USN | | | | | | | | | | |

INTERNAL ASSESSMENT TEST – I

Sub	Computer Organization and Architecture							Code	BEC306C
Date	19/12/2023	Duration	90 mins	Max Marks	50	Sem	III	Branch	ECE

Answer any 5 full questions

		Marks	CO	RE
1.	a) Describe the basic functional units of a computer. b) Describe the basic performance equation of the computer processor.	[7] [3]	CO1 CO1	L L
2	List and detail the steps needed to execute the machine instruction: Add LOCA, R0	[10]	CO1	L
3	Explain the following i) Byte addressability ii) Big Endian assignment iii) Little Endian assignment.	[10]	CO1	L
4	Discuss the steps involved in running one application program using OS routines with a neat time-line diagram.	[10]	CO1	L
5	a. What is Straight Line Sequencing? Explain with an example. b. What is word alignment of a machine? Explain with examples	[5] [5]	CO1 CO1	L L
6	Write a program that can evaluate the expression $A \times B + C \times D$ in a single accumulator processor. Assume that the processor has Load, Store, Multiply and Add instructions that has all values fit in the accumulator.	[10]	CO1	L
7	a. Write a program to evaluate the expression $A \times B + C \times D$ using 3-operand (Address) Instructions b. Write a program to evaluate the expression $A \times B + C \times D$ using 2-operand (Address) Instructions	[5] [5]	CO1 CO1	L L
8	Give a short sequence of machine instruction for the task "Add the contents of memory location A to those of location B and place the answer in location C". Use only Load and Store instructions to transfer the data between memory and general purpose registers. Do not destroy the contents of either location A or B.	[10]	CO1	L

Solution:

Q1.

1.	a) Describe the basic functional units of a computer. b) Describe the basic performance equation of the computer processor.

Sol:

A.

Functional Units

A computer consists of five functionally independent main parts: input, memory, arithmetic and logic, output and control units as shown in fig 1.1.

