Ahead of Time Generation for GPSA Protection in RISC-V Embedded Cores

Louis Savary, Simon Rokicki and Steven Derrien

Université de Rennes

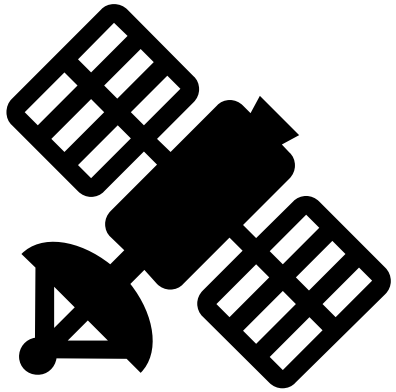
UMR IRISA

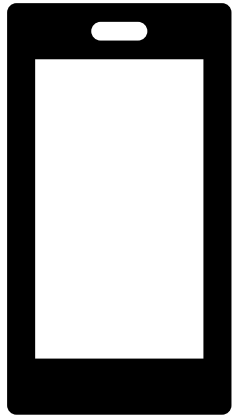
Context

Embedded Systems

- Energy constrained
- Everywhere, for every usage

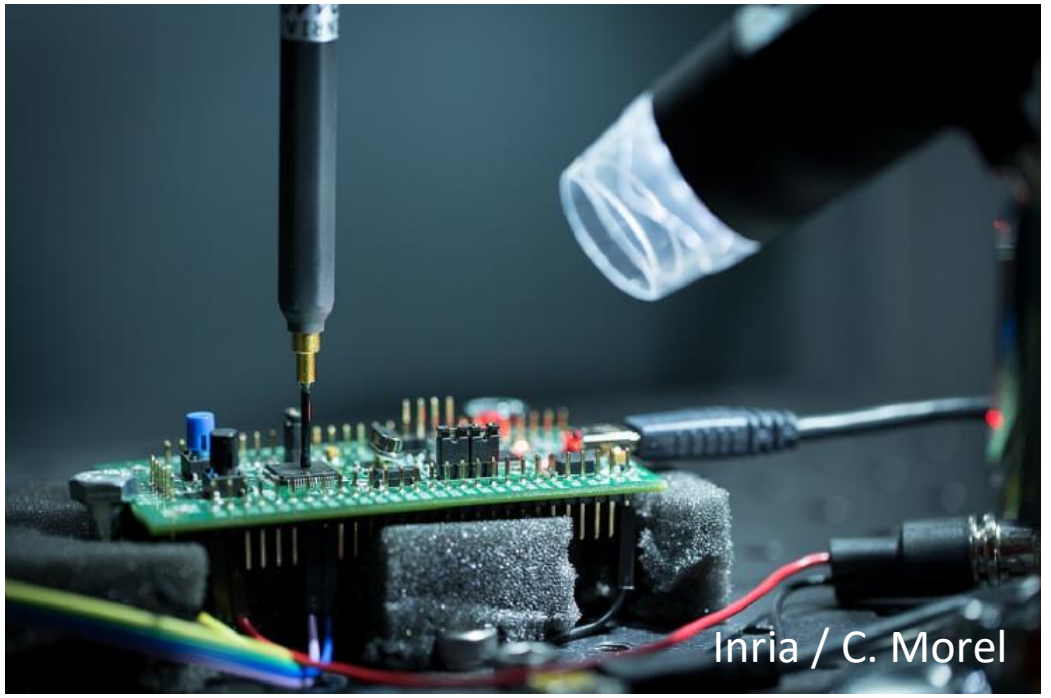
- Remote, mobile
  - allows invasive attacks





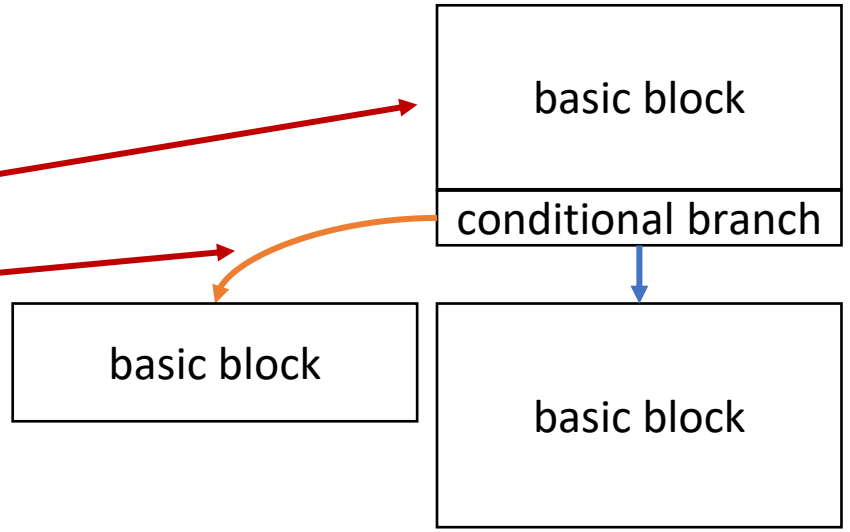
Context

Fault Injection Attacks (FIA)



