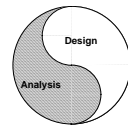


CSE775: Computer Architecture

Chapter 1: Fundamentals of Computer Design

1

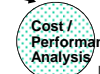
Measurement and Evaluation



Architecture is an iterative process:

- Searching the space of possible designs
- At all levels of computer systems

Creativity

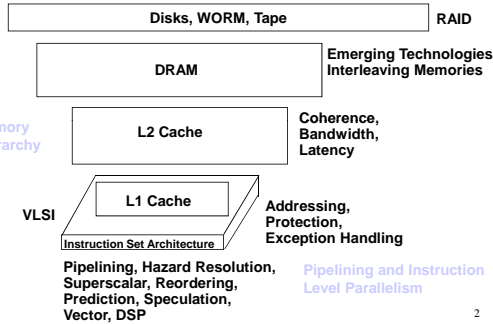


Bad Ideas

4

Computer Architecture Topics

Input/Output and Storage



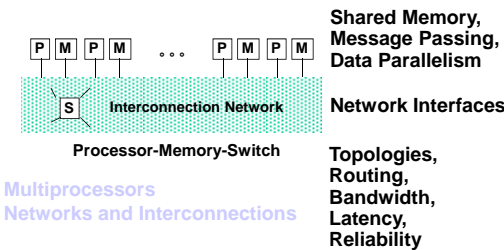
2

Issues for a Computer Designer

- Functional Requirements Analysis (Target)
 - Scientific Computing – High Performance Floating pt.
 - Business – transactional support/decimal arithmetic
 - General Purpose – balanced performance for a range of tasks
- Level of software compatibility
 - PL level
 - Flexible, Need new compiler, portability an issue
 - Binary level (x86 architecture)
 - Little flexibility, Portability requirements minimal
- OS requirements
 - Address space issues, memory management, protection
- Conformance to Standards
 - Languages, OS, Networks, I/O, IEEE floating pt.

5

Computer Architecture Topics



3

Computer Systems: Technology Trends

- 1988
 - Supercomputers
 - Massively Parallel Processors
 - Mini-supercomputers
 - Minicomputers
 - Workstations
 - PC's
- 2008
 - Powerful PC's and laptops
 - Clusters delivering Petaflop performance
 - Embedded Computers
 - PDAs, I-Phones, ...

6

Technology Trends

- **Integrated circuit logic technology** – a growth in transistor count on chip of about 40% to 55% per year.
- **Semiconductor RAM** – capacity increases by 40% per year, while cycle time has improved very slowly, decreasing by about one-third in 10 years. Cost has decreased at rate about the rate at which capacity increases.
- **Magnetic disc technology** – in 1990's disk density had been improving 60% to 100% per year, while prior to 1990 about 30% per year. Since 2004, it dropped back to 30% per year.
- **Network technology** – Latency and bandwidth are important. Internet infrastructure in the U.S. has been doubling in bandwidth every year. High performance Systems Area Network (such as InfiniBand) delivering continuous reduced latency.

7

Growth in Performance of RAM & CPU

Figure 5.2

- Mismatch between CPU performance growth and memory performance growth!!
- And, almost unchanged memory latency
- Little instruction-level parallelism left to exploit efficiently
- Maximum power dissipation of air-cooled chips reached

10

Why Such Change in 20 years?

- Performance
 - Technology Advances
 - CMOS (complementary metal oxide semiconductor) VLSI dominates older technologies like TTL (Transistor Transistor Logic) in cost **AND** performance
 - Computer architecture advances improves low-end
 - RISC, pipelining, superscalar, RAID, ...
- Price: Lower costs due to ...
 - Simpler development
 - CMOS VLSI: smaller systems, fewer components
 - Higher volumes
 - Lower margins by class of computer, due to fewer services

8

Cost of Six Generations of DRAMs

11

Growth in Microprocessor Performance

Figure 1.1

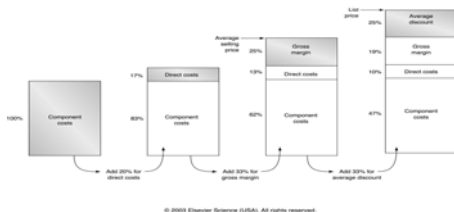
In 90's, the main source of innovations in computer design has come from **RISC-style pipelined** processors. In the last several years, the annual growth rate is (only) 10-20%.

