Runahead Execution

An Alternative to Very Large Instruction Windows for Out-of-order Processors

Onur Mutlu
Yale N. Patt

Jared Stark
Chris Wilkerson

Outline

- Motivation
- Overview
- Mechanism
- Experimental Evaluation
- Conclusions

Motivation

- Out-of-order processors require very large instruction windows to tolerate today’s main memory latencies.
  - Even in the presence of caches and prefetchers

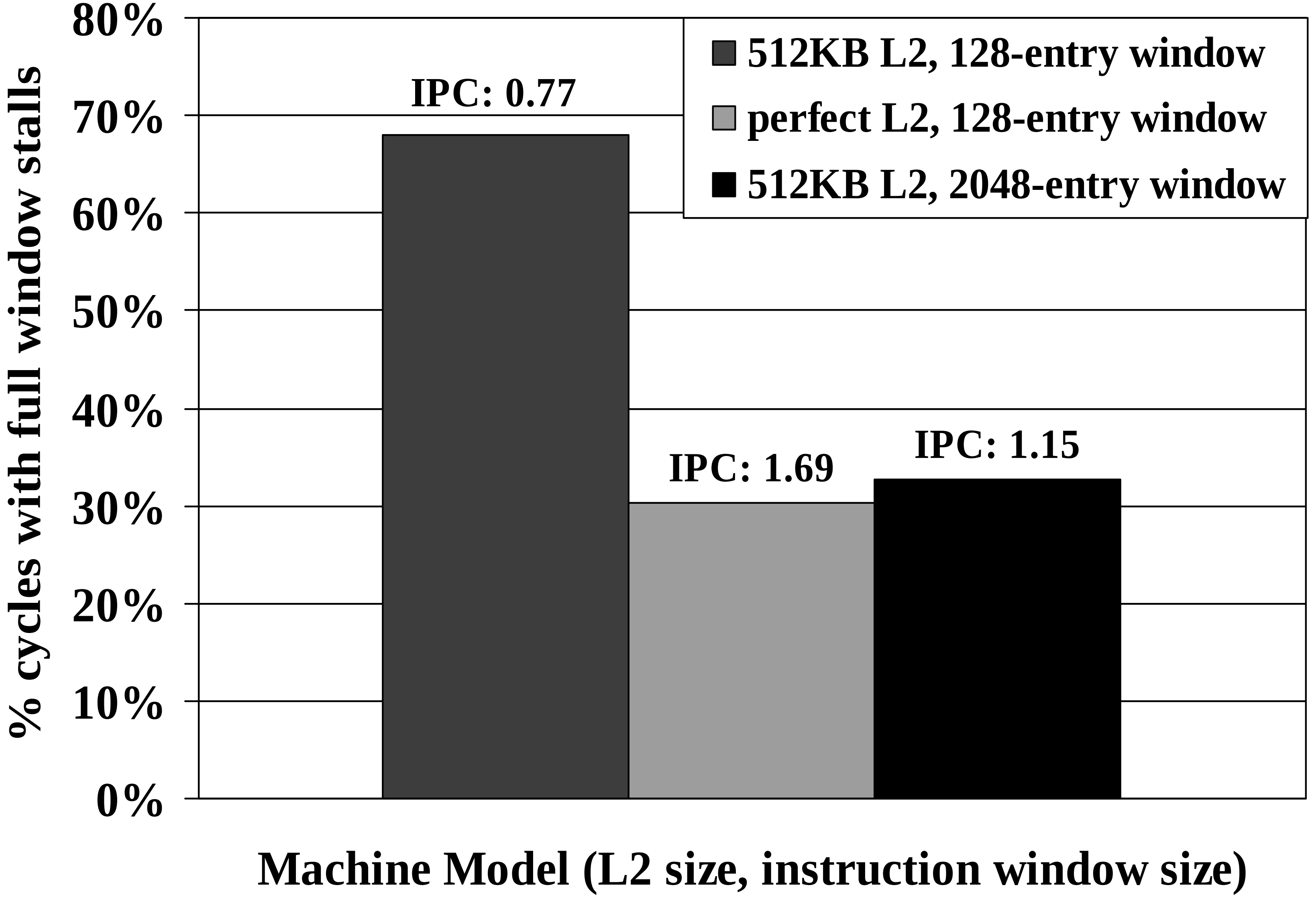
- As main memory latency (in terms of processor cycles) increases, instruction window size should also increase to fully tolerate the memory latency.

- Building a very large instruction window is not an easy task.

Small Windows: Full-window Stalls

- Instructions are retired in-order from the instruction window to support precise exceptions.
- When a very long-latency instruction is not complete, it blocks retirement and incoming instructions fill the instruction window if the window is not large enough.
- Processor cannot place new instructions into the window if the window is already full. This is called a full-window stall.
- L2 misses are responsible for most full-window stalls.

Impact of Full-window Stalls



Overview of Runahead Execution

- During a significant percentage of full-window stall cycles, no work is performed in the processor.
- Runahead execution unblocks the full window stall caused by a long-latency L2-miss instruction.
- Enter *runahead mode* when the oldest instruction is an L2-miss load and remove that load from the processor.
- While in runahead mode, keep processing instructions without updating architectural state and without blocking the instruction window due to L2 misses.
- When the original load miss returns back, resume normal-mode execution starting with the runahead-causing load.

Benefits of Runahead Execution

- Loads and stores independent of L2-miss instructions generate useful prefetch requests:
  - From main memory to L2
  - From L2 to L1

- Instructions on the predicted program path are prefetched into the trace cache and L2.

- Hardware prefetcher tables are trained using future memory access information. The prefetcher also runs ahead along with the processor.

Mechanism

Entry into Runahead Mode

When an L2-miss load instruction reaches the head of the instruction window:

- Processor checkpoints architectural register state, branch history register, return address stack.
- Processor records the address of the L2-miss load.
- Processor enters *runahead mode*.
- L2-miss load marks its destination register as *invalid* and is removed from the instruction window.

Processing in Runahead Mode

- Two types of results are produced: INV (invalid), VALID
- First INV result is produced by the L2-miss load that caused entry into runahead mode.
- An instruction produces an INV result
  - If it sources an INV result
  - If it misses in the L2 cache (A prefetch request is generated)
- INV results are marked using INV bits in the register file, store buffer, and runahead cache.
  - INV bits prevent introduction of bogus data into the pipeline.
  - Bogus values are not used for prefetching/branch resolution.

Pseudo-retirement in Runahead Mode

- An instruction is examined for pseudo-retirement when it reaches the head of the instruction window.
- An INV instruction is removed from window immediately.
- A VALID instruction is removed when it completes execution and updates only the microarchitectural state.
- Pseudo-retired instructions free their allocated resources.
- Pseudo-retired runahead stores communicate their data and INV status to dependent runahead loads.

Runahead Cache

- An auxiliary structure that holds the data values and INV bits for memory locations modified by pseudo-retired runahead stores.

- Its purpose is memory communication during runahead mode.

- Runahead loads access store buffer, runahead cache, and L1 data cache in parallel.

- Size of runahead cache is very small (512 bytes).

Runahead Branches

- Runahead branches use the same predictor as normal branches.
- VALID branches are resolved and trigger recovery if mispredicted.
- INV branches cannot be resolved.

Exit from Runahead Mode

When the data for the instruction that caused entry into runahead returns from main memory:

- All instructions in the machine are flushed.
- INV bits are reset. Runahead cache is flushed.
- Processor restores the state as it was before the runahead-inducing instruction was fetched.
- Processor starts fetch beginning with the runahead-inducing L2-miss instruction.

