Explain left shift and right shift algorithms for binary multiplications with examples.

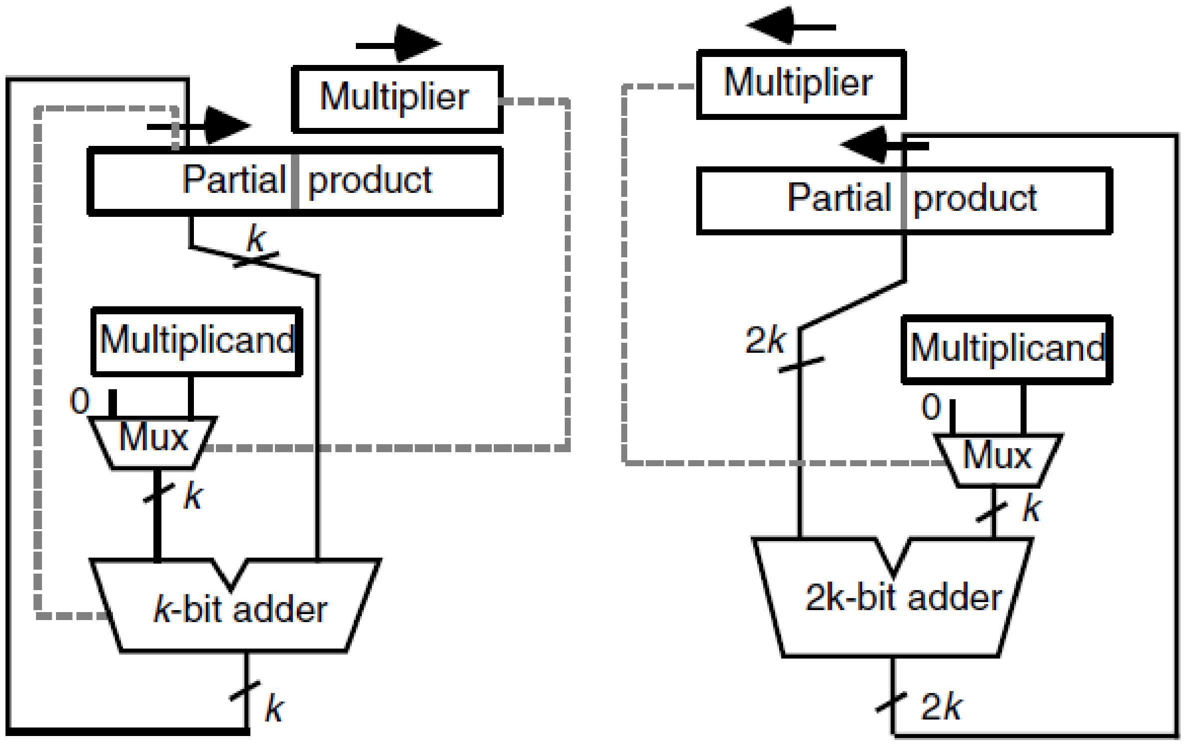
(a) Right-shift algorithm

a	1 0 1 0	
x	1 0 1 1	
$p^{(0)}$	0 0 0 0	
$+x_0a$	1 0 1 0	
$2p^{(1)}$	0 1 0 1 0	
$p^{(1)}$	0 1 0 1	0
$+x_1a$	1 0 1 0	
$2p^{(2)}$	0 1 1 1 1	0
$p^{(2)}$	0 1 1 1	1 0
$+x_2a$	0 0 0 0	
$2p^{(3)}$	0 0 1 1 1	1 0
$p^{(3)}$	0 0 1 1	1 1 0
$+x_3a$	1 0 1 0	
$2p^{(4)}$	0 1 1 0 1	1 1 0
$p^{(4)}$	0 1 1 0	1 1 1 0

(b) Left-shift algorithm

a		1 0 1 0
x		1 0 1 1
$p^{(0)}$		0 0 0 0
$2p^{(0)}$	0	0 0 0 0
$+x_3a$		1 0 1 0
$p^{(1)}$	0	1 0 1 0
$2p^{(1)}$	0 1	0 1 0 0
$+x_2a$		0 0 0 0
$p^{(2)}$	0 1	0 1 0 0
$2p^{(2)}$	0 1 0	1 0 0 0
$+x_1a$		1 0 1 0
$p^{(3)}$	0 1 1	0 0 1 0
$2p^{(3)}$	0 1 1 0	0 1 0 0
$+x_0a$		1 0 1 0
$p^{(4)}$	0 1 1 0	1 1 1 0

Show the block diagram implementation of left shift and right shift algorithms for binary multiplication.



