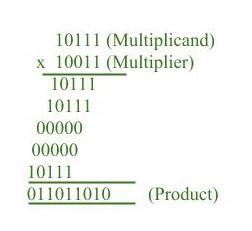
**### Define computer architecture. explain about instruction codes and instruction types in brief? 🡪** Computer architecture refers to the design and organization of a computer's core components and the interactions between them. It encompasses the structure, behavior, and implementation of the computer's fundamental elements, including the central processing unit (CPU), memory, input/output (I/O) systems, and data pathways (buses). Essentially, computer architecture focuses on how the hardware and software work together to achieve efficient computing. **Instruction Codes: Instruction codes** (or machine codes) are binary codes that the CPU can execute directly. Each instruction code corresponds to a specific operation that the CPU can perform, such as arithmetic calculations, data movement, or control operations. The set of all instruction codes that a CPU can execute is known as its **instruction set architecture (ISA)**. **Key Components of an Instruction Code: Opcode (Operation Code):** Specifies the operation to be performed (e.g., add, subtract, load, store). **Operands:** Specifies the data to be operated on. Operands can be constants, memory addresses, or registers. **Addressing Mode:** Indicates how the operand should be interpreted (e.g., direct, indirect, immediate) **Instruction Types 1)Data Transfer Instructions:** **Move (MOV):** Transfers data from one location to another. **Load (LD):** Loads data from memory into a register. **Store (ST):** Stores data from a register into memory. 2)**Arithmetic Instructions:** **Add (ADD):** Adds two operands. **Subtract (SUB):** Subtracts one operand from another. **Multiply (MUL):** Multiplies two operands. **Divide (DIV):** Divides one operand by another. 3)**Logical Instructions:** **AND:** Performs a bitwise AND operation. **OR:** Performs a bitwise OR operation. **XOR:** Performs a bitwise XOR operation. **NOT:** Performs a bitwise NOT operation (complement).

**## Define address sequencing. Explain about general resister organization in CPU briefly.** 🡪Address sequencing refers to the process by which the control unit of a CPU determines the sequence in which memory addresses are accessed during program execution. It ensures that the correct sequence of instructions is fetched, decoded, and executed. **Components of Address Sequencing: Program Counter (PC):** Holds the address of the next instruction to be executed. After an instruction is fetched, the PC is updated to point to the subsequent instruction. **Instruction Register (IR):** Holds the currently executing instruction. **Address Register (AR):** Holds the address of the memory location to be accessed. **\* General Register Organization in CPU ::** In the context of CPU architecture, a general register organization refers to a CPU design where multiple general-purpose registers are used for various operations such as arithmetic, logic, and data transfer. These registers can be accessed quickly by the CPU, enhancing the overall performance and flexibility of the system. **\*\* Characteristics of General Registers: General-Purpose Use:** Registers can be used for a variety of tasks, such as storing intermediate results, counters, pointers, and indexes. **Fast Access:** Registers provide the fastest form of data storage and retrieval compared to other memory types. **Versatility:** They support various data types and operations, making them essential for complex instruction execution \*\***Typical General Registers in a CPU:, Accumulator (AC):** Used for arithmetic and logic operations. **Base Register (BR):** Holds the base address for memory operations. **Index Register (IR or XR):** Used for indexed addressing modes. **Stack Pointer (SP):** Points to the top of the stack, used in stack operations. **Instruction Register (IR):** Holds the current instruction being executed **General-Purpose Registers (R0, R1, ..., Rn):** Used for general data storage and manipulation.

