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

INTERNAL ASSESSMENT TEST – I

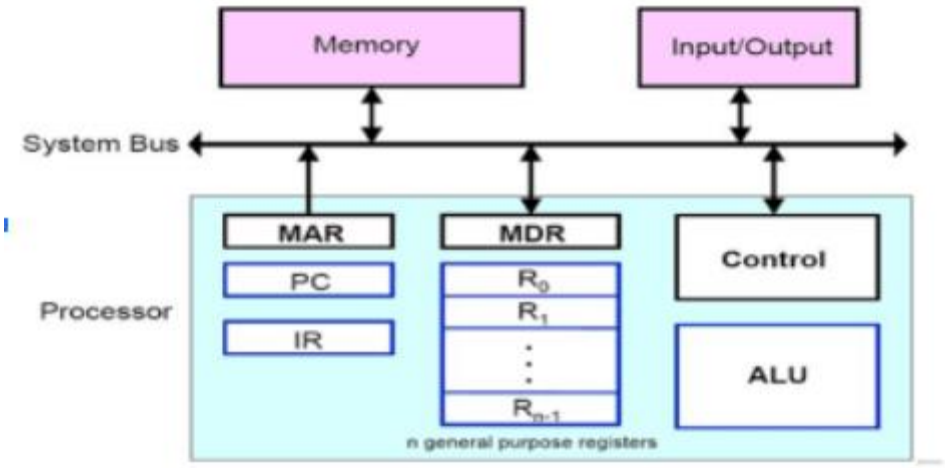
Sub:	Computer Organization and Architecture							Code:	BEC306C
Date:	08/11/2024	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	ECE

Answer any 5 full questions

		Marks	CO	RBT
1.	a) With a neat diagram, discuss the operation concepts in a computer highlighting the role of PC, MAR, MDR & IR. b) Describe the basic performance equation of the computer processor.	[7] [3]	CO1 CO1	L2 L2
2	List and detail the steps needed to execute the machine instruction: Add LOCA, R0	[10]	CO1	L3
3	Explain following with an example (a) Three- address instruction (b) Two-address instruction (c) One-address instruction	[10]	CO1	L2
4	What is operating system? Explain user program and OS routine sharing the processor.	[10]	CO1	L2

		Marks	CO	RBT
5	Consider a register R1 to 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	L2
6	Define Addressing mode. Discuss the following modes with example: i)Register ii)Direct iii)Indirect iv)Index.	[10]	CO2	L2
7	(a) What are assembler directives? (b) What is a subroutine? Illustrate subroutine function with parameter passing by value and reference with suitable diagram.	[2] [8]	CO2 CO2	L1 L2

1. a) With a neat diagram, discuss the operation concepts in a computer highlighting the role of PC, MAR, MDR & IR.



- Instruction register (IR) – holds the instruction that is currently being executed
- Program counter (PC)- contains the memory address of the next instruction to be fetched and executed.
- General-purpose registers (R_0 – R_{n-1})
- Memory address register (MAR) – contains the address of the location to be accessed
- Memory data register (MDR) – contains the data to be 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

b) Describe the basic performance equation of the computer processor.

Basic Performance Equation

- 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)
- S – average number of basic steps needed to execute one machine instruction. Each step completes in one clock cycle
- R – clock rate
- Note: these are not independent to each other

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

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

- Activity in a computer is governed by instructions.
- To perform a 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

- Add LOCA, R0
- Add the operand at memory location LOCA to the operand in a register R0 in the processor.
- Place the sum into register R0.
- The original contents of LOCA are preserved.
- The original contents of R0 is overwritten.
- Instruction is fetched from the memory into the processor – the operand at LOCA is fetched and added to the contents of R0 – the resulting sum is stored in register R0.

Separate Memory Access and ALU Operation

- Load LOCA, R1
- Add R1, R0

- The first of these instructions transfers the contents of memory location LOCA into processor register R1, and the second instruction adds the contents of registers R1 and R0 and places the sum into R0.
- The former contents of both registers R1 and R0 are destroyed, whereas the original contents of memory location LOCA are preserved.

