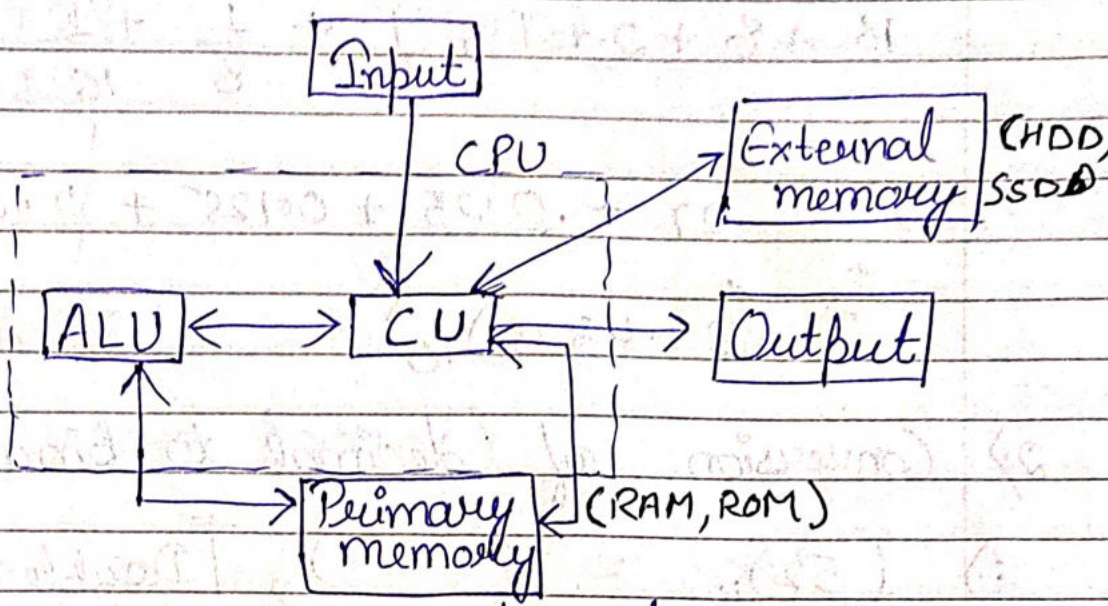
Date _/_/_

COMPUTER ORGANISATION



ALU → Arithmetic logical Unit
CU → Control Unit.

NUMBER SYSTEM

Number Conversion

1.) Conversion of binary to decimal no. System

i) $(10101)_2 = (21)_{10}$

$2^4 \times 1 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 16 + 4 + 1 = \underline{21}$

Date _/_/_

(ii) $(11011.0111)_2 \rightarrow (\quad)_{10}$

$1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$

$= 16 + 8 + 2 + 1 + \frac{1}{4} + \frac{1}{8} + \frac{1}{16}$

$\rightarrow 27 + 0.25 + 0.0125 + 0.0625$

$\Rightarrow (27.4375)_{10}$

2) Conversion of decimal to binary

i) $(52)_{10} = (\quad)_2$ [Double-dabble method]

2	52	0
2	26	0
2	13	1
2	6	0
2	3	1
	1	

$(110100)_2$

ii) $(0.75)_{10} = (0.11)_2$

~~0.75 x 2 =~~

$0.75 * 2 = 1.50$ ↓

$0.50 \times 2 = 1.00$

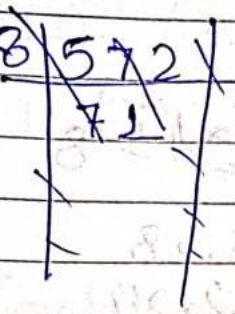
0.75
2
1.50

* Bit length of Octal no. system = $\frac{1}{3}$ x bit length of binary system

Date _/_/_

3.) Decimal Conversion of Octal to binary

i) $(572)_8$ to binary



572
$(101111010)_2$

572

ii.) $(678.05)_8 \rightarrow (\quad)_2$

8421

$(110111100.000101)_2$

4.) Conversion of binary to Octal.

i) $(011\,000\,101 . 010\,110)_2 \rightarrow (\quad)_8$

$(305.26)_8$

5.) Conversion of Octal to decimal.

ii) $(456.77)_8 \rightarrow (\quad)_{10}$

$4\times8^2 + 5\times8^1 + 6\times8^0 + 7\times8^{-1} + 7\times8^{-2}$

$\rightarrow 256 + 40 + 6 + 0.875 + 0.1093$

$\rightarrow (302.984)_{10}$

Date _/_/_

6.) Conversion of decimal to Octal

ii.) $(987.66)_{10}$ to $(\quad)_8$

8	987	3
8	123	3
8	15	7
	1	

$0.66 \times 8 = 5.28$
$0.28 \times 8 = 2.24$
$0.24 \times 8 = 1.92$
$0.92 \times 8 = 7.36$

$(1733.5217)_8$

7.) Conversion of Hexadecimal to binary

i) $(ABCD.76A)_{16} \rightarrow (\quad)_2$

$(1010.1011.1100.1101 . 0111.0110.1010)_2$

8.) Conversion of binary to Hexadecimal

i) $(0001.1011.0001.0011 . 0110.1110)_2 \rightarrow (\quad)_{16}$

$(1B1.6E)_{16}$

9.) Conversion of decimal to Hexadecimal

i.) $(5072)_{10} \rightarrow (\quad)_{16}$

16	5072	0
16	317	13
16	19	3
	1	

$(13D0)_{16}$

Date _/_/_

10.) Conversion of ~~decimal~~ Octal to Hexadecimal

i) $(456)_8 \rightarrow (12E)_{16}$

000 | 010 | 110

$(12E)_{16}$

Complements

1) 9's Complement

i) 54378

```
  99999
- 54378
  45621   → 9's Complement
```

2) 10's complement
add 1 to the 9's Complement

i) 54378

```
  45621
+     1
  45622   → 10's complement
```

Date _/_/_

★ Subtraction using 9's complement method

Q. Using 9's complement subtract

72532 - 3250 (M>N)

Ans. 1.

(i) 99999
-03250
96749

(ii) 72532
+96749
(1)69281

end Carry

* If end-carry is generated answer is positive

69281
+ 1
69282 (Ans)

2. 50780 - 1468

(i) 99999
-01468
98531

ii) 50780
+98531
149311

iii) 49311
+ 1
49312 Ans

Note → If end carry = No. is +positive
If not " " = No. is -ve (Don't write)

Case 2

3) 3250 - 72532 (M < N)
(M) (N)

i) Find (9's complement) of N

```
  99999
- 72532
  27467
```

ii) Add M with obtained result

```
  27467
+  3250
  30717
```

NOTE- No. of digits = No. of digit result no. So end carry not generated.

4) 20512 - 60886
(M < N)

i)
```
  99999
- 60886
  39113
```

ii)
```
  99999
- 59625
  40374
```

iii)
```
  39113
+ 20512
  59625
```

• End Carry discarded in 10's complmt

Date _/_/_

★ Subtraction using 10's Complement method.

Q1 72532 − 3250 (M > N)

i) Find 10's complement of N

```
    1
  96749   (9's complement)
+     1
-------
  96750   10's "
```

ii)

```
  (1)
  96750
+ 72532
-------
 (1)69282
  ↳ end Carry
```

iii) Discarding the end Carry

ans := 69282

Q2. 20512 − 60886

i)

```
  39113  ← 9's complement
+     1
-------
  39114
```

iii)

ii)

```
  39114
+ 20512
-------
  59626
```

Date _/_/_

Q3. 3250 - 72532 (M<N)

i) 10's complement of N

```
  99999
- 72532
  27467
+     1
  27468
```

ii)

```
  3250
+27468
 30718
```

iii) No end carry so find 10's complement

```
  99999
-30718
  69281
+     1
  69282
```

★ 1's Complement

```
  111111
  101001
  010110   → Ans. (1's Complement)
```

given →

★ 2's complement

```
  010110   → 1's complement
+     1
  010111   → 2's complement
```

Date _/_/_

Subtraction Using 1's complement Method

Q.1 Using 1's complement method, subtract 84-67

2	84	0
2	42	0
2	21	1
2	10	0
2	5	1
2	2	0
	1	

84 → 1010100

67 → 1000011

2	67	1
2	33	1
2	16	0
2	8	0
2	4	0
2	2	0
	1	

Step 1:- Finding 1's complement of 'N'

```
  1111111
- 1000011
  0111100
```

Step 2:- Addition of M with obtained result.

```
  1010100
+ 0111100
  1 0010000
```

Carry

Step 3:-

```
  0010000
+         1
  0010001 → ans
```

* add carry in i's complement sub.

(M>N) 9=1, 10=2

(end carry added) (end carry removed)

Q2. 90 − 55 (M>N)

2	90	0
2	45	1
2	22	0
2	11	1
2	5	1
2	2	0
	1	

2	55	1
2	27	1
2	13	1
2	6	0
2	3	1
.	1	

1's complement of N

1 1 1 1 1 1 1 1

− 0 1 1 0 1 1 1 1

1 0 0 1 0 0 0 0

Add →

1 0 1 1 0 1 0

+ 1 0 0 1 0 0 0

(1) 0 1 0 0 0 1 0

Again find 1's complement

$2^0+2^2+2^3+2^4$

1 1 0 0 0 1 0

1's 0 0 1 1 1 0 1

0 1 0 0 0 1 0

+ 1

0 1 0 0 0 1 1 → ans

M<N | 9's ↓ no carry | 10's ↓ No carry | 1's ↓ again find 1's compl | 2's ↓ Again, find 2's.

Q3.) 67 − 84 (M<N)

2	67	1	67 → 1000011
2	33		84 → 1010100
	1	→	

1's complement

```
  1 1 1 1 1 1 1
- 1 0 1 0 1 0 0
-----------------
  0 1 0 1 0 1 1
```

Add →

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

1's complement

```
  1 1 0 1 1 1 0
    ↓
-----------------
1's → 0 0 1 0 0 0 1
```

Date _/_/_

★ Using 2's complement method subtract

1) 84 - 67 (M>N)

84 → 1010100
67 → 1000011

i) 2's complement of N -

1's 0111100
+ 1
2's 0111101

ii) Add →

1
0111101
+1010100
(1)0010001

iii) Discard the 1. → 0010001

2) 63 - 89

2	63	1
2	31	1
2	15	1
2	7	1
2	3	1
	1	

2	89	1
2	44	0
2	22	0
2	11	1
2	5	1
2	2	0
	1	

Date - / - / -

63 → 1 1 1 1 1 1
89 → 1 0 1 1 0 0 1

2's complement of 89

```
      1 0 1 1 0 0 1
1's → 0 1 0 0 1 1 0
2's → +           1
      0 1 0 0 1 1 1
```

Add →

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

Again find 2's complement

```
1's →   0 0 1 1 0 0 1
2's → +             1
      0 0 1 1 0 1 0  → ans
```

Signed number System and Unsigned number system:

* 1 at the MSB side represent a -ve no.
* 0 at the " " " " +ve no.

Range → -2^{n-1} to (+) $2^{n-1}-1$

Date _/_/_

Addition by using signed 2's complement

1.) $+6$ $+13$ $+19$

$$\begin{array}{r} 00000110 \\ +\ 00001101 \\ \hline 00010011 \end{array}$$

2.) $-6 + 13 = +7$

$$\begin{array}{r} 11111001 \\ +\ \quad\quad 1 \\ \hline 11111010 \end{array} \rightarrow \text{2's complement of } +6$$

$$\begin{array}{r} 11111010 \\ +\ 00001101 \\ \hline (1)\ 00000111 \end{array}$$

MSB discard

$\Rightarrow$ +ve $\quad +7$

3) $+6 + (-13) = -7$

$$\begin{array}{r} 11111001 \\ +\ \quad\quad 1 \\ \hline 11111010 \end{array} \rightarrow \text{2's of } +13$$

$$\begin{array}{r} 11111010 \\ +\ 00000110 \\ \hline 11111001 \end{array} \quad +6$$

$$\begin{array}{r} 10000110 \\ +\ \quad\quad 1 \\ \hline (1)\ 00000111 \end{array} = -7$$

Signed bit

Date _/_/_

4.3) $-6 + (-13)$

8 4 2 1
0 1 1 0
1 1 0 1

```
  0 0 0 0 0 1 1 0
1's→ 1 1 1 1 1 0 0 1
2's→ +             1
  1 1 1 1 1 1 0 1 0  → 2's of 6

  0 0 0 0 1 1 0 1
1's  1 1 1 1 0 0 1 0
   +               1
2's  1 1 1 1 0 0 1 1  → 2's of 13

    1 1 1       1
  1 1 1 1 1 0 1 0
+ 1 1 1 1 0 0 1 1
① 1 1 1 0 1 1 0 1
  ↓
  1 0 0 1 0 0 1 0
+               1
2's 1 0 0 1 0 0 1 1  → -19
```

Date _/_/_

Vonn Neumann Architecture	Harvard Architecture
1) Instruction and data are stored in the same memory.	1.) Instruction and data are stored in different memories.
2) Same bus for both instruction and data	2.) Different bus.

