Fondements de la conception des ordinateurs

Prof. David A. Patterson

CS252
Graduate Computer Architecture
Lecture 1

Introduction

January 22, 2002
Prof. David E Culler
Computer Science 252
Spring 2002

Plan de cours (avant)

Introduction, architecture du jeu d'instructions
performance, coût (1 ::1,2)
Introduction à la modélisation du matériel, niveaux d'abstraction (2::notes)
Pipelines (1::3)
Pipelines avancés et parallélisme au niveau d'instructions (2::4)
Mémoires cache et virtuelle (révision) (1::5)
Réseaux d'interconnexion (2::7)
Multiprocesseurs, traitement vectoriel (1::8)
Cohérence de caches, synchronisation (1::8)
Consistance de mémoire; conclusions (1::8)

Edition 3

- Ch1 - Fundamentals of Computer Design
  Ch2 - Instruction Set Principles and Examples
  Ch3 - Instruction-Level Parallelism and Its Dynamic Exploitation
  Ch4 - Exploiting Instruction-Level Parallelism with Software Approaches
  Ch5 - Memory Hierarchy Design
  Ch6 - Multiprocessors and Thread-Level Parallelism
  Ch7 - Storage Systems
  Ch8 - Interconnection Networks and Clusters
  App A - Pipelining: Basic and Intermediate Concepts
  App B - Solutions to Selected Exercises Online Appendices
  App C - A Survey of RISC Architectures for Desktop, Server, and Embedded Computers
  App D - An Alternative to RISC: The Intel 80x86
  App E - Another Alternative to RISC: The VAX Architecture
  App F - The IBM 360/370 Architecture for Mainframe Computers
  App G - Vector Processors Revised by Krste Asanovic
  App H - Computer Arithmetic by David Goldberg
  Appendix I - Implementing Coherence Protocols

Evaluation

- Travaux pratiques 30%
  - (4 et projet pour IFT6380)
- Examen intra 30%
  ME 15:30 17:30 2003-02-19 A-A 1355
- Examen final 40%

Plan

- Introduction
- Tâches d'un concepteur
- Tendances technologiques
- Tendances dans le coût
- Peformances : mesures
- Principes quantitatifs
- Conclusion

Why take CS252?

- To design the next great instruction set?...well...
  - instruction set architecture has largely converged
  - especially in the desktop / server / laptop space
  - dictated by powerful market forces
- Tremendous organizational innovation relative to established ISA abstractions
- Many New instruction sets or equivalent
  - embedded space, controllers, specialized devices, ...
- Design, analysis, implementation concepts vital to all aspects of EE & CS
  - systems, PL, theory, circuit design, VLSI, comm.
- Equip you with an intellectual toolbox for dealing with a host of systems design challenges

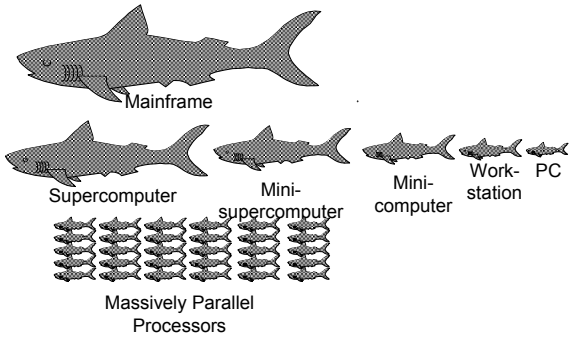
Example Hot Developments ca. 2002

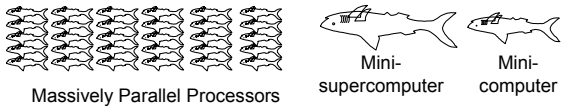
- Manipulating the instruction set abstraction
  - itanium: translate ISA64 -> micro-op sequences
  - transmeta: continuous dynamic translation of IA32
  - Xsilica: synthesize the ISA from the application
  - reconfigurable HW
- Virtualization
  - vmware: emulate full virtual machine
  - JIT: compile to abstract virtual machine, dynamically compile to host
- Parallelism
  - wide issue, dynamic instruction scheduling, EPIC
  - multithreading (SMT)
  - chip multiprocessors
- Communication
  - network processors, network interfaces
- Exotic explorations
  - nanotechnology, quantum computing

