Aligned Scheduling: Cache-efficient Instruction Scheduling for VLIW Processors

Vasileios Porpodas [†] and Marcelo Cintra [†⋆]

School of Informatics, University of Edinburgh[†]
Intel Labs Braunschweig[⋆]

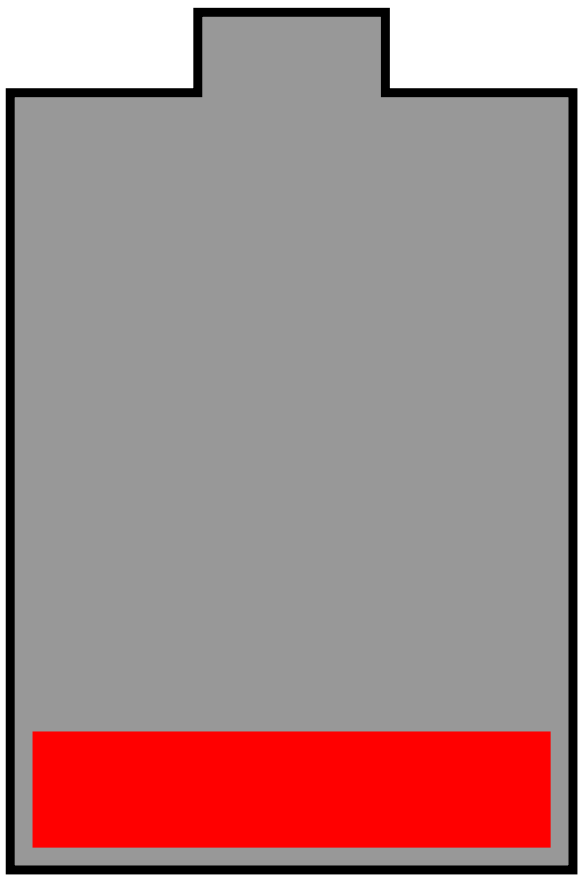
LCPC 2013

Energy efficiency, VLIWs and Scheduling

- Mobile era

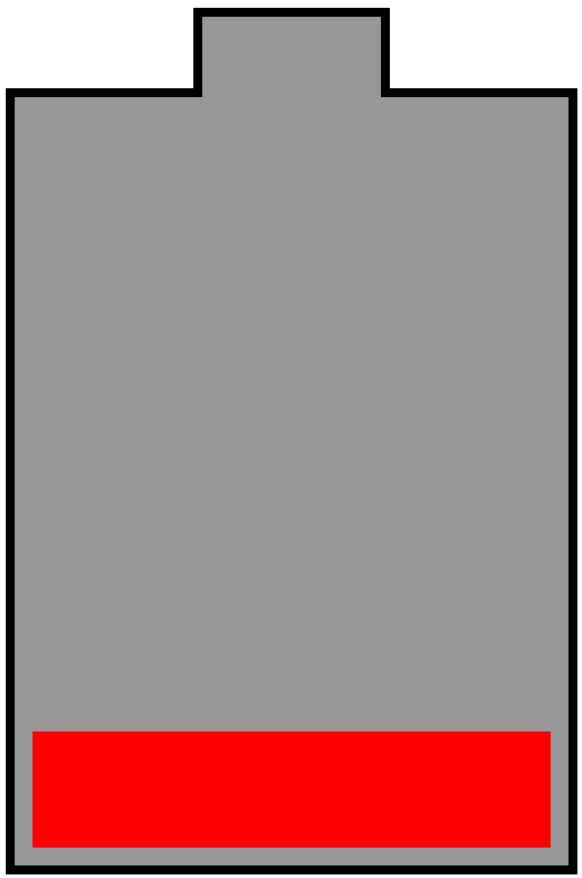
Energy efficiency, VLIWs and Scheduling

- Mobile era
- Energy becomes a major design constraint



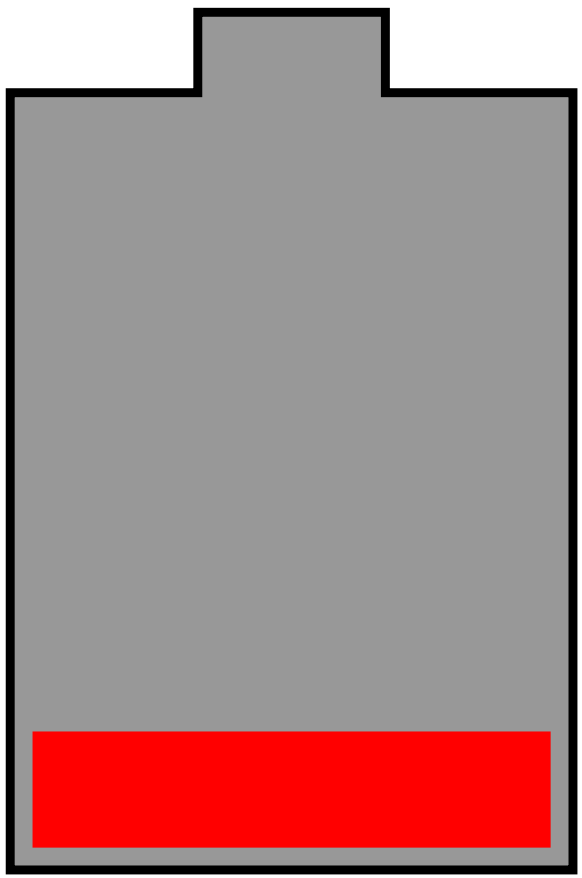
Energy efficiency, VLIWs and Scheduling

- Mobile era
- Energy becomes a major design constraint
- Hardware Instruction scheduling consumes a lot of energy



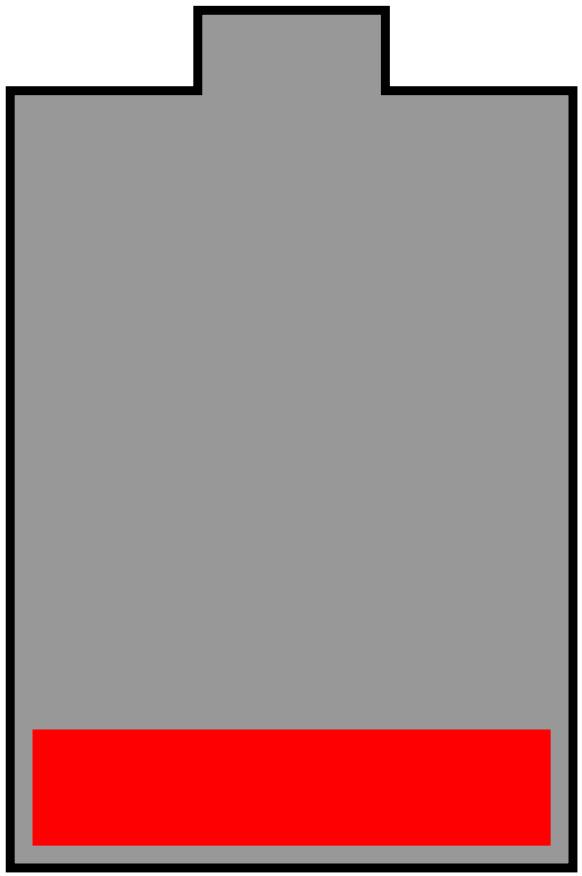
Energy efficiency, VLIWs and Scheduling

- Mobile era
- Energy becomes a major design constraint
- Hardware Instruction scheduling consumes a lot of energy
- VLIW processors: high-performance and statically scheduled



Energy efficiency, VLIWs and Scheduling

- Mobile era
- Energy becomes a major design constraint
- Hardware Instruction scheduling consumes a lot of energy
- VLIW processors: high-performance and statically scheduled
- Suffer from unpredictable cache latencies



Hardware Load-Use interlocking (1/3)

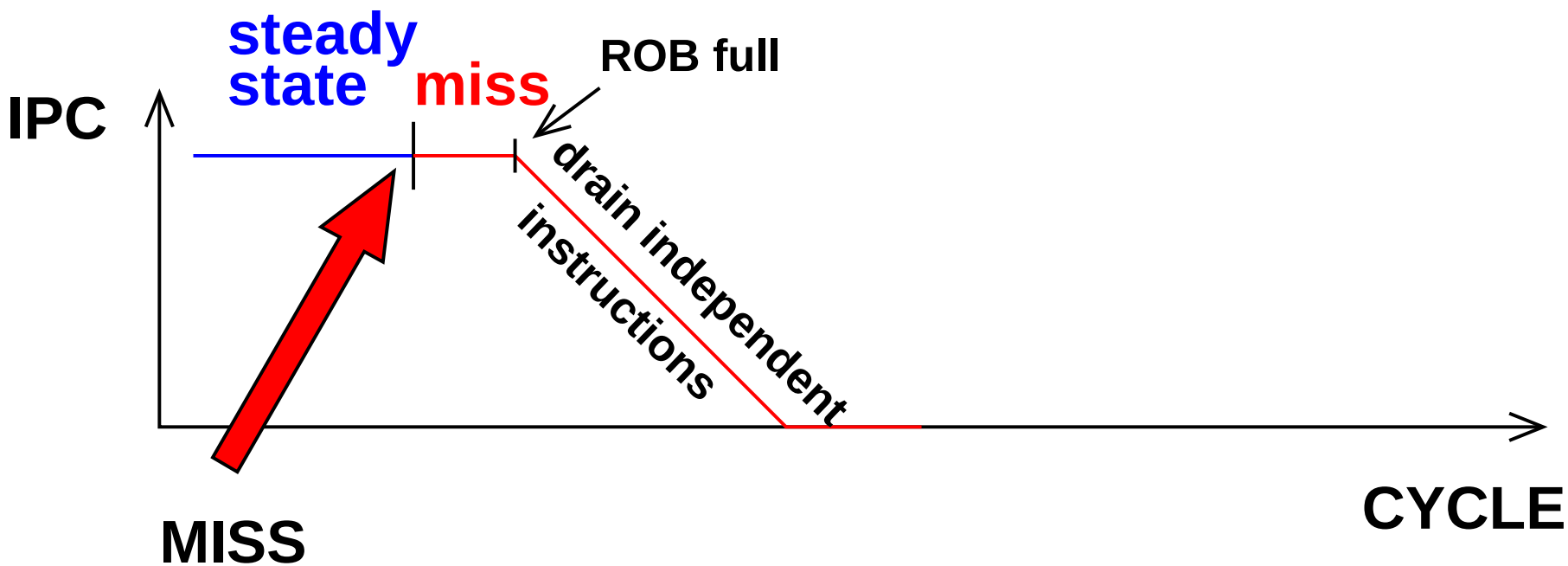
- Out-Of-Order Processors support full scoreboarding
- Performance model (Karkhanis & Smith ISCA'04)

Hardware Load-Use interlocking (1/3)

- Out-Of-Order Processors support full scoreboarding
- Performance model (Karkhanis & Smith ISCA'04)

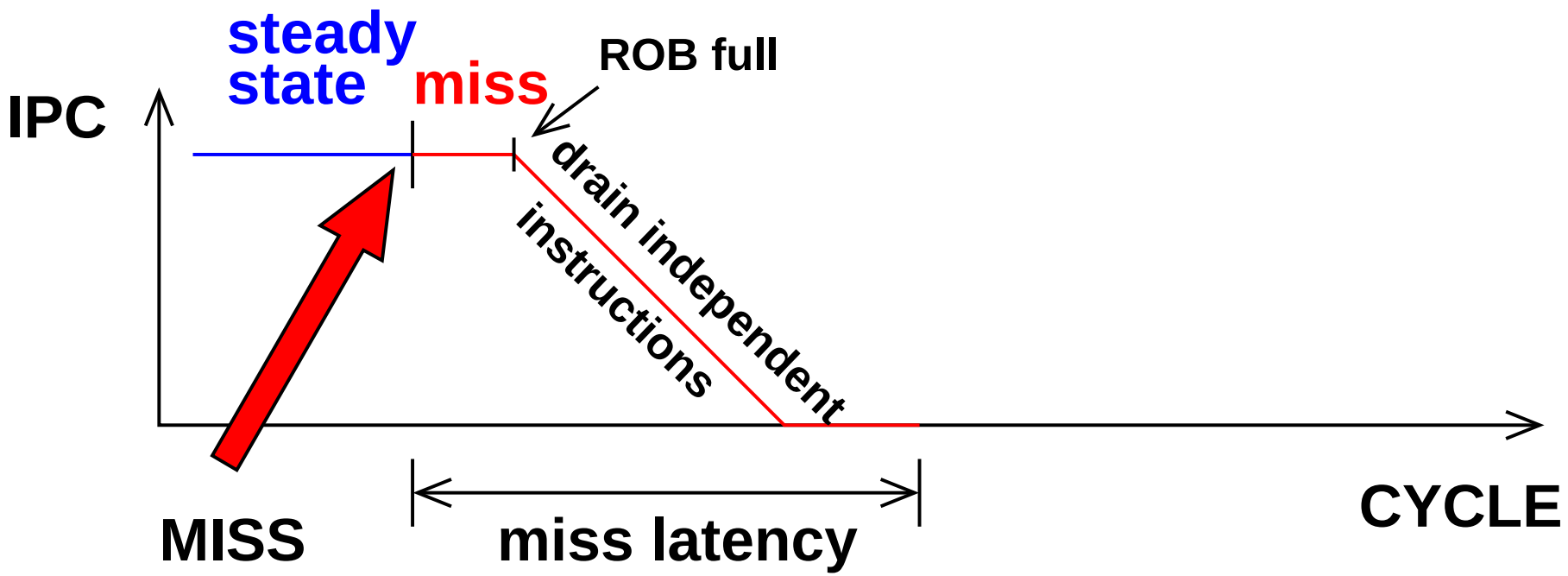
Hardware Load-Use interlocking (1/3)

- Out-Of-Order Processors support full scoreboarding
- Performance model (Karkhanis & Smith ISCA'04)



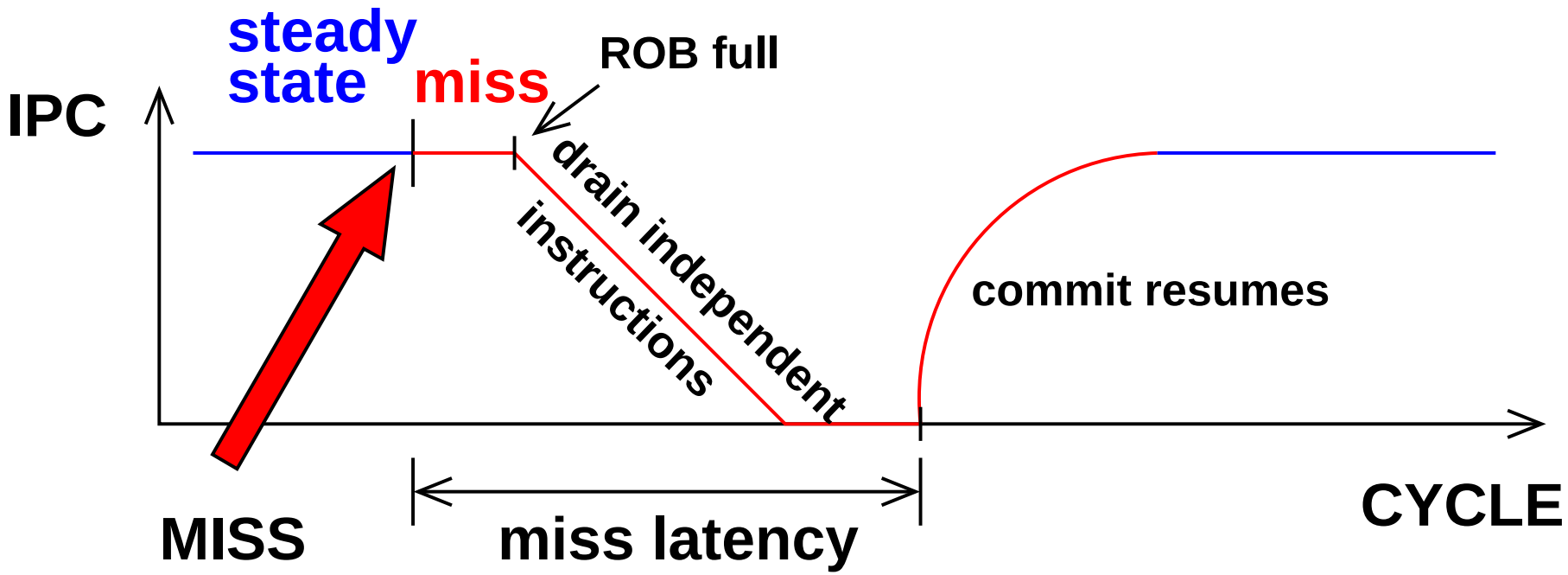
Hardware Load-Use interlocking (1/3)

- Out-Of-Order Processors support full scoreboarding
- Performance model (Karkhanis & Smith ISCA'04)



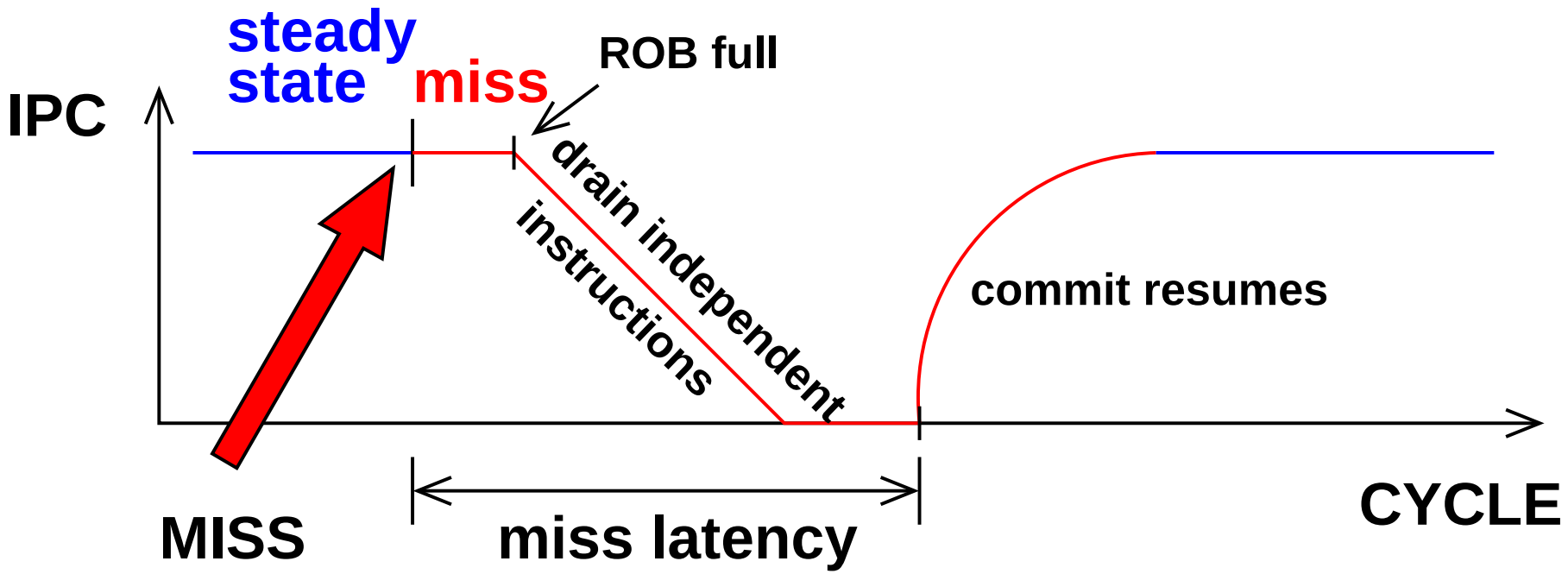
Hardware Load-Use interlocking (1/3)

- Out-Of-Order Processors support full scoreboarding
- Performance model (Karkhanis & Smith ISCA'04)



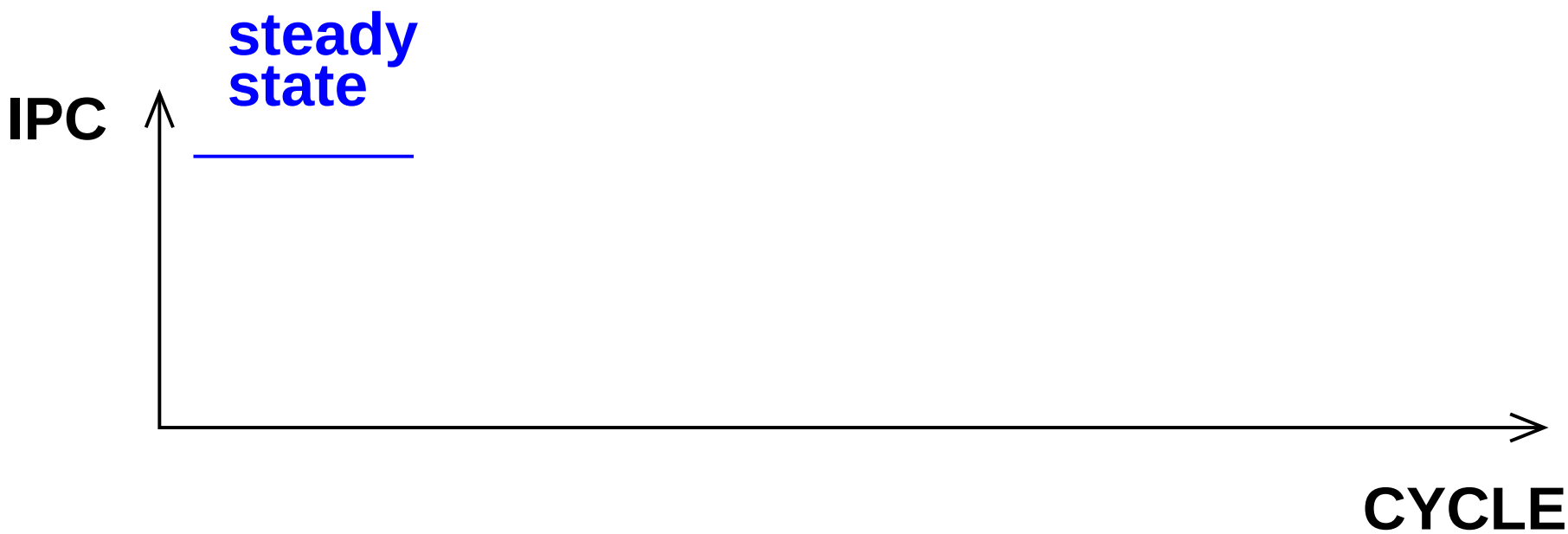
Hardware Load-Use interlocking (1/3)

- Out-Of-Order Processors support full scoreboarding
- Performance model (Karkhanis & Smith ISCA'04)
- Most effective at hiding latency - Expensive hardware



Hardware Load-Use interlocking (2/3)

- VLIW Processors with VLIW-word-level scoreboarding (Stall-On-Use)

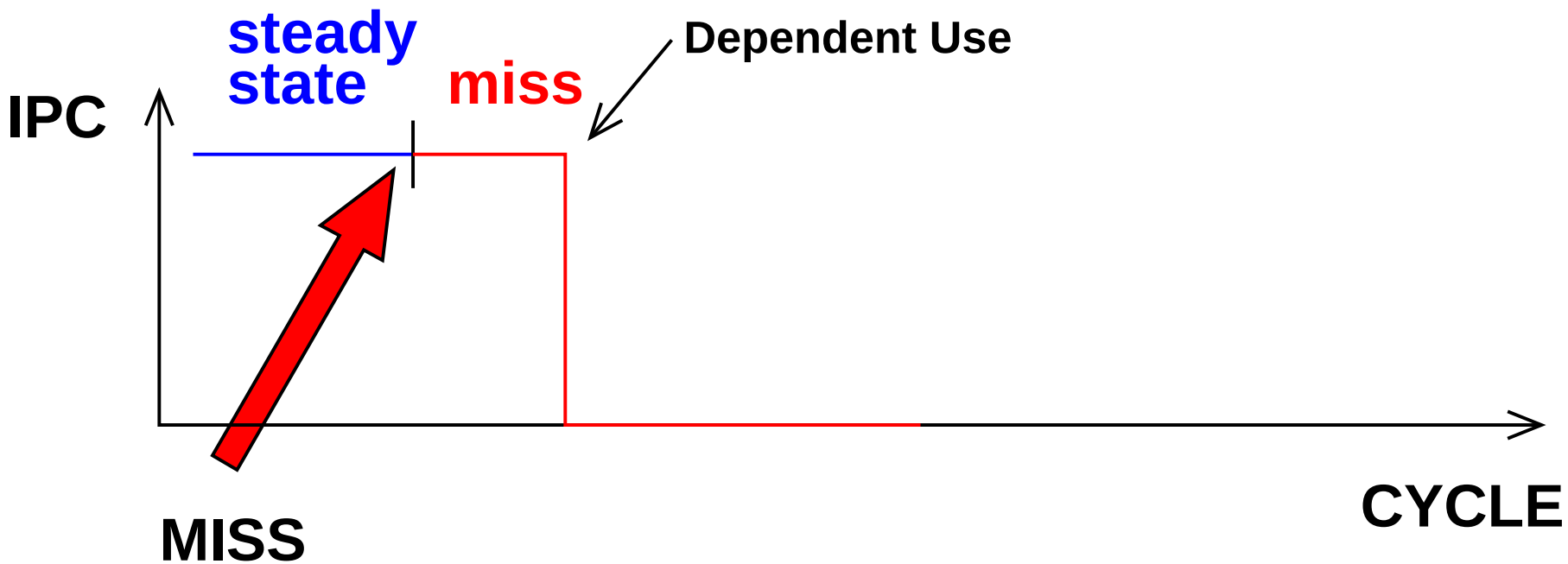


Hardware Load-Use interlocking (2/3)

- VLIW Processors with VLIW-word-level scoreboarding (Stall-On-Use)

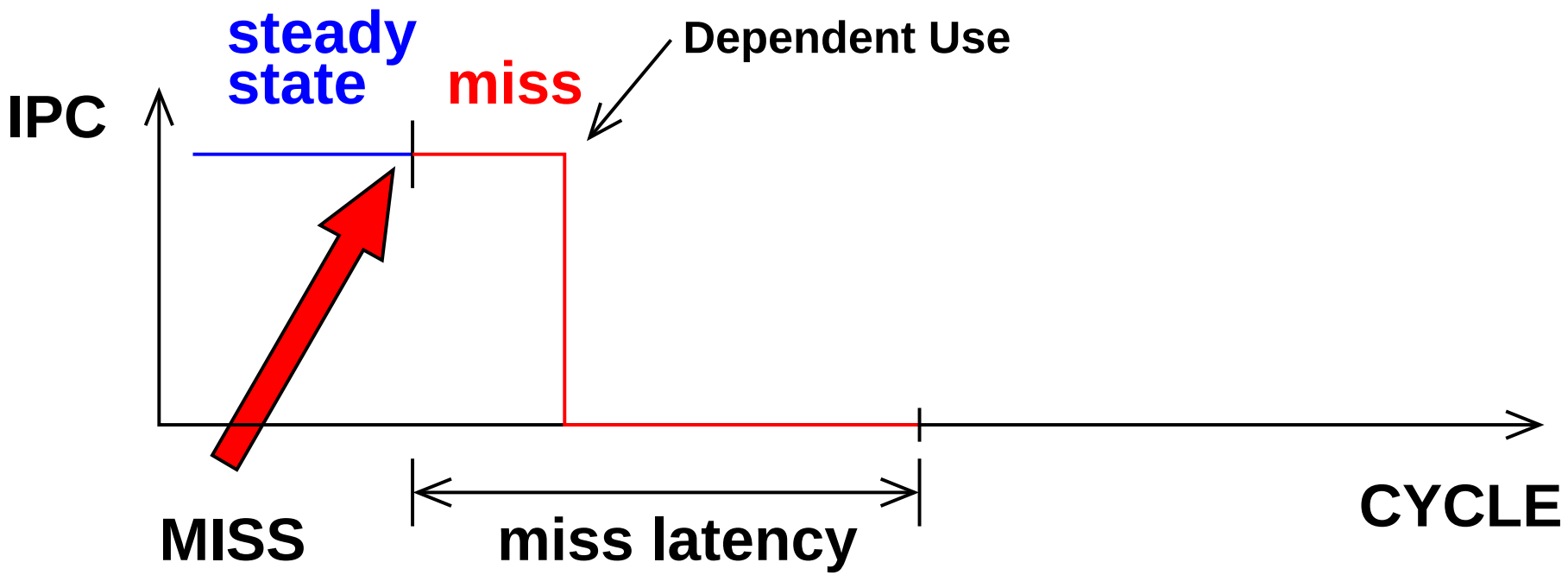
Hardware Load-Use interlocking (2/3)

