SHANGHAI, CHINA

ISCAS 2026

IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS

CVA6-CFI: A First Glance at RISC-V Control-Flow Integrity Extensions

Simone Manoni[*], Emanuele Parisi[‡], Riccardo Tedeschi[*], Davide Rossi[*†], Andrea Acquaviva[*], Andrea Bartolini[*]

[*]*University of Bologna, Italy*

[†]*Fondazione Chips-IT, Italy*

[‡]*Barcelona Supercomputing Center, Spain*

Introduction

- Software security is a first-class concern across the computing stack
  - Cloud & datacenters

Introduction

- Software security is a first-class concern across the computing stack
  - Cloud & datacenters
  - Desktop & mobile

Introduction

- Software security is a first-class concern across the computing stack
  - Cloud & datacenters
  - Desktop & mobile
  - Safety-critical CPS (autonomous vehicles, robotics, medical, smart infrastructure)

Introduction

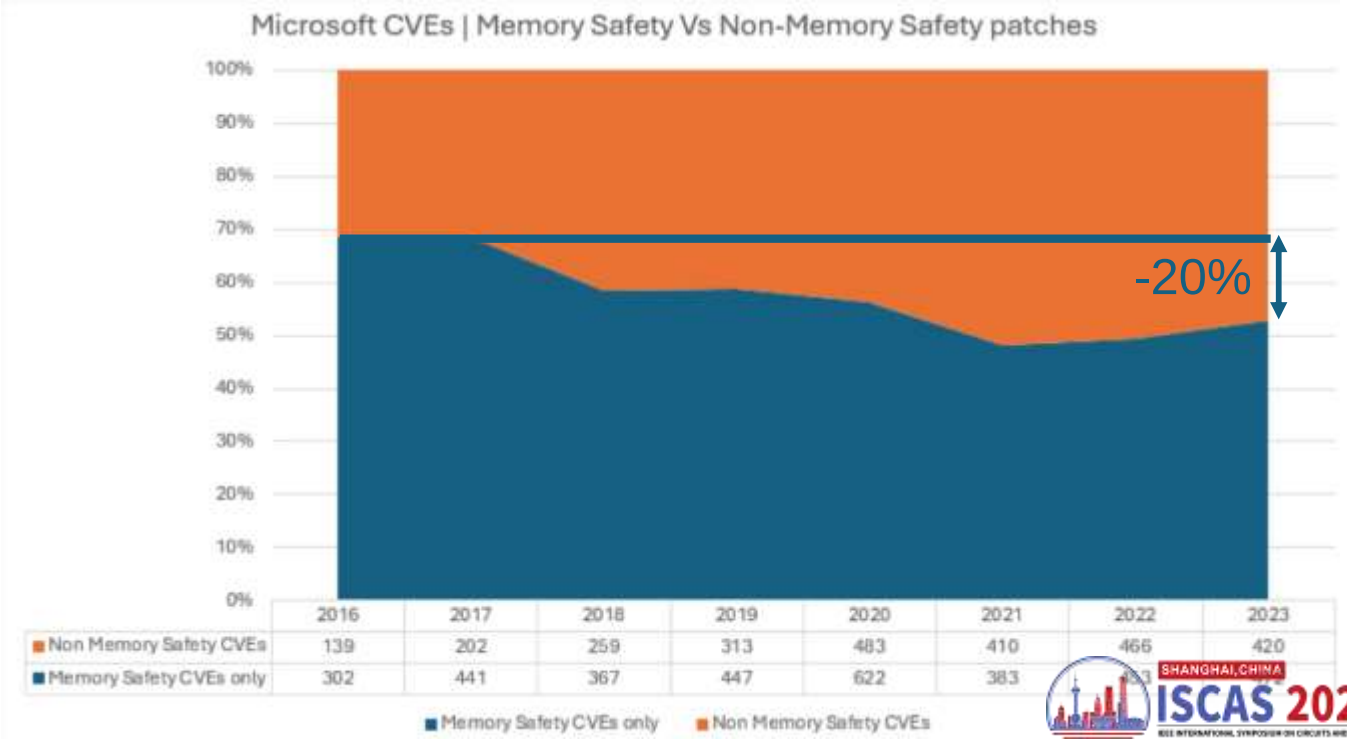
- Software security is a first-class concern across the computing stack
  - Cloud & datacenters
  - Desktop & mobile
  - Safety-critical CPS (autonomous vehicles, robotics, medical, smart infrastructure)

- A compromise can lead to
  - data leaks
  - service disruption
  - financial loss
  - physical harm

- Yet defenses must fit tight performance, memory, and area budgets

Background: Memory safety

- Memory-safety bugs are the dominant attack vector against C/C++ SoC firmware
- In 2016, 70% of Microsoft *Common Vulnerabilities & Exposures* (CVEs) were memory-safety related
- Despite improvements, ~50% of Microsoft patches in 2023 still address memory-safety issues
- These bugs enable **control-flow hijacking** (code-reuse attacks) [NSA/CISA, 2025]

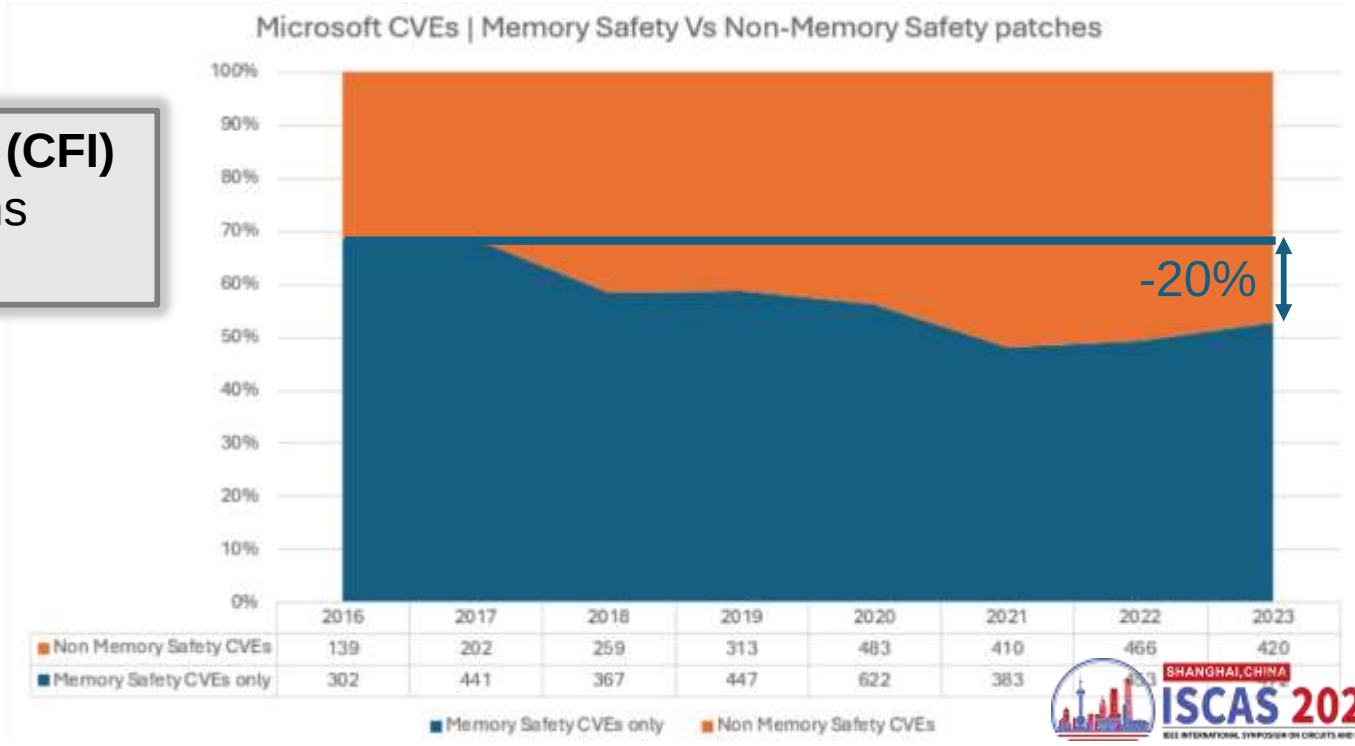


Source:"Memory Safe Languages: Reducing Vulnerabilities in Modern Software Development" *[NSA & CISA, June 2025]*

Background: Memory safety

- Memory-safety bugs are the dominant attack vector against C/C++ SoC firmware
- In 2016, 70% of Microsoft *Common Vulnerabilities & Exposures* (CVEs) were memory-safety related
- Despite improvements, ~50% of Microsoft patches in 2023 still address memory-safety issues
- These bugs enable **control-flow hijacking** (code-reuse attacks) [NSA/CISA, 2025]

State-of-the-art defense: Control-Flow Integrity (CFI)
- Constrains program execution to legitimate paths
- Blocks code-reuse attacks at their root



Source:"Memory Safe Languages: Reducing Vulnerabilities in Modern Software Development" *[NSA & CISA, June 2025]*

Background: CFI

- A program can be represented by a Control-Flow graph (CFG)
  - **Nodes** = Basic blocks
  - **Edges** = Possible Control transfers

