UNIT-2

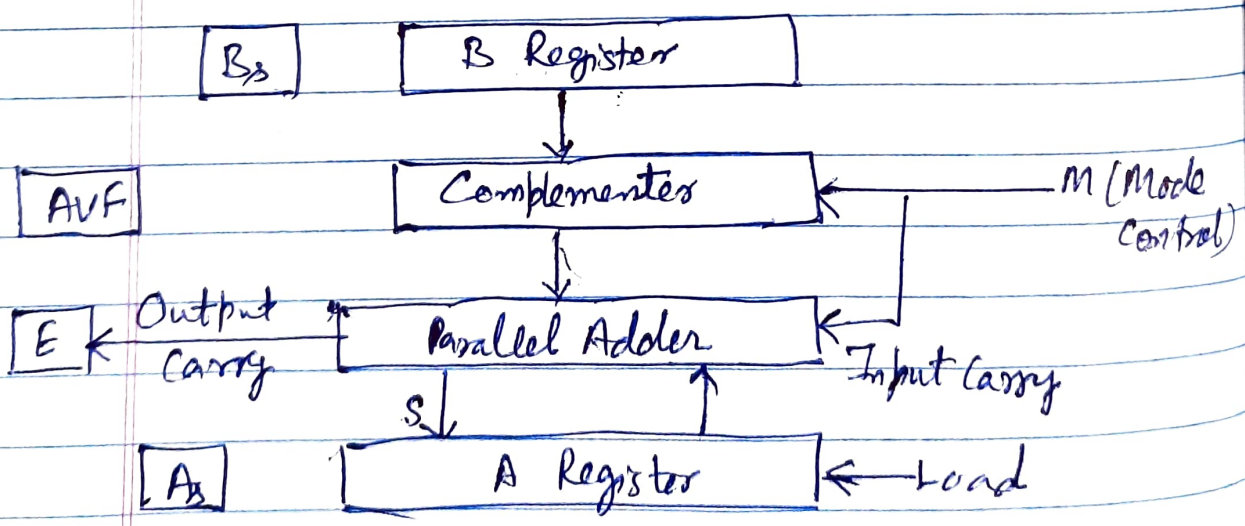
Lecture-9

Date.
Page No.

Addition & Substraction with Signed Magnitude Data

operation	Add Magnitudes	Substract Magnitudes when A>B	when A<B	when A=B
(+A)+(+B)	+(A+B)			
(+A)+(-B)		+(A-B)	-(B-A)	+(A-B)
(-A)+(+B)		-(A-B)	+(B-A)	+(A-B)
(-A)+(-B)	-(A+B)			
(+A)-(+B)		+(A-B)	-(B-A)	+(A-B)
(+A)-(-B)	+(A+B)			
(-A)-(+B)	-(A+B)			
(-A)-(-B)		-(A-B)	+(B-A)	+(A-B)

Hardware for Addition and Substraction

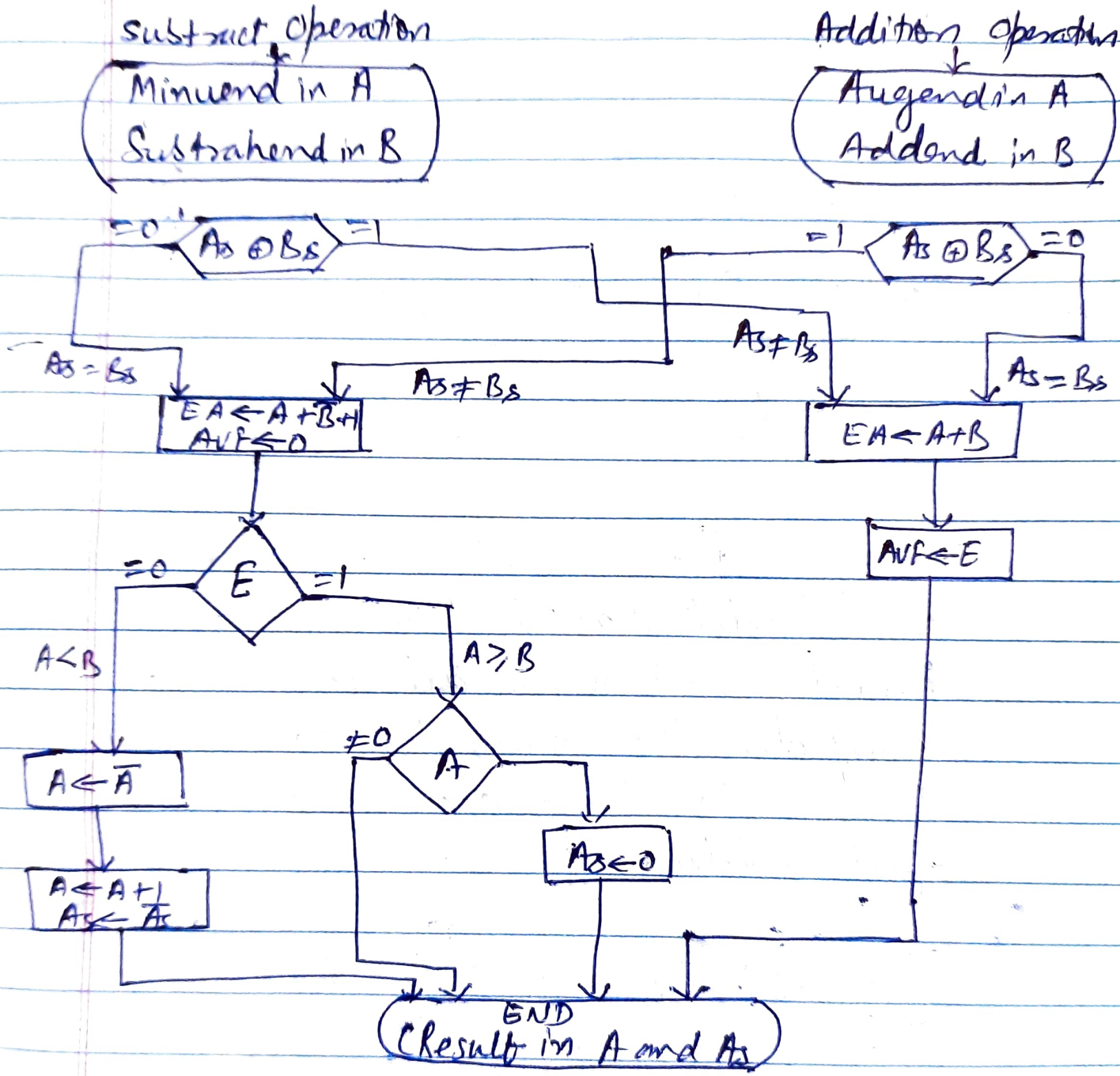


As Sign of A
Bs Sign of B
A & As Accumulator
AVF Overflow bit for A±B
E output carry for parallel Adder

