LIMITS OF ILP

B649
Parallel Architectures and Programming
A Perfect Processor

• Register renaming
  ✫ infinite number of registers
  ✫ hence, avoids all WAW and WAR hazards

• Branch prediction
  ✫ perfect prediction

• Jump prediction
  ✫ perfect jump and return prediction

• Memory address alias analysis
  ✫ addresses perfectly disambiguated

• Cache
  ✫ no misses
Available ILP on a Perfect Processor

SPEC benchmarks

- gcc: 55
- espresso: 63
- li: 18
- fppp: 75
- dodd: 119
- tomcatv: 150

Instruction issues per cycle

© 2007 Elsevier, Inc. All rights reserved.
What Needs to Happen?

• Look arbitrarily ahead
• Rename all registers
• Rename within an issue packet to avoid dependences
• Handle memory dependences
• Provide sufficient replicated functional units
What Needs to Happen?

• Look arbitrarily ahead
• Rename all registers
• Rename within an issue packet to avoid dependences
• Handle memory dependences
• Provide sufficient replicated functional units

Comparisons needed:
\[ 2n-2 + 2n-4 + \ldots + 2 = 2 \sum_{i=1}^{i=n-1} i = 2((n-1)n/2) = n^2-n \]
What Needs to Happen?

- Look arbitrarily ahead
- Rename all registers
- Rename within an issue packet to avoid dependences
- Handle memory dependences
- Provide sufficient replicated functional units

Comparisons needed:

\[ 2n-2 + 2n-4 + \ldots + 2 = 2 \sum_{i=1}^{i=n-1} i = 2((n-1)n/2) = n^2 - n \]
What Needs to Happen?

• Look arbitrarily ahead
• Rename all registers
• Rename within an issue packet to avoid dependences
• Handle memory dependences
• Provide sufficient replicated functional units

Comparisons needed:
\[ 2n - 2 + 2n - 4 + \ldots + 2 = 2 \sum_{i=1}^{i=n-1} i = 2((n-1)n/2) = n^2 - n \]
What Needs to Happen?

• Look arbitrarily ahead
• Rename all registers
• Rename within an issue packet to avoid dependences
• Handle memory dependences
• Provide sufficient replicated functional units

Comparisons needed:
\[ 2n - 2 + 2n - 4 + \ldots + 2 = 2 \sum_{i=1}^{i=n-1} i = 2((n-1)n/2) = n^2 - n \]
What Needs to Happen?

- Look arbitrarily ahead
- Rename all registers
- Rename within an issue packet to avoid dependences
- Handle memory dependences
- Provide sufficient replicated functional units

Comparisons needed:
\[2n-2 + 2n-4 + ... + 2 = 2 \sum_{i=1}^{i=n-1} i = 2((n-1)n/2) = n^2 - n\]
Effect of Window Size

Benchmarks:
- gcc: 10, 10, 8
- espresso: 15, 13, 8
- li: 18, 15, 12, 11
- fpppp: 14, 15
- doduc: 16, 15
- tomcatv: 14, 15

Instruction issues per cycle:
- Infinite
- 2K
- 512
- 128
- 32
Effect of Window Size

Recall techniques of Appendix G
Realistic Branch and Jump Prediction
(2K window, 64 issue)

Benchmarks:
- gcc
- espresso
- li
- fpppp
- doduc
- tomcatv

Instruction issues per cycle

Branch predictor:
- Perfect
- Tournament predictor
- Standard 2-bit
- Profile-based
- None

© 2007 Elsevier, Inc. All rights reserved.
Branch Misprediction Rate

(2K window, 64 issue)
Limited Number of Registers

(2K window, 64 issue, 150K bits tournament predictor)
Imperfect Alias Analysis

(2K window, 64 issue, 150K bits tournament predictor, 256 integer + 256 FP registers)
Realizable Processor

• 64 issues per cycle
  ★ no issue restriction
  ★ 10 times widest available in 2005
• Tournament branch predictor with 1K entries
  ★ comparable to best in 2005, not a primary bottleneck
• Perfect memory reference disambiguation
  ★ may be practical for small window sizes
• Register renaming with 64 integer and 64 FP registers
  ★ comparable to IBM Power5
Performance on a Realizable Processor

