Speculative Execution with Late Recovery

Speculative Execution with Early Recovery
  2.1. Adding Speculative Bits
  2.2. Adding Rename-Table Snapshots

Complete Out-of-Order Superscalar TinyRV2 Processor
1. Speculative Execution with Late Recovery

- Every instruction is actually speculative because an older in-flight instruction might cause an exception
- We recover from exceptions at the commit point (C-stage) which is late in the pipeline

- With out-of-order load/store issue, loads (and dependent instructions) are also speculative
- We recover from incorrect speculation in the C stage which is late in the pipeline
1. Speculative Execution with Late Recovery

- Branches also require speculative execution
- Recover mispredictions late in the pipeline?

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
a: lw  x1, 0(x2)
b: mul x3, x1, x4
c: sw  x3, 0(x5)
d: addi x2, x2, 4
e: addi x5, x5, 4
f: addi x6, x6, -1
g: bne x6, x0, loop
```

- Branches are far more common than exceptions and memory-dependence violations
- Accurate branch prediction helps, but some branches are just inherently difficult to predict
- **Key Idea:** Recover from branch mispredictions as soon as possible
2. Speculative Execution with Early Recovery

We will explore early recovery in two steps:

- Adding speculative bits
- Adding rename-table snapshots

2.1. Adding Speculative Bits

- Add a speculative bit to the IQ, ROB, FSB, FLB, and functional units
- Add a speculative mode bit in the D stage

In D stage for a branch
- Set speculative mode bit
- All inst after branch carry speculative bit into IQ, ROB, FSB, LB, func units

In X stage for a correctly predicted branch
- Broadcast clear speculative bit from X stage to all data structures

In X stage for an incorrectly predicted branch
- Broadcast squash signal from X stage to all of these data structures
- Each data structure invalidates entry/inst for which speculative bit is set
- Start fetching from correct address

Multiple speculative bits enable multiple spec branches in flight
- Given instruction can be squashed by multiple branches
- Treat multiple speculative bits as “branch mask”
**Do not copy ARF into PRF on branch misprediction recovery**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>a:</td>
<td>addi x1, x2, 1</td>
<td></td>
<td></td>
<td></td>
<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>b:</td>
<td>branch L1</td>
<td></td>
<td></td>
<td></td>
<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>c:</td>
<td>addi x1, x3, 1</td>
<td></td>
<td></td>
<td></td>
<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>d:</td>
<td>opA</td>
<td></td>
<td></td>
<td></td>
<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>e:</td>
<td>opB</td>
<td></td>
<td></td>
<td></td>
<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>f:</td>
<td>opC</td>
<td></td>
<td></td>
<td></td>
<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>g:</td>
<td>opD</td>
<td></td>
<td></td>
<td></td>
<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>h:</td>
<td>L1: addi x4, x1, 1</td>
<td></td>
<td></td>
<td></td>
<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>

**Copy ARF into PRF on branch misprediction recovery**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>a:</td>
<td>addi x1, x2, 1</td>
<td></td>
<td></td>
<td></td>
<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>b:</td>
<td>addi x1, x3, 1</td>
<td></td>
<td></td>
<td></td>
<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>c:</td>
<td>addi x4, x1, 1</td>
<td></td>
<td></td>
<td></td>
<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>d:</td>
<td>branch L1</td>
<td></td>
<td></td>
<td></td>
<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>e:</td>
<td>opA</td>
<td></td>
<td></td>
<td></td>
<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>f:</td>
<td>opB</td>
<td></td>
<td></td>
<td></td>
<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>g:</td>
<td>opC</td>
<td></td>
<td></td>
<td></td>
<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>h:</td>
<td>opD</td>
<td></td>
<td></td>
<td></td>
<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>i:</td>
<td>L1: addi x5, x6, 1</td>
<td></td>
<td></td>
<td></td>
<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>

- Need to make copy of “precise” ARF in D on every branch ...
- ... but ARF is not precise in D
- Need “view” of what precise ARF would be in D on every branch ...
- ... this is the rename table!
2.2. Adding Rename-Table Snapshots

- Add a speculative bit to the IQ, ROB, FSB, FLB, and functional units
- Add a speculative mode bit in the D stage
- Add a rename table snapshot in the D stage

- In D stage for a branch
  - Set speculative mode bit
  - All inst after branch carry speculative bit into IQ, ROB, FSB, LB, func units
  - Create a RT snapshot to save “view” of precise ARF for branch

- In X stage for a correctly predicted branch
  - Broadcast clear speculative bit from X stage to all data structures

- In X stage for an incorrectly predicted branch
  - Broadcast squash signal from X stage to all of these data structures
  - Each data structure invalidates entry/inst for which speculative bit is set
  - Restore RT from snapshot
  - Start fetching from correct address

- Need multiple speculative bits and multiple snapshots to support multiple speculative branches in flight
RT snapshots squash speculative state

<table>
<thead>
<tr>
<th></th>
<th>a: addi x1, x2, 1</th>
<th>b: branch L1</th>
<th>c: addi x1, x3, 1</th>
<th>d: opA</th>
<th>e: opB</th>
<th>f: opC</th>
<th>g: opD</th>
<th>h: L1: addi x4, x1, 1</th>
</tr>
</thead>
</table>

RT snapshots prevent overwriting non-speculative state

<table>
<thead>
<tr>
<th></th>
<th>a: addi x1, x2, 1</th>
<th>b: addi x1, x3, 1</th>
<th>c: addi x4, x1, 1</th>
<th>d: branch L1</th>
<th>e: opA</th>
<th>f: opB</th>
<th>g: opC</th>
<th>h: opD</th>
<th>i: L1: addi x5, x6, 1</th>
</tr>
</thead>
</table>
3. Complete Out-of-Order Superscalar TinyRV1 Processor

- **Superscalar execution**: two-way every stage, aligned fetch blocks
- **Out-of-order execution**: IO2L with IQ and ROB
- **Register renaming**: pointer-based scheme with URF and ART
- **Memory disambiguation**: OOO load/store issue with FSB and FLB
- **Branch prediction**: BTB with generalized two-level BHT
- **Speculative execution**: speculative bits with rename table snapshots

### Vector-Vector Add Microbenchmark

<table>
<thead>
<tr>
<th>Microarchitecture</th>
<th>cycles/itr</th>
<th>actual CPI</th>
<th>actual IPC</th>
<th>peak IPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-Order Single-Issue TinyRV1</td>
<td>12</td>
<td>1.33</td>
<td>0.75</td>
<td>1</td>
</tr>
<tr>
<td>In-Order Dual-Issue TinyRV1</td>
<td>10</td>
<td>1.11</td>
<td>0.90</td>
<td>2</td>
</tr>
<tr>
<td>Out-of-Order Dual-Issue TinyRV1</td>
<td>5</td>
<td>0.55</td>
<td>1.80</td>
<td>2</td>
</tr>
</tbody>
</table>