

Компьютерные основы программирования
Представление данных
часть 2

Лекция 3, 2 марта 2017

Лектор: Чуканова Ольга
Владимировна

Кафедра информатики

602 АК

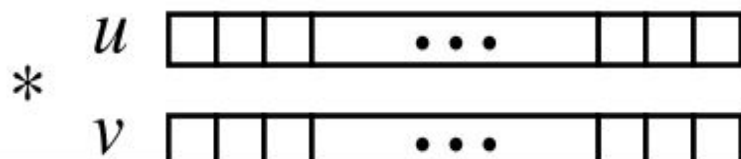
ovcha@mail.ru

Умножение

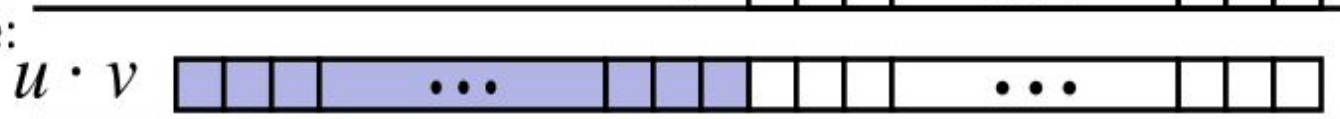
- Вычисление полного произведения w -битовых чисел x, y
 - Или знаковых или беззнаковых
- Границы
 - Беззнаковые: $0 \leq x * y \leq (2^w - 1)^2 = 2^{2w} - 2^{w+1} + 1$
 - До $2w$ бит
 - Минимальное в доп.коде: $x * y \geq (-2^{w-1}) * (2^{w-1} - 1) = -2^{2w-2} + 2^{w-1}$
 - До $2w-1$ бит
 - Максимальное в доп.коде: $x * y \leq (-2^{w-1})^2 = 2^{2w-2}$
 - До $2w$ бит, но только для $(TMin_w)^2$
- Обеспечение полноты результата
 - Необходимо расширение слова для каждого результата
 - Реализуется лишь в программных пакетах арифметики “произвольной точности”

Беззнаковое произведение в Си

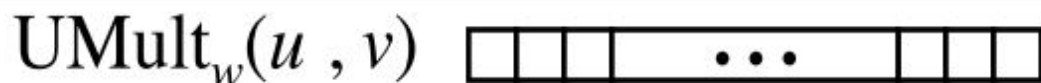
Операнды: w бит



Полное произведение:
 $2 * w$ бит



Без w старших бит:
 w младших бит



- Стандартная функция умножения
 - Игнорирует старшие w бит
- Реализует умножение по модулю
$$\text{UMult}_w(u, v) = u \cdot v \bmod 2^w$$

Пример небезопасного кода

```
void* copy_elements(void *ele_src[], int ele_cnt, size_t ele_size) {
    /*
     * Занимаем буфер для ele_cnt объектов, каждый ele_size байт
     * и копируем с места указанного ele_src
     */
    void *result = malloc(ele_cnt * ele_size);
    if (result == NULL)
        /* Ошибка malloc */
        return NULL;
    void *next = result;
    int i;
    for (i = 0; i < ele_cnt; i++) {
        /* Копирование объектов по назначению */
        memcpy(next, ele_src[i], ele_size);
        /* Перемещение указателя на следующую позицию в памяти */
        next += ele_size;
    }
    return result;
}
```

`malloc(ele_cnt * ele_size)`

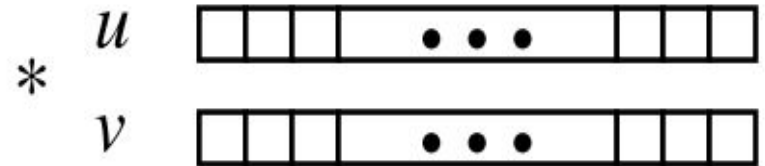
■ Что если:

- `ele_cnt` = $2^{20} + 1$
- `ele_size` = 4096 = 2^{12}
- Выделится = ??

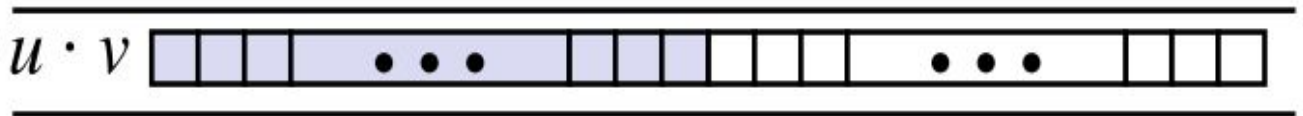
■ Как сделать функцию безопасной?

Знаковое умножение в Си

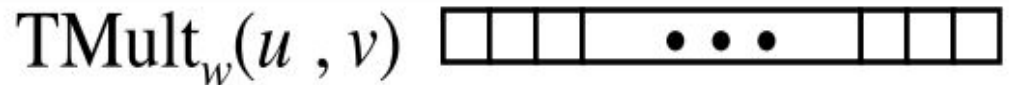
Операнды: w бит



Полное произведение: $u \cdot v$
 $2 * w$ бит



Без w старших бит:
 w младших бит



- Стандартная функция умножения
 - Игнорирует старшие w бит
 - Некоторые из них различаются для знакового и беззнакового умножений
 - Младшие биты такие-же

**Таблица 2.15. Умножение трехбитовых чисел без знака
и в дополнительном коде**

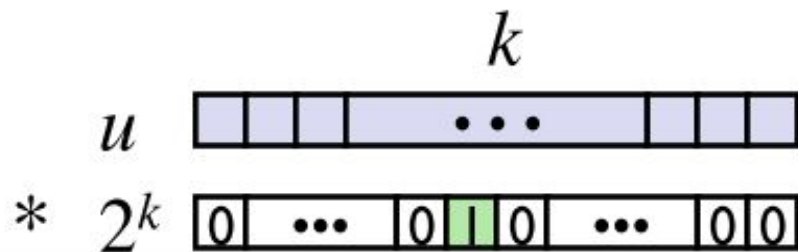
Режим	x		y		$x \cdot y$		Усеченное $x \cdot y$	
Без знака	5	[101]	3	[011]	15	[001111]	7	[111]
В доп. коде	-3	[101]	3	[011]	-9	[110111]	-1	[111]
Без знака	4	[100]	7	[111]	28	[011100]	4	[100]
В доп. коде	-4	[100]	-1	[111]	4	[000100]	-4	[100]
Без знака	3	[011]	3	[011]	9	[001001]	1	[001]
В доп. коде	3	[011]	3	[011]	9	[001001]	1	[001]

Умножение сдвигом на степень двойки

■ Операция

- $u \ll k$ даёт $u * 2^k$
- Для знаковых и беззнаковых

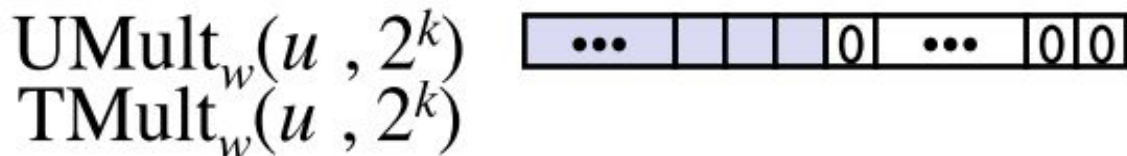
Операнд: w бит



Полное произведение:
 $w+k$ бит



Без k старших бит:
 w младших бит



■ Примеры

- $u \ll 3 \quad == \quad u * 8$
- $u \ll 5 - u \ll 3 \quad == \quad u * 24$
- Большинство машин сдвигает и складывает быстрее умножения
 - Компилятор создаёт соответствующий код автоматически

