



Selective speculation: Exploring speculation barriers for Spectre attacks mitigation

Results in collaboration with :

- Ronan LASHERMES - LHS & SED
- Simon ROKICKI - TARAN
- Joseph PATUREL - TARAN

Herinomena ANDRIANATREHINA - TARAN
Thomas RUBIANO - PACAP

INRIA

Plan



PREREQUISITE

OVERVIEW

SPECULATION

SPECTRE

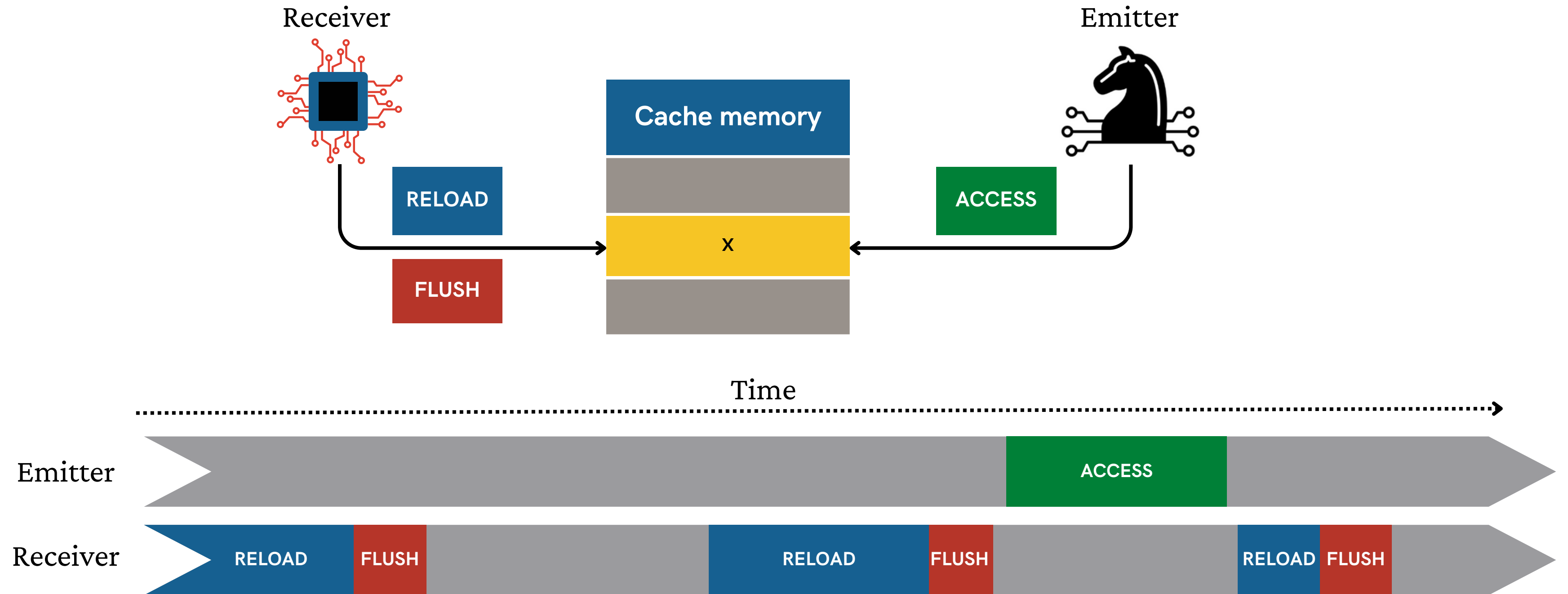
MITIGATIONS

SELECTIVE SPECULATION

RESULTS

PREREQUISITE

Covert-channel attacks - Flush/Reload:



Any microarchitectural state can be exploited as covert channels

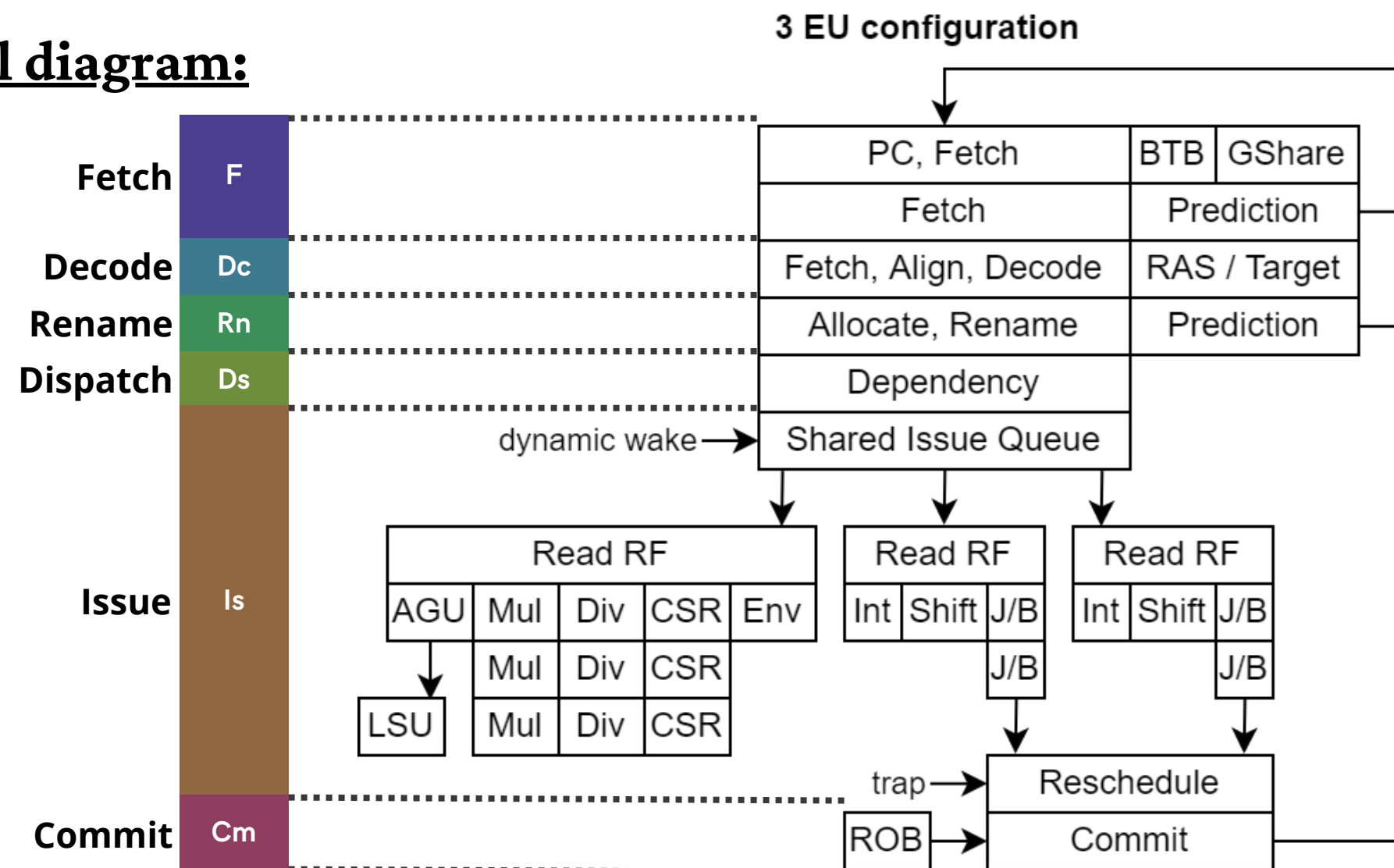
OVERVIEW

Objectif: ensuring confidentiality in modern Out-of-Order cores.

Target core: NaxRiscV

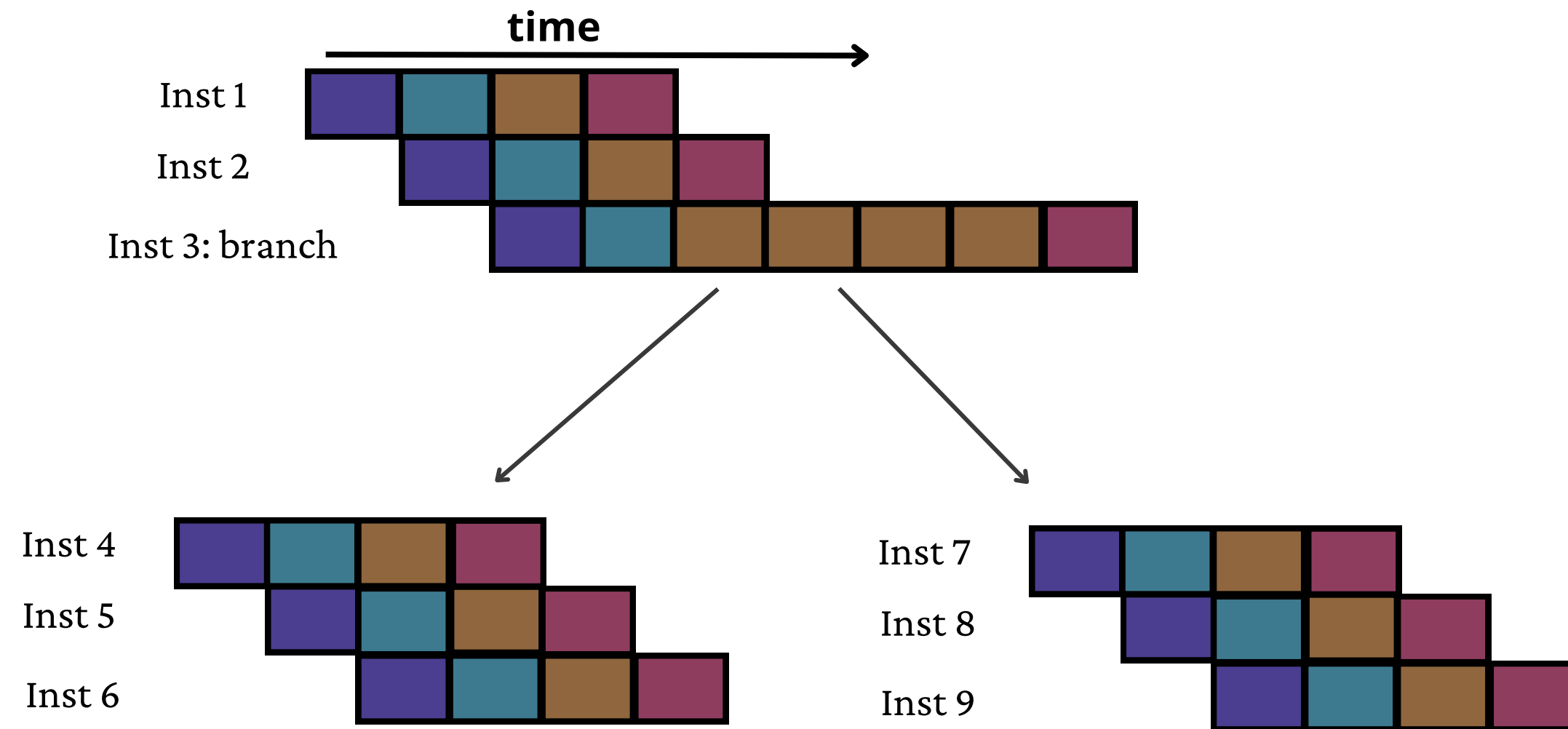
Core characteristics: Out of order and speculative execution

General architectural diagram:

