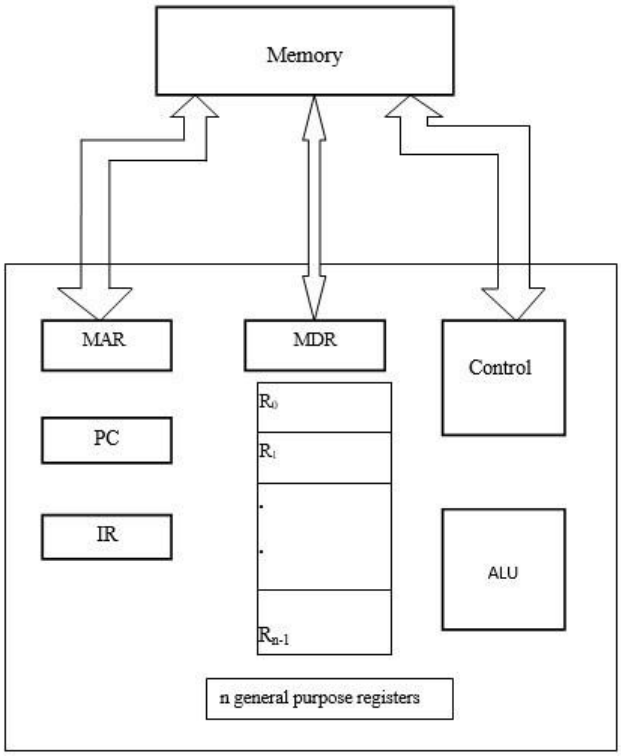
CBCS SCHEME

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

B.E./B.Tech. Degree Examination, Dec.2025/Jan.2026

Computer Organization and Architecture

Answer any FIVE full questions, choosing ONE full Question from each module

		Module 1	M	L	C
Q1	**a**	**With a neat diagram, describe the functional units of a computer.**	**8**	**L2**	**C1**
					

Transfers between memory and processor are started by sending the address of the memory location to be accessed to the memory unit and issuing the appropriate control signals. The data is transferred to or from the memory. The memory and processor connection is shown in above figure.

The I**nstruction register (IR) h**olds the instruction that is currently being executed. Its output is available to control circuits which generate the timing signals that control various processing elements involved in executing the instruction.

The **Program Counter (PC)** holds the address of the next instruction to be fetched and executed. During the execution of an instruction, the contents of the PC are updated to correspond to the address of the next instruction to be executed. MAR and MDR facilitate communication with the memory.

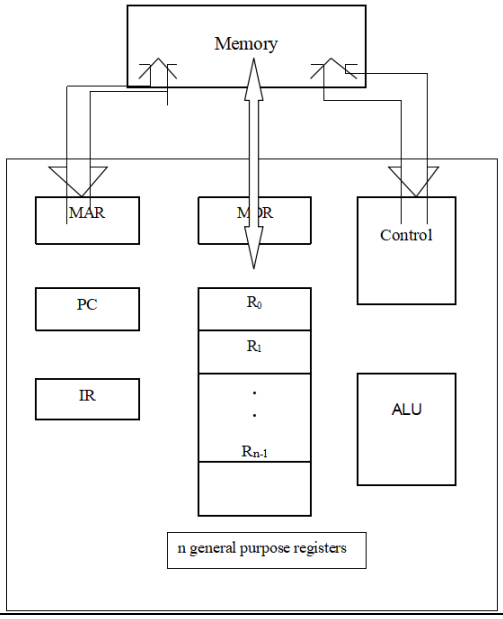
MAR (Memory Address Register) hold the address of the location to be accessed and

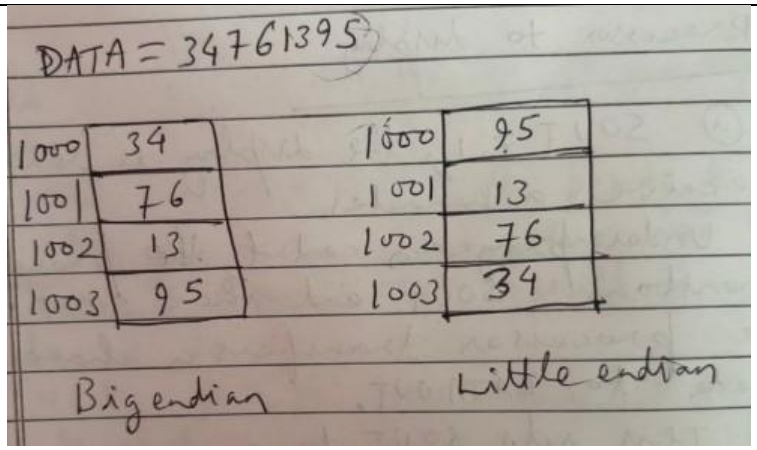
MDR (Memory Data Register) contains data written into or read out of the addressed location.

Typical Operating Steps:

		• Programs reside in the memory through input devices • PC is set to point to the first instruction • The contents of PC are transferred to MAR • A Read signal is sent to the memory • The first instruction is read out and loaded into MDR • The contents of MDR are transferred to IR • Decode and execute the instruction • Get operands for ALU from General- purpose register or from Memory (address to MAR – Read – MDR to ALU) • Perform operation in ALU • Store the result back To general-purpose register or to memory (address to MAR, result to MDR – Write) • During the execution, PC is incremented to the next instruction If some devices require urgent servicing then they raise the interrupt signal interrupting the normal execution of the current program. The processor provides the requested service by executing the appropriate interrupt service routine.			
Q1	**b**	b. Explain following with an example: 6 i) One address instruction ii) Two addrcss instruction iii) Three address instruction	6	L2	C01
		asic Instruction Types: Three-Address Instructions: There are 3-operands or three addresses (labels used to specify the location of data) present along with the Op-code in the instruction. In these instructions at-most only two location can be in memory Eg: Add R1, R2, R3 R3 ← R1 + R2 , Here R1, R2 and R3 are the general purpose registers (GPRs) present in the Proessor Chip. Two-Address Instructions: There are 2-operands or two addresses (labels used to specify the location of data) present along with the Op-code in the instruction. In these instructions at-most only one location can be in memory Eg: Add R1, R2 R2 ← R1 + R2, Here R1 and R2 are the general purpose registers (GPRs) present in the Proessor Chip. One-Address Instructions: There is only 1-operands or 1- address (label used to specify the location of data) present along with the Op-code in the instruction which may be a memory location or any internal Processor Registers (GPRs). Here all operations encoded in the Op-code of Instruction is carried out with respect to the data in Accumulator-register (AC) present inside the Processor with the Accumulator register also acting as the detination register after the operation. Eg: Add M AC ← AC + [M] , Here the M represents any memory Location data specified by memory-address M □			

Q1	c	Write short note on 6 i) Basic Performance equation ii) Clock rate	6	L2	CO1
		The total time required to execute the program is known as *elapse time.* This is a measure of performance of entire computer system. The periods during which processor is active is used to measure the performance of processor. The sum of these periods is referred to as *processor time*. The processor time depends on the hardware involved in the execution of individual machine instructions. Processor circuits are controlled by a timing signal called clock. The clock defines regular time intervals called clock cycles. To execute a machine instruction, the processor divides the action to be performed into a sequence of basic steps, such that each can be completed in one clock cycle. The length P of one clock cycle is an important parameter that affects the processor performance. Its inverse is the clock rate, R= 1/P which is measured in cycles per second (Hz). T – processor time required to execute a program that has been prepared in high-level language N – number of actual machine language instructions needed to complete the execution (note: loop) MOV R1,# 05H AGAIN: DJNZ R1, AGAIN S – the average number of basic steps needed to execute one machine instruction. Each step is completed in one clock cycle R – clock rate Note: these are not independent to each other $T = \frac{N \times S}{R}$			
Q2	a	With a neat diagram, explain basic.operational concept of computer.**Basic Operational Concepts** To perform a given task, an appropriate program consisting of a list of instructions is stored in the memory. Individual instructions are brought from the memory into the processor, which executes the specified operations. Data to be used as operands are also stored in the memory. A typical instruction may be *Add LOCA, R0* This instruction adds the operand at memory location LOCA to the operand in a register in the processor, R0, and places the sum in the register R0. The original contents of location LOCA are preserved whereas those of R0 are overwritten. First the instruction is fetched from the memory into the processor. Next the operand at LOCA is fetched and added to the contents of R0. Finally the resulting sum is stored in register R0. Transfers between memory and processor are started by sending the address of the memory location to be accessed to the memory unit and issuing the appropriate control signals. The data is transferred to or from the memory. The memory and processor connection is shown in Fig 1.2.			