Компилированный код умножения

Функция Си

```
int mul12(int x)
{
    return x*12;
}
```

Арифметические операции в результате компиляции

```
leal (%eax,%eax,2), %eax
sall $2, %eax
```

Пояснение

```
t ← x+x*2
return t << 2;
```

- Компилятор Си автоматически создаёт код сдвига и сложения при умножении на константу

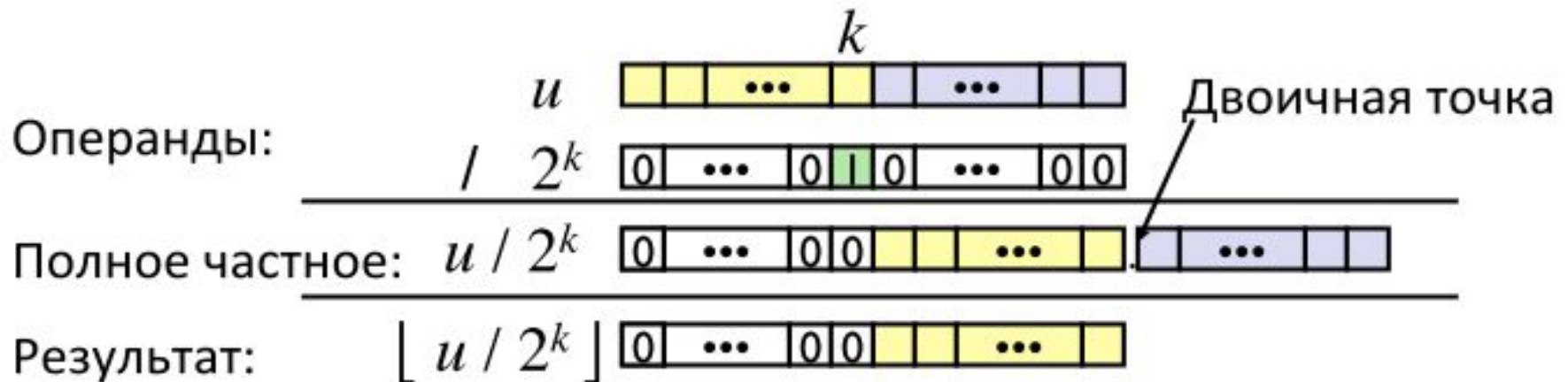
```
leal (%eax, %eax, 2), %eax
sall $2, %eax
```

```
lea eax, [eax+eax*2]
sal eax, 2
```

Деление сдвигом беззнакового на степень двойки

■ Частное от деления беззнакового на степень двойки

- $u \gg k$ даёт $\lfloor u / 2^k \rfloor$
- Используется логический сдвиг



	Полное частное	Результат	Шестн.	Двоичный
x	15213	15213	3В 6D	00111011 01101101
$x \gg 1$	7606.5	7606	1D B6	00011101 10110110
$x \gg 4$	950.8125	950	03 B6	00000011 10110110
$x \gg 8$	59.4257813	59	00 3В	00000000 00111011

Компилированный код деления беззнакового

Функция Си

```
unsigned udiv8(unsigned x)
{
    return x/8;
}
```

Арифметические операции
в результате компиляции

```
shrl $3, %eax
```

Пояснение

```
# Логический сдвиг
return x >> 3;
```

- Для беззнаковых используется логический сдвиг
- Для пользователей Явы
 - Логический сдвиг пишется как >>>

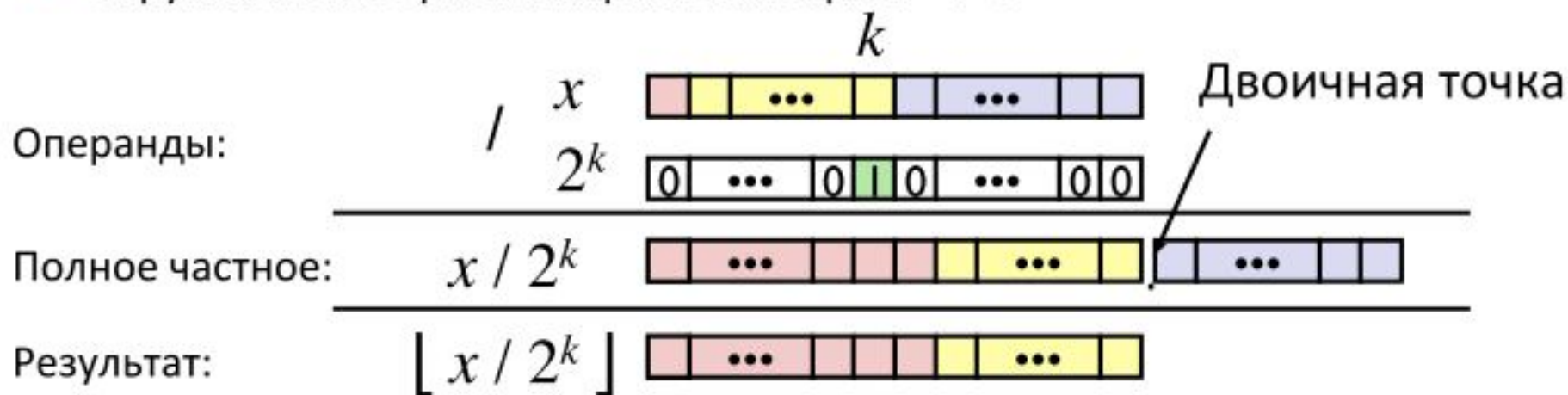
```
shrl $3, %eax
```

```
shr eax, 3
```

Деление сдвигом знакового на степень двойки

■ Частное от деления знакового на степень двойки

- $x \gg k$ даёт $\lfloor x / 2^k \rfloor$
- Использует арифметический сдвиг
- Округляет в неверном направлении при $x < 0$

