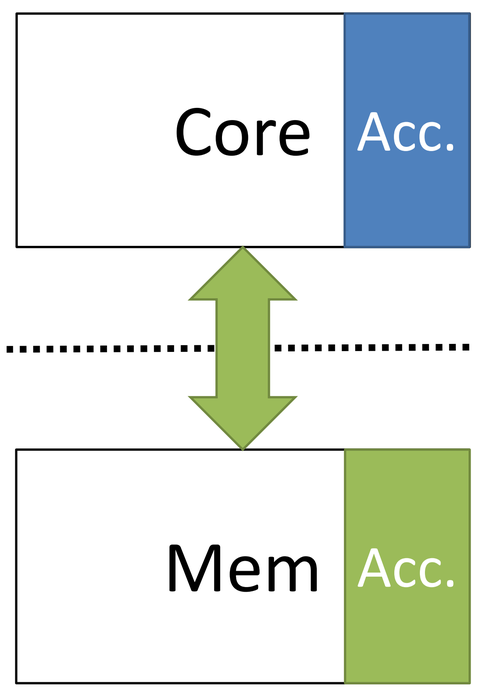
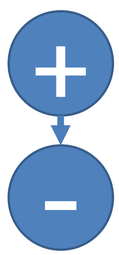
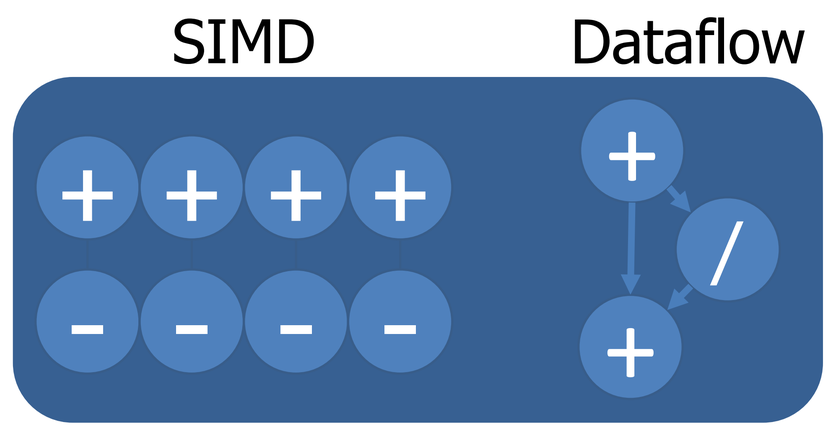
Stream-based Memory Specialization for General Purpose Processors

Zhengrong Wang
Prof. Tony Nowatzki

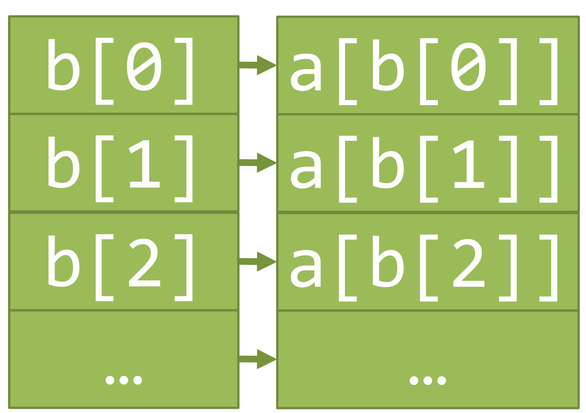
Computation & Memory Specialization







New ISA abstraction for certain computation pattern.

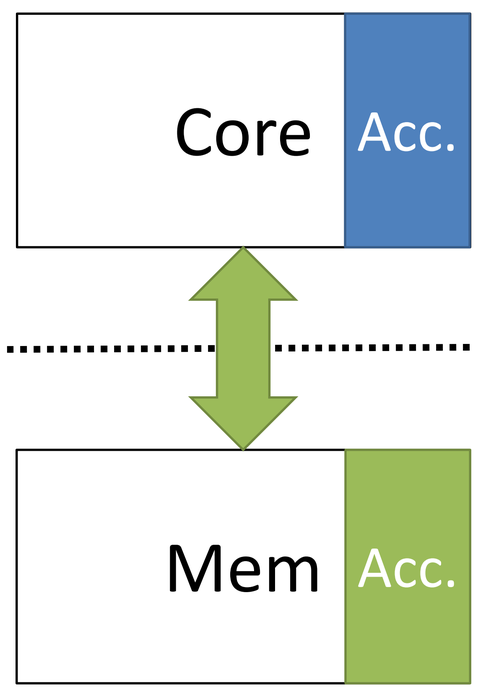


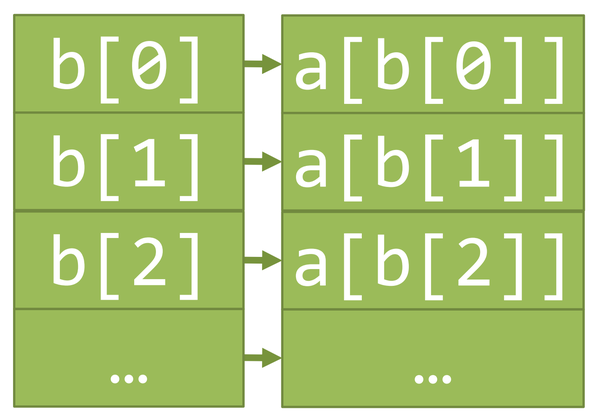
Stream

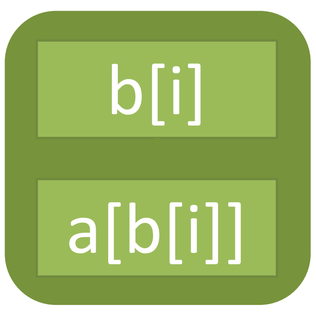
New ISA abstraction for memory access pattern?

Stream: A New ISA Memory Abstraction

- Stream: A decoupled memory access pattern.
- Higher level abstraction in ISA.
  - Decouple memory access.
  - Enable efficient prefetching.
  - Leverage stream information in cache policies.
- 60% memory accesses → streams.
- 1.37× speedup over a traditional O3 processor.







Stream

Outline

- Insight & Opportunities.
- Stream Characteristics.
- Stream ISA Extension.
- Stream-Aware Policies.
- Microarchitecture Extension.
- Evaluation.

Outline

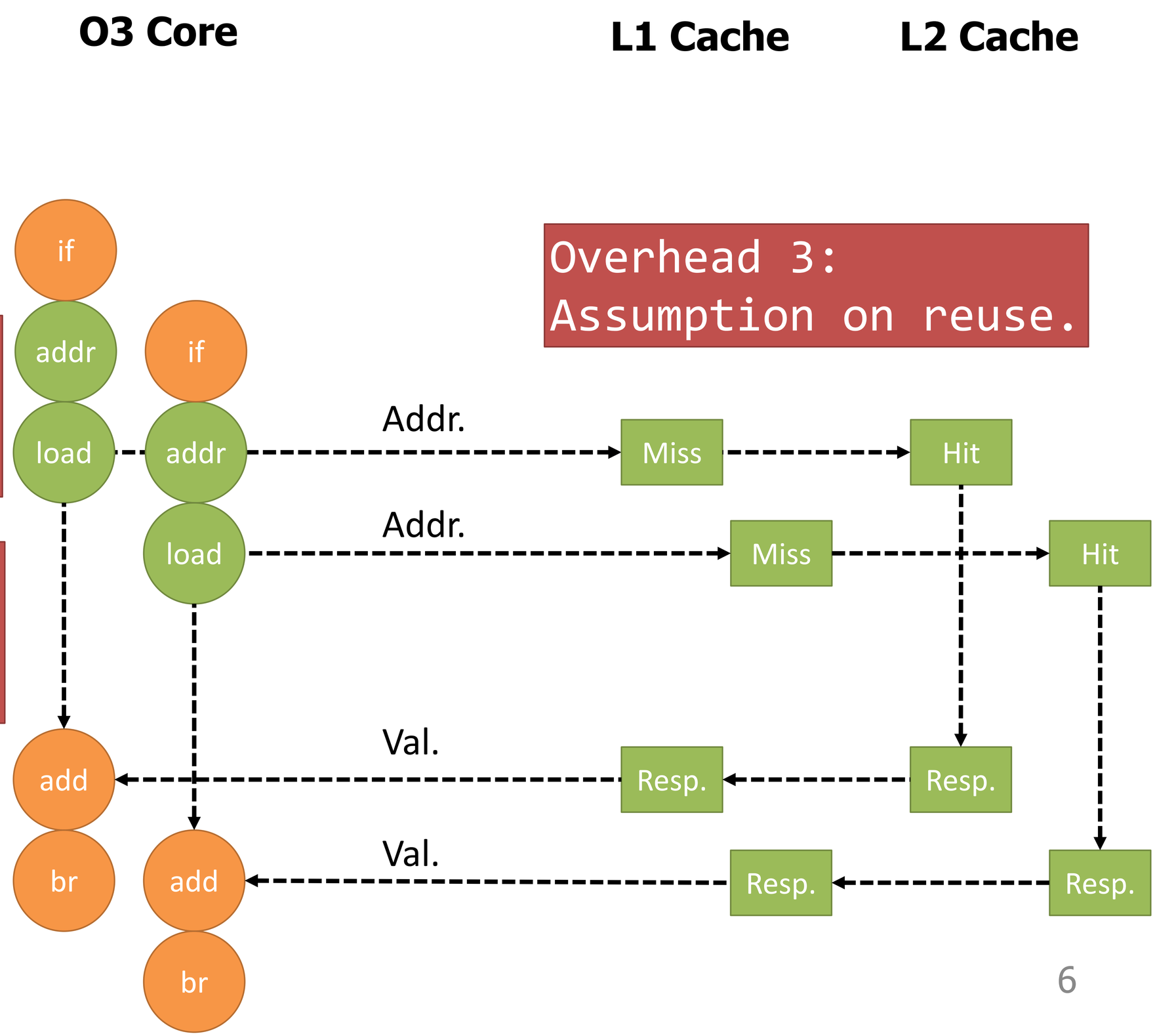
- Insight & Opportunities.
- Stream Characteristics.
- Stream ISA Extension.
- Stream-Aware Policies.
- Microarchitecture Extension.
- Evaluation.

Conventional Memory Abstraction

```
while (i < N) {
  if (cond)
    v += [redacted]
  i++;
}
```

Overhead 2:
Similar address computation/loads.

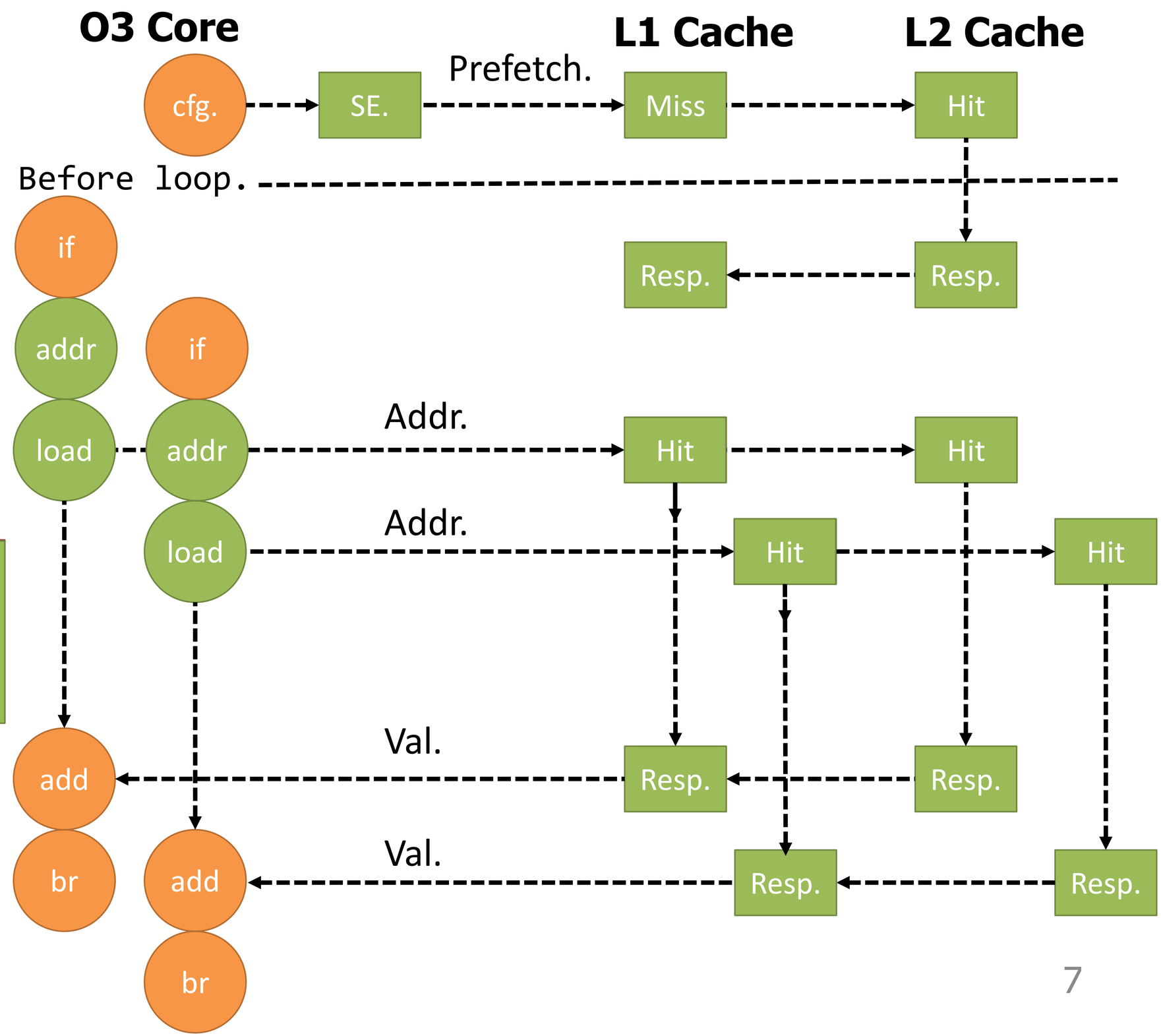
Overhead 1:
Hard to prefetch with control flow.



Overhead 3:
Assumption on reuse.

Opportunity 1: Prefetch with Ctrl. Flow

```
cfg(a[i]);
while (i < N) {
  if (cond)
    v += a[i];
  i++;
}
```



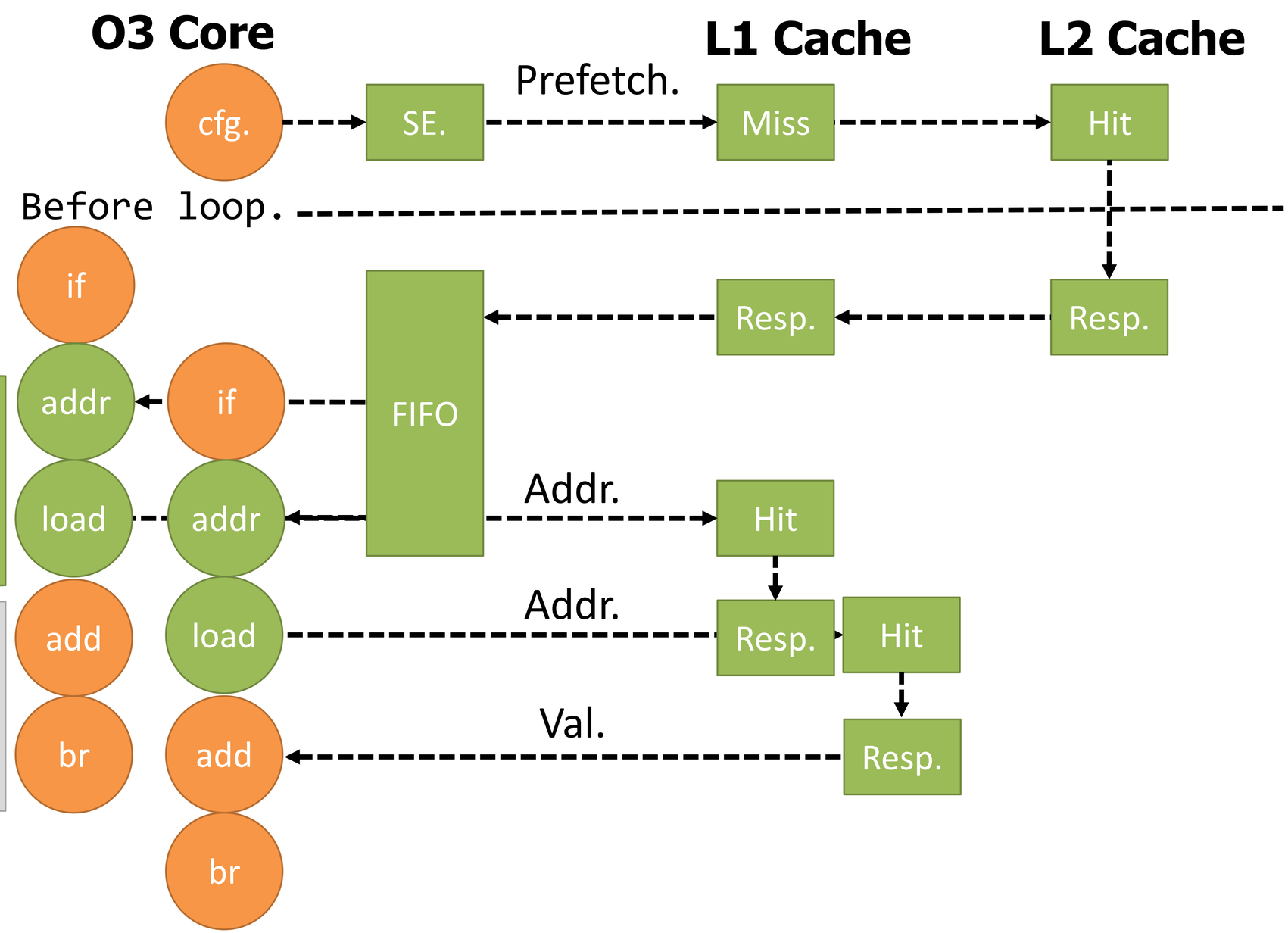
Opportunity 1: Prefetch with control flow.

Opportunity 2: Semi-Binding Prefetch

```
s_a = cfg();
while (i < N) {
  if (cond)
    v += s_a;
  i++;
}
```

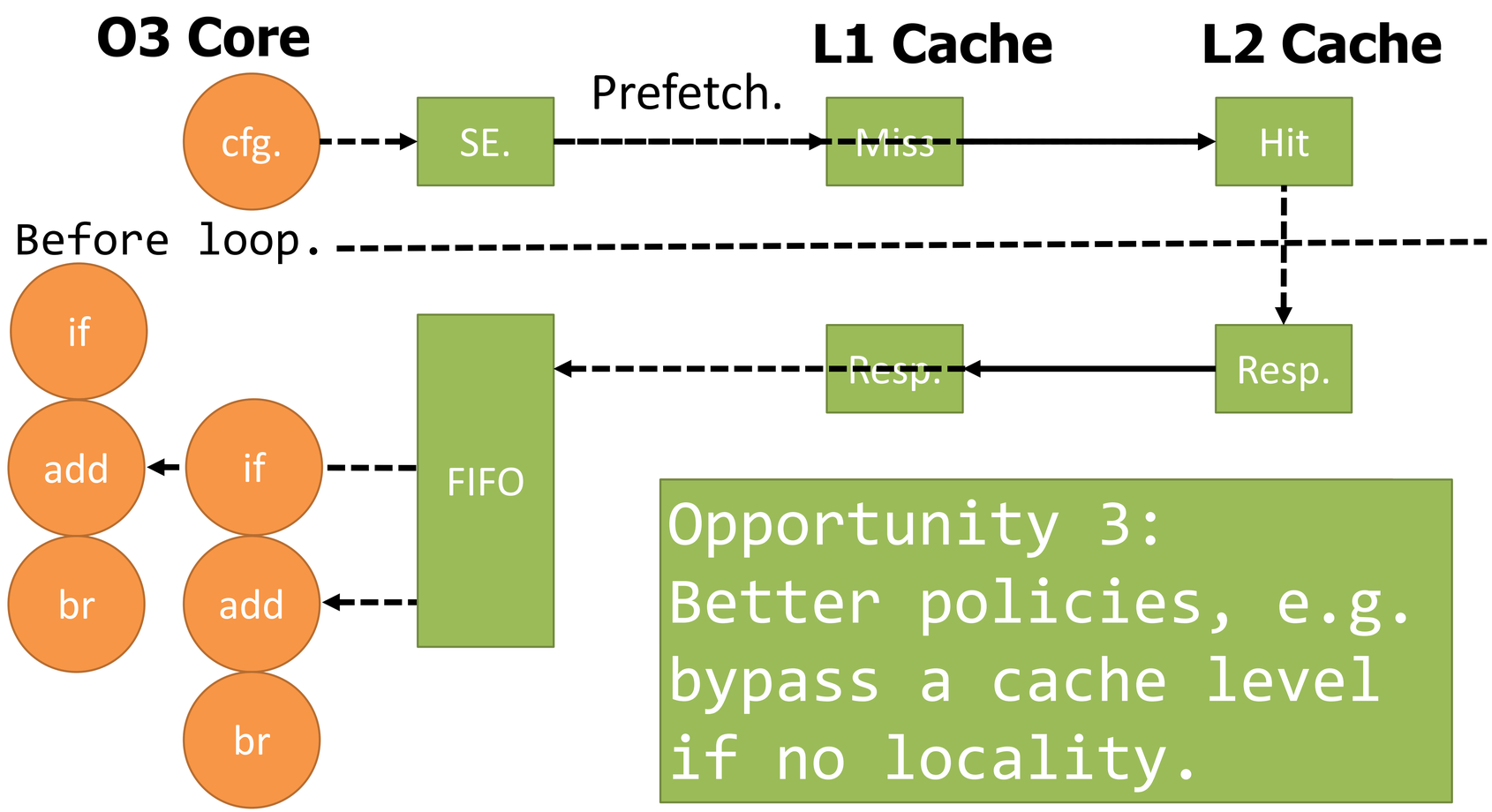
Opportunity 2: Semi-binding prefetch.

Opportunity 1: Prefetch with control flow.



Opportunity 3: Stream-Aware Policies

```
s_a = cfg();
while (i < N) {
  if (cond)
    v += s_a;
}
```



Opportunity 2: Semi-binding prefetch.

Opportunity 1: Prefetch with control flow.

Opportunity 3: Better policies, e.g. bypass a cache level if no locality.

Related Work

- Decouple access execute.
  - Outrider [ISCA'11], DeSC [MICRO'15], etc.
  - Ours: New ISA abstraction for the access engine.
- Prefetching.
  - Stride, IMP [MICRO'15], etc.
  - Ours: Explicit access pattern in ISA.
- Cache bypassing policy.
  - Counter-based [ICCD'05], LLC bypassing [ISCA'11], etc.
  - Ours: Incorporate static stream information.

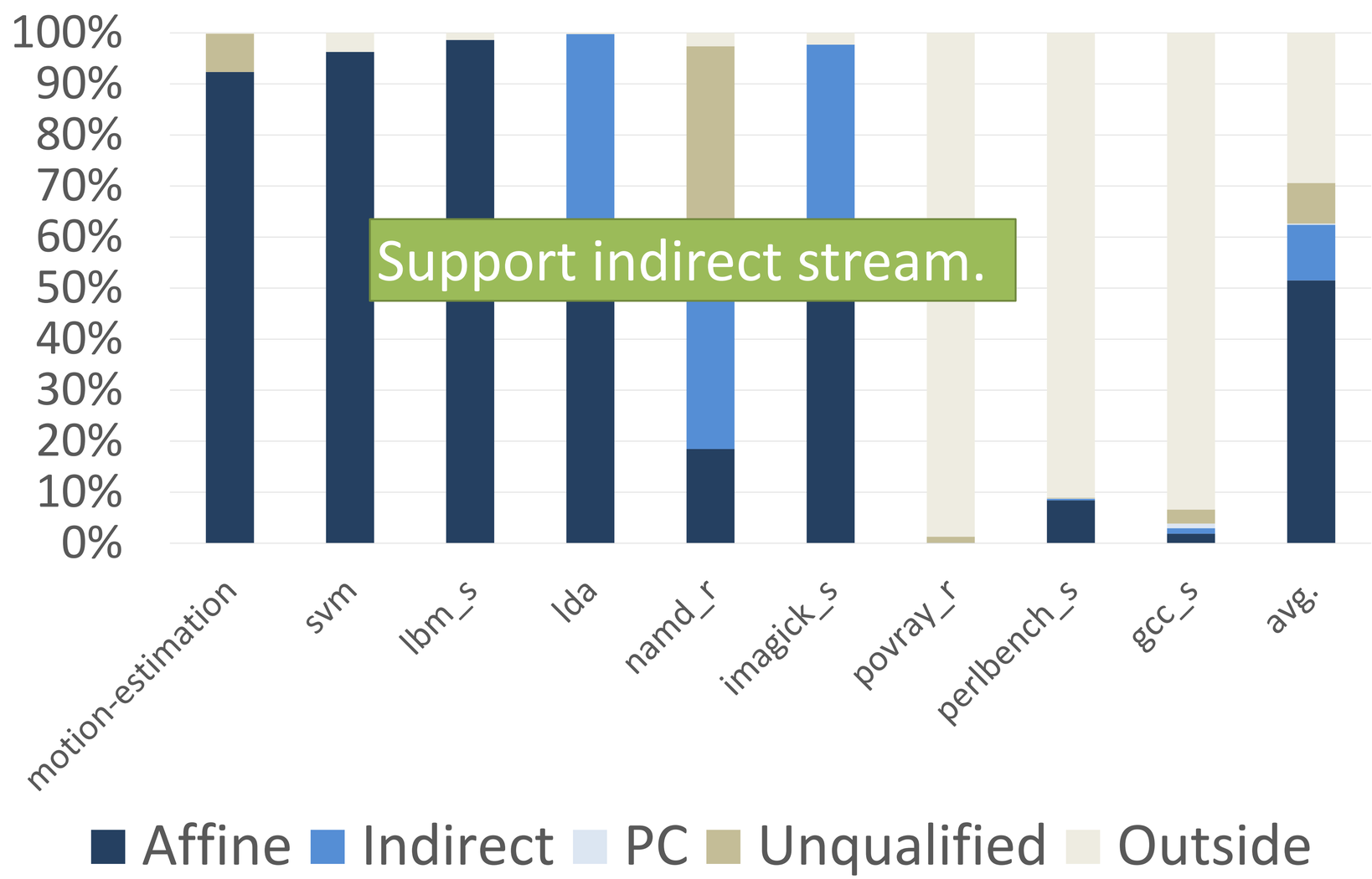
Outline

- Insight & Opportunities.
- Stream Characteristics.
- Stream ISA Extension.
- Stream-Aware Policies.
- Microarchitecture Extension.
- Evaluation.

Stream Characteristics – Stream Type

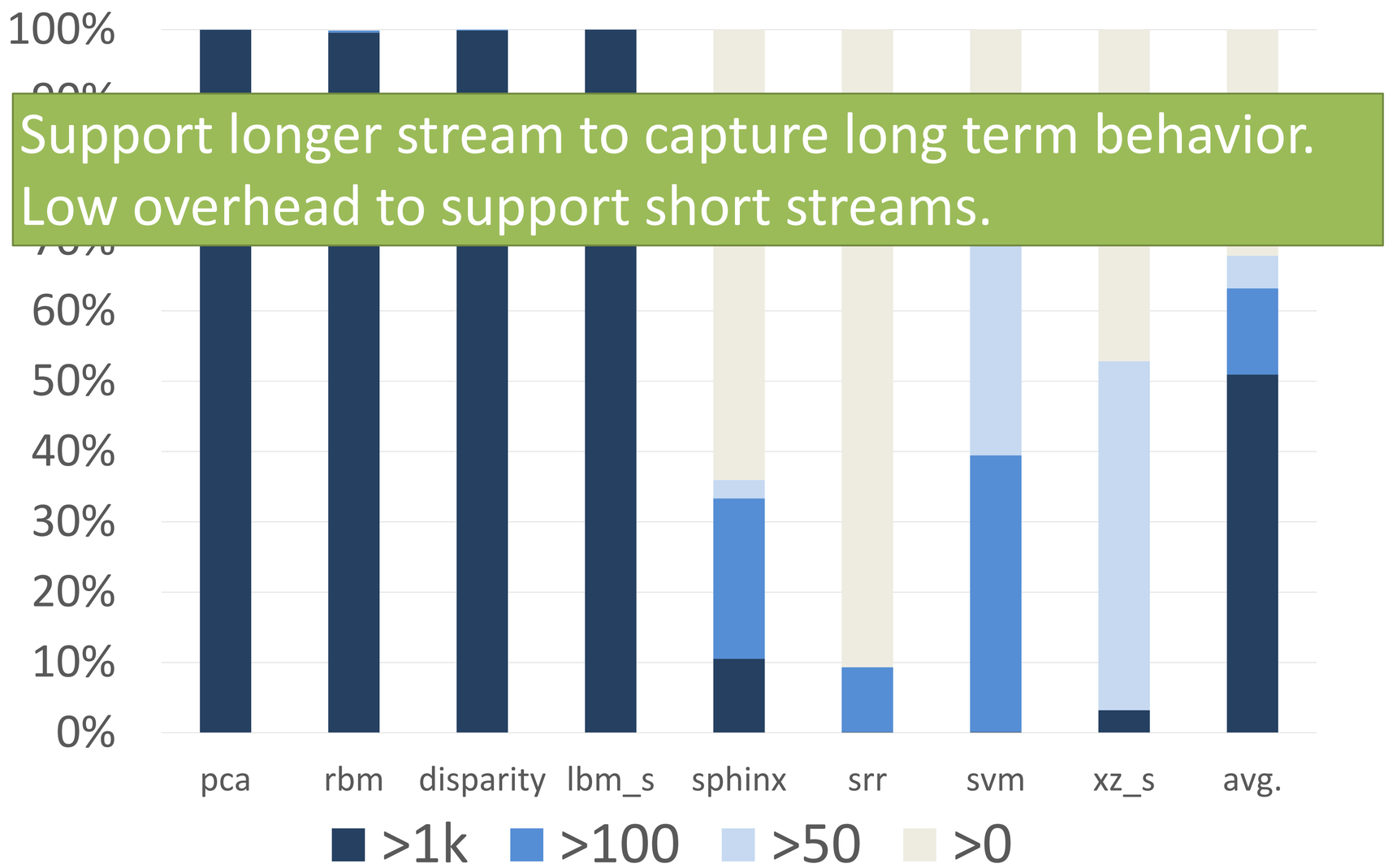
Trace analysis on CortexSuite/SPEC CPU 2017.

- 51.49% affine, 10.19% indirect.
- Indirect streams can be as high as 40%.



Stream Characteristics – Stream Length

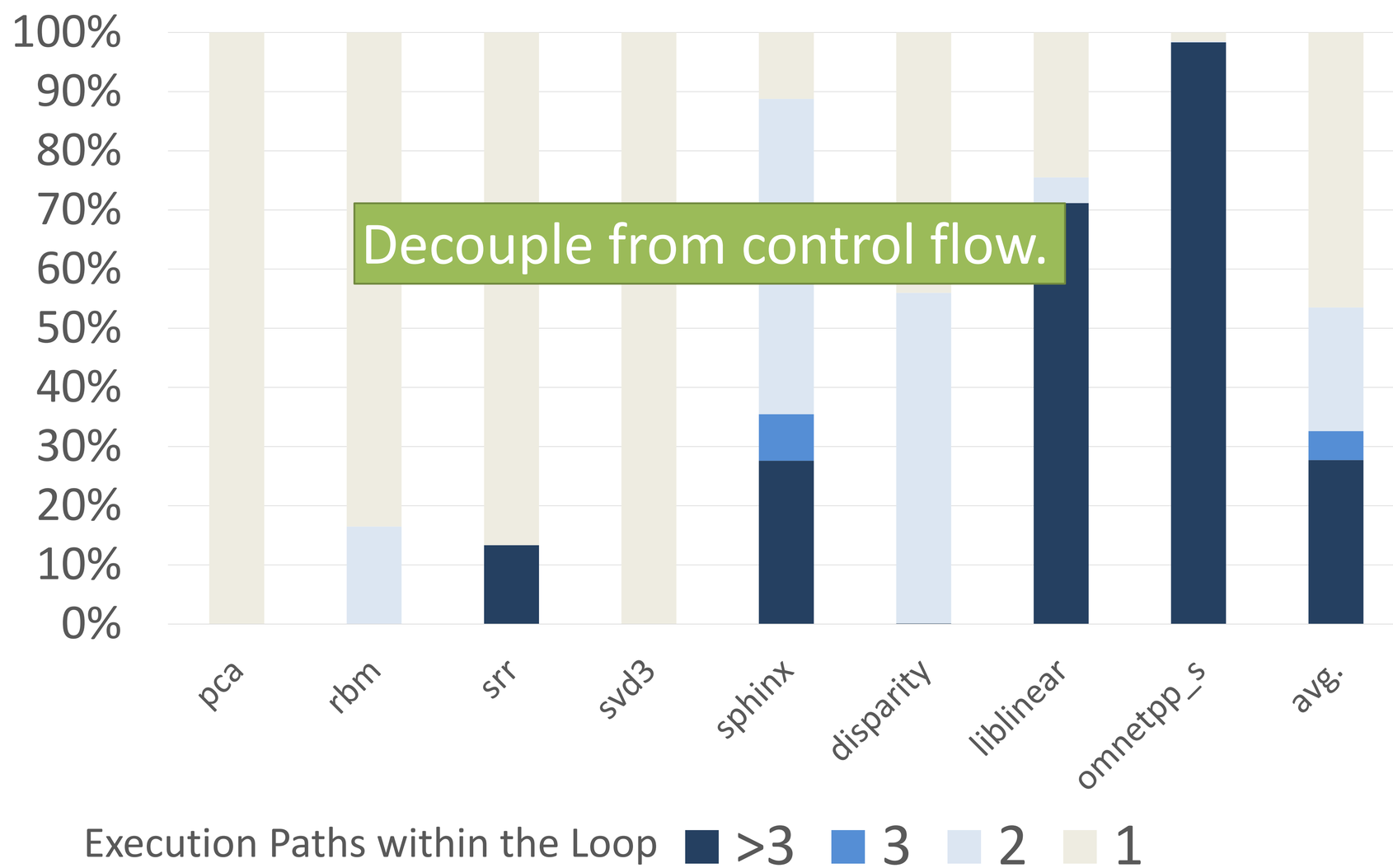
- 51% stream accesses from stream longer than 1k.
- Some benchmarks contain short streams.



Support longer stream to capture long term behavior.
Low overhead to support short streams.

Stream Characteristics – Control Flow

- 53% stream accesses from loop with control flow.



Outline

- Insight & Opportunities.
- Stream Characteristics.
- Stream ISA Extension.
- Stream-Aware Policies.
- Microarchitecture Extension.
- Evaluation.

Stream ISA Extension – Basic Example

Original C Code

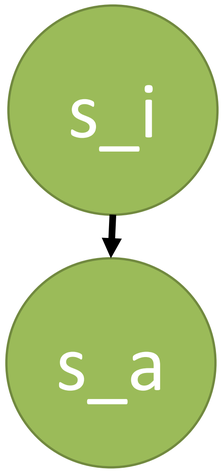
```
int i = 0;
while (i < N) {
  sum +=
  i++;
}
```

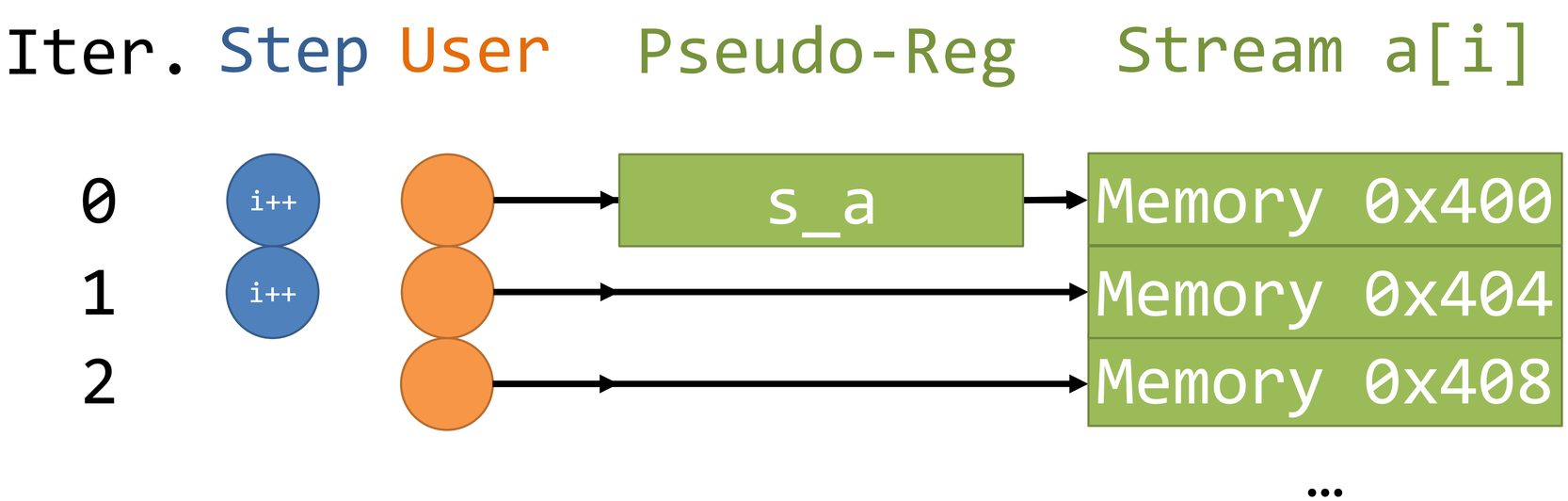
Stream Decoupled Pseudo Code

```
stream_cfg(s_i,
while (s_i < N) {

}
stream_end(s_i, s_a);
```

Stream Dependence Graph





Stream ISA Extension – Control Flow

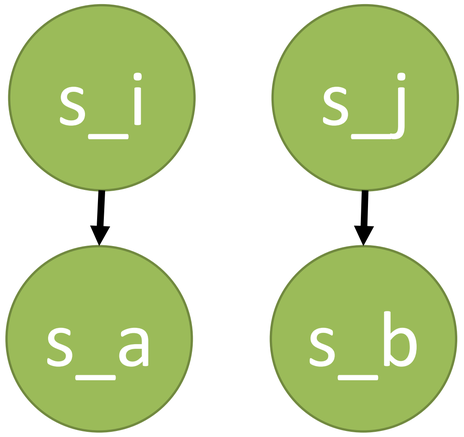
Original C Code

```
int i = 0, j = 0;
while (cond) {
  if (a[i] < b[j])
    [redacted]
  else
    [redacted]
}
```

Stream Decoupled Pseudo Code

```
stream_cfg(s_i, s_a, s_j, s_b);
while (cond) {
  if [redacted]
    [redacted]
  else
    [redacted]
}
stream_end(s_i, s_a, s_j, s_b);
```

Stream Dependence Graph



Iter. Step User Pseudo-Reg Stream a[i]

Stream ISA Extension – Indirect Stream

Original C Code

```
int i = 0;
while (i < N) {
  sum +=
  i++;
}
```

Stream Decoupled Pseudo Code

```
stream_cfg(s_i, s_a, s_b);
while (s_i < N) {

}
stream_end(s_i, s_a, s_b);
```

Stream Dependence Graph

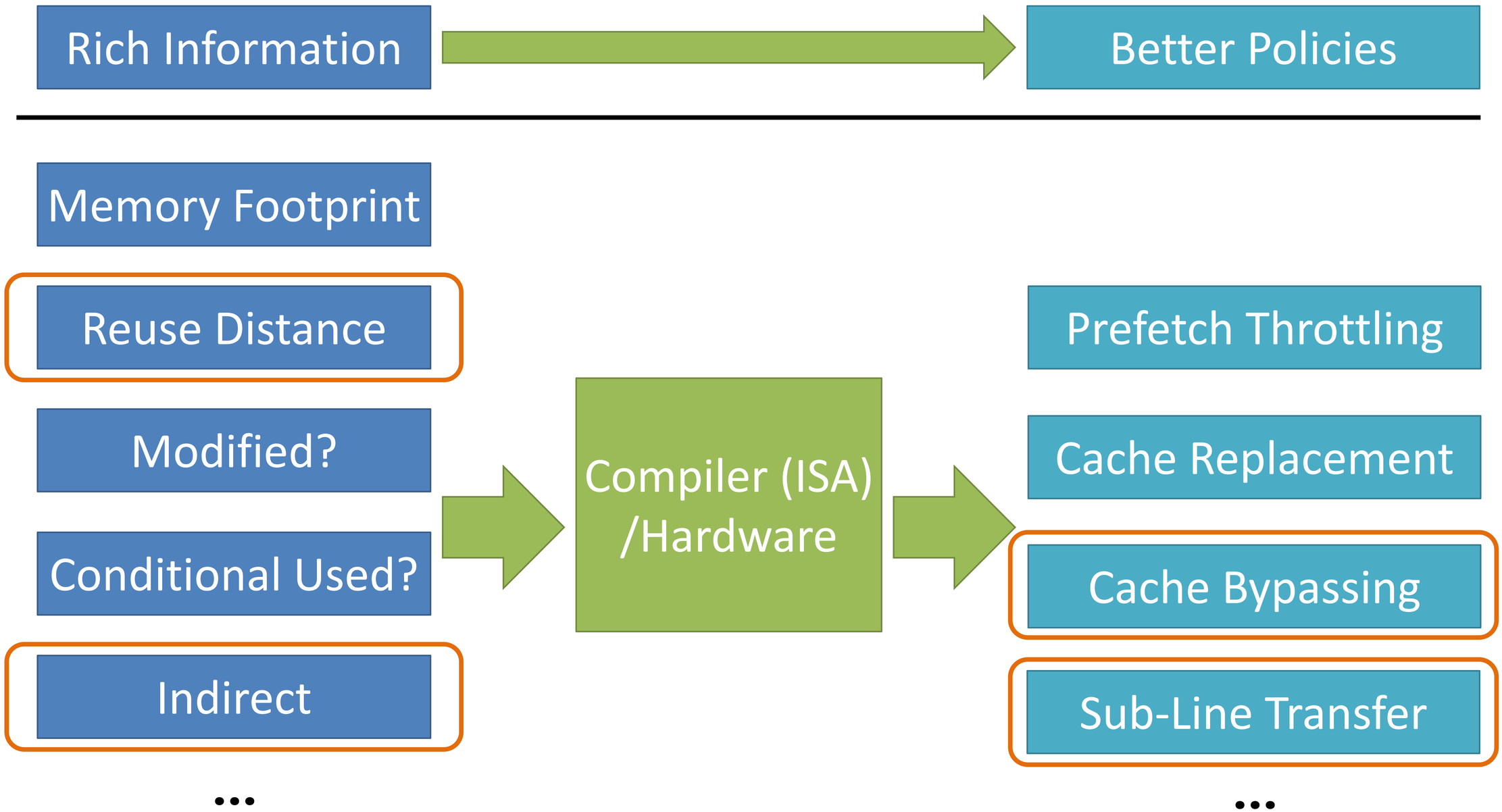
Stream ISA Extension – ISA Semantic

- New architectural states:
  - Stream configuration.
  - Current iteration's data.
- New speculation in ISA:
  - Stream elements will be used.
  - Streams are long.
- Maintain the memory order.
  - Load → first use of the pseudo-register after configured/stepped.
  - Store → every write to the pseudo-register.

Outline

- Insight & Opportunities.
- Stream Characteristics.
- Stream ISA Extension.
- Stream-Aware Policies.
- Microarchitecture Extension.
- Evaluation.

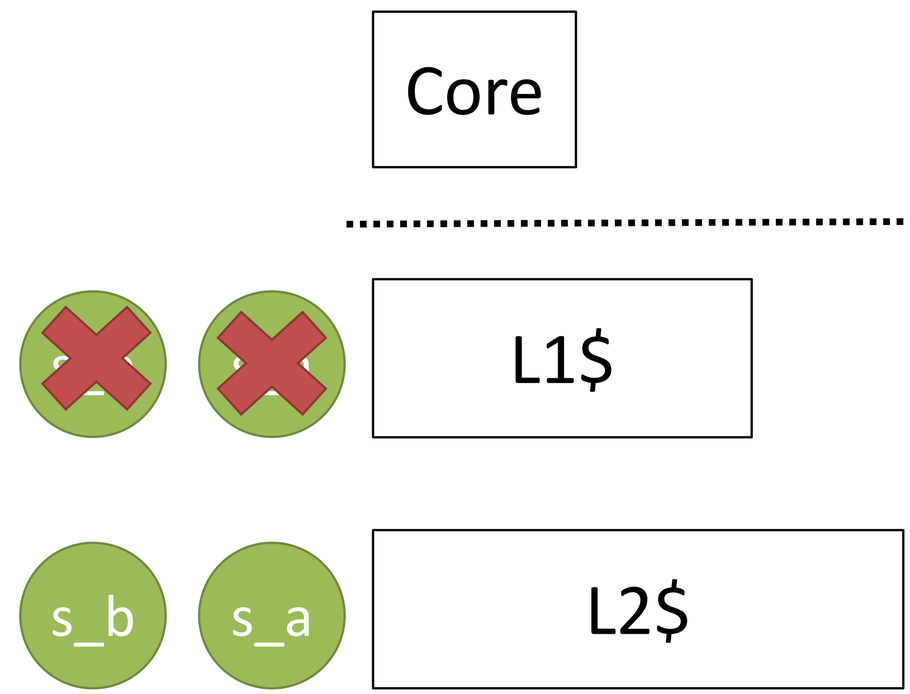
Stream-Aware Policies



Stream-Aware Policies – Cache Bypass

- Stream: Access Pattern → Precise Memory Footprint.

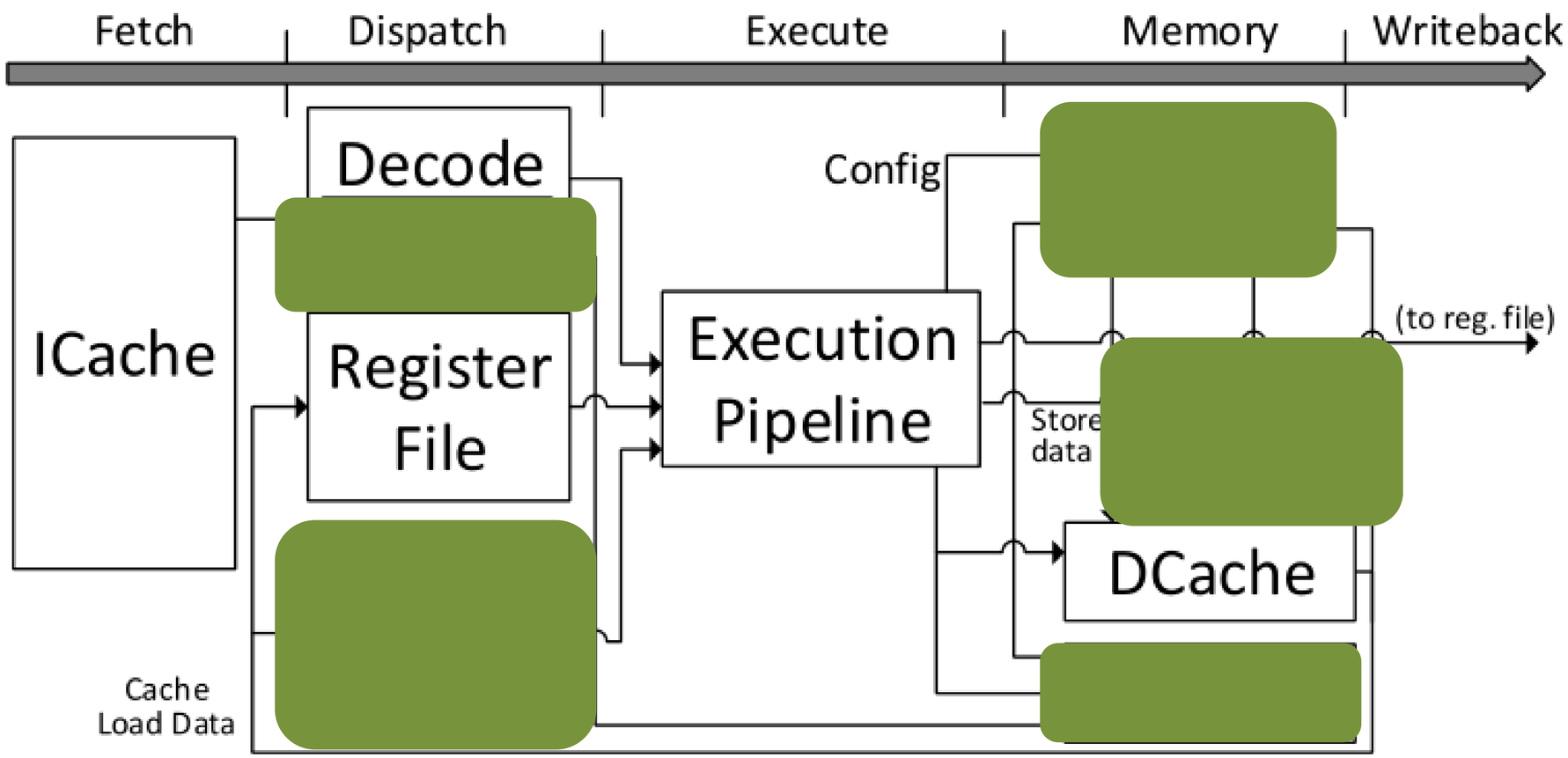
```
while (i < N)
  while (j < N)
    while (k < N)
      sum +=
            *
```

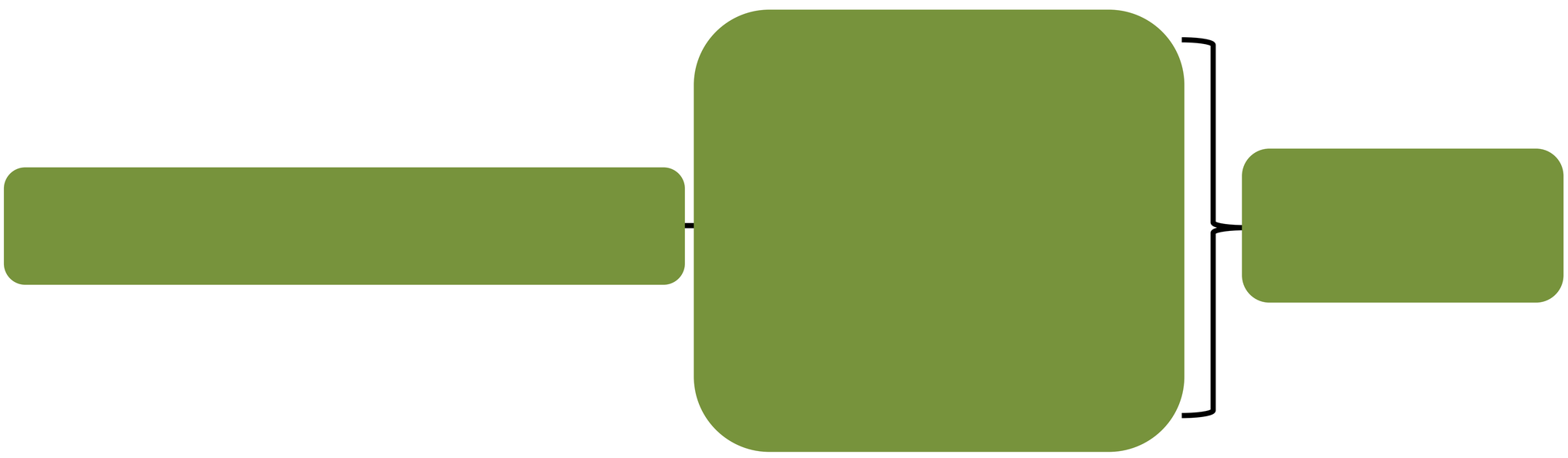


Outline

- Insight & Opportunities.
- Stream Characteristics.
- Stream ISA Extension.
- Stream-Aware Policies.
- Microarchitecture Extension.
- Evaluation.