(a) Right shift (b) Left shift

Why right shift algorithm is preferred in binary multiplication?

Multiplication with right shifts is the preferred method because it requires a k-bit adder contrary to 2k-bits adder in case of left shift algorithm.

State the limitations of left shift and right shift algorithms. How many ADD & SHIFT operations are required in each algorithms?

If multiplier or multiplicand or both are negative numbers, could these algorithms be used for multiplications? Explain/justify your answer with examples.

What is Booth's algorithm? Explain multiplication of signed numbers using Booth's algorithm. How does Booth's algorithm save computations compared to conventional binary multiplication technique?

Explain how does Booth's algorithm save computations compared to conventional binary multiplication technique?

If multiplier of a 16-bit multiplication is 1000111110110011, how many ADD, SUBTRACT and SHIFT operations are required with Booth's algorithm? Compare this against those of left shift and right shift algorithms.

Design a multiplier of signed numbers using Booth's algorithm.

Explain basic observation of Booth's algorithm.

- Booth observed that whenever there are a large number of consecutive 1s in *x*, multiplication can be speeded up by replacing the corresponding sequence of additions with a subtraction at the least-significant end and an addition in the position immediately to the left of its most-significant end. In other words
- The longer the sequence of 1s, the larger the savings achieved.
- The effect of this transformation is to change the binary number *x* with digit set [0, 1] to the binary signed digit number *y* using the digit set [−1, 1].

$$2^j + 2^{j-1} + \cdots + 2^{i+1} + 2^i = 2^{j+1} - 2^i$$

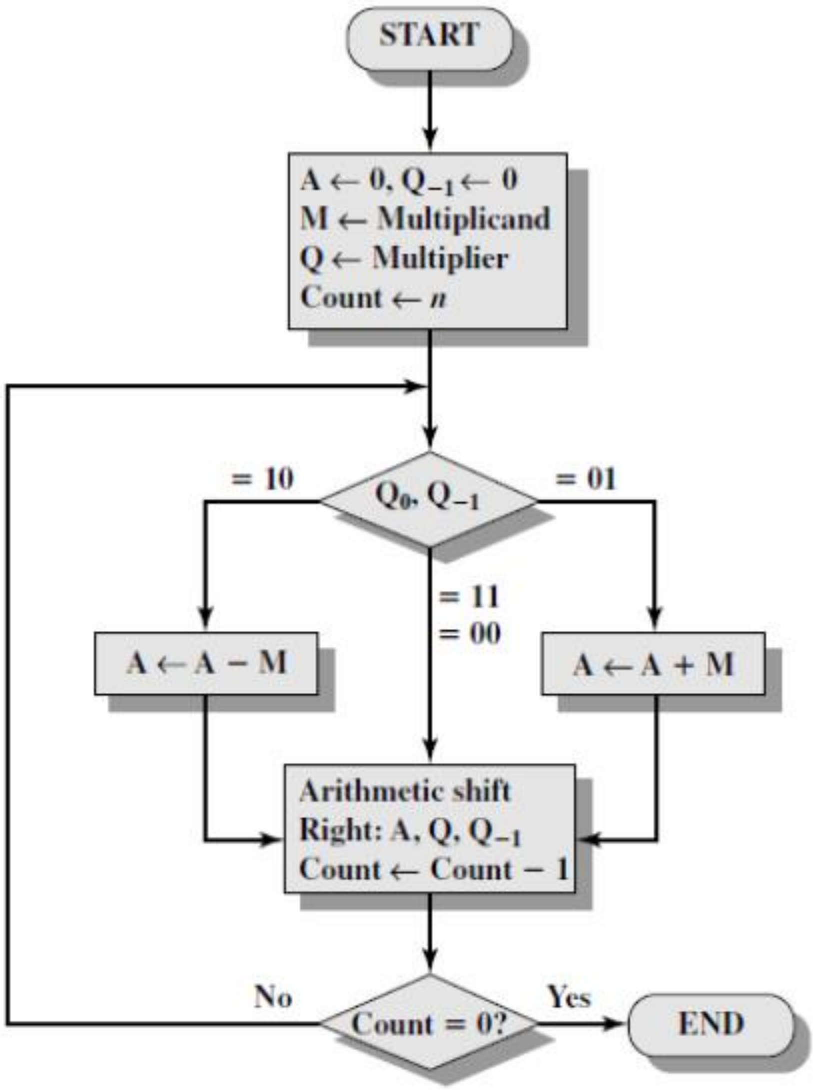
What is Booth's recording?

