COMPUTER ARITHMETIC

- The Addition, subtraction, multiplication and division are the four basic arithmetic operations.
- There are two types of representation for computer Arithmetic operations

1. Fixed-point binary data
2. Floating-point binary data

Addition and Subtraction algorithm

- Consider the magnitude of the two numbers by A and B. When the signed numbers are added or subtracted, there are eight different conditions
- Four conditions for addition and four for subtraction
- If two operands are same then perform <u>addition operation</u>
- If two operands are different then perform <u>subtraction operation</u>

Addition and Subtraction algorithm

Addition :

- When the signs of A and *B are equal add the two magnitudes and attach the sign of A to the result.*

- **subtraction: can divided into three parts (A>B)(A<B)(A=B)**

If(A>B): when the sign of A and B are different compare the magnitudes and subtract the smaller number from the larger. Choose the sign of the result to be the same as *A*

If(A<B): when the sign of A and B are different compare the magnitudes and subtract B-A . Choose the sign of the result to be the complement of *A*

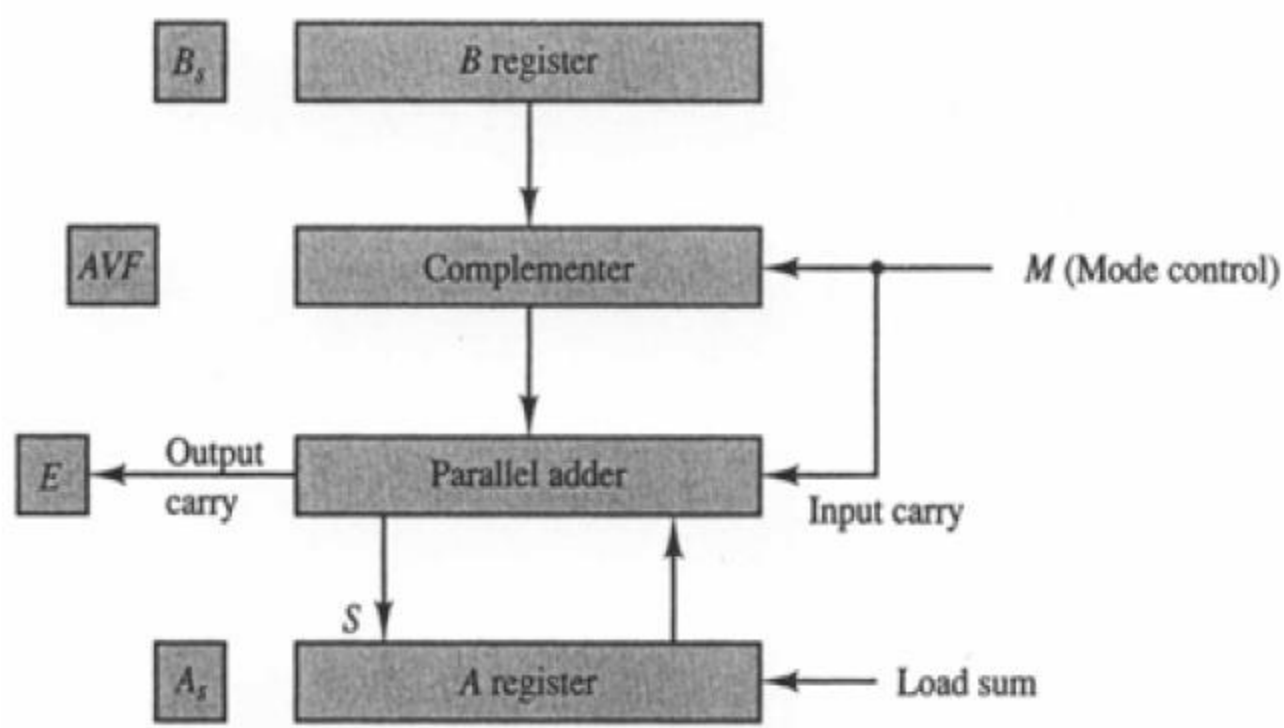
---

if(A=B) :if two magnitudes are equal subtract A-B and make the sign of the result is positive

Addition and Subtraction algorithm

Operation	Add Magnitudes	Subtract Magnitudes		
		When $A > B$	When $A < B$	When $A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

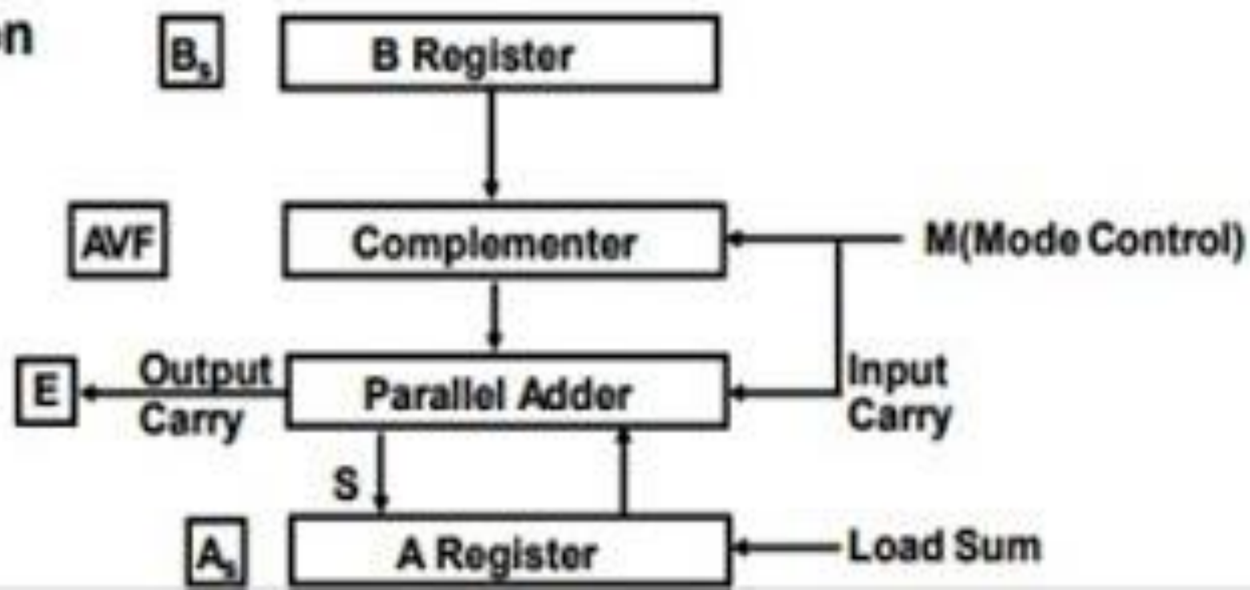
Hardware Implementation

- Let A & B are two registers that holds the magnitudes of numbers As and Bs are two flip-flops That holds the sign of corresponding registers
- 

