

Limits of Instruction-Level Parallelism

CS 282 – KAUST – Spring 2010

Muhamed Mudawar

Original slides created by: David Patterson



Outline

- Limits to ILP
- Thread Level Parallelism
- Multithreading
- Simultaneous Multithreading
- Power 4 vs. Power 5
- Head to Head: VLIW vs. Superscalar vs. SMT

Limits to ILP

- Conflicting studies of amount of ILP
 - Selected Benchmarks
 - Hardware sophistication
 - Compiler sophistication
- How much ILP is available using existing mechanisms with increasing HW budgets?
- Do we need to invent new HW/SW mechanisms to keep on processor performance curve?
 - Intel MMX, SSE (Streaming SIMD Extensions)
 - Intel SSE2: 128 bit, including 2 64-bit Fl. Pt. per clock
 - Motorola AltiVec: 128 bit ints and FPs
 - Supersparc Multimedia ops, etc.

3

Overcoming Limits

- Advances in compiler technology + significantly new and different hardware techniques *may* be able to overcome limitations assumed in studies
- However, unlikely such advances when coupled *with realistic hardware* will overcome these limits in near future

4

Limits to ILP

Initial HW Model here; MIPS compilers.

Assumptions for ideal/perfect machine to start:

1. *Register renaming* – infinite virtual registers
=> all register WAW & WAR hazards are avoided
2. *Branch prediction* – perfect; no mispredictions
3. *Jump prediction* – all jumps perfectly predicted (returns, case statements)
2 & 3 => no control dependencies; perfect speculation & an unbounded buffer of instructions available
4. *Memory-address alias analysis* – addresses known & a load can be moved before a store provided addresses not equal; 1&4 eliminates all but RAW

Also: perfect caches; 1 cycle latency for all instructions (FP *,/); unlimited instructions issued/clock cycle;

5

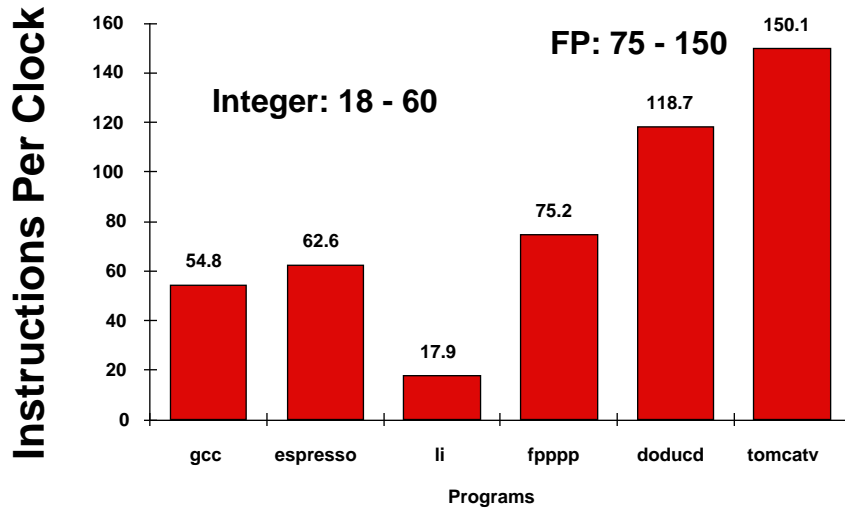
Limits to ILP HW Model comparison

	Model	Power 5
Instructions Issued per clock	Infinite	4
Instruction Window Size	Infinite	200
Renaming Registers	Infinite	48 integer + 40 Fl. Pt.
Branch Prediction	Perfect	2% to 6% misprediction (Tournament Branch Predictor)
Cache	Perfect	64KI, 32KD, 1.92MB L2, 36 MB L3
Memory Alias Analysis	Perfect	??

6

Upper Limit to ILP: Ideal Machine

(Figure 3.1)



7

Limits to ILP HW Model comparison

	New Model	Model	Power 5
Instructions Issued per clock	Infinite	Infinite	4
Instruction Window Size	Infinite, 2K, 512, 128, 32	Infinite	200
Renaming Registers	Infinite	Infinite	48 integer + 40 Fl. Pt.
Branch Prediction	Perfect	Perfect	2% to 6% misprediction (Tournament Branch Predictor)
Cache	Perfect	Perfect	64KI, 32KD, 1.92MB L2, 36 MB L3
Memory Alias	Perfect	Perfect	??

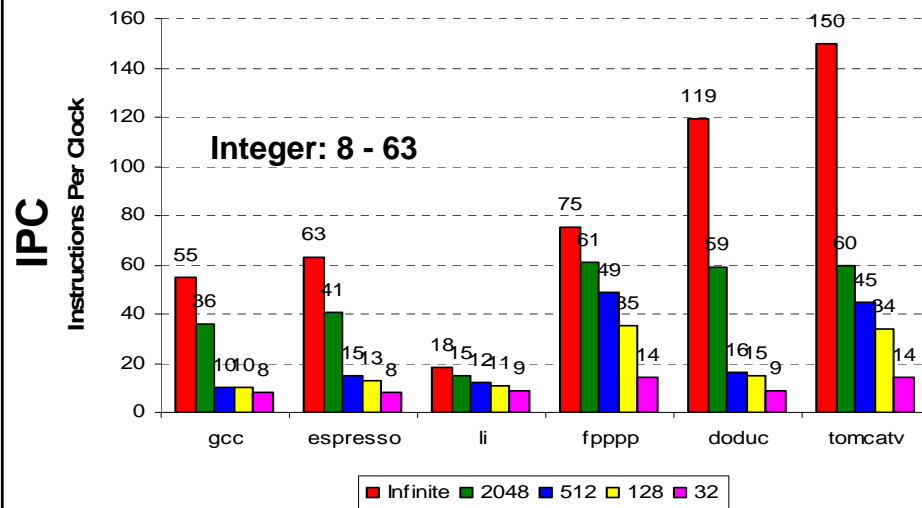
More Realistic HW: Window Impact

Figure 3.2

Instruction Window: Set of instructions that is examined for simultaneous execution

