Computer Science 146
Computer Architecture

Fall 2019

Harvard University

Instructor: Prof. David Brooks
dbrooks@eecs.harvard.edu

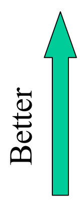
Lecture 2: Performance Metrics, ISA Intro

Lecture Outline

- Performance Metrics
- Averaging
- Amdahl's Law
- Benchmarks
- The CPU Performance Equation
  - Optimal Pipelining Case Study
- Modern Processor Analysis
- Begin ISAs…

Performance Metrics

- Execution Time is often what we target
- Throughput (tasks/sec) vs. latency (sec/task)
- How do we decide the tasks? Benchmarks



  - What the customer cares about, real applications
  - Representative programs (SPEC, SYSMARK, etc)
  - Kernels: Code fragments from real programs (Linpack)
  - Toy Programs: Sieve, Quicksort
  - Synthetic Programs: Just a representative instruction mix (Whetsone, Dhrystone)

Measuring Performance

- Total Execution Time:

$$\frac{1}{n}\sum_{i=1}^{n} \text{Time}_i$$

- This is *arithmetic* mean
  - This should be used when measuring performance in execution *times* (CPI)

Measuring Performance

- Weighted Execution Time:

$$\sum_{i=1}^{n} \text{Weight}_i \times \text{Time}_i$$

- What if P1 and P2 are not run equally?

Measuring Performance

- Normalized Execution Time
- Normalize to *reference* machine
- Can only use geometric mean (arithmetic mean can vary depending on the reference machine)

$$\sqrt[n]{\prod_{i=1}^{n} ExecutionTimeRatio_i}$$

- Problem: Ratio not Execution Time is the result

Harmonic Mean: Motivation

- 30 mph for the first 10 miles
- 90 mph for the next 10 miles
- Average speed? (30+90)/2 = 60mph

- WRONG! Average speed = total distance / total time
- 20/(10/30+10/90) = 45mph

Harmonic Mean

- Each program has *O* operations
- n programs executed n*O* operations in $\Sigma\ T_i$
- Execution rate is then

$$nO / \Sigma\ T_i = n / \Sigma\ (T_i/O) = n / \Sigma\ 1/P_i$$

where $1/P_i$ is the rate of execution of program i

- Harmonic mean should be used when measuring performance in execution *rates* (IPC)

Amdahl's Law (Law of Diminishing Returns)

- Very Intuitive – Make the Common case fast

$$\text{Speedup} = \frac{\text{Execution Time for task without enhancement}}{\text{Execution Time for task using enhancement}}$$

$$\text{Execution time}_{new} = \text{Execution time}_{old} \times \left((1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}} \right)$$

Amdahl's Law Corollary

$$\text{Speedup}_{Overall} = \frac{1}{\left((1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}} \right)}$$

As $\text{Speedup}_{Enhanced} >> 0$, $\text{Speedup}_{Overall} = \frac{1}{(1 - \text{Fraction}_{enhanced})}$

Amdahl's Law Example

$Fraction_{enhanced}$=95%, $Speedup_{Enhanced}$=1.1x

$Speedup_{Overall}$ =1/((1-.95)+(.95/1.1))=1.094

Make the common case fast!

$Fraction_{enhanced}$=5%, $Speedup_{Enhanced}$=10x

$Speedup_{Overall}$ =1/((1-.05)+(.05/10))=1.047

$Fraction_{enhanced}$=5%, $Speedup_{Enhanced}$=Infinity

$Speedup_{Overall}$ =1/(1-.05)=1.052

MIPS

- MIPS = instruction count/(execution time x 10^6)
  = clock rate/(CPI x 10^6)

- Problems
  – ISAs are not equivalent, e.g. RISC vs. CISC
    - 1 CISC instruction may equal many RISC!
  – Programs use different instruction mixes
  – May be ok when comparing same benchmarks, same ISA, same compiler, same OS

MFLOPS

- Same as MIPS, just FP ops
- Not useful either
  - FP-intensive apps needed
  - Traditionally, FP ops were slow, INT can be ignored
  - BUT, now memory ops can be the slowest!
- "Peak MFLOPS" is a common marketing fallacy
  - Basically, it just says #FP-pipes X Clock Rate

