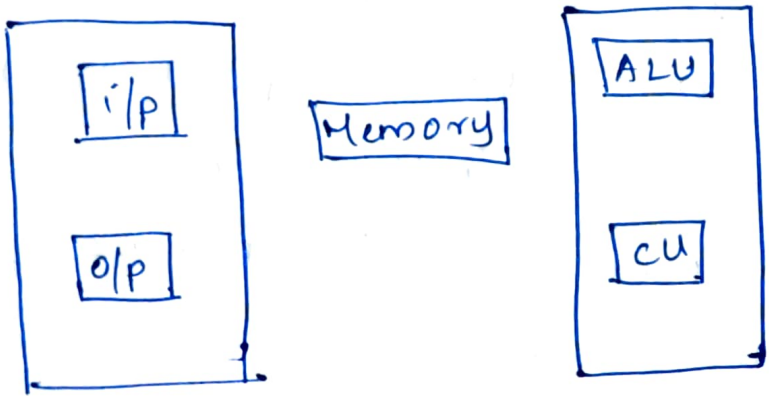
18/12/24

Module - I

Difference b/w computer Architecture & Computer Organization.

Computer Architecture	Computer Organization
* Computer Architecture is concerned with the way hardware components are connected together to form a computer system.	* Computer Organization is concerned with the structured & behaviour of a computer system as seen by the user.
* It acts as the interface b/w hardware & software	* It deals with components of a connection in a system.
* It helps us to understand functionalities of a system.	* It tells us how exactly all the units in the system are arranged & interconnected.

19/12/24 Basic Functional Units of a Computer



There are 5 basic functional units

→Input Unit

→Output Unit

→Memory

→ ALU

→Control Unit (CU)

Input & Output Unit

The computer system accept coded information through input units which reads the data. The most well known input unit is keyboard. The computer output units convert electric impulses to human readable form.

Memory Unit

Memory unit functions used to store programs & data. There are two types of memory:

→Primary memory

→secondary memory.

1. Primary memory:- It is a fast memory that operates at electronic speed. The programs must be stored in the memory while they are being executed. The primary memory refers to the main memory storage of the computer because it holds data & application

that are currently used by the computer.
eg: RAM, ROM, Cache Memory, Registers.

2. Secondary memory:- Secondary memory is also called as auxillary memory. It is used when large amount of data & many programs have to be stored.