To change the binary digits, [0, 1] of multiplier to the binary signed digit [−1, 1] as follows

Radix-2 Booth's recoding.

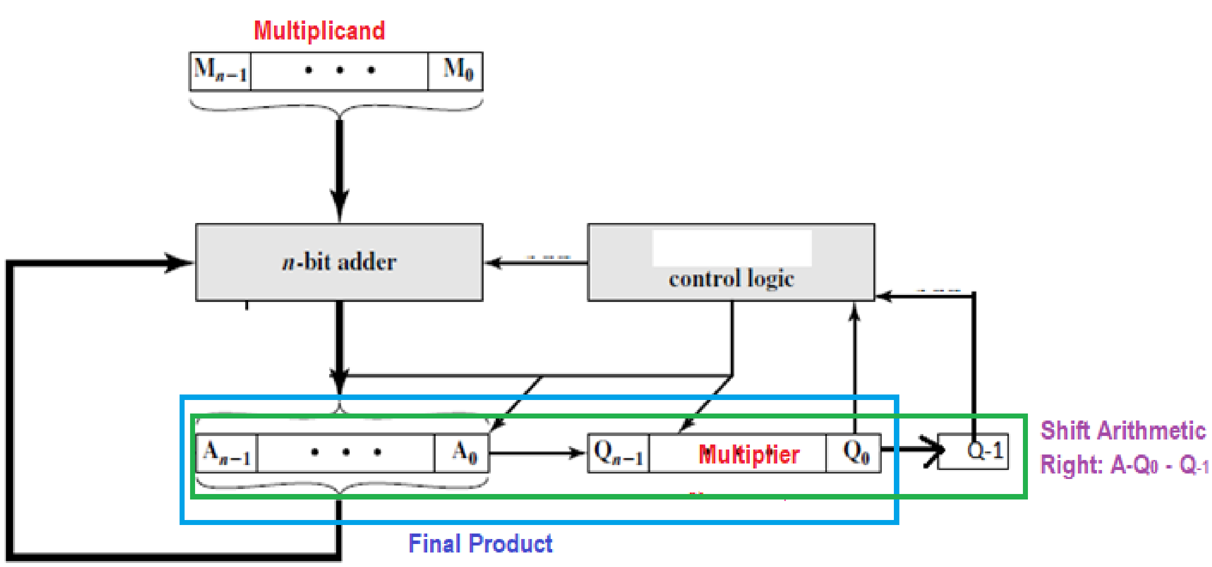
x_i	x_{i-1}	y_i	**Explanation**
0	0	0	No string of 1s in sight
0	1	1	End of string of 1s in x
1	0	$\bar{1}$	Beginning of string of 1s in x
1	1	0	Continuation of string of 1s in x

Show the flowchart of Booth’s algorithm



Booth’s Algorithm for Twos Complement Multiplication

Show a hardware implementation for Booth’s algorithm



Explain Booth's algorithm with an example

a	1 0 1 1 0	
x	1 0 1 0 1	Multiplier
y	-1 1 -1 1 -1	Booth-recoded
$p^{(0)}$	0 0 0 0 0	
$+y_0a$	0 1 0 1 0	
$2p^{(1)}$	0 0 1 0 1 0	
$p^{(1)}$	0 0 1 0 1	0
$+y_1a$	1 0 1 1 0	
$2p^{(2)}$	1 1 1 0 1 1	0
$p^{(2)}$	1 1 1 0 1	1 0
$+y_2a$	0 1 0 1 0	
$2p^{(3)}$	0 0 0 1 1 1	1 0
$p^{(3)}$	0 0 0 1 1	1 1 0
$+y_3a$	1 0 1 1 0	
$2p^{(4)}$	1 1 1 0 0 1	1 1 0
$p^{(4)}$	1 1 1 0 0	1 1 1 0
$+y_4a$	0 1 0 1 0	
$2p^{(5)}$	0 0 0 1 1 0	1 1 1 0
$p^{(5)}$	0 0 0 1 1	0 1 1 1 0