Benchmarks

Instruction issues per cycle

Window size
- Infinite
- 256
- 128
- 64
- 32

gcc
- 10
- 10
- 8

espresso
- 15
- 10
- 13

li
- 12
- 12
- 11
- 9

Bppo
- 14
- 22
- 35
- 47

doduc
- 17
- 16
- 15
- 9

tomcafv
- 14
- 22
- 34
- 45
- 56

© 2007 Elsevier Inc. All rights reserved
Beyond These Limits

- WAW and WAR hazards through memory
  - stack frames (reuse of stack area)
- “Unnecessary” dependences
  - recurrences
  - code generation conventions (e.g., loop index, use of specific registers)
    - can we eliminate some of these?
- Data flow limits
  - value prediction (not very successful, so far)
CROSS-CUTTING ISSUES
Role of Software vs Hardware

• Memory reference disambiguation
  ★ alias analysis

• Speculation
  ★ hardware-based better with unpredictable branches
  ★ precise exceptions: both hardware and software
  ★ bookkeeping code not needed in hardware-based approach

• Scheduling
  ★ compiler has a bigger picture

• Architecture independence
  ★ hardware-based approach might be better (?)
SIMULTANEOUS MULTITHREADING
Why Multithreading?

- Limitations of ILP
  - inherent limitations, in availability of instruction-level parallelism
  - hardware limitations
  - hardware complexities limit further improvements
- Two ways to multithread
  - coarse-grained
  - fine-grained
Why Multithreading?

• Limitations of ILP
  ★ inherent limitations, in availability of instruction-level parallelism
  ★ hardware limitations
  ★ hardware complexities limit further improvements

• Two ways to multithread
  ★ coarse-grained
  ★ fine-grained

Beware: textbook uses multithreading and multiprocessing interchangeably
Simultaneous Multithreading

![Diagram showing issue slots and time for Superscalar, Coarse MT, Fine MT, and SMT]

- **Superscalar**
- **Coarse MT**
- **Fine MT**
- **SMT**
Design and Challenges

- Build on top of existing hardware
  - need per-thread register renaming tables
  - separate PCs
  - ability to commit from multiple threads
- Throughput vs per-thread performance
  - preferred thread
  - fetching far ahead for single thread vs throughput
- Large register file
- Maintaining clock cycle speed
- Handling cache and TLB misses
IBM Power5 Approach

• Increase the associativity of L1 instruction cache and instruction address translation buffers
• Added per-thread load and store queues
• Increased the sizes of L2 and L3 caches
• Added separate instruction prefetch and buffering
• Increased the number of virtual registers from 152 to 240
• Increased the size of several issue queues
Potential Performance Improvement
## Broad Comparison

<table>
<thead>
<tr>
<th>Processor</th>
<th>Microarchitecture</th>
<th>Fetch/issue/execute</th>
<th>Func. units</th>
<th>Clock rate (GHz)</th>
<th>Transistors and die size</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Pentium 4 Extreme</td>
<td>Speculative dynamically scheduled; deeply pipelined; SMT</td>
<td>3/3/4</td>
<td>7 int. 1 FP</td>
<td>3.8</td>
<td>125M 122 mm²</td>
<td>115 W</td>
</tr>
<tr>
<td>AMD Athlon 64 FX-57</td>
<td>Speculative dynamically scheduled</td>
<td>3/3/4</td>
<td>6 int. 3 FP</td>
<td>2.8</td>
<td>114M 115 mm²</td>
<td>104 W</td>
</tr>
<tr>
<td>IBM Power5 1 processor</td>
<td>Speculative dynamically scheduled; SMT; two CPU cores/chip</td>
<td>8/4/8</td>
<td>6 int. 2 FP</td>
<td>1.9</td>
<td>200M 300 mm² (estimated)</td>
<td>80 W (estimated)</td>
</tr>
<tr>
<td>Intel Itanium 2</td>
<td>EPIC style; primarily statically scheduled</td>
<td>6/5/11</td>
<td>9 int. 2 FP</td>
<td>1.6</td>
<td>592M 423 mm²</td>
<td>130 W</td>
</tr>
</tbody>
</table>
What Limits These Processors?

- ILP limitations as already seen
- Hardware complexity increases rapidly
- Power
  - dynamic power dominates
  - multiple issues required much more hardware, increasing power cost
  - growing gap between peak and sustained performance
  - speculation is inherently energy inefficient
NEXT: MULTICORE