Date 09/Aug/19

Unit - 2

Arithmetic & Logic Unit

Carry Look Ahead Adder :-

Delay in Propagation of carry in 4-bit binary adder. So, fast adder is required called as carry look ahead adder. In this adder, we calculate the carry of each stage in advance.

FULL ADDER -

$$\text{SUM } (S_i) = A_i \oplus B_i \oplus C_i \quad \text{— (1)}$$

$$\text{Carry } (C_{i+1}) = \underbrace{(A_i \oplus B_i)}_{P_i \text{ (carry Propagator)}} C_i + \underbrace{A_i B_i}_{G_i \text{ (carry generator)}} \quad \text{— (2)}$$

Let us assume,

$$A_i \oplus B_i = P_i.$$

$$A_i B_i = G_i.$$

$$\text{Sum } (S_i) = P_i \oplus C_i \quad \text{— (3)}$$

$$\text{Carry } (C_{i+1}) = P_i \cdot C_i + G_i \quad \text{— (4)}$$

At, $i = 0$

$$S_0 = P_0 \oplus C_0 \quad \text{— (5)}$$

$$C_1 = P_0 \cdot C_0 + G_0 \quad \text{— (6)}$$

* C_0 is alway zero.
$C_0 = 0$.

Scanned with CamScanner

Date

At, $i=1$

$$S_1 = P_1 \oplus C_1 \quad \text{--- (7)}$$
$$C_2 = P_1 \cdot C_1 + G_1 \quad \text{--- (8)}$$

Putting (6) in (8).

$$C_2 = P_1 [P_0 \cdot C_0 + G_0] + G_1$$
$$C_2 = P_1 P_0 C_0 + P_1 G_0 + G_1 \quad \text{--- (9)}$$

At, $i=2$

$$S_2 = P_2 \oplus C_2 \quad \text{--- (10)}$$
$$C_3 = P_2 \cdot C_2 + G_2 \quad \text{--- (11)}$$

Putting (9) in (11)

$$C_3 = P_2 \cdot [P_1 P_0 C_0 + P_1 G_0 + G_1] + G_2.$$
$$C_3 = P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2 \quad \text{--- (12)}$$

At, $i=3$

$$S_3 = P_3 \oplus C_3 \quad \text{--- (13)}$$
$$C_4 = P_3 \cdot C_3 + G_3 \quad \text{--- (14)}$$

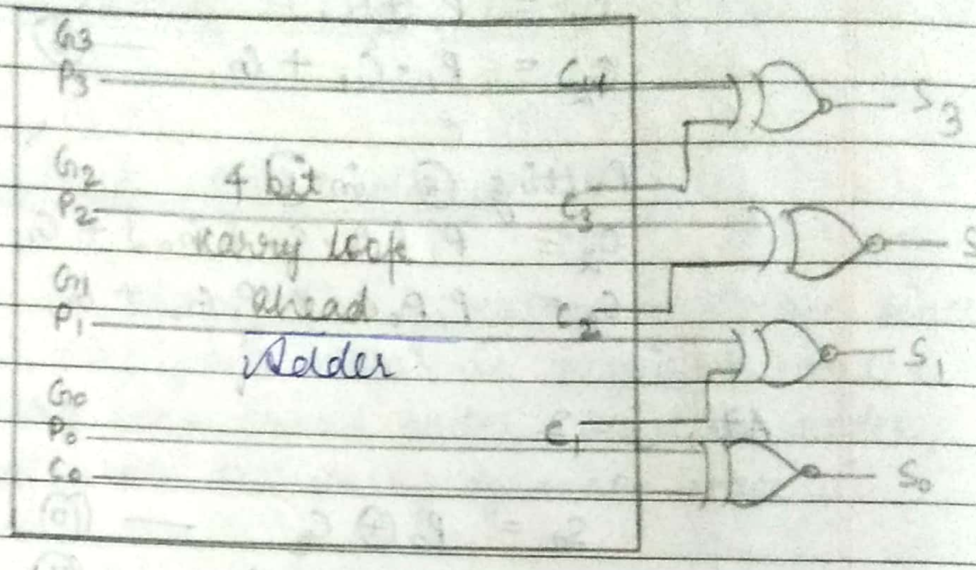
Putting (12) in (14).

$$C_4 = P_3 [P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2] + G_3$$
$$C_4 = P_3 P_2 P_1 P_0 C_0 + P_3 P_2 P_1 G_0 + P_3 P_2 G_1 + P_3 G_2 + G_3 \quad \text{--- (15)}$$

* $C_1, C_2, C_3, C_4 \ldots$ are called carry Generator.

1. It is used to eliminate delay in carry propagation in Binary adder.
2. In Binary adder each stage depend upon carry generated by previous stage.
3. It create lots of delay if we want to add two large binary no.
4. So, in this we calculate carry ~~for~~ each stage in advance in order to avoid delay.

fig. carry look ahead adder.



Imp.

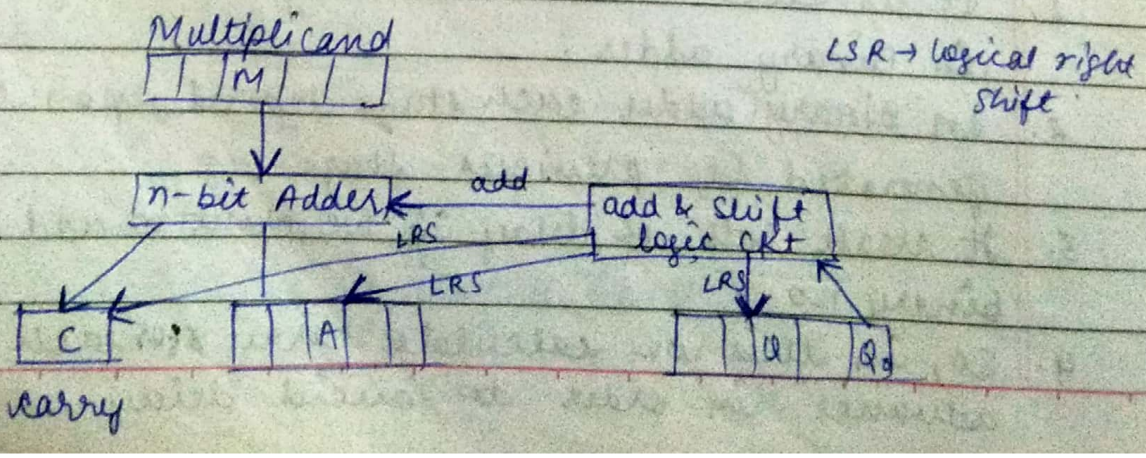
~~Booth Algorithm~~

Multiplication of Unsigned numbers :-

e.g. 11 × 13.

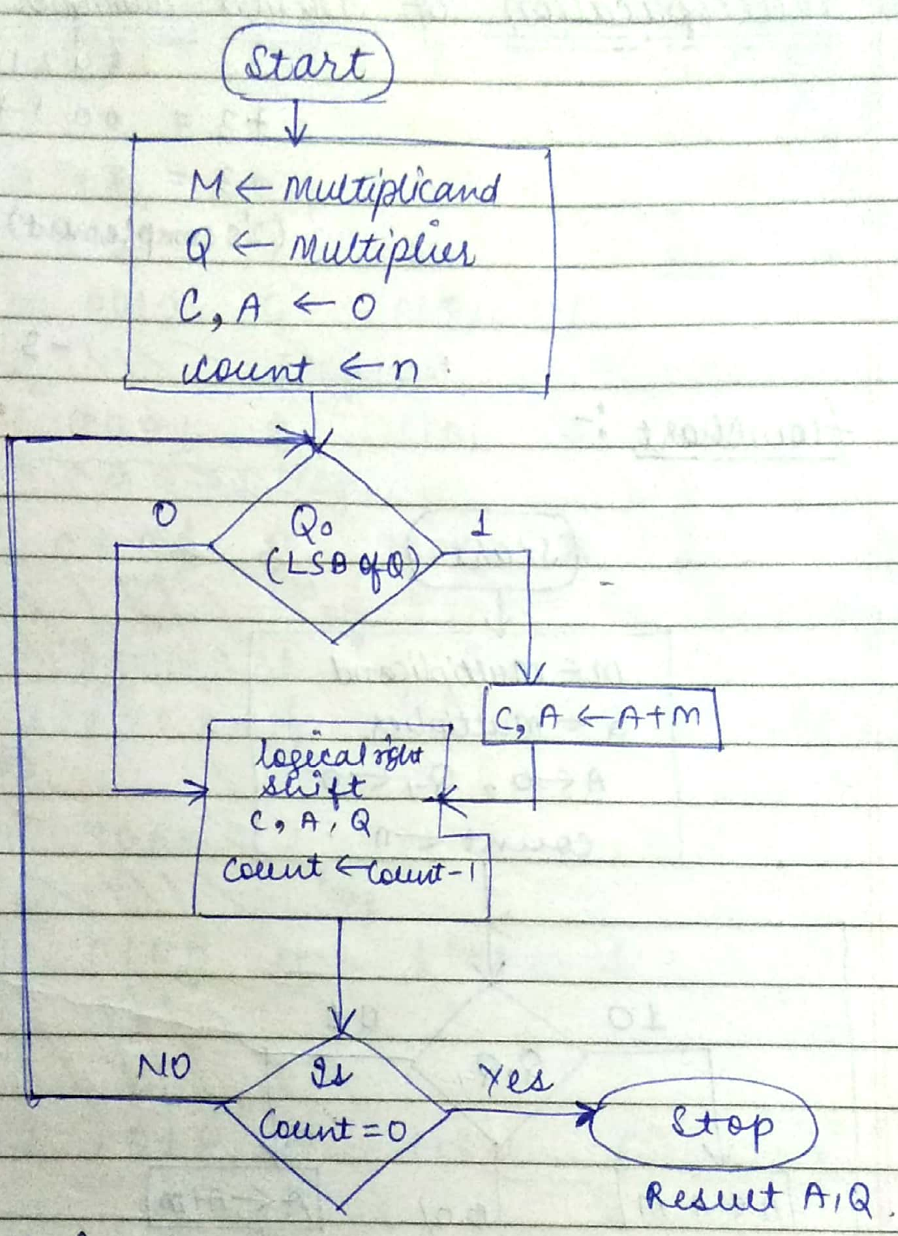
```
      1 0 1 1   → multiplicand (M)
    × 1 1 0 1   → Multiplier (Q0)
    ---------
      1 0 1 1
    0 0 0 0 ×
  0 1 0 1 1 × ×
+ 1 0 1 1 × × ×
-----------------
AQ = 1 0 0 0 1 1 1 1
```