Change from Infinite window 2048, 512, 128, 32

FP: 9 - 150



Complexity of Issuing N Instructions

- To determine whether N issuing instructions have any register dependences among themselves requires:

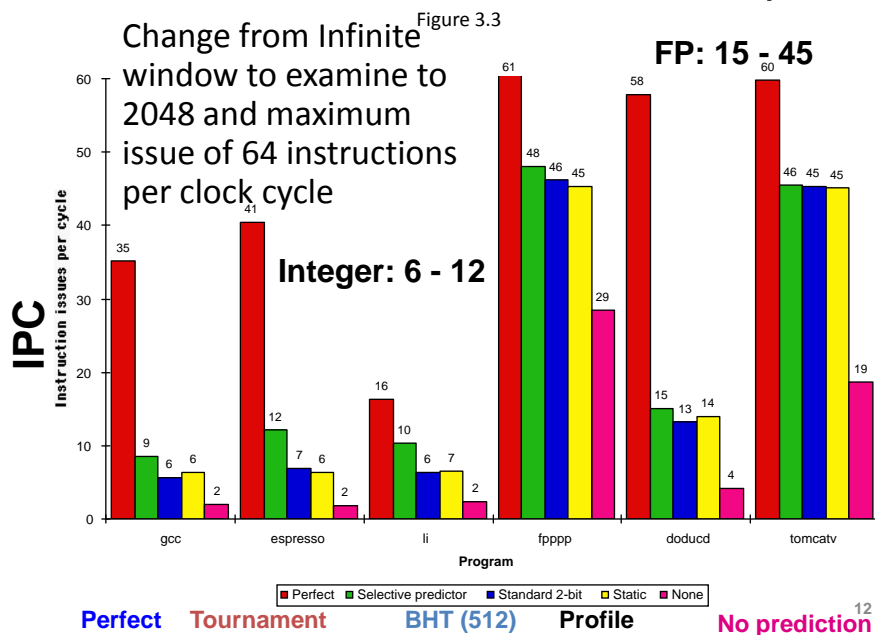
$$(2n - 2) + (2n - 4) + \dots + 2 = 2 \sum_{i=1}^{n-1} i = 2 \frac{(n-1)n}{2} = n^2 - n \text{ comparisons}$$

- Assuming all instructions are register-register and number of registers is unbounded
- To issue 50 instruction per cycle requires 2450 comparisons
- To issue 2000 instructions per cycle requires almost 4 millions comparisons!

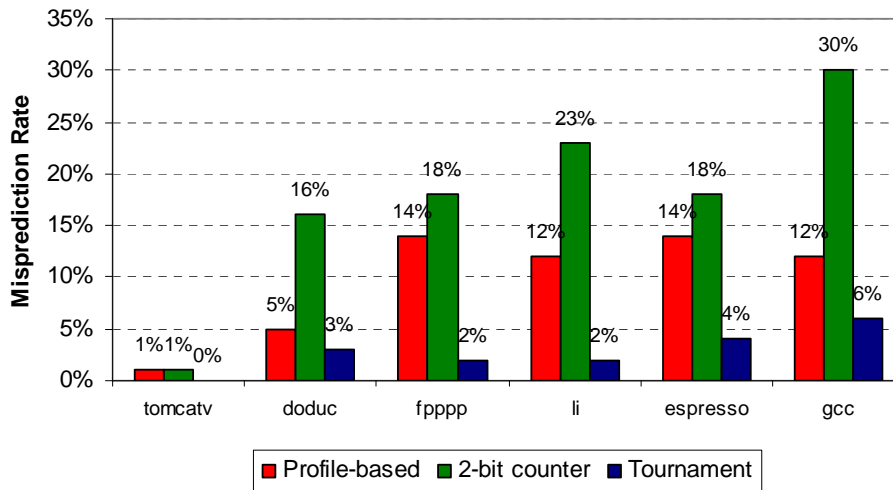
Limits to ILP HW Model comparison

	New Model	Model	Power 5
Instructions Issued per clock	64	Infinite	4
Instruction Window Size	2048	Infinite	200
Renaming Registers	Infinite	Infinite	48 integer + 40 Fl. Pt.
Branch Prediction	Perfect vs. 8K Tournament vs. 512 2-bit vs. profile vs. none	Perfect	2% to 6% misprediction (Tournament Branch Predictor)
Cache	Perfect	Perfect	64KI, 32KD, 1.92MB L2, 36 MB L3
Memory Alias	Perfect	Perfect	??

