Rebasing Microarchitectural Research with Industry Traces

J. Feliu[1], A. Perais[2], D. A. Jiménez[3], and A. Ros[4]

[1] Universitat Politècnica de València, Spain

[2] Université Grenoble Alpes, CNRS, Grenoble INP, TIMA, France

[3] Texas A&M University, USA

[4] University of Murcia, Spain

Introduction

The evaluation of an idea is only as good as the workloads used in the evaluation.

Introduction

The evaluation of an idea is only as good as the workloads used in the evaluation.

- *Good* is ambiguous:
  - Relevant (to users, industry, academia).
  - Representative (of different classes of algorithms or behaviors).

Introduction

The evaluation of an idea is only as good as the workloads used in the evaluation.

- *Good* is ambiguous:
  - Relevant (to users, industry, academia).
  - Representative (of different classes of algorithms or behaviors).

- Workloads can be simulated via emulation or by extracting traces to feed a simulator.
  - Information might be lost in translation → A good workload is incorrectly simulated.

Introduction

- The release of workloads by industry is a boon for the research community.
  - Provides relevant and representative workloads.
  - Crafting and bringing up good workloads is arduous.

Introduction

- The release of workloads by industry is a boon for the research community.
  - Provides relevant and representative workloads.
  - Crafting and bringing up good workloads is arduous.

- CVP-1 Aarch64 traces (generated at Qualcomm) are of large interest:
  - They are numerous (135 *public* and 2013 *secret*).
  - Wide range of workloads of interest: Compute INT/FP, cryptography, and datacenter.
  - They embed output register values.
  - They include system activity (typically not the case with Pin).

Introduction

- The release of workloads by industry is a boon for the research community.
  - Provides relevant and representative workloads.
  - Crafting and bringing up good workloads is arduous.

- CVP-1 Aarch64 traces (generated at Qualcomm) are of large interest:
  - They are numerous (135 *public* and 2013 *secret*).
  - Wide range of workloads of interest: Compute INT/FP, cryptography, and datacenter.
  - They embed output register values.
  - They include system activity (typically not the case with Pin).

- Interest in running CVP-1 traces in a microarchitectural simulator (e.g., ChampSim).

Introduction

- A CVP-1 to ChampSim converter is already available.
  - Emphasis on capturing the behavior of applications with very large instruction working sets.
    - Goal: studying instruction cache and branch predictor optimizations.
    - A subset of the traces used in the IPC-1 Championship.

Introduction

- A CVP-1 to ChampSim converter is already available.
  - Emphasis on capturing the behavior of applications with very large instruction working sets.
    - Goal: studying instruction cache and branch predictor optimizations.
    - A subset of the traces used in the IPC-1 Championship.

Competition traces		
Rank	Prefetcher	SpeedUp
1	EPI	1.2951
2	D-JOLT	1.2884
3	FNL+MMA	1.2861
4	BARÇA	1.2832
5	PIPS	1.2799
6	JIPS	1.2768
7	MANA	1.2658
8	TAP	1.2351

Introduction

- A CVP-1 to ChampSim converter is already available.
  - Emphasis on capturing the behavior of applications with very large instruction working sets.
    - Goal: studying instruction cache and branch predictor optimizations.
    - A subset of the traces used in the IPC-1 Championship.
  - Less emphasis on the conversion of aspects unrelated to the front-end.

Introduction

- A CVP-1 to ChampSim converter is already available.
  - Emphasis on capturing the behavior of applications with very large instruction working sets.
    - Goal: studying instruction cache and branch predictor optimizations.
    - A subset of the traces used in the IPC-1 Championship.
  - Less emphasis on the conversion of aspects unrelated to the front-end.
- CVP-1 traces come with some limitations (most can be patched).
  - No addressing mode for memory instructions.
  - No special-purpose registers.

Introduction

- A CVP-1 to ChampSim converter is already available.
  - Emphasis on capturing the behavior of applications with very large instruction working sets.
    - Goal: studying instruction cache and branch predictor optimizations.
    - A subset of the traces used in the IPC-1 Championship.
  - Less emphasis on the conversion of aspects unrelated to the front-end.
- CVP-1 traces come with some limitations (most can be patched).
  - No addressing mode for memory instructions.
  - No special-purpose registers.
- Despite the limitations, the CVP-1 converted traces use has spread.
  - Could skew observations and lead to inaccurate conclusions.

Introduction

- A CVP-1 to ChampSim converter is already available.
  - Emphasis on capturing the behavior of applications with very large instruction working sets.
    - Goal: studying instruction cache and branch predictor optimizations.
    - A subset of the traces used in the IPC-1 Championship.
  - Less emphasis on the conversion of aspects unrelated to the front-end.
- CVP-1 traces come with some limitations (most can be patched).
  - No addressing mode for memory instructions.
  - No special-purpose registers.
- Despite the limitations, the CVP-1 converted traces use has spread.
  - Could skew observations and lead to inaccurate conclusions.

We revisit the CVP-1 traces and perform a thorough analysis, allowing a better conversion to the ChampSim format.

Outline

- Introduction
- Conversion of memory instructions
  - Improvement mem-regs
  - Improvement base-update
  - Improvement mem-footprint
- Conversion of branch instructions
- Experimental evaluation
- Conclusion

Conversion of memory instructions

Improvement mem-regs

CVP-1 trace format

(Relevant fields for memory instructions only and not to scale)

Conversion of memory instructions

Improvement mem-regs

```
LDP X1, X0, [X0]
```

Load pair → X1 = Mem[X0], X0 = Mem[X0+8]

Conversion of memory instructions

Improvement mem-regs

```
LDP X1, X0, [X0]
```

Load pair → X1 = Mem[X0], X0 = Mem[X0+8]

Conversion of memory instructions

Improvement mem-regs

```
LDP X1, X0, [X0]
```

Load pair → X1 = Mem[X0], X0 = Mem[X0+8]

ChampSim trace format
(Relevant fields for memory instructions only and not to scale)

Conversion of memory instructions

Improvement mem-regs

```
LDP X1, X0, [X0]
```

Load pair → X1 = Mem[X0], X0 = Mem[X0+8]

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

cvp2champsim

Memory dests.	Memory sources	Source regs.	Dest. regs.
--	addr	X0	X1

ChampSim trace format

Conversion of memory instructions

Improvement mem-regs

```
LDP X1, X0, [X0]
```

Load pair → X1 = Mem[X0], X0 = Mem[X0+8]

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

cvp2champsim

Memory dests.	Memory sources	Source regs.	Dest. regs.
--	addr	X0	X1

ChampSim trace format

Conversion of memory instructions

Improvement mem-regs

```
LDP X1, X0, [X0]
```

Load pair → X1 = Mem[X0], X0 = Mem[X0+8]

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

cvp2champsim

Memory dests.	Memory sources	Source regs.	Dest. regs.
--	addr	X0	X1

ChampSim trace format

Not all destination registers are conveyed in the conversion.

Conversion of memory instructions

Improvement mem-regs

`LDP X1, X0, [X0]`

Load pair → X1 = Mem[X0], X0 = Mem[X0+8]

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

cvp2champsim

Memory dests.	Memory sources	Source regs.	Dest. regs.
--	addr	X0	X1, X0

ChampSim trace format

Convey all destination registers in the conversion.

- Dependencies with younger instructions are preserved.

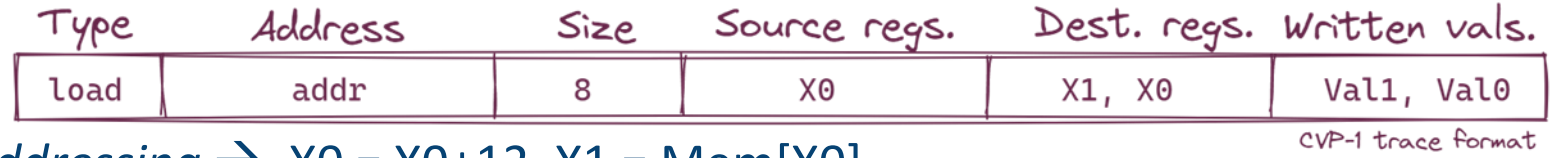
Outline

- Introduction
- Conversion of memory instructions
  - Improvement mem-regs
  - Improvement base-update
  - Improvement mem-footprint
- Conversion of branch instructions
- Experimental evaluation
- Conclusion

Conversion of memory instructions

Improvement base-update

```
LDR X1, [X0, #12]!
```



Load with pre-indexing increment addressing → X0 = X0+12, X1 = Mem[X0]

Conversion of memory instructions

Improvement base-update

```
LDR X1, [X0, #12]!
```

Load with pre-indexing increment addressing → X0 = X0+12, X1 = Mem[X0]

CVP-1 trace format

cvp2champsim

ChampSim trace format

Conversion of memory instructions

Improvement base-update

```
LDR X1, [X0, #12]!
```

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

Load with pre-indexing increment addressing → X0 = X0+12, X1 = Mem[X0]

cvp2champsim

Memory dests.	Memory sources	Source regs.	Dest. regs.
--	addr	X0	X1, X0

ChampSim trace format

No difference between:

- X0, generated by an ALU operation.
- X1, loaded from memory.

Conversion of memory instructions

Improvement base-update

```
LDR X1, [X0, #12]!
```

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

Load with pre-indexing increment addressing → X0 = X0+12, X1 = Mem[X0]

- Infer the addressing mode.
- Detect the base register.
- Differentiate between pre- and post-indexing increments:
  - Pre: ALU + MEM
  - Post: MEM + ALU

Conversion of memory instructions

Improvement base-update

`LDR X1, [X0, #12]!`

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

Load with pre-indexing increment addressing → X0 = X0+12, X1 = Mem[X0]

cvp2champsim

- Infer the addressing mode.
- Detect the base register.
- Differentiate between pre- and post-indexing increment:
  - Pre: ALU + MEM
  - Post: MEM + ALU

	Memory dests.	Memory sources	Source regs.	Dest. regs.
ALU:	--	--	X0	X0
MEM:	--	addr	X0	X1

ChampSim trace format

Outline

- Introduction
- Conversion of memory instructions
  - Improvement mem-regs
  - Improvement base-update
  - Improvement mem-footprint
- Conversion of branch instructions
- Experimental evaluation
- Conclusion

Conversion of memory instructions

Improvement mem-footprint

```
LDP X1, X0, [X0]
```

Load pair → X0 = Mem[X0], X1 = Mem[X0+8]

Conversion of memory instructions

Improvement mem-footprint

`LDP X1, X0, [X0]`

Load pair → X0 = Mem[X0], X1 = Mem[X0+8]

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

Addr	Cacheline (Addr)	Addr+8	Cacheline (Addr+8)
0x0ad70030	0x0ad70000	0x0ad70038	0x0ad70000

← Belong to the same cacheline

Converted instructions access a single cacheline.

Conversion of memory instructions

Improvement mem-footprint

```
LDP X1, X0, [X0]
```

Load pair → X0 = Mem[X0], X1 = Mem[X0+8]

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

Addr	Cacheline (Addr)	Addr+8	Cacheline (Addr+8)
0x0ad70030	0x0ad70000	0x0ad70038	0x0ad70000
0x0ad70038	0x0ad70000	0x0ad70040	0x0ad700010

← Cross a cacheline boundary

Converted instructions access a single cacheline.

Conversion of memory instructions

Improvement mem-footprint

`LDP X1, X0, [X0]`

Load pair → X0 = Mem[X0], X1 = Mem[X0+8]

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format

Addr	Cacheline (Addr)	Addr+8	Cacheline (Addr+8)
0x0ad70030	0x0ad70000	0x0ad70038	0x0ad70000
0x0ad70038	0x0ad70000	0x0ad70040	0x0ad700010

Cross a cacheline boundary

cvp2champsim

If an access crosses a cacheline boundary:
→ Add a second memory source.

Memory dests.	Memory sources	Source regs.	Dest. regs.
--	addr, addr+8	X0	X1, X0

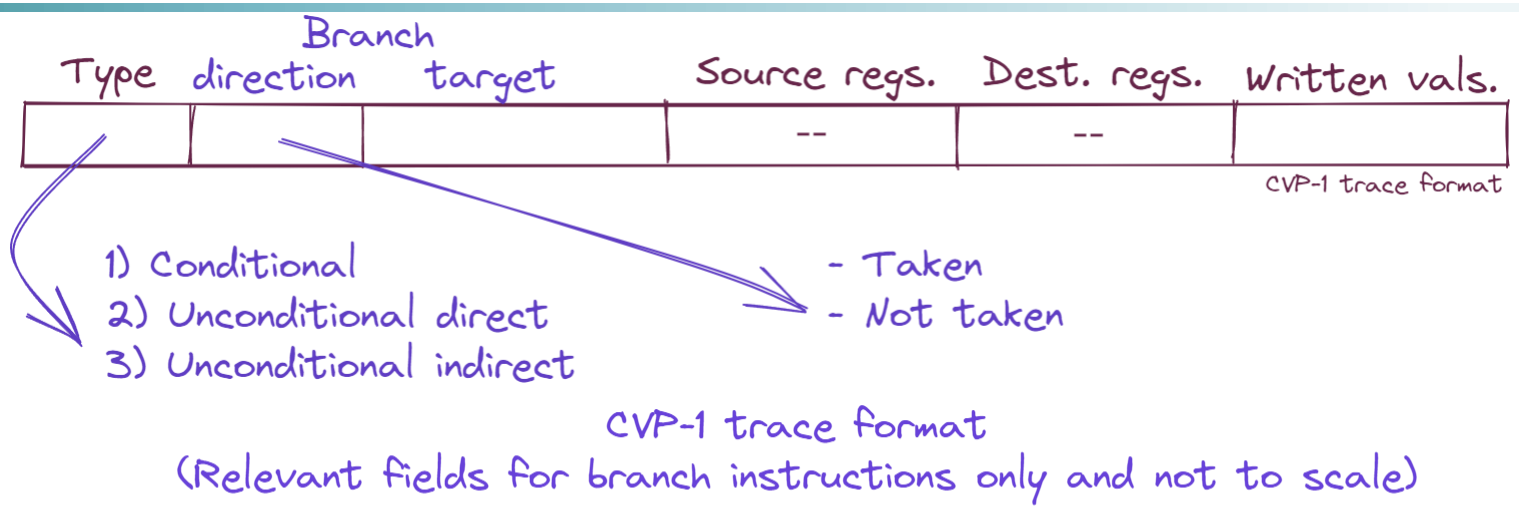
ChampSim trace format

Outline

- Introduction
- Conversion of memory instructions
- Conversion of branch instructions
  - Improvement call-stack
  - Improvement branch-regs
  - Improvement flag-reg
- Experimental evaluation
- Conclusion

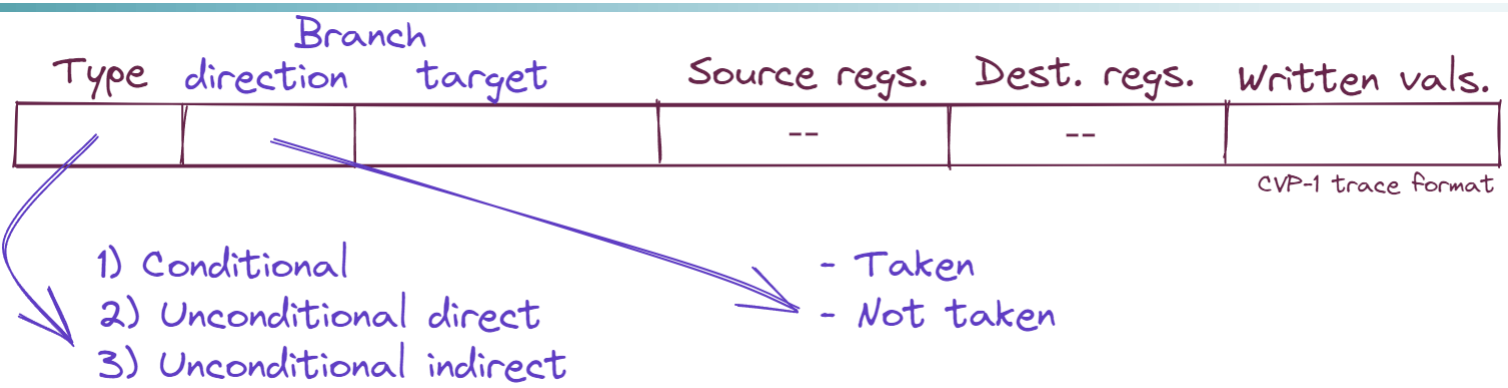
Conversion of branch instructions

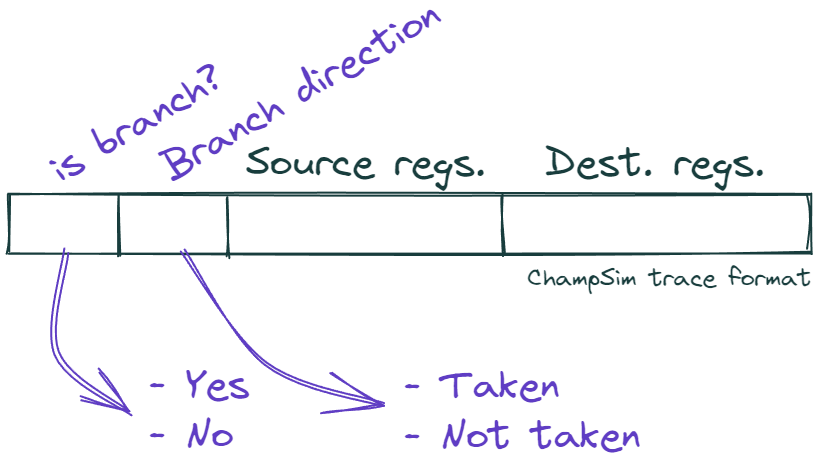
Improvement call-stack



Conversion of branch instructions

Improvement call-stack

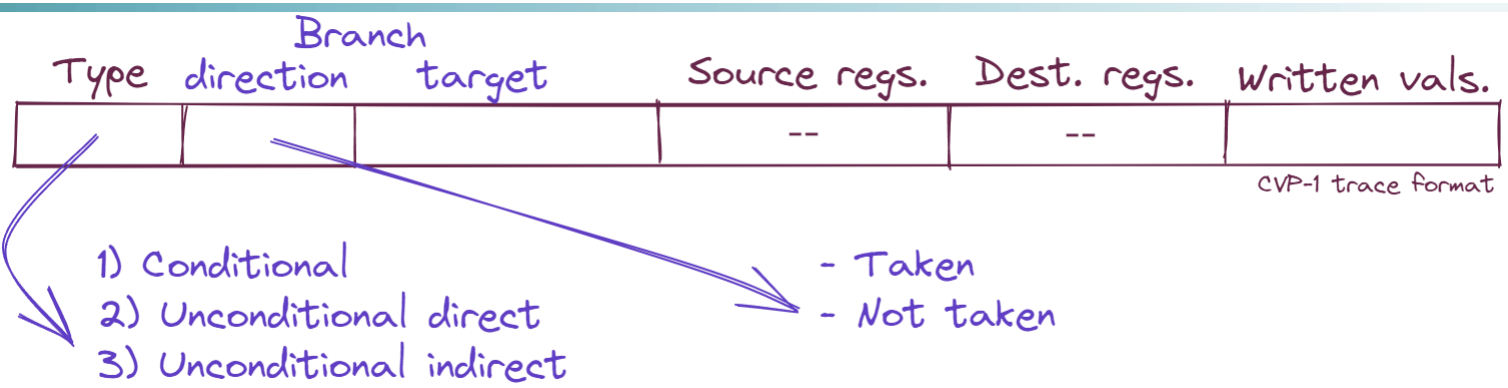


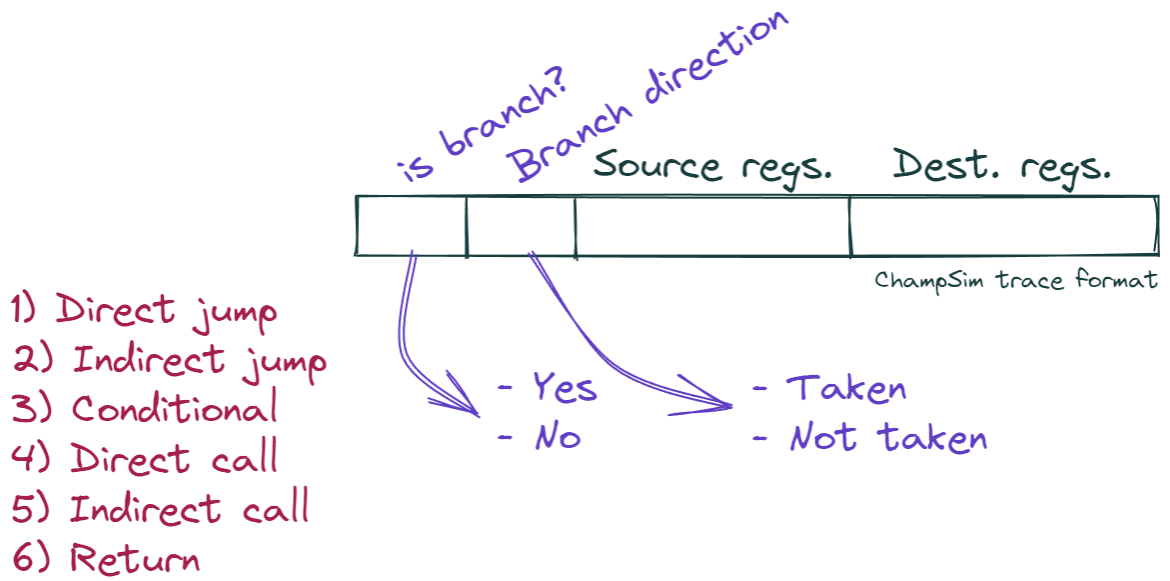


ChampSim trace format
(Relevant fields for branch instructions only and not to scale)

Conversion of branch instructions

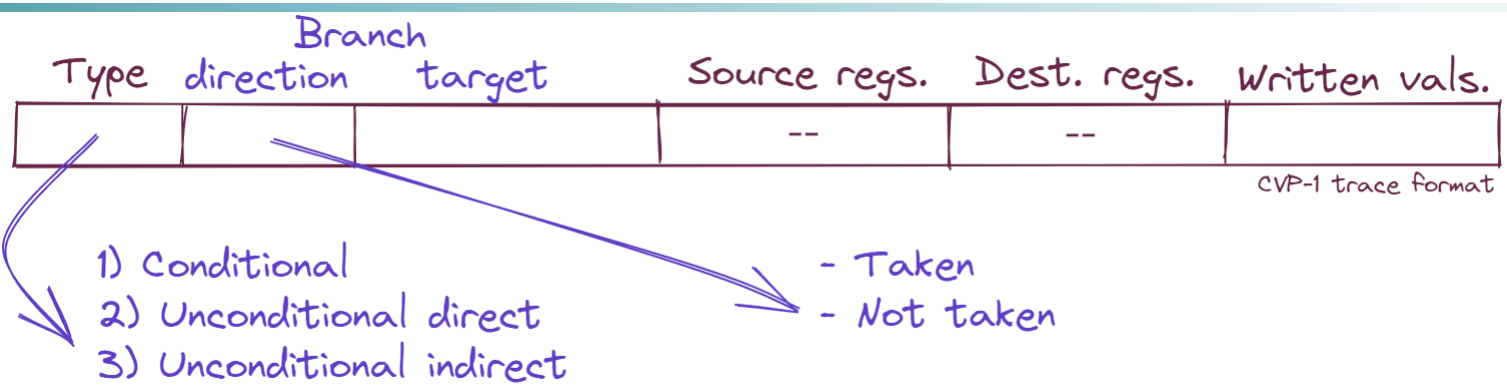
Improvement call-stack

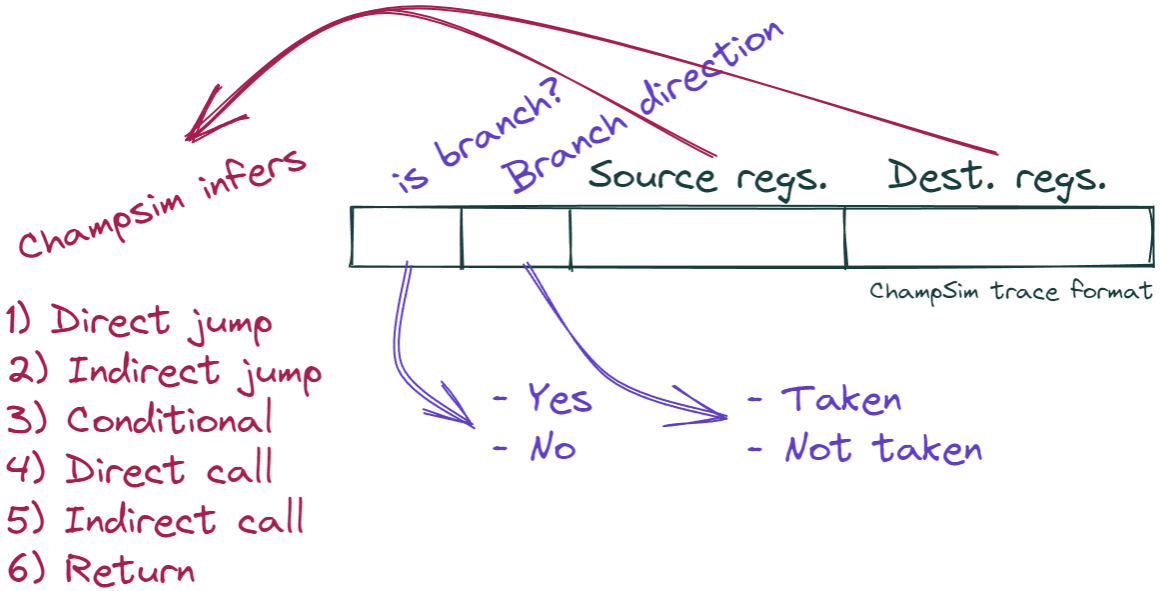




Conversion of branch instructions

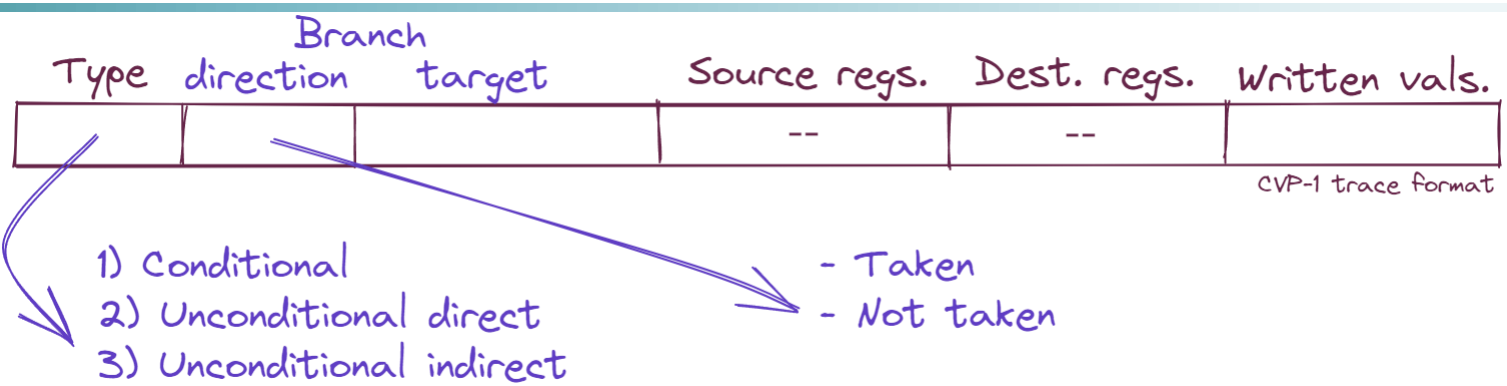
Improvement call-stack

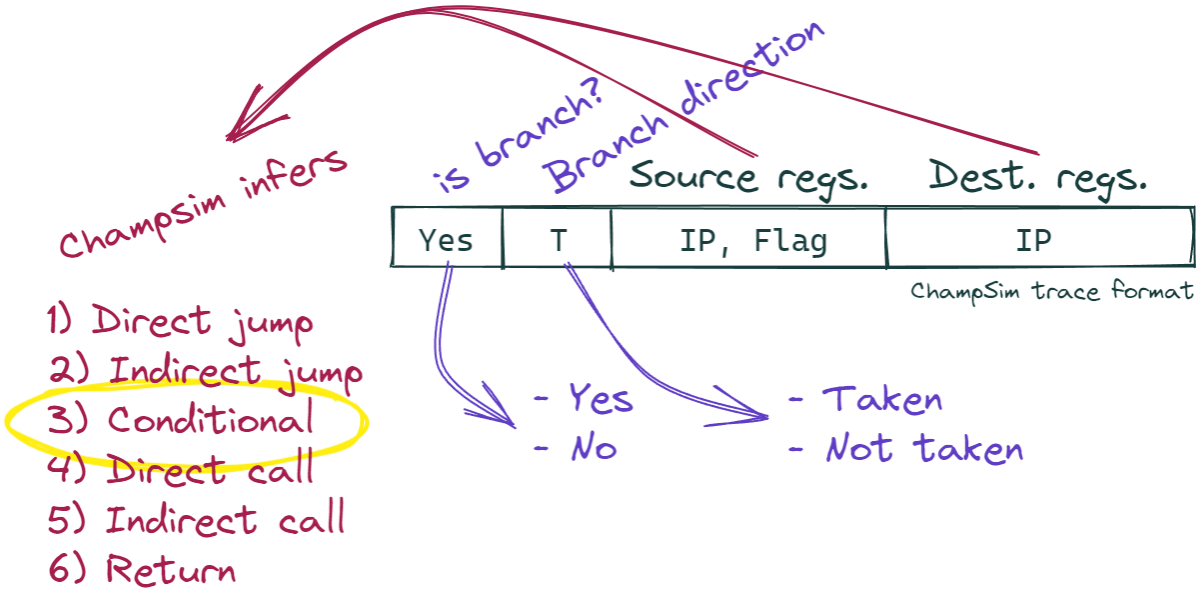




Conversion of branch instructions

Improvement call-stack





Conversion of branch instructions

Improvement call-stack

BLR X1

Branch with link to register

Type	Branch direction	target	Source regs.	Dest. regs.	Written vals.
Unc. ind.	Taken	Taken_address	X1	X30	Return_addr

CVP-1 trace format

is branch?	Branch direction	Source regs.	Dest. regs.

ChampSim trace format

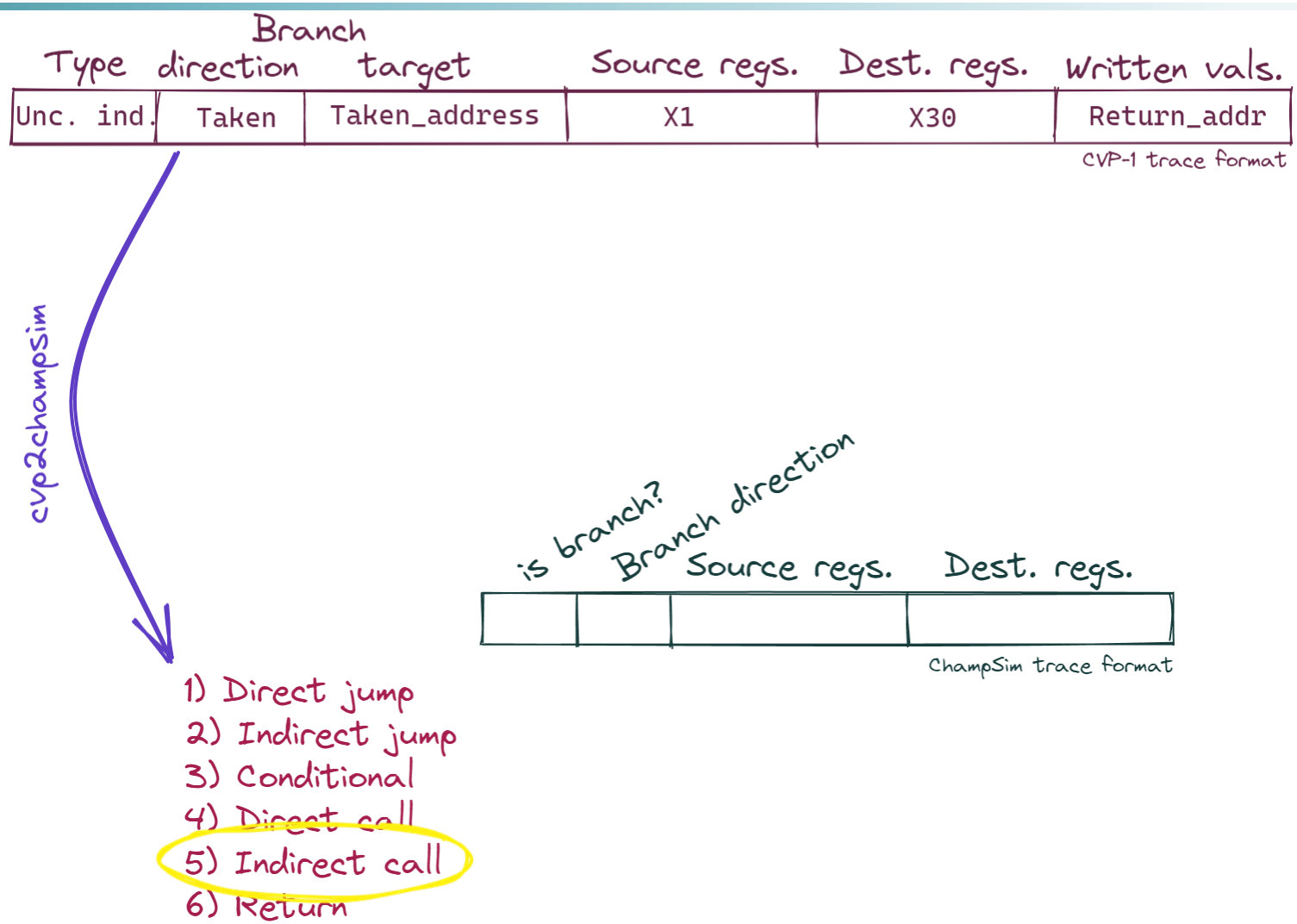
1) Direct jump
2) Indirect jump
3) Conditional
4) Direct call
5) Indirect call
6) Return

Conversion of branch instructions

Improvement call-stack

BLR X1

Branch with link to register

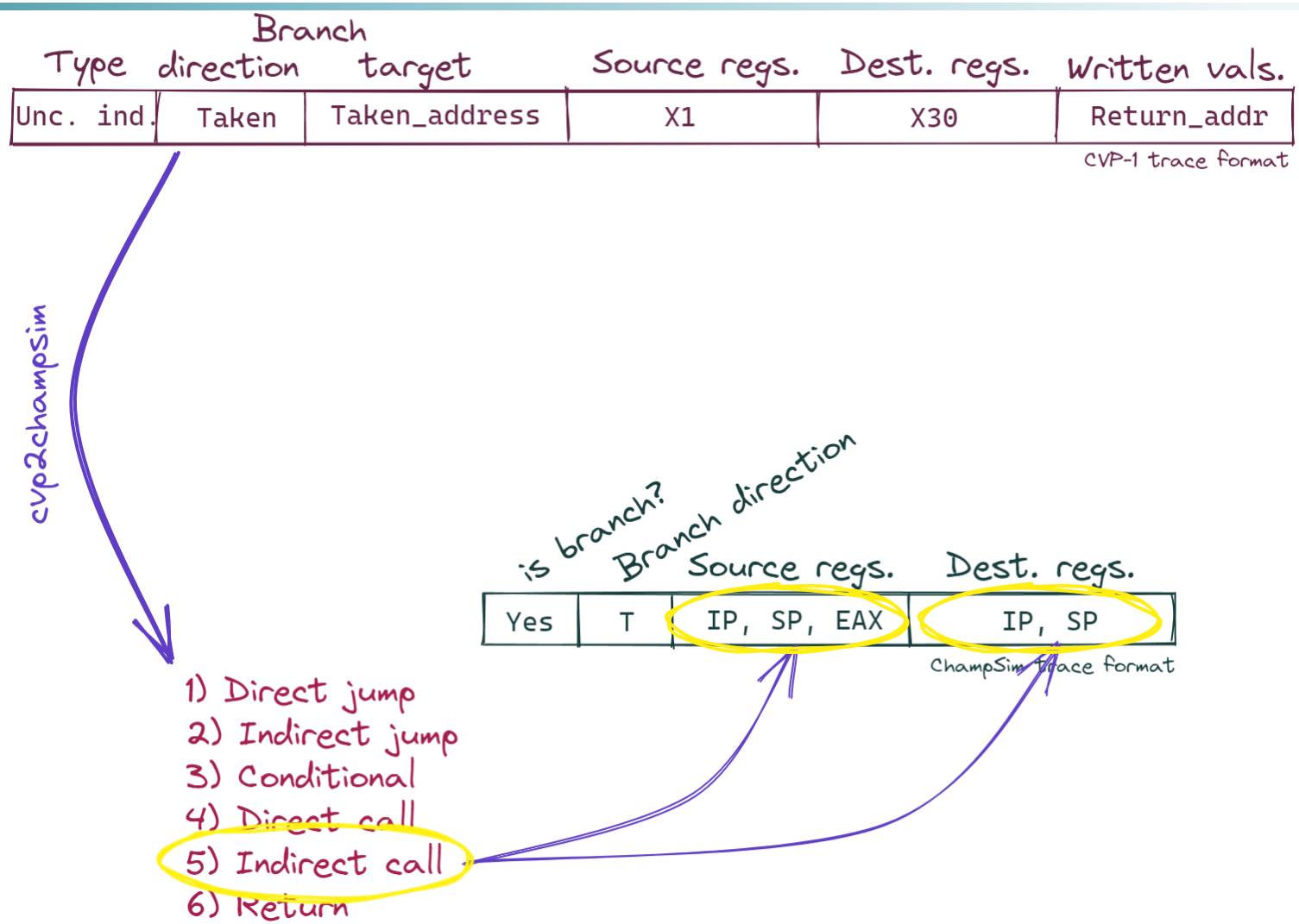


Conversion of branch instructions

Improvement call-stack

BLR X1

Branch with link to register



Conversion of branch instructions

Improvement call-stack

BLR X30

Branch with link to register

Type	Branch direction	target	Source regs.	Dest. regs.	Written vals.
Unc. ind.	Taken	Taken_address	X30	X30	Return_addr

CVP-1 trace format

is branch?	Branch direction	Source regs.	Dest. regs.

ChampSim trace format

Misclassifies BLR X30

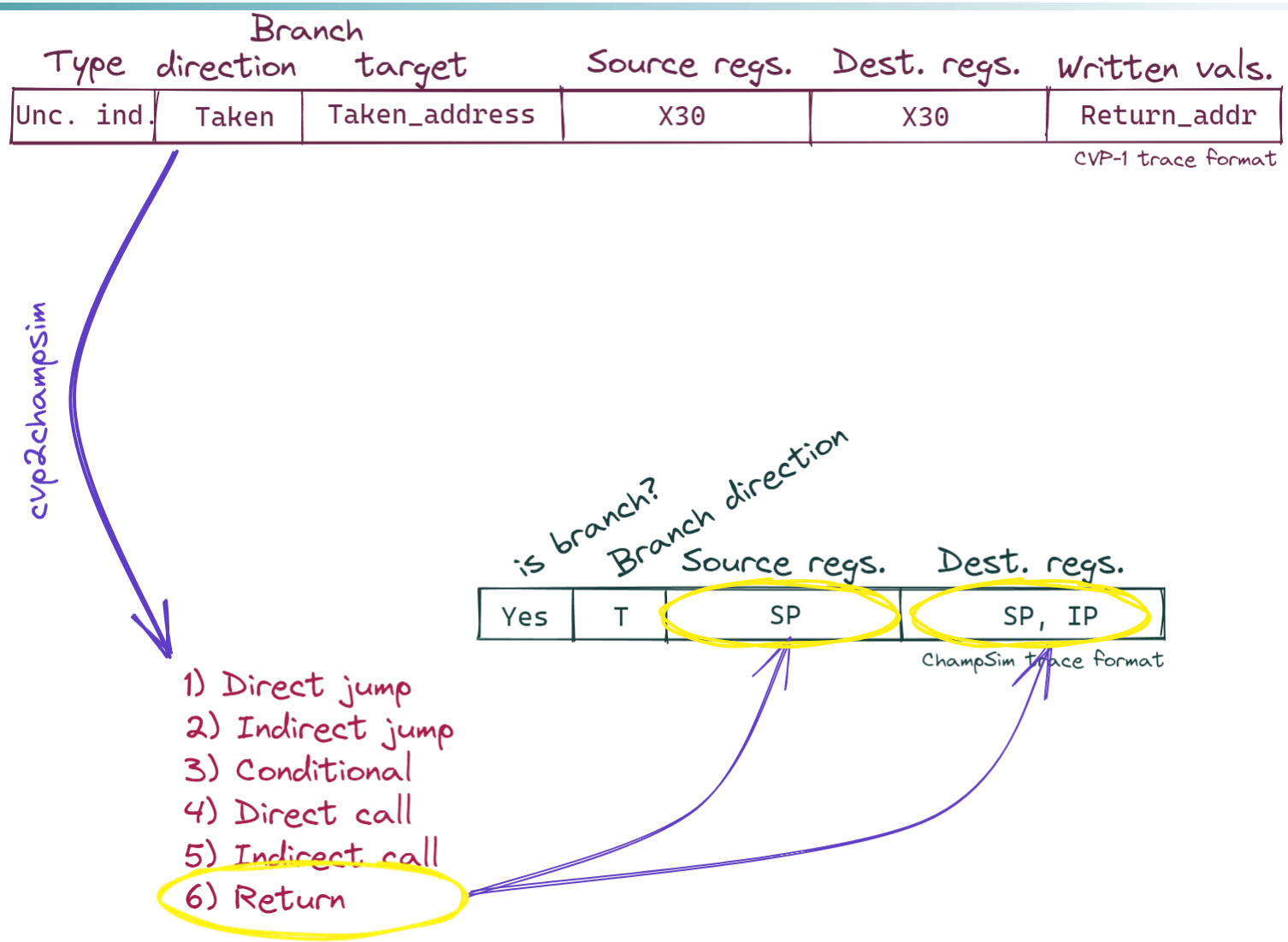
1) Direct jump
2) Indirect jump
3) Conditional
4) Direct call
5) Indirect call
6) Return

Conversion of branch instructions

Improvement call-stack

BLR X30

Branch with link to register



Misclassifies BLR X30

Conversion of branch instructions

Improvement call-stack

BLR X30

Branch with link to register

Type	Branch direction	Branch target	Source regs.	Dest. regs.	Written vals.
Unc. ind.	Taken	Taken_address	X30	X30	Return_addr

CVP-1 trace format

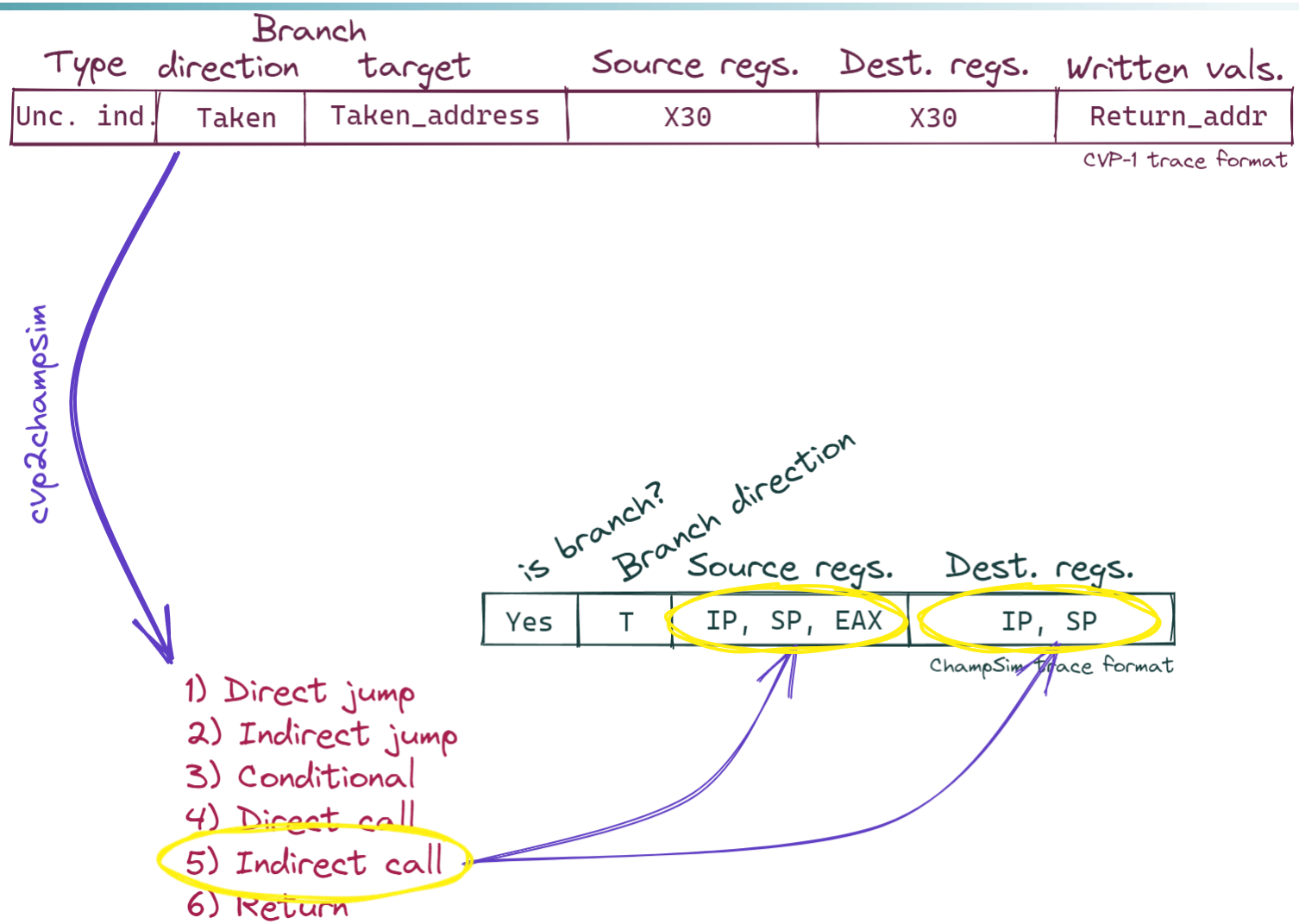
Fix the branch-type identification

Conversion of branch instructions

Improvement call-stack

BLR X30

Branch with link to register



Fix the branch-type identification

Outline

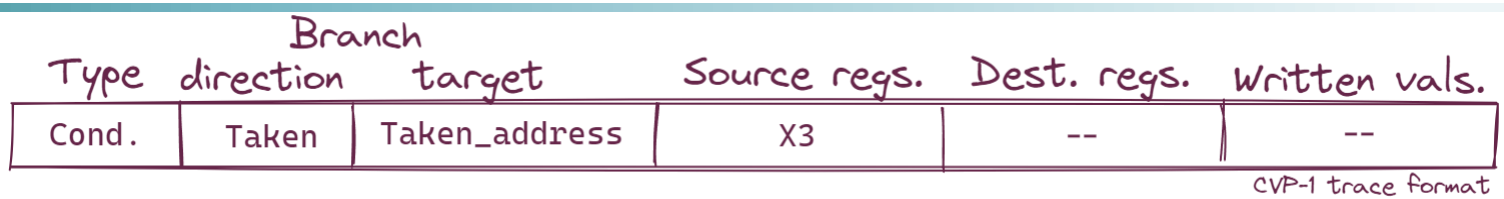
- Introduction
- Conversion of memory instructions
- Conversion of branch instructions
  - Improvement call-stack
  - Improvement branch-regs
  - Improvement flag-reg
- Experimental evaluation
- Conclusion

Conversion of branch instructions

Improvement branch-regs

```
CBZ X3, Taken_Address
```

Compare and branch on zero

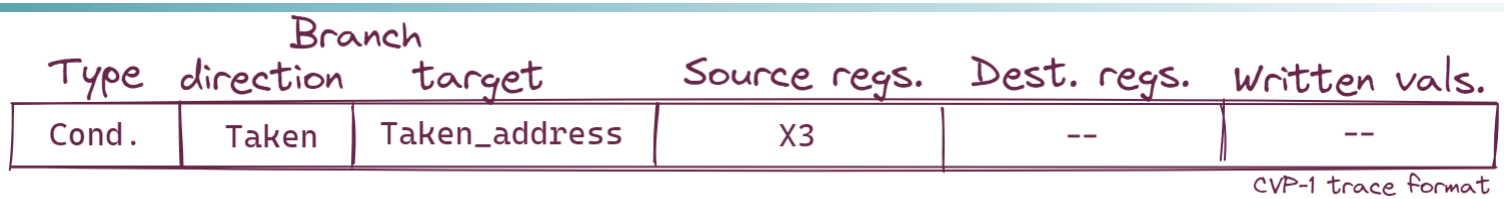


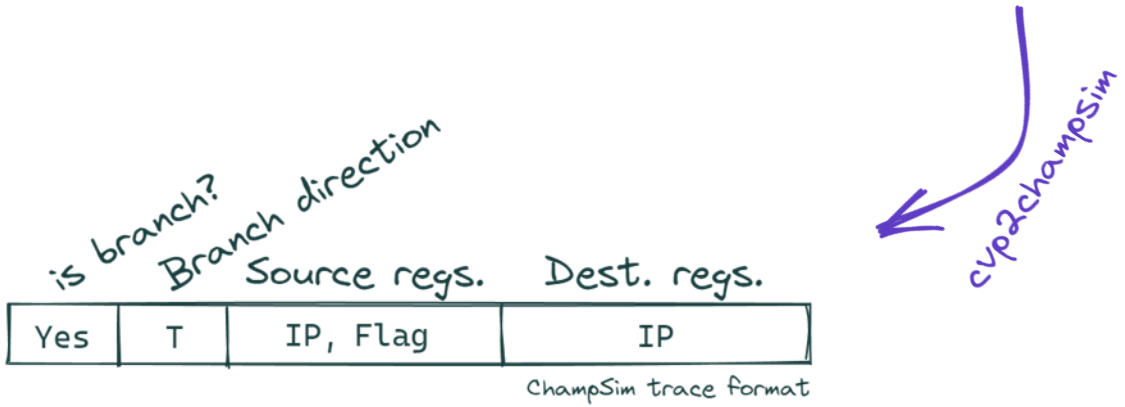
Conversion of branch instructions

Improvement branch-regs

```
CBZ X3, Taken_Address
```

Compare and branch on zero



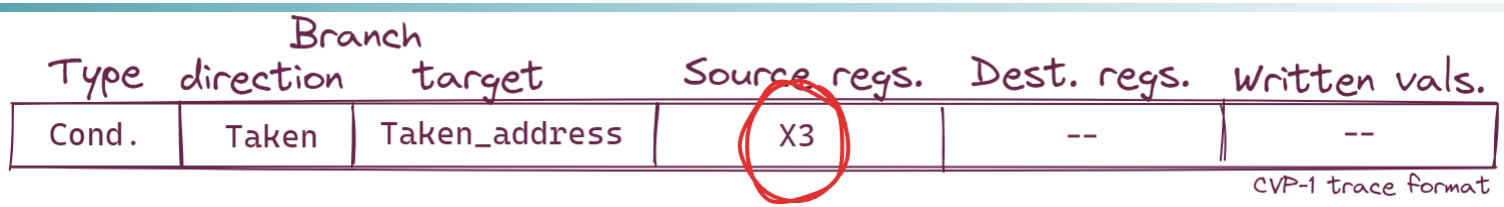


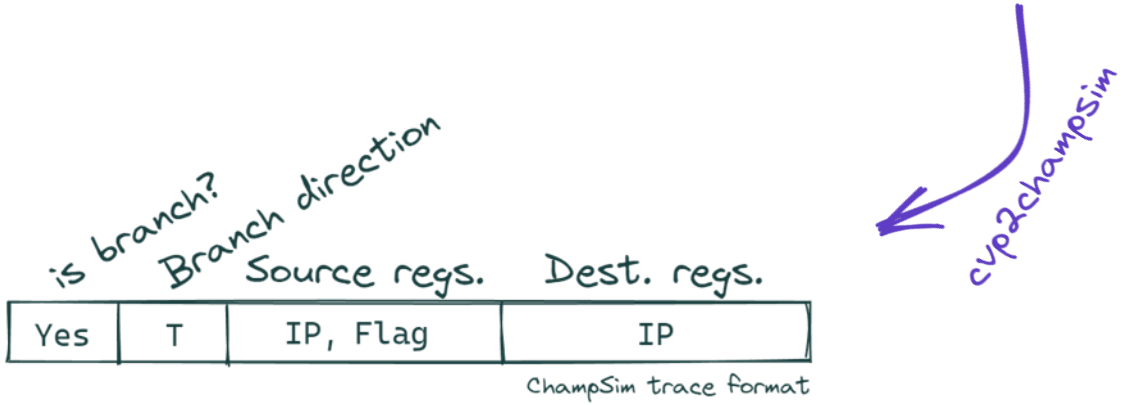
Conversion of branch instructions

Improvement branch-regs

```
CBZ X3, Taken_Address
```

Compare and branch on zero





Registers of the CVP-1 branch instructions are not conveyed.

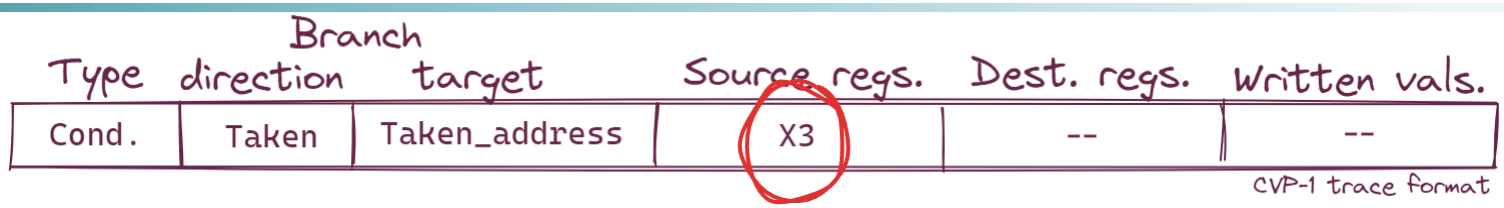
- Dependences with previous instructions are lost.

Conversion of branch instructions

Improvement branch-regs

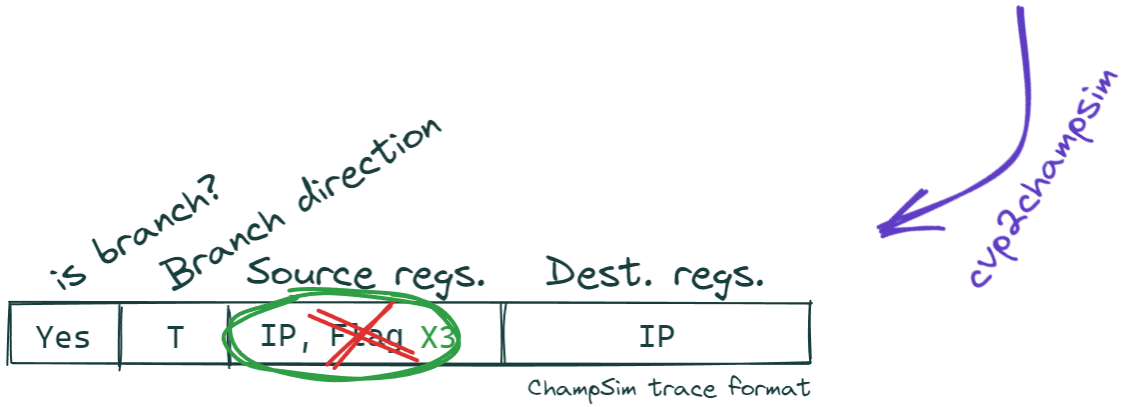
```
CBZ X3, Taken_Address
```

Compare and branch on zero



Convey the registers in the CVP-1 trace to preserve the original dependences.

- Applies similarly to the other types of branches.
- Minor change in ChampSim branch inference.



Outline

- Introduction
- Conversion of memory instructions
- Conversion of branch instructions
  - Improvement call-stack
  - Improvement branch-regs
  - Improvement flag-reg
- Experimental evaluation
- Conclusion

Conversion of branch instructions

Improvement flag-reg

```
CMP X3, #0
```

Dependence through the flag register

```
BEQ Taken_Address
```

Branch if the Z flag is set

Conversion of branch instructions

Improvement flag-reg

```
CMP X3, #0
```

Type	Source regs.	Dest. regs.	Written vals.
Alu	X3	--	--

CVP-1 trace format

Dependence through the flag register

```
BEQ Taken_Address
```

Branch if the Z flag is set

Type	Branch direction	Branch target	Source regs.	Dest. regs.	Written vals.
Cond.	Taken	Taken_address	--	--	--

CVP-1 trace format

Special-purpose registers not present in the CVP trace.

Conversion of branch instructions

Improvement flag-reg

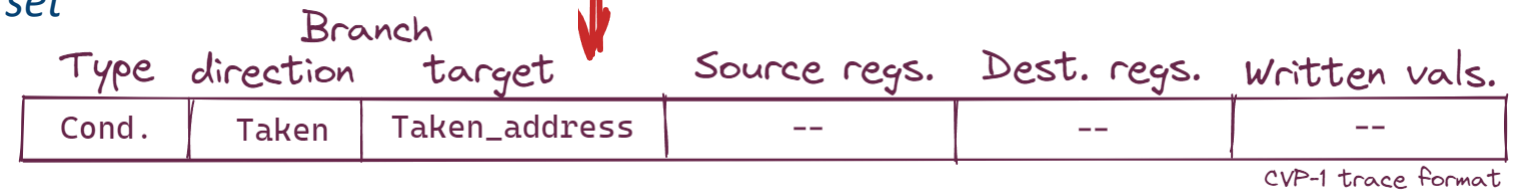
```
CMP X3, #0
```

Dependence through the flag register

Dependence lost

```
BEQ Taken_Address
```

Branch if the Z flag is set



Special-purpose registers not present in the CVP trace.

Conversion of branch instructions

Improvement flag-reg

```
CMP X3, #0
```

Type	Source regs.	Dest. regs.	Written vals.
Alu	X3	--	--

CVP-1 trace format

Dependence through the flag register

Dependence lost

```
BEQ Taken_Address
```

Branch if the Z flag is set

Type	Branch direction	Branch target	Source regs.	Dest. regs.	Written vals.
Cond.	Taken	Taken_address	--	--	--

CVP-1 trace format

cvp2champsim

is branch?	Branch direction	Source regs.	Dest. regs.
Yes	T	IP, Flag	IP

ChampSim trace format

Conversion of branch instructions

Improvement flag-reg

```
CMP X3, #0
```

Dependence through the flag register

```
BEQ Taken_Address
```

Branch if the Z flag is set

Dependence lost

Type	Source regs.	Dest. regs.	Written vals.
Alu	X3	--	--

CVP-1 trace format

cvp2champsim

is branch?	Branch direction	Source regs.	Dest. regs.
No	--	X3	Flag

ChampSim trace format

Type	Branch direction	Branch target	Source regs.	Dest. regs.	Written vals.
Cond.	Taken	Taken_address	--	--	--

CVP-1 trace format

cvp2champsim

is branch?	Branch direction	Source regs.	Dest. regs.
Yes	T	IP, Flag	IP

ChampSim trace format

Add the flag register as a destination to ALU and FP instructions without any.

Conversion of branch instructions

Improvement flag-reg

```
CMP X3, #0
```

Dependence through the flag register

```
BEQ Taken_Address
```

Branch if the Z flag is set

Type	Source regs.	Dest. regs.	Written vals.
Alu	X3	--	--

CVP-1 trace format

cvp2champsim

is branch?	Branch direction	Source regs.	Dest. regs.
No	--	X3	Flag

ChampSim trace format

Dependence lost

Type	Branch direction	Branch target	Source regs.	Dest. regs.	Written vals.
Cond.	Taken	Taken_address	--	--	--

CVP-1 trace format

cvp2champsim

is branch?	Branch direction	Source regs.	Dest. regs.
Yes	T	IP, Flag	IP

ChampSim trace format

Dependence restored

Add the flag register as a destination to ALU and FP instructions without any.

Outline

- Introduction
- Conversion of memory instructions
- Conversion of branch instructions
- Experimental evaluation
- Conclusion

Experimental evaluation
Setup

Analysis of the impact of the converter improvements.

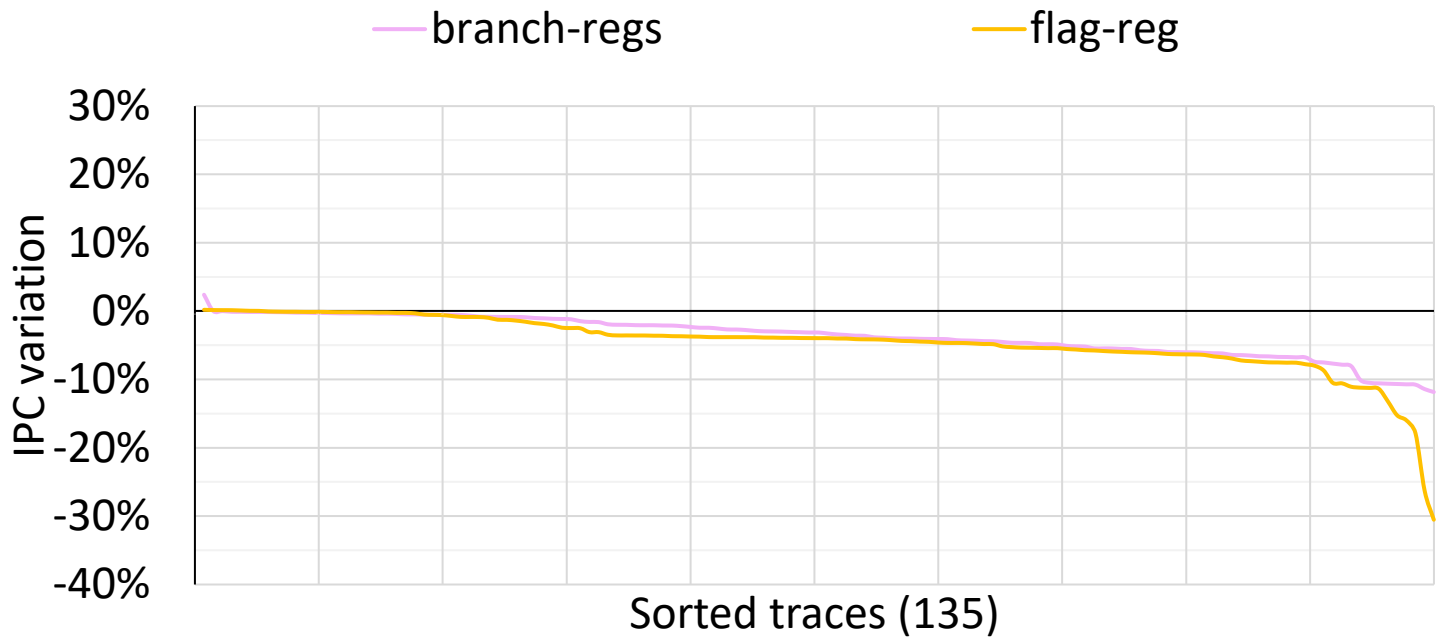
- CVP-1 public traces.
- ChampSim main branch (*commit 2bba2bd*).
- Processor configuration resembling Intel Icelake
  - 16K-entry BTB, 64KB ITTAGE, and TAGE-SC-L predictors.
  - Ip-stride prefetcher at the L1D cache and a next-line prefetcher at the L2 cache.

Reevaluation of the IPC-1 Championship (just for fun, prefetchers were tuned for the old traces).

- IPC-1 traces.
- ChampSim version provided in the championship.
- Original code of the prefetchers.

Experimental evaluation

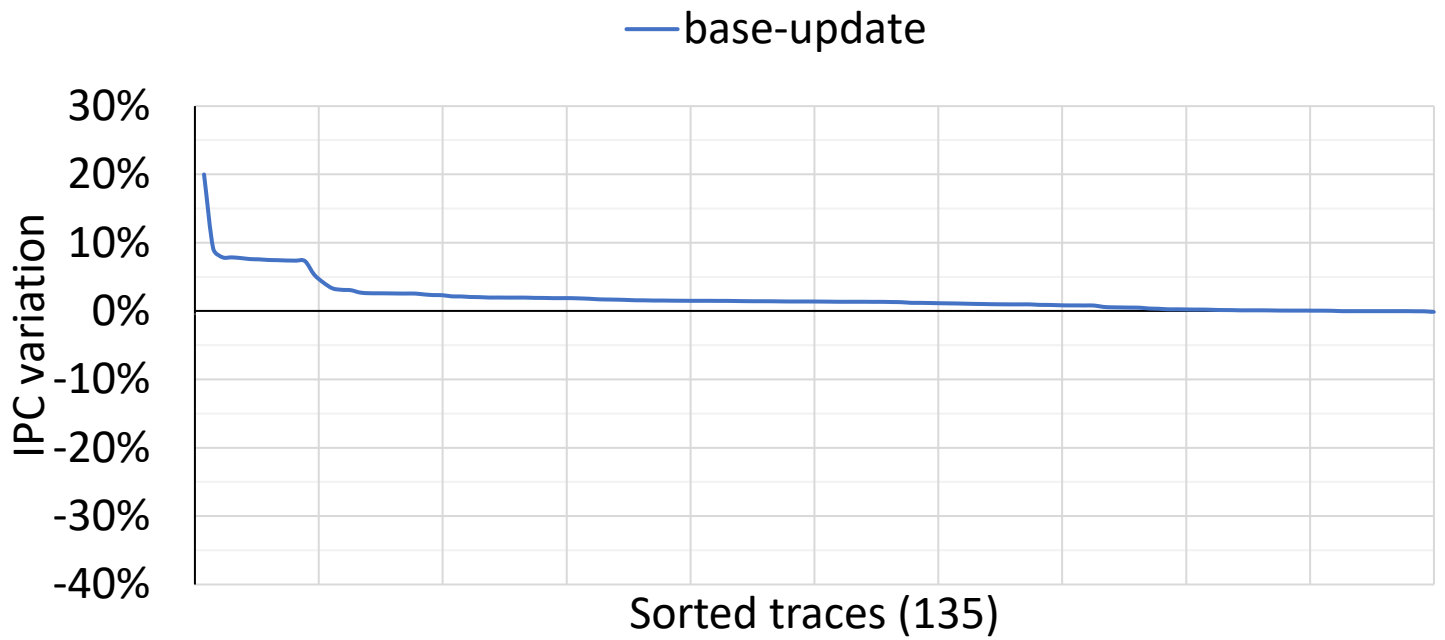
Performance impact



Improvements *flag-reg* and *branch-regs*:

- Restore the dependence of branches with previous instructions → Reduce IPC.

Experimental evaluation
Performance impact



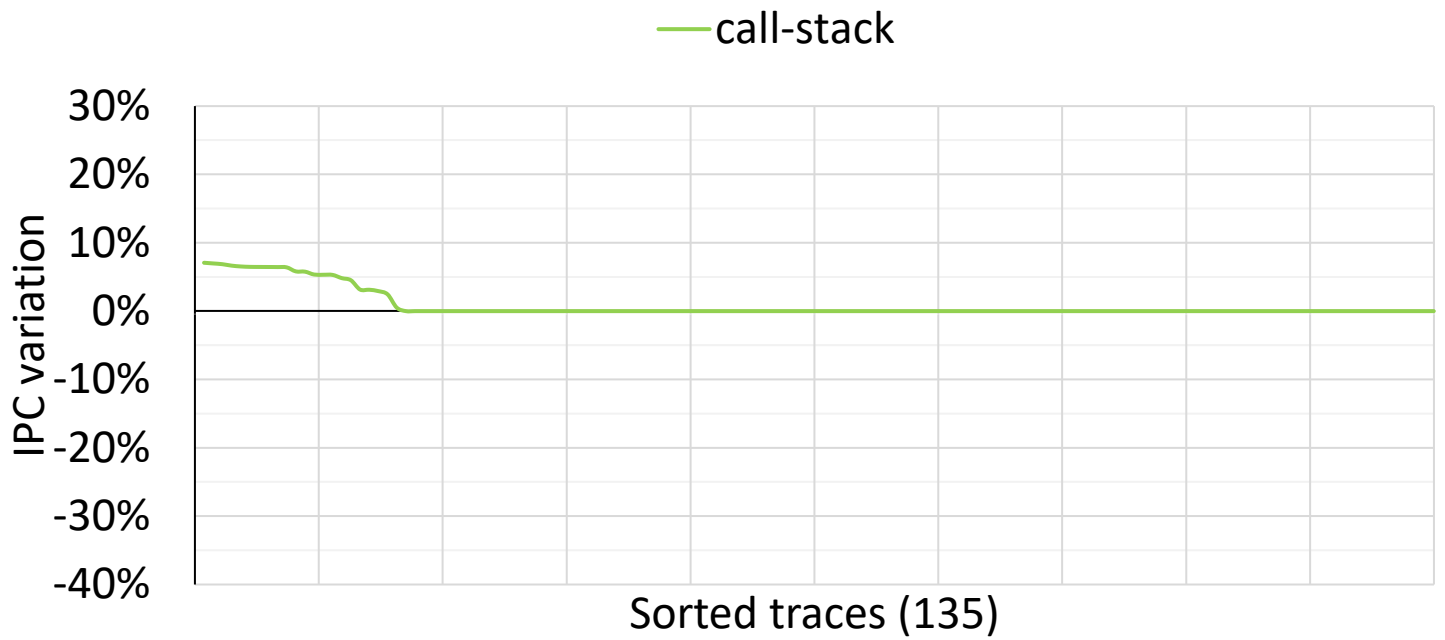
Improvements *flag-reg* and *branch-regs*:

- Restore the dependence of branches with previous instructions → Reduce IPC.

Improvement *base-update*:

- Makes base registers available earlier → Increase IPC.

Experimental evaluation
Performance impact



Improvements *flag-reg* and *branch-regs*:

- Restore the dependence of branches with previous instructions → Reduce IPC.

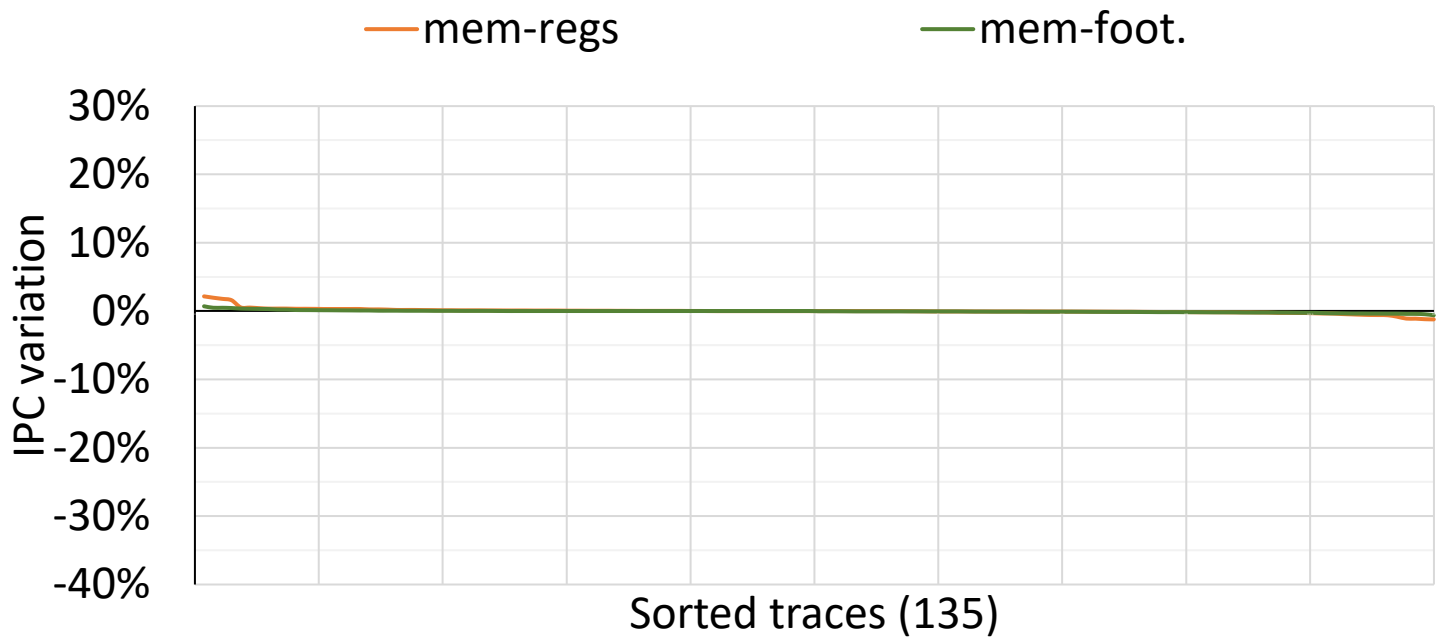
Improvement *base-update*:

- Makes base registers available earlier → Increase IPC.

