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:

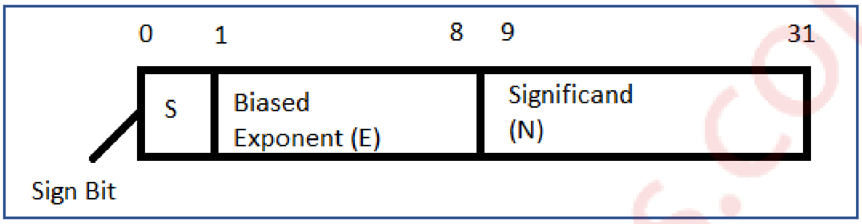
Harvard Architecture	Von Neumann Architecture
1. In Harvard architecture, the CPU is connected with both the data memory (RAM) and program memory (ROM), separately.	**1.** In Von-Neumann architecture, there is no separate data and program memory. Instead, a single memory connection is given to the CPU.
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 = $(132)_{10} = (10000100)_2$

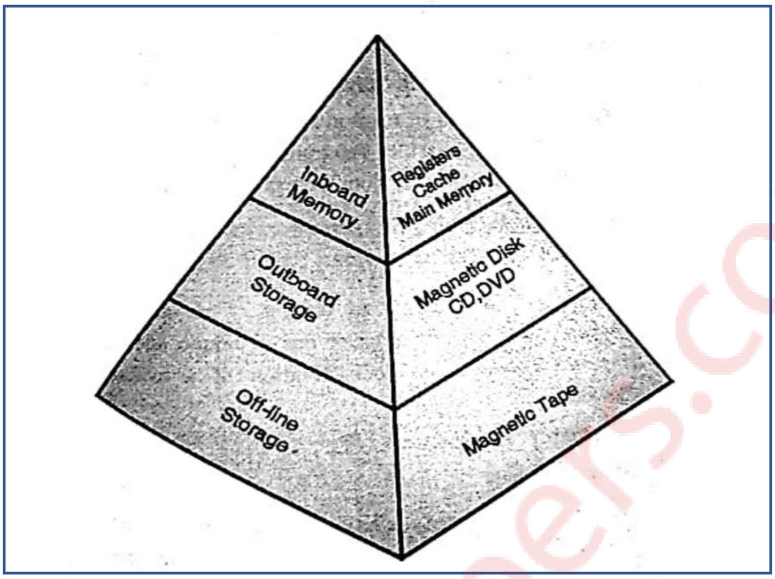
---

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.

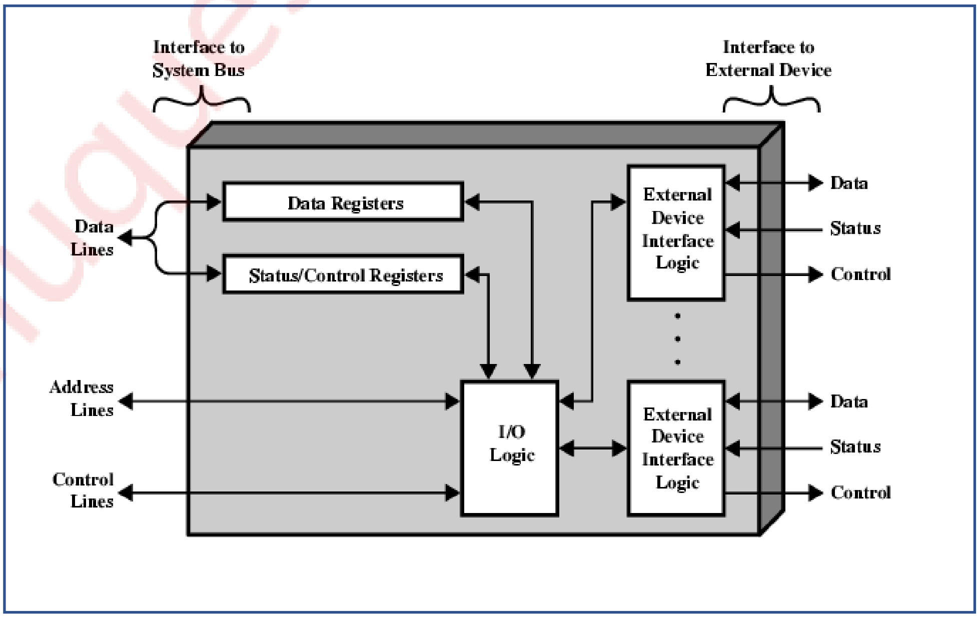


- To have a large faster memory is very costly and hence the different memory at different memory at different levels gives the memory hierarchy.

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.

