Computer Architecture: Fundamentals, Tradeoffs, Challenges
Chapter 14: Pot Pourri

Yale Patt

The University of Texas at Austin

Austin, Texas

Spring, 2023

Outline

- ***Measurement Methodology***
- ***RISC***
- ***Spatial Computing***
- ***FPGAs, ASICs, CPUs***
- ***GPUs***

Measurement Methodology

- *The Basic Equation (and what is wrong with it*
- *How do we measure*
  - *Real Hardware, Simulator, Analytic model*
  - *Hardware instrumentation, Microcode, Software monitoring*
- *What do we measure (i.e., benchmarks)*
  - *The ADD instruction*
  - *The Gibson mix*
  - *Synthetic code*
  - *Kernels*
  - *Toy Benchmarks*
  - *SPEC*
  - *The Perfect Club*
  - *Your relevant workload*
- *Serious Abuses*

Why do we measure?

- ***Before the fact***
  - ***So we will know what to build***
- ***After the fact***
  - ***So we will know what to build next time***

The Standard Performance Equation (Everyone uses it, and what is wrong with it!)

$$T = L \times CPI \times t$$

L = Dynamic number of instructions executed (ISA)

CPI = Cycles per instruction (ISA, Microarchitecture)

- *Pipelining, Issue rate, branch handling*
- *How do we compute CPI?*

t = Clock (technology, marketing)

How do we measure? (Degree of Sanitizing)

- ***Real Hardware***
  - ***Gotchas have a chance to get in the way (a good thing!)***
  - ***Least flexible***
  - ***Slow (since you have to build the hdwr), Fast (once you have hdwr)***
  - ***Fast for doing a thorough job (only one that does a thorough job!)***
- ***Simulation***
  - ***Not thorough, some effects are missing***
  - ***Most flexible***
  - ***Slowest for actual measuring***
- ***Analytic Model***
  - ***Good for gross effects, not thorough at all***
  - ***Therefore, must be validated***

How do we measure? (Invasiveness)

- ***Hardware instrumentation***
  - ***Most expensive***
  - ***Non-invasive***
  - ***Least flexible***
- ***Microcoded instrumentation***
  - ***Best of both worlds (e.g., performance counters, SSMT)***
  - ***SPAM (System Performance Analysis using Microcode)***
- ***Software monitoring***
  - ***Cheap***
  - ***Very invasive***
  - ***Most flexible***

Benchmarks

- ***Why benchmarks?***
  - ***Find a set of programs or program fragments REPRESENTATIVE of the WORKLOAD you need the machine for***
- ***Types***
  - ***The ADD instruction***
  - ***Instruction MIX (Gibson Mix, 1959)***
  - ***Kernels (Livermore Loops, Berkeley's 13 dwarfs, micro-benchmarks)***
  - ***Synthetic Benchmarks (Easy parameterization, but RRW is not RWR)***
  - ***Toy benchmarks (easy to hand-compile, strongly disparaged today)***
  - ***SPEC (Systems Performance Evaluation Co-operative), Agreement!***
  - ***Real workload – Why you actually bought the machine!***

A few of my concerns

- *One number: SpecMARK (Better than ADD time)*
- *SimplScalar (very low bar to entry, and not bug-free)*
- *In the literature (1.85 IPC max, Issue width does not matter)*
- *400 floating point ops or 1 LLC miss*
- *Power models*

BAD ways to measure performance (…and each has been used and published)

- ***Apples and Oranges***
  - ***RISC (Counting Simulated Cycles) vs A lightly loaded VAX***
- ***Who should get the credit***
  - ***The architecture or the compiler (Berkeley Pascal or VMS Pascal)***
  - ***Algorithm optimizations (disallowed by SPEC, determininant concat)***
  - ***Instruction set or Register Windows (Bob Colwell)***
- ***Choice of Benchmarks***
  - ***Overstates significance of a feature (procedure call, no floating point)***
  - ***Small size (100% fits in cache, TLB hits, no I/O)***

BAD ways to measure performance (continued)

- ***Play with Statistics (Which machine is better)***

	Program A	*Program B*
Machine 1:	***1 unit***	***2 units***
Machine 2:	***2 units***	***1 unit***

Machine 1 is twice as fast on A, half as fast on B
Speedup is ½ (2 + ½) = 1.25 …Hello!