Original Food Chain Picture

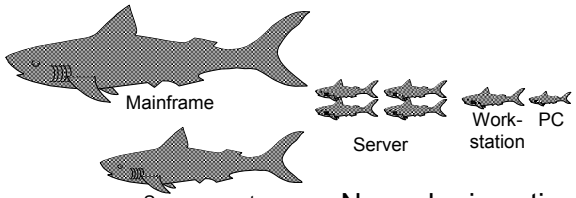
Big Fishes Eating Little Fishes

1988 Computer Food Chain





1998 Computer Food Chain

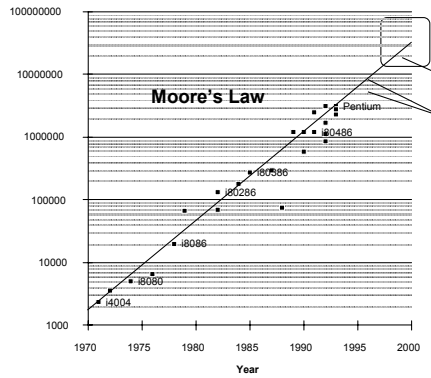


Now who is eating whom?

Why Such Change in 10 years?

- **Performance**
  - **Technology Advances**
    - » **CMOS VLSI dominates older technologies (TTL, ECL) in cost *AND* performance**
  - **Computer architecture advances improves low-end**
    - » **RISC, superscalar, RAID, …**
- **Price: Lower costs due to …**
  - **Simpler development**
    - » **CMOS VLSI: smaller systems, fewer components**
  - **Higher volumes**
    - » **CMOS VLSI : same dev. cost 10,000 vs. 10,000,000 units**
  - **Lower margins by class of computer, due to fewer services**
- **Function**
  - **Rise of networking/local interconnection technology**

Technology Trends: Microprocessor Capacity

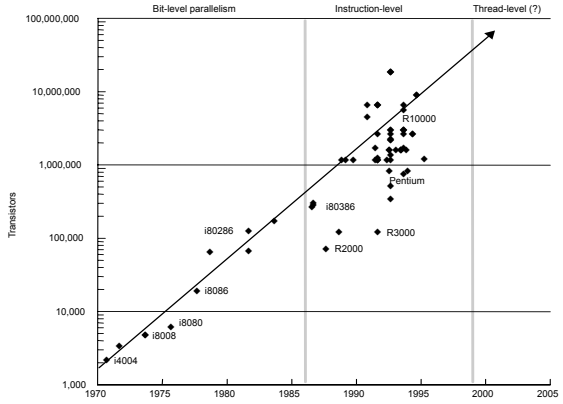


"Graduation Window"

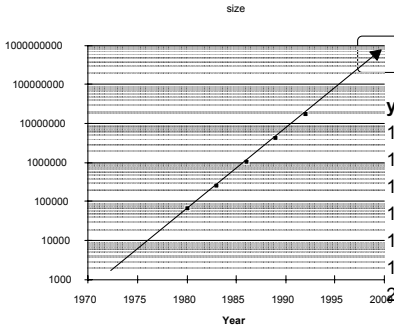
Alpha 21264: 15 million
Pentium Pro: 5.5 million
PowerPC 620: 6.9 million
Alpha 21164: 9.3 million
Sparc Ultra: 5.2 million

CMOS improvements:
- **Die size: 2X every 3 yrs**
- **Line width: halve / 7 yrs**

A take on Moore's Law



Memory Capacity (Single Chip DRAM)

