

Форматы представления данных в компьютере

Для вопросов по курсу:
natalya.razmochaeva@moevm.info

*Берленко Татьяна Андреевна
Шевская Наталья Владимировна
СПбГЭТУ «ЛЭТИ», ФКТИ, МОЭВМ*

План работы: $\min(\text{Теории}) + \max(\text{Практики})$

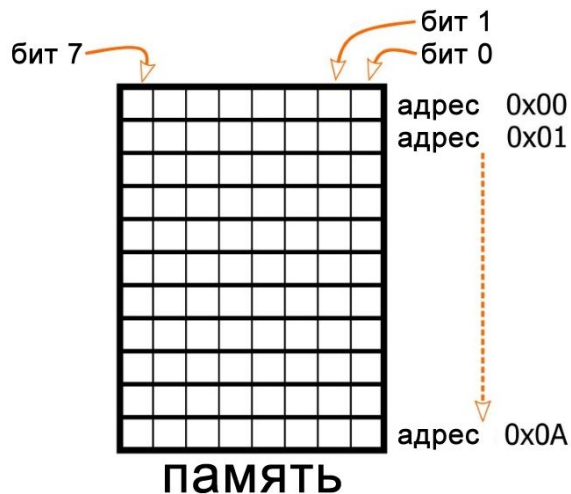
1. Little-endian VS Big-endian
2. Дополнительный код
3. Порядок и мантисса
- ? Пример для MT

Little-endian VS Big-endian

	Low address				High address			
Address	0	1	2	3	4	5	6	7
Little-endian	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Big-endian	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Memory content	0x11	0x22	0x33	0x44	0x55	0x66	0x77	0x88
64 bit value on Little-endian				64 bit value on Big-endian				
0x8877665544332211				0x1122334455667788				

- при работе с многобайтными данными (какой байт считать первым -- младшим, какой последним -- старшим?)

Прямой и обратный порядок байтов



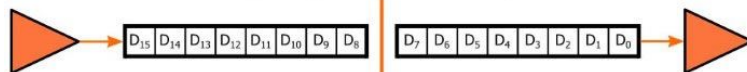
прямой порядок
big endian

D ₃₁	D ₃₀	D ₂₉	D ₂₈	D ₂₇	D ₂₆	D ₂₅	D ₂₄	0x00
D ₂₃	D ₂₂	D ₂₁	D ₂₀	D ₁₉	D ₁₈	D ₁₇	D ₁₆	0x01
D ₁₅	D ₁₄	D ₁₃	D ₁₂	D ₁₁	D ₁₀	D ₉	D ₈	0x02
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	0x03

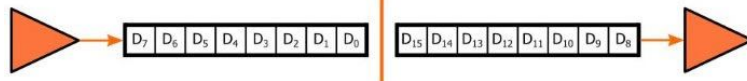
обратный порядок
little endian

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	0x00
D ₁₅	D ₁₄	D ₁₃	D ₁₂	D ₁₁	D ₁₀	D ₉	D ₈	0x01
D ₂₃	D ₂₂	D ₂₁	D ₂₀	D ₁₉	D ₁₈	D ₁₇	D ₁₆	0x02
D ₃₁	D ₃₀	D ₂₉	D ₂₈	D ₂₇	D ₂₆	D ₂₅	D ₂₄	0x03

обратный порядок little endian
(первым передается младший байт)



прямой порядок big endian
(первым передается старший байт)



Пример

Записан текст.

Char* - 1 байт.

Разделим на байты и конвертируем в символы.

1 байт -- 8 бит.