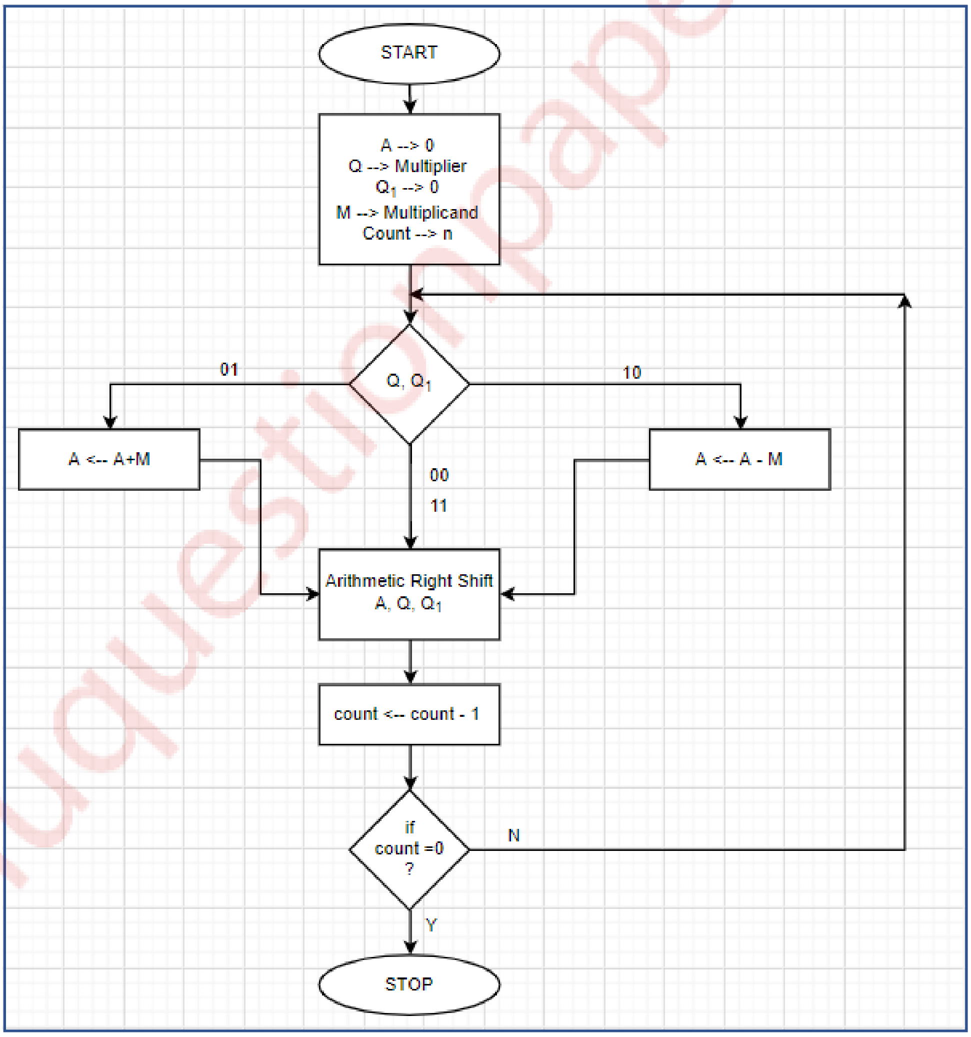


- I/O module is interface to CPU and memory with one or more peripherals.
- The general model of I/O module interfacing with system bus.
- The various functions of the I/O module involve:
  - Issue of control and timing signals.
  - Communication with CPU
  - Communication with peripheral
  - Buffering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

---

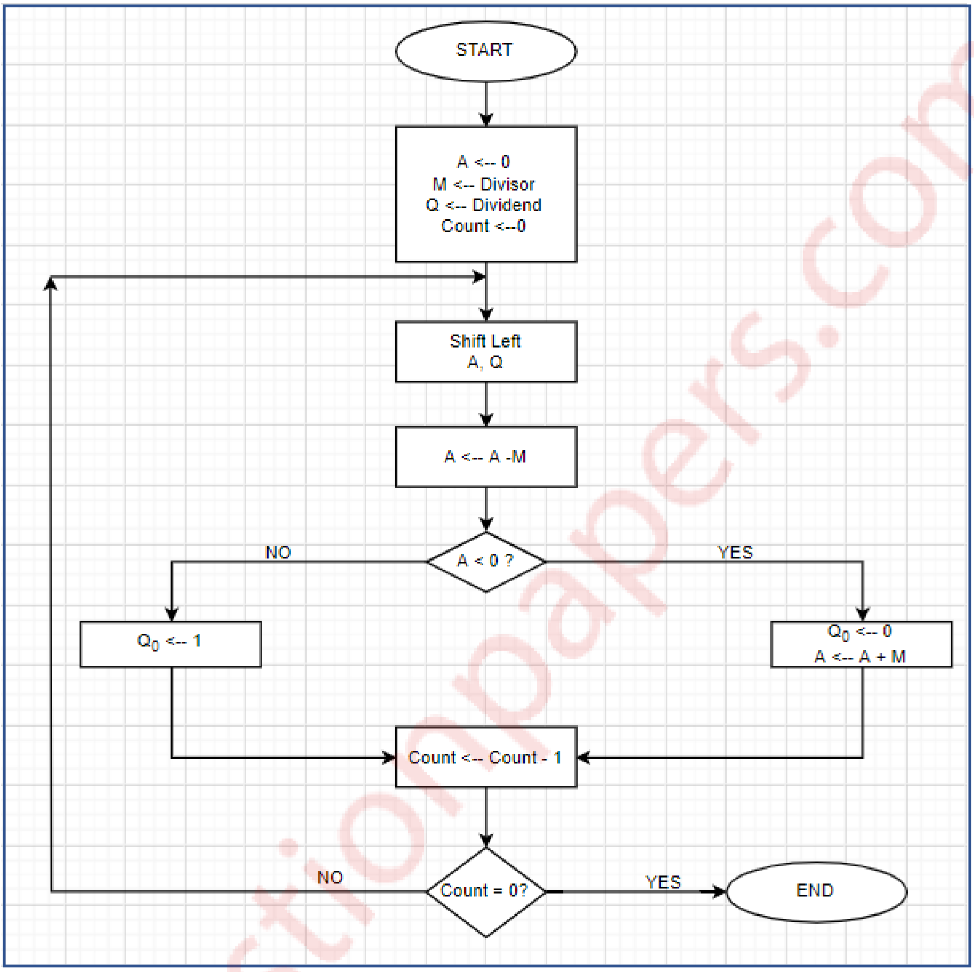
b) Explain the restoring method of binary division with algorithm. Divide $(7)_{10}$ by $(3)_{10}$ using restoring method of binary division. (10 M)

