федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Московский институт электронной техники»

Направление подготовки 09.03.04 «Программная инженерия»
Дисциплина «Электротехника»

Лабораторная РАБОТА №3
Целочисленная арифметика и арифметика с плавающей запятой

Работу выполнил студенты группы ПИН-24 Баранов Д.А. и Демочкина А.В.
Работу проверил ассистент Института СПИНТех Фомин Р.А.

Москва 2022

Цель работы: изучить особенности целочисленной арифметики и арифметики с плавающей запятой.

Вариант 2

2	$m = 564, n = -322$

Задание 1.

Измените функции *print*16() и *print*32() заданий Л1.з3–Л1.з4 так, чтобы все данные, выводимые одним вызовом *print*16() либо *print*32(), занимали одну строку; а ширина каждого из представлений была бы постоянной: для шестнадцатеричного (а) и двоичного (б) представлений необходимо выводить ведущие нули (но не более, чем фактически присутствует: так, двоичное представление 16-битного числа должно содержать 16 бит); для десятичных (в), (е), (ж), (з) дополнять пробелами. Каждое из дублирующихся представлений — шестнадцатеричное (а) и (г), двоичное (б) и (д) — выводить в одном экземпляре.

```
#include <bitset>
#include <iomanip>
#include <iostream>
#include <typeinfo>
using namespace std;

void print16(void *p) {
    unsigned short *i = reinterpret_cast<unsigned short *>(p);
    cout << setfill('0');
    cout << setw(4) << hex << *i << '/';
    bitset<16> bitform(*i);
    cout << setw(16) << bitform << '/';
    cout << setfill(' ') << setw(5) << dec << *i << '/';
    short *c = reinterpret_cast<short *>(p);
    cout << setfill(' ') << showpos << setw(6) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
        << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
        << *b << endl
        << endl;
};

void print32(void *p) {
    unsigned int *i = reinterpret_cast<unsigned int *>(p);
    cout << setfill('0');
    cout << setw(8) << hex << *i << '/';
    bitset<32> bitform(*i);
    cout << setw(32) << bitform << '/';
    cout << setfill(' ') << setw(10) << dec << *i << '/';
    int *c = reinterpret_cast<int *>(p);
    cout << setfill(' ') << showpos << setw(11) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
        << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
        << *b << endl
        << endl;
};

int main() {
    int p;
    p = 4;
    print16(&p);
    long q;
    q = 6;
    print32(&q);

}
```

```
#include <bitset>
#include <iomanip>
#include <iostream>
#include <typeinfo>
using namespace std;

void print16(void *p) {
    unsigned short *i = reinterpret_cast<unsigned short *>(p);
    cout << setfill('0');
    cout << setw(4) << hex << *i << '/';
    bitset<16> bitform(*i);
    cout << setw(16) << bitform << '/';
    cout << setfill(' ') << setw(5) << dec << *i << '/';
    short *c = reinterpret_cast<short *>(p);
    cout << setfill(' ') << showpos << setw(6) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

void print32(void *p) {
    unsigned int *i = reinterpret_cast<unsigned int *>(p);
    cout << setfill('0');
    cout << setw(8) << hex << *i << '/';
    bitset<32> bitform(*i);
    cout << setw(32) << bitform << '/';
    cout << setfill(' ') << setw(10) << dec << *i << '/';
    int *c = reinterpret_cast<int *>(p);
    cout << setfill(' ') << showpos << setw(11) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

int main() {
    int p;
    p = 4;
    print16(&p);
    long q;
    q = 6;
    print32(&q);

}
```

Пример вывода:

```
0004/0000000000000100/     4/    +4/
+0.00000000000000000000000000000000000000000000560519/+5.605e-45

00000006/00000000000000000000000000000110/          6/          +6/
+0.00000000000000000000000000000000000000000000840779/+8.408e-45
```

Задание 2.

Разработайте программу на языке C++, которая расширяет значение целочисленной переменной из 16 бит до 32 бит, рассматривая числа как:

– знаковые (signed);

– беззнаковые (unsigned).

Проверьте её работу на значениях *m* и *n*. Каждое из двух значений — как *m*, так и *n* — должно расширяться двумя способами — как знаковым, так и беззнаковым.

```
int main() {
    unsigned short m1 = 564;
    unsigned int n1 = static_cast<unsigned int>(m1);
    print16(&m1);
    print32(&n1);
    cout << "___" << endl;

    signed short m2 = 564;
    signed int n2 = static_cast<signed int>(m2);
    print16(&m2);
    print32(&n2);
    cout << "___" << endl;

    unsigned short m3 = -322;
    unsigned int n3 = static_cast<unsigned int>(m3);
    print16(&m3);
    print32(&n3);
    cout << "___" << endl;

    signed short m4 = -322;
    signed int n4 = static_cast<signed int>(m4);
    print16(&m4);
    print32(&n4);
    cout << "___" << endl;
}
```

