**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

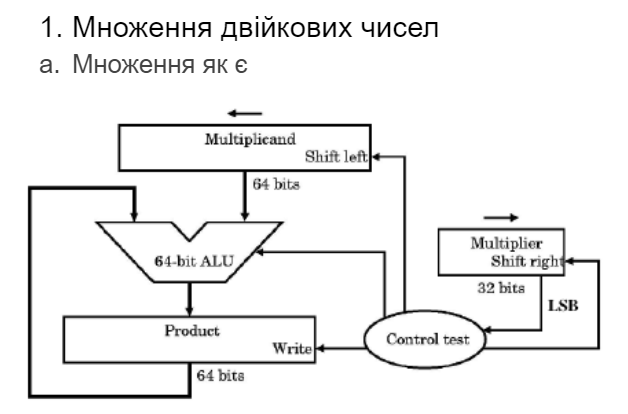
Хід роботи

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](https://gl.vlabs.knu.ua/frecs/ce/cs/2021-2022/Oleksandra/laba2_ks)

1. Створимо програму, що ілюструє покрокове виконання алгоритму 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);

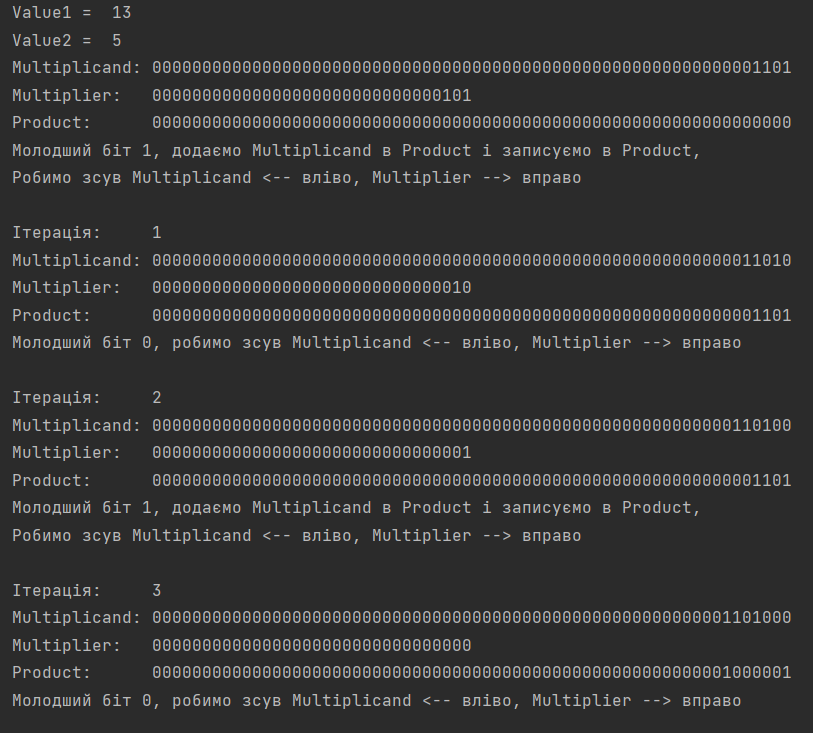
- якщо він 1, тоді додаємо(ALU) 1-й множник(multiplicand) в результат(рroduct) і записуємо в рroduct. Потім робимо зсув вправо 2-го множника(multiplier) та зсув вліво 1-го множника(multiplicand);

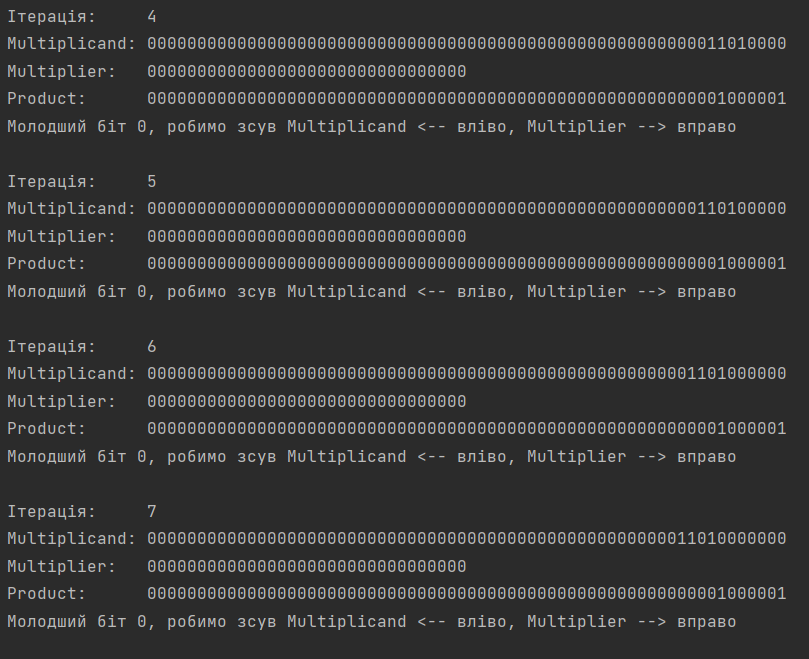
3. При роботі з від’ємними числами переводимо число в додатне, відпрацьовуємо алгоритм, а в кінці повертаємо знак відповідно до вхідних параметрів.