GHz

- Is this a metric? Maybe as good as the others…
- One number, no benchmarks, what can be better?
- Many designs *are* frequency driven

Processor	Clock Rate	SPEC FP2000
IBM POWER3	450 MHz	434
Intel PIII	1.4 GHz	456
Intel Pentium 4	2.4 GHz	833
Itanium-2	1.0 GHz	1356

Benchmark Suites

- SPEC CPU2000 (int and float) (Desktop, Server)
- EEMBC ("embassy"), SPECjvm (Embedded)
- TPC-C, TPC-H, SPECjbb, ECperf (Server)

SPEC CPU2000: Integer Benchmarks

164.gzip	C	Compression
175.vpr	C	FPGA Circuit Placement and Routing
176.gcc	C	C Programming Language Compiler
181.mcf	C	Combinatorial Optimization
186.crafty	C	Game Playing: Chess
197.parser	C	Word Processing
252.eon	C++	Computer Visualization
253.perlbmk	C	PERL Programming Language
254.gap	C	Group Theory, Interpreter
255.vortex	C	Object-oriented Database
256.bzip2	C	Compression
300.twolf	C	Place and Route Simulator

SPEC CPU2000: Floating Point Benchmarks

168.wupwise	Fortran 77	Physics / Quantum Chromodynamics
171.swim	Fortran 77	Shallow Water Modeling
172.mgrid	Fortran 77	Multi-grid Solver: 3D Potential Field
173.applu	Fortran 77	Parabolic / Elliptic Partial Differential Equations
177.mesa	C	3-D Graphics Library
178.galgel	Fortran 90	Computational Fluid Dynamics
179.art	C	Image Recognition / Neural Networks
183.equake	C	Seismic Wave Propagation Simulation
187.facerec	Fortran 90	Image Processing: Face Recognition
188.ammp	C	Computational Chemistry
189.lucas	Fortran 90	Number Theory / Primality Testing
191.fma3d	Fortran 90	Finite-element Crash Simulation
200.sixtrack	Fortran 77	High Energy Nuclear Physics Accelerator Design
301.apsi	Fortran 77	Meteorology: Pollutant Distribution

Server Benchmarks

- TPC-C (Online-Transaction Processing, OLTP)

System	# / Processor	tpm	$/tpm
Fujitsu PrimePower	128, 563MHz SPARC64	455K	$28.58
HP SuperDome	64, 875MHz PA8700	423K	$15.64
IBM p690	32, 1300MHz POWER4	403K	$17.80

- TPC-H (Ad-hoc, decision support)

CPU Performance Equation

- Execution Time = seconds/program

$$\frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

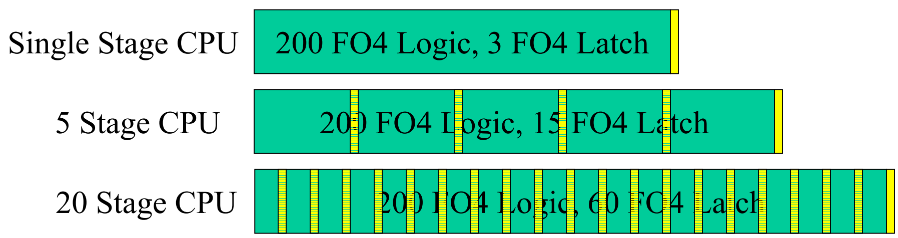
instructions/program	cycles/instruction	seconds/cycle
Program	Compiler (Scheduling)	Technology
Architecture (ISA)	Organization (uArch)	Physical Design
Compiler	Microarchitects	Circuit Designers

Common Architecture Trick

- Instructions/Program (Path-length) is constant
  - Same benchmark, same compiler
  - Ok usually, but for some ideas compiler may change
- Seconds/Cycle (Cycle-time) is constant
  - "My tweak won't impact cycle-time"
  - Often a bad assumption
  - Current designs are ~15-20FO4 Inverter Delays per cycle
- Just focus on Cycles/Instruction (CPI or IPC)
- Most academic architecture studies do just this!

CPU Performance Equation: Case Study

- Fundamental Architecture Decision: How do we choose the number of pipeline stages?