**## Define Addressing modes in 8085. Explain different types of addressing mode used in 8085 suitable example . 🡪**Addressing modes define the way in which the operand of an instruction is specified. They determine how the effective address of the operand is calculated and accessed during program execution. Addressing modes enhance the flexibility and efficiency of a CPU by providing various methods to specify operands. **\*\* Types of Addressing Modes Immediate Addressing Mode**: The operand is directly specified in the instruction itself.Example: MOV R1, #10 (Here, 10 is the immediate operand).Usage: Quick access to constant values. **Direct Addressing Mode**:The address of the operand is given explicitly within the instruction.Example: MOV R1, 500 (Here, 500 is the direct address of the operand).Usage: Simple and easy to understand, but limited by the address space. **Indirect Addressing Mode:** The address of the operand is specified by a pointer or reference in a register or memory location.Example: MOV R1, (R2) (Here, R2 contains the address of the operand).usage: Allows for more flexible and dynamic address calculations. **Register Addressing Mode:**  The operand is located in a register specified by the instruction.Example: MOV R1, R2 (Here, the operand is in register R2).Usage: Fast access to operands since registers are faster than memory. **Register Indirect Addressing Mode:**: The address of the operand is held in a register. Example: MOV R1, (R2) (Here, R2 contains the memory address of the operand).Usage: Useful for pointers and dynamicmemory access.

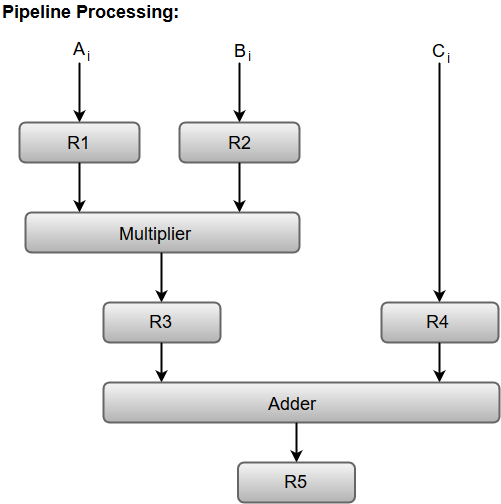
|  |  |
| --- | --- |
| **RISC** | **CISC** |
| it is a Reduce Instruction Set Computer. | It is a Complex Instruction Set Computer |
| It focuses on software | It focuses on hardware |
| It uses only hardwired control unit | It use both hardware and microprogrammed control unit |
| Code size large | Code size is small |
| The uses of the pipeline are simple in RICS | Uses of the pipeline are difficult in CISC |

**# Explain the multiplication process of signed magnitude data with suitable example. 🡪** The representation of numbers in signed-magnitude is familiar because it is used in everyday arithmetic calculation. The procedure for adding or subtracting two signed binary numbers with paper and pencils simple and straight-forward. A review of this procedure will be helpful for driving the hardware algorithm.

**## Define asynchronous data transfer. Explain different types of modes of transfer in brief.** 🡪 Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted . Two Asynchronous data transfer methods : \*\* **Strobe pulse:** A strobe pulse is supplied by one unit to indicated the other unit when the transfer has to occur. **Characteristics of a Strobe Pulse Timing Signal**: The strobe pulse is typically a short-duration signal that precisely marks the timing for data transfer. **Synchronization**: It is used to synchronize data transmission between devices that do not share a common clock. **Control Mechanism**: The strobe pulse can be used to control various operations such as reading data from memory or writing data to an output device. **\*\*\*Handshaking** : A control signal is a accompanied with each data . The receiving unit responds with another control signal to acknowledge receipt of the data.  **Importance of Handshaking Synchronization**: Ensures both devices are synchronized and ready for data transfer. **Error Control**: Provides mechanisms to detect and correct errors in data transmission. **Control**: Manages the rate of data transfer to prevent buffer overflow or underflow.

**## Define pipelining. Explain about vector processing with vector operation.**

**🡪** Pipelining is a technique of decomposing a sequential process into sub-operation, with each sub-operation being executed in a special dedicated segments that operates concurrently with all other segments. Each segment performs partial processing dictated by the way task is partitioned . The result obtained from each segments is transferred to next segment. The final result is obtained when data have passed through all segments.



