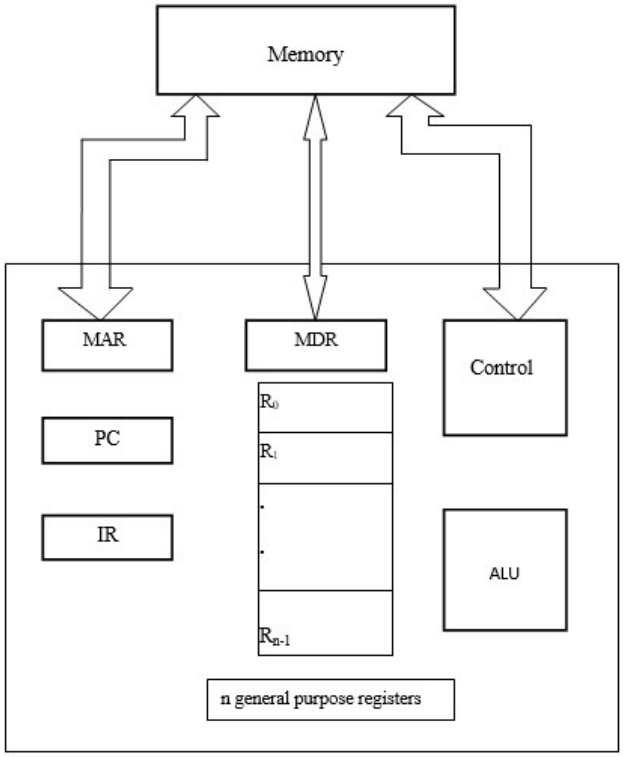
USN										

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

Internal Assessment Test 1 – October 2025

Sl.	Answer any FIVE FULL Questions	Marks	CO	RBT
1	Explain the operation of a computer with a neat block diagram. Also, discuss the operation concepts in a computer, highlighting the role of PC, MAR, MDR & IR.	10	CO1	L2

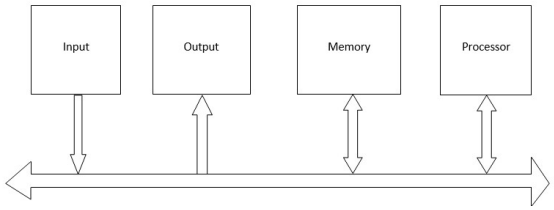


Transfers between memory and processor are started by sending the address of the memory location to be accessed to the memory unit and issuing the appropriate control signals. The data is transferred to or from the memory. The memory and processor connection is shown in above figure.

The 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. 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.			
2	i. Define Bus. Explain a single bus structure with a neat block diagram. When a word of data is transferred between units, the bits are transferred simultaneously over many wires, or lines, one pipeline **A group of lines that serves as a connecting path for several devices is called a *bus*.** In addition to the lines that carry the data, the bus must have lines for address and control purposes. The simplest way to interconnect functional units is to use a *single bus*. The bus can be used for only one transfer at a time, only two units can actively use the bus at any given time.  	03+07	CO1 `	L2

ii. List functions of system software and explain how the processor is shared between user program and the OS routine.

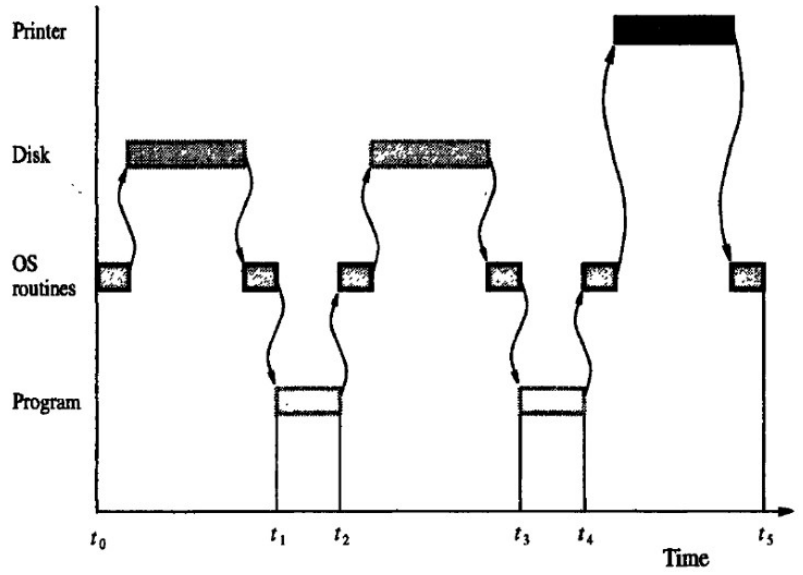
System software is a collection of programs that are executed as needed to perform functions such as:

- Receiving and interpreting user commands.
- Entering and editing application programs and storing them as files in secondary storage devices.
- Managing the storage and retrieval of files in secondary storage devices.
- Running standard application programs such as word processors, spreadsheets or games with data supplied by users.
- Controlling I/O units to receive input information and produce output results.
- Translating programs from source form prepared by user into object form consisting of machine instructions.

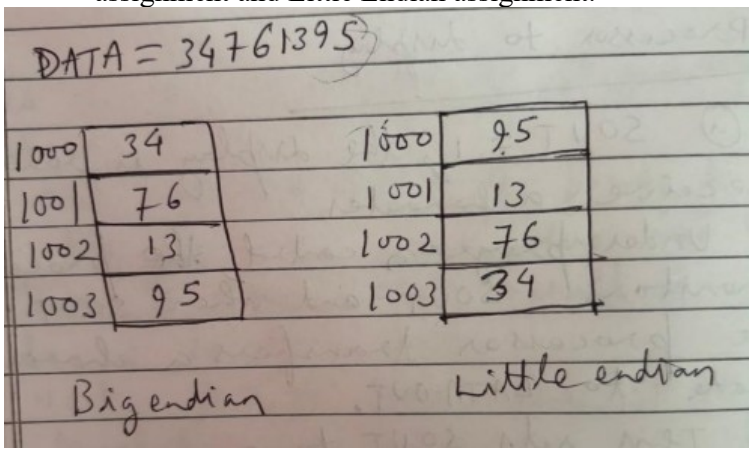
Linking and running user-written application programs with existing standard library routines such as numerical computational packages,

OS is one type of system software.

The execution control passes back and forth between application program and OS routines. This sharing of processor execution time is illustrated by a time line diagram as shown in Fig 1.5. During the time period t_(0)to t_(1), an OS routine initiates the application loading program from the disk to the memory, waits until the transfer is completed, and then passes execution control to the application program. A similar pattern of activity occurs during period t_(2) to t_(3) and period t_(4) to t_(5), when the operating system transfers the data file from the disk and print the results. At t_(5), the operating system may load and execute another application program. Notice that the disk and processor are idle during most of the time period t_(4) to t_(5). The operating system manages the concurrent execution of several application programs to make best possible use of computer resources. This pattern of concurrent execution is called multiprogramming or multitasking.



