Register Renaming

6.375 Final Project Presentation
Abdulsami Aldahlawi
Khalid Al-Hawaj
Agenda

- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results
Agenda

- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results
Problem Description

- Recap on scoreboard for data hazard handling ...

- Problem: unnecessary stalls for WAW hazards.
Agenda

- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results
Background

- Compiler Scheduling (static) vs CPU Scheduling (dynamic)
  - ISA limitation
  - More information available in runtime

- Example: \[
  \text{LD } R1, \ 0 (R5) \\
  \text{ADD } R1, \ R2, \ R3
\]

- Using scoreboard, we have to stall …
- Solution: reassign destination registers in runtime
  \[
  \text{LD } PR32, \ 0 (PR13) \\
  \text{ADD } PR54, \ PR22, \ PR33
  \]
Background – How to Rename?

- Tomasulo’s Algorithm (e.g. Intel-P6 style)
  - Use of multiple distributed Reservation Station and a Common Data Bus
  - Issues: - Bus is expensive
  - Infeasible for modern systems

- Sohi’s Method (e.g. MIPS R10K)
  - Separate Logical RegFile (defined by ISA) from Physical RegFile.
  - Maintain Mapping Table and Free List.
Agenda

- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results
Fetch, Decode, Execute, and WriteBack are almost the same as the SMIPS we implemented in the Lab.
Agenda

- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results
Microarchitectural Description

• Rename:
  • Takes ADD R1, R2, R3 as input

  • Consult ‘Mapping Table’ (holds the current mapping of Logical registers to Physical registers) to get physical registers corresponding to sources R2 and R3. (e.g. PR15 and PR18).

  • Assigns a new physical destination register to R1 from the ‘Free List’ and change its previous mapping in ‘Mapping Table’.
Microarchitectural Description

- Re-Order Buffer:
  - Holds renamed instructions and their status.
  - Keeps track of instruction flow in the pipeline.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Busy</th>
<th>Instruction</th>
<th>State</th>
<th>Destination</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>LD F6, 34(R2)</td>
<td>commit</td>
<td>F6</td>
<td>Mem[load1]</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>LD F2, 45(R3)</td>
<td>commit</td>
<td>F2</td>
<td>Mem[load2]</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>MULT F0, F2, F4</td>
<td>Ex8</td>
<td>F0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>SUBD F8, F6, F2</td>
<td>write</td>
<td>F8</td>
<td>F6 + #2</td>
</tr>
<tr>
<td>5</td>
<td>Yes</td>
<td>DIVD F10, F0, F6</td>
<td>issue</td>
<td>F10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Yes</td>
<td>ADDD F6, F8, F2</td>
<td>write</td>
<td>F6</td>
<td>#4 + F2</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Commit:
  - Evict the head of the ROB when its results has been written back
  - Frees the previous mapping of the physical register file.
Agenda

- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results
Simulation and FPGA Results

- Comparison in terms of performance (IPC and MIPS), Maximum achievable frequency, and Device Utilization were made on the following implementations:
  - Baseline: Basic 5-Stage in-order processor (Lab version)
  - Renaming: ROB Size = 2, RegFile Size = 32
  - Renaming: ROB Size = 4, RegFile Size = 64
  - Renaming: ROB Size = 8, RegFile Size = 64
  - Renaming: ROB Size = 16, RegFile Size = 64
Simulation and FPGA Results

Performance

IPC
Baseline
ROB=2
ROB=4
ROB=8
ROB=16

Median multiply qsort towers vvadd

0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Performance in Million Instruction Per Second (MIPS)
Baseline
ROB=2
ROB=4
ROB=8
ROB=16

MIPS
20
30
40
50
60
70
80
90
Simulation and FPGA Results

**Maximum Frequency**

- Frequency (MHz) vs. ROB size (Baseline, ROB=2, ROB=4, ROB=8, ROB=16).
  - Blue bars represent Pre-Routing Freq.
  - Red bars represent Post-Routing Freq.

**Device Utilization**

- Utilization vs. ROB size (Baseline, ROB=2, ROB=4, ROB=8, ROB=16).
  - Blue bars represent Slice Registers.
  - Red bars represent Slice LUTs.
Questions
References

