Представлення чисел в комп'ютері

Доц. Скакун С.В.

Позиционная система счисления

- Целое число *x* в *b*-ричной системе счисления представляется в **виде конечной линейной комбинации степеней числа** *b*

$x = \sum_{k=0}^{n-1} a_k b^k$, где a_k — это целые числа, называемые **цифрами**, удовлетворяющие неравенству $0 \leq a_k \leq b-1$

$x = a_{n-1}a_{n-2}\dots a_0 \qquad x = (a_{n-1}a_{n-2}\dots a_0)_b$

- Важные системы счисления
  - двоичная, восьмеричная и шестнадцатеричная

2, 8, 10, 16 представление

Десятичное	Двоичное	Восьмеричное	Шестнадцатеричное
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Число 2001

Двоичное число

1 1 1 1 1 0 1 0 0 0 1

$1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

1024 + 512 + 256 + 128 + 64 + 0 + 16 + 0 + 0 + 0 + 1

Восьмеричное число

3 7 2 1

$3 \times 8^3 + 7 \times 8^2 + 2 \times 8^1 + 1 \times 8^0$

1536 + 448 + 16 + 1

Десятичное число

2 0 0 1

$2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 1 \times 10^0$

2000 + 0 + 0 + 1

Шестнадцатеричное число

7 D 1

$7 \times 16^2 + 13 \times 16^1 + 1 \times 16^0$

1792 + 208 + 1

Правила

- 2
  - 1 + 1 = 10
- 8
  - 7 +1 = 10
- 16
  - F + 1 = 10

$$\begin{array}{r} {\scriptstyle 1\ 1\ \ \ } \\ 1A9F \\ +\ 3AB \\ \hline 1E4A \end{array}$$

(1) F + B = F + 1 + A = 10 + A = 1A

(2) 1 + 9 + A = A + A = 14

(3) 1 + A + 3 = E

Преобразование чисел из одной системы счисления в другую

- Перевод произвольной позиционной системы счисления в десятичную

Если число в *b*-ричной системе счисления равно

$$a_1\ a_2\ a_3 \ldots a_n,$$

то для перевода в десятичную систему вычисляем такую сумму:

$$\sum_{i=1}^{n} a_i \cdot b^{n-i}$$

или, в более наглядном виде:

$$a_1 \cdot b^{n-1} + a_2 \cdot b^{n-2} + \ldots + a_{n-1} \cdot b^1 + a_n \cdot b^0,$$

- например

$$101100_2 =$$
$$= 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 1 =$$
$$= 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 =$$
$$= 32 + 8 + 4 + 0 = 44_{10}$$

Перевод из десятичной в произвольную позиционную систему счисления

- Целая часть
  - 1. Последовательно **делить целую** часть десятичного числа на **основание**, пока десятичное число не станет равно нулю.
  - 2. Полученные при делении **остатки** являются **цифрами** нужного числа. Число **в новой системе** записывают, начиная с **последнего остатка**.

44_{10} переведём в двоичную систему

```
44 делим на 2. частное 22, остаток 0
22 делим на 2. частное 11, остаток 0
11 делим на 2. частное  5, остаток 1
 5 делим на 2. частное  2, остаток 1
 2 делим на 2. частное  1, остаток 0
 1 делим на 2. частное  0, остаток 1
```

Частное равно нулю, деление закончено.

Теперь записав все остатки снизу вверх получим число 101100_2

Перевод из двоичной в восьмеричную и шестнадцатеричную системы и наборот

- Для восьмеричной
  - разбиваем переводимое число на количество цифр, равное степени 2, т.е. 3
- Для шестнадцатеричной
  - на 4

преобразуем 101100_2
восьмеричная — 101 100 → 54_8
шестнадцатеричная — 0010 1100 → $2C_{16}$

преобразуем
54_8 → 101 100
$2C_{16}$ → 0010 1100

Отрицательные двоичные числа

- Существует несколько систем
- Рассмотрим дополнение до двух
- **X + (-X) = 0**
- Правило преобразования в отрицательное число
  - Сначала каждая единица меняется на ноль, а каждый ноль — на единицу
  - Затем к полученному результату прибавляется единица
- Число +6:
  - 00000110 ☐ 11111001
  - -6: 11111010
  - Проверка:
    - 00000110 + 11111010 = 0

