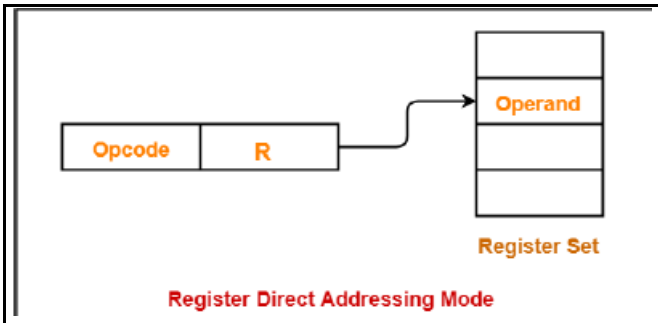

USN

Internal Assessment Test 2 – December 2024

Sub:	Digital Design and Computer Organization					Sub Code:	BCS302	Branch:	CSE
Date:	10/12/2024	Duration:	90 minutes	Max Marks:	50	Sem / Sec:	III / A, B, C		OBE

Answer any FIVE FULL Questions

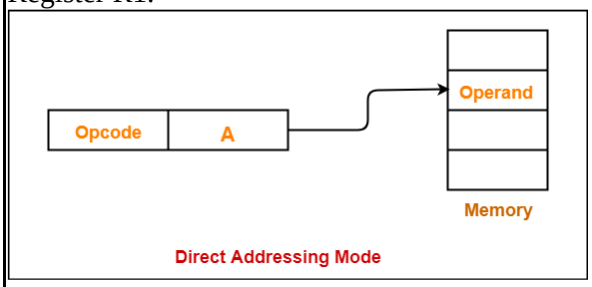
		MARKS	CO	RBT
1	a) Write a program that can evaluate the expression (A-B) + (C*D) using One and Two address instruction **Ans-** **Two-Address Instructions** MOV R1, A //Move A into register R1 SUB R1, B //Subtract B from R1 (R1 = A - B) MOV R2, C // Move C into register R2 MUL R2, D // Multiply D with R2 (R2 = C * D) ADD R1, R2 //Add R2 (C * D) to R1 MOV RESULT, R1 //Move the final result from R1 into memory RESULT **One address** LOAD A // Load A into the accumulator SUB B //Subtract B from the accumulator STORE T1 //Store the result T1 LOAD C //Load C into the accumulator MUL D // Multiply D with the accumulator STORE T2 // Store the result (C * D) into T2 LOAD T1 // Load T1 into the accumulator ADD T2 // Add T2 to the accumulator (ACC = (A - B) + (C * D)) STORE RESULT Store the final result into RESULT **N.B- "//" are comment lines**	[3+2]	3	L3
	(b) Explain any 5 addressing modes. Ans--- The various formats of representing operand in an instruction or location of an operand is called as "Addressing Mode". The different types of Addressing Modes are a) Register Addressing b) Direct Addressing c) Immediate Addressing d) Indirect Addressing e) Relative Addressing **a. REGISTER ADDRESSING:** In this mode operands are stored in the registers of CPU. The name of the register is directly specified in the instruction. **Ex:** MOVE R1,R2 Where R1 and R2 are the Source and Destination registers respectively. This instruction transfers 32 bits of data from R1 register into R2 register. This instruction does not refer memory for operands. The operands are directly available in the registers.	[5]	3	L3



Register Direct Addressing Mode

b. DIRECT ADDRESSING

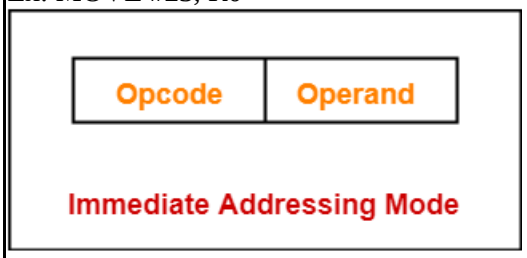
It is also called as Absolute Addressing Mode. In this addressing mode operands are stored in the
memory locations. The name of the memory location is directly specified in the instruction.
Ex: MOVE LOCA, R1 : Where LOCA is the memory location and R1 is the Register.
This instruction transfers 32 bits of data from memory location X into the General Purpose Register R1.