Hardware Diagram :-



count = no. of bits in multiplier

Flowchart :-



Example :

	C	A	Q	M	n (count)
Initially	0	0	1101	1011	4
1st cycle (A+M)	0	1011	1101	1011	
	0	0101	1110	1011	3
2nd cycle	0	0010	1111	1011	2
3rd cycle (A+M)	0	1101	1111	1011	
	0	0110	1111	1011	1
4th cycle (A+M)	1	0001	1111	1011	
	0	1000	1111	1011	0

neg

neg

0010 + 1011 = 1101

0110 + 1011 = 0001, C = 1

AQ = 10001111 Ans

Multiplication of Signed Number : (Booth Algorithm).

8 4 2 1

+3 = 0 0 1 1

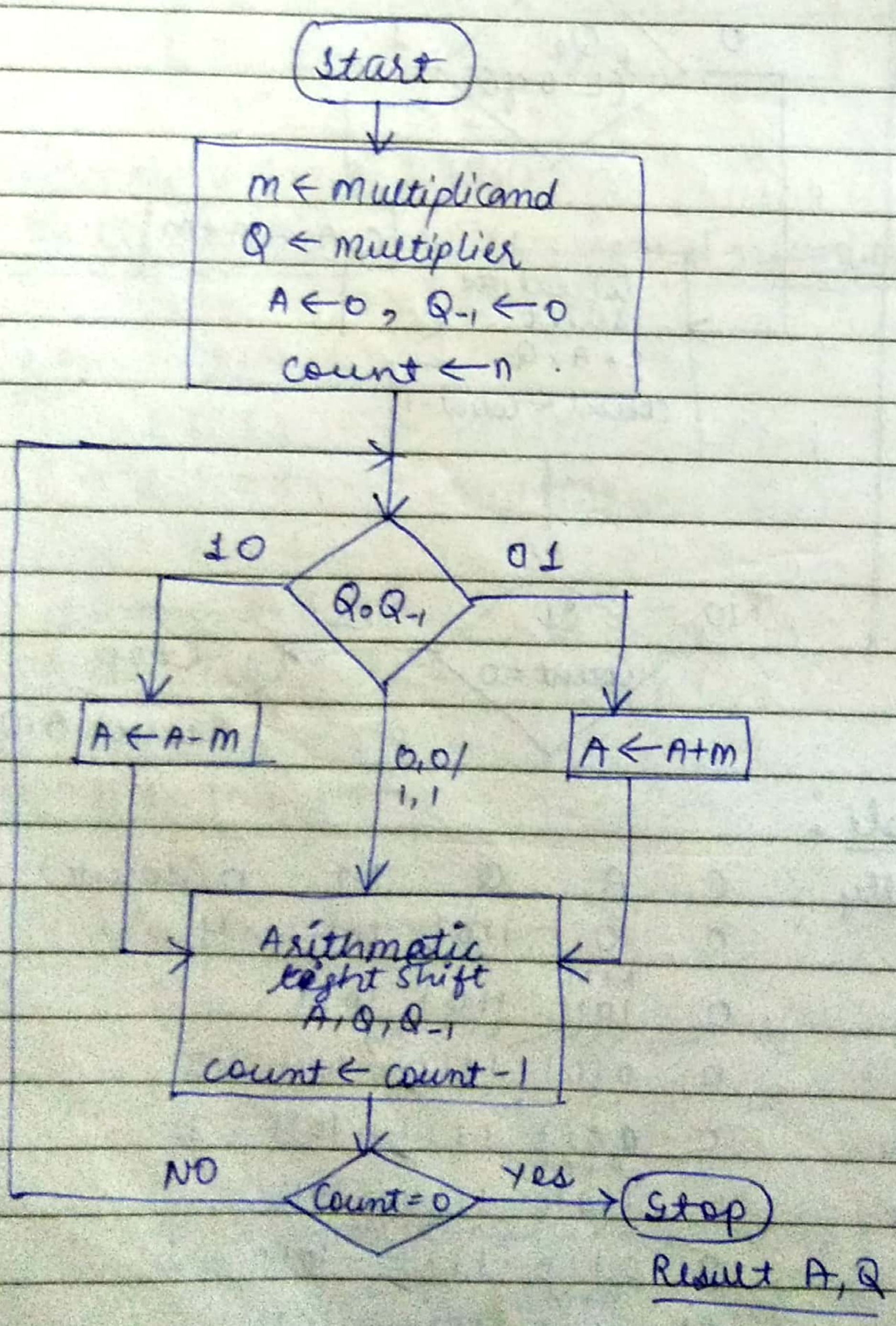
−3 = 0 0 1 1

(2's complement) 1 1 0 0 ← 1'S

\+ 1 ← 1

−3 = 1 1 0 1.

Flowchart :-



1. $-3 \times -2 = 6$.

$3 = 0011$
$-3 = 1100 + 1 = 1101$ — M

$2 = 0010$
$-2 = 1101 + 1 = 1110$ — Q

	A	Q	Q-1	M	n
Init	0	1110	0 → neg	1101	4
1st	0	0111	0	1101	3
2nd	A-M 0011	0111	0 → neg	1101	
	0001	1011	1 → neg	1101	2
3rd	0000	1101	1	1101	1
4th	0000	0110	1	1101	0

$6 = 0110$

0000
− 1101

0000
0010
+ 1
0011

AQ = 00000110. Ans.

Range formula :- (For 2's Complement)

$$= \boxed{(-2^{n-1}) \text{ to } +(2^{n-1}-1)}$$

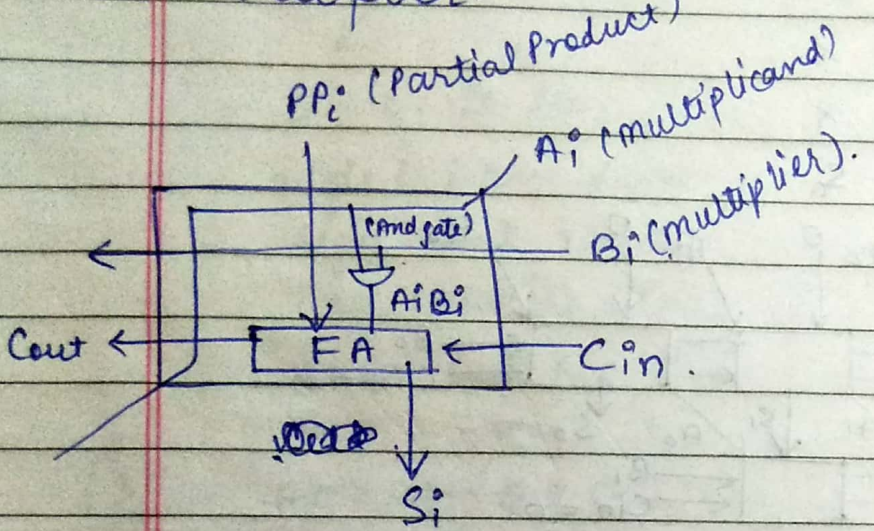
where n = no. of bits

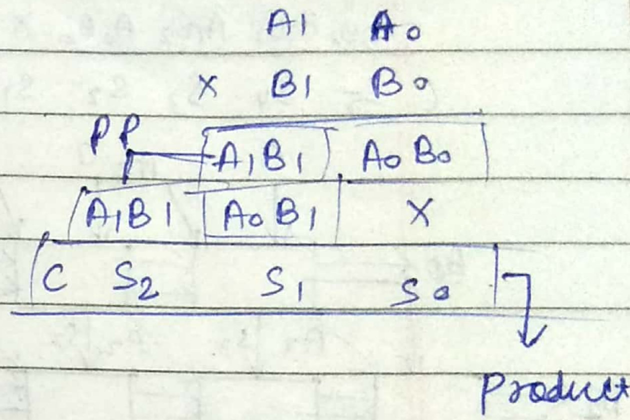
$n = 4$ bit, -8 to $+7$.
$n = 5$ bit, -16 to $+15$.
$n = 6$ bit, -32 to $+31$.

