Welcome to Computer Architecture

Credit for the Lecture Slides

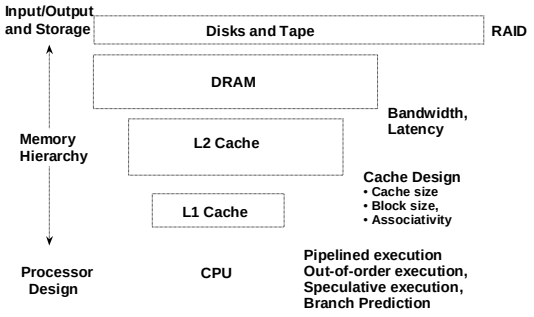
- Professor Sang Lyul Min of Seoul National University
- Professor David A. Patterson of UC Berkeley
- Professor Kevin Jeffay of U North Carolina
- Professor Ashok K. Agrawala of UMCP

Goal

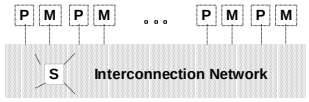
- **Understand**
  - Engineering methodology
  - Design techniques
  - Correctness criteria
  - Evaluation methods
  - Technology trends

  involved in the design of computer systems

Design Techniques



Design Techniques



Topologies, Routing, Bandwidth, Latency,

Message passing, shared memory, coherence, synchronization

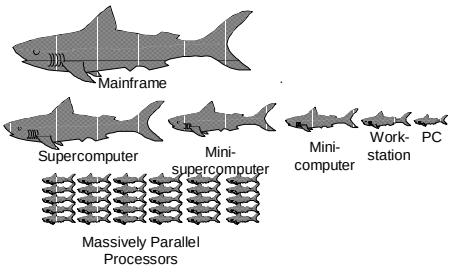
Technology Trends

1965	1977	1998
		
IBM System 360/50 0.15 MIPS 64 KB $1M $6.6M per MIPS $16M per MB	DEC VAX11/780 1 MIPS (reported) 0.5 MIPS (actual) 1 MB $200K $200K to $400 per MIPS $200K per MB	Apple iMac 700 MIPS (peak) 427 MIPS (estimated) 32 MB $1229 (September 1998) $1.75 to $2.90 per MIPS $38 per MB

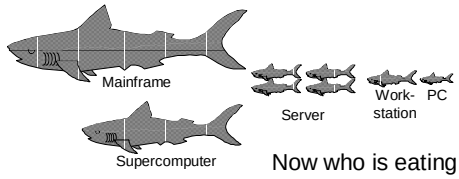
Original Food Chain Picture

Big Fishes Eating Little Fishes

1988 Computer Food Chain



1998 Computer Food Chain

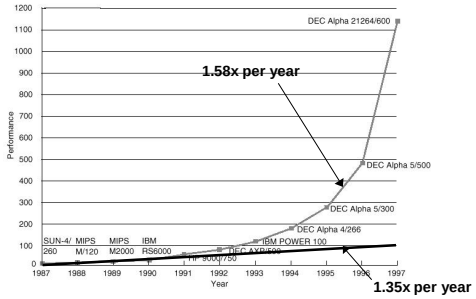


Now who is eating whom?

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



Processor performance: 1.58x per year

How/why has this been happening?

Understanding the trend

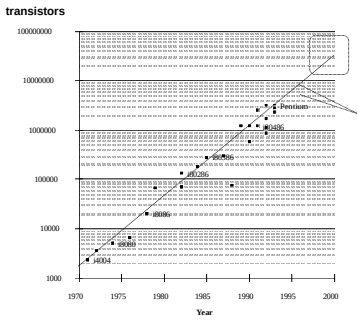
- **Hardware technology**
- **Software technology**
- **Application characteristics**

➔ **foresee the future**

Computer Usage Trends

- **Increased amount of memory usage**
  - **Program and data by 1.5 – 2 times a year**
- **Replace assembly language with high-level language**
  - **Role of compiler**

Integrated Circuit Logic Technology



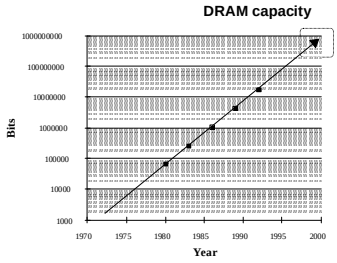
Pentium Pro: 5.5 million
PowerPC 620: 6.9 million
Alpha 21164: 9.3 million
Sparc Ultra: 5.2 million

