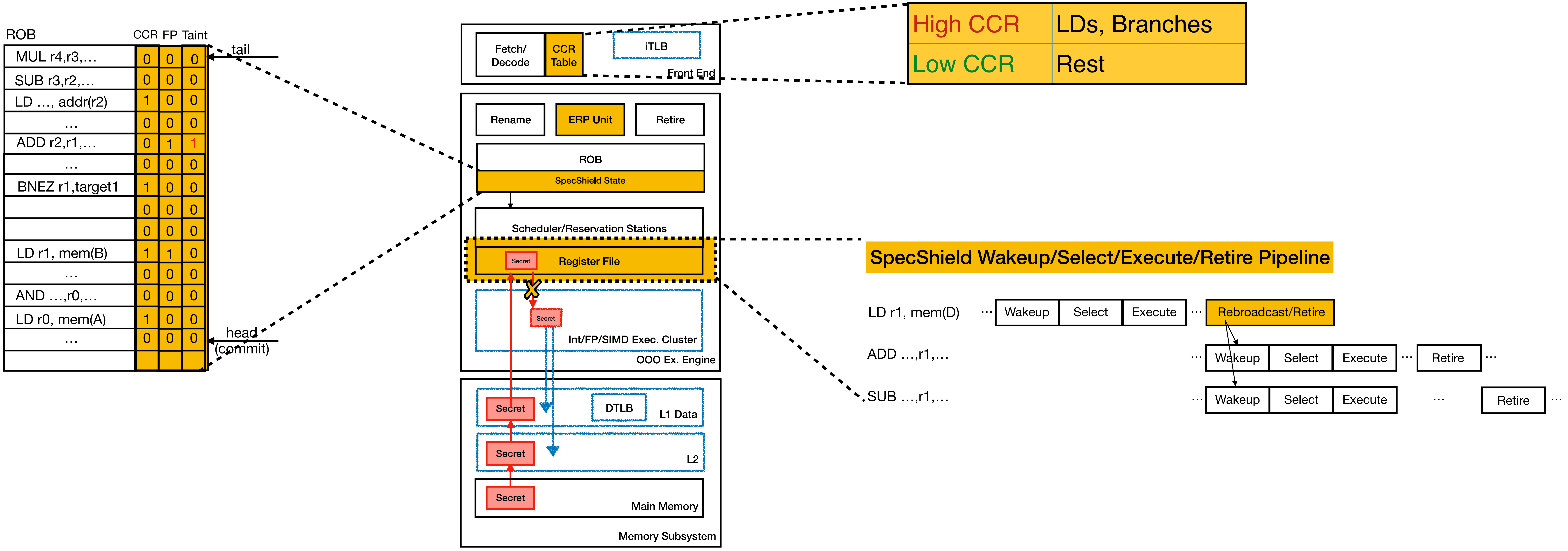
LD r1, mem(D) … Wakeup | Select | Execute … Rebroadcast/Retire

ADD …,r1,… … Wakeup | Select | Execute … Retire …

SUB …,r1,… … Wakeup | Select | Execute … Retire …

🟨 SpecShield Changes/Additions

SpecShield Hardware Support

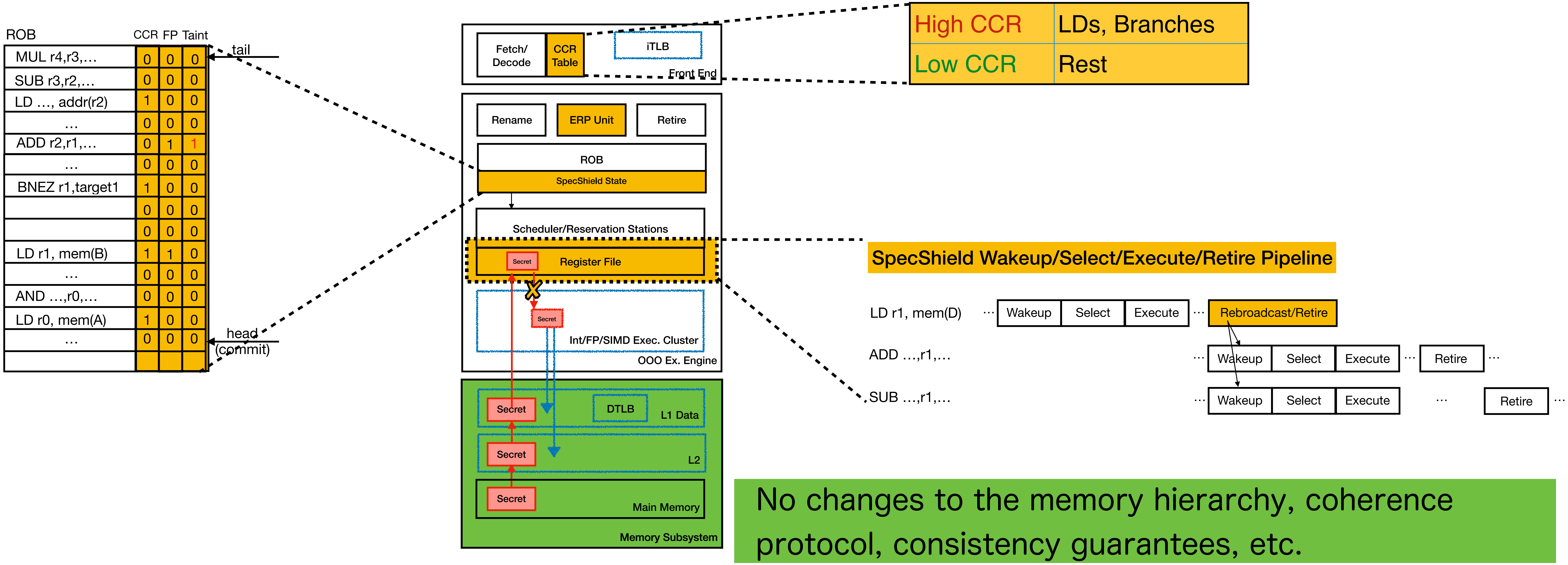


ROB	CCR	FP	Taint
MUL r4,r3,…	0	0	0
SUB r3,r2,…	0	0	0
LD …, addr(r2)	1	0	0
…	0	0	0
ADD r2,r1,…	0	1	1
…	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
…	0	0	0
AND …,r0,…	0	0	0
LD r0, mem(A)	1	0	0
…	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

🟨 SpecShield Changes/Additions

SpecShield Hardware Support



ROB	CCR	FP	Taint
MUL r4,r3,…	0	0	0
SUB r3,r2,…	0	0	0
LD …, addr(r2)	1	0	0
…	0	0	0
ADD r2,r1,…	0	1	1
…	0	0	0
BNEZ r1,target1	1	0	0
	0	0	0
	0	0	0
LD r1, mem(B)	1	1	0
…	0	0	0
AND …,r0,…	0	0	0
LD r0, mem(A)	1	0	0
…	0	0	0

High CCR	LDs, Branches
Low CCR	Rest

No changes to the memory hierarchy, coherence protocol, consistency guarantees, etc.

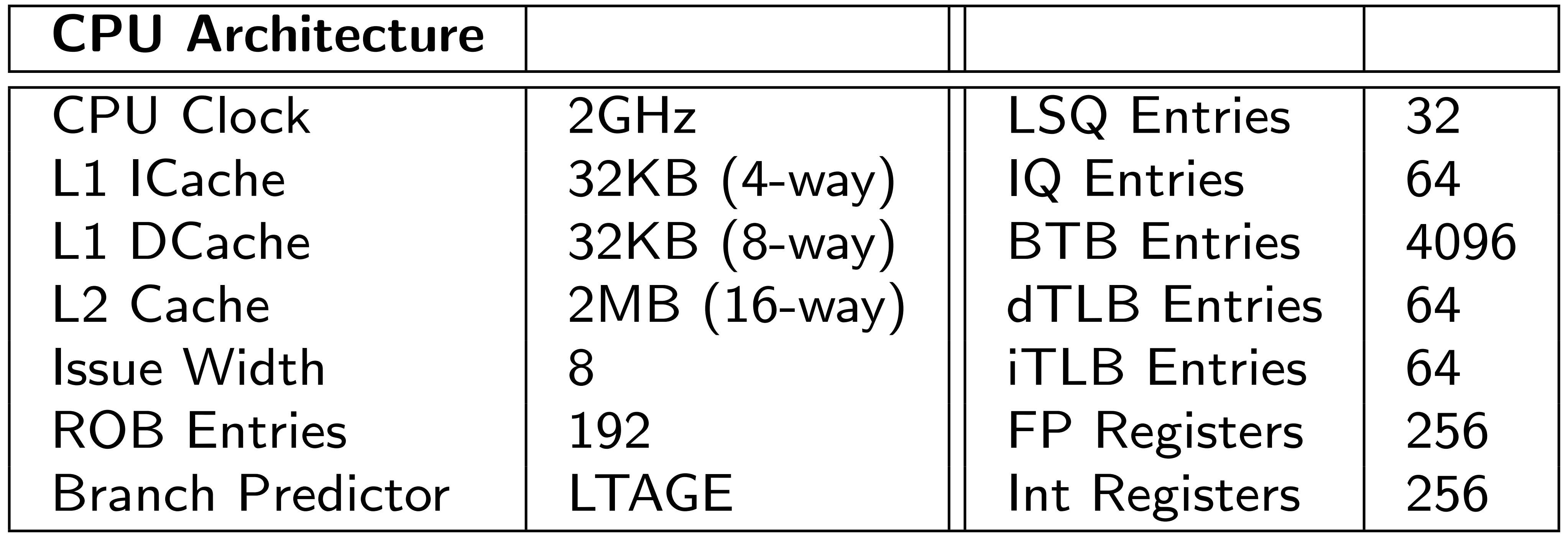
🟨 SpecShield Changes/Additions

Evaluation Methodology

- Experimental Platform:
  - Simulator: gem5, full-system mode, Ubuntu 14.04 OS
  - Benchmarks: spec2006, reference input set
  - Simpoints: Used to select 10 most representative regions of 1B instructions

Evaluation Methodology

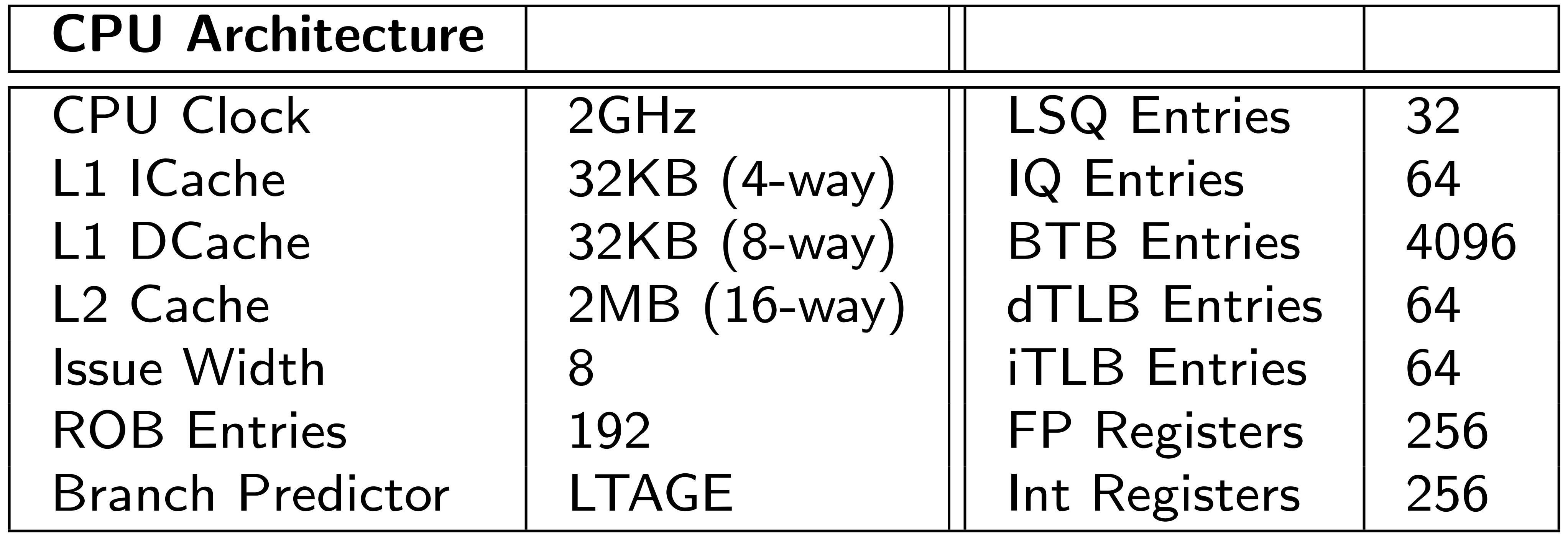
- Experimental Platform:
  - Simulator: gem5, full-system mode, Ubuntu 14.04 OS
  - Benchmarks: spec2006, reference input set
  - Simpoints: Used to select 10 most representative regions of 1B instructions