- CFI restricts control transfers to **valid edges** in the CFG

```
A: addi  sp, sp, -32
   [...]
   addi  sp, sp, 32
   ret
```

```
B: addi    sp, sp, -32
   [...]
   addi    sp, sp, 32
   ret
```

```
C: addi    sp, sp, -48
   [...]
   jalr    a1
```

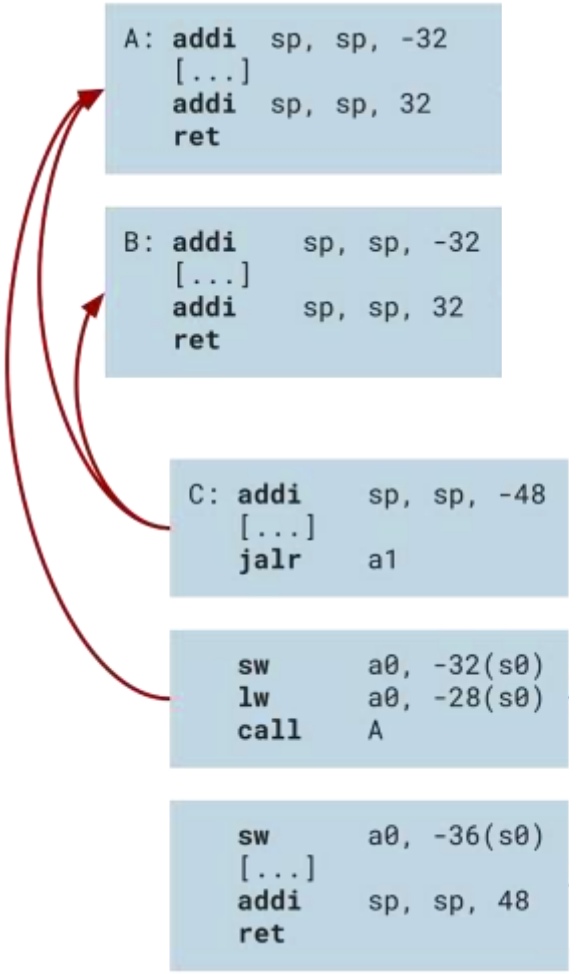
```
   sw      a0, -32(s0)
   lw      a0, -28(s0)
   call    A
```

```
   sw      a0, -36(s0)
   [...]
   addi    sp, sp, 48
   ret
```

Background: CFI

- A program can be represented by a Control-Flow graph (CFG)
  - **Nodes** = Basic blocks
  - **Edges** = Possible Control transfers

- CFI restricts control transfers to **valid edges** in the CFG
  - Backward edges:
    - Function `ret`
    - Possible protection mechanism: *Shadow Stack*
    - Store a copy of `ret` address in a protected memory region (s. stack)

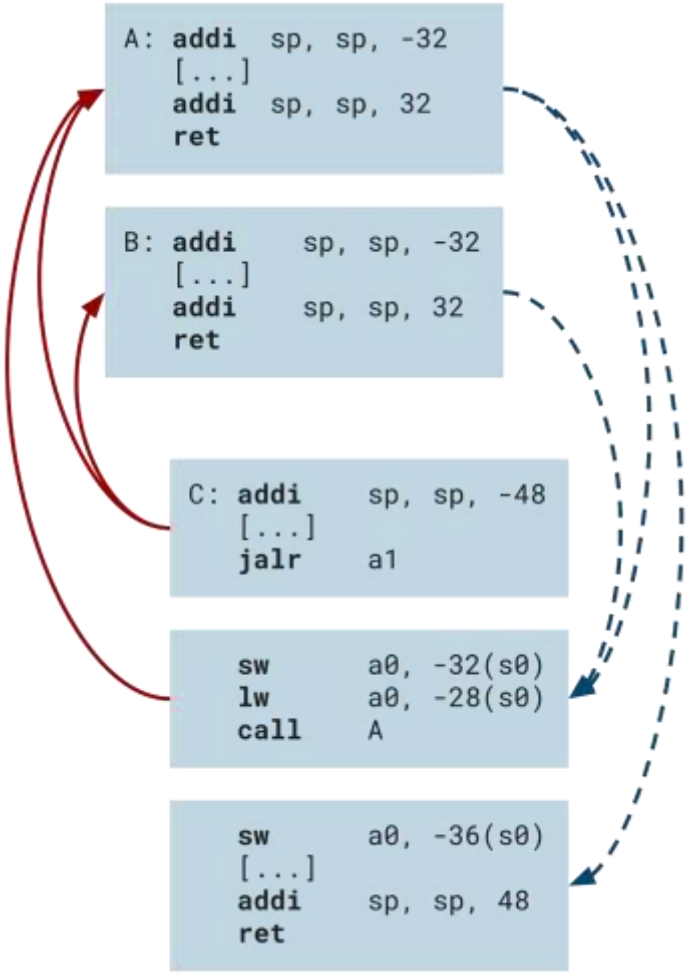


Background: CFI

- A program can be represented by a Control-Flow graph (CFG)
  - **Nodes** = Basic blocks
  - **Edges** = Possible Control transfers

- CFI restricts control transfers to **valid edges** in the CFG
  - Backward edges:
    - Function `ret`
    - Possible protection mechanism: *Shadow Stack*
    - Store a copy of `ret` address in a protected memory region (s. stack)

  - Forward edges:
    - Indirect calls and jumps
    - Possible protection mechanism: *Landing pads*
    - Mark valid targets in CFG



Background: CFI commercial solutions

- **Intel Control-flow Enforcement (CET)**
  - Shadow Stack – write protected memory region
  - Indirect Branch Tracking (IBT) – `ENDBRANCH` instr. marks valid indirect-branch targets (similar to LP)

Background: CFI commercial solutions

- **Intel Control-flow Enforcement (CET)**
  - Shadow Stack – write protected memory region
  - Indirect Branch Tracking (IBT) – `ENDBRANCH` instr. marks valid indirect-branch targets (similar to LP)

- **ARM**
  - Pointer Authentication (PAC) - Signs return addresses in pointer upper bits
  - Branch Target Instruction (BTI) - Branch Target Instructions mark valid forward-edge targets

Background: CFI commercial solutions

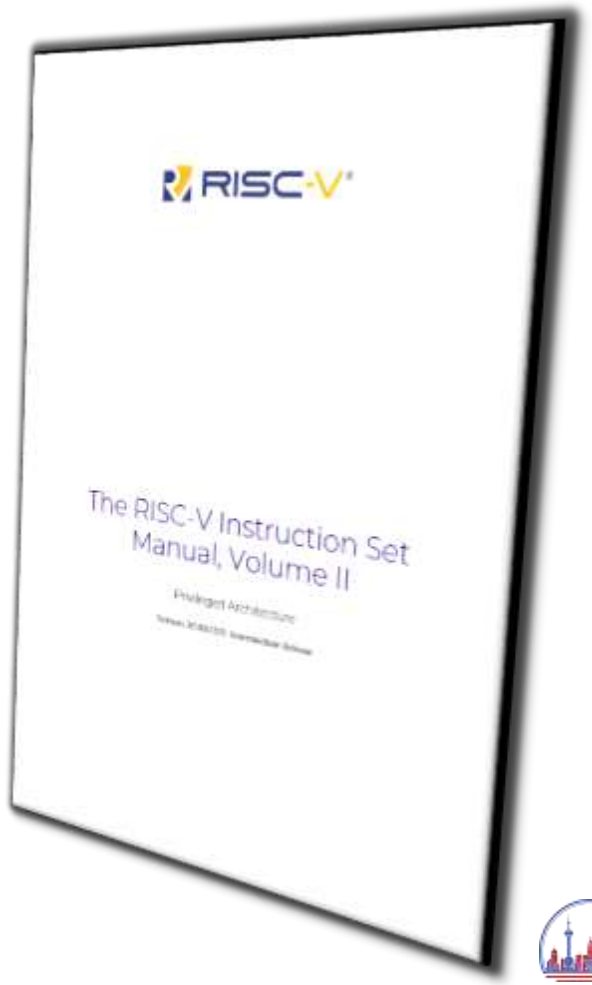
- **Intel Control-flow Enforcement (CET)**
  - Shadow Stack – write protected memory region
  - Indirect Branch Tracking (IBT) – `ENDBRANCH` instr. marks valid indirect-branch targets (similar to LP)

- **ARM**
  - Pointer Authentication (PAC) - Signs return addresses in pointer upper bits
  - Branch Target Instruction (BTI) - Branch Target Instructions mark valid forward-edge targets

- **IBM POWER10**
  - *hashst* / *hashchk* – cryptographic hash of return address stored alongside stack frame

