CMR GROUP OF INSTITUTIONS
EXPLORE TO INVENT

CMR INSTITUTE OF TECHNOLOGY
UGC AUTONOMOUS
Approved by AICTE, New Delhi & Permanently Affiliated to JNTUH, Hyderabad & Accredited by NBA

B.Tech. III – Semester- II- Mid Term Examinations Key Paper – JANUARY – 2023
Digital Logic Design & Computer Organization(22ES32)
(Department of CSE/CSE(AI&ML)/CSE(DS)/AI&ML/CSE(CS))

Note: 1. This question paper contains two parts A and B.
2. Part A is compulsory which carries 5 marks. Answer all questions in Part A.
3. Part B consists of 5 questions. Answer all 5 questions. Each question carries 5 marks
4. Illustrate your answers with NEAT sketches wherever necessary.

PART-A **5 x 1M=5M**

1. a.List various commonly used flags.

Conditional Flags: Carry Flag,Auxiliary Flag,Parity Flag,Zero Flag,Sign Flag,Overflow
Control Flags: Trap Flag,Interrupt Flag,Direction Flag

b.What are the advantages of Booth's algorithm?

Booth's algorithm is efficient and 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.

c.Define: I/O Interface.

Input-output interface provides a method for transferring information between internal storage and external I/O devices

d.What are the different types of ROM?

Programmable ROM: It is a type of ROM where the data is written after the memory chip has been created. It is non-volatile.

Erasable Programmable ROM: It is a type of ROM where the data on this non-volatile memory chip can be erased by exposing it to high-intensity UV light.

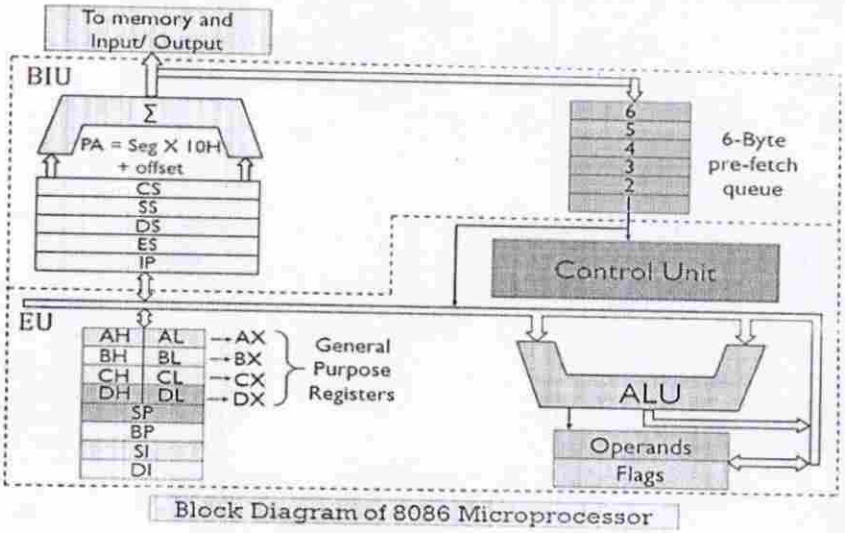
Electrically Erasable Programmable ROM: It is a type of ROM where the data on this non-volatile memory chip can be electrically erased using field electron emission.

e.Define: Hit and Miss.

If the processor finds that the memory location is in the cache, a **Cache Hit** has occurred and data is read from the cache.
If the processor does not find the memory location in the cache, a **cache miss** has occurred.

PART-B 5x5M=25M

2.With a neat sketch, explain the internal architecture of Intel 8086 microprocessor.



Block Diagram of 8086 Microprocessor

Bus Interface Unit (BIU)

The segment registers, instruction pointer and 6-byte instruction queue are associated with the bus interface unit (BIU).

The BIU:

•Handles transfer of data and addresses,

•Fetches instruction codes, stores fetched instruction codes in first-in-first-out register set called a **queue**,

•Reads data from memory and I/O devices,

•Writes data to memory and I/O devices,

•It relocates addresses of operands since it gets un-relocated operand addresses from EU. The EU tells the BIU from where to fetch instructions or where to read data.

It has the following functional parts:

•**Instruction Queue:** When EU executes instructions, the BIU gets 6-bytes of the next instruction and stores them in the instruction queue and this process is known as instruction pre fetch. This process increases the speed of the processor.

