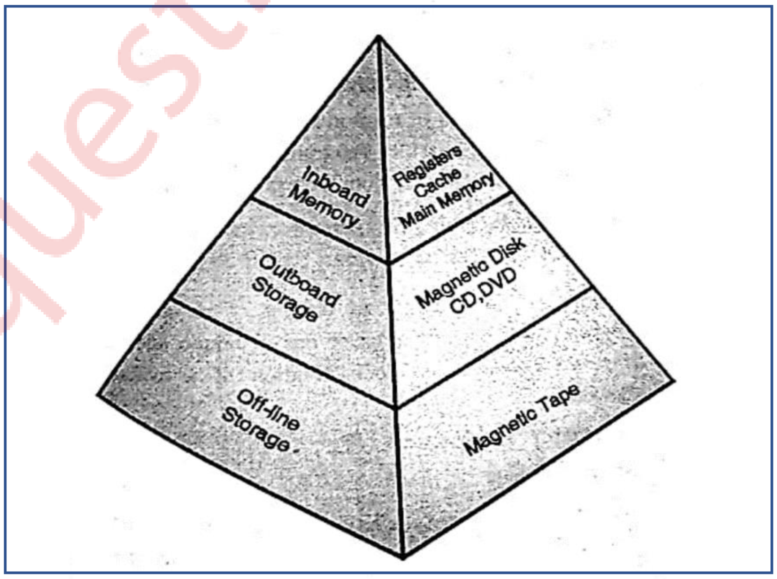
COMPUTER ORGANISATION AND ARCHITECTURE (DEC 2019)

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.



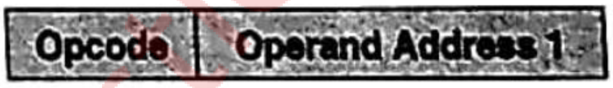
- To have a large faster memory is very costly and hence the different memory at different memory at different levels gives the memory hierarchy.

---

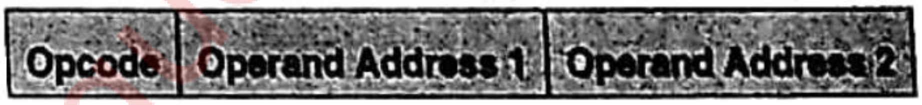
b) Give different instruction format. (5 M)

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.



(a) Single address instruction format



(b) Two address instruction format

---

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.

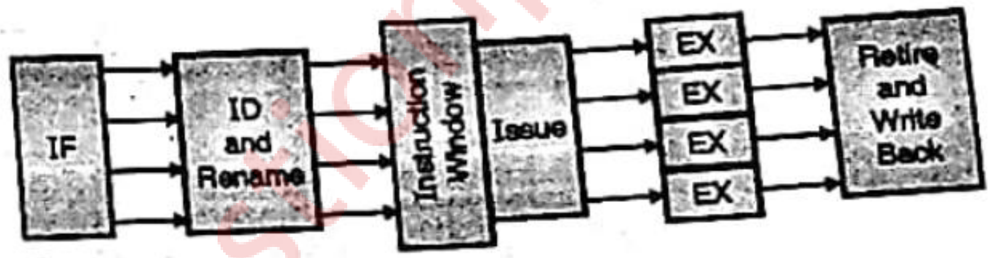
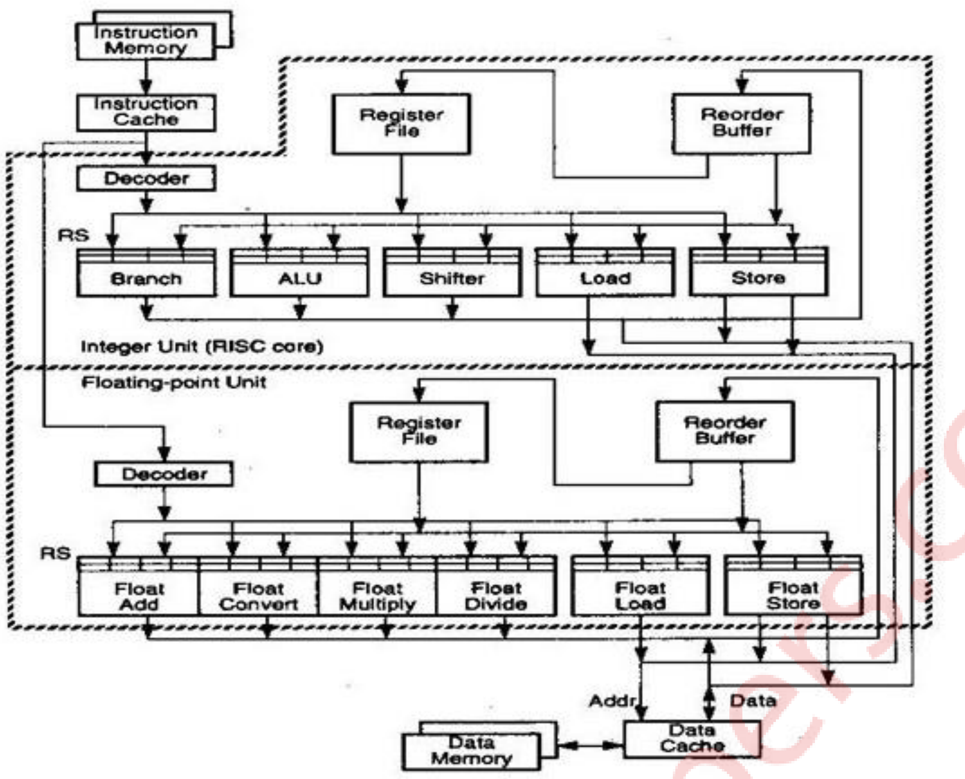