Addition: A + B ; A: Augend; B: Addend
Subtraction: A - B: A: Minuend; B: Subtrahend

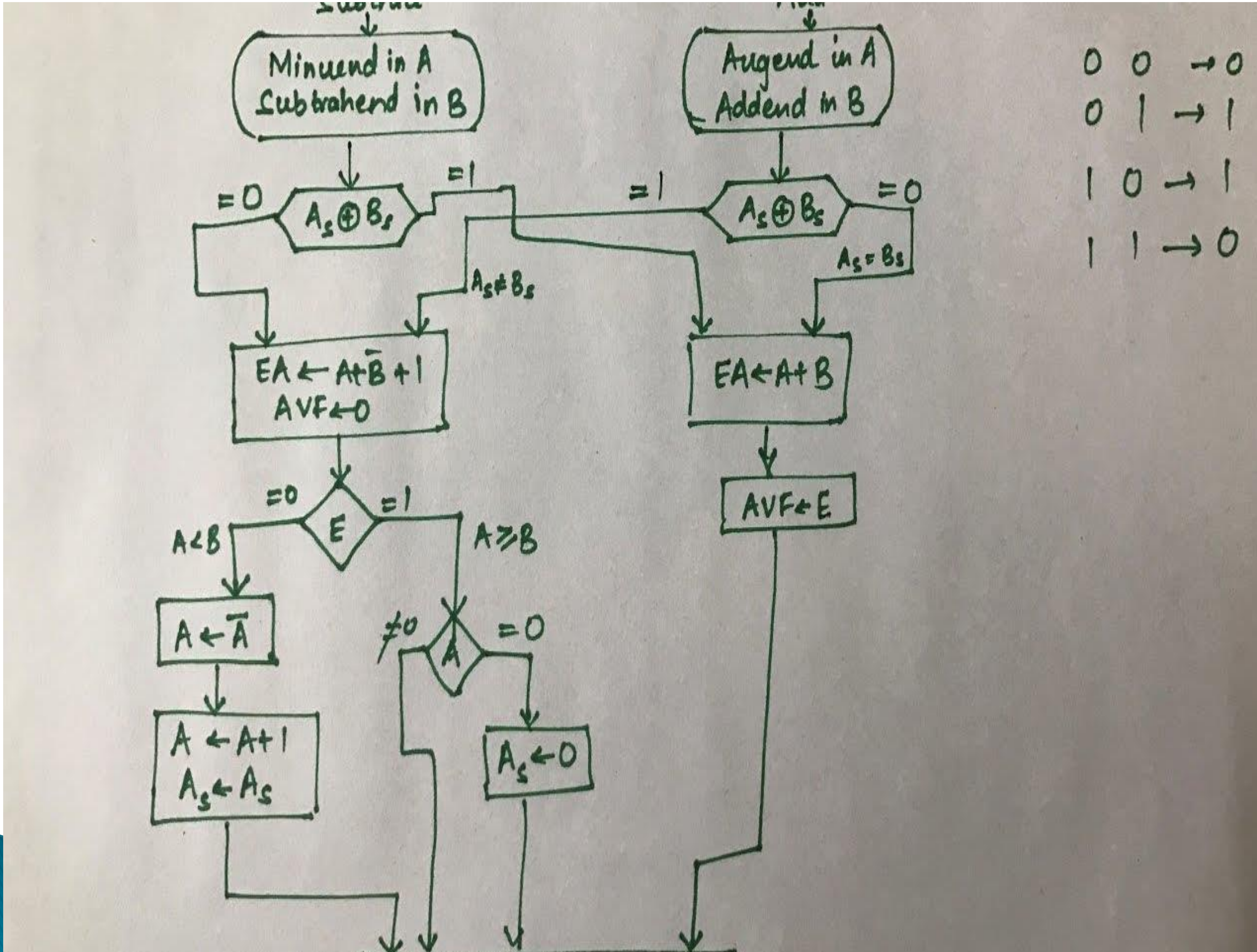
Operation	Add Magnitude	Subtract Magnitude		
		When A>B	When A<B	When A=B
(+A) + (+B)	+(A + B)			
(+A) + (- B)		+(A - B)	- (B - A)	+(A - B)
(- A) + (+B)		- (A - B)	+(B - A)	+(A - B)
(- A) + (- B)	- (A + B)			
(+A) - (+B)		+(A - B)	- (B - A)	+(A - B)
(+A) - (- B)	+(A + B)			
(- A) - (+B)	- (A + B)			
(- A) - (- B)		- (A - B)	+(B - A)	+(A - B)