•**Segment Registers:** A segment register contains the addresses of instructions and data in memory which are used by the processor to access memory locations. It points to the starting address of a memory segment currently being used. There are 4 segment registers in 8086 as given below:

Code Segment Register (CS): Code segment of the memory holds instruction codes of a program.

Data Segment Register (DS): The data, variables and constants given in the program are held in the data segment of the memory.

Stack Segment Register (SS): Stack segment holds addresses and data of subroutines. It also holds the contents of registers and memory locations given in PUSH instruction.

Extra Segment Register (ES): Extra segment holds the destination addresses of some data of certain string instructions.

Instruction Pointer (IP): The instruction pointer in the 8086 microprocessor acts as a program counter. It indicates to the address of the next instruction to be executed.

Execution Unit (EU)

•The **EU** receives opcode of an instruction from the queue, decodes it and then executes it. While Execution, unit decodes or executes an instruction, then the BIU fetches instruction codes from the memory and stores them in the queue.

•The BIU and EU operate in parallel independently. This makes processing faster.

•General purpose registers, stack pointer, base pointer and index registers, ALU, flag registers (FLAGS), instruction decoder and timing and control unit constitute execution unit (EU).

General Purpose Registers: There are four 16-bit general purpose registers: AX (Accumulator Register), BX (Base Register), CX (Counter) and DX. Each of these 16-bit registers are further subdivided into 8-bit registers as shown below: The use of general-purpose registers is to store temporary data. While the instructions are executed in the control unit, they may work on some numeric value or some operands. These need to be stored somewhere so that the processor can operate on them easily. So, these registers are used in these cases. There are 4 general-purpose registers of 16-bit length each. Each of them is further divided into two subparts of 8-bit length each: one high, which stores the higher-order bits and another low which stores the lower order bits.

•AX = [AH:AL]

•BX = [BH:BL]

•CX = [CH:CL]

•DX = [DH:DL]

Index Register: The following four registers are in the group of pointer and index registers:

•Stack Pointer (SP)

•Base Pointer (BP)

•Source Index (SI)

•Destination Index (DI)

3.Describe about ALP with an example program with output.

Assembly language is a low-level programming language for a computer or other programmable device specific to a particular computer architecture in contrast to most high-level programming languages, which are generally portable across multiple systems. Assembly language is converted into executable machine code by a utility program referred to as an assembler like NASM, MASM, etc.

Advantages:

An understanding of assembly language provides knowledge of:

- Interface of programs with OS, processor and BIOS;
- Representation of data in memory and other external devices;

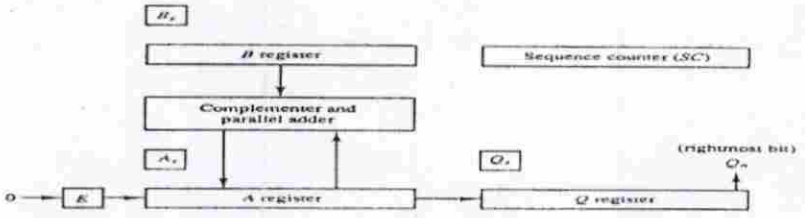
Example: To Write assembly language programs to evaluate the expressions:

- i) a = b + c – d * e
- PROGRAM:

```
assume cs:code, ds:data
data segment
b db 02
c db 04
d db 01
e db 03
a db 01 dup()
data ends
code segment
 start:mov ax,data
 mov ds,ax
 mov al,b
 mov bl,c
 add al,bl
 mov cl,d
 sub al,cl
 mov dl,e
 mul d
 mov a,al
 int 03
code ends
end start
```