More Realistic HW: Branch Impact



Misprediction Rates



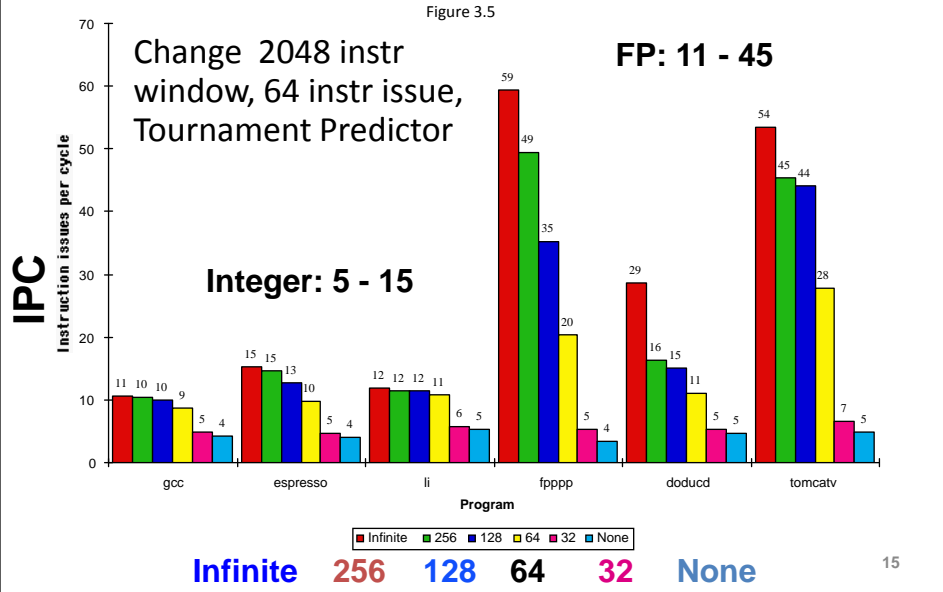
13

Limits to ILP HW Model comparison

	New Model	Mode I	Power 5
Instructions Issued per clock	64	Infinite	4
Instruction Window Size	2048	Infinite	200
Renaming Registers	Infinite v. 256, 128, 64, 32, none	Infinite	48 integer + 40 FI. Pt.
Branch Prediction	Ambitious Tournament Predictor (150Kbits)	Perfect	Tournament Branch Predictor
Cache	Perfect	Perfect	64KI, 32KD, 1.92MB L2, 36 MB L3
Memory Alias	Perfect	Perfect	Perfect

14

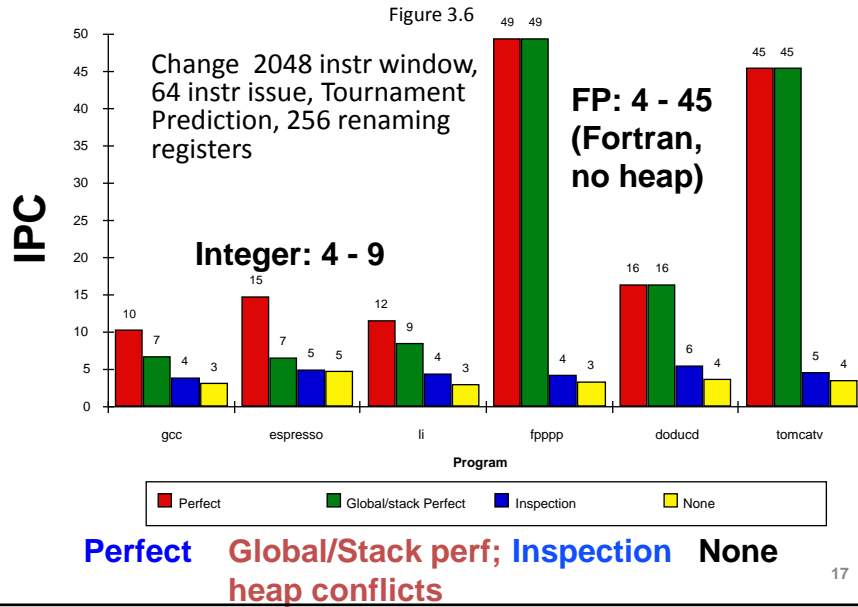
More Realistic HW: Renaming Register Impact (N int + N fp)



Limits to ILP HW Model comparison

	New Model	Model	Power 5
Instructions Issued per clock	64	Infinite	4
Instruction Window Size	2048	Infinite	200
Renaming Registers	256 Int + 256 FP	Infinite	48 integer + 40 Fl. Pt.
Branch Prediction	Ambitious Tournament	Perfect	Tournament
Cache	Perfect	Perfect	64KI, 32KD, 1.92MB L2, 36 MB L3
Memory Alias	Perfect v. Stack v. Inspect v. none	Perfect	Perfect

More Realistic HW: Memory Address Dependence Analysis



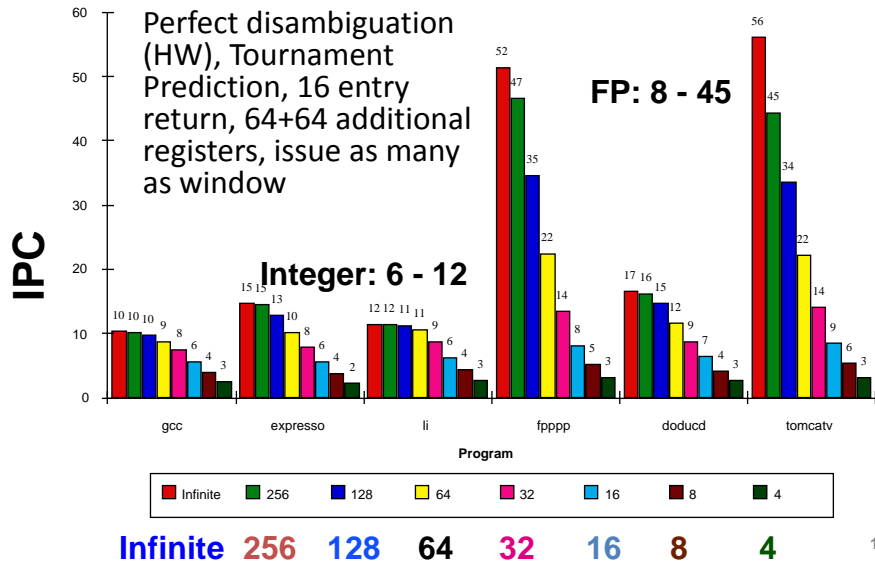
Limits to ILP HW Model comparison

	New Model	Model	Power 5
Instructions Issued per clock	64 (no restrictions)	Infinite	4
Instruction Window Size	Infinite vs. 256, 128, 64, 32	Infinite	200
Renaming Registers	64 Int + 64 FP	Infinite	48 integer + 40 Fl. Pt.
Branch Prediction	Tournament with 16 entry return predictor	Perfect	Tournament
Cache	Perfect	Perfect	64KI, 32KD, 1.92MB L2, 36 MB L3
Memory Alias	Perfect HW disambiguation	Perfect	Perfect