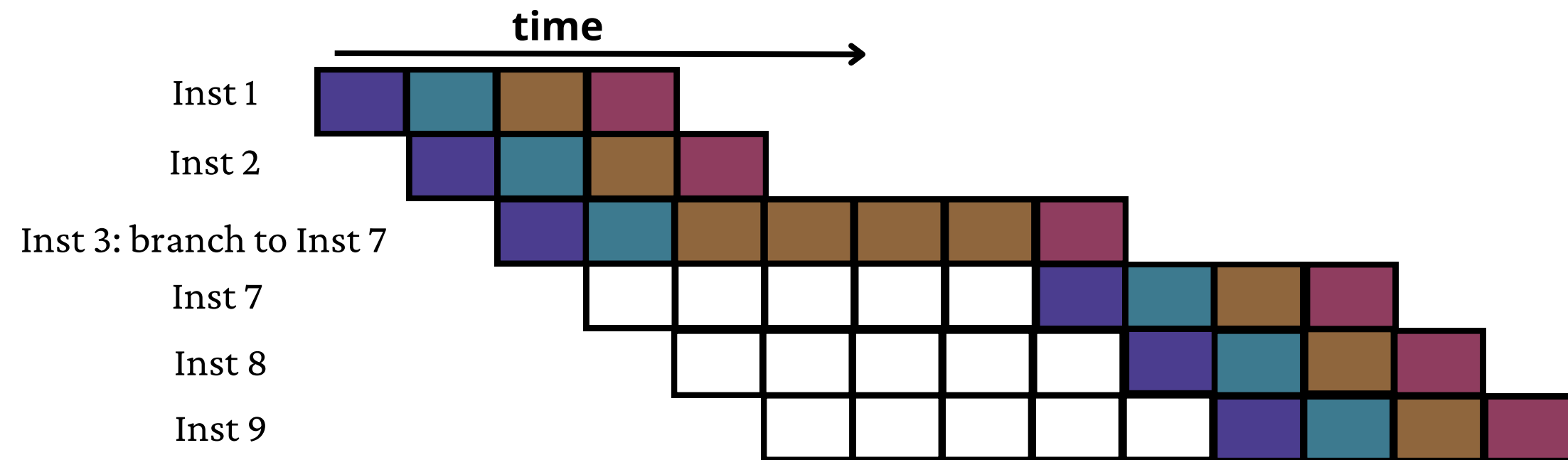


link : <https://spinalhdl.github.io/NaxRiscv-Rtd/main/index.html>

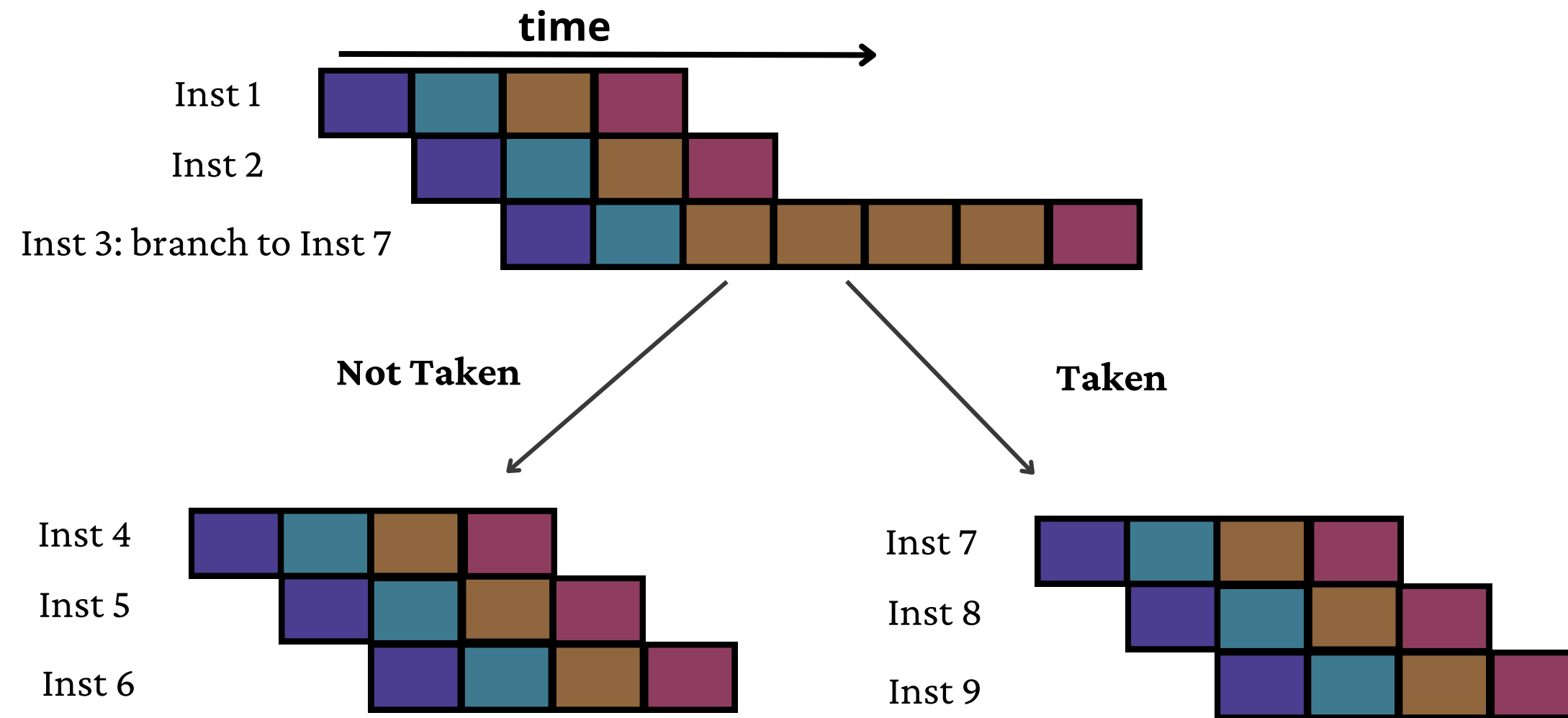
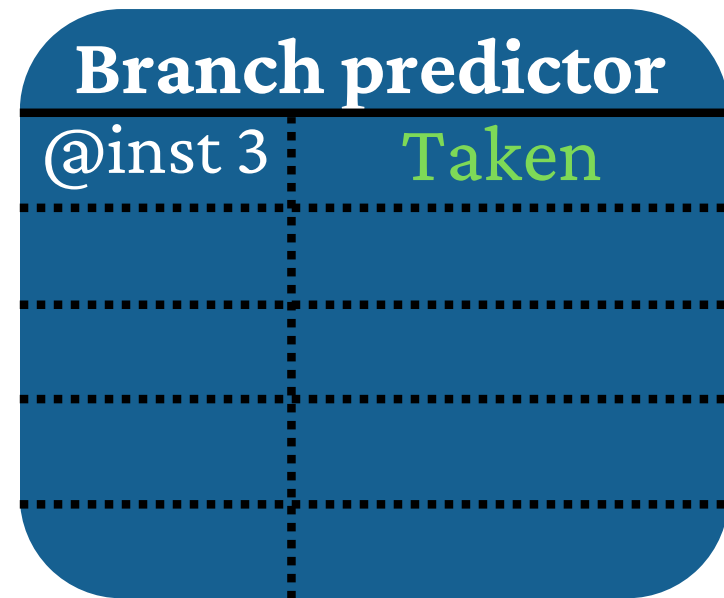
SPECULATION



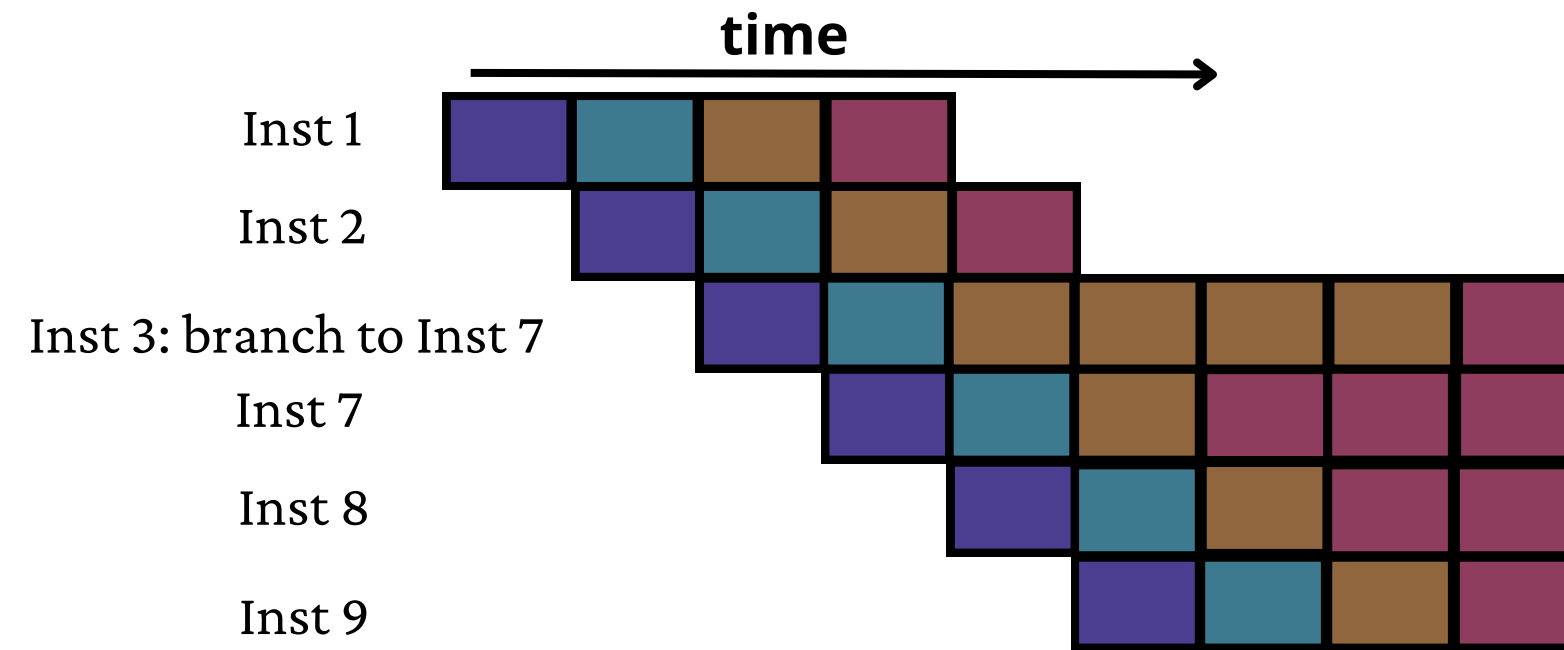
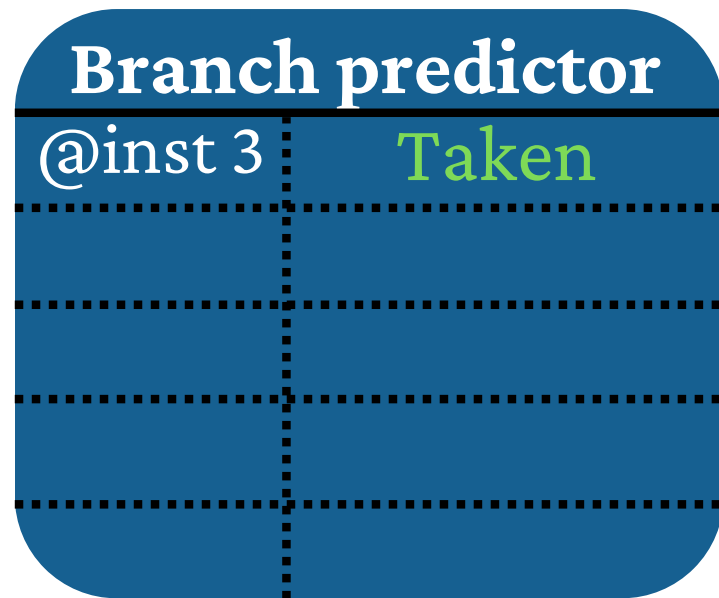
SPECULATION