- Alter transistors state by external means
  - laser, clock, power, EM

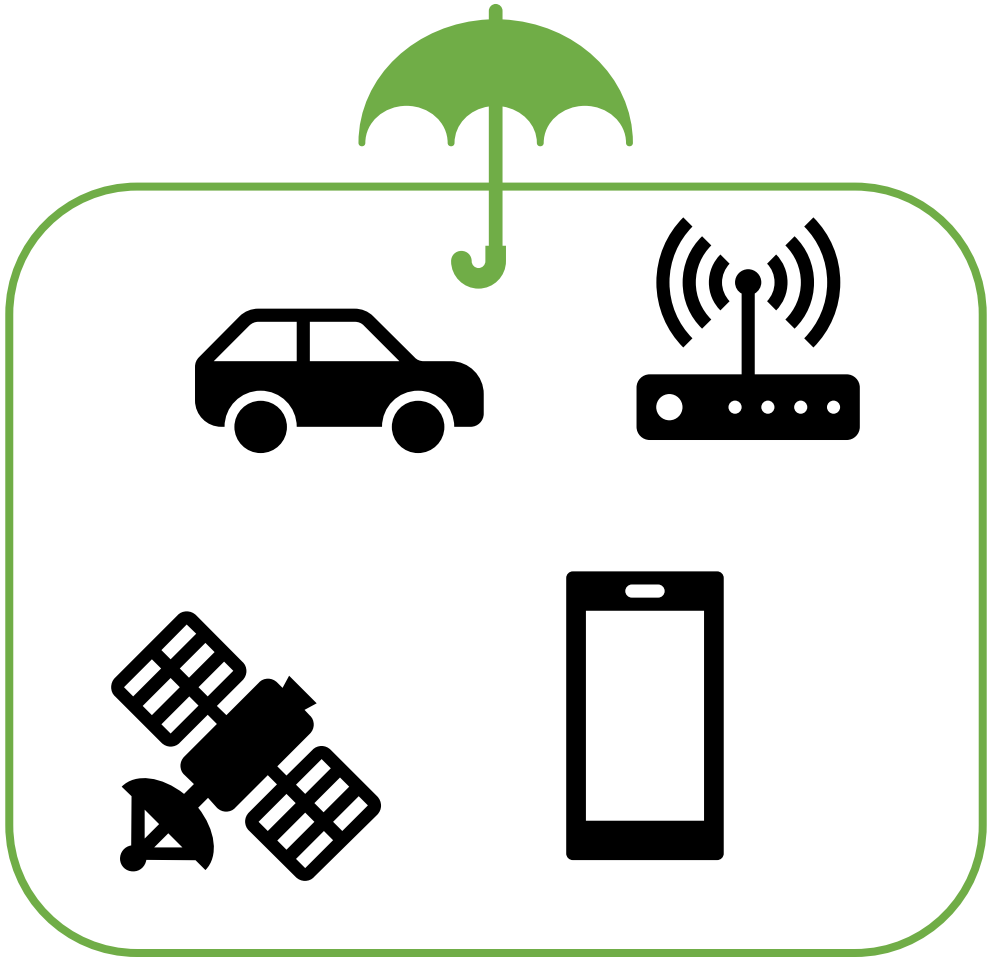
- Can propagate and cause errors:
  - Data corruption
  - Control flow errors
    - Instruction skip/repeat
    - Branch faulting



Context

Countermeasures against Fault Injection

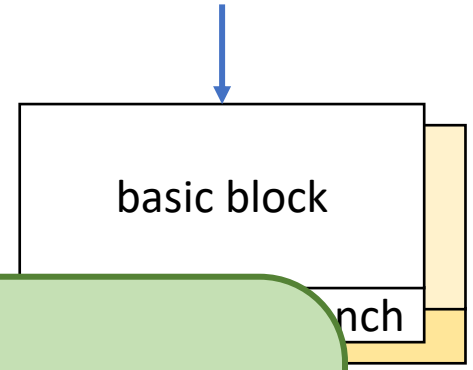
- Techniques for fault detection
  - redundancy, signatures
- Multiple implementations
  - modifying program
  - modifying architecture



GPSA and CSM

Global Path Signature Analysis & Continuous Signature Monitoring

- Verifying the Control Flow
- Signature processing
- At compile time
  - **Each instruction** has a **signature**
  - ***f* fu**
  - co
- Dynar
  - ins
  - ver
  - pat



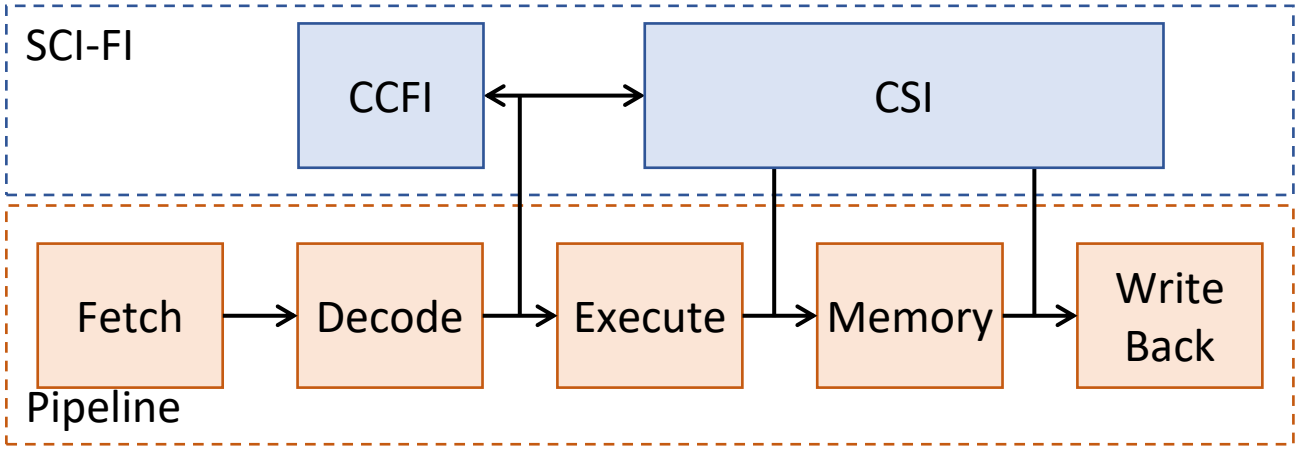
Requirements (usually)

- (micro)architecture
  - dynamic signature **computation** and **comparison**
- compiler
  - reference **signature** and **patch generation**

