# CHERRY: CHECKPOINTED EARLY RESOURCE RECYCLING

José F. Martínez[1], Jose Renau[2]

Michael C. Huang[3], Milos Prvulovic[2], and Josep Torrellas[2]

1 CORNELL

2 ILLINOIS UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

3 UNIVERSITY OF ROCHESTER

# MOTIVATION

- **Problem**: Limited processor resources
- **Goal**: More efficient use by aggressive recycling
- **Opportunity**: Resources reserved until retirement

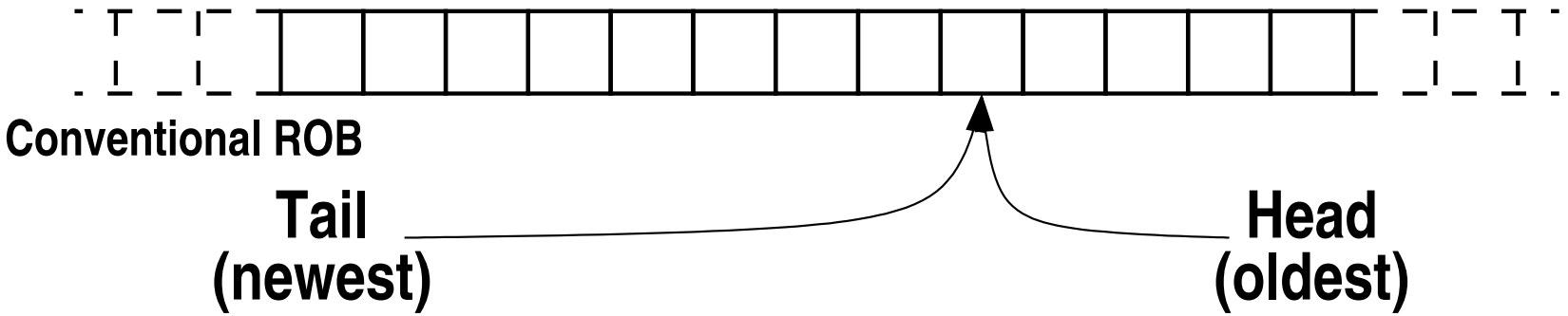
- **Solution**: Decouple recycling from retirement