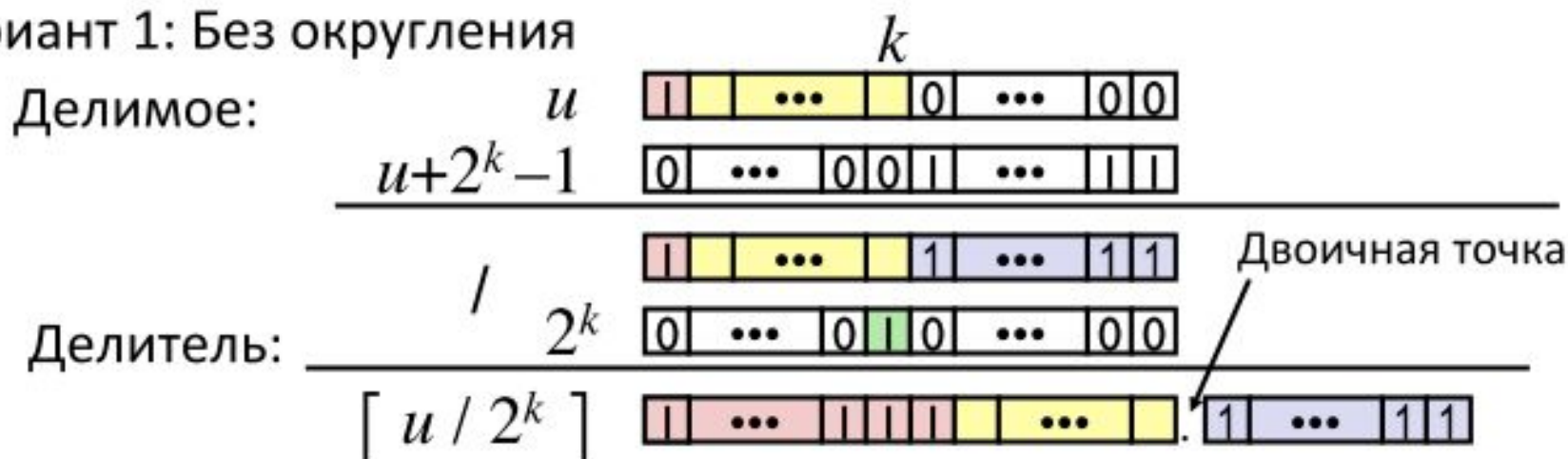


	Полное частное	Результат	Шестн.	Двоичный
y	-15213	-15213	C4 93	11000100 10010011
$y \gg 1$	-7606.5	-7607	E2 49	11100010 01001001
$y \gg 4$	-950.8125	-951	FC 49	11111100 01001001
$y \gg 8$	-59.4257813	-60	FF C4	11111111 11000100

Правильное деление на степень двойки(1)

- Частное от деления знакового на степень двойки
 - Необходимо $\lceil x / 2^k \rceil$ (Округление к 0)
 - Вычисляется как $\lfloor (x+2^k-1) / 2^k \rfloor$
 - В Си: $(x + (1 \ll k) - 1) \gg k$
 - Смещение делимого к 0

Вариант 1: Без округления

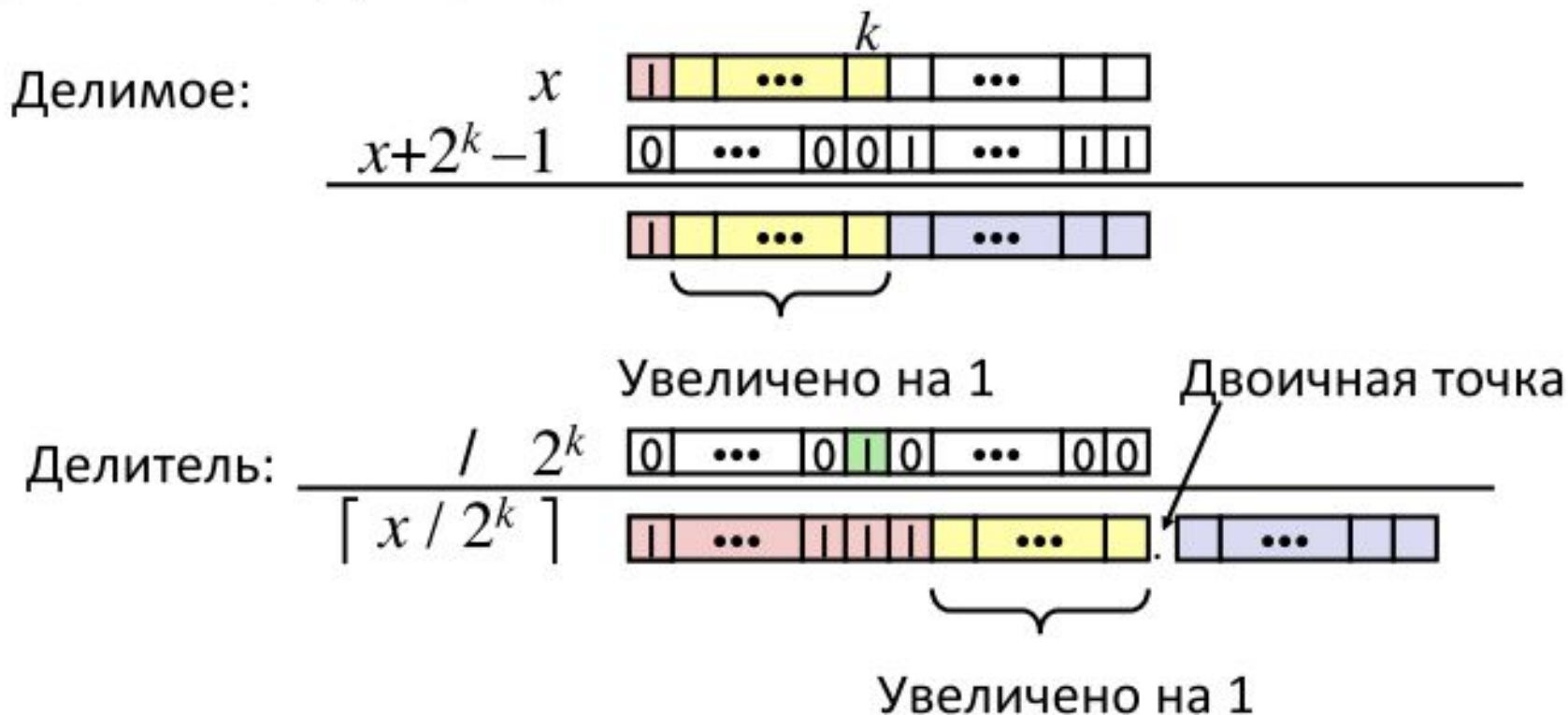


Смещение не имеет эффекта

- Пример
- -5 делим на 2 –Ю сдвиг вправо на 1
- 1011 1101 → -3, а не -2
- Добавляем смещение $1 \ll 2 - 1 = 2 - 1 = 1$
- $1011 + 1 = 1100$ сдвиг на 1 разряд вправо
получаем 1110 → -2

Правильное деление на степень двойки(2)

Вариант 2: округление



Смещение добавляет 1 к результату

Компилированный код деления знакового

Функция Си

```
int idiv8(int x)
{
    return x/8;
}
```

Арифметические операции в результате компиляции

```
testl %eax, %eax
js    L4
L3:
    sarl $3, %eax
    ret
L4:
    addl $7, %eax
    jmp  L3
```

```
test  eax, eax
      js   L4
L3:   sar  eax, 3
      ret
L4:   add  eax, 7
      jmp  L3
```

Пояснения

```
if x < 0
    x += 7;
# Арифметический сдвиг
return x >> 3;
```

- Арифметический сдвиг int
- Для пользователей Ява
 - Арифм. сдвиг пишется как >>

Арифметика: основные правила(1)

■ Сложение:

- (Без)знаковые: Нормальное сложение с отсечением, идентичные действия с битами
- Беззнаковые: сложение по модулю 2^w
 - Математическое сложение и возможное уменьшение на 2^w
- Знаковые: изменённое сложение по модулю 2^w (результат в допустимом диапазоне)
 - Математическое сложение и возможное добавление или уменьшение на 2^w

■ Умножение:

- (Без)знаковые: Нормальное умножение с отсечением, разные действия с битами
- Беззнаковые: умножение по модулю 2^w
- Знаковые: изменённое умножение по модулю 2^w (результат в допустимом диапазоне)

Арифметика: основные правила(2)

■ Сдвиг влево

- Беззнаковые: умножение на 2^k
- Всегда логический сдвиг

■ Сдвиг вправо

- Беззнаковые: логический сдвиг, деление на 2^k с округлением к 0
- Знаковые: арифметический сдвиг
 - Положительные: деление на 2^k с округлением к 0
 - Отрицательные: деление на 2^k с округлением от 0 используется смещение для исправления

Зачем использовать беззнаковые?

