Plan for Today

• Announcements
  – Final project requirements are on-line

• Questions

• Today’s discussion: Limits of ILP
  – First: Ideal, unrealistic case
  – Then consider:
    • Limit on window size
    • Limit on control speculation
    • Limit on renaming registers
    • Limit on memory disambiguation
Project Logistics

• Groups of three

• Three major deadlines
  – Design due Oct 19
  – Project Demo Nov 23
  – Report due Dec 7

• This week: PM2 assigned

<table>
<thead>
<tr>
<th>Task</th>
<th>Due</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Design</td>
<td>Oct 19</td>
<td>Project Milestone 1</td>
</tr>
<tr>
<td>Detailed Design &amp; VHDL</td>
<td>Nov 2</td>
<td>Project Milestone 2</td>
</tr>
<tr>
<td>Presentation</td>
<td>Nov 4</td>
<td>Design Presentations</td>
</tr>
<tr>
<td>Demo working VHDL model</td>
<td>Nov 23</td>
<td>Project Demo</td>
</tr>
<tr>
<td>Final report</td>
<td>Dec 7</td>
<td>Final Report</td>
</tr>
</tbody>
</table>
ILP Limit Study

- ILP is a *program property*
- The processors we have considered all exploit ILP to improve performance
  - Simple pipelines
  - Score-boarding, Tomasulo
  - Speculation (e.g., Tomasulo + ROB)
- Designer’s question:
  - How much ILP exists? (And what is needed to exploit it?)
Methodology

1. Assume an ideal (and impractical) processor
2. Add limitations one at a time to measure individual impact
3. Consider ILP limits on a hypothetically practical processor

Analysis performed on benchmark programs with varying characteristics
Ideal Processor

Remove all architectural limits on ILP

1. Register renaming
   - Unlimited number; no WAW or WAR hazards
2. Branch prediction
   - Conditional branches always predicted correctly
3. Jump prediction
   - All branch targets are guessed correctly
4. Perfect memory disambiguation
   - All memory addresses known exactly and loads can be moved before a store provided that the addresses are not the same
5. Infinite resources (e.g., functional units, perfect caches, memory bandwidth)
Ideal Processor Attributes

• Only true data dependences limit parallelism
  – 2 and 3 eliminate control dependences
  – 1 and 4 leave only true data dependences

• All functional unit latencies: one clock cycle
  – Any dynamic instruction can execute in the cycle
    after its predecessor executes
Experimental Method

- Programs compiled and optimized w/ standard MIPS optimizing compiler
- The programs were instrumented to produce a trace of instruction and data references over the entire execution
- Each instruction was subsequently re-scheduled as early as the true data dependencies would allow
  - No control dependence
- New IPC was calculated
### Benchmark from SPEC-92

<table>
<thead>
<tr>
<th>Program</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc</td>
<td>INT</td>
<td>Gnu C compiler</td>
</tr>
<tr>
<td>espresso</td>
<td>INT</td>
<td>Boolean expression minimizer</td>
</tr>
<tr>
<td>li</td>
<td>INT</td>
<td>Lisp interpreter</td>
</tr>
<tr>
<td>fpppp</td>
<td>FP</td>
<td>Quantum chemistry problem</td>
</tr>
<tr>
<td>doduc</td>
<td>FP</td>
<td>Monte-carlo simulation</td>
</tr>
<tr>
<td>tomcatv</td>
<td>FP</td>
<td>Generation of 2D coordinate system around a geometric region</td>
</tr>
</tbody>
</table>
ILP on Ideal Processor

SPEC benchmarks

- gcc: 55
- espresso: 63
- li: 18
- fpppp: 75
- doduc: 119
- tomcatv: 150

Instruction issues per cycle

© 2003 Elsevier Science (USA). All rights reserved.
Achieving Ideal Speculation

• Look $n$ instructions ahead with perfect branch prediction to find independent instructions
• Rename all registers along the way (avoid WAR and WAW hazards)
• Resolve data dependencies
• Resolve memory dependences
• Provide $n$ functional units
Limited Instruction Window

• Limit window size to $n$ (no longer arbitrary)
  – Window = number of instructions candidate for concurrent execution in a cycle

• Window size determines
  – Instruction storage needed within the pipeline
  – Number of *operand comparisons* needed for dependence checking, $O(n^2)$
  – Maximum *issue rate* (although this is usually smaller anyway)

• Additional factors limit instruction issue
  – Register file ports
  – Reservation stations of unit queues
  – Number of functional units
  – Restrictions on branch/load/store issues per clock
  – Commit limitations
Effect of Reduced Window Size

Instruction issues per cycle vs. Window size for different applications:
- gcc
- fpppp
- espresso
- doduc
- li
- tomcatv

© 2003 Elsevier Science (USA). All rights reserved.
Effect of Reduced Window Size
Effect of Reduced Window Size

• IPC reduced considerably
• Highly dependent on loop-level parallelism
  – Instructions that can execute in parallel across loop iterations cannot be found with small window sizes without compiler help
• From now on,
  – 2k window size
  – Maximum of 64 instructions issued per cycle
Realistic Control Speculation

• So far, all branch outcomes are known before the first instruction executes
  – This is difficult to achieve in hardware or software
• Consider 5 alternatives
  – Perfect
  – Tournament predictor
    • Correlating and non-correlating 2-bit predictors plus selector
  – Standard (non-correlating) 2-bit predictor
  – Static (profile-based)
  – None (parallelism limited to within current basic block)
• No penalty for mispredicted branches except for unseen parallelism
Branch Predictor Accuracy