# CURRENT PROCESSORS

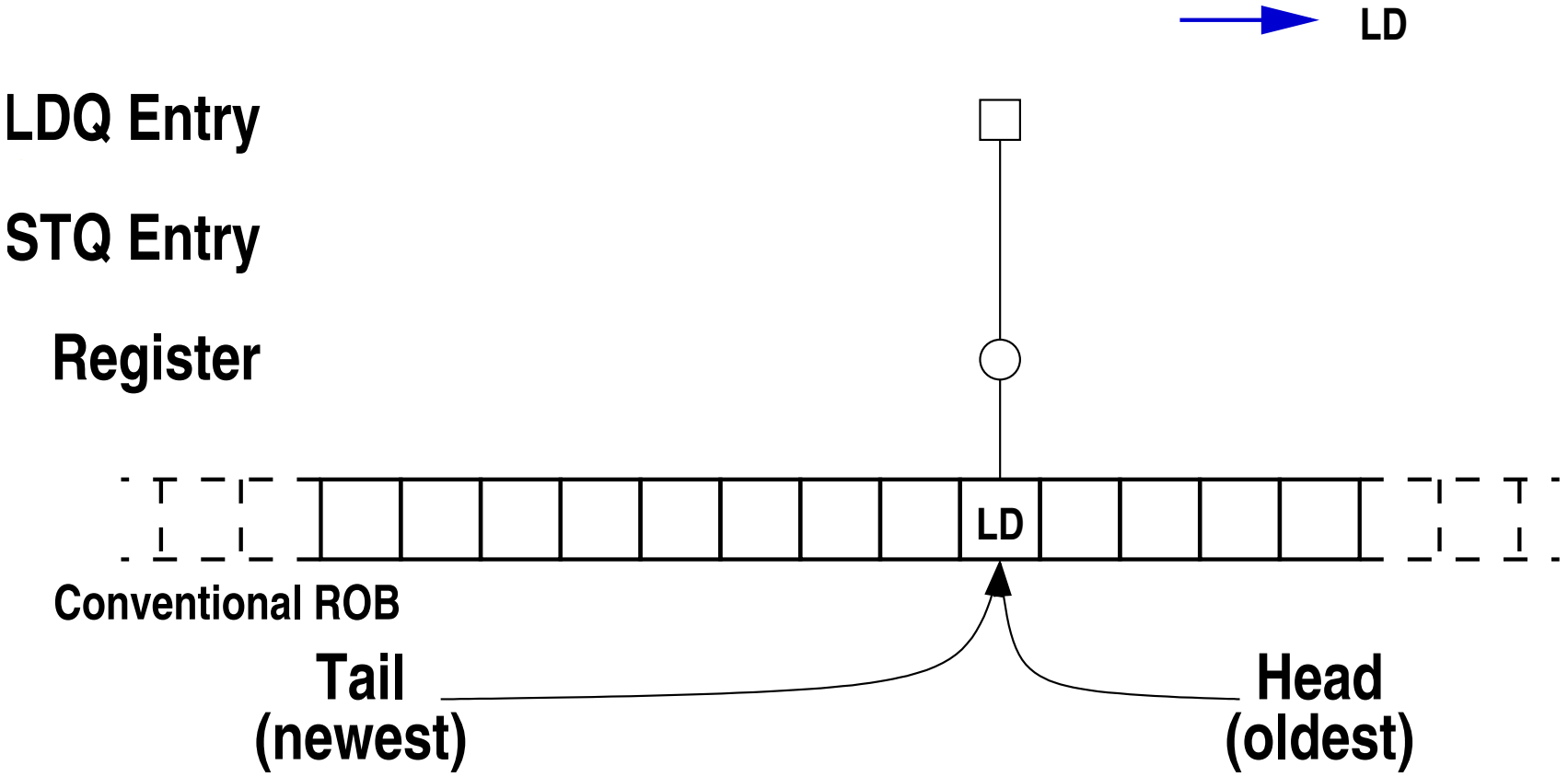
LDQ Entry

STQ Entry

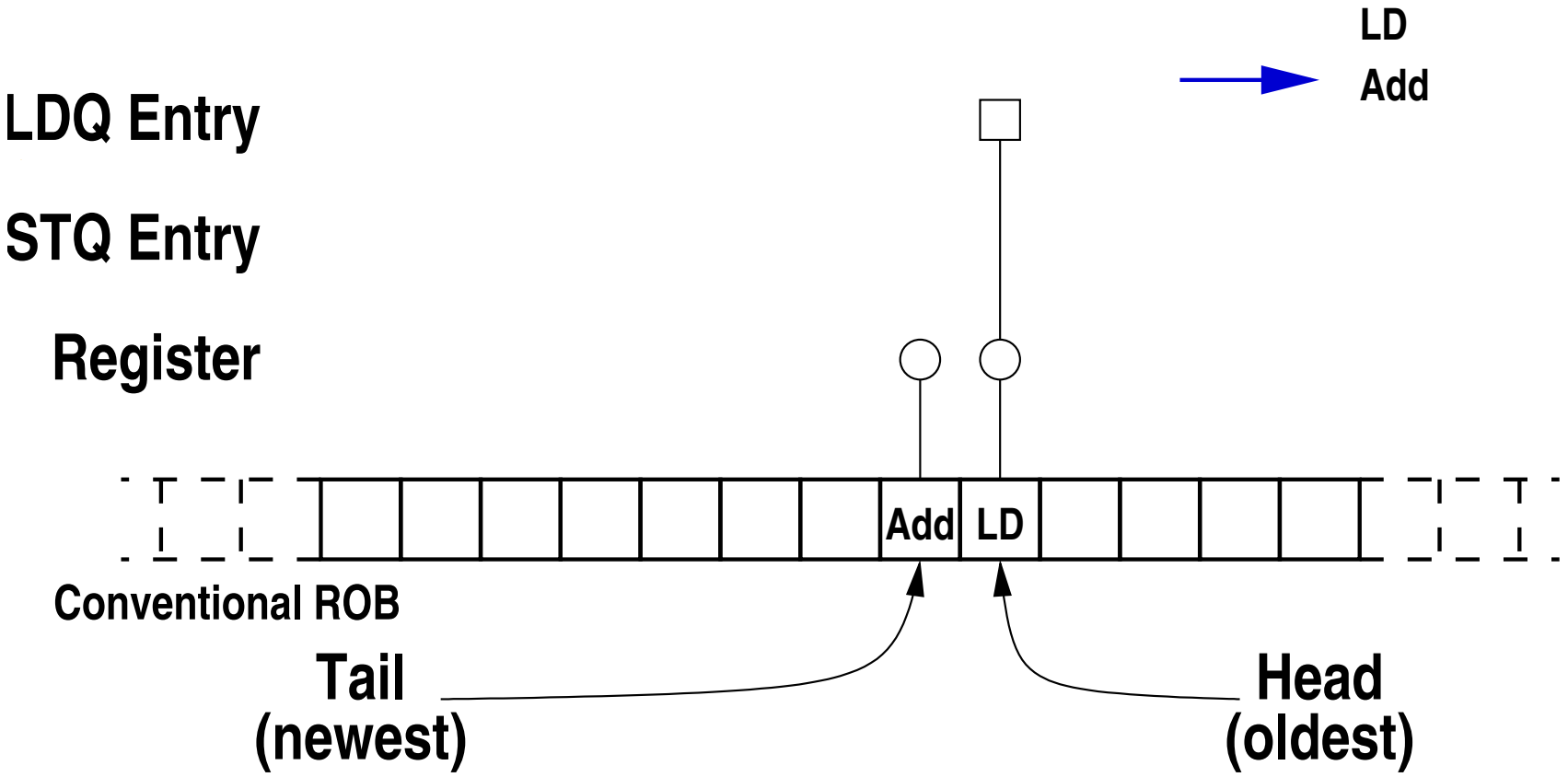
Register



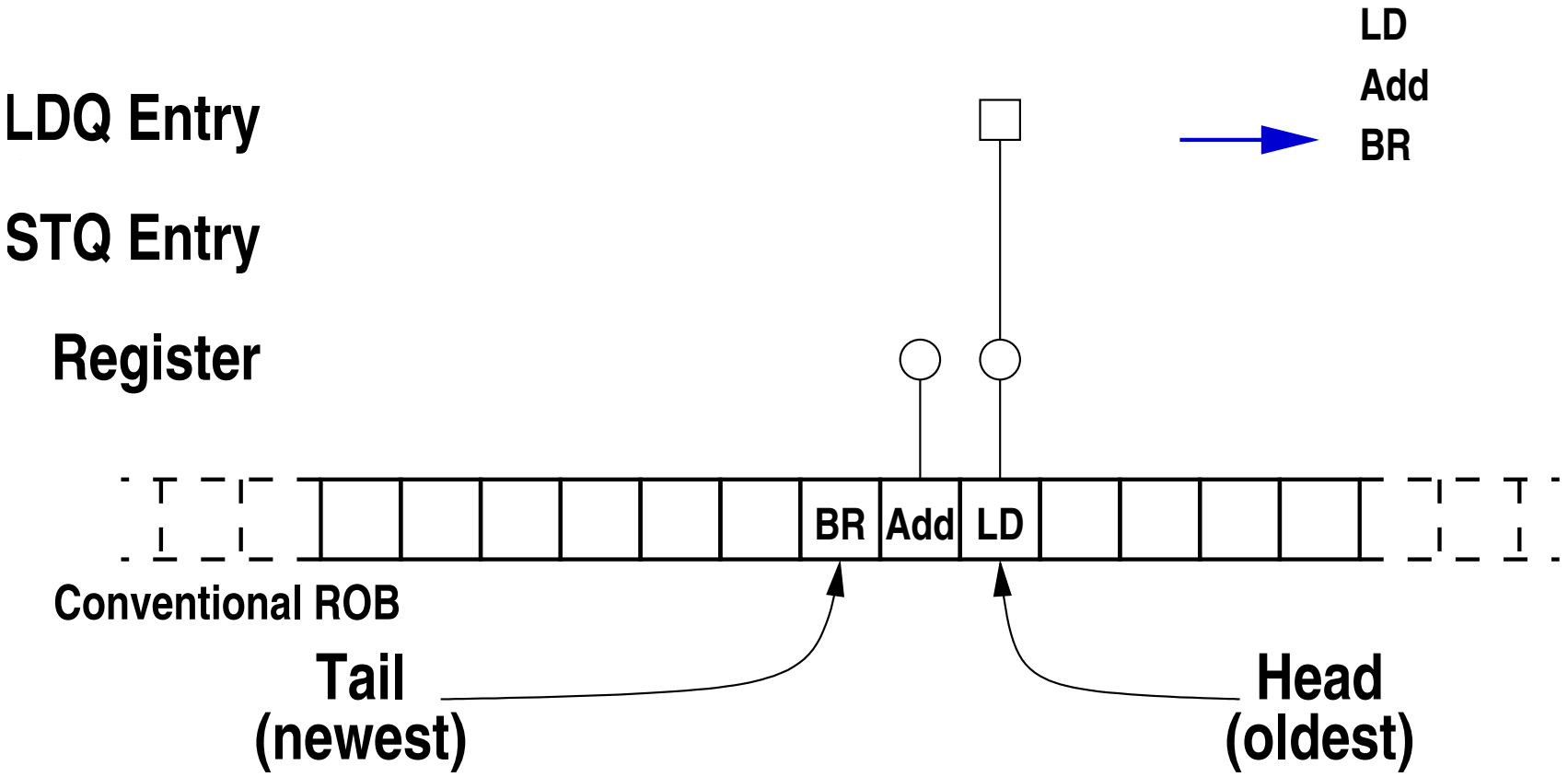
# CURRENT PROCESSORS



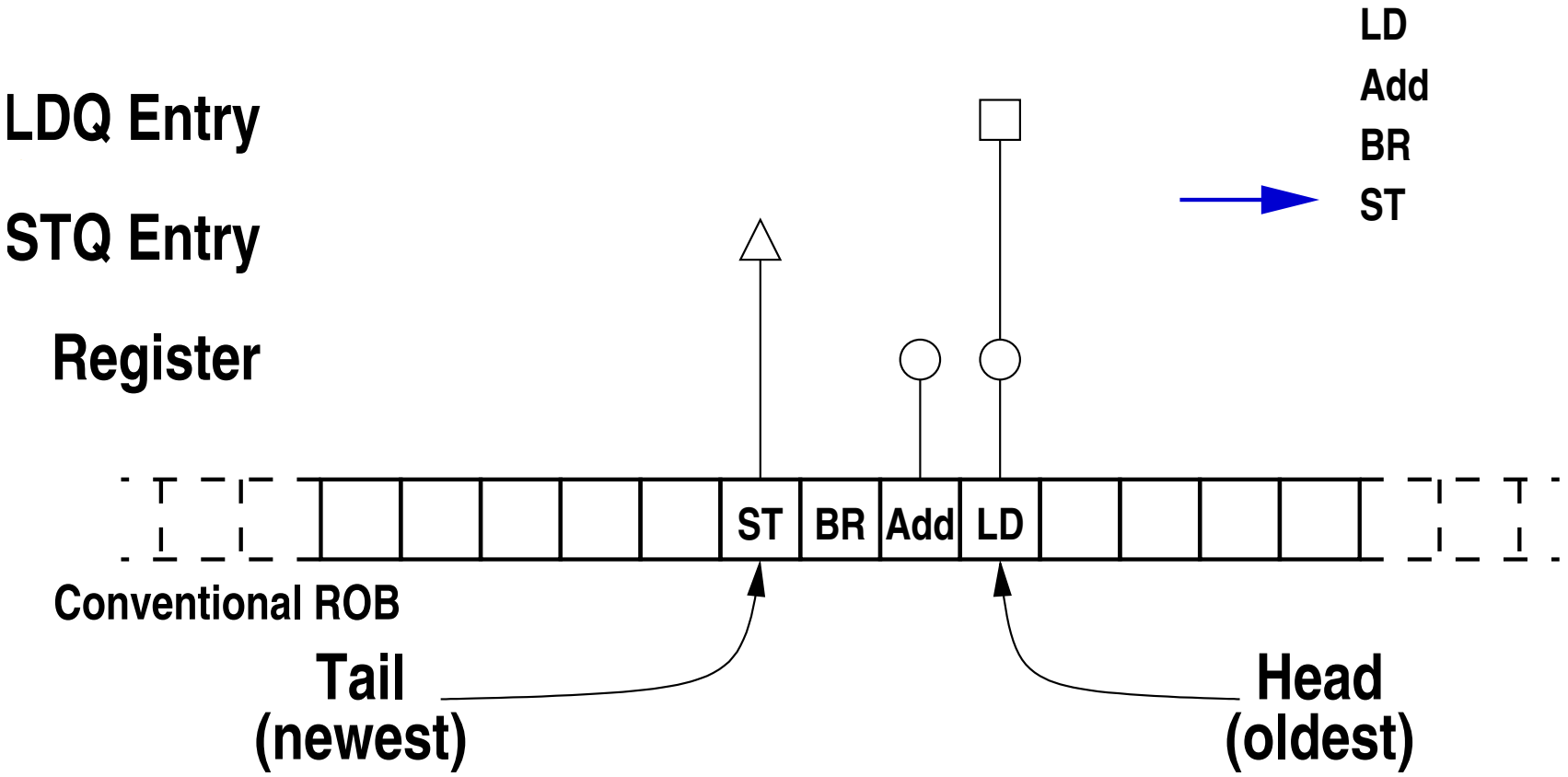
# CURRENT PROCESSORS



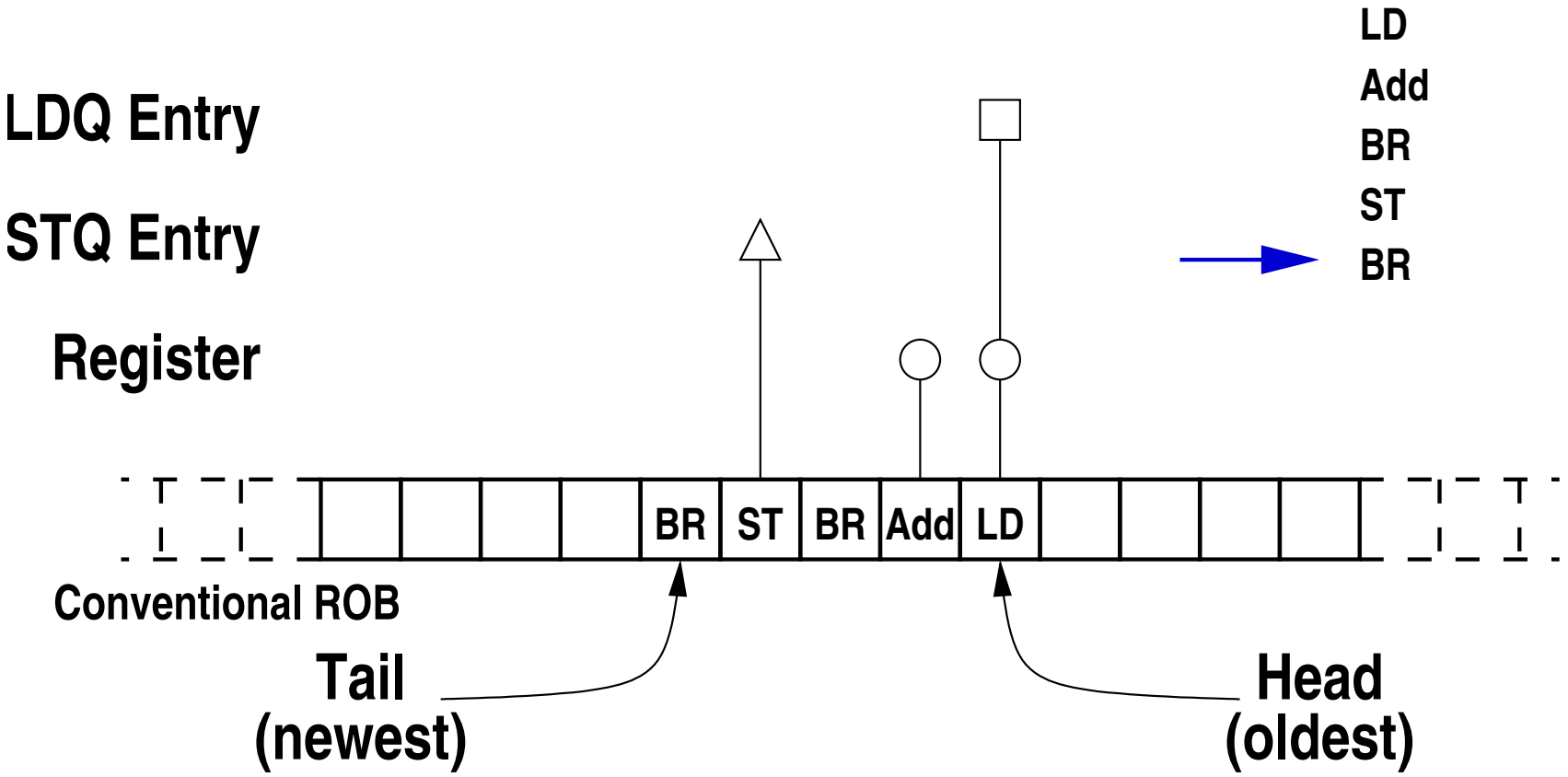
# CURRENT PROCESSORS



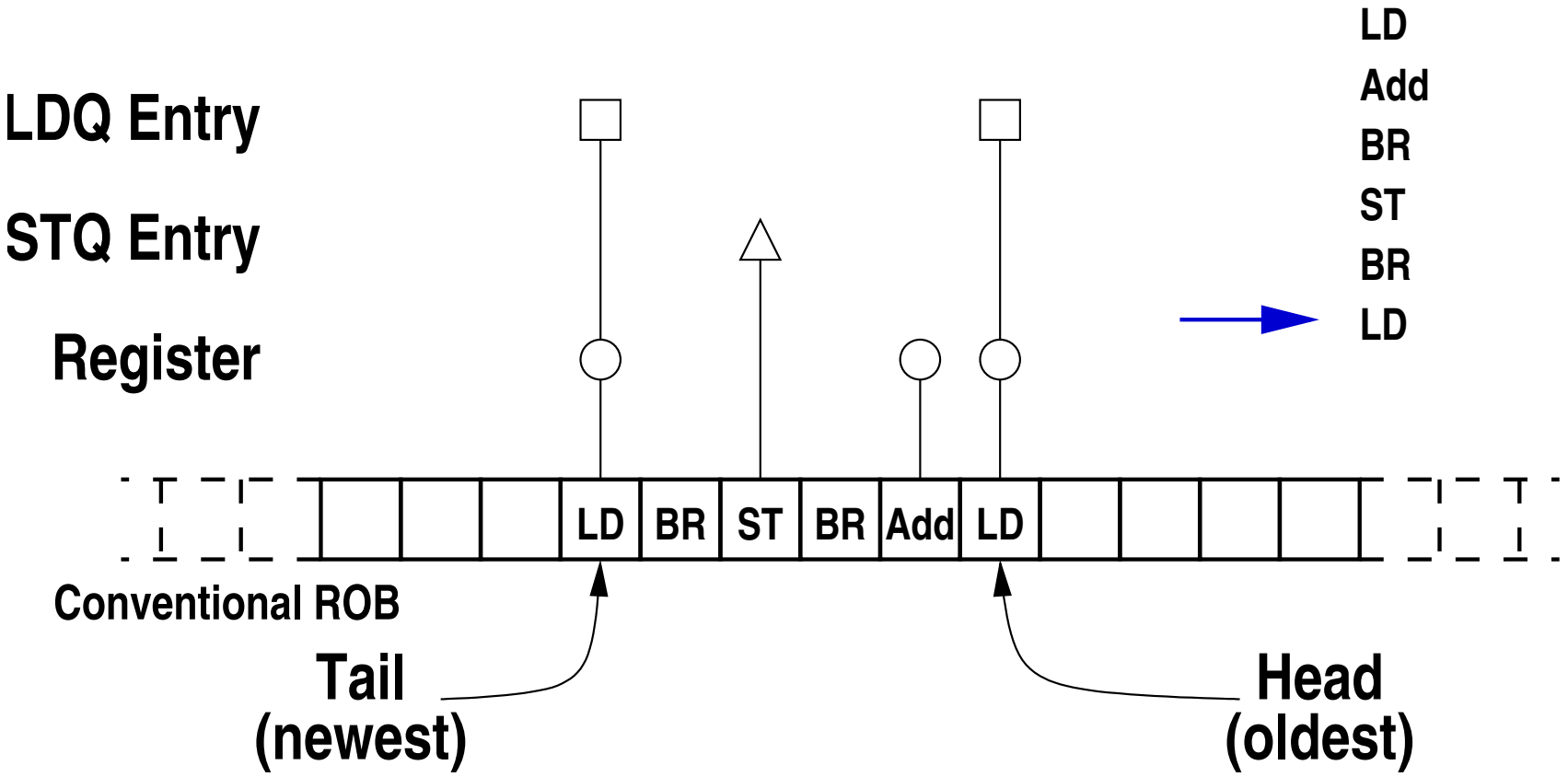
# CURRENT PROCESSORS



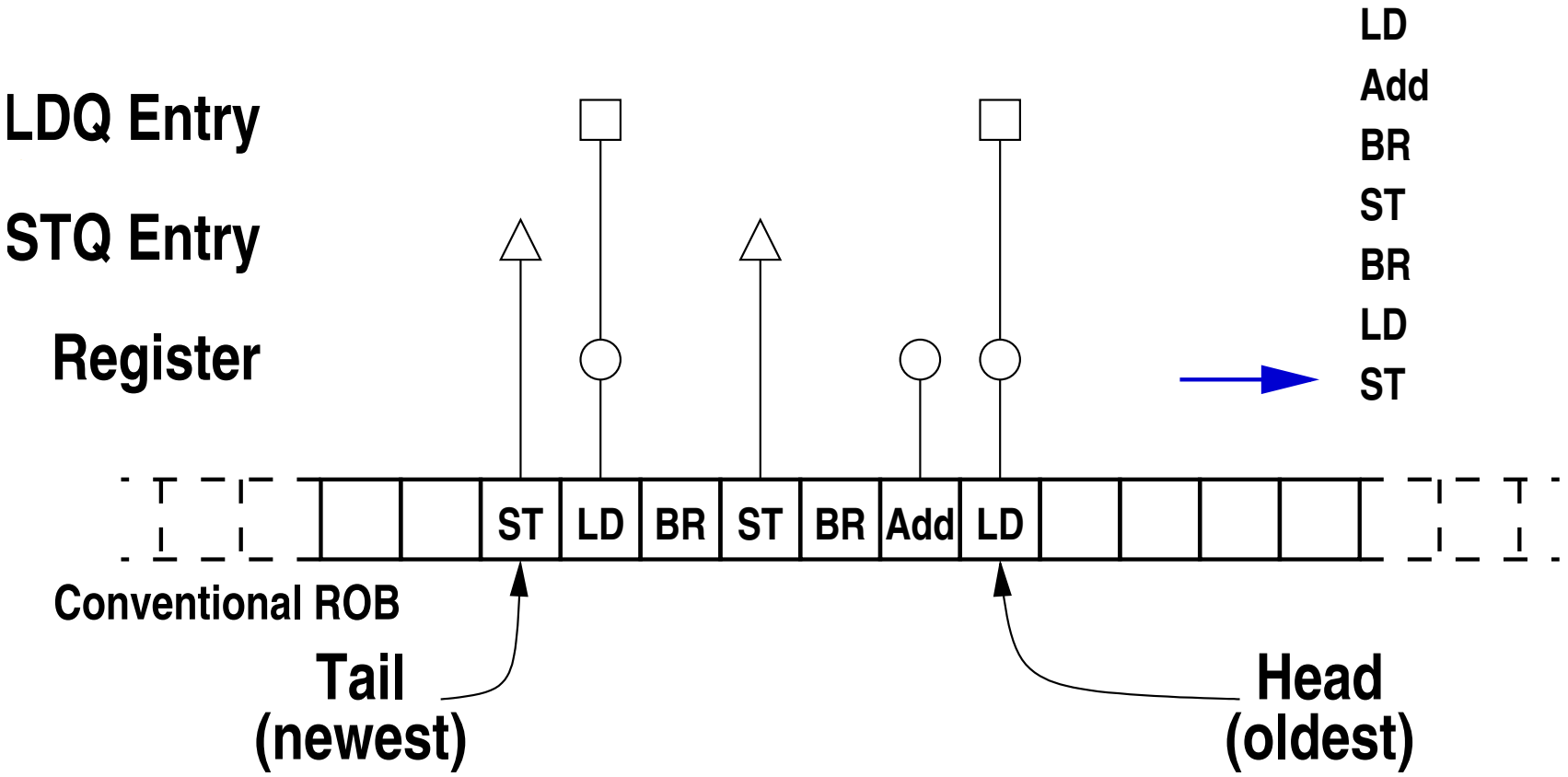
# CURRENT PROCESSORS



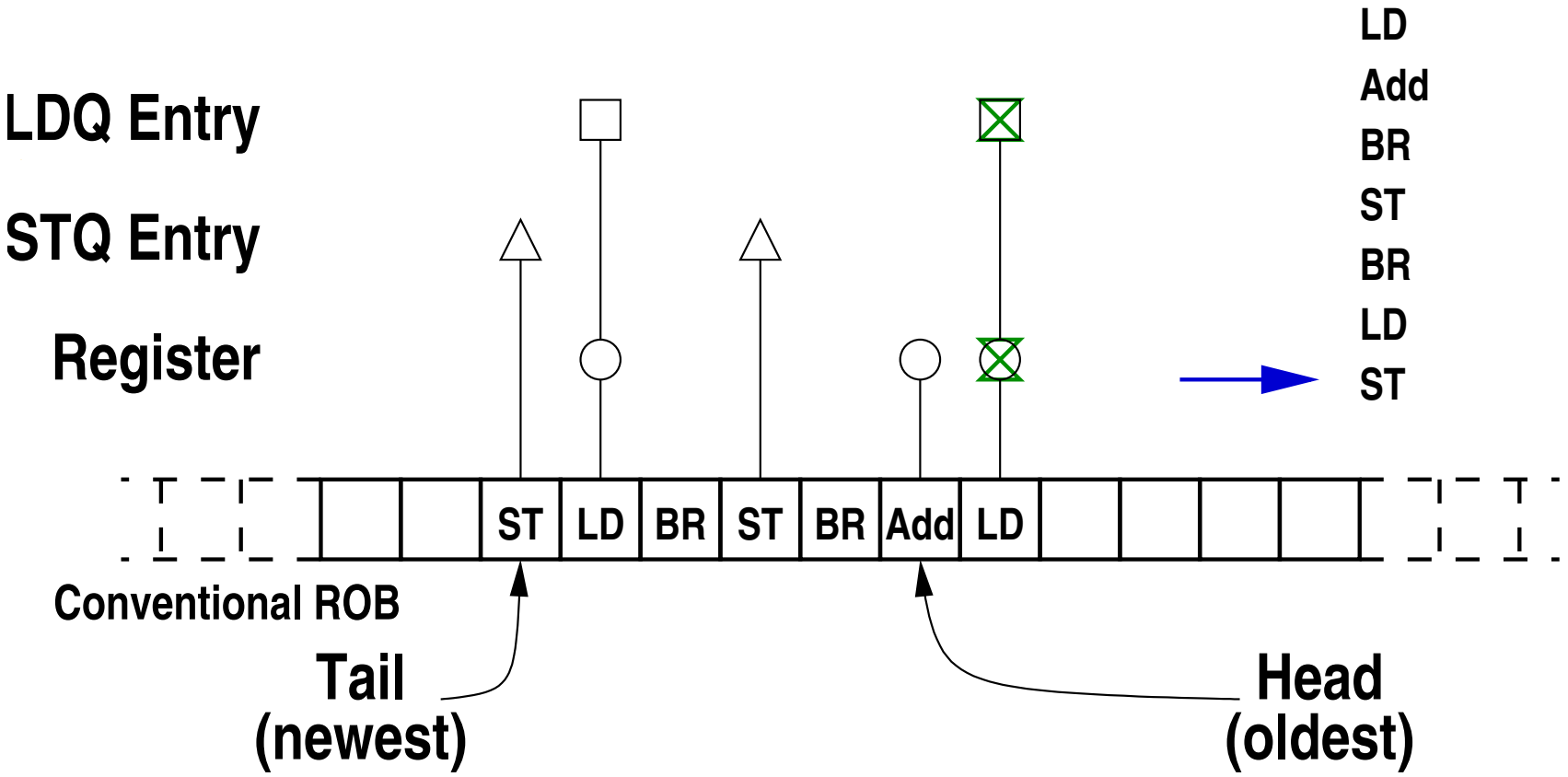
# CURRENT PROCESSORS



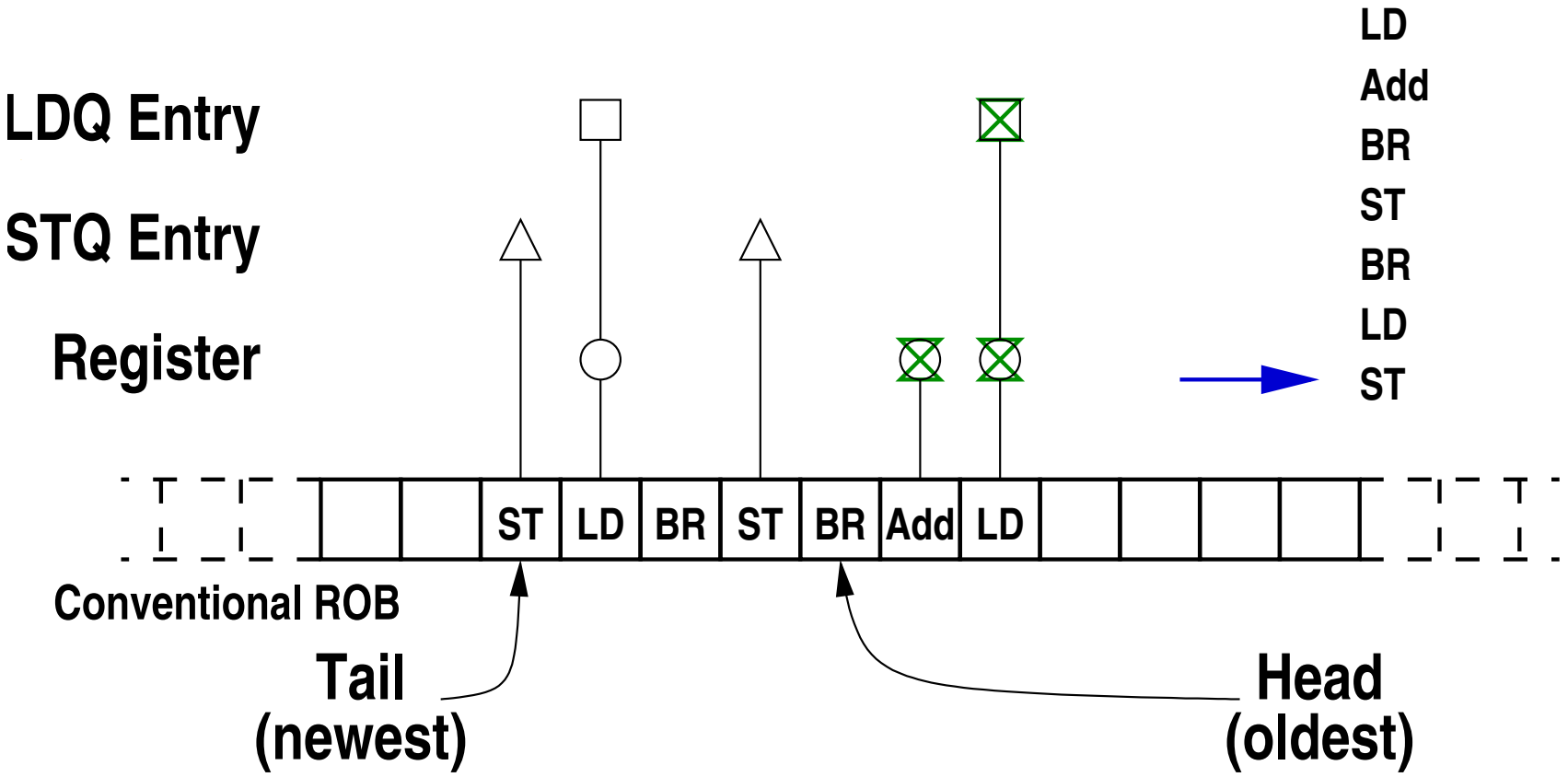
# CURRENT PROCESSORS



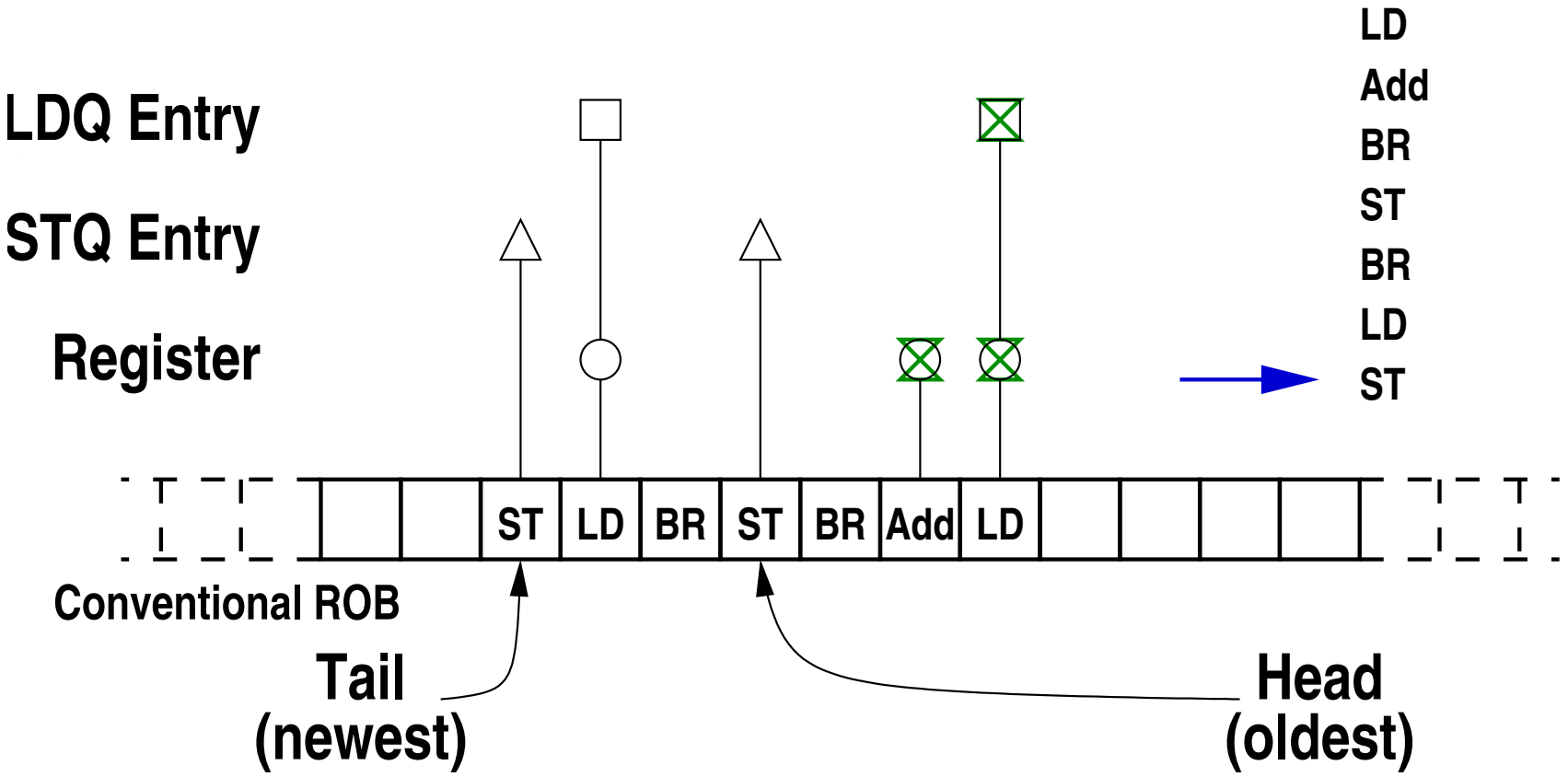
# CURRENT PROCESSORS



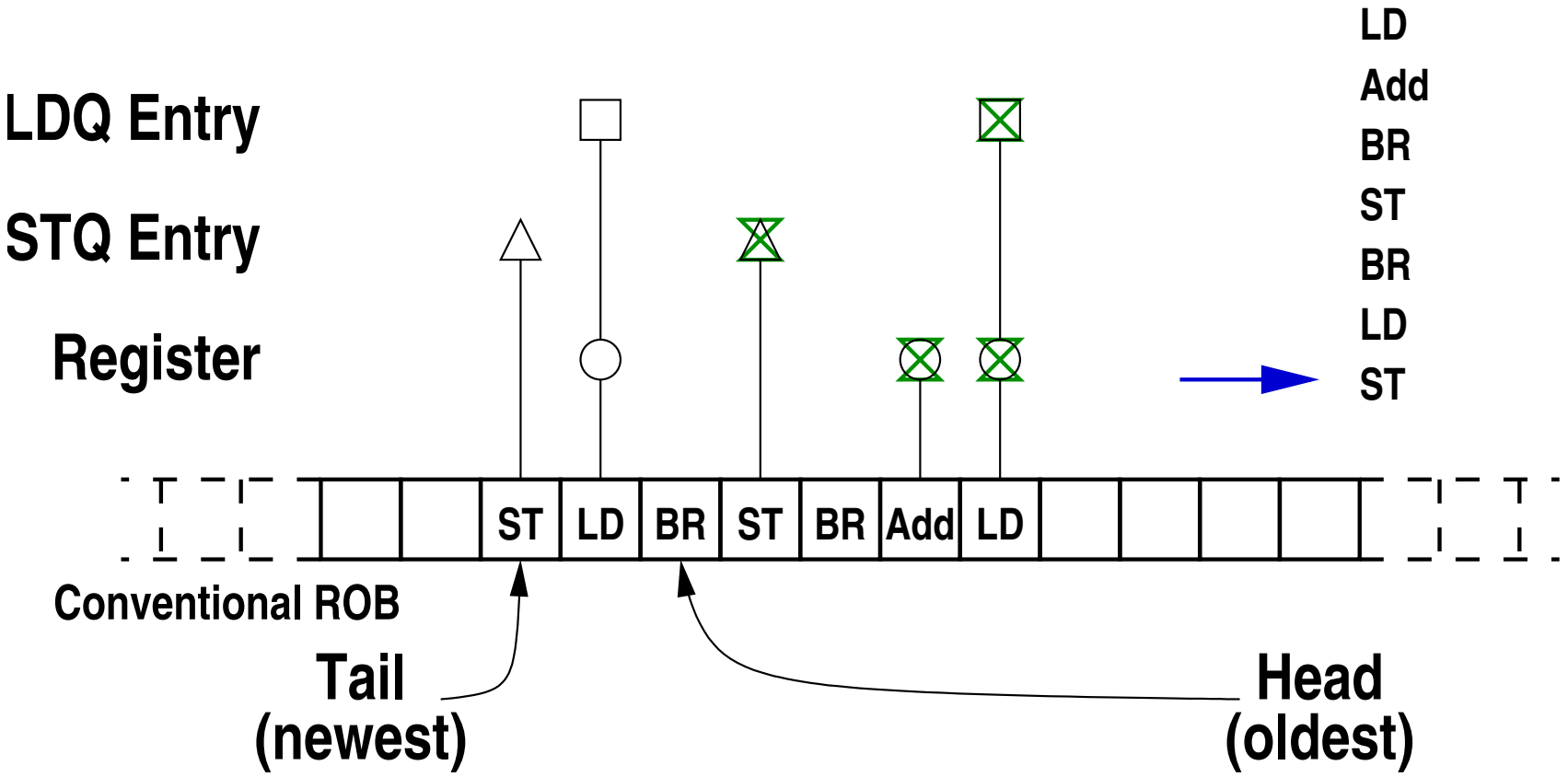
# CURRENT PROCESSORS



# CURRENT PROCESSORS



# CURRENT PROCESSORS

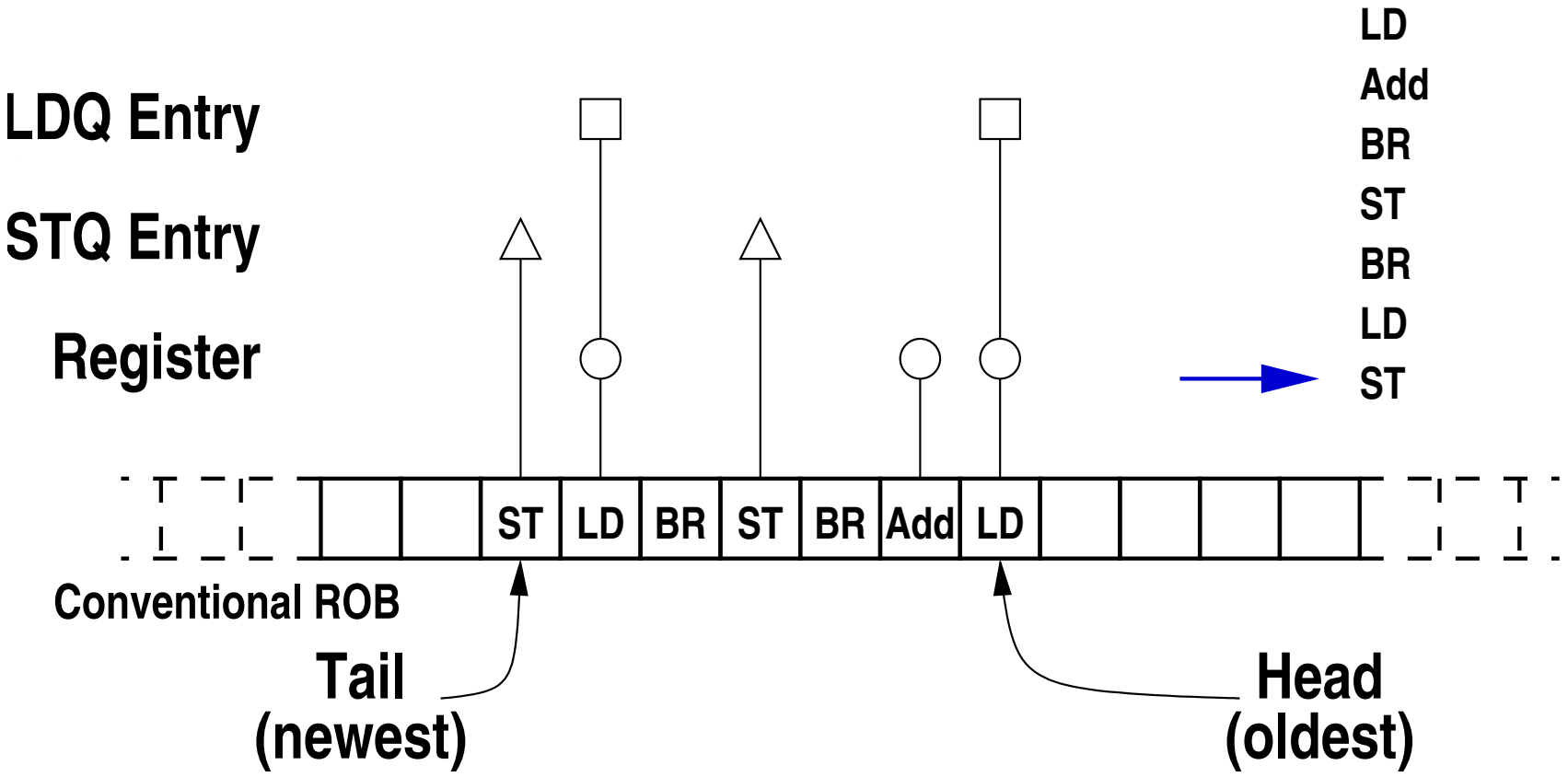


# PROPOSAL: EARLY RECYCLING

- Decouple resource recycling from instruction retirement
  - Recycle when no longer needed
- Targeted resources:
  - Load queue entries
  - Store queue entries
  - Physical registers
- Potential when targeted resources are unlimited:
  - 1.12 speedup in SPECint2000
  - 1.32 speedup in SPECfp2000

