

Selective Re-execution of Long-retired Misspeculated Instructions Using Forward Slicing



Smruti R. Sarangi, **Wei Liu**, Josep Torrellas, Yuanyuan Zhou

University of Illinois at Urbana-Champaign

<http://iacoma.cs.uiuc.edu>

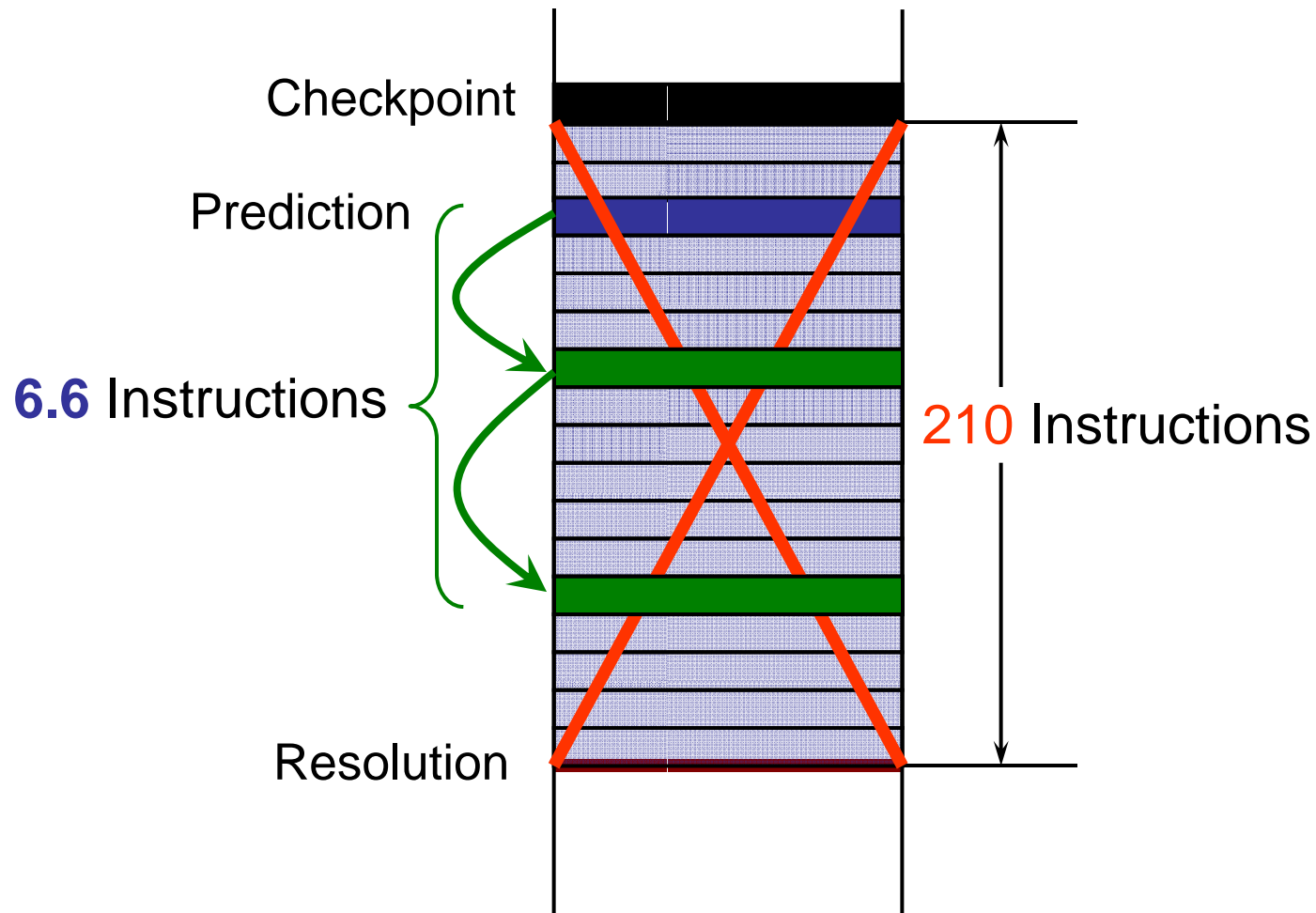
Data Value Speculation

- Predict the value and proceed speculatively
- When the correct value comes in, if missprediction, squash and re-execute

- Initial Proposals
 - L1 data
 - Data Dependences
- Aggressive Novel Proposals
 - Values of L2 misses
 - Thread independence in Thread-Level Speculation
 - Speculative Synchronization



