Architecting Value Prediction around In-Order Execution

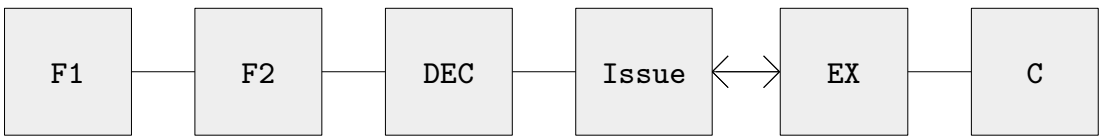
IEEE HPCA - March 1-5, 2025

Pierre RAVENEL[†‡] **Arthur PERAIS**[†] **Frédéric PETROT**[†] **Benoît DUPONT DE DINECHIN**[‡]

† Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA
‡ Kalray

INTRODUCTION

Can InO Execution with Data Speculation Compete with OoO Execution?



BARE CVA6

- Open Source
- Low power
- Simple
- Shallow pipeline
- In order

Outline

[0] Value Prediction Primer

[1] Microarchitecture

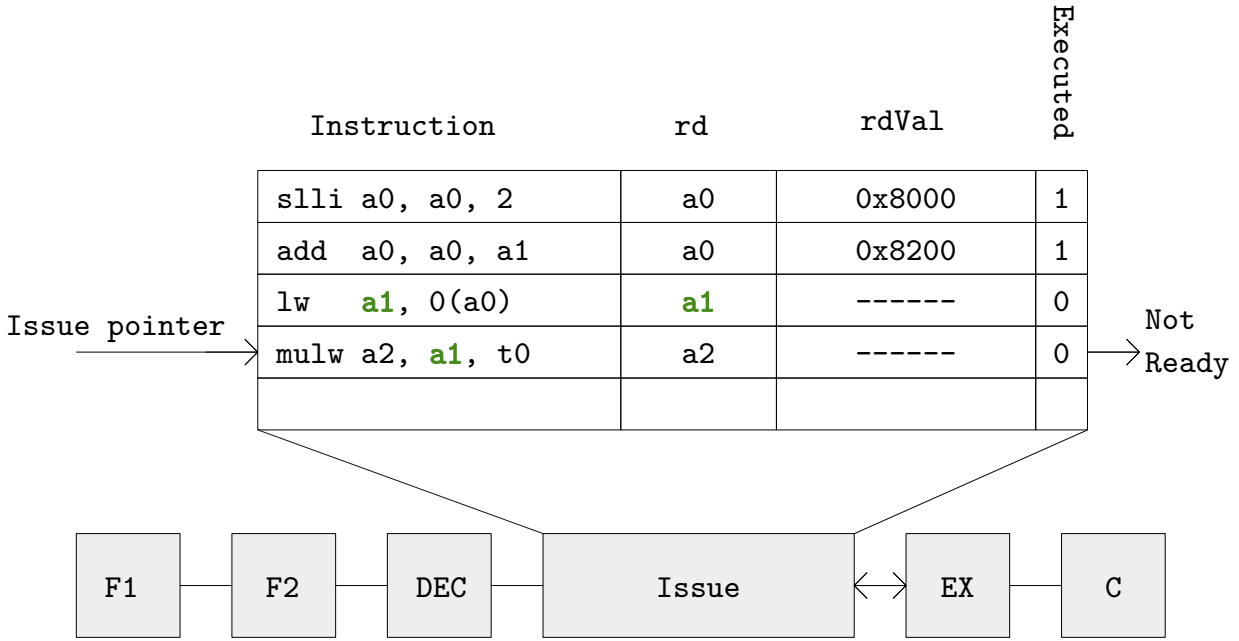
[2] Results

[3] Conclusion

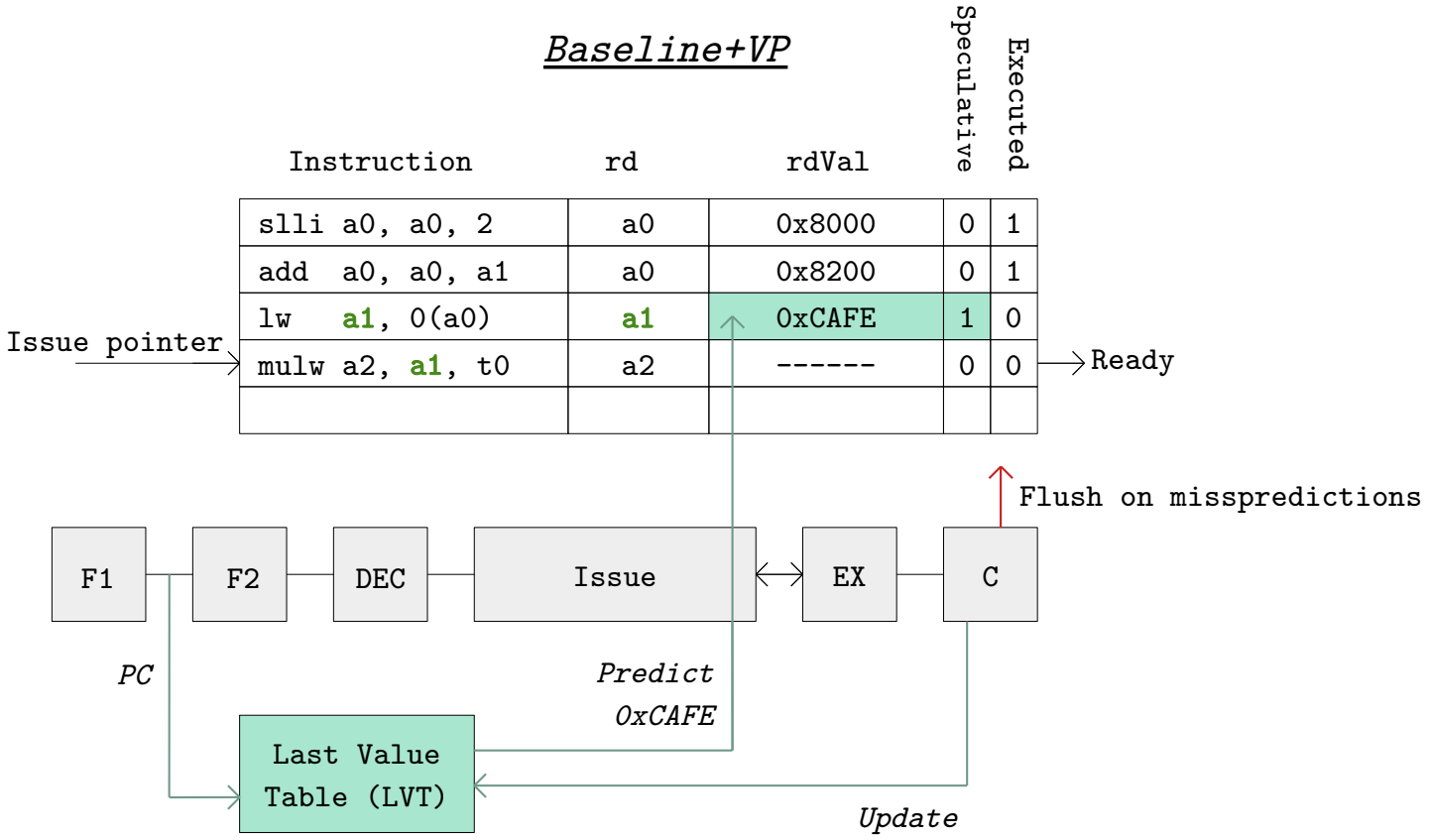
[0] Value Prediction Primer

Value Prediction Primer (1/2)

Baseline: Long latency loads stall the whole machine



Value Prediction Primer (2/2)

