II - ASSIGNMENT

(Start Writing From Here)

1. Explain about Addressing modes of 8086?

Ans The different ways in which a source operand is denoted in an instruction is known as addressing modes. There are 8 different addressing modes in 8086 programming.

1. Immediate addressing mode:-

The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

Example:- MOV CX, 4929H, ADD AX, 2387H, MOV AL, FFH

2. Register addressing mode:-

It means that the register is the source of an operand for an instruction.

Example:- MOV CX, AX; copies the contents of the 16-bit Ax register into; the 16-bit CX register),
ADD BX, AX

3. Direct addressing mode:-

The addressing mode in which the effective address of the memory location is written directly in the instruction.

Example:- MOV AX, [1592H], MOV AL, [0300H]

4. Register indirect addressing mode:-

This addressing mode allows data to be addressed at any memory location through an offset address held in any of the

following registers: BP, BX, DI & SI.

Example:- MOV AX, [BX]; Suppose the register BX contains 4895H, then the contents; 4895H are moved to AX

ADD CX, [BX]

5. Based addressing mode:-

In this addressing mode, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement.

Example:- MOV DX, [BX+04], ADD CL, [BX+08]

6. Indexed addressing mode:-

In this addressing mode, the operands offestet address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements.

Example:- MOV BX, [SI+16], ADD AL, [DI+16]

7. Based-index addressing mode:-

In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.

Example:- ADD CX, [AX+SI], MOVAX, [AX+DI]

8. Based indexed with displacement mode:-

In this addressing mode, the operands offset is computed

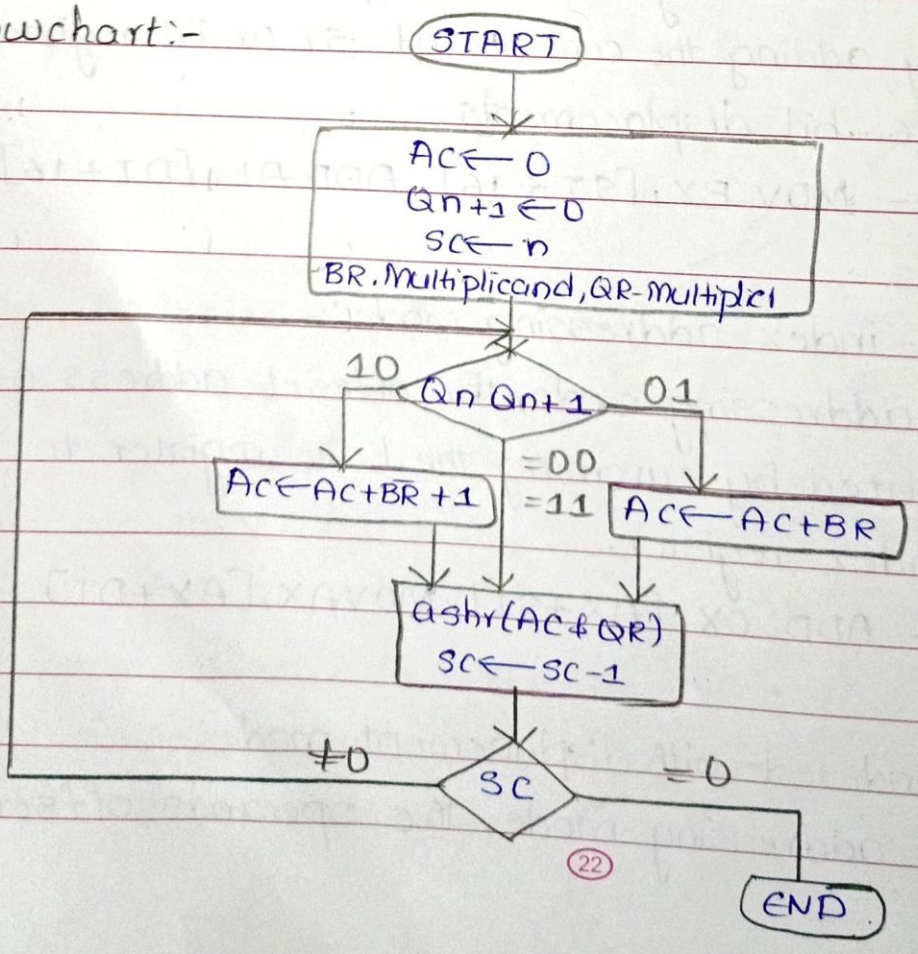
by adding the base register contents. An Index register contents and 8 or 16-bit displacement.