**\*\*Vector processing :**  There is a class of computational problems that are beyond the capabilities of the conventional computer. There are characterized by the fact that they require vast number of computation and it take a conventional computer days or even weeks to complete. Computers with vector processing are able to handle such instruction and they have application in following fields. Long range weather forecasting. **\*\* Vector Operation:** A vector V of length n is represented as a row vector by V=[ V1 V2 V3…..Vn ] . The element Vi of vector V is written as V(i) and the index i refers to a memory address or register where the number is stored. **\*\* Matrix Multiplication :** Let us consider the multiplication of two 3\*3 matrix A and B. This requires three multiplication and ( after initializing C11 to O ) three addition. Total number of addition or multiplication required is 3 \* 9. In general inner product consists of the sum of k product terms of the form: C= A1 B1 + A2 B2 + A3 B3 + A4 B4 +….+ Ak Bk  In typical application value of K may be 100 or even 1000. The inner product calculation on a pipeline vector processor is show below . Floating point adder and multiplier are assumed to have four segments each.

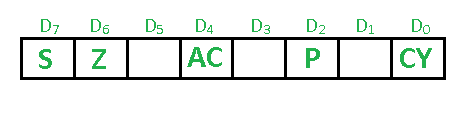
**\*\* Types of pipeline: Arithmetic Pipeline :** Pipeline arithmetic units are usually found in very high speed computers. They are used to implement floating ppoint operations. We will now discuss the pipeline unit for the floating point addition and subtraction. **Instruction Pipeline:** Pipeline processing can occur not only in the data stream but in the instruction stream as well. An instruction pipeline reads consecutive instruction from memory while previous instruction are being executed in other segments. This caused the instruction fetch and execute segments to overlap and performs simultaneous operation.

## Direct memory access is a capability provided by DSP computer bus architectures that allows data to be sent directly from an attached device (such as a disk drive or external memory) to other memory locations in the DSP address space. The DSP is freed from involvement with the [data transfer](https://www.sciencedirect.com/topics/engineering/data-transfer), thus speeding up overall computer operation.

**\*\* Direct Memory Access:**

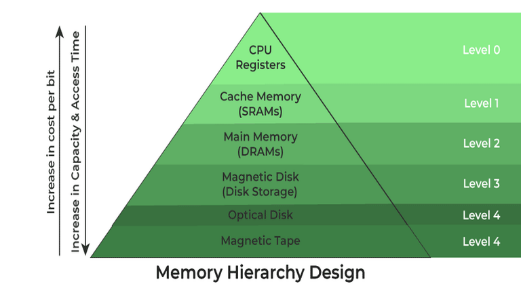
DMA devices are partially dependent [coprocessors](https://www.sciencedirect.com/topics/engineering/coprocessor) which offload data transfers from the main CPU. DMA offers a high performance capability to DSPs. Modern DSP DMA controllers such as the TMS320C6x can transfer data at up to 800 Mbytes/sec (sustained). This DMA [controller can](https://www.sciencedirect.com/topics/computer-science/can-controller) read and write one 32-bit word every cycle. Offloading the work of transferring data allows the CPU to focus on computation. **Typical types of transfers that a DMA may control are:** •Memory to memory transfers (internal to external).•Transfers from I/O devices to memory. • Transfers from memory to I/O devices.•Transfers from/to communications ports and memory.•Transfers from/to [serial ports](https://www.sciencedirect.com/topics/engineering/serial-port) and memory.

**\*\* Flag register:** The Flag register is a 8-bit register in the 8085 microprocessor that contains information about the status of the arithmetic and logic operations performed by the processor. The bits in the Flag register are used to indicate whether the result of an operation is zero, positive, negative, or if there was a carry or borrow during the operation. The 5 flags are:

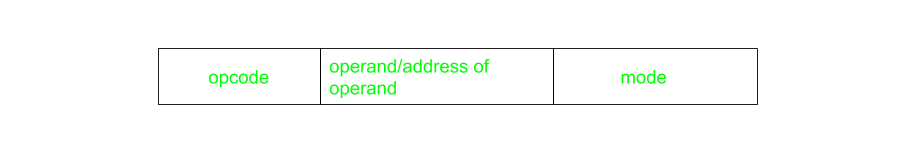
**Types of flag :**

* Sign flag (s)
* Zero flag (Z)
* Auxiliary Carry Flag (AC)
* Parity Flag (P)
* Carry Flag (CY)