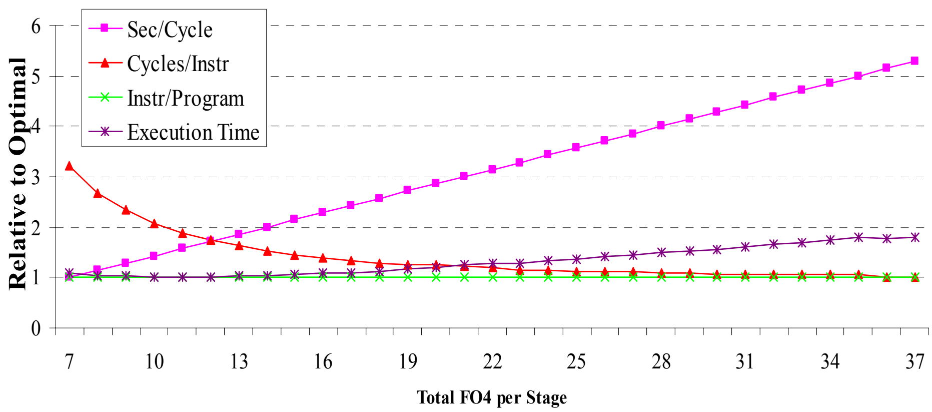
$$\frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

Measuring Instruction Counts and CPI

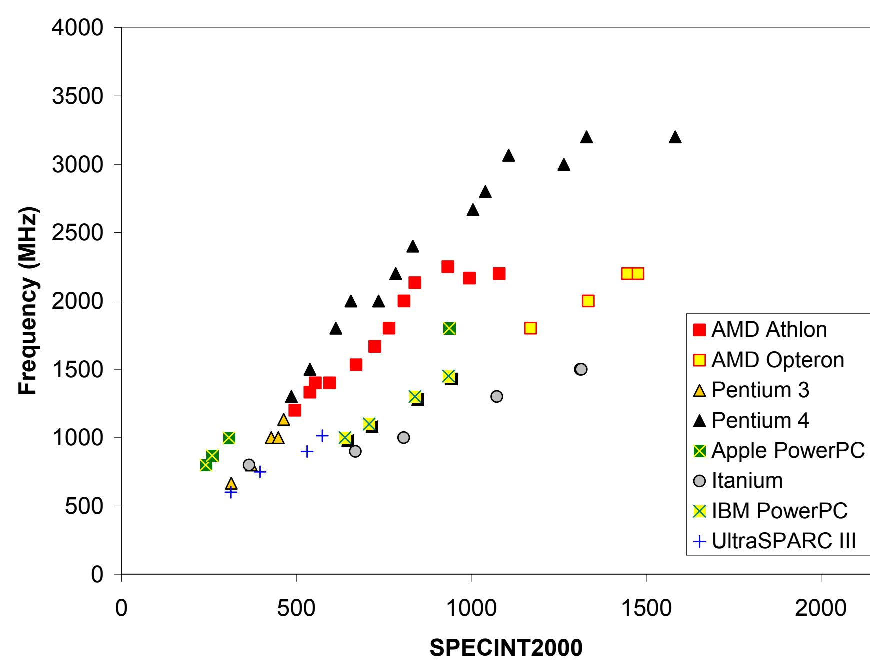
- Hardware counters on processors
- Instrumented execution (ATOM)
- Instruction-level interpretation (instruction counts)

