Отчет по 3 лабораторной работе.

Команда: Холодов А.В. Иванов А.А. ПИН-21

Вывод в консоли:

1 задание – изменен код.

Задание 2

```
Exercise 2
For 986:
         binary| hex| un decimal|    decimal
0000001111011010| 3da|        986|        986
                         binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000001111011010|     3da|        986|        986|  0.0000|1.3817e-42

         binary| hex| un decimal|    decimal
0000001111011010| 3da|        986|        986
                         binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000001111011010|     3da|        986|        986|  0.0000|1.3817e-42
For -126:
         binary| hex| un decimal|    decimal
1111111110000010|ff82|      65410|       -126
                         binary|     hex| un decimal|    decimal|    float| exponent
11111111111111111111111110000010|ffffff82| 4294967170|       -126|      nan|      nan

         binary| hex| un decimal|    decimal
1111111110000010|ff82|      65410|       -126
                         binary|     hex| un decimal|    decimal|    float| exponent
00000000000000001111111110000010|    ff82|      65410|      65410|  0.0000|9.1659e-41
```

Задание 3

```
For m:
Initial m:
          binary| hex| un decimal|    decimal
0000001111011010| 3da|         986|         986
Signed m*2:
          binary| hex| un decimal|    decimal
0000011110110100| 7b4|        1972|        1972
Unsigned m*2:
          binary| hex| un decimal|    decimal
0000011110110100| 7b4|        1972|        1972
Signed m/2:
          binary| hex| un decimal|    decimal
0000000111101101| 1ed|         493|         493
Unsigned m/2:
          binary| hex| un decimal|    decimal
0000000111101101| 1ed|         493|         493
m mod 16:
          binary| hex| un decimal|    decimal
0000000000001010|   a|          10|          10
Rounding down:
          binary| hex| un decimal|    decimal
0000001111010000| 3d0|         976|         976

For n:
Initial n:
          binary| hex| un decimal|    decimal
1111111110000010|ff82|       65410|        -126
Signed m*2:
          binary| hex| un decimal|    decimal
1111111100000100|ff04|       65284|        -252
Unsigned m*2:
          binary| hex| un decimal|    decimal
1111111100000100|ff04|       65284|        -252
Signed m/2:
          binary| hex| un decimal|    decimal
1111111111000001|ffc1|       65473|         -63
Unsigned m/2:
          binary| hex| un decimal|    decimal
0111111111000001|7fc1|       32705|       32705
m mod 16:
          binary| hex| un decimal|    decimal
0000000000000010|   2|           2|           2
Rounding down:
          binary| hex| un decimal|    decimal
1111111101111000|ff78|       65400|        -136
```