- VLIW Processors with VLIW-word-level scoreboarding (Stall-On-Use)



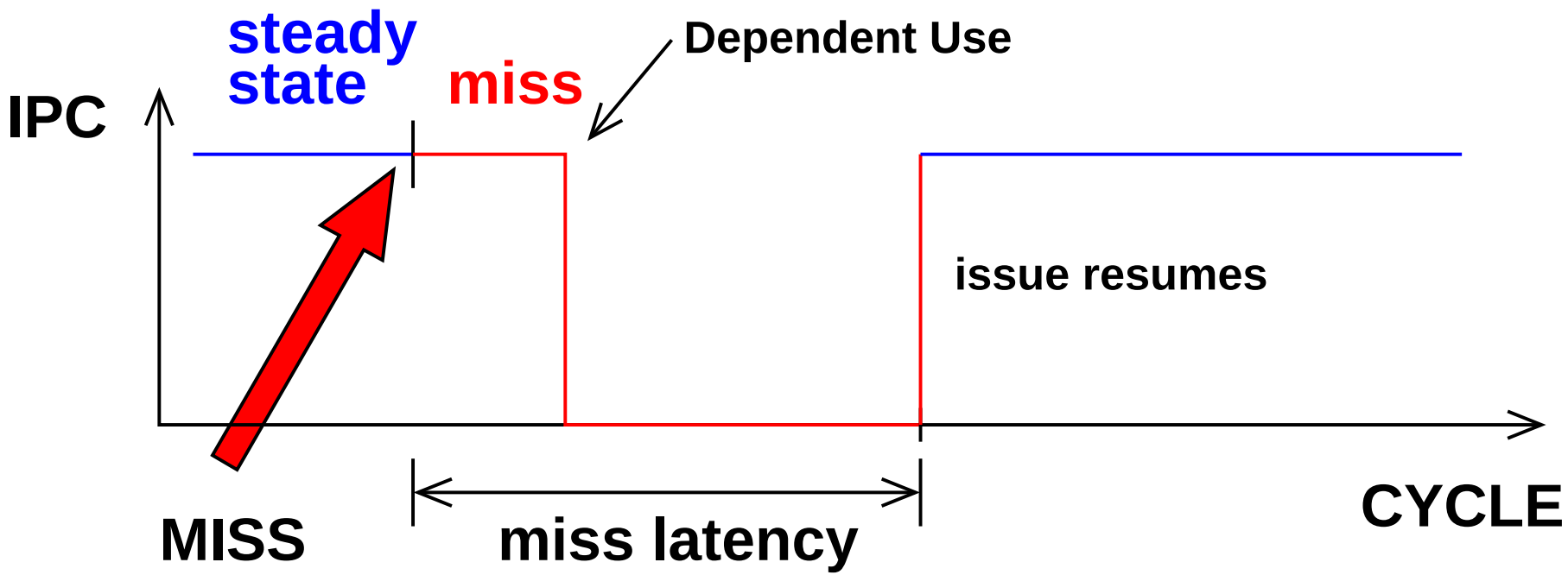
Hardware Load-Use interlocking (2/3)

- VLIW Processors with VLIW-word-level scoreboarding (Stall-On-Use)



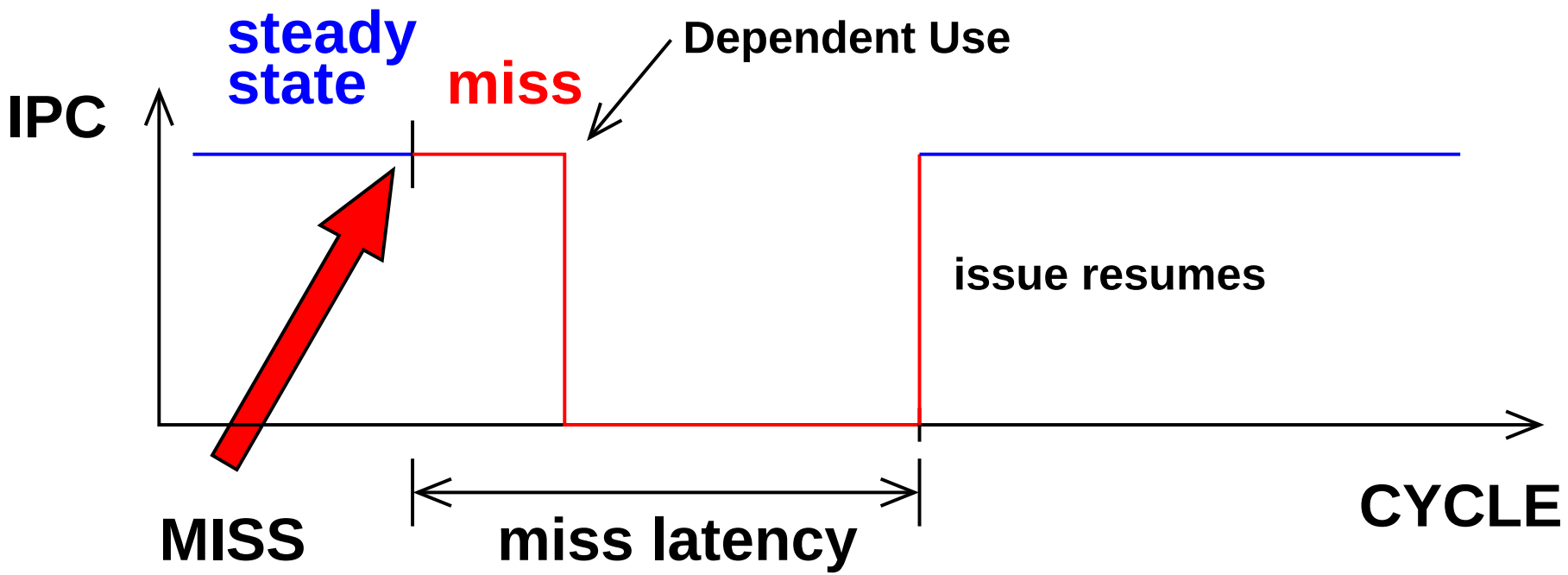
Hardware Load-Use interlocking (2/3)

- VLIW Processors with VLIW-word-level scoreboarding (Stall-On-Use)



Hardware Load-Use interlocking (2/3)

- VLIW Processors with VLIW-word-level scoreboarding (Stall-On-Use)
- Effective at hiding some latency



Hardware Load-Use interlocking (3/3)

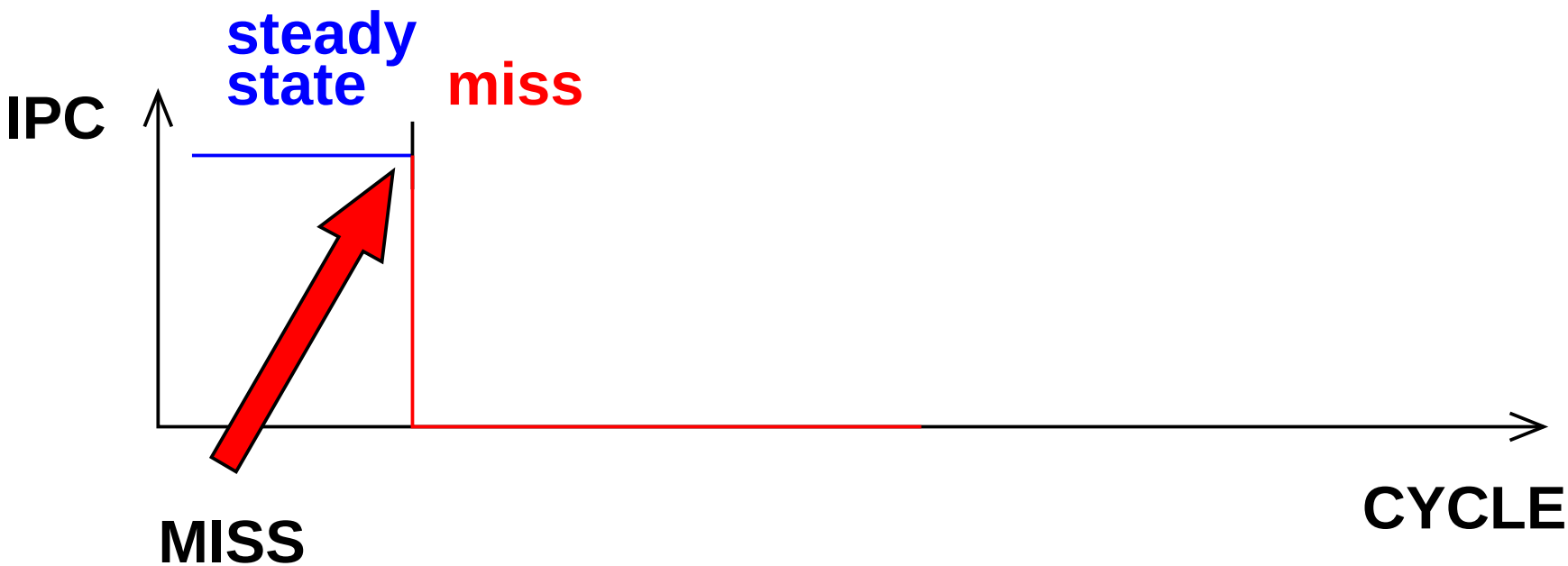
- VLIW Processors with NO Load-Use Interlocking (Stall-On-Miss)

Hardware Load-Use interlocking (3/3)

- VLIW Processors with NO Load-Use Interlocking (Stall-On-Miss)

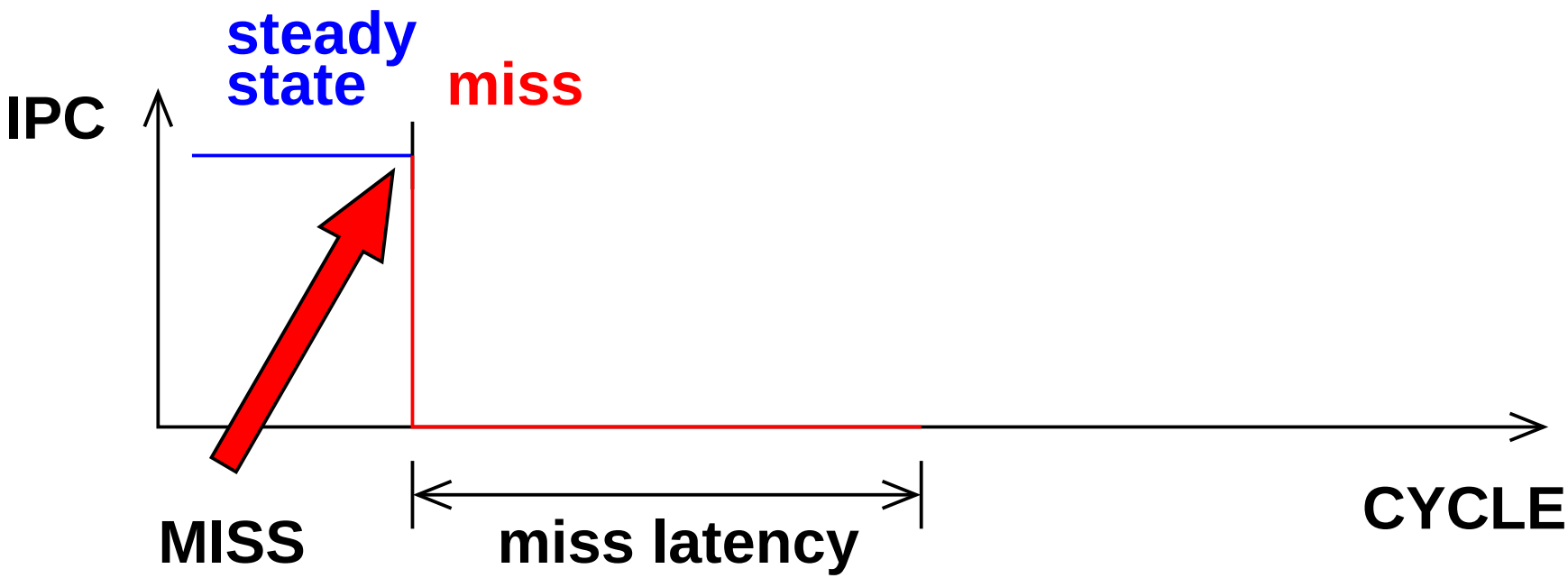
Hardware Load-Use interlocking (3/3)

- VLIW Processors with NO Load-Use Interlocking (Stall-On-Miss)



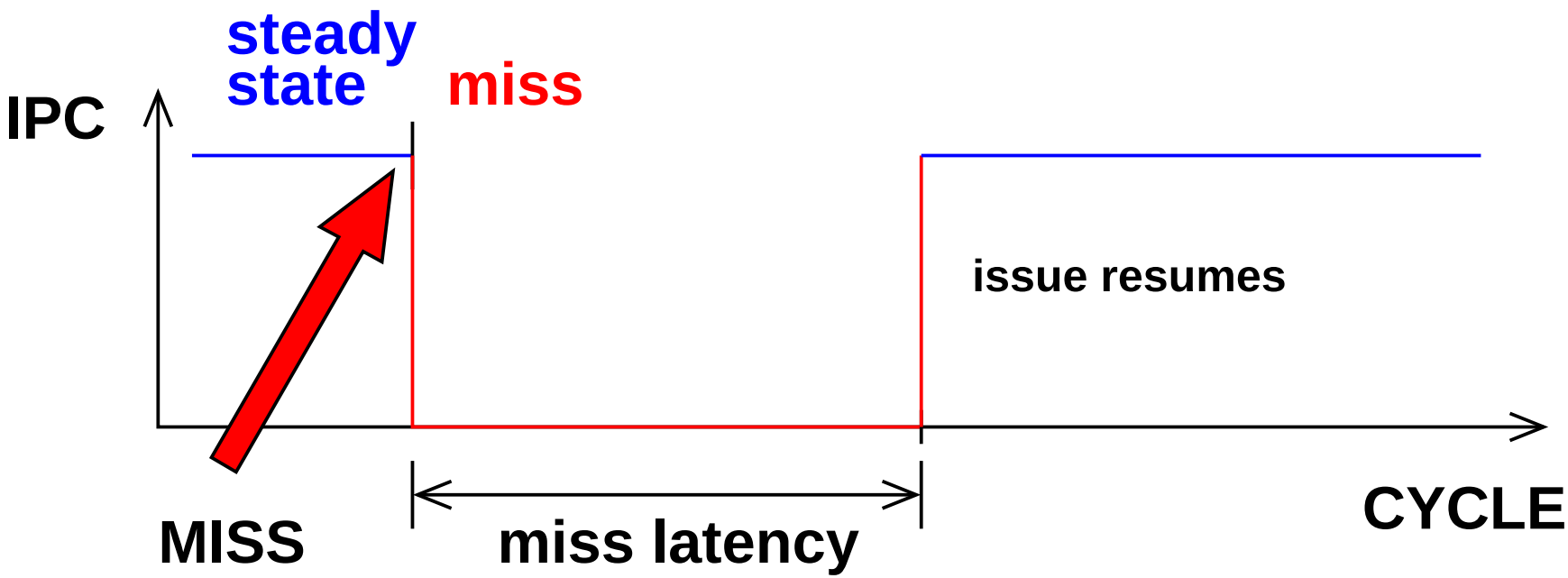
Hardware Load-Use interlocking (3/3)

- VLIW Processors with NO Load-Use Interlocking (Stall-On-Miss)



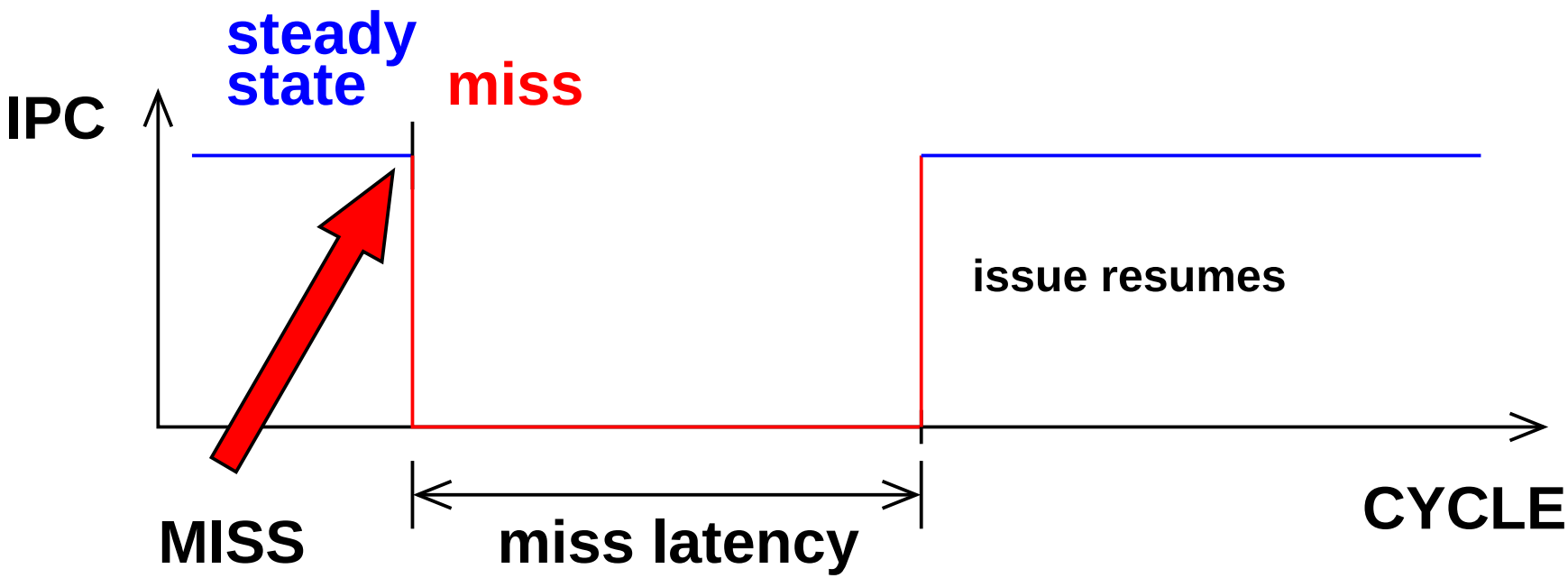
Hardware Load-Use interlocking (3/3)

- VLIW Processors with NO Load-Use Interlocking (Stall-On-Miss)



Hardware Load-Use interlocking (3/3)

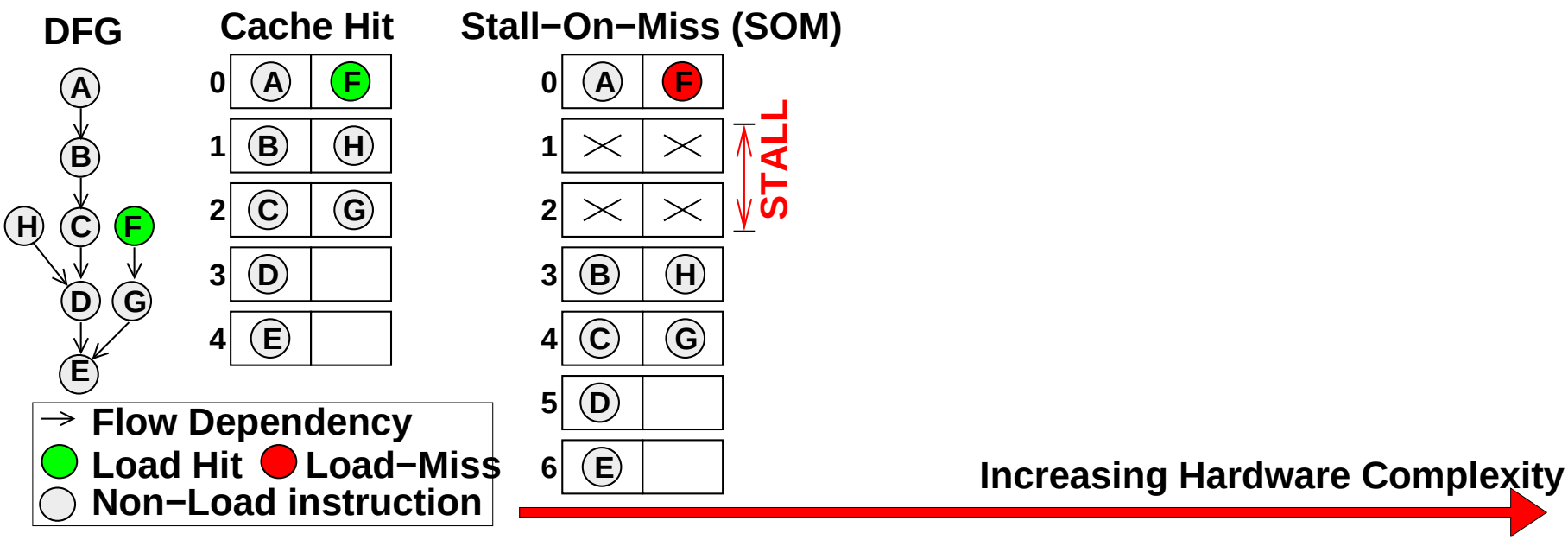
- VLIW Processors with NO Load-Use Interlocking (Stall-On-Miss)
- Poor performance under lots of misses



Levels of Load-Use interlocking

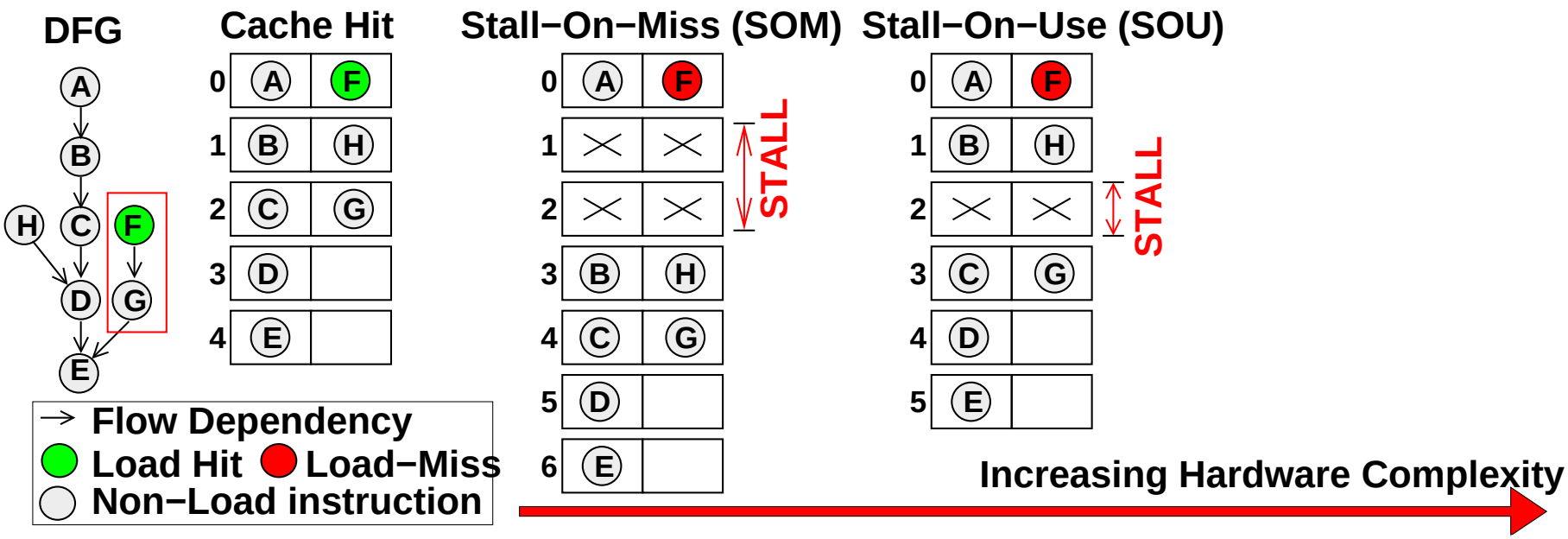
Levels of Load-Use interlocking

- No interlocks (Stall-On-Miss SOM)



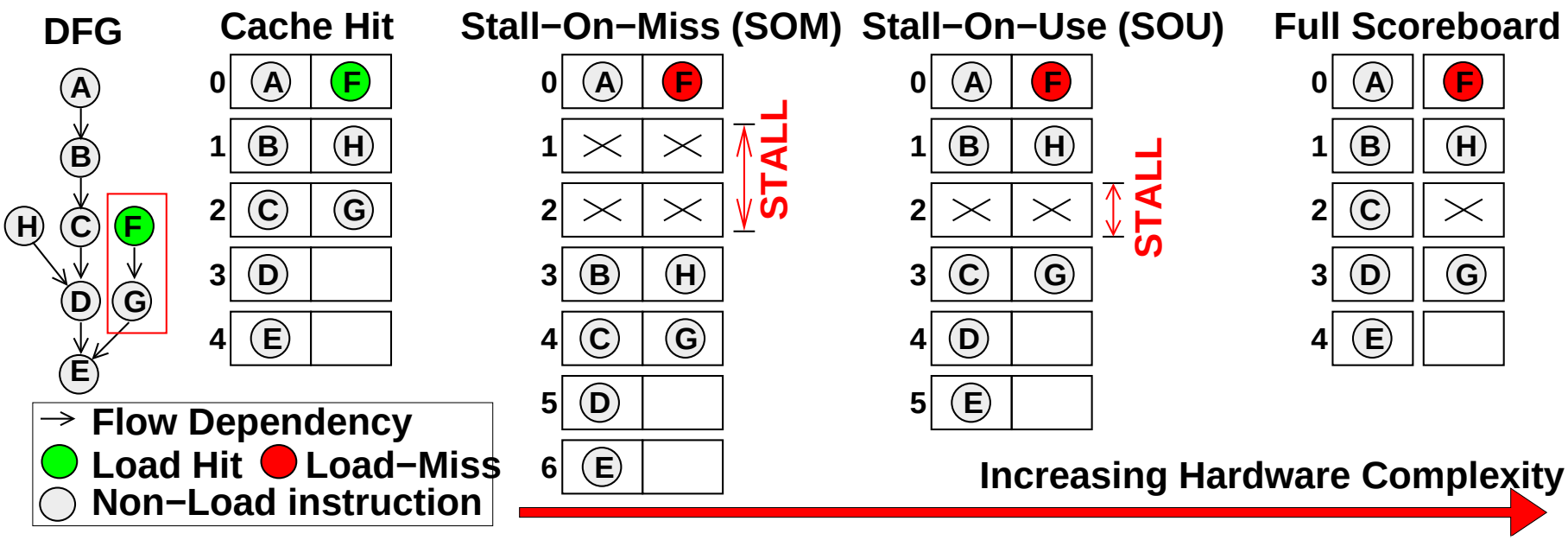
Levels of Load-Use interlocking

- No interlocks (Stall-On-Miss SOM)
- VLIW-level interlocks (Stall-On-Use SOU)



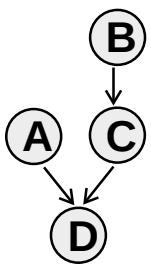
Levels of Load-Use interlocking

- No interlocks (Stall-On-Miss SOM)
- VLIW-level interlocks (Stall-On-Use SOU)
- Instruction-level Full Scoreboarding