18

Realistic HW: Window Impact

(Figure 3.7)



Limitations of this Study

- Configurations is more complex and expensive than any existing implementation
- Number of issued instructions (64) is more than 10 times larger than widest-issue processors
- No restrictions on the number of execution units or data cache ports (number of loads/stores per cycle)
- No cache misses (not realistic)
- Latency of all function units = 1 (not realistic)
- In reality, cache misses and non-unit latency of function units will impact the performance.
- Actual performance levels are much lower

Outline

- Limits to ILP
 - Thread Level Parallelism
 - Multithreading
 - Simultaneous Multithreading
 - Power 4 vs. Power 5
 - Head to Head: VLIW vs. Superscalar vs. SMT

21

How to Exceed ILP Limits of this study?

- These are not laws of physics; just practical limits for today, and perhaps overcome via research
- Compiler and ISA advances could change results
- WAR and WAW hazards through memory: eliminated WAW and WAR hazards through register renaming, but not in memory usage
 - Can get conflicts via allocation of stack frames as a called procedure reuses the memory addresses of a previous frame on the stack

22

HW v. SW to increase ILP

- Memory disambiguation: HW is best
- HW Speculation:
 - HW best when dynamic branch prediction better than compile time prediction
 - HW base speculation maintains Precise Exceptions
 - HW doesn't need bookkeeping code or compensation code, which is needed by ambitious software speculation mechanism
 - HW speculation is compiler independent. Does not require new compiler, or recompilation to run well
 - Major disadvantage of HW speculation is increased complexity and cost
- SW Speculation: SW can look ahead to schedule better

23

Performance beyond single thread ILP

- There can be much higher natural parallelism in some applications (e.g., Database or Scientific codes)
- Explicit **Thread Level Parallelism** or **Data Level Parallelism**
- **Thread**: process with own instructions and data
 - Thread may be part of a parallel program of multiple threads, or it may be an independent program
 - Each thread has all the state (instructions, data, PC, register state, and so on) necessary to allow it to execute
- **Data Level Parallelism**: Perform identical operations on data, and lots of data

24

Thread Level Parallelism (TLP)

- ILP exploits implicit parallel operations within a loop or straight-line code segment
- TLP explicitly represented by the use of multiple threads of execution that are inherently parallel
- Goal: Use multiple instruction streams to improve
 1. Throughput of computers that run many programs
 2. Execution time of multi-threaded programs
- TLP could be more cost-effective to exploit than ILP

25

Outline

- Limits to ILP
- Thread Level Parallelism
- Multithreading
- Simultaneous Multithreading
- Power 4 vs. Power 5
- Head to Head: VLIW vs. Superscalar vs. SMT

26

New Approach: Multithreaded Execution

- **Multithreading:** multiple threads to share the functional units of 1 processor via overlapping
 - Processor must duplicate independent state of each thread e.g., a separate copy of register file, a separate PC, and for running independent programs, a separate page table
 - Memory shared through the virtual memory mechanisms, which already support multiple processes
 - HW for fast thread switch; much faster than full process switch \approx 100s to 1000s of clocks
- **When to switch?**
 - Alternate instruction per thread (fine grain)
 - When a thread is stalled, perhaps for a cache miss, another thread can be executed (coarse grain)

27

Fine-Grained Multithreading

- Switches between threads on each instruction, causing the execution of multiples threads to be interleaved
- Usually done in a round-robin fashion, skipping any stalled threads
- CPU must be able to switch threads every clock
- Advantage is it can hide both short and long stalls, since instructions from other threads executed when one thread stalls
- Disadvantage is it slows down execution of individual threads, since a thread ready to execute without stalls will be delayed by instructions from other threads
- Used on Sun's Niagara

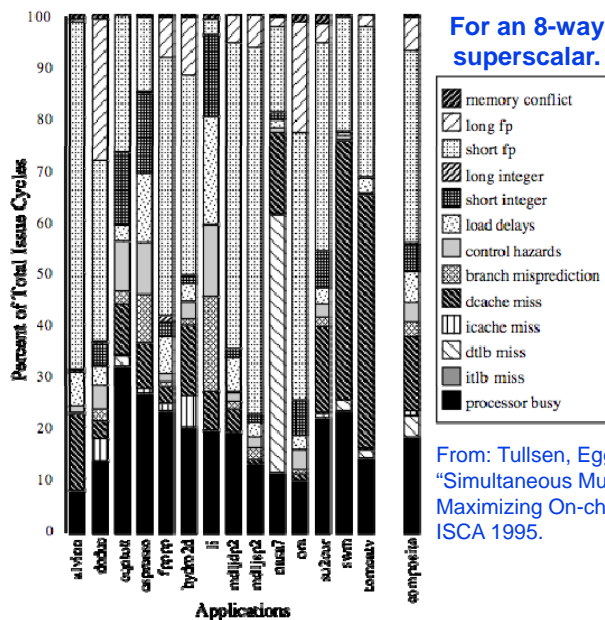
28

Coarse-Grained Multithreading

- Switches threads only on costly stalls, such as L2 cache misses
- Advantages
 - Relieves need to have very fast thread-switching
 - Doesn't slow down thread, since instructions from other threads issued only when the thread encounters a costly stall
- Disadvantage is hard to overcome throughput losses from shorter stalls, due to pipeline start-up costs
 - Since CPU issues instructions from 1 thread, when a stall occurs, the pipeline must be emptied or frozen
 - New thread must fill pipeline before instructions can complete
- Because of this start-up overhead, coarse-grained multithreading is better for reducing penalty of high cost stalls, where pipeline refill \ll stall time
- Used in IBM AS/400

29

For most apps, most execution units lie idle



From: Tullsen, Eggers, and Levy, "Simultaneous Multithreading: Maximizing On-chip Parallelism, ISCA 1995.

