Київський національний університет імені Тараса Шевченка
факультет радіофізики, електроніки та комп'ютерних систем

Лабораторна робота № 2

Тема: «Арифметичні операції над двійковими числами»

Роботу виконала
студентка 3-го курсу
напрямку СА
Прищепа Олександра Віталіївна

Київ 2022

Хід роботи

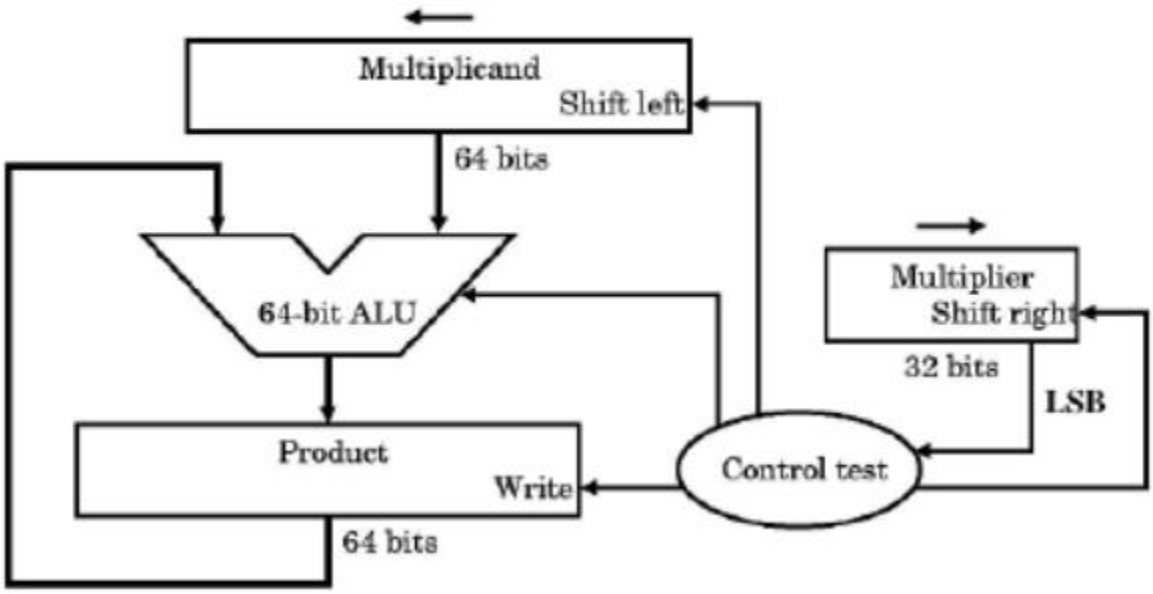
Мета: Дослідити алгоритми, що використовуються в мікропроцесорах для множення та ділення цілих чисел та підходи до роботи з дійсними числами.

Варіант: a b b

Посилання на [репозиторій]

1. Створимо програму, що ілюструє покрокове виконання алгоритму множення як є:

a. Множення як є



Пояснення алгоритму:
1. Перший множник(multiplicand) переводимо з десяткової системи в двійкову, а також переводимо в систему числення з основою 64. Другий множник(multiplier) переводимо з десяткової системи в двійкову, а також переводимо в систему числення з основою 32.

2. Перевіряємо молодший біт другого множника(multiplier):

- якщо він 0, тоді робимо зсув вправо 2-го множника(multiplier) та зсув вліво 1-го множника(multiplicand);

- якщо він 1, тоді додаємо(ALU) 1-й множник(multiplicand) в результат(product) і записуємо в product. Потім робимо зсув вправо 2-го множника(multiplier) та зсув вліво 1-го множника(multiplicand);

3. При роботі з від'ємними числами переводимо число в додатне, відпрацьовуємо алгоритм, а в кінці повертаємо знак відповідно до вхідних параметрів.

Перевірення роботи програми:

Multiplicand	**Multiplier**	**Product**
13 (00000000000000000000000000000000000000000000000000000000000001101)	**5** (00000000000000000000000000000101)	**65** (00000000000000000000000000000000000000000000000000000000001000001)
200 00000000000000000000000000000000000000000000000000000000011001000	**15** 00000000000000000000000000001111	**3000** (00000000000000000000000000000000000000000000000000000000101110111000)
22	**-2**	**-44**

00000000000000000000000000000000 00000000000000010110	11111111111111111111 1111111110	11111111111111111111111111111111111111111111 11111111111111010100
-100 11111111111111111111111111111111111111111111 11111111111110011100	**-3** 11111111111111111111 1111111101	**300** 00000000000000000000000000000000000000000000 00000000000100101100

