

Demystifying and Mitigating Cross-Layer Deficiencies of Soft Error Protection in Instruction Duplication

Zhengyang He[†]

Yafan Huang[†]

Hui Xu[◇]

Dingwen Tao[‡]

Guanpeng Li[†]

[†] The University of Iowa

[◇] Fudan University

[‡] Indiana University Bloomington

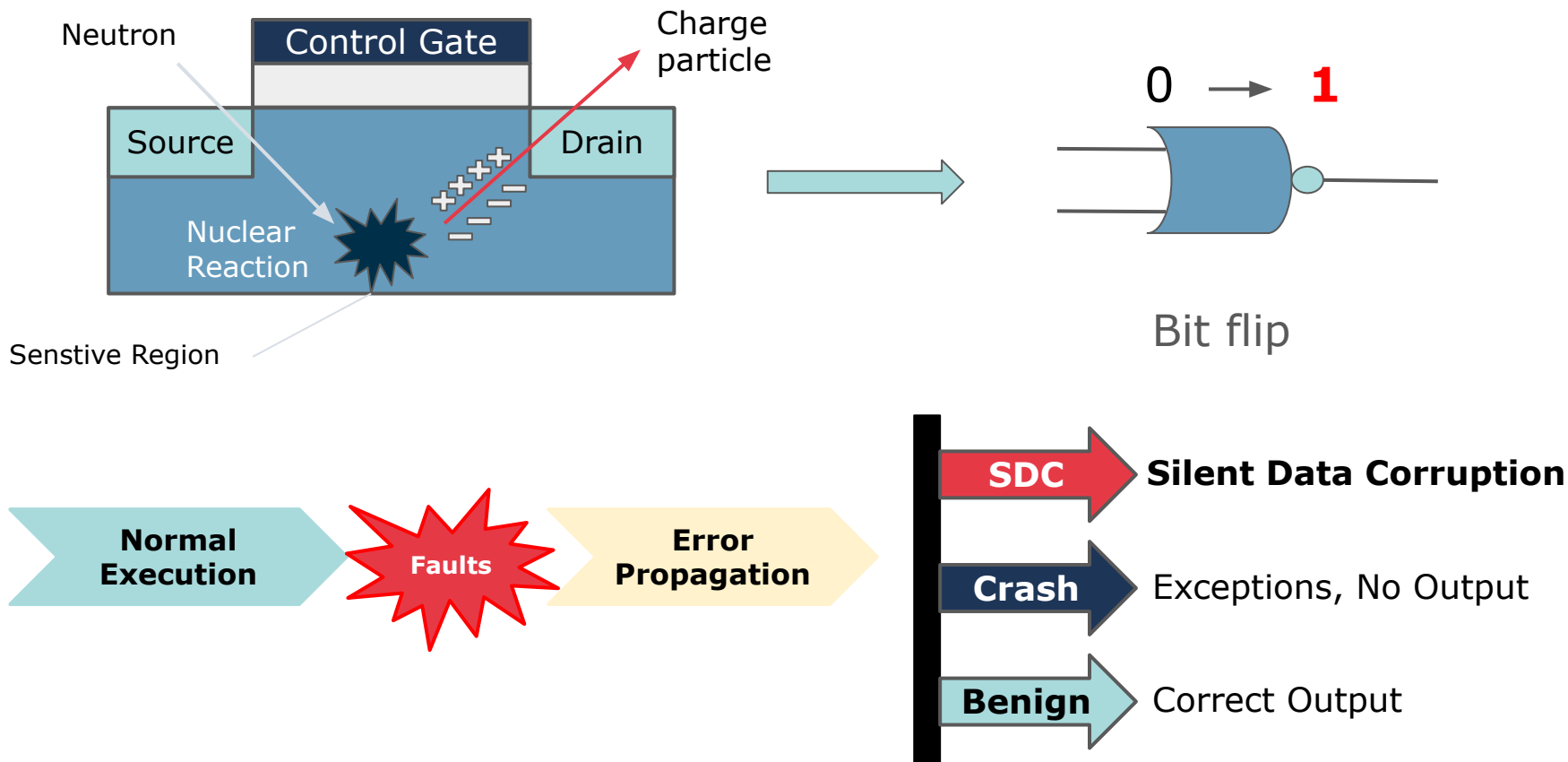
IOWA



復旦大學
FUDAN UNIVERSITY

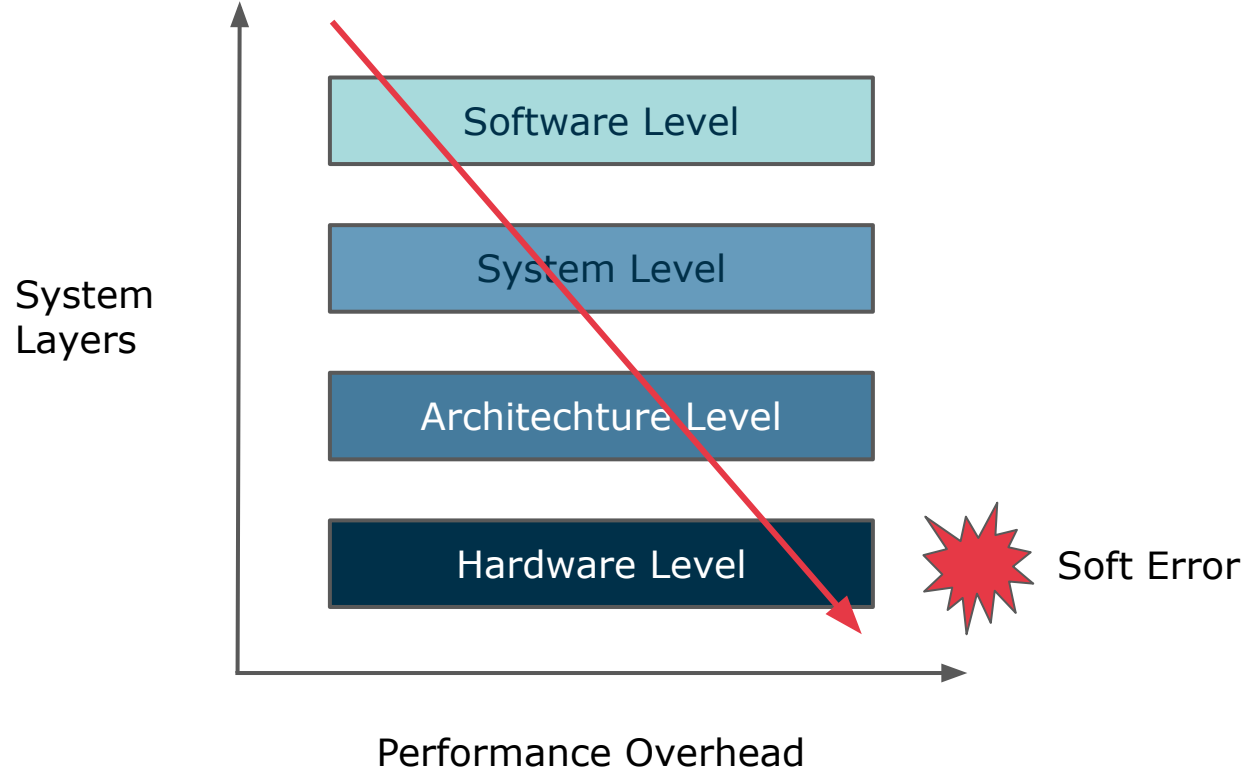


Soft Errors



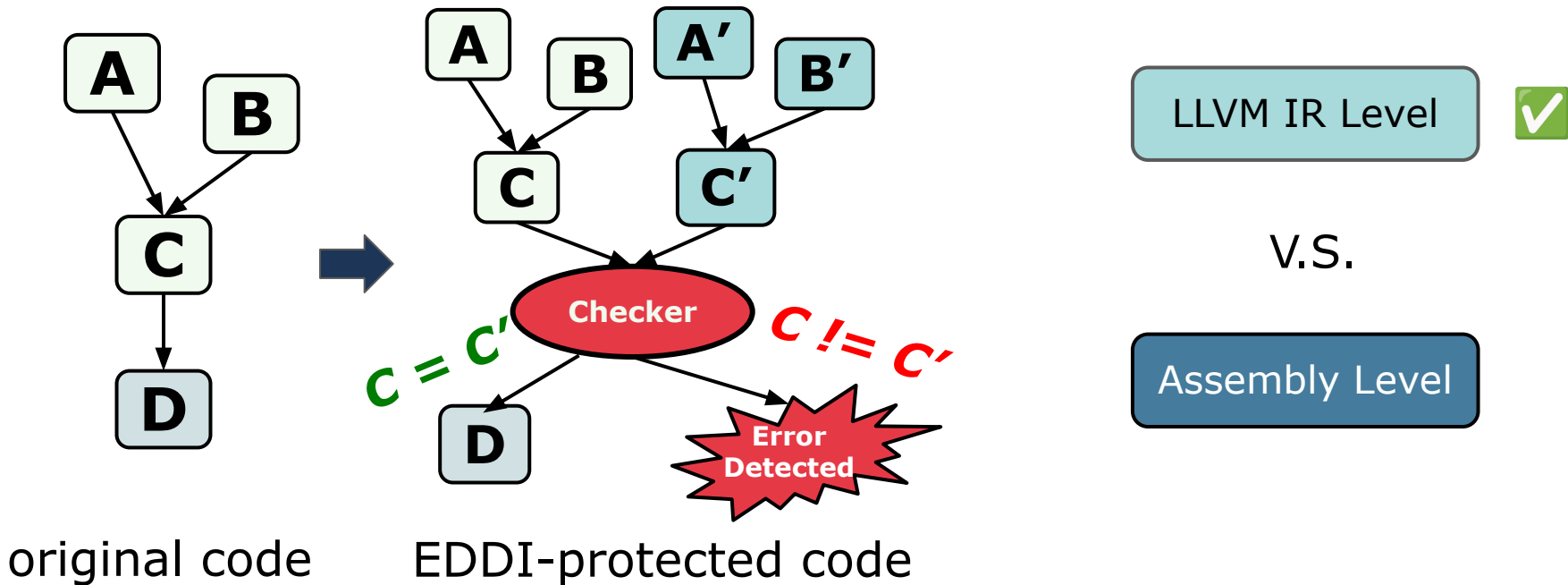
Software Solutions

- **Software solution** is more flexible and cost-effective.



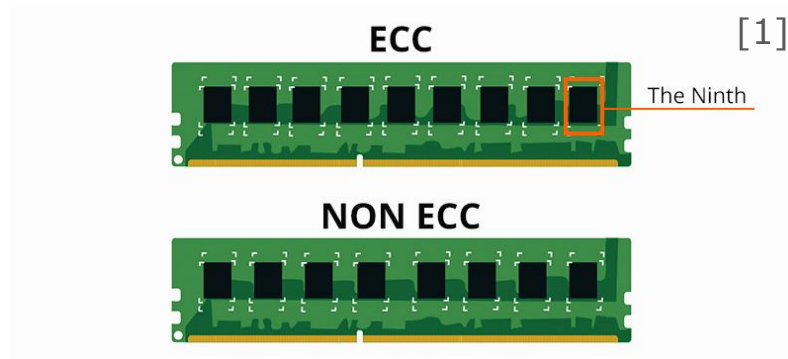
Error Detection by Duplicating Instructions (EDDI)

- **EDDI** duplicates instruction at *compile time* and detects errors at *run time*.
- Compiler-level transformation, hence **program-agnostic**.



Fault Model

- Single-bit flip, which is accurate enough to evaluate SDC
- Errors in computation units/data path
- One fault per program execution
- Memory errors can be protected by ECC, so do not consider



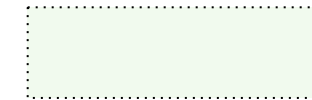
Fault Injection Methodology

- **IR:** Instructions that contains return value



```
%3 = icmp slt i32 %1, %2  
%4 = load i32* %1, align 4  
%5 = mul i64 1, %4
```

...



