СибГУТИ

Кафедра вычислительных систем

Дисциплина
"ПРОГРАММИРОВАНИЕ"

Математика и вычислительная техника

Преподаватель:

Ст. преп. Кафедры ВС

Перышкова Евгения Николаевна

План лекции

1. Особенности арифметики в вычислительной технике.
2. Циклическая обработка данных.

Работа с числами

Вычислительная техника оперирует числами, выполняя над ними простейшие преобразования по правилам арифметики, тригонометрии и алгебры. Однако арифметика в ВТ имеет ряд особенностей, которые отличают ее от классической арифметики.

1. Ограниченная разрядность.
2. Не предусмотрены знаки: "–" и ".".
3. Двоичная система счисления.
4. Наличие безусловного и условного переходов.
5. Итеративная обработка данных.

Переменная

Математика

В математике переменная — это величина, характеризующаяся множеством значений, которое она может принимать.

- Короткие имена (*a*, *x*, *β*, …)

- Является частью выражения.
- Может принимать произвольно большие или малые значения.
- Диапазон значений, обычно, множество вещественных чисел.

Вычислительная техника

Переменная – *область памяти фиксированного размера*, расположенная по некоторому адресу, которой сопоставлено символьное имя – идентификатор. Величина, хранящаяся в данной области называется значением переменно.

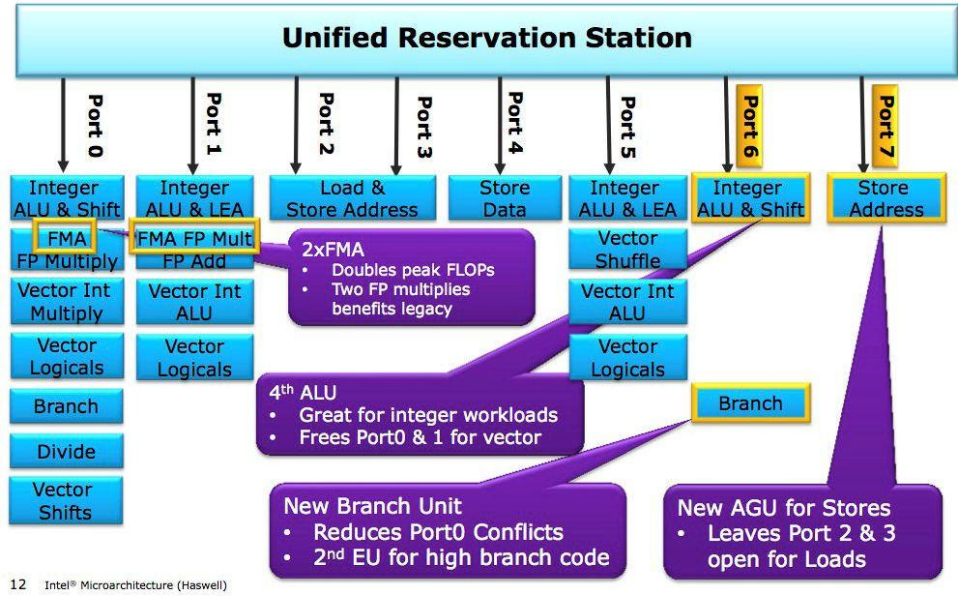
- Длинные "говорящие" имена (*length*, *str*, *array*, *device*, …)
- Существует независимо.
- Диапазон значений ограничен размером области памяти (ячейки).
- Диапазон значений определяется типом данных. Различают целые и вещественные переменные.

Переменная [формат]

В вычислительной технике различают три основных формата представления данных:

- целочисленный беззнаковый
- целочисленный знаковый
- вещественный

Наличие этих форматов напрямую обусловлено архитектурой современных процессоров.



Переменная [формат] (2)

Формат переменной определяет:

1. Тип значений, которые могут быть сохранены в данной ячейке: целые (со знаком или без) или вещественные.

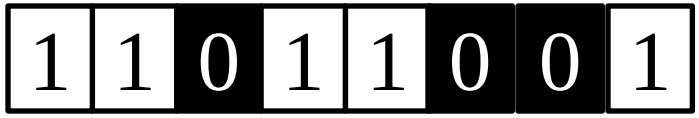
2. Набор допустимых операций над данной ячейкой.

3. Формат ячейки в сочетании с ее размером определяет диапазон допустимых значений.

