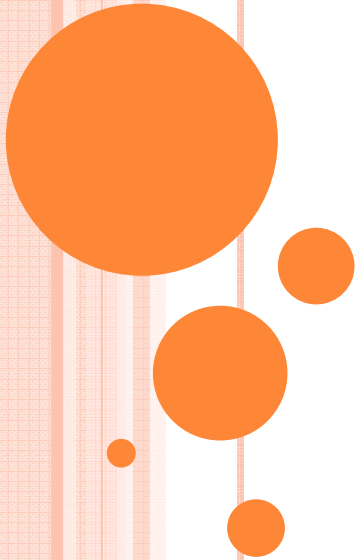


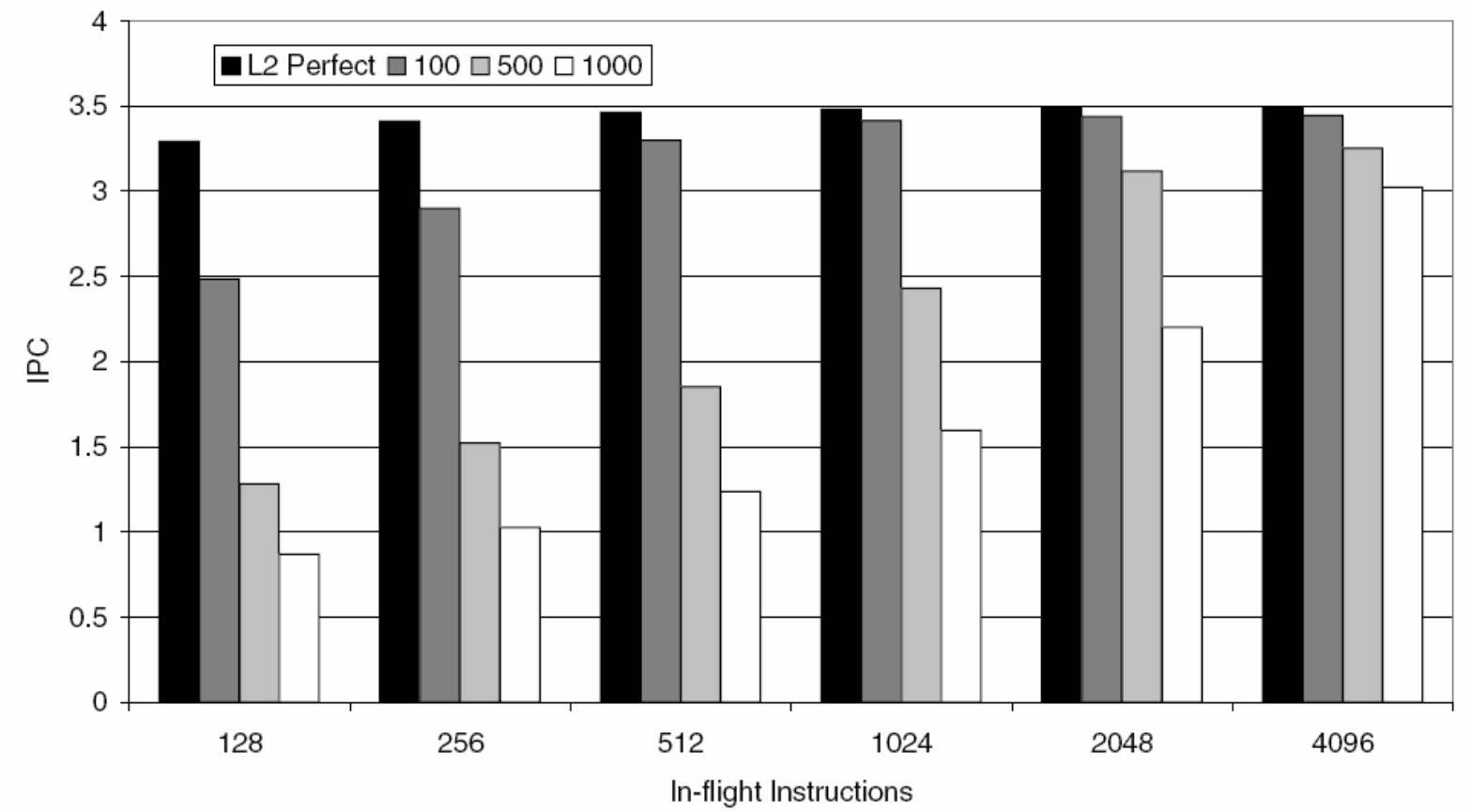
# OUT-OF-ORDER COMMİT PROCESSORS

**Paper by; Adrian Cristal, Daniel Ortega, Josep Llosa and Mateo Valero**

# INTRODUCTİON

- Increasing gap between processor speed and memory speed is steadily increasing memory latencies with each new processor generation.
- Tolerate these latencies, caches and prefetching are very useful techniques.
- Processors cannot keep up with growing disparity(higher ILP), and as a result, long latency operations are increasingly more crucial for performance.

# A SAMPLE GRAPHİC



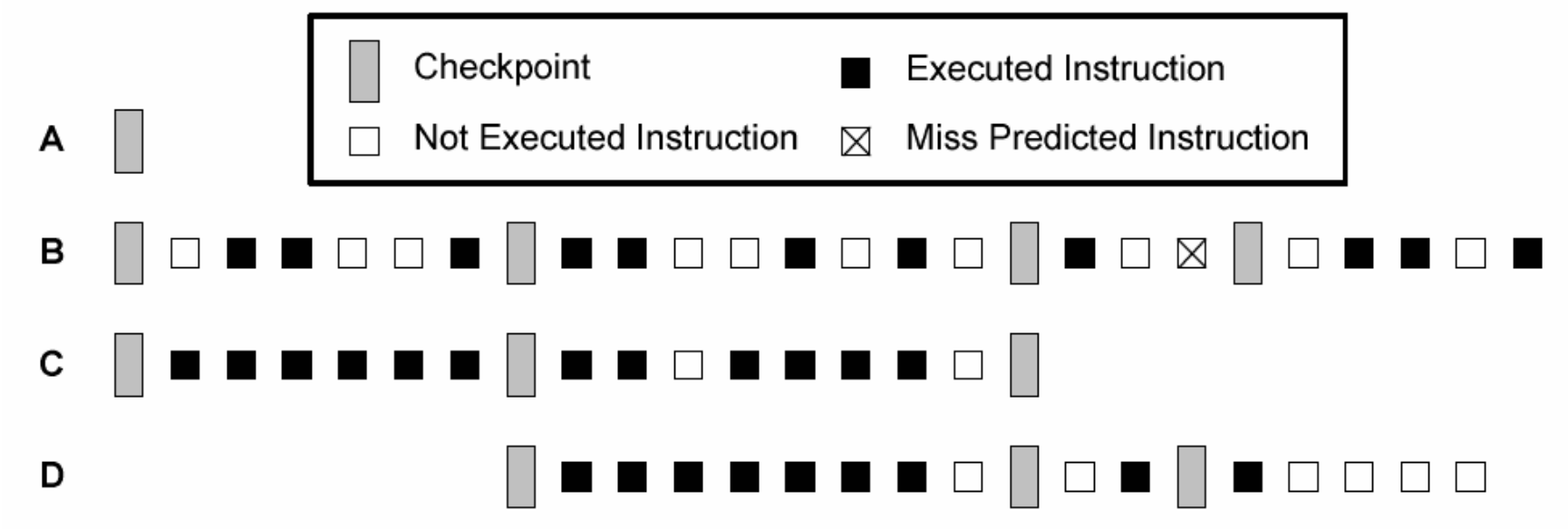

increasing the number of in-flight instructions is capable of achieving nearly perfect memory behavior

# OUT-OF-ORDER COMMİT

- After an instruction is fetched and decoded in a superscalar out-of-order processor, it is inserted in both its corresponding instruction queue and in the re-order buffer(ROB).
- Allows for perfect interrupts by implementing in-order commit.
- Controls exactly when stores may change the memory state (thus the machine state). It also correctly frees the physical registers when they are no longer in use.
- Keeps a history window of all in-flight instructions, as the latency of memory operations increases, it is needed to support a larger number of in-flight instructions to hide this access latency and achieve high performance.