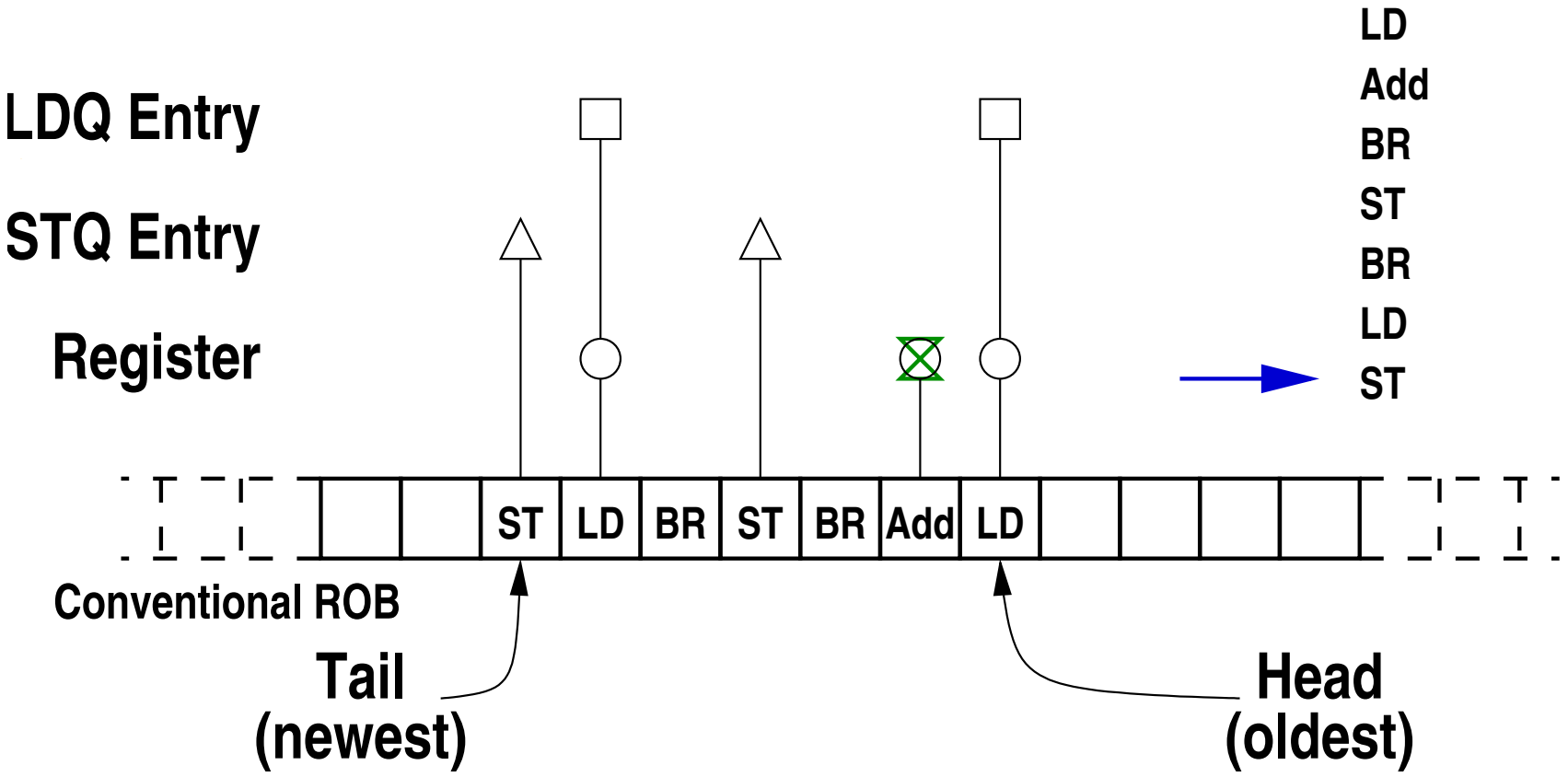
# EARLY RECYCLING

- Decouple resource recycling from retirement



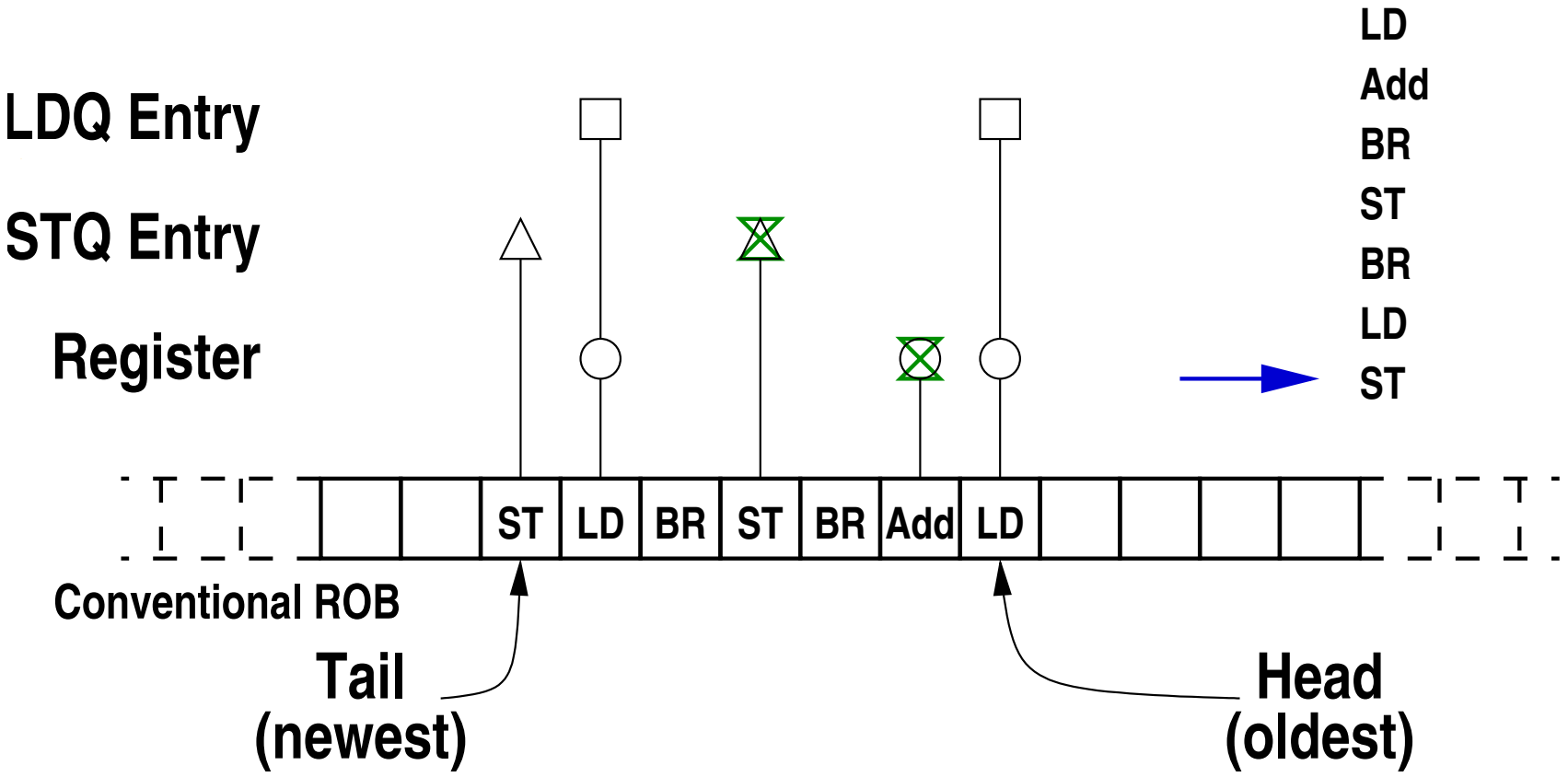
# EARLY RECYCLING

- Decouple resource recycling from retirement



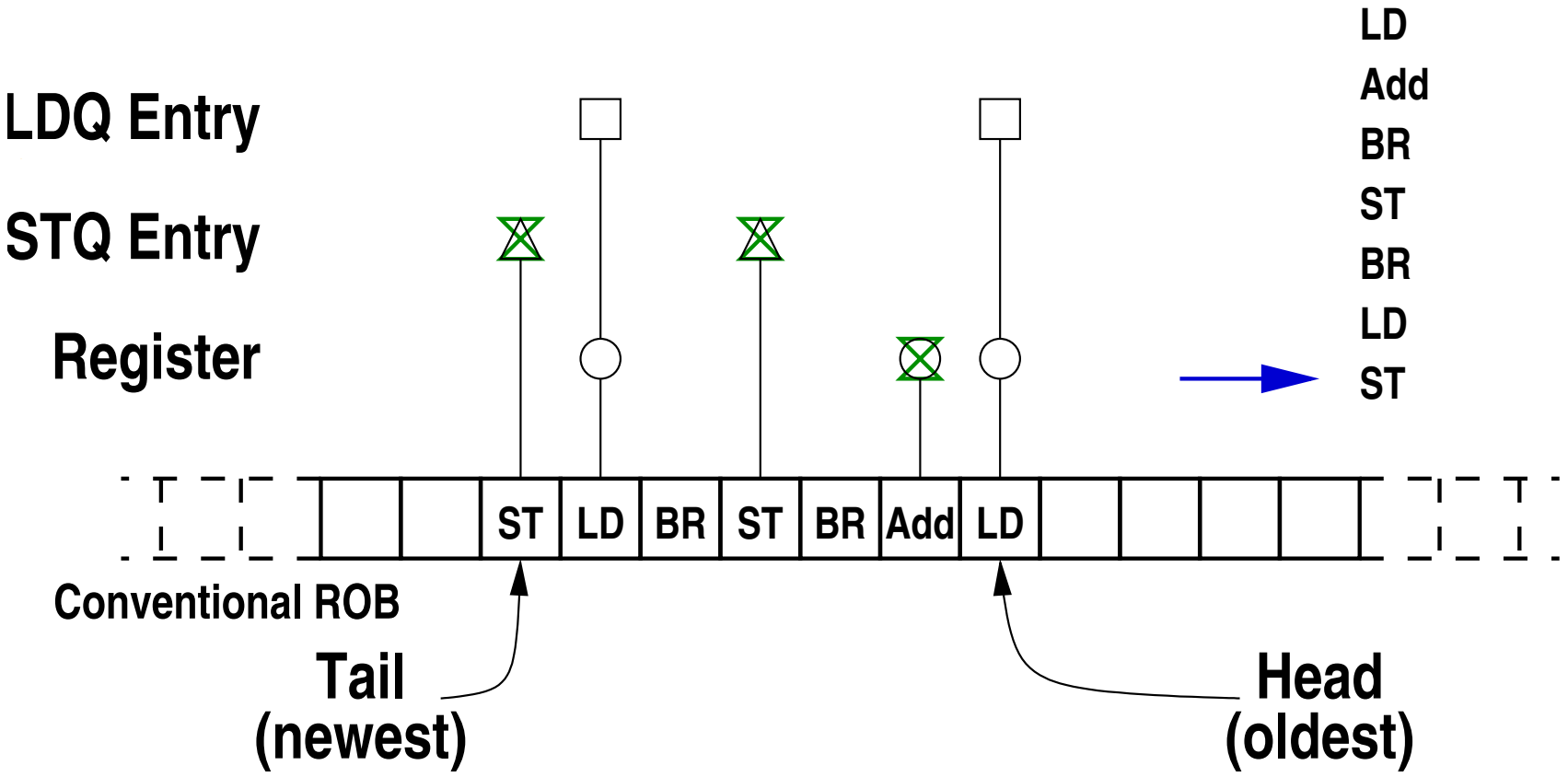
# EARLY RECYCLING

- Decouple resource recycling from retirement



# EARLY RECYCLING

- Decouple resource recycling from retirement



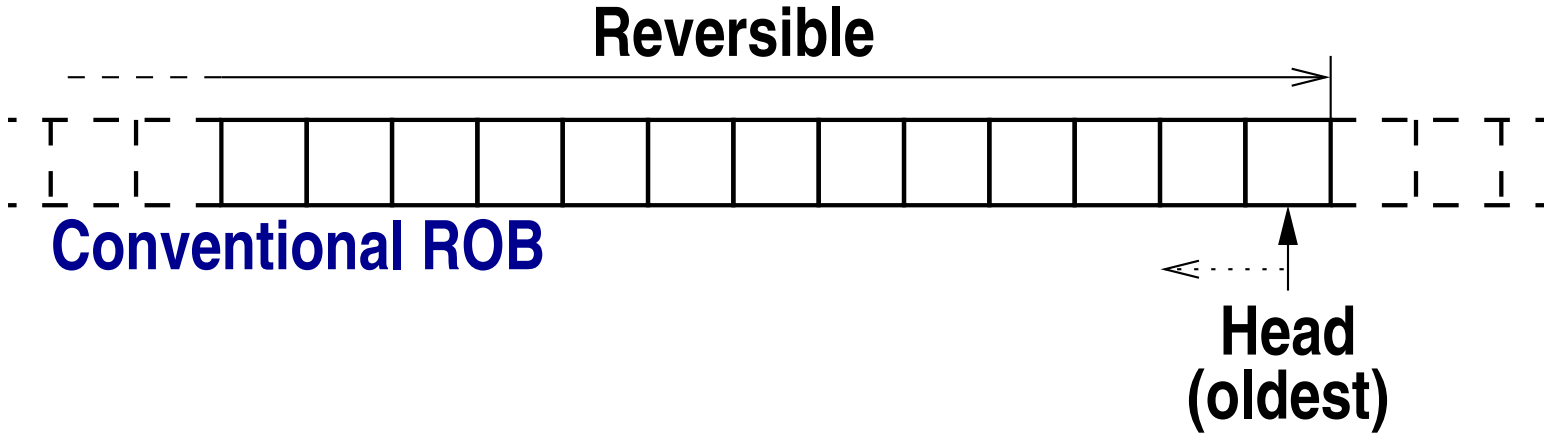
# CONSTRAINT: PRECISE STATE

- Events that require precise state:
  - Branch mispredictions
  - Memory replay traps
  - Exceptions
  - Interrupts
- Early recycling makes this difficult
  - E.g. recycled registers no longer available
  - E.g. memory may be overwriten prematurely

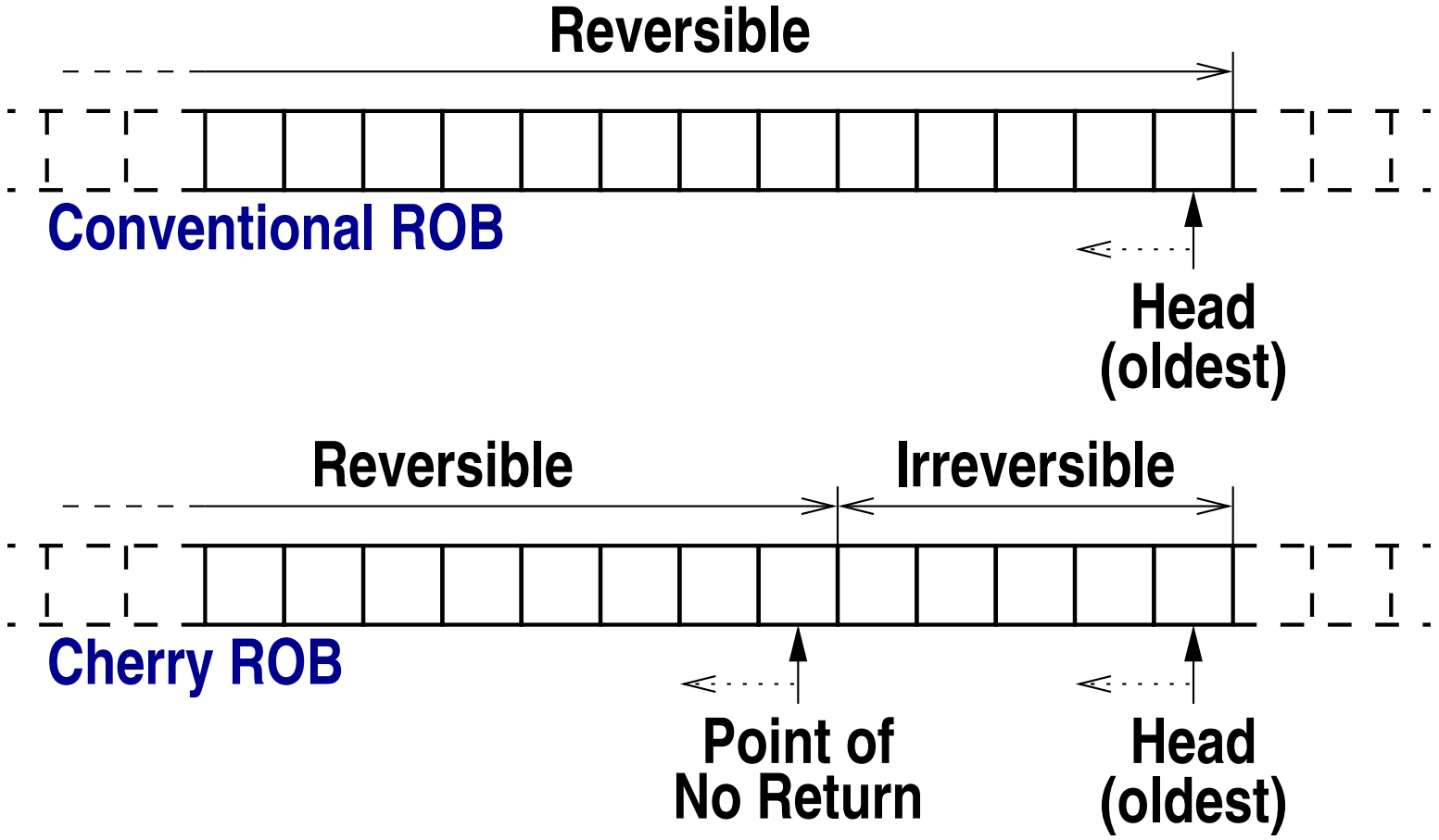
# PROPOSAL: POINT OF NO RETURN

- Classify events according to frequency:
  - Frequent: branch mispredictions, memory replays
  - Infrequent: exceptions
- Split ROB into two logical sets of instructions
  - **Irreversible set**
    - Subject to infrequent events only
    - Checkpoint recovery
  - **Reversible set**:
    - Subject to frequent and infrequent events
    - Conventional ROB recovery mechanisms
  - Boundary: **Point of No Return** (PNR)

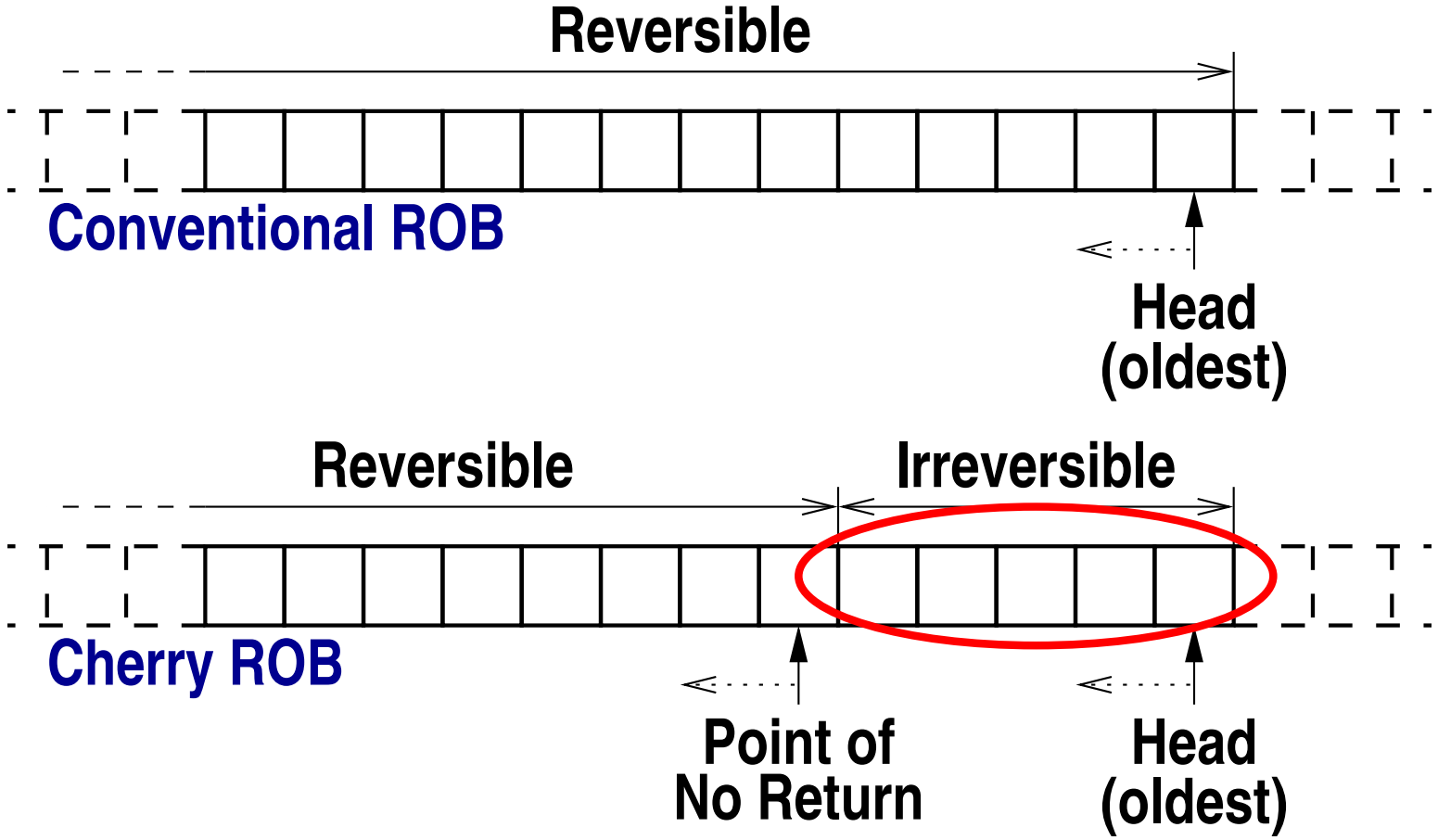
# CONVENTIONAL VS CHERRY ROB



# CONVENTIONAL VS CHERRY ROB



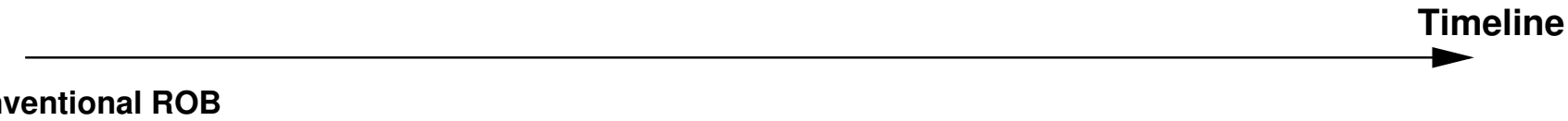
# CONVENTIONAL VS CHERRY ROB



# OUTLINE

- Motivation
- Overview
- Implementation
  - Checkpointing
  - Exception and interrupt handling
  - Early resource recycling
- Results