Example:- MOV AX, [BX+DI+08], ADD CX, [BX+SI+16]

2. With an example, explain Bell Booth multiplication algorithm.

Ans The booth algorithm is a multiplication algorithm that allows us to multiply the two signed binary integers in 2's complement, respectively. It is also used to speed up the performance of the multiplication process. It is very efficient too. It works on the string bits 0's in the multiplier that requires no additional bit only shift the right-most string bits and a string of 1's in a multiplier bit weight 2^k to weight 2^m that can be considered as $2^{k+1} - 2^m$.

Flowchart:-



Example:- Multiply the two numbers 7 and 3 by using the Booth's multiplication algorithm.

Here we have two numbers, 7 and 3. First of all, we need to convert 7 and 3 into binary numbers like 7 = (0111) and 3 = (0011). Now set 7 (in binary 0111) as Multiplicand (M) and 3 (in binary 0011) as a multiplier (Q). And SC (sequence count) represents the number of bits, and here we have 4 bits, so set the SC = 4. Also, it shows the number of iteration cycles of the booth's algorithm and then cycles run SC = SC-1 times.

Qn	Qn+1	M = (0111) M'+1 = (1001) & Operation	AC	Q	Qn+1	SC
1	0	Initial	0000	0011	0	4
		Subtract (M'+1)	1001			
			1001			
		Perform Arithmetic Right shift operations (ashr)	1100	1001	1	3
1	1	Perform Arithmetic Right shift operations (ashr)	1110	0100	1	2

Qn	Qn+1	M = (0111) M' + 1 = (1001) & operation	AC	Q	Qn+1	SC
0	1	Addition (A+M)	0111 0101	0100		
		Perform Arithmetic right shift operation	0010	1010	0	1
0	0	Perform Arithmetic right shift operation	0001	0101	0	0

The numerical example of the Booth's Multiplication Algorithm is 7x3 = 21 and the binary representation of 21 is 10101. Here, we get the resultant in binary 00010101. Now, we convert it into decimal, as $(00010101)_{10} = 2*4 + 2*3 + 2*2 + 2*1 + 2*0$

$\Rightarrow 21.$

3. Write short notes on I/O processor.

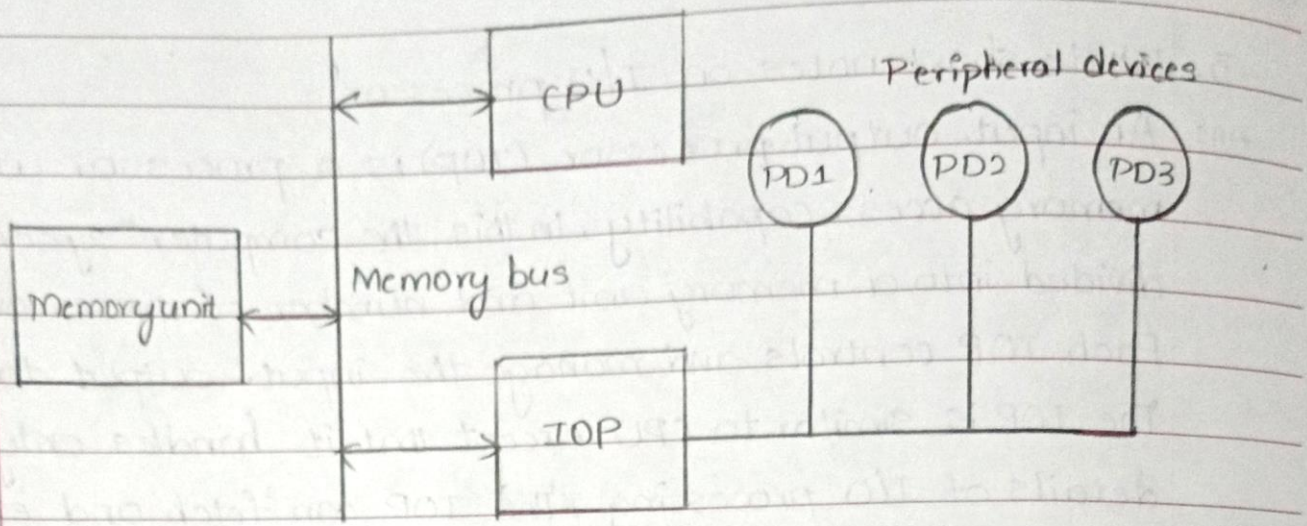
Ans An input-output processor (IOP) is a processor with direct memory access capability. In this, the computer system is divided into a memory unit and number of processors. Each IOP controls and manage the input-output tasks. The IOP is similar to CPU except that it handles only the details of I/O processing. The IOP can fetch and execute its own instructions. These IOP instructions are designed to manage I/O transfers only.