Использованы те же функции print16 и print32 из предыдущего задания.

```
#include <bitset>
#include <iomanip>
#include <iostream>
#include <typeinfo>
using namespace std;

void print16(void *p) {
    unsigned short *i = reinterpret_cast<unsigned short *>(p);
    cout << setfill('0');
    cout << setw(4) << hex << *i << '/';
    bitset<16> bitform(*i);
    cout << setw(16) << bitform << '/';
    cout << setfill(' ') << setw(5) << dec << *i << '/';
    short *c = reinterpret_cast<short *>(p);
```

```
    cout << setfill(' ') << showpos << setw(6) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

void print32(void *p) {
    unsigned int *i = reinterpret_cast<unsigned int *>(p);
    cout << setfill('0');
    cout << setw(8) << hex << *i << '/';
    bitset<32> bitform(*i);
    cout << setw(32) << bitform << '/';
    cout << setfill(' ') << setw(10) << dec << *i << '/';
    int *c = reinterpret_cast<int *>(p);
    cout << setfill(' ') << showpos << setw(11) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

int main() {
    unsigned short m1 = 564;
    unsigned int n1 = static_cast<unsigned int>(m1);
    print16(&m1);
    print32(&n1);
    cout << "___" << endl;

    signed short m2 = 564;
    signed int n2 = static_cast<signed int>(m2);
    print16(&m2);
    print32(&n2);
    cout << "___" << endl;

    unsigned short m3 = -322;
    unsigned int n3 = static_cast<unsigned int>(m3);
    print16(&m3);
    print32(&n3);
    cout << "___" << endl;

    signed short m4 = -322;
    signed int n4 = static_cast<signed int>(m4);
    print16(&m4);
    print32(&n4);
    cout << "___" << endl;
}
```

```
0234/0000001000110100/  564/  +564/
+0.00000000000000000010842750677769856695713008437609/+1.084e-19

00000234/00000000000000000000001000110100/       564/       +564/
+0.00000000000000000000000000000000000000079033233/+7.903e-43
```

```
___
0234/0000001000110100/  564/  +564/
+0.00000000000000000000000000000000000013224943712618/+1.322e-37

00000234/00000000000000000000001000110100/       564/       +564/
+0.00000000000000000000000000000000000000079033233/+7.903e-43


___
febe/1111111010111110/65214/  -322/
+0.00000000000000000000000000000000000013297418869193/+1.330e-37

0000febe/00000000000000001111111010111110/     65214/     +65214/
+0.00000000000000000000000000000000000009138427805/+9.138e-41


___
febe/1111111010111110/65214/
-322/-126938008129513277943375318749315858432.00000000000000000000000000000000000
0000000000000/-1.269e+38

fffffebe/11111111111111111111111010111110/4294966974/       -322/
-nan/      -nan

___
```

Задание 3.