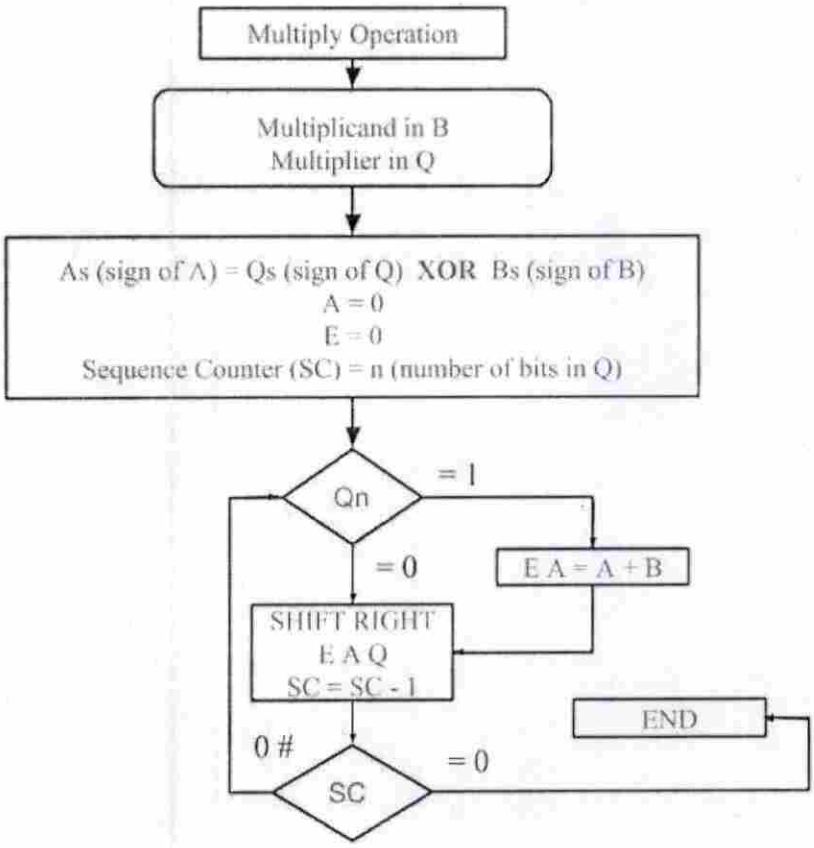
RESULT:AX=000F

4.Analyze multiplication algorithm with an example
Hardware Implementation for Signed-Magnitude Data:



Multiplication of two fixed-point binary numbers in signed magnitude representation is

```
23      10111 Multiplicand
19  x   10011 Multiplier
```



Multiplicand B = 10111	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
Q_n = 1; add B		10111		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
Q_n = 1; add B		10111		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
Q_n = 0; shift right EAQ	0	01000	10110	010
Q_n = 0; shift right EAQ	0	00100	01011	001
Q_n = 1; add B		10111		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final product in AQ = 0110110101				

5.Outline division algorithm with an example.