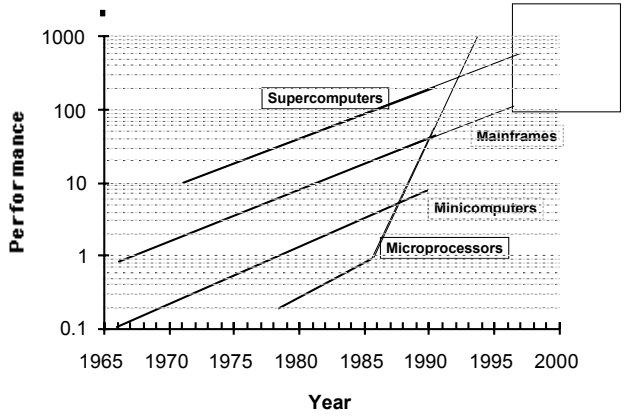


year	size(Mb)	cyc time
1980	0.0625	250 ns
1983	0.25	220 ns
1986	1	190 ns
1989	4	165 ns
1992	16	145 ns
1996	64	120 ns
2000	256	100 ns

Technology Trends (Summary)

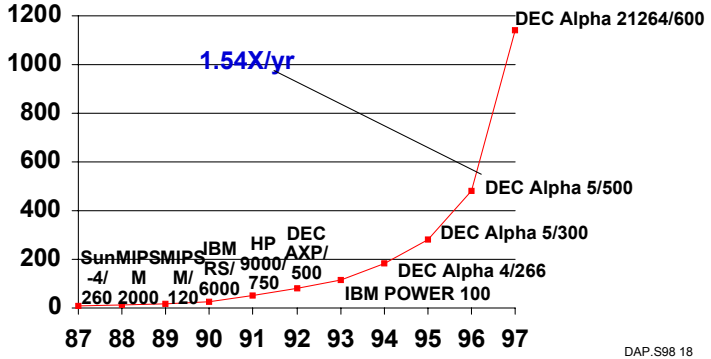
	Capacity	Speed (latency)
Logic	**2x in 3 years**	**2x in 3 years**
DRAM	**4x in 3 years**	**2x in 10 years**
Disk	**4x in 3 years**	**2x in 10 years**

Processor Performance Trends

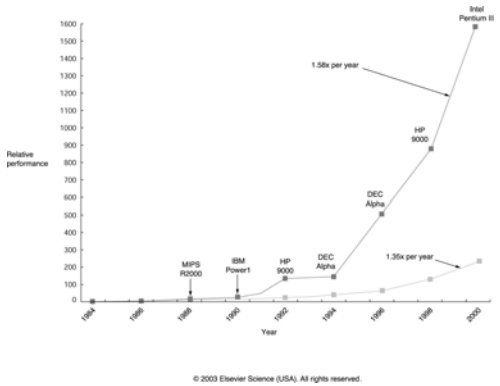


DAP.S98 17

Processor Performance (1.35X before, 1.55X now)



DAP.S98 18

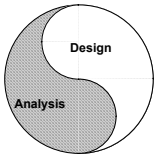


Performance Trends (Summary)

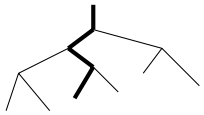
- Workstation performance (measured in Spec Marks) improves roughly 50% per year (2X every 18 months)
- Improvement in cost performance estimated at 70% per year

DAP.S98 20

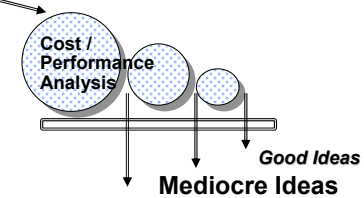
Measurement and Evaluation



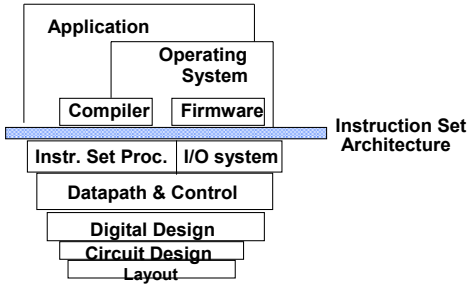
Architecture is an iterative process:
- Searching the space of possible designs
- At all levels of computer systems



