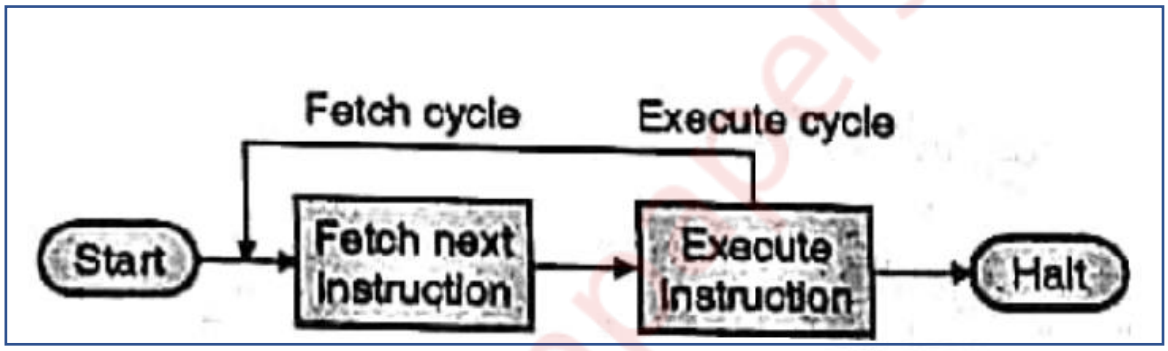
COMPUTER ORGANISATION AND ARCHITECTURE
(NOV 2018) Q.P. 57916

Q.1) Write the following (20 M)

a) Explain Instruction and instruction cycle. (5 M)

Ans:

- The instruction cycle is a representation of the states that the computer or the microprocessor performs when executing an instruction.
- The instruction cycle comprises of two main steps to be followed to execute the instruction namely the fetch operation in the fetch cycle and the execution operation during the execute cycle.

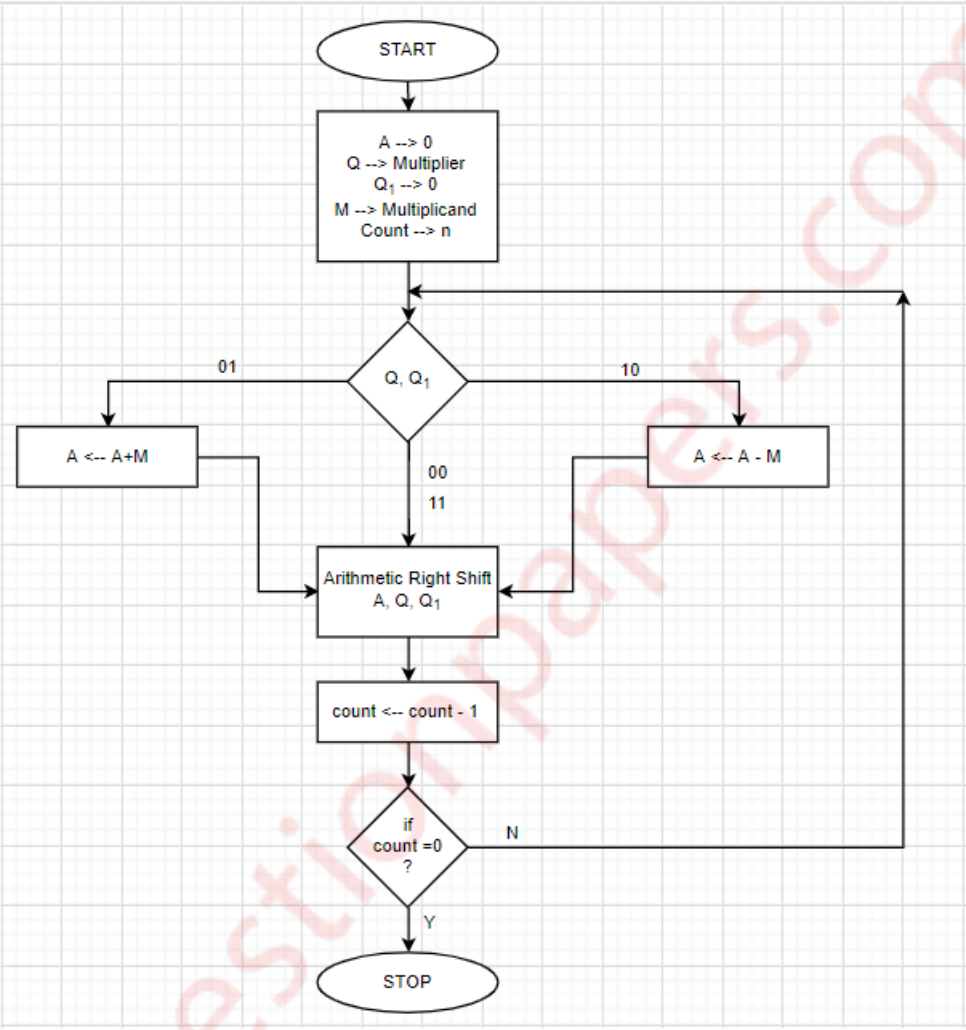


- It comprises of the fetch cycle and executes cycle in a loop to execute huge number of instructions, until it reaches the halt instruction.
- The fetch cycle comprises of the following operations:
    - Program centre holds address of next instruction to fetch; hence the CPU fetches instruction from memory pointed to by PC. This is done by providing the value of PC to the MAR and giving the Read control signal to the memory. On this memory provides the value in the given address.
    - The PC values has to be incremented to point to the next instruction.
    - The instruction is loaded into Instruction Register from the MBR.
    - Finally the processor interprets or decodes the instruction. The processor performs required operations in execute cycle.
- In the execute cycle the operation asked to be performed by the instruction is done. It may comprise of one or more of the following:
    - Transfer of data between processor and memory or between processor or IO module.
    - Processing of data like some arithmetic or logical operation of data.
    - Change the sequence of operation i.e. branching instructions.

b) Explain Booths algorithm with an example. (5 M)

Ans:

Booth's principle states that "The value of series of 1's of binary can be given as the weight of the bit preceding the series minus the weight of the last bit in the series.**"**



Example: Multiply 7 * -3 using Booth's algorithm

A	Q	Q_{-1}	m	count
0000	1101	0	0111	4
+1001				
1001	1101	0		
1100	1110	1		3
+0111				
0011	1110	1		
0001	1111	0		2

1001				
1010	1111	0		
1101	0111	1		1
1110	1011	1		0

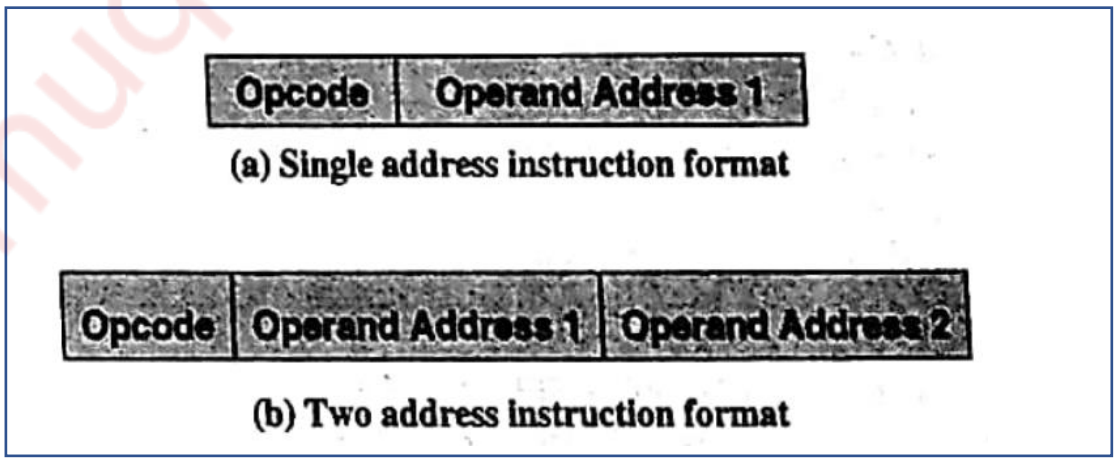
Hence 11101011 = $-(00010101)_2$

$= -(21)_{10}$

c) Give different instruction formats. (5 M)

Ans:

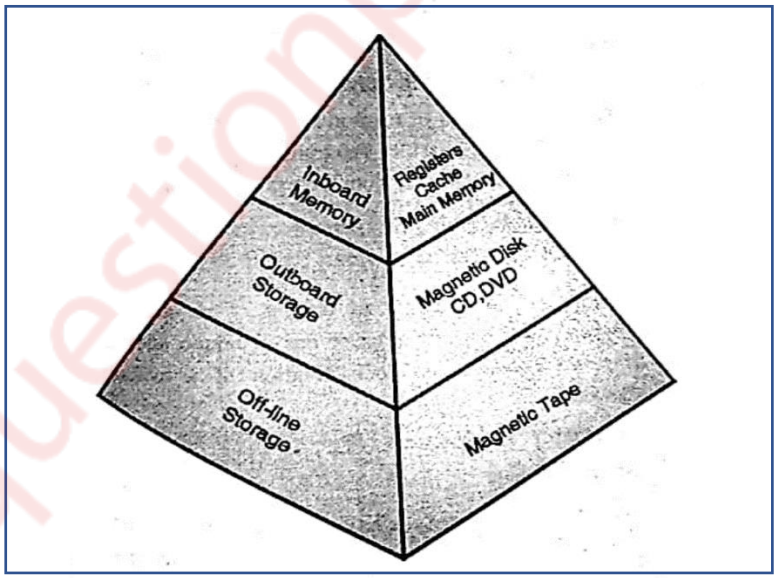
- Input devices are required to give the instructions and data to the system. The output devices are used to give the output devices.
- The instructions and data given by the input device are to be stored, and for storage we require memory.
- Elements of Single, two and three address instructions are as follows:
  - Operation code is that part of the instruction which gives the code for the operation to be performed.
  - Source operand reference or address 1, gives the reference of the data on which the operation is to be performed. This address could be a register, memory or an input device.
  - Source operand reference or address 2, gives the reference of the second data on which the operation is to be performed. This address could again be a register, memory or an input device.
  - Result operand reference gives the reference where the result after performing operation is to be stored.
  - An instruction may have only one address with the other two fixed, or may have two addresses with one of the source operand address as the result operand address. Hence the instruction can have one, two or three addresses.



d) Describe the memory hierarchy in the computer system. (5 M)

Ans:

- Memory hierarchy explains that the nearer the memory to the processor, faster is its access. But costlier the memory becomes as it goes closer to the processor. The following sequence is in faster to slower or costlier to cheaper memory.
    - Registers i.e. inside the CPU.
    - Internal memory that includes one or more levels of cache and the main memory. Internal memory is always RAM, SRAM, DRAM for main memory. This is called as the primary memory.
    - External memory or removable memory includes the hard disk, CDs, DVDs etc. this is the secondary memory.
- The registers as discussed are the closest to the processor and hence are the fastest while off-line storage like magnetic tape are the farthest and also the slowest. The list of memories from closest to the processor to the farthest is given as below:
    - Registers
    - L1 cache
    - L2 cache
    - Main memory
    - Magnetic disk
    - Optical
    - Tape.



- To have a large faster memory is very costly and hence the different memory at different memory at different levels gives the memory hierarchy.

---

e) Explain Superscalar Architecture. (5 M)

Ans:

- Superscalar processor are those processors that have multiple execution units.

- Hence these processors can execute the independent instructions simultaneously and hence with the help of this parallelism it increases the speed of the processor.
- It has been that the number of independent consecutive instructions 2 to 5. Hence the instruction issue degree in a superscalar processor is restricted from 2 to 5.

Pipelining in Superscalar Processor:

- The pipelining is the most important representation of demonstrating the speed increase by the superscalar feature of processor.
- Hence to implement multiple operations simultaneously, we need to have multiple execution units to execute each instruction independently.
- The ID and rename unit, decodes the instruction and then by the use of register renaming avoids instruction dependency. The instruction window takes the decoded instructions and based on some pair ability rules, issues them to the respective execution units.
- The instructions once executed move to the Retire and write back unit, wherein the instructions retire and the result is written back to the corresponding destination.

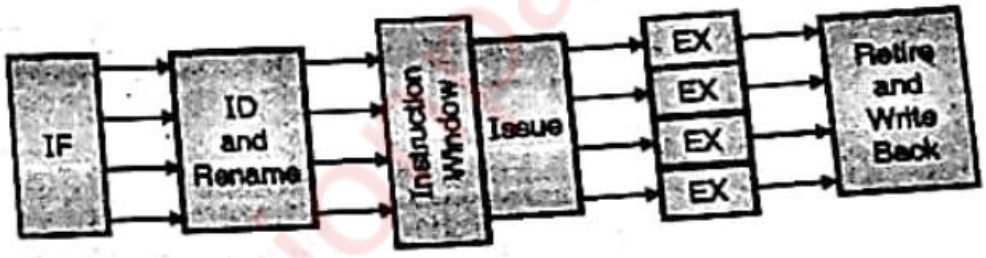
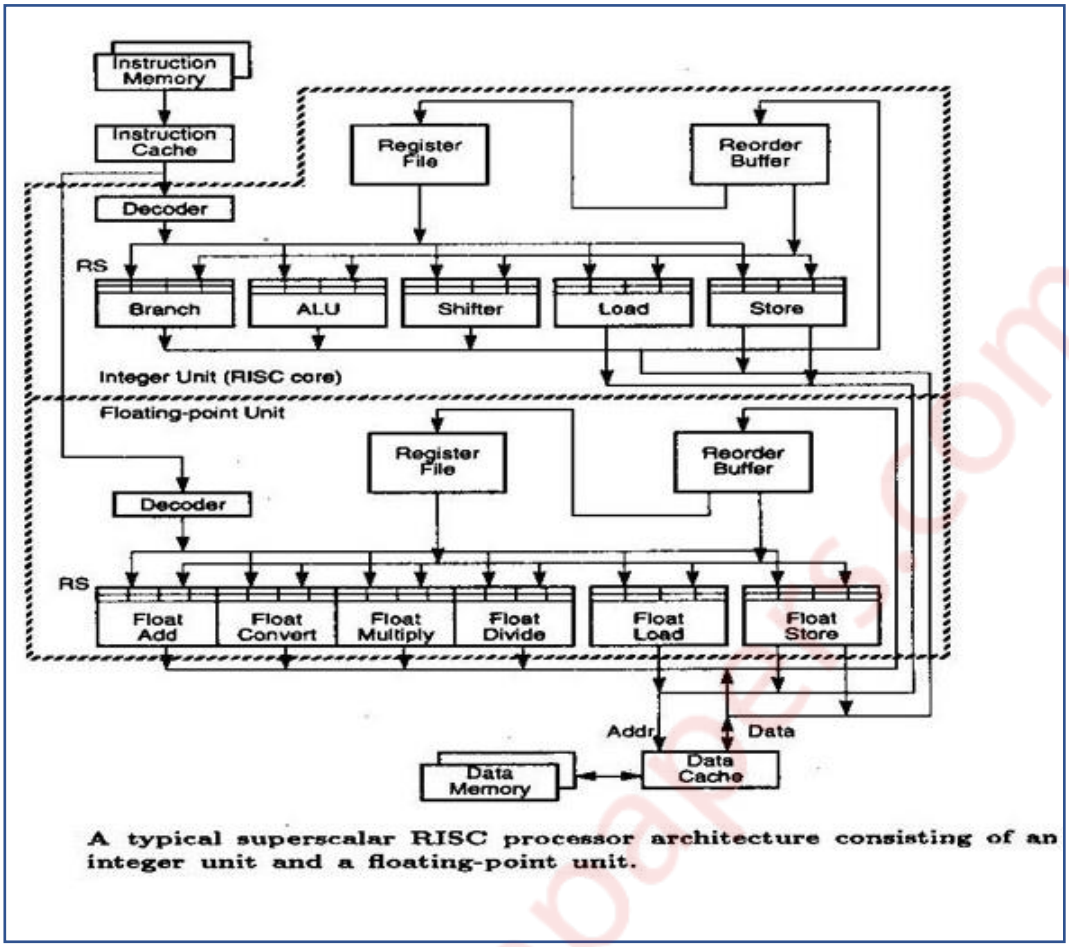


Fig. 6.3.2 : Block diagram of a typical superscalar processor

- A RISC or CISC processors execute one instruction per cycle. Their performance can be improved with superscalar architecture:
  - Multiple instruction pipelines are used.
  - Multiple instructions are issued for execution per cycle.
  - Multiple results are generated per cycles.

- Superscalar processors can exploit more instruction level parallelism in user program.



A typical superscalar RISC processor architecture consisting of an integer unit and a floating-point unit.

Q.2)

a) Explain Branch Prediction Logic and Delayed Branch. (10 M)

Ans:

- Performance gain through pipelining can be reduced by the presence of program transfer instructions (such as JMP, CALL, RET and conditional jumps).
- They change the sequence causing all the instructions that entered the pipeline after program transfer instruction invalid.
- Suppose instruction I3 is a conditional jump to I50 at some other address (target address), then the instructions that entered after I3 is invalid and new sequence beginning with I50 need to be loaded in.
- This causes bubbles in pipeline, where no work is done as the pipeline stages are reloaded.
- To avoid this problem, the Pentium uses a scheme called Dynamic Branch Prediction.
- In this scheme, a prediction is made concerning the branch instruction currently in pipeline.
- Prediction will be either taken or not taken.

- If the prediction turns out to be true, the pipeline will not be flushed and no clock cycles will be lost. If the prediction turns out to be false, the pipeline is flushed and started over with the correct instruction.
- It results in a 3 cycle penalty if the branch is executed in the u-pipeline and 4 cycle penalty in v-pipeline.
- It is implemented using a 4-way set associative cache with 256 entries. This is referred to as the Branch Target Buffer (BTB).
- The directory entry for each line contains the following information:
- Valid Bit : Indicates whether or not the entry is in use.
- History Bits: track how often the branch has been taken.
- Source memory address that the branch instruction was fetched from (address of I3).
- If its directory entry is valid, the target address of the branch is stored in corresponding data entry in BTB.

Delayed Branch:

- Compiler detects branch instruction and rearranges the machine language code sequence by inserting useful instructions and rearranges the code sequence to reduce the delays incurred by Branch Instruction.

---

b) List and explain various data dependencies, data and branch hazards that occur in the computer system. (10 M)

Ans:

- Instruction hazards occur when instructions read or write registers that are used by other instructions. The type of conflicts are divided into three categories:
    - Structural Hazards (resource conflicts)
    - Data Hazards (Data dependency conflicts)
    - Branch difficulties (Control Hazards)
- **Structural hazards:** these hazards are caused by access to memory by two instructions at the same time. These conflicts can be slightly resolved by using separate instruction and data memories.
- It occurs when the processor's hardware is not capable of executing all the instructions in the pipeline simultaneously.
- Structural Hazards within a single pipeline are rare on modern processors because the instructions Set Architecture is designed to support pipelining.
- **Data Hazards(Data Dependency):** This hazard arises when an instruction depends on the result of a previous instruction, but this result is not available.
- These are divided into four categories.
    - RAW - Hazard
    - RAR - Hazard
    - WAW - Hazard
    - WAR – Hazard

- **RAR Hazard:** RAR Hazard occurs when two instructions both from the same register. This hazard does not came from problem for the processor because reading a register does not change the register's value. Therefore, two instructions that have RAR Hazard can execute on successive cycles.
- **RAW Hazard:** This hazard occurs when an instruction reads a register that was written by a previous instruction. These are called as data dependencies.
- **WAR and WAW** are also called as name dependencies.
- These hazards occurs when the output register of an instruction has been either read or written by a previous instruction.
- If the processor executes instructions in the order that they appear in the program and uses the same pipeline for the instructions, WAR and WAW hazards do not cause any problem in execution process.
- **Branch Hazards:** Branch instructions, particularly conditional branch instructions, create data dependencies between the branch instruction and the previous instruction, fetch stage of the pipeline.
- Since the branch instruction computes the address of the next instruction that the instruction fetch stage should fetch from, it consumes some time and also some time is required to flush the pipeline and fetch instructions from target location. This time wasted is called as branch penalty.

Q.3)

a) A program having 10 instructions (without Branch and Call Instructions) is executed on non-pipeline and pipeline processors. All instructions are of same length and having 4 pipeline stages and time required to each stage is 1nsecc.

I. Calculate time required to execute the program or Non-pipeline and Pipeline processor.

II. Calculate Speedup. (10 M)

Ans:

I. Given: n = 10 instructions, K = 4, t = 1nsec

Execution time pipelined = (4 + 100 - 1) * t

= (4 + 90) * 1

= 94 nsec.

Execution time unpipelined = (K * t) n

= (4 * 1) 1

= 4 nsec.

II. Speedup = 4/94 **= 0.043 times.**

b) What is micro program? Write microprogram for following operations

I. ADD R1, M, Register R1 and Memory location M are added and result store at Register R1.

II. MUL R1, R2 Register R1 and Register R2 are multiplied and result store at Register R1. (10 M)

Ans:

- Microprogramming is a process of writing microcode for a microprocessor. Microcode is low-level code that defines how a microprocessor should function when it executes machine-language instructions.
- Typically, one machine language instruction translates into several microcode instruction, on some computers, the microcode is stored in ROM and cannot be modified.
- **Micro Program to add R1, M.**

T-state	Operation	Microinstructions
T 1	PC → MAR	PCout, Marin, Read, Clear y, set Cin, Add, Zinn
T 2	M → MBR PC ← PC + 1	Zout, PCin, Wait for memory fetch cycle
T 3	MBR → IR	MBRout, IRin
T 4	R1 → x	R1out, Xin, CLRC
T 5	M → ALU	Mout, ADD, Zin
T 6	Z → R1	Zout, R1in
T 7	Check for intr	Assumption enabled intr pending, CLRX, SETC, Spout, SUB, Zin
T 8	SP ← SP – 1	Zout, Spin, MARin
T 9	PC → MDR	PCout, MDRin, WRITE
T 10	MDR → [SP]	Wait for Mem access
T11	PC ← IS Raddr	PCin IS Raddr out.

- **Micro Program to MUL R1, R2**

T-state	Operation	Microinstructions
T 1	PC → MAR	PCout, Marin, Read, Clear y, set Cin, Add, Zinn
T 2	M → MBR PC ← PC + 1	Zout, PCin, Wait for memory fetch cycle
T 3	MBR → IR	MBRout, IRin
T 4	R1 → x	R1out, Xin, CLRC
T 5	R2 → ALU	R2out, MUL, Zin

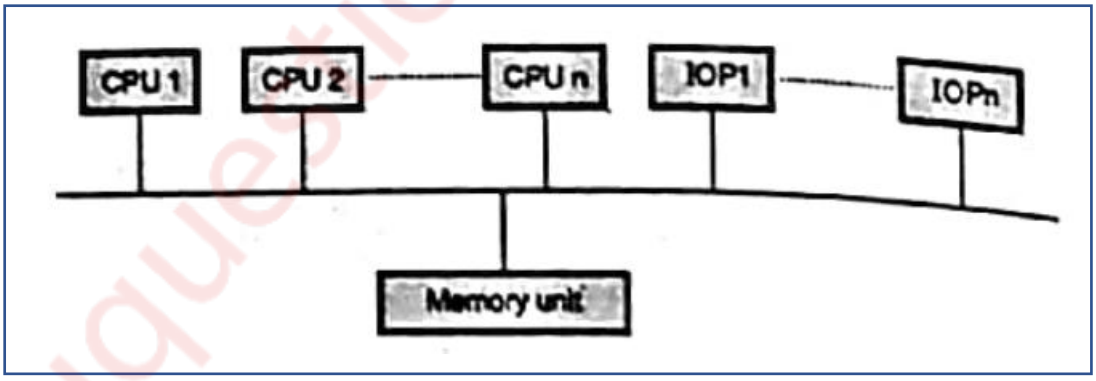
T 6	Z → R1	Zout, R1in
T 7	Check for intr	Assumption enabled intr pending, CLRX, SETC, Spout, SUB, Zin
T 8	SP ← SP – 1	Zout, Spin, MARin
T 9	PC → MDR	PCout, MDRin, WRITE
T 10	MDR → [SP]	Wait for Mem access
T11	PC ← IS Raddr	PCin IS Raddr out.

Q.4)

a) Explain Bus contention and Different method to resolve it. (10 M)

Ans:

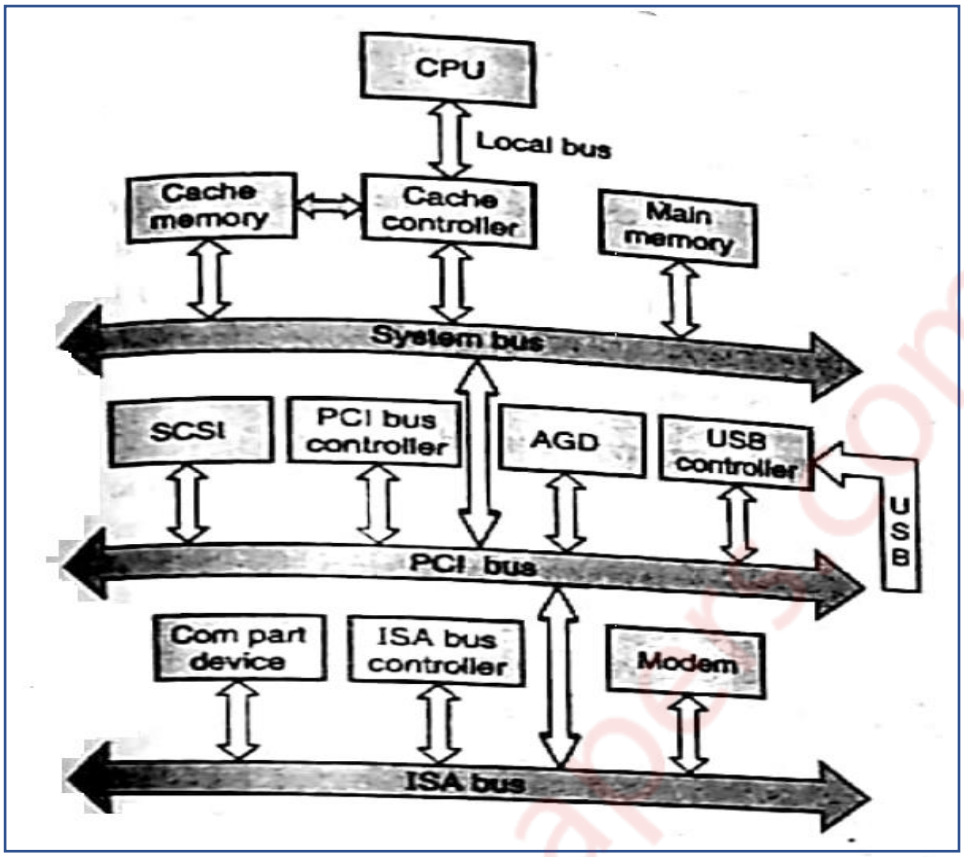
- In a bus system, processors, memory modules and peripheral devices are attached to the bus. The bus can handle only one transaction at a time between a master and slave. In case of multiple requests, the bus arbitration logic must be able to allocate or deallocate and it should service request at a time.
- Thus, such a bus is a time sharing or contention bus among multiple functional modules. As only one transfer can take place at any time on the bus, the overall performance of the system is limited by the bandwidth of the bus.
- When number of processors contending to acquire a bus exceeds the limit then a single bus architecture may become a major bottleneck. This may cause a serious delay in servicing a transaction.