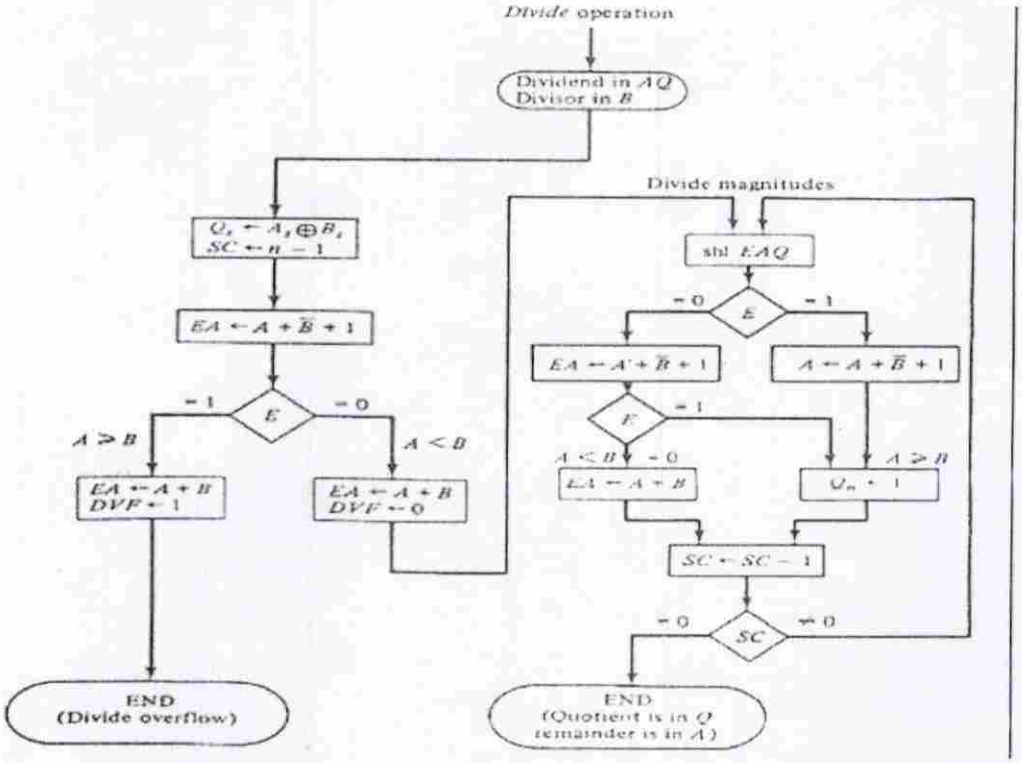
Divisor B:10001

Dividend:0111000000

Divisor $B = 10001$, $\overline{B} + 1 = 01111$

	E	A	Q	SC
Dividend:		01110	00000	5
shl EAQ	0	11100	00000	
add $\overline{B} + 1$		01111		
$E = 1$	1	01011		
Set $Q_n = 1$	1	01011	00001	4
shl EAQ	0	10110	00010	
Add $\overline{B} + 1$		01111		
$E = 1$	1	00101		
Set $Q_n = 1$	1	00101	00011	3
shl EAQ	0	01010	00110	
Add $\overline{B} + 1$		01111		
$E = 0$; leave $Q_n = 0$	0	11001	00110	
Add B		10001		
Restore remainder	1	01010		2
shl EAQ	0	10100	01100	
Add $\overline{B} + 1$		01111		
$E = 1$	1	00011		
Set $Q_n = 1$	1	00011	01101	1
shl EAQ	0	00110	11010	
Add $\overline{B} + 1$		01111		
$E = 0$; leave $Q_n = 0$	0	10101	11010	
Add B		10001		
Restore remainder	1	00110	11010	0
Neglect E				
Remainder in A:		00110		
Quotient in Q:			11010	

Hardware Algorithm:



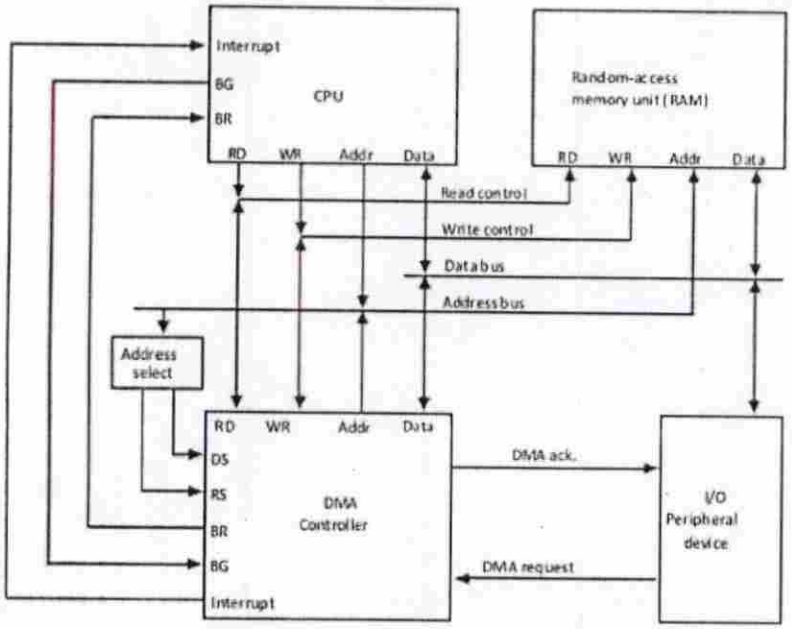
6.Explain briefly about DMA transfer with the help of a block diagram.

Direct Memory Access (DMA) The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called direct memory access (DMA).

The position of the DMA controller among the other components in a computer system is illustrated in Fig. The CPU communicates with the DMA through the address and data buses as with any interface unit. The DMA has its own address, which activates the DS and RS lines. The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory. When the peripheral device sends a DMA request, the DMA controller activates the BR line

Informing the CPU to relinquish the buses. The CPU responds with its BG line, informing the DMA that its buses are disabled. The DMA then puts the current value of its address register into the address bus, initiates the RD or WR signal, and sends a DMA acknowledge to the peripheral device. Note that the RD and WR lines in the DMA controller are bidirectional. The direction of transfer depends on the status of the BG line.

