

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

Onur Mutlu § Jared Stark †

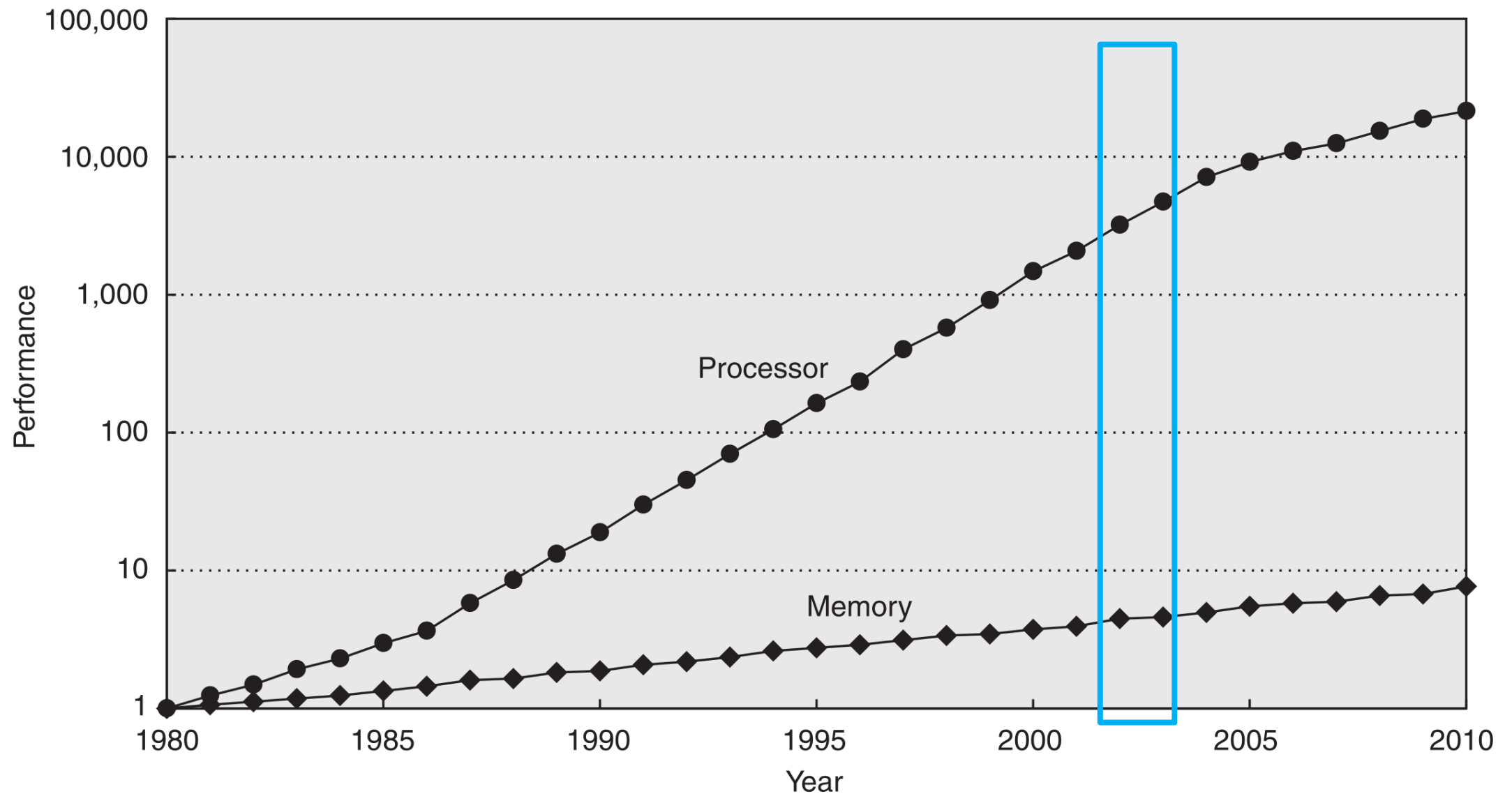
Chris Wilkerson ‡ Yale N. Patt §

§ ECE Department
The University of Texas at Austin
{onur,patt}@ece.utexas.edu

† Microprocessor Research
Intel Labs
jared.w.stark@intel.com

‡ Desktop Platforms Group
Intel Corporation
chris.wilkerson@intel.com

Background, Problem & Goal



The gap in performance between memory and processors is plotted over time
Hennessy, John L., and David A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2007.

Goal: No delays due to cache misses

■ How to achieve?

- ❑ Make the caches bigger?
- ❑ Inform the CPU of future accesses?
- ❑ Let the CPU guess future accesses?
- ❑ Let the memory system guess future accesses?

Expensive

**Consumes
Bandwidth**

Requires Predictor

Pollutes Caches

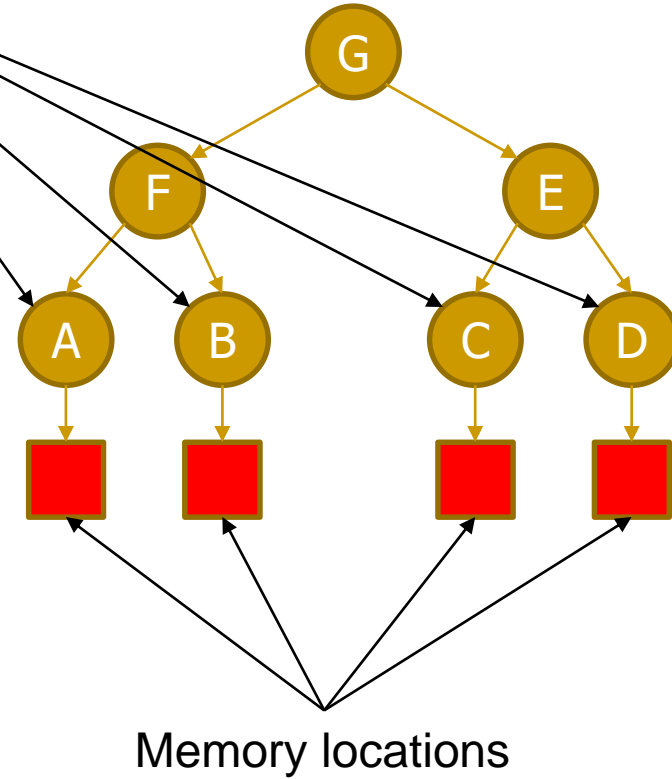
In-order architecture

← Currently executing

PC	Instruction
0	A
1	B
2	C
3	D
4	E
5	F
6	G



Load operations



In-order architecture

■ Advantages

- ❑ Simple to understand, program
- ❑ Cheap to produce
- ❑ Low energy consumption

■ Disadvantages

- ❑ Slow
- ❑ Dependency-unaware
 - Almost no ILP

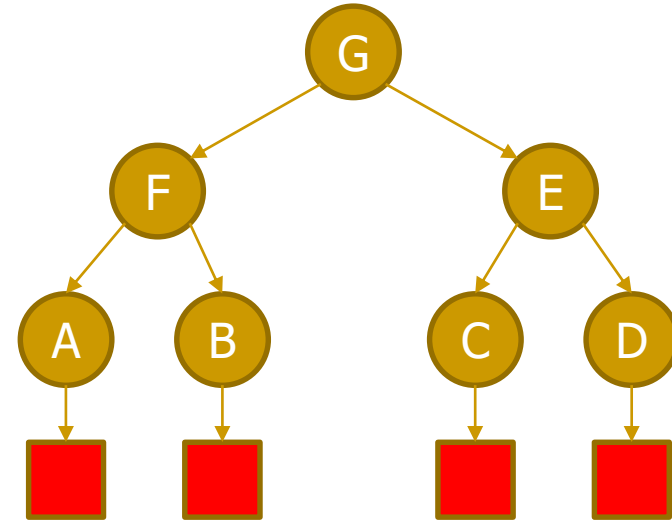
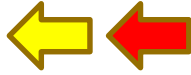
In-order with runahead execution

- Dundas, James, and Trevor Mudge. "Improving data cache performance by pre-executing instructions under a cache miss." *Proceedings of the 11th international conference on Supercomputing*. ACM, 1997.
- Idea: Instead of blocking on memory operations, run ahead and touch everything
 - But do not change the architectural state

In-order architecture with runahead

← Currently executing ← Runahead execution

PC	Instruction
0	A
1	B
2	C
3	D
4	E
5	F
6	G



In-order architecture with runahead

- Advantages

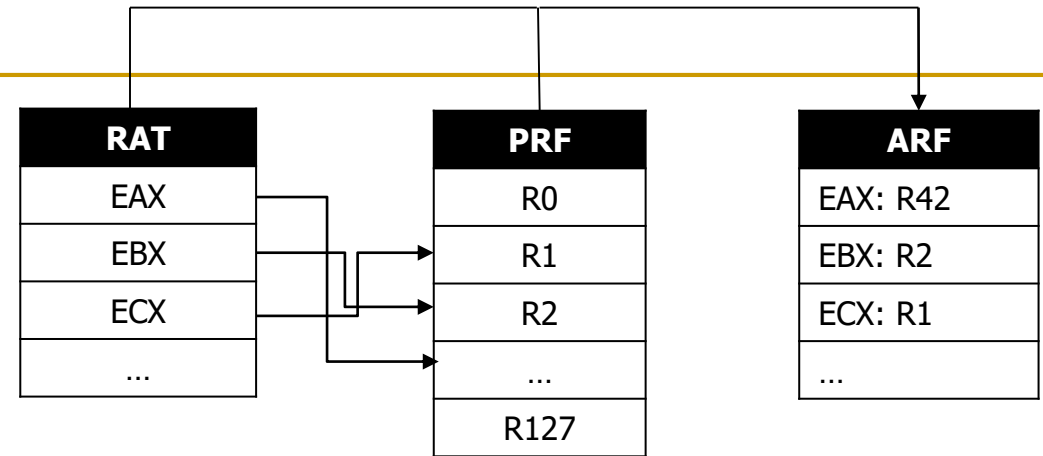
- Simple
- MLP

- Disadvantages

- Small additional cost
- Some executed instructions are repeated
 - Results of runahead execution are not reused

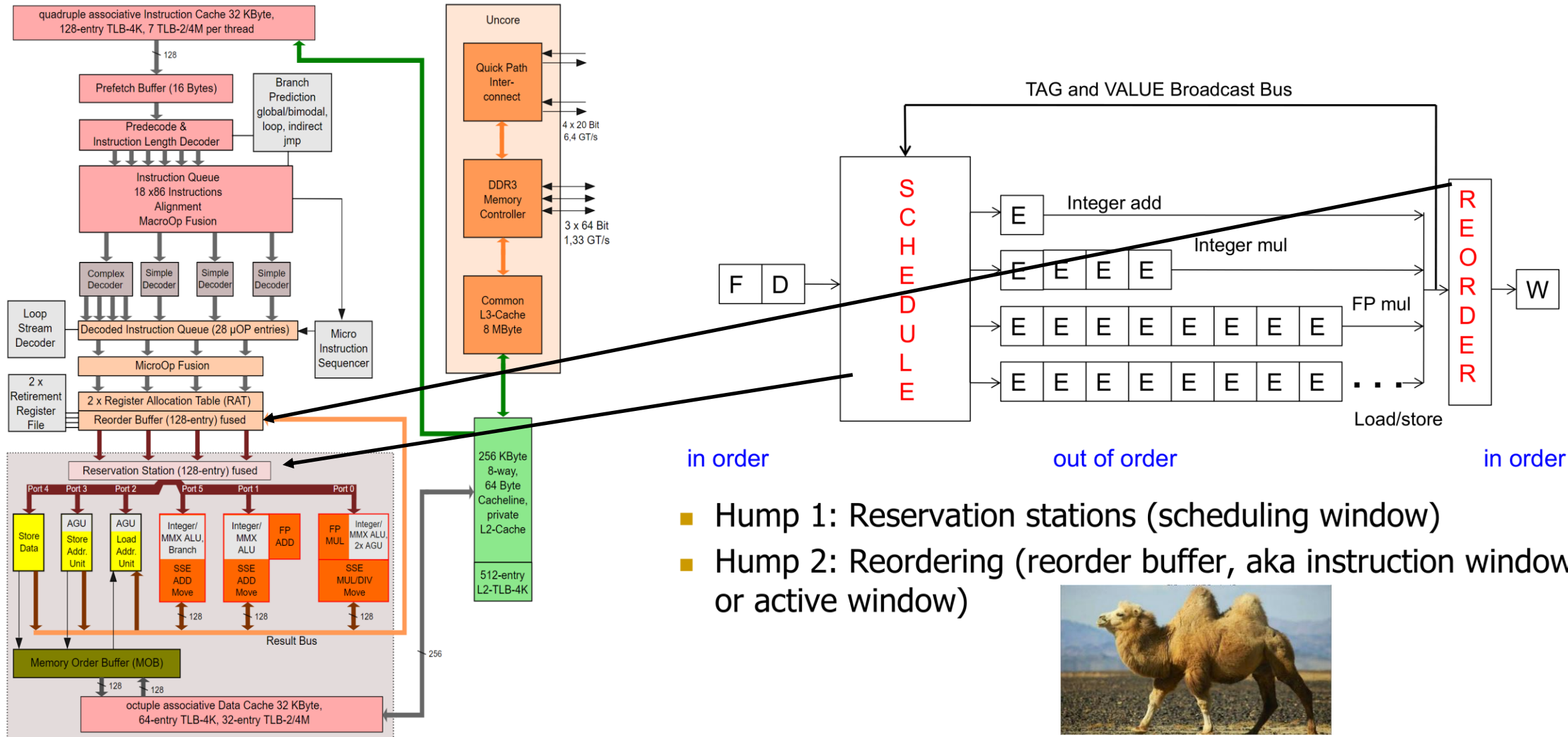
Out-of-Order architecture

- **Physical Register File (PRF)**
 - ❑ Physical Memory for Registers
- **Architectural Register File (ARF)**
 - ❑ “Programmer model”
- **Register Alias Table (RAT)**
 - ❑ Mapping architectural (virtual) registers to physical registers
- **PRF much larger than ARF**
- **Register Renaming**
 - ❑ Rename the Architectural Register of an instruction to a Physical Register (and back)
- **Retirement**
 - ❑ Effects of Instruction become observable
 - ❑ In-order (only head of instruction window can retire)



Out-of-order architecture

Intel Nehalem microarchitecture



Out-of-order architecture

■ Scheduling Window

- How many instructions are waiting for execution
- Element on chip: Reservation Station

■ Instruction Window

- How many instructions are waiting to be retired
- Element on chip: Reorder Buffer (ROB)

■ In reality: Instruction Window larger than Scheduling Window

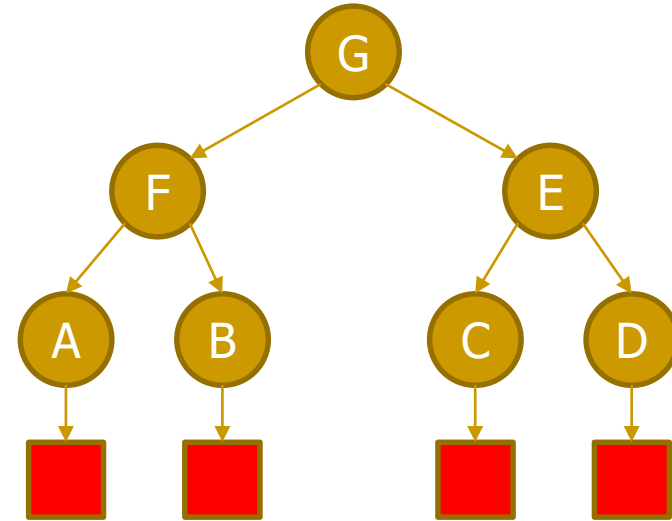
- Sched. W. subset of Inst. W.

