Exploring Instruction Fusion Opportunities in General Purpose Processors

Sawan Singh[1] Arthur Perais[2] Alexandra Jimborean[1]
Alberto Ros[1]

[1]Computer Engineering Department
University of Murcia, Spain

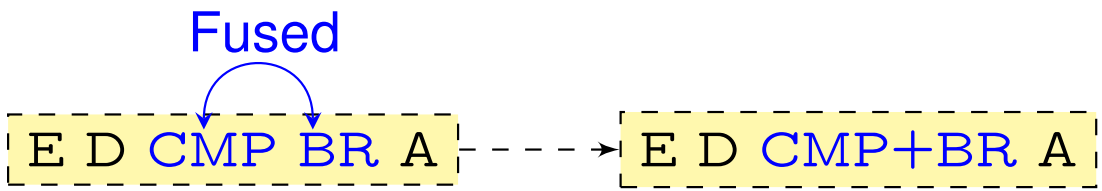
[2]TIMA, Univ. Grenoble Alpes, CNRS,
Grenoble INP, Grenoble, France

MICRO'55, Session 1B, October 3, 2022

Overview

→ Instruction Fusion:

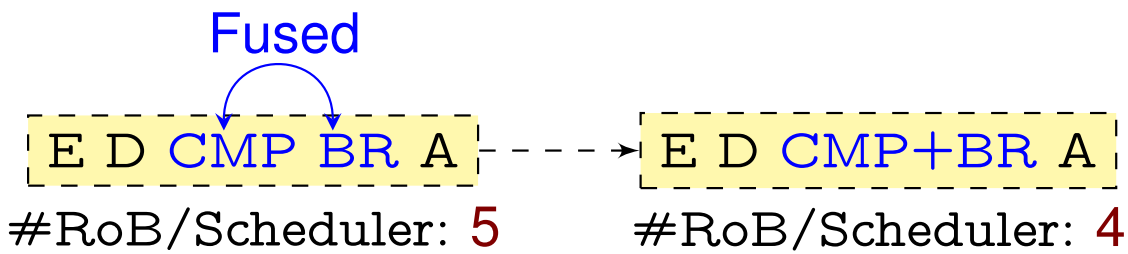
- For example CMP + branch instructions in x86



Overview

→ Instruction Fusion:

- For example CMP + branch instructions in x86
- One Scheduler/RoB entry for two instructions



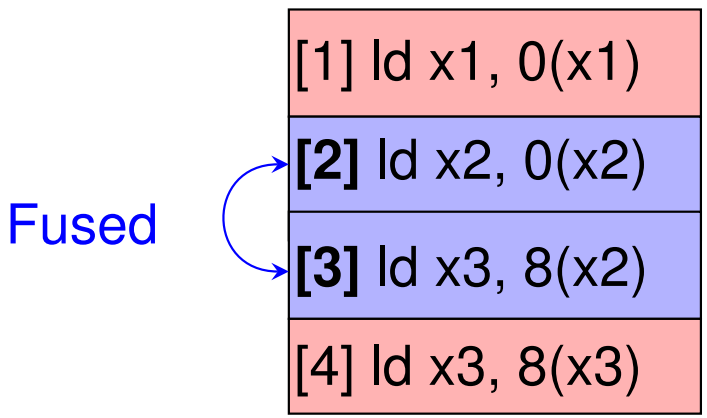
Overview

→ Instruction Fusion:

- For example CMP + branch instructions in x86
- One Scheduler/RoB entry for two instructions

→ State of the art:

- Consecutive instructions in the instruction stream



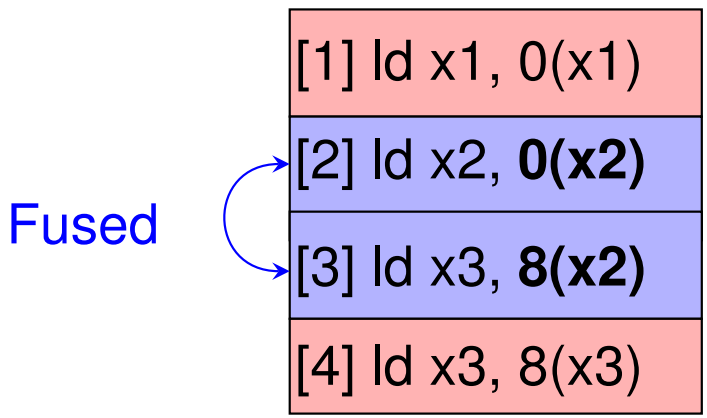
Overview

→ Instruction Fusion:

- For example CMP + branch instructions in x86
- One Scheduler/RoB entry for two instructions

→ State of the art:

- Consecutive instructions in the instruction stream
- Accessing Contiguous memory locations (memory μ-ops)



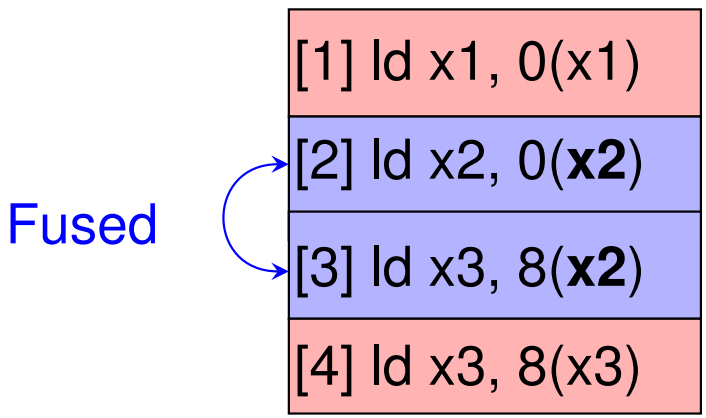
Overview

→ Instruction Fusion:

- For example CMP + branch instructions in x86
- One Scheduler/RoB entry for two instructions

→ State of the art:

- Consecutive instructions in the instruction stream
- Accessing Contiguous memory locations (memory μ-ops)
- Same base register (memory μ-ops)



Overview

→ **Instruction Fusion:**

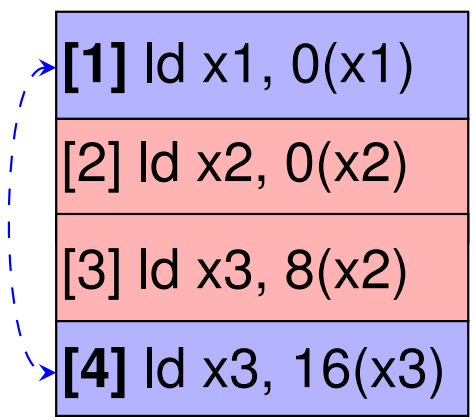
- For example CMP + branch instructions in x86
- One Scheduler/RoB entry for two instructions

→ **State of the art:**

- Consecutive instructions in the instruction stream
- Accessing Contiguous memory locations (memory μ-ops)
- Same base register (memory μ-ops)

→ **Helios:** Predict and fuse distant memory instructions:

- In code, non-consecutive instructions



Overview

→ **Instruction Fusion:**

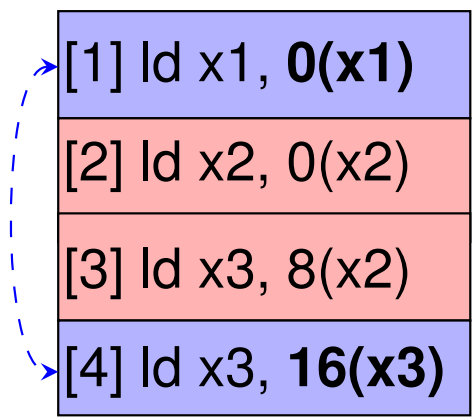
- For example CMP + branch instructions in x86
- One Scheduler/RoB entry for two instructions

→ **State of the art:**

- Consecutive instructions in the instruction stream
- Accessing Contiguous memory locations (memory μ-ops)
- Same base register (memory μ-ops)

→ **Helios:** Predict and fuse distant memory instructions:

- In code, non-consecutive instructions
- In memory, non-contiguous locations



Overview

→ Instruction Fusion:

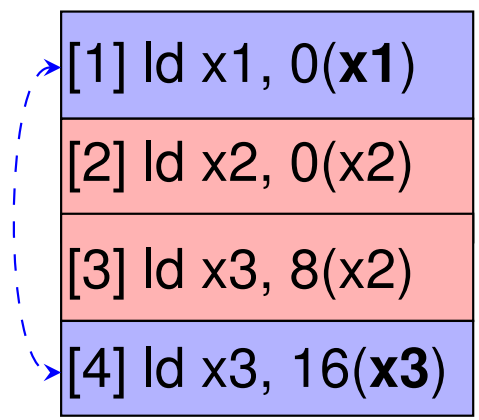
- For example CMP + branch instructions in x86
- One Scheduler/RoB entry for two instructions

→ State of the art:

- Consecutive instructions in the instruction stream
- Accessing Contiguous memory locations (memory μ-ops)
- Same base register (memory μ-ops)

→ Helios: Predict and fuse distant memory instructions:

- In code, non-consecutive instructions
- In memory, non-contiguous locations
- In similarity, different base register



Outline

Background & Motivation

→ RISC-V Fusion:

- Simple ISA = Numerous fusion opportunities

Background & Motivation

→ RISC-V Fusion:

- Simple ISA = Numerous fusion opportunities
- Various consecutive pairs proposed by Celio *et al.* [1]
  - ⋆ Contiguous memory pairs (*load-load* & *store-store*)
  - ⋆ Other pairs (*slli-srli*, *add-ld* ...)