- Aggregate data transfer demand should never exceed the capacity of the bus. This problem can be countered to some extent by increasing the data rate of the bus and by using a wider bus.
- Method of avoiding contention is multiple bus hierarchy.

Multiple-Bus Architecture:

- If a greater number of devices are connected to the bus, performance will suffer due to following reasons:



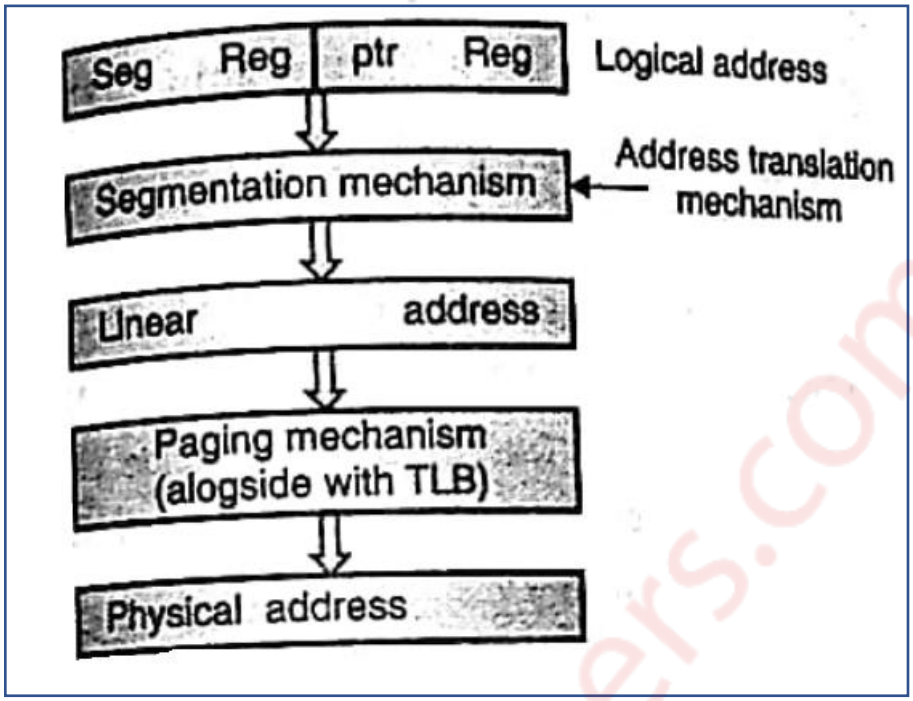
- In general, the more devices attached to the bus, the greater will be propagation delay.
- The bus may become a bottleneck as the aggregate data transfer demand approaches the capacity of the bus.
- This problem can be countered to some extent by increasing the data rate the bus can carry and by using wider buses.
- Most computer systems enjoy the use of multiple buses. These buses are arranged in a hierarchy.

---

b) Describe memory segmentation in detail. Explain how address translation is performed in virtual memory. (10 M)

Ans:

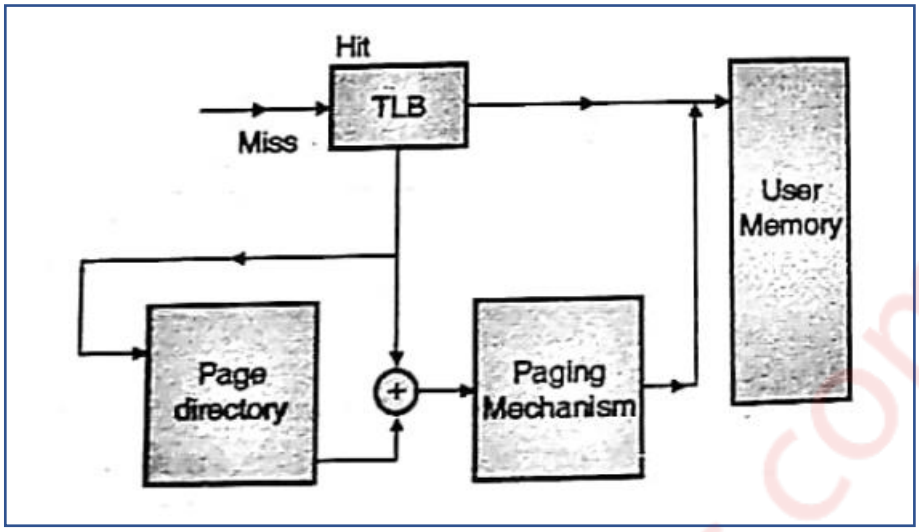
- Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.
- When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.



- Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.
- A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a segment map table for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory.
- For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.

Translation in Virtual Memory.

- This is a on chip buffer within the CPU, used to speed up the paging process. Since a page from Virtual Memory can get stored into any frame of main memory, the OS maintains a page Table which indicates which page of virtual memory is stored in each page frame of main memory.
- Hence for accessing the page CPU has to perform 2 Memory Operations:-
- First access the page table to get information about where the page is stored in main memory, than access the main memory for the page. To solve this problems CPU copies the pages table information of the most recently used pages in the on-chip TLB. Therefore, subsequent access to the pages will be faster and information will be provided by the TBL and CPU need not Access the Table.



Q.5)

a) State the various types of data transfer techniques. Explain DMA in detail. (10 M)

Ans:

Programmed I/O

- In the programmed I/O method of interfacing. CPU has direct control over I/O.
- The processor checks the status of the devices and issues read or write commands and then transfer data. During the data transfer. CPU waits for I/O module to complete operation and hence this system wastes the CPU time.
- The sequence of operations to be carried out in programmed I/O operation are:
  - CPU requests for I/O operation.
  - I/O module performs the said operation.
  - I/O module update the status bits.
  - CPU checks these status bits periodically. Neither the I/O module can inform CPU directly nor can I/O module interrupt CPU.
  - CPU may wait for the operation to complete or may continue the operation later.

Interrupt driven I/O

- Interrupt driven I/O overcomes the disadvantage of programmed I/O i.e. the CPU waiting for I/O device.
- This disadvantage is overcome by CPU not repeatedly checking for the device being ready or not instead the I/O module interrupts when ready.
- The sequence of operations for interrupt Driven I/O is as below:
  - CPU issues the read command to I/O device.
  - I/O module gets data from peripheral while CPU does other work.
  - Once the I/O module completes the data transfer from I/O device, it interrupts CPU.

- On getting the interrupt, CPU requests data from the I/O module.
- I/O module transfers the data to CPU.

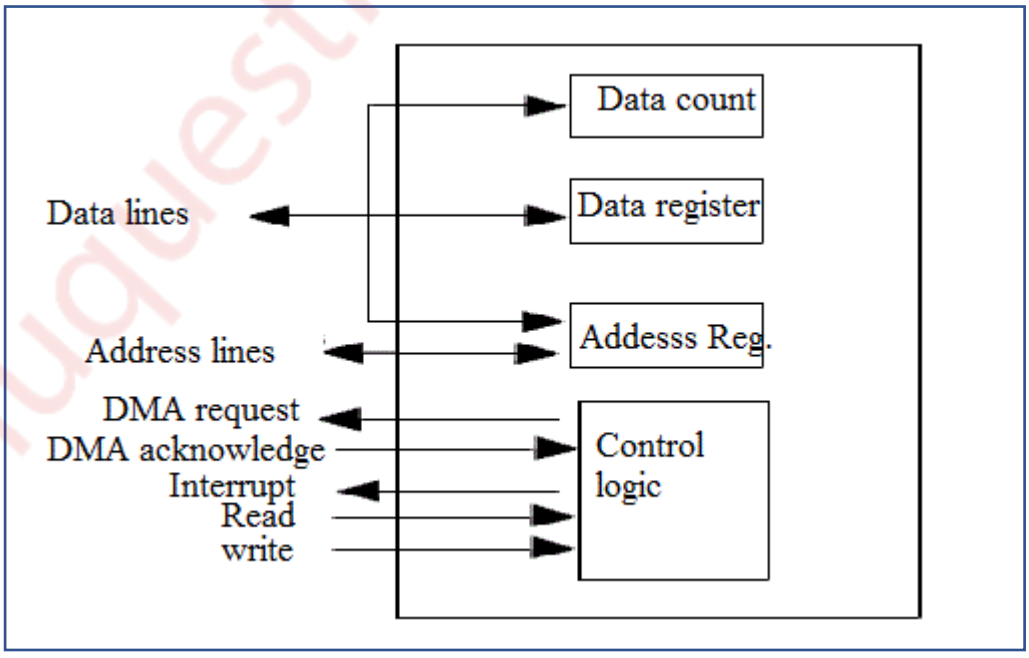
- The interrupt driven I/O mechanism for transferring a block of data.
- After issuing the read command the CPU performs its work, but checks for the interrupt after every instruction cycle.
- When CPU gets an interrupt, it performs the following operation in sequence:
  - Save context i.e. the contents of the registers on the stack
  - Processes interrupt by executing the corresponding ISR
  - Restore the register context from the stack.

Transferring a word of data

- CPU issues a 'READ' command to I/O device and then switches to some other program. CPU may be working on different programs.
- Once the I/O device is ready with the data in its data register. I/O device signals an interrupt to the CPU.
- When then interrupt from I/O device occurs, it suspends execution of the current program, reads from the port and then resumes execution of the suspended program.

Data Transfer Modes

- DMA stands for Direct Memory Access. The I/O can directly access the memory using this method.
- Interrupt driven and programmed I/O require active operation of the CPU. Hence transfer rate is limited and CPU is also busy doing the transfer operation. DMA is the solution to this problem.
- DMA controller takes over the control of the bus form CPU for I/O transfer.



- The address register is used to hold the address of the memory location from which the data is to be transferred. There may be multiple address registers to hold multiple addresses.
- The address may be incremented or decremented after every transfer based on mode of operation.
- The data count register is used to keep a track of the number of bytes to be transferred. The counter register is decremented after every transfer.
- The data register is used in a special case i.e. when the transfer of a block is to be done from one memory location to another memory location.
- The DMA controller is initially programmed by the CPU, for the count of bytes to bee transferred address of the memory block for the data to be transferred etc.
- During this programming DMAC, the read and write lines work as inputs for DMAC.
- Once the DMAC takes the control of the system bus i.e. transfers the data between the memory and I/O device, these read and write signals work as output signals.
- They are used to tell the memory that the DMAC wants to read or write from the memory according to the operation being data transfer from memory to I/O or from I/O to memory.

DMA Transfer Modes:

- **Single transfer mode:** In this, the device is programmed to make one byte transfer only after getting the control of system bus.
- After transferring one byte the control of the bus will be returned back to the CPU.
- The word count will be decremented and the address decremented or incremented following each transfer.
- **Block transfer Mode:** In this, the device is activated by DREQ or software request and continues making transfers during the service until a Terminal Count, or an external End of Process is encountered.
- The advantage is that the I/O device gets the transfer of data a very faster speed.
- **Demand Transfer Mode:** In this, the devices continues making transfer until a Terminal Count or external EOP is encountered, or until DREQ goes inactive.
- Thus, transfer may continue until the I/O device has exhausted its data handling capacity.
- **Hidden Transfer Mode:** In this, the DMA controller takes over the charge on the system bus and transfers data when processor does not needs system bus.
- The processor does not even realize of this transfer being taking place.
- Hence these transfer are hidden from the processor.

b) Consider a cache memory of 16 words. Each block consists of 4 words. Size of the main memory is 256 bytes. Draw associative mapping and calculate TAG, and word size. (10 M)

Ans:

Given:

Cache Memory = 16 words

Block consist of 4 words

Main memory = 256 bytes

Main memory = 256 * 16 words = $2^8 * 2^4 = 2^{12}$

TAG field = 2^{12} = 12 bits consist of both set field and TAG field.

SET + TAG = 12

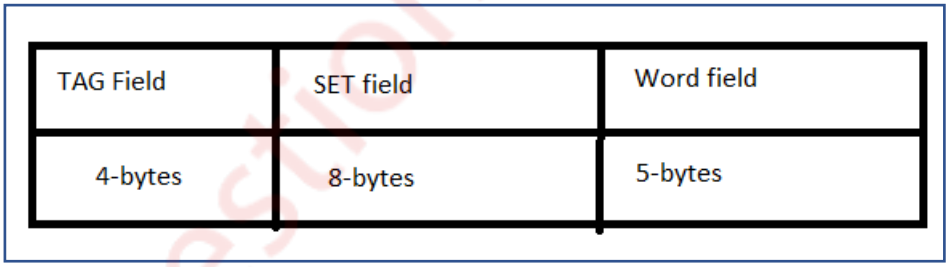
4 + TAG = 12

TAG = 12 – 4

TAG = 8 bytes.

Word size = As there are 8 blocks and each block consist of 4 words,

Hence 8 * 4 = 32 = 2^5



Q.6)

a) Write a short note on performance measures. (10 M)

Ans:

- There are various parameters used to measure the performance of a parallel system.
- Different parameters used to measure performance are:
- **Sequential Execution Time:** The time required for a program to be executed on a sequential system is called as the sequential execution time with respect to parallel processors. It is represented by T(1).
- **Parallel Execution Time:** The time required for a program to be executed on a n-parallel processor system is called as parallel execution time for 'n' processors. It is represented as T(n), where 'n' is the number of processors.

- **Speed Up:** The speed increase because of the parallel system compared to the uni-processor system is called as the speed up. It is the ratio of the speed of parallel system to that of the sequential system. It can also be given as the ratio of time required to execute a program on sequential system to that parallel system. It is very important metric to measure the performance of a parallel system. It is represented as S(n) and given as below:

  S(n) = T(1) / T(n)

- **Efficiency:** Efficiency of a parallel system is the ratio of thee actual speed-up obtained by a system to the ideal speed-up that should be achieved according to the number of processors in the parallel system. The ideal time required to execute a program using 'n' processors should be T(1)/n i.e. the time required should be 1/n of the time required on a sequential or single processor system. Thus the efficiency can be given as: **E(n) = Actual speed Up / Ideal Speed Up**
- **Clocks per Instruction:** This is as the name says a measure of the clock pulses require per instruction. It is the ratio of clock cycles required for a program to the number of instructions in the program. The time for one clock pulse is given as 't' and is the inverse of frequency (f). let the number of instructions in the program be 'I_c' thus the time required to execute a program (T) can be given as:

  $T = I_C * CPI * t$

- **Million Instruction Per Second(MIPS):** This is very widely used performance measure. As the name says it is the count of instructions executed per second in millions. Number of instruction executed in one second:

  Instruction Per Second = 1 / CPI * t

- Thus, the MIPS count of instruction can be given as:

  $MIPS = 1 / CPI * t * 10^6$

- **Million Floating point Instructions per second (MFLOPS):** This is similar to the MIPS, only the difference being here floating point instructions are taken into account. The same equations will work for MFLOPS, if the instruction count and CPI are replaced according to floating point Instructions.
- **Throughput:** The throughput of a system is defined as the number programs executed per unit time. This is represented as W_S and is given below:

  W_S = Number of programs / Time in seconds.

- **Scalability:** A parallel system is said to be scalable if the efficiency is obtained by increasing the number of processor. Since the efficiency is dependent on the number of processors, and it normally keeps the decreasing with the increase in the number of processors.

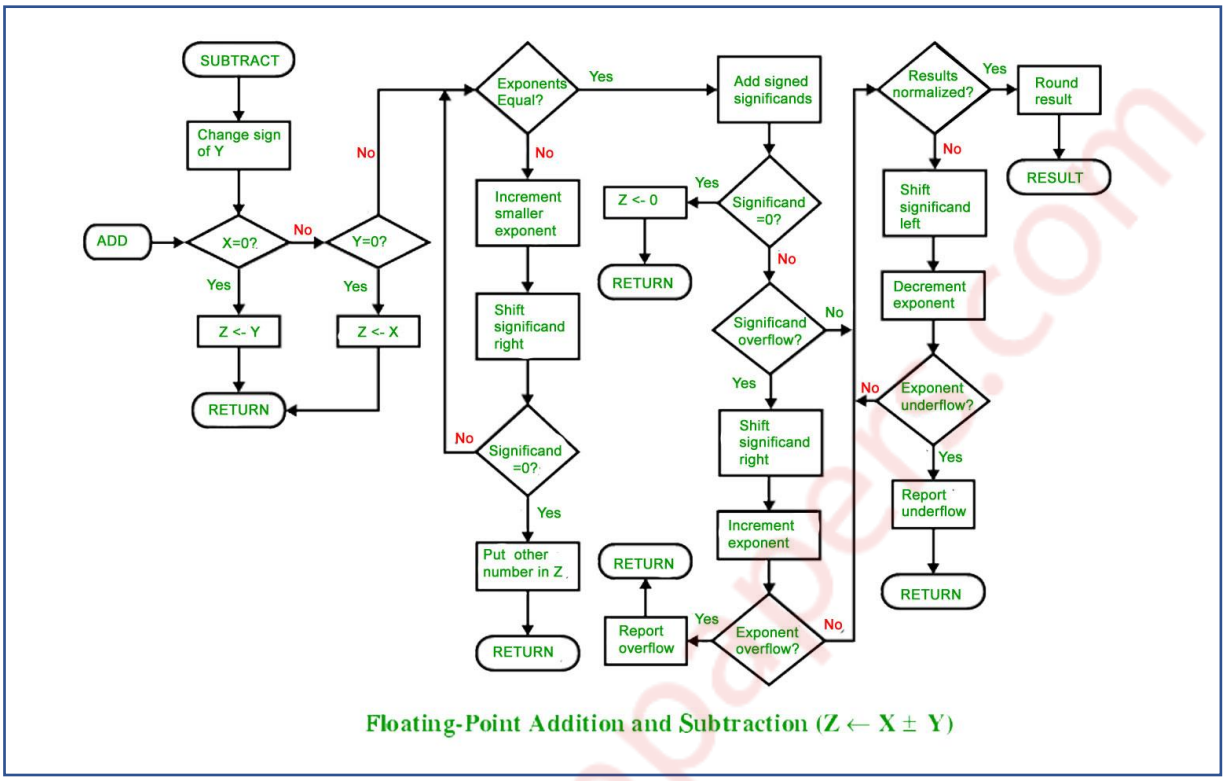
---

b) Draw and explain floating point addition and subtraction algorithm. (10 M)

Ans:

- In floating point arithmetic, addition and subtraction are more complex than multiplication and division. Addition and subtraction operations are carried out in four basic phases.

- Check for zeros
- Align the significant
- Add or subtract the significant
- Normalize the result



Floating-Point Addition and Subtraction ($Z \leftarrow X \pm Y$)

- In the next phase, exponents of the two numbers X and Y are made equal. Alignment is achieved by shifting either the smaller number to its right or shifting the larger number to the left.
- Since either operation may result is loss of digits, it is the smaller number is shifted. The alignment is achieved by repeatedly shifting the magnitude portion of the significant right 1 digit and incrementing the exponent until the two digit exponents are equal.
- Next, the two significant are added together, taking into account their signs. Since the sign may differ, the result may be 0. There is also the possibility of significant overflow.
- Next, result Is normalized. Normalisation consists of significant digits left until the most significant digits is nonzero. Each shift causes a decrement of the exponent and thus could cause an exponent overflow.

COMPUTER ORGANISATION AND ARCHITECTURE (MAY 2018)

Q.P. Code: 39078

Q.1) Write the following (20 M)

a) Compare Von Neumann architecture and Harvard Architecture. (5 M)

Ans:

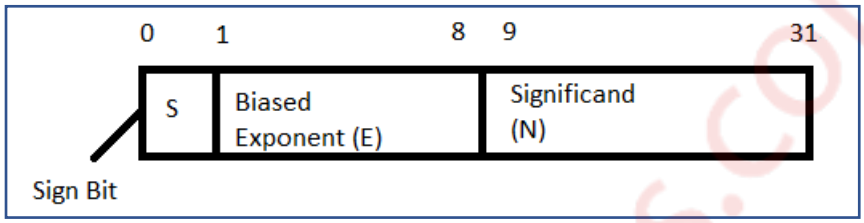
Harvard Architecture	Von Neumann Architecture
1. In Harvard architecture, the CPU is connected with both the data memory (RAM) and program memory (ROM), separately.	**1.** In Von-Neumann architecture, there is no separate data and program memory. Instead, a single memory connection is given to the CPU.
2. It requires more hardware since it will be requiring separate data and address bus for each memory.	**2.** In contrast to the Harvard architecture, this requires less hardware since only a common memory needs to be reached.
3. This requires more space.	**3.** Von-Neumann Architecture requires less space.
4. Speed of execution is faster because the processor fetches data and instructions simultaneously.	**4.** Speed of execution is slower since it cannot fetch the data and instructions at the same time.
5. It results in wastage of space since if the space is left in the data memory then the instructions memory cannot use the space of the data memory and vice-versa.	**5.** Space is not wasted because the space of the data memory can be utilized by the instructions memory and vice-versa.
6. Controlling becomes complex since data and instructions are to be fetched simultaneously.	**6.** Controlling becomes simpler since either data or instructions are to be fetched at a time.

b) Explain IEEE 754 floating point representation formats and represent $(34.25)_{10}$ to single precision format. (5 M)

Ans:

- IEEE floating point standards addresses a number o such problems.
- Zero has definite representation in IEEE format.

- +-∞ has been represented in IEEE format. A +∞ indicated that the result of an arithmetic expression is too large to be stored.
- If an underflow occurs, implying that a result is too small to be represented as a normalized number, it is encoded in a denormalized scale.
- Figure gives the representation of single precision floating point numbers.



Given: $(34.25)_{10}$

Step 1: $(34)_{10} = (00100010)_2$

$(0.25)_{10} = (0.01)_2$

Hence, $(34.25)_{10} = (00100010.01)_2$

Step 2: $(35.25)_{10} = (1.0001001)_2 * 2^5$

Step 3: Biased exponent = 127+5 = $(132)_{10} = (10000100)_2$

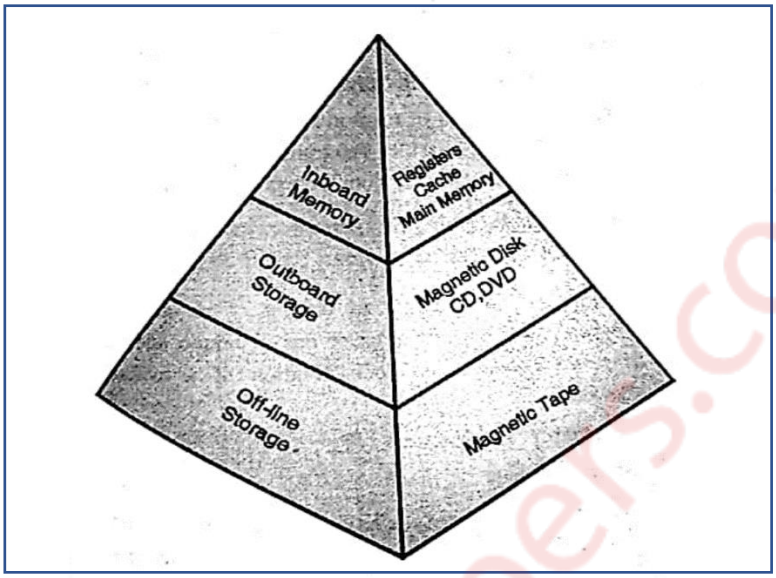
---

c) Explain memory hierarchy in the computer system (5 M)

Ans:

- Memory hierarchy explains that the nearer the memory to the processor, faster is its access. But costlier the memory becomes as it goes closer to the processor. The following sequence is in faster to slower or costlier to cheaper memory.
  - Registers i.e. inside the CPU.
  - Internal memory that includes one or more levels of cache and the main memory. Internal memory is always RAM, SRAM, DRAM for main memory. This is called as the primary memory.
  - External memory or removable memory includes the hard disk, CDs, DVDs etc. this is the secondary memory.
- The registers as discussed are the closest to the processor and hence are the fastest while off-line storage like magnetic tape are the farthest and also the slowest. The list of memories from closest to the processor to the farthest is given as below:
  - Registers
  - L1 cache
  - L2 cache

- Main memory
- Magnetic disk
- Optical
- Tape.



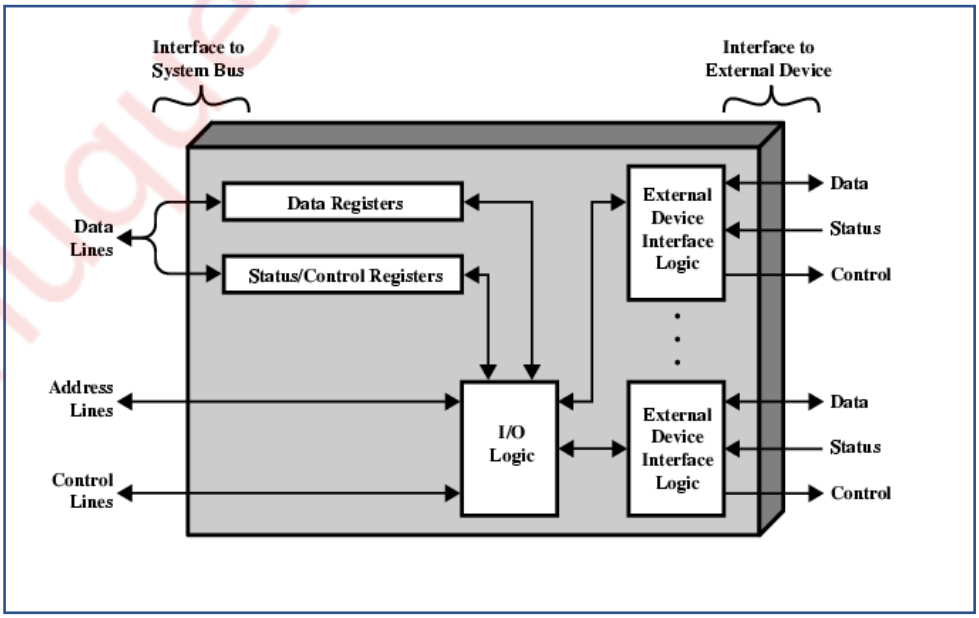
- To have a large faster memory is very costly and hence the different memory at different memory at different levels gives the memory hierarchy.

---

d) Explain the requirements of I/O modules. (5 M)

Ans:

- There are a wide variety of peripherals or I/O devices that deliver different amounts of data at different speeds and in different formats. All these devices are slower than CPU and RAM and hence to interface these devices to the CPU there is a need of I/O modules.