Date. _____

Page No. _____

Flow chart

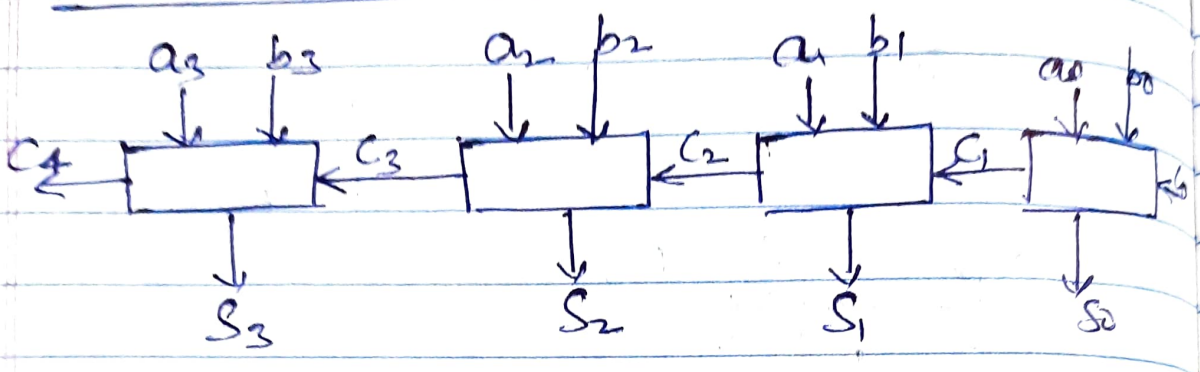


Date.
Page No.

Lecture-10

Look Ahead Carry Adder

4 Bit Adder



The n bit adder above is called a ripple-carry adder as the carry C_i need to be passed on through all lower bits to compute the sums for the higher bits. Recall the logic operations in the i^{th} full adder.

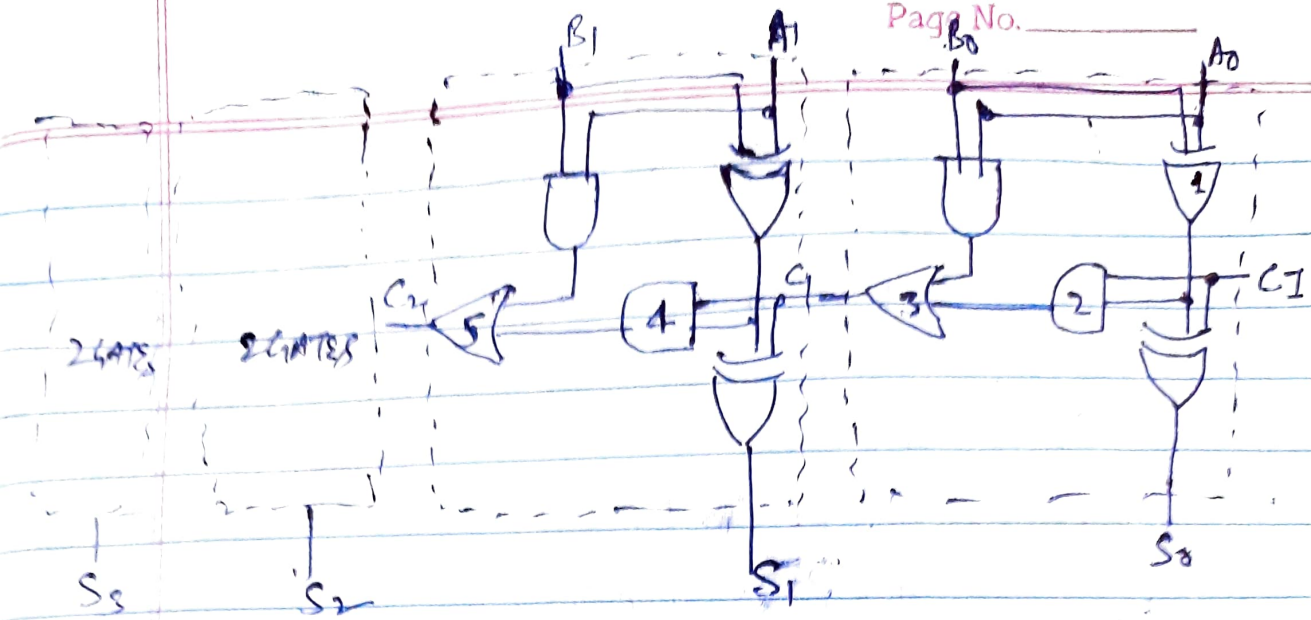
Logical operations in Adder

$$C_{i+1} = x_i y_i + x_i C_i + y_i C_i \qquad \text{(2 Gate Delay)}$$

$$S_i = x_i \bar{y_i} \bar{C_i} + x_i y_i C_i + \bar{x_i} y_i \bar{C_i} + \bar{x_i} \bar{y_i} C_i \qquad \text{(3 Gate Delay)}$$