Date 19.08.19.

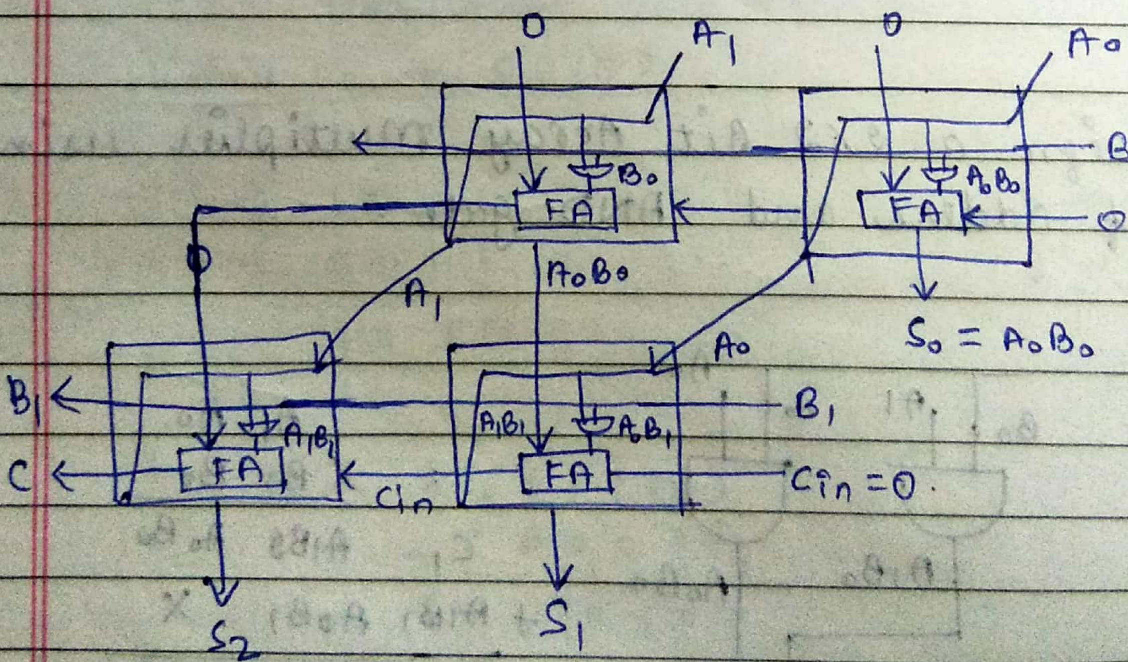
Array Multiplier :-

Arrangement of cells in the form of Array is called Array Multiplier or Binary Multiplier.





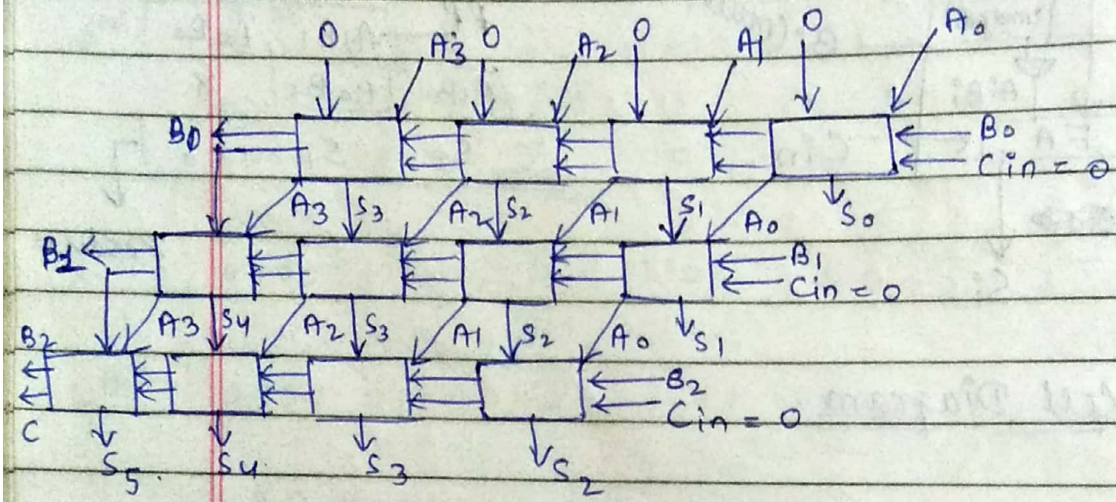
Cell Diagram.



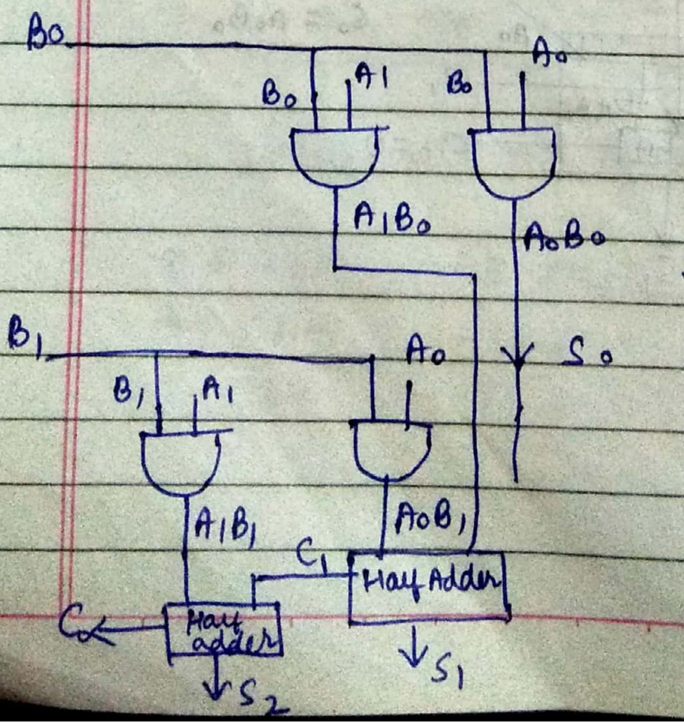
Date

4X3 Array multiplier :-

$$
\begin{array}{rrrrrrr}
 & & A_3 & A_2 & A_1 & A_0 \\
 & & \times & B_2 & B_1 & B_0 \\
\hline
 & & A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 \\
 & A_3B_1 & A_2B_1 & A_1B_1 & A_0B_1 & \times \\
A_3B_2 & A_2B_2 & A_1B_2 & A_0B_2 & \times & \times \\
\hline
C\ S_5 & S_4 & S_3 & S_2 & S_1 & S_0
\end{array}
$$



Ques. Design a 2X2 Bit Array multiplier using half adders and AND gate.



$$
\begin{array}{rrrr}
 & & A_1 & A_0 \\
 & & \times B_1 & B_0 \\
\hline
 & C_1 & A_1B_0 & A_0B_0 \\
+ & A_1B_1 & A_0B_1 & \times \\
\hline
C_0 & S_2 & S_1 & S_0
\end{array}
$$

Date

Ques. Addition & Subtraction using 2's complement.

(1)

$$\begin{array}{rr} +5 \rightarrow & 0101 \\ +3 \rightarrow & +0011 \\ \hline & 1000 \end{array}$$

If no. to be added have same sign but result has opposite sign then overflow

(2)

$$\begin{array}{rr} +5 \rightarrow & 0101 \\ +2 \rightarrow & 0010 \\ \hline & 0111 \end{array}$$

not overflow.

(3)

$$\begin{array}{rr} +5 \rightarrow & 0101 \\ -3 \rightarrow & +1101 \\ \hline & (1)\,0010 \end{array}$$

neglected

overflow.