**## Memory hierarchy**: Memory Hierarchy is one of the most required things in [Computer Memory](https://www.geeksforgeeks.org/computer-memory/) as it helps in optimizing the memory available in the computer. There are multiple levels present in the memory, each one having a different size, different cost, etc. Some types of memory like cache, and main memory are faster as compared to other types of memory but they are having a little less size and are also costly whereas some memory has a little higher storage value, but they are a little slower. Accessing of data is not similar in all types of memory, some have faster access whereas some have slower access. **Types of Memory Hierarchy** This Memory Hierarchy Design is divided into 2 main types: **--External Memory or Secondary Memory**:  Comprising of Magnetic Disk, Optical Disk, and Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via an I/O Module. **--Internal Memory or Primary Memory:**  Comprising of Main Memory, Cache Memory &[CPU registers](https://www.geeksforgeeks.org/different-classes-of-cpu-registers/). This is directly accessible by the processor.



**## Machine Language**: Machine code, also known as machine language, is the elemental language of computers. It is read by the computer's central processing unit ([CPU](https://www.techtarget.com/whatis/definition/processor)), is composed of digital [binary](https://www.techtarget.com/whatis/definition/binary) numbers and looks like a very long sequence of zeros and ones. Ultimately, the source code of every human-readable programming language must be translated to machine language by a compiler or an interpreter, because binary code is the only language that computer hardware can understand. The processor reads and handles [instructions](https://www.techtarget.com/whatis/definition/instruction), which tell the CPU to perform a simple task. Instructions are comprised of a certain number of [bits](https://www.techtarget.com/whatis/definition/bit-binary-digit). If instructions for a particular processor are 8 bits, for example, the first 4 bits part (the opcode) tells the computer what to do and the second 4 bits (the operand) tells the computer what data to use.

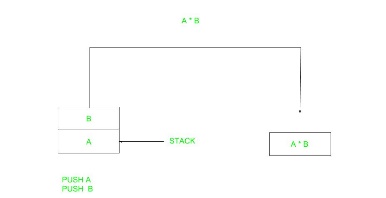
**## Subroutines :**  A subroutine is a sequence of program instructions that perform a specific task, packaged as a unit. This unit can then be used in programs wherever that particular task have to be performed. A subroutine is often coded so that it can be started (called) several times and from several places during one execution of the program, including from other subroutines, and then branch back (return) to the next instruction after the call, once the subroutine’s task is done. It is implemented by using Call and Return instructions. **Advantages of Subroutine : –**Decomposing a complex programming task into simpler steps. **--**Reducing duplicate code within a program. **--**Enabling reuse of code across multiple programs. **--**Improving tractability or makes debugging of a program easy.

**## Computer Register:** A processor register is one of a small set of data holding places that are part of the computer processor. A register may hold an instruction , a storage address, or any kind of data (such as a bit sequence or individual characters). Some instruction specify register as part of the instruction. For example: an instruction may specify that the contents of two defined registers be added together and then placed in a specified register.

**##** **Instruction Format:** In computer organization, instruction formats refer to the way instructions are encoded and represented in machine language. There are several types of instruction formats, including zero, one, two, and three-address instructions.

Each type of instruction format has its own advantages and disadvantages in terms of code size, execution time, and flexibility. Modern computer architectures typically use a combination of these formats to provide a balance between simplicity and power.

