Київський національний університет імені Тараса Шевченка
радіофізичний факультет

Звіт до
лабораторної роботи № 2
з предмету «Комп’ютерні системи»
Тема: «Арифметичні операції над двійковими числами»

Роботу виконала
студентка 3 курсу
КІ-СА
Олішевська Олена

Київ 2019

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

Репозиторій: https://github.com/helishe/CS2020

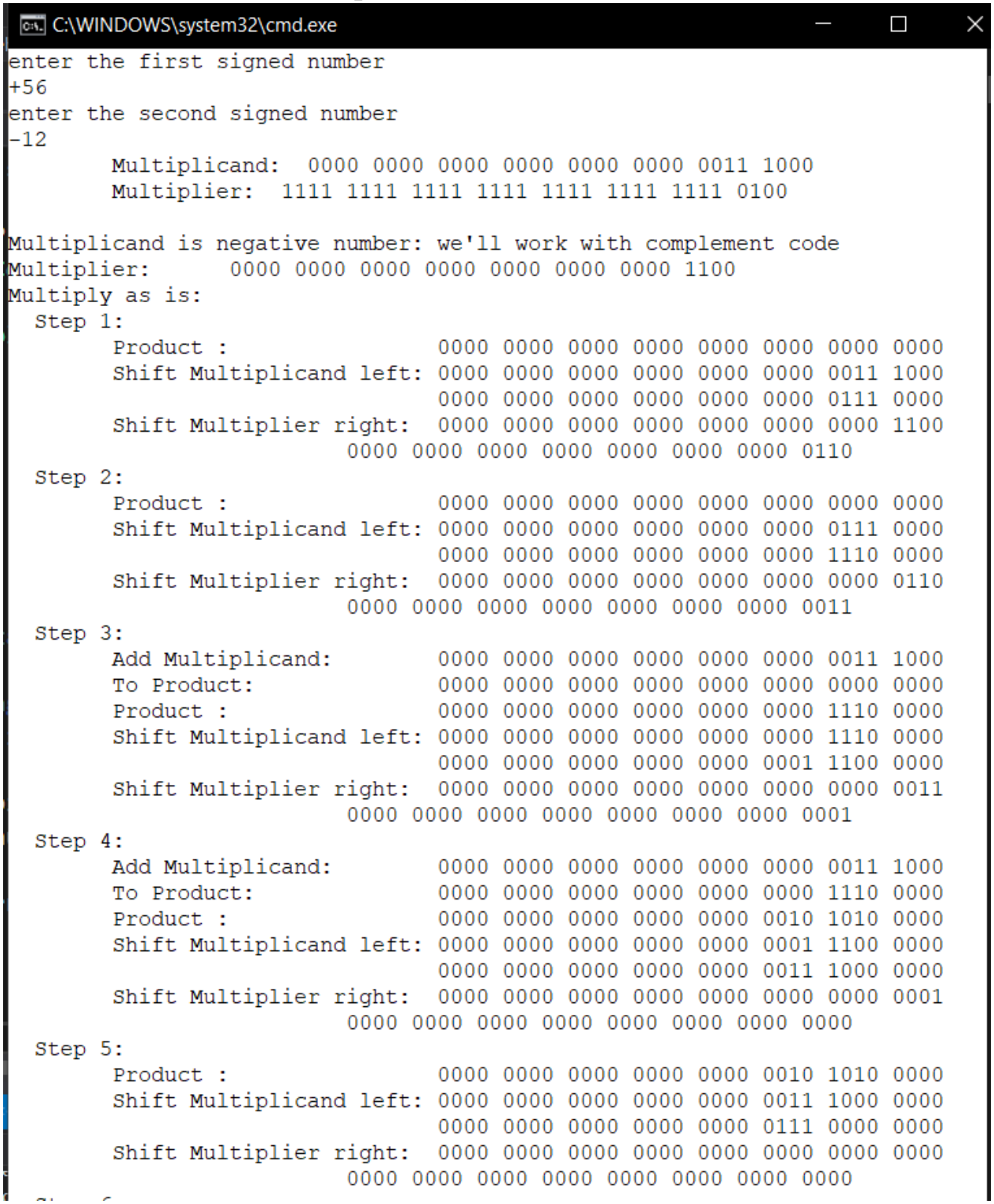
Варіанти завдань: 1 –а, 2 – а, 3 - а

Хід роботи:
Створити програму, що ілюструє покрокове виконання наступних алгоритмів (за варіантами в Moodle).

Під покроковим виконанням мається на увазі вивід в двійковому представленні значень регістрів, що використовуються в процесі обрахунку на кожній ітерації, а також виводу самої логіки роботи алгоритму у вигляді опису (наприклад: “Значення регістру DIVISOR > 0: додаємо біт 0 до QUOTIENT, сзуваємо….”).

Код завантажте в свій репозиторій в GitHub.
В звіті навести приклад покрокового виконання кожного з варіантів, посилання на код та завантажити в Moodle.