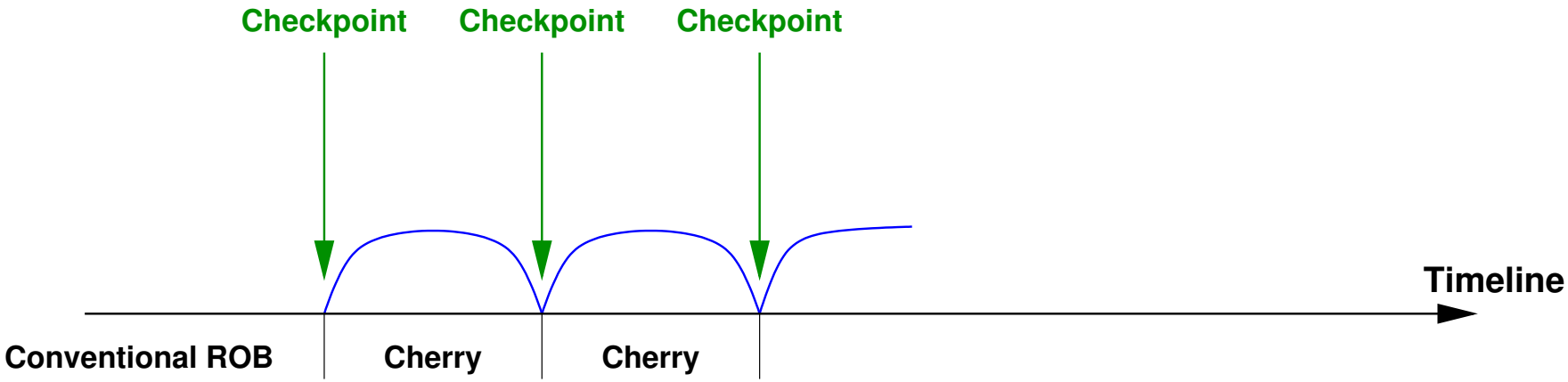
# TIMELINE



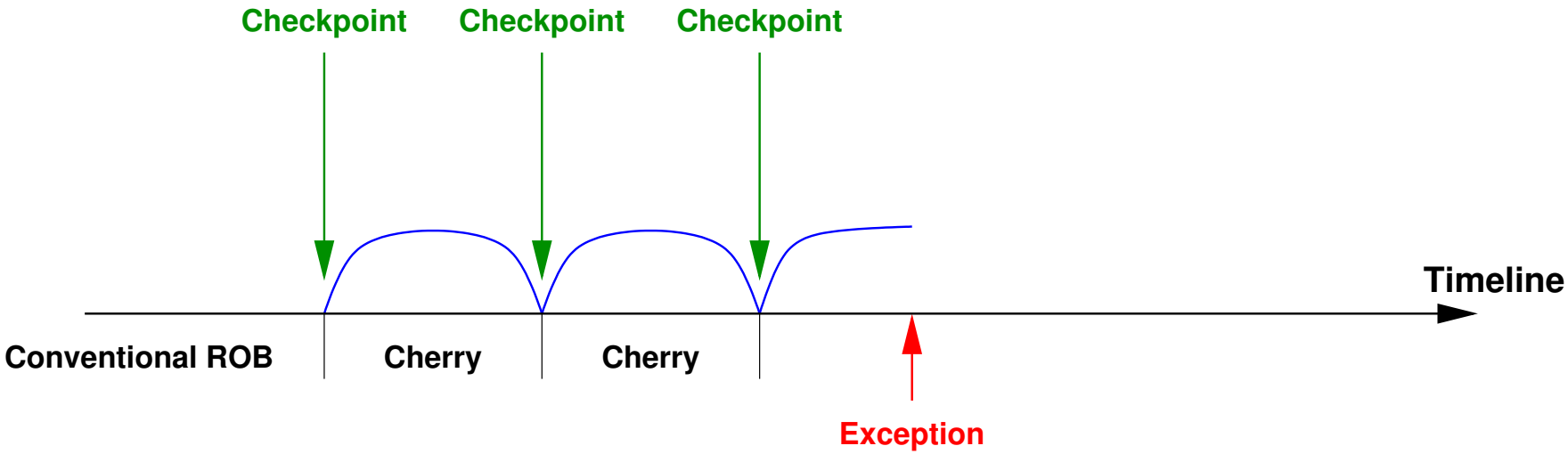

**Conventional ROB**

# TIMELINE

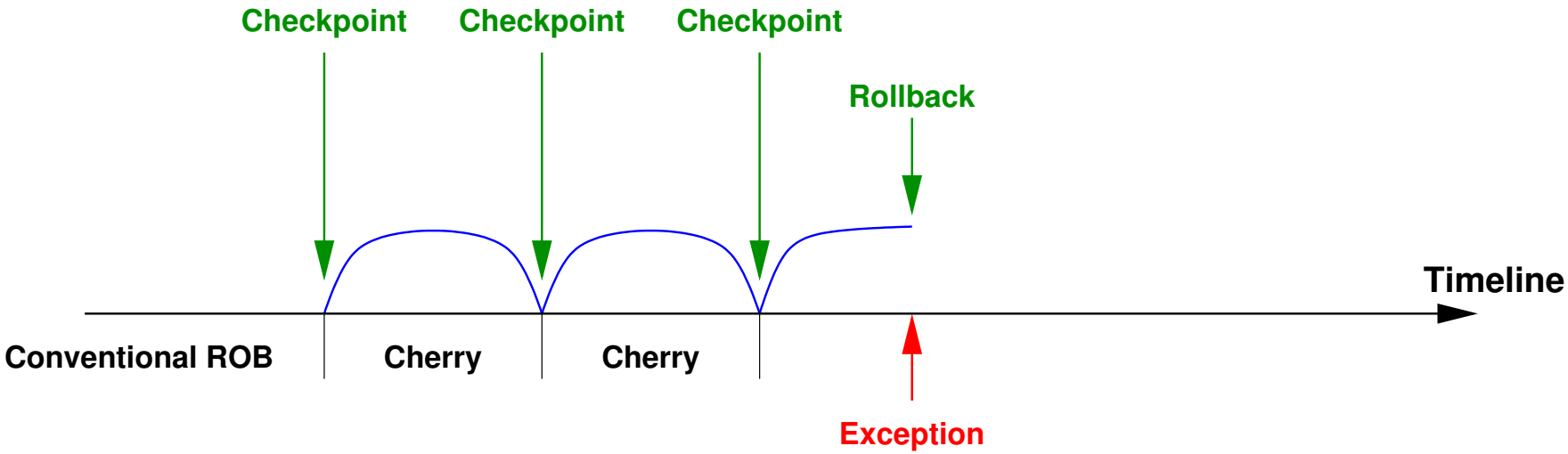
# TIMELINE


Checkpoint
Checkpoint
Checkpoint
Timeline
Conventional ROB
Cherry
Cherry

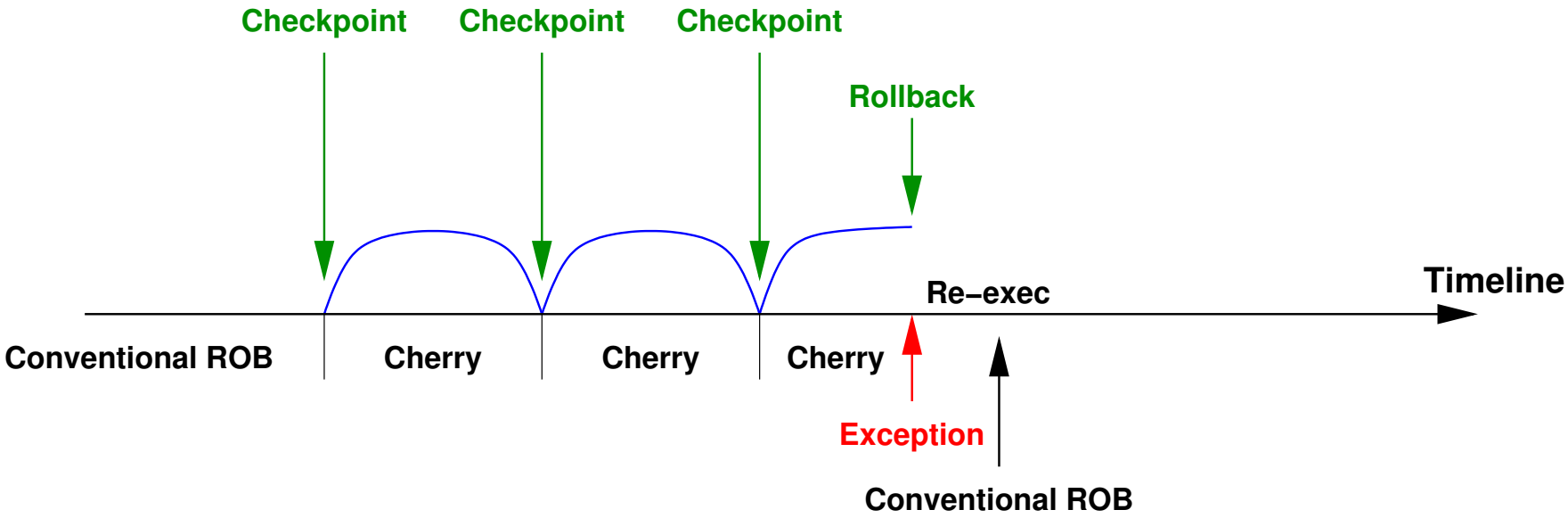
# TIMELINE


Checkpoint
Checkpoint
Checkpoint
Timeline
Conventional ROB
Cherry
Cherry
Exception

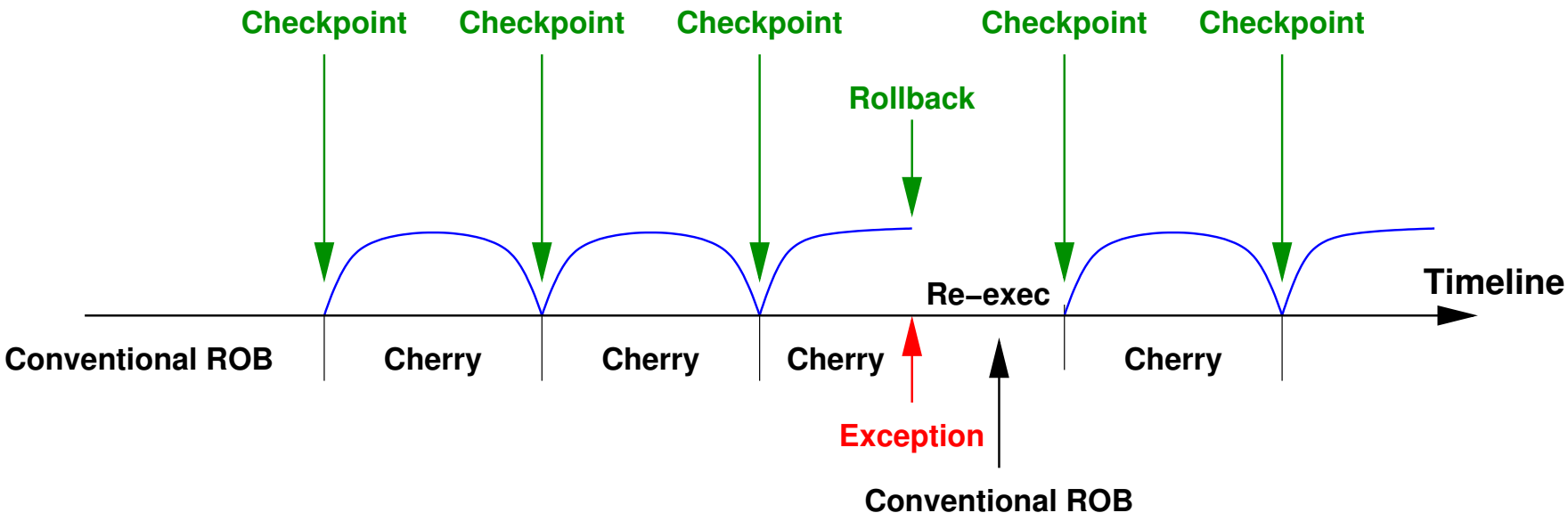
# TIMELINE


Checkpoint
Checkpoint
Checkpoint
Rollback
Timeline
Conventional ROB
Cherry
Cherry
Exception

# TIMELINE

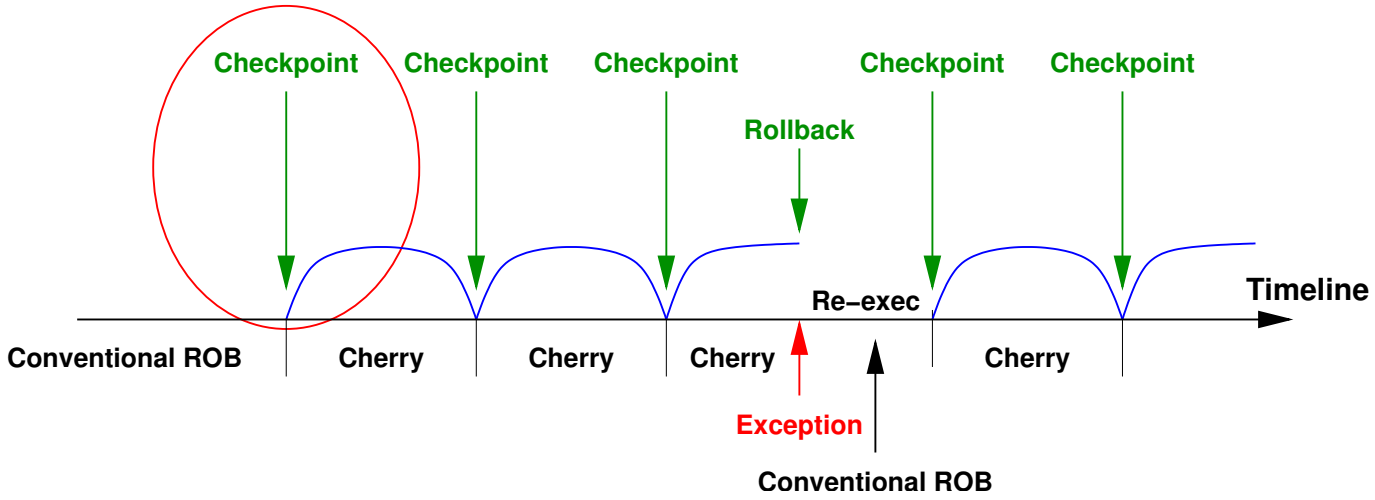

Checkpoint
Checkpoint
Checkpoint
Rollback
Re−exec
Timeline
Conventional ROB
Cherry
Cherry
Cherry
Exception
Conventional ROB

# TIMELINE


Checkpoint
Checkpoint
Checkpoint
Checkpoint
Checkpoint
Rollback
Re−exec
Timeline
Conventional ROB
Cherry
Cherry
Cherry
Cherry
Exception
Conventional ROB

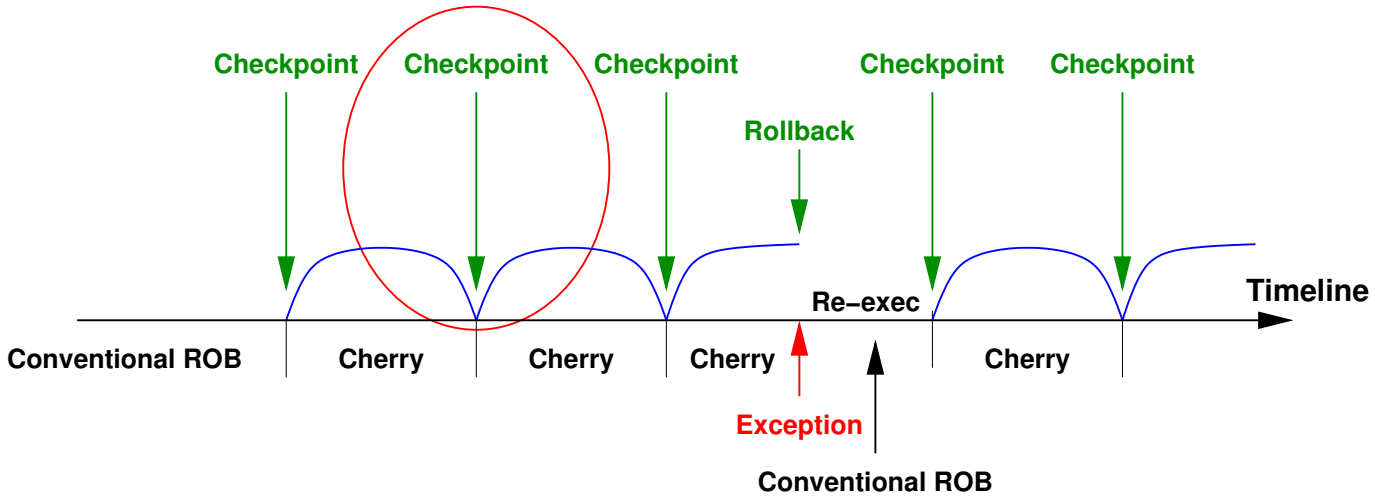
# Entering Cherry Mode

- Backup architectural registers
  - Can be done in a handful of cycles
- Allow PNR to race ahead of ROB head
- Updates to cache hierarchy set Volatile bit



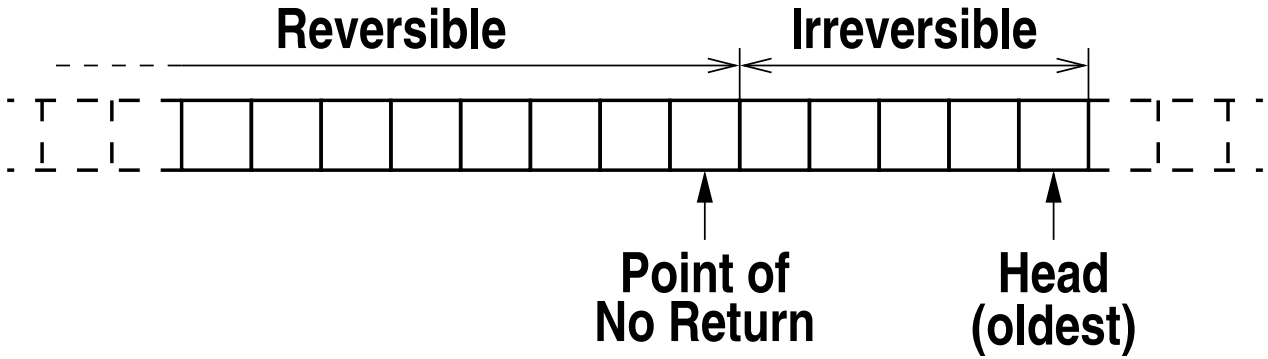
# PERIODIC CHECKPOINTING

- Done regularly to bound re-execution overhead
- Stop early recycling to create checkpoint (collapse)
  - Freeze PNR (stop recycling)
- Once ROB head catches up with PNR:
  - Clear Volatile bits from cache
  - Backup architectural registers



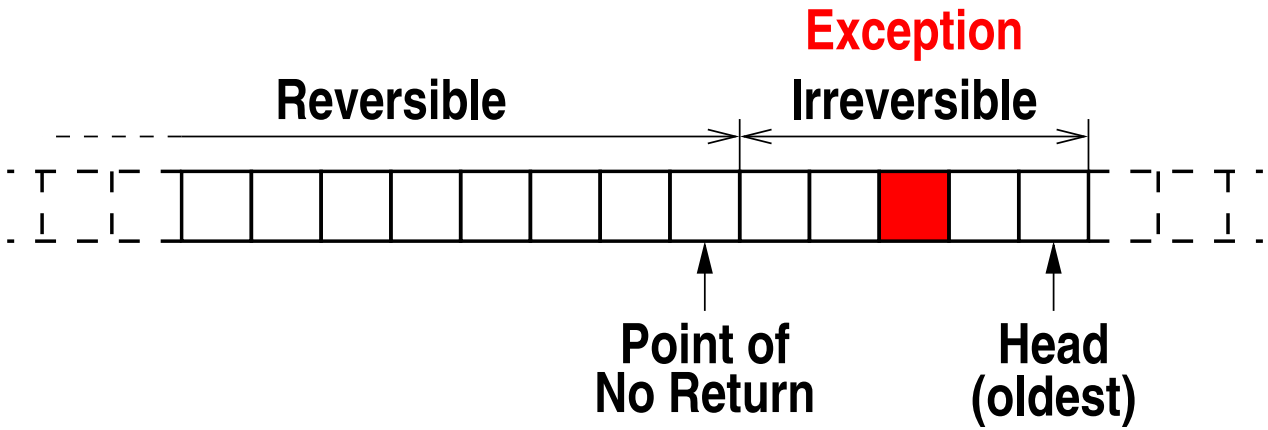
# PRECISE EXCEPTION HANDLING

- Exception in Irreversible set:
  - Roll back to checkpoint
  - Temporarily disable recycling (for some time)
  - Re-execute in conventional OOO mode (w/o Cherry)
  - Allow exception to re-occur



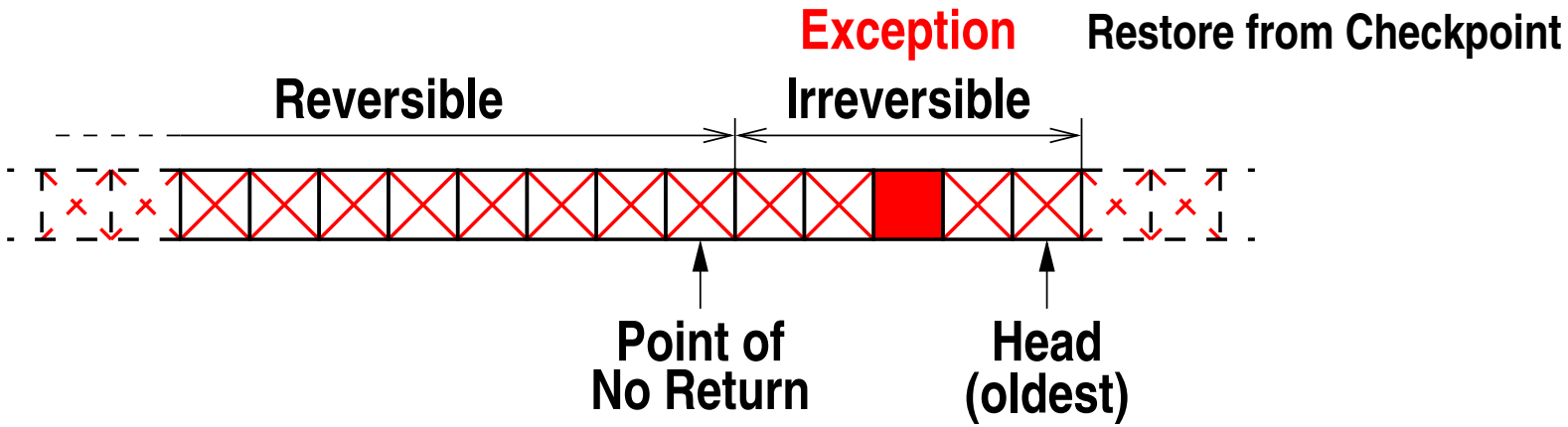
# Precise Exception Handling

- Exception in Irreversible set:
  - Roll back to checkpoint
  - Temporarily disable recycling (for some time)
  - Re-execute in conventional OOO mode (w/o Cherry)
  - Allow exception to re-occur



