COMBINED PREVIOUS YEAR QUESTIONS OF COMPUTER ARCHITECTURE AND ASSEMBLY LANGUAGE

SECTION A (very short answers)

1. What is the use of register? Name any two registers.

Ans 1. Registers are used to store data and instructions in a computer's central processing unit (CPU) and to speed up processes:

- **Program Counter (PC)**: Keeps track of the memory address of the next instruction to be executed
- **Instruction Register (IR)**: Stores the instruction that is currently being executed
- **Memory Address Register (MAR)**: Retrieves instructions and data from memory
- **Memory Data Register (MDR)**: Stores data that will be stored or fetched from memory

2. What do you mean by cycle stealing?

Ans 2. Cycle stealing is a memory sharing technique in computer architecture that allows a memory to serve two masters at once. One master is usually the central processing unit (CPU), and the other is usually an I/O channel or device controller.

In cycle stealing, the CPU program can start an operation on an I/O device and then continue the mainline program while the I/O device is

performing its operation. The CPU is only "tied up" for one cycle while a data character is being transferred.

3. Differentiate between hardwired and micro-programmed controls.

Ans 3.

Hardwired Control Unit	Micro programmed Control Unit
The hardwired control unit generates the processor's necessary control signals.	The micro programmed control unit generates the control signals using microinstructions.
In comparison to micro programmed control units, hardwired control units operate more quickly since the necessary control signals are produced by hardware.	Due to the use of microinstructions for signal generation, this is slower than the other.
It is difficult to change.	It is simple to change.
More expensive since logic gates must be used to implement everything.	Comparatively less expensive than hardwired control since control signals are solely produced using microinstructions.
Due to the complexity of the circuit design, it cannot handle complicated commands.	It can manage complex instructions.

4. Write a note on computer registers.

Ans 4.

- Computer registers are small storage spaces in a computer's processor that temporarily hold data while the CPU processes

instructions. They are also known as memory registers or processor registers.

- Registers can store many types of data, including: Instructions, Storage addresses, Bit sequences, Individual characters, and Dates.
- Registers help the CPU work faster by providing quick access to important information. They are made up of multiple flip-flops, which are electronic circuits that can store a single bit of information.

5. What do you understand by interleaved D.M.A.?

Ans5. In interleaved DMA, data is divided into smaller blocks and transferred between devices in an alternating pattern. This allows for parallel data transfers, which can reduce bottlenecks and optimize memory bus utilization.

Interleaved DMA is especially useful for real-time processing and high-speed data transfer.

In interleaved DMA, data transfer occurs without stopping the microprocessor. In contrast, in interleaving mode, the CPU is not blocked by DMA, but this mode is slower because the DMAC has to wait for access to system buses.

6. What is asynchronous data transfer ?

Ans6. Asynchronous data transfer is a method of sending data that doesn't require active synchronization between the sender and receiver. It's also known as start/stop transmission.

Here are some characteristics of asynchronous data transfer:

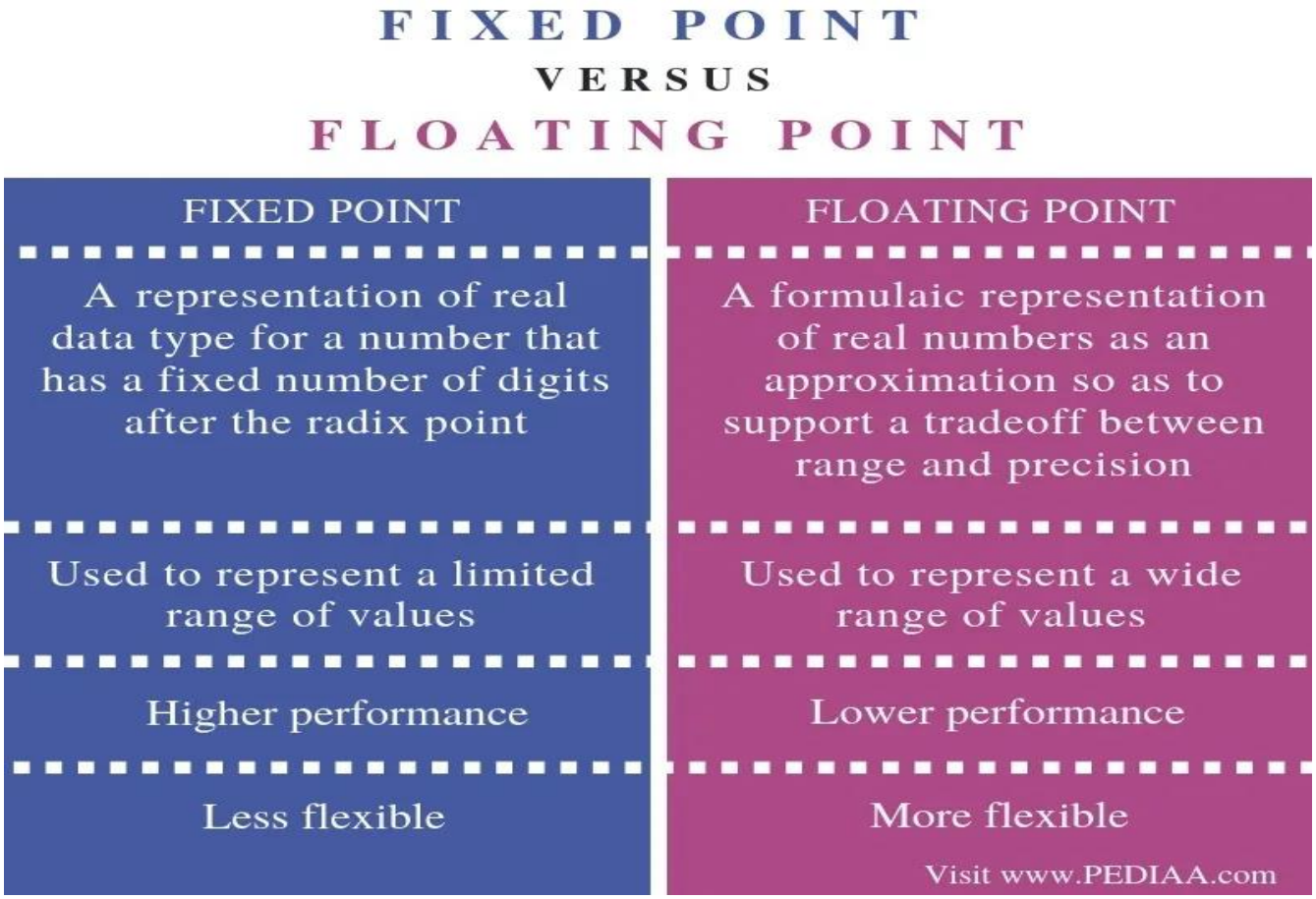
- **Start and stop bits**: Each character is preceded by a start bit and followed by one or more stop bits.

- **Character-based synchronization**: Data is transmitted one character or 8 bits at a time.

- **No two-way communication**: Asynchronous data transfer doesn't require two-way communication to work.
- **Receiver unaware**: The receiver is usually unaware of when data will arrive.
- **Parity bits**: Parity bits are used to inform the receiver about data translation.

7. Distinguish between fixed point and floating point representation.

Ans7.