		The Instruction register (IR) holds the instruction that is currently being executed. Its output is available to control circuits which generate the timing signals that control various processing elements involved in executing the instruction. The Program Counter (PC) holds the address of the next instruction to be fetched and executed. During the execution of an instruction, the contents of the PC are updated to correspond to the address of the next instruction to be executed. MAR and MDR facilitate communication with the memory. MAR (Memory Address Register) hold the address of the location to be accessed and MDR (Memory Data Register) contains data written into or read out of the addressed location. If some devices require urgent servicing then they raise the interrupt signal interrupting the normal execution of the current program. The processor provides the requested service by executing the appropriate interrupt service routine. 			
Q2	b	With help ot" example, explain little Endian and Big Endian byte **Big-Endian and Little-Endian Assignments:** **Big-Endian: lower byte addresses are used for storing the most significant bytes of the word** **Little-Endian: opposite ordering. lower byte addresses are used for storing the less significant bytes of the word**	6	L2	C01



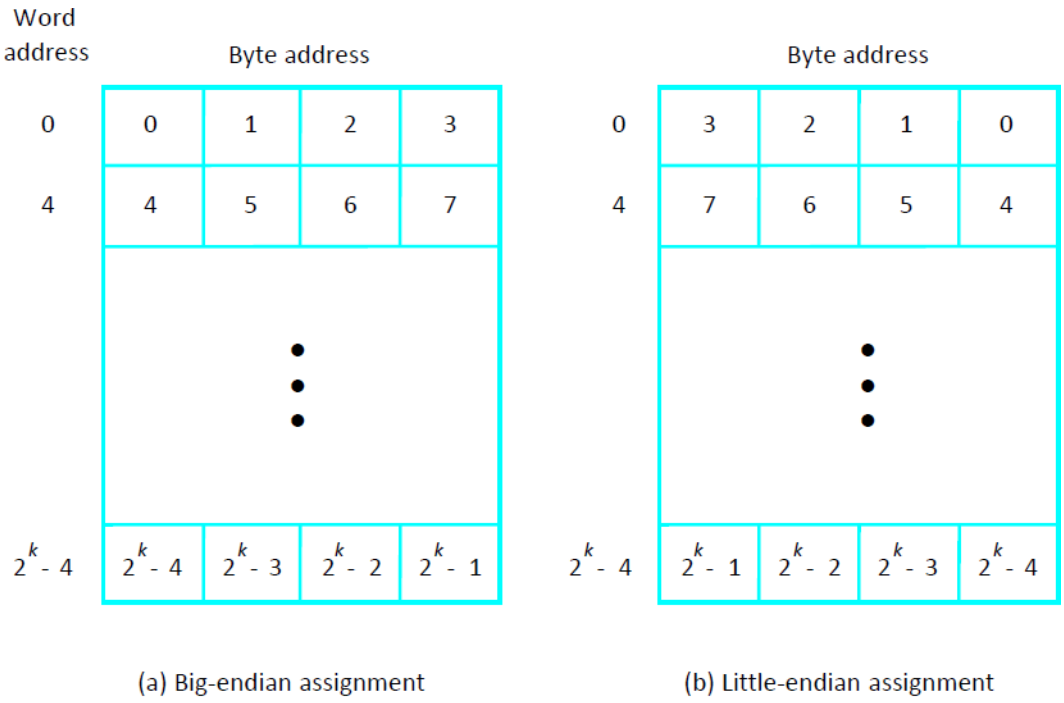


Figure 2.7. Byte and word addressing.

	c	Explain memory operations. Memory Operations Both program instructions and data operands are stored in the memory. To execute an instruction, the processor control circuits must cause the word (or words) containing the instructions to be transferred from the memory to the processor. Operands and results must also be moved between memory and processor. Thus the two basic operations involving memory are needed, namely, Load (or Read or Fetch) and Store (or Write). The Load operation transfers a copy of the contents of a specified memory location to the processor. The memory content remains unchanged. The Store operation transfers an item of information from the processor to a specific memory location destroying the contents of that location.	7	L2	CO1
Q3	a	What are Assembler Directives? Explain various Assembly directives used in ALP. The assembly language allows the programmer to specify other information needed to translate the source program to object program. Suppose the name SUM is used to represent the value 200. This fact may be conveyed to the assembler program through a statement such as SUM EQU 200 This statement does not denote the instruction that will be executed when the object program is	10	L2	CO2

		run. It informs the assembler that the name SUM should be replaced by the value 200 wherever it appears in the program. Such statements are *assembler directives* (or commands) that are used by the assembler when it translates the source program in to a object program. **ORIGIN** is a directive that tells the assembler program where in the memory to place the data block. **DATAWORD** directive is used to inform the assembler to place the data in the address. **RESERVE** directive declares a memory block and does not cause any data to be loaded in these locations.			
Q3	b	Define subroutine and parameter passing. Explain how to pass the parameter by value and by reference. Parameter Passing When calling a subroutine, a program must provide to the subroutine the parameters, that is, the operands or their addresses, to be used in the computation. The subroutine returns other parameters, in this case, the results of the computation. This exchange of information between a calling program and a subroutine is referred to as parameter passing. The parameters may be placed in registers or in memory locations where they can be accessed by the subroutine. The parameters may also be placed on the processor stack used for saving the return address. Passing parameters through processor registers is straightforward and efficient. Fig 2.7 shows a program for adding a list of numbers using a subroutine with parameters passed through registers.	10	L2	C02
4.	a	Define stack, explain PUSH and POP operations on stack with neat diagram. A stack is a list of data elements usually words or bytes with the accessing restriction that elements can be added or removed at one end of the list only. This end is called the top of the stack and the	10	L2	CO2

Calling program

```
Move     N, R1              R1 serves as a counter
Move     #NUM1, R2          R2 points to the list
Call     LISTADD            Call subroutine
Move     R0, SUM            Save result
           •
           •
           •
```

Subroutine

```
LISTADD  Clear      R0          Initialize sum to 0
LOOP     Add        (R2)+, R0   Add entry from list
         Decrement  R1
         Branch > 0 LOOP
         Return                 Return to calling program
```

Fig 2.7: Program written as a subroutine; parameters passed through registers

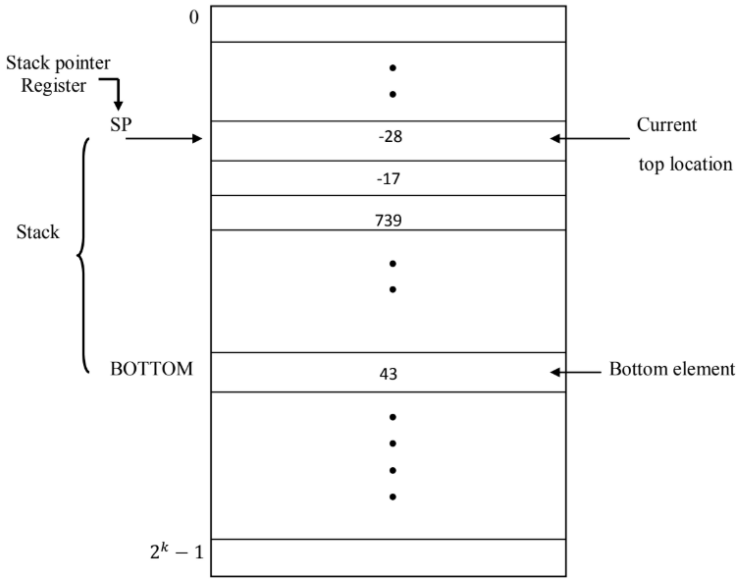
If many parameters are involved, there may not be enough general purpose registers available for passing them to subroutine. Using a stack is highly flexible as stack can handle a large number of parameters. The following example illustrates this approach.