IR level Code



Assembly Code

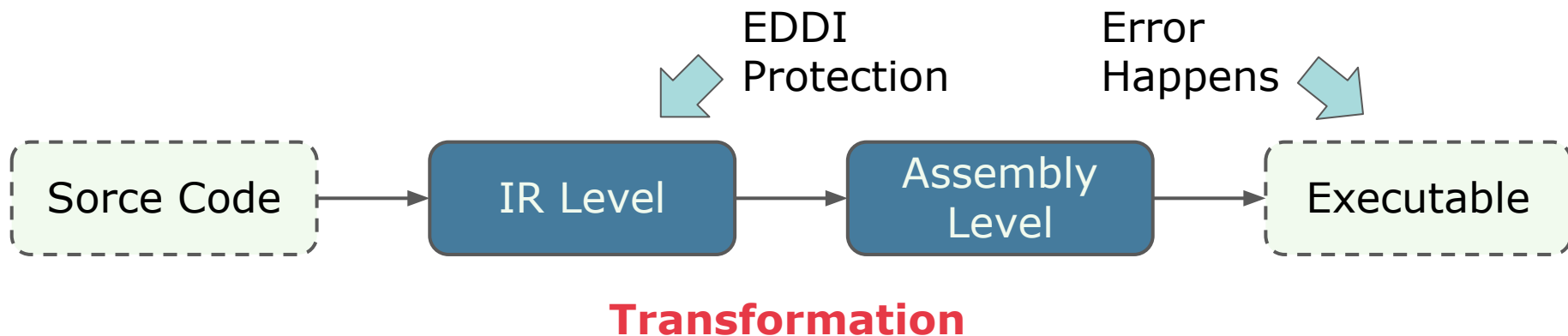
- **Assembly:** Instructions whose computation destination is a register



```
test  $0x1,%al  
sub   %ecx,%eax  
mov   -0x40(%rbp),%rax
```

...

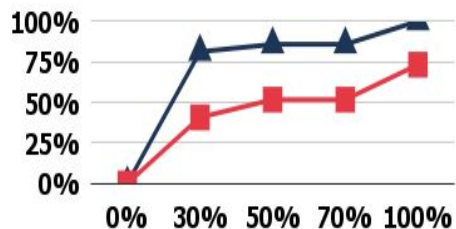
Motivation: Code transformation from IR to Assembly



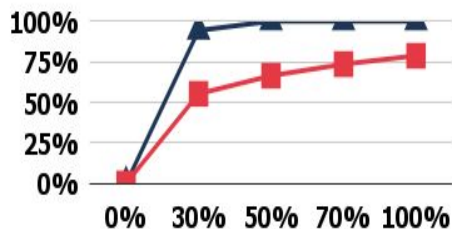
IR-level EDDI may not provide full protection on Assembly-level!

Motivation: Code transformation from IR to Assembly

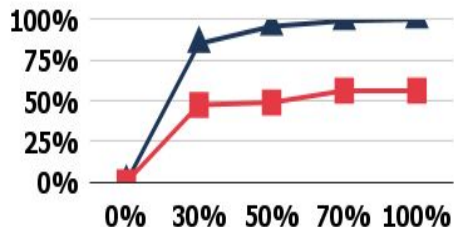
- Compare of: SDC coverage evaluation at IR and Assembly level



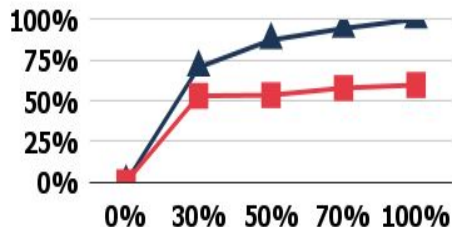
(a) Backprop



(b) Pathfinder



(c) Patricia



(d) Basicmath

X-axis : Protection level

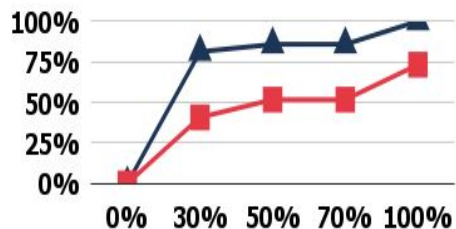
Y-axis : SDC coverage

— Assembly level

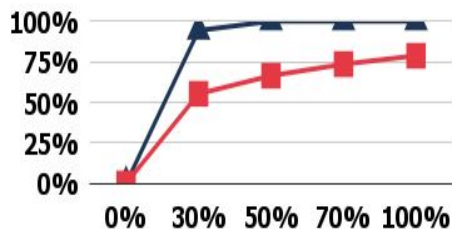
— IR level

Motivation: Code transformation from IR to Assembly

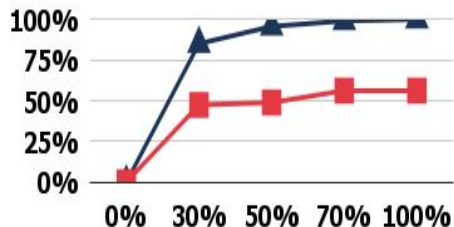
- Compare of: SDC coverage evaluation at IR and Assembly level



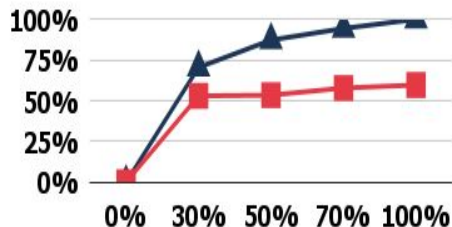
(a) Backprop



(b) Pathfinder



(c) Patricia



(d) Basicmath

X-axis : Protection level

Y-axis : SDC coverage

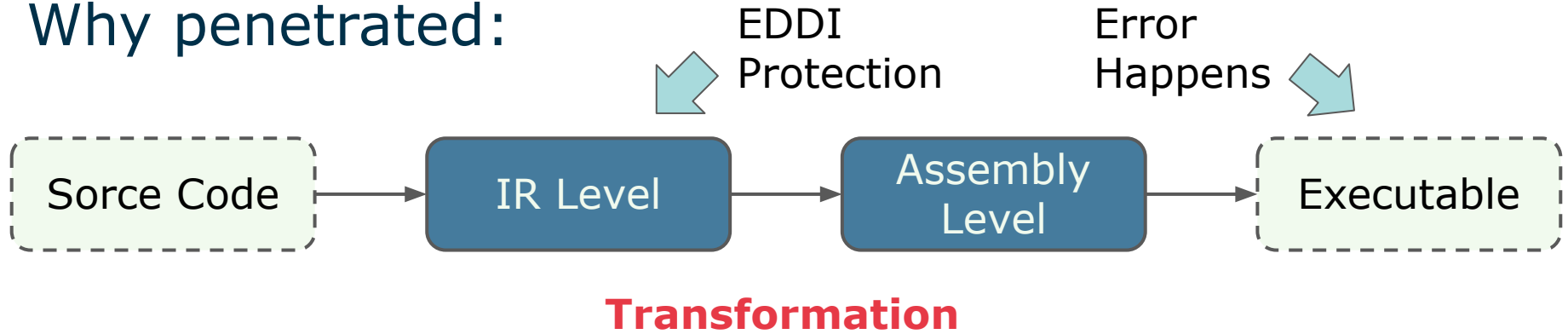
- **Application specific** SDC coverages
- **SDC coverage gap** in IR and Assembly protection

— Assembly level

— IR level

LLVM SDC coverage often **over-estimates** Assembly level benefits.