Ans = +(0010)

$$\begin{array}{rr} 3 \Rightarrow & 0011 \\ 1' \rightarrow & 1100 \\ + & 1 \\ \hline & 1101 \end{array}$$

$$\begin{array}{rr} -5 \rightarrow & 1011 \\ +3 \rightarrow & 0011 \\ \hline & 1110 \end{array}$$

$$\begin{array}{rr} -5 \rightarrow & 0101 \\ 1' \rightarrow & 1010 \\ + & 1 \\ \hline & 1011 \end{array}$$

Ans = −(1110)

$$= -\left(\begin{array}{r} 0001 \\ + \quad 1 \\ \hline 0010 \end{array}\right)$$

8421
1110

Ans. = −(0010)

Scanned with CamScanner

Date

Rules for 2's complement :-

1. If result has carry, neglect it
check the sign bit (s)
S = 0, Result will be +ve
S = 1, Result will be −ve
−(2's complement of Result).

2. If result has no carry, check the sign bit (s)
S = 0, Result will be +ve.
S = 1, Result will be −ve
−(2's complement of Result).

3. Overflow,
If sign bit of no's are same but Result has different sign bit, then overflow occurs.

MSB or sign bit ↑ ↗ LSB
0 1 0 1

If Signed no.
Sign bit = 0, +ve
= 1, −ve.

Date

1.

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

neglected

not overflow

Solve -5, -5 using 2's complement.

4 2 1
1 0 1

2. 5 = 0101

```
-5 = 1010
       + 1
     ----
     1011
```

(-2^{n-1}) to $+(2^{n-1}-1)$

```
-5 = 1011
-5 = 1011
   -----
  1 0110
```

neglected. overflow

So answer is not in range.

```
+5 = 00101
-5 = 11010
        + 1
     -----
-5 = 11011
```

```
  11011
+ 11011
-------
1 10110
```

neglect

$-(01001 + 1) = -(01010) = -(10)$

Date

3.

$$\begin{array}{r} -5 \\ -4 \\ \hline -9 \end{array}$$

$5 = 0101$

$-5 = 1010$
$\quad\quad + 1$
$\quad\quad 1011$

$4 = 0100$

$-4 = 1011$
$\quad\quad + 1$
$\quad\quad 1100$

$$\begin{array}{r} 1011 \\ +\ 1100 \\ \hline 1\,0111 \end{array}$$

neglected

overflow.

Ques. Perform Booth multiplication for $\overset{M}{-5} \times \overset{Q}{+3}$.

	A	Q	Q_{-1}	M	n
Initially	0	0011	0	1011	4
1^{st}	A − M 0101	0011	0 → neg	1011	4
	0010	1001	1	1011	3
2^{nd}	0001	0100	1 → neg	1011	2
3^{rd}	A + M 1100	0100	1 → neg	1011	2
	1110	0010	0	1011	1
4^{th}	1111	0001	0	1011	0

$$\begin{array}{r} 0000 \\ -1011 \end{array}$$

$$\begin{array}{r} 0000 \\ 0100 \\ +\ 1 \\ \hline 0101 \end{array}$$

$$\begin{array}{r} 0001 \\ +\ 1011 \\ \hline 1100 \end{array}$$

15

8 4 2 1
1 1 1 1

$15 = 0000\,1111$
$15' = 1111\,0000$
$\quad\quad + 1$
$\quad\quad 1111\,0001$

2's compl AQ = 11110001

AQ = −(1110 001)

= −15 Ans

Date

Que. $-13 \times +11$ using Booth Algorithm

$M = -13$, $Q = +11$

$13 = 01101$ $\qquad$ $11 = 01011$

$-13 = 10010 + 1 = 10011$ $\qquad$ ~~$-11 = 10100 + 1 = 10101$~~

	8	4	2	1
13	01	1	0	1
11	01	0	1	1

	A	Q	Q_{-1}	M	n
Initially	0	01011	0	10011	5
1st (A−M)	01101	01011	0	10011	5
→ ASR	00110	10101	1	10011	4
2nd → ASR	00011	01010	1	10011	3
3rd (A+M)	10110	01010	1	10011	3
→ ASR	11011	00101	0	10011	2
4th (A−M)	01000	00101	0	10011	2
→ ASR	00100	00010	1	10011	1
5th (A+M)	10111	00010	1	10011	1
→ ASR	11011	10001	0	10011	0

1101110001

$-(1101110001)$

$-(143)$ Ans

Side work:

10 / 01 / 11: $A \leftarrow A - M$, $A \leftarrow A + M$, ASR A, Q, Q_{-1}, $C \leftarrow C - 1$

$01100 + 1 = 01101$

$00011 + 10011 = 10110$

11011: $01100 + 1 = 01000$

$00100 + 10011 = 10111$

$13 \times 11 = 143$; $143 - 128 = 15$

	128	64	32	16	8	4	2	1
143 → 001	0	0	0	1	1	1	1	1
1' → 110	1	1	1	0	0	0	0	0
+								1
1101	1	1	0	0	0	1		

Because 2's complement has unique representation of zero.

Date

(For 1's complement)

Range Formula :-

$$-(2^{n-1}-1) \text{ to } +(2^{n-1}-1).$$

where, n = no. of bits required.

n = 4, −7 to +7.

n = 5, −15 to +15.

Ques. 1 +12 → 01100

Sol −12 → 10011 (1's complement)

~~10100~~

Ques. 2 +21 → 010101

Sol. −21 → 101010 (1's complement)

32 16 8 4 2 1
0 1 0 1 0 1

2^5-1 to 2^5-1

~~−31 to +31~~

Ques. using 1's complement :

(a) −5 + 3

5 = 0101

−5 = 1010

+3 = 0011

(1's)

1010
+ 0011
1101

Ans ⇒ −(0010)

−2 Ans

(2's)

−5+1 = 1010 + 1 = 1011

−5 1011
+3 + 0011
1110

Ans −(1110)

Ans −(0001) + 1

Ans −(0010)

−(2)

(b) +5 − 3

+5 = 0101

+3 = 0011

−3 = 1100

0101
+ 1100
(1)0001

10001
+ 1
10010

Ans = (0010)

Date

Rule for 1's Complement Arithmetic :-

1. If carry, add again to result, then check the sign bit
   Sign bit → 0, Result will be +ve

   Sign bit → 1, Result
   = −(1's of Result).

2. If no carry then check the sign bit.
   Sign bit → 0, Result will be +ve.
   Sign bit → 1, Result
   = −(1's of Result).

Que (c) −5 −3.

5 = 00101
−5 = 11010
3 = 00011
−3 = 11100.

$$\begin{array}{r} -5 = 11010 \\ -3 = +11100 \\ \hline 110110 \end{array}$$

$$\begin{array}{r} \cancel{1}10110 \\ +\quad 1 \\ \hline \cancel{1}10111 \end{array}$$

8 4 2 1
−($\cancel{1}$01000)
−8 Ans

Date

(d) $+5+3$.

$+5 = 00101$

$+3 = 00011$

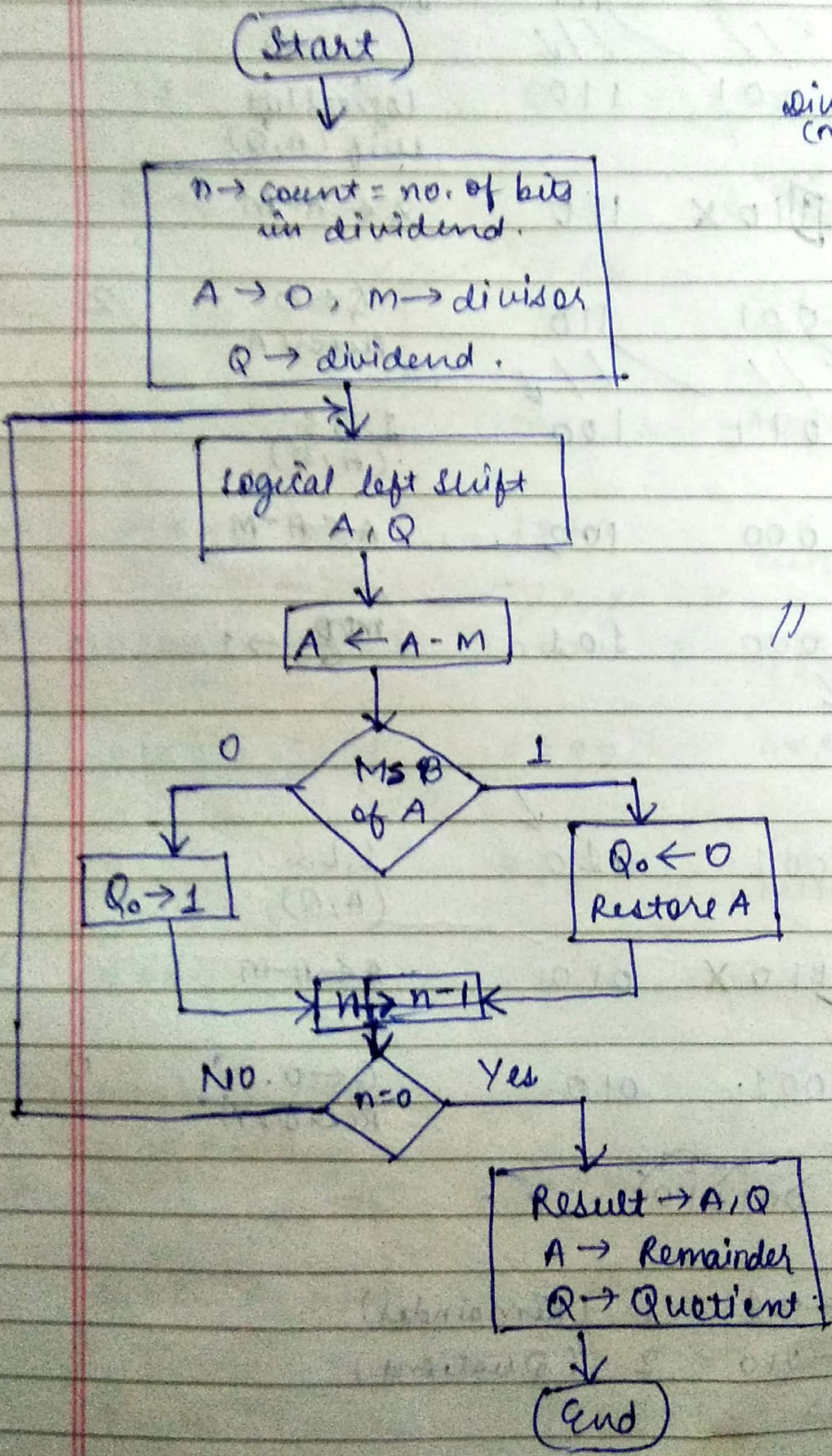
$$\begin{array}{r} 00101 \\ +\ 00011 \\ \hline 01000 \\ \hline {\scriptstyle 8421} \end{array}$$

