**MODULE 3: Computer Abstraction and Technology**

**INTRODUCTION**

**Computer**

A computer is a fast-electronic calculating machine which accepts digital input, processes it according to the internally stored instructions and produces the result on the output device.

**Functionalities of a computer**

Any digital computer carries out five functions

* Takes data as input.
* Store the data/instructions in its memory and use them when required.
* Process the data and converts it into useful information.
* Generates the output.
* Control all the above 4 steps.



Any kind of computers consists of **HARDWARE** AND **SOFTWARE**.

**Hardware**: Computer hardware is the collection of physical elements that constitutes a computer system. Computer hardware refers to the physical parts or components of a computer such as the monitor, mouse, keyboard, computer data storage, hard drive disk, etc. all of which are physical objects that can be touched.

**Software: Software** is a collection of [instructions](https://www.computerhope.com/jargon/c/compinst.htm) that enable the user to interact with a computer, its hardware, or perform tasks. Without software, most computers would be useless. For example, without your Internet [browser](https://www.computerhope.com/jargon/b/browser.htm) software, you could not surf the Internet or read this page.

**Classes of Computing Applications**

Computers are used in different classes of applications.

* **Personal computer**: A personal computer; designed to meet the computer needs of an individual. Provides access to a wide variety of computing applications such as word processing, photo editing, E-mail and internet.
* **Server:** Servers are the modern form of what were once much larger computers, and are usually accessed only via a network. Servers are oriented to carrying large workloads, which may consist of either single complex applications—usually a scientific or engineering application. Servers are built from the same basic technology as desktop computers, but provide for greater computing, storage, and input/output capacity.
* **Supercomputer**: An extremely fast computer that can perform hundreds of millions of instructions per second. It consist thousands of processors and many terabytes of memory. Supercomputers are usually used for high-end scientific and engineering calculations, such as weather forecasting, oil exploration, protein structure determination, and other large-scale problems.
* **Embedded computer:** As its name suggests, Embedded means something that is attached to another thing. Embedded computing systems are designed to run one application or one set of related applications that are normally integrated with the hardware and delivered as a single system. Embedded computers include the microprocessors found in your car, the computers in a television set, and the networks of processors that control a modern airplane or cargo ship.

**The Five Classic Components of a Computer**

A computer consists of five functionally independent main parts **input, memory, arithmetic logic unit (ALU), and output and control unit.**

**Input Unit:-** This accepts data or instructions as input,

**Storage Unit:-** This stores data and instruction.

**Arithmetic logic unit (contained in CPU):-** It processes data as per the instructions.

**Control unit (contained in CPU):-** It controls all operations inside a computer.

**Output Unit:-** This gives results in the form of output.

**CONTROL UNIT**

**ARITHMETIC & LOGIC UNIT**

**OUTPUT UNIT**

**INPUT UNIT**

**I/O**

**MEMORY**

**CPU/PROCESSOR**

**Input Devices**: Data and instructions must enter the computer system before any computation

can be performed on the supplied data. Input Devices link the external environment with the computer system to perform this task. Input Devices are used for entering data and programs into the computer system by the user for processing. Eg: Keyboard, Mouse.

**Storage Unit:**The data and instructions which are entered into the computer system through input devices (keyboard, mouse etc.) have to be stored inside the computer before the actual processing starts. Similarly, after processing, the results produced by the computer must be kept somewhere inside the computer system before being passed onto the output devices (monitor/screen or printer). Thus, the storage unit of a computer system is designed to take care of all these needs. Storage unit generally has primary memory and secondary/auxiliary memory for storing data.

**Arithmetic Logic Unit:** All calculations and comparisons, based on the instructions provided, are carried out within the ALU. It performs arithmetic functions like addition, subtraction, multiplication, division and also logical operations like greater than, less than and equal to etc.  
  
 **Control Unit:** Controlling of all operations like input, processing and output are performed by control unit. It takes care of step by step processing of all operations inside the computer.

**Output Unit:**Output Unit handles theprocess of producing results from the data for getting useful information through the screen or printer. The output generated by a computer can be transmitted to the user via some devices or media. These devices are called output devices. The following are some examples of different types of output devices commonly found on a computer. Eg: Monitor,Printer

**Computer Architecture**

Computer architecture in general covers three aspects of computer design namely: **Computer Hardware, Instruction Set Architecture and Computer Organization.**

Computer hardware consists of electronic circuits, displays, magnetic and optical storage media and communication facilities.

