Floating Point Arithmetic

Floating Point Addition

- Example: $9.999_{\text{ten}} \times 10^{1} + 1.610_{\text{ten}} \times 10^{-1}$

- Assumption:
- We can store 4 decimal digits for significand part and 2 decimal digits for the exponent.

Floating Point Addition

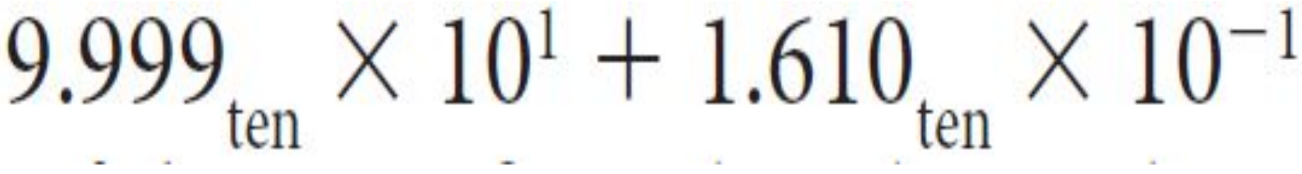
- Example: $9.999_{\text{ten}} \times 10^{1} + 1.610_{\text{ten}} \times 10^{-1}$

- Step 1: Align decimal point of the number that has smaller exponent.

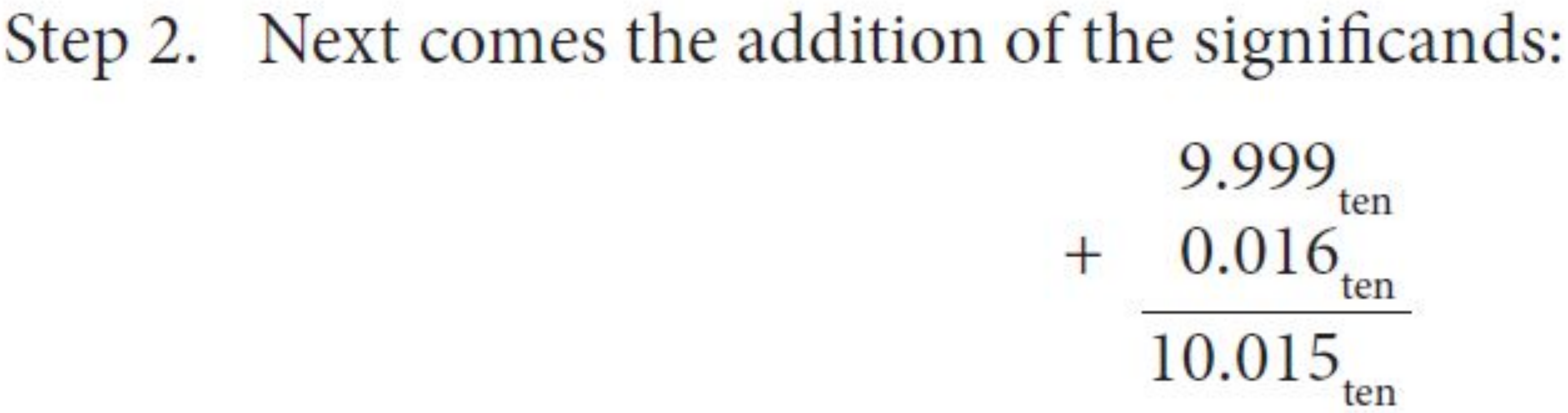
  (exponents of both the numbers must match!!!)

$$0.016 \times 10^{1}$$

Floating Point Addition

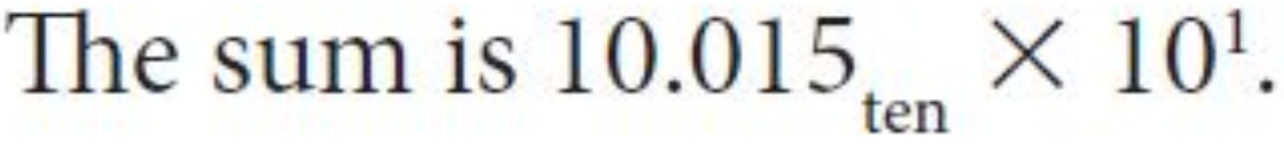
- Example: 

$$9.999_{\text{ten}} \times 10^{1} + 1.610_{\text{ten}} \times 10^{-1}$$

- 

Step 2. Next comes the addition of the significands:

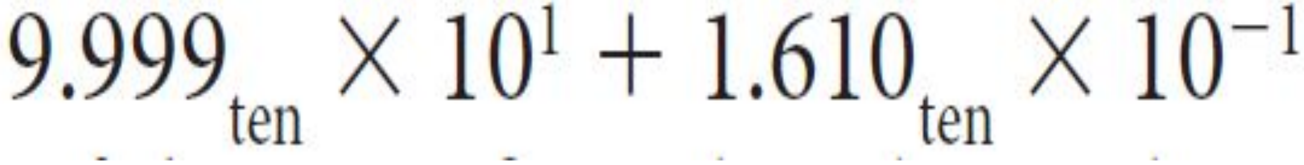
$$\begin{array}{rr} & 9.999_{\text{ten}} \\ + & 0.016_{\text{ten}} \\ \hline & 10.015_{\text{ten}} \end{array}$$



The sum is $10.015_{\text{ten}} \times 10^{1}$.

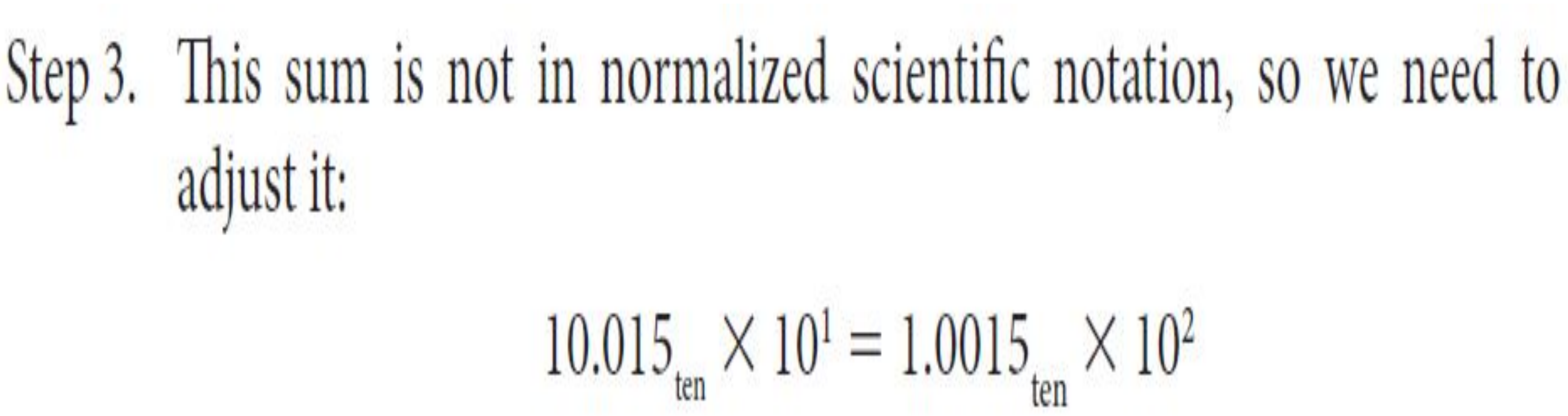
Floating Point Addition

- Example:



$$9.999_{\text{ten}} \times 10^{1} + 1.610_{\text{ten}} \times 10^{-1}$$

-



Step 3. This sum is not in normalized scientific notation, so we need to adjust it:

$$10.015_{\text{ten}} \times 10^{1} = 1.0015_{\text{ten}} \times 10^{2}$$

Floating Point Addition

- Step 4: Rounding up (since we have assumed that four digits will be stored.)