9

Cost of Microprocessors

12

Components of Price for a \$1000 PC



13

Performance and Cost

Plane	DC to Paris	Speed	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concorde	3 hours	1350 mph	132	178,200

- **Time to run the task (ExTime)**
 - Execution time, response time, latency
- **Tasks per day, hour, week, sec, ns ... (Performance)**
 - Throughput, bandwidth

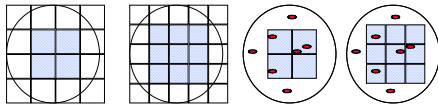
16

Integrated Circuits Costs

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} \cdot \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \cdot (\text{Wafer_diam} / 2)^2}{\text{Die Area}} - \frac{\pi \cdot \text{Wafer_diam}}{2 \cdot \text{Die Area}} - \text{Test dies}$$



$$\text{Die Yield} = \text{Wafer yield} \cdot \left\{ 1 + \frac{\text{Defects_per_unit_area} \cdot \text{Die_Area}}{\alpha} \right\}^{-\alpha}$$

Die Cost goes roughly with die area⁴

DAP.S98.1

The Bottom Line: Performance (and Cost)

"X is n times faster than Y" means

$$\frac{\text{ExTime}(Y)}{\text{ExTime}(X)} = \frac{\text{Performance}(X)}{\text{Performance}(Y)}$$

- Speed of Concorde vs. Boeing 747
- Throughput of Boeing 747 vs. Concorde

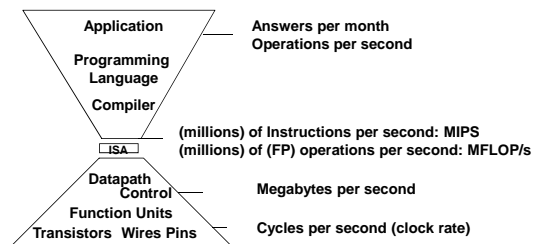
17

Failures and Dependability

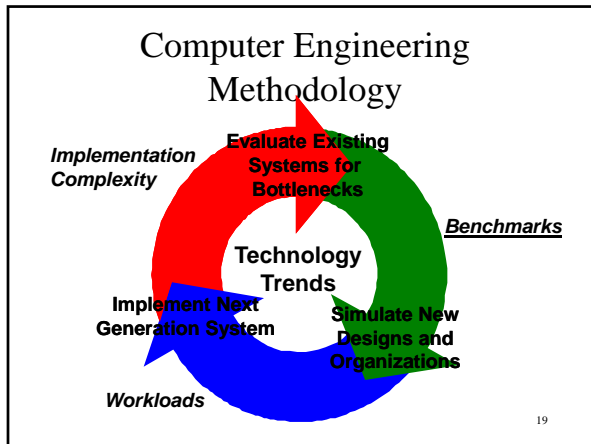
- Failures at any level costs money
 - Integrated circuits (processor, memory)
 - Disks
 - Networks
- Costs Millions of Dollars for 1hour downtime (Amazon, Google, ..)
- No concept of downtime at the middle of night
- Systems need to be designed with fault-tolerance
 - Hardware
 - Software

15

Metrics of Performance

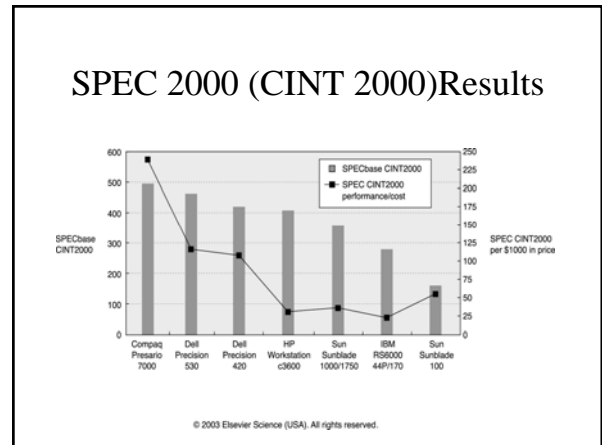


18

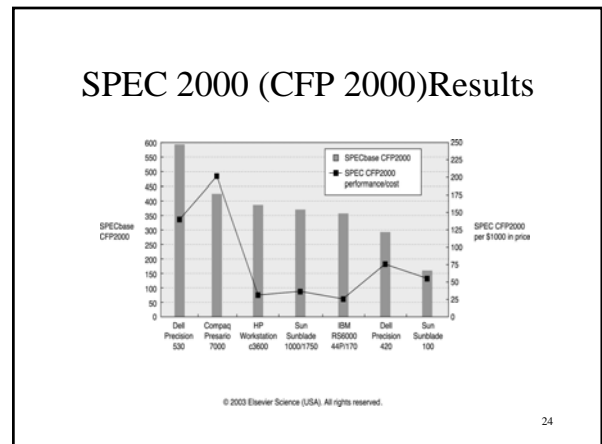


- ## SPEC: System Performance Evaluation Cooperative
- First Round 1989
 - 10 programs yielding a single number (“SPECmarks”)
 - Second Round 1992
 - SPECInt92 (6 integer programs) and SPECfp92 (14 floating point programs)
 - “benchmarks useful for 3 years”
 - SPEC CPU2000 (11 integer benchmarks – CINT2000, and 14 floating-point benchmarks – CFP2000)
 - SPEC 2006 (CINT2006, CFP2006)
 - Server Benchmarks
 - SPECWeb
 - SPECFS
 - TPC (TPA-A, TPC-C, TPC-H, TPC-W, ...)
- 22