Разработайте программу на языке C/C++, которая выполняет над 16-битной целочисленной переменной:
– знаковое умножение на 2;
– беззнаковое умножение на 2;
– знаковое деление на 2;
– беззнаковое деление на 2;
– расчёт остатка от беззнакового деления на 16;
– округление вниз до числа, кратного 16 (беззнаковое).

```
int main() {
    signed short m1 = 564;
    unsigned short m2 = 564;
    signed short n1 = -322;
    unsigned short n2 = -322;

    signed int count1 = 2;
    signed int pr1 = m1 * count1;
    cout << endl
         << "Знаковое умножение m и n на 2, значение и результат для числа m и "
            "числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m1);
    cout << "result:" << endl;
    print16(&pr1);
    pr1 = n1 * count1;
    cout << "n:" << endl;
    print16(&n1);
    cout << "result:" << endl;
    print16(&pr1);

    unsigned int count2 = 2;
    unsigned int pr2 = m2 * count2;
    cout << endl
         << "Беззнаковое умножение m и n на 2, значение и результат для числа "
            "m и числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&pr2);
    pr2 = n2 * count2;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&pr2);
```

```
  signed int del1 = m1 / count1;
  cout << endl
       << "Знаковое деление m и n на 2, значение и результат для числа m и "
          "числа n:"
       << endl;
  cout << "m:" << endl;
  print16(&m1);
  cout << "result:" << endl;
  print16(&del1);
  del1 = n1 / count1;
  cout << "n:" << endl;
  print16(&n1);
  cout << "result:" << endl;
  print16(&del1);

  unsigned int del2 = m2 / count2;
  cout << endl
       << "Беззнаковое деление m и n на 2, значение и результат для числа m "
          "и числа n:"
       << endl;
  cout << "m:" << endl;
  print16(&m2);
  cout << "result:" << endl;
  print16(&del2);
  del2 = n2 / count2;
  cout << "n:" << endl;
  print16(&n2);
  cout << "result:" << endl;
  print16(&del2);
```

```
    count2 = 16;
    unsigned int ost = m2 % count2;
    cout << endl
         << "Беззнаковое деление m и n на 16, значение и результат для числа m "
            "и числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&ost);
    ost = n2 % count2;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&ost);

    unsigned int round = m2 - (m2 % count2);
    cout << endl << "Округление вниз до числа, кратного 16" << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&round);
    ost = n2 - (n2 % count2);
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&round);
```

```
#include <bitset>
#include <iomanip>
#include <iostream>
#include <typeinfo>

using namespace std;

void print16(void *p) {
    unsigned short *i = reinterpret_cast<unsigned short *>(p);
    cout << setfill('0');
    cout << setw(4) << hex << *i << '/';
    bitset<16> bitform(*i);
    cout << setw(16) << bitform << '/';
    cout << setfill(' ') << setw(5) << dec << *i << '/';
    short *c = reinterpret_cast<short *>(p);
    cout << setfill(' ') << showpos << setw(6) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

void print32(void *p) {
    unsigned int *i = reinterpret_cast<unsigned int *>(p);
    cout << setfill('0');
    cout << setw(8) << hex << *i << '/';
    bitset<32> bitform(*i);
    cout << setw(32) << bitform << '/';
    cout << setfill(' ') << setw(10) << dec << *i << '/';
    int *c = reinterpret_cast<int *>(p);
    cout << setfill(' ') << showpos << setw(11) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
```

```
        << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
        << *b << endl
        << endl;
};

int main() {
    signed short m1 = 564;
    unsigned short m2 = 564;
    signed short n1 = -322;
    unsigned short n2 = -322;

    signed int count1 = 2;
    signed int pr1 = m1 * count1;
    cout << endl
        << "Знаковое умножение m и n на 2, значение и результат для числа m и "
           "числа n:"
        << endl;
    cout << "m:" << endl;
    print16(&m1);
    cout << "result:" << endl;
    print16(&pr1);
    pr1 = n1 * count1;
    cout << "n:" << endl;
    print16(&n1);
    cout << "result:" << endl;
    print16(&pr1);

    unsigned int count2 = 2;
    unsigned int pr2 = m2 * count2;
    cout << endl
        << "Беззнаковое умножение m и n на 2, значение и результат для числа "
           "m и числа n:"
        << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&pr2);
    pr2 = n2 * count2;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&pr2);

    signed int del1 = m1 / count1;
    cout << endl
        << "Знаковое деление m и n на 2, значение и результат для числа m и "
           "числа n:"
        << endl;
    cout << "m:" << endl;
    print16(&m1);
    cout << "result:" << endl;
    print16(&del1);
    del1 = n1 / count1;
    cout << "n:" << endl;
    print16(&n1);
    cout << "result:" << endl;
    print16(&del1);
```

```
    unsigned int del2 = m2 / count2;
    cout << endl
         << "Беззнаковое деление m и n на 2, значение и результат для числа m "
            "и числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&del2);
    del2 = n2 / count2;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&del2);

    count2 = 16;
    unsigned int ost = m2 % count2;
    cout << endl
         << "Беззнаковое деление m и n на 16, значение и результат для числа m "
            "и числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&ost);
    ost = n2 % count2;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&ost);

    unsigned int round = m2 - (m2 % count2);
    cout << endl << "Округление вниз до числа, кратного 16" << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&round);
    ost = n2 - (n2 % count2);
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&round);
}
```

```
Знаковое умножение m и n на 2, значение и результат для числа m и числа n:
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000001561344434352222366492457528153182/+1.561e-17

result:
0468/0000010001101000/ 1128/ +1128/
+0.00000000000000000000000000000000000000158066467/+1.581e-42

n:
febe/1111111010111110/65214/  -322/
+0.00000000000000000000000000000000013297418869193/+1.330e-37
```

```
result:
fd7c/1111110101111100/64892/  -644/
-nan/      -nan


Беззнаковое умножение m и n на 2, значение и результат для числа m и числа n:
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000000000000000000000000013224943712618/+1.322e-37

result:
0468/0000010001101000/ 1128/ +1128/
+0.00000000000000000000000000000000000000000158066467/+1.581e-42

n:
febe/1111111010111110/65214/
-322/-126938008129513277943375318749315858432.00000000000000000000000000000000000000
0000000000000/-1.269e+38

result:
fd7c/1111110101111100/64892/  -644/
+0.00000000000000000000000000000000000000018276855610/+1.828e-40


Знаковое деление m и n на 2, значение и результат для числа m и числа n:
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000015613444343522223664924575281531 82/+1.561e-17

result:
011a/0000000100011010/  282/  +282/
+0.00000000000000000000000000000000000000000039516617/+3.952e-43

n:
febe/1111111010111110/65214/  -322/
+0.00000000000000000000000000000000000013297418869193/+1.330e-37

result:
ff5f/1111111101011111/65375/  -161/
-nan/      -nan


Беззнаковое деление m и n на 2, значение и результат для числа m и числа n:
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000000000000000000000000013224943712618/+1.322e-37

result:
011a/0000000100011010/  282/  +282/
+0.00000000000000000000000000000000000000000039516617/+3.952e-43

n:
febe/1111111010111110/65214/
-322/-126938008129513277943375318749315858432.00000000000000000000000000000000000000
0000000000000/-1.269e+38

result:
7f5f/0111111101011111/32607/+32607/
+0.00000000000000000000000000000000000000004569213903/+4.569e-41
```

```
Остаток от деления m и n на 16, значение и результат для числа m и числа n:
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000000000000000000000000013224943712618/+1.322e-37

result:
0004/0000000000000100/    4/    +4/
+0.00000000000000000000000000000000000000000000560519/+5.605e-45

n:
febe/1111111010111110/65214/
-322/-126938008129513277943375318749315858432.00000000000000000000000000000000000000
0000000000000/-1.269e+38

result:
000e/0000000000001110/   14/   +14/
+0.00000000000000000000000000000000000000000001961818/+1.962e-44


Округление вниз до числа, кратного 16
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000000000000000000000000013224943712618/+1.322e-37

result:
0230/0000001000110000/  560/  +560/
+0.00000000000000000000000000000000000000000078472714/+7.847e-43

n:
febe/1111111010111110/65214/
-322/-126938008129513277943375318749315858432.00000000000000000000000000000000000000
0000000000000/-1.269e+38

result:
0230/0000001000110000/  560/  +560/
+0.00000000000000000000000000000000000000000078472714/+7.847e-43
```

Задание 4.

Разработайте программу на языке C/C++, которая выполняет над 16-битной целочисленной переменной *x*:

– знаковый сдвиг влево на 1 бит;

– беззнаковый сдвиг влево на 1 бит;

– знаковый сдвиг вправо на 1 бит;

– беззнаковый сдвиг вправо на 1 бит;

– рассчитывает *x* & 15;