- **tomcatv**: 99% (Profile-based), 99% (2-bit counter), 100% (Tournament)
- **doduc**: 84% (Profile-based), 95% (2-bit counter), 97% (Tournament)
- **fppp**: 82% (Profile-based), 86% (2-bit counter), 98% (Tournament)
- **li**: 77% (Profile-based), 88% (2-bit counter), 98% (Tournament)
- **espresso**: 82% (Profile-based), 86% (2-bit counter), 96% (Tournament)
- **gcc**: 70% (Profile-based), 88% (2-bit counter), 94% (Tournament)

Branch-prediction accuracy

© 2003 Elsevier Science (USA). All rights reserved.
Effect of Realistic Branch Prediction
Effect of Realistic Branch Prediction

- gcc
- espresso
- li
- Benchmarks
- fppp
- doudc
- tomcatv

© 2003 Elsevier Science (USA). All rights reserved.
Realistic Branch Prediction

- Accurate prediction is critical to finding ILP
- Independent instructions are separated by many branches
  - INT programs
- From now on, assume tournament predictor
  - 8K entries
  - 2K-entry jump and return predictors
Finite Renaming Registers

• So far, we have had unlimited physical registers
  – No name dependences
• Alpha 21264 has the most: 41 g-p and f-p renaming registers
Effect of Finite Renaming Registers
Effect of Finite Renaming Registers
Finite Registers

• Has impact, but overshadowed by window size limitation and branch prediction for integer applications

• From now on, assume 256 g-p and 256 f-p registers
Realistic Memory Disambiguation

• So far, memory alias analysis has been perfect
• Consider 3 models
  – Global/stack perfect: idealized static program analysis (heap references are assumed to conflict)
  – Inspection: a simpler, realizable compiler technique limited to inspecting base registers and constant offsets
  – None: all references are assumed to conflict
Effect of Realistic Alias Analysis

![Graph showing the effect of realistic alias analysis on instruction issues per cycle. The x-axis represents different levels of alias analysis technique (Perfect, Global/stack perfect, Inspection, None), and the y-axis represents instruction issues per cycle. Different data points are marked for various alias analysis tools like gcc, fppp, espresso, doduc, li, and tomcatv.](image-url)
Effect of Realistic Alias Analysis

- gcc
- espresso
- li
- fpppp
- doduc
- tomcatv

Benchmark

Instruction issues per cycle

© 2003 Elsevier Science (USA). All rights reserved.
Realizable Alias Analysis

• Alias analysis has enormous impact on parallelism
• Dynamic memory disambiguation constrained by
  – Each load address must be compared with all \textit{in-flight} stores
  – The number of references that can be analyzed each clock cycle
  – The amount of load/store buffering determines how far a load/store instruction can be moved
ILP for Realizable Processors

• Given sophisticated hardware support in the near future, assume
  – 64 issues per clock with no restrictions
  – Tournament predictor with 1K entries and 16-entry return predictor
  – Perfect dynamic memory disambiguation (OK for small window sizes or with memory dependence predictor)
  – 64 additional g-p and f-p renaming registers
ILP on Hypothetical Processor

© 2003 Elsevier Science (USA). All rights reserved.
Realizable Processor

• Ambitious/impractical hardware assumptions
  – Unrestricted issue (particularly memory ops)
  – Single cycle operations
  – Perfect caches

• Other directions
  – Data value prediction and speculation
    • Address value prediction and speculation
  – Speculation on multiple paths
# Recent High-Performance Processors

<table>
<thead>
<tr>
<th>MicroProcessor</th>
<th>Date</th>
<th>MHz</th>
<th>Window /ROB size</th>
<th>Renaming registers</th>
<th>Issue Rate: max/mem/int/FP/branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPS R14000</td>
<td>2000</td>
<td>400</td>
<td>48</td>
<td>32/32</td>
<td>4/1/2/2/1</td>
</tr>
<tr>
<td>UltraSPARC III</td>
<td>2001</td>
<td>900</td>
<td>N.A.</td>
<td>None</td>
<td>4/1/4/3/1</td>
</tr>
<tr>
<td>Pentium III</td>
<td>2000</td>
<td>1000</td>
<td>40</td>
<td>Total: 40</td>
<td>3/2/2/1/1</td>
</tr>
<tr>
<td>Pentium 4</td>
<td>2001</td>
<td>1700</td>
<td>126</td>
<td>Total: 128</td>
<td>3/2/3/2/1</td>
</tr>
<tr>
<td>HP PA-8600</td>
<td>2001</td>
<td>552</td>
<td>56</td>
<td>Total: 56</td>
<td>4/2/2/2/1</td>
</tr>
<tr>
<td>ALPHA 21264B</td>
<td>2001</td>
<td>833</td>
<td>80</td>
<td>41/41</td>
<td>4/2/4/2/1</td>
</tr>
<tr>
<td>PowerPC 7400 (G4)</td>
<td>2000</td>
<td>450</td>
<td>5</td>
<td>6/6</td>
<td>3/1/2/1/1</td>
</tr>
<tr>
<td>AMD Athlon</td>
<td>2001</td>
<td>1330</td>
<td>72</td>
<td>36/36</td>
<td>3/2/3/3/1</td>
</tr>
<tr>
<td>IBM Power3-II</td>
<td>2000</td>
<td>450</td>
<td>32</td>
<td>16/24</td>
<td>4/2/2/2/2</td>
</tr>
<tr>
<td>Intel Nehalem</td>
<td>2009</td>
<td>~2000</td>
<td>128</td>
<td>Total: 164</td>
<td>4/2/3/3/1</td>
</tr>
</tbody>
</table>
ILP versus TLP


One 6-way processor

Four 2-way processors
Assignment

• Readings
  – For Monday
    • Commentary: *The Superblock: An Effective Technique for VLIW and Superscalar Compilation*
  – For Wednesday
    • H&P: Read C.1-C.2, 5.1