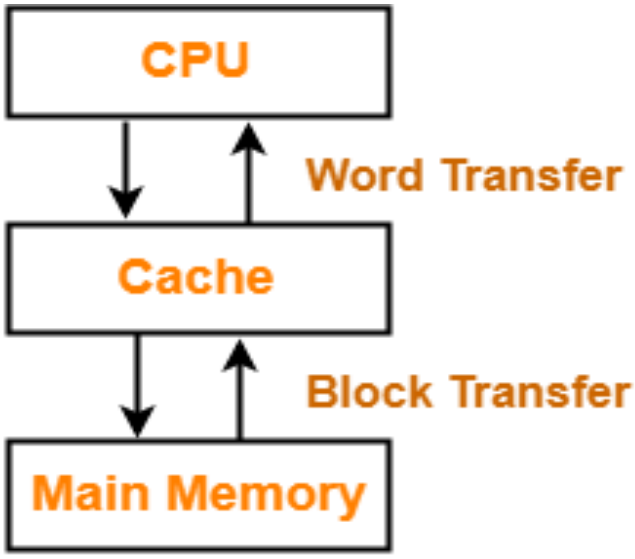
8. What are Macros ?

Ans8. In Computer Architecture, a macro is a set of rules or programmable patterns that translate a specific sequence of input into a predetermined sequence of output. Macros are typically used for a small set of instructions and can be used anywhere in a program by invoking its name.

Macros can be used to automate actions that are performed repeatedly or on a regular basis. For example, in Microsoft Mouse and Keyboard Center, you can record a macro of a sequence of events, such as keystrokes, mouse clicks, and delays, and assign it to a key or mouse button.

9. What is cache memory ? describe its operations.

Ans9. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. Cache Memory holds frequently requested data and instructions so that they are immediately available to the CPU when needed.



Cache and Main Memory

- **How it works**

  When a CPU needs data, it first checks the cache to see if it's already there. If it is, the cache returns the data to the CPU. If it's not, the CPU retrieves it from the main memory.

- **How it's organized**

  Cache memory is organized into levels, with the fastest portion being the register file. The next fastest is the Level 1 cache, which is located on the CPU. The Level 2 cache is connected to, but outside of, the CPU.

- **How it's used**

  Cache memory is used for multitasking, advanced applications, and processing large datasets.

- **How it's accessed**

  The percentage of accesses that result in cache hits is called the hit rate. When a cache miss occurs, the data is retrieved from the main memory and copied into the cache.

- **How it's replaced**

  When data is added to the cache, an existing entry is typically removed to make room. This is called the replacement policy. A popular replacement policy is least recently used (LRU), which replaces the oldest entry.

10. Define & explain Booth's algorithm.

Ans10. Booth's algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. It's a fundamental component of many processor designs and is often used in computer hardware and software.

Here's how Booth's algorithm works:

- **Recognize patterns**

  The algorithm identifies patterns in the binary representation of the multiplier.

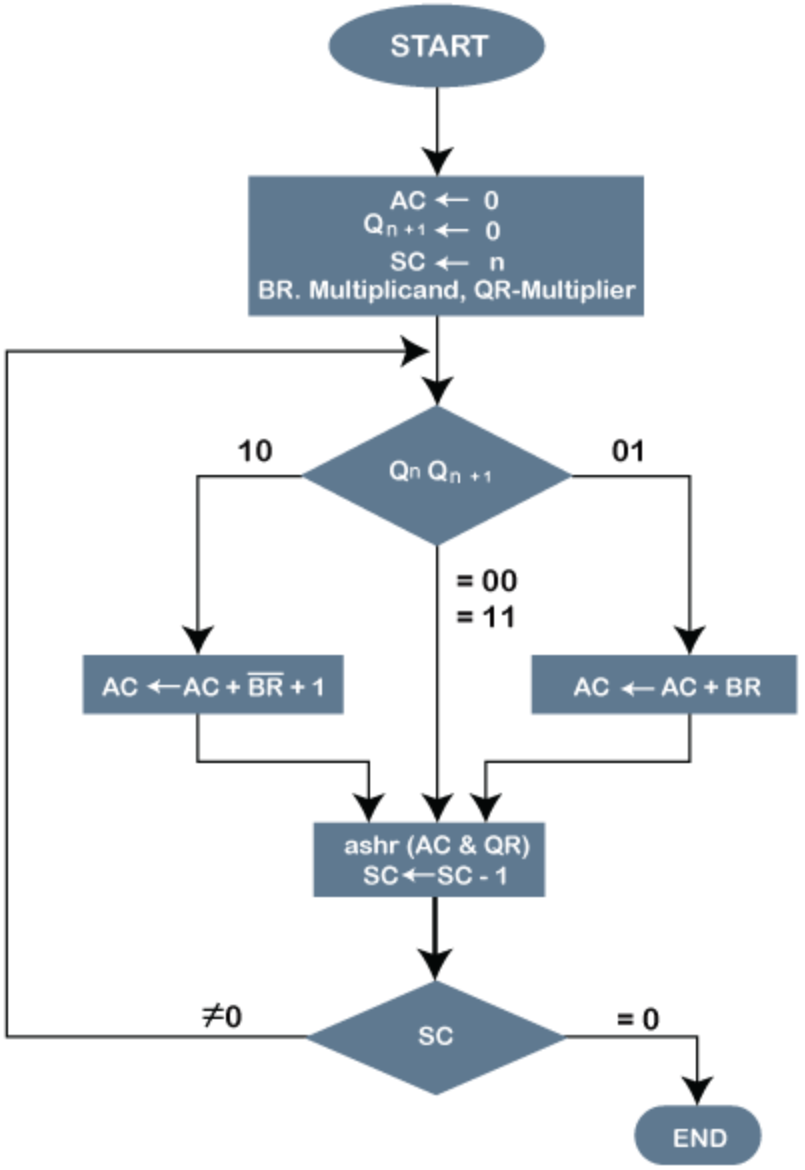
- **Adjust multiplicand**

  The algorithm adjusts the multiplicand based on the patterns it identifies.

- **Shift and add or subtract**

  The algorithm repeatedly shifts the multiplicand and adds or subtracts it from the partial product.

  - **Build product**
    The algorithm builds up the product one bit at a time in a third register.

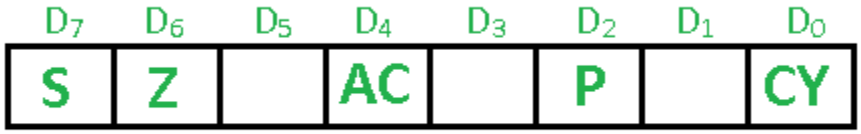


11. Write about Flag register in 8085.

Ans11. 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 flag register of 8085 microprocessor consists of 5 flags. The flag register is connected to ALU. When an operation is performed by ALU the result is transferred on data bus and status of result will be stored in flip

flops. The different flags and their positions in flag register are shown in following fig.



12. Write an assembly language program to add two nos.

Ans12.

```
;program for 8-bit addition in assembly language
programming

org 0000
mov a,#5
mov b,#5
add a,b
mov 50h,a
mov r0,50h
end
```

Program Description

- ORG 0000H means it set the statement at memory address 0000H.
- mov a,#5 using immediate Addressing mode we are transferring hexadecimal '5' to accumulator.
- mov b,#5 Same as above we are transferring hexadecimal 5 to b.
- add a,b here add opcode will adds the data of a and b
- mov 50h,a this line uses to store the output to the memory address 50h from accumulator.
- mov r0,50h this line uses to store the output to the register from memory 50h.
- end This 'end' opcode to stops the program
- To Write comments in Assembly language we use a "**;**".