30

Do both: ILP and TLP

- TLP and ILP exploit two different kinds of parallel structure in a program
- Could a processor oriented at ILP to exploit TLP?
 - functional units are often idle in data path designed for ILP because of either stalls or dependences in the code
- Could the TLP be used as a source of independent instructions that might keep the processor busy during stalls?
- Could TLP be used to employ the functional units that would otherwise lie idle when insufficient ILP exists?

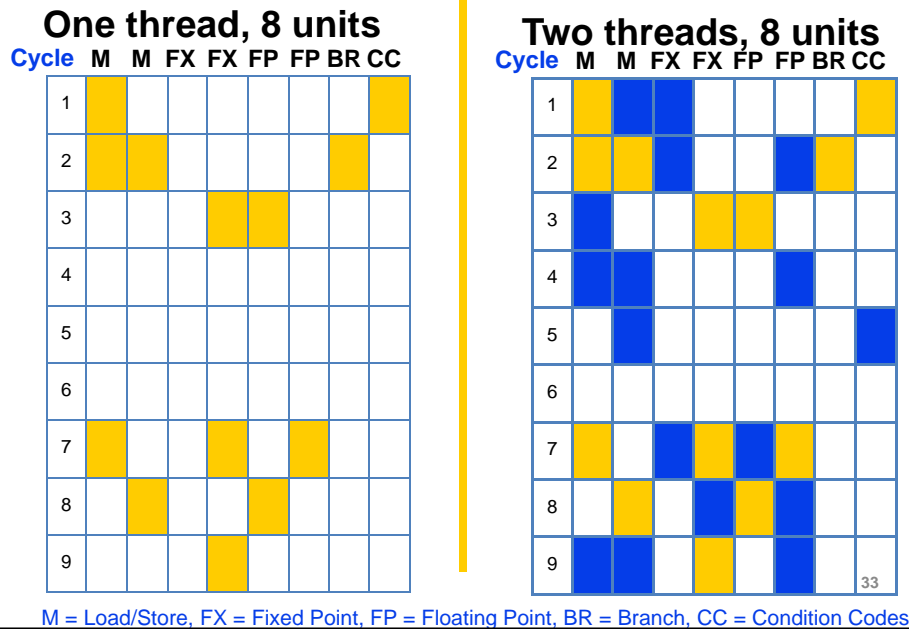
31

Outline

- Limits to ILP
- Thread Level Parallelism
- Multithreading
- Simultaneous Multithreading
- Power 4 vs. Power 5
- Head to Head: VLIW vs. Superscalar vs. SMT

32

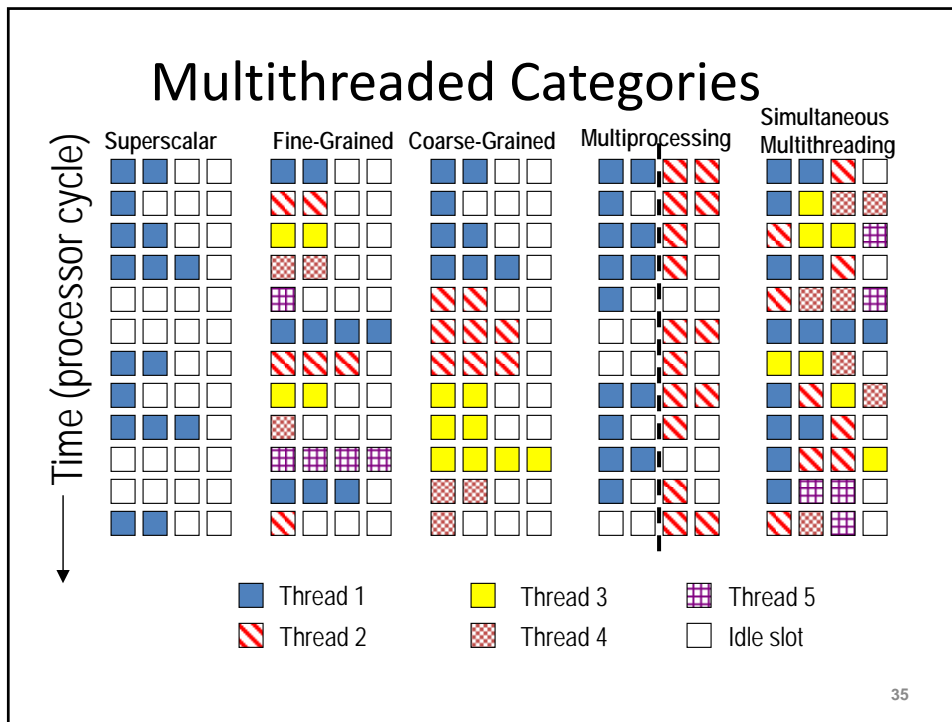
Simultaneous Multi-threading ...