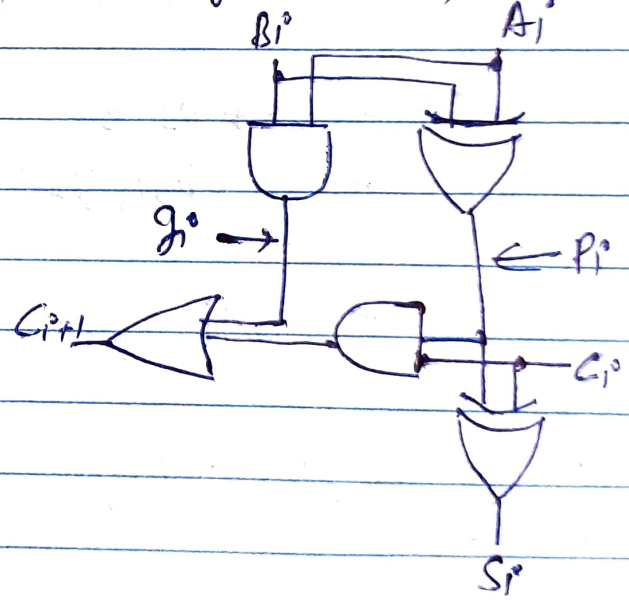
Delay in Ripple carry adder

- There is a very long path from A_0, B_0 and CI to CO and S_3
- For an n-bit ripple carry adder, the longest path has $2n+1$ gates.



Total Gate Delay = 9 Gates

A faster Way to Compute Carry outs



Ai	Bi	Ci	Ci+1
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

Algebraic Carry Out

$$C_1 = g_0 + p_0 C_0$$

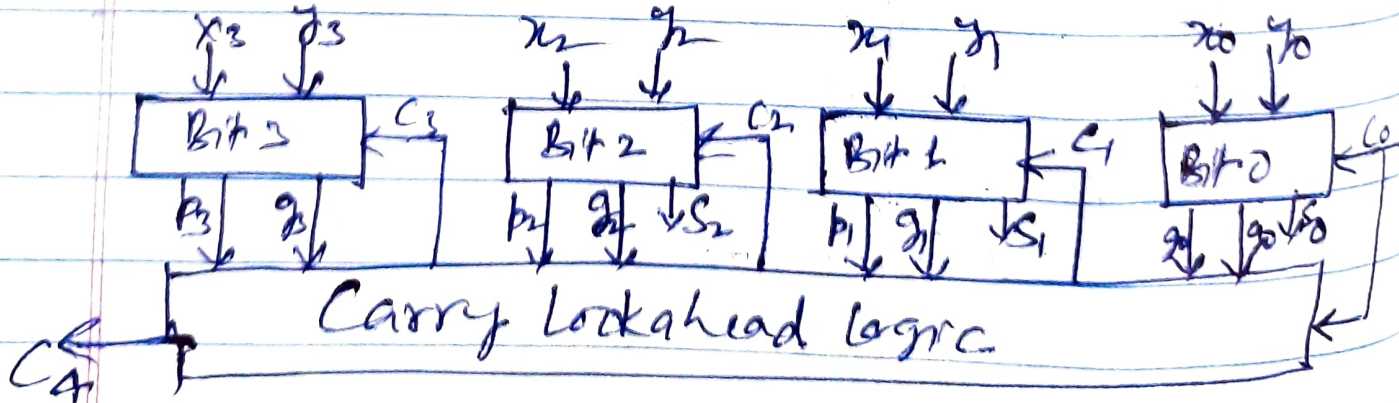
$$C_2 = g_1 + p_1 C_1$$
$$= g_1 + p_1 (g_0 + p_0 C_0)$$
$$= g_1 + p_1 g_0 + p_1 p_0 C_0$$

$$C_3 = g_2 + p_2 C_2$$
$$= g_2 + p_2 (g_1 + p_1 g_0 + p_1 p_0 C_0)$$
$$= g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 C_0$$

$$C_4 = g_3 + p_3 C_3$$
$$= g_3 + p_3 (g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 C_0)$$

$$= g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 C_0$$

Carry Lookahead Logic

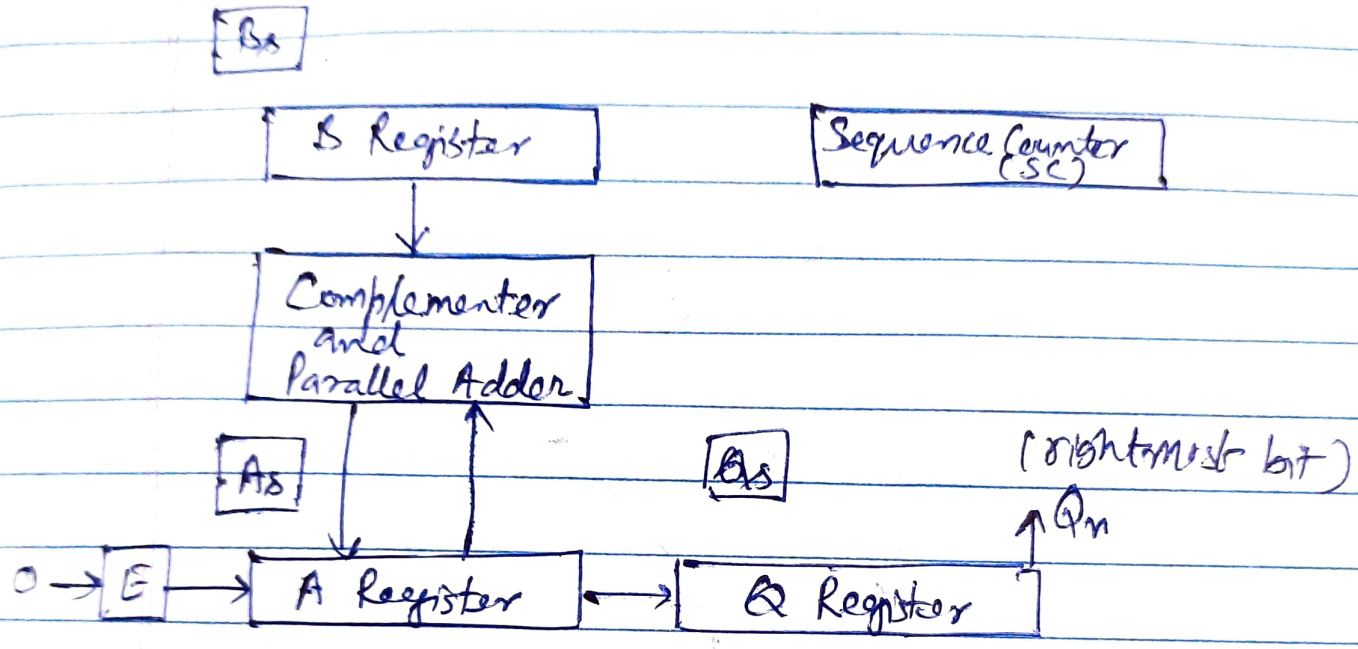


Lecture-11

Date.
Page No.

