Speculative Execution

Daniel Sanchez
Computer Science and Artificial Intelligence Laboratory
M.I.T.
Speculative Execution Recipe

1. Proceed ahead despite unresolved dependencies using a prediction for an architectural or micro-architectural value

2. Maintain both old and new values on updates to architectural (and often micro-architectural) state

3. After sure that there was no mis-speculation and there will be no more uses of the old values, discard old values and just use new values

OR

3. In event of mis-speculation, dispose of all new values, restore old values, and re-execute from point before mis-speculation

Why might one use old values? O-O-O WAR hazards
Value Management Strategies

Greedy (or Eager) Update:
- Update value in place, and
- Provide means to reconstruct old values for recovery
  • often this is a log of old values

Lazy Update:
- Buffer new value, leaving old value in place
- Replace old value only at ‘commit’ time

Why leave an old value in place?
• When there will be limited use of new value
• To make it easy to use old value after new value is generated
• To simplify recovery
Exception Handling
(In-Order Five-Stage Pipeline)

Strategy for Registers?
Where are ‘new’ values?
Strategy for PC?
Where is ‘log’?

Lazy – update at commit
In execution pipeline
Greedy – update immediately
In pipeline of PC latches
Misprediction Recovery

In-order execution machines:
- Guarantee no instruction issued after branch can write-back before branch resolves by keeping values in the pipeline
- Kill all values from all instructions in pipeline behind mispredicted branch

Out-of-order execution?
- Multiple instructions following branch in program order can generate new values before branch resolves
Data-Driven Execution

Renaming table & reg file

Reorder buffer

Basic Operation:
- Enter op and tag or data (if known) for each source
- Replace tag with data as it becomes available
- Issue instruction when all sources are available
- Save dest data when operation finishes

Update strategy? Greedy – update at execute
Rollback and Renaming

Register File
(now holds only committed state)

Reorder buffer

<table>
<thead>
<tr>
<th>Ins#</th>
<th>use</th>
<th>exec</th>
<th>op</th>
<th>p1</th>
<th>src1</th>
<th>b2</th>
<th>src2</th>
<th>pd</th>
<th>dest</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Load Unit → FU → FU → FU → Store Unit → Commit

Convert to lazy by holding data in ROB.

But how do we find values before they are committed?

Search the “dest” field in the reorder buffer
Renaming Table

Micro-architectural speculative cache to speed up tag look up.

What is the update policy of rename table?
What events cause mis-speculation?
How can we respond to mis-speculation?
After being cleared, when can instructions be added to ROB?

Greedy
Exceptions & branch mispredicts
Clear valid bits
After drain
Take snapshot of register rename table at each predicted branch, recover earlier snapshot if branch mispredicted.
### Speculative value management of microarchitectural state

<table>
<thead>
<tr>
<th></th>
<th>Reg Map</th>
<th></th>
<th>Snap Map</th>
<th></th>
<th>Snap Map</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>V</td>
<td></td>
<td>V</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>R0</td>
<td></td>
<td></td>
<td>T20</td>
<td>X</td>
<td>T20</td>
<td>X</td>
</tr>
<tr>
<td>R1</td>
<td></td>
<td></td>
<td>T73</td>
<td>X</td>
<td>T73</td>
<td>X</td>
</tr>
<tr>
<td>R2</td>
<td></td>
<td></td>
<td>T45</td>
<td>X</td>
<td>T45</td>
<td>X</td>
</tr>
<tr>
<td>R3</td>
<td></td>
<td></td>
<td>T128</td>
<td></td>
<td>T128</td>
<td></td>
</tr>
<tr>
<td>R30</td>
<td></td>
<td></td>
<td>T54</td>
<td></td>
<td>T54</td>
<td></td>
</tr>
<tr>
<td>R31</td>
<td></td>
<td></td>
<td>T88</td>
<td>X</td>
<td>T88</td>
<td>X</td>
</tr>
</tbody>
</table>