Simultaneous Multithreading (SMT)

- Simultaneous multithreading (SMT): insight that dynamically scheduled processor already has many HW mechanisms to support multithreading
 - Large set of virtual registers that can be used to hold the register sets of independent threads
 - Register renaming provides unique register identifiers, so instructions from multiple threads can be mixed in datapath without confusing sources and destinations across threads
 - Out-of-order completion allows the threads to execute out of order, and get better utilization of the HW
- Just adding a per thread renaming table and keeping separate PCs
 - Independent commitment can be supported by logically keeping a separate reorder buffer for each thread

34



- ## Design Challenges in SMT
- Since SMT makes sense only with fine-grained implementation, what is the impact of fine-grained scheduling on single thread performance?
 - A preferred thread approach sacrifices neither throughput nor single-thread performance
 - Unfortunately, with a preferred thread, the processor is likely to sacrifice some throughput, when preferred thread stalls
 - Larger register file needed to hold multiple contexts
 - Not affecting clock cycle time, especially in
 - Instruction issue - more candidate instructions need to be considered
 - Instruction completion - choosing which instructions to commit may be challenging
 - Ensuring that cache and TLB conflicts generated by SMT do not degrade performance
- 36

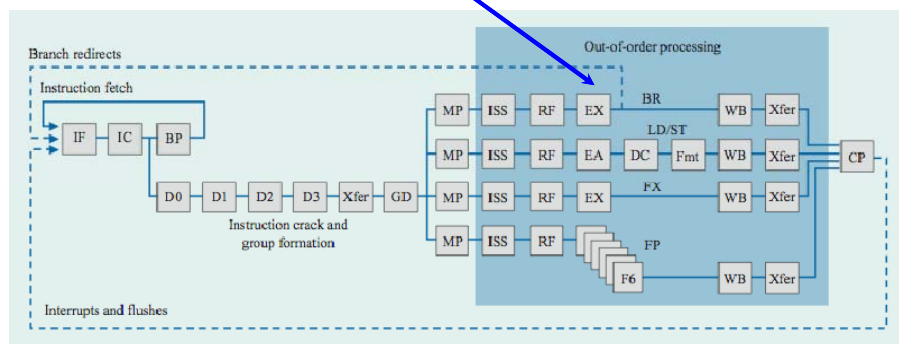
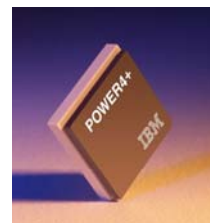
Outline

- Limits to ILP
- Thread Level Parallelism
- Multithreading
- Simultaneous Multithreading
- Power 4 vs. Power 5
- Head to Head: VLIW vs. Superscalar vs. SMT

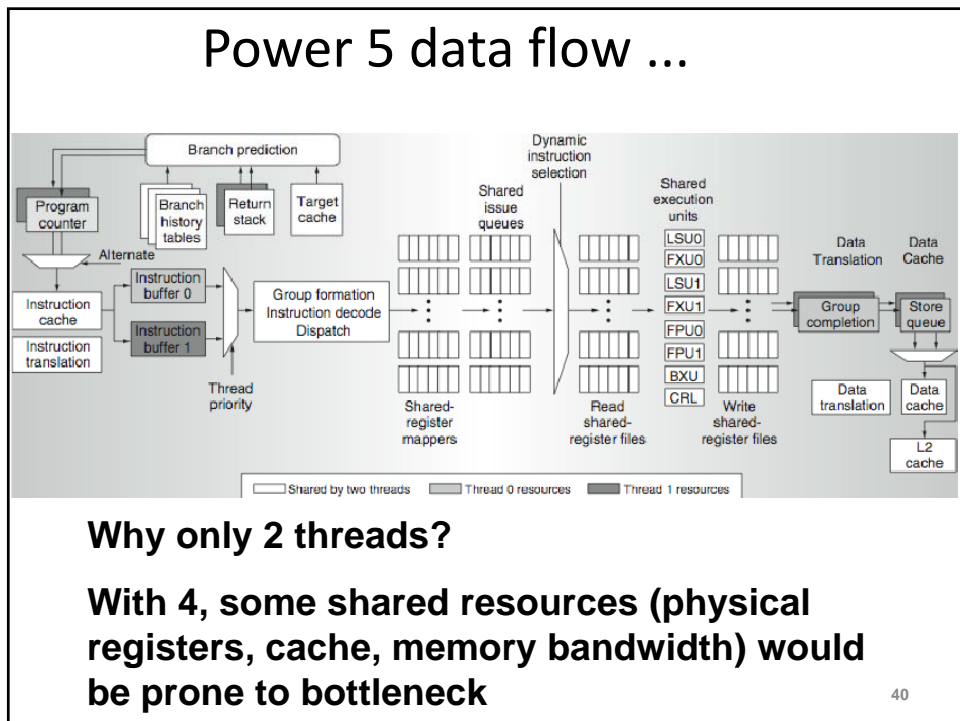
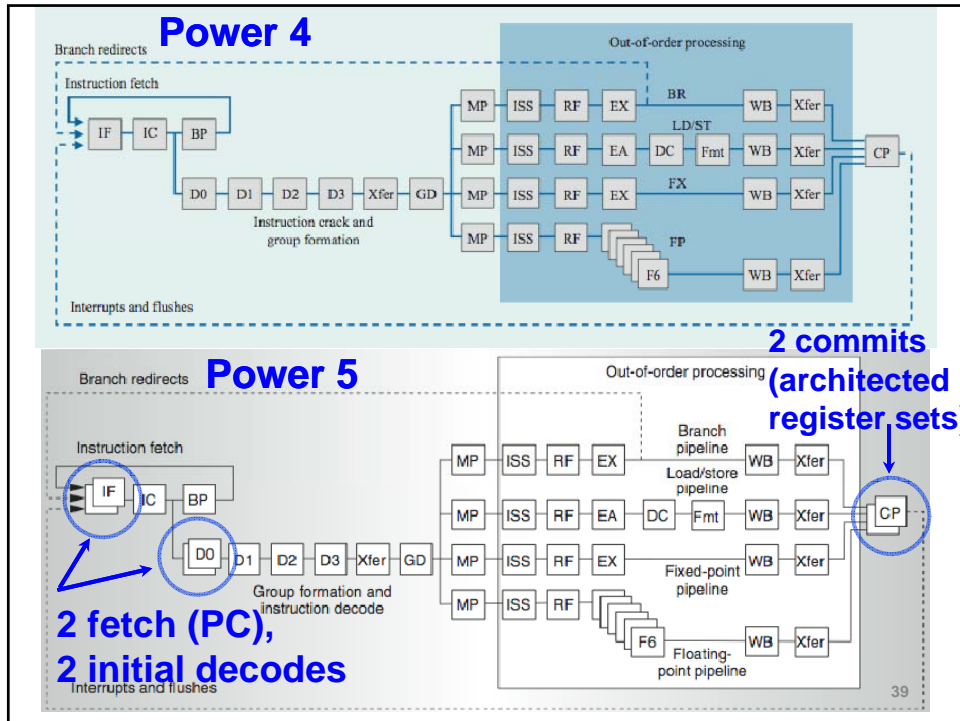
37

Power 4

Single-threaded predecessor to Power 5. 8 execution units in out-of-order engine, each may issue an instruction each cycle.