Fig. 6.3.2 : Block diagram of a typical superscalar processor

- 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
……	……	…….

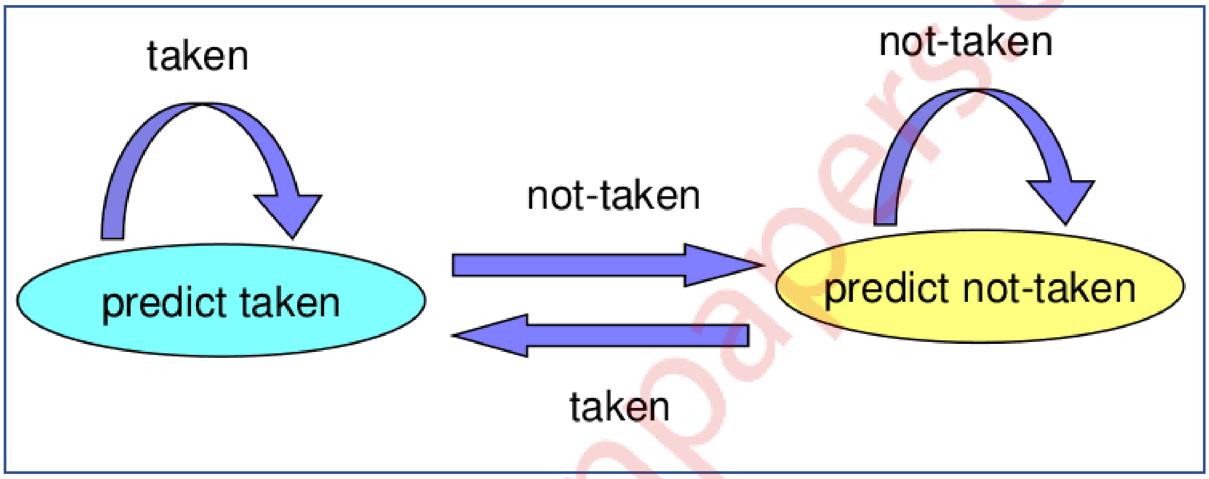
Dynamic Branch Prediction:

- 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.

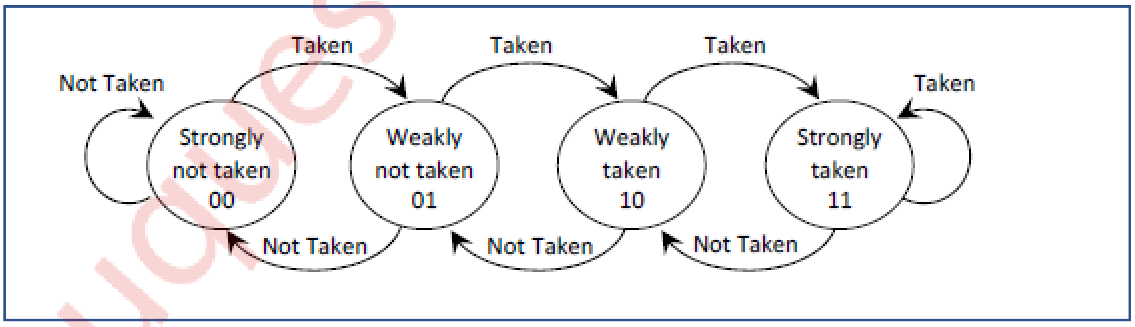
One-bit Dynamic Branch Predictor:

- 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.