Block Diagram of I/O Processor

Below is a block diagram of a computer along with various I/O processors. The memory unit occupies the central position and can communicate with each processor.

The CPU processes the data required for solving the computational tasks. The IOP provides a path for transfer of data between peripherals and memory. The CPU assigns the task of initiating the I/O program.

The IOP operates independent from CPU and transfer data between peripherals and memory. The communication blw the IOP and the devices is similar to the program control method of transfer. And the communication with the memory is similar to the direct memory access method.



In large scale computers, each processor is independent of other processors and any processor can initiate the operation.

The CPU can act as master and the IOP acts as slave processor. The CPU assigns the task of initiating operations but it is the IOP, who executes the instructions, and not the CPU. CPU instructions provide operations to start an I/O transfer. The IOP asks for CPU through interrupt.

Instructions that are read from the memory by an IOP are also called commands to distinguish them from instructions that are read by CPU. Commands are prepared by programmers and are stored in memory. Command words make the program for IOP. CPU informs the IOP where to find the commands in memory

Features :-

- Specialized Hardware
- Protocol handling.
- DMA Capability
- Interrupt Handling

Applications:-

- Data Acquisition Systems
- Industrial Control Systems
- Multimedia Applications
- Network Communication Systems:-

Advantages:-

- Improved Data Transfer Rates
- Reduced Processor Workload
- Increased System Reliability
- Scalability.

Disadvantages:-

- Increased Complexity
- Synchronization Issues
- Limited Performance Gains
- Lack of Standardization.

4. What is Cache memory? Explain different types of cache memory mapping techniques.

Ans Cache memory is a chip-based computer component that makes retrieving data from the computer's memory more efficient. It acts as a temporary storage area that the computer's processor can retrieve data from easily

Different types of chache memory mapping techniques

- Direct Mapping
- Associative Mapping

Set-Associative Mapping

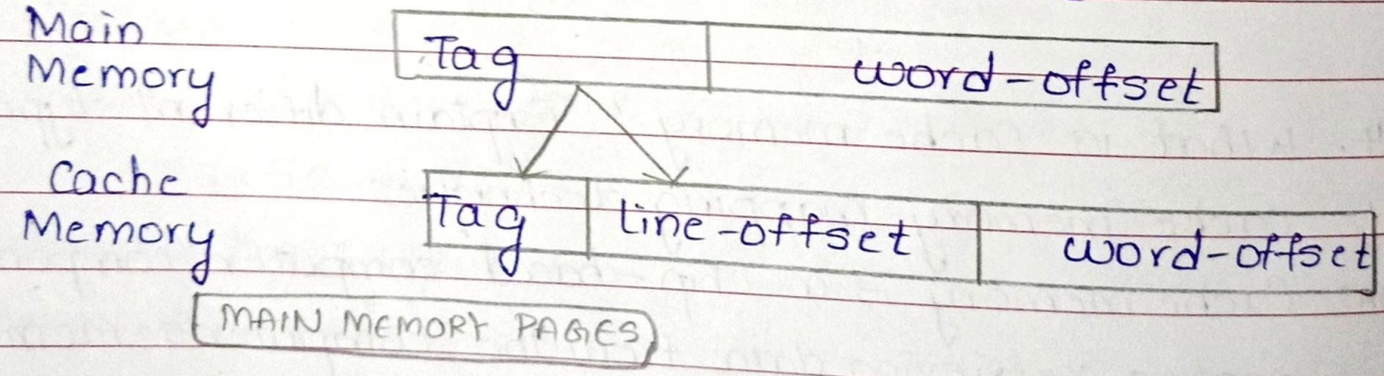
1. Direct Mapping :- The simpliest technique, known as direct mapping, maps each block of main memory into only one possible cache line. or In Direct mapping, assign each memory block to a specific line in the cache. If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed. An address space is split into two parts index field. ~~The~~ and a tag field. The cache is used to store the tag field whereas the rest is stored in the main memory. Direct Mapping's performance is directly proportional to the Hit ratio.