Assume that top of the stack is at level 1 below.

```
Move     #NUM1,  -(SP)     Push parameters onto stack
Move     N, -(SP)
Call     LISTADD           Call subroutine   (top of stack at level 2)

Move     4(SP), SUM        Save result.
Add      #8, SP            Restore top of stack    (top of stack at level 1)
           •
           •
           •

LISTADD  MoveMultiply      R0-R2, -(SP)      Save registers   (top of stack at level 3)

         Move              16(SP), R1        Initialize counter to n.
         Move              20(SP), R2        Initialize pointer to the list.
         Clear             R0                Initialize sum to 0.
LOOP     Add               (R2) +, R0        Add entry from list
         Decrement         R1
         Branch>0          LOOP
         Move              R0, 20(SP)        Put result on the stack
         MoveMultiply      (SP) +, R0-R2     Restore registers
         Return                              Return to calling program
```

other end is called the bottom. The structure is called pushdown stack. Last-in-first-out (LIFO) stack is a type of storage mechanism where the last data item placed on the stack is the first one removed when retrieval begins. The terms push and pop are used to describe placing a new item on the stack and removing top item from the stack. The stack grows in the direction of decreasing memory address.

Fig 4.1 shows a stack of word data items in the memory of a computer. It contains the numerical values, with 43 at the bottom and -28 at the top. A processor register is used to keep track of the address of the element of the stack that is at the top at any given time. This register is called the stack-pointer (SP).



Assuming a byte addressable memory with 32 bit word length the Push operation can be implemented as

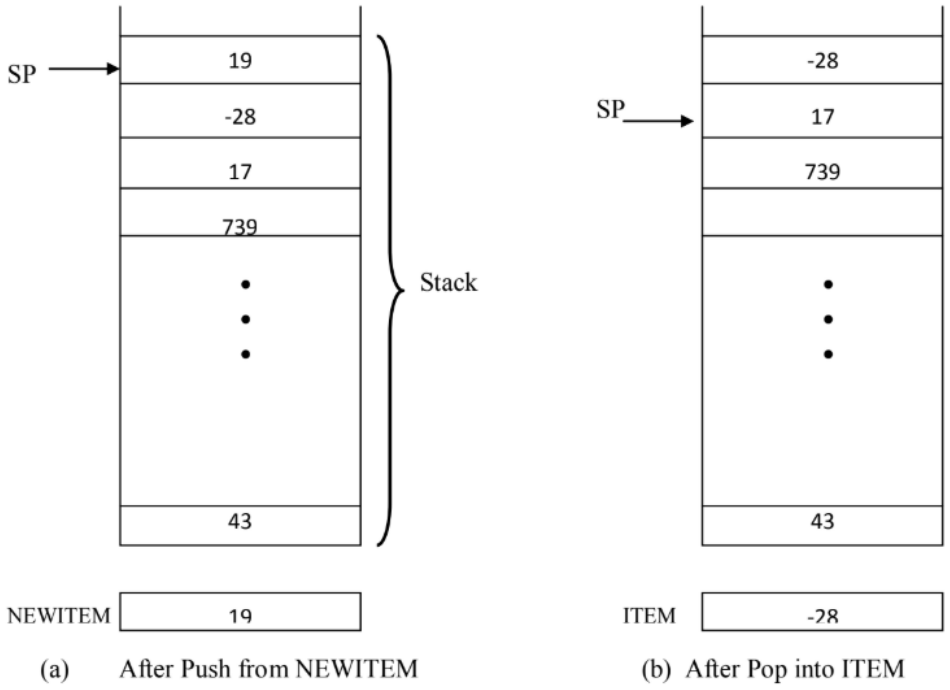
Subtract #4, SP
Move NEWITEM, (SP)

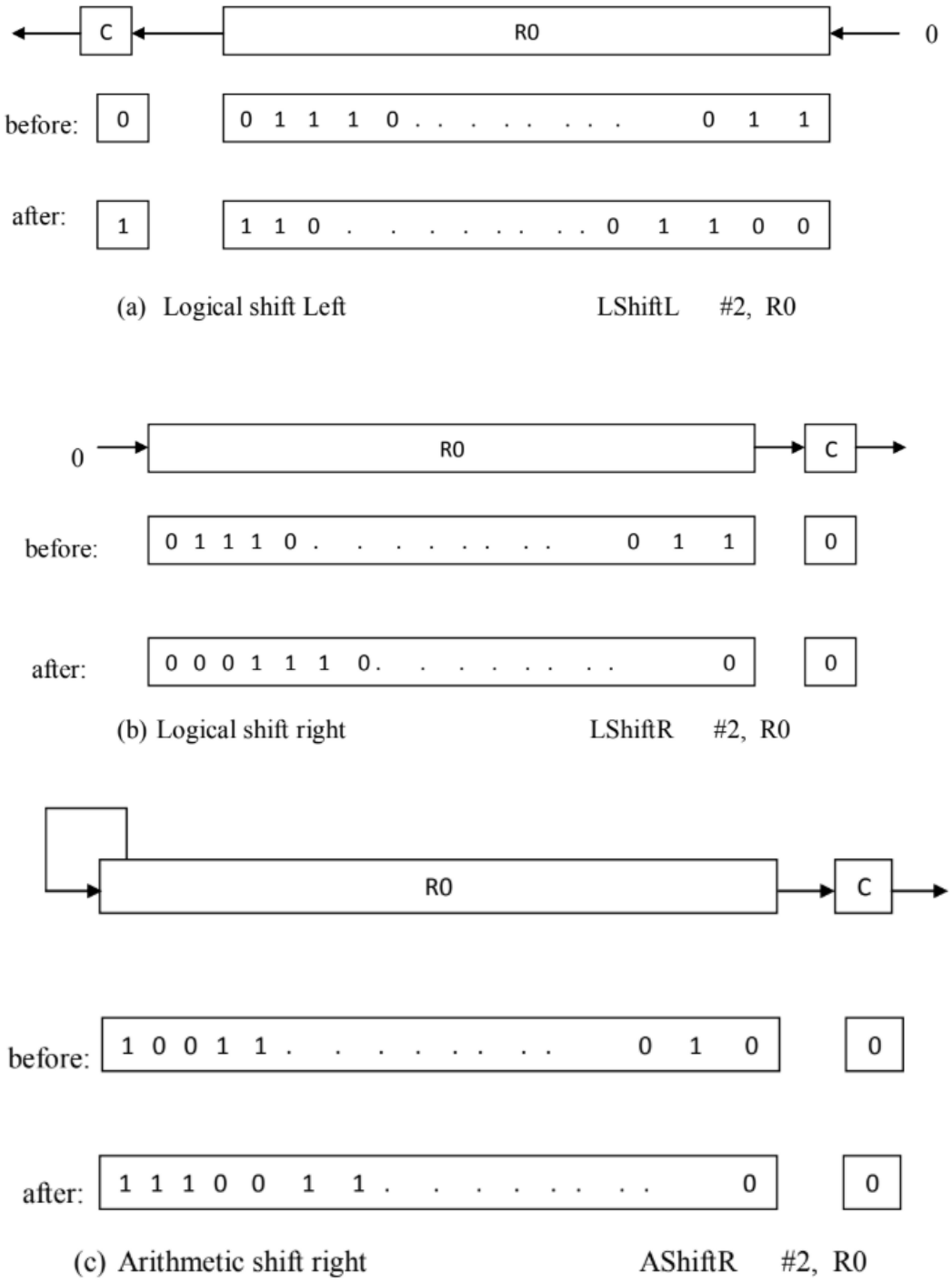
These two instructions move word from location NEWITEM onto the top of the stack, decrementing the stack pointer by 4 before the move.

The Pop operation can be implemented as

Move (SP), ITEM
Add #4, SP

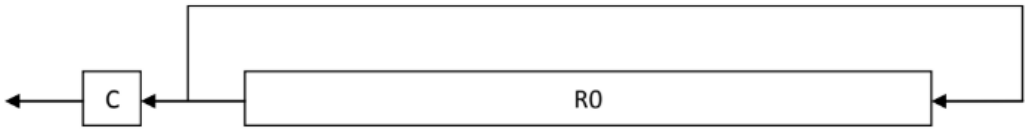
These two instructions move the top value from the stack into location ITEM and then increment the stack pointer by 4 so that it points to the new top element. Below fig shows the effect of each of these operations on the stack.