Direct Addressing Mode

c. IMMEDIATE ADDRESSING

In this Addressing Mode operands are directly specified in the instruction. The source field is used
to represent the operands. The operands are represented by # (hash) sign.
Ex: MOVE #23, R0



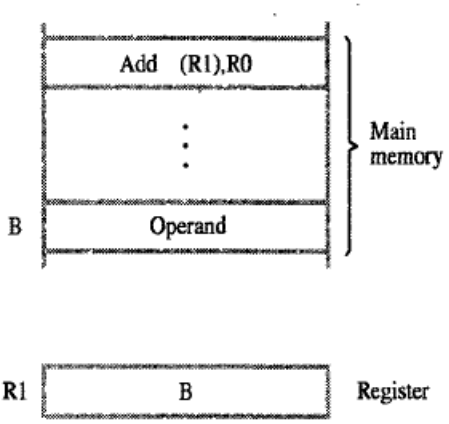
Immediate Addressing Mode

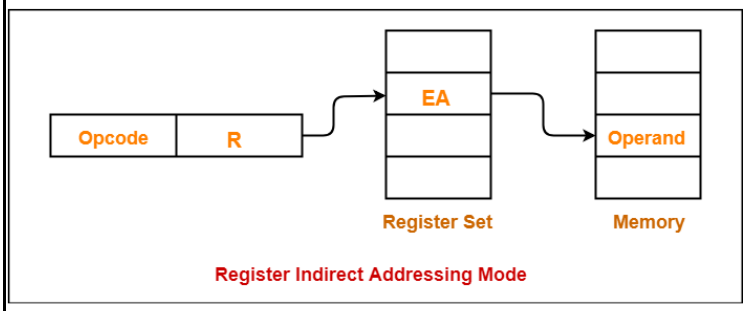
d. INDIRECT ADDRESSING

In this Addressing Mode effective address of an operand is stored in the memory location or General Purpose Register.
The memory locations or GPRs are used as the memory pointers.
Memory pointer: It stores the address of the memory location.
There are two types Indirect Addressing
i) Indirect through GPRs
ii) Indirect through memory location

i) Indirect Addressing Mode through GPRs

In this Addressing Mode the effective address of an operand is stored in the one of the General
Purpose Register of the CPU.
Ex: ADD (R1), R0 ; Where R1 and R0 are GPRs
This instruction adds the data from the memory location whose address is stored in R1 with the
contents of R0 Register and the result is stored in R0 register as shown in the fig.
The diagrammatic representation of this addressing mode is as shown in the fig.





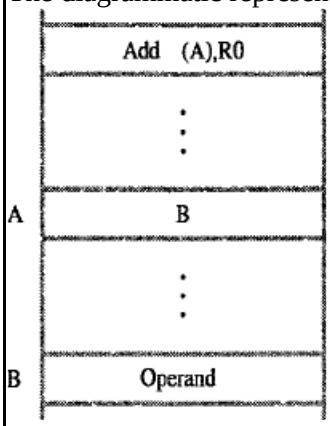
ii) Indirect Addressing Mode through Memory Location.

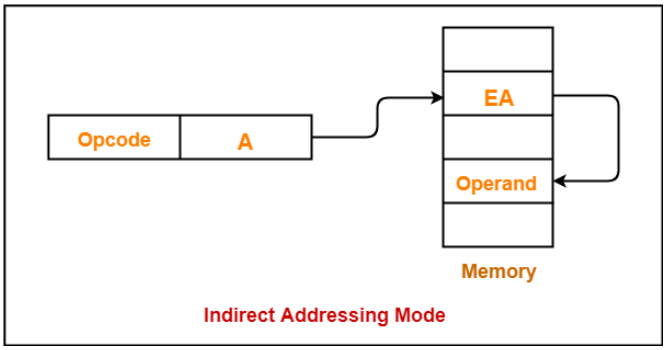
In this Addressing Mode, effective address of an operand is stored in the memory location.

Ex: ADD (X), R0

This instruction adds the data from the memory location whose address is stored in „X" memory location with the contents of R0 and result is stored in R0 register.

