ATR: Out-of-Order Register Release Exploiting Atomic Regions

Yinyuan Zhao, Surim Oh, Mingsheng Xu, and Heiner Litz

University of California, Santa Cruz

Scaling RF Size is Important

Increase ILP of modern processors

↓

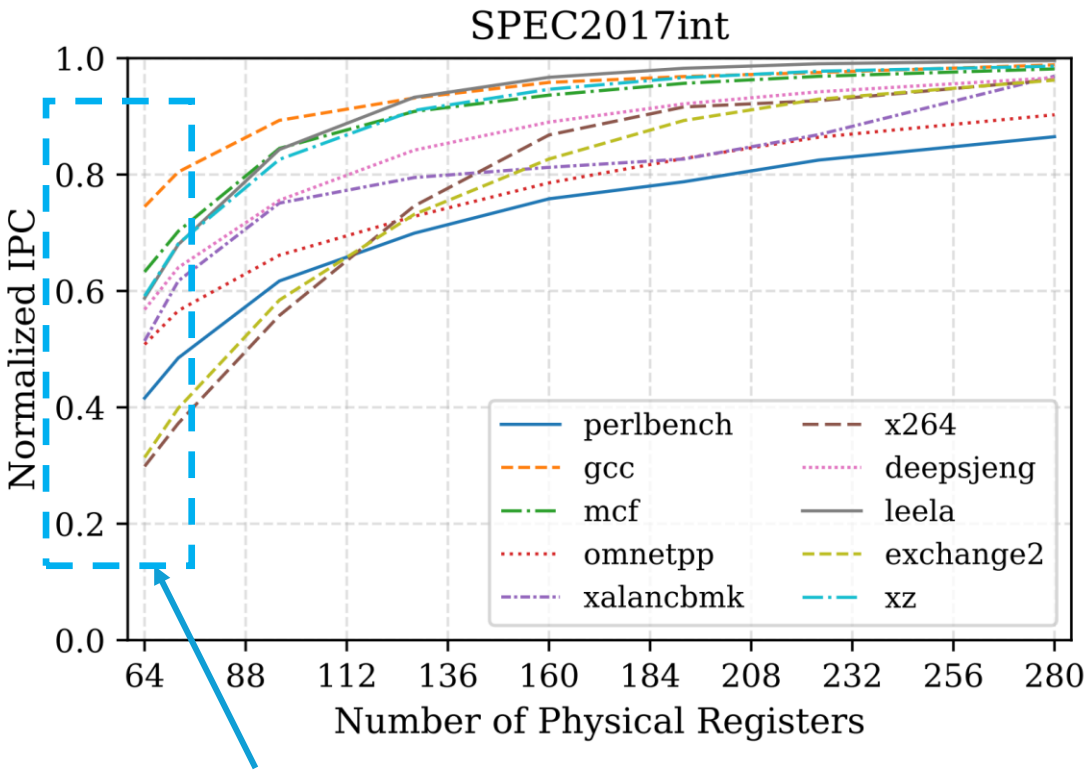
Support large numbers of in-flight ops

↓

Deeper instruction window (ROB)

↓

Larger register file (RF)



With 64 physical registers, the average IPC reaches only 37.7% of the ideal case

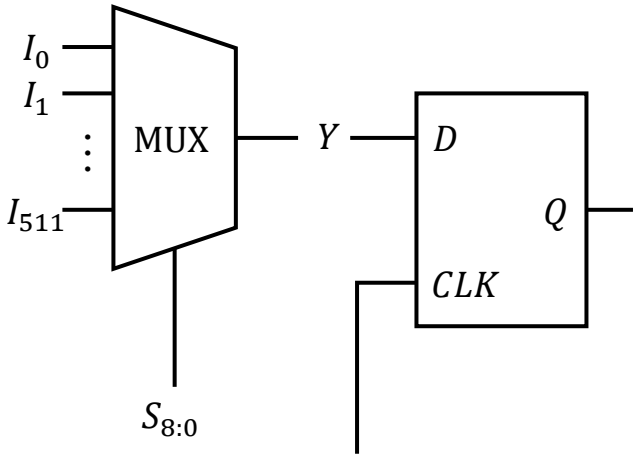
A larger RF is crucial for sustaining high IPC

Scaling RF Size is Challenging

10-wide u-Arch
- 20 read RF ports
- 10 write RF ports

+

512-entry RF
- 512:1 multiplexers
- 64-bit flip-flops



2M transistors

$$1024 \times 64 \times 20$$
$$+\quad 512 \times 64 \times 20$$
$$1.3M + 650K$$

Introduce challenges in area, power, and clock frequency

Conventional Renaming

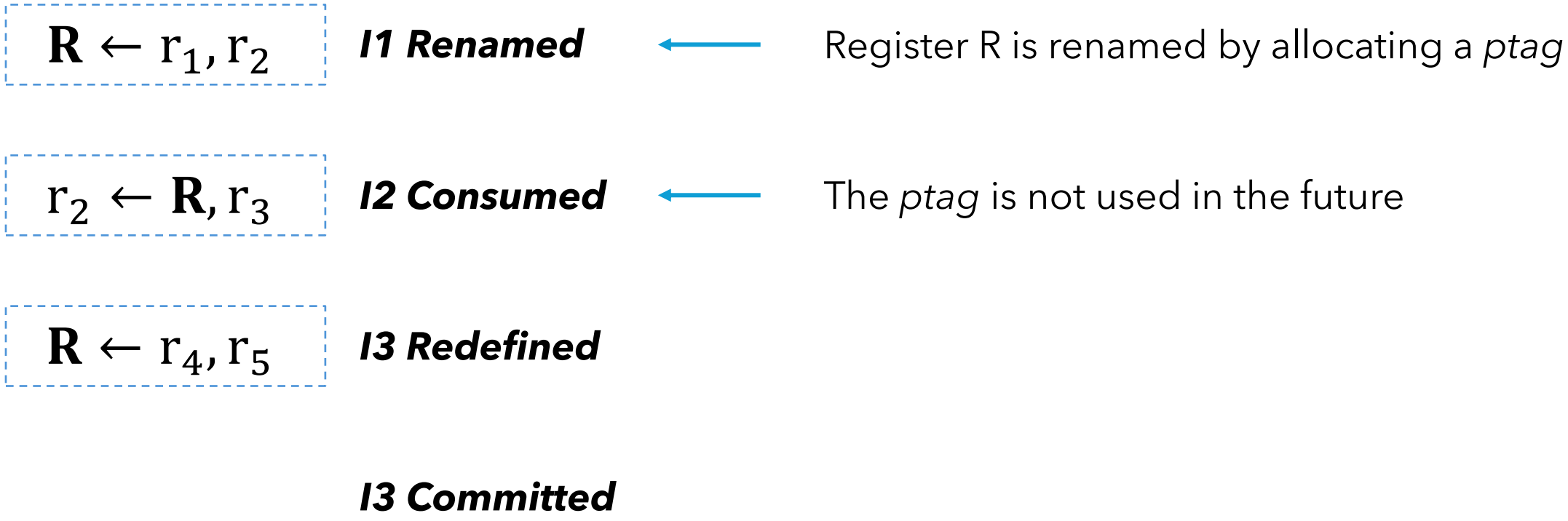
$\mathbf{R} \leftarrow r_1, r_2$ ***I1 Renamed*** ← Register R is renamed by allocating a *ptag*