Experimental Evaluation

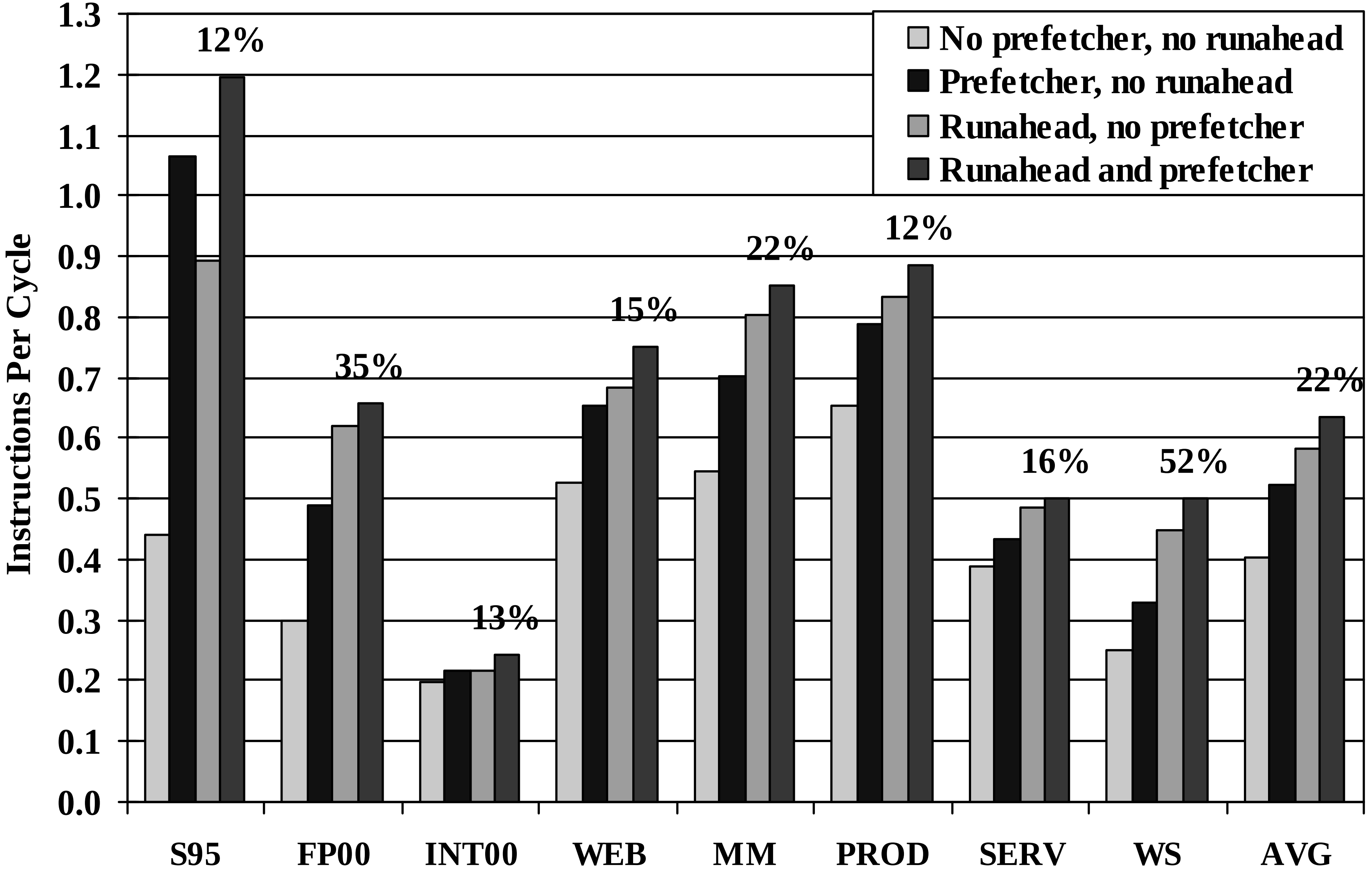
Baseline Processor

- 3-wide fetch, 29-stage pipeline
- 128-entry instruction window
- 32 KB, 8-way, 3-cycle L1 data cache, write-back
- 512 KB, 8-way, 16-cycle L2 unified cache, write-back
- Approximately 500-cycle penalty for L2 misses
- Streaming prefetcher (16 streams)

Benchmarks

- Selected traces out of a pool of 280 traces
- Evaluated performance on those that gain at least 10% IPC improvement with perfect L2 cache
- 147 traces simulated for 30 million x86 instructions
- Trace Suites
  - SPEC CPU95 (S95): 10 benchmarks, mostly FP
  - SPEC FP2k (FP00): 11 benchmarks
  - SPECint2k (INT00): 6 benchmarks
  - Internet (WEB): 18 benchmarks: SpecJbb, Webmark2001
  - Multimedia (MM): 9 benchmarks: mpeg, speech rec., quake
  - Productivity (PROD): 17 benchmarks: Sysmark2k, winstone
  - Server (SERV): 2 benchmarks: tpcc, timesten
  - Workstation (WS): 7 benchmarks: CAD, nastran, verilog