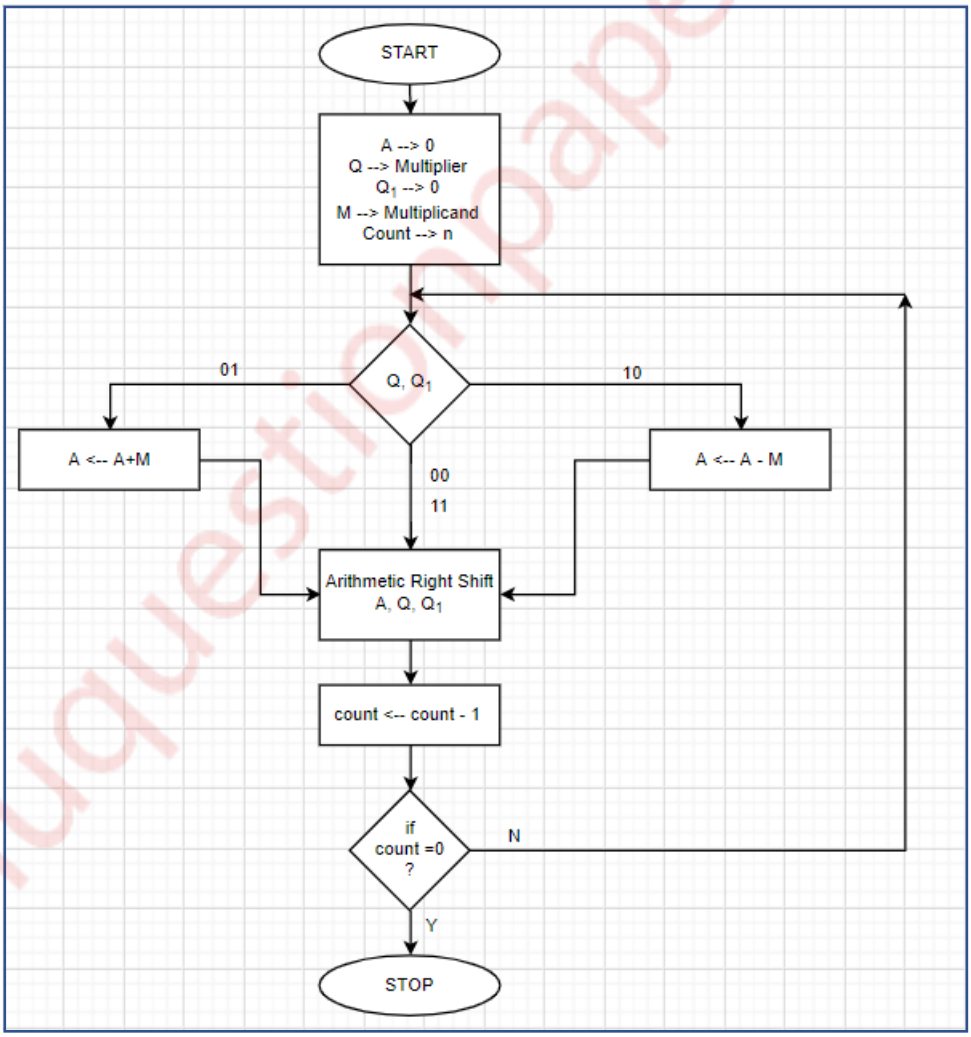


- I/O module is interface to CPU and memory with one or more peripherals.
- The general model of I/O module interfacing with system bus.
- The various functions of the I/O module involve:
    - Issue of control and timing signals.
    - Communication with CPU
    - Communication with peripheral
    - Buffering of data between the CPU and peripheral
    - Detection of errors.

Q.2)

a) Draw the flowchart of Booth's algorithm. Perform following multiplication using Booth's algorithm M = $(-9)_{10}$ Q = $(6)_{10}$ (10 M)

Ans:



Given: M = -9

$(9)_{10} = (01001)_2$

2's complement of 9 = 10111

M = 10111

And - M = 01001

$Q = (6)_{10} = 00110$

Multiplication will require 5 cycles as the register size n = 5 bits

AC	Q	Q-1	
00000	00110	0	
00000	00011	0	AC = AC – M, 00000 – 01001 = 01001
01001	00011	0	
00100	10001	1	
00010	01000	1	AC = AC + M, 00010 + 10111 = 11001
11001	01000	1	
01100	10100	0	
00110	01010	0	

Hence, product = 0011001010

= -(0000110110) = -54

---

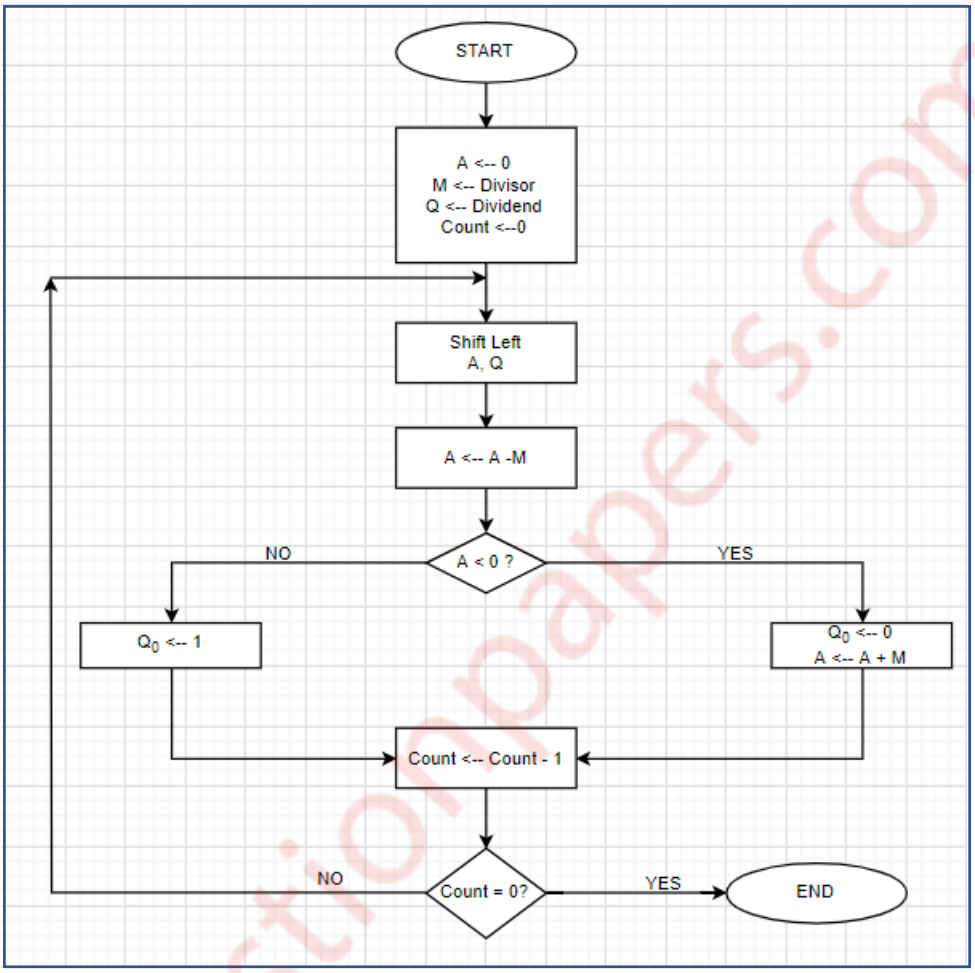
b) Explain the restoring method of binary division with algorithm. Divide $(7)_{10}$ by $(3)_{10}$ using restoring method of binary division. (10 M)

Ans:

- Similar to the multiplication algorithm to multiple binary numbers, we also have a method for division called as the restoring method of division for binary numbers.
- Here also we have registers namely 'A', 'M', 'Q' and count to store the result, dividend, divisor and count respectively.
- In this case we shift left the registers 'A' and 'Q' to their left, and then check whether the value in 'A' is greater than the divisor or not. This is done by subtracting the divisor from the value of register 'A'.
- To find out whether greater or not we check the result is positive or not. If yes then we put '1' in the LSB of the Q register, which was initially left blank while

shifting. If no, then we put a '0' in the LSB of the Q register and add the divisor back to the value of register 'A', hence name 'Restoring Division method'.

- The count is decremented and the above process is repeated until the count is not equal to zero.



Given: Q = $(7)_{10}$ = $(0111)_2$

M = $(3)_{10}$ = $(0011)_2$

A	Q	M	Count	Remark
0000	0111	0011	4	Initialisation
0000	1110			Left Shift
+ 1101	1011			
1101				A←A – M
+ 0011				
00000	1110		3	A←A + M
0001	1100			Left Shift
+ 1101				
1110				A←A – M
+ 0011				
0001	1100		2	A←A + M

0011 +1101 0000	1001 1001		 1	Left Shift A←A – M
0001 + 1101 1110 + 0011 0001	0010 0010		 0	Left shift A←A – M A← A + M

Quotient: $(0010)_2 = (2)_{10}$
Remainder: $(0001)_2$ $(1)_{10}$

Q.3)

a) What is necessity of cache memory? Explain set Associative cache Mapping. (10 M)

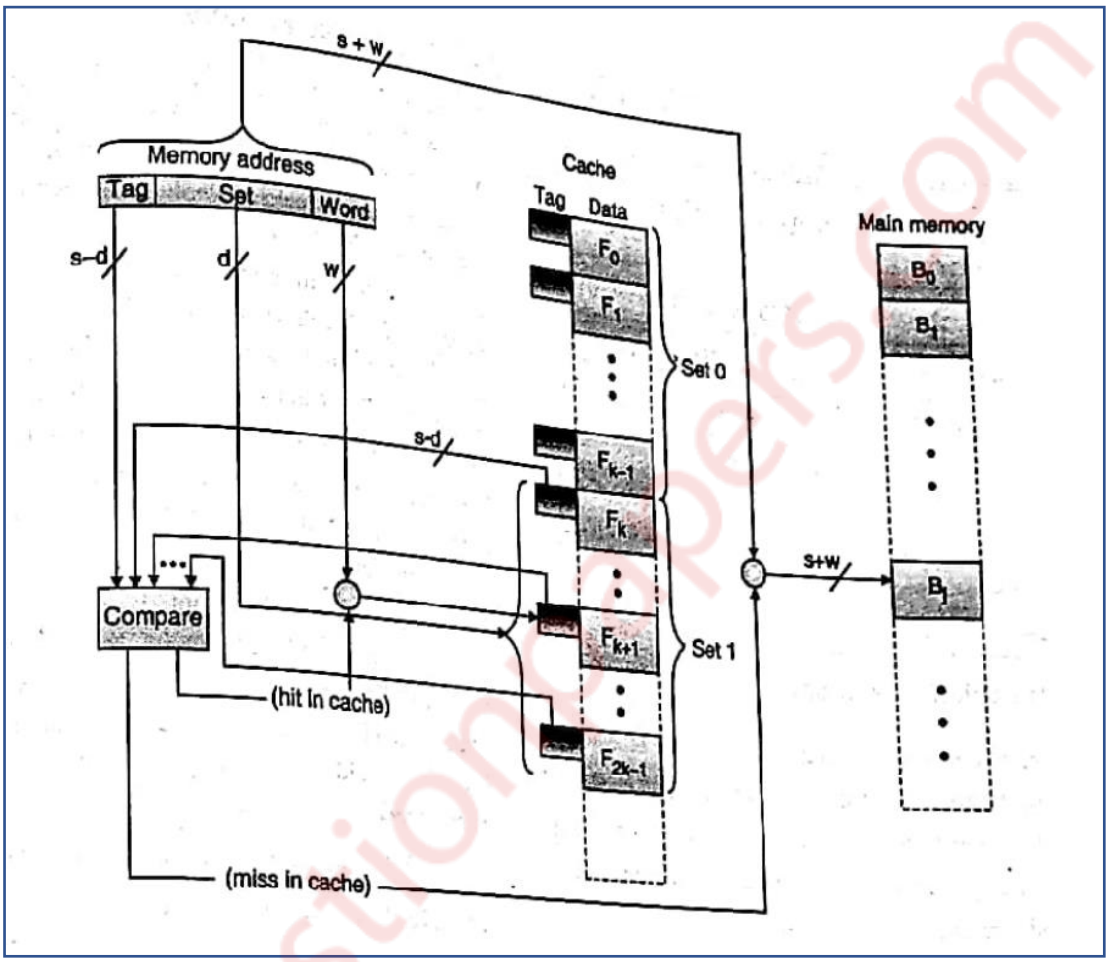
Ans:

- The cache memory lies in the path between the processor and the memory. The cache memory therefore, has lesser access time than memory and is faster than the main memory.
- The need for the cache memory is due to the mismatch between the speeds of the main memory and the CPU. The CPU clock as discussed earlier is very fast, whereas the main memory access time is comparatively slower.
- Hence, no matter how fast the processor is, the processing speed depends more on the speed of the main memory (the strength of a chain is the strength of its weakest link). It is because of this reason that a cache memory having access time closer to the processor speed is introduced.
- The cache memory stores the program (or its part) currently being executed or which may be executed within a short period of time. The cache memory also stores temporary data that the CPU may frequently require for manipulation.
- It acts as a high speed buffer between CPU and main memory and is used to temporary store very active data and action during processing since the cache memory is faster then main memory, the processing speed is increased by making the data and instructions needed in current processing available in cache. The cache memory is very expensive and hence is limited in capacity.

Set Associative Cache :

- In set associative cache mapping, cache is divided into a number of sets. Each set contains a number of lines.

- A given block maps to any line in a given set (i mod j), where i is the line number of the main memory to be mapped and j is the total number of sets in cache memory.
- For example, if there are 2 lines per set, it is called as 2 way associative mapping i.e. given block can be In one of 2 lines in only one set.



---

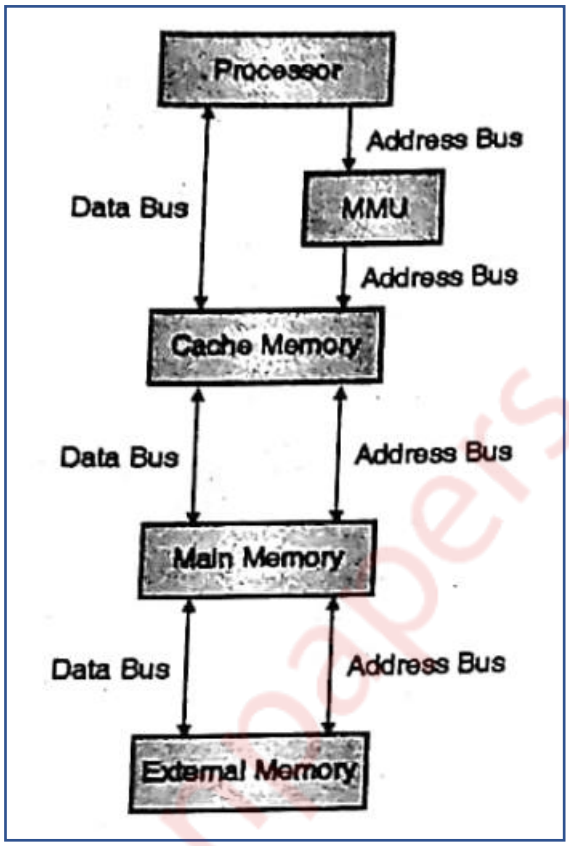
b) Explain the page address translation in case of virtual memory and explain TLB. (10 M)

Ans:

Virtual Memory:-

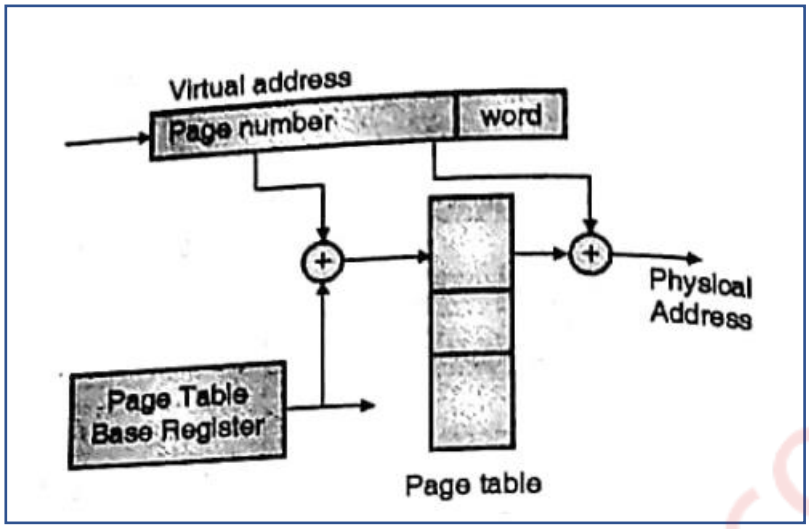
- Virtual Memory was introduced in the system in order to increase the size the size of memory.
- A hardware unit called Memory Management Unit (MMU) translates Virtual addresses into physical addresses.
- If CPU wants data from main memory and it is not present in main memory then MMU causes operating system to bring the data into the Memory from disk.

- As the disk limit is beyond the main memory address, the desired data address has to be translated from Virtual to physical address. MMU does the address translation.
- Figure below shows Virtual Memory Organization:-



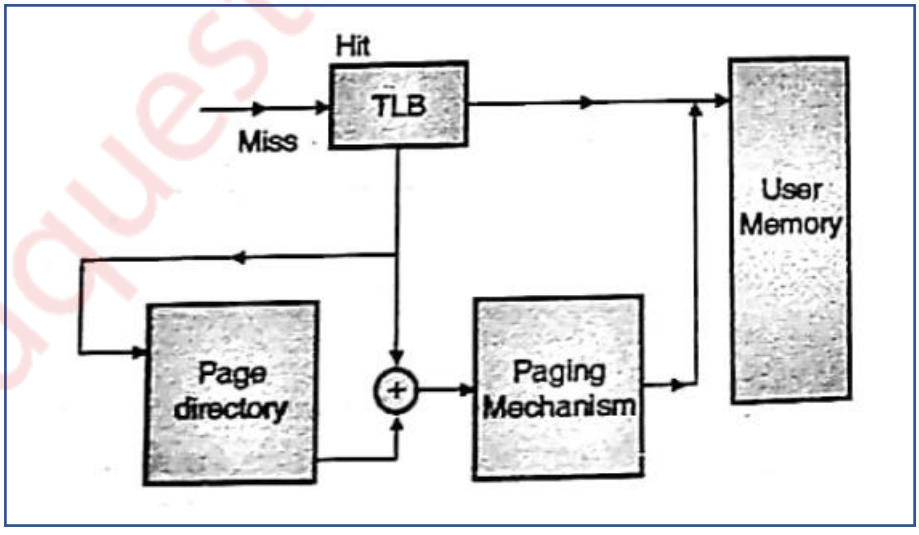
Paging:

- Virtual Memory space is divided into equal size pages.
- Main Memory space is divided into equal size page frames each frame can hold any page from Virtual Memory.
- When CPU wants to access page, it first looks into main memory. If it is found in main memory then it is called Hit and page is transfer from main memory to CPU.
- If CPU needs page that is not present in main memory then it is called as page fault. The page has to be loaded from Virtual Memory to main memory.
- There are different page replacement schemes such as FIFO, LRU, LFU, Random Etc.
- During page replacement, it the old page has been modified in the main memory, then it needs to be first copied into the Virtual Memory and then replaced. CPU keeps track of such updated pages by maintaining Dirty bit for each page. When page is updated in main memory dirty bit is set then this dirty page first copied into Virtual Memory & then replaced.
- Pages are loaded into main memory only when required by the CPU, then it is called demand paging. Thus pages are loaded only after page faults.



Translation Look-Aside Buffer (TLB) :-

- This is a on chip buffer within the CPU, used to speed up the paging process. Since a page from Virtual Memory can get stored into any frame of main memory, the OS maintains a page Table which indicates which page of virtual memory is stored in each page frame of main memory.
- Hence for accessing the page CPU has to perform 2 Memory Operations:-
- First access the page table to get information about where the page is stored in main memory, than access the main memory for the page. To solve this problems CPU copies the pages table information of the most recently used pages in the on-chip TLB. Therefore, subsequent access to the pages will be faster and information will be provided by the TBL and CPU need not Access the Table.

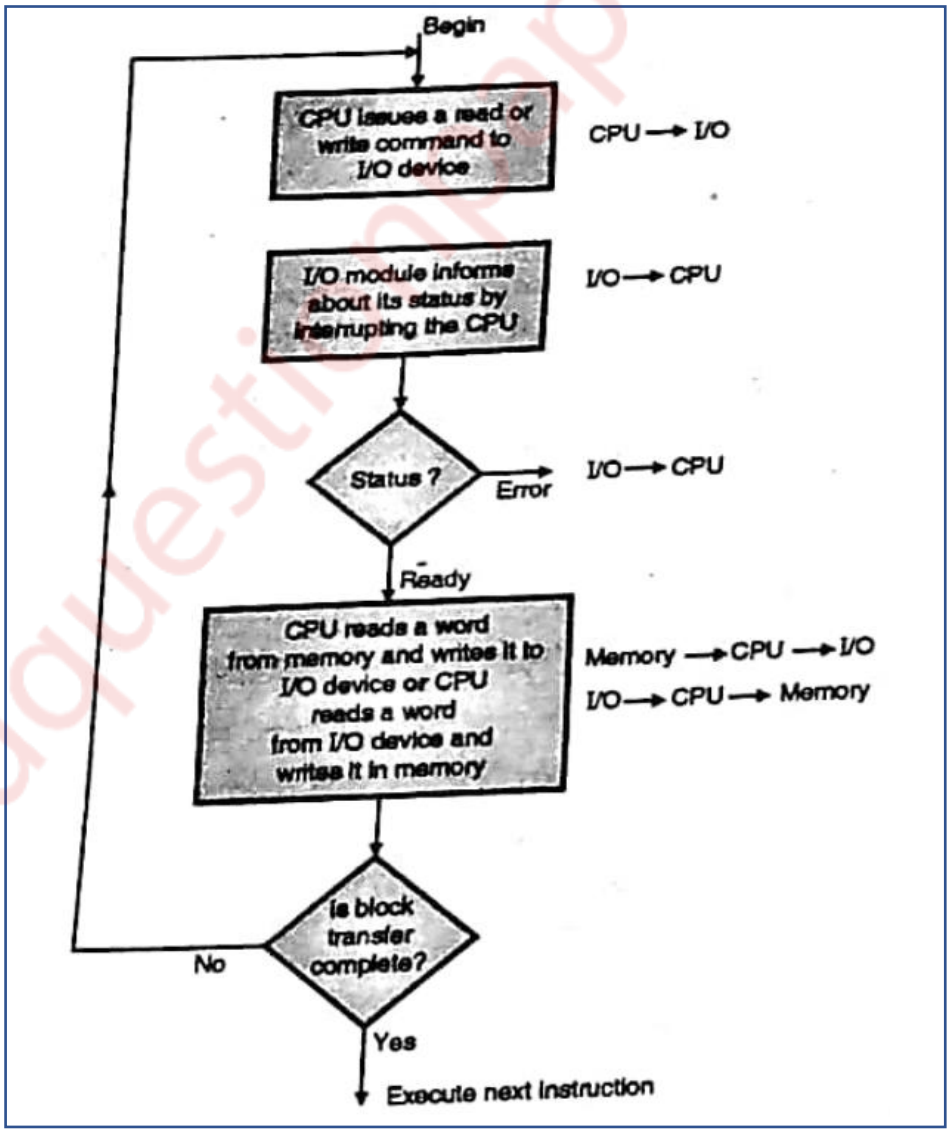


Q.4)

a) Explain interrupt driven I/O method of data transfer. (10 M)

Ans:

- Interrupt driven I/O overcomes the disadvantage of programmed I/O i.e. the CPU waiting for I/O device.
- This disadvantage is overcome by CPU not repeatedly checking for the device being ready or not instead the I/O module interrupts when ready.
- The sequence of operations for interrupt Driven I/O is as below:
    - CPU issues the read command to I/O device.
    - I/O module gets data from peripheral while CPU does other work.
    - Once the I/O module completes the data transfer from I/O device, it interrupts CPU.
    - On getting the interrupt, CPU requests data from the I/O module.
    - I/O module transfers the data to CPU.
- The interrupt driven I/O mechanism for transferring a block of data.



- After issuing the read command the CPU performs its work, but checks for the interrupt after every instruction cycle.
- When CPU gets an interrupt, it performs the following operation in sequence:
    - Save context i.e. the contents of the registers on the stack
    - Processes interrupt by executing the corresponding ISR
    - Restore the register context from the stack.

Transferring a word of data

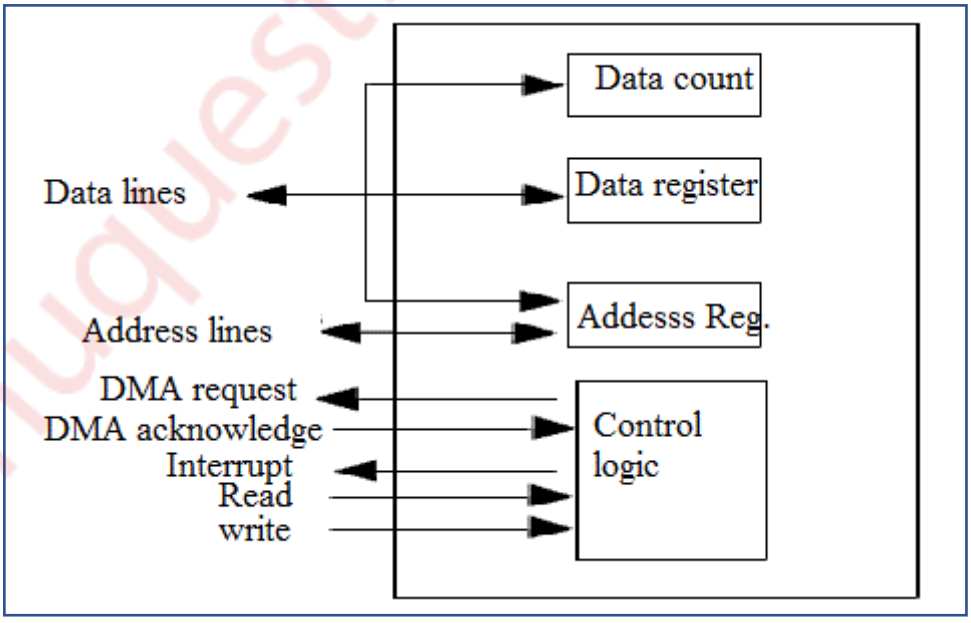
- CPU issues a 'READ' command to I/O device and then switches to some other program. CPU may be working on different programs.
- Once the I/O device is ready with the data in its data register. I/O device signals an interrupt to the CPU.
- When then interrupt from I/O device occurs, it suspends execution of the current program, reads from the port and then resumes execution of the suspended program.

---

b) Explain DMA method of I/O data transfer. (10 M)

Ans:

- DMA stands for Direct Memory Access. The I/O can directly access the memory using this method.
- Interrupt driven and programmed I/O require active operation of the CPU. Hence transfer rate is limited and CPU is also busy doing the transfer operation. DMA is the solution to this problem.
- DMA controller takes over the control of the bus form CPU for I/O transfer.