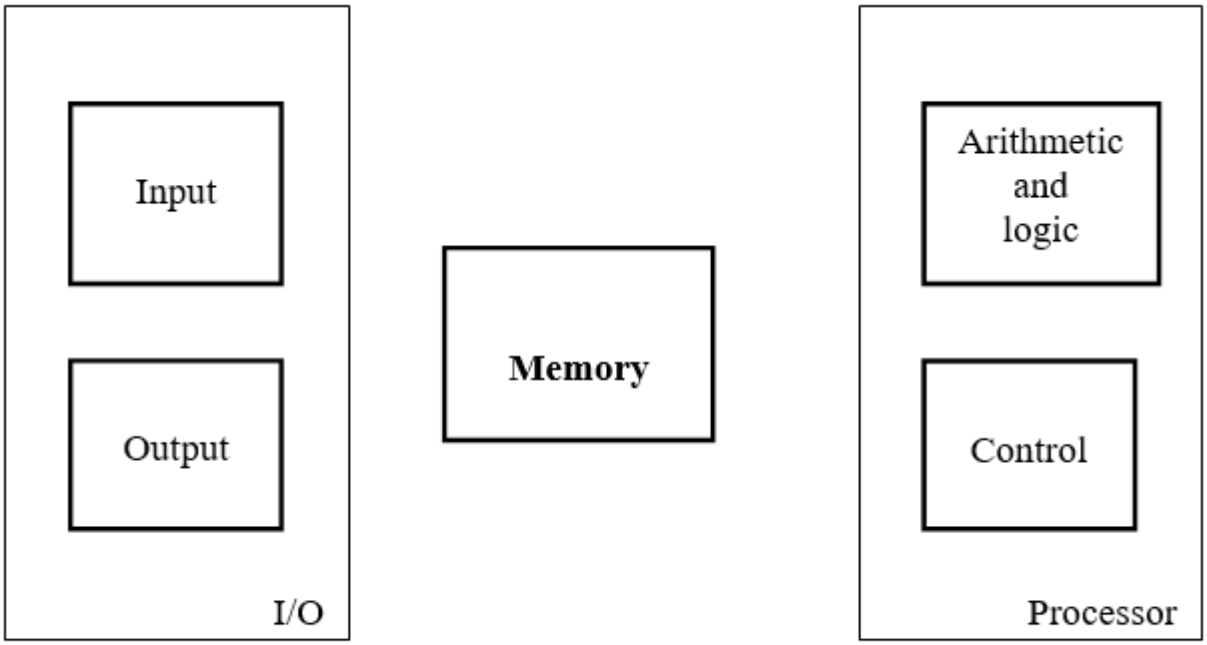

Fig 1.1 Basic functional units of a computer

The input unit accepts coded information from human operators, from electromechanical devices such as keyboards or from other computers over digital communication line. The information received is either stored in the computer memory for later reference or immediately used by the arithmetic and logic circuitry to perform the desired operations. The processing steps are determined by the program stored in the memory. Finally the results are shown on the output unit. All of these actions are co-ordinated by the control unit. We refer to the arithmetic and logic circuits in conjunction to the control circuits as the processor and input and output units are referred to as input-output (I/O) unit.

Instructions or *machine instructions* are explicit commands that

- Govern the transfer of information within a computer as well as between the computers and its I/O devices.
- Specify the arithmetic and logic operations to be performed.

A list of instructions that perform a task is called *program.* The computer is completely controlled by a stored program except for a possible external interruption by an operator or by I/O

devices connected to the machine. Data is used to mean any digital information. Each number, character or instruction is encoded as a string of binary digits known as bits each having one of two possible values 0 or 1.

Input Unit

Computers accept the coded information through input units which read the data. The well known input device is Keyboard. When a key is pressed, the corresponding letter or digit is automatically translated into its corresponding binary code and transmitted over a cable to either the memory or processor.

Memory Unit

The memory unit is used to store program and data. There are two classes of storage known as primary and secondary.

Primary memory is a fast storage that operates at electronic speeds. Programs are stored in the memory while they are executed. The memory contains large number of semiconductor storage cells each capable of storing one bit but instead are processed as groups of fixed size called words. The memory is organized so that a word can be stored or retrieved in one basic operation. A distinct address is associated to each word in the memory. Addresses are numbers that identify successive locations.

Programs must reside in the memory during execution. Instructions and data can be read out or written into the memory under the control of the processor. Memory in which any location can be reached in short and fixed amount of time after specifying its address is called random-access memory (RAM). The small, fast RAM units are called *caches*.

