System I

The Processor: Performance

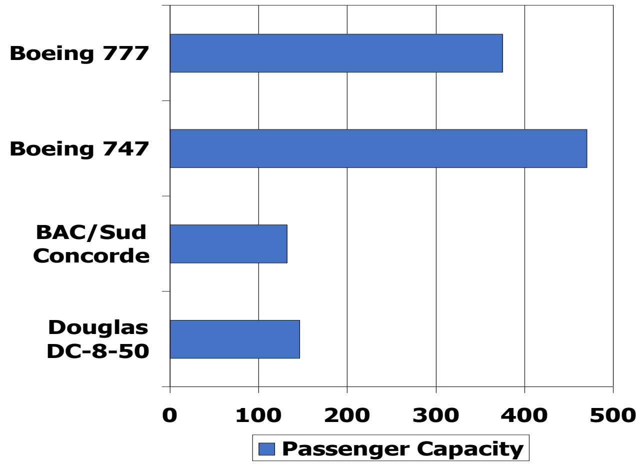
Li Lu

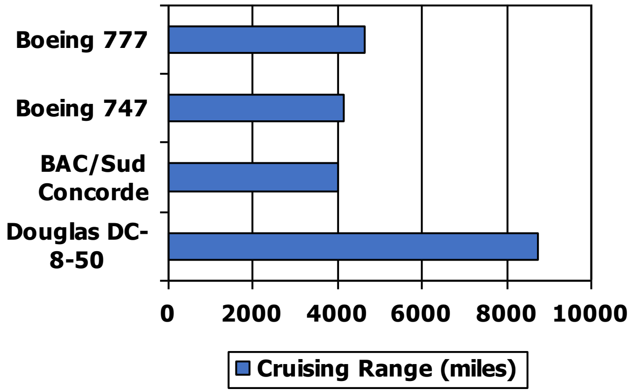
Zhejiang University

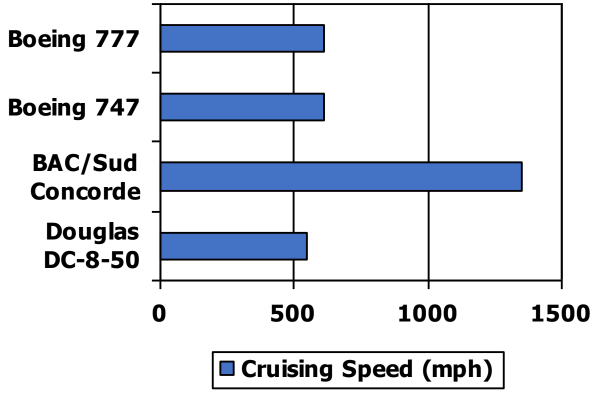
Performance?

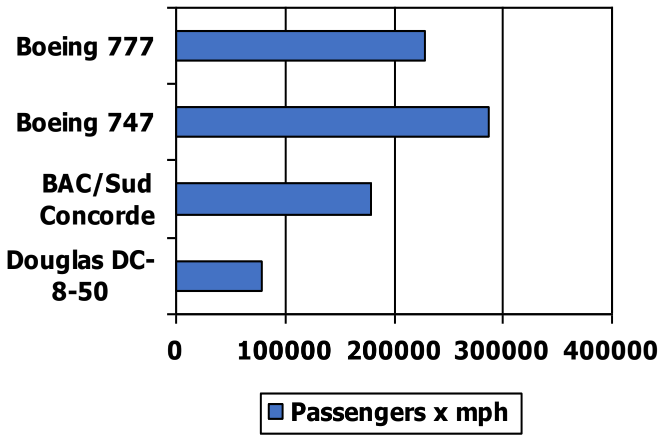
Defining Performance

- Which airplane has the best performance?









Aircraft type	Passenger Capacity	Cruising Range(miles)	Cruising Speed(mph)	Passengers *mph
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

Performance

Being able to gauge the relative performance of a computer is an important but tricky task. There are a lot of factors that can affect performance.

- Architecture
- Hardware implementation of the architecture
- Compiler for the architecture
- Operating system

Furthermore, we need to be able to define a measure of performance.