[1] Celio *et al.*: **"... Macro-op Fusion for RISC-V.** *arXiv 2016.*

Background & Motivation

→ RISC-V Fusion:

- Simple ISA = Numerous fusion opportunities
- Various consecutive pairs proposed by Celio *et al.* [1]
  - ⋆ Contiguous memory pairs (*load-load* & *store-store*) →5.6%
  - ⋆ Other pairs (*slli-srli*, *add-ld* ...) →1.1%

[1] Celio *et al.*: **"... Macro-op Fusion for RISC-V.** *arXiv 2016.*

Background & Motivation

→ RISC-V Fusion:

- Simple ISA = Numerous fusion opportunities
- Various consecutive pairs proposed by Celio *et al.* [1]
  - ⋆ Contiguous memory pairs (*load-load* & *store-store*) →5.6%
  - ⋆ Other pairs (*slli-srli*, *add-ld* ...) →1.1%

→ Observations:

[1] Celio *et al.*: **"... Macro-op Fusion for RISC-V.** *arXiv 2016.*

Background & Motivation

→ RISC-V Fusion:

- Simple ISA = Numerous fusion opportunities
- Various consecutive pairs proposed by Celio *et al.* [1]
  - ⋆ Contiguous memory pairs (*load-load* & *store-store*) →5.6%
  - ⋆ Other pairs (*slli-srli*, *add-ld* ...) →1.1%

→ Observations:

⋆ Focus on Memory Fusion:

- Majority of fused μ-ops are memory instructions
- IPC improvement is similar when fusing all pairs

[1] Celio *et al.*: **"... Macro-op Fusion for RISC-V.** *arXiv 2016.*

Background & Motivation

→ RISC-V Fusion:

- Simple ISA = Numerous fusion opportunities
- Various consecutive pairs proposed by Celio *et al.* [1]
  - ⋆ Contiguous memory pairs (*load-load* & *store-store*) →5.6%
  - ⋆ Other pairs (*slli-srli*, *add-ld* ...) →1.1%

→ Observations:

⋆ Focus on Memory Fusion:

- Majority of fused μ-ops are memory instructions
- IPC improvement is similar when fusing all pairs

⋆ Missing Opportunities:

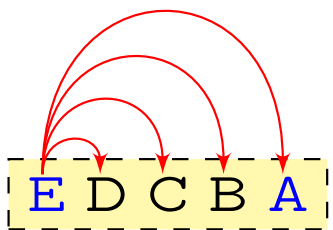
- Non-consecutive & non-contiguous memory fusion
- Doubles the number of fused memory pairs (+7.58%)

[1] Celio *et al.*: **"... Macro-op Fusion for RISC-V.** *arXiv 2016.*

BACKGROUND & MOTIVATION

→ NON-CONSECUTIVE FUSION CHALLENGES

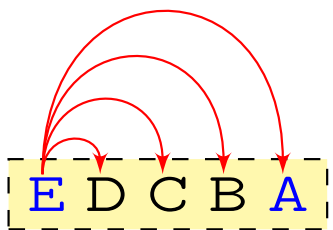
- FINDING PAIRS: Non-consecutive fusion does not scale

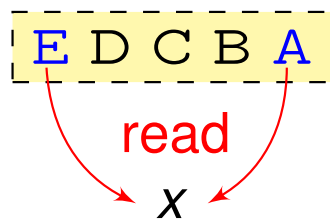


Background & Motivation

→ Non-consecutive fusion challenges

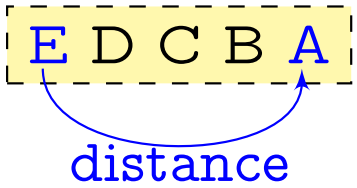
- Finding pairs: Non-consecutive fusion does not scale
- Memory access: Base register might be modified by another instruction





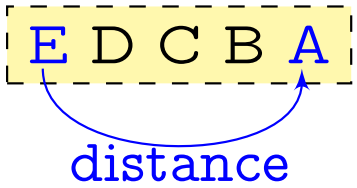
Outline

HELIOS



- → Fusion distance predictor
- → Fuse only memory instructions
- → Implement a *tournament-like* fusion predictor

HELIOS



- → Fusion distance predictor
- → Fuse only memory instructions
- → Implement a *tournament-like* fusion predictor
- → Predict instructions:
  - ⋆ Non-Consecutive
  - ⋆ Non-Contiguous
  - ⋆ Different base register

HELIOS

→ Helios consists of 2 structures: the Unfused Committed History (UCH) and the Fusion Predictor (FP)

HELIOS

→ Helios consists of 2 structures: the Unfused Committed History (UCH) and the Fusion Predictor (FP)

- UNFUSED COMMITTED HISTORY:
  - ⋆ Save the sequence number of not-already-fused μ-ops
  - ⋆ Separate history for Loads (6 entries) and Stores (1 entry)
  - ⋆ Fills the Fusion Predictor

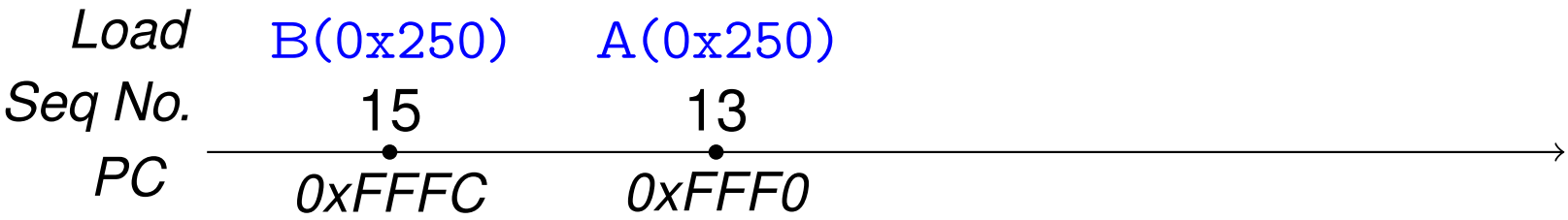
HELIOS

→ Helios consists of 2 structures: the Unfused Committed History (UCH) and the Fusion Predictor (FP)

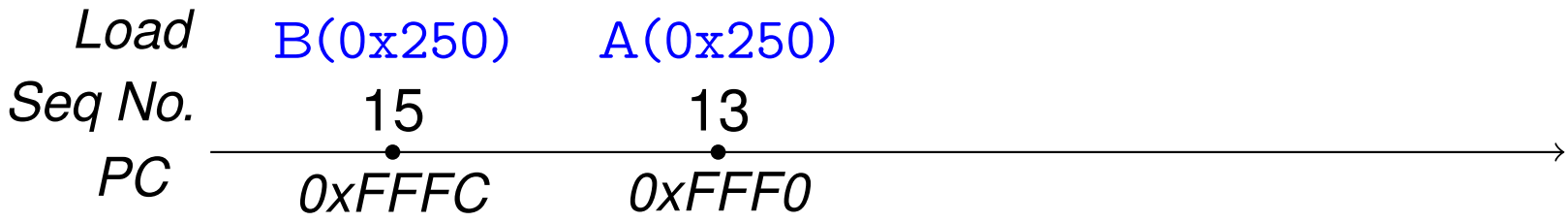
- UNFUSED COMMITTED HISTORY:
  - ⋆ Save the sequence number of not-already-fused μ-ops
  - ⋆ Separate history for Loads (6 entries) and Stores (1 entry)
  - ⋆ Fills the Fusion Predictor

- FUSION PREDICTOR:
  - ⋆ Provide predicted distance to μ-ops at decode
  - ⋆ The predictor's confidence is updated at execute

HELIOS
OVERVIEW



HELIOS
OVERVIEW



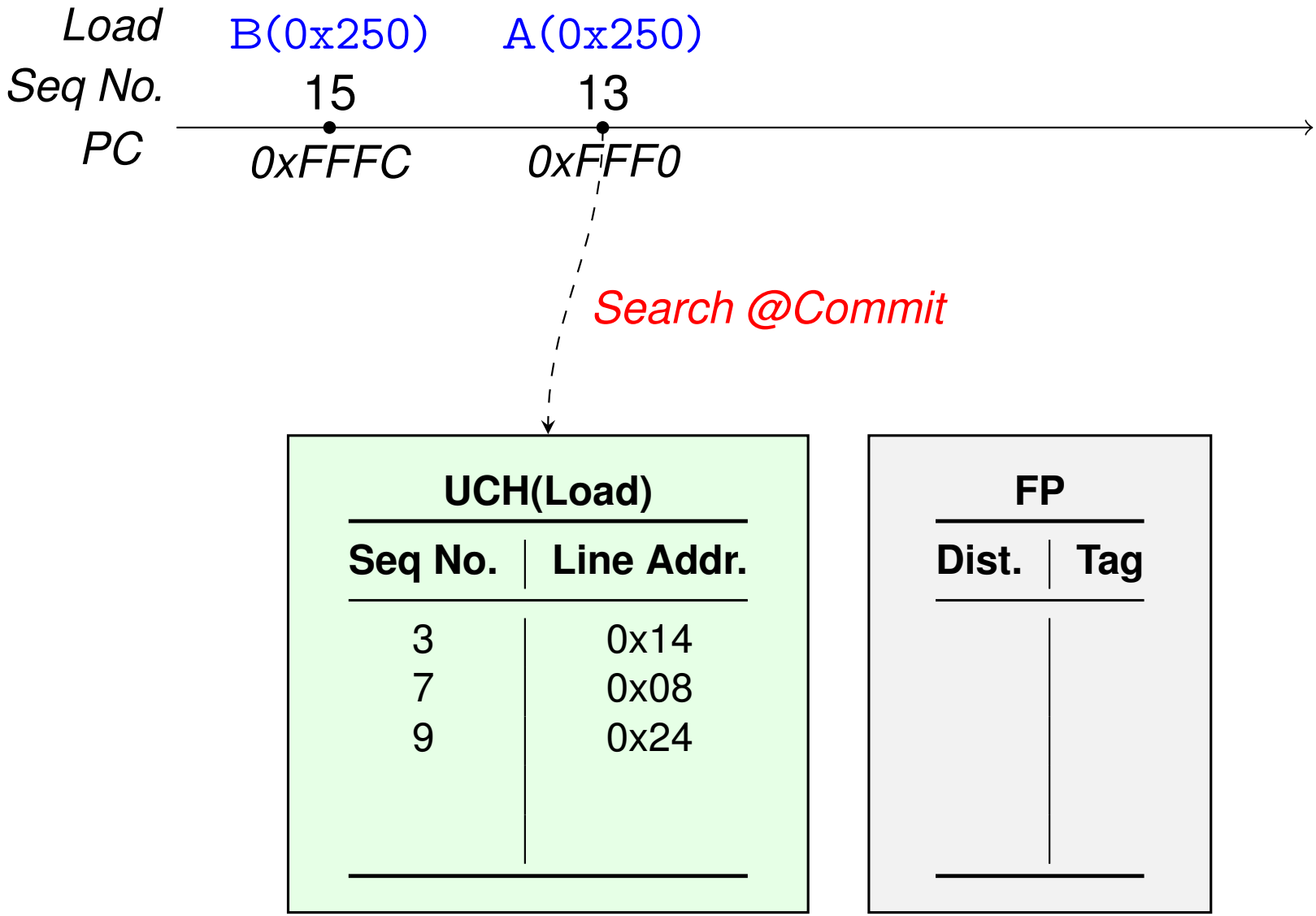
UCH(Load)