Виконання програми на прикладі чисел 13 та 5:

```
Value1 =  13
Value2 =  5
Multiplicand: 00000000000000000000000000000000000000000000000000000000000001101
Multiplier:   00000000000000000000000000000101
Product:      00000000000000000000000000000000000000000000000000000000000000000
Молодший біт 1, додаємо Multiplicand в Product і записуємо в Product,
Робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     1
Multiplicand: 00000000000000000000000000000000000000000000000000000000000011010
Multiplier:   00000000000000000000000000000010
Product:      00000000000000000000000000000000000000000000000000000000000001101
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     2
Multiplicand: 00000000000000000000000000000000000000000000000000000000000110100
Multiplier:   00000000000000000000000000000001
Product:      00000000000000000000000000000000000000000000000000000000000001101
Молодший біт 1, додаємо Multiplicand в Product і записуємо в Product,
Робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     3
Multiplicand: 00000000000000000000000000000000000000000000000000000000001101000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо
```

```
Ітерація:     4
Multiplicand: 00000000000000000000000000000000000000000000000000000011010000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     5
Multiplicand: 00000000000000000000000000000000000000000000000000000110100000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     6
Multiplicand: 00000000000000000000000000000000000000000000000000001101000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     7
Multiplicand: 00000000000000000000000000000000000000000000000000011010000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо
```

```
Ітерація:     8
Multiplicand: 00000000000000000000000000000000000000000000000000110100000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     9
Multiplicand: 00000000000000000000000000000000000000000000000001101000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     10
Multiplicand: 00000000000000000000000000000000000000000000000011010000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     11
Multiplicand: 00000000000000000000000000000000000000000000000110100000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо
```

```
Ітерація:     12
Multiplicand: 00000000000000000000000000000000000000000000001101000000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     13
Multiplicand: 00000000000000000000000000000000000000000000011010000000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     14
Multiplicand: 00000000000000000000000000000000000000000000110100000000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     15
Multiplicand: 00000000000000000000000000000000000000000001101000000000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо
```

```
Ітерація:     16
Multiplicand: 00000000000000000000000000000000000000000011010000000000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     17
Multiplicand: 00000000000000000000000000000000000000000110100000000000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     18
Multiplicand: 00000000000000000000000000000000000000000011010000000000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     19
Multiplicand: 00000000000000000000000000000000000000000110100000000000000000
Multiplier:   00000000000000000000000000000000
Product:      00000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо
```

```
Ітерація:     20
Multiplicand: 0000000000000000000000000000000000000000011010000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     21
Multiplicand: 0000000000000000000000000000000000000000110100000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     22
Multiplicand: 0000000000000000000000000000000000000001101000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     23
Multiplicand: 0000000000000000000000000000000000000011010000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо
```

```
Ітерація:     24
Multiplicand: 0000000000000000000000000000000000000110100000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     25
Multiplicand: 0000000000000000000000000000000000001101000000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     26
Multiplicand: 0000000000000000000000000000000000011010000000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     27
Multiplicand: 0000000000000000000000000000000000110100000000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо
```

```
Ітерація:     28
Multiplicand: 0000000000000000000000000000001101000000000000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     29
Multiplicand: 0000000000000000000000000000011010000000000000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     30
Multiplicand: 0000000000000000000000000000110100000000000000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо

Ітерація:     31
Multiplicand: 0000000000000000000000000001101000000000000000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001
Молодший біт 0, робимо зсув Multiplicand <-- вліво, Multiplier --> вправо
```

```
Ітерація:     32
Multiplicand: 0000000000000000000000000011010000000000000000000000000000000000
Multiplier:   00000000000000000000000000000000
Product:      0000000000000000000000000000000000000000000000000000000001000001

Result:       0000000000000000000000000000000000000000000000000000000001000001
```

Інші перевірки:

```
 Value1 =  200
 Value2 =  15
 Multiplicand: 0000000000000000000000000000000000000000000000000000000011001000
 Multiplier:   00000000000000000000000000001111
```

```
 Result:       0000000000000000000000000000000000000000000000000000101110111000
```