SECTION B (Short answer Type)

1. Define interrupt. explain its various types.

Ans1. An interrupt is a signal that temporarily stops a processor's current activities to execute a specific service routine. Interrupts can be planned or unplanned, and they can be generated by hardware or software:

- **Planned interrupts**

  Requested by the currently running program

- **Unplanned interrupts**

  Caused by an event that may or may not be related to the currently running program

- **Hardware interrupts**

  Generated by hardware components, such as when a hardware component fails, an I/O is completed, or a timer expires

- **Software interrupts**

  Caused by software, such as when a program terminates, a service is requested, or a system call is made

2. Differentiate between RISC and CISC.

Ans2.

RISC	CISC
RISC refers to Reduced Instruction Set Computer.	CISC is the short form for Complex Instruction Set Computer.

RISC focuses more on software such as codes or compilers to execute instructions.	CISC focuses on hardware, such as transistors, to execute instructions.
RISC lacks special memory and thus utilizes specialized hardware to execute instructions.	Implementation of complex instructions is enabled through memory units.
RISC devices are embedded with a hardwired programming unit.	CISC devices are installed with a microprogramming unit.
RISC is provided with a reduced instruction set, which is typically primitive in nature.	CISC uses a variety of instructions to accomplish complex tasks.
RISC processors use hardwired units to control CPUs.	CISC processors are generally micro-coded, thereby allowing ROM-based CPU control. However, modern CISC processors also use hardwired units for easy CPU control.
A RISC processor utilizes 32 bits to execute each instruction.	A CISC processor works with 16 bits to 64 bits to execute each instruction.

3. What is serial communication. explain.

Ans3. Serial communication is a method of sending data between devices one bit at a time over a single wire or channel. It's a common way to

exchange information between computers and other devices, and is a key part of embedded systems.

Here are some things to know about serial communication:

- **How it works**

  Serial communication uses a digital binary method to exchange data. The data is sent at a specific speed, called the Baud rate, which is the number of bits sent per second.

- **Types of serial communication**

  There are two main types of serial communication: synchronous and asynchronous. Synchronous transmission uses a clock signal to send data, while asynchronous transmission doesn't.

- **Standards and protocols**

  There are many standards and protocols for serial communication, including RS-232C, RS-422A, RS-485, SPI, and I2C.

- **Transmission modes**

  Serial communication can use different transmission modes, including simplex and half-duplex:

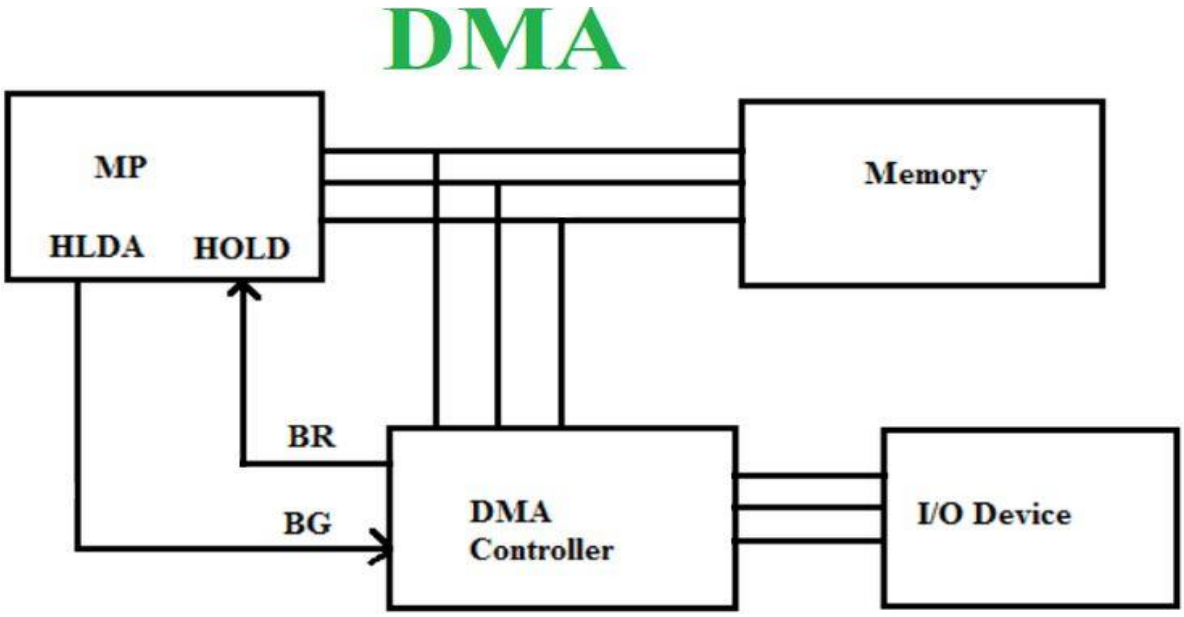
- **Simplex**: A one-way communication mode where only one device can transmit data at a time. Radio and television are examples of simplex mode.

- **Half-duplex**: A mode where both the sender and receiver can be active, but not at the same time. The internet is an example of half-duplex mode.

4. What do you understand by DMA. explain giving a diagram.

Ans4. Direct Memory Access (DMA) is a computer system feature that allows data to be transferred between devices and the main memory without the central processing unit (CPU). DMA is useful when the CPU is

unable to keep up with the data transfer rate or when it needs to perform other tasks while waiting for a slow data transfer.



Here's how DMA works:

1. An input-output device sends a DMA Request (DRQ) to the DMA controller.

2. The DMA controller asks the CPU to hold a few clock cycles by requesting a Hold request (HLD).

3. The CPU releases the bus and sends a Hold acknowledgment (HLDA) to the DMA controller.

4. The DMA controller transfers the data between the device and the main memory.

5. The DMA controller sends an interrupt to the CPU when the transfer is complete.

5. What is pipeline? Explain the four segment pipeline in reference of space time diagram.

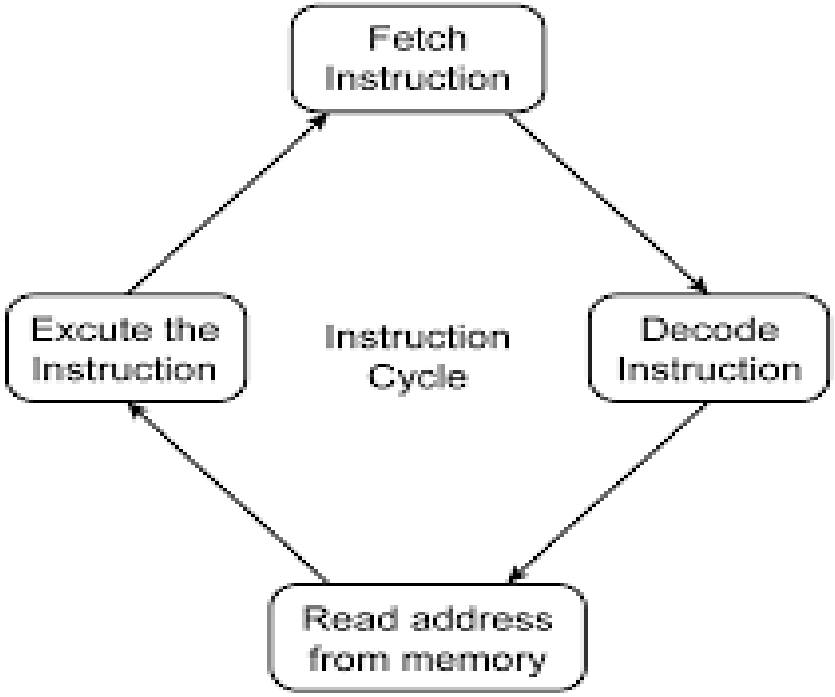
Ans5. A pipeline is a technique that breaks down a sequential process into sub-operations and executes them in parallel in different segments. The segments are connected to form a pipe-like structure, with instructions entering from one end and exiting from the other. The main advantage of pipelining is that it allows multiple computations to run at the same time.