Seq No.	Line Addr.
3	0x14
7	0x08
9	0x24

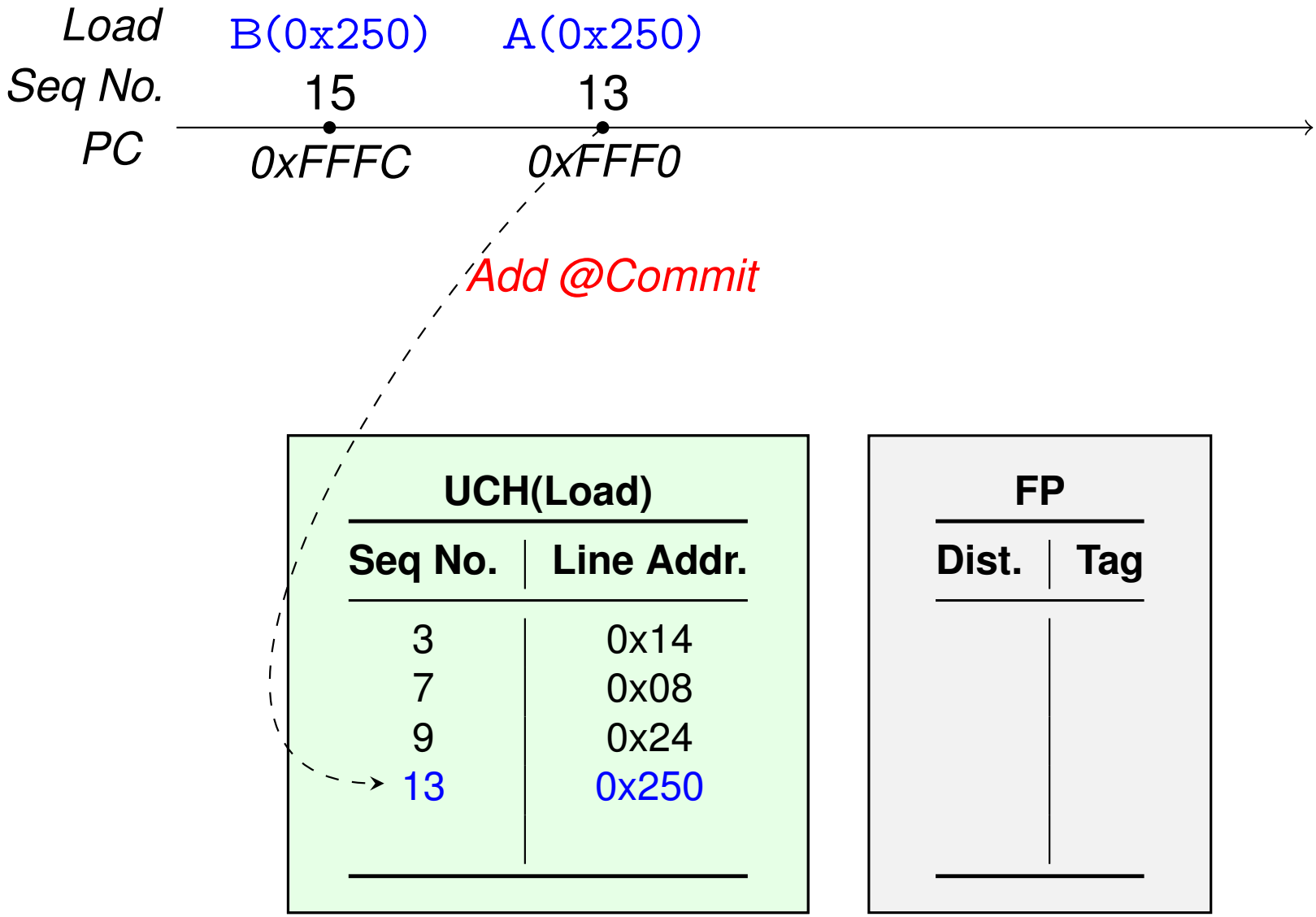
FP

Dist.	Tag

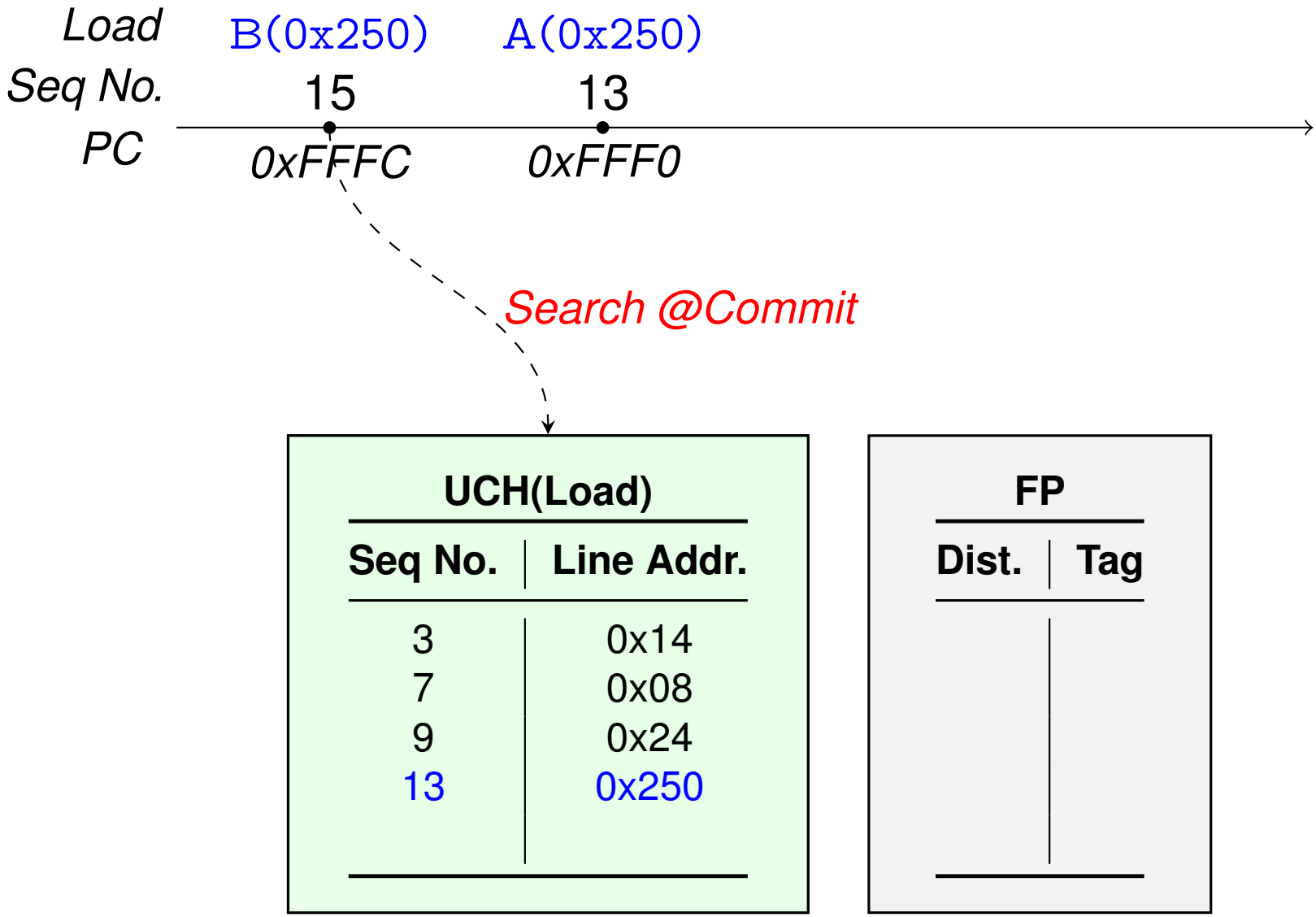
HELIOS
OVERVIEW



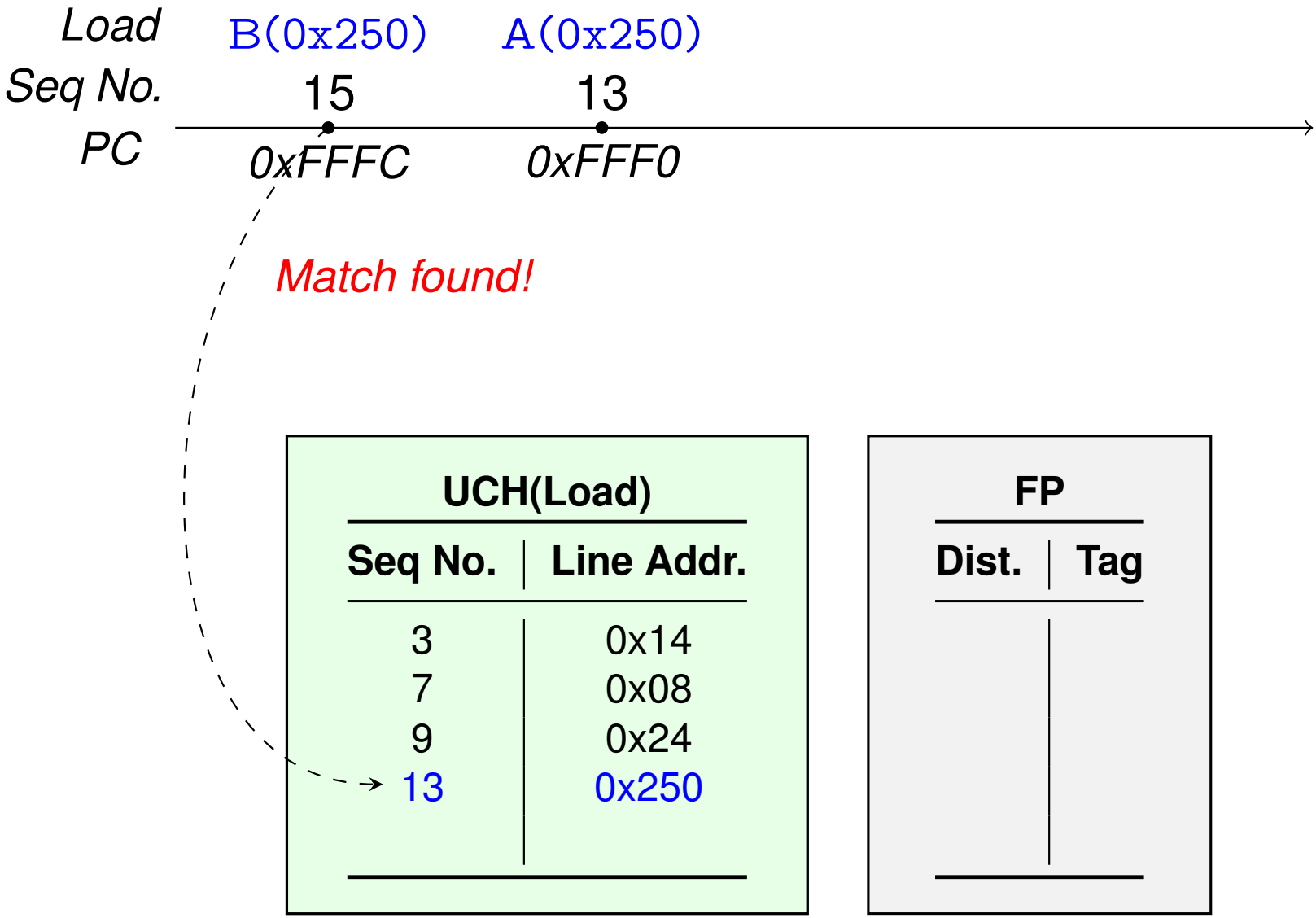
HELIOS
OVERVIEW