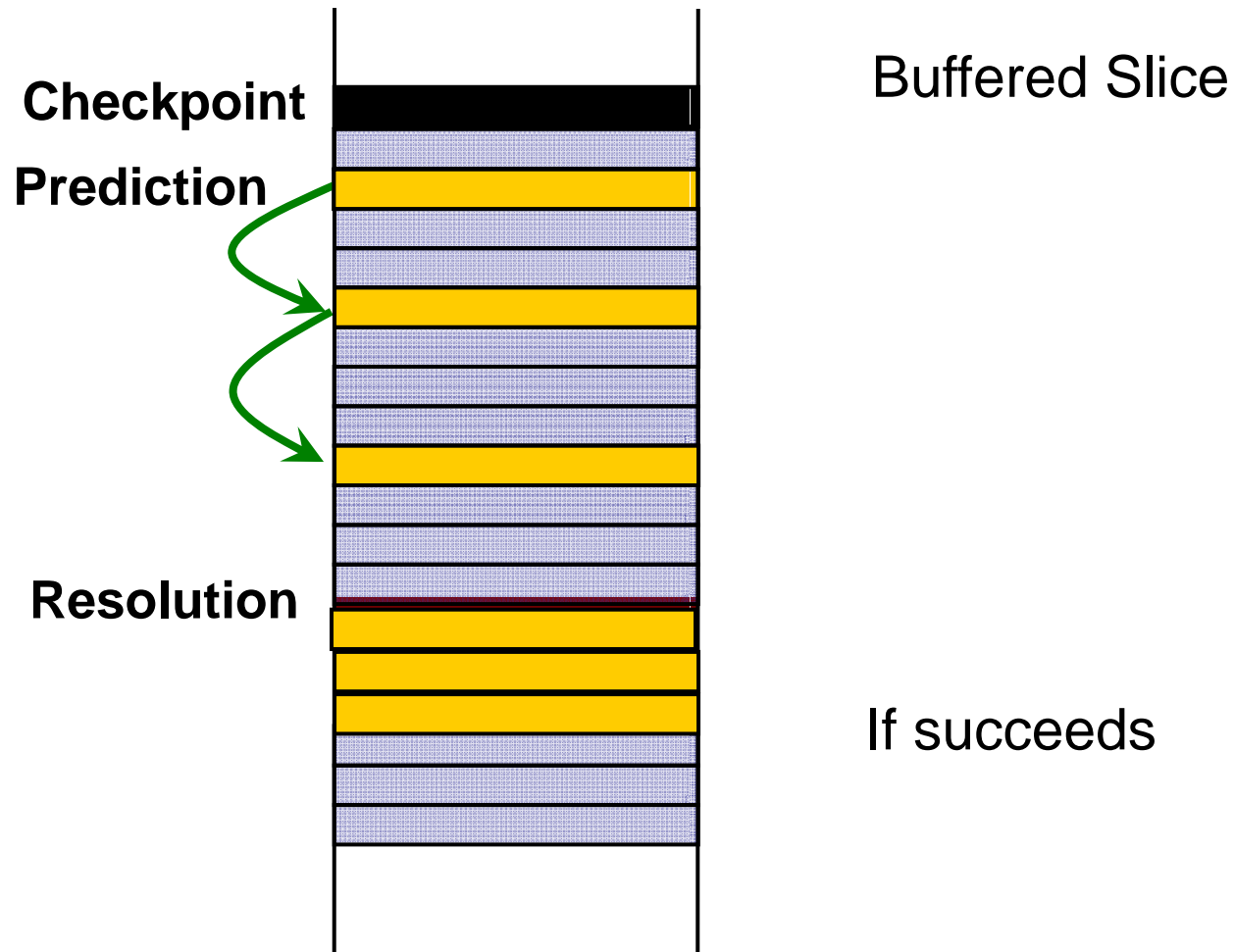
Long-latency Speculation



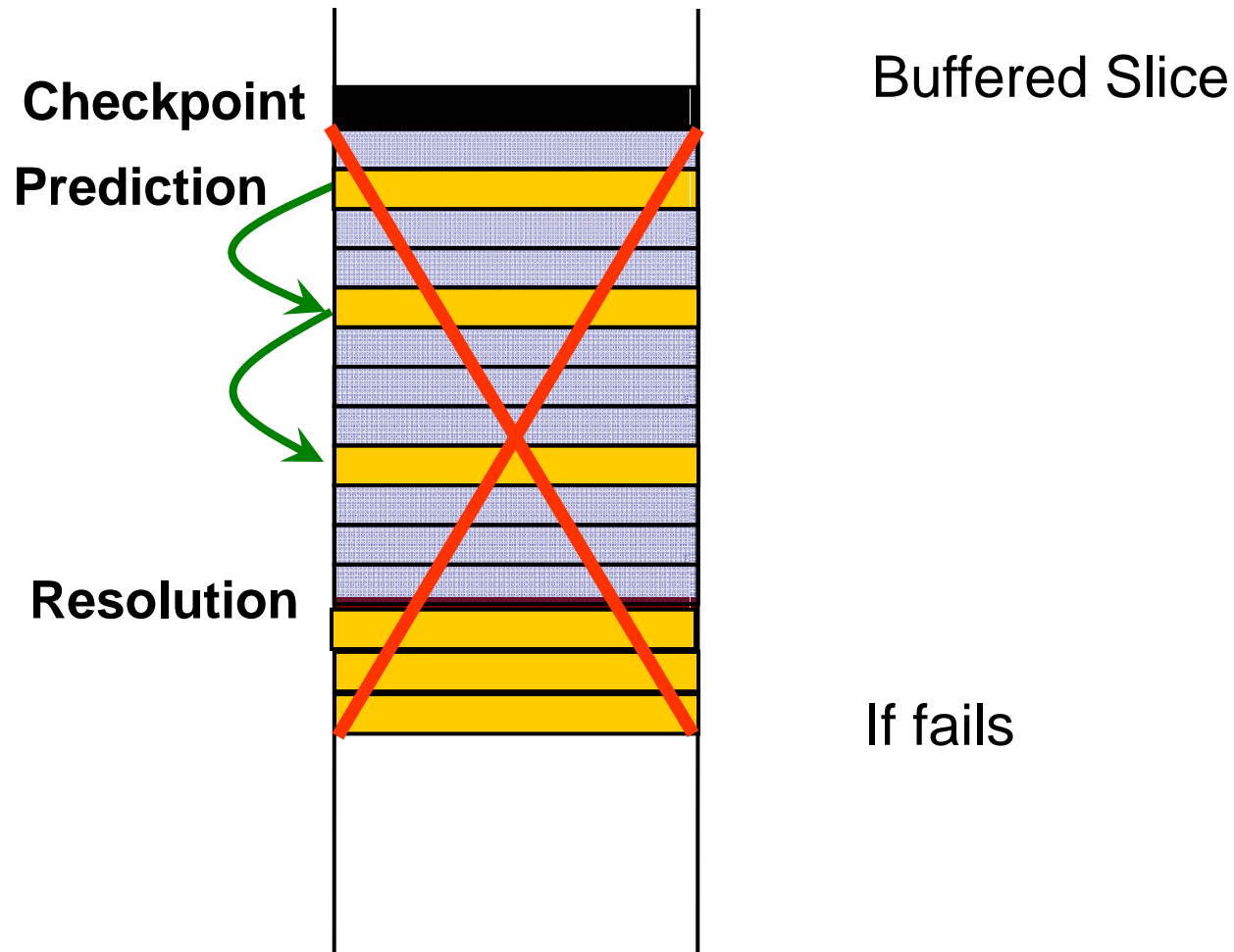
- Misprediction recovery is very wasteful
- Most discarded instructions are still useful



- ReSlice: Architecture to buffer forward slice and re-execute it on misprediction



- ReSlice: Architecture to buffer forward slice and re-execute it on misprediction



- A Sufficient Condition
 - Guarantee to correctly repair the program state

- Application to TLS:
 - Speedup: geometric mean **12%** over TLS
 - ExD² reduction: **20%**



- Motivation and Contributions
- Idea of ReSlice
- Architecture Design
- Experimental Results
- Conclusions



- Initial execution of the task
 - Predict value of “risky” load and continue
 - Buffer in HW the forward slice of the load
- When a misprediction is confirmed
 - Re-execute the slice with the new value
 - If succeed: merge the register and memory state and continue
 - If fail: revert to the conventional recovery: roll back and re-execute



Why is It Challenging?

New values may induce new addresses => Slice changes

“Risky” Load

Initial Execution

...
#1: LD R1 mem[0x0]
...
#2: ST R2 (R1)
...
#3: LD R5 mem[0x20]
...

Buffered Slice

LD R1 < -0x10
ST R2 mem[0x10]

Correct Execution

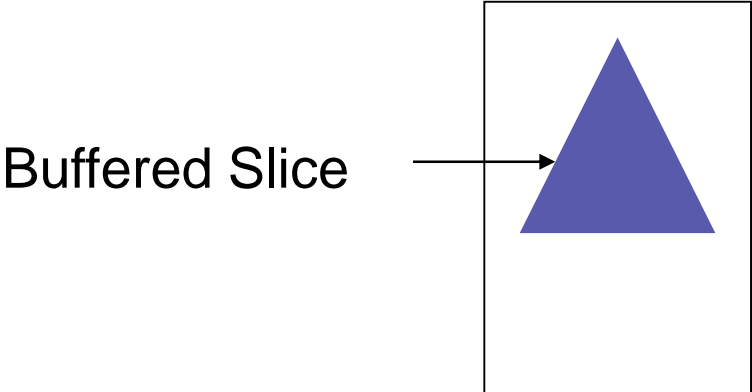
LD R1 < -0x20
ST R2 mem[0x20]
LD R5 mem[0x20]

Problem: Instruction #3 is not buffered!