Performance of Runahead Execution



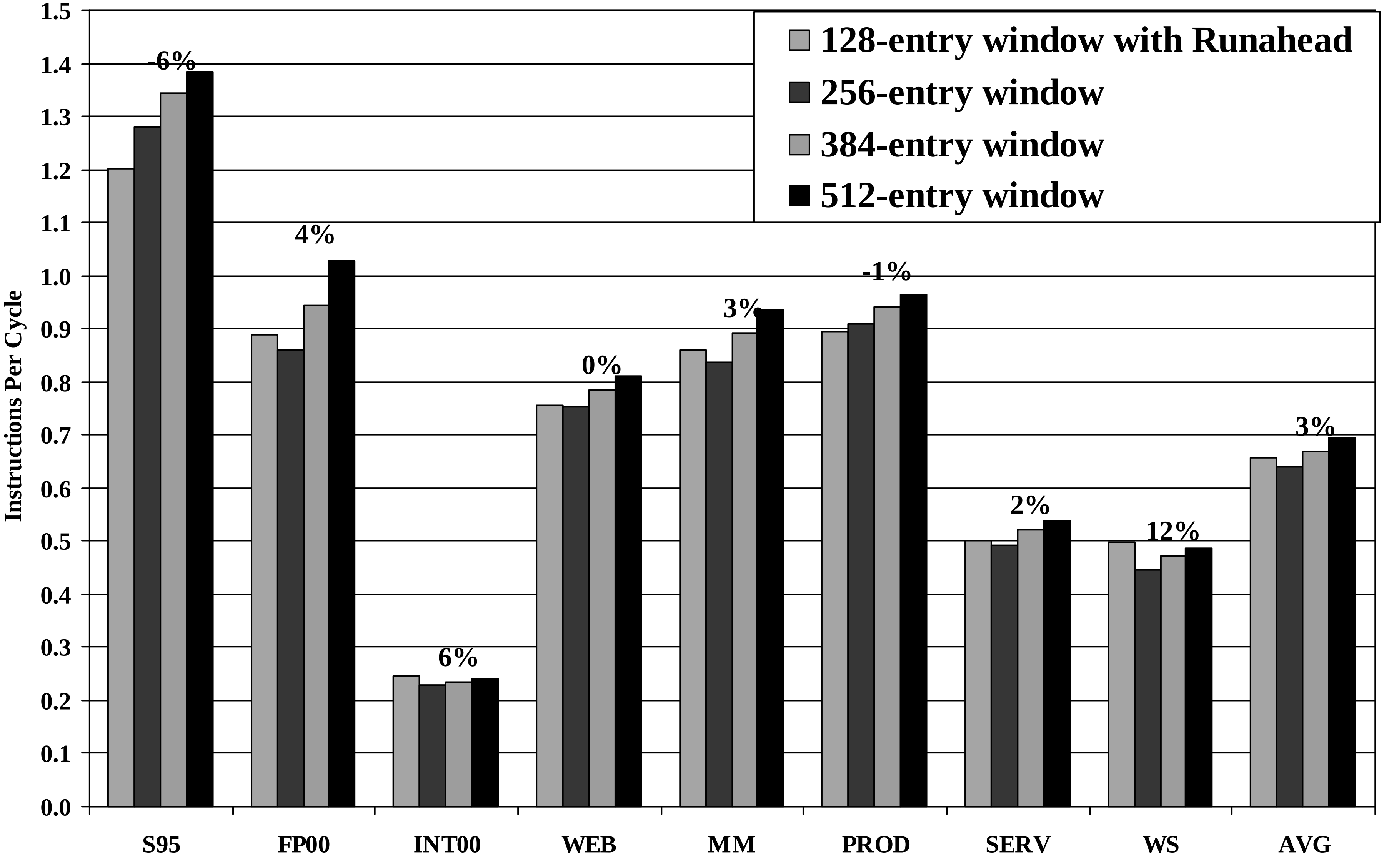
Effect of Frontend on Runahead

- Average number of instructions during runahead: 711
  - Before mispredicted INV branch: 431

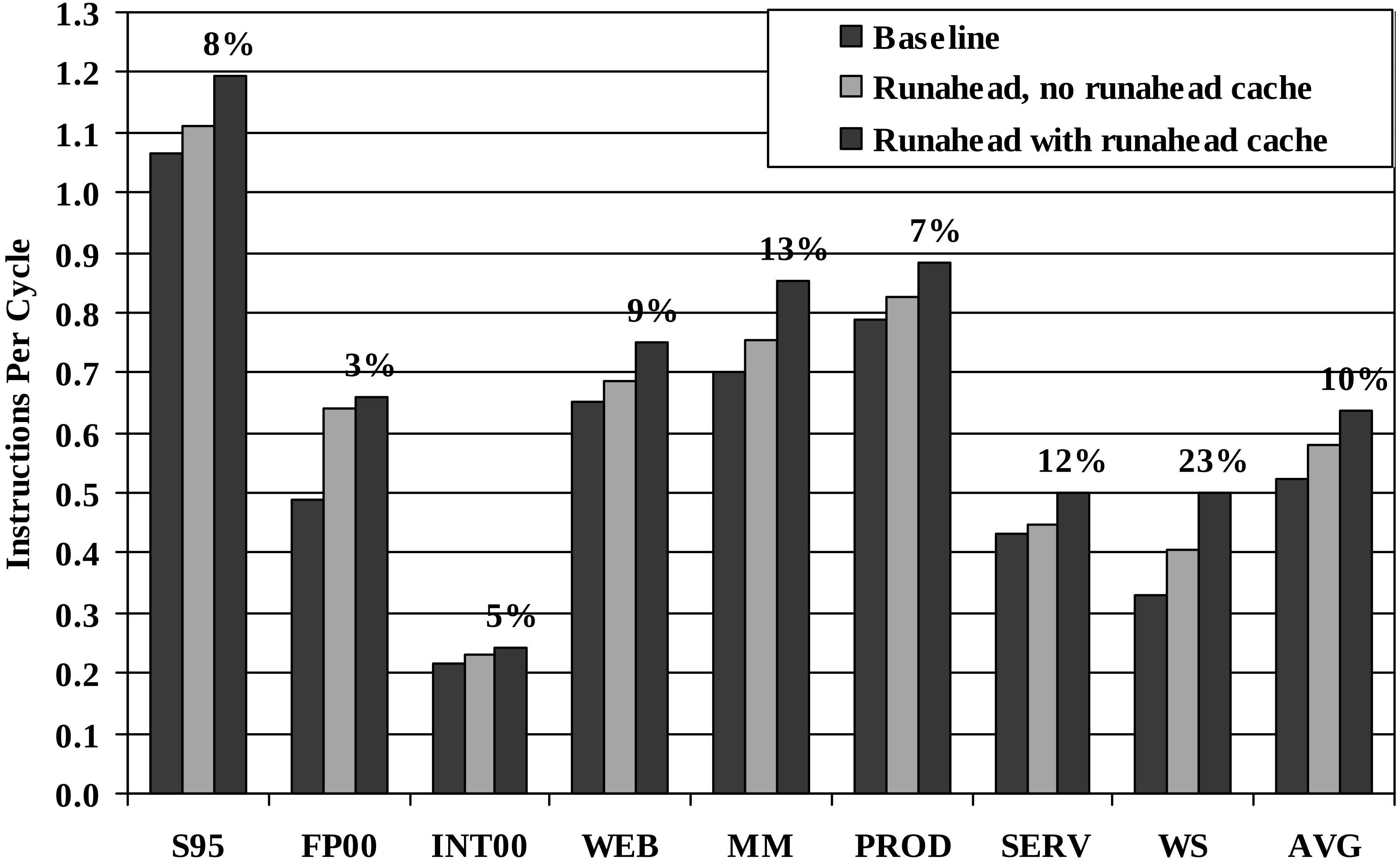
- Average number of L2 misses during runahead: 2.6
  - Before mispredicted INV branch: 2.38

- Runahead becomes more effective with a better frontend:
  - Real trace cache: 22% IPC improvement
  - Perfect trace cache: 27% IPC improvement
  - Perfect branch predictor and trace cache: 31% IPC improvement

Runahead vs. Large Windows



Importance of Store-Load Data Communication



Conclusions

- Runahead execution results in 22% IPC increase over the baseline processor with a 128-entry window and a streaming prefetcher.