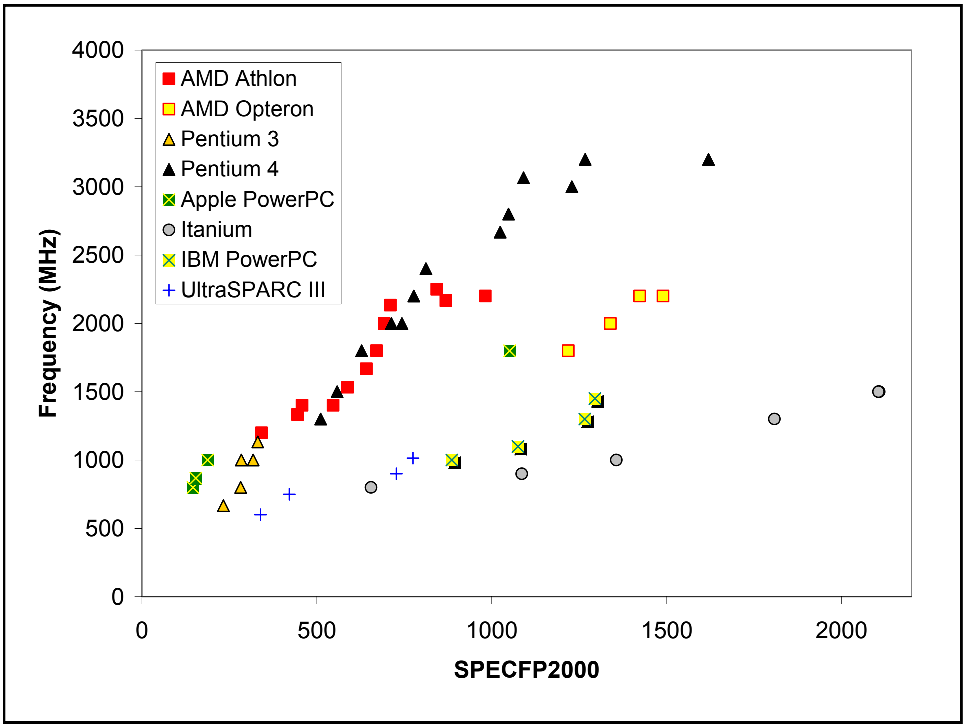
- Execution-driven simulation
  - Detailed simulation of execution core and memory hierarchy

Pipelining Case Study

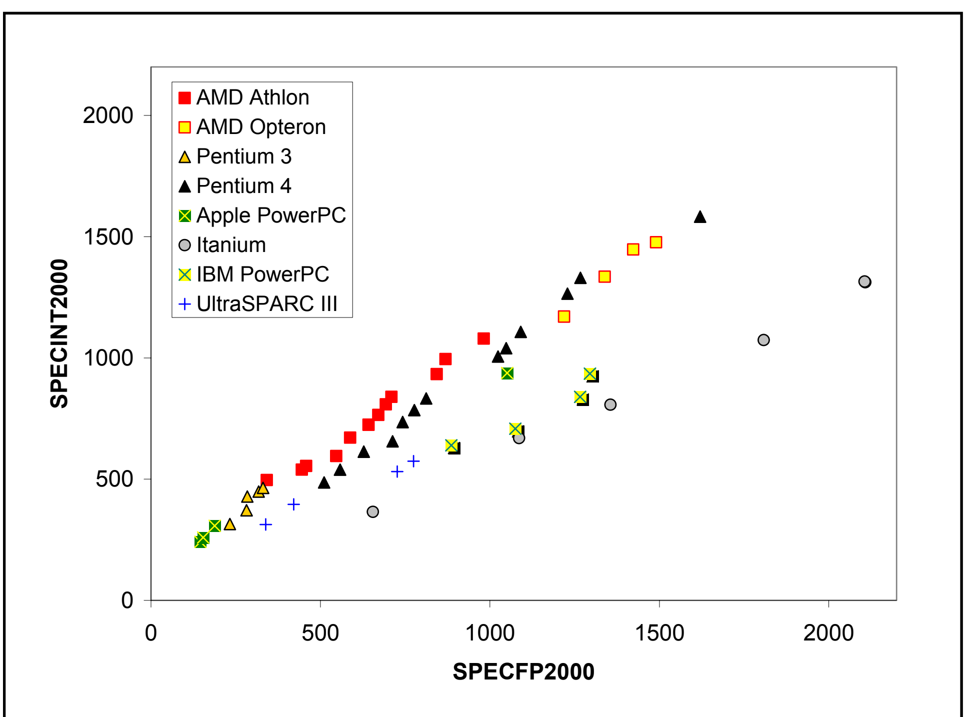


- Caveat: Fixed architecture sizings (only latencies vary)