Обозначение	**Размер, байт**	**Диапазон значений**
[signed] char	1	[-128; 127]
[signed] short	2	[-32768; 32767]
[signed] int	4	[–2147483648; 2147483647]
[signed] long	4 или 8	$[-2^{63}; 2^{63} - 1]$ (8 байт)
float	4	от $\pm 3.4 \cdot 10^{-38}$ до $\pm 3.4 \cdot 10^{38}$ (~ 7 значащих цифр)
double	8	от $\pm 1.7 \cdot 10^{-308}$ до $\pm 1.7 \cdot 10^{308}$ (~ 15 значащих цифр)

Целые беззнаковые переменные [размер]

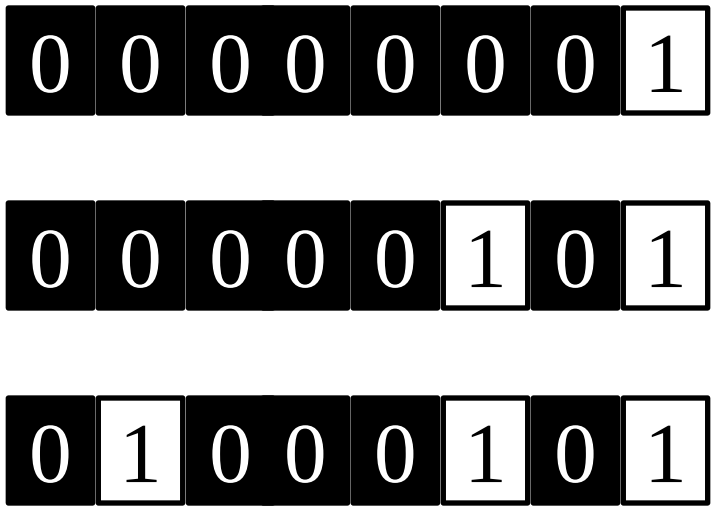
Размер является одной из важнейших характеристик переменной и измеряется в количестве байт, используемых для ее хранения. Размер сопоставляется переменной при ее создании и не может изменяться.



- Как хранить числа меньшей разрядности?
- Что будет, если увеличить значение ячейки, содержащей $1111\ 1111_2$ на 1?
- Что будет, если уменьшить значение ячейки, содержащей 0_2 на 1?
- Что будет, если записать 101 0010 1001 в ячейку размером 1 байт?
- Как хранить числа большей разрядности?

Целые беззнаковые переменные [размер] (2)

Для записи чисел, разрядность которых меньше разрядности переменной, к ним слева приписываются незначащие нули:



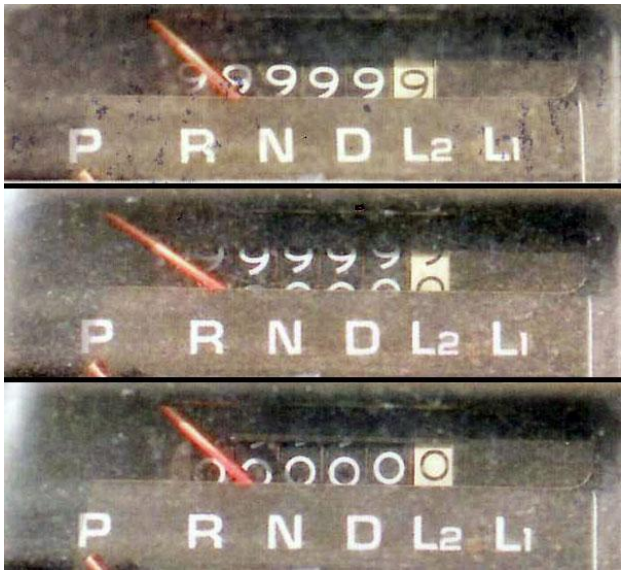
$$1 = 01 = 001 = 0001 = \ldots$$

Переполнение целой беззнаковой переменной

В программировании целочисленное переполнение происходит, когда результат операции сложения превышает размер переменной.

Например, добавление 1 к максимальному числу данной разрядности:

$$1111\ 1111_2 + 1 = 1\ 0000\ 0000_2$$



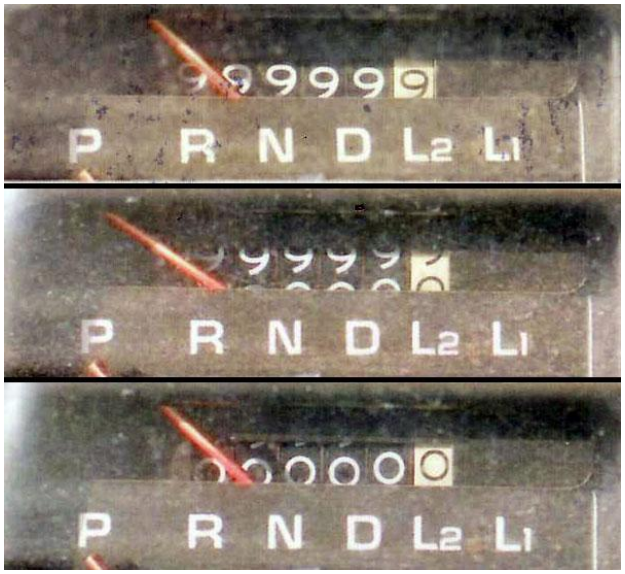
Математика: 255 + 1 = 256

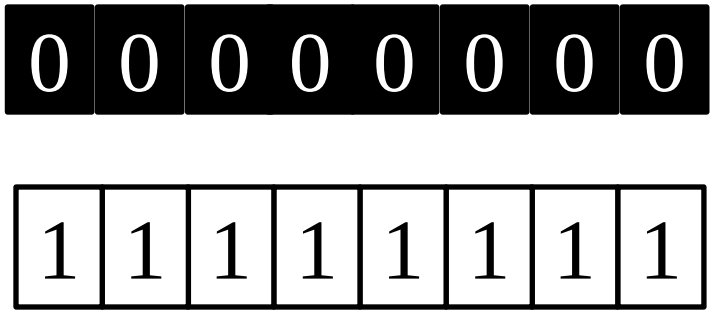
Вычислительная техника: 255 + 1 = 0

Переполнение целой беззнаковой переменной (2)

Ситуация, в которой результат операции вычитания меньше *значения* переменной, обрабатывается путем дописывания виртуального старшего разряда к уменьшаемому:

$$\mathbf{0 - 1 = 1\ 0000\ 0000_2 - 1 = 1111\ 1111_2}$$



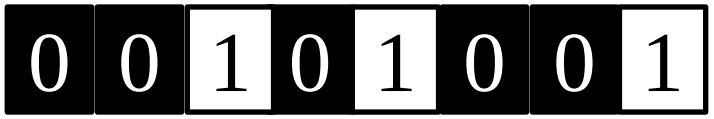


Аналогия с механическим одометром: набором шестерней, связанных друг с другом определенным передаточным числом

Целые беззнаковые переменные [размер] (3)

Запись в ячейку значения, превышающего ее размер приводит к ее переполнению и, как следствие, отбрасыванию старших разрядов, не помещающихся в отведенной памяти:

x = ~~101~~ 0010 1001

101 

Для хранения чисел большей разрядности необходимо использовать переменные, размер которых составляет 2, 4 или 8 байт.

T04.1 Переполнение целых беззнаковых переменных

Система счисления констант (Си). Первые символы константы:

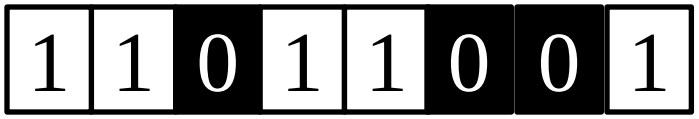
- **[1-9]** – десятичное число
- **0** – восьмеричное число
- **0x** – шестнадцатеричное число

```
unsigned char c;
unsigned short s;

1)  c = 057;
2)  c = 0xF0;
3)  c = 0450;
4)  c = 0x450;
5)  c = 0x3416;
6)  c = 03416;
7)  s = 0x450;
8)  s = 0x3416;
9)  s = 03416;
10) s = 0x1234567;
11) s = 01234567;
```

Целая знаковая переменная

Как было сказано ранее для представления знака в ВТ не предусмотрено специальных средств. Знаковые числа представляются в дополнительном коде. При этом знак числа определяется старшим битом целочисленной ячейки.



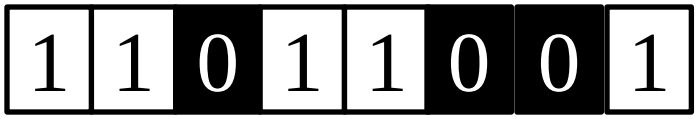
Целое беззнаковое: 217
Целое знаковое: -39
256 – 39 = 217

$$x' = \begin{cases} x, & x \geq 0 \\ 0 - |x| = 100000000_2 - |x| = 256 - |x|, & x < 0 \end{cases}$$

$$x = \begin{cases} x', & x \leq 127 \\ x' - 100000000_2 = x' - 256, & x > 127 \end{cases}$$

Целая знаковая переменная (2)

Как было сказано ранее для представления знака в ВТ не предусмотрено специальных средств. Знаковые числа представляются в дополнительном коде. При этом знак числа определяется старшим битом целочисленной ячейки.



Целое беззнаковое: 217
Целое знаковое: -39
256 – 39 = 217

```
char c = -39;
printf("char: %hhd\n", c);
printf("unsigned char = %hhu\n", c);
```

Математика: –39 = – 39
Вычислительная техника: – 39 = 217

Переполнение целых знаковых переменных

```
#include <stdio.h>

int main()
{
    char c = -200;
    printf("char: %hhd\n", c);
    printf("unsigned char = %hhu\n", c);
    return 0;
}
```

```
$ gcc -o test test.c
test.c: In function ‘main’:
test.c:5:5: warning: overflow in implicit constant conversion [-Woverflow]
    char c = -200;
    ^
```

Переполнение целых знаковых переменных (2)

```
#include <stdio.h>

int main()
{
    char c = -200;
    printf("char: %hhd\n", c);
    printf("unsigned char = %hhu\n", c);
    return 0;
}
```

```
$ ./test
char: 56
unsigned char = 56
```

x = -200
x < 0
x' = 256 – 200 = 56 !