Creativity

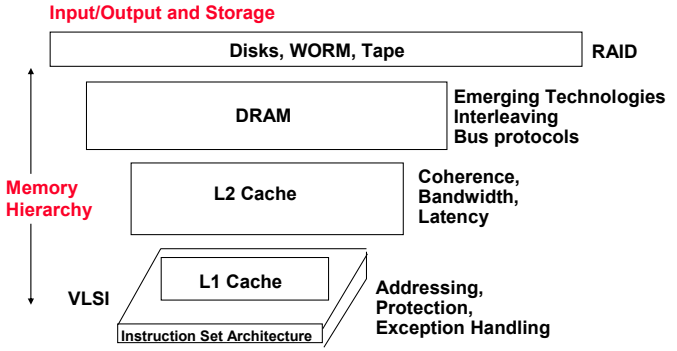


What is "Computer Architecture"?

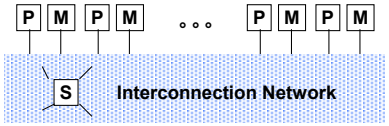


- Coordination of many *levels of abstraction*
- Under a rapidly changing set of forces
- Design, Measurement, *and* Evaluation

Computer Architecture Topics



Computer Architecture Topics



Processor-Memory-Switch

Multiprocessors
Networks and Interconnections

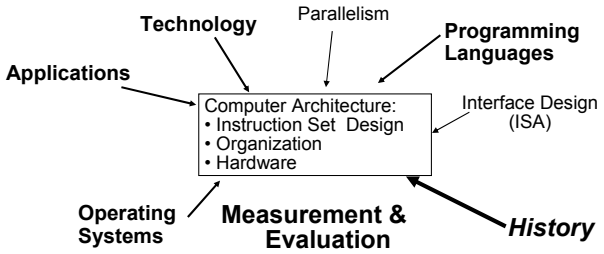
Shared Memory,
Message Passing,
Data Parallelism

Network Interfaces

Topologies,
Routing,
Bandwidth,
Latency,
Reliability

Course Focus

Understanding the design techniques, machine structures, technology factors, evaluation methods that will determine the form of computers in 21st Century

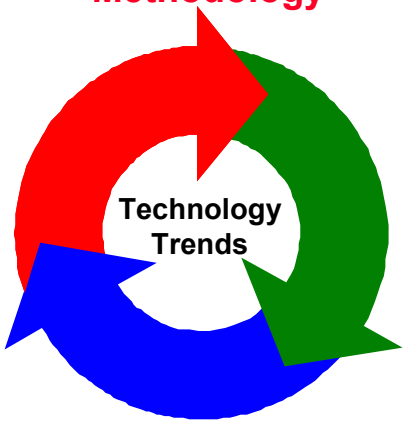


Topic Coverage

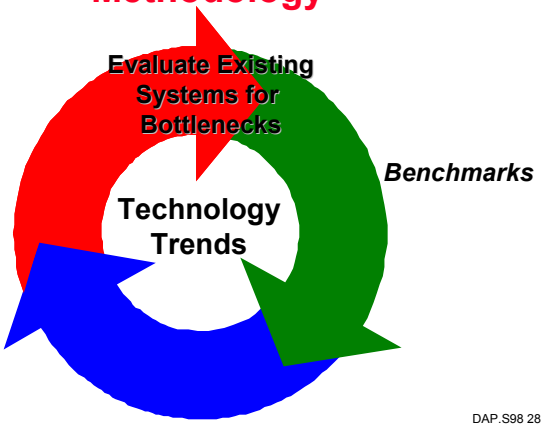
Textbook: Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 2nd Ed., 1996.

- **1.5 weeks Review: Fundamentals of Computer Architecture (Ch. 1), Instruction Set Architecture (Ch. 2), Pipelining (Ch. 3)**
- **1 week: Pipelining and Instructional Level Parallelism (Ch. 4)**
- **2.5 weeks: Vector Processors and DSPs (Appendix B)**
- **1 week: Memory Hierarchy (Chapter 5)**
- **1.5 weeks: Input/Output and Storage (Chapter 6)**
- **1.5 weeks: Networks and Interconnection Technology (Chapter 7)**
- **1.5 weeks: Multiprocessors (Ch. 8 + Culler book draft Chapter 1)**
- **Research Guest Lectures: Reconfigurable MPer("BRASS"), DRAM+MPer("IRAM"), Systems of Systems ("Millennium")**