The additional cheaper secondary storage is used when large amount of data and many programs have to be stored particularly for information that is accessed infrequently.

Arithmetic and Logic Unit (ALU)

Any arithmetic and logic operation is initiated by bringing the required operands into the processor where the operation is performed by the ALU. When the operands are brought into the processor they are stored in high speed storage elements called *registers*. Access time to register is faster than access time to the fastest cache unit in the memory hierarchy. The control and the arithmetic logic units are many times faster than any other devices connected to a computer system.

Output Unit

The output unit is a counterpart of input unit. Its function is used the processed results to the outside world. The most familiar example of such a device is a printer.

Control Unit

The control unit is a well defined physically separate unit that interacts with other parts of the machine. The control unit sends the control signals to other units and senses their states. Timing signals are generated by the control circuits that determine when a given action is to take place. Data transfer between memory and processor is also controlled unit through timing signals. A large set of control lines (wires) carries the signals used for timing and synchronization of events in all units.

B.

Basic Performance Equation

Let T be the processor time required to execute a program that has been prepared by some high level language. The compiler generates machine level object program that corresponds to source program. Assume that complete execution of the program requires the execution of N machine language instructions. Suppose that the average number of basic steps needed to execute one machine instruction is S, where each basic step is completed in one clock cycle. If

the clock rate is R cycles per second, the program execution time is given by *basic performance equation.*

$$T = \frac{N \times S}{R}$$

To achieve high performance, the value of T must be reduced which can be done by reducing N and S, and increasing R. The value of N is reduced if the source program is compiled in fewer machine instructions. The value of S is reduced if instructions have a smaller number of basic steps to perform or if the execution of instructions are overlapped. Using a higher-frequency clock increases the value of R which means the time required to complete a basic execution step is reduced.