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	$111\underline{0}$			Left Shift
+ 1101	1011			
1101				A←A – M
+ 0011				
00000	1110		3	A←A + M
0001	$110\underline{0}$			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)

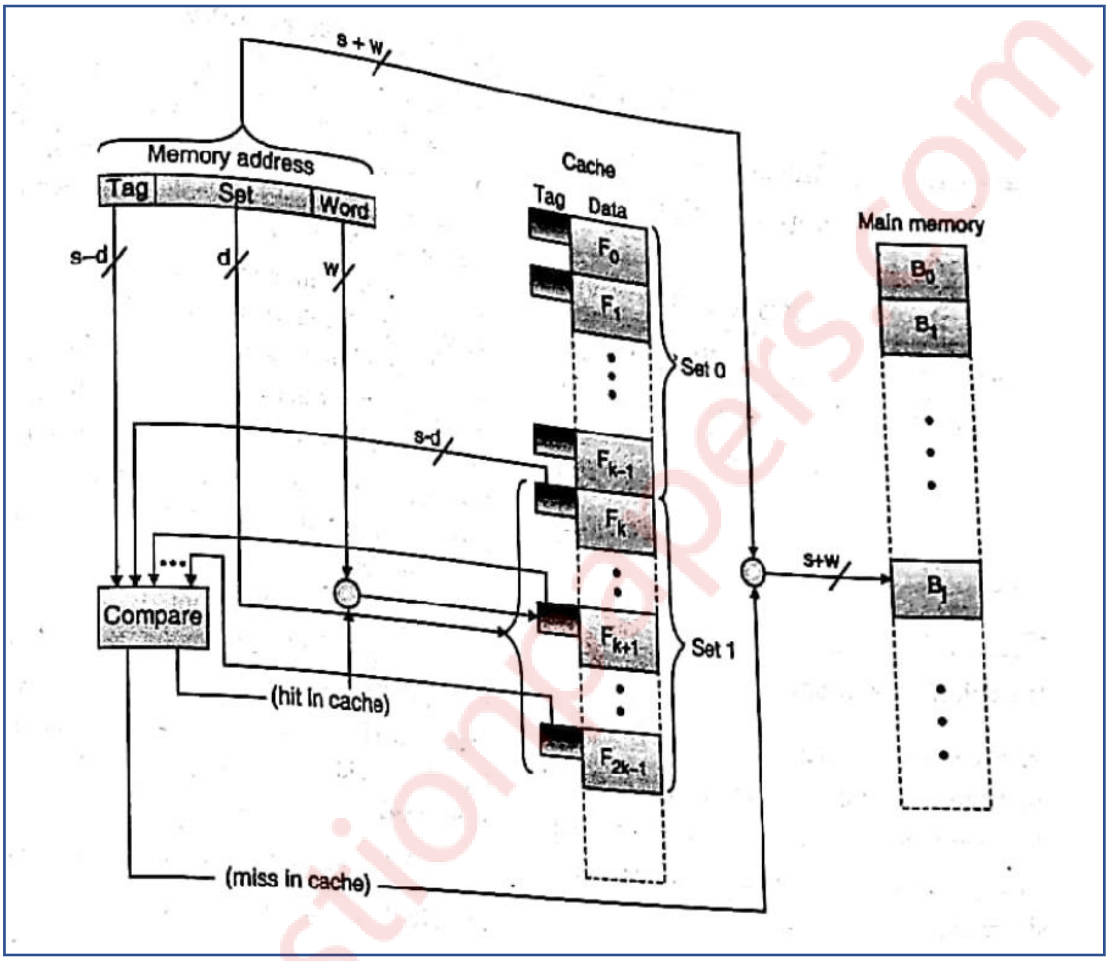
Ans:

- 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.



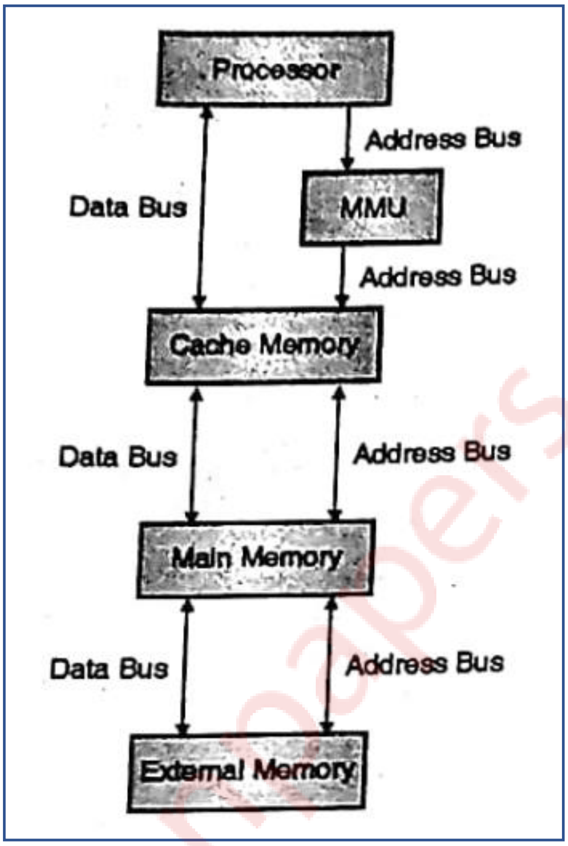
b) Explain the page address translation in case of virtual memory and explain TLB. (10 M)