Hardware Implementation

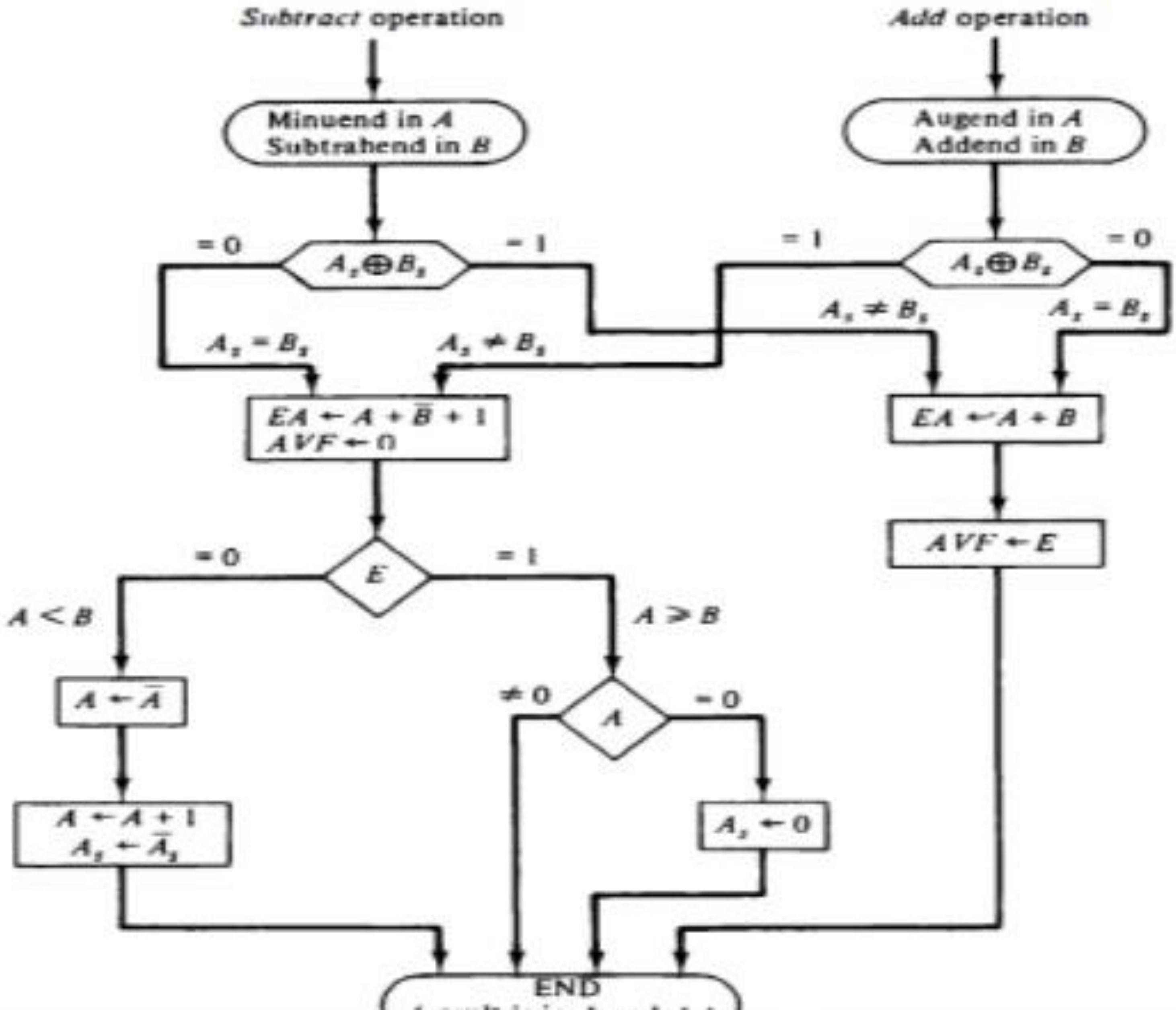


Hardware Implementation

- Parallel adder is needed to perform the micro operation ie $S \leftarrow A+B, S \leftarrow A+B+1$
- Complement is needed to establish $A<B, A>B, A=B$
- AVF that holds the overflow bit when A and B is Added
- When Mode(M)=0 then perform addition operation ($S \leftarrow A+B$)
- When Mode(M)=1 then perform Subtraction operation ($S \leftarrow A+B+1$)


Minuend in A
Subtrahend in B
Augend in A
Addend in B
$A_s \oplus B_s$
=0
=1
=1
$A_s \oplus B_s$
=0
$A_s \neq B_s$
$A_s = B_s$
$EA \leftarrow A + \bar{B} + 1$
$AVF \leftarrow 0$
$EA \leftarrow A + B$
$AVF \leftarrow E$
E
=0
=1
A<B
A≥B
$A \leftarrow \bar{A}$
≠0
A
=0
$A \leftarrow A + 1$
$A_s \leftarrow A_s$
$A_s \leftarrow 0$
0 0 → 0
0 1 → 1
1 0 → 1
1 1 → 0

with Signed-Magnitude Data



Multiplication

- Multiplication of two fixed-point binary numbers in signed magnitude representation is done with successive shift and adds operations.

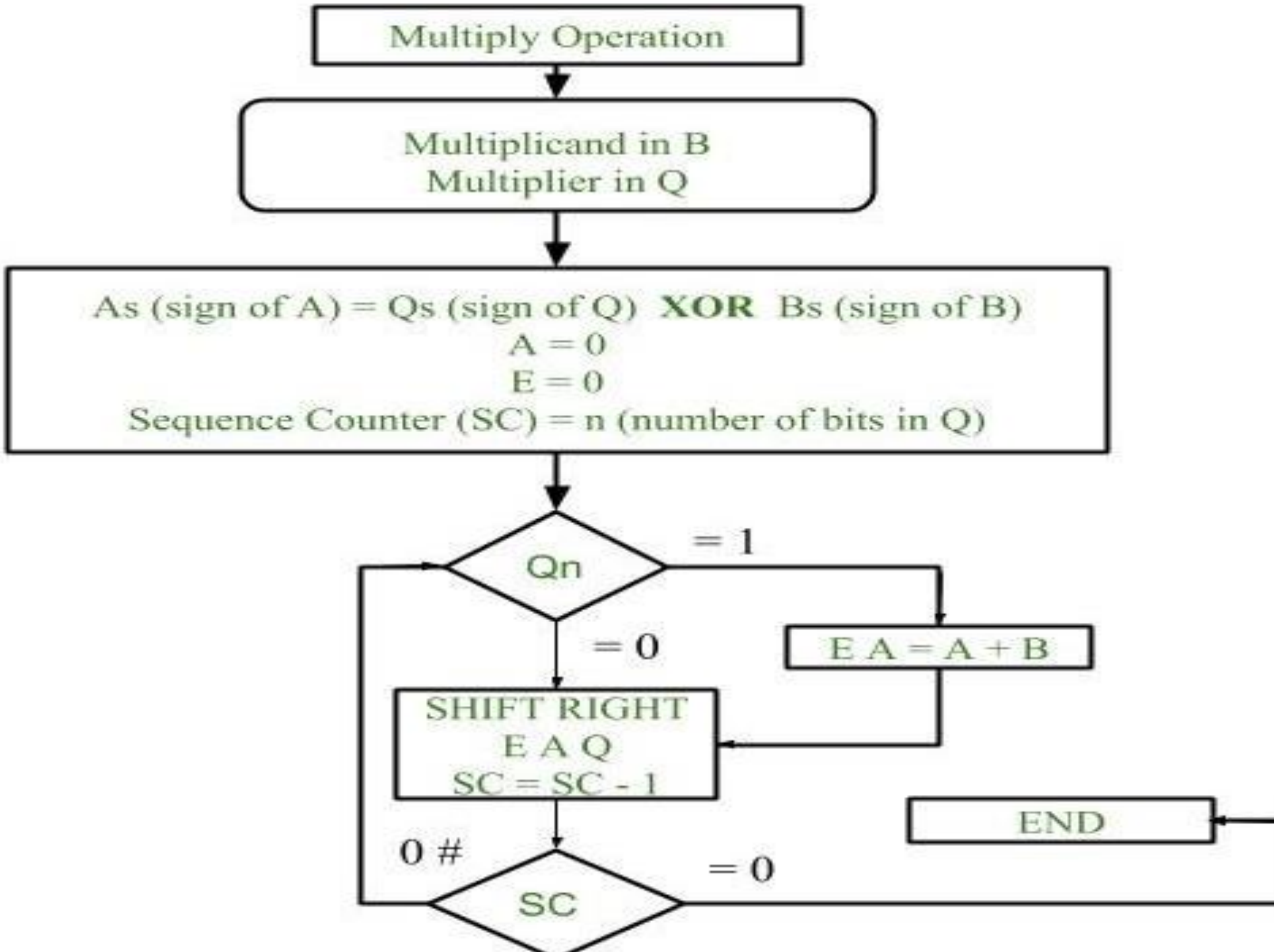
B*Q where B is a **multificand** Q is **multiplier**