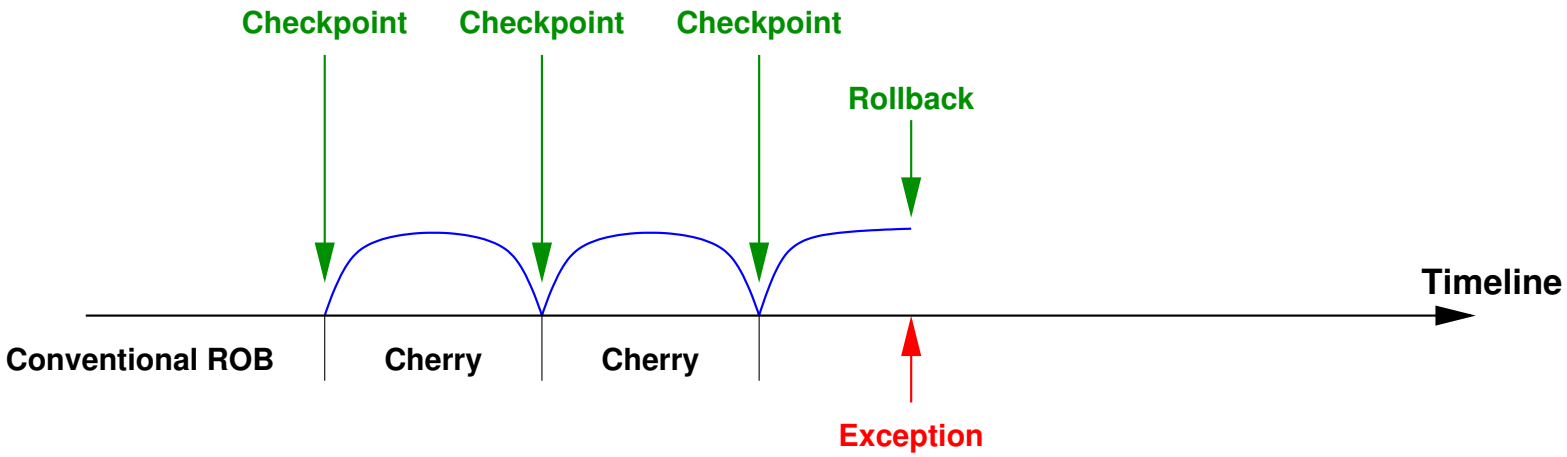
# Precise Exception Handling

- Exception in Irreversible set:
  - Roll back to checkpoint
  - Temporarily disable recycling (for some time)
  - Re-execute in conventional OOO mode (w/o Cherry)
  - Allow exception to re-occur



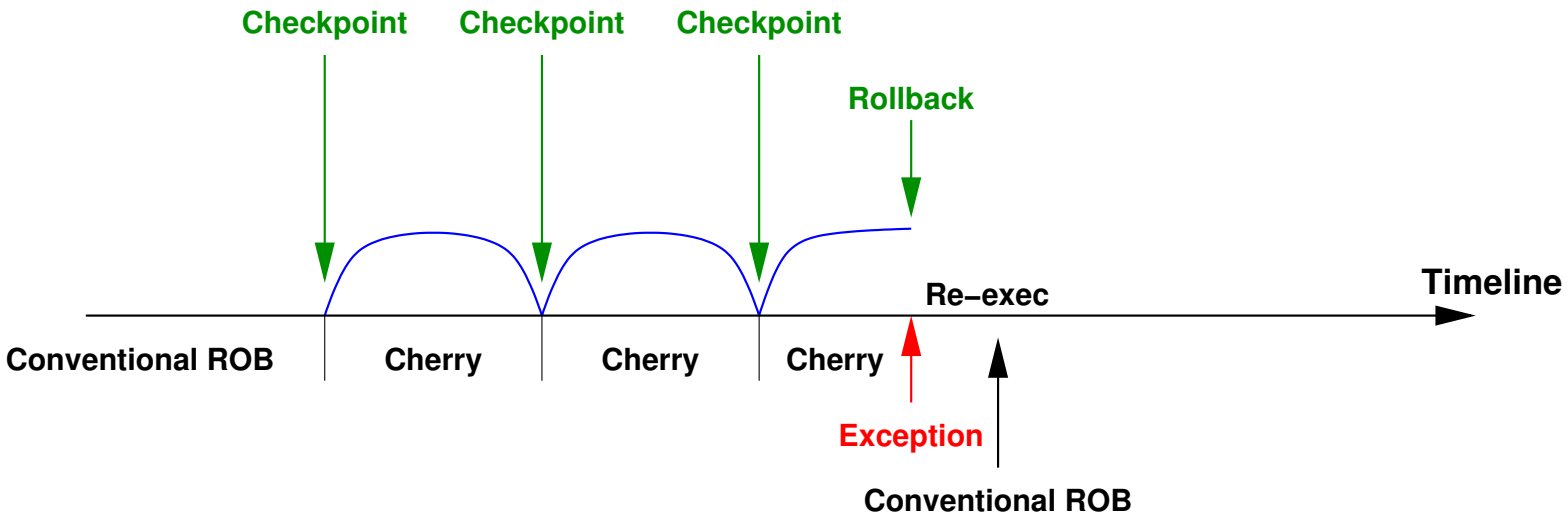
# Precise Exception Handling

- Exception in Irreversible set:
  - Roll back to checkpoint
  - Temporarily disable recycling (for some time)
  - Re-execute in conventional OOO mode (w/o Cherry)
  - Allow exception to re-occur



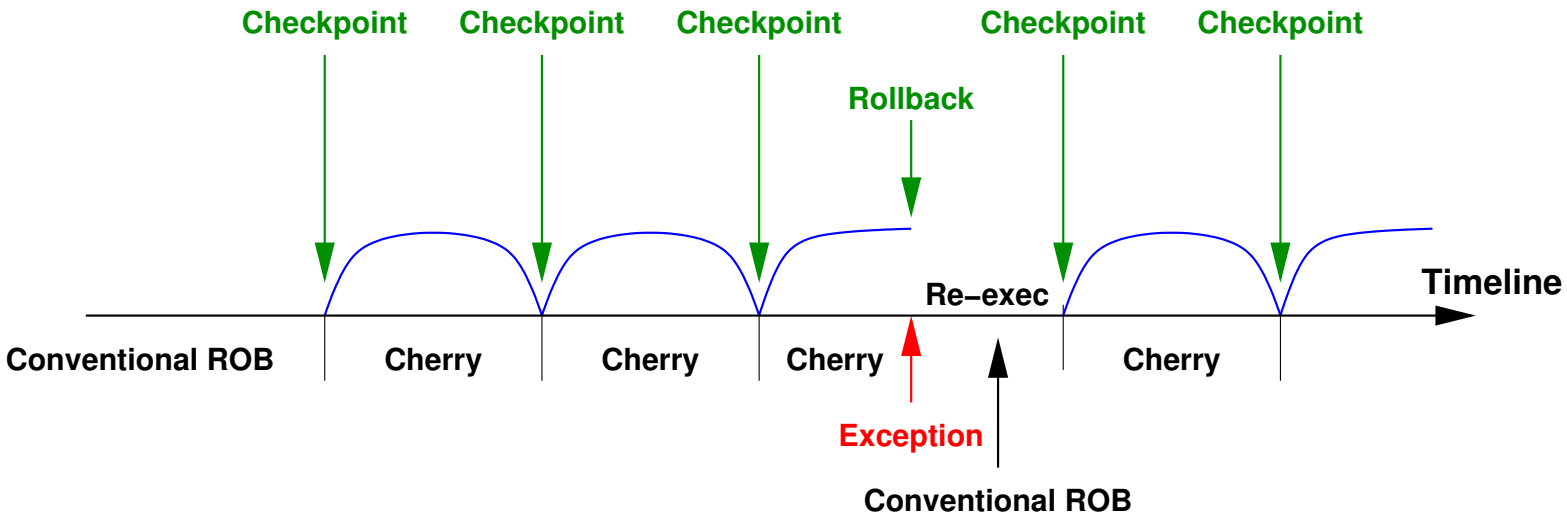
# Precise Exception Handling

- Exception in Irreversible set:
  - Roll back to checkpoint
  - Temporarily disable recycling (for some time)
  - Re-execute in conventional OOO mode (w/o Cherry)
  - Allow exception to re-occur



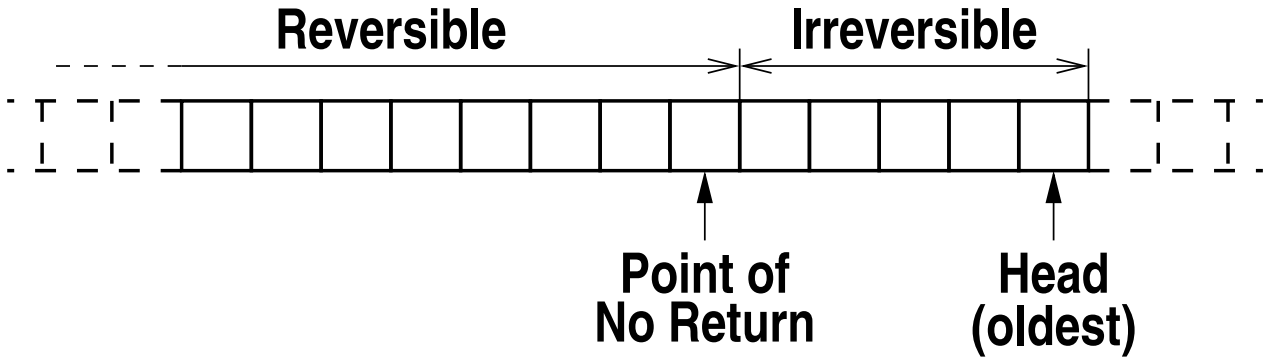
# Precise Exception Handling

- Exception in Irreversible set:
  - Roll back to checkpoint
  - Temporarily disable recycling (for some time)
  - Re-execute in conventional OOO mode (w/o Cherry)
  - Allow exception to re-occur



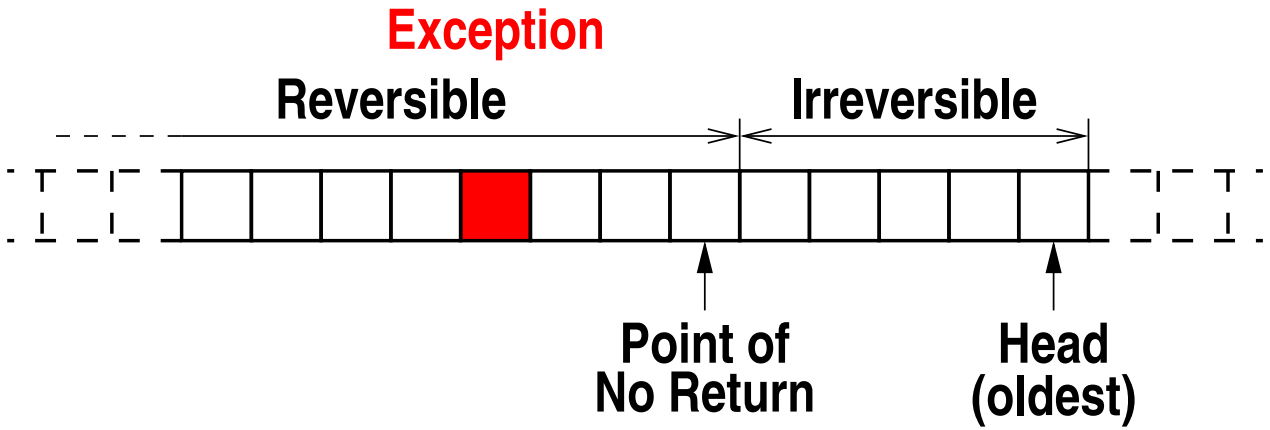
# PRECISE EXCEPTION HANDLING

- Exception in Reversible set:
  - Disable resource recycling (freeze PNR)
  - Process exception
  - Take new checkpoint; re-enter Cherry mode



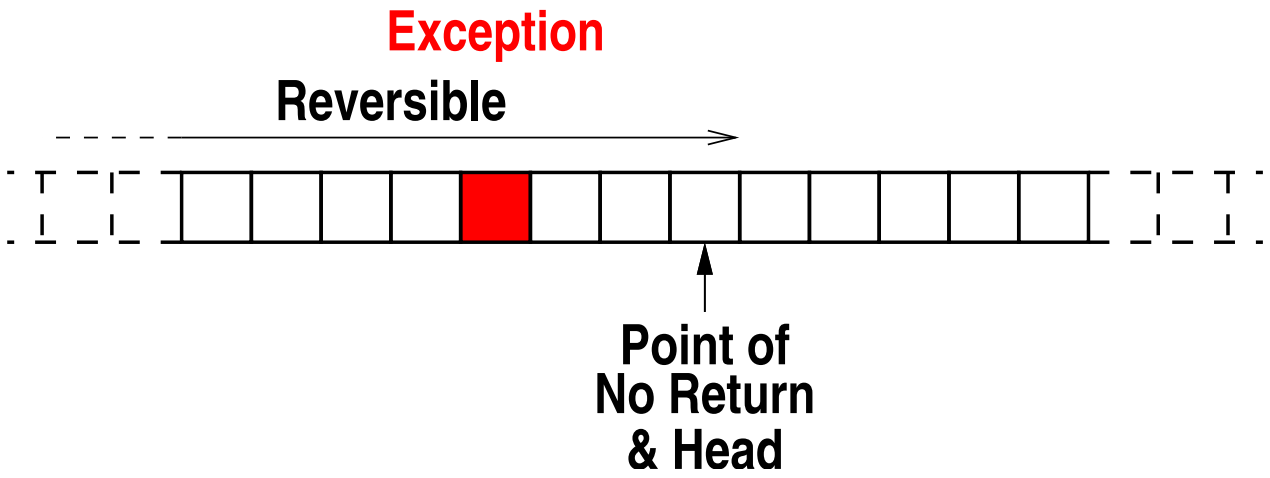
# PRECISE EXCEPTION HANDLING

- Exception in Reversible set:
  - Disable resource recycling (freeze PNR)
  - Process exception
  - Take new checkpoint; re-enter Cherry mode



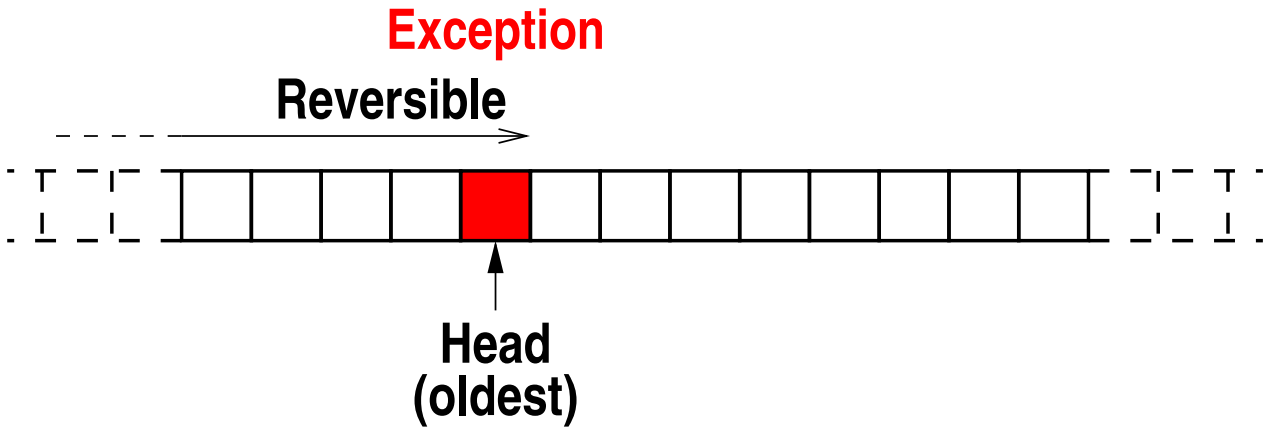
# PRECISE EXCEPTION HANDLING

- Exception in Reversible set:
  - Disable resource recycling (freeze PNR)
  - Process exception
  - Take new checkpoint; re-enter Cherry mode



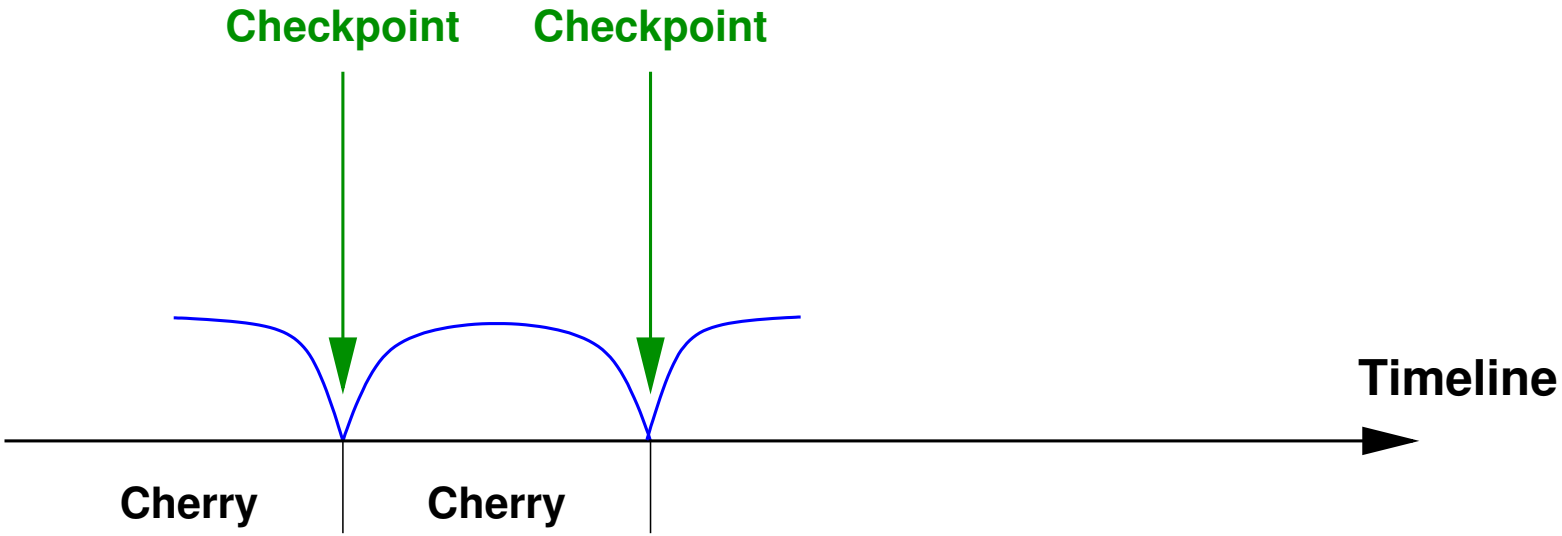
# PRECISE EXCEPTION HANDLING

- Exception in Reversible set:
  - Disable resource recycling (freeze PNR)
  - Process exception
  - Take new checkpoint; re-enter Cherry mode



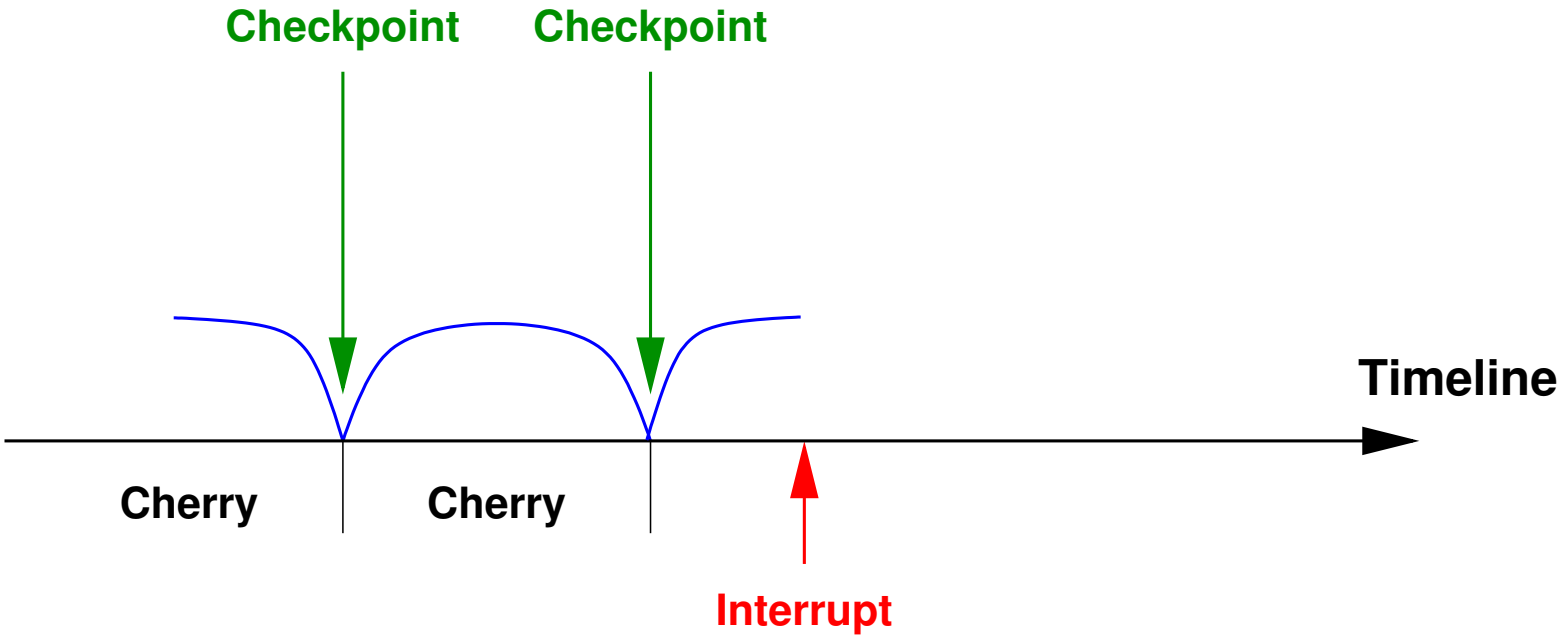
# INTERRUPT HANDLING

- Interrupts are asynchronous: delay handling
  - Disable resource recycling (freeze PNR)
  - Process interrupt
  - Take new checkpoint and re-enter Cherry mode



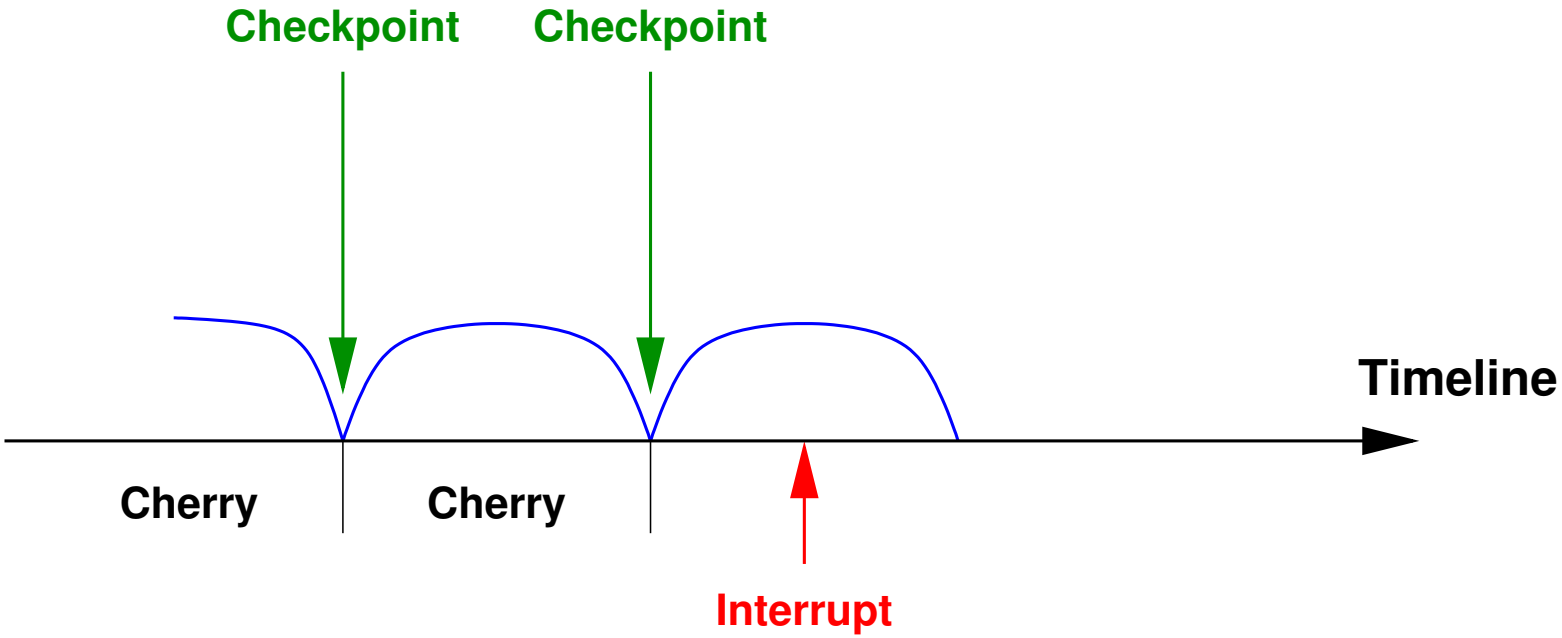
# INTERRUPT HANDLING

- Interrupts are asynchronous: delay handling
  - Disable resource recycling (freeze PNR)
  - Process interrupt
  - Take new checkpoint and re-enter Cherry mode