Частина 1
Множення двійкових чисел
Варіант: *а) множення як є*



```
Step 6:
      Product :                 0000 0000 0000 0000 0000 0010 1010 0000
      Shift Multiplicand left: 0000 0000 0000 0000 0000 0111 0000 0000
                               0000 0000 0000 0000 0000 1110 0000 0000
      Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
Step 7:
      Product :                 0000 0000 0000 0000 0000 0010 1010 0000
      Shift Multiplicand left: 0000 0000 0000 0000 0000 1110 0000 0000
                               0000 0000 0000 0000 0001 1100 0000 0000
      Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
Step 8:
      Product :                 0000 0000 0000 0000 0000 0010 1010 0000
      Shift Multiplicand left: 0000 0000 0000 0000 0001 1100 0000 0000
                               0000 0000 0000 0000 0011 1000 0000 0000
      Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
Step 9:
      Product :                 0000 0000 0000 0000 0000 0010 1010 0000
      Shift Multiplicand left: 0000 0000 0000 0000 0011 1000 0000 0000
                               0000 0000 0000 0000 0111 0000 0000 0000
      Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
Step 10:
      Product :                 0000 0000 0000 0000 0000 0010 1010 0000
      Shift Multiplicand left: 0000 0000 0000 0000 0111 0000 0000 0000
                               0000 0000 0000 0000 1110 0000 0000 0000
      Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
Step 11:
      Product :                 0000 0000 0000 0000 0000 0010 1010 0000
      Shift Multiplicand left: 0000 0000 0000 0000 1110 0000 0000 0000
                               0000 0000 0000 0001 1100 0000 0000 0000
      Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
Step 12:
      Product :                 0000 0000 0000 0000 0000 0010 1010 0000
      Shift Multiplicand left: 0000 0000 0000 0001 1100 0000 0000 0000
                               0000 0000 0000 0011 1000 0000 0000 0000
      Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
Step 13:
      Product :                 0000 0000 0000 0000 0000 0010 1010 0000
      Shift Multiplicand left: 0000 0000 0000 0011 1000 0000 0000 0000
                               0000 0000 0000 0111 0000 0000 0000 0000
      Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
```

```
                        0000 0000 0000 0000 0000 0000 0000 0000
 Step 14:
       Product :                0000 0000 0000 0000 0000 0010 1010 0000
       Shift Multiplicand left: 0000 0000 0000 0111 0000 0000 0000 0000
                                0000 0000 0000 1110 0000 0000 0000 0000
       Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
 Step 15:
       Product :                0000 0000 0000 0000 0000 0010 1010 0000
       Shift Multiplicand left: 0000 0000 0000 1110 0000 0000 0000 0000
                                0000 0000 0001 1100 0000 0000 0000 0000
       Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
 Step 16:
       Product :                0000 0000 0000 0000 0000 0010 1010 0000
       Shift Multiplicand left: 0000 0000 0001 1100 0000 0000 0000 0000
                                0000 0000 0011 1000 0000 0000 0000 0000
       Shift Multiplier right:  0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000

 Result is complement code, because our multiplicands have different sign
 Answer is:
       In decemal: -672
       In binary:  1111 1111 1111 1111 1111 1101 0110 0000
Press any key to continue . . .
```

Частина 2
Ділення двійкових чисел

Варіант: *а) ділення як є*

C:\WINDOWS\system32\cmd.exe

```
Enter the sign divident
+912
Enter the sign divisor
-6
def = 1110010000 mod Divident
110 mod Divisor

Divisor align
1110010000 mod Divident
1100000000 mod Divisor

Add Divident & Divisor in additional code
Quotient = 10010000
Remainder = 1

110000000  Divisor Right Shift
11111111111111111111111100010000 Sub Qoutient & Divisor
10 Remainder Left Shift

11000000  Divisor Right Shift
11111111111111111111111111010000 Sum Qoutient & Divisor
100 Remainder Left Shift

1100000  Divisor Right Shift
110000 Sum Qoutient & Divisor
1001 Remainder Left Shift & add 1

110000  Divisor Right Shift
0 Sub Qoutient & Divisor
10011 Remainder Left Shift & add 1

11000  Divisor Right Shift
11111111111111111111111111101000 Sub Qoutient & Divisor
100110 Remainder Left Shift

1100  Divisor Right Shift
11111111111111111111111111110100 Sum Qoutient & Divisor
1001100 Remainder Left Shift

110  Divisor Right Shift
11111111111111111111111111111010 Sum Qoutient & Divisor
10011000 Remainder Left Shift

Determine Qoutient
0 Qoutient += |Divisor|

Analyze sign bit of Dividend & Divisor and set sign bit to Remainder and Quotient
Remainder = 11111111111111111111111101101000 Quotient = 0

Remainder = -152, Quotient = 0
 Press any key to continue . . .
```

Частина 3
Робота з IEEE 754 Floating Point (Представити лише ключові кроки при виконанні операцій)

Варіант: *Додавання*

Align binary points

Add significands

Normalize result