(PyCharm)

```
0101001001100001011000110110010100100000011001100110000101110011011101000
0101100001000000111001101100001011001100110010100100000011000110110000101
110010
```

Формат представления **беззнаковых** целых чисел

123:

7	6	5	4	3	2	1	0
0	1	1	1	1	0	1	1

Диапазон значений:

$0 \dots 2^n - 1$,

где n - разрядность архитектуры

Для $n = 8$:

$0 \dots 2^8 - 1 \Rightarrow 0 \dots 255$

42:

7	6	5	4	3	2	1	0
0	0	1	0	1	0	1	0

128:

7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0

255:

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1

305:

7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	1

ПЕРЕПОЛНЕНИЕ

Формат представления **знаковых** целых чисел

$n = 8$

➤ Прямой код $(-2^{n-1} + 1 \dots 2^{n-1} - 1)$ $(-2^{8-1} + 1 \dots 2^{8-1} - 1) \Rightarrow (-2^7 + 1 \dots 2^7 - 1) \Rightarrow$
 $(-128 + 1 \dots 128 - 1) \Rightarrow \text{(-127 ... 127)}$

-123:

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

➤ Обратный код $(-2^{n-1} + 1 \dots 2^{n-1} - 1)$ $(-2^{8-1} + 1 \dots 2^{8-1} - 1) \Rightarrow \dots \Rightarrow \text{(-127 ... 127)}$

-123:

1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

➤ Доп. код $(-2^{n-1} \dots 2^{n-1} - 1)$ $(-2^{8-1} \dots 2^{8-1} - 1) \Rightarrow (-2^7 \dots 2^7 - 1) \Rightarrow \text{(-128 ... 127)}$

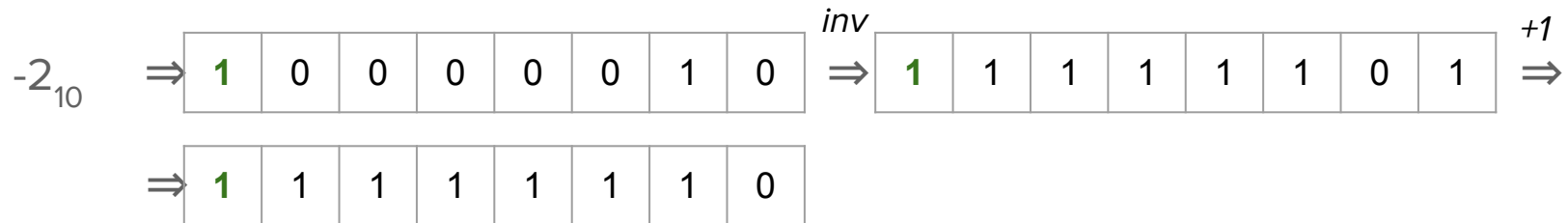
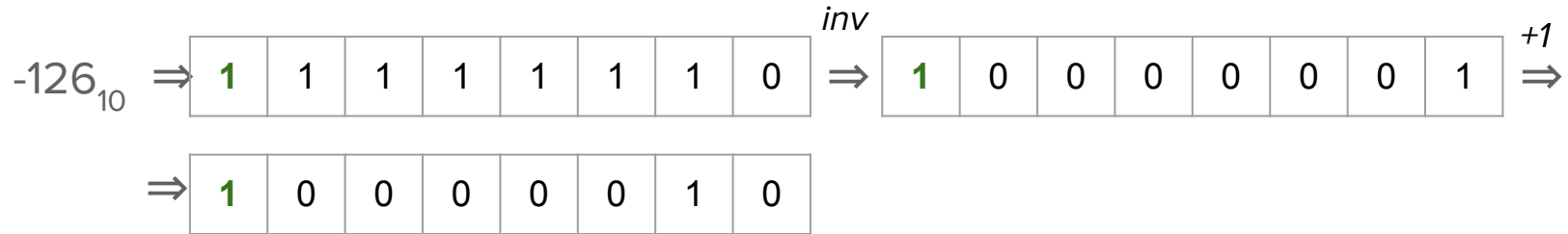
-123:

1	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

Прямой, обратный и дополнительный код

Пример для 8 бит	Прямой	Обратный	Дополнительный
123	0 1 1 1 1 0 1 1	0 1 1 1 1 0 1 1	0 1 1 1 1 0 1 1
-123	1 1 1 1 1 0 1 1	1 0 0 0 0 1 0 0	1 0 0 0 0 1 0 1
-42	1 0 1 0 1 0 1 0	1 1 0 1 0 1 0 1	1 1 0 1 0 1 1 0
-127	1 1 1 1 1 1 1 1	1 0 0 0 0 0 0 0	1 0 0 0 0 0 0 1
-128	impossible	impossible	1 0 0 0 0 0 0 0