Transistor count: 1.60x - 1.80x per year

Transistor density: 1.50x per year

Die size: 1.10x - 1.25x per year

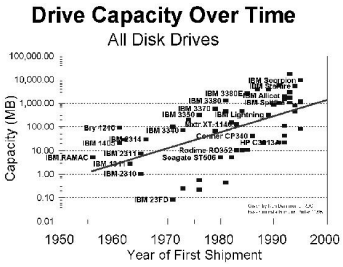
Single chip DRAM Technology



year	size	cycle time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1995	64 Mb	125 ns

DRAM density: 1.60x per year (4x in three years)

Magnetic Disk Technology



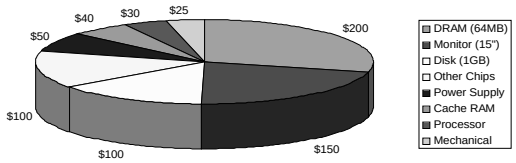
Disk density: 1.50x - 1.60x per year (4x in three years)

Cost and Performance

- **Desire for increased performance for low cost**
  - **Everyday sight**

➔ **impact on computer system architecture as well**

Cost Components



Processor	30	4%
DRAM (64MB)	200	29%
Cache RAM	40	6%
Other Chips	100	14%
Power Supply	50	7%
Disk (1GB)	100	14%
Mechanical	25	4%
Monitor (15")	150	22%
Total	695	100%

Effect on Cost

- **Yield**
  - **Percentage of manufactured devices that survives the testing procedure**
  - **Learning curve : manufacturing cost decreases over time**
- **Volume**
  - **Commodities**
  - **Rule of thumb: doubling volume decreases cost by 10%**

Chip Cost

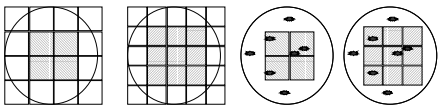
- **Chip cost is primarily a function of die area**
  - **increases much faster than linearly due to yield**
  - **going larger gives diminishing performance returns**

Wafer Cost		2,500.00			
Wafer Diameter		200	mm		
Wafer Area		31416	mm^2		
Chip Size	Die Area	Die/Wafer	Yield	Good Die	Cost/Die
1	1	30971	1.00	30971	$0.08
5	25	1167	0.98	1145	$2.18
7.5	56.25	499	0.83	414	$6.04
10	100	269	0.63	170	$14.71
15	225	110	0.36	39	$64.10
17.5	306.25	77	0.28	21	$119.05

Chip Cost

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

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



Die Cost goes roughly with die area^4

Real World Examples

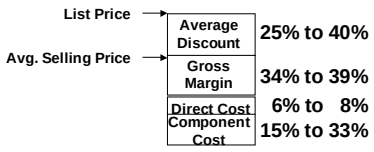
Chip	Metal layers	Line width	Wafer cost	Defect /cm^2	Area mm^2	Dies/ wafer	Yield	Die Cost
386DX	2	0.90	$900	1.0	43	360	71%	$4
486DX2	3	0.80	$1200	1.0	81	181	54%	$12
PowerPC 601	4	0.80	$1700	1.3	121	115	28%	$53
HP PA 7100	3	0.80	$1300	1.0	196	66	27%	$73
DEC Alpha	3	0.70	$1500	1.2	234	53	19%	$149
SuperSPARC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



Importance of Cost/Performance Design

- **Spectrum of systems**
  - High-performance design like supercomputers to low-cost design like IBM PC clones
  - Need to balance cost and performance

Performance

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 (ExecutionTime)**
  - **Execution time, response time, latency**
- **Tasks per day, hour, week, sec, ns … (Performance)**
  - **Throughput, bandwidth**

Performance

"X is n times faster than Y" means

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

Example: Y takes 15 seconds to complete a task, X takes 10 seconds. How many times is X faster?

Performance Terminology

"X is n% faster than Y" means:

$$\frac{\text{Execution Time(Y)}}{\text{Execution Time(X)}} = \frac{\text{Performance(X)}}{\text{Performance(Y)}} = 1 + \frac{n}{100}$$

$$n = \frac{100(\text{Performance(X)} - \text{Performance(Y)})}{\text{Performance(Y)}}$$

$$n = \frac{100(\text{Execution Time(Y)} - \text{Execution Time(X)})}{\text{Execution Time(X)}}$$

Example: Y takes 15 seconds to complete a task, X takes 10 seconds. What % faster is X?