■ For this presentation: Instruction Window = Scheduling Window

Out-of-order architecture

← Currently executing ← Out-of-Order execution

PC	Instruction
0	A
1	B
2	C
3	D
4	E
5	F
6	G



 Instruction Window

Out-of-order architecture

■ Advantages

- ❑ Dependency-Aware
- ❑ Fast (ILP, MLP)
- ❑ Instructions executed once

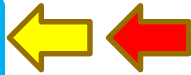
■ Disadvantages

- ❑ Expensive
- ❑ Performance largely dependent on window size
- ❑ Blocking

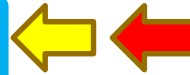
Instruction Window Size

← Currently executing ← Out-of-Order execution

0	A
1	B
2	C
3	D
4	E
5	F
6	G



0	A
1	B
2	C
3	D
4	E
5	F
6	G



More transistors
More addressing bits
More comparators
Higher **Memory Contention**
Higher **Power Consumption**
“Dark Silicon” with cache-local code
More **cache pollution** on mispredictions

Key Approach and Ideas

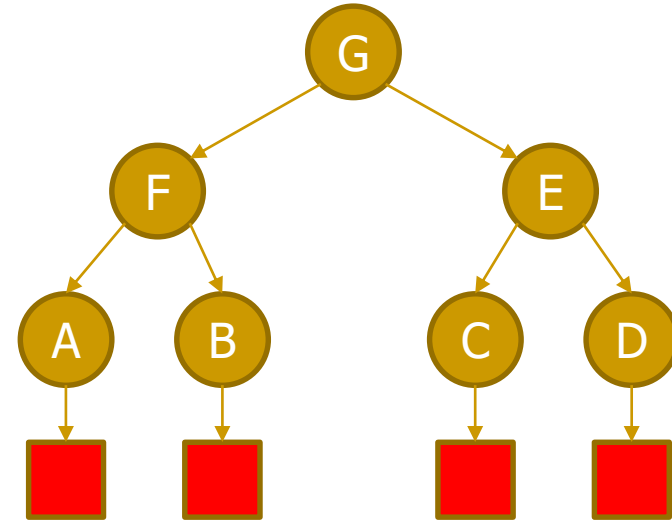
Make the window non-blocking



- A non-blocking window behaves like a bigger blocking window
 - But costs less
- Existing hardware can be used while otherwise idle

Out-of-order architecture with Runahead

← Currently executing ← Runahead/Out-of-order execution

PC	Instruction
0	A
1	B
2	C
3	D
4	E
5	F
6	G



 Instruction Window
 Runahead Instruction Window

Out-of-order architecture with Runahead

■ Advantages

- ❑ Dependency-Aware
- ❑ Fast (ILP, **increased** MLP)
- ❑ Less hardware cost than bigger instruction windows and OoO-only
- ❑ **Increases usage**, less misses

■ Disadvantages

- ❑ Expensive
- ❑ Slight additional hardware cost
- ❑ **Instructions are repeated**

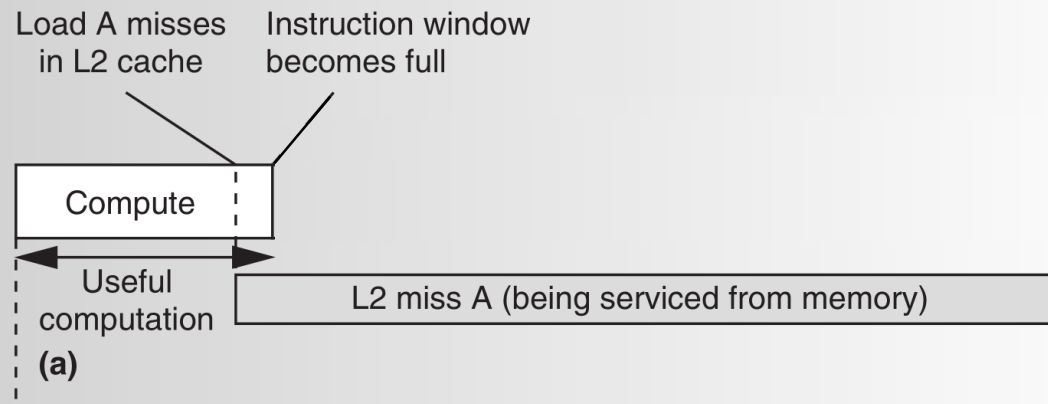
Conventional OoO

Load A misses
in L2 cache

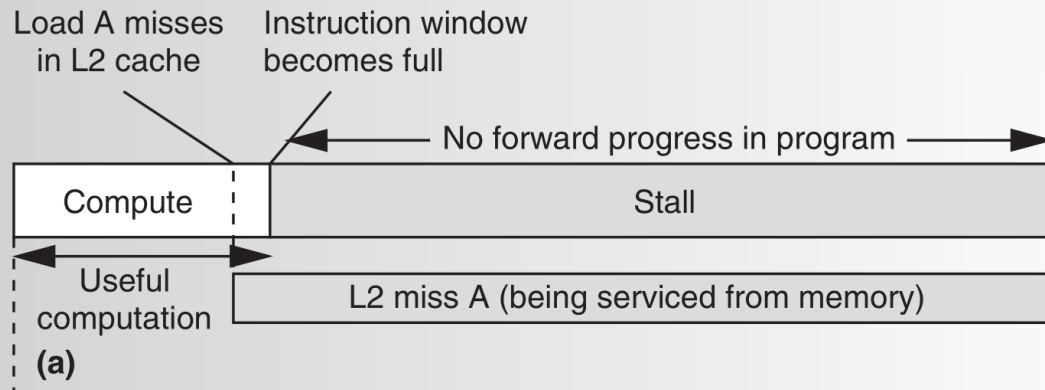
Compute

Useful
computation
(a)

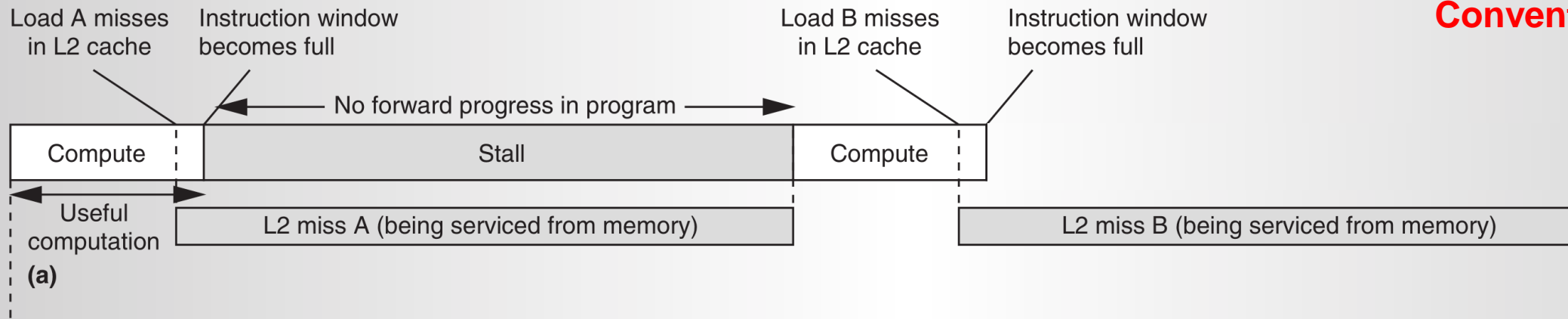
Conventional OoO



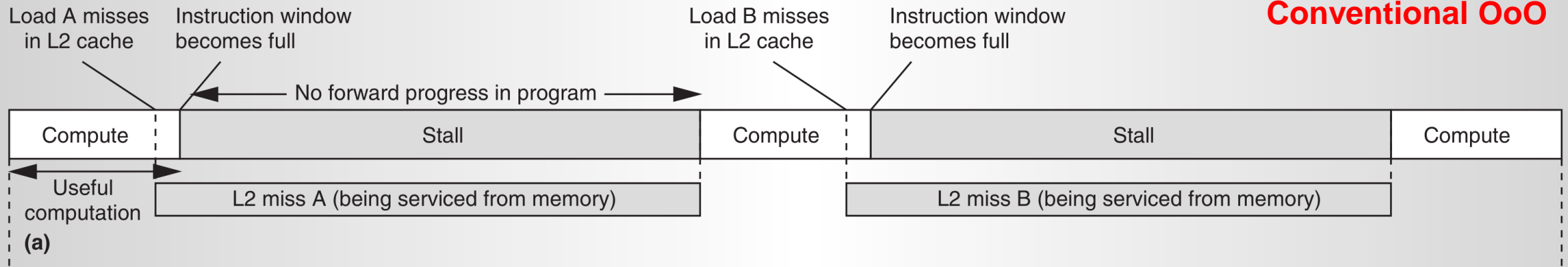
Conventional OoO

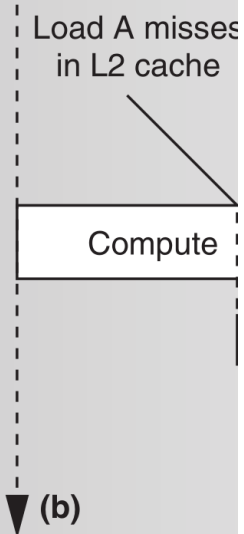
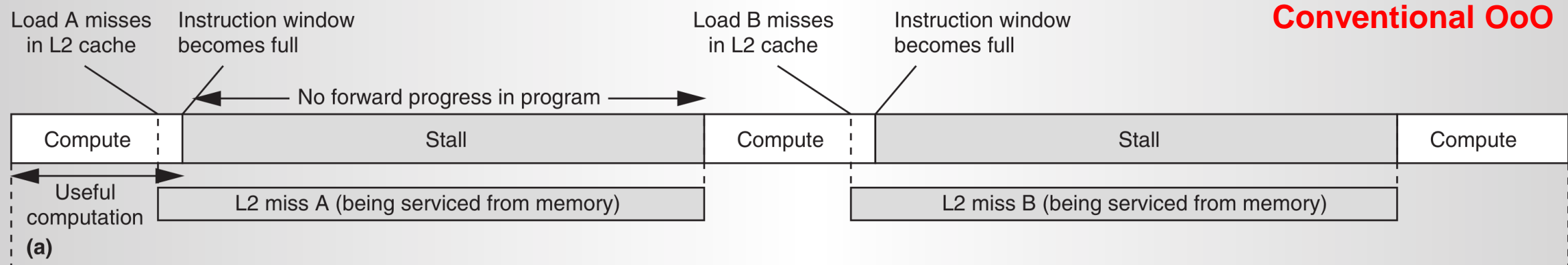


