Housekeeping

- Your goal today
  - transition from sequential to parallel
  - enjoy (you will not be tested on this)

- Notices
  - Midterm 2 on Monday
  - Practice midterm at course Hub
  - HW4 due Friday; HW5 out next Wed
  - Handout #14: HW4 solutions (out on Friday)

- Readings (advanced optional)
  - MIPS R10K Superscalar Microprocessor, Yeager
  - Synthesis Lectures: Processor Microarchitecture: An Implementation Perspective, 2010
Parallelism Defined

- $T_1$ (work measured in time):
  - time to do work with 1 PE
- $T_\infty$ (critical path):
  - time to do work with infinite PEs
  - $T_\infty$ bounded by dataflow dependence
- Average parallelism:
  \[ P_{\text{avg}} = \frac{T_1}{T_\infty} \]
- For a system with $p$ PEs
  \[ T_p \geq \max\{ \frac{T_1}{p}, T_\infty \} \]
- When $P_{\text{avg}} \gg p$
  \[ T_p \approx \frac{T_1}{p}, \text{aka “linear speedup”} \]
Superscalar Speculative Out-of-Order Execution
**ILP: Instruction-Level Parallelism**

- Average **ILP** = \( \frac{T_1}{T_\infty} \)
  
  = no. instruction / no. cyc required

**code1: ILP = 1**

i.e., must execute serially

**code2: ILP = 3**

i.e., can execute at the same time

| code1: | \r1 \leftarrow r2 + 1 | \r3 \leftarrow r1 / 17 | \r4 \leftarrow r0 - r3 |
| code2: | \r1 \leftarrow r2 + 1 | \r3 \leftarrow r9 / 17 | \r4 \leftarrow r0 - r10 |
Exploiting **ILP** for Performance

Scalar in-order pipeline with forwarding

- operation latency (**OL**) = 1 base cycle
- peak **IPC** = 1
- required **ILP** ≥ 1 to avoid stall


**Superpipelined Execution**

\[
\text{OL} = M \text{ minor-cycle}; \text{ same as } 1 \text{ base cycle}
\]

peak \(\text{IPC} = 1\) per minor-cycle

required \(\text{ILP} \geq M\)

Achieving full performance requires always finding \(M\) “independent” instructions in a row

---

18-447-S19-L20-S7, James C. Hoe, CMU/ECE/CALCM, ©2019
Superscalar (Inorder) Execution

\[ \text{OL} = 1 \text{ base cycle} \]
peak \( \text{IPC} = N \)
required \( \text{ILP} \geq N \)

Achieving full performance requires finding \( N \) “independent” instructions on every cycle
Limitations of Inorder Pipeline

- Achieved **IPC** of inorder pipelines degrades rapidly as **NxM** approaches **ILP**
- Despite high peak **IPC** potential, pipeline never full due to frequent dependency stalls!!
Out-of-Order Execution

- **ILP** is scope dependent

ILP = 1

\[
\begin{align*}
\text{r1} & \leftarrow \text{r2} + 1 \\
\text{r3} & \leftarrow \text{r1} / 17 \\
\text{r4} & \leftarrow \text{r0} - \text{r3} \\
\text{r11} & \leftarrow \text{r12} + 1 \\
\text{r13} & \leftarrow \text{r19} / 17 \\
\text{r14} & \leftarrow \text{r0} - \text{r20}
\end{align*}
\]

ILP = 2

Accessing **ILP** = 2 requires (1) larger scheduling window and (2) out-of-order execution
Superscalar Speculative Out-of-Order Execution
Dataflow Execution Ordering

• Maintain a buffer of many pending instructions, a.k.a. reservation stations (RSs)
  – wait for functional unit to be free
  – wait for register RAW hazards to resolve (i.e., required input operands to be produced)

• Issue instructions for execution out-of-order
  – select instructions in RS whose operands are available
  – give preference to older instructions (heuristical)