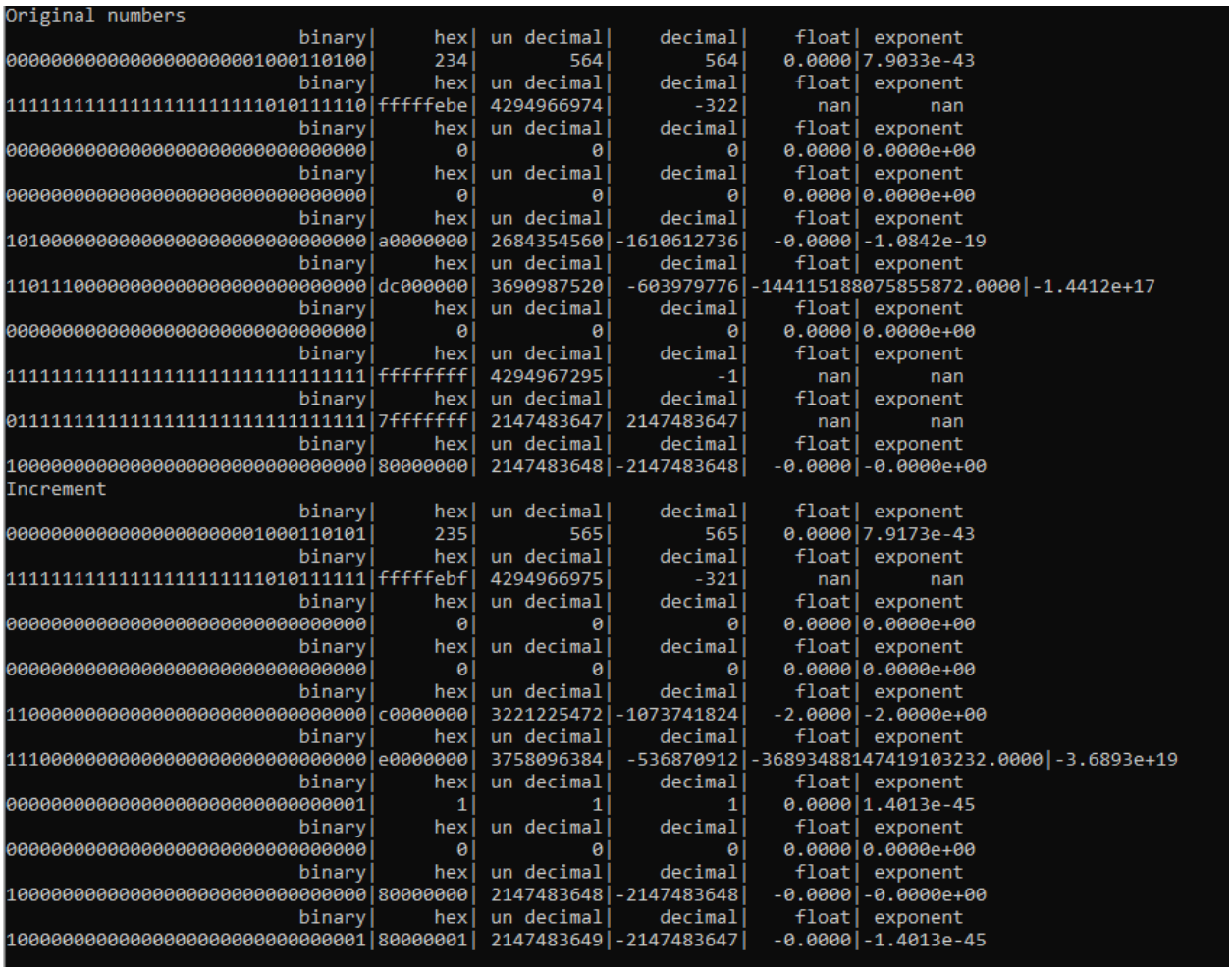
Задание 4.

```
Initial data
          binary| hex| un decimal|    decimal
0000001111011010| 3da|        986|        986
          binary| hex| un decimal|    decimal
1111111110000010|ff82|      65410|       -126
          binary| hex| un decimal|    decimal
0000001111011010| 3da|        986|        986
          binary| hex| un decimal|    decimal
1111111110000010|ff82|      65410|       -126
Sign shift left by 1 bit
          binary| hex| un decimal|    decimal
0000011110110100| 7b4|       1972|       1972
          binary| hex| un decimal|    decimal
1111111100000100|ff04|      65284|       -252
Unsigned left shift by 1 bit
          binary| hex| un decimal|    decimal
0000011110110100| 7b4|       1972|       1972
          binary| hex| un decimal|    decimal
1111111100000100|ff04|      65284|       -252
Sign shift right by 1 bit
          binary| hex| un decimal|    decimal
0000000111101101| 1ed|        493|        493
          binary| hex| un decimal|    decimal
1111111111000001|ffc1|      65473|        -63
Unsigned right shift by 1 bit
          binary| hex| un decimal|    decimal
0000000111101101| 1ed|        493|        493
          binary| hex| un decimal|    decimal
0111111111000001|7fc1|      32705|      32705
X & 15
          binary| hex| un decimal|    decimal
0000000000001010|   a|          10|         10
          binary| hex| un decimal|    decimal
0000000000000010|   2|           2|          2
X & -16
          binary| hex| un decimal|    decimal
0000001111010000| 3d0|        976|        976
          binary| hex| un decimal|    decimal
1111111110000000|ff80|      65408|       -128
```