When BG line. When BG = 0, the RD and WR are input lines allowing the CPU to communicate with the internal DMA registers. When BG = 1, the RD and WR and output lines from the DMA controller to the random-access memory to specify the read or write operation for the data. When the peripheral device receives a DMA acknowledge, it puts a word in the data us (for write) or receives a word from the data bus (for read). Thus the DMA controls the read or write operations and supplies the address for the memory. The peripheral unit can then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled.



7.Illustrate Asynchronous data transfer and communication Interface.

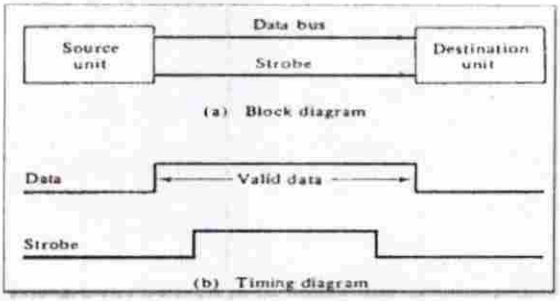
The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator. Clock pulses are applied to all registers within a unit and all data transfers among internal registers occur simultaneously during the occurrence of a clock pulse. Two units, such as a CPU and an I/O interface, are designed independently of each other.

There are two types of asynchronous data transmission methods

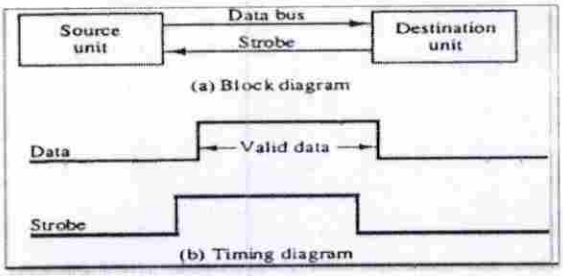
•Strobe control 2. Handshaking

1) **Strobe Control**

i) source-initiated transfer.

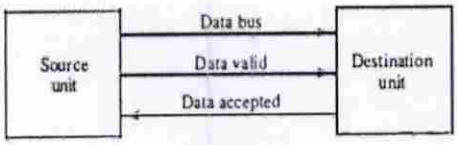


ii) Destination-initiated strobe for data transfer

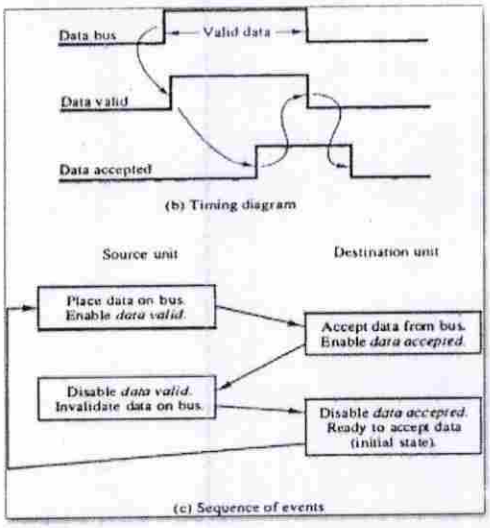


HANDSHAKING

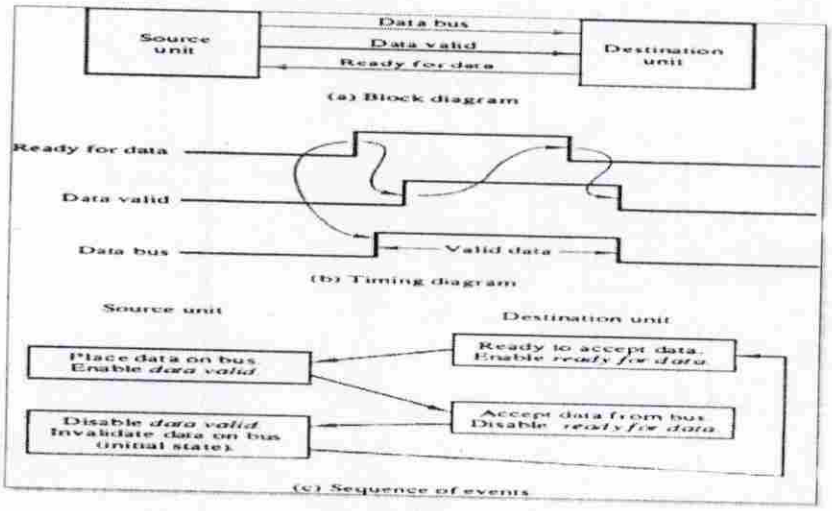
i)Data Transfer Procedure When Initiated By the Source



(a) Block diagram



ii)Destination -Initiated Transfer Using Handshaking



8.Demonstrate about Associative memory.