Числа с плавающей точкой

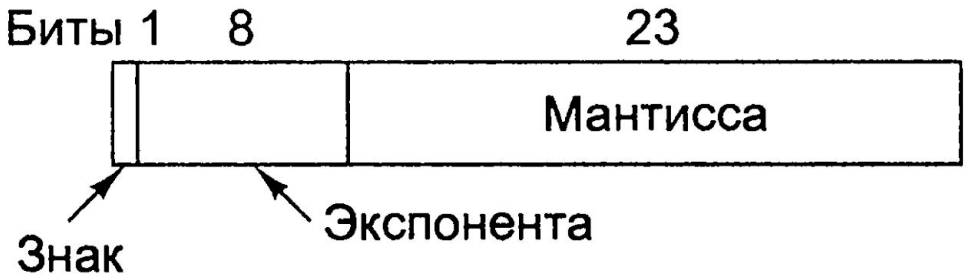
- Дробное число хранится в форме **мантиссы** и **показателя степени**
- Стандарт **IEEE 754**
- Реализация математических операций с числами с плавающей запятой в вычислительных системах может быть как **аппаратная**, так и **программная**

IEEE 754

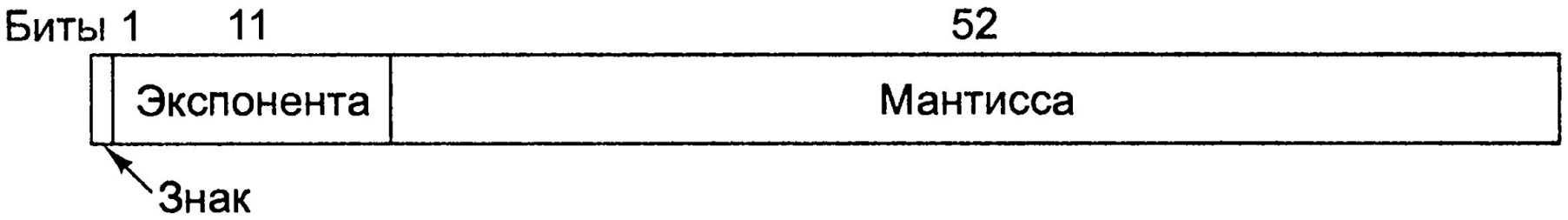
- Определения форматов хранения мантиссы, экспоненты и знака, форматы положительного и отрицательного нуля, плюс и минус бесконечностей, а также определение "не числа" (NaN).
- Методы, которые будут использоваться для округления числа в процессе математических операций.
- Обработка исключительных ситуаций, таких как деление на ноль, переполнение и т.д.

IEEE 754

Одинарная точность



Двойная точность



IEEE 754

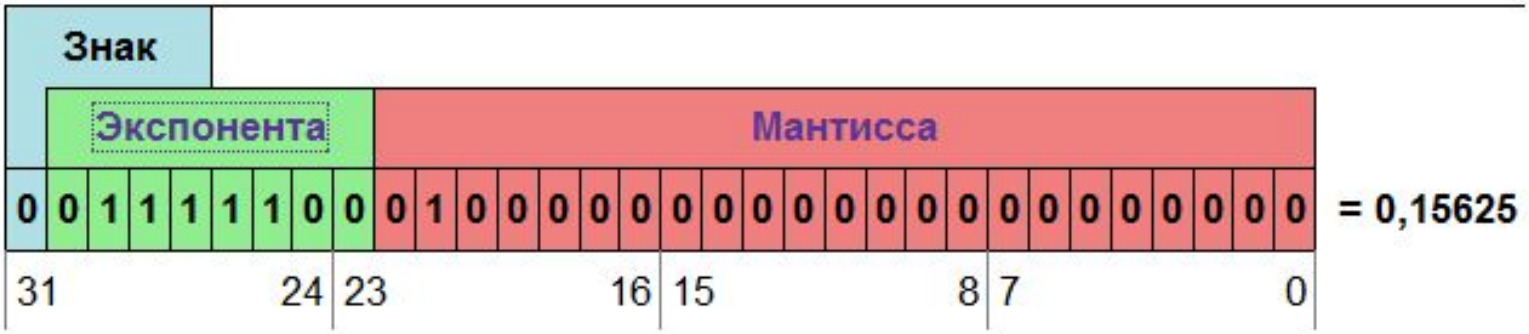
	Одинарная точность	Удвоенная точность
Количество битов в знаке	1	1
Количество битов в экспоненте	8	11
Количество битов в мантиссе	23	52
Общее число битов	32	64
Смещение экспоненты	Смещение 127	Смещение 1023
Область значений экспоненты	От –126 до +127	От –1022 до +1023
Самое маленькое нормализованное число	2^{-126}	2^{-1022}
Самое большое нормализованное число	Приблизительно 2^{128}	Приблизительно 2^{1024}
Диапазон десятичных дробей	Приблизительно от 10^{-38} до 10^{38}	Приблизительно от 10^{-308} до 10^{308}
Самое маленькое ненормализованное число	Приблизительно 10^{-45}	Приблизительно 10^{-324}