Observation: Load Semantics on SOM

- Missing Loads have different semantics from Long-Latency instructions

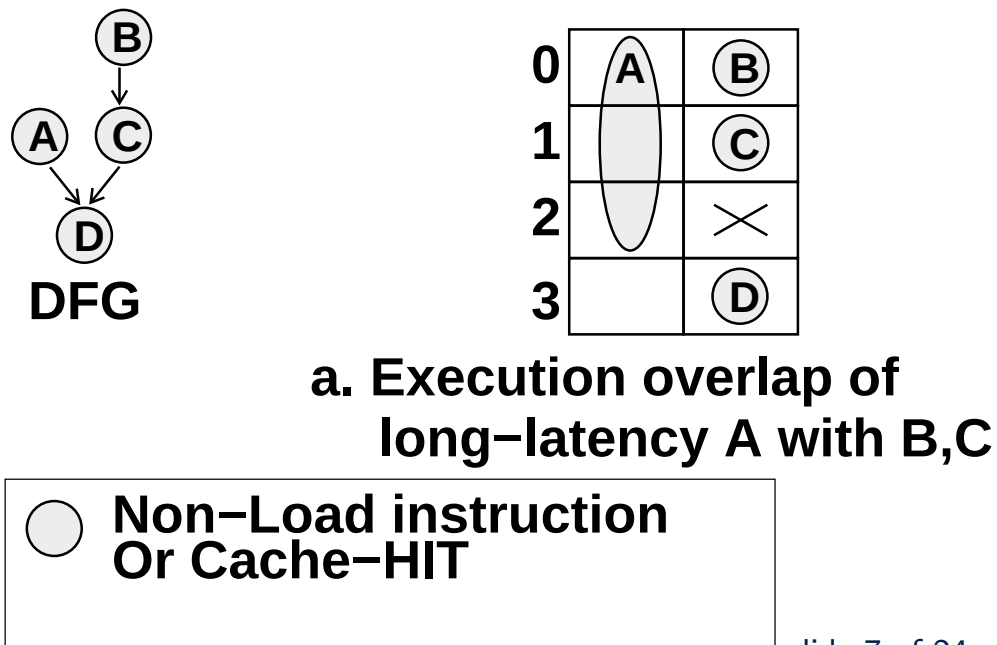


DFG

Non−Load instruction
Or Cache−HIT

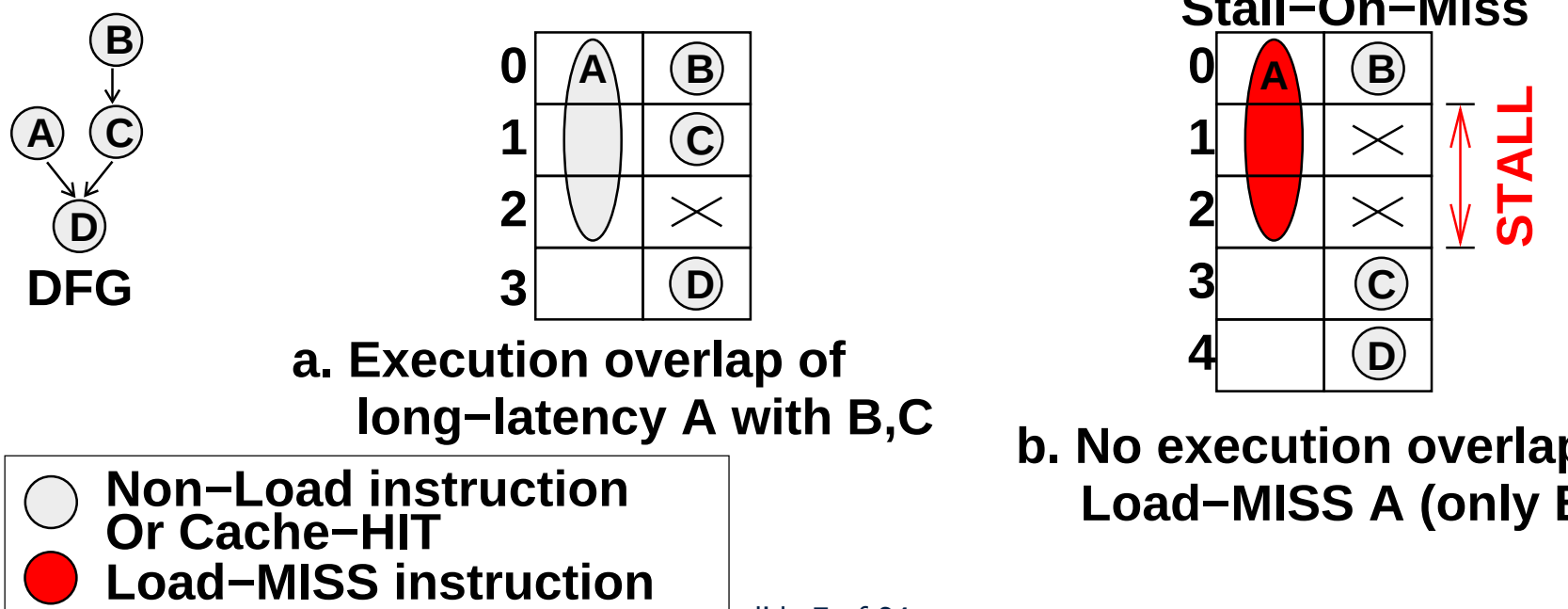
Observation: Load Semantics on SOM

- Missing Loads have different semantics from Long-Latency instructions
- Existing schedulers treat them as equivalent



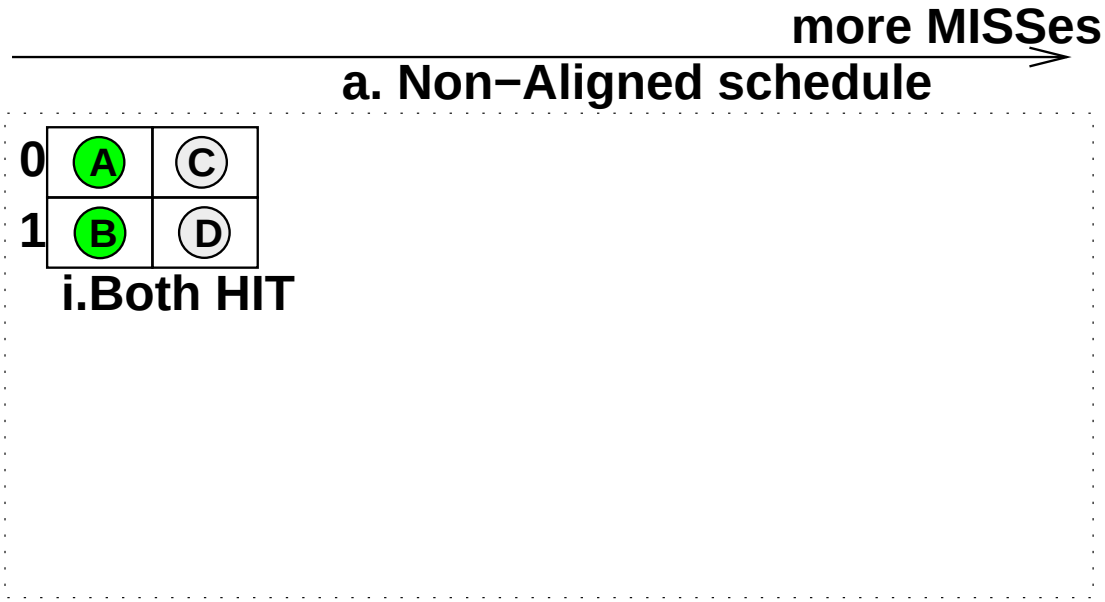
Observation: Load Semantics on SOM

- Missing Loads have different semantics from Long-Latency instructions
- Existing schedulers treat them as equivalent



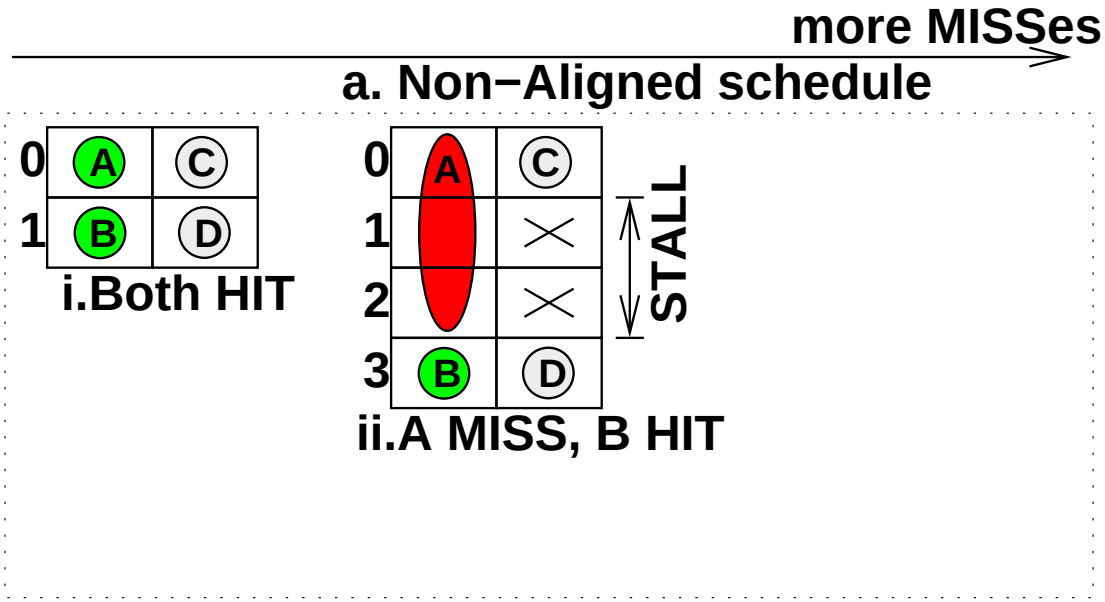
Motivation: Miss Overlapping

- Baseline Scheduler not effective on SOM



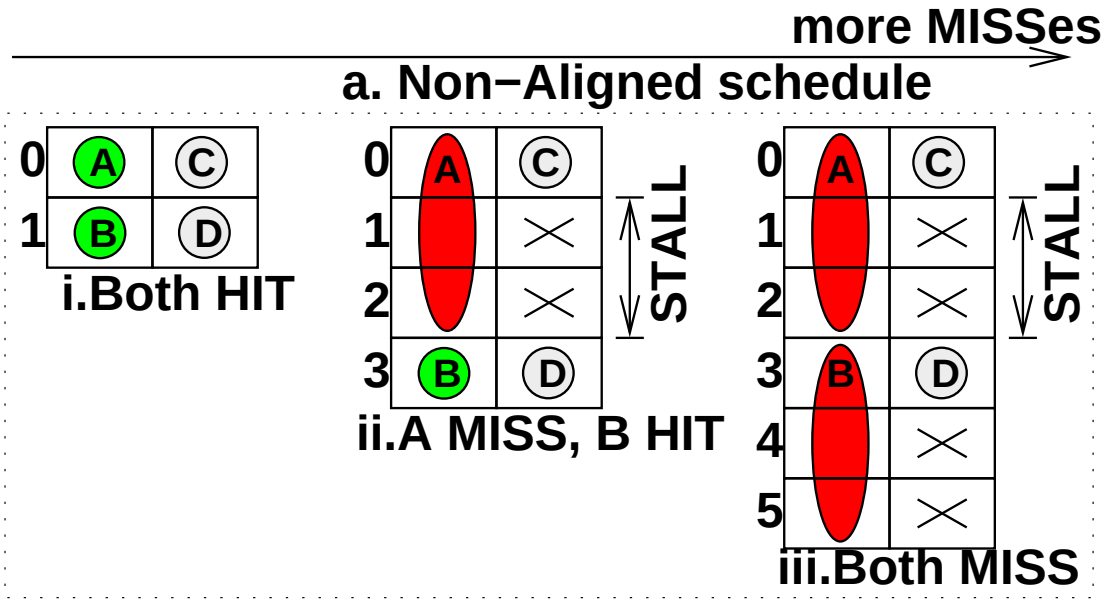
Motivation: Miss Overlapping

- Baseline Scheduler not effective on SOM



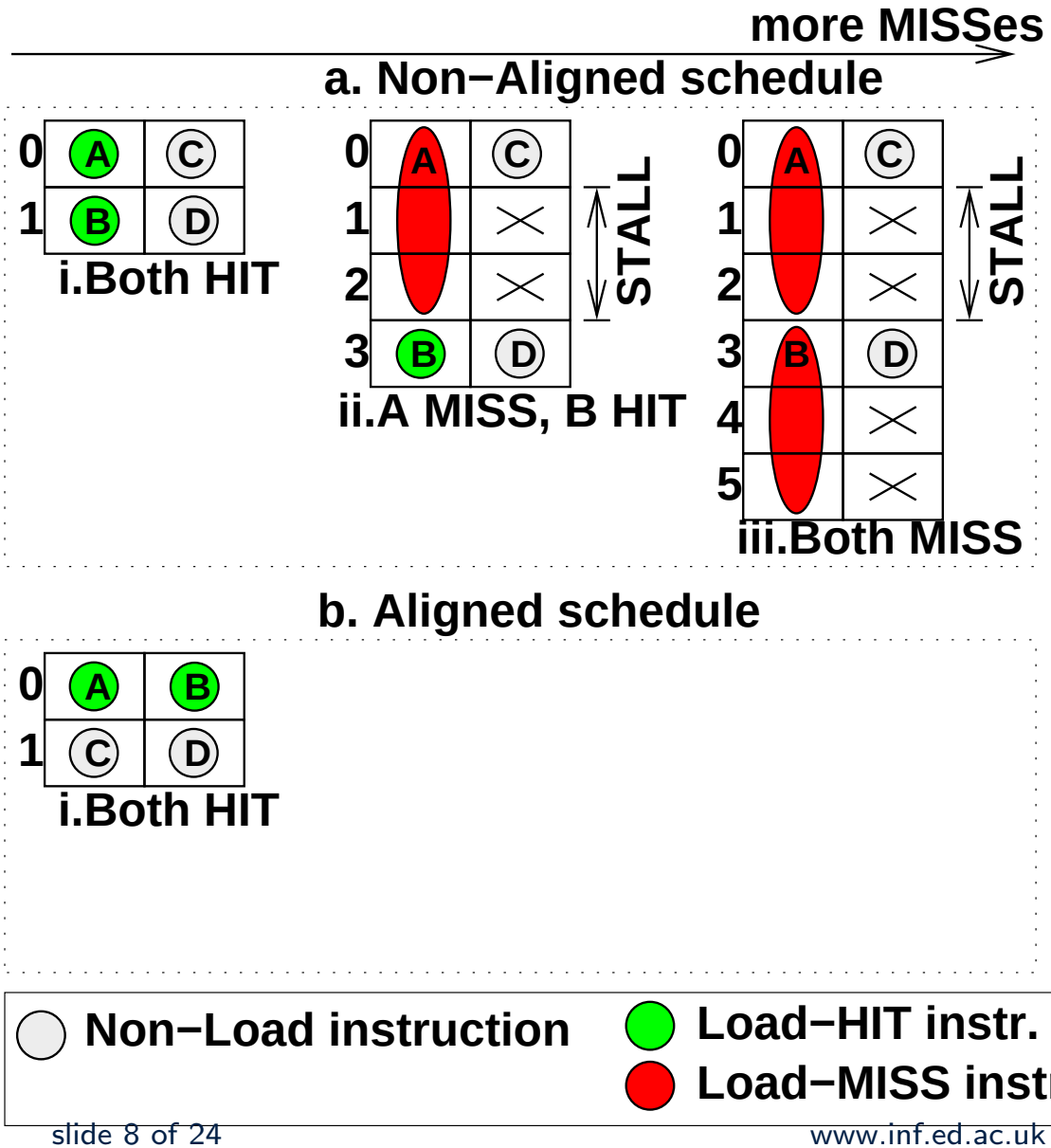
Motivation: Miss Overlapping

- Baseline Scheduler not effective on SOM
- Suffers from consecutive stalls



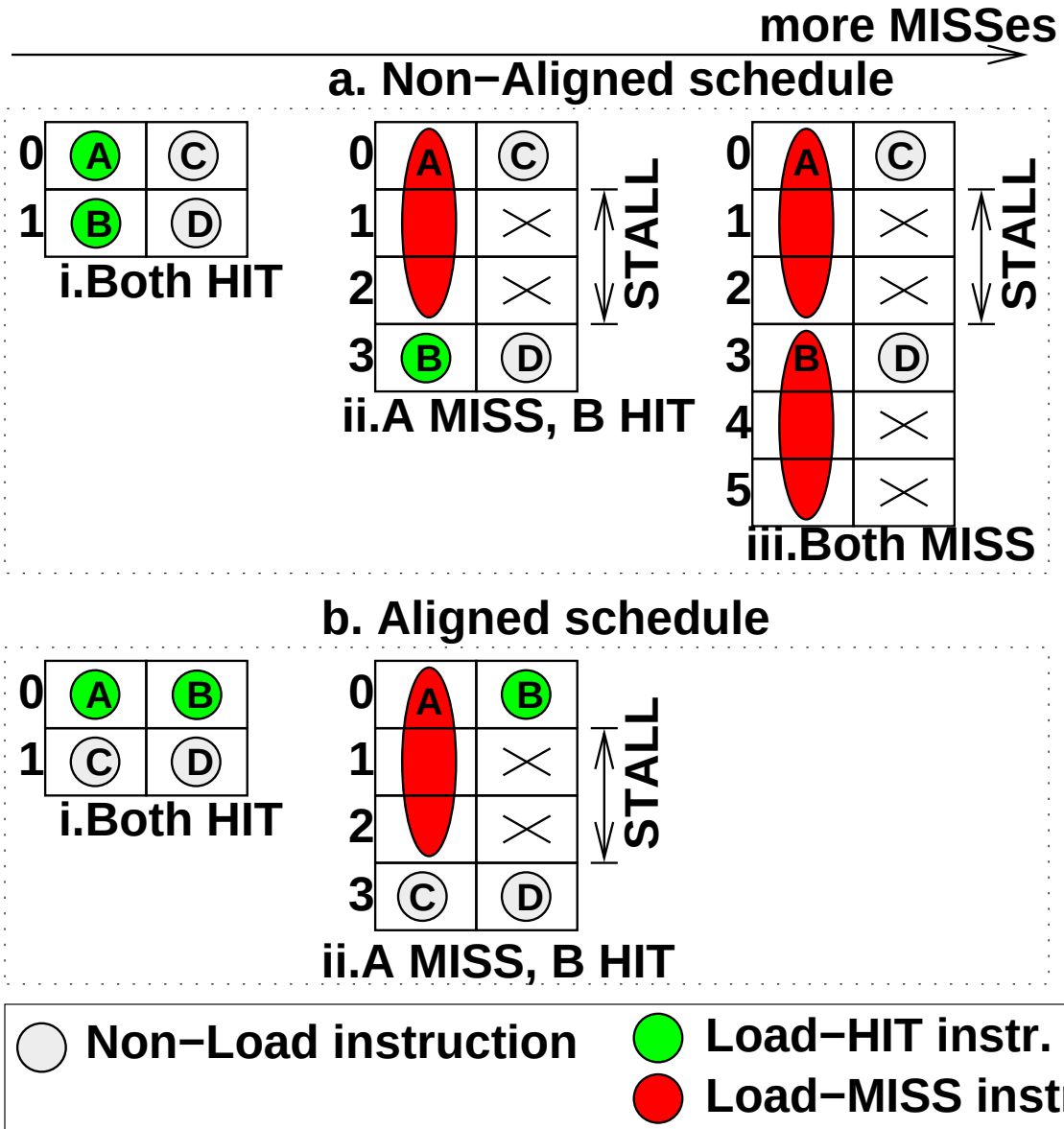
Motivation: Miss Overlapping

- Baseline Scheduler not effective on SOM
- Suffers from consecutive stalls
- Aligned Scheduling optimized for SOM



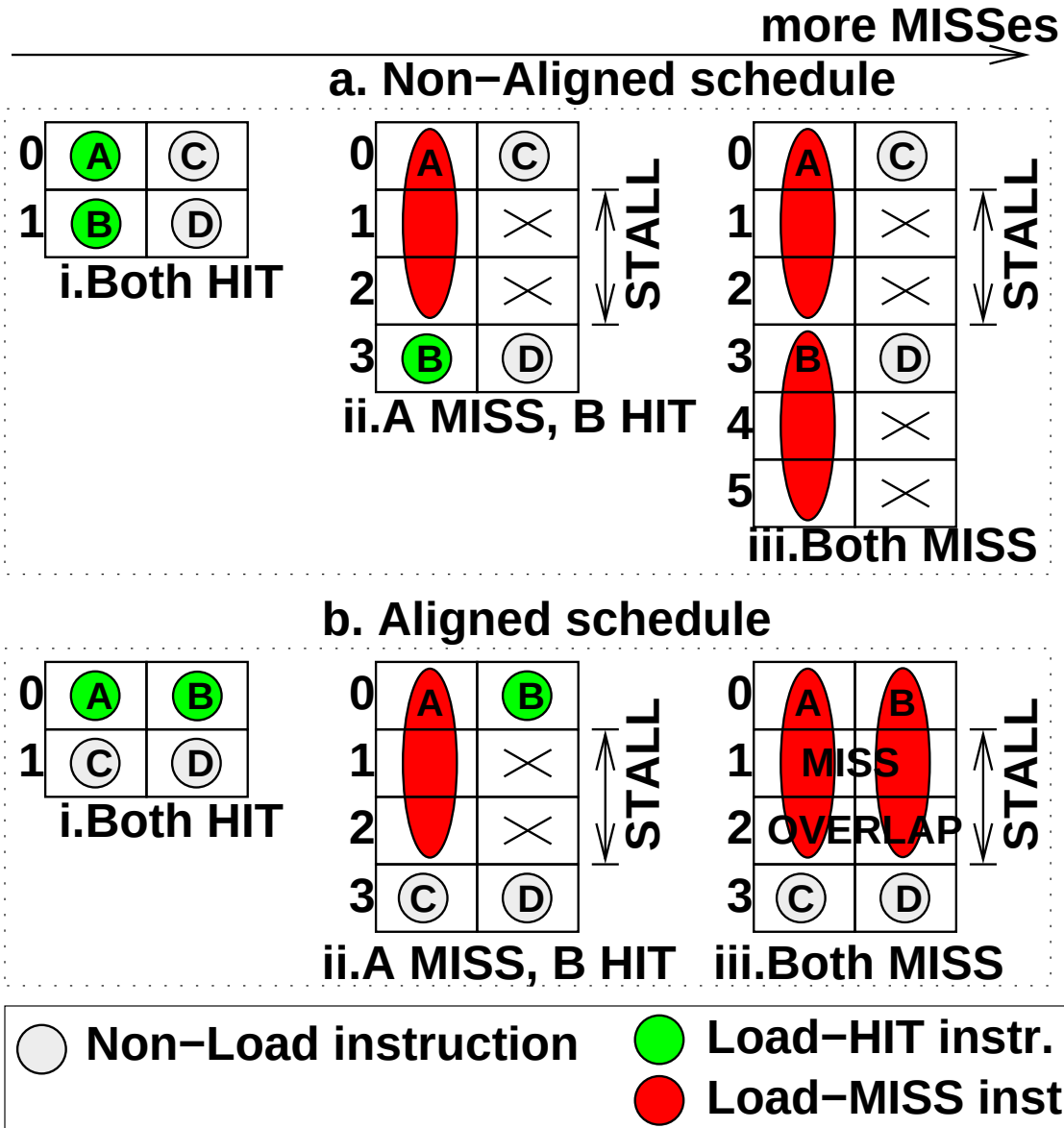
Motivation: Miss Overlapping

- Baseline Scheduler not effective on SOM
- Suffers from consecutive stalls
- Aligned Scheduling optimized for SOM



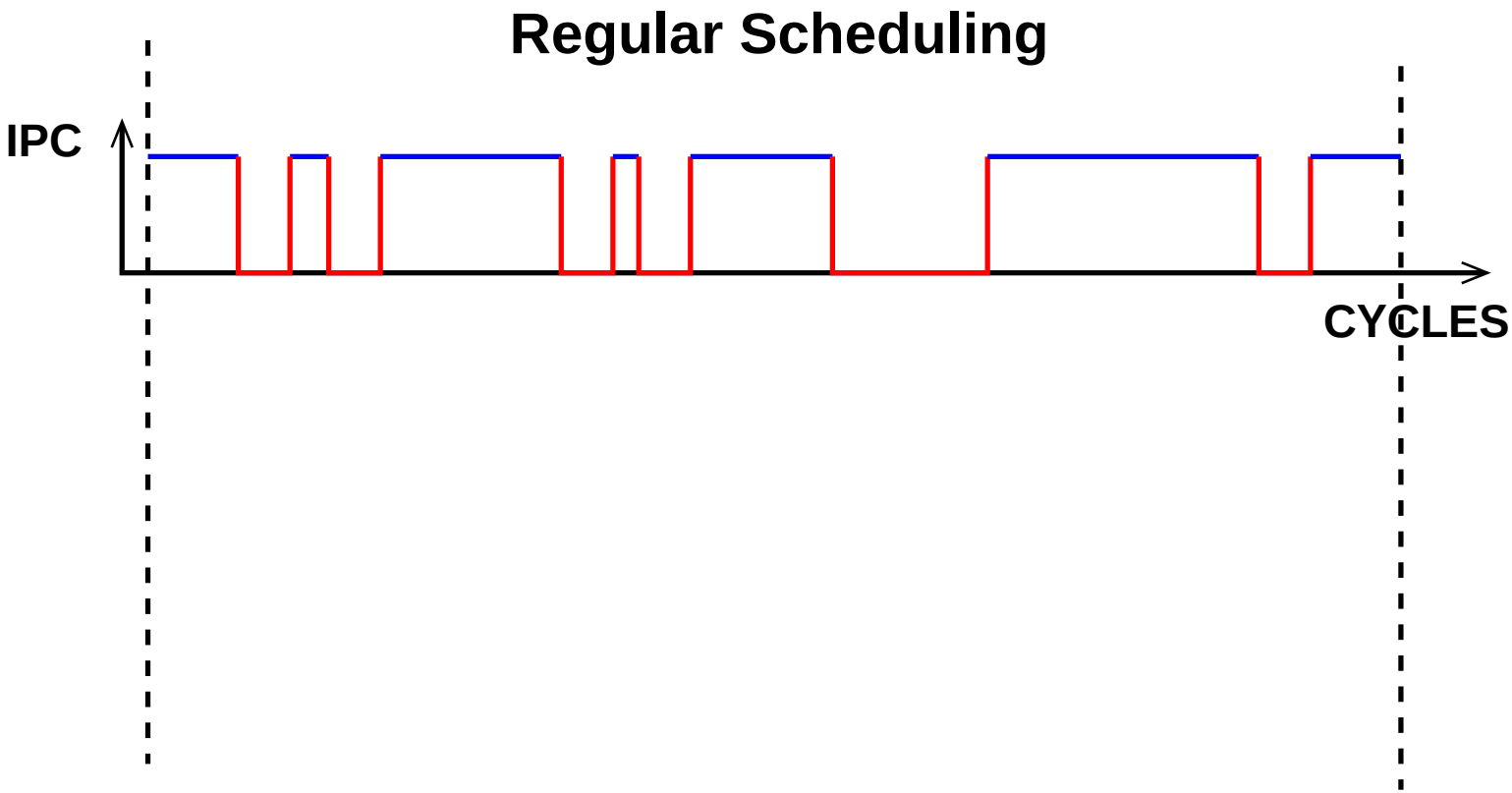
Motivation: Miss Overlapping

- Baseline Scheduler not effective on SOM
- Suffers from consecutive stalls
- Aligned Scheduling optimized for SOM
- Exploit Multiple simultaneous Misses