Ans:

Virtual Memory:-

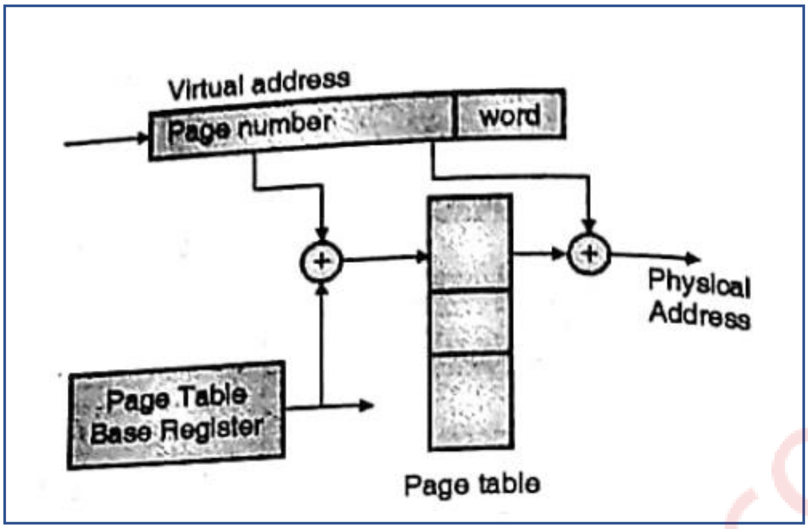
- Virtual Memory was introduced in the system in order to increase the size the size of memory.
- A hardware unit called Memory Management Unit (MMU) translates Virtual addresses into physical addresses.
- If CPU wants data from main memory and it is not present in main memory then MMU causes operating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



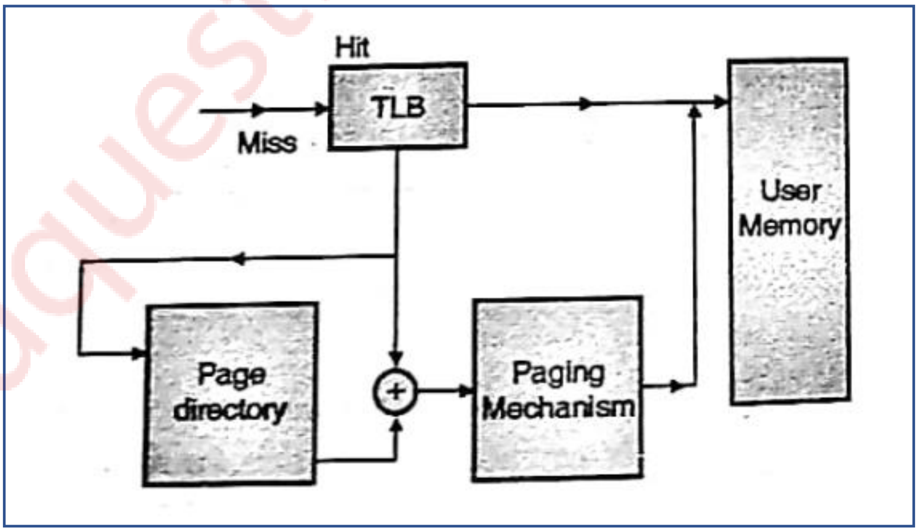
Paging:

- 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.