CPU Architecture			
CPU Clock	2GHz	LSQ Entries	32
L1 ICache	32KB (4-way)	IQ Entries	64
L1 DCache	32KB (8-way)	BTB Entries	4096
L2 Cache	2MB (16-way)	dTLB Entries	64
Issue Width	8	iTLB Entries	64
ROB Entries	192	FP Registers	256
Branch Predictor	LTAGE	Int Registers	256

Evaluation Methodology

- Experimental Platform:
  - Simulator: gem5, full-system mode, Ubuntu 14.04 OS
  - Benchmarks: spec2006, reference input set
  - Simpoints: Used to select 10 most representative regions of 1B instructions

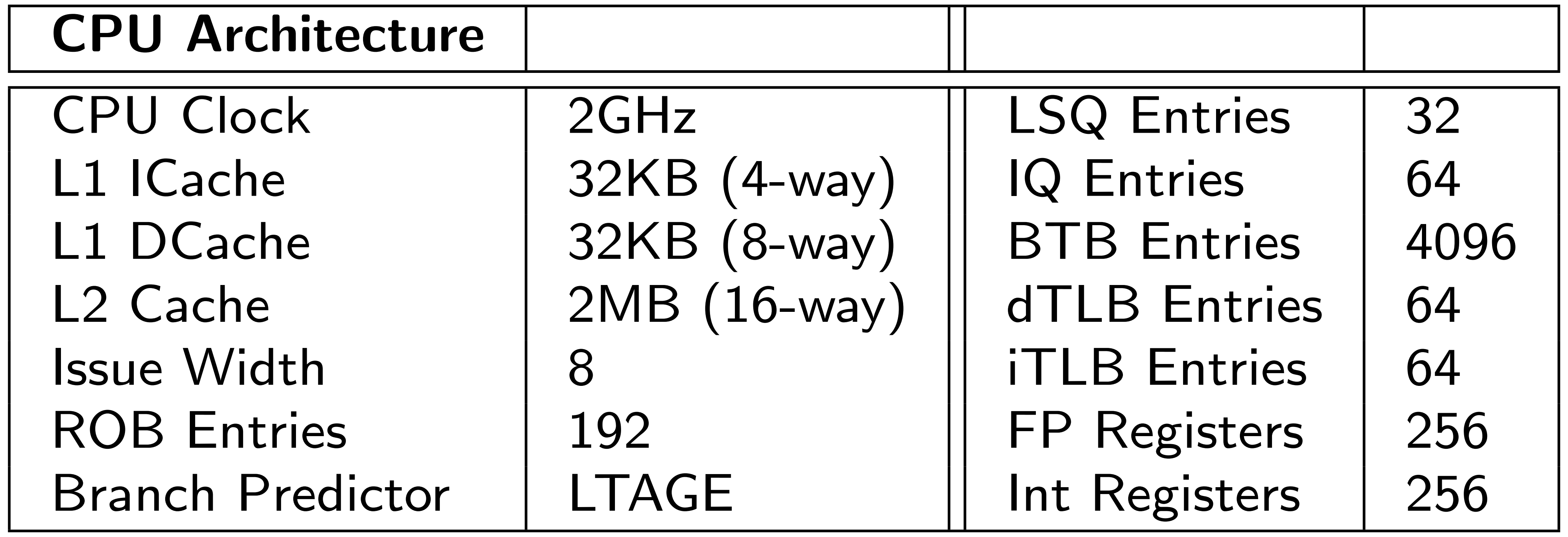


CPU Architecture			
CPU Clock	2GHz	LSQ Entries	32
L1 ICache	32KB (4-way)	IQ Entries	64
L1 DCache	32KB (8-way)	BTB Entries	4096
L2 Cache	2MB (16-way)	dTLB Entries	64
Issue Width	8	iTLB Entries	64
ROB Entries	192	FP Registers	256
Branch Predictor	LTAGE	Int Registers	256

- SpecShield STL, ERP, ERP+

Evaluation Methodology

- Experimental Platform:
  - Simulator: gem5, full-system mode, Ubuntu 14.04 OS
  - Benchmarks: spec2006, reference input set
  - Simpoints: Used to select 10 most representative regions of 1B instructions

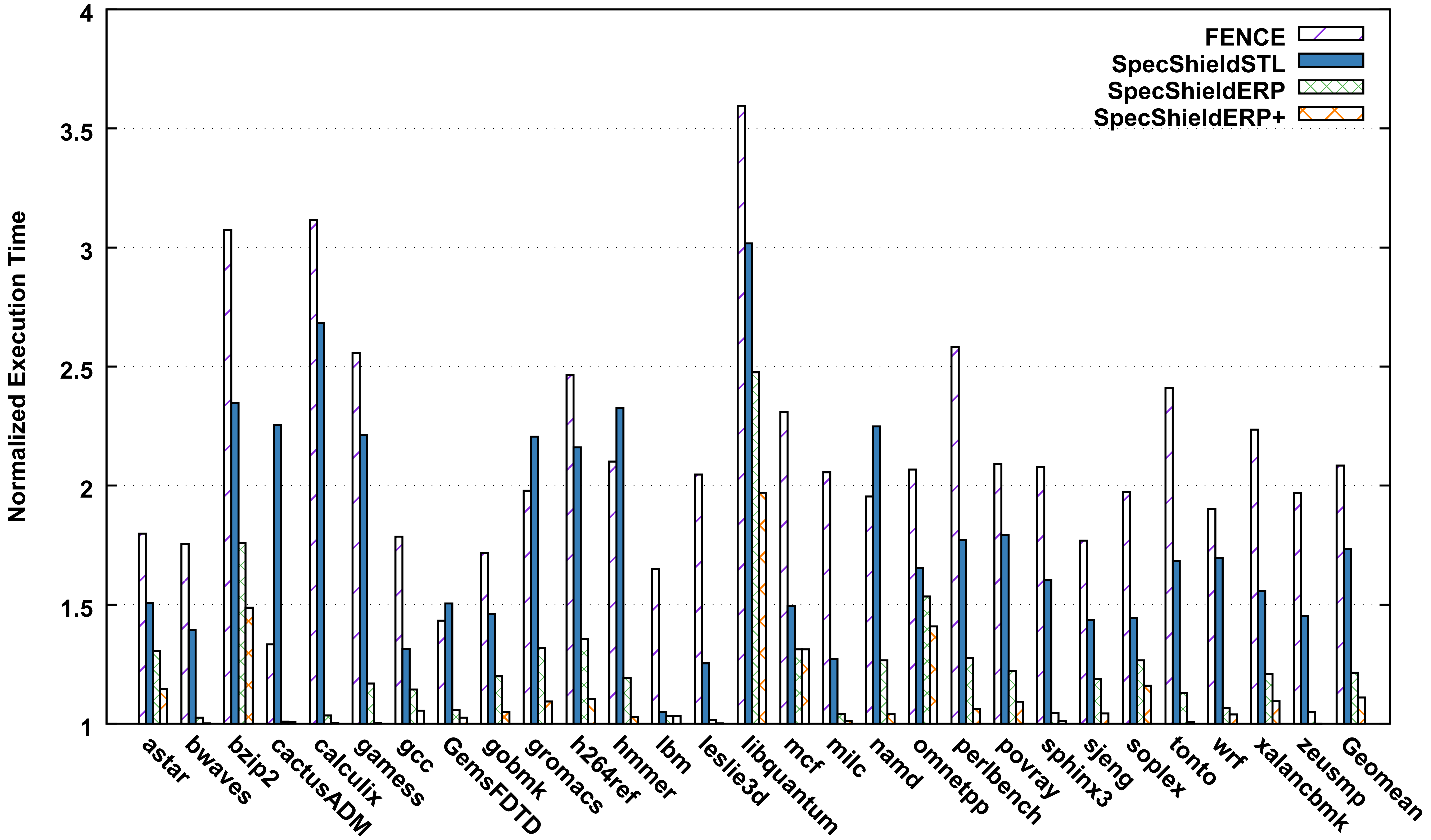


CPU Architecture			
CPU Clock	2GHz	LSQ Entries	32
L1 ICache	32KB (4-way)	IQ Entries	64
L1 DCache	32KB (8-way)	BTB Entries	4096
L2 Cache	2MB (16-way)	dTLB Entries	64
Issue Width	8	iTLB Entries	64
ROB Entries	192	FP Registers	256
Branch Predictor	LTAGE	Int Registers	256

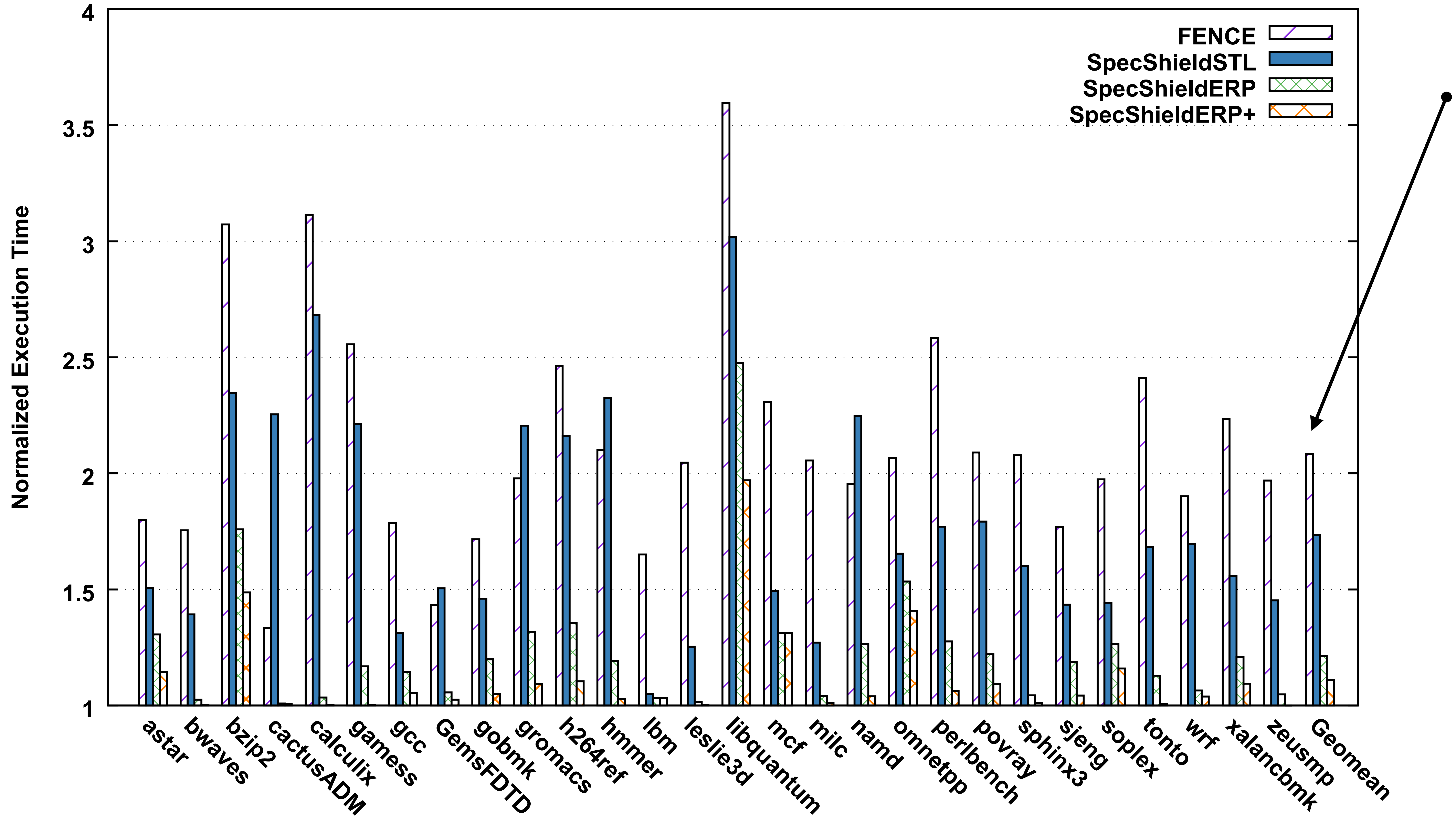
- SpecShield STL, ERP, ERP+
- LFENCE* serialization after every branch