- ## Measurement Tools
- Benchmarks, Traces, Mixes
 - Hardware: Cost, delay, area, power estimation
 - Simulation (many levels)
 - ISA, RT, Gate, Circuit
 - Queuing Theory
 - Rules of Thumb
 - Fundamental “Laws”/Principles
 - Understanding the limitations of any measurement tool is crucial.
- 20



- ## Issues with Benchmark Engineering
- Motivated by the bottom dollar, good performance on classic suites → more customers, better sales.
 - Benchmark Engineering → Limits the longevity of benchmark suites
 - Technology and Applications → Limits the longevity of benchmark suites.
- 21



Reporting Performance Results

- Reproducibility
- → Apply them on publicly available benchmarks. Pecking/Picking order
 - Real Programs
 - Real Kernels
 - Toy Benchmarks
 - Synthetic Benchmarks

25

Performance Evaluation

- “For better or worse, benchmarks shape a field”
- Good products created when have:
 - Good benchmarks
 - Good ways to summarize performance
- Given sales is a function in part of performance relative to competition, investment in improving product as reported by performance summary
- If benchmarks/summary inadequate, then choose between improving product for real programs vs. improving product to get more sales; Sales almost always wins!
- **Execution time is the measure of computer performance!**

28

How to Summarize Performance

- Arithmetic mean (weighted arithmetic mean) tracks execution time: $\frac{\sum(T_i)}{n}$ or $\frac{\sum(W_i * T_i)}{n}$
- Harmonic mean (weighted harmonic mean) of rates (e.g., MFLOPS) tracks execution time: $\frac{n}{\sum(1/R_i)}$ or $\frac{1}{\sum(W_i/R_i)}$

26

Simulations

- When are simulations useful?
- What are its limitations, I.e. what real world phenomenon does it not account for?
- The larger the simulation trace, the less tractable the post-processing analysis.

29

How to Summarize Performance (Cont'd)

- Normalized execution time is handy for scaling performance (e.g., X times faster than SPARCstation 10)
- But do not take the arithmetic mean of normalized execution time, use the Geometric Mean = $(\text{Product}(R_i))^{1/n}$

27

Queuing Theory

- What are the distributions of arrival rates and values for other parameters?
- Are they realistic?
- What happens when the parameters or distributions are changed?

30

Quantitative Principles of Computer Design

- Make the Common Case Fast
 - Amdahl's Law
- CPU Performance Equation
 - Clock cycle time
 - CPI
 - Instruction Count
- Principles of Locality
- Take advantage of Parallelism

31

Amdahl's Law (Cont'd)

- Floating point instructions improved to run 2X; but only 10% of actual instructions are FP

$ExTime_{new} =$

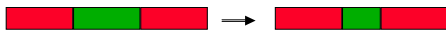
$Speedup_{overall} =$

34

Amdahl's Law

Speedup due to enhancement E:

$$Speedup(E) = \frac{ExTime_{w/o\ E}}{ExTime_{w/\ E}} = \frac{Performance_{w/\ E}}{Performance_{w/o\ E}}$$



Suppose that enhancement E accelerates a fraction F of the task by a factor S, and the remainder of the task is unaffected

DAP.S98.32

CPU Performance Equation

$$CPU\ time = \frac{Seconds}{Program} = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Organization		X	X
Technology			X

35

Amdahl's Law

$$ExTime_{new} = ExTime_{old} \times \left[(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right]$$

$$Speedup_{overall} = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

33

Cycles Per Instruction

"Average Cycles per Instruction"

$$CPI = \frac{CPU\ Time \times Clock\ Rate}{Instruction\ Count} = \frac{Cycles}{Instruction\ Count}$$

$$CPU\ time = CycleTime \times \sum_{i=1}^n CPI_i \cdot I_i$$

"Instruction Frequency"

$$CPI = \sum_{i=1}^n CPI_i \cdot F_i \quad \text{where } F_i \neq \frac{I_i}{Instruction\ Count}$$

Invest Resources where time is Spent!

36

Example: Calculating CPI

Base Machine (Reg / Reg)

Op	Freq	Cycles	CPI(i)	(% Time)
ALU	50%	1	.5	(33%)
Load	20%	2	.4	(27%)
Store	10%	2	.2	(13%)
Branch	20%	2	.4	(27%)
			<u>1.5</u>	

Typical Mix