Q.2. List and detail the steps needed to execute the machine instruction: Add LOCA, R0

Sol:

Assume that the instruction is stored in memory location INSTR and this address is initially in register PC.

The steps needed to execute this machine instruction are listed below:

- Transfer the contents of PC to register MAR
- Issue a Read command to memory, and then wait until it has transferred the requested word into register MDR.
- Transfer the instruction from MDR into IR and decode it.
- Transfer the address LOCA from IR to MAR.
- Issue a Read command and wait until MDR is loaded.
- Transfer contents of MDR to the ALU.
- Transfer contents of R0 to the ALU.
- Perform addition of the two operands in the ALU and transfer result into R0.
- Transfer contents of PC to ALU.
- Add 1 to operand in ALU and transfer incremented address to PC.

Q.3. Explain the following

i) Byte addressability ii) Big Endian assignment iii) Little Endian assignment

Sol:

Byte Addressability

- A byte is always 8 bits, but the word length typically ranges from 16 to 64 bits.
- It is impractical to assign distinct addresses to individual bit locations in the memory.
- The most practical assignment is to have successive addresses refer to successive byte locations in the memory – byte-addressable memory.
- Byte locations have addresses 0, 1, 2, … If word length is 32 bits, they successive words are located at addresses 0, 4, 8,…

Big-Endian and Little-Endian Assignments

The name *big-endian* is used when the lower byte addresses are used for the most significant bytes (the leftmost bytes) of the word. The *little-endian* is used for the opposite ordering, when the lower byte addresses for the less significant bytes (the rightmost bytes) of the word. In both cases, byte addresses 0,4,8, ..., are taken as the address for the successive words in the memory and are the

addresses used when specifying the memory read and write operation for the words. The two ways that the byte addresses can be used across the words as shown in Fig 1.11.

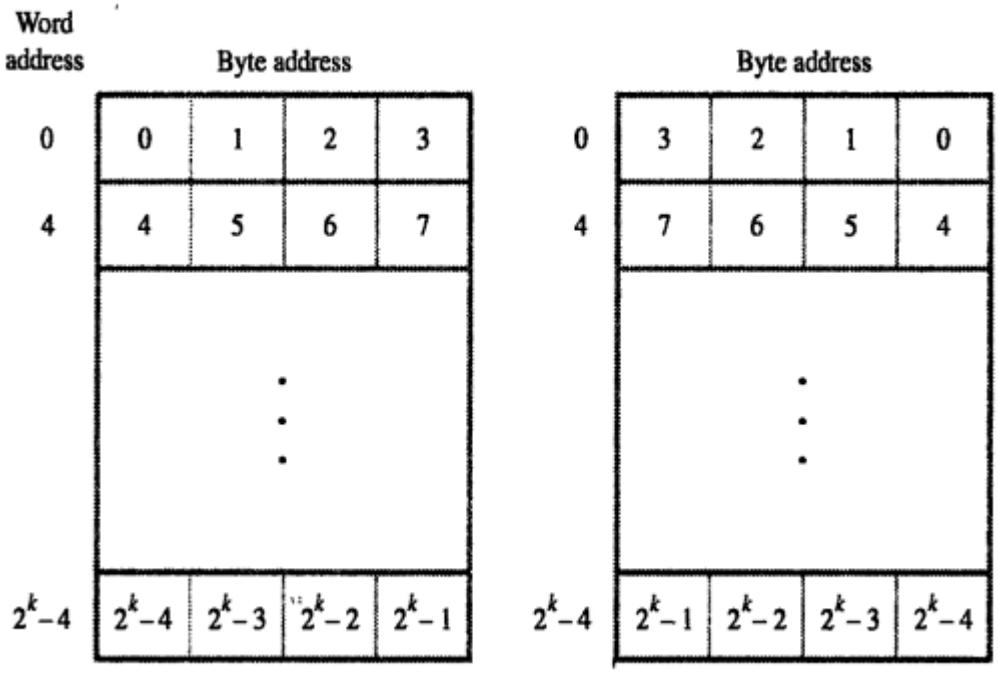

Fig 1.11: a) Big-endian assignment b) Little-endian assignment

The word locations have *aligned* addresses where the word begins at a byte address that is a multiple of number of bytes in a word. If the word length is 16 (2 bytes), aligned words begins at byte addresses 0,2,4 ....