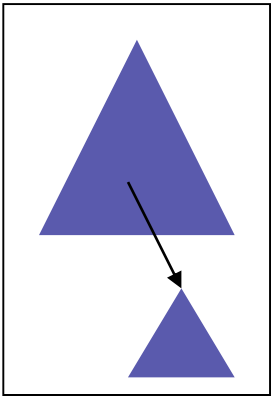


More Challenges

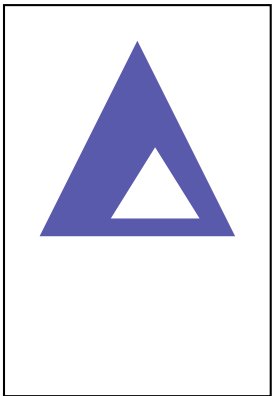
Initial Execution



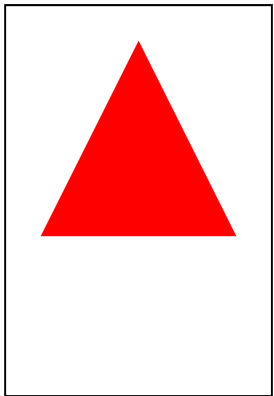
Re-Execution



Extra Insts



Missing Insts

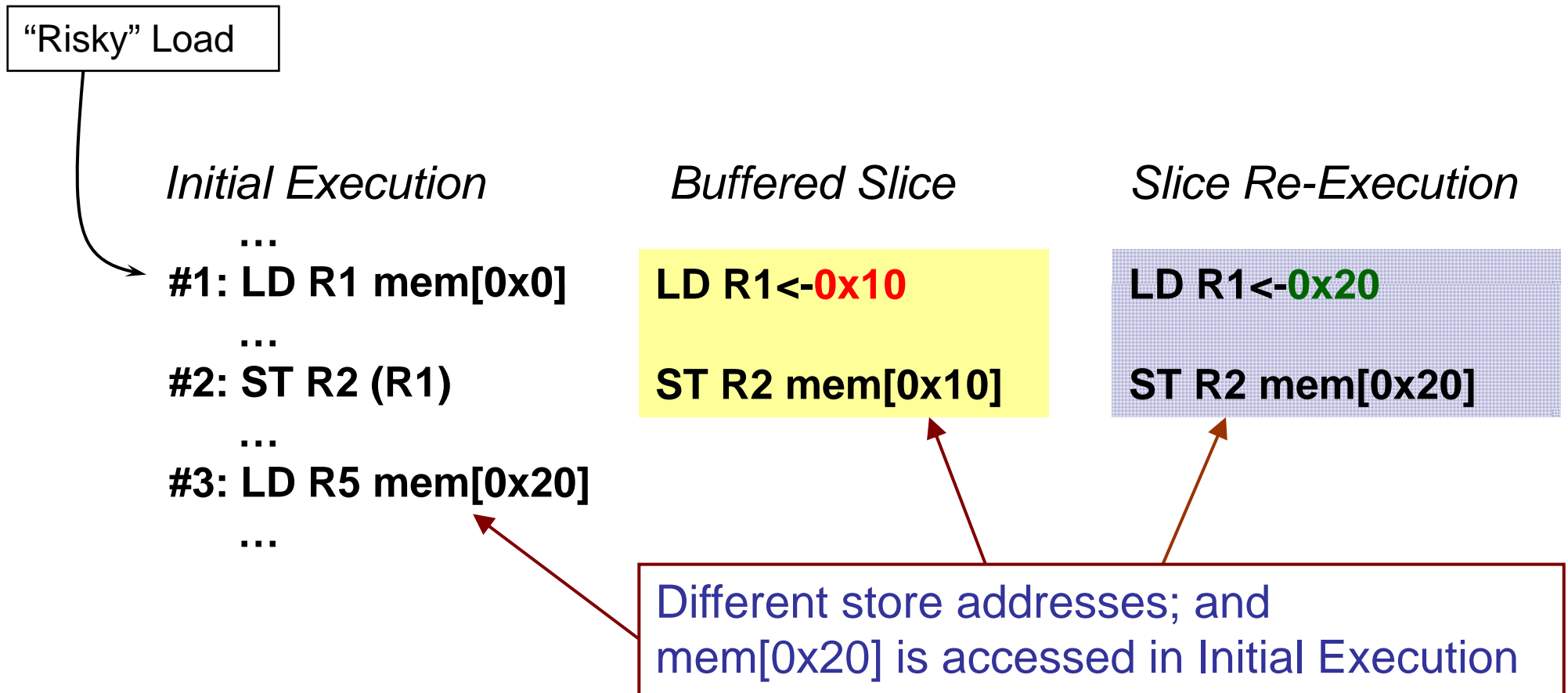


Data Corruption



Solution: The Example Again

Run-time checking during slice re-execution



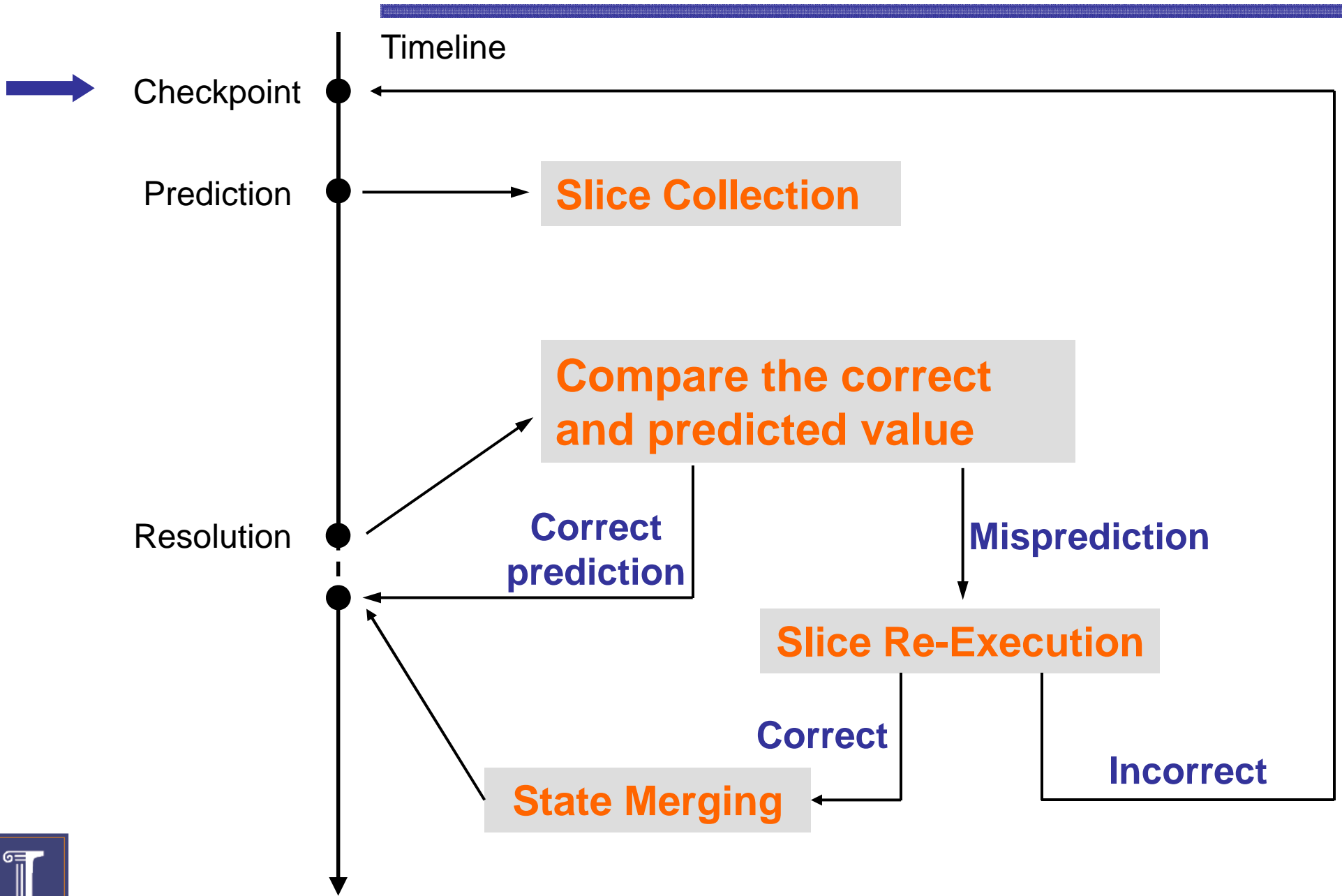
A Sufficient Condition

- Guarantee Correct Slice Re-Execution and Merge
- Easy for HW to check at run-time