Signed Number Subtraction

1.) (+6) − (+13)

8421
0110

00000110 → +6

+13 → 00001101
1's → 11110010
+ 1
2's 11110011 → 13

+ 00000110
11111001
↓
10000110
+ 1
10000111 → −7

Date _/_/_

2) (+20) - (+27)

16 8 4 2 1
1 0 1 0 0
1 1 0 1 1

00010100 → 20

00011011
1's 11100100
+ 1
2's 11100101 → 27

00010100
+11100101
11111001
↓
10000110
+ 1
10000111 → -7

3) (-6) - (+15)

00000110
1's 11111001
+ 1
2's 11111010 → 6

00001111
1's 11110000
+ 1
11110001 → 15

11111010
+11110001
(1)11101011
discard ↓

00010100 → -21
+ 1
2's → 00010101

PASSION

Date _ / _ / _

4.) (+6) - (-15)

```
  0 0 0 0 0 1 1 0
+ 0 0 0 0 0 0 0 0
  1 1 1 1 0 1 1 0
  ↓
  1 0 0 0 1 0 0 1
+               1
  1 0 0 0 1 0 1 0  → -
```

8 2

8421
1010

15
+6

```
  0 0 0 0 0 1 1 0
+ 0 0 0 0 0 0 0 0
  1 1 1 1 0 1 1 0
  ↓
  1 0 0 0 1 0 0 1
+               1
  1 0 0 0 1 0 1 0  → 2's
```

```
  0 0 0 0 0 1 1 0
  0 0 0 0 1 1 1 1
  0 0 0 1 0 1 0 1  → +21
```

16 8 4 2 1
1 0 1 0 1
10100

5.) (-20) - (-10)

```
  0 0 0 1 0 1 0 0
1's 1 1 1 0 1 0 1 1
+               1
2's 1 1 1 0 1 1 0 0
+   0 0 0 0 1 0 1 0
    1 1 1 1 0 1 1 0
```

```
    1 0 0 0 1 0 0 1
+               1
  1 0 0 0 1 0 1 0  → -10
```

Overflow

Rule :- If the input carry is different from the o/p carry then there is an overflow

(OR)

$+70 \rightarrow 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0$

$+80 \quad 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0$

$+150 \quad 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0$

Case of Overflow

XOR is used.

Number Representation

- Floating point Representation
- IEEE 754 Floating point Representation

1. Floating Point Representation

Denoted by $\Rightarrow m.r^e$

m = mantissa e = exponent

r = radix

Q. 613.456 → floating point

Mantissa	Exponent
.613456	+3

$\therefore$ No. is $.613456 \times 10^3$

Date _/_/_

2) 1.0.111.001 to floating point

Mantiss	Exp.
.10111001	+5

Floating no. is — $.10111001 \times 2^{5}$

64 32 16 8 4 2 1

2.) IEEE 754 Floating Point Representation

i) Signed bit
ii) Biased Exponent
iii) Normalized Mantissa

} 3 components of IEEE 754

i) Signed Bit

+ve, 0 will be in the MSB side.
-ve, 1 " " " " " "

ii) Biased Exponent

If exponent size is of 8 bit, then the biased no. will be —

$$2^{\text{size}-1} - 1 = 2^{7} - 1 = 127$$

If exp. size is of 11 bit, then biased no —

$$2^{\text{size}-1} - 1 = 2^{10} - 1 = 1023$$

Date _/_/_

3) Normalized Matissa

$1011.11 \quad (1.---)$

$= 1.01111 \times 2^3$

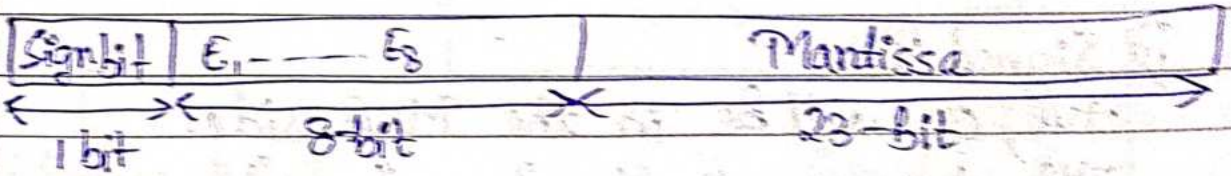
$=$

Two fouts

1) Single bit precision
2) Double bit precision

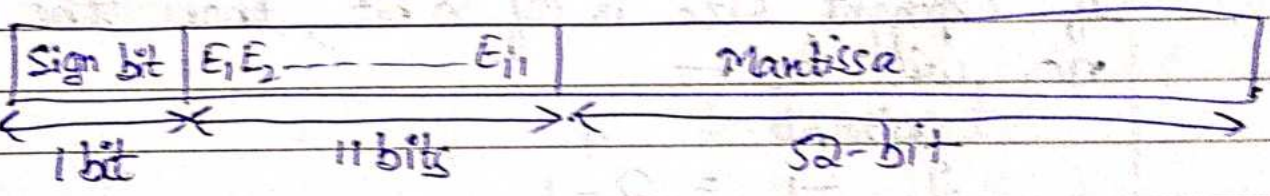
1. Single bit precision

It consists of 32-bit



2. Double bit precision

It consists of 64-bit



Date _/_/_

Circuits

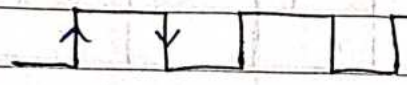
They are of 2 types -

1.) Combinational Circuit

2.) Sequential Circuit

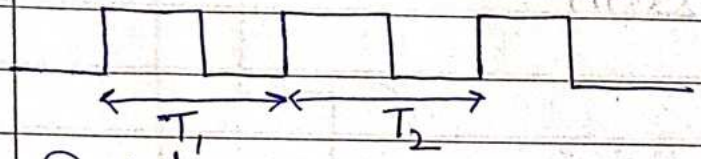
Difference b/w both →

Clock is present only in sequential circuit.



Rising Edge → 0-1
level → always 1
Falling Edge → 1-0

Frequency of a clock



Period = 10 ns

Frequency, $f = \frac{1}{T}$ =

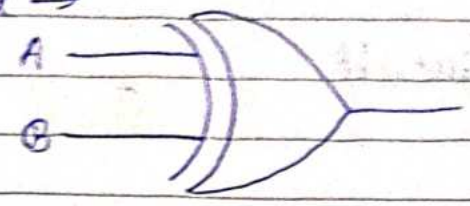
$= \frac{1}{10 \times 10^{-9}} = 100\,MHz$

Date _/_/_

Combinational Circuit

1) XOR Gate

Symbol →



Boolean expression

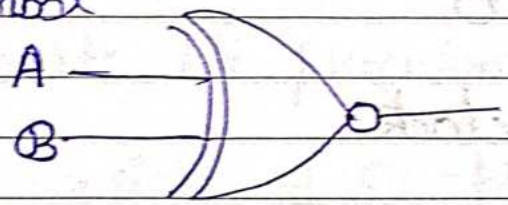
$$Y = A \oplus B$$
$$= A\bar{B} + \bar{A}B$$
$$= 1 \cdot 0 + 0 \cdot 1$$
$$= 0 + 0$$
$$= 0$$

Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

2) XNOR Gate

Symbol



Boolean Expression

$$Y = \overline{A \oplus B}$$
$$= \overline{A\bar{B} + \bar{A}B}$$
$$= \overline{A\bar{B}} \cdot \overline{\bar{A}B}$$
$$= (\bar{A} + B)(A + \bar{B})$$
$$= \bar{A}A + \bar{A}\bar{B} + BA + B\bar{B}$$
$$= 0 + \bar{A}\bar{B} + BA$$

$$\boxed{Y = \bar{A}\bar{B} + AB}$$

Date _/_/_

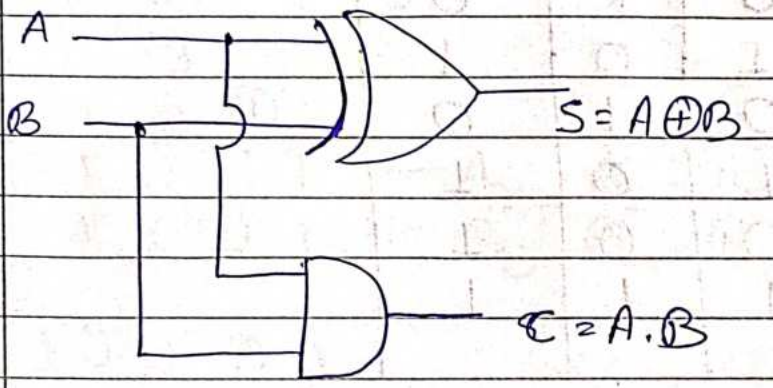
Adder

- **Half adder** → To add 2 bit no.s
- **Full adder** → To add three 1-bit number

★ Half Adder.

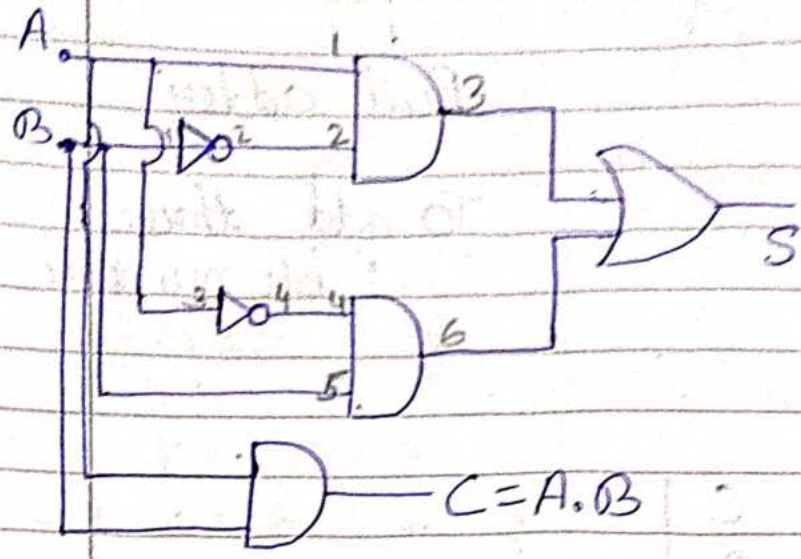
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Circuit diagram



Date _/_/_

Half Adder Design using AND, OR, NOT.



★ Full Adder

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

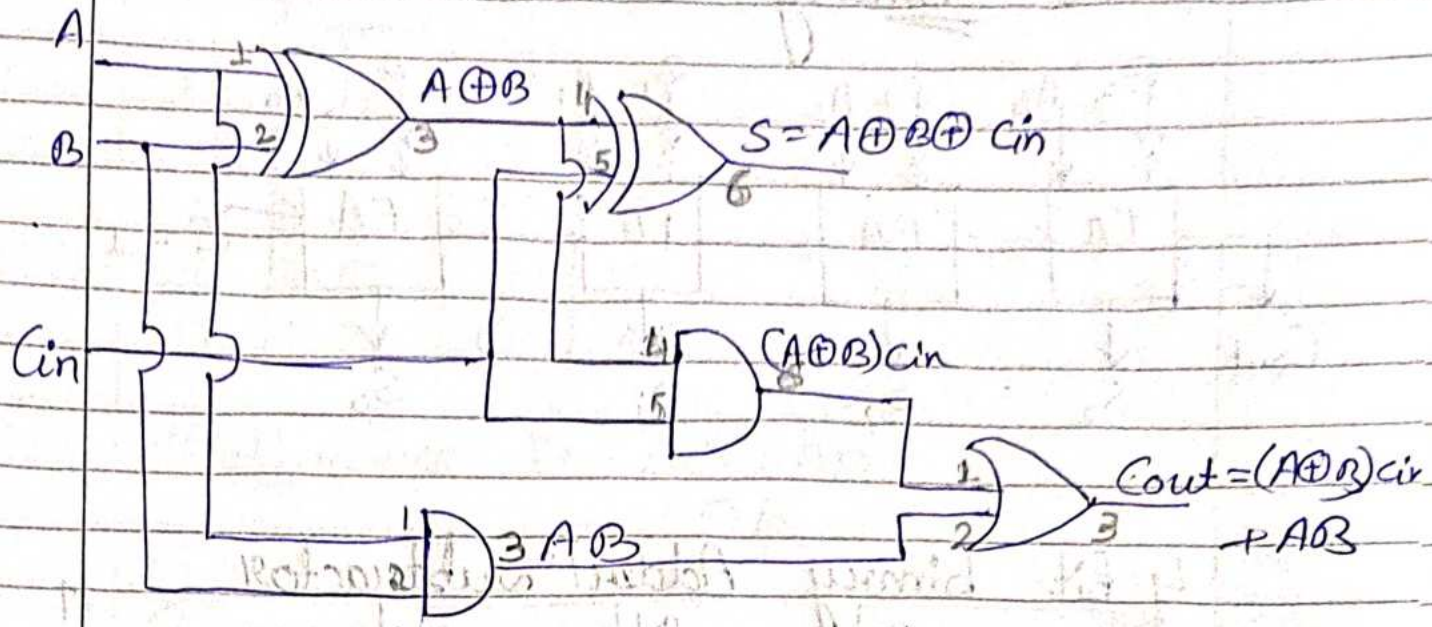
Boolean Expression → $A \oplus B \oplus Cin$

