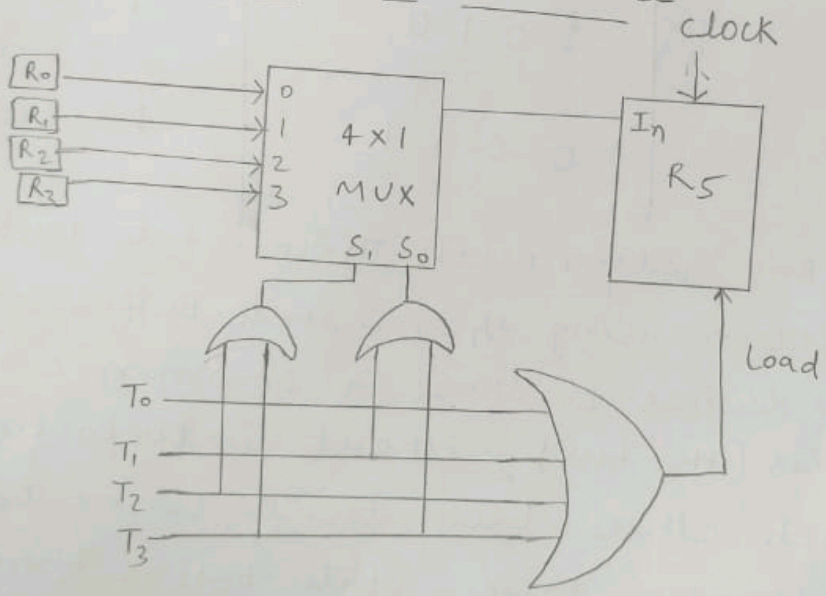
M-1

1.(a) Register Transfer Language (RTL)

– Register Transfer language (RTL) is a high-level description language used to specify the operations in digital circuit. RTL is used to design and describe the behaviours of digital systems at a level of abstraction system that is closer to the hardware.

(b) Block diagram of Hardware



$S_1 = T_2 + T_3$

$S_0 = T_1 + T_3$

$Load = T_0 + T_1 + T_2 + T_3$

(C) Connections from Timing variables to Multiplexer and Register R_5.

→ The connection can from the timing variables to the selection inputs of the multiplexer and the Laad input of register R_5 can be design as follows :-

Timing variables	MUX selection	Load Input of R_5
T_0	0 0 0 0	1
T_1	0 0 0 1	1
T_2	0 0 1 0	1
T_3	0 0 1 1	1

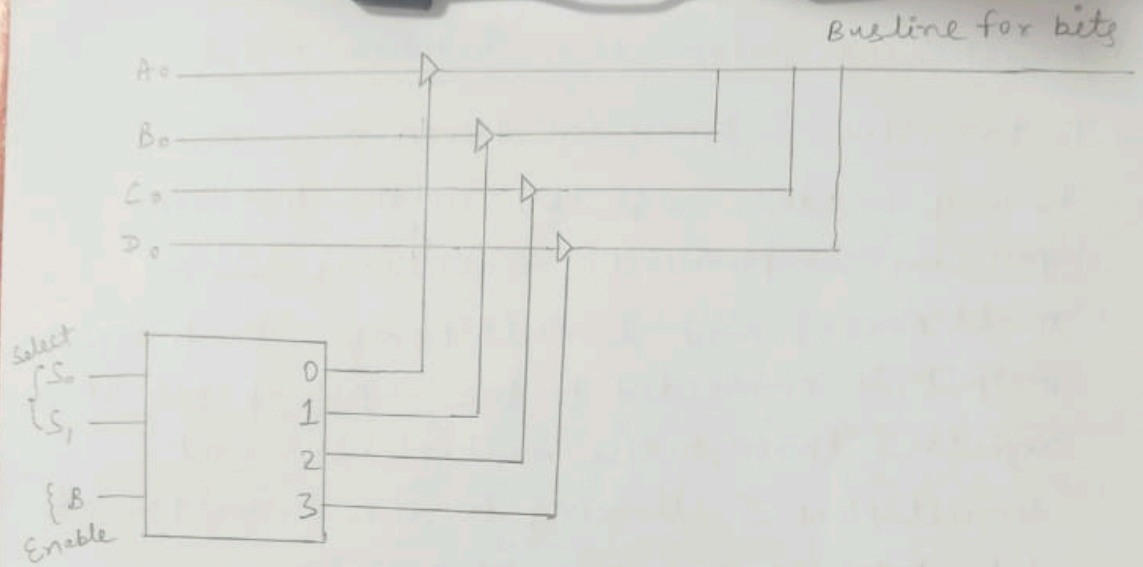
2.(a) System Bus System with Three state Buffers

- A bus system using three-state buffers allows multiple devices to share a common on set of worries (the bus) without interfacing with each other. Each device connected to the bus has three-state buffer that can be in one of three states.

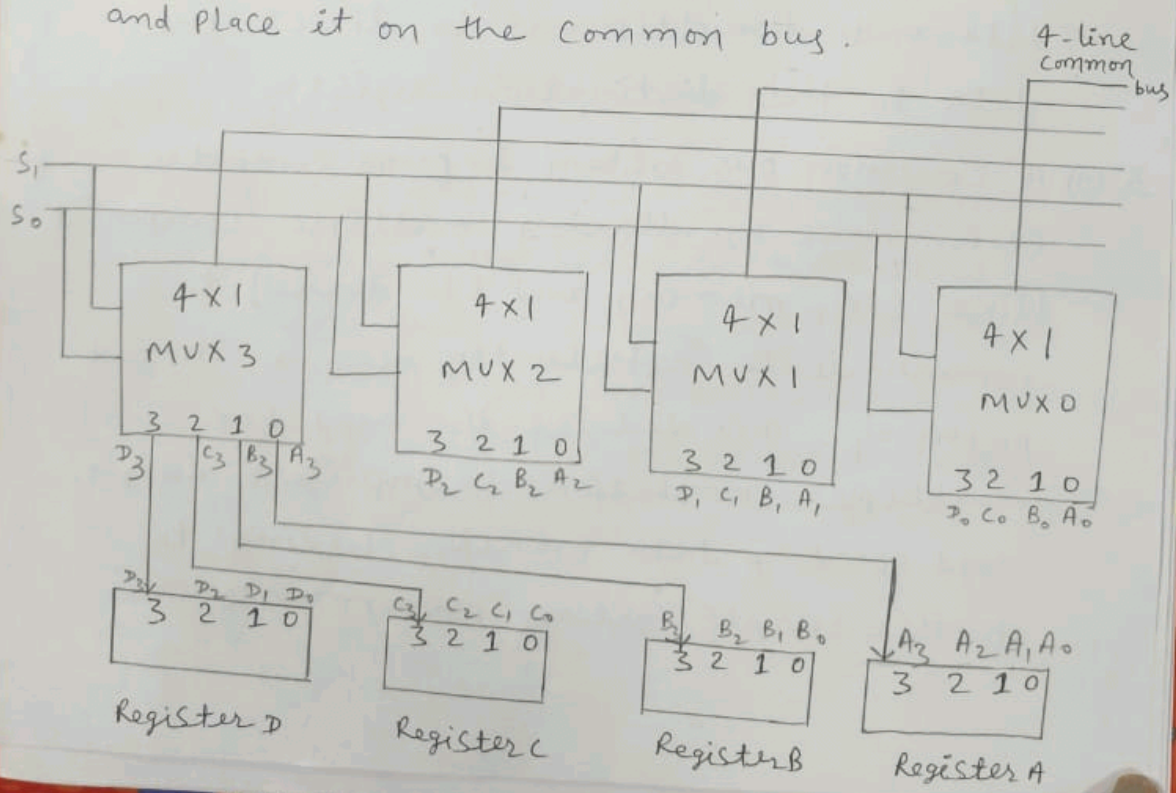
High - output is connected to a high voltage level (Logic 1)

Low - output is connected to a low voltage level (Logic 0)

High Impedence - output is disconnected from the bus, acting like an open circuit.



(b) In a bus system with four Register (R_0, R_1, R_2, R_3) and each register having four bits (0-3) we use multiplexer to select the data from one multiplexer to select the data from one register and place it on the common bus.



(C) Transferring Information between Registers

→ To transfer information from any register to any another register in the bus system, you can implement connections using multiplexers and demultiplexers. Each register output is connected to the input of all other registers through the multiplexers and demultiplexers allowing for the transfer of data between any pair of registers.

The connections required to provide a path from the outputs of any one register to the input of other registers involve using multiplexer to select the source register's data and demultiplexers to direct the data to the destination register.

3.(9) A common bus system improves computer performance by allowing multiple component (like CPU, memory and I/o devices) to communicate efficiently over a shared pathway. This reduces the need for multiple connections simplifies design, and speed up data transfer, leading to better overall system performance.

6.(a) The difference between floating-point and fixed-point representation.

1. Range :-
   - floating-point :- Can represent a much wider range of values (both very large and very small).
   - fixed-point :- Has a limited range based on the number of bits allocated for the integer and fractional parts.

2. Precision :-
   - floating-point :- offers variable precision depending on the exponent and mantisa.
   - fixed-point :- offers fixed determined by the no. of bits allocated to the fractional part.

3. Flexibility :-
   - floating-point :- More flexible can dynamically scale to represent number with having varying magnitudes.
   - fixed-point :- less flexible, better suited for specific application where numbers fall within a known range.

(b) Convert the following numbers into single precision floating – IEEE 754.