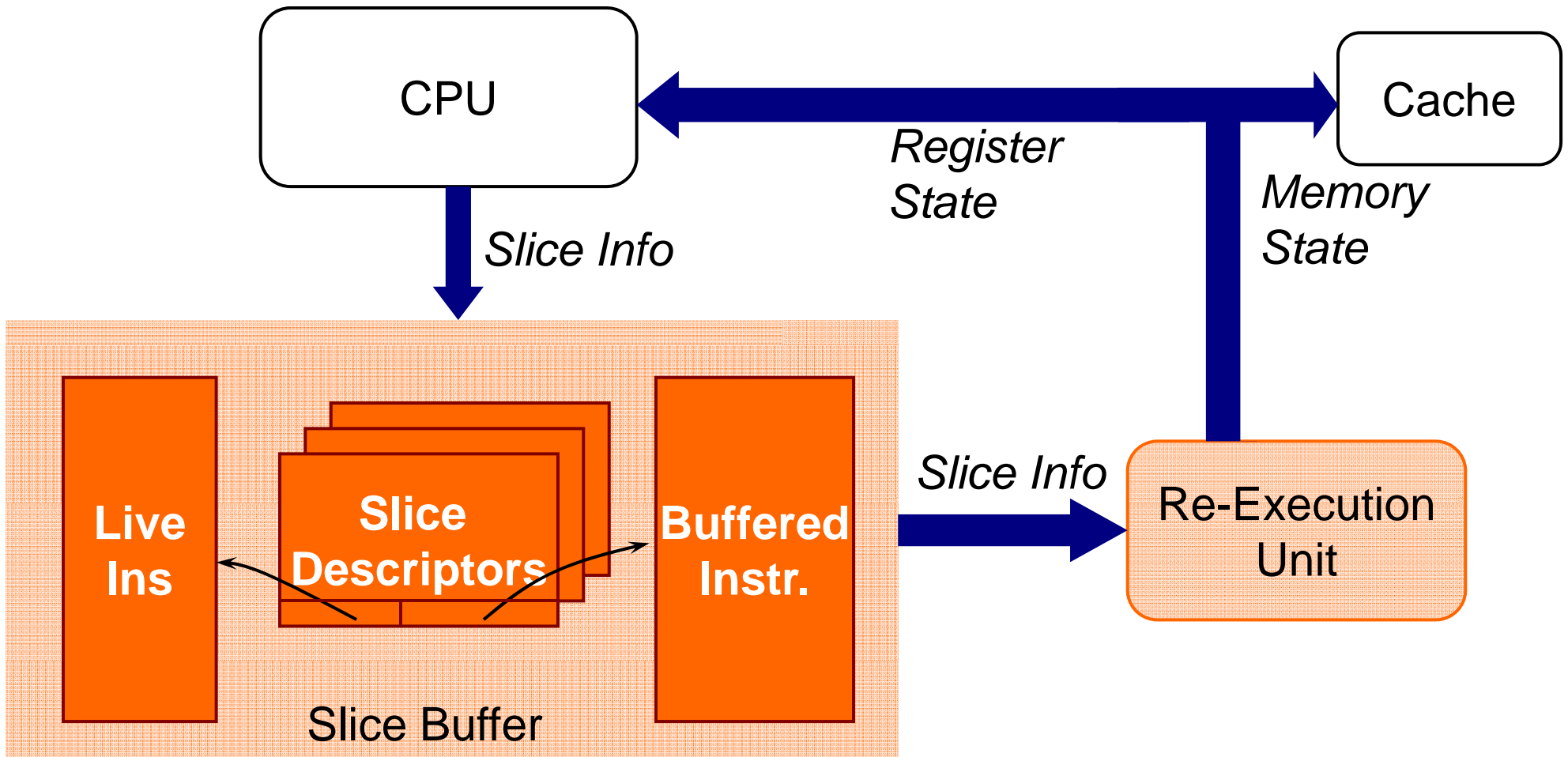
- Details please see our paper



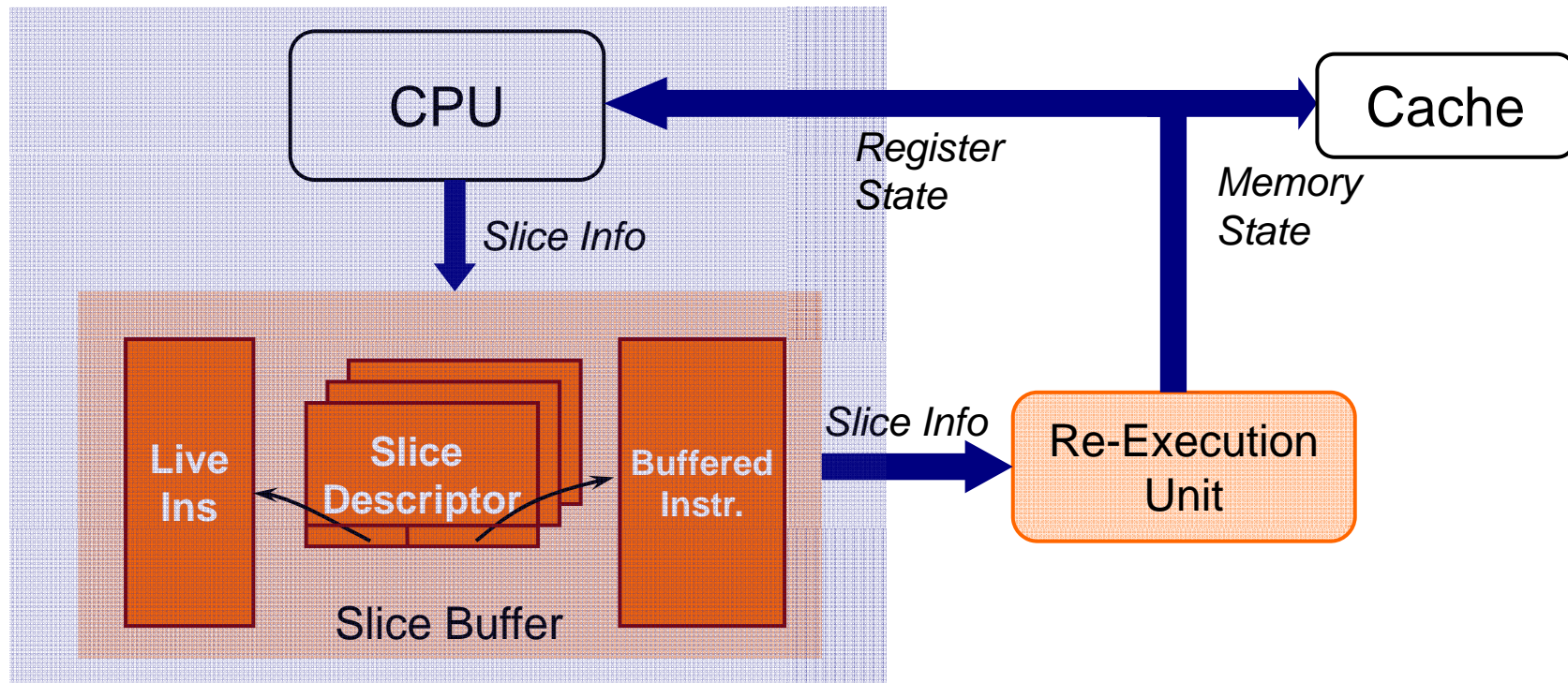
How does ReSlice work?