Microarchitecture





Microarchitecture – Misspeculation

- Control misspeculated stream_step.
  - Decrement the iteration map.
  - No need to flush the FIFO and re-fetch data (decoupled) !
- Other misspeculation.
  - Revert the stream states, including stream FIFO.
- Memory fault delayed until the use of the element.

Outline

- Insight & Opportunities.
- Stream Characteristics.
- Stream ISA Extension.
- Microarchitecture Extension.
- Stream-Aware Policies.
- Evaluation.

Methodology

- Compiler in LLVM:
  - Identify stream candidates.
  - Generate stream configuration.
  - Transform the program.
- Gem5 + McPAT simulation.
- 33 Benchmarks:
  - SPEC2017 C/CPP benchmarks.
  - CortexSuite.
- SimPoint:
  - 10 million instructions' simpoints.
  - ~10 simpoints per benchmark.

CPU	2.0GHz 8-Way OoO Cores 8-wide fetch/issue/commit 64 IQ, 32 LQ, 32 SQ, 192 ROB 256 Int RF, 256 FP RF speculative scheduling
Function Units	6 Int ALU (1 cycle) 2 Int Mult/Div (3/20 cycles) 4 FP ALU (2 cycles) 2 FP Mult/Div (4/12 cycles) 4 SIMD (1 cycle)
Private L1 ICache	32KB / 8-way 8 MSHRs / 2-cycle latency
Private L1 DCache	32KB / 8-way 8 MSHRs / 2-cycle latency
Private L2 Cache	256KB / 16-way 16 MSHRs / 15-cycle latency
To L3 Bus	16-byte width
Shared L3 Cache	8MB / 8-way 20 MSHRs / 20-cycle latency
DRAM	2 channel / 1600MHz DDR3 12.8 GB/s

Configurations

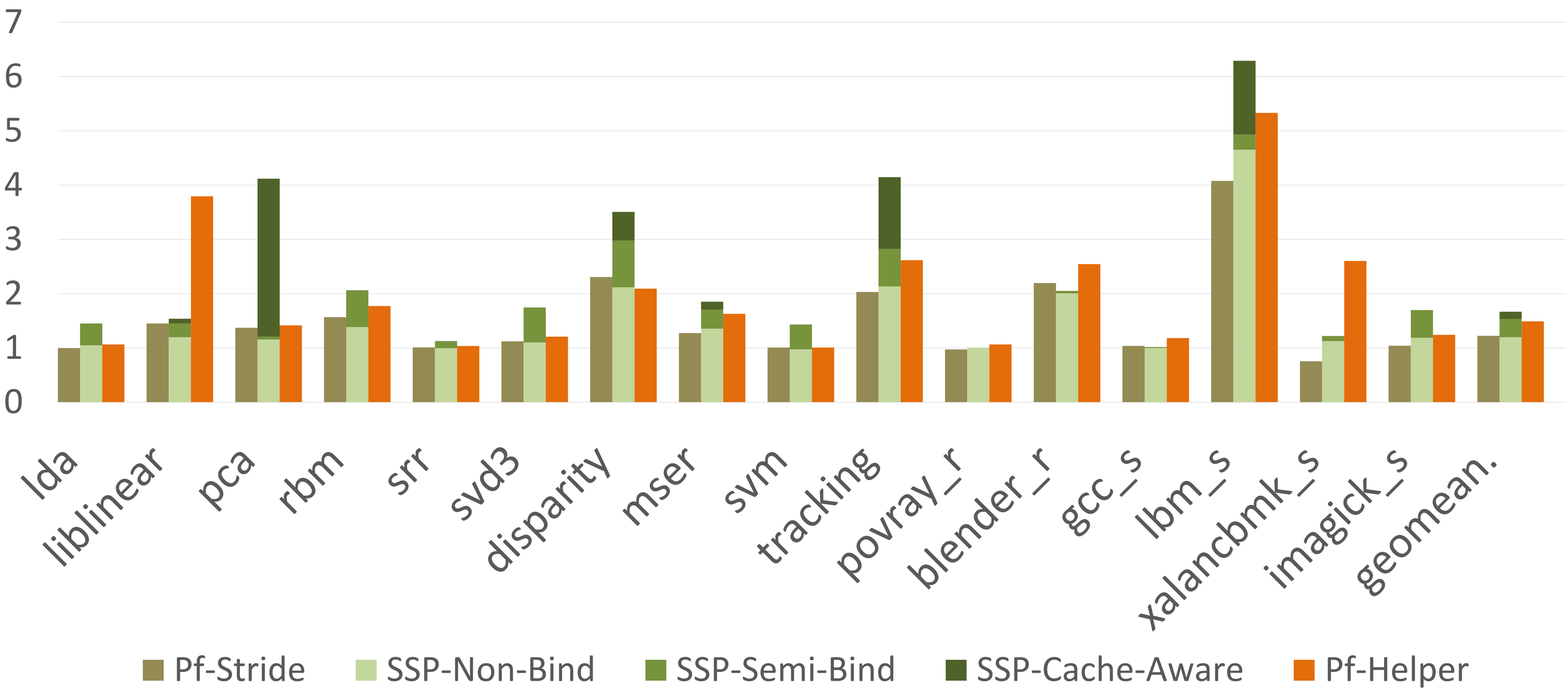
Baseline.

- Baseline O3.
- Pf-Stride:
  - Table-based prefetcher.
- Pf-Helper:
  - SMT-based ideal helper thread.
  - Requires no HW resources (ROB, etc.).
  - Exactly 1k instruction before the main thread.

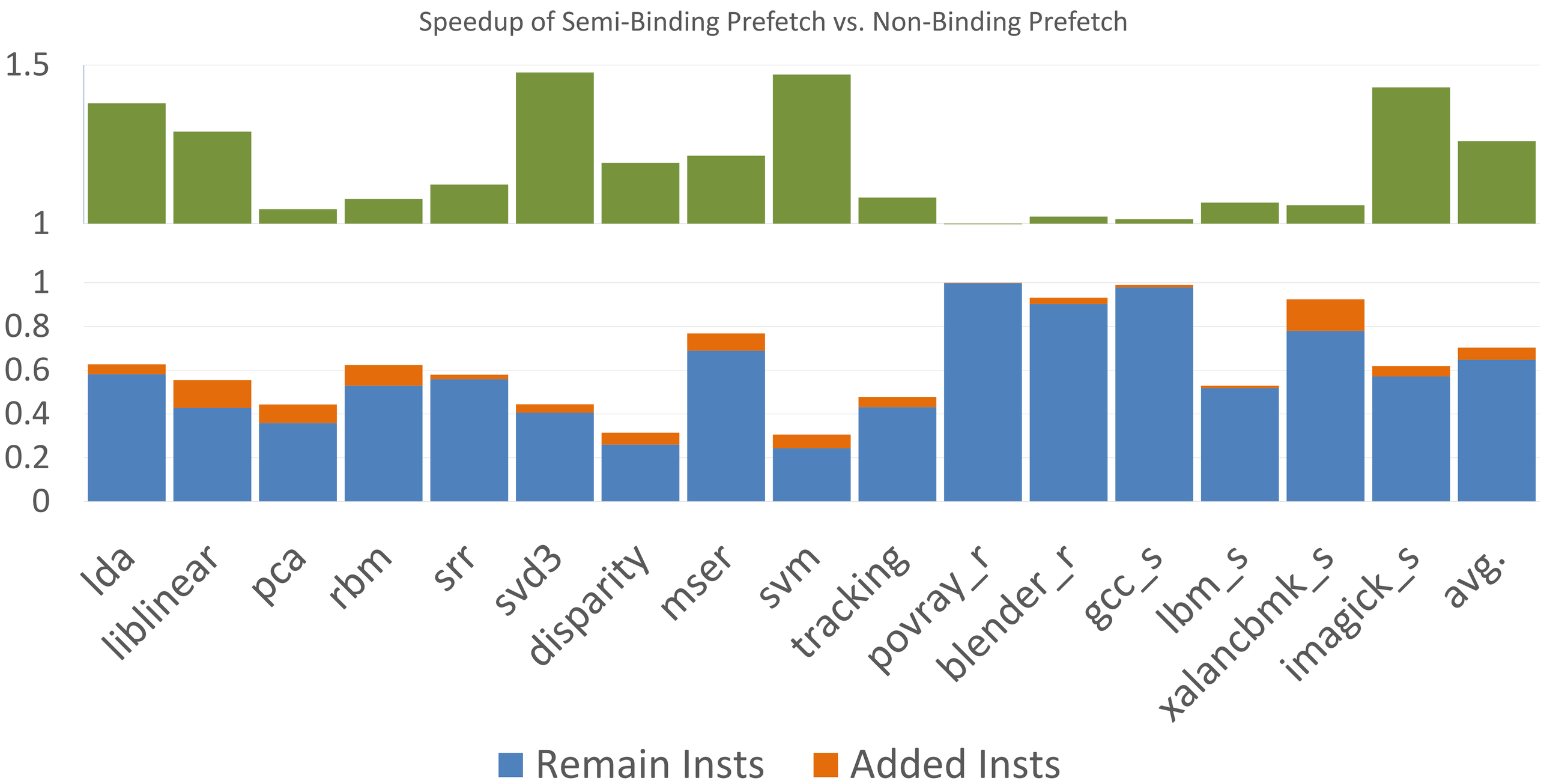
Stream Specialized Processor.