- Single users on a PC → a minimization of response time.
- Large data → a maximization of throughput

Response Time and Throughput

- Latency (Response time)
  - is the time between the start and completion of an event
  - How long it takes to do a task
- Throughput (bandwidth)
  - is the total amount of work done in a given period of time
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - Replacing the processor in a computer with a faster processor
  - Adding more processors?
- We’ll focus on program response time for now

Performance

- Define Performance = 1/Execution Time
- “X is n time faster than Y”

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A
    = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Performance

Performance has an inverse relationship to execution time.

$$Performance = \frac{1}{Execution\ Time}$$

Comparing the performance of two machines can be accomplished by comparing execution times.

$$Performance_X > Performance_Y$$

$$\Longrightarrow \quad \frac{1}{Execution_X} > \frac{1}{Execution_Y}$$

$$\Longrightarrow \quad Execution_Y > Execution_X$$

Measuring Execution Time

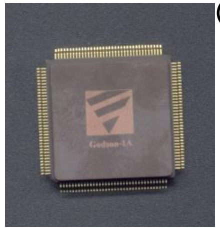
- Elapsed time
  - Total response time, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines system performance
- CPU time
  - Time spent processing a given job
    - Discounts I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time
    - User CPU time : CPU time spent in the program itself
    - System CPU time: CPU time spent in the OS, performing tasks on behalf of the program.
  - Different programs are affected differently by CPU and system performance

The main goal of architecture improvement is to improve the performance of the system

How can computers run fast?

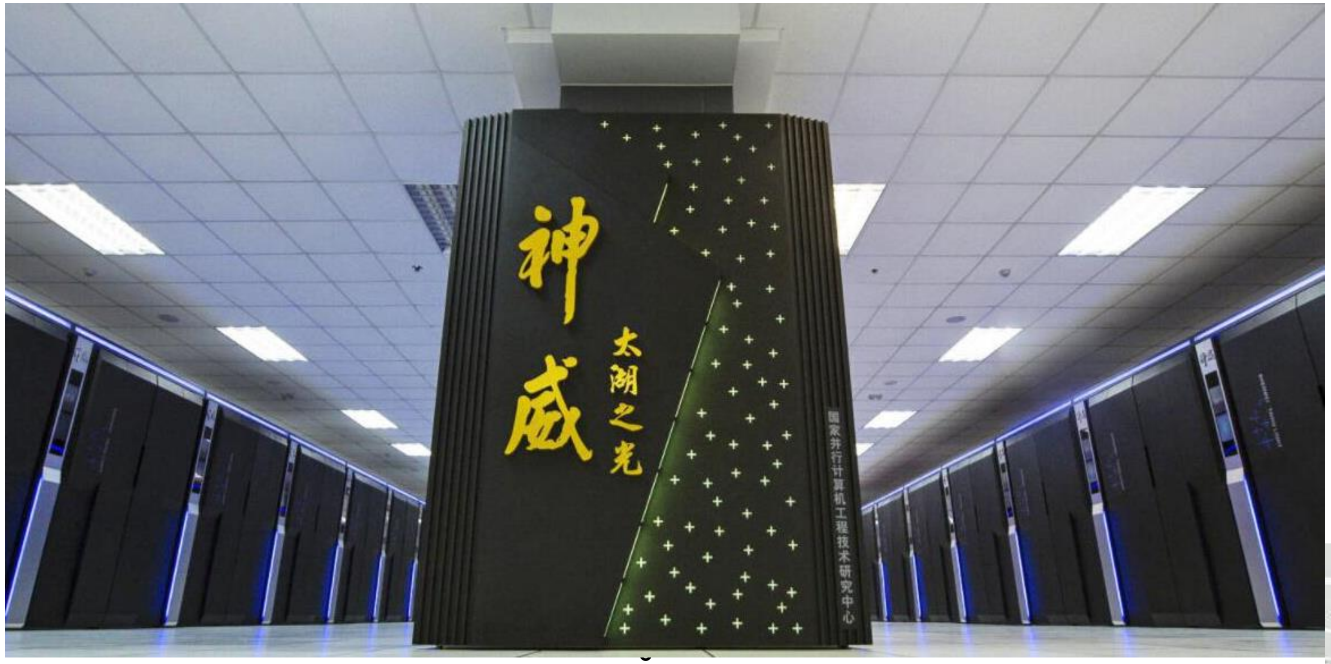
Describe a thing with the least instructions
--Algorithms, compiling

