

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО
Факультет обчислювальної техніки, інтелектуальних та управляючих систем
Кафедра інформаційних технологій

Звіт
з лабораторної роботи №4
Тема: Програмне моделювання машинних алгоритмів додавання
та віднімання чисел з плаваючою крапкою
Варіант: 7

Виконала:

студентка групи КС-242-1 Гук Марія Олегівна

Перевірив:

ст. викладач кафедри ПЗАС Мисник Б.В.

ЗАВДАННЯ

Програмно промодельовати машинний алгоритм віднімання чисел з плаваючою крапкою. Перший операнд представити у форматі 16 розрядів, другий – у форматі 8 розрядів.

ЛІСТИНГ ПРОГРАМИ

[GitHub](#)

```
using System;
using System.Reflection.Metadata;
using System.Text;
using System.Text.RegularExpressions;

namespace Lab4
{
    internal class Program
    {
        static void Main()
        {
            Console.InputEncoding = System.Text.Encoding.Unicode;
            Console.OutputEncoding = System.Text.Encoding.Unicode;
            //Ввід
            string num1, num2;
            Input(out num1, out num2);

            //Перевід в прямий код
            num1 = Transform(num1);
            if (num2[0] == '-') num2 = Transform(num2[1..]);
            else num2 = Transform('-' + num2);
            Console.WriteLine("КРОК 0 (Запис в прямому коді:");
```

```
Console.WriteLine($"Перше число в прямому коді: {num1}");  
Console.WriteLine($"Друге число в прямому коді: {num2} (Зміна знаку  
для виконання віднімання)");
```

```
Console.WriteLine("КРОК 1 (Урівноваження порядків доданків):");  
(int, int) power = PowerDifference(num1[18..], num2[10..]);  
Console.WriteLine($"Різниця між степенями: {power.Item2}. Менший  
ступінь в числі #{power.Item1}");  
if (power.Item1 == 1) //Зсув елементів в першому числі.  
{  
    int temp = 17 - power.Item2;  
    num1 = num1[0..2] + new string('0', power.Item2) + num1[2..temp] +  
num2[9..];  
    Console.WriteLine($"Зсуваємо перше число на {power.Item2}:  
{num1}");  
}  
else //В другому числі.  
{  
    int temp = 9 - power.Item2;  
    num2 = num2[0..2] + new string('0', power.Item2) + num2[2..temp] +  
num1[17..];  
    Console.WriteLine($"Зсуваємо друге число на {power.Item2}:  
{num2}");  
}
```

```
Console.WriteLine("КРОК 2 (Перетворення мантис чисел в додатковий код:");
```

```
num1 = ConvertCodes(num1[0..1] + num1[2..17]) + num1[17..];
```

```
num2 = ConvertCodes(num2[0..1] + num2[2..9]) + num2[9..];
```

```
num1 = num1[0..1] + "|" + num1[1..];
```

```
num2 = num2[0..1] + "|" + num2[1..];
```

```
Console.WriteLine($"Перше число в додатковому коді: {num1}");
```

```
Console.WriteLine($"Друге число в додатковому коді: {num2}");
```

```
Console.WriteLine("КРОК 3 (Додавання мантис:");
```

```
string sum = SumBinary(num1[0..1] + num1[2..17], num2[0..1] + new  
string('0', 8) + num2[2..9]);
```

```
if (sum.Length == 16) sum = sum[0..1] + "|" + sum[1..] + num2[9..];
```

```
else sum = sum[0..2] + "|" + sum[2..] + num2[9..];
```

```
Console.WriteLine($"Різниця цих чисел в додатковому коді: {sum}");
```

```
Console.WriteLine("КРОК 4 (Денормалізація результату:");
```

```
string power2 = num2[9..];
```

```
if (sum.IndexOf('|') == 2)
```

```
{
```

```
    power2 = "|" + SumBinary(num2[10..], "1");
```

```
    sum = sum[0..1] + "|" + sum[1..2] + sum[3..17] + power2;
```

```
    Console.WriteLine($"Має місце денормалізація результату вліво, тому  
зсуваємо вправо на 1: {sum}");
```

```
}
```

```
else if (sum[0] == sum[2])
```

```

{
    int cnt = 0;
    char sign='1';
    if (sum[0] == '1') sign = '0';
    Console.WriteLine($"Має місце денормалізація результату вправо,
тому зсуваємо вліво, поки не знайдемо {sign}:");
    while (sum[2] != sign && cnt<7)
    {
        cnt++;
        power2 = "|" + SumBinary(num2[10..], "-1");
        sum = sum[2..3] + "|" + sum[3..17] + '0' + power2;
        Console.WriteLine(sum);
    }
}
else Console.WriteLine("Уже нормалізований.");

```

```

Console.WriteLine("КРОК 5 (Подання результату):");
sum = ConvertCodes(sum[0..1] + sum[2..17]);
sum = sum[0..1] + "|" + sum[1..] + power2;
Console.WriteLine($"Різниця цих чисел в прямому коді: {sum}");
sum = TransformResult(sum);
Console.WriteLine($"Результат: {sum}");
}

