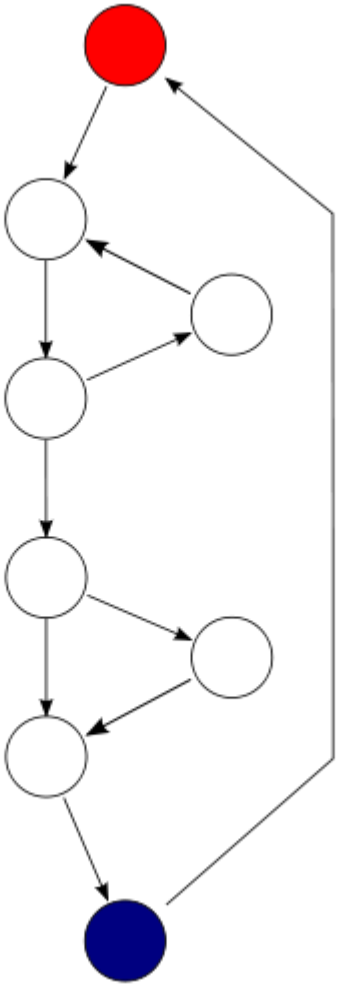
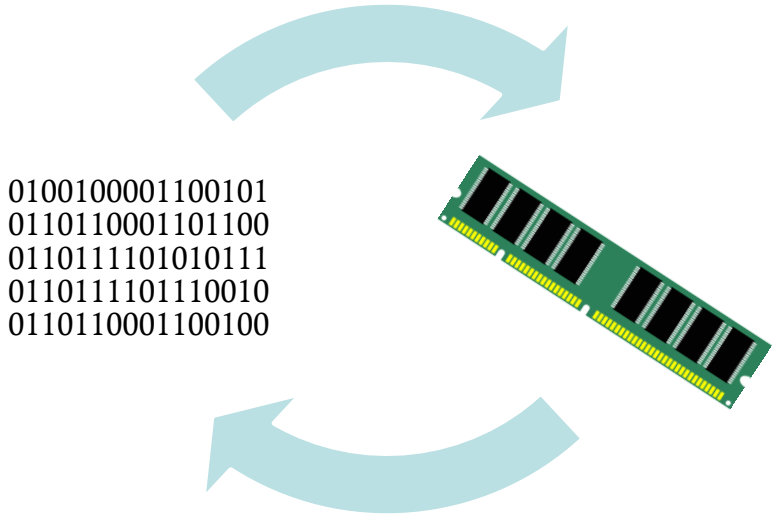
The 19th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2016)

Taming Transactions: Towards Hardware-Assisted Control Flow Integrity using Transactional Memory

Marius Muench, Fabio Pagani, Yan Shoshitaishvili, Christopher Kruegel, Giovanni Vigna, and Davide Balzarotti