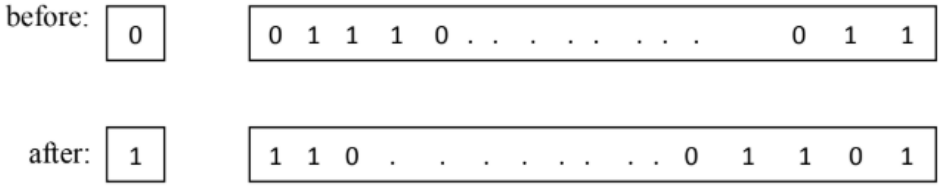
					
4	b	Explain shift and rotate operations with examples. **Logical Shifts** Two logical shift instructions are needed, one for shifting left (LShiftL) and another for shifting right (LShiftR). These instructions shift an operand over a number of bit positions specified in a count operand contained in the instruction. The general form of logical left shift instruction is **LShiftL count, dst** The count operand may be given as an immediate operand or it may be contained in the processor register. Vacated positions are filled with zeros, and the bits shifted out are passed through the Carry flag C, and then dropped. Involving the C flag in shifts is useful in arithmetic operations on large numbers that occupy more than one word.	10	L2	CO3



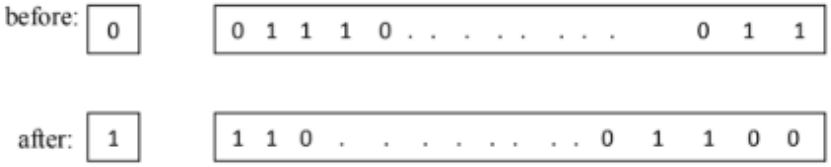
Rotate Operations

To preserve all the bits, a set of rotate can be used. They move the bits that are shifted out of one end of the operand back in to the other end. Two versions of both left and right rotate instructions are provided. In one version, bits are of the operand are simply rotated. In the other version, the rotation includes the C flag. Figure 2.11 shows the left and right rotate operations with and without C flag being include in the rotation. Note that when C flag is not included in the rotation, it still retains the last bit shifted out of the end of the register.

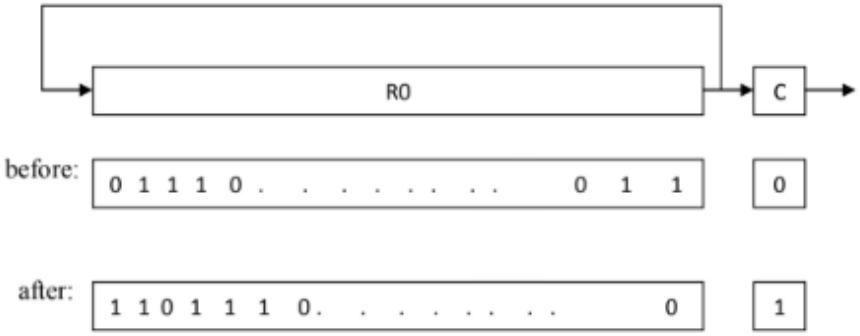




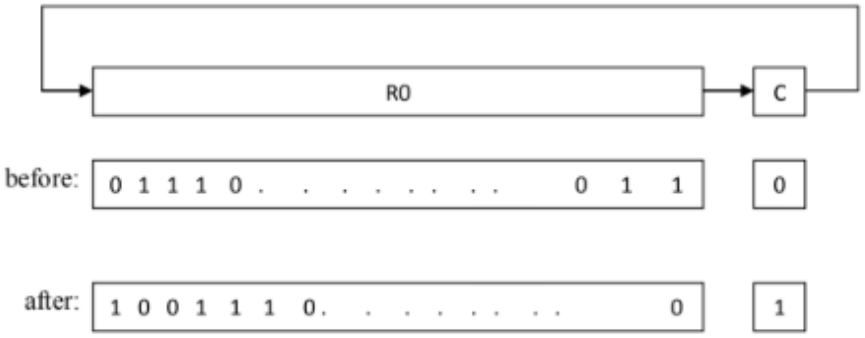
(a) Rotate left without carry RotateL #2, R0



(b) Rotate left with carry RotateLC #2, R0

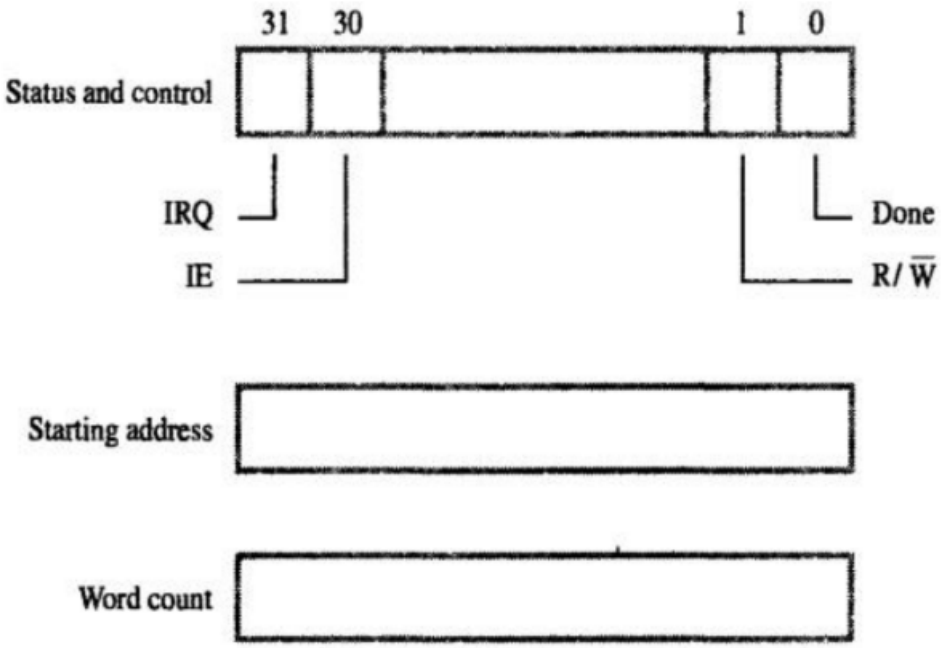


(c) Rotate right without carry RotateR #2, R0



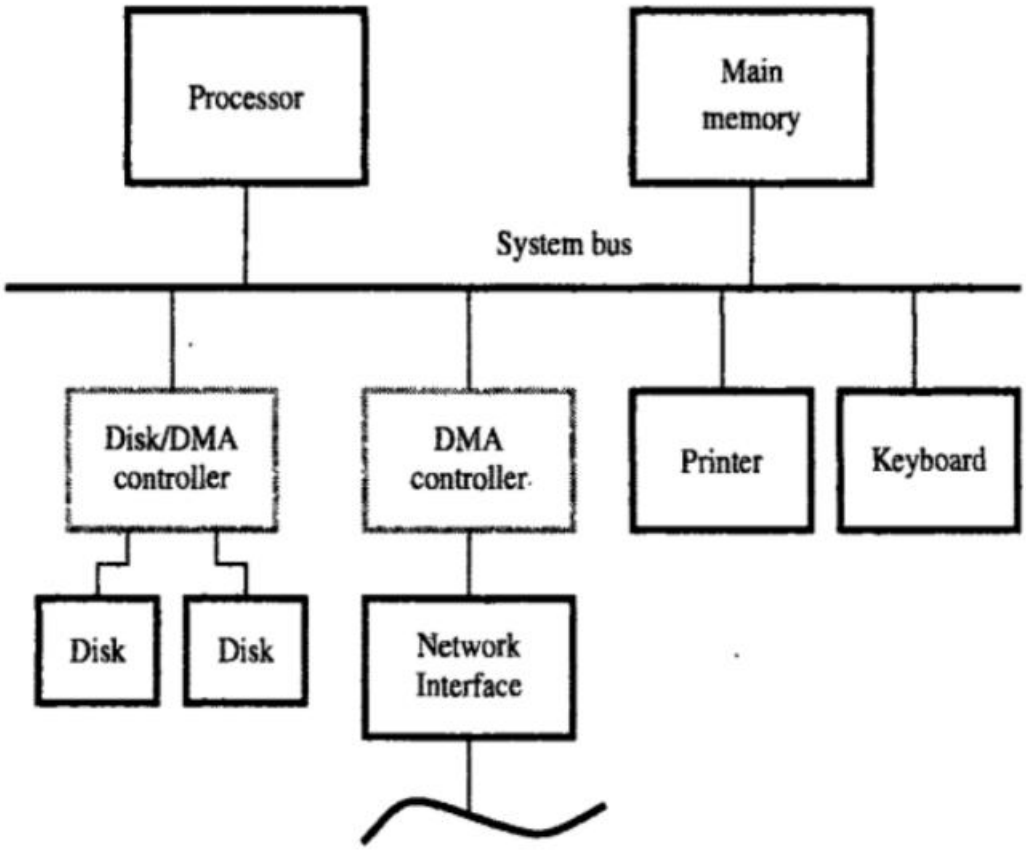
(d) Rotate right with carry RotateRC #2, R0