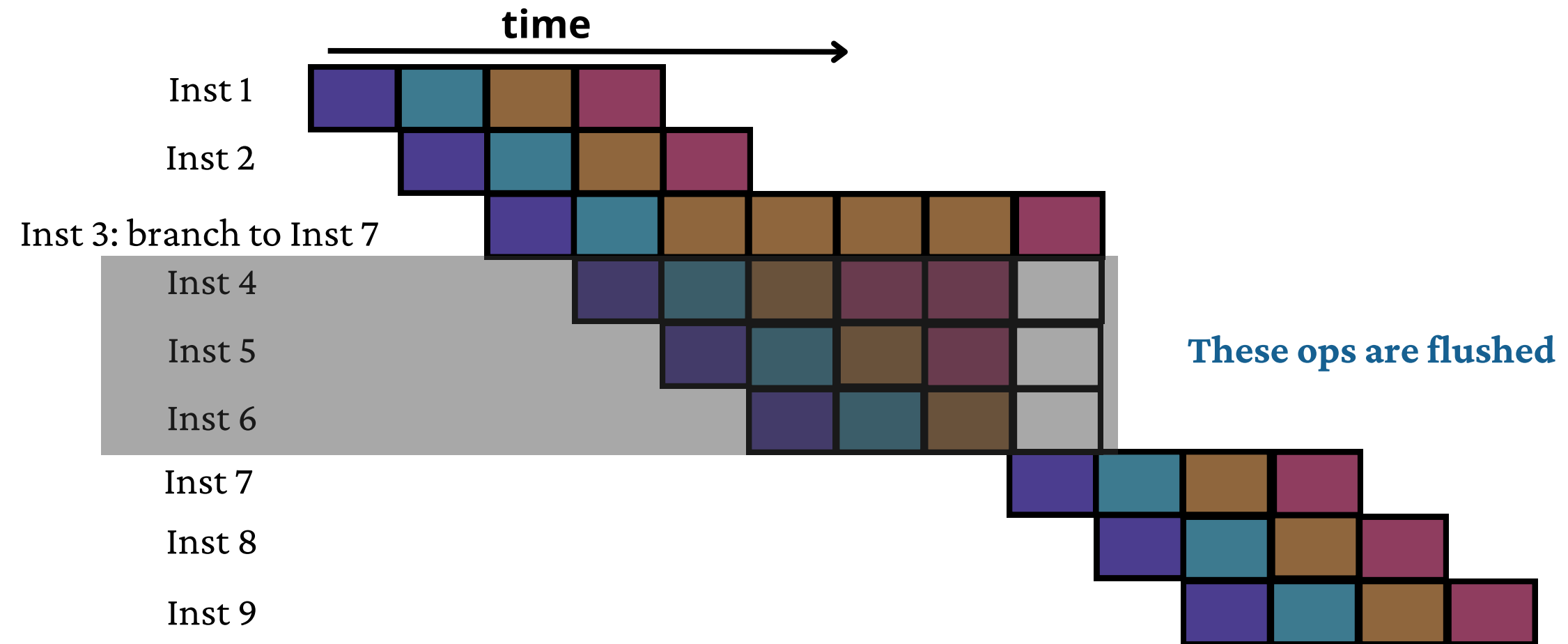
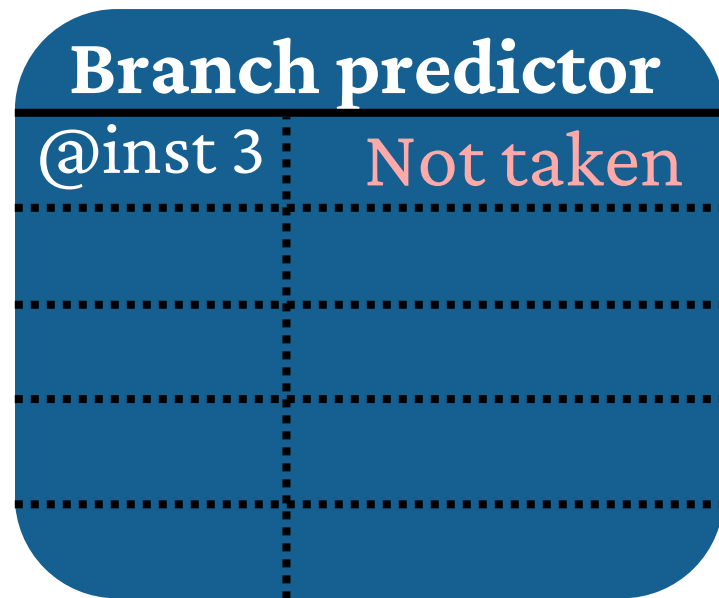
SPECULATION



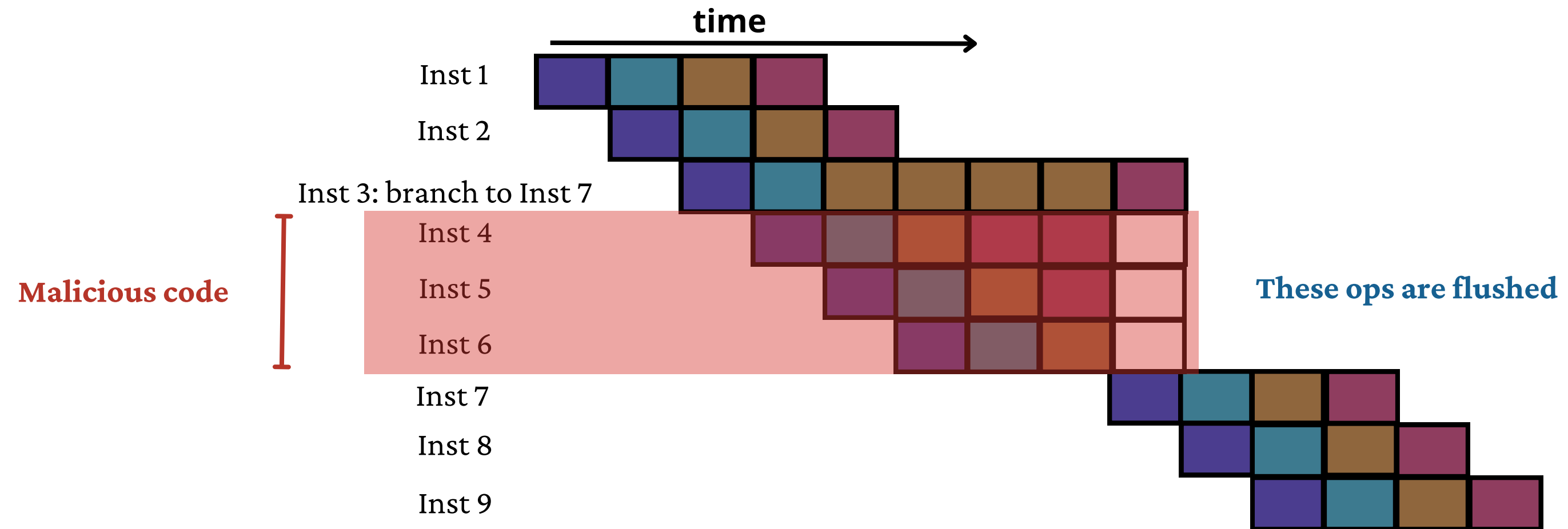
SPECULATION



SPECULATION



TRANSIENT EXECUTION VULNERABILITY



PREREQUISITE

OVERVIEW

ENHANCING EXECUTION
SPEED

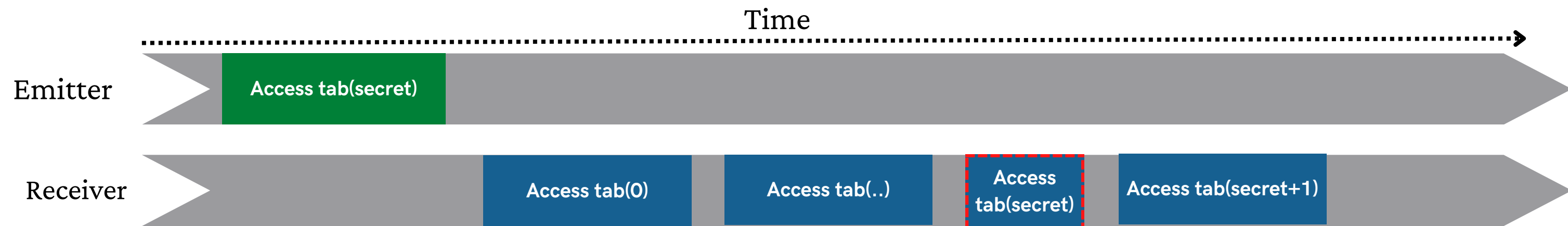
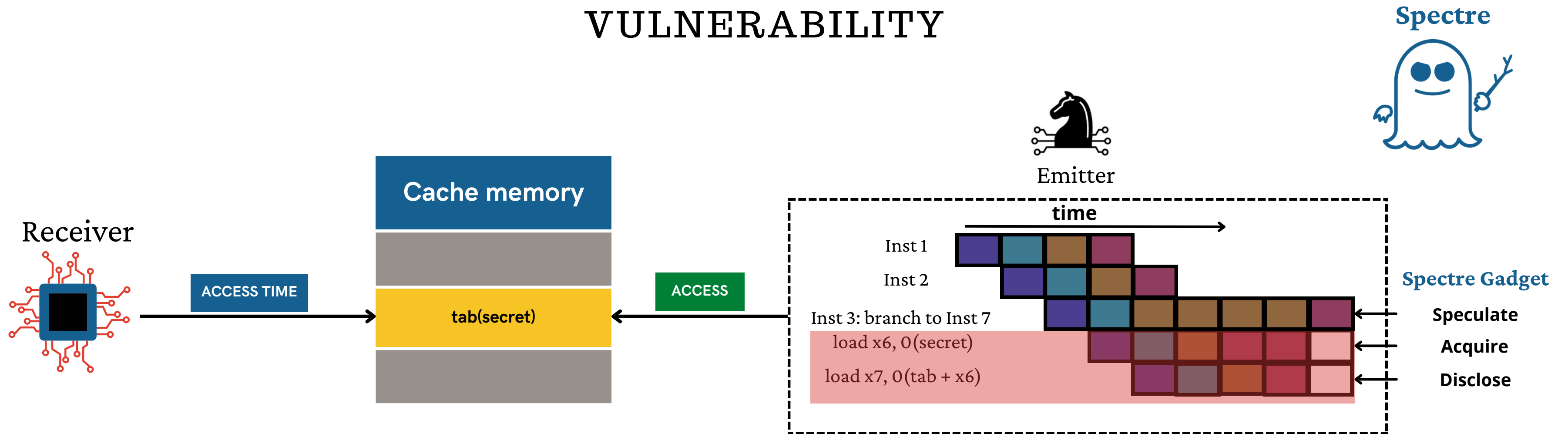
SPECTRE

MITIGATIONS

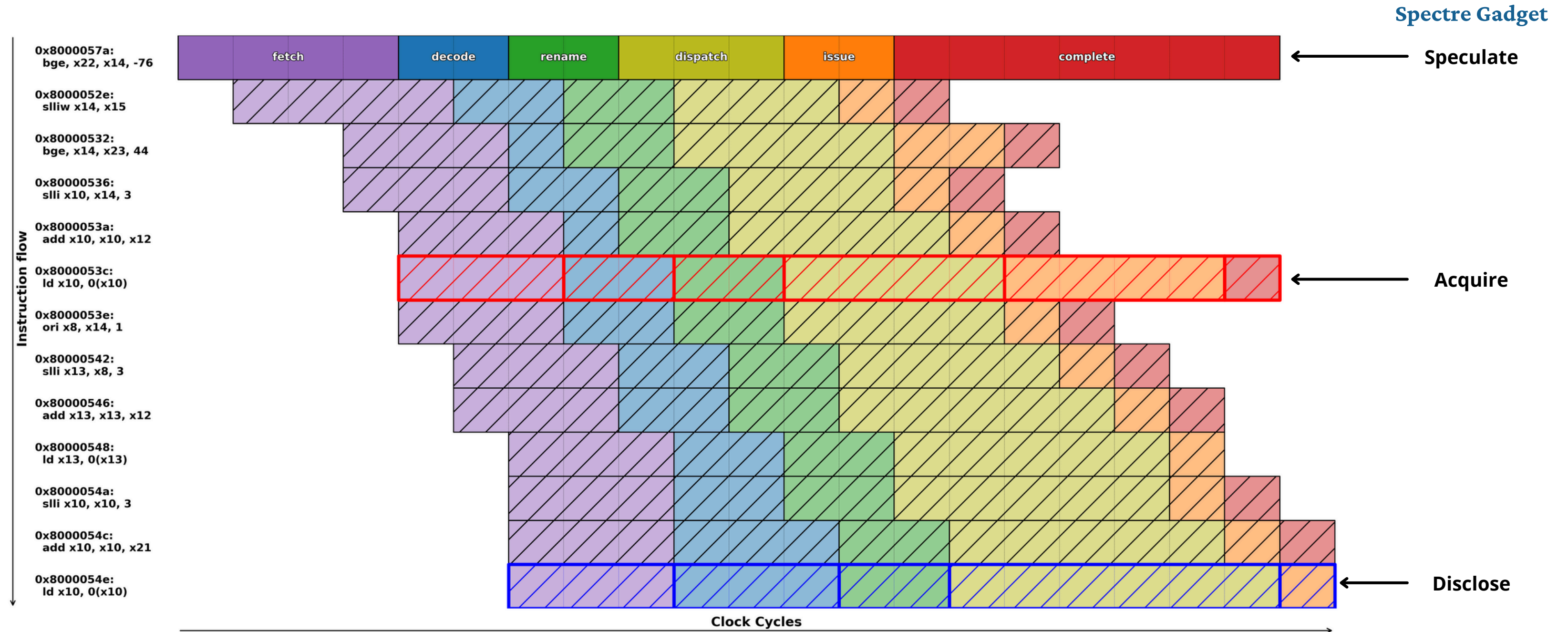
SELECTIVE
SPECULATION

RESULTS

TRANSIENT EXECUTION VULNERABILITY







TRANSIENT EXECUTION VULNERABILITY



Real exemple (random gadget from Embench execution)