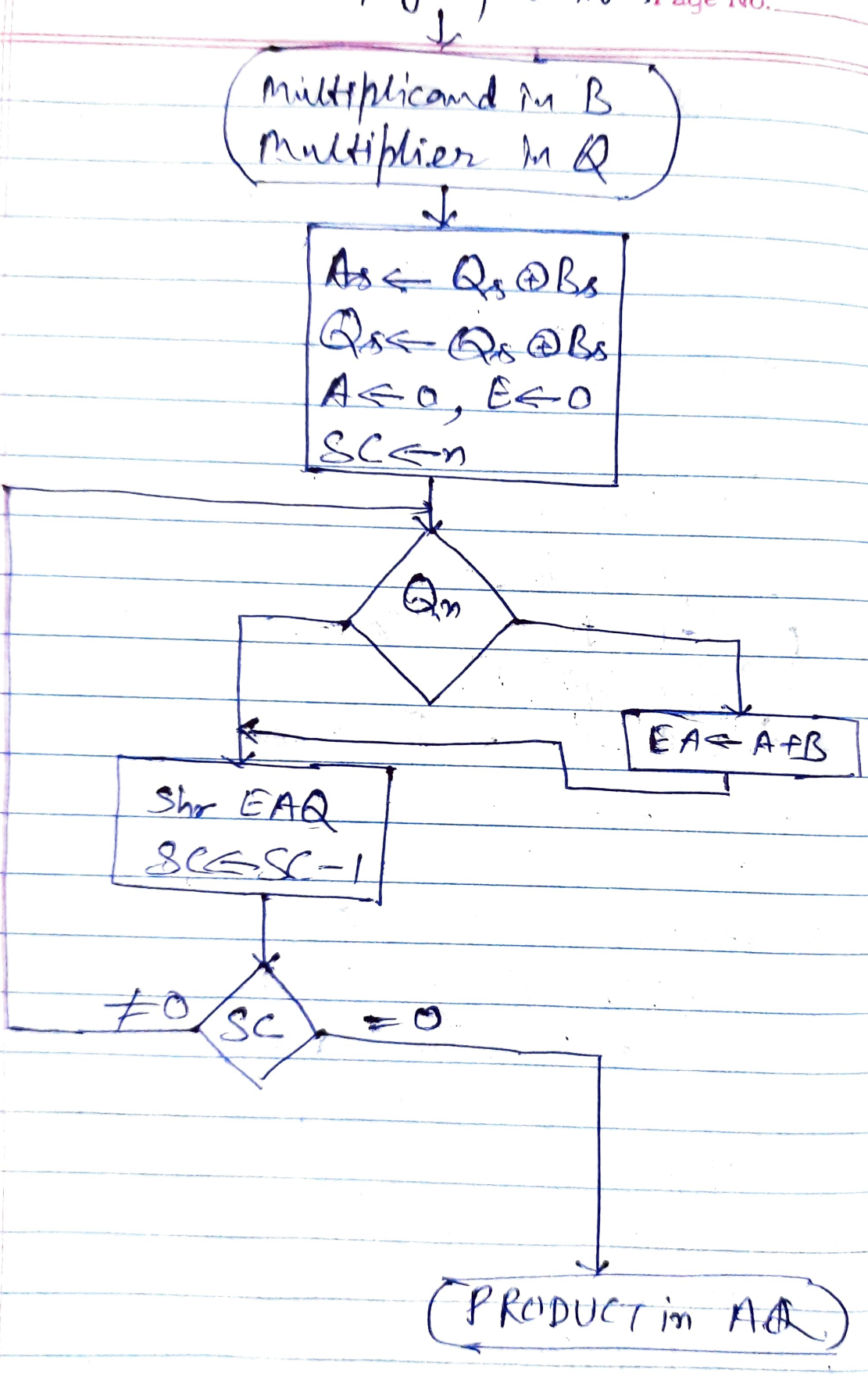
Signed Multiplication

H/W for Binary Multiplication



- Multiplier stored in Q register
- Sign of multiplier in Qs
- Sequence counter initially set to a value equal to no. of bits in multiplier
- Multiplicand stored in B.
- Qn is the rightmost bit of the multiplier which must be inspected next
- final product in EAQ register.

Date
Page No.



Flow chart for Binary Multiplications

Date.
Page No.

Examples 23 × 19 = 437

Multiplicand B = 10111	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
$Q_n = 1$; add B		10111		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
$Q_n = 1$; add B		10111		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
$Q_n = 0$; Shift right EAQ	0	01000	10110	010
$Q_n = 0$; Shift right EAQ	0	00100	01011	001
$Q_n = 1$; add B		10111		
Fifth partial Product	0	11011		
Shift EAQ	0	01101	10101	000