Ans $= 01000$.

$= 8$ Ans

Restoring division Method for unsigned numbers:

2 → Quotient
3) 7 → dividend (Q)
Divisor (M)
− 6
1 → Remainder (A)



// $A \leftarrow A + M$ (signed)

Date

M Q

$4 \overline{) 12 } (3 - Q$

-12

$0 \rightarrow R$

$M = 4 = 0100$ (8421)

$Q = 12 = 1100$

	M	A	Q	operation	Count (n)
	0100	0000	1100	Initial	4
1st		0001	1000	LLS (AQ)	
		1101	1000	A←A−M	
0001 + 1011 + 1 = 1101	0100	0001	1000	Q←0, Restore A.	3
2nd	0100	0011	0000	LLS (AQ)	
	0100	1111	0000	A←A−M	
0011 + 1011 + 1 = 1111	0100	0011	0000	Q←0, Restore A	2
3rd	0100	0110	0000	LLS (AQ)	
	0100	0010	0000	A←A−M	
0110 + 1011 + 1 = 0010	0100	0010	0001	Q←1	1
4th	0100	0100	0010	LLS (AQ)	
				~~A←A−M~~	
0100 + 1011 + 1 = 0000	0100	0000	0010	A←A−M	
	0100	0000	0011	Q←1.	0

A → 0 (Rem)

$Q \rightarrow (0011)_{10} \Rightarrow 3$ (Quot)

Scanned with CamScanner

Date

$M \leftarrow 3 \overline{)\,7\,(} 2 \rightarrow Q$
$\underline{6}$
$1 \rightarrow$

	M (3)	A	Q (7)	operation	Count (n)
	011	000	111	initial	3
1st	011	001	110	logical left shift (A, Q)	
A = 001, M = 011, M' = 100, +1, 101	011	$\boxed{1}$10 X	110	A ← A − M	
	011	001	110	Q ← 0, Restore A	2
2nd: 001, 101, 110	011	011	100	l.l.s (A, Q)	
	011	000	100	A ← A − M	
011, 100, +1, 000	011	000	101	~~MSB~~ Q_0 → 1.	1
3rd	011	001	010	l.l.s. (A, Q)	
	011	110 X	010	A ← A − M	
001, 101, 110	011	001	010	Q ← 0, Restore A	0

~~AQ~~ = 001 010 1

Result : A ← 001 = 1 (Remainder)
Q ← 010 = 2 (Quotient)

Scanned with CamScanner

Date

Ques. Find the no. of address lines and input output data lines for the following memory capacity.

(i) 2K × 16

(ii) 64 K × 8

(iii) 1M × 8

(iv) 16 × 16.

(v) 16 KB

Add lines.

(i) $2K = 2 \times 2^{10} = 11$

Data Lines / I/O data bits.

16 bits

(ii) Add Lines.

$64K = 2^6 \cdot 2^{10} = 16$ Lines.

Data Lines

8 bits

(iii) Add Lines

$1M = 1 \times 2^{20} = 20$

Data Lines

8 bits

(iv) Add lines

$= 1 \times 2^{30}$

$= 30$

Data Lines

16 bits

(v) 16 KB

$= 16K \times 8$

Add $= 2^4 \times 2^{10}$.

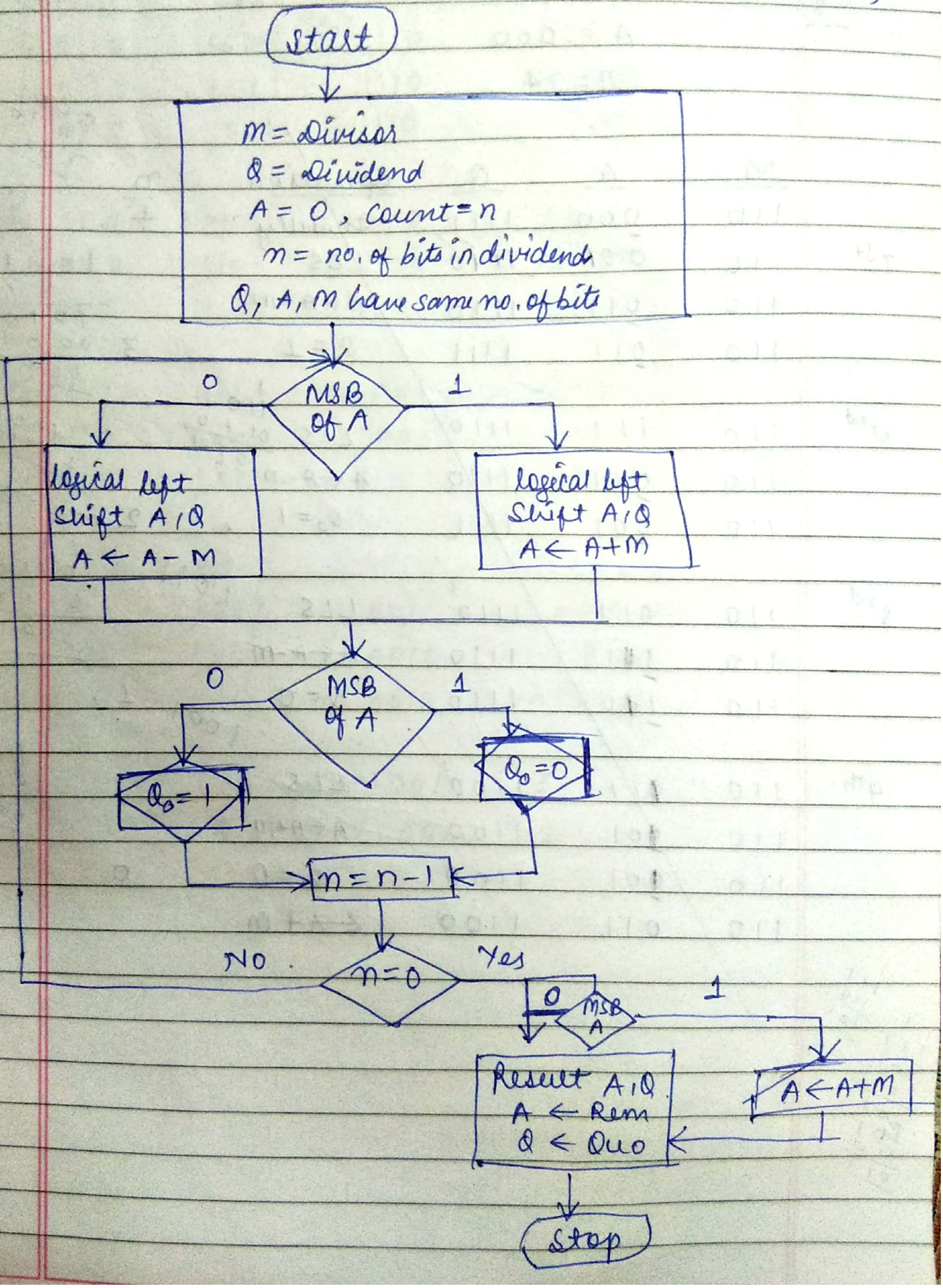
$= 14$ lines

Data Lines = 8.