Conventional OoO

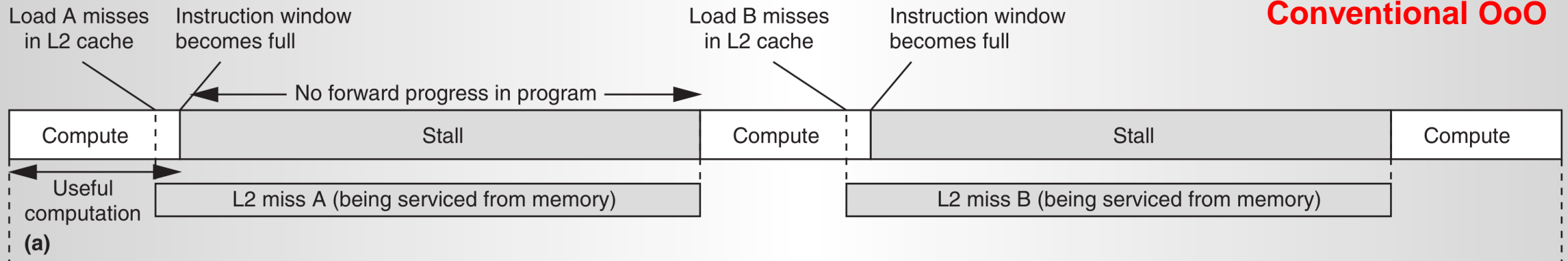


Conventional OoO

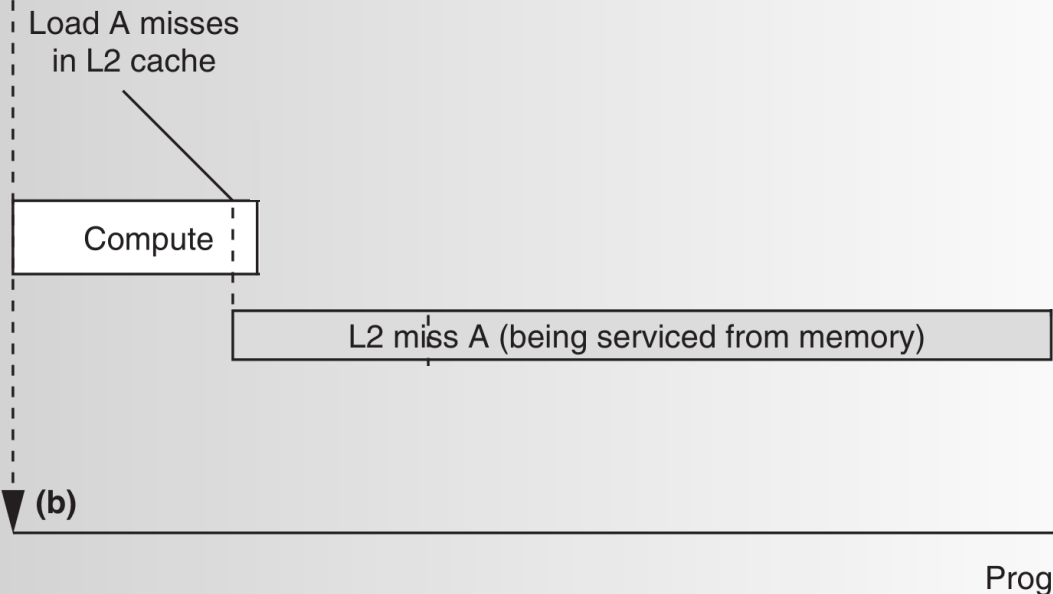


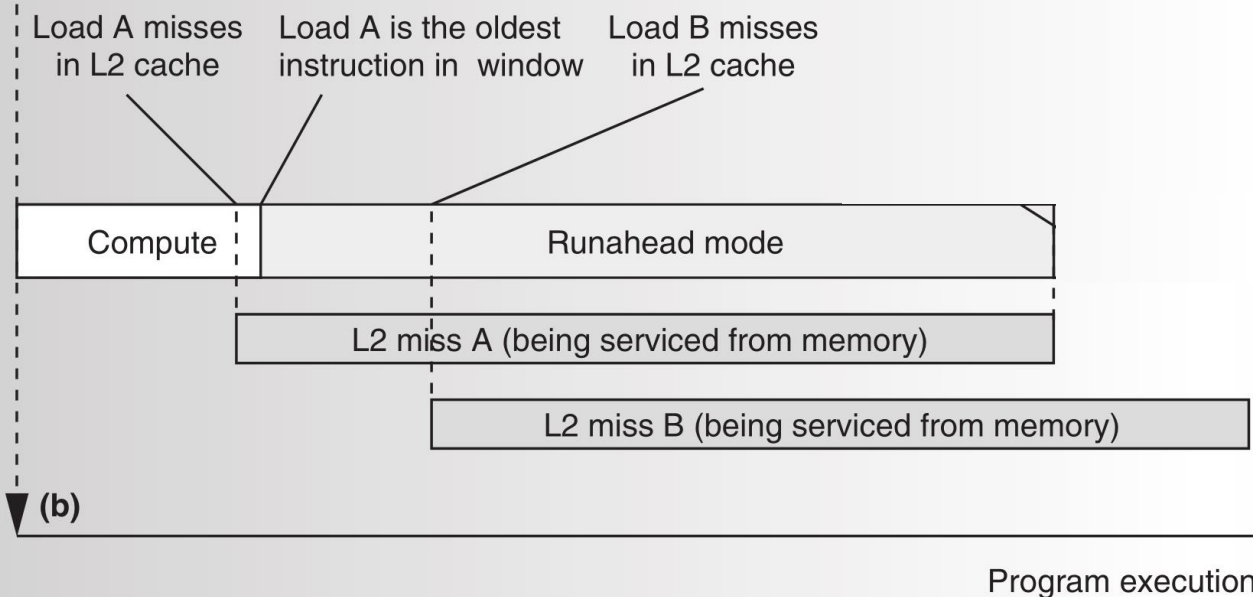
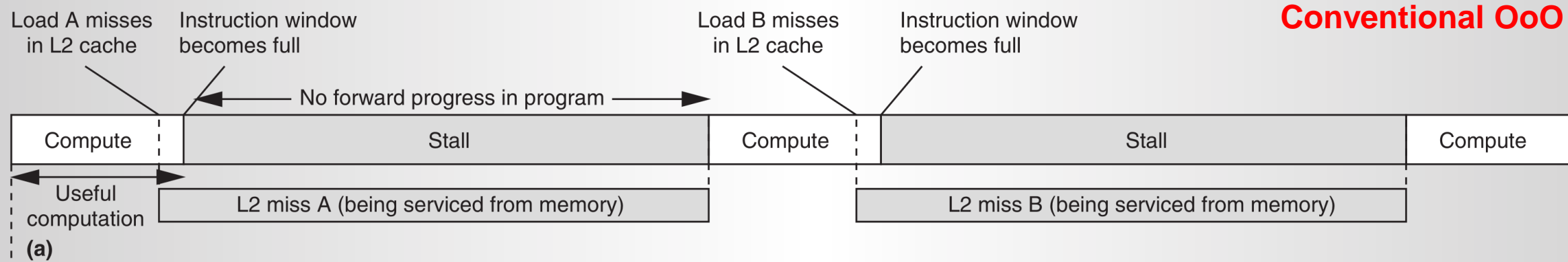


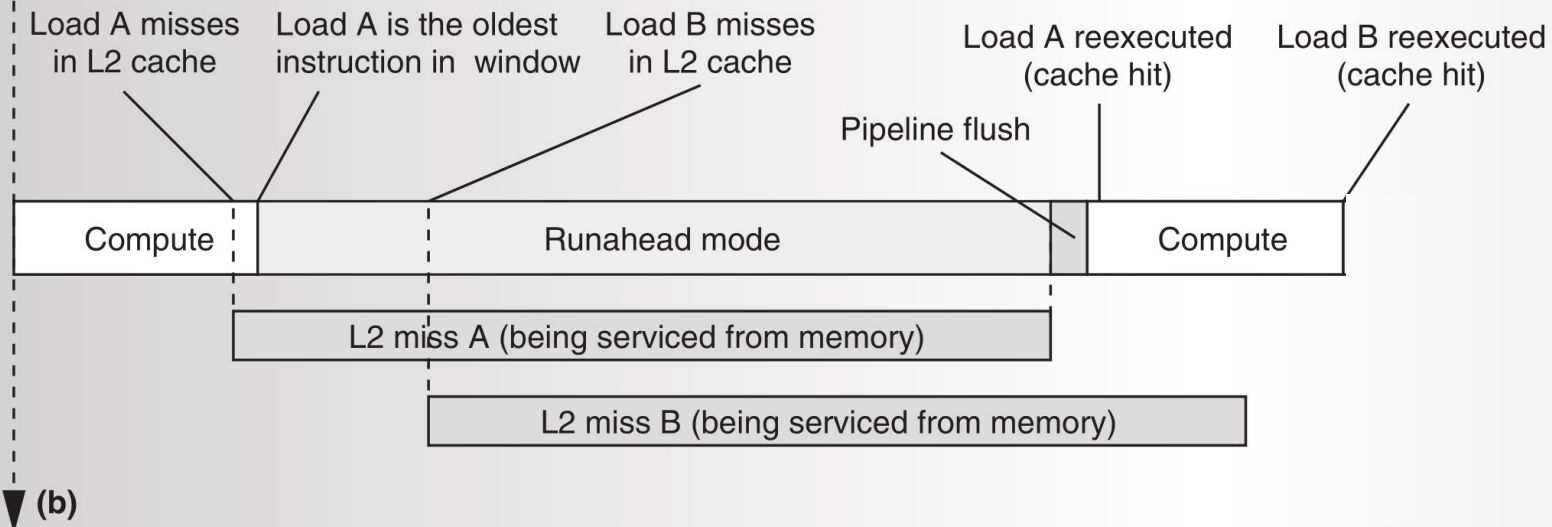
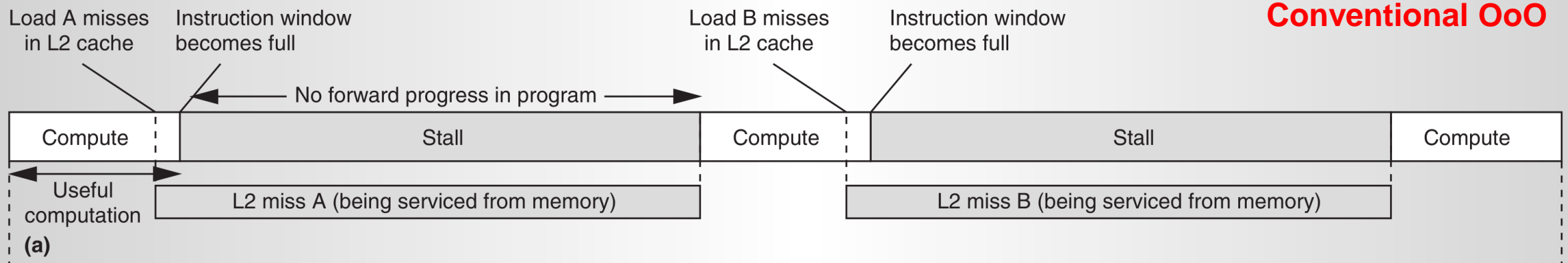
Conventional OoO



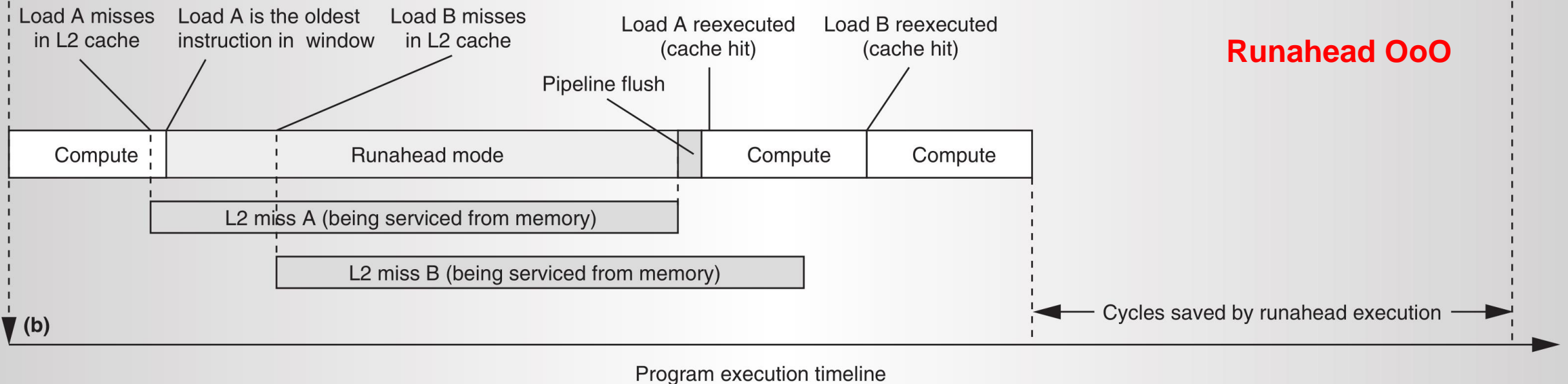
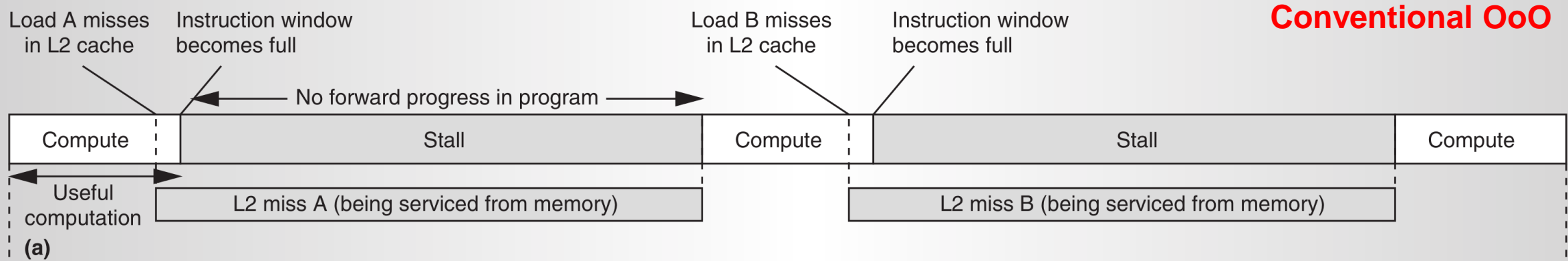
Runahead OoO







Program execution timeline



Mechanisms (in some detail)

Runahead Mode

- Turning the CPU into an expensive (and smart) prefetcher
- Everything runs the same as in “Normal Mode”
- Exceptions:
 - Interrupts
 - I/O Accesses
 - Stores
- Has no effect on the architectural state
 - “Hidden from the programmer”