– рассчитывает *x* & −16.

```
int main() {
    signed short m1 = 564;
    unsigned short m2 = 564;
    signed short n1 = -322;
    unsigned short n2 = -322;


    signed int sd1 = m1 << 1;
    cout << endl
         << "Знаковый сдвиг влево на 1 бит, значение и результат для числа m и "
            "числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m1);
    cout << "result:" << endl;
    print16(&sd1);
    sd1 = n1 << 1;
    cout << "n:" << endl;
    print16(&n1);
    cout << "result:" << endl;
    print16(&sd1);

    unsigned int sd2 = m2 << 1;
    cout << endl
         << "Безнаковый сдвиг влево на 1 бит, значение и результат для числа m и "
            "числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&sd2);
    sd2 = n2 << 1;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&sd2);
```

```
  signed int sd_right1 = m1 >> 1;
  cout << endl
       << "Знаковый сдвиг вправо на 1 бит, значение и результат для числа m и "
          "числа n"
       << endl;
  cout << "m:" << endl;
  print16(&m1);
  cout << "result:" << endl;
  print16(&sd_right1);
  sd_right1 = n1 >> 1;
  cout << "n:" << endl;
  print16(&n1);
  cout << "result:" << endl;
  print16(&sd_right1);

  unsigned int sd_right2 = m2 >> 1;
  cout << endl
       << "Безнаковый сдвиг вправо на 1 бит, значение и результат для числа m и "
          "числа n::"
       << endl;
  cout << "m:" << endl;
  print16(&m2);
  cout << "result:" << endl;
  print16(&sd_right2);
  sd_right2 = n2 >> 1;
  cout << "n:" << endl;
  print16(&n2);
  cout << "result:" << endl;
  print16(&sd_right2);
```

```
    unsigned int ost = m2 & 15;
    cout << endl
         << "Рассчёт x & 15 для m и n, значение и результат для числа m "
            "и числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&ost);
    ost = n2 & 15;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&ost);

    unsigned int round = m2 & -16;
    cout << endl << "Рассчёт x & -16 для m и n, значение и результат для числа m "
            "и числа n:" << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&round);
    ost = n2 & -16;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&round);
}
```

```
#include <bitset>
#include <iomanip>
#include <iostream>
#include <typeinfo>

using namespace std;

void print16(void *p) {
    unsigned short *i = reinterpret_cast<unsigned short *>(p);
    cout << setfill('0');
    cout << setw(4) << hex << *i << '/';
    bitset<16> bitform(*i);
    cout << setw(16) << bitform << '/';
    cout << setfill(' ') << setw(5) << dec << *i << '/';
    short *c = reinterpret_cast<short *>(p);
    cout << setfill(' ') << showpos << setw(6) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

void print32(void *p) {
    unsigned int *i = reinterpret_cast<unsigned int *>(p);
    cout << setfill('0');
    cout << setw(8) << hex << *i << '/';
    bitset<32> bitform(*i);
    cout << setw(32) << bitform << '/';
    cout << setfill(' ') << setw(10) << dec << *i << '/';
    int *c = reinterpret_cast<int *>(p);
    cout << setfill(' ') << showpos << setw(11) << *c << '/';
```

```
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

int main() {
    signed short m1 = 564;
    unsigned short m2 = 564;
    signed short n1 = -322;
    unsigned short n2 = -322;


    signed int sd1 = m1 << 1;
    cout << endl
         << "Знаковый сдвиг влево на 1 бит, значение и результат для числа m и "
            "числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m1);
    cout << "result:" << endl;
    print16(&sd1);
    sd1 = n1 << 1;
    cout << "n:" << endl;
    print16(&n1);
    cout << "result:" << endl;
    print16(&sd1);

    unsigned int sd2 = m2 << 1;
    cout << endl
         << "Беззнаковый сдвиг влево на 1 бит, значение и результат для числа m и "
            "числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&sd2);
    sd2 = n2 << 1;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&sd2);

    signed int sd_right1 = m1 >> 1;
    cout << endl
         << "Знаковый сдвиг вправо на 1 бит, значение и результат для числа m и "
            "числа n"
         << endl;
    cout << "m:" << endl;
    print16(&m1);
    cout << "result:" << endl;
    print16(&sd_right1);
    sd_right1 = n1 >> 1;
    cout << "n:" << endl;
    print16(&n1);
    cout << "result:" << endl;
```

```
    print16(&sd_right1);

    unsigned int sd_right2 = m2 >> 1;
    cout << endl
         << "Беззнаковый сдвиг вправо на 1 бит, значение и результат для числа m и "
            "числа n::"
         << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&sd_right2);
    sd_right2 = n2 >> 1;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&sd_right2);


    unsigned int ost = m2 & 15;
    cout << endl
         << "Рассчёт x & 15 для m и n, значение и результат для числа m "
            "и числа n:"
         << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&ost);
    ost = n2 & 15;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&ost);

    unsigned int round = m2 & -16;
    cout << endl << "Рассчёт x & -16 для m и n, значение и результат для числа m "
            "и числа n:" << endl;
    cout << "m:" << endl;
    print16(&m2);
    cout << "result:" << endl;
    print16(&round);
    ost = n2 & -16;
    cout << "n:" << endl;
    print16(&n2);
    cout << "result:" << endl;
    print16(&round);
}
```

```
Знаковый сдвиг влево на 1 бит, значение и результат для числа m и числа n:
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000001387872086754541657051265080013991/+1.388e-17

result:
0468/0000010001101000/ 1128/ +1128/
+0.00000000000000000000000000000000000000158066467/+1.581e-42

n:
```

```
febe/1111111010111110/65214/  -322/
+0.00000000000000000000000000000000000013297418869193/+1.330e-37

result:
fd7c/1111110101111100/64892/  -644/
-nan/      -nan


Беззнаковый сдвиг влево на 1 бит, значение и результат для числа m и числа n:
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000000000000000000000000013224943712618/+1.322e-37

result:
0468/0000010001101000/ 1128/ +1128/
+0.00000000000000000000000000000000000000158066467/+1.581e-42

n:
febe/1111111010111110/65214/
-322/-126938008129513277943375318749315858432.00000000000000000000000000000000000000000000000000/-1.269e+38

result:
fd7c/1111110101111100/64892/  -644/
+0.00000000000000000000000000000000000018276855610/+1.828e-40


Знаковый сдвиг вправо на 1 бит, значение и результат для числа m и числа n
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000001387872086754541657051265080013991/+1.388e-17

result:
011a/0000000100011010/  282/  +282/
+0.00000000000000000000000000000000000000039516617/+3.952e-43

n:
febe/1111111010111110/65214/  -322/
+0.00000000000000000000000000000000000013297418869193/+1.330e-37

result:
ff5f/1111111101011111/65375/  -161/
-nan/      -nan


Беззнаковый сдвиг вправо на 1 бит, значение и результат для числа m и числа n::
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000000000000000000000000013224943712618/+1.322e-37

result:
011a/0000000100011010/  282/  +282/
+0.00000000000000000000000000000000000039516617/+3.952e-43

n:
febe/1111111010111110/65214/
-322/-126938008129513277943375318749315858432.00000000000000000000000000000000000000000000000000/-1.269e+38
```

```
result:
7f5f/0111111101011111/32607/+32607/
+0.00000000000000000000000000000000000004569213903/+4.569e-41


Рассчёт x & 15 для m и n, значение и результат для числа m и числа n:
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000000000000000000000000013224943712618/+1.322e-37

result:
0004/0000000000000100/    4/    +4/
+0.00000000000000000000000000000000000000000000560519/+5.605e-45

n:
febe/1111111010111110/65214/
-322/-126938008129513277943375318749315858432.00000000000000000000000000000000000000
0000000000000/-1.269e+38

result:
000e/0000000000001110/   14/   +14/
+0.00000000000000000000000000000000000000000001961818/+1.962e-44


Рассчёт x & -16 для m и n, значение и результат для числа m и числа n:
m:
0234/0000001000110100/  564/  +564/
+0.00000000000000000000000000000000000013224943712618/+1.322e-37

result:
0230/0000001000110000/  560/  +560/
+0.00000000000000000000000000000000000000000078472714/+7.847e-43

n:
febe/1111111010111110/65214/
-322/-126938008129513277943375318749315858432.00000000000000000000000000000000000000
0000000000000/-1.269e+38

result:
0230/0000001000110000/  560/  +560/
+0.00000000000000000000000000000000000000000078472714/+7.847e-43
```