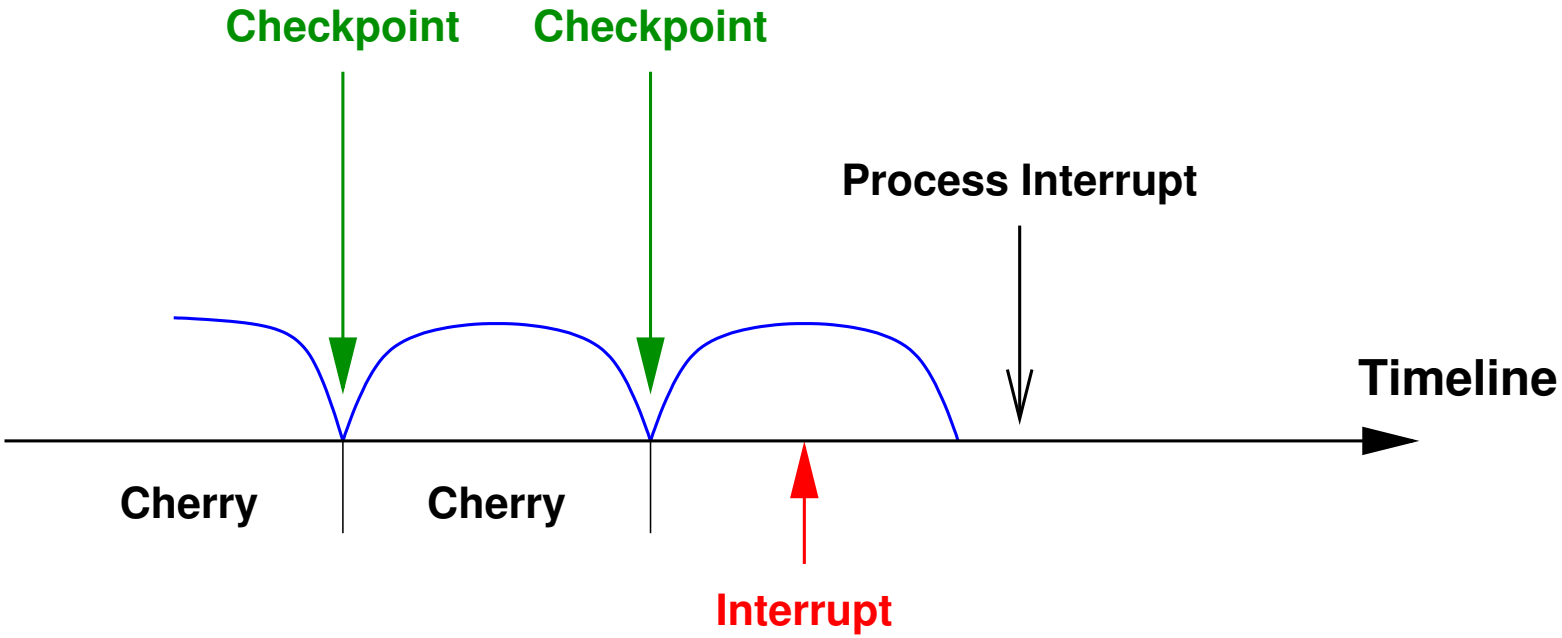
# INTERRUPT HANDLING

- Interrupts are asynchronous: delay handling
  - Disable resource recycling (freeze PNR)
  - Process interrupt
  - Take new checkpoint and re-enter Cherry mode



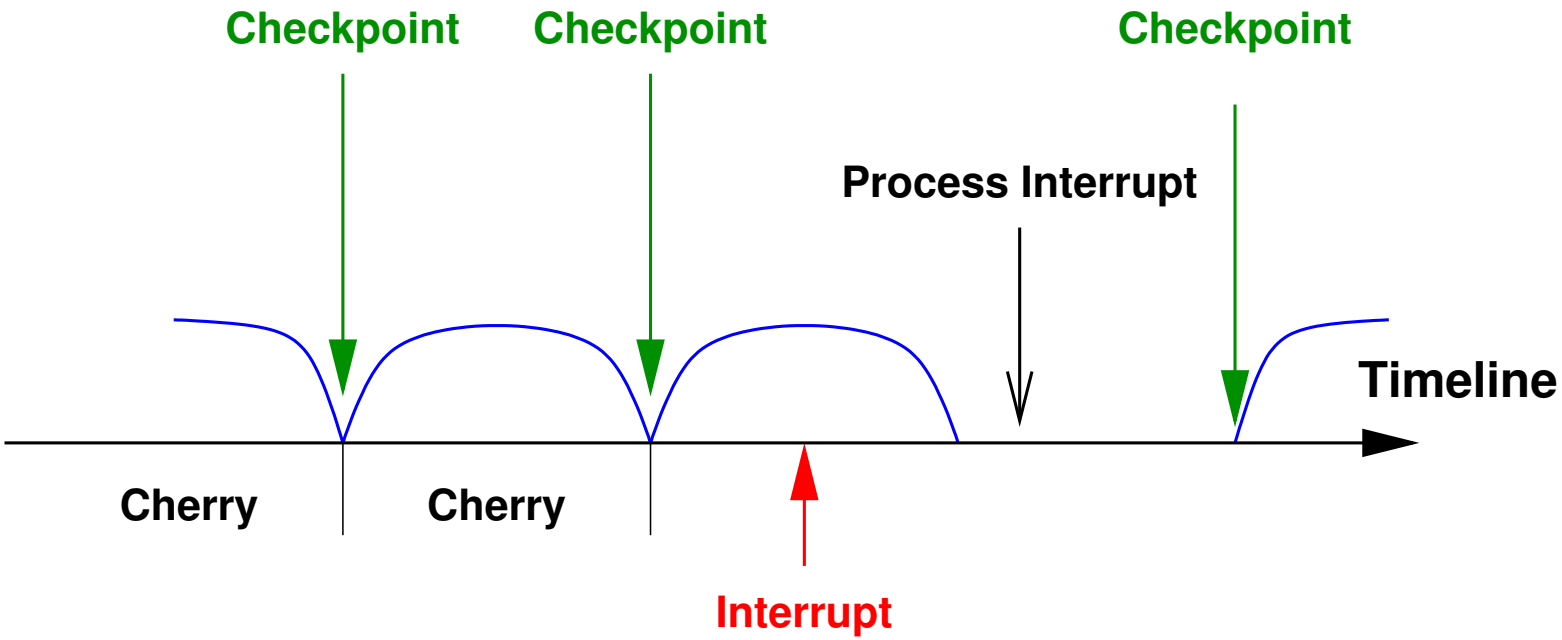
# INTERRUPT HANDLING

- Interrupts are asynchronous: delay handling
  - Disable resource recycling (freeze PNR)
  - Process interrupt
  - Take new checkpoint and re-enter Cherry mode



# INTERRUPT HANDLING

- Interrupts are asynchronous: delay handling
  - Disable resource recycling (freeze PNR)
  - Process interrupt
  - Take new checkpoint and re-enter Cherry mode



# OUTLINE

- Motivation
- Overview
- Implementation
  - Checkpointing
  - Exception and interrupt handling
  - Early resource recycling
- Results

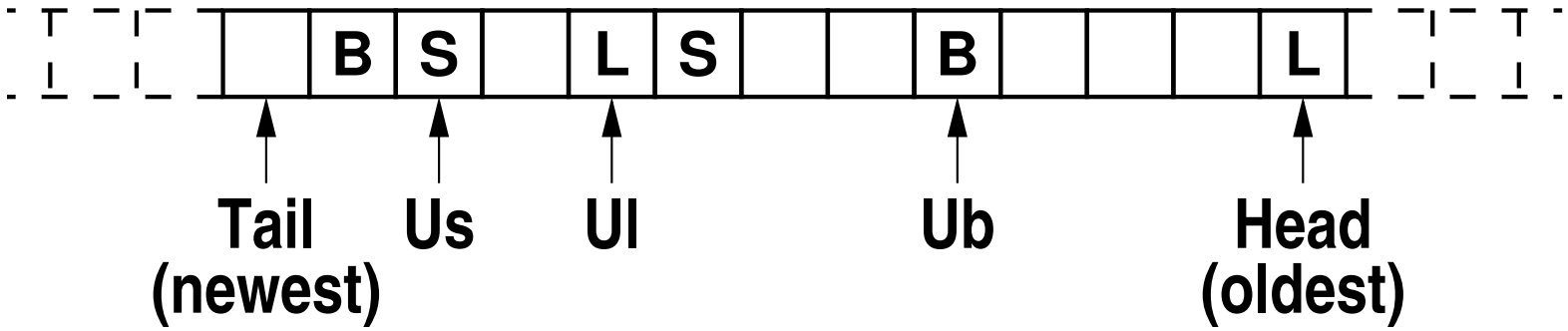
# DEFINING THE PNRS

- Each resource defines its own PNR
- Based on three ROB pointers:

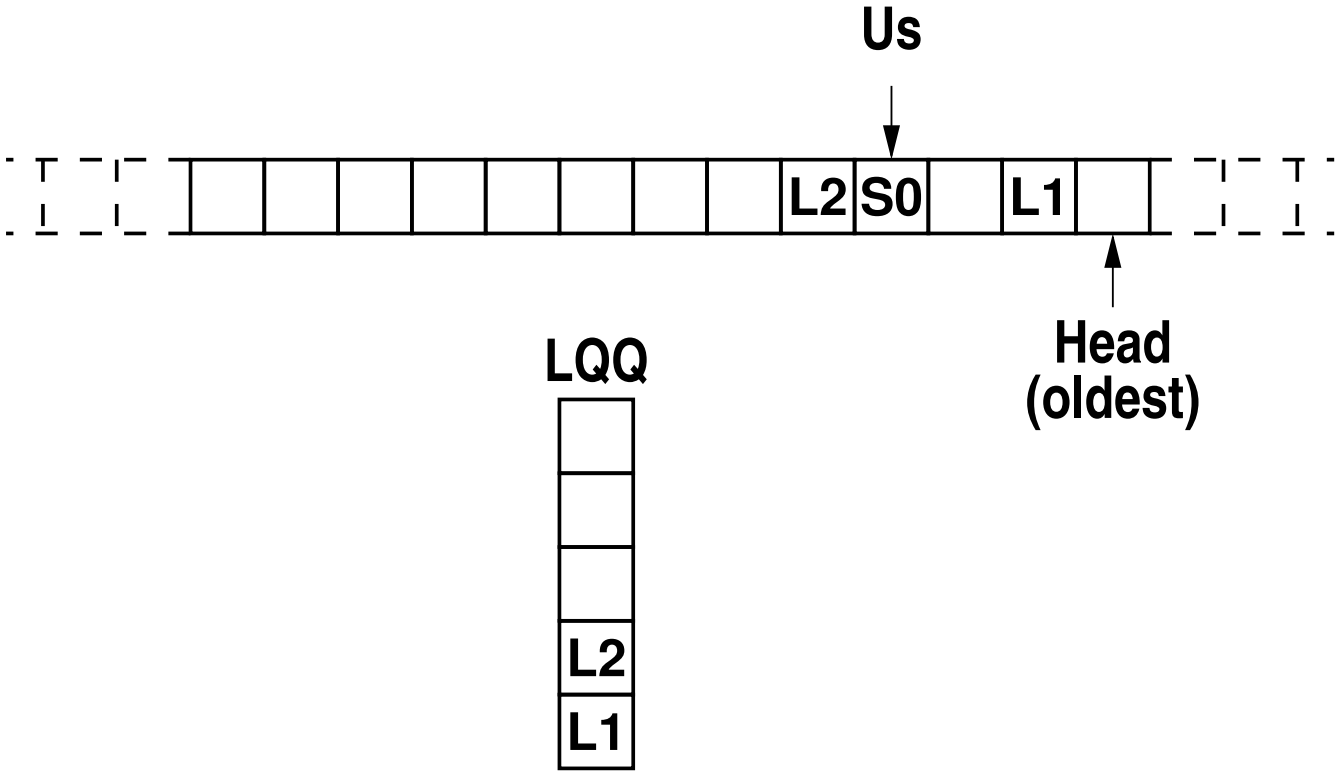
  $U_b$: oldest unresolved branch

  $U_s$: oldest store with unresolved address

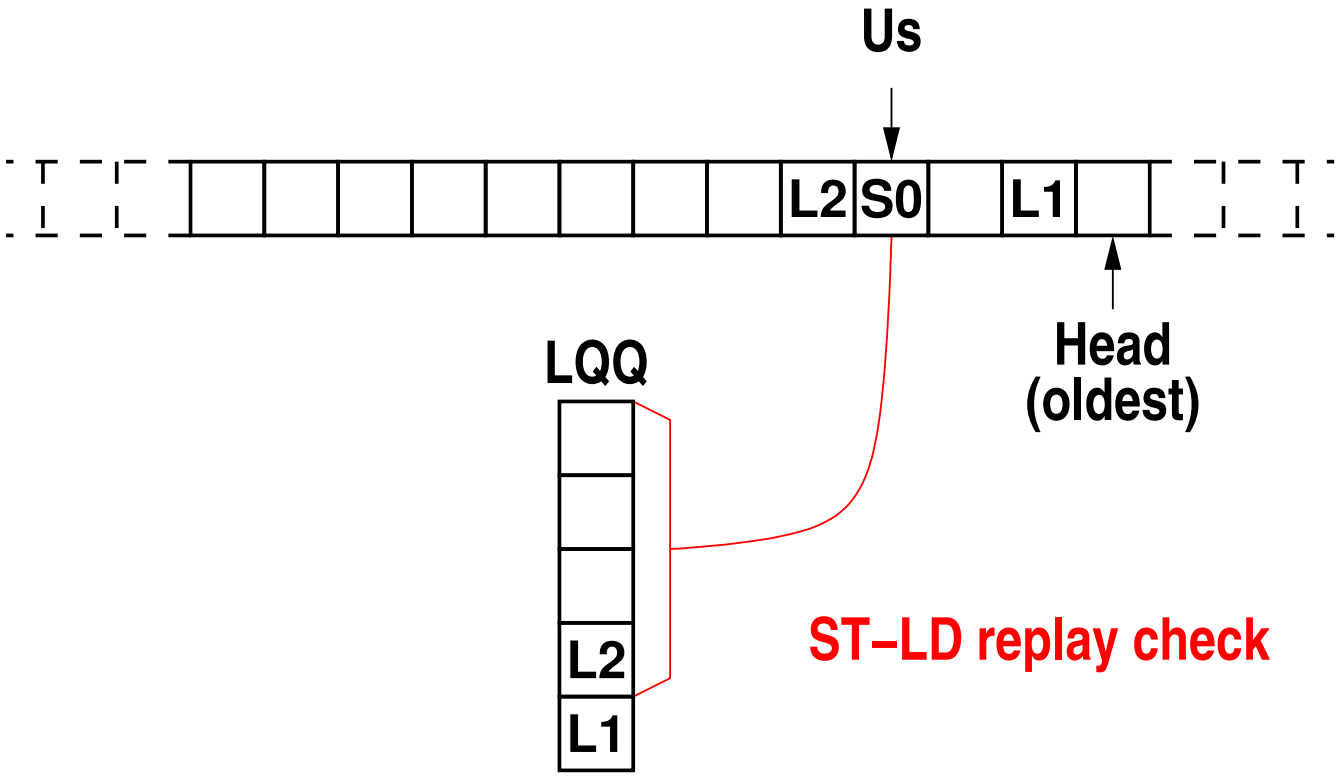
  $U_l$: oldest load with unresolved address



# EARLY LDQ ENTRY RECYCLING



# EARLY LDQ ENTRY RECYCLING



# EARLY LDQ ENTRY RECYCLING

- Constraint: Must detect memory replay traps
  - older($U_s$): ST-LD replay trap
  - Not affected by branches

$$\text{PNR}_{\text{LDQ}} = U_s$$

# EARLY REGISTER RECYCLING

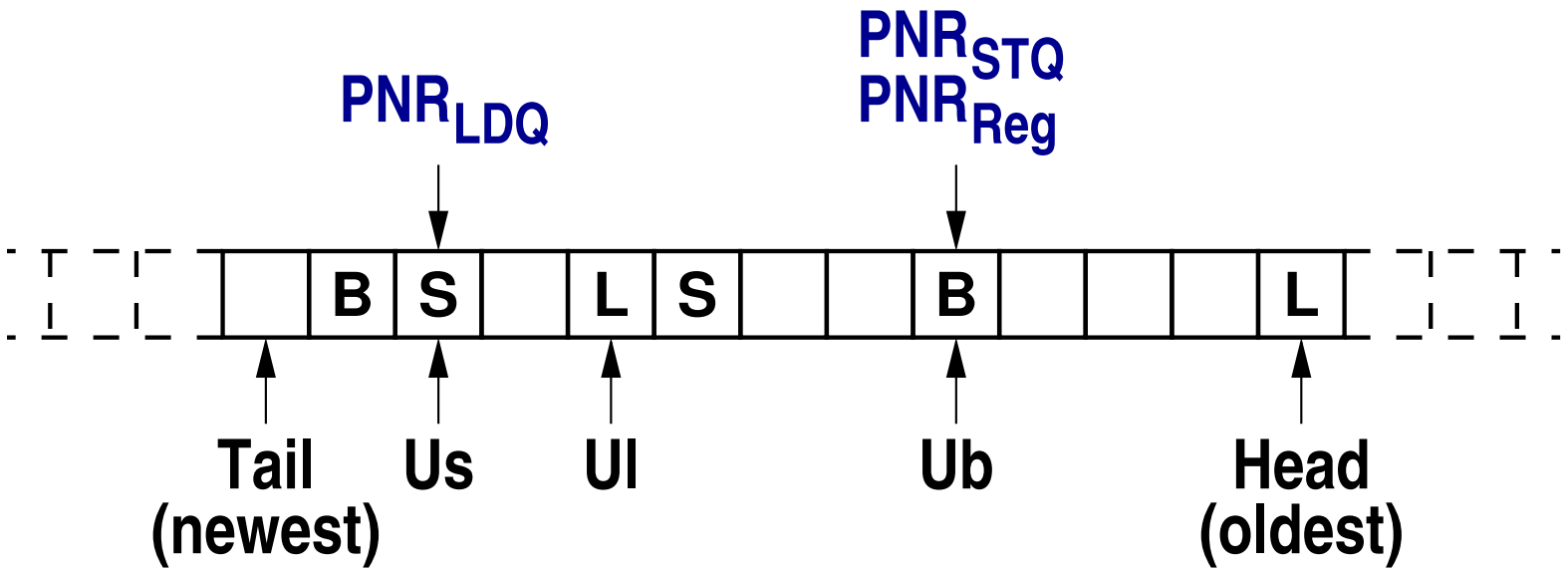
- Constraint: Must not contain potentially useful data
  - older($U_s$): Free of ST-LD replay trap
  - older($U_b$): Free of branch misprediction

$$\text{PNR}_{\text{Reg}} = \text{oldest}(U_S, U_B)$$

- Of course register must be dead (paper)

# Early Recycling Summary

| Resource | PNR Value |
|---|---|
| LDQ entries | $U_S$ |
| STQ entries | oldest$(U_L, U_S, U_B)$ |
| Registers | oldest$(U_S, U_B)$ |



# EVALUATION

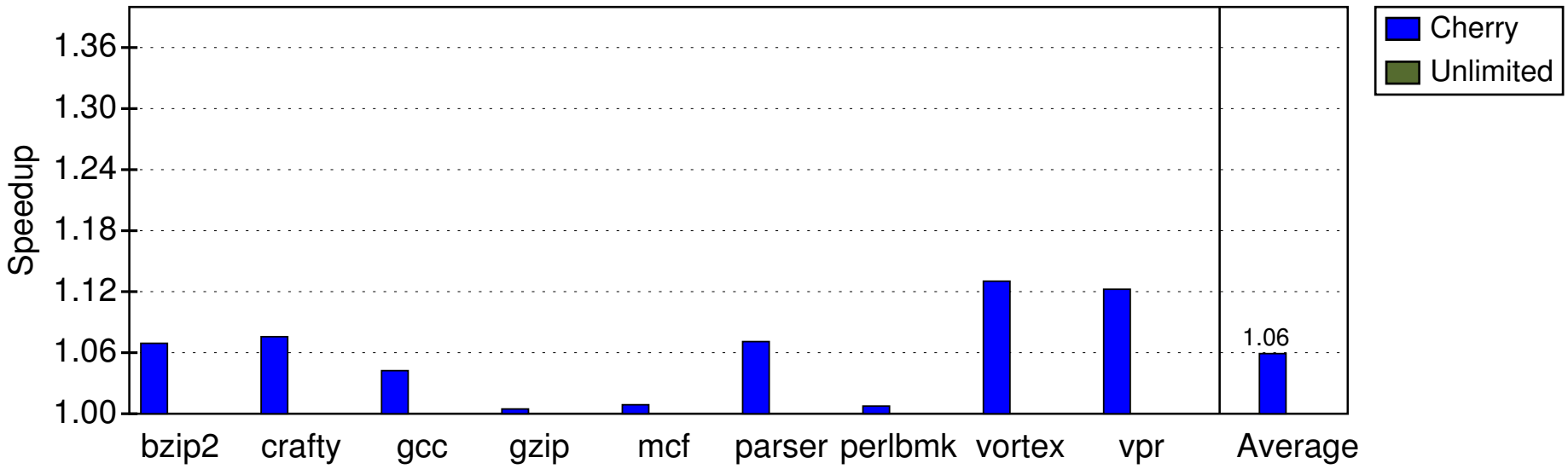
- Execution-driven simulation
- SPEC 2000 applications
- SGI MIPSPro compiler
- Compare Cherry against:
  - Processor with unlimited resources
  - Three aggressive base processor designs (paper)

