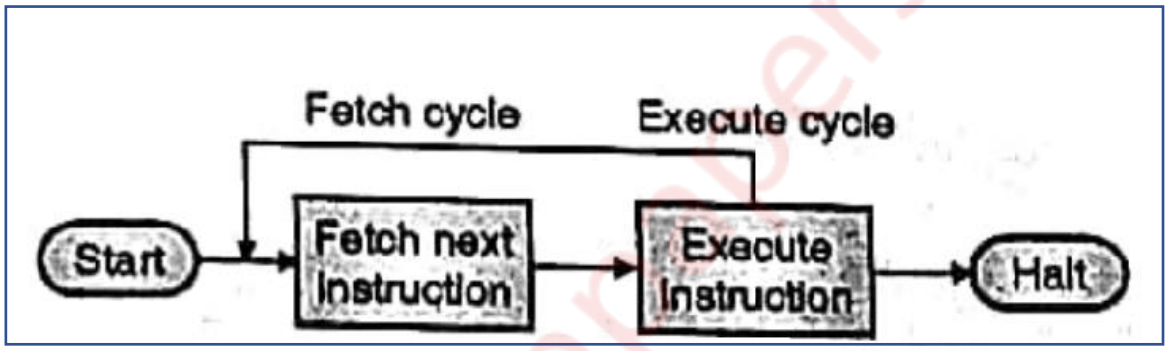
COMPUTER ORGANISATION AND ARCHITECTURE
(NOV 2018) Q.P. 57916

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.