The LSB of multiplier is 1 then multiplicand is copied down using shift left operation if it is 0 then 0s are copies down using shift left operation

Finally the numbers are added and their sum produce the result

Example

```
23          10111 Multiplicand
19        x 10011 Multiplier
------------------------------
            10111
           10111
          00000
         00000
        10111
------------------------------
437     110110101 Product
```



Multiply Operation
Multiplicand in B
Multiplier in Q
As (sign of A) = Qs (sign of Q) XOR Bs (sign of B)
A = 0
E = 0
Sequence Counter (SC) = n (number of bits in Q)
Qn
= 1
= 0
E A = A + B
SHIFT RIGHT
E A Q
SC = SC - 1
END
0 #
= 0
SC

EXAMPLE

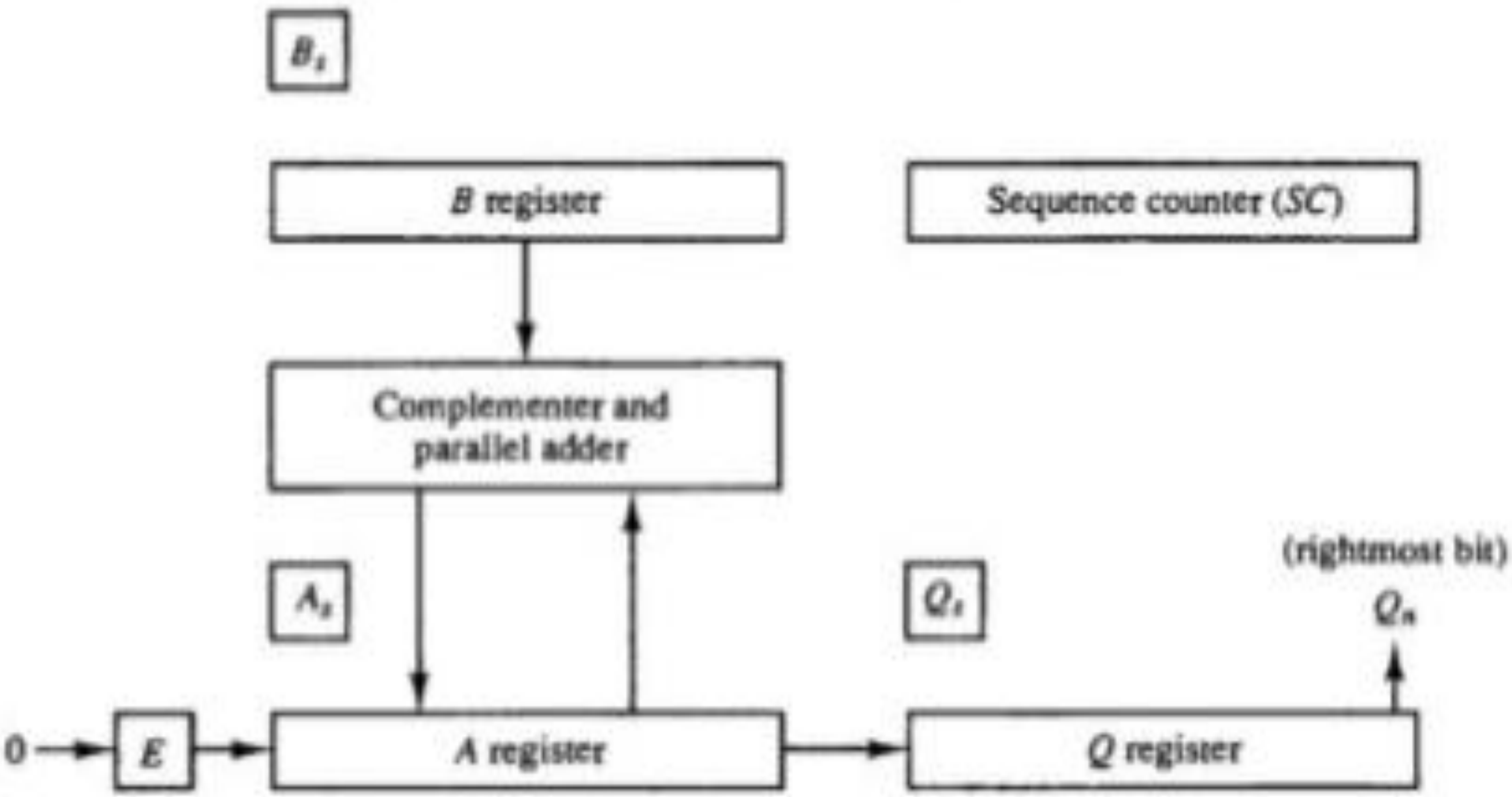
- B=23,Q=19
- B=10111
- Q=10011
- A=00000
- E=0
- SC=101

B	E	A	Q	SC
10111	0	00000	10011	101
Qn=1				
ADD A+B		00000		
		10111		
	0	10111	10011	100
SHR EAQ	—	01011	11001	
Qn=1				
ADD A+B		01011		
		10111		
	1	00010	11001	
SHR EAQ	—	10001	01100	011
Qn=0				
SHR EAQ	0	10001	01100	010
	—	01000	10110	
Qn=0	0	01000	10110	
SHR EAQ	—	00100	01011	001
Qn=1				
ADD A+B		00100		
		10111		
	0	11011	01011	
SHR EAQ	—	01101	10101	000

- AQ->0110110101
- THE FINAL RESULT IS 0110110101=437

HARD WARE IMPLEMENTATION FOR MULTIPLICATION WITH SIGNED MAGNITUDE REPRESENTATION(BxQ)

Figure 10-5 Hardware for multiply operation.



- B X Q where B IS Multiplicant and Q is Multiplier
- Multiplicant is stored in B register and its sign is B’s
- Initially the multiplier is stored in a register and its sign is Q’s
-

SEQUENCE COUNTER

- It is set to a number which is equal to number of bits Is multiplier
- Is shifted operations ‘E’ bit is shifted to most significant bit of ‘A’ register and least significant bit of ‘A’ register is shifted to most significant bit of register
- Equation for the above description is

E->MSB(A)

LSB(A)->MSB(Q)

MULTIPLICATION ALGORITHM WITH SIGNED 2'S COMPLIMENT REPRESENTATION

- It is also called booths algorithm

(-9)X(-13)=+117

BR->MULTIPLICANT

QR->MULTIPLIER

-9=2'S COMPLIMENT OF BR

-13=2'S COMPLIMENT OF QR

9=1001

    0110

        1

  10111

13 = 1101
-0010
1

10011 → multiplier

Recoded table :

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

→ This recoded table is applied to the multiplier.