- The address register is used to hold the address of the memory location from which the data is to be transferred. There may be multiple address registers to hold multiple addresses.
- The address may be incremented or decremented after every transfer based on mode of operation.
- The data count register is used to keep a track of the number of bytes to be transferred. The counter register is decremented after every transfer.
- The data register is used in a special case i.e. when the transfer of a block is to be done from one memory location to another memory location.
- The DMA controller is initially programmed by the CPU, for the count of bytes to bee transferred address of the memory block for the data to be transferred etc.
- During this programming DMAC, the read and write lines work as inputs for DMAC.
- Once the DMAC takes the control of the system bus i.e. transfers the data between the memory and I/O device, these read and write signals work as output signals.
- They are used to tell the memory that the DMAC wants to read or write from the memory according to the operation being data transfer from memory to I/O or from I/O to memory.

DMA Transfer Modes:

- **Single transfer mode:** In this, the device is programmed to make one byte transfer only after getting the control of system bus.
- After transferring one byte the control of the bus will be returned back to the CPU.
- The word count will be decremented and the address decremented or incremented following each transfer.
- **Block transfer Mode:** In this, the device is activated by DREQ or software request and continues making transfers during the service until a Terminal Count, or an external End of Process is encountered.
- The advantage is that the I/O device gets the transfer of data a very faster speed.
- **Demand Transfer Mode:** In this, the devices continues making transfer until a Terminal Count or external EOP is encountered, or until DREQ goes inactive.
- Thus, transfer may continue until the I/O device has exhausted its data handling capacity.
- **Hidden Transfer Mode:** In this, the DMA controller takes over the charge on the system bus and transfers data when processor does not needs system bus.
- The processor does not even realize of this transfer being taking place.
- Hence these transfer are hidden from the processor.

Q.5)

a) Explain the superscalar Architecture. (10 M)

Ans:

- Superscalar processor are those processors that have multiple execution units.
- Hence these processors can execute the independent instructions simultaneously and hence with the help of this parallelism it increases the speed of the processor.
- It has been that the number of independent consecutive instructions 2 to 5. Hence the instruction issue degree in a superscalar processor is restricted from 2 to 5.

Pipelining in Superscalar Processor:

- The pipelining is the most important representation of demonstrating the speed increase by the superscalar feature of processor.
- Hence to implement multiple operations simultaneously, we need to have multiple execution units to execute each instruction independently.
- The ID and rename unit, decodes the instruction and then by the use of register renaming avoids instruction dependency. The instruction window takes the decoded instructions and based on some pair ability rules, issues them to the respective execution units.
- The instructions once executed move to the Retire and write back unit, wherein the instructions retire and the result is written back to the corresponding destination.

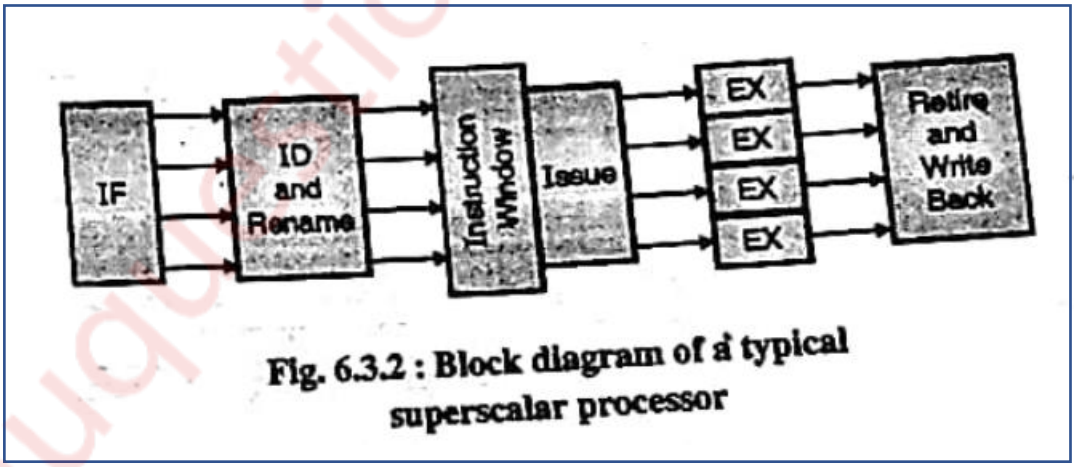
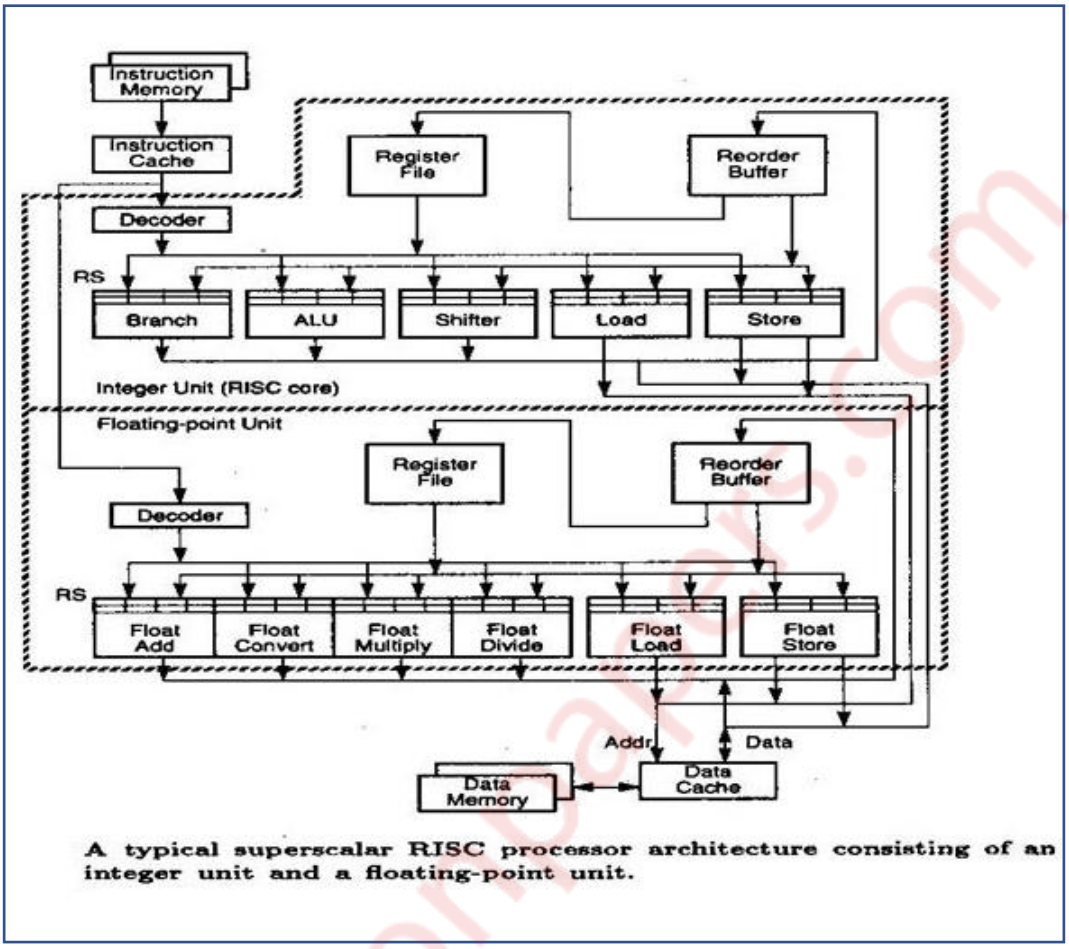


Fig. 6.3.2 : Block diagram of a typical superscalar processor

- A RISC or CISC processors execute one instruction per cycle. Their performance can be improved with superscalar architecture:
  - Multiple instruction pipelines are used.
  - Multiple instructions are issued for execution per cycle.
  - Multiple results are generated per cycles.

- Superscalar processors can exploit more instruction level parallelism in user program.



A typical superscalar RISC processor architecture consisting of an integer unit and a floating-point unit.

---

b) State the functions of control unit. Explain Micro-programmed control unit. (10 M)

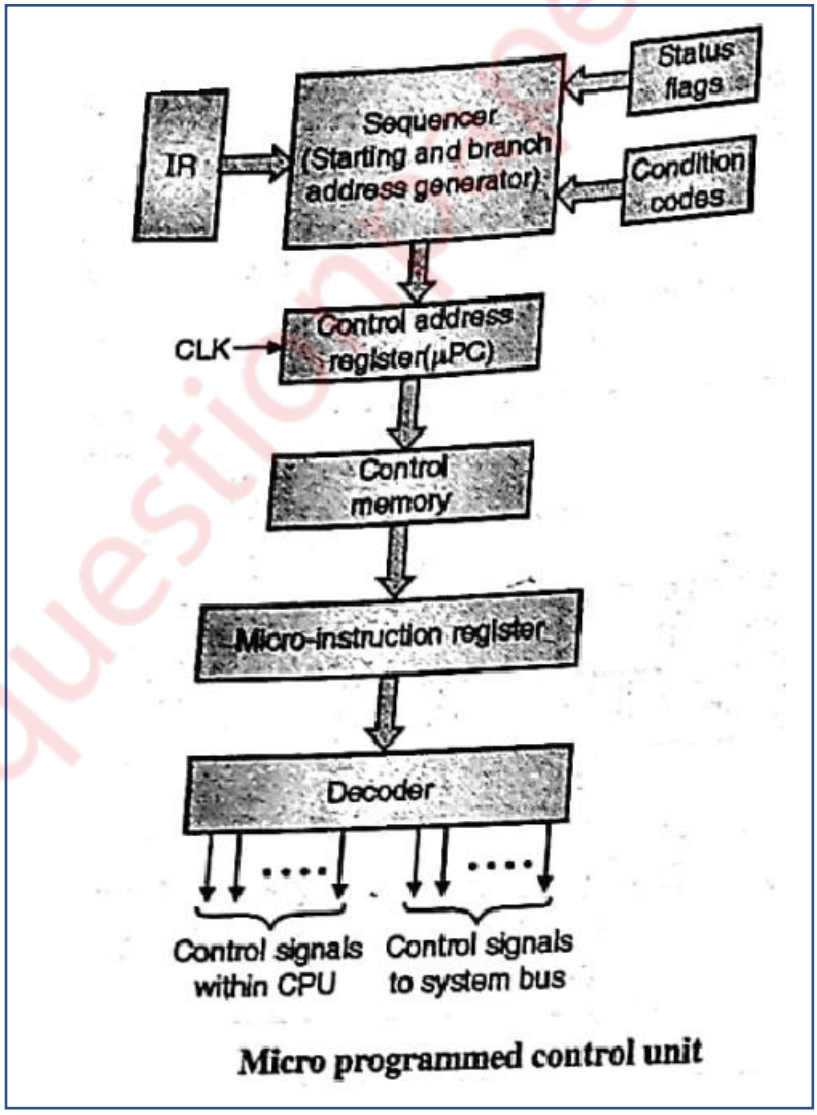
Ans:

Functions of Control Unit:

- Fetching instructions one by one from primary memory and gather required data and operands to perform those instructions.
- Sending instructions to ALU to perform additions, multiplication etc.
- Receiving and sending results of operations of ALU to primary memory
- Fetching programs from input and secondary memory and bringing them to primary memory
- Sending results from ALU stored in primary memory to output

Micro-Programmed Control Unit:

- Micro programmed control unit generates control signals based on the microinstructions stored in a special memory called as the control memory.
- Each instruction points to a corresponding location in the control memory that loads the control signals in the control register.
- The control register is then read by a sequencing logic that issues the control signals in a proper sequence.
- The implementation of the micro programmed.
- The instruction register (IR), status flag and condition codes are read by the sequencer that generates the address of the control memory location for the corresponding instruction in the IR.
- This address is stored in the control address register that selects one of the locations in the control memory having the corresponding control signals.
- These control signals are given to the microinstruction register, decoded and then given to the individual components of the processor and the external devices.



Micro programmed control unit

Q.6) Write Short notes **(20 M)**

a) Principle of locality of reference: **(10 M)**

Ans:

- Locality of reference is the term used to explain the characteristics of program that run in relatively small loops in consecutive memory locations.
- The locality of reference principle comprises of two components:
  - Temporal locality.
  - Spatial locality.
- **Temporal locality:** since the programs have loops, the same instructions are required frequently, i.e. the programs tend to use the most recently used information again and again.
- If for a long time a information in cache is not used, then it is less likely to be used again.
- This is known as the principle of temporal locality.
- **Spatial Locality:** Programs and the data accessed by the processor mostly reside in consecutive memory locations.
- This means that processor is likely to need code or data that are close to locations already accessed.
- This is known as the principle of spatial Locality.
- The performance gains are realized by the use of cache memory subsystem are because of most of the memory accesses that require zero wait states due to principles of locality.

---

b) Instruction Pipelining and its hazards. **(10 M)**

Ans:

- Instruction pipelining is a technique for overlapping the execution of several instructions to reduce the execution time of set of instructions.
- Generally, the processor fetches an instruction from memory, decodes it to determine what the instructions was, read the instruction inputs from the register file, performs the computation required by the instruction and writes the result back into the register file, this approach is called unpipelined approach.
- The problem with this approach is that, the hardware needed to perform each of these steps fetch, instruction decode, register read, instruction is different of the hardware is idle at any given moment, waiting for the other parts of the processor to complete their part of executing an instruction.
- It is a technique for overlapping the execution of several to reduce the execution time of set of instructions.
- Each instruction takes the same amount of time to execute in a pipelined processor as it would in a pipelined processor, but the rate at which instructions can be executed in increased by overlapping instruction execution.

- Latency is the amount of time that a single operation takes to execute.
- Throughput is the rate at which operations get executed.
- In a non-pipelined processor,

  Throughput = 1/latency

- In a pipelined processor,

  Throughput > 1/latency

Pipeline Hazards:

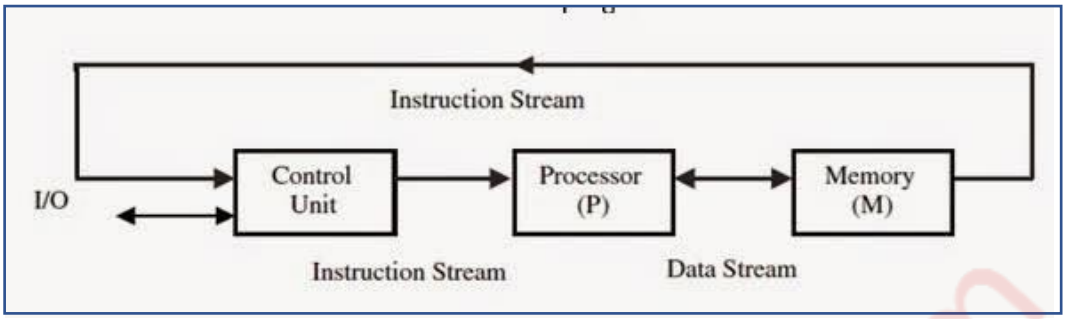
- Instruction hazards occur when instructions read or write registers that are used by other instructions. The type of conflicts are divided into three categories:
  - Structural Hazards (resource conflicts)
  - Data Hazards (Data dependency conflicts)
  - Branch difficulties (Control Hazards)
- **Structural hazards:** these hazards are caused by access to memory by two instructions at the same time. These conflicts can be slightly resolved by using separate instruction and data memories.
- It occurs when the processor's hardware is not capable of executing all the instructions in the pipeline simultaneously.
- **Data Hazards:** This hazard arises when an instruction depends on the result of a previous instruction, but this result is not available.
- **Branch Hazards:** Branch instructions, particularly conditional branch instructions, create data dependencies between the branch instruction and the previous instruction, fetch stage of the pipeline.

---

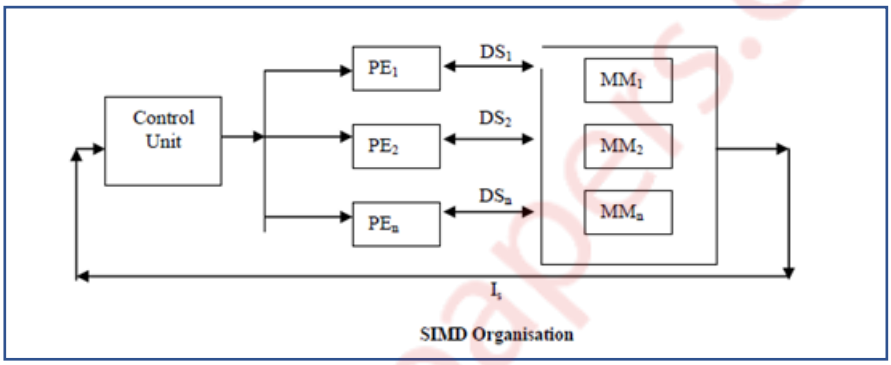
c) Flynn's Classification.

Ans:

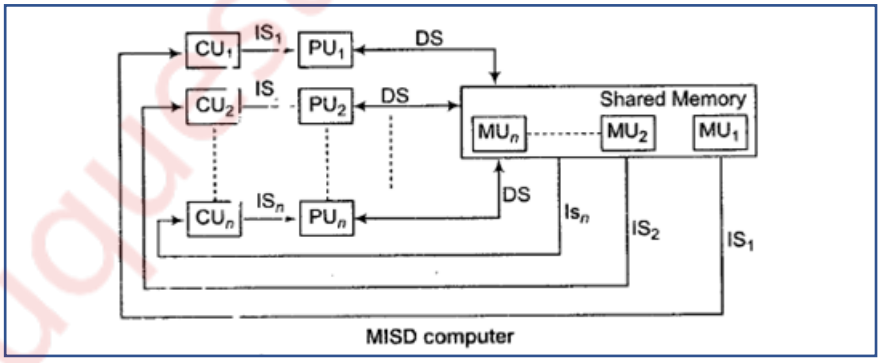
- A method introduced by Flynn, for classification of parallel processors is most common. This classification is based on the number of instruction Streams and Data Streams in the system. There may be single or multiple streams of each of these. Hence accordingly, Flynn classified the parallel processing into four categories:
  - Single instruction Single Data (SISD)
  - Single instruction Multiple Data (SIMD)
  - Multiple Instruction Single Data (MISD)
  - Multiple Instruction Multiple Data (MIMD)
- **SISD:** In this case there is a single processor that executes one instruction at a time on single data stored in the memory.
- In fact, this type of processing can be said to be unit processing, hence unit processors fall into this category.
- The processing Element accesses the data from the memory and performs the operation on this data as per the signal given by control unit.



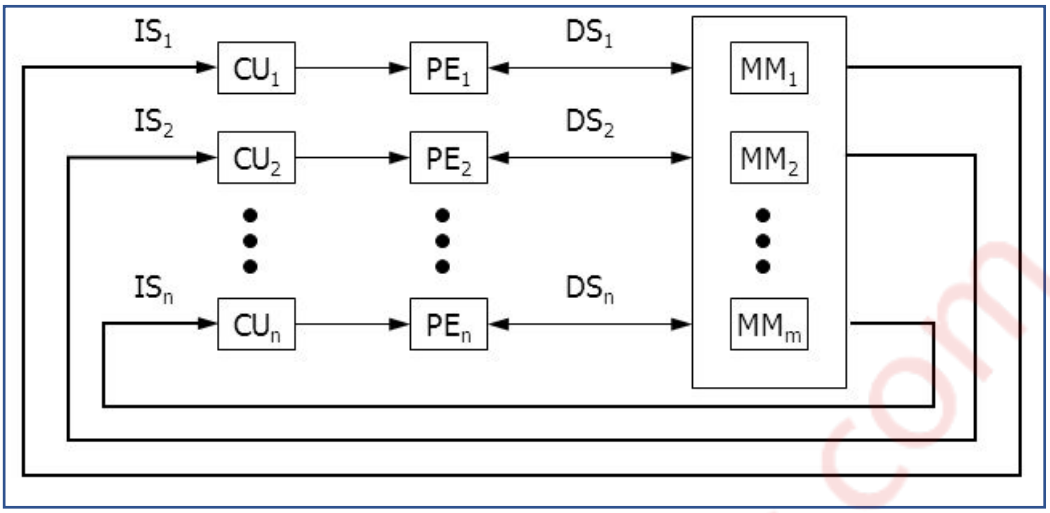
- **SIMD:** In this case the same instruction is given to multiple processing elements, but different data.
- This kind of system is mainly used when many data have to be operated with same operation.



- **MISD:** In case of MISD, there are multiple instruction streams and hence multiple control units to decode these instructions.
- Each control unit takes a different instructions from the different memory module in same memory.
- The data stream is single. In this case the data is taken by the first processing element.



- **MIMD:** This is a complete parallel processing example. Here each processing element is having a different set of data and different instructions.
- Examples of this kind of systems are SMPs, clusters and NUMA.

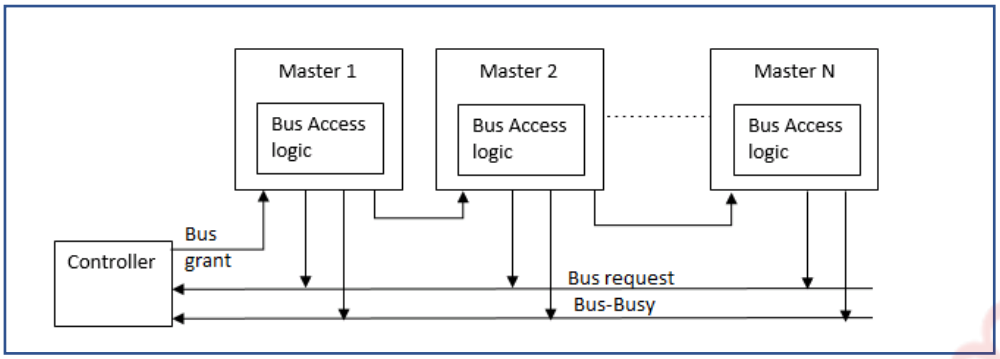


---

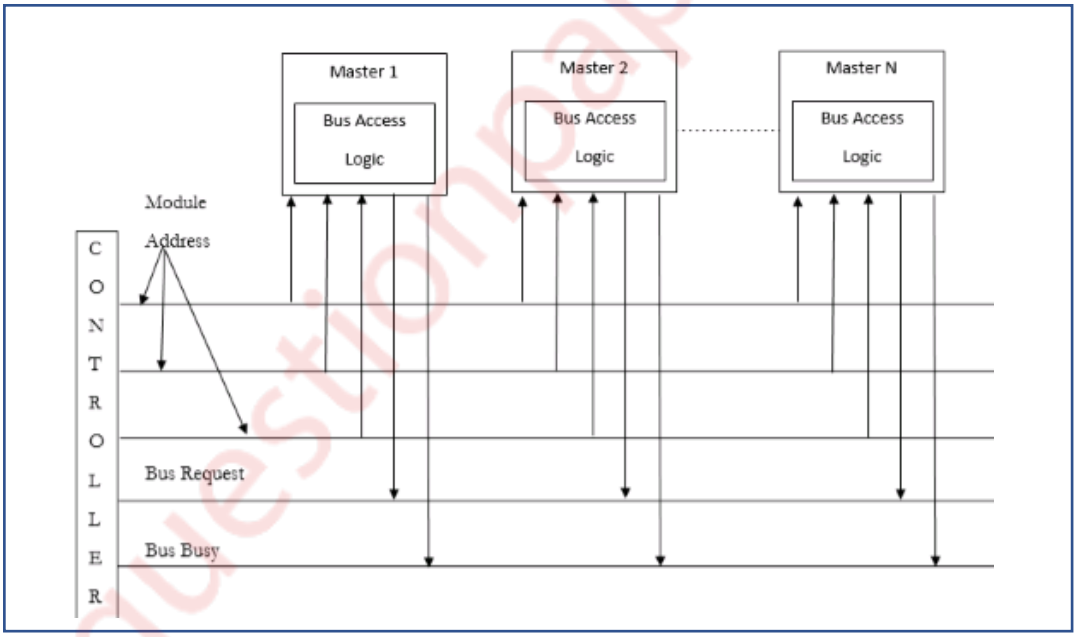
d) Bus Arbitration. (10 M)

Ans:

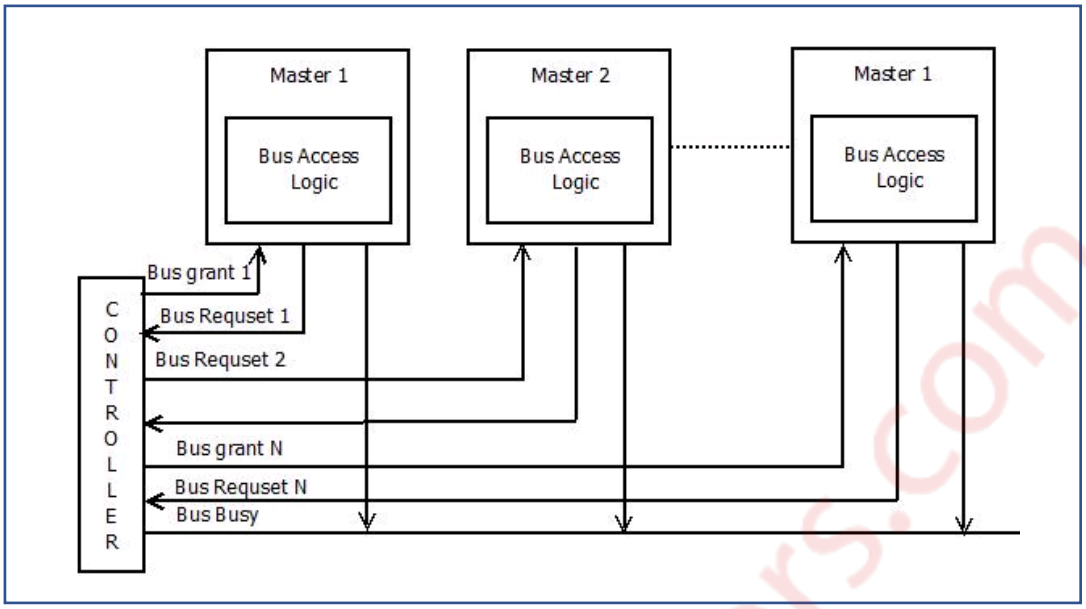
- When many bus masters are connected to a single bus, there is bus congestion. To avoid or reduce the problem of bus congestion we use bus arbitration.
- The process of selecting the processor among requesting processors is known ass arbitration.
- A selection mechanism must be based on fairness or priority basis.
- There are three representative arbitration schemes:
    - Daisy chaining
    - Polling
    - Independent Requesting.
- Arbitration process could be centralized or distributed. In the centralized scheme a hardware circuit device that is referred to as bus controller or bus arbiter decides about processor to be granted access of the bus among requesting processors.
- Similarly, CPU also needs the bus for various activities. Therefore, the system buses have I/O processor and CPU that need control of bus for data transfer.
- Now this is up to the bus controller to resolve the simultaneous data transfer requests on bus.
- **Daisy Chaining:** It is characterized by Bus Grant Signal connected serially from master. The process involves three control signals: bus busy, bus requests, bus grant.
- The controller responds to bus request signal only if bus busy is inactive. Bus busy signal remains active during the period it is being used by any of the processors.