Example: 7 X 3 using Booth's algorithm

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial values	
1001	0011	0	0111	A ← A − M	First cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second cycle
0101	0100	1	0111	A ← A + M	Third cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth cycle

Example: 7 x (-6) using Booth's algorithm

CYCLE	OPERATIONS	Content of A	Content of Q	Q_{-1}	Comments, if any	Content of M
Initial Value	**Initialization**	**0000**	**1010**	**0**	Q_0Q_{-1}**=00**	
Cycle-1						
	Arithmetic Shift right (A Q Q_{-1})	**0000**	**0101**	**0**	Q_0Q_{-1}**=10**	
Cycle-2	**A= A-M**	**1001**	**0101**	**0**	Q_0Q_{-1}**=10**	
	Shift right (A Q Q_{-1})	**1100**	**1010**	**1**	Q_0Q_{-1}**=01**	
Cycle-3	**A=A+M**	**0011**	**1010**	**1**	Q_0Q_{-1}**=01**	**0111**
	Shift right (A Q Q_{-1})	**0001**	**1101**	**0**	Q_0Q_{-1}**=10**	
Cycle-4	**A=A-M**	**1010**	**1101**	**0**	Q_0Q_{-1}**=10**	
	Shift right (A Q Q_{-1})	**1101**	**0110**	**1**	Q_0Q_{-1}**=01**	
Product is in: AQ pair: 11010110 in 2's complement since MSB is 1, to get magnitude, take 2's complement of AQ: 00101001 + 1 = 00101010 = 42 Answer: - 42						

What is floating point representation? What is IEEE 754 standard for floating point representation?

IEEE 754 uses a bias of 127 for single precision, so an exponent of -1 is represented by the bit pattern of the value $-1 + 127_{ten}$, or $126_{ten} = 0111\ 1110_{two}$, and $+1$ is represented by $1 + 127$, or $128_{ten} = 1000\ 0000_{two}$. The exponent bias for double precision is 1023. Biased exponent means that the value represented by a floating-point number is really

$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

The range of single precision numbers is then from as small as

$$\pm 1.00000000000000000000000_{two} \times 2^{-126}$$

to as large as

$$\pm 1.11111111111111111111111_{two} \times 2^{+127}.$$

- Consider the number F = -3.75

  $-3.75_{10} = -11.11_2 = -1.111 \times 2^1$

- Mantissa will be stored as: $M = 11100000000000000000000_2$

- Here, EXP = 1, BIAS = 127. ➔ $E = 1 + 127 = 128 = 10000000_2$

1	10000000	11100000000000000000000

40700000 in hex

Floating-Point Representation

EXAMPLE

Show the IEEE 754 binary representation of the number -0.75_{ten} in single and double precision.

ANSWER

The number -0.75_{ten} is also

$$-3/4_{ten} \text{ or } -3/2^2_{ten}$$

It is also represented by the binary fraction

$$-11_{two}/2^2_{ten} \text{ or } -0.11_{two}$$

In scientific notation, the value is

$$-0.11_{two} \times 2^0$$

and in normalized scientific notation, it is

$$-1.1_{two} \times 2^{-1}$$

The general representation for a single precision number is

$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - 127)}$$

Subtracting the bias 127 from the exponent of $-1.1_{two} \times 2^{-1}$ yields

$$(-1)^1 \times (1 + .1000\ 0000\ 0000\ 0000\ 0000\ 000_{two}) \times 2^{(126-127)}$$

The single precision binary representation of -0.75_{ten} is then

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 bit 8 bits 23 bits

The double precision representation is

$$(-1)^1 \times (1 + .1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two}) \times 2^{(1022-1023)}$$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 bit 11 bits 20 bits

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

32 bits

EXAMPLE

Converting Binary to Decimal Floating Point

What decimal number is represented by this single precision float?

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.	.	.

