Module - 1

1. Functional units

Functional units of a computer system are:

i) Input Unit
ii) ALU
iii) Control unit
iv) Memory unit
v) Output unit.

i) Input & Output unit:

The method of feeding data and program to a computer is done by input device. Input device read data from a source such as magnetic disk and translate that data into electronic impulse for transfer into CPU.

eg: Mouse, keyboard.

Output device convert electronic impulse into human readable form

eg: display screen, printer.

Memory unit: collection of storage cells along with associated circuits needed to transfer information in and out of storage.

Primary storage (Random Access Memory / Main memory) refers to main storage of computer which holds data & applications currently in use by computer.

Secondary storage: external storage devices.

Arithmetic Logical Unit (ALU)

ALU perform actual processing of data and instruction. The major operations performed are addition, subtraction, multiplication, division, logic and comparison.

Control Unit : determines the sequence in which computer programs and ~~se~~ instructions are executed.
Things like processing of program stored in main memory, interpretation of instructions, issuing of signals for other units of computer to execute them.
Control unit controls and coordinates the entire operation of the computer.

---

2. Addressing Modes : The different ways for specifying the location of instruction operands are known as addressing modes.

I. Implementation of Variables & Constants

a) Register mode : The operand is the contents of a processor register. The name of the register is given in the instruction.
eg: Add R_4, R_2, R_3

b) Absolute Mode : Operand is in a memory location.
eg: Load R_2, NUM1 loads the value in

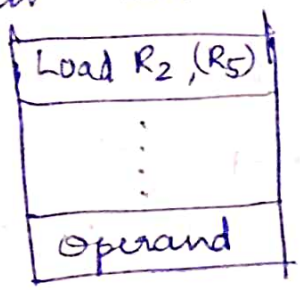
memory location ~~LOCA~~ NUM1 into Register R2.

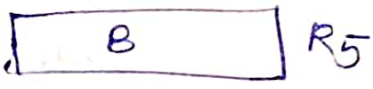
c) Immediate mode : The operand is given explicitly in the instruction.

Add R4, R6, #200

II. Indirection And Pointers : The effective address of operand is the contents of a register that is specified in the instruction.

eg: Load R2, (R5)



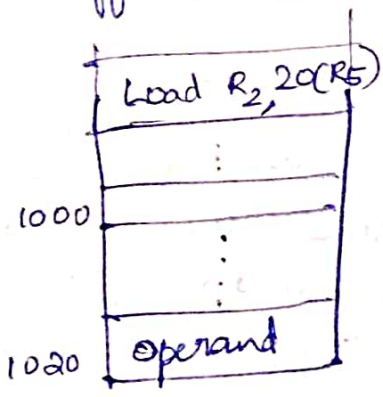


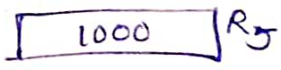
The processor uses the value B, which is in register R5, as the effective address of the operand.

III Indexing And Arrays : The effective address of the operand is generated by adding a constant value called offset to the content of a register

eg: X(Ri)

EA = X + [Ri]





IV Relative addressing : In index addressing, if the program counter PC, is used instead of a general purpose register, then, X(PC) can be used to address a memory location that is X bytes away from the location presently

pointed to by program counter.
It is used to specify the target address in branch instruction.

5. Additional Modes

a) Auto-increment: The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next item in a list eg: (Ri)+

b) Auto-decrement mode: The content of register specified in the instruction are first automatically decremented and then used as effective address of the operand

eg: -(Ri)

---

3. Instruction Types

The arrangement of a computer's register determines the different address fields in the instruction format.

3-Address instructions

eg: ADD X, Y, Z.

Source operands are X and Y.

Z is the destination operand.

2 - Address Instruction

eg: Add X, Y

X is source operand while Y is used both as source and destination operand.

1 - Address Instruction

eg: Add A

'ADD' is the operation implemented in operand A. This instruction adds the content of variable A into the accumulator and saves the result in the accumulator by restoring content of accumulator

0 - Address Instruction

operands are represented implicitly.

Store operands in a structure known as push-down stack

---

4. Control Sequence for MUL (R2), R1

1. PC_{out}, MAR_{in}, Read
2. MDR_{out}, IR_{in}
3. $R2_{out}$, MARin, Read
4. ~~MDR_{out}, Y_{in}~~ R_1out, Y_{in}, WMFC
5. MDR_{out}, SelectY, & MUL, Z_{in}
6. Z_{out}, $R1_{in}$, End

Executing the instruction requires

1. Fetch the instruction
2. Fetch the first operand (content of memory location pointed by R_2)
3. Perform Multiplication.

4. Load result into R_1

5. SINGLE BUS ORGANIZATION

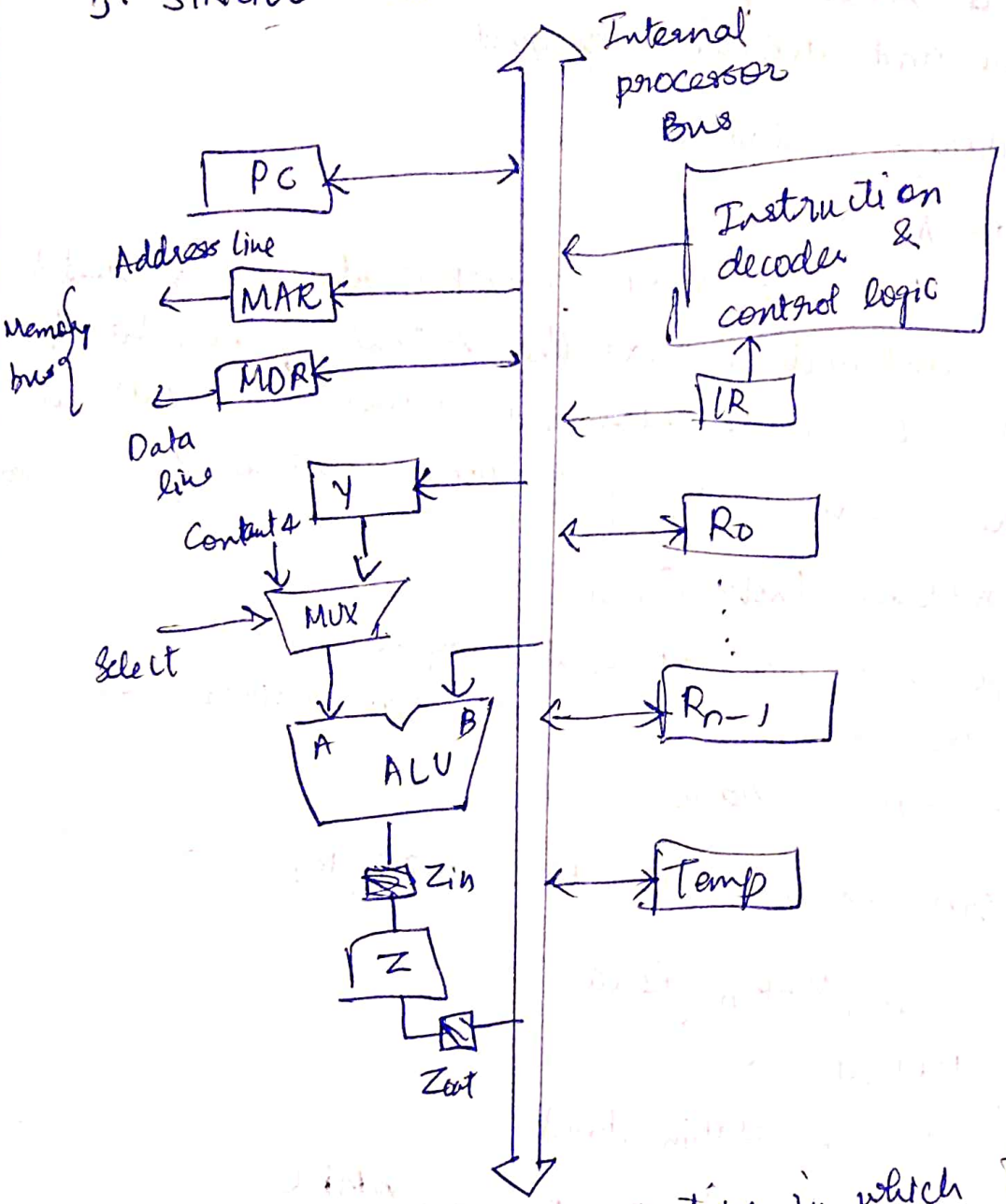


Figure shows the organisation in which the arithmetic and logic unit (ALU) and all the registers are interconnected via a single common bus.

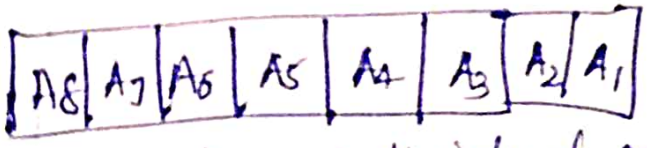
The data stored in MDR may be placed on either bus. The input of MAR is connected to the internal bus, and its output is

connected to external bus.
Three registers Y, Z and TEMP registers are used by processor for temporary storage during execution of some instruction. The multiplexer MUX selects either the output of register Y or a constant value 4 to be provided as input A of the ALU. The constant 4 is used to increment the content of program counter.

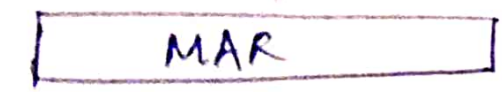
MODULE - 2

1. Inter register transfer: Do not change the information content when the binary information moves from one register to another.
   - designated by capital letter (eg: MAR, MDR, IR)
   - Represented in 4 ways

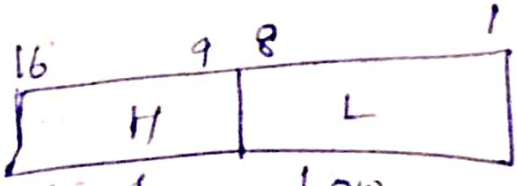
a) Register A



b) showing individual cell



c) Numbering of cell



Register declaration: DECLARE REGISTER A(8).

A ← B: Information transfer from B to A.

Content of B does not change.

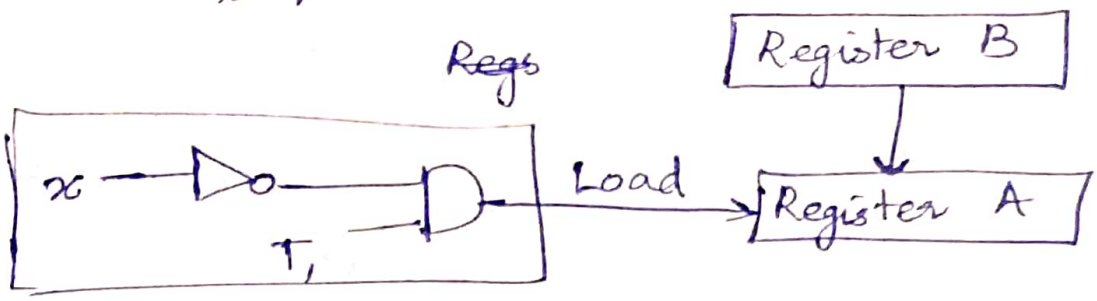
A will be lost and replaced by

Content of

new data transferred from B.