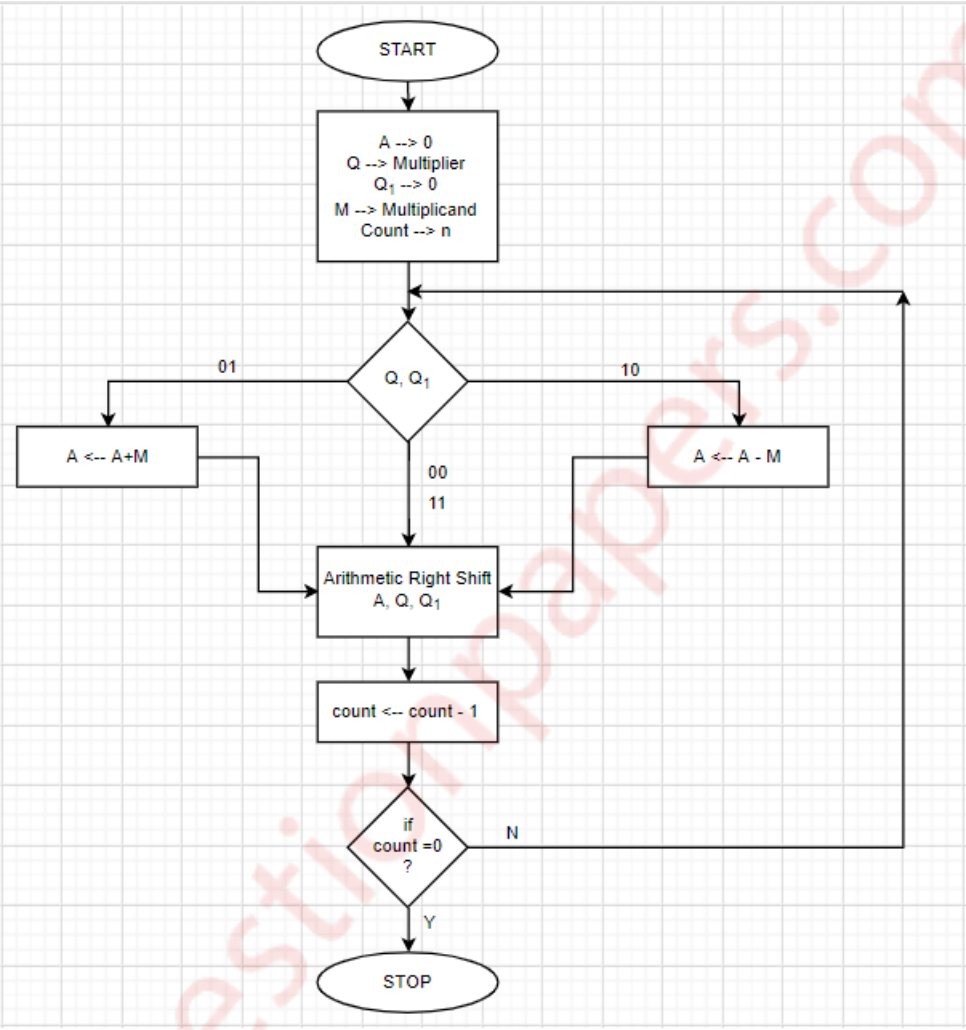


- It comprises of the fetch cycle and executes cycle in a loop to execute huge number of instructions, until it reaches the halt instruction.
- The fetch cycle comprises of the following operations:
    - Program centre holds address of next instruction to fetch; hence the CPU fetches instruction from 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

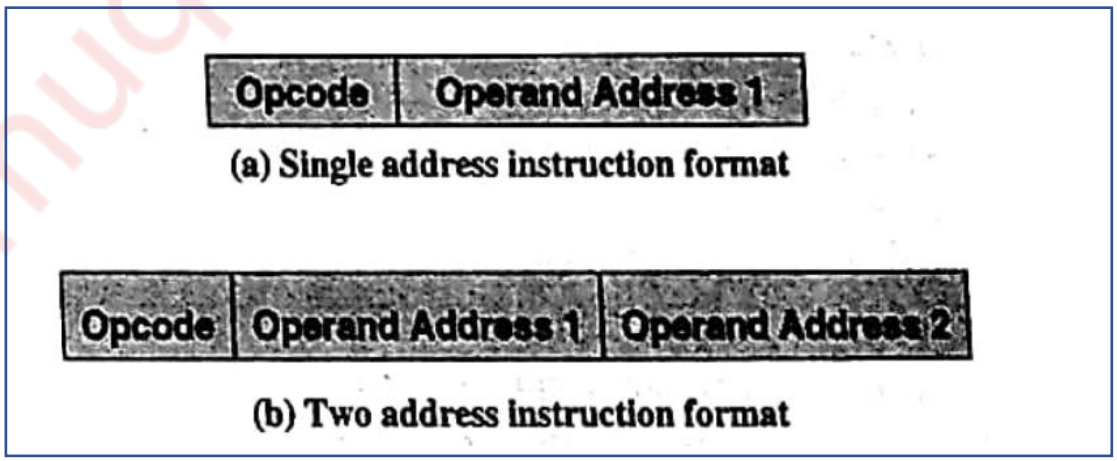
Hence 11101011 = $-(00010101)_2$

$= -(21)_{10}$

c) Give different instruction formats. (5 M)

Ans:

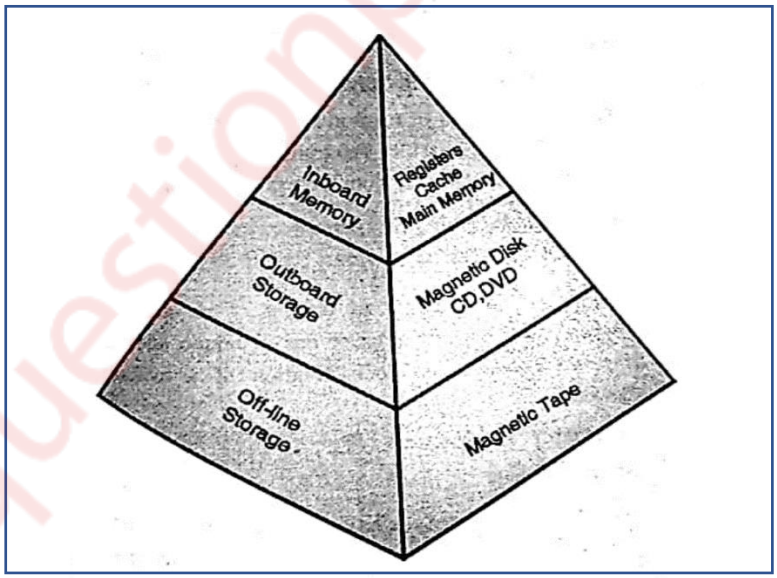
- Input devices are required to give the instructions and data to the system. The output devices are used to give the output devices.
- The instructions and data given by the input device are to be stored, and for storage we require memory.
- Elements of Single, two and three address instructions are 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.