5	a	Define Interrupt. Explain the various ways of enabling and disabling interrupts An interrupt is a hardware or software signal that temporarily halts the CPU's current execution to handle a higher-priority event. When the device activates interrupt request signal, it keeps this signal activated until it learns that the processor has accepted its request. It is essential to ensure that this active signal does not lead to successive interruptions causing the system to enter an infinite loop from which it cannot recover. There are three ways to avoid this problem. 1. The first option is that processor must ignore the interrupt request line until the execution of first instruction of interrupt service routine (ISR) has been completed. Then by using the Interrupt disable instruction as the first instruction in the ISR, programmer can ensure that no further interruption will occur. 2. The second option is to have processor automatically disable the interrupts before staring the execution of Interrupt service routine. After saving the contents of PS on the stack with Interrupt enable bit equal to 1, the processor clears the Interrupt enable bit in PS register thus disabling further interrupts. When a return from interrupt instruction is executed, the contents of PS are restored from the stack setting the Interrupt enable bit back to 1. Hence interrupts are again enabled. 3. In the third option, processor has a special interrupt request line for which the interrupt handling circuit responds only to leading edge of the signal. Such a line is said to edge triggered. The sequences of events involved in handling the interrupt request from a single device are as follows. 1. The device raises an interrupt request. 2. The processor interrupts the program currently being executed. 3. Interrupts are disabled by changing the control bits in the PS register. 4. The device is informed that the request has been recognized and in response deactivates the interrupt-request signal. 5. The action requested by the interrupt is being performed by the interrupt service routine. 6. Interrupts are enabled and the execution of interrupted program is being resumed.	10	L2	CO3
5	b	Explain DMA technique and its importance.	10	L2	CO3



To transfer large blocks of data at high speeds, an alternate approach is used. A special control unit may be provided to allow transfer of block of data directly between external device and main memory without intervention by processor. This approach is called direct memory access or DMA.
DMA transfers are performed by control circuits that are part of I/O interface called DMA controller. The DMA controller performs functions that would normally be carried out by processor when accessing main memory.

The R/ bit determine the direction of transfer. When this bit is set to 1 by a program instruction, the controller performs read operation that is it transfers data from memory to I/O device. When transfer is complete, it sets done flag to 1. When IE is1, it causes the controller to raise an interrupt after it has completed transferring block of data. Finally, IRQ bit is set to 1 when it has requested interrupt.
Requests from DMA devices are given high priority than processor requests. Among different DMA devices high priority is given to high speed peripherals such as disks, high speed network interface or graphic display device. The processor originates most memory cycles, the DMA controller is said to steal memory cycles from processor. This technique is called cycle stealing. DMA controller is given access to main memory to transfer a block of data without interruption. This is called as block or burst mode.

					
6	a	Explain Different ways of handling multiple devices The information needed to determine whether a device is requesting an interrupt is available in its status register. When a device raises an interrupt request, one of the bits of the status register is set to 1 which we call IRQ bit. KIRQ, DIRQ are the interrupt request bits for keyboard and display. The polling scheme has the disadvantage that the time spent interrogating the IRQ bits of all the devices that may not be requesting any service. An alternate approach is to use vectored interrupts. **Vectored Interrupts** A device requesting an interrupt can identify itself by sending special code to the processor over the bus. The code supplied by the device represents the starting address of the interrupt service routine. The code length is 4 to 8 bits. The processor reads this address called the interrupt vector and stores it in to the PC. The interrupt vector may also include a new value for a processor status register. The interrupted device must wait to put on the bus only when the processor is ready to receive it. When the processor is ready to receive the vector interrupt code, it activates the interrupt acknowledge line INTA. The I/O device responds by sending its interrupt vector code and turning off INTR signal.	10	L2	CO3
6	b	Illustrate Interrupt priority scheme with neat diagram If several devices share one interrupt request line, some other mechanism is needed. When several devices raises interrupt request and line is activated, the processor responds by setting the INTA line to 1.	10	L2	CO3

The signal is received by device 1. Device 1 passes the signal onto device 2 only if it does not require any service. If device 1 has pending request for interrupt, it blocks the INTA signal and proceeds to put its identification code on to data lines. In daisy chain the device that is electrically closest to the processor has the highest priority.

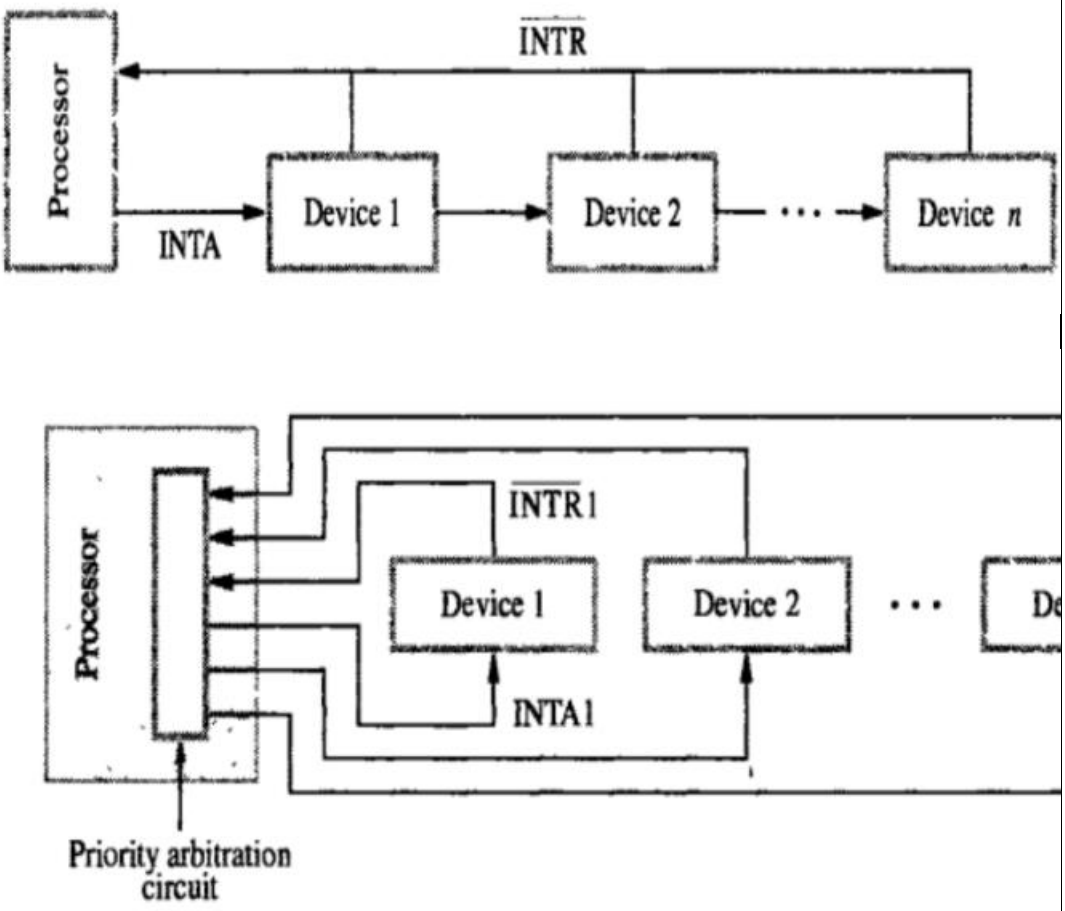
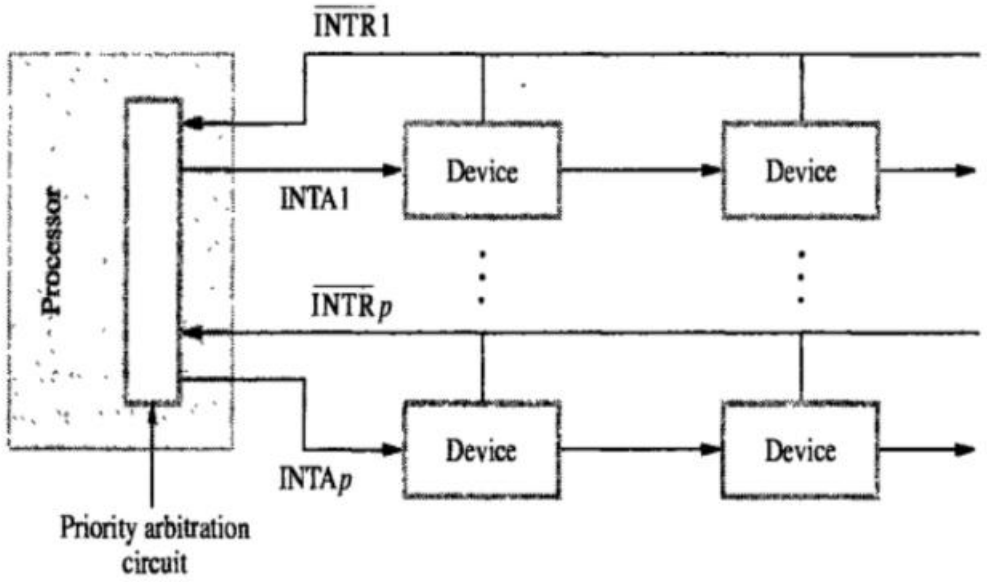