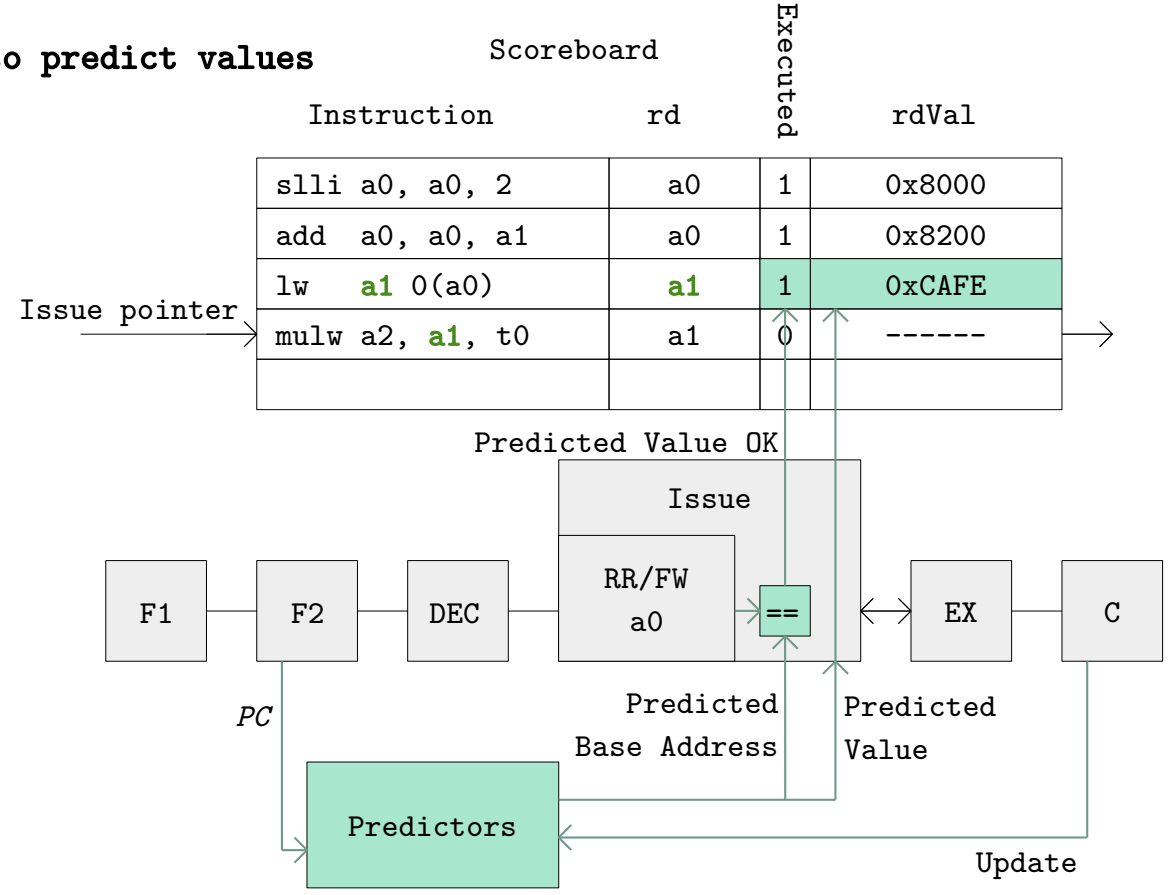


[1/3] MICROARCHITECTURE

NO MISSPREDICTION VALUE PREDICTION ?

Predict address to predict values

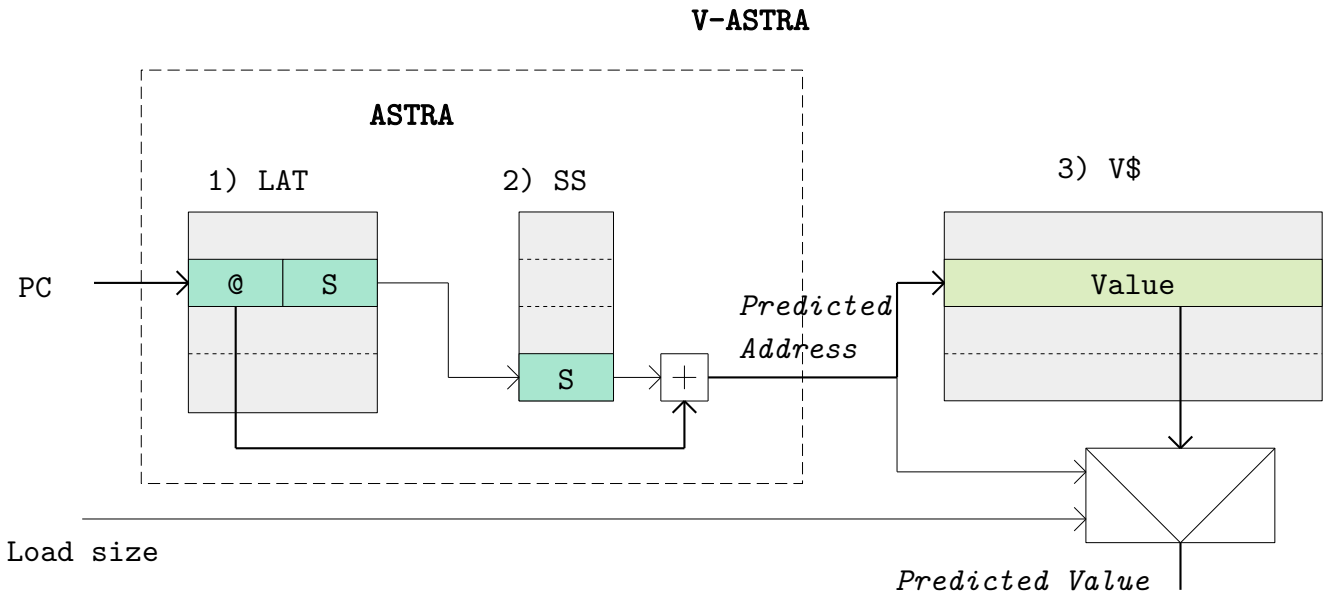
Based on DLVP[%]



% [Sheikh+, Load Value Prediction via Path-based Address Prediction, MICRO'17]

V-ASTRA : Value-Address STRide Aliasing

A predictor that maximizes coverage rather than accuracy

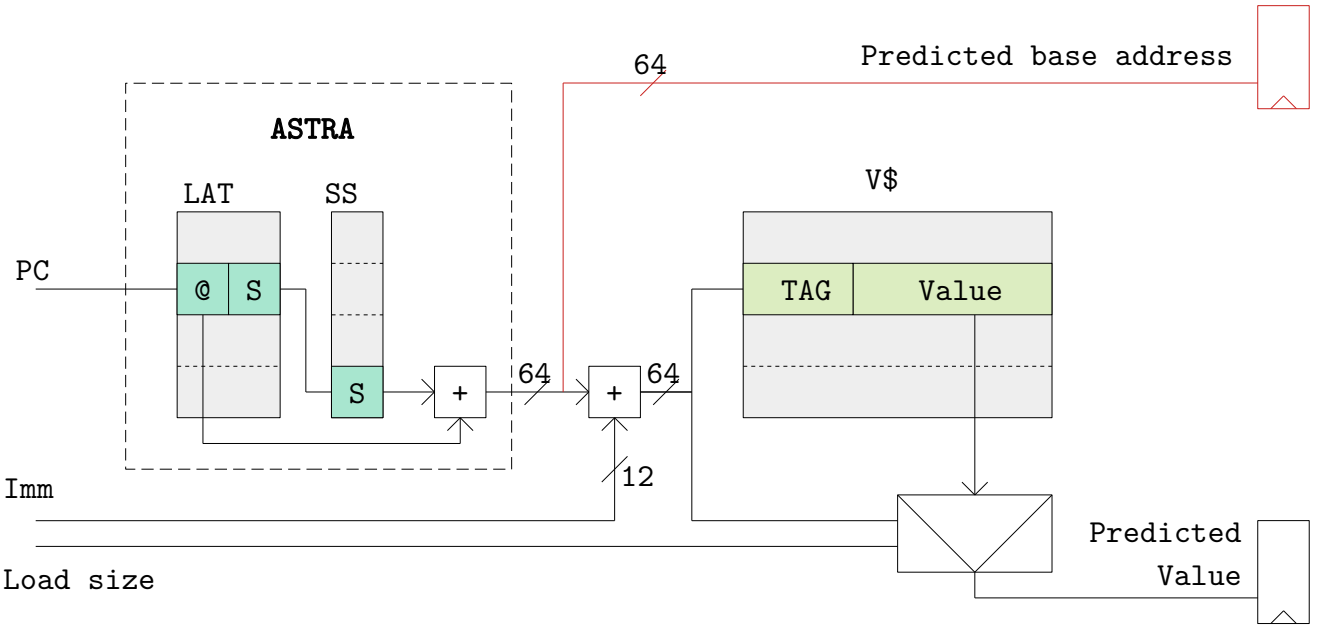


LAT: Last Address Table
SS: Shared Stride Table
V$: Value Cache

ENHANCED V-ASTRA : (1/2) [+No Misprediction]

Predict base address for fast check at issue before prediction is used
Effective address is needed to index V$
* In RISC-V AGEN is REG + IMM
* Easy to do in the predictor

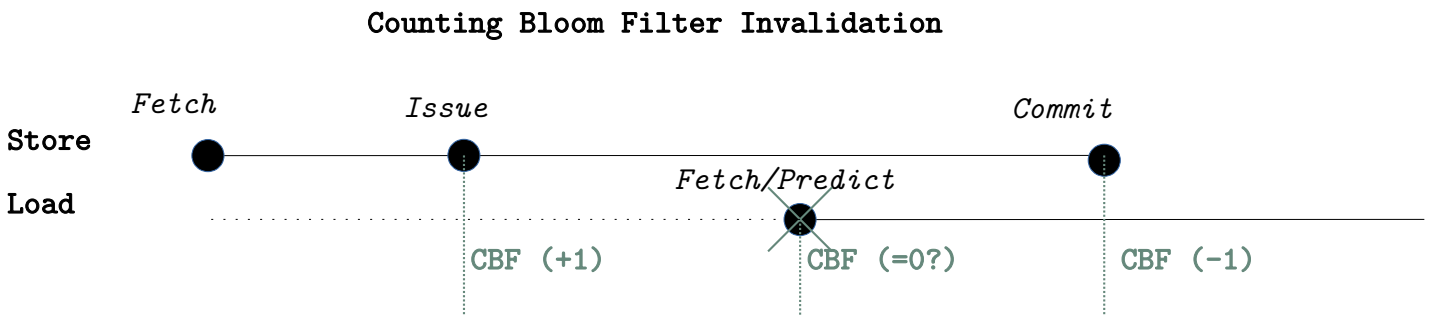
V$ is coherent



ENHANCED V-ASTRA : (2/2) [+No Misspredication] [+Efficient Hazard Handling]

How to handle in-flight stores ?