A four-segment pipeline is an example of a pipeline that has the following stages:

- **Segment 1**: The instruction fetch segment, which can be implemented using a FIFO buffer.
- **Segment 2**: The memory instruction is decoded, and the effective address is determined
- **Segment 3**: Operands are fetched from memory
- **Segment 4**: The instructions are executed

6. What are the steps for a simple instrution cycle. Explain fetch cycle and indirect cycle using Register transfer language.

Ans6. The instruction cycle, also known as the fetch–decode–execute cycle, is the basic operational process of a computer.



The steps involved in the instruction cycle are:

- **Fetch**: The processor retrieves the instruction from memory. The address of the memory location where the instruction is stored is called the Program Counter (PC).

- **Decode**: The instruction decoder decodes the instruction from the Instruction Register (IR) to a set of control signals. These signals determine which operation to perform.

- **Execute**: The computer carries out the actions dictated by the instruction.

Explaination fetch and indirect behalf of RTL:

1. Fetch Cycle

The fetch cycle is responsible for retrieving an instruction from memory. The typical RTL for a fetch cycle can be described as follows:

```
1. MAR ← PC           // Load the Memory Address Register
   (MAR) with the Program Counter (PC)
2. MDR ← Memory[MAR] // Load the Memory Data Register
   (MDR) with the instruction at the address in MAR
3. IR ← MDR           // Transfer the instruction from MDR
   to the Instruction Register (IR)
4. PC ← PC + 1      // Increment the Program Counter (PC)
   to point to the next instruction
```

2. Indirect Cycle

The indirect cycle is used when an instruction specifies an address indirectly, typically through a pointer. The RTL for an indirect cycle might look like this:

```
1. MAR ← IR[address] // Load MAR with the address
   specified in the instruction register
2. MDR ← Memory[MAR] // Load MDR with the data at the
   address specified by MAR
3. A ← MDR           // Transfer the data from MDR to a
   general-purpose register (A) for further use
```

7. Explain programmed I/O with a flowchart.

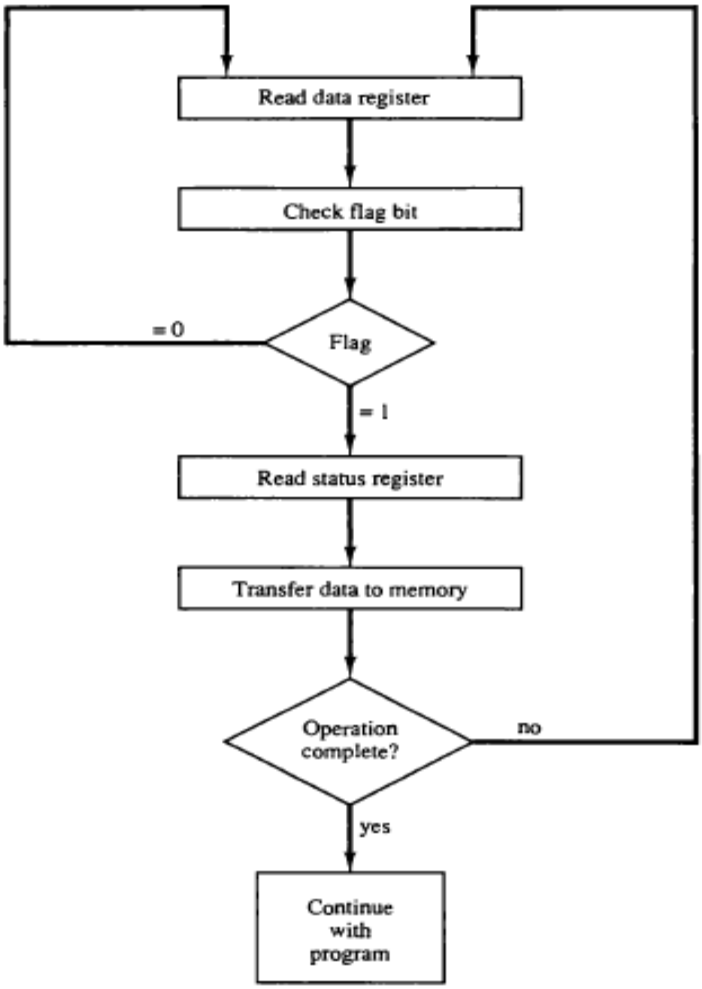
Ans7.

Programmable I/O is one of the I/O technique other than the interrupt-driven I/O and direct memory access (DMA). The programmed I/O was the most simple type of I/O technique for the exchanges of data or any types of communication between the processor and the external devices. With programmed I/O, data are exchanged between the processor and the I/O module. The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.

The overall operation of the programmed I/O can be summaries as follow:

1. *The processor is executing a program and encounters an instruction relating to I/O operation.*
2. *The processor then executes that instruction by issuing a command to the appropriate I/O module.*

3. *The I/O module will perform the requested action based on the I/O command issued by the processor (READ/WRITE) and set the appropriate bits in the I/O status register.*
4. *The processor will periodically check the status of the I/O module until it find that the operation is complete.*



8. Explain subroutine in assembly language.

Ans8. A subroutine is a block of code that performs a task based on some arguments and optionally returns a result. By convention, registers R0 to R3 are used to pass arguments to subroutines, and R0 is used to pass a result back to the callers.

Or

Subroutines are sequences of instructions that perform *specific tasks* and can be called from different parts of a program. A subroutine is a self-contained

module that is independent of the code that calls it. It performs *a specific task* and returns control to the calling program. Subroutines can be used to reduce the size of a program by reusing code, making the code more organized and readable.

How do Subroutines work?

When a subroutine is called, the program's control is transferred to the subroutine, and the subroutine performs its task. When the subroutine completes its task, it returns control to the calling program. *The calling program* can then continue from where it left off. Subroutines are usually called using a CALL instruction, and they are exited using *a RET instruction*.

Passing Parameters to Subroutines

Subroutines may require *some input data* to work on, which is known as parameters. Parameters can be passed to subroutines in different ways. One of the most common ways is by pushing the parameters onto the stack before calling the subroutine. The subroutine can then access the parameters from the stack.

Examples of Subroutines

Here's an example of *a simple subroutine* that adds two numbers and returns the result.

```
ADD_TWO_NUMBERS:
```

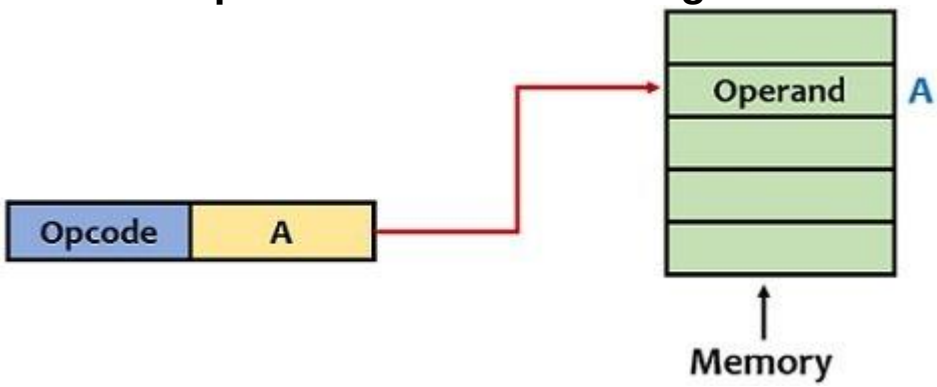
MOV AX, [BP+4] ; Get the first number from the stack