Переполнение целых знаковых переменных (3)

```
#include <stdio.h>

int main()
{
    char c = -400;
    printf("char: %hhd\n", c);
    printf("unsigned char = %hhu\n", c);
    return 0;
}
```

400 > 256

$400_{10} = 1\ 0111\ 0000_2$

8 разрядов: ~~1~~ $0111\ 0000_2 = 0111\ 0000_2 = 144_2$

Знак "-": **256 – 144 = 112**

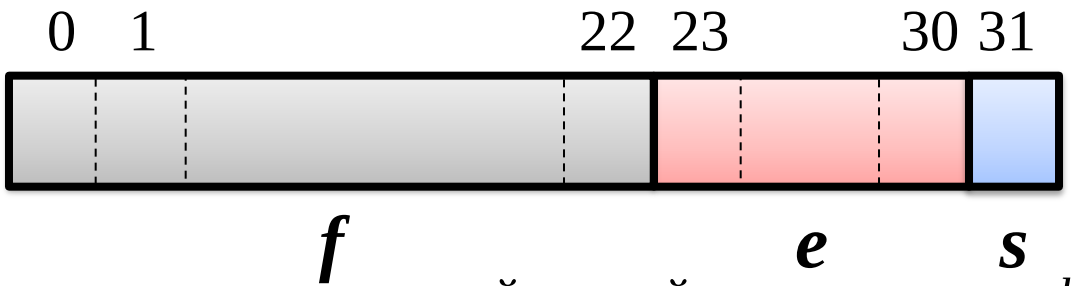
```
$ ./test
char: 112
unsigned char = 112
```

T04.2 Переполнение целых знаковых переменных

$$x' = \begin{cases} x, & x \geq 0 \\ 256 - |x|, & x < 0 \end{cases} \qquad x = \begin{cases} x', & x \leq 127 \\ x' - 256, & x > 127 \end{cases}$$

```
unsigned char c;
            1) c = 057;
            2) c = -0xF0;
            3) c = -0450;
            4) c = 0x450;
            5) c = -0x3415;
            6) c = -03415;
```

Числа с плавающей точкой



p-разрядным числом с плавающей точкой по основанию *b* с избытком *q* называется пара величин (*e*,*f*), которой соответствует значение:

$$(e, f) = f \cdot b^{(e-q)}$$

e – порядок – ***беззнаковое*** целое число.

f – мантисса – нормализованное знаковое с ***фиксированной точкой***.

q – избыток, для знакового представления ***порядка***.

Распространенным методом нормализации является приведение числа к виду, в котором целая часть является нулевой:

1) наиболее значимая цифра в представлении *f* отлична от нуля:

$$1/b \le |f| < 1 \quad (1/10=0.1 \le |f| < 1)$$

2) $f = 0$ и *e* принимает наименьшее возможное значение

Например: Только выделенное представление числа 650 нормализовано:

$$\underline{6500}\cdot 10^{-1},\ \underline{650},\ \underline{65}.0\cdot 10,\ \underline{6}.5\cdot 10^{2},\ \mathbf{0.65\cdot 10^{3}},\ 0.\underline{0}65\cdot 10^{4}$$

Представление вещественных чисел (2)

Размер, выделяемый для хранения: 4 байта (32 бита)

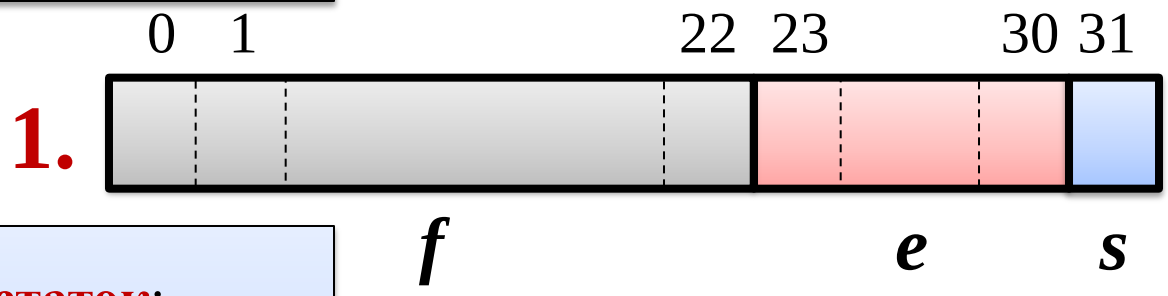
$(e, f, s) = s\mathbf{1.}f \cdot b^{e-q}$

f: 23 бит; e: 8 бит; s: 1 бит,

$b = 2$, $q = 2^{8-1} - 1 = 127$

Стандарт IEEE 754-2008 Single precision

Мантисса
23 **реальных** разряда
24 **виртуальных** разряда



Недостаток:
мантисса *не может быть нулевой*!
0 представляется наименьшим возможным числом, представимом в данном формате