Associative Memory

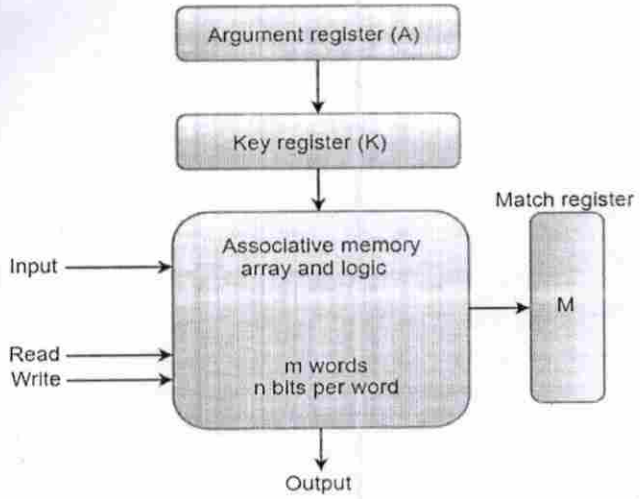
An associative memory can be considered as a memory unit whose stored data can be identified for access by the content of the data itself rather than by an address or memory location.

Associative memory is often referred to as **Content Addressable Memory (CAM).**

When a write operation is performed on associative memory, no address or memory location is given to the word. The memory itself is capable of finding an empty unused location to store the word.

On the other hand, when the word is to be read from an associative memory, the content of the word, or part of the word, is specified. The words which match the specified content are located by the memory and are marked for reading.

The following diagram shows the block representation of an Associative memory.



From the block diagram, we can say that an associative memory consists of a memory array and logic for 'm' words with 'n' bits per word.

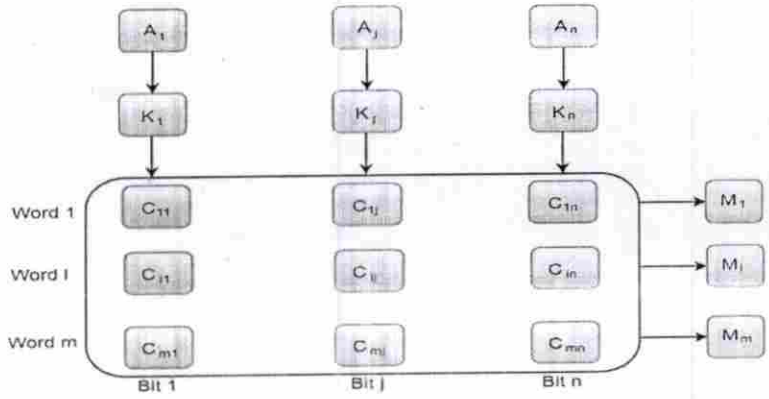
The functional registers like the argument register **A** and key register **K** each have **n** bits, one for each bit of a word. The match register **M** consists of **m** bits, one for each memory word.

The words which are kept in the memory are compared in parallel with the content of the argument register.

The key register (K) provides a mask for choosing a particular field or key in the argument word. If the key register contains a binary value of all 1's, then the entire argument is compared with each memory word. Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared. Thus, the key provides a mask for identifying a piece of information which specifies how the reference to memory is made.

The following diagram can represent the relation between the memory array and the external registers in an associative memory.

Associative memory of m word, n cells per word:



The cells present inside the memory array are marked by the letter C with two subscripts. The first subscript gives the word number and the second specifies the bit position in the word. For instance, the cell C_{ij} is the cell for bit **j** in word **i**.

A bit A_j in the argument register is compared with all the bits in column **j** of the array provided that $K_j = 1$. This process is done for all columns **j** = 1, 2, 3....., n.