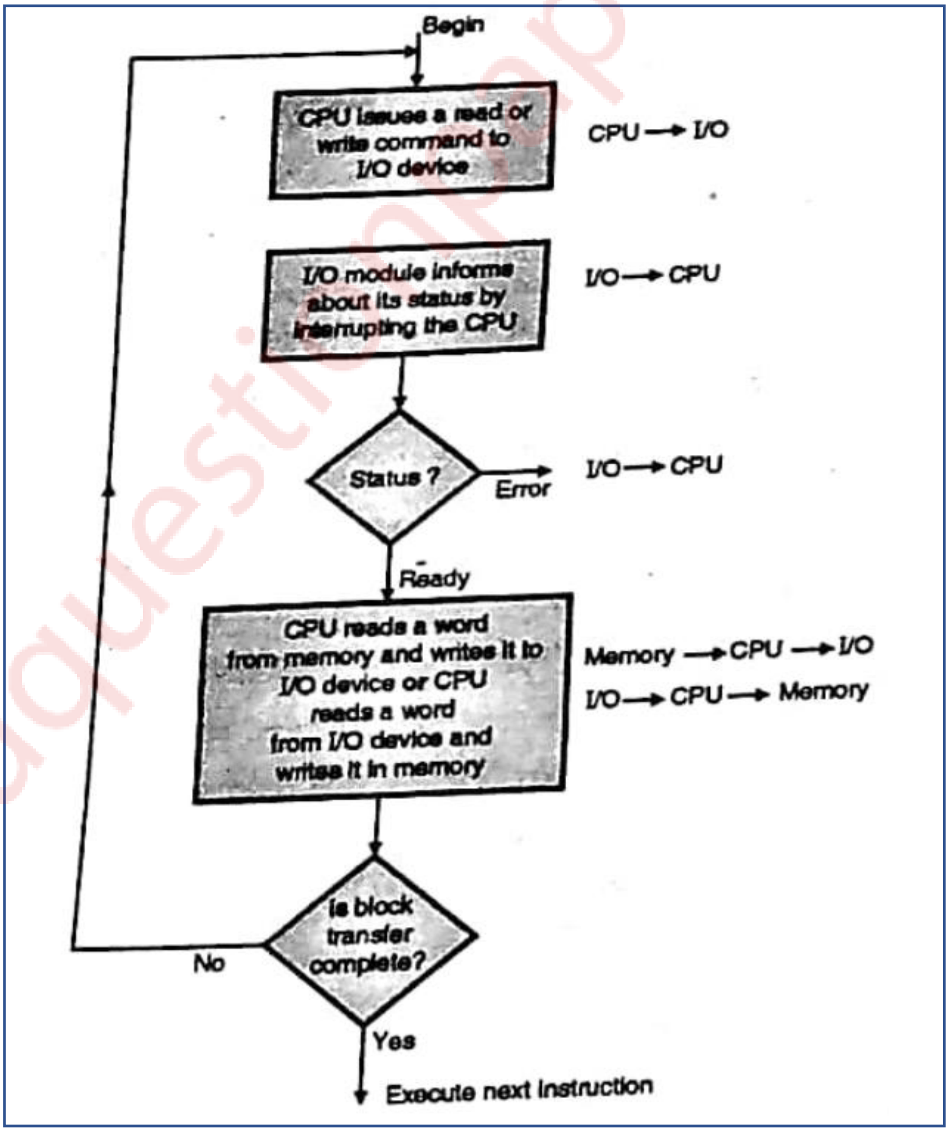


Q.4)

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.


Begin
CPU issues a read or write command to I/O device
CPU → I/O
I/O module informs about its status by interrupting the CPU
I/O → CPU
Status ?
Error
I/O → CPU
Ready
CPU reads a word from memory and writes it to I/O device or CPU reads a word from I/O device and writes it in memory
Memory → CPU → I/O
I/O → CPU → Memory
Is block transfer complete?
No
Yes
Execute next instruction

- 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

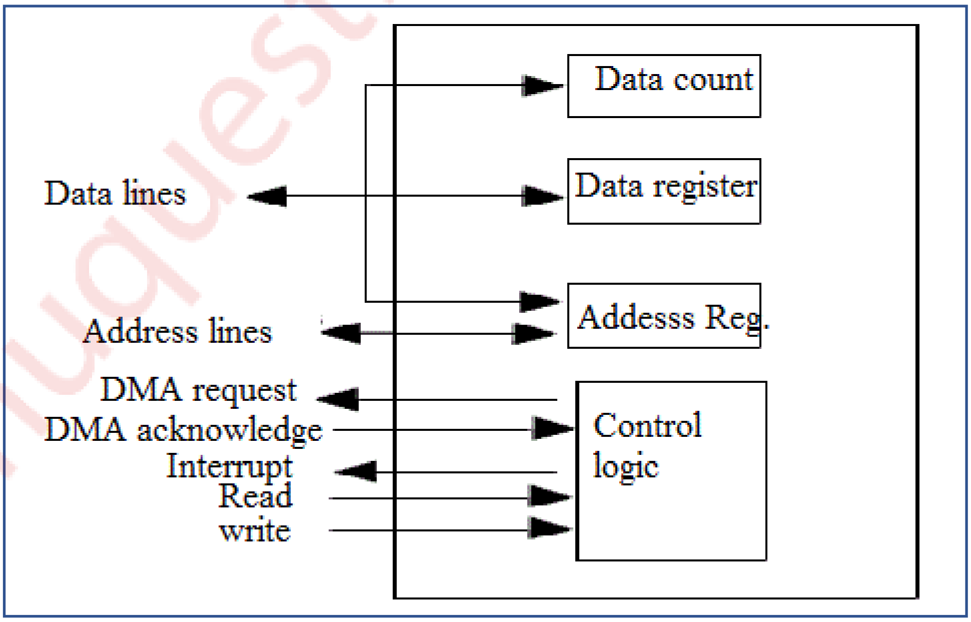
- CPU issues a ‘READ‘ command to I/O device and then switches to some other program. CPU may be working on different programs.
- Once the I/O device is ready with the data in its data register. I/O device signals an interrupt to the CPU.
- When then interrupt from I/O device occurs, it suspends execution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