SPECTRE

Covert channels		VARIANT 1	VARIANT 2	VARIANT 3	VARIANT 4	...
		SPECTRE-PHT	SPECTRE-BTB	SPECTRE-RSB	SPECTRE-STL	
	Cache timing	Paul Kocher, 2018		Giorgi Maisuradze, 2018	Jann Horn, 2018	
		Spectre Attacks: Exploiting Speculative Execution	N/A	ret2spec: Speculative Execution Using Return Stack Buffers	Speculative store bypass	...
	Branch target buffers		Andrea Mambretti, 2019			
		N/A	Two methods for exploiting speculative control flow hijacks	N/A	N/A	...
	Vector instructions	Michael Schwarz, 2018				
		NetSpectre: Read Arbitrary Memory over Network	N/A	N/A	N/A	...
	Port contention		Atri Bhattacharyya, 2019			
		N/A	SMoTherSpectre: Exploiting Speculative Execution through Port Contention	N/A	N/A	...
...	

MITIGATION

1

SOFTWARE-ONLY MITIGATION

2

HARDWARE-ONLY MITIGATION

PREREQUISITE

OVERVIEW

ENHANCING EXECUTION
SPEED

SPECTRE

MITIGATIONS

SELECTIVE
SPECULATION

RESULTS

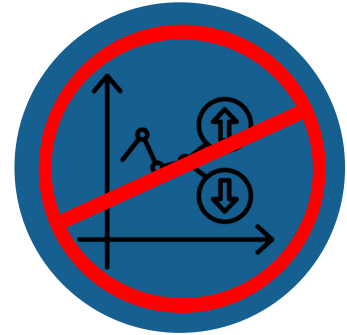


SELECTIVE SPECULATION

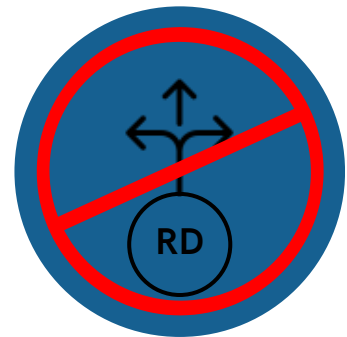
SPECULATION
&
SERIALIZATION
&
CONDITIONNAL FENCES

SELECTIVE SPECULATION: BASED ON REGISTER DEPENDENCIES

Speculation Fence: **fence.spec** rd, rs1



Fence.spec can only finalize its execution in **non-speculative mode**.



Ensures that the rd register is not used by other instructions in speculative mode.

fence.xxx rd, rs1

Serialization Fence: **fence.ser** rd, rs1



Fence.ser can finalize its execution in **speculative mode**.



Fence.ser introduces new data dependencies.

SELECTIVE SPECULATION

Different implementations of Speculation *fence* (fence.spec rd, rs1)

	Execute-Stall Fence	Dispatch-Stall Fence	Operand-Stall Fence
Able to execute speculatively	No	No	Yes (Instruction semantics are not respected)
Completion Condition	non-speculative	non-speculative	Operands are architecturally correct
Stage where fence.spec stall	Execution	Dispatch	Dispatch
Fence.spec Out-of-order	no	yes	yes

SELECTIVE SPECULATION

```
fence.xxx rd, rs1
```

rd = x0
rs1 = x0

Fence all

- Rs1 depends on all previous registers in program order.
- All subsequent instructions in the program order depend on fence.

PREREQUISITE

OVERVIEW

ENHANCING EXECUTION
SPEED

SPECTRE

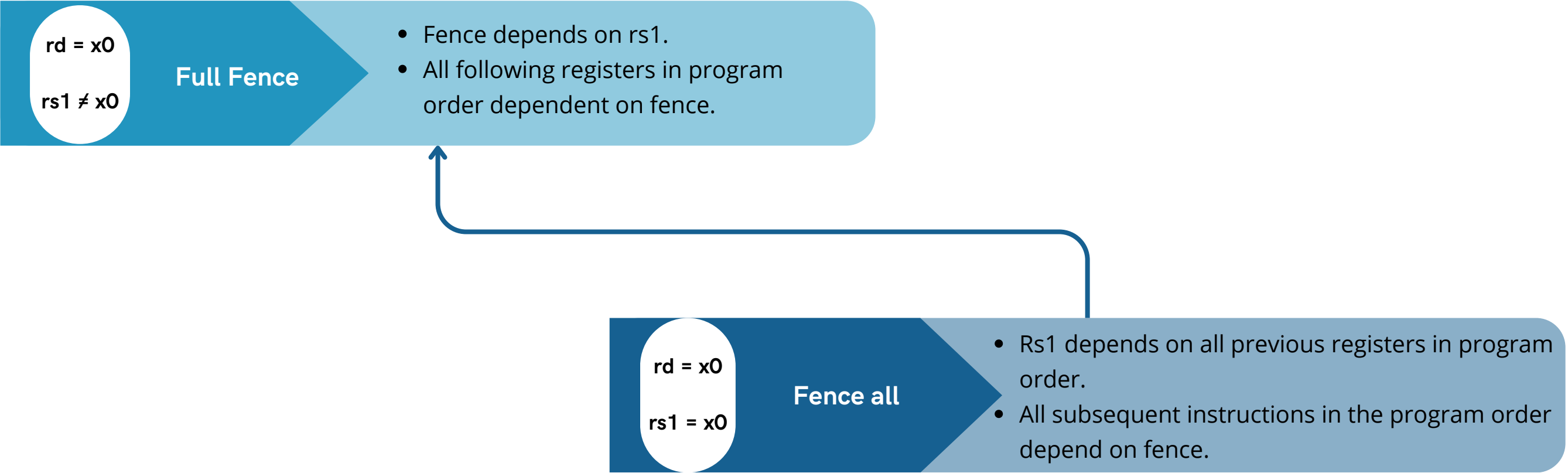
MITIGATIONS

SELECTIVE
SPECULATION

RESULTS

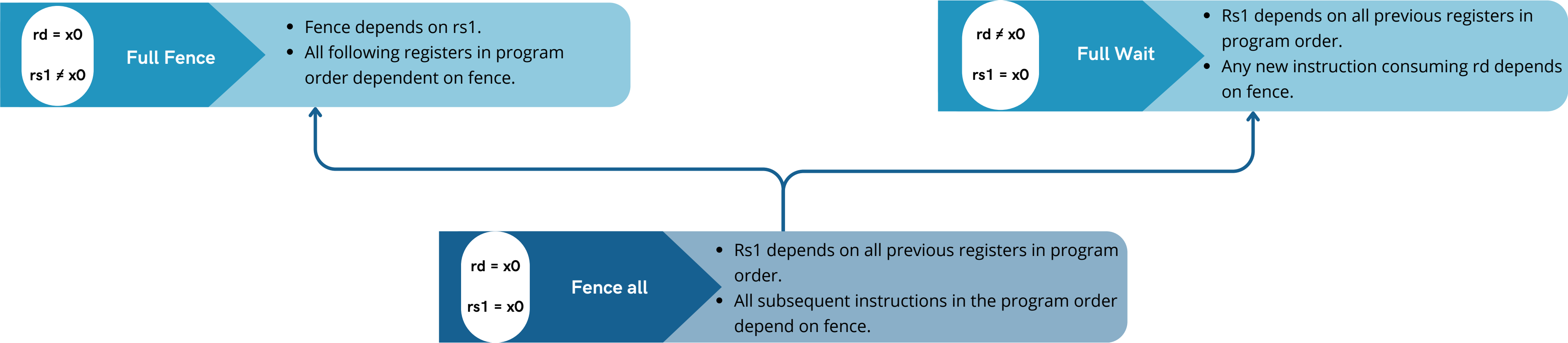
SELECTIVE SPECULATION

`fence.xxx rd, rs1`



SELECTIVE SPECULATION

`fence.xxx rd, rs1`



SELECTIVE SPECULATION

`fence.xxx rd, rs1`

`rd ≠ x0`
`rs1 ≠ x0`

Minimal fence

- Fence depends on rs1.
- Any new instruction consuming rd depends on fence.

`rd = x0`
`rs1 ≠ x0`

Full Fence

- Fence depends on rs1.
- All following registers in program order dependent on fence.

`rd ≠ x0`
`rs1 = x0`

Full Wait

- Rs1 depends on all previous registers in program order.
- Any new instruction consuming rd depends on fence.

`rd = x0`
`rs1 = x0`

Fence all

- Rs1 depends on all previous registers in program order.
- All subsequent instructions in the program order depend on fence.

SELECTIVE SPECULATION: BASED ON PREDICATE

Conditionnal Fence: `fence.cond` Rs1

ALL FOLLOWING REGISTERS IN PROGRAM ORDER DEPENDENT ON FENCE.COND.

Predicate value on Rs1 based on `branch condition`:



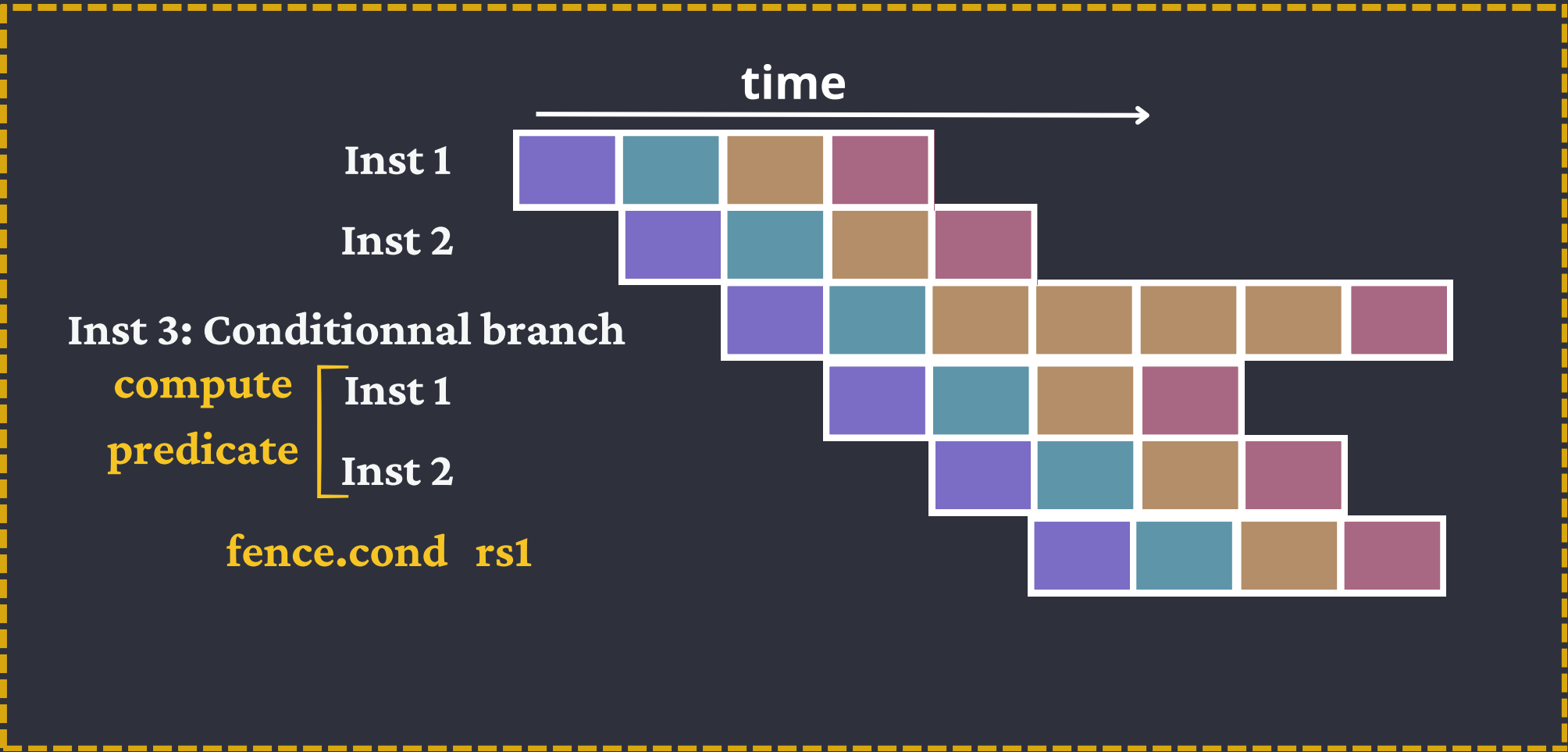
Good speculation: Rs1 = 0

=> `fence.cond` can be executed speculatively



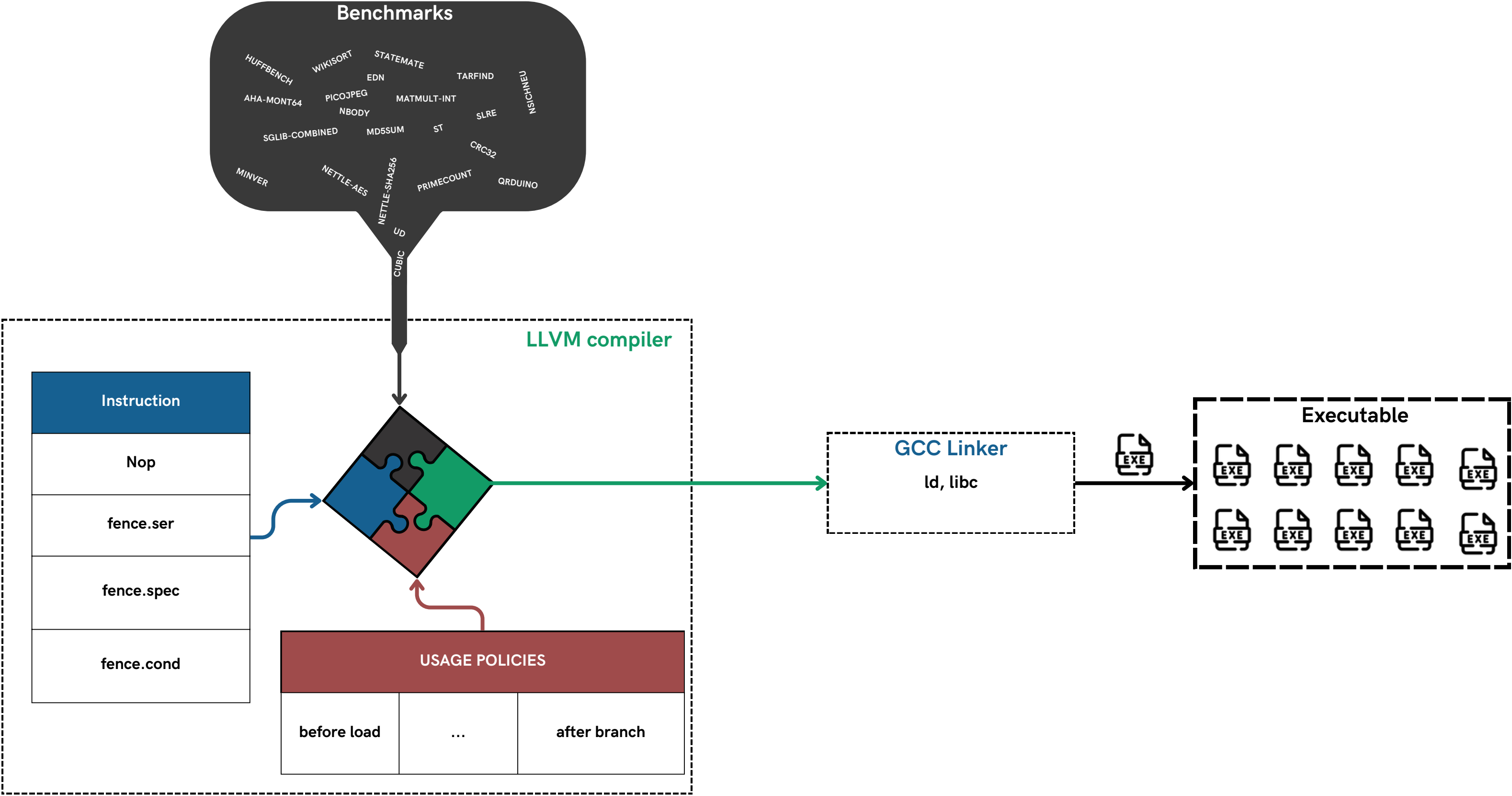
Miss-speculation : RS1 \neq 0

=> `fence.cond` waits for end of speculative execution



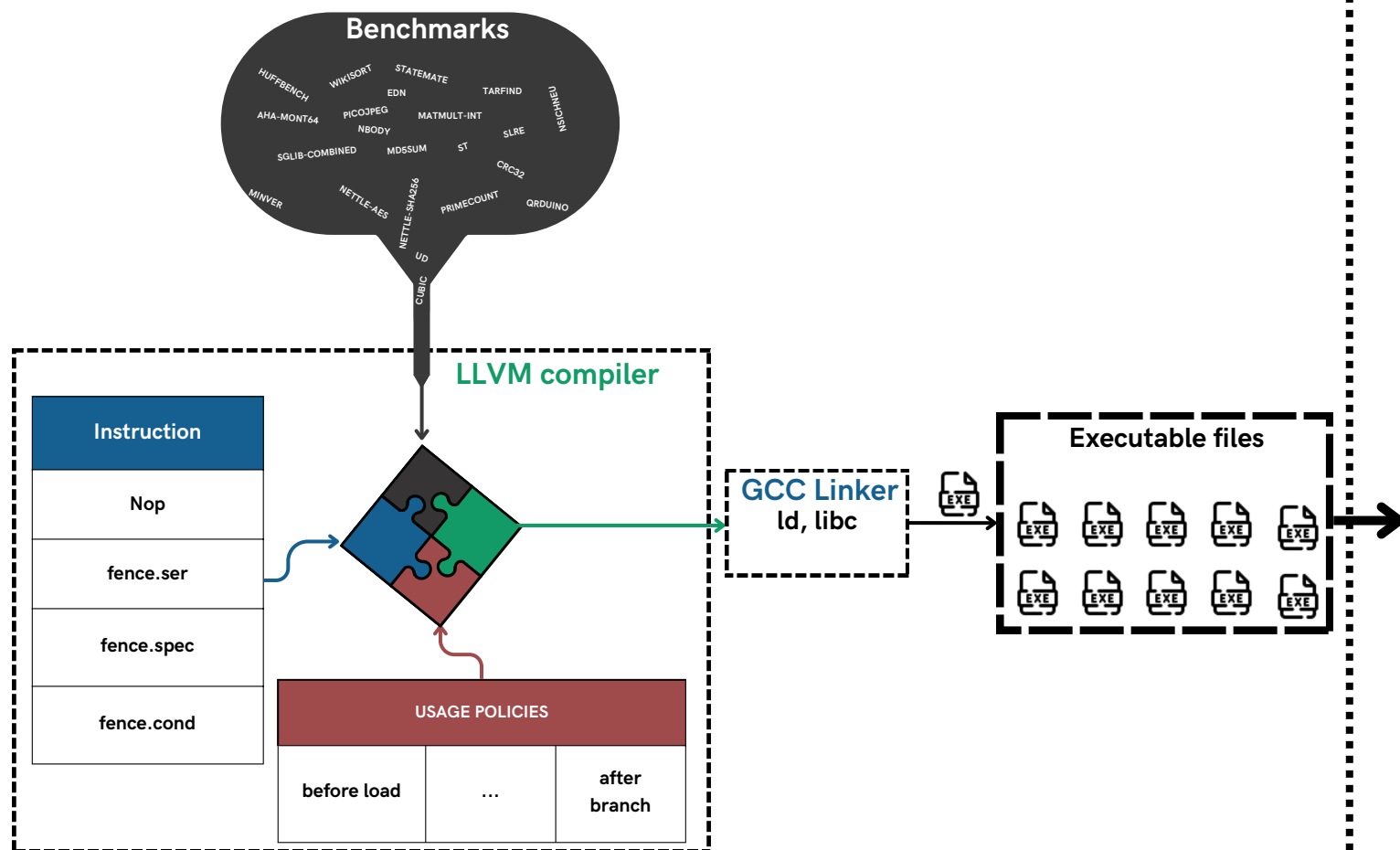
TESTING OUR HYPOTHESIS

COMPILATION AND TOOLCHAIN

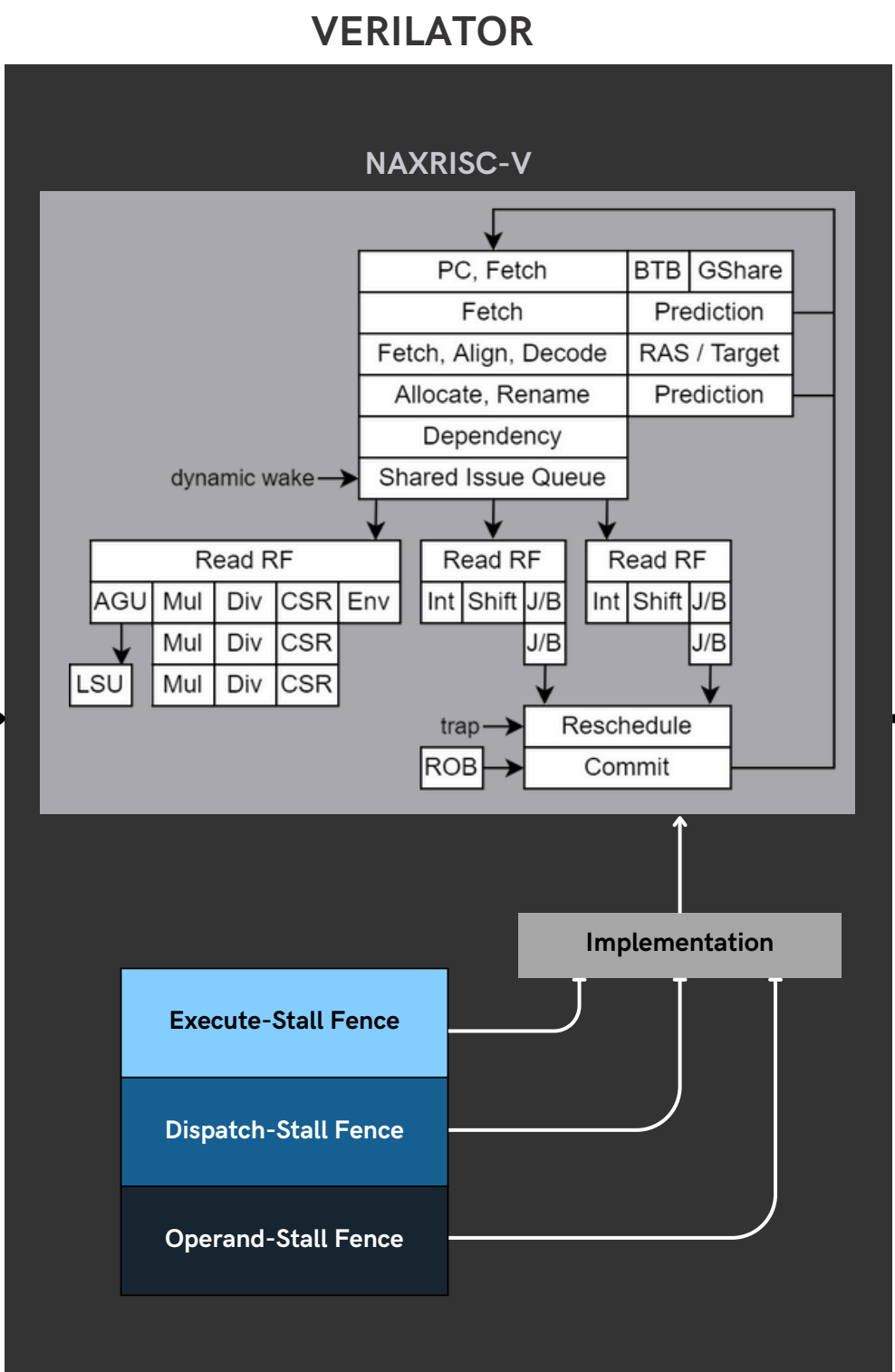


EVALUATION ENVIRONMENT

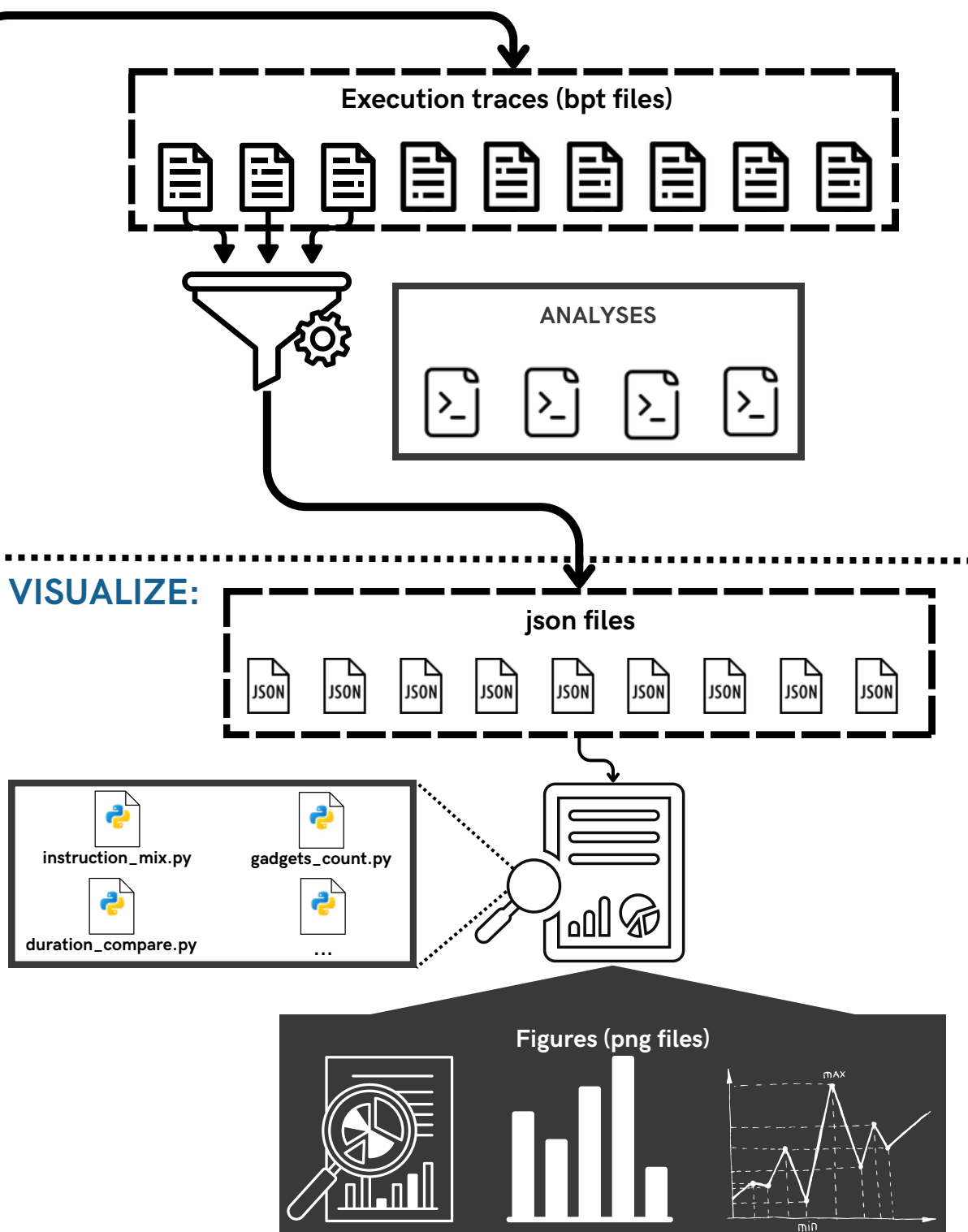