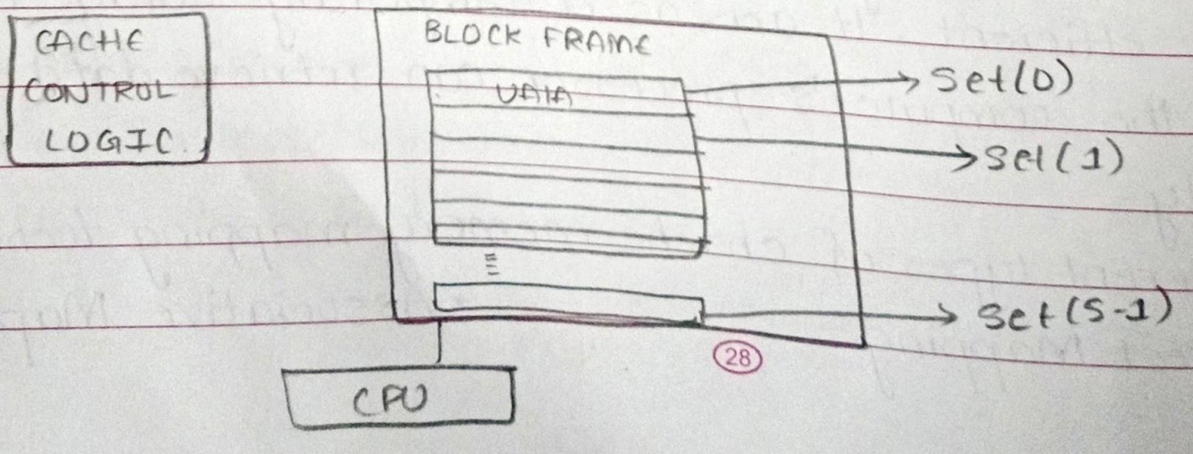
$i = j \text{ modulo } m$

where, i = cache line number

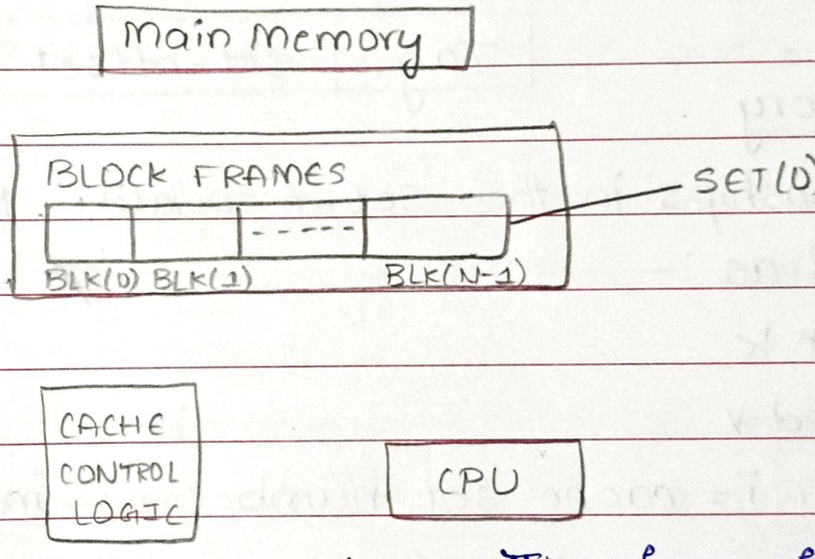
j = main memory block number

m = number of lines in the cache



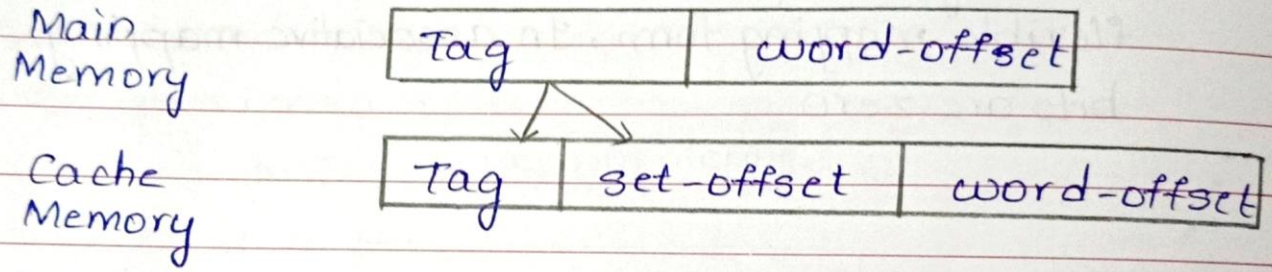


2. Associative Mapping:- In this type of mapping, associative memory is used to store the content and addresses of the memory word. Any block can go into any line of the cache. This means that the word id bits are used to identify which word in the block is needed, but the tag becomes all of the remaining bits. This enables the placement of any word at any place in the cache memory. It is considered to be the fastest and most flexible mapping form. In associative mapping, the index bits are zero.