- *Не используйте только потому, что величина число неотрицательное*

- Легко сделать ошибку

```
unsigned i;  
for (i = cnt-2; i >= 0; i--)  
    a[i] += a[i+1];
```

- Может быть очень коварным

```
#define DELTA sizeof(int)  
int i;  
for (i = CNT; i-DELTA >= 0; i-= DELTA)  
    . . .
```

- *Используйте для представления множеств*
 - Логический сдвиг вправо, без расширения знака

Целочисленные головоломки Си

- Для каждого следующего выражения Си :
 - Либо объясните почему оно верно для любых значений аргументов
 - Либо - почему неверно

Инициализация

```
int x = function1();  
int y = funciton2();  
unsigned ux = x;  
unsigned uy = y;
```

- $x < 0 \Rightarrow ((x^2) < 0)$
- $ux \geq 0$
- $x \& 7 == 7 \Rightarrow (x \ll 30) < 0$
- $ux > -1$
- $x > y \Rightarrow -x < -y$
- $x * x \geq 0$
- $x > 0 \&\& y > 0 \Rightarrow x + y > 0$
- $x \geq 0 \Rightarrow -x \leq 0$
- $x \leq 0 \Rightarrow -x \geq 0$
- $(x|-x) \gg 31 == -1$
- $ux \gg 3 == ux/8$
- $x \gg 3 == x/8$
- $x \& (x-1) != 0$

1. $(x \geq 0) \parallel ((2 * x) < 0)$. Ложно. Пусть x будет равным -2147483648 ($TMin_{32}$). Тогда получим, что $2 * x$ будет равно 0.
2. $(x \& 7) \neq 7 \parallel (x \ll 30 < 0)$. Истинно. Если $(x \& 7) \neq 7$ стремится к нулю, тогда должен быть бит x_2 , равный единице. При сдвиге влево на 30 получится знаковый разряд (бит).
3. $(x * x) \geq 0$. Ложно. Когда x равно 65535 ($0xFFFF$), $x * x$ равно -131071 ($0xFFFFE0001$).
4. $x < 0 \parallel -x \leq 0$. Истинно. Если x — неотрицательная величина, тогда $-x$ — величина не положительная.
5. $x > 0 \parallel -x \geq 0$. Ложно. Пусть x равно -2147483648 ($TMin_{32}$). Тогда x и $-x$ отрицательные величины.

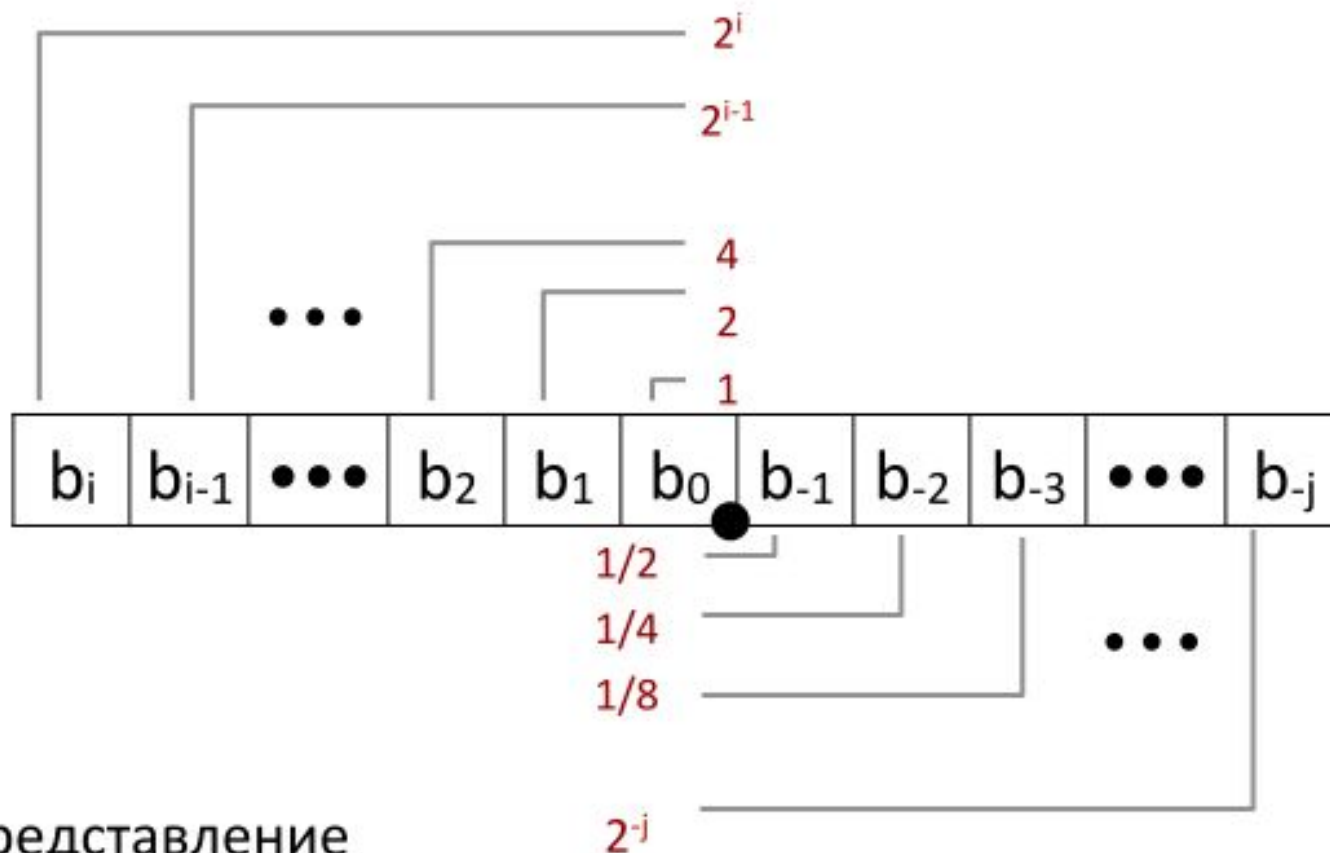
Плавающая точка

- Основы: Двоичные дроби
- Стандарт «плавающей точки» IEEE : Определение
- Примеры и свойства
- Округление, сложение, умножение
- Плавающая точка в Си
- Сводка

Двоичные дроби

- Что такое 1011.101_2 ?