- All processors are connected to the bus request line. A processor makes a request to controller for bus grant by activating Bus request Line.
- When the first unit requesting access to the bus receives bus grant signal, it blocks further propagation of signal, activities Bus busy signal, and starts using bus.
- Selection priority is determined by the proximity of the requesting processor from the controller.
- **Polling:** It is process of calling each master turn by turn. A master is called by its address. Address of a master is generated on poll count lines.
- Poll count lines are connected to each device. Bus request and bus busy line has the same meaning as in the context of daisy chaining.



- A request to use bus is made on the bus request line. Bus request will not be responded to till the bus busy line is active.
- When the poll count matches, the address of a particular master is requesting for the bus, the master activates the Bus busy signal and starts using the bus.
- **Independent Requesting:** In this scheme, each master has its independent bus request and Bus grant Line. In this scheme the identification of requesting master is almost immediate and the request can be responded quickly.



- The arbitration among masters is still carried out by a central arbiter. In case of multiple requests, a requesting master can be selected on the basis of priority.
- Normally, we use priority-based policy for I/O transactions and fairness-based policy among the processors.

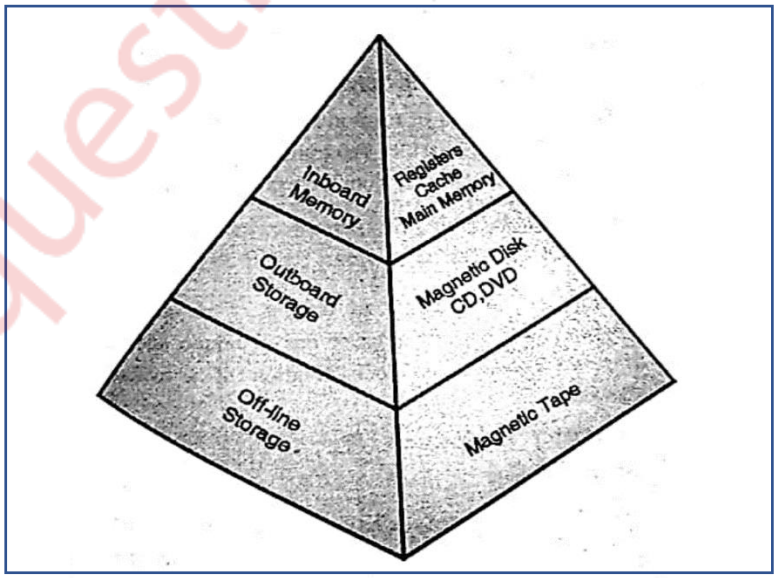
COMPUTER ORGANISATION AND ARCHITECTURE (DEC 2019)

Q.1) Write the following (20 M)

a) Describe the memory hierarchy in the computer system. (5 M)

Ans:

- Memory hierarchy explains that the nearer the memory to the processor, faster is its access. But costlier the memory becomes as it goes closer to the processor. The following sequence is in faster to slower or costlier to cheaper memory.
    - Registers i.e. inside the CPU.
    - Internal memory that includes one or more levels of cache and the main memory. Internal memory is always RAM, SRAM, DRAM for main memory. This is called as the primary memory.
    - External memory or removable memory includes the hard disk, CDs, DVDs etc. this is the secondary memory.
- The registers as discussed are the closest to the processor and hence are the fastest while off-line storage like magnetic tape are the farthest and also the slowest. The list of memories from closest to the processor to the farthest is given as below:
    - Registers
    - L1 cache
    - L2 cache
    - Main memory
    - Magnetic disk
    - Optical
    - Tape.



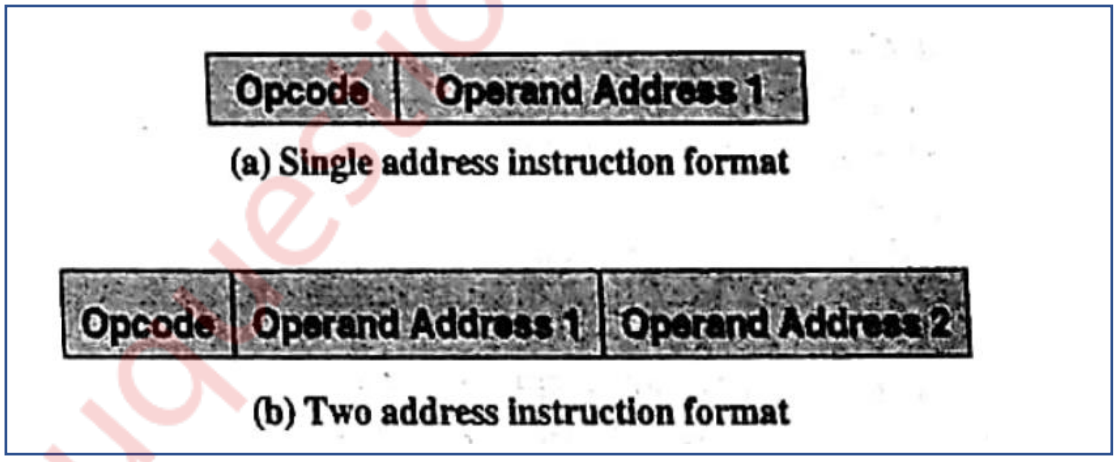
- To have a large faster memory is very costly and hence the different memory at different memory at different levels gives the memory hierarchy.

---

b) Give different instruction format. (5 M)

Ans:

- Input devices are required to give the instructions and data to the system. The output devices are used to give the output devices.
- The instructions and data given by the input device are to be stored, and for storage we require memory.
- Elements of Single, two and three address instructions are as follows:
    - Operation code is that part of the instruction which gives the code for the operation to be performed.
    - Source operand reference or address 1, gives the reference of the data on which the operation is to be performed. This address could be a register, memory or an input device.
    - Source operand reference or address 2, gives the reference of the second data on which the operation is to be performed. This address could again be a register, memory or an input device.
    - Result operand reference gives the reference where the result after performing operation is to be stored.
    - An instruction may have only one address with the other two fixed, or may have two addresses with one of the source operand address as the result operand address. Hence the instruction can have one, two or three addresses.



---

c) Explain principle of locality of reference in detail. (5 M)

Ans:

- Locality of reference is the term used to explain the characteristics of program that run in relatively small loops in consecutive memory locations.
- The locality of reference principle comprises of two components:
  - Temporal locality.
  - Spatial locality.
- **Temporal locality:** since the programs have loops, the same instructions are required frequently, i.e. the programs tend to use the most recently used information again and again.
- If for a long time a information in cache is not used, then it is less likely to be used again.
- This is known as the principle of temporal locality.
- **Spatial Locality:** Programs and the data accessed by the processor mostly reside in consecutive memory locations.
- This means that processor is likely to need code or data that are close to locations already accessed.
- This is known as the principle of spatial Locality.
- The performance gains are realized by the use of cache memory subsystem are because of most of the memory accesses that require zero wait states due to principles of locality.

---

d) Differentiate between Memory Mapped IO and IO Mapped IO. (5 M)

Ans:

Memory Mapped IO	IO Mapped IO
A method to perform IO operations between the CPU and peripheral devices in a computer that uses one address space for memory and IO devices.	A method to perform IO operations between the CPU and peripheral devices in a computer that uses two separate address spaces for memory and IO device.
Uses the same address space for both memory and IO devices.	Uses two separate address space for memory and IO device
As the memory mapped IO uses one address space for both IO and memory, the available addresses for memory are minimum.	All the addresses can be used by the memory.
Uses the same instructions for both IO and memory operations.	Uses separate instructions for read and write operations in IO and memory.
Less efficient	More Efficient.

---

e) Explain Superscalar Architecture. (5 M)

Ans:

- Superscalar processor are those processors that have multiple execution units.
- Hence these processors can execute the independent instructions simultaneously and hence with the help of this parallelism it increases the speed of the processor.
- It has been that the number of independent consecutive instructions 2 to 5. Hence the instruction issue degree in a superscalar processor is restricted from 2 to 5.

Pipelining in Superscalar Processor:

- The pipelining is the most important representation of demonstrating the speed increase by the superscalar feature of processor.
- Hence to implement multiple operations simultaneously, we need to have multiple execution units to execute each instruction independently.
- The ID and rename unit, decodes the instruction and then by the use of register renaming avoids instruction dependency. The instruction window takes the decoded instructions and based on some pair ability rules, issues them to the respective execution units.
- The instructions once executed move to the Retire and write back unit, wherein the instructions retire and the result is written back to the corresponding destination.

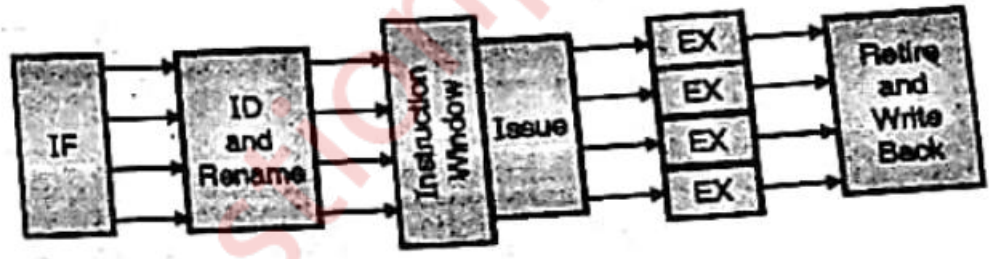
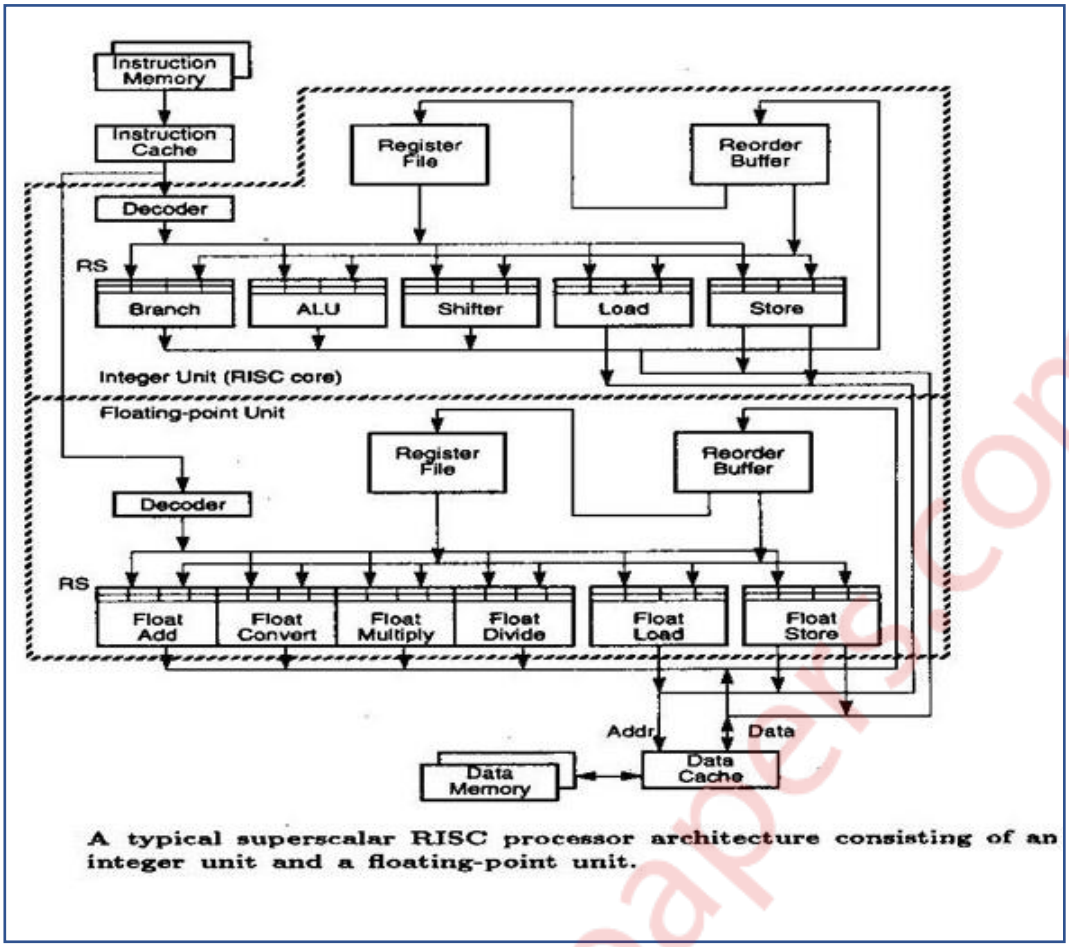


Fig. 6.3.2 : Block diagram of a typical superscalar processor

- A RISC or CISC processors execute one instruction per cycle. Their performance can be improved with superscalar architecture:
  - Multiple instruction pipelines are used.
  - Multiple instructions are issued for execution per cycle.
  - Multiple results are generated per cycles.

- Superscalar processors can exploit more instruction level parallelism in user program.



A typical superscalar RISC processor architecture consisting of an integer unit and a floating-point unit.

Q.2)

a) A program having 10 instructions (without Branch and Call Instructions) is executed on non-pipeline and pipeline processors. All instructions are of same length and having 4 pipeline stages and time required to each stage is 1nsecc.

I. Calculate time required to execute the program or Non-pipeline and Pipeline processor.

II. Calculate Speedup. (10 M)

Ans:

I. Given: n = 10 instructions, K = 4, t = 1nsec

Execution time pipelined = (4 + 100 - 1) * t

= (4 + 90) * 1

= 94 nsec.

Execution time unpipelined = (K * t) n

= (4 * 1) 1

= 4 nsec.

II. Speedup = 4/94 **= 0.043 times.**

---

b) With a neat diagram, explain branch prediction in detail. (10 M)

Ans:

- Branch prediction foretells the outcome of conditional branch instructions. Excellent branch handling techniques are essential for todays and for future microprocessors. Requirements of high performance branch handling.
    - An early determination of the branch outcome.
    - Buffering of the branch target address in a BTAC.
    - An excellent branch predictor and speculative execution mechanism.
    - An efficient rerolling mechanism when a branch is mis predicted.

Static Branch Prediction:

- It predicts always the same direction for the same branch during the whole program execution.
- It comprises hardware-fixed prediction and compiler-directed prediction.
- Simple hardware-fixed direction mechanism can be:
    - Predict always not taken
    - Predict always taken
    - Backward branch predict to be taken, forward branch prediction not to be taken
- Sometimes a bit in the branch opcode allows the compiler to decide the prediction direction.

Branch Target Buffer:

- The branch target buffer(BTB) stores branch and jump addresses, their target addresses, and optionally prediction information.
- The BTB is accessed during IF stage.

Branch Address	Target Address	Predicted Bits
……	……	…….

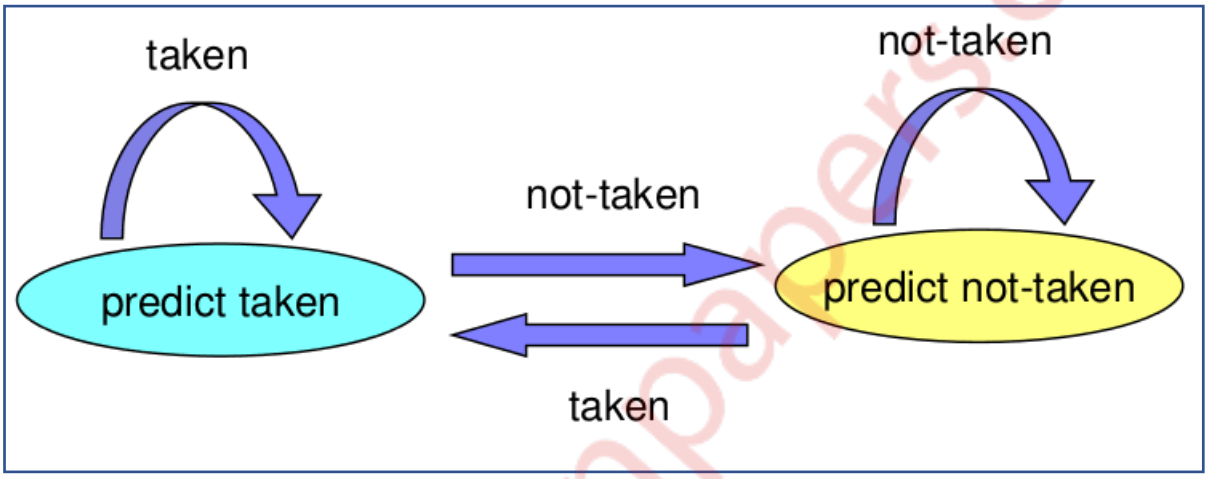
Dynamic Branch Prediction:

- The hardware influences the prediction while execution proceeds. Prediction Is decided on the computation of the program.
- During the start-up phase of the program execution, the history information is gathered and dynamic branch prediction gets more effective.

- In general, dynamic branch prediction gives better results than static branch prediction, but at the cost of increased hardware completely.

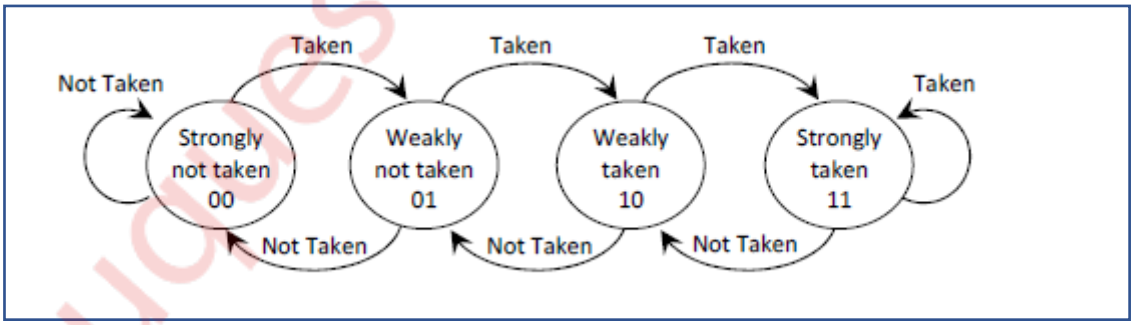
One-bit Dynamic Branch Predictor:

- A one-bit predictor correctly predicts a branch of end of loop iteration, as long as the loop does not exit.
- In nested loops, a one-bit prediction scheme will have two misprediction for the inner loop:
    - One at the end of the loop, when the iteration exits the loop instead of looping again, and one when executing the first loop iteration, when it predicts exit instead of looping.
    - Such a double misprediction is nested loops is available by a two-bit predictor scheme.



Two-Bit Prediction:

- A prediction must miss twice before it is changed when a two-bit prediction scheme is applied.



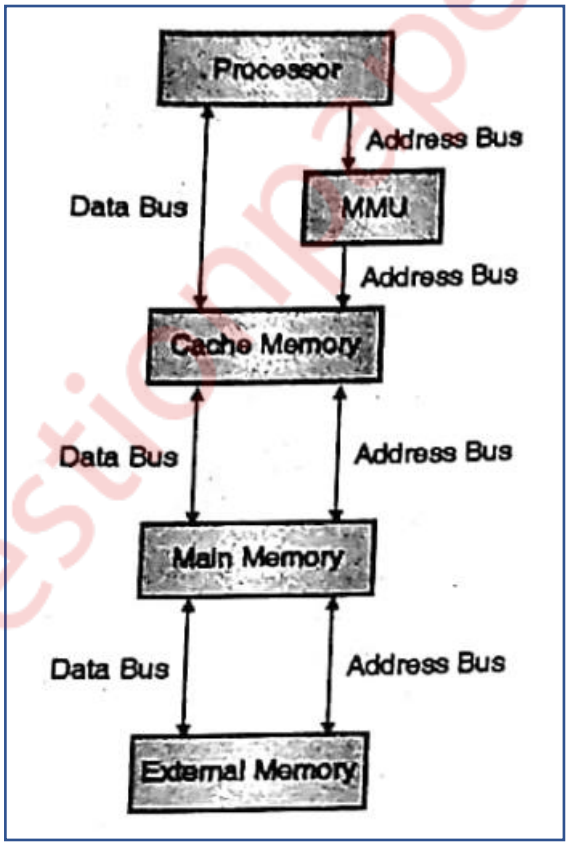
Q.3)

a) Explain page address translation with respect to virtual memory and further explain TLB in detail. (10 M)

Ans:

Virtual Memory:-

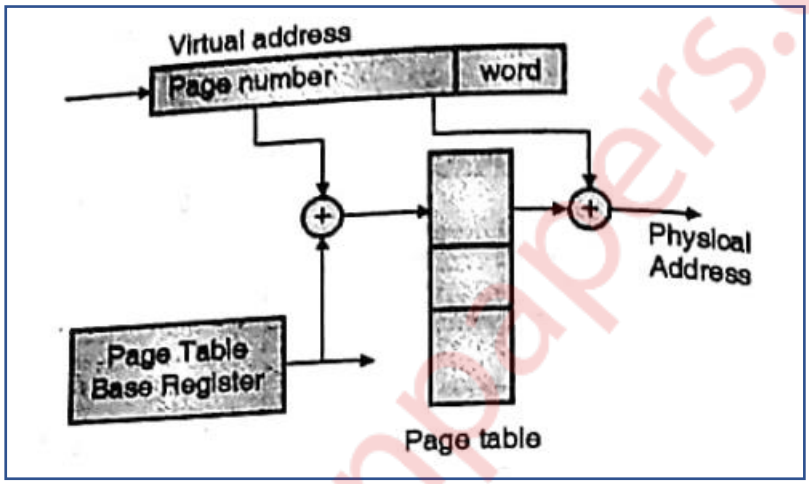
- Virtual Memory was introduced in the system in order to increase the size the size of memory.
- A hardware unit called Memory Management Unit (MMU) translates Virtual addresses into physical addresses.
- If CPU wants data from main memory and it is not present in main memory then MMU causes operating system to bring the data into the Memory from disk.
- As the disk limit is beyond the main memory address, the desired data address has to be translated from Virtual to physical address. MMU does the address translation.
- Figure below shows Virtual Memory Organization:-



Paging:

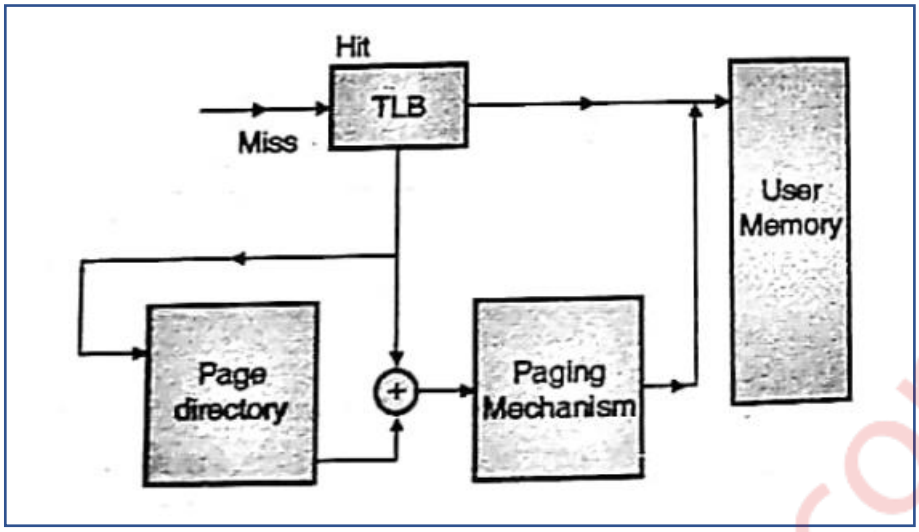
- Virtual Memory space is divided into equal size pages.
- Main Memory space is divided into equal size page frames each frame can hold any page from Virtual Memory.

- When CPU wants to access page, it first looks into main memory. If it is found in main memory then it is called Hit and page is transfer from main memory to CPU.
- If CPU needs page that is not present in main memory then it is called as page fault. The page has to be loaded from Virtual Memory to main memory.
- There are different page replacement schemes such as FIFO, LRU, LFU, Random Etc.
- During page replacement, it the old page has been modified in the main memory, then it needs to be first copied into the Virtual Memory and then replaced. CPU keeps track of such updated pages by maintaining Dirty bit for each page. When page is updated in main memory dirty bit is set then this dirty page first copied into Virtual Memory & then replaced.
- Pages are loaded into main memory only when required by the CPU, then it is called demand paging. Thus pages are loaded only after page faults.



Translation Look-Aside Buffer (TLB) :-

- This is a on chip buffer within the CPU, used to speed up the paging process. Since a page from Virtual Memory can get stored into any frame of main memory, the OS maintains a page Table which indicates which page of virtual memory is stored in each page frame of main memory.
- Hence for accessing the page CPU has to perform 2 Memory Operations:-
- First access the page table to get information about where the page is stored in main memory, than access the main memory for the page. To solve this problems CPU copies the pages table information of the most recently used pages in the on-chip TLB. Therefore, subsequent access to the pages will be faster and information will be provided by the TBL and CPU need not Access the Table.



---

b) What is micro program? Write microprogram for following operations

I. ADD R1, M, Register R1 and Memory location M are added and result store at Register R1.

II. MUL R1, R2 Register R1 and Register R2 are multiplied and result store at Register R1. (10 M)

Ans:

- Microprogramming is a process of writing microcode for a microprocessor. Microcode is low-level code that defines how a microprocessor should function when it executes machine-language instructions.
- Typically, one machine language instruction translates into several microcode instruction, on some computers, the microcode is stored in ROM and cannot be modified.
- **Micro Program to add R1, M.**

T-state	Operation	Microinstructions
T 1	PC → MAR	PCout, Marin, Read, Clear y, set Cin, Add, Zinn
T 2	M → MBR PC ← PC + 1	Zout, PCin, Wait for memory fetch cycle
T 3	MBR → IR	MBRout, IRin
T 4	R1 → x	R1out, Xin, CLRC
T 5	M → ALU	Mout, ADD, Zin
T 6	Z → R1	Zout, R1in
T 7	Check for intr	Assumption enabled intr pending, CLRX, SETC, Spout, SUB, Zin
T 8	SP ← SP – 1	Zout, Spin, MARin
T 9	PC → MDR	PCout, MDRin, WRITE

T 10	MDR → [SP]	Wait for Mem access
T11	PC ← IS Raddr	PCin IS Raddr out.