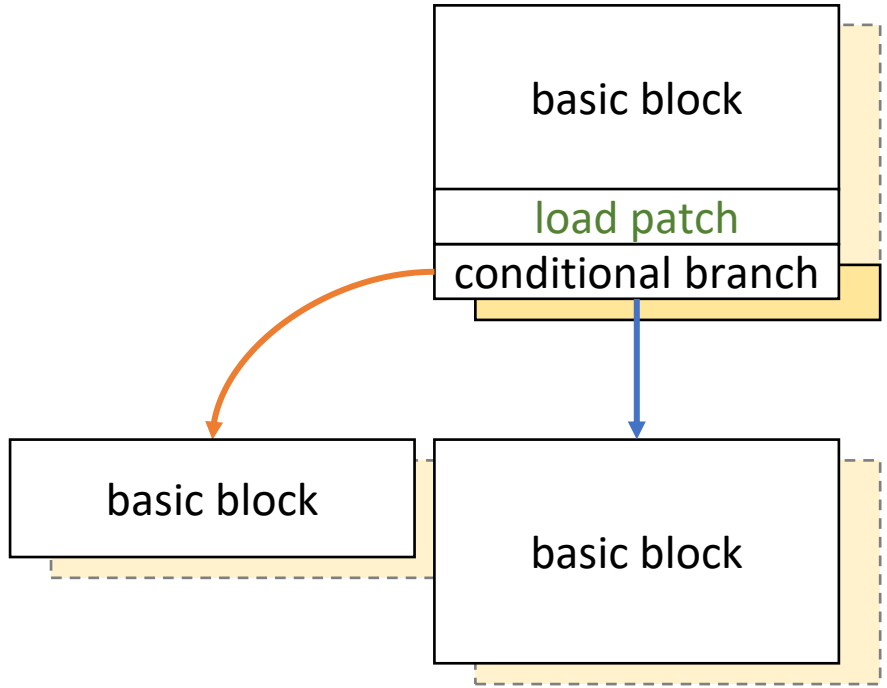
M. Werner et al. "*Protecting the Control Flow of Embedded Processors against Fault Attacks*". CARDIS 2016

SCI-FI
Countermeasure

- GPSA and CSM implementation
- microarchitecture modification to a pipeline
  - **CSI**: ensures pipeline execution integrity
  - **CCFI**: dynamic signature processing



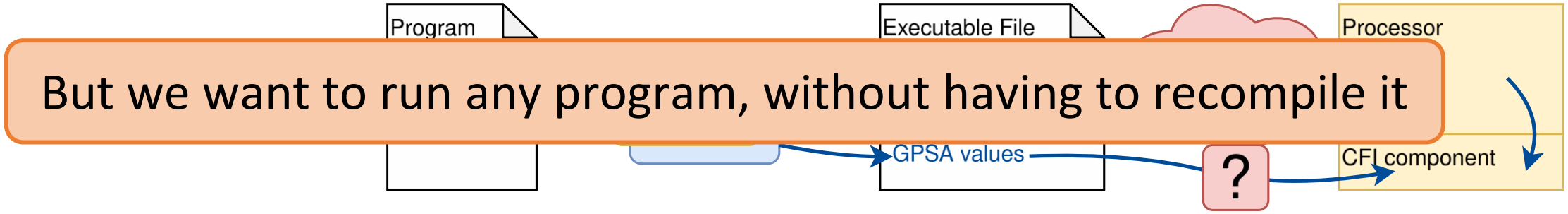
- compiler modification:
  - store in the .data section
  - additional instructions



T. Chamelot, D. Couroussé, and K. Heydeman *"SCI-FI , Code, and Control Flow Integrity against Fault Injection Attacks,"* in *DATE*, 2022

SCI-FI
Countermeasure

- Only considering faults in the processor logic
- GPSA and CSM implementation
  - that ensures Control Flow, Code and Control Signal Integrities
- Limits :
  - requires compiling the application
  - indirect jumps (unknown in CFG)
  - context switch

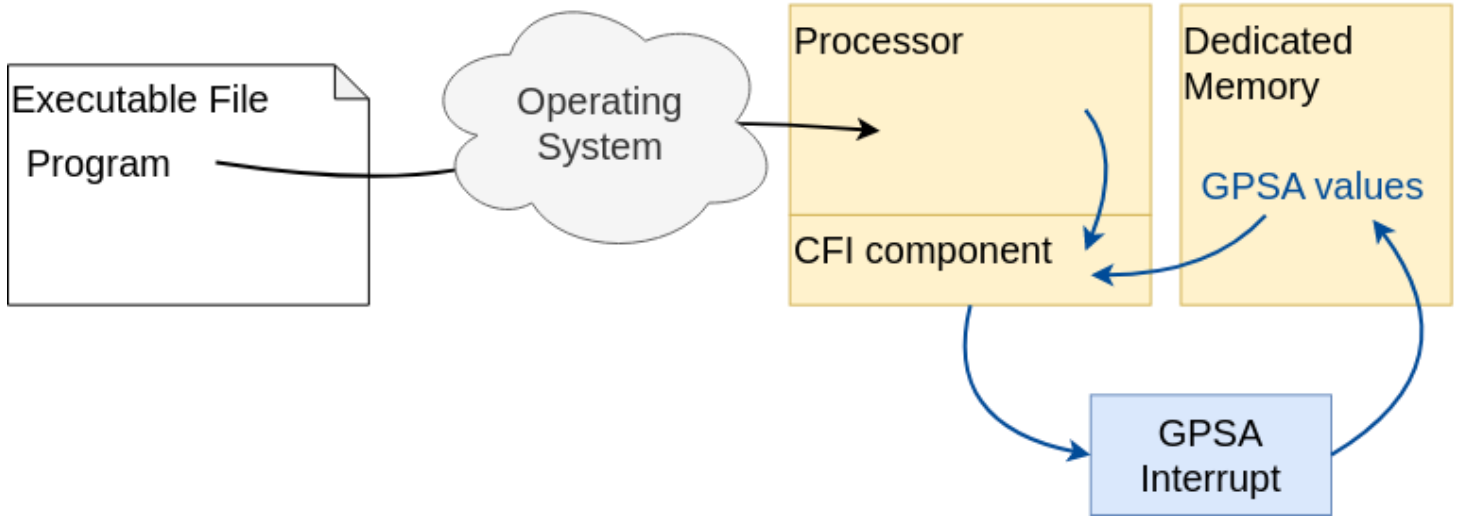


T. Chamelot, D. Couroussé, and K. Heydeman *"SCI-FI , Code, and Control Flow Integrity against Fault Injection Attacks,"* in *DATE*, 2022

Runtime GPSA generation

Another solution

- Rely on runtime environment for GPSA generation
  - Indirect jumps
  - Context switches