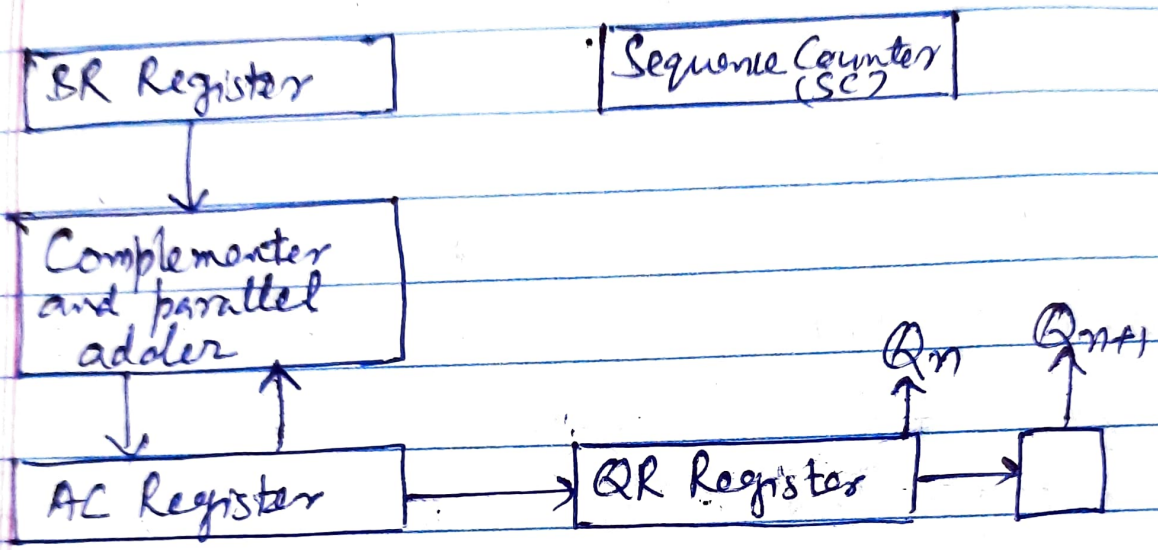
Final Product in AQ = 0110110101

Date.
Page No.

Lecture-12

Booth Multiplier

Hardware for Booth Algorithm



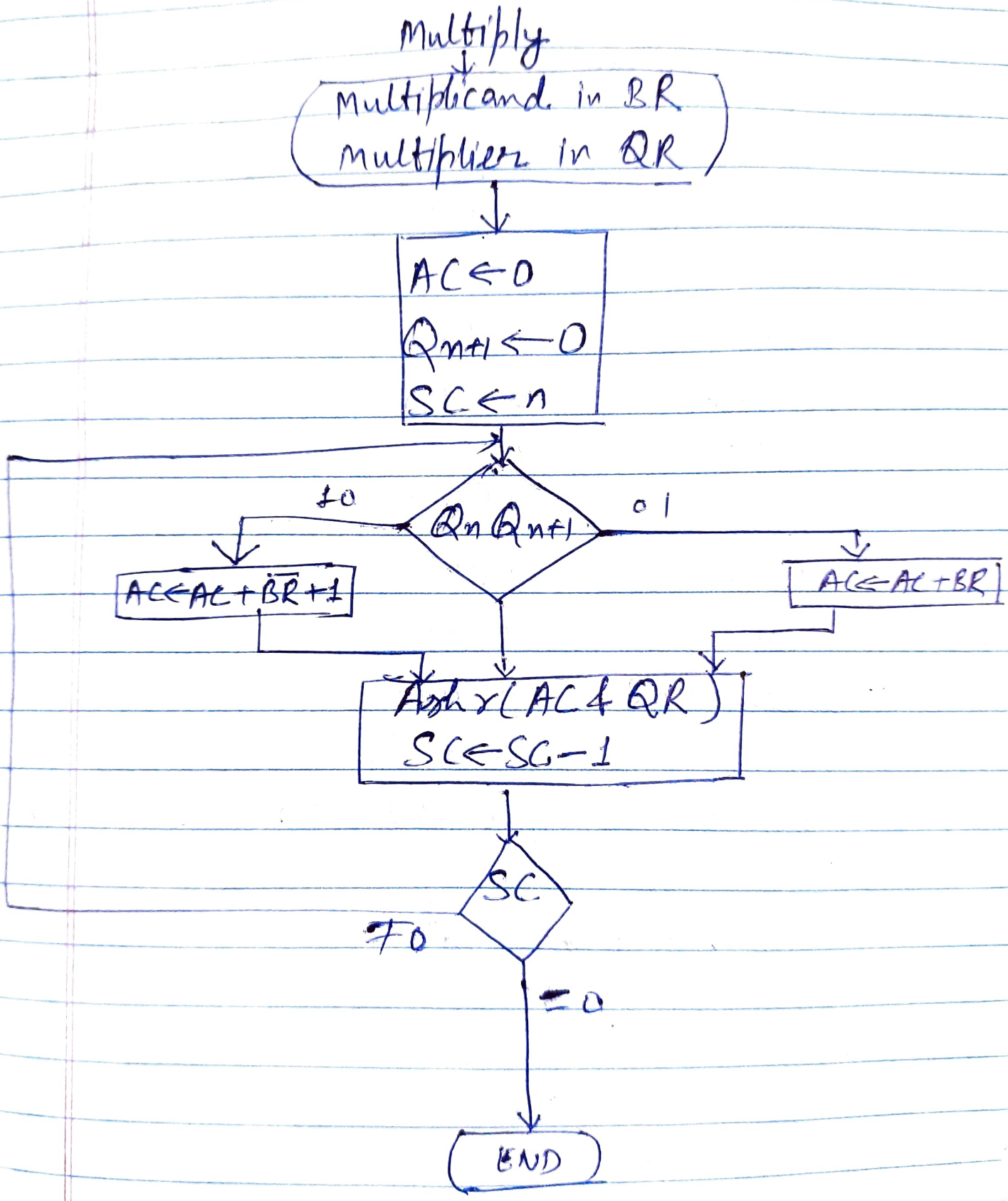
The Sign bits are not separated from the rest of registers.

Qn is the LSB of the multiplier in register QR

Qn+1 is an extra flip-flop appended to QR to facilitate a double bit inspection of the multiplier.

Date.
Page No.

Flow chart for Booth Algorithm



Date.
Page No.