Нормализованная дробь (IEEE 754-2008) :

$$1_2 \le |f| < 10_2$$

Например:

$10101.11 \rightarrow \mathbf{1.}010111 \cdot 2^4$,

$0.0001010 \rightarrow \mathbf{1.}01 \cdot 2^{-4}$

T04.3 Машинный ноль

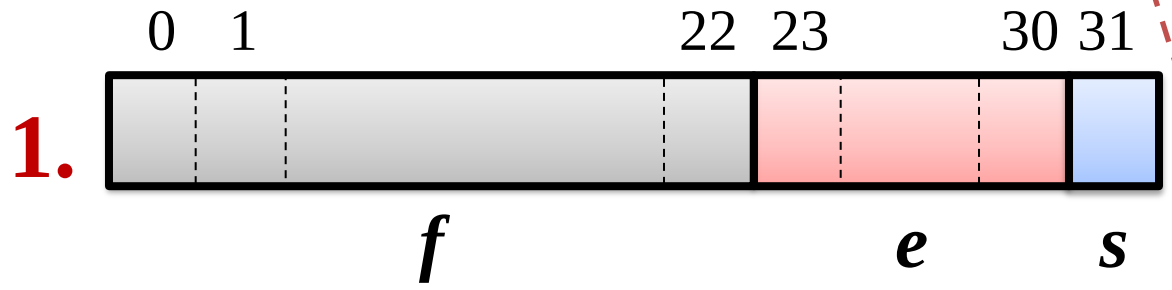
Размер, выделяемый для хранения: 4 байта (32 бита)

$(e, f, s) = s\mathbf{1}.f \cdot b^{e-q}$

f: 23 бит; e: 8 бит; s: 1 бит,

$b = 2$, $q = 2^{8-1} - 1 = 127$

Стандарт IEEE 754-2008 Single precision



Нормализованная дробь (IEEE 754-2008) : $1_2 \le |f| < 10_2$

Например:

$10101.11 \rightarrow \mathbf{1}.010111 \cdot 2^4$, $0.0001010 \rightarrow \mathbf{1}.01 \cdot 2^{-4}$

Правило: Ноль представляется наименьшим возможным числом, представимом в данном формате.

Учитывая приведенное выше правило, запишите представление нуля в числе с плавающей точкой.

T04.3 Машинный ноль [ответ]

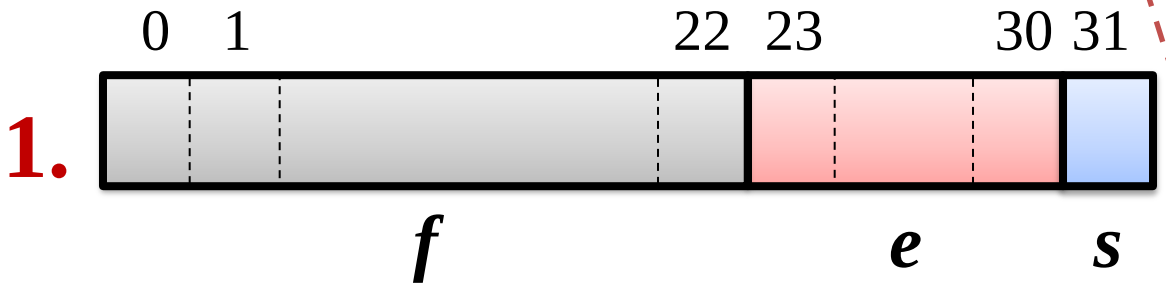
Размер, выделяемый для хранения: 4 байта (32 бита)

$(e, f, s) = s\mathbf{1.}f \cdot b^{e-q}$

f: 23 бит; e: 8 бит; s: 1 бит,

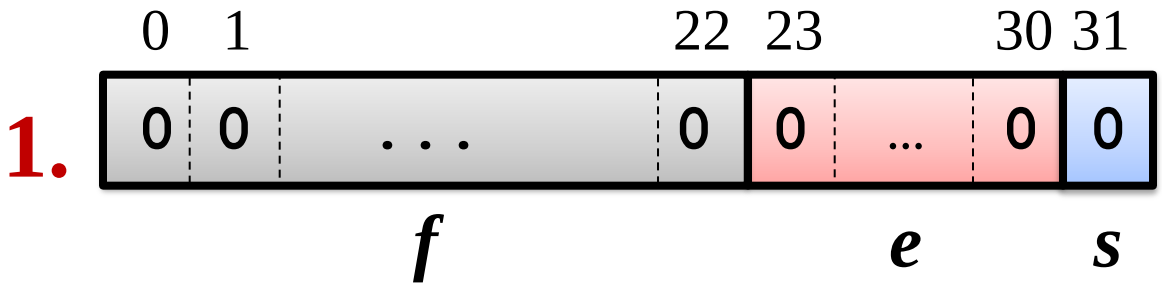
$b = 2$, $q = 2^{8-1} - 1 = 127$

Стандарт IEEE 754-2008 Single precision



Правило: Ноль представляется наименьшим возможным числом, представимом в данном формате.

