Department of CSE and IT

Arithmetic and logic unit

Unit: 2

Computer Organization & Architecture

B Tech 3rd Sem

Ashok Kumar Yadav

CSE and IT

Department

11/12/2020

Content

Arithmetic and logic unit:

- Look ahead carries adders.
- Multiplication: Signed operand multiplication,
- Booths algorithm and array multiplier.
- Division and logic operations.
- Floating point arithmetic operation,
- Arithmetic & logic unit design.
- IEEE Standard for Floating Point Numbers

Course Objective

- Analysis of the design of arithmetic & logic unit and understanding of the fixed point and floating-point arithmetic operations.
- Study of various multiplication algorithm and division for signed no. like booth algorithm and others.

Course Outcome

- Analysis and design of arithmetic & logic unit and understanding of the fixed point and floating-point arithmetic operations.

CO-PO Mapping

COMPUTER ORGANIZATION AND ARCHITECTURE (KCS-302)												
CO.K	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
KCS-302.2	2	2	2	2	1	1	-	1	1	1	1	2

CO- PSO Mapping

COMPUTER ORGANIZATION AND ARCHITECTURE (KCS-302)				
CO.K	**PSO1**	**PSO2**	**PSO3**	**PSO4**
KCS-302.2	2	2	2	1

Prerequisite and Recap

- **Fundamental of computer**
- **Interconnection of computer**

UNIT 2

Arithmetic Logic Unit (ALU)

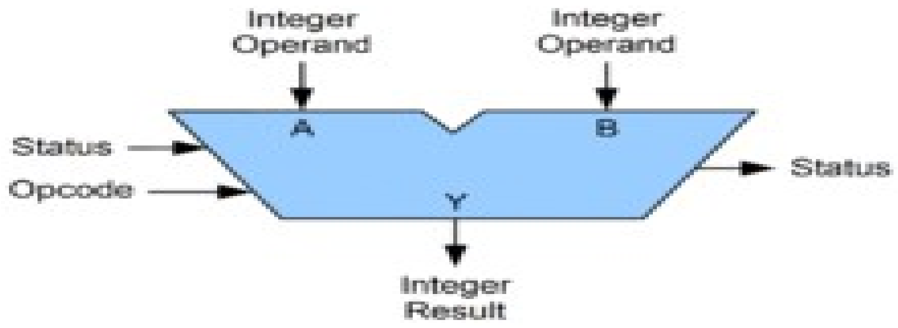
•An **arithmetic logic unit** (**ALU**) is a digital circuit used perform **arithmetic** and **logic** operations.

•It represents the fundamental building block of the central processing **unit** (CPU) of a computer. Modern CPUs contain very powerful and complex ALUs.

• In addition to ALUs, modern CPUs contain a **control unit** (CU)

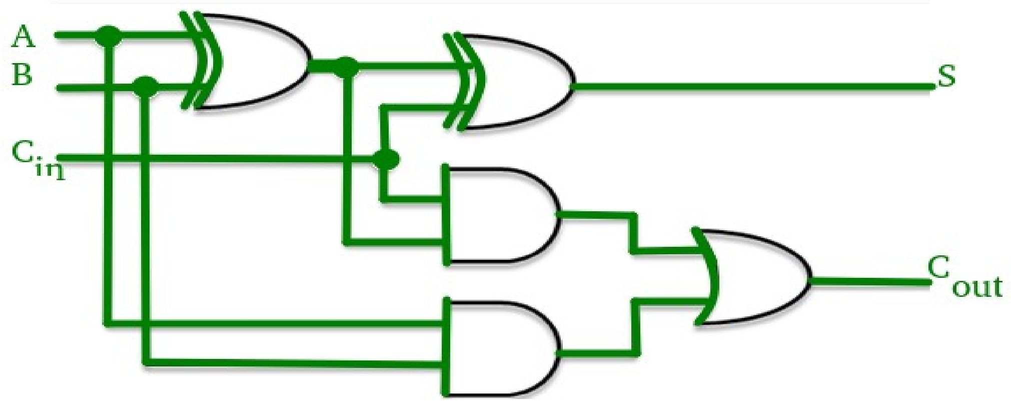
•The inputs to an ALU are the data to be operated on, called operands, and a code indicating the operation to be performed.

•A symbolic representation of an ALU and its input and output signals



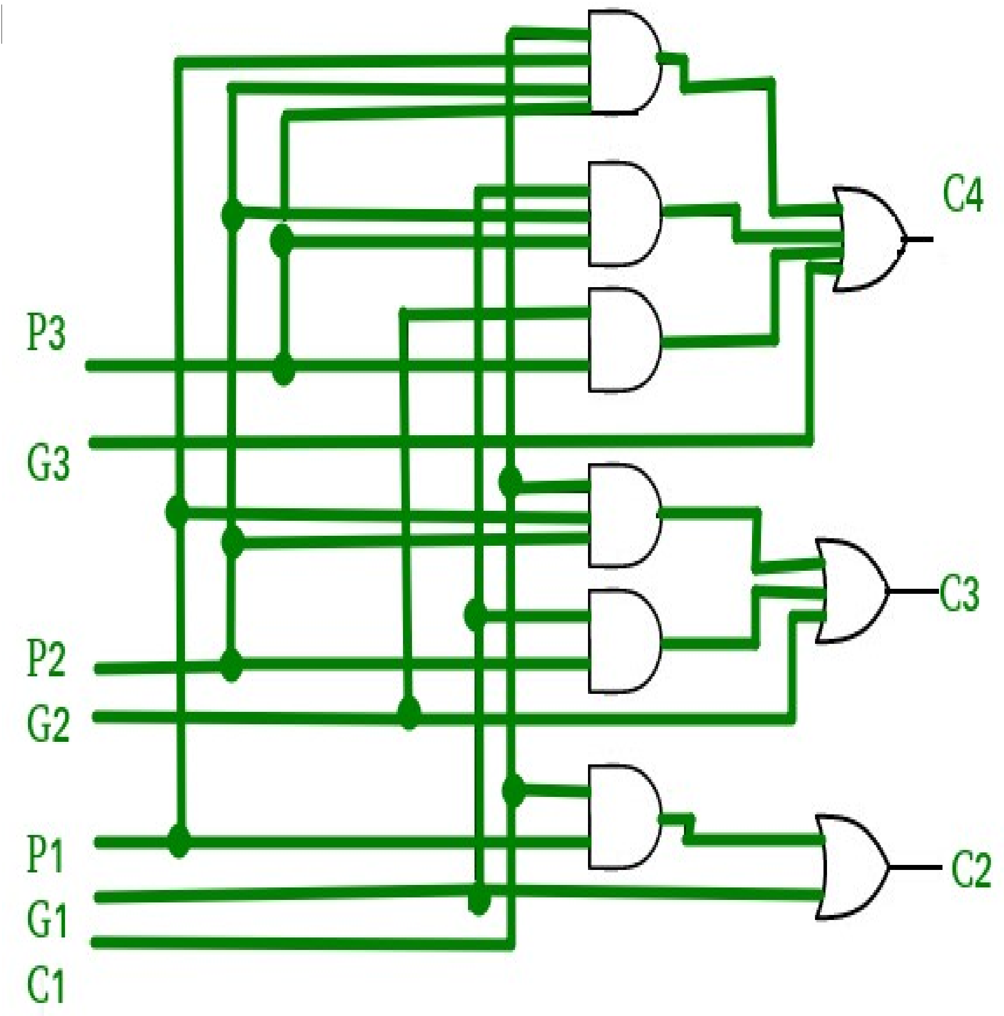
Carry Look ahead adder

- A carry-look ahead adder (CLA) or fast adder is a type of electronic adder used in digital logic. A carry-look ahead adder improves speed by reducing the amount of time required to determine carry bits.
- A carry look-ahead adder reduces the propagation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two-level logic.



Carry Look ahead adder

- The implementation of three Boolean functions for each carry output (C2,C3 and C4) for a carry look-ahead carry generator shown in diagram



Advantages –

•The propagation delay is reduced.

•It provides the fastest addition logic.

Disadvantages –

•The Carry Look-ahead adder circuit gets complicated as the number of variables increase.

•The circuit is costlier as it involves more number of hardware.