The diagrammatic representation of this addressing mode is as shown in the fig.





e. RELATIVE ADDRESSING MODE:

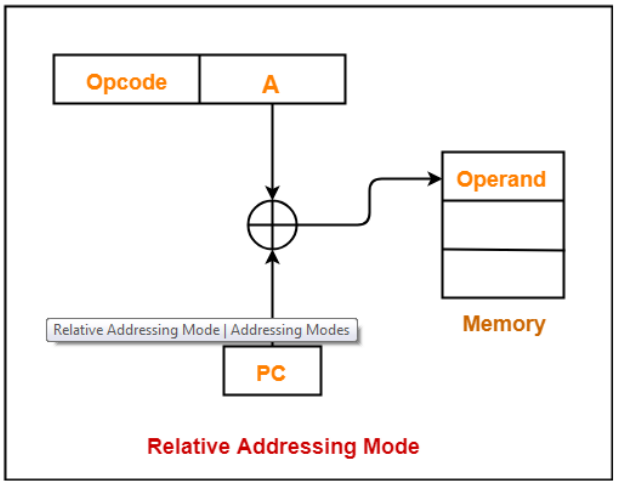
In this Addressing Mode EA of an operand is computed by the Index Addressing Mode. This Addressing Mode uses PC (Program Counter)to store the EA of the next instruction instead of GPR.

The symbolic representation of this mode is X (PC).Where X is the offset value and PC is the Program Counter to store the address of the next instruction to be executed.

It can be represented as EA of an operand = X + (PC).

This Addressing Mode is useful to calculate the EA of the target memory location.

Effective Address
= Content of Program Counter + Address part of the instruction



N.B Following is the summary of all addressing modes. Don't write only the table.

Table 2.1 Generic addressing modes

Name	Assembler syntax	Addressing function
Immediate	#Value	Operand = Value
Register	Ri	EA = Ri
Absolute (Direct)	LOC	EA = LOC
Indirect	(Ri) (LOC)	EA = [Ri] EA = [LOC]
Index	X(Ri)	EA = [Ri] + X
Base with index	(Ri,Rj)	EA = [Ri] + [Rj]
Base with index and offset	X(Ri,Rj)	EA = [Ri] + [Rj] + X
Relative	X(PC)	EA = [PC] + X
Autoincrement	(Ri)+	EA = [Ri]; Increment Ri
Autodecrement	–(Ri)	Decrement Ri; EA = [Ri]

EA = effective address
Value = a signed number

2	What is cache memory ? Explain different type of cache mapping function. Ans-	[10]	4	L2
3	(a) What is pipeline ? Explain different pipeline hazards. Ans- Pipeline is particularly effective way of organizing concurrent activity in a computer system to enhance the performance of a processor. It breaks down the execution of an instruction into separate stages, with each stage performing a specific task. This allows multiple instructions to be processed simultaneously in a "pipeline," significantly increasing instruction throughput. The execution of an instruction in a pipeline is divided into stages, such as:	[5]	5	L2

1. **Instruction Fetch (IF):** Retrieve the instruction from memory.
2. **Instruction Decode (ID):** Decode the instruction and identify the operation and operands.
3. **Execute (EX):** Perform the operation (e.g., arithmetic or logic).
4. **Memory Access (MEM):** Read or write data from/to memory if required.
5. **Write Back (WB):** Write the result back to the register.

Each stage works concurrently on a different instruction. For example:

- While instruction 1 is in the Decode stage, instruction 2 is in the Fetch stage, and instruction 3 can enter the pipeline.

Clock Cycle	IF	ID	EX	MEM	WB
1	Instr 1				
2	Instr 2	Instr 1			
3	Instr 3	Instr 2	Instr 1		
4	Instr 4	Instr 3	Instr 2	Instr 1	
5	Instr 5	Instr 4	Instr 3	Instr 2	Instr 1

-

Any condition that causes the pipeline to stall is called **a hazard**.
Different types hazards are-

1) **A data hazard is any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result, some operation has to be delayed, and the pipeline stalls.**
2) **Control hazards** or **instruction hazards**: The pipeline may also be stalled because of a **delay in the availability of an instruction**.

For example, this may be a **result of a miss in the cache** .

3) A **third type of hazard** known as a **structural hazard**: **This is the situation when two instructions require the use of a given hardware resource at the same time**

(b) Explain execution steps for ADD (R3),R1 — [5] — 5 — L1
Ans-

4 Explain single bus organization — [10 — 5
Ans-
Here the processor contains only a single bus for the movement of data, address and instructions.
ALU and all the registers are interconnected via a **Single Common Bus** (Figure).
Data & address lines of the external **memory-bus** is connected to the internal processor-bus via MDR & MAR respectively.
MDR has 2 inputs and 2 outputs. Data may be loaded
MAR‟s input is connected to internal-bus; MAR‟s output is connected to external- bus. (address sent from processor to memory only)
Instruction Decoder & Control Unit is responsible for
→ Decoding the instruction and issuing the control-signals to all the units inside the processor.