COMPILATION:



SIMULATION:



ANALYSIS:



PREREQUISITE

OVERVIEW

ENHANCING EXECUTION
SPEED

SPECTRE

MITIGATIONS

SELECTIVE
SPECULATION

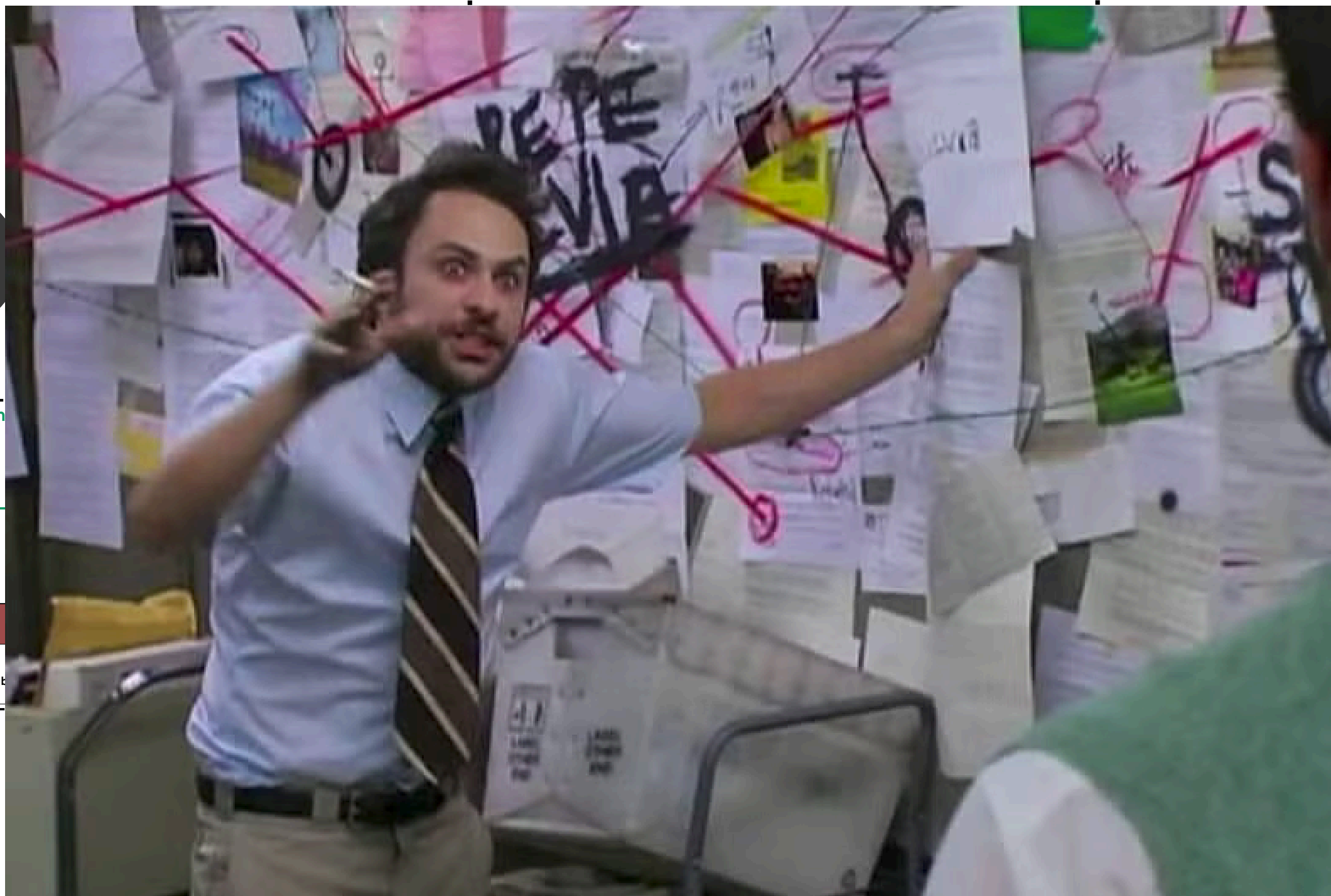
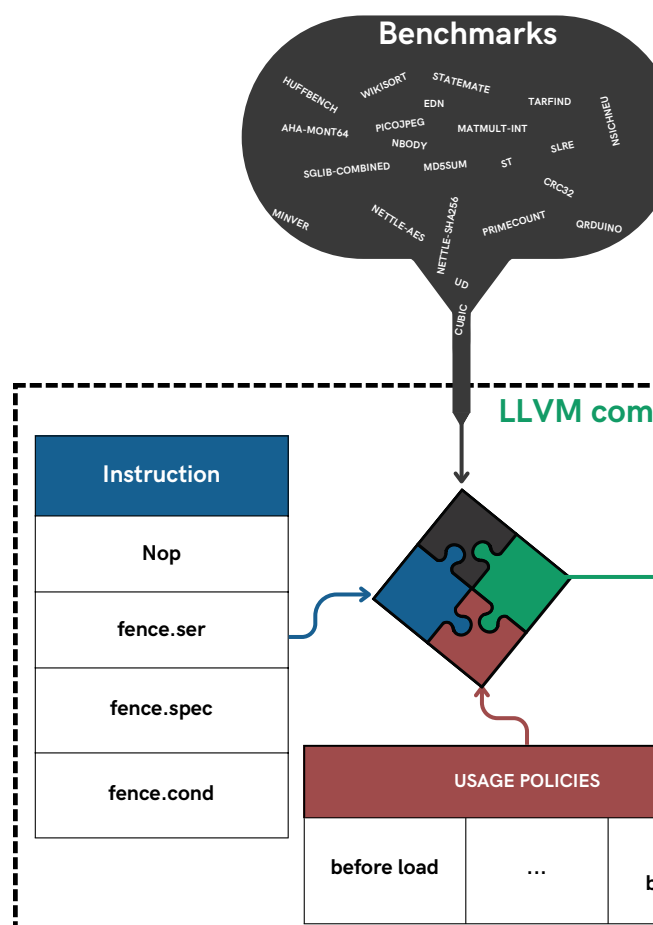
RESULTS

EVALUATION ENVIRONMENT

COMPILATION:

SIMULATION:

■ ANALYSIS:



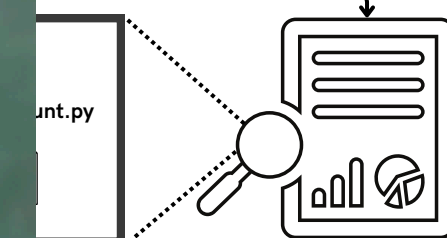
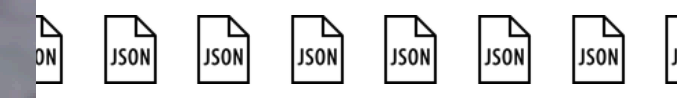
Execution traces (bpt files)



