Superscalar Organization

Nima Honarmand
Instruction-Level Parallelism (ILP)

- Recall: “Parallelism is the number of independent tasks available”
- ILP is a measure of inter-dependencies between insns.
- Average ILP = num. instruction / num. cyc required in an “ideal machine”

\[
\text{code1: } \quad \text{ILP} = 1 \\
\quad \text{i.e. must execute serially}
\]

\[
\text{code2: } \quad \text{ILP} = 3 \\
\quad \text{i.e. can execute at the same time}
\]

\[
\begin{align*}
\text{code1:} & \quad r1 & \leftarrow & r2 + 1 \\
& \quad r3 & \leftarrow & r1 / 17 \\
& \quad r4 & \leftarrow & r0 - r3 \\
\text{code2:} & \quad r1 & \leftarrow & r2 + 1 \\
& \quad r3 & \leftarrow & r9 / 17 \\
& \quad r4 & \leftarrow & r0 - r10
\end{align*}
\]
ILP != IPC

- **ILP** usually assumes
  - Infinite resources
  - Perfect fetch
  - Unit-latency for all instructions

- **ILP** is a property of the program dataflow

- **IPC** is the “real” observed metric
  - How many insns. are executed per cycle

- **ILP** is an upper-bound on the attainable **IPC**
  - Specific to a particular program
## Purported Limits on ILP

<table>
<thead>
<tr>
<th>Study</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weiss and Smith [1984]</td>
<td>1.58</td>
</tr>
<tr>
<td>Sohi and Vajapeyam [1987]</td>
<td>1.81</td>
</tr>
<tr>
<td>Tjaden and Flynn [1970]</td>
<td>1.86</td>
</tr>
<tr>
<td>Tjaden and Flynn [1973]</td>
<td>1.96</td>
</tr>
<tr>
<td>Uht [1986]</td>
<td>2.00</td>
</tr>
<tr>
<td>Smith et al. [1989]</td>
<td>2.00</td>
</tr>
<tr>
<td>Jouppi and Wall [1988]</td>
<td>2.40</td>
</tr>
<tr>
<td>Johnson [1991]</td>
<td>2.50</td>
</tr>
<tr>
<td>Acosta et al. [1986]</td>
<td>2.79</td>
</tr>
<tr>
<td>Wedig [1982]</td>
<td>3.00</td>
</tr>
<tr>
<td>Butler et al. [1991]</td>
<td>5.8</td>
</tr>
<tr>
<td>Melvin and Patt [1991]</td>
<td>6</td>
</tr>
<tr>
<td>Wall [1991]</td>
<td>7</td>
</tr>
<tr>
<td>Kuck et al. [1972]</td>
<td>8</td>
</tr>
<tr>
<td>Riseman and Foster [1972]</td>
<td>51</td>
</tr>
<tr>
<td>Nicolau and Fisher [1984]</td>
<td>90</td>
</tr>
</tbody>
</table>
ILP Limits of Scalar Pipelines (1)

• Scalar upper bound on throughput
  – Limited to CPI $\geq 1$
  – Solution: superscalar pipelines with multiple insns at each stage

Pentium Pipeline
ILP Limits of Scalar Pipelines (2)

- Inefficient unified pipeline
  - Lower resource utilization and longer instruction latency
  - Solution: diversified pipelines
ILP Limits of Scalar Pipelines (3)

• Rigid pipeline stall policy
  – A stalled instruction stalls all newer instructions
  – Solution 1: out-of-order execution
ILP Limits of Scalar Pipelines (3)

• Rigid pipeline stall policy
  – A stalled instruction stalls all newer instructions

  – Solution 1: **out-of-order** execution
  – Solution 2: **inter-stage buffers**

In Program Order

Out of Order

In Program Order

Fetch

Decode

Dispatch

Issuing Buffer

Execute

Completion Buffer

Complete

Store Buffer

Retire
ILP Limits of Scalar Pipelines (4)

• Instruction dependencies limit parallelism
  – Frequent stalls due to data and control dependencies
  – Solution 1: renaming – for WAR and WAW register dependences
  – Solution 2: speculation – for control dependences and memory dependences
ILP Limits of Scalar Pipelines (Summary)

1. Scalar upper bound on throughput
   - Limited to CPI >= 1
   - Solution: superscalar pipelines with multiple insns at each stage

2. Inefficient unified pipeline
   - Lower resource utilization and longer instruction latency
   - Solution: diversified pipelines

3. Rigid pipeline stall policy
   - A stalled instruction stalls all newer instructions
   - Solution: out-of-order execution and inter-stage buffers

4. Instruction dependencies limit parallelism
   - Frequent stalls due to data and control dependencies
   - Solutions: renaming and speculation

State of the art: Out-of-Order Superscalar Pipelines
Overall Picture

• Fetch issues:
  – Fetch multiple insns
  – Branches
  – Branch target mis-alignment

• Decode issues:
  – Identify insns
  – Find dependences

• Execution issues:
  – Dispatch insns
  – Resolve dependences
  – Bypass networks
  – Multiple outstanding memory accesses

• Completion issues:
  – Out-of-order completion
  – Speculative instructions
  – Precise exceptions

State of the art: Out-of-Order Superscalar Pipelines