# CHECKPOİNTİNG PROCESS

- Scaling-up the number of entries in this structure is impractical (due to cycle time limitations)



# CAM REGİSTER MAPPİNG

*CAM Register Mapping*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Physical** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | … |
| **Logical** | 3 | 4 | 2 | 1 | 8 | 9 | 7 | … |
| **Valid** | 1 | 1 | 1 | 1 | 0 | 0 | 0 | … |
| **Future Free** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … |
| **Free List** | 0 | 0 | 0 | 0 | 1 | 1 | 1 | … |

- Only four physical registers are mapped, they are not going to change until these registers are freed and used for a new instruction.
- The cost of a checkpoint in this mechanism can be computed as the number of physical register times two bits per register.

# CAM REGİSTER MAPPİNG

Checkpoint

R1 = R2 + R3

*Ph5  Ph3  Ph1*

***CAM Register Mapping***

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Physical** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | … |
| **Logical** | 3 | 4 | 2 | 1 | 1 | 9 | 7 | … |
| **Valid** | 1 | 1 | 1 | 0 | 1 | 0 | 0 | … |
| **Future Free** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | … |
| **Free List** | 0 | 0 | 0 | 0 | 0 | 1 | 1 | … |

# CAM REGİSTER MAPPİNG

Checkpoint

…

R1 = R4 + R1

*Ph6  Ph2  Ph5*

***CAM Register Mapping***

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Physical** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | … |
| **Logical** | 3 | 4 | 2 | 1 | 1 | 1 | 7 | … |
| **Valid** | 1 | 1 | 1 | 0 | 0 | 1 | 0 | … |
| **Future Free** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | … |
| **Free List** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | … |

# CAM REGİSTER MAPPİNG

Checkpoint

…

R4 = R1 + R3

*Ph7 Ph6 Ph1*

***CAM Register Mapping***

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Physical** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | … |
| **Logical** | 3 | 4 | 2 | 1 | 1 | 1 | 4 | … |
| **Valid** | 1 | 0 | 1 | 0 | 0 | 1 | 1 | … |
| **Future Free** | 0 | 1 | 0 | 1 | 1 | 0 | 0 | … |
| **Free List** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | … |

# INFERENCES

- A checkpoint is taken by storing the Valid and the Future Free bits in our checkpoint table.
- After the checkpoint is taken, all Future Free bits are cleared.
- When an instruction finishes it uses index (from checkpoint table) to decrease this counter. When this counter arrives to zero and this particular checkpoint has no previous checkpoints, this checkpoint has committed and modify the state of the machine to assert this point.

# FREEİNG PHYSİCAL REGİSTERS

- A physical register is normally freed at commit phase of the following instruction that defines the same logical register as the one to which this physical register is mapped.
- When an instruction defines a particular register, it will have to keep the previous mapping of this register so as to free it at commit phase.
- This is handled by the Future Free bits. These bits record which registers need to be freed between checkpoints. When a particular checkpoint commits, it uses these bits to free the registers associated to its history window.
- Increases the lifetime of these registers, (ROB mechanism would be freed beforehand, by their re-defining instructions but in this way, they must wait until the following checkpoint gets committed)

# COMMİTTİNG STORE INSTRUCTİONS

- Stores must wait until commit stage to send their data to memory.
- Normal way; the data is stored in the Load/Store queue until the particular store commits, when it is send to the cache.
- Data is kept in the Load/Store queue and when a checkpoint is committed, all the stores relative to the previous checkpoint are considered to be safe and are sent to memory.

# TAKİNG CHECKPOİNTS

- Take a checkpoint at the first branch after 64 instructions. (select branches as good places to take checkpoints so as to minimize the work done after branch mis-speculation)
- Take a checkpoint after 512 instructions at whatever instruction appears.
- Force a checkpoint after 64 stores.

# SLOW LANE INSTRUCTİON QUEUİNG (SLIQ)

- Some instructions take a very long time to even get issued for execution. Maintaining these instructions in a instruction queue just takes away issue slots from other instructions that will be executed more quickly.

- First of all detecting those instructions which will take a very long time to get issued for execution.