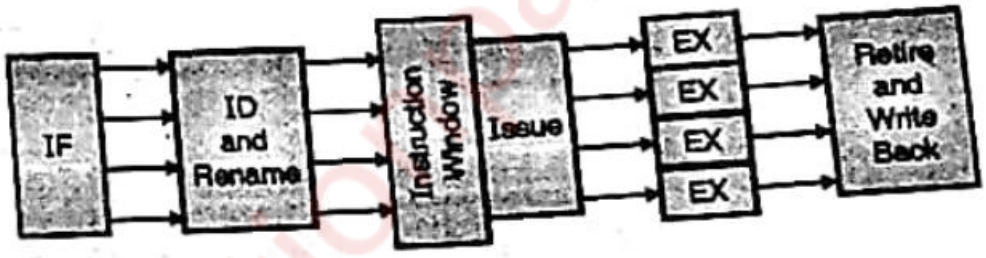
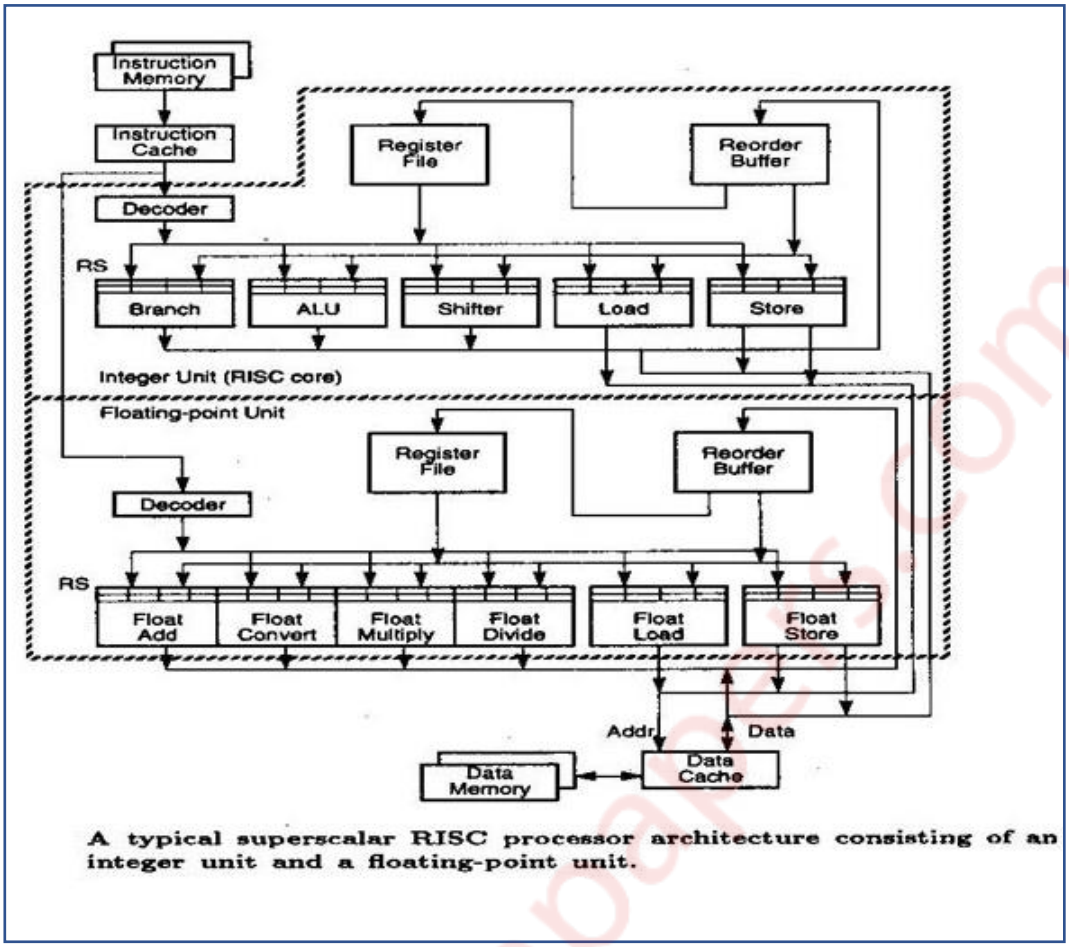