Why penetrated:



Mapping penetration

2.5%

Comparison penetration

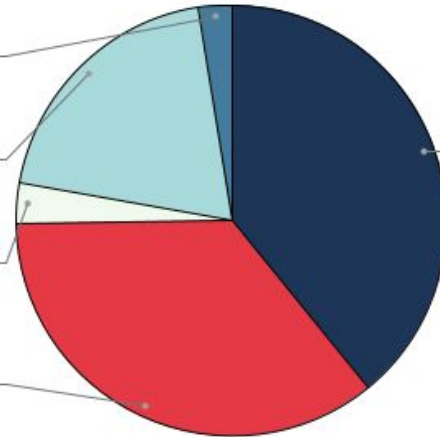
19.7%

Call penetration

3.1%

Branch penetration

35.7%

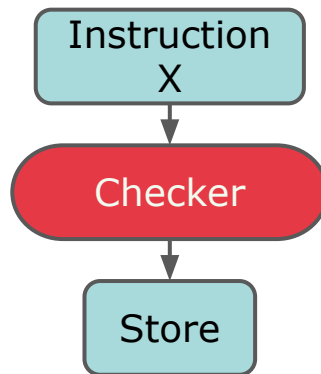
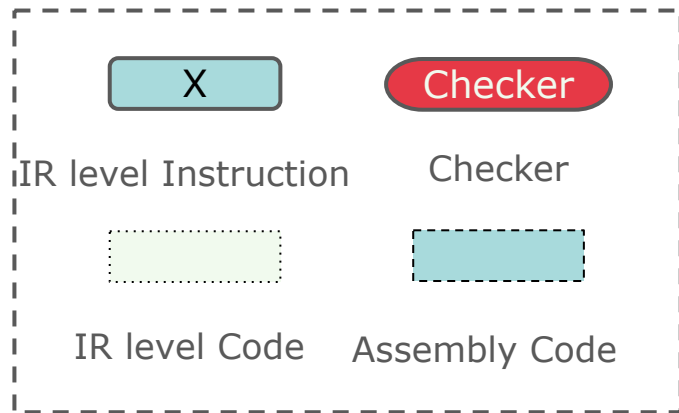