Execution Time

- **Wall-clock, response, elapsed time**
- **CPU time**
  - **User**
  - **System**
- **But executing what?**

Benchmark Programs

1. Real programs - SPEC benchmarks
2. Kernels - Livermore Loops and Linpack
3. Toy benchmarks - Quicksort, *etc*
4. Synthetic benchmarks - Dhrystone and Whetstone

SPEC: System Performance Evaluation Cooperation

- First Round 1989
  - 10 programs yielding a single number
- Second Round 1992
  - CINT92 (6 integer programs) and CFP92 (14 floating point programs)
  - Different compiler flags are allowed for different programs
- Third Round 1995
  - CINT95 (8 integer programs) and CFP95 (10 floating point programs)
  - Same compiler flags for all programs of a given language
  - measures both execution time and throughput

Other SPEC Benchmarks

- CPU2000 – CPU Intensive Application Performance
- Web99 - WWW Server Performance
- HPC96 - High-end System Performance
- MAIL2001 – Mail server Performance
- SPECapc - Graphics System Performance

http://www.spec.org

Summarizing Performance

Arithmetic mean $\frac{1}{n}\sum_{i=1}^{n} T_i$

Harmonic mean $\frac{n}{\sum_{i=1}^{n} \frac{1}{R_i}}$ — Represents total execution time

Geometric mean $\sqrt[n]{\prod_{i=1}^{n} \frac{X_i}{Y_i}}$ — Consistent independent of reference

Quantitative Principle of Computer Design

- **Make the Common Case Fast**
  - **Favor the frequent case for the infrequent case**

How much improvement is possible?

Amdahl's Law: Speedup

Defines the speedup that can be gained using a particular feature

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement}}{\text{Performance for the entire task without it}}$$

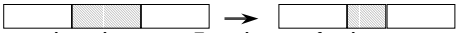
or

$$\text{Speedup} = \frac{\text{Execution time for entire task without the enhancement}}{\text{Execution time for entire task with the enhancement}}$$

Amdahl's Law: assessing enhancement

Speedup due to enhancement E:

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



Suppose that enhancement E accelerates a fraction $\text{Fraction}_{\text{enhanced}}$ of the task by a factor $\text{Speedup}_{\text{enhanced}}$, and the remainder of the task is unaffected.

What are the new execution time and the overall speedup due to the enhancement?

Amdahl's Law

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

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

Amdahl's Law

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

$ExTime_{new}$ =

$Speedup_{overall}$ =

Amdahl's Law

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

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

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

Amdahl's Law Implication on Design

Implementation of floating-point (FP) square root : square root is responsible for 20% of execution time

Two proposals:

1. Add FPSQR hardware → speed up this operation by factor of 10
2. Make all FP instructions faster; FP instructions responsible for 50% of execution time
   → make all FP instructions run 2 times faster

What is your choice?

Solution

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

$$Speedup_{FPSQR} = \frac{1}{(1 - 0.2) + \frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$

$$Speedup_{FP} = \frac{1}{(1 - 0.5) + \frac{0.5}{2}} = \frac{1}{0.75} = 1.33$$

CPU Performance

CPU Time = CPU clock cycles for a program X Clock cycle time

CPI = (CPU clock cycles for a program)/IC

CPI - Clock Cycles Per Instruction
IC - Instruction Count

CPU Time = (IC X CPI) /(Clock Rate)

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

Using the CPU Performance Equation

- **Frequency of FP operations = 25%**
- **Average CPI of FP operations = 4.0**
- **Average CPI of other instructions = 1.33**
- **Frequency of FPSQR = 2%**
- **CPI of FPSQR = 20**

- **Reduce the CPI of FPSQR to 2?**
- **Reduce average CPI of all FP operations to 2?**

- **What is your choice?**

Solution

- **Only CPI changes ➔ instruction count, clock rate remain unchanged**

- **Original CPI = (4 x 25%) + (1.33 x 75%) = 2**

- **CPI w/ new FPSQR = Original CPI – CPI saved**
  = 2.0 – 0.36 = 1.64

[CPI saved = 2% x (CPI old – CPI of new only)
= 2% x (20-2) = 0.36]

- **CPI new FP = 75% x 1.33 + 25% x 2.0 = 1.5**