Do more things in a Clock cycle
--Architecture



Increase the speed of "core"
-- Main frequency

These big guys are strategic

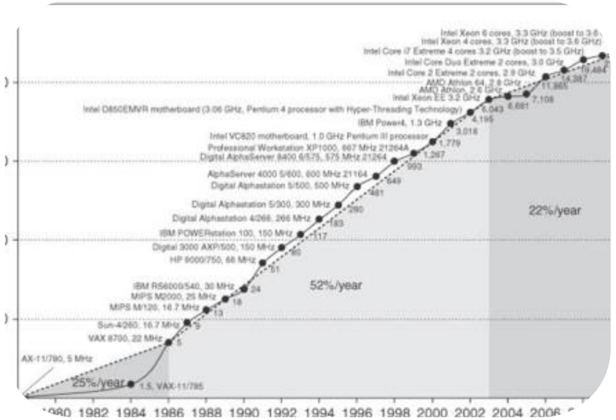


More recently

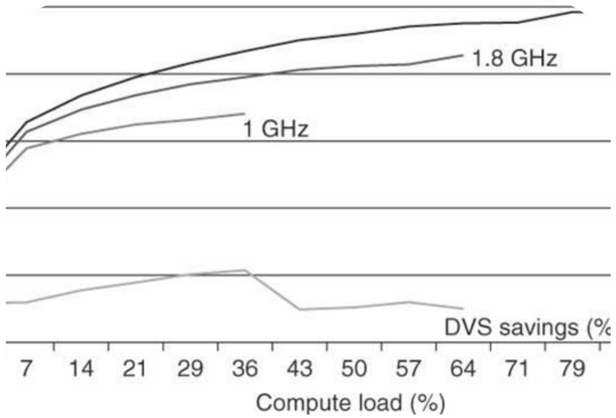
Every minute counts.
The goal is to be as fast as possible!

What does performance matters?

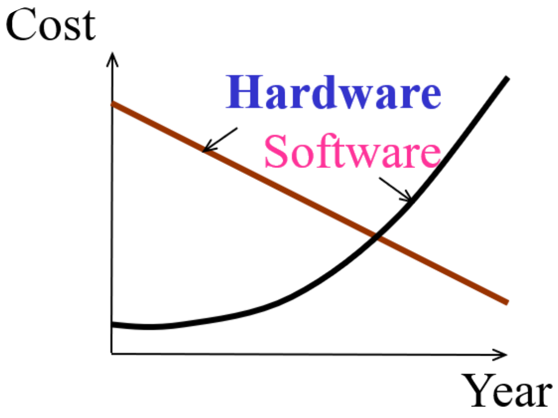
- Computers are becoming more and more common
- Ubiquitous CPU
- How can **batteries** last long?
- To make it more affordable, **price** matters



