Computer Organization & Architecture

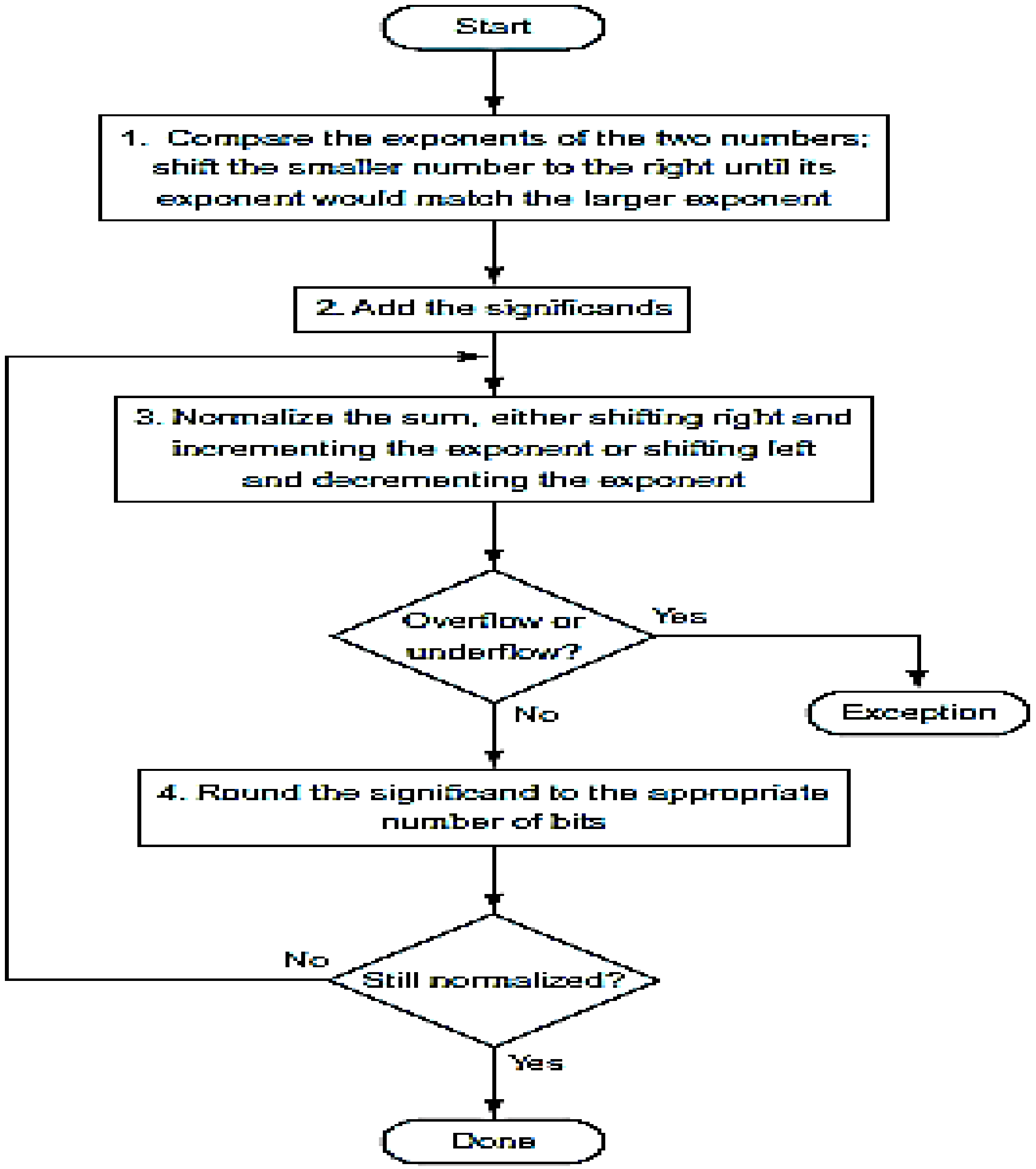
Moazzam Ali Sahi

Floating Point Arithmetic

In this lecture

a) Floating point Arithmetic
b) Division
c) Multiplication
d) Intro to Assembly Language Programming

Floating Point Addition



Example 3 (2.375 ÷ 8.25)

2.375 ÷ 8.25 = 0.287

IEEE representation of ‘2.375’:

0 10000000 00110000000000000000000

[0—128—0.187500]

IEEE representation of ‘8.25’:

0 10000010 00001000000000000000000

[0—130—0.031250]

Preliminary exponent:128 – 130 + 127 = 125

Example 3 Continue...

Mantissa:

```
             1. 0 0 1 0 0 1 1 0 1 1 ...
          -----------------------------
1.00001... ) 1. 0 0 1 1 0 ...
             1. 0 0 0 0 1
             ------------
                    1 0 1 0 0 0
                    1 0 0 0 0 1
                    -----------
                          1 1 1 0 0 0
                          1 0 0 0 0 1
                          -----------
                            1 0 1 1 1 0
                            1 0 0 0 0 1
                            -----------
                                1 1 0 1 0 0
```

IEEE representation of result: ...

0 01111101 00100110110010011011001

[0—125—0.151515]

Binary Floating-Point Multiplication

Let's try multiplying the numbers 0.5_{ten} and -0.4375_{ten}:

In binary, the task is multiplying $1.000_{two} \times 2^{-1}$ by $-1.110_{two} \times 2^{-2}$.