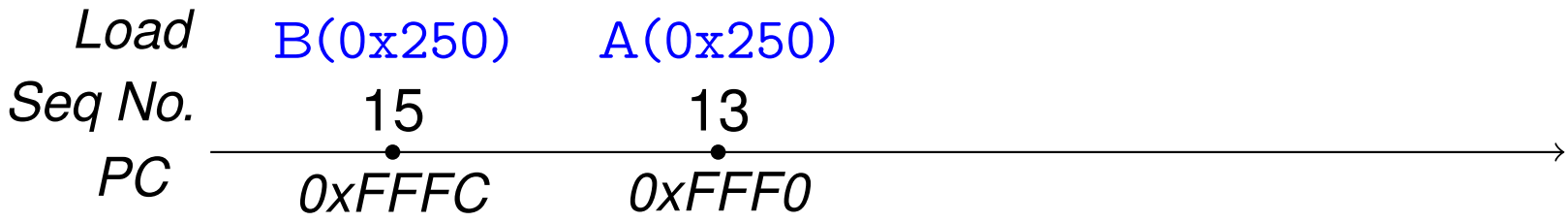
HELIOS
OVERVIEW

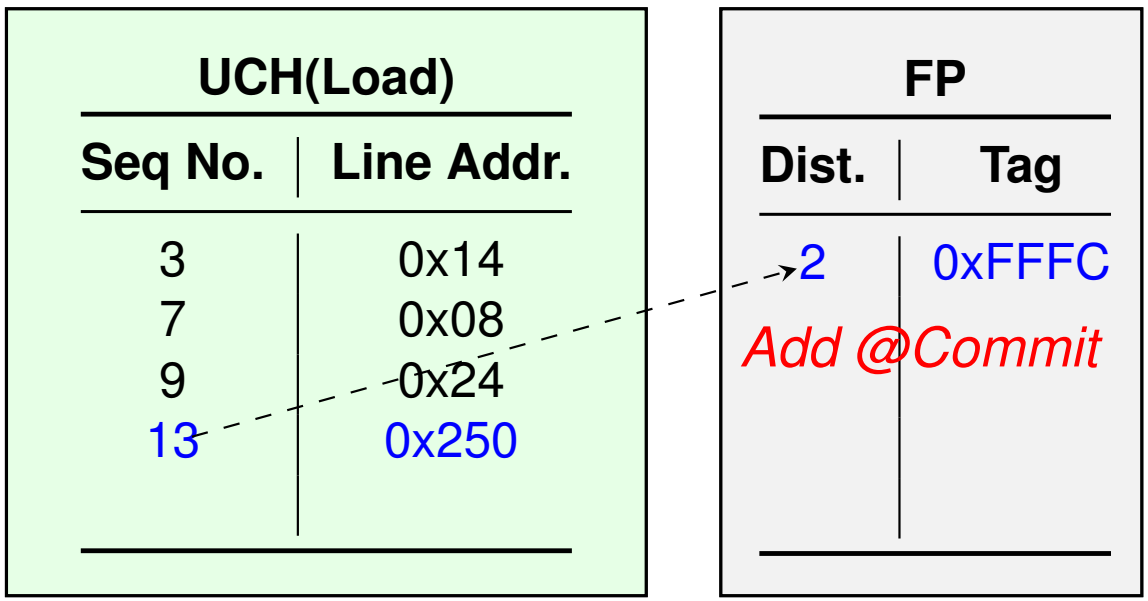


HELIOS
OVERVIEW

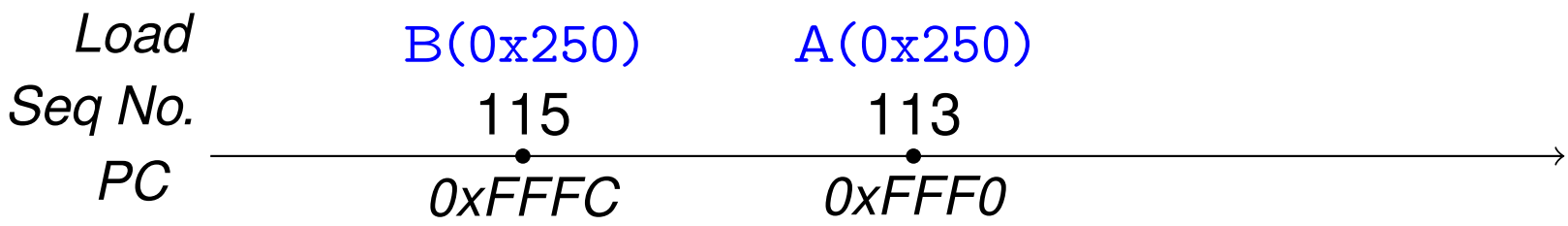


HELIOS
OVERVIEW

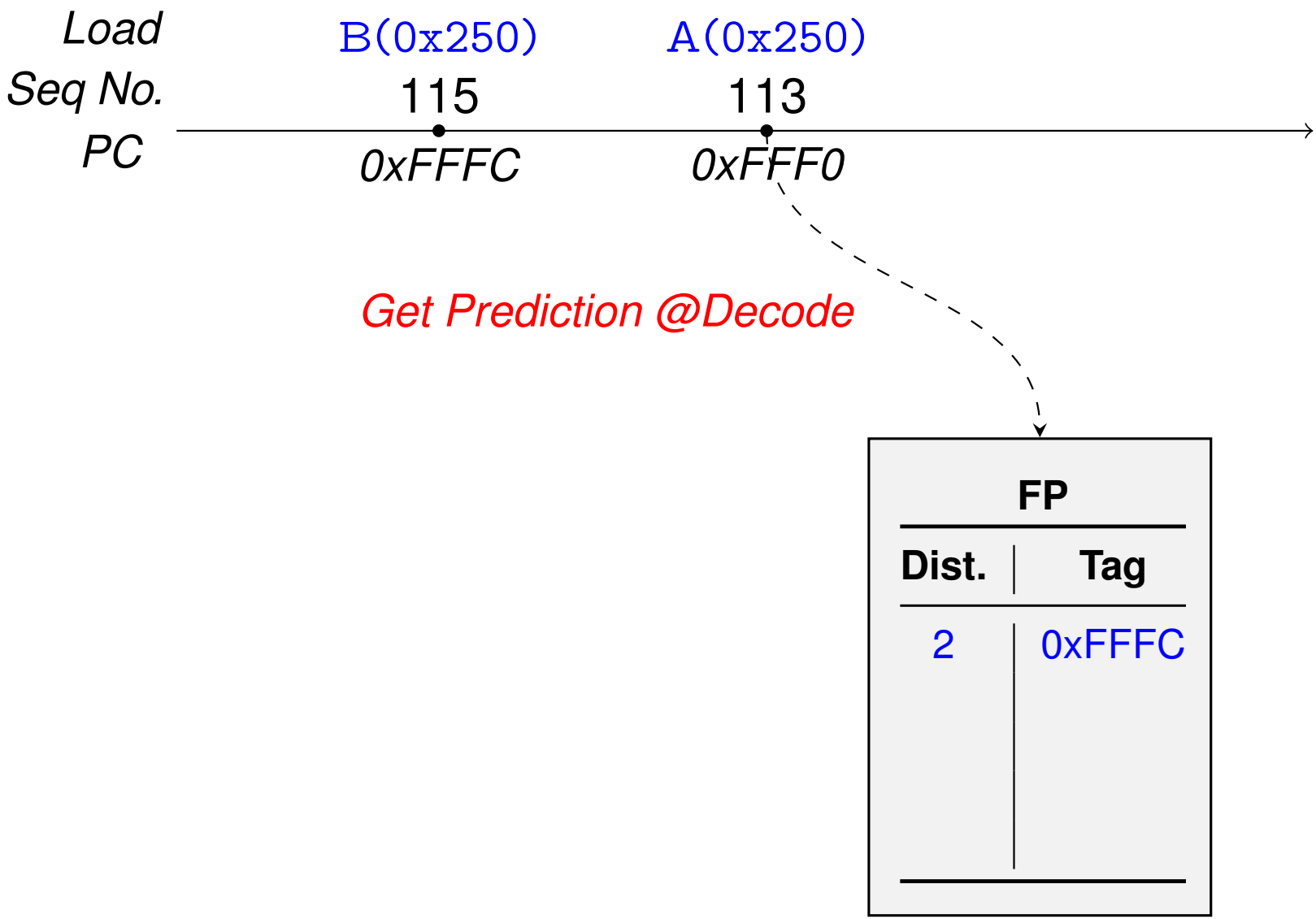




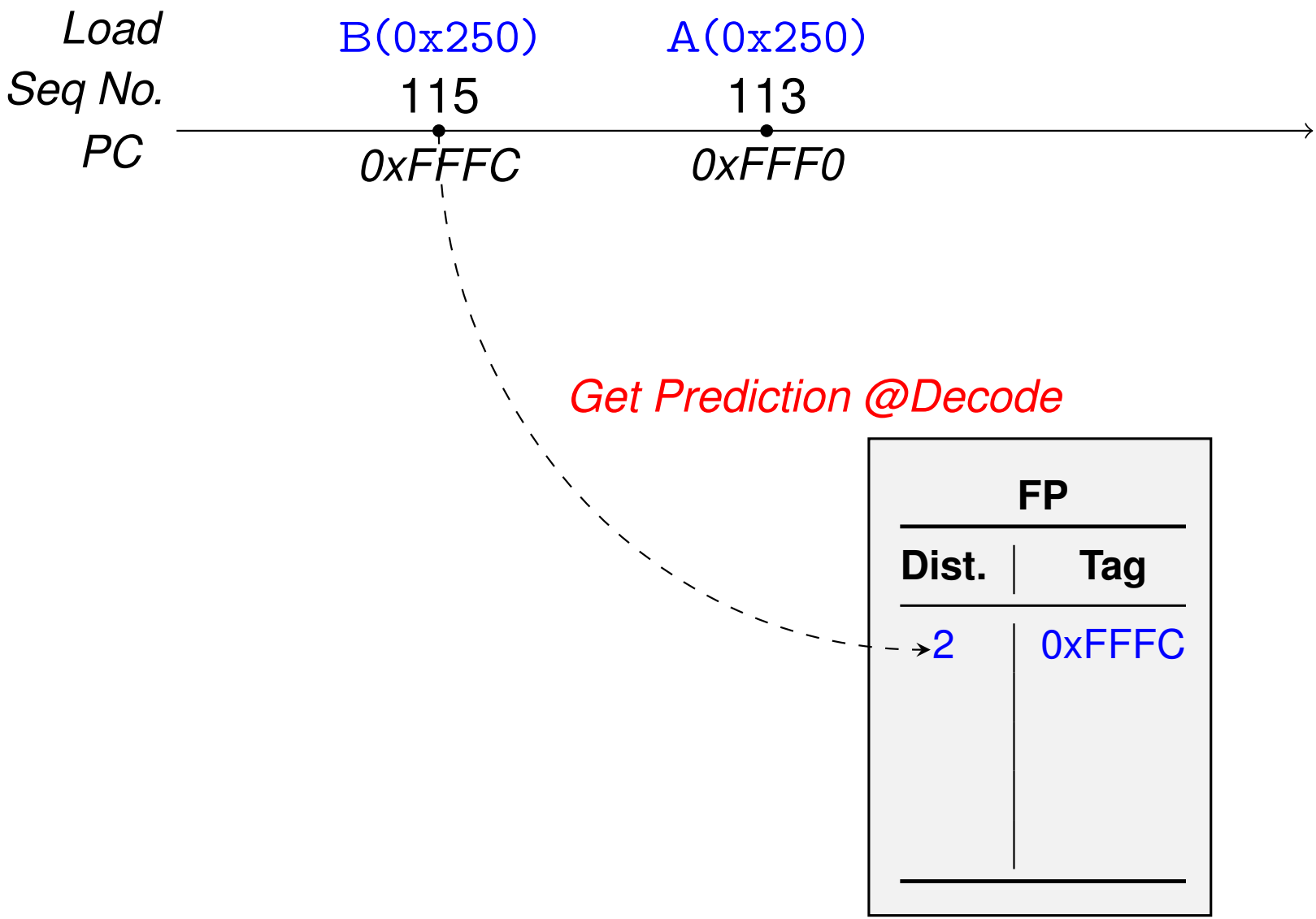