Runtime GPSA generation

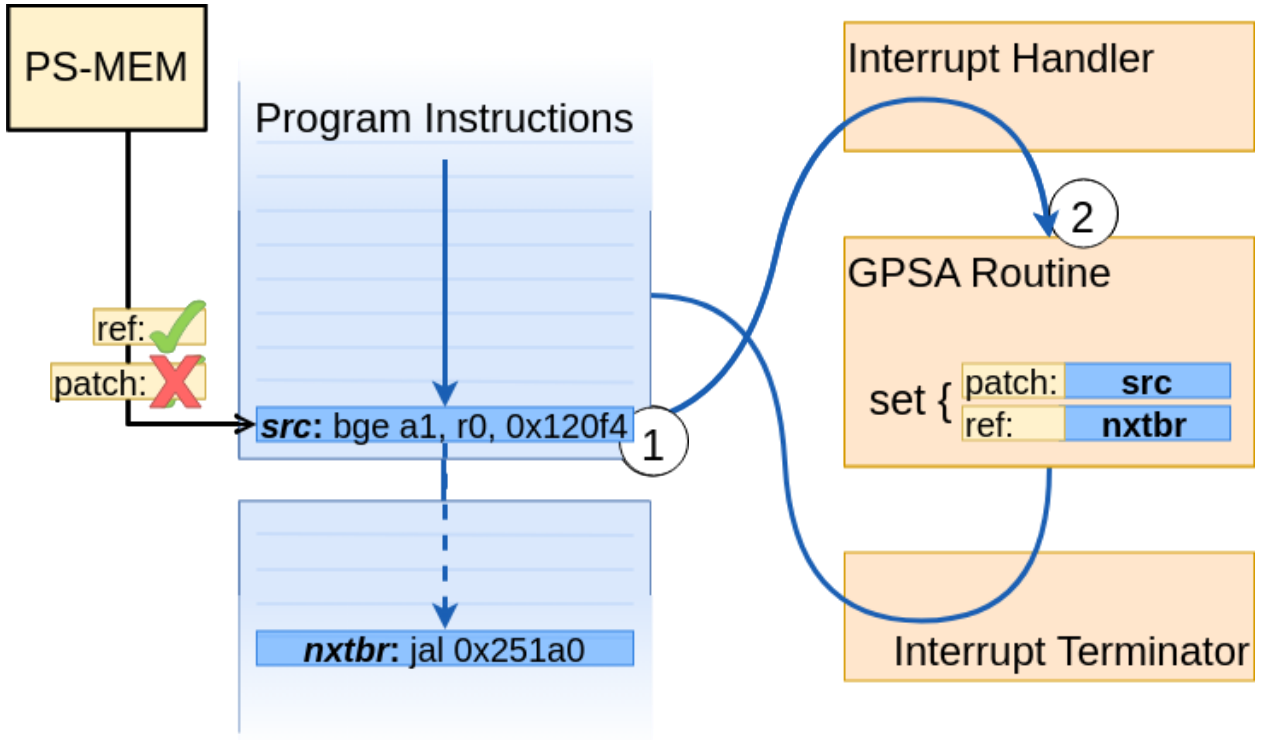
Execution Scheme

- Dynamic GPSA and CSM generation

1. Branch execution
   - signature must exist
   - if patch doesn't exist: **interrupt**

2. GPSA interrupt execution
   - reference for *nxtbr*
   - patch for *src*

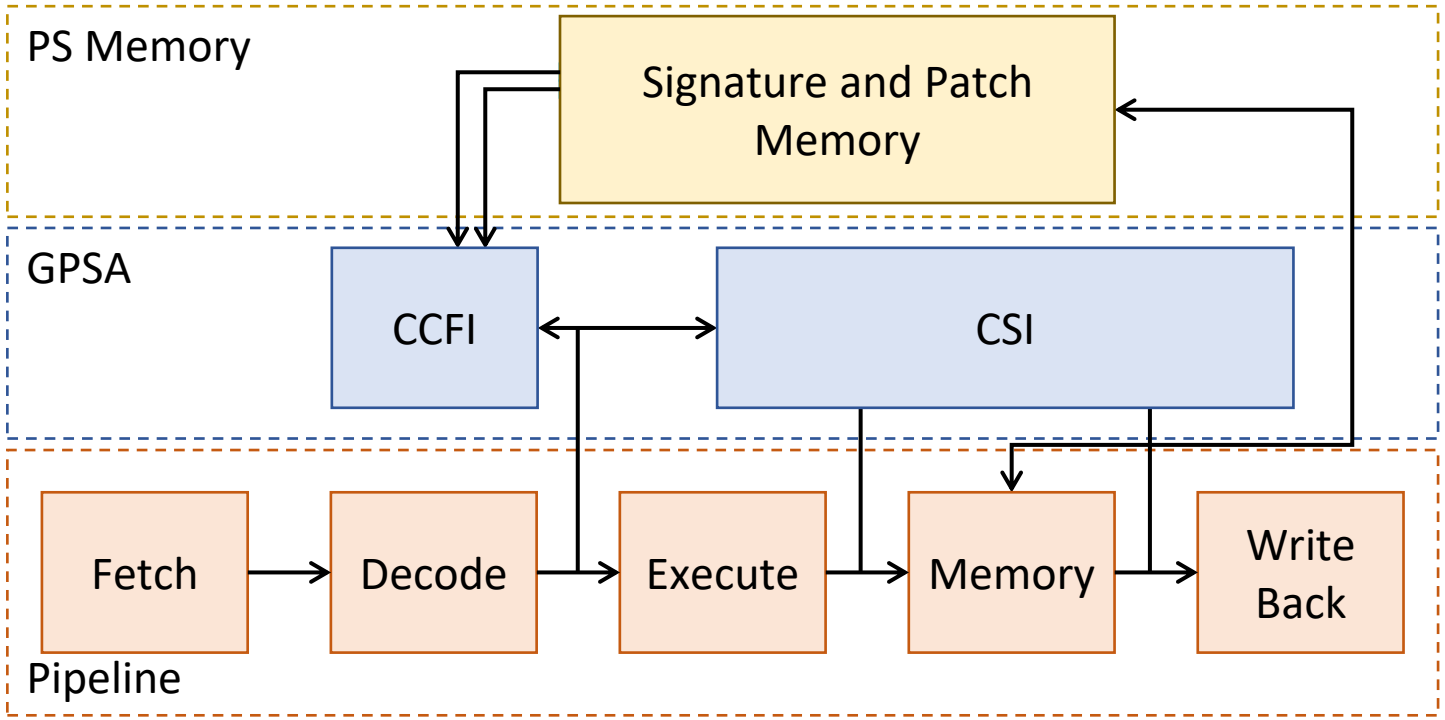
3. Resume execution until next node



Runtime GPSA generation

Hardware/Software runtime

- Avoiding patches and references additional load instructions
- Adding a dedicated memory
  - 2-values read port for CCFI
  - reserved instructions
- and some operators …