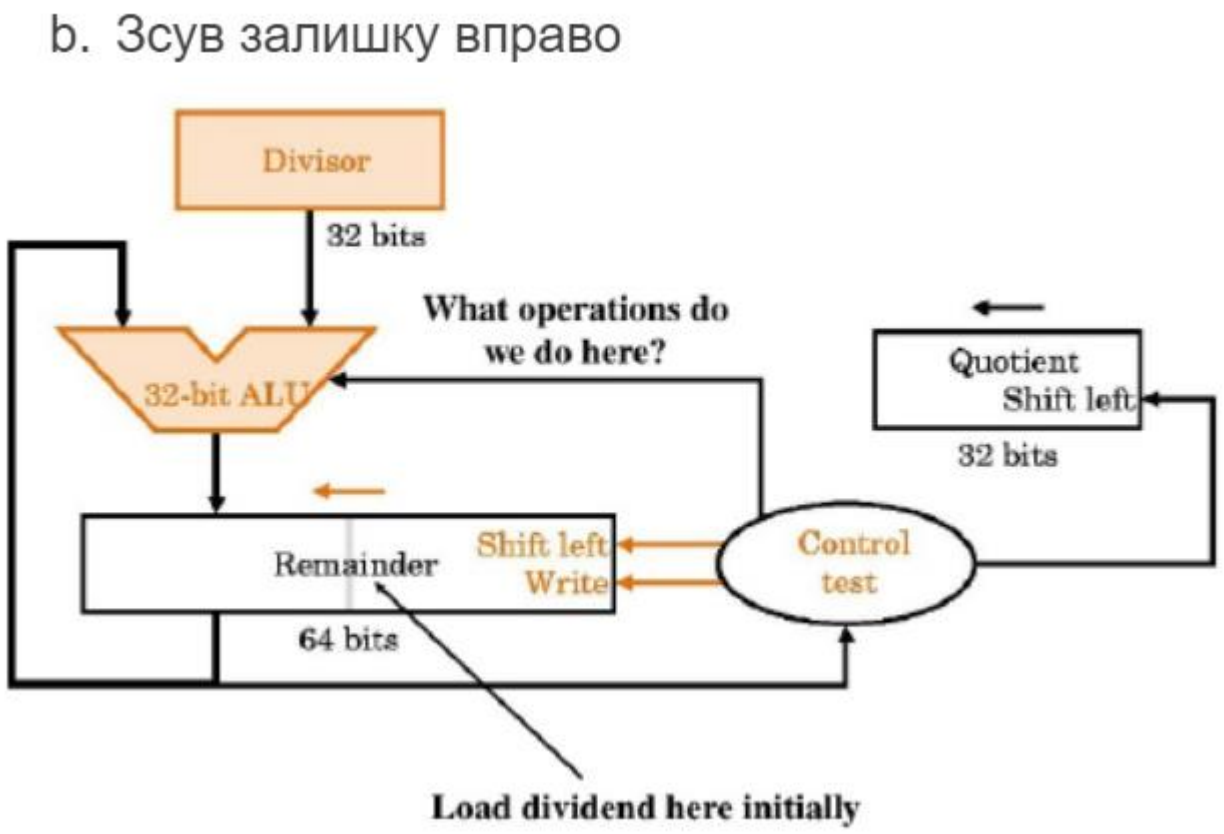
```
Value1 =  22
Value2 =  -2
Multiplicand: 0000000000000000000000000000000000000000000000000000000000010110
Multiplier:   00000000000000000000000000000010
Product:      0000000000000000000000000000000000000000000000000000000000000000
```

```
 Result:       1111111111111111111111111111111111111111111111111111111010100
```

```
Value1 =  -100
Value2 =  -3
Multiplicand: 00000000000000000000000000000000000000000000000000000001100100
Multiplier:   00000000000000000000000000000011
Product:      0000000000000000000000000000000000000000000000000000000000000000
```

```
Result:       0000000000000000000000000000000000000000000000000000100101100
```

2. Створимо програму, що ілюструє покрокове виконання алгоритму ділення двійкових чисел:



Пояснення алгоритму:

1. Ділене(dividend) та дільник(divisor) переводимо з десяткової системи в двійкову, а також переводимо в систему числення з основою 32.

2. Зсуваємо дільник до першої значимої «1» та розташовуємо під найстаршим бітом діленого.

3. Порівнюємо дільник з діленим:

- якщо він більше, то в результат(quotient) записуємо «0» та зсуваємо результат вліво;

- якщо він менше, то в результат(quotient) записуємо «1» та зсуваємо результат вліво;

4. Проходимося циклом по алгоритму, в кінці отримуємо частку та, за наявності остачу(remainder).