* Counting Bloom Filter

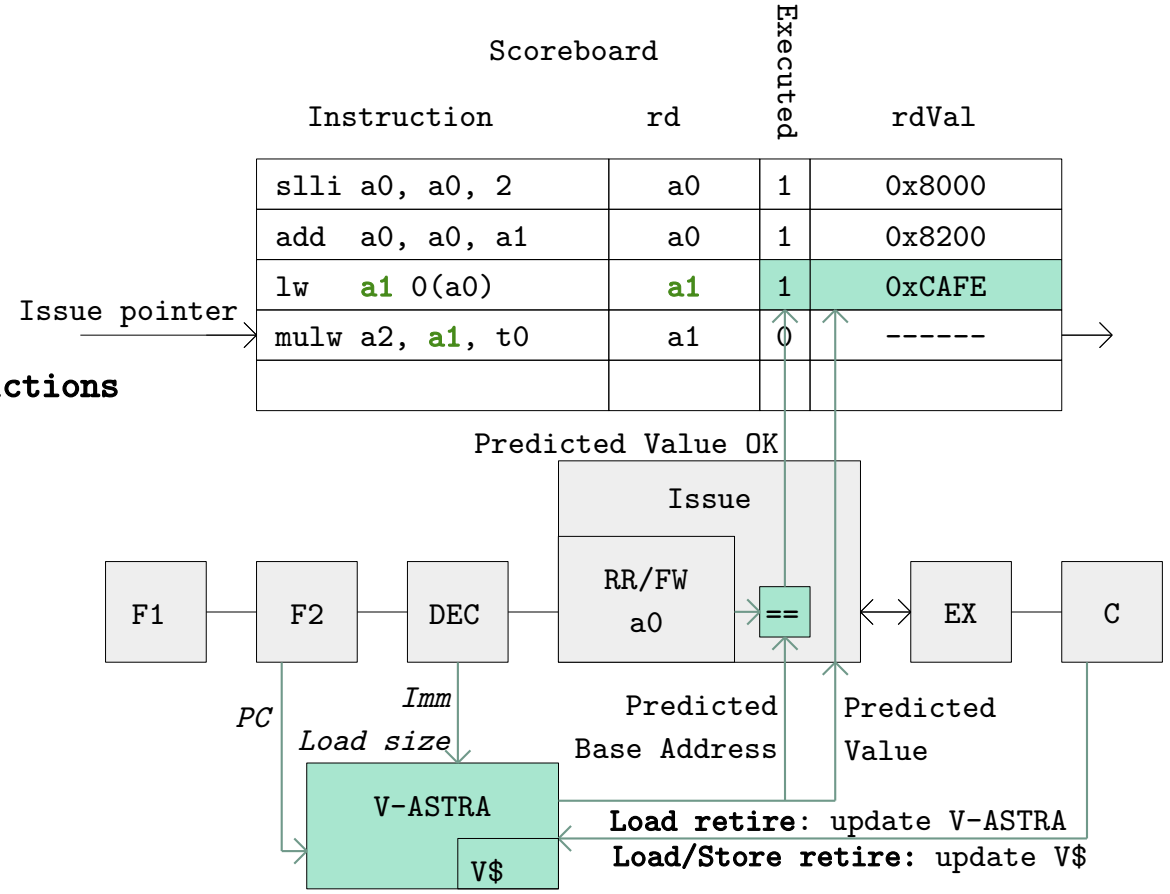


ENHANCED V-ASTRA INTEGRATION

No misprediction penalty
(It's cool)

0 cycles load to use
(It's super cool)

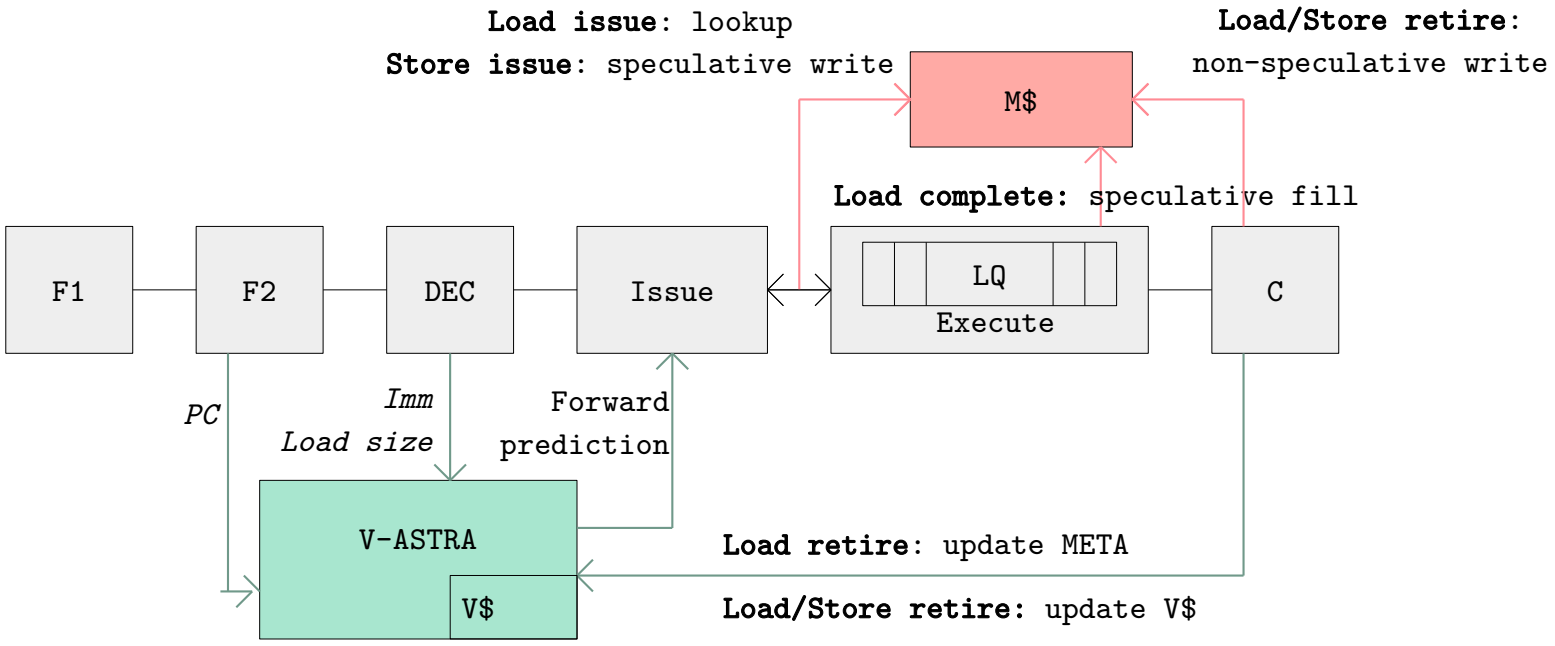
No L1 access on correct predictions



STORES DISRUPT VP: A PIPELINE WITH V-ASTRA AND M$

M$: Minicache

* A speculatively written LOD$ inserted in parallel of the LSU
* Lightweight store to load forwarding



0) V-ASTRA hit: 0 cycle load to use
1) M$ hit: 1 cycle load to use
2) Otherwise: LSU+Memory latency load to use

META PREDICTOR (V-ASTRA + LVP) AND REPLAY

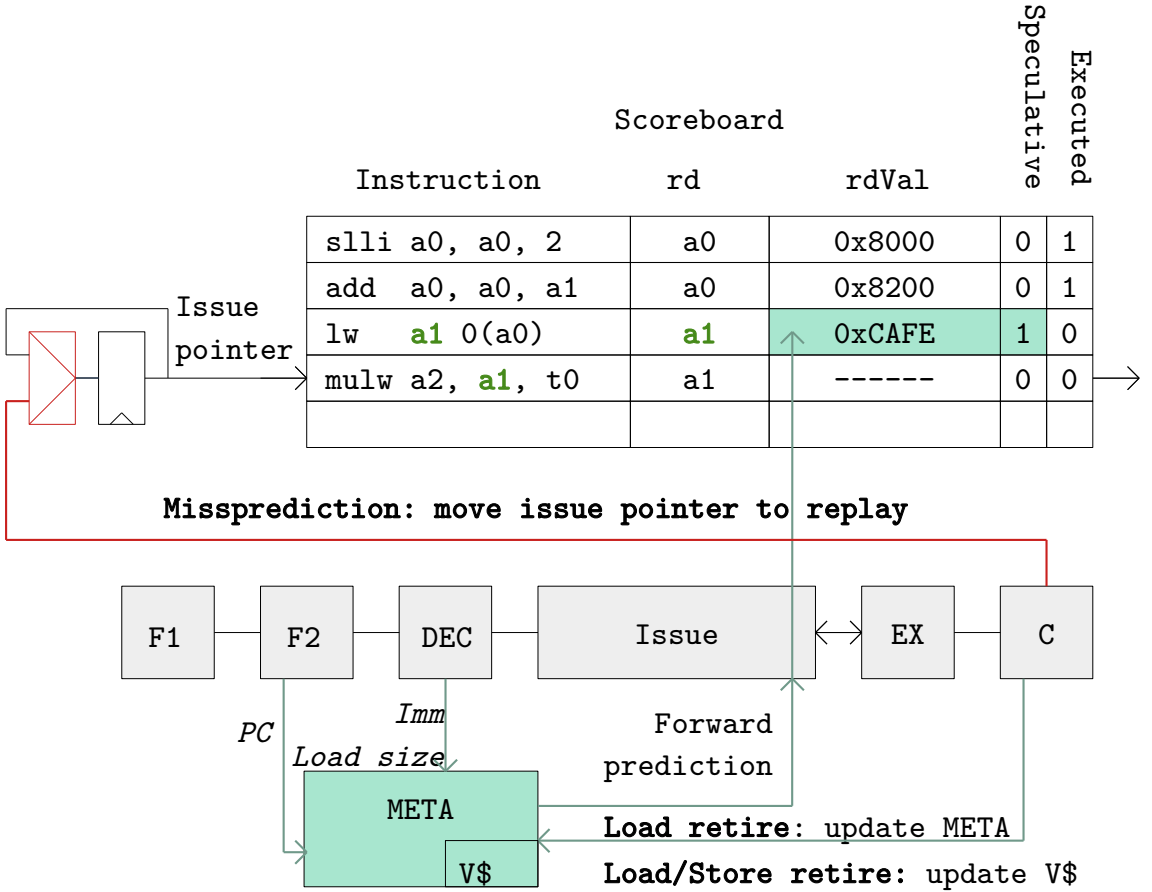
META = V-ASTRA + LVP

Orthogonal prediction scheme[1]