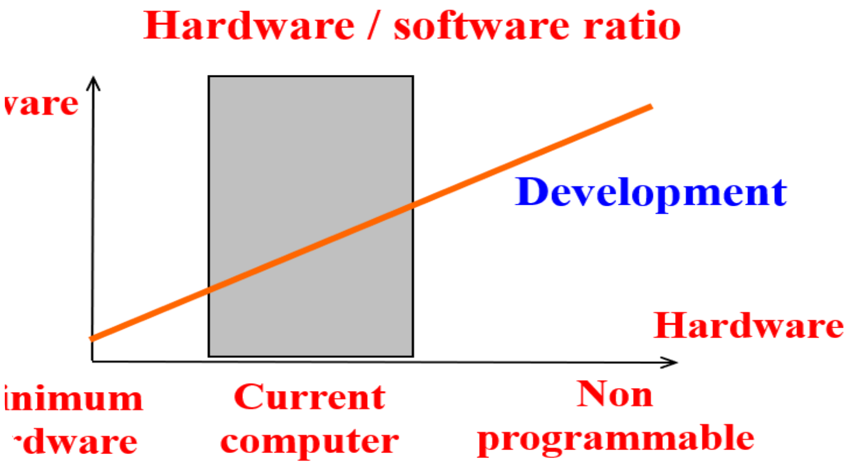
Performance trend



Power and energy consumption

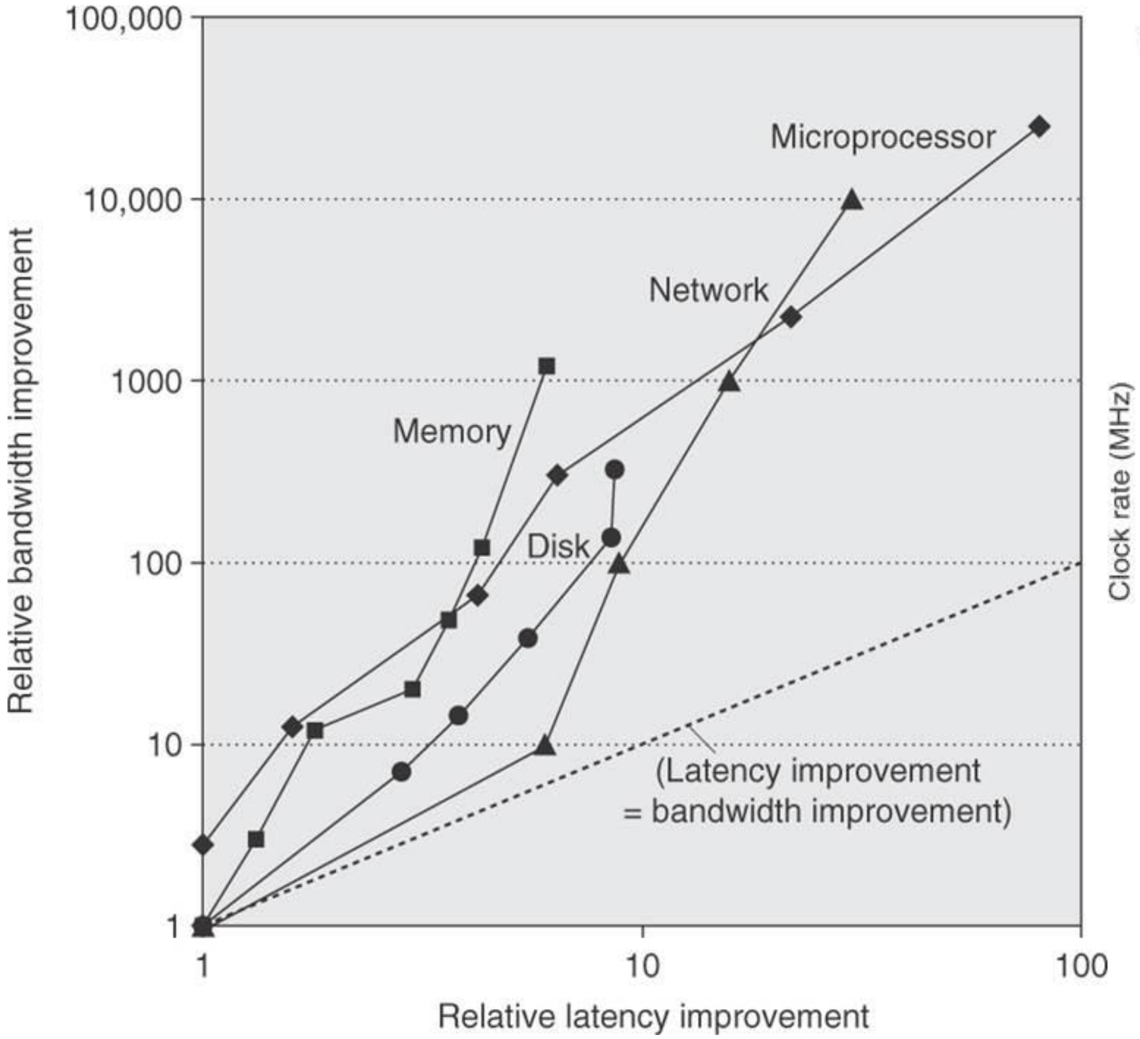


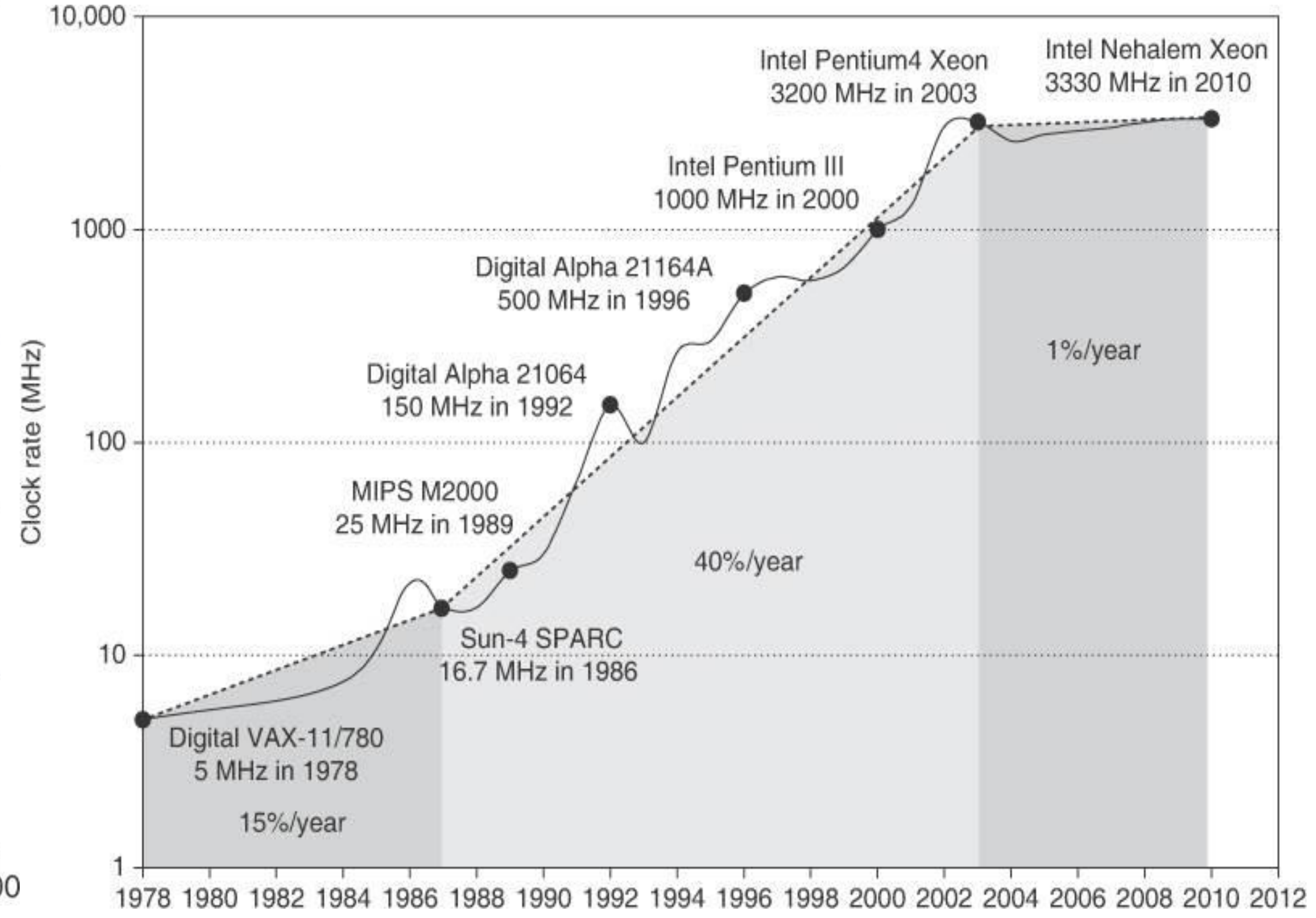
Cost trend