Store penetration

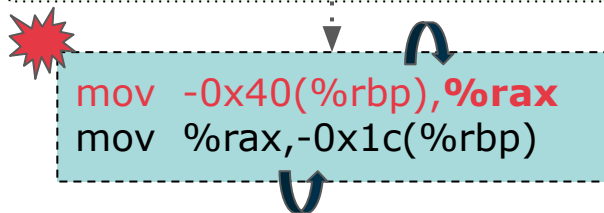
39.1%

Deficiency Cases

Why penetrated: 1. Store Penetration

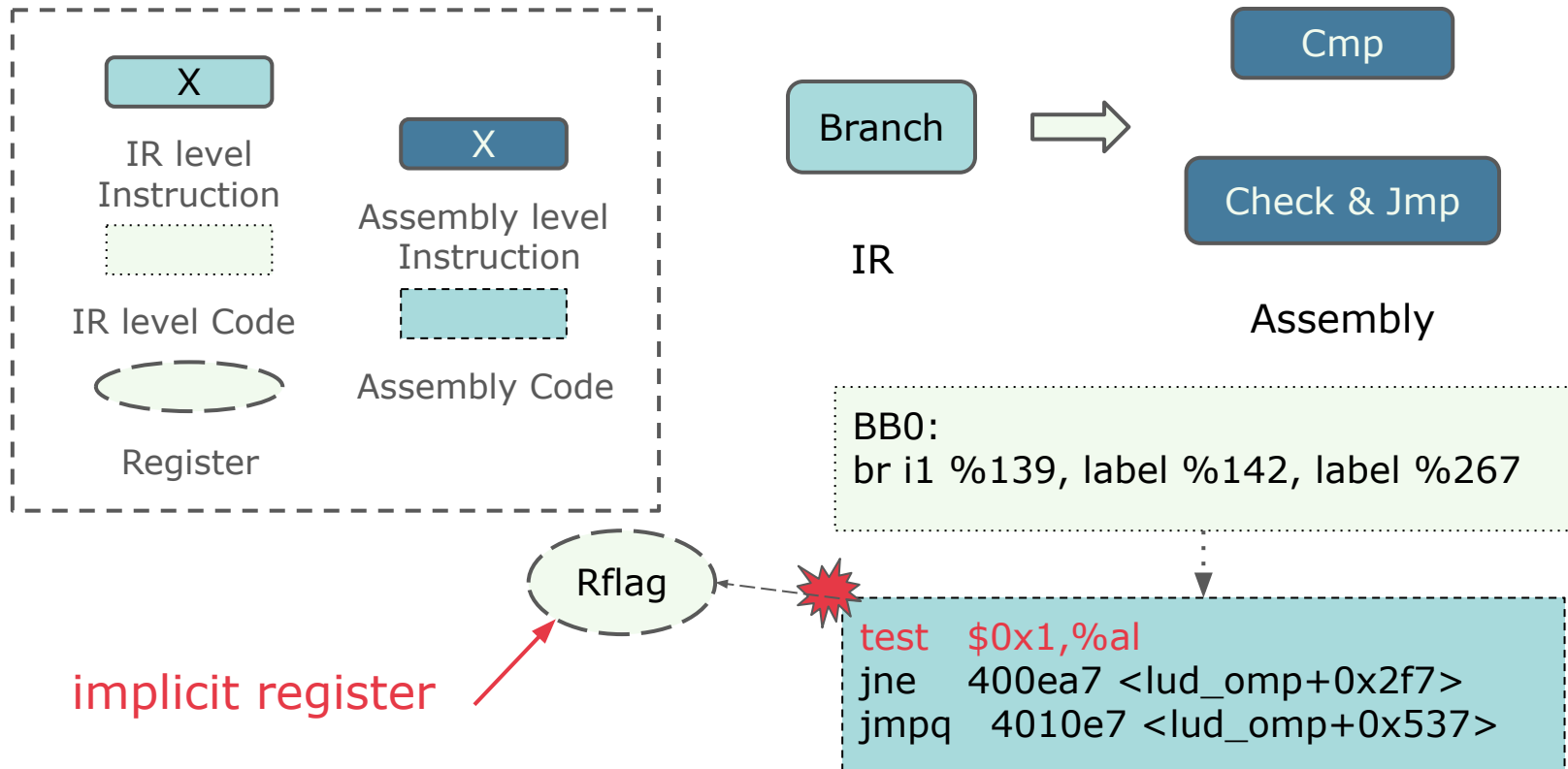


store float %92, float* %sum

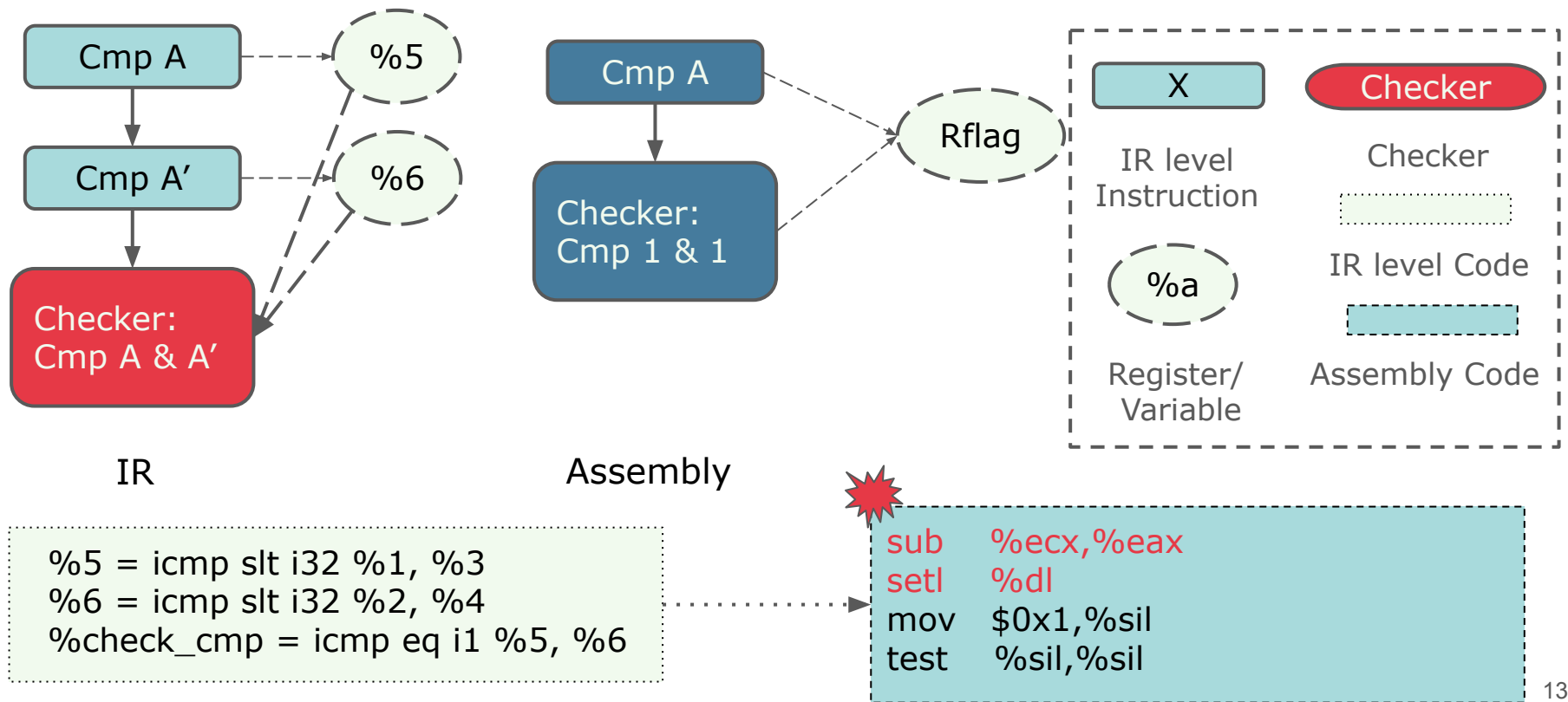


implicit register

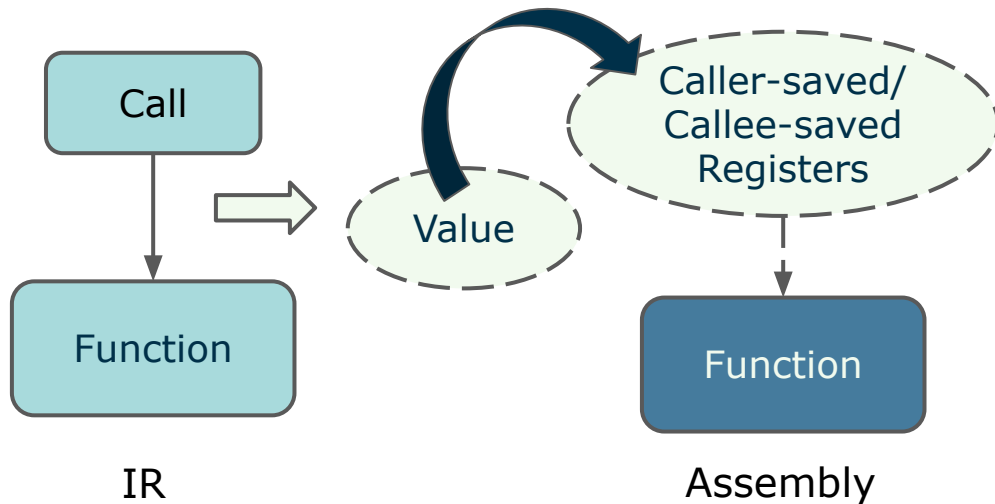
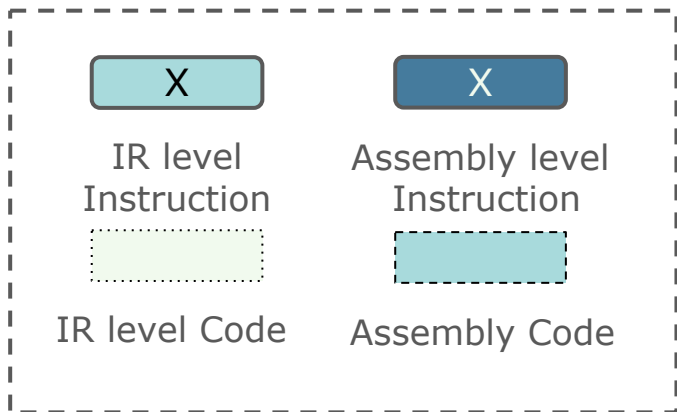
Why penetrated: 2. Branch Penetration