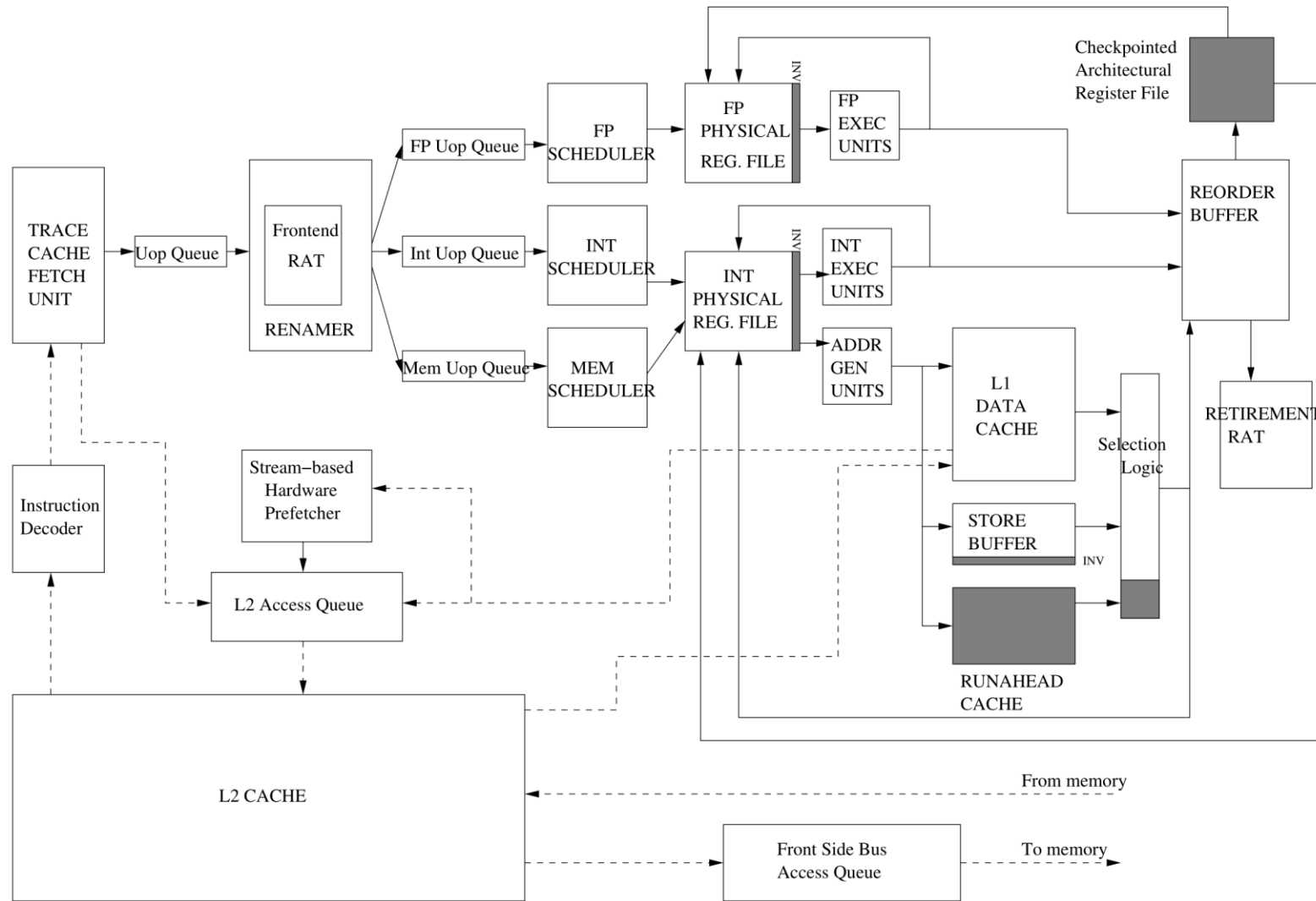
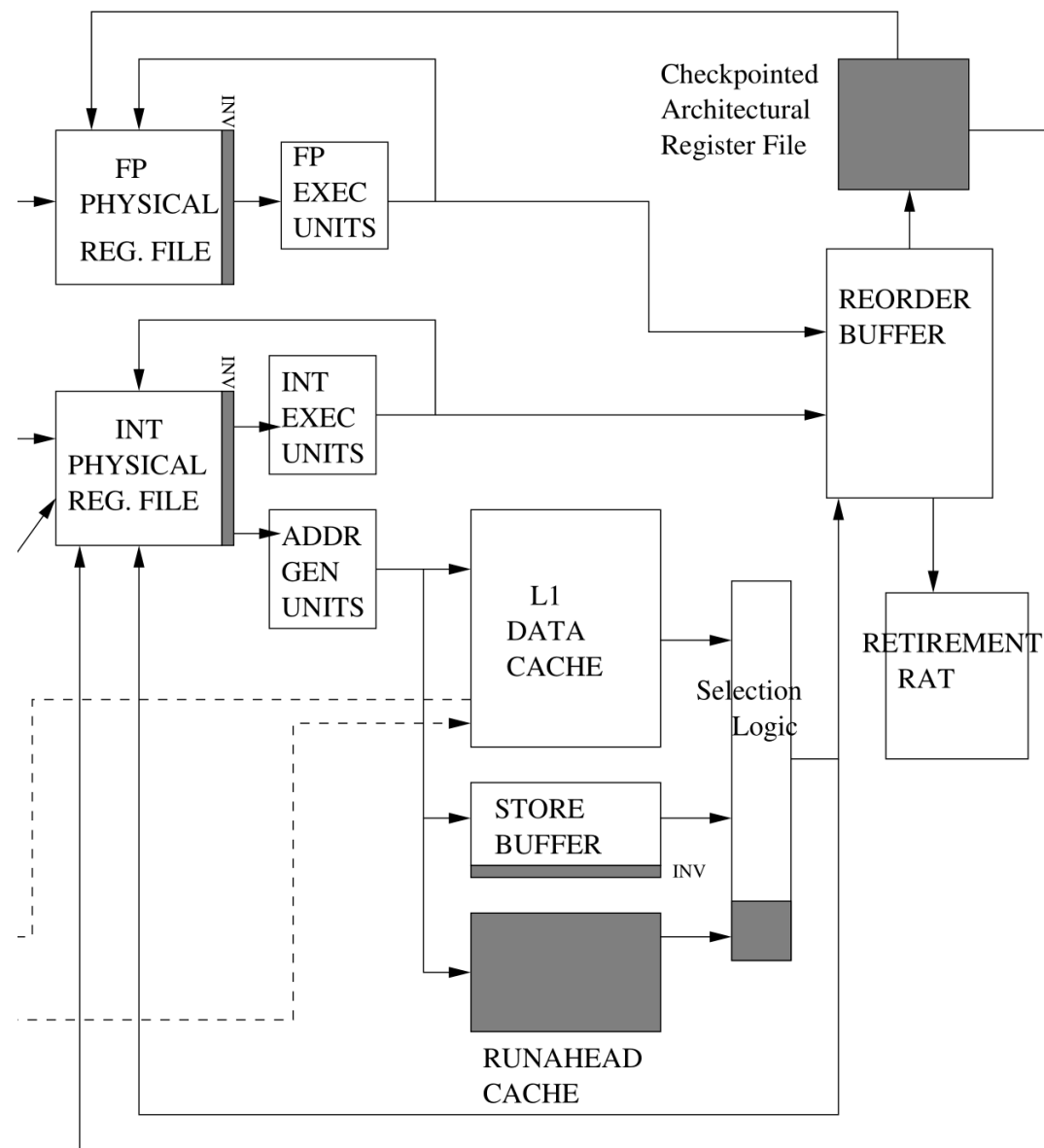


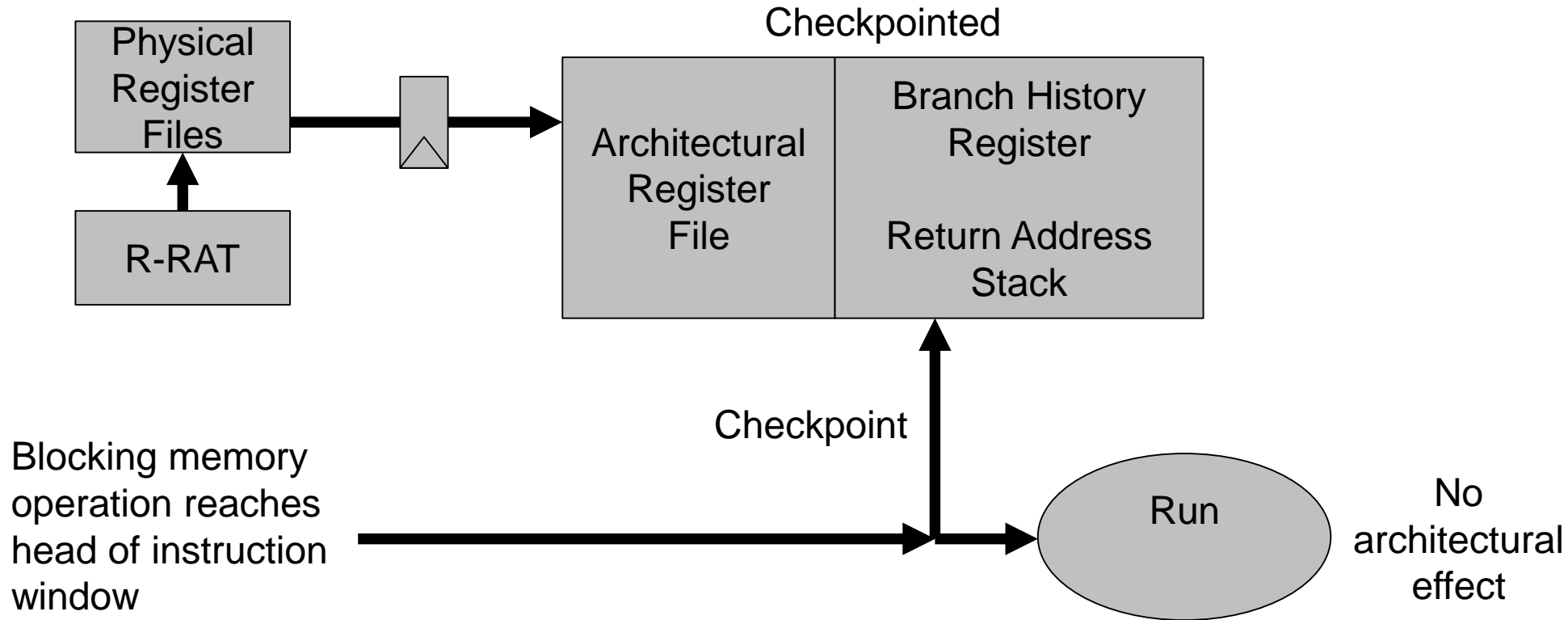
Figure 2. Processor model used for description and evaluation of runahead. Figure is not to scale.

Mutlu, Onur, et al. "Runahead execution: An alternative to very large instruction windows for out-of-order processors." *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*. IEEE, 2003.

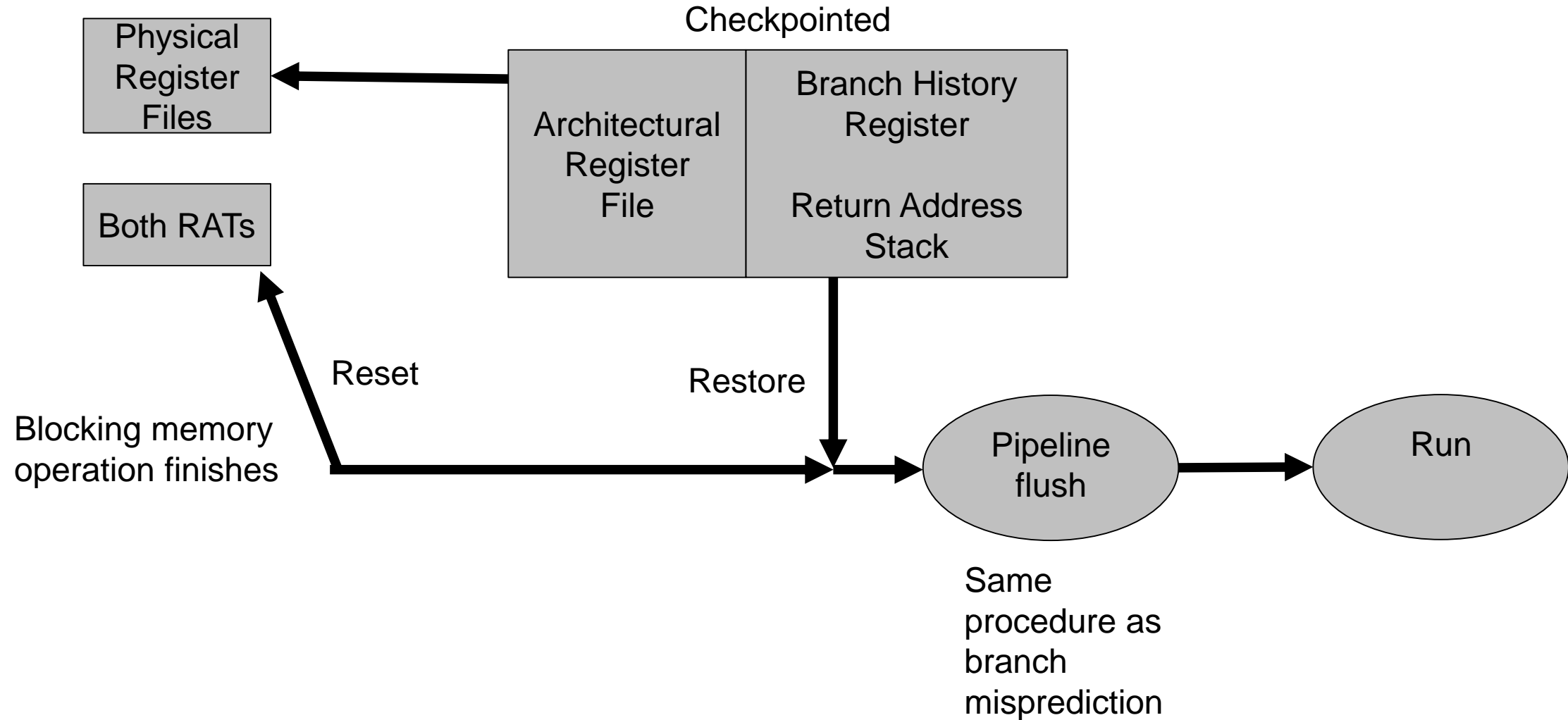


Mutlu, Onur, et al. "Runahead execution: An alternative to very large instruction windows for out-of-order processors." *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on.* IEEE, 2003.