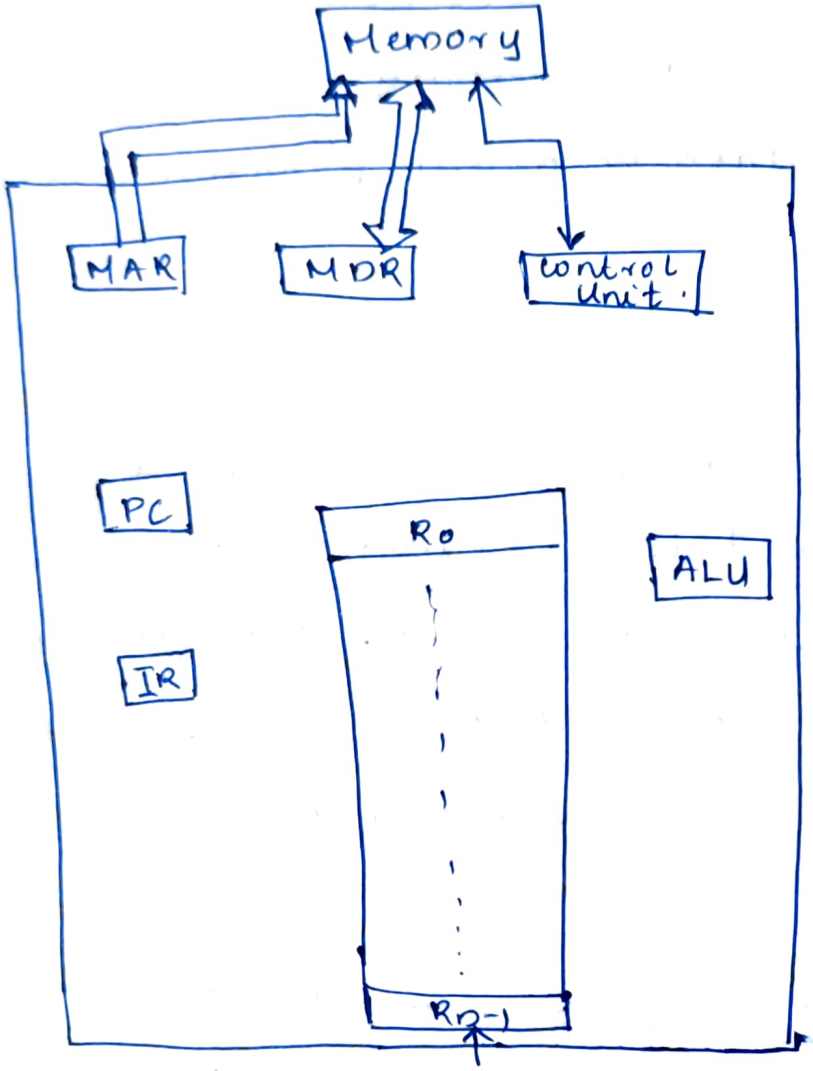
ALU

Most of the computer operations are executed in ALU. The data is transferred to the ALU from storage unit when required & after processing the output is return back to storage unit for further processing.

Control Unit

control Unit controls & coordinate every activities performed in a system. It send control signals to other units & senses their states. It consists of large set of control lines & carries the signals used for timing & synchronization of events of all units.

Basic Operational Unit



Registers

Registers are fast, small, stand alone storage location that hold data temporarily.

IR (Instruction Register)

It holds the instruction that is currently been executed.

PC (Program counter)

It is a specialized register that keeps track of execution of a program. It contains the memory address of the next instruction to be fetched or executed.

General Purpose Registers

MAR (Memory Address Register)

It holds the address of the memory location to be accessed.

MDR (Memory Data Register)

It contains the data to be written into or readout of the addressed memory location.

step 1: Program resides in the memory through the input unit.

step 2: Execution of the program starts when the PC is said to point at the first instruction of the program.

Step 3: The contents of the PC are transferred to MAR & a read control signal is generated by the control unit & is send to the memory.

Step 4: The instruction is fetched & is loaded into the MDR

Step 5: The contents of the MDR are transferred to the IR.

Step 6: The instruction in IR is decoded.

Step 7: If the instruction involves an ALU operation, it obtains the required operand from GPR (General Purpose Register) or memory.

Step 8: If an operand is in the GPR, it is transferred to the ALU for operation

Step 9: If an operand is in the memory, it is fetched by sending its address to MAR & initiates a read signal.

Step 10: When operand has been read, from the memory into the MDR, it is transferred from MDR to the ALU.

Step 11: When all operands are available, the ALU performes the desired operation

Step 12: Results are stored back to ~~GIR~~ GPRs or memory.

Step 13: If the result of this operation is to stored in the memory, the result is send to MDR.

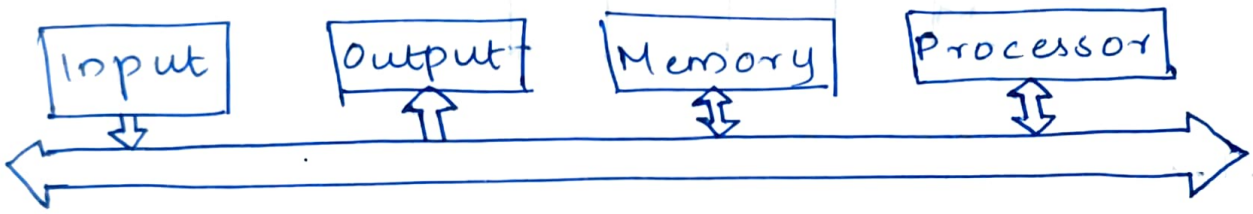
Step 14: Address of location where the result is stored is send to MAR & write ~~cycle~~ signal is initiated.

Step 15: During the execution, the content of the PC is incremented so that PC points to the next instruction that is to be executed.

20/12/24 **Bus**

A bus is a group of lines that serves as a connecting path for several devices.

Single bus



Memory location & Address

Endianess is a term that describes the order in which a sequence of bytes is stored in computer memory.

1. Big Endian

Most significant Bit is stored in the lower byte address. eg: JOHNSON

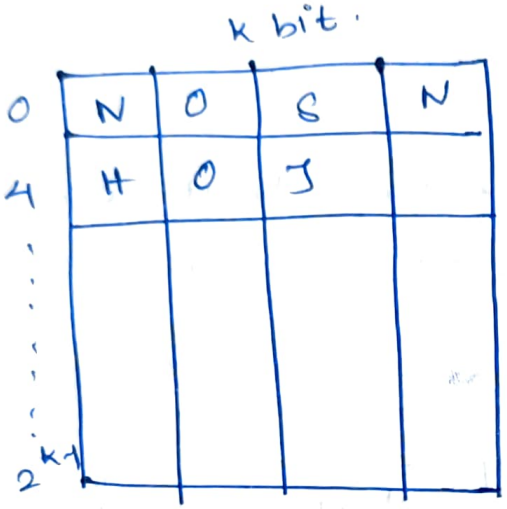
~~If the size of the data~~

kbit

0	J	O	H	N
4	S	O	N	
k-1				
2				

2. Little Endian

The least significant bit is stored in the lower byte address.



03/01/25

Instruction & Instruction sequencing

Instructions

A computer may have instruction capable of performing 4 types of operations.

(i) Data transfer b/w the memory & the processor register.

(ii) Arithmetic instruction.

(iii) Logical instruction

(iv) Branching instruction.

- Different notations are used to represent the 4 basic operations. They are:

(i) Register transfer notation

(ii) Assembly language notation.

Assembly Language Notation

These types of notations are used to represent machine instructions & programs.

eg: C = A+B

ADD A, B

MOV B, C

Register Transfer Notation

eg: C ← [A] + [B]

R0 ← [A]

R_1 ← [B]

R_3 ← [A] + [B]

C ← R_3

Types of Instructions

1) 3 Address Instruction
2) 2 Address Instruction
3) 1 Address Instruction
4) zero Address Instruction.

eg: 1) ADD A, B, C

2) ADD A, B

3) ADD A

4) PUSH.

1) 3 Address Instruction

Syntax:-

Operation Source 1, Source 2, Destination.

eg: ADD A, B, C

Adding A & B & result stored in C.

2) 2 Address Instruction

Syntax:-

Operation Source1, Source2/Destination

eg: ADD A, B.

Adding A & B & result is stored in B

3) 1 Address Instruction

Syntax:-

Operation source/Destination.

eg: Add A.

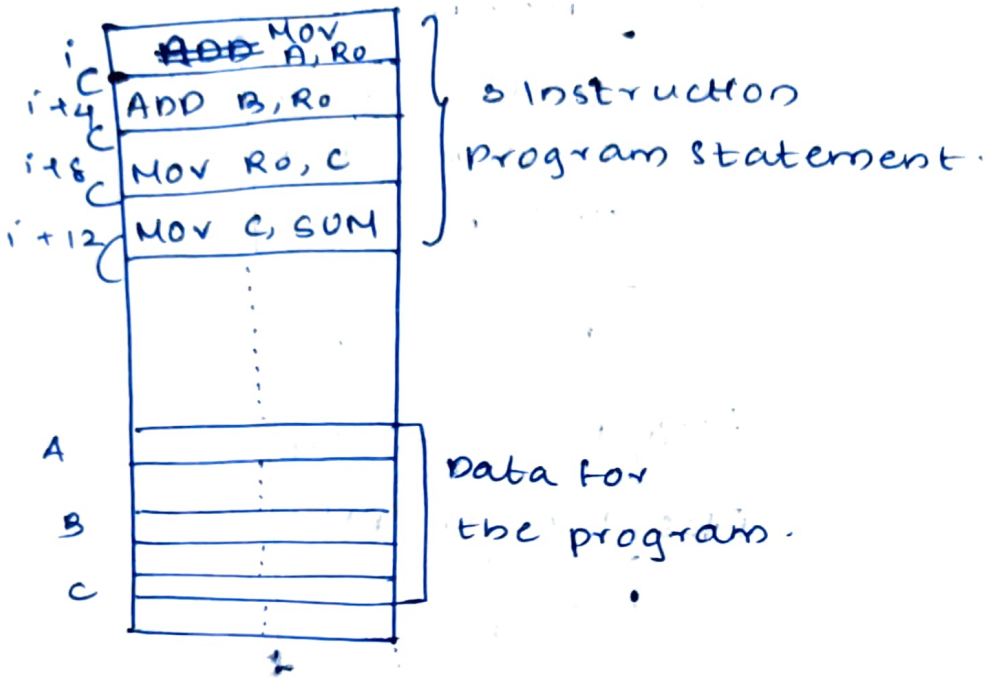
The content of the memory location A will be added to the content of the accumulator & the result will be stored in A or accumulator (depending upon the process).

4) zero address Instruction

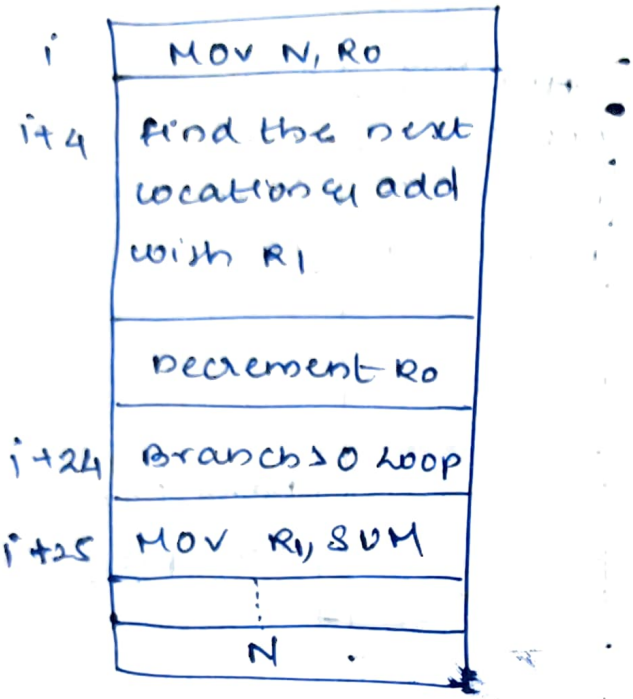
Typically used with stack.

eg: PUSH.

Instruction sequencing



PC point to the first instruction. The control unit fetch the data & execute the first instruction. While executing the first instruction, the PC automatically points to i+4. The CU always watching the PC value after completing the execution of the first instruction, CU check the value in the PC, then second instruction get executed. This is called straight line execution. ie, fetching & executing one at a time in the order of increasing addresses.



When we are using branch instructions, it loads a new value into the programs counter instead of following the sequential address order.

06/01/25 Addressing Modes

1) Immediate Addressing Mode.

Data is directly given, i.e, operand = value.

eg: MOV #100, R1

No memory reference is needed to fetch the data.

2) Register Addressing Mode

The operands is held in register whose name (Address) given in the instruction in address field.

No memory access.

operands are stored from R1 to R6

eg: MOV R_1, R_2
ADD R_1, R_2

3) Direct / Absolute Addressing Mode (1 memory referencing)

Address fields contain address of operand.

eg: MOV LOC, R_1

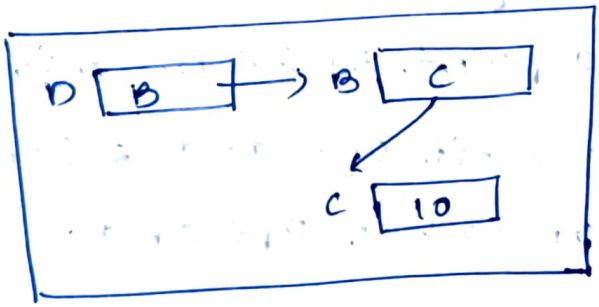
Here, the LOC is a memory location where the operand is situated. Single memory reference to access the data.

4) Indirect Addressing

Multiple memory access is needed for indirect addressing mode.

eg: MOV (D), E.

Here D acts as the pointer to the operand.



5) Indexed / Displacement Addressing Mode

a) Index X(Ri)
b) Base with index (Ri, Rj)
c) Base with index & offset X(Ri, Rj)

a) It is used for accessing arrays. X + Ri generate the address of the operand.

b) Content of Ri + Content of Rj give the address of operand.

c) Sum of constant x + content of registers Ri & Rj gives the address of the operand

x + Ri + Rj

6) Relative Addressing mode

eg: MOV 100(PC), Ri

The address of the operand is determined by adding the constant with the PC's value

7) Auto Increment Addressing Mode [(Ri)+]

Post increment

indirect

The address of the operand is specified in the instruction. After accessing the operand, the content of registers are automatically incremented to point to the next item. The increment is 1 for 8 bit operand, 2 for 16 bit operand & 4 for 32 bit operand.

8) Auto decrement

The content of the register specified in the instruction are first automatically decremented & then used as the address of the operand.

Questions

1) MOV −(R4), R3

16 bit.

↳ 8 bit.

~~8 bit → 1 bit~~

12

R4 | 1018

1015 | 16

1014 | 15

1011 | 12

2) MOV (R4)+, R3 → 16 bit operand.

R4 | 33

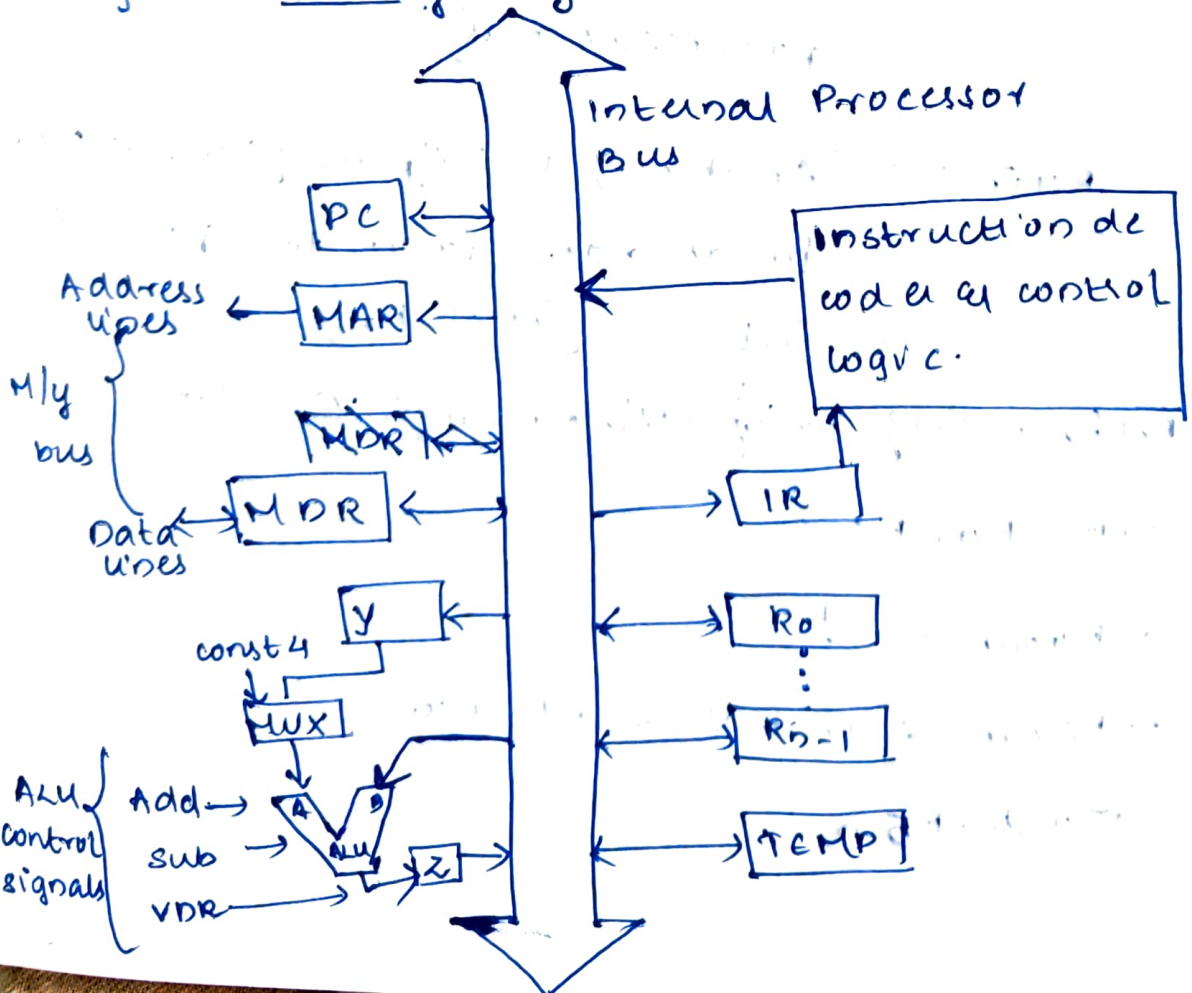
33 | 2

2

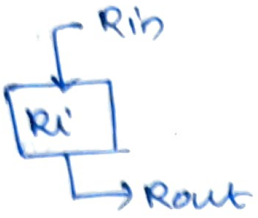
R4 increment to 35.

08/01/25 Basic Processing Unit

Single Bus Organization



Registered Transfers



For each register, two control signals are used.

(i) Place the contents of the register on the bus. (Rout)

(ii) To load the data on the bus into register (Rin)

* MOV R_1, R_4

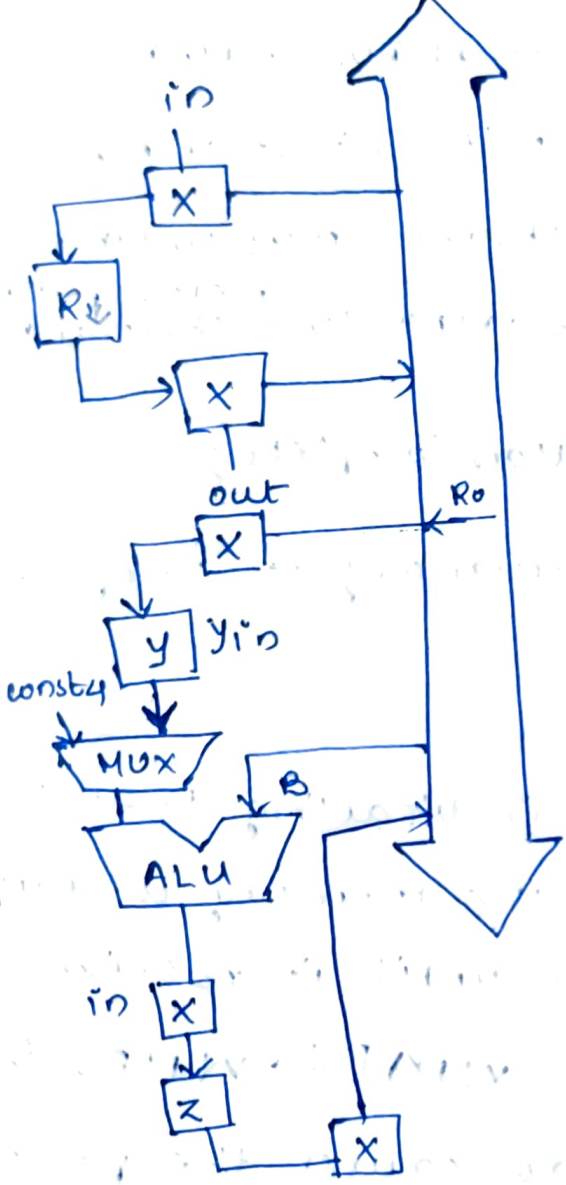
$R_{1out} = 1$ (Enabling the o/p of register R_1, i.e, this place the content of R_1 on the processor bus)

$R_{4in} = 1$ (Enabling the i/p register of R_4, this loads data from the processor bus into register R_4)

Performing Arithmetic or Logic Operation

eg: SUB R_4, R_5.

1. $R_{4out} = 1$, y_{in}
2. $R_5 out$, Select y, Subtract, R_{in}
3. Z_{out}, $R_5 in$.



10/01/25

Fetching a word from memory.

To fetch an instruction or operand from the memory, the processor needs its location. The address of this is given by MAR. The program or data which is read from the memory is received in MDR. MDR consists of 4 control signals, MDRout, MDRin, MDRoutE & MDRinE

MDRin ⇒ from internal Bus to MDR

MDRout ⇒ from MDR to internal Bus

MDRinE ⇒ from external Bus to MDR

MDRoutE ⇒ from ~~eg:~~ MDR to external Bus (memory data)

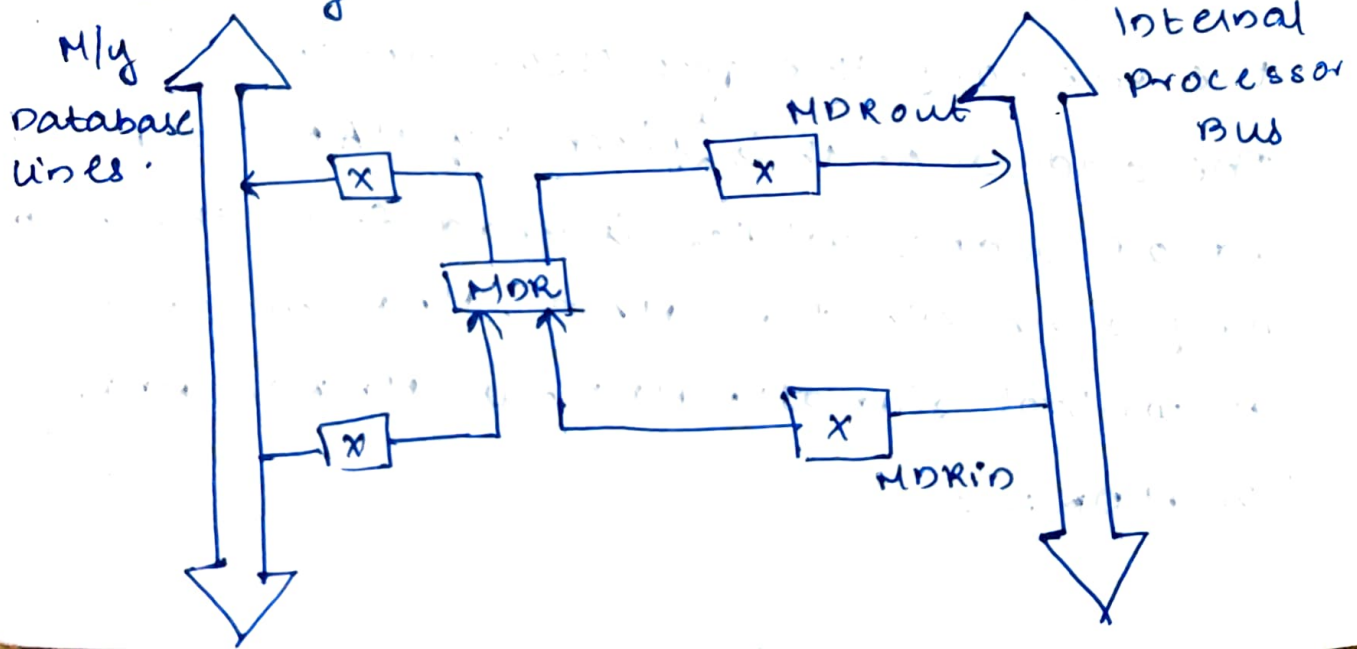
MFC (Memory Function Complete)

WMFC (wait for memory to complete)

~~eg: MOV (R3), R2~~

+ MFC is used for indicating that the requested action from the memory is complete. The processor waits until it receives an MFC control signal ie, WMFC. WMFC is the control signal that causes the processors' control circuit to wait for the arrival of MFC signal.

storing a word from memory.

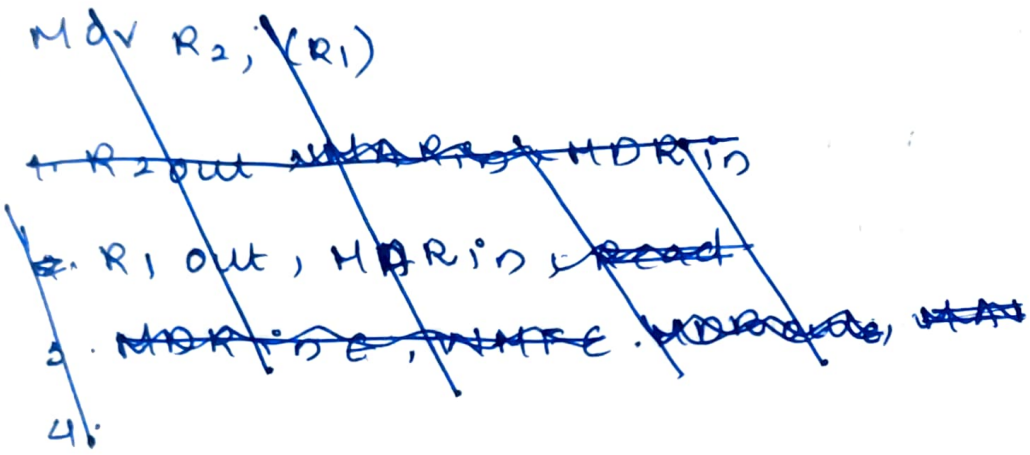


eg: MOV R_3, R_2

1. R_3out, MARin, Read
2. MDRinE, WMFC
3. MDRout, R_2in

Storing a word to memory

1) MOV R_2, (R1)



1. R_2out, MARin, Write
2. R1 out, MDRin
3. MDRinE, WMFC
4. MDRout, WMFC

1) MOV R_2, (R1)

1. R1out, MARin, Write
2. R_2out, MDRin
3. MDRoutG, WMFC.

2-Word instruction Execution

MOV R_2, (R1)

PCout, MARin, Read, Select 4, Add

Instruction execution is classified into two phase:

Fetching phase
Execution phase.

Write a control sequence for performing the following:

1) MOV R_2, (R_1)

1. PCout, MARin, Read, Select 4, Add Zin
2. Zout, PCin, Yin, WMFC
3. MDRout, IRin
4. R_1out, MARin, Write
5. R_2out, MDRin
6. MDRoutE, WMFC