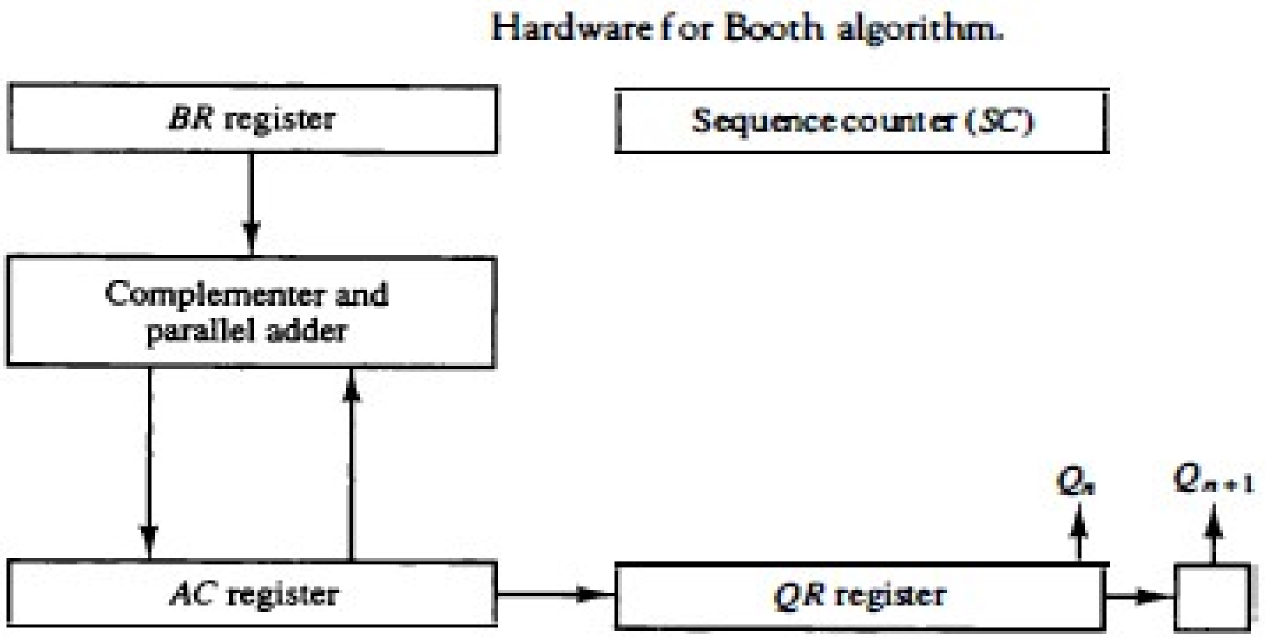
Multiplication: Signed operand multiplication

- **Multiplication** of two fixed point binary number in **signed** magnitude representation is done with **process** of successive shift and add operation.
- The numbers copied down in successive lines are shifted one position to the left from the previous number.
- Finally numbers are added and their sum form the product.