ANSWER

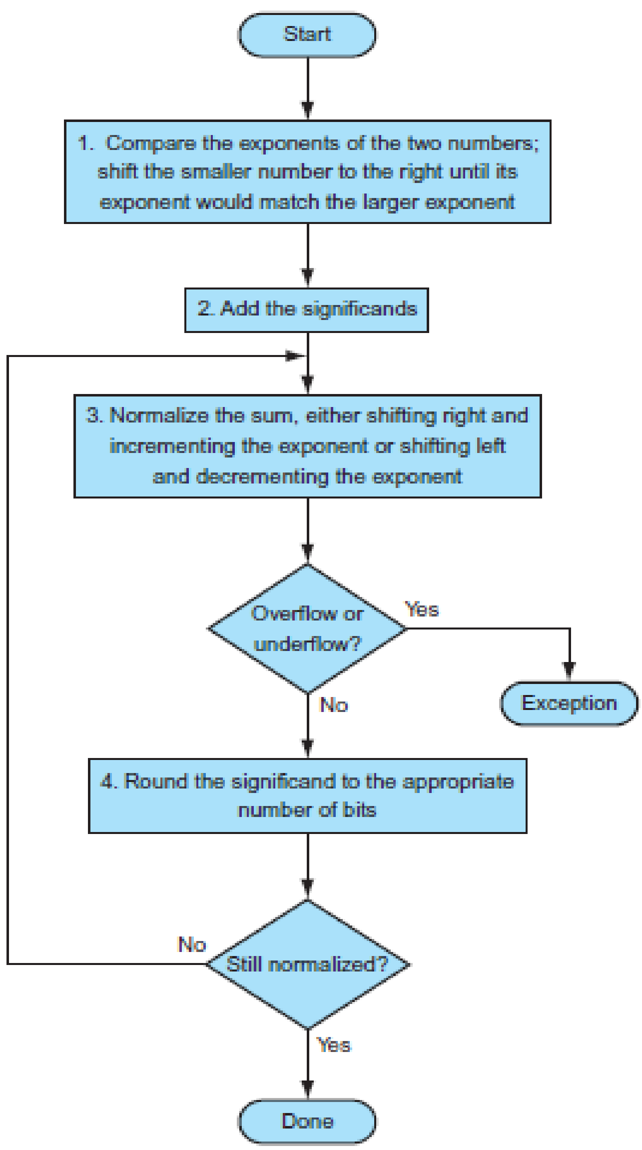
The sign bit is 1, the exponent field contains 129, and the fraction field contains $1 \times 2^{-2} = 1/4$, or 0.25. Using the basic equation,

$$
\begin{aligned}
(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent}-\text{Bias})} &= (-1)^1 \times (1 + 0.25) \times 2^{(129-127)} \\
&= -1 \times 1.25 \times 2^2 \\
&= -1.25 \times 4 \\
&= -5.0
\end{aligned}
$$

Convert -15.552 in IEEE 754 format.

Convert the following IEEE 754 binary bit pattern to decimal:
1 00111100 11101000000000000000000