ADD AX, [BP+6] ; Add the second number to the first number

RET ; Return to the calling program with the result in AX

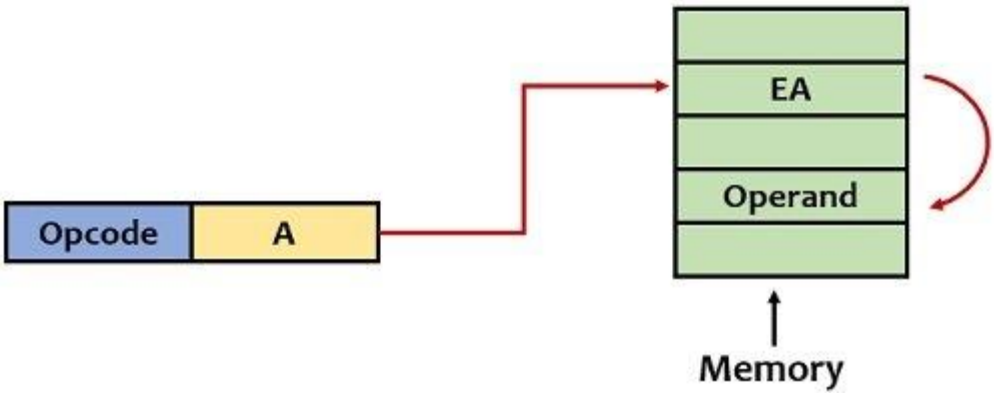
```

9. Differentiate between direct and indirect addressing with an example.

Ans9.

| **Basis for Comparison** | **Direct Addressing Mode** | **Indirect Addressing Mode** |
|---|---|---|
| Basic | It contains the actual address of the data. | It contains the effective address of the memory location where the actual address resides. |
| Principle | First, through instruction, the address is read and then data is read. | First, through the instruction, the effective address is read and then the actual address is read and lastly the data is achieved. |
| Speed of operation | Fast | Comparatively slow |
| Address space required | Small | Large |
| memory reference | Single | Double |
| Classification | No further classification exists. | Further classified into two categories namely register indirect and memory indirect. |
```

---->>**Example of Direct Addressing Mode :**



Then this instruction will be decoded as to get the actual operand over which operation is to be performed one must reach the memory location at address A.

---**---->>Example of Indirect Addressing Mode:**



This instruction will be decoded as to get the original operand the memory location at address A must be accessed. However, we will not get the data right there because at that particular address, another address will be present and on reaching that specific location data will be obtained. It is called so because the exact location is to be obtained indirectly.

10. Describe basic computer organization. How is it different from the computer architecture.

Ans10. Computer architecture describes how a computer system is designed, while computer organization explains how it works. Computer architecture is a blueprint for the design of a computer system, while

computer organization is how the operational parts of a computer system are linked together.

Here are some differences between computer architecture and computer organization:

- **Purpose**

  Computer architecture explains what a computer should do, while computer organization explains how it works.

- **Design**

  Computer architecture deals with high level design, while computer organization deals with low level design.

- **Actors**

  Actors in computer architecture are hardware parts, while actors in computer organization are performance.

- **Focus**

  Computer architecture focuses on the functional behavior of computer systems, while computer organization focuses on the structural relationship and internal working of a system.

SECTION C (Long Answer Type)

1. Write a short note on the following :

(a). Parallel Processor

(a). Parallel processing is a method in computing of running two or more processors, or CPUs, to handle separate parts of an overall task. Breaking

up different parts of a task among multiple processors helps reduce the amount of time it takes to run a program.

(b). Array Processor

(b). Array processing is a wide area of research in the field of signal processing that extends from the simplest form of 1 dimensional line arrays to 2 and 3 dimensional array geometries. Array structure can be defined as a set of sensors that are spatially separated, e.g. radio antenna and seismic arrays.

(c). Memory Interfacing

(c). Memory interfacing is a crucial aspect of computer system design that involves connecting various types of memory devices to the central processing unit (CPU) and other peripheral devices. Memory is essential for storing data and instructions that the CPU needs to process during program execution.

(d). Architecture of 8085

(d). 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

(e). Floating point representation

(e). Floating-point representation is a method for storing real numbers in digital computers using a base 2 exponent. It's used to convert input data into binary form, and then into scientific notation, before being represented as floating-point.

Floating-point representation has three parts:

- **Sign bit**: Indicates the sign of the number

- **Exponent**: Indicates the position of the decimal point
- **Mantissa**: The first part of the representation, which can be a fraction or an integer

Floating-point representation is more flexible than other methods and can handle a wide range of values.

Here's an example of how a decimal number is represented in floating-point:

- **Decimal number**: +6132.789
- **Floating-point representation**: +0.6132789 * 10 +4

(f). Index Register

(f). An index register in a computer's CPU is a processor register (or an assigned memory location) used for pointing to operand addresses during the run of a program. It is useful for stepping through strings and arrays. It can also be used for holding loop iterations and counters.

Index registers are commonly used to change the address of operands during program execution, especially when performing array or vector operations.

(g). Arithmetic Pipeline

(g). The arithmetic pipeline is a powerful tool in computer architecture that breaks down complex arithmetic problems into smaller, manageable subproblems. These subproblems are then processed in separate pipeline segments. This technique is widely used for operations such as multiplication and floating-point calculations.

(h). Instruction set

(h). An instruction set, also known as an instruction set architecture (ISA), is a collection of commands that a computer's processor can understand and execute. The ISA acts as a bridge between the software and the hardware, specifying what the processor can do and how it does it.

The ISA defines:

- The processor's capabilities and functionality
- The supported data types
- The registers
- How the hardware manages main memory
- Key features, such as virtual memory
- The input/output model

(i) Vector Processor

(i) A vector processor is a central processing unit (CPU) that executes instructions on arrays of data, or vectors, instead of individual data elements. This improves the performance of computing tasks that involve large amounts of data, such as numerical simulations, image processing, and artificial intelligence.

Here are some characteristics of vector processors:

- **Parallel processing**

  Vector processors use parallel processing, where multiple processors operate at the same time.

- **Vector processing instructions**

  Most modern CPU designs include some vector processing instructions, which are often called SIMD.

(j) General Register Organization

(j). General Register Organization (GRO) in computer architecture refers to the structure and usage of general-purpose registers in a CPU. These registers are used for various tasks, such as data manipulation and computation, during program execution.

GRO defines the following:

- **Number of registers**: The number of registers available
- **Size of registers**: The size of the registers
- **Roles of registers**: The specific roles of the registers, such as temporary data storage, addressing, and operand manipulation

(k) Input/Output Processor (IOP)

(k). In computer architecture, IOP stands for Input-Output Processor, which is a processor that manages input and output tasks. It's similar to a CPU, but only handles I/O processing.

Here are some things to know about IOP:

- **Direct memory access**: IOPs have direct memory access capabilities, which means they can directly access and store data into memory.
- **Interface**: IOPs act as an interface between the computer and I/O devices.
- **Freeing up the CPU**: IOPs free up the CPU from I/O data transfer operations. The CPU only informs the IOP of the I/O program's address in memory.

- **Raising interrupts**: After data transfer is complete, the IOP raises an interrupt to the CPU.

(l) Operation Code

(l). In computer architecture, an operation code (opcode) is a numeric code that tells the central processing unit (CPU) which operation to perform. Opcodes are used in hardware devices like CPUs and arithmetic logic units (ALUs), as well as in some software instruction sets.

Here are some more details about opcodes:

- **What they represent**

  Opcodes are a group of bits that represent basic operations like addition, subtraction, shift, and complement.