*Intel, Speculative execution side channel mitigations. Intel, 2018. https://software.intel.com/security-software-guidance/api-app/sites/default/files/336996-Speculative-Execution-Side- Channel-Mitigations.pdf

Performance

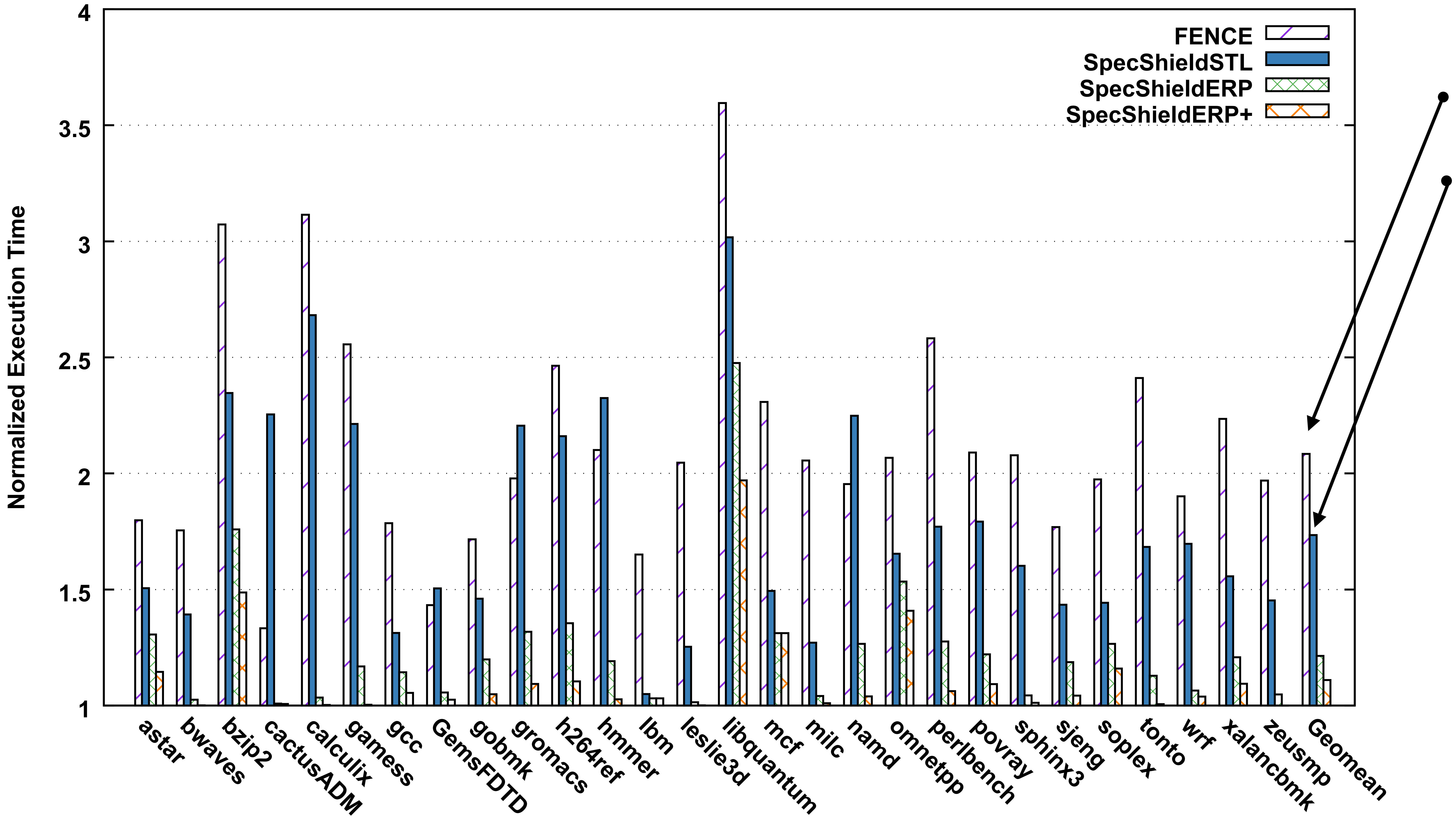


Performance



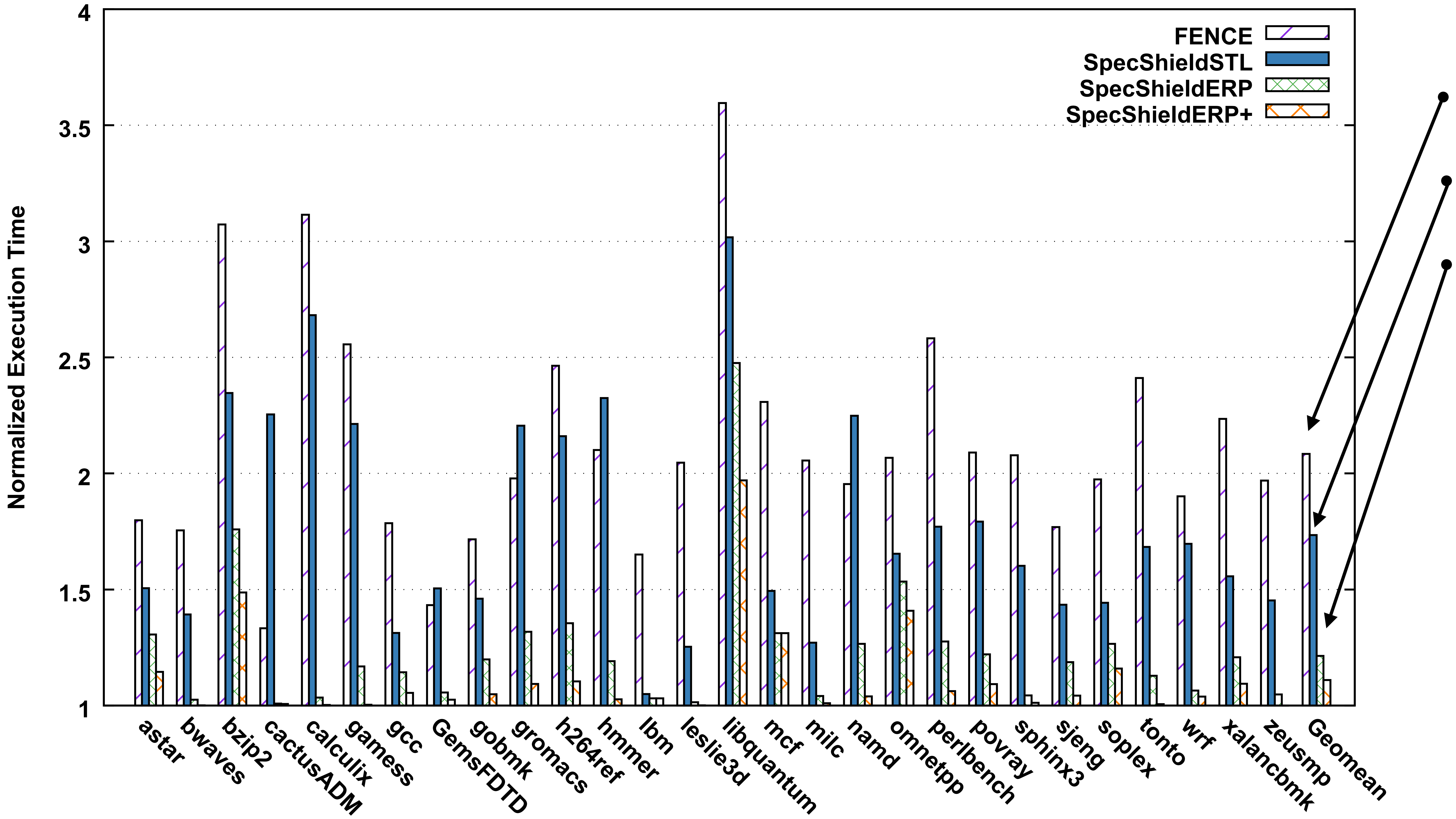
- LFENCE > 2X

Performance



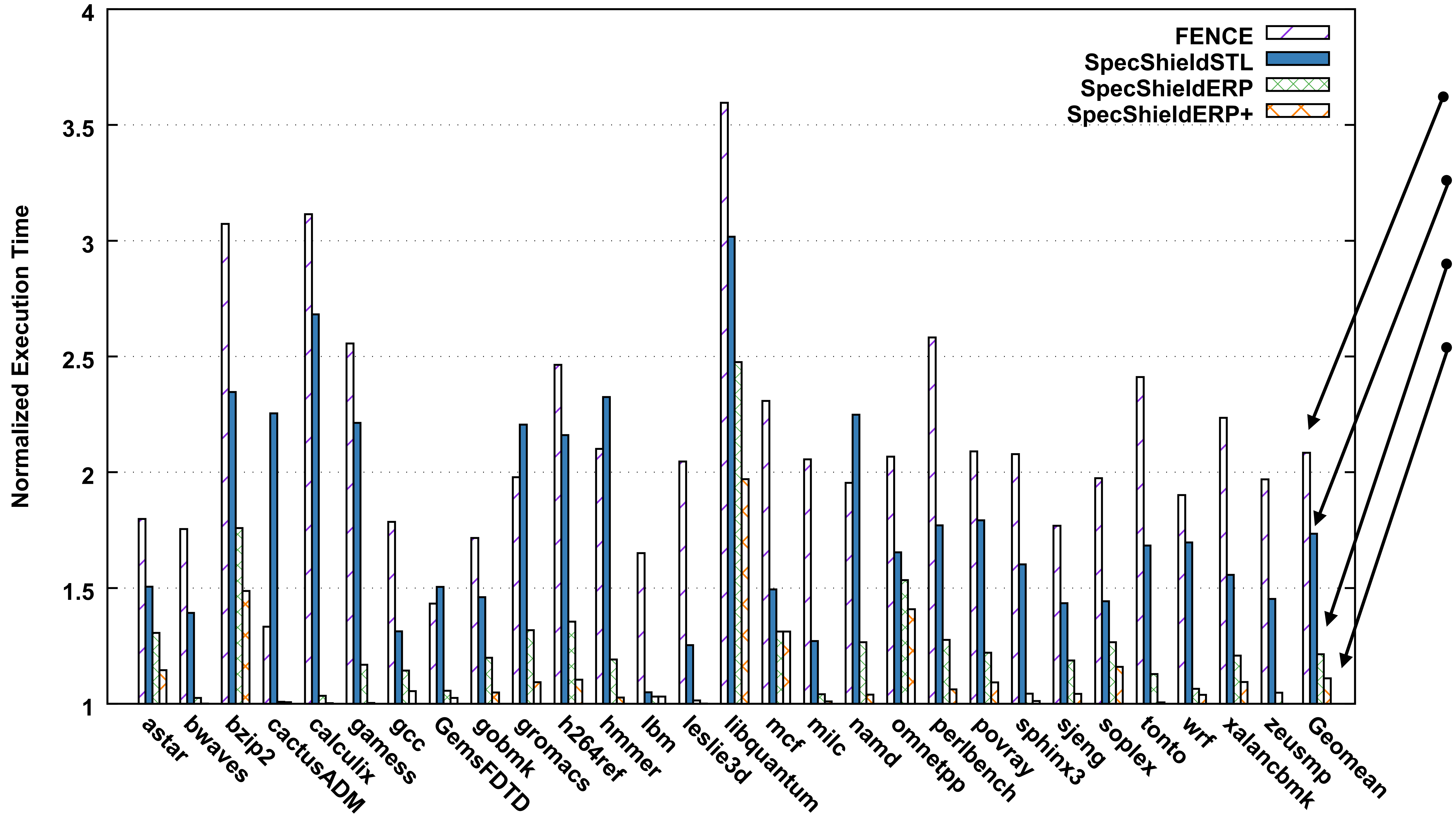
- LFENCE > 2X
- SpecShieldSTL 73%

Performance



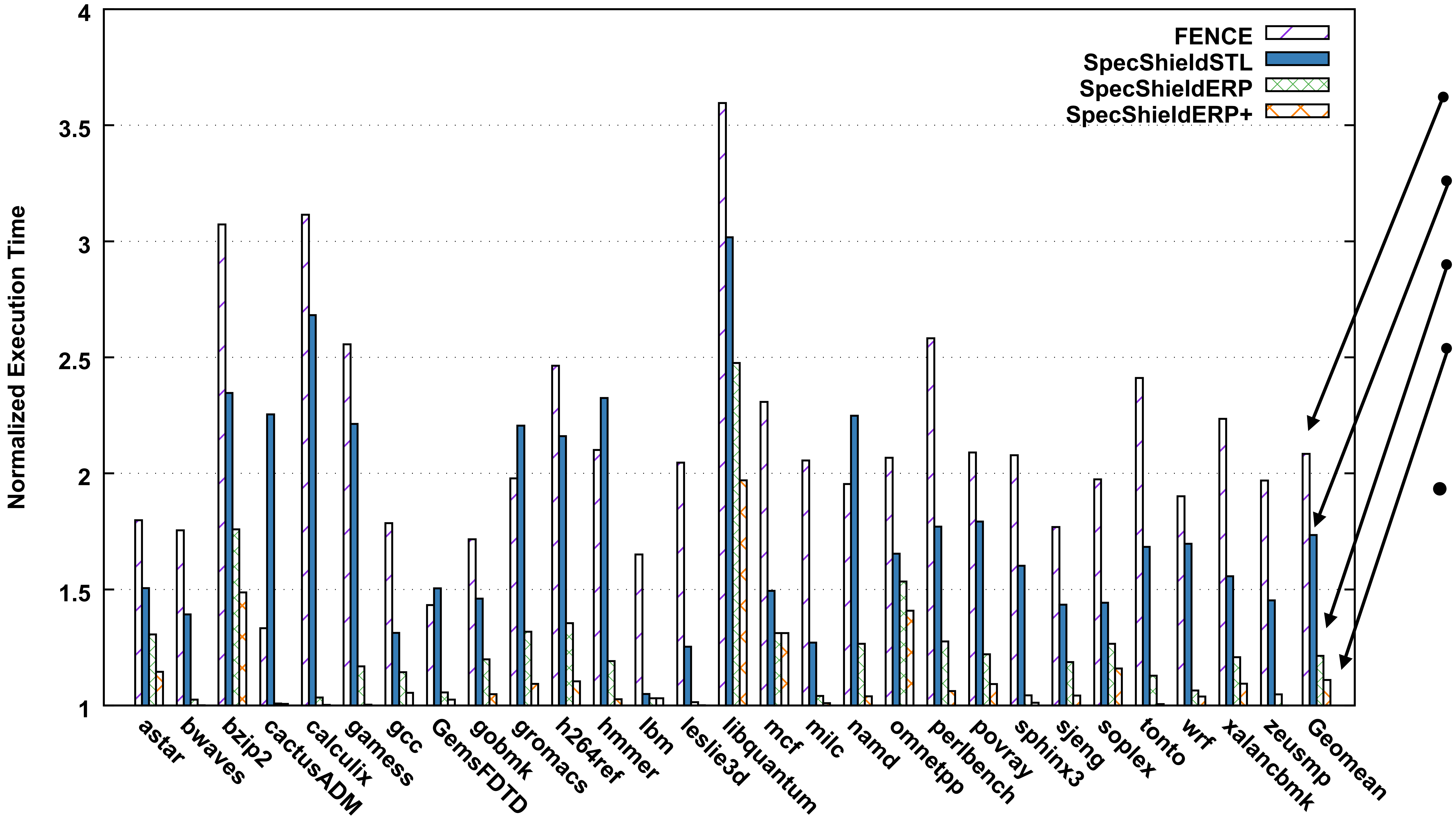
- LFENCE > 2X
- SpecShieldSTL 73%
- SpecShieldERP 21%