* More valid predictions

Pure VP has mispredictions

* Replay is simple
* And has minimal latency cost



1. [Sheikh+, Efficient Load Value Prediction using Multiple Predictors and Filters, HPCA'19]

[2/3] RESULTS

EXPERIMENTAL SETUP

Custom Gem5 model of the Cva6 based on the open RTL code

* *Few enhancements*
  - 4 degree superscalar
  - State-of-art 32KB TAGE Branch Predictor
  - Scoreboard of size 32, no WaW dependancy
* Standard Memory hierarchy
  - L1I: 32kB, 4-way, 1-cycle access latency, 16 MSHR,
  - L1D: 64kB, 4-way, 4-cycle load-to-use, 16 MSHR,
      + Stride prefetcher 64 entries, 4 degree
  - L2: 256kB, 8-way, 24-cycle load-to-use, 20 MSHR,
      + AMPM prefetcher 256 entries
  - 64-Bytes cacheline for all caches
  - single DDR4-2400, 64-cycle load-to-use on average.

Value Predictor and Minicache

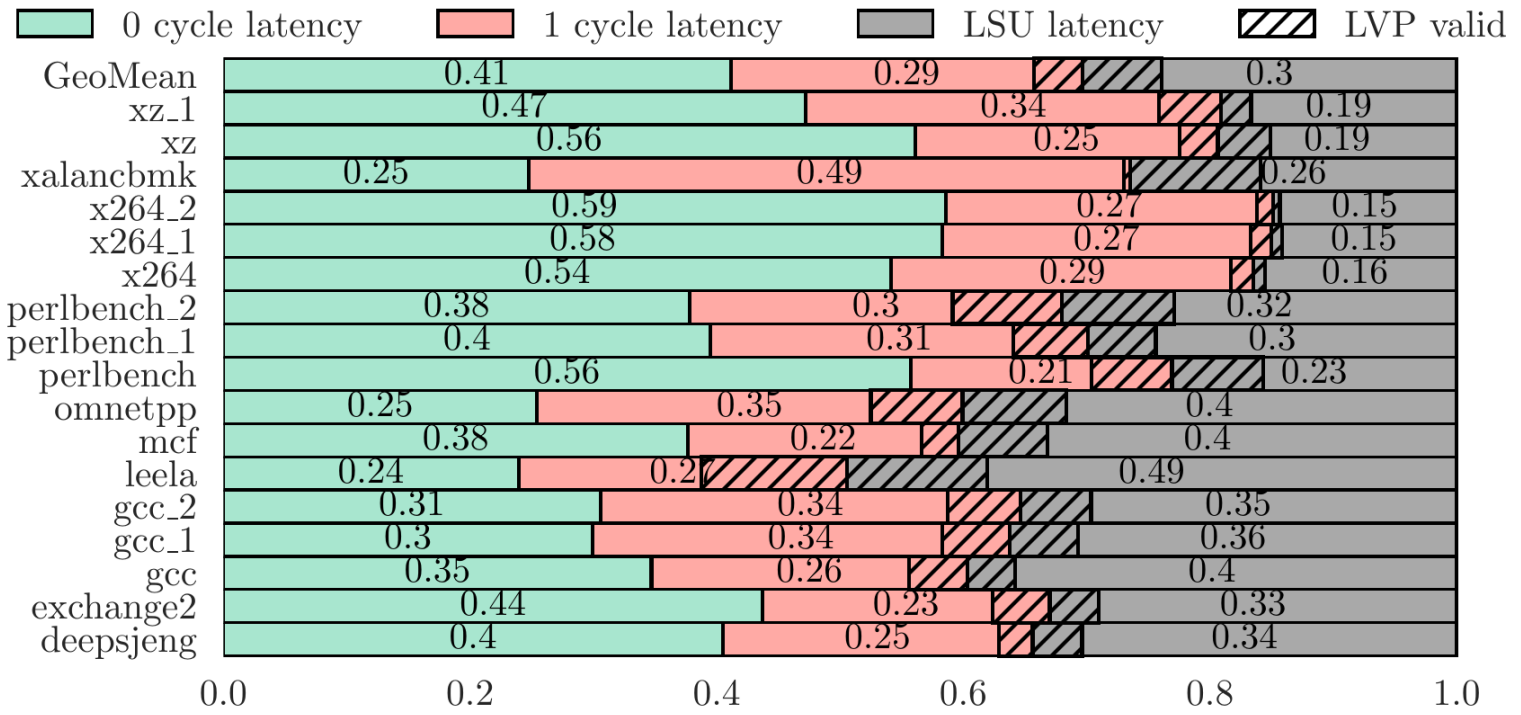
* META: 1024 entries, direct-mapped, Total budget 19.40kB.
* Minicache: 2 times 32 entries of 2 × 16 Bytes, direct-mapped, Total budget 1.35kB.

SPEC CPU 2017 benchmark

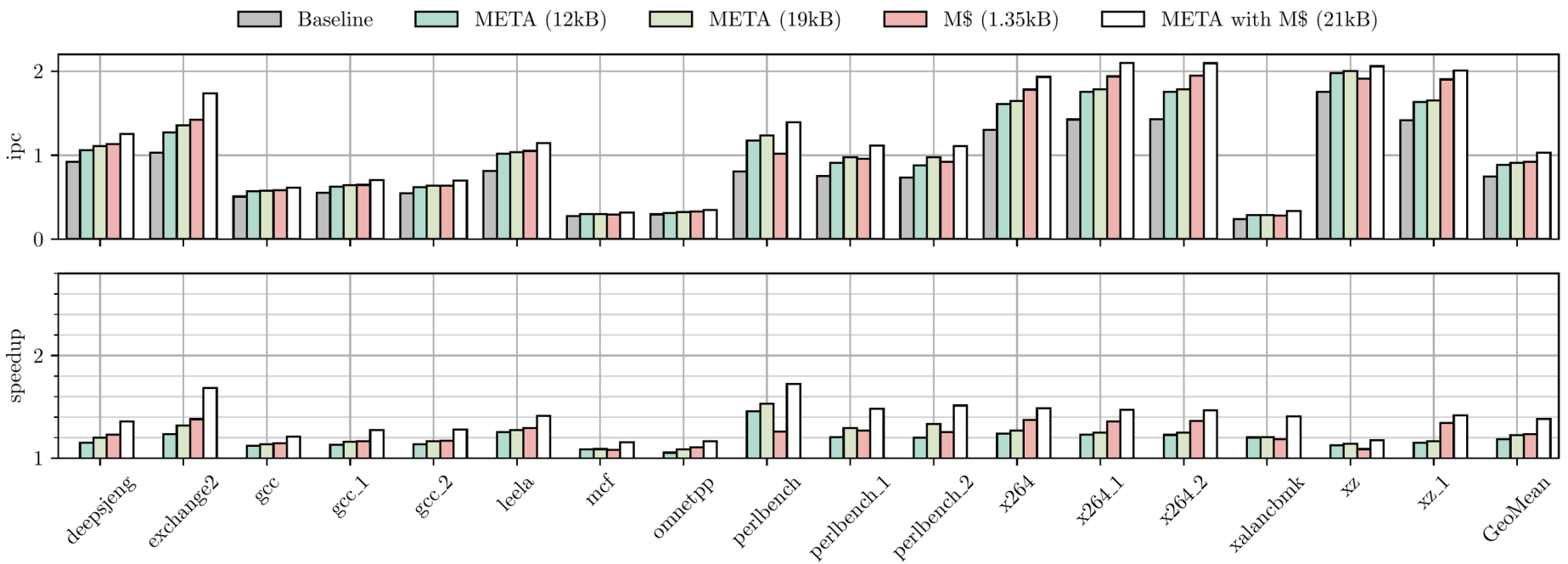
* Integer benchmarks
* 100M instructions Simpoints methodology