HELIOS
OVERVIEW



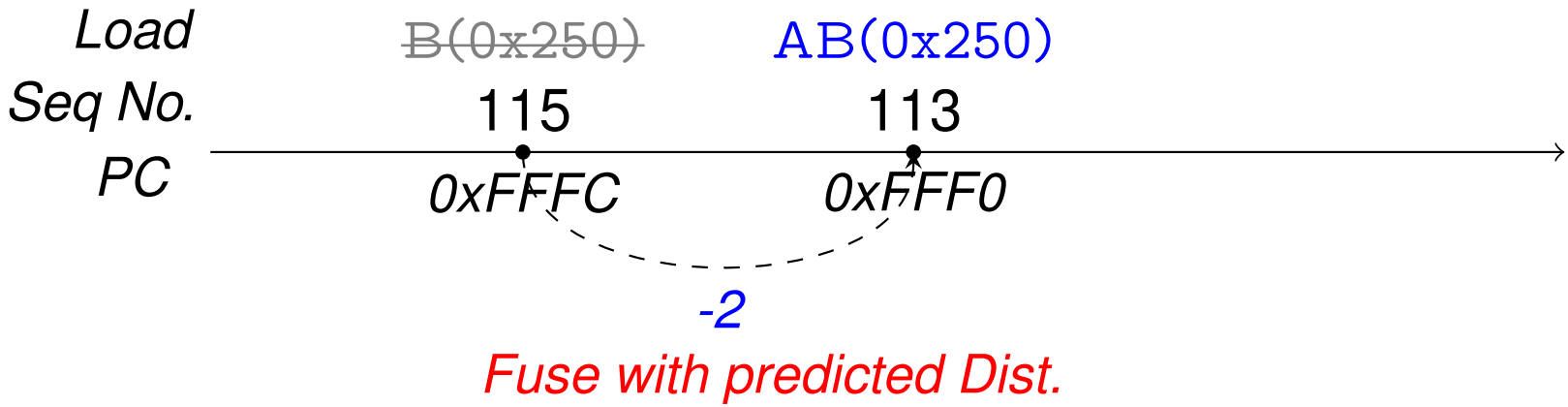
HELIOS
OVERVIEW



HELIOS
OVERVIEW



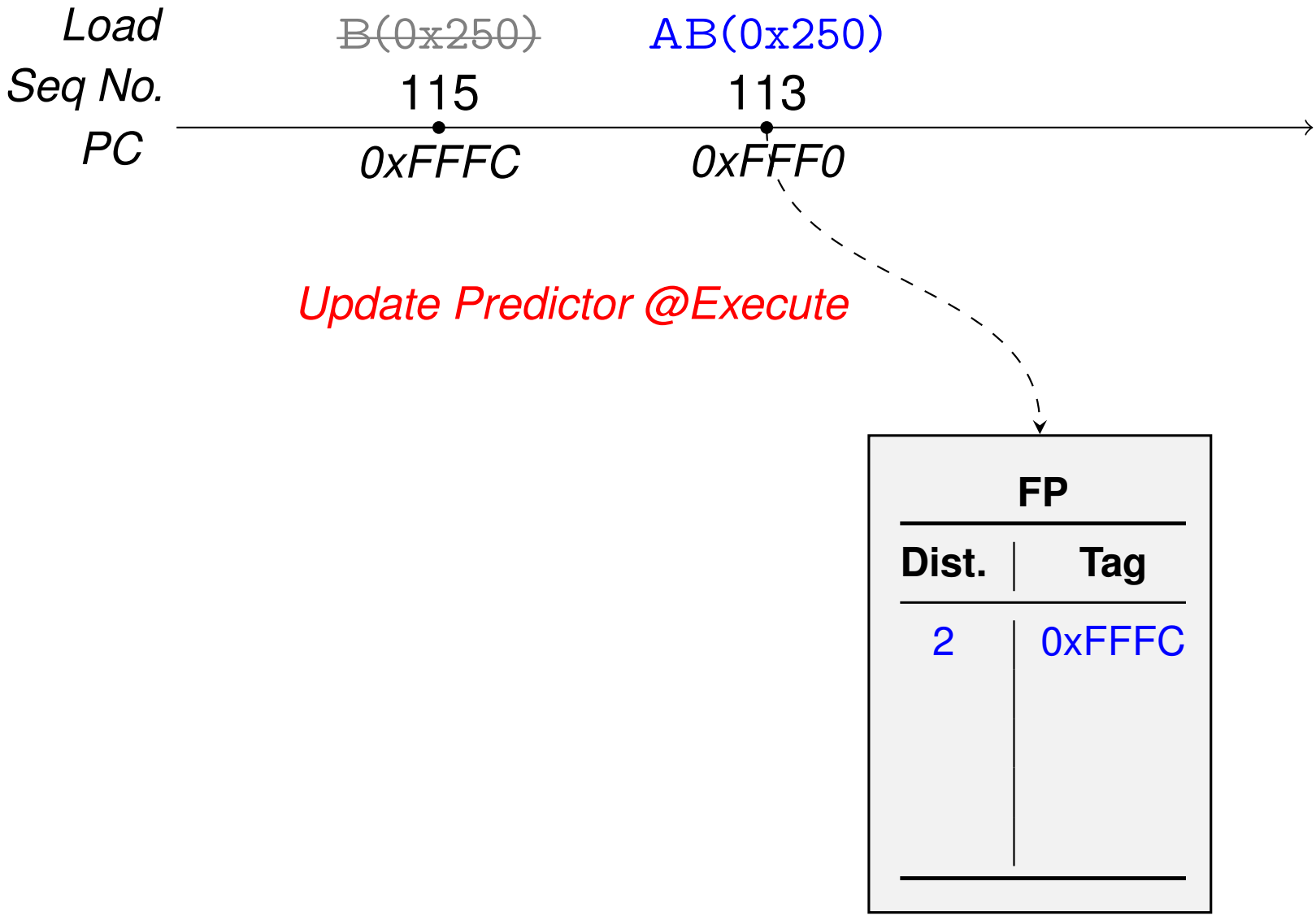
HELIOS
OVERVIEW



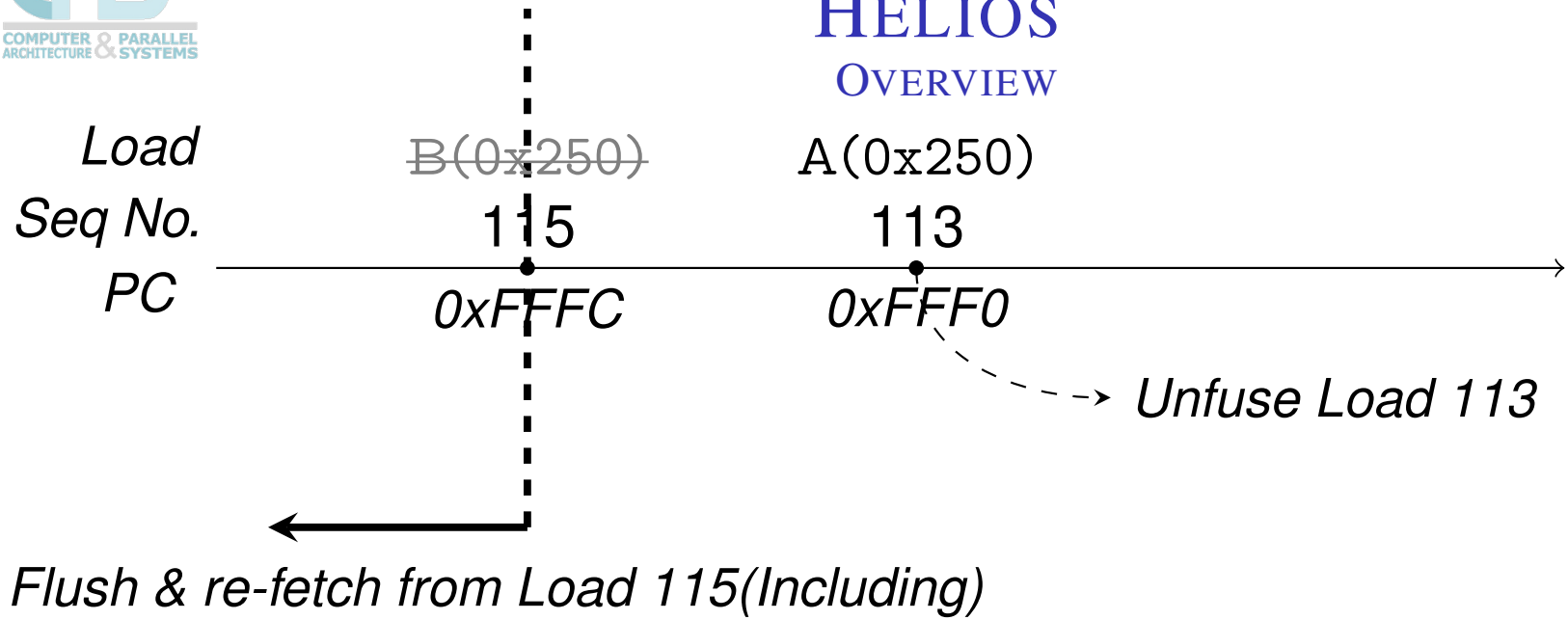
Fuse with predicted Dist.