• A completing instruction frees pending, RAW-dependent instructions to execute

What about WAW and WAR?
Tomasulo’s Algorithm [IBM 360/91, 1967]

- Dispatch an instruction to a RS slot after decode
  - decode received from RF either operand value or placeholder RS-tag
  - mark RF dest with RS-tag of current inst’s RS slot
- A inst in RS can issue when all operand values ready
- Completing instruction, in addition to updating RF dest, broadcast its RS-tag and value to all RS slots
- RS slot holding matching RS-tag placeholder pickup value
Removing False Dependencies

- With out-of-order execution comes WAW and WAR hazards
- Anti and output dependencies are false dependencies on register names rather than data

\[
\begin{align*}
    r_3 & \leftarrow r_1 \text{ op } r_2 \\
    r_5 & \leftarrow r_3 \text{ op } r_4 \\
    r_3 & \leftarrow r_6 \text{ op } r_7
\end{align*}
\]

- With infinite number of registers, anti and output dependencies avoidable by using a new register for each new value
Register Renaming: Example

Original

\[
\begin{align*}
    r1 & \leftarrow r2 / r3 \\
    r4 & \leftarrow r1 * r5 \\
    r1 & \leftarrow r3 + r6 \\
    r3 & \leftarrow r1 - r5
\end{align*}
\]

Renamed

\[
\begin{align*}
    r1 & \leftarrow r2 / r3 \\
    r4 & \leftarrow r1 * r5 \\
    r8 & \leftarrow r3 + r6 \\
    r9 & \leftarrow r8 - r5
\end{align*}
\]
On-the-fly HW Register Renaming

- Maintain mapping from ISA reg. names to physical registers
- When decoding an instruction that updates \( r_x \):
  - allocate unused physical register \( t_y \) to hold inst result
  - set new mapping from \( r_x \) to \( t_y \)
  - younger instructions using \( r_x \) as input finds \( t_y \)
- De-allocate a physical register for reuse when it is \underline{never needed again}?

\[
\begin{align*}
\text{ISA name} & \quad \text{rename} & \quad \text{physical} \\
\text{e.g. } r_{12} & \quad \text{table} & \quad \text{registers} \\
& \quad \text{rename} & \quad (t_0 \ldots t_{63}) \\
& \quad t_{56} \\
\end{align*}
\]

\[
\begin{align*}
\text{r1} & \leftarrow r_2 / r_3 \\
r_4 & \leftarrow r_1 \ast r_5 \\
r_1 & \leftarrow r_3 + r_6
\end{align*}
\]
Superscalar Speculative Out-of-Order Execution
Control Speculation

• Modern CPUs can have over 100 instructions in out-of-order execution scope
  – if 14% of avg. instruction mix is control flow, what is average distance between control flow?
  – instruction fetch must make multiple levels of branch predictions (condition and target) to fetch far ahead of execution and commit

• **BTW**, large OOO is more about cache misses
  – keep working around long cache miss stalls
  – get started on future cache misses as early as possible (to overlap/hide latency of cache misses)
Speculative Out-of-order Execution

• A mispredicted branch after resolution must be rewound and restarted and fast!
• Much trickier than 5-stage pipeline . . . 
  – can rewind to an intermediate speculative state
  – a rewound branch could still be speculative and itself be discarded by another rewind!
  – rewind must reestablish both architectural state (register value) and microarchitecture state (e.g., rename table)
  – rewind/restart must be fast (not infrequent)
• Also need to rewind on exceptions . . . .but easier
Instruction Reorder Buffer (ROB)

• Program-order bookkeeping (circular buffer)
  – instructions enter and leave in program order
  – tracks 10s to 100s of in-flight instructions in different stages of execution

• Dynamic juggling of state and dependency
  – oldest finished instruction “commit” architectural state updates on exit
  – all ROB entries considered “speculative” due to potential for exceptions and mispredictions
In-order vs Speculative State

• In-order state:
  – cumulative architectural effects of all instructions committed in-order so far
  – can never be undone!!