Scanned with CamScanner

Date ______

Non-Restoring Division method (for Unsigned No.)



$$\begin{array}{r} 6\overline{)15}(2 \\ \underline{12} \\ 3 \end{array}$$

Date

Ques. 15 ÷ 6 by non-Restoring.

	M	A	Q	operation	n
	0110	0000	1111	Initially.	4
1.	0110	0001	1110	LLS	
	0110	1011	1110	A ← A−M	
	0110	1011	1110	$Q_0=0$	3
2.	0110	0111	1100	LLS	
	0110	1101	1100	A ← A+M	
	0110	1101	1100	$Q_0=0$	2.
3.	0110	1011	1000	LLS	
	0110	0001	1000	A ← A+M	
	0110	0001	1001	$Q_0=1$	1.
4.	0110	0011	0010	LLS	
	0110	1101	0010	A ← A−M	
	0110	1101	0010	$Q_0=0$	0.
	0110	0011	0010	A ← A+M	

A = R = 0011 = 3

Q = Q = 0010 = 2.

AKTU

$3 \overline{)15} (5$
-15
$\overline{\times}$

Ques. Perform the division process of $\underset{15}{0000\underline{1111}}$ by $\underset{3}{\underline{0011}}$ (use a dividend of 8 bits)

	M	A	Q	operation	n
	0011	0000	1111	Initially	4
1.	0011	0001	1110	LLS	
	0011	1110	1110	A ← A-M	
	0011	1110	1110	$Q_0 = 0$	3
2.	0011	1101	1100	LLS	
	0011	0000	1100	A ← A+M	
	0011	0000	1101	$Q_0 = 1$	2
3.	0011	0001	1010	LLS	
	0011	1110	1010	A ← A-M	
	0011	1110	1010	$Q_0 = 0$	1
4.	0011	1101	0100	LLS	
	0011	0000	0100	A ← A+M	
	0011	0000	010~~0~~	$Q_0 = 1$	

A = 0000 ⇒ R = 0

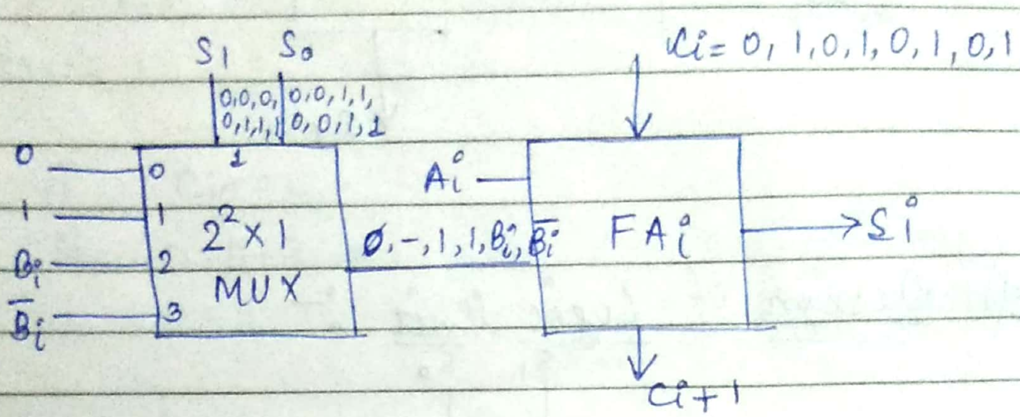
Q = 0101 ⇒ Q = 5

Date 18/09/19

Design of ALU (Arithmetic Logic Shift) Unit

(i) Design of Arithmetic Circuit :

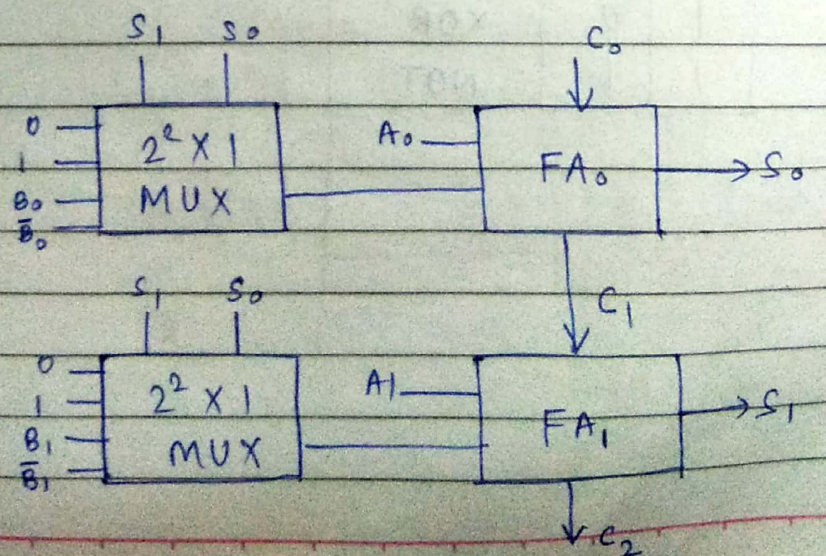
1×1



S_1	S_0	C_i	operation
0	0	0	A_i (Transfer)
0	0	1	A_i+1 (Increment)
0	1	0	A_i-1 (Decrement)
0	1	1	$A_i-\cancel{1}+\cancel{1}$ (Transfer)
1	0	0	A_i+B_i (Addition)
1	0	1	A_i+B_i+1 (Addition with carry)
1	1	0	$A_i+\overline{B_i}$ (Subtraction with Borrow)
1	1	1	$A_i+\overline{B_i}+1$ (Subtraction)