A M$ STRONGLY COUPLED TO V-ASTRA+LVP

V-ASTRA, LVP and M$ are mainly orthogonal



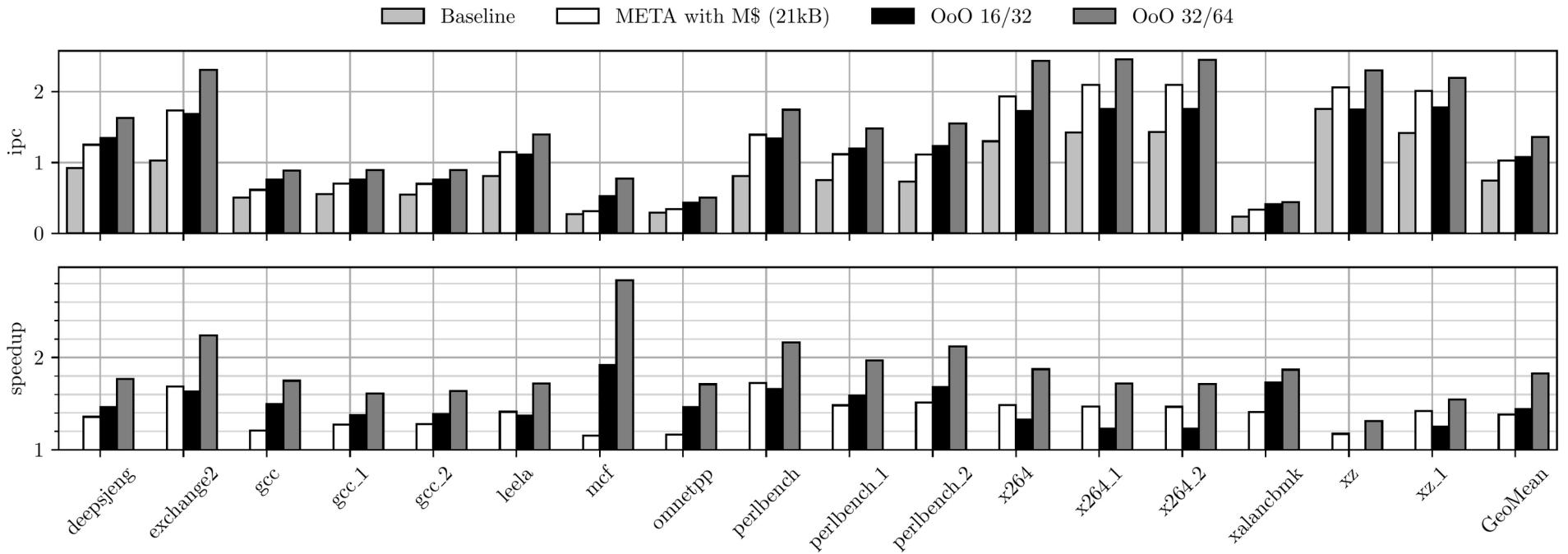
SPEEDUP (1/2)



META with M$: The acceleration provided by VP (~20%) and M$ (~20%) adds up to ~40% as they targets different kind of loads.

SPEEDUP (2/2)

The combination of M$ and META approaches the smaller out-of-order processor (86.7% of the gains) but lags behind the larger one (46.3% of the gains)



OoO 16/32 : 16 entries IQ / 32 entries ROB

OoO 32/64 : 32 entries IQ / 64 entries ROB

CONCLUSION : CAN InO EXECUTION WITH DATA SPECULATION COMPETE WITH OoO EXECUTION ?

Key Observation: In Order CPU suffers from load latencies

Key Idea: Value Prediction with low penalty and high coverage

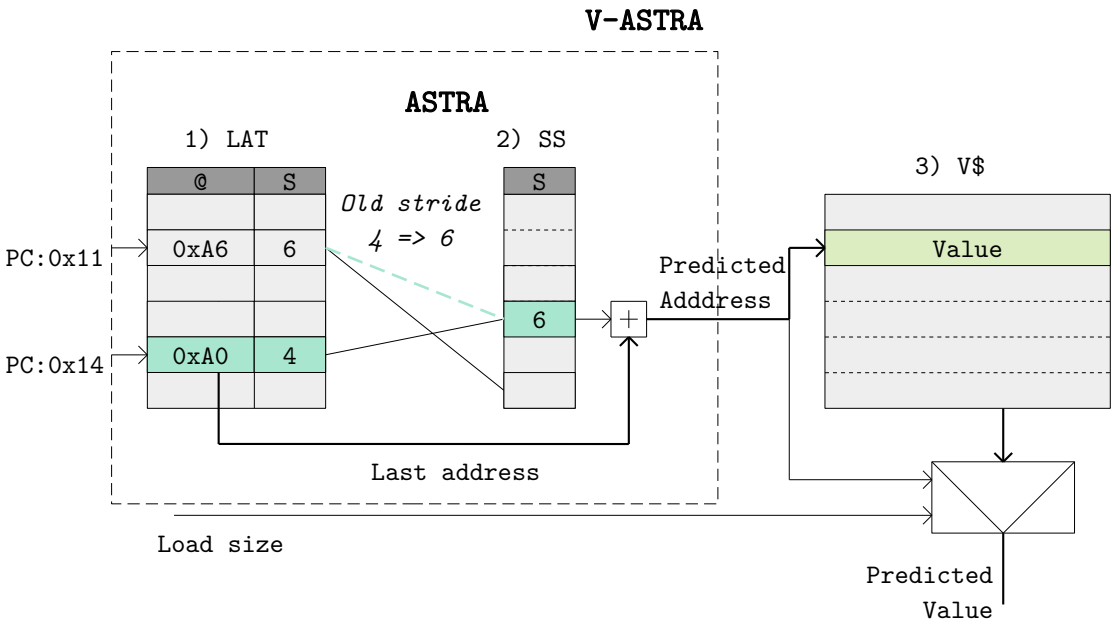
Key Results:

* 0-cycle latency loads and zero-cost misprediction
* Achieved by validating address predictions during issue for in-order micro-architectures
* Aggressive prediction scheme that maximizes coverage
* 38.4% geomean speedup, 1.03 vs. 0.76 IPC

[BACKUP SLIDES]

V-ASTRA : Value-Address STRide Alias (2/2)

> Motivation: maximize coverage with aliasing



LAT: Last Address Table
SS: Shared Stride Table
V$: Value Cache

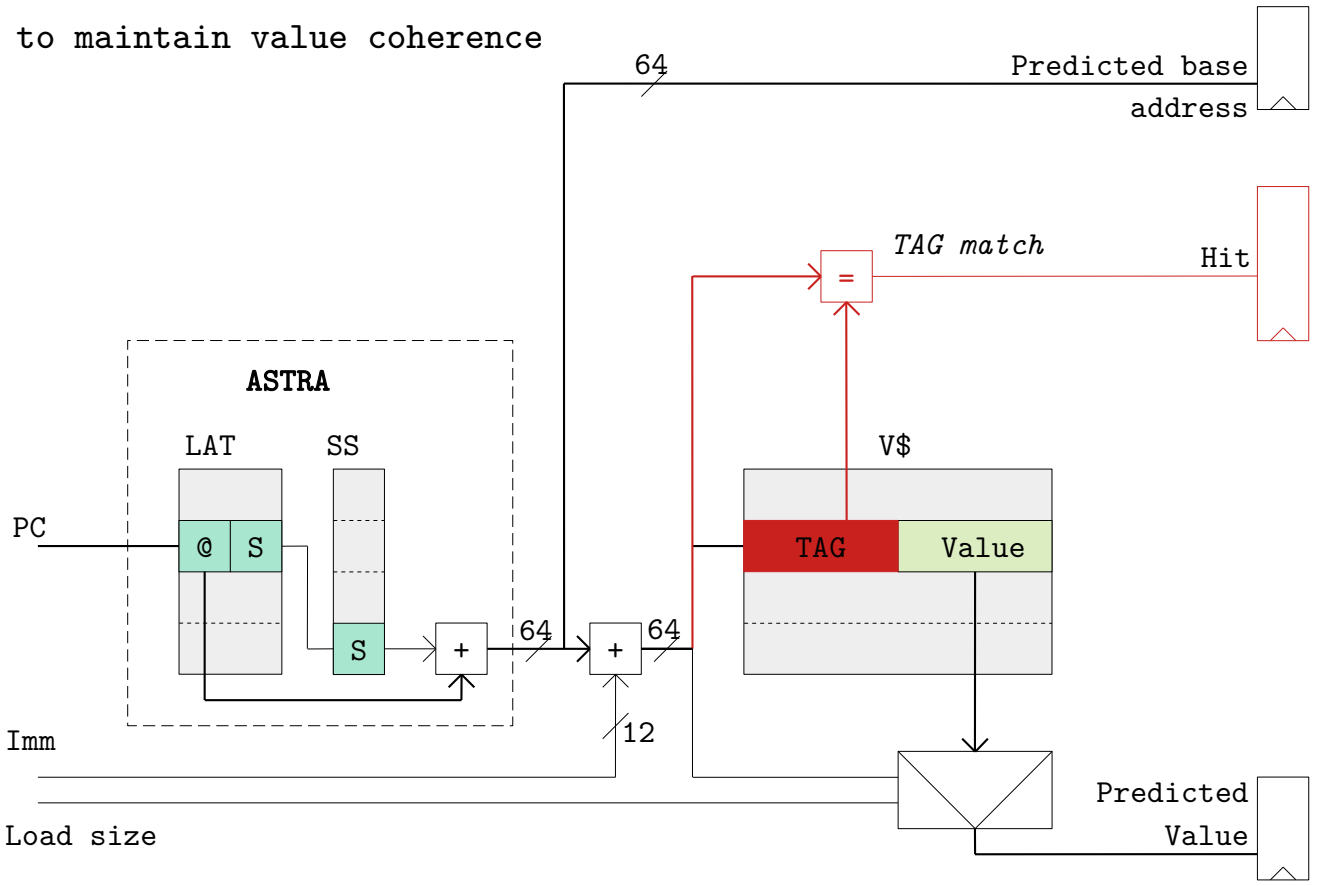
```
int satd_8x4(u8 *p1, u8 *p2) {
    v0 = (p1[0] - p2[0]) + ((p1[4] - p2[4]));
    v1 = (p1[1] - p2[1]) + ((p1[5] - p2[5]));
    v2 = (p1[2] - p2[2]) + ((p1[6] - p2[6]));
    v3 = (p1[3] - p2[3]) + ((p1[7] - p2[7]));
}
```

Code excerpt of the sum of 4x4 absolute Hadamard transformed differences taken from x264