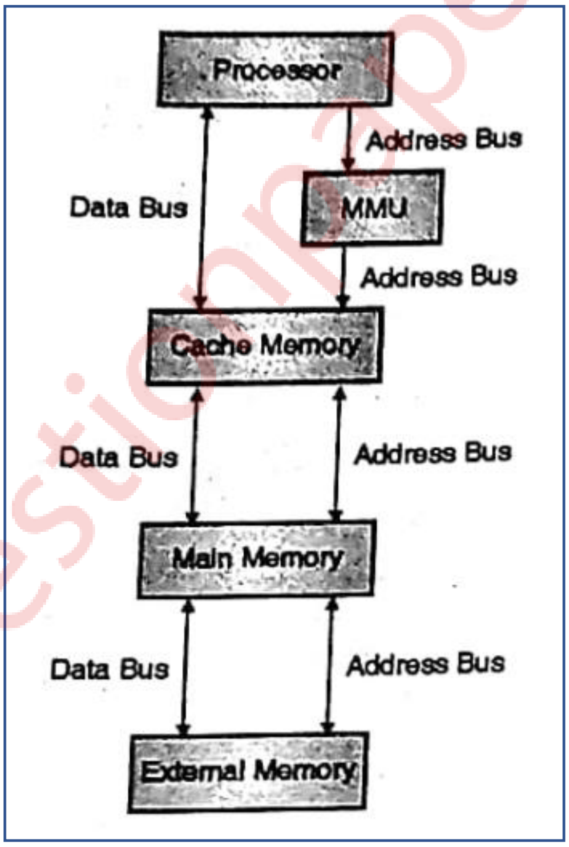
Q.3)

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

Ans:

Virtual Memory:-

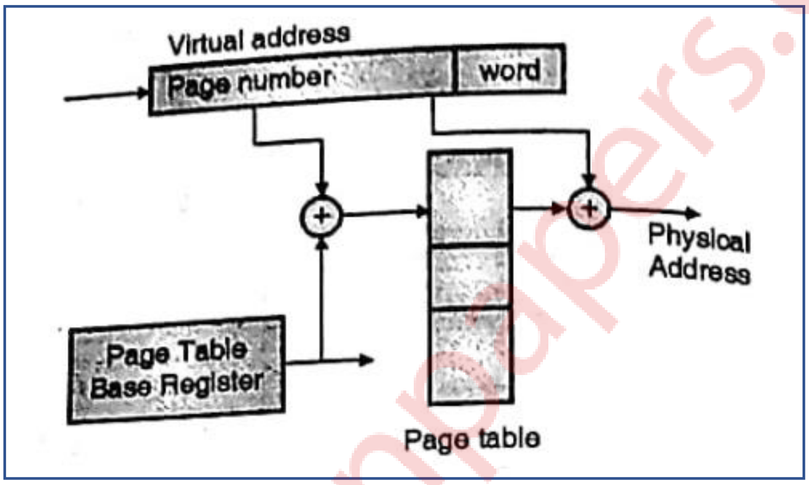
- Virtual Memory was introduced in the system in order to increase the size the size of memory.
- A hardware unit called Memory Management Unit (MMU) translates Virtual addresses into physical addresses.
- If CPU wants data from main memory and it is not present in main memory then MMU causes operating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

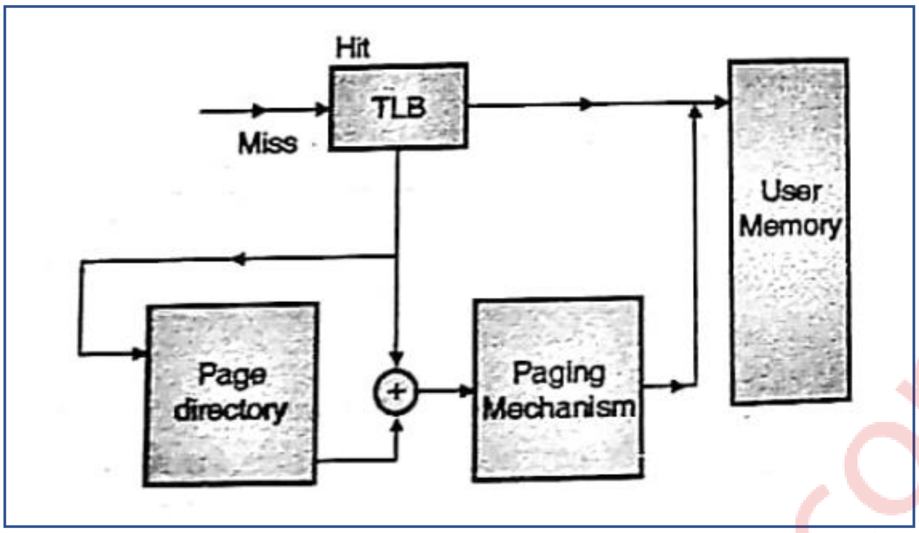
- Virtual Memory space is divided into equal size pages.
- Main Memory space is divided into equal size page frames each frame can hold any page from Virtual Memory.