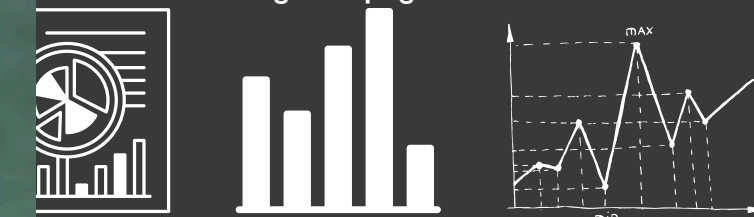
ANALYSES



json files



Figures (png files)



PREREQUISITE

OVERVIEW

ENHANCING EXECUTION SPEED

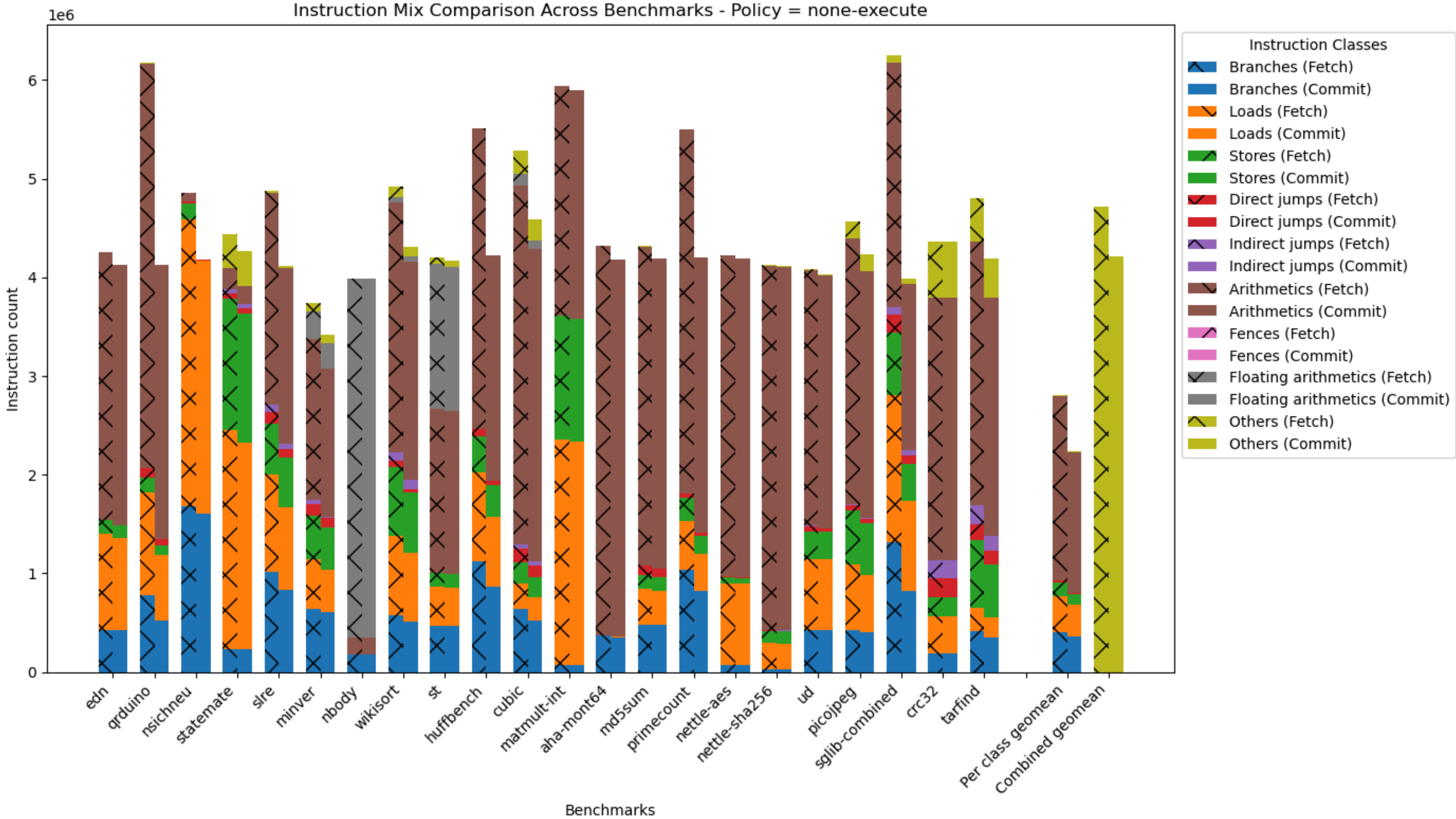
SPECTRE

MITIGATIONS

SELECTIVE SPECULATION

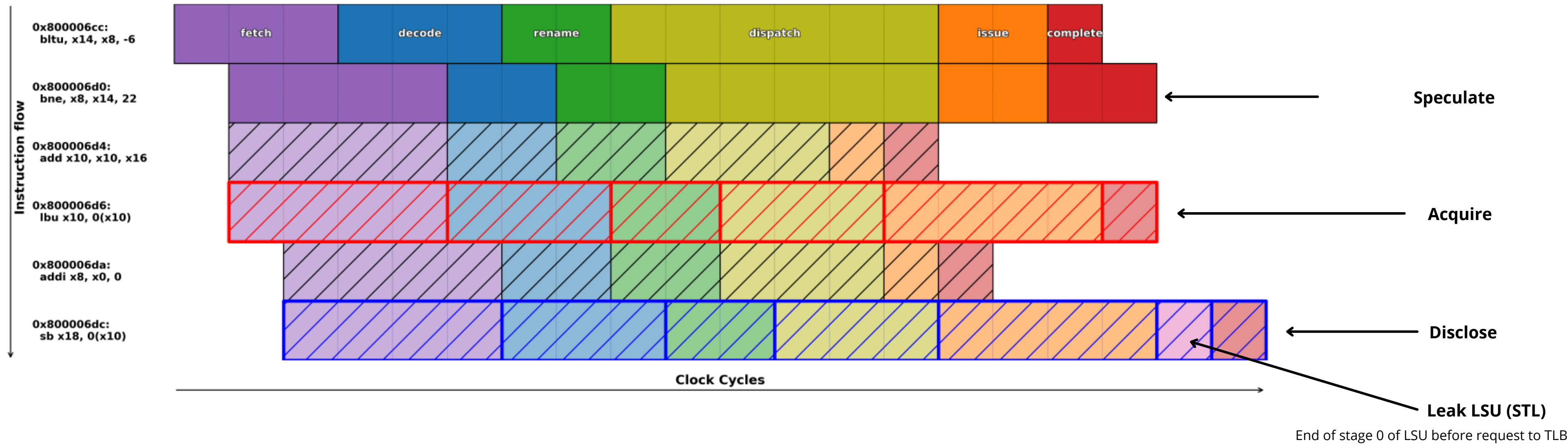
RESULTS

TRACE AND ANALYSIS



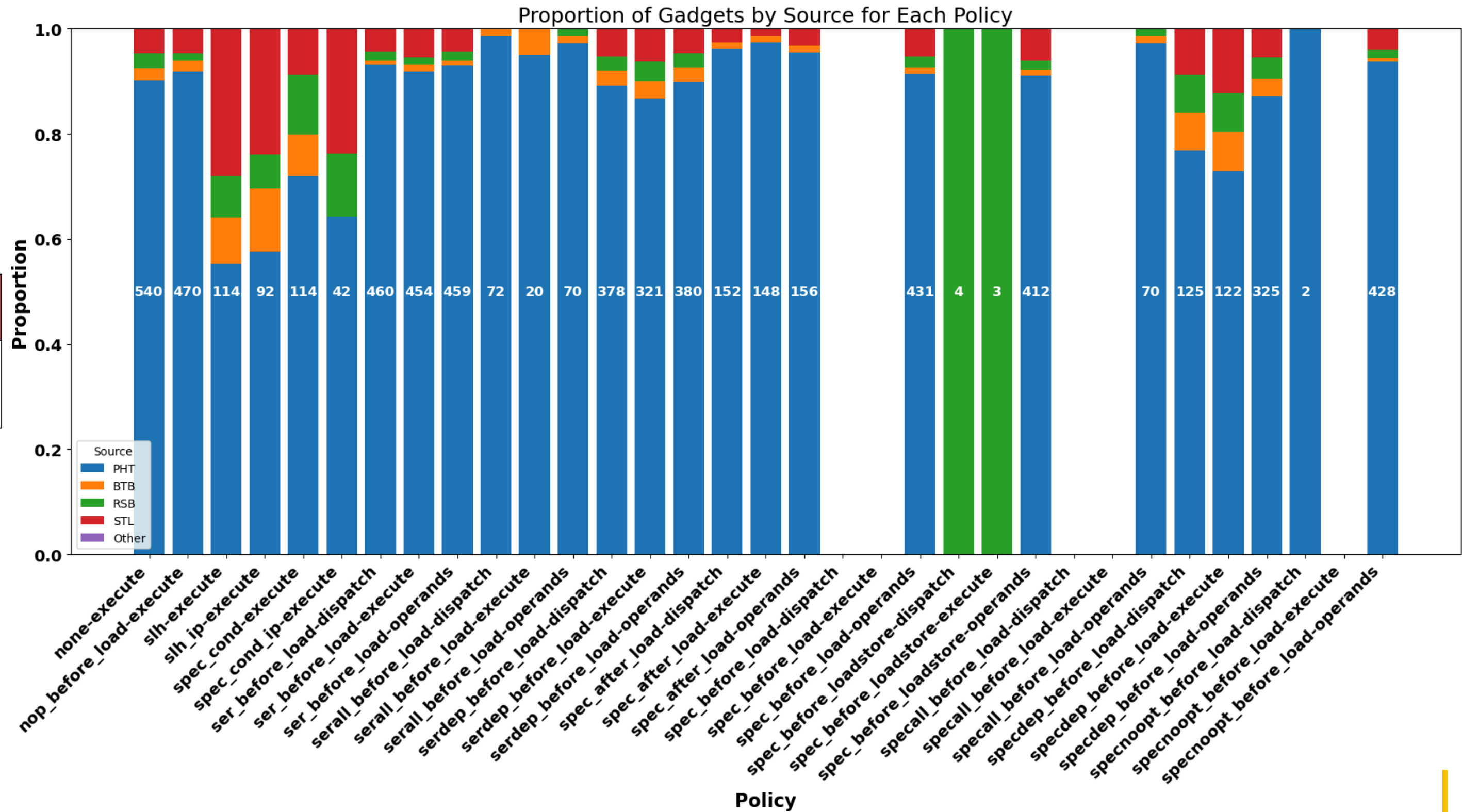
FINE-GRAINED CUSTOM ANALYSIS

Spectre Gadget



SPECTRE SOURCE TYPE DETECTION

USAGE POLICIES			
before load	after load	...	after branch



PREREQUISITE

OVERVIEW

ENHANCING EXECUTION
SPEED

SPECTRE

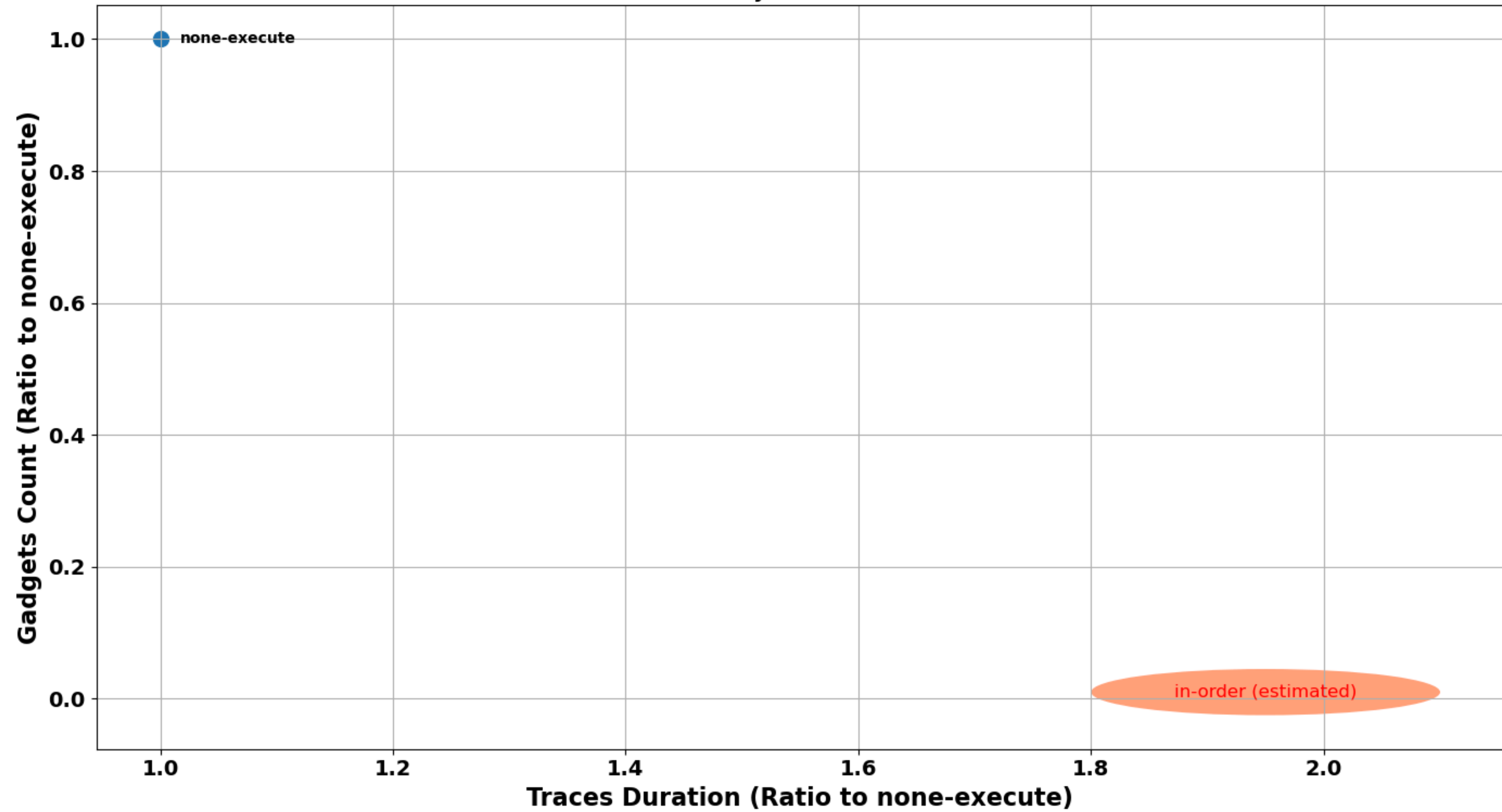
MITIGATIONS

SELECTIVE
SPECULATION

RESULTS

PRELIMINARY RESULTS

Security vs. Duration



PREREQUISITE

OVERVIEW

ENHANCING EXECUTION
SPEED

SPECTRE

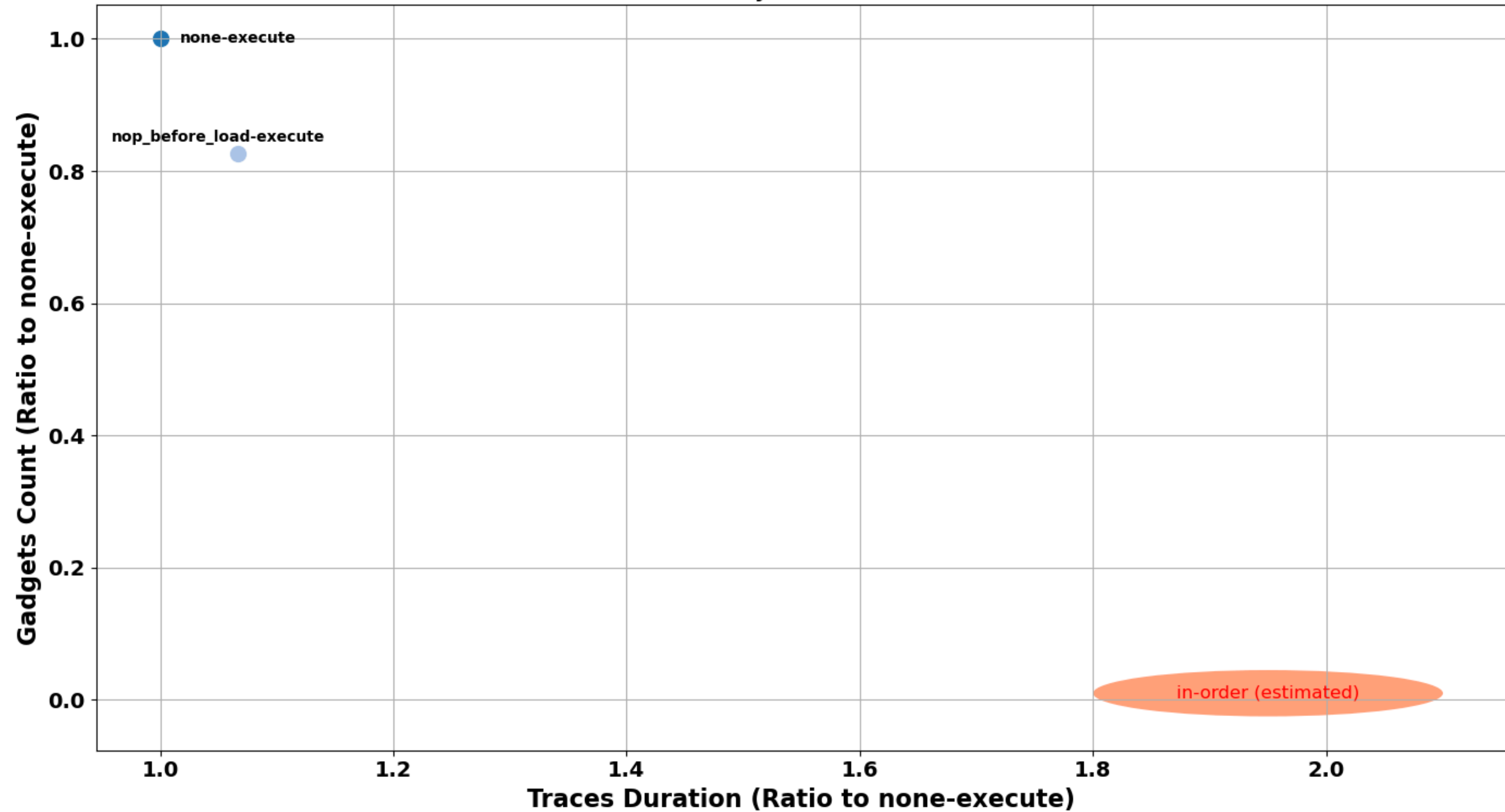
MITIGATIONS

SELECTIVE
SPECULATION

RESULTS

PRELIMINARY RESULTS

Security vs. Duration



PREREQUISITE

OVERVIEW

ENHANCING EXECUTION
SPEED

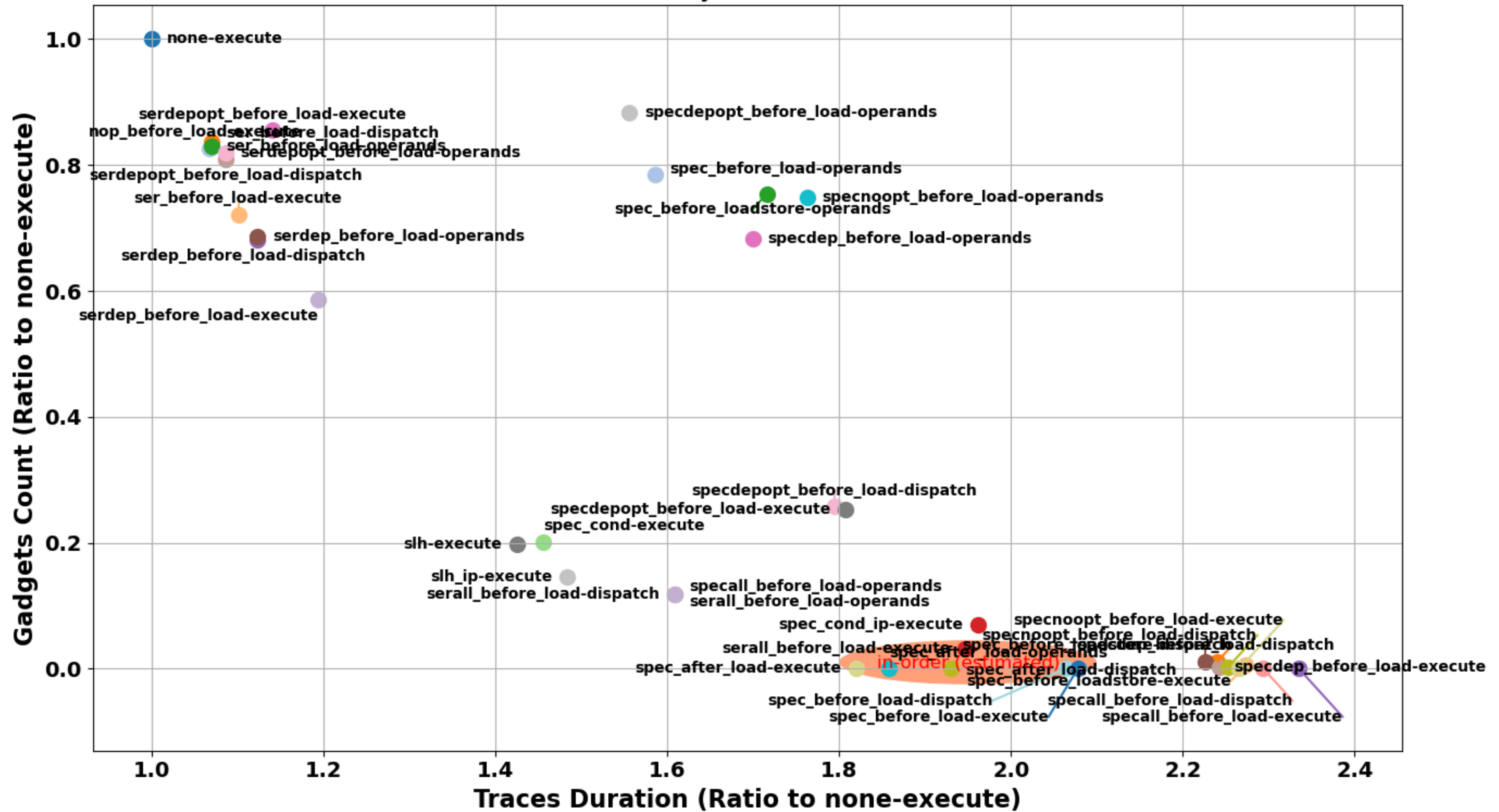
SPECTRE

MITIGATIONS

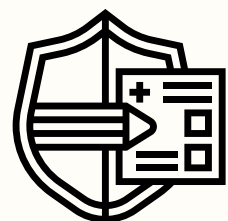
SELECTIVE
SPECULATION

RESULTS

Security vs. Duration



CONCLUSION



POLICY EFFICIENCY

Some policies offer better trade-offs between security and performance.



INSTRUCTION BARRIERS INSUFFICIENT

Achieving full protection would double execution times, highlighting the significant performance cost of secure solutions.



SPECTRE ATTACKS ARE HERE TO STAY

Is there truly a more effective approach to mitigate Spectre vulnerabilities?



WHAT NEXT?

By incorporating the concept of **secret** data into our protection strategy, can we strike a better balance between security and performance?

THANK YOU FOR YOUR ATTENTION



<https://hal.science/IRISA/hal-05061555v1>



https://gitlab.inria.fr/arsene-pepr/eval/spectre/-/tree/embench?ref_type=heads



<https://www.linkedin.com/in/herinomena-andrianatrehina-1127a419b/>



OPEN TO OPPORTUNITIES STARTING JANUARY 2026.