Задание 5

```
Enter the number
654
Rounding down: 640
Rounding up: 704
```

Задание 6 (инкремент)



```
Original numbers
                            binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000001000110100|     234|        564|        564|   0.0000|7.9033e-43
                            binary|     hex| un decimal|    decimal|    float| exponent
11111111111111111111111010111110|fffffebe| 4294966974|       -322|      nan|      nan
                            binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                            binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                            binary|     hex| un decimal|    decimal|    float| exponent
10100000000000000000000000000000|a0000000| 2684354560|-1610612736|  -0.0000|-1.0842e-19
                            binary|     hex| un decimal|    decimal|    float| exponent
11011100000000000000000000000000|dc000000| 3690987520| -603979776|-144115188075855872.0000|-1.4412e+17
                            binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                            binary|     hex| un decimal|    decimal|    float| exponent
11111111111111111111111111111111|ffffffff| 4294967295|         -1|      nan|      nan
                            binary|     hex| un decimal|    decimal|    float| exponent
01111111111111111111111111111111|7fffffff| 2147483647| 2147483647|      nan|      nan
                            binary|     hex| un decimal|    decimal|    float| exponent
10000000000000000000000000000000|80000000| 2147483648|-2147483648|  -0.0000|-0.0000e+00
Increment
                            binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000001000110101|     235|        565|        565|   0.0000|7.9173e-43
                            binary|     hex| un decimal|    decimal|    float| exponent
11111111111111111111111010111111|fffffebf| 4294966975|       -321|      nan|      nan
                            binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                            binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                            binary|     hex| un decimal|    decimal|    float| exponent
11000000000000000000000000000000|c0000000| 3221225472|-1073741824|  -2.0000|-2.0000e+00
                            binary|     hex| un decimal|    decimal|    float| exponent
11100000000000000000000000000000|e0000000| 3758096384| -536870912|-36893488147419103232.0000|-3.6893e+19
                            binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000001|       1|          1|          1|   0.0000|1.4013e-45
                            binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                            binary|     hex| un decimal|    decimal|    float| exponent
10000000000000000000000000000000|80000000| 2147483648|-2147483648|  -0.0000|-0.0000e+00
                            binary|     hex| un decimal|    decimal|    float| exponent
10000000000000000000000000000001|80000001| 2147483649|-2147483647|  -0.0000|-1.4013e-45
```

Задание 6 (декремент)

```
Original numbers
                          binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000001000110100|     234|        564|        564|   0.0000|7.9033e-43
                          binary|     hex| un decimal|    decimal|    float| exponent
11111111111111111111111010111110|fffffebe| 4294966974|       -322|      nan|      nan
                          binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                          binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                          binary|     hex| un decimal|    decimal|    float| exponent
10100000000000000000000000000000|a0000000| 2684354560|-1610612736|  -0.0000|-1.0842e-19
                          binary|     hex| un decimal|    decimal|    float| exponent
11011100000000000000000000000000|dc000000| 3690987520| -603979776|-144115188075855872.0000|-1.4412e+17
                          binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                          binary|     hex| un decimal|    decimal|    float| exponent
11111111111111111111111111111111|ffffffff| 4294967295|         -1|      nan|      nan
                          binary|     hex| un decimal|    decimal|    float| exponent
01111111111111111111111111111111|7fffffff| 2147483647| 2147483647|      nan|      nan
                          binary|     hex| un decimal|    decimal|    float| exponent
10000000000000000000000000000000|80000000| 2147483648|-2147483648|  -0.0000|-0.0000e+00
Decrement
                          binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000001000110011|     233|        563|        563|   0.0000|7.8893e-43
                          binary|     hex| un decimal|    decimal|    float| exponent
11111111111111111111111010111101|fffffebd| 4294966973|       -323|      nan|      nan
                          binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                          binary|     hex| un decimal|    decimal|    float| exponent
00000000000000000000000000000000|       0|          0|          0|   0.0000|0.0000e+00
                          binary|     hex| un decimal|    decimal|    float| exponent
10000000000000000000000000000000|80000000| 2147483648|-2147483648|  -0.0000|-0.0000e+00
                          binary|     hex| un decimal|    decimal|    float| exponent
11011000000000000000000000000000|d8000000| 3623878656| -671088640|-562949953421312.0000|-5.6295e+14
                          binary|     hex| un decimal|    decimal|    float| exponent
11111111111111111111111111111111|ffffffff| 4294967295|         -1|      nan|      nan
                          binary|     hex| un decimal|    decimal|    float| exponent
11111111111111111111111111111110|fffffffe| 4294967294|         -2|      nan|      nan
                          binary|     hex| un decimal|    decimal|    float| exponent
01111111111111111111111111111110|7ffffffe| 2147483646| 2147483646|      nan|      nan
                          binary|     hex| un decimal|    decimal|    float| exponent
01111111111111111111111111111111|7fffffff| 2147483647| 2147483647|      nan|      nan
```

Контрольные вопросы:

1.

Код со сдвигом [править]

При использовании **кода со сдвигом** (англ. *Offset binary*) целочисленный отрезок от нуля до 2^n (n — количество бит) сдвигается влево на 2^{n-1}, а затем получившиеся на этом отрезке числа последовательно кодируются в порядке возрастания кодами от $000\ldots0$ до $111\ldots1$. Например, число -5 в восьмибитном типе данных, использующем код со сдвигом, превратится в $-5 + 128 = 123$, то есть будет выглядеть так: 01111011.

По сути, при таком кодировании:

- к кодируемому числу прибавляют 2^{n-1};
- переводят получившееся число в двоичную систему исчисления.

Можно получить диапазон значений $[-2^{n-1}; 2^{n-1} - 1]$.