Sum ⇒ $\bar{A}\bar{B}Cin + \bar{A}B\bar{Cin} + A\bar{B}\bar{Cin} + ABCin$

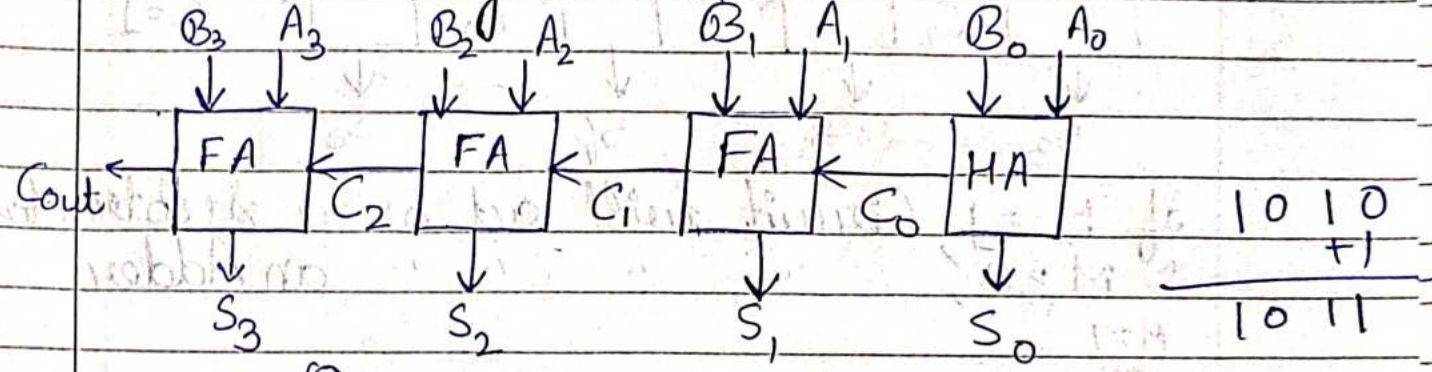
Carry ⇒ $(A \oplus B)Cin + A.B$

Date _/_/_

Circuit Diagram



4-bit Binary Adder IC-7483



1 0 1 1 ← A_3 A_2 A_1 A_0

1 0 0 1 ← B_3 B_2 B_1 B_0

(1) 0 1 0 0

Cout = 1

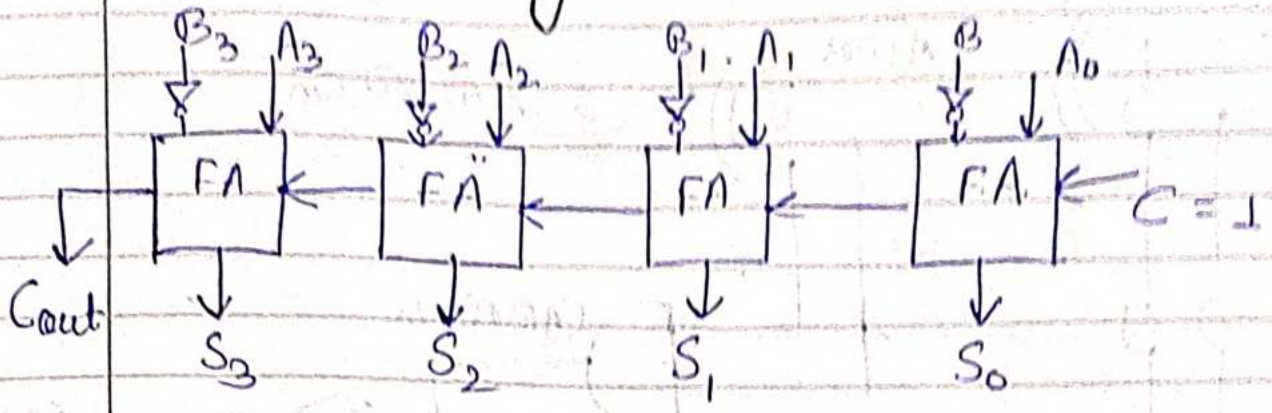
1 0 1 0
+1
1 0 1 1

1 1 1 0
0 1 0 1
1 0 1 1

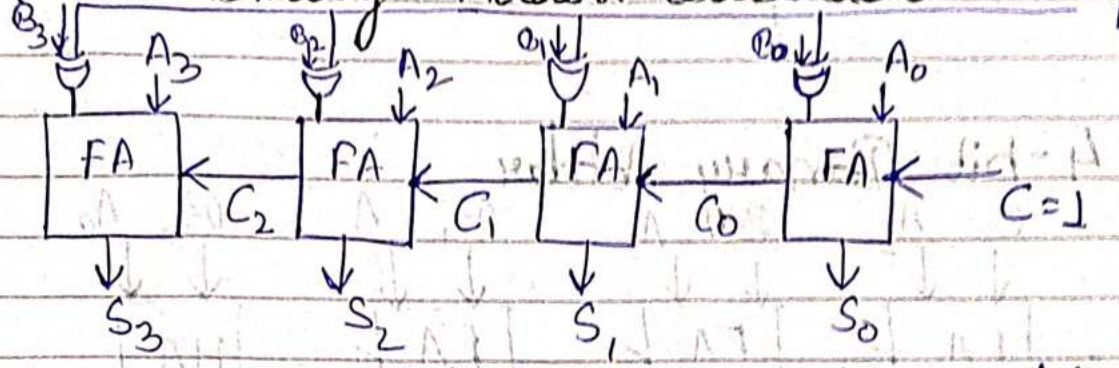
1 1 0 1 1 0 0 1
0 1 0 1
(1) 0 0 1 0

Date : / / :

4 bit Binary Subtractor.



4 bit binary Adder/ Subtractor



If M = 1 Circuit will act as a Subtractor
" M = 0 " " " " " an Adder.

- M = 1

$A\bar{B} + AB$

$1 \cdot \bar{B} + 0 \cdot B$

- M = 0

$0 \cdot \bar{B} + 1 \cdot B$

$= B$

Date _/_/_

★ Half Subtractor

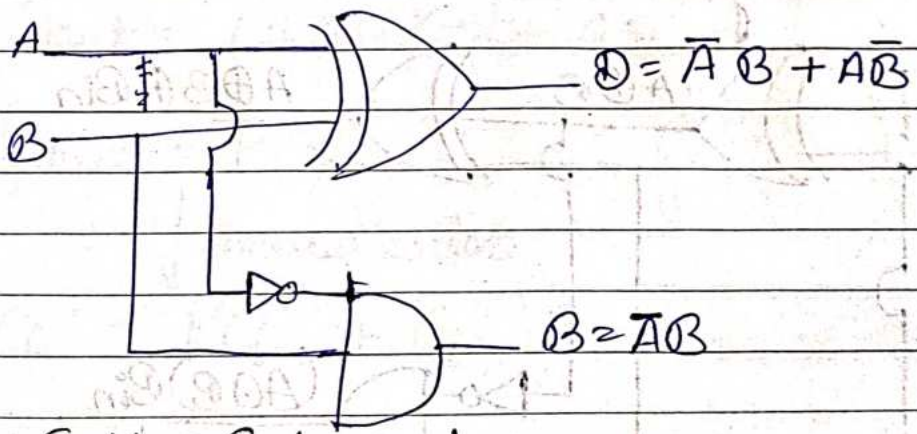
A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Difference, $D = \bar{A}B + A\bar{B}$

$= A \oplus B$

Borrow, $B = \bar{A}B$

Circuit Diagram



★ Full Subtractor

A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Date _/_/_

Difference $D = \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + \bar{A}\bar{B}B_{in} + AB\bar{B}_{in}$

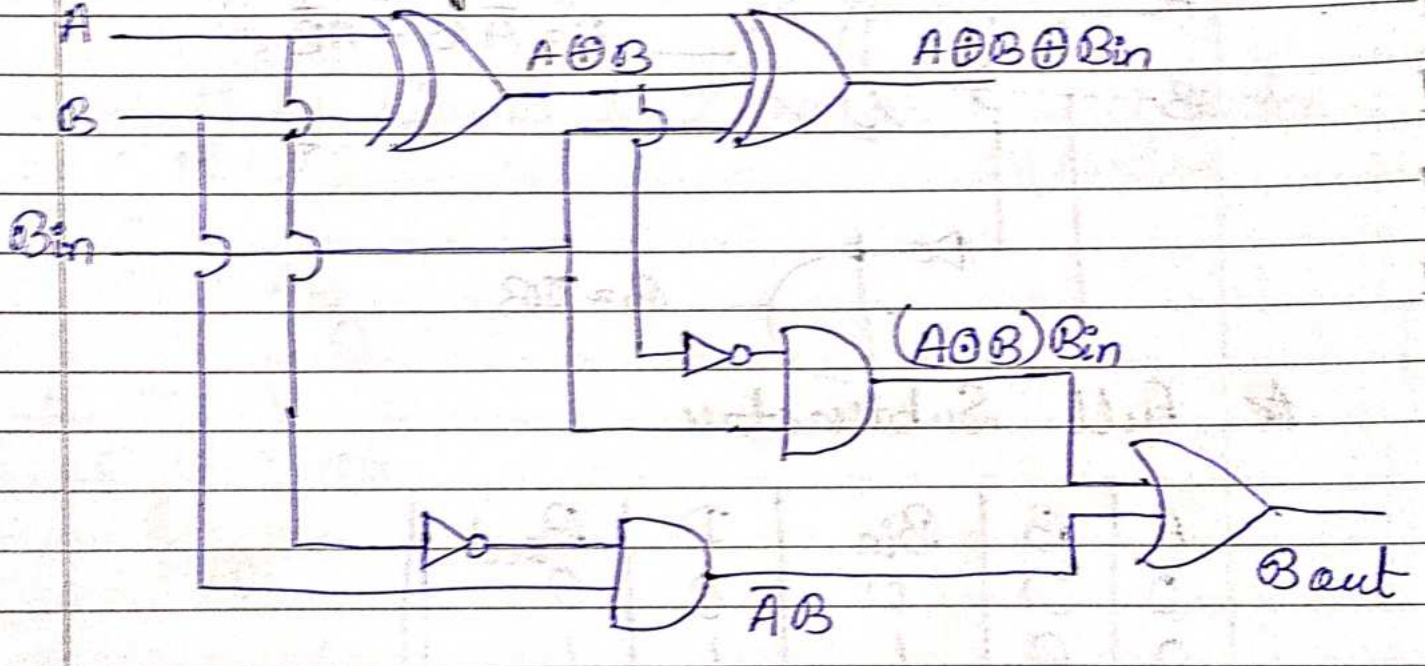
$\Rightarrow A \oplus B \oplus B_{in}$

$B_{out} = \bar{A}BB_{in} + \bar{A}B\bar{B}_{in} + \bar{A}\bar{B}B_{in} + AB\bar{B}_{in}$

$\Rightarrow B_{in}(AB + \bar{A}\bar{B}) + \bar{A}B(\bar{B}_{in} + B_{in})$

$B_{out} \Rightarrow B_{in}(A \odot B) + \bar{A}B$

Circuit Diagram.