Instruction set Architecture is a programmer visible interface such as instruction set, registers, memory organization and exception handling.

Computer organization includes the high-level aspects of a design, such as memory system, the bus structure and the design of internal CPU

***In short, computer architecture refers to how a computer system is designed and what technologies it is compatible with.***

**Layers of abstraction**

In [computing](https://en.wikipedia.org/wiki/Computing), an **abstraction layer** or **abstraction level** is a way of hiding the working details of a subsystem. The hardware in a computer can only execute extremely simple low-level instructions. To go from a complex application to the simple instructions involves several layers of software that interpret or translate high-level operations into simple computer instructions, an example of the great idea of **abstraction**.

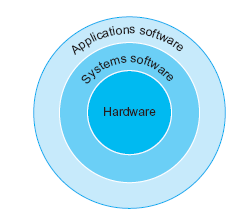
Applications being the outermost ring and a variety of systems softwaresitting between the hardware and applications software

There are many types of systems soft ware, but two types of systems software are central to every computer system today: an operating system and a compiler.

An **operating system** interfaces between a user’s program and the hardware and provides a variety of services and supervisory functions.

* Memory Management
* Processor Management
* Device Management
* File Management
* Security

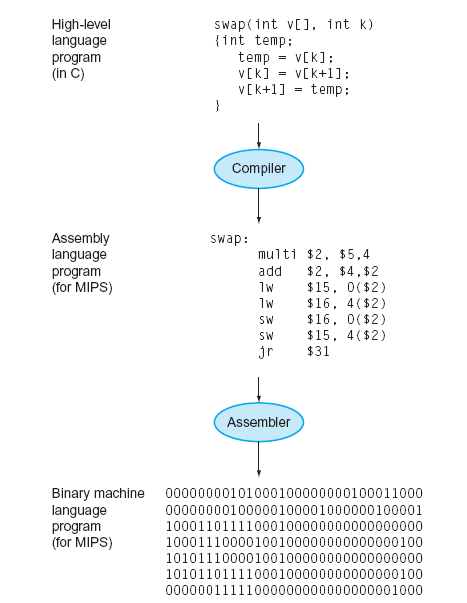
Eg: Linux, iOS, and Windows



**Compilers** perform another vital function: the translation of a program written in a high-level language, such as C, C++, Java, or Visual Basic into instructions that the hardware can execute.

The difference between compiler and assembler is that a compiler is used to convert high-level programming language code into machine language code. On the other hand, an assembler converts assembly level language code into machine language code. Both these terms are relevant in context to program execution.

**From a High-Level Language to the Language of Hardware**

****

High-level programming languages offer several important benefits.

* First, they allow the programmer to think in a more natural language, using English words and algebraic notation.
* The second advantage of programming languages is improved programmer productivity.
* The final advantage is that programming languages allow programs to be independent of the computer on which they were developed, since compilers and assemblers can translate high-level language programs to the binary instructions of any computer.

**Eight Great Ideas in Computer Architecture ( 8 design features)**

Application of the following great ideas has accounted for much of the tremendous growth in computing capabilities over the past 60 years.

* **1,Design for Moore’s Law: Moore’s law**, prediction made by American engineer [Gordon Moore](https://www.britannica.com/biography/Gordon-Moore) in 1965.It states that integrated circuit resource double every 18–24 months.

 The speed and capability of computers can be expected to double every two years, as a result of increases in the number of transistors a microchip can contain.

Computer architects must anticipate where the technology will be when the design finishes rather than design for where it starts.(**vacuum tube, transistors, IC, VLSI, quantum computing…..)**

* **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.

(Abstraction uses multiple levels with each level hiding the details of levels below it. For example:

* The instruction set of a processor hides the details of the activities involved in executing an instruction.
* High-level languages hide the details of the sequence of instructions need to accomplish a task.
* Operating systems hide the details involved in handling input and output devices.)
* **3,Make the Common Case Fast: Speed up =** Performance with enhancement

(The most significant improvements in computer performance come from improvements to the *common case* areas where the current design is spending the most time.

Eg**:IDE,SATA,SSD**….)

* **4,Performance via Parallelism:** Computer architects have offered designs that get more performance by performing operations in parallel. i.e A form of computation in which many calculations are carried out simultaneously

(Doing different parts of a task in parallel accomplishes the task in less time than doing them sequentially. A processor engages in several activities in the execution of an instruction. It runs faster if it can do these activities in parallel.eg: the concept like distributing computing facility in cloud………….)