IEEE 754 - типы

Нормализованное число	±	0 < Exp < Max	Любой набор битов
Ненормализованное число	±	0	Любой ненулевой набор битов
Нуль	±	0	0
Бесконечность	±	1 1 1...1	0
Не число	±	1 1 1...1	Любой ненулевой набор битов

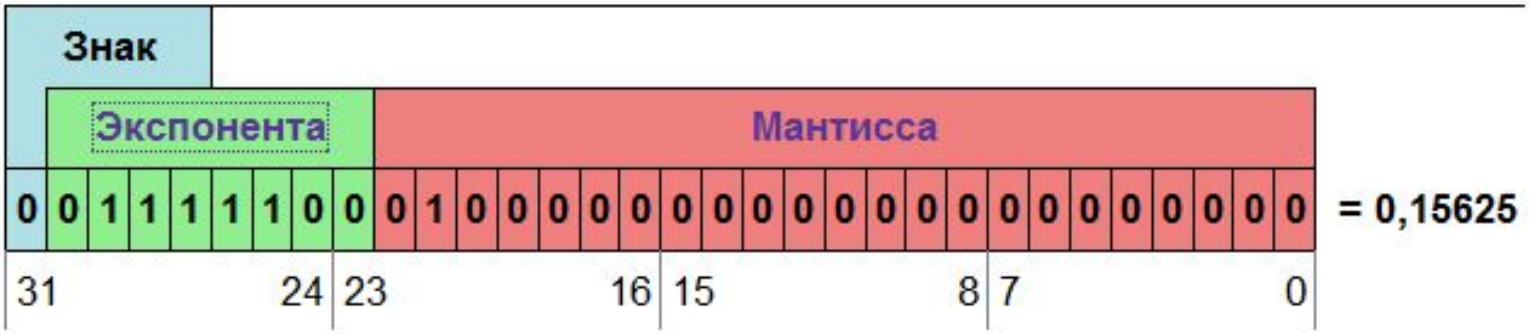
Знаковый бит

Преобразование из IEEE 754



- Экспонента
  - $01111100_2 = 124_{10}$
- Мантисса
  - 01000000000000000000000
- Для вычисления показателя степени из восьмиразрядного поля экспоненты вычитается смещение экспоненты равное 127_{10}
  - $01111100_2 - 01111111_2 = 124_{10} - 127_{10} = -3_{10}$

Преобразование из IEEE 754



- Для вычисления мантиссы к единице добавляется дробная часть мантиссы из 23-х разрядного поля дробной части мантиссы $1,01000000000000000000000_2$.
- Число равно произведению мантиссы со знаком на двойку в степени экспоненты
  - $1,01_2 * 2^{-3} = 101_2 * 2^{-5} = 5 * 2^{-5} = 0,15625$
  - $1,01_2 * 2^{-3} = (1 + 0 * 2^{-1} + 1 * 2^{-2}) * 2^{-3} = 2^{-3} + 2^{-5} = 0,15625$