- **How they work**

  In CPUs, the opcode is part of a machine language instruction that specifies the operation. In ALUs, the opcode is applied to circuitry directly via an input signal bus.

(m) 8-bit microprocessor

(m). An 8-bit microprocessor is a computer hardware device or software program that can process and represent information using 8 binary digits, or bits, at a time. 8-bit microprocessors were the first microprocessors to be widely used in the computing industry, and their introduction in the 1970s led to the popularization of personal computers

(n) Program loops in assembly language

(n). In computer programming, a loop is a control structure that repeats a block of code or instructions until a certain condition is met. Loops are used

to simplify and optimize the coding process, and are especially helpful when dealing with complex or repetitive actions.

Here are some things to know about loops in assembly language:

- **Structure**

  A loop has two parts: a control statement and a body of the loop. The control statement contains the conditions that must be met for the loop to execute, and the body of the loop contains the block of code or logical statements that are executed multiple times.

- **Types**

  There are different types of loops, and each programming language has its own syntax. For example, there are for loops and while loops.

2. What is priority interrupt ? Explain polling and chaining periority.

Ans2. In computer architecture, a priority interrupt is a mechanism that allows the CPU to prioritize and respond to I/O requests from devices:

- **How it works**

  When multiple devices send interrupt signals simultaneously, the CPU will service the device with the highest priority first. This allows the CPU to suspend its current operation and respond to the most important requests.

- **How priority is established**

  Priority is established by assigning higher priority levels to requests that could have serious consequences if delayed or interrupted. For example,

high-speed transfer devices like magnetic disks are given higher priority than slow devices like keyboards.

- **How priority is established in software or hardware**
  Priority can be established using software or hardware. One method, called the parallel priority interrupt method, uses a register to set bits separately for each device's interrupt signal. The position of the bits in the register determines the priority.
  In computer architecture, polling and daisy chaining are two methods for establishing priority for interrupts:

- **Polling**

  A software method that involves periodically checking the status of devices to see if they need attention

- **Daisy chaining**
  A hardware method that involves connecting devices in a serial chain, with the device closest to the CPU having the highest priority

  Here are some more details about these methods:

- **Daisy chaining**

  This method is straightforward to implement and provides deterministic priority. The device with the highest priority is placed first in the chain, and the device with the lowest priority is placed last. The interrupt request line is shared by all devices, and when a device has an interrupt signal, the interrupt line goes low, enabling the CPU's interrupt input.

- **Polling**

  This method involves sequentially checking interrupts in software to determine priority.

- **Hardware methods**

Hardware methods like daisy chaining allow for faster interrupt handling than software polling

3.

(a) List the five important characteristics of Risc Architecture.

(a). Here are five important characteristics of RISC architecture:

- **Simple instructions**: RISC instructions are simple and standardized, and can be easily decoded and implemented.
- **Single clock cycle**: RISC instructions are executed in a single clock cycle.
- **Small instruction set**: The instruction set is small, sometimes as small as a single word.
- **Large number of registers**: RISC has a large number of general-purpose registers.
- **Simple addressing modes**: RISC has simple addressing modes.

(b) explain the need of different addressing mode by taking suitable example.

(b). There are some following need of different addressing mode:

1. **Flexibility in Data Access**: Different addressing modes allow programmers to access data in various ways, enabling more flexible and efficient coding. This is particularly useful for data structures like arrays, linked lists, and complex data types.
2. **Efficiency**: Certain addressing modes can reduce the number of instructions needed to perform an operation. For example, modes

like indirect addressing can simplify accessing data stored in memory, which can lead to more efficient code execution.

3. **Memory Management**: Different modes help manage memory effectively, allowing access to data in different segments (like stack, heap, etc.). For example, base-register addressing can help in accessing data structures that reside in different memory segments.
4. **Ease of Programming**: High-level programming languages often abstract away the details of memory addressing. By providing multiple addressing modes, microprocessors can better support these abstractions, making it easier for programmers to write efficient code without needing to manage memory manually.
5. **Performance Optimization**: Some addressing modes can be optimized for specific tasks, such as immediate addressing for constants or indexed addressing for iterative operations. This optimization can lead to better performance in specific applications.
6. **Hardware Simplification**: Different addressing modes allow the hardware to be simpler and more efficient. For instance, using a mode that relies on registers can speed up operations compared to accessing memory directly.

4.

(a). What are various data transfer schemes ? Briefly discuss each scheme.

(a). Data transfer schemes can be classified into two modes – Serial Data Transfer and Parallel Data Transfer. The Intel 8085 Microprocessor, is a parallel device. Thus, it transfers 8 bits of information simultaneously over 8 data lines in the parallel I/O mode.

Here are some data transfer schemes:

- **Direct Memory Access (DMA)**

This is the fastest method for transferring data between a device and computer memory. It allows direct transfer between memory and I/O devices without the CPU.

- **Interrupt Request (IRQ)**

  This method uses the CPU to service data transfer requests. The device notifies the CPU when it is ready to transfer data.

- **Programmed I/O**

  This method involves reading and writing directly to the device without using a buffer. It is typically used for software-timed (on-demand) operations.

- **Synchronous transfer**

  This method uses precise timing to coordinate data transfer with I/O device speed.

- **Asynchronous and interrupt-driven transfers**

  These methods use handshaking or interrupt signals to reconcile μP and I/O device timing differences.

- **Serial data transfer**

  This method transfers data one bit at a time. It is preferred when data is to be sent over a long distance.

- **Parallel data transfer**

  This method transfers data several bits at the same time. It is preferred for short-distance communication.

(b) A CISC chip is a complex instruction set computing chip the alternative to RISC chips. How do they differ ?

(b). The main difference between CISC (Complex Instruction Set Computer) and RISC (Reduced Instruction Set Computer) chips is how they execute instructions:

- **CISC**

  CISC chips can perform multiple actions with a single instruction, which can take multiple clock cycles to complete. CISC instructions can have different lengths, which can increase processing time. CISC chips also require more transistors to decode complex instructions.

- **RISC**

  RISC chips execute one instruction per clock cycle, and each instruction has a set memory size. RISC chips focus on executing commands and reducing the number of cycles per instruction.

  The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction. RISC does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program.

5. What do you understand by Instruction Cycle ? What are the different phases of instruction cycle ?

Ans5. The instruction cycle is defined as the basic cycle in which a computer system fetches an instruction from memory, decodes it, and then executes it. Fetch-Execute-Cycle is another name for it. All instructions in a computer system are executed in the RAM of the computer system.

The instruction cycle has four phases:

- **Fetch**

The processor copies the instruction data from the RAM. The program-counter mechanism obtains the address of the next instruction and transfers it to the memory address register (MAR).

- **Decode**

  The CPU determines which instruction to fetch and what action to take. The instruction's opcode is retrieved from memory, and the related operation is decoded.

- **Read the address from memory**

  The address is read from memory.

- **Instruction execution**

  The instruction is executed. The instruction execution may involve several operations and depends on the nature of the instruction.

6. Difference Between :

(a) Register Stack and Memory Stack

(a).

Register Stack	Memory Stack
Registers hold operands or instructions that CPU would be currently processing.	Memory holds instructions and the data about the currently executing program required by the CPU.
They contain small amounts of data– 32bits to 64bits.	Memory can range from some GB (Giga bytes) to TB (Tera bytes).

CPU can operate on the contents of the register at the rate of more than one operation during one clock cycle.	The CPU accesses memory at a slower rate in comparison to memory access in a register.
There are many types of registers– Accumulator register, Program counter, Instruction register, Address register.	There are different types of memory– RAM, ROM.
They can be controlled, i.e. information can be stored and retrieved from them.	The memory can’t be controlled.
It is quick in comparison to memory.	RAM is slow in comparison to registers.