$r_2 \leftarrow \mathbf{R}, r_3$ ***I2 Consumed***

$\mathbf{R} \leftarrow r_4, r_5$ ***I3 Redefined***

I3 Committed

Conventional Renaming



Conventional Renaming

$\mathbf{R} \leftarrow r_1, r_2$ ***I1 Renamed*** ← Register R is renamed by allocating a *ptag*

$r_2 \leftarrow \mathbf{R}, r_3$ ***I2 Consumed*** ← The *ptag* is not used in the future

$\mathbf{R} \leftarrow r_4, r_5$ ***I3 Redefined***

I3 Committed ← The *ptag* of register R is released

Conventional Renaming holds physical registers much longer than needed!

Speculative Early Release

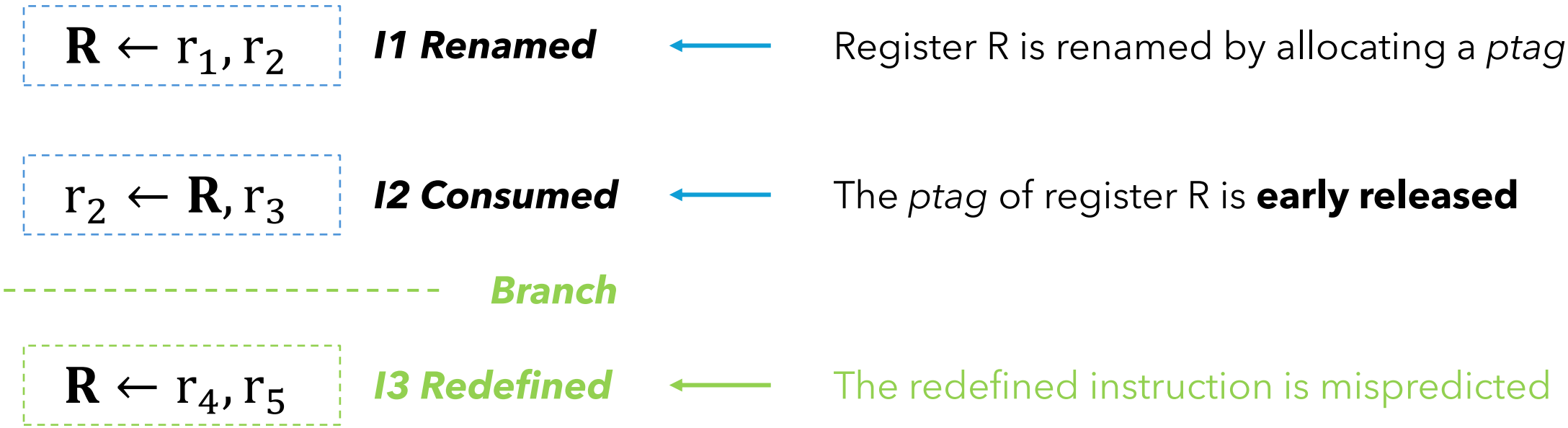
$\mathbf{R} \leftarrow r_1, r_2$ ***I1 Renamed*** ← Register R is renamed by allocating a *ptag*

$r_2 \leftarrow \mathbf{R}, r_3$ ***I2 Consumed*** ← The *ptag* of register R is **early released**

Branch

$\mathbf{R} \leftarrow r_4, r_5$ ***I3 Redefined***

Speculative Early Release



Speculative Early Release

$\mathbf{R} \leftarrow r_1, r_2$ ***I1 Renamed*** ← Register R is renamed by allocating a *ptag*

$r_2 \leftarrow \mathbf{R}, r_3$ ***I2 Consumed*** ← The *ptag* of register R is **early released**

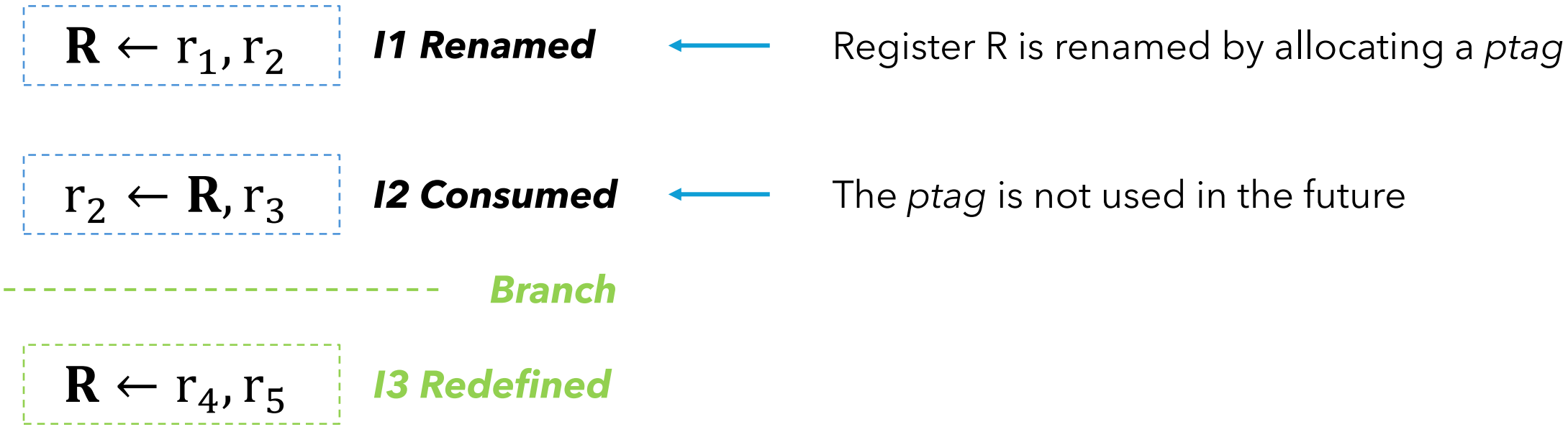
Branch

$\mathbf{R} \leftarrow r_4, r_5$ ***I3 Redefined*** ← The redefined instruction is mispredicted

$r_2 \leftarrow \mathbf{R}, r_3$ ***I4 Consumed*** ← The released *ptag* is consumed!!!