3. Set-Associative Mapping:- This form of mapping is an enhanced form of direct mapping where the drawbacks of direct mapping are removed. Set Associate addresses the problem of possible thrashing in the direct mapping method. It does this by saying that instead of having exactly one line that a block can map to in the cache, we will group a few lines together creating a set. Then a block in memory can map to any one of the lines of

a specific set. Set-associative mapping allows each word that is present in the cache can have two or more words in the cache can have two or more words in the main memory for the same index address. set associative cache mapping combines the best of direct and associative cache mapping the index bits are given by the set offset bits. In this case, the cache consists of a number of sets, each of which consists of a number of lines.

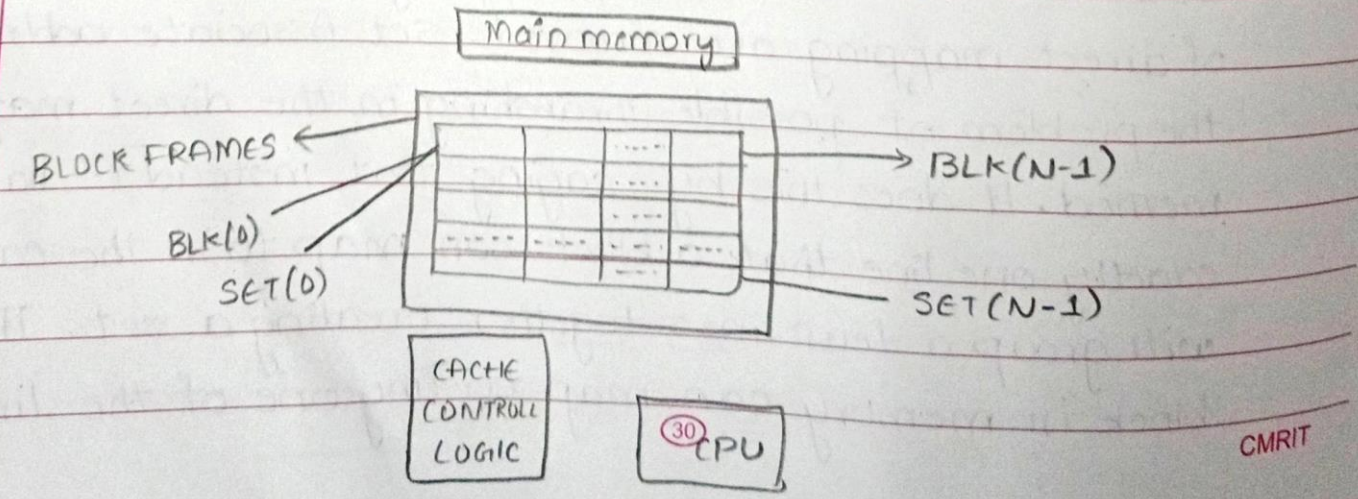


Relationships in the Set-Associative Mapping can be defined as :-

$m = v * k$

$i = j \bmod v$

where, i = cache set number, j = main memory block number
v = number of sets, m = no. of lines in the cache number
k = no. of lines in each set.



5. Describe in detail about Pipeline processing.

Ans The term pipelining refers to a technique of decomposing a sequential process into sub-operations, with each sub-operation being executed in a dedicated segment that operators concurrently with all other segments. The most important characteristic of a pipeline technique is that several computations can be in progress in distinct segments at the same time. The overlapping of computation is made possible by associating a register with each segment in the pipeline. The registers provide isolation between each segment so that each can operate on distinct data simultaneously.

Let us consider an example of combined multiplication and addition operation to get a better understanding of the pipeline organization.

The combined multiplication and addition operation isdone with a stream of numbers such as:

$A_i * B_i + C_i$ for $i = 1, 2, 3 \ldots\ldots, 7$

The operation to be perfermed on the number is decomposed into sub-operations with each sub-operation to be implemented in a segment within a pipeline.