- 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**

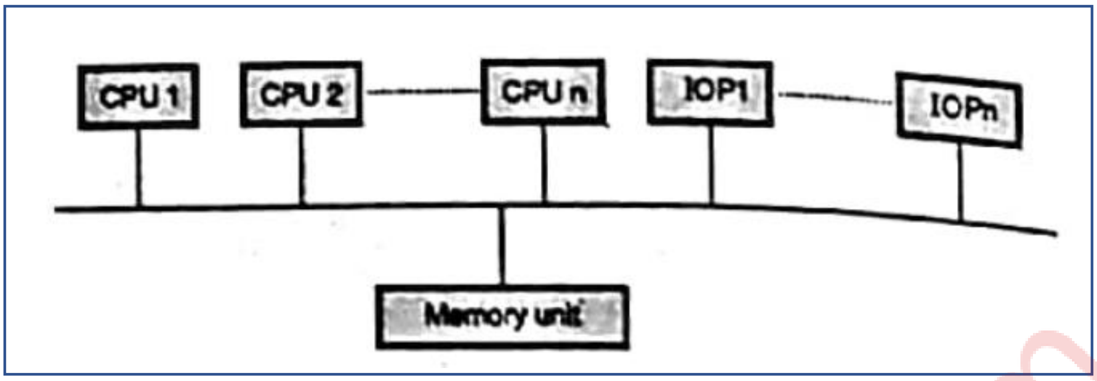
T-state	Operation	Microinstructions
T 1	PC → MAR	PCout, Marin, Read, Clear y, set Cin, Add, Zinn
T 2	M → MBR PC ← PC + 1	Zout, PCin, Wait for memory fetch cycle
T 3	MBR → IR	MBRout, IRin
T 4	R1 → x	R1out, Xin, CLRC
T 5	R2 → ALU	R2out, MUL, Zin
T 6	Z → R1	Zout, R1in
T 7	Check for intr	Assumption enabled 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:

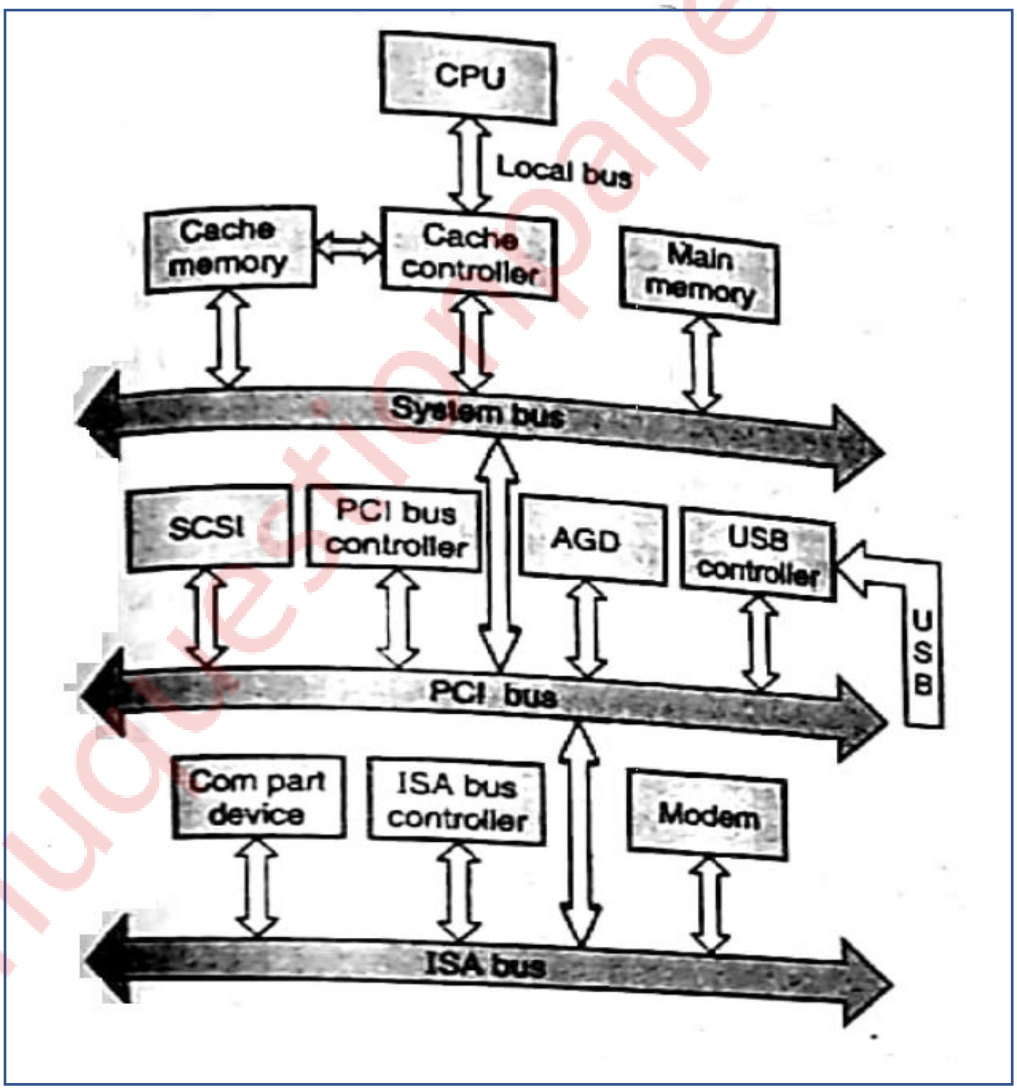
- In a bus system, processors, memory modules and peripheral devices are attached to the bus. The bus can handle only one transaction at a time between a master and slave. In case of multiple requests, the bus arbitration logic must be able to allocate or deallocate and it should service request at a 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.