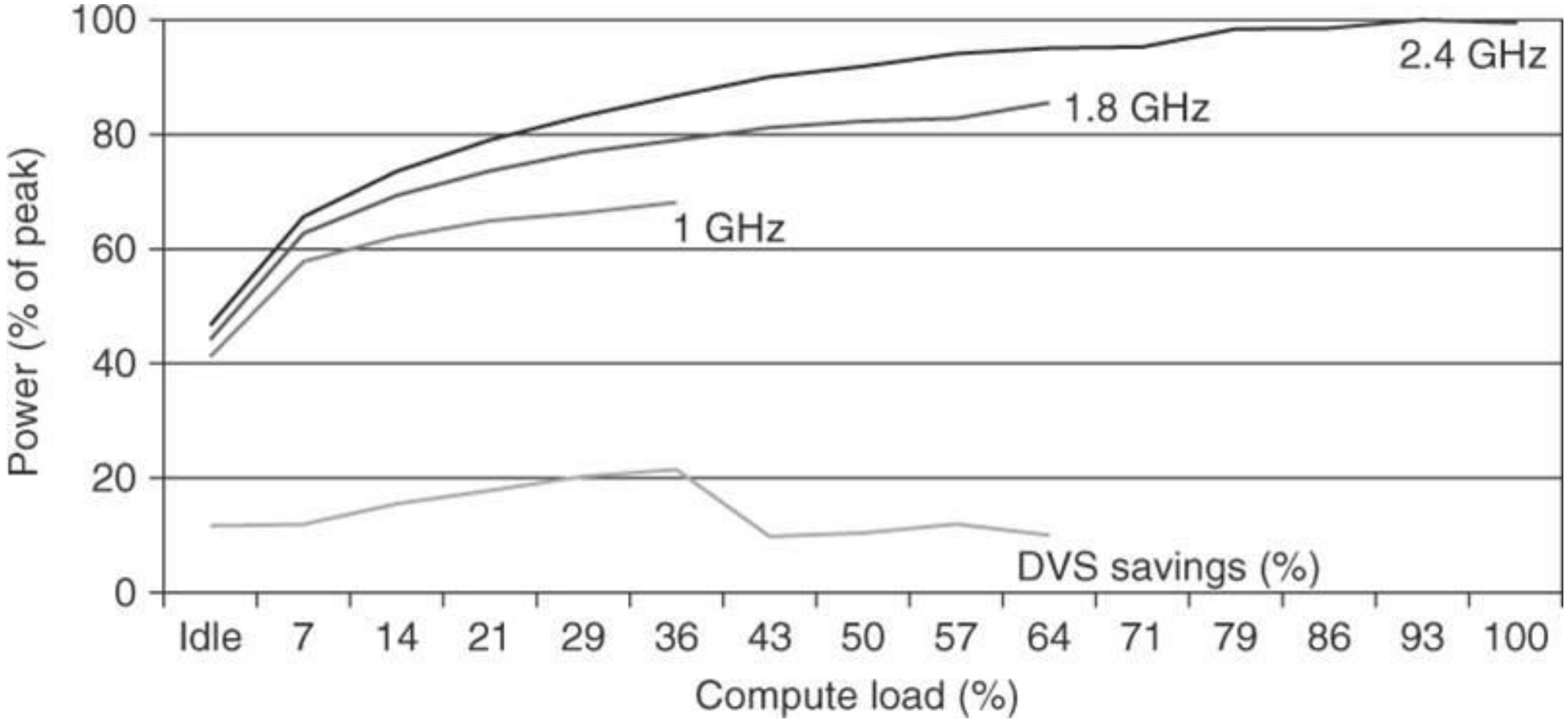
Software and hardware trend

Performance Trend

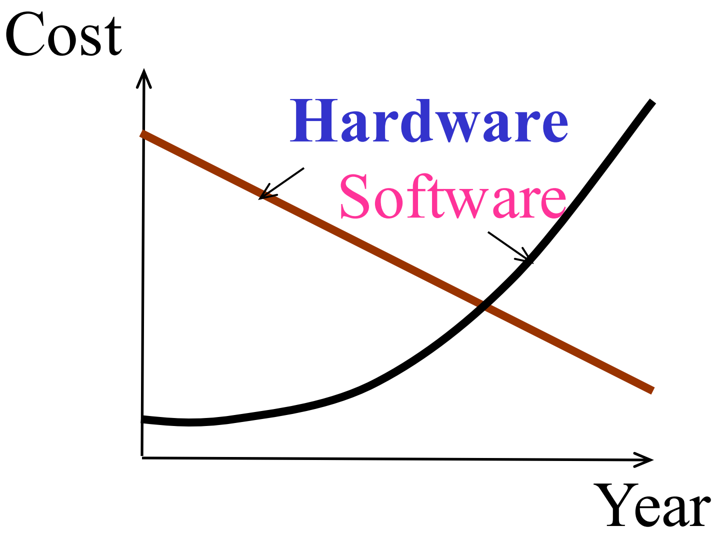


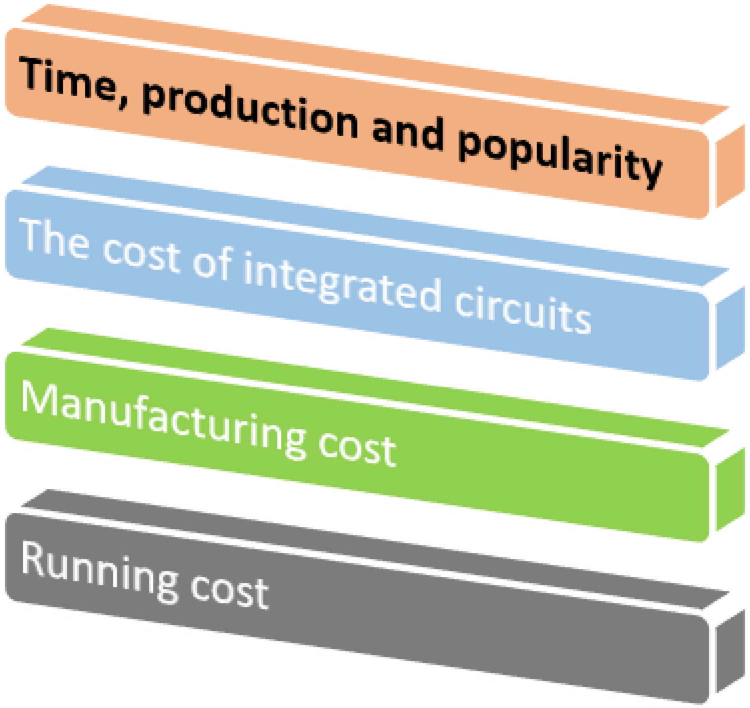


Power and energy consumption

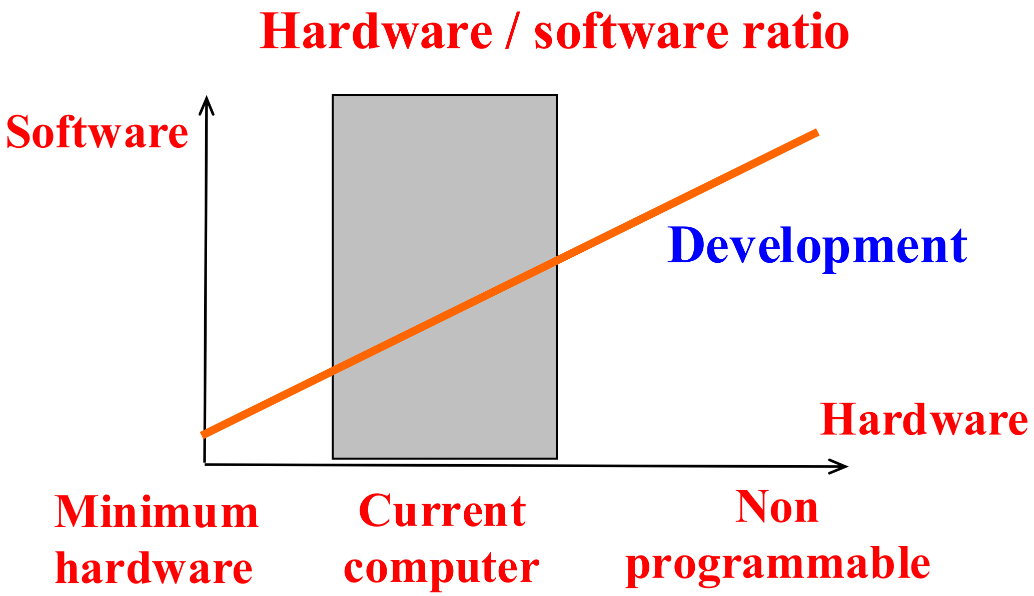


Cost Trend


Cost
Hardware
Software
Year


Time, production and popularity
The cost of integrated circuits
Manufacturing cost
Running cost

Software and Hardware Trend



- The proportion of hardware implementation is higher, and the cost of hardware is much lower

  For a computer system with the same functions, the proportion of software and hardware functions can be changed within a certain range