1 0 0 1 1 0

-1 0 +1 0 -1 → Recoded multiplier

Multificant = 10111

Recorded multiplier = -1 0 +1 0 -1

Condition :

→ If the recorded multiplier is -1 then 2's compliment of multificant is copied down.

→ If the recorded multiplier is 0 then 0's are copied down.

→ If the recorded multiplier is +1 then using shift left operation of multification.

Using shift left operation ...

1 ~~0~~ 1 0 1 1 1 → multificant

−1 0 +1 0 −1 → Recorded multiplier

0 0 0 0 0 1 0 0 1 0

0 0 0 0 0 0 0 0 1 1

1 1 1 0 1 1 1

0 0 0 0 0 0

0 1 0 0 1

1 1 1 1

0 0 1 1 1 0 1 0 1

64 32 16 8 4 2 1

10111

01000

01001

11111

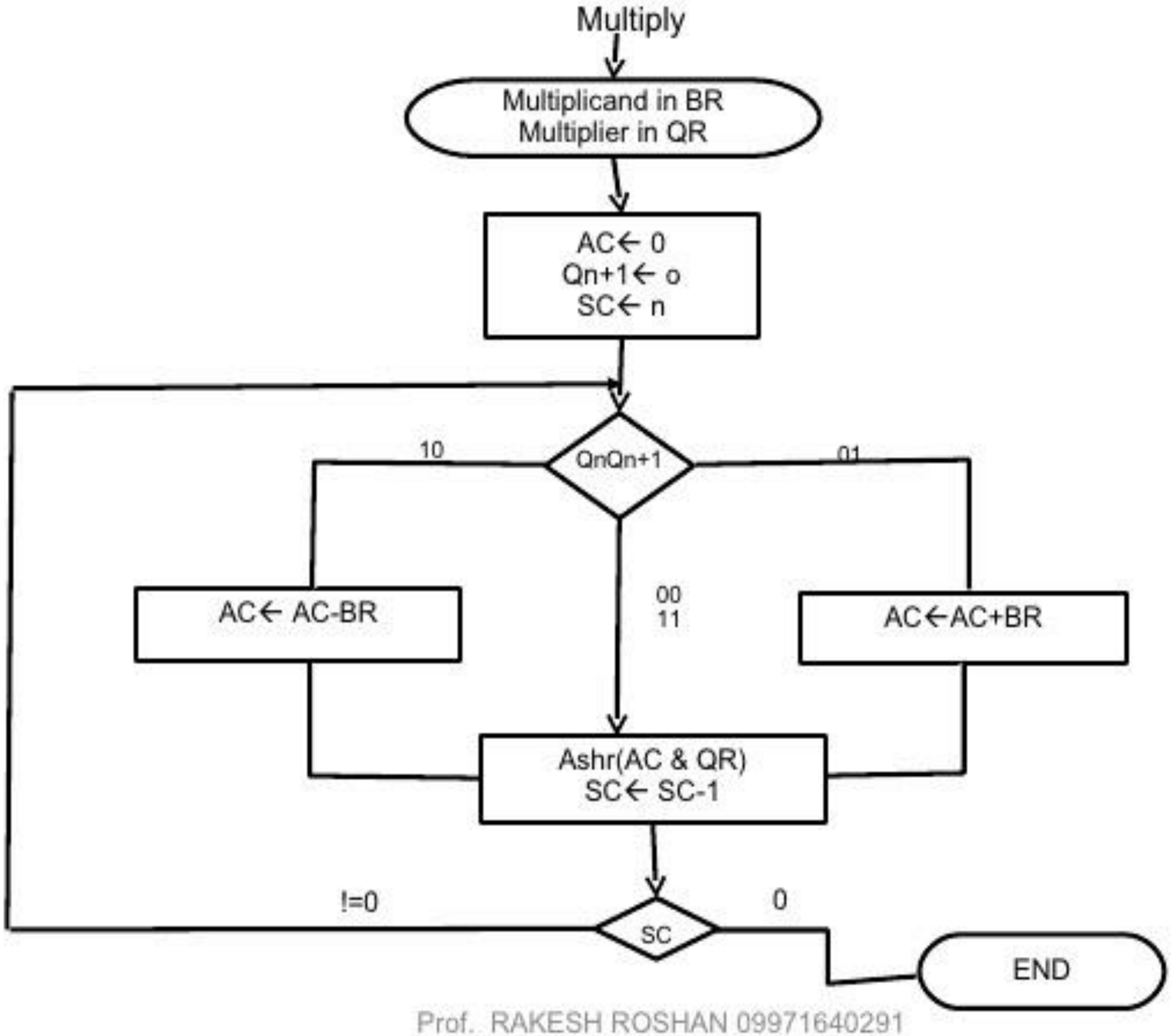
01000

01001

Ex (2)