Q4. Discuss the steps involved in running one application program using OS routines with a neat time-line diagram

Sol:

Consider a system with one processor, one disk and one printer. When the application program has been compiled from a high language form to machine language form and stored on the disk. The first step is to transfer this file into the memory. When the transfer is complete, execution of the program is started. Assume part of the program's task involves reading a data file from the disk in to the memory, performing some computations on the data and printing the results. When the execution of the program reaches the point where the data file is needed, the program requests the operating system to transfer the data file from the disk to the memory. The OS performs this task and passes the execution control back to the application program, which then proceeds to perform the required computation. When the computation is completed and the results are ready to be printed, the application program again sends the request to the operating system. An OS routine is then executed to cause the printer to print the results.

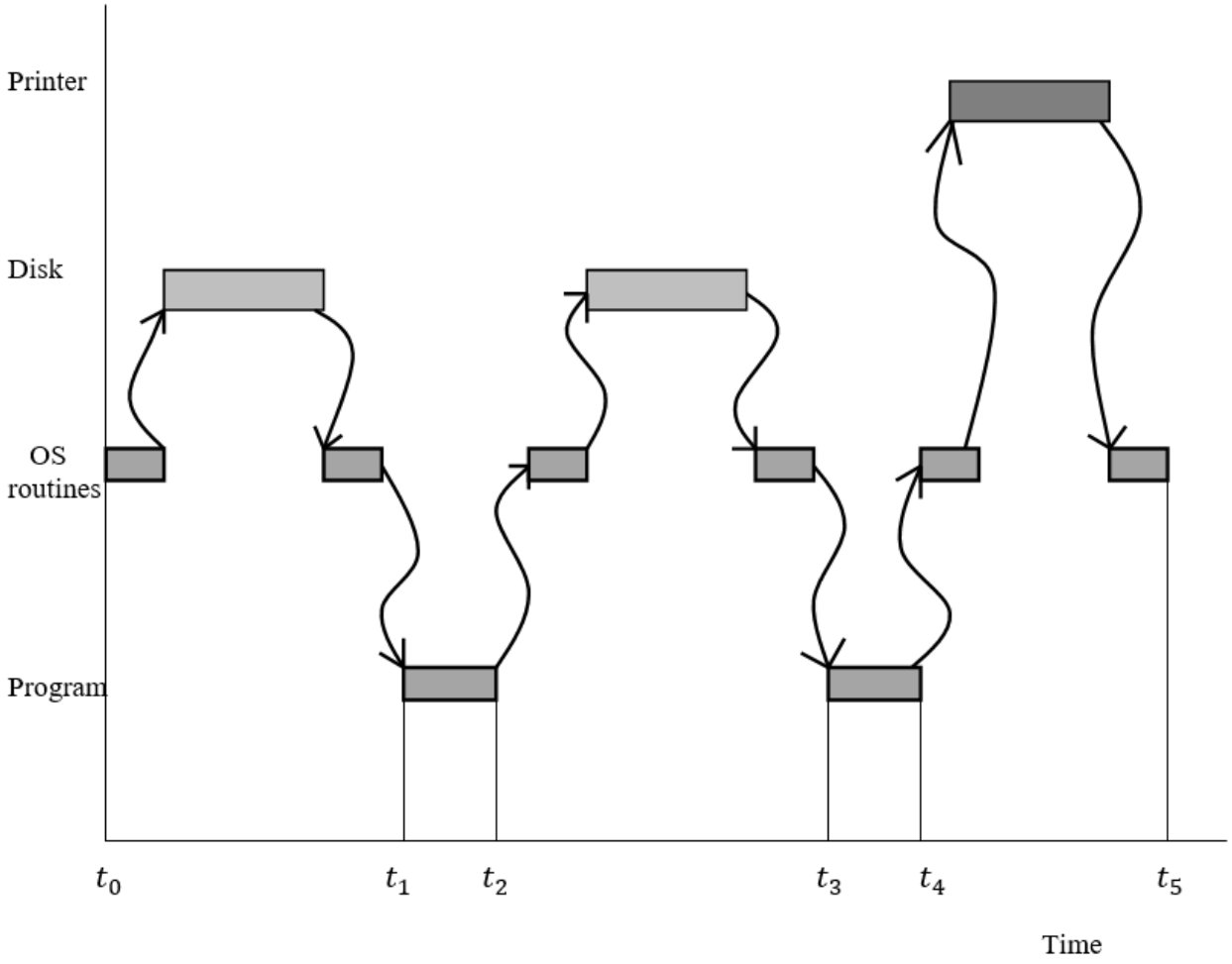

Fig 1.5: User program and OS routine sharing of the processor

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*

Q.5

a. What is Straight Line Sequencing? Explain with an example.
b. What is word alignment of a machine? Explain with examples

Sol:

Instruction Execution and Straight-Line Sequence

We assume computer allows one memory operand per instruction and has a number of processor registers. Fig 1.12 shows a program segment in the memory of a computer. The word

length is 32 bits and the memory is byte addressable. Each instruction is 4 bytes long, the second and third instructions start at addresses $i + 4$ and $i + 8$.

The Program Counter (PC) contains the address of the instruction to be executed next. To begin executing a program, the address of its fist instruction must be placed in to PC. Then the processor control circuits use the information in the PC to fetch and execute the instructions, one at a time, in the order of increasing addresses. This is called *straight-line sequencing.*

Executing a given instruction is a two phase-procedure. In the first phase called *instruction fetch*, the instruction is fetched from the memory location whose address is in the PC. This instruction is placed in the *instruction register* (IR) in the processor. At the start of second phase called *instruction execute*, the instruction in the IR is examined to determine which operation is to be determined.