Example. $(-9) \times (-13) = +117$ for $n=5$

$BR = 10111$ (Negative so in 2's Complement)

$\overline{BR} + 1 = 01001$

Qn	Qn+1		AC	QR	Qn+1	SC
		Initial	00000	10011 (2's complement)	0	101
1	0	Subtract BR	01001			
			01001			
		ashr	00100	11001	1	100
1	1	ashr	00010	01100	1	011
0	1	Add BR	10111			
			11001			
		ashr	11100	10110	0	010
0	0	ashr	11110	01011	0	001
1	0	Subtract BR	01001			
			00111			
		ashr	00011	10101	1	000

Takeaway's ~~cost~~ $+117$

Lecture-13

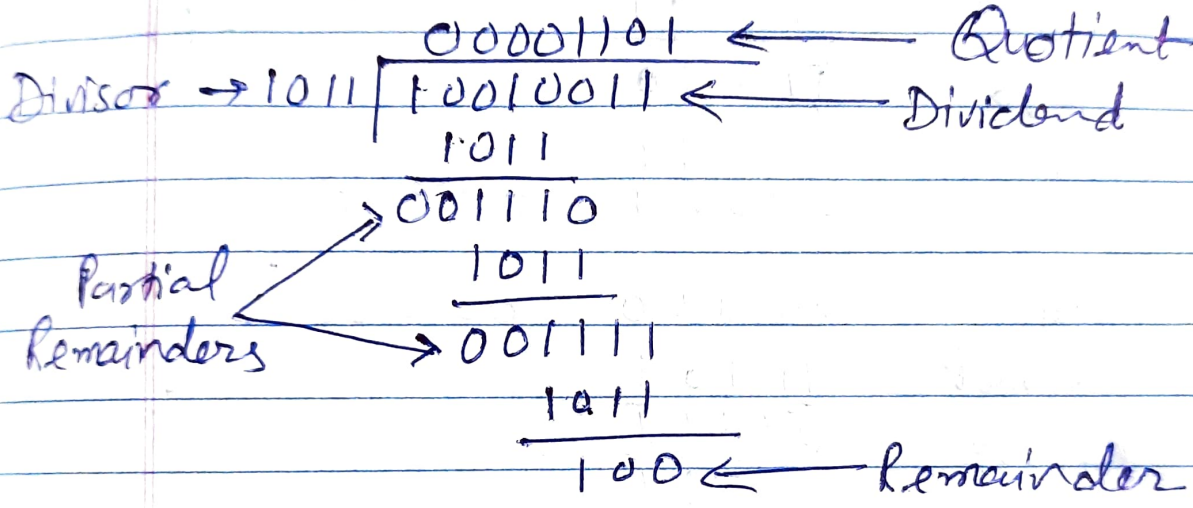
Date.
Page No.

Binary Division

Division of unsigned Binary Integers.

→ can be implemented by shift and subtract.

→ The multiplication hardware can be used for the division as well.



Dividend = Quotient * Divisor + Remainder

Algorithms for Division

The restoring division algorithms:

St: Do n times

(a) Shift A and Q left one binary position

(b) Subtract M from A, placing the answer back in A

(c) If the sign of A is 1, set q_0 to 0 and add M back to A (restore A); otherwise, set q_0 to 1.

Date.
Page No.

Restoring Division Example

```
    10
11 ) 1000
     11
     --
     10
```

Initially	00000	1000	
	00011		
Shift	00001	000□	
Subtract	11101		
Set qo	(1)1110		
Restore	11		
	00001	0000	First Cycle
Shift	00010	000□	
Subtract	11101		
Set qo	(1)1111		
Restore	11		
	00010	0000	Second Cycle
Shift	00100	000□	
Subtract	11101		
set qo	(0)0001		
Shift	00010	0001	Third Cycle
Subtract	11101	001□	
Set qo	(1)1111		
Restore	11		
	00010	0010	Fourth Cycle
	Remainder	Quotient	

Non-Restoring Division

- STEP1: Do n TIMES
  - If the sign of A is 0, shift A and Q left one binary position and subtract M from A
  - otherwise, shift A and Q left and add M to A. If the sign of A is 0, set Q_0 to 1; otherwise set Q_0 to 0.
- STEP2: If the sign of A is 1, add M to A

e.g. for same. 2/3

Initially	00000	1000	
	00011		
Shift	00001	000□	First cycle
Subtract	11101		
Set q_0	11110	000[0]	
Shift	11100	000□	Second cycle
~~Subtract~~ ADD	00011		
Set q_0	11111	000[0]	
Shift	11110	000□	Third cycle
Add	00011		
Set q_0	00001	000[1]	
Shift	00010	001□	Forth cycle
Subtract	11101		
Set q_0	11111	001[0]	

Quotient

Step2

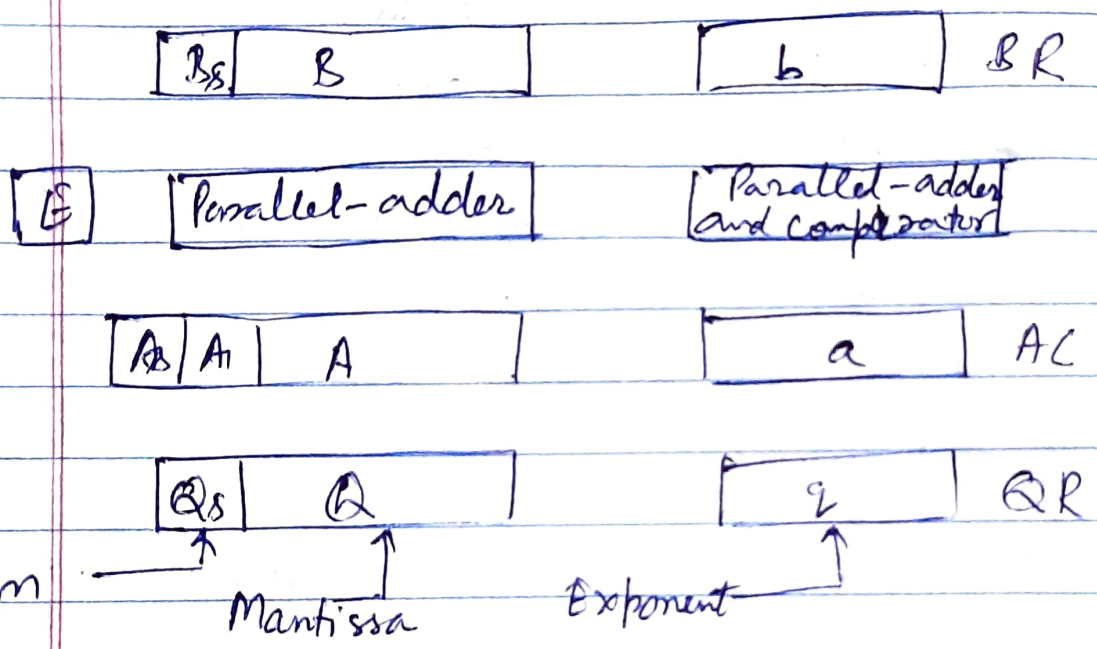
```
  11111
Add 00011
-----
  00010  ← Remainder
```