3. Explain following with an example

Three- address instruction (b) Two-address instruction (c) One-address instruction

Instruction Formats

- Three-Address Instructions
  - ADD R1, R2, R3 R1 ← R2 + R3
- Two-Address Instructions
  - ADD R1, R2 R1 ← R1 + R2
- One-Address Instructions
  - ADD M AC ← AC + M[AR]
- Zero-Address Instructions
  - ADD TOS ← TOS + (TOS – 1)
- RISC Instructions
  - Lots of registers. Memory is restricted to Load & Store

Example: Evaluate (A+B) * (C+D)

- Three-Address

1. ADD R1, A, B ; R1 ← M[A] + M[B]
2. ADD R2, C, D ; R2 ← M[C] + M[D]

MUL X, R1, R2 ; M[X] ← R1 * R2

Example: Evaluate (A+B) * (C+D)

- Two-Address

1. MOV R1, A; R1 ← M[A]
2. ADD R1, B; R1 ← R1 + M[B]
3. MOV R2, C; R2 ← M[C]
4. ADD R2, D; R2 ← R2 + M[D]
5. MUL R1, R2 ; R1 ← R1 * R2

MOV X, R1; M[X] ← R1

Example: Evaluate (A+B) * (C+D)

- One-Address

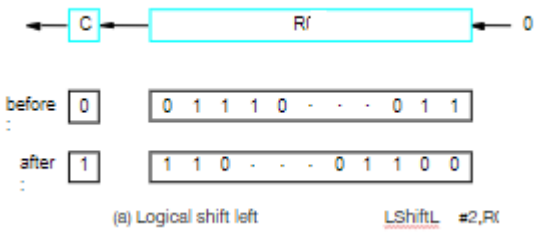
1. LOAD A ; AC ← M[A]
2. ADD B ; AC ← AC + M[B]
3. STORE T ; M[T] ← AC
4. LOAD C ; AC ← M[C]
5. ADD D ; AC ← AC + M[D]
6. MUL T ; AC ← AC * M[T]
7. STORE X ; M[X] ← AC

4.What is operating system? Explain user program and OS routine sharing the processor.

- System software is a collection of programs that are executed as needed to perform functions such as
  - Receiving and interpreting user commands
  - 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 the user
  - Controlling I/O units to receive input information and produce output results
  - Translating programs from source form prepared by the user into object form consisting of machine instructions
  - Linking and running user-written application programs with existing standard library routines, such as numerical computation packages
- Application programs are usually written in a high-level programming language, such as C, C++, Java, or Fortran, in which the programmer specifies mathematical or text-processing operations.
  A system software program called a *compiler* translates the high-level language program into a suitable machine language program.
- A large program or a collection of routines, that is used to control the sharing of and interaction among various computer units is called operating system (OS).
- The OS routines perform the tasks required to assign computer resources to individual application programs.
- These task include assigning memory and magnetic disk space to program and data files, moving data between memory and disk units, and handling I/O operations.

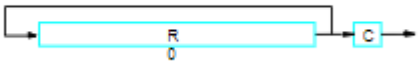
5.Consider a register R1 to size 16 bits with initial data 5876d. With a neat diagram, depict the output in each case after performing the following operations i) LshiftL #2,R1 ii)AshiftR #1,R1 iii) RotateR #1,R1.

(i)



(ii)

(iii)



6.Define Addressing mode. Discuss the following modes with example: i)Register ii)Direct iii)Indirect iv)Index.