Speculative early release is unsafe or not feasible

Non-Speculative Early Release



Non-Speculative Early Release

$\mathbf{R} \leftarrow r_1, r_2$ ***I1 Renamed*** ← Register R is renamed by allocating a *ptag*

$r_2 \leftarrow \mathbf{R}, r_3$ ***I2 Consumed*** ← The *ptag* is not used in the future

Branch

All the older branches or exception-causing instructions are executed

$\mathbf{R} \leftarrow r_4, r_5$ ***I3 Redefined***

I3 Precommitted

Non-Speculative Early Release

$\mathbf{R} \leftarrow r_1, r_2$ ***I1 Renamed*** ← Register R is renamed by allocating a *ptag*

$r_2 \leftarrow \mathbf{R}, r_3$ ***I2 Consumed*** ← The *ptag* is not used in the future

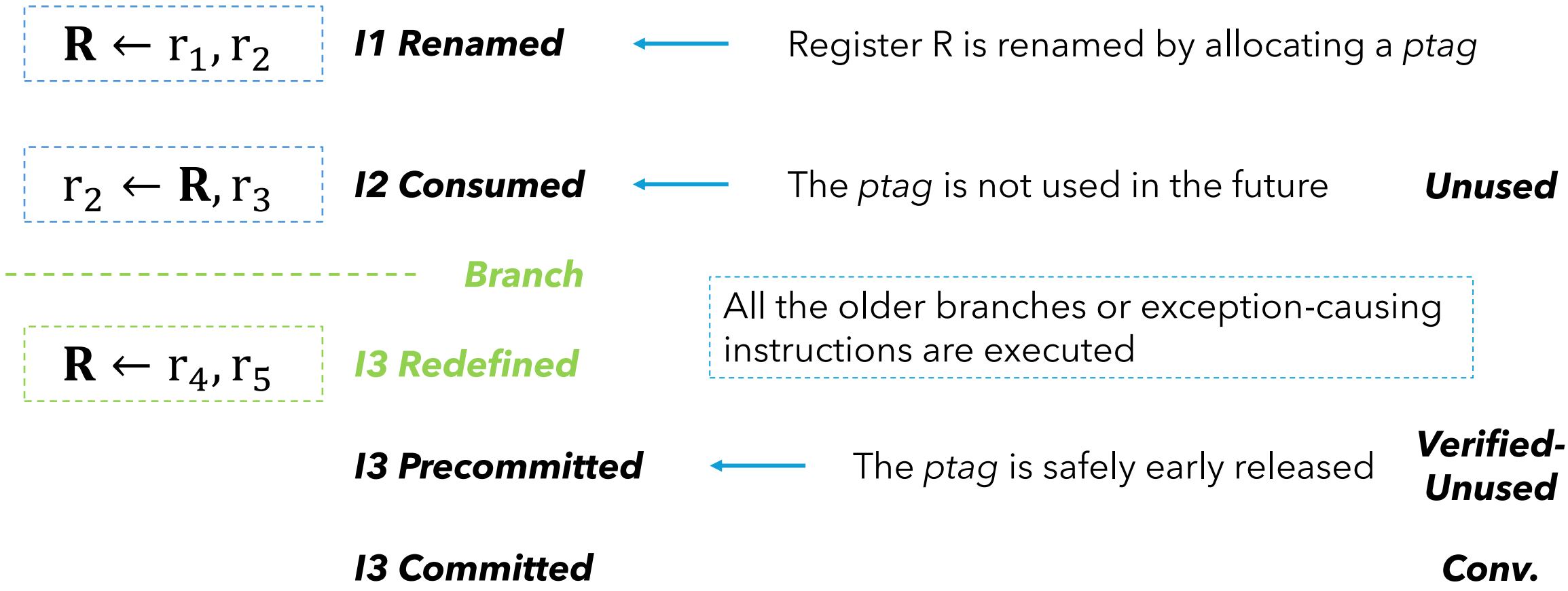
Branch

All the older branches or exception-causing instructions are executed

$\mathbf{R} \leftarrow r_4, r_5$ ***I3 Redefined***

I3 Precommitted ← The *ptag* is safely early released