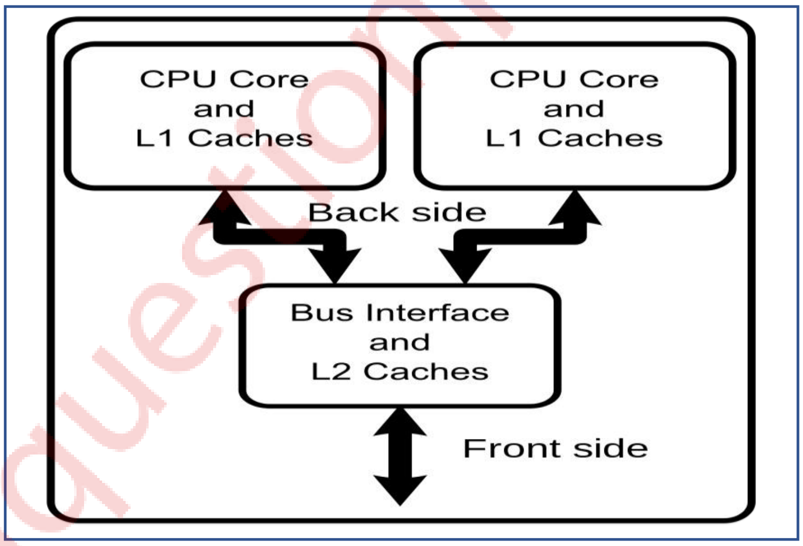
- It occurs when the processor's hardware is not capable of executing all the instructions in the pipeline simultaneously.
- **Data Hazards:** This hazard arises when an instruction depends on the result of a previous instruction, but this result is not available.
- **Branch Hazards:** Branch instructions, 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

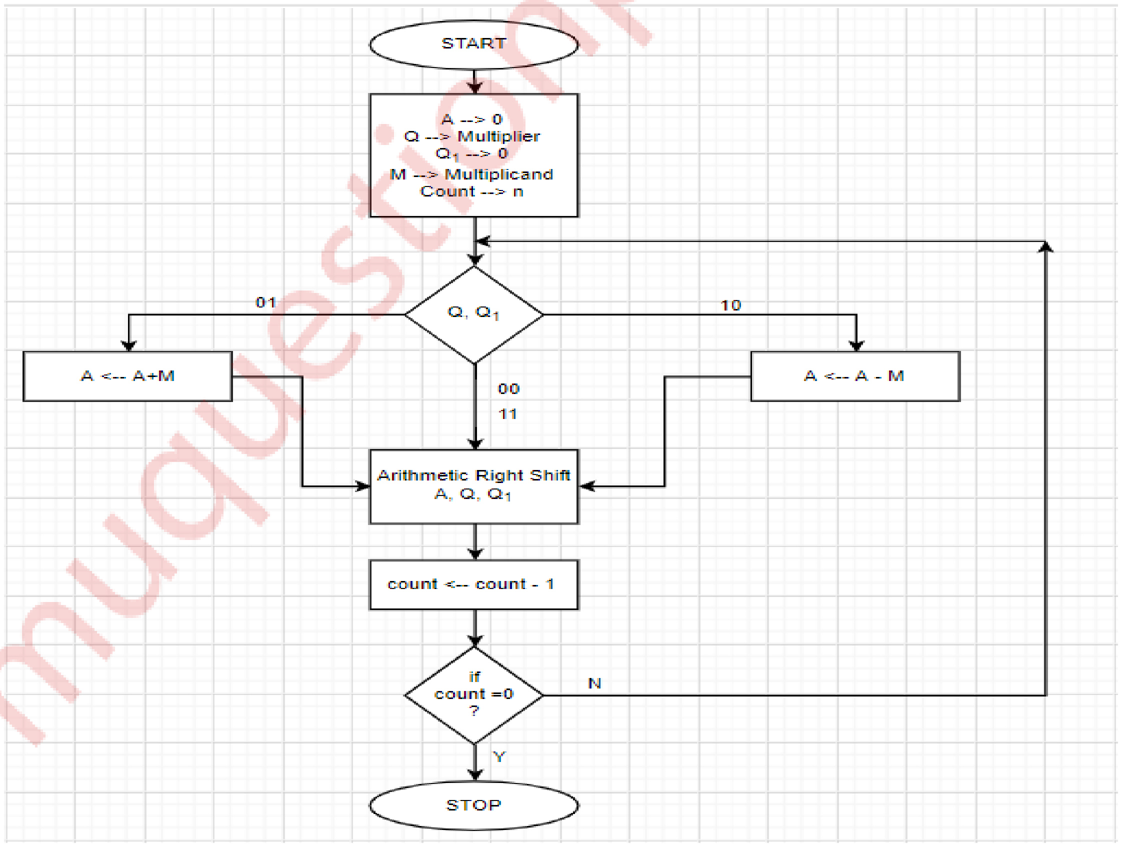
Just as with single-processor systems, cores in multi-core systems may implement architectures such as VLIW, superscalar, vector, or multithreading.