Перевірення роботи програми:

Dividend	Divisor	Quotient	Remainder

2000000000 00000000000000 0110000110101 000000	19 00000000000000 00000000000000 010011	105263157 00000110010000 11000110000000 110101	17 00000000000000000000000000000 00000000000000000000000000000 0000010001
20 00000000000000 00000000000000 010100	2 00000000000000 00000000000000 000010	10 00000000000000 00000000000000 001010	0 00000000000000000000000000000 00000000000000000000000000000 0000000000
5616	3	1872 00000000000000 00000000111010 10000	0 00000000000000000000000000000 00000000000000000000000000000 0000000000

Виконання програми на прикладі чисел 5616 та 3:

```
Value1 =  5616
Value2 =  3
Value1 =  00000000000000000001010111110000
Value2 =  11000000000000000000000000000000      Перший знаковий біт дільника нормалізовуємо з першим бітом діленого

Ітерація:   1
Value1      00000000000000000001010111110000
Value2      01100000000000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   2
Value1      00000000000000000001010111110000
Value2      00110000000000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   3
Value1      00000000000000000001010111110000
Value2      00011000000000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   4
Value1      00000000000000000001010111110000
Value2      00001100000000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво
```

```
Ітерація:   5
Value1      00000000000000000001010111110000
Value2      00000110000000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   6
Value1      00000000000000000001010111110000
Value2      00000011000000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   7
Value1      00000000000000000001010111110000
Value2      00000001100000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   8
Value1      00000000000000000001010111110000
Value2      00000000110000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   9
Value1      00000000000000000001010111110000
Value2      00000000011000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво
```

```
Ітерація:   10
Value1      00000000000000000001010111110000
Value2      00000000011000000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   11
Value1      00000000000000000001010111110000
Value2      00000000001100000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   12
Value1      00000000000000000001010111110000
Value2      00000000000110000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   13
Value1      00000000000000000001010111110000
Value2      00000000000011000000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   14
Value1      00000000000000000001010111110000
Value2      00000000000001100000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво
```

```
Ітерація:   15
Value1      00000000000000000001010111110000
Value2      00000000000000011000000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   16
Value1      00000000000000000001010111110000
Value2      00000000000000001100000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   17
Value1      00000000000000000001010111110000
Value2      00000000000000000110000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   18
Value1      00000000000000000001010111110000
Value2      00000000000000000011000000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   19
Value1      00000000000000000001010111110000
Value2      00000000000000000001100000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   20
Value1      00000000000000000001010111110000
Value2      00000000000000000000110000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво
```

```
Ітерація:   21
Value1      00000000000000000000100111110000
Value2      00000000000000000000011000000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000001    Порівнюємо, дільник менше діленого, тому в результат записуємо 1 та зсуваємо результат вліво

Ітерація:   22
Value1      00000000000000000000001111110000
Value2      00000000000000000000001100000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000011    Порівнюємо, дільник менше діленого, тому в результат записуємо 1 та зсуваємо результат вліво

Ітерація:   23
Value1      00000000000000000000000111110000
Value2      00000000000000000000000110000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000000111    Порівнюємо, дільник менше діленого, тому в результат записуємо 1 та зсуваємо результат вліво

Ітерація:   24
Value1      00000000000000000000000011110000
Value2      00000000000000000000000011000000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000001110    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   25
Value1      00000000000000000000000001110000
Value2      00000000000000000000000001100000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000011101    Порівнюємо, дільник менше діленого, тому в результат записуємо 1 та зсуваємо результат вліво

Ітерація:   26
Value1      00000000000000000000000000110000
Value2      00000000000000000000000000110000    Робимо зсув діленого вліво
Quotient    00000000000000000000000000111010    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво
```

```
Ітерація:   27
Value1      00000000000000000000000000000000
Value2      00000000000000000000000000011000    Робимо зсув діленого вліво
Quotient    00000000000000000000000001110101    Порівнюємо, дільник менше діленого, тому в результат записуємо 1 та зсуваємо результат вліво

Ітерація:   28
Value1      00000000000000000000000000000000
Value2      00000000000000000000000000001100    Робимо зсув діленого вліво
Quotient    00000000000000000000000011101010    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   29
Value1      00000000000000000000000000000000
Value2      00000000000000000000000000000110    Робимо зсув діленого вліво
Quotient    00000000000000000000000111010100    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   30
Value1      00000000000000000000000000000000
Value2      00000000000000000000000000000011    Робимо зсув діленого вліво
Quotient    00000000000000000000001110101000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Ітерація:   31
Value1      00000000000000000000000000000000
Value2      00000000000000000000000000000001    Робимо зсув діленого вліво
Quotient    00000000000000000000011101010000    Порівнюємо, дільник більше діленого, тому в результат записуємо 0 та зсуваємо результат вліво

Результат   00000000000000000000011101010000
Остача      00000000000000000000000000000000000000000000000000000000000000000
```