- **Micro Program to MUL R1, R2**

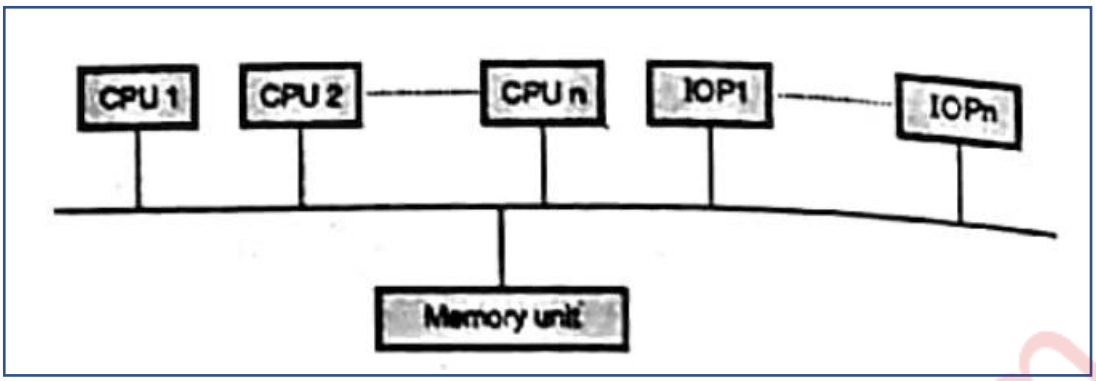
T-state	Operation	Microinstructions
T 1	PC → MAR	PCout, Marin, Read, Clear y, set Cin, Add, Zinn
T 2	M → MBR PC ← PC + 1	Zout, PCin, Wait for memory fetch cycle
T 3	MBR → IR	MBRout, IRin
T 4	R1 → x	R1out, Xin, CLRC
T 5	R2 → ALU	R2out, MUL, Zin
T 6	Z → R1	Zout, R1in
T 7	Check for intr	Assumption enabled intr pending, CLRX, SETC, Spout, SUB, Zin
T 8	SP ← SP – 1	Zout, Spin, MARin
T 9	PC → MDR	PCout, MDRin, WRITE
T 10	MDR → [SP]	Wait for Mem access
T11	PC ← IS Raddr	PCin IS Raddr out.

Q.4)

a) Explain Bus Contention and different method to resolve it. (10 M)

Ans:

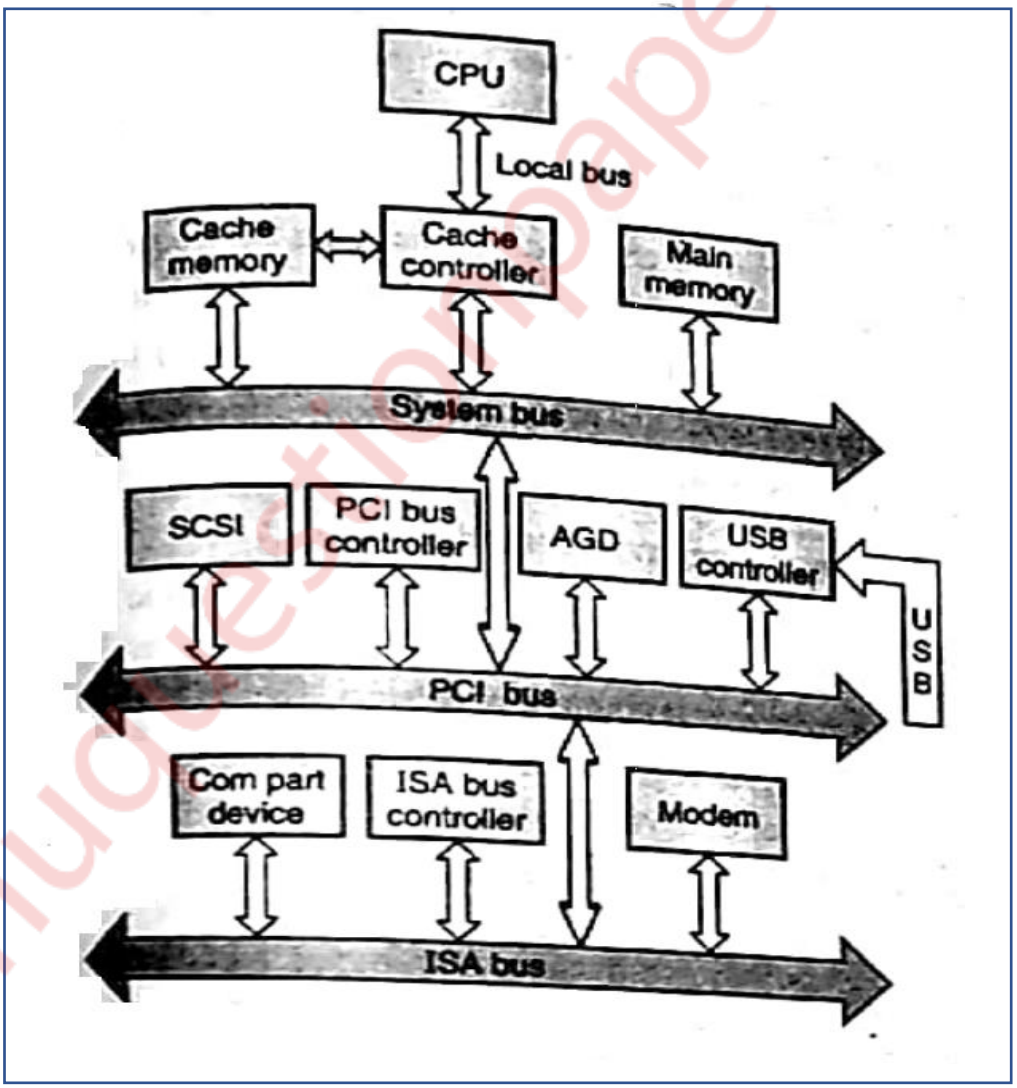
- In a bus system, processors, memory modules and peripheral devices are attached to the bus. The bus can handle only one transaction at a time between a master and slave. In case of multiple requests, the bus arbitration logic must be able to allocate or deallocate and it should service request at a time.
- Thus, such a bus is a time sharing or contention bus among multiple functional modules. As only one transfer can take place at any time on the bus, the overall performance of the system is limited by the bandwidth of the bus.
- When number of processors contending to acquire a bus exceeds the limit then a single bus architecture may become a major bottleneck. This may cause a serious delay in servicing a transaction.



- Aggregate data transfer demand should never exceed the capacity of the bus. This problem can be countered to some extent by increasing the data rate of the bus and by using a wider bus.
- Method of avoiding contention is multiple bus hierarchy.

Multiple-Bus Architecture:

- If a greater number of devices are connected to the bus, performance will suffer due to following reasons:



- In general, the more devices attached to the bus, the greater will be propagation delay.
- The bus may become a bottleneck as the aggregate data transfer demand approaches the capacity of the bus.
- This problem can be countered to some extent by increasing the data rate the bus can carry and by using wider buses.
- Most computer systems enjoy the use of multiple buses. These buses are arranged in a hierarchy.

---

b) Define instruction pipelining and its various hazards in detail. (10 M)

Ans:

- Instruction pipelining is a technique for overlapping the execution of several instructions to reduce the execution time of set of instructions.
- Generally, the processor fetches an instruction from memory, decodes it to determine what the instructions was, read the instruction inputs from the register file, performs the computation required by the instruction and writes the result back into the register file, this approach is called unpipelined approach.
- The problem with this approach is that, the hardware needed to perform each of these steps fetch, instruction decode, register read, instruction is different of the hardware is idle at any given moment, waiting for the other parts of the processor to complete their part of executing an instruction.
- It is a technique for overlapping the execution of several to reduce the execution time of set of instructions.
- Each instruction takes the same amount of time to execute in a pipelined processor as it would in a pipelined processor, but the rate at which instructions can be executed in increased by overlapping instruction execution.
- Latency is the amount of time that a single operation takes to execute.
- Throughput is the rate at which operations get executed.
- In a non-pipelined processor,

  Throughput = 1/latency

- In a pipelined processor,

  Throughput > 1/latency

Pipeline Hazards:

- Instruction hazards occur when instructions read or write registers that are used by other instructions. The type of conflicts are divided into three categories:
  - Structural Hazards (resource conflicts)
  - Data Hazards (Data dependency conflicts)
  - Branch difficulties (Control Hazards)
- **Structural hazards:** these hazards are caused by access to memory by two instructions at the same time. These conflicts can be slightly resolved by using separate instruction and data memories.

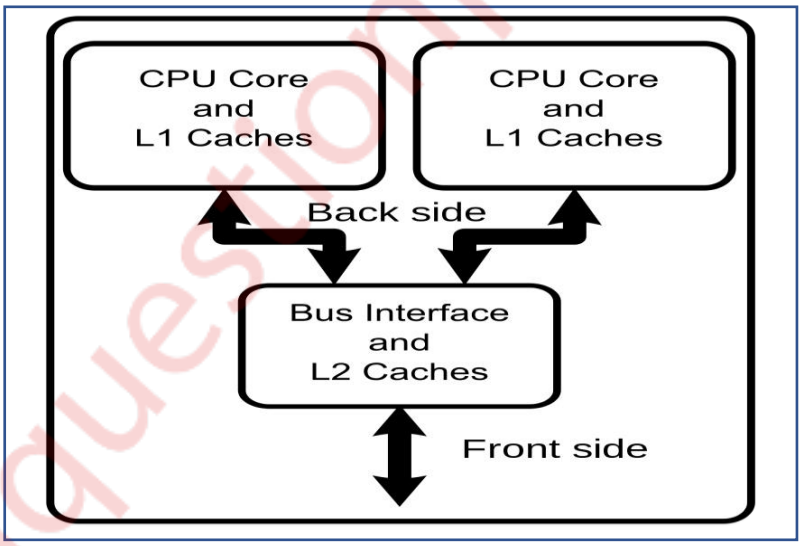
- It occurs when the processor's hardware is not capable of executing all the instructions in the pipeline simultaneously.
- **Data Hazards:** This hazard arises when an instruction depends on the result of a previous instruction, but this result is not available.
- **Branch Hazards:** Branch instructions, particularly conditional branch instructions, create data dependencies between the branch instruction and the previous instruction, fetch stage of the pipeline.

Q.5)

a) Explain multicore processor architecture in detail. (10 M)

Ans:

- A multi-core processor is a computer processor integrated circuit with two or more separate processing units, called cores, each of which reads and executes program instructions, as if the computer had several processors.
- The instructions are ordinary CPU instructions (such as add, move data, and branch) but the single processor can run instructions on separate cores at the same time, increasing overall speed for programs that support multithreading or other parallel computing techniques.
- Manufacturers typically integrate the cores onto a single integrated circuit die (known as a chip multiprocessor or CMP) or onto multiple dies in a single chip package. The microprocessors currently used in almost all personal computers are multi-core.



- A multi-core processor implements multiprocessing in a single physical package. Designers may couple cores in a multi-core device tightly or loosely. For example, cores may or may not share caches, and they may implement message passing or shared-memory inter-core communication methods.
- Common network topologies to interconnect cores include bus, ring, two-dimensional mesh, and crossbar. Homogeneous multi-core systems include only identical cores; heterogeneous multi-core systems have cores that are not identical

Just as with single-processor systems, cores in multi-core systems may implement architectures such as VLIW, superscalar, vector, or multithreading.

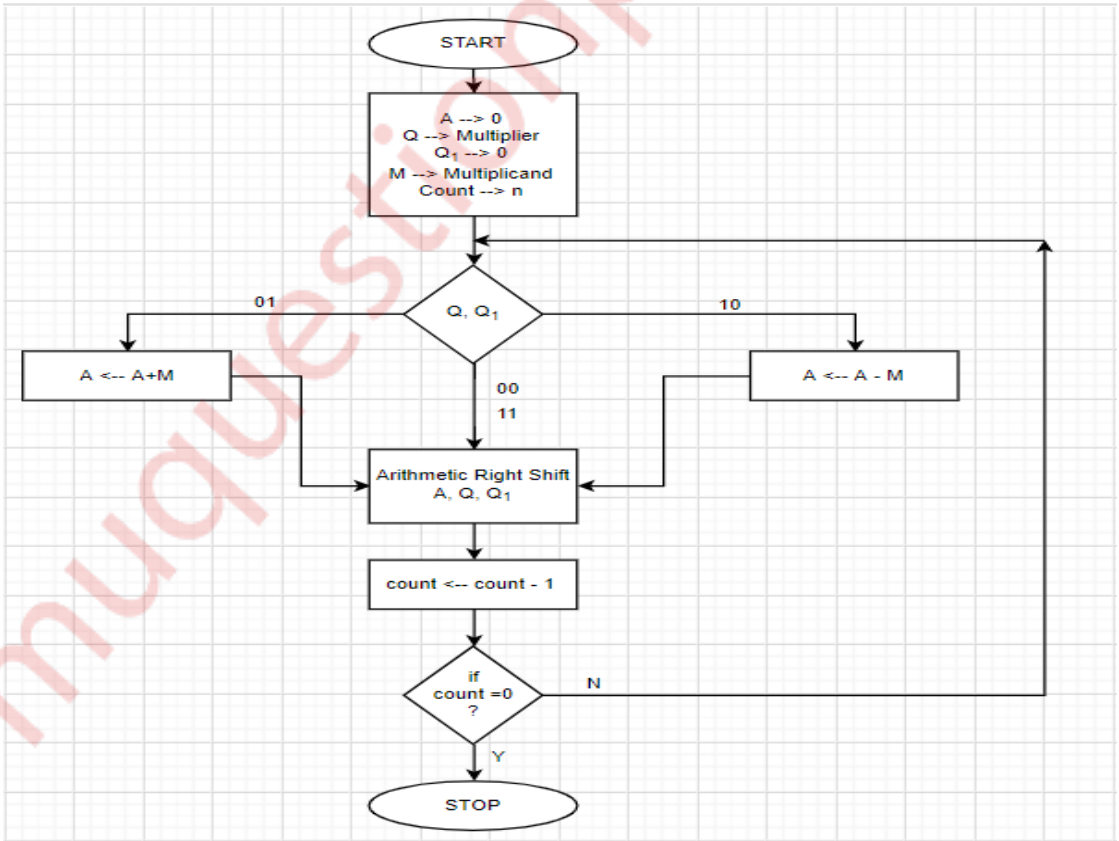
- Multi-core processors are widely used across many application domains, including general-purpose, embedded, network, digital signal processing (DSP), and graphics (GPU).
- The improvement in performance gained by the use of a multi-core processor depends very much on the software algorithms used and their implementation. In particular, possible gains are limited by the fraction of the software that can run in parallel simultaneously on multiple cores; this effect is described by Amdahl's law.
- In the best case, so-called embarrassingly parallel problems may realize speedup factors near the number of cores, or even more if the problem is split up enough to fit within each core's cache(s), avoiding use of much slower main-system memory. Most applications, however, are not accelerated so much unless programmers invest a prohibitive amount of effort in re-factoring the whole problem.
- The parallelization of software is a significant ongoing topic of research. Cointegration of multiprocessor applications provides flexibility in network architecture design. Adaptability within parallel models is an additional feature of systems utilizing these protocols.

---

b) Explain Booth's Multiplication algorithm and perform $(17)_{10}*(-5)_{10}$ (10 M)

Ans:

Booth's principle states that "The value of series of 1's of binary can be given as the weight of the bit preceding the series minus the weight of the last bit in the series.**"**



Given:

$Q = -5 = (11011)_2$

$M = 17 = (10001)_2$

$-M = (01111)_2$

AC	Q	Q_{-1}	M	count
00000	11011	0	10001	5
+01111				
01111	11011	0		
00111	11101	1		4
00011	11110	1		3
+10001				
10100	11110	1		
11010	01111	0		2
+01111				
01001	01111	0		
10100	10111	1		1
11101	10111	1		0

$-(0001010101)_2 = -(85)_2$

Q.6) Write Short Notes on (20 M)

a) Data Transfer Techniques:

Ans:

Programmed I/O

- In the programmed I/O method of interfacing. CPU has direct control over I/O.
- The processor checks the status of the devices and issues read or write commands and then transfer data. During the data transfer. CPU waits for I/O module to complete operation and hence this system wastes the CPU time.
- The sequence of operations to be carried out in programmed I/O operation are:
  - CPU requests for I/O operation.
  - I/O module performs the said operation.
  - I/O module update the status bits.

- CPU checks these status bits periodically. Neither the I/O module can inform CPU directly nor can I/O module interrupt CPU.
- CPU may wait for the operation to complete or may continue the operation later.

Interrupt driven I/O

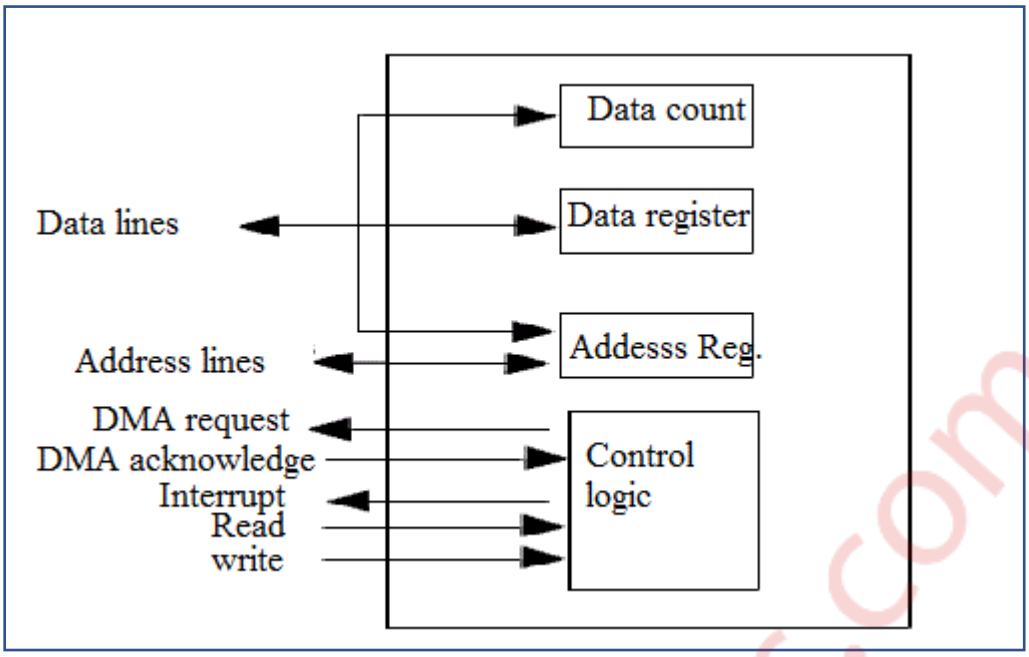
- Interrupt driven I/O overcomes the disadvantage of programmed I/O i.e. the CPU waiting for I/O device.
- This disadvantage is overcome by CPU not repeatedly checking for the device being ready or not instead the I/O module interrupts when ready.
- The sequence of operations for interrupt Driven I/O is as below:
  - CPU issues the read command to I/O device.
  - I/O module gets data from peripheral while CPU does other work.
  - Once the I/O module completes the data transfer from I/O device, it interrupts CPU.
  - On getting the interrupt, CPU requests data from the I/O module.
  - I/O module transfers the data to CPU.
- The interrupt driven I/O mechanism for transferring a block of data.
- After issuing the read command the CPU performs its work, but checks for the interrupt after every instruction cycle.
- When CPU gets an interrupt, it performs the following operation in sequence:
  - Save context i.e. the contents of the registers on the stack
  - Processes interrupt by executing the corresponding ISR
  - Restore the register context from the stack.

Transferring a word of data

- CPU issues a 'READ' command to I/O device and then switches to some other program. CPU may be working on different programs.
- Once the I/O device is ready with the data in its data register. I/O device signals an interrupt to the CPU.
- When then interrupt from I/O device occurs, it suspends execution of the current program, reads from the port and then resumes execution of the suspended program.

Data Transfer Modes

- DMA stands for Direct Memory Access. The I/O can directly access the memory using this method.
- Interrupt driven and programmed I/O require active operation of the CPU. Hence transfer rate is limited and CPU is also busy doing the transfer operation. DMA is the solution to this problem.
- DMA controller takes over the control of the bus form CPU for I/O transfer.



- The address register is used to hold the address of the memory location from which the data is to be transferred. There may be multiple address registers to hold multiple addresses.
- The address may be incremented or decremented after every transfer based on mode of operation.
- The data count register is used to keep a track of the number of bytes to be transferred. The counter register is decremented after every transfer.
- The data register is used in a special case i.e. when the transfer of a block is to be done from one memory location to another memory location.
- The DMA controller is initially programmed by the CPU, for the count of bytes to bee transferred address of the memory block for the data to be transferred etc.
- During this programming DMAC, the read and write lines work as inputs for DMAC.
- Once the DMAC takes the control of the system bus i.e. transfers the data between the memory and I/O device, these read and write signals work as output signals.
- They are used to tell the memory that the DMAC wants to read or write from the memory according to the operation being data transfer from memory to I/O or from I/O to memory.

DMA Transfer Modes:

- **Single transfer mode:** In this, the device is programmed to make one byte transfer only after getting the control of system bus.
- After transferring one byte the control of the bus will be returned back to the CPU.
- The word count will be decremented and the address decremented or incremented following each transfer.

- **Block transfer Mode:** In this, the device is activated by DREQ or software request and continues making transfers during the service until a Terminal Count, or an external End of Process is encountered.
- The advantage is that the I/O device gets the transfer of data a very faster speed.
- **Demand Transfer Mode:** In this, the devices continues making transfer until a Terminal Count or external EOP is encountered, or until DREQ goes inactive.
- Thus, transfer may continue until the I/O device has exhausted its data handling capacity.
- **Hidden Transfer Mode:** In this, the DMA controller takes over the charge on the system bus and transfers data when processor does not needs system bus.
- The processor does not even realize of this transfer being taking place.
- Hence these transfer are hidden from the processor.

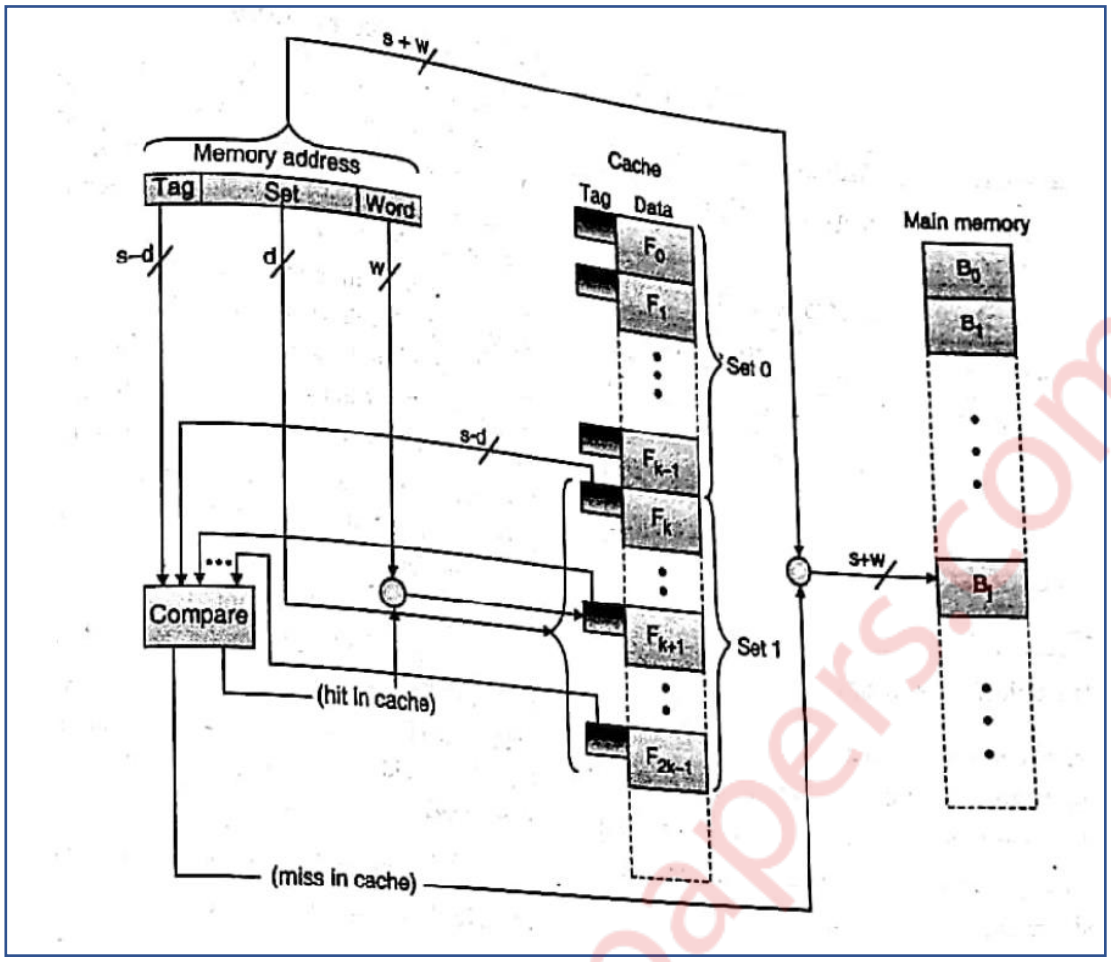
---

b) Set Associative Cache Mapping (10 M)

Ans:

Set Associative Cache :

- In set associative cache mapping, cache is divided into a number of sets. Each set contains a number of lines.
- A given block maps to any line in a given set (i mod j), where i is the line number of the main memory to be mapped and j is the total number of sets in cache memory.
- This form of mapping is an enhanced form of direct mapping where the drawbacks of direct mapping are removed.
- Set associative 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 Then a block in memory can map to any one of the lines of a specific set.
- Set-associative mapping allows that each word that is present 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 techniques.
- For example, if there are 2 lines per set, it is called as 2 way associative mapping i.e. given block can be In one of 2 lines in only one set.