Improvement *call-stack*:

- Reduces branch MPKI → Increase IPC.

Experimental evaluation
Performance impact

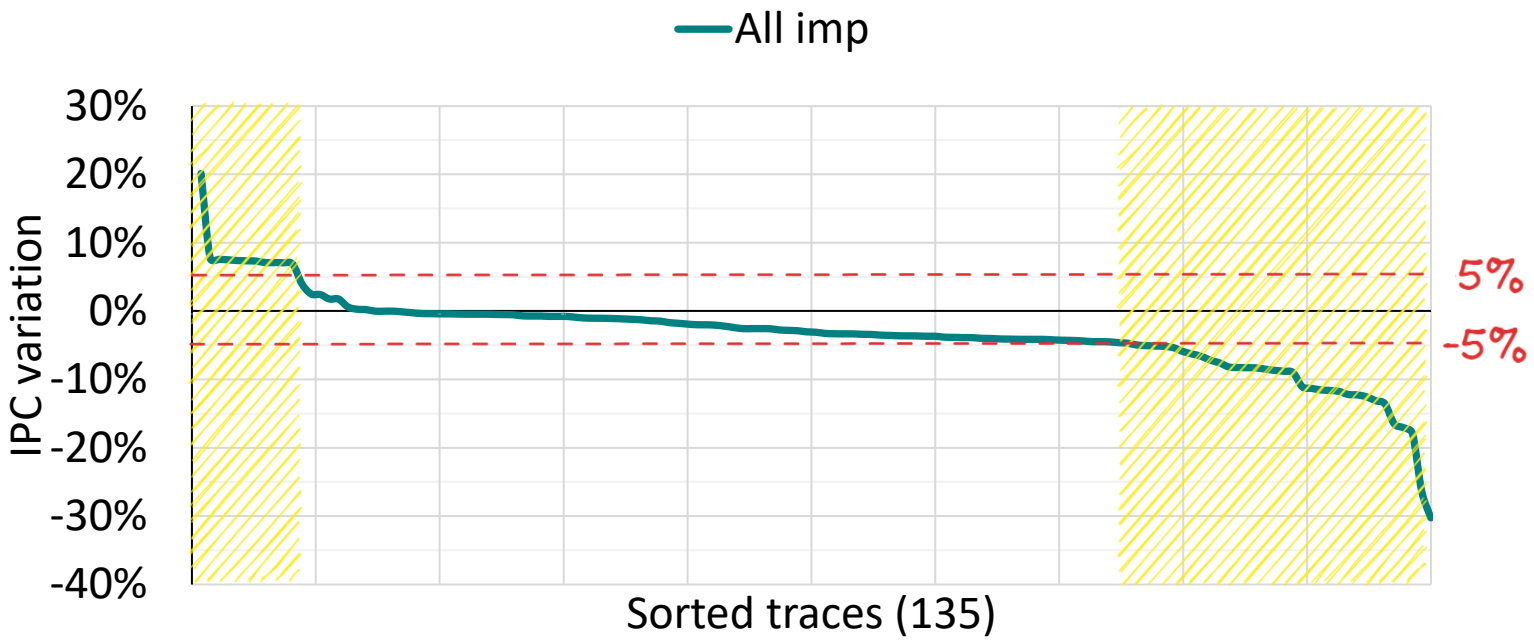


Improvements *mem-footprint* and *mem-regs* have a minimal impact:

- *Mem-footprint*: cacheblocks are accessed anyway plus are easy to prefetch.
- *Mem-regs*: no clear positive or negative effect but enables the base-update improvement.

Experimental evaluation

Performance impact



Performance deviates > 5% for 43 out of the 135 traces.

On average IPC reduces by 3.5%.

Experimental evaluation

Performance impact

Competition traces			Improved traces		
Rank	Prefetcher	SpeedUp	Rank	Prefetcher	SpeedUp
1	EPI	1.2951	1	EPI	1.3818
2	D-JOLT	1.2884	2	D-JOLT	1.3696
3	FNL+MMA	1.2861	3	JIP	1.3588
4	BARÇA	1.2832	4	BARÇA	1.3570
5	PIPS	1.2799	5	FNL+MMA	1.3517
6	JIP	1.2768	6	PIPS	1.3444
7	MANA	1.2658	7	MANA	1.3092
8	TAP	1.2351	8	TAP	1.2915

With the new traces, speedup grows for all prefetchers.

No big changes in the competition (except for JIP).

Conclusion

Our study highlights the need for carefully vetting the tools to conduct research.

We advocate for sharing tools, but

- Some are built with very specific uses in mind.
- Using them for a different purpose can lead to significant inaccuracies.

We should model the behavior of workloads as faithfully as possible.

We propose several improvements to the CVP-1 to Champsim trace converter:

- Better conveys the characteristics of the original workloads.
  - Performance difference greater than 5% in 1/3 of the traces.
- Easily adaptable to a new trace format.

Rebasing Microarchitectural Research with Industry Traces

J. Feliu[1], A. Perais[2], D. A. Jiménez[3], and A. Ros[4]

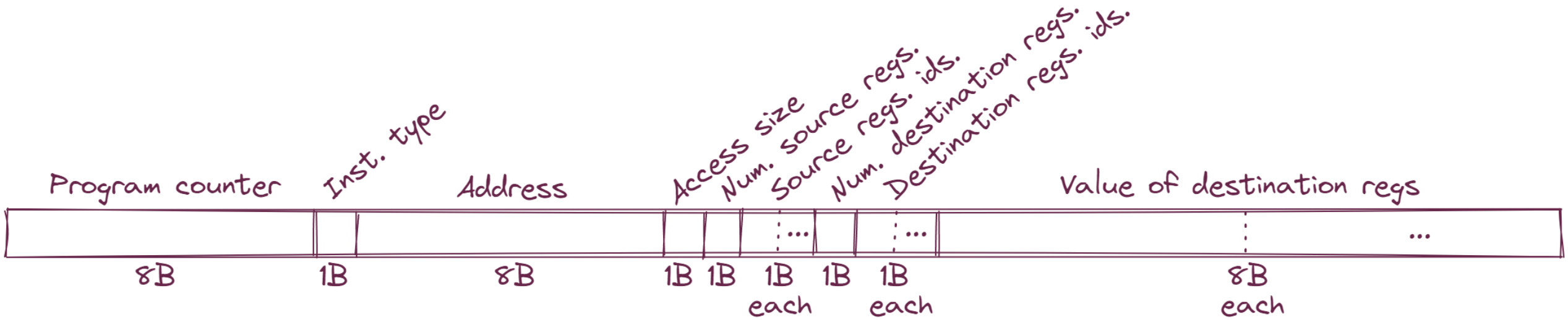
[1] Universitat Politècnica de València, Spain
[2] Université Grenoble Alpes, CNRS, Grenoble INP, TIMA, France
[3] Texas A&M University, USA
[4] University of Murcia, Spain

Thank you! Questions??

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 819134), from the MCIN/AEI/10.13039/501100011033/ and the "ERDF A way of making Europe", EU (grants PID2021-123627OB-C51 and PID2022-136315OB-I00) and the European Union NextGenerationEU/PRTR (grants RYC2021-030862-I and TED2021-130233B-C33/C32), from the National Science Foundation (grants CNS-1938064 and CCF-1912617), as well as generous gifts from Intel. Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

Conversion of memory instructions

Improvement mem-regs



CVP-1 trace format for memory instructions

Conversion of memory instructions

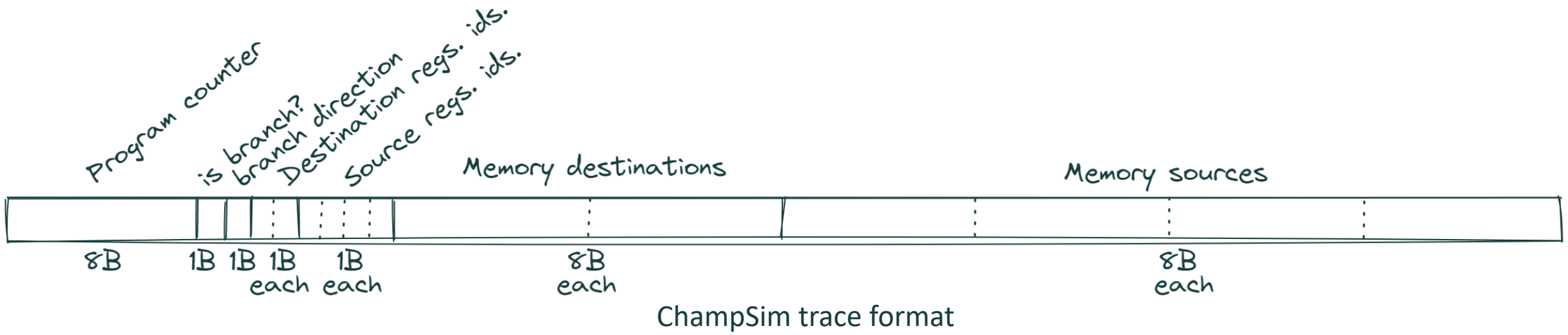
Improvement mem-regs

```
LDP X1, X0, [X0]
```

Load pair → X1 = Mem[X0], X0 = Mem[X0+8]

Type	Address	Size	Source regs.	Dest. regs.	Written vals.
load	addr	8	X0	X1, X0	Val1, Val0

CVP-1 trace format



ChampSim trace format