Booth Multiplication Algorithm

Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation



For example

- -9 X -13

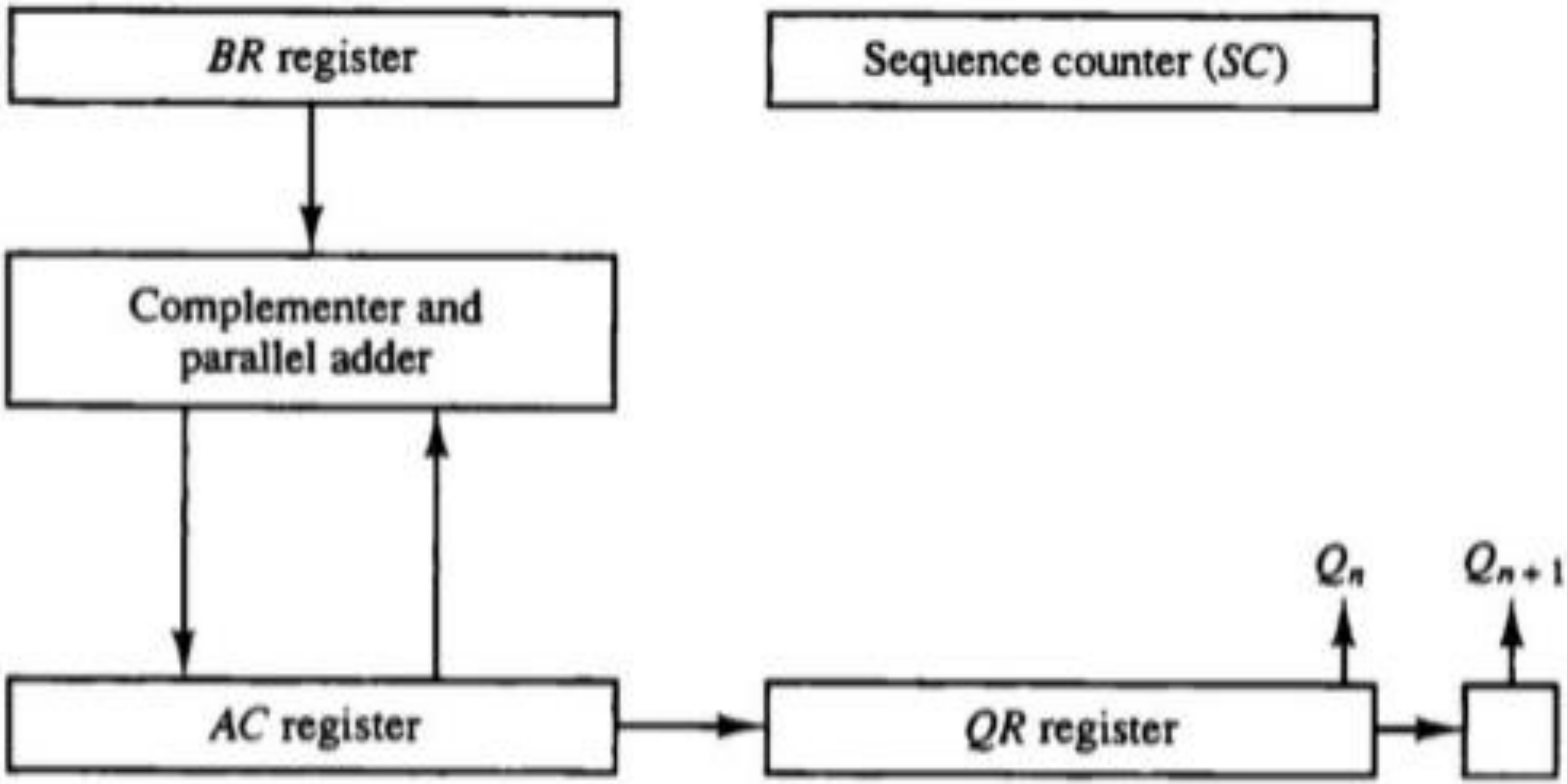
-9 = BR=10111->MULTIFICANT

-13=QR=10011->MULTIFIER

Qn,Qn+1	BR	AC	QR	Qn+1	SC.
(1,0)	10111	00000	10011	0	101
(1,0)	AC+$\overline{BR}$+1 Ash R ACQR	00000 01001 01001 00100	10011 11001	1	100
(1,1)	A sh R AC QR	00100 00010	11001 01100	1	011
(0,1)	A sh R ACQR	00010 10111 11001 11100	01100 10110	0	010
(0,0)	A SR R AC QR	11100 11110	10110 01011	0	001
(1,0)	A SR R AC+$\overline{BR}$+1 ~~Ash~~	11110 01001 00111 00011	01011 10101	1	000

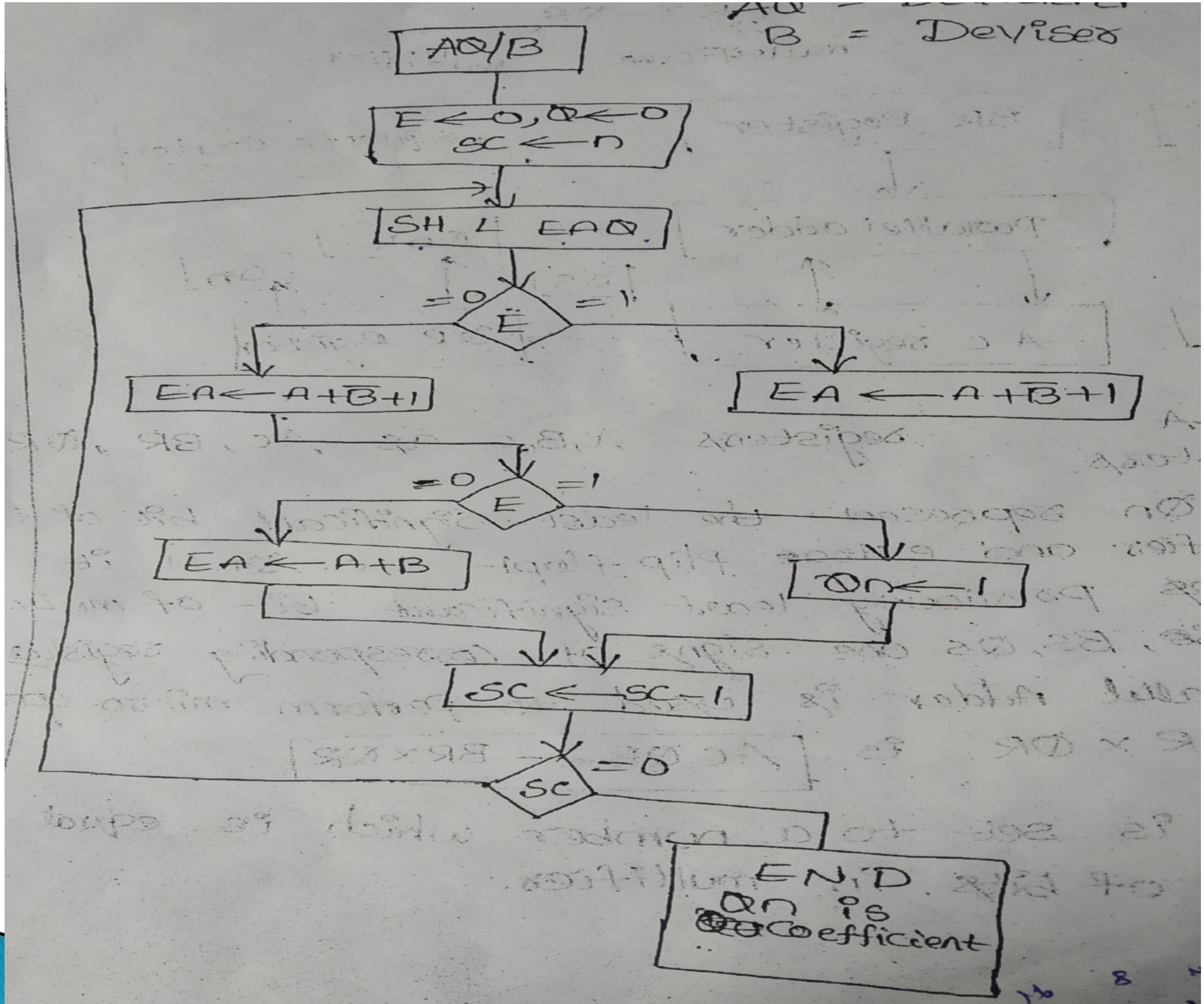
Hardware Implementation For Boots Algorithm

Figure 10-7 Hardware for Booth algorithm.