FP	
Dist.	**Tag**
2	0xFFFC

HELIOS
OVERVIEW



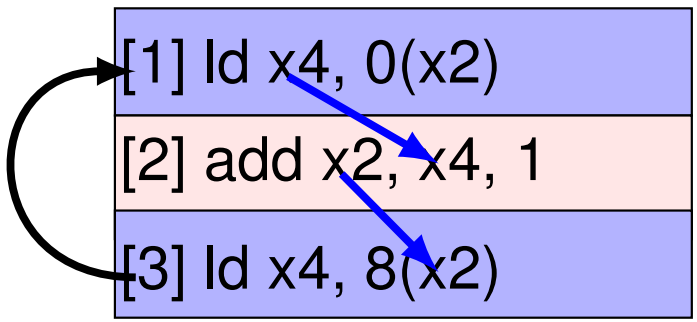
HELIOS
OVERVIEW



If mispredicted

FP	
Dist.	**Tag**
2	0xFFFC

→ DEADLOCKS: Cyclic dependencies can result in deadlock



HELIOS
CHALLENGES

→ DEADLOCKS: Cyclic dependencies can result in deadlock

→ CONSISTENCY: Fusing non-consecutive memory μ-ops may break memory order

HELIOS
CHALLENGES

→ DEADLOCKS: Cyclic dependencies can result in deadlock

→ CONSISTENCY: Fusing non-consecutive memory μ-ops may break memory order

Details in the paper!

Outline

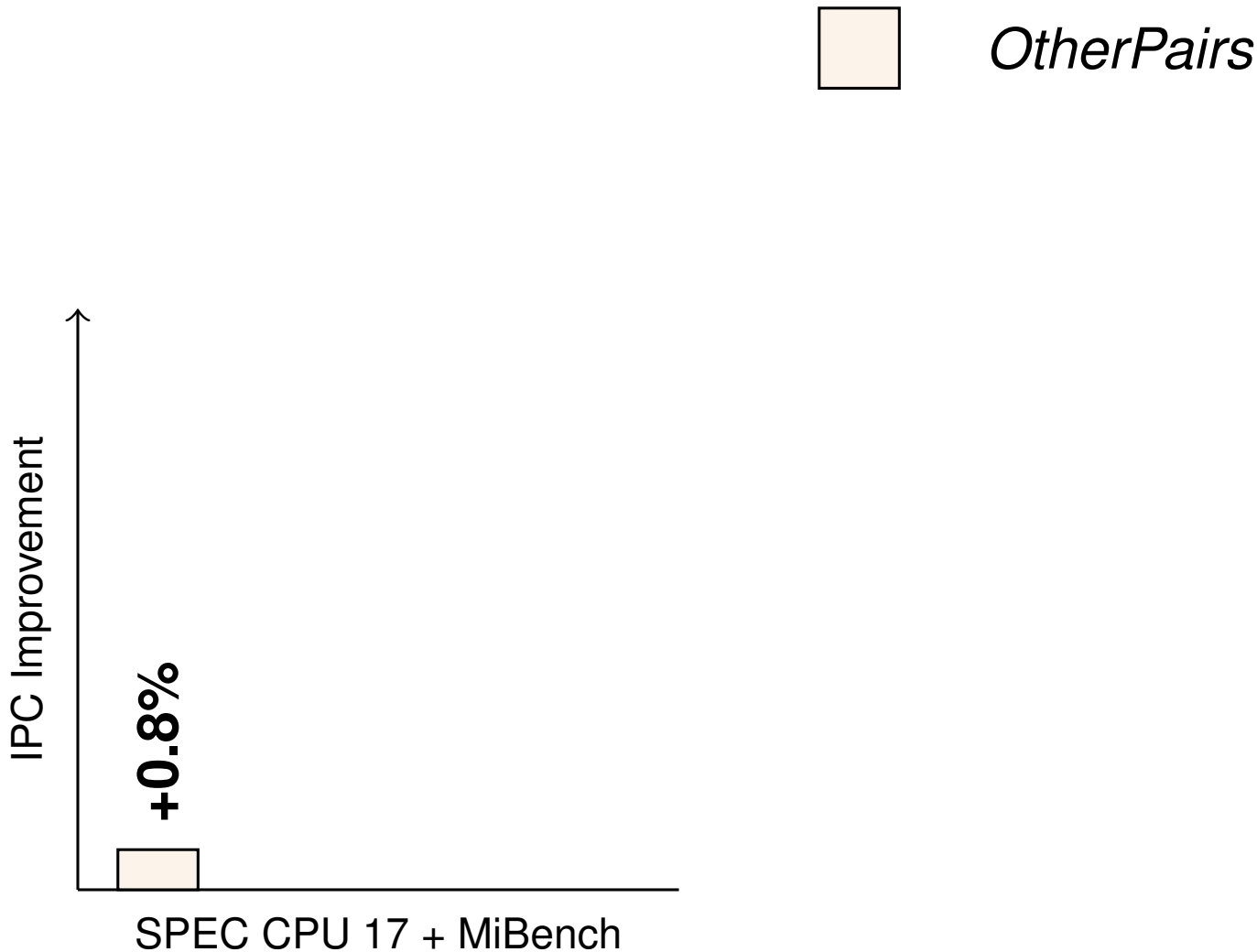
Methodology & Results

→ Spike + in house OoO core (Intel Icelake)

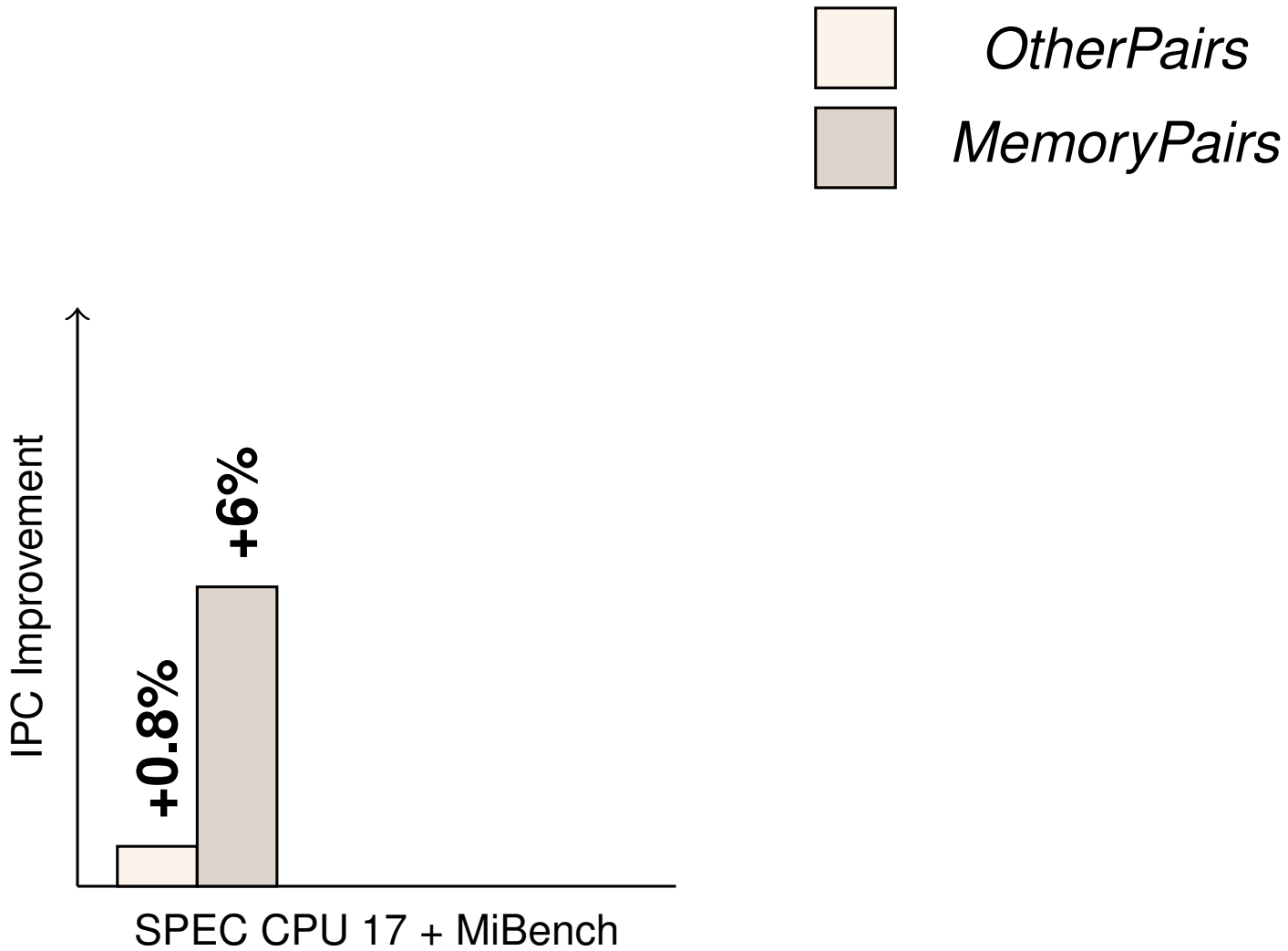
→ SPEC CPU 2017 + MiBench applications

- *SPEC CPU 2017:* Skip 10B, collect results for 500M
- *MiBench:* Run from begin to end