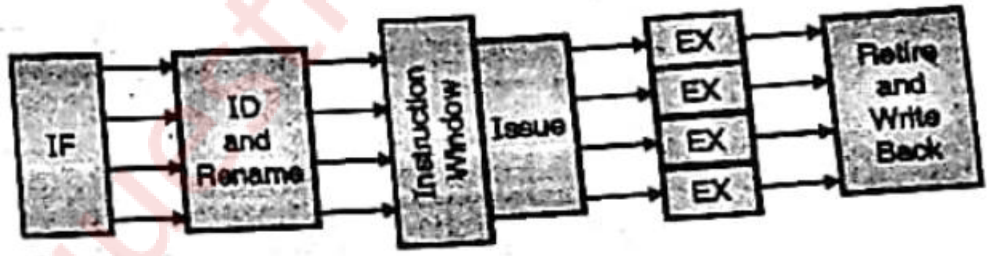
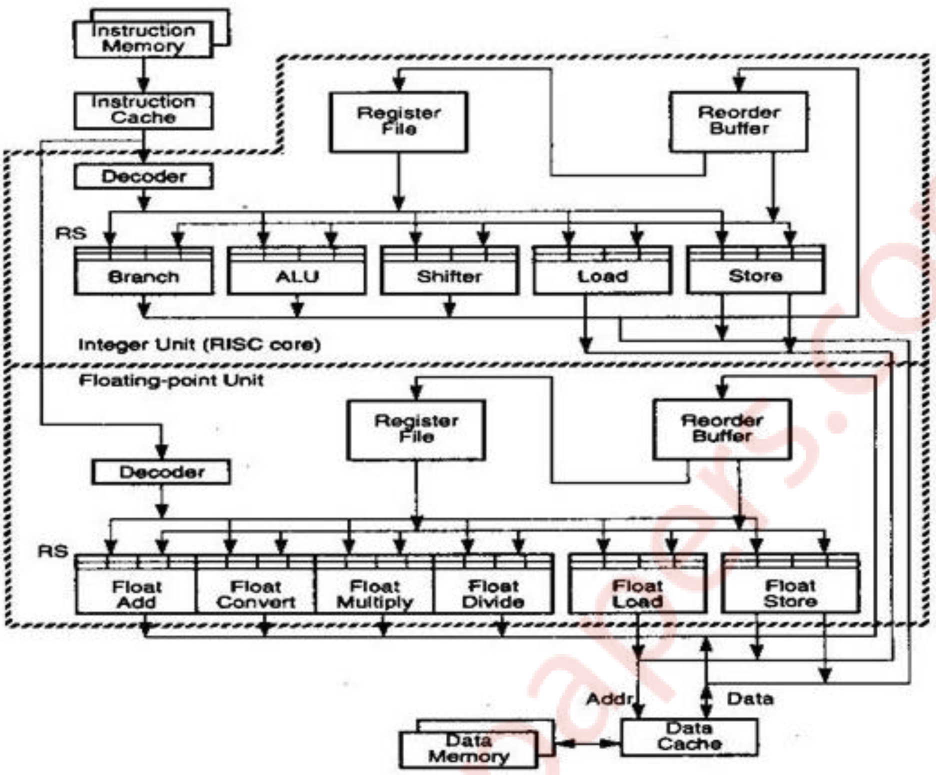


Fig. 6.3.2 : Block diagram of a typical superscalar processor

- 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)**

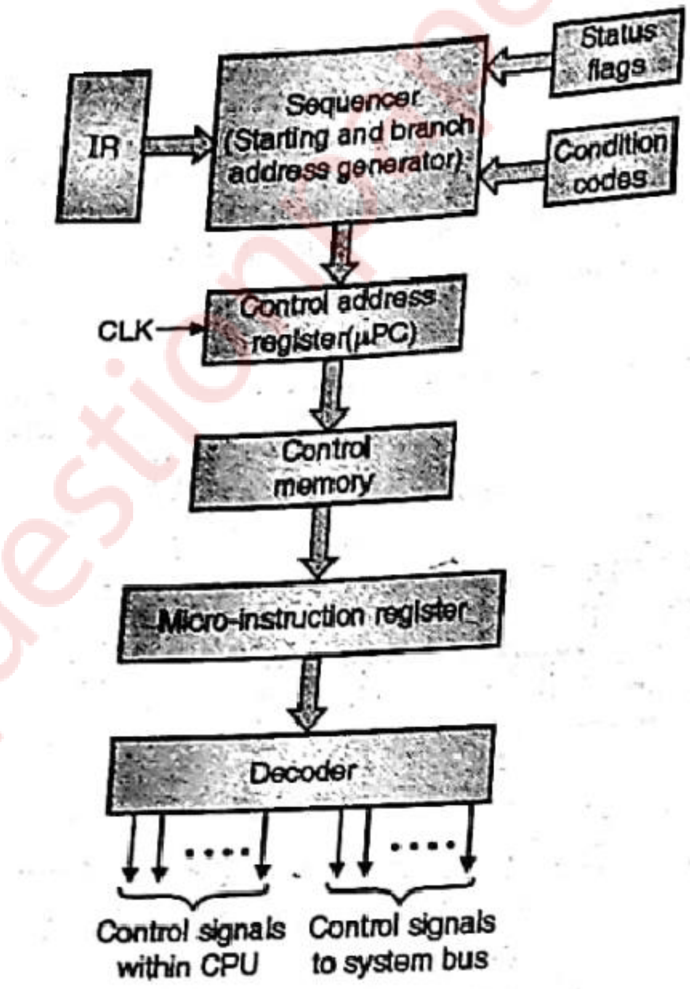
Ans:

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.


Status flags
IR
Sequencer (Starting and branch address generator)
Condition codes
CLK
Control address register(μPC)
Control memory
Micro-instruction register
Decoder
Control signals within CPU
Control signals to system bus

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

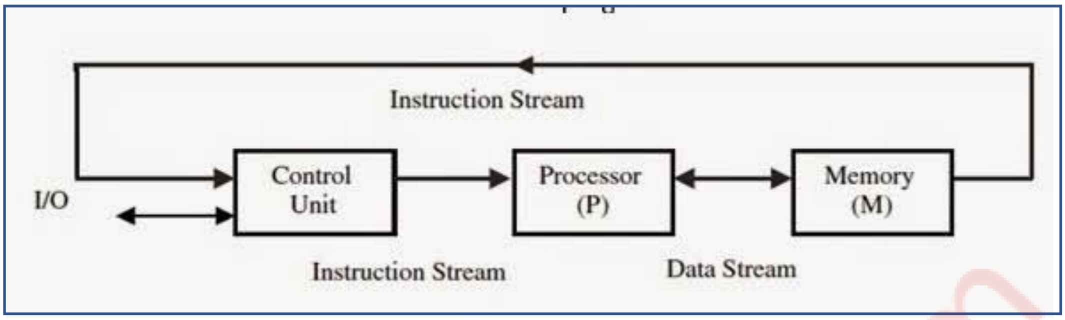
Pipeline Hazards:

- 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.

