Superscalar Organization

Nima Honarmand
Review: Instruction-Level Parallelism (ILP)

• “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”

code1:

\[ \text{ILP} = 1 \]

\[ \text{i.e. must execute serially} \]

code2:

\[ \text{ILP} = 3 \]

\[ \text{i.e. can execute at the same time} \]

code1: \[
\begin{align*}
    r1 & \leftarrow r2 + 1 \\
    r3 & \leftarrow r1 / 17 \\
    r4 & \leftarrow r0 - r3
\end{align*}
\]

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

- **ILP** usually assumes
  - Infinite resources
  - Perfect fetch and branch prediction
  - 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>Author(s) and Year</th>
<th>Limit</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 IPC <= 1
  - Solution: superscalar pipelines with multiple insns at each stage

Pentium Pipeline

Prefetch -> Decode1 -> Decode2 -> Execute -> Writeback

U-pipe -> V-pipe
ILP Limits of Scalar Pipelines (2)

- **Unified pipeline**: a pipeline where all instructions go through the same stages
  - Like our 5-stage pipeline

- Unified pipelines are inefficient
  - 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
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
Summary : ILP Limits of Scalar Pipelines

1) Scalar upper bound on throughput
   - Limited to IPC \( \leq 1 \)
   - Solution: \textit{superscalar} pipelines with multiple insns at each stage

2) Inefficient unified pipeline
   - Lower resource utilization and longer instruction latency
   - Solution: \textit{diversified} pipelines

3) Rigid pipeline stall policy
   - A stalled instruction stalls all newer instructions
   - Solution: \textit{out-of-order} execution and \textit{inter-stage buffers}

4) Instruction dependencies limit parallelism
   - Frequent stalls due to data and control dependencies
   - Solutions: \textit{renaming} and \textit{speculation}

State of the art: \textit{Out-of-Order Superscalar Speculative Pipelines}
Superscalar Pipelines: Overall Picture

- **Fetch issues:**
  - Fetch multiple insns
  - Branches and speculation

- **Decode issues:**
  - Identify insns
  - Find dependences

- **Execution issues:**
  - Dispatch insns
  - Resolve dependences
  - Forwarding networks
  - Multiple outstanding memory accesses

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

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