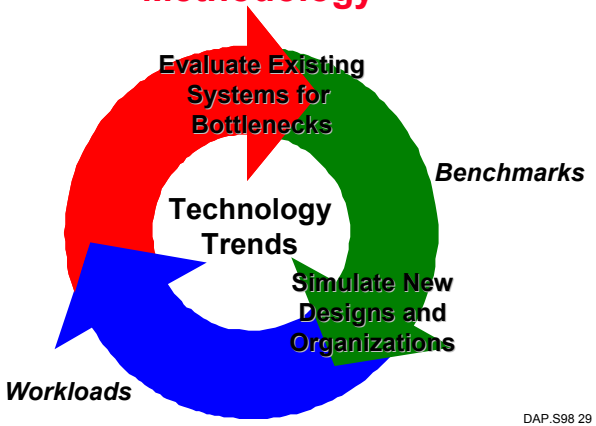
Computer Engineering Methodology



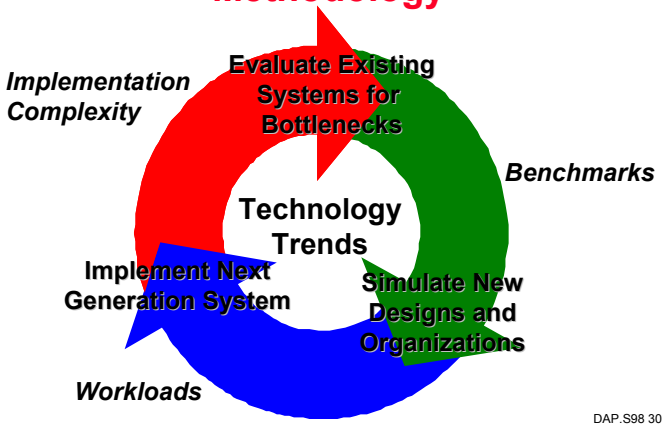
Computer Engineering Methodology



Computer Engineering Methodology



Computer Engineering Methodology



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

The Bottom Line: Performance (and Cost)

Plane	DC to Paris	Speed	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concodre	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

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**

Amdahl's Law

Speedup due to enhancement E:

$$\text{Speedup(E)} = \frac{\text{ExTime w/o E}}{\text{ExTime w/ E}} = \frac{\text{Performance w/ E}}{\text{Performance w/o E}}$$

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

Amdahl's Law

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

$$\text{Speedup}_{overall} = \frac{\text{ExTime}_{old}}{\text{ExTime}_{new}} = \frac{1}{(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

Amdahl's Law

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

$\text{ExTime}_{new} =$

$\text{Speedup}_{overall} =$

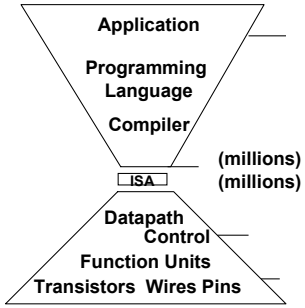
Amdahl's Law

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

$$ExTime_{new} = ExTime_{old} \times (0.9 + .1/2) = 0.95 \times ExTime_{old}$$

$$Speedup_{overall} = \frac{1}{0.95} = 1.053$$

Metrics of Performance



Answers per month
Operations per second

(millions) of Instructions per second: MIPS
(millions) of (FP) operations per second: MFLOP/s

Megabytes per second

Cycles per second (clock rate)

Aspects of CPU Performance

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

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

Cycles Per Instruction

"Average Cycles per Instruction"

CPI = (CPU Time * Clock Rate) / Instruction Count
= Cycles / Instruction Count

$$\text{CPU time} = \text{Cycle Time} * \sum_{i=1}^{n} CPI_i * IC_i$$

"Instruction Frequency"

$$CPI = \sum_{i=1}^{n} CPI_i * F_i \quad \text{where} \quad F_i = \frac{IC_i}{\text{Instruction Count}}$$

Invest Resources where time is Spent!