+/- 1.0 ·2^{-127}



T04.3 Машинный ноль [ответ] (2)

Размер, выделяемый для хранения: 4 байта (32 бита)

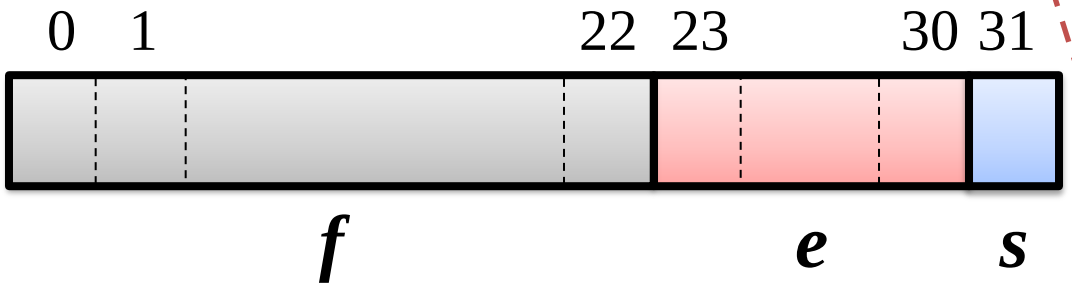
$(e, f, s) = s\mathbf{1}.f \cdot b^{e-q}$

f: 23 бит; e: 8 бит; s: 1 бит,

$b = 2$, $q = 2^{8-1} - 1 = 127$

Стандарт IEEE 754-2008 Single precision

1.




```
int main()
{
    float f = 1E-50;
    printf("char: %e", f);
    char *ptr = (char*)&f;
    return 0;
}
```

Машинный ноль

Идентификатор	Размер, байт	Диапазон значений
float	4	от $\pm 3.4 \cdot 10^{-38}$ до $\pm 3.4 \cdot 10^{38}$ (~ 7 значащих цифр)

```
#include <stdio.h>

int main()
{
    float f = 1;
    double d = 1;
    int i;
    for(i=0;i<160;i++){
        f /= 2;
        d /= 2;
        printf("%d: %e = %le\n", i, f, d);
    }
    return 0;
}
```

Машинный ноль (2)

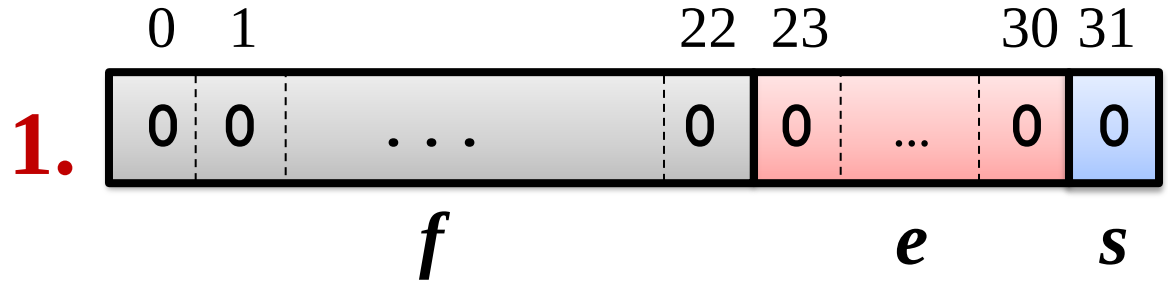
Идентификатор	Размер, байт	Диапазон значений
float	4	от $\pm$~~$3.4 \cdot 10^{-38}$~~ до $\pm 3.4 \cdot 10^{38}$ от $\pm \mathbf{1.4 \cdot 10^{-45}}$ до $\pm 3.4 \cdot 10^{38}$ (~ 7 значащих цифр)

```
for(i=0;i<160;i++){
    f /= 2;
    d /= 2;
    printf("%d: %e = %le\n", i, f, d);
}
```