Booth Multiplication Algorithm

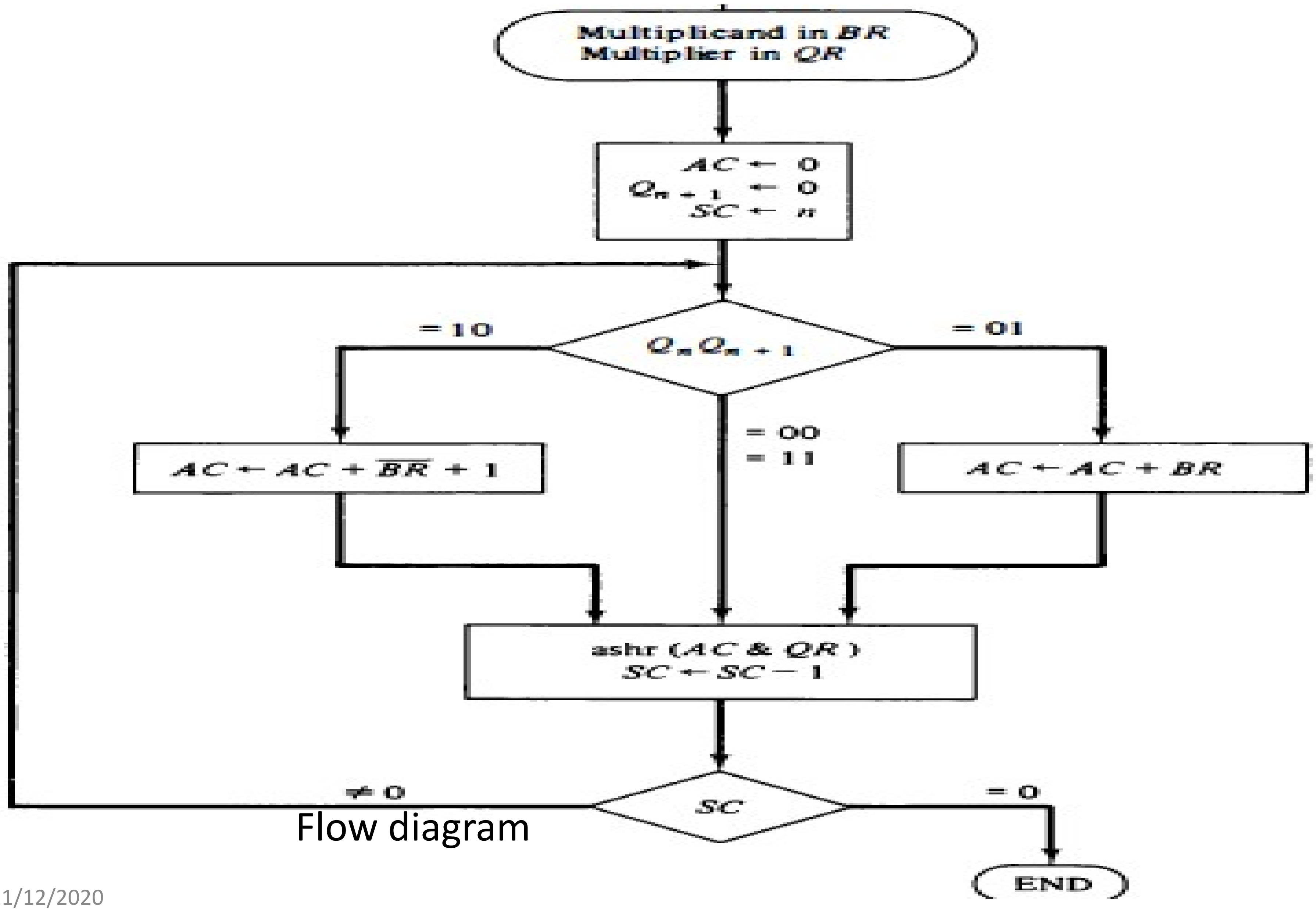
Booth Multiplication Method

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



Hardware for booth algorithm

Booth Multiplication Algorithm

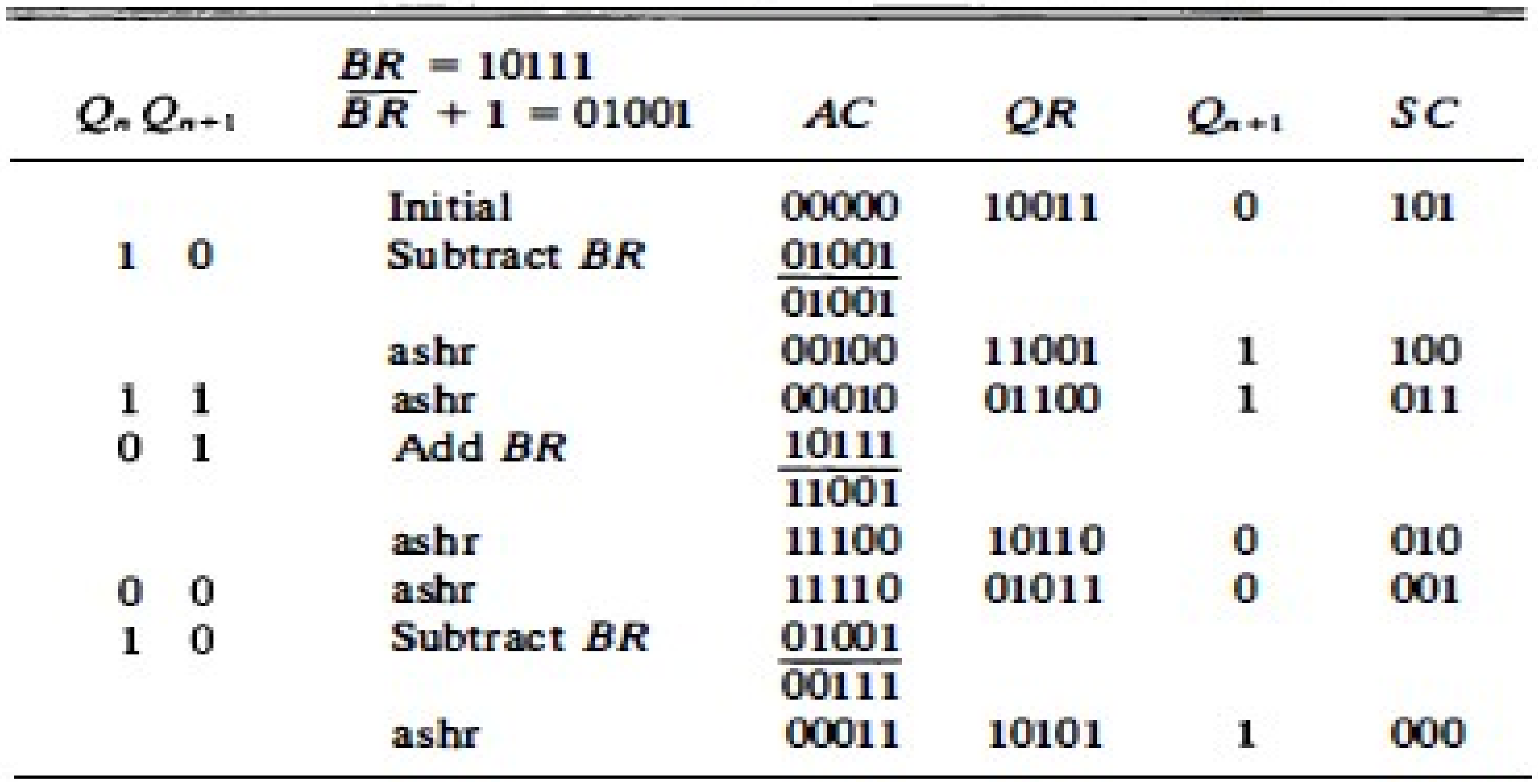


Flow diagram

Booth Multiplication Algorithm

Example

Example of Multiplication with Booth Algorithm

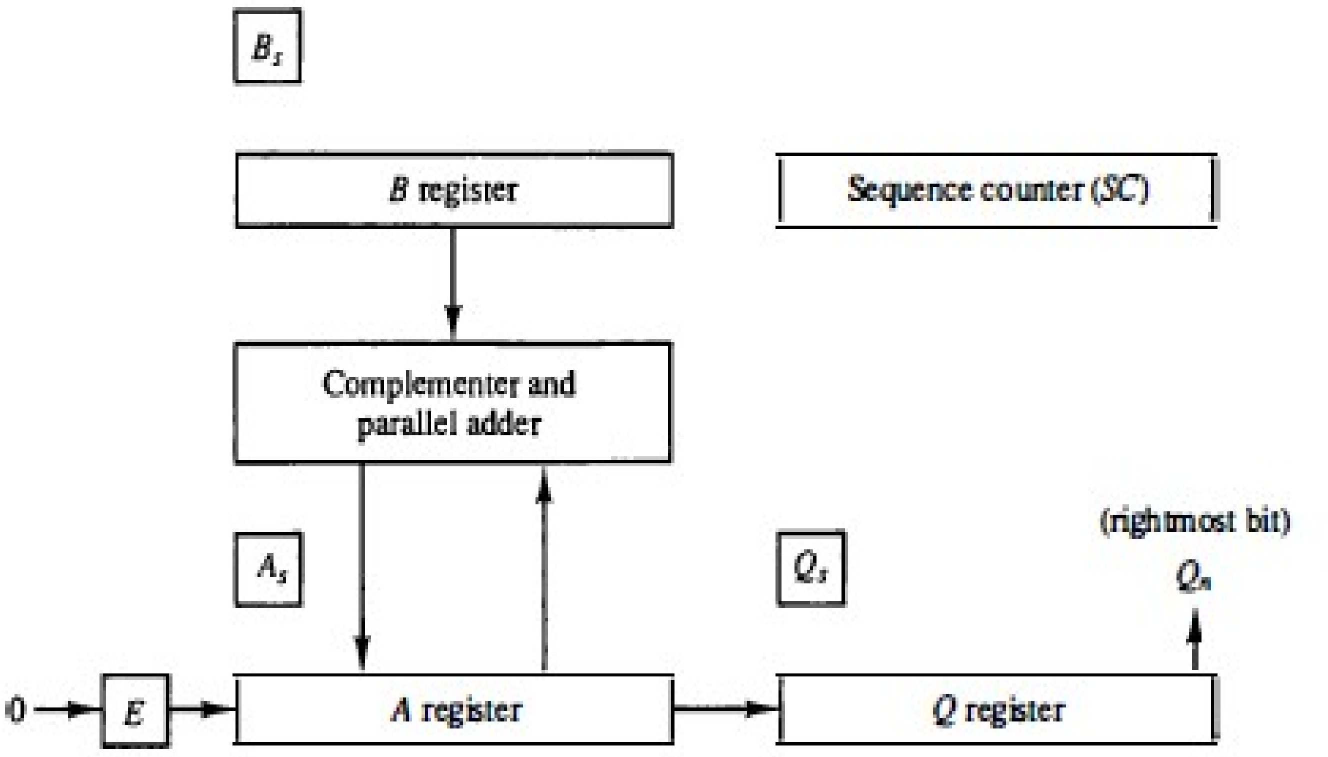


	$BR = 10111$				
$Q_n Q_{n+1}$	$\overline{BR} + 1 = 01001$	AC	QR	Q_{n+1}	SC
	Initial	00000	10011	0	101
1 0	Subtract BR	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	Add BR	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	Subtract BR	01001			
		00111			
	ashr	00011	10101	1	000

Multiplication for Signed magnitude data

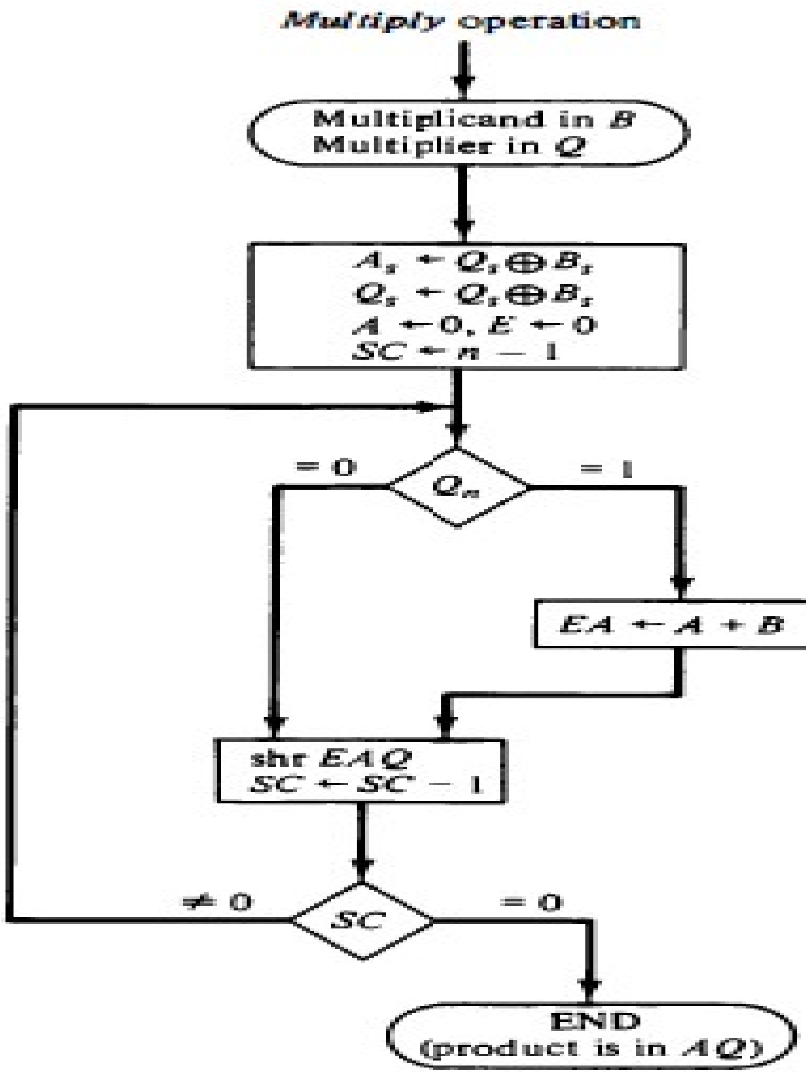
Hardware for multiply operation

Hardware for multiply operation.



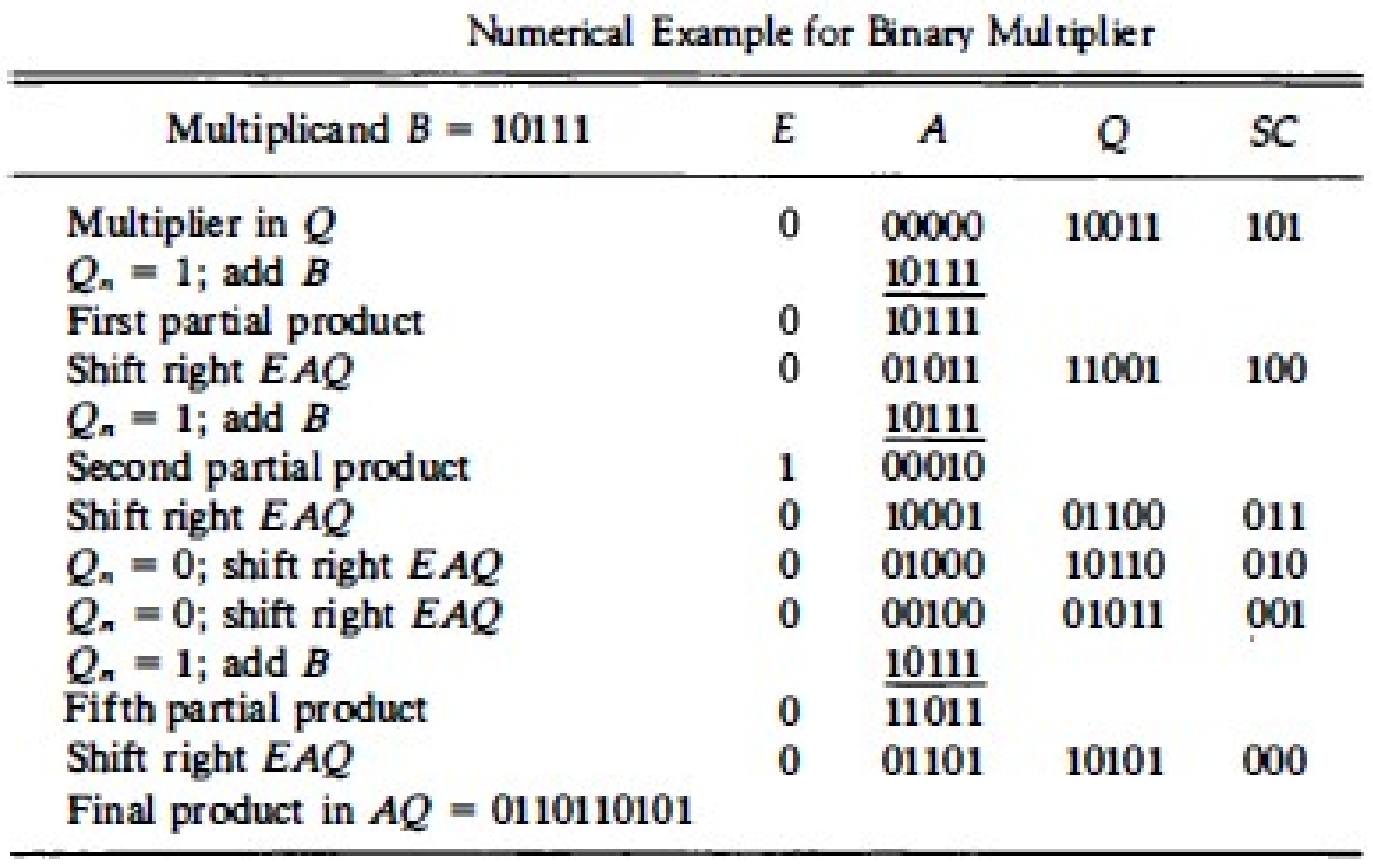
Multiplication for Signed magnitude data

Flowchart for multiply operation.



Multiplication for Signed magnitude data

Example



Numerical Example for Binary Multiplier

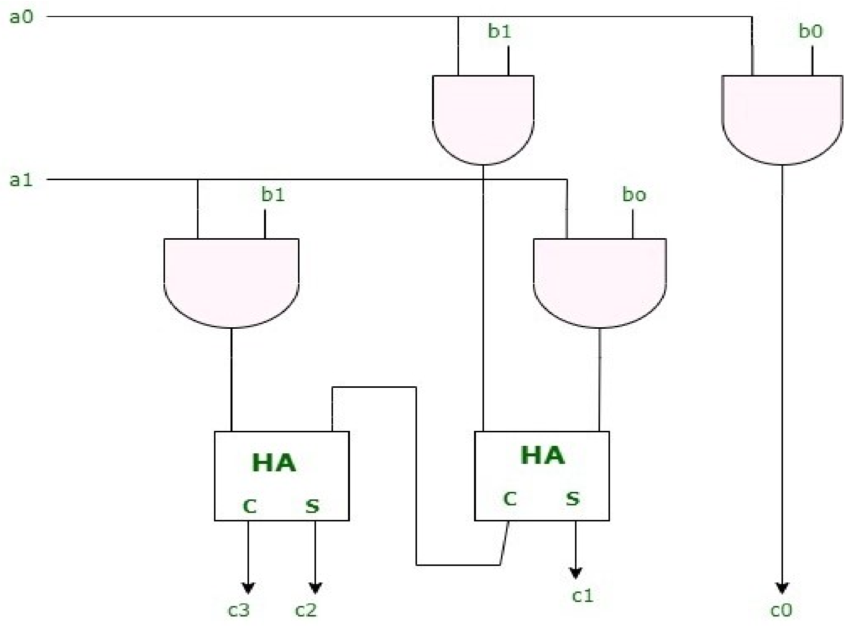
Multiplicand B = 10111	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
Q_n = 1; add B		$\underline{10111}$		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
Q_n = 1; add B		$\underline{10111}$		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
Q_n = 0; shift right EAQ	0	01000	10110	010
Q_n = 0; shift right EAQ	0	00100	01011	001
Q_n = 1; add B		$\underline{10111}$		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final product in AQ = 0110110101				

Array Multiplier

•An **array multiplier** is a digital combinational circuit used for multiplying two binary numbers by employing an array of full adders and half adders.

•This array is used for the nearly simultaneous addition of the various product terms involved.

•The multiplication of two 2-bit numbers as shown in figure. The multiplicand bits are b1 and b0, the multiplier bits are a1 and a0, and the product is -



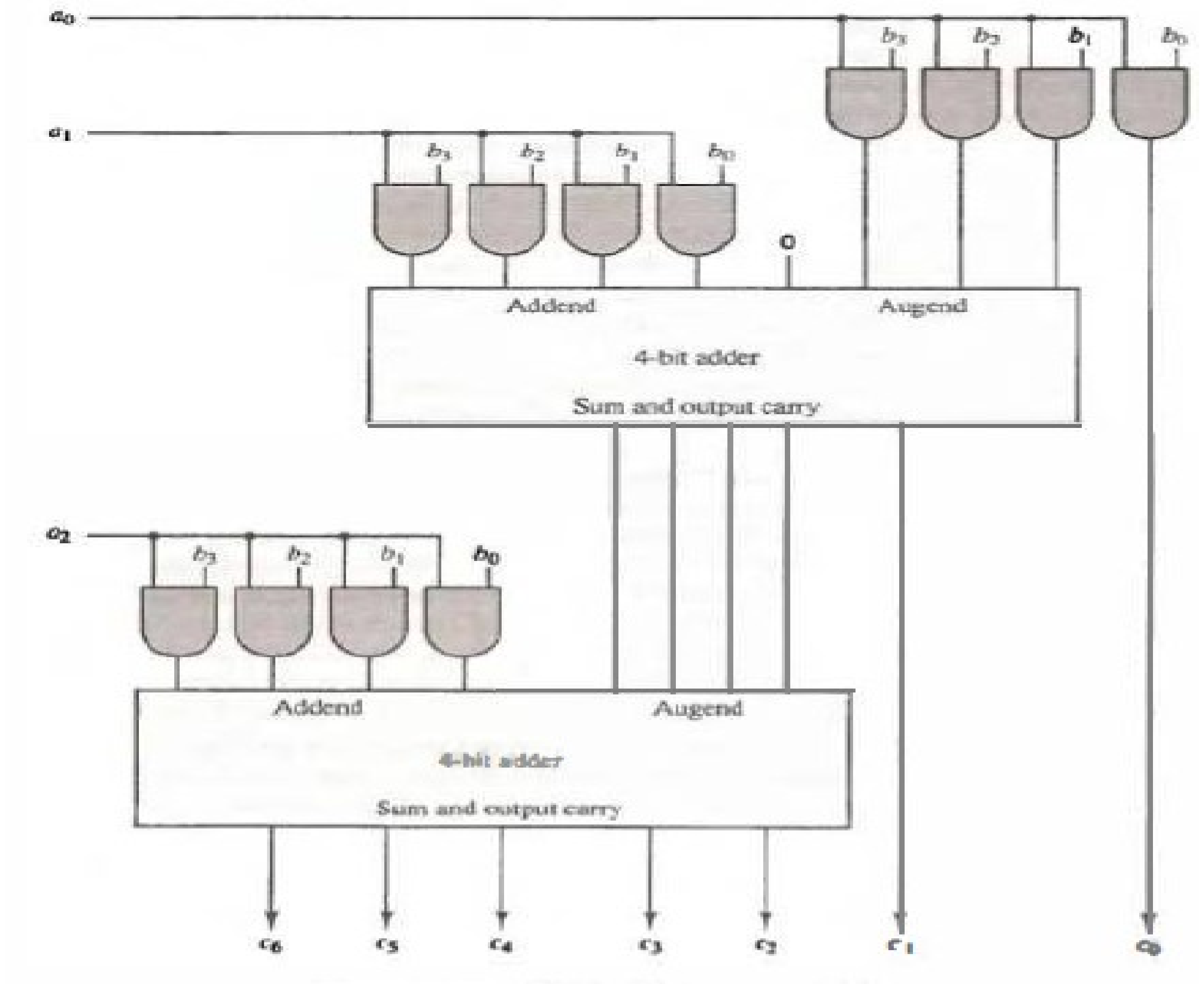
		b1	b0
		a1	a0
		a0b1	a0b0
	a1b1	a1b0	
c3	c2	c1	c0

Array Multiplier

- **4 bit Array multiplier B=b3 b2 b1 b0 and A= a2 a1 a0.**

For **j multipliers** &
k multiplicands bits ,
we needs

- **j x k AND gates**
- **(j-1)x k bit adders** produce
- A product of **j + k bits**

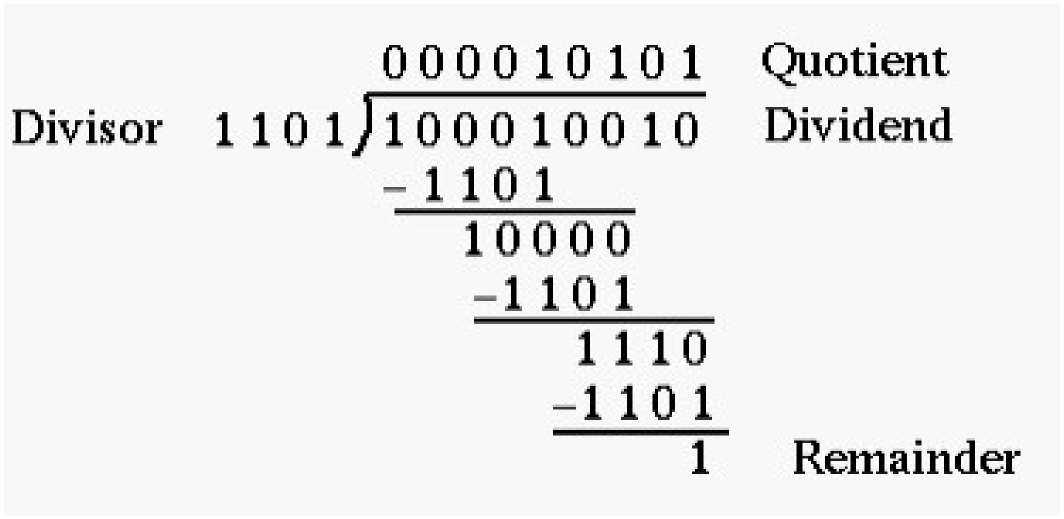


4-bit by 3-bit array multiplier.

➢Division and logic operations

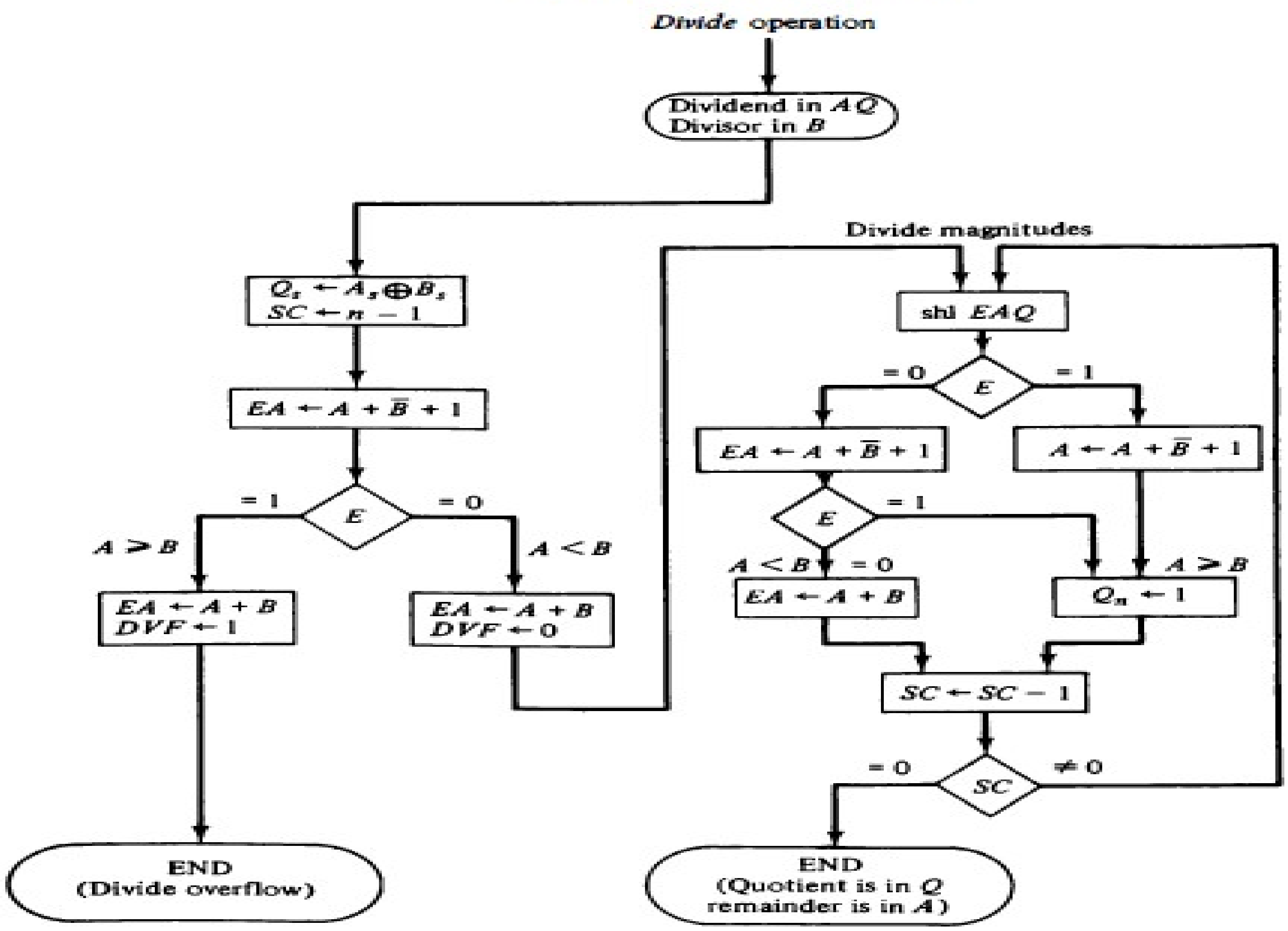
Division

•Division of two fixed-point binary numbers in signed magnitude representation is performed with paper and pencil by a process of successive compare, shift and subtract operations.
•Binary division is much simpler than decimal division because here the quotient digits are either 0 or 1 and there is no need to estimate how many times the dividend or partial remainder fits into the divisor. .



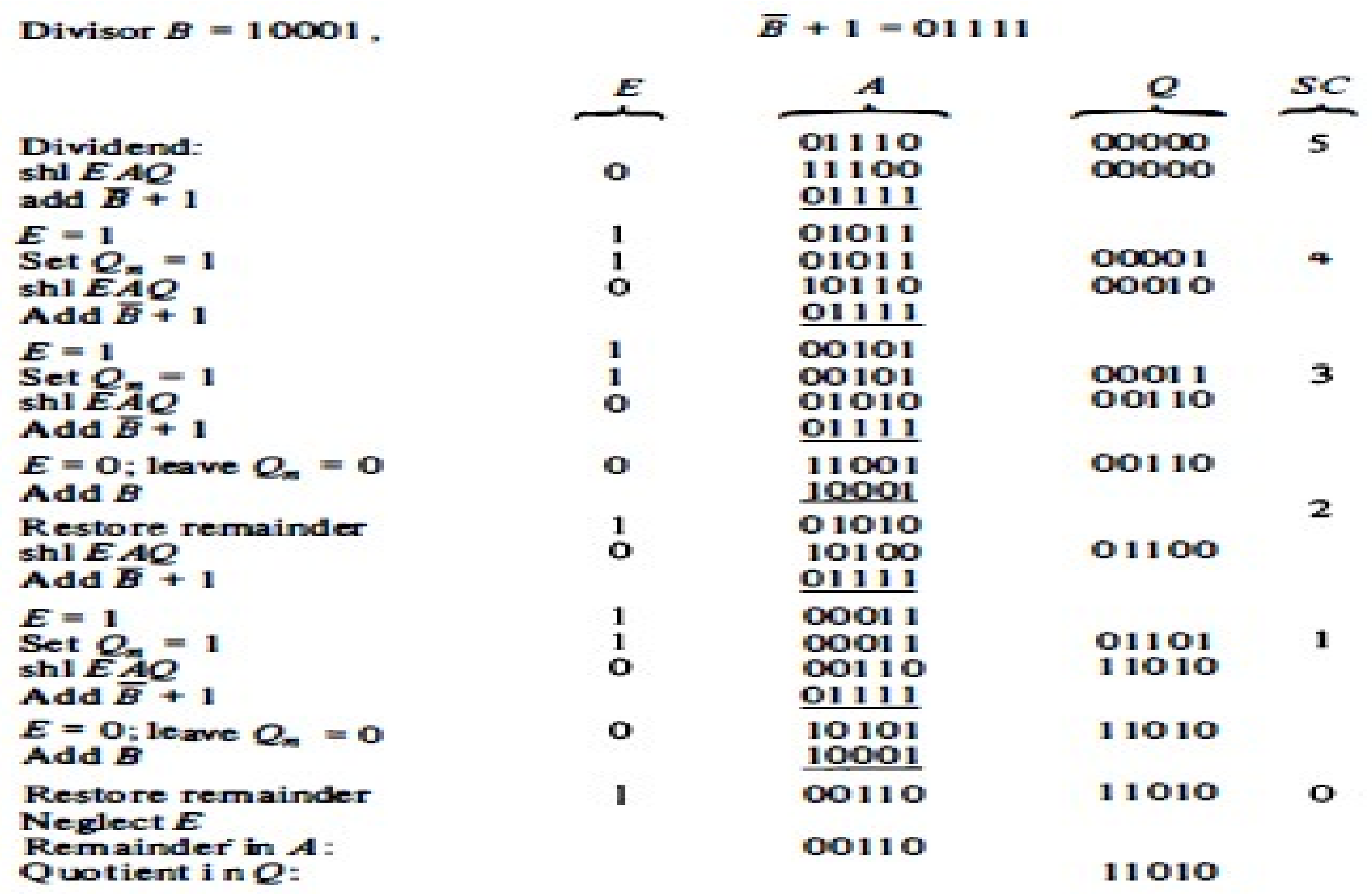
➢Division Algorithm

Flowchart for divide operation.



➢Division Algorithm

Example



Divisor B = 10001, $\bar{B} + 1$ = 01111

	E	A	Q	SC
Dividend:		01110	00000	5
shl EAQ	0	11100	00000	
add $\bar{B} + 1$		01111		
$E = 1$	1	01011		
Set $Q_n = 1$	1	01011	00001	4
shl EAQ	0	10110	00010	
Add $\bar{B} + 1$		01111		
$E = 1$	1	00101		
Set $Q_n = 1$	1	00101	00011	3
shl EAQ	0	01010	00110	
Add $\bar{B} + 1$		01111		
$E = 0$; leave $Q_n = 0$	0	11001	00110	
Add B		10001		
Restore remainder	1	01010		2
shl EAQ	0	10100	01100	
Add $\bar{B} + 1$		01111		
$E = 1$	1	00011		
Set $Q_n = 1$	1	00011	01101	1
shl EAQ	0	00110	11010	
Add $\bar{B} + 1$		01111		
$E = 0$; leave $Q_n = 0$	0	10101	11010	
Add B		10001		
Restore remainder	1	00110	11010	0
Neglect E				
Remainder in A:		00110		
Quotient in Q:			11010	

Example of binary division with digital hardware.

➢Floating point arithmetic operation

FLOATING POINT Addition & Subtraction

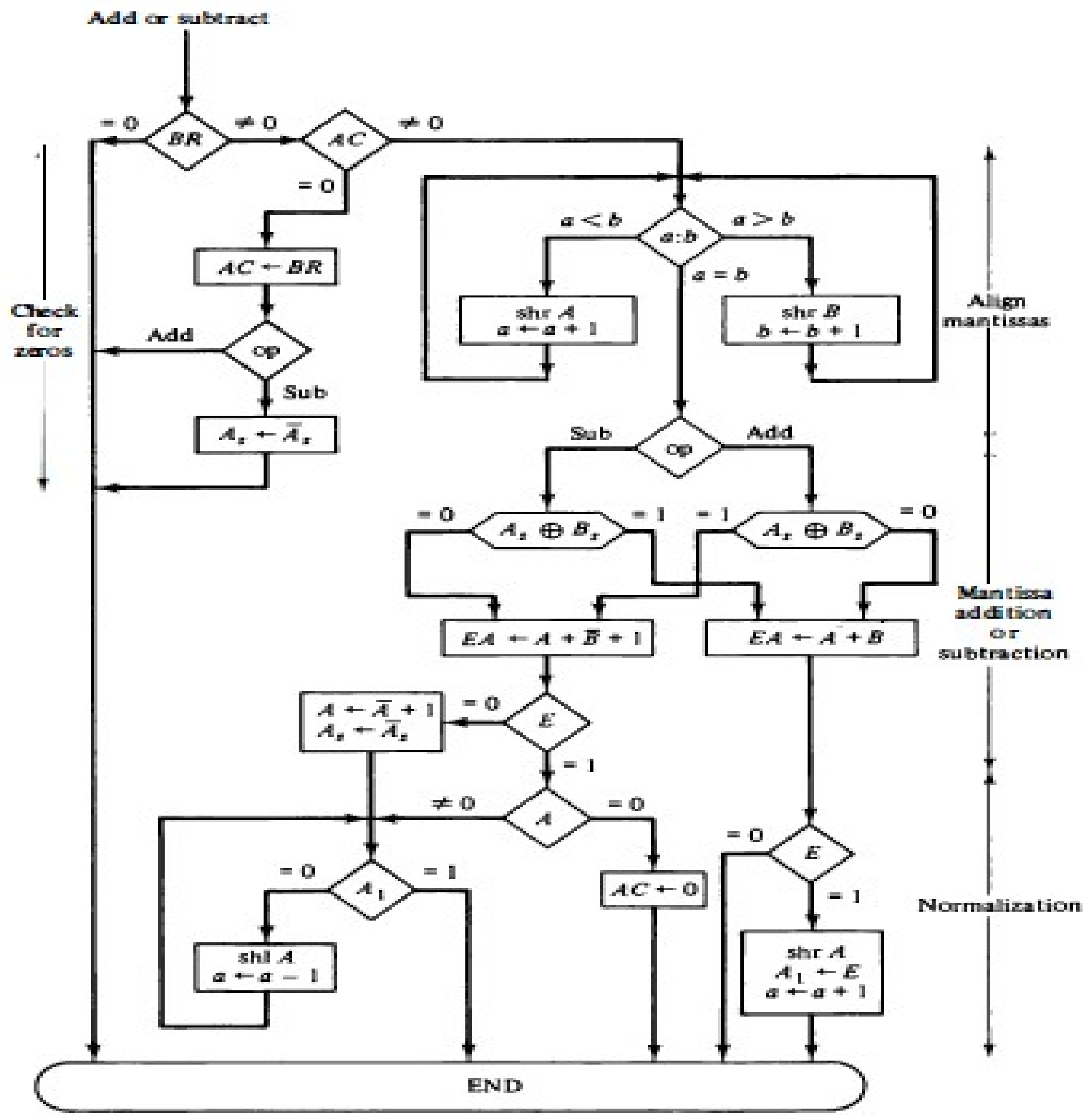
A floating- point number in computer registers consists of two parts: a mantissa m and an exponent e. The two parts represent a number obtained from multiplying m times a radix r raised to the value of e; thus $m \times r^e$

Example $.53725 \times 10^3$

During addition or subtraction, the two floating-point operands are in AC and BR. The sum or difference is formed in the AC. The algorithm can be divided into four consecutive parts:

- Check for zeros.
- Align the mantissas.
- Add or subtract the mantissas.
- Normalize the result.

FLOATING POINT Addition & Subtraction



➢Floating point arithmetic operation

FLOATING POINT Multiplication

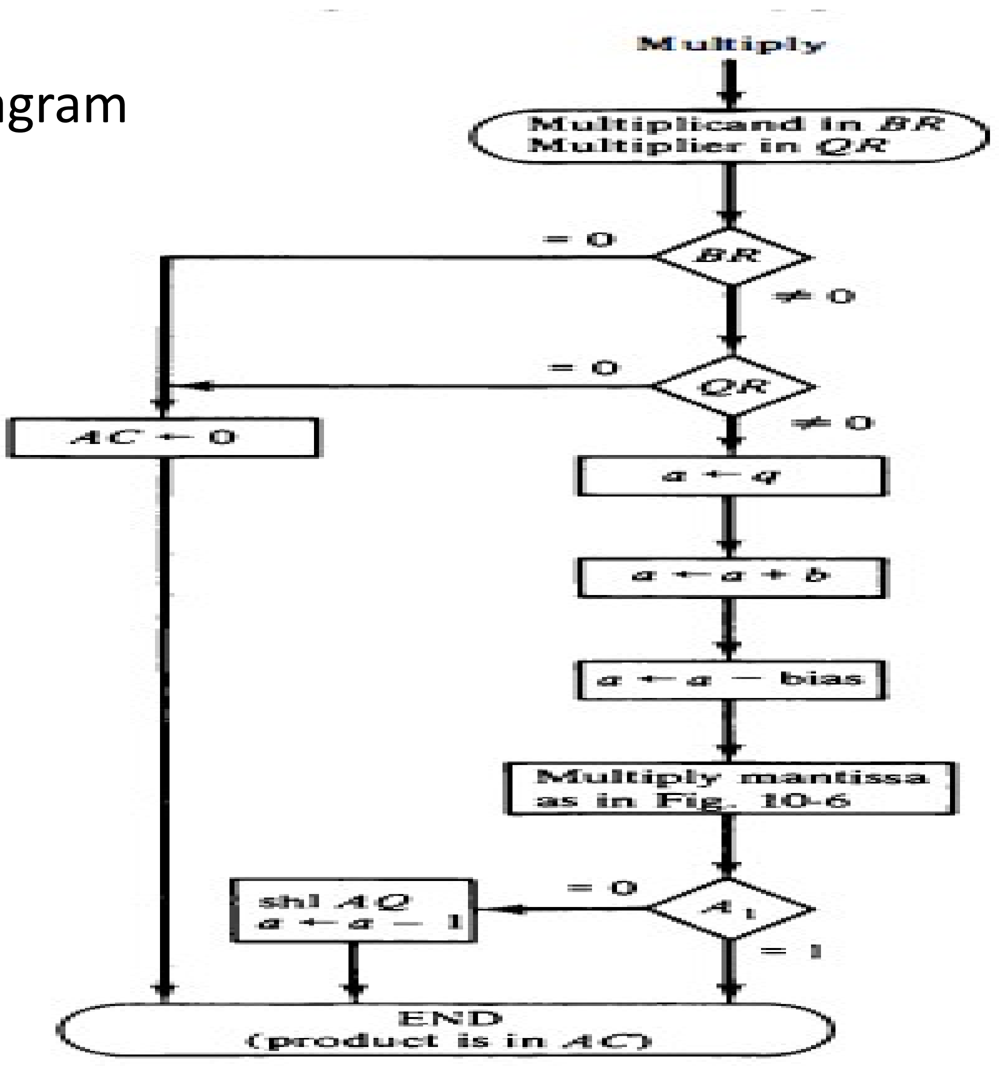
The multiplication of two floating-point numbers requires that we multiply the mantissas and add the exponents. The multiplication of the mantissas is performed in the same way as in fixed-point to provide a double-precision.

The multiplication algorithm can be subdivided into four parts:

- Check for zeros.
- Add the exponents.
- Multiply the mantissas.
- Normalize the product.

FLOATING POINT Multiplication

Flow diagram



➢Arithmetic & logic unit design

•Inside a computer, there is an Arithmetic Logic Unit (ALU), which is capable of performing logical operations (e.g. AND, OR, Ex-OR, Invert etc.)

•In addition to the arithmetic operations (e.g. Addition, Subtraction etc.).

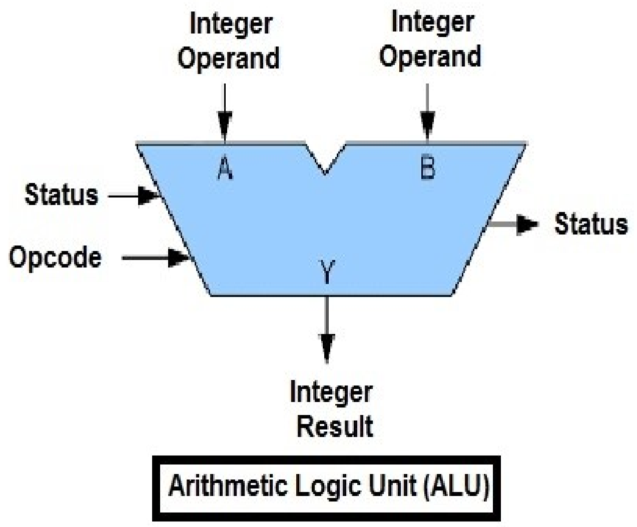
•The control unit supplies the data required by the ALU from memory, or from input devices, and directs the ALU to perform a specific operation based on the instruction fetched from the memory. ALU is the "calculator" portion of the computer.

Different operation as carried out by ALU can be categorized as follows

•**Logical operations**

•**Bit-Shifting Operations**

•**Arithmetic operations**



➢Arithmetic & logic unit design

1-bit ALU Design

•Construct a simple ALU that performs a arithmetic operation (1 bit addition)and does 3 logical operations namely AND, NOR and XOR as shown below.

•The multiplexer selects only one operation at a time. The operation selected depends on the selection lines of the multiplexer as shown in the truth table.

➢Arithmetic & logic unit design

1-bit ALU Design

Inputs		Outputs
M1	*M0*	*Operation*
0	0	SUM
1	0	AND
0	1	OR
1	1	XOR

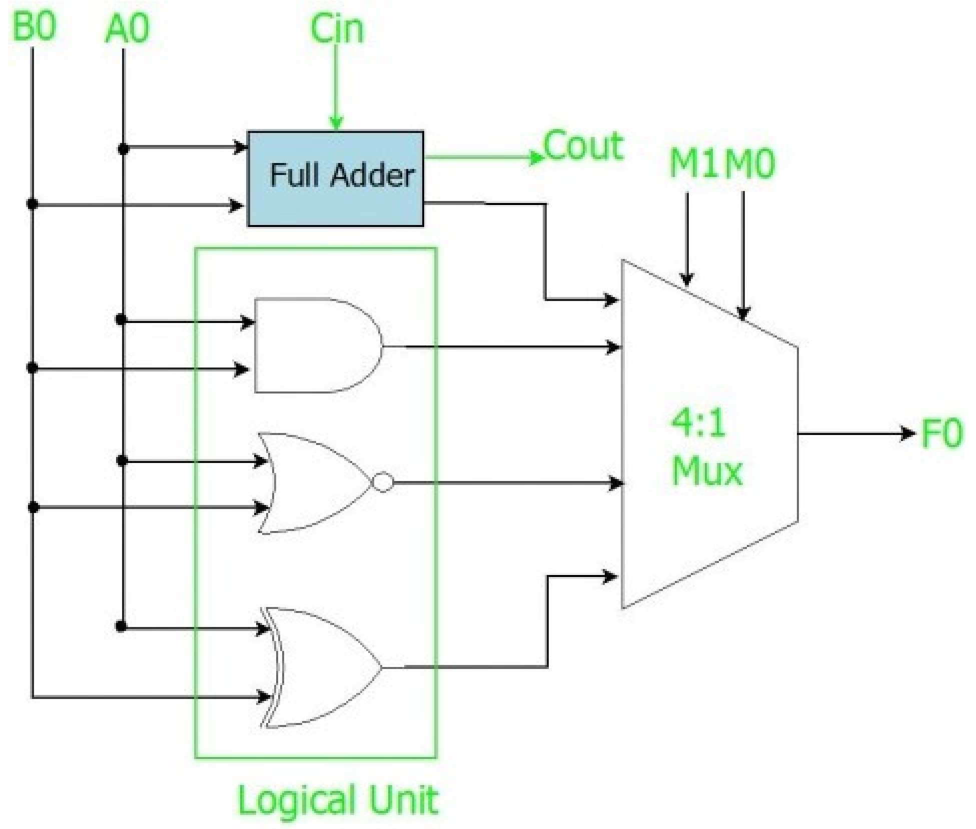


Figure: 1 bit ALU

➢IEEE Standard for Floating Point Numbers

•The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the **Institute of Electrical and Electronics Engineers (IEEE)**.

IEEE 754 has 3 basic components:

The Sign of Mantissa –

0 represents a positive number while 1 represents a negative number.

The Biased exponent –

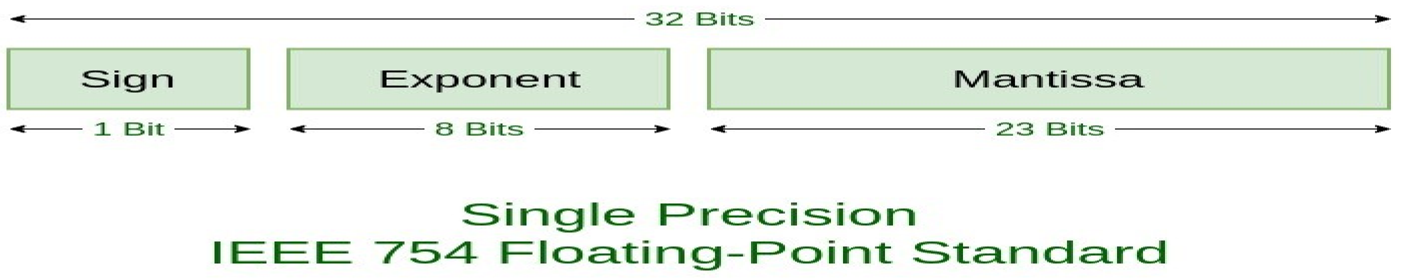
A bias is added to the actual exponent in order to get the stored exponent.

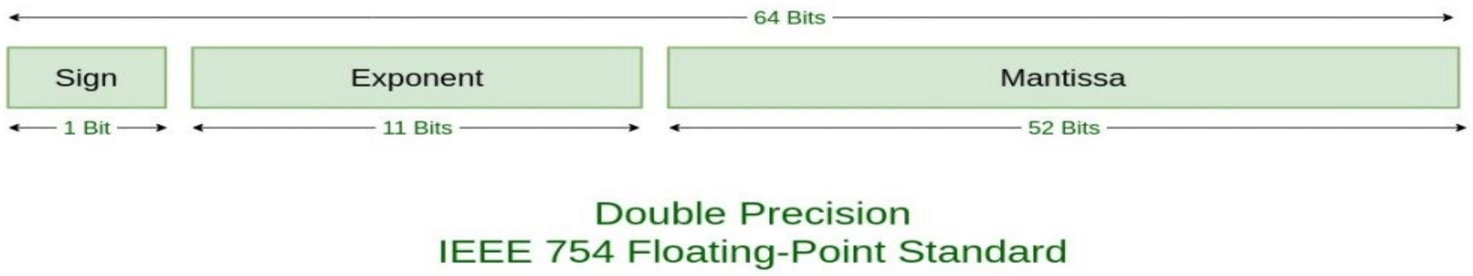
The Normalized Mantissa –

A normalized mantissa is one with only one 1 to the left of the decimal.

➢IEEE Standard for Floating Point Numbers

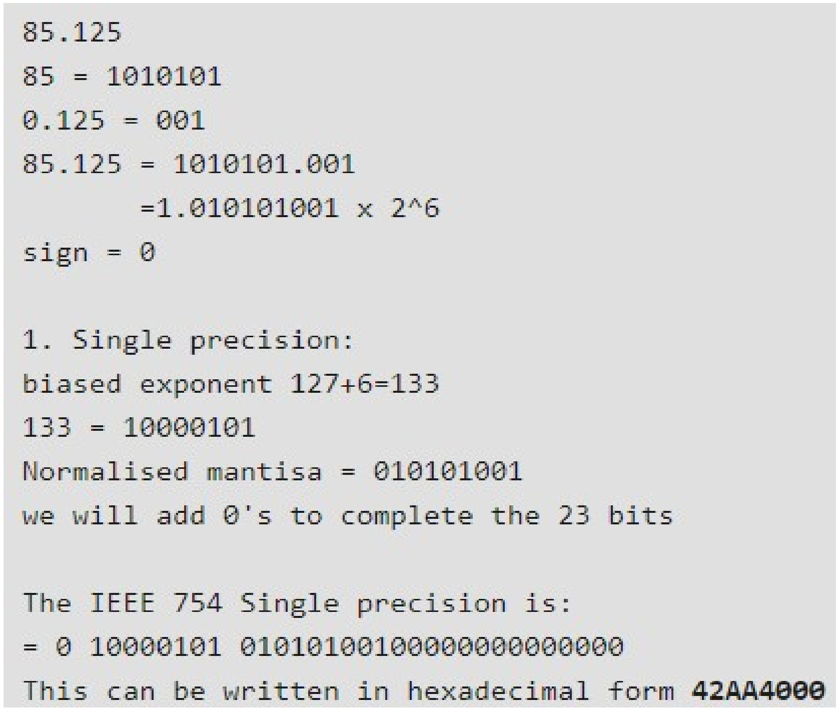
IEEE 754 numbers are divided into two based on the above three components: single precision and double precision.





➢IEEE Standard for Floating Point Numbers

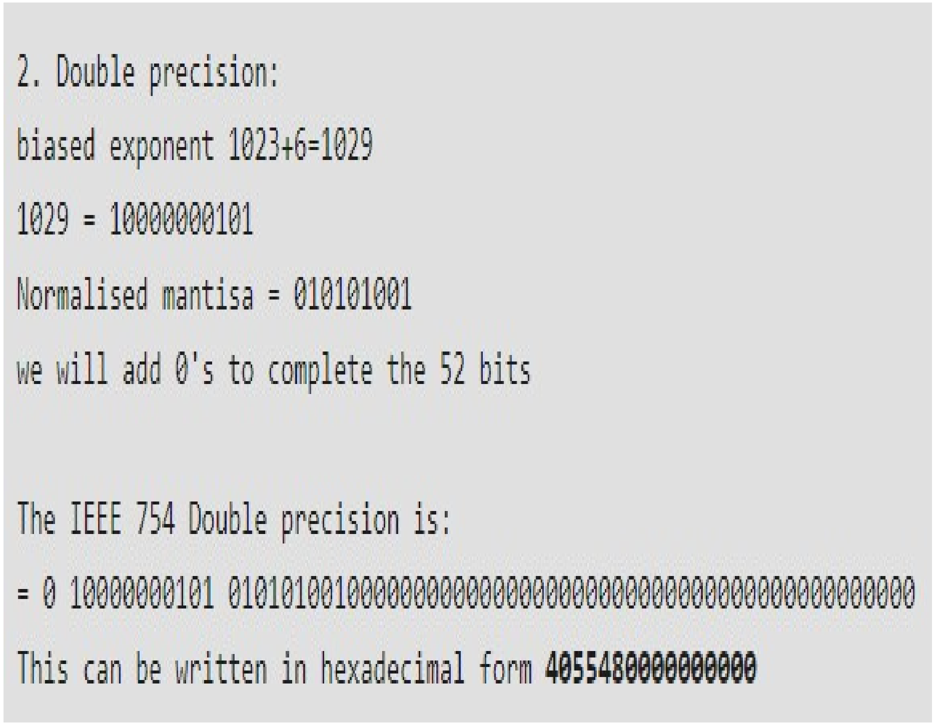
TYPES	SIGN	BIASED EXPONENT	NORMALISED MANTISA	BIAS
Single precision	1(31st bit)	8(30-23)	23(22-0)	127
Double precision	1(63rd bit)	11(62-52)	52(51-0)	1023



```
85.125
85 = 1010101
0.125 = 001
85.125 = 1010101.001
       =1.010101001 x 2^6
sign = 0

1. Single precision:
biased exponent 127+6=133
133 = 10000101
Normalised mantisa = 010101001
we will add 0's to complete the 23 bits

The IEEE 754 Single precision is:
= 0 10000101 01010100100000000000000
This can be written in hexadecimal form 42AA4000
```



```
2. Double precision:

biased exponent 1023+6=1029

1029 = 10000000101

Normalised mantisa = 010101001

we will add 0's to complete the 52 bits


The IEEE 754 Double precision is:

= 0 10000000101 0101010010000000000000000000000000000000000000000000

This can be written in hexadecimal form 4055480000000000
```

Faculty Video Links, You tube Courses Details

You tube/other Video Links

- https://www.youtube.com/watch?v=vcvgvqnH7GA
- https://www.youtube.com/watch?v=U62iP8RkZIk
- https://www.youtube.com/watch?v=6ToR6vuRb3M
- https://www.youtube.com/watch?v=9nkCLdhLDZk
- https://www.youtube.com/watch?v=0HiGruw9VcQ

Daily Quiz

- The result of 0 – 1 in binary is _______________?
- Which flag indicates the number of 1 bit that results from an operation?
- Define IEEE 754 Standard.
- Draw hardware diagram of booth algorithm.
- Draw array multiplier of 2 bit, A=a1a0, B=b1 b0 .

Weekly Assignment

1. Design array multiplier for b1 b0 X a1 a0 and b3 b2 b1 b0 X a2 a1 a0 and also write down the formula for no. of AND gate calculations.
2. Show the contents of registers E, A, Q and SC during the process of division 00001111 by 0011.
3. Show the hardware diagram & flow chart of Booth algorithm for multiplying two numbers.
4. Demonstrate the Cary Look Ahead adder with suitable diagram and advantages .
5. Define IEEE 754 standard for floating point representation with suitable example.

   $(23.175)_{10}$ b) $(0.6128)_{10}$ c) $(1000.010101)_2$

Expected Questions for University Exam

- Explain CLA and design 4bit CLA consists of 3 level of logic.

- Draw the flow chart for multiplying algorithm of sign magnitude data.
- Show the contents of registers E, A, Q & SC during the process of division of 10100011 by 1011
- Explain the IEEE 754 for floating point no. with examples
- Show the flow diagram & multiplication process using Booth's Algorithm for (-4) X (+3)

Summary

In previous slides we discuss in details

- Carry Look ahead adder.
- Multiplication: Signed operand multiplication,
- Booths algorithm and array multiplier.
- Division and logic operations.
- Floating point arithmetic operation,
- Arithmetic & logic unit design.
- IEEE Standard for Floating Point Numbers