Chapter 3 – Instruction-Level Parallelism and its Exploitation (Part 4)

ILP vs. Parallel Computers
Dynamic Scheduling (Section 3.4, 3.5)
Dynamic Branch Prediction (Section 3.3)
Hardware Speculation and Precise Interrupts (Section 3.6)
Multiple Issue (Section 3.7)
Static Techniques (Section 3.2, Appendix H)
Limitations of ILP (Section 3.10)
Multithreading (Section 3.12)
Putting it Together (Mini-projects)
How much can ILP buy us?

Limits studies make optimistic assumptions to find the limit for ILP

But may miss impact of compiler, future advances

A highly optimistic study [Wall’93]

- Infinite number of physical registers (no register WAW, WAR)
- Infinite number of in-flight instructions
- Perfect branch prediction
- Perfect memory address alias analysis
- Single cycle FU
- Single cycle memory (perfect caches)
Limits of ILP (contd.)

(This and next four figures are from an old edition of the book)

<table>
<thead>
<tr>
<th>SPEC benchmarks</th>
<th>Instruction issues per cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc</td>
<td>55</td>
</tr>
<tr>
<td>espresso</td>
<td>63</td>
</tr>
<tr>
<td>li</td>
<td>18</td>
</tr>
<tr>
<td>fppp</td>
<td>75</td>
</tr>
<tr>
<td>doduc</td>
<td>119</td>
</tr>
<tr>
<td>tomcatv</td>
<td>150</td>
</tr>
</tbody>
</table>

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

* This figure has been taken from Computer Architecture, A Quantitative Approach, 3rd Edition Copyright 2003 by Elsevier Inc. All rights reserved. It has been used with permission by Elsevier Inc.
Figure 3.35 ILP available in a perfect processor for six of the SPEC92 benchmarks.
The first three programs are integer programs, and the last three are floating-point programs.
The floating-point programs are loop-intensive and have large amounts of loop-level parallelism.

© 2003 Elsevier Science (USA). All rights reserved.
Limits of ILP – Impact of Optimistic Assumptions

Limiting Instruction window size

Finding dependences among n instr requires n^2 comparisons

2000 instructions implies 4 million comparisons!

Following use 2K window and 64 issue limit

* This figure has been taken from Computer Architecture, A Quantitative Approach, 3rd Edition Copyright 2003 by Elsevier Inc. All rights reserved. It has been used with permission by Elsevier Inc.

© 2003 Elsevier Science (USA). All rights reserved.
Limiting Instruction window size

Finding dependences among n instr requires n^2 comparisons

2000 instructions implies 4 million comparisons!

Following use 2K window and 64 issue limit

Figure 3.36 The effects of reducing the size of the window. The window is the group of instructions from which an instruction an execute. The start of the window is the earliest uncompleted instruction (remember that instructions complete in one cycle), and the last instruction in the window is determined by the window size. The instructions in the window are obtained by perfectly predicting branches and selecting instructions until the window is full.

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

* This figure has been taken from Computer Architecture, A Quantitative Approach, 3rd Edition Copyright 2003 by Elsevier Inc. All rights reserved. It has been used with permission by Elsevier Inc.
Limits of ILP – Impact of Optimistic Assumptions

Realistic branch prediction

No charge for mispredictions

Following use tournament predictor
Realistic branch prediction

No charge for mispredictions

Following use tournament predictor

---

**Figure 3.38 The effect of branch-prediction schemes.**

This graph shows the impact of going from a perfect model of branch prediction (all branches predicted correctly arbitrarily far ahead), to various dynamic predictors (selective and 2-bit), to compile time, profile-based prediction, and finally to using no predictor. The predictors are described precisely in the text.

---

* This figure has been taken from Computer Architecture, A Quantitative Approach, 3rd Edition Copyright 2003 by Elsevier Inc. All rights reserved. It has been used with permission by Elsevier Inc.
Limits of ILP – Impact of Optimistic Assumptions

Finite registers

Following uses 256 int and 256 fp for renaming

© 2003 Elsevier Science (USA). All rights reserved.
* This figure has been taken from Computer Architecture, A Quantitative Approach, 3rd Edition Copyright 2003 by Elsevier Inc. All rights reserved. It has been used with permission by Elsevier Inc.
Limits of ILP – Impact of Optimistic Assumptions

Finite registers

Following uses 256 int and 256 fp for renaming

Figure 3.41 The effect of finite numbers of registers available for renaming. Both the number of FP registers and the number of GP registers are increased by the number shown on the x-axis. The effect is most dramatic on the FP programs, although having only 32 extra GP and 32 extra FP registers has significant impact on all the programs. As stated earlier, we assume a window size of 2K entries and a maximum issue width of 64 instructions. “None” implies no extra registers available.