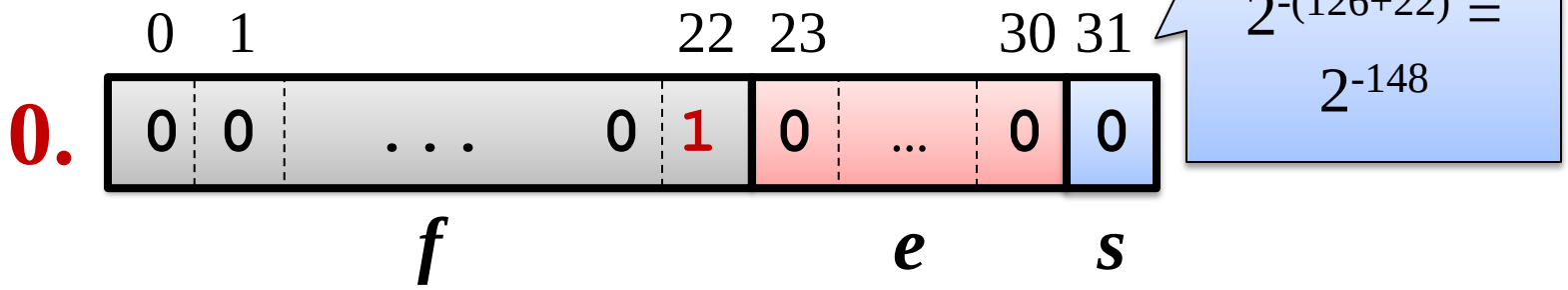
```
145: 1.121039e-44 = 1.121039e-44
146: 5.605194e-45 = 5.605194e-45
147: 2.802597e-45 = 2.802597e-45
148: 1.401298e-45 = 1.401298e-45
149: 0.000000e+00 = 7.006492e-46
150: 0.000000e+00 = 3.503246e-46
```

Машинный ноль [диапазон]

Возможность увеличения нижнего диапазона возникает из следующей особенности IEEE 754-2008. Двоичное представление вещественного нуля предусматривает нули во всех разрядах мантиссы и порядка:



Однако если мантисса не нулевая, а порядок – нулевой, то виртуальная единица **игнорируется**:




```
147: 2.802597e-45 = 2.802597e-45
148: 1.401298e-45 = 1.401298e-45
149: 0.000000e+00 = 7.006492e-46
```

Машинный ноль

Идентификатор	Размер, байт	Диапазон значений
float	4	от $\pm$~~$3.4 \cdot 10^{-38}$~~ до $\pm 3.4 \cdot 10^{38}$ от $\pm \mathbf{1.4 \cdot 10^{-45}}$ до $\pm 3.4 \cdot 10^{38}$ (~ 7 значащих цифр)

```
#include <stdio.h>

int main()
{
    float f = 0.3;
    double d = 0.3;
    int i;
    for(i=0;i<160;i++){
        f /= 2;
        d /= 2;
        printf("%d: %.7e = %.7le\n", i, f, d);
    }
    return 0;
}
```

Машинный ноль (3)

Идентификатор	Размер, байт	Диапазон значений
float	4	**от $\pm 3.4 \cdot 10^{-38}$ до $\pm 3.4 \cdot 10^{38}$** ~~от $\pm$ **$1.4 \cdot 10^{-45}$** до $\pm 3.4 \cdot 10^{38}$~~ (~ 7 значащих цифр)

```
for(i=0;i<160;i++){
    f /= 2;
    d /= 2;
    printf("%d: %.7e = %.7le\n", i, f, d);
}
```

```
126: 1.7632412e-39 = 1.7632415e-39
127: 8.8162132e-40 = 8.8162076e-40
128: 4.4081066e-40 = 4.4081038e-40
129: 2.2040463e-40 = 2.2040519e-40
130: 1.1020232e-40 = 1.1020260e-40
```

Машинный ноль (4)

Идентификатор	Размер, байт	Диапазон значений
float	4	**от $\pm 3.4 \cdot 10^{-38}$ до $\pm 3.4 \cdot 10^{38}$** ~~от $\pm$ **$1.4 \cdot 10^{-45}$** до $\pm 3.4 \cdot 10^{38}$~~ (~ 7 значащих цифр)

```
126: 1.7632412e-39 = 1.7632415e-39
127: 8.8162132e-40 = 8.8162076e-40
128: 4.4081066e-40 = 4.4081038e-40
129: 2.2040463e-40 = 2.2040519e-40
130: 1.1020232e-40 = 1.1020260e-40
131: 5.5101858e-41 = 5.5101298e-41
132: 2.7550929e-41 = 2.7550649e-41
133: 1.3774764e-41 = 1.3775324e-41
134: 6.8873820e-42 = 6.8876622e-42
135: 3.4443916e-42 = 3.4438311e-42
136: 1.7221958e-42 = 1.7219156e-42
137: 8.6039726e-43 = 8.6095778e-43
```