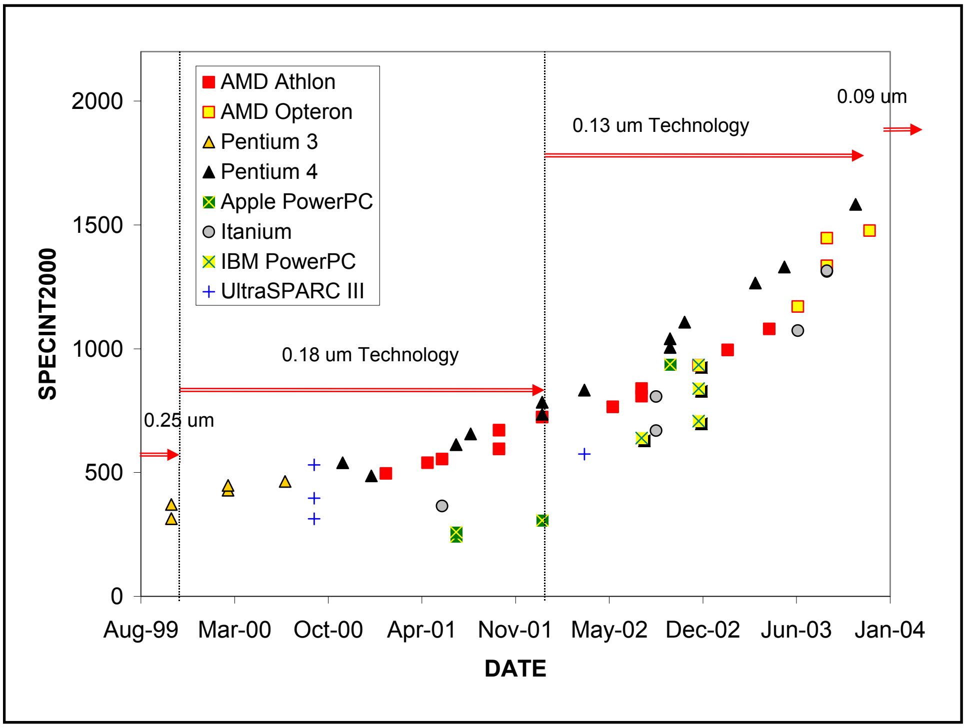




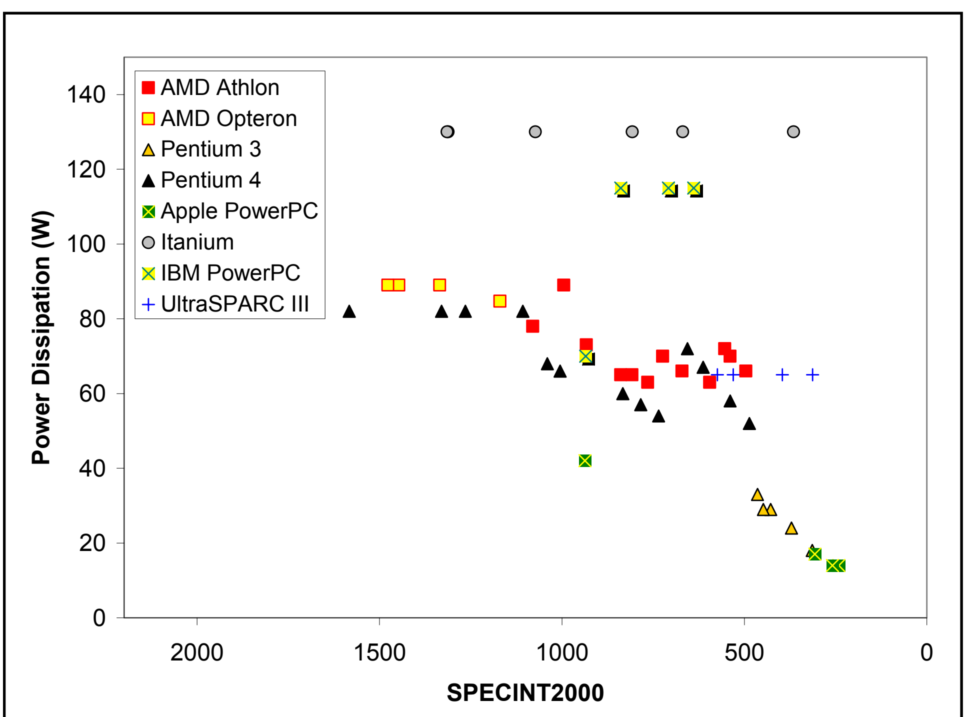
4000
3500
3000
2500
2000
1500
1000
500
0
Frequency (MHz)
AMD Athlon
AMD Opteron
Pentium 3
Pentium 4
Apple PowerPC
Itanium
IBM PowerPC
UltraSPARC III
0
500
1000
1500
2000
SPECFP2000