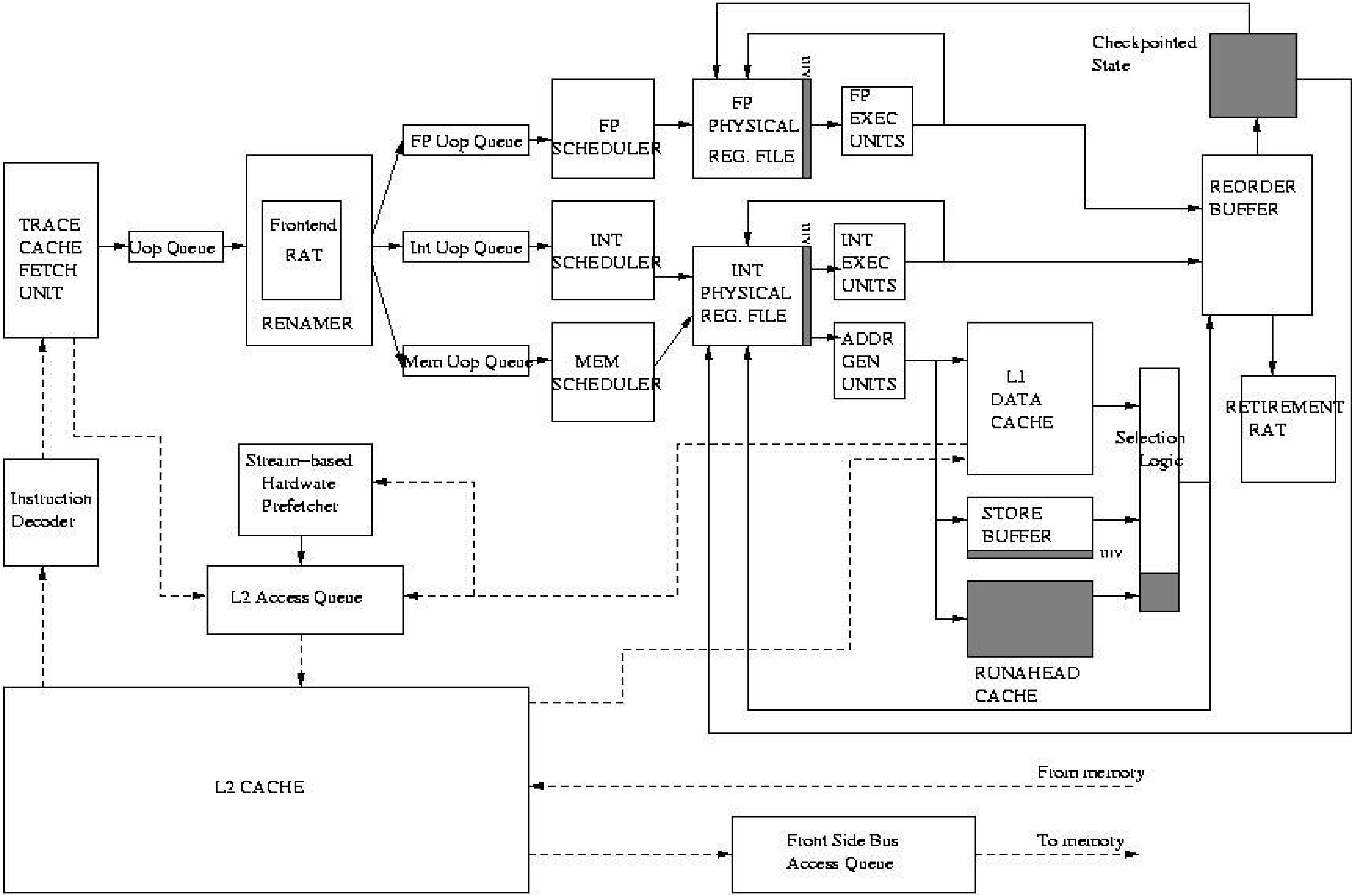
- This is within 1% of the IPC of a 384-entry window machine.

- Runahead and the streaming prefetcher interact positively.

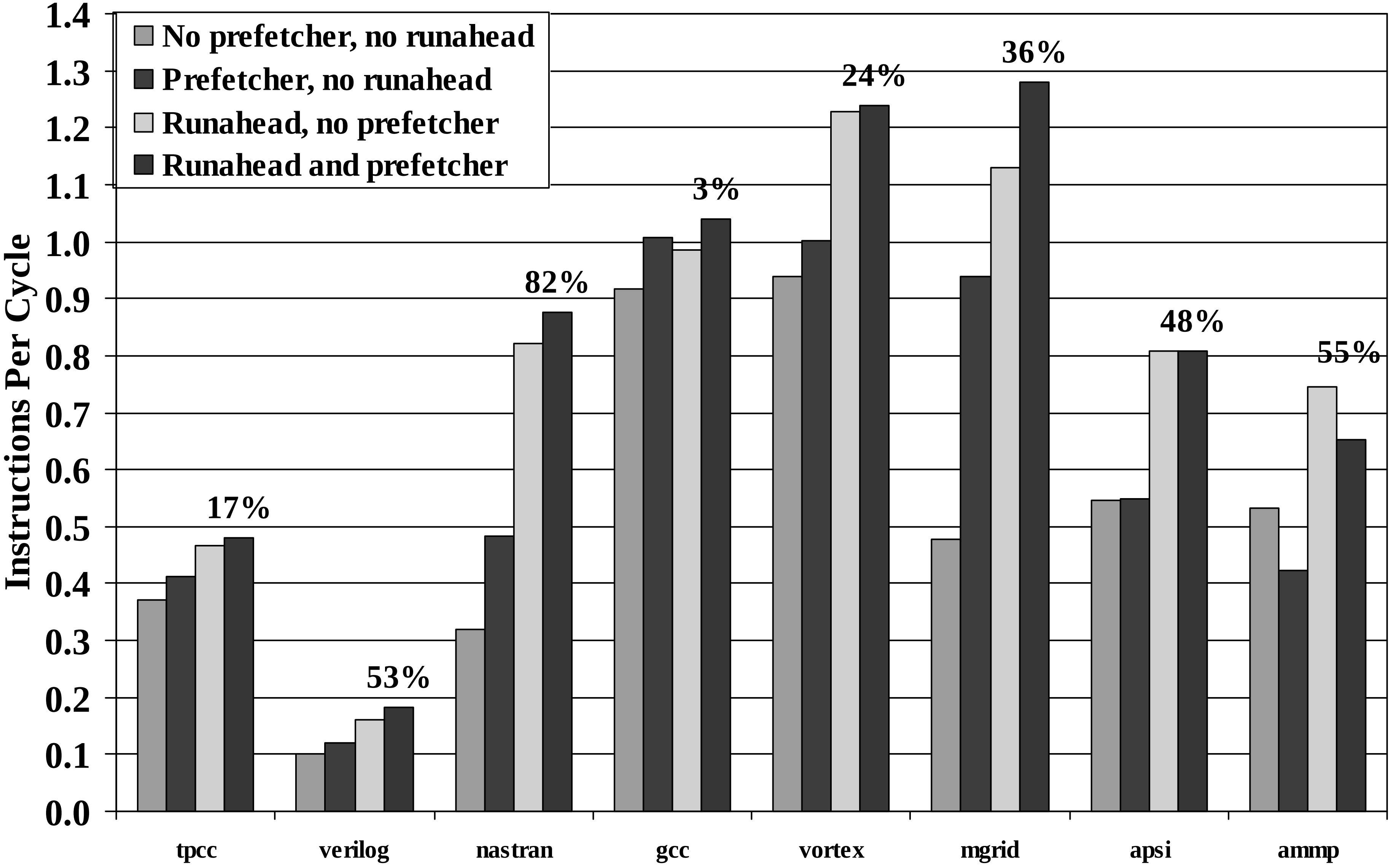
- Store-load data communication through memory in runahead mode is vital for high performance.