Результаты Задания 4 совпадают с результатами Задания 3. Значит данные операции их обеих заданий эквиваленты.

Задание 5.

Разработайте программу на языке C/C++, которая, используя только сложение, вычитание и побитовые операции, округляет целочисленное беззнаковое значение x до кратного значению D (таблица Л3.2) двумя способами:

а) вниз;

б) вверх.

Варинат 2

2	$D = 32$

```
int main() {
    unsigned int d = 32;
    unsigned int x = 88;

    cout << "Округление безрнакого целого числа вниз: " << x << endl;
    d = d - 1;  // вычитаем 1, чтобы применить конъюнкцию
    unsigned int remains =
        (x & d);  // получаем остаток от деления при помощи конъюнкции
    cout << "Результат: " << x - remains << endl;
    unsigned int q1  = x + remains + 1;
    print16(&q1);
    d = d + 1; //возвращение изначального значения d
    // Находим остаток от деления на d-1. Вычитаем из числа.

    cout << "Округление безрнакого целого числа вверх: " << x << endl;
    x = x - 1;
    d = d - 1;
    remains = ~(x & d);
    remains = (remains & d);
    cout << "Результат: " << x + remains + 1 << endl;
    unsigned int q2  = x + remains + 1;
    print16(&q2);
    // Наглядный пример. У нас есть два случая: 1) число является степенью
    // двойки, 2) число не является степенью двaойки.
    // Необходимо обработать два этих случае вместе одинаково. Пример для 1
    // случая: возьмём 100000. Вычтем 1. Получим 11111. Найдём остаток от
    // деления на 32. И возьмём отрицание каждого разряда операнда. Получим
    // 00000. Прибавим это значение значению, полученному в результате вычитания
    // единички. И получим искомое значение 100000. То есть получили 32. Для
    // 100100. Вычитаем 1 => 100011. Остаток = 00011. Отрицание остатка: 11100.
    // Прибавляем к 100011 остаток 11100 и 1. Получаем 1000000 = 64.
}
```

```
#include <bitset>
#include <iomanip>
#include <iostream>
#include <typeinfo>

using namespace std;

void print16(void *p) {
    unsigned short *i = reinterpret_cast<unsigned short *>(p);
    cout << setfill('0');
    cout << setw(4) << hex << *i << '/';
    bitset<16> bitform(*i);
    cout << setw(16) << bitform << '/';
    cout << setfill(' ') << setw(5) << dec << *i << '/';
    short *c = reinterpret_cast<short *>(p);
    cout << setfill(' ') << showpos << setw(6) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

void print32(void *p) {
    unsigned int *i = reinterpret_cast<unsigned int *>(p);
    cout << setfill('0');
    cout << setw(8) << hex << *i << '/';
    bitset<32> bitform(*i);
    cout << setw(32) << bitform << '/';
```

```
    cout << setfill(' ') << setw(10) << dec << *i << '/';
    int *c = reinterpret_cast<int *>(p);
    cout << setfill(' ') << showpos << setw(11) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

int main() {
    unsigned int d = 32;
    unsigned int x = 88;

    cout << "Округление безрнакого целого числа вниз: " << x << endl;
    d = d - 1;  // вычитаем 1, чтобы применить конъюнкцию
    unsigned int remains =
        (x & d);  // получаем остаток от деления при помощи конъюнкции
    cout << "Результат: " << x - remains << endl;
    unsigned int q1  = x + remains + 1;
    print16(&q1);
    d = d + 1; //возвращение изначального значения d
    // Находим остаток от деления на d-1. Вычитаем из числа.

    cout << "Округление безрнакого целого числа вверх: " << x << endl;
    x = x - 1;
    d = d - 1;
    remains = ~(x & d);
    remains = (remains & d);
    cout << "Результат: " << x + remains + 1 << endl;
    unsigned int q2  = x + remains + 1;
    print16(&q2);
    // Наглядный пример. У нас есть два случая: 1) число является степенью
    // двойки, 2) число не является степенью двайки.
    // Необходимо обработать два этих случае вместе одинаково. Пример для 1
    // случая: возьмём 100000. Вычтем 1. Получим 11111. Найдём остаток от
    // деления на 32. И возьмём отрицание каждого разряда операнда. Получим
    // 00000. Прибавим это значение значению, полученному в результате вычитания
    // единички. И получим искомое значение 100000. То есть получили 32. Для
    // 100100. Вычитаем 1 => 100011. Остаток = 00011. Отрицание остатка: 11100.
    // Прибавляем к 100011 остаток 11100 и 1. Получаем 1000000 = 64.
}
```

```
кругление безрнакого целого числа вниз: 88
Результат: 64
0071/0000000001110001/  113/  +113/
+0.00000000000000000000000000000000000000000015834673/+1.583e-43

Округление безрнакого целого числа вверх: 88
Результат: 96
0060/0000000001100000/   96/   +96/
+0.00000000000000000000000000000000000000000013452465/+1.345e-43
```