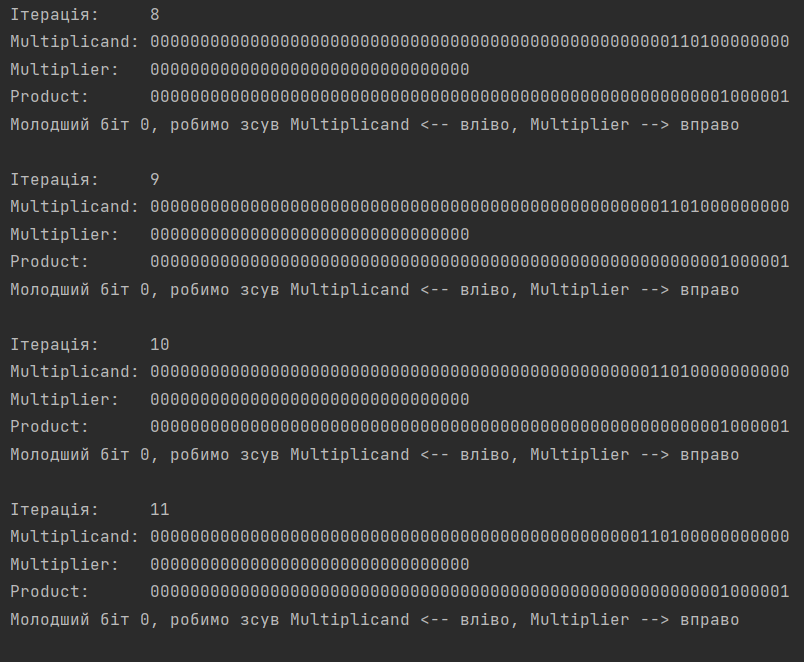
Перевірення роботи програми:

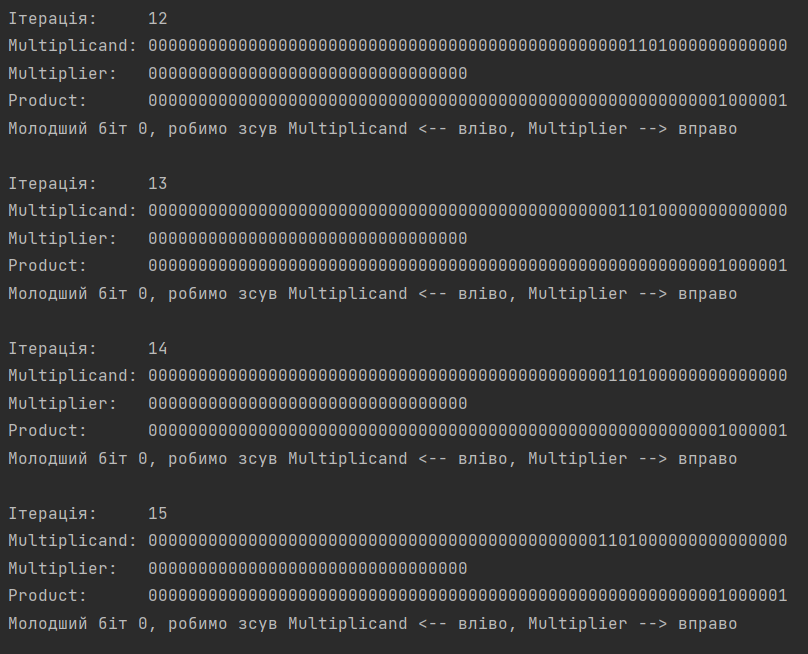
|  |  |  |
| --- | --- | --- |
| **Multiplicand** | **Multiplier** | **Product** |
| **13** (0000000000000000000000000000000000000000000000000000000000001101) | **5** (00000000000000000000000000000101) | **65** (0000000000000000000000000000000000000000000000000000000001000001) |
| **200**  0000000000000000000000000000000000000000000000000000000011001000 | **15**  00000000000000000000000000001111 | **3000**  (0000000000000000000000000000000000000000000000000000101110111000) |
| **22**  0000000000000000000000000000000000000000000000000000000000010110 | **-2**  11111111111111111111111111111110 | **-44**  1111111111111111111111111111111111111111111111111111111111010100 |
| **-100**  1111111111111111111111111111111111111111111111111111111110011100 | **-3**  11111111111111111111111111111101 | **300**  0000000000000000000000000000000000000000000000000000000100101100 |

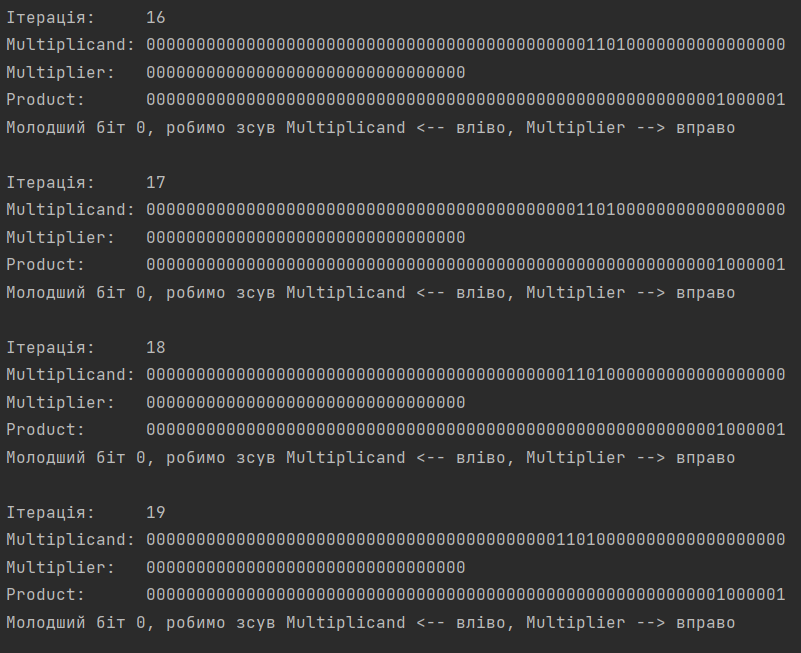
Виконання програми на прикладі чисел 13 та 5:

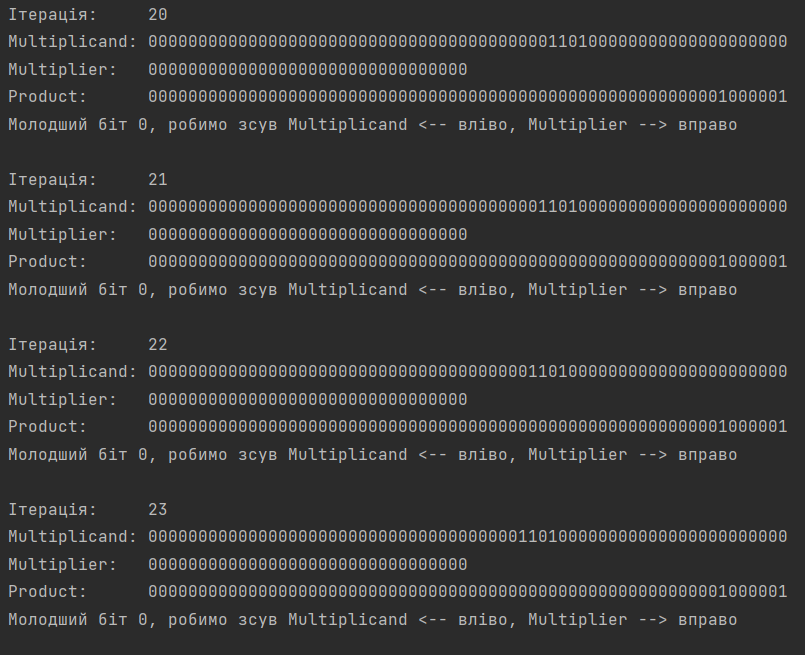


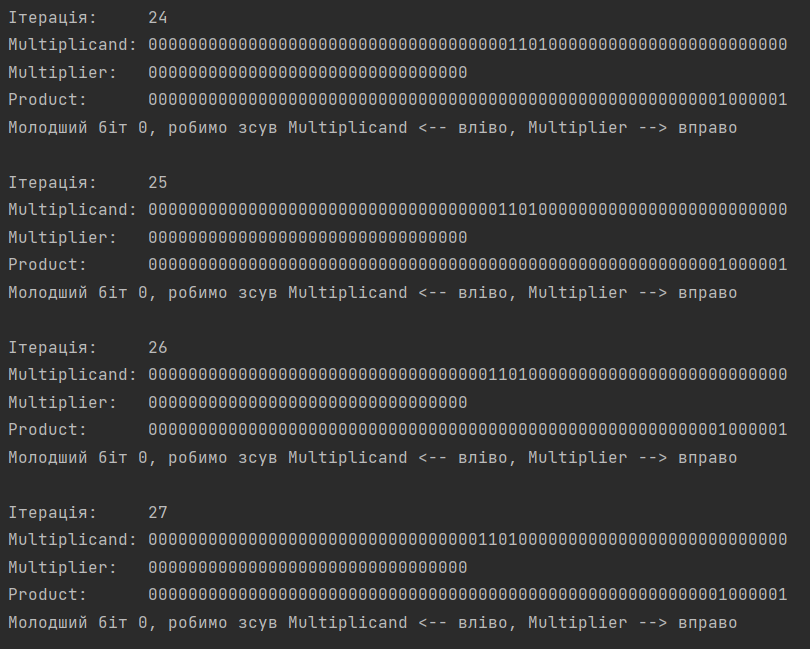


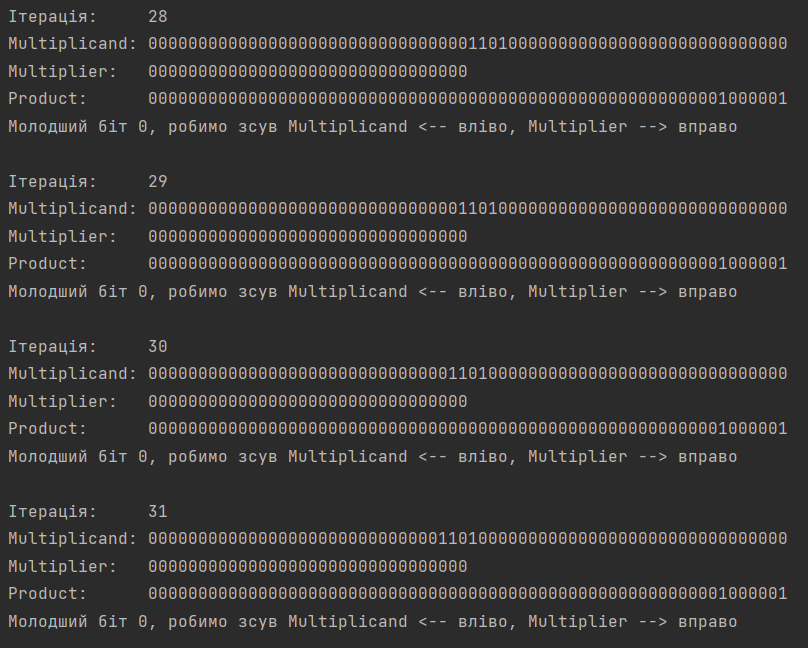


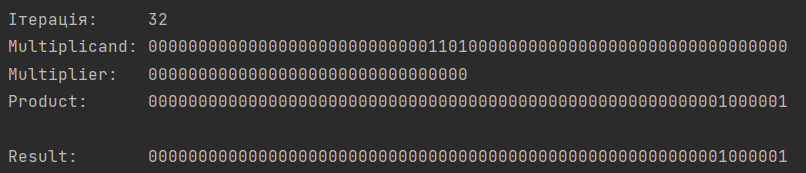




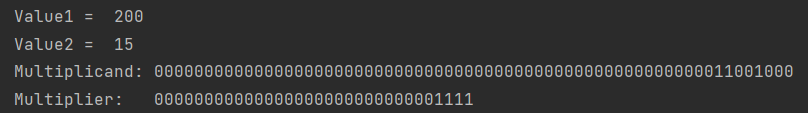




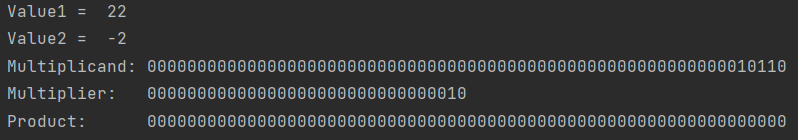




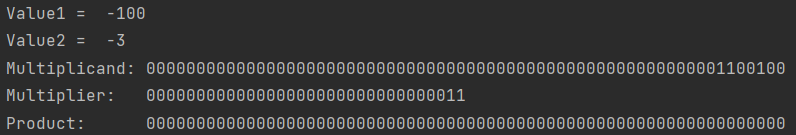
Інші перевірки:





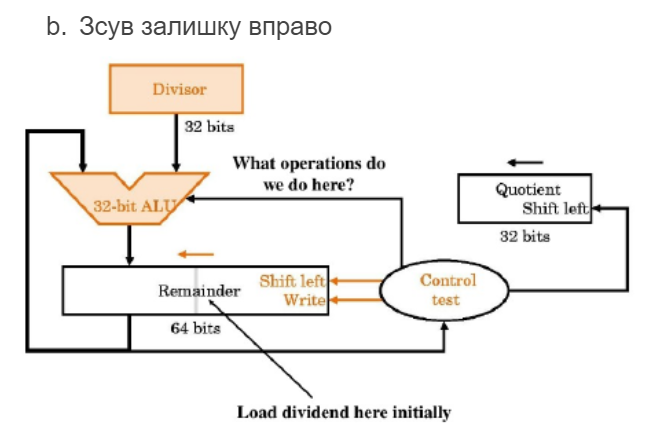








2. Створимо програму, що ілюструє покрокове виконання алгоритму ділення двійкових чисел:



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;

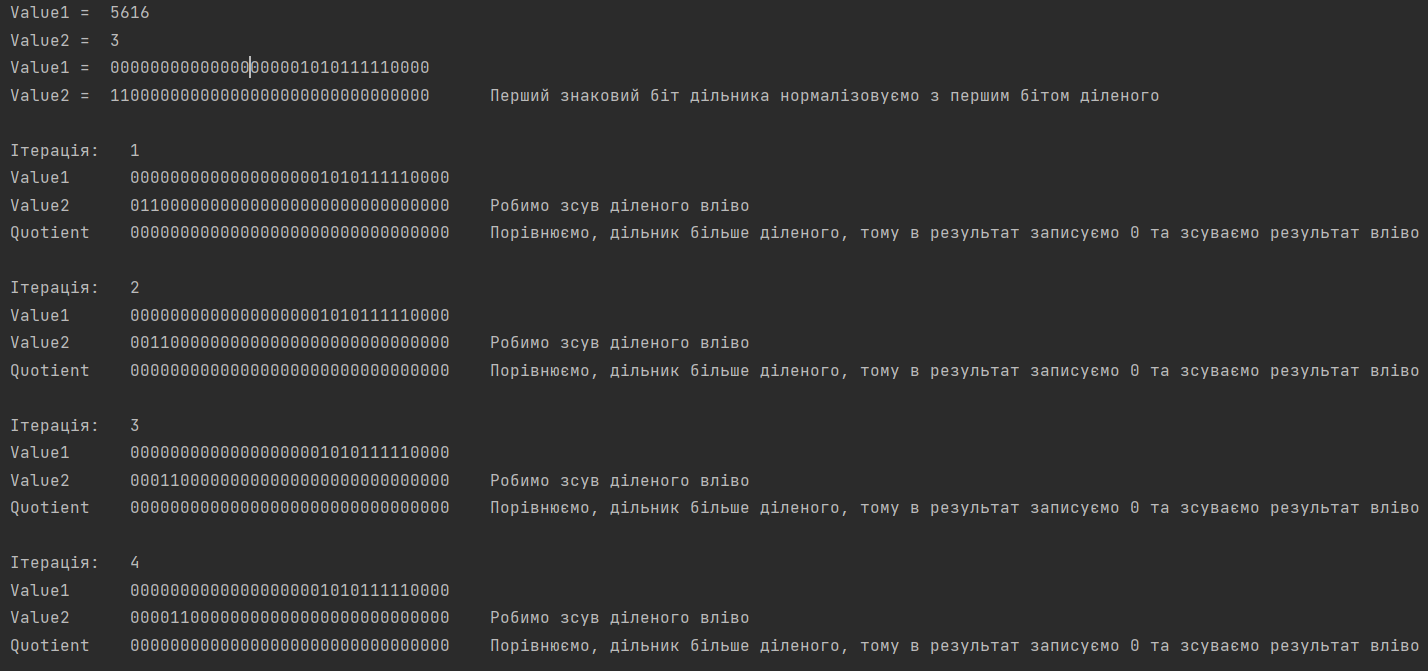
- якщо він менше, то в результат(quotient) записуємо «1» та зсуваємо результат вліво;