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

* This figure has been taken from Computer Architecture, A Quantitative Approach, 3rd Edition Copyright 2003 by Elsevier Inc. All rights reserved. It has been used with permission by Elsevier Inc.
Limits of ILP – Impact of Optimistic Assumptions

Imperfect memory alias analysis

* This figure has been taken from Computer Architecture, A Quantitative Approach, 3rd Edition Copyright 2003 by Elsevier Inc. All rights reserved. It has been used with permission by Elsevier Inc.
Imperfect memory alias analysis

Figure 3.43 The effect of various alias analysis techniques on the amount of ILP. Anything less than perfect analysis has a dramatic impact on the amount of parallelism found in the integer programs, and global/stack analysis is perfect (and unrealizable) for the FORTRAN programs. As we said earlier, we assume a maximum issue width of 64 instructions and a window of 2K instructions.

© 2003 Elsevier Science (USA). All rights reserved.
But Limits Studies may be Pessimistic!

For most optimistic study
- WAR and WAW hazards through memory
- Unnecessary dependences (e.g., loop iteration count)
- Overcoming data flow limit – value prediction

For more realistic studies
- Address value prediction and speculation
- Speculating on multiple paths
Multithreading: Instruction + Thread Level Parallelism

Often superscalar instruction slots are wasted

Why not use them for other threads?

**Multithreading**

- Coarse-grained
- Fine-grained
- Simultaneous multithreading (SMT) or hyperthreading

(Vs. multiprocessing)

4 way
Multithreading: Instruction + Thread Level Parallelism

Vs. Multiprocessing?
Figure 3.28 How four different approaches use the functional unit execution slots of a superscalar processor. The horizontal dimension represents the instruction execution capability in each clock cycle. The vertical dimension represents a sequence of clock cycles. An empty (white) box indicates that the corresponding execution slot is unused in that clock cycle. The shades of gray and black correspond to four different threads in the multithreading processors. Black is also used to indicate the occupied issue slots in the case of the superscalar without multithreading support. The Sun T1 and T2 (aka Niagara) processors are fine-grained multithreaded processors, while the Intel Core i7 and IBM Power7 processors use SMT. The T2 has eight threads, the Power7 has four, and the Intel i7 has two. In all existing SMTs, instructions issue from only one thread at a time. The difference in SMT is that the subsequent decision to execute an instruction is decoupled and could execute the operations coming from several different instructions in the same clock cycle.
Impact of SMT: 1 vs. 4 threads for TPC-C

Relative increase in miss rate or latency

L1 I miss rate
L1 D miss rate
L2 miss rate
L1 I miss latency
L1 D miss latency
L2 miss latency
Impact of SMT: 1 vs. 4 threads for TPC-C **

Figure 3.30 The relative change in the miss rates and miss latencies when executing with one thread per core versus four threads per core on the TPC-C benchmark. The latencies are the actual time to return the requested data after a miss. In the four-thread case, the execution of other threads could potentially hide much of this latency.
SMT Speedup & Energy Efficiency: 1 vs. 4 threads

Copyright © 2011, Elsevier Inc. All rights Reserved.
Figure 3.35 The speedup from using multithreading on one core on an i7 processor averages 1.28 for the Java benchmarks and 1.31 for the PARSEC benchmarks (using an unweighted harmonic mean, which implies a workload where the total time spent executing each benchmark in the single-threaded base set was the same). The energy efficiency averages 0.99 and 1.07, respectively (using the harmonic mean). Recall that anything above 1.0 for energy efficiency indicates that the feature reduces execution time by more than it increases average power. Two of the Java benchmarks experience little speedup and have significant negative energy efficiency because of this. Turbo Boost is off in all cases. These data were collected and analyzed by Esmaeilzadeh et al. [2011] using the Oracle (Sun) HotSpot build 16.3-b01 Java 1.6.0 Virtual Machine and the gcc v4.4.1 native compiler.