Інші випадки:

```
Value1 =  2000000000
Value2 =  19
Value1 =  01110111001101011001010000000000
Value2 =  10011000000000000000000000000000      Перший знаковий біт дільника нормалізовуємо з першим бітом діленого
```

```
Результат   00000110010001100011000000110101
Остача      00000000000000000000000000000000000000000000000000000000010001
```

```
Value1 =  20
Value2 =  2
Value1 =  00000000000000000000000000010100
Value2 =  10000000000000000000000000000000      Перший знаковий біт дільника нормалізовуємо з першим бітом діленого
```

```
Результат   00000000000000000000000000001010
Остача      00000000000000000000000000000000000000000000000000000000000000
```

3. Створимо програму, що ілюструє покрокове виконання алгоритму множення двійкових чисел в IEEE 754 Floating Point:

b. Множення
   i. Compute exponents
   ii. Multiply significands
   iii. Normalize result
   iv. Set sign

Пояснення алгоритму:
1. Виконуємо XOR біта зі знаком множника1 із бітовим знаком множника2. Результат поміщаємо в результуючий знаковий біт.
2. Перемножуємо мантиси м1 та м2 , а результат поміщаємо в результуюче поле мантиси.

3. Додаємо біти експонент м1 та м2, а базове значення віднімаємо від доданого результату(127). Результат поміщається в експоненційне поле блоку результату.
4. Нормалізовуємо суму, або зсуваючи праворуч і збільшуючи показник, або зсуваючи вліво та зменшуючи показник.
5. Перевірте відсутність/переповнення.
6. Якщо (E1 + E2 - зміщення) >= до Emax, то встановлюємо добуток на нескінченність.
7. Якщо E1 + E2 - зміщення) менше/дорівнює Emin, то встановлюємо добуток на нуль.

Перевірення роботи програми:

Множник	Множник	Добуток
7.875 01000000111111000000000000000000	15.875 01000001011111100000000000000000	125.015625 01000010111110100000100000000000
1.5 00111111110000000000000000000000	105.00000001 01000010110100100000000000000000	157.50000001499998 01000011000111011000000000000000
55555555555.555 01010001010011101111010111101000	105.875 01000010110100111100000000000000	5881944444444.386 01010100101010110010111111100110

Виконання програми:

```
Value1 =  7.875
Value2 =  15.875
Value1 =  01000000111111000000000000000000
Value2 =  01000001011111100000000000000000
Виконуємо XOR між бітом зі знаком:                0
Перемножуємо мантиси множника1 та множника2:     11111010000100000000000000000000000000000000000000
Нормалізовуємо мантису:                           11110100001000000000000000000000000000000000
Додаємо порядки множника1 та множника2:          100000011
Віднімаємо зміщення (127 біт) та додаємо 1:      10000101
Результат:  01000010111110100001000000000000
```

```
Value1 =  1.5
Value2 =  105.00000001
Value1 =  00111111110000000000000000000000
Value2 =  01000010110100100000000000000000
Виконуємо XOR між бітом зі знаком:                0
Перемножуємо мантиси множника1 та множника2:     10011101100000000000000000000000000000000000000000
Нормалізовуємо мантису:                           00111011000000000000000000000000000000000000
Додаємо порядки множника1 та множника2:          100000100
Віднімаємо зміщення (127 біт) та додаємо 1:      10000110
Результат:  01000011000111011000000000000000
```

```
Value1 =  555555555555.555
Value2 =  105.875
Value1 =  01010001010011101111010111101000
Value2 =  01000010110100111100000000000000
Виконуємо XOR між бітом зі знаком:                0
Перемножуємо мантиси множника1 та множника2:     10101011001011111110011010100110000000000000000000
Нормалізовуємо мантису:                           01010110010111111100110101001100000000000000000000
Додаємо порядки множника1 та множника2:          100100111
Віднімаємо зміщення (127 біт) та додаємо 1:      10101001
Результат:  01010100101010110010111111100110
```

Висновок

У ході лабораторної роботи було досліджено алгоритми множення, ділення та множення в IEEE 754 Floating Point, які використовуються в процесорі. Отже, алгоритми в процесорі реалізовані набагато складніше на відміну від примітивних операцій, які ми звикли виконувати множивши, додававши, або діливши числа в стовбчик. Також при множення в системі IEEE 754 чисел з плаваючою комою, спостерігалася деяка похибка, вона зумовлена обмеженими можливостями процесора, а саме тому що ми не можемо закодувати число з плаваючою комою з 100% правильністю.