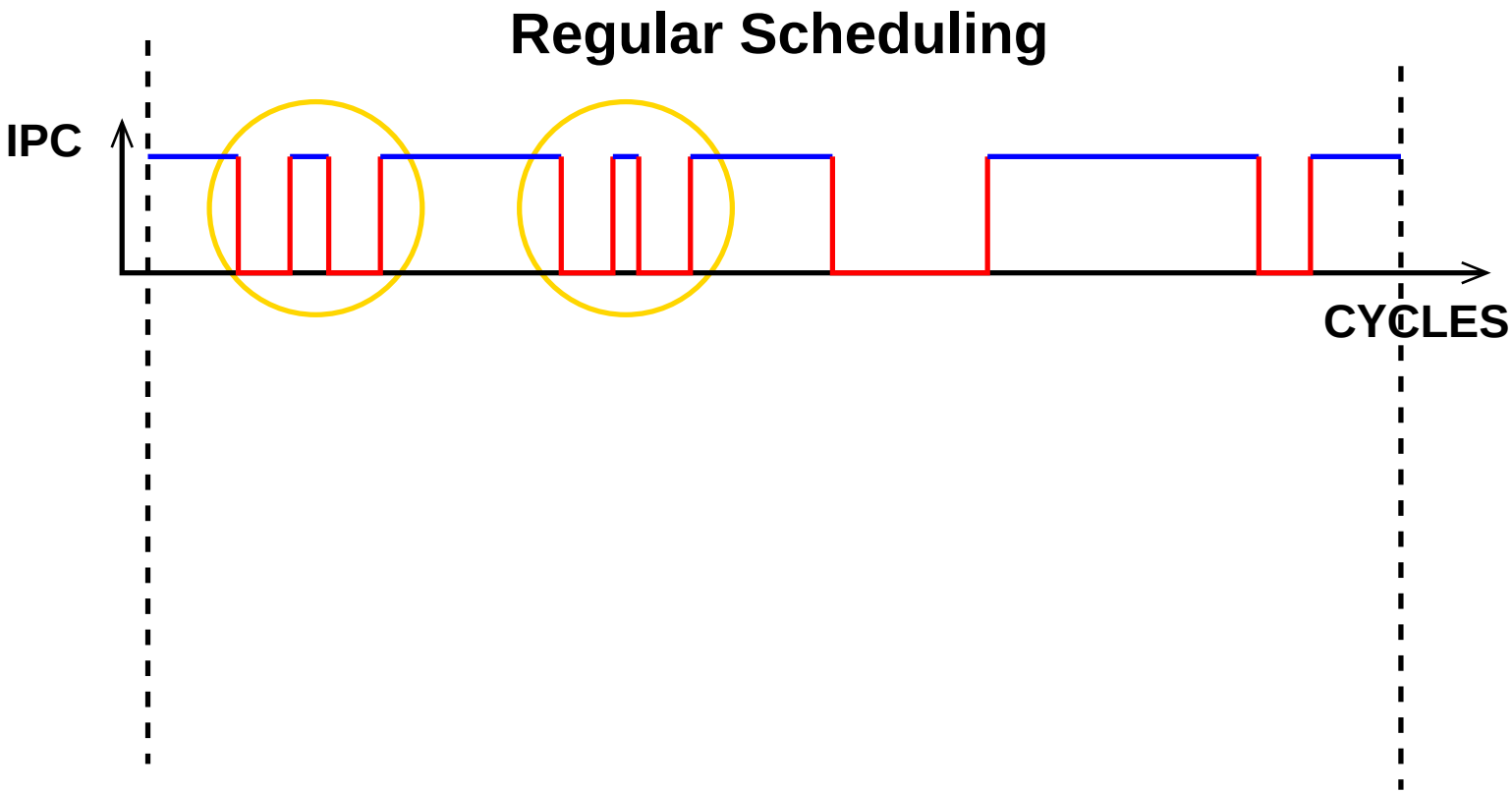
Motivation: Aligning Multiple Loads

- SOM hardware is incapable of hiding misses



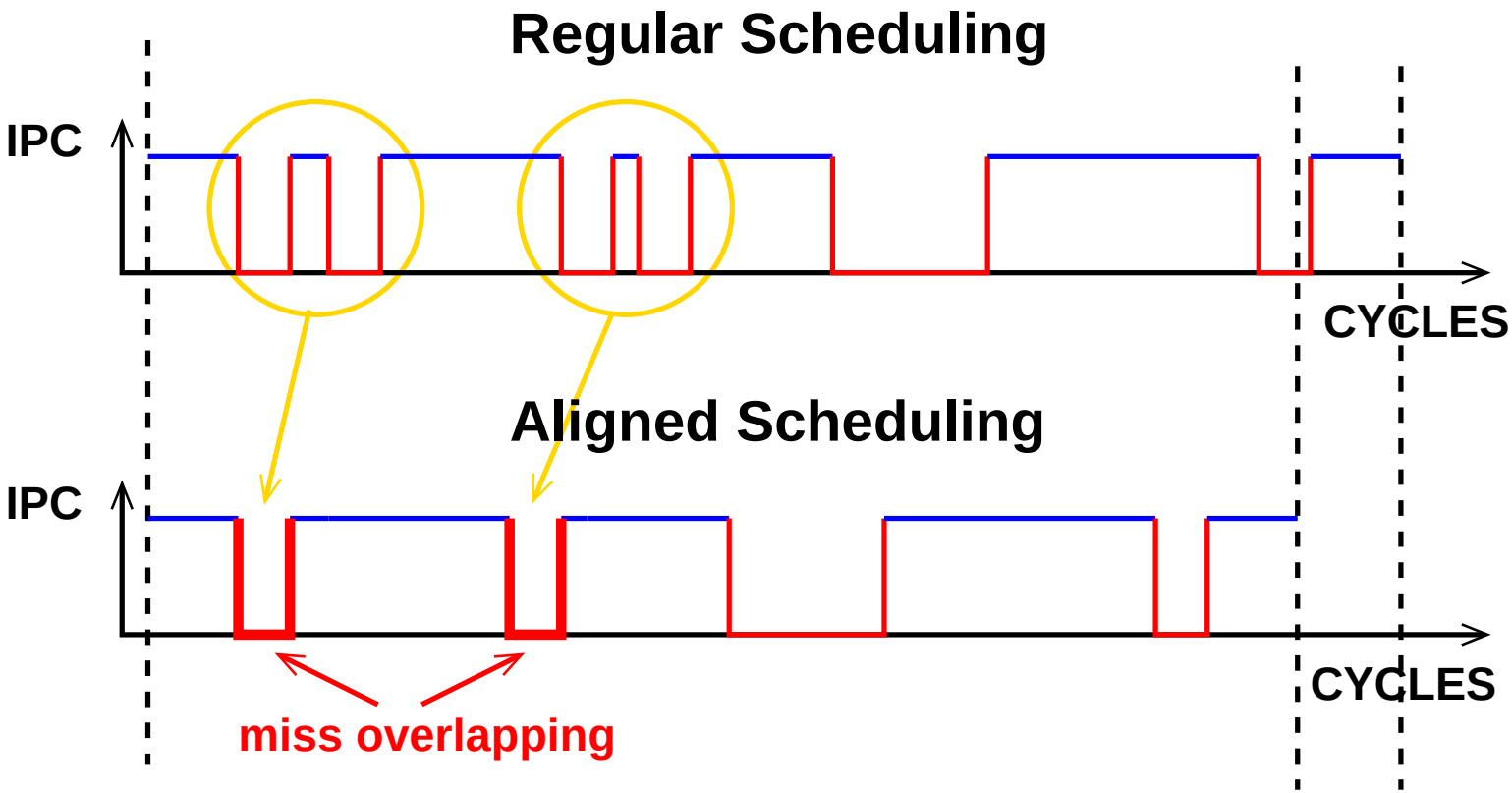
Motivation: Aligning Multiple Loads

- SOM hardware is incapable of hiding misses
- Improve performance by aligning Loads



Motivation: Aligning Multiple Loads

- SOM hardware is incapable of hiding misses
- Improve performance by aligning Loads
- Multiple misses occur simultaneously $\longrightarrow$ fewer stalls



Outline

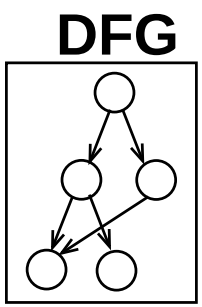
Introduction

Aligned Scheduling

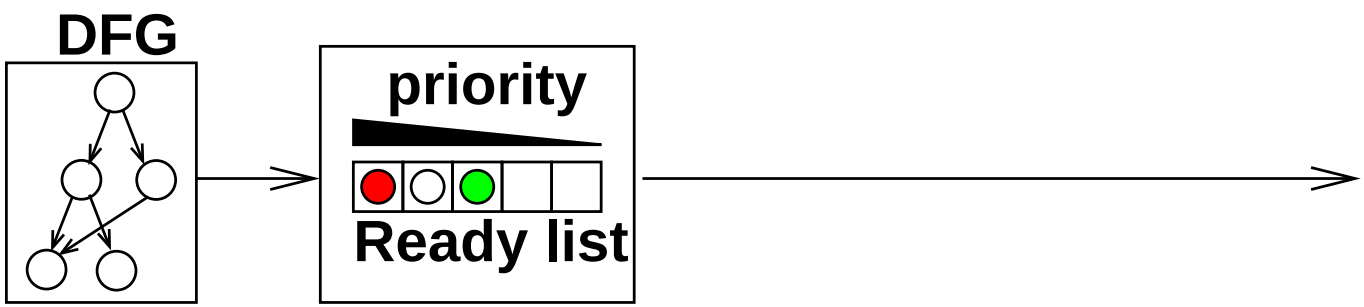
Experimental Setup and Results

Conclusion

Aligned Scheduling

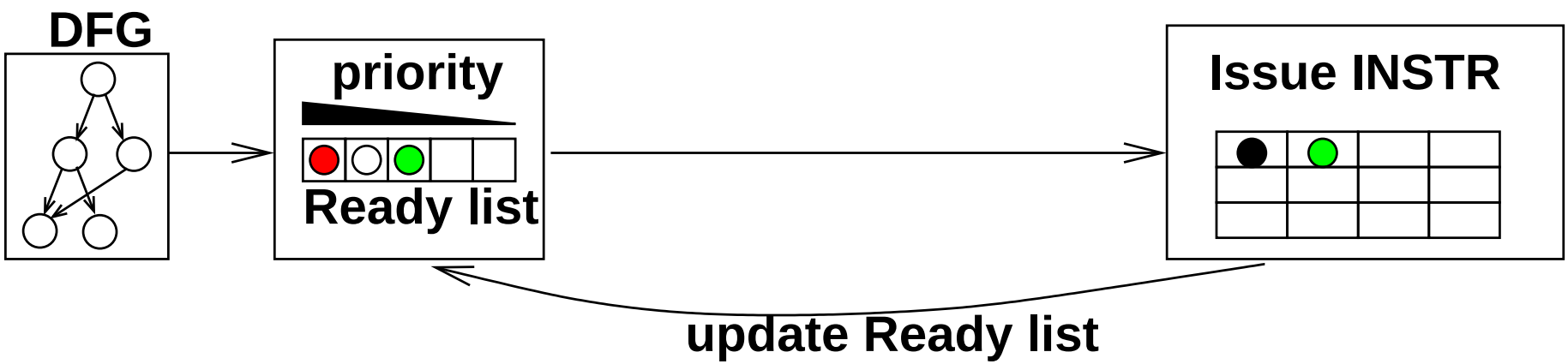


Aligned Scheduling



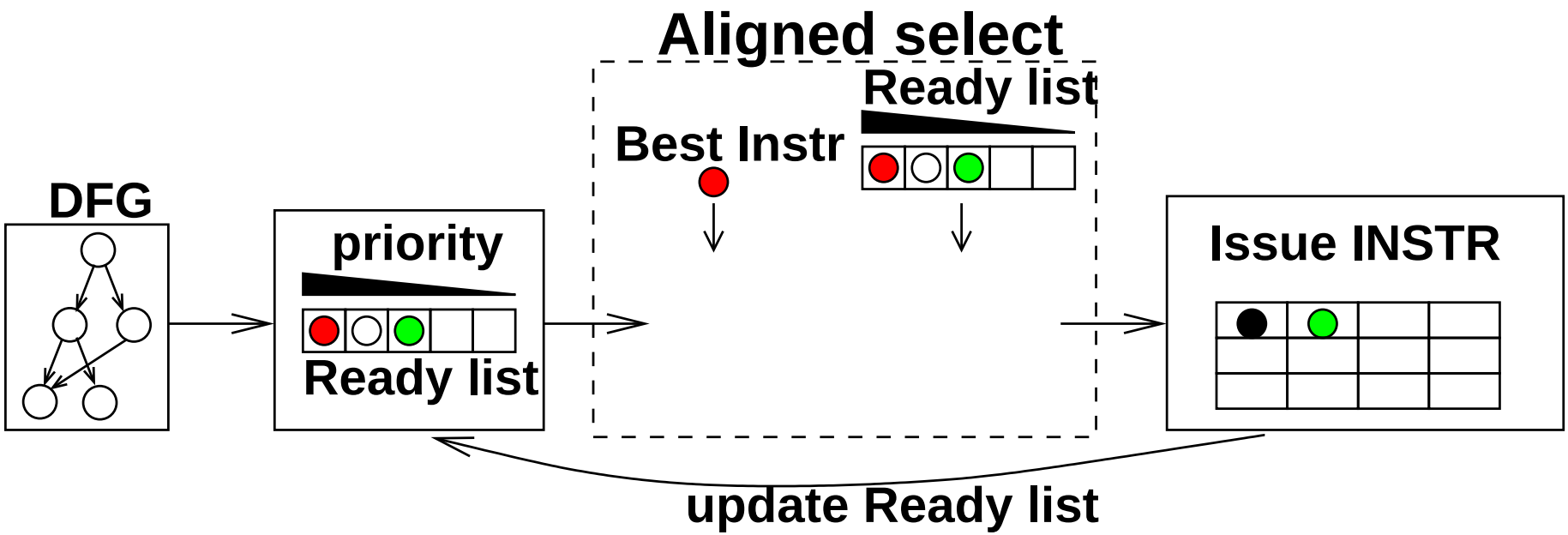
Aligned Scheduling

Aligned Scheduling



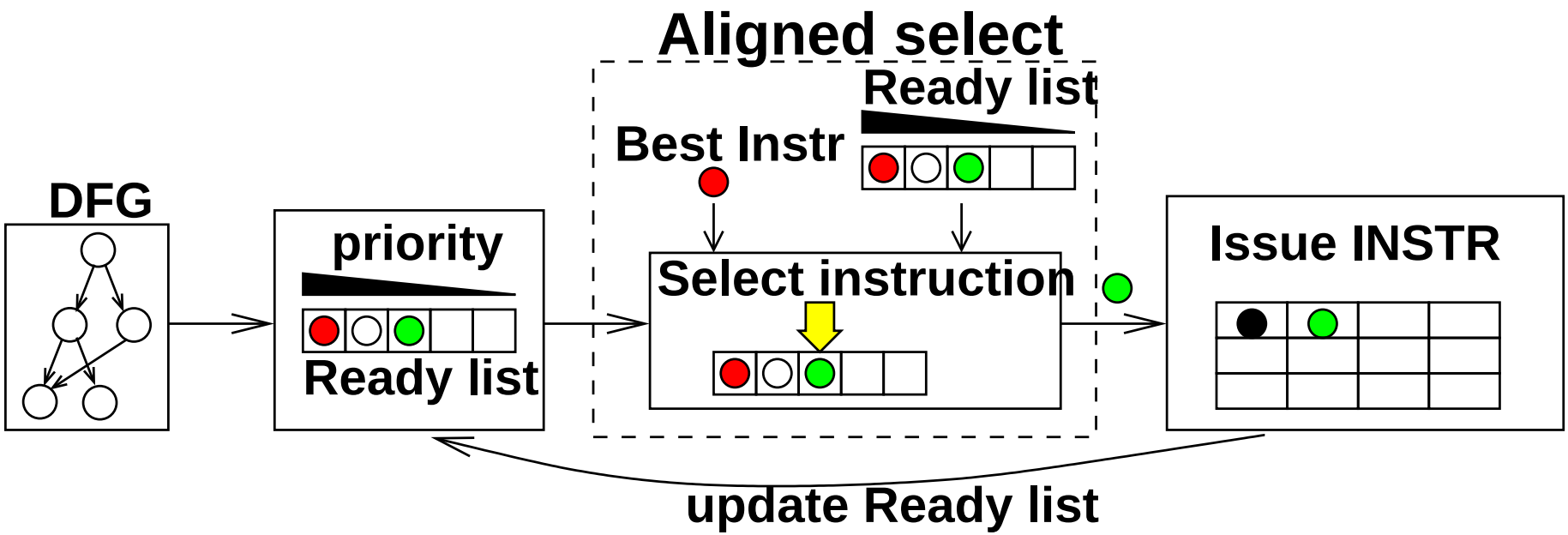
Aligned Scheduling

- Plug-in to well established scheduler structure



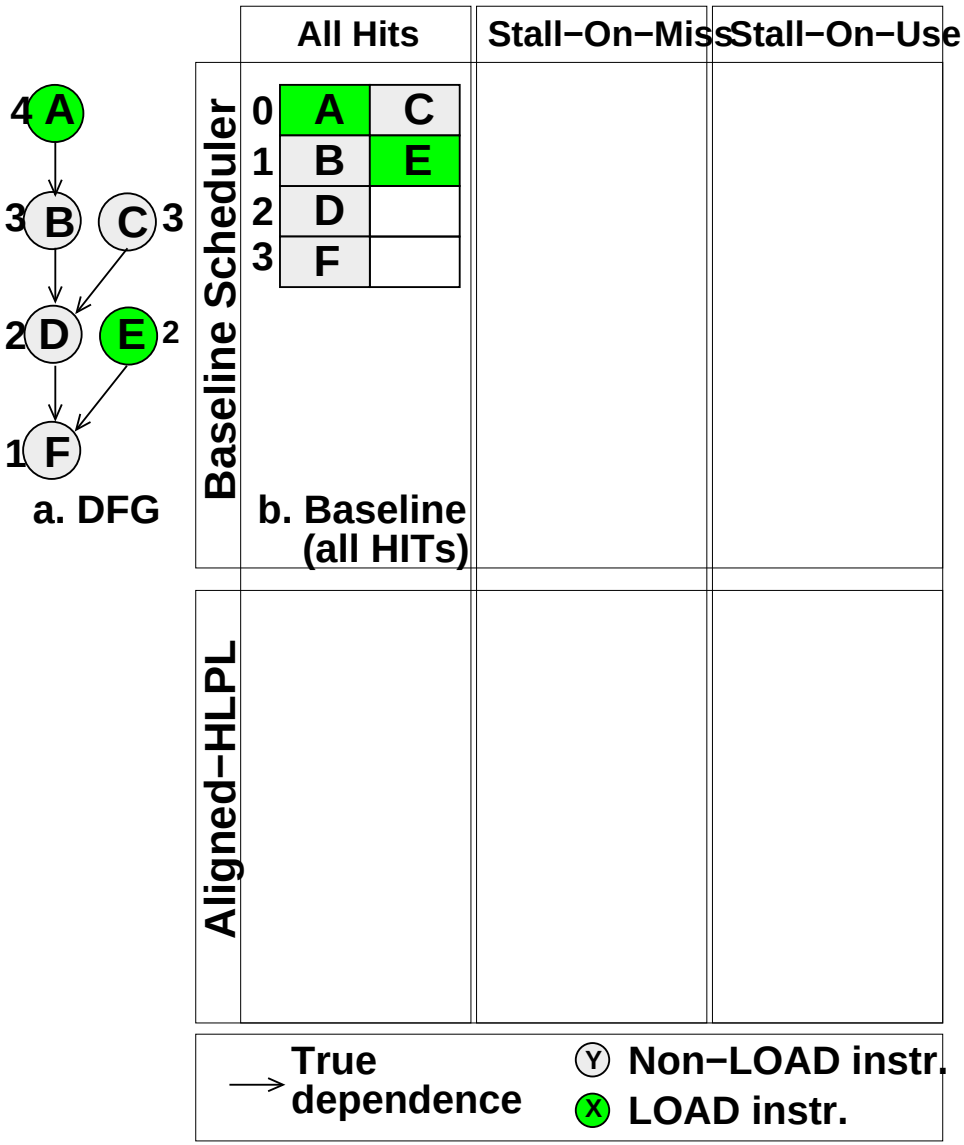
Aligned Scheduling

- Plug-in to well established scheduler structure
- Better selection of instruction to be scheduled
- Override default priorities



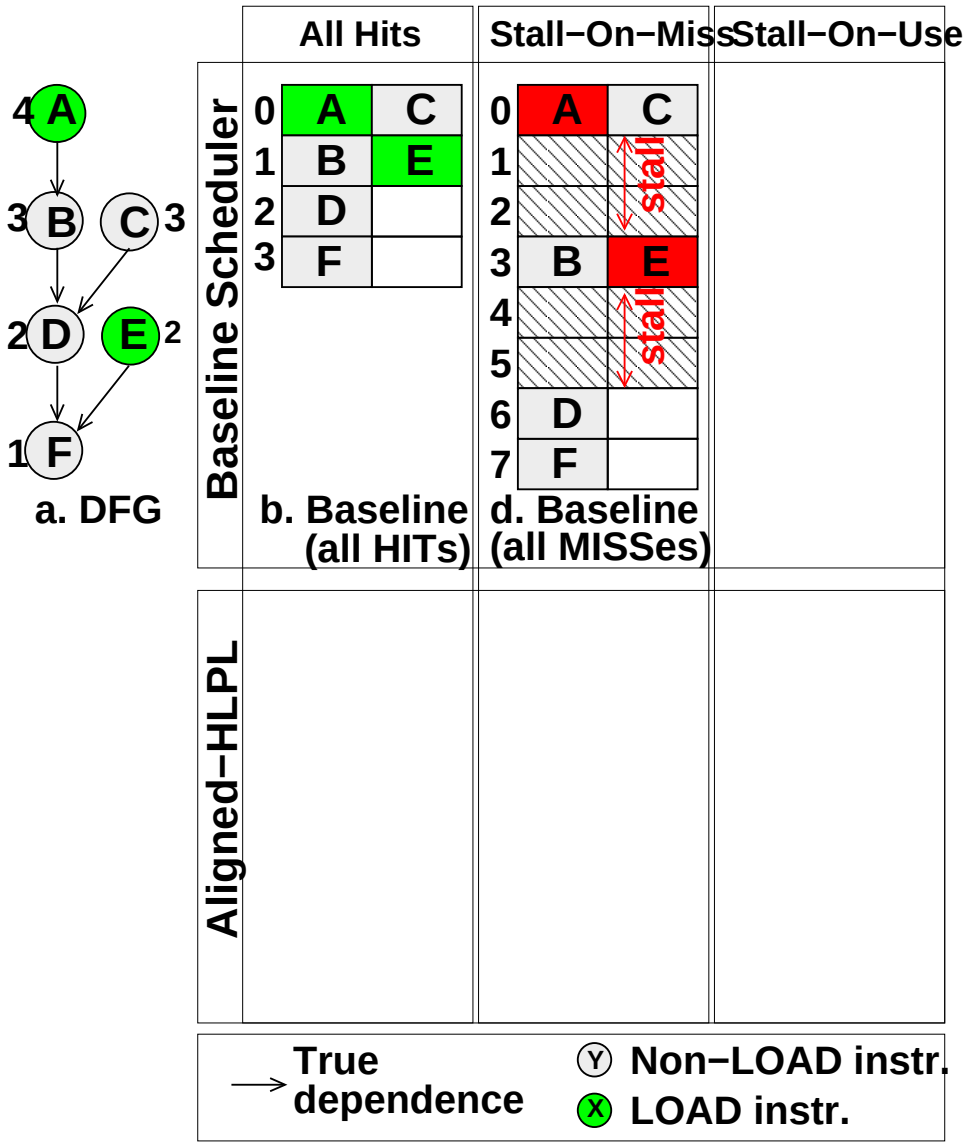
Aligned-HLPL (Hoist Low Priority Loads)

- Unaligned Loads



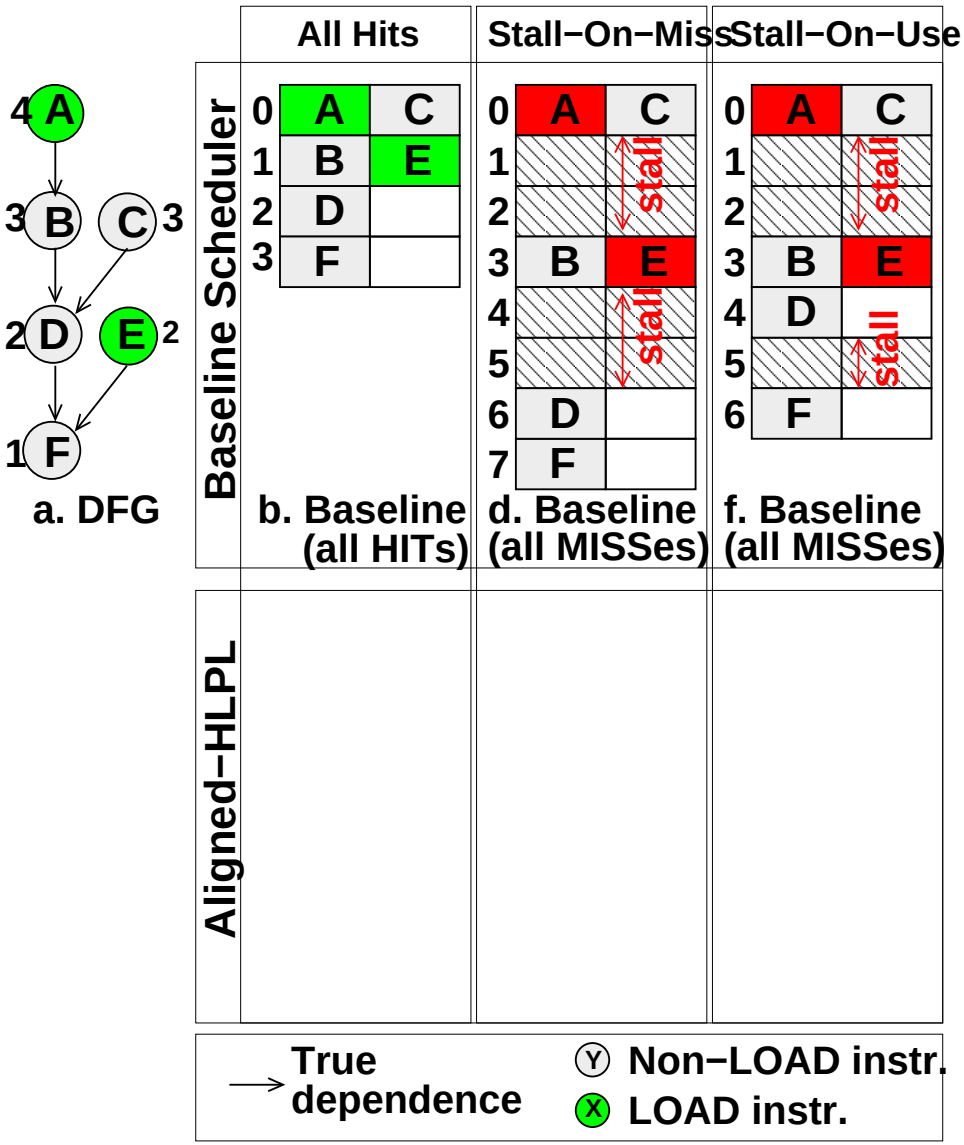
Aligned-HLPL (Hoist Low Priority Loads)

- Unaligned Loads



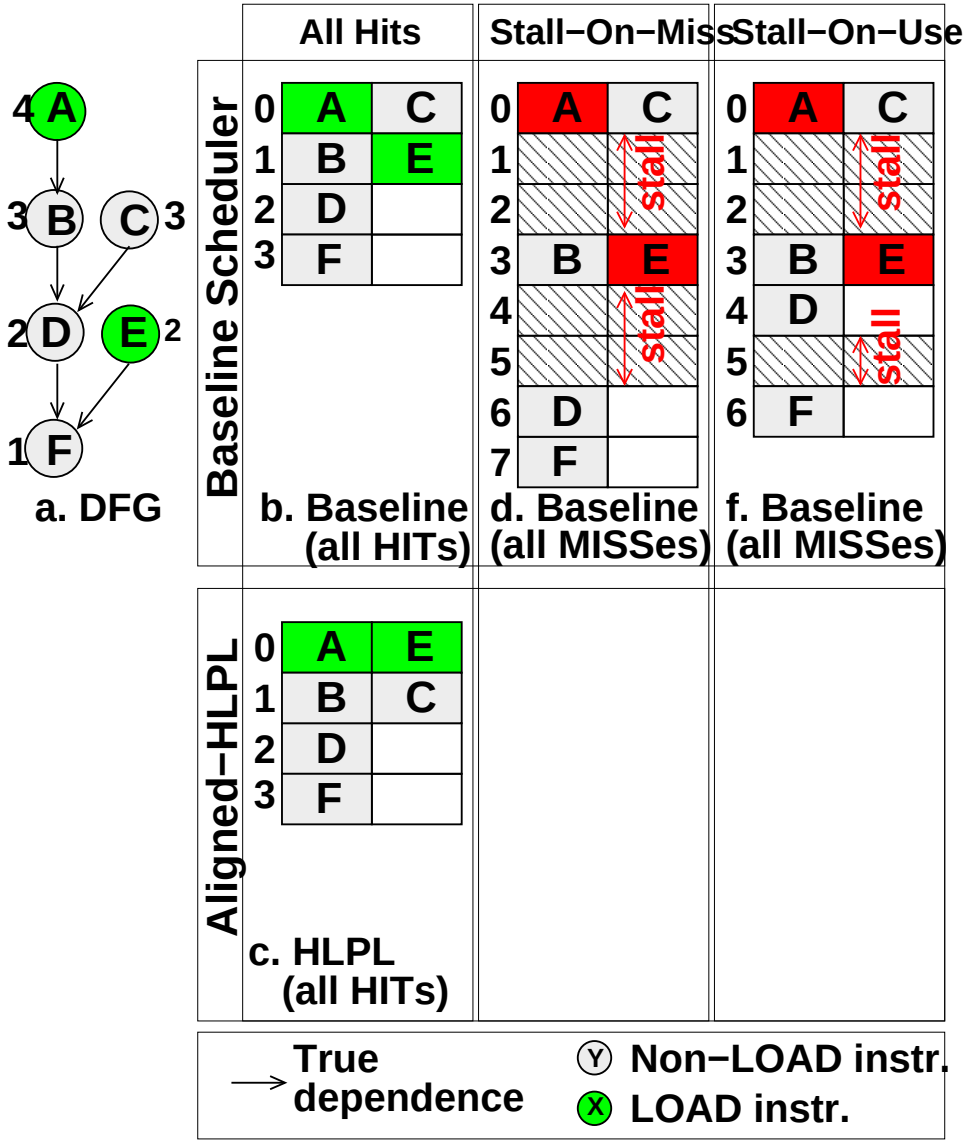
Aligned-HLPL (Hoist Low Priority Loads)