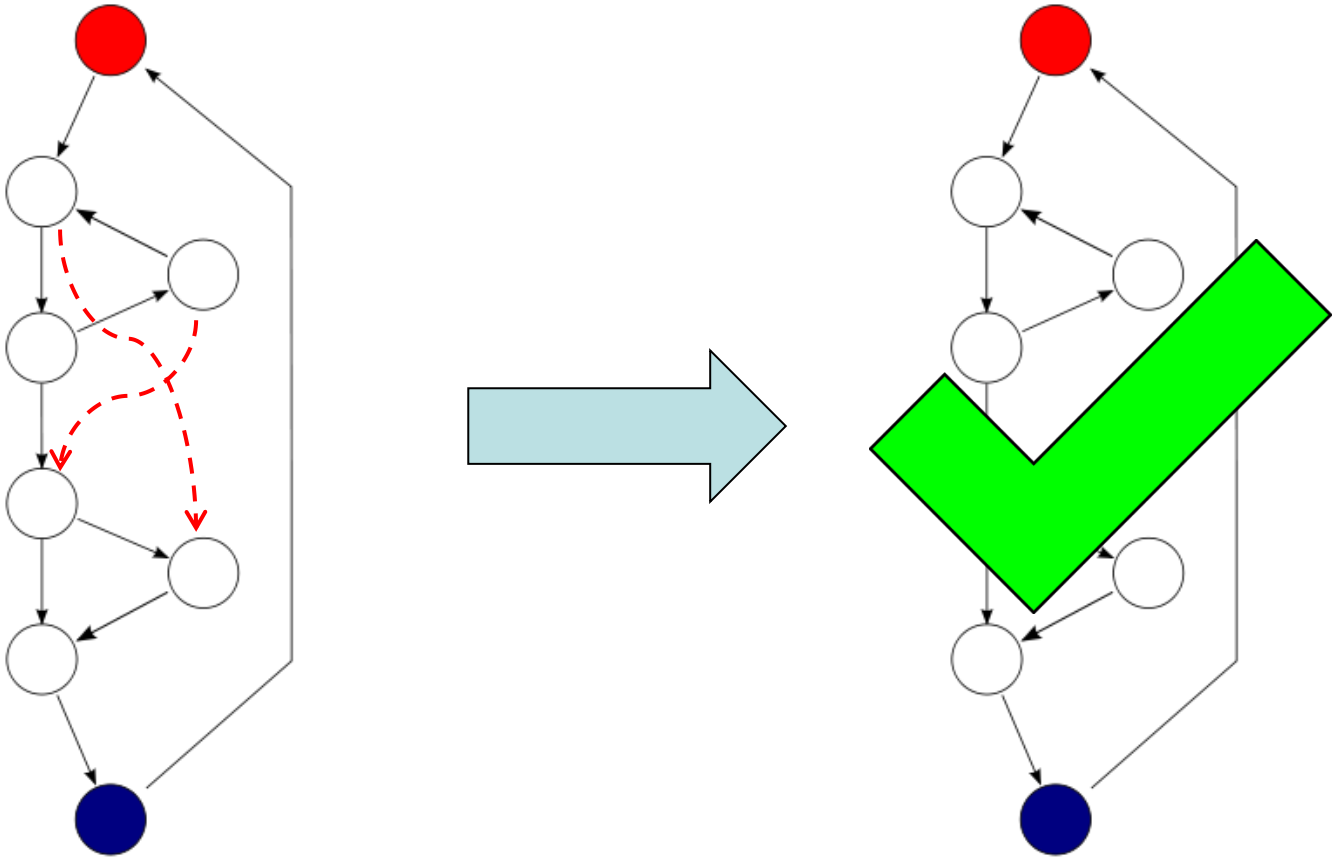
Outline



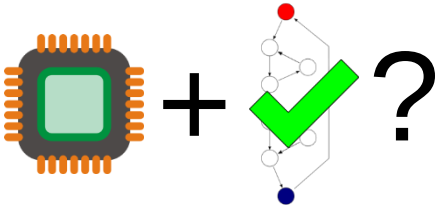

0100100001100101
0110110001101100
0110111101010111
0110111101110010
0110110001100100

Control Flow Integrity

Abadi et al., ‘05



Hardware-Assisted CFI



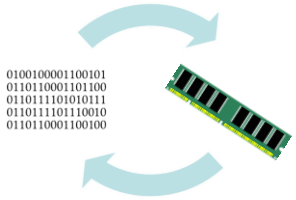
Architectural Support

- HAFIX (Dave et al., ’15)
- SOFIA (de Clarq et al., ’16)
- HCFI (Christoulakis et al., ’16)

Commodity Features

- CFImon (Xia et al., ’12)
- PathArmor (van der Veen et al., ‘15)
- CCFI (Mashtizadeh et al, ’15)

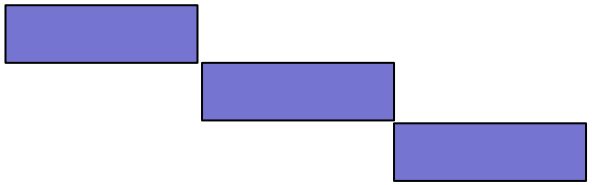
Transactional Memory



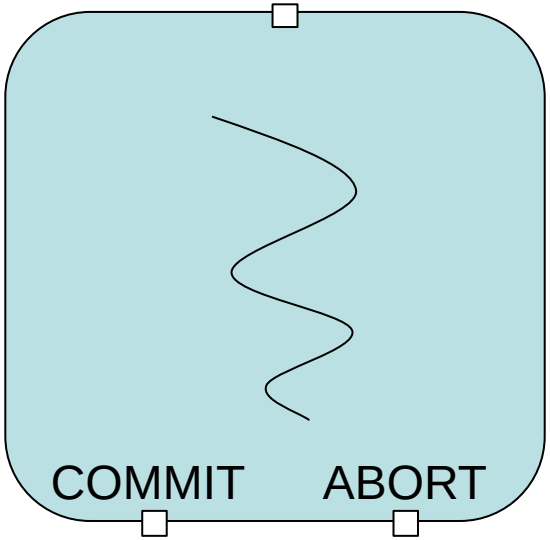
Herlihy & Moss: “Transactional Memory: Architectural Support for Lock-Free Data Structures” (1993)