Backup Slides

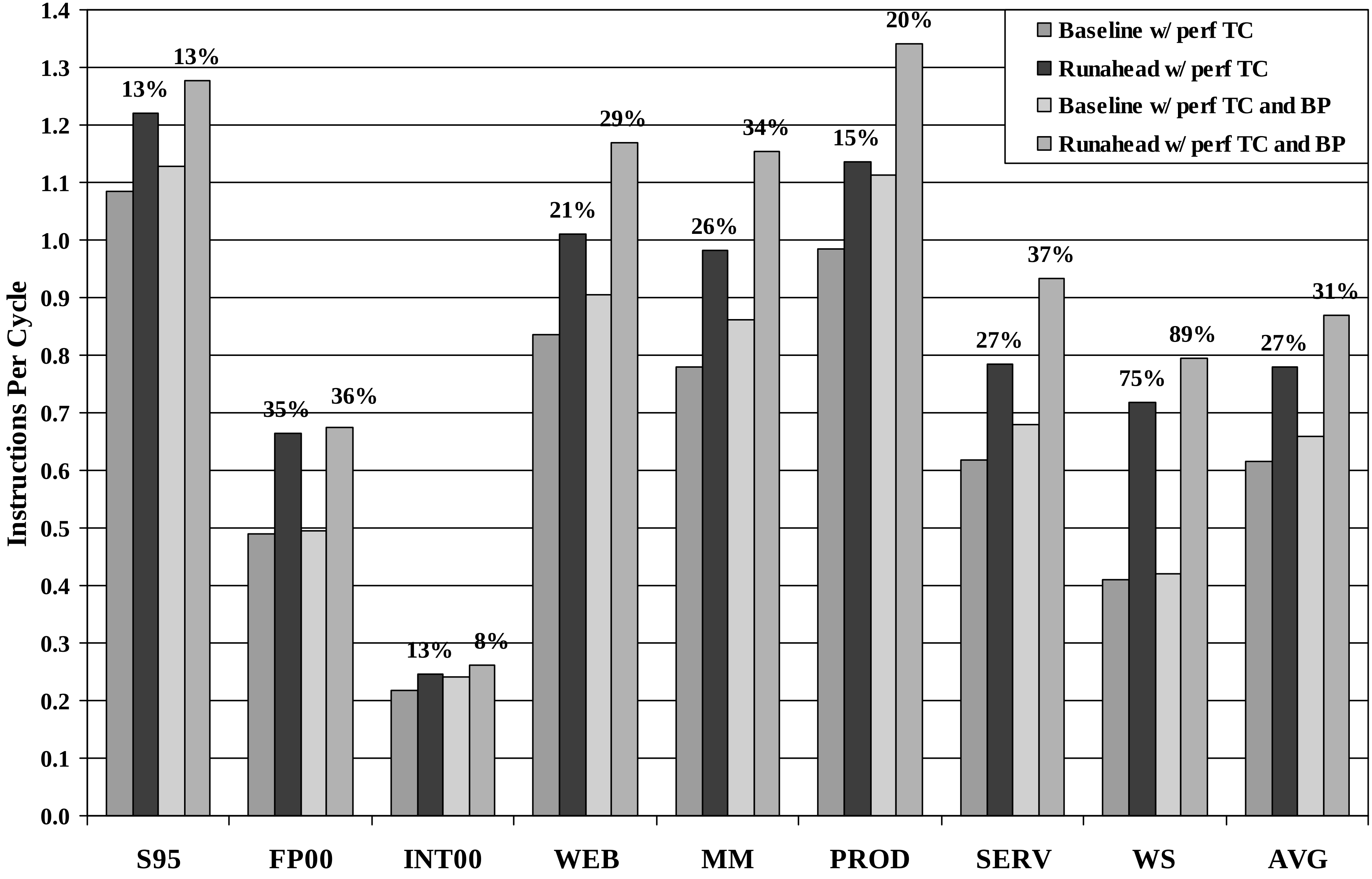
Added Bits & Structures



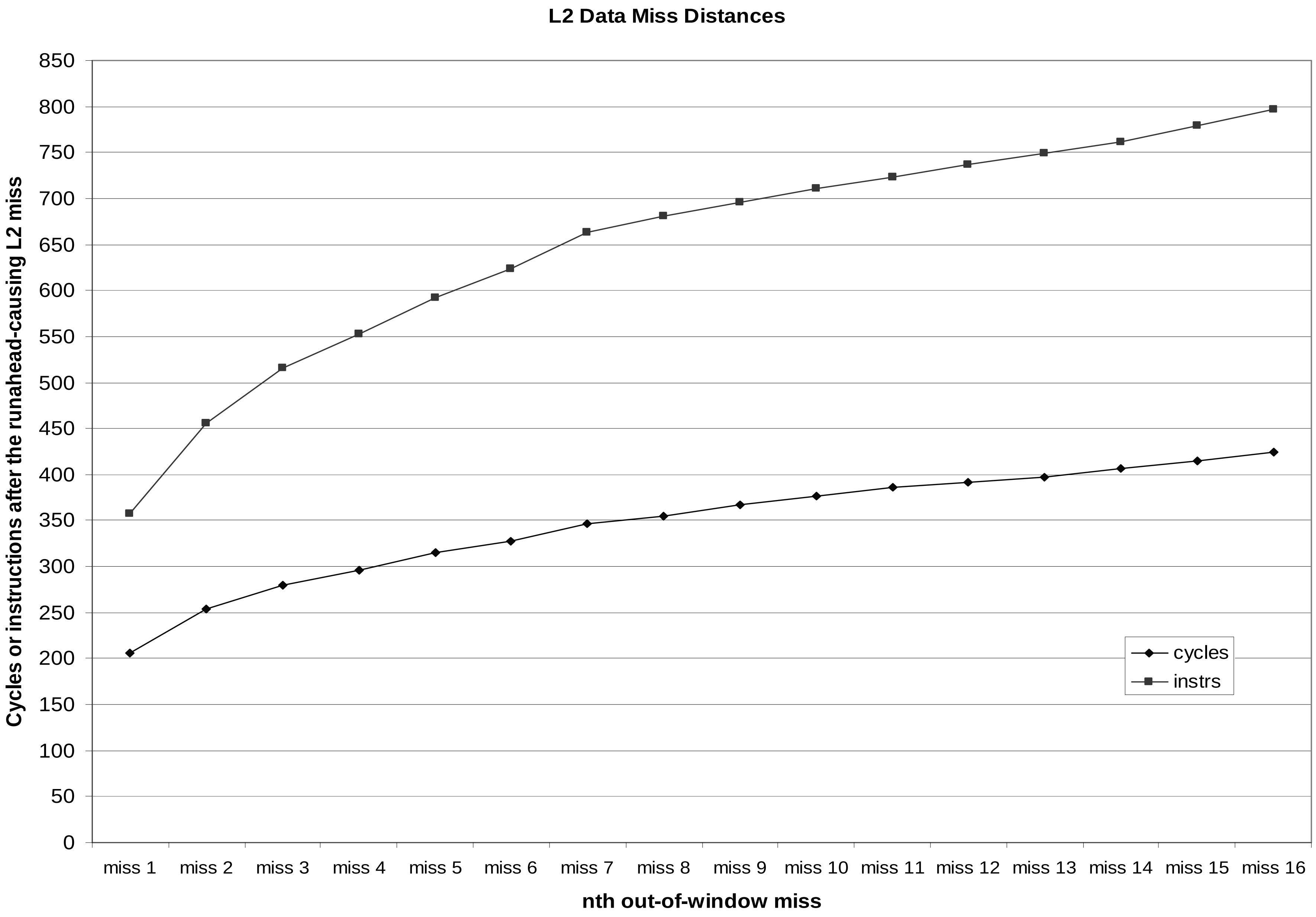
Runahead-Prefetcher Interaction