The sub-operations performed in each segment of the pipeline are defined as

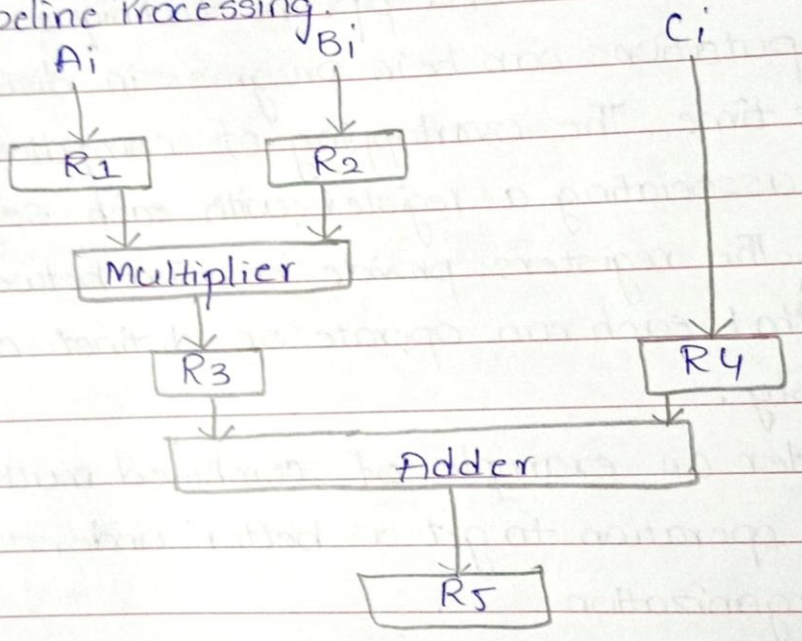
$R_1 \leftarrow A_{ij}$, $R_2 \leftarrow B_i$ Input A_i, and B_i

$R_3 \leftarrow R_1 * R_2, R_4 \leftarrow C_i$ Multiply and input C_i

$R_5 \leftarrow R_3 + R_4$ Add C_i to product

The following block diagram represents the combined as well as the sub-operations performed in each segment of the pipeline.

pipeline Processing:



Registers R1, R2, R3 and R4 hold the data and the combinational circuits operate in a particular segment. The output generated by the combinational circuit in a given segment is applied as an input register of the next segment.

In general, the pipeline organization is applicable for two areas of computer design which includes:

1. Arithmetic Pipeline
2. Instruction Pipeline

Arithmetic Pipeline :-

Arithmetic pipelines are mostly used in high-speed computers. They are used to implement floating-point numbers, and similar computations encountered in scientific problems.

The inputs to the floating-point adder pipeline are two normalized floating-point binary numbers defined as:

$X = A * 2^a = 0.9504 * 10^3$

$Y = B * 2^b = 0.8200 * 10^2$

Where A and B are two fractions that represent the mantissa and a and b are the exponents.

The combined operation of floating-point addition and subtraction is divided into four segments. Each segment the corresponding suboperation to be performed in the given pipeline. The suboperations that are shown in the four segments are:

1. Compare the exponents by subtraction.
2. Align the mantissas.
3. Add or subtract the mantissas
4. Normalize the result.

1. Compare exponents by subtraction:-

The exponents are compared by subtracting them to determine their difference. The larger exponent is choosen as the exponent of the result.

The difference of the exponents, i.e, 3-2=1

determines how many times the mantissa associated with the smaller exponent must be shifted to the right.

2. Align the mantissas:-

The mantissa associated with the smaller exponent is shifted according to the difference of exponents determined in segment one.

$X = 0.9504 * 10^3$

$Y = 0.08200 * 10^3$

3. Add Mantissas:-

The two mantissas are added in segment three

$Z = X + Y = 1.0324 * 10^3$

4. Normalize the result:-

After the normalization, the result is written as

$Z = 0.1324 * 10^4$

2. Instruction Pipeline:-

In this a stream of instructions can be executed by overlapping fetch, decode and execute phases of an instruction cycle.

Steps:-

1) Fetch the instruction from memory (FI)
2) Decode the instruction (DA)
3) Calculate the effective address
4) Fetch the operands from the memory (FO)
5) Execute the instruction (EX)

6) Store the result in the proper place.

Flowchart:-