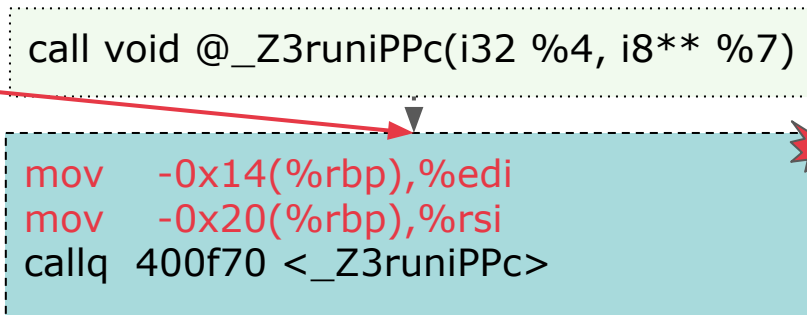
Why penetrated: 3. Comparison Penetration



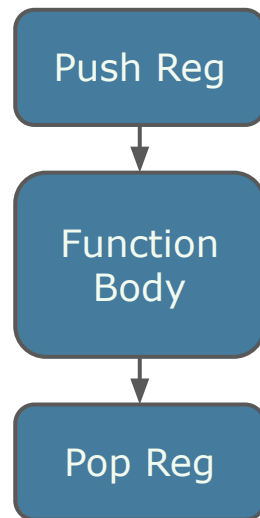
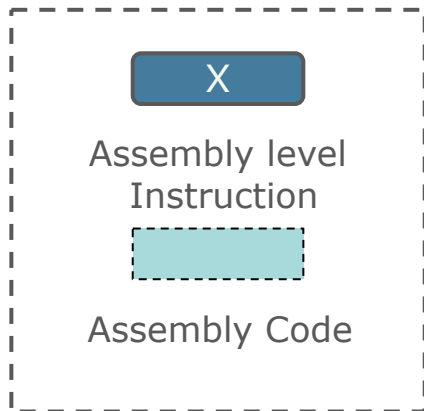
Why penetrated: 4. Call Penetration