Entering Runahead Mode



Leaving Runahead Mode



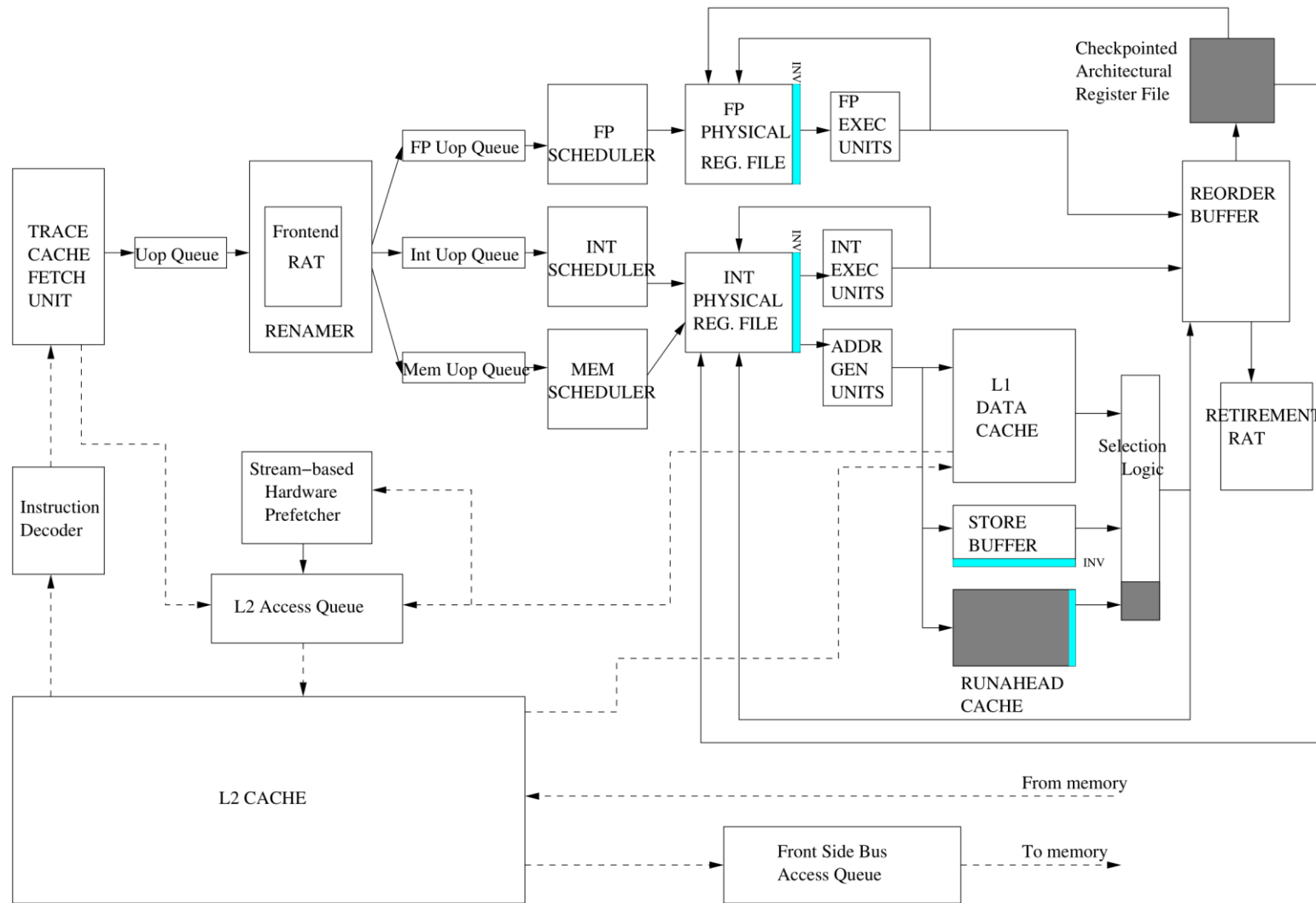
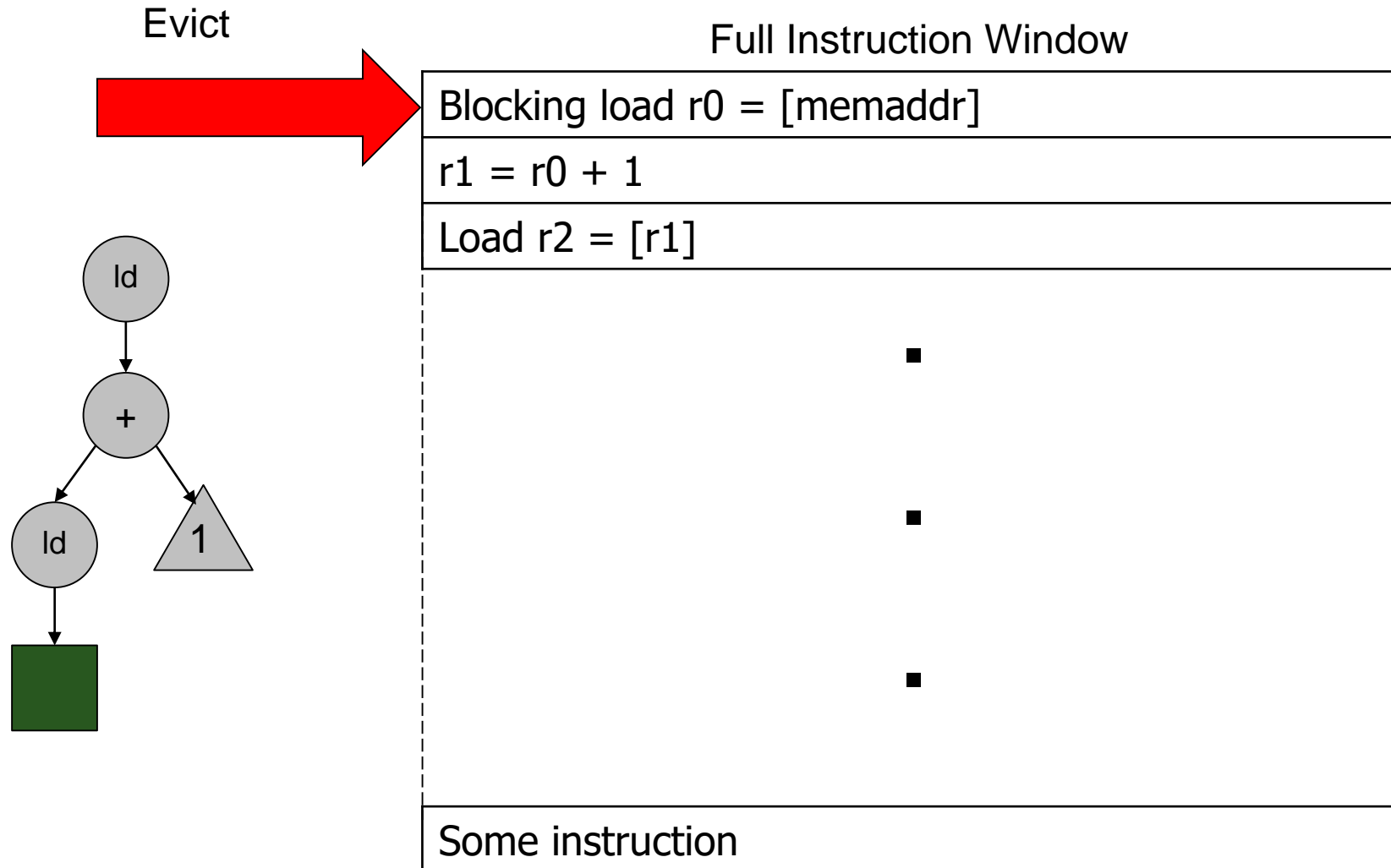


Figure 2. Processor model used for description and evaluation of runahead. Figure is not to scale.

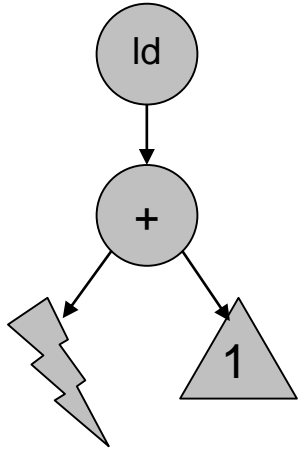
Mutlu, Onur, et al. "Runahead execution: An alternative to very large instruction windows for out-of-order processors." *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on. IEEE, 2003.*

The root of all evil



Dependency

What is r0 now?



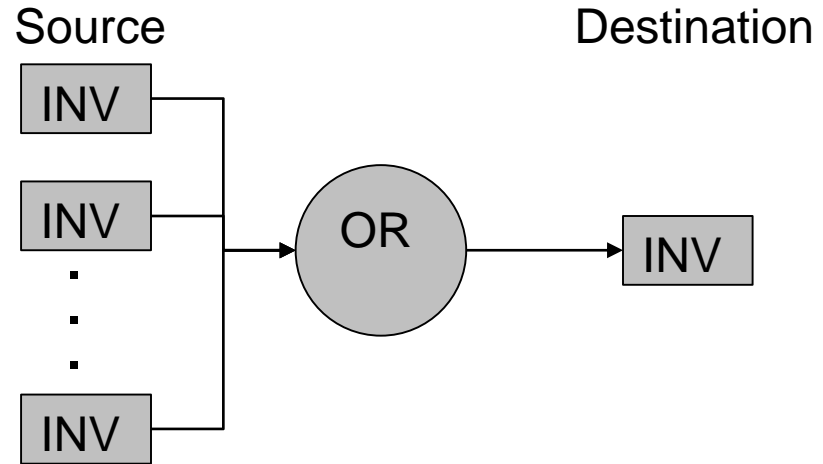
Instruction Window

$$r1 = r0 + 1$$

Load r2 = [r1]

This load is imprecise
Pollutes cache

Instruction and Data Validity



Any source invalid implies destination invalid,
makes instruction “invalid”

The instruction causing the runahead mode is
invalid by definition

If an instruction reaches the head of the
instruction window:

- if invalid: pseudo-retire immediately
- else: wait for execution

Instruction and Data Validity

; eax is 42

mov ebx, eax

; eax is invalid

; eax is valid

; ebx is invalid

; ebx is 42, valid

What about store operations?

- Instructions in Runahead mode must not change the architectural state
- In previous work (ACM 1997), store operations were ignored
 - But they are actually essential to performance

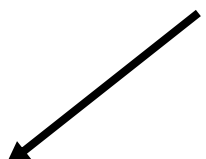
```
mov dword ptr[edx+8], eax
```

```
//...
```

```
mov ebx, dword ptr[edx+8]
```

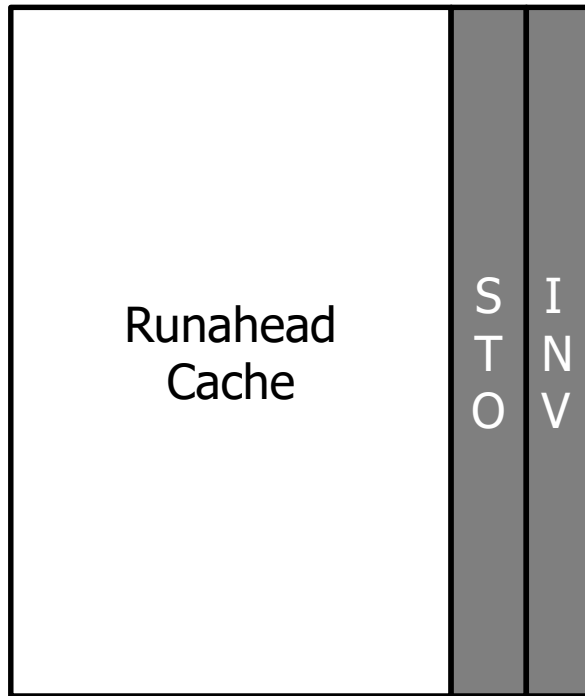
```
mov ecx, dword ptr[ebx]
```

```
ptr[2] = in;  
//...  
tmp = *(ptr[2]);
```



ecx depends on ebx
and memory state,
ebx depends on eax

New “cache”



- Never writes back
- 512B
- STO-bit
 - Inverse cache-cold-bit
- INV-bit

Store operations

Invalid store instruction scheduled

Sets the invalid bit of the store buffer

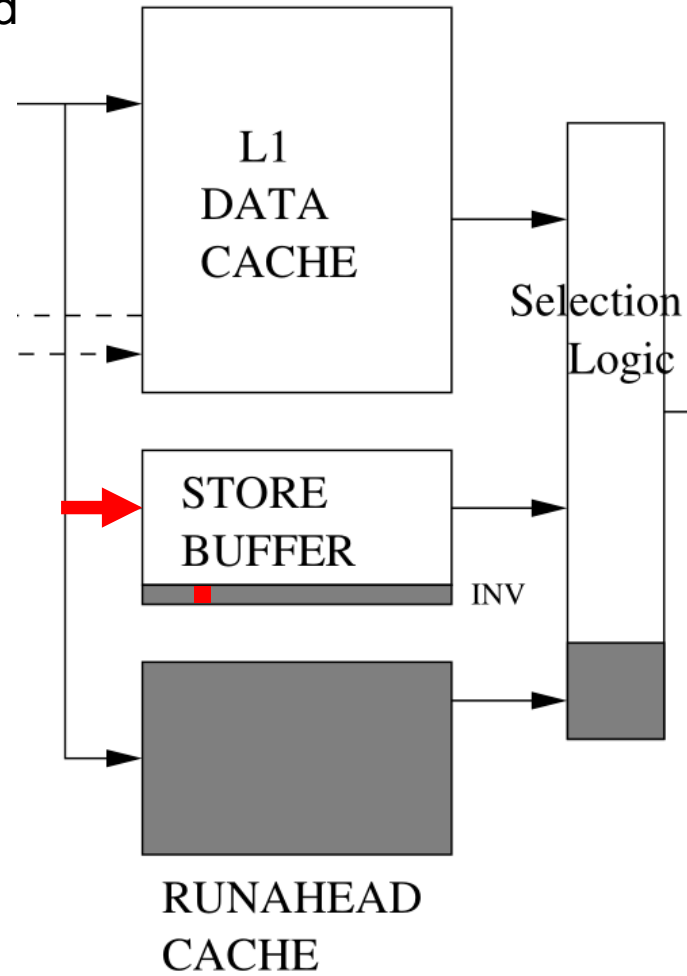
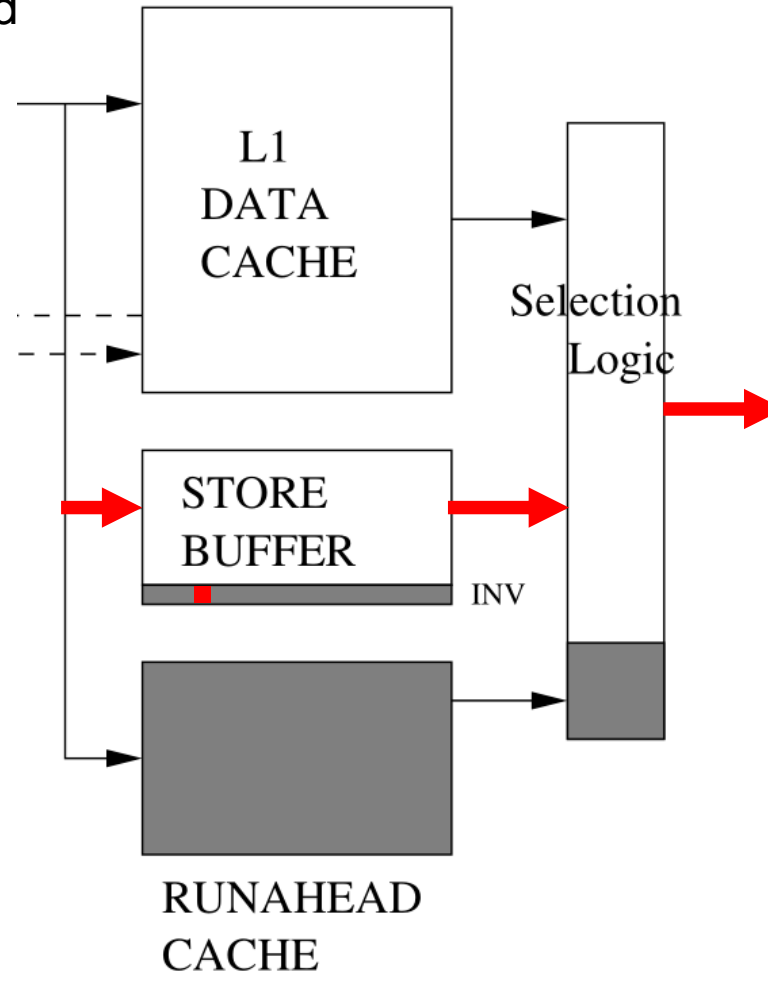


