

CSE 4305
Computer Organization and Architecture

Lecture 1

Introduction & History

Sabrina Islam
Lecturer, CSE, IUT
Contact: +8801832239897
E-mail: sabrinaislam22@iut-dhaka.edu

Computer organization refers to the way the hardware components of a computer system are structured and interconnected to achieve a specific set of goals.

- Deals with the internal organization of a computer's components and how they work together to execute instructions.
- Contributes to realizing the architectural specification.

Organizational attributes include the design of components such as registers, data paths, control units, memory systems, and input/output interfaces.

Computer architecture encompasses a broader view of the design and operation of a computer system.

- It is concerned with optimizing the performance of a computer system and ensuring that it can execute instructions quickly and efficiently.

Computer architecture involves designing the instruction set architecture (ISA), which defines the instructions that a processor can execute, as well as the memory hierarchy, system interconnects, parallel processing techniques, and performance optimization strategies.

CO vs CA

- 'What' and 'How'
- Instruction Set Architecture (ISA)
- Role
- Performance Optimization
- Technology Trends
- Attributes

CO VS CA



Desktop PC

- multi-core processor for high performance.
- dedicated graphics card for gaming and multimedia.
- large amount of RAM.
- I/O interfaces are designed for peripherals like monitors, keyboards, etc.

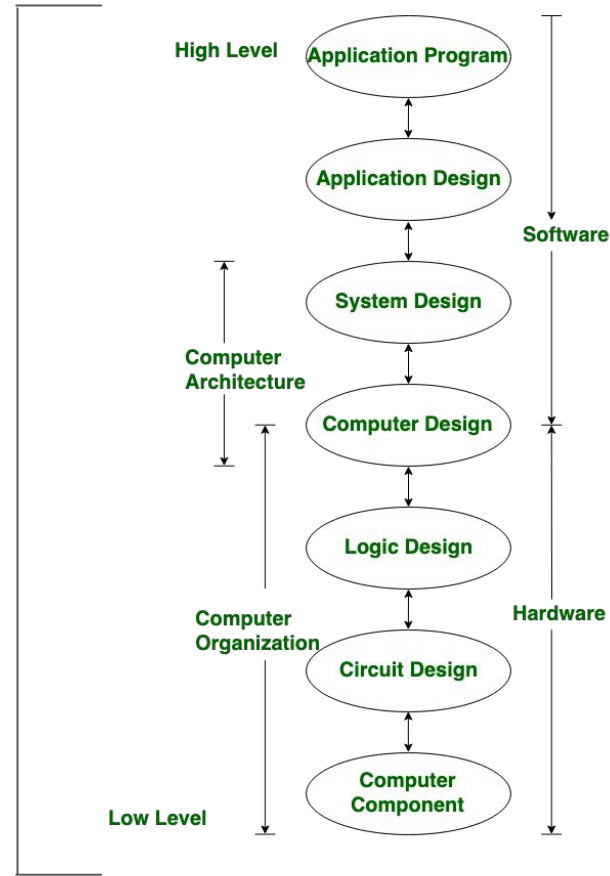


Embedded System

- single-core or lower-power processor to conserve energy.
- integrated graphics or no graphical interface at all.
- limited amount of RAM suitable for specific tasks.
- I/O interfaces are tailored for specific applications

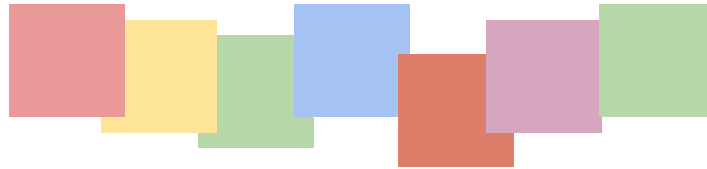
Architecture: x86 (Intel/AMD instruction set architecture)

Computer System





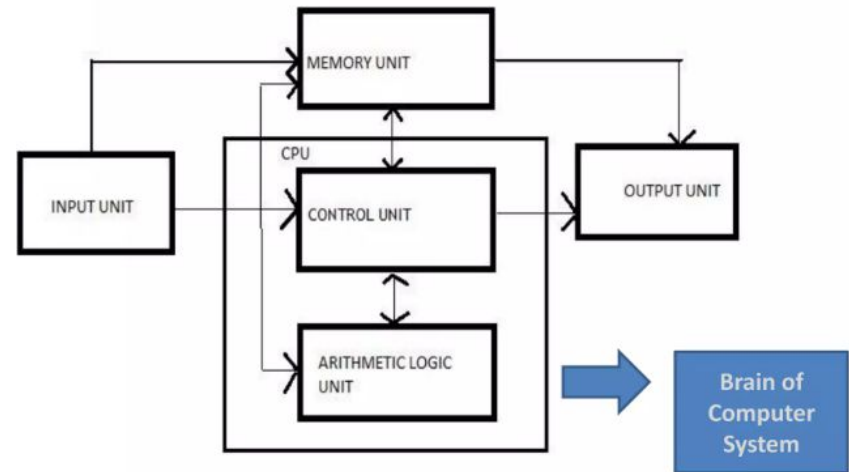
Interface



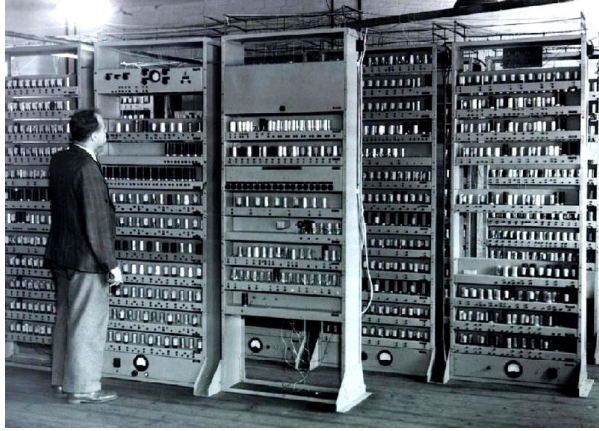
Basic organization of a computer

5 classic and functionally independent units of a computer: input, memory, arithmetic and logic, output and control units.

- **Central Processing Unit:** Controls the operation and performs its data processing– known as CPU or Processor.
- **Main Memory:** Stores data.
- **I/O:** Moves data between the computer and external world
- **System Interconnection:** Provides communication among CPU, memory, I/O through system bus. Keep components attached.



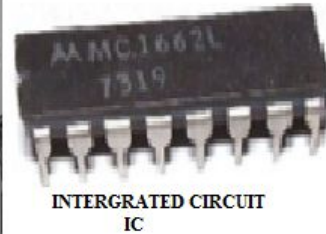
A brief history of computers



1st Gen



2nd Gen



3rd Gen

Year	Technology used in computers	Relative performance/unit cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit	900
1995	Very large-scale integrated circuit	2,400,000
2013	Ultra large-scale integrated circuit	250,000,000,000

End of Moore's law...?

In truth, it's been more a gradual decline than a sudden death.

The threats to Moore's law:

- Physical Limitations
- Heat Dissipation
- Manufacturing Costs
- Energy Efficiency

Online Material: [What Is Moore's Law? Is It Dead?](#)

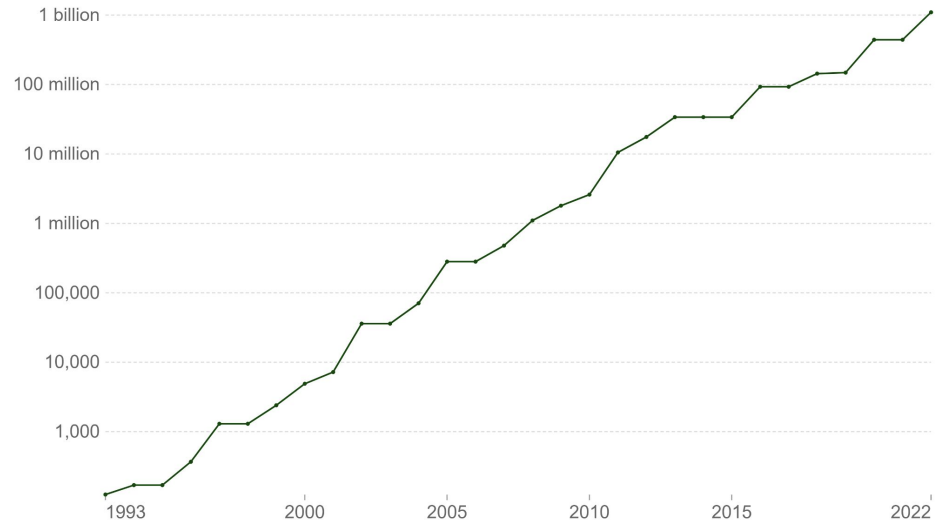
Other exponential technological changes

1. Computational capacity

The computational capacity of computers has increased exponentially, doubling every 1.5 years as well.

Computational capacity of the fastest supercomputers

The number of floating-point operations¹ carried out per second by the fastest supercomputer in any given year. This is expressed in gigaFLOPS, equivalent to 10^9 floating-point operations per second.



Source: TOP500 Supercomputer Database (2023)

OurWorldInData.org/technological-change • CC BY

1. Floating-point operation: A floating-point operation (FLOP) is a type of computer operation. One FLOP is equivalent to one addition, subtraction, multiplication, or division of two decimal numbers.

Other exponential technological changes

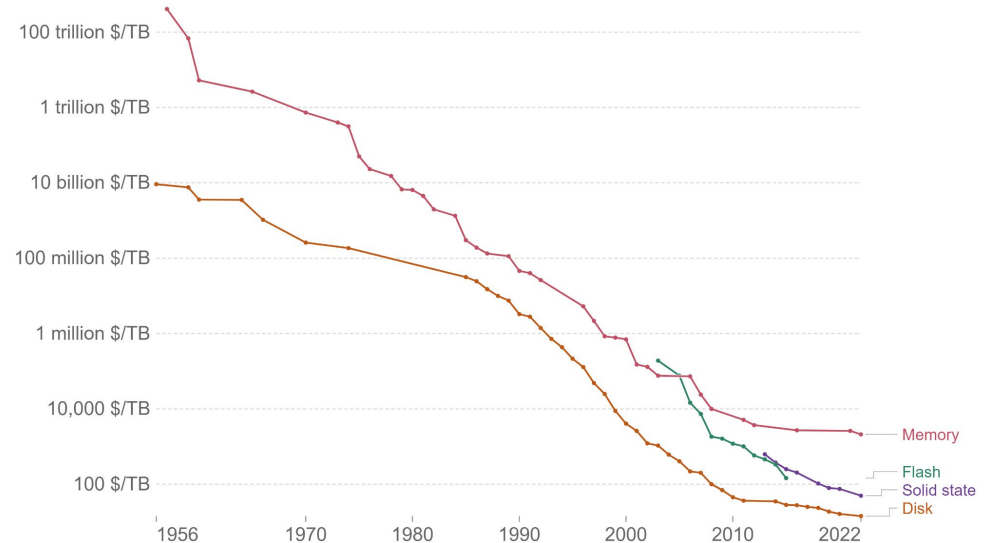
2. Computing efficiency and cost

Computing efficiency – measuring the energy use of computers – has halved every 1.5 years over the last 60 years.

Historical cost of computer memory and storage

This data is expressed in US dollars per terabyte (TB). It is not adjusted for inflation.

Our World
in Data



Source: John C. McCallum (2022)

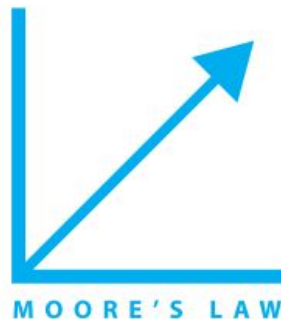
OurWorldInData.org/technological-change • CC BY

Note: For each year, the time series shows the cheapest historical price recorded until that year.

The eight great ideas...

1. Design for Moore's Law

Computer architects must anticipate where the technology will be when the design finishes rather than design for where it starts.



2. Use Abstraction to Simplify Design

A major productivity technique for hardware and software is to use abstractions to represent the design at different levels of representation. This is necessary to keep up with the dramatical growth of resources defined by Moore's law



The eight great ideas...

3. Make the Common Case Fast

By making common cases fast, computer systems achieve better performance and responsiveness in real-world scenarios.



COMMON CASE FAST

4. Performance via Parallelism

This idea promotes the use of parallel processing to achieve higher throughput and improved execution times.



PARALLELISM

The eight great ideas...

5. Performance via Pipelining

Pipelining involves breaking down the execution of instructions into stages and overlapping them to improve throughput. This idea emphasizes pipelining as a means to enhance performance.



6. Performance via Prediction

In some cases it can be faster on average to guess and start working rather than wait until you know for sure, assuming that the mechanism to recover from a misprediction is not too expensive and your prediction is relatively accurate.



The eight great ideas...

7. Hierarchy of Memories

This principle advocates the use of multiple levels of memory hierarchy, including registers, caches, and main memory, to provide a trade-off between speed and capacity keeping in mind the cost.



8. Dependability via Redundancy

This idea highlights the importance of optimizing the storage hierarchy for efficient data access. We make systems dependable by including redundant components that can take over when a failure occurs and to help detect failures.



Processor performance (clock rate) and power are correlated...

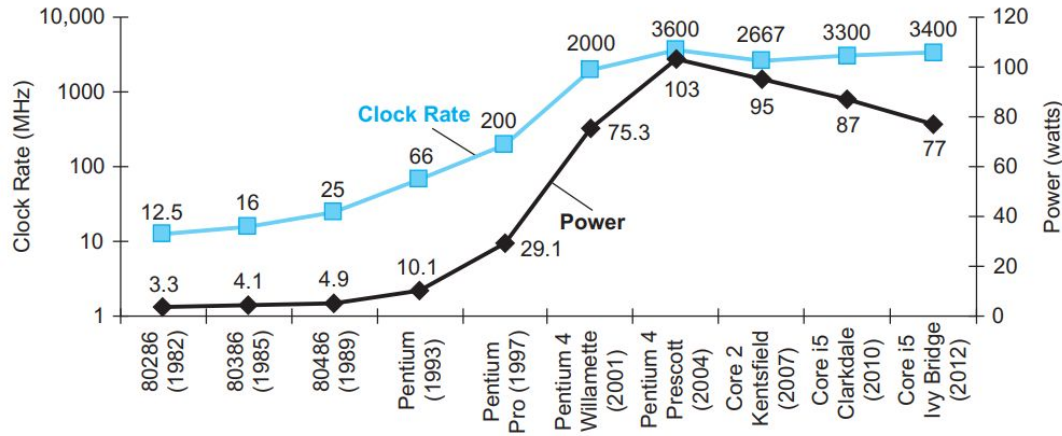


FIGURE 1.16 Clock rate and Power for Intel x86 microprocessors over eight generations and 25 years.

Processor performance (clock rate) and power are correlated...

The voltage in each transistors was lowered which resulted in reducing power consumption.

The power required per transistor:

$$Power \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}$$

Practice problem:

Suppose we developed a new, simpler processor that has 85% of the capacitive load of the more complex older processor. Further, assume that it has adjustable voltage so that it can reduce voltage 15% compared to processor B, which results in a 15% shrink in frequency. What is the impact on dynamic power?

$$Power_{new} = 0.52 * Power_{old}$$

But how further we can lower the voltage?

Issues:

- Current Leakage
- Expensive cooling technology to deal with the heat dissipation.

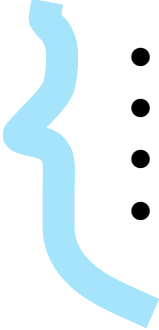

The Power Wall

The Power Wall

A point beyond which it becomes extremely difficult to increase the clock speed or overall performance of a processor due to the constraints imposed by **power consumption** and **heat dissipation**.

The power wall posed a major challenge for traditional approaches to improving processor performance. This shift in the landscape of processor design prompted the computer designers to choose a different path to improve efficiency. For example, **the switch from Uniprocessors to Multiprocessors...**

The Switch from Uniprocessors to Multiprocessors...

- 
- Improved performance
 - Increased throughput
 - Energy efficiency
 - Fault tolerance
- 

To utilize the best of these advantages, programmers must design or rewrite codes that effectively use multiprocessors. They will have to continue to improve performance of their code as the number of cores increases.

Fallacies and Pitfalls

Fallacy 1: Computers at low utilization use little power

The relationship between computer utilization and power consumption is more complex. Several factors contribute to power consumption such as fixed overhead, peripheral devices, maintenance etc.

A specially configured computer with the best results in 2012, still used 33% of the peak power at 10% of the load.

Fallacy 2: Designing for performance and designing for energy efficiency are unrelated goals.

The hardware or software optimizations that take less time, save energy overall even if the optimization takes a bit more energy when it is used.

Pitfall 1: Expecting the improvement of one aspect of a computer to increase overall performance by an amount proportional to the size of the improvement.

Enhancing a single component or aspect of a computer system does not lead to a linear improvement in overall performance. For instance, doubling the speed of a processor doesn't necessarily lead to a doubling in overall system performance because other components might not be able to keep up with the faster processor.

We will illustrate this fact by looking at a problem and solving it using **Amdahl's Law** (A rule stating that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used.)

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

Suppose a program runs in 100 seconds on a computer, with multiply operations responsible for 80 seconds of this time. How much do I have to improve the speed of multiplication if I want my program to run five times faster?

Solution:

$$\text{Execution time after improvement} = \frac{80 \text{ seconds}}{n} + (100 - 80 \text{ seconds})$$

$$20 \text{ seconds} = \frac{80 \text{ seconds}}{n} + 20 \text{ seconds}$$

$$0 = \frac{80 \text{ seconds}}{n}$$

So, there is **no amount** by which we can enhance-multiply to achieve a fivefold increase in performance, if multiply accounts for only 80% of the workload.

Pitfall 2: Using a subset of the performance equation as a performance metric.

It can lead to an incomplete or misleading understanding of a system's overall performance characteristics. The three factors to measure performance are Clock rate, Instruction count and CPI. Latency and Throughput, power consumption etc. are also considered. CPI-->cycles per instruction

A processor with a higher clock speed might not necessarily outperform another processor with a lower clock speed if the latter achieves higher instruction per cycle due to better instruction-level parallelism or more advanced microarchitectural features.

Thank You!