Architecture Design



Step 1: Slice Collection



- Fill up the Slice Buffer when a prediction is made
 - Track both register and memory dependence
 - Save live-in operands and slice instructions
- Slices are buffered when instructions are retired



An Example

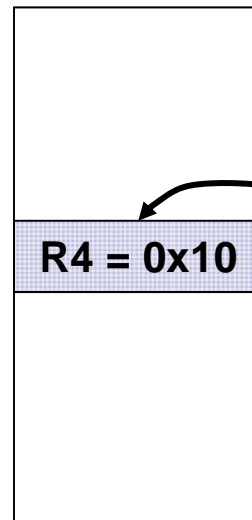
LD R1 mem[0x20]

ADD R3, R1, R2

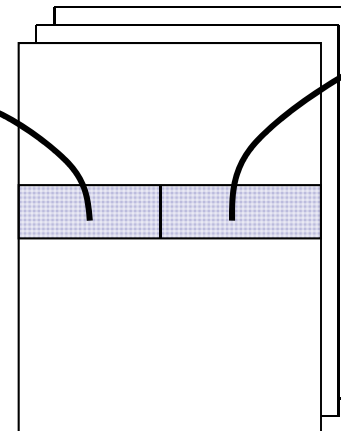
R4 = 0x10

ST R3, mem[R4]

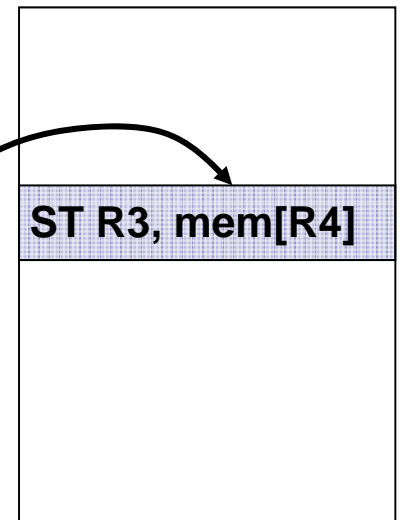
Live Ins



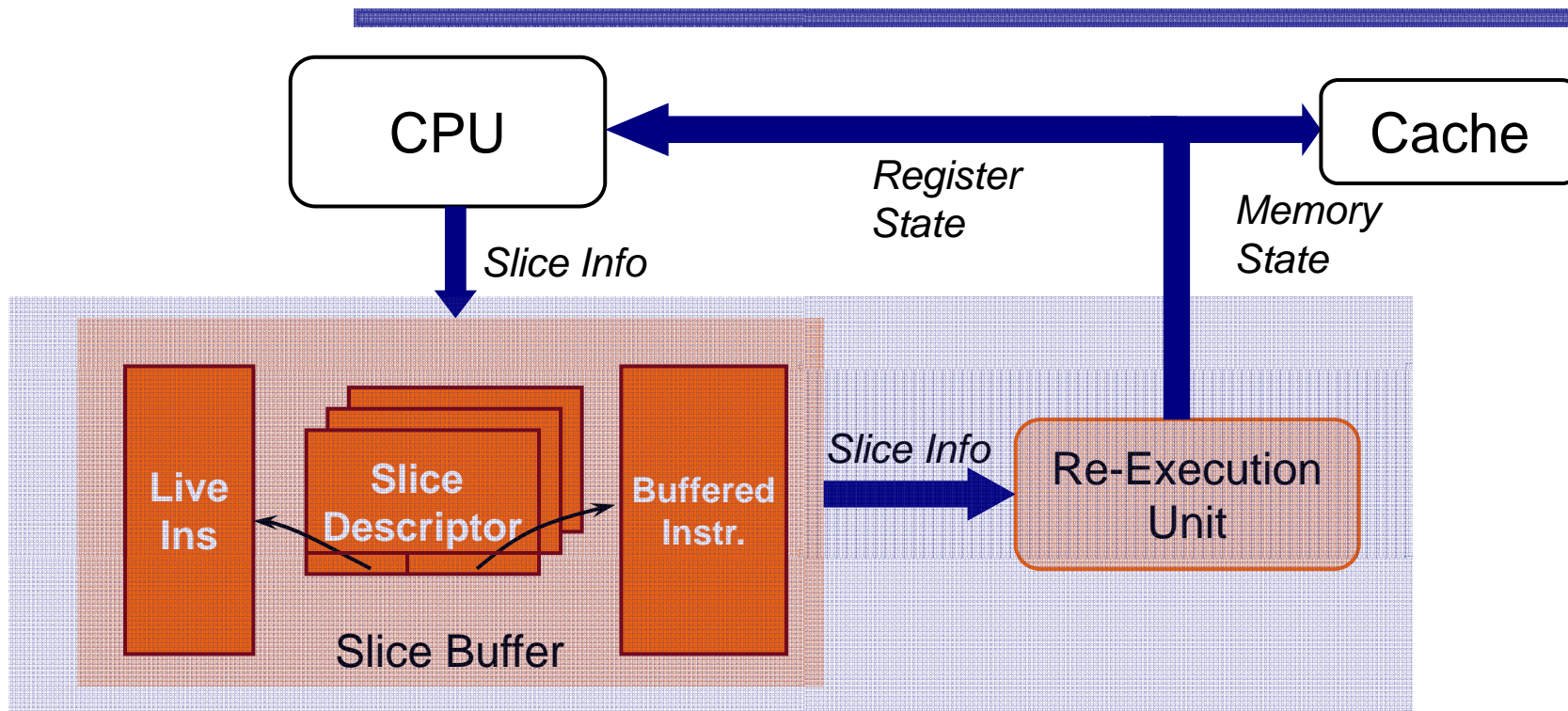
Slice Desc.