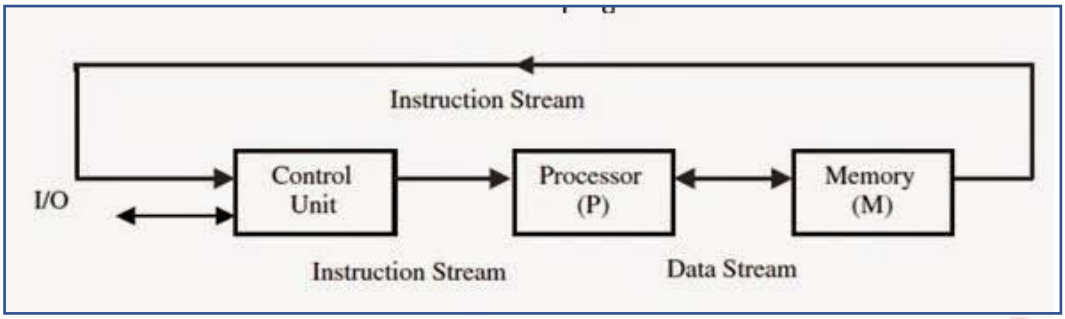


---

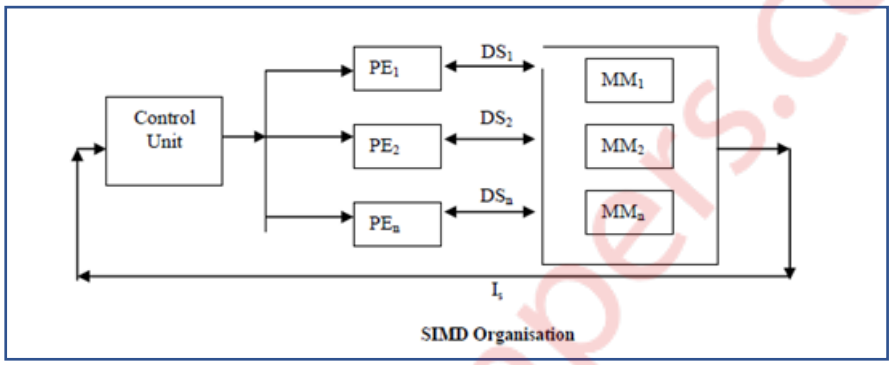
c) Flynn's Classification. (10 M)

Ans:

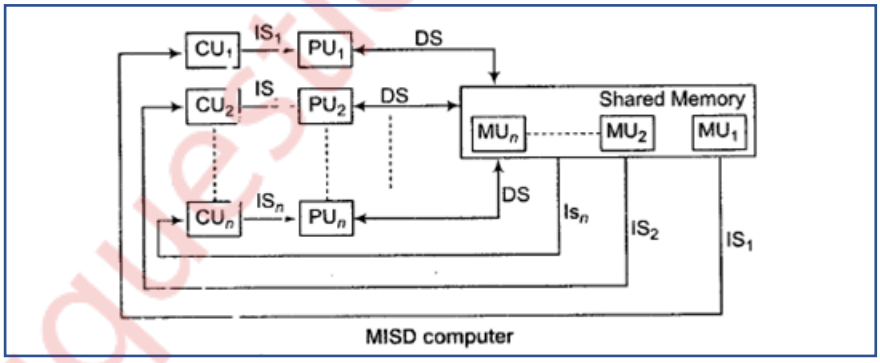
- A method introduced by Flynn, for classification of parallel processors is most common. This classification is based on the number of instruction Streams and Data Streams in the system. There may be single or multiple streams of each of these. Hence accordingly, Flynn classified the parallel processing into four categories:
  - Single instruction Single Data (SISD)
  - Single instruction Multiple Data (SIMD)
  - Multiple Instruction Single Data (MISD)
  - Multiple Instruction Multiple Data (MIMD)
- **SISD:** In this case there is a single processor that executes one instruction at a time on single data stored in the memory.
- In fact, this type of processing can be said to be unit processing, hence unit processors fall into this category.
- The processing Element accesses the data from the memory and performs the operation on this data as per the signal given by control unit.



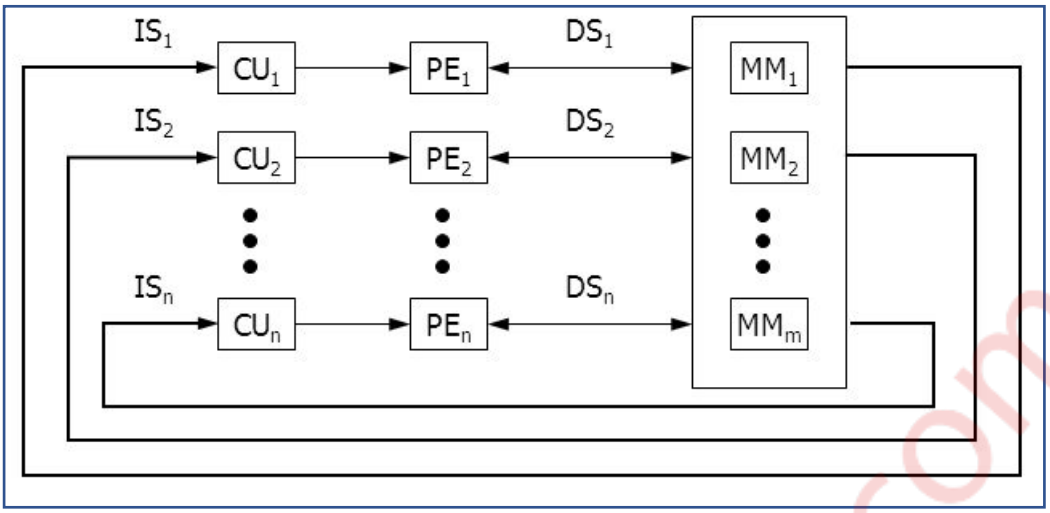
- **SIMD:** In this case the same instruction is given to multiple processing elements, but different data.
- This kind of system is mainly used when many data have to be operated with same operation.



- **MISD:** In case of MISD, there are multiple instruction streams and hence multiple control units to decode these instructions.
- Each control unit takes a different instructions from the different memory module in same memory.
- The data stream is single. In this case the data is taken by the first processing element.



- **MIMD:** This is a complete parallel processing example. Here each processing element is having a different set of data and different instructions.
- Examples of this kind of systems are SMPs, clusters and NUMA.



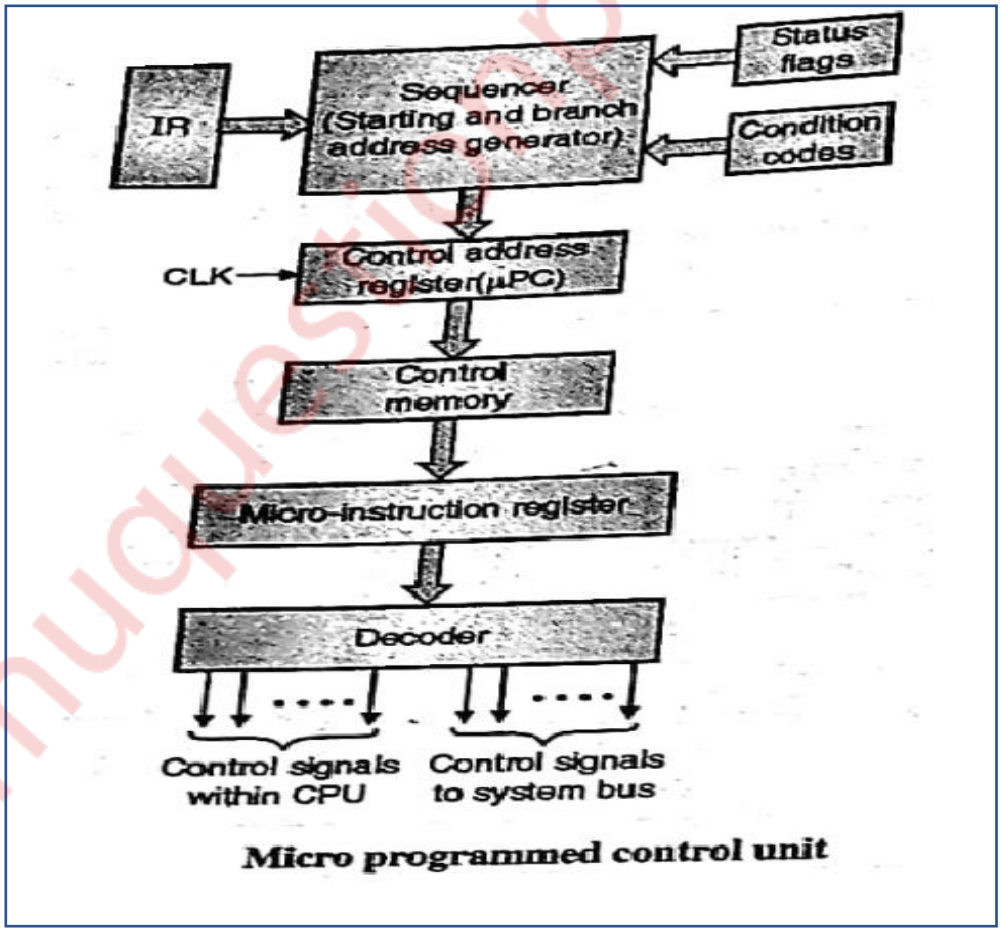
---

d) Control Unit of processor. (10 M)

Ans:

Micro Programmed Control unit:

- Micro programmed control unit generates control signals based on the microinstructions stored in a special memory called as the control memory.



Micro programmed control unit

- Each instruction points to a corresponding location in the control memory that loads the control signals in the control register.
- The control register is then read by a sequencing logic that issues the control signals in a proper sequence.
- The implementation of the micro programmed.
- The instruction register (IR), status flag and condition codes are read by the sequencer that generates the address of the control memory location for the corresponding instruction in the IR.
- This address is stored in the control address register that selects one of the locations in the control memory having the corresponding control signals.
- These control signals are given to the microinstruction register, decoded and then given to the individual components of the processor and the external devices.

Wilkie's Microprogrammed(Hard Wired) Control Unit:

- First working model of micro-programmed control unit was proposed by Wilkies in 1952. In above design, a microinstruction has two major components:
    - Control field
    - Address field
- The control memory access register can be loaded from an external source as well as from the address field of microinstructions. A machine instruction typically provides the starting address of a micro-program in control memory.
- On the basis of starting address from instruction register, decoder activates one of the eight output lines.
- This activated lines, in turn generates control signals and the address of the next microinstruction to be executed.
- This address is once again fed to the CMAR resulting in activation of another control line and address field.
- This process is repeated till the execution of the instruction is achieved.

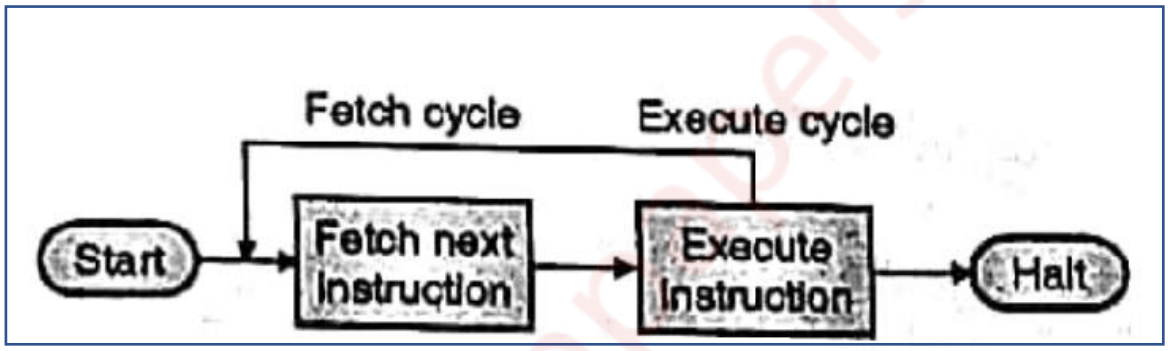
COMPUTER ORGANISATION AND ARCHITECTURE (MAY 2019)

Q.1) Write the following **(20 M)**

a) Explain Instruction and instruction cycle. **(5 M)**

Ans:

- The instruction cycle is a representation of the states that the computer or the microprocessor performs when executing an instruction.
- The instruction cycle comprises of two main steps to be followed to execute the instruction namely the fetch operation in the fetch cycle and the execution operation during the execute cycle.



- It comprises of the fetch cycle and executes cycle in a loop to execute huge number of instructions, until it reaches the halt instruction.
- The fetch cycle comprises of the following operations:
    - Program centre holds address of next instruction to fetch; hence the CPU fetches instruction from memory pointed to by PC. This is done by providing the value of PC to the MAR and giving the Read control signal to the memory. On this memory provides the value in the given address.
    - The PC values has to be incremented to point to the next instruction.
    - The instruction is loaded into Instruction Register from the MBR.
    - Finally the processor interprets or decodes the instruction. The processor performs required operations in execute cycle.
- In the execute cycle the operation asked to be performed by the instruction is done. It may comprise of one or more of the following:
    - Transfer of data between processor and memory or between processor or IO module.
    - Processing of data like some arithmetic or logical operation of data.]
    - Change the sequence of operation i.e. branching instructions.

b) Differentiate between Memory based IO and IO mapped IO. (5 M)

Ans:

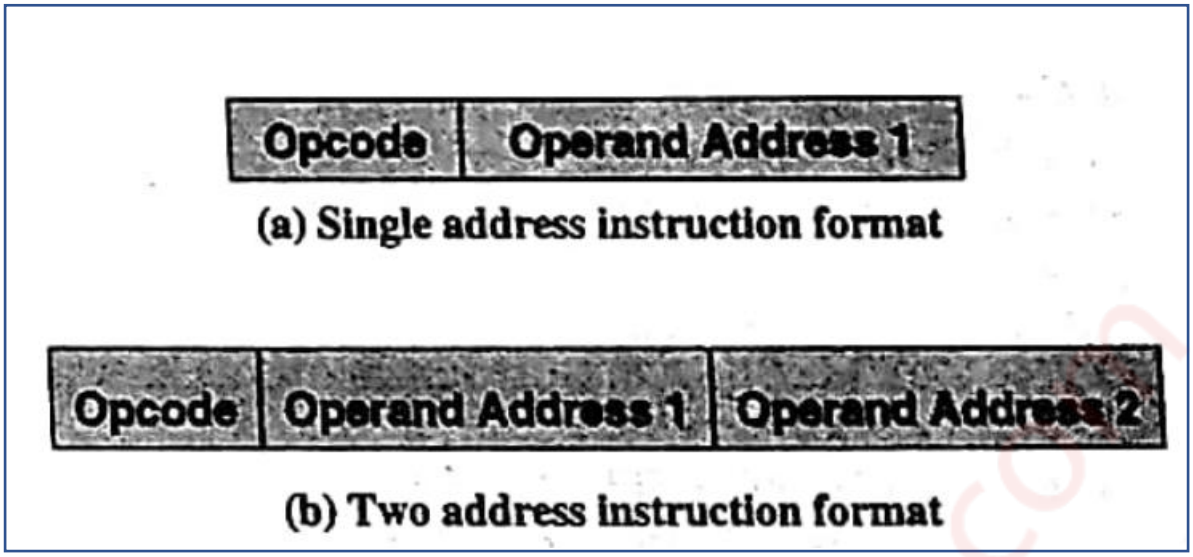
Memory Mapped IO	IO Mapped IO
A method to perform IO operations between the CPU and peripheral devices in a computer that uses one address space for memory and IO devices.	A method to perform IO operations between the CPU and peripheral devices in a computer that uses two separate address spaces for memory and IO device.
Uses the same address space for both memory and IO devices.	Uses two separate address space for memory and IO device
As the memory mapped IO uses one address space for both IO and memory, the available addresses for memory are minimum.	All the addresses can be used by the memory.
Uses the same instructions for both IO and memory operations.	Uses separate instructions for read and write operations in IO and memory.
Less efficient	More Efficient.

---

c) Give different instruction formats. (5 M)

Ans:

- Input devices are required to give the instructions and data to the system. The output devices are used to give the output devices.
- The instructions and data given by the input device are to be stored, and for storage we require memory.
- Elements of Single, two and three address instructions are as follows:
    - Operation code is that part of the instruction which gives the code for the operation to be performed.
    - Source operand reference or address 1, gives the reference of the data on which the operation is to be performed. This address could be a register, memory or an input device.
    - Source operand reference or address 2, gives the reference of the second data on which the operation is to be performed. This address could again be a register, memory or an input device.
    - Result operand reference gives the reference where the result after performing operation is to be stored.
    - An instruction may have only one address with the other two fixed, or may have two addresses with one of the source operand address as the result operand address. Hence the instruction can have one, two or three addresses.



d) Explain memory interleaving Techniques. (5 M)

Ans:

- Interleaved memory implements the concept of accessing more words in single memory access cycle. Memory can be partitioned into N separate memory modules. Thus N accesses can be carried out to the memory simultaneously.
- Once presented with a memory address, each memory module returns one word per cycle. It is possible to present different addresses to different memory modules so that parallel access to multiple words can be done simultaneously or in pipelined fashion.
- The maximum processor bandwidth in interleaved memory can be equal to the number of modules i.e. N words per cycle.
- To achieve the address interleaving consecutive addresses are distributed among the N interleaved modules.
- Interleaving spreads contiguous memory locations across m modules horizontally. This implies that the low-order a bits of the memory address are used to identify the memory module.
- The high order b bits are used to address a word inside a module. Same word address is applied to all memory modules simultaneously. A module address decoder is used to distribute module addresses.\access of the m modules can be overlapped in a pipelined fashion. For this purpose, the memory cycle is subdivided into m sub cycles.

e) Explain superscalar architecture. (5 M)

Ans:

- Superscalar processor are those processors that have multiple execution units.

- Hence these processors can execute the independent instructions simultaneously and hence with the help of this parallelism it increases the speed of the processor.
- It has been that the number of independent consecutive instructions 2 to 5. Hence the instruction issue degree in a superscalar processor is restricted from 2 to 5.

Pipelining in Superscalar Processor:

- The pipelining is the most important representation of demonstrating the speed increase by the superscalar feature of processor.
- Hence to implement multiple operations simultaneously, we need to have multiple execution units to execute each instruction independently.
- The ID and rename unit, decodes the instruction and then by the use of register renaming avoids instruction dependency. The instruction window takes the decoded instructions and based on some pair ability rules, issues them to the respective execution units.
- The instructions once executed move to the Retire and write back unit, wherein the instructions retire and the result is written back to the corresponding destination.

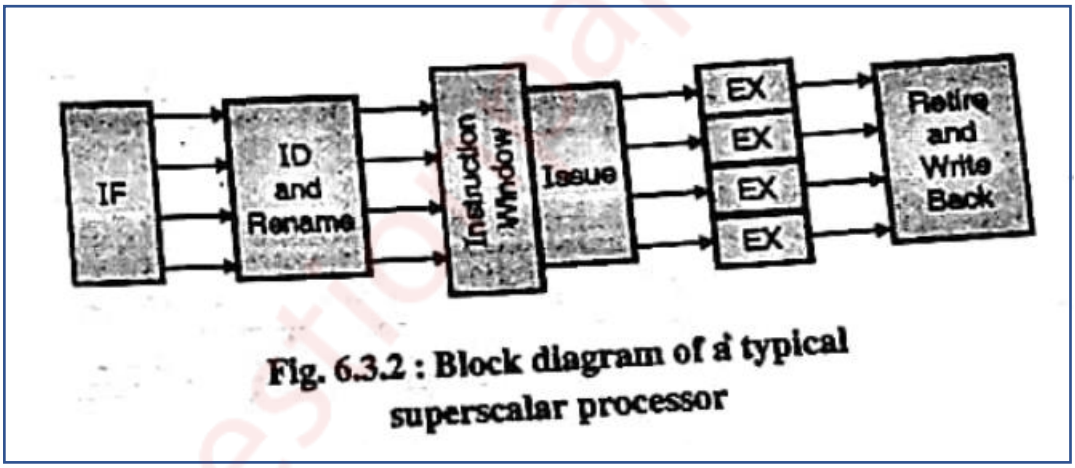
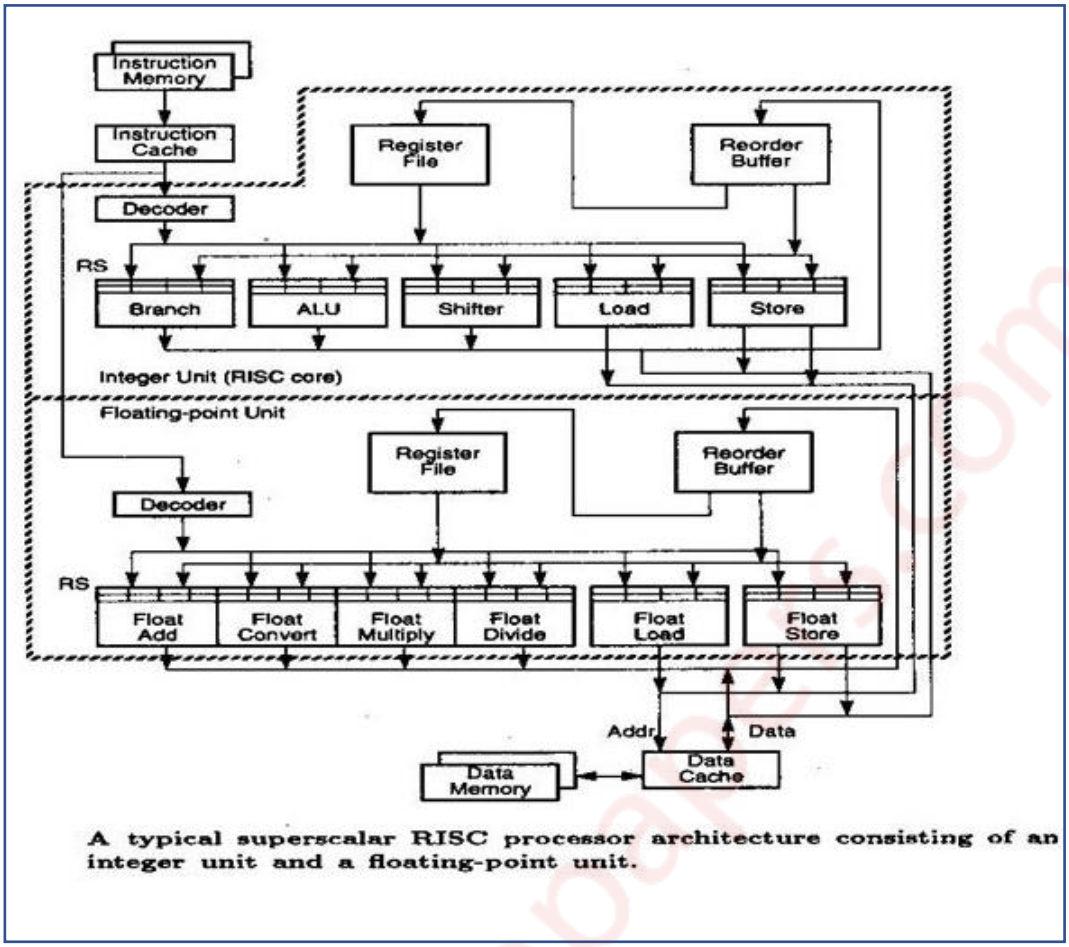


Fig. 6.3.2 : Block diagram of a typical superscalar processor

- A RISC or CISC processors execute one instruction per cycle. Their performance can be improved with superscalar architecture:
  - Multiple instruction pipelines are used.
  - Multiple instructions are issued for execution per cycle.
  - Multiple results are generated per cycles.

- Superscalar processors can exploit more instruction level parallelism in user program.



A typical superscalar RISC processor architecture consisting of an integer unit and a floating-point unit.

Q.2)

a) Explain Branch Prediction Logic and Delayed Branch. (10 M)

Ans:

- Performance gain through pipelining can be reduced by the presence of program transfer instructions (such as JMP, CALL, RET and conditional jumps).
- They change the sequence causing all the instructions that entered the pipeline after program transfer instruction invalid.
- Suppose instruction I3 is a conditional jump to I50 at some other address (target address), then the instructions that entered after I3 is invalid and new sequence beginning with I50 need to be loaded in.
- This causes bubbles in pipeline, where no work is done as the pipeline stages are reloaded.
- To avoid this problem, the Pentium uses a scheme called Dynamic Branch Prediction.
- In this scheme, a prediction is made concerning the branch instruction currently in pipeline.
- Prediction will be either taken or not taken.

- If the prediction turns out to be true, the pipeline will not be flushed and no clock cycles will be lost. If the prediction turns out to be false, the pipeline is flushed and started over with the correct instruction.
- It results in a 3 cycle penalty if the branch is executed in the u-pipeline and 4 cycle penalty in v-pipeline.
- It is implemented using a 4-way set associative cache with 256 entries. This is referred to as the Branch Target Buffer (BTB).
- The directory entry for each line contains the following information:
- Valid Bit : Indicates whether or not the entry is in use.
- History Bits: track how often the branch has been taken.
- Source memory address that the branch instruction was fetched from (address of I3).
- If its directory entry is valid, the target address of the branch is stored in corresponding data entry in BTB.

Delayed Branch:

- Compiler detects branch instruction and rearranges the machine language code sequence by inserting useful instructions and rearranges the code sequence to reduce the delays incurred by Branch Instruction.

---

b) A program having 10 instructions (without Branch and Call Instructions) is executed on non-pipeline and pipeline processors. All instructions are of same length and having 4 pipeline stages and time required to each stage is 1nsecc.

I. Calculate time required to execute the program or Non-pipeline and Pipeline processor.

II. Calculate Speedup. (10 M)

Ans:

I. Given: n = 10 instructions, K = 4, t = 1nsec

Execution time pipelined = (4 + 100 - 1) * t

= (4 + 90) * 1

= 94 nsec.

Execution time unpipelined = (K * t) n

= (4 * 1) 1

= 4 nsec.

II. Speedup = 4/94 **= 0.043 times.**

Q.3)

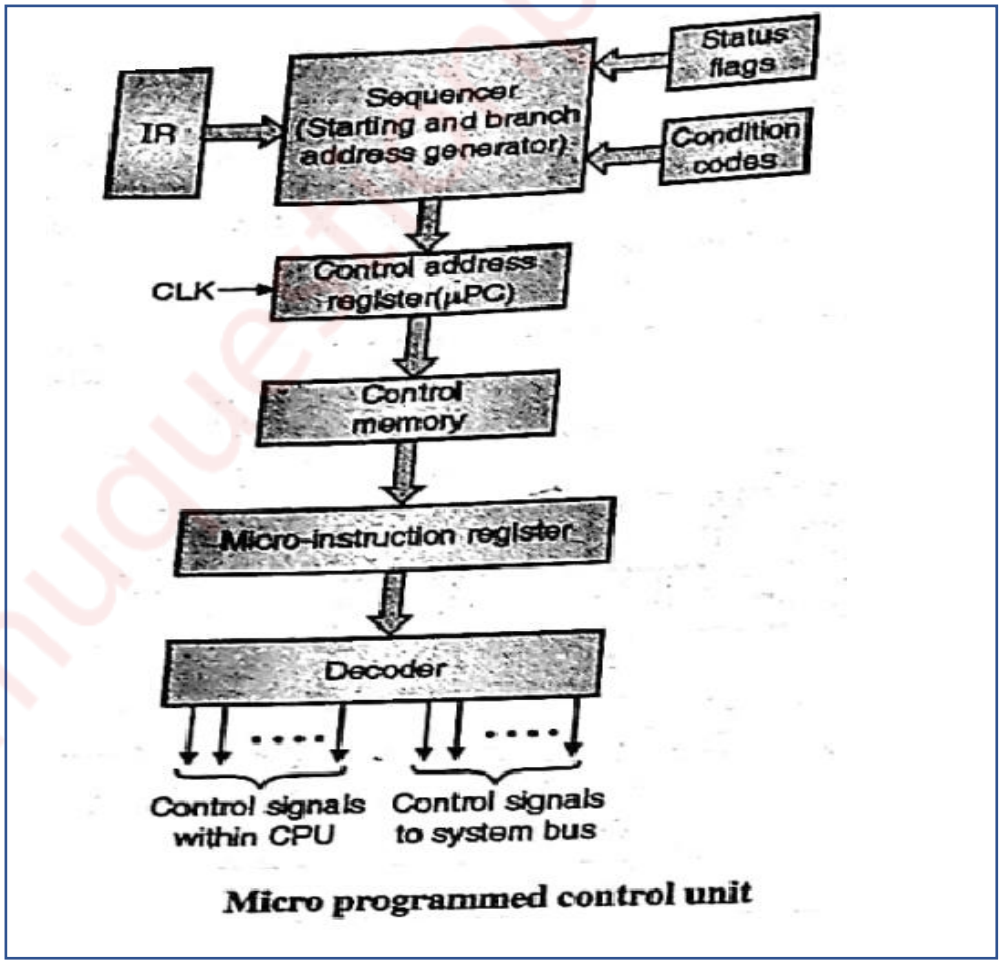
a) Explain different technique for design of control unit of computer.