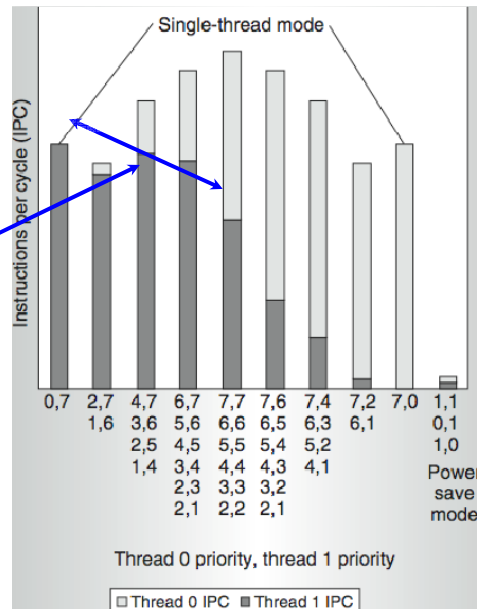
38



Power 5 thread performance ...

Relative priority of each thread controllable in hardware.

For balanced operation, both threads run slower than if they “owned” the machine.



41

Changes in Power 5 to support SMT

- Increased associativity of L1 instruction cache and the instruction address translation buffers
- Added per thread load and store queues
- Increased size of the L2 (1.92 vs. 1.44 MB) and L3 caches
- Added separate instruction prefetch and buffering per thread
- Increased the number of virtual registers from 152 to 240
- Increased the size of several issue queues
- The Power5 core is about 24% larger than the Power4 core because of the addition of SMT support

42

Outline

- Limits to ILP
- Thread Level Parallelism
- Multithreading
- Simultaneous Multithreading
- Power 4 vs. Power 5
- Head to Head: VLIW vs. Superscalar vs. SMT

43

Initial Performance of SMT

- Pentium 4 Extreme SMT yields 1.01 speedup for SPECint_rate benchmark and 1.07 for SPECfp_rate
 - Pentium 4 is dual threaded SMT
 - SPECRate requires that each SPEC benchmark be run against a vendor-selected number of copies of the same benchmark
- Running on Pentium 4 each of 26 SPEC benchmarks paired with every other (26^2 runs) speed-ups from 0.90 to 1.58; average was 1.20
- Power 5, 8 processor server 1.23 faster for SPECint_rate with SMT, 1.16 faster for SPECfp_rate
- Power 5 running 2 copies of each app speedup between 0.89 and 1.41
 - Most gained some
 - Fl.Pt. apps had most cache conflicts and least gains

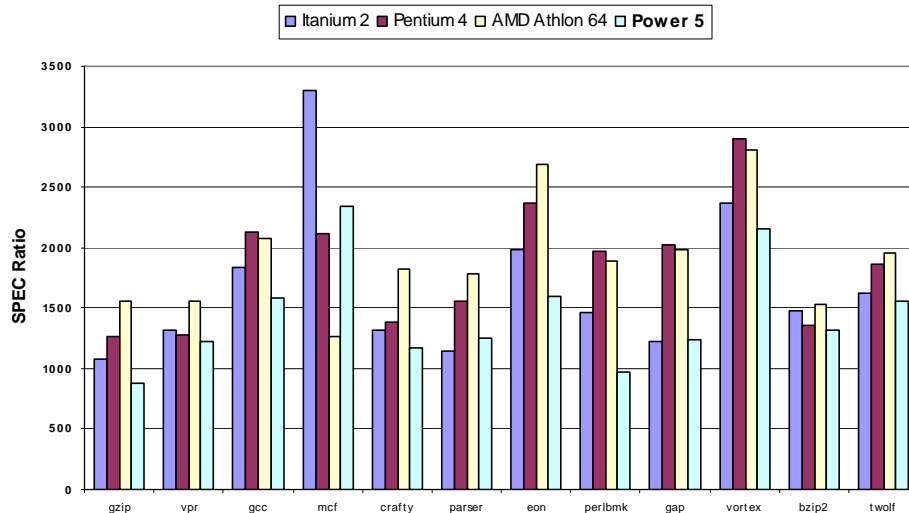
44

Head to Head ILP competition

Processor	Micro architecture	Fetch / Issue / Execute	FU	Clock Rate (GHz)	Transistors Die size	Power
Intel Pentium 4 Extreme	Speculative dynamically scheduled; deeply pipelined; SMT	3/3/4	7 int. 1 FP	3.8	125 M 122 mm ²	115 W
AMD Athlon 64 FX-57	Speculative dynamically scheduled	3/3/4	6 int. 3 FP	2.8	114 M 115 mm ²	104 W
IBM Power5 (1 CPU only)	Speculative dynamically scheduled; SMT; 2 CPU cores/chip	8/4/8	6 int. 2 FP	1.9	200 M 300 mm ² (est.)	80W (est.)
Intel Itanium 2	Statically scheduled VLIW-style	6/5/11	9 int. 2 FP	1.6	592 M 423 mm ²	130 W