(b) General Purpose and Special Purpose Register

(b). The main difference between general purpose registers (GPRs) and special purpose registers is that GPRs can be used for a variety of purposes, while special purpose registers have a specific function:

- **General purpose registers**

  These versatile registers can be used for many functions, including storing addresses, calculating data, and temporarily storing data. They can also hold the results and operands of logical and arithmetic operations. GPRs provide flexibility in programming because they aren't restricted to a single purpose.

- **Special purpose registers**

  These registers are designed for specific control tasks within the processor. They have fixed functions, and putting data into them changes the functionalities in the processor or microcontroller. For example, the Program Counter (PC) keeps track of the next instruction to be executed, while the Instruction Register (IR) holds the current instruction being executed.

7. What is Addressing Mode. Explain Different types of addressing modes. Differentiate Direct & Indirect addressing mode.

Ans7.The **addressing mode** is the method to specify the operand of an instruction. The job of a microprocessor is to execute a set of instructions stored in memory to perform a specific task.

Operations require the following:

1. The operator or opcode which determines what will be done
2. The operands which define the data to be used in the operation

For example, if we wanted to add the numbers 1 and 2 and get a result, mathematically we would likely write this as 1 + 2. In this case, our operator is (+), or the addition, and our operands are the numbers 1 and 2.

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 Addressing Mode

1. Implied Addressing Mode-

In this addressing mode,

- The definition of the instruction itself specify the operands implicitly.
- It is also called as **implicit addressing mode**.

Examples-

- The instruction “Complement Accumulator” is an implied mode instruction.
- In a stack organized computer, Zero Address Instructions are implied mode instructions.

(since operands are always implied to be present on the top of the stack)

2. Stack Addressing Mode-

In this addressing mode,

- The operand is contained at the top of the stack.

Example-

- This instruction simply pops out two symbols contained at the top of the stack.
- The addition of those two operands is performed.
- The result so obtained after addition is pushed again at the top of the stack.

3. Immediate Addressing Mode-

In this addressing mode,

- The operand is specified in the instruction explicitly.
- Instead of address field, an operand field is present that contains the operand.

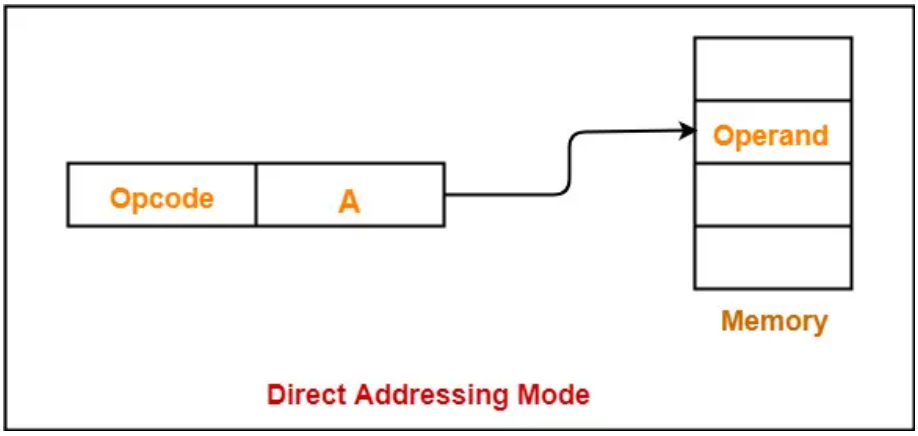
Examples-

- ADD 10 will increment the value stored in the accumulator by 10.
- MOV R #20 initializes register R to a constant value 20.

4.Direct Addressing Mode-

In this addressing mode,

- The address field of the instruction contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand.
- It is also called as **absolute addressing mode**.



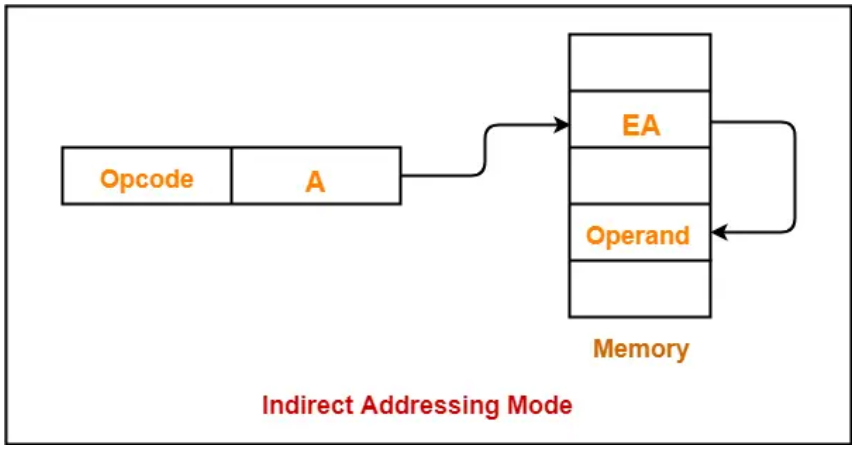
- ADD X will increment the value stored in the accumulator by the value stored at memory location X.

AC ← AC + [X]

5. Indirect Addressing Mode-

In this addressing mode,

- The address field of the instruction specifies the address of memory location that contains the effective address of the operand.
- Two references to memory are required to fetch the operand.



Example-

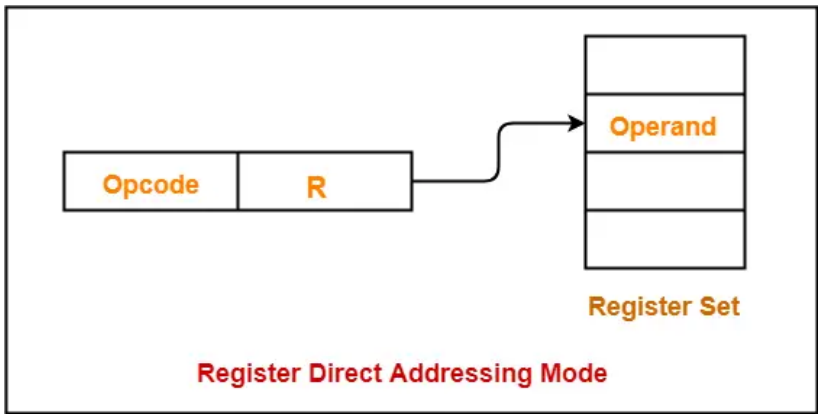
- ADD X will increment the value stored in the accumulator by the value stored at memory location specified by X.

$$AC \leftarrow AC + [[X]]$$

6. Register Direct Addressing Mode-

In this addressing mode,

- The operand is contained in a register set.
- The address field of the instruction refers to a CPU register that contains the operand.
- No reference to memory is required to fetch the operand.



Example-

- ADD R will increment the value stored in the accumulator by the content of register R.