* **5,Performance via Pipelining:** A particular pattern of parallelism. A set of data processing elements connected in series, so that the output of one element is the input of the next one.

(This idea is an extension of the idea of parallelism. It is essentially handling the activities involved in instruction execution as an assembly line. As soon as the first activity of an instruction is done you move it to the second activity **and** start the first activity of a new instruction. This results in executing more instructions per unit time compared to waiting for all activities of the first instruction to complete before starting the second instruction).

* **6,Performance via Prediction:** It can be faster on average to guess and start working rather than wait.

(A conditional branch is a type of instruction determines the next instruction to be executed based on a condition test. Conditional branches are essential for implementing high-level language if statements and loops.

Unfortunately, conditional branches interfere with the smooth operation of a pipeline — the processor does not know where to fetch the next instruction until after the condition has been tested.

Many modern processors reduce the impact of branches with speculative execution: make an informed guess about the outcome of the condition test and start executing the indicated instruction. )

* **7,Hierarchy of Memories:** Programmers want memory to be fast, large, and cheap, as memory speed often shapes performance and the cost of memory today is often the majority of computer cost. Architects have found that they can address these conflicting demands with a **hierarchy of memories**, with the fastest, smallest, and most expensive memory per bit at the top of the hierarchy and the slowest, largest, and cheapest per bit at the bottom.

(The principle of locality states that memory that has been accessed recently is likely to be accessed again in the near future. That is, accessing recently accessed data is a common case for memory accesses. To make this common case faster you need a cache — a small high-speed memory designed to hold recently accessed data.

Modern processors use as many as 3 levels of caches. This is motivated by the large difference in speed between processors and memory.)

* **8,Dependability via Redundancy:** Computers not only need to be fast; they need to be dependable. Since any physical device can fail, we make systems dependable by including redundant components that can take over when a failure occurs andto help detect failures.

**Technologies for Building Processors and Memory**

Processors and memory have improved at an incredible rate, because computer designers have accepted the latest in electronic technology to try to win the race to design a better computer.

**Year Technology Relative performance/cost**

1951 Vacuum tube 1

1965 Transistor 35

1975 Integrated circuit (IC) 900

1995 Very large scale IC (VLSI) 2,400,000

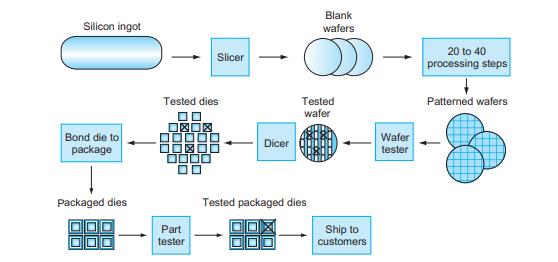
2013 Ultra large scale IC 250,000,000,000

**The basics of integrated circuits.**

*Transistor*: An on/off switch controlled by an electric signal. The integrated circuit (IC) combined dozens to hundreds of transistors into a single chip.

*Very large-scale integrated* (VLSI) circuit: A device containing hundreds of thousands to millions of transistors**.**

**The chip manufacturing process.**

****

Silicon crystal ingot: A rod composed of a silicon crystal that is between 8 and 12 inches in diameter and about 12 to24 inches long

Wafer: A slice from a silicon ingot no more than 0.1 inches thick, used to create chips.

*Die or Chip*: The individual rectangular sections that are cut from a wafer, more informally known as chips.

*Yield*: The percentage of good dies from the total number of dies on the wafer.

The process starts with a silicon crystal ingot. After being sliced from the silicon ingot, blank wafers are put through 20 to 40 steps to create patterned wafers. These patterned wafers are then tested with a wafer tester, and a map of the good parts is made. Then, the wafers are diced into dies. In this figure, one wafer produced 20 dies, of which 17 passed testing. (X means the die is bad.) The yield of good dies in this case was 17/20, or 85%. These good dies are then bonded into packages and tested one more time before shipping the packaged parts to customers. One bad packaged part was found in this final test.

<https://www.youtube.com/watch?v=tcVlBnIlTMg>

**Performance of Computer**

[**https://www.geeksforgeeks.org/computer-organization-performance-of-computer/**](https://www.geeksforgeeks.org/computer-organization-performance-of-computer/)

**Computer performance** is the amount of work accomplished by a computer system. The word performance in computer performance means “How well is the computer doing the work it is supposed to do?”. It basically depends on **response time, throughput and execution time of** a computer system.

**Response time** is the time from start to completion of a task. This also includes:

* Operating system overhead.
* Waiting for I/O and other processes
* Accessing disk and memory
* Time spent executing on the CPU or execution time.

**Throughput** is the total amount of work done in a given time.

**CPU execution time** is the total time a CPU spends computing on a given task. It also excludes time for I/O or running other programs. This is also referred to as simply CPU time.

**[[**. Response time is the sum of service time and waiting time Again, **CPU time is the actual time the CPU spends computing for a specific task**.

Performance is determined by execution time as performance is inversely proportional to execution time.]]

**Performance = (1 / Execution time)**

And,

(Performance of A / Performance of B) = (Execution Time of B / Execution Time of A)

If given that Processor A is faster than processor B, that means execution time of A is less than that of execution time of B. Therefore, performance of A is greater than that of performance of B.

**Example –**

Machine A runs a program in 100 seconds, Machine B runs the same program in 125 seconds

(Performance of A / Performance of B)

= (Execution Time of B / Execution Time of A)

= 125 / 100 = 1.25

That means machine A is 1.25 times faster than Machine B.

The time to execute a given program can be computed as:

Execution time = CPU clock cycles x clock cycle time

Since clock cycle time and clock rate are reciprocals, so,

Execution time = CPU clock cycles / clock rate

Clock rate= Number of cycles /Second

The number of CPU clock cycles can be determined by,

CPU clock cycles

= (No. of instructions / Program ) x (Clock cycles / Instruction)

= Instruction Count x CPI

(The term clock cycles per instruction, which is the average number of clock cycles each instruction takes to execute, is often abbreviated as CPI.)

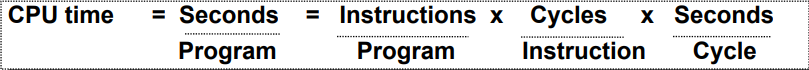
Which gives,

Execution time

= Instruction Count x CPI x clock cycle time

= Instruction Count x CPI / clock rate

**The units for CPU Execution time are:**



**How to Improve Performance?**  
To improve performance you can either:

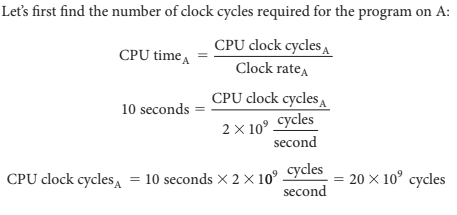
* Decrease the CPI (clock cycles per instruction) by using new Hardware.
* Decrease the clock time or Increase clock rate by reducing propagation delays or by use pipelining.
* Decrease the number of required cycles or improve ISA or Compiler.

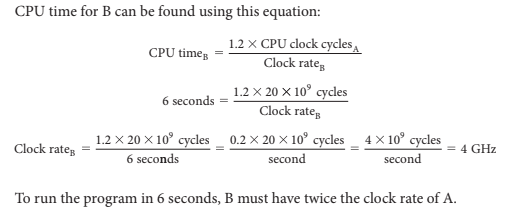
The instruction set architecture (ISA) is **a protocol that defines how a computing machine appears to a machine language programmer or compiler**. The ISA describes the (1) memory model, (2) instruction format, types and modes, and (3) operand registers, types, and data addressing.

**Example:**

Q, A program runs on computer A with a 2 GHz clock in 10 seconds. What clock rate must a computer B run at to run this program in 6 seconds? Unfortunately, to accomplish this, computer B will require 1.2 times as many clock cycles as computer A to run the program.

((**1 GHz = 109 Hz = 1000000kHz =1000 MHz**.))





**Instruction Performance**

The performance equations above did not include any reference to the number of instructions needed for the program. The compiler clearly generated instructions to execute, and the computer had to execute the instructions to run the program, the execution time must depend on the number of instructions in a program. One way to think about execution time is that it equals the number of instructions executed multiplied by the average time per instruction. Therefore, the number of clock cycles required for a program can be written as

**CPU clock cycles = Instructions for a program x Average clock cycles per instruction**

The term clock cycles per instruction, which is the average number of clock cycles each instruction takes to execute, is often abbreviated as CPI.

Eg: Suppose we have two implementations of the same instruction set architecture.

Q, Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which computer is faster for this program and by how much?