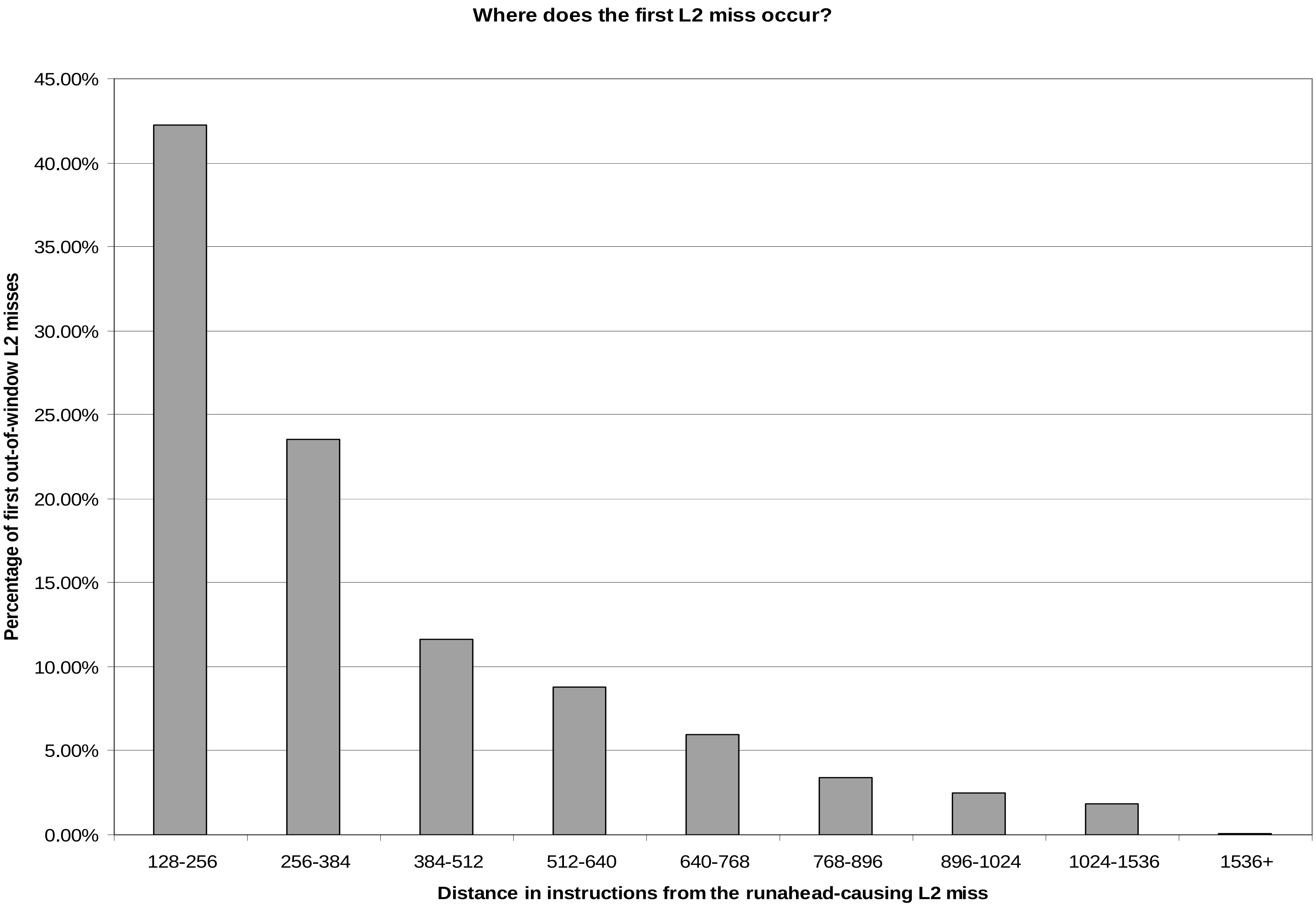
Effect of a Better Frontend



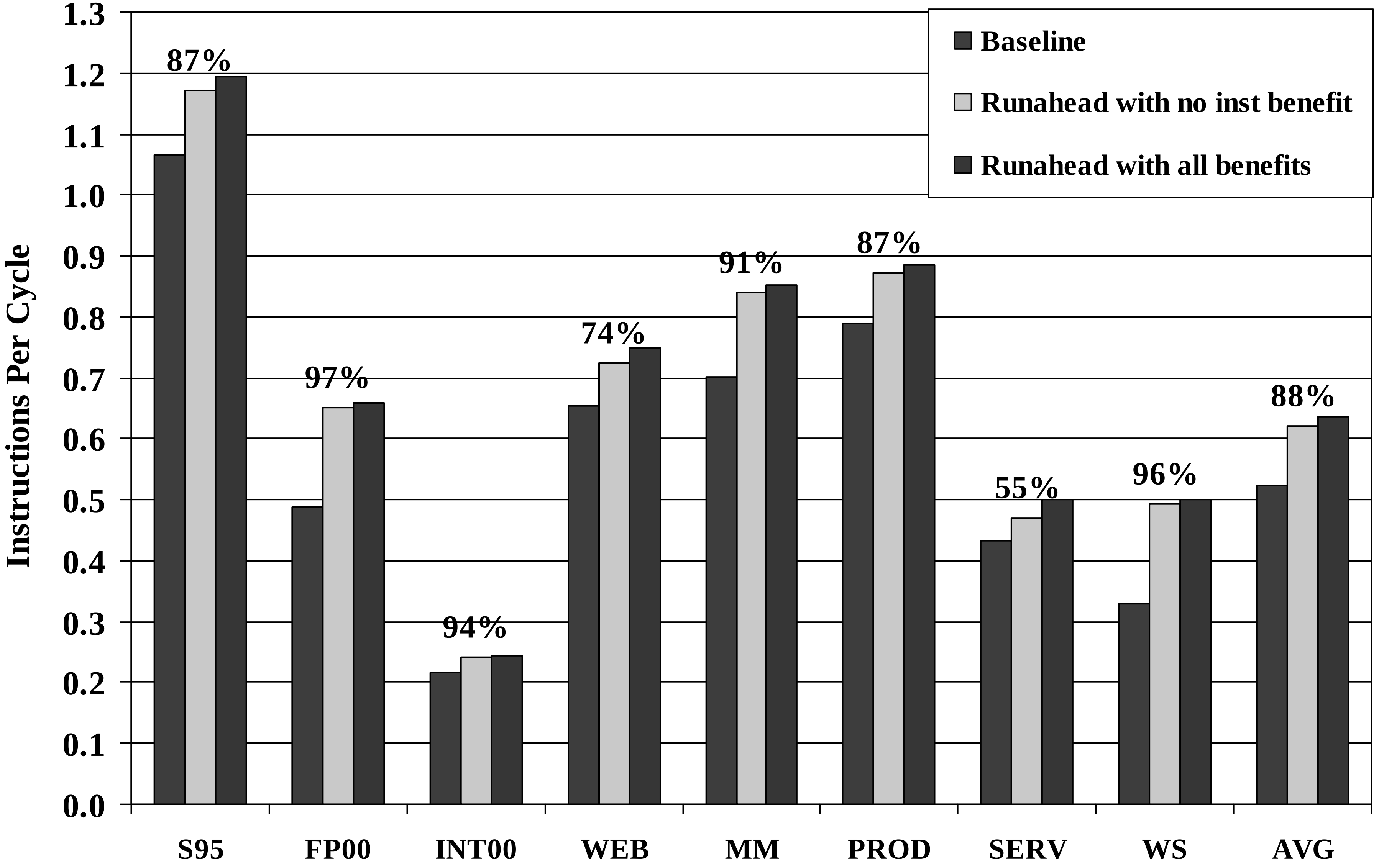
When do we see the L2 misses in Runahead?