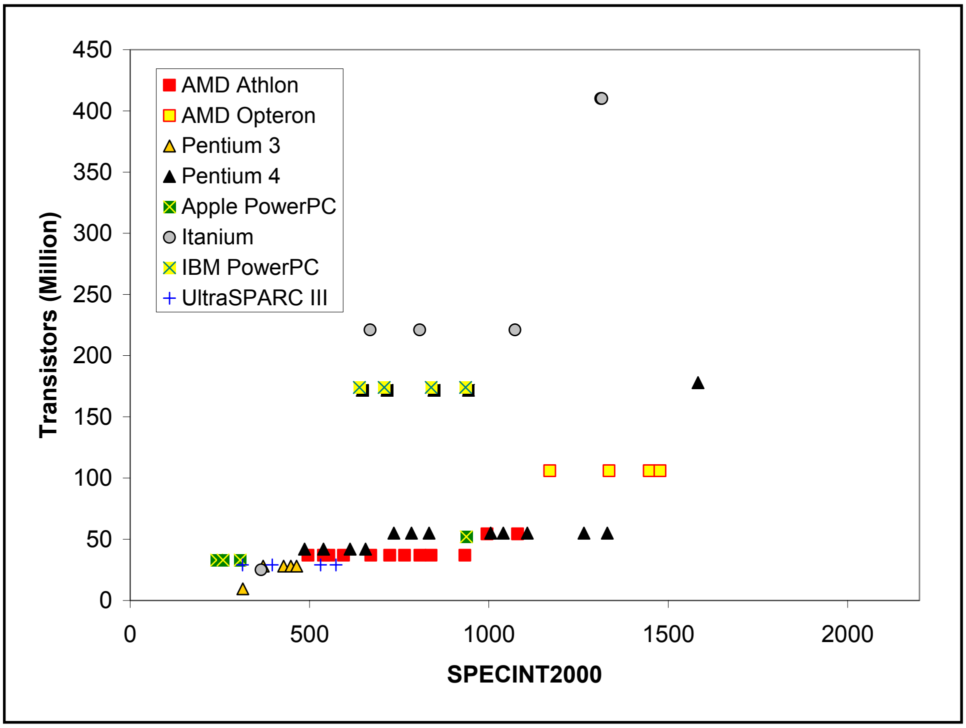
2000
1500
1000
500
0
SPECINT2000
AMD Athlon
AMD Opteron
Pentium 3
Pentium 4
Apple PowerPC
Itanium
IBM PowerPC
UltraSPARC III
0
500
1000
1500
2000
SPECFP2000



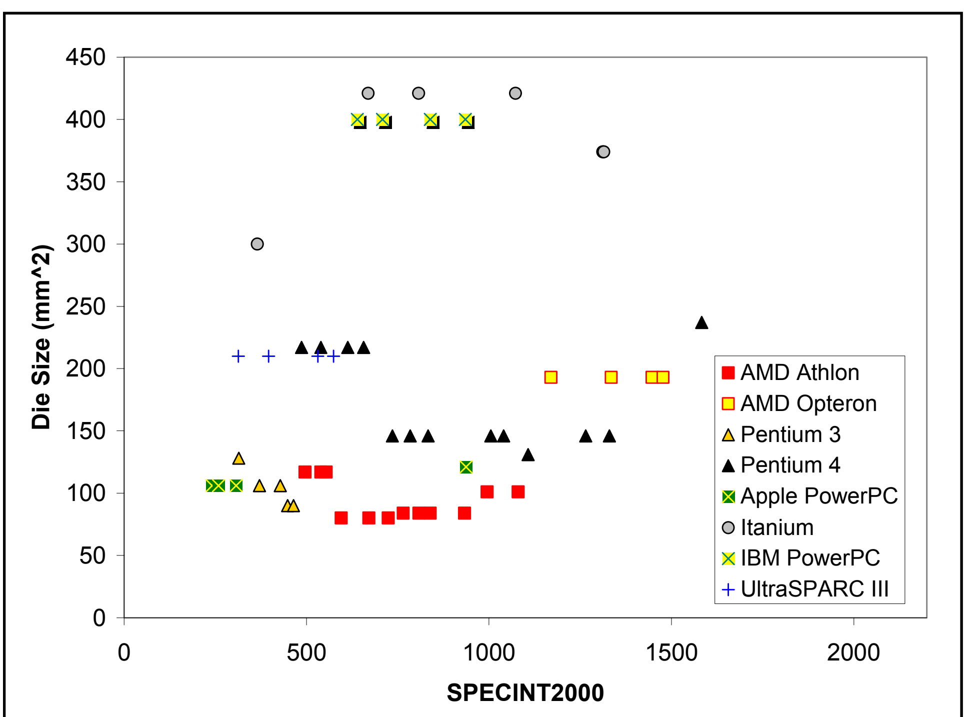
AMD Athlon
AMD Opteron
Pentium 3
Pentium 4
Apple PowerPC
Itanium
IBM PowerPC
UltraSPARC III
0.25 um
0.18 um Technology
0.13 um Technology
0.09 um
SPECINT2000
DATE
2000
1500
1000
500
0
Aug-99
Mar-00
Oct-00
Apr-01
Nov-01
May-02
Dec-02
Jun-03
Jan-04