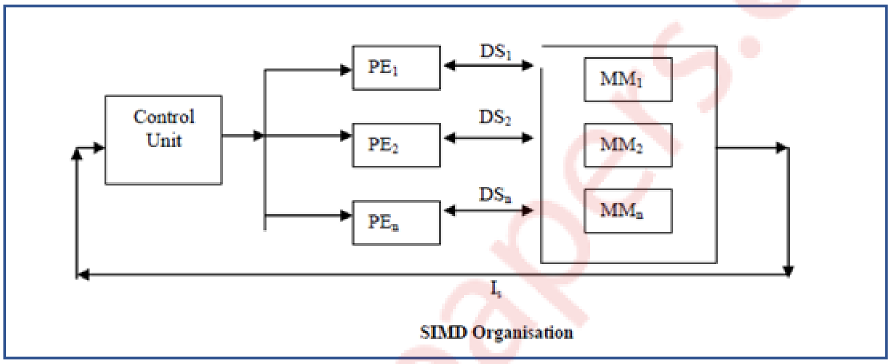
c) Flynn's Classification.

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.

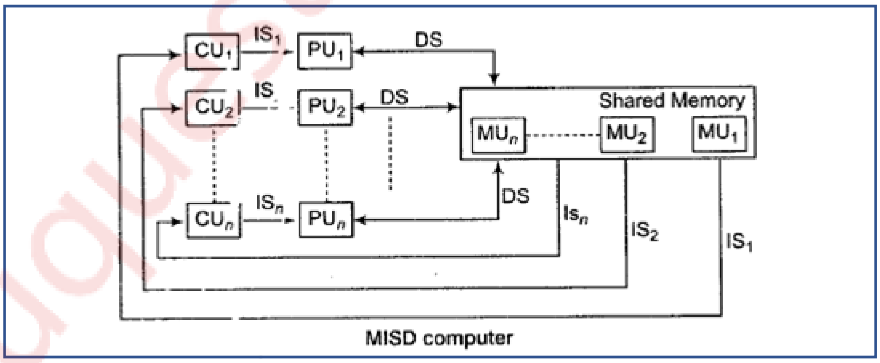


- **SIMD:** In this case the same instruction is given to multiple processing elements, but different data.
- This kind of system is mainly used when many data have to be operated with same operation.



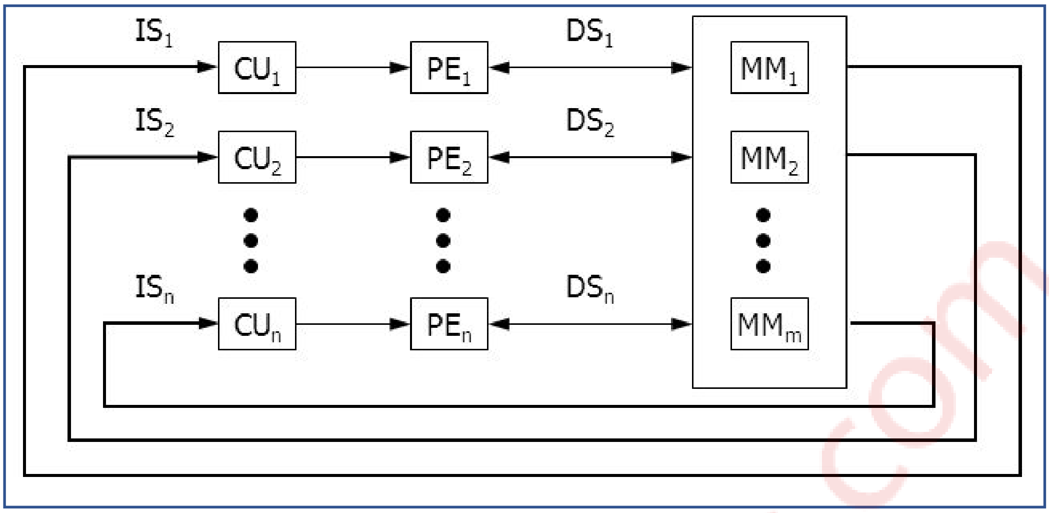
SIMD Organisation

- **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.