Date _/_/_

★ Carry Look ahead adder

A	B	Cin	Cout
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$P = A \oplus B$

$G = AB$

$C_{out} = (A \oplus B) C_{in} + AB$

$C_{out} = P C_{in} + G$

↓ Generalization

$C_i = P C_{i-1} + G_i$

$i = 0$

$C_0 = P_0 C_{-1} + G_0$

$i = 1$

$C_1 = P_1 C_0 + G_1$

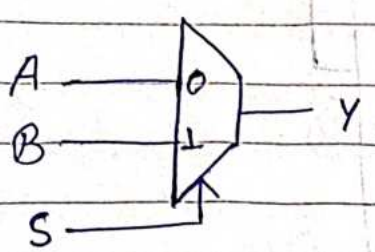
$C_1 = P_1 (P_0 C_{-1} + G_0) + G_1$

$= P_0 P_1 C_{-1} + P_1 G_0 + G_1$

Date _/_/_

Multiplexer

1) 2×1 Multiplexer



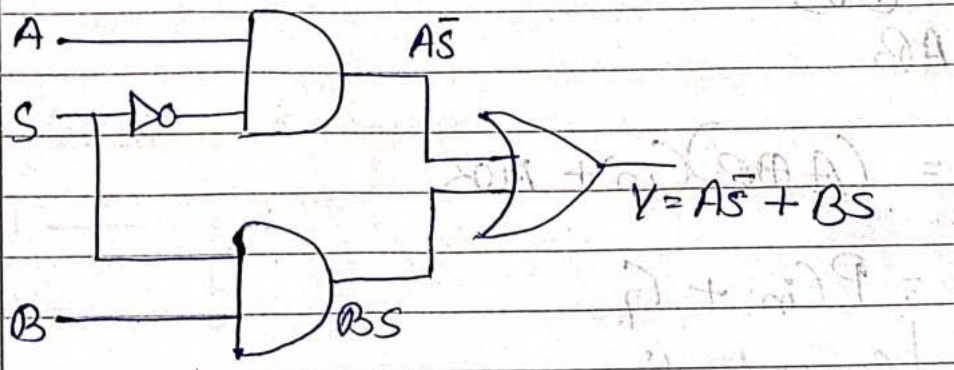
S	Y
0	A
1	B

S = Select Line

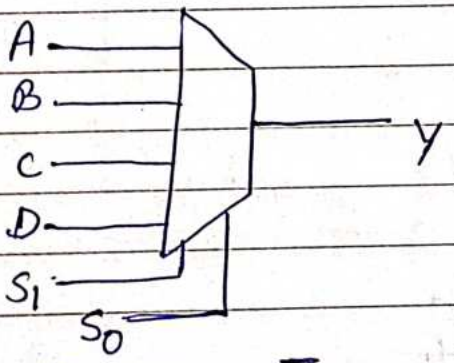
Boolean Expression →

$$Y = \bar{S}A + SB$$

Circuit Diagram



2) 4×1 Multiplexer (MUX)

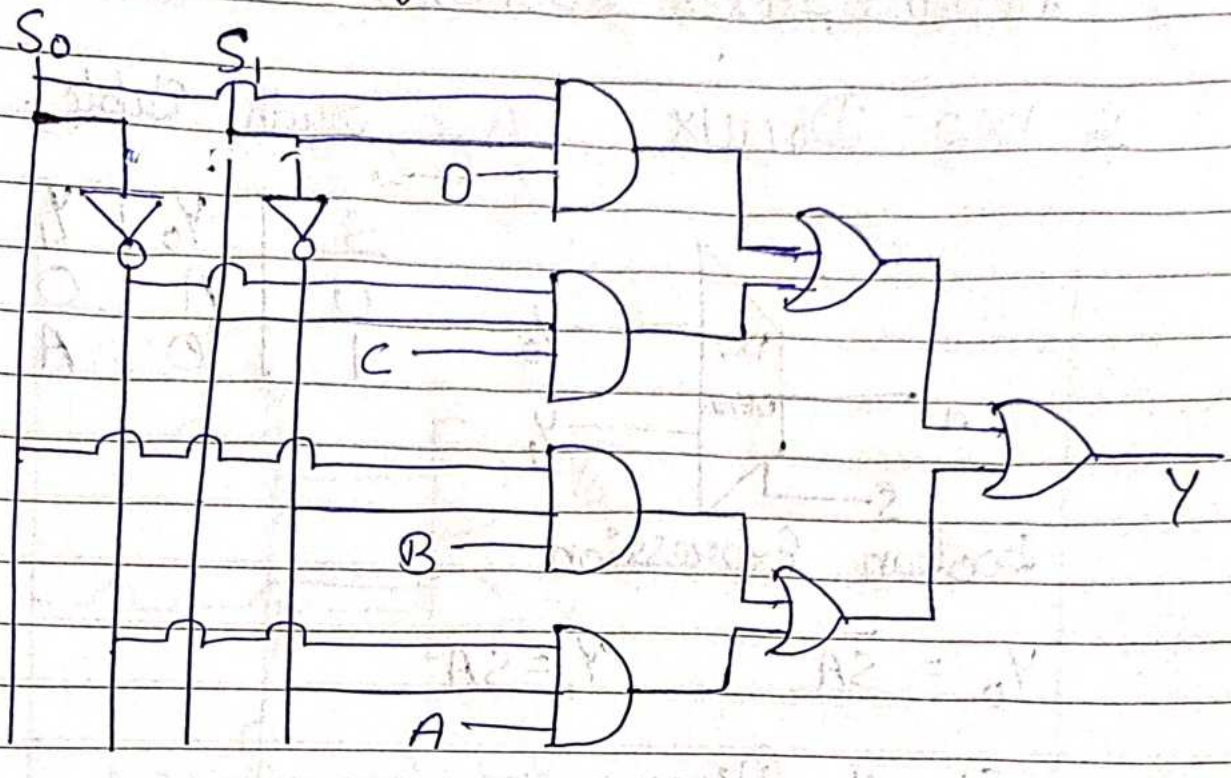


S_1	S_0	Y
0	0	A
0	1	B
1	0	C
1	1	D

$$Y = \bar{S}_0\bar{S}_1A + \bar{S}_1S_0B + S_1\bar{S}_0C + S_1S_0D$$

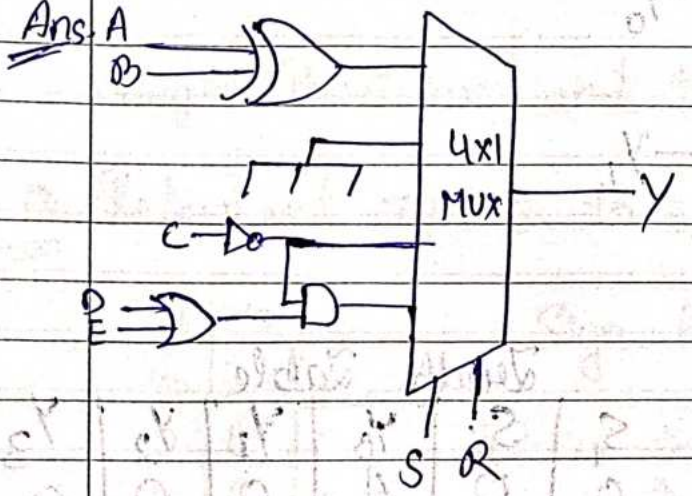
Date _/_/_

Circuit Diagram



Q1. Write down the TT and Boolean Expression of the given Diagram.

Ans.



Truth Table

S	R	Y
0	0	$A \oplus B$
0	1	0
1	0	$\bar{C}$
1	1	$(D+E)\bar{C}$

$$Y = \bar{S}R(A \oplus B) + \bar{S}R(0) + S\bar{R}(\bar{C}) + SR\bar{C}(D+E)$$

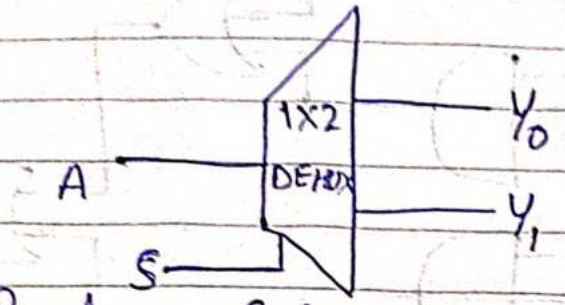
Date _/_/_

Demultiplexer (DEMUX)

1) 1X2 DEMUX

Truth Table

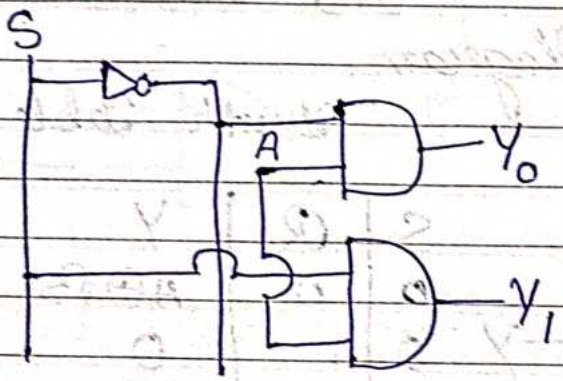
S	Y_0	Y_1
0	A	0
1	0	A



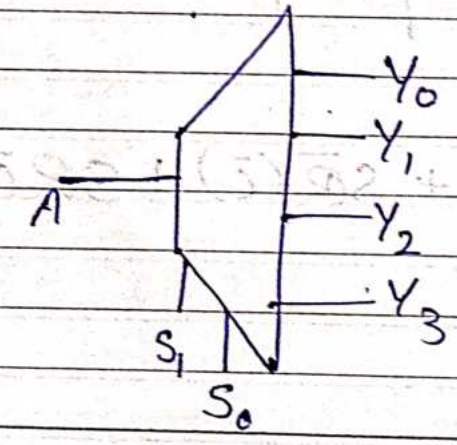
Boolean Expression

$Y_0 = \bar{S}A$, $Y_1 = SA$

Circuit Diagram.



2) 1X4 DEMUX



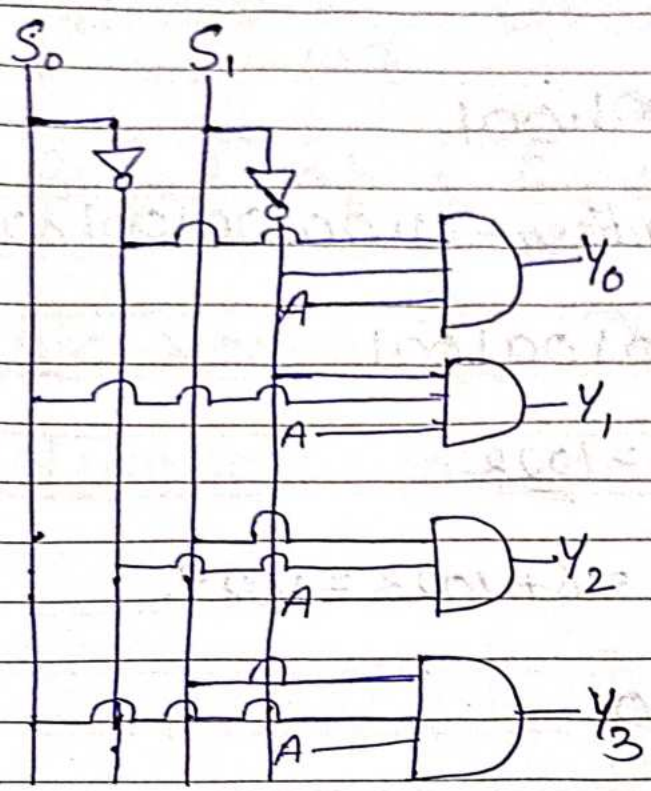
Truth Table

S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	A	0	0	0
0	1	0	A	0	0
1	0	0	0	A	0
1	1	0	0	0	A

Date _/_/_

$Y_0 = \bar{S_0}\bar{S_1}A$, $Y_1 = \bar{S_1}S_0A$, $Y_2 = S_1\bar{S_0}A$

$Y_3 = S_1S_0A$



Single Precision and Double Precision

Q. Represent 73.125 to 32 bit and 64 bit Precision

2	73	1
2	36	0
2	18	0
2	9	1
2	4	0
2	2	0
	1	

Sign bit = 0

$73.125 \Rightarrow 1001001.001$

$\Rightarrow 1.001001001 \times 2^6$

Mantissa $\Rightarrow$ 00100100 1

Biased-Exponent = 6 + 127

= 133

Size $\rightarrow 2^{8-1} - 1$

= 127

$133 \rightarrow 10000101$

Date _/_/_

0	10000101	00100100100000000000000

← 1 → ← 8 → ← 23 bit →

64-bit

73.125 = 1001001.001

Normalized Mantissa - 1.001001001×2^6

Mantissa ⇒ 001001001

$2^{size-1} - 1 = 2^{11-1} - 1 = 1023$

Biased exponent = 6 + 1023 = 1029.

1029 - 10000000101

0	10000000101	0010010010 → all zeroes upto 52

2. Represent 84.05 to 32-bit and 64-bit.

Sol.

2	84	0
2	42	0
2	21	1
2	10	0
2	5	1
2	2	0
	1	

.05 × 2 = 0.10
.05 × 2 = 0.10
.1 × 2 = 0.2
.2 × 2 = 0.4
.4 × 2 = 0.8
0.8 × 2 = 1.6

84.05 → 1010100.00001...

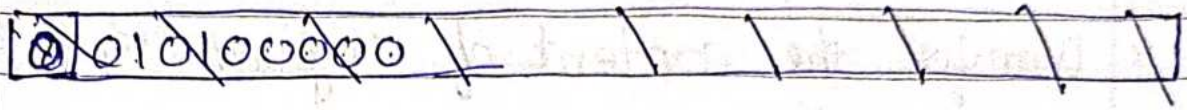
Date _/_/_

Normalised Mantissa → 1.0101000000×10^6

Mantissa → 0101000000

$2^{8-1} - 1 = 127$

Biased exp. = 6 + 127 = 133



0	10000101	01010000000000000000000

Date _/_/_

☆ MICRO-OPERATION

Types of micro-operation

→ Register transfer micro-operation

Register → ~~fitted~~ present in ALU.