Performance



- LFENCE > 2X
- SpecShieldSTL 73%
- SpecShieldERP 21%
- SpecShieldERP+ 10%

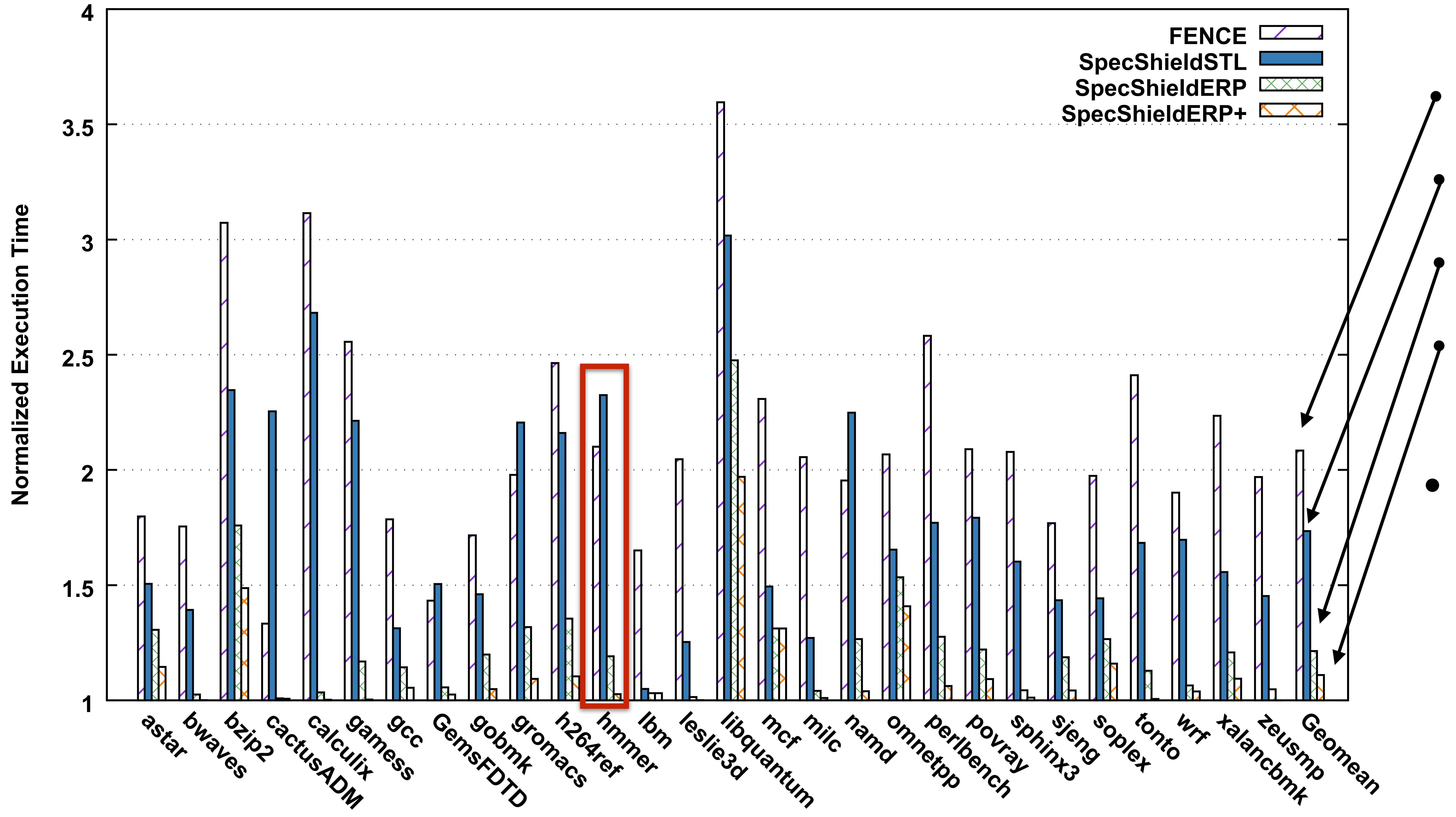
Performance



- LFENCE > 2X
- SpecShieldSTL 73%
- SpecShieldERP 21%
- SpecShieldERP+ 10%

- Benchmarks with low miss rates most impacted

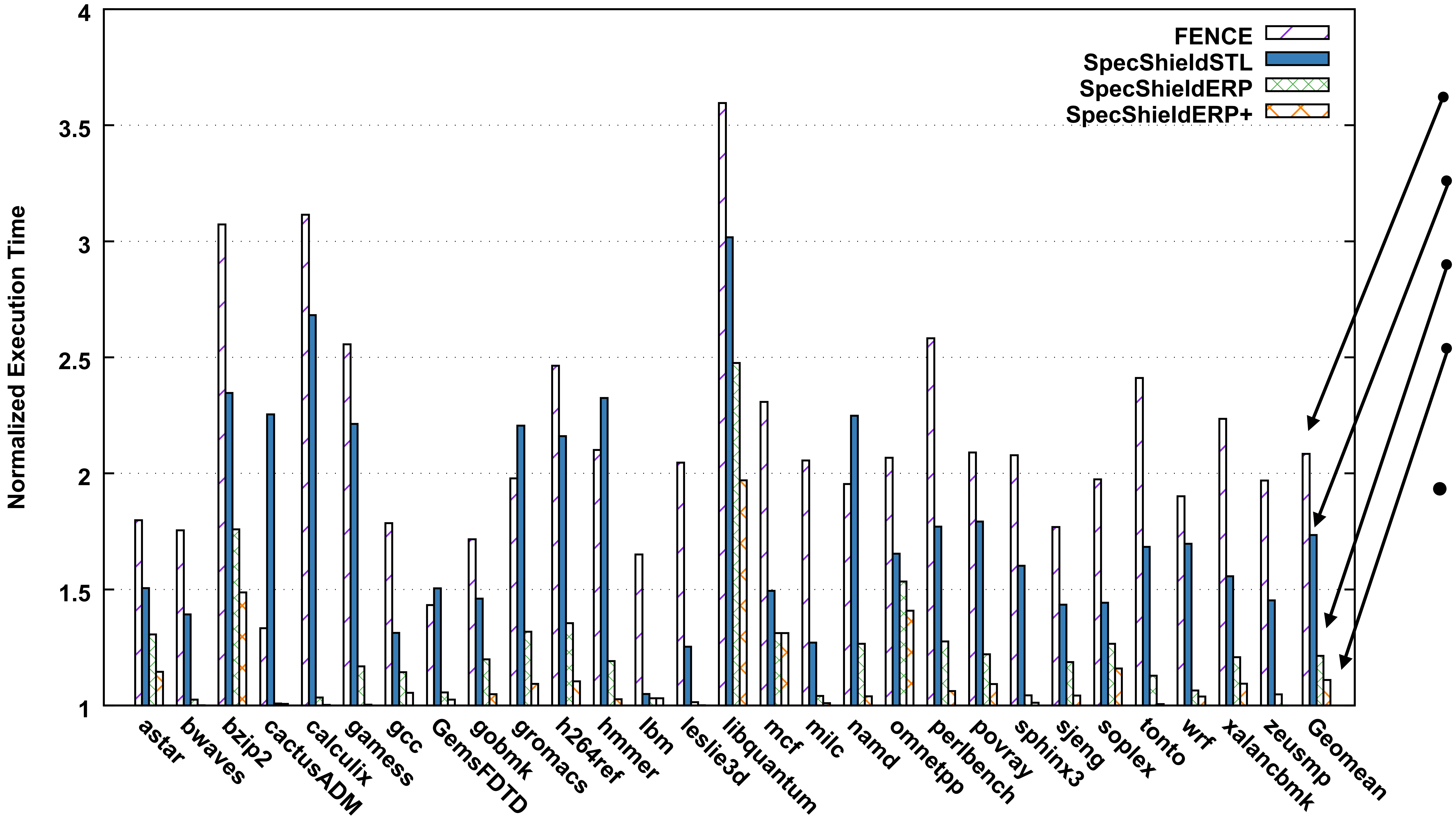
Performance



- LFENCE > 2X
- SpecShieldSTL 73%
- SpecShieldERP 21%
- SpecShieldERP+ 10%