Задание 6.

Разработайте программу на языке C/C++, которая выполняет для 32-битной переменной целочисленный инкремент (то есть целочисленная интерпретация соответствующего 32-битного участка памяти должна увеличиться на 1) и целочисленный декремент (аналогично, целочисленная интерпретация должна уменьшиться на 1)

Вариант 2

2	$a = 0, b = 1, c = 12233445, d = 122334455$

```
int main() {
    unsigned int m = 564;
    m++;
    cout << "Инкремент целочисленной беззнаковой переменной m: " << endl;
    print32(&m);

    unsigned int n = -322;
    n++;
    cout << "Инкремент целочисленной беззнаковой переменной n: " << endl;
    print32(&n);

    signed int m2 = 564;
    m2++;
    cout << "Инкремент целочисленной знаковой переменной m: " << endl;
    print32(&m2);

    signed int n2 = -322;
    n2++;
    cout << "Инкремент целочисленной знаковой переменной n: " << endl;
    print32(&n2);

    float a = 0;
    a++;
    cout << "Инкремент переменной a с плавающей запятой: " << endl;
    print32(&a);

    float b = 1;
    b++;
    cout << "Инкремент переменной b с плавающей запятой: " << endl;
    print32(&b);

    float c = 12233445;
    c++;
    cout << "Инкремент переменной c с плавающей запятой: " << endl;
    print32(&c);
```

```
float d = 122334455;
d++;
cout << "Инкремент переменной d с плавающей запятой: " << endl;
print32(&d);

int count1 = 0;
count1++;
cout << "Инкремент нуля: " << endl;
print32(&count1);

unsigned int count2 = 4294967295;
count2++;
cout << "Инкремент максимального целого 32-битного значения без знака: " << endl;
print32(&count2);

int count3 = -2147483648;
count3++;
cout << "Инкремент минимального целого 32-битного значения со знаком: " << endl;
print32(&count3);

unsigned int count4 = 2147483647;
count4++;
cout << "Инкремент максимального целого 32-битного значения со знаком: " << endl;
print32(&count4);
```

```
    unsigned int m = 564;
    m--;
    cout << "Декремент целочисленной беззнаковой переменной m: " << endl;
    print32(&m);

    unsigned int n = -322;
    n--;
    cout << "Декремент целочисленной беззнаковой переменной n: " << endl;
    print32(&n);

    signed int m2 = 564;
    m2--;
    cout << "Декремент целочисленной знаковой переменной m: " << endl;
    print32(&m2);

    signed int n2 = -322;
    n2--;
    cout << "Декремент целочисленной знаковой переменной n: " << endl;
    print32(&n2);

    float a = 0;
    a--;
    cout << "Декремент переменной a с плавающей запятой: " << endl;
    print32(&a);

    float b = 1;
    b--;
    cout << "Декремент переменной b с плавающей запятой: " << endl;
    print32(&b);

    float c = 12233445;
    c--;
    cout << "Декремент переменной c с плавающей запятой: " << endl;
    print32(&c);
```

```
    float d = 122334455;
    d--;
    cout << "Декремент переменной d с плавающей запятой: " << endl;
    print32(&d);

    int count1 = 0;
    count1--;
    cout << "Декремент нуля: " << endl;
    print32(&count1);

    unsigned int count2 = 4294967295;
    count2--;
    cout << "Декремент максимального целого 32-битного значения без знака: " << endl;
    print32(&count2);

    int count3 = -2147483648;
    count3--;
    cout << "Декремент минимального целого 32-битного значения со знаком: " << endl;
    print32(&count3);

    unsigned int count4 = 2147483647;
    count4--;
    cout << "Декремент максимального целого 32-битного значения со знаком: " << endl;
    print32(&count4);
```

```
#include <bitset>
#include <iomanip>
#include <iostream>
#include <typeinfo>

using namespace std;

void print16(void *p) {
    unsigned short *i = reinterpret_cast<unsigned short *>(p);
    cout << setfill('0');
    cout << setw(4) << hex << *i << '/';
```

```
    bitset<16> bitform(*i);
    cout << setw(16) << bitform << '/';
    cout << setfill(' ') << setw(5) << dec << *i << '/';
    short *c = reinterpret_cast<short *>(p);
    cout << setfill(' ') << showpos << setw(6) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

void print32(void *p) {
    unsigned int *i = reinterpret_cast<unsigned int *>(p);
    cout << setfill('0');
    cout << setw(8) << hex << *i << '/';
    bitset<32> bitform(*i);
    cout << setw(32) << bitform << '/';
    cout << setfill(' ') << setw(10) << dec << *i << '/';
    int *c = reinterpret_cast<int *>(p);
    cout << setfill(' ') << showpos << setw(11) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

int main() {
    unsigned int m = 564;
    m++;
    cout << "Инкремент целочисленной беззнаковой переменной m: " << endl;
    print32(&m);

    unsigned int n = -322;
    n++;
    cout << "Инкремент целочисленной беззнаковой переменной n: " << endl;
    print32(&n);

    signed int m2 = 564;
    m2++;
    cout << "Инкремент целочисленной знаковой переменной m: " << endl;
    print32(&m2);

    signed int n2 = -322;
    n2++;
    cout << "Инкремент целочисленной знаковой переменной n: " << endl;
    print32(&n2);

    float a = 0;
    a++;
    cout << "Инкремент переменной a с плавающей запятой: " << endl;
    print32(&a);

    float b = 1;
    b++;
```

```
    cout << "Инкремент переменной b с плавающей запятой: " << endl;
    print32(&b);

    float c = 12233445;
    c++;
    cout << "Инкремент переменной c с плавающей запятой: " << endl;
    print32(&c);

    float d = 122334455;
    d++;
    cout << "Инкремент переменной d с плавающей запятой: " << endl;
    print32(&d);

    int count1 = 0;
    count1++;
    cout << "Инкремент нуля: " << endl;
    print32(&count1);

    unsigned int count2 = 4294967295;
    count2++;
    cout << "Инкремент максимального целого 32-битного значения без знака: " << endl;
    print32(&count2);

    int count3 = -2147483648;
    count3++;
    cout << "Инкремент минимального целого 32-битного значения со знаком: " << endl;
    print32(&count3);

    unsigned int count4 = 2147483647;
    count4++;
    cout << "Инкремент максимального целого 32-битного значения со знаком: " << endl;
    print32(&count4);

}

#include <bitset>
#include <iomanip>
#include <iostream>
#include <typeinfo>

using namespace std;

void print16(void *p) {
    unsigned short *i = reinterpret_cast<unsigned short *>(p);
    cout << setfill('0');
    cout << setw(4) << hex << *i << '/';
    bitset<16> bitform(*i);
    cout << setw(16) << bitform << '/';