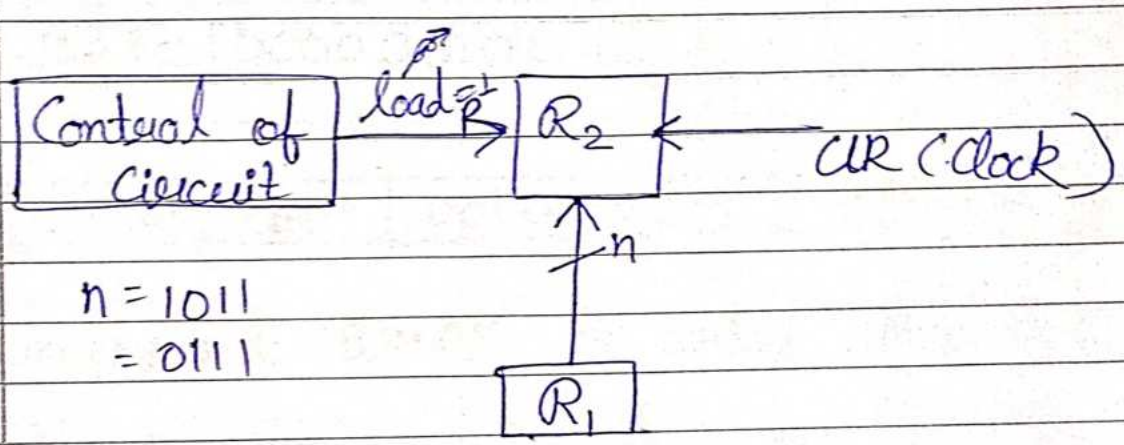
$R_2 \leftarrow R_1$

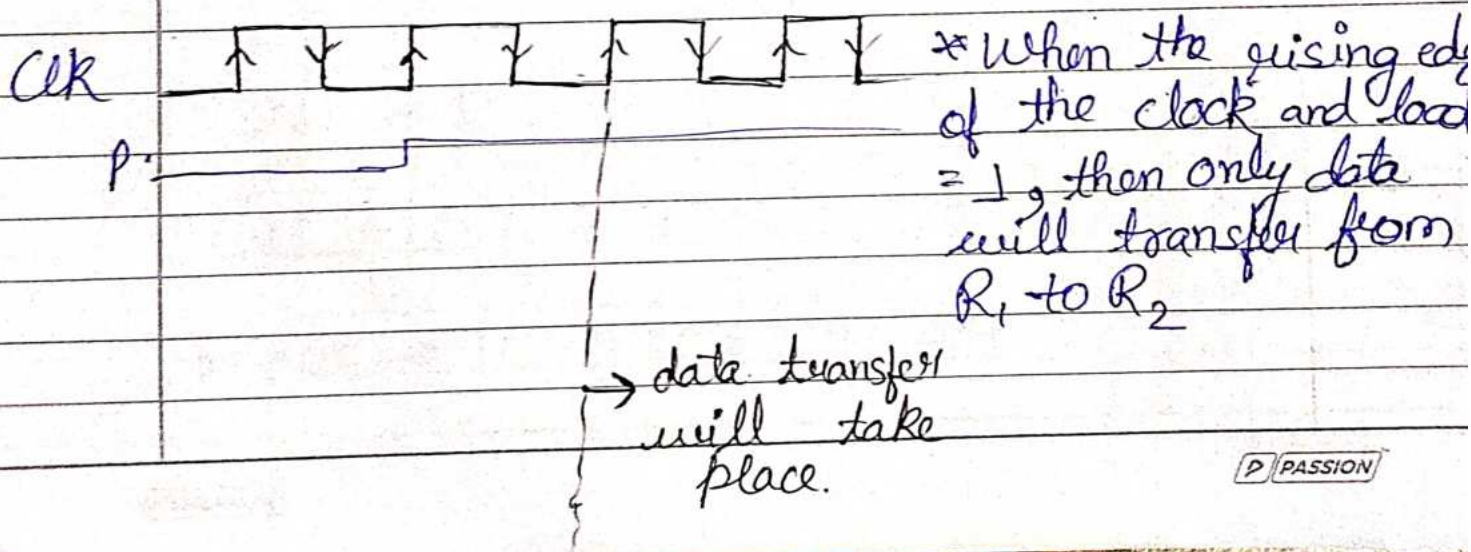
* Transfer the content of register R_1 to R_2

* If ($P == 1$)

$R_2 \leftarrow R_1$

*





Date _ / _ / _

→ Arithmetic Micro-operation

$R_3 \leftarrow R_1 + R_2$

$R_3 \leftarrow R_1 - R_2$

$R_3 \leftarrow R_1 + \overline{R_2} + 1$

$R_1 \leftarrow \overline{R_1}$

$R_1 \leftarrow \overline{R_1} + 1$

$R_1 \leftarrow R_1 - 1$

$R_1 \leftarrow R_1 + 1 \Rightarrow$ Increment

→ Logic - Micro-operation

Boolean Function	Micro-operation
$F_0 = 0$	$F \leftarrow 0$
$F_1 = xy$	$F \leftarrow X \wedge Y$
$F_2 = xy'$	$F \leftarrow X \wedge \overline{Y}$
$F_3 = x$	$F \leftarrow X$
$F_4 = x'y$	$F \leftarrow \overline{X} \wedge Y$
$F_5 = y$	$F \leftarrow Y$
$F_6 = x \oplus y$	$F \leftarrow X \oplus Y$
$F_7 = x + y$	$F \leftarrow X \vee Y$
$F_8 = (x+y)'$	$F \leftarrow (X \vee Y)'$
$F_9 = (x \oplus y)'$	$F \leftarrow (X \oplus Y)'$
$F_{10} = y'$	$F \leftarrow Y'$

Date _/_/_

$F_{11} = x + y'$	$F \leftarrow X \vee \bar{Y}$
$F_{12} = x'$	$F \leftarrow \bar{X}$
$F_{13} = x' + y$	$F \leftarrow \bar{X} \vee Y$
$F_{14} = (xy)'$	$F \leftarrow (X \wedge Y)'$
$F_{15} = 1$	$F \leftarrow 1$

→ Shift Micro-operation

shlR → Shift-left the content of Register R

shrR → Shift-right " " " " "

cilR → Circular-Shift-left the content of Reg. R

cirR → Circular-shift-right " " " " "

AshlR → Arithmetic shift-left the content of Reg. R.

AshrR → " " right " " " " "

☆ Shift
MSB ⟶ LSB

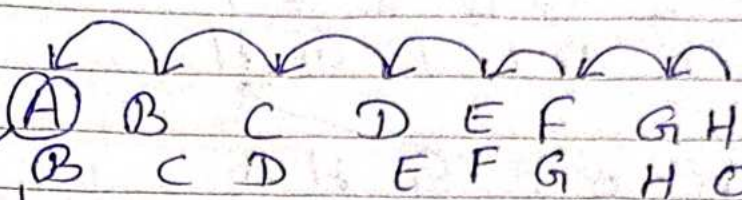
10 → right shift, then no. will be divided by 2.
↓
5

LSB ⟶ MSB

↓ left shift, then no. will be multiplied by 2.
20

Date _/_/_

* Shift Left Operation (shl)



Discarded.

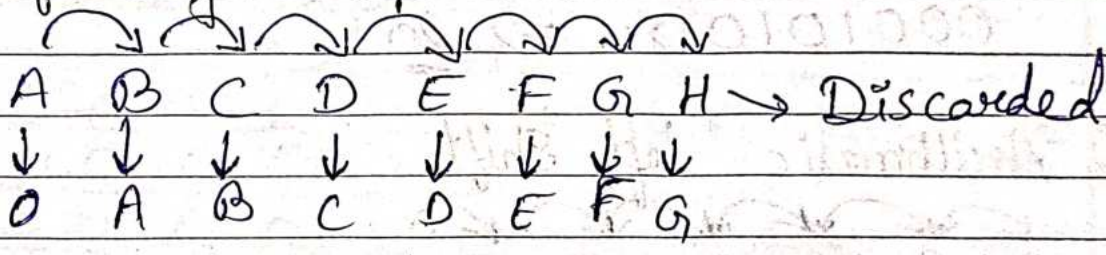
Eg. 1.) 00001010 ⇒ 10
L.S. By 1

00010100 ⇒ 20

128 64 32 16 8 4 2 1
2.) 0010101000 ⇒ 168

0101010000 ⇒ 336

* Shift Right Operation (shr)



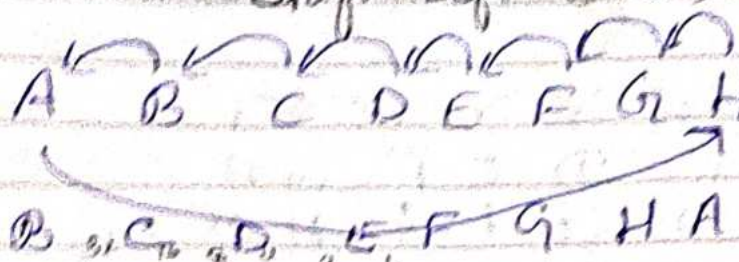
32 16 8 4 2 1
1. 00101000 ⇒ 40
00010100 ⇒ 20

64 32 16 8 4 2 1
2.) 01010100 ⇒ 84

00101010 ⇒ 42

Date _/_/_

* Circular Shift Left (CSL)



eg. 1) 00101000 ⇒ 40

01010000 = 80

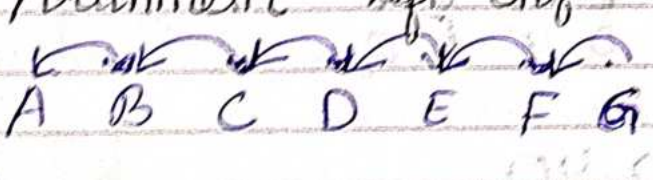
* Circular Shift Right (CSR)

eg. 1) 00101000

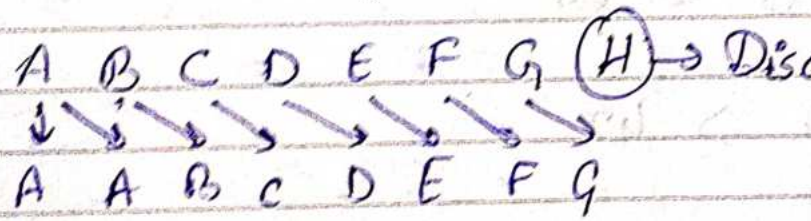
"2 ur 2"

00010100 ⇒ 20

* 2) Arithmetic left Shift



* Arithmetic Right shift



Date _/_/_

☆ Selective Set

set

Set → 1
Reset → 0

Ex. 1 0 0 1
+ 0 1 1 0
1 1 1 1

Perform OR operation

☆ Selective Complement

In this, we use XOR gate

1 0 1 0
⊕ 0 0 1 1
1 0 0 1

☆ Selective Clear

→ Clear

0 0 1 1
0 0 0 0
0 0 0 0

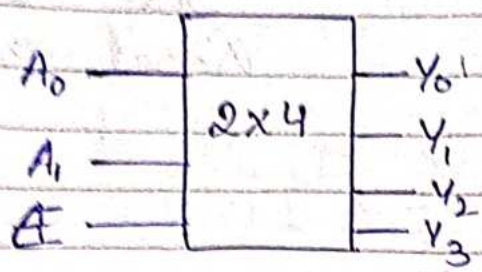
Use AND gate

☆ Decoder

① 2×4 Decoder ② 3×8 Decoder

* for n i/p lines, there will be 2^n o/ps

Date _/_/_



E → Enable

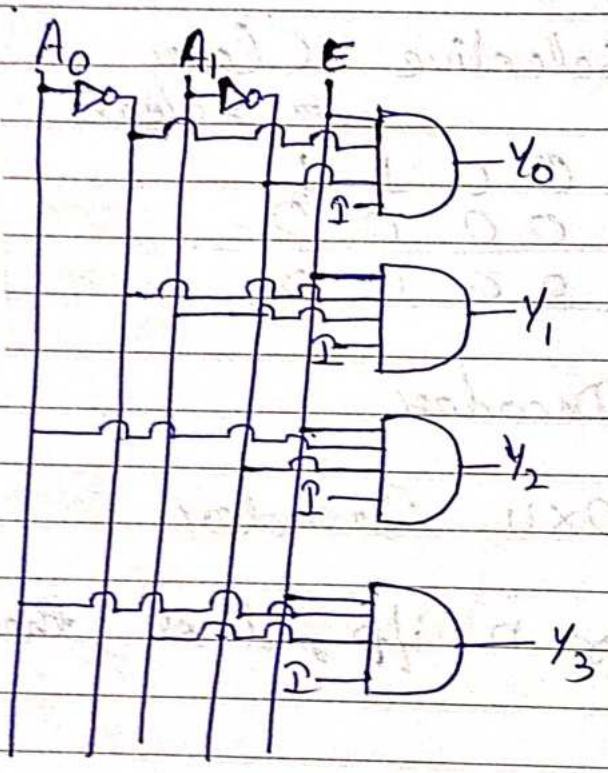
E	A_0	A_1	Y_0	Y_1	Y_2	Y_3
0	x	x	x	x	x	x
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$Y_0 = E.\bar{A_0}\bar{A_1}$