Image:
Mutlu, O.
HPCA03

Load operations

Invalid store instruction scheduled



Sets the invalid bit of the store buffer

```
mov dword ptr[esp+8], eax
// few instructions
mov ebx, dword ptr[esp+8]
mov ecx, dword ptr[ebx]
```

ebx is now INV

Image:
Mutlu, O.
HPCA03

Store operations

Invalid store instruction retired

sets INV,
sets STO

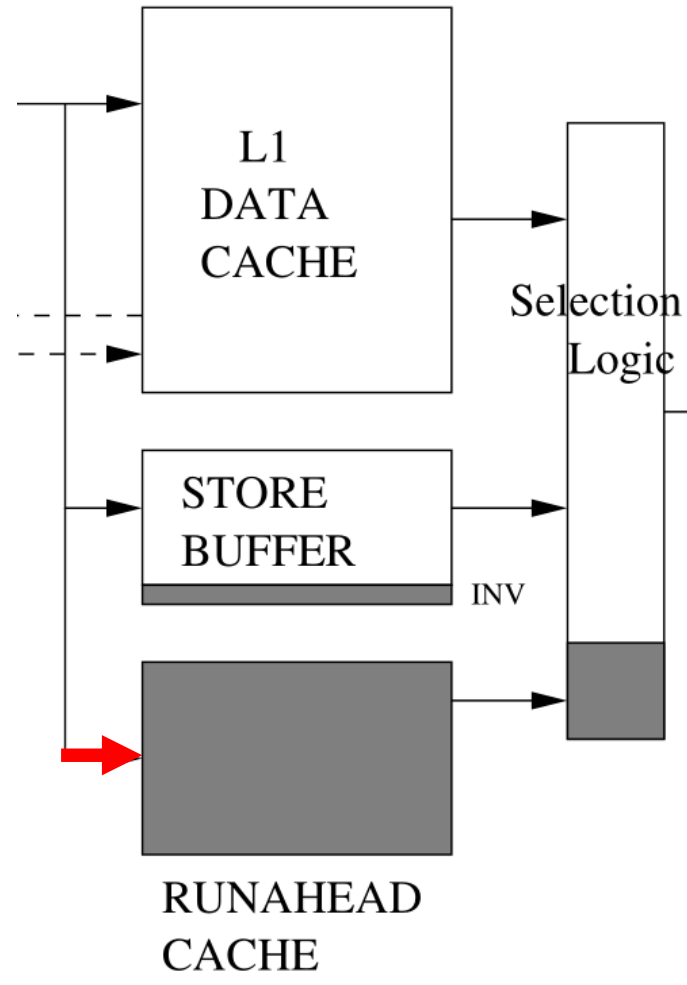
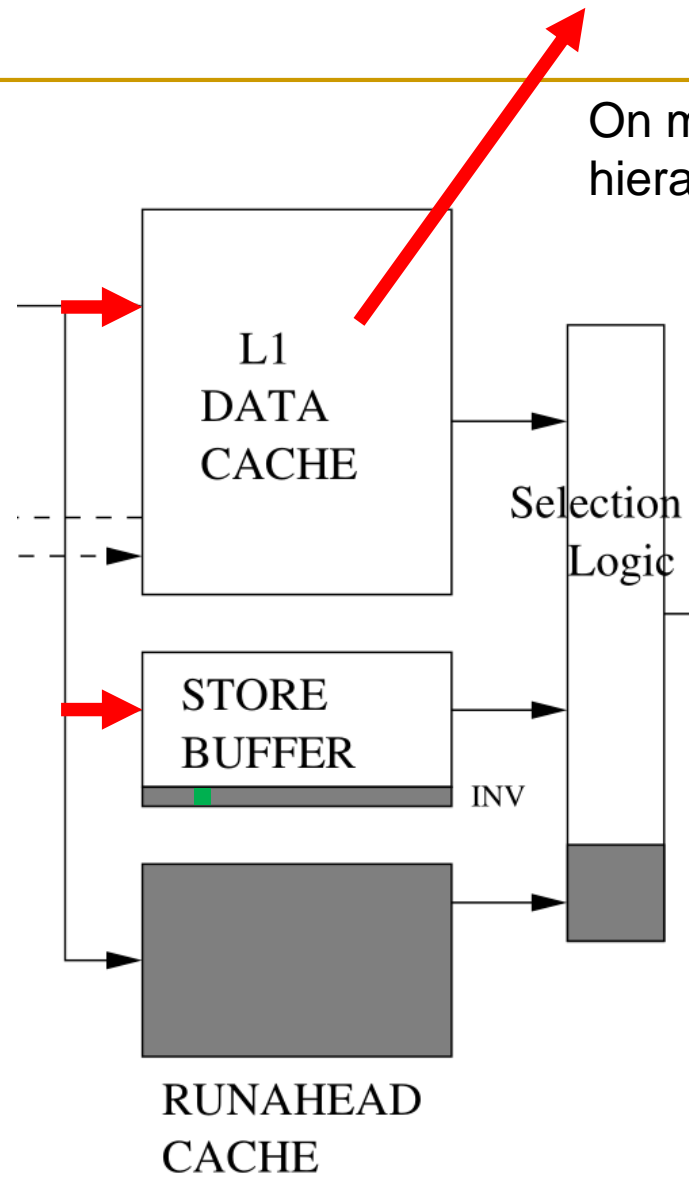


Image:
Mutlu, O.
HPCA03

Store operations

Valid store instruction executed
Requests the affected cache line

Clears the invalid bit of the store
buffer



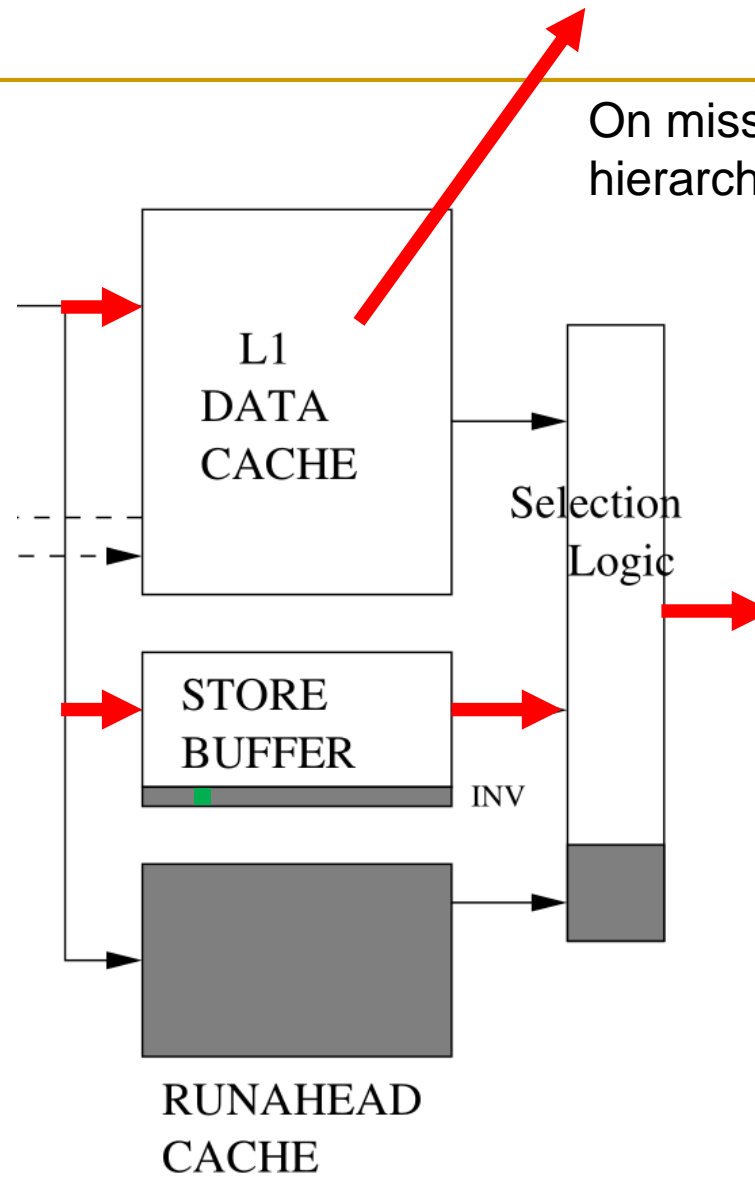
On miss: Propagate through
hierarchy

Image:
Mutlu, O.
HPCA03

Load operations

Valid store instruction executed
Requests the affected cache line

Clears the invalid bit of the store buffer



On miss: Propagate through hierarchy

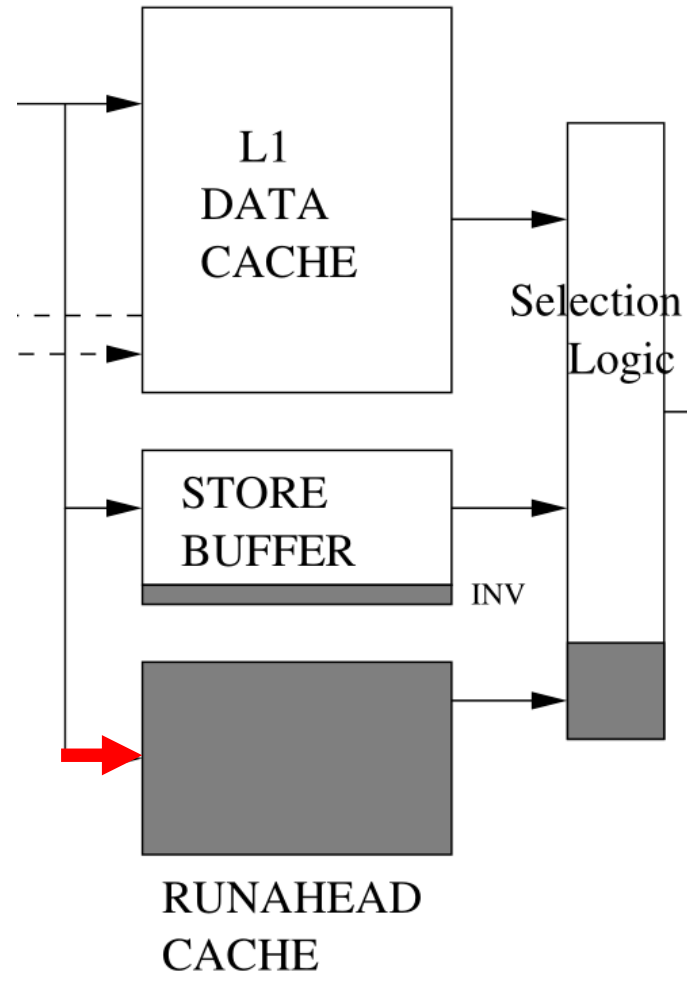
```
mov dword ptr[esp+8], eax
// some instructions
mov ebx, dword ptr[esp+8]
mov ecx, dword ptr[ebx]
```

ebx is now valid

Image:
Mutlu, O.
HPCA03

Store operations

Valid store instruction retired

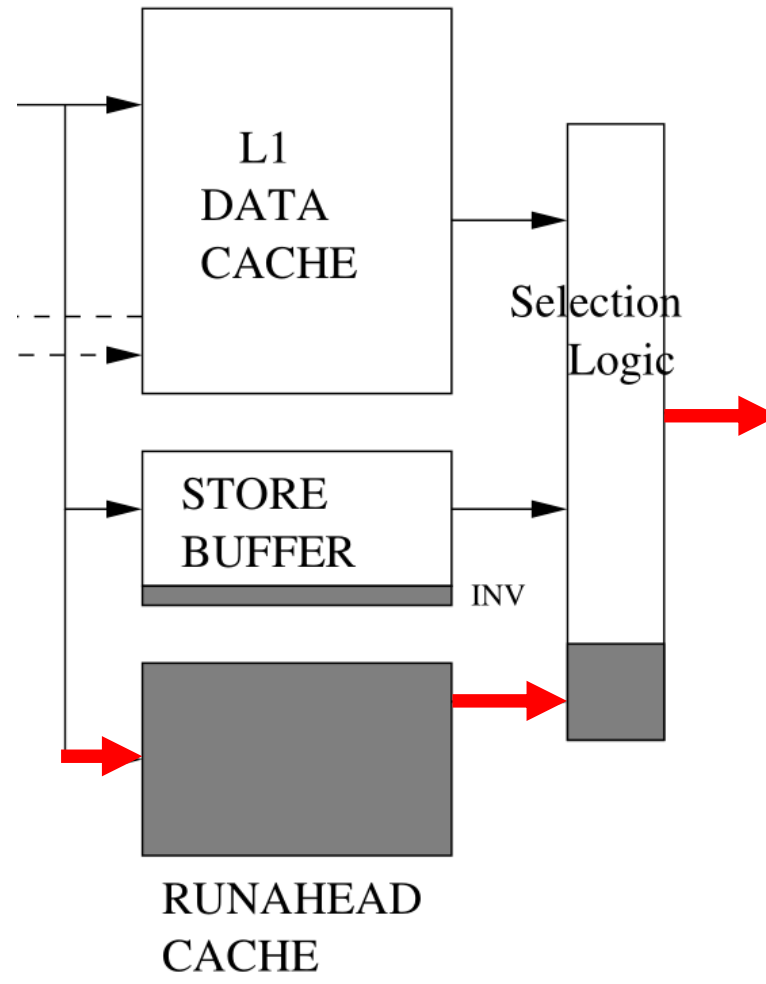


Writes value, clears INV, sets STO

Image:
Mutlu, O.
HPCA03

Load operations

Valid store instruction retired



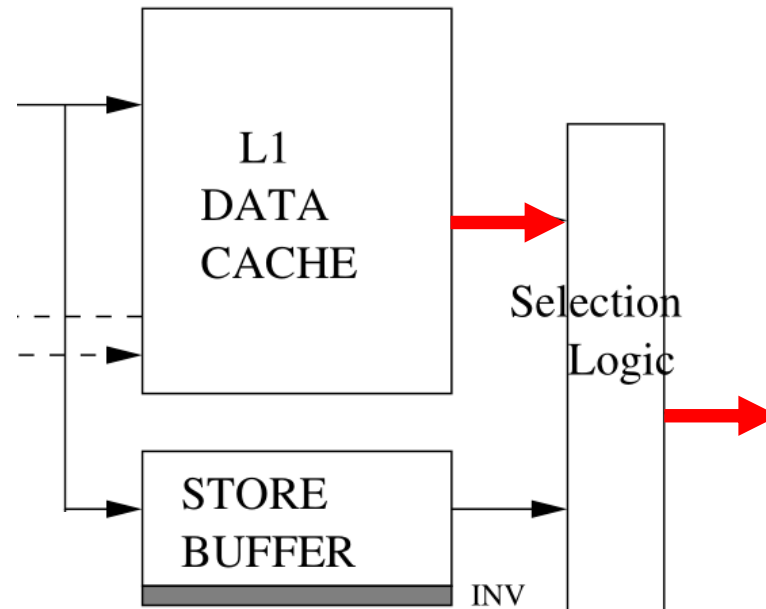
```
mov dword ptr[esp+8], eax
// many instructions
mov ebx, dword ptr[esp+8]
mov ecx, dword ptr[ebx]
```

ebx is now valid

Image:
Mutlu, O.
HPCA03

Valid store instruction retired

Store decays to NOP

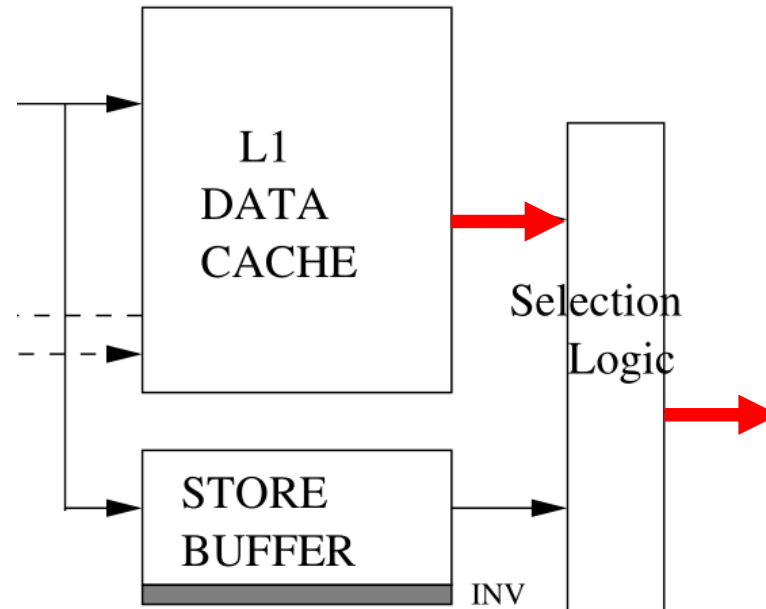


```
mov dword ptr[esp+8], eax
// many instructions
mov ebx, dword ptr[esp+8]
mov ecx, dword ptr[ebx]
```