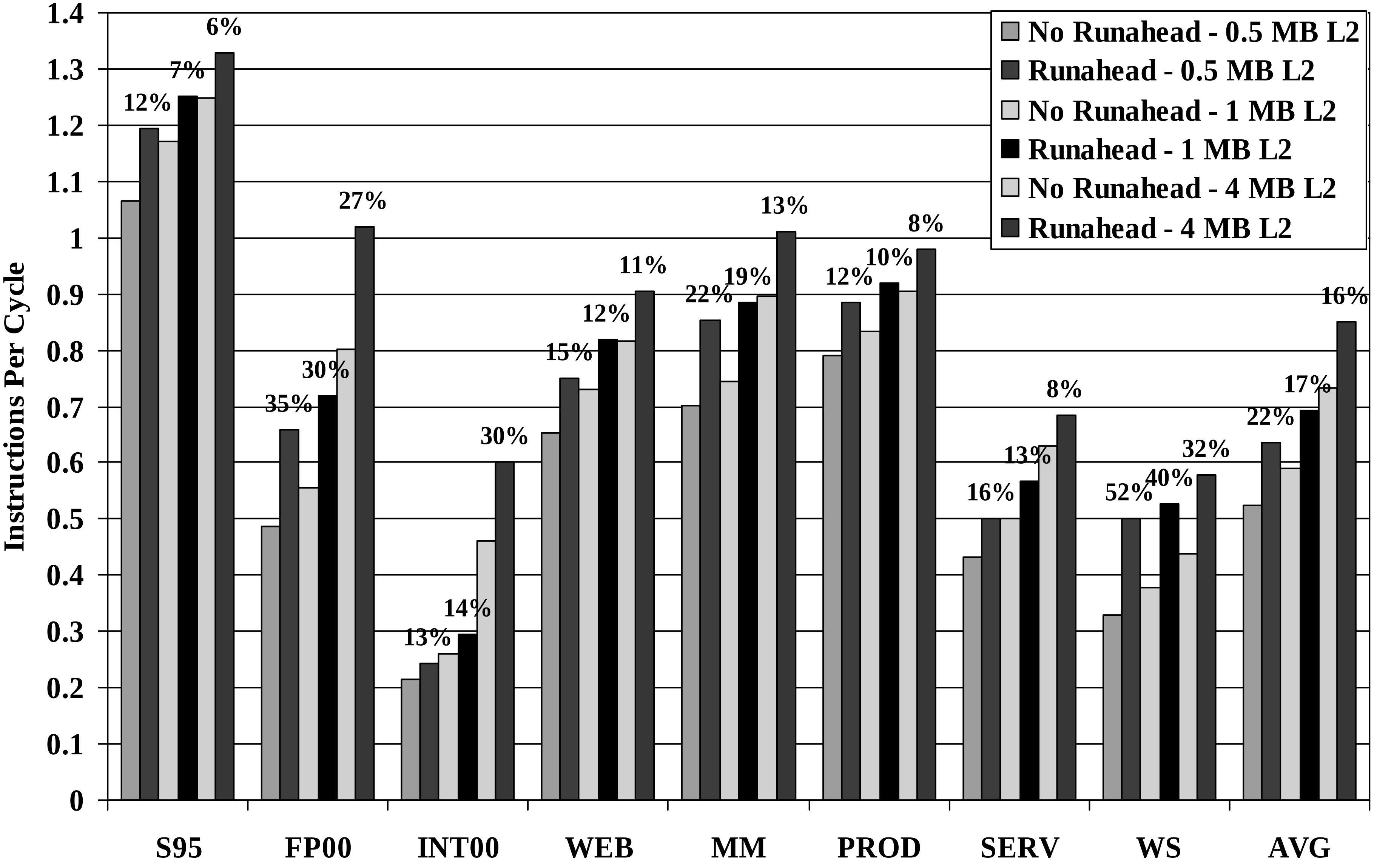
Distance of the first L2 miss in runahead mode



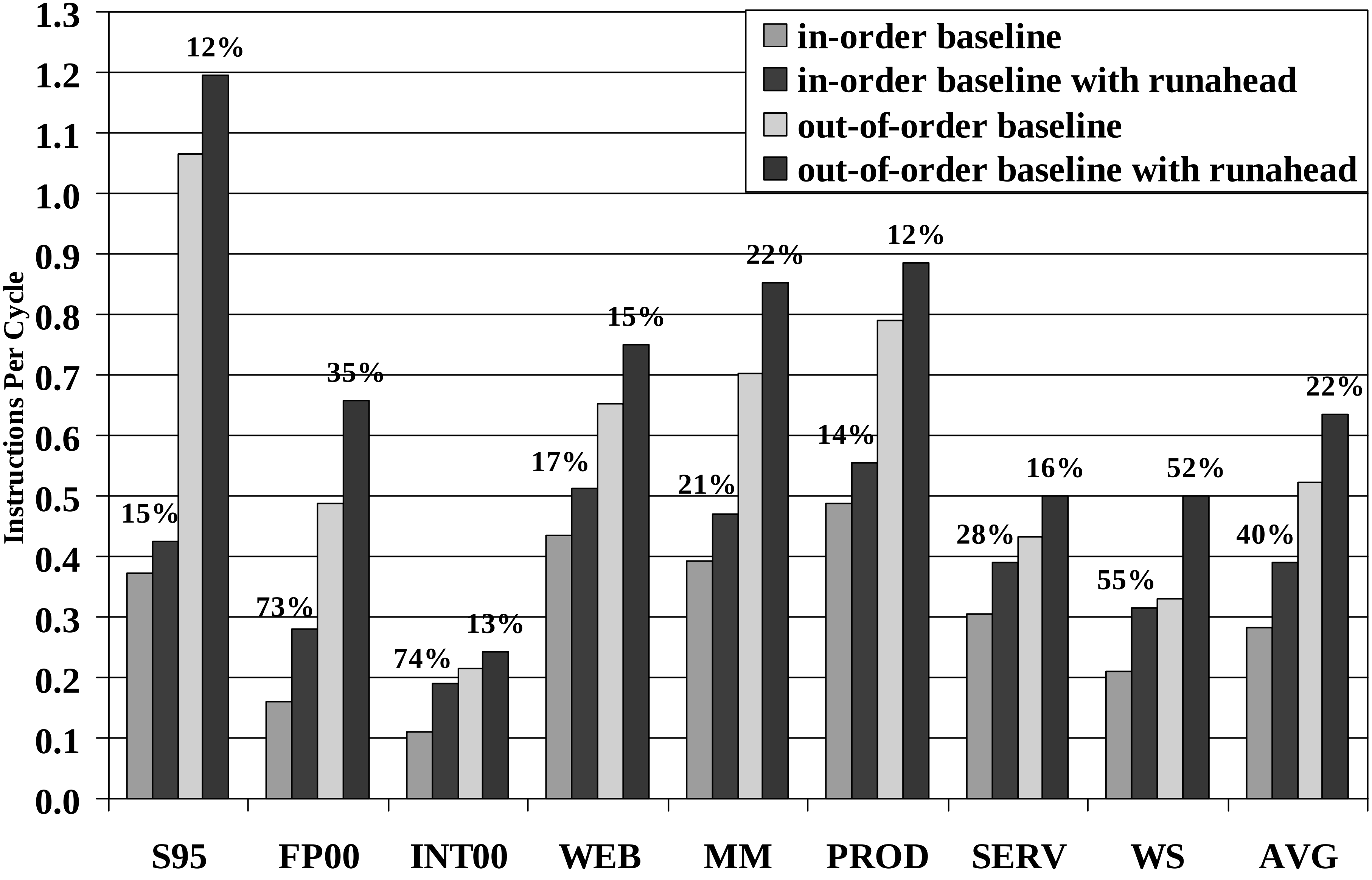
Instruction vs. Data Prefetching Benefit