Runtime GPSA generation

Another solution

- Rely on runtime environment for GPSA generation
  - Indirect jumps
  - Context switches
  - High performance overhead
  - No Code Integrity
  - No reuse from previous executions

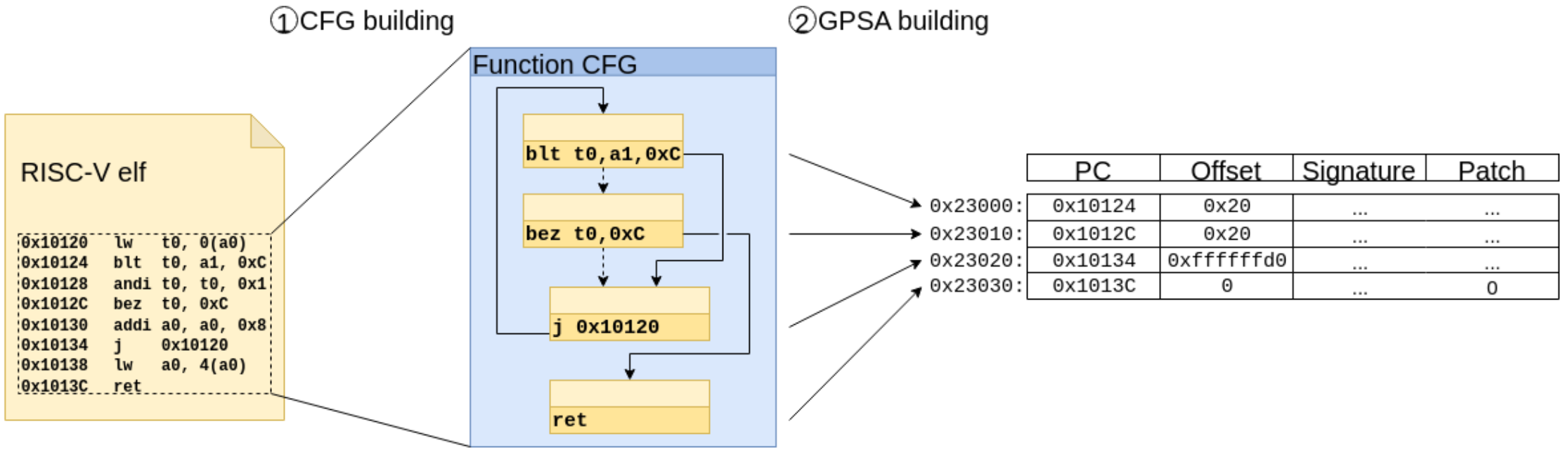
Our approach

Hardware/Software runtime

- Replacing the compiler with
  - a static analysis for GPSA generation
  - and a runtime environment to handle dynamic events
  - Indirect jumps
  - Context switches
  - Code integrity