Background: CFI in RISC-V

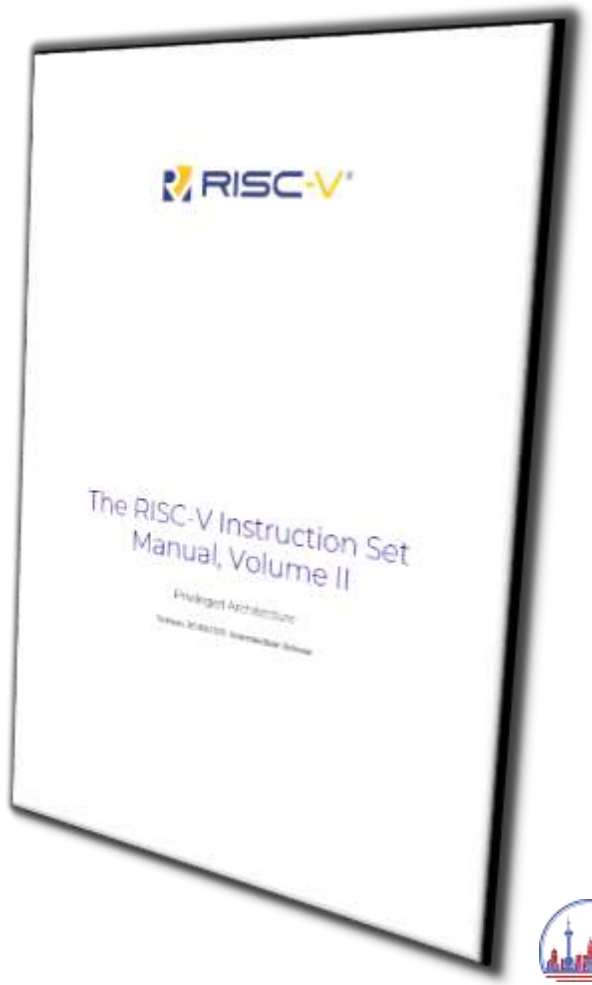
- RISC-V: open, royalty-free ISA, modular and extensible by design
- Increasingly adopted: >10 billion cores shipped since launch in 2015
- June 2024, Unprivileged ISA release
  - CFI ISA extensions ratified
    - *Zicfiss – Shadow Stack*
    - *Zicfilp – Landing Pads*



Background: CFI in RISC-V

- RISC-V: open, royalty-free ISA, modular and extensible by design
- Increasingly adopted: >10 billion cores shipped since launch in 2015
- June 2024, Unprivileged ISA release
  - CFI ISA extensions ratified
    - *Zicfiss – Shadow Stack*
    - *Zicfilp – Landing Pads*

What's missing
- Microarchitectural impact has never been evaluated
- No existing implementation of the CFI spec for RISC-V



Background: CFI in RISC-V

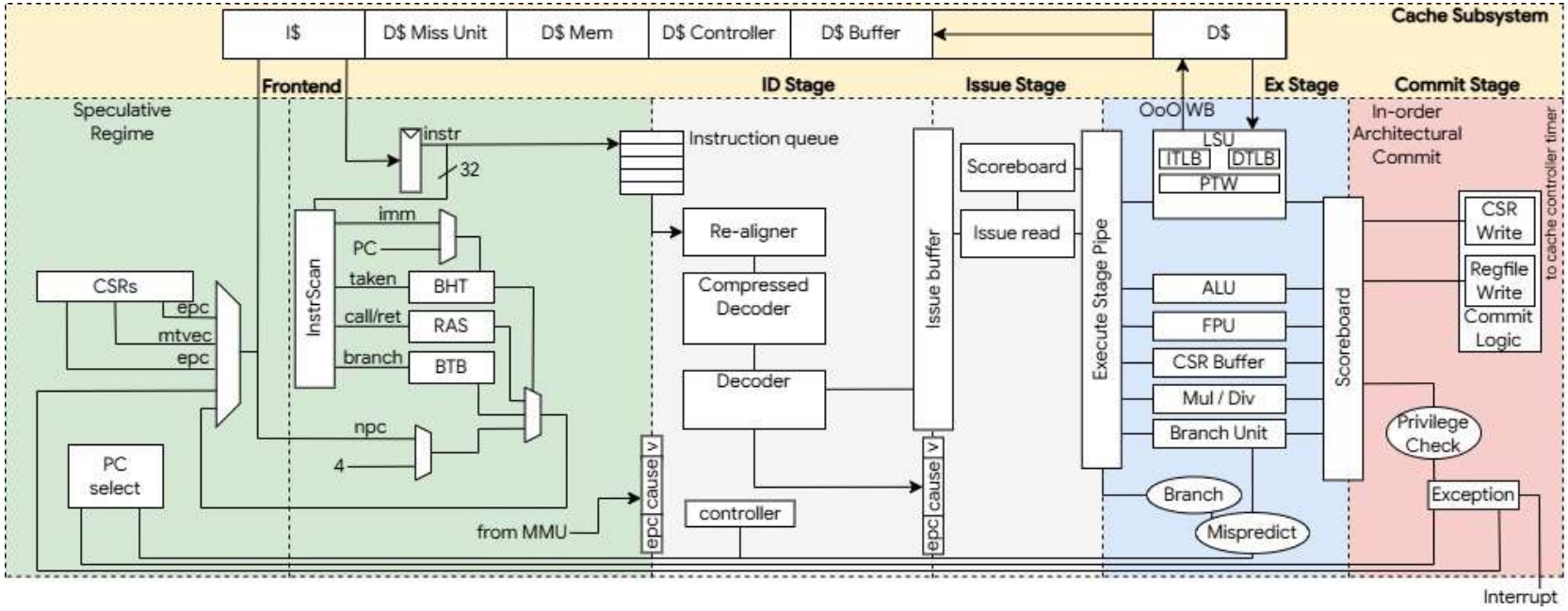
- RISC-V: open, royalty-free ISA, modular and extensible by design
- Increasingly adopted: >10 billion cores shipped since launch in 2015
- June 2024, Unprivileged ISA release
  - CFI ISA extensions ratified
    - *Zicfiss – Shadow Stack*
    - *Zicfilp – Landing Pads*

Our contribution:

- The first implementation and evaluation of the RISC-V CFI ISA extension on an application-class, open-source CPU

CVA6-CFI

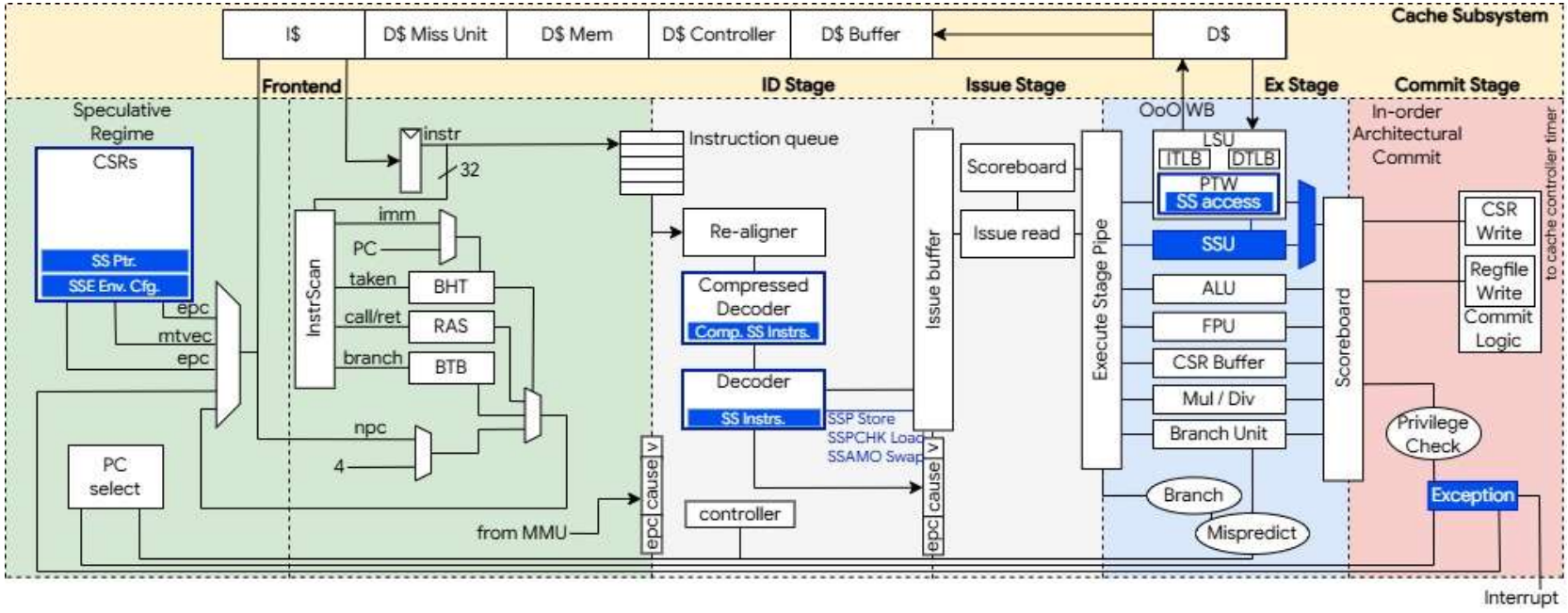
- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU



[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*

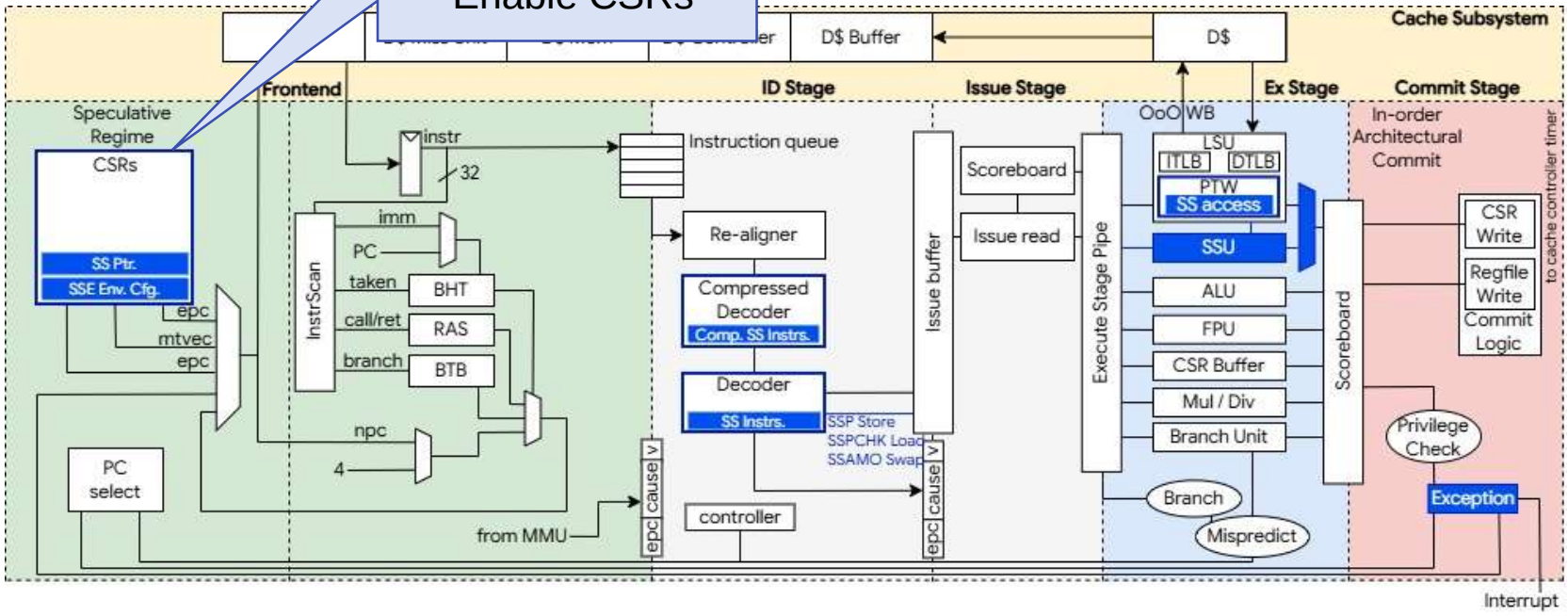


[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*

- **Shadow Stack Pointer** CSR
- Shadow Stack Enable CSRs

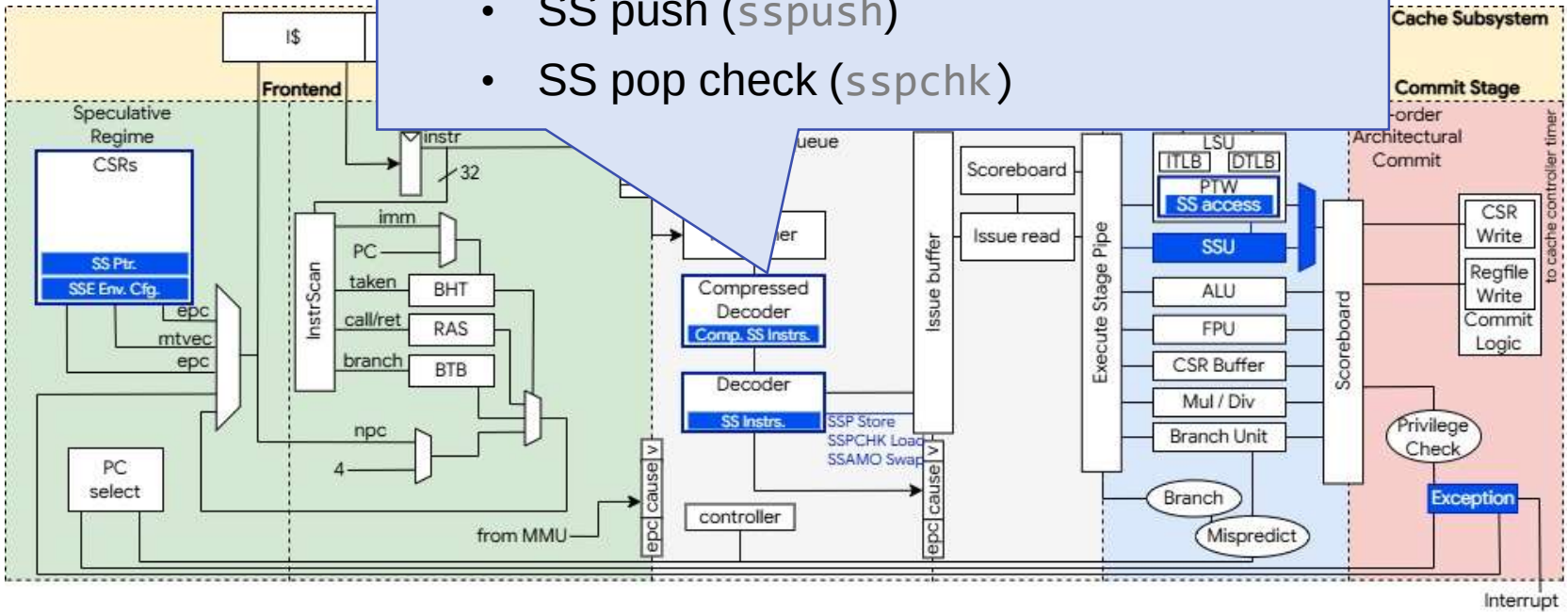


[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*

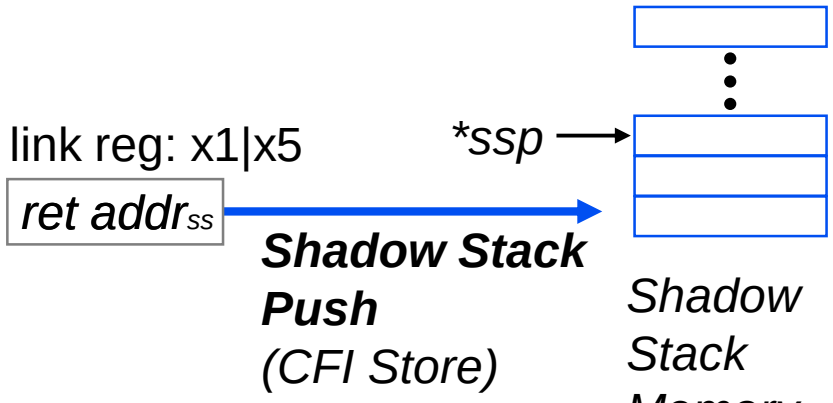
- SS pointer read (`ssrdp`)
- SS atomic swap (`ssamoswap`)
- SS push (`sspush`)
- SS pop check (`sspchk`)



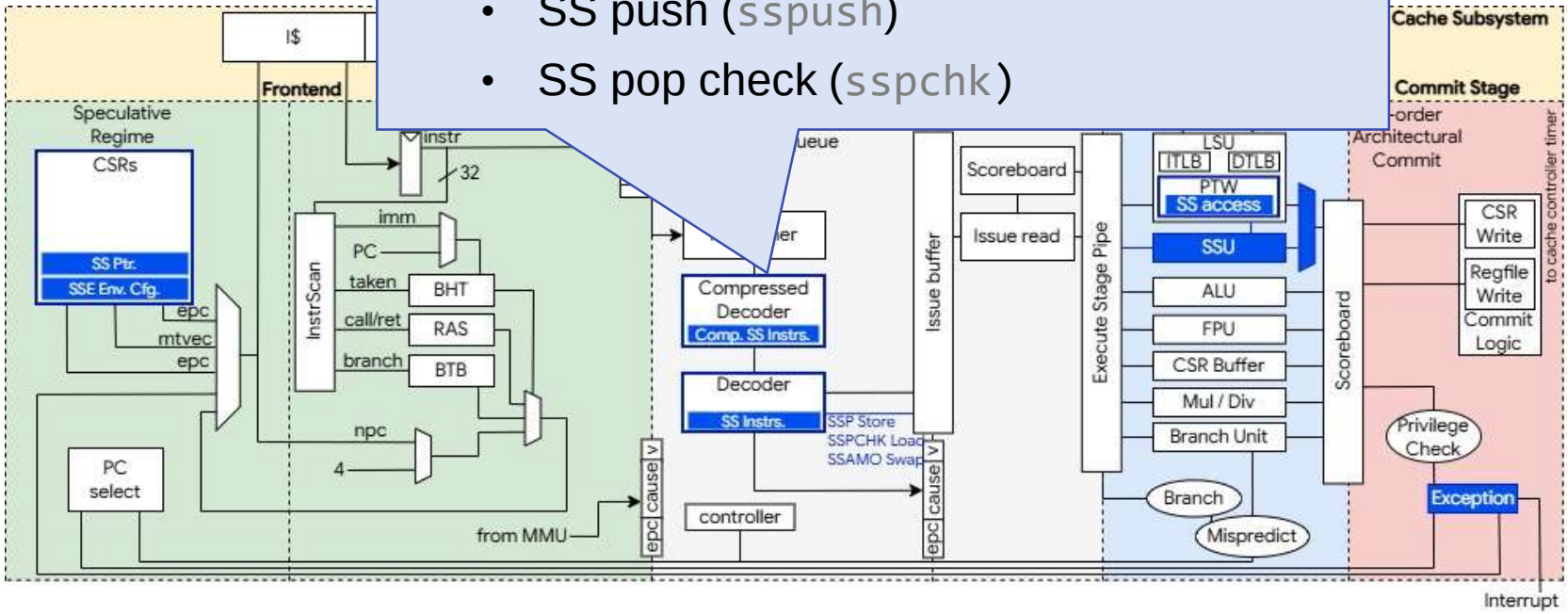
[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*