What kind of value management is this? **Greedy!!**
# Branch Predictor Recovery

## 1-Bit Counter Recovery

<table>
<thead>
<tr>
<th>PC</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

- **Lazy**

## 2-Bit Counter Recovery

| PC | 00 | 11 | 01 | 10 |

- **Lazy**

## Global History Recovery

10101010

- **Greedy**

## Local History Recovery

<table>
<thead>
<tr>
<th>PC</th>
<th>10101010</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01010101</td>
</tr>
</tbody>
</table>

- **Greedy!!**
O-o-O Execution with ROB
Data-in-ROB design

Basic Operation:
• Enter op and tag or data (if known) for each source
• Replace tag with data as it becomes available
• Issue instruction when all sources are available
• Save dest data when operation finishes
• Commit saved dest data when instruction commits
Unified Physical Register File
(MIPS R10K, Alpha 21264, Pentium 4)

- One regfile for both *committed* and *speculative* values (no data in ROB)
- During decode, instruction result allocated new physical register, source regs translated to physical regs through rename table
- Instruction reads data from regfile at start of execute (not in decode)
- Write-back updates reg. busy bits on instructions in ROB (assoc. search)
- Snapshots of rename table taken at every branch to recover mispredictions
- On exception, renaming undone in reverse order of issue (*MIPS R10000*)
Speculative & Out-of-Order Execution

- **Fetch**
- **Decode & Rename**
- **Reorder Buffer**
- **Commit**

**In-Order**
- **PC**
- **Fetch**
- **Decode & Rename**
- **Reorder Buffer**
- **Commit**

**Out-of-Order**
- **Branch Prediction**
- **Branch Resolution**
- **Update predictors**

**Execute**
- **Physical Reg. File**
- **Branch Unit**
- **ALU**
- **MEM**
- **Store Buffer**
- **D$**
Lifetime of Physical Registers

- Physical regfile holds committed and speculative values
- Physical registers decoupled from ROB entries (*no data in ROB*)

a) \( \text{ld } r1, (r3) \)  
b) \( \text{add } r3, r1, #4 \)  
c) \( \text{sub } r1, r3, r9 \)  
d) \( \text{add } r3, r1, r7 \)  
e) \( \text{ld } r6, (r1) \)  
f) \( \text{add } r8, r6, r3 \)  
g) \( \text{st } r8, (r1) \)  
h) \( \text{ld } r3, (r11) \)

\( \text{ld } P1, (Px) \)  
\( \text{add } P2, P1, #4 \)  
\( \text{sub } P3, P2, Py \)  
\( \text{add } P4, P3, Pz \)  
\( \text{ld } P5, (P3) \)  
\( \text{add } P6, P5, P4 \)  
\( \text{st } P6, (P3) \)  
\( \text{ld } P7, (Pw) \)

When can we reuse a physical register?  
*When next write to same architectural register commits*
Physical Register Management

### Rename Table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>P8</td>
</tr>
<tr>
<td>R2</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>P7</td>
</tr>
<tr>
<td>R4</td>
<td></td>
</tr>
<tr>
<td>R5</td>
<td></td>
</tr>
<tr>
<td>R6</td>
<td>P5</td>
</tr>
<tr>
<td>R7</td>
<td>P6</td>
</tr>
</tbody>
</table>

### Physical Regs

- P0
- P1
- P2
- P3
- P4
- P5 <R6>
- P6 <R7>
- P7 <R3>
- P8 <R1>

### Free List

- P0
- P1
- P3
- P2
- P4

### ROB

<table>
<thead>
<tr>
<th></th>
<th>use</th>
<th>ex</th>
<th>op</th>
<th>p1</th>
<th>PR1</th>
<th>p2</th>
<th>PR2</th>
<th>Rd</th>
<th>LPRd</th>
<th>PRd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example Instructions

- `ld r1, 0(r3)`
- `add r3, r1, #4`
- `sub r6, r7, r6`
- `add r3, r3, r6`
- `ld r6, 0(r1)`