Our approach

Ahead-of-Time analysis



a) binary code sample

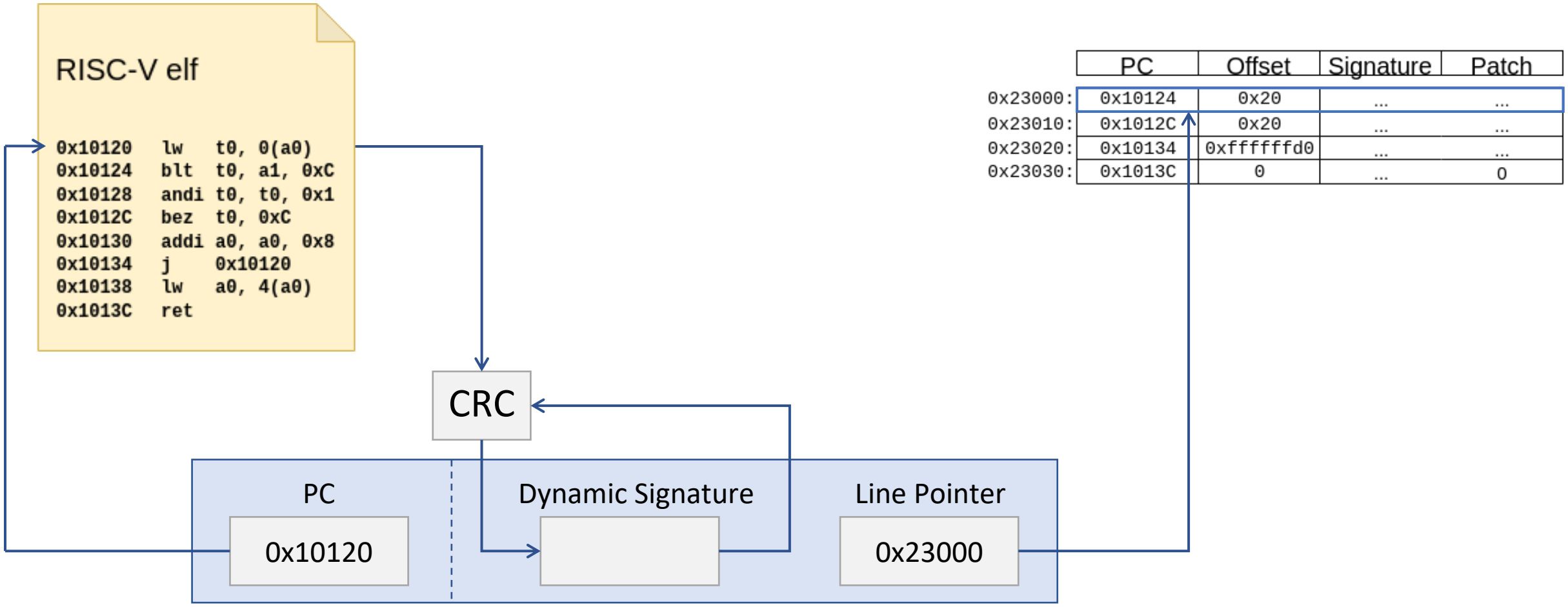
b) CFG and list of basic blocks

c) sorted list of GPSA values

Upon installation

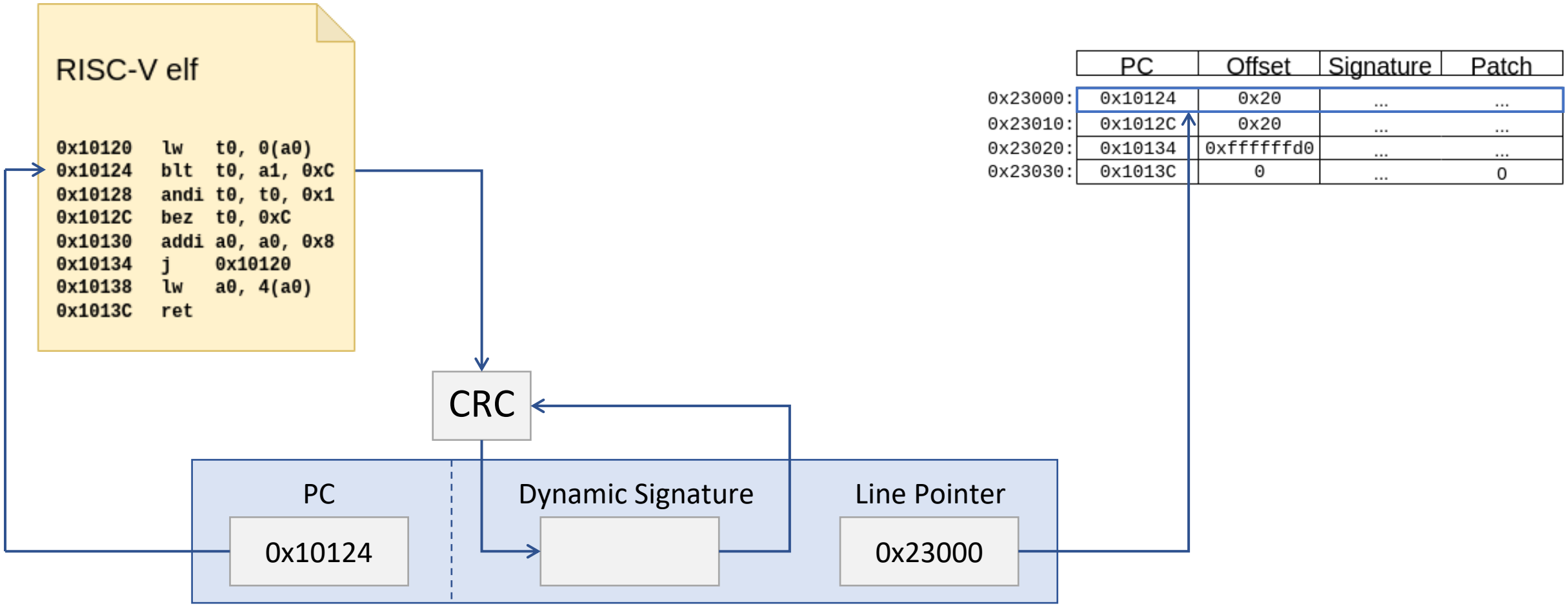
Our approach