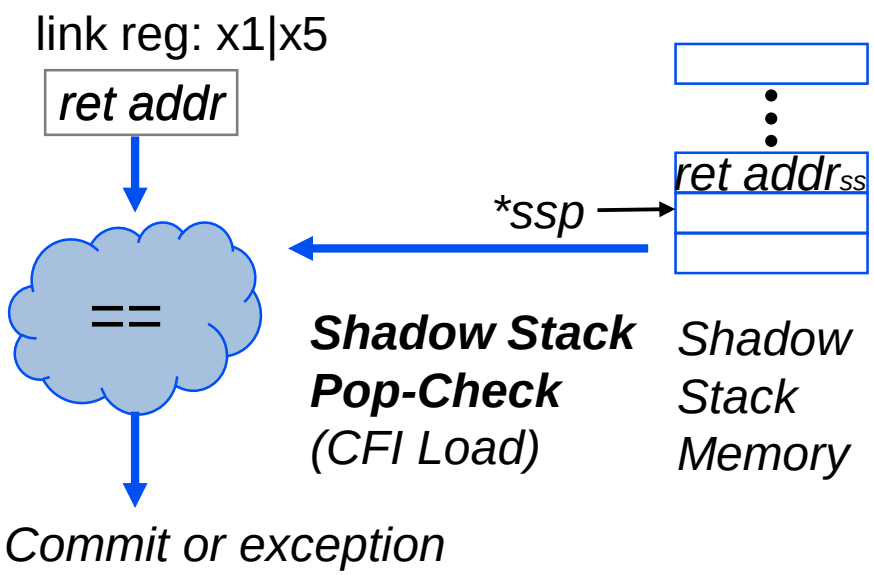
- SS pointer read (`ssrdp`)
- SS atomic swap (`ssamoswap`)
- SS push (`sspush`)
- SS pop check (`sspchk`)



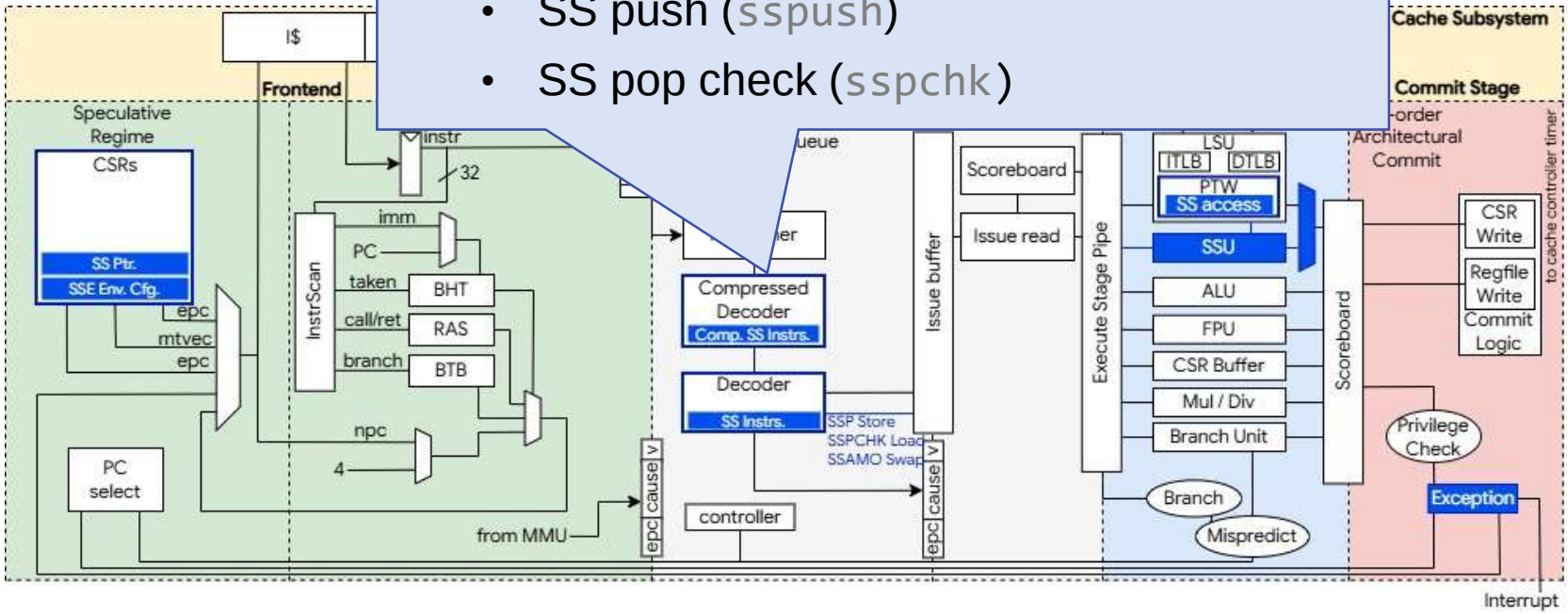
[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*



- SS pointer read (`ssrdp`)
- SS atomic swap (`ssamoswap`)
- SS push (`sspush`)
- SS pop check (`sspchk`)

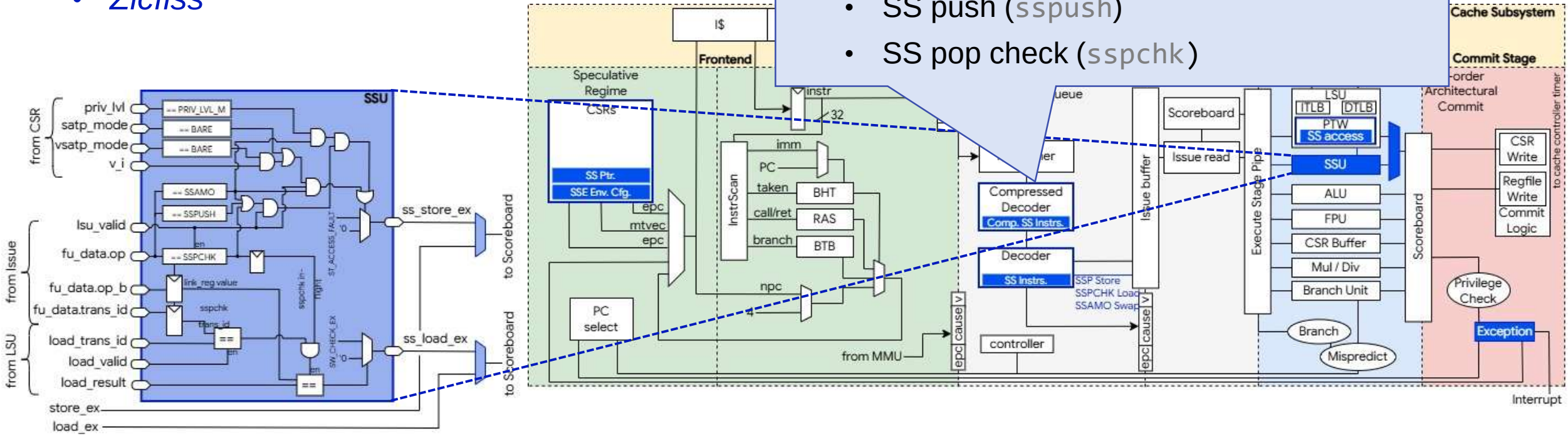


[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*

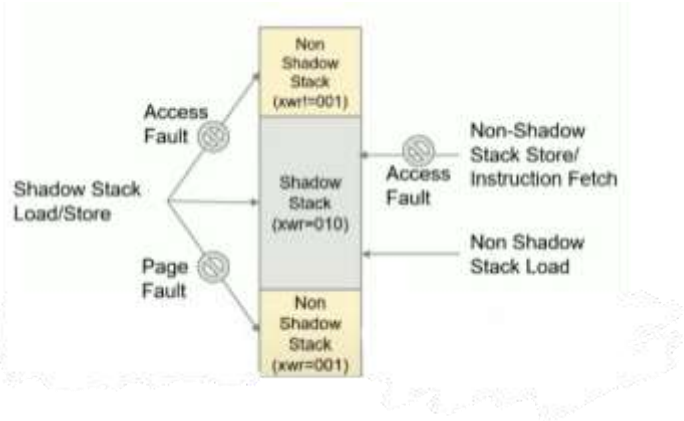
- SS pointer read (`ssrdp`)
- SS atomic swap (`ssamoswap`)
- SS push (`sspush`)
- SS pop check (`sspchk`)