- 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

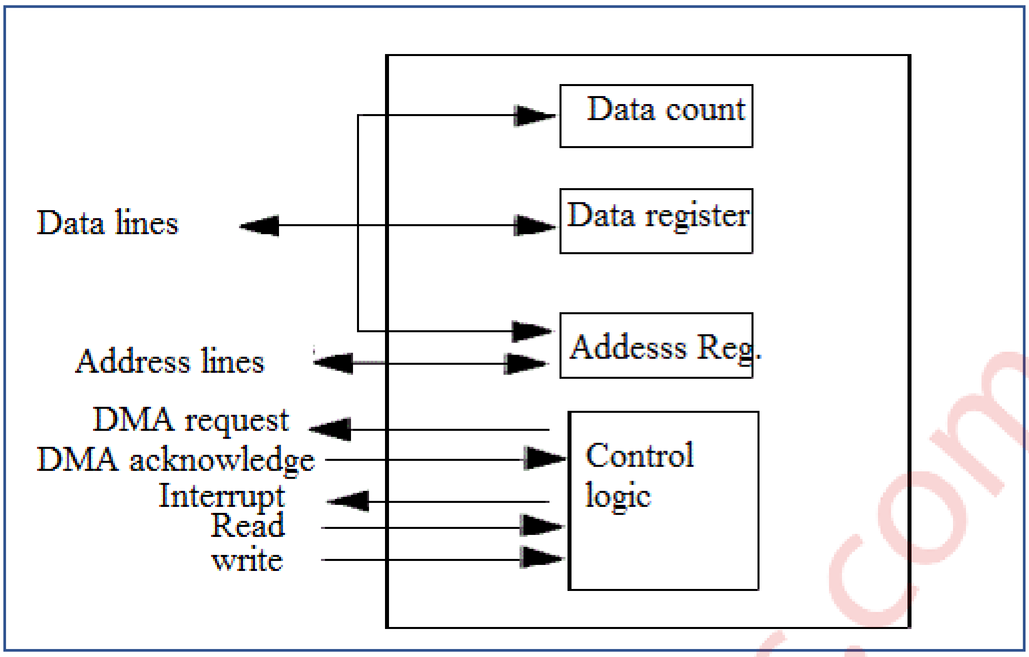
- Interrupt driven I/O overcomes the disadvantage of programmed I/O i.e. the CPU waiting for I/O device.
- This disadvantage is overcome by CPU not repeatedly checking for the device being ready or not instead the I/O module interrupts when ready.
- The sequence of operations for interrupt Driven I/O 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.