# PROCESSOR MODEL

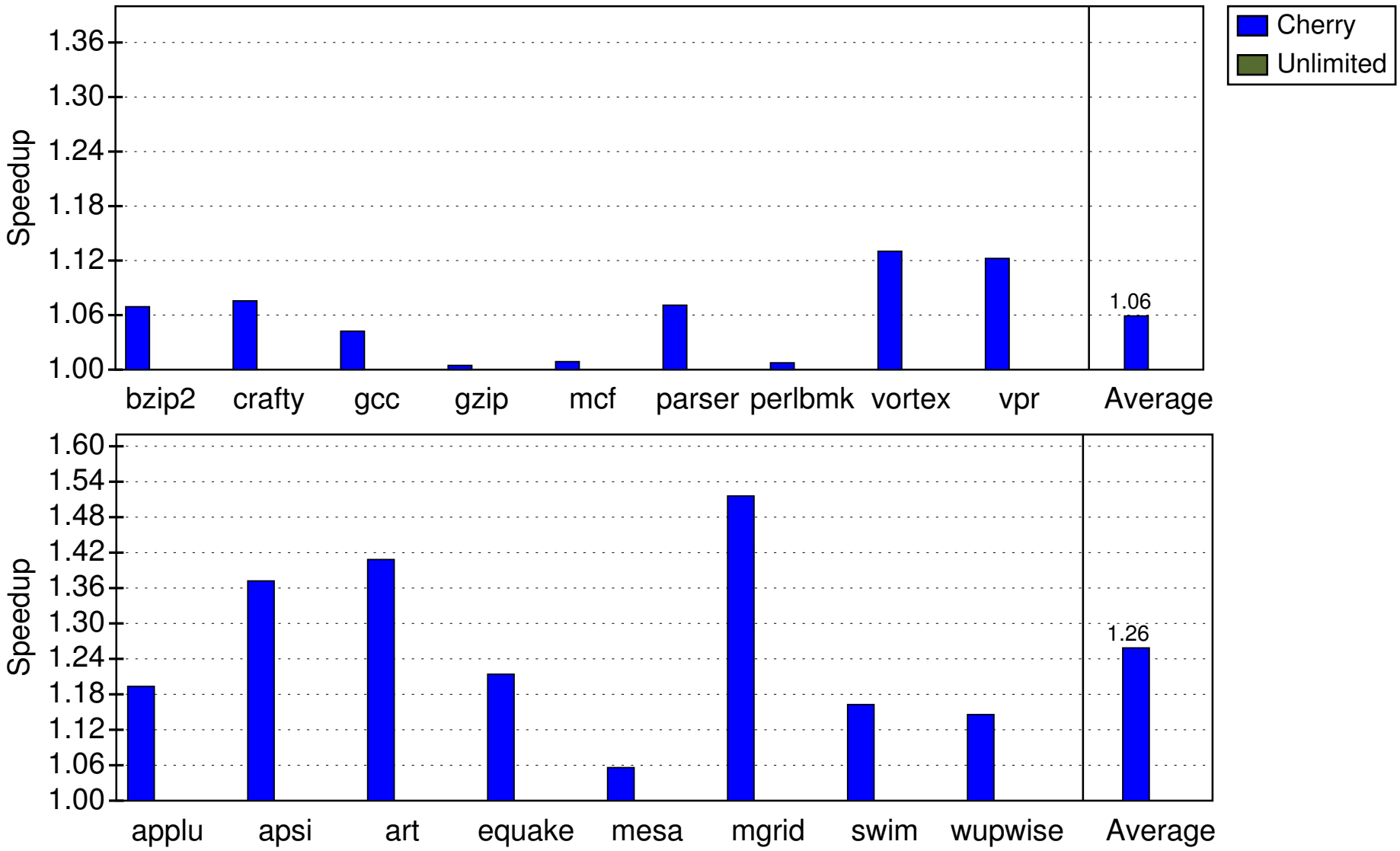
| | |
|---|---|
| Frequency: 3.2GHz | Cherry checkpoint frequency: $5\mu s$ |
| Fetch/issue/commit width: 8/8/12 | Up to 1 taken branch/cycle |
| I. window/ROB size: 128/384 | RAS: 32 entries |
| Int/FP registers : 192/128 | BTB: 4K entries, 4-way assoc. |
| Ld/St units: 2/2 | Branch predictor: |
| Int/FP/branch units: 7/5/3 | Hybrid with speculative update |
| Ld/St queue entries: 32/32 | Bimodal size: 8K entries |
| MSHRs: 24 | Two-level size: 64K entries |

| Cache | L1 | L2 | Bus & Memory |
|---|---|---|---|
| Size: | 32KB | 512KB | FSB frequency: 400MHz |
| RT: | 2 cycles | 10 cycles | FSB width: 128bit |
| Assoc: | 4-way | 8-way | Memory: 4-channel Rambus |
| Line size: | 64B | 128B | DRAM bandwidth: 6.4GB/s |
| Ports: | 4 | 1 | Memory RT: 120ns |

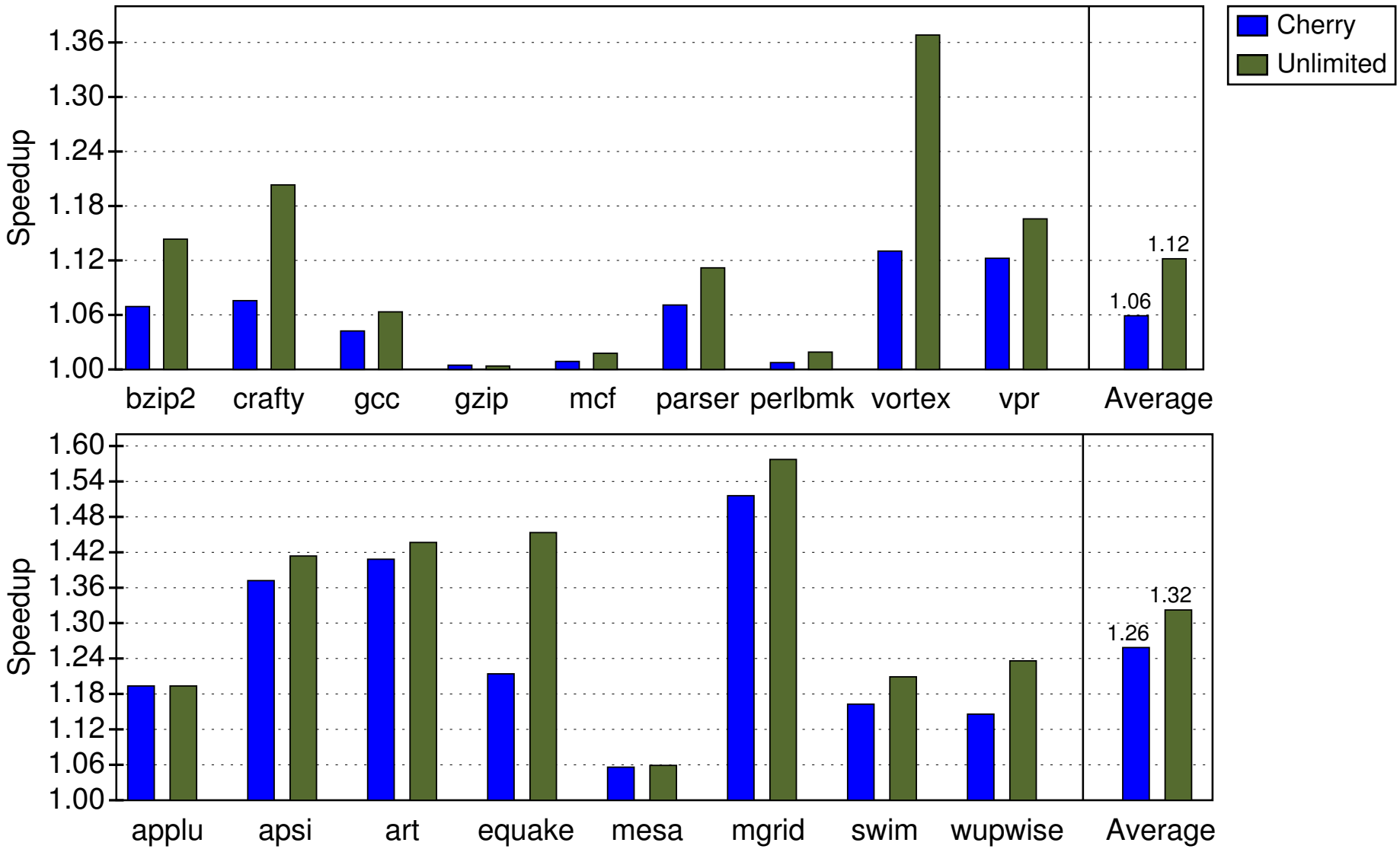
# OVERALL RESULTS



# OVERALL RESULTS

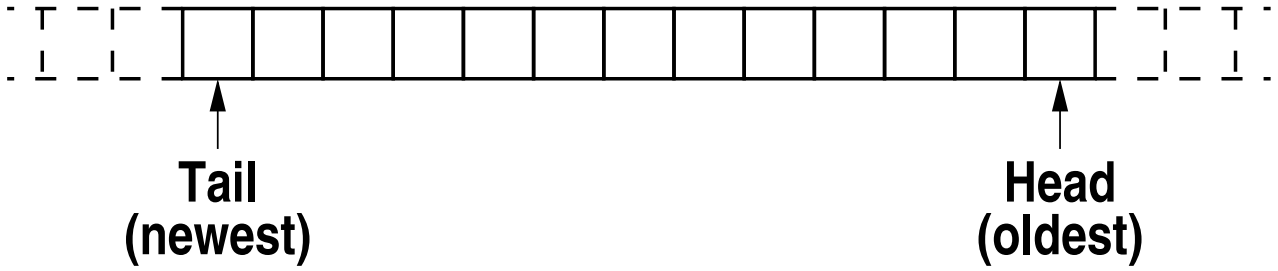


# OVERALL RESULTS



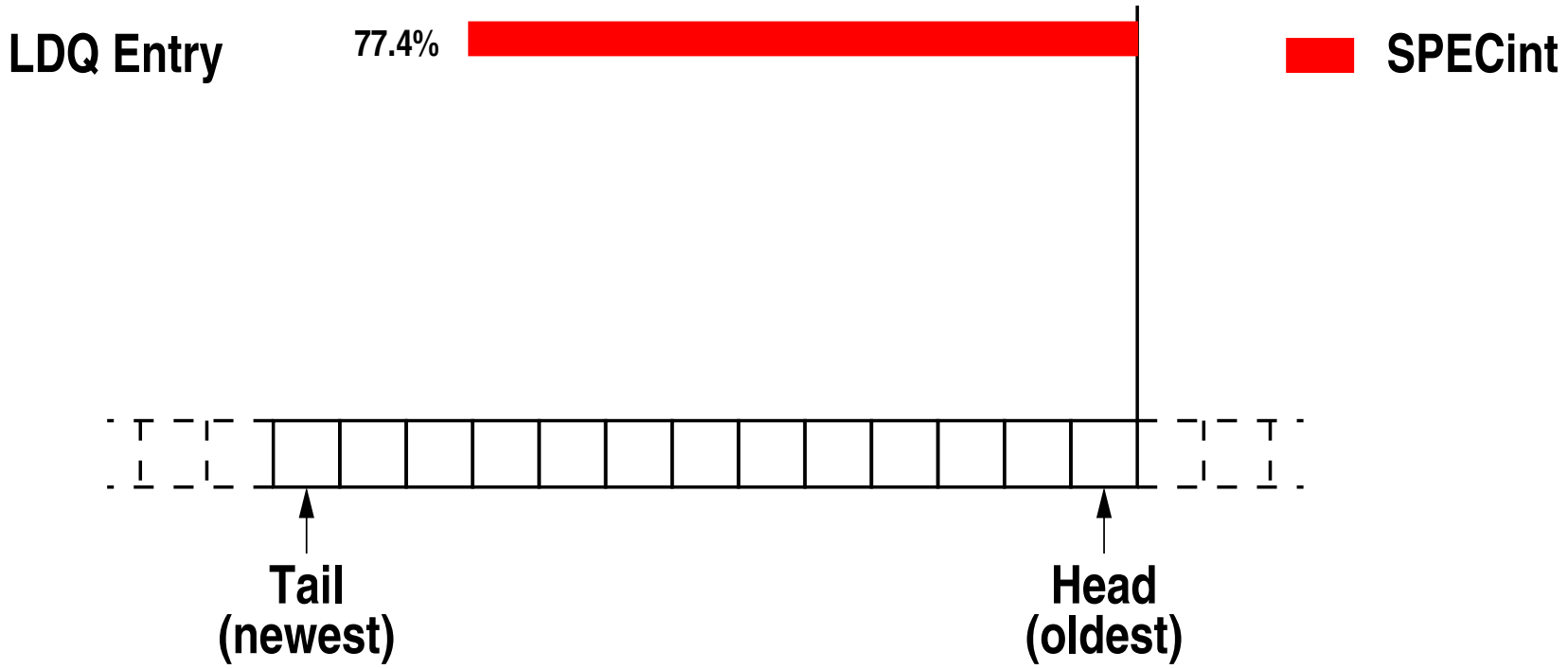
# PNR ADVANCE

- PNR advance as a % of ROB use



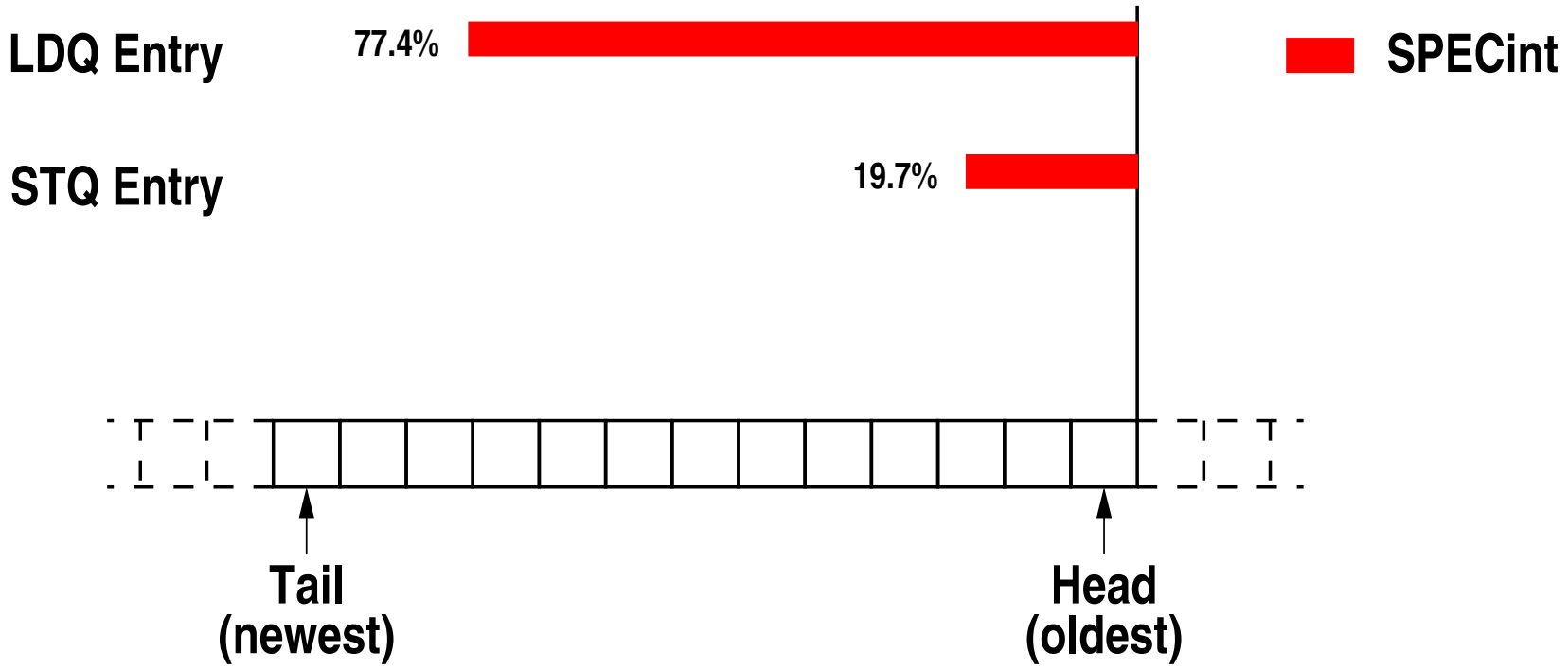
# PNR ADVANCE

- PNR advance as a % of ROB use



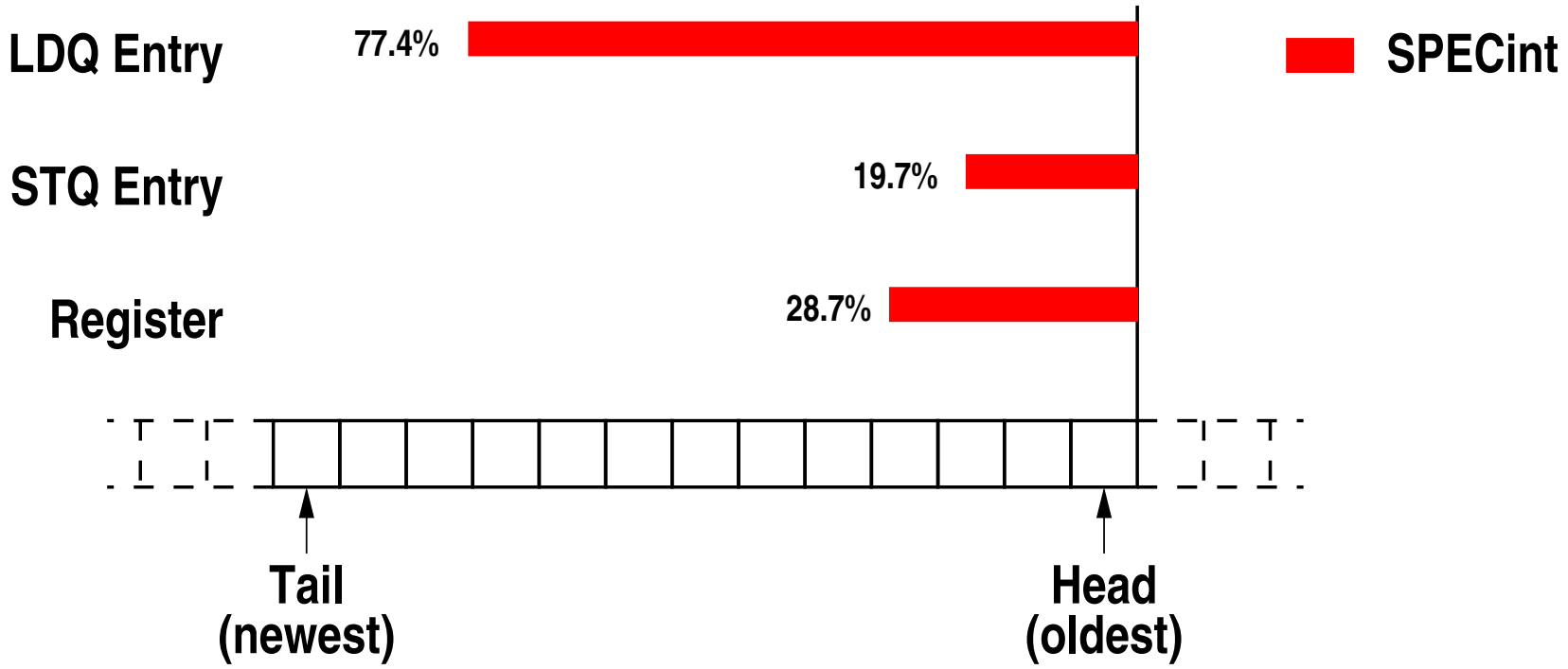
# PNR ADVANCE

- PNR advance as a % of ROB use



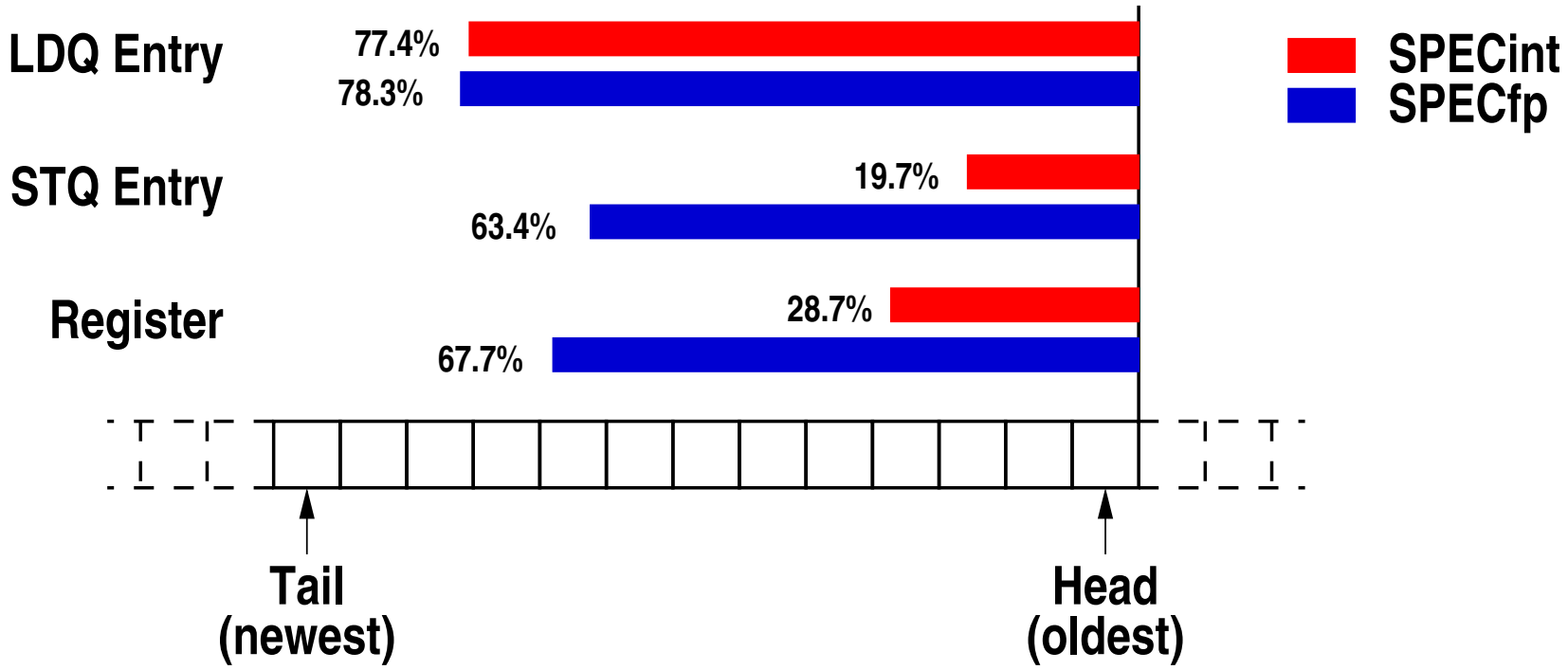
# PNR ADVANCE

- PNR advance as a % of ROB use



# PNR ADVANCE

- PNR advance as a % of ROB use



# SUMMARY

- Decoupling resource recycling from retirement
  - Split ROB into two logical sets
  - Use of ROB+checkpointing simultaneously
- Early recycling mechanism: LDQ, STQ, Registers
- Integration with speculative multithreading (paper)
- Speedup over baseline
  - 1.06 for SPECint2000
  - 1.26 for SPECfp2000