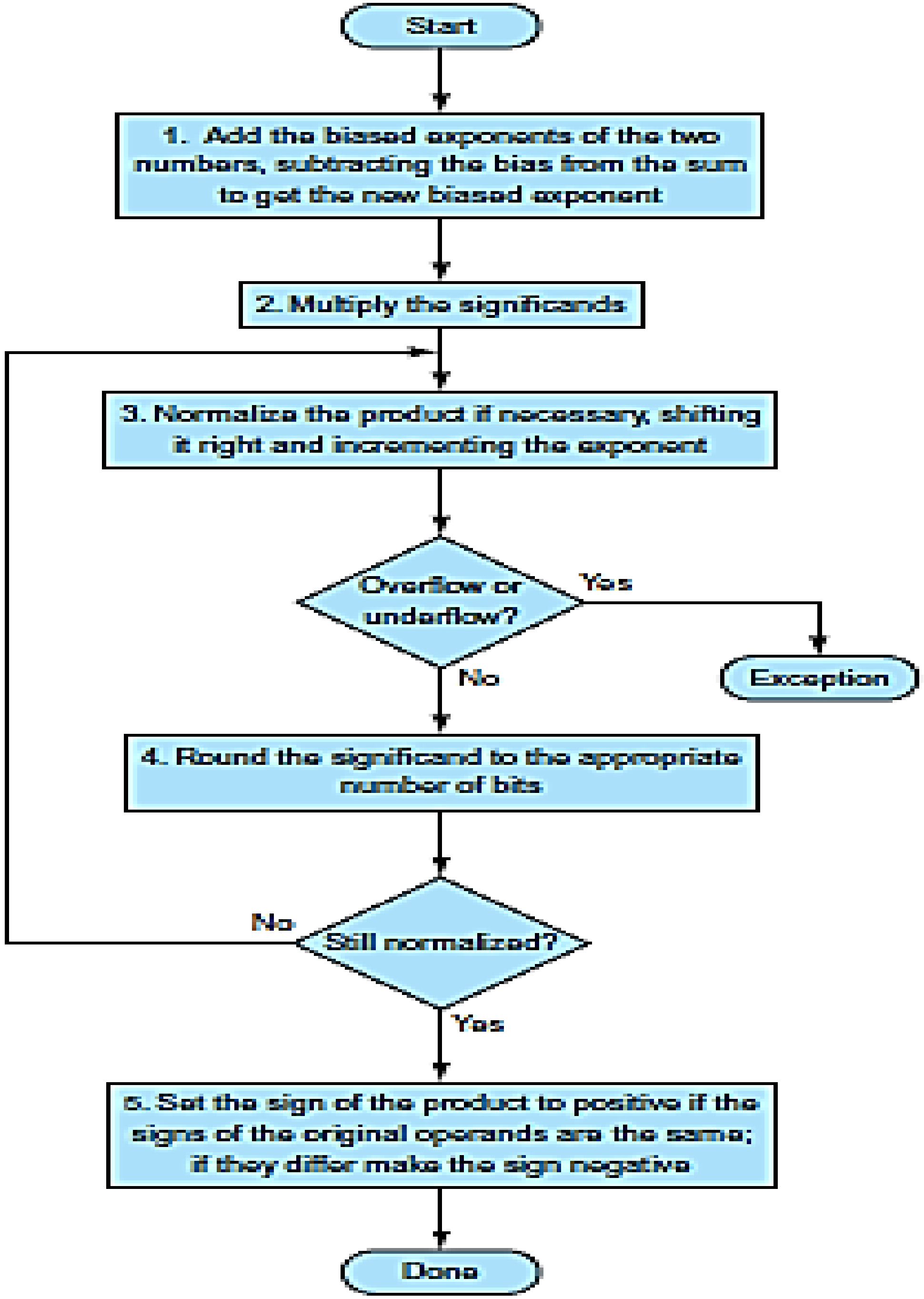
Step 1. Adding the exponents without bias:

$$-1 + (-2) = -3$$

or, using the biased representation:

$$(-1 + 127) + (-2 + 127) - 127 = (-1 - 2) + (127 + 127 - 127)$$
$$= -3 + 127 = 124$$

Binary Floating-Point Multiplication



Binary Floating-Point Multiplication

Step 2. Multiplying the significands:

$$
\begin{array}{r}
1.000_{\text{two}} \\
\times \quad 1.110_{\text{two}} \\
\hline
0000 \\
1000 \\
1000 \\
1000 \\
\hline
1110000_{\text{two}}
\end{array}
$$

The product is $1.110000_{\text{two}} \times 2^{-3}$, but we need to keep it to 4 bits, so it is $1.110_{\text{two}} \times 2^{-3}$.

Binary Floating-Point Multiplication

Step 3. Now we check the product to make sure it is normalized, and then check the exponent for overflow or underflow. The product is already normalized and, since $127 \geq -3 \geq -126$, there is no overflow or underflow. (Using the biased representation, $254 \geq 124 \geq 1$, so the exponent fits.)

Step 4. Rounding the product makes no change:

$$1.110_{\text{two}} \times 2^{-3}$$

Binary Floating-Point Multiplication

Step 5. Since the signs of the original operands differ, make the sign of the product negative. Hence, the product is

$$-1.110_{\text{two}} \times 2^{-3}$$

Converting to decimal to check our results:

$$\begin{aligned} -1.110_{\text{two}} \times 2^{-3} &= -0.001110_{\text{two}} = -0.00111_{\text{two}} \\ &= -7/2^5{}_{\text{ten}} = -7/32_{\text{ten}} = -0.21875_{\text{ten}} \end{aligned}$$

The product of 0.5_{ten} and -0.4375_{ten} is indeed -0.21875_{ten}.

Block diagram of an arithmetic unit dedicated to Floating-point addition