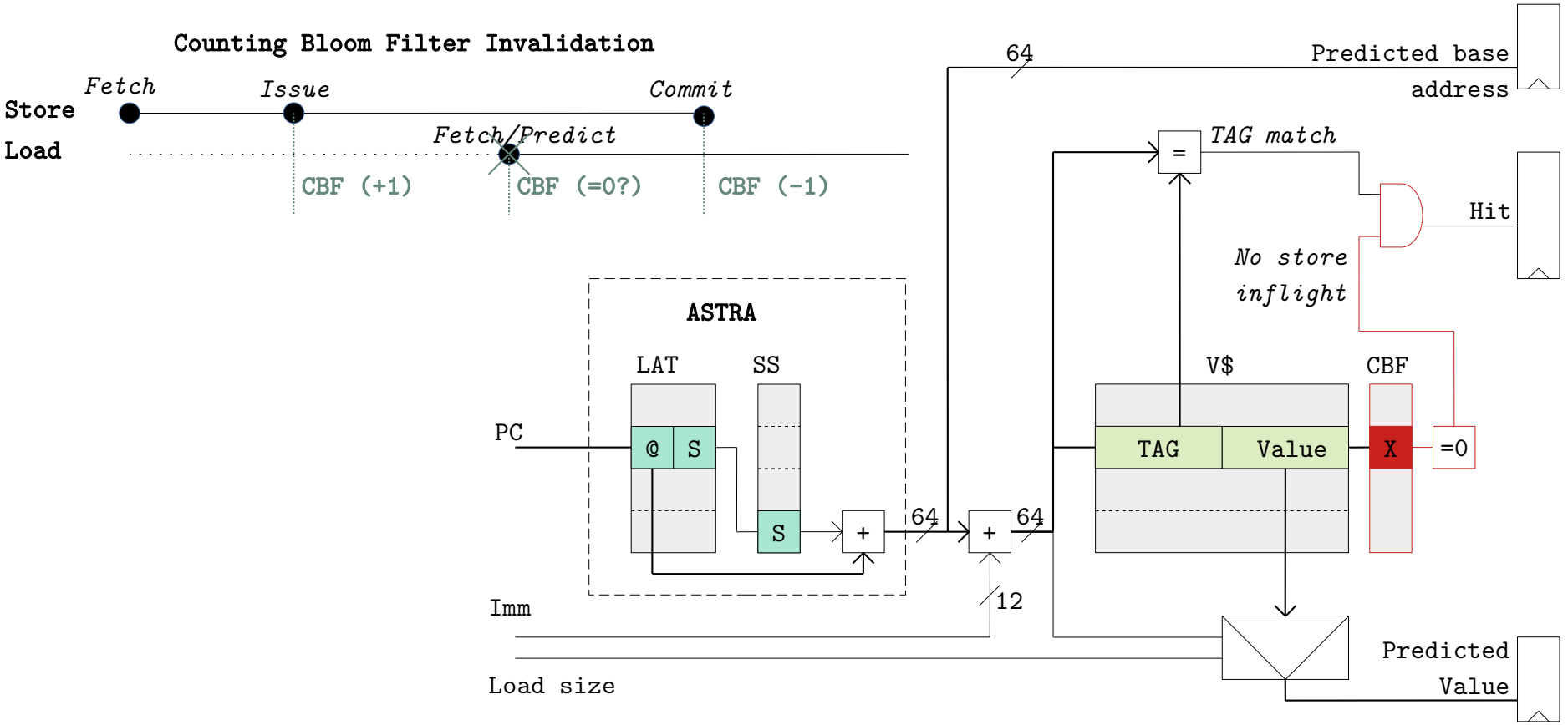
```
.globl satd_8x4_unscheduled
satd_8x4_unscheduled:
0x10: lbu t0, 0(a0)  # p1[0]
0x11: lbu t1, 0(a1)  # p2[0]
0x12: sub x4, t0, t1 # v0 = (p1[0] - p2[0])
0x13: lbu t0, 1(a0)  # p1[1]
0x14: lbu t1, 1(a1)  # p2[1]
0x15: sub x5, t0, t1 # (p1[1] - p2[1])
0x16: add x4, x4, x5 # v0 = v0 + ((p1[4] - p2[4]))
...
```

ENHANCED V-ASTRA : (1/3) [+No Misprediction(2/2)]

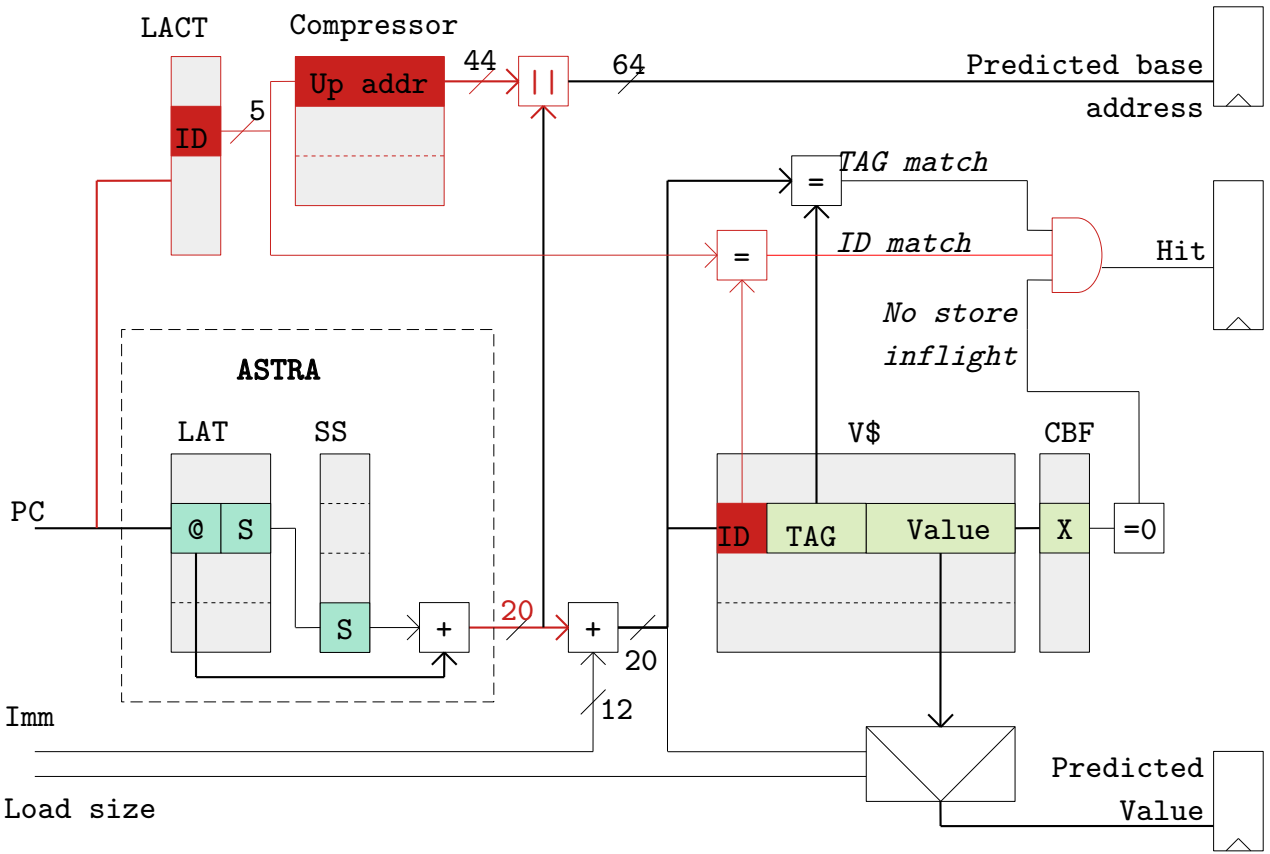
* Need a full tag to maintain value coherence



ENHANCED V-ASTRA : (2/3) [+No Misspredication] [+No Hazards]



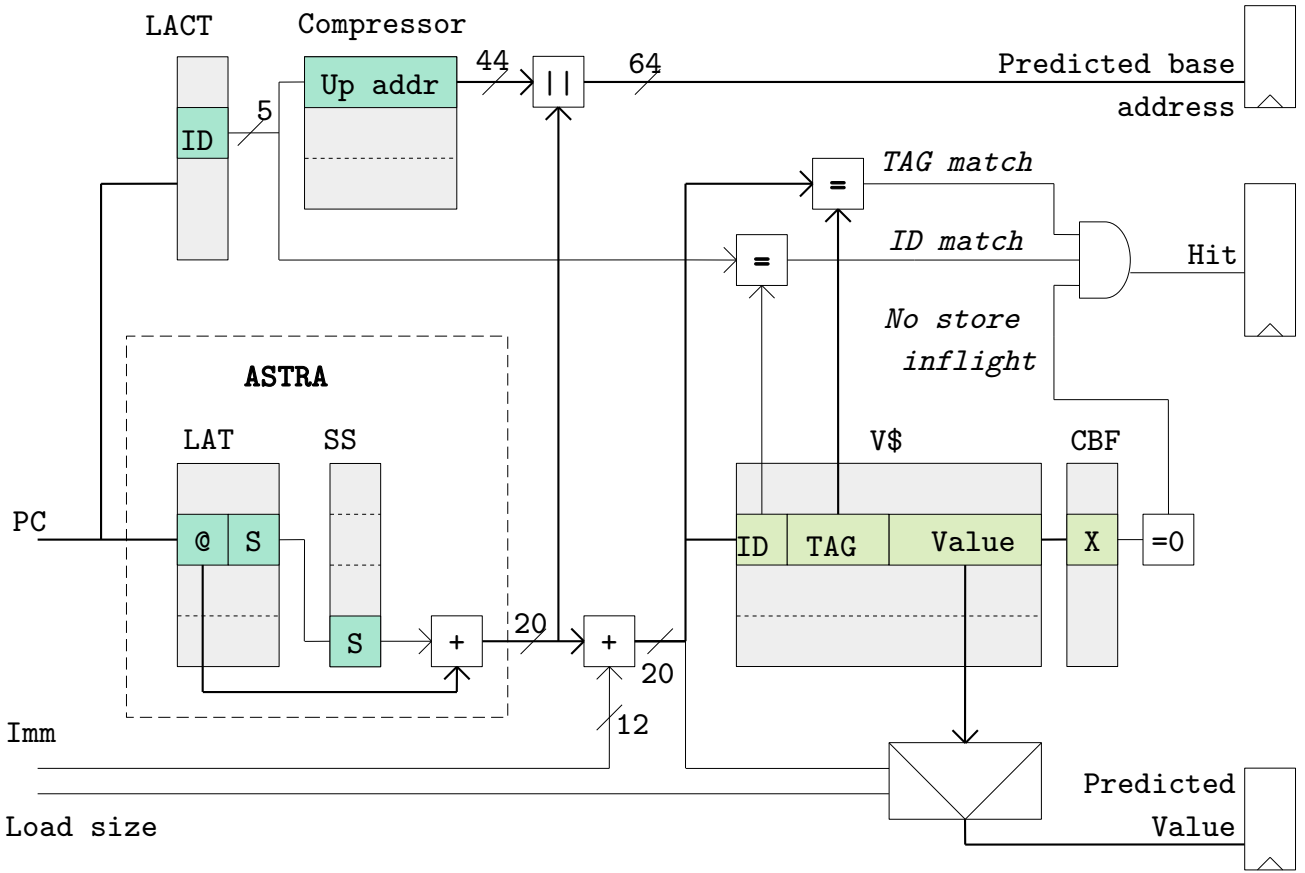
ENHANCED V-ASTRA : (3/3) [+No Misspredict ion] [+No Hazards] [+Compression]