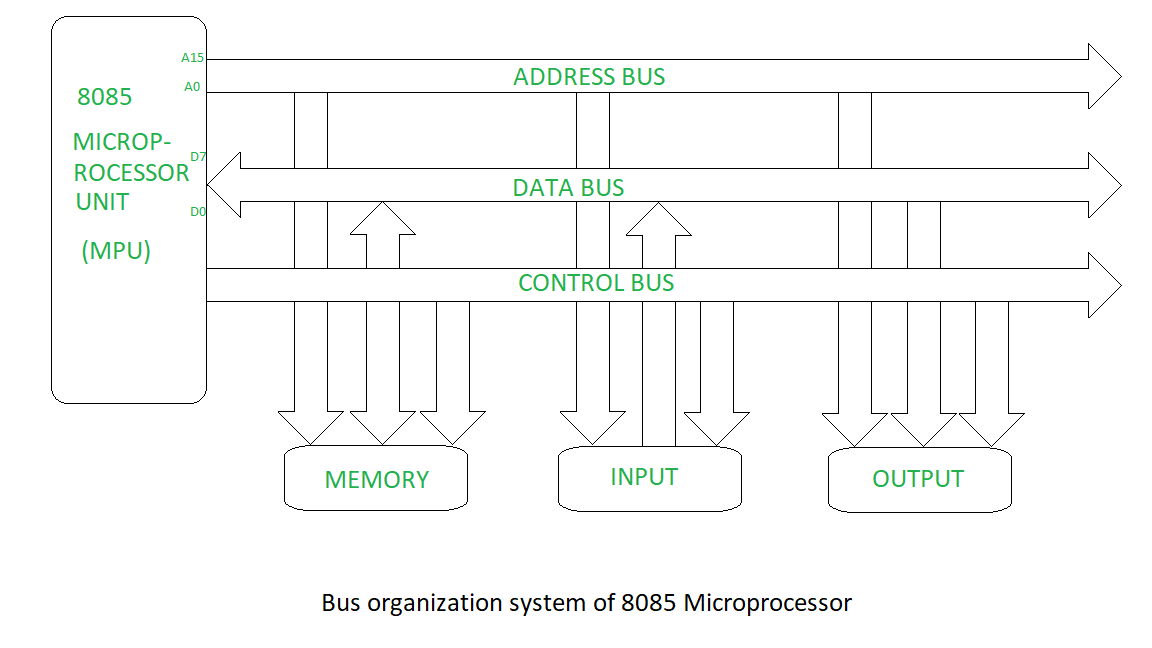
**Types of Instruction Format: Zero Address Instructions :** These instructions do not specify any operands or addresses. Instead, they operate on data stored in registers or memory locations implicitly defined by the instruction. For example, a zero-address instruction might simply add the contents of two registers together without specifying the register names.



**One Address Instructions :** These instructions specify one operand or address, which typically refers to a memory location or register. The instruction operates on the contents of that operand, and the result may be stored in the same or a different location. For example, a one-address instruction might load the contents of a memory location into a register.

**## Stack pointer**: A stack pointer is a small register that stores the memory address of the lastdata [element](https://www.techtarget.com/whatis/definition/element) added to the stack or, in some cases, the first available address in the stack. A [stack](https://www.techtarget.com/whatis/definition/stack) is a specialized buffer that is used by a program's functions to store data such as parameters, local [variables](https://www.techtarget.com/whatis/definition/variable) and other function-related information. The stack pointer -- also referred to as the extended stack pointer (ESP) -- ensures that the program always adds data to the right location in the stack. The stack stores data from the top down, following a last in, first out (LIFO) [data structure](https://www.techtarget.com/searchdatamanagement/definition/data-structure). This means that **the**[**program**](https://www.techtarget.com/searchsoftwarequality/definition/program)**adds** data to the top of the stack and removes data from the top of the stack. In this way, the top of the stack always contains the most recently stored data that has not yet been removed.

**## Define the bus organization in 8085 microprocessor with clear diagram? 🡪**Buses are the means by which information is shared between the registers in a multiple-register configuration system. A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during each particular register transfer. Von-Neumann Architecture comprised of three major bus systems for data transfer.



**Types of bus: Address Bus:** Address Bus carries the address of data (but not the data) between the processor and the memory. **Data Bus:** Data Bus carries data between the processor, the memory unit and the input/output devices. **Control Bus:** Control Bus carries signals/commands from the CPU.

**## Fixed point representation :** Fixed-point representation is a method for representing real numbers in a computer. It is particularly useful for systems where floating-point hardware is not available, or where a fixed precision is required, such as in embedded systems and digital signal processing (DSP). In fixed-point representation, numbers are expressed with a fixed number of digits after the decimal (or binary) point. **Advantages**: **--** Easier to implement than floating-point arithmetic. **--** Faster arithmetic operations on hardware that lacks floating-point support. **--** Fixed precision is beneficial for certain applications like DSP. **Disadvantages**: **--**Fixed range determined by the number of bits; can lead to overflow or underflow. --Proper scaling must be managed manually, which can be complex.