```

```

private static void Input(out string num1, out string num2)
{
    do
    {

```

```
Console.WriteLine("Введіть перше бінарне число в такому вигляді: -  
0,001010101001111 * 2^10 (16 знаків).");
```

```
num1 = Console.ReadLine();  
string pattern = @"^[-]?0[.][01]{15}\s*\s2\^[01]+$";  
if (Regex.IsMatch(num1, pattern)) break;  
else Console.WriteLine("Помилка! Спробуйте ще раз.");  
}
```

```
while (true);
```

```
do
```

```
{
```

```
Console.WriteLine("Введіть друге бінарне число в такому вигляді:  
0,0011101 * 2^01 (8 знаків).");
```

```
num2 = Console.ReadLine();  
string pattern = @"^[-]?0[.][01]{7}\s*\s2\^[01]+$";  
if (Regex.IsMatch(num2, pattern)) break;  
else Console.WriteLine("Помилка! Спробуйте ще раз.");  
}
```

```
while (true);
```

```
}
```

```
static string Transform(string num)
```

```
{
```

```
StringBuilder result = new StringBuilder();  
int i = 0;  
result.Append(num[i] == '-' ? "1" : "0");  
i += (num[i] == '-') ? 3 : 2;  
while (num[i] != ' ') result.Append(num[i++]);  
i += 5;  
result.Append("|" + num[i..]);  
return result.ToString();
```

```

}
static string TransformResult(string num)
{
    StringBuilder result = new StringBuilder();
    int i = 0;
    result.Append(num[i] == '1' ? "-0," : "0,");
    i += 2;
    while (num[i] != '|') result.Append(num[i++]);
    i++;
    result.Append(" * 2^" + num[i..]);
    return result.ToString();
}

static (int, int) PowerDifference(string num1, string num2)
{
    int temp1 = Convert.ToInt32(num1, 2);
    int temp2 = Convert.ToInt32(num2, 2);

    int diff = Math.Abs(temp1 - temp2);

    return ((temp1 < temp2) ? 1 : 2, diff);
}

static string ConvertCodes(string directCode)
{
    if (directCode[0] == '0') return directCode;
    char[] additionalCode = directCode.ToCharArray();
    bool foundOne = false;
    for (int i = additionalCode.Length - 1; i > 0; i--)
    {
        if (foundOne)
        {

```

```

        additionalCode[i] = additionalCode[i] == '0' ? '1' : '0';
    }
    if (additionalCode[i] == '1' && !foundOne)
    {
        foundOne = true;
    }
}
return new string(additionalCode);
}
static string SumBinary(string num1, string num2)
{
    int maxLength = Math.Max(num1.Length, num2.Length);
    num1 = num1.PadLeft(maxLength, '0');
    num2 = num2.PadLeft(maxLength, '0');

    char remember = '0';
    char[] result = new char[maxLength];

    for (int i = maxLength - 1; i >= 0; i--)
    {
        int sum = (num1[i] - '0') + (num2[i] - '0') + (remember - '0');
        if (sum >= 2)
        {
            remember = '1';
            sum -= 2;
        }
        else
        {
            remember = '0';
        }
    }
}

```



```
        result[i] = (sum == 0) ? '0' : '1';  
    }  
  
    if (remember == '1') return "1" + new string(result);  
    else return new string(result);  
}  
}  
}
```

РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

1. Виконання програми (Рис. 1.1) та (Рис. 1.2):

