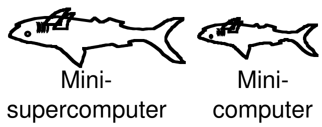
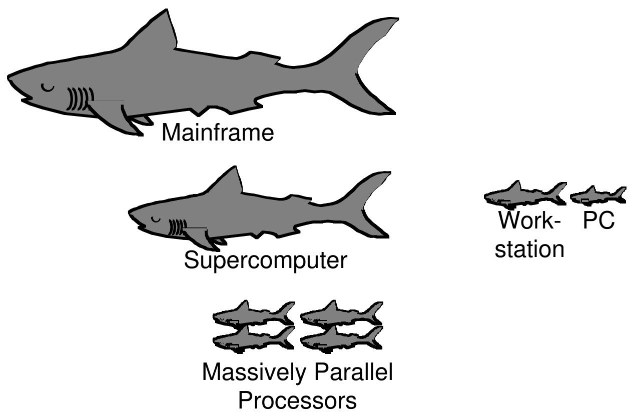
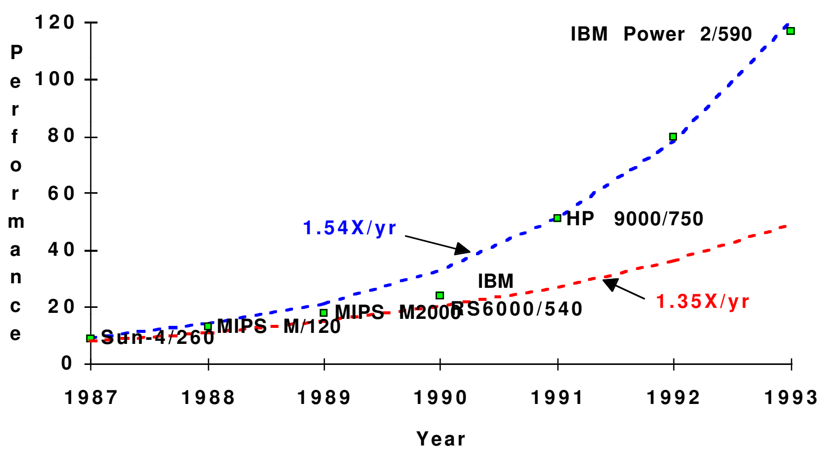


1994 Computer Food Chain



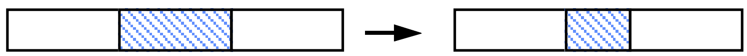
Processor Performance



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 then,

ExTime(E) =

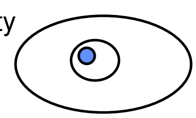
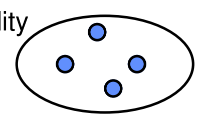
Speedup(E) =

Corollary: Make The Common Case Fast

- **All instructions require an instruction fetch, only a fraction require a data fetch/store.**

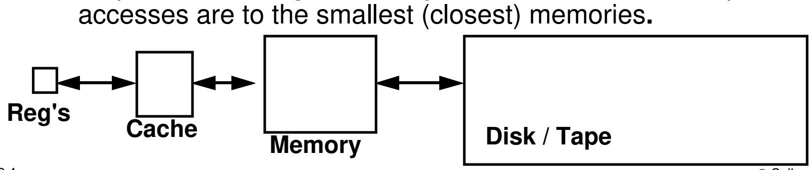
  => Optimize instruction access over data access

- **Programs exhibit *locality***

  Spatial Locality  Temporal Locality 

Access to small memories is faster

=> provide a *storage hierarchy* such that the most frequent accesses are to the smallest (closest) memories**.**



Occam's Toothbrush

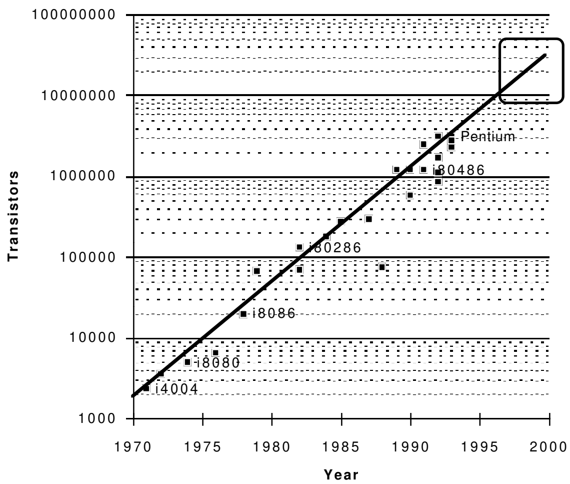
- The simple case is usually the most frequent and the easiest to optimize!

- Do simple, fast things in hardware and be sure the rest can be handled correctly in software.

Functional Requirements

- Application Area
  - Special Purpose (e.g., DSP) / General Purpose
  - Scientific (FP intensive) / Commercial (Mainframe)
- Level of Software Compatibility
  - Object Code/Binary Compatible (cost HW v. SW;360)
  - Assembly Language (dream to be different from binary)
  - Programming Language; Why not?
- Operating System Requirements for General Purp.
  - Size of address space
  - Memory management/Protection
  - Context Switch
  - Interrupts and Traps
- Standards; innovation vs. competition
  - IEEE 754 Floating Point
  - I/O Bus
  - Networks
  - Operating Systems / Programming Languages ...