ebx is now marked
valid, but is actually
stale

Invalid store instruction retired

Store decays to NOP

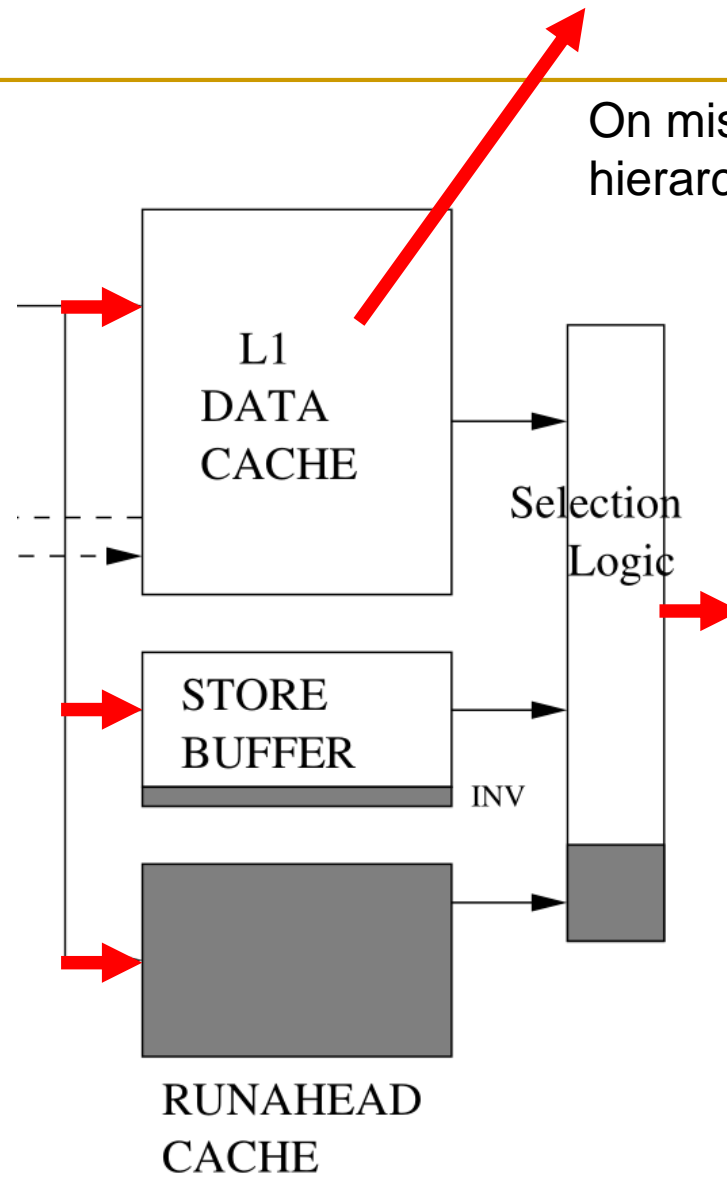


```
mov dword ptr[esp+8], eax
// many instructions
mov ebx, dword ptr[esp+8]
mov ecx, dword ptr[ebx]
```

ebx is now marked
valid, but is actually
stale and invalid

Load operations

Store Buffer \Rightarrow R. Cache \Rightarrow L1 \Rightarrow Miss



On miss: Propagate through hierarchy

Image:
Mutlu, O.
HPCA03

Key Results:

Methodology and Evaluation

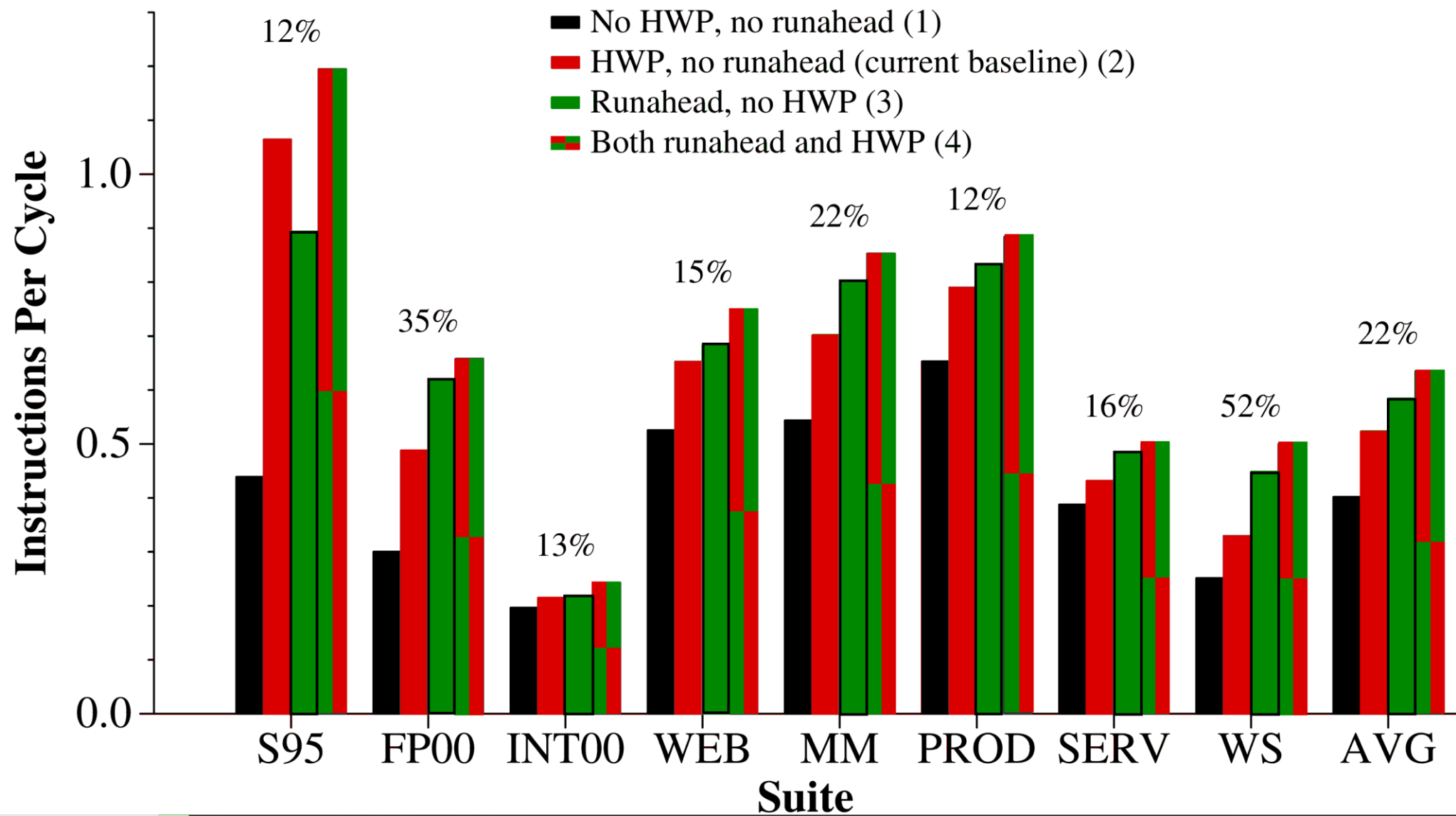
Methodology

- Running Long Instruction Traces (LITs) in a simulator
 - Each LIT is $30 \cdot 10^6$ instructions
 - Chosen to be representative of benchmark
 - Injected instructions to simulate interrupts
 - In total 147 LITs

Target Machine



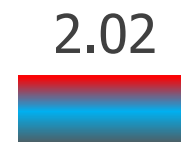
Core Frequency	4 GHz
Instruction Window Size	128
Scheduling Window Size	16 int, 8 mem, 24 fp
Load and store buffer size	48 load, 32 store
L1 Cache	32 KB 8-way
L2 Cache	512 KB 8-way
Bus Latency (L2 Miss Latency)	495 CPU cycles



Mutlu, O.
HPCA03
(recolored)

% cycles w/ full inst window stalls

100
90
80
70
60
50
40
30
20
10
0



Scheduling Window:
L2:
Instruction Window:
Runahead

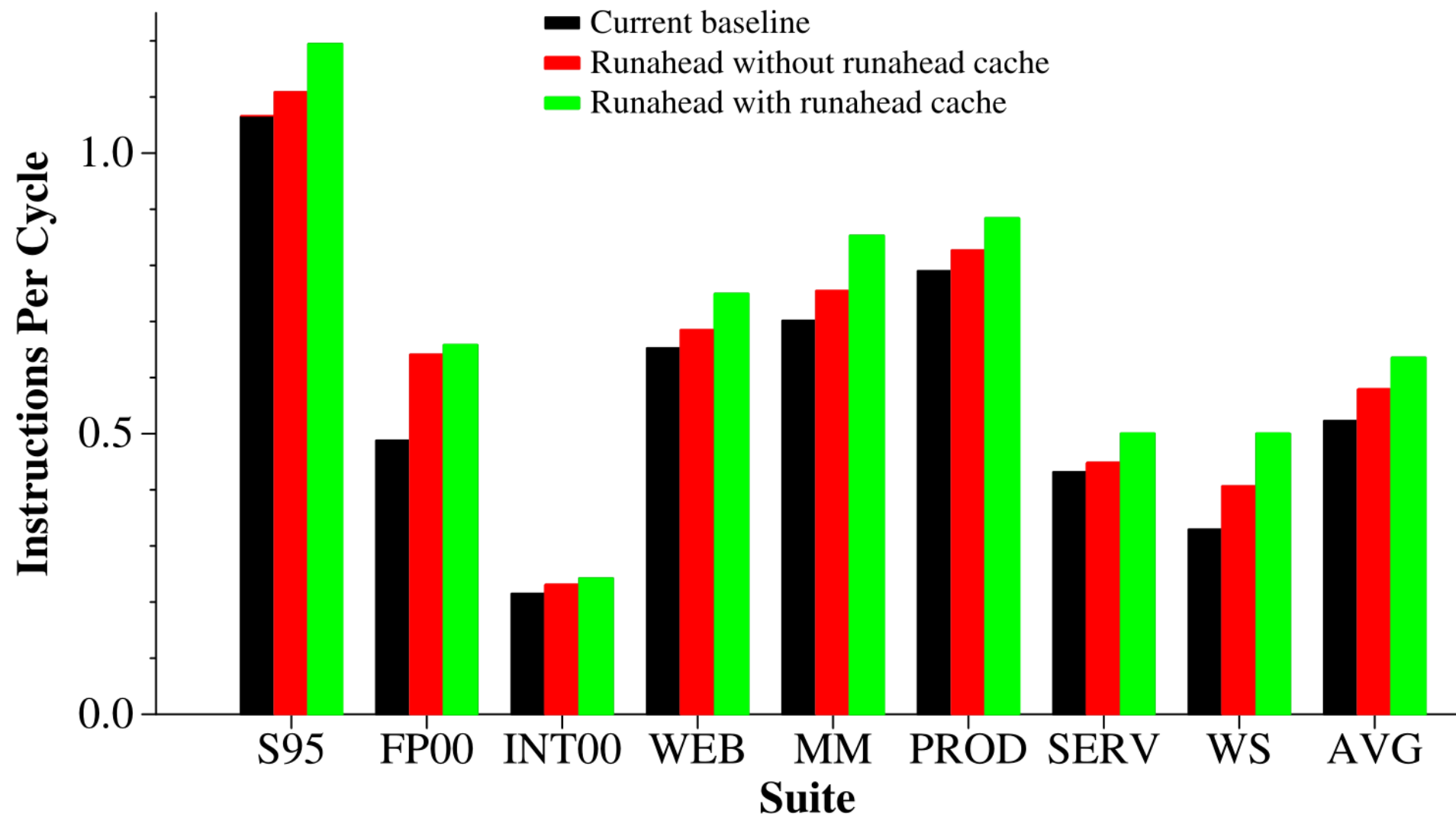
finite | **infinite**
real | **perfect**
128 | **2048**

Numbers over bars: IPC

- finite sched, real L2, 128-entry inst window
- infinite sched, real L2, 128-entry inst window
- infinite sched, real L2, 2048-entry inst window
- infinite sched, perfect L2, 2048-entry inst window
- finite sched, runahead on real L2, 128-entry inst window

Mutlu, O.
HPCA03

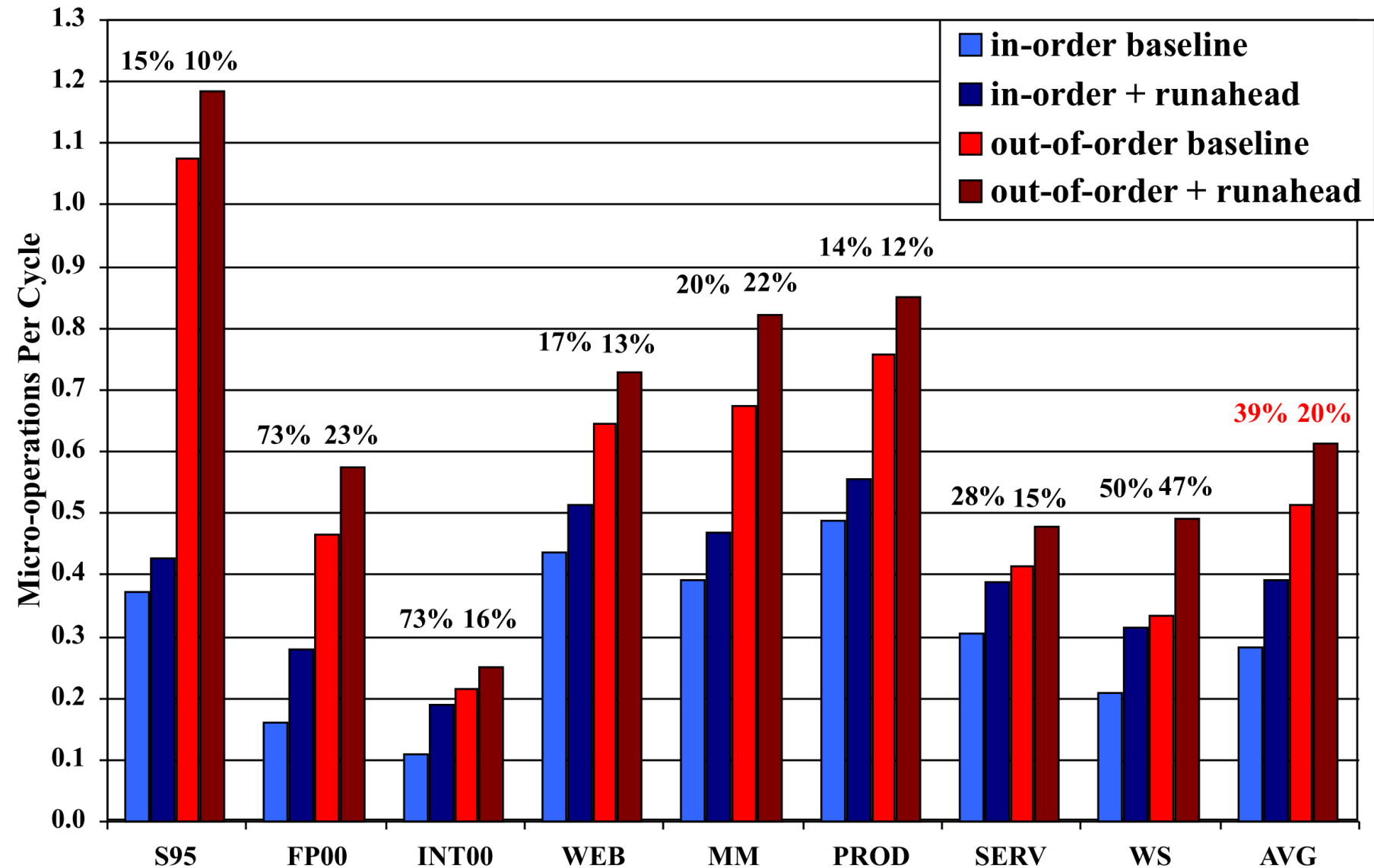
(re-designed)