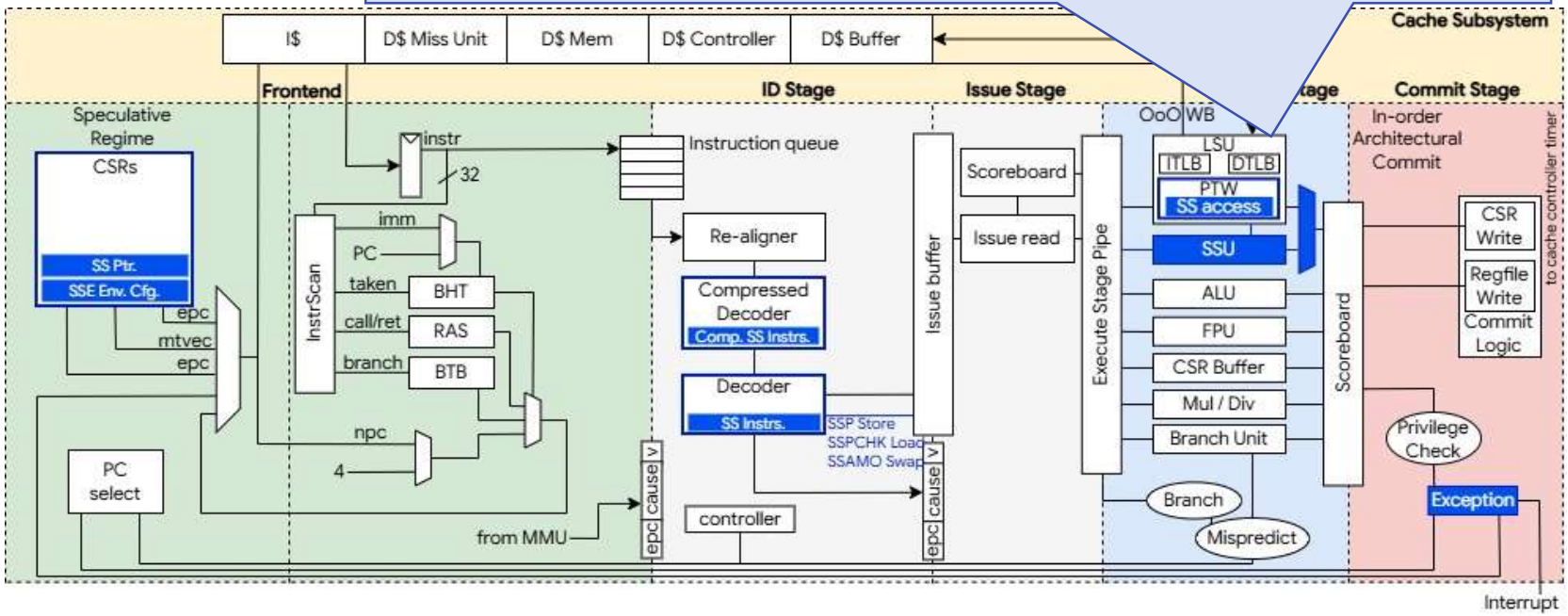
[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*



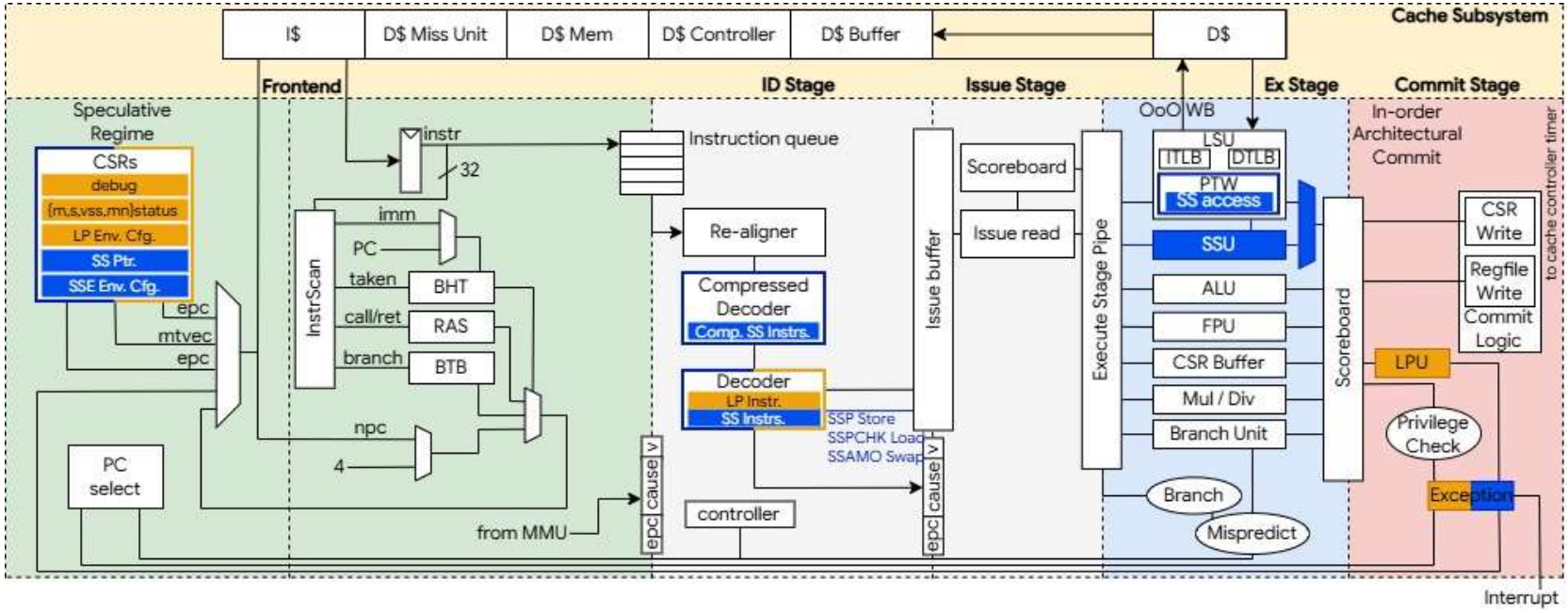
- Enforce checks on the type of accessed page
- Non-Shadow Stack instr. Only non Shadow-Stack pages and viceversa