MISD computer

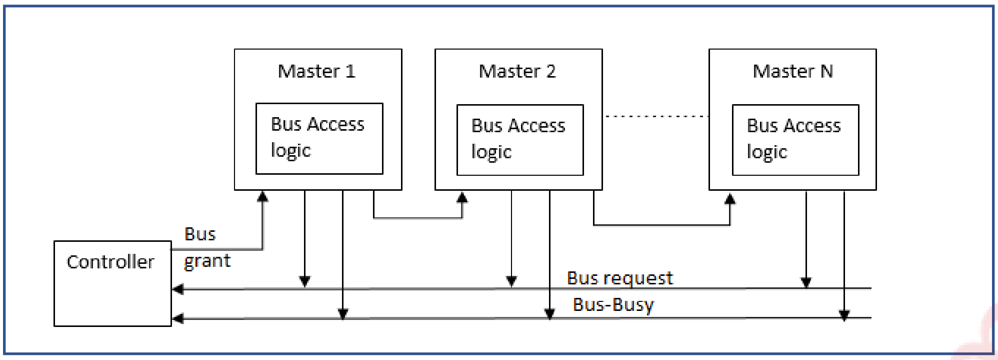
- **MIMD:** This is a complete parallel processing example. Here each processing element is having a different set of data and different instructions.
- Examples of this kind of systems are SMPs, clusters and NUMA.



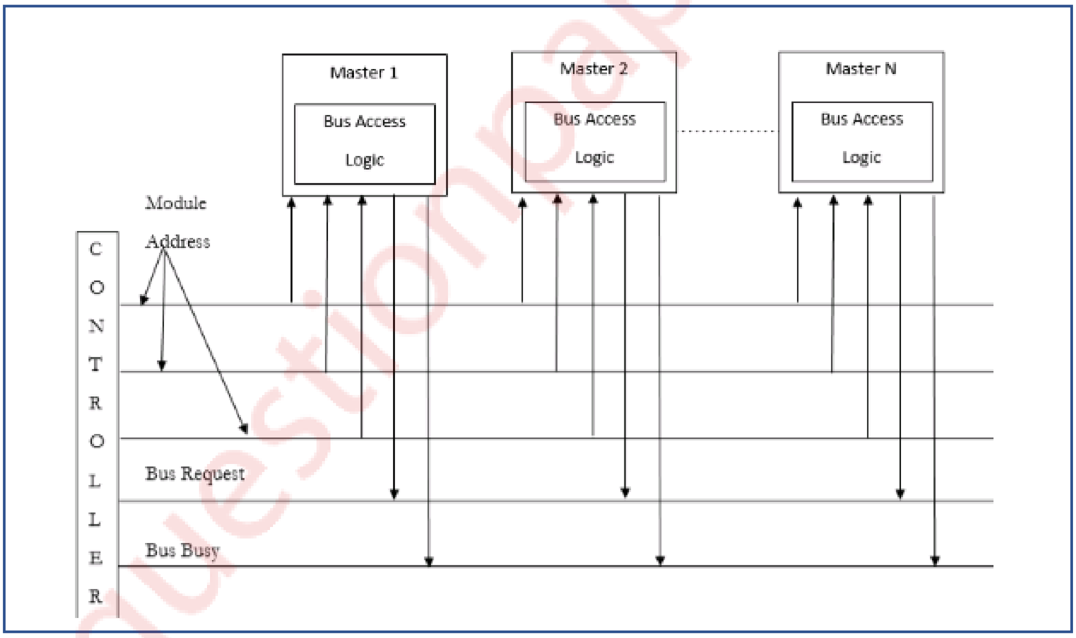
d) Bus Arbitration. (10 M)

Ans:

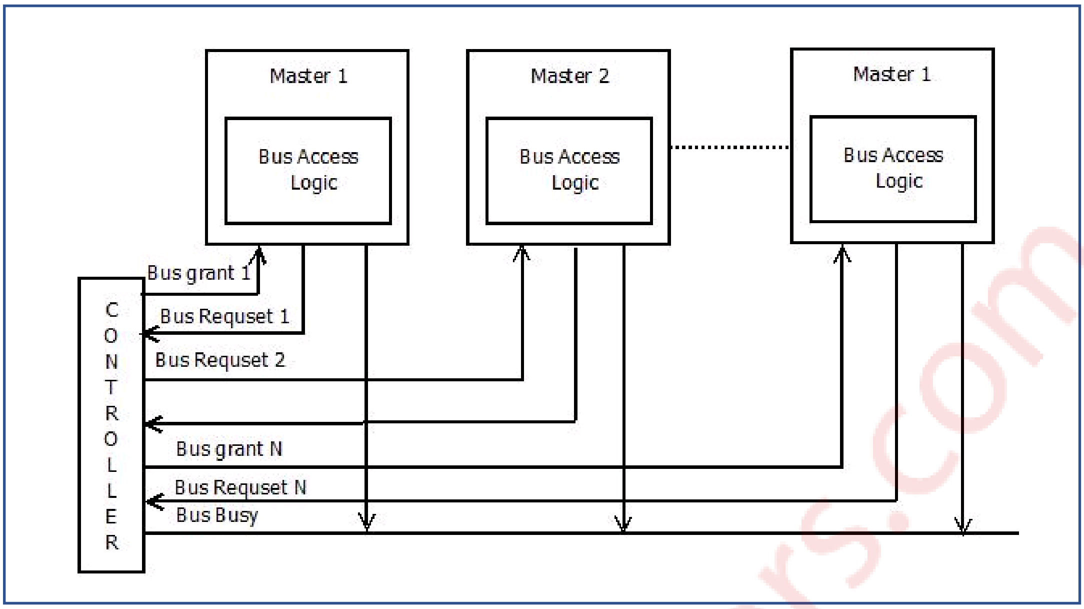
- When many bus masters are connected to a single bus, there is bus congestion. To avoid or reduce the problem of bus congestion we use bus arbitration.
- The process of selecting the processor among requesting processors is known ass arbitration.
- A selection mechanism must be based on fairness or 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.