DIVISON ALGORITHM

- AQ=DIVIDEND
- B=DIVISOR
- A=01110
- B=10001
- AQ=0111000000


B = Deviser
AQ/B
E ← 0, Q ← 0
SC ← n
SH L EAQ
=0
E
=1
EA ← A+B̄+1
EA ← A+B̄+1
=0
E
=1
EA ← A+B
Qn ← 1
SC ← SC-1
SC
=0
END
Qn is
Coefficient

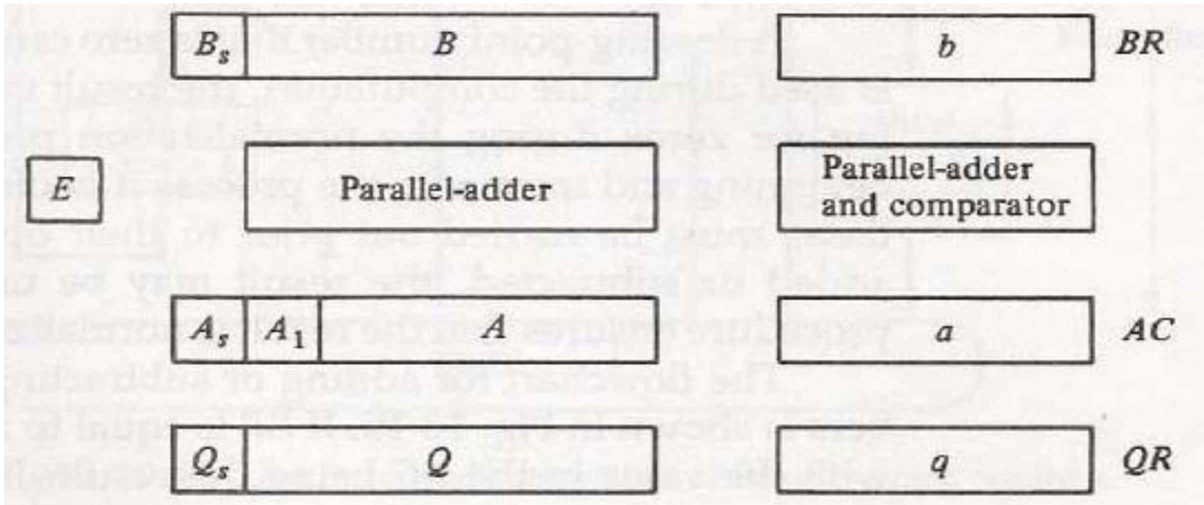
Divisor	E	A	Q	SC
	0	01110	00000	101
SHL EAQ EA ← A+$\bar{B}$+1	0 1	11100 01111 01011	00000_ 00001	100
SHL EAQ EA ← A+$\bar{B}$+1	0 1	10110 01111 00101	0001_ 00011	011
SHL EAQ EA ← A+$\bar{B}$+1	0 0 0 1	01010 01111 11001 10001 01010	00110 00110	010
SHL EAQ EA ← A+$\bar{B}$+1 10	0 1	10100 01111 00011	01101 01101	001
SHL EAQ EA ← A+$\bar{B}$+1 0 1	0 0 1	00110 01111 10101 10001 00110	11010	000

Floating-Point arithmetic Operations:

- The scientific notation for floating point numbers is as shown below:
- M * Re   **10010.100100**
- Where M indicates Mantissa, R indicates Radix, and e indicates exponent.

- Different types of operations that can be performed on floating point numbers are addition, subtraction, division and multiplication

Floating-Point arithmetic Operations:



There are three registers, BR, AC and QR.

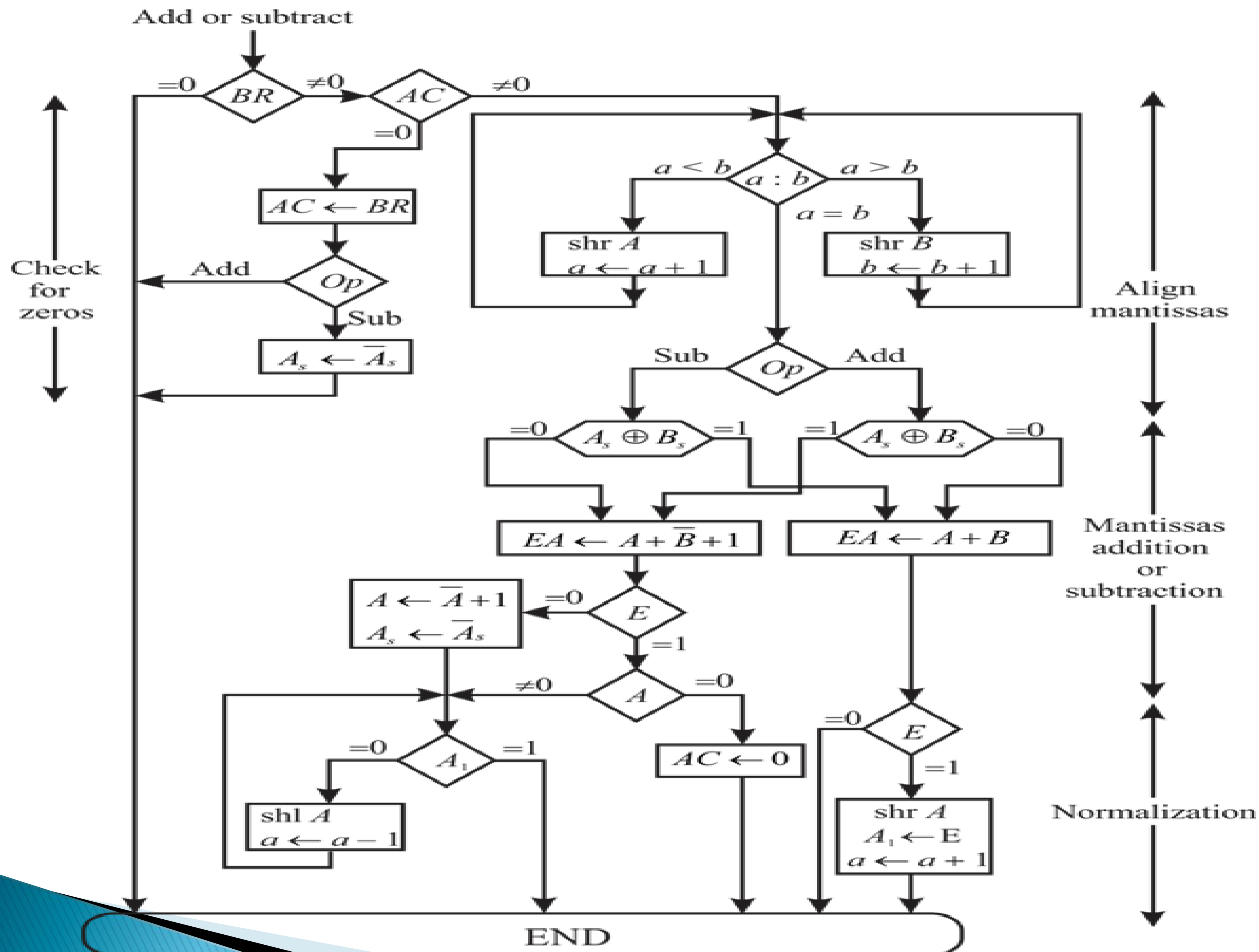
Each register is subdivided into two parts.