Buffered Insts



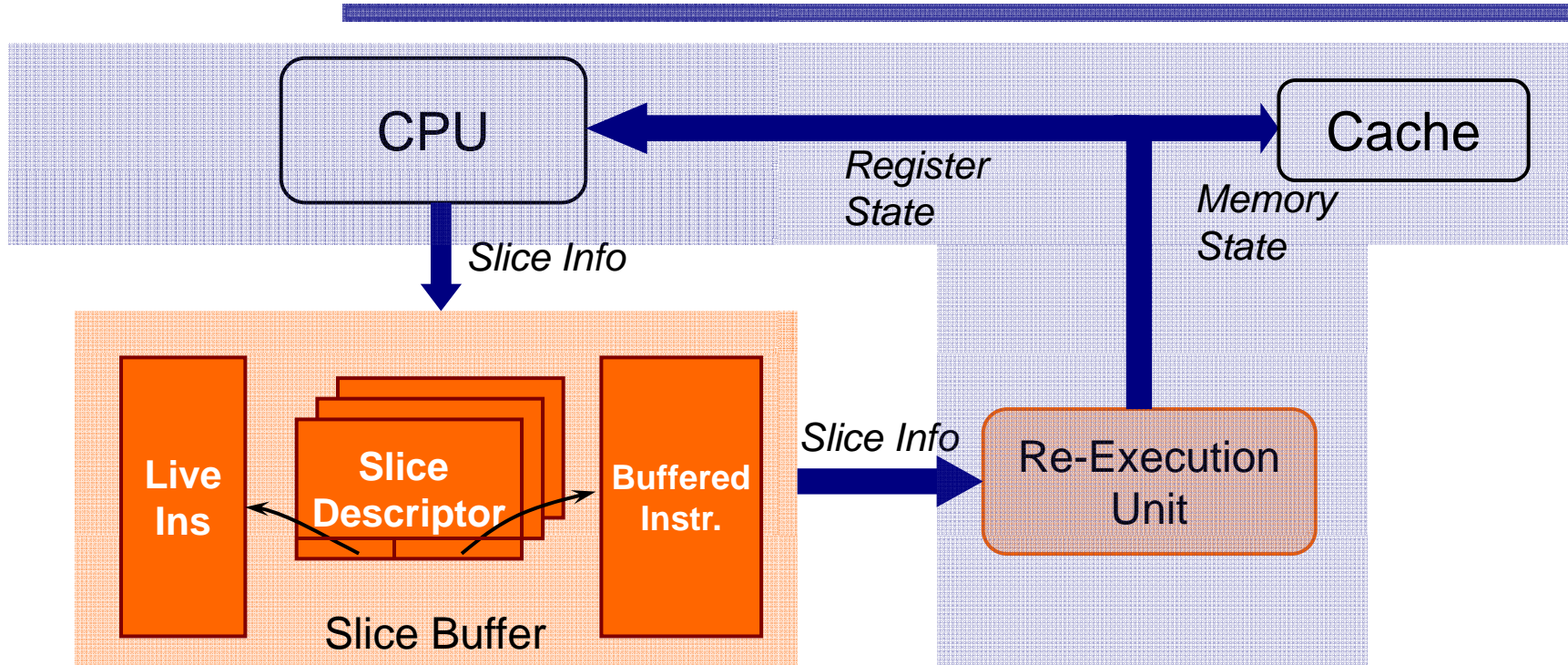
Step 2: Slice Re-Execution



- REU takes over after a violation is verified
 - In-order execution
- Sufficient condition is checked during the re-execution
- If succeeds, merge the register and memory state; otherwise, squash the task and restart



Step 3: State Merging



- Copy **live** registers back to the main process register file
- Merge memory state (details please see the paper)



Multiple Overlapping Slices

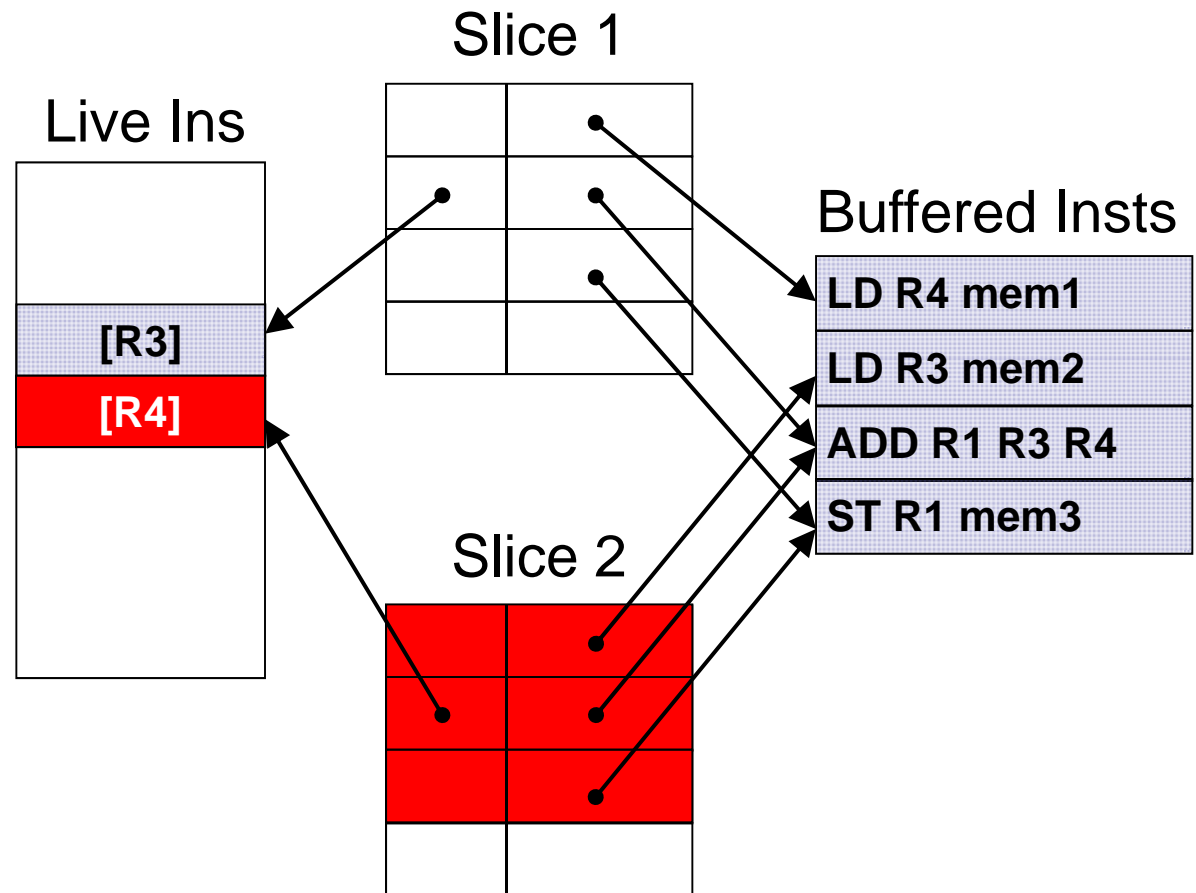
- One slice might corrupt live-ins of the other slice

#1: LD R4 mem1

#2: LD R3 mem2

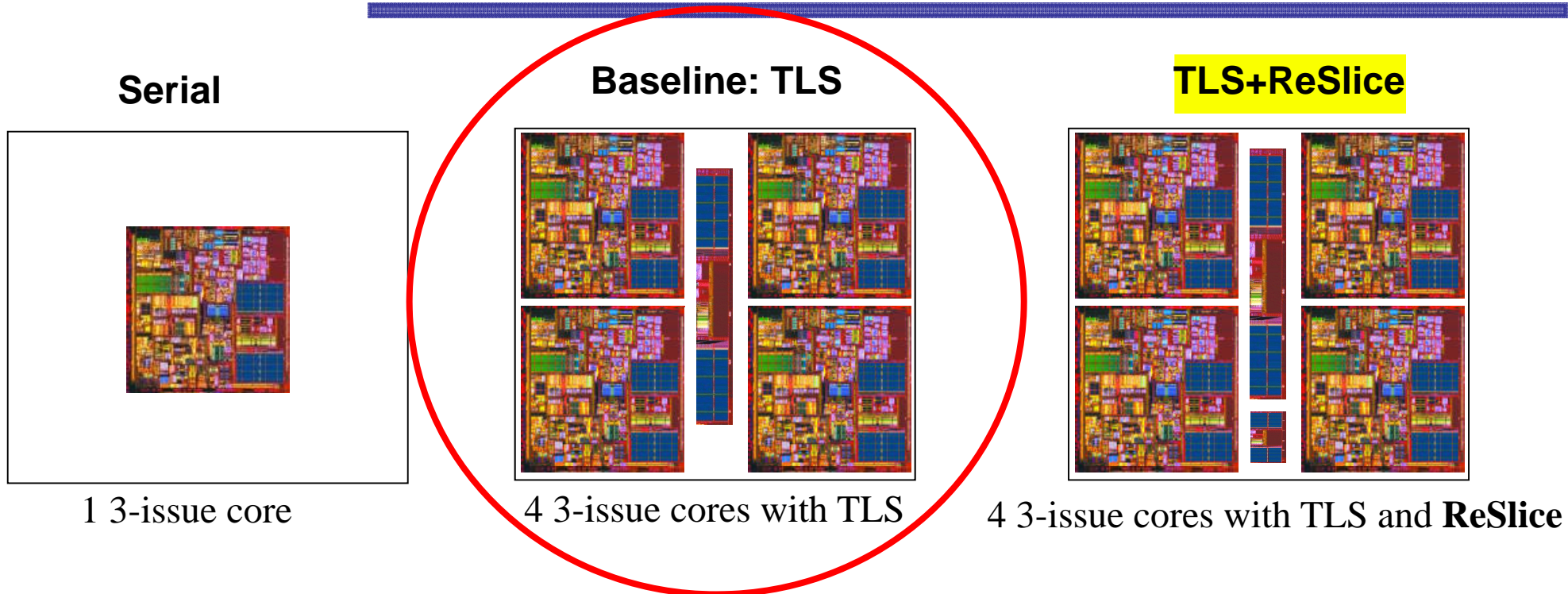
#3: ADD R1 R3 R4

#4: ST R1 mem3



- Motivation and Contributions
- Idea of ReSlice
- Architecture Design
- Experimental Results
- Conclusions