- Unaligned Loads



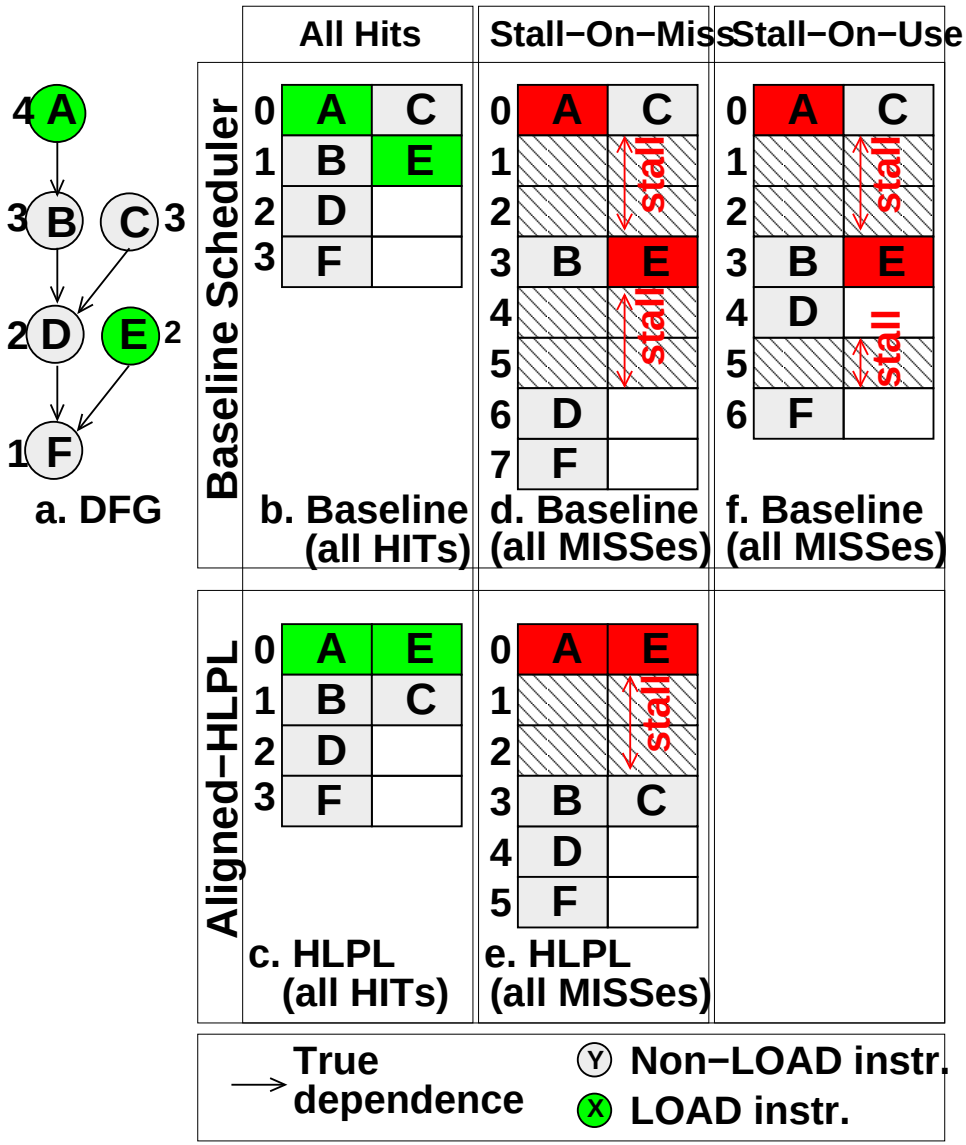
Aligned-HLPL (Hoist Low Priority Loads)

- Unaligned Loads
- Prefer Low Priority Load if already scheduled a Load



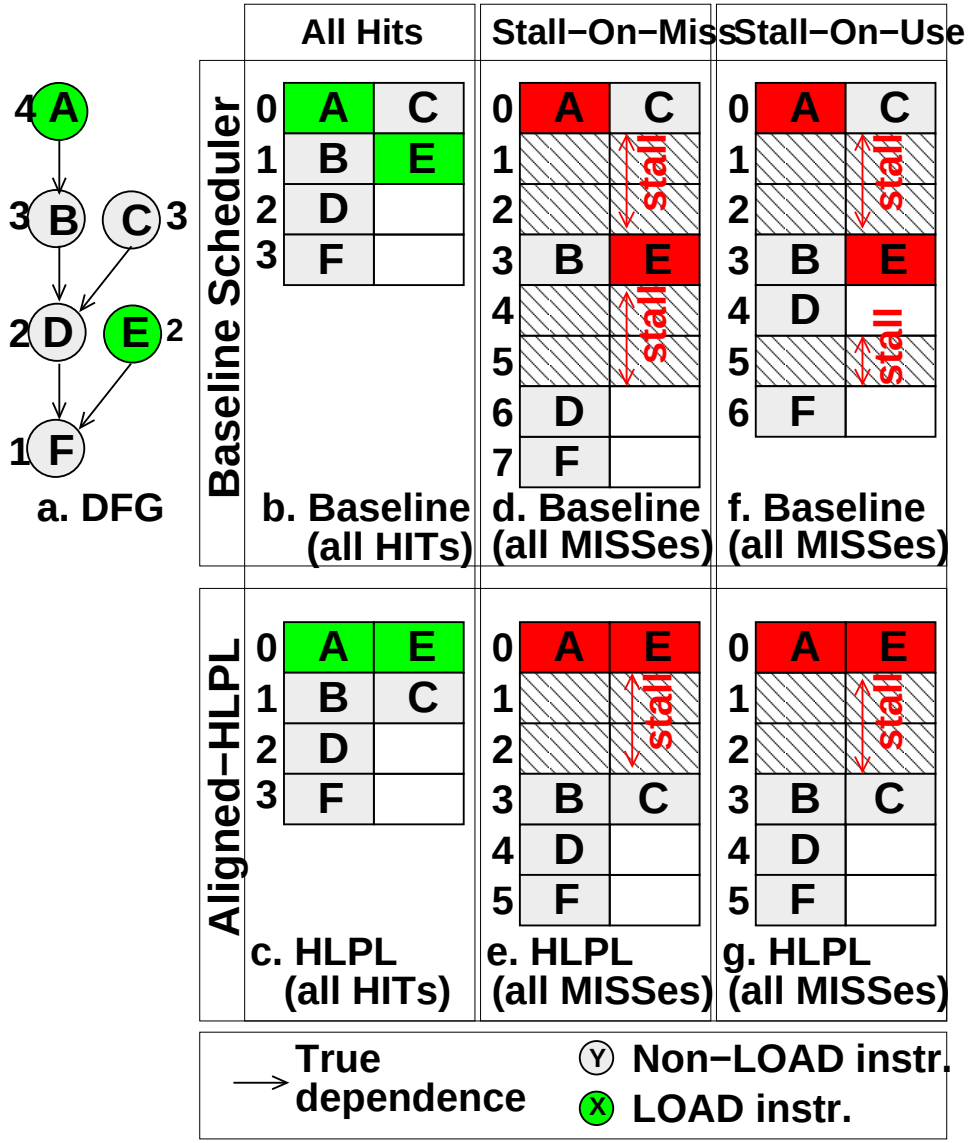
Aligned-HLPL (Hoist Low Priority Loads)

- Unaligned Loads
- Prefer Low Priority Load if already scheduled a Load
- Miss overlapping
- Faster execution



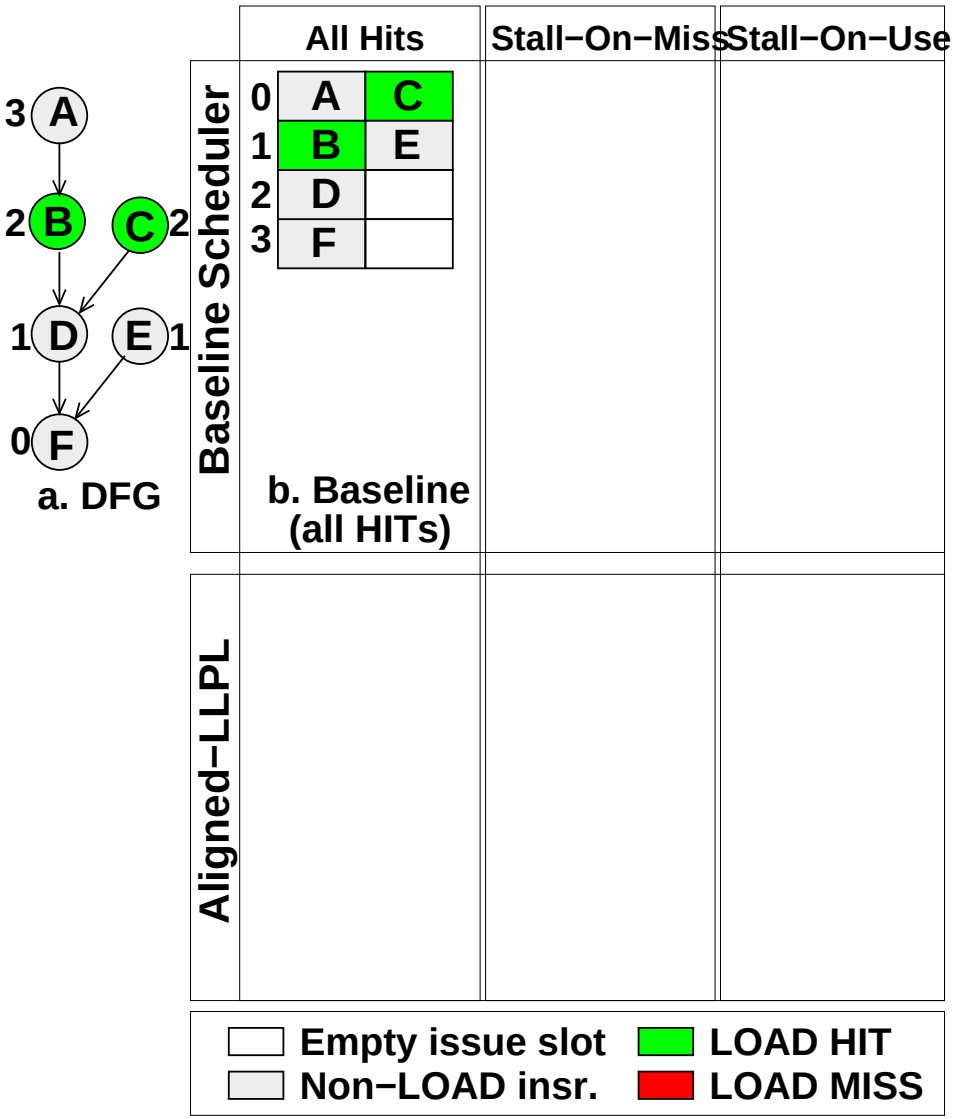
Aligned-HLPL (Hoist Low Priority Loads)

- Unaligned Loads
- Prefer Low Priority Load if already scheduled a Load
- Miss overlapping
- Faster execution



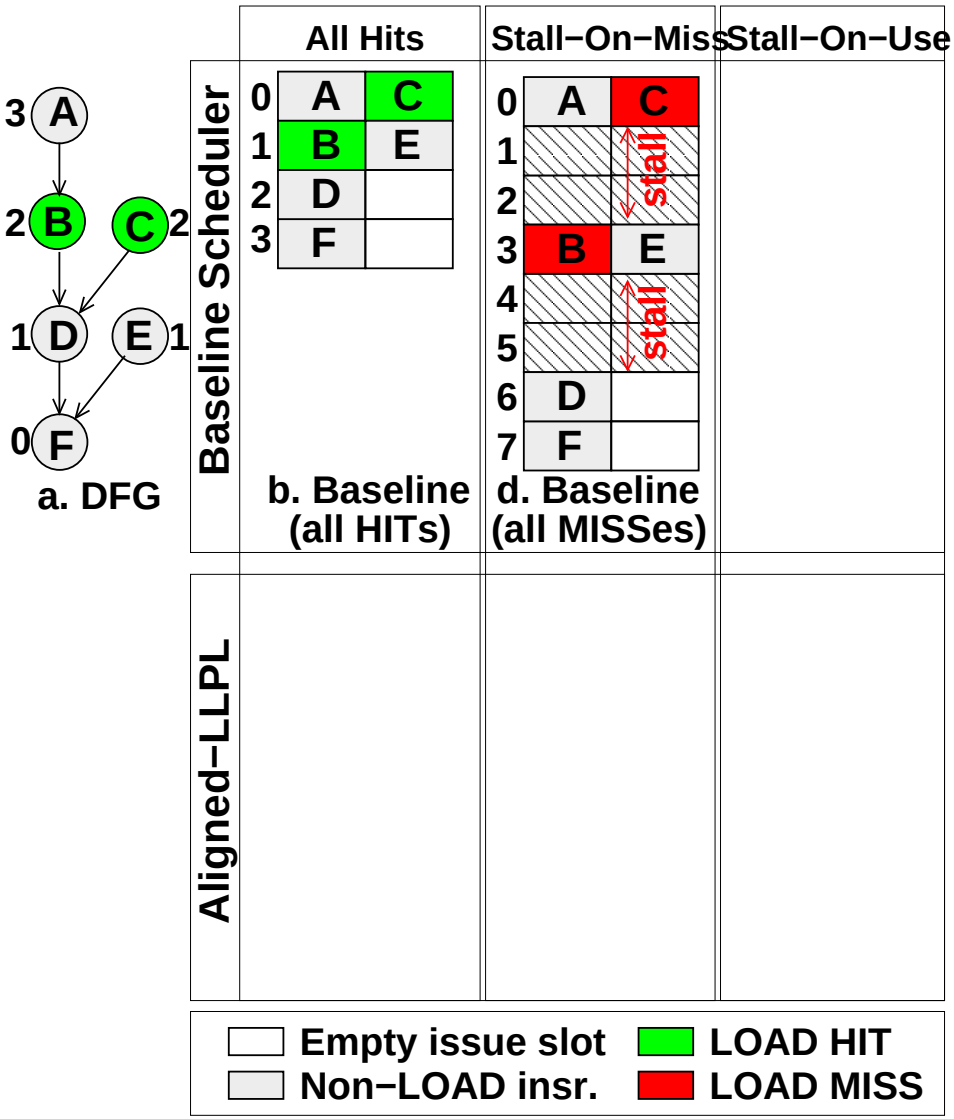
Aligned-LLPL (Lower Low Priority Loads)

- Unaligned Loads



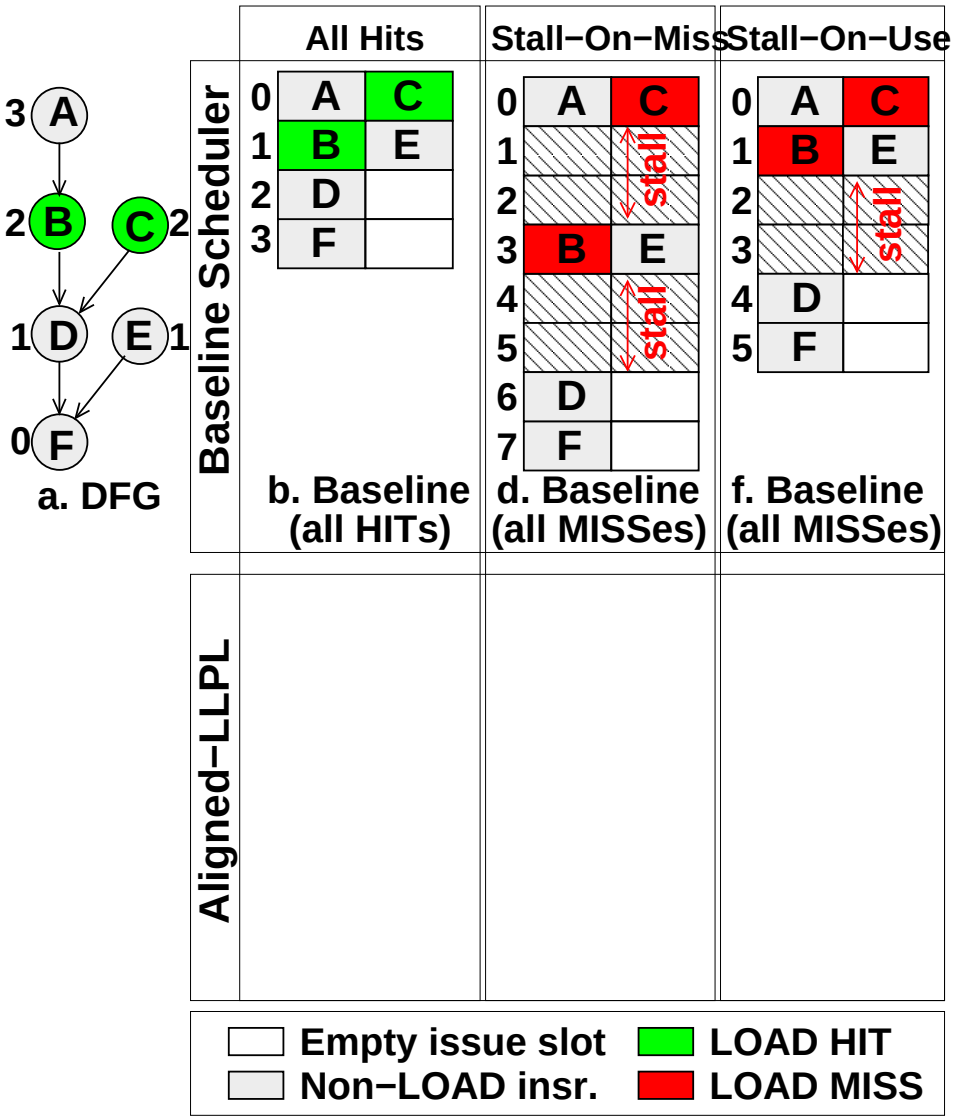
Aligned-LLPL (Lower Low Priority Loads)

- Unaligned Loads



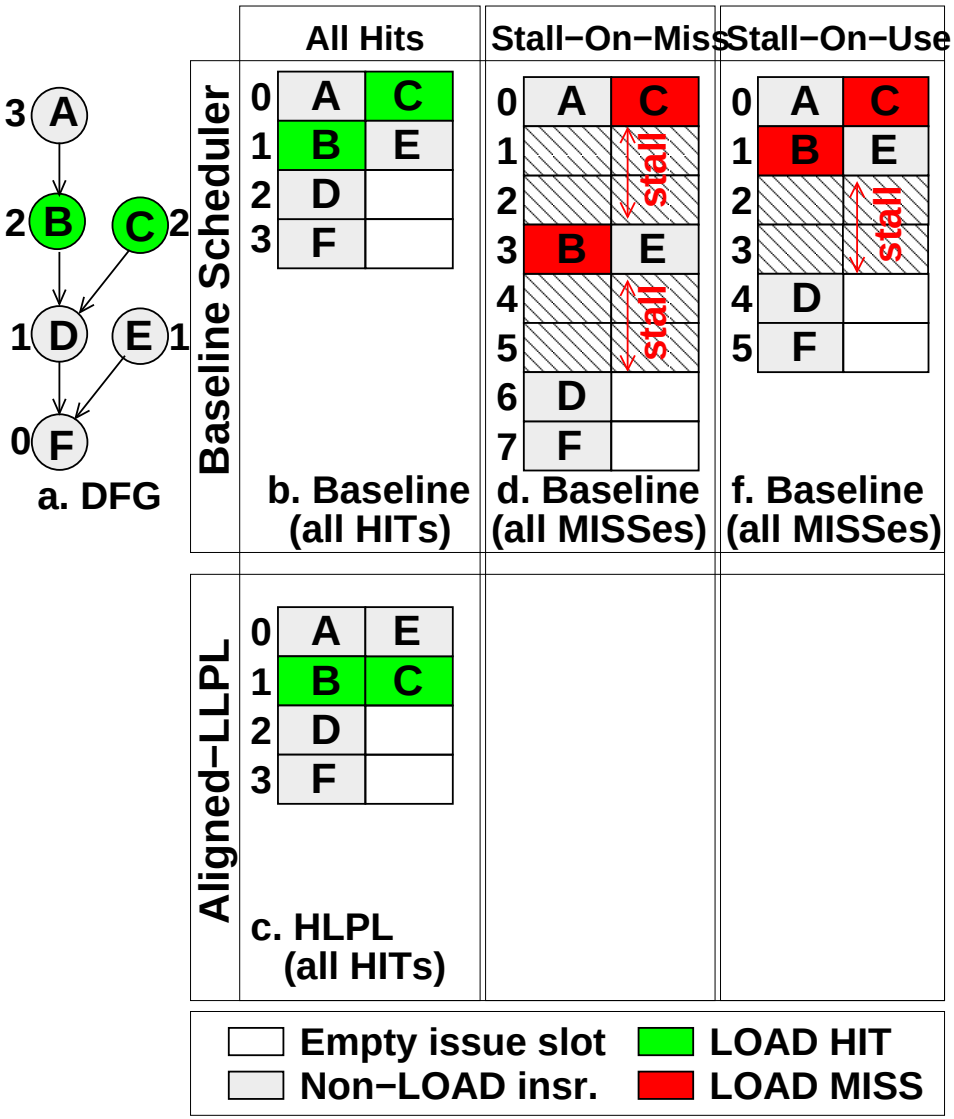
Aligned-LLPL (Lower Low Priority Loads)

- Unaligned Loads



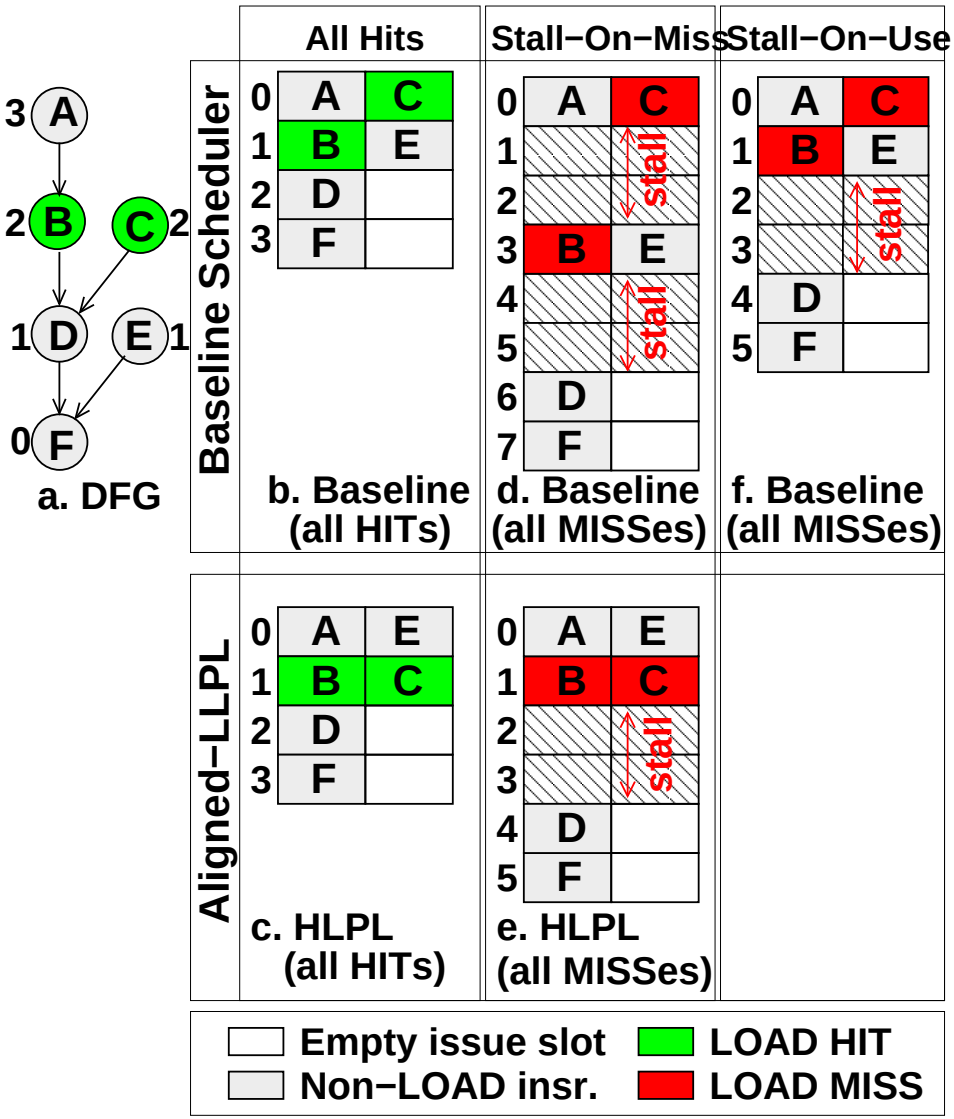
Aligned-LLPL (Lower Low Priority Loads)

- Unaligned Loads
- Delay the execution of low-priority Loads



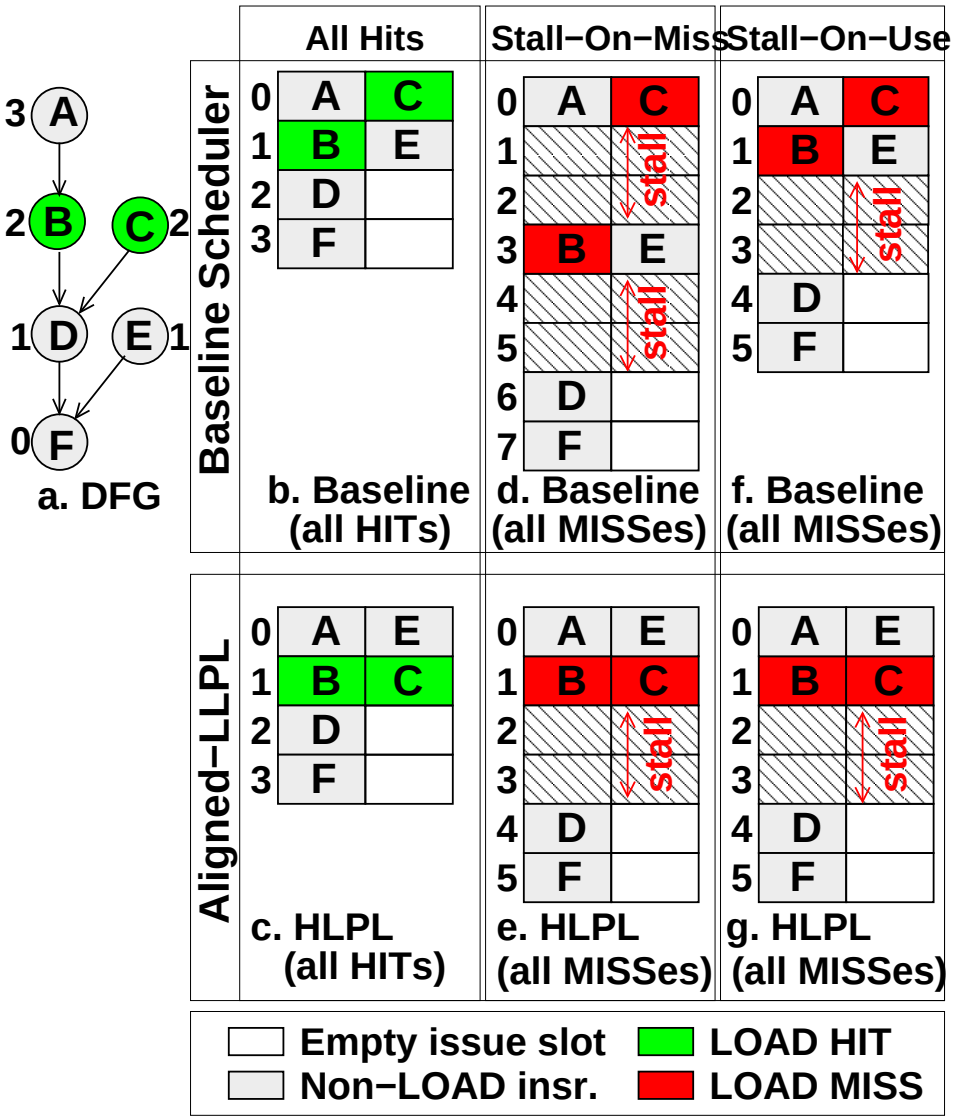
Aligned-LLPL (Lower Low Priority Loads)