The mantissa part can be represented using uppercase letters and exponent part can be represented using lower case letters.

Addition and Subtraction:

- During the addition and subtraction, the floating point operands are in AC and BR.
- The resultant sum or difference is stored in AC. The algorithm can be divided into 4 parts.

1. Check for zero's
2. Align the mantissas
3. Add or subtract the mantissas
4. Normalize the result.



Add or subtract
BR
AC
=0
≠0
AC ← BR
Op
Add
Sub
a < b
a : b
a > b
a = b
shr A
a ← a + 1
shr B
b ← b + 1
Check for zeros
Align mantissas
Mantissas addition or subtraction
Normalization
E
A
=1
AC ← 0
A1
shl A
a ← a − 1
shr A
A1 ← E
a ← a + 1
EA ← A + B
END

Multiplication algorithm with floating point

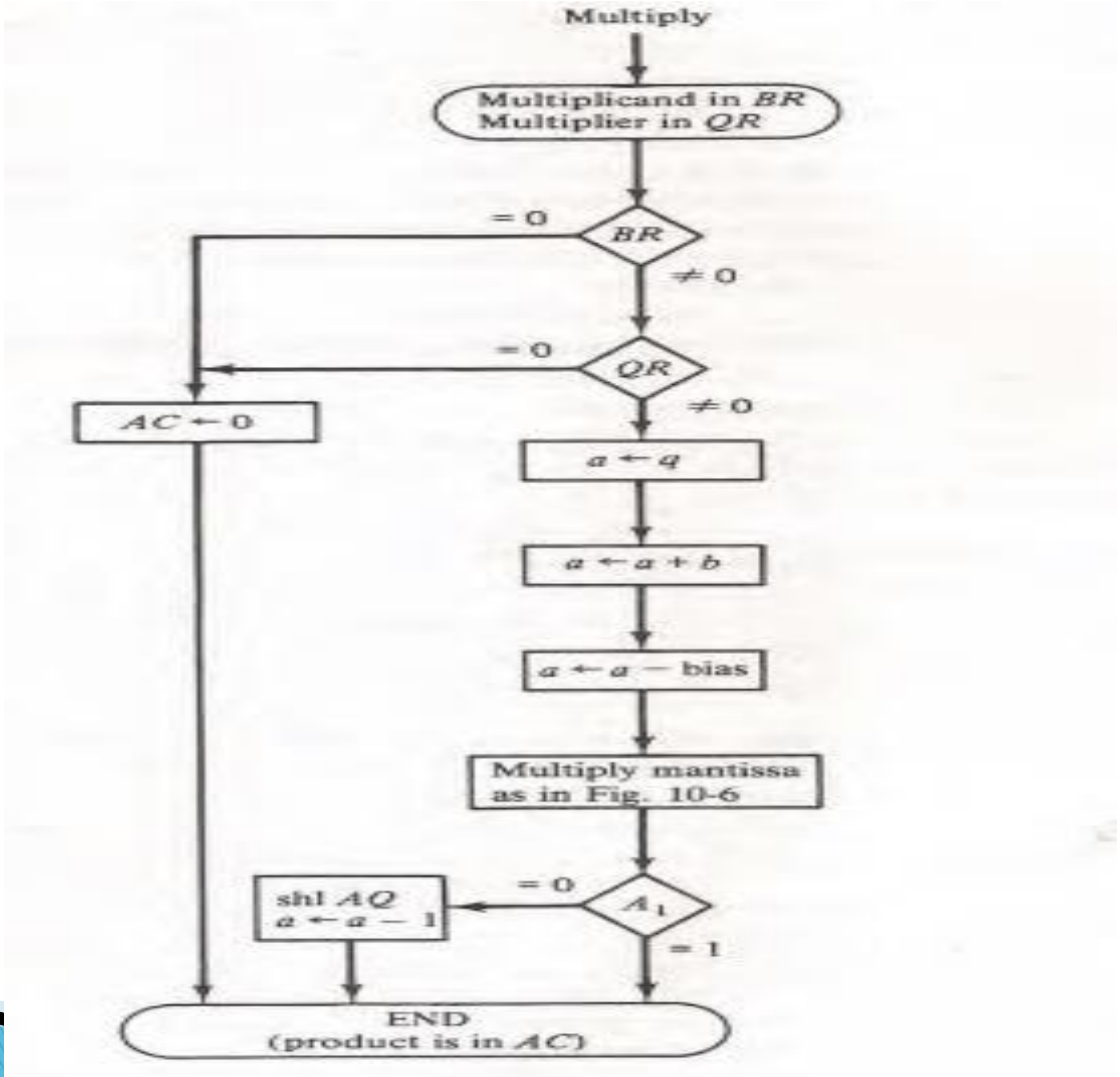
- The multiplication of two floating point numbers requires that multiply the mantissa and add the exponents.

- The multiplication of mantissa is performed in the same way as in fixed point numbers.

- The multiplication algorithm can be sub divided into four parts:

Multiplication:

1. Check for zero’s
2. Add the exponents
3. Multiply the mantissas
4. Normalize the product


Multiply
Multiplicand in BR
Multiplier in QR
= 0
BR
≠ 0
= 0
QR
≠ 0
AC ← 0
a ← q
a ← a + b
a ← a − bias
Multiply mantissa
as in Fig. 10-6
= 0
A1
= 1
shl AQ
a ← a − 1
END
(product is in AC)

Division algorithm with floating point

- Floating point division requires that the exponents be subtracted and the mantissas divided.
- The division algorithm can be sub divided into five parts:

1. Check for zeros
2. Initiate registers and evaluate the sign
3. Align the dividend
4. Subtract the exponents
5. Divide the mantissa