**Chapter-1:**

**Memory Hierarchy-**

* Memory hierarchy is the hierarchy of memory and storage devices found in a computer system.
* It ranges from the slowest but high capacity auxiliary memory to the fastest but low capacity cache memory.

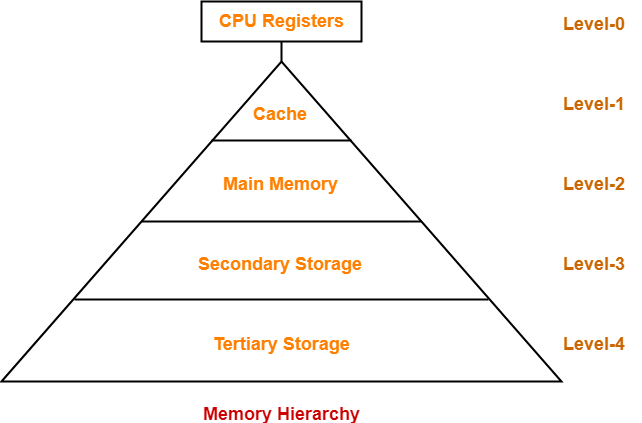
**Need-**

There is a trade-off among the three key characteristics of memory namely-

* Cost
* Capacity
* Access time

Memory hierarchy is employed to balance this trade-off.

**Memory Hierarchy Diagram-**



**Level-0:**

* At level-0, registers are present which are contained inside the CPU.
* Since they are present inside the CPU, they have least access time.
* They are most expensive and therefore smallest in size (in KB).
* Registers are implemented using [**Flip-Flops**](https://www.gatevidyalay.com/latches-and-flip-flops/).

**Level-1:**

* At level-1, [**Cache Memory**](https://www.gatevidyalay.com/cache-memory/) is present.
* It stores the segments of program that are frequently accessed by the processor.
* It is expensive and therefore smaller in size (in MB).
* Cache memory is implemented using static RAM.

**Level-2:**

* At level-2, main memory is present.
* It can communicate directly with the CPU and with auxiliary memory devices through an I/O processor.
* It is less expensive than cache memory and therefore larger in size (in few GB).
* Main memory is implemented using dynamic RAM.

**Level-3:**

* At level-3, secondary storage devices like [**Magnetic Disk**](https://www.gatevidyalay.com/magnetic-disk-secondary-memory-coa/) are present.
* They are used as back up storage.
* They are cheaper than main memory and therefore much larger in size (in few TB).

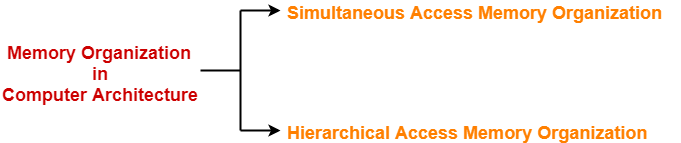
**Level-4:**

* At level-4, tertiary storage devices like magnetic tape are present.
* They are used to store removable files.
* They are cheapest and largest in size (1-20 TB).

## ****Memory Organization in Computer Architecture-****

In a computer,

* Memory is organized at different levels.
* CPU may try to access different levels of memory in different ways.
* On this basis, the memory organization is broadly divided into two types-



1. Simultaneous Access Memory Organization
2. Hierarchical Access Memory Organization

**Simultaneous Access Memory Organization**

* All the levels of memory are directly connected to the CPU.
* Whenever CPU requires any word, it starts searching for it in all the levels simultaneously

**Hierarchical Access Memory Organization**

In this memory organization, memory levels are organized as-

* Level-1 is directly connected to the CPU.
* Level-2 is directly connected to level-1.
* Level-3 is directly connected to level-2 and so on.

Whenever CPU requires any word,

* It first searches for the word in level-1.
* If the required word is not found in level-1, it searches for the word in level-2.
* If the required word is not found in level-2, it searches for the word in level-3 and so on.

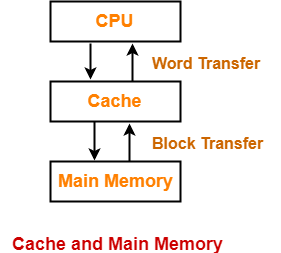
**Chapter-2:**

**Cache Memory-**

* Cache memory is a Random Access Memory.
* The main advantage of cache memory is its very fast speed.
* It can be accessed by the CPU at much faster speed than main memory.

**Location-**

* Cache memory lies on the path between the CPU and the main memory.
* It facilitates the transfer of data between the processor and the main memory at the speed which matches to the speed of the processor.



* Data is transferred in the form of words between the cache memory and the CPU.
* Data is transferred in the form of blocks or pages between the cache memory and the main memory.

**Purpose-**

* The fast speed of the cache memory makes it extremely useful.
* It is used for bridging the speed mismatch between the fastest CPU and the main memory.
* It does not let the CPU performance suffer due to the slower speed of the main memory.

**Working Methods Of Cache**

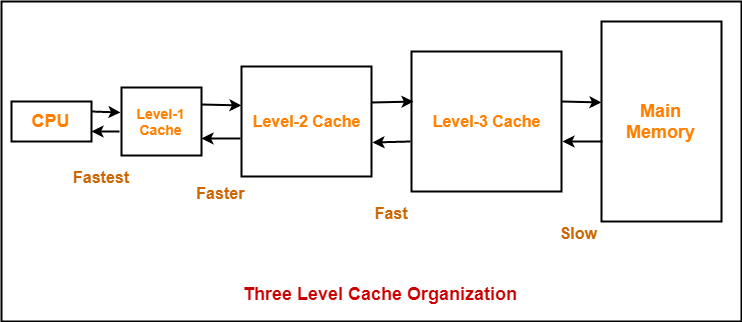
* Whenever any program has to be executed, it is first loaded in the main memory.
* The portion of the program that is mostly probably going to be executed in the near future is kept in the cache memory.
* This allows CPU to access the most probable portion at a faster speed

**Different Layer of Cache**

* A multilevel cache organization is an organization where cache memories of different sizes are organized at multiple levels to increase the processing speed to a greater extent.
* The smaller the size of cache, the faster its speed.
* The smallest size cache memory is placed closest to the CPU.
* This helps to achieve better performance in terms of speed.

Three level cache organization consists of three cache memories of different size organized at three different levels as shown below-

**Size (L1 Cache) < Size (L2 Cache) < Size (L3 Cache) < Size (Main Memory)**



**Cache Mapping Technique**

**Direct Mapping-**

In direct mapping,

* A particular block of main memory can map only to a particular line of the cache.
* The line number of cache to which a particular block can map is given by

**Cache line number**

**= ( Main Memory Block Address ) Modulo (Number of lines in Cache)**

## ****Fully Associative Mapping****

In fully associative mapping,

* A block of main memory can map to any line of the cache that is freely available at that moment.
* This makes fully associative mapping more flexible than direct mapping.

In fully associative mapping, the physical address is divided as-



**K-way Set Associative Mapping-**

In k-way set associative mapping,

* Cache lines are grouped into sets where each set contains k number of lines.
* A particular block of main memory can map to only one particular set of the cache.
* However, within that set, the memory block can map any cache line that is freely available.
* The set of the cache to which a particular block of the main memory can map is given by-

|  |
| --- |
| **Cache set number**  **= ( Main Memory Block Address ) Modulo (Number of sets in Cache)** |

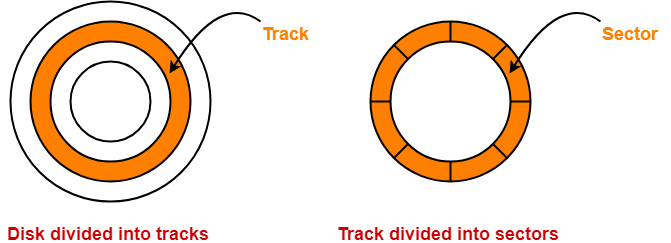
**Chapter-3:**

## ****Magnetic Disk in Computer Architecture****

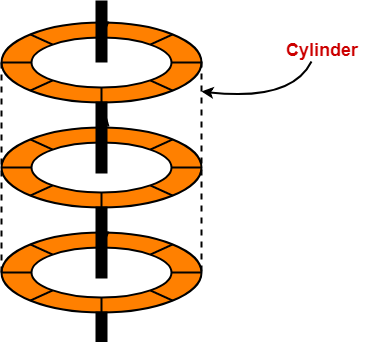
* Magnetic disk is a storage device that is used to write, rewrite and access data.
* It uses a magnetization process.

**Architecture-**

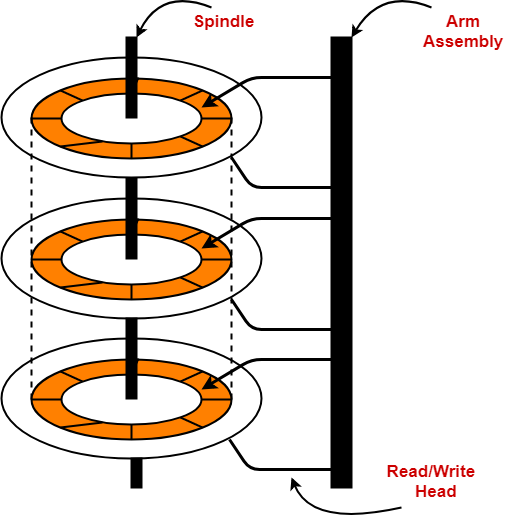
* The entire disk is divided into **platters**.
* Each platter consists of concentric circles called as **tracks**.
* These tracks are further divided into **sectors** which are the smallest divisions in the disk.



* A **cylinder** is formed by combining the tracks at a given radius of a disk pack.



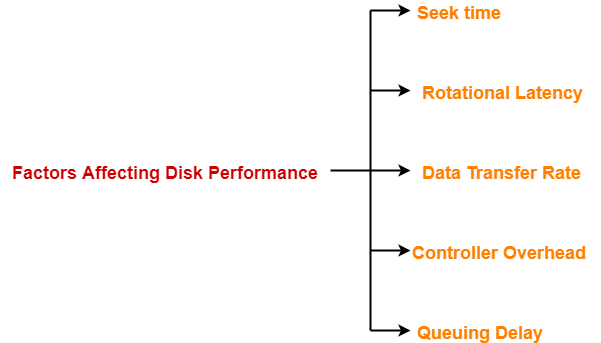
* There exists a mechanical arm called as **Read / Write head**.
* It is used to read from and write to the disk.
* Head has to reach at a particular track and then wait for the rotation of the platter.
* The rotation causes the required sector of the track to come under the head.
* Each platter has 2 surfaces- top and bottom and both the surfaces are used to store the data.
* Each surface has its own read / write head.



**Disk Performance Parameters-**

The time taken by the disk to complete an I/O request is called as **disk service time** or **disk access time**.

Components that contribute to the service time are-



**Seek Time-**

* The time taken by the read / write head to reach the desired track is called as **seek time**.
* It is the component which contributes the largest percentage of the disk service time.
* The lower the seek time, the faster the I/O operation.

**Rotational Latency-**

* The time taken by the desired sector to come under the read / write head is called as **rotational latency**.
* It depends on the rotation speed of the spindle.

|  |
| --- |
| Average rotational latency = 1 / 2 x Time taken for full rotation |

**Data Transfer Rate-**

* The amount of data that passes under the read / write head in a given amount of time is called as **data transfer rate**.
* The time taken to transfer the data is called as **transfer time**.

It depends on the following factors-

1. Number of bytes to be transferred
2. Rotation speed of the disk
3. Density of the track
4. Speed of the electronics that connects the disk to the computer

**Disk Access Time-**

Disk access time is calculated as-

|  |
| --- |
| Disk access time  = Seek time + Rotational delay + Transfer time |

**Chapter-4:**

## ****Addressing Modes****

|  |
| --- |
| The different ways of specifying the location of an operand in an instruction are called as **addressing modes**. |

## ****Types of Addressing Modes****

1. Implied / Implicit Addressing Mode
2. Stack Addressing Mode
3. Immediate Addressing Mode
4. Direct Addressing Mode
5. Indirect Addressing Mode
6. Register Direct Addressing Mode
7. Register Indirect Addressing Mode
8. Relative Addressing Mode
9. Indexed Addressing Mode
10. Base Register Addressing Mode
11. Auto-Increment Addressing Mode
12. Auto-Decrement Addressing Mode

.

**Chapter-5:**

# [Pipelining in Computer Architecture](https://www.gatevidyalay.com/pipelining-in-computer-architecture/)

* A program consists of several number of instructions.
* These instructions may be executed in the following two ways-
* Non-Pipelined Execution
* Pipelined Execution

## ****Non-Pipelined Execution****

* All the instructions of a program are executed sequentially one after the other.
* A new instruction executes only after the previous instruction has executed completely.
* This style of executing the instructions is highly inefficient

## ****Pipelined Execution****

* Multiple instructions are executed parallely.
* This style of executing the instructions is highly efficient.

**Pipelined Architecture-**

In pipelined architecture,

* The hardware of the CPU is split up into several functional units.
* Each functional unit performs a dedicated task.
* The number of functional units may vary from processor to processor.
* These functional units are called as stages of the pipeline.
* Control unit manages all the stages using control signals.
* There is a register associated with each stage that holds the data.
* There is a global clock that synchronizes the working of all the stages.
* At the beginning of each clock cycle, each stage takes the input from its register.
* Each stage then processes the data and feed its output to the register of the next stage.

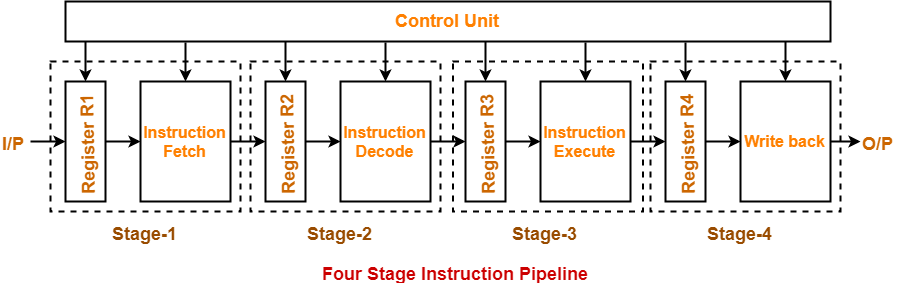
In four stage pipelined architecture, the execution of each instruction is completed in following 4 stages-

**Instruction fetch (IF)**

**Instruction decode (ID)**

**Instruction Execute (IE)**

**Write back (WB)**



**Chapter-6:**

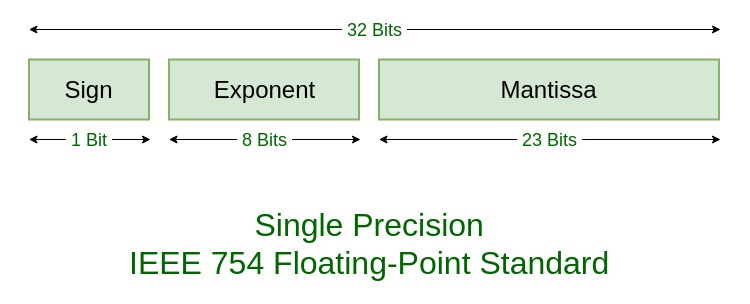
# IEEE Standard 754 Floating Point Numbers

The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the **Institute of Electrical and Electronics Engineers (IEEE)**. The standard addressed many problems found in the diverse floating point implementations that made them difficult to use reliably and reduced their portability. IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC’s, Macs, and most Unix platforms.

There are several ways to represent floating point number but IEEE 754 is the most efficient in most cases. IEEE 754 has 3 basic components:

1. **The Sign of Mantissa –**  
   This is as simple as the name. 0 represents a positive number while 1 represents a negative number.
2. **The Biased exponent –**  
   The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent.
3. **The Normalised Mantissa –**  
   The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. O and 1. So a normalised mantissa is one with only one 1 to the left of the decimal.

**IEEE 754 numbers are divided into two based on the above three components: single precision and double precision.**

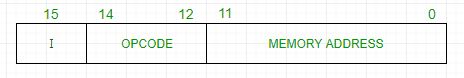


**Chapter-7:**

# Basic Computer Instructions

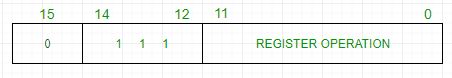
The basic computer has 16-bit instruction register (IR) which can denote either memory reference or register reference or input-output instruction.

1. **Memory Reference –** These instructions refer to memory address as an operand. The other operand is always accumulator. Specifies 12-bit address, 3-bit opcode (other than 111) and 1-bit addressing mode for direct and indirect addressing.



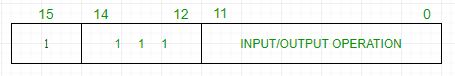
**Example –**  
IR register contains = 0001XXXXXXXXXXXX, i.e. ADD after fetching and decoding of instruction we find out that it is a memory reference instruction for ADD operation.

1. **Register Reference –** These instructions perform operations on registers rather than memory addresses. The IR(14 – 12) is 111 (differentiates it from memory reference) and IR(15) is 0 (differentiates it from input/output instructions). The rest 12 bits specify register operation.



**Example –**  
IR register contains = 0111001000000000, i.e. CMA after fetch and decode cycle we find out that it is a register reference instruction for complement accumulator.

1. **Input/Output –**These instructions are for communication between computer and outside environment. The IR(14 – 12) is 111 (differentiates it from memory reference) and IR(15) is 1 (differentiates it from register reference instructions). The rest 12 bits specify I/O operation.



**Example –**  
IR register contains = 1111100000000000, i.e. INP after fetch and decode cycle we find out that it is an input/output instruction for inputing character. Hence, INPUT character from peripheral device.

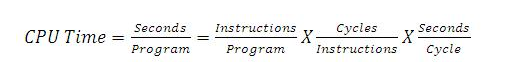
# RISC and CISC

**Reduced Set Instruction Set Architecture (RISC) –**  
The main idea behind is to make hardware simpler by using an instruction set composed of a few basic steps for loading, evaluating and storing operations just like a load command will load data, store command will store the data.

**Complex Instruction Set Architecture (CISC) –**  
The main idea is to make hardware complex as a single instruction will do all loading, evaluating and storing operations just like a multiplication command will do stuff like loading data, evaluating and storing it.

Both approaches try to increase the CPU performance

* **RISC:** Reduce the cycles per instruction at the cost of the number of instructions per program.
* **CISC:** The CISC approach attempts to minimize the number of instructions per program but at the cost of increase in number of cycles per instruction.



Earlier when programming was done using assembly language, a need was felt to make instruction do more task because programming in assembly was tedious and error prone due to which CISC architecture evolved but with uprise of high level language dependency on assembly reduced RISC architecture prevailed.

**Characteristic of RISC –**

1. Simpler instruction, hence simple instruction decoding.
2. Instruction come under size of one word.
3. Instruction take single clock cycle to get executed.
4. More number of general purpose register.
5. Simple Addressing Modes.
6. Less Data types.
7. Pipeling can be achieved.

**Characteristic of CISC –**

1. Complex instruction, hence complex instruction decoding.
2. Instruction are larger than one word size.
3. Instruction may take more than single clock cycle to get executed.
4. Less number of general purpose register as operation get performed in memory itself.
5. Complex Addressing Modes.
6. More Data types.