Примеры с доп. кодом: $-126 - 2 = -128$



Примеры с доп. кодом: $-126 - 2 = -128$

-126_{10}

1	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

+

-2_{10}

1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

-128_{10}

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---



Примеры с доп. кодом: $126 - 2 = 124$

126_{10}

0	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

+

-2_{10}

1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

124_{10}

0	1	1	1	1	1	0	0
---	---	---	---	---	---	---	---



Примеры с доп. кодом: $-126 + 2 = -124$

-126_{10}

1	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

+

2_{10}

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

-124_{10}

1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

из доп. в прямой код:

1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

inv

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

+1

1	1	1	1	1	1	0	0
---	---	---	---	---	---	---	---



чтобы проверить

Примеры с доп. кодом: $-126 - 2 = -128$

-126_{10}

1	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

+

-2_{10}

1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

-128_{10}

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

чтобы проверить

из доп. в прямой код:

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

inv

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

+1

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---



Одинарная точность

- 1 бит - знак (0 - положительные числа, 1 - отрицательные)
- 8 бит - порядок
- 23 бита - дробная значащая часть числа - мантисса
- 127 - смещение

$$111,1101 = 1,111101 * 2^2$$

1,111101 - мантисса, записывается только дробная часть

2 - *истинный* порядок, 129 - *смещенный* порядок

знак	порядок								мантисса									
0	1	0	0	0	0	0	0	1	1	1	1	1	0	1	...	0	0	0

Двойная точность

- 1 бит - знак (0 - положительные числа, 1 - отрицательные)
- 11 бит - порядок
- 52 бита - дробная значащая часть числа - мантисса
- 1023 - смещение

$$111,1101 = 1,111101 * 2^2$$

1,111101 - мантисса

2 - *истинный* порядок, 1025 - *смещенный* порядок

знак	порядок							мантисса						
0	0	1	0	...	0	0	1	1	1	1	...	0	0	0

Пример 446.15625 в одинарной точности

$446,15625_{10} \Rightarrow$ *отдельно обрабатываем целую часть*

$446_{10} \Rightarrow 110111110_2$ *ставим запятую и начинаем думать про хвост*

не понимаем, какой хвост?

$\Rightarrow 110111110,?????...$ \Rightarrow *сначала выделим порядок*

порядок: 8 знаков влево $\Rightarrow 1,10111110$???????????????? *хвост из 15 знаков*

собираем хвост....

0.15625 в бинарном виде

0,15625	* 2	=	0,3125	⇒	1,10111100??????????????
0,3125	* 2	=	0,625	⇒	1,1011110000??????????????
0,625	* 2	=	1,25	-1 ⇒	1,1011110001??????????????
0,25	* 2	=	0,5	⇒	1,10111100010??????????????
0,5	* 2	=	1,0	-1 ⇒	1,101111000101??????????????

Где взять все остальные знаки? Забить нулями!

1,101111000101000000000000

превратим в **знак | порядок | мантисса**


446.15625 в памяти компьютера

Знак: 0

Порядок: истинный 8 + смещение $127 = 135_{10} \Rightarrow 1000\ 011_2$

Мантисса: 1,101111100010100000000000

Итого:

0 | 1000 0111 | 101111100010100000000000 

Двойная точность:

Знак: 0

Порядок: истинный 8 + смещение $1023 = 1031_{10} \Rightarrow 100\ 0000\ 011_2$

Мантисса: 1,101111100010100000000000 + 29 нулей

Собираем число обратно в 10-СС

1 | 1000 0111 | 110 0011 1011 0000 0000 0000

Знак: 1 -- отрицательное

Порядок: $1000\ 0111_2 \Rightarrow 135_{10}$ (смещенный) $\Rightarrow 135 - 127 = 8$ (истинный)

Мантисса:

110 0011 1011 0000 0000 0000 *вспоминаем про единицу*

1,110 0011 1011 0000 0000 0000 *двигаем запятую на 8 знаков вправо*

1110 0011 1,011 0000 0000 0000

1 1100 011,011 0000 0000 0000 *извлекаем целую часть и переводим обычным образом*