    cout << setfill(' ') << setw(5) << dec << *i << '/';
    short *c = reinterpret_cast<short *>(p);
    cout << setfill(' ') << showpos << setw(6) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};
```

```
void print32(void *p) {
    unsigned int *i = reinterpret_cast<unsigned int *>(p);
    cout << setfill('0');
    cout << setw(8) << hex << *i << '/';
    bitset<32> bitform(*i);
    cout << setw(32) << bitform << '/';
    cout << setfill(' ') << setw(10) << dec << *i << '/';
    int *c = reinterpret_cast<int *>(p);
    cout << setfill(' ') << showpos << setw(11) << *c << '/';
    float *b = reinterpret_cast<float *>(p);
    cout << setfill(' ') << setw(55) << dec << setprecision(50) << fixed << *b
         << '/';
    cout << setfill(' ') << setw(10) << dec << setprecision(3) << scientific
         << *b << endl
         << endl;
};

int main() {
    unsigned int m = 564;
    m--;
    cout << "Декремент целочисленной беззнаковой переменной m: " << endl;
    print32(&m);

    unsigned int n = -322;
    n--;
    cout << "Декремент целочисленной беззнаковой переменной n: " << endl;
    print32(&n);

    signed int m2 = 564;
    m2--;
    cout << "Декремент целочисленной знаковой переменной m: " << endl;
    print32(&m2);

    signed int n2 = -322;
    n2--;
    cout << "Декремент целочисленной знаковой переменной n: " << endl;
    print32(&n2);

    float a = 0;
    a--;
    cout << "Декремент переменной a с плавающей запятой: " << endl;
    print32(&a);

    float b = 1;
    b--;
    cout << "Декремент переменной b с плавающей запятой: " << endl;
    print32(&b);

    float c = 12233445;
    c--;
    cout << "Декремент переменной c с плавающей запятой: " << endl;
    print32(&c);

    float d = 122334455;
    d--;
    cout << "Декремент переменной d с плавающей запятой: " << endl;
    print32(&d);
```

```
    int count1 = 0;
    count1--;
    cout << "Декремент нуля: " << endl;
    print32(&count1);

    unsigned int count2 = 4294967295;
    count2--;
    cout << "Декремент максимального целого 32-битного значения без знака: " << endl;
    print32(&count2);

    int count3 = -2147483648;
    count3--;
    cout << "Декремент минимального целого 32-битного значения со знаком: " << endl;
    print32(&count3);

    unsigned int count4 = 2147483647;
    count4--;
    cout << "Декремент максимального целого 32-битного значения со знаком: " << endl;
    print32(&count4);
}
```

```
Инкремент целочисленной беззнаковой переменной m:
00000235/00000000000000000000001000110101/       565/       +565/
+0.00000000000000000000000000000000000000000079173363/+7.917e-43

Инкремент целочисленной беззнаковой переменной n:
fffffebf/11111111111111111111111010111111/4294966975/       -321/
-nan/      -nan

Инкремент целочисленной знаковой переменной m:
00000235/00000000000000000000001000110101/       565/       +565/
+0.00000000000000000000000000000000000000000079173363/+7.917e-43

Инкремент целочисленной знаковой переменной n:
fffffebf/11111111111111111111111010111111/4294966975/       -321/
-nan/      -nan

Инкремент переменной a с плавающей запятой:
3f800000/00111111100000000000000000000000/1065353216/+1065353216/
+1.00000000000000000000000000000000000000000000000000/+1.000e+00

Инкремент переменной b с плавающей запятой:
40000000/01000000000000000000000000000000/1073741824/+1073741824/
+2.00000000000000000000000000000000000000000000000000/+2.000e+00

Инкремент переменной c с плавающей запятой:
4b3aaae6/01001011001110101010101011100110/1262136038/+1262136038/+12233446.00000000
00000000000000000000000000000000000000000000/+1.223e+07

Инкремент переменной d с плавающей запятой:
4ce9559f/01001100111010010101010110011111/1290360223/+1290360223/+122334456.0000000
00000000000000000000000000000000000000000000/+1.223e+08

Инкремент нуля:
00000001/00000000000000000000000000000001/         1/         +1/
+0.00000000000000000000000000000000000000000000140130/+1.401e-45
```

```
Инкремент максимального целого 32-битного значения без знака:
00000000/00000000000000000000000000000000/         0/         +0/
+0.00000000000000000000000000000000000000000000000000000/+0.000e+00

Инкремент минимального целого 32-битного значения со знаком:
80000001/10000000000000000000000000000001/2147483649/-2147483647/
-0.00000000000000000000000000000000000000000000140130/-1.401e-45

Инкремент максимального целого 32-битного значения со знаком:
80000000/10000000000000000000000000000000/2147483648/-2147483648/
-0.00000000000000000000000000000000000000000000000000/-0.000e+00




Декремент целочисленной беззнаковой переменной m:
00000233/00000000000000000000001000110011/       563/       +563/
+0.00000000000000000000000000000000000000000078893104/+7.889e-43

Декремент целочисленной беззнаковой переменной n:
fffffebd/11111111111111111111111010111101/4294966973/       -323/
-nan/      -nan

Декремент целочисленной знаковой переменной m:
00000233/00000000000000000000001000110011/       563/       +563/
+0.00000000000000000000000000000000000000000078893104/+7.889e-43

Декремент целочисленной знаковой переменной n:
fffffebd/11111111111111111111111010111101/4294966973/       -323/
-nan/      -nan

Декремент переменной a с плавающей запятой:
bf800000/10111111100000000000000000000000/3212836864/-1082130432/
-1.00000000000000000000000000000000000000000000000000000/-1.000e+00

Декремент переменной b с плавающей запятой:
00000000/00000000000000000000000000000000/         0/         +0/
+0.00000000000000000000000000000000000000000000000000000/+0.000e+00

Декремент переменной c с плавающей запятой:
4b3aaae4/01001011001110101010101011100100/1262136036/+1262136036/+12233444.00000000
00000000000000000000000000000000000000000000/+1.223e+07

Декремент переменной d с плавающей запятой:
4ce9559f/01001100111010010101010110011111/1290360223/+1290360223/+122334456.0000000
00000000000000000000000000000000000000000000/+1.223e+08

Декремент нуля:
ffffffff/11111111111111111111111111111111/4294967295/         -1/
-nan/      -nan

Декремент максимального целого 32-битного значения без знака:
fffffffe/11111111111111111111111111111110/4294967294/         -2/
-nan/      -nan

Декремент минимального целого 32-битного значения со знаком:
7fffffff/01111111111111111111111111111111/2147483647/+2147483647/
+nan/      +nan
```