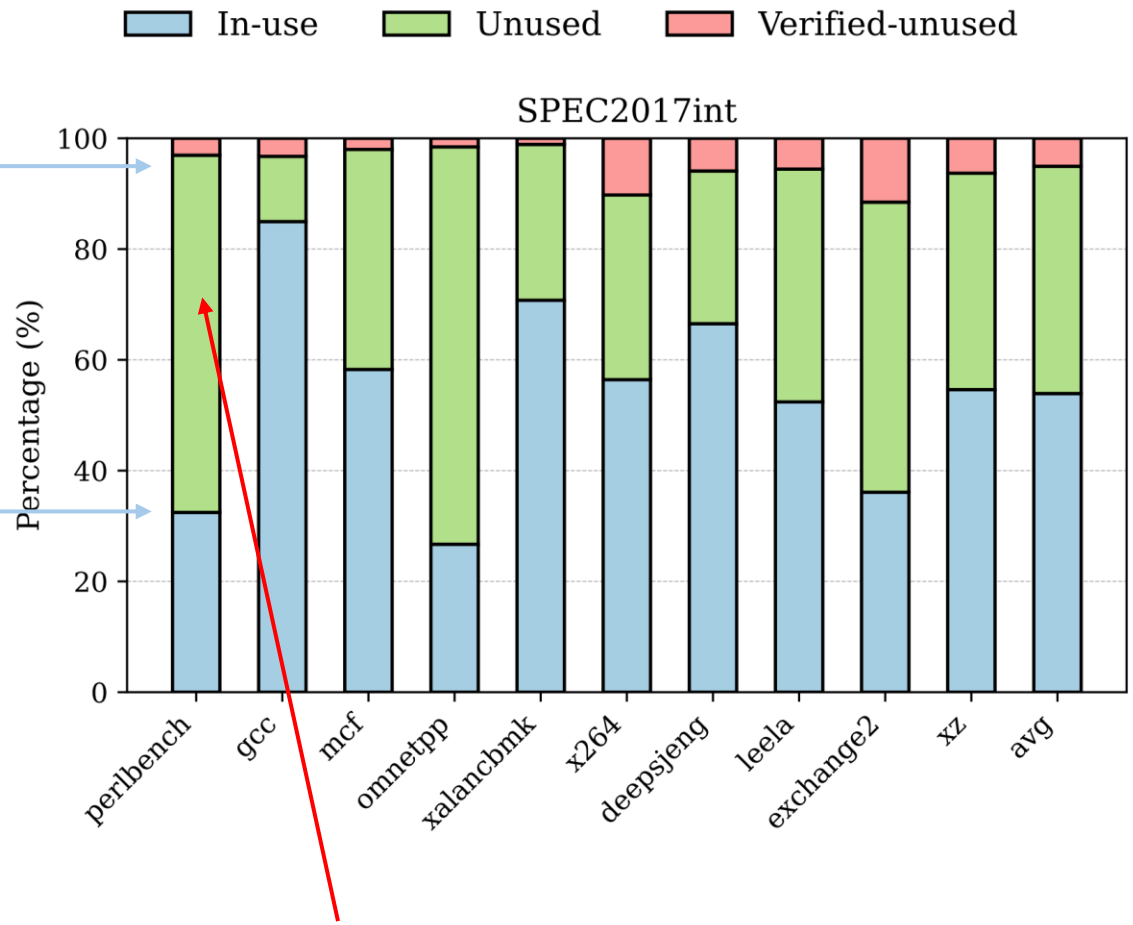
Non-Speculative Early Release



Gap between Non-Spec and Spec ER

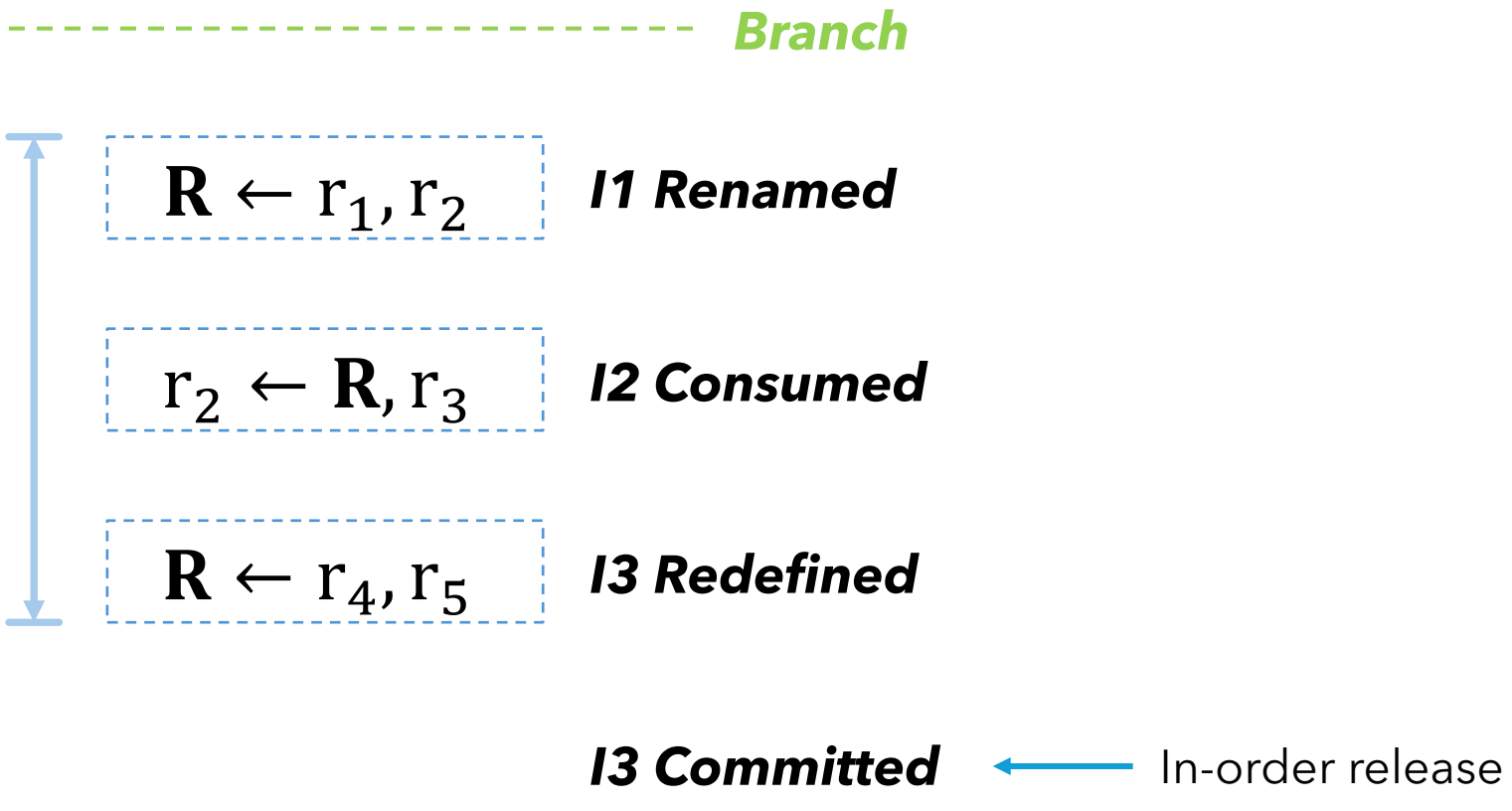
Non-Speculative ER

Speculative ER



Opportunity of early releasing unused register

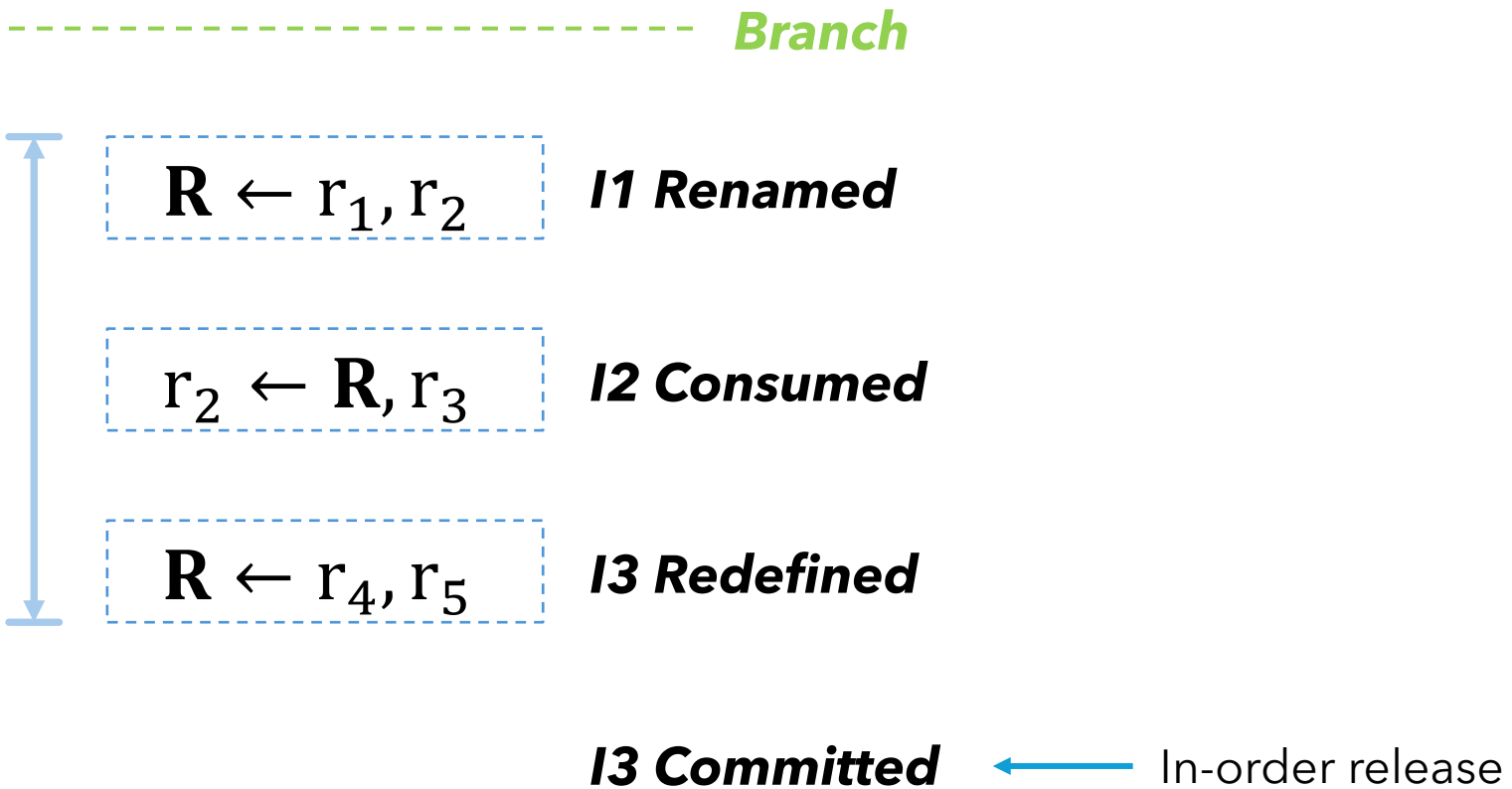
Our Proposal



Our Proposal

Atomic Commit Region

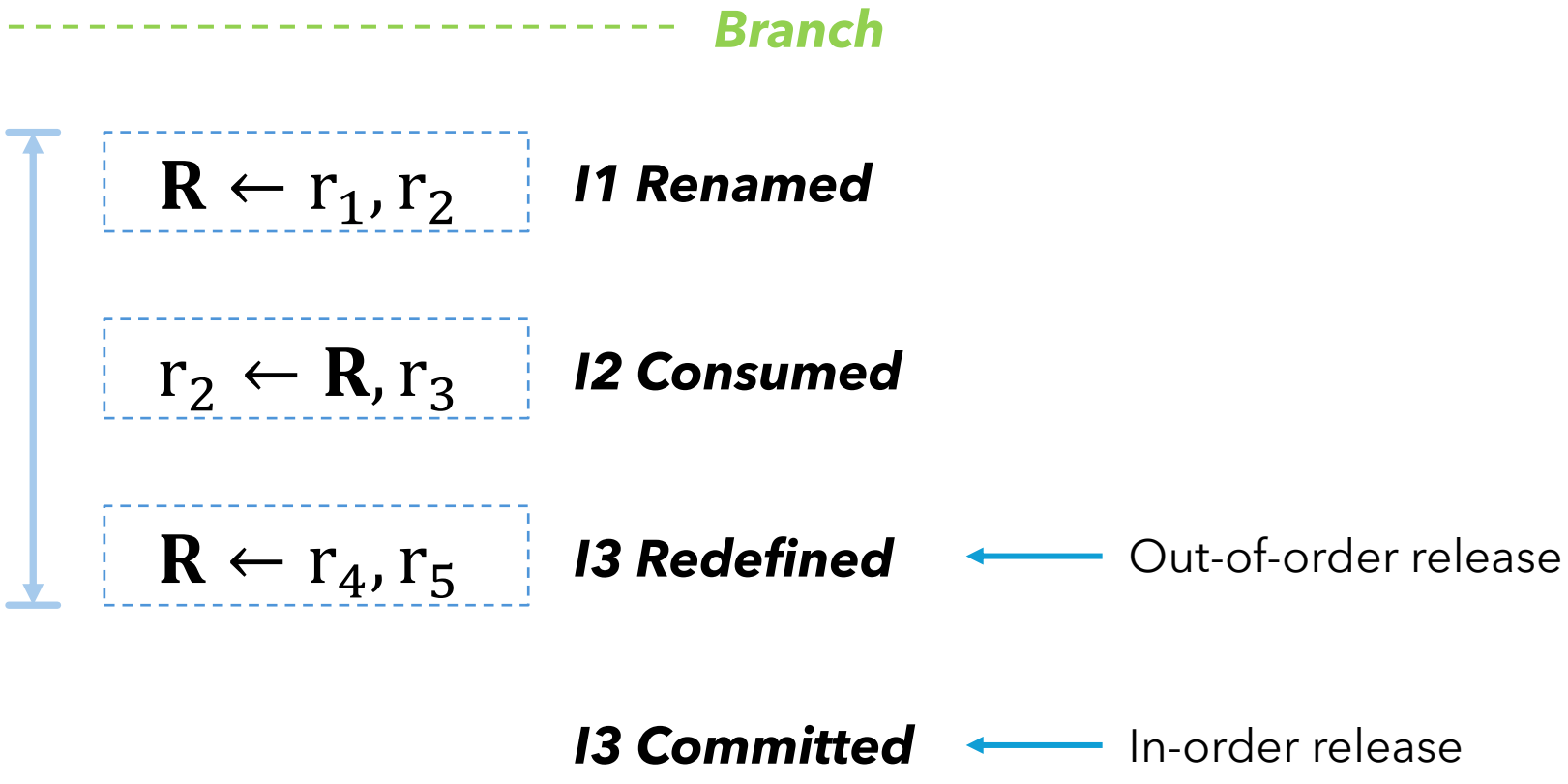
Branches/Exceptions-free
- *Either entirely off-path or on-path*