implicit register



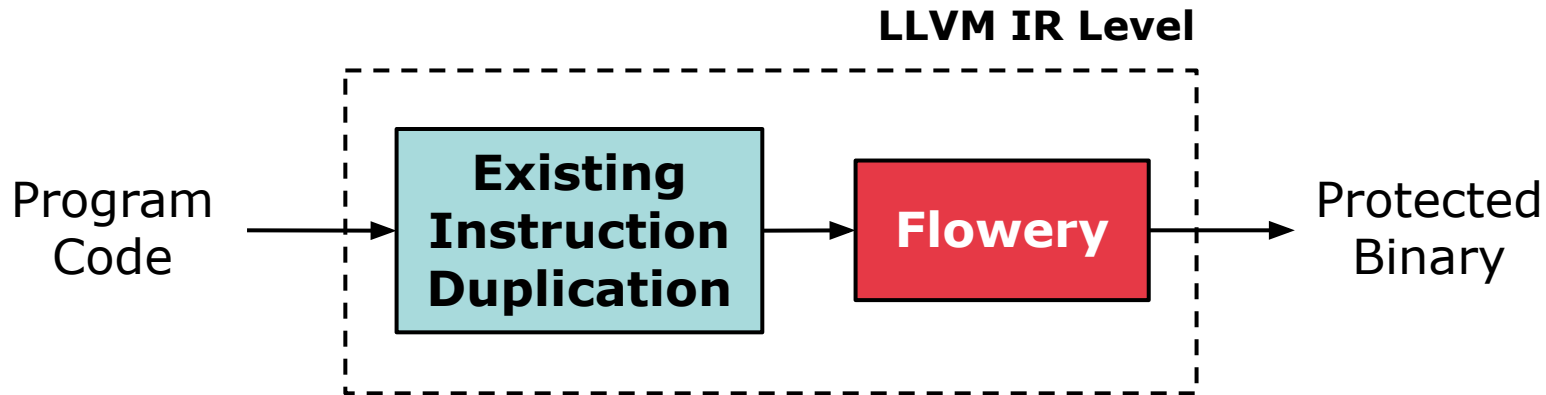
Why penetrated: 5. Mapping Penetration



 `push %rbp`
... ; function body
`pop %rbp`
`retq`

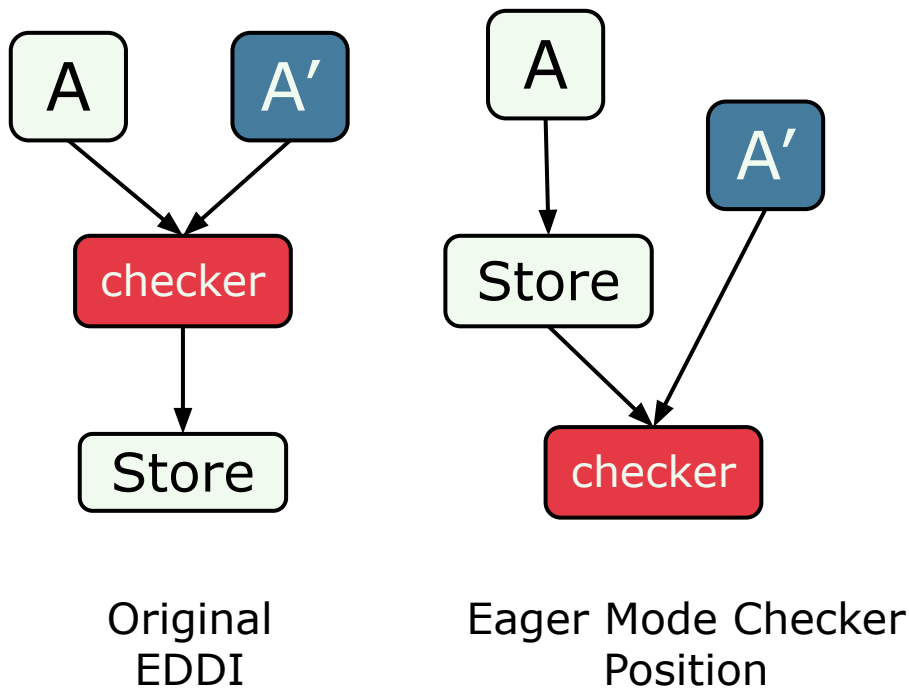
Flowery

- Goal: **Boost** assembly level protection via IR level
- Observations:
 - IR level offers simple instruction **tracking and modification**
 - IR level can alter assembly-level **register allocation**



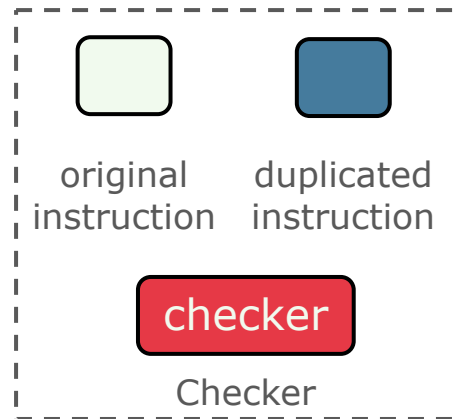
Flowery

□ Eager Mode of Store



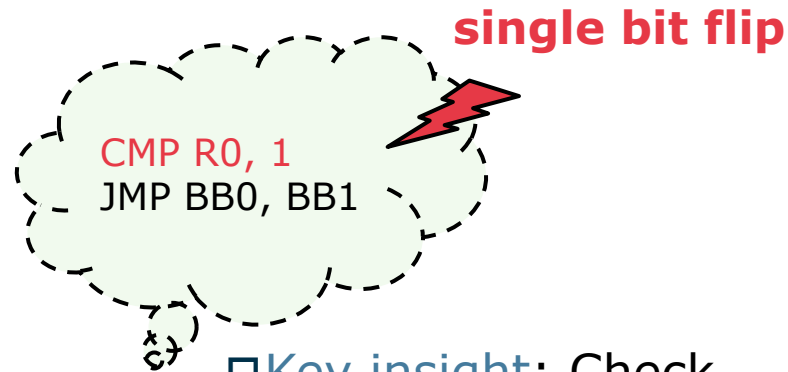
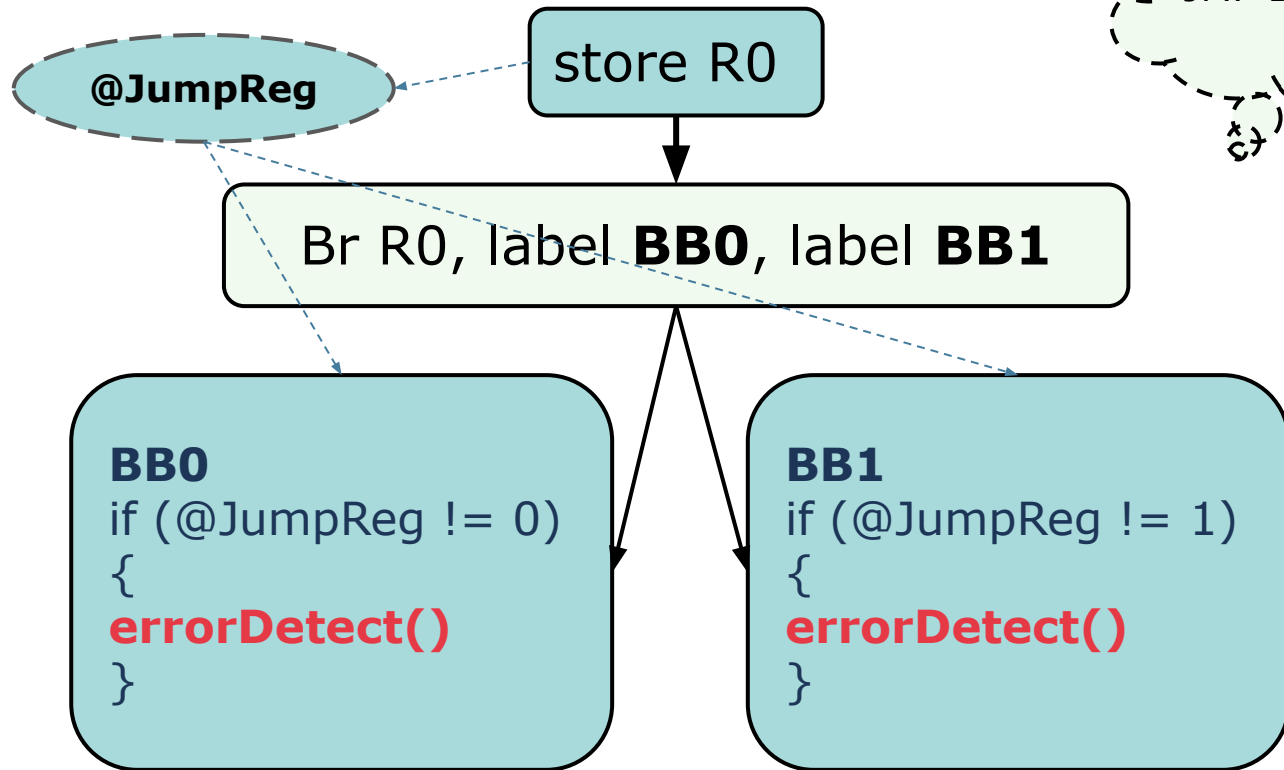
□ **Key insight:** Reallocate the store instruction.

□ **Result:** Move the temporary value to a register without extra computations.