Fig. 6.3.2 : Block diagram of a typical superscalar processor

- 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

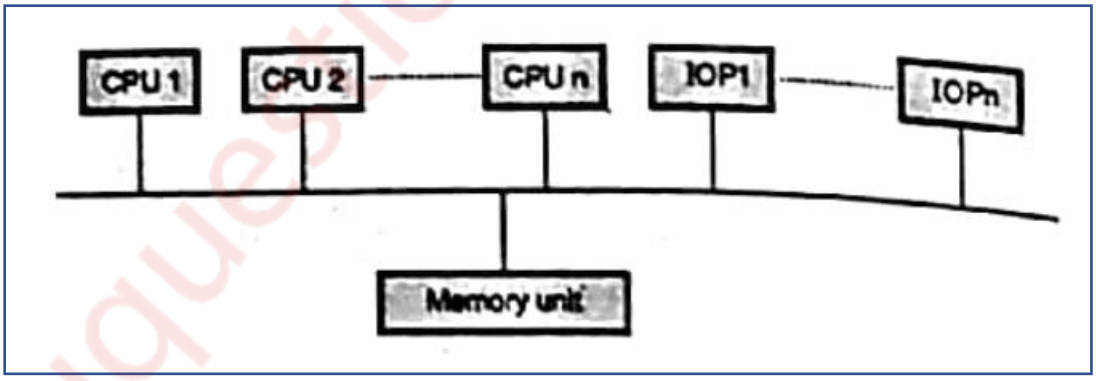
T 6	Z → R1	Zout, R1in
T 7	Check for intr	Assumption enabled intr pending, CLRX, SETC, Spout, SUB, Zin
T 8	SP ← SP – 1	Zout, Spin, MARin
T 9	PC → MDR	PCout, MDRin, WRITE
T 10	MDR → [SP]	Wait for Mem access
T11	PC ← IS Raddr	PCin IS Raddr out.

Q.4)

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

Ans:

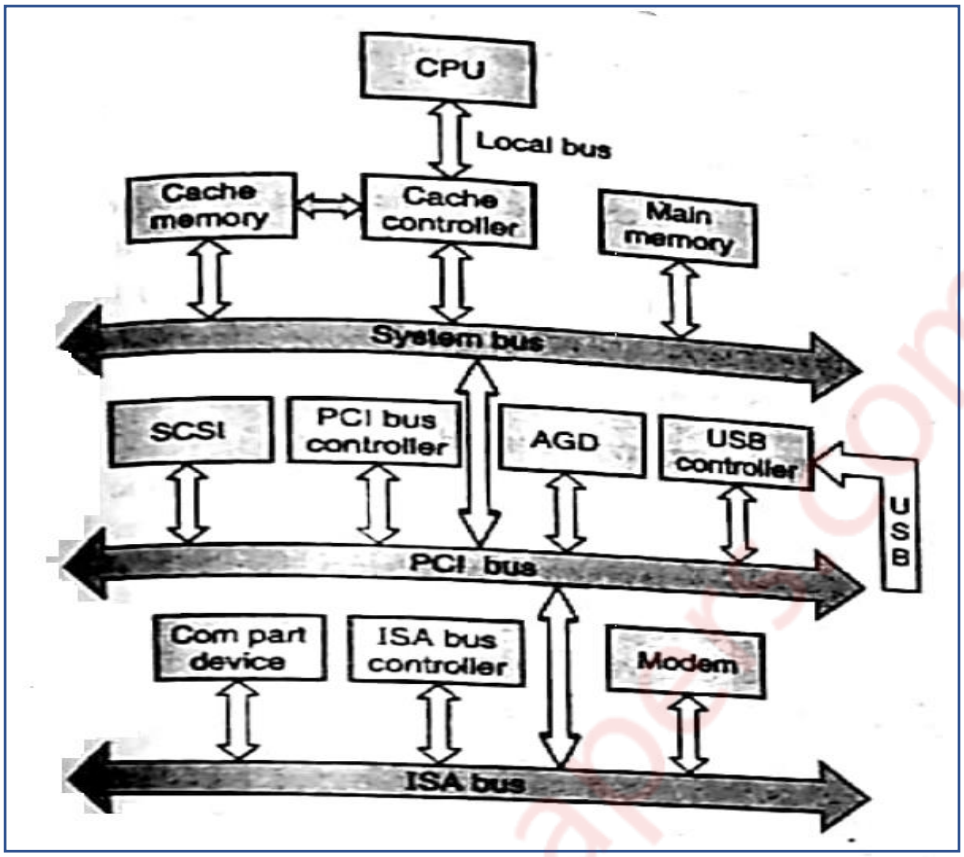
- In a bus system, processors, memory modules and peripheral devices are attached to the bus. The bus can handle only one transaction at a time between a master and slave. In case of multiple requests, the bus arbitration logic must be able to allocate or deallocate and it should service request at a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



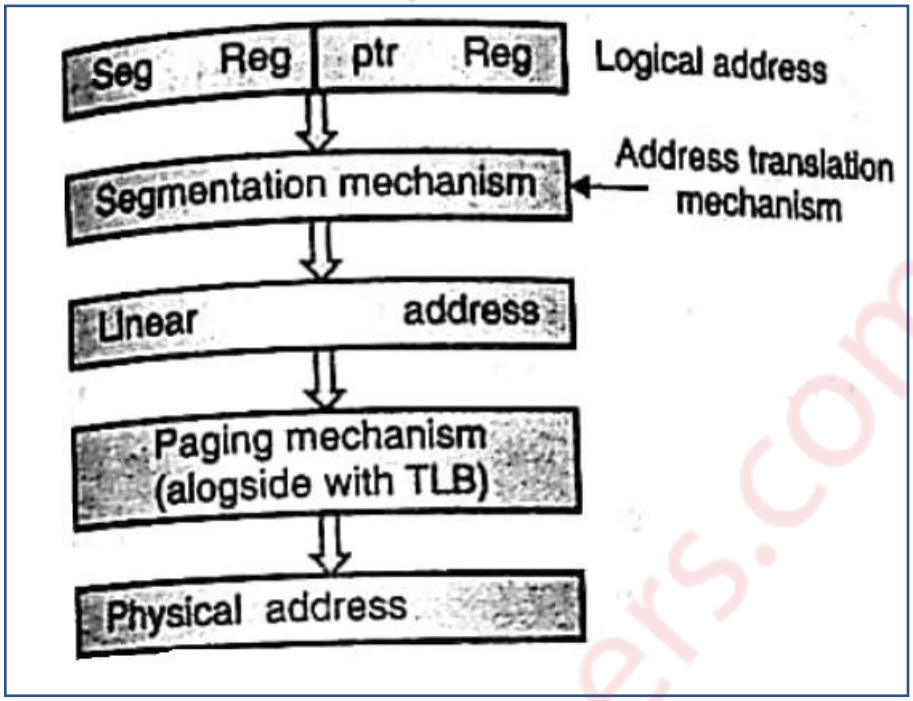
- In general, the more devices attached to the bus, the greater will be propagation delay.
- The bus may become a bottleneck as the aggregate data transfer demand approaches the capacity of the bus.
- This problem can be countered to some extent by increasing the data rate the bus can carry and by using 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:

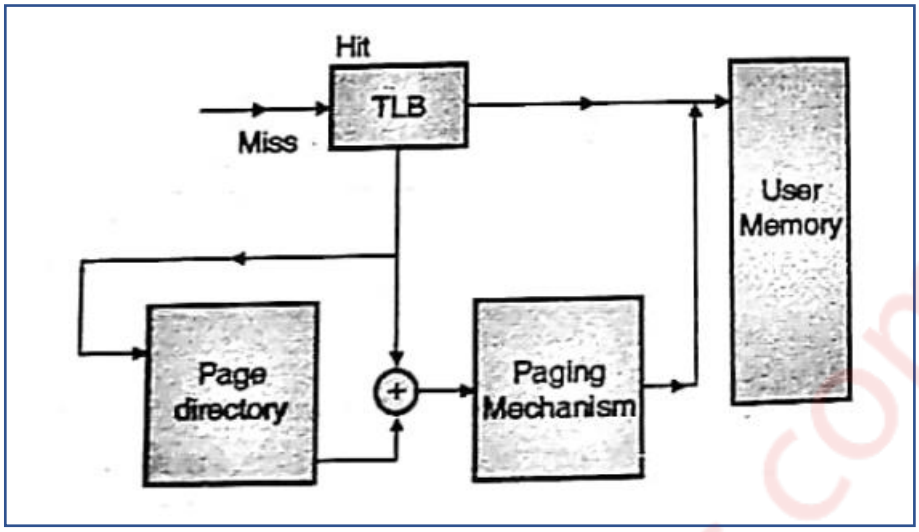
- Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.
- When a process is to be executed, 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.

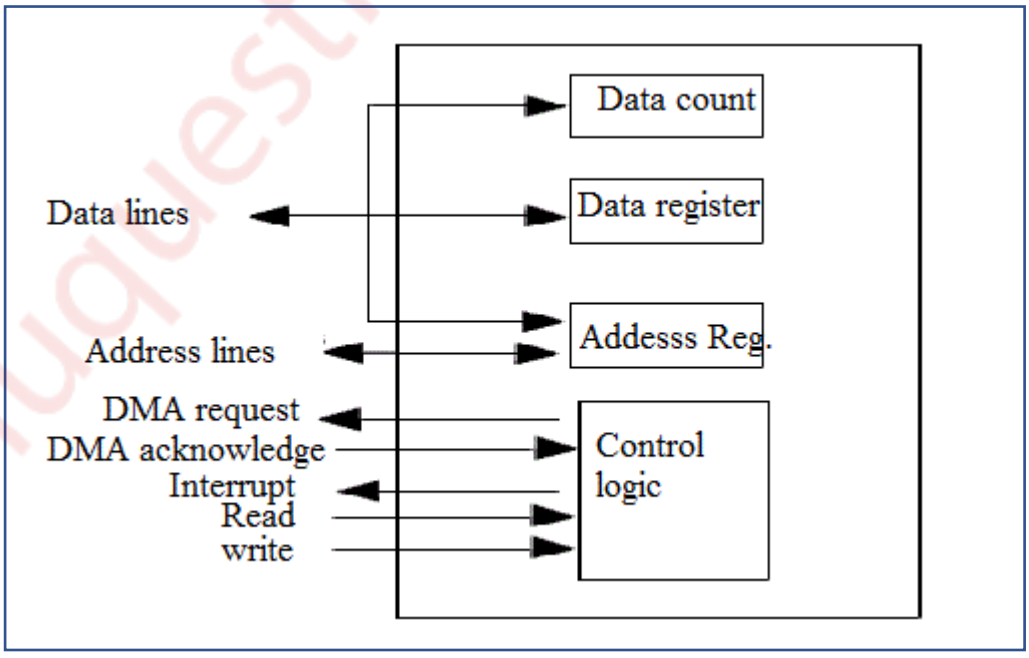
- The interrupt driven I/O mechanism for transferring a block of data.
- After issuing the read command the CPU performs its work, but checks for the interrupt after every instruction cycle.
- When CPU gets an interrupt, it performs the following operation in sequence:
  - Save context i.e. the contents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