LACT: Last Address Compressed Table

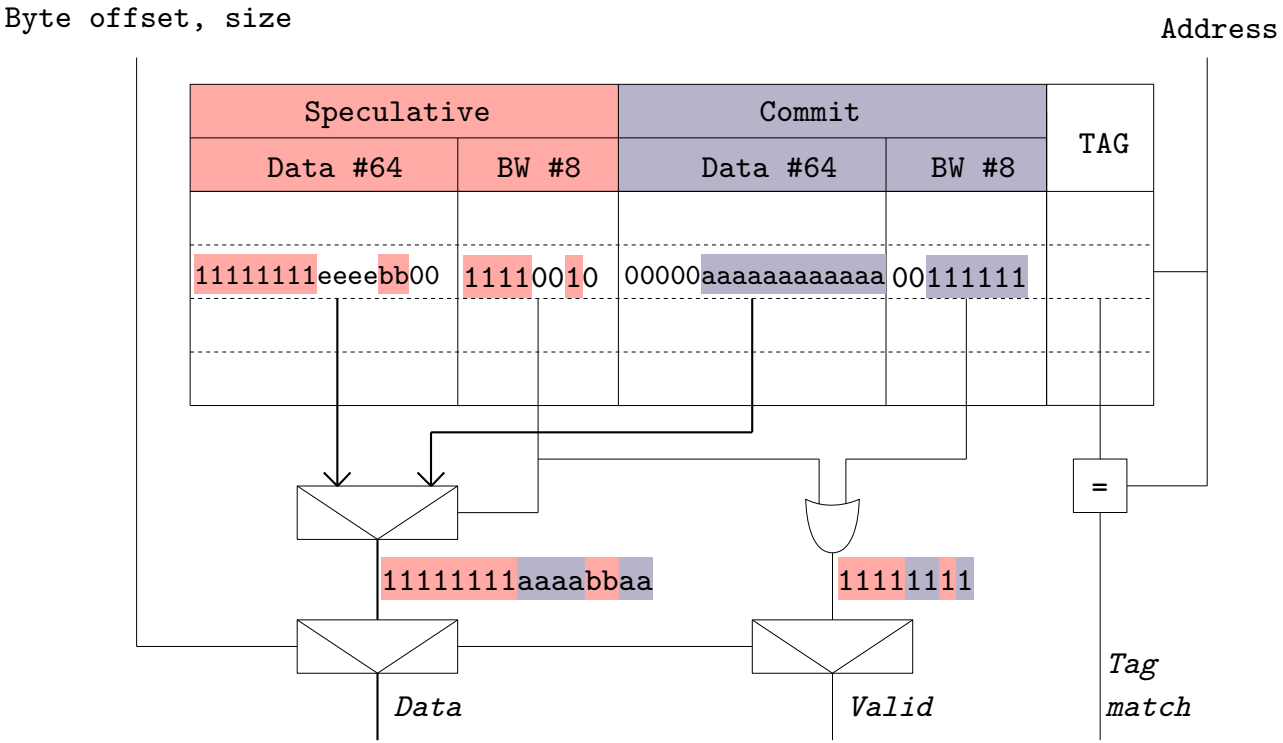
% [Seznec, Don't use the page number, but a pointer to it, ISCA'97]

ENHANCED V-ASTRA : (3/3) [+No Missprediction] [+No Hazards] [+Compression]