Двоичные дроби



■ Представление

- Биты справа от «двоичной точки» представляют дробные степени 2
- Представление рациональных чисел:

$$\sum_{k=-j}^i b_k \times 2^k$$

Двоичные дроби: Примеры

- | ■ Значение | Представление |
|------------|---------------|
| 5 и 3/4 | 101.11_2 |
| 2 и 7/8 | 10.111_2 |
| 1 и 7/16 | 1.0111_2 |
| 23/32 | 0.10111_2 |
-
- Важно
 - Деление на 2 сдвигом вправо
 - Умножение на 2 сдвигом влево
 - Числа вида $0.111111\dots_2$ меньше 1.0
 - $1/2 + 1/4 + 1/8 + \dots + 1/2^i + \dots \rightarrow 1.0$
 - Обозначим $1.0 - \epsilon$

Представимые числа

■ Ограничения

- Двоичными дробями точно представимы только числа вида $n/2^k$
- Другие дроби представимы бесконечными периодическими «дробями»

■ Значение Представление

- $1/3$ $0.01[01]..._2$
- $1/5$ $0.0011[0011]..._2$
- $1/10$ $0.00011 [0011]..._2$

Плавающая точка IEEE

■ Стандарт IEEE 754

- Принят в 1985 как единый стандарт арифметики с плавающей точкой
 - До этого множество уникальных стандартов
- Поддерживается всеми основными CPU/FPU

■ В основе - вопросы вычислений

- Удачно стандартизованы
 - округления,
 - переполнения,
 - потеря значимости
- Сложно сделать быстрым в аппаратуре
 - При создании стандарта численные аналитики доминировали над разработчиками аппаратуры
- Есть реализации «с отклонениями»

■ Есть версии и альтернативы

Представление с плавающей точкой

■ Численная форма:

$$(-1)^s M 2^E$$

- Знаковый бит **s** определяет положительность/отрицательность
- Мантисса **M** - обычно дробь в интервале [1.0,2.0).
- Порядок **E** изменяет значение на степень двойки

■ Кодирование

- Наиболее значимый бит **S** – знаковый
- Поле **exp** кодирует **E**, но не совпадает с двоичным значением **E**
- Поле **frac** кодирует **M**, но не совпадает с двоичным значением **M**



Представление вещественных чисел в ЭВМ на базе процессора INTEL

- $1_2 \leq m < 10_2$

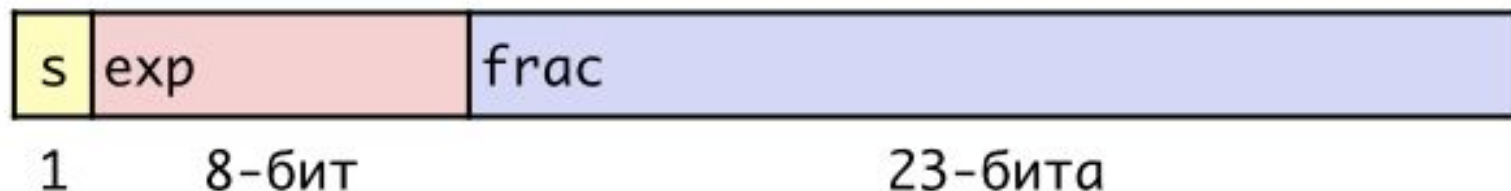
	Знак числа	Порядок	Смещение порядка	Мантисса
Короткое	31	30-23	127	22-0
Длинное	63	62-52	1023	51-0
Расширенное	79	78-64	16383	63-0

$$-1.0_{10} = -1.0e0_2 = \underbrace{1}_{\text{знак}} \underbrace{01111111}_{\text{порядок}} \underbrace{000\dots_2}_{\text{смещение}} = \text{BF800000}_{16}$$

$$5.25_{10} = 101.01_2 = +1.0101E+10_2 = \underbrace{0}_{\text{знак}} \underbrace{10000001}_{\text{порядок}} \underbrace{010100\dots_2}_{\text{мантисса}} = \text{40A80000}_{16}$$

Применяемые точности

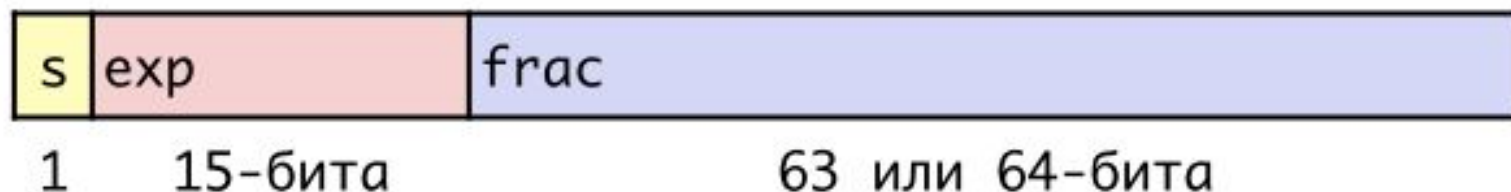
- Одинарная: 32 бита



- Двойная: 64 бита



- Расширенная: 80 бит (только для Intel)



Нормализованные значения

- Признак: $\text{exp} \neq 000\dots 0$ и $\text{exp} \neq 111\dots 1$
- Порядок кодируется со смещением : $E = \text{Exp} - \text{Bias}$
 - Exp : беззнаковое значение поля **exp**
 - $\text{Bias} = 2^{k-1} - 1$, где k - количество бит порядка – смещение
 - Одинарная точность: 127 (Exp : 1...254, E : -126...127)
 - Двойная точность: 1023 (Exp : 1...2046, E : -1022...1023)
- Код мантииссы подразумевает старшую 1: $M = 1.\text{xxx}\dots\text{x}_2$
 - $\text{xxx}\dots\text{x}$: биты поля **frac**
 - Минимальное значение $M = 1.0$, когда **frac** = 000...0
 - Минимальное значение $M = 2.0 - \epsilon$, когда **frac** = 111...1
 - Старший бит мантииссы не расходует ресурсы оборудования
 - Не подразумевается для расширенной точности 80-бит Intel !