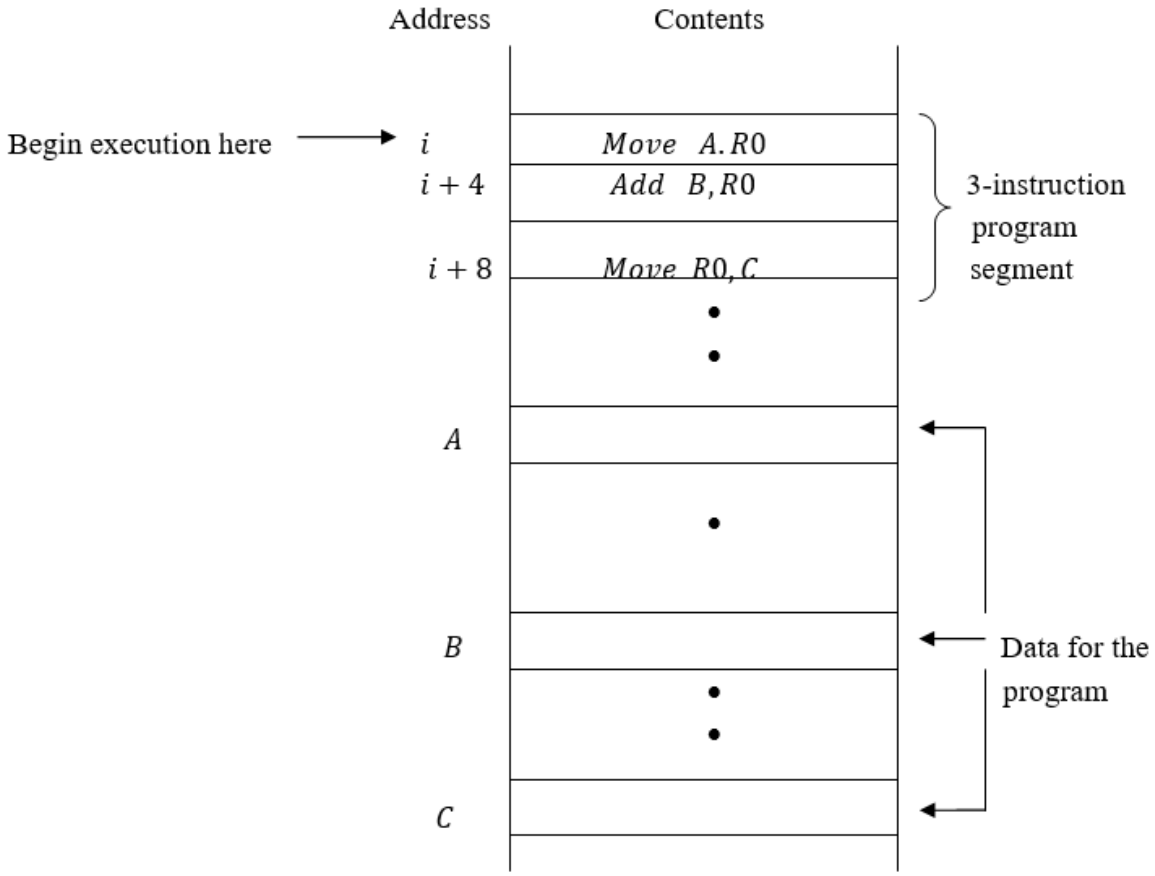

Fig 1.12: A program for $C \leftarrow [A] + [B]$

- **<u>Word alignment</u>**

  A byte is always 8 bits but the word length typically ranges from 16 to 64 bits. The successive addresses refer to successive byte locations in the memory. The term *byte-addressable memory* is used for this assignment. Byte locations of addresses 0,1,2 ...Thus, if word length of the machine is 32 bits, successive words are located at addresses 0,4,8, ..., with each word consisting of four bytes.

- Words are said to be aligned in memory if they begin at a byte addr. that is a multiple of the num of bytes in a word.
    - 16-bit word: word addresses: 0, 2, 4,….
    - 32-bit word: word addresses: 0, 4, 8,….
    - 64-bit word: word addresses: 0, 8,16,….

Q6. Write a program that can evaluate the expression $A \times B + C \times D$ in a single accumulator processor. Assume that the processor has Load, Store, Multiply and Add instructions that has all values fit in the accumulator.

Sol:

Example: Evaluate $A \times B + C \times D$

One-Address

Load A	; AC ← A
Multiply B	; AC ← AC * B
Store T	; T ← AC
Load C	; AC ← [C]
Multiply D	; AC ← AC * [D]
Add T	; AC ← AC + [T]
Store X	; X ← AC

Q.7:

Write a program to evaluate the expression $A \times B + C \times D$ using 3-operand (Address) Instructions

Sol:

Three-Address

Multiply A, B, R1	; R1 ← [A] * [B]
Multiply C, D, R2	; R2 ← [C] * [D]
Add R1, R2, X	; X ← [R1] + [R2]

Write a program to evaluate the expression $A \times B + C \times D$ using 2-operand (Address) Instructions

Sol:

Two-Address

Move A, R1	; R1 ← [A]
Multiply B, R1	; R1 ← [R1] + [B]

```
Move C, R2          ; R2 ← [C]
Multiply D, R2      ; R2 ← [R2] + [D]
Add R1, R2          ; R2 ← [R1] + [R2]
Move R2, X          ; X ← [R2]
```

Q.8: Give a short sequence of machine instruction for the task "Add the contents of memory location A to those of location B and place the answer in location C". Use only Load and Store instructions to transfer the data between memory and general purpose registers. Do not destroy the contents of either location A or B.

Sol:

```
Load A, R1          ; R1 ← [A]
Add B, R1           ; R1 ← [B] + [R1]
Store R1, C         ; C ← [R1]
```