Fig 6.1: Implementation of interrupt priority using individual interrupt request & acknowledge
lines

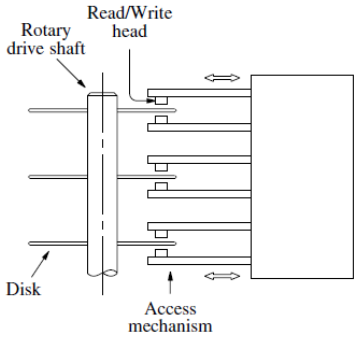
Devices can be organized in groups and each group is connected at a different priority
level. Within group devices are connected in daisy chain.



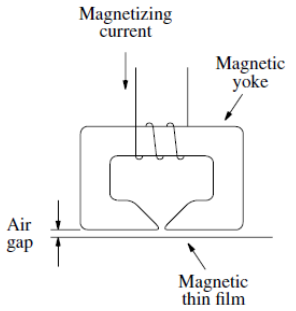
7	a	Explain the principle of working of magnetic disc	10	L2	CO4

Magnetic Hard Disks

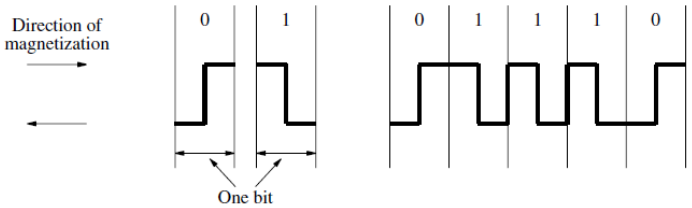
- The storage medium in a magnetic-disk system consists of one or more disks mounted on a common spindle.
- A thin magnetic film is deposited on each disk, usually on both sides.
- The disks are placed in a rotary drive so that the magnetized surfaces move in close proximity to read/write heads, as shown in Figure a.
- The disks rotate at a uniform speed.
- Each head consists of a magnetic yoke and a magnetizing coil, as indicated in Figure b.



(a) Mechanical structure



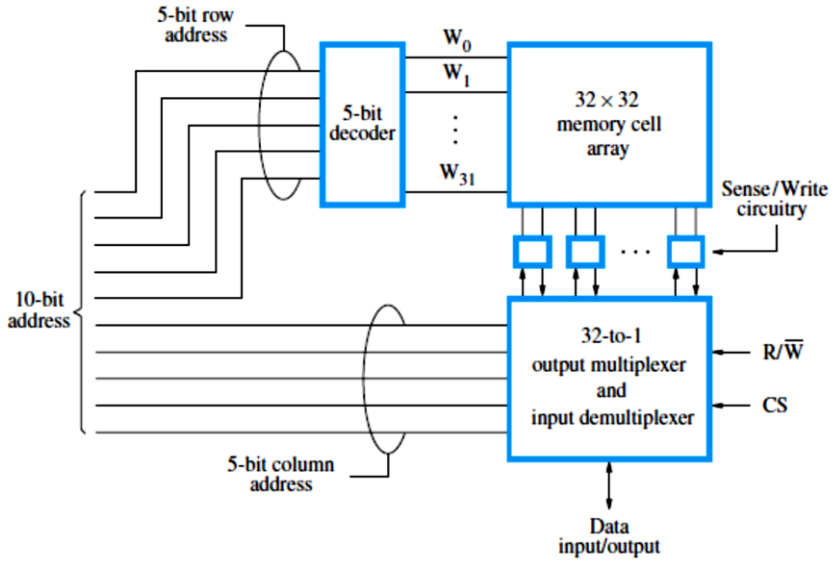
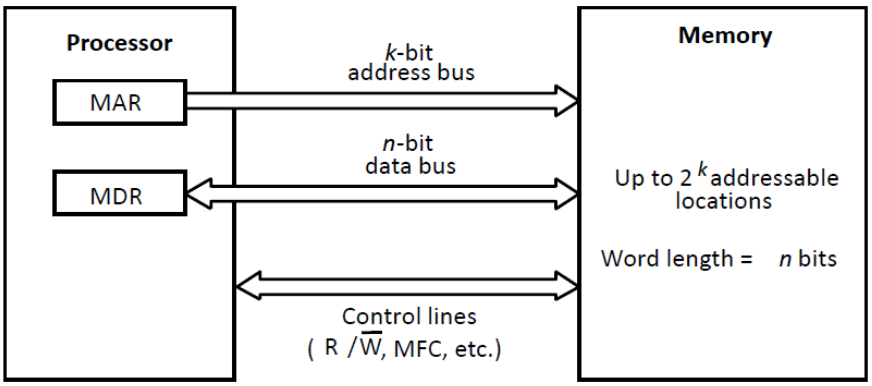
(b) Read/Write head detail