Пример нормализованного кода

■ Значение: Float $F = 15213.0$;

$$\begin{aligned} \blacksquare 15213_{10} &= 11101101101101_2 \\ &= 1.1101101101101_2 \times 2^{13} \end{aligned}$$

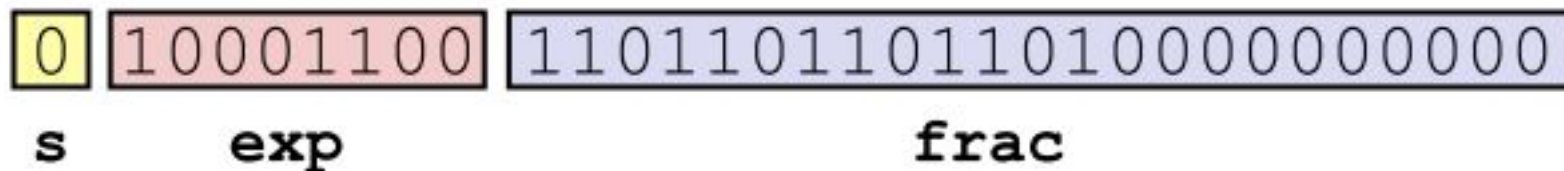
■ Мантисса

$$\begin{aligned} M &= 1.\underline{1101101101101}_2 \\ \text{frac} &= \underline{110110110110100000000000}_2 \end{aligned}$$

■ Порядок

$$\begin{aligned} E &= 13 \\ \text{Bias} &= 127 \\ \text{Exp} &= 140 = 10001100_2 \end{aligned}$$

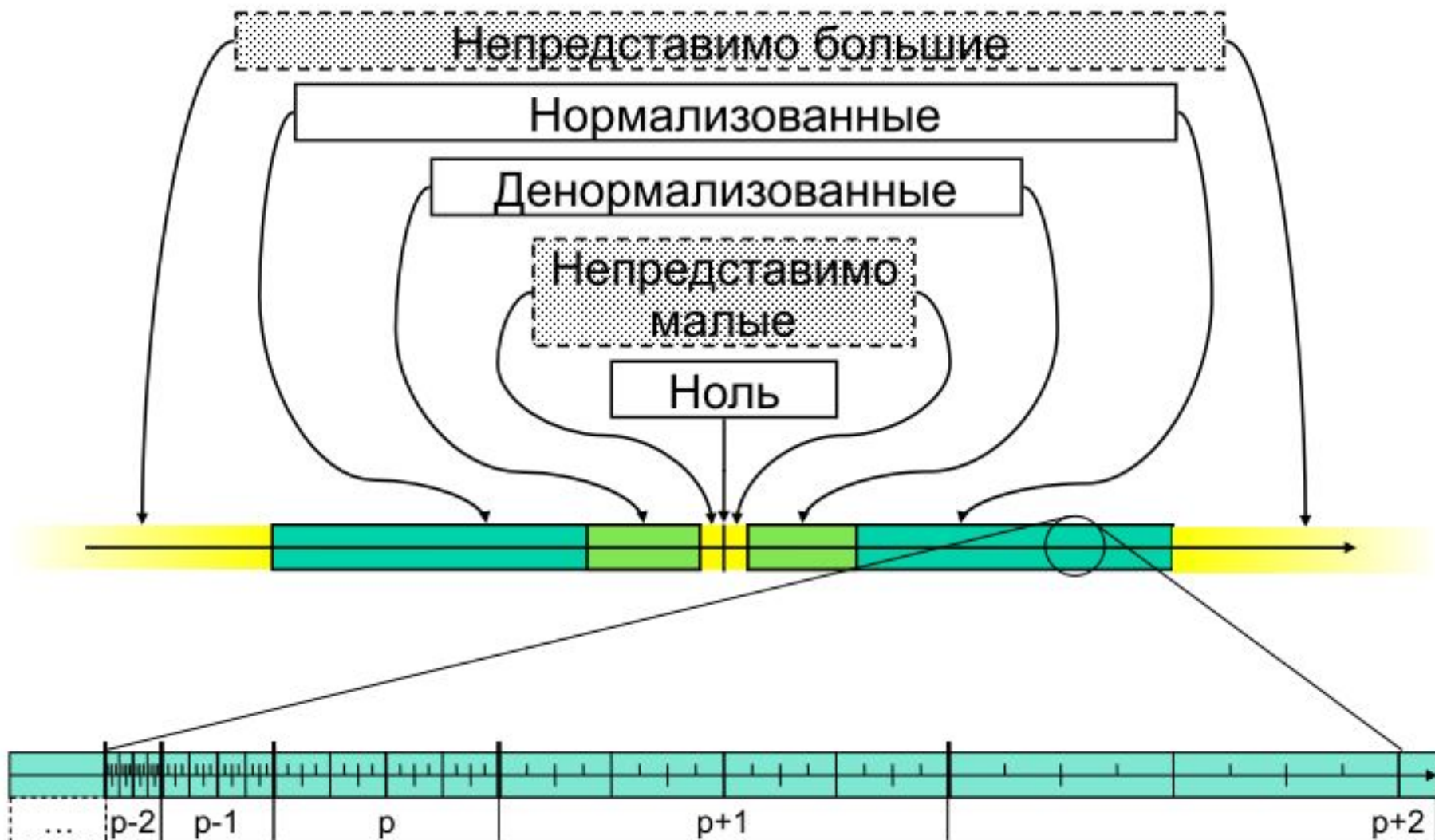
■ Результат:



Денормализованные значения

- Признак: $\text{exp} = 000\dots 0$
- Значение порядка: $E = 1 - \text{Bias}$ (вместо $E = 0 - \text{Bias}$)
- Код мантииссы подразумевает старший 0: $M = 0.\text{xxx}\dots\text{x}_2$
 - $\text{xxx}\dots\text{x}$: биты **frac**
- Варианты
 - $\text{exp} = 000\dots 0, \text{frac} = 000\dots 0$
 - Обозначает нулевое значение
 - Представление неоднозначно: $+0$ and -0 (почему?)
 - $\text{exp} = 000\dots 0, \text{frac} \neq 000\dots 0$
 - числа очень близкие к 0.0
 - чем меньше, тем хуже относительная погрешность
 - равноотстоящие

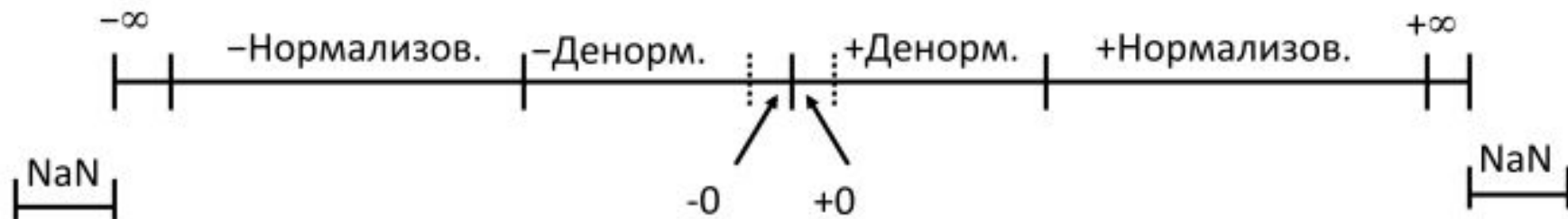
Что (не)представимо ?