• Speculative state, as viewed by a given inst in ROB
  – in-order state + effects of older insts in ROB
  – effects of some older insts may be pending

• Speculative state effects must be reversible
  – remember both in-order and speculative values for an RF register (may have multiple speculative values)
  – store inst updates memory only at commit time

• Discard younger speculative state to rewind execution to oldest remaining inst in ROB
Supercalar BP Complicated: e.g., 2 way

Tag Table
Branch History Table (BHT)
Branch Target Buffer (BTB)

cache block offset
last inst in cache block?

hit

first?
taken?
predPC

PC+4
PC+8
Trace Caching

- Static 90%
- Dynamic 10%

- Compiler
- Static

- Hardware
- Dynamic

Recall
Superscalar Speculative OOO All Together

Read [Yeager 1996, IEEE Micro] if you are interested
At the 2005 Peak of Superscalar OOO

<table>
<thead>
<tr>
<th></th>
<th>Alpha 21364</th>
<th>AMD Opteron</th>
<th>Intel Xeon</th>
<th>IBM Power5</th>
<th>MIPS R14000</th>
<th>Intel Itanium2</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock (GHz)</td>
<td>1.30</td>
<td>2.4</td>
<td><strong>3.6</strong></td>
<td>1.9</td>
<td>0.6</td>
<td>1.6</td>
</tr>
<tr>
<td>issue rate</td>
<td>4</td>
<td>3 (x86)</td>
<td>3 (rop)</td>
<td><strong>8</strong></td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>pipeline int/fp</td>
<td>7/9</td>
<td>9/11</td>
<td><strong>22/24</strong></td>
<td>12/17</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>inst in flight</td>
<td>80</td>
<td>72 (rop)</td>
<td>126 rop</td>
<td><strong>200</strong></td>
<td>48</td>
<td>inorder</td>
</tr>
<tr>
<td>rename reg</td>
<td>48+41</td>
<td>36+36</td>
<td>128</td>
<td>48/40</td>
<td>32/32</td>
<td><strong>328</strong></td>
</tr>
<tr>
<td>transistor (10^6)</td>
<td>135</td>
<td>106</td>
<td>125</td>
<td>276</td>
<td>7.2</td>
<td><strong>592</strong></td>
</tr>
<tr>
<td>power (W)</td>
<td><strong>155</strong></td>
<td>86</td>
<td>103</td>
<td>120</td>
<td>16</td>
<td>130</td>
</tr>
<tr>
<td>SPECint 2000</td>
<td>904</td>
<td>1,566</td>
<td>1,521</td>
<td>1,398</td>
<td>483</td>
<td><strong>1,590</strong></td>
</tr>
<tr>
<td>SPECfp 2000</td>
<td>1279</td>
<td>1,591</td>
<td>1,504</td>
<td>2,576</td>
<td>499</td>
<td><strong>2,712</strong></td>
</tr>
</tbody>
</table>