If a match occurs between all the unmasked bits of the argument and the bits in word **i**, the corresponding bit M_i in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, M_i is cleared to 0.

9.Explain in detail about Arithmetic Pipelining with an example.

Arithmetic Pipeline

Arithmetic Pipelines are mostly used in high-speed computers. They are used to implement floating-point operations, multiplication of fixed-point numbers, and similar computations encountered in scientific problems.

To understand the concepts of arithmetic pipeline in a more convenient way, let us consider an example of a pipeline unit for floating-point addition and subtraction.

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.

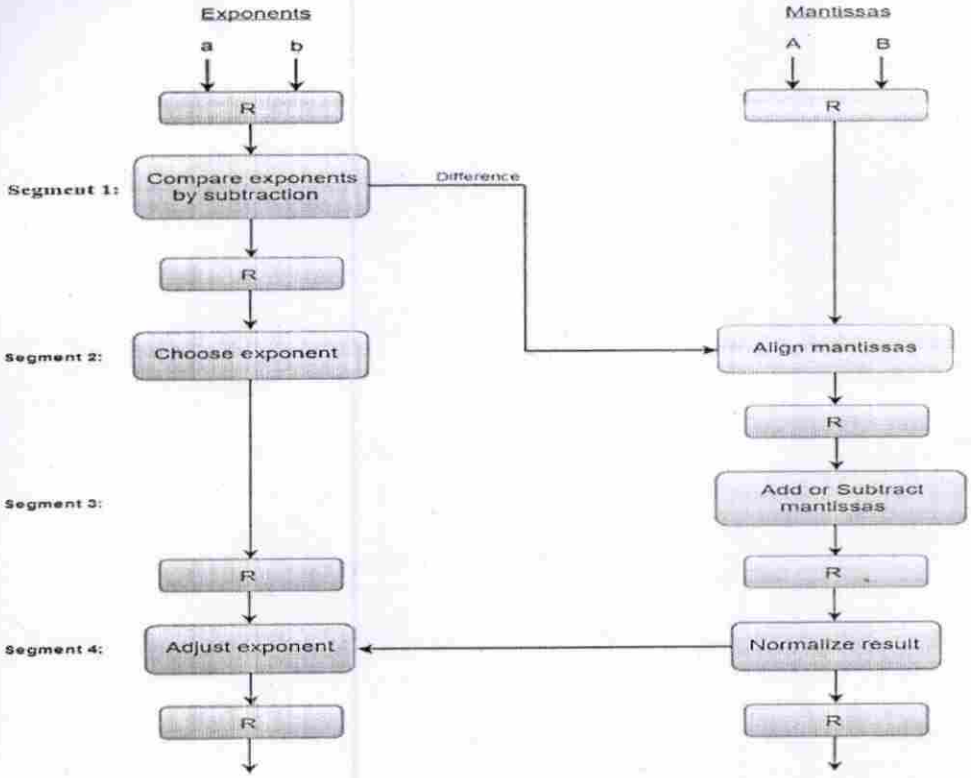
The combined operation of floating-point addition and subtraction is divided into four segments. Each segment contains the corresponding suboperation to be performed in the given pipeline. The suboperations that are shown in the four segments are:

- Compare the exponents by subtraction.
- Align the mantissas.
- Add or subtract the mantissas.
- Normalize the result.

We will discuss each suboperation in a more detailed manner later in this section.

The following block diagram represents the suboperations performed in each segment of the pipeline.

Pipeline organization for floating point addition and subtraction:



1. Compare exponents by subtraction:

The exponents are compared by subtracting them to determine their difference. The larger exponent is chosen 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 normalization, the result is written as:

$$Z = 0.1324 * 10^4$$

10.Using a block diagram Illustrate memory hierarchy in a computer system.

Memory Hierarchy

In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behavior known as locality of references. The figure below clearly demonstrates the different levels of the memory hierarchy.

Why Memory Hierarchy is Required in the System?