Декремент максимального целого 32-битного значения со знаком:
7ffffffe/01111111111111111111111111111110/2147483646/+2147483646/

Вопросы

1. Что такое расширение чисел со знаком и без знака? Для чего нужны операции расширения?

Расширение числа - операция увеличения разрядности.

– беззнаковое расширение — расширяемая часть заполняется нулями (такая операция сохраняет значение беззнаковой интерпретации *x*);

– знаковое расширение — расширяемая часть заполняется значением знакового бита (сохраняет значение знаковой интерпретации *x*).

Для неотрицательных (в знаковой интерпретации) чисел знаковое и беззнаковое расширение выполняется одинаково.

Операции расширения позволяют работать с числами разных типов.

2. Как выполняются логические операции, побитовые операции и сдвиги над строкой битов?

Логические:

Отрицание: к каждому разряду единственного операнда применяется логическое отрицание

Конъюнкция: каждой паре разрядов операндов применяется логическое “и”

Дизъюнкция: каждой паре разрядов операндов применяется логическое «или»

Сложение по модулю два: каждой паре разрядов операндов применяется исключающее «или»

3. Как представляются в памяти компьютера числа с плавающей запятой?

Представление с фиксированной запятой: *N* битов, отведённых под представление числа, для дробной части используется *n*, для целой остаётся *N* − n

Двоичная запятая, отделяющая дробную часть от целой, всегда расположена между разрядами *n* и *n* − 1.

Целая часть числа может принимать значения от 0 до $2^{N-n}-1$, дробная — от $0=\overline{0{,}00\ldots00}$ до $1-\frac{1}{2^n}=\overline{0{,}11\ldots11}$ (всего 2^n значений). Таким образом, числа, представимые в формате с фиксированной запятой с помощью N бит, n из которых отведены под дробную часть, заключены в диапазоне $\left[0,\ 2^{N-n}-\frac{1}{2^n}\right]$.

Представление с плавающей запятой: в виде экспоненциальной формы записи, где кроме порядка и мантиссы определён и знак числа.

$$X=(-1)^s\cdot N^p\cdot \mu$$