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%)
			1.5	

Typical Mix

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)
    - » Compiler Flags unlimited. March 93 of DEC 4000 Model 610:

```
spice: unix.c:/def=(sysv,has_bcopy,"bcopy(a,b,c)=
       memcpy(b,a,c)"
wave5: /ali=(all,dcom=nat)/ag=a/ur=4/ur=200
nasa7: /norecu/ag=a/ur=4/ur2=200/lc=blas
```

- **Third Round 1995**
  - new set of programs: SPECint95 (8 integer programs) and SPECfp95 (10 floating point)
  - "benchmarks useful for 3 years"
  - Single flag setting for all programs: SPECint_base95, SPECfp_base95

How to Summarize Performance

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

Temps d'execution

	A	B	C
P1	1	10	20
P2	1000	100	20
Total	1001	110	40

Qui est plus rapide que qui?

Moyenne arith. pondérée

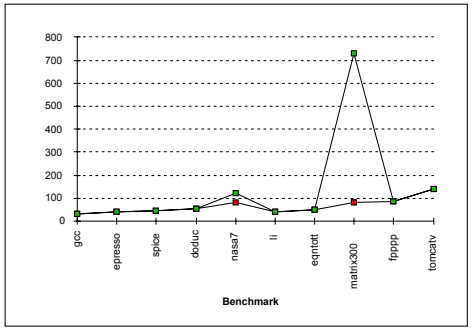
	A	B	C
P1	1.00	10.00	20.00
P2	1000.00	100.00	20.00
Moy (0.5-0.5)	500.5	55.00	20.00
Moy (.909,.091)	91.91	18.19	20
Moy (.999,.001)	2.00	10.09	20.00

Normalisation

	Normalisé à A			Normalisé à B			Normalisé à C		
	A	B	C	A	B	C	A	B	C
P1	1	10	20	0.1	1	2	.05	.5	1
P2	1	0.1	.02	10	1	.2	50	5	1
MA	1	5.05	10.01	5.05	1	1.1	25.03	2.75	1
MG	1	1	0.63	1	1	0.63	1.58	1.58	1
Tps Tot	1	0.11	0.04	9.1	1	0.36	25.03	2.75	1

SPEC First Round

- One program: 99% of time in single line of code
- New front-end compiler could improve dramatically



Impact of Means on SPECmark89 for IBM 550

	Ratio to VAX:		Time:		Weighted Time:	
Program	*Before*	*After*	*Before*	*After*	*Before*	*After*
gcc	30	29	49	51	8.91	9.22
espresso	35	34	65	67	7.64	7.86
spice	47	47	510	510	5.69	5.69
doduc	46	49	41	38	5.81	5.45
nasa7	78	144	258	140	3.43	1.86
li	34	34	183	183	7.86	7.86
eqntott	40	40	28	28	6.68	6.68
matrix300	78	730	58	6	3.43	0.37
fpppp	90	87	34	35	2.97	3.07
tomcatv	33	138	20	19	2.01	1.94
Mean	54	72	124	108	54.42	49.99
		Geometric			*Arithmetic*	*Weighted Arith.*
	Ratio	1.33	Ratio	1.16	Ratio	1.09

Performance Evaluation

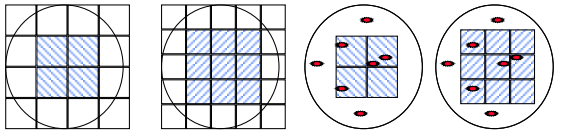
- "For better or worse, benchmarks shape a field"
- Good products created when have:
  - Good benchmarks
  - Good ways to summarize performance
- Given sales is a function in part of performance relative to competition, investment in improving product as reported by performance summary
- If benchmarks/summary 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!

