Computer Architecture and Organisation :

Computer Architecture → It is concerned with structure and behavior of computer as seen by user. It include information format, instruction set & techniques for addressing memory.

Computer Organization → It is concerned with the way h/w components operate & the way they are connected together to form the computer system.

Basic computer data type, Complements

- Basic computer data type ↓ Number, Conversion
  - B D H O
- Complements ↓
  - 1's 2's 9's 10's

Fixed point Representation

(There is fixed no. of digits after decimal part)

word size → bit

- Integer
  - Unsigned
    - 8 0-255
    - 16bit 0-65535
    - 0 to +ve
  - Signed
    - Signed Magnitude
    - 1's complement
    - 2's complement
- Fraction

→ When intger binary no. is +ve, the sign is represented by 0 and magnitude by a +ve binary number.

→ When no. is negative, the sign is represented by 1 but rest of no. may be represented in one of 3 possible ways.

Signed Magnitude ⇒ Left most significant bit → sign
remaining bit → magnitude

0 → +ve no.
1 → −ve no.

0 0001111 (↳ m)

Eg → +7 → 8 bit memory allocation
First change to binary 111
Add 4 0's to make (N−1) bit
0 0000111

−7 → 111
10000111

Signed 1's complement ⇒

+7 → 8 bit memory
First 111
Add 5 0's to make N bit 00000111 ✓

11111000

−7 → First 111
Add 5 0's → 00000111
sign −ve → 11111000

−256 →
100000000
↓
011111111

Signed 2's complement ⇒

2's comp. can be achieved by reversing all bits except the right most bits upto first 1 (include one)

00000101
11111011 ✓

+7 → 111
−7 → 00000111
11111001 ✓

+40 → 101000
−40 → 011000

Floating point Representation ⇒ It has two part:

→ The first part represents a signed fixed point no. called **mantissa**. Mantissa may be fraction or integer.

→ The second part of designates the position of decimal point & is called **exponent**.

→ Floating point is always interpreted to represent a no. in the following form: $\boxed{m \times r^e}$

→ mantissa ⇒ m & exponent e are physically represented in register. The radix r & the radix point position of mantissa.

+7.32

n		0
Sign	Exponent	Mantissa

It has 3 parts :- Mantissa, Base, Exponent

Number	Mantissa	Base	Exponent
5×10^2	5	10	2
212.22	21222	10	-2
55×2^6	55	2	6

IEEE 754 Floating point number representation :-

(a) **Single precision format**: 32 bit

31	30 ... 23	22 ... 0
Sign	Exponent	Mantissa
1 bit	8 bit	23 bit

(b) **Double precision format**: 64 bit

63	62 ... 52	51 ... 0
Sign	Exponent	Mantissa
1 bit	11 bit	52 bit

Q. Represent $(1259.125)_{10}$ in single & double precision format.

Ans -

Step ① **Convert decimal no. to binary number**

$(1259)_{10} = ?$

2	1259	1
2	629	1
2	314	0
2	157	1
2	78	0
2	39	1
2	19	1
2	9	1
2	4	0
2	2	0
	1	

$0.125 \times 2 = 0.250$
$0.25 \times 2 = 0.50$
$0.5 \times 2 = 1.0$
1×2

$(10011101011.001)_2$

Step ② **Normalize the number**

Single $\rightarrow (1.N)\, 2^{E-127}$
double $\rightarrow (1.N)\, 2^{E-1023}$

$1 \overbrace{0011101011.001}^{N}$

$1001110101 1001 \times 2^{10}$

Step ③ **Single precision format**

$(1.N)\, 2^{E-127}$

$2^{E-127} = 2^{10} \Rightarrow E-127 = 10.$

$E = 127+10 = 137$

$E = (10001001)_2$

31	30 … 23	22 … 0
0	10001001	1001110101100100…
1 bit	8 bit	23 bit

Step ④ **Double precision format**

$(1.N)\, 2^{E-1023}$ $\Rightarrow$ $E - 1023 = 10 \Rightarrow \boxed{E = 1033}$

$E = (10000001001)_2$

63	62 … 52	51 … 0
0	10000001001	1001110101 1001 ---- 00 ----
1	11	52

Q. Convert following floating binary number to decimal. (Single precision)

1 10000011 00000000000000000000000

Ans –

$$(-1)^s \times (1 + \text{fraction}) \times 2^{\text{exp} - 127}$$

S (31)	E (30 … 23)	M (22 … 0)
1	10000011	10000000000000000000000
1 bit	8 bit	23 bit

$2^{-1}\ 2^{-2}\ 2^{-3}$ … 2^{-23}

S = 1

$$E = 10000011 = 2^7 + 2^1 + 2^0 = 128 + 2 + 1 = 131$$

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Fraction $= 2^{-1} = 0.5$

$$(-1)^1 \times (1 + 2^{-1}) \times 2^{131-127} = \cancel{-(3)2^4}\ -(1+0.5)\, 2^4$$

$$= -1.5 \times 2^4 = -24.0 = \boxed{-24}$$

Eg – Represent Fixed pt repr of unsigned binary no. 0101110 using 4 integer bits & 3 fractional bits.

Sol^n:- $(0101.110)_2$

$$2^3 \times 0 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 + 2^{-1} \times 1 + 2^{-2} \times 1 + 2^{-3} \times 0$$

$$0 + 4 + 0 + 1 + 0.5 + 0.25 + 0 = (5.75)_{10}$$

Eg – Represent Fixed pt repr. of signed $(-7.5)_{10}$ using 4 integer & 4 fraction

$(-7.5)_{10}$

0111.1000

magnitude → 2's

$7 \to 111$ $\quad 0.5 \times 2 = 1.0$

0

coz no. is negative use 2's com.

1's → 1000.0111

+1

2's → $(1000.1000)_2$

0111.1000

$\boxed{1111.1000}$

Examples of fixed point Number Representation

Eg-1 Compute 0.75 + (−7.5) using fixed point Numbers.

Ans -

Step → ① 0.75

$(0000.1100)_2$

$0.75 \times 2 = 1.5$
$0.5 \times 2 = 1.0$
$0.0 \times 2 = 0.0$
⋮
0.0

Step - ② −7.5

0111.1000

$0.5 \times 2 = 1.0$
$0.0 \times 2 = 0.0$

2's compl. $(1000.1000)_2$

00001100
10001000

$(-6.75) \Leftarrow (1001.0100)_2$ ✓

Eg-2 Assume −43.625 convert into binary with 1 bit for sign, 15 bits for integer & 16 bits for fraction.

2	43	
2	21	1
2	10	1
2	5	0
2	2	1
	1	0

101011

$0.625 \times 2 = 1.25$
$0.25 \times 2 = 0.50$
$0.50 \times 2 = 1.0$
$0.0 \times 2 = 0.0$

10100 - - -

1000000000101011.1010000000000000

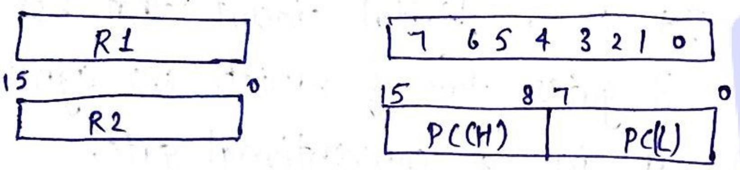
Register Transfer Language ⇒ It is the symbolic representation of notations used to specify the sequence of micro-operations.

→ Information transfer from one register to another register is called Register Transfer.

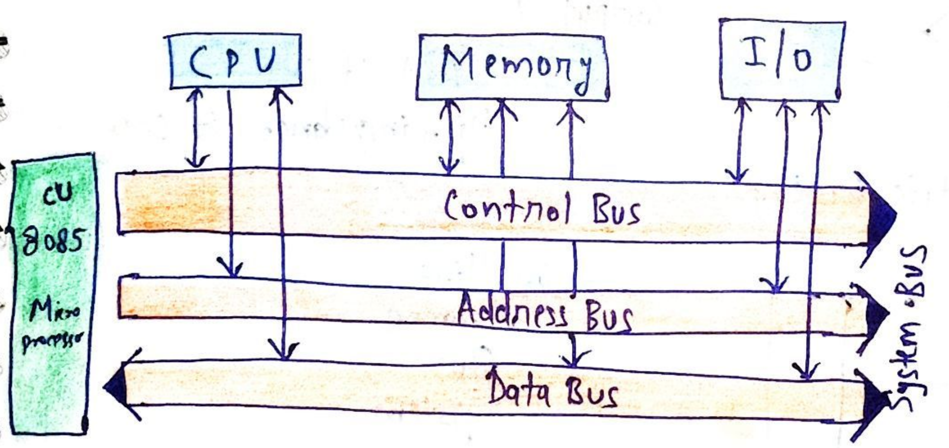
→ Registers are designated by capital letters, sometime followed by numbers (eg - A, R1, R13, IR)

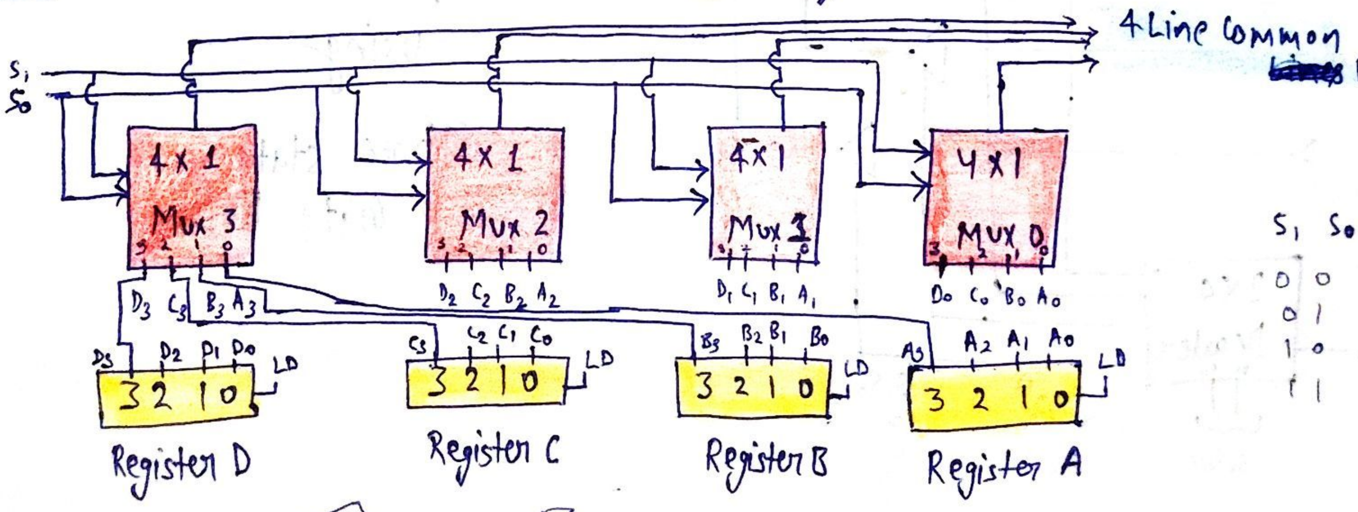
Symbols	Description	Example
Capital letters	Denotes register	MAR, R2
( )	Denotes a part of a register	R(0-7), R2(L)
$\leftarrow$	Denotes transfer of info.	R2 $\leftarrow$ R1
:	Denotes termination of control function	
,	Separates two micro-operations	A$\leftarrow$B, B$\leftarrow$A

MAR $\rightarrow$ Memory address register
PC $\rightarrow$ program counter
IR $\rightarrow$ Instruction register



BUS SYSTEM $\Rightarrow$ The bus is a comm^n channel. A bus which is used to provide comm^n b/w the major components of computer is called a System Bus.





S_1	S_0
0	0
0	1
1	0
1	1

[No. of Mux needed = No. of bits in Register
No. of I/P in Mux (size of) = No. of Register]

Q: A common bus system for 16 registers of 32 bits each.

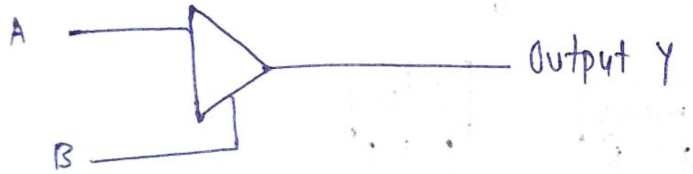
(i) Selection inputs = $2^4 = 2^n$ ⇒ $\boxed{n=4}$

(ii) 16 X 1

(iii) 32 multiplexer

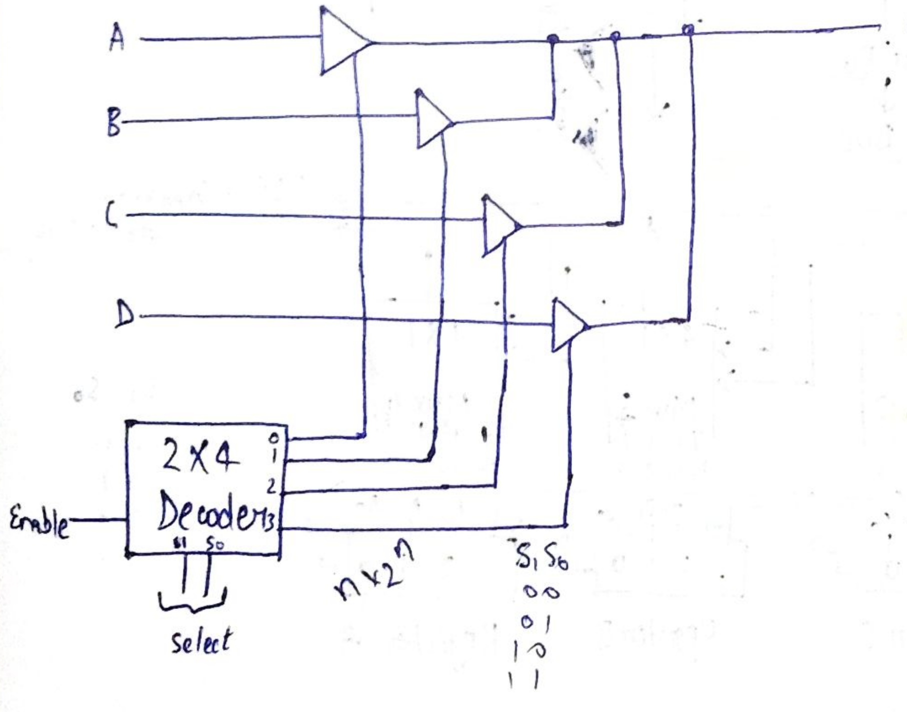
Tri-state Gates :- It is a digital circuit that has 3 gates two of which are signals equivalent to logic 1 & 0 as in a conventional gate. However, the third gate exhibits a high-impedance gate.

→ The most commonly used 3 state gates in case of bus system is a buffer gate.



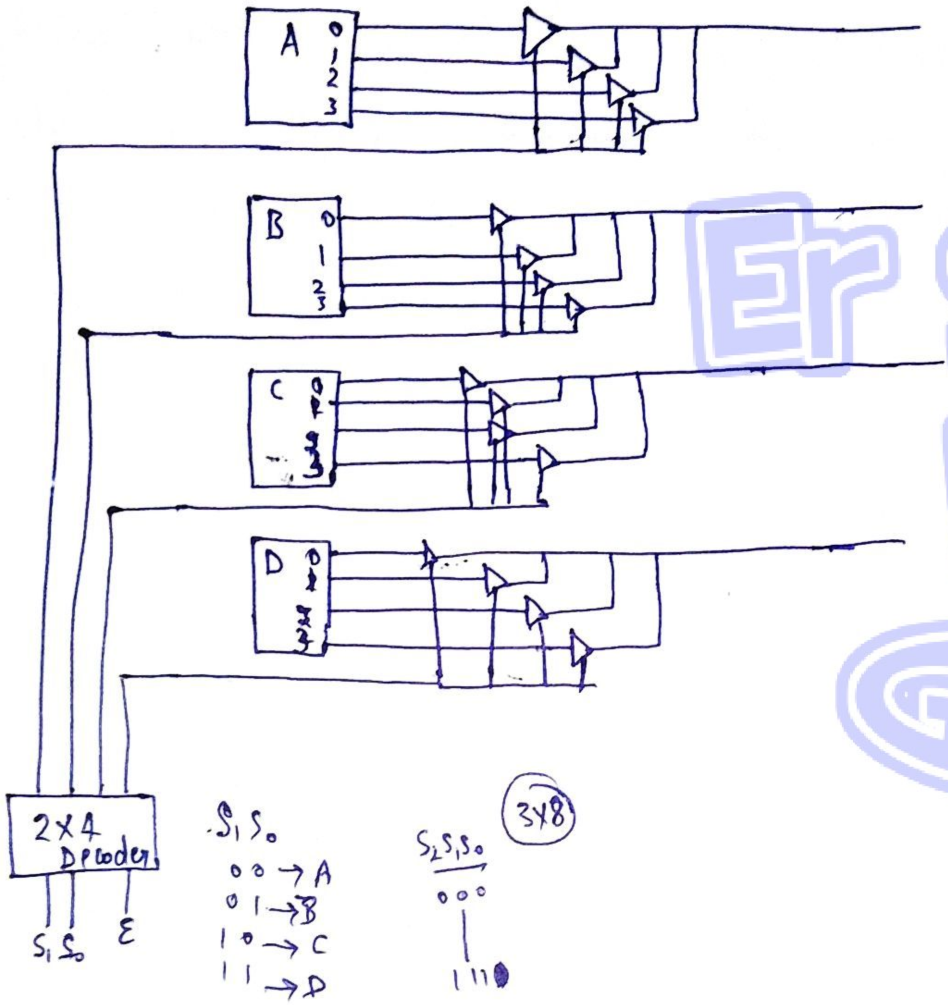
Y = A if C = 1
High impedance if C = 0

Bus line with 3 state Buffer:-



One line Common Bus system using Three state Gate Buffer

Q. Using tri-state buffer & decoder construct bus system to transfer info from 4 registers. Each register is 4 bit wide.



Micro Operations ⇒ The operation executed on data stored in registers is called Micro-operations.

eg - shift, load, clear, count.

Type of MO:-

(i) Register Transfer Micro operation:- Transfer binary Info.

(ii) Arithmetic MO:- Perform arithmetic operation on numeric data stored in registers.

(iii) Logical MO:- Perform bit manipulation operations on data stored in registers.

(iv) Shift MO:- Perform shift operations on data stored in registers.

8 bit ALU was formed by combining three 4 bit ALU's with 5 multiplexers. The 8 bit ALU is based on use of carry select line. The Four lowest bits are fed into one of the 4 bit ALU's.

The O/p of selectable ALU's are multiplexed together forming the upper & lower 4 bits & carry out for 8 bit ALU.

SAHIL KHAN

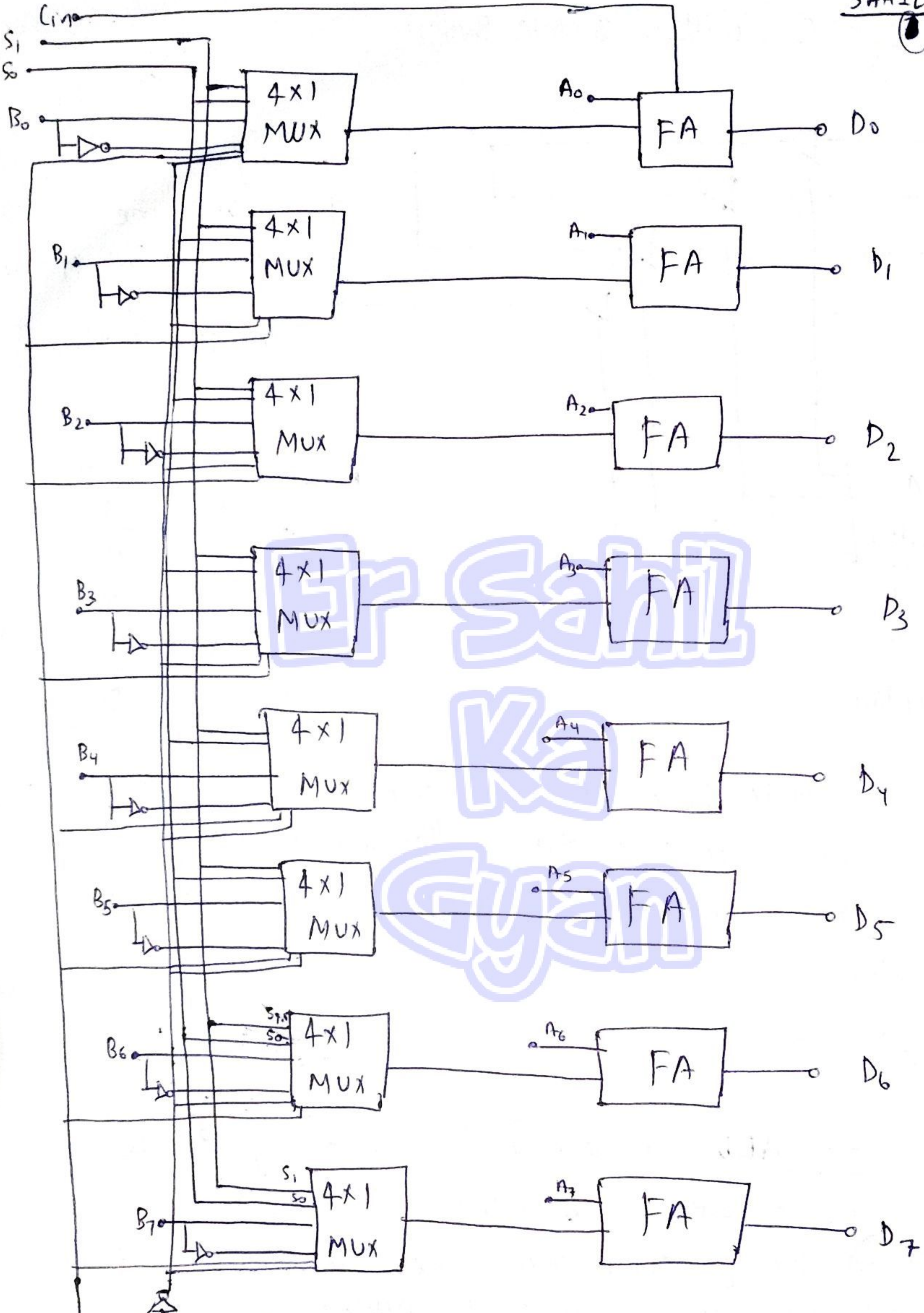


fig:- 8 Bit Arithmetic Circuit

Introduction of Assembler:-

Assembler is a program for converting instructions written in low-level assembly code into relocatable machine code & generating along information for the loader.