(i) $-1.1000\,1111\,001 \times 2^{1001101}$

$(1001101)_2 = (77)_{10}$

$-1.1000\,1111\,001 \times 2^{77}$

Comparing with

$1.N \times 2^{E-127}$

$-1.\underbrace{1000\,1111\,001}_{N} \times 2^{77} = 1.N \times 2^{E-127}$

$\therefore E - 127 = 77$

$E = 77 + 127$

$= 204$

$= 1100\,1100$

Single precision

1	11001100	00011110010000000000000
Sign	Exponential	Mantisa
1 bit	8 bit	23

-1010001

(ii) $1.1011110000101 \times 2^{-1010001}$

$(1010001)_2 = (81)_{10}$

Comparing with

$\Rightarrow 1.1011110000101 \times 2^{-81} = 1.N \times 2^{E-127}$

$E - 127 = -81$

$E = -81 + 127$

$= 46$

$= 101110$

Single precision

0	10111000	10111100001010000000000
1 bit	8 bits	23 bits

(C) Convert the following numbers into double precision floating-point format - IEEE 754.

(i) $-1.10001111001 \times 2^{1001101}$

$(1001101)_2 = (77)_{10}$

Comparing with

$1.N \times 2^{77}$

$-1.10001111001 \times 2^{77} = 1.N \times 2^{E-1023}$

$E - 1023 = 77$

$E = 77 + 1023$

$= 1100$

$= (10001001100)_2$

double precision

1	10001001100	1000111100100000000000000000000000000000000000000000
Sign 1-bit	Exponential 11-bit	Mantisa 52 bit

(ii) $1.10111100001 01 \times 2^{-1010001}$

$1010001 = 81$

Comparing with

$1.N \times 2^{E-1023}$

$1.10111100001 01 \times 2^{-81} = 1.N \times 2^{E-1023}$

$E - 1023 = -81$

$E = -81 + 1023$

$= 942$

$= 1110101110$

double precision

0	11101011110	1011110000101000000000000000000000000000000000000000000000
Sign 1-bit	11-bits	52-bits

7. c) (ii) 1011 0001 0010 0100

It is a BSD (Brance and save return address)

(saves the return address and transfer Control to a Subroutine)

Here 15 bit or MSB = 1 then it is indirect addressing.

0 - 11 → [0001 0010 0100]. Addressing bit

[decimal] 292

12 - 15 → [1011] op code

Hexadecimal Code : B124

7.(a) Micro operations :- Micro operations are elementary operations performed on data stored in registers. these are basic building blocks for executing instructions in a Computer System.

1) Arithmetic operations :- Addition, subtraction, increment, decrement.

Eg :- $R_3 \leftarrow R_1 + R_1$

2) Logic No — AND, OR, XOR, NOT

Eg :- $R_2 \leftarrow R_1$ AND R_0

3) Shift no — Shifts the bits of a register left or right.

Eg :- $R_1 \leftarrow$ SHR (R_1)

4) Transfer No — Transfer data from one register to another.

Eg :- $R_2 \leftarrow R_1$

(b)

(i) $(-7) + (+5)$

$-7 \rightarrow 1001$ (2's Complement)

$+5 \rightarrow 0101$

$-2 \rightarrow 1110$

$2 \rightarrow 0010$

1's $\rightarrow 1101$

$+1$

1110

∴ The 2's Complement of -2 is 1110.

(b) (+4) + (-3)

+4 → 0100

-3 → 1011 (2's Complement)

+1 → 1111

1 → 1111

2's → 0000

+1

0001

∴ The 2's of 1 is 0001.

(c)

1) Consider a 16-bit binary number
2) Here, 0-11 bit as address
3) 12-15 as opcode
4) If the MSB = 0 then it is direct addressing (Address point to the memory location directly).
5) If the MSB = 1 then it is indirect addressing (Address points to another address which contains actual data).

(i) 0001 0000 0010 0100

It is LSD (Load to special register)

↓

Loads the data into register.

0-11 [0000 0010 0100] addressing bits

↳ decimal 36

12-15 [0001] opcode.

Here MSB = 0 it is direct addressing

Hexadecimal code = 124

M-1

Register	Hexadecimal value
PC	03AfH
AR	value not provided
DR	"
AC	7EC3H
iR	932EH
E	value not provided
q	"
SC	"

10. a) Instruction fetch and execution :-

The next instruction to be fetched and executed can be determined by examining the context of the memory at the address specified by the program Counter (PC). In this case, the content of the memory at address 03AFH is 932EH. Therefore, the instruction to be fetched and executed next is 932EH.

b) Binary operation in AC :- To implement the binary operation that will be performed in the Accumulator (AC) when the instruction is executed, we need to decode the instruction 932EH. This involves understanding the specific operation code and addressing mode used in the instruction, which is not provided in the given information.

c) Contents of Registers and sequence counter

At the end of the instruction cycle, the contents of the register and the sequence counter in hexa decimals.