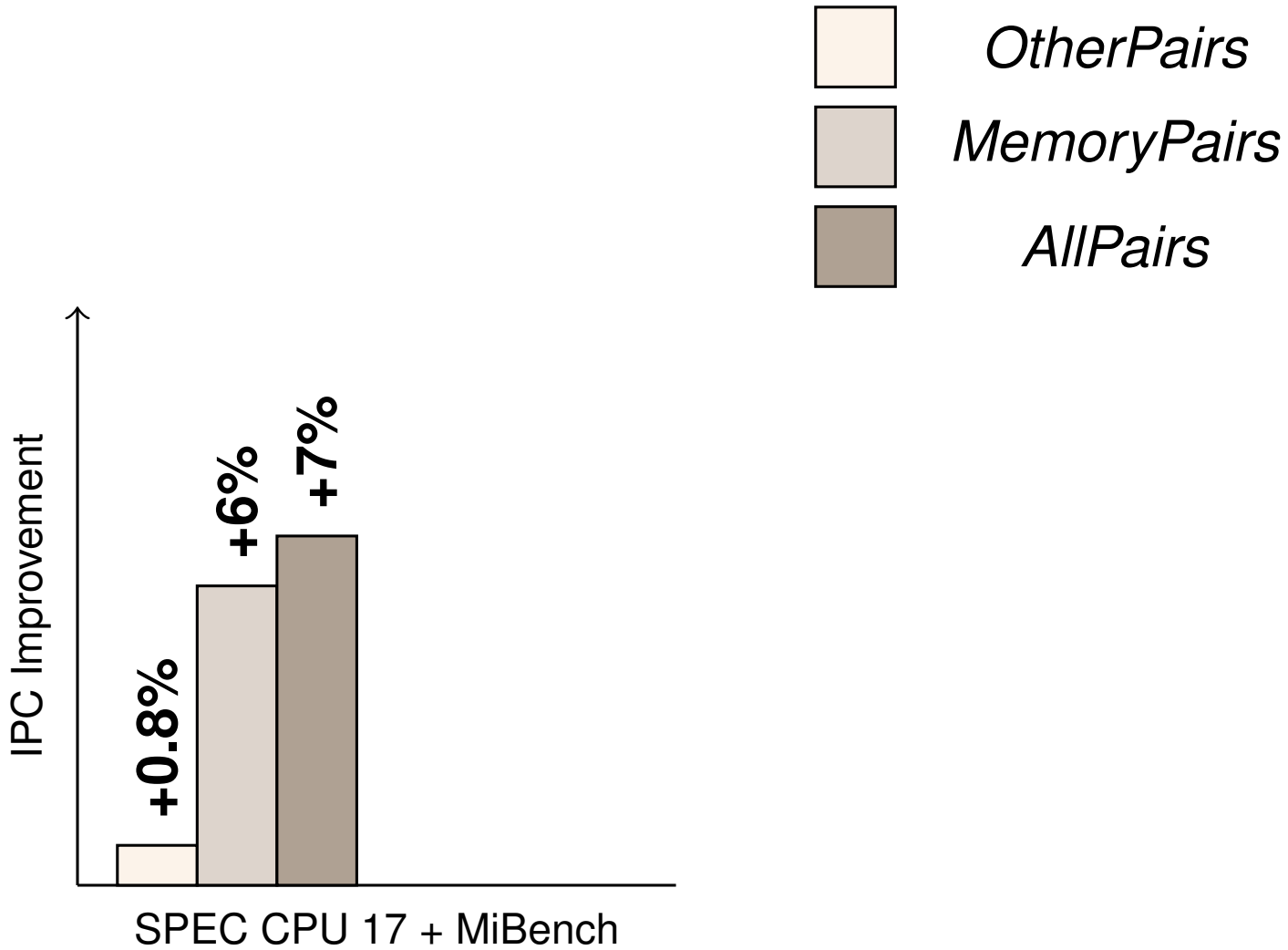
METHODOLOGY & RESULTS



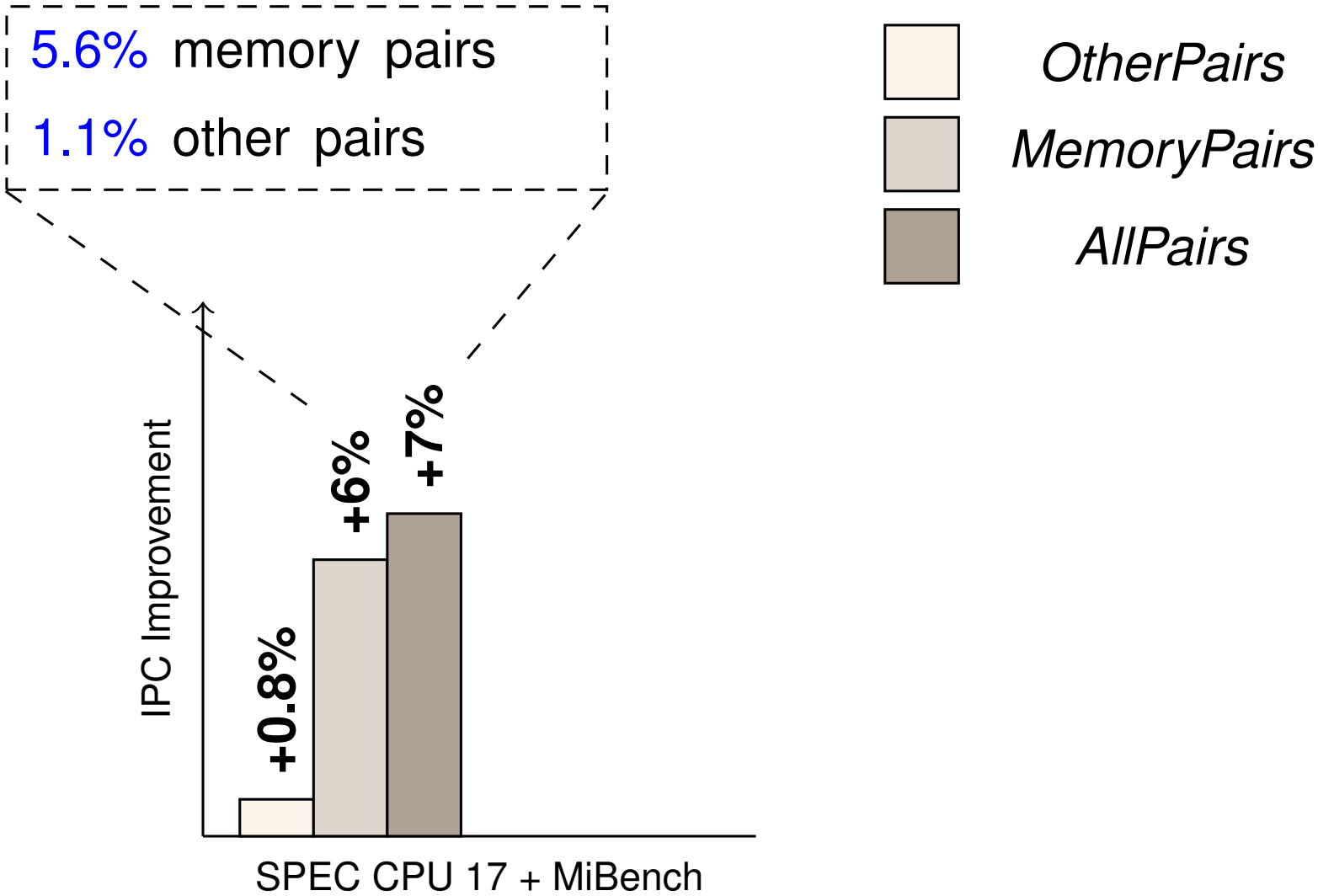
Methodology & Results



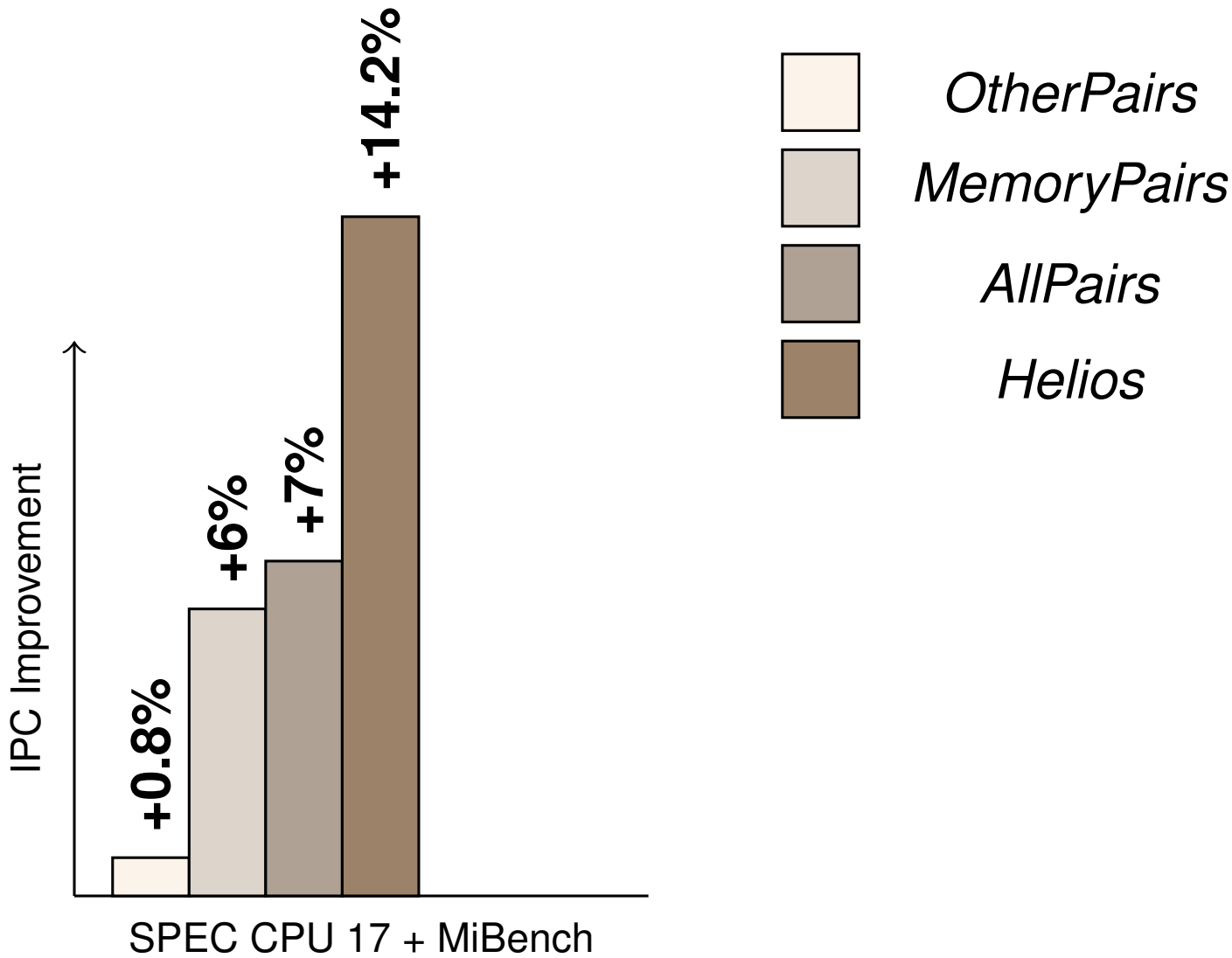
METHODOLOGY & RESULTS



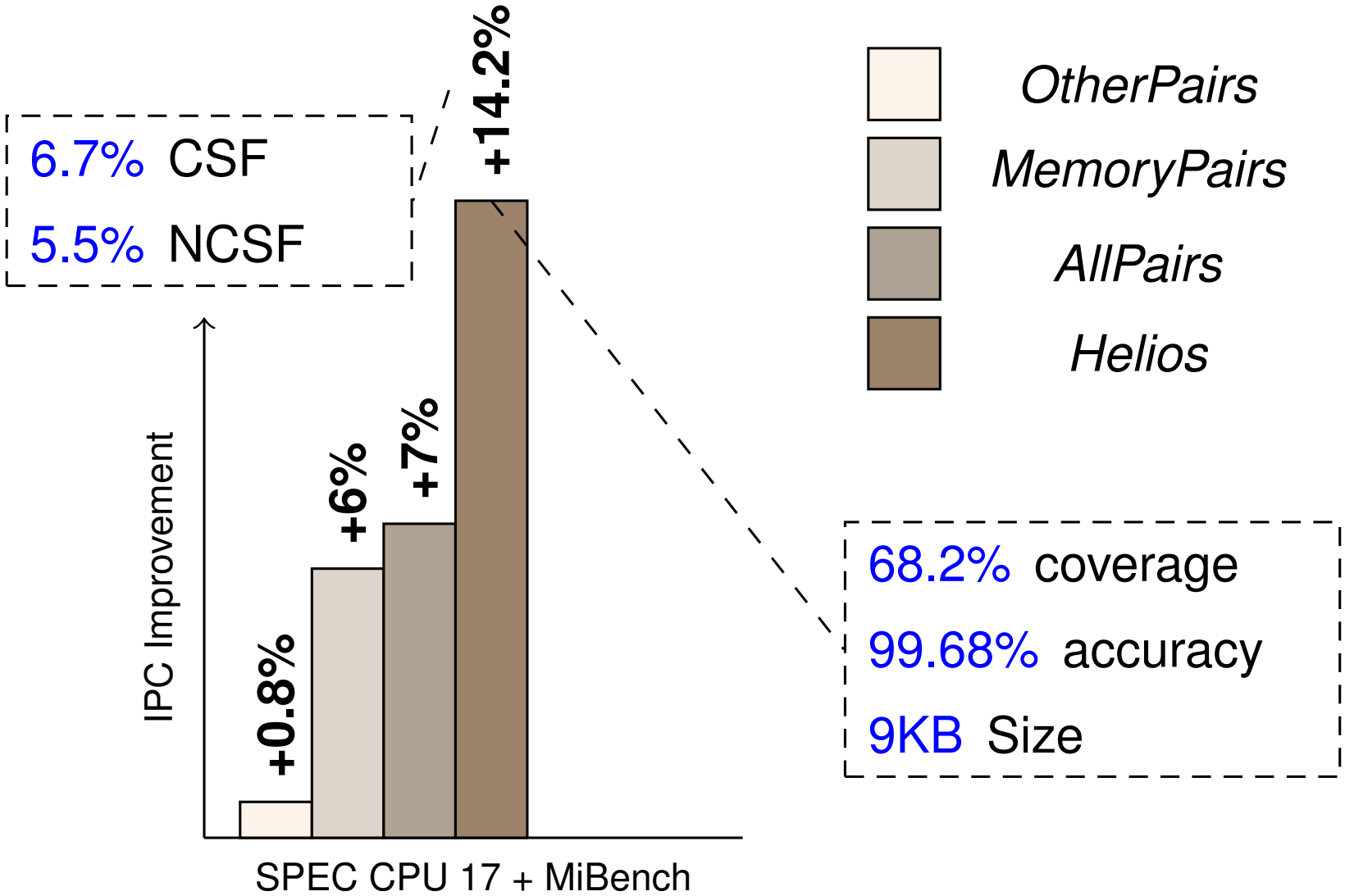
Methodology & Results



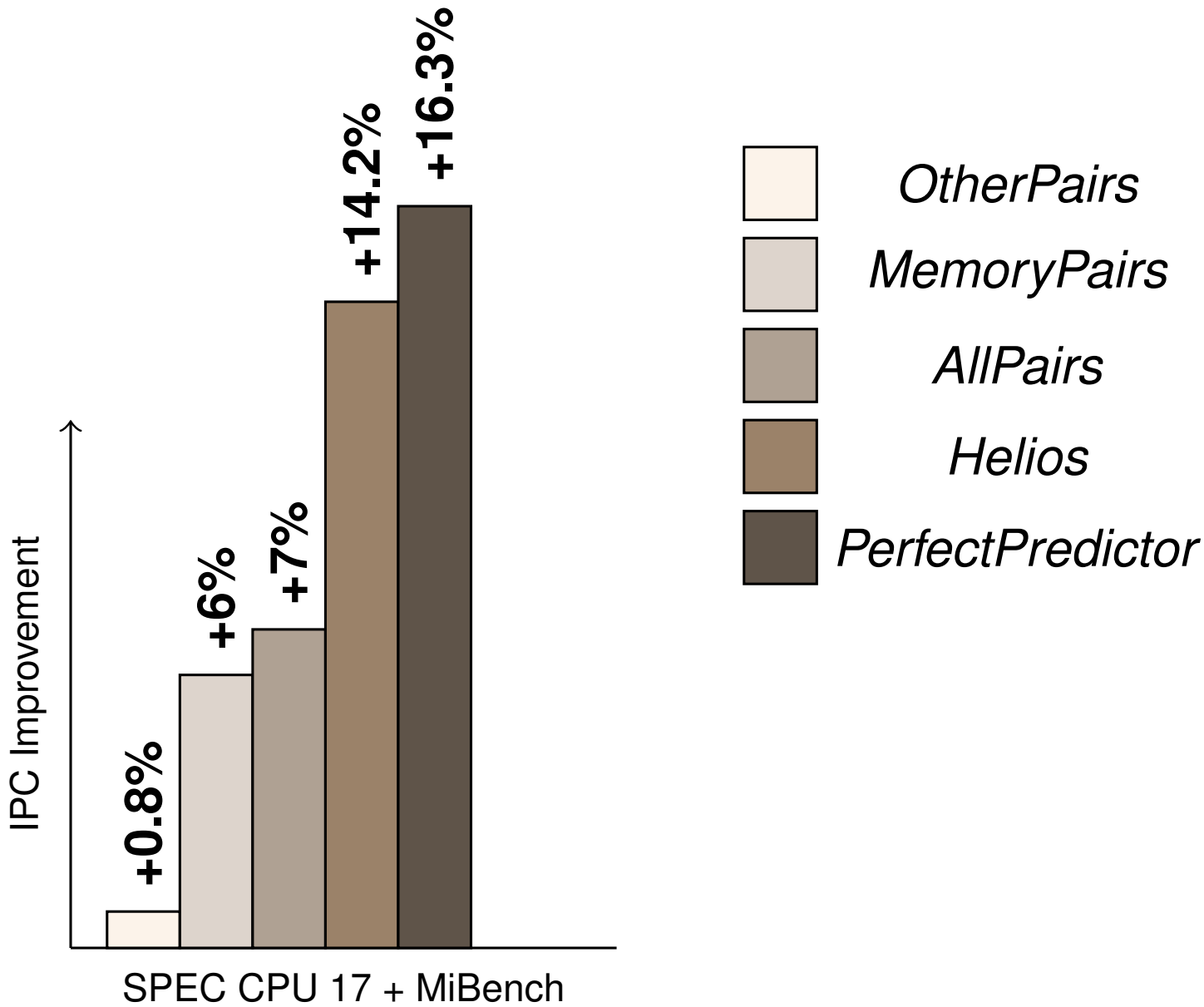
METHODOLOGY & RESULTS



Methodology & Results



METHODOLOGY & RESULTS



Outline

CONCLUSIONS

- HELIOS: First detailed design proposal to fuse memory instructions:-
  - ⋆ Non-consecutive
  - ⋆ Non-contiguous
  - ⋆ Different base register
- Easy to scale