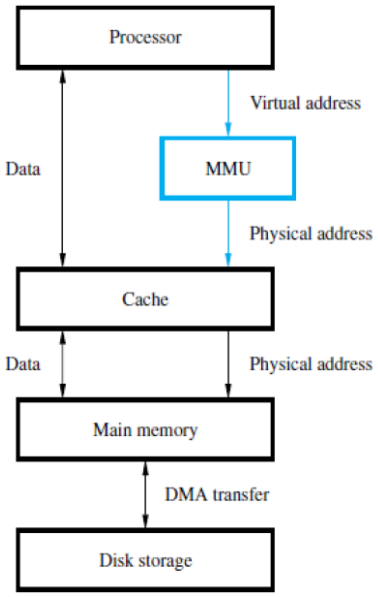


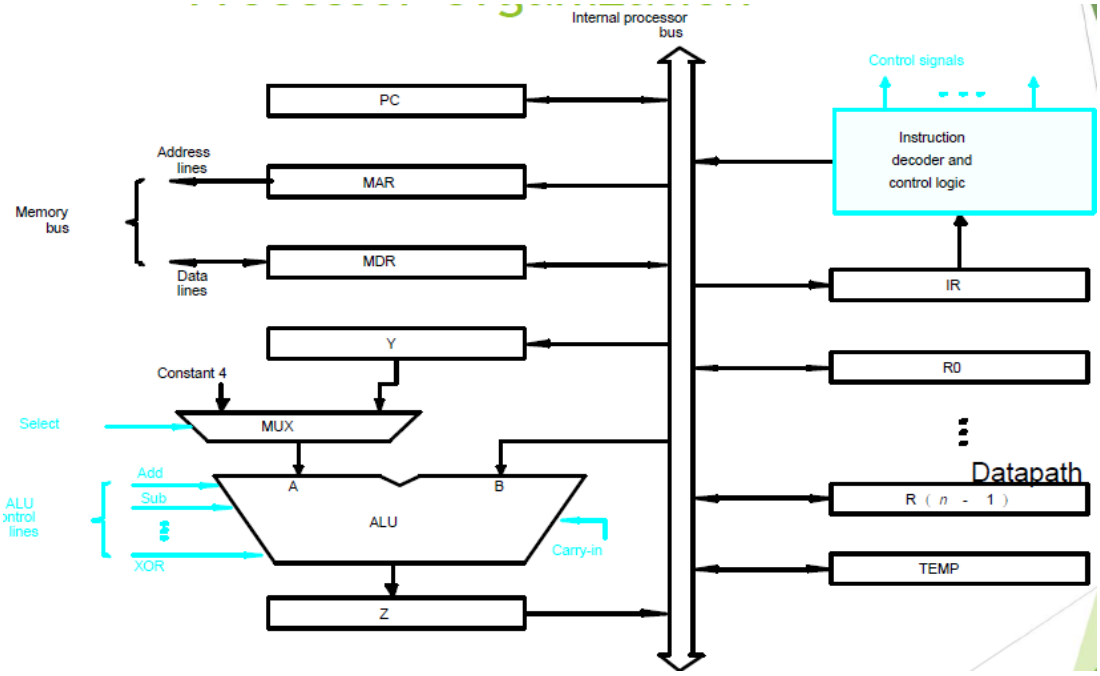
(c) Bit representation by phase encoding

Magnetic Disc Principles

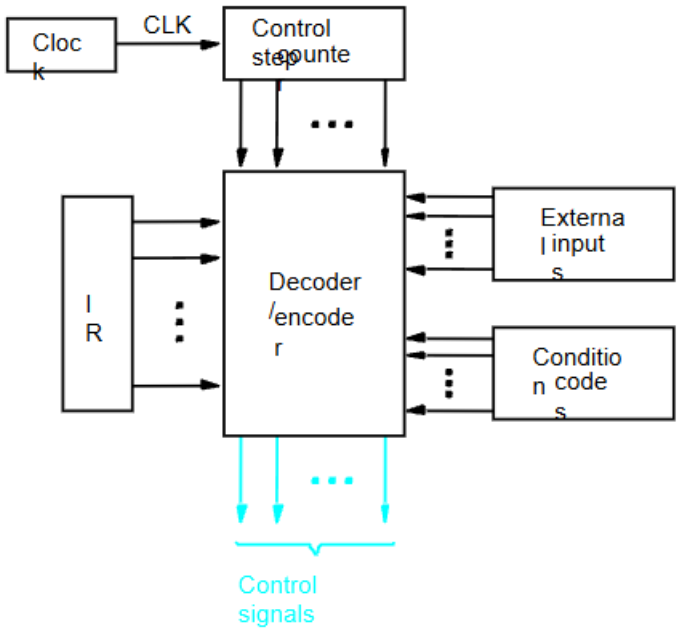
		➢ Digital information can be stored on the magnetic film by applying current pulses of suitable polarity to the magnetizing coil. ➢ This causes the magnetization of the film in the area immediately underneath the head to switch to a direction parallel to the applied field. ➢ The same head can be used for reading the stored information. ➢ In this case, changes in the magnetic field in the vicinity of the head caused by the movement of the film relative to the yoke induce a voltage in the coil, which now serves as a sense coil. ➢ The polarity of this voltage is monitored by the control circuitry to determine the state of magnetization of the film. ➢ Only changes in the magnetic field under the head can be sensed during the Read operation. ➢ Therefore, if the binary states 0 and 1 are represented by two opposite states of magnetization, a voltage is induced in the head only at 0-to-1 and at 1-to-0 transitions in the bit stream. ➢ A long string of 0s or 1s causes an induced voltage only at the beginning and end of the string. ➢ To determine the number of consecutive 0s or 1s stored, a clock must provide information for synchronization. ➢ In some early designs, a clock was stored on a separate track, where a change in magnetization is forced for each bit period. ➢ Using the clock signal as a reference, the data stored on other tracks can be read correctly. ➢ The modern approach is to combine the clocking information with the data (*self-clocking schemes*). ➢ One simple scheme, depicted in Figure c, is known as *phase encoding* or *Manchester encoding*. ➢ In this scheme, changes in magnetization occur for each data bit, as shown in the figure. ➢ A change in magnetization is guaranteed at the midpoint of each bit period, thus providing the clocking information. ➢ The drawback of Manchester encoding is its poor bit-storage density. ➢ The space required to represent each bit must be large enough to accommodate two changes in magnetization. ➢ Read/write heads must be maintained at a very small distance from the moving disk surfaces in order to achieve high bit densities and reliable read/write operations. ➢ The flexible spring connection between the head and its arm mounting permits the head to fly at the desired distance away from the surface in spite of any small variations in the flatness of the surface. ➢ In most modern disk units, the disks and the read/write heads are placed in a sealed, air-filtered enclosure. ➢ This approach is known as *Winchester technology*.			
7	b	Explain the internal organization of 1k*1 dynamic memory chip with neat diagrams	10	L2	CO4
		➢ Practically an1k*1 dynamic memory chip consists of 1024 memory cells. ➢ In this case, a 10-bit address is needed, but there is only one data line, resulting in 15 external			

		connections.  ➢ Figure shows such an organization. ➢ The required 10-bit address is divided into two groups of 5 bits each to form the row and column addresses for the cell array. ➢ A row address selects a row of 32 cells, all of which are accessed in parallel. ➢ But, only one of these cells is connected to the external data line, based on the column address.			
8	a	Explain the interfacing of main memory to the processor	10	L2	CO4
		 ➢ The processor reads data from the memory by loading the address of the required memory location into the MAR register ➢ The R/ $\overline{W}$ line is set to 1. ➢ The memory responds by placing the data from the addressed location onto the data lines, and confirms this action by asserting the MFC signal. ➢ Upon receipt of the MFC signal, the processor loads the data on the data lines into the MDR register. ➢ The processor writes data into a memory location by loading the address of this location into MAR and loading the data into MDR. ➢ The R/ $\overline{W}$ line is set to 0. ➢ If read or write operations involve consecutive address locations in the main memory, then a "block transfer" operation can be performed. ➢ The only address sent to the memory is the one that identifies the first location. ➢ Memory accesses may be synchronized using a clock, or they may be controlled using special			