*(LPRd requires third read port on Rename Table for each instruction)*
ld r1, 0(r3)
add r3, r1, #4
sub r6, r7, r6
add r3, r3, r6
ld r6, 0(r1)
Physical Register Management

Rename Table

Physical Regs

Free List

ld r1, 0(r3)
add r3, r1, #4
sub r6, r7, r6
add r3, r3, r6
ld r6, 0(r1)

ROB

<table>
<thead>
<tr>
<th>use</th>
<th>ex</th>
<th>op</th>
<th>p1</th>
<th>PR1</th>
<th>p2</th>
<th>PR2</th>
<th>Rd</th>
<th>LPRd</th>
<th>PRd</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td>ld</td>
<td>p</td>
<td>P7</td>
<td></td>
<td></td>
<td>r1</td>
<td>P8</td>
<td>P0</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>add</td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
<td>r3</td>
<td>P7</td>
<td>P1</td>
</tr>
</tbody>
</table>
Physical Register Management

### Rename Table

<table>
<thead>
<tr>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P0</td>
<td></td>
<td>P1</td>
<td></td>
<td></td>
<td>P5</td>
<td>P3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P6</td>
</tr>
</tbody>
</table>

### Physical Regs

<table>
<thead>
<tr>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;R6&gt;</td>
<td>p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;R7&gt;</td>
<td>p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;R3&gt;</td>
<td>p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;R1&gt;</td>
<td>p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Free List

- P0
- P1
- P3
- P2
- P4

### ROB

<table>
<thead>
<tr>
<th>use</th>
<th>ex</th>
<th>op</th>
<th>p1</th>
<th>PR1</th>
<th>p2</th>
<th>PR2</th>
<th>Rd</th>
<th>LPRd</th>
<th>PRd</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td>ld</td>
<td>p</td>
<td>P7</td>
<td></td>
<td></td>
<td>r1</td>
<td>P8</td>
<td>P0</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>add</td>
<td>P0</td>
<td></td>
<td>r3</td>
<td>P7</td>
<td>P1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>sub</td>
<td>p</td>
<td>P6</td>
<td>P5</td>
<td>r6</td>
<td>P5</td>
<td>P3</td>
<td></td>
</tr>
</tbody>
</table>
Physical Register Management

**Rename Table**

<table>
<thead>
<tr>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td>P6</td>
</tr>
</tbody>
</table>

**Physical Regs**

<table>
<thead>
<tr>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P5</td>
<td>P6</td>
<td>P7</td>
<td>P8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;R6&gt;</td>
<td>&lt;R7&gt;</td>
<td>&lt;R3&gt;</td>
<td>&lt;R1&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>p</td>
<td>p</td>
<td>p</td>
<td>p</td>
</tr>
</tbody>
</table>

**Free List**

<table>
<thead>
<tr>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>P2</td>
<td>P4</td>
</tr>
</tbody>
</table>

**Rob**

<table>
<thead>
<tr>
<th>use</th>
<th>ex</th>
<th>op</th>
<th>p1</th>
<th>PR1</th>
<th>p2</th>
<th>PR2</th>
<th>Rd</th>
<th>LPRd</th>
<th>PRd</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td>ld</td>
<td>p</td>
<td>P7</td>
<td></td>
<td></td>
<td>r1</td>
<td>P8</td>
<td>P0</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>add</td>
<td></td>
<td>P0</td>
<td>p</td>
<td>P5</td>
<td>r3</td>
<td>P7</td>
<td>P1</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>sub</td>
<td>p</td>
<td>P6</td>
<td></td>
<td></td>
<td>r6</td>
<td>P5</td>
<td>P3</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>add</td>
<td></td>
<td>P1</td>
<td>P3</td>
<td></td>
<td>r3</td>
<td>P1</td>
<td>P2</td>
</tr>
</tbody>
</table>