Специальные коды

- Признак: $\text{exp} = 111\dots 1$
- Вариант 1: $\text{exp} = 111\dots 1, \text{frac} = 000\dots 0$
 - Обозначает значение ∞ (бесконечность)
 - Результат операции при переполнении
 - Есть оба варианта: отрицательная и положительная
 - E.g., $1.0/0.0 = -1.0/-0.0 = +\infty, 1.0/-0.0 = -\infty$
- Вариант 2: $\text{exp} = 111\dots 1, \text{frac} \neq 000\dots 0$
 - Не число, Not-a-Number (NaN)
 - Обозначает случаи когда невозможно определить численное значение
 - Примеры: $\text{sqrt}(-1), \infty - \infty, \infty \times 0$

Коды с плавающей точкой визуально



Укороченный пример с плавающей точкой



- 8-битное представление с плавающей точкой
 - Знаковый бит – самый значимый
 - следующие четыре бита - показатель со смещением $2^{4-1}-1 = 7$
 - последние три бита - код мантиссы
- Форма таже что в стандарте IEEE
 - нормализованные и денормализованные
 - обозначаются 0, NaN, бесконечность

Динамический диапазон

	s	exp	frac	E	Значение	
Денормализ. числа	0	0000	000	-6	0	
	0	0000	001	-6	$1/8 * 1/64 = 1/512$	Ближайшее к нулю
	0	0000	010	-6	$2/8 * 1/64 = 2/512$	
	...					
	0	0000	110	-6	$6/8 * 1/64 = 6/512$	
	0	0000	111	-6	$7/8 * 1/64 = 7/512$	Наибольшее денорм.
	0	0001	000	-6	$8/8 * 1/64 = 8/512$	Наименьшее нормализ.
Нормализов. числа	0	0001	001	-6	$9/8 * 1/64 = 9/512$	
	...					
	0	0110	110	-1	$14/8 * 1/2 = 14/16$	
	0	0110	111	-1	$15/8 * 1/2 = 15/16$	Ближайшее к 1 снизу
	0	0111	000	0	$8/8 * 1 = 1$	
	0	0111	001	0	$9/8 * 1 = 9/8$	Ближайшее к 1 сверху
	0	0111	010	0	$10/8 * 1 = 10/8$	
	...					
0	1110	110	7	$14/8 * 128 = 224$		
0	1110	111	7	$15/8 * 128 = 240$	Наибольшее нормализ.	
0	1111	000	n/a	inf		

Таблица 2.17. Пример гипотетического 8-битового формата

Описание	Битовое представление	e	E	f	M	V
Ноль	0 0000 000	0	-6	0	0	0
Наименьшее положительное	0 0000 001	0	-6	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{512}$
	0 0000 010	0	-6	$\frac{2}{8}$	$\frac{2}{8}$	$\frac{2}{512}$
	0 0000 011	0	-6	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{3}{512}$
	...					
	0 0000 110	0	-6	$\frac{6}{8}$	$\frac{6}{8}$	$\frac{6}{512}$
Наибольшее ненормализованное	0 0000 111	0	-6	$\frac{7}{8}$	$\frac{7}{8}$	$\frac{7}{512}$
Наименьшее нормализованное	0 0001 000	1	-6	0	$\frac{7}{8}$	$\frac{8}{512}$
	0 0001 001	1	-6	$\frac{1}{8}$	$\frac{9}{8}$	$\frac{9}{512}$
	...					
	0 0110 110	6	-1	$\frac{6}{8}$	$\frac{14}{8}$	$\frac{14}{16}$
	0 0110 111	6	-1	$\frac{7}{8}$	$\frac{15}{8}$	$\frac{15}{16}$
Единица	0 0111 000	7	0	0	$\frac{8}{8}$	1
	0 0111 001	7	0	$\frac{1}{8}$	$\frac{9}{8}$	$\frac{9}{8}$
	0 0111 010	7	0	$\frac{2}{8}$	$\frac{10}{8}$	$\frac{9}{8}$
	...					
	0 1110 110	14	7	$\frac{6}{8}$	$\frac{14}{8}$	224
Наибольшее нормализованное	0 1110 111	14	7	$\frac{7}{8}$	$\frac{15}{8}$	240
Бесконечность	0 1111 000	-	-	-	-	$+\infty$

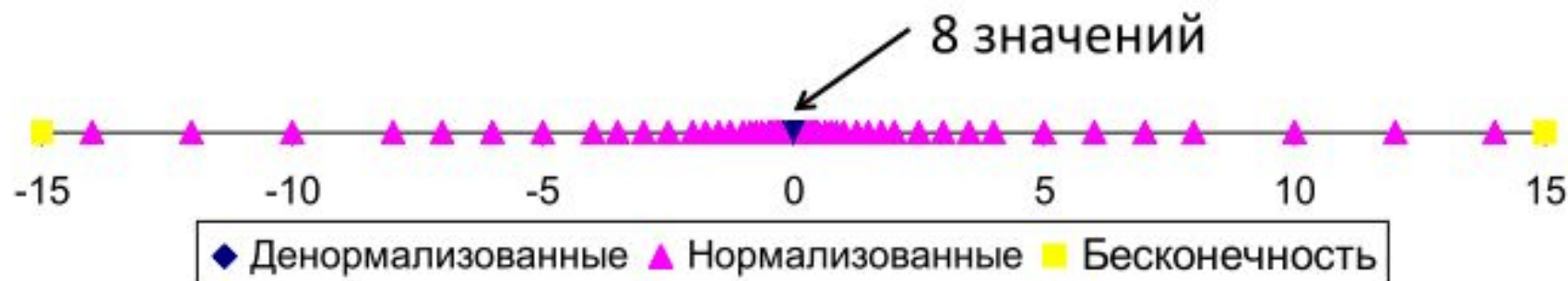
Распределение значений

■ Шестибитный формат подобный IEEE

- 3 бита порядка
- 2 дробных бита
- смещение: $2^{3-1}-1 = 3$



■ Сгущение вокруг 0



6 битов						
к=3 порядок						
п=2 мантисса						
смещение = 3						
Описание	битовое представление	e	E	f	M	V
ноль	0 000 00	0	-2	0	0	0
наименьшее положительное	0 000 01	0	-2	1/4	1/4	1/16
	0 000 10	0	-2	2/4	2/4	2/16
.....						
наибольшее ненормализованное	0 000 11	0	-2	3/4	3/4	3/16
наименьшее нормализованное	0 001 00	1	-2	0	4/4	4/16
	0 001 01	1	-2	1/4	5/4	5/16
.....						
	0 010 10	2	-1	2/4	6/4	6/8
	0 010 11	2	-1	3/4	7/4	7/8
единица	0 011 00	3	0	0	4/4	1
	0 011 01	3	0	1/4	5/4	5/4
.....						
	0 110 10	6	3	2/4	6/4	12
наибольшее нормализованное	0 110 11	6	3	3/4	7/4	14
бесконечность	0 111 00	--	--	--	--	+--
	0 111 01	--	--	--	--	NaN
	0 111 10	--	--	--	--	NaN
	0 111 11	--	--	--	--	NaN

• Представление вещественных чисел в ЭВМ на базе процессора INTEL

- Знаковые нули
- Денормализованные конечные числа
- Нормализованные конечные числа
- Знаковые бесконечности
- NaN (нечисла)
- Неопределимые числа