- Unaligned Loads
- Delay the execution of low-priority Loads
- Miss overlapping
- Increases chances that Loads will be aligned later



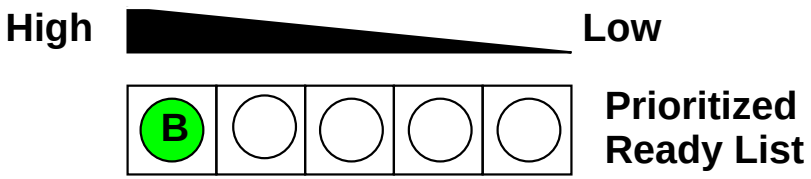
Aligned-LLPL (Lower Low Priority Loads)

- Unaligned Loads
- Delay the execution of low-priority Loads
- Miss overlapping
- Increases chances that Loads will be aligned later



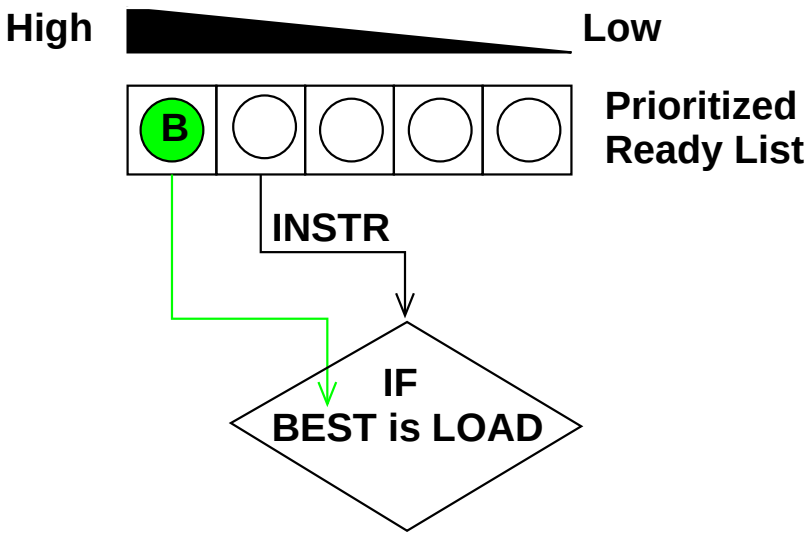
Alignment Heuristic

- Choose between HLPL and LLPL depending on highest priority instruction in the ready list (B)



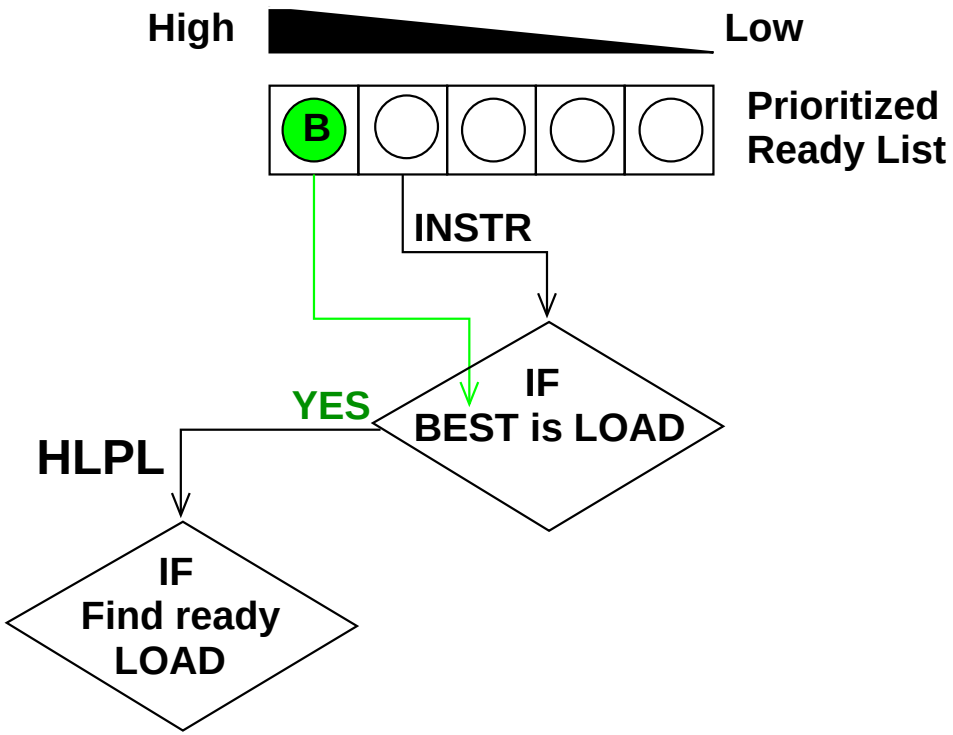
Alignment Heuristic

- Choose between HLPL and LLPL depending on highest priority instruction in the ready list (B)



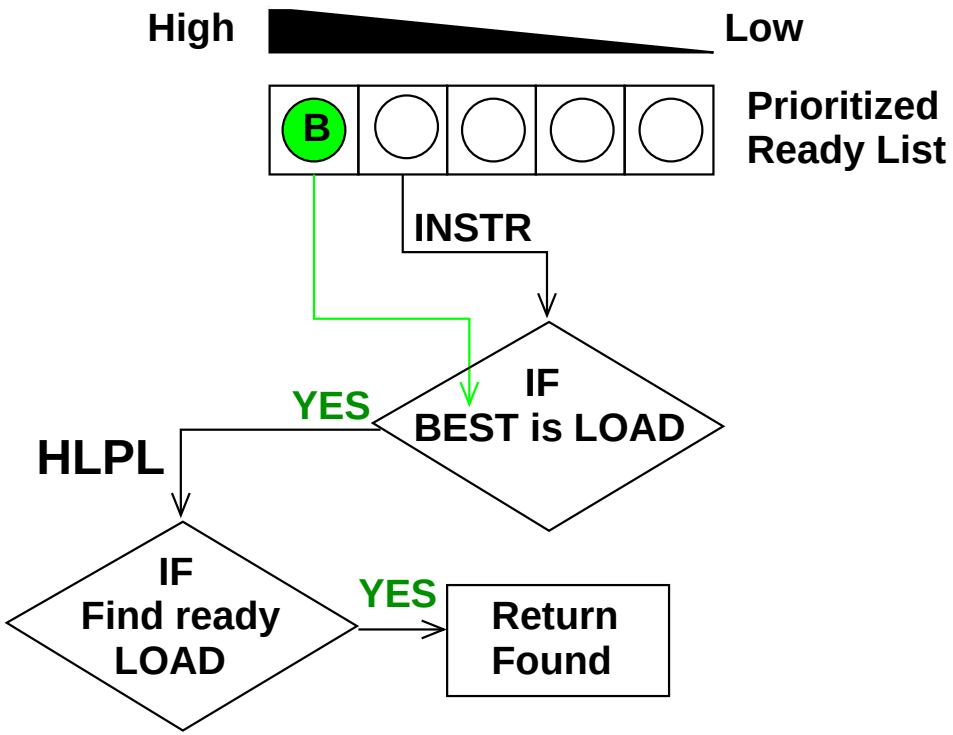
Alignment Heuristic

- Choose between HLPL and LLPL depending on highest priority instruction in the ready list (B)



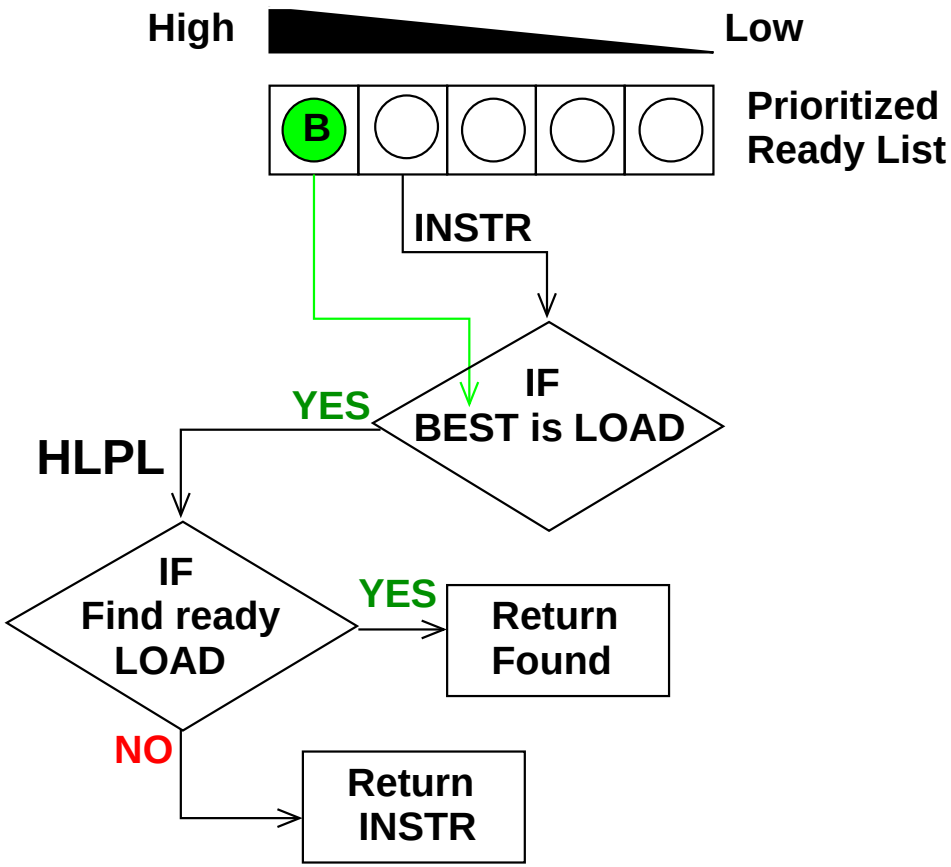
Alignment Heuristic

- Choose between HLPL and LLPL depending on highest priority instruction in the ready list (B)



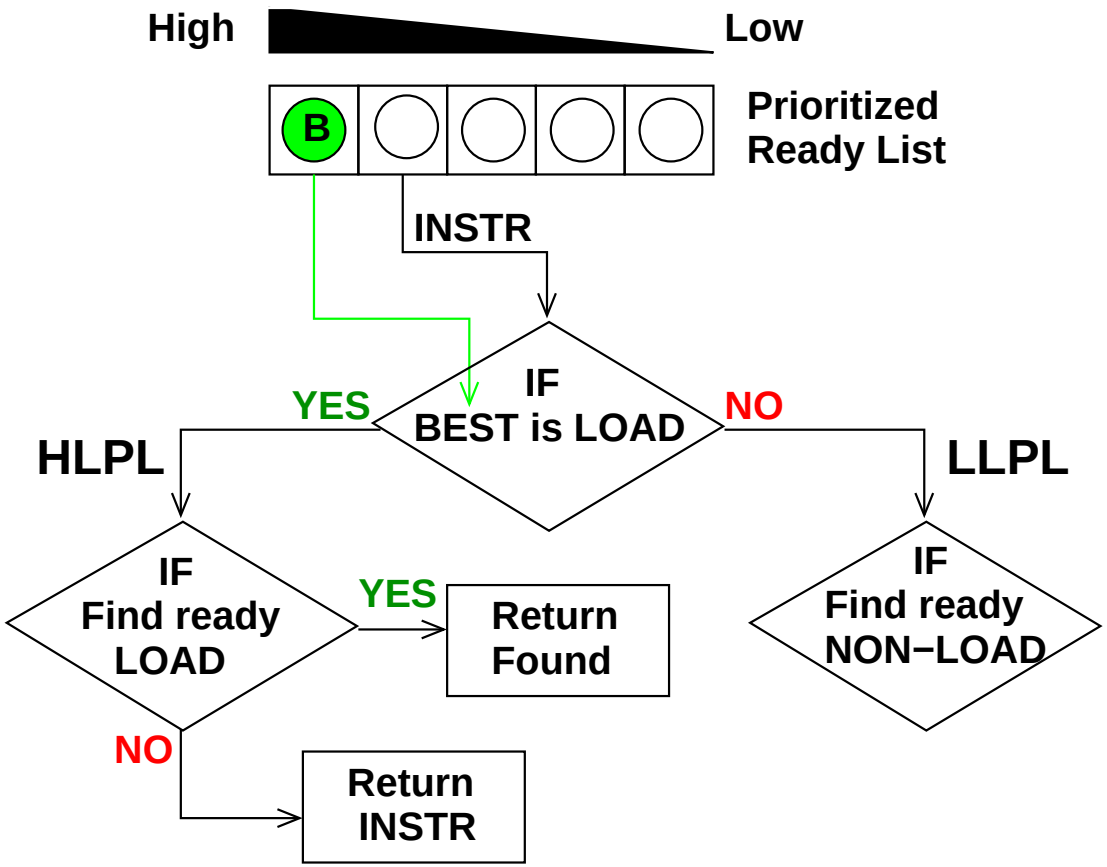
Alignment Heuristic

- Choose between HLPL and LLPL depending on highest priority instruction in the ready list (B)



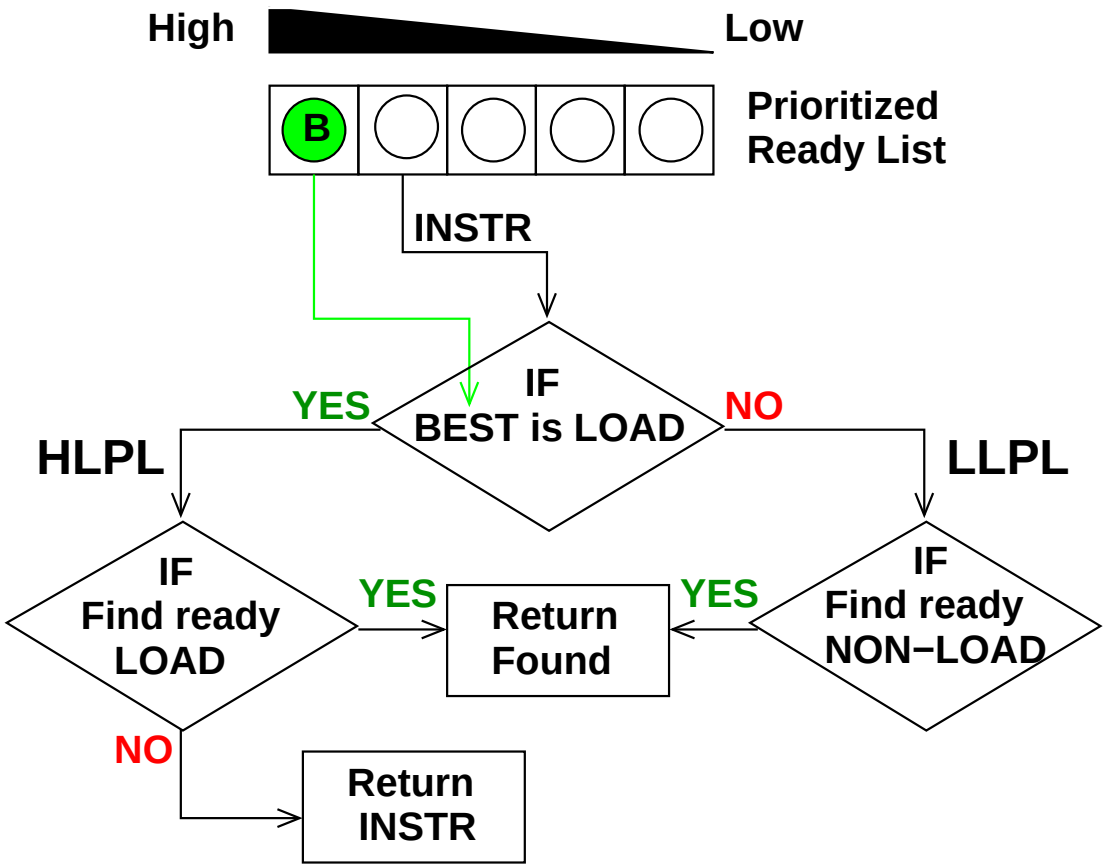
Alignment Heuristic

- Choose between HLPL and LLPL depending on highest priority instruction in the ready list (B)



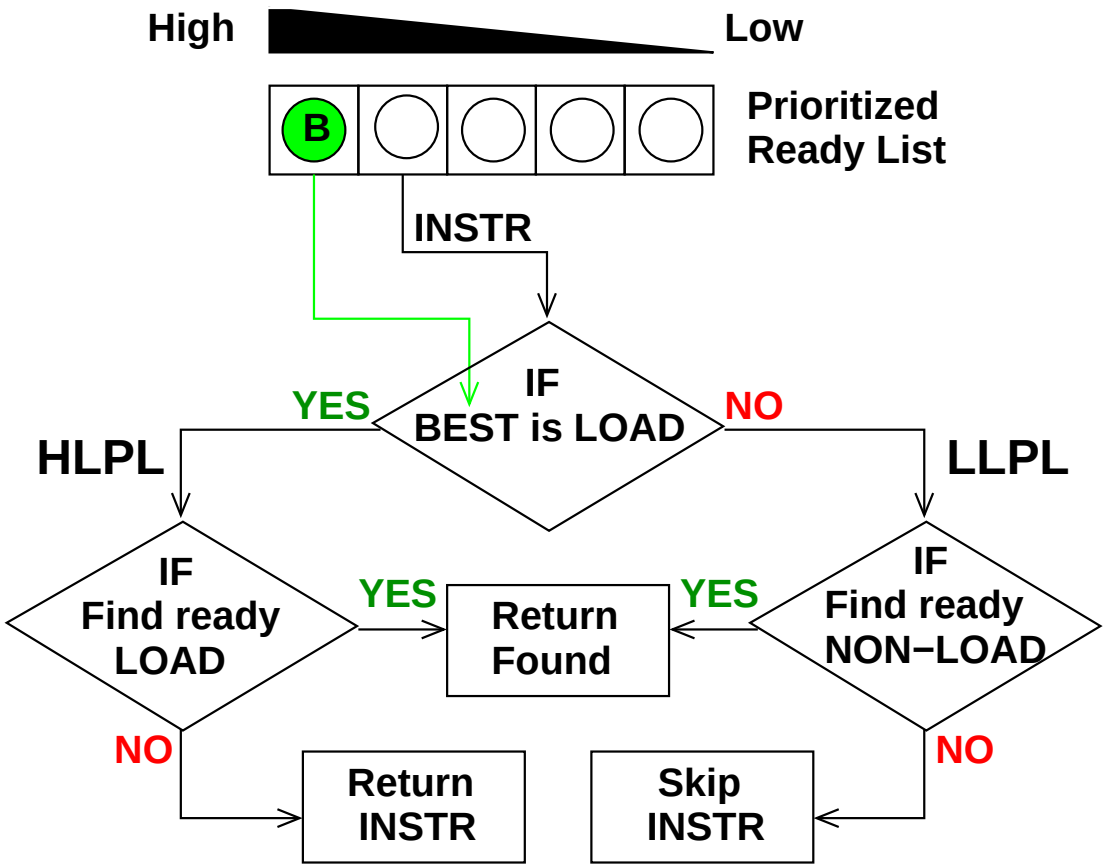
Alignment Heuristic

- Choose between HLPL and LLPL depending on highest priority instruction in the ready list (B)



Alignment Heuristic

- Choose between HLPL and LLPL depending on highest priority instruction in the ready list (B)



Outline

Introduction

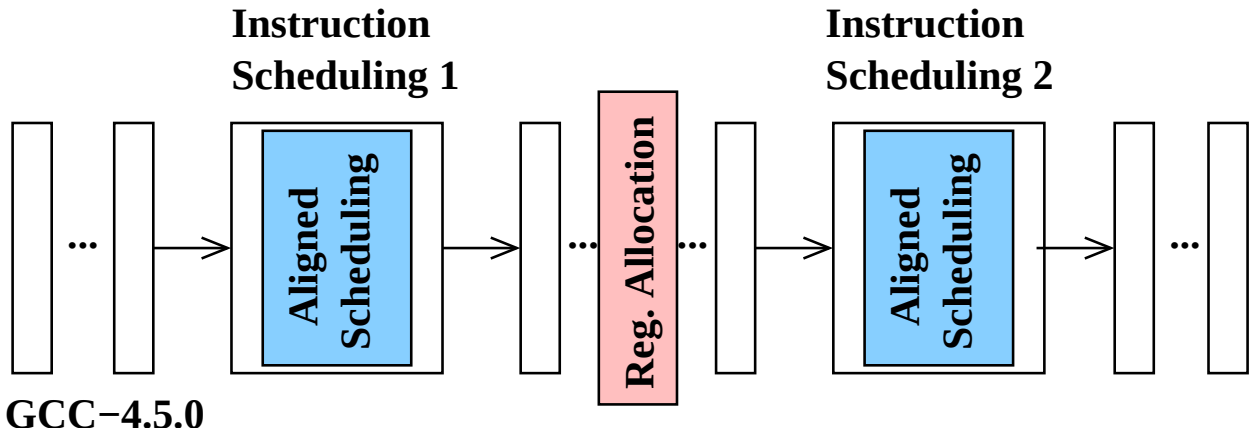
Aligned Scheduling

Experimental Setup and Results

Conclusion

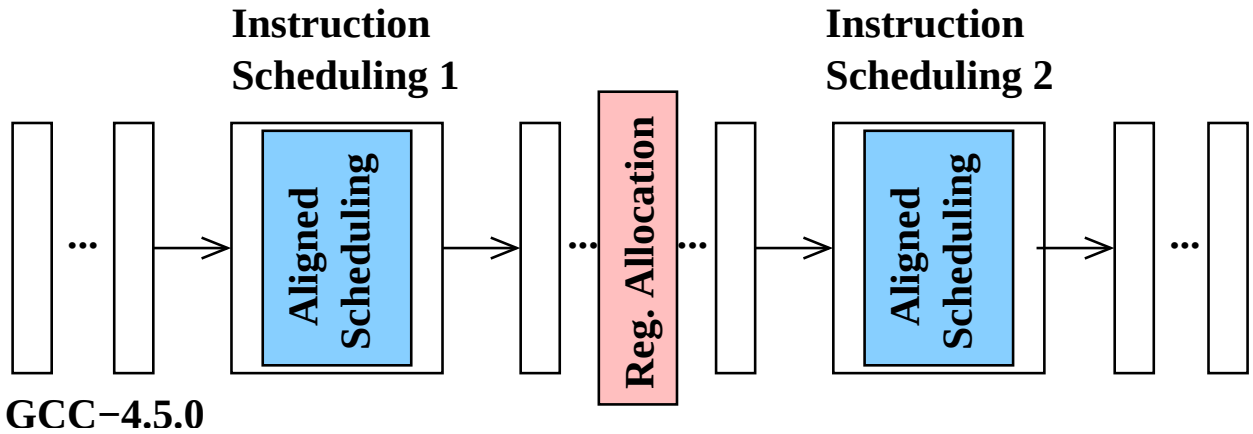
Experimental Setup

- Compiler: GCC-4.5.0, Modified Haifa-Scheduler



Experimental Setup

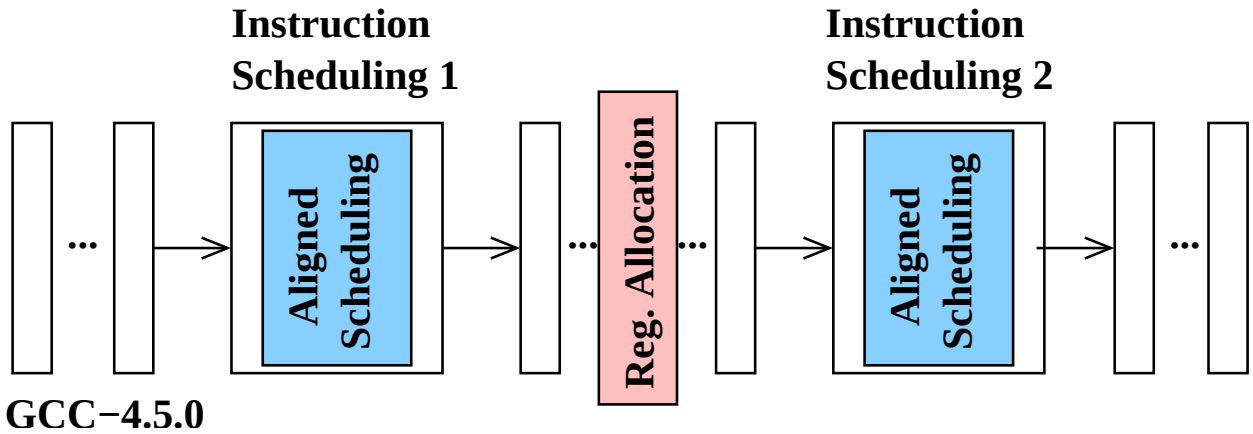
- Compiler: GCC-4.5.0, Modified Haifa-Scheduler