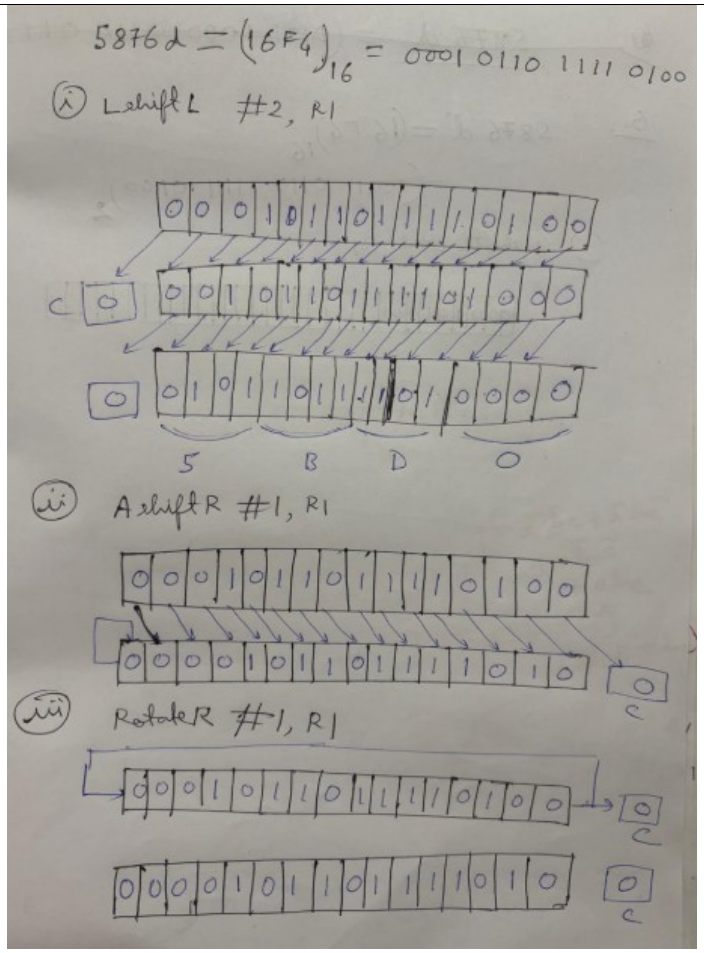
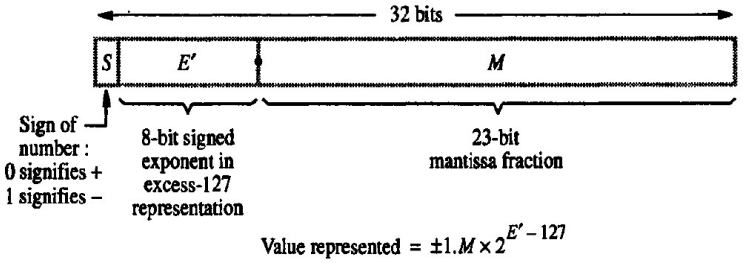
CI CCI HOD

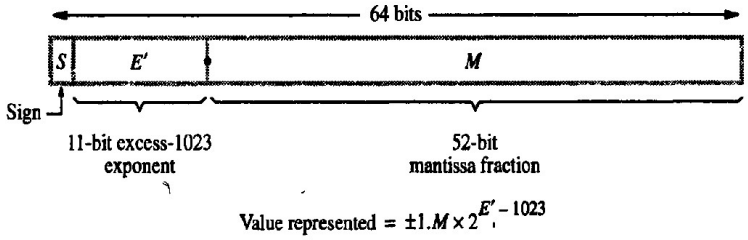
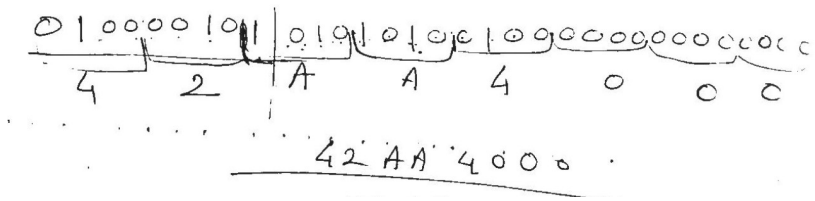
3.	i. Explain the various parameters affecting the performance of a computer and also provide the basic performance equation The total time required to execute the program is known as *elapse time.* This is a measure of performance of entire computer system. The periods during which processor is active 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 $T = \frac{N \times S}{R}$ ii. Show how the number 34761395 is stored using Big Endian assignment and Little Endian assignment.  	04+06	CO1	L2

4	Discuss the following addressing modes with examples:
	i. Immediate
	ii. Register
	iii. Indirect
	iv. Direct
	v. Index
	
	i. Immediate mode:
	Address and data constants can be represented in assembly language using the Immediate mode addressing where the operand is given explicitly in the instruction. For example, the instruction
	MOVE $200_{immediate}$, R0
	places the value 200 in register R0.
	This is also written , MOVE #200, R0
	
	ii. *Register mode* – The operand is the contents of a processor register; the name of the register is given in the instruction.
	MOV R0, R1
	
	iii. *Indirect mode* – The effective address of the operand is the contents of a register or memory location whose address appears in the instruction.
	