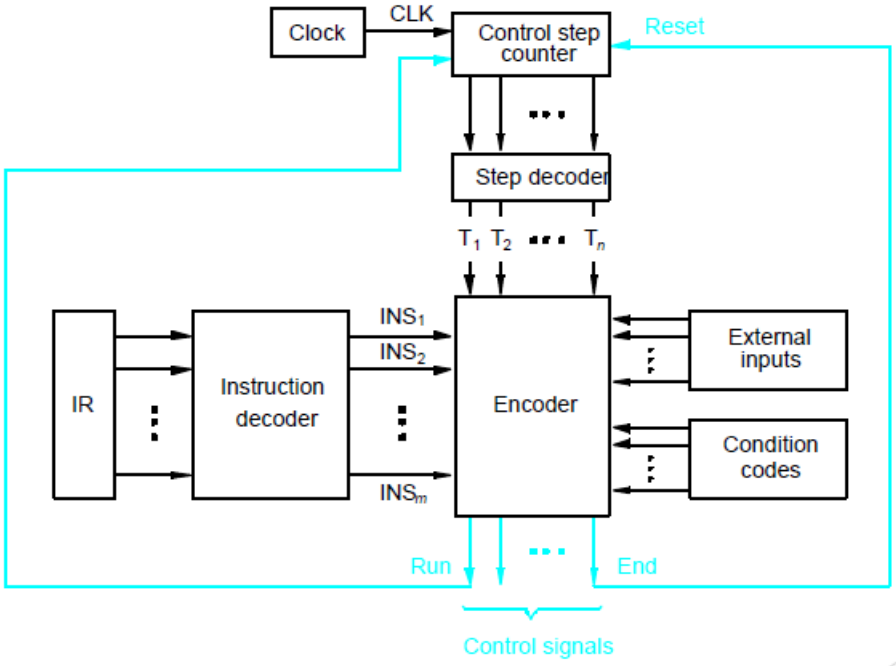
		signals that control transfers on the bus.			
8	b	Explain Virtual Memory Organization with a neat diagram	10	L2	CO4
		Virtual Memories ➢ Recall that an important challenge in the design of a computer system is to provide a large, fast memory system at an affordable cost. ➢ Architectural solutions to increase the effective speed and size of the memory system. ➢ Cache memories were developed to increase the effective speed of the memory system. ➢ Virtual memory is an architectural solution to increase the effective size of the memory system. ➢ In most modern computer systems, the physical main memory is not as large as the address space spanned by an address issued by the processor. ➢ For example, a processor that issues 32-bit addresses has an addressable space of 4G bytes. ➢ The size of the main memory in a typical computer range from a few hundred megabytes to 1G bytes. ➢ When a program does not completely fit into the main memory, the parts of it not currently being executed are stored on secondary storage devices, such as magnetic disks. ➢ All parts of a program that are eventually executed are first brought into the main memory. ➢ When a new segment of a program is to be moved into a full memory, it must replace another segment already in the memory. ➢ In modern computers, the operating system moves programs and data automatically between the main memory and secondary storage. ➢ Thus, the application programmer does not need to be aware of limitations imposed by the available main memory. ➢ Techniques that automatically move program and data blocks into the physical main memory when they are required for execution are called *virtual-memory* techniques. ➢ Programs, and hence the processor, reference an instruction and data space that is independent of the available physical main memory space. ➢ The binary addresses that the processor issues for either instructions or data are called *virtual* or *logical addresses*. ➢ These addresses are translated into physical addresses by a combination of hardware and software components. ➢ If a virtual address refers to a part of the program or data space that is currently in the physical memory, then the contents of the appropriate location in the main memory are accessed immediately. ➢ If the referenced address is not in the main memory, its contents must be brought into a suitable location in the memory before they can be used.  **Virtual Memory Organization**			

		➢ Figure above shows a typical organization that implements virtual memory. ➢ A special hardware unit, called the *Memory Management Unit* (MMU), translates virtual addresses into physical addresses. ➢ When the desired data (or instructions) are in the main memory, these data are fetched. ➢ If the data are not in the main memory, the MMU causes the operating system to bring the data into the memory from the disk. ➢ Transfer of data between the disk and the main memory is performed using the DMA scheme.			
9	a	Explain single bus organization of the datapath inside a processor with neat diagram.  Data Path: The registers, ALU, and interconnecting bus are collectively called the data path The arithmetic and logic unit (ALU) and all the registers are interconnected via a single common bus. This bus is internal to the processor ALU Registers for temporary storage Various digital circuits for executing different micro-operations (gates, MUX, decoders, counters) Internal path for movement of data between ALU and registers Driver circuits for transmitting signals to external units Receiver circuits for incoming signals from external units PC: Keeps track of execution of a program. Contains the memory address of the next instruction to be fetched and executed. MAR: Holds the address of the location to be accessed. I/P of MAR is connected to Internal bus and an O/p to external bus. MDR: Contains data to be written into or read out of the addressed location.	10	L2	CO5

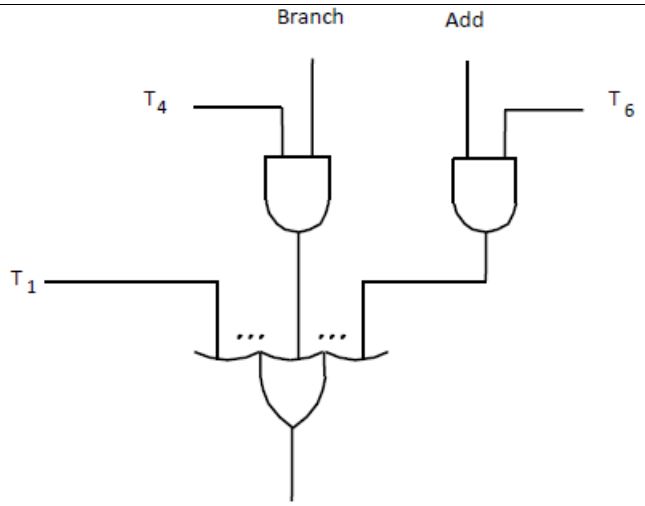
		It has 2 inputs and 2 Outputs. Data can be loaded into MDR either from memory bus or from internal processor bus The data and address lines are connected to the internal bus via MDR and MAR.			
9	b	Discuss hardwired control unit organization with relevant diagram.	10	L2	CO5



To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence.
Hardwired system can operate at high speed; but with little flexibility.

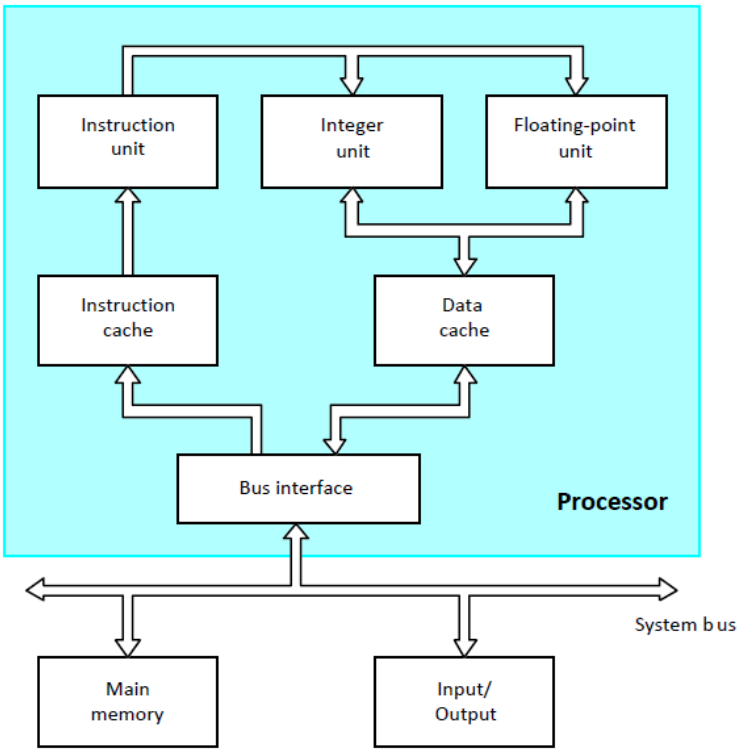


Zin=T1+T6•ADD+T4•BR+…



10	a	Explain complete processor with a neat diagram	10	L2	CO5

A Complete Processor:



- A complete processor can be designed using the structure shown in Figure above
- This structure has an instruction unit that fetches instructions from an instruction cache or from the main memory when the desired instructions are not already in the cache.
- It has separate processing units to deal with integer data and floating-point data.
- A data cache is inserted between these units and the main memory.
- A single cache can be used to store both instructions and data or separate caches can be used for instructions and data.
- The processor is connected to the system bus and, hence, to the rest of the computer, by means of a bus interface.
- A processor may include several integer or floating-point units to increase the potential for concurrent operations.

10	b	Develop the complete control sequence for the execution of the instruction: Add (R_3), R1	10	L5	CO5

Consider the given instruction Add (R3), R1
Executing this instruction requires the following actions:
1.Fetch the instruction
2.Fetch the first operand (the contents of the memory location pointed to by R3)
3.Perform the addition
4.Load the result into R1

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Figure 7.6 Control sequence for execution of the instruction Add (R3),R1.

- Figure 7.6 gives the sequence of control steps required to perform these operations for the single-bus architecture.
- Steps 1 through 3 constitute the instruction fetch phase,
- This is the same for all instructions.
- The instruction decoding circuit interprets the contents of the IR at the beginning of step 4.
- This enables the control circuitry to activate the control signals for steps 4 through 7, which constitute the execution phase.