- Architecture
  - IA64-based 4 issue VLIW
  - Modified SKI IA64 simulator

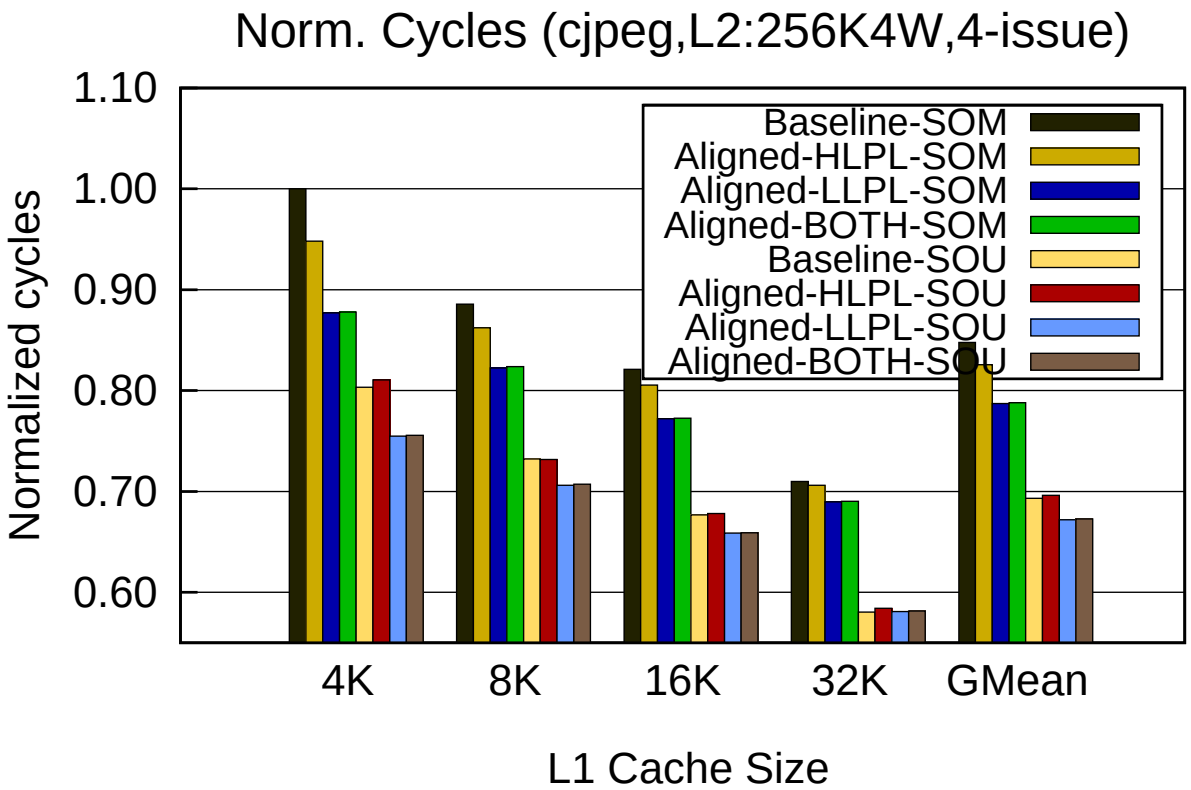
Experimental Setup

- Compiler: GCC-4.5.0, Modified Haifa-Scheduler



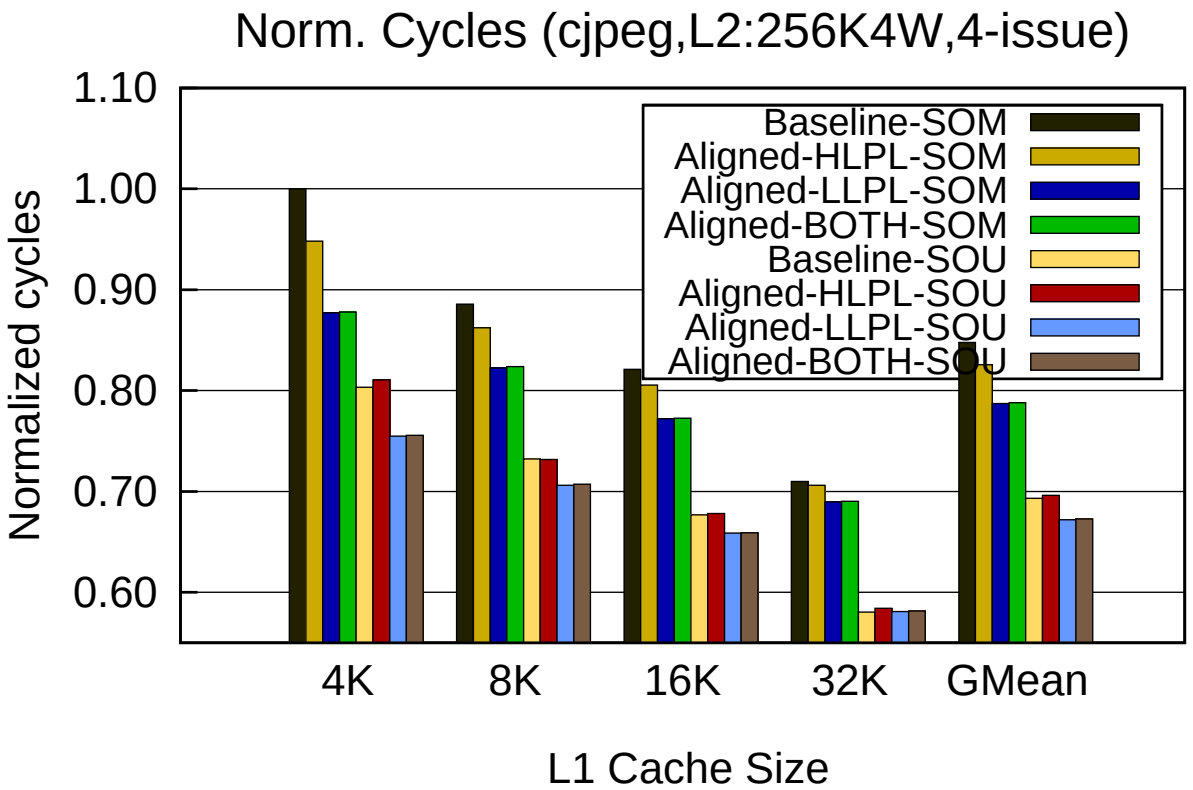
- Architecture
  - IA64-based 4 issue VLIW
  - Modified SKI IA64 simulator
- Benchmarks: SPEC CINT2000 and MediabenchII Video

Results: cjpeg (4-issue), Cycles



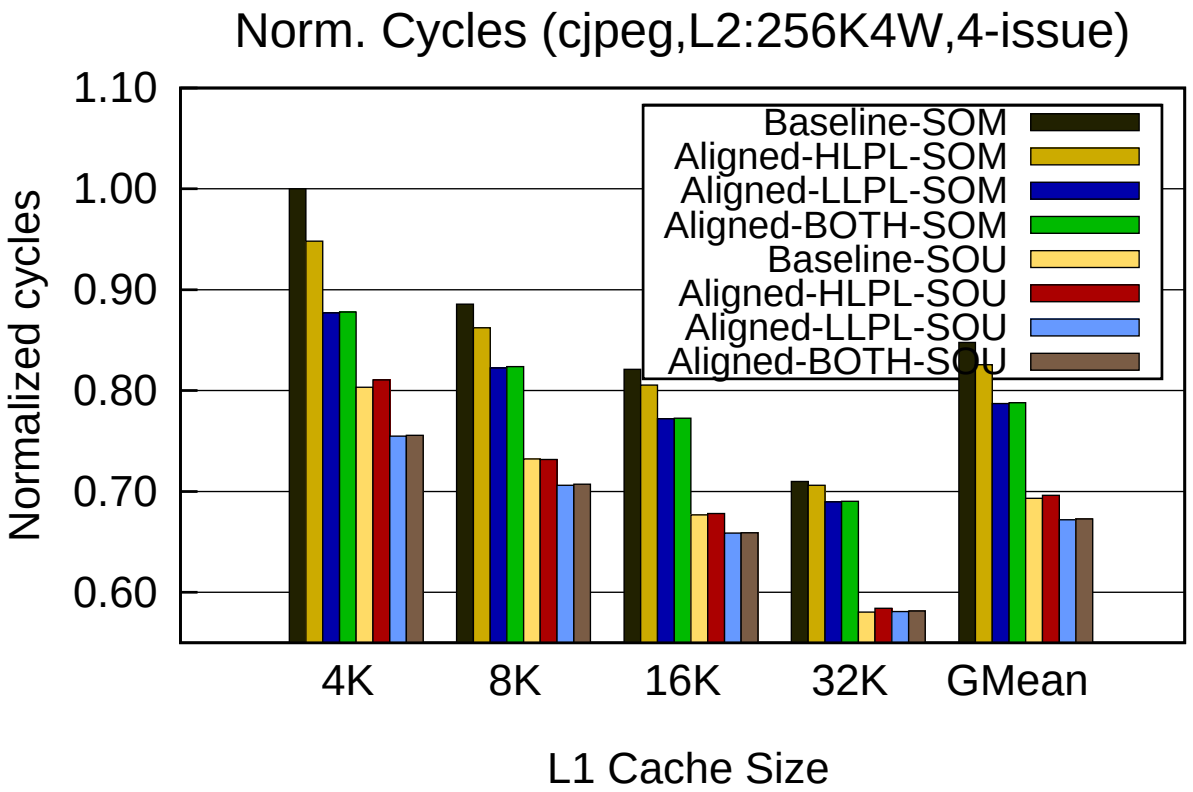
- Big improvement for small L1

Results: cjpeg (4-issue), Cycles



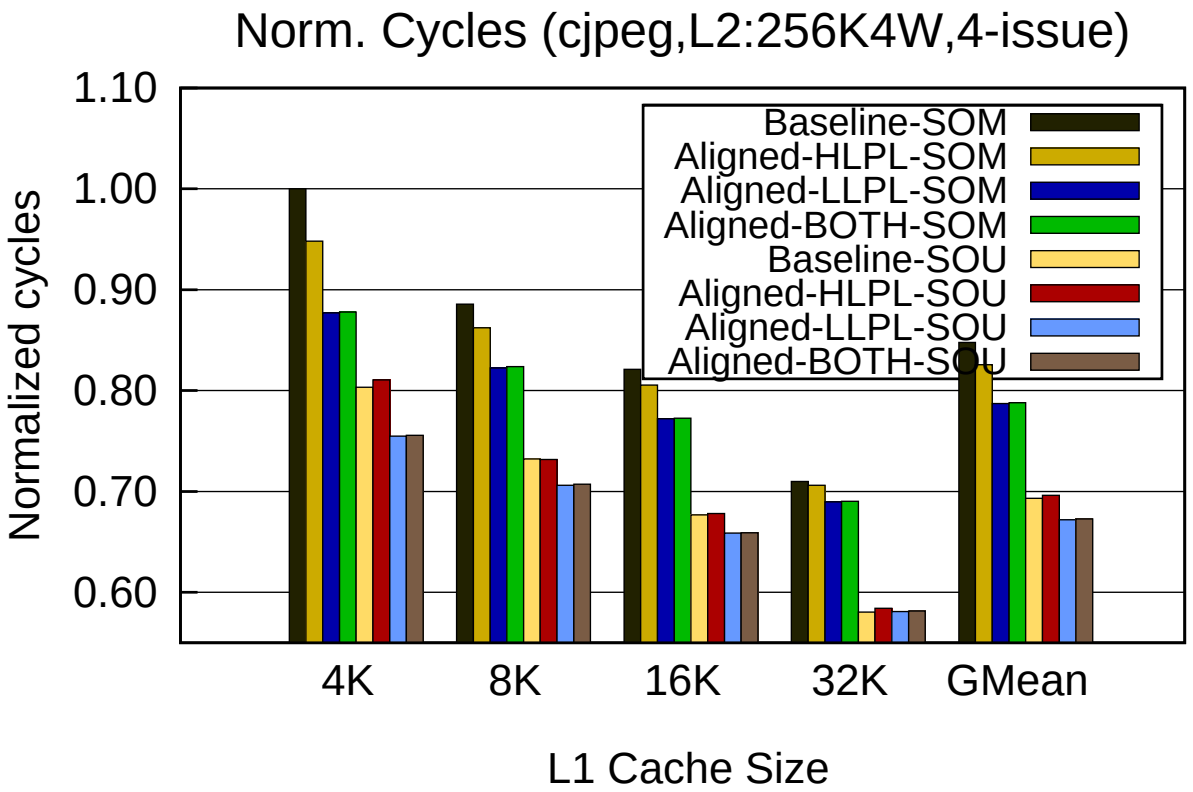
- Big improvement for small L1
- HLPL and LLPL act cooperatively

Results: cjpeg (4-issue), Cycles



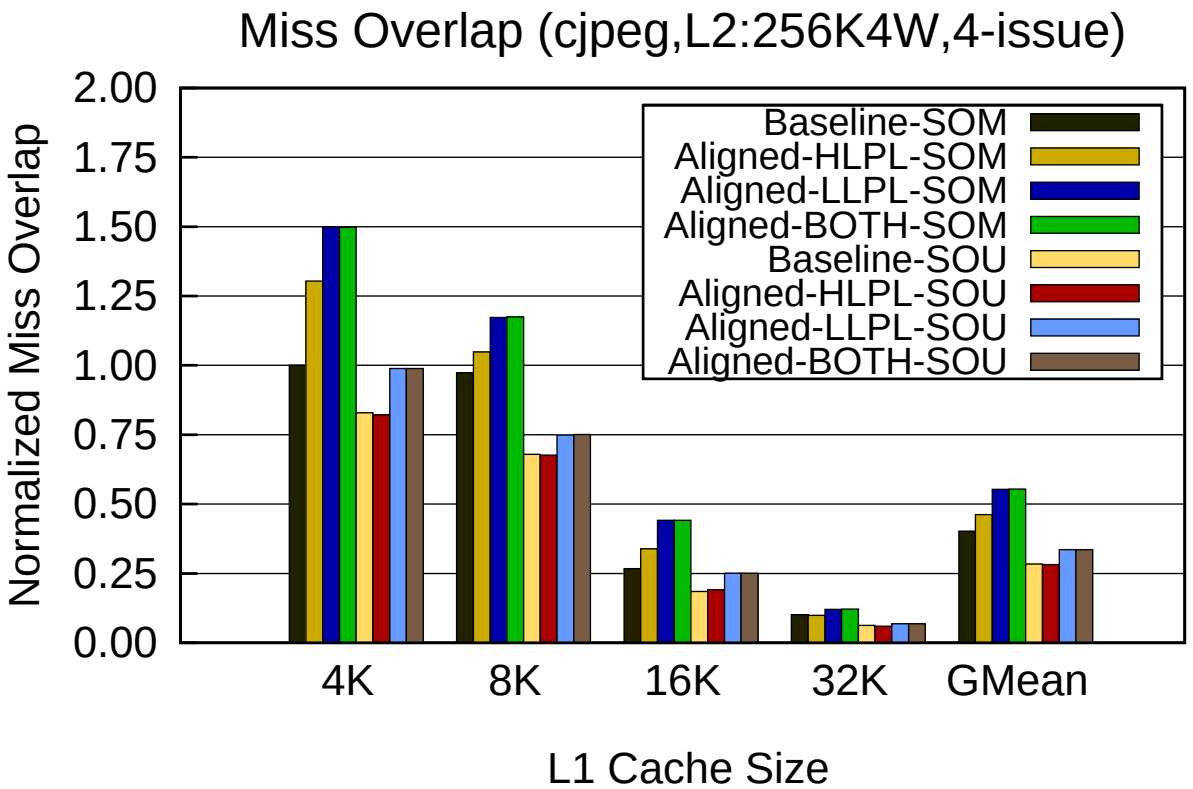
- Big improvement for small L1
- HLPL and LLPL act cooperatively
- Improves SOM, performance close to SOM on double the cache.

Results: cjpeg (4-issue), Cycles



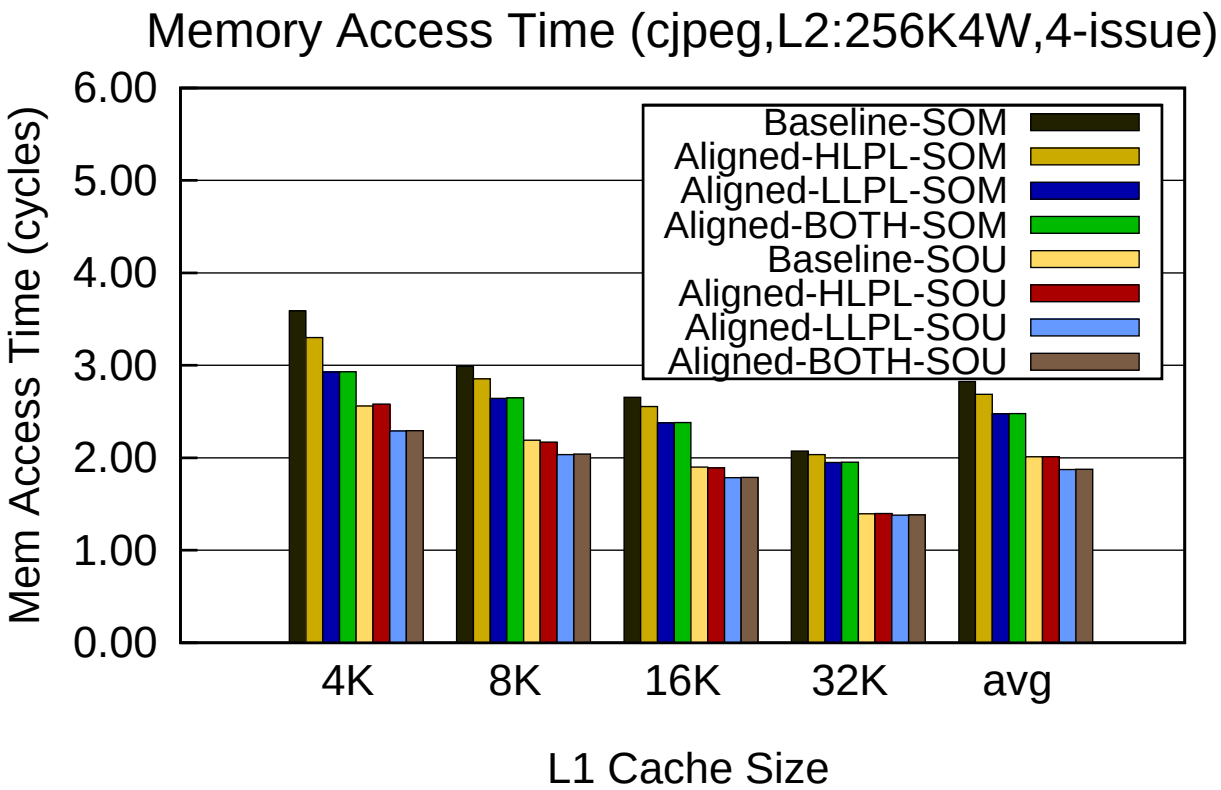
- Big improvement for small L1
- HLPL and LLPL act cooperatively
- Improves SOM, performance close to SOM on double the cache.
- Improves SOU

Results: cjpeg (4-issue), Miss-Overlap



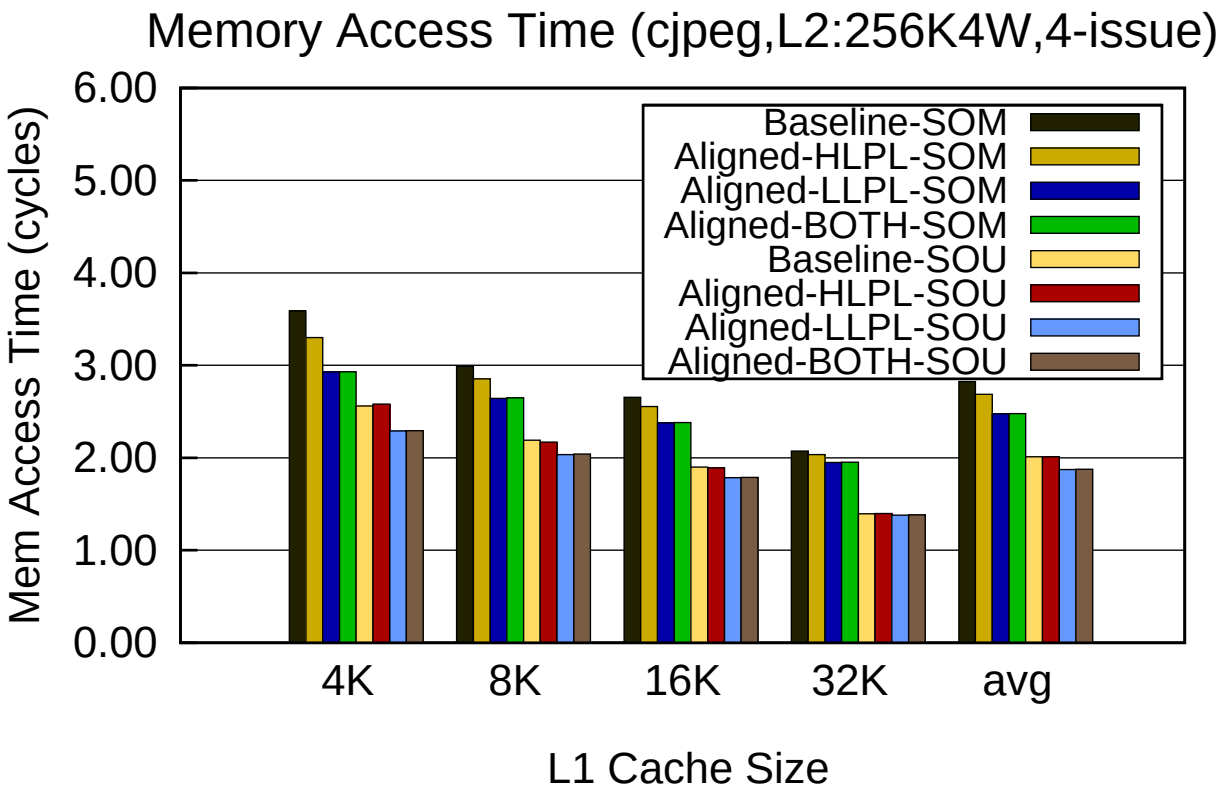
- Increase Miss overlapping, proof that Aligned Scheduling achieves its goal

Results: cjpeg (4-issue), Mem Access Time



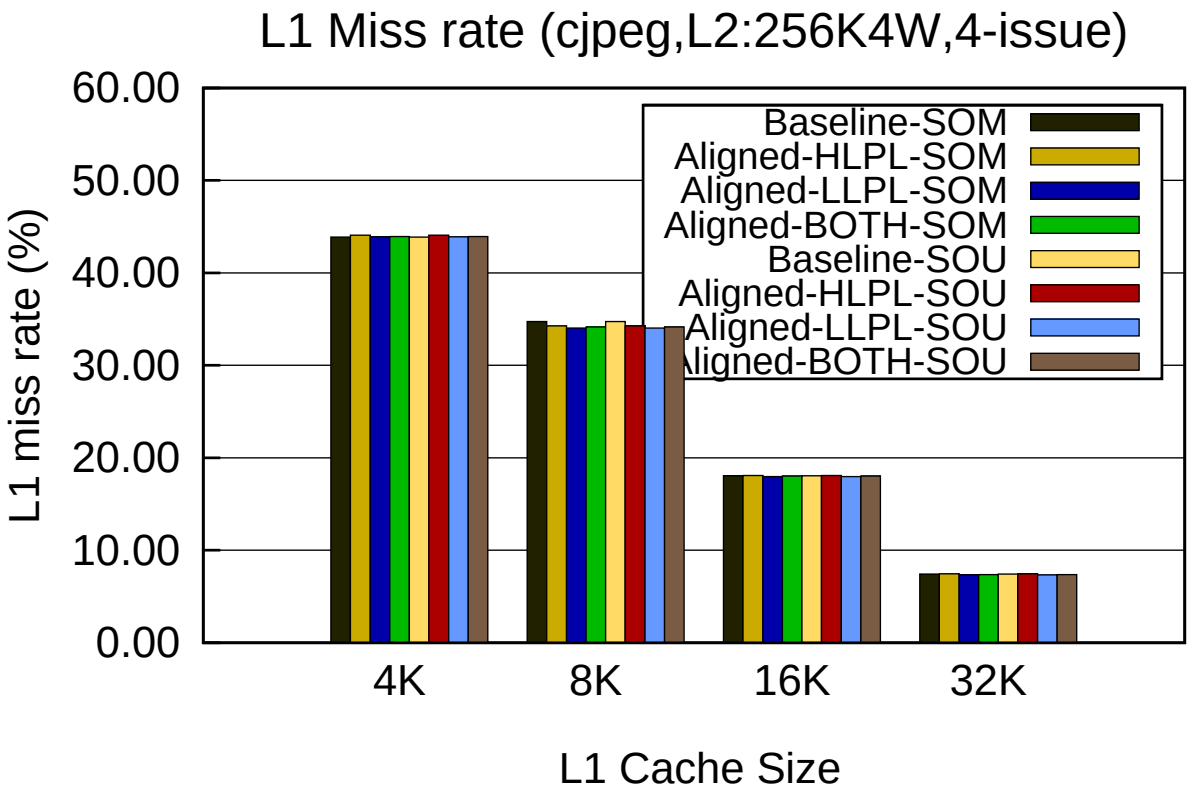
- Aligned scheduling improves the average memory access time

Results: cjpeg (4-issue), Mem Access Time



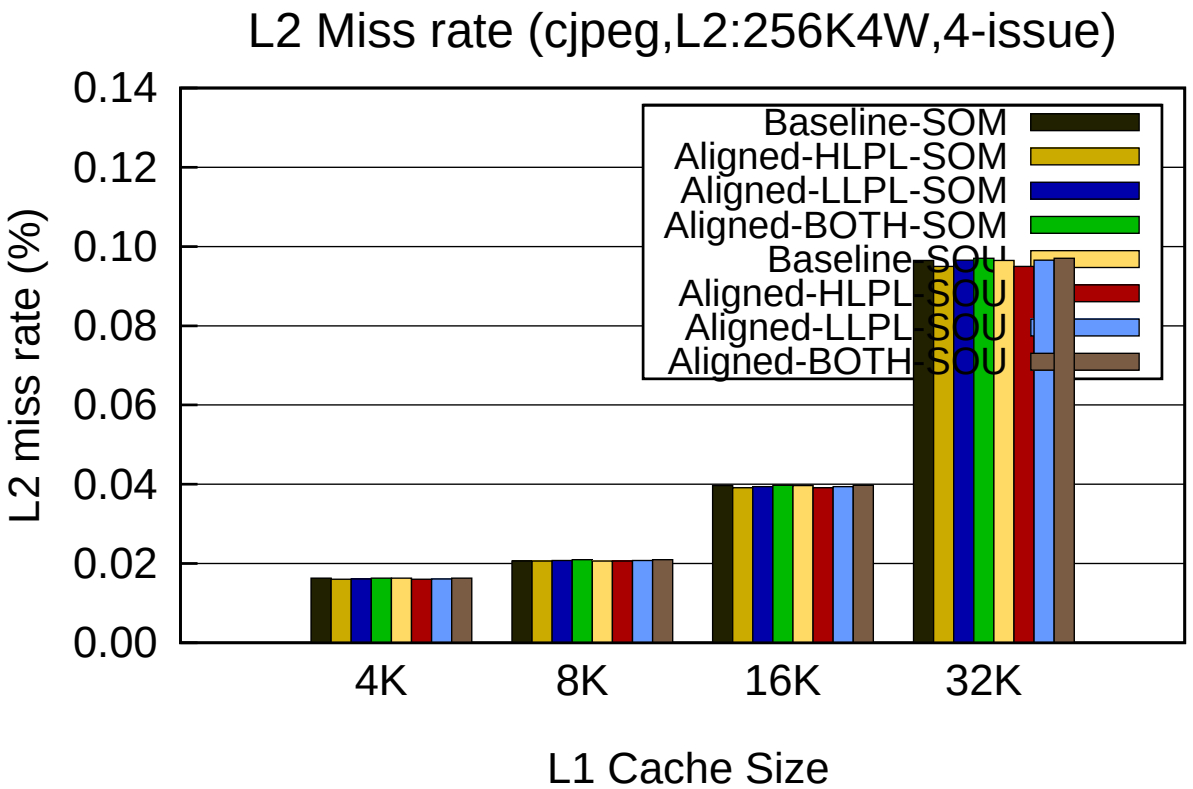
- Aligned scheduling improves the average memory access time
- Proof that Aligned Scheduling helps hide cache misses

Results: cjpeg (4-issue), Miss Rates



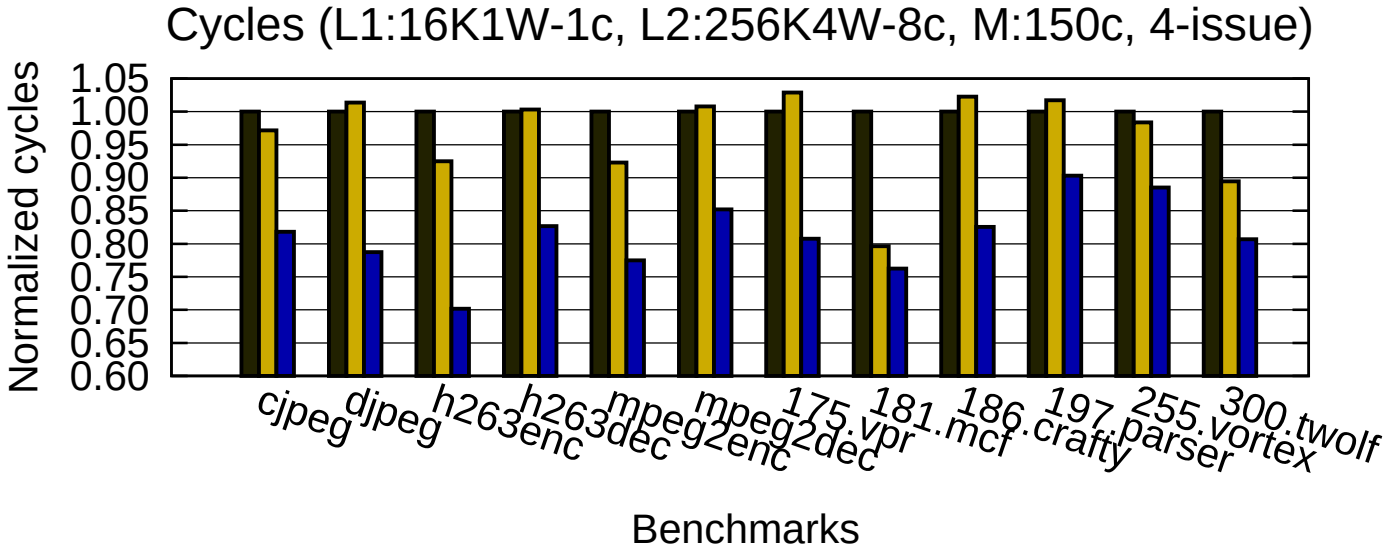
- Miss Rates don't change
- A miss is count as a miss even if it overlaps