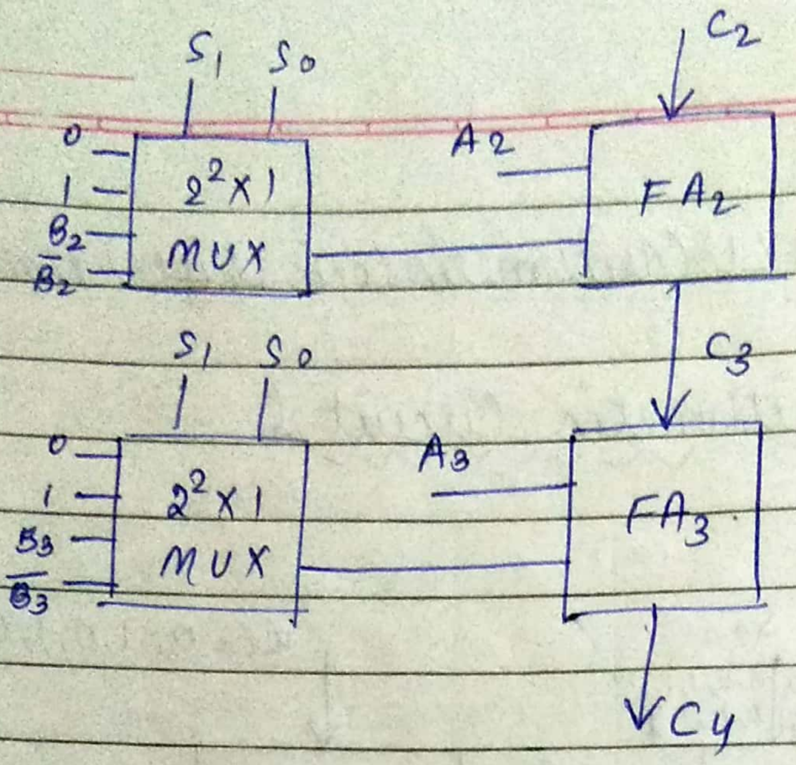
Diagram

4×4

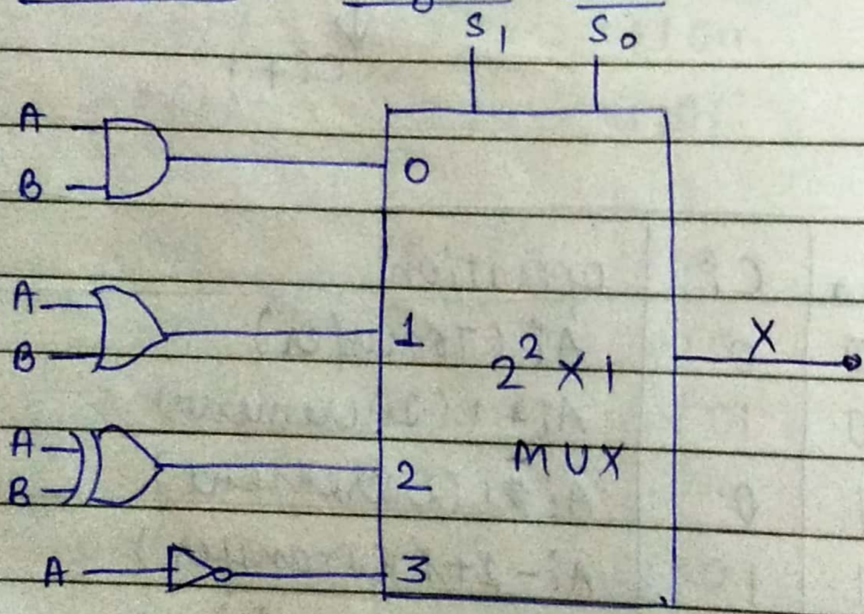


Scanned with CamScanner

Date



iii) # Design of Logic Unit :-



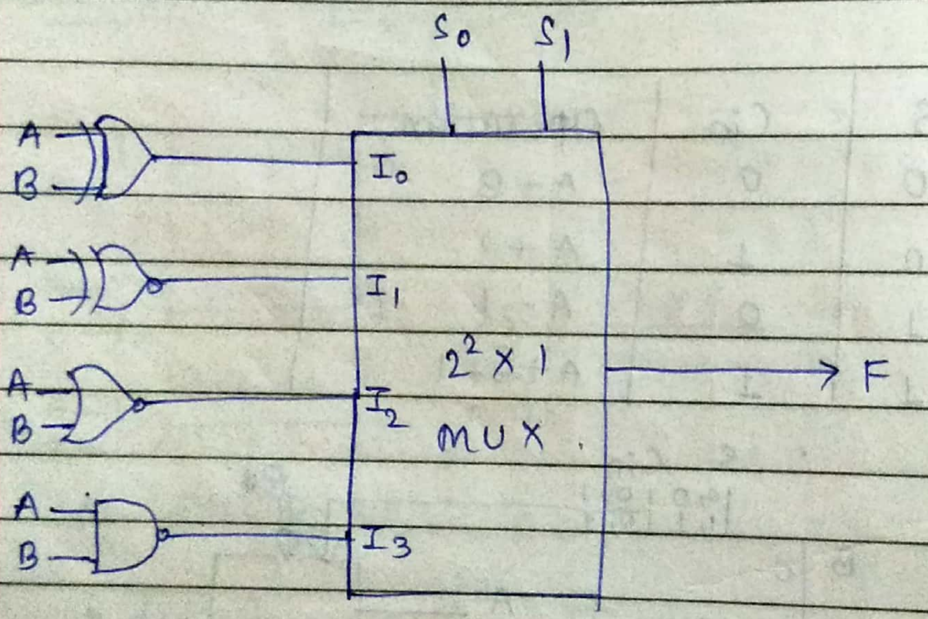
S_1	S_0	Operation (X)
0	0	AND
0	1	OR
1	0	XOR
1	1	NOT

Scanned with CamScanner

Date

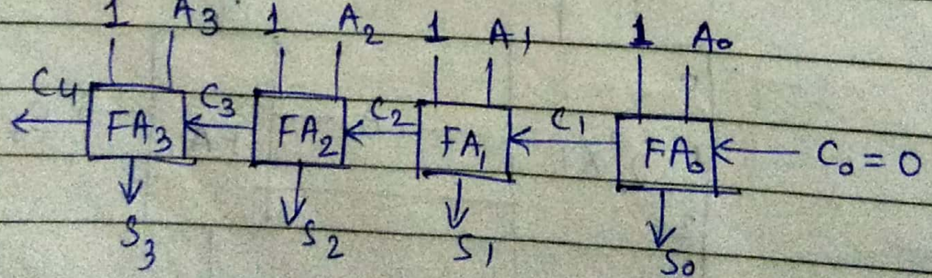
Ques. 2 Design a digital circuit that perform the logic operation XOR, XNOR, NOR, NAND. Use two selection variables ~~mange~~. Show the logic diagram of 1 stage.

S_1	S_0		
0	0	I_0	$S_1' S_0' I_0$
0	1	I_1	$S_1' S_0 I_1$
1	0	I_2	$S_1 S_0' I_2$
1	1	I_3	$S_1 S_0 I_3$.



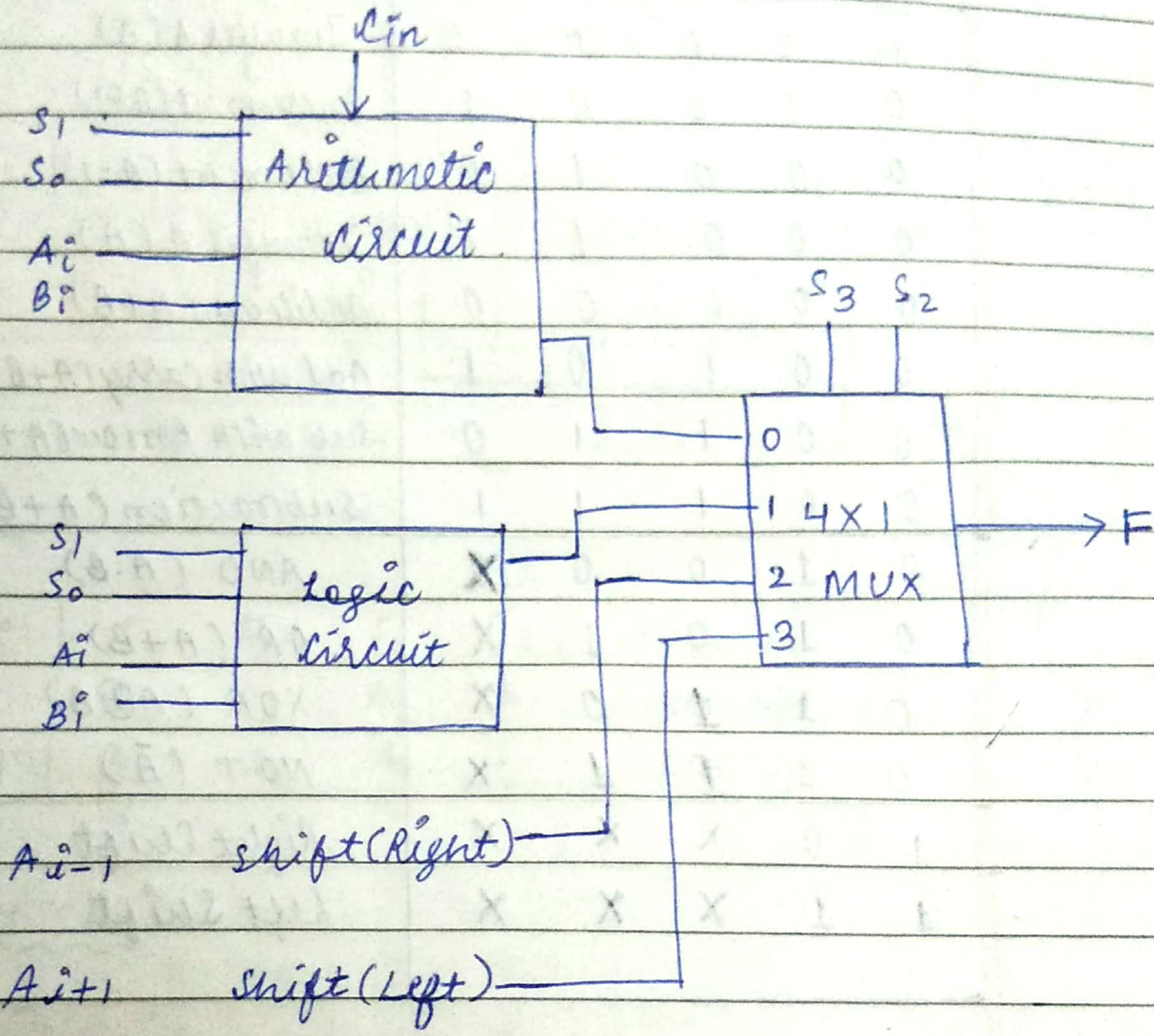
Ques. 3. Design a 4-bit decremental circuit using block diagram. $A - 1 = A + 1111$

Sol.



Date 19/9/19.

Design of ALU :-



Truth Table for MUX.