T04.4 Машинная бесконечность

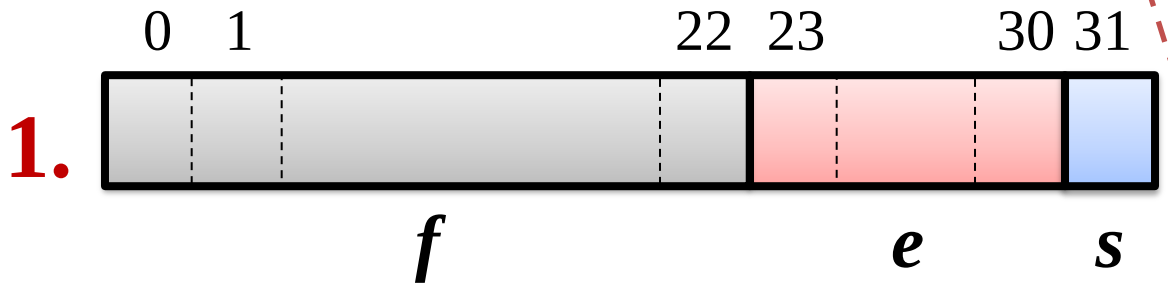
Размер, выделяемый для хранения: 4 байта (32 бита)

$(e, f, s) = s\mathbf{1}.f \cdot b^{e-q}$

f: 23 бит; e: 8 бит; s: 1 бит,

$b = 2$, $q = 2^{8-1} - 1 = 127$

Стандарт IEEE 754-2008 Single precision



Нормализованная дробь (IEEE 754-2008) : $1_2 \le |f| < 10_2$

Например:

$10101.11 \rightarrow \mathbf{1}.010111 \cdot 2^4$, $0.0001010 \rightarrow \mathbf{1}.01 \cdot 2^{-4}$

Правило: Число с плавающей точкой считается бесконечным, если порядок достигает значения 127.

По аналогии с программой вычисления нуля, программно вычислите первое вещественное число равное бесконечности.

T04.4 Машинная бесконечность [ответ]

Идентификатор	Размер, байт	Диапазон значений
float	4	от $\pm 3.4 \cdot 10^{-38}$ до $\pm 3.4 \cdot 10^{38}$ (~ 7 значащих цифр)

```
#include <stdio.h>

int main()
{
    float f = 1;
    double d = 1;
    int i;
    for(i=0;i<160;i++){
        f *= 2;
        d *= 2;
        printf("%d: %.7e = %.7le\n", i, f, d);
    }
    return 0;
}
```

T04.4 Машинная бесконечность [ответ] (2)

Идентификатор	Размер, байт	Диапазон значений
float	4	от $\pm 3.4 \cdot 10^{-38}$ до $\pm 3.4 \cdot 10^{38}$ (~ 7 значащих цифр)

```
124: 4.2535296e+37 = 4.2535296e+37
125: 8.5070592e+37 = 8.5070592e+37
126: 1.7014118e+38 = 1.7014118e+38
127: inf = 3.4028237e+38
128: inf = 6.8056473e+38
```

Различия между системами счисления

Еще одним источником неточности является различия между системами счисления: вещественные числа в языке Си могут быть только десятичные (с точки зрения программиста), однако в памяти данные все равно хранятся в двоичной системе счисления.
Поэтому числа, равные степени 2 имеют максимальную точность:

$$2^{-148} = 2.8025969e-45$$

В то время как некоторые конечные десятичные дроби в двоичной системе счисления являются бесконечными и точно представлены быть не могут:
0.3 = 0.0100110011001100110011001100110011001100110011(0011) (50 знаков).

```
#include <stdio.h>
int main()
{
    float f = 0.1;
    printf("%f/%.9f\n", f, f);
    return 0;
}
```

```
$ ./float_precise
0.300000/0.300000012
```

Математика и ВТ (вещественные числа)

float	**4**	**от $\pm 3.4 \cdot 10^{-38}$ до $\pm 3.4 \cdot 10^{38}$ (~ 7 значащих цифр)**

Математика	ВТ (float)
$2^{-1} = 0.5$	$2^{-1} = 0.5$
$0.3 = 0.3$	$0.3 = 0.300000\mathbf{12}\ldots$
$2^{-127} = 2.938736 \cdot 10^{-39}$	$2^{-127} = 2.93873610^{-39}$
$0.3 \cdot 2^{-127} = 8.8162132 \cdot 10^{-40}$	$0.3 \cdot 2^{-127} = 8.8162132 \cdot 10^{-40}$
$0.3 \cdot 2^{-128} = 4.4081038 \cdot 10^{-40}$	$0.3 \cdot 2^{-128} = 4.4081066 \cdot 10^{-40}$
$2^{-148} = 1.401298 \cdot 10^{-45}$	$2^{-148} = 1.401298 \cdot 10^{-45}$
$2^{-149} = 7.006492 \cdot 10^{-46}$	**$2^{-149} = 0$**
$2^{126} = 1.7014118 \cdot 10^{38}$	$2^{126} = 1.7014118 \cdot 10^{38}$
$2^{127} = 3.4028237 \cdot 10^{38}$	**$2^{127} = +\infty$**