Addressing Modes

- Implied
  - AC is implied in “ADD M[AR]” in “One-Address” instr.
  - TOS is implied in “ADD” in “Zero-Address” instr.
- Immediate
  - The use of a constant in “MOV R1, 5”, i.e. R1 ← 5
- Register
  - Indicate which register holds the operand

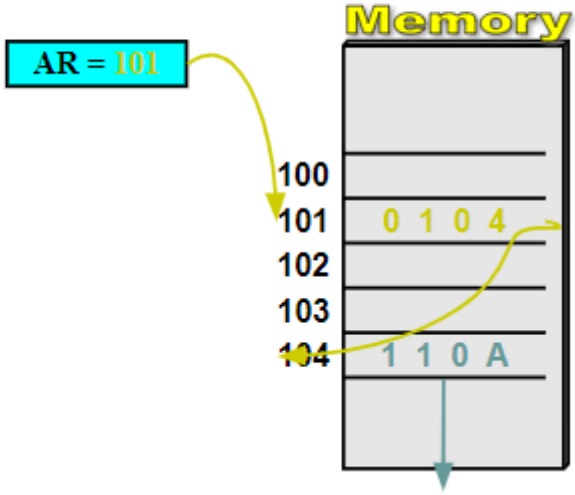
Addressing Modes

- Register Indirect
  - Indicate the register that holds the number of the register that holds the operand

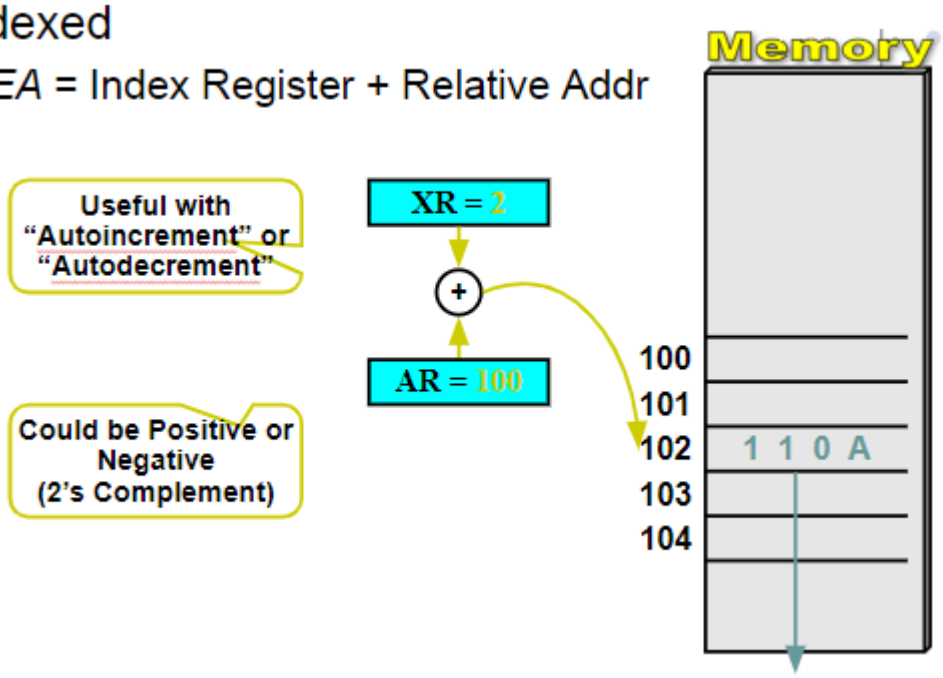
MOV R1, (R2)

- Autoincrement / Autodecrement
  - Access & update in 1 instr.
- Direct Address
  - Use the given address to access a memory location

- Indirect Address
  - Indicate the memory location that holds the address of the memory location that holds the data



- Indexed
  - *EA* = Index Register + Relative Addr



7.(a)What are assembler directives?

Assembler directives are commands that are part of the assembler syntax and control the assembly process.

(b)What is a subroutine? Illustrate subroutine function with parameter passing by value and reference with suitable diagram.

Subroutines are smaller, named sections of code that are written within a larger program close program Sequences of instructions for a computer. The purpose of a subroutine is to perform a specific task. This task may need to be done more than once at various points in the main program.

Parameter Passing

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 of Figure 2.16 written as a subroutine; parameters passed through registers.