- ld r1, 0(r3)
- add r3, r1, #4
- sub r6, r7, r6
- add r3, r3, r6
- ld r6, 0(r1)
Physical Register Management

Rename Table

Physical Regs

Free List

ld r1, 0(r3)
add r3, r1, #4
sub r6, r7, r6
add r3, r3, r6
ld r6, 0(r1)

ROB

<table>
<thead>
<tr>
<th>use</th>
<th>ex</th>
<th>op</th>
<th>p1</th>
<th>PR1</th>
<th>p2</th>
<th>PR2</th>
<th>Rd</th>
<th>LPRd</th>
<th>PRd</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td>ld</td>
<td>p</td>
<td>P7</td>
<td></td>
<td></td>
<td>r1</td>
<td>P8</td>
<td>P0</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>add</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td>r3</td>
<td>P7</td>
<td>P1</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>sub</td>
<td>p</td>
<td>P6</td>
<td>p</td>
<td>P5</td>
<td>r6</td>
<td>P5</td>
<td>P3</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>add</td>
<td></td>
<td>P1</td>
<td></td>
<td>P3</td>
<td>r3</td>
<td>P1</td>
<td>P2</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>ld</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td>r6</td>
<td>P3</td>
<td>P4</td>
</tr>
</tbody>
</table>

March 13, 2019
Physical Register Management

**Rename Table**

<table>
<thead>
<tr>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P6</td>
<td></td>
<td>P6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Physical Regs**

<table>
<thead>
<tr>
<th>Physical Regs</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
</tr>
<tr>
<td>&lt;R1&gt;</td>
</tr>
<tr>
<td>p</td>
</tr>
</tbody>
</table>

**Free List**

<table>
<thead>
<tr>
<th>Free List</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
</tr>
</tbody>
</table>

**ROB**

<table>
<thead>
<tr>
<th>ROB</th>
</tr>
</thead>
<tbody>
<tr>
<td>use</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>x</td>
</tr>
</tbody>
</table>

Execute & Commit

- ld r1, 0(r3)
- add r3, r1, #4
- sub r6, r7, r6
- add r3, r3, r6
- ld r6, 0(r1)
Physical Register Management

### Rename Table
- R0
- R1: PR0
- R2
- R3: PR2
- R4
- R5: PR4
- R6: PR6
- R7: PR7

### Physical Regs
- P0: <R1>
- P1: <R3>
- P2
- P3
- P4
- P5: <R6>
- P6: <R7>
- P7: <R3>
- P8

### Free List
- P0
- P1
- P2
- P3
- P4
- P5
- P6
- P7
- P8

### ROB

<table>
<thead>
<tr>
<th>use</th>
<th>ex</th>
<th>op</th>
<th>p1</th>
<th>PR1</th>
<th>p2</th>
<th>PR2</th>
<th>Rd</th>
<th>LPRd</th>
<th>PRd</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>ld</td>
<td>p</td>
<td>P7</td>
<td></td>
<td></td>
<td>r1</td>
<td>P8</td>
<td>P0</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>add</td>
<td>p</td>
<td>P0</td>
<td></td>
<td></td>
<td>r3</td>
<td>P7</td>
<td>P1</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>sub</td>
<td>p</td>
<td>P6</td>
<td>p</td>
<td>P5</td>
<td>r6</td>
<td>P5</td>
<td>P3</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>P1</td>
<td>P3</td>
<td>r3</td>
<td></td>
<td>P1</td>
<td>P2</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td>r6</td>
<td>P3</td>
<td>P4</td>
</tr>
</tbody>
</table>

- ld r1, 0(r3)
- add r3, r1, #4
- sub r6, r7, r6
- add r3, r3, r6
- ld r6, 0(r1)

- Execute & Commit
Reorder Buffer Holds
Active Instruction Window

... (Older instructions)

ld r1, (r3)
add r3, r1, r2
sub r6, r7, r9
add r3, r3, r6
ld r6, (r1)
add r6, r6, r3
st r6, (r1)
ld r6, (r1)
... (Newer instructions)