Transactions

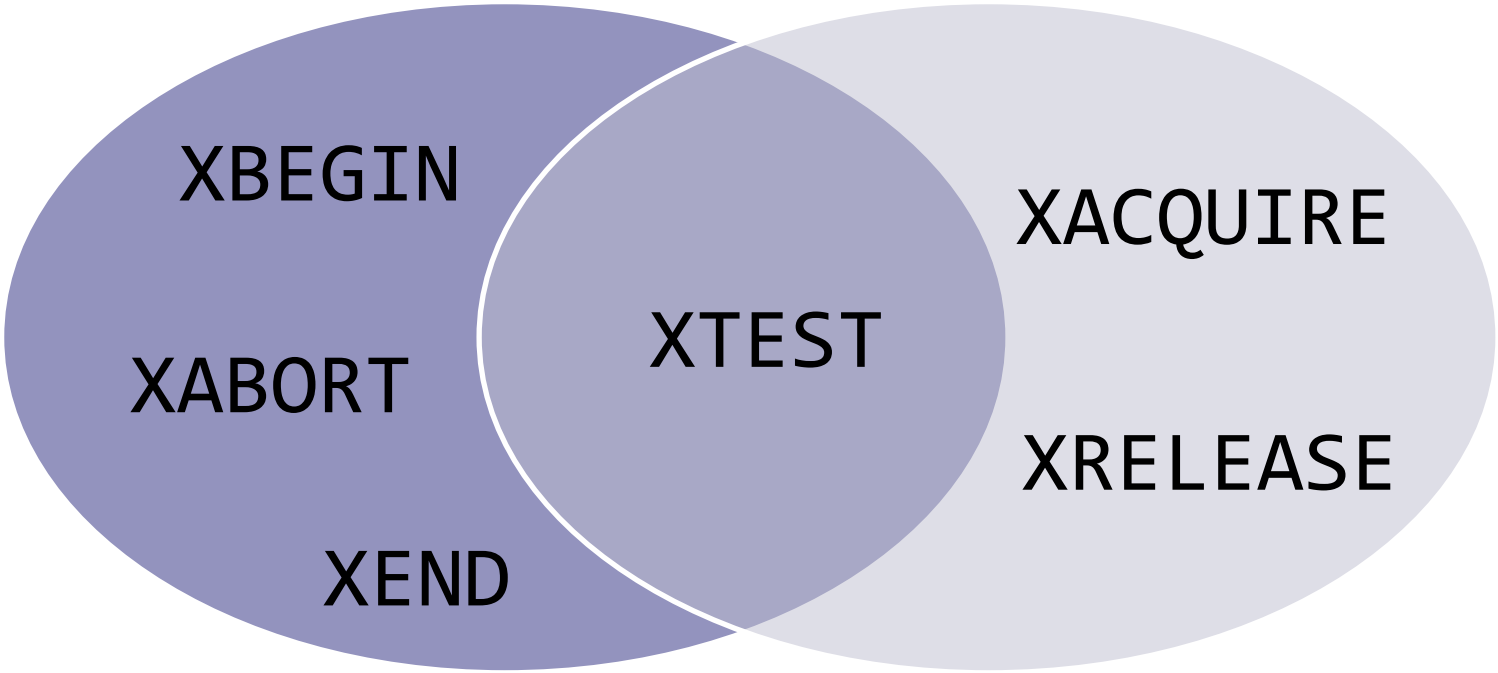
Serializability



Atomicity

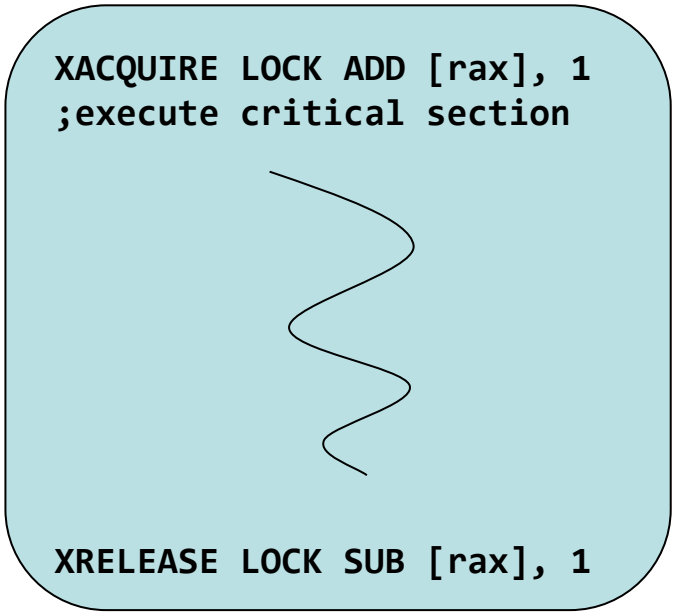


Transactional Synchronization eXtensions



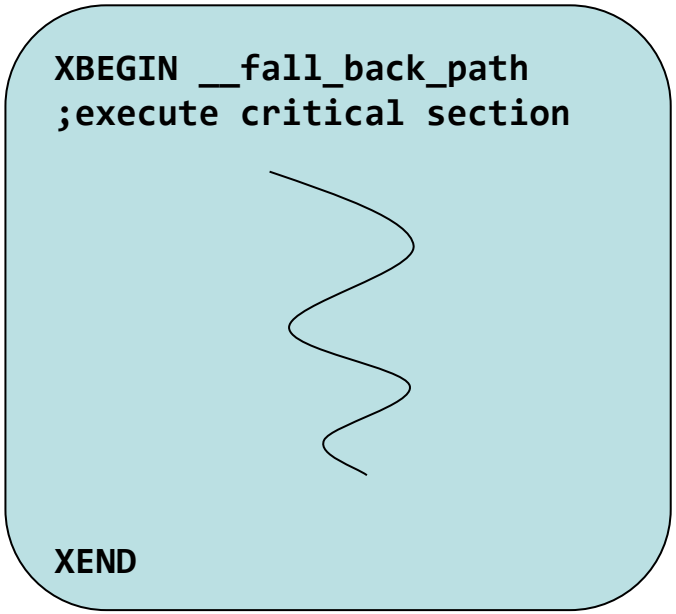
Hardware Lock Elision

- **Elides Hardware Locks**
- **Prefix Based**
  - **`XACQUIRE`, `XRELEASE`**
  - Used instead of **`LOCK`**-prefix
  - Backwards compatible
- **Failed Transaction**
  - Rollback of changed memory
  - Re-execution with traditional locking



Restricted Transactional Memory

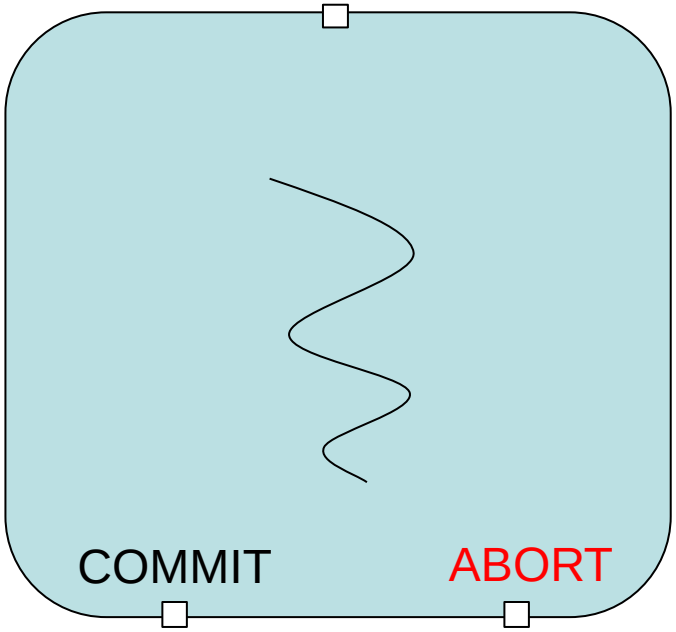
- **Marks Code Regions as Transactional**
- **Instruction Based**
  - **XBEGIN, XEND, XABORT**
  - Not backwards compatible
- **Failed Transaction**
  - Rollback of changed memory
  - Execution of fall-back path
  - Reason of failure stored in RAX