Exploring Instruction Fusion Opportunities in General Purpose Processors

Sawan Singh[1] Arthur Perais[2] Alexandra Jimborean[1]
Alberto Ros[1]

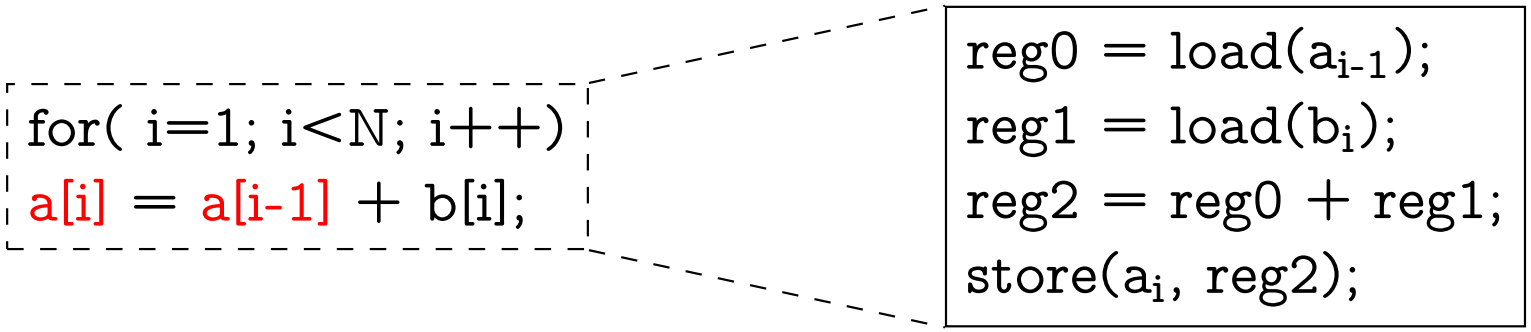
singh.sawan@um.es

Thank you for your attention!

ECHO, ERC Consolidator Grant (No 819134)

BACKUP SLIDES!

HELIOS VS VECTORIZATION



→ Loop dependencies can make code not vectorizable

→ Helios can predict and fuse $\texttt{load}(a_{i-1})$ with $\texttt{load}(b_i)$

→ Helios can also try to fuse $\texttt{load}(a_{i-1})$, $\texttt{load}(b_i)$ & $\texttt{store}(a_i, \texttt{reg2})$ with itself from the previous or next iteration over the branch