$Y_1 = E\,\bar{A_0}A_1$

$Y_2 = E\,A_0\bar{A_1}$

$Y_3 = EA_0A_1$

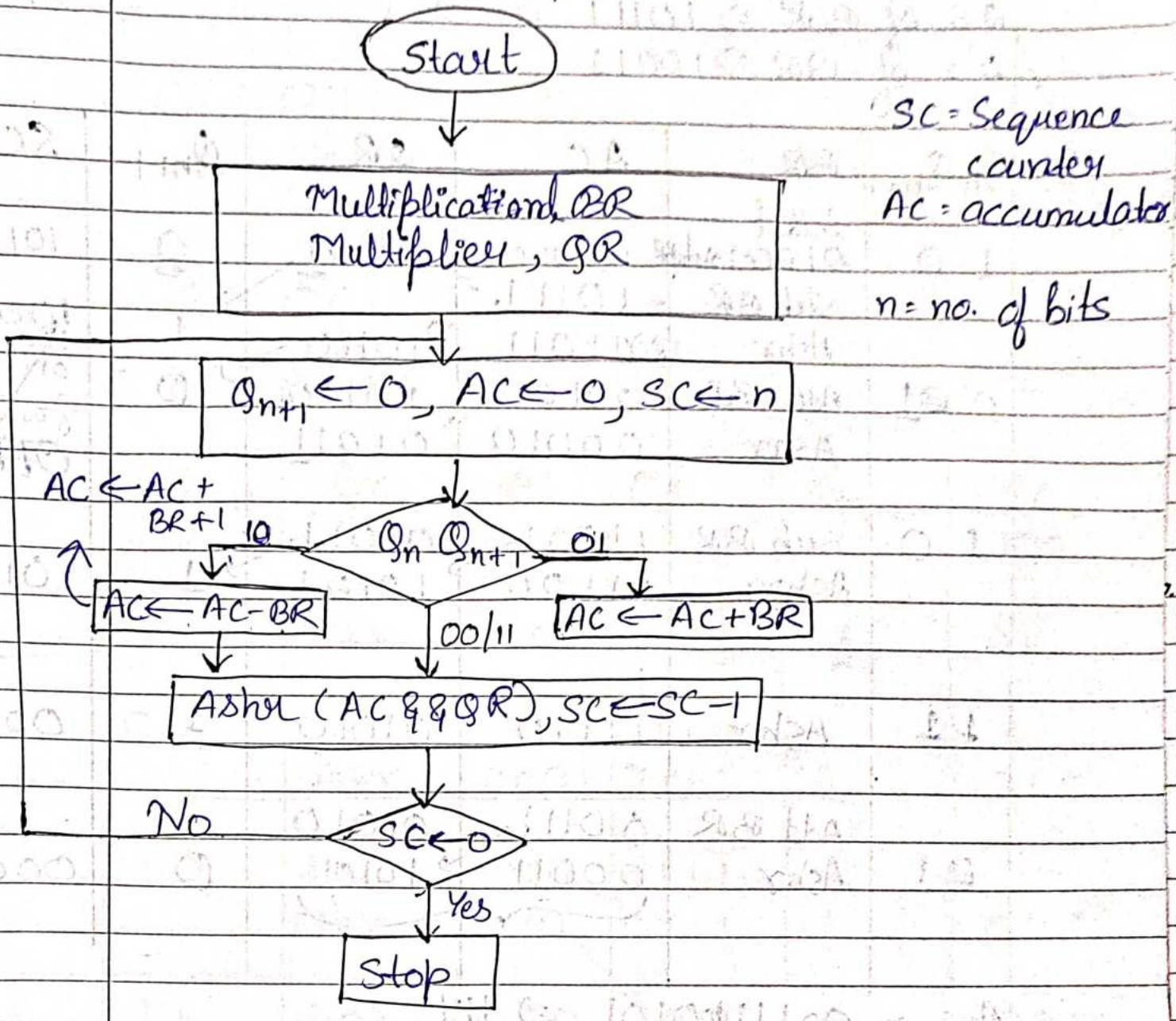


A → Multiplicand
X B → Multiplier

Date _/_/_

Booth's Multiplication Algorithm

→ We perform the multiplication by using shift and addition operation.



Q_n → LSB of QR

Date: / / /

Q1. Multiply 9 X 13 using Booth's Algo

BR = 9 = 01001
QR = 13 = 01101

Range = $-(2^{n-1}-1)$ to 2^{n-1}
n = 5
= −15 to 16

SC = 5
= 101

2's of BR ⇒ 10111
2's of QR ⇒ 10011

Q_n Q_{n+1}	BR	AC	QR	Q_{n+1}	SC
	Initial	00000	01101	0	101
1 0	Sub BR	−10111			
	Ashr	11011	10110	1	100
0 1	Add BR	00100	10110	0	011
	Ashr	00010	01011		010
1 0	Sub BR	11001	01011		
	Ashr	11100	10101	1	001
1 1	Ashr	11110	01010	1	000
0 1	Add BR	01111	01010		
	Ashr	00111	10101	0	

Ans ⇒ 00111 & 01010 ⇒ 117

Date _/_/_

2. Multiply +7 x -3 using booth's Algorithm

Sol. Range $= -2^{n-1}$ to 2^{n-1}

$n = 4$

$= -7$ to $+8$

$7 = 0111 \leftarrow BR$

↓ 2's

1001

$3 = 0011$

↓ 2's

$QR \Leftrightarrow 1101$

$Q_n Q_{n+1}$	BR	AC	QR	Q_{n+1}	SC
10	initial	0000	1101	0	100
	sub.	1001	1101 →		
	Ashr	1100	1110	1	011
01	add	0011 →	1110 →	0	
	Ashr	0001	1111		010
10	sub	1010 →	1111		
	ashr	1101	0111	1	001
11	ashr	1110	1011	1	000

Ans → 1110 1011 = -21

0001 0101

+ 1001 0111

PASSION

AC-(-BR)

Date _/_/_

3.) Multiply (-16 x 18) using Booth's Algo

Sol. $16 \rightarrow 01.0000 \rightsquigarrow n=6$

$18 \rightarrow 010010 \longrightarrow QR$

2's $16 \Rightarrow 10000$

$+ \quad 1$

$\underline{110000} \rightarrow BR$

2's $18 \Rightarrow 101101 \rightsquigarrow$

$+ \quad 1$

$\underline{101110} \Rightarrow$

$Q_n Q_{n+1}$	BR	AC	QR	Q_{n+1}	SC
	Initial	000000	010010	0	110
00	Ashr	000000	001001	0	101
10	sub.	010000	001001		
	Ashr	001000	000100	1	100
01	add	111000	000100		
		111100	000010	0	11
00	ashr	111110	000001	0	10
10	sub	001110	000001		
	ashr	000111	000000	1	01
01	add	110111	000000		
	ashr	111011	000000	0	00

PASSION

Date _/_/_

111011100000

1001000111111

+ 1

1001001000000 ⇒ -288

Q. -8x-9 using Booth's Algorithm

8 → 1000

9 → 1001

Range = $-(2^{n-1}-1)$ to 2^{n-1}

= -15 to 16

8 complement ⇒ 01000 → BR

9 " ⇒ 01+1 10111 → QR

$Q_n Q_{n+1}$	BR	AC	QR	Q_{n+1}	SC
10	Initial	00000	10111	0	101
	Sub	01000	10111		
	Ashr	00100	01011	1	100
11	Ashr	00010	00101	1	011
11	Ashr	00001	00010	1	10
01	add	11001	00010		
	Ashr	11100	10001	0	01
10	Sub	00100	10001		
		00010	01000	1	001

000111
110000
110111

110111
010000

110111
110000
100111

-8 X -9 = 0001001000 ⇒ 72

110011|100000

use in selection of register.

Date _/_/_

☆ 3×8 Decoder

For n i/p = 2^n of o/p

$2_2 \times 4$

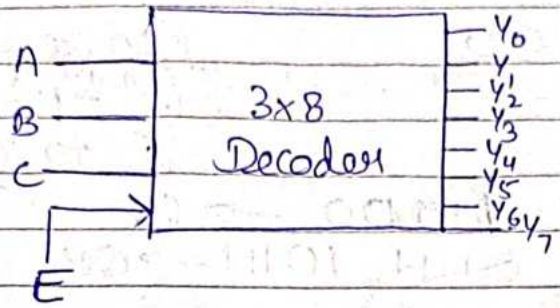
2

$= 4.$

if i/p = 3

$2^3 = 8$

3×8 Decoder



E	A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_6	Y_7
0	X	X	X	X	X	X	X	X	X	0
1	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0
1	1	0	0	0	0	0	0	1	0	0
1	1	0	1	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	1

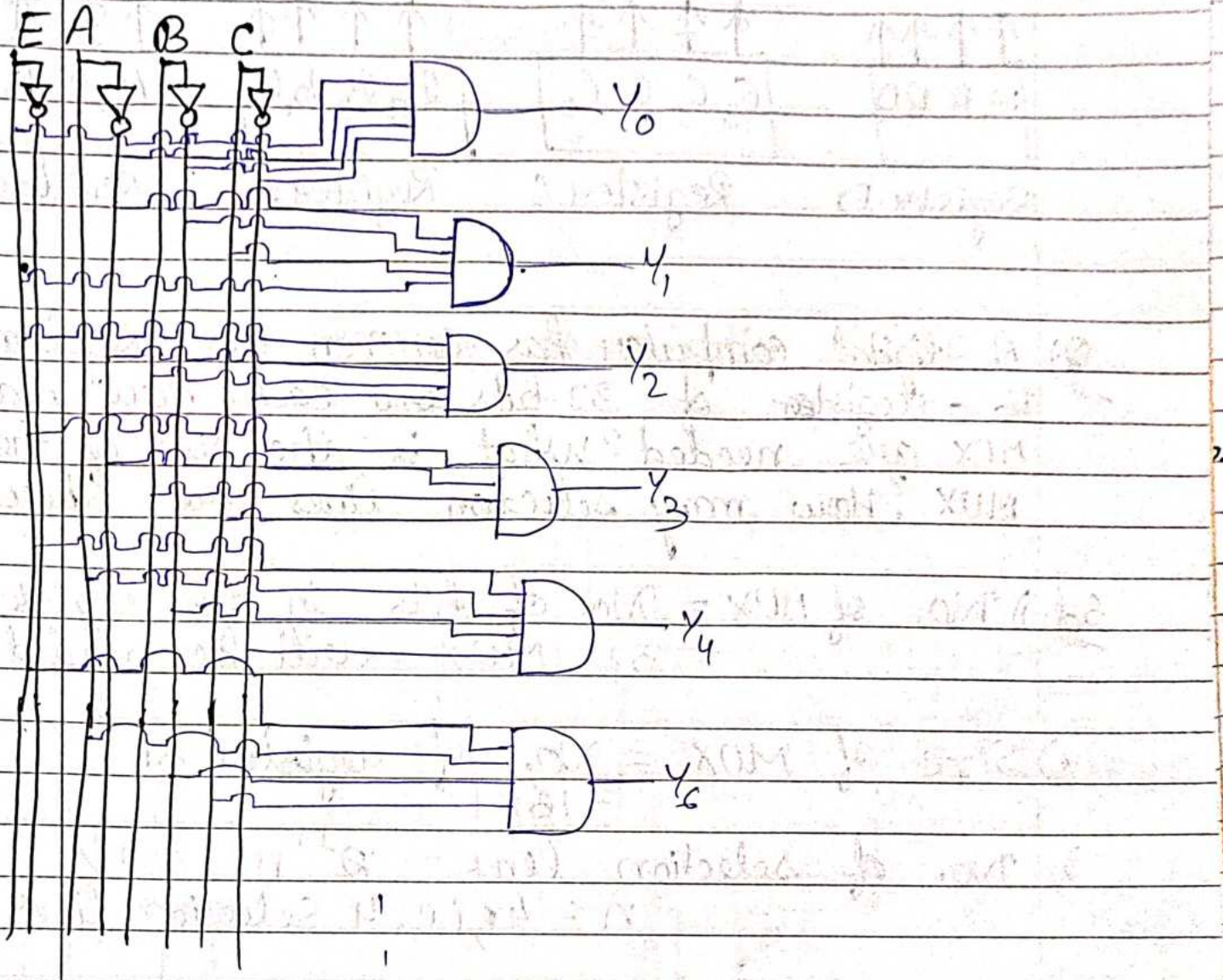
Date _/_/_

$Y_0 = E\bar{A}\bar{B}\bar{C}$, $Y_1 = \bar{A}\bar{B}CE.1$, $Y_2 = E\bar{A}B\bar{C}$