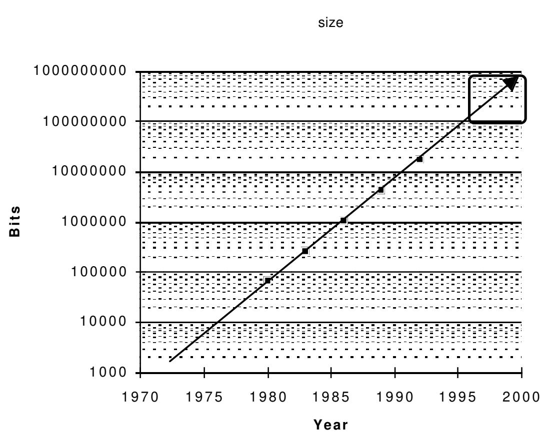
Technology Trends: Microprocessor Capacity



"Graduation Window"

CMOS improvements:
•Die size: 2x / 3 years
•Line width: halve / 7 yrs

Memory Capacity (single chip dRAM)



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	64Mb??	??

Technology Trends (summary)

	Capacity	Speed
Logic	2x in 3 years	2x in 3 years
DRAM	4x in 3 years	1.4x in 10 years
disk	4x in 3 years	1.4x in 10 years

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)

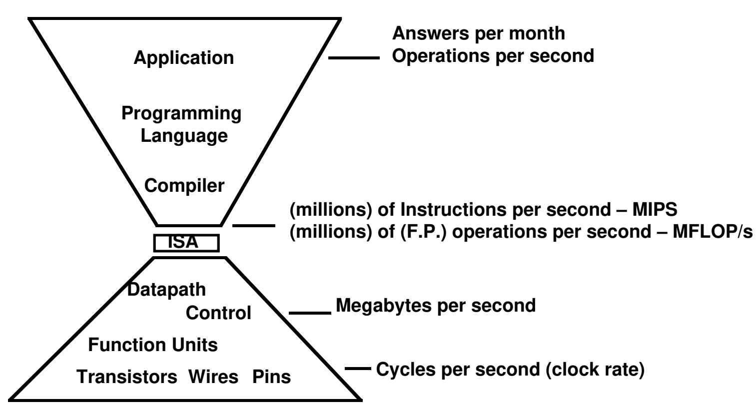
" X is n times faster than Y" means

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

- Speed of Concorde vs. Boeing 747

- Throughput of Boeing 747 vs. Concodre

Metrics of performance



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			
Compiler			
Inst. Set.			
Organization			
Technology			

Marketing Metrics

MIPS = Instruction Count / Time * 10^6

= Clock Rate / CPI * 10^6

•machines with different instruction sets ?

•programs with different instruction mixes ?

• dynamic frequency of instructions

• uncorrelated with performance

MFLOP/S = FP Operations / Time * 10^6

•machine dependent

•often not where time is spent

Normalized:

add,sub,compare,mult	1
divide, sqrt	4
exp, sin, . . .	8

CPI

"Average cycles per instruction"

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

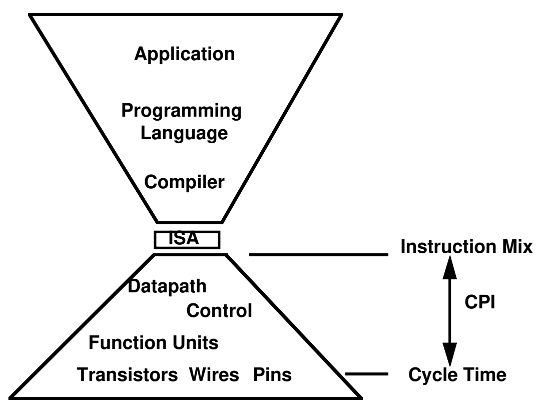
$$\text{CPU time} = \text{CycleTime} * \sum_{i=1}^{n} \text{CPI}_i * I_i$$

"instruction frequency"

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

Invest Resources where time is Spent!

Organizational Trade-offs



Example

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

Example

Base Machine (Reg / Reg)		
Op	Freq	Cycles
ALU	50%	1
Load	20%	2
Store	10%	2
Branch	20%	2

Typical Mix

Suppose we add register / memory operations that have one source operand in memory. These have a cycle count of 2. The change also causes the Branch cycle count to increase to 3.

What fraction of the loads must be eliminated (combined with ALU) for this to pay off?

Workload

- **Real programs**
  - **eg: Spec**
- **Kernels**
  - **Time critical exerpts of real programs**
  - **eg: Livermore loops**
- **(Toy) Benchmarks**
  - **10-100 line**
  - **eg: sieve, puzzle, quicksort**
- **Synthetic Benchmarks**
  - **attempt to match average frequencies of real workloads**
  - **eg: Whetstone, dhrystone**