→ implementing the actions specified by the instruction (loaded in the IR).

Processor Registers - Register R0 through R(n-1) are also called as General Purpose Register.
The programmer can access these registers for general-purpose use.

Temporary Registers – There are 3 temporary registers in the processor. Registers

- **Y**, **Z** & **Temp** are used for temporary storage during program-execution. The programmer cannot access these 3 registers.

In **ALU**, "A" input gets the operand from the output of the multiplexer(MUX). "B" input gets the operand directly from the processor-bus.

There are 2 options provided for "A" input of the ALU.

MUX is used to select one of the 2 inputs.

MUX selects either

→ output of Y or

→ constant-value 4(which is used to increment PC content).

An instruction is executed by performing one or more of the following operations:

1) Transfer a word of data from one register to another or to the ALU.

2) Perform arithmetic or a logic operation and store the result in a register.

3) Fetch the contents of a given memory-location and load them into a register.

4) Store a word of data from a register into a given memory-location.

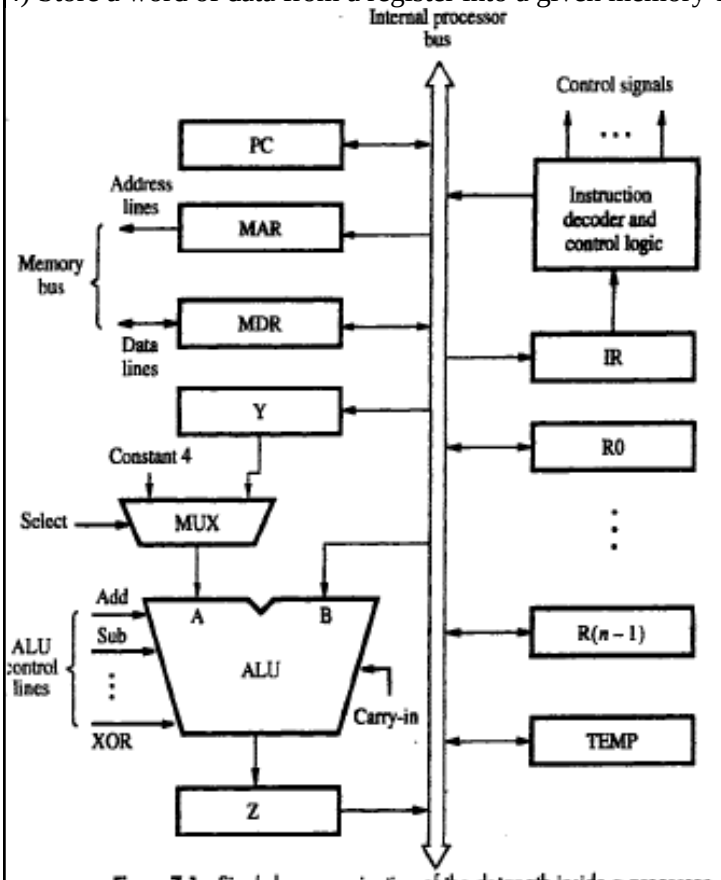


Figure 7.1 Single-bus organization of the datapath inside a processor.

5	What is bus arbitration. Explain types of bus arbitrations Ans-	10		
6	a)Explain DMA with a neat diagram. Ans- b)Explain vector interrupt and simultaneous interrupt requests Ans-	5+5		

(4 to 8 bits)

The device sends a special code, to the processor over the bus.

- The code contains :
  identification of the device,
  starting address of ISR,
  address of the branch to ISR (if ISR not at that location).

The location pointed to by the interrupting device is used to store the starting address of the interrupt service routine. This address is called interrupt vector. Processor reads it and loads it into PC.

* When the processor is ready to receive interrupt-vector code, it may activate interrupt-acknowledge line, INTA. The I/O device responds by sending its interrupt-vector code and turning off the INTR signal.

VECTORED INTERRUPTS

Device requesting an interrupt identifies itself directly to the processor.

CI CCI HoD