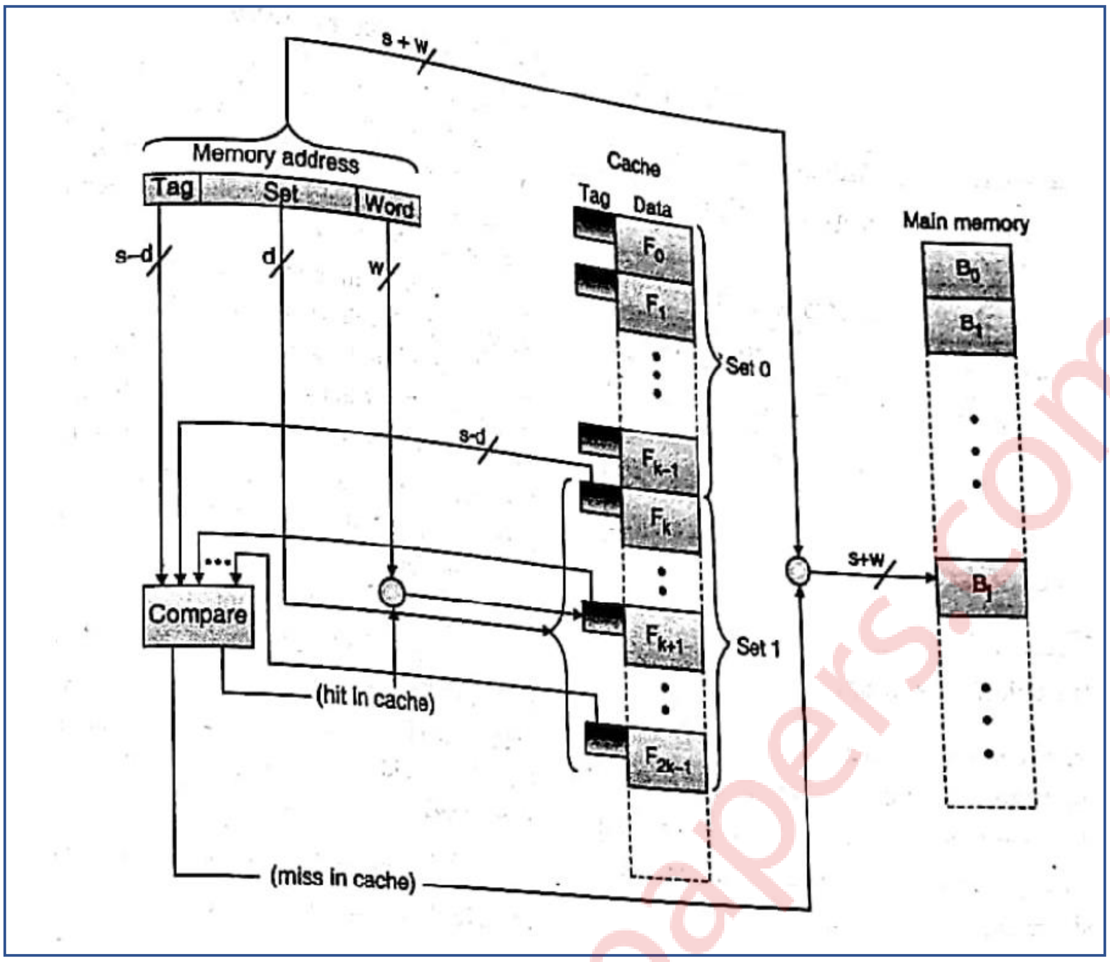
---

b) Set Associative Cache Mapping (10 M)

Ans:

Set Associative Cache :

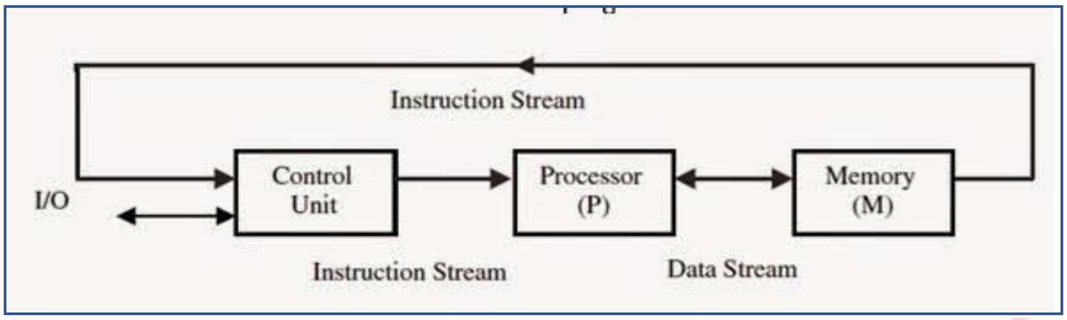
- In set associative cache mapping, cache is divided into a number of sets. Each set contains a number of lines.
- A given block maps to any line in a given set (i mod j), where i is the line number of the main memory to be mapped and j is the total number of sets in cache memory.
- This form of mapping 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.



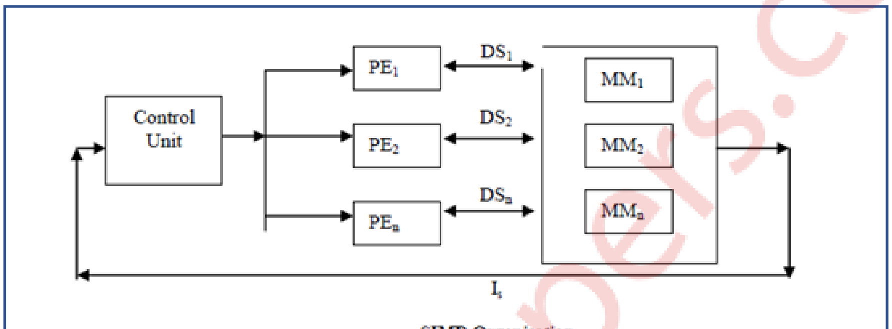
c) Flynn's Classification. (10 M)

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.

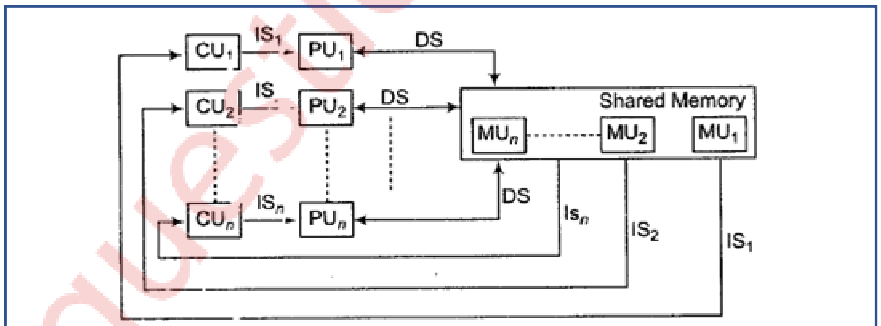


- **SIMD:** In this case the same instruction is given to multiple processing elements, but different data.
- This kind of system is mainly used when many data have to be operated with same operation.