(10 M)

Ans:

Micro Programmed Control unit:

- Micro programmed control unit generates control signals based on the microinstructions stored in a special memory called as the control memory.
- Each instruction points to a corresponding location in the control memory that loads the control signals in the control register.
- The control register is then read by a sequencing logic that issues the control signals in a proper sequence.
- The implementation of the micro programmed.
- The instruction register (IR), status flag and condition codes are read by the sequencer that generates the address of the control memory location for the corresponding instruction in the IR.
- This address is stored in the control address register that selects one of the locations in the control memory having the corresponding control signals.
- These control signals are given to the microinstruction register, decoded and then given to the individual components of the processor and the external devices.



Micro programmed control unit

Wilkie's Microprogrammed(Hard Wired) Control Unit:

- First working model of micro-programmed control unit was proposed by Wilkies in 1952. In above design, a microinstruction has two major components:
  - Control field
  - Address field
- The control memory access register can be loaded from an external source as well as from the address field of microinstructions. A machine instruction typically provides the starting address of a micro-program in control memory.
- On the basis of starting address from instruction register, decoder activates one of the eight output lines.
- This activated lines, in turn generates control signals and the address of the next microinstruction to be executed.
- This address is once again fed to the CMAR resulting in activation of another control line and address field.
- This process is repeated till the execution of the instruction is achieved.

---

b) What is micro program? Write microprogram for following operations

I. ADD R1, M, Register R1 and Memory location M are added and result store at Register R1.

II. MUL R1, R2 Register R1 and Register R2 are multiplied and result store at Register R1. (10 M)

Ans:

- Microprogramming is a process of writing microcode for a microprocessor. Microcode is low-level code that defines how a microprocessor should function when it executes machine-language instructions.
- Typically, one machine language instruction translates into several microcode instruction, on some computers, the microcode is stored in ROM and cannot be modified.
- **Micro Program to add R1, M.**

T-state	Operation	Microinstructions
T 1	PC → MAR	PCout, Marin, Read, Clear y, set Cin, Add, Zinn
T 2	M → MBR PC ← PC + 1	Zout, PCin, Wait for memory fetch cycle
T 3	MBR → IR	MBRout, IRin
T 4	R1 → x	R1out, Xin, CLRC
T 5	M → ALU	Mout, ADD, Zin
T 6	Z → R1	Zout, R1in

T 7	Check for intr	Assumption enabled intr pending, CLRX, SETC, Spout, SUB, Zin
T 8	SP ← SP – 1	Zout, Spin, MARin
T 9	PC → MDR	PCout, MDRin, WRITE
T 10	MDR → [SP]	Wait for Mem access
T11	PC ← IS Raddr	PCin IS Raddr out.

- **Micro Program to MUL R1, R2**

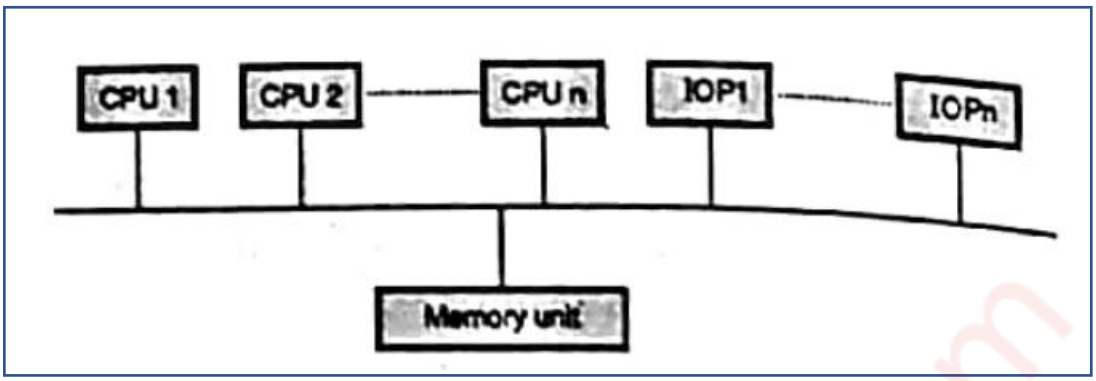
T-state	Operation	Microinstructions
T 1	PC → MAR	PCout, Marin, Read, Clear y, set Cin, Add, Zinn
T 2	M → MBR PC ← PC + 1	Zout, PCin, Wait for memory fetch cycle
T 3	MBR → IR	MBRout, IRin
T 4	R1 → x	R1out, Xin, CLRC
T 5	R2 → ALU	R2out, MUL, Zin
T 6	Z → R1	Zout, R1in
T 7	Check for intr	Assumption enabled intr pending, CLRX, SETC, Spout, SUB, Zin
T 8	SP ← SP – 1	Zout, Spin, MARin
T 9	PC → MDR	PCout, MDRin, WRITE
T 10	MDR → [SP]	Wait for Mem access
T11	PC ← IS Raddr	PCin IS Raddr out.

Q.4)

a) Explain Bus Contention and different method to resolve it. (10 M)

Ans:

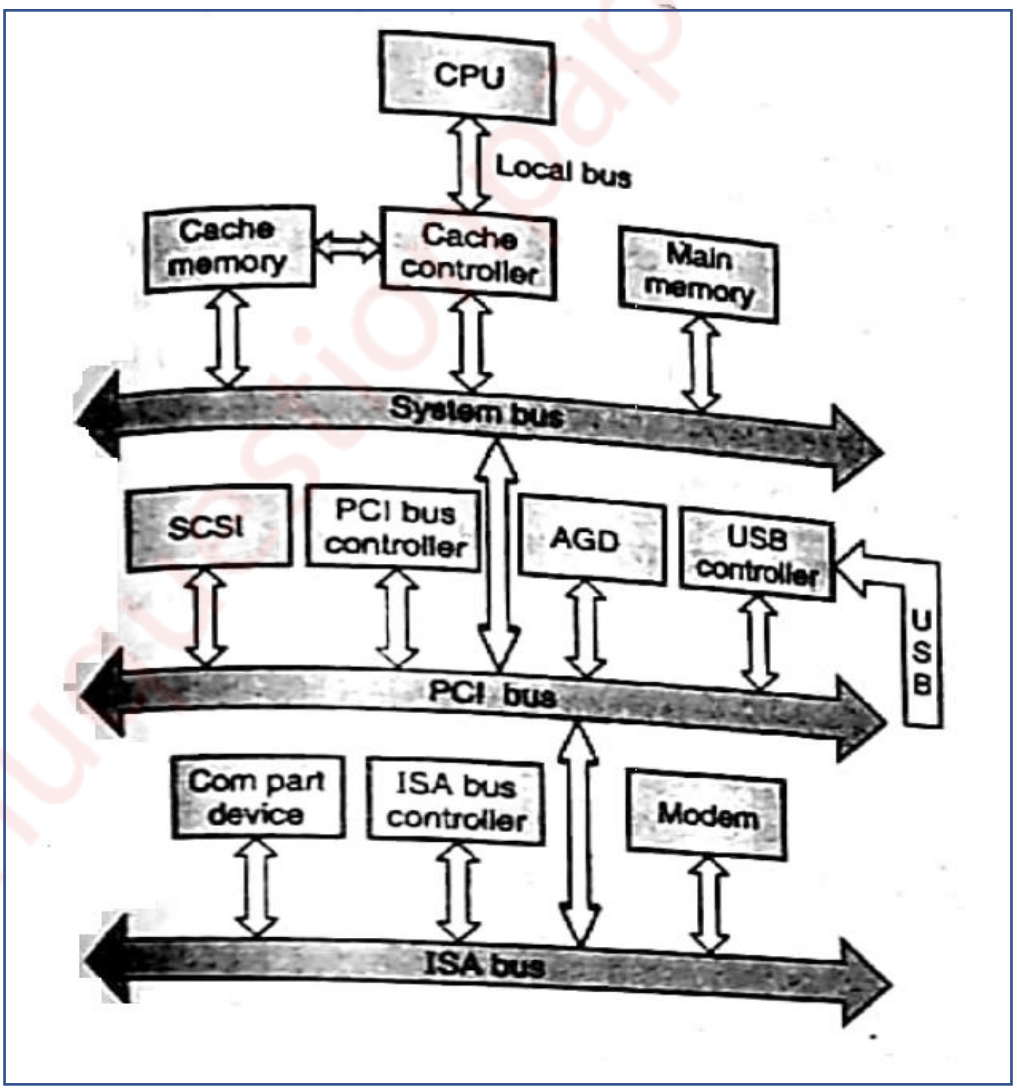
- In a bus system, processors, memory modules and peripheral devices are attached to the bus. The bus can handle only one transaction at a time between a master and slave. In case of multiple requests, the bus arbitration logic must be able to allocate or deallocate and it should service request at a time.
- Thus, such a bus is a time sharing or contention bus among multiple functional modules. As only one transfer can take place at any time on the bus, the overall performance of the system is limited by the bandwidth of the bus.
- When number of processors contending to acquire a bus exceeds the limit then a single bus architecture may become a major bottleneck. This may cause a serious delay in servicing a transaction.



- Aggregate data transfer demand should never exceed the capacity of the bus. This problem can be countered to some extent by increasing the data rate of the bus and by using a wider bus.
- Method of avoiding contention is multiple bus hierarchy.

Multiple-Bus Architecture:

- If a greater number of devices are connected to the bus, performance will suffer due to following reasons:



- In general, the more devices attached to the bus, the greater will be propagation delay.
- The bus may become a bottleneck as the aggregate data transfer demand approaches the capacity of the bus.
- This problem can be countered to some extent by increasing the data rate the bus can carry and by using wider buses.
- Most computer systems enjoy the use of multiple buses. These buses are arranged in a hierarchy.

---

b) Explain Different data transfer Technique. (10 M)

Ans:

Programmed I/O

- In the programmed I/O method of interfacing. CPU has direct control over I/O.
- The processor checks the status of the devices and issues read or write commands and then transfer data. During the data transfer. CPU waits for I/O module to complete operation and hence this system wastes the CPU time.
- The sequence of operations to be carried out in programmed I/O operation are:
  - CPU requests for I/O operation.
  - I/O module performs the said operation.
  - I/O module update the status bits.
  - CPU checks these status bits periodically. Neither the I/O module can inform CPU directly nor can I/O module interrupt CPU.
  - CPU may wait for the operation to complete or may continue the operation later.

Interrupt driven I/O

- Interrupt driven I/O overcomes the disadvantage of programmed I/O i.e. the CPU waiting for I/O device.
- This disadvantage is overcome by CPU not repeatedly checking for the device being ready or not instead the I/O module interrupts when ready.
- The sequence of operations for interrupt Driven I/O is as below:
  - CPU issues the read command to I/O device.
  - I/O module gets data from peripheral while CPU does other work.
  - Once the I/O module completes the data transfer from I/O device, it interrupts CPU.
  - On getting the interrupt, CPU requests data from the I/O module.
  - I/O module transfers the data to CPU.
- The interrupt driven I/O mechanism for transferring a block of data.
- After issuing the read command the CPU performs its work, but checks for the interrupt after every instruction cycle.
- When CPU gets an interrupt, it performs the following operation in sequence:

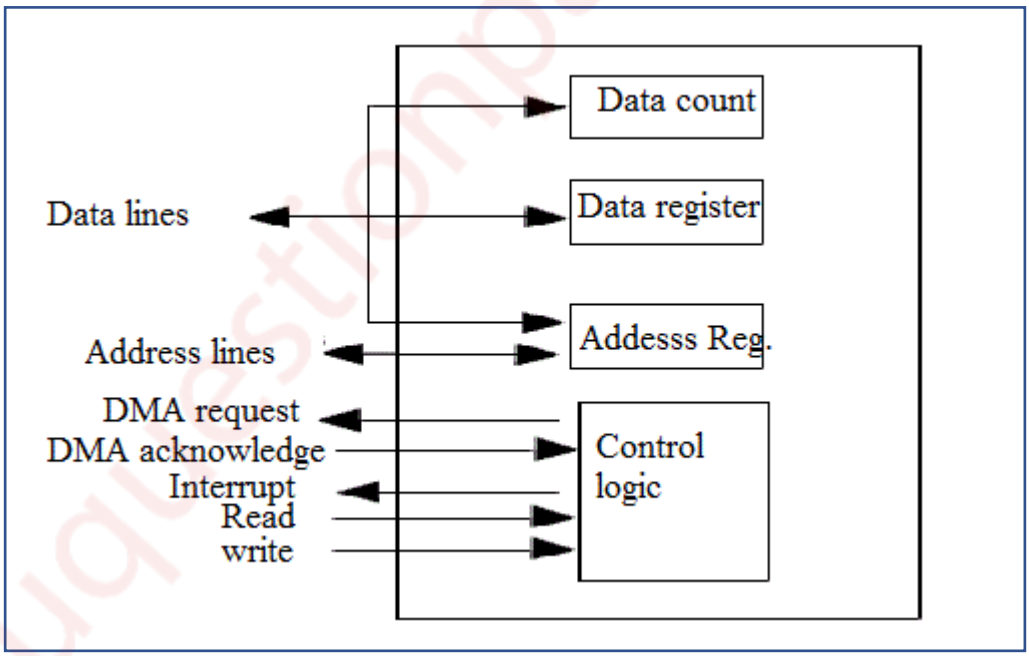
- Save context i.e. the contents of the registers on the stack
- Processes interrupt by executing the corresponding ISR
- Restore the register context from the stack.

Transferring a word of data

- CPU issues a 'READ' command to I/O device and then switches to some other program. CPU may be working on different programs.
- Once the I/O device is ready with the data in its data register. I/O device signals an interrupt to the CPU.
- When then interrupt from I/O device occurs, it suspends execution of the current program, reads from the port and then resumes execution of the suspended program.

Data Transfer Modes

- DMA stands for Direct Memory Access. The I/O can directly access the memory using this method.
- Interrupt driven and programmed I/O require active operation of the CPU. Hence transfer rate is limited and CPU is also busy doing the transfer operation. DMA is the solution to this problem.
- DMA controller takes over the control of the bus form CPU for I/O transfer.



- The address register is used to hold the address of the memory location from which the data is to be transferred. There may be multiple address registers to hold multiple addresses.
- The address may be incremented or decremented after every transfer based on mode of operation.
- The data count register is used to keep a track of the number of bytes to be transferred. The counter register is decremented after every transfer.

- The data register is used in a special case i.e. when the transfer of a block is to be done from one memory location to another memory location.
- The DMA controller is initially programmed by the CPU, for the count of bytes to bee transferred address of the memory block for the data to be transferred etc.
- During this programming DMAC, the read and write lines work as inputs for DMAC.
- Once the DMAC takes the control of the system bus i.e. transfers the data between the memory and I/O device, these read and write signals work as output signals.
- They are used to tell the memory that the DMAC wants to read or write from the memory according to the operation being data transfer from memory to I/O or from I/O to memory.

DMA Transfer Modes:

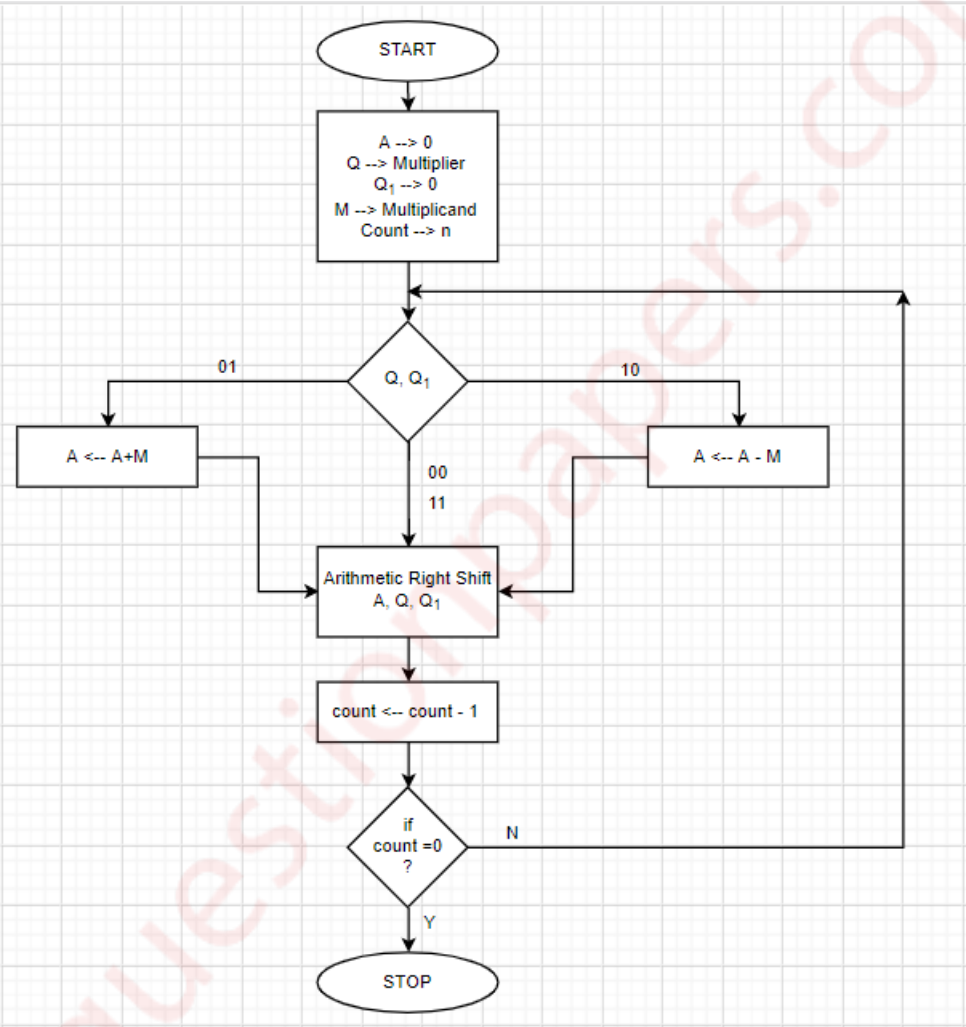
- **Single transfer mode:** In this, the device is programmed to make one byte transfer only after getting the control of system bus.
- After transferring one byte the control of the bus will be returned back to the CPU.
- The word count will be decremented and the address decremented or incremented following each transfer.
- **Block transfer Mode:** In this, the device is activated by DREQ or software request and continues making transfers during the service until a Terminal Count, or an external End of Process is encountered.
- The advantage is that the I/O device gets the transfer of data a very faster speed.
- **Demand Transfer Mode:** In this, the devices continues making transfer until a Terminal Count or external EOP is encountered, or until DREQ goes inactive.
- Thus, transfer may continue until the I/O device has exhausted its data handling capacity.
- **Hidden Transfer Mode:** In this, the DMA controller takes over the charge on the system bus and transfers data when processor does not needs system bus.
- The processor does not even realize of this transfer being taking place.
- Hence these transfer are hidden from the processor.

Q.5)

a) Explain Booth's Multiplication algorithm and Perform $(17)_{10} * (5)_{10}$ (10 M)

Ans:

Booth's principle states that "The value of series of 1's of binary can be given as the weight of the bit preceding the series minus the weight of the last bit in the series.**"**



Given: m = 17 = $(010001)_2$

Q = 5 = $(000101)_2$

-m = $(111010)_2$

A	Q	Q_{-1}	m
000000	000101	0	010001
+111010			
111010	000101	0	

111101	000010	1
+010001		
001110	000010	1
000111	000001	0
+111010		
100001	000001	0
010000	100000	1
+111010	100000	1
011010	100000	1
001101	010000	0
000110	101000	0
000011	010100	0
000001	101010	0
000000	110101	0
000000	101010	1

$(1010101)_2 = (85)_{10}$

b) Consider a cache memory of 16 words. Each block consists of 4 words. Size of the main memory is 256 bytes. Draw associative mapping and calculate TAG, and word size. (10 M)

Ans:

Given:

Cache Memory = 16 words

Block consist of 4 words

Main memory = 256 bytes

Main memory = 256 * 16 words = $2^8 * 2^4 = 2^{12}$

TAG field = 2^{12} = 12 bits consist of both set field and TAG field.

SET + TAG = 12

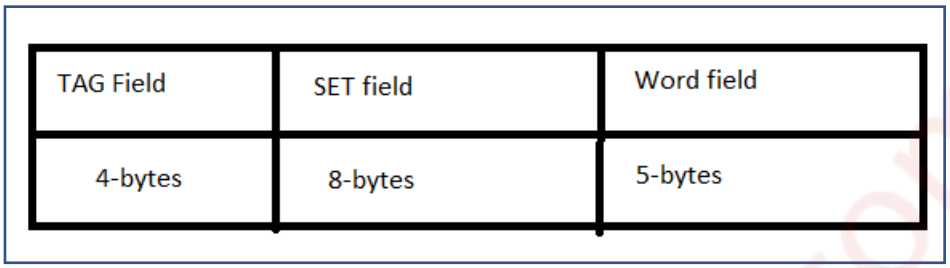
4 + TAG = 12

TAG = 12 – 4

TAG = 8 bytes.

Word size = As there are 8 blocks and each block consist of 4 words,

Hence 8 * 4 = 32 = 2^5



Q.6)

a) Explain Different type of pipeline hazards. (10 M)

Ans:

- Instruction hazards occur when instructions read or write registers that are used by other instructions. The type of conflicts are divided into three categories:
  - Structural Hazards (resource conflicts)
  - Data Hazards (Data dependency conflicts)
  - Branch difficulties (Control Hazards)
- **Structural hazards:** these hazards are caused by access to memory by two instructions at the same time. These conflicts can be slightly resolved by using separate instruction and data memories.
- It occurs when the processor's hardware is not capable of executing all the instructions in the pipeline simultaneously.
- Structural Hazards within a single pipeline are rare on modern processors because the instructions Set Architecture is designed to support pipelining.
- **Data Hazards:** This hazard arises when an instruction depends on the result of a previous instruction, but this result is not available.
- These are divided into four categories.
  - RAW - Hazard
  - RAR - Hazard
  - WAW - Hazard
  - WAR – Hazard
- **RAR Hazard:** RAR Hazard occurs when two instructions both from the same register. This hazard does not came from problem for the processor because reading a register does not change the register's value. Therefore, two instructions that have RAR Hazard can execute on successive cycles.
- **RAW Hazard:** This hazard occurs when an instruction reads a register that was written by a previous instruction. These are called as data dependencies.
- **WAR and WAW** are also called as name dependencies.

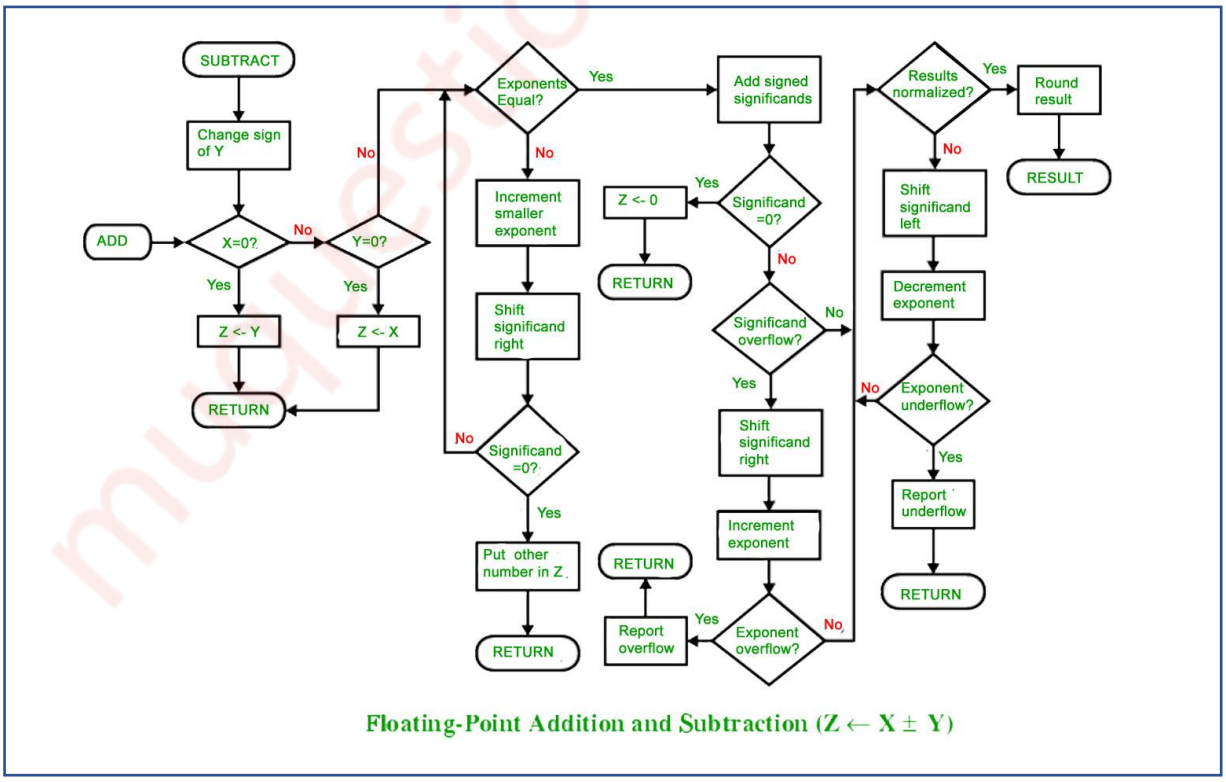
- These hazards occurs when the output register of an instruction has been either read or written by a previous instruction.
- If the processor executes instructions in the order that they appear in the program and uses the same pipeline for the instructions, WAR and WAW hazards do not cause any problem in execution process.
- **Branch Hazards:** Branch instructions, particularly conditional branch instructions, create data dependencies between the branch instruction and the previous instruction, fetch stage of the pipeline.
- Since the branch instruction computes the address of the next instruction that the instruction fetch stage should fetch from, it consumes some time and also some time is required to flush the pipeline and fetch instructions from target location. This time wasted is called as branch penalty.

---

b) Draw and explain floating point addition and subtraction algorithm. (10 M)

Ans:

- In floating point arithmetic, addition and subtraction are more complex than multiplication and division. Addition and subtraction operations are carried out in four basic phases.
    - Check for zeros
    - Align the significant
    - Add or subtract the significant
    - Normalize the result



Floating-Point Addition and Subtraction ($Z \leftarrow X \pm Y$)

- In the next phase, exponents of the two numbers X and Y are made equal. Alignment is achieved by shifting either the smaller number to its right or shifting the larger number to the left.
- Since either operation may result is loss of digits, it is the smaller number is shifted. The alignment is achieved by repeatedly shifting the magnitude portion of the significant right 1 digit and incrementing the exponent until the two digit exponents are equal.
- Next, the two significant are added together, taking into account their signs. Since the sign may differ, the result may be 0. There is also the possibility of significant overflow.
- Next, result Is normalized. Normalisation consists of significant digits left until the most significant digits is nonzero. Each shift causes a decrement of the exponent and thus could cause an exponent overflow.