# QUESTIONS?

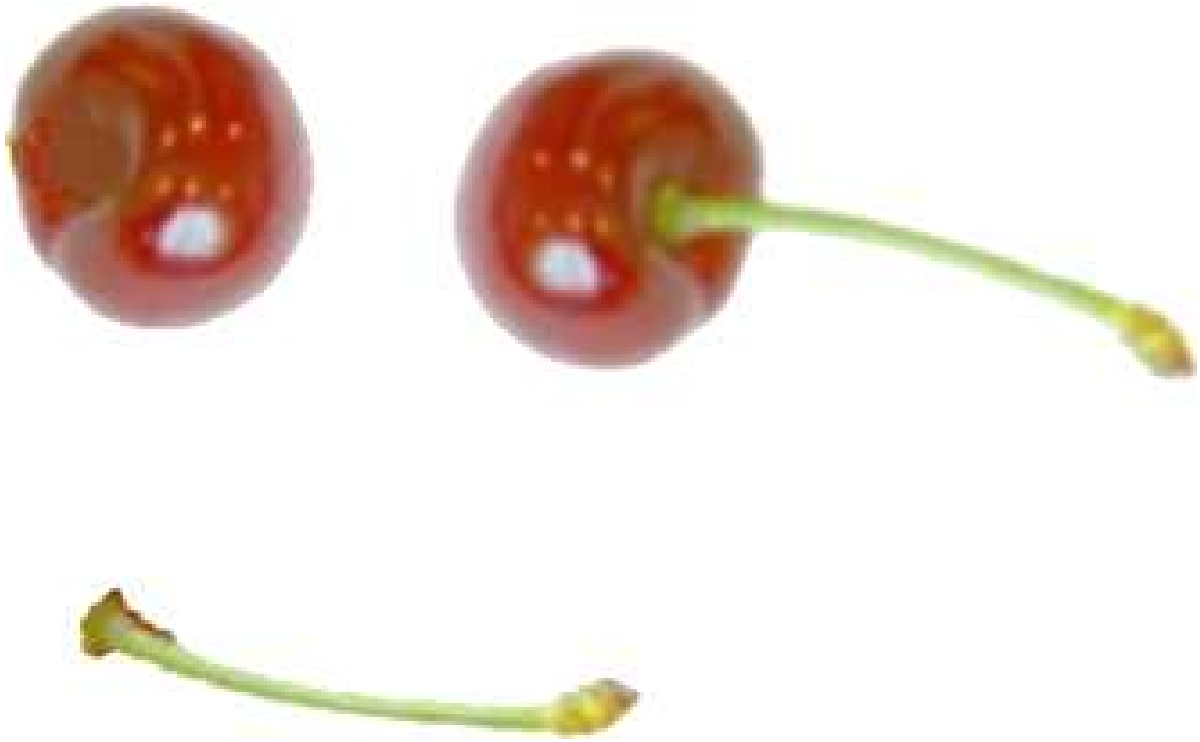

José F. Martínez[1]

Jose Renau[2]

Michael C. Huang[3]

Milos Prvulovic[2]

Josep Torrellas[2]

1 CORNELL

2 ILLINOIS UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

3 UNIVERSITY OF ROCHESTER

# BACKUP SLIDES

# RECYCLING PNRS

| Resource | PNR Value |
|---|---|
| LDQ entries (uniprocessor) | $U_S$ |
| LDQ entries (multiprocessor) | oldest$(U_L, U_S)$ |
| STQ entries | oldest$(U_L, U_S, U_B)$ |
| Registers (uniprocessor) | oldest$(U_S, U_B)$ |
| Registers (multiprocessor) | oldest$(U_L, U_S, U_B)$ |

# EARLY STQ ENTRY RECYCLING

- Constraint: Must not overwrite potentially useful data
  - older($U_l$): Unresolved LD may need old data
  - older($U_s$): No older LD is replayable
  - older($U_b$): Free of branch misprediction

$$\text{PNR}_{\text{STQ}} = \text{oldest}(U_L, U_S, U_B)$$

# CHECKPOINT FREQUENCY

$$T_o = c_k + p_e c_e \tag{1}$$

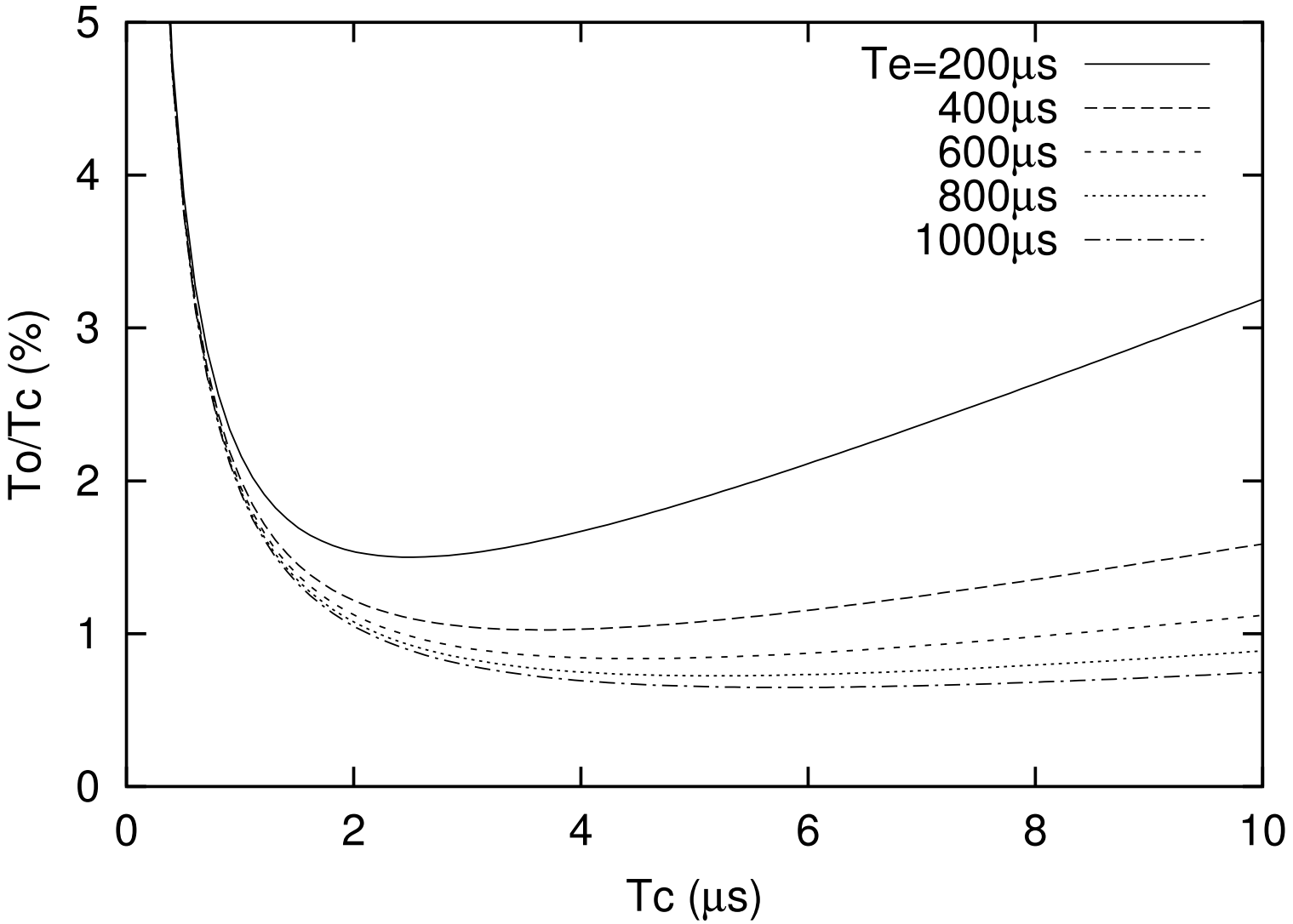
$$p_e = \frac{T_c}{T_e} \tag{2}$$

$$c_e = s\frac{T_c}{2} + (s-1) \tag{3}$$

$$\frac{T_o}{T_c} = \frac{c_k}{T_c} + \left(s - \frac{1}{2}\right)\frac{T_c}{T_e} \tag{4}$$

$$T_c = \sqrt{\frac{c_k T_e}{s - \frac{1}{2}}} \tag{5}$$

# CHECKPOINT SELECTION



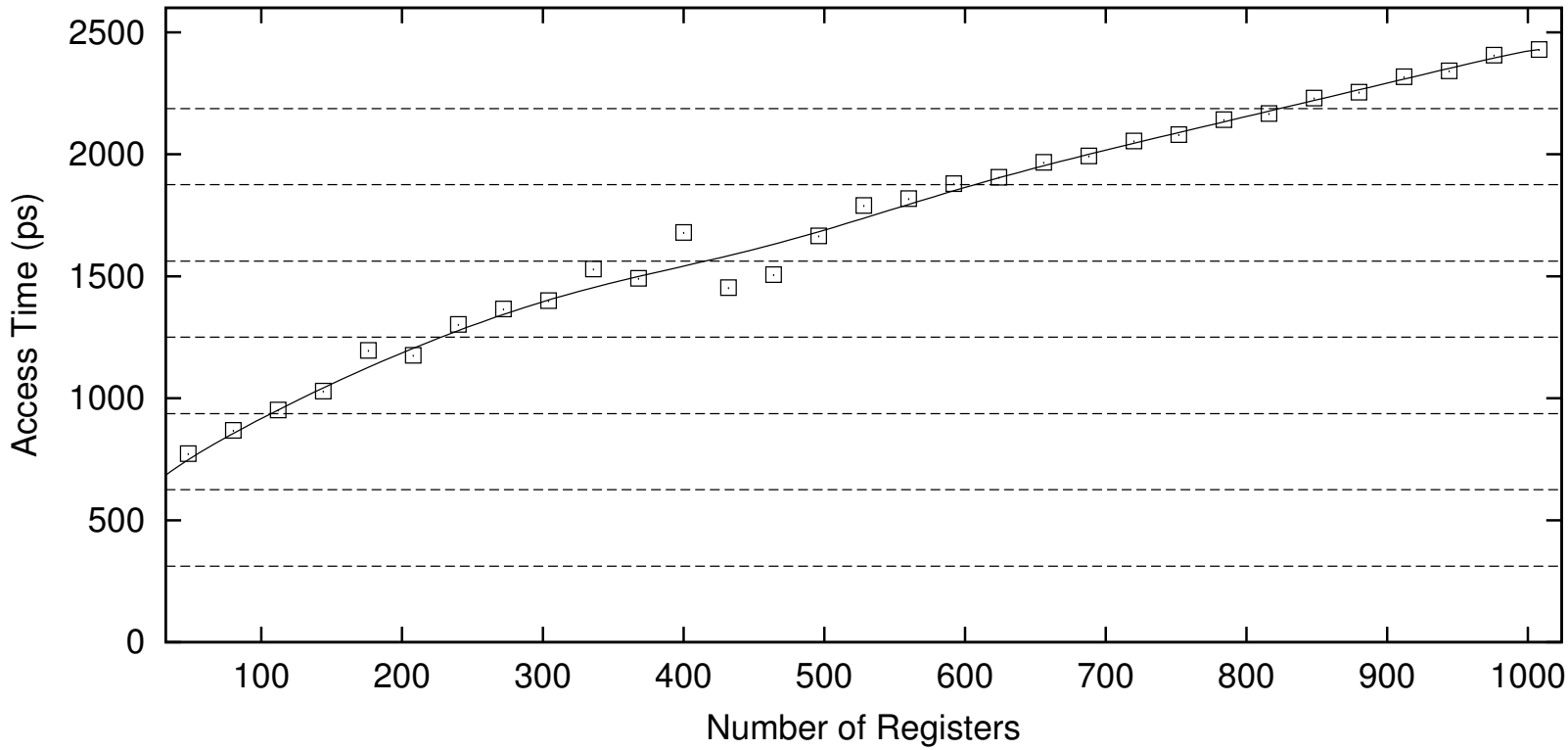
# SPECINT2000 PARAMETERS

| | Apps | Used ROB (%) | Irreversible Set (% of Used ROB) | | | Collapse Step (Cycles) |
|---|---|---|---|---|---|---|
| | | | Reg | LQ | SQ | |
| SPECint | bzip2 | 29.9 | 24.3 | 55.8 | 19.5 | 292.3 |
| | crafty | 28.8 | 33.4 | 97.6 | 28.6 | 41.9 |
| | gcc | 19.1 | 19.0 | 82.3 | 17.8 | 66.9 |
| | gzip | 28.5 | 65.5 | 81.7 | 8.5 | 47.1 |
| | mcf | 30.1 | 14.6 | 37.7 | 13.8 | 695.6 |
| | parser | 30.7 | 26.1 | 80.7 | 21.8 | 109.2 |
| | perlbmk | 12.2 | 24.6 | 89.9 | 20.5 | 23.3 |
| | vortex | 39.3 | 26.3 | 87.1 | 24.9 | 64.4 |
| | vpr | 32.9 | 25.2 | 83.6 | 21.5 | 165.1 |
| | **Average** | **27.9** | **28.7** | **77.4** | **19.7** | **167.3** |

# SPECfp2000 Parameters

| | Apps | Used ROB (%) | Irreversible Set (% of Used ROB) | | | Collapse Step (Cycles) |
|---|---|---|---|---|---|---|
| | | | Reg | LQ | SQ | |
| SPECfp | applu | 62.2 | 61.6 | 62.4 | 60.7 | 411.5 |
| | apsi | 76.8 | 82.3 | 83.1 | 81.6 | 921.1 |
| | art | 88.0 | 54.3 | 62.6 | 29.2 | 1247.3 |
| | equake | 41.6 | 61.6 | 69.1 | 57.3 | 135.2 |
| | mesa | 29.8 | 35.1 | 44.6 | 34.6 | 33.7 |
| | mgrid | 65.1 | 91.5 | 93.5 | 91.3 | 335.9 |
| | swim | 59.4 | 64.8 | 65.4 | 64.7 | 949.1 |
| | wupwise | 71.9 | 90.3 | 71.2 | 87.9 | 190.7 |
| | **Average** | **61.9** | **67.7** | **78.3** | **63.4** | **528.1** |

# Register Access Time



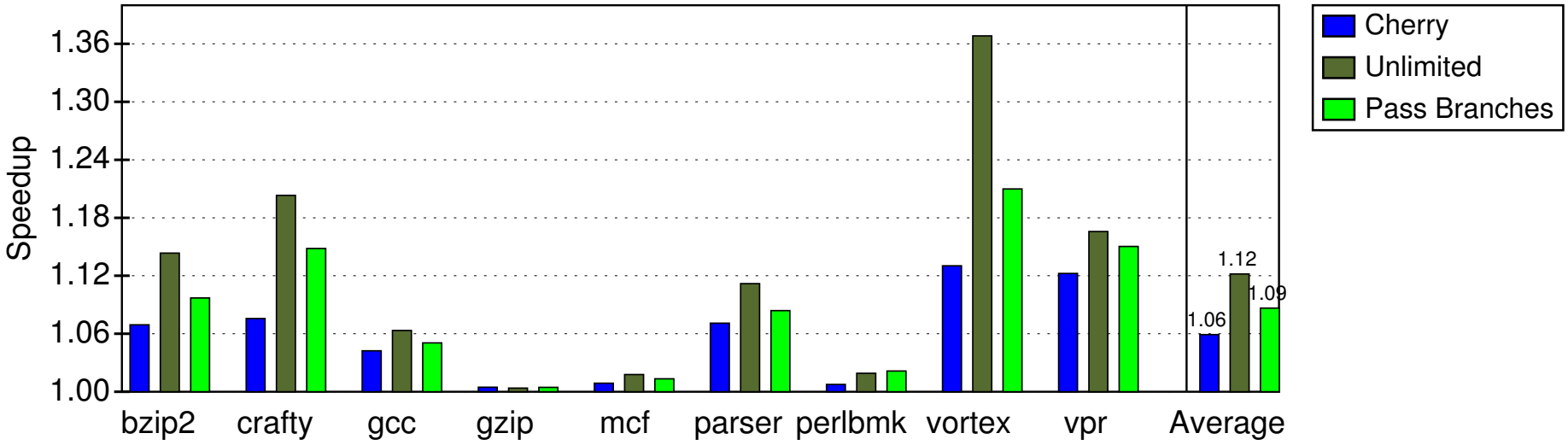
# Other Considerations (paper)

- Exception on instruction older than youngest PNR:
  - Roll back to checkpoint
- Each PNR recycles at its own pace
- Not all PNRs critical, e.g. $PNR_{LDQ}$
- Early register recycling:
  - Must count in-flight consumers (Moudgill et al 1993)
- Optimal interval between checkpoints

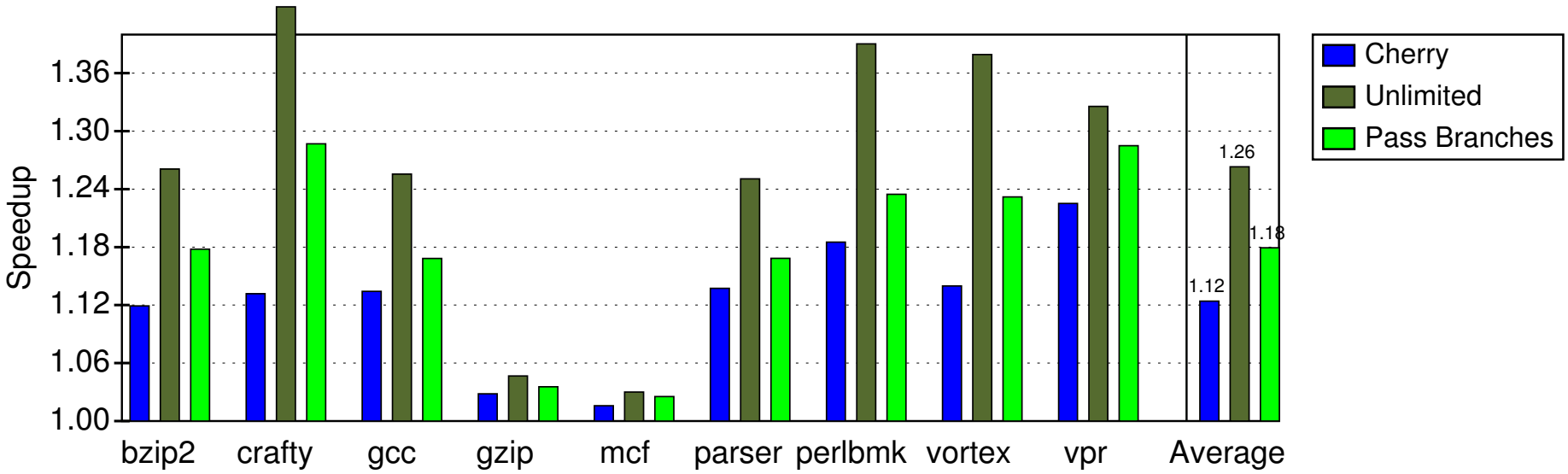
# CHERRY PARAMETERS

- Cherry checkpoint frequency: $5\mu s$
- Checkpoint overhead: 60 cycles ($18.75ns$)
- Avg time between exceptions: $800\mu s$ (not modeled)

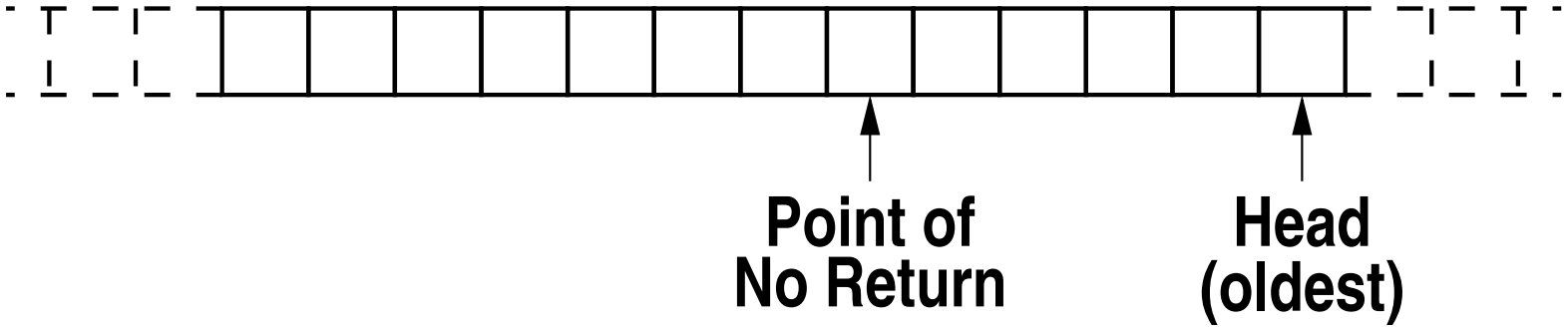
# IGNORING $U_b$


Speedup
1.36
1.30
1.24
1.18
1.12
1.06
1.00
bzip2
crafty
gcc
gzip
mcf
parser
perlbmk
vortex
vpr
Average
1.06
1.12
1.09
Cherry
Unlimited
Pass Branches

# PERFECT BRANCH PREDICTION

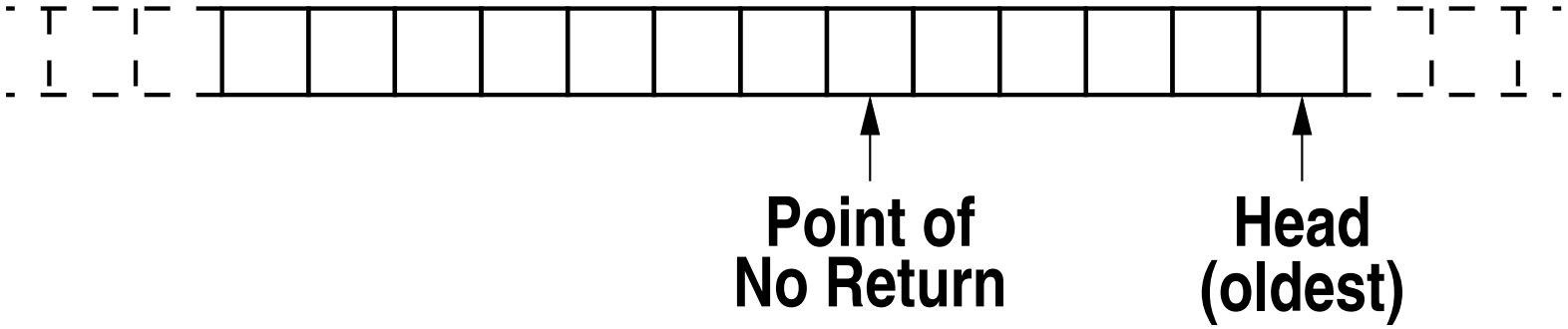


# INTERRUPT COLLAPSING



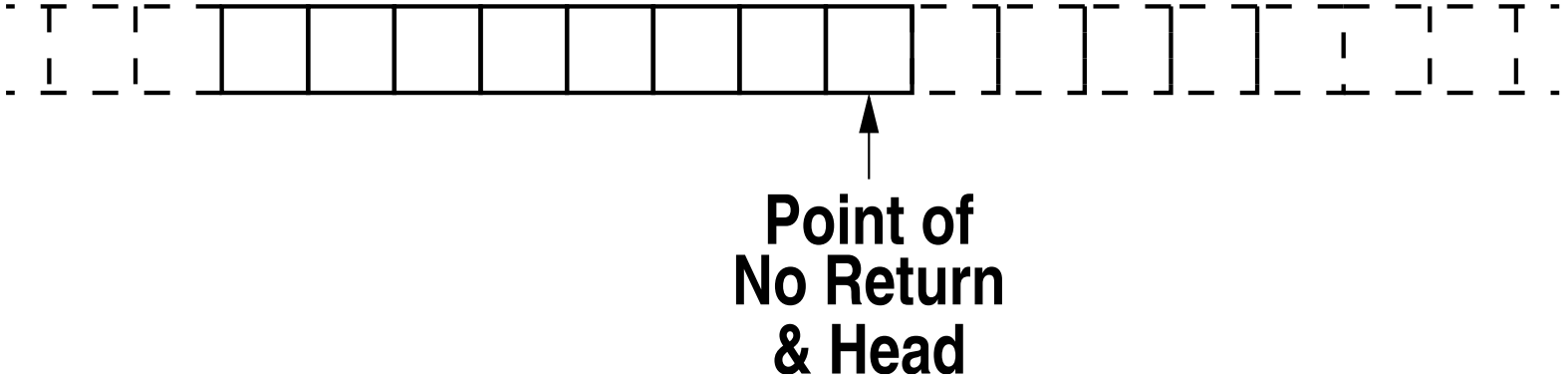
# INTERRUPT COLLAPSING

Interrupt



# INTERRUPT COLLAPSING

Interrupt



# EQUIVALENT SYSTEM

| Resource | Baseline | SPECint2000 | SPECfp2000 |
|---|---|---|---|
| LDQ entries | 32 | 80 | 128 |
| STQ entries | 32 | 64 | 112 |
| Int Registers | 192 | 240 | 240 |
| FP Registers | 128 | 128 | 192 |