- Benchmarks with low miss rates most impacted

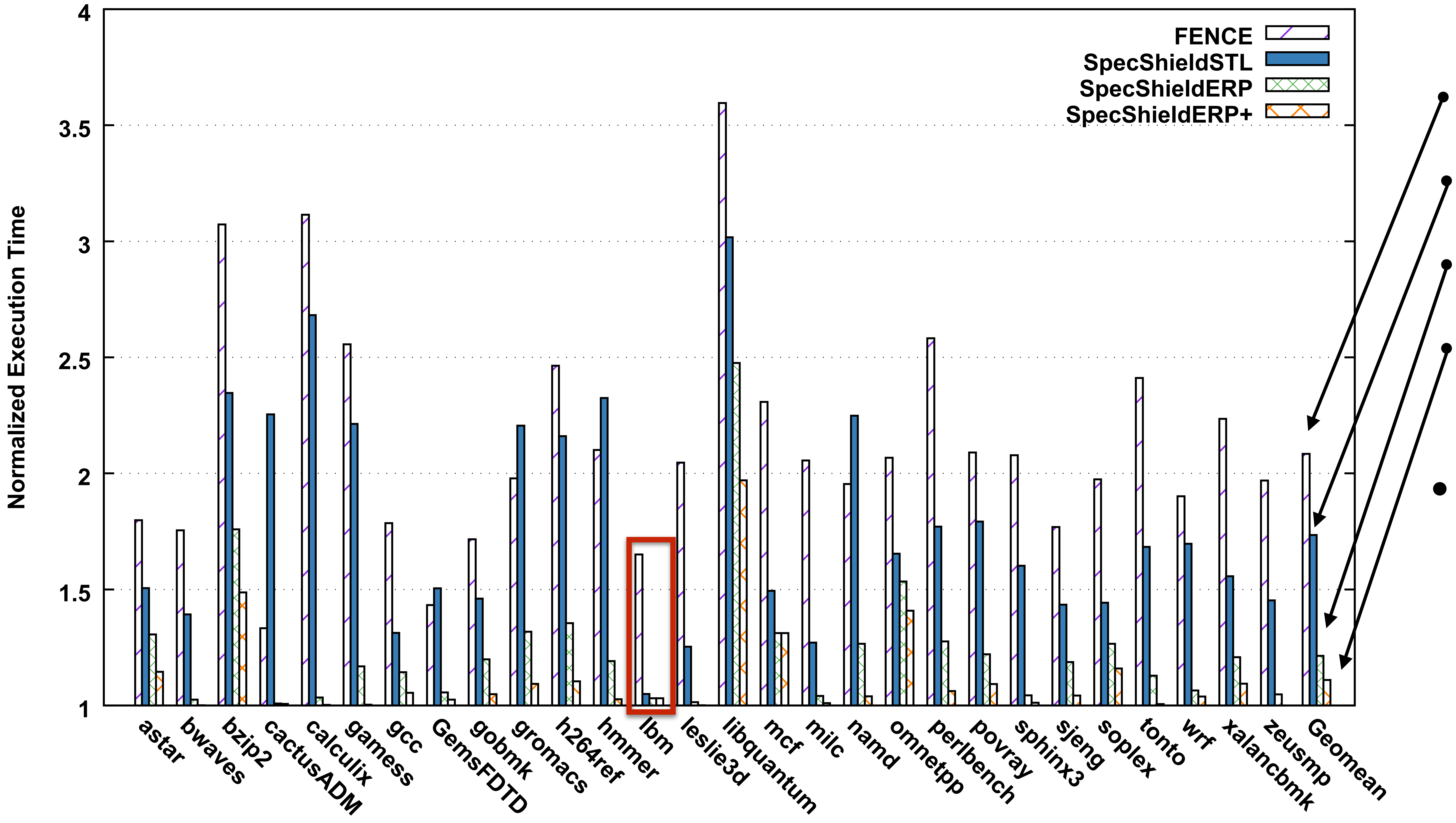
Performance



- LFENCE > 2X
- SpecShieldSTL 73%
- SpecShieldERP 21%
- SpecShieldERP+ 10%

- Benchmarks with low miss rates most impacted

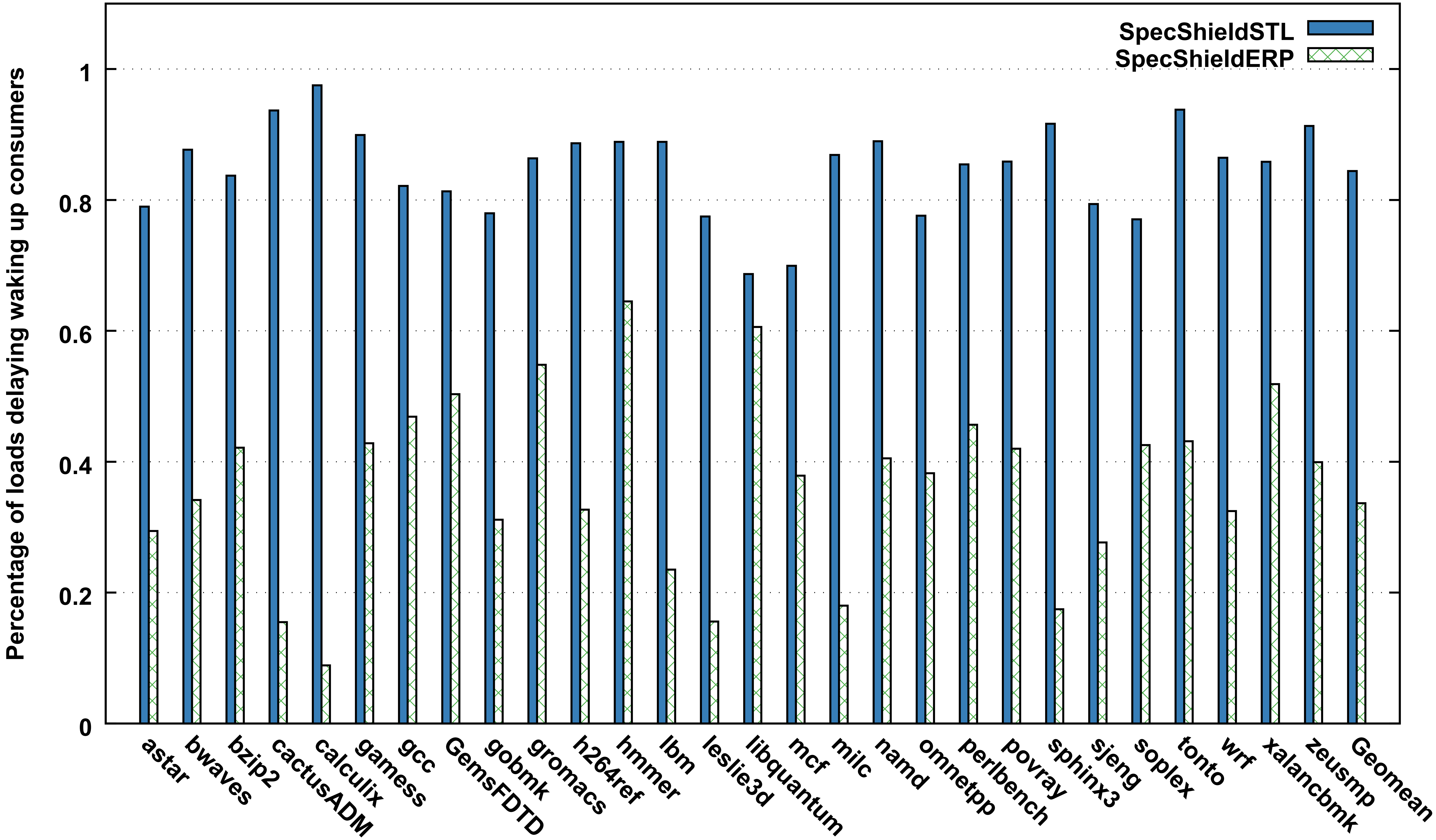
Performance



- LFENCE > 2X
- SpecShieldSTL 73%
- SpecShieldERP 21%
- SpecShieldERP+ 10%

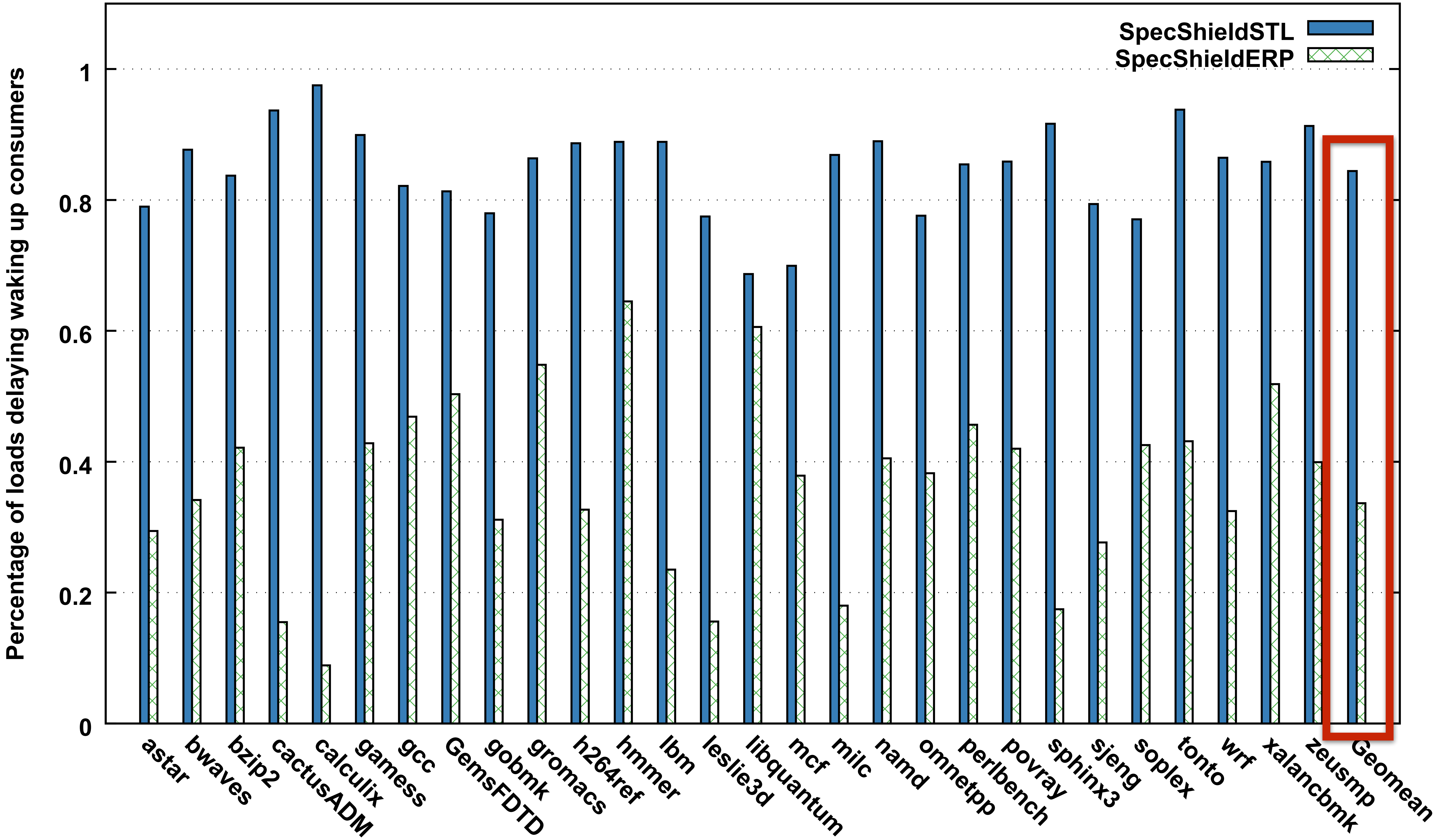
- Benchmarks with low miss rates most impacted