Transactional Aborts

- **Conflicts on shared data**
  - Different value of elided lock (HLE)
- **Instruction based aborts**
  - Imperative
    - **`XABORT, CPUID, PAUSE`**
  - Implementation dependent

    → Context switch sensitivity
- **Transactional Nesting Limit**



TSX-based CFI

Can we leverage Intel's TSX to enforce CFI?

TSX-based CFI

- ***Enclose*** **every control-flow transfer with a transaction**
- **Use fall-back paths to verify integrity**
- **Focus on label-based approaches**

TSX-based CFI

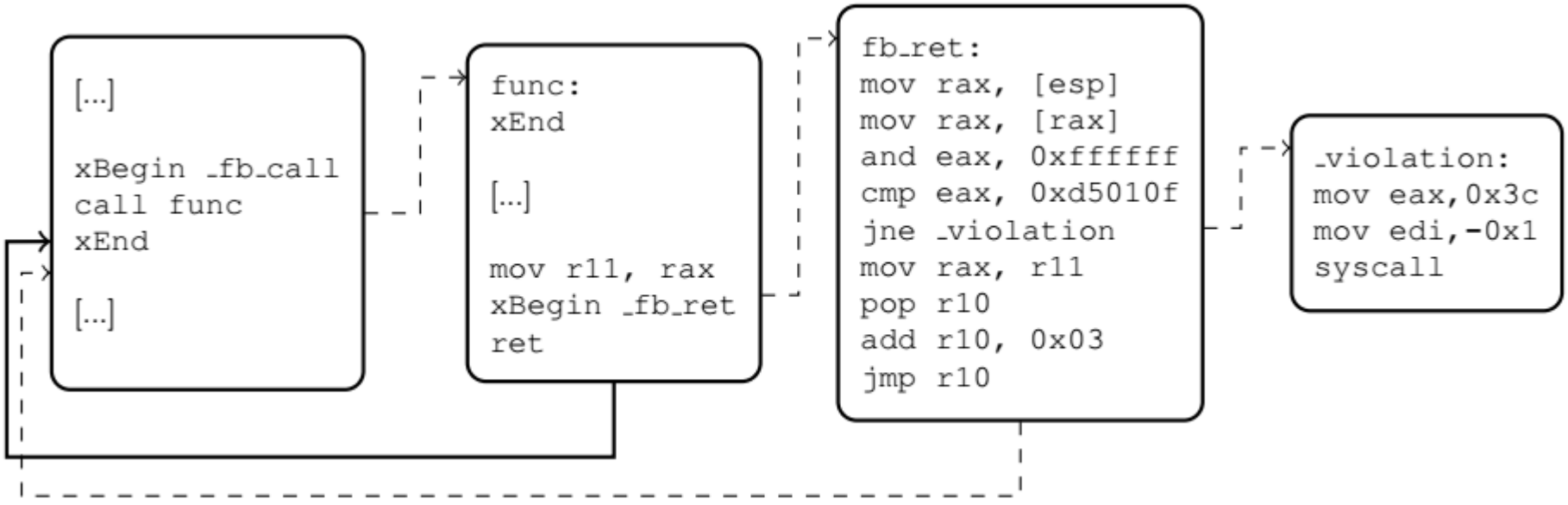
RTM

- No labels
- Clobbered `RAX` in Fall-back Path
- `XEND` outside of transaction yields SEGFAULT