$$1.002_{\text{ten}} \times 10^2$$

Binary Floating Point Addition

- Perform: $0.5 + (-0.4375)$

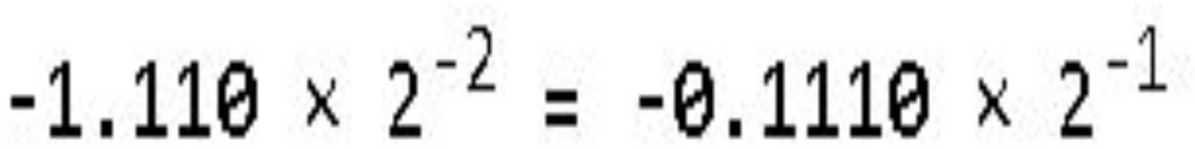
- Step1: Convert the numbers into binary and normalize them.

$$0.5 = 0.1 \times 2^{0} = 1.000 \times 2^{-1} \text{ (normalised)}$$

$$-0.4375 = -0.0111 \times 2^{0} = -1.110 \times 2^{-2} \text{ (normalised)}$$

Binary Floating Point Addition

- Step 2: Rewrite the smaller number such that its exponent matches with the exponent of the larger number.



$$-1.110 \times 2^{-2} = -0.1110 \times 2^{-1}$$

Binary Floating Point Addition

- Step 3: Add the mantissas/significands

$$1.000 \times 2^{-1} + -0.1110 \times 2^{-1} = 0.001 \times 2^{-1}$$

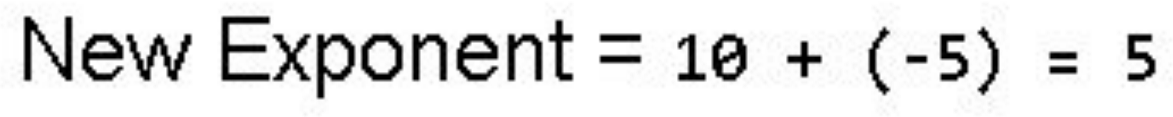
- Step 4: Normalise the sum, checking for overflow/underflow.

$$0.001 \times 2^{-1} = 1.000 \times 2^{-4}$$

$-126 <= -4 <= 127$ ===> No overflow or underflow

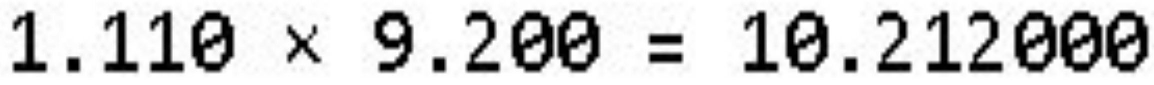
Floating Point Multiplication

- Example : $1.110 \times 10^{10} \times 9.200 \times 10^{-5}$
- Step 1: Add the exponents:



$$\text{New Exponent} = 10 + (-5) = 5$$

- Step 2: Multiply the mantissas:



$$1.110 \times 9.200 = 10.212000$$

Floating Point Multiplication

- Step 3: Normalise the result

$$1.0212 \times 10^{6}$$

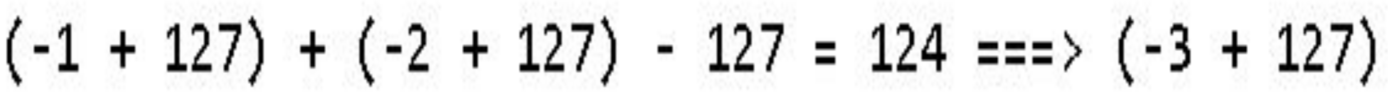
- Step 4: Round it

$$1.021 \times 10^{6}$$

Example multiplication in binary:

- $1.000 \times 2^{-1} \times -1.110 \times 2^{-2}$

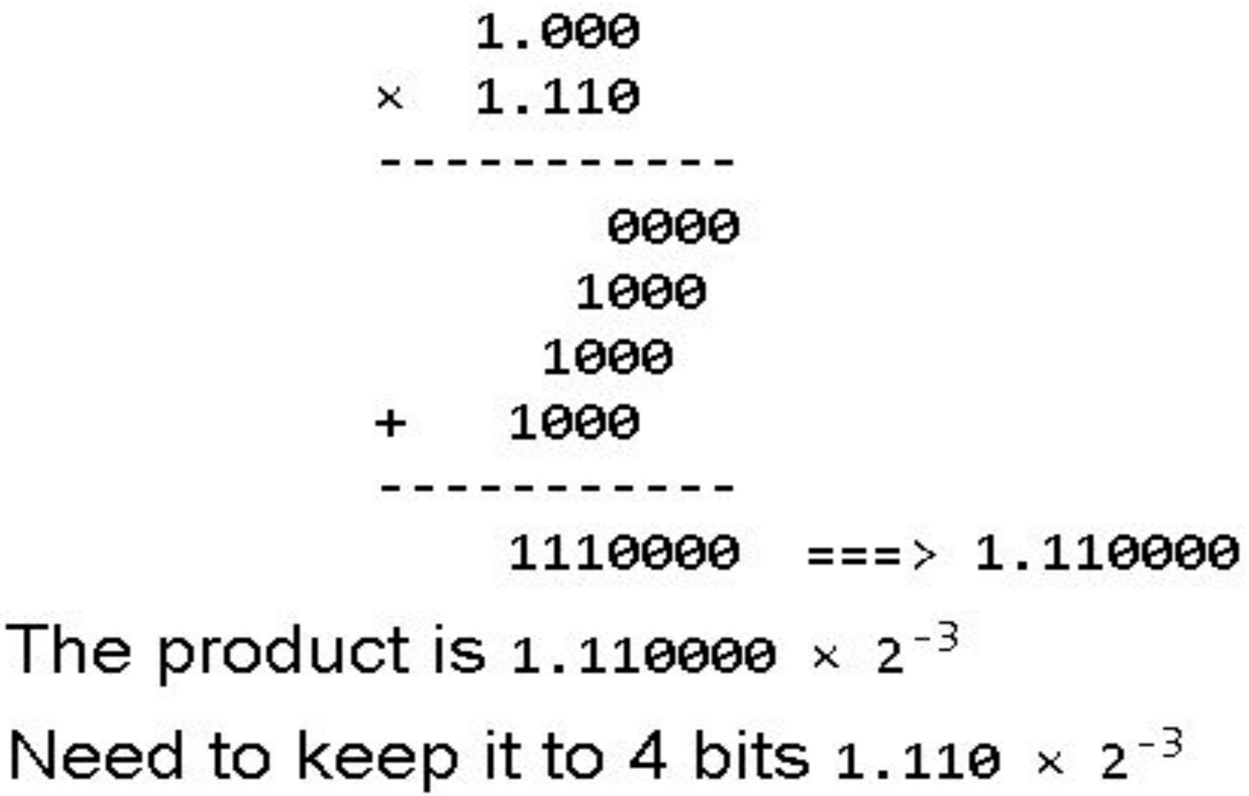
Step 1: Add the biased exponents



```
(-1 + 127) + (-2 + 127) - 127 = 124 ===> (-3 + 127)
```

Example multiplication in binary:

- Step 3: Multiply the mantissas



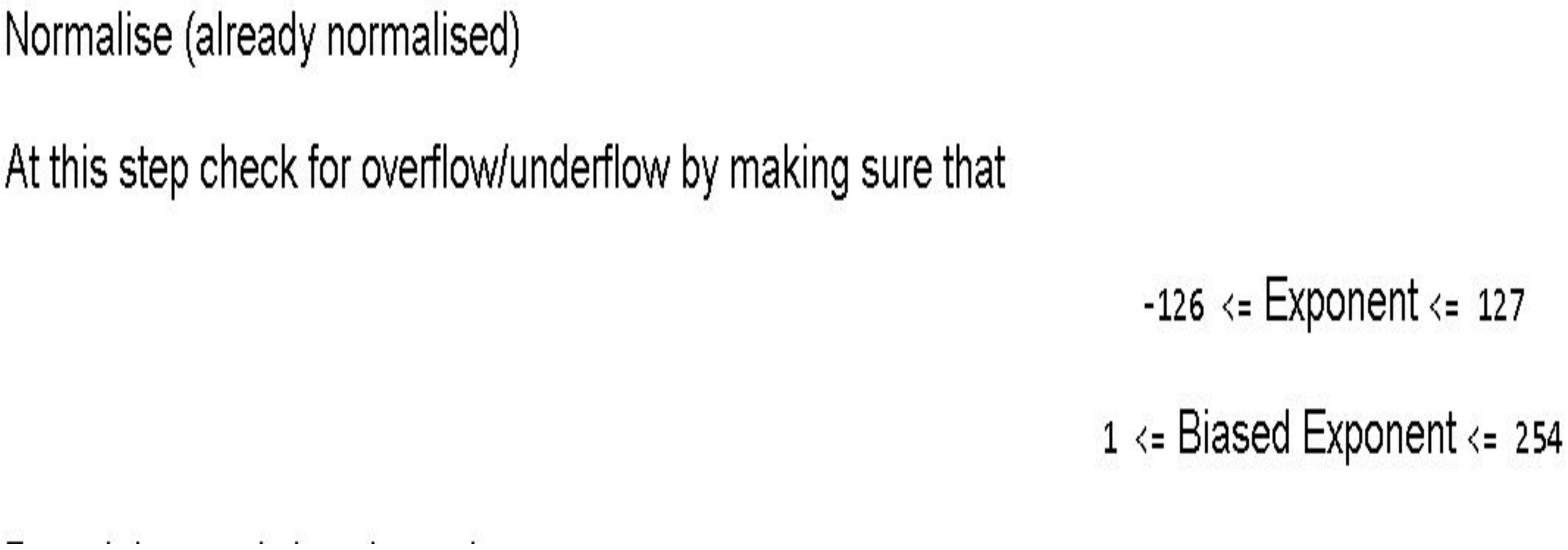
```
   1.000
×  1.110
-----------
       0000
      1000
     1000
+   1000
-----------
    1110000  ===> 1.110000
```

The product is 1.110000×2^{-3}

Need to keep it to 4 bits 1.110×2^{-3}

Example multiplication in binary:

- Step 3:



Normalise (already normalised)

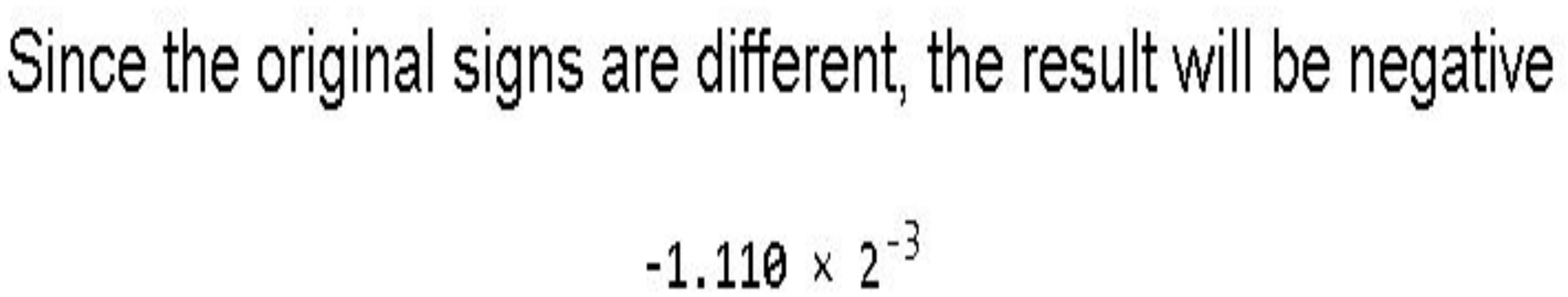
At this step check for overflow/underflow by making sure that

$$-126 <= \text{Exponent} <= 127$$

$$1 <= \text{Biased Exponent} <= 254$$

Example multiplication in binary:

- Step 5: Adjust the signs.



Since the original signs are different, the result will be negative

-1.110×2^{-3}