Ahead-of-Time analysis



Our approach

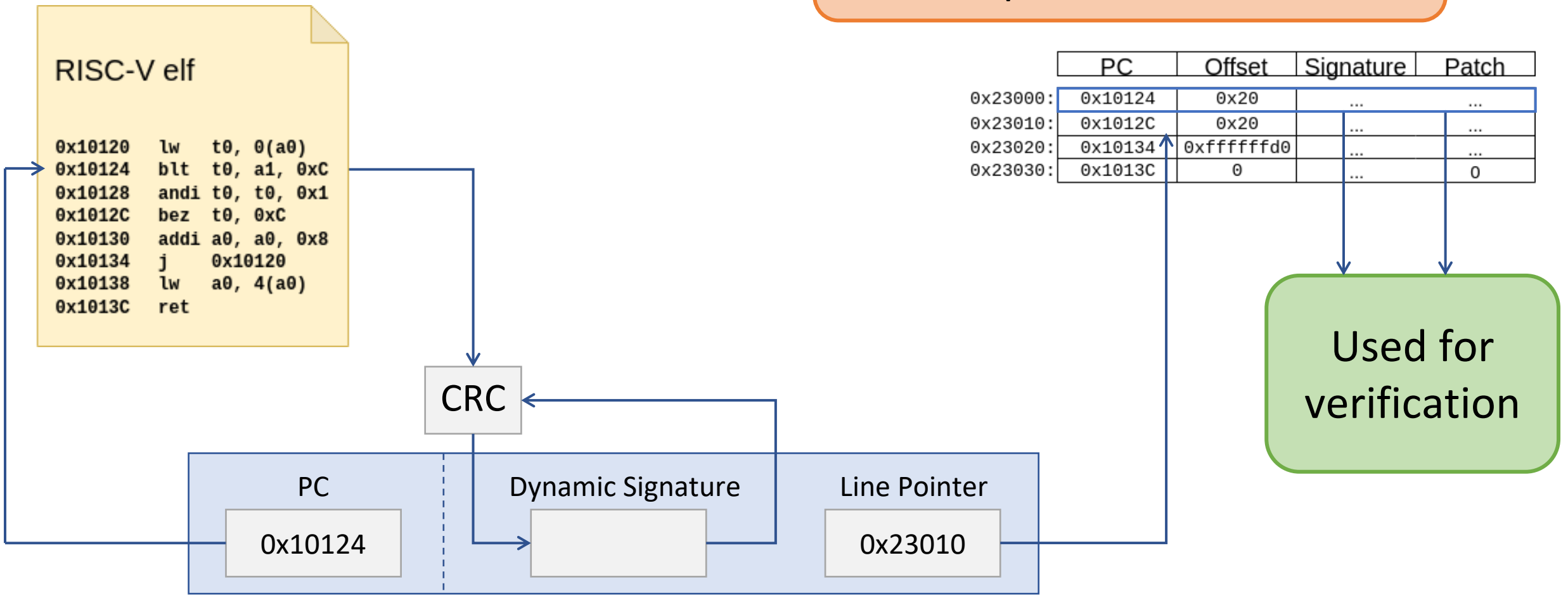
Ahead-of-Time analysis



Our approach

Ahead-of-Time analysis

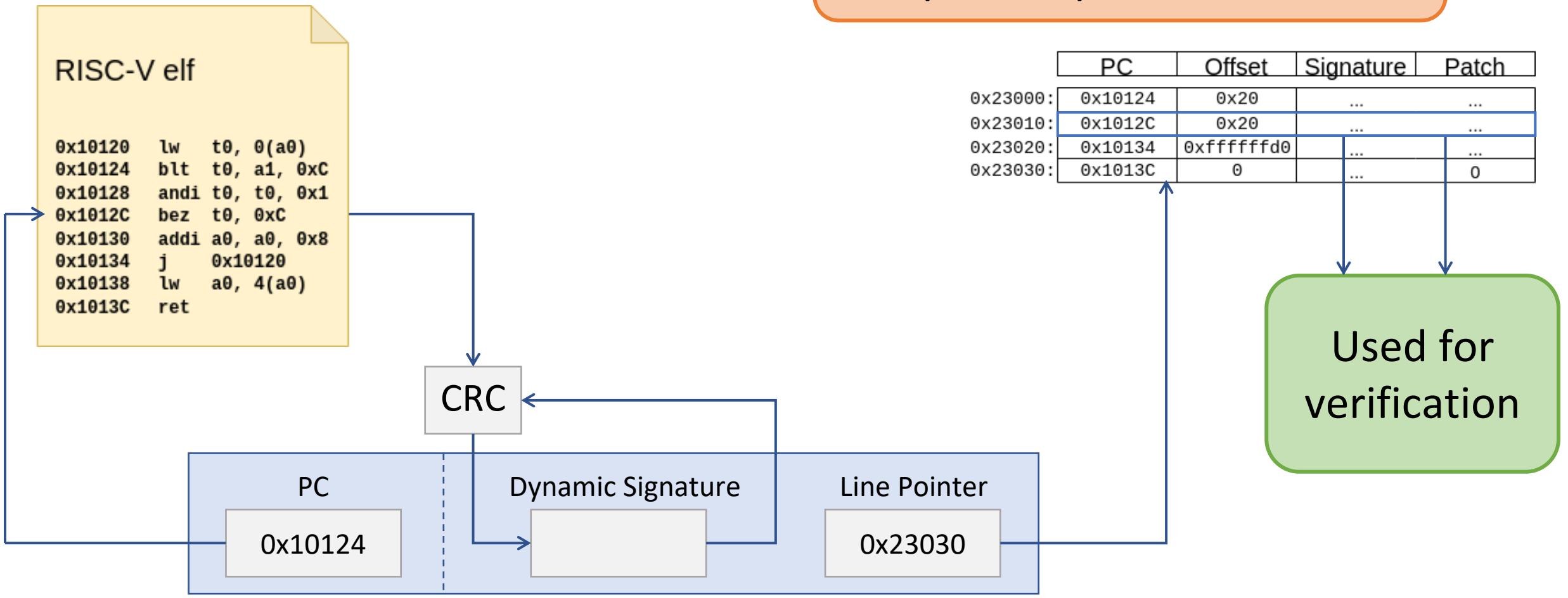
Branch not taken: Offset not used, pointer to next line