Flowchart

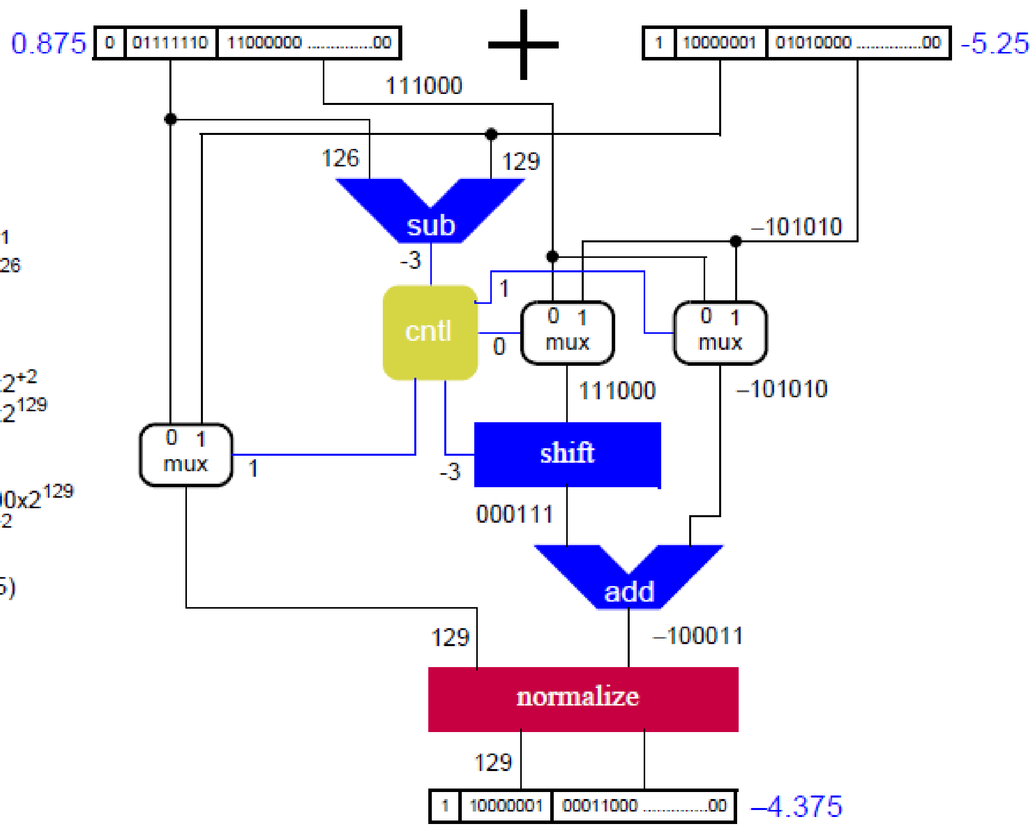


Show the hardware implementation of a floating point adder/subtractor and explain the steps.

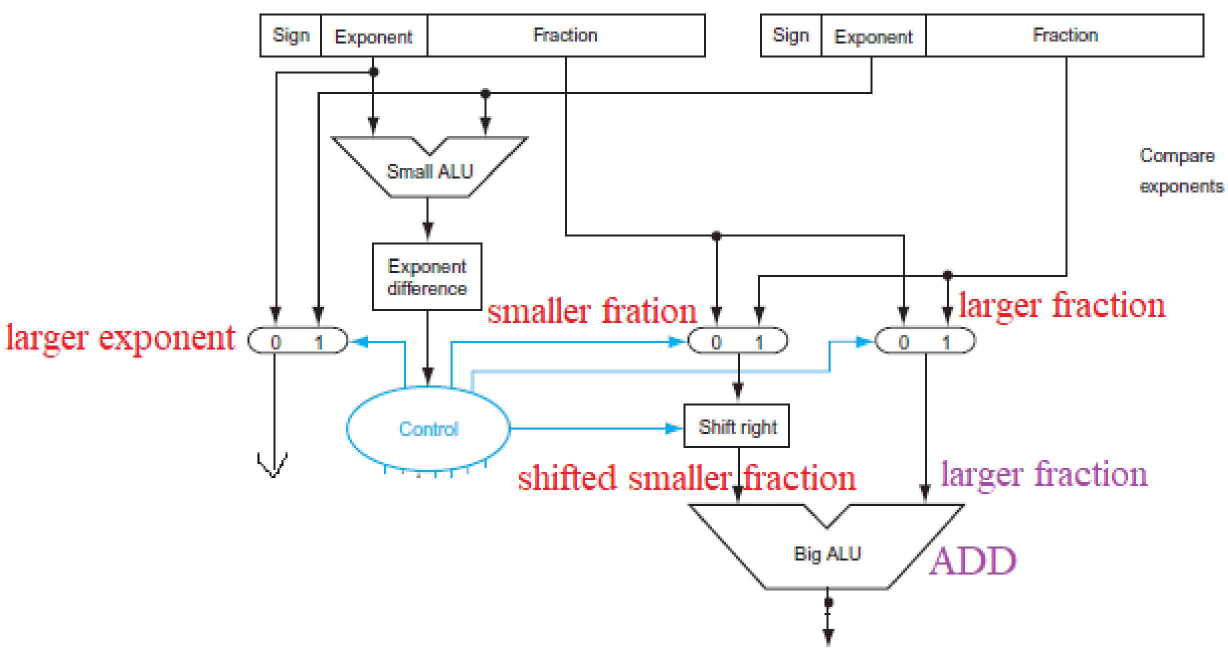
0.875
$= (-1)^0 \text{x} 0.111$
$= (-1)^0 \text{x} 1.110 \text{x} 2^{-1}$
$= (-1)^0 \text{x} 1.110 \text{x} 2^{126}$