Cycle \( t \)

Key: predecode, decoded, issued, executed, committed

... Commit

... Issue

... Execute

... Decode

... (Newer instructions)

ld r1, (r3)
add r3, r1, r2
sub r6, r7, r9
add r3, r3, r6
ld r6, (r1)
add r6, r6, r3
st r6, (r1)
ld r6, (r1)
... (Older instructions)

Cycle \( t + 1 \)
### Issue Timing

<table>
<thead>
<tr>
<th>i1</th>
<th>Add R1,R1,#1</th>
<th>Issue(_1)</th>
<th>Execute(_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i2</td>
<td>Sub R1,R1,#1</td>
<td>Issue(_2)</td>
<td>Execute(_2)</td>
</tr>
</tbody>
</table>

How can we issue earlier?

**Using knowledge of execution latency (bypass)**

<table>
<thead>
<tr>
<th>i1</th>
<th>LD R1, (R3)</th>
<th>Issue(_1)</th>
<th>Execute(_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i2</td>
<td>Sub R1,R1,#1</td>
<td>Issue(_2)</td>
<td>Execute(_2)</td>
</tr>
</tbody>
</table>

What might make this schedule fail?

**If execution latency wasn’t as expected**
### Issue Queue with latency prediction

**Issue Queue (Reorder buffer)**

- **Fixed latency**: latency included in queue entry (‘bypassed’)
- **Predicted latency**: latency included in queue entry (speculated)
- **Variable latency**: wait for completion signal (stall)

---

<table>
<thead>
<tr>
<th>Inst#</th>
<th>use</th>
<th>exec</th>
<th>op</th>
<th>p1</th>
<th>lat1</th>
<th>src1</th>
<th>p2</th>
<th>lat2</th>
<th>src2</th>
<th>dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEQZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Speculative Instructions**
Data-in-ROB vs. Single Register File

How does issue speculation differ, e.g., on cache miss?
Dependency loop shorter for data-in-ROB style
### Superscalar Register Renaming

- During decode, instructions allocated new physical destination register
- Source operands renamed to physical register with newest value
- Execution unit only sees physical register numbers

#### Rename Table

<table>
<thead>
<tr>
<th>Op</th>
<th>Src1</th>
<th>Src2</th>
<th>Dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op</td>
<td>PSrc1</td>
<td>PSrc2</td>
<td>PDest</td>
</tr>
</tbody>
</table>

#### Register Free List

- Write
- Read Data
- Read Addresses
- Rename Table

#### Does this work?

---

March 13, 2019
Superscalar Register Renaming

Must check for RAW hazards between instructions issuing in same cycle. Can be done in parallel with rename lookup.

*MIPS R10K renames 4 serially-RAW-dependent insts/cycle*
Split Issue and Commit Queues

• How large should the ROB be?
  – Think Little’s Law...

• Can split ROB into issue and commit queues

  **Issue Queue**

<table>
<thead>
<tr>
<th>use</th>
<th>op</th>
<th>p1</th>
<th>PR1</th>
<th>p2</th>
<th>PR2</th>
<th>tag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  **Commit Queue**

<table>
<thead>
<tr>
<th>ex</th>
<th>Rd</th>
<th>LPRd</th>
<th>PRd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Commit queue: Allocate on decode, free on commit
• Issue queue: Allocate on decode, free on dispatch
• Pros: Smaller issue queue → simpler dispatch logic
• Cons: More complex mis-speculation recovery
Speculating Both Directions?

An alternative to branch prediction is to execute both directions of a branch *speculatively*

- Resource requirement is proportional to the number of concurrent speculative executions

- Only half the resources engage in useful work when both directions of a branch are executed speculatively

- Branch prediction takes less resources than speculative execution of both paths

*With accurate branch prediction, it is more cost effective to dedicate all resources to the predicted direction*
Thank you!