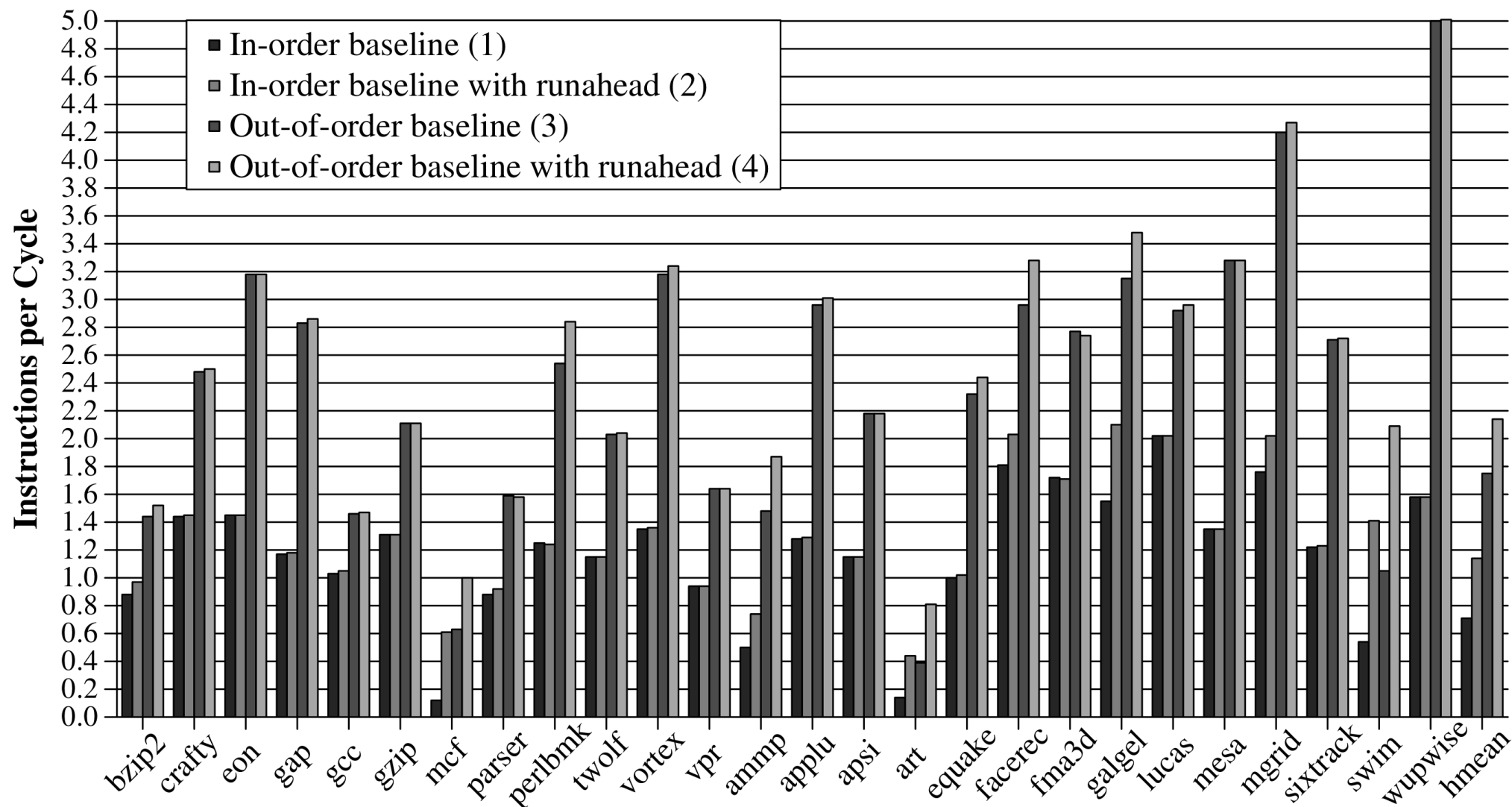


Mutlu, O.
HPCA03

(recolored)

Runahead on In-order vs. Out-of-order





Summary

Summary

■ Goal

- Efficiently increase performance by removing the bottleneck of memory latency

■ Mechanisms

- Transform the blocking instruction window into a nonblocking instruction window
- Add a runahead cache to delay the divergence point

■ Results

- Runahead itself gives a performance increase of 11% on the evaluated workload
- When working with a runahead cache, this improvement is doubled to 20%

Strengths

Strengths

- Small changes with big effects
- Allows for combination with other optimizations
- Successful adaption and extension of in-order runahead
- Increases Memory Level Parallelism (MLP)
- Well-written

Weaknesses

Weaknesses

- Parts of paper did not age well
- Missing/Hidden information in paper
 - e.g. What happens on a page fault?
- Limited by memory bandwidth
- Prefetch distance limited by memory speed
 - The faster a full window stall resolves, the less prefetch requests are generated

Future?

PARAMETER	CURRENT	FUTURE
Processor Frequency	4 GHz	8 GHz
Fetch/Issue/Retire Width	3	6
Branch Misprediction Penalty	29 stages	58 stages
Instruction window size	128	512
Scheduling window size	16 int, 8 mem, 24 fp	64 int, 32 mem, 96 fp
Load and store buffer sizes	48 load, 32 store	192 load, 128 store
Functional units	3 int, 2 mem, 1 fp	6 int, 4 mem, 2 fp
Branch predictor	1000-entry 32-bit history perceptron [15]	3000-entry 32-bit history perceptron
Hardware Data Prefetcher	Stream-based (16 streams)	Stream-based (16 streams)
Trace Cache	12k-uops, 8-way	64k-uops, 8-way
Memory Disambiguation	Perfect	Perfect

Memory Subsystem

L1 Data Cache	32 KB, 8-way, 64-byte line size	64 KB, 8-way, 64-byte line size
L1 Data Cache Hit Latency	3 cycles	6 cycles
L1 Data Cache Bandwidth	512 GB/s, 2 accesses/cycle	4 TB/s, 4 accesses/cycle
L2 Unified Cache	512 KB, 8-way, 64-byte line size	1 MB, 8-way, 64-byte line size
L2 Unified Cache Hit Latency	16 cycles	32 cycles
L2 Unified Cache Bandwidth	128 GB/s	256 GB/s
Bus Latency	495 processor cycles	1008 processor cycles
Bus Bandwidth	4.25 GB/s	8.5 GB/s
Max Pending Bus Transactions	10	20

Table 2. Parameters for Current and Future Baselines.

Future?

PARAMETER	CURRENT	FUTURE	Now
Processor Frequency	4 GHz	8 GHz	5 GHz
Fetch/Issue/Retire Width	3	6	
Branch Misprediction Penalty	29 stages	58 stages	
Instruction window size	128	512	224
Scheduling window size	16 int, 8 mem, 24 fp	64 int, 32 mem, 96 fp	97 unified
Load and store buffer sizes	48 load, 32 store	192 load, 128 store	72 load, 56 store
Functional units	3 int, 2 mem, 1 fp	6 int, 4 mem, 2 fp	
Branch predictor	1000-entry 32-bit history perceptron [15]	3000-entry 32-bit history perceptron	
Hardware Data Prefetcher	Stream-based (16 streams)	Stream-based (16 streams)	
Trace Cache	12k-uops, 8-way	64k-uops, 8-way	
Memory Disambiguation	Perfect	Perfect	

Memory Subsystem

L1 Data Cache	32 KB, 8-way, 64-byte line size	64 KB, 8-way, 64-byte line size	32 KB, 8-way, 64-byte line size
L1 Data Cache Hit Latency	3 cycles	6 cycles	5 cycles
L1 Data Cache Bandwidth	512 GB/s, 2 accesses/cycle	4 TB/s, 4 accesses/cycle	
L2 Unified Cache	512 KB, 8-way, 64-byte line size	1 MB, 8-way, 64-byte line size	256KB, 4-way, 64-byte line size
L2 Unified Cache Hit Latency	16 cycles	32 cycles	12 cycles
L2 Unified Cache Bandwidth	128 GB/s	256 GB/s	
Bus Latency	495 processor cycles	1008 processor cycles	320 cycles-ish (80ns / 4 GHz)
Bus Bandwidth	4.25 GB/s	8.5 GB/s	
Max Pending Bus Transactions	10	20	

Table 2. Parameters for Current and Future Baselines.

Thoughts and Ideas

Sun Rock

- <https://arstechnica.com/gadgets/2008/02/sun-can-you-smell-what-the-rock-is-cookin/>
- Magic Everything-CPU
 - ❑ Out-of-order retirement
 - ❑ Hardware scout
 - ❑ Hardware Transactional Memory
- Cancelled in 2010
- “This processor had two incredible virtues: It was incredibly slow and it consumed vast amounts of energy. It was so hot that they had to put about 12 inches of cooling fans on top of it to cool the processor,” said [Larry] Ellison. “It was just madness to continue that project.”
- Chaudhry, Shailender, et al. "High-performance throughput computing." *IEEE Micro* 25.3 (2005): 32-45.
- <https://www.reuters.com/article/us-oracle/special-report-can-that-guy-in-ironman-2-whip-ibm-in-real-life-idUSTRE64B5YX20100512>, accessed 1.11.18

Thoughts and ideas

- How to reuse the added structures?
 - Easier hardware debugging by having the architectural register file collected anyways
 - Adding instructions to use runahead cache as a scratch buffer?
 - As transactional memory?
 - Using the checkpointed architectural register file for context switches?
 - pushad

Takeaways

Takeaways

- It is easier to reuse resources
- Adapting existing techniques might work very well

Further reading

- Mutlu, Onur. *Efficient runahead execution processors*. Diss. 2006.
- Mutlu, Onur, Hyesoon Kim, and Yale N. Patt. "Efficient runahead execution: Power-efficient memory latency tolerance." *IEEE Micro* 26.1 (2006): 10-20.
- Mutlu, Onur, et al. "On reusing the results of pre-executed instructions in a runahead execution processor." *IEEE Computer Architecture Letters* 4.1 (2005): 2-2.
- Chappell, Robert S., et al. "Simultaneous subordinate microthreading (SSMT)." *Computer Architecture, 1999. Proceedings of the 26th International Symposium on*. IEEE, 1999.
- Hashemi, Milad, Onur Mutlu, and Yale N. Patt. "Continuous runahead: Transparent hardware acceleration for memory intensive workloads." *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Press, 2016.
- Ramirez, Tanausu, et al. "Runahead threads to improve SMT performance." *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*. IEEE, 2008.
- Chaudhry, Shailender, et al. "High-performance throughput computing." *IEEE Micro* 25.3 (2005): 32-45.
- Cain, Harold W., and Priya Nagpurkar. "Runahead execution vs. conventional data prefetching in the IBM POWER6 microprocessor." *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on*. IEEE, 2010.
- "Port Contention for Fun and Profit" (brand new, not published yet)

Questions

Open Discussion

Open Discussion

- What's a simple worst case where Runahead Execution would not give any benefits?
- Would it be beneficial to also catch and treat page faults in runahead mode?
- If you had to choose between SMT and Runahead Execution: Which one?
 - It is possible to combine them (at a small cost). Is there a reason you would not want to?
 - SMT leak: "Port Contention for Fun and Profit" ("PortSmash") CVE-2018-5407
- Runahead Execution implemented in in-Order CPUs, but not in OoO-CPU's
 - Why?
 - How does the addition of L3-cache impact Runahead Execution?
 - What if instead of having an L3, the L2 was just bigger? What changes?

Open Discussion

- Intel Atom processors used to be in-Order Architectures, but did not feature runahead execution. Why?
- Other ideas for runahead execution?
 - Continuous Runahead Execution
 - Subordinate Simultaneous Multithreading
- Other ideas to overcome the memory wall?

Backup Slides














Intel® Pentium® 4 Processor supporting HT Technology 3.40E GHz, 1M Cache, 800 MHz FSB

☐ Add to Compare

- Specifications
- Essentials
- Performance
- Supplemental Information
- Package Specifications
- Advanced Technologies
- Security & Reliability
- Product Images
- Downloads and Software

Essentials


[Export specifications](#)

Product Collection	Legacy Intel® Pentium® Processor
Code Name	Products formerly Prescott
Vertical Segment	Desktop
Status	Discontinued
Launch Date 	Q1'04
Lithography 	90 nm
Recommended Customer Price 	N/A
<div>Performance</div>	
# of Cores 	1
Processor Base Frequency 	3.40 GHz
Cache 	1 MB L2
Bus Speed 	800 MHz FSB
FSB Parity 	No
TDP 	103 W
VID Voltage Range 	1.250V-1.400V

TDP [?](#) 103 W

VID Voltage Range [?](#) 1.250V-1.400V

% cycles w/ full inst window stalls

Scheduling Window:
L2:
Instruction Window:
Runahead

finite | infinite
real | perfect
128 | 2048

Numbers over bars: IPC

- finite sched, real L2, 128-entry inst window
- infinite sched, real L2, 128-entry inst window
- finite sched, perfect L2, 128-entry inst window
- infinite sched, perfect L2, 128-entry inst window
- infinite sched, real L2, 2048-entry inst window
- infinite sched, perfect L2, 2048-entry inst window
- finite sched, runahead on real L2, 128-entry inst window

Mutlu, O.
HPCA03

(re-designed)