Integrated Circuits Costs

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

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

$$\text{Dies per wafer} = \frac{\pi * (\text{Wafer\_diam} / 2)^2}{\text{Die Area}} - \frac{\pi * \text{Wafer\_diam}}{\sqrt{2 * \text{Die Area}}} - \text{Test dies}$$



$$\text{Die Yield} = \text{Wafer yield} * \left\{1 + \frac{\text{Defects\_per\_unit\_area} * \text{Die\_Area}}{\alpha}\right\}^{-\alpha}$$

Die Cost goes roughly with die area^4

Real World Examples

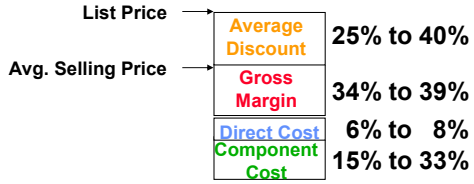
Chip	Metal layers	Line width	Wafer cost	Defect /cm^2	Area mm^2	Dies/ wafer	Yield	Die Cost
386DX	2	0.90	$900	1.0	43	360	71%	$4
486DX2	3	0.80	$1200	1.0	81	181	54%	$12
PowerPC 601	4	0.80	$1700	1.3	121	115	28%	$53
HP PA 7100	3	0.80	$1300	1.0	196	66	27%	$73
DEC Alpha	3	0.70	$1500	1.2	234	53	19%	$149
SuperSPARC	3	0.70	$1700	1.6	256	48	13%	$272
Pentium	3	0.80	$1500	1.5	296	40	9%	$417

– From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15

Cost/Performance

What is Relationship of Cost to Price?

- Component Costs
- Direct Costs (add 25% to 40%) recurring costs: labor, purchasing, scrap, warranty
- Gross Margin (add 82% to 186%) nonrecurring costs: R&D, marketing, sales, equipment maintenance, rental, financing cost, pretax profits, taxes
- Average Discount to get List Price (add 33% to 66%): volume discounts and/or retailer markup



Calcul des coûts

coût composante		100	33%
coûts directs	25%	125	8%
coûts non-recurrents	82%	228	34%
marge d'escompte	33%	303	25%

	Marge	Prix	%P.V.
coût composante		100	15%
coût direct	40%	140	6%
coûts non-recurrents	186%	400	39%
marge d'escompte	66%	665	40%

DAP.S98 53

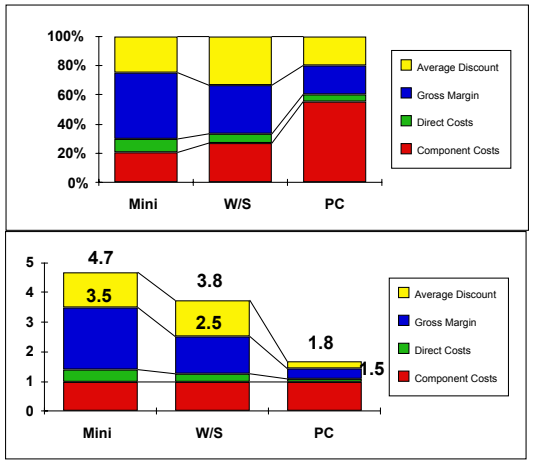
Chip Prices (August 1993)

- Assume purchase 10,000 units

Chip	Area mm^2	Mfg. cost	Price	Multi-plier	Comment
386DX	43	$9	$31	3.4	Intense Competition
486DX2	81	$35	$245	7.0	No Competition
PowerPC 601	121	$77	$280	3.6	
DEC Alpha	234	$202	$1231	6.1	Recoup R&D?
Pentium	296	$473	$965	2.0	Early in shipments

DAP.S98 54

Summary: Price vs. Cost

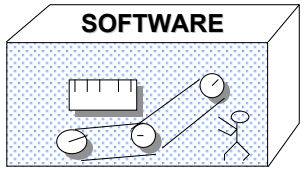


DAP.S98 55

Computer Architecture Is …

the attributes of a [computing] system as seen by the programmer, i.e., the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation.