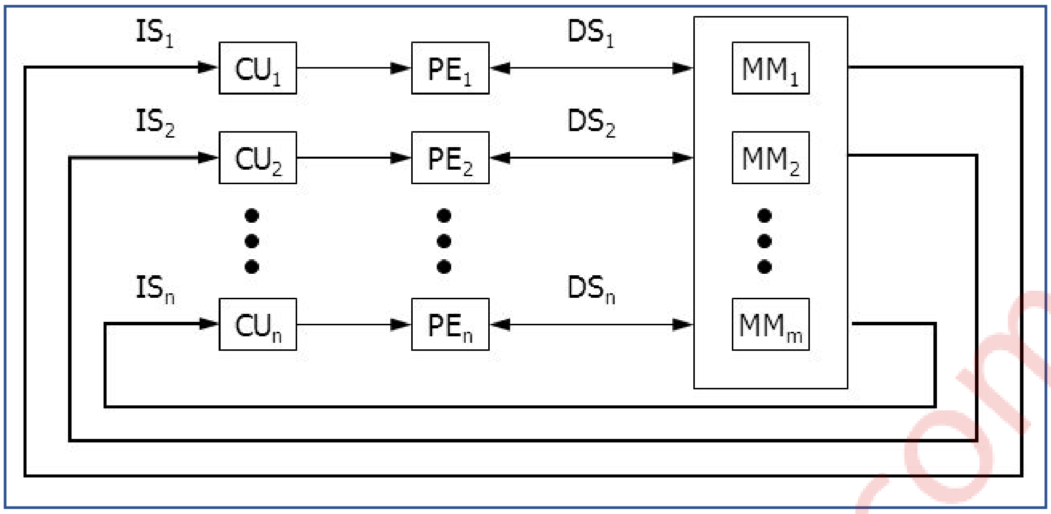
SIMD Organisation

- **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.



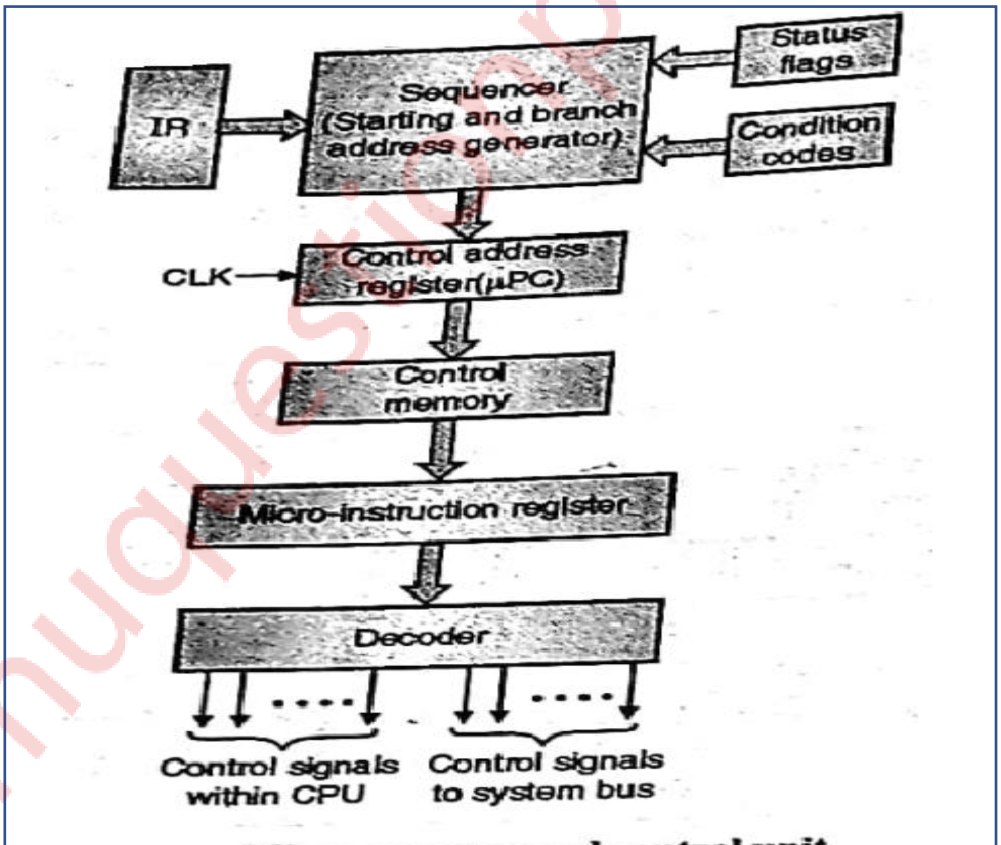
---

d) 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.


IR
Sequencer (Starting and branch address generator)
Status flags
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

- 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.