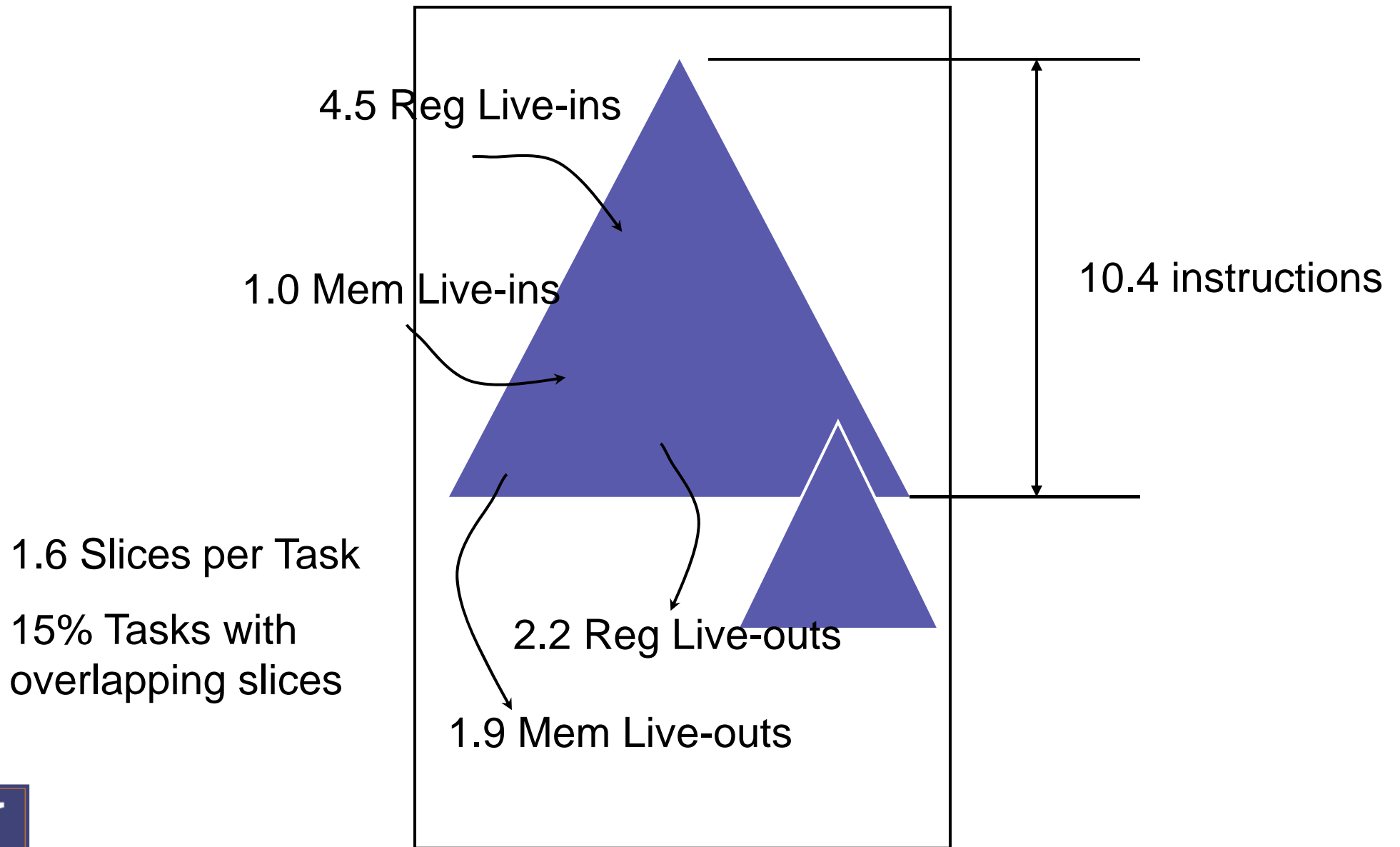




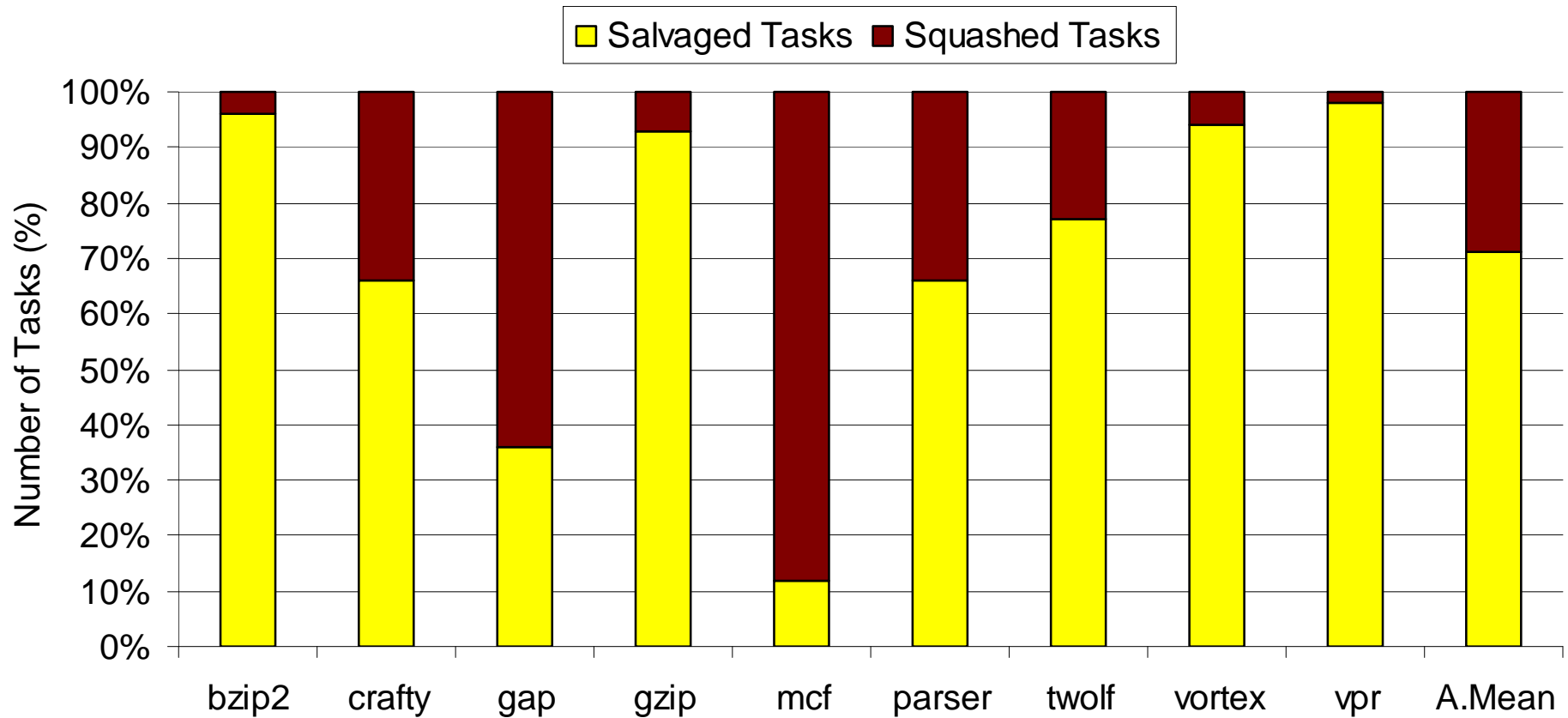
- Simulated Architecture
 - 5 GHz @ 70nm
 - Private 16k L1 per core; MB Shared L2
 - Main memory latency of 500 cycles
- About 0.75-1.50 Billion instructions simulated