Описание	exp	frac	Одинарная точность		Удвоенная точность	
			Значение	Десятичное	Значение	Десятичное
Ноль	00...00	00...00	0	0.0	0	0.0
Наименьшее ненормализованное	00...00	0...01	$2^{-23} \times 2^{-126}$	1.4×10^{-45}	$2^{-52} \times 2^{-1022}$	4.9×10^{-324}
Наибольшее ненормализованное	00...00	1...11	$(1-\epsilon) \times 2^{-126}$	1.2×10^{-38}	$(1-\epsilon) \times 2^{-1022}$	2.2×10^{-308}
Наименьшее нормализованное	00...01	0...00	1×2^{-126}	1.2×10^{-38}	1×2^{-1022}	2.2×10^{-308}
Единица	01...11	0...00	1×2^0	1.0	1×2^0	1.0
Наибольшее нормализованное	11...10	1...11	$(2-\epsilon) \times 2^{127}$	3.4×10^{38}	$(2-\epsilon) \times 2^{1023}$	1.8×10^{308}

Интересные числа

{одинарные, двойны

Описание	<i>exp</i>	<i>frac</i>	Численное значение ^{e}}
■ Ноль	00...00	00...00	0.0
■ Наименьшее денормализ.	00...00	00...01	$2^{-\{23,52\}} \times 2^{-\{126,1022\}}$
▪ Одинарное $\approx 1.4 \times 10^{-45}$			
▪ Двойное $\approx 4.9 \times 10^{-324}$			
■ Наибольшее денормализ.	00...00	11...11	$(1.0 - \epsilon) \times 2^{-\{126,1022\}}$
▪ Одинарное $\approx 1.18 \times 10^{-38}$			
▪ Двойное $\approx 2.2 \times 10^{-308}$			
■ Наименьшее нормализ.	00...01	00...00	$1.0 \times 2^{-\{126,1022\}}$
▪ Ближайшее большее наибольшего денормализованного			
■ Единица	01...11	00...00	1.0
■ Наибольшее нормализ.	11...10	11...11	$(2.0 - \epsilon) \times 2^{\{127,1023\}}$
▪ Одинарное $\approx 3.4 \times 10^{38}$			
▪ Двойное $\approx 1.8 \times 10^{308}$			

Сравнение целых чисел и чисел с плавающей точкой

- 12345 и 12,345
- 0x3039
00000000000000000000000011000000111001
- $1.1000000111001 \times 2^{13}$ $13+127=140$ 10001100
- 0x4640E400
- 01000110010000001110010000000000

Операции с плав. точкой: Основная идея

■ $x +_f y = \text{Round}(x + y)$

■ $x \times_f y = \text{Round}(x \times y)$

■ Основная идея

- Сначала вычислить полный результат
- Затем втиснуть его в желаемую точность
 - Возможно с переполнением, если порядок слишком велик
 - Возможно с округлением, чтобы уместиться во **frac**

Округление

■ Режим округления	1.40	1.60	1.50	2.50	-1.50
■ К нулю	1	1	1	2	-1
■ Вниз ($-\infty$)	1	1	1	2	-2
■ Вверх ($+\infty$)	2	2	2	3	-1
■ К чётному (по умолч.)	1	2	2	2	-2

- В чём преимущества режимов?

Подробнее об округлении к чётному

- Режим округления по умолчанию
 - Любой другой режим требует обращения к ассемблеру
 - Все остальные режимы дают статистический сдвиг
 - Сумма округлённых устойчиво меньше или устойчиво больше округлённой суммы

- Применяя к десятичным разрядам

- Если точно посередине между двумя возможными значениями
 - Округляется так, что младшая цифра чётная
- Например, округление к ближайшей сотой

1.2349999	1.23	(Меньше половины - вниз)
-----------	------	--------------------------

1.2350001	1.24	(Больше половины - вверх)
-----------	------	---------------------------

1.2350000	1.24	(Половина - вверх)
-----------	------	--------------------

1.2450000	1.24	(Половина - вниз)
-----------	------	-------------------

Округление двоичных чисел

■ Двоичные дроби

- “Чётные” когда наименьший бит = 0
- “Половина” когда биты справа от позиции округления = $100..._2$

■ Примеры

- Округление к $1/4$ (2 бита справа от двоичной точки)

Значение	Двоичное	Двоичное Округление	Действие	Округлённое Значение
$2 \frac{3}{32}$	10.00011_2	10.00_2	($<1/2$ —вниз)	2
$2 \frac{3}{16}$	10.00110_2	10.01_2	($>1/2$ —вверх)	$2 \frac{1}{4}$
$2 \frac{7}{8}$	10.11100_2	11.00_2	($1/2$ —вверх)	3
$2 \frac{5}{8}$	10.10100_2	10.10_2	($1/2$ —вниз)	$2 \frac{1}{2}$

Умножение с плавающей точкой

- $(-1)^{s1} M1 2^{E1} \times (-1)^{s2} M2 2^{E2}$
- Полный результат: $(-1)^s M 2^E$
 - Знак s : $s1 \wedge s2$
 - Мантисса M : $M1 \times M2$
 - Порядок E : $E1 + E2$
- Нормализация
 - Если $M \geq 2$, сдвинуть M вправо, увеличив E
 - Если E за рамками, то переполнение
 - Округлить M чтобы уложиться в точность **frac**
- Реализация
 - Трудоёмкая часть - перемножение мантисс

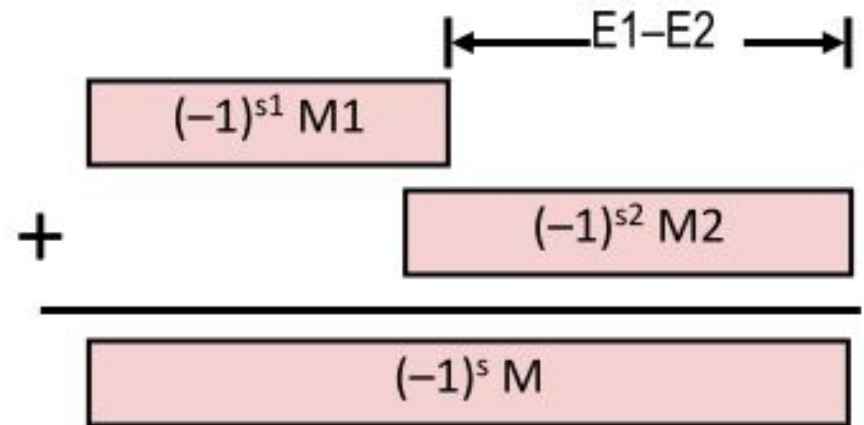
Сложение с плавающей точкой

$$\blacksquare (-1)^{s1} M1 2^{E1} + (-1)^{s2} M2 2^{E2}$$

- Предположим $E1 > E2$

$$\blacksquare \text{Полный результат: } (-1)^s M 2^E$$

- Знак s , мантисса M :
 - Результат знакового выравнивания и сложения
- Порядок E : $E1$



■ Нормализация

- Если $M \geq 2$, сдвинуть M вправо, увеличив E
- Если $M < 1$, сдвинуть M влево на k разрядов, уменьшив E на k
- Если E за рамками, то переполнение
- Округлить M чтобы уложиться в точность **frac**

```
x = a + b + c;
```

```
y = b + c + d;
```

```
t = b + c;
```

```
x = a + t;
```

```
y = t + d;
```

Математические свойства сложения с ПТ

- Сравним со свойствами абелевой группы
 - Замкнута относительно сложения ? Да
 - Может выдавать бесконечность или нечисло
 - Коммутативно ? Да
 - Ассоциативно ? Нