ASSIGNMENT-0

જય સ્વામિનારાયણ
Dt.: / / Pg.no.:

1. Convert the following binary numbers to decimal.

(a) $(101110)_2 = (1\times2^5) + (0\times2^4) + (1\times2^3) + (1\times2^2) + (1\times2^1) + (0\times2^0)$
$= 32 + 0 + 8 + 4 + 2 + 0$
$= (46)_{10}$

(b) $(1110101)_2 = (1\times2^6) + (1\times2^5) + (1\times2^4) + (0\times2^3) + (1\times2^2) + (0\times2^1) + (1\times2^0)$
$= 64 + 32 + 16 + 0 + 4 + 0 + 1$
$= (117)_{10}$

(c) $(110110100)_2 = (1\times2^8) + (1\times2^7) + (0\times2^6) + (1\times2^5) + (1\times2^4) + (0\times2^3) + (1\times2^2) + (0\times2^1) + (0\times2^0)$
$= 256 + 128 + 0 + 32 + 16 + 0 + 4 + 0 + 0$
$= (436)_{10}$

2. Convert the following numbers with the indicated bases to decimal.

(a) $(12121)_3 = (1\times3^4) + (2\times3^3) + (1\times3^2) + (2\times3^1) + (1\times3^0)$
$= 81 + 54 + 9 + 6 + 1$
$= (151)_{10}$

(b) $(4310)_5 = (4 \times 5^3) + (3 \times 5^2) + (1 \times 5^1) + (0 \times 5^0)$
$= 500 + 75 + 5 + 0$
$= (580)_{10}$

(c) $(50)_7 = (5 \times 7^1) + (0 \times 7^0)$
$= 35 + 0$
$= (35)_{10}$

(d) $(198)_{12} = (1 \times 12^2) + (9 \times 12^1) + (8 \times 12^0)$
$= 144 + 108 + 8$
$= (260)_{10}$

3. Convert the following decimal numbers to the bases indicated.

(a) 7562 to octal

	Quotient	Remainder	Coefficient
7562/8	945	2	$a_0 = 2$
945/8	118	1	$a_1 = 1$
118/8	14	6	$a_2 = 6$
14/8	1	6	$a_3 = 6$
1/8	0	1	$a_4 = 1$

$(7562)_{10} = (16612)_8$

(b) 1938 to hexadecimal

	Quotient	Remainder	Coefficient
1938/16	121	2	$a_0 = 2$
121/16	7	9	$a_1 = 9$
7/16	0	7	$a_2 = 7$

$(1938)_{10} = (792)_{16}$

(c) 175 to binary

	Quotient	Remainder	Coefficient
175/2	87	1	$a_0 = 1$
87/2	43	1	$a_1 = 1$
43/2	21	1	$a_2 = 1$
21/2	10	1	$a_3 = 1$
10/2	5	0	$a_4 = 0$
5/2	2	1	$a_5 = 1$
2/2	1	0	$a_6 = 0$
1/2	0	1	$a_7 = 1$

$(175)_{10} = (10101111)_2$

4. Show the value of all bits of a 12 bit register that hold the number equivalent to decimal 215 in (a) Binary (b) Binary coded octal (c) Binary coded hexadecimal (d) Binary coded decimal

Ans: (a) Binary :

	Quotient	Remainder	Coefficient
215/2	107	1	$a_0 = 1$
107/2	53	1	$a_1 = 1$
53/2	26	1	$a_2 = 1$
26/2	13	0	$a_3 = 0$
13/2	6	1	$a_4 = 1$
6/2	3	0	$a_5 = 0$
3/2	1	1	$a_6 = 1$
1/2	0	1	$a_7 = 1$

$(215)_{10} = (11010111)_2$

In 12-bit register, it is stored as "0000 1101 0111"

(b) Binary coded Octal :

	Quotient	Remainder	Coefficient
215/8	26	7	$a_0 = 7$
26/8	3	2	$a_1 = 2$
3/8	0	3	$a_2 = 3$

$(215)_{10} = (327)_8$

In 2-bit register it is stored as "000 011 010 111"

(c) Binary coded hexadecimal :

	Quotient	Remainder	Coefficient
215/16	13	7	$a_0 = 7$
13/16	0	13	$a_1 = 13 = D$

$(215)_{10} = (D7)_{16}$

In 12-bit register, it is stored as "0000 1101 0111"

(d) Binary Coded decimal

	Quotient	Remainder	Coefficient
215/2	107	1	$a_0 = 1$
107/2	53	1	$a_1 = 1$
53/2	26	1	$a_2 = 1$
26/2	13	0	$a_3 = 0$
13/2	6	1	$a_4 = 1$
6/2	3	0	$a_5 = 0$
3/2	1	1	$a_6 = 1$
1/2	0	1	$a_7 = 1$

$(215)_{10} = (110\ 1011\ 1)_2$

In 12-bit register it is stored as "0000 1101 0111"

5. Perform the arithmetic operation (+42) + (-13) and (-42) - (-13) in binary using signed 2's complement representation for negative numbers.

	Quotient	Remainder	Coefficient
42/2	21	0	$a_0 = 0$
21/2	10	1	$a_1 = 1$
10/2	5	0	$a_2 = 0$
5/2	2	1	$a_3 = 1$
2/2	1	0	$a_4 = 0$
1/2	0	1	$a_5 = 1$

$(42)_{10} = (101010)_2$

	Quotient	Remainder	Coefficient
13/2	6	1	$a_0 = 1$
6/2	3	0	$a_1 = 0$
3/2	1	1	$a_2 = 1$
1/2	0	1	$a_3 = 1$

$(13)_{10} = (1101)_2$

→ (+42) + (-13)

2's complement of 001101 is,

```
  110010    1's complement of 001101
+      1    Add 1
--------
  110011
```

```
    1          carry
  101010
+ 110011       2' complement of (-13)10
1011101
```

Here, carry generated on MSB is ignored

$\therefore (+42)_{10} + (-13)_{10} = (101010)_2 + (-001101)_2$

$= (011101)_2$

$= (29)_{10}$

→ (-42) - (-13)

= 13 - 42

$= (001101)_2 - (101010)_2$

2's complement of 101010 is,

```
     1          carry
  010101        1's complement of 101010
+      1        Add 1
  010110
```

```
   111          carry
  001101        (13)10
+ 010110        2's complement of (42)10
  100011
```

2's complement of 100011 ;

```
  011100        1's complement of 100011
+      1        Add 1
  011101
```

$-(01110 1)_2 = (-29)_{10}$

$\therefore (-42)_{10} - (-13)_{10} = (-29)_{10}$

6. Perform the arithmetic operations $(+70)+(+80)$ and $(-70)+(-80)$ with binary number in signed 2's complement representation. Use eight bits to accomodate each number together with its sign. Show that overflow occur in both cases, that the last two carries are unequal and that there is a sign reversal.

Sol^n:

	Quotient	Remainder	Coefficient
70/2	35	0	$a_0 = 0$
35/2	17	1	$a_1 = 1$
17/2	8	1	$a_2 = 1$
8/2	4	0	$a_3 = 0$
4/2	2	0	$a_4 = 0$
2/2	1	0	$a_5 = 0$
1/2	0	1	$a_6 = 1$

$(70)_{10} = (1000110)_2$

	Quotient	Remainder	Coefficient
80/2	40	0	$a_0 = 0$
40/2	20	0	$a_1 = 0$
20/2	10	0	$a_2 = 0$
10/2	5	0	$a_3 = 0$

5/2	2	1	$a_4 = 1$
2/2	1	0	$a_5 = 0$
1/2	0	1	$a_6 = 1$

$(80)_{10} = (1010000)_2$

→ (+70) + (+80)

1 carry

0 1 0 0 0 1 1 0
+0 1 0 1 0 0 0 0
1 0 0 1 0 1 1 0

Here, overflow occurs as the sign bit of two nos (+70) and (+80) is '0' in 8-bit representation. But, the sign bit of the ans is '1'.

→ (-70) + (-80)

2's complement of 0100 0110 is 1011100 1 (1)
+ 1
1011010

2's complement of 0101 0000 is 1010111 1 (1)
+ 1
1011000

$$\begin{array}{r} 10111010 \\ + \ 10110000 \\ \hline \boxed{1}\,01101010 \end{array}$$ Carry generated.

If carry is ignored, it means (-80)+(-70) =106 which is wrong. So, overflow occurs.

In case I, Cin = 1 but Cout = 0
In case II, Cin = 0 but Cout = 1

In both cases, the two carries are not equal so an overflow occured.

7. Represent (8620)10 in (a) binary (b) Excess-3 code and (c) 2421 code

Sol^{n}: (a) Binary

	Quotient	Remainder	Coefficient
8620/2	4310	0	$a_0 = 0$
4310/2	2155	0	$a_1 = 0$
2155/2	1077	1	$a_2 = 1$
1077/2	538	1	$a_3 = 1$
538/2	269	0	$a_4 = 0$
269/2	134	1	$a_5 = 1$
134/2	67	0	$a_6 = 0$
67/2	33	1	$a_7 = 1$
33/2	16	1	$a_8 = 1$

16|2	8	0	$a_9 = 0$
8|2	4	0	$a_{10} = 0$
4|2	2	0	$a_{11} = 0$
2|2	1	0	$a_{12} = 0$
1|2	0	1	$a_{13} = 1$

$(8620)_{10} = (10000110101100)_2$

(b) Excess - 3 code

$(8620)_{10} = (1000\ 0110\ 0010\ 0000)_{BCD}$

$= (1011\ 1001\ 0101\ 0011)_{Ex-3}$

(c) 2421 code :

$(8620)_{10} = (1110\ 1100\ 0010\ 0000)_{2421\text{-code}}$

8. Convert the hexadecimal number F3A7C2 to binary and Octal.

Sol^n:

F 3 A 7 C 2

$= (1111\ 0011\ 1010\ 0111\ 1100\ 0010)_2$

111	100	111	010	011	111	000	010
↓	↓	↓	↓	↓	↓	↓	↓
7	4	7	2	3	7	0	2

$(F3A7C2)_{16} = (1111\ 0011\ 1010\ 0111\ 1100\ 0010)_2$

$(F3A7C2)_{16} = (74723702)_8$

જય સ્વામિનારાયણ
Dt.: / / Pg.no.:

9. Perform the arithmetic operation (+42) + (-13) and (-42) - (-13) in binary using signed 2's complement representation for negative number.

Sol^{n}: $(42)_{10} = (101010)_2$
$(13)_{10} = (001101)_2$

→ (+42) + (-13)
2's complement of 001101 is $\begin{array}{r} 110010 \\ +\quad\ 1 \\ \hline 110011 \end{array}$

$$\begin{array}{rl} & 1 \quad \text{carry} \\ & 101010 \\ + & 110011 \\ \hline \boxed{1} & 011101 \end{array}$$ carry generated

$(+42)_{10} + (-13)_{10} = (+101010)_2 + (-001101)_2$
$= (+011101)_2$
$= (29)_{10}$

→ (-42) - (-13)
2's complement of 101010 is $\begin{array}{r} 010101 \\ +\quad\ 1 \\ \hline 010110 \end{array}$

2's complement of 001101 is $\begin{array}{r} 110010 \\ +\quad\ 1 \\ \hline 110011 \end{array}$

2's complement of 11001 is 00110 + 1 = 00111

```
  1 1 1
  0 1 0 1 1 0
+ 0 0 1 1 0 1
  1 0 0 0 1 1
```

2's complement of 10011 is 01100 + 1 = 01101

$(-42)_{10} - (-13)_{10} = (-101010)_2 - (-001101)_2$
$= -(011101)_2$
$= -29$
$= +(29)_{10}$

10. Perform the subtraction with the following unsigned numbers by taking the 2's complement of the subtrahend.

(a) 11010 - 10000

2's complement of 10000 is :

```
  1 1 1 1   carry
  0 1 1 1   1's complement
+       1   add 1
  1 0 0 0 0
```

$$\begin{array}{r} 11010 \\ +\ 10000 \\ \hline \boxed{1}\,01010 \end{array}$$

$(11010)_2 - (10000)_2 = (01010)_2$

(b) 11010 - 01101

2's complement of 01101 is $\begin{array}{r} 10010 \\ +\quad\ 1 \\ \hline 10011 \end{array}$

$$\begin{array}{r} \ 1 \quad \text{carry} \\ 11010 \\ +\ 10011 \\ \hline \boxed{1}\,01101 \end{array}$$

$(11010)_2 - (01101)_2 = (01101)_2$

ASSIGNMENT-1

જય સ્વામિનારાયણ
Dt.: / / Pg no.:

1. A digital computer has a common bus system for 16 registers of 32 bit each. The bus is constructed with multiplexers.

(a) How many selection inputs are there in each multiplexer?

→ 4 selection inputs are there in each multiplexer.

(b) What size of multiplexers are needed?

→ 16 x 1 sized multiplexer is needed.

(c) How many multiplexers are there in the bus?

→ 32 multiplexers are there in the bus.

2. The following transfer statements specify a memory. Explain the memory operation in each case.

(a) $R_2 \leftarrow M[AR]$

→ Read memory word specified by the address in AR into register R_2.
i.e. It would transfer the contents of Memory word that has the address specified by AR into register R_2.

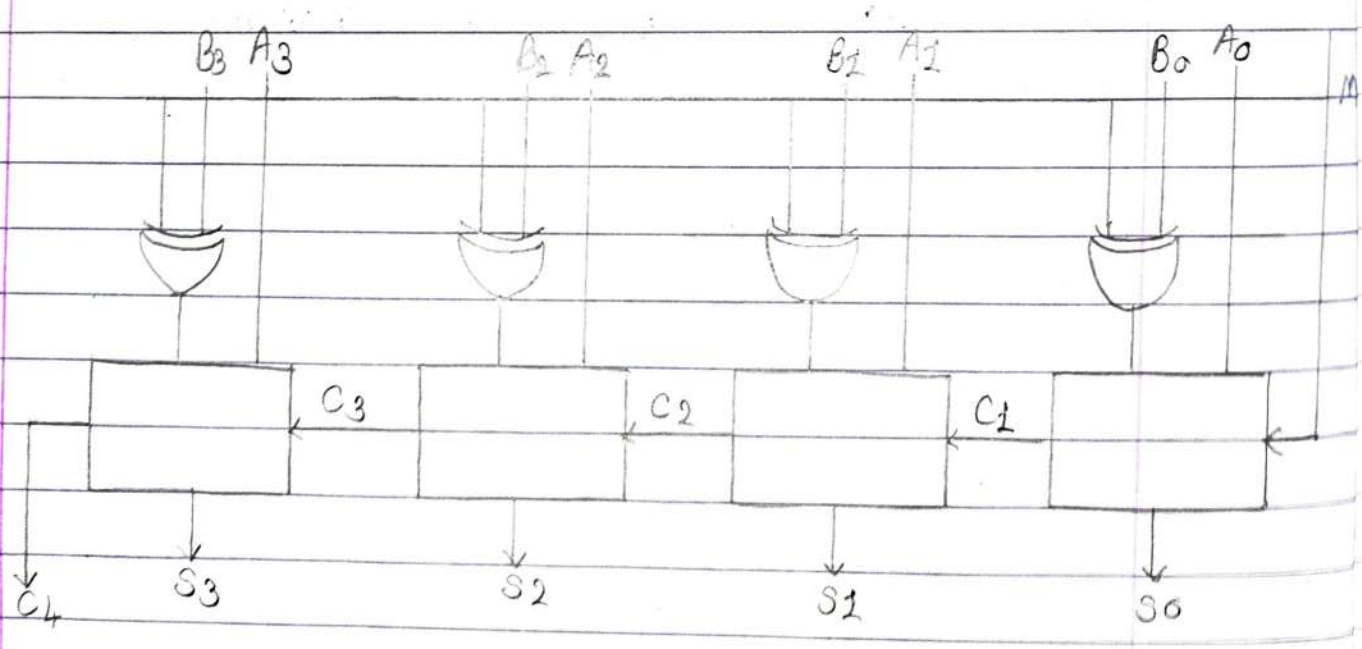
(b) $M[AR] \leftarrow R_3$

→ Write content of register R_3 into the memory word specified by the address in AR.

(c) $R_5 \leftarrow M[R_5]$

→ Read memory word specified by the address in R_5 and transfer context to the same register R_5 (over writing the previous value)

3. Draw the Binary adder - subtractor circuit which has the following values for input mode M and data inputs A and B. In each case, determine the values of the output : S_3, S_2, S_1, S_0 and C_4.



	M	A	B	Operation
(a)	0	0111	0110	7 + 6
(b)	0	1000	1001	8 + 9
(c)	1	1100	1000	12 - 8
(d)	1	0101	1010	5 - 10
(e)	1	0000	0001	0 - 1

(a) $C_0 = 0$

carry: 1 1

$$\begin{array}{r} 0111 \\ +\ 0110 \\ \hline S = 1101 \end{array} \Rightarrow C_4 = 0$$

(b) $C_0 = 0$

$$\begin{array}{r} 1000 \\ +\ 1001 \\ \hline \boxed{1}0001 \end{array} \quad S = 0001 \ , \ C_4 = 1$$

(c) $C_0 = 1$

$$\begin{array}{r} 1100 \\ +\ 1000 \\ \hline \boxed{1}0100 \end{array} \quad S = 0100 \ , \ C_4 = 1$$

(d) $C_0 = 1$

carry: 1

$$\begin{array}{r} 0101 \\ +\ 0110 \\ \hline 1011 \end{array} \quad S = 1011 \ , \ C_4 = 0$$

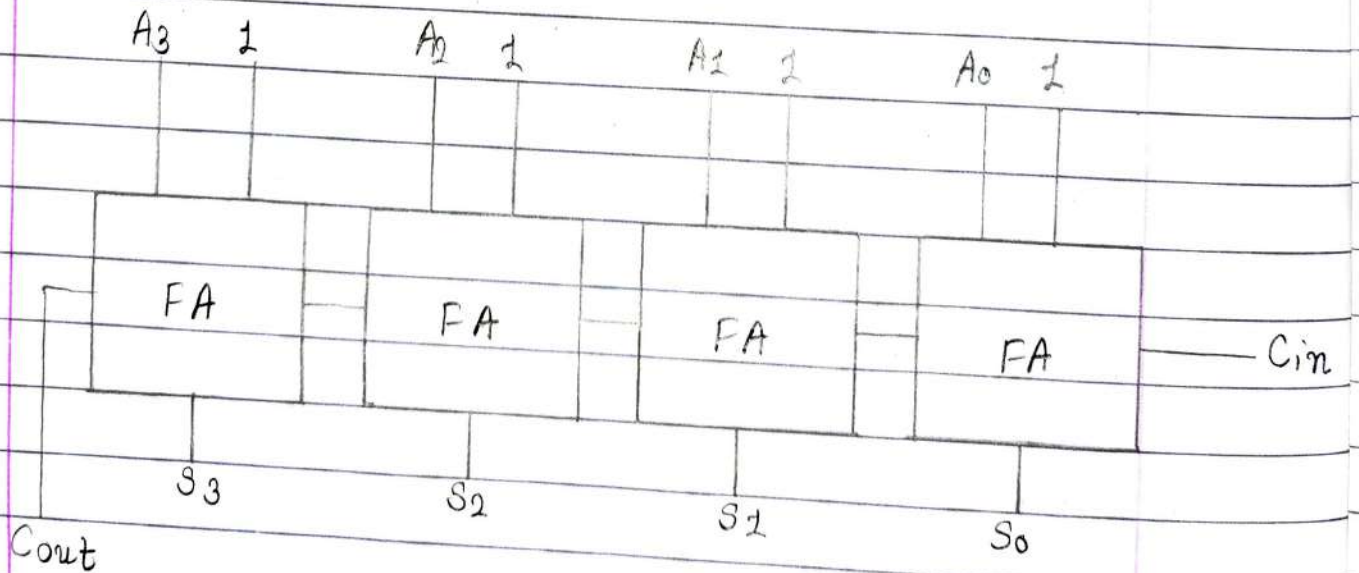
(e) $C_0 = 1$

$$\begin{array}{r} 0\,0\,0\,0 \\ +\ 1\,1\,1\,1 \\ \hline 1\,1\,1\,1 \end{array} \qquad S = 1111 \;, C_4 = 0$$

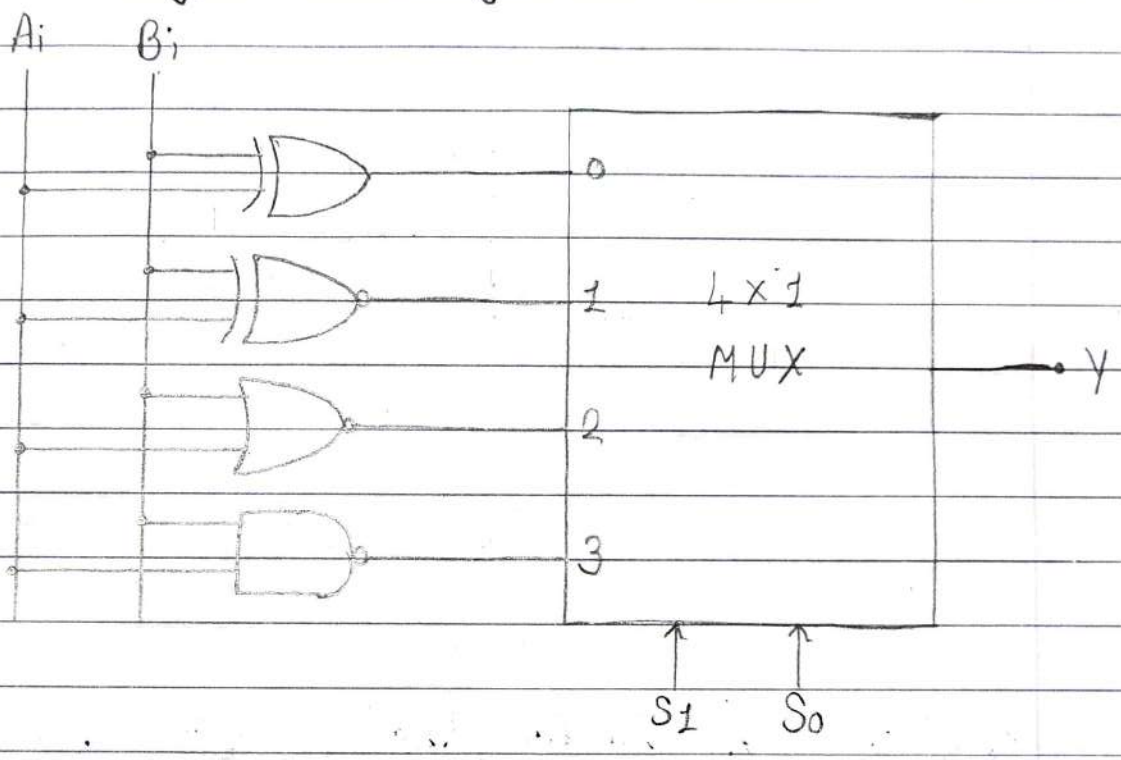
	S_3	S_2	S_1	S_0	C_4
a	1	1	0	1	0
b	0	0	0	1	1
c	0	1	0	0	1
d	1	0	1	1	0
e	1	1	1	1	0

4. Design a 4-bit combinational circuit decremented using four-full adder circuits

Sol^n: $A-1 = A + 2$'s complement of 1
$= A + 1111$



5. Design a digital circuit that performs the four logic operations of exclusive-OR, exclusive-NOR, NOR and NAND. use two selection variables. Show the logic diagram of one typical stage.



S_1	S_0	Y	Operation
0	0	$A \oplus B$	X-OR
0	1	$\overline{A \odot B}$	X-NOR
1	0	$\overline{A+B}$	NOR
1	1	$\overline{AB}$	NAND

→ Logic diagram of one typical stage

Let $A_i = 1$

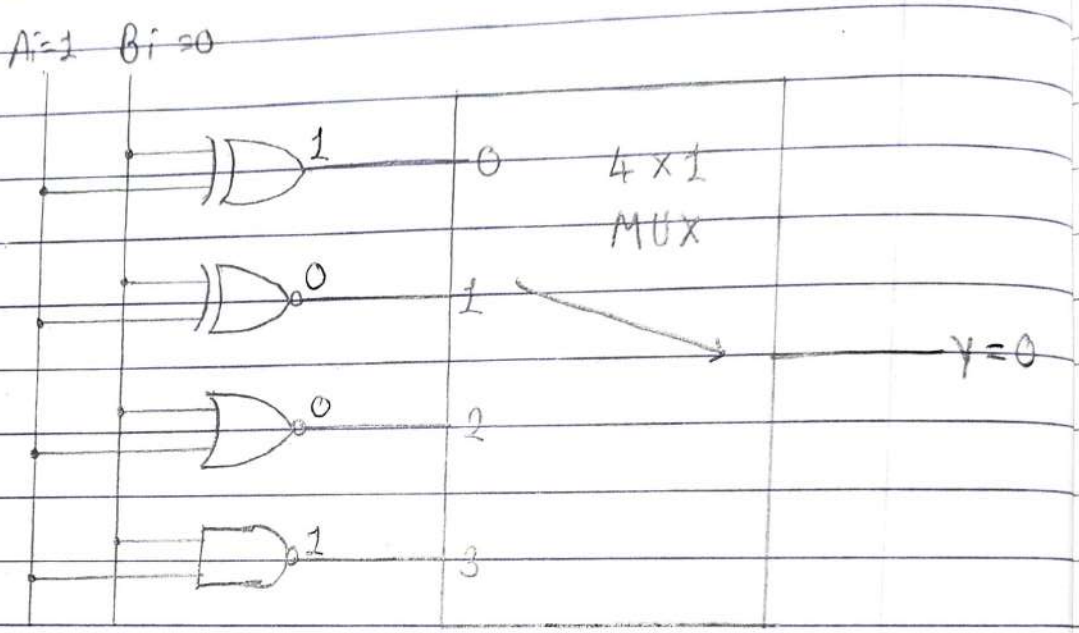
$B_i = 0$

and $S_1 = 0$, $S_0 = 1$ $\Rightarrow$ x-NOR gate is selected.

$Y = A_i \odot B_i$

$= 1\ 0\ 0$

$\therefore Y = 0$



6. Register A holds the 8 bit binary 11011001. Determine the B operand and the logic microoperation to be performed in order to change the value in A to:
(a) 01101101 (b) 11111101

Soln: (a)

$A \rightarrow$ 1 1 0 1 1 0 0 1
$B \rightarrow$ 1 0 1 1 0 1 0 0
$A^+ \rightarrow$ 0 1 1 0 1 1 0 1

$A^+ = A \oplus B$ [X-OR] Selective complement

જય સ્વામિનારાયણ
Dt.: / / Pg.no.:

(b) $A \rightarrow 11011001$

$B \rightarrow 11111101$

$A^+ \rightarrow 11111101$

$A^+ = A + B$ [OR] → Selective set

7. The 8-bit registers AR, BR, CR and DR initially have the following values

AR = 11110010 CR = 10111001

BR = 11111111 DR = 11101010

Determine the 8-bit values in each register after the execution of the following sequence of microoperations.

$AR \leftarrow AR + BR$

$CR \leftarrow CR \wedge DR$, $BR \leftarrow BR + 1$

$AR \leftarrow AR - CR$

Sol^n:

$AR + BR \Rightarrow$

```
  11111111
  11110010
+ 11111111
 [1]11110001
```

$AR = 11110001$

$CR \wedge DR \Rightarrow$

```
  10111001
^ 11101010
  10101000
```

$CR = 10101000$

$$\begin{array}{r} 11111111 \\ BR + 1 \Rightarrow \quad 11111111 \\ +\quad\quad\quad\quad 1 \\ \hline \boxed{1}\,00000000 \end{array}$$

$BR = 00000000$

$$\begin{array}{r} \cdot 1 \quad \cdot 11 \\ AR - CR \Rightarrow \quad 11110010 \\ +\,01000111 \\ \hline \boxed{1}\,00111001 \end{array}$$

$AR = 00111001$

Micro operation	AR	BR	CR	DR
Initially	11110010	11111111	10111001	11101010
AR ← AR+BR	11110001	11111111	10111001	11101010
CR ← CR ^ DR	11110001	11111111	10101000	11101010
BR ← BR+1	11110001	00000000	10101000	11101010
AR ← AR-CR	01001001	00000000	10101000	11101010

8. An 8 bit register contains the binary value 10011100. What is the register value after

an arithmetic shift right 2. Starting from the initial number 10011100. determine the register value after an arithmetic shift left, and state whether there is an overflow.

Soln: $R = 10011100$

→ Register value after right shift.

$R = 11001110$

→ Register value after left shift.

$R = 00111000$

There is an overflow condition because a negative no. changes to positive no.

9. Represent the following conditional control statement by two register transfer statements with control functions.

if $CP = 1$ then $R_1 \leftarrow R_2$ else if $CQ = 1$ then $R_1 \leftarrow R_3$.

Soln:

$P: R_1 \leftarrow R_2$

$P'Q: R_1 \leftarrow R_3$

10. What is wrong with the following register transfer statements ?

(a) $xT: AR \leftarrow AR', AR \leftarrow 0$

→ We cannot perform two microoperations 'complement and clear' at same time on same register.

(b) y T : $R_1 \leftarrow R_2$, $R_1 \leftarrow R_3$

→ We cannot transfer two different values (R_2 and R_3) to the same register (R_1) at same time.

(c) zT : $PC \leftarrow AR$, $PC \leftarrow PC + 1$

→ We cannot transfer a new value into register (PC) and increment the original value by one at the same time. Both are conflicting microoperations.

ASSIGNMENT - 2

જય સ્વામિનારાયણ
Dt.: / / Pg.no.:

1. A computer uses a memory unit with 256K words of 32 bit each. A binary instruction code is stored in one word of memory. The instruction has four parts: an indirect bit, an operation code, a register code part to specify one of 64 registers and an address part.

(a) How many bits are there in the operation code, the register code part, and the address part?

→ Indirect = 1 bit
Address 2^8 (256 KB) * 2^{10} (1024 byte / KB)
$= 2^{18} \Rightarrow 18$ bits
Register (64 registers) = $2^6 \Rightarrow 6$ bits
Opcode 32 - 1 - 18 - 6 = 7 bits

(b) Draw the instruction word format and indicate the number of bits in each part.

→ Instruction Word Format

Indirect	Op-code	Register	Address
1	7	6	18

32 bit

2. What is the difference between a direct and indirect address instruction? How many references to memory are needed for each type of instruction to bring an operand into a processor register?

	Direct Address Instruction	Indirect Address Instruction
1.	Address field contains the effective address of operand.	Address field contains reference of effective address
2.	Requires only one memory reference	Require two memory references
3.	Fast addressing	Slower than direct addressing mode
4.	No further classification.	Further classified into two categories.
5.	No further calculation is required to perform the operation.	Require further calculation to find the effective address.

→ A direct address instruction needs two refereneces to memory. (1) Read instruction

(2) Read operand

→ An indirect address instruction needs three references to memory (1) Read instruction (2) Read effective address (3) Read operand.

3. What are the two instruction needed in the basic computer in order to set the E flip flop to 1 ?

Soln: CLE clear E
CME complement E

4. An output program resides in memory starting from address 2300. It is executed after the computer recognizes an interrupt when FGO becames a 1 (While IEN = 1)

5. Explain following terms :

(a) Instruction code

→ An instruction code is a group of bits that instruct the computer to perform a specific operation.

(b) Operation code

→ The operation code of an instruction is a group of bits that define operations such group of bits that as addition, subtraction, shift, complement etc.

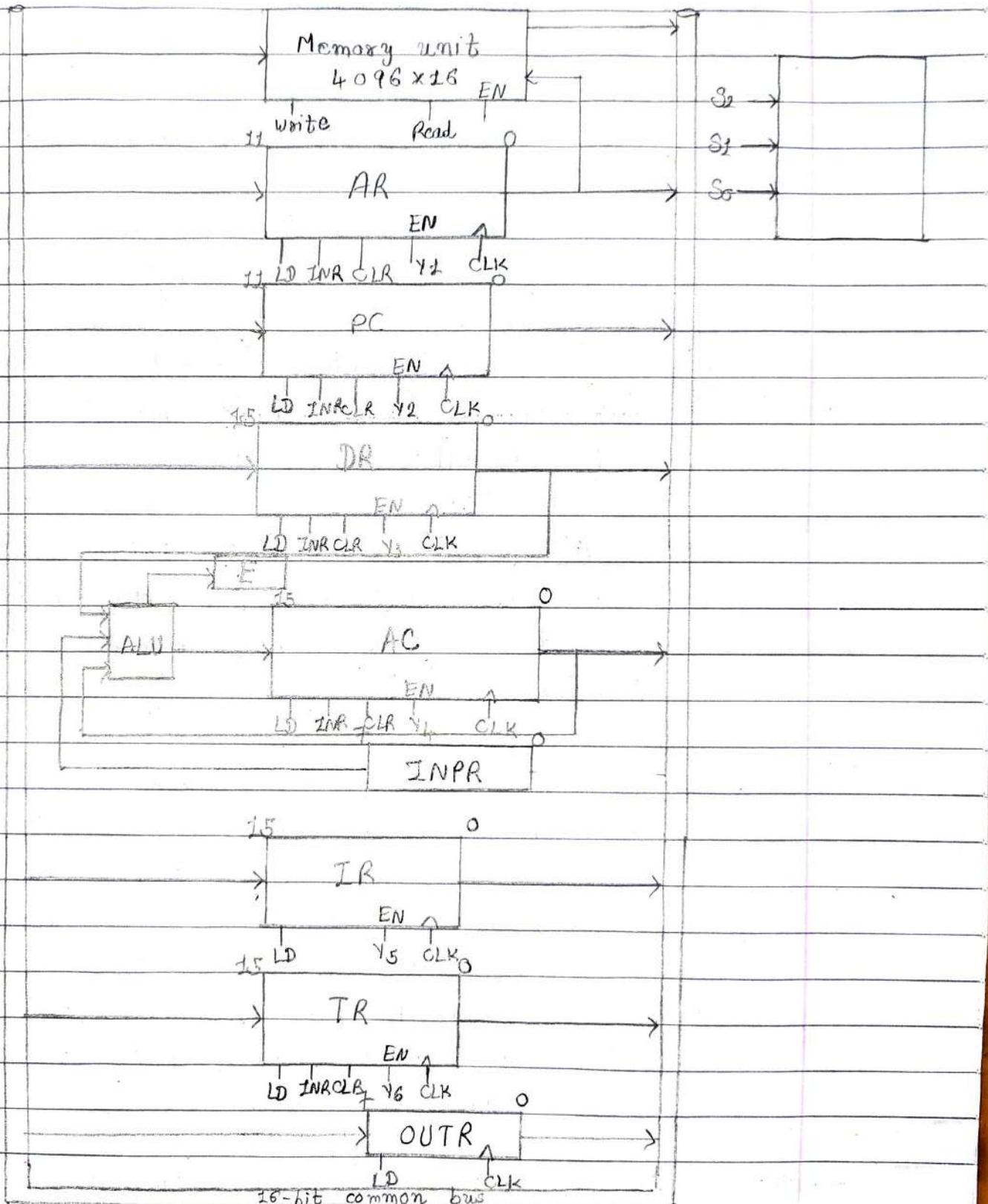
(c) Effective Address

→ The effective address is a term that describes the address of an operand that is stored in memory.

6. Draw and Explain the common bus system for a basic computer registers.

Ans: The computer uses a common bus system for transferring information from register to register or between registers and memory instead of one to one connections to avoid excessive connections.

→ The control inputs LD (Load) and EN (Enable) are used to load data into register from



common bus and to send data from register on to the common bus, respectively. The 3:8 decoder is used to select the specific register whose data to be sent on the common bus.

→ The register AR, PC, DR and AC have a CLR (Clear) and INP inputs along with LD and OE inputs. There is register IR is used only to hold instruction read from memory so it does not have INR and CLR or control signals.

→ Registers DR, AC, IR and TR are 16-bit registers. On the other hand, AR and PC are 12-bit registers, thus they hold a 12-bit memory address.

→ INPR and OUTR are the 8-bit registers and they communicate with the eight least significant bits in the bus. INPR receives a character from an input device which is then transferred to AC register, thus it serves as an information provider. OUTR receives a character from AC and delivers it to an output device.

→ AR register provides address to the memory and

data input and data output of memory is connected to the common bus. The ALU has three 16-bit input sources and 16-bit destination, i.e. AC register. The three 16-bit input source are AC, DR and INPR registers. The ALU performs arithmetic and logic micro-operations and result is transferred to the AC register. Thus AC gets input from ALU.

→ In this configuration, the contents of any register can be applied onto the bus and an operation can be performed in the ALU during the same clock cycle.

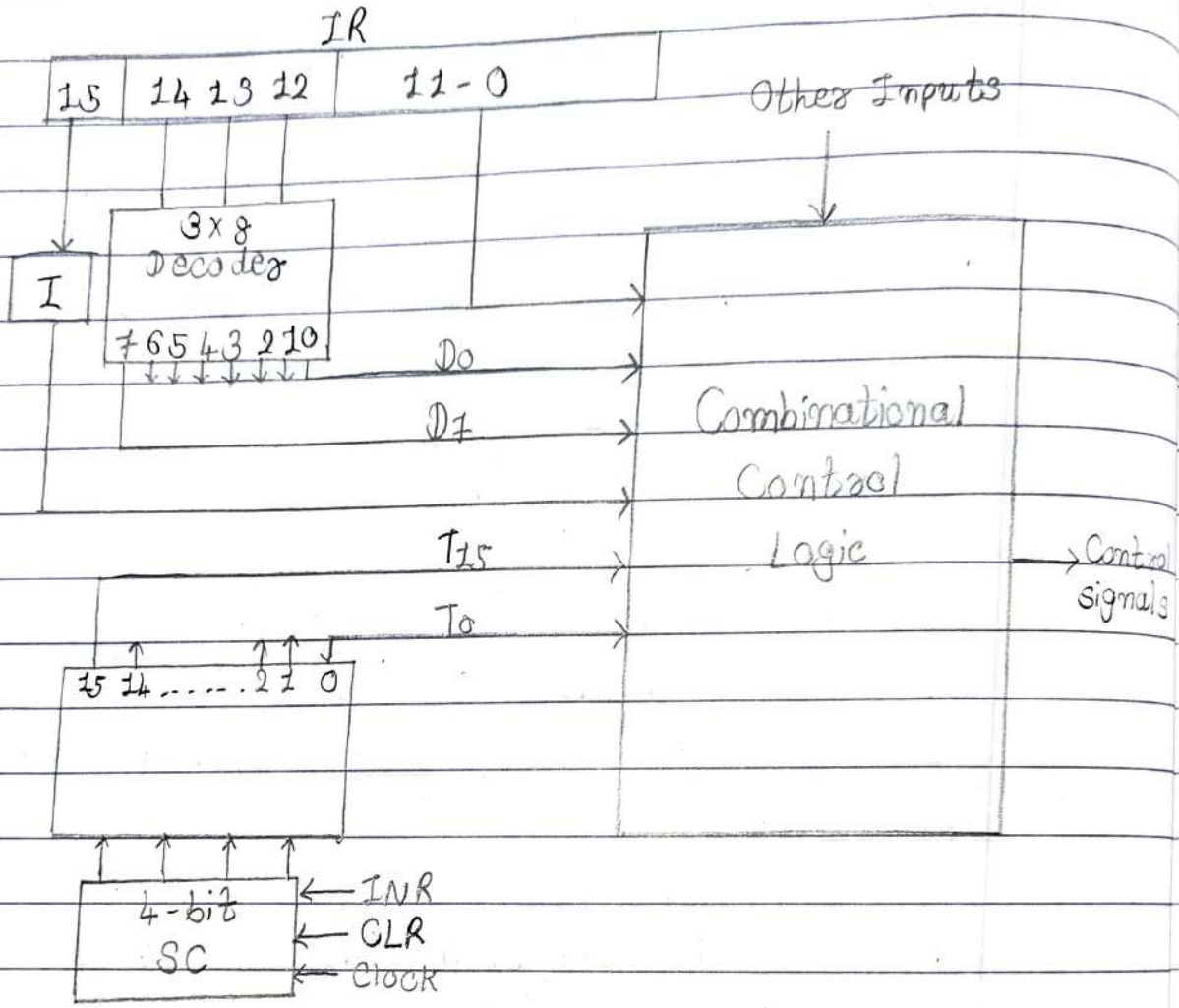
7. Draw and explain the control unit of basic computer.

Ans: Control unit (CU) of a processor translates from machine instructions to the control signal for the micro-operations that implement them.

→ Control units are implemented in one of two ways:

(1) Hard wired Control

CU is made up of sequential and combinational circuits to generate the control signals.



(2) Microprogrammed Control

A control memory on the processor contains ~~microoperation~~ microprograms that activate the necessary control signals.

→ We will consider a hardwired implementation of the control unit for the basic computer.

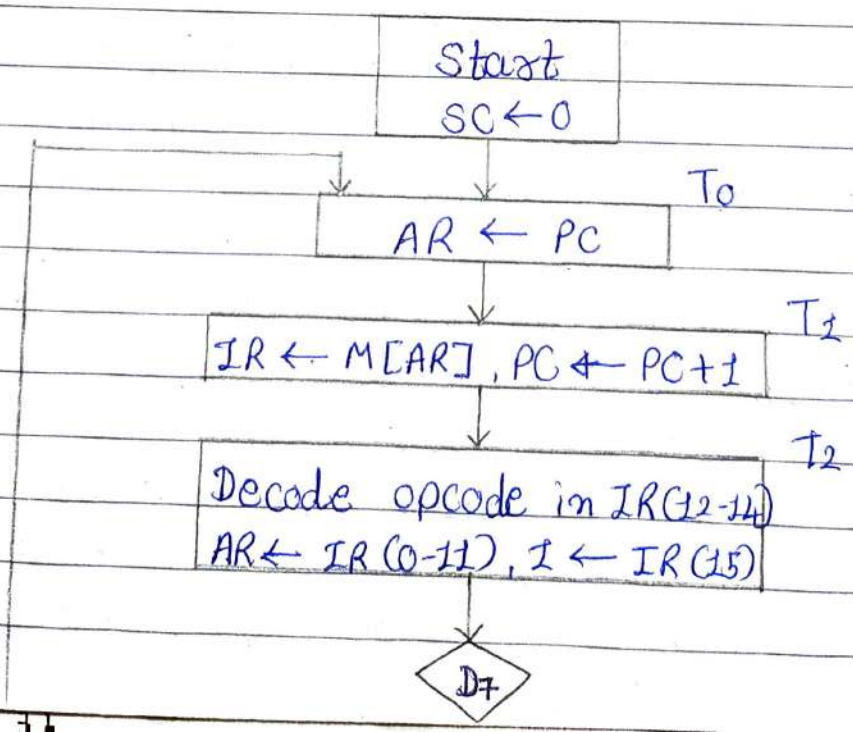
8. Explain Instruction cycle with the help of flow chart.

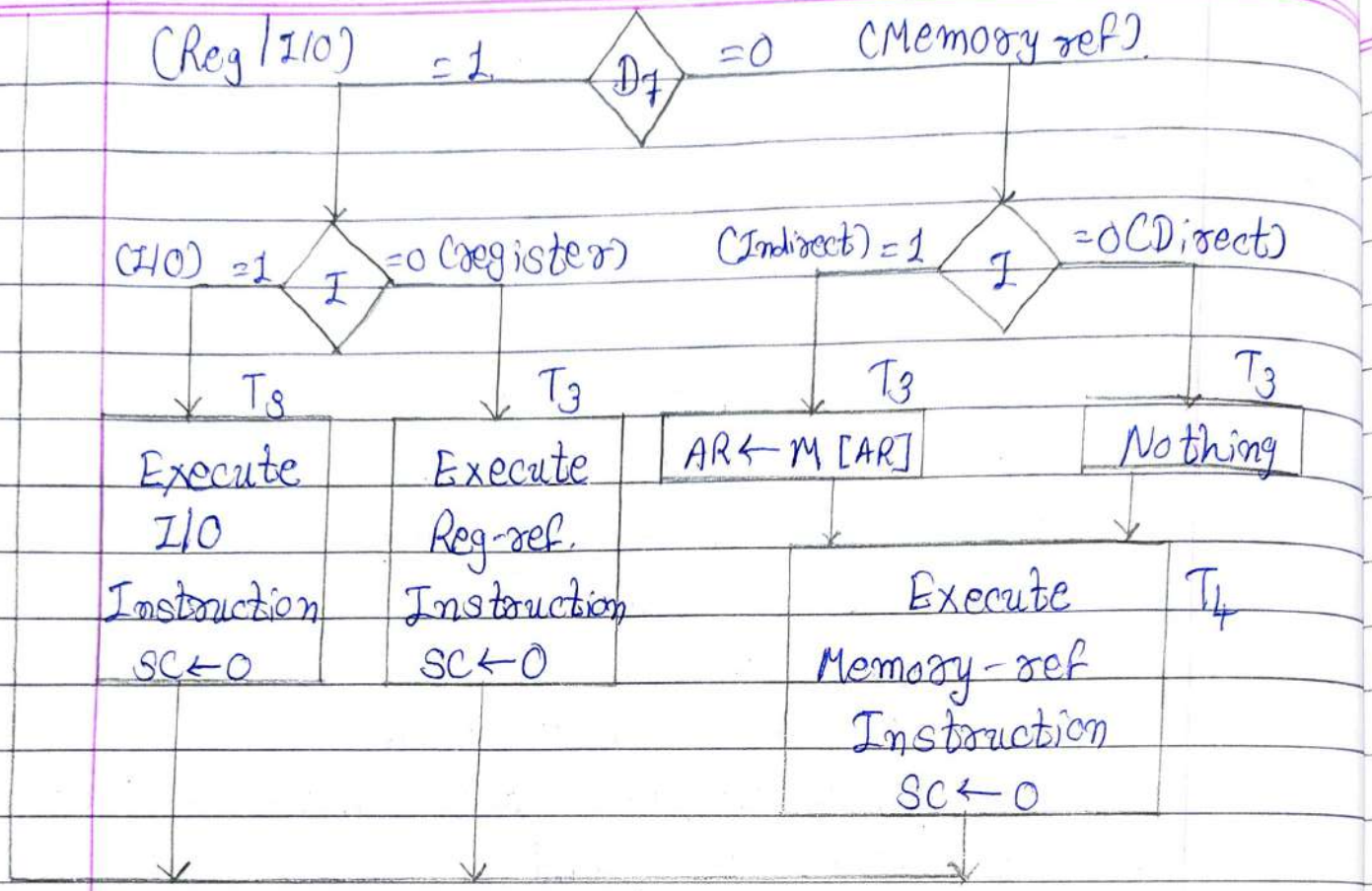
Ans: A program residing in the memory unit of the computer consists of a sequence of instructions. In the basic computer each instruction cycle consists of following phases.

(a) Fetch an instruction from memory
(b) Decode the Instruction.
(c) Read the effective address from memory if the instruction has an indirect address.
(d) Execute the instruction.

→ Control signals

→ After step 4, control goes back to step 1 to fetch, decode and execute the next instruction. This process continues unless a HALT instruction is encountered.





→ If $D_7 = 1$, the instruction must be register-reference or input-output type. If $D_7 = 0$, the operation code must be one of the other seven values 110, specifying a memory-reference instruction.

→ If $D_7 = 0$ and $I = 1$, we have a memory-reference instruction with an indirect address. It is then necessary to read the effective address from memory.

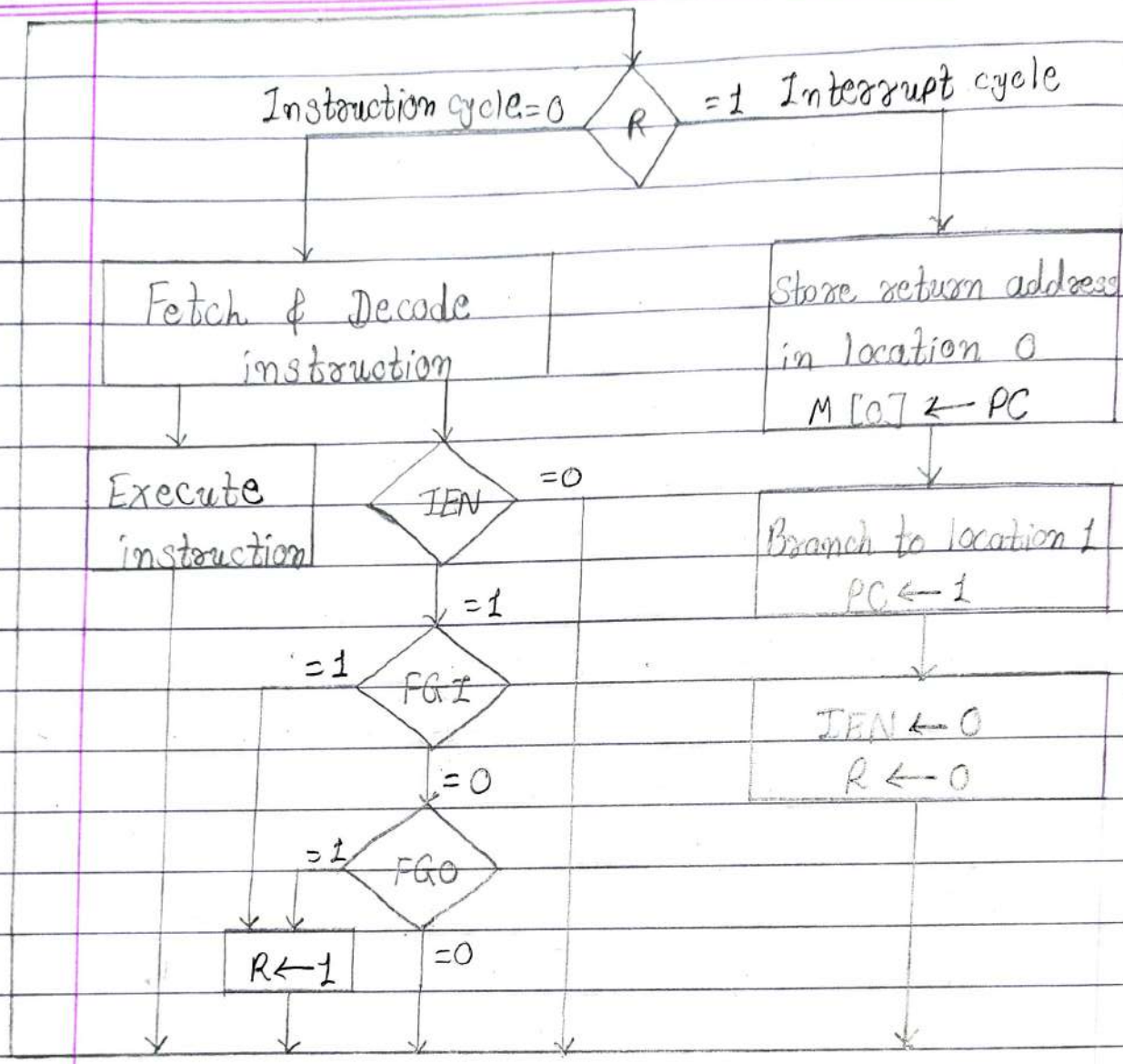
→ The selected operation is activated with the clock transition associated with timing signal T_3. This can be symbolized as :

$D'_7 I T_3$: $AR \leftarrow M[AR]$
$D'_7 I' T_3$: Nothing
$D_7 I' T_3$: Execute register reference inst.
$D_7 I T_3$: Execute input output instruction

→ When a memory-reference instruction with $I=0$ is encountered it is not necessary to do anything since the effective address is already in AR.
→ However, the sequence counter SC must be incremented when $D'_7 I T_3 = 1$. So that the execution of the memory reference instruction can be continued with timing variable T_4.
→ A register reference or input-output instruction can be executed with the click associated with timing signal T_3. After the instruction is executed, SC is cleared to 0 and control I returns to fetch phase with $T_0 = 1$. SC is either incremented or cleared to 0 with every positive clock transition.

9. Explain interrupt cycle with the flowchart.

Ans: The interrupt cycle is a hardware implementation of a branch and save return address operation.
→ An interrupt flip-flop R is included in the computer.



→ When R=0, the computer goes through an instruction cycle. During the execute phase of the instruction cycle IEN is checked by the control.

→ If it is 0, it indicates that the programmer doesn't want to use the interrupt, so control continues with the next instruction cycle. If IEN is 1, control checks the flag bits.

If both flags are 0, it indicates that neither the input nor the output register are ready for transfer of information.

→ In this case, control continues with the next instruction cycle. If either flag is set to 1 while IEN = 1, flip-flop R is set to 1. At the end of the execute phase, control checks the value of R, and if it is equal to 1, it goes to an interrupt cycle instead of an instruction cycle.

→ The flip-flop is set to 1 if IEN = 1 and either FGI or FGO are equal to 1. This can happen with any clock transition except when timing signals T_0, T_1 or T_2 are active.

- The condition for setting flip-flop R = 1 can be expressed with the following register transfer statement:

$$T_0' T_1' T_2' (IEN)(FGI + FGO) : R \leftarrow 1$$

- The symbol + between FGI and FGO in the control function designates a logic OR operation. This is AND with IEN and $T_0' T_1' T_2'$. The fetch and decode phases of the instruction cycle must be modified and replace

T_0, T_1, T_2 with $R'T_0$, $R'T_1$, $R'T_2$.
Therefore the interrupt cycle statements are:

RT_0 : $AR \leftarrow 0$, $TR \leftarrow PC$
RT_1 : $M[AR] \leftarrow TR$, $PC \leftarrow 0$
RT_2 : $PC \leftarrow PC+1$, $IEN \leftarrow 0$, $R \leftarrow 0$, $SC \leftarrow 0$

0			0		256
1	0 BUN 1120		PC=1	0 BUN 1120	
255			255		
PC=256	Main Program		256	Main Program	
1120			1120		
	I/O program			I/O program	
	1 BUN 0			1 BUN 0	
	Before Interrupt			After Interrupt	

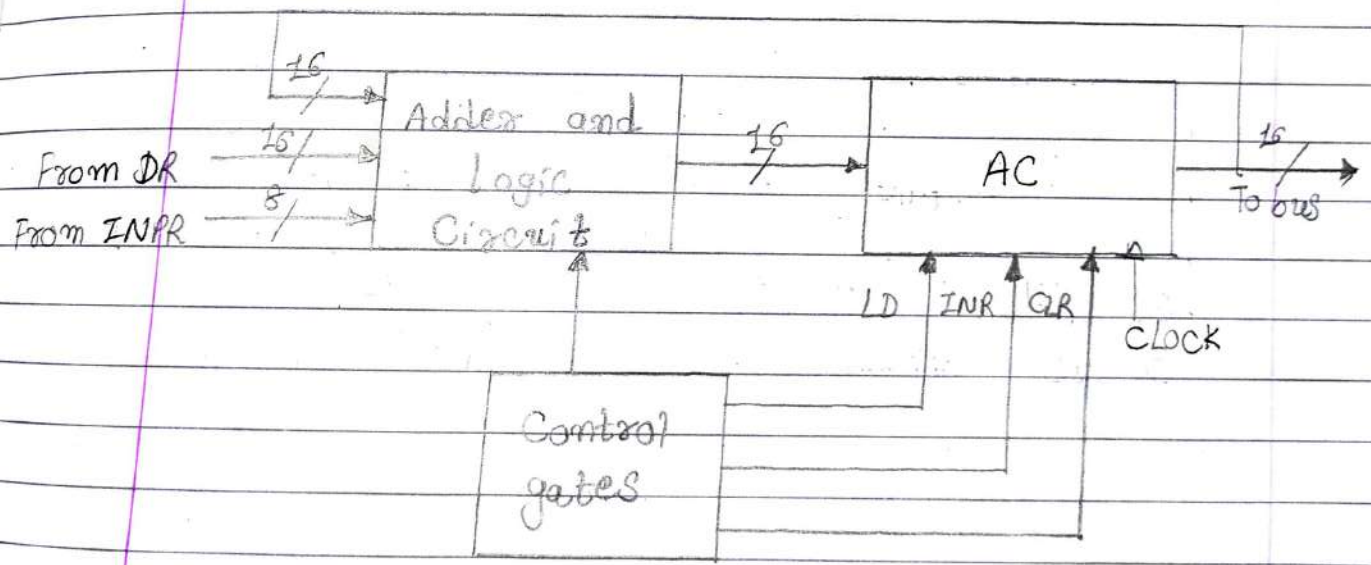
→ During the first timing signal AR is cleared to 0, and the content of PC is transferred to the temporary register TR with the second timing signal, the return address is stored in memory at location 0 and PC is cleared to 0.

→ The third timing signal increments PC to 1, clears IEN and R, and control goes back to T_0

by clearing SC to 0. The beginning of the next instruction cycle has the condition R To and the content of PC is equal to 1. The control then goes through an instruction cycle that fetches and executes the BUN instruction in location 1.

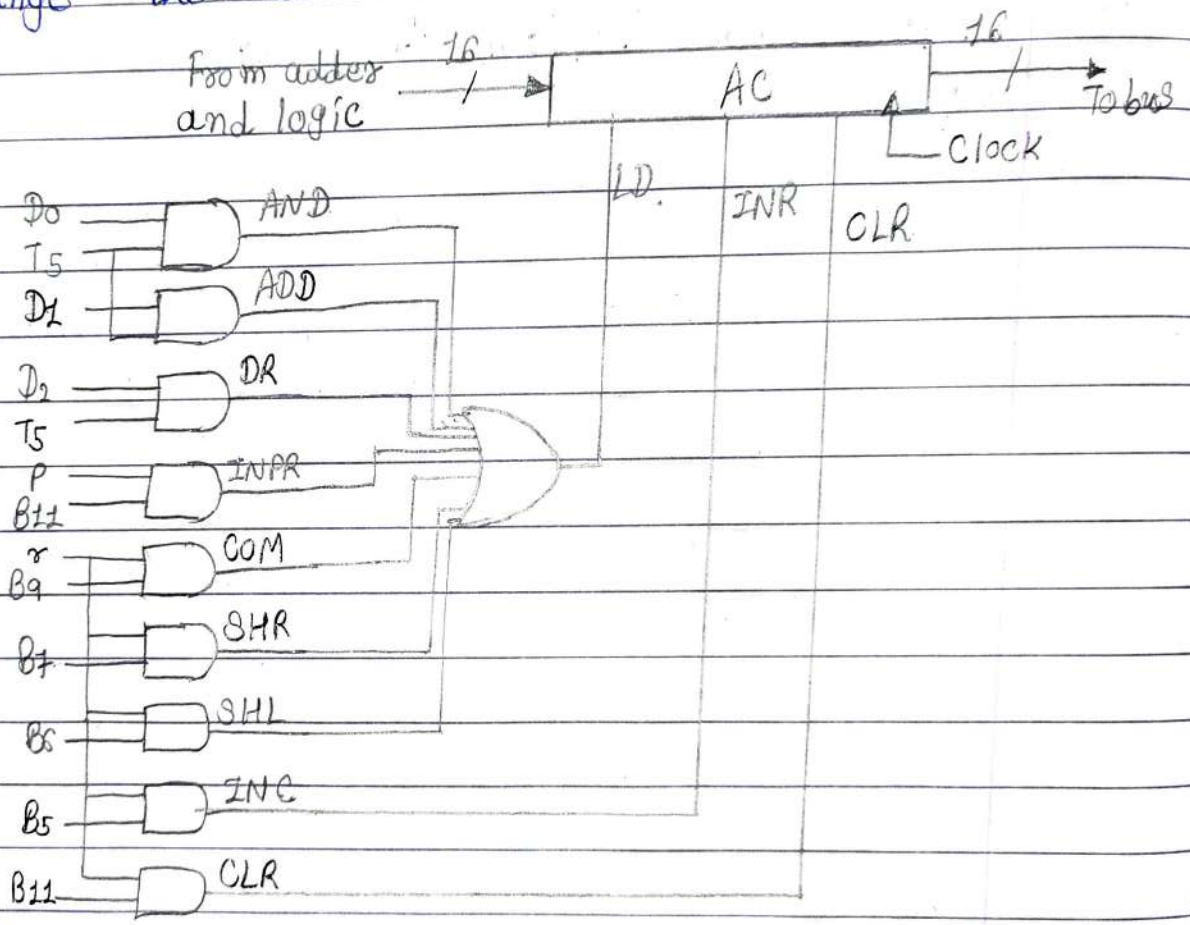
10. Draw and Explain the design of Accumulator Logic.

Ans:



→ The adder and logic circuit has three sets of inputs. One set of 16 inputs comes from the output AC. Another set of 16 input comes from the data register DR. A third set of eight inputs comes from the input register INPR. The output of the adder and logic circuit provide the data inputs for register.

→ In addition, it is necessary to include logic gates for controlling the LD, INP and CLR in register and for controlling the operation of the adder and logic circuit. In order to design the logic associated with AC, it is necessary to extract all the statements that change the content of AC.



$D_0 T_5$:	$AC \leftarrow AC \wedge DR$	AND with DR
$D_1 T_5$:	$AC \leftarrow AC + DR$	Add with DR
$D_2 T_5$:	$AC \leftarrow DR$	Transfer from DR
$p B_{11}$:	$AC(0\text{-}7) \leftarrow INPR$	Transfer from INPR
$r B_9$:	$AC \leftarrow AC'$	Complement
$r B_7$:	$AC \leftarrow shr\ AC, AC(15) \leftarrow E$	Shift right

rB_6 : $AC \leftarrow shl\ AC,\ AC(0) \leftarrow E$ Shift left
rB_{11} : $AC \leftarrow 0$ clear
rB_5 : $AC \leftarrow AC + 1$ Increment

→ The control function for the clear microoperation is rB_{11}, where $r = D_7 I' T_3$ and $B_{11} = IR(11)$. The output of the AND gate that generates this control function is connected to the CLR input of the register.

→ Similarly, the output of the gate that implements the increment microoperation is connected to the INR input of register.

→ The other seven microoperations are generated in the adder and logic circuit and are loaded into AC at the proper time.

→ The output of the gates for each control function are marked with a symbolic name and used in the design of the adder and logic circuit.

ASSIGNMENT-3

જય સ્વામિનારાયણ

Dt.: / / Pg.no.:

1. Write an assembly level program for the following pseudocode.

```
SUM = 0
SUM = SUM + A + B
DIF = DIF - C
SUM = SUM + DIF
```

Ans:

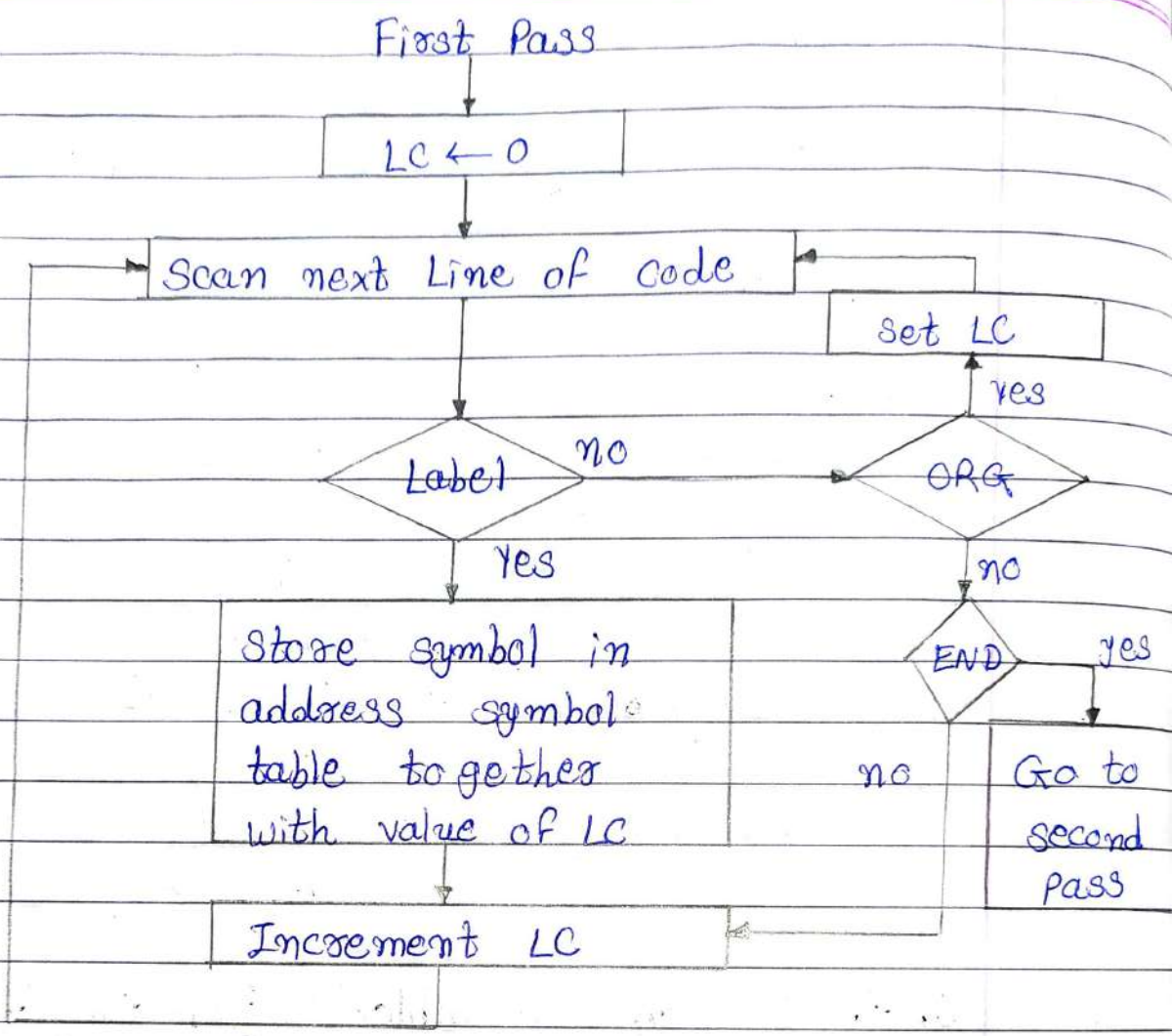
```
CLA                 / SUM = 0
STA  SUM
LDA  SUM            / Load current sum
ADD  A              / Add A to sum
ADD  B              / Add B to sum
STA  SUM            / Save sum
LDA  C              / Load C to AC
CMA                 / Create 2's complement
INC
ADD  DIF            / Subtract C from DIF
STA  DIF            / Save DIF
ADD  SUM            / Add SUM to DIF
STA  SUM            / Save SUM
HLT                 / Halt
```

2. Write a program loop using a pointer and a counter to clear the contents of hex corations 500 to 5FF with 0.

Ans:
```
LDA   NBR     ; Initialize counter
CMA           ; 2's complement of NBR INC
STA   CTR     ; save- NBR to counter
LDA   ADR     ; Save start address
STA   PTR     ; Initialize pointer PTR
LOP , CLA     ; Clear AC
STA   PTRI    ; Reset memory word
ISZ   PTR     ; Increment pointer
ISZ   CTR     ; Increment counter
BUN   LOP     ; Branch to LOP (CTR <0)
HLT           ; Halt when CTR=0
NBR , HEX FF  ; NBR of cleared words
CTR , -       ; Counter
ADR , HEX 500 ; Start Address
PTR , -       ; Pointer
```

3. What is an assembler? Explain the first pass of an assembler with a flowchart.

Ans: An assembler is a program that accepts a symbolic language program and produces its binary machine language equivalent.
- The input symbolic program is called the source program and the resulting binary program is called the ~~logic~~ object program.
- The assembler is a program that operates on character strings and produces an equivalent binary interpretation.



→ First Pass of an Assembler :