**##Define instructions set. Explain its type with suitable example. 🡪** An instruction set is a group of commands for a central processing unit in machine language. The term can refer to all possible instruction for a CPU or a subset of instruction to enhance its performance in certain situations. All CPUs have instruction sets that enable commands directing the CPU to switch the relevant transistors. The instructions tell the CPU to perform tasks . foe instance , the CPUs instructions might be 8 bits, where the first 4 bits make up the operation code that tells the computer what do to. \* \*\***Instruction types** : A computer instruction refers to a binary code that controls how a computer performs micro-operations in a series. They , together with the information, are saved in the memory. Every computer has its own set of instructions. **– Three Address Instruction:** A three-address instruction has the following general format : source 1 operation, source 2 operation, source 3 operation, destination ADD X, Y, Z **–Two Address instruction:** A two- address instruction has the following general format: source and destination of the operation ADD X, Y **--One Address Instruction:** one address instruction has the following general format: Operation source INCLUDE X

**## Floating Point representation:** Here the decimal point is not fixed after certain digit but it can change its position within a numeral. Example=25.9874 . The decimal point may be shifted in many ways and the exponent is to be taken suitably so that the value of the number is not affected.  **## Addition and subtraction with signed2’s complement data :** The left most bit of binary number represents the sign bit; 0 for positive and 1 for negative. If the sign bit is 1, the entire number is represented in 2’s compliment form . The addition of two numbers in signed-2’s complement form consists of adding the number with the sign bits treated the same as the other bits of the number. A carry out of the sign bit position is discarded. The subtraction consist of first taking the 2’s compliment of the subtrahend and then adding it to the number. When two numbers of n digits each are added and the sum occupation n+1 Digits, we say that an overflow occurred. **## What is Booth Algorithm ? Write its algorithm with suitable example.**

Booth's Algorithm is an efficient algorithm for multiplying binary integers in two's complement representation. It handles both positive and negative multipliers and multiplicands and reduces the number of required addition and subtraction operations**. Example** Let's consider multiplying two 4-bit numbers: 555 (0101) and −3-3−3 (1101 in two's complement). Initialization: --M=0101M = 0101M=0101 (multiplicand) -- Q=1101Q = 1101Q=1101 (multiplier) --Q−1=0Q-1 = 0Q−1=0 --A=0000A = 0000A=0000 --n=4n = 4n=4 \*\*\* **Steps:** **Iteration 1**: --Q0 = 1 and Q-1 = 0: A = A - M → A = 0000 - 0101 = 1011 (in two's complement) --Arithmetic right shift (A, Q, Q-1): A = 1101, Q = 1110, Q-1 = 1 n = 3 **Iteration 2:** ---Q0 = 0 and Q-1 = 1: A = A + M → A = 1101 + 0101 = 0010 ---Arithmetic right shift (A, Q, Q-1): A = 0001, Q = 0111, Q-1 = 0 n = 2 **Iteration 3: --**Q0 = 1 and Q-1 = 0: A = A - M → A = 0001 - 0101 = 1011 --Arithmetic right shift (A, Q, Q-1): A = 1101, Q = 1011, Q-1 = 1 n = 1 **Iteration 4: -**Q0 = 1 and Q-1 = 1: No operation on A. --Arithmetic right shift (A, Q, Q-1): A = 1110, Q = 1101, Q-1 = 1 n = 0 **Result:** The final result is the concatenation of A and Q, which is 11101101. This is the 8-bit binary representation of -15 (since the original signs of operands indicate that the result should be negative). ---Thus, 5×−3=−155 \times -3 = -155×−3=−15 which is correctly represented in binary two's complement as 11101101.