Conditional transfer occurs under a control condition :

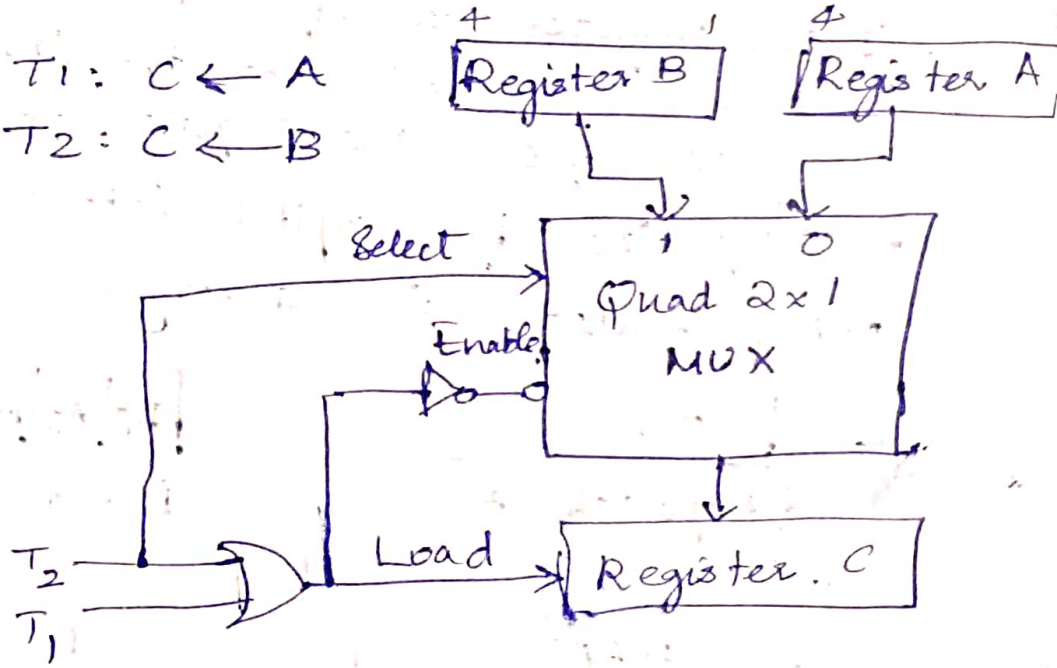
$x' T_1 : A \leftarrow B$



Destination register receives information from two sources but not at the same time.

$T1 : C \leftarrow A$

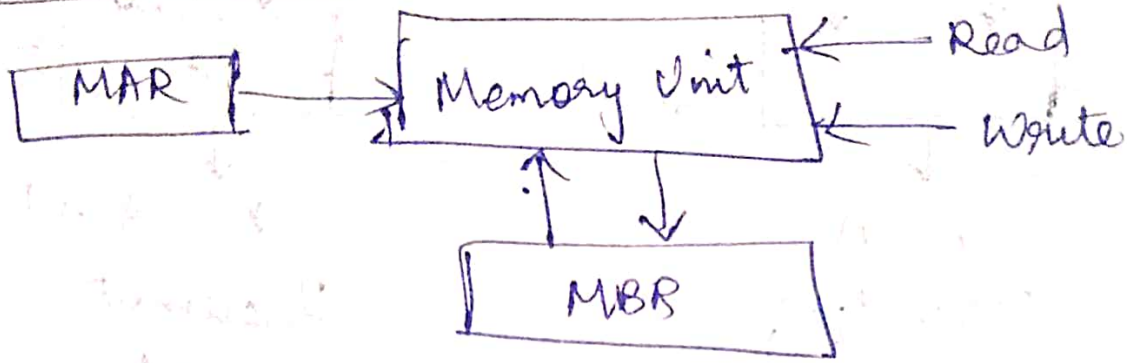
$T2 : C \leftarrow B$



BUS TRANSFER

A more efficient scheme for transferring information between registers in a multiple-register configuration is a common bus system.

Memory Transfer



Read : DR ← M[AR]

Write : M[AR] ← R)