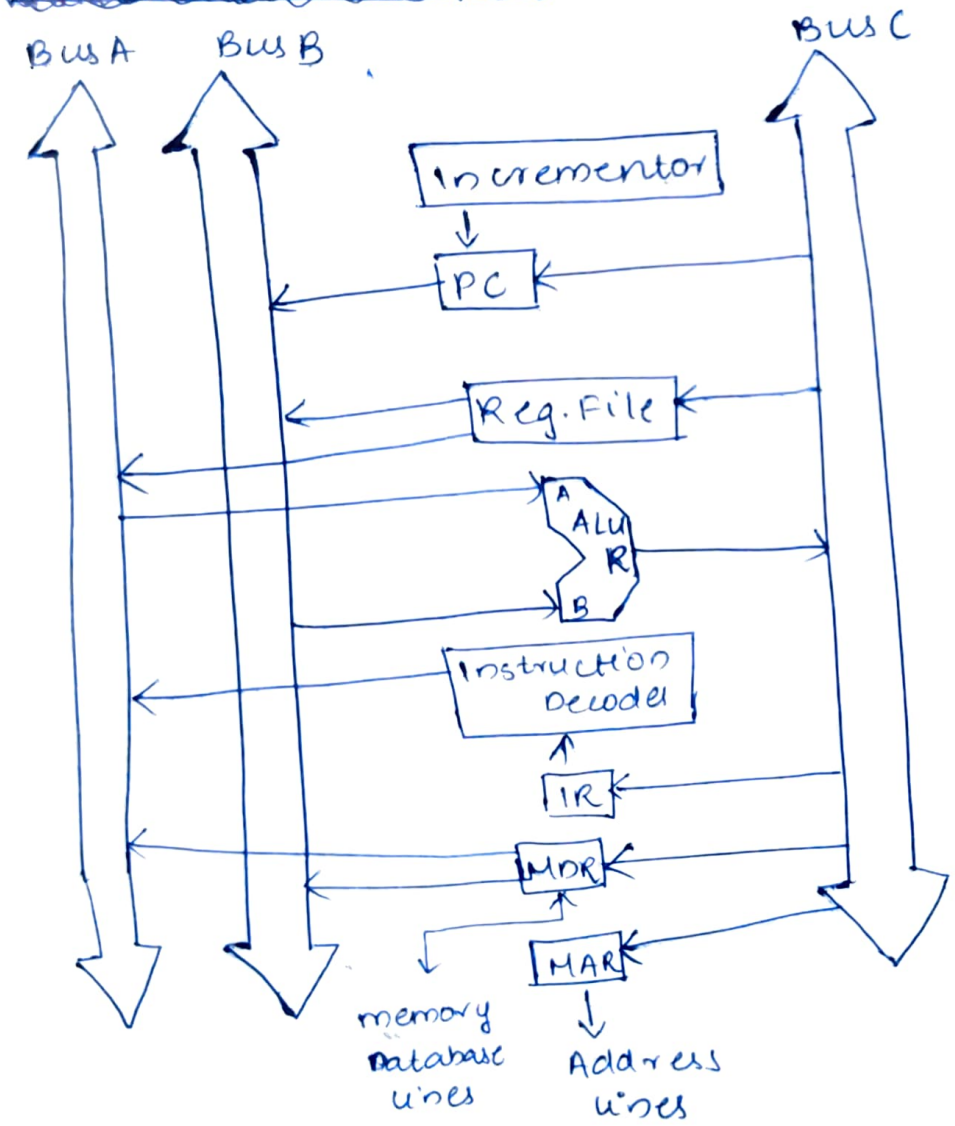
(1, 2, 3 — fetching phase / common to all)

2) Write a control sequence to perform Add (R_2), R_1

1. PCout, MARin, Read, Select 4, Add Zin
2. Zout, PCin, Yin, WMFC
3. MDRout, IRin
4. R_2out MDRin
5. MDRoutE, MARin, Read
6. MDRinE, WMFC
7. R_1out Yin
8. MDRout, Select y, Add, Zin
9. Zout, R_1in

Multiple Bus Organization

control sequence for the instruction used by the single bus is quite long because only one data item can be transferred over the bus. To reduce the no. of steps, use multiple internal paths that enable several transfers to take place in parallel.



1) Add R4, R5, R6

Step	Action
1.	PCout, R=B, MARin, Read, IncPC
2.	WMFC

3. MDRoutB, R=B, IRin

4. R4outA, R5outB, SelectA, Add, R6in, ADD

CA-01-25

Module-2

Register Transfer Logic

A digital system is a sequential logic system constructed with flipflops, gates, adders, decoders, counters, etc. To specify a large digital system with a state table would be very difficult. To describe a digital system in terms of functions such as decoders, adders & registers, it is necessary to employ a higher level mathematical notation. Thus the registers are selected to be the primitive components in the digital system. The register transfer logic describes the information flow & processing tasks among the data stored in the registers in a precise & concise manner.

The components of a digital s/m

1. The set of registers in a system & their fns.
2. The binary coded information stored in the register.
3. The operation performed on the information stored in register.
4. The control fns. that initiate the sequence of operations