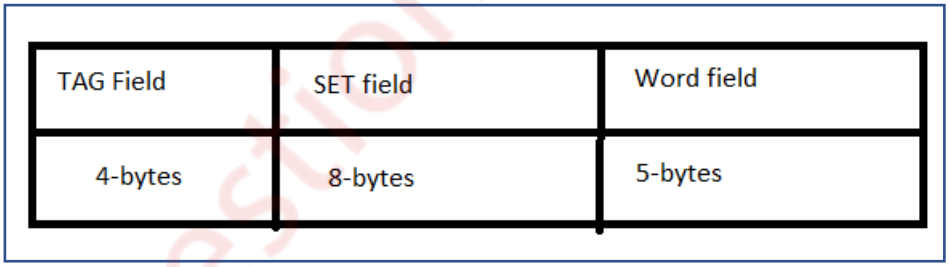
4 + TAG = 12

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.

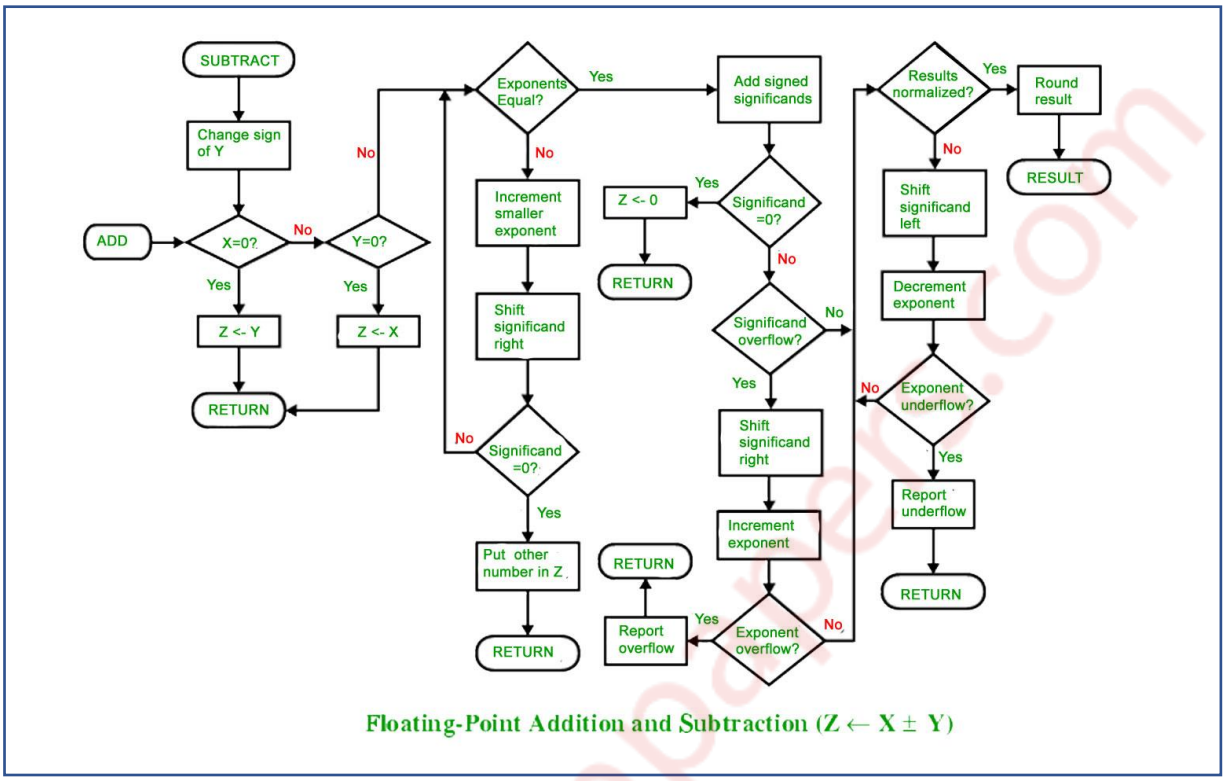
---

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.