Our approach

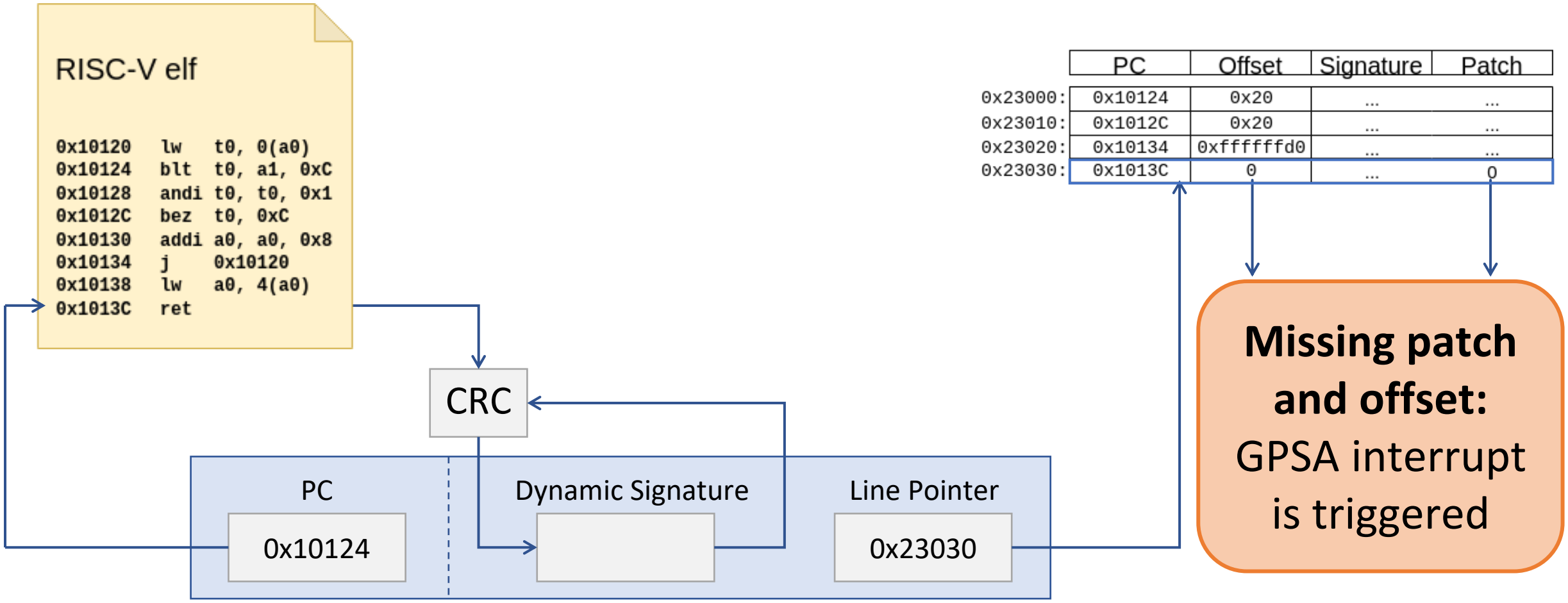
Ahead-of-Time analysis

Branch taken: Offset used, pointer plus two lines



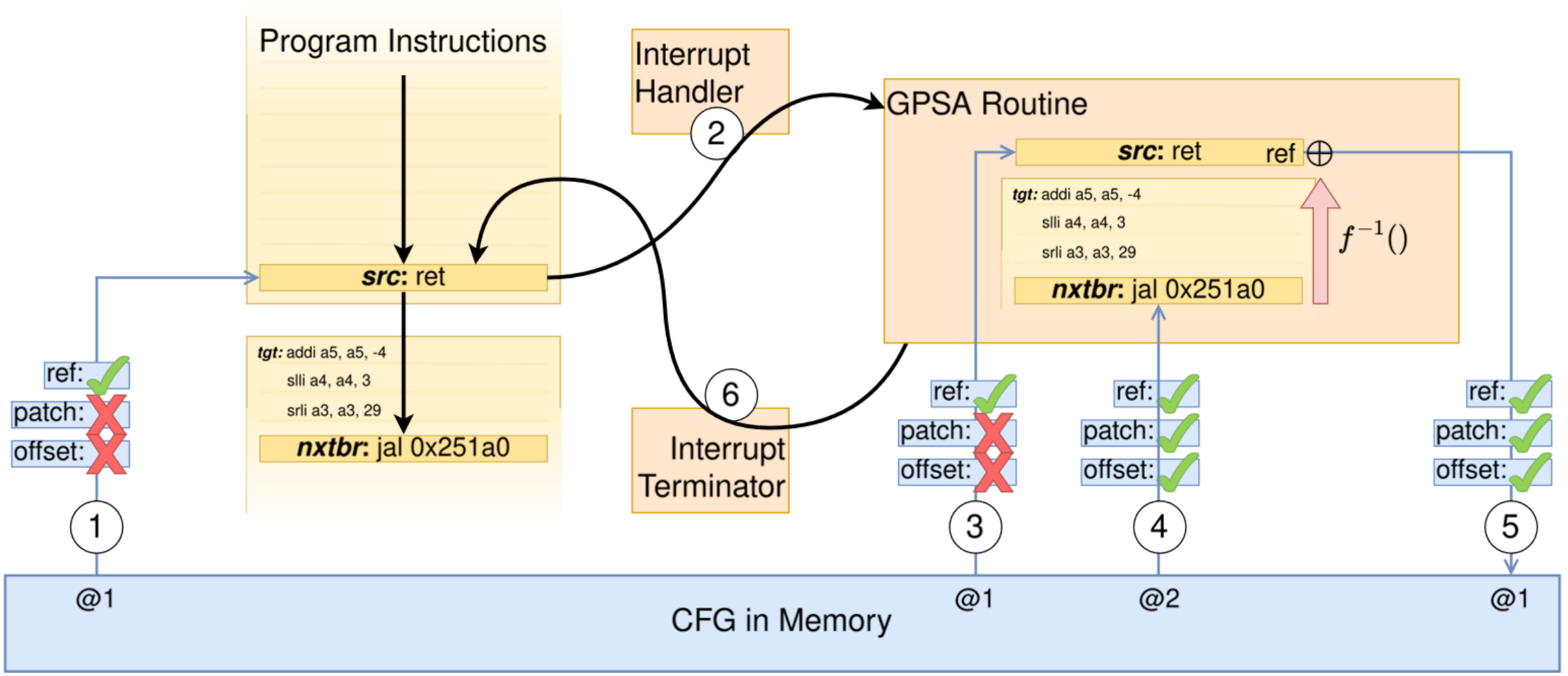
Our approach

Ahead-of-Time analysis



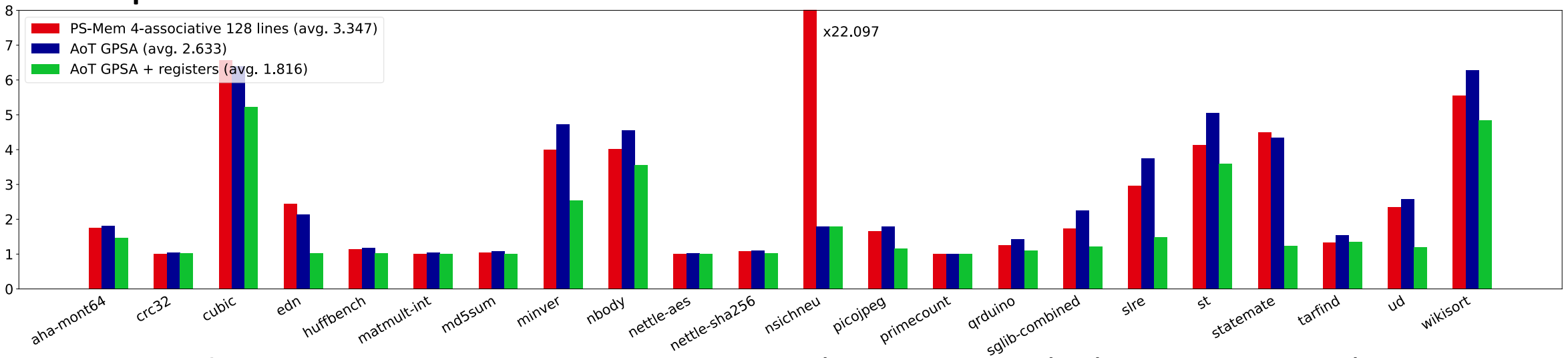
Our approach

JALR protection



Experimental Study
Results

- implemented on Comet



- performance overhead : ≈ ×3,4 (SCI-FI: ×1,25) (TMR-SW: ≥ ×3)
- area overhead : ≈ ×2 (SCI-FI: ×1,23) (TMR-HW: ×3)
- But we handle indirect jumps

S. Rokicki et al. “What You Simulate Is What You Synthesize: Designing a Processor Core from C++ Specifications”. In: ICCAD 2019. IEEE

Conclusion & Future Works

- GPSA Solution for unmodified binary

- Routine Algorithm variations to tackle edge cases
  - Taking some advance building the CFG
  - Simplifying CFG with Fallthrough instead of JAL
- Study the protection of different lengths of signatures
  - 16-bit signatures and patches could reduce area overhead