'If assembler do all this work in one scan then it is called multiple pass assembler.
Here assembler divide these tasks in 2 passes

Pass 1 :- Define symbols & literals and remember them in symbol table & literal table respectively.
keep track of location counter.
Process pseudo-operation

Pass 2 :- Generate object code by converting symbolic op-code into respective numeric op-code.
Generate data for literals and look for values of symbols.

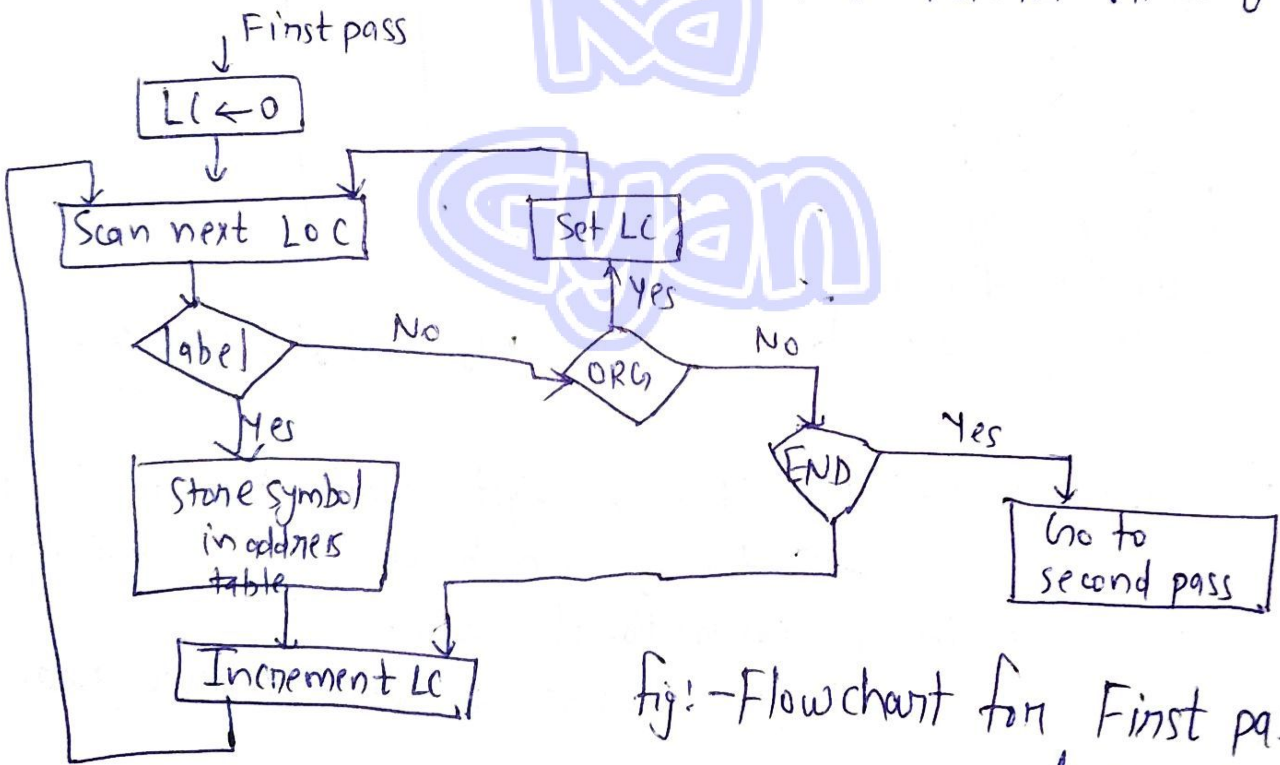
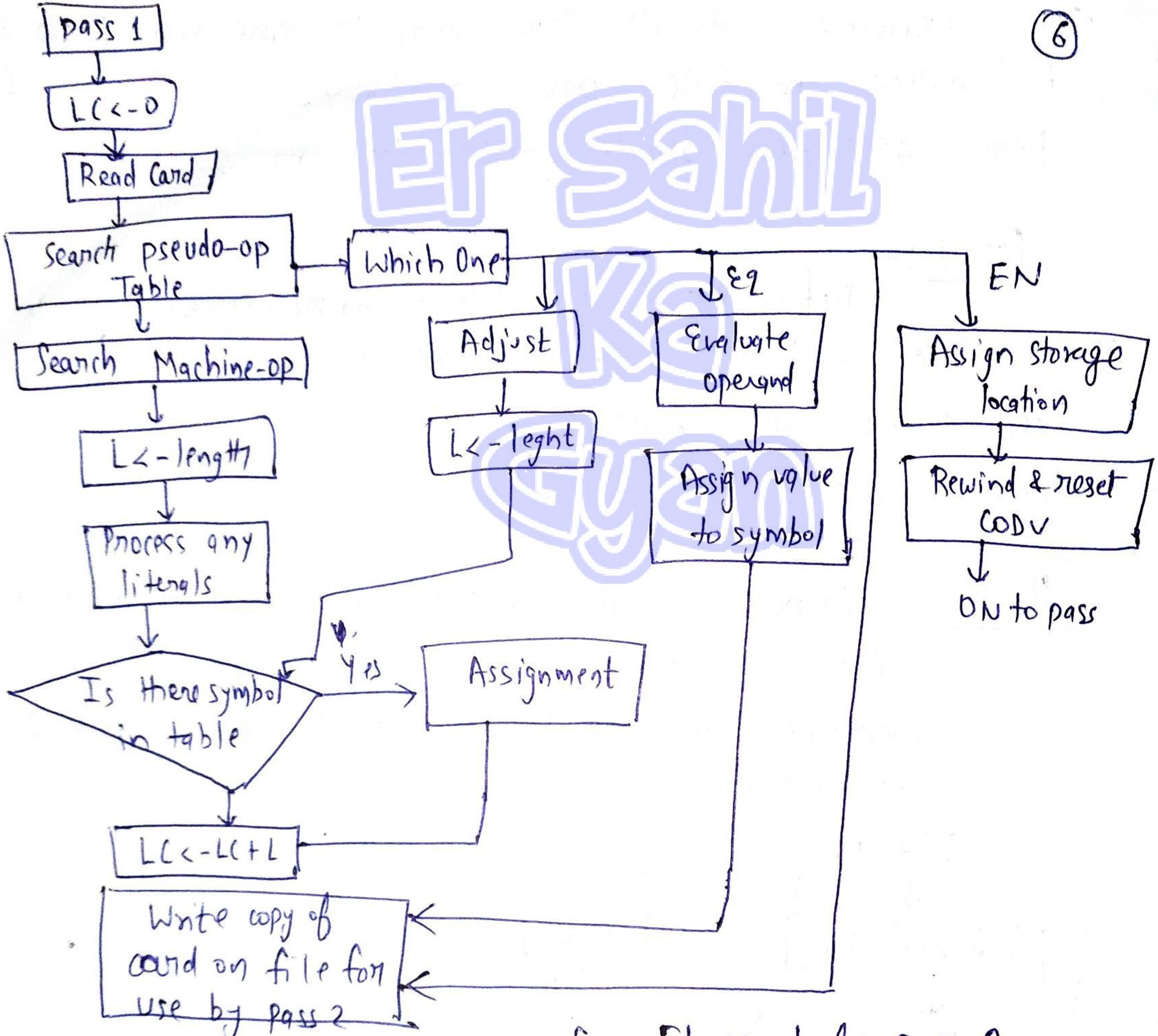


fig:- Flowchart for First pass of assembler



fig! - Flowcart for pass 2