Amdahl, Blaaw, and Brooks, 1964

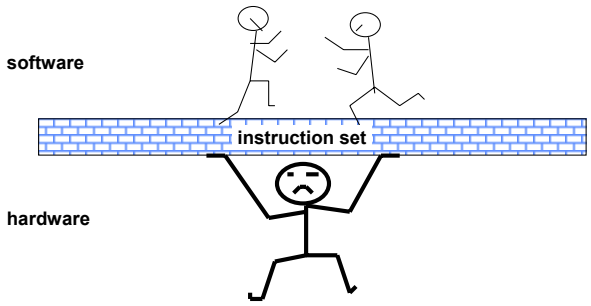


DAP.S98 56

Computer Architecture's Changing Definition

- 1950s to 1960s: Computer Architecture Course
  Computer Arithmetic
- 1970s to mid 1980s: Computer Architecture Course
  Instruction Set Design, especially ISA appropriate for compilers
- 1990s: Computer Architecture Course
  Design of CPU, memory system, I/O system, Multiprocessors

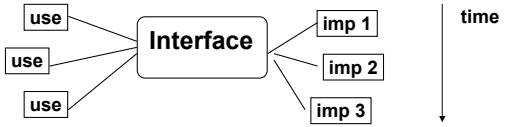
Instruction Set Architecture (ISA)



Interface Design

A good interface:

- Lasts through many implementations (portability, compatability)
- Is used in many differeny ways (generality)
- Provides convenient functionality to higher levels
- Permits an efficient implementation at lower levels



Evolution of Instruction Sets

Single Accumulator (EDSAC 1950)

Accumulator + Index Registers
(Manchester Mark I, IBM 700 series 1953)

Separation of Programming Model from Implementation

High-level Language Based
(B5000 1963)

Concept of a Family
(IBM 360 1964)

General Purpose Register Machines

Complex Instruction Sets
(Vax, Intel 432 1977-80)

Load/Store Architecture
(CDC 6600, Cray 1 1963-76)

RISC
(Mips,Sparc,HP-PA,IBM RS6000, . . .1987)

Evolution of Instruction Sets

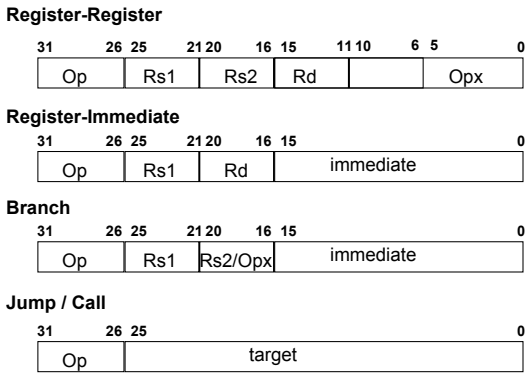
- Major advances in computer architecture are typically associated with landmark instruction set designs
  - Ex: Stack vs GPR (System 360)
- Design decisions must take into account:
  - technology
  - machine organization
  - programming languages
  - compiler technology
  - operating systems
- And they in turn influence these

A "Typical" RISC

- 32-bit fixed format instruction (3 formats)
- 32 32-bit GPR (R0 contains zero, DP take pair)
- 3-address, reg-reg arithmetic instruction
- Single address mode for load/store: base + displacement
  - no indirection
- Simple branch conditions
- Delayed branch

see: SPARC, MIPS, HP PA-Risc, DEC Alpha, IBM PowerPC, CDC 6600, CDC 7600, Cray-1, Cray-2, Cray-3

Example: MIPS



Summary, #1

- Designing to Last through Trends

	Capacity	Speed
Logic	2x in 3 years	2x in 3 years
DRAM	4x in 3 years	2x in 10 years
Disk	4x in 3 years	2x in 10 years

- 6yrs to graduate => 16X CPU speed, DRAM/Disk size
- Time to run the task
  - Execution time, response time, latency
- Tasks per day, hour, week, sec, ns, …
  - Throughput, bandwidth
- "X is n times faster than Y" means

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

Summary, #2

- Amdahl's Law:

$$\text{Speedup}_{\text{overall}} = \frac{\text{ExTime}_{\text{old}}}{\text{ExTime}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

- CPI Law:

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

- Execution time is the REAL measure of computer performance!
- Good products created when have:
  - Good benchmarks, good ways to summarize performance
- Die Cost goes roughly with die area^4
- Can PC industry support engineering/research investment?