Lecture-14

Date.
Page No.

Floating Point Arithmetic Operations

Floating Point Registers



Floating Point Addition and Subtraction

Registers:

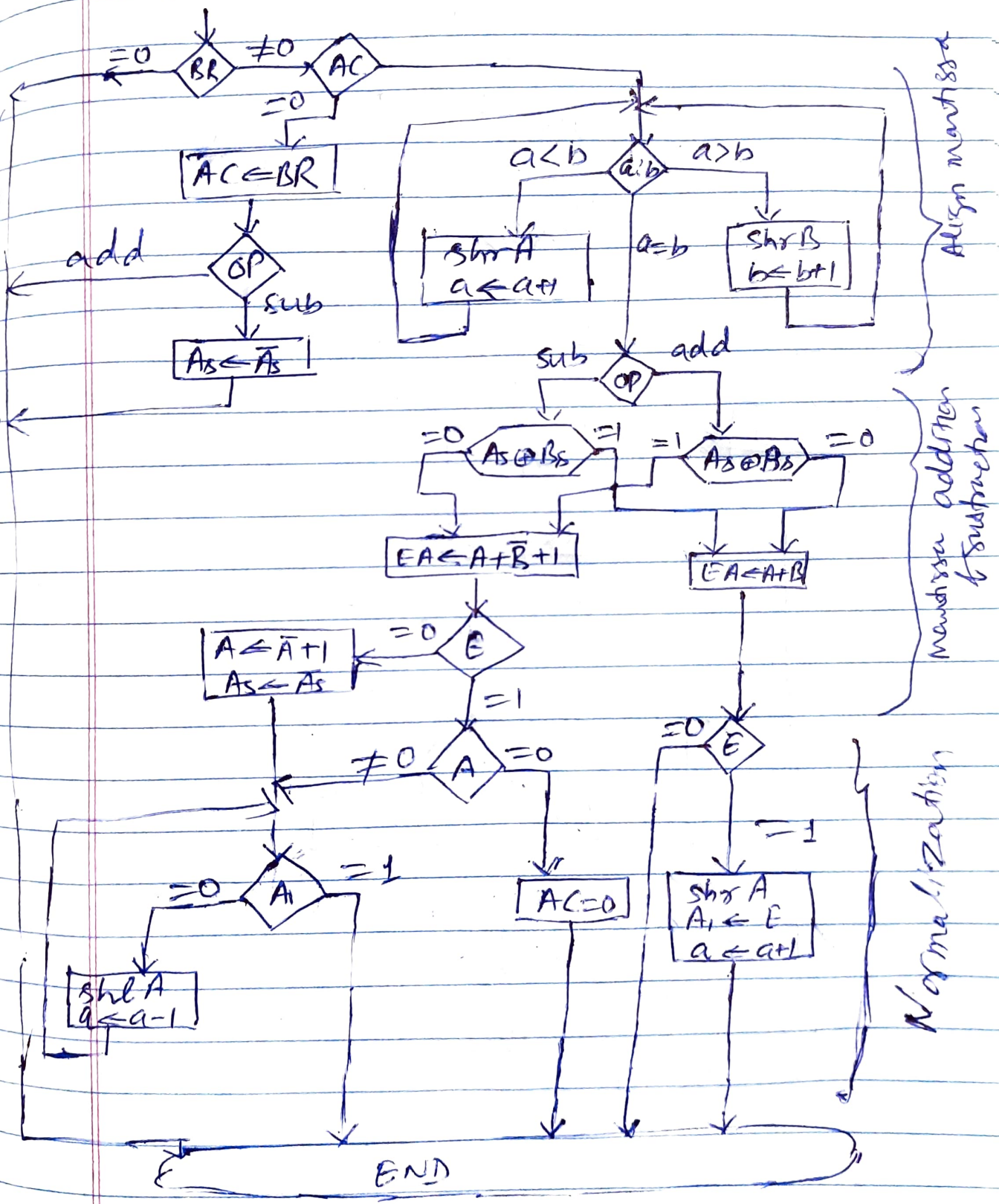
- $AC \leftarrow AC + BR$
- $AC \leftarrow AC - BR$

Algorithms

1. check for zeros
2. Align the mantissas
3. Add or subtract the mantissas
4. Normalize the result.

Date.
Page No.