$1\ 1100\ 011_2 \Rightarrow 455_{10}$

переводим хвост привычным способом:


Переводим 0,011 в 10-СС

0,011 0000 0000 0000

0	,	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16
0	,	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

$$= 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} + 0 \cdot 2^{-5} + \dots + 0 \cdot 2^{-16} = 0 + \frac{1}{4} + \frac{1}{8} + 0 + \dots + 0 = 0,375$$

Соединяем “голову” и хвост, припоминая про отрицательный знак:

- 455,375 

446,15625

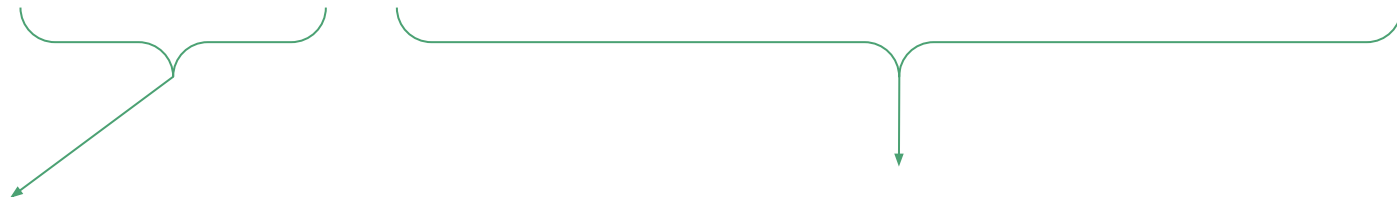


переводим 2-СС делением на 2
собираем **остатки в обратном порядке**



переводим в 2-СС умножением на 2
собираем **целые части в прямом порядке**

1 | 1000 0111 | 110 0011 1011 0000 0000 0000



1) вычисляем истинный порядок

2) идем в мантиссу и выделяем оттуда целую часть 10-го числа

целая часть

дробная часть

3) переводим двоичное представление целого в 10-СС

4) двоичный “хвост” переводим в 10-СС с отрицательными степенями

5) собираем число из знака, целой и вещественной части

Чем двойная точность отличается от одинарной?

Знак -- тот же самый (1 бит)

Порядок -- вместо 8 работаем с **11** битами
(смещение из-за этого равно не 127, а **1023**)

Мантисса -- вместо 23 бита работаем с **52** битами
(в худшем случае при переводе дробной части выполним на 29 итераций больше)

Пример с “плохим” хвостом числа

0,8919

Какая целая часть у этого числа?

Какой **порядок** у этого числа?

Сколько итераций для вычисления **мантиссы** придется сделать?

0,8919 переводим в 2-СС

$$1) 0,8919 * 2 = 1,7838 \implies 0,1 \quad (-1)$$

$$2) 0,7838 * 2 = 1,5676 \implies 0,11 \quad (-1)$$

$$3) 0,5676 * 2 = 1,1352 \implies 0,111 \quad (-1)$$

$$4) 0,1352 * 2 = 0,2704 \implies 0,1110$$

$$5) 0,2704 * 2 = 0,5408 \implies 0,11100$$

$$6) 0,5408 * 2 = 1,0816 \implies 0,111001 \quad (-1)$$

....

Мантисса и порядок для числа 0,8919

$$0,111001?.. = 1,11001?.. * 2^{(-1)}$$

....

$$0,0816 * 2 =$$

истинный -1 ==> -1 + 127 = 126 ==> 0111 1110

0 | 0111 1110 | 11001?

сколько итераций нужно сделать еще?

$$23-5 = 18$$

Источники

- Ч. Петцольд “Код”
- Э. Таненбаум “Архитектура Компьютера”
- <https://stepik.org/course/253> Курс на Stepik “Введение в Архитектуру ЭВМ”

Вопросы по курсу можно задавать:

Шевская Наталья Владимировна
natalya.razmochaeva@moevm.info,

Берленко Татьяна Андреевна
tatyana.berlenko@moevm.info