[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*
  - *Zicfilp*

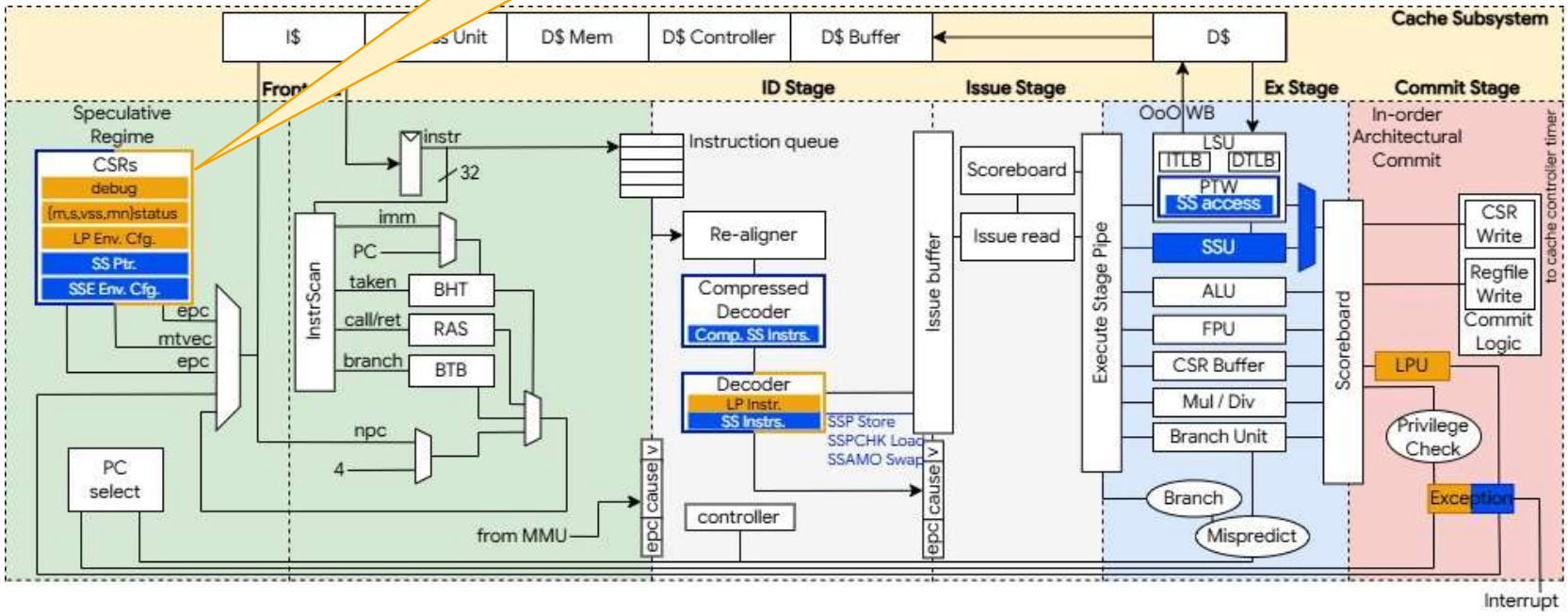


[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*
  - *Zicfilp*

- Landing Pad Enable CSRs



[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*
  - *Zicfilp*

- Landing pad instruction: `lp label`

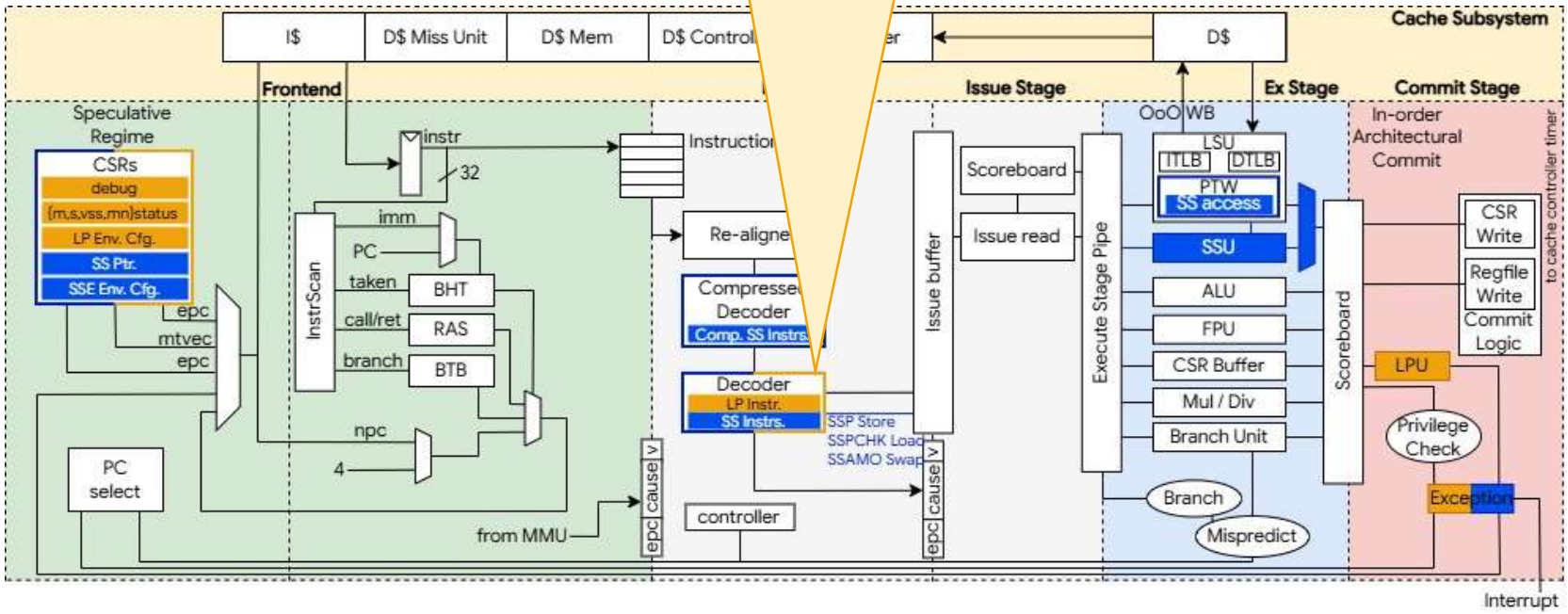
x7

label

==

auipc x7, %pcrel_hi(label)
addi x7, x7, %pcrel_lo(label)
jalr x1, xN, 0
addi sp, sp, -16

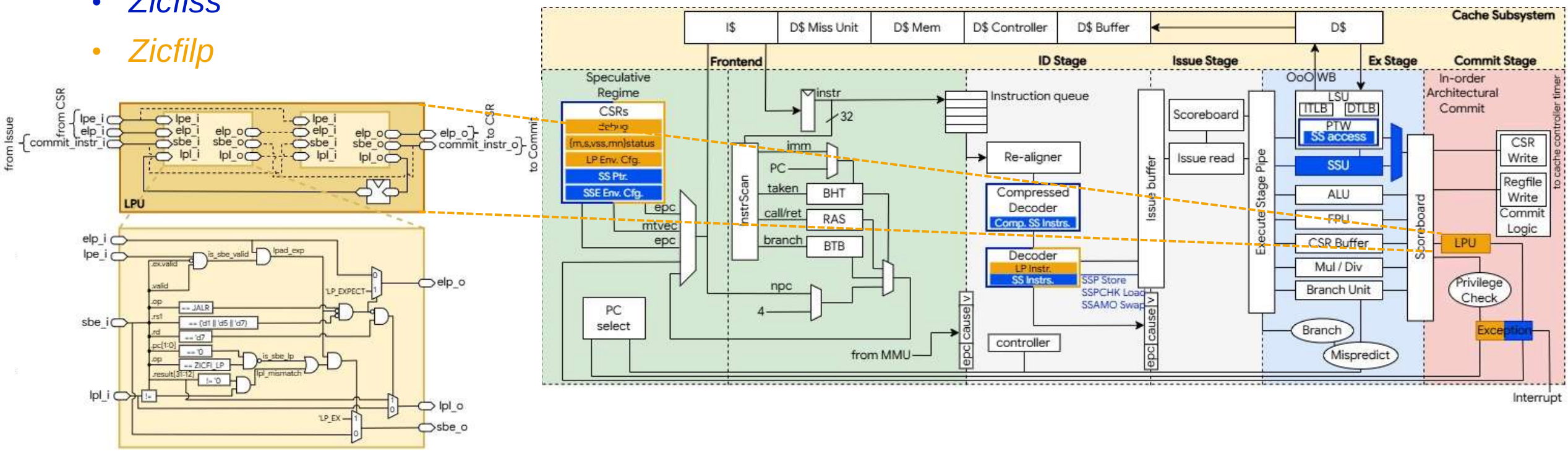
Exception



[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

CVA6-CFI

- Implementation & evaluation CFI ISA Extension on CVA6 [1]
  - Open-source
  - RV64, 6-stage, in-order, single-issue CPU
  - *Zicfiss*
  - *Zicfilp*



[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in *IEEE TVLSI 2019*

Results

- Area overhead
  - Synthesis in GF22FDX, @800 MHz
  - *Zicfiss* and *Zcfilp* introduce no degradation in operating frequency

CVA6 Pipeline	Area [μm^2]		Overhead [%]
	Baseline	w/ CFI ext.	
CSR Reg. File	7831	8220	5.0
Fetch & Decode	1101	1124	2.1
Issue	25637	25788	0.6
Execute	61631	61854	0.4
Commit	182	372	103.6
Total	96382	97358	1.0

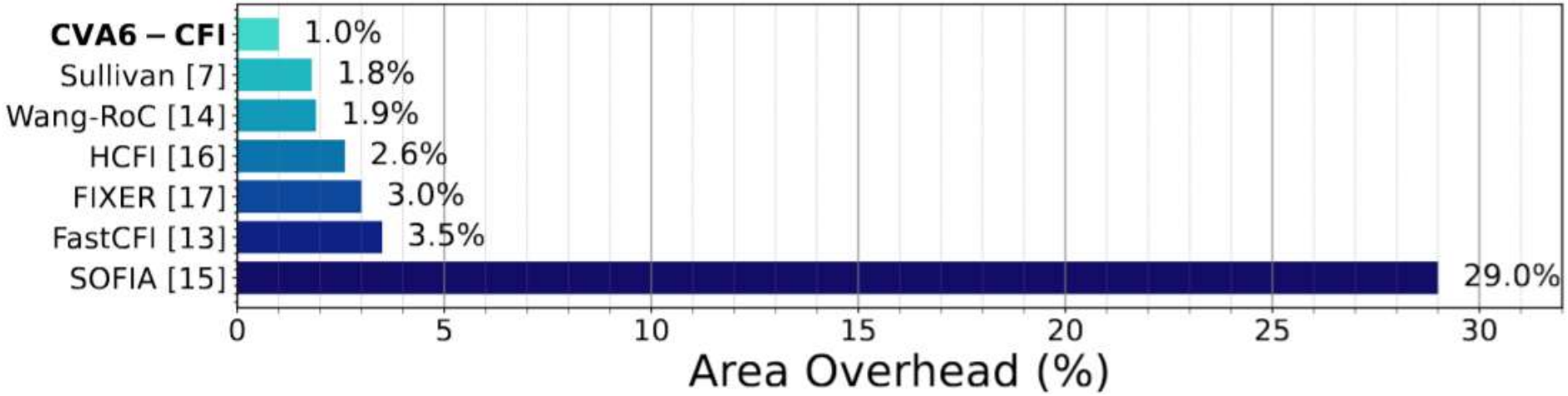
Results

- Area overhead
  - Synthesis in GF22FDX, @800 MHz
  - *Zicfiss* and *Zcfilp* introduce no degradation in operating frequency

CVA6 Pipeline	Area [μm^2]		Overhead [%]
	Baseline	w/ CFI ext.	
CSR Reg. File	7831	8220	5.0
Fetch & Decode	1101	1124	2.1
Issue	25637	25788	0.6
Execute	61631	61854	0.4
Commit	182	372	103.6
Total	96382	97358	1.0

Results

- Area overhead
  - We compare the area overhead of our proposed CFI extension with prior hardware-centric CFI approaches [7, 13–17]
  - Overhead values are normalized to their respective baseline architectures



[7] D. Sullivan, e al,, “Efficiently Enforcing Control-Flow Integrity with a Security Coprocessor,” in IEEE SecDev, 2016

[13] W. Wang, et al., “Hardware-assisted control-flow integrity enhancement for iot devices,” IEEE DATE 2024

[14] L. Feng et al., “Fastcfi: Real-time control-flow integrity using fpga without code instrumentation,” ACM TODAES 2021