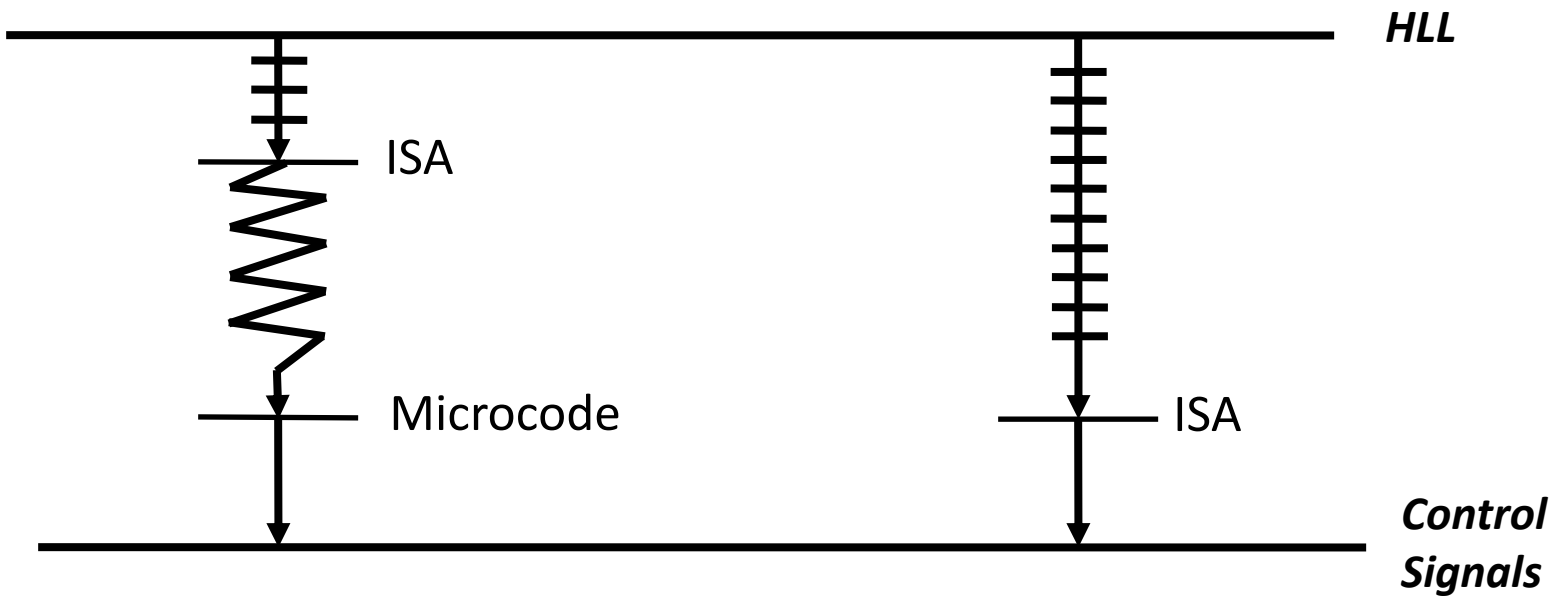
Short Retrospective on RISC

- *From RISC I to RISC V*
- *Open Microcode*
- *What is it (technical)*
- *Characteristics*
- *What is it (non-technical)*
- *Why did it happen*
- *Comments on the Hype*
- *The Notable Ventures*
- *Berkeley and Stanford*

From RISC 1 to RISC V

- ***RISC I***
  - ***Patterson's initial RISC activity at Berkeley (1980)***
  - ***Carlo Sequin provided the name: Reduced Instruction Set Computer***
  - ***Emphasis on Simple set of Simple Instructions with NO microcode***
- ***RISC 2***
  - ***Second version, by Manolis Katvenis, Robert Sherborne, D. Loupis***
- ***SOAR (Smalltalk on a RISC)***
  - ***Recently renamed RISC 3***
  - ***David Ungar, Joan Pendleton***
- ***SPUR (Symmetric Processor under RISC)***
  - ***Recently renamed RISC 4***
  - ***David Wood, Susan Eggers, Mark Hill***
- ***RISC V (to continue the numbering sequence)***
  - ***Open source***

Open Microcode



- ***Compiler Generates Lowest Level of Interpretation***
  - ***No Microcode***
  - ***Single Cycle Execution***
- ***Complex Compiler vs. Complex Hardware***
- ***Issues:***
  - ***Bandwidth, Compiler Complexity, On Chip Tailoring***
  - ***Wasted Cycles***

What is it?