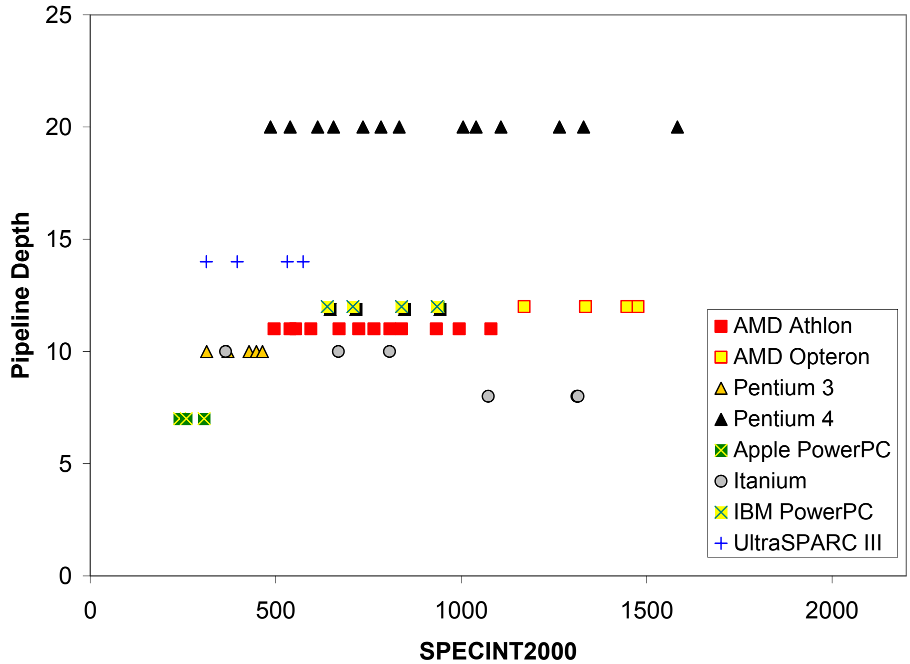
AMD Athlon
AMD Opteron
Pentium 3
Pentium 4
Apple PowerPC
Itanium
IBM PowerPC
UltraSPARC III
Power Dissipation (W)
SPECINT2000
140
120
100
80
60
40
20
0
2000
1500
1000
500
0



Transistors (Million)
450
400
350
300
250
200
150
100
50
0
0
500
1000
1500
2000
SPECINT2000
AMD Athlon
AMD Opteron
Pentium 3
Pentium 4
Apple PowerPC
Itanium
IBM PowerPC
UltraSPARC III



Die Size (mm^2)
450
400
350
300
250
200
150
100
50
0
0
500
1000
1500
2000
SPECINT2000
AMD Athlon
AMD Opteron
Pentium 3
Pentium 4
Apple PowerPC
Itanium
IBM PowerPC
UltraSPARC III



For next time

- Some CISC vs. RISC commentary
- Multimedia ISAs
- Paper available on webpage
  - http://www.eecs.harvard.edu/cs146/
  - **R. Lee, "Subword Parallelism with MAX-2", In IEEE Micro, August 1996.**
- Read through Ch1 and Ch2…
- Compiler/ISA overview, may start pipelining review