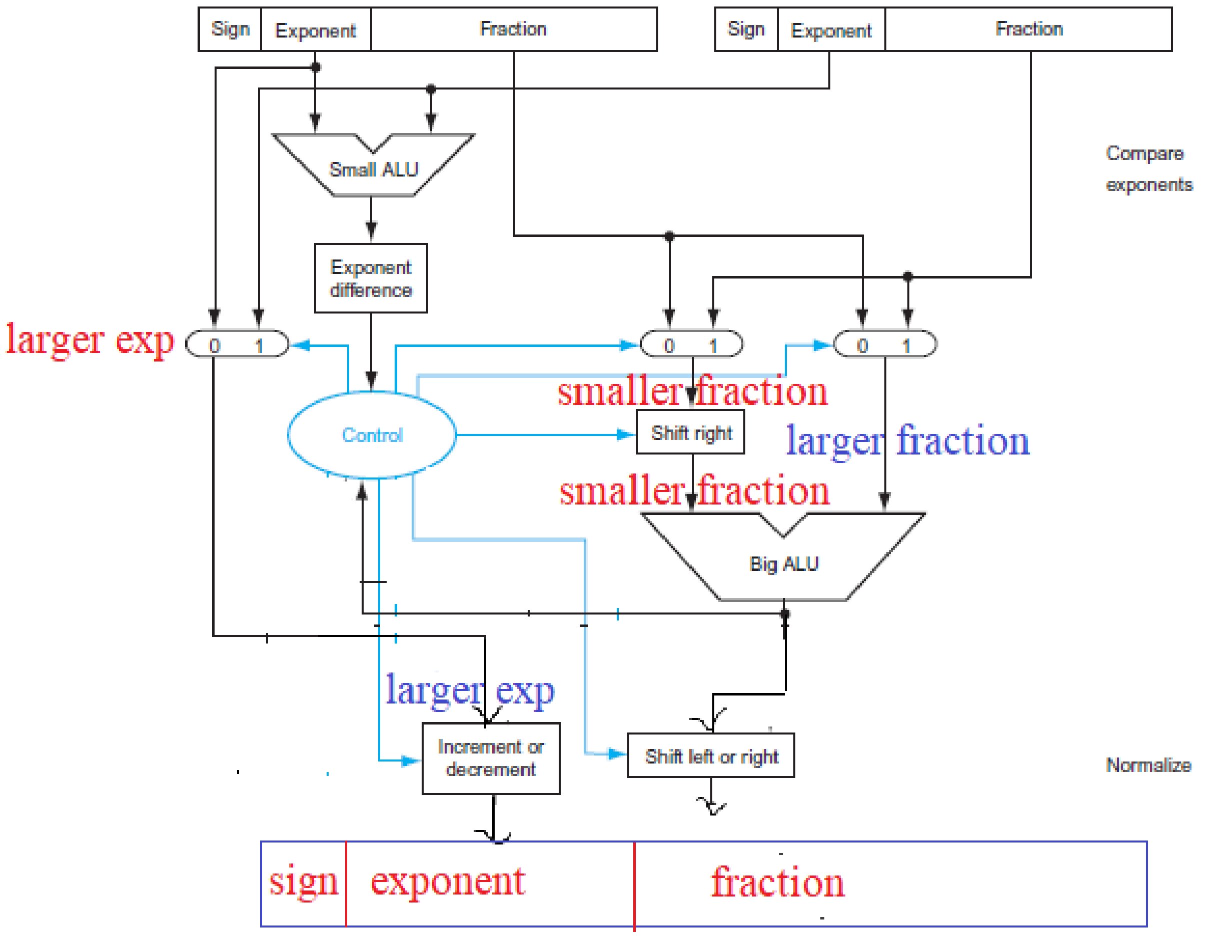
–5.25
$= (-1)^1 \text{x} 101.010$
$= (-1)^1 \text{x} 1.01010 \text{x} 2^{+2}$
$= (-1)^1 \text{x} 1.01010 \text{x} 2^{129}$

0.875 + (–5.25)
$= (-1)^1 \text{x} 1.0001100 \text{x} 2^{129}$
$= -1.0001100 \text{x} 2^{+2}$
$= -100.01100$
$= -(4+0.25+0.125)$
$= -4.375$



$$1.000_{two} \times 2^{-1} \qquad -1.110_{two} \times 2^{-2}$$





Show the hardware implementation of a floating point multiplier/divisor and explain the steps.