	We denote indirection by placing the name of the register or the memory address given in the instruction in parentheses.
	
	iv. *Direct or Absolute mode* – The operand is in a memory location; the address of this location is given explicitly in the instruction.
	The instruction Move LOC,R2 uses two modes. Processor registers are temporary storage locations where data in a register is accessed using the Register mode.
	v. The effective address of the operand is generated by adding a constant value to the contents of a register. This register is referred to as index register,
	The register used may be either a special register provided for this purpose, or, more commonly, it may be any one of a set of general-purpose registers.
	We indicate the Index m de symbolically as,
	X(Ri)
	where X denotes the constant value contained in the instruction and Ri is the name of the register involved. The effective address of the operand is given by
	
	EA=X+[Ri}
		10	CO2	L2

5	Define a subroutine. With a program segment, illustrate parameter passing using registers. What is link register? It is often necessary to perform a particular subtask many times on different data values. Such subtask is called *subroutine.* E.g. For example, a subroutine may evaluate the sine function. a. When a program branches to a subroutine we call that it is *calling* a subroutine. b. The instruction that performs this branch	10	CO2	L2

	operation is called a Call instruction. c. The subroutine is said to *return* to program that called it by executing a Return instruction. d. The location where the calling program resumes execution is the location pointed by the updated PC while the Call instruction being executed. e. Hence the contents of the PC must be saved by the Call instruction to enable correct return to the calling program. **f. This way in which the computer makes it possible to call and return from subroutines is referred to as *subroutine linkage method.*** Passing parameters through processor registers is straightforward and efficient. Below is a program for adding a list of numbers using a subroutine with parameters passed through registers.			
	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			
	Program written as a subroutine; parameters passed through registers			
6	Consider a register R1 of size 16 bits with initial data 5876d. With a neat diagram, depict the output in each case after performing the following operations: i. LshiftL #2,R1 ii. AshiftR #1,R1 iii. RotateR #1,R1.	10	CO2	L3

				
7	Explain the IEEE standard used for single & double precision floating point number representation. Represent 85.125 in IEEE floating point using single-precision. ➡ General form for a floating-point number in the decimal numbering system: $\pm XI.X2X3X4X5X6X7 \times 10^{\pm Y1Y2}$ where , *Xi* and *Yj* are decimal digits Number of significant digits = 7 Exponent range = ±*99* **IEEE standard floating-point format – 32 bit**  Value represented = $\pm 1.M \times 2^{E'-127}$ **Description of Exponent:** • $0 < E' < 255$ • 0 and 255, are used to represent special values • Therefore, the range of E' for normal values is • $1 < E' < 254$ • *Hence the actual exponent E* is in the range $-126 < E < 127$	10	CO1	L3

	• The scale factor has a range of $2^{-126}$to 2^{+127},which is approximately equal to $10^{\pm 38}$ **IEEE standard floating-point format – 64 bit**  Value represented = $\pm 1.M \times 2^{E'-1023}$ • The 11-bit exponent • Called the excess-1023 representation • Exponent E' has the range $1 < E' < 2046$ for normal values, with • 0 and 2047 used to indicate special values, • The actual exponent E is in the range $-1022 < E < 1023$ • Scale factors of 2^{-1022} to 2^{1023} (approximately$10^{\pm 308}$). • The 52-bit mantissa provides a precision equivalent to about 16decimal digits **85.125 in IEEE floating point format:**  			
8	What are Assembler Directives? List and detail any Five of them. 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 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.	CO2	L2	10

	END is directive which indicates the end of the source program text. The END directive includes the label START, which is the address of the location at which execution of the program is to begin. **RETURN** is an assembler directive that identifies the point at which the execution of the program should be terminated.			