Hardware-Based Speculation

- Execute instructions along predicted execution paths but only commit the results if prediction was correct
  - Finds parallelism among instructions beyond branches
- Instruction commit: allowing an instruction to update the register file/memory when instruction is no longer speculative
- Need an additional piece of hardware to prevent any irrevocable action until an instruction commits
  - I.e. updating state or taking an execution
Reorder Buffer

- Reorder buffer – holds the result of instruction between completion and commit

- Four fields:
  - Instruction type: branch/store/register
  - Destination field: register number
  - Value field: output value
  - Ready field: completed execution?

- Modify reservation stations:
  - Operand source is now reorder buffer instead of functional unit
Reorder Buffer

- **Issue:**
  - Allocate RS and ROB, read available operands

- **Execute:**
  - Begin execution when operand values are available

- **Write result:**
  - Write result and ROB tag on CDB

- **Commit:**
  - When ROB reaches head of ROB, update register
  - When a mispredicted branch reaches head of ROB, discard all entries
Reorder Buffer

- Register values and memory values are not written until an instruction commits

- On misprediction:
  - Speculated entries in ROB are cleared
  - The Spectre security vulnerability arises because some microstate may not be reset.

- Exceptions:
  - Not recognized until it is ready to commit
  - Because a misspeculated instruction should cause exception
Dynamic Scheduling, Multiple Issue, and Speculation

Modern microarchitectures:
- Dynamic scheduling + multiple issue + speculation

Two approaches:
- Assign reservation stations and update pipeline control table in half clock cycles
  - Only supports 2 instructions/clock
- Design logic to handle any possible dependencies between the instructions

Issue logic is the bottleneck in dynamically scheduled superscalars
Multiple Issue

- Examine all the dependencies among the instructions in an issue “bundle”
- If dependences exist in bundle, in parallel encode them assigning reservations stations.
- Also need multiple completion/commit
- To simplify RS allocation:
  - Limit the number of instructions of a given class that can be issued in a “bundle”, i.e. on FP, one integer, one load, one store
Register Renaming

- Register renaming vs. reorder buffers
  - Instead of virtual registers from reservation stations and reorder buffer, create a single register pool (Skylake has 180+168!)
    - Contains visible registers and virtual registers
    - Also used for Simultaneous Multithreading
  - Use hardware-based map to rename registers during issue
  - WAW and WAR hazards are avoided
  - Speculation recovery occurs by copying during commit

- Renaming is useful with multithreading
  - Next lecture: how renaming works.
Integrated Issue and Renaming

- Combining instruction issue with register renaming:
  - Issue logic pre-reserves enough physical registers for the bundle
  - Issue logic finds dependencies within bundle, maps registers as necessary
  - Issue logic finds dependencies between current bundle and already in-flight bundles, maps registers as necessary

<table>
<thead>
<tr>
<th>Instr. #</th>
<th>Instruction</th>
<th>Physical register assigned or destination</th>
<th>Instruction with physical register numbers</th>
<th>Rename map changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>add x1,x2,x3</td>
<td>p32</td>
<td>add p32,p2,p3</td>
<td>x1-&gt; p32</td>
</tr>
<tr>
<td>2</td>
<td>sub x1,x1,x2</td>
<td>p33</td>
<td>sub p33,p32,p2</td>
<td>x1-&gt; p33</td>
</tr>
<tr>
<td>3</td>
<td>add x2,x1,x2</td>
<td>p34</td>
<td>add p34,p33,x2</td>
<td>x2-&gt; p34</td>
</tr>
<tr>
<td>4</td>
<td>sub x1,x3,x2</td>
<td>p35</td>
<td>sub p35,p3,p34</td>
<td>x1-&gt; p35</td>
</tr>
<tr>
<td>5</td>
<td>add x1,x1,x2</td>
<td>p36</td>
<td>add p36,p35,p34</td>
<td>x1-&gt; p36</td>
</tr>
<tr>
<td>6</td>
<td>sub x1,x3,x1</td>
<td>p37</td>
<td>sub p37,p3,p36</td>
<td>x1-&gt; p37</td>
</tr>
</tbody>
</table>
How Much?

- How much to speculate
  - Misspeculation degrades performance and power relative to no speculation
    - May cause additional misses (cache, TLB)
  - Prevent speculative code from causing higher costing misses (e.g. L2)

- Speculating through multiple branches
  - Complicates speculation recovery

- Speculation and energy efficiency
  - Note: speculation is only energy efficient when it improves performance
Skylake Pipeline (Speculative, Multiissue)
MISSPECULATION: WASTED WORK ON THE INTEL CORE i7

Data collected by Professor Lu Peng and student Ying Zhang at LSU.