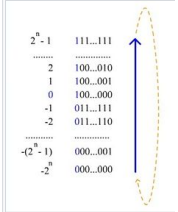
Достоинства представления чисел с помощью кода со сдвигом [править]

1. Не требуется усложнение архитектуры процессора.
2. Нет проблемы двух нулей.

Недостатки представления чисел с помощью кода со сдвигом [править]

1. При арифметических операциях нужно учитывать смещение, то есть проделывать на одно действие больше (например, после «обычного» сложения двух чисел у результата будет двойное смещение, одно из которых необходимо вычесть).
2. Ряд положительных и отрицательных чисел несимметричен.

Из-за необходимости усложнять арифметические операции код со сдвигом для представления целых чисел используется не часто, но зато применяется для хранения порядка вещественного числа.



Код со сдвигом. Как видно, двоичное представление зациклено по модулю $1000\ldots000_{(2)}$ (n нулей)

2.

Принцип работы [править]

Логические побитовые операции [править]

Битовые операторы И (AND, &), ИЛИ (OR, |), НЕ (NOT, $\sim$) и исключающее ИЛИ (XOR, $\textasciicircum$, $\oplus$) используют те же таблицы истинности, что и их логические эквиваленты.

Побитовое И [править]

Побитовое И используется для выключения битов. Любой бит, установленный в 0, вызывает установку соответствующего бита результата также в 0.

&	11001010 11100010
	11000010

Побитовое ИЛИ [править]

Побитовое ИЛИ используется для включения битов. Любой бит, установленный в 1, вызывает установку соответствующего бита результата также в 1.

|	11001010 11100010
	11101010

Побитовое НЕ [править]

Побитовое НЕ инвертирует состояние каждого бита исходной переменной.

~	11001010
	00110101

Побитовое исключающее ИЛИ [править]

Исключающее ИЛИ устанавливает значение бита результата в 1, если значения в соответствующих битах исходных переменных различны.

^	11001010 11100010
	00101000

Побитовые сдвиги [править]

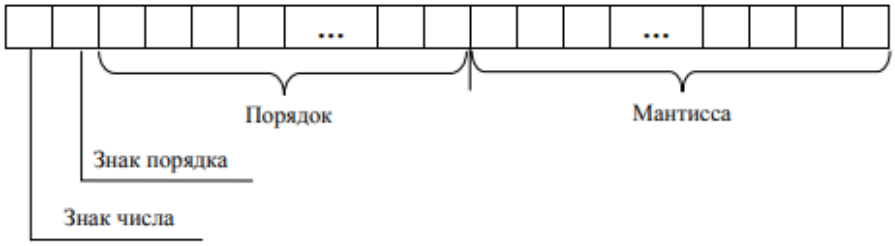
Операторы сдвига $<<$ и $>>$ сдвигают биты в переменной влево или вправо на указанное число. При этом на освободившиеся позиции устанавливаются нули (кроме сдвига вправо отрицательного числа, в этом случае на свободные позиции устанавливаются единицы, так как числа представляются в двоичном дополнительном коде и необходимо поддерживать знаковый бит).

Сдвиг влево может применяться для умножения числа на два, сдвиг вправо — для деления.

3.

4.2 Представление в виде набора битов

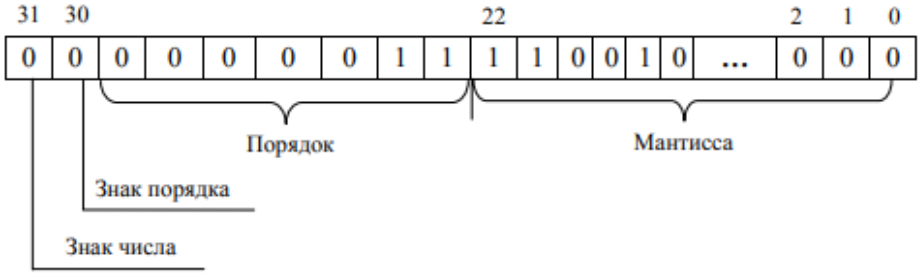
Числа с плавающей точкой представляются в виде битовых наборов, в которых отводяся разряды для мантиссы, порядка, знака числа и знака порядка:



Чем больше разрядов отводится под запись мантиссы, тем выше точность представления числа. Чем больше разрядов занимает порядок, тем шире диапазон от наименьшего отличного от нуля числа до наибольшего числа, представимого в машине при заданном формате.

Покажем на примерах, как записываются некоторые числа в нормализованном виде в четырехбайтовом формате с семью разрядами для записи порядка.

Число $6.25_{10} = 110.01_2 = 0.11001 \cdot 2^{11}$:



Число $-0.125_{10} = -0.001_2 = -0.1 \cdot 2^{-10}$ (отрицательный порядок записан в дополнительном коде):