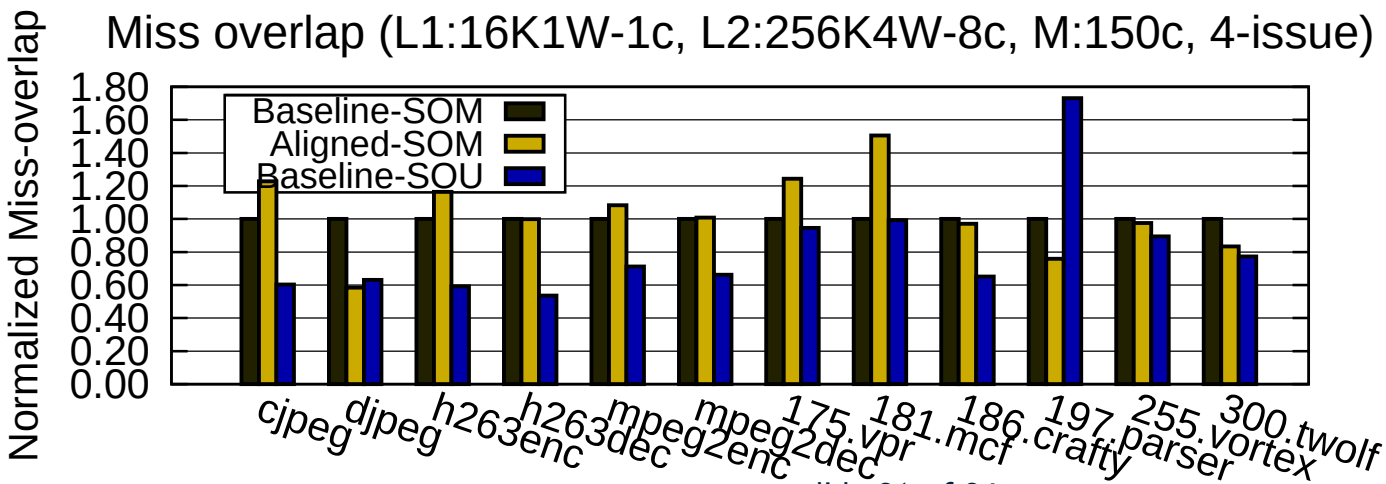
Results: cjpeg (4-issue), Miss Rates



- Miss Rates don't change
- A miss is count as a miss even if it overlaps

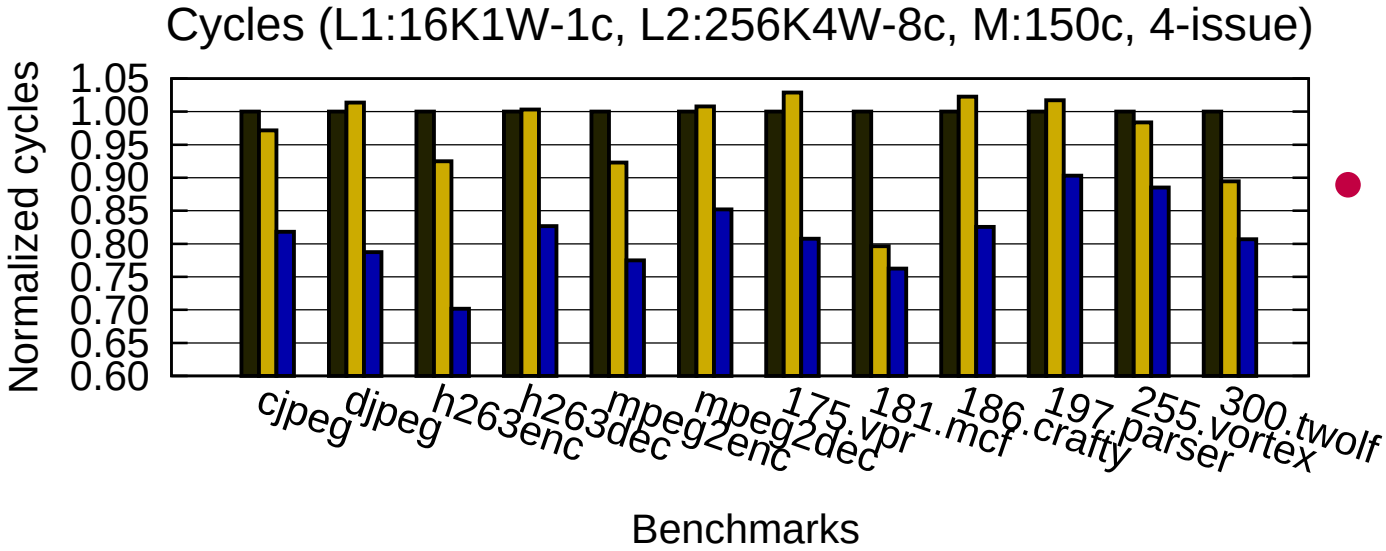
Results: All Benchmarks, Performance

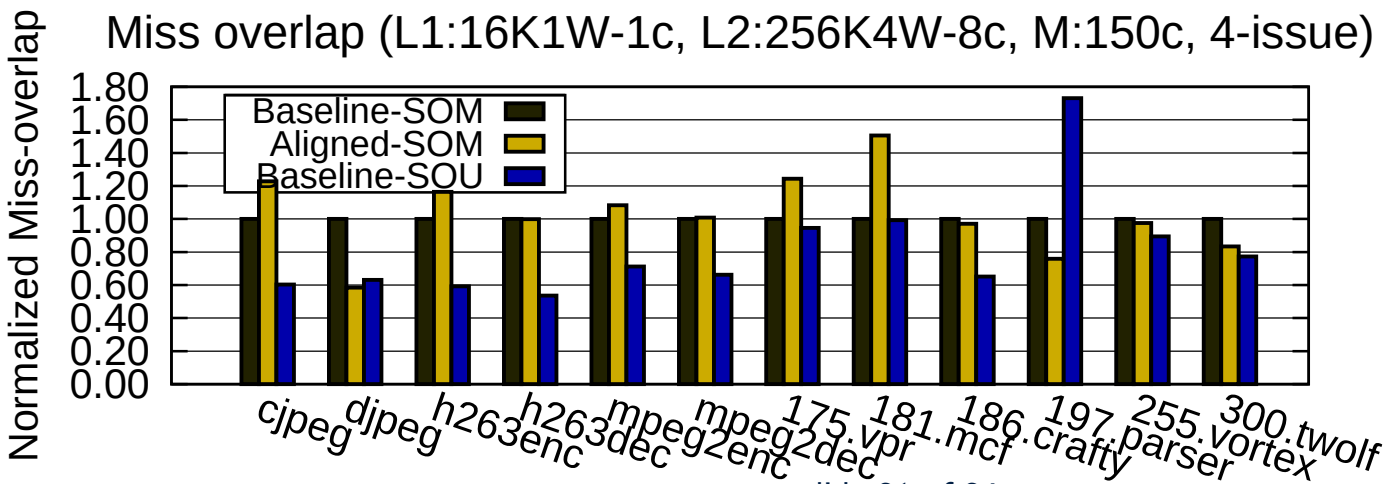




- Improves SOM on avg

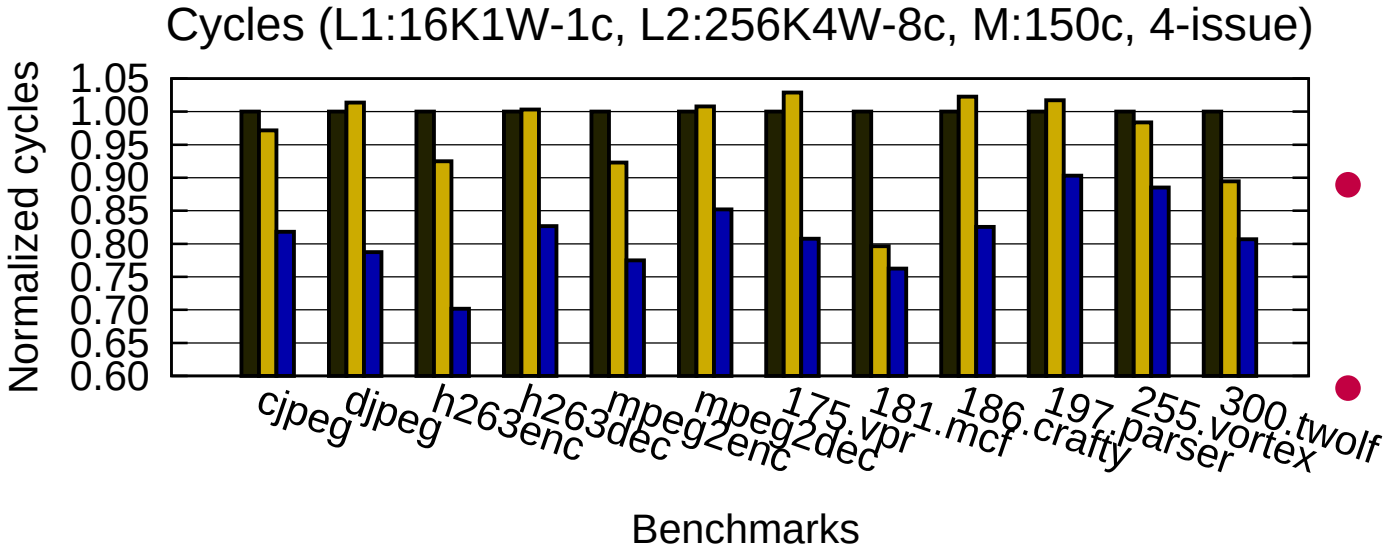
Results: All Benchmarks, Performance



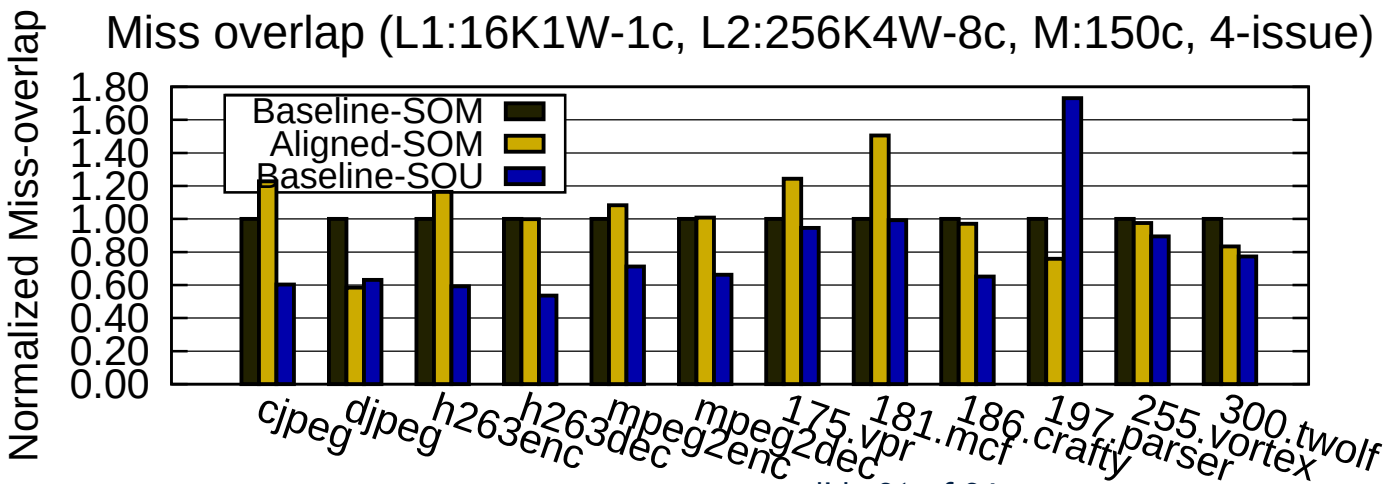


- Improves SOM on avg
- Up to 20% speedup

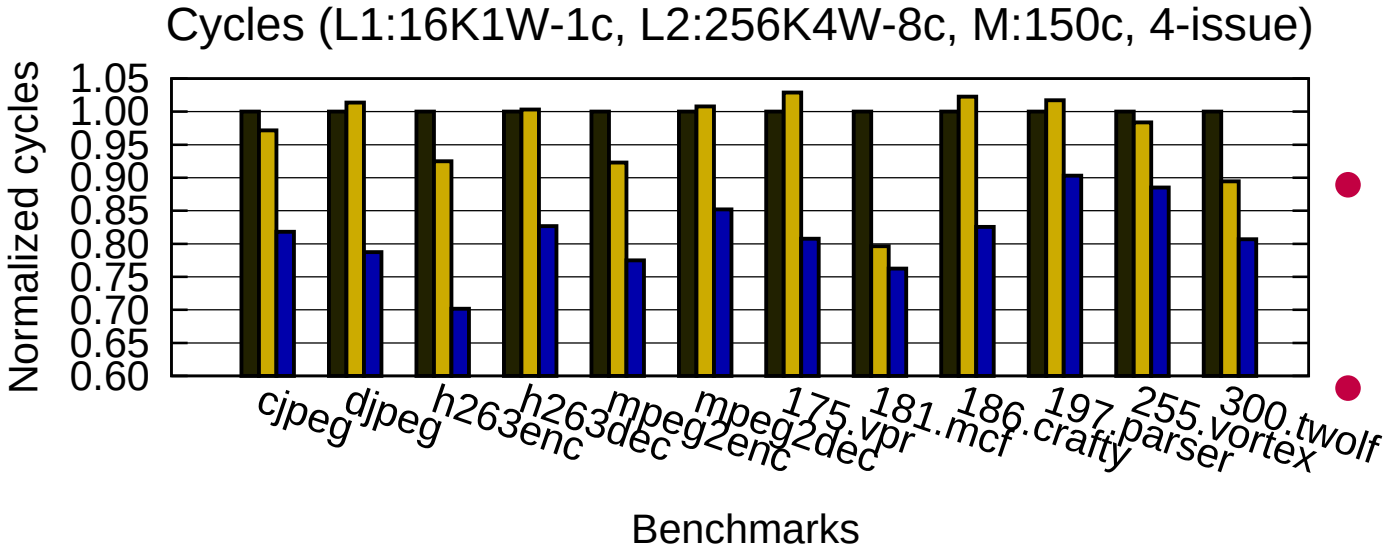
Results: All Benchmarks, Performance

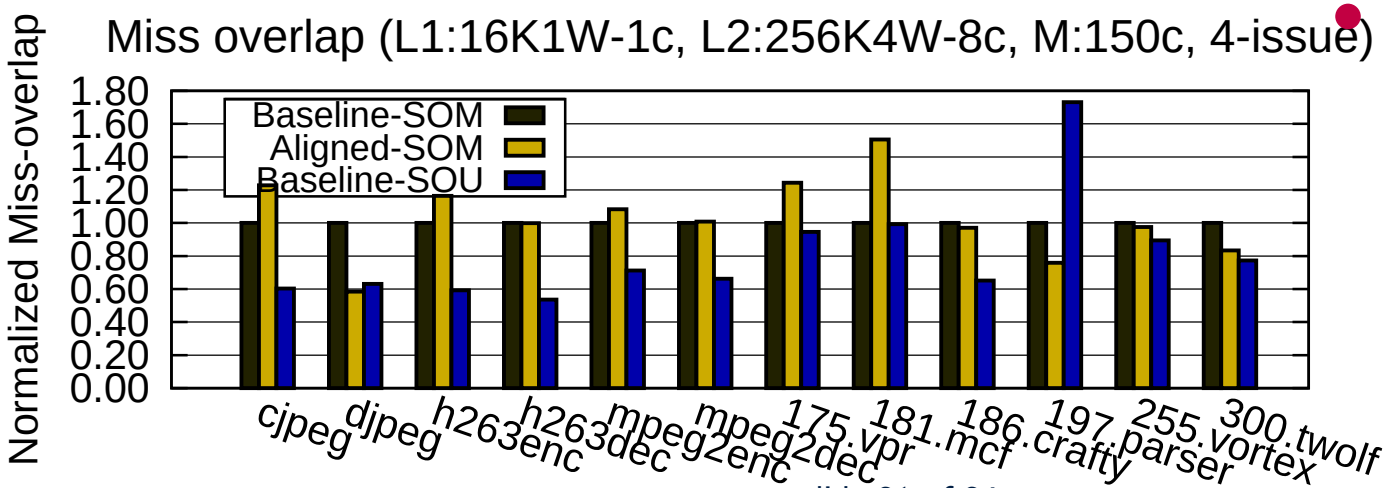


- Improves SOM on avg
- Up to 20% speedup
- Up to 3% slowdown



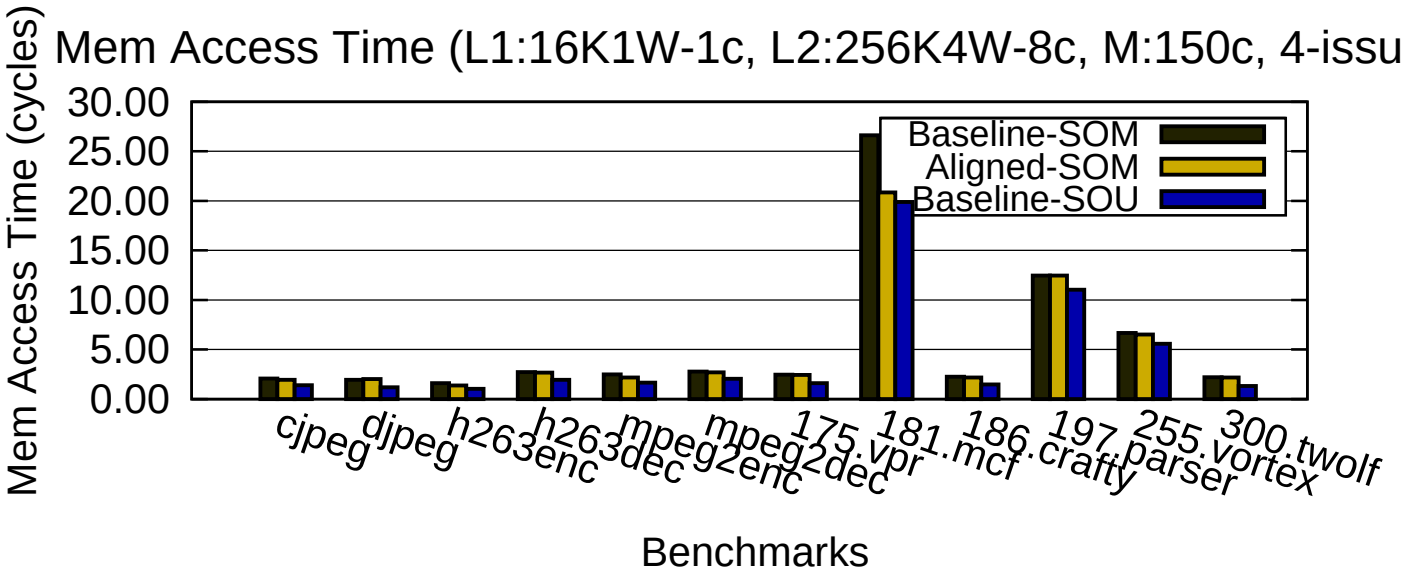
Results: All Benchmarks, Performance





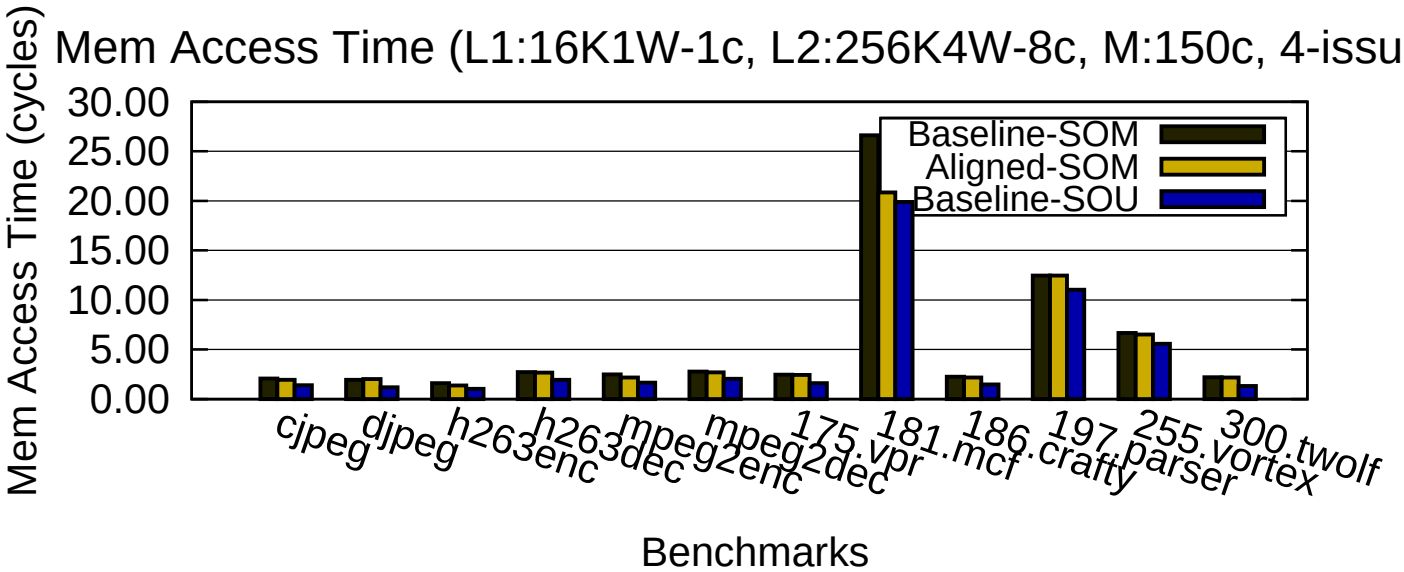
- Improves SOM on avg
- Up to 20% speedup
- Up to 3% slowdown
- Correlation between Miss overlapping and performance

Results: All Benchmarks, Mem Access Time



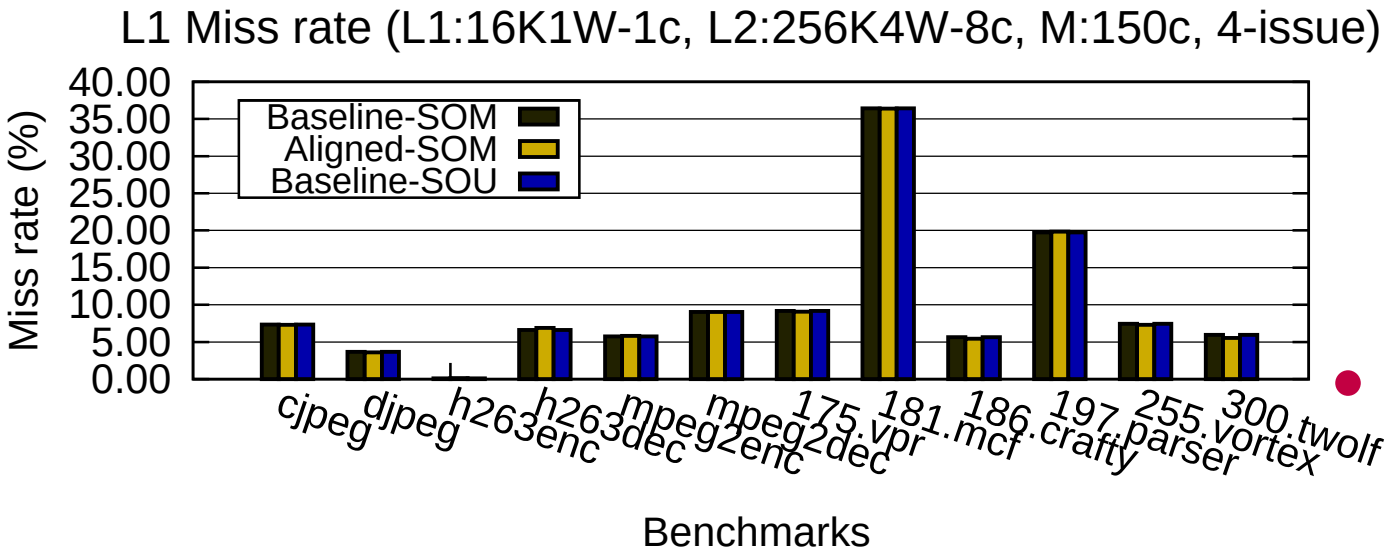
- Improved memory access latency

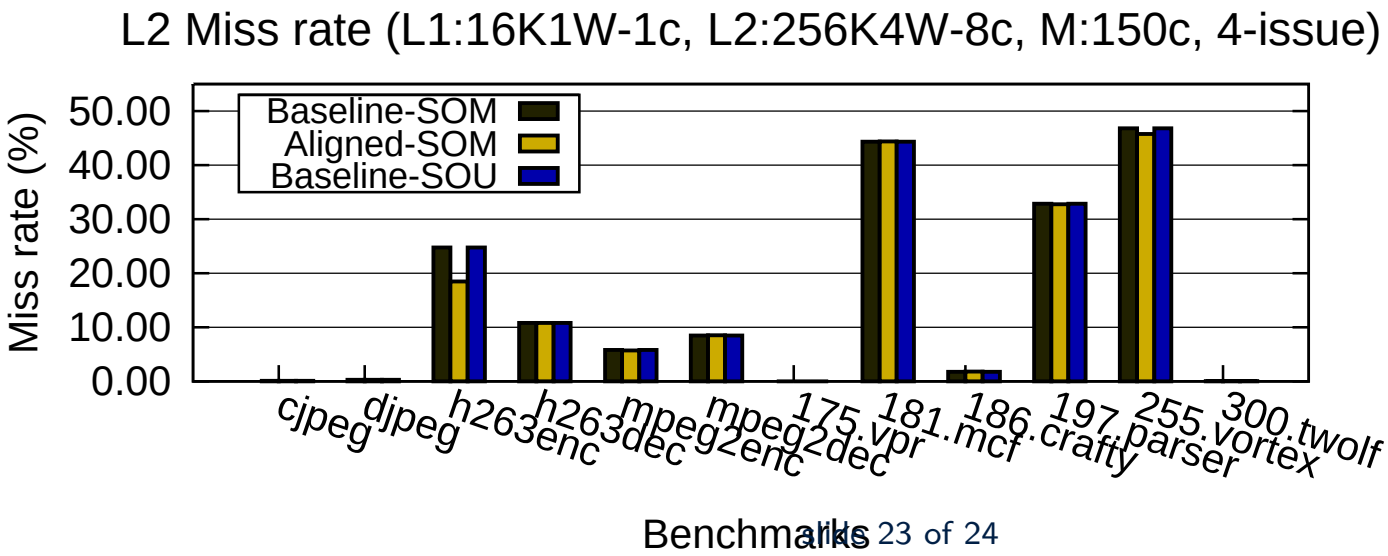
Results: All Benchmarks, Mem Access Time



- Improved memory access latency
- SOU consistently better at hiding latencies (hardware support)

Results: All Benchmarks, Miss Rates





- Miss rates remain largely unchanged

Conclusion

Proposed Aligned Scheduling, a scheduler for VLIWs that:

- Incorporates micro-architectural knowledge of load-use interlocking hardware
- Exploits statically known MLP
- Generates schedules resilient to cache misses on VLIWs

Aligned Scheduling: Cache-efficient Instruction Scheduling for VLIW Processors

Vasileios Porpodas [†] and Marcelo Cintra [†⋆]

School of Informatics, University of Edinburgh[†]
Intel Labs Braunschweig[⋆]

LCPC 2013