AC ← AC + [R]

NOTE-

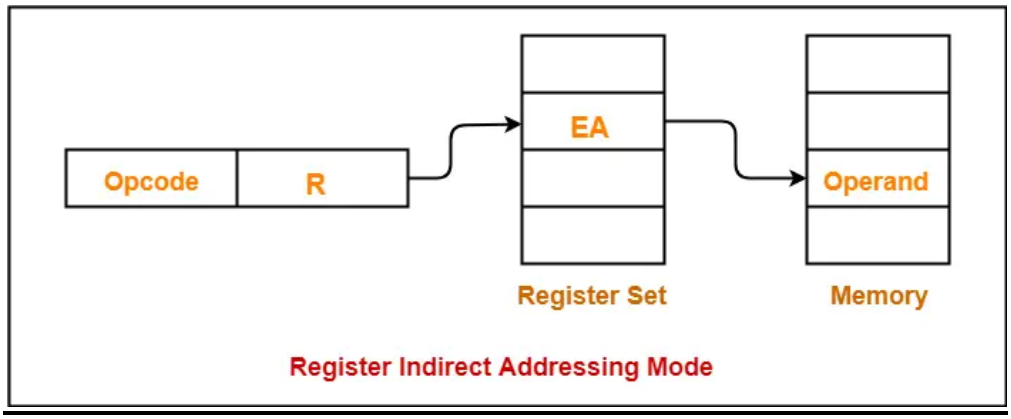
It is interesting to note-

- This addressing mode is similar to direct addressing mode.
- The only difference is address field of the instruction refers to a CPU register instead of main memory.

7. Register Indirect Addressing Mode-

In this addressing mode,

- The address field of the instruction refers to a CPU register that contains the effective address of the operand.
- Only one reference to memory is required to fetch the operand.



Example-

- ADD R will increment the value stored in the accumulator by the content of memory location specified in register R.

$$AC \leftarrow AC + [[R]]$$

NOTE-

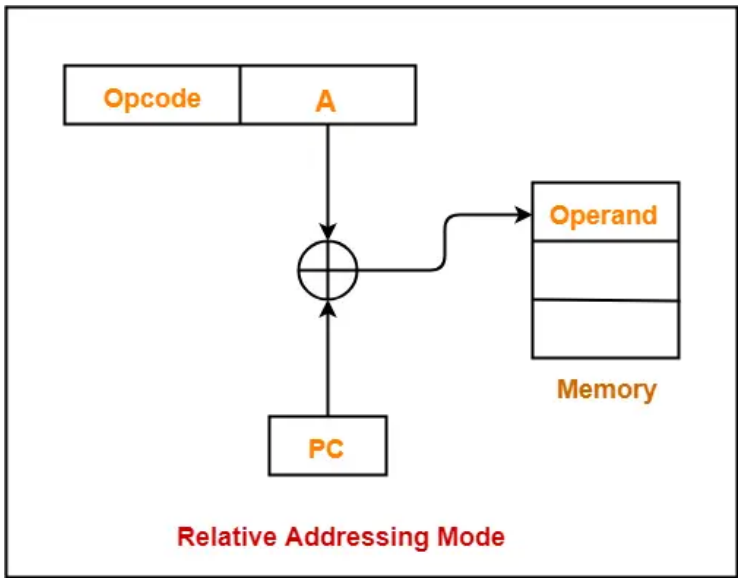
It is interesting to note-

- This addressing mode is similar to indirect addressing mode.
- The only difference is address field of the instruction refers to a CPU register.

8. Relative Addressing Mode-

In this addressing mode,

- Effective address of the operand is obtained by adding the content of program counter with the address part of the instruction.



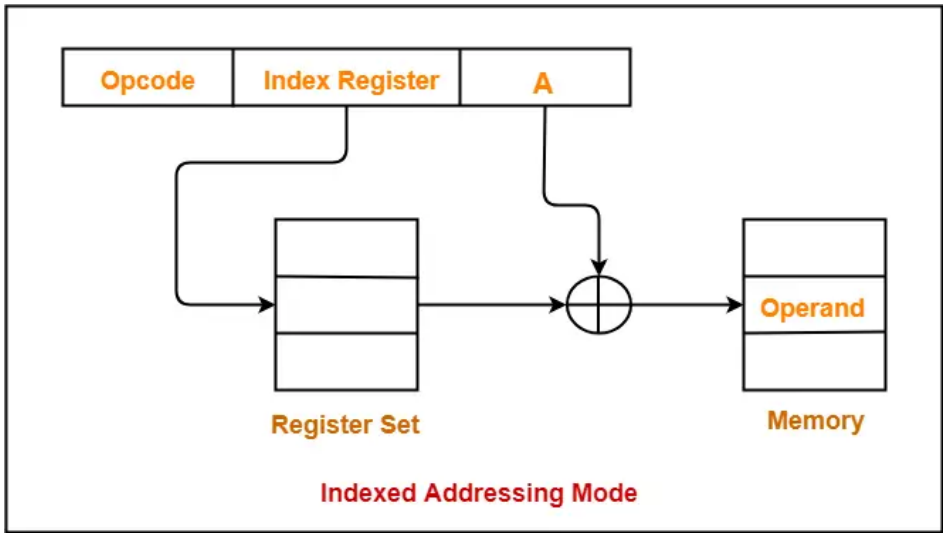
NOTE-

- **Program counter** (PC) always contains the address of the next instruction to be executed.
- After fetching the address of the instruction, the value of program counter immediately increases.
- The value increases irrespective of whether the fetched instruction has completely executed or not.

9. Indexed Addressing Mode-

In this addressing mode,

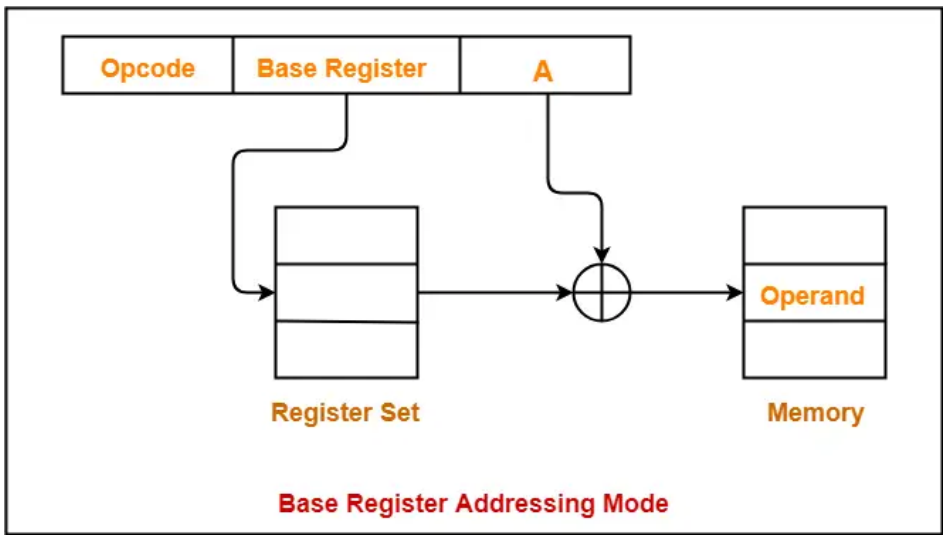
- Effective address of the operand is obtained by adding the content of index register with the address part of the instruction.



10. Base Register Addressing Mode-

In this addressing mode,

- Effective address of the operand is obtained by adding the content of base register with the address part of the instruction.



- Differenciate between direct and indirect addressing mode :

Direct Addressing Mode	Indirect Addressing Mode

In this mode, the address field contains the effective address of the operand.	In this mode, the address field contains the effective address of the operand.
This addressing mode requires one memory reference only.	It requires two memory references.
It is a quick addressing mode.	It is slower in comparison to direct addressing mode
There is no further classification in this mode.	It can be further classified into two categories.
There are no further calculations required to perform the operation in this addressing mode.	It requires further calculation to determine the effective address of the operand.

THANKS FOR STUDYING IN THIS USEFUL PDF

I HOPE YOU LIKE IT

JAI HIND JAI BHARAT

ABHINAV KAUSHIK (TEAM ADMIN)