- SSP-Non-Bind:
  - Prefetch only.
- SSP-Semi-Bind:
  - + Semi-binding prefetch.
- SSP-Cache-Aware:
  - + Stream-Aware cache bypassing.

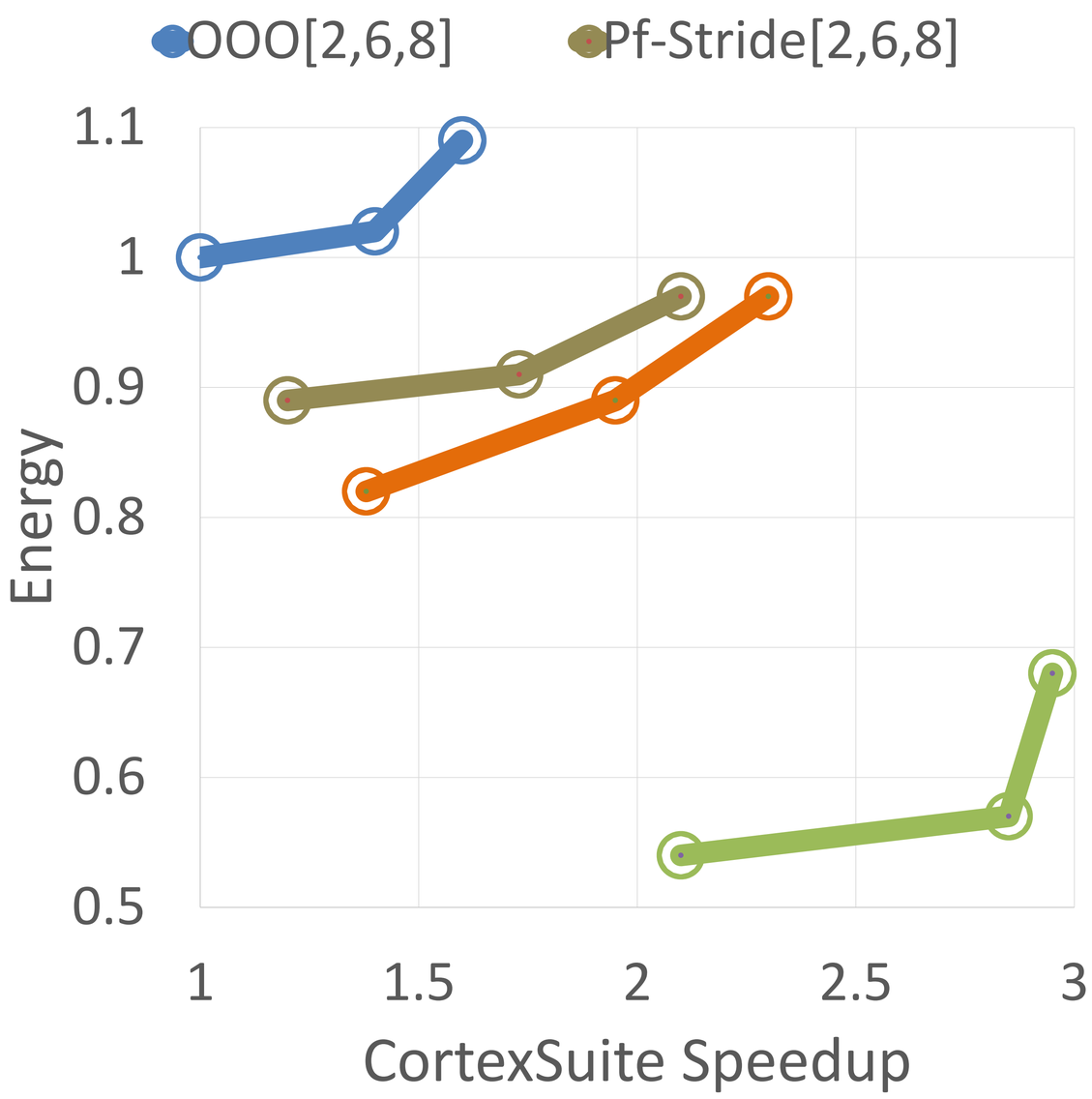
Results – Overall Performance

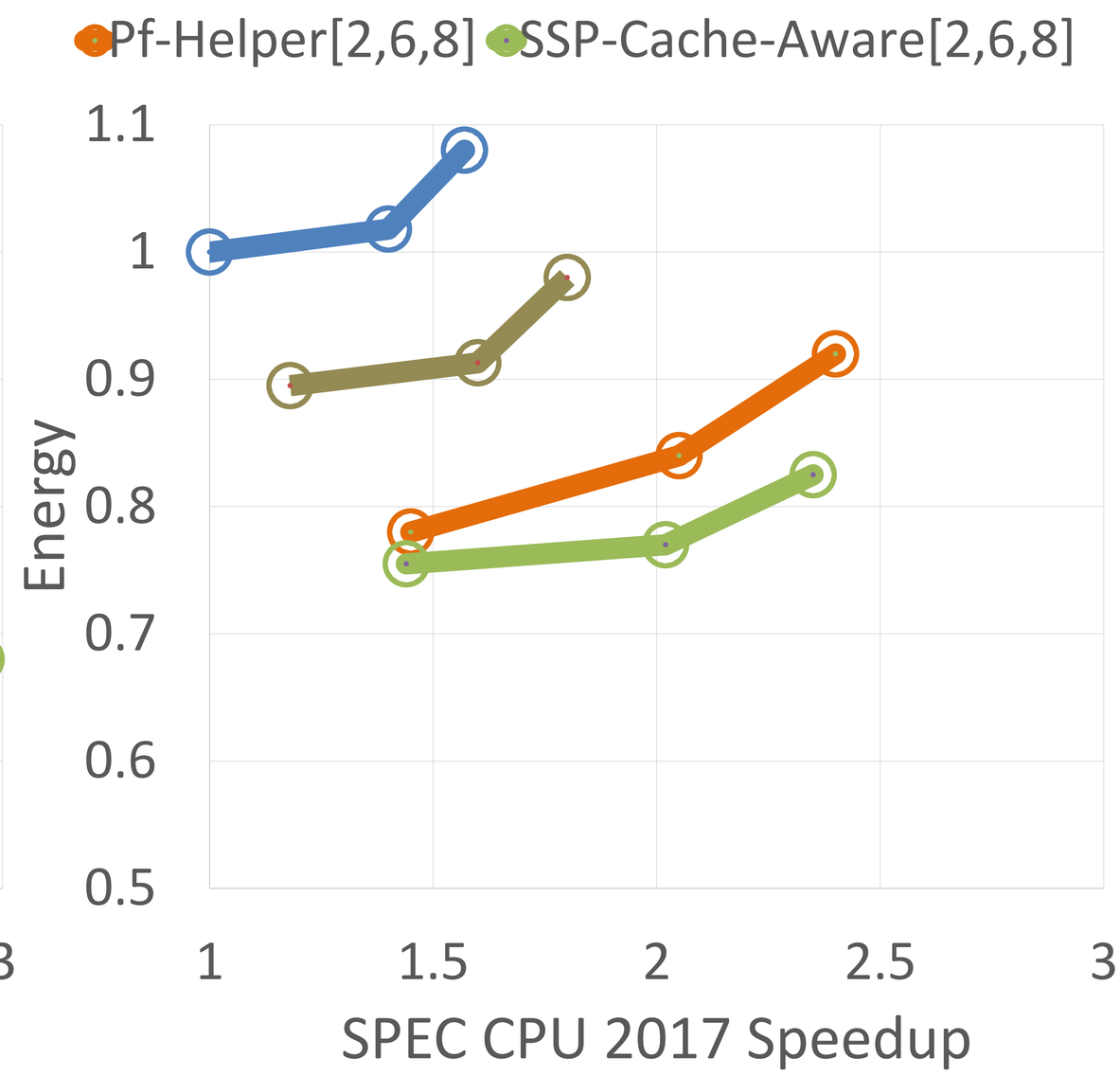


Results – Semi-Binding Prefetching



Results – Design Space Interaction





Conclusion

- Stream as a new memory abstraction in ISA.
  - ISA/Microarchitecture extension.
  - Stream-aware cache bypassing.
- New paradigm of memory specialization.
  - New direction for improving cache architectures.
  - Combine memory and computation specialization.