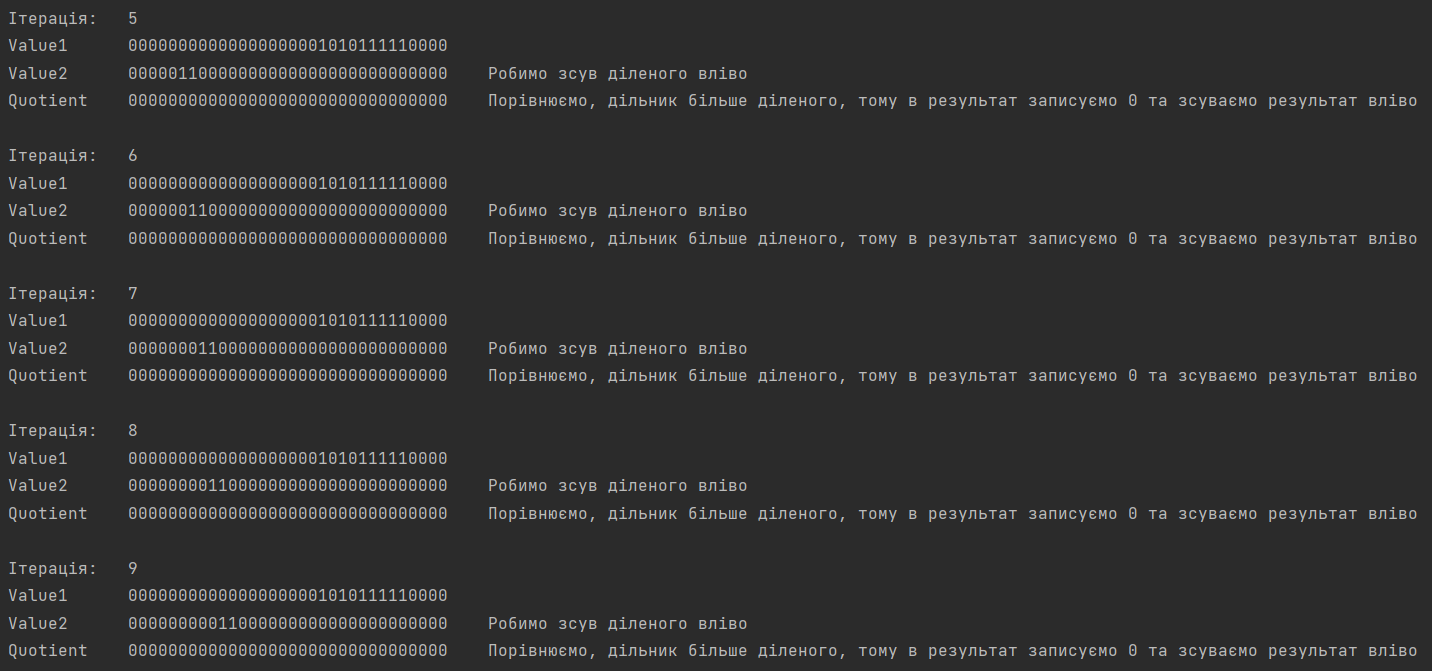
4. Проходимося циклом по алгоритму, в кінці отримуємо частку та, за наявності остачу(remainder).

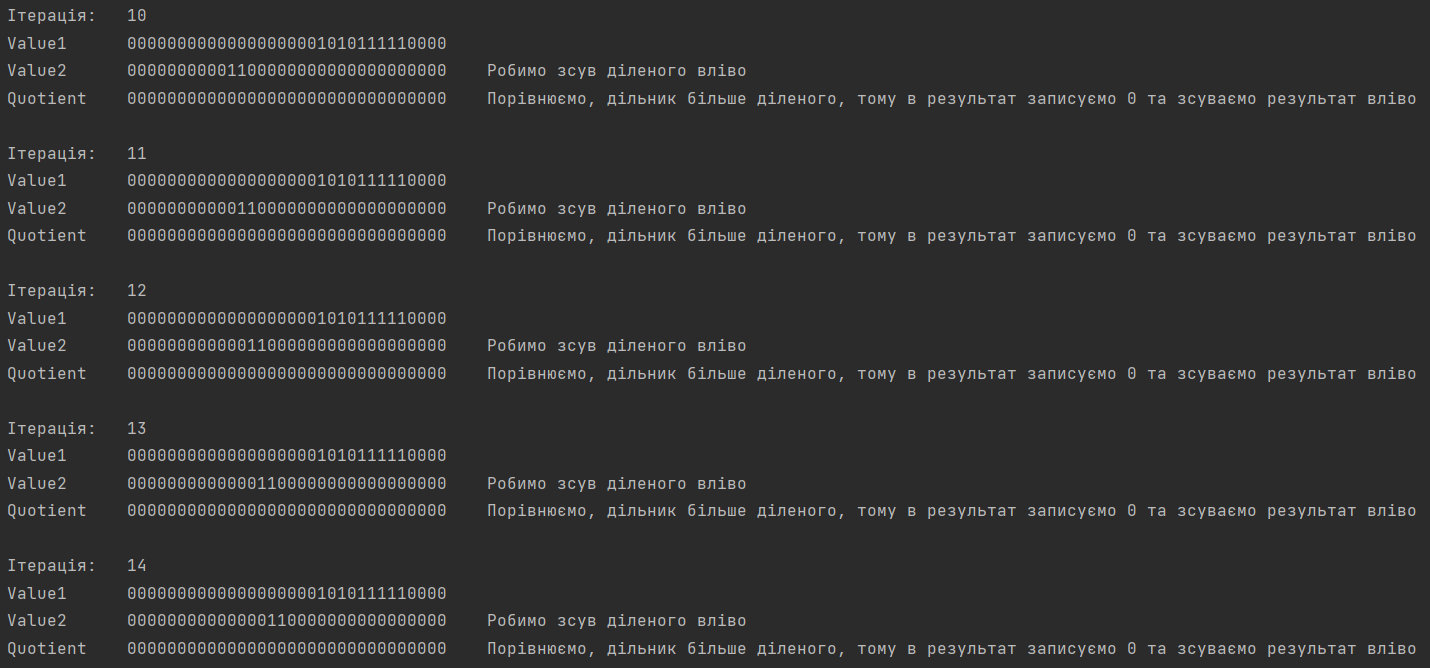
Перевірення роботи програми:

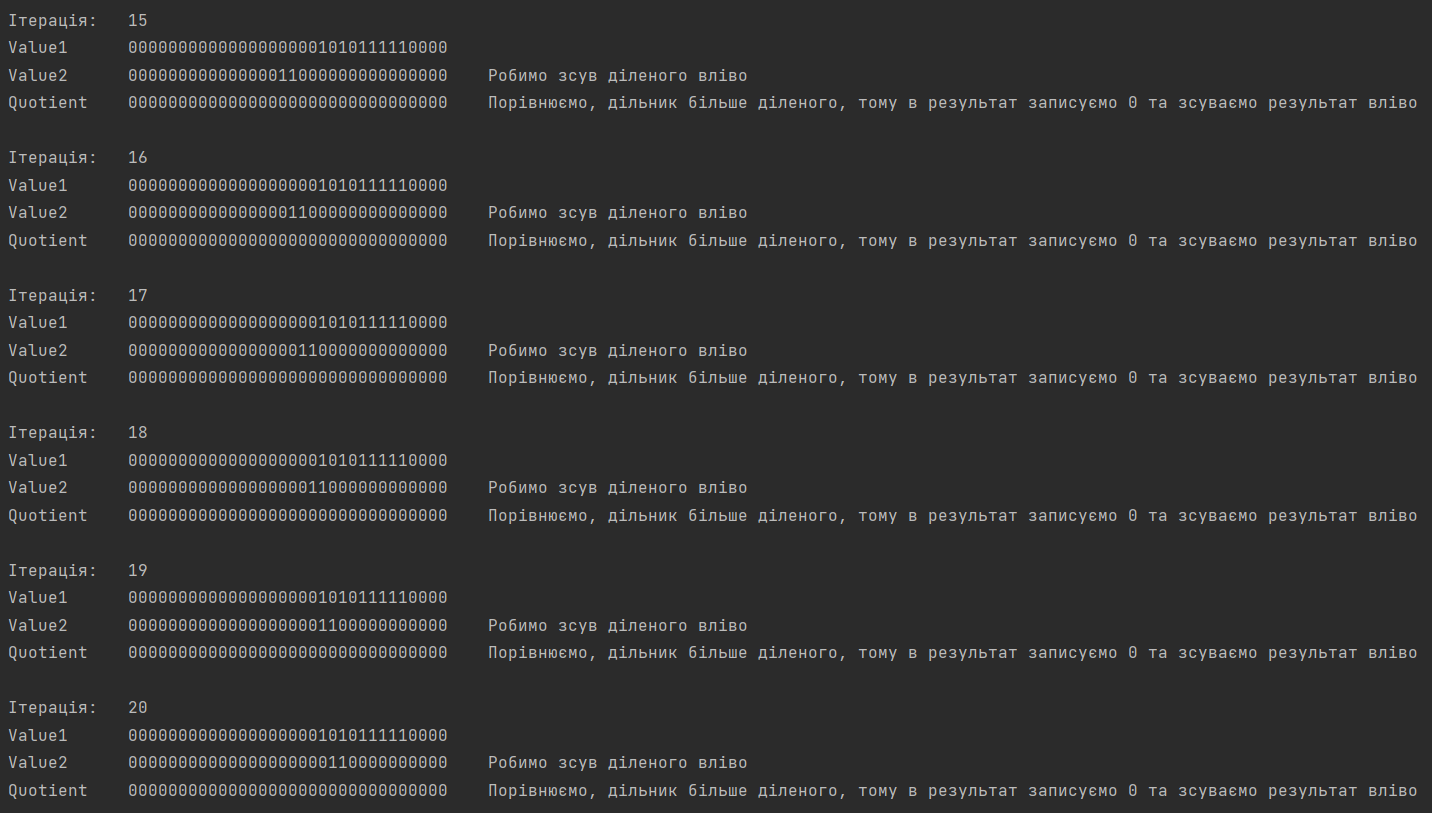
|  |  |  |  |
| --- | --- | --- | --- |
| Dividend | Divisor | Quotient | Remainder |
| 2000000000 00000000000000110000110101000000 | 19  00000000000000000000000000010011 | 105263157  00000110010001100011000000110101 | 17  000000000000000000000000000000000000000000000000000000  0000010001 |
| 20  00000000000000000000000000010100 | 2  00000000000000000000000000000010 | 10  00000000000000000000000000001010 | 0  0000000000000000000000000000000000000000000000000000000000000000 |
| 5616 | 3 | 1872  00000000000000000000011101010000 | 0  0000000000000000000000000000000000000000000000000000000000000000 |