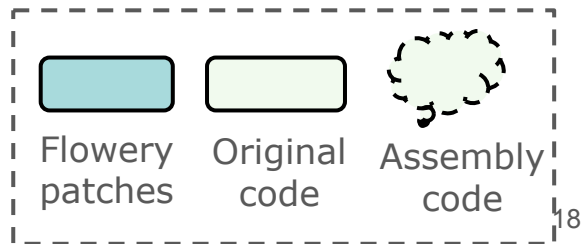
Flowery

□ Postponed Branch Condition Check



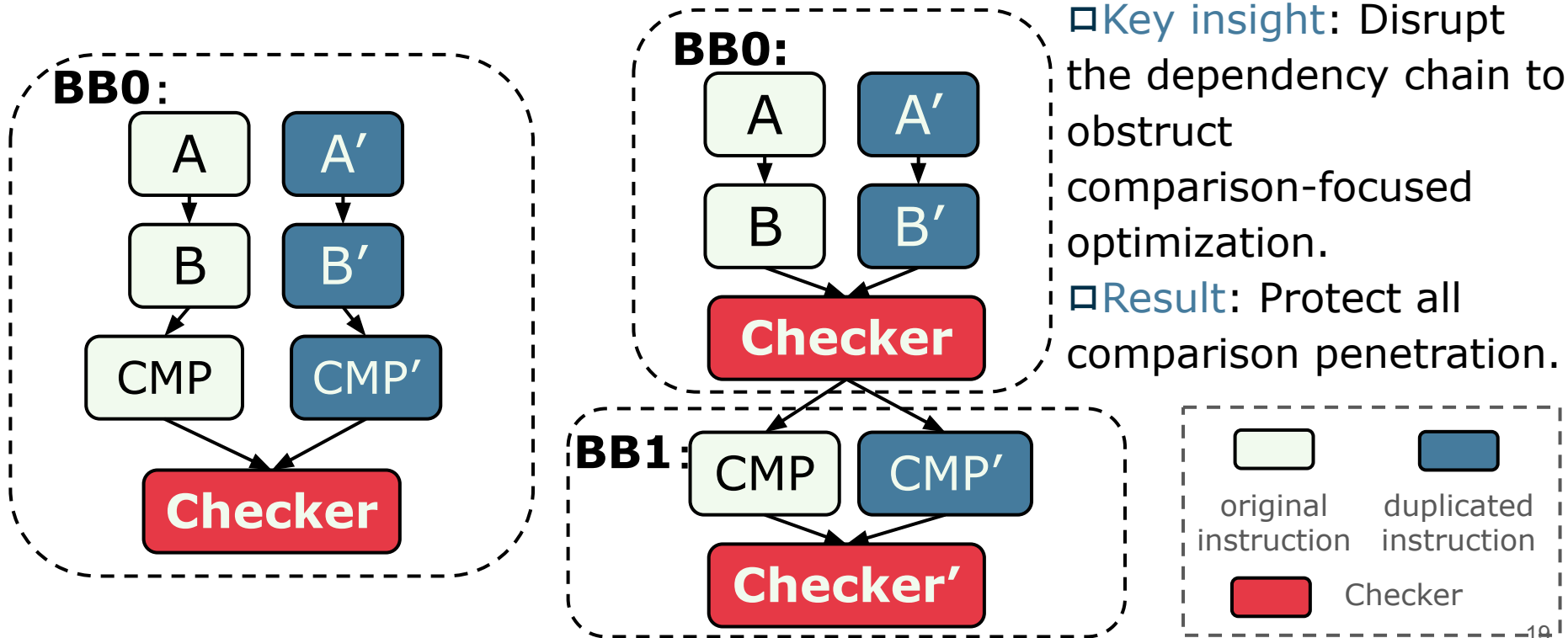
□ **Key insight:** Check branch flag after it jumps.

□ **Result:** Soft errors occur at branch can be detected.



Flowery

□ Anti-Comparison Duplication Optimization



Evaluation

□SDC coverage

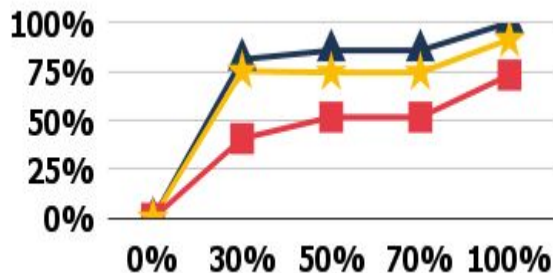
— Optimized assembly level

— Assembly level

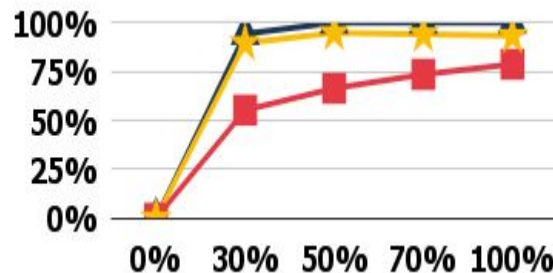
— IR level

X-axis : protection level

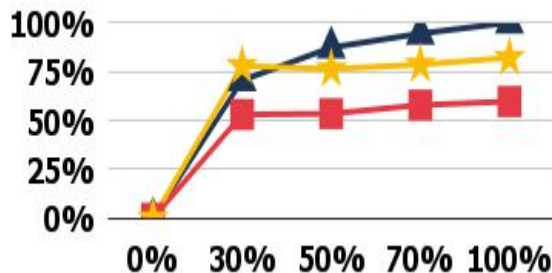
Y-axis : SDC coverage



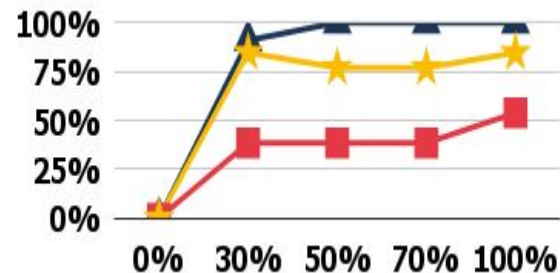
(a) Backprop



(b) Pathfinder



(d) Basicmath



(c) Patricia

EDDI Protection efficiency is significantly enhanced

Conclusion

- We observe **EDDI protection deficiencies** across IR and Assembly levels.
- There are **5 penetration cases** responsible for such deficiencies.
- We propose **Flowery**, which is a set of IR-level modifications.
- Flowery can **mitigate such deficiencies** with **no obvious overhead**.
- Open source: <https://github.com/hyfshishen/SC23-FLOWERY>

IOWA



復旦大學
FUDAN UNIVERSITY