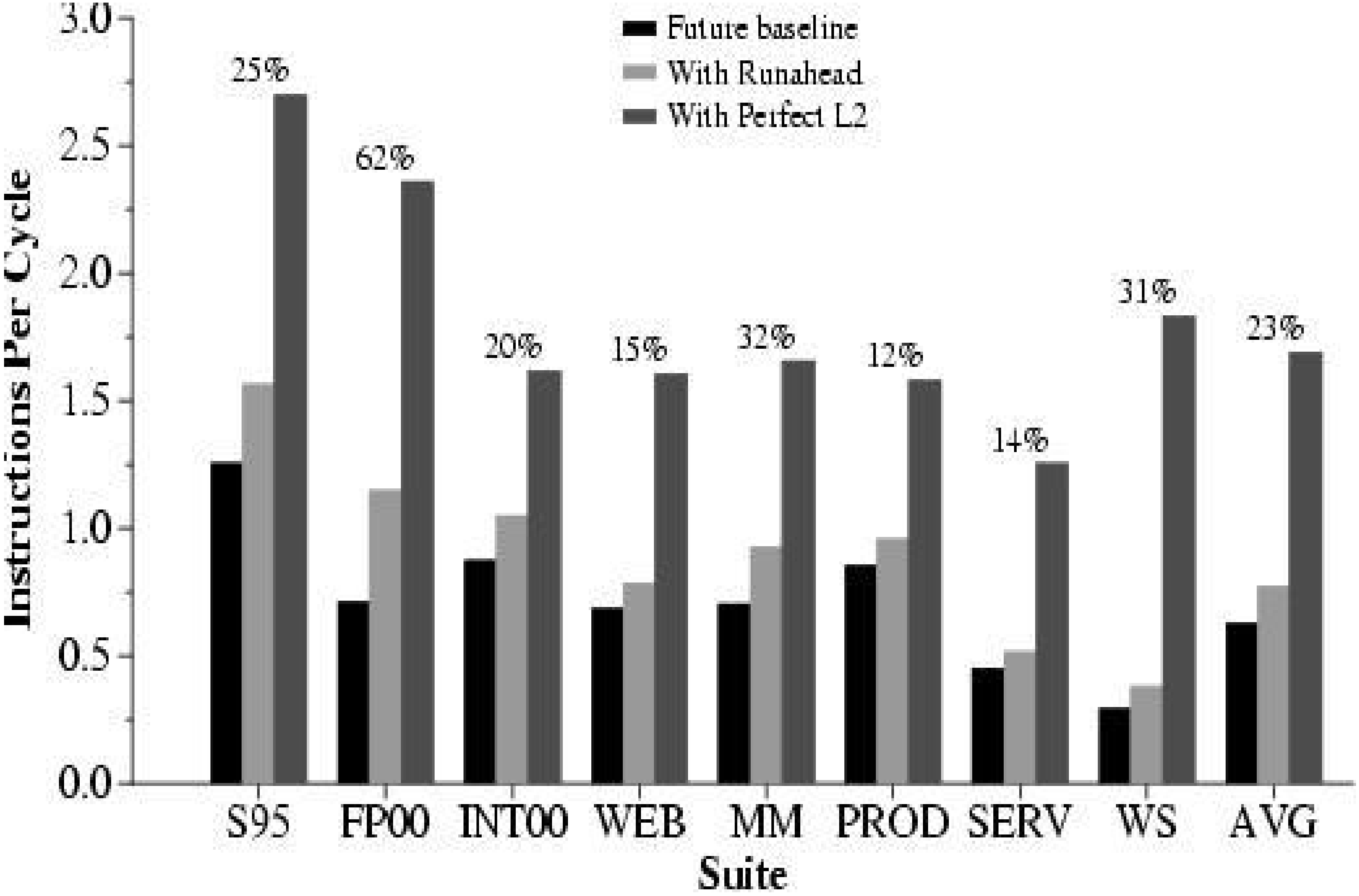
Runahead with a Larger L2 Cache



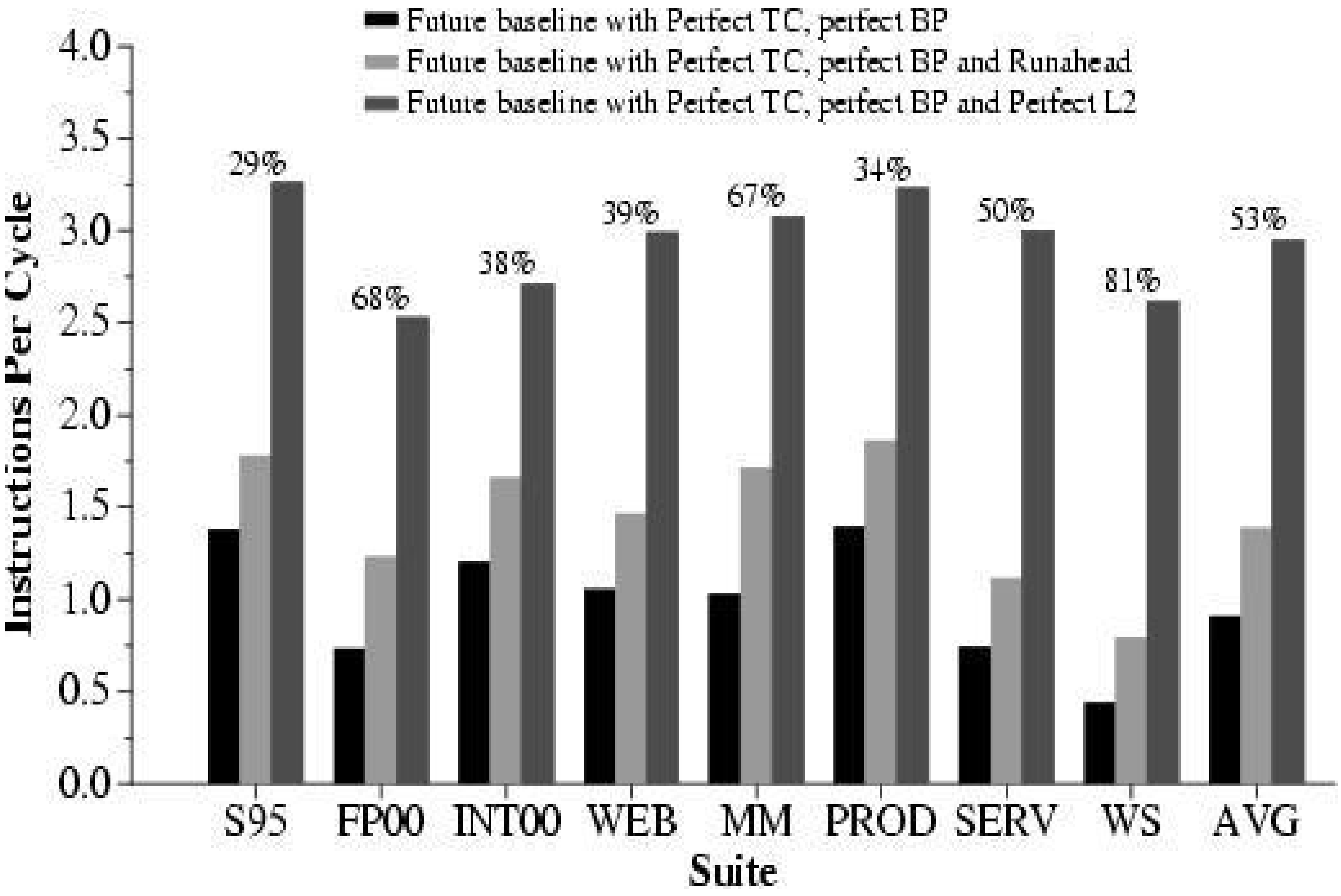
Runahead on in-order?



Future Model Results



Future Model with Better Frontend


Future baseline with Perfect TC, perfect BP
Future baseline with Perfect TC, perfect BP and Runahead
Future baseline with Perfect TC, perfect BP and Perfect L2
Instructions Per Cycle
S95
FP00
INT00
WEB
MM
PROD
SERV
WS
AVG
Suite