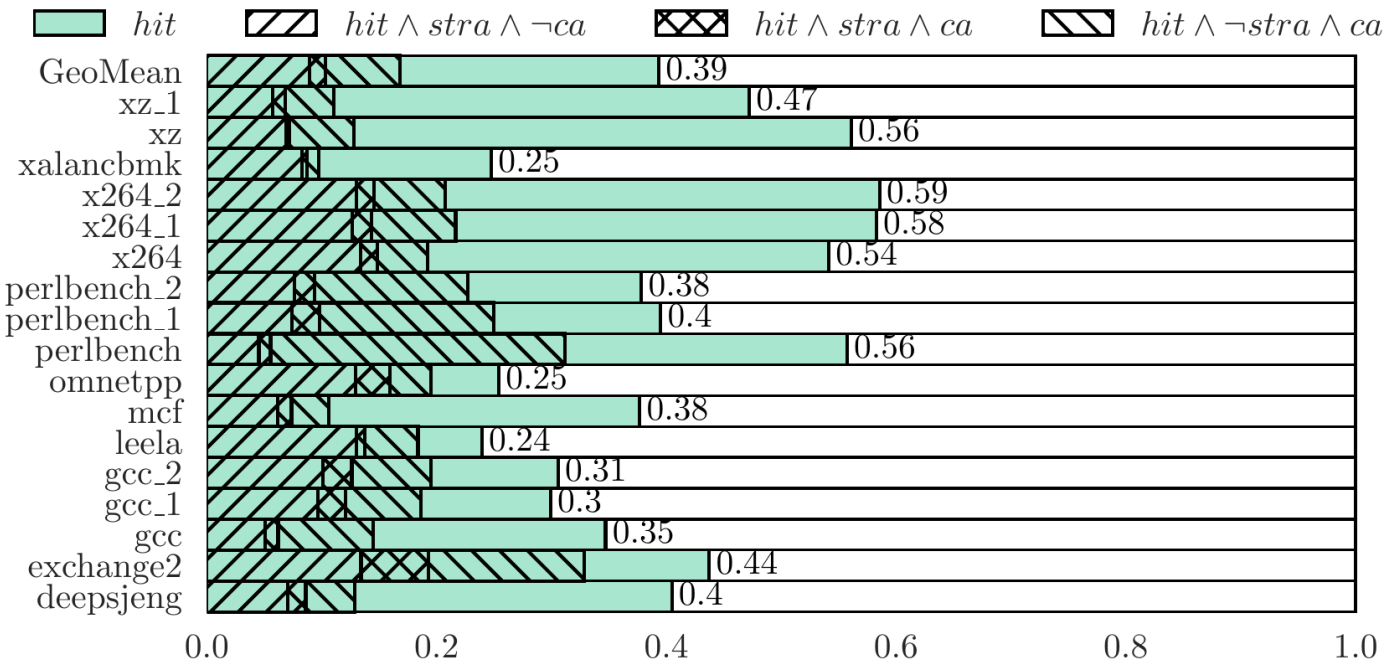
M$: MINICACHE

* Lightweight store to load forwarding
* Parallel to the LSU



V-ASTRA: SOURCES OF ALIASING

A significant part of hits are made by aliasing



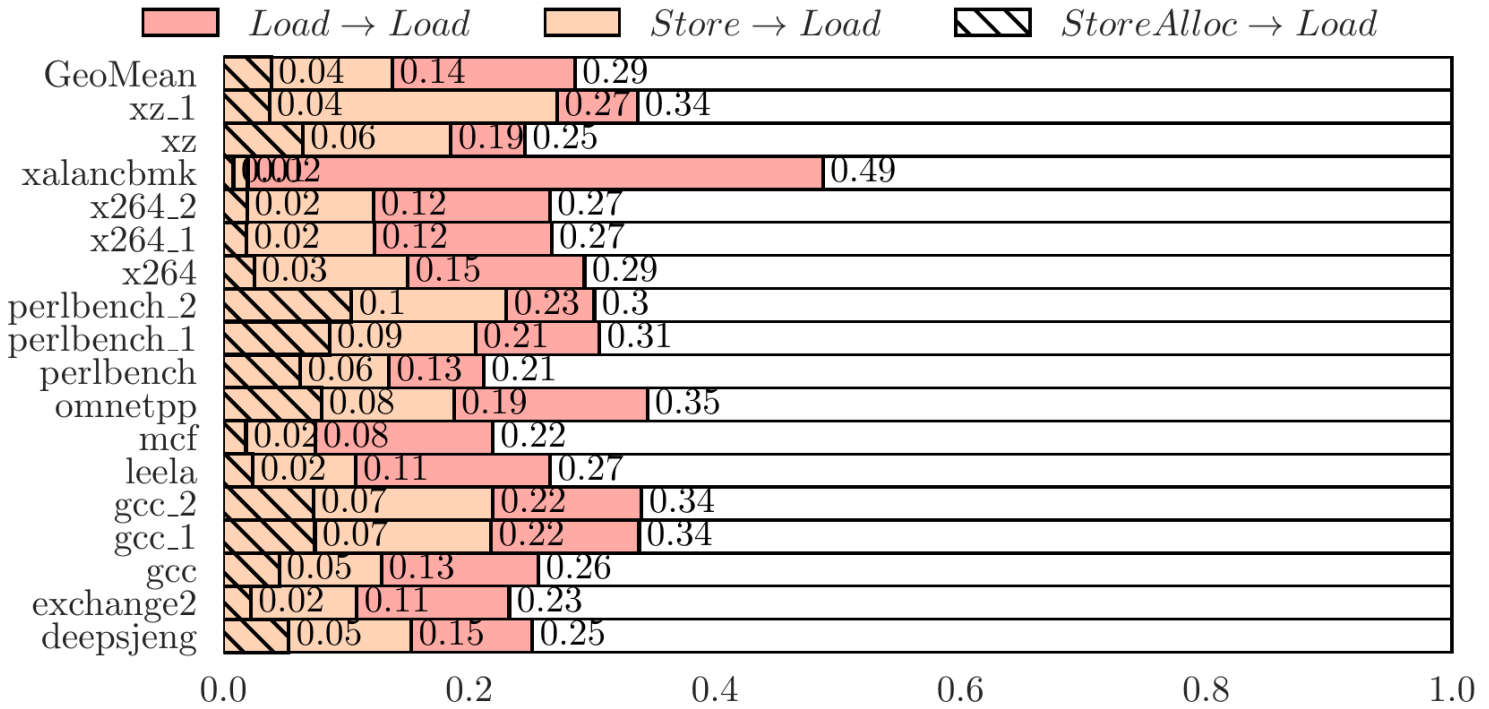
HIT : prediction is valid

STRA : STRides Alias in SS table

CA : Cache Alias in V$ table

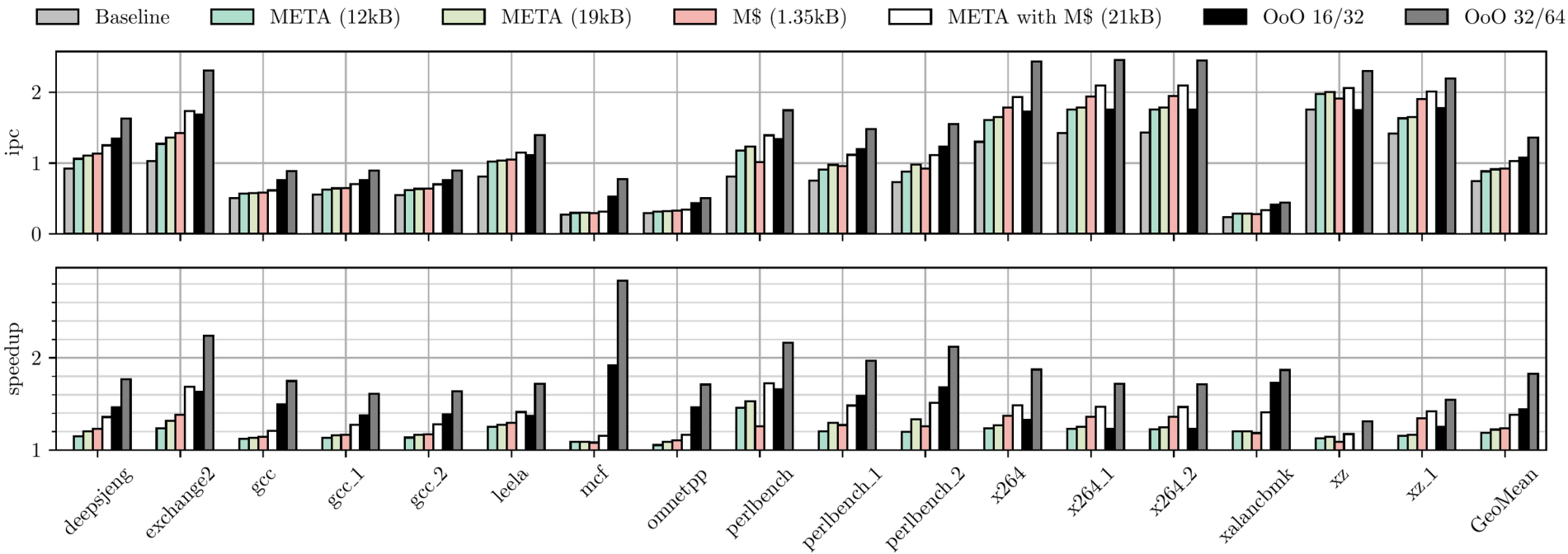
M$: PROPORTION OF HITS

Cost-Efficient Store to Load forwarding with direct map speculative M$



SPEEDUP (3/3)

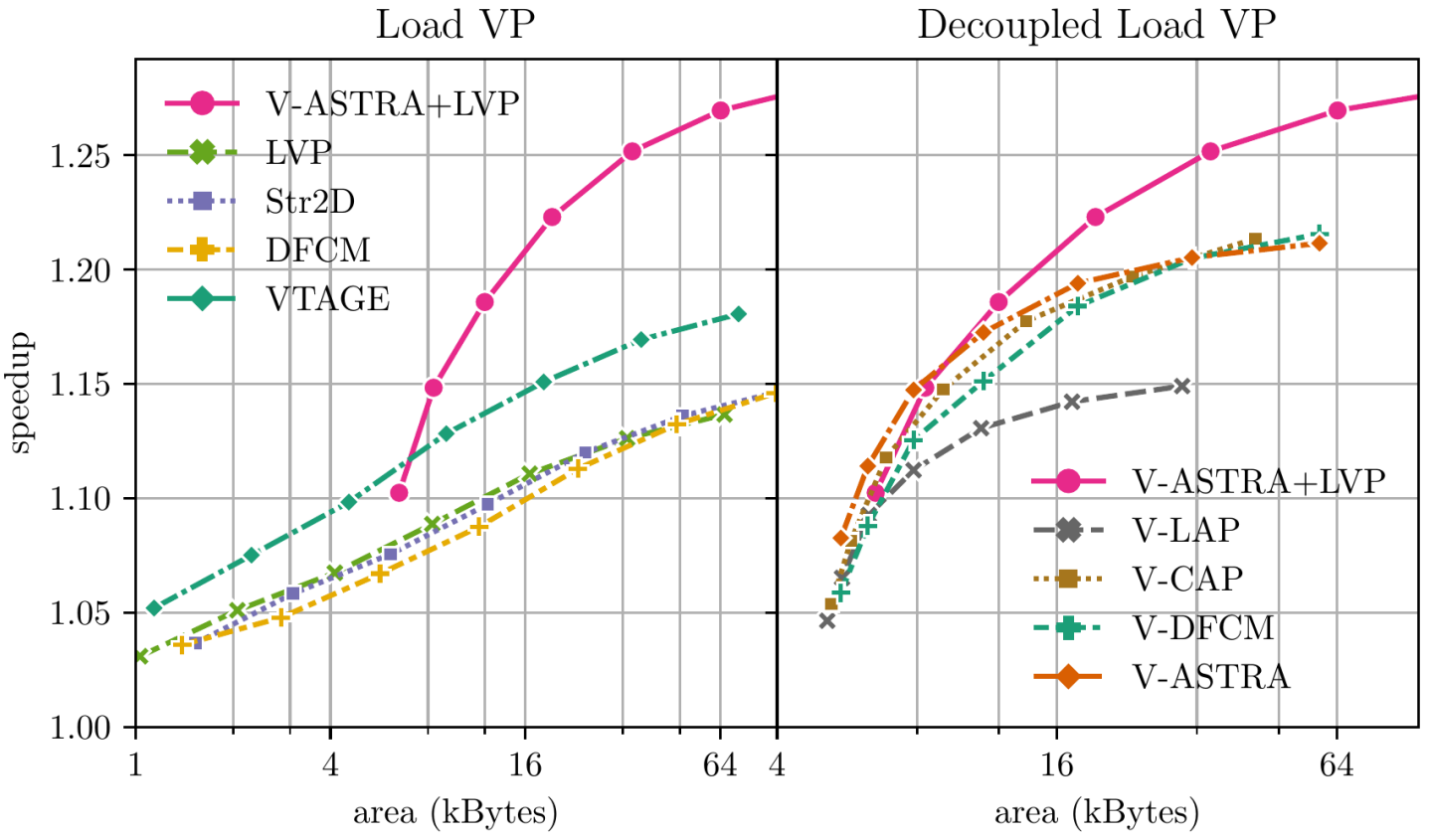
The combination of M$ and META approaches the smaller out-of-order processor (86.7% of the gains) but lags behind the larger one (46.3% of the gains)



OoO 16/32 : 16 entries IQ / 32 entries ROB

OoO 32/64 : 32 entries IQ / 64 entries ROB

PERF/AREA


Load VP
Decoupled Load VP
V-ASTRA+LVP
LVP
Str2D
DFCM
VTAGE
V-ASTRA+LVP
V-LAP
V-CAP
V-DFCM
V-ASTRA
speedup
1.25
1.20
1.15
1.10
1.05
1.00
1
4
16
64
4
16
64
area (kBytes)
area (kBytes)