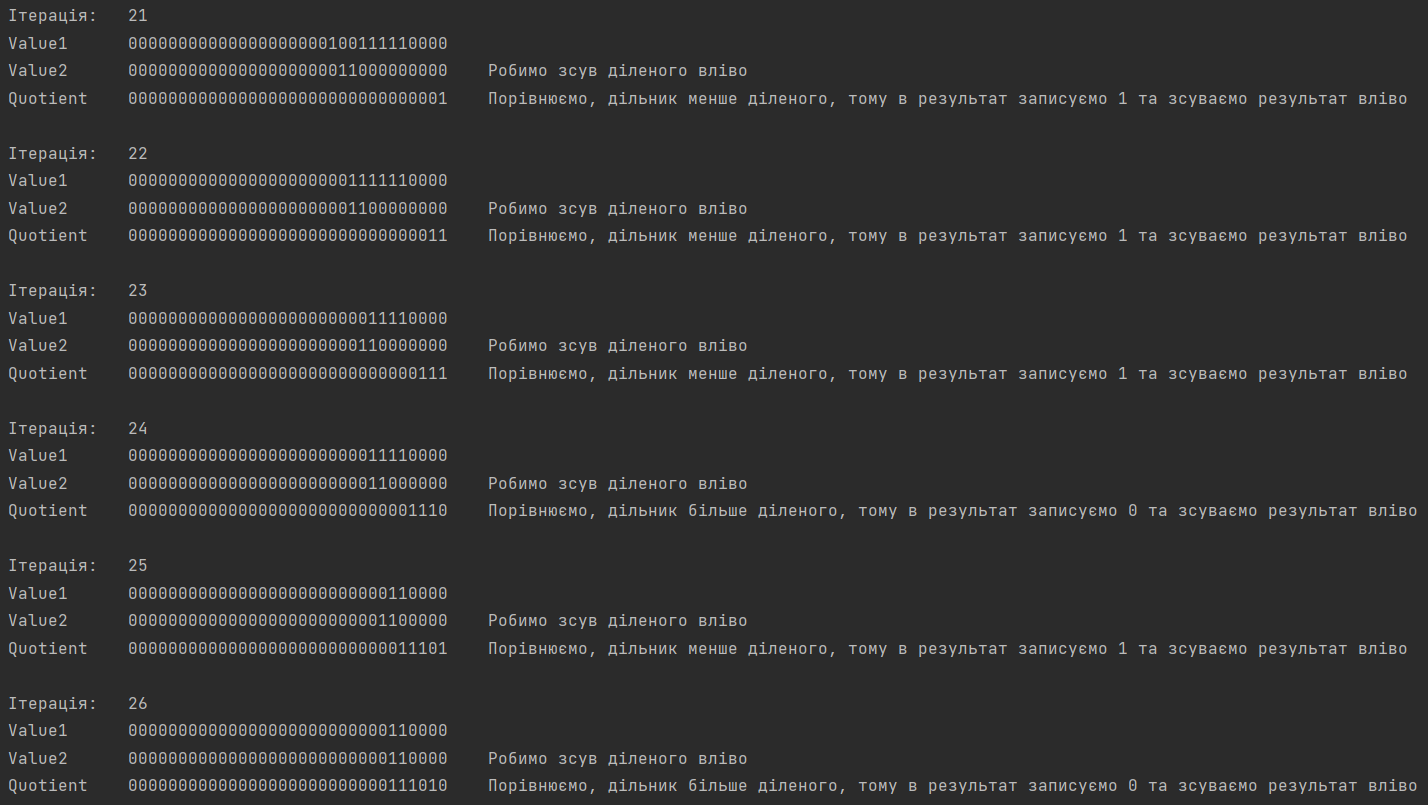
Виконання програми на прикладі чисел 5616 та 3:

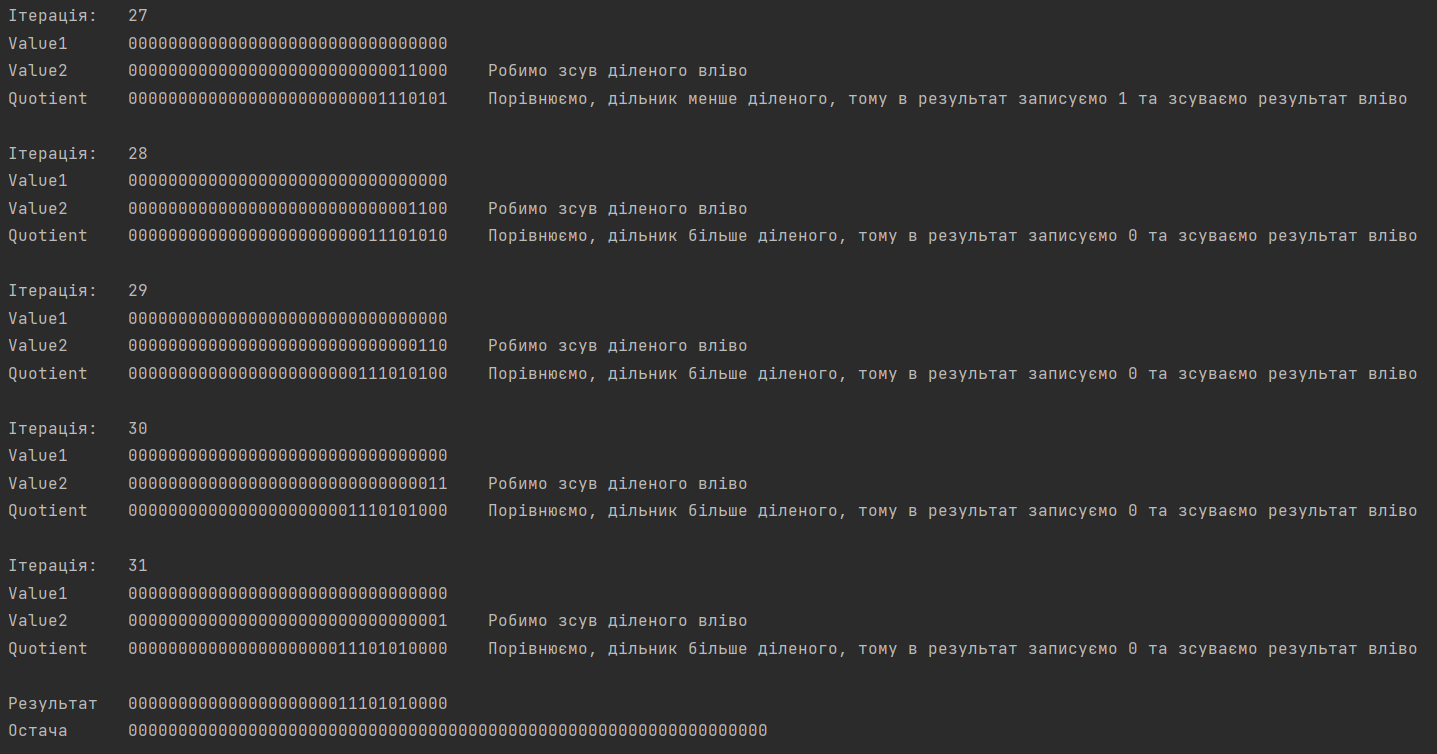




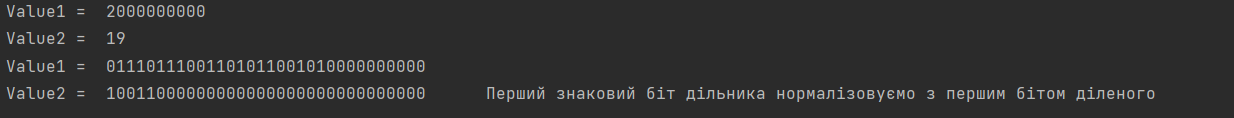




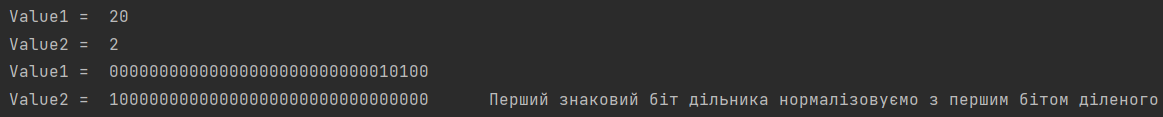


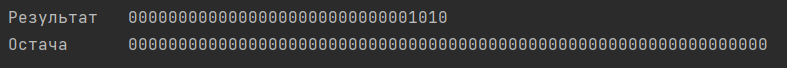


Інші випадки:

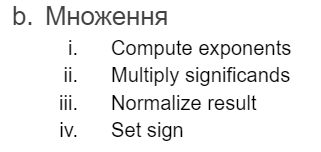








3. Створимо програму, що ілюструє покрокове виконання алгоритму множення двійкових чисел в IEEE 754 Floating Point:

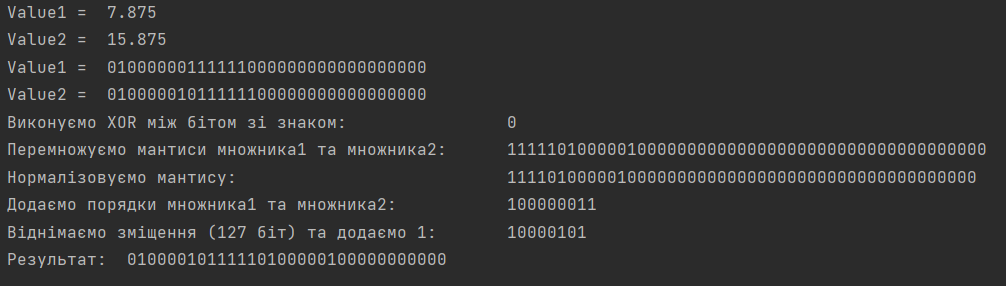


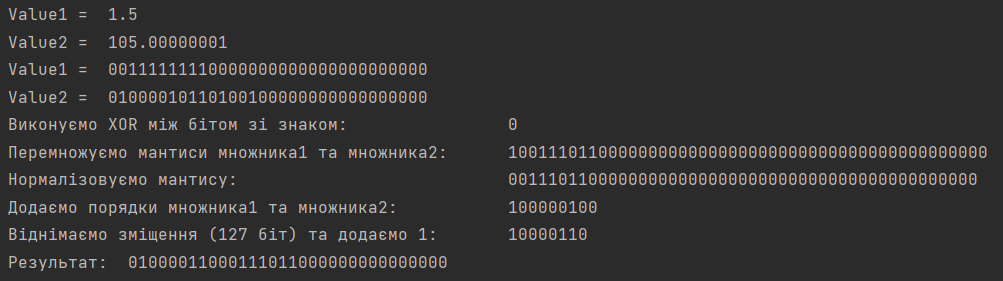
Пояснення алгоритму:  
1. Виконуємо XOR біта зі знаком множника1 із бітовим знаком множника2. Результат поміщаємо в результуючий знаковий біт.  
2. Перемножуємо мантиси м1 та м2 , а результат поміщаємо в результуюче поле мантиси.  
  
3. Додаємо біти експонент м1 та м2, а базове значення віднімаємо від 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.

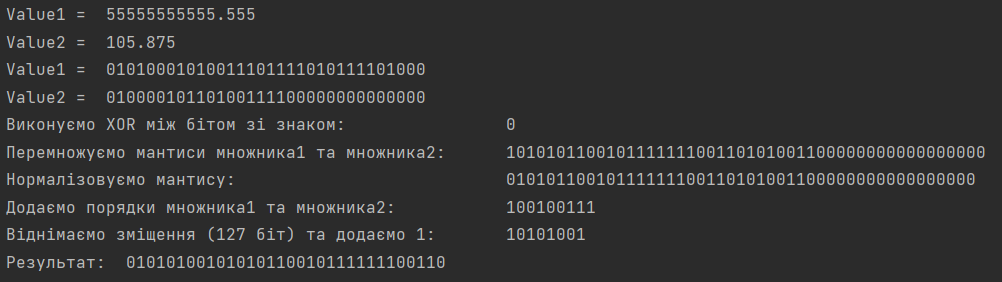
Перевірення роботи програми:

|  |  |  |
| --- | --- | --- |
| 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:







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.