Microprocessor Report, December 2004
## At peak minus 5 years

<table>
<thead>
<tr>
<th></th>
<th>Alpha 21264</th>
<th>AMD Athlon</th>
<th>Intel P4</th>
<th>MIPS R12000</th>
<th>IBM Power3</th>
<th>HP PA8600</th>
<th>SUN Ultra3</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock (MHz)</td>
<td>833</td>
<td>1200</td>
<td>1500</td>
<td>400</td>
<td>450</td>
<td>552</td>
<td>900</td>
</tr>
<tr>
<td>issue rate</td>
<td>4</td>
<td>3 (x86)</td>
<td>3 (rop)</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>pipeline int/fp</td>
<td>7/9</td>
<td>9/11</td>
<td>22/24</td>
<td>6</td>
<td>7/8</td>
<td>7/9</td>
<td>14//15</td>
</tr>
<tr>
<td>inst in flight</td>
<td>80</td>
<td>72(rop)</td>
<td>126 rop</td>
<td>48</td>
<td>32</td>
<td>56</td>
<td>inorder</td>
</tr>
<tr>
<td>rename reg</td>
<td>48+41</td>
<td>36+36</td>
<td>128</td>
<td>32+32</td>
<td>16+24</td>
<td>56</td>
<td>inorder</td>
</tr>
<tr>
<td>transistor ($10^6$)</td>
<td>15.4</td>
<td>37</td>
<td>42</td>
<td>7.2</td>
<td>23</td>
<td>130</td>
<td>29</td>
</tr>
<tr>
<td>power (W)</td>
<td>75</td>
<td>76</td>
<td>55</td>
<td>25</td>
<td>36</td>
<td>60</td>
<td>65</td>
</tr>
<tr>
<td>SPECint 2000</td>
<td>518</td>
<td>524</td>
<td>320</td>
<td>286</td>
<td>417</td>
<td>438</td>
<td>438</td>
</tr>
<tr>
<td>SPECfp 2000</td>
<td><strong>590</strong></td>
<td>304</td>
<td>549</td>
<td>319</td>
<td>356</td>
<td>400</td>
<td>427</td>
</tr>
</tbody>
</table>

**Note:** The table compares the performance metrics of different microprocessors at peak minus 5 years. The metrics include clock speed, issue rate, pipeline, instruction in flight, rename registers, transistor count, power consumption, and SPEC scores.

*Microprocessor Report, December 2000*
Performance (In)efficiency

- To hit “expected” performance target
  - push frequency harder by deepening pipelines
  - used the 2x transistors to build more complicated microarchitectures so fast/deep pipelines don’t stall (i.e., caches, BP, superscalar, out-of-order)

- The consequence of performance inefficiency is

[limit of economical cooling [ITRS]]

Recall

2005, Intel P4 Tehas 150W

[Borkar, IEEE Micro, July 1999]
Efficiency of Parallel Processing

Better to replace 1 of this by 2 of these; Or N of these

Power $\approx$ Perf$^{1.75}$

[Energy per Instruction Trends in Intel® Microprocessors, Grochowski et al., 2006]
Moore’s Law Era

Era of Killer Micros:
increasing transistor count, freq, perf, and power

Multicore Era:
growing transistor count & aggr. perf;
flattened power & seq. perf; lowering freq.
## At peak plus 1 year

<table>
<thead>
<tr>
<th></th>
<th>AMD 285</th>
<th>Intel 5160</th>
<th>Intel 965</th>
<th>Intel Itanium2</th>
<th>IBM P5+</th>
<th>MIPS R16000</th>
<th>SUN Ultra4</th>
</tr>
</thead>
<tbody>
<tr>
<td>cores/threads</td>
<td>2x1</td>
<td>2x2</td>
<td>2x2</td>
<td>2x2</td>
<td>2x2</td>
<td>1x1</td>
<td>2x1</td>
</tr>
<tr>
<td>clock (GHz)</td>
<td>2.6</td>
<td>3.03</td>
<td>3.73</td>
<td>1.6</td>
<td>2.3</td>
<td>0.7</td>
<td>1.8</td>
</tr>
<tr>
<td>issue rate</td>
<td>3 (x86)</td>
<td>4 (rop)</td>
<td>3 (rop)</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>pipeline depth</td>
<td>11</td>
<td>14</td>
<td>31</td>
<td>8</td>
<td>17</td>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>inst in flight</td>
<td>72(rop)</td>
<td>96(rop)</td>
<td>126(rop)</td>
<td>inorder</td>
<td>200</td>
<td>48</td>
<td>inorder</td>
</tr>
<tr>
<td>on-chip$ (MB)</td>
<td>2x1</td>
<td>4</td>
<td>2x2</td>
<td>2x13</td>
<td>1.9</td>
<td>0.064</td>
<td>2</td>
</tr>
<tr>
<td>transistor (10^6)</td>
<td>233</td>
<td>291</td>
<td>376</td>
<td>1700</td>
<td>276</td>
<td>7.2</td>
<td>295</td>
</tr>
<tr>
<td>power (W)</td>
<td>95</td>
<td>80</td>
<td>130</td>
<td>104</td>
<td>100</td>
<td>17</td>
<td>90</td>
</tr>
<tr>
<td>SPECint 2000 per core</td>
<td>1942 (1556*)</td>
<td>1870</td>
<td>1474</td>
<td>1820</td>
<td>560</td>
<td>1300</td>
<td></td>
</tr>
<tr>
<td>SPECfp 2000 per core</td>
<td>2260 (1694+)</td>
<td>2232</td>
<td>3017</td>
<td>3369</td>
<td>580</td>
<td>1800</td>
<td></td>
</tr>
</tbody>
</table>