Преобразование в IEEE 754

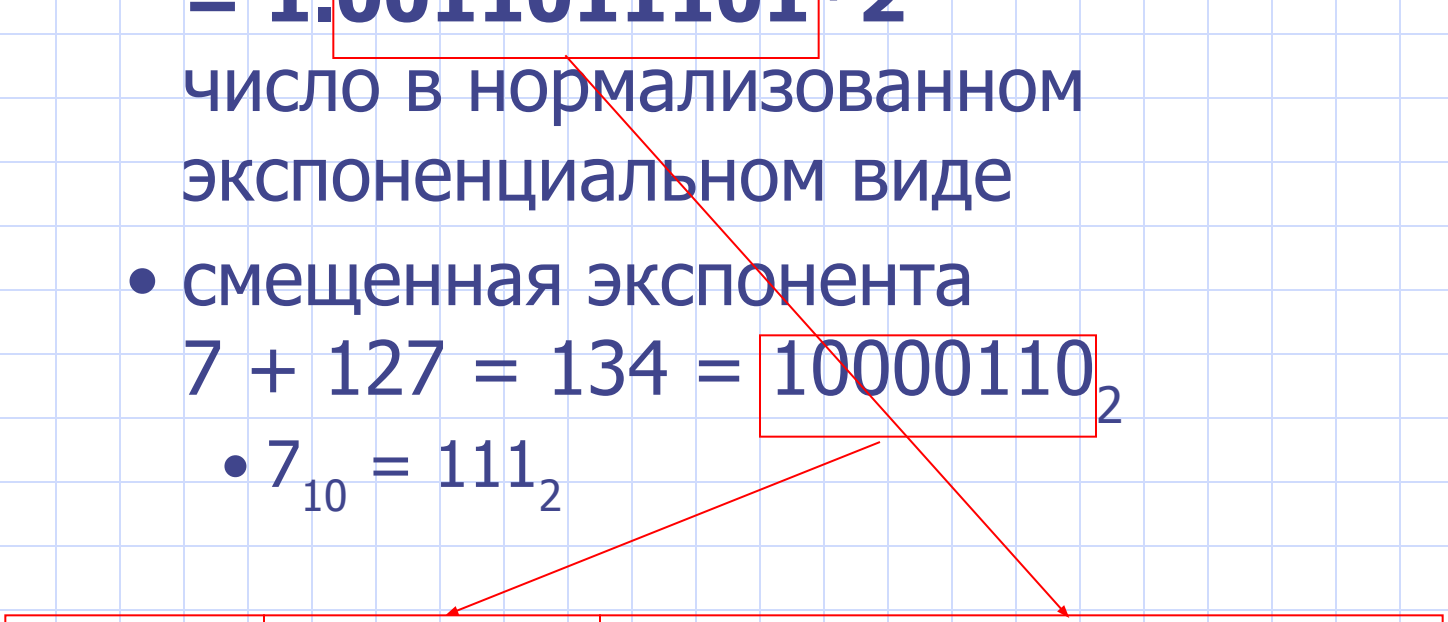
- **13** = 1101 = 1.101 * 2^3
- Экспонента
  - 3 + 127 = 130 = 10000010
- Мантисса
  - 10100000000000000000000
- Число 13 в IEEE 754
  - 0.10000010.10100000000000000000000

Преобразование в IEEE 754

- **155.625**
  - $1*2^7 +0*2^6+0*2^5+1*2^4+1*2^3+0*2^2+1* 2^1+ 1*2^0+1*2^{-1}+0*2^{-2}+1*2^{-3}$
  - 128 + 0 + 0 + 16 + 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125
  - **155.625 = 10011011.101** - число в двоичной системе с плавающей точкой

Преобразование в IEEE 754

- **155.625**
  - **$10011011.101 = 1.0011011101 * 2^{+7} =$**
    $= 1.0011011101 * 2^{+111}$
    число в нормализованном экспоненциальном виде
  - смещенная экспонента
    $7 + 127 = 134 = 10000110_2$
    - $7_{10} = 111_2$



1 бит	8 бит	23 бит
0	10000110	$00110111010000000000000_2$

Примеры - Python

```
>>> a = 2**1000
>>> a
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581275946729175531468251871452856923140435984577574698574803934567774824230985421074605062371141877954182153046474983581941267398767559165543946077062914571196477686542167660429831652624386837205668069376L

>>> x = 0.1
>>> x
0.10000000000000001

>>> s=0.0
>>> for i in range(10): s += 0.1
>>> s
0.9999999999999989
>>> print(s)
1.0
```

Примеры - Python

```
>>> import math
>>> a = math.sqrt(2)
>>> a
1.4142135623730951
>>> a*a == 2
False
>>> a*a
2.0000000000000004
```

Примеры – C++

```
int main(int argc, char* argv[])
{

   double x = 0.1;

   cout << x << endl;
   printf("%1.17lf \n",  x);

   double sum = 0.0;

   for (int i = 0; i < 10; i++)
      sum += 0.1;

   cout << sum << endl;
   printf("%1.17lf \n",  sum);
   return 0;
}
```