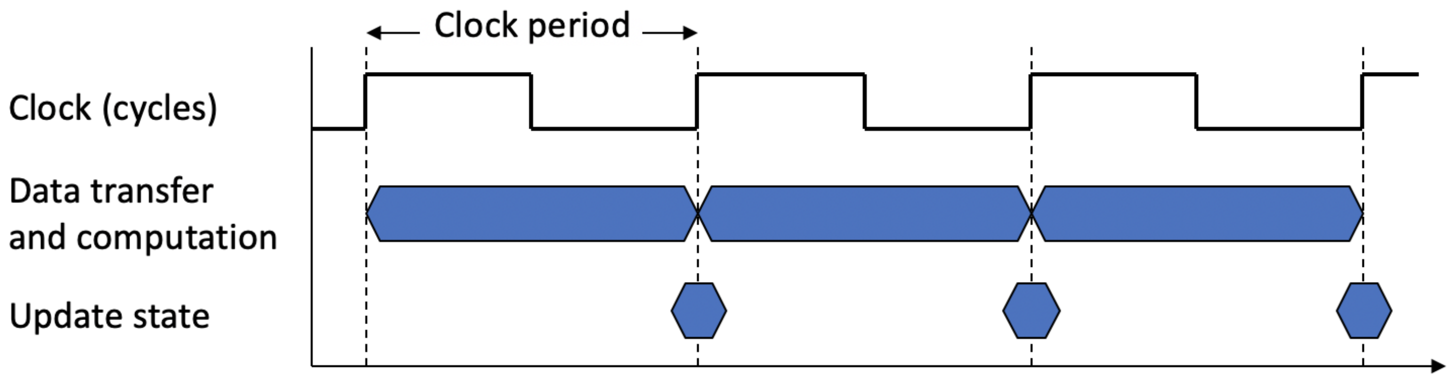
Improvement

- Improvement of input / output
- The development of memory organization structure
    - Associative memory and associated processor
    - General register group
    - Cache
- Two directions of instruction set development：
    - CISC
    - RISC
- Parallel processing technology
    - How to develop parallelism in traditional machines?
    - Develop parallel technologies at different levels
    - For example, micro-operation level, instruction level, thread level, process level, task level, etc.

CPU performance

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



- Clock period: duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Performance

In order to determine the effect of a design change on the performance experienced by the user, we can use the following relation:

$$CPU\ Execution\ Time = CPU\ Clock\ Cycles \times Clock\ Period$$

Alternatively,

$$CPU\ Execution\ Time = \frac{CPU\ Clock\ Cycles}{Clock\ Rate}$$

CPU Time

$$\begin{aligned}\text{CPU Time} &= \text{CPU Clock Cycles} \times \text{Clock Cycle Time} \\ &= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}\end{aligned}$$

- Performance **improved** by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 $\times$ clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6s}$$

$$\begin{aligned}\text{Clock Cycles}_A &= \text{CPU Time}_A \times \text{Clock Rate}_A \\ &= 10s \times 2\text{GHz} = 20 \times 10^9\end{aligned}$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6s} = \frac{24 \times 10^9}{6s} = 4\text{GHz}$$

Metrics of CPU performance

The basic metrics of performance and how each is measured.

Component	Units of Measure
CPU Execution Time for a Program	Seconds for the Program
Instruction Count	Instructions Executed for the Program
Clock Cycles per Instruction	Average Number of Clock Cycles per Instruction
Clock Cycle Time (Clock Period)	Seconds per Clock Cycle

Instruction Count, CPI, and Clock Period combine to form the three important components for determining CPU execution time. *Just analyzing one is not enough!* Performance between two machines can be determined by examining non-identical components.

Instruction Count and CPI

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction (CPI)
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

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

$$CPU\ Clock\ Cycles = Instructions\ for\ a\ Program \times Average\ Clock\ Cycles\ Per\ Instruction$$

$$CPU\ Time = Instruction\ Count \times CPI \times Clock\ Period$$

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

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
  - Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= \text{I} \times 2.0 \times 250\text{ps} = \text{I} \times 500\text{ps}$$
$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= \text{I} \times 1.2 \times 500\text{ps} = \text{I} \times 600\text{ps}$$
$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{\text{I} \times 600\text{ps}}{\text{I} \times 500\text{ps}} = 1.2$$

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instructio n Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instructio n Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instructio n Count}_i}{\text{Instructio n Count}}}_{\text{Relative frequency}} \right)$$

Weighted CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5 Clock Cycles = $2 \times 1 + 1 \times 2 + 2 \times 3 = 10$
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6 Clock Cycles = $4 \times 1 + 1 \times 2 + 1 \times 3 = 9$
  - Avg. CPI = 9/6 = 1.5

Performance Summary

The BIG Picture

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

What about the performance of Single-cycle CPU?

Performance in Single Cycle Implementation

- Let's see the following table:

Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
ld	200ps	100 ps	200ps	200ps	100 ps	800ps
sd	200ps	100 ps	200ps	200ps		700ps
R-type	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps

The conclusion:

Different instructions needs different time.

The clock cycle must meet the need of the slowest instruction. So, some time will be wasted.

Performance Issues

- Longest delay determines clock period
  - Critical path: load instruction
  - Instruction memory → register file → ALU → data memory → register file
- Wasteful of area. If the instruction needs to use some functional unit multiple times.
  - E.g., the instruction ‘mult’needs to use the ALU repeatedly. So, the CPU will be very large
- Violates design principle
  - Making the common case fast