*3086/+2884 according to www.spec.org

Microprocessor Report, Aug 2006
At peak plus 3 years

<table>
<thead>
<tr>
<th></th>
<th>AMD Opteron 8360SE</th>
<th>Intel Xeon X7460</th>
<th>Intel Itanium 9050</th>
<th>IBM P5</th>
<th>IBM P6</th>
<th>Fijitsu SPARC 7</th>
<th>SUN T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>cores/threads</td>
<td>4x1</td>
<td><strong>6x1</strong></td>
<td>2x2</td>
<td>2x2</td>
<td>2x2</td>
<td>4x2</td>
<td><strong>8x8</strong></td>
</tr>
<tr>
<td>clock (GHz)</td>
<td>2.5</td>
<td>2.67</td>
<td>1.60</td>
<td>2.2</td>
<td><strong>5</strong></td>
<td>2.52</td>
<td>1.8</td>
</tr>
<tr>
<td>issue rate</td>
<td>3 (x86)</td>
<td>4 (rop)</td>
<td>6</td>
<td>5</td>
<td><strong>7</strong></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>pipeline depth</td>
<td>12/17</td>
<td>14</td>
<td>8</td>
<td>15</td>
<td>13</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>out-of-order</td>
<td>72(rop)</td>
<td>96(rop)</td>
<td><strong>inorder</strong></td>
<td><strong>200</strong></td>
<td>limited</td>
<td>64</td>
<td>inorder</td>
</tr>
<tr>
<td>on-chip$ (MB)</td>
<td>2+2</td>
<td><strong>9+16</strong></td>
<td>1+12</td>
<td>1.92</td>
<td>8</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>transistor (10^6)</td>
<td>463</td>
<td><strong>1900</strong></td>
<td>1720</td>
<td>276</td>
<td>790</td>
<td>600</td>
<td>503</td>
</tr>
<tr>
<td>power max(W)</td>
<td>105</td>
<td>130</td>
<td>104</td>
<td>100</td>
<td>&gt;100</td>
<td><strong>135</strong></td>
<td>95</td>
</tr>
<tr>
<td>SPECint 2006</td>
<td>14.4/170</td>
<td><strong>22/274</strong></td>
<td>14.5/1534</td>
<td>10.5/197</td>
<td>15.8/1837</td>
<td><strong>10.5/2088</strong></td>
<td>--/142</td>
</tr>
<tr>
<td>SPECfp 2006</td>
<td>18.5/156</td>
<td><strong>22/142</strong></td>
<td>17.3/1671</td>
<td>12.9/229</td>
<td>20.1/1822</td>
<td><strong>25.0/1861</strong></td>
<td>--/111</td>
</tr>
</tbody>
</table>

Microprocessor Report, Oct 2008
On to Mainstream Parallelism in Multicores and Manycores

Remember, we got here because we need to compute faster while using less energy per operation.