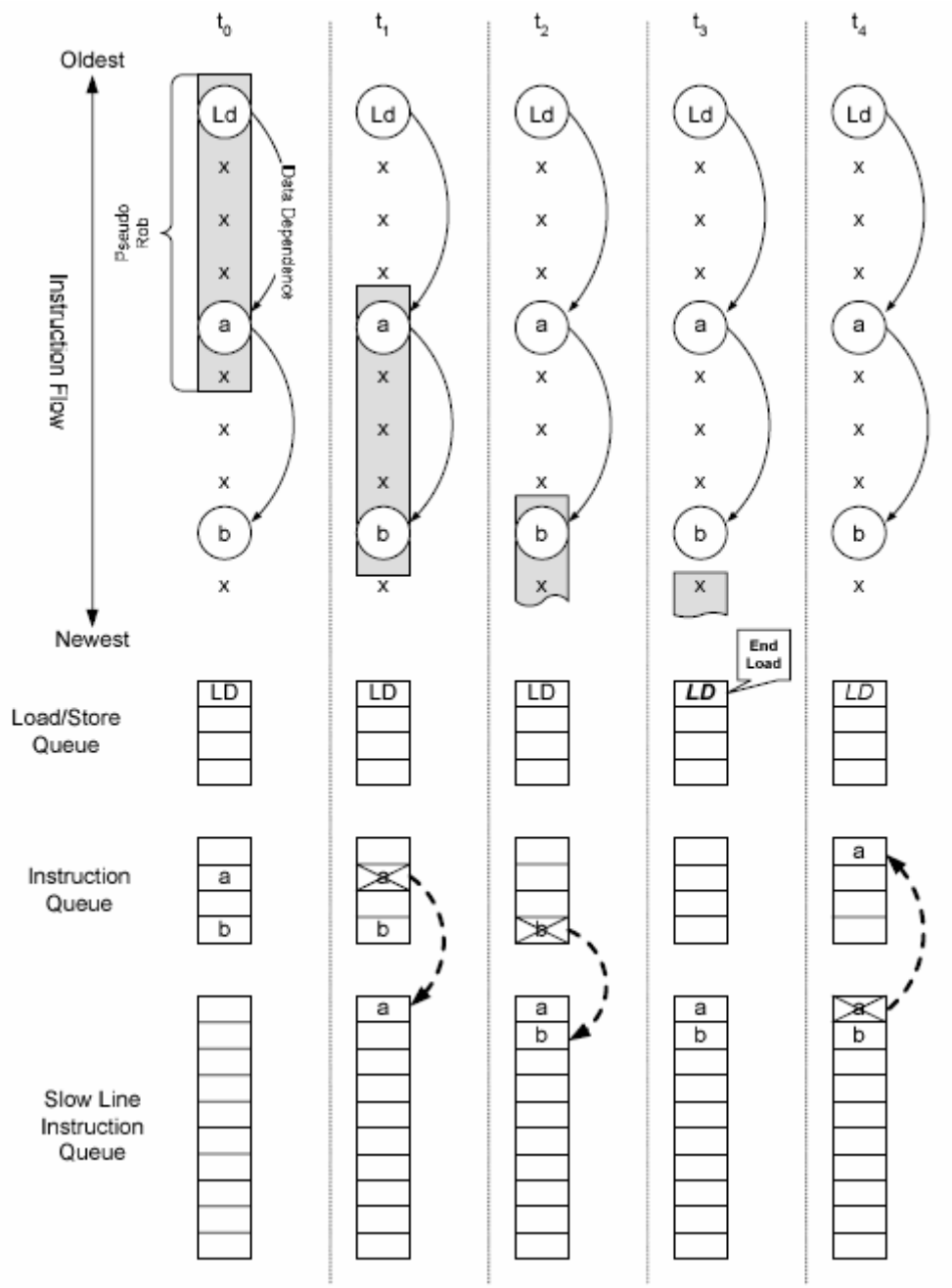
- Move them to a secondary buffer where they would stay until there is any need for them to return to their respective instruction queue.

Load dependency

32 bit register

Move it to SLIQ

FIFO



# CONCLUSİONS

- Check pointed based commit mechanism and Slow Lane Instruction Queuing mechanism possible to implement the functionality of a big ROB and big instruction queues, requiring a reduced number of entries.
- Performance degradation of only 10% over a conventional processor with 4096 entries ROB and instruction queues.
- An increase in performance of 204% relative to a conventional processor with both structures having 128 entries.

# QUESTİONS

?