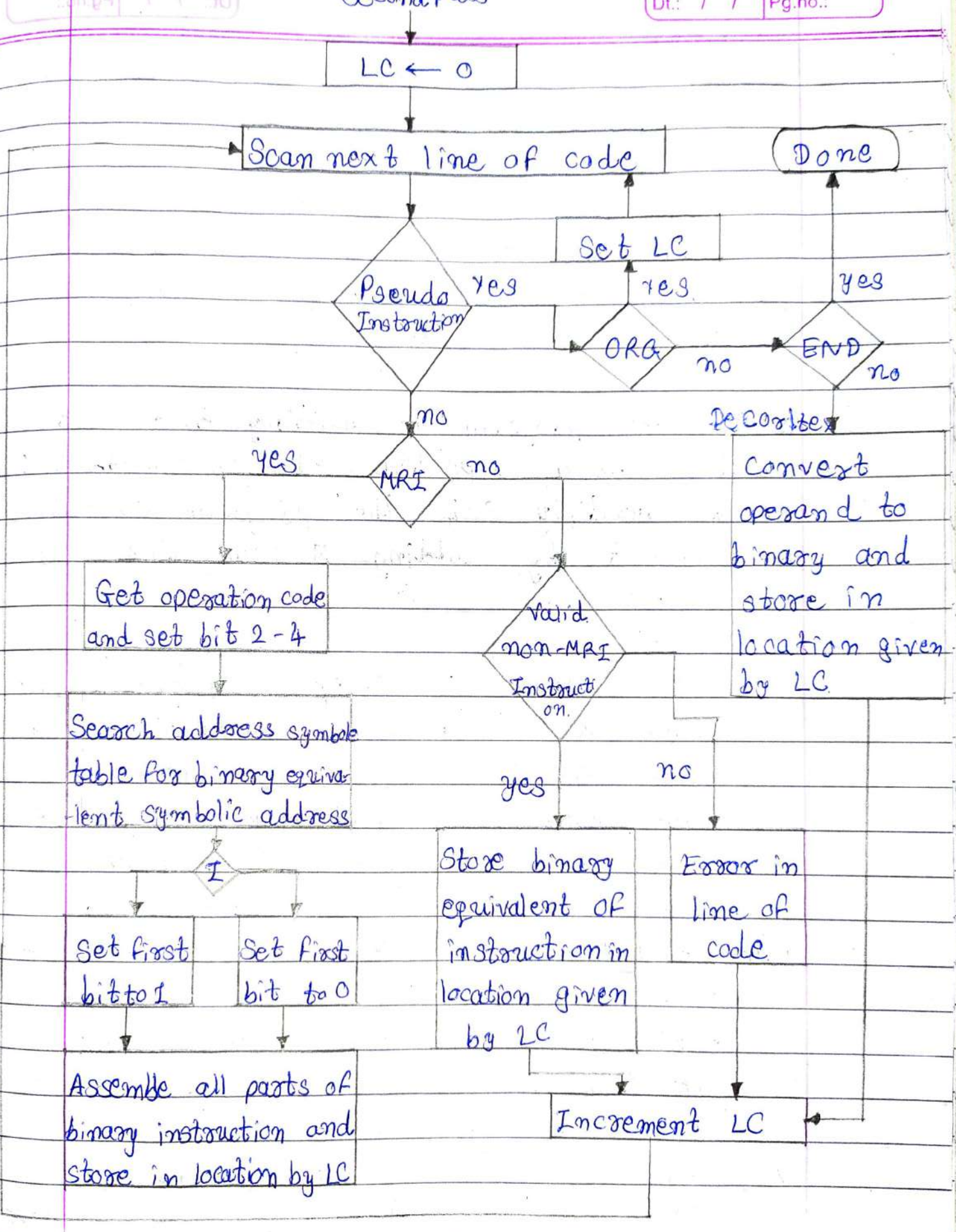
- During the first pass, it generates a table that correlates all user - defined address symbols with their binary equivalent value.
- The binary translation is done during the second pass.
- To keep track of the location of instructions, the assembler uses a memory word called a location counter.

- The content of LC stores the values of the memory location assigned to the instruction or operand presently being processed.
- The ORG pseudo instruction initializes the location counter to the value of first location.
- Since instruction are stored in sequential locations, the content of LC is incremented by 1 after processing each line of code.
- To avoid ambiguity in case ORG is missing, the assembler sets the location counter to 0 initially.
- The task performed by the assembler during the first pass are described in the flowchart
- LC is initially set to 0.
- A line of symbolic code is analyzed to determine if it has a label.
- If the line of code has no label, the assembler checks the symbol in the instruction field.
- If it contains an ORG pseudo instruction, the assembler sets LC to number that follows ORG and goes back to process the next line.
- If the line has an END pseudo instruction, the assembler terminates the first pass and goes to the second pass.
- If the line of code contains a label, it is stored in the address symbole table

together with its binary equivalent number specified by the content of LC. Nothing is stored in the table if no label is encountered.
- LC is then incremented by 1 and a new line of code is processed.

4. Explain the working of second pass Assembler with its flowchart.

Ans: Machine instructions are translated during the second pass by means of table-look up procedure.
→ During the second pass, the assembler examines the operands for symbolic references to storage locations and resolves these symbolic references using information in the symbole table
→ On the second pass, the assembler:
- Examines the operands for symbolic reference to storage locations and resolves these symbolic reference using information in the symbol table.
- Ensure that no instructions contain an invalid instruction form
- Translates source statements into machine code and constants, thus filling the allocated space with object code.


Second Pass
LC ← 0
Scan next line of code
Pseudo Instruction
yes
Set LC
ORG
yes
no
END
yes
no
Done
no
MRI
yes
no
Decode
Convert operand to binary and store in location given by LC
Get operation code and set bit 2-4
Valid non-MRI Instruction.
yes
no
Search address symbol table for binary equivalent symbolic address
I
Set first bit to 1
Set first bit to 0
Store binary equivalent of instruction in location given by LC
Error in line of code
Assemble all parts of binary instruction and store in location by LC
Increment LC

- Produces a file containing error message, if any have occured.
- At the beginning of the second pass, the assembler scans each source statements a second time. As the assembler translates each instruction, it increments the value contained in the location counter.

5. Write the program to multiply two positive numbers by a repeated addition method. For ex. to multiply 5x4, the program evaluates the product by adding 5 four times, or 5+5+5+5.

Ans:

```
        ORG  100        ; Origin of program is HEX 100
        LDA  X          ; Load first add. of operand
        STA  PTR        ; Store in pointer
        LDA  NBR        ; load minus second operand
        STA  Y          ; Store in counter
        CLA             ; Clear accumulator
LOP,    ADD  PTR I      ; Add an operand to AC
        ISZ  Y          ; increment counter
        BUN  LOP        ; Repeat loop again
        STA  MUL        ; Store sum
        HLT             ; Halt
X,      DEC  5          ; First operand
PTR,    HEX  0          ; This loc. reved for a pointer.
```

```
NBR,  DEC -4        ; second operand
CTR,  HEX 0         ; This loc. reserved for a pointer
MUL,  HEX 0         ; Sum is stored here
      END           ; End of symbolic program
```

6. Write a program that evaluates the logic ex-or of two logic operands.

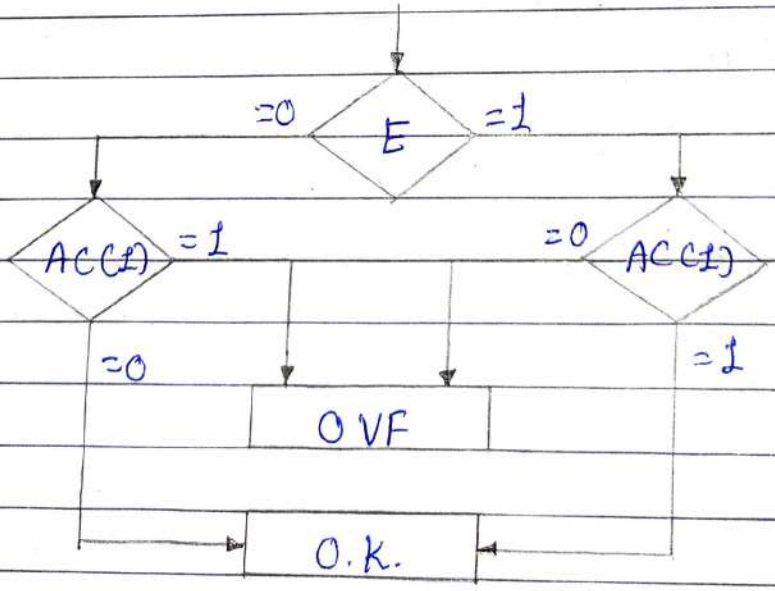
Ans: $Z = x \oplus y = xy' + x'y = [(xy')' . (x'y)']'$

```
    LDA  y
    CMA
    AND  x
    CMA
    STA  TMP
    LDA  x
    CMA
    AND  Y
    CMA
    AND  IMP
    CMA
    STA  Z
    HLT
X,  ....
Y,  ....
Z,  ....
TMP, ....
```

7. Write a program for the arithmetic shift-left operation. Branch to OVF if an overflow occurs.

Ans:

```
        LDA  X
        CLE
        CIL        ; Zero to low order bit
        SZE
        BUN  ONE
        SPA
        BUN  OVF
        BUN  EXT
ONE,    SNA
        BUN  OVF
EXT,    HLT
```



9. Write an assembly language program to multiply two positive numbers.

```
        ORG 100
LOP,    CLE             ; Clear E
        LDA Y           ; Load multiplier
        CIR             ; Transfer multiplier bit to E
        STA Y           ; Store shifted multiplier
        SZE             ; check if bit is zero
        BUN ONE         ; Bit is one, go to ONE
        BUN ZRO         ; Bit is zero, go to ZERO
ONE,    LDA X           ; Load Multiplicand
        ADD P           ; Add to partical product
        STA P           ; Store partial product
        CLE             ; Clear E
ZRO,    LDA X           ; load multiplicand
        CIL             ; Shift left
        STA X           ; Store shifted multiplicand
        ISZ CTR         ; Increment counter
        BUN LOP         ; Counter not zero, repeat loop
        HLT             ; Counter is zero halt
CTR,    DEC -8          ; This location serves as a counter
X,      HEX 000F        ; Multiplicand stored here
Y,      HEX 000B        ; Multiplier stored here
P,      HEX 0           ; Product formed here
        END             ; END of symbolic program.
```

10. Write an assembly language program to Add two double precision numbers.

Ans:

```
LDA AL      ; Load A low
ADD BL      ; Add B low, Carry in E
STA CL      ; Store in C Low
CLA         ; clear AC
CIL         ; Circulate to bring into AC (16)
ADD AH      ; Add A high and carry
ADD BH      ; Add B high
STA CH      ; Store C high
HLT         ; Increment counter 0
AL, ...     ; Locations of operand
AH, ...
BL, ...
BH, ...
CL, ...
CH, ...
    END     ; End of symbolic program.
```

8. For the given program below:

(a) Explain in words what the program accomplishes when it is executed. What is the value of location CTR when the computer halts?

(b) List the address symbol table obtained during the first pass of the assembler.

```
    ORG 100
    CLE
    CLA
    STA CTR
```

```
      LDA  WRD
      SZA
      BUN  ROT
      BUN  STP
ROT,  CIL
      SZE
      BUN  AGN
      BUN  ROT
AGN,  CLE
      ISZ  CTR
      SZA
      BUN  ROT
STP ,  HLT
CTR,  HEX 0
WRD,  HEX
      END
```