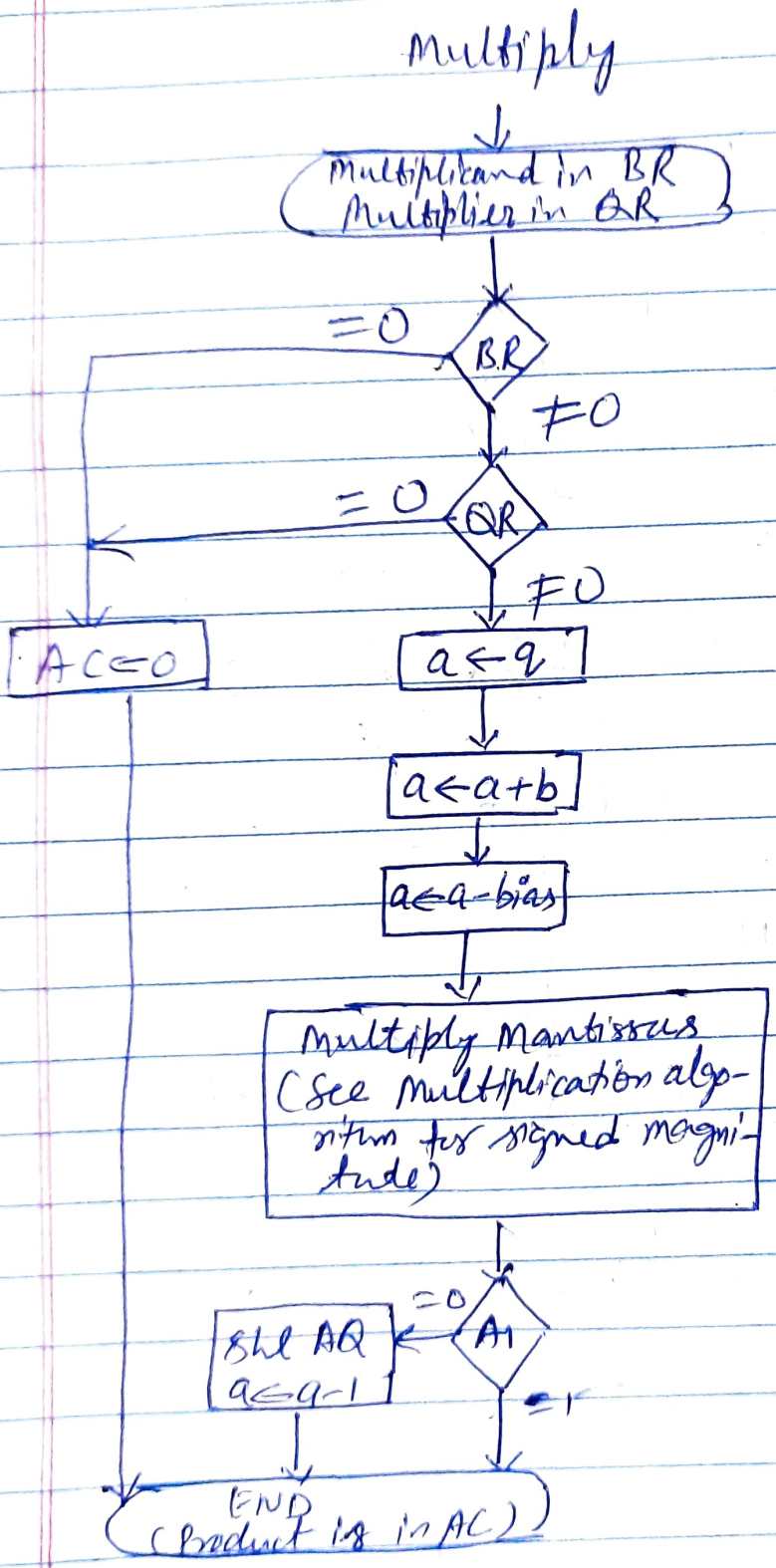
Flow chart for addition and subtraction:



Lecture - 15

Date.
Page No.

Floating Point Multiplication

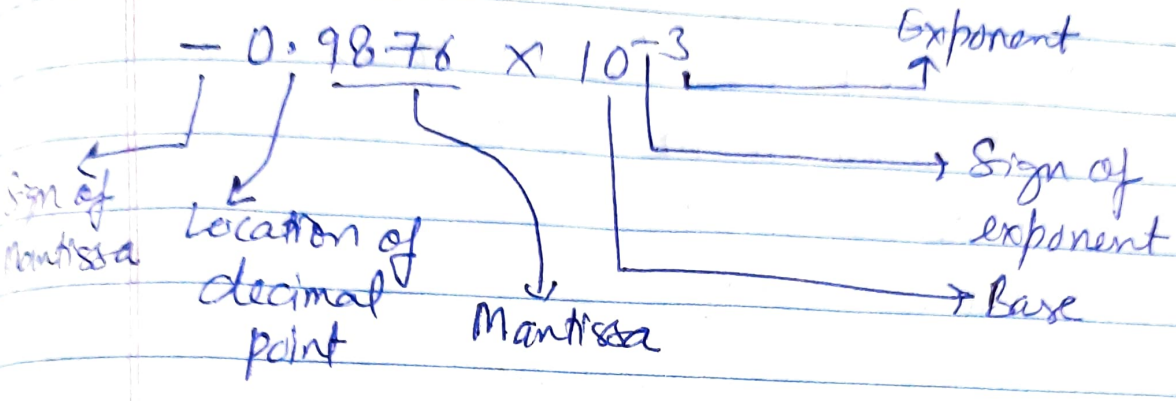


Lecture – 16

Date.
Page No.

IEEE Standard for Floating point Numbers.
IEEE 754

Part of Floating point Numbers.



Floating Point Number should be in Normalized form.

Example:

2.0×10^{-9} Normalized
0.2×10^{-8} Not-Normalized
20.0×10^{-10} Not-Normalized

Binary Numbers in Normalized form

ex: 1.1010000000000000
$= 1.101_2$

Biased Exponent

When we store any number then we need

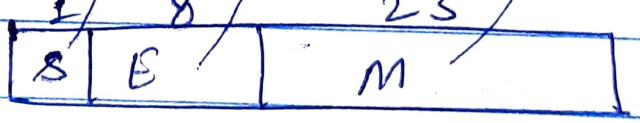
Date.
Page No.

2 bits for storing sign of mantissa and exponent. If we biased a number then exponent should be positive and we always store it as positive.

IEEE 754 standard Represent Floating point Numbers in single precision or in double precision

1. Single precision

$$\text{Value} = N = (-1)^S \times 2^{E-127} \times (1.M)$$



$0 < E < 255$

$e = E - 127$

Actual exponent

2. Double precision

$$\text{Value} = N = (-1)^S \times 2^{E-1023} \times (1.M)$$