$Y_3 = E\bar{A}BC$, $Y_4 = EA\bar{B}\bar{C}$, $Y_5 = EA\bar{B}C$

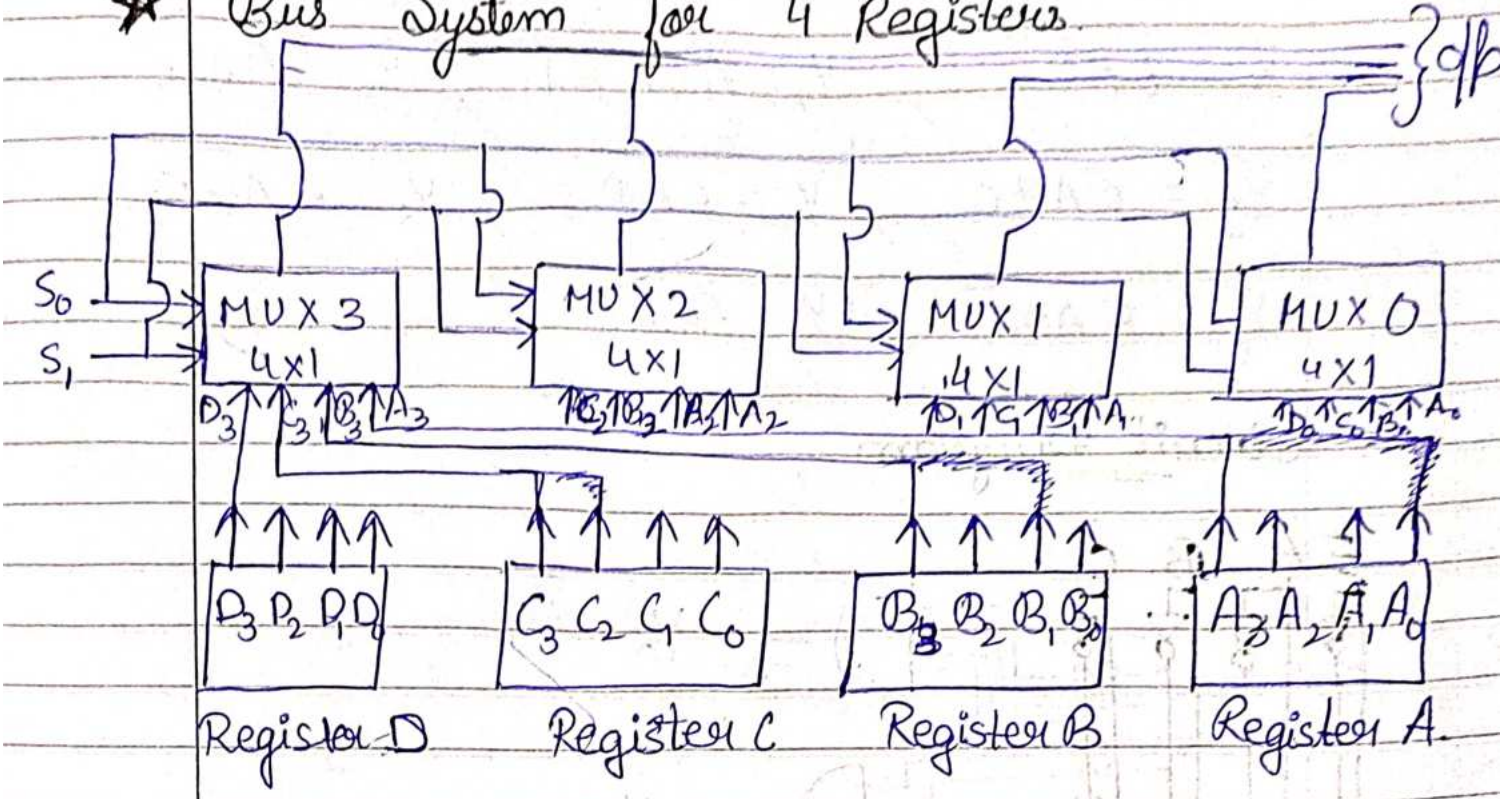
$Y_6 = EAB\bar{C}$, $Y_7 = EABC$

Circuit Diagram



Date _/_/_

★ Bus System for 4 Registers



Q.1 A digital computer has common bus system for 16 - register of 32-bits ~~ss~~ each. How many MUX are needed? What is the size of the MUX? How many selection lines are there?

Sol. 1) No. of MUX = No. of bits in the register.
= 32 MUX will be needed

2) Size of MUX = No. of register x1
= 16x1

3) No. of selection line = 2^n n (2^n)
n = 4, i.e. 4 Selection lines.

Date _/_/_

Q2.

32 - Registers
64 - bits each

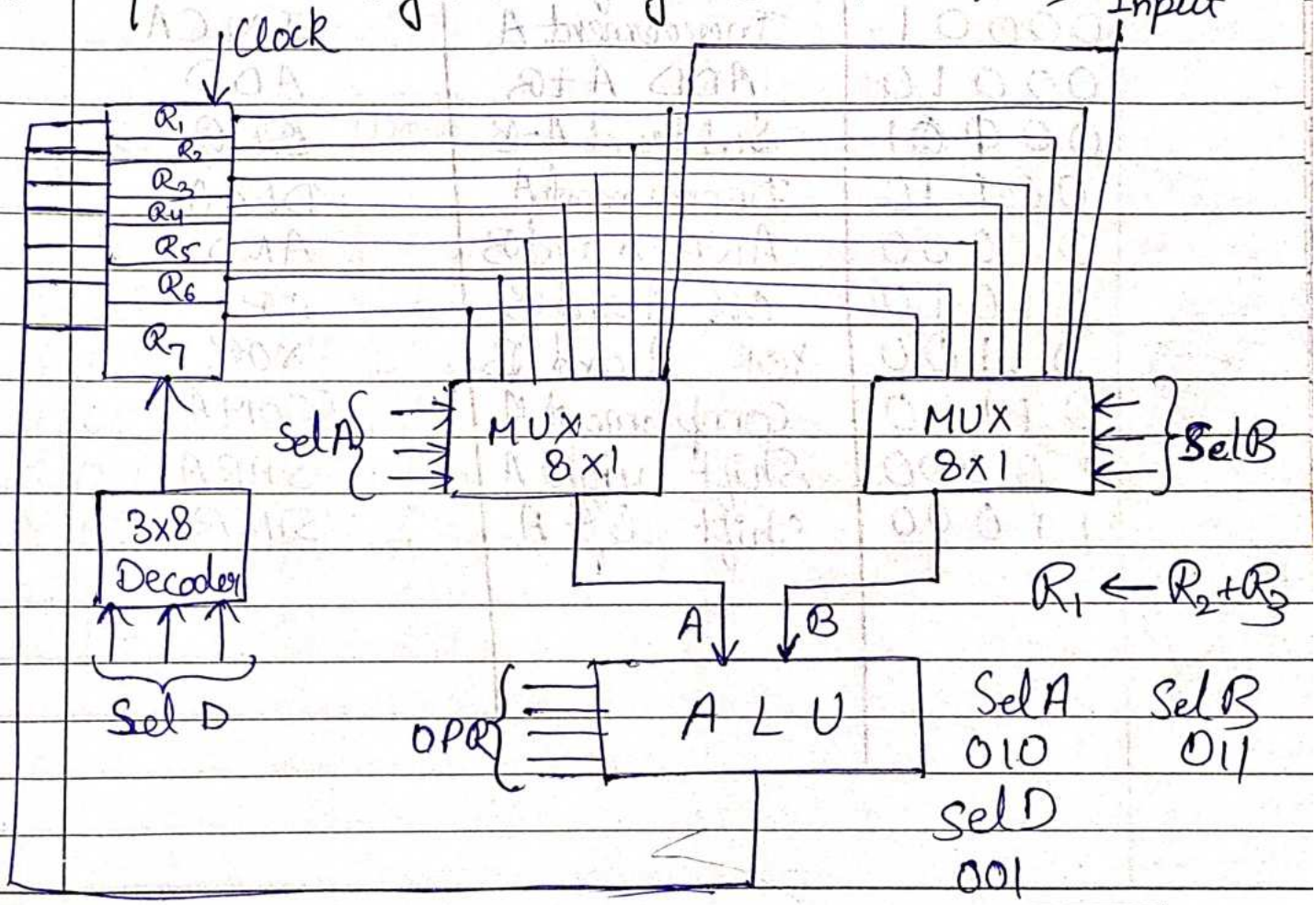
Sol 1) No. of MUX = 64 bits

2) Size of MUX = 32×1

3) No. of selection line =

$$2^n = 32$$
$$2^n = 2^5$$
$$n = 5$$

★ General Register Organization (GRO)



Date _/_/_

Binary Code	Sel A	Sel B	Sel D
000	Input	Input	None
001	R_1	R_1	R_1
010	R_2	R_2	R_2
011	R_3	R_3	R_3
100	R_4	R_4	R_4
101	R_5	R_5	R_5
110	R_6	R_6	R_6
111	R_7	R_7	R_7

OPR	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	ADD A+B	ADD
00011	Subtract A-B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift right A	SHRA
11000	Shift left A	SHLR

Date _/_/_

Control Word

Ex: 1

	Microoperation	SelA	SelB	SelD
1	$R_1 \leftarrow R_2 - R_3$	R_2	R_3	R_1
2	$R_4 \leftarrow R_4 \vee R_5$	R_4	R_5	R_4
3	$R_6 \leftarrow R_6 + 1$	R_6	—	R_6
4	$R_7 \leftarrow R_1$	R_1	—	R_7
5	output $\leftarrow R_2$	R_2	—	None
6	Output $\leftarrow$ Input	Input	—	None
7	$R_4 \leftarrow$ Shl R_4	R_4	—	R_4
8	$R_5 \leftarrow 0$	R_5	R_5	R_5

Control word.

1	010	011	001	00101
2	100	101	100	01010
3	110	000	110	00001
4	001	000	111	00000
5	010	000	000	00000
6	000	000	000	00000
7	100	000	100	11000
8	101	101	101	01100

Date _/_/_

Ex.-2

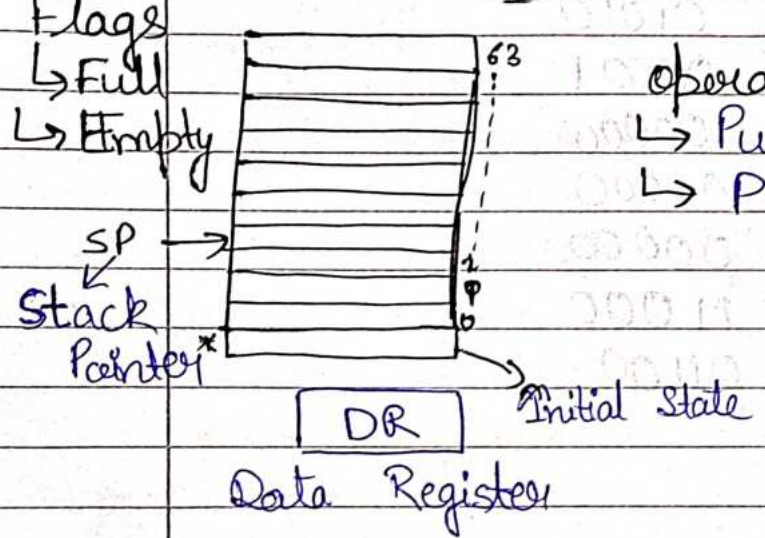
Microoperation	Sel A	Sel B	Sel D
$R_1 \leftarrow R_2 + R_3$	R_2	R_3	R_1
$R_4 \leftarrow R_4$	R_4	—	R_4
$R_5 \leftarrow R_5 - 1$	R_5	—	R_5
$R_6 \leftarrow Shl R_1$	R_1	—	R_6
$R_7 \leftarrow$ Input	Input	—	R_7

Control word

8421 101

	010	011	001	010
8421	100	—	100	00000
	101	—	101	00010
	110	—	110	11000
	111	—	—	00000

STACK ORGANISATION



operation
- PUSH
- POP

PUSH
$SP \leftarrow SP + 1$
$M[SP] \leftarrow DR$
(address of block)

if $(SP \leftarrow 0)$ then full

Empty $\leftarrow 0$

Date _/_/_

POP

DR ← M[SP]
SP ← SP - 1

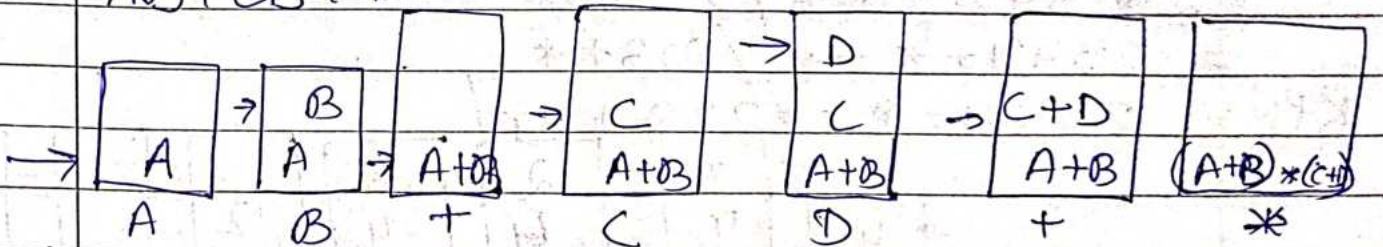
If (SP ← 0) then Empty ← 1
Full ← 0

★ Reverse Polish Notation (RPN)

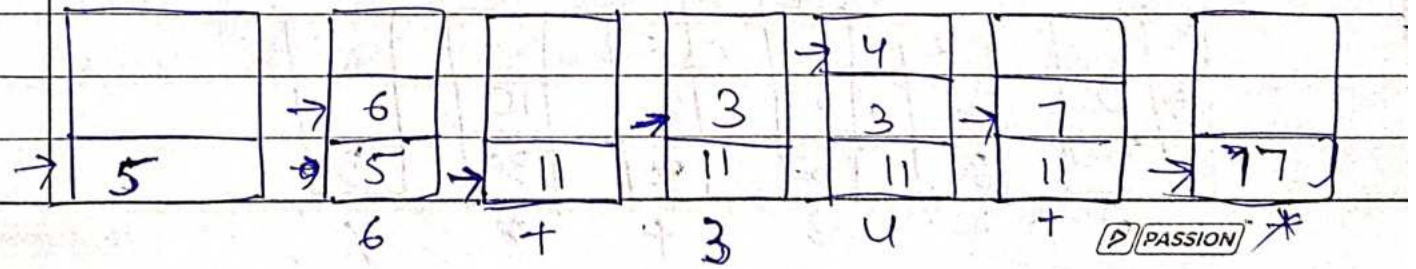
A+B → Infix Notation
+AB → Prefix or Polish Notation
AB+ → Postfix or RPN.