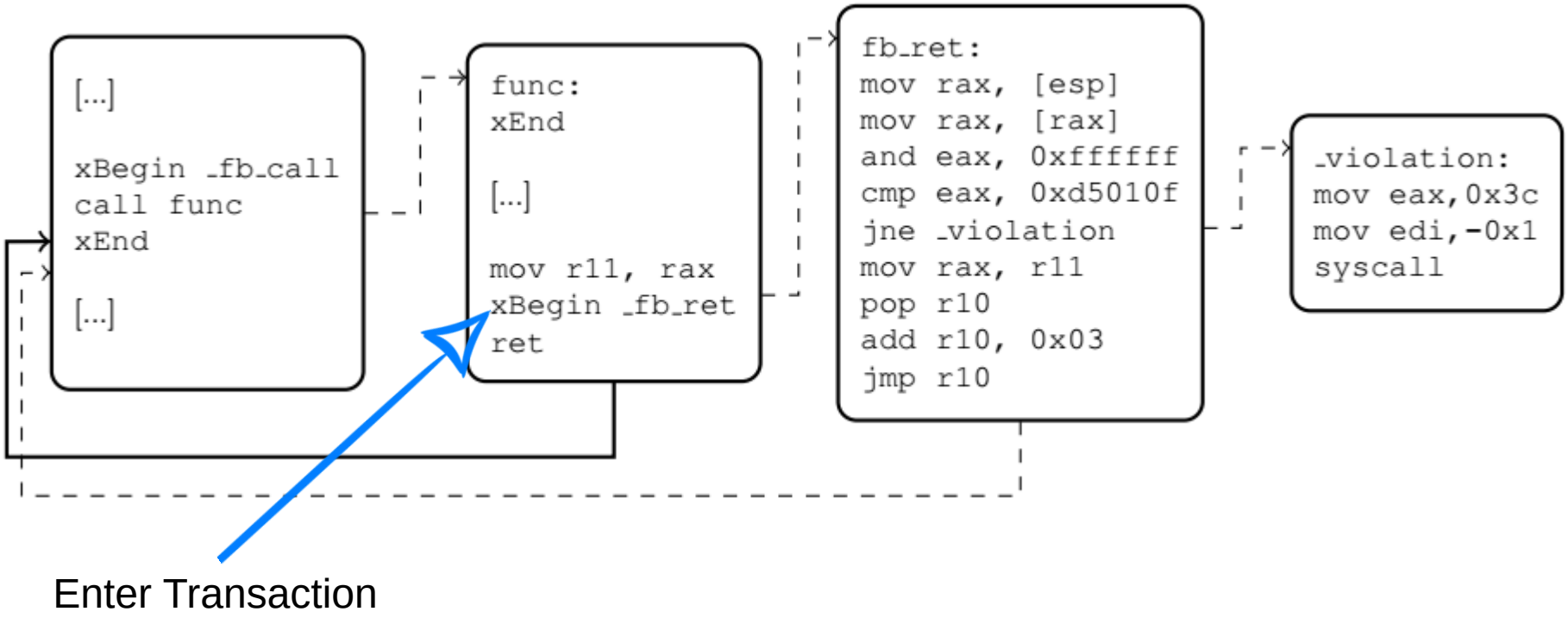
HLE

- Elided Lock Value as Label
- Virtual Fall-back path required

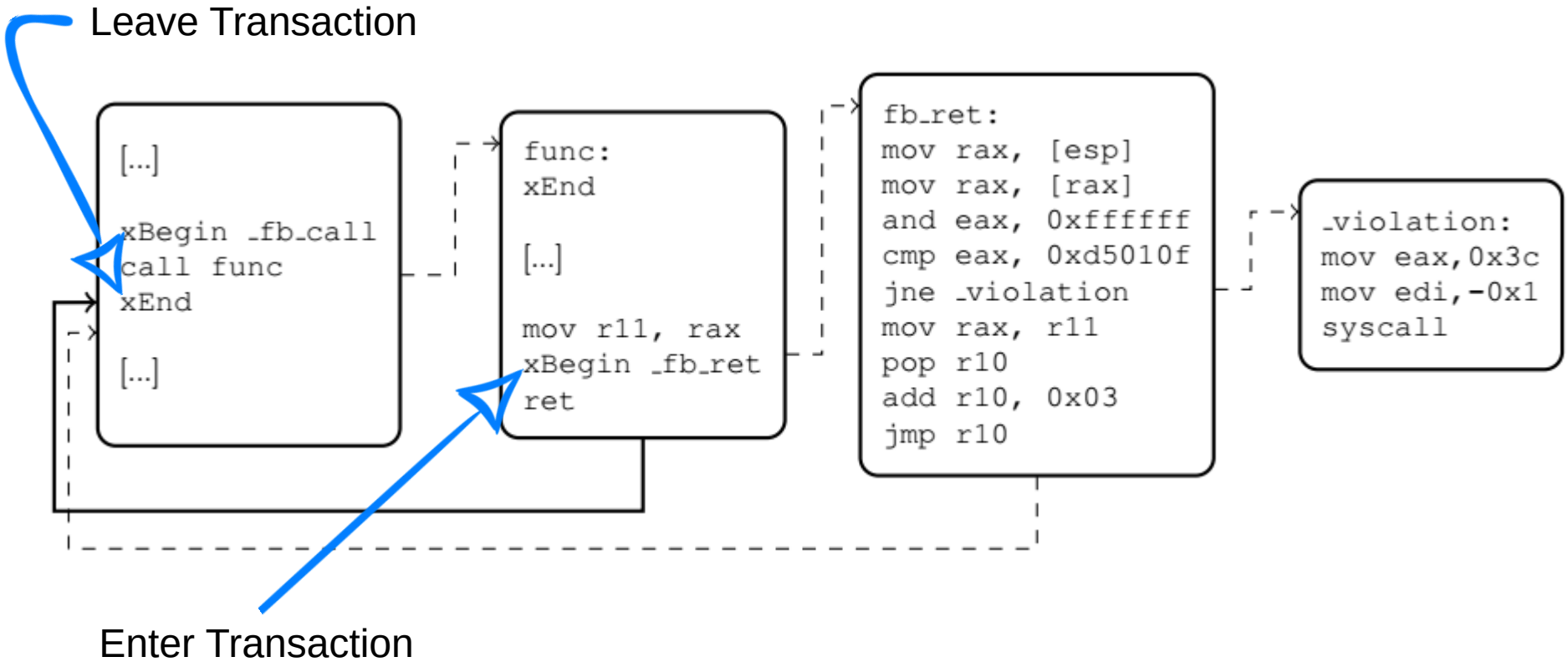
TSX-based CFI: Example



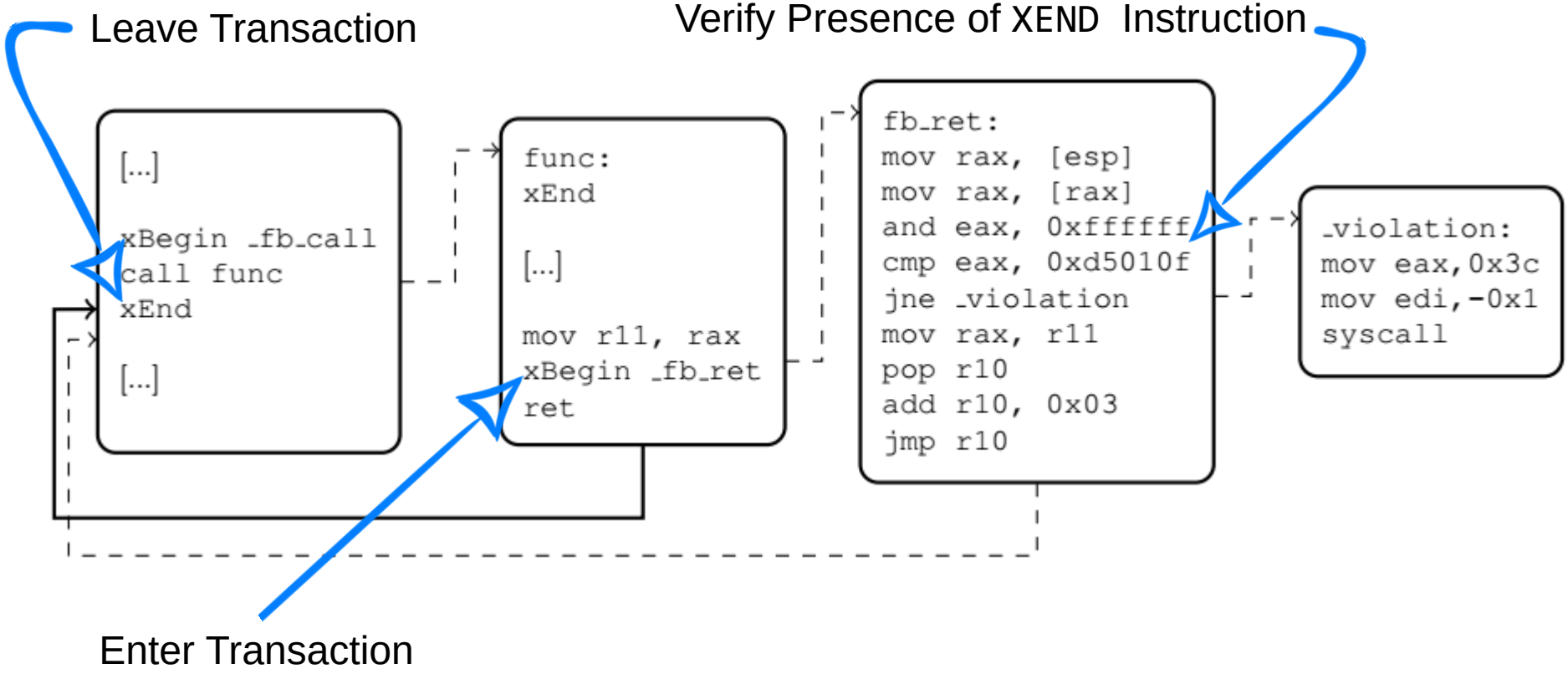
TSX-based CFI: Example



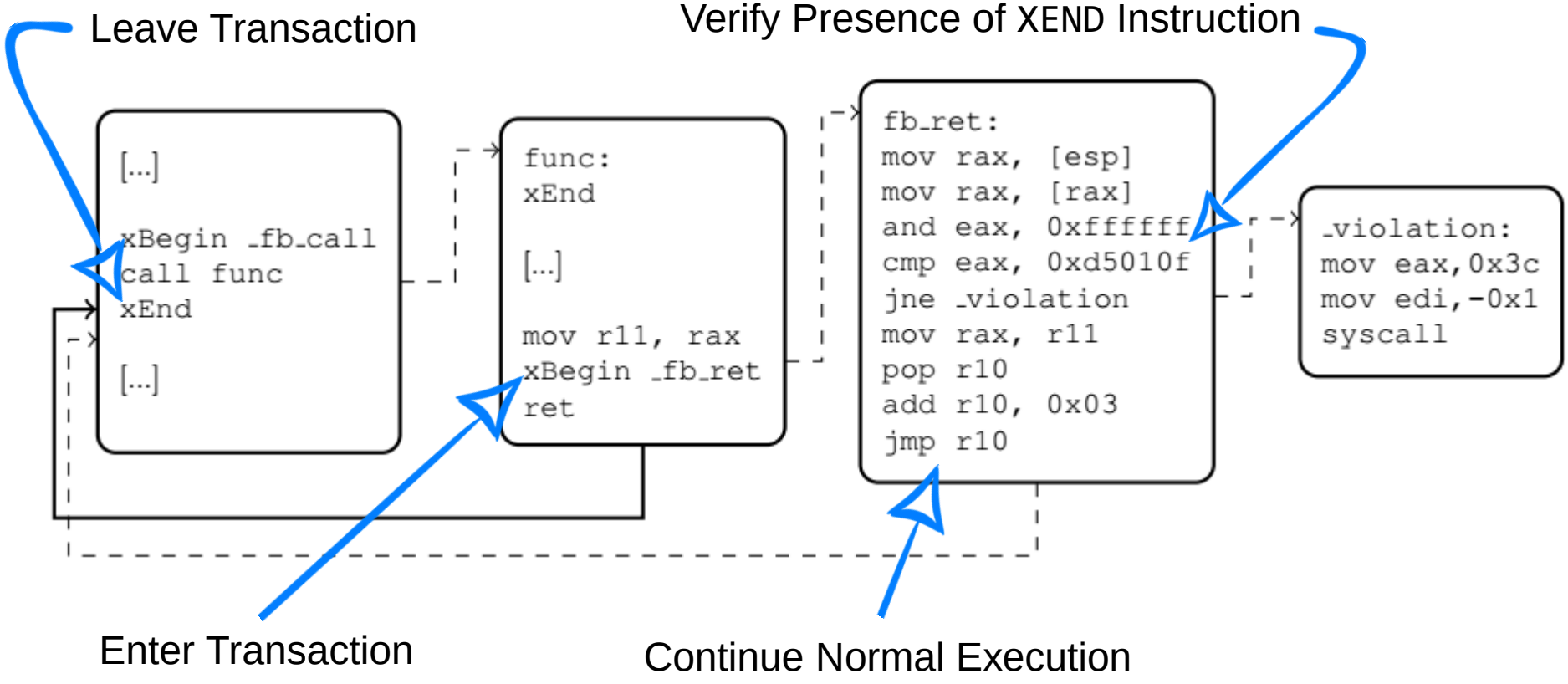
TSX-based CFI: Example



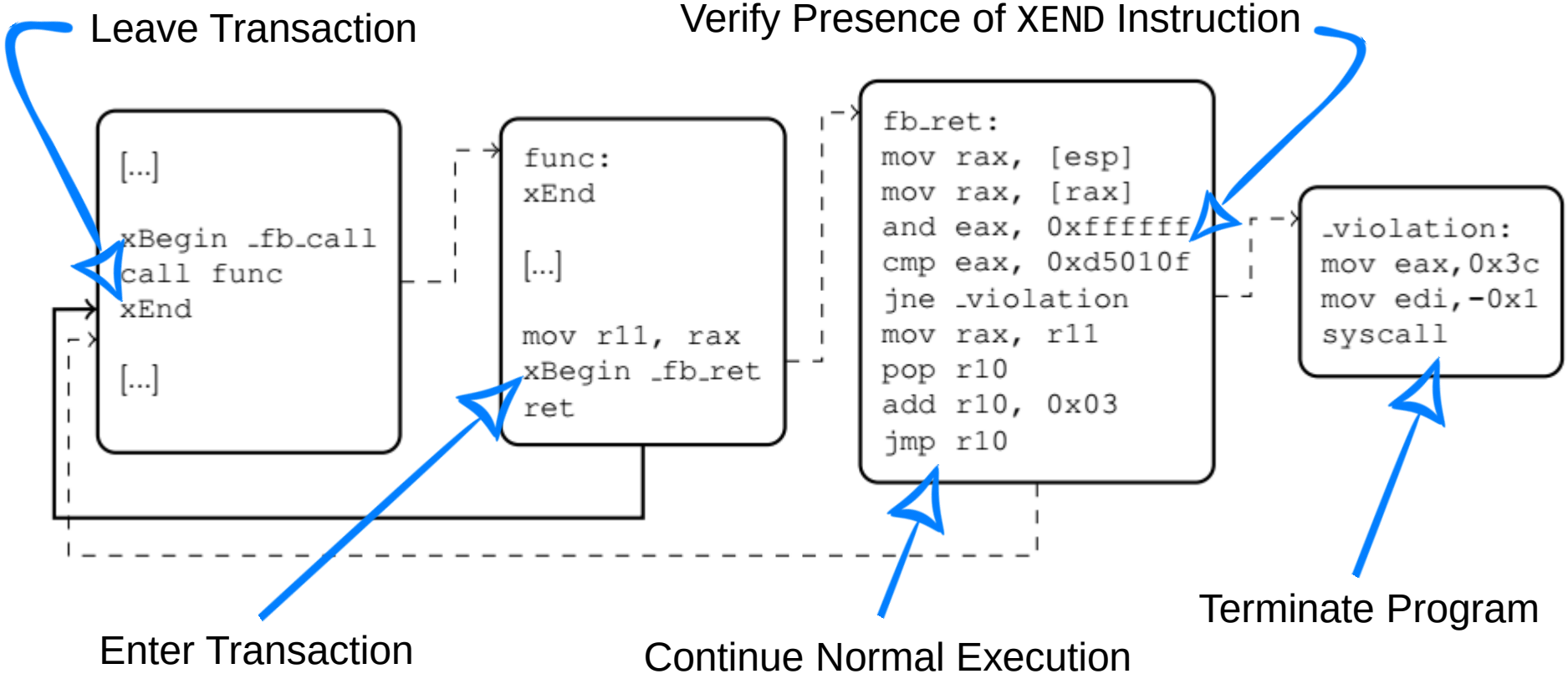