Our Proposal

Atomic Commit Region

Branches/Exceptions-free
- *Either entirely off-path or on-path*



Does not violate precise exceptions nor suffers from mispredictions

Analysis

			Re	Ex	Cm	Pr
I1	MOVE	RAX ← RAX	510	675	839	675
I2	TEST + JNZ	ZPS ← RAX	510	841	841	841
	…	…				
I3	LEA	RAX ← RDI	709	716	716	841
	…	…				
I4	LEA	RBX ← RAX	729	737	737	842
I5	SHR	RBX ← RBX, ZPS	729	738	738	842

I1 is a high-latency load

An example segment of instructions from SPEC2017int omnetpp

Analysis

			Re	Ex	Cm	Pr
I1	MOVE	RAX ← RAX	510	675	839	675
I2	TEST + JNZ	ZPS ← RAX	510	841	841	841
	...	...				
I3	LEA	RAX ← RDI	709	716	716	841
	...	...				
I4	LEA	RBX ← RAX	729	737	737	842
I5	SHR	RBX ← RBX, ZPS	729	738	738	842

An example segment of instructions from SPEC2017int omnetpp

I1 is a high-latency load

I2 needs to wait until wake-up

Analysis

			Re	Ex	Cm	Pr
I1	MOVE	RAX ← RAX	510	675	839	675
I2	TEST + JNZ	ZPS ← RAX	510	841	841	841
	...	...				
I3	LEA	RAX ← RDI	709	716	716	841
	...	...				
I4	LEA	RBX ← RAX	729	737	737	842
I5	SHR	RBX ← RBX, ZPS	729	738	738	842

I1 is a high-latency load

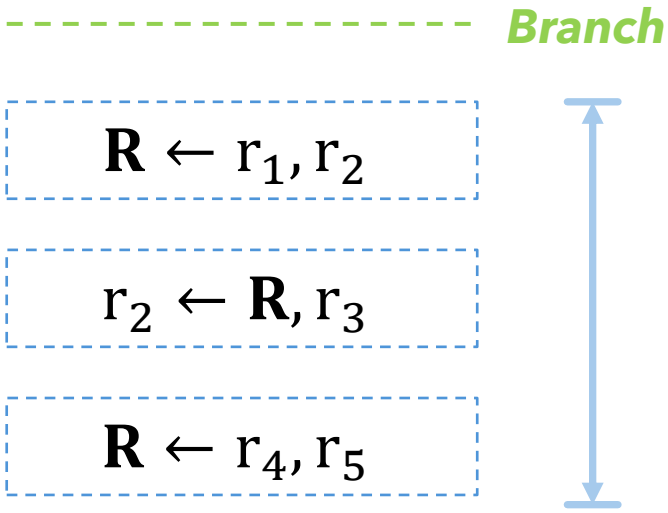
I2 needs to wait until wake-up

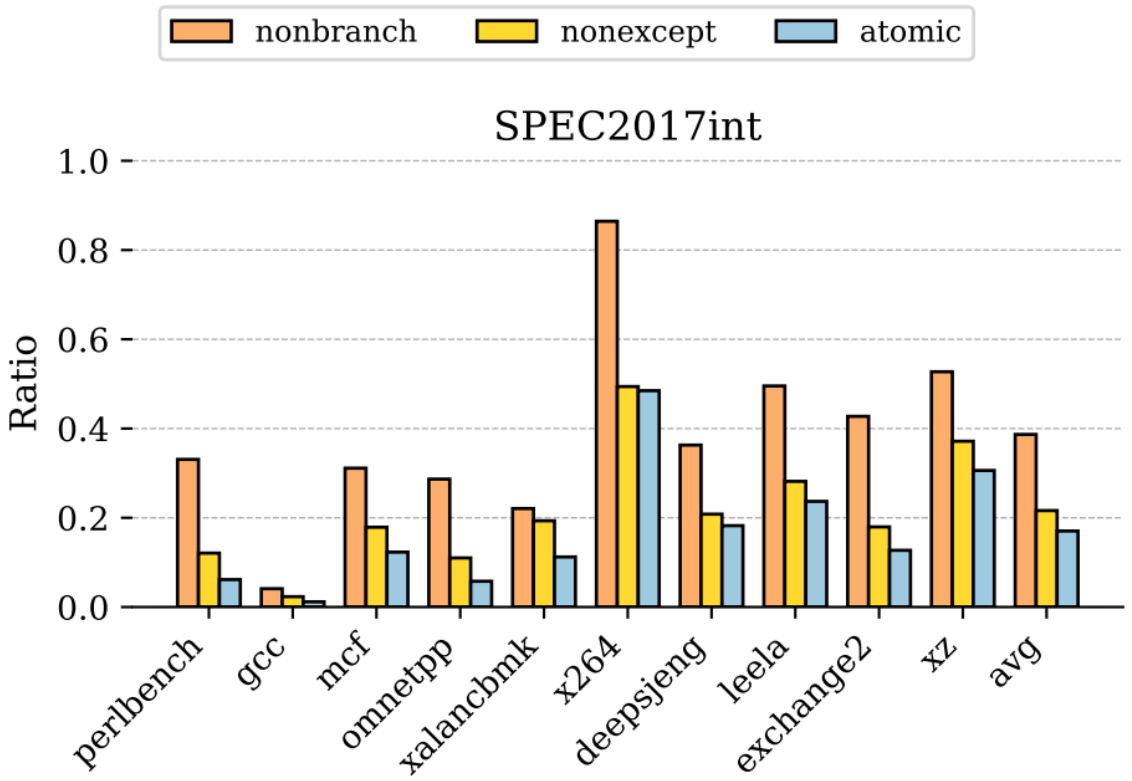
I4 precommit is delayed

An example segment of instructions from SPEC2017int omnetpp

Lots of cycles are wasted in waiting for precommitting

Atomic Ratio





From the atomic register to the redefined one:

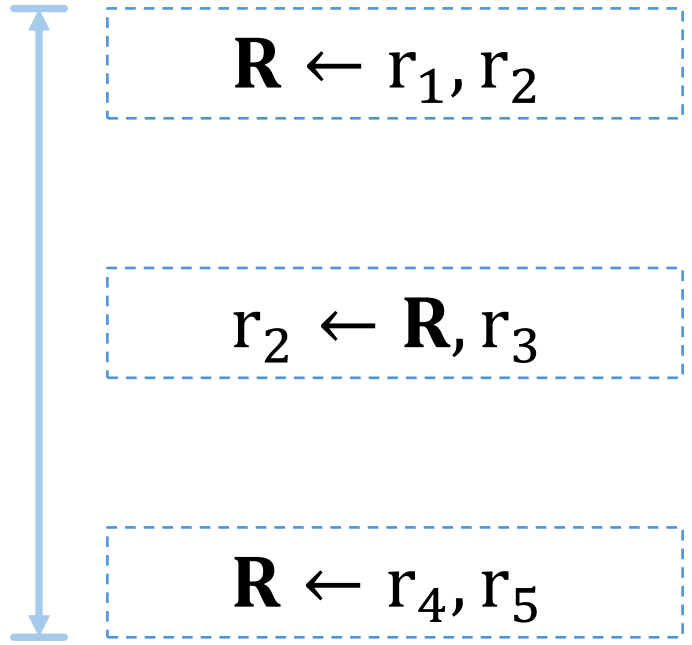
- no branch/jump
- no exception-causing instructions (ld/st/div)

Over 17% of all allocated registers are located within atomic commit regions

ATR Algorithm

1. Atomic

- No **br/jmp/ld/st/div** between the renaming and redefining instruction



2. Consumed

- All consumers have consumed R

3. Redefined

- R has been redefined by a later instruction

No need to wait for commit or pre-commit of any instruction!