- ***Originally : Open Microcode***
  - ***John Cocke (1970's)***
- ***1980: Simple Set of Simple Instructions***
  - ***Sequin, Patterson (1980)***
- ***1989: Short, Tight Pipelines***
  - ***John Hennessy***
- ***1994: VLIW***
  - ***Wall Street Journal***

Characteristics

- *Fixed Length, Uniform Decode Instructions*
- *No Microcode*
- *Load/Store*
- *Larger Register Set*

- *Delayed Branch*
- *Register Windows*

What is it (Non-Technical)

- ***Everything New Since 1983***
- ***"Good"***
  - ***Motorola 68010 Article***
  - ***Microcoded RISC Article***
  - ***MicroVAX – 2***
  - ***VAX 9000 Literature***
- ***SPARC System***
  - ***The "RISC" Core***

Why Did It Happen

- ***Masterful Marketing***
  - ***Published Berkeley Benchmarks***
  - ***RISC Chip took Weeks, VAX took Years***
  - ***Simple is Beautiful***
  - ***4-on-floor vs. Automatic***
- ***Time-to-Market Curve***
  - ***VAX 8600 Was Very Late***
  - ***Track Technology Curve***
- ***Why Was it Taken Seriously***
  - ***My opinion: Because HP Bet the Family Store***

Comments on the Hype

- ***Simple is Beautiful***
  - ***Complex Instructions Provide Opportunity for Speed-Up***
    - ***1st add Fl.Pt.***
    - ***Graphics***
    - ***MMX***
  - ***Compilers Never Use It***
    - ***Some BAD Implementations***
    - ***One Compiler or All Compilers***
- ***Published Berkeley Benchmarks***
- ***Why did H-P jump in ?***

The Notable Ventures

- ***The first project (pre-1980)***
  - ***IBM 801 (IBM Yorktown, John Cocke)***

- ***University Projects***
  - ***RISC (Berkeley, 1980, Patterson, Emphasis on "Simple")***
  - ***MIPS (Stanford, 1981, Hennessy, Emphasis on Compiler)***

- ***Commercial Products (1980s: RISC's Golden Years)***
  - ***IBM 801 (IBM Yorktown, John Cocke, the first project, pre-1980)***
  - ***HP-PA (The IBM Team, Emphasis on Compiler)***
  - ***SPARC ("modified" Berkeley RISC)***
  - ***MIPS (Simple)***
  - ***AMD 29000***
  - ***Motorola 88000 (Initially intended to replace M68000)***
  - ***IBM RISC System 6000 (IBM Austin, John Cocke, Return to Past)***

- ***Notable later product lines***
  - ***Digital Equipment Corporation (ALPHA 21064, 21164, 21264)***
  - ***IBM, Motorola, Apple (Power PC 601, 604, 620, …)***

Berkeley and Stanford

- ***Berkeley***
  - ***Register Windows***
  - ***Delayed Branch***
  - ***No microcode → use the area for registers***
- ***Stanford***
  - ***Two instructions per 32 bit instruction word***
  - ***Pipeline Reorganizer***
  - ***Delayed Branch***

The RISC/CISC Wars

- ***What was it?***
  - ***RISC: Reduced Instruction Set Computers***
    - ***Most emphasized simple***
    - ***"Speed demons"***
  - ***CISC: Complex (or, Comprehensive) Instruction Set Computers***
    - ***Emphasixed dense encoding of the instructions***
    - ***A lot of work specified in each instruction***
    - ***"Brainiacs"***
  - ***A continual debate as to which was better (1980s, 1990s)***

- ***Why did CISC (i.e., x86) win the war?***
  - ***X86 more than doubled the pipeline depth, allowing increased frequency***
  - ***Application base for x86 was much larger than for any of the RISCs***
  - ***Worth noting: War was about desktops and NOT embedded processors!***

Spatial Computing

- *Code laid out in space as a data flow graph*
- *Data passed from producer unit to consumer unit*
- *Excellent if the code has to work on large number of data sets*
- *Critical problem: turning sequential code into a data flow graph*
- *Enormous performance possible since most nodes are processing*

FPGAs, ASICs, CPU

- ***FPGAs and ASICs are accelerators***
  - ***Both provide speed-up over code running on a CPU***
- ***Performance vs Flexibility***
  - ***FPGA consumes more energy, takes more space, and is slower***
  - ***But it is more flexible***
  - ***My analogy: a set of wrenches vs a single adjustable wrench***

GPUs

- *SMT, SIMD, and Predication*
- *Bottlenecks to Performance*
  - *Branch Divergence*
  - *Memory Coalescing*
- *Ten thousand processors per chip*
  - *32 SIMD lanes*
  - *32 Stream Processors*
  - *10 way SMT*

Blagodarya!!