TSX-based CFI: Example



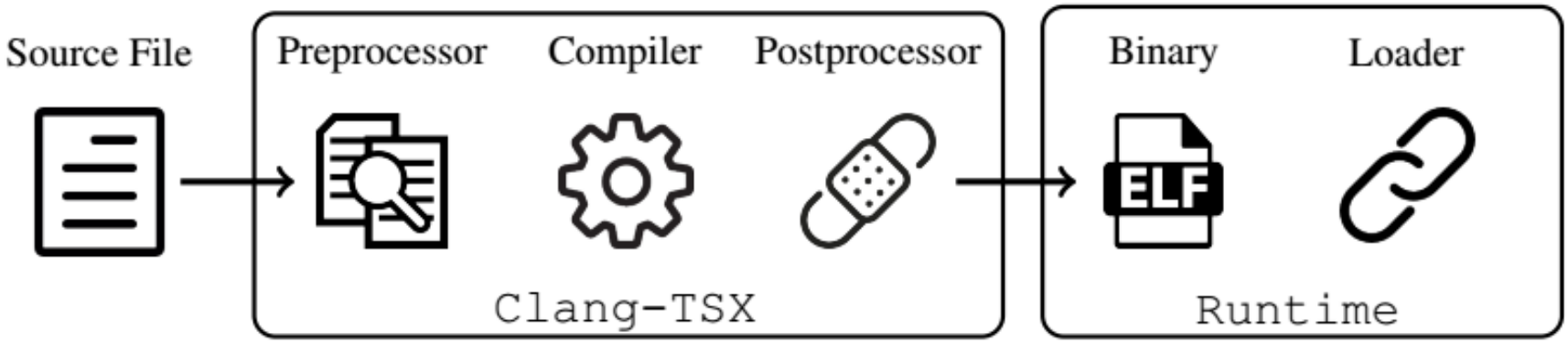
TSX-based CFI: Example



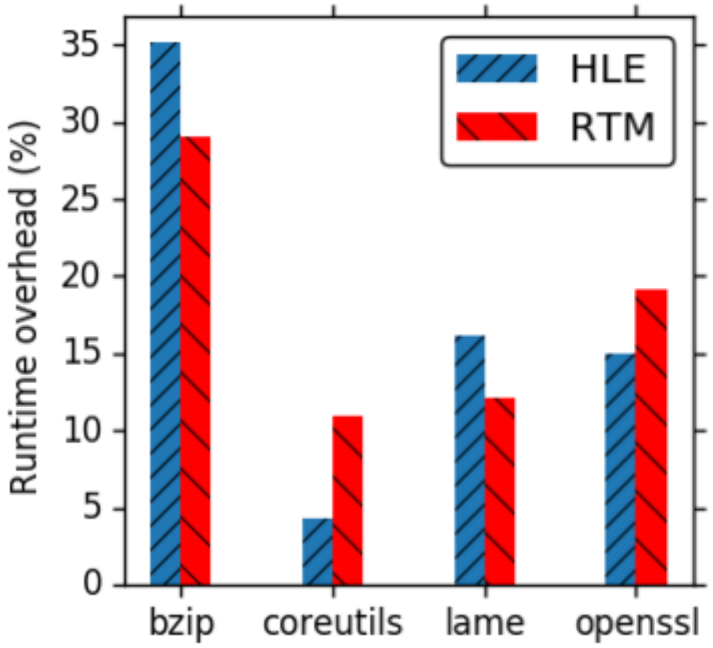
TSX-based CFI: Example

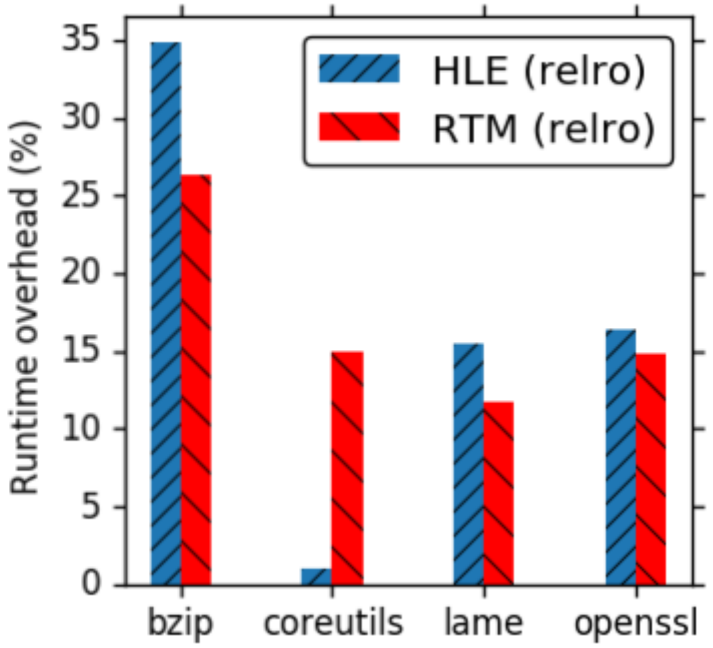


Prototype Implementation



Evaluation


Runtime overhead (%)
0
5
10
15
20
25
30
35
HLE
RTM
bzip
coreutils
lame
openssl


Runtime overhead (%)
0
5
10
15
20
25
30
35
HLE (relro)
RTM (relro)
bzip
coreutils
lame
openssl

Conclusion

- **Can we leverage Intel's TSX to enforce CFI?**
  - **Yes!**

- **We proposed two methods for CFI enforcement:**
  - RTM-based
  - HLE-based
- **Interesting side-effects**
- **Mediocre performance (for now)**
- **Implementation will be released on github:**
  - https://github.com/eurecom-s3/tsxcfi

Intel's Control Flow Enforcement Technology

- **Preview released in June 2016**
- **Backward-Edges: Shadow Stack**
- **Forward-Edges: ENDBRANCH Instruction**
  - Indirect branch forces CPU to enter WAIT_FOR_ENDBRANCH state
  - Similar to RTM-based CFI
- **No hardware available yet!**

This Slide is Intentionally Left Blank

Bonus-Example: TSX-based CFI (HLE)