Slice Characterization



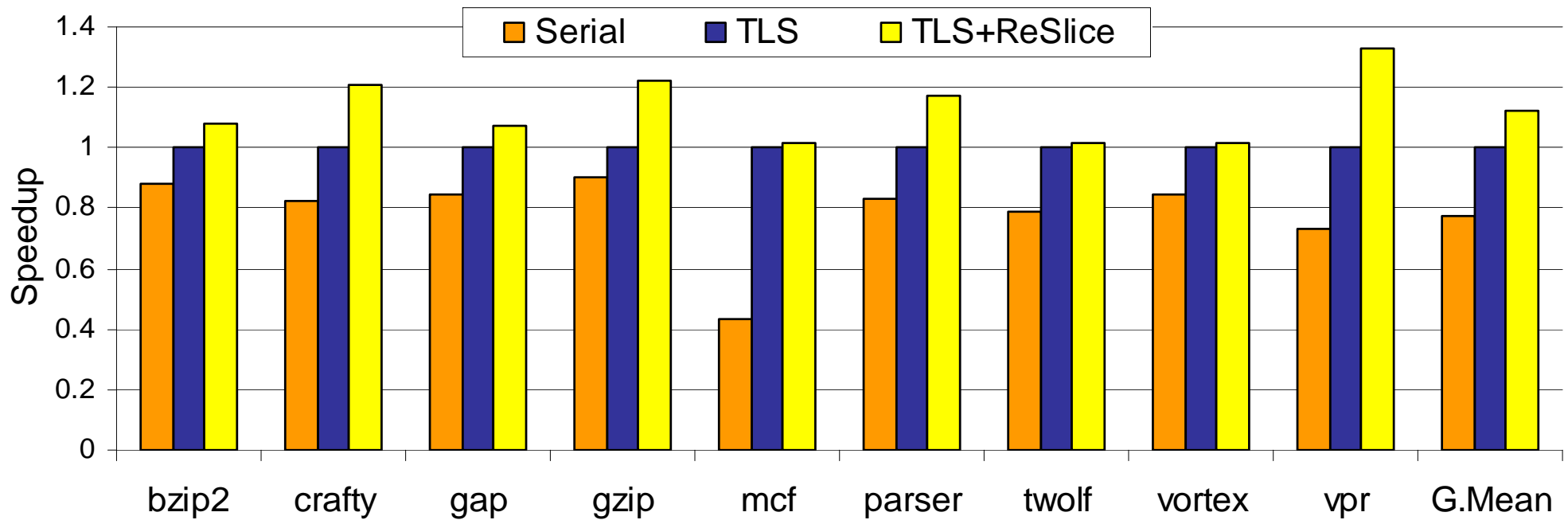
Accuracy of ReSlice



More **70%** tasks are salvaged because of ReSlice



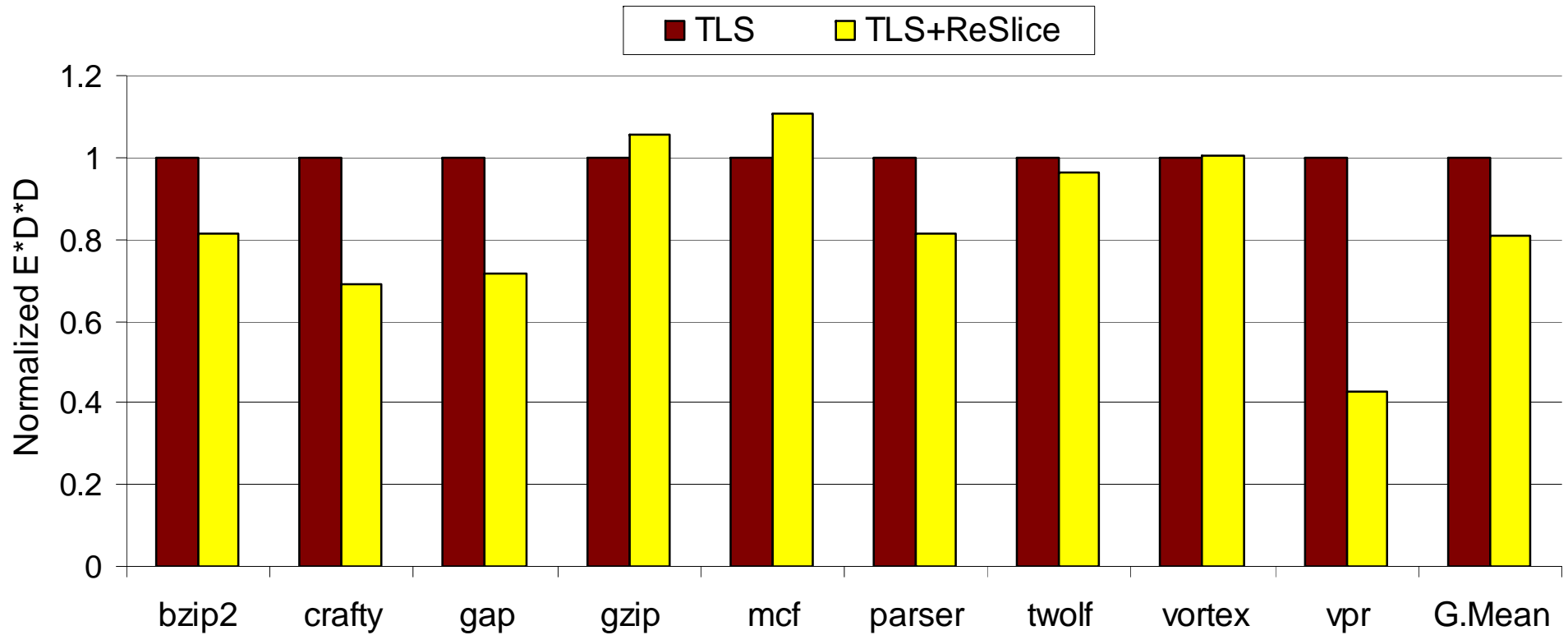
Performance



TLS+ReSlice speeds up **12%** over TLS and **45%** over Serial



Energy \times Delay²



$E \times D^2$ reduction: **20%** over TLS



- Generic Architecture for Forward slice re-execution
- A Sufficient Condition for correct re-execution and merge
- Improve state of the art TLS on SpecInt
 - Speedups: **1.12 over TLS, 1.45 over Serial**
 - ExD² reduction: **20%** over TLS
- **Recovering wasted work is a promising approach**
 - **Boost performance**
 - **Energy efficient**



Selective Re-execution of Long-retired Misspeculated Instructions Using Forward Slicing



Smruti R. Sarangi, **Wei Liu**, Josep Torrellas, Yuanyuan Zhou

University of Illinois at Urbana-Champaign

<http://iacoma.cs.uiuc.edu>