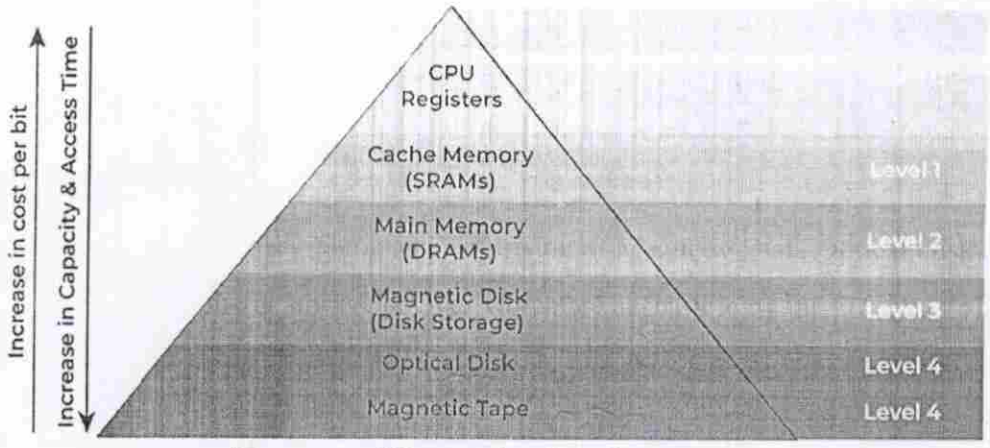
Memory Hierarchy is one of the most required things in Computer Memoryas it helps in optimizing the memory available in the computer. There are multiple levels present in the memory, each one having a different size, different cost, etc.

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**. This is directly accessible by the processor.



Memory Hierarchy Design

Memory Hierarchy Design

1. Registers-Registersare small, high-speed memory units located in the CPU. They are used to store the most frequently used data and instructions. Registers have the fastest access time and the smallest storage capacity, typically ranging from 16 to 64 bits.

2. Cache Memory-Cache memoryis a small, fast memory unit located close to the CPU. It stores frequently used data and instructions that have been recently accessed from the main memory. Cache memory is designed to minimize the time it takes to access data by providing the CPU with quick access to frequently used data.

3. Main Memory-Main memory, also known as RAM (Random Access Memory), is the primary memory of a computer system. It has a larger storage capacity than cache memory, but it is slower. Main memory is used to store data and instructions that are currently in use by the CPU.

4. Secondary Storage-Secondary storage, such as **hard disk drives (HDD) and solid-state drives (SSD)**, is a non-volatile memory unit that has a larger storage capacity than main memory. It is used to store data and instructions that are not currently in use by the CPU. Secondary storage has the slowest access time and is typically the least expensive type of memory in the memory hierarchy.

5. Magnetic Disk-**Magnetic Disks**are simply circular plates that are fabricated with either a metal or a plastic or a magnetized material. The Magnetic disks work at a high speed inside the computer and these are frequently used.

6. Magnetic Tape-**Magnetic Tape**is simply a magnetic recording device that is covered with a plastic film. It is generally used for the backup of data. In the case of a magnetic tape, the access time for a computer is a little slower and therefore, it requires some amount of time for accessing the strip.

Characteristics of Memory Hierarchy

Capacity:It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.

Access Time:It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.

11.Describe in detail about Pipeline processing.

Pipelining

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 operates 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.

The structure of a pipeline organization can be represented simply by including an input register for each segment followed by a combinational circuit.

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 is done with a stream of numbers such as:

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

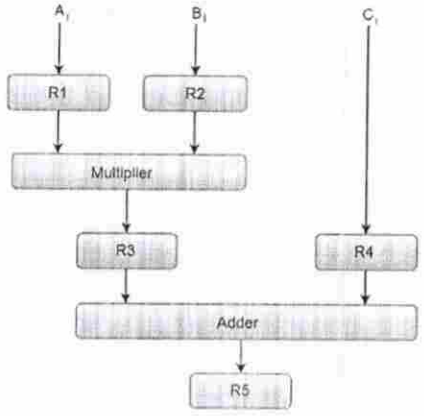
The operation to be performed on the numbers is decomposed into sub-operations with each sub-operation to be implemented in a segment within a pipeline.

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

$R1 \leftarrow A_i$, $R2 \leftarrow B_i$	Input A_i, and B_i
$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$	Multiply, and input C_i
$R5 \leftarrow R3 + R4$	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. For instance, from the block diagram, we can see that the register R3 is used as one of the input registers for the combinational adder circuit.

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

- Arithmetic Pipeline
- Instruction Pipeline

DLD & CO

Divya.P
8247692362
CSE(DS) Department