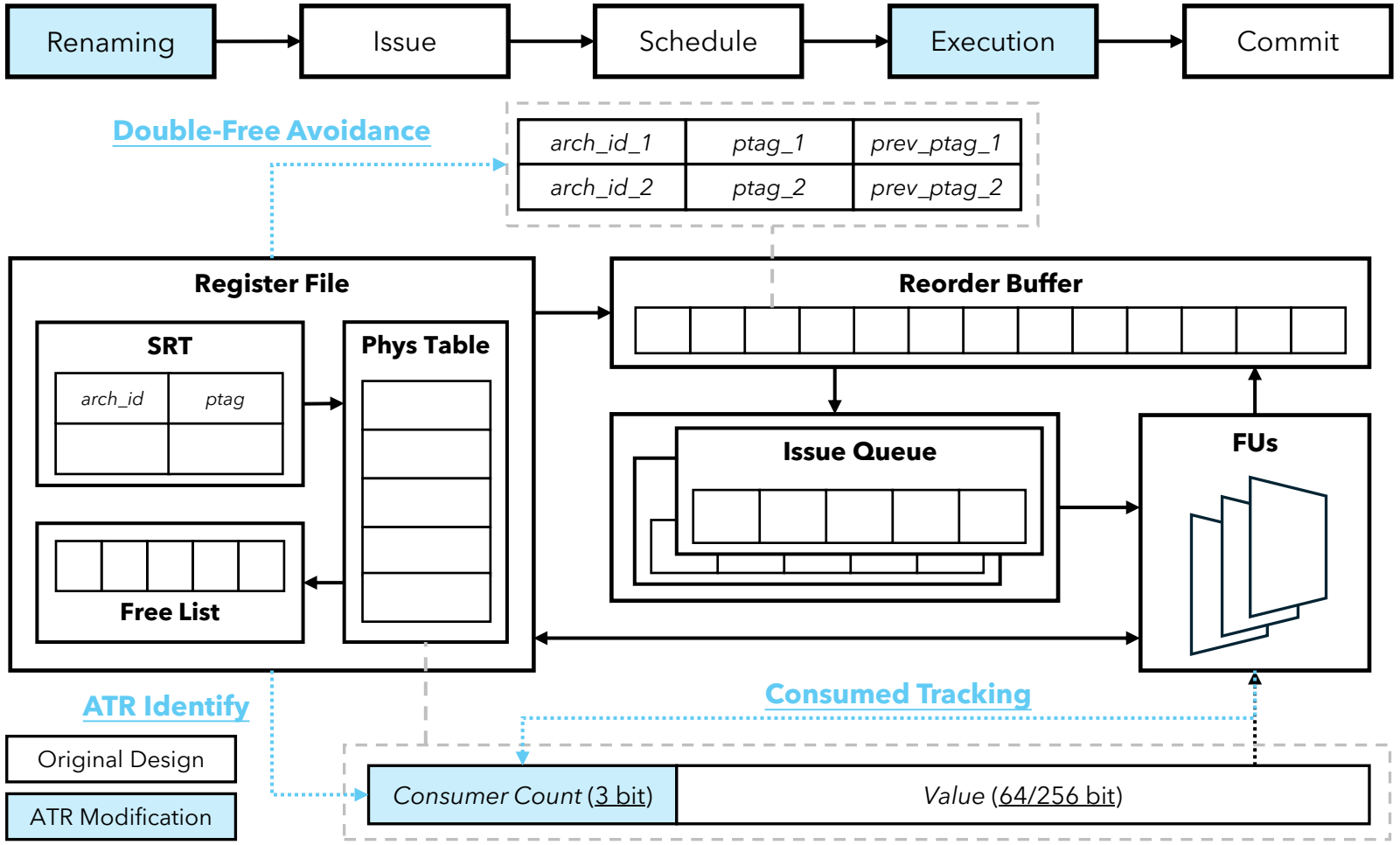
Design

Consumer Renaming
- Consumer Count + +

Consumer Execution
- Consumer Count - -

3-bit consumer count for each ptag

BR/JMP/LD/ST/DIV Renaming
- Consumer Count = *MAX*



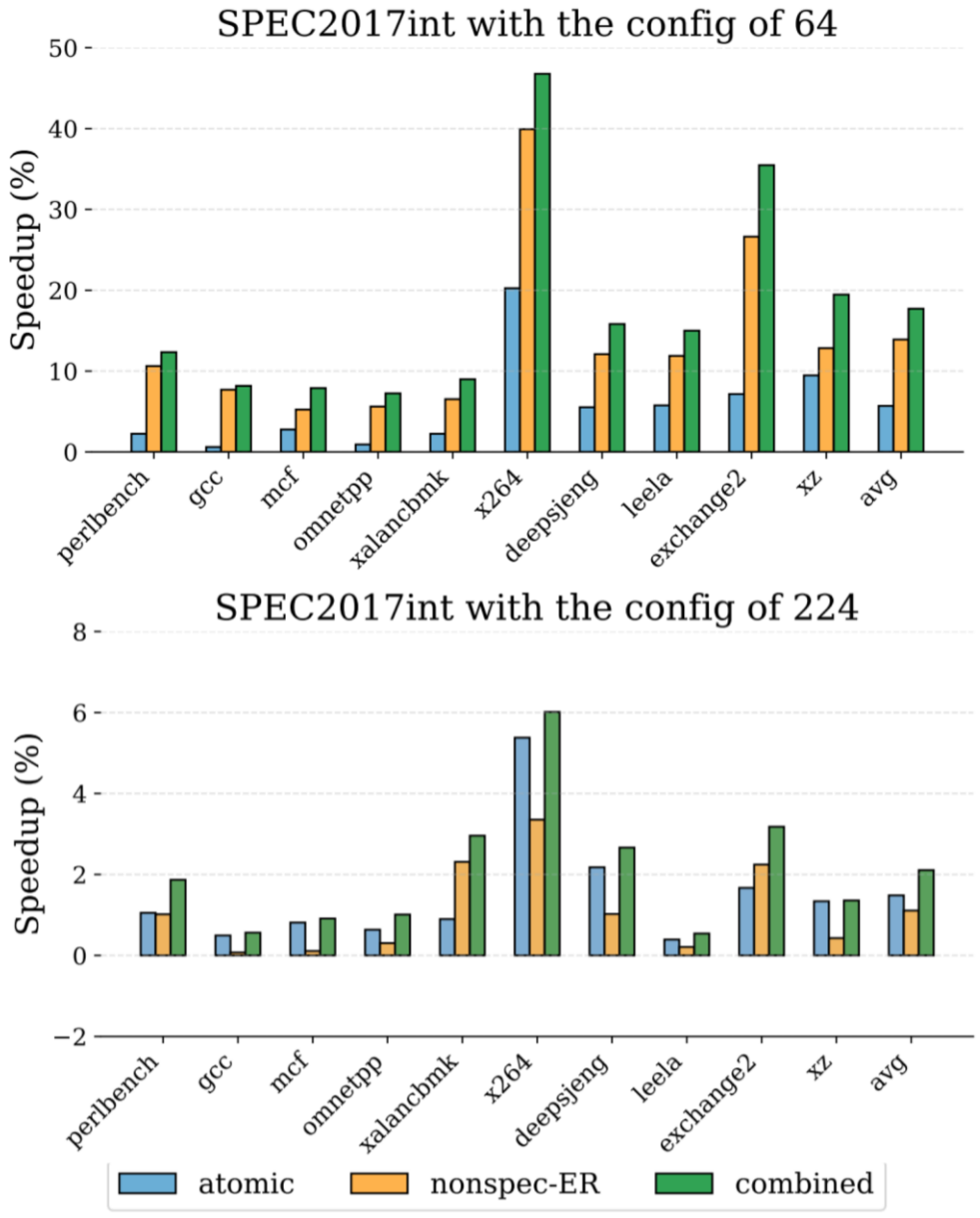
Evaluation

Configuration

- Arch: Golden_Cove
- Dataset: SPEC2017int

Results

- @64
  - atomic: + 5.70% over baseline
  - combine: + 3.23% over nonspec
- @224
  - atomic: + 1.48% over baseline
    + 0.37% over nonspec



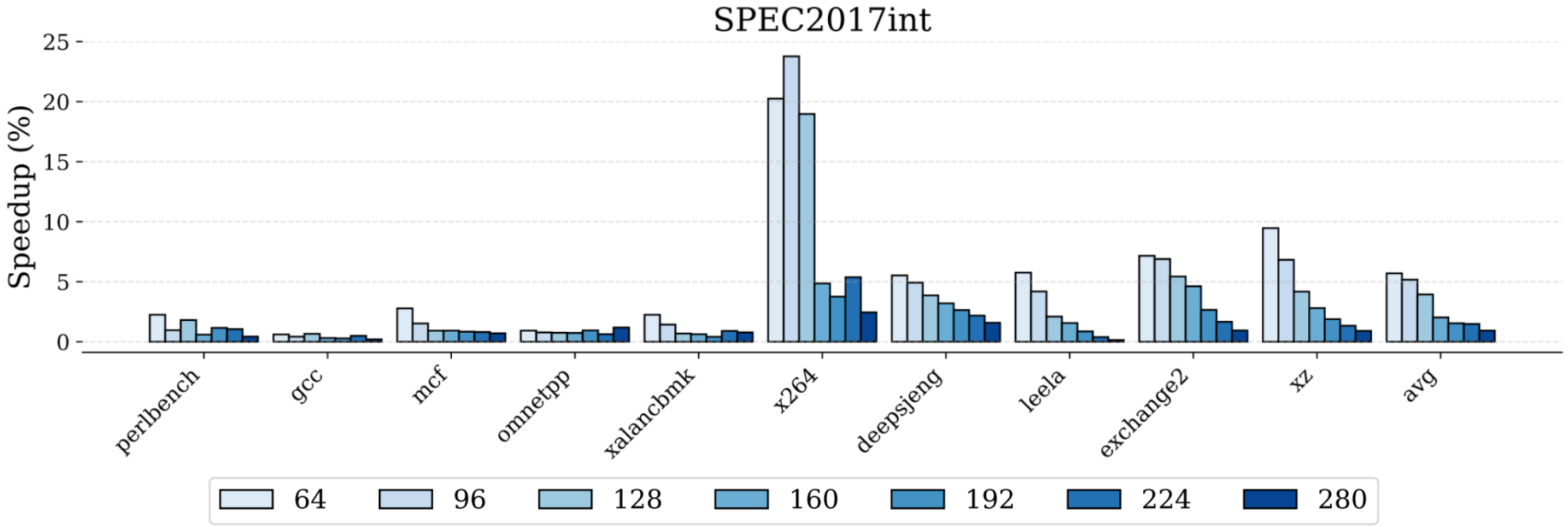
Evaluation

Configuration

- Arch: Golden_Cove
- Dataset: SPEC2017int

Results

- The speedup decreases as the register size increases



Thank you!