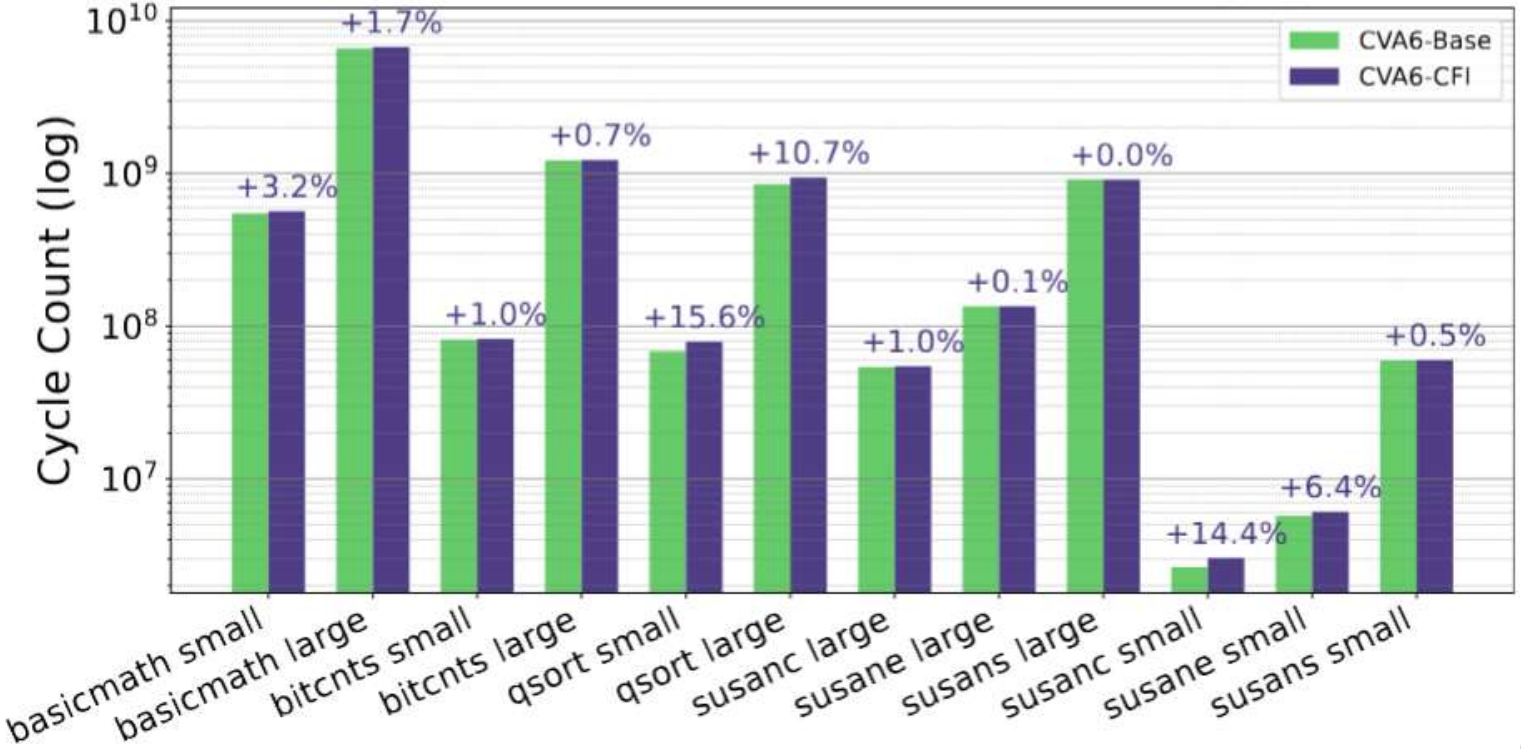
[15] R. D. Clercq, et al., “Sofia: Software and control flow integrity architecture,” IEEE CCS 2017

[16] N. Christoulakis, et al., “Hcfi: Hardware-enforced control-flow integrity,” ACM CODASPY 2016

[17] A. De, et al., “Fixer: Flow integrity extensions for embedded risc-v,” IEEE DATE 2019

Results

- Performance overhead
  - We evaluated the runtime overhead using *MiBench Automotive benchmark* subset
  - Compiled w/ CFI-aware SiFive GCC 13.3 [1]
  - The CFI extension introduces a modest slowdown (up to 15.6%)
  - Overhead is mainly due to shadow stack operations (`sspush`, `sspchk`)



[1] https://github.com/sifive/riscv-gnu-toolchain/tree/cfi-dev

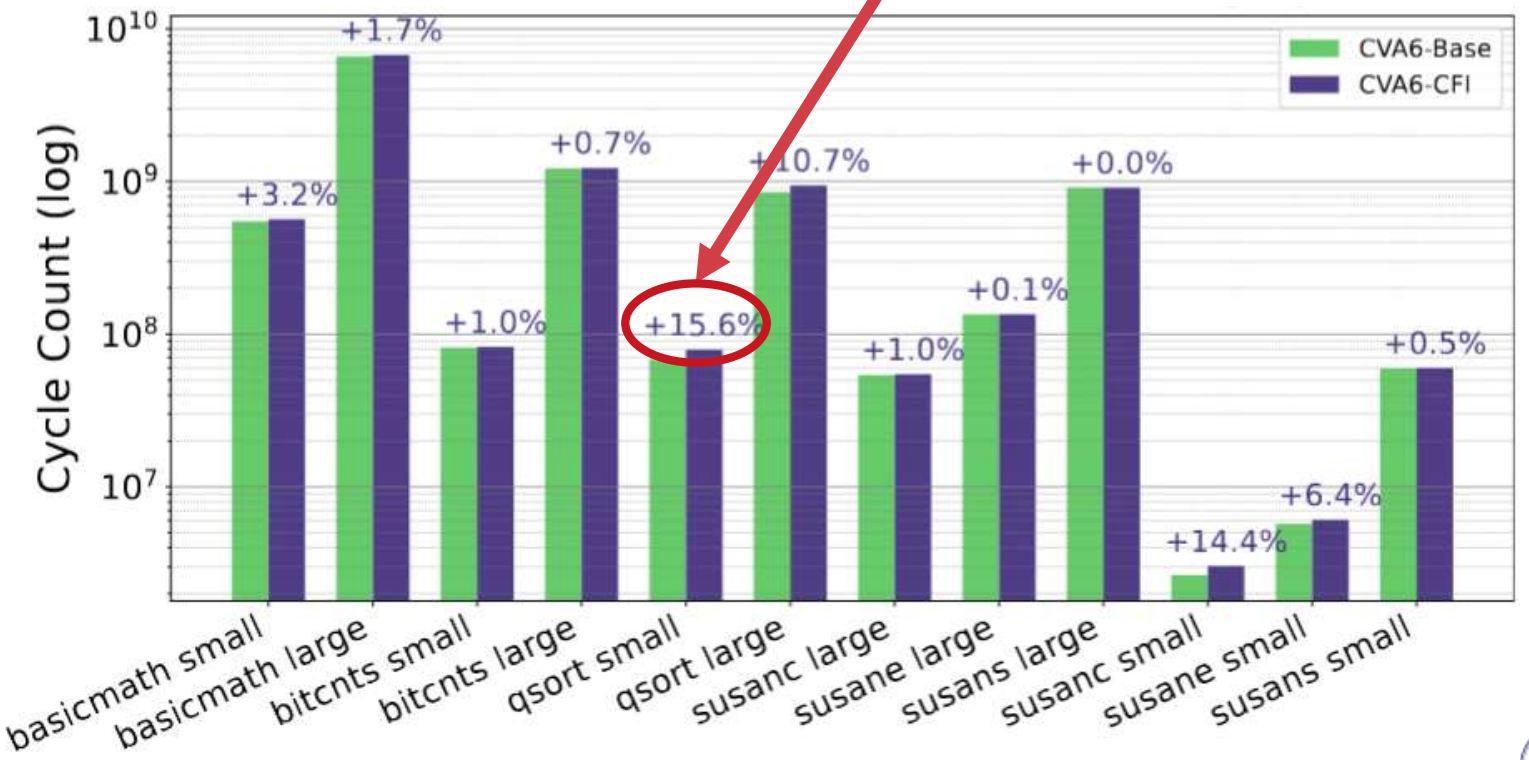
Results

- Performance overhead
  - We evaluated the runtime overhead using *MiBench Automotive benchmark* subset
  - Compiled w/ CFI-aware SiFive GCC 13.3 [1]
  - The CFI extension introduces a modest slowdown (up to 15.6%)
  - Overhead is mainly due to shadow stack operations (`ssp`**`u`**`sh`, `sspchk`)

[1] https://github.com/sifive/riscv-gnu-toolchain/tree/cfi-dev

Results

- Performance overhead
  - We evaluated the runtime overhead using *MiBench Automotive benchmark* subset
  - Compiled w/ CFI-aware SiFive GCC 13.3 [1]
  - The CFI extension introduces a modest slowdown (up to 15.6%)
  - Overhead is mainly due to shadow stack operations (`ssp`**`u`**`sh`, `sspchk`)
  - Open-Source! [2]



[1] https://github.com/sifive/riscv-gnu-toolchain/tree/cfi-dev
[2] https://github.com/AlSaqr-platform/cva6/tree/pulp-v1-culsans

Thank you!

Questions?