Priorities :-
[ ] { } ( )
^
* /
+ −

1) (A+B) * (C+D)
AB+ * CD+
AB+CD+ *



2) (5+6) * (3+4)
56 +34 *



Date _/_/_

ii) $(3+4)*[10*(2+6)+8]$

$(3+4)*[10*26++8]$

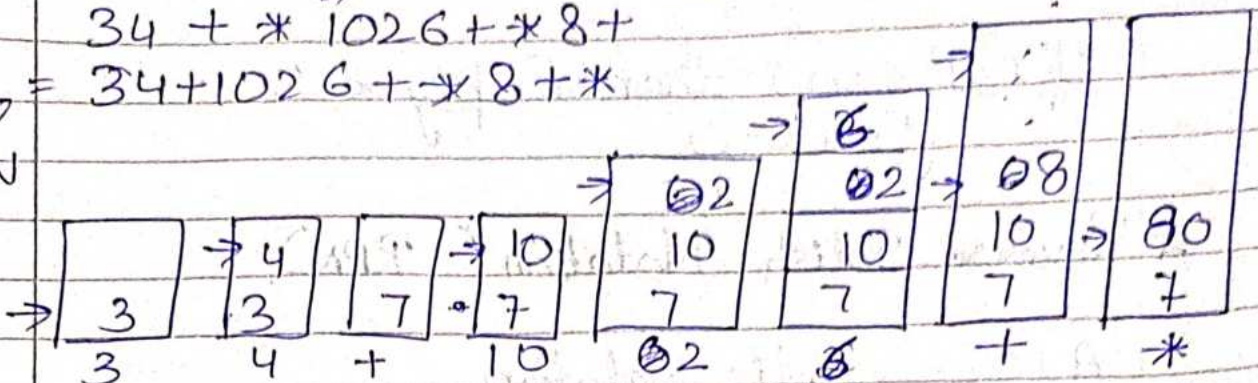
$(3+4)*[1026+*+8]$

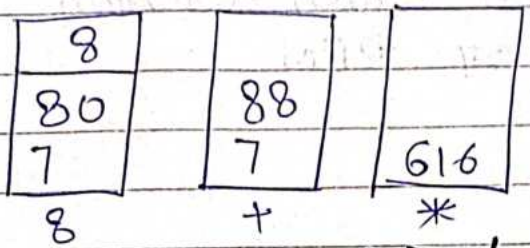
$(3+4)*[1026+*8+]$

$34+*1026+*8+$

$= 34+1026+*8+*$

RPN





iii) $2*[3+4*(2+1)]/(2*(2+3))$

$= 2*[3+4*21+]/2*(2+3)$

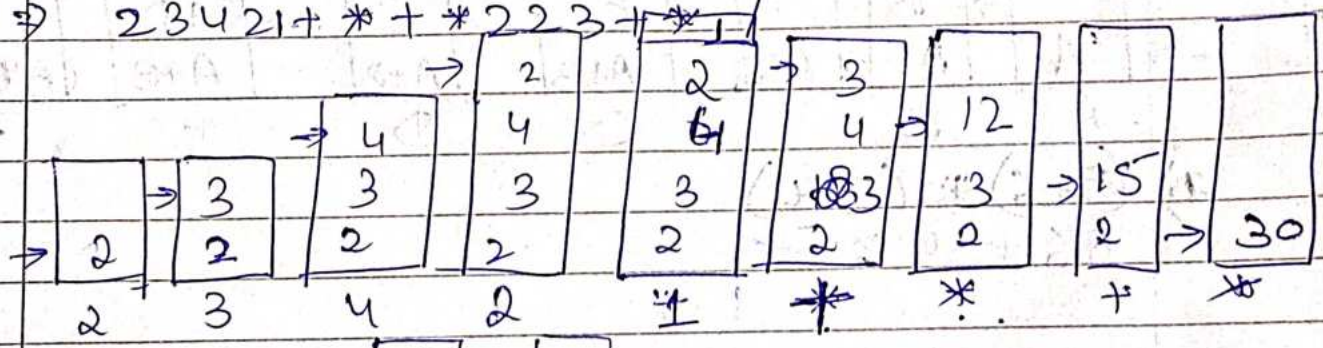
$= 2*[3+421+*]/2*(2+3)$

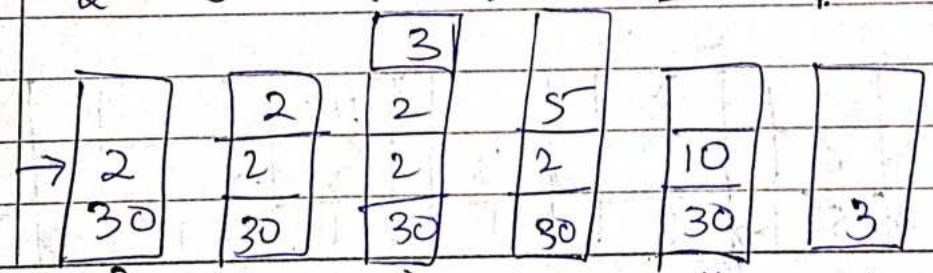
$= 2*3421+*+/2*23+$

$= 2*3421+*+/223+*$

$= 23421+*+*/223+*$

RPN ⇒ $23421+*+*223+*/$





Date _/_/_

Computer Instruction / Instructions format

Mode	Op code	Operand Address

↓ operation to be performed
+, −, /, *

↳ 2000

Types :-

↳ Three address instruction
↳ Two " "
↳ One " "
↳ Zero " "

Q1.) (A+B) * (C+D)

→ Three address
ADD R_1, A, B ; $R_1 \leftarrow M[A] + M[B]$
ADD R_2, C, D ; $R_2 \leftarrow M[C] + M[D]$
MUL X, R_1, R_2 ; $M[X] \leftarrow R_1 * R_2$
↳ Register

→ Two address
MOV R_1, A ; $R_1 \leftarrow M[A]$
ADD R_1, B ; $R_1 \leftarrow R_1 + M[B]$
MOV R_2, C ; $R_2 \leftarrow M[C]$
ADD R_2, D ; $R_2 \leftarrow R_2 + M[C]$
MUL R_1, R_2 ; $R_1 \leftarrow R_1 * R_2$
MOV X, R_1 ; $X[M] \leftarrow R_1$

∴ AC → accumulator

Date _/_/_

→ One Address

LOAD A ; AC ← M[A]
ADD B ; AC ← AC + M[B]
STORE T ; M[T] ← AC → A+B
LOAD C ; AC ← M[C]
ADD D ; AC ← AC + M[D] → C+D
MUL T ; AC ← AC * M[T] → (A+B) * (C+D)
(C+D) (A+B)

STOREX ; M[X] ← AC

→ Zero Address. TOS → Top of stack

PUSH A , TOS ← M[A]
PUSH B , TOS ← M[B]
ADD , TOS ← M[A] + M[B]
PUSH C , TOS ← M[C]
PUSH D , TOS ← M[D]
ADD , TOS ← M[C] + M[D]
MUL , TOS ← (M[A] + M[B]) * [M[C] + M[D]]
POP X , M[X] ← TOS

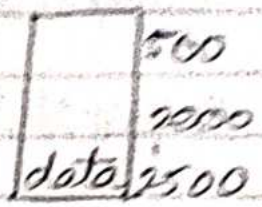
★ Addressing Modes

(i) Direct Addressing Mode
(ii) Indirect Addressing Mode
(iii) Register Direct " "
iv) Register Indirect " "
v) Immediate Addressing "
vi) Implicit " "
vii) Indexed " "
viii) Relative " "
ix) Auto increment " "
x) Auto decreement A.M

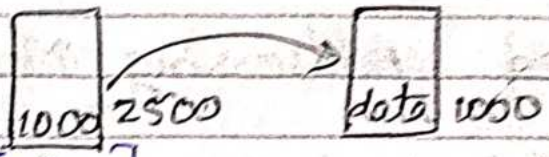
i) Direct AM

We can get the data directly by going on that address.

LOAD R_1, 2500
Effective Address = 2500



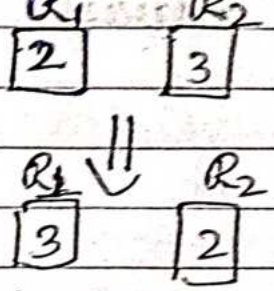
ii) Indirect AM
Move to the given address, then move to the address which is stored as data in the primary address.



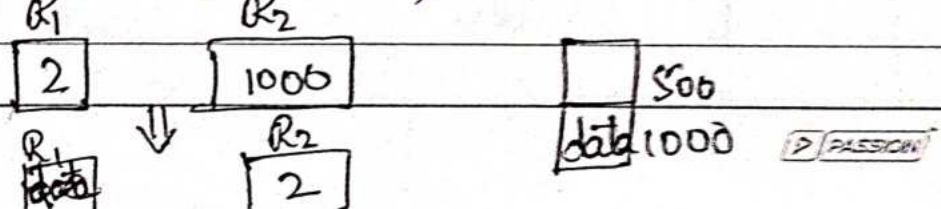
LOAD R_1, [2500]
Effective address → 1000

iii) Register Direct AM

Data moves from 2nd to 1st Register
MOV R_1, R_2



iv) Register Indirect AM
MOV R_1, [R_2]
Here R_2 consist of address, not data.



Date _/_/_

(vi) Immediate A.M.

Immediately data is given in instructions

LOAD R1, 05 → data

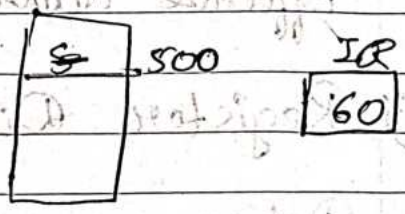
vi) Implicit / Implied A.M.

Operand of the instruction is specified in the opcode itself.

→ CMA → Complement the accumulator
→ Ashr

vii) Indexed Addressing Mode.

EA = [Base Register + Indexed Register]



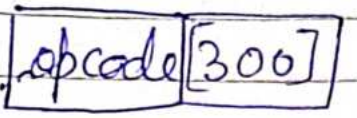
viii) Relative Address Mode

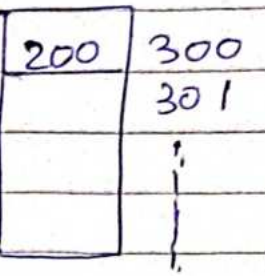
EA = PC + offset ; PC = program counter

EA = 500 + 300
EA = 501 + 299

Date _/_/_

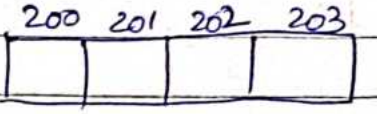
ix) Auto increement / Auto decreement AM.





↙LOAD R1 +

Increement



↘LOAD R2 -

Decreement

Q.1 An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R_1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is -

1) Direct
2) Immediate
3) Relative
4) Register Indirect
5) Indexed with R_1 as the index Register.

Sol. 1) Direct ⇒ EA = 400

2) Immediate ⇒ EA = 301

3) Relative:

EA = PC + offset

= 302 + 400

= 702