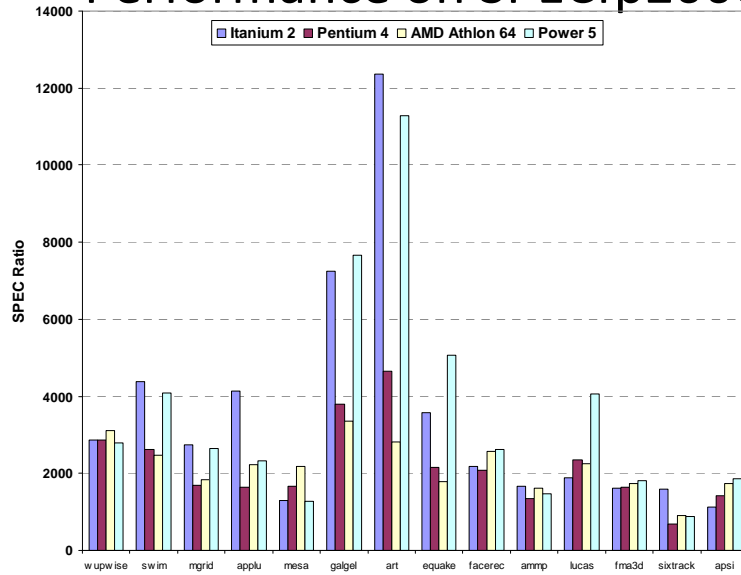
45

Performance on SPECint2000



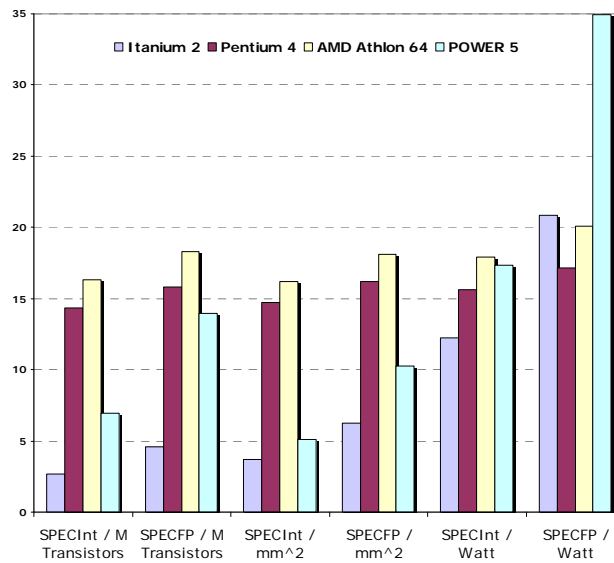
46

Performance on SPECfp2000



47

Normalized Performance: Efficiency



Rank	Itanium 2	Pentium 4	Athlon 64	Power 5
Int/Trans	4	2	1	3
FP/Trans	4	2	1	3
Int/area	4	2	1	3
FP/area	4	2	1	3
Int/Watt	4	3	1	2
FP/Watt	2	4	3	1

48

No Silver Bullet for ILP

- No obvious over all leader in performance
- The AMD Athlon leads on SPECInt performance followed by the Pentium 4, Itanium 2, and Power5
- Itanium 2 and Power5, which perform similarly on SPECFP, clearly dominate the Athlon and Pentium 4 on SPECFP
- Itanium 2 is the most **inefficient** processor both for FI. Pt. and integer code for all but one efficiency measure (SPECFP/Watt)
- Athlon and Pentium 4 both make good use of transistors and area in terms of efficiency,
- IBM Power5 is the most effective user of energy on SPECFP and essentially tied on SPECINT

49

Limits to ILP

- Doubling issue rates above today's 3-6 instructions per clock, say to 6 to 12 instructions, probably requires a processor to
 - issue 3 or 4 data memory accesses per cycle,
 - resolve 2 or 3 branches per cycle,
 - rename and access more than 20 registers per cycle, and
 - fetch 12 to 24 instructions per cycle.
- The complexities of implementing these capabilities is likely to mean sacrifices in the maximum clock rate
 - E.g, widest issue processor is the Itanium 2, but it also has the slowest clock rate, despite the fact that it consumes the most power!

50

Limits to ILP

- Most techniques for increasing performance increase power consumption
- The key question is whether a technique is *energy efficient*: does it increase power consumption faster than it increases performance?
- Multiple issue processors techniques all are energy inefficient:
 1. Issuing multiple instructions incurs some overhead in logic that grows faster than the issue rate grows
 2. Growing gap between peak issue rates and sustained performance
- Number of transistors switching = $f(\text{peak issue rate})$, and performance = $f(\text{sustained rate})$,
growing gap between peak and sustained performance
⇒ increasing energy per unit of performance

51

Commentary

- Itanium architecture does *not* represent a significant breakthrough in scaling ILP or in avoiding the problems of complexity and power consumption
- Instead of pursuing more ILP, architects are increasingly focusing on TLP implemented with single-chip multiprocessors
- In 2000, IBM announced the 1st commercial single-chip, general-purpose multiprocessor, the Power4, which contains 2 Power3 processors and an integrated L2 cache
 - Since then, Sun Microsystems, AMD, and Intel have switch to a focus on single-chip multiprocessors rather than more aggressive uniprocessors.
- Right balance of ILP and TLP is unclear today
 - Perhaps right choice for server market, which can exploit more TLP, may differ from desktop, where single-thread performance may continue to be a primary requirement

52

And in conclusion ...

- Limits to ILP (power efficiency, compilers, dependencies ...) seem to limit to 3 to 6 issue for practical options
- Explicitly parallel (Data level parallelism or Thread level parallelism) is next step to performance
- Coarse grain vs. Fine grained multithreading
 - Only on big stall vs. every clock cycle
- Simultaneous Multithreading if fine grained multithreading based on OOO superscalar microarchitecture
 - Instead of replicating registers, reuse rename registers
- Itanium/EPIC/VLIW is not a breakthrough in ILP
- Balance of ILP and TLP decided in marketplace

53