S_3	S_2	Operation
0	0	Arithmetic
0	1	Logic
1	0	Right Shift
1	1	Left Shift

Date

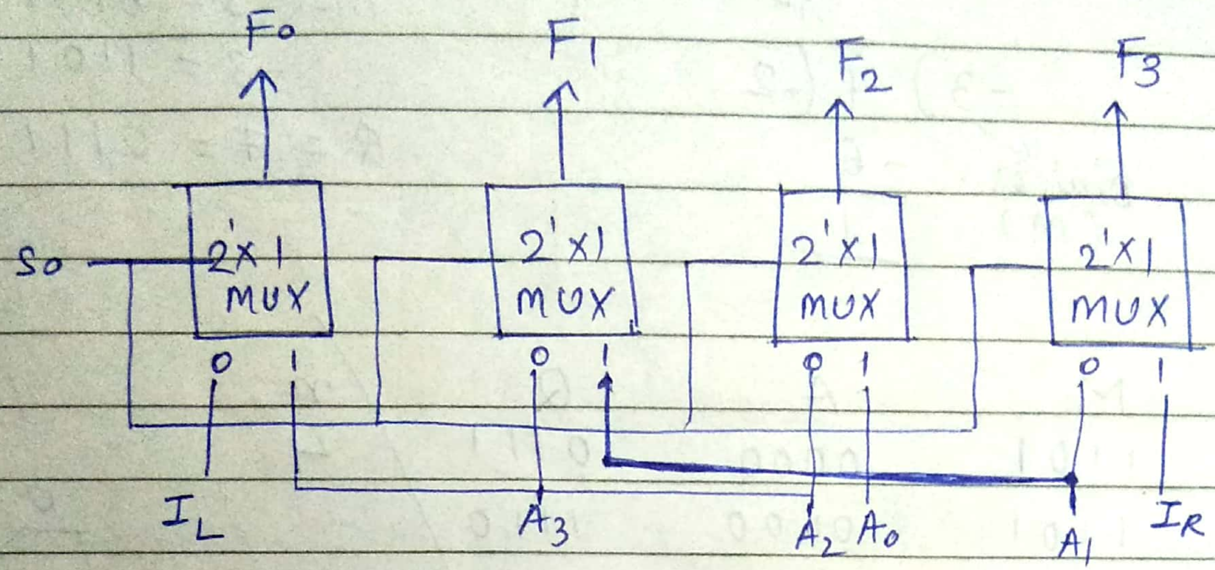
X → either 0 or 1

Truth Table for ALU Circuit:

16	8	4	2	1	
S_3	S_2	S_1	S_0	C_{in}	F
0	0	0	0	0	Transfer A (A)
0	0	0	0	1	Increment (A+1)
0	0	0	1	0	Decrement (A−1)
0	0	0	1	1	Transfer A (A)
0	0	1	0	0	Addition (A+B)
0	0	1	0	1	Add with carry (A+B+1)
0	0	1	1	0	Sub with Borrow ($A+\bar{B}$)
0	0	1	1	1	Subtraction ($A+\bar{B}+1$)
0	1	0	0	X	AND (A.B)
0	1	0	1	X	OR (A+B)
0	1	1	0	X	XOR ($A \oplus B$)
0	1	1	1	X	NOT ($\bar{A}$)
1	0	X	X	X	Right Shift
1	1	X	X	X	Left Shift

Date

(iii) Design of Shift circuit :



S	F_0	F_1	F_2	F_3	Operation
0	I_L	A_3	A_2	A_1	Right Shift
1	A_2	A_1	A_0	I_R	Left Shift.

Right shift :-

I_L F_0 F_1 F_2 F_3

A_3 A_2 A_1 A_0

I_L A_3 A_2 A_1

Left ~~Right~~ Shift :-

A_3 A_2 A_1 A_0 I_R

A_2 A_1 A_0 I_R

Ques. 7/(-3) by Restoring division.

Date

M	A	Q	n
1101	0000	0111	4
1101	0000	1110	LLS
1101	0001	1110	A ← A+M
1101	0000	1110	3
1101	0000	1110	~~LLS~~
1101	0001	1100	LLS
1101	1110	1100	A ← A+M
1101	1110	1100	$Q_0 \leftarrow 0$
1101	0001	1100	Restore A
1101	0001	1100	2
1101	0011	1000	LLS
1101	0000	1000	A ← A+M
1101	0000	1001	$Q_0 \leftarrow 1$, 1
1101	0001	0010	LLS
1101	1110	0010	A ← A+M
1101	1110	0010	$Q_0 \leftarrow 0$
1101	0001	0010	0

A = 0001 = 1

Q = 0010 = 2.

0011

0011

1101
0001
1110

1101
0011
0000

1101
0001
1110

Date 25/9/19

Floating Point Representation :-

Position of pt. is fluctuating either in decimal or binary

To represent floating point number

$$\boxed{\pm m \times r^{\pm e}}$$

where, m = mantissa / significand
↓
obsolete

r = Radix (base)

$r = 10$ for Decimal no.

$r = 2$ for Binary.

e = exponent (Power).

For example, 1. Decimal.

235.000

In floating pt. → 23.5×10

→ 2.35×10^2

→ ~~235~~ 23500×10^{-2}

2. Binary.

1101.00

In floating pt. → 110.1×2^1

→ 11.01×2^2

→ 11010×2^{-1}

Normalization of floating pt. no.:-

'1' should be placed as integer before point. Then only it is called normalization of floating pt. no.

Date

e.g. 0.26154×10^{2}

Ques. Convert into normalized form :

→ 0.001011

$\boxed{1.011 \times 2^{-3}}$

IEEE 754 floating point Representation :

1. Single Precision (32 bits) floating pt. standard.

$\pm e + (2^{8-1} - 1)$

$\pm e + 127$

Sign bit	Biased exponent	Normalised Mantissa
1 bit	8 bits	23 bits

2. Double Precision (64 bits) floating pt. Standard

$\pm e + (2^{11-1} - 1)$

$\pm e + 1023$

Sign bit	Biased Exponent	Normalised Mantissa
1 bit	11 bits	52 bits

where,

Biased exponent = every no. must be represent in +ve ~~ex~~ exponent form.

Sign bit = Sign of mantissa

Date

$1 \rightarrow -ve$

128	64	32	16	8	4	2	1
1					1	0	1

Ques.

~~-110.1×10~~

-110.1×2^2

$\Rightarrow -1.101 \times 2^4$

101 ↓ only save

Single precision :-

131

1	+4+127	101 00000000000000000000
1 bit	8 bits	23 bits.

1	10000011	10100000000000000000000
1 bit	8 bit	23 bits.

Double Precision :-

1	+4+1023	1010000- - - -20
1 bit	11 bits	52 bits.

1	10000000011	1010 - - 48 times 0
1 bit	11 bits	52 bits

Date

Ques. Represent -5.2 using single precision floating point standard.

$-5.2 \Rightarrow -101.0011 \Rightarrow -1.010011 \times 2^6$

Single Precision :

133

1	+6+127	0100110 0000- - -17
1 bit	8 bits	23 bits

1	10000101	01001100- - -17
1 bit	8 bits	23 bits.

Ques.1 Represent -11.5 using IEEE 32 bit single precision format.

Ques.2. Represent 384 using 64 bit floating point representation.

Ques.3 What are the four essential component of a no. in floating point notation.

Ques.4 Represent 0.1825 using 32 bit floating pt. format

Solution:

Sol.1

2	11	1
2	5	1
2	2	0
	1	

$0.5 \times 2 = 1.0$ 1

$0 \times 2 = 0$ 0

$-11.5 = 1011.10$

sign	biased exponent	normalised mantissa
1 bit	8 bit	23 bits

-1.01110×2^3

1 | +e + 127 | 01110000 ...18

1	+3 + 127	01110000 ... 18

1	10000010	01110000 ... 18
1 bit	8 bits	23 bit.

Sol. 2. $384 = 110000000.00$

1.10000000×2^{8}

0	+8+1023	1000000000 44
1 bit	11 bit	52 bits.

0	10000000111	1000000000 ... 44
1 bit.	11 bits.	52 bits.

Sol. 3.
1. Sign.
2. Mantissa / Significand.
3. exponent
4. Radix.

Sol. 4. 0.1825

0.1825×2	=	0.365	0
0.365×2	=	0.73	0
0.73×2	=	1.46	1
0.46×2	=	0.92	0
0.92×2	=	1.84	1

0.00101

$\Rightarrow 1.01 \times 2^{-3}$

1	-3+127	01000 21

1	01111100	01000 21
1	8 bit.	23 bit.

Date

Arithmetic Operations on floating no :-

1. Addition of floating pt. no.
2. Subtraction of F.P.N.
3. Multiplication of F.P.N
4. Division of F.P.N.

Types of Overflows & underflows in F.P.N :-

Overflow :- If result of operation is more than the maximum possible value.

Underflow :- If result of operation is less than the minimum possible value.

* Mantissa (Significand) overflow/underflow.
* Exponent overflow/underflow

0.2×10^{3}
0.03×10^{3}
0.2

Subtraction / Addition Algorithm :-

1. Check for zero.
2. Align the significand (make exponent of both no. equal) [make smaller exponent no. equal to larger exp. no.]
3. Perform Addition or Subtraction.
4. Normalize the Result.

example, $f_1 = 0.2 \times 10^{+3}$
$f_2 = 0.3 \times 10^{+2}$

Date

$f_3 = f_1 + f_2$

$f_2 = 0.03 \times 10^{+3}$

$f_1 = 0.2 \times 10^{+3}$

$f_3 = 0.23 \times 10^{+3}$

$f_3 = 2.3 \times 10^{+2}$

$\boxed{f_3 = 10.0100 \times 10^{+2}}$

$0.3 \times 2 = 0.6 = 0$

$0.6 \times 2 = 1.2 = 1$

$0.2 \times 2 = 0.4 = 0$

$0.4 \times 2 = 0.8 = 0$