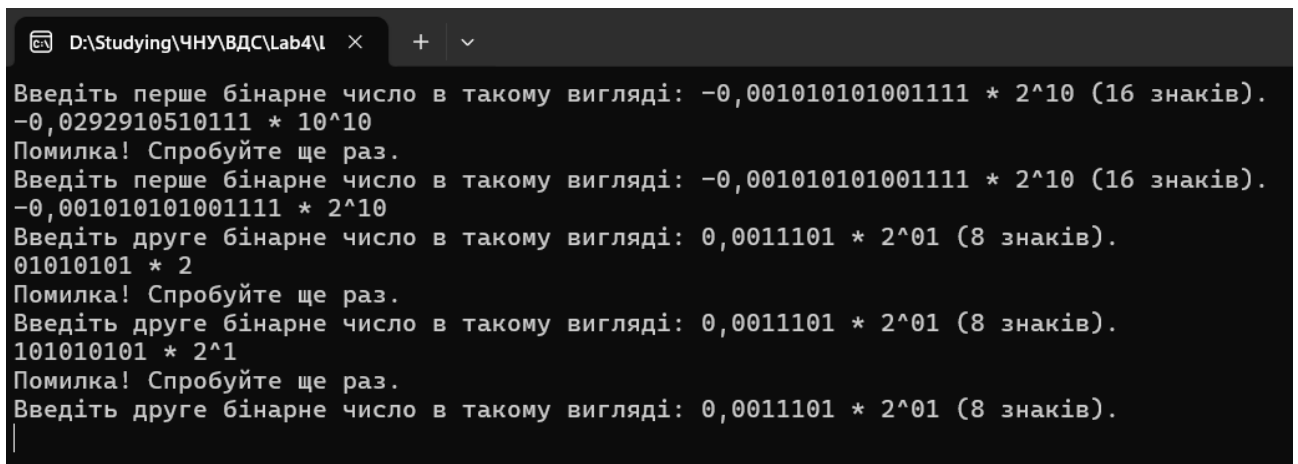
```
Microsoft Visual Studio Debu  X + v
Введіть перше бінарне число в такому вигляді: -0,001010101001111 * 2^10 (16 знаків).
-0,001010101001111 * 2^10
Введіть друге бінарне число в такому вигляді: 0,0011101 * 2^01 (8 знаків).
0,0011101 * 2^01
КРОК 0 (Запис в прямому коді):
Перше число в прямому коді: 1|001010101001111|10
Друге число в прямому коді: 1|0011101|01 (Зміна знаку для виконання віднімання)
КРОК 1 (Урівноваження порядків доданків):
Різниця між степенями: 1. Менший степінь в числі #2
Зсуваємо друге число на 1: 1|0001110|10
КРОК 2 (Перетворення мантис чисел в додатковий код):
Перше число в додатковому коді: 1|110101010110001|10
Друге число в додатковому коді: 1|1110010|10
КРОК 3 (Додавання мантис):
Різниця цих чисел в додатковому коді: 10|110101100100011|10
КРОК 4 (Денормалізація результату):
Має місце денормалізація результату вліво, тому зсуваємо вправо на 1: 1|011010110010001|11
КРОК 5 (Подання результату):
Різниця цих чисел в прямому коді: 1|100101001101111|11
Результат: -0,100101001101111 * 2^11
```

Рис 1.1. Приклад виконання програми в консолі.

```
Microsoft Visual Studio Debu  X + v
Введіть перше бінарне число в такому вигляді: -0,001010101001111 * 2^10 (16 знаків).
0,101010010101111 * 2^101
Введіть друге бінарне число в такому вигляді: 0,0011101 * 2^01 (8 знаків).
0,0001111 * 2^010
КРОК 0 (Запис в прямому коді):
Перше число в прямому коді: 0|101010010101111|101
Друге число в прямому коді: 1|0001111|010 (Зміна знаку для виконання віднімання)
КРОК 1 (Урівноваження порядків доданків):
Різниця між степенями: 3. Менший степінь в числі #2
Зсуваємо друге число на 3: 1|0000001|101
КРОК 2 (Перетворення мантис чисел в додатковий код):
Перше число в додатковому коді: 0|101010010101111|101
Друге число в додатковому коді: 1|1111111|101
КРОК 3 (Додавання мантис):
Різниця цих чисел в додатковому коді: 1|101010100101110|101
КРОК 4 (Денормалізація результату):
Має місце денормалізація результату вправо, тому зсуваємо вліво, поки не знайдемо 0:
1|010101001011100|110
КРОК 5 (Подання результату):
Різниця цих чисел в прямому коді: 1|101010110100100|110
Результат: -0,101010110100100 * 2^110
```

Рис 1.2. Приклад виконання програми в консолі.

2. Виконання програми при помилках у введенні (Рис. 1.3);



```
D:\Studying\ЧНУ\ВДС\Lab4\l  × + v
Введіть перше бінарне число в такому вигляді: -0,001010101001111 * 2^10 (16 знаків).
-0,0292910510111 * 10^10
Помилка! Спробуйте ще раз.
Введіть перше бінарне число в такому вигляді: -0,001010101001111 * 2^10 (16 знаків).
-0,001010101001111 * 2^10
Введіть друге бінарне число в такому вигляді: 0,0011101 * 2^01 (8 знаків).
01010101 * 2
Помилка! Спробуйте ще раз.
Введіть друге бінарне число в такому вигляді: 0,0011101 * 2^01 (8 знаків).
101010101 * 2^1
Помилка! Спробуйте ще раз.
Введіть друге бінарне число в такому вигляді: 0,0011101 * 2^01 (8 знаків).
|
```

Рис. 1.3. Приклад виконання програми в консолі при помилках у введенні.

ВИСНОВОК

У ході виконання лабораторної роботи №4 було програмно змодельовано машинний алгоритм віднімання чисел у форматі з плаваючою крапкою. Було реалізовано операцію для операндів різної розрядності: 16 біт для першого та 8 біт для другого.

Реалізована програма успішно виконує віднімання, що включає етапи вирівнювання порядків, перетворення мантис у відповідний код для додавання, безпосереднє додавання мантис та нормалізацію отриманого результату. Програма також коректно обробляє введені дані, у тому числі випадки помилкового введення, що було продемонстровано під час тестування.

Отримані результати підтвердили правильність реалізації змодельованого алгоритму та основних кроків обробки чисел з плаваючою крапкою. Виконана робота сприяла глибшому розумінню процесів машинної арифметики з плаваючою крапкою, особливостей роботи з різними форматами представлення чисел, та їхнього програмного моделювання.