Loads Delaying in SpecShield STL and ERP



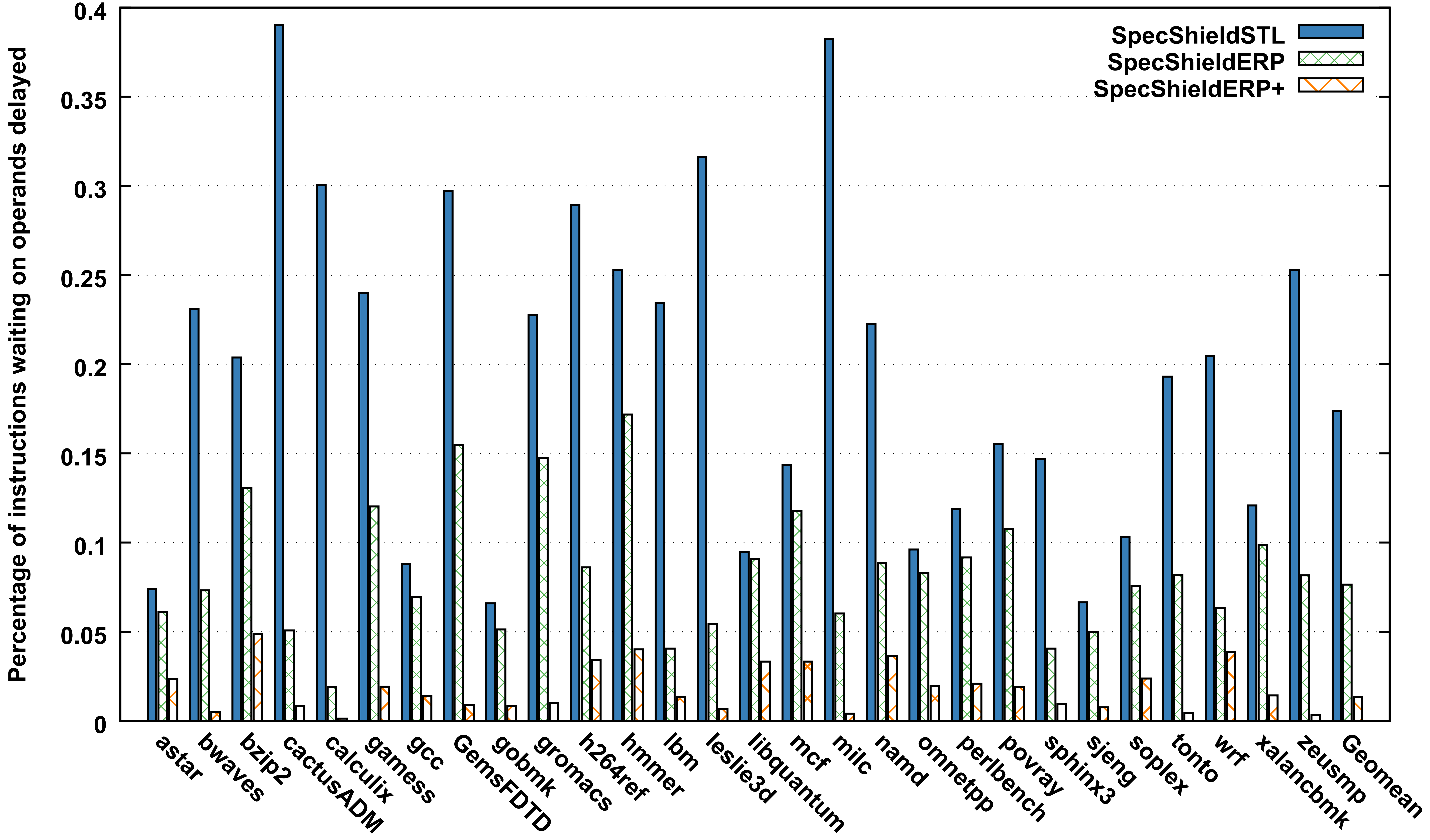
- STL forces most loads to delay (84%)
- ERP cuts that to < 40%

Loads Delaying in SpecShield STL and ERP



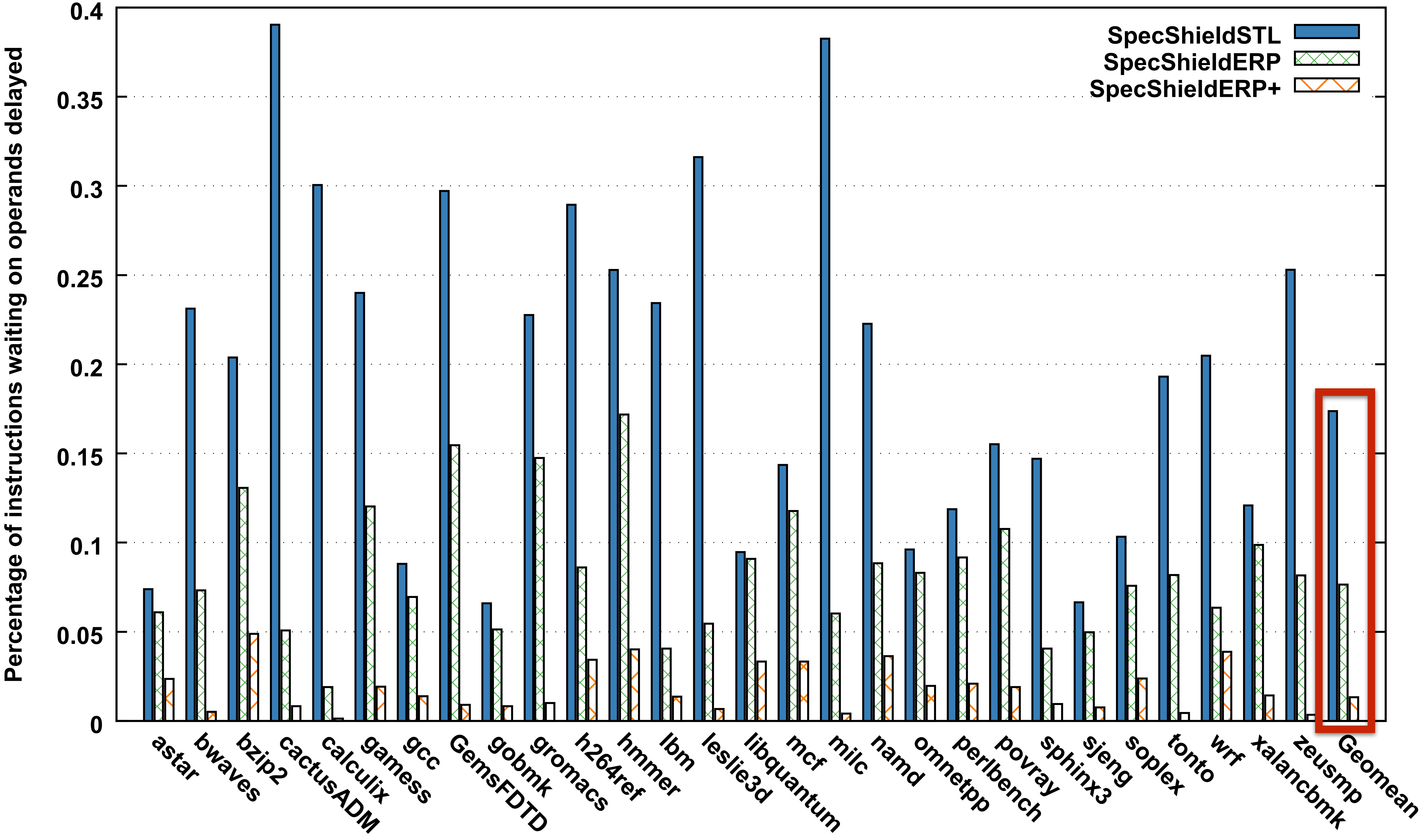
- STL forces most loads to delay (84%)
- ERP cuts that to < 40%

Percentage Instructions Delayed



- STL: 17%
- ERP: 7%
- ERP+: 1%

Percentage Instructions Delayed



- STL: 17%
- ERP: 7%
- ERP+: 1%

Cache Latency for Spectre Attack

Cache Latency for Spectre Attack

- Spectre-v1 attack, using cache as covert channel

Cache Latency for Spectre Attack

- Spectre-v1 attack, using cache as covert channel

```
if  (x < array1_size)
   y = array2[array1[x] *  256];
```

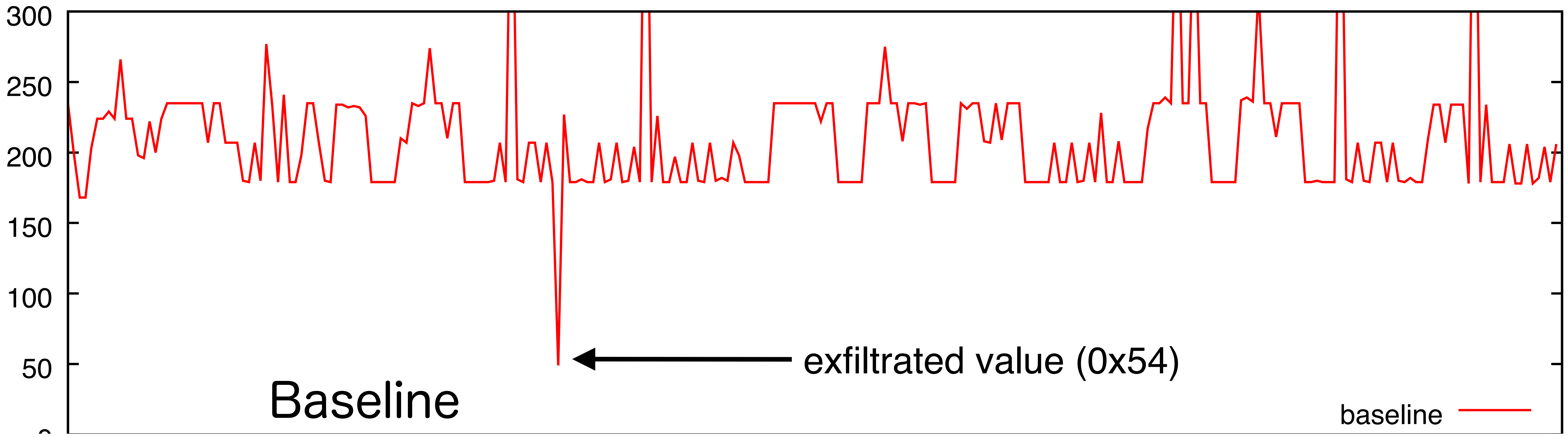
Cache Latency for Spectre Attack

- Spectre-v1 attack, using cache as covert channel

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

- Exfiltrated value visible in access latency

Cache Latency for Spectre Attack

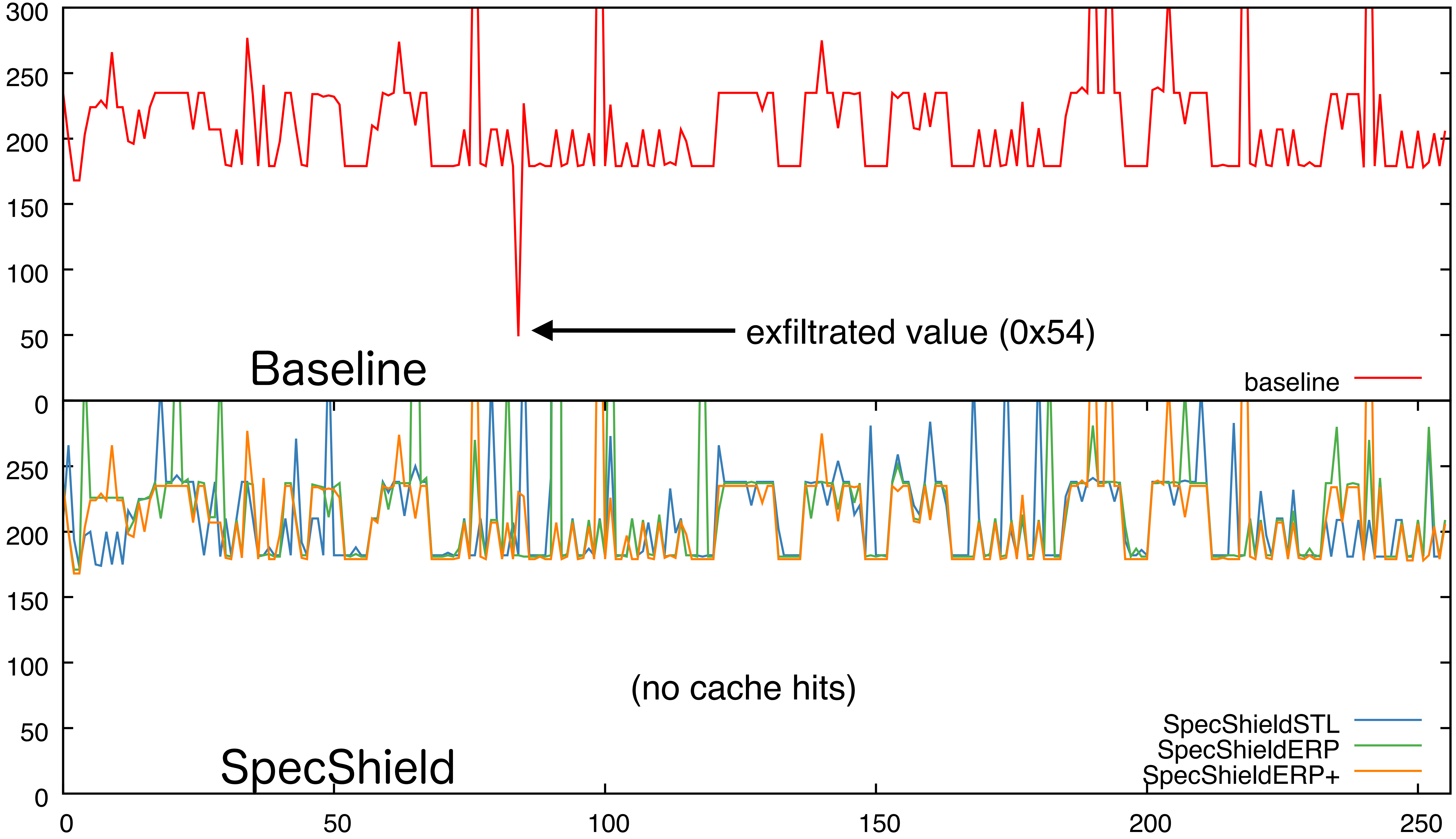


- Spectre-v1 attack, using cache as covert channel

```
if (x < array1_size)
  y = array2[array1[x] *  256];
```

- Exfiltrated value visible in access latency

Cache Latency for Spectre Attack

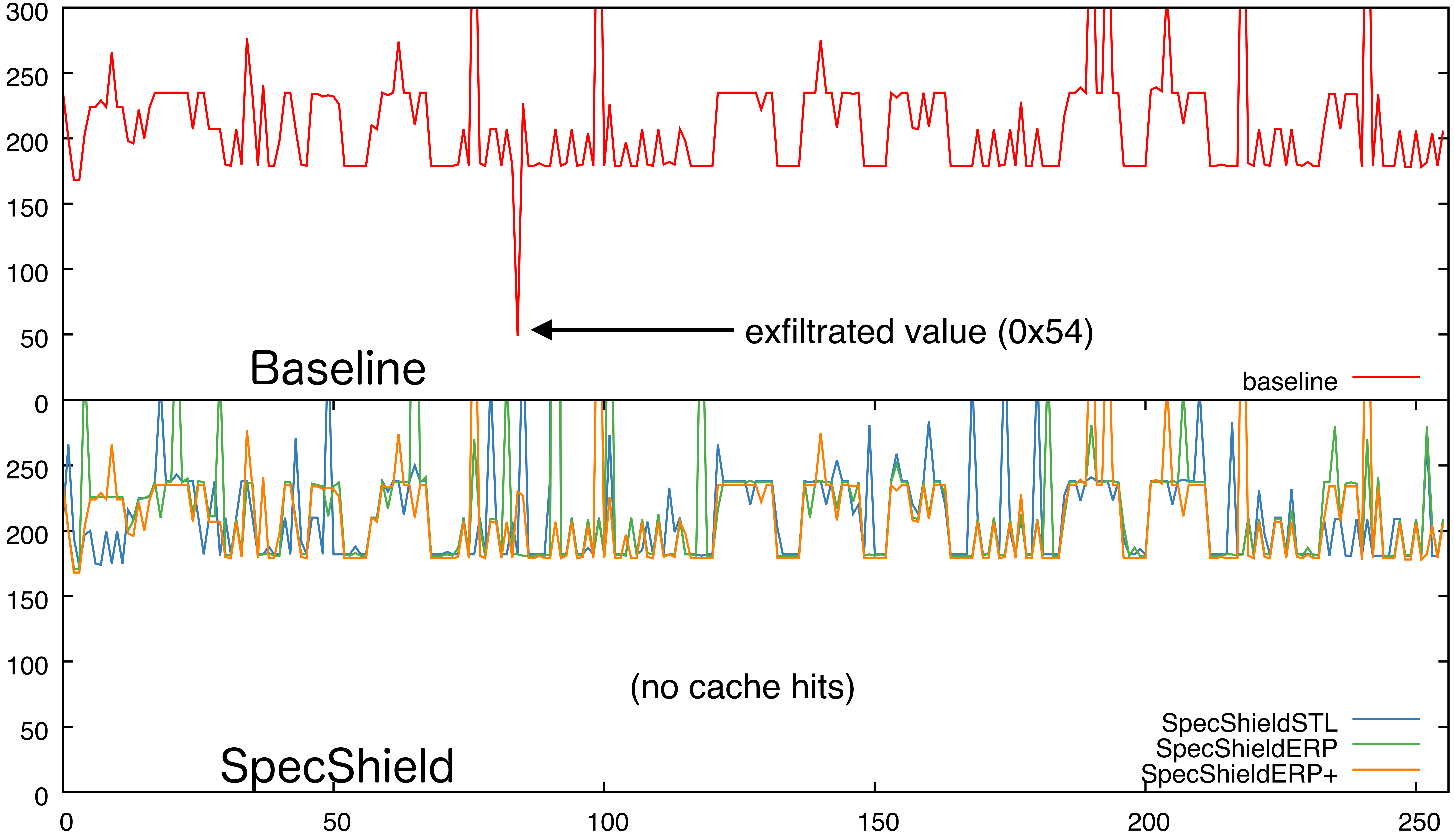


- Spectre-v1 attack, using cache as covert channel

```
if (x < array1_size)
   y = array2[array1[x] *  256];
```

- Exfiltrated value visible in access latency

Cache Latency for Spectre Attack



exfiltrated value (0x54)

Baseline

(no cache hits)

SpecShield

- Spectre-v1 attack, using cache as covert channel

```
if (x < array1_size)
  y = array2[array1[x] *  256];
```

- Exfiltrated value visible in access latency

- Secret value no longer appears in the cache channel

Conclusions

- Microarchitectural framework for preventing transient execution attacks on arbitrary memory
- SpecShield is more general
  - Unlike prior work that has focused on closing specific covert channels, SpecShield controls all speculative data-flow within the pipeline, preventing channel formation.
- SpecShield is easier to implement
  - No changes to the memory hierarchy, coherence protocol, consistency guarantees, etc.
- Performance-security tradeoff possible by only restricting select covert channels

Thank You!

Questions?

SpecShield STL: Implementation

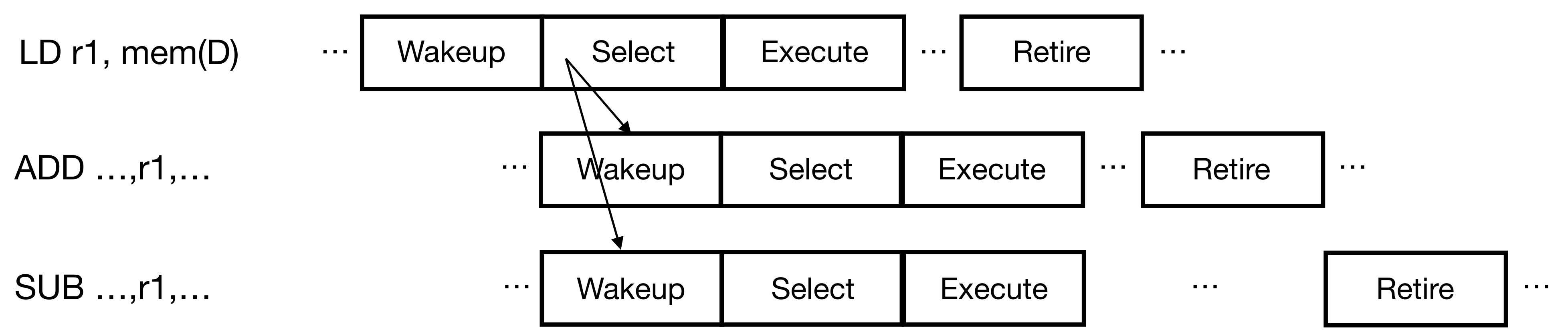
SpecShield STL: Implementation

- Impact on wakeup/select logic

SpecShield STL: Implementation

- Impact on wakeup/select logic
- Baseline: load dependents speculatively woken up on select

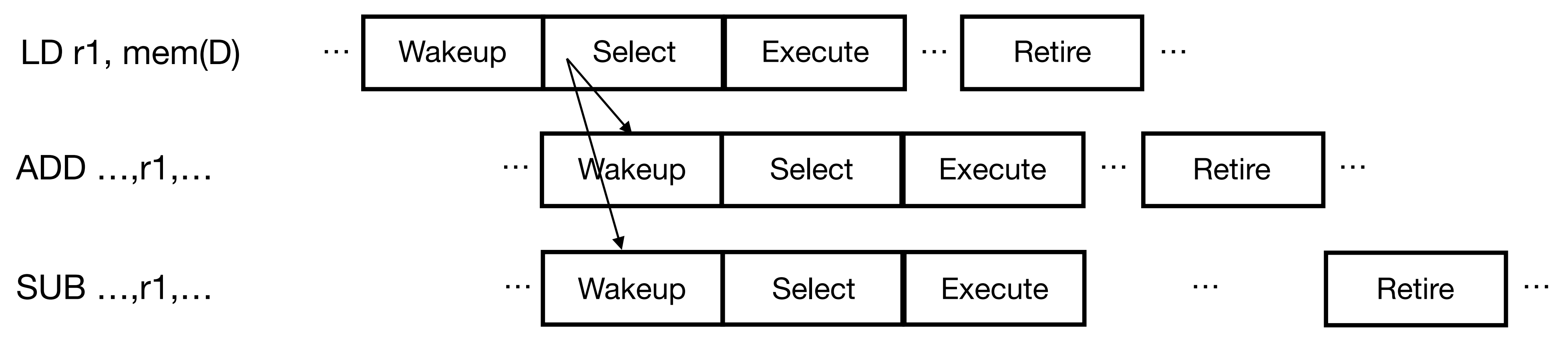
Baseline Wakeup/Select/Execute/Retire Pipeline


LD r1, mem(D)
… Wakeup Select Execute … Retire …
ADD …,r1,…
… Wakeup Select Execute … Retire …
SUB …,r1,…
… Wakeup Select Execute … Retire …

SpecShield STL: Implementation

- Impact on wakeup/select logic
- Baseline: load dependents speculatively woken up on select
  - Speculating that load will be hit

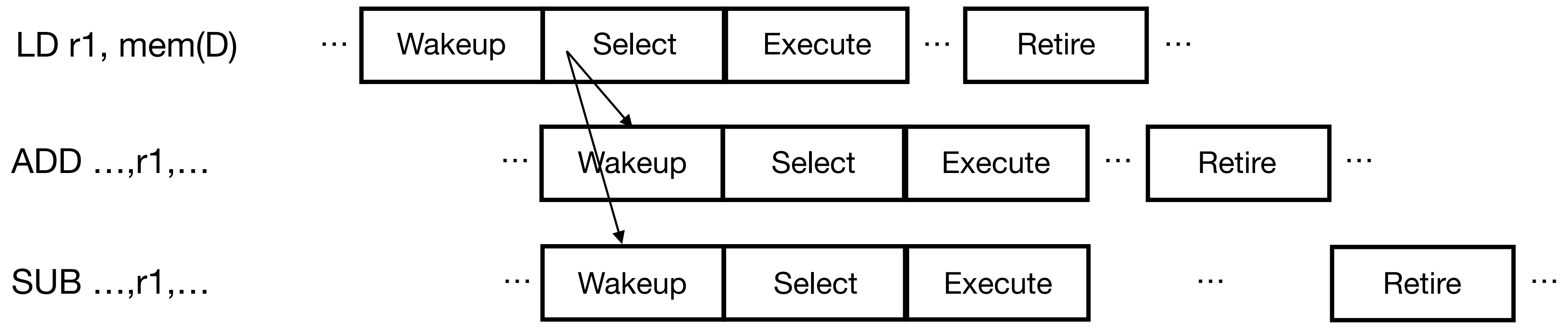
Baseline Wakeup/Select/Execute/Retire Pipeline



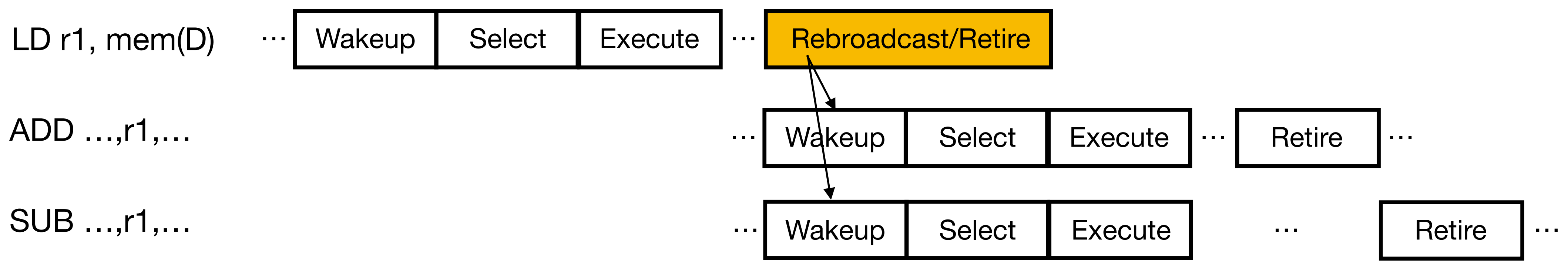
SpecShield STL: Implementation

- Impact on wakeup/select logic
- Baseline: load dependents speculatively woken up on select
  - Speculating that load will be hit
- SpecShield: Wakeup delayed until retirement

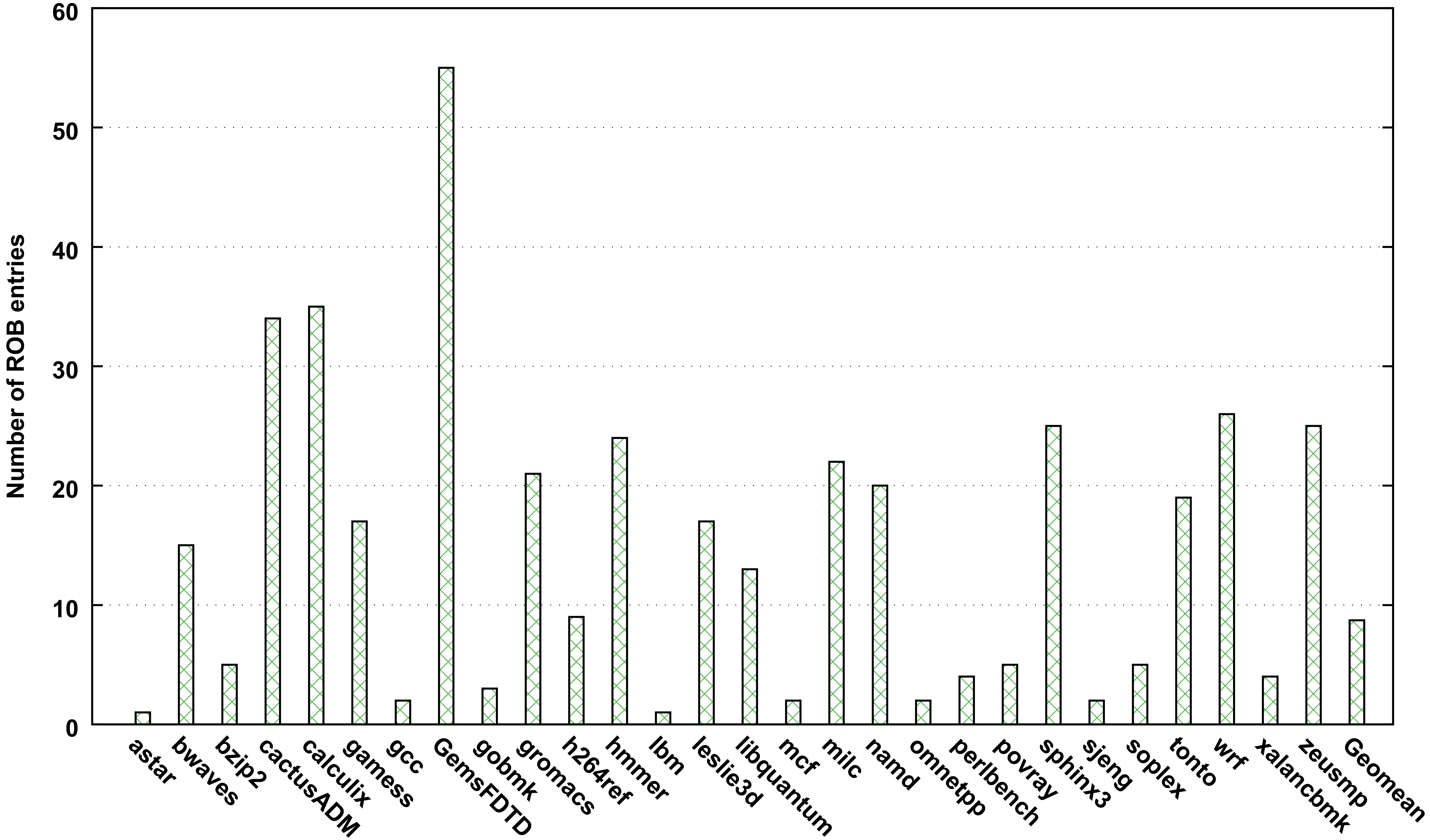
Baseline Wakeup/Select/Execute/Retire Pipeline


LD r1, mem(D)
… Wakeup Select Execute … Retire …
ADD …,r1,…
… Wakeup Select Execute … Retire …
SUB …,r1,…
… Wakeup Select Execute … Retire …

SpecShield Wakeup/Select/Execute/Retire Pipeline

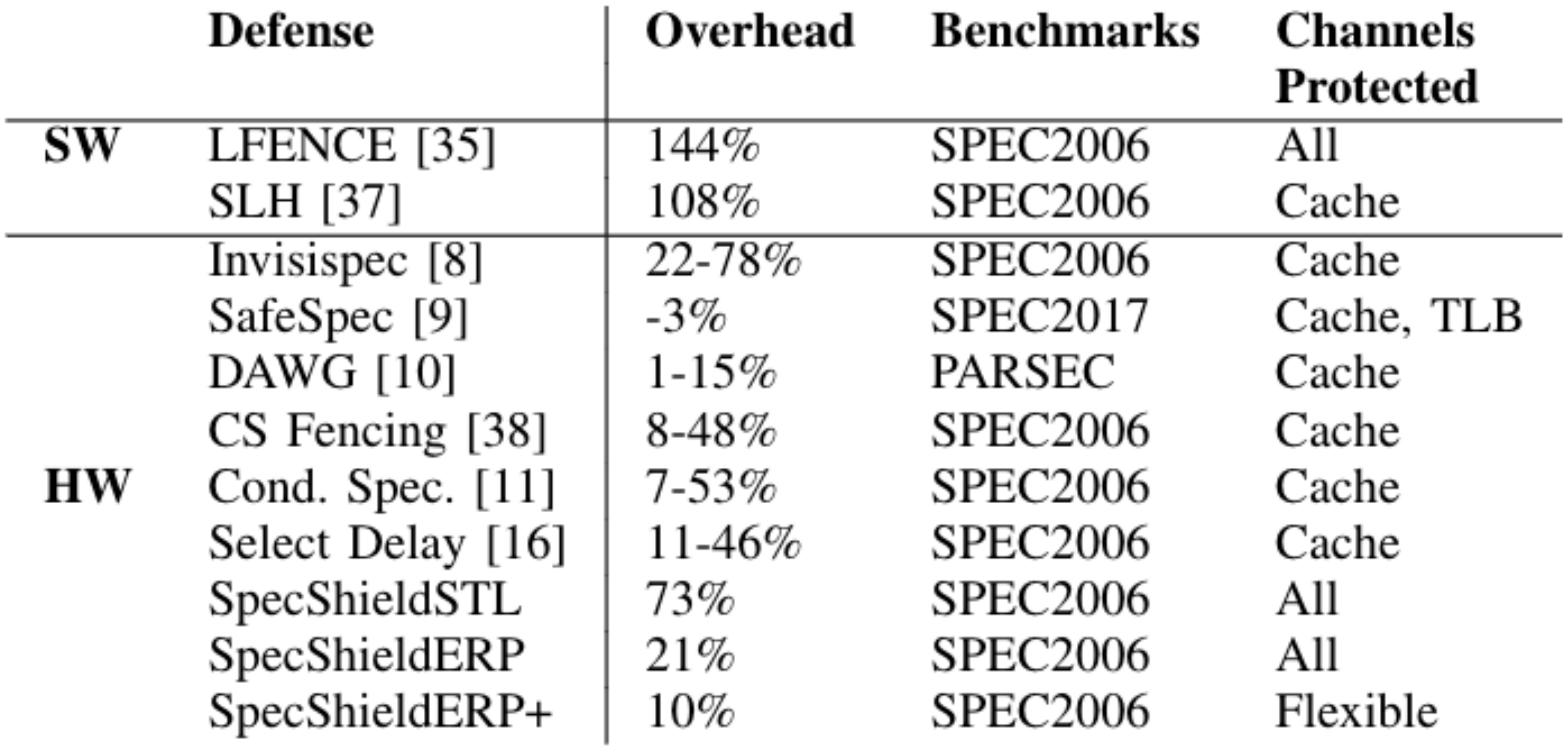

LD r1, mem(D)
… Wakeup Select Execute … Rebroadcast/Retire
ADD …,r1,…
… Wakeup Select Execute … Retire …
SUB …,r1,…
… Wakeup Select Execute … Retire …

Benefits of Early Resolution



- Average distance between ERP and ROB Head
- 1-55 entries, 9 average

Comparison with other solutions



	Defense	Overhead	Benchmarks	Channels Protected
SW	LFENCE [35]	144%	SPEC2006	All
	SLH [37]	108%	SPEC2006	Cache
HW	Invisispec [8]	22-78%	SPEC2006	Cache
	SafeSpec [9]	-3%	SPEC2017	Cache, TLB
	DAWG [10]	1-15%	PARSEC	Cache
	CS Fencing [38]	8-48%	SPEC2006	Cache
	Cond. Spec. [11]	7-53%	SPEC2006	Cache
	Select Delay [16]	11-46%	SPEC2006	Cache
	SpecShieldSTL	73%	SPEC2006	All
	SpecShieldERP	21%	SPEC2006	All
	SpecShieldERP+	10%	SPEC2006	Flexible