**## What is Microprocessor ? Explain about evolution of microprocessor in details**. 🡪 The 8085 microprocessor is an 8-bit microprocessor that was developed by Intel in the mid-1970s. It was widely used in the early days of personal computing and was a popular choice for hobbyists and enthusiasts due to its simplicity and ease of use. The architecture of the 8085 microprocessor consists of several key components, including the accumulator, registers, program counter, stack pointer, instruction register, flags register, data bus, address bus, and control bus. The accumulator is an 8-bit register that is used to store arithmetic and logical results. It is the most commonly used register in the 8085 microprocessor and is used to perform arithmetic and logical operations such as addition, subtraction, and bitwise operations. \*\*Evolution of Microprocessors 1. First Generation (1971-1972): Intel 4004:Released in 1971 by Intel. It was the first commercially available microprocessor.4-bit processor with 2300 transistors.Operated at a clock speed of 740 kHz.Used in calculators and basic embedded systems. **2. Second Generation (1973-1978): Intel 8008 and 8080:** Intel 8008 was released in 1972 as an 8-bit microprocessor.Intel 8080, launched in 1974, was an enhanced version of the 8008 with improved performance and more instructions.Used in early personal computers like the Altair 8800. **3. Third Generation (1978-1980s): Intel 8086/8088:** Released in 1978, the Intel 8086 was a 16-bit microprocessor with a 20-bit address bus.The 8088, a variant of the 8086 with an 8-bit external data bus, was used in the IBM PC, released in 1981, marking the beginning of the x86 architecture. **4. Fourth Generation (1980s-1990s): Intel 80386:** Released in 1985.A 32-bit microprocessor with enhanced performance and support for multitasking.Supported virtual memory, which allowed more efficient use of system memory. 5.**Seventh Generation (2000s-2010s): Intel Core Series:** Introduced in 2006 with the Core 2 Duo, followed by Core i3, i5, and i7.Emphasized power efficiency, multi-core processing, and improved performance per watt.

**## Von-Neimann’s Architecture:** 🡪The Von Neumann architecture, also known as the Von Neumann model or the Princeton architecture, is a computer architecture based on a 1945 description by the mathematician and physicist John von Neumann and others in the "First Draft of a Report on the EDVAC." This architecture describes a design for an electronic digital computer with components that include a processing unit, a control unit, memory to store both data and instructions, external mass storage, and input and output mechanisms. **Key Components of Von Neumann Architecture** **Arithmetic Logic Unit (ALU):** Performs arithmetic and logical operations. **Control Unit (CU):** Directs the operation of the processor, telling the ALU, memory, and input/output devices how to respond to program instructions. **Primary Memory (RAM):** Stores data and instructions that are currently being used by the CPU. In Von Neumann architecture, the same memory is used to store both data and instructions **Secondary Memory:** Non-volatile storage like hard drives or SSDs, used for storing data and programs not currently in use. **Input/Output (I/O) Mechanisms:** Devices that allow data to be inputted into the computer (like keyboards, mice) and outputted from the computer (like monitors, printers). ***Data Bus:*** Transfers data between the CPU, memory, and I/O devices. **Address Bus:** Carries the address of data (not the data itself) to be read or written.

**## General Register Organization:** The central processing unit is called the brain of the computer that performs data processing operations. Intermediate data is stored in the register set during the exaction of the instruction. The microoperation required for executing the instructions are performed by the arithmetic logic unit whereas the control units takes care of transfer of information among the registers and guides the ALU. The control units services the transfer of information among the register and instructs the ALU and which operation is to be performed . The computer instruction set is meant for providing the specifications for the design of the CPU.

**## Isolated vs. Memory mapped I/O:**

**Isolated** I/O uses a separate set of lines for reading and writing data to I/O devices, while memory-mapped I/O uses the same address space for both memory and I/O devices. Isolate I/O is more complex to implement than memory-mapped I/O , but it offers several advantages, such are : **-Improved security:** Because I/O device have their own separate address space, it is more difficult for them to interfere with the memory or other I/O devices . –**Greater flexibility:**  I/O devices can be added or removed from the system without affecting the memory address space. **– Improved reliability:** If an I/O device fails, it is less likely to affect the memory or other I/O devices. \* **Memory-Mapped I/O**  is simple to implement than isolated I/O, but it has some disadvantages, such as: **--Reduced security:**  Because I/O devices share the same address space as memory, it is easier for them to interfere with the memory or other I/O devices . –**Reduced flexibility:**  I/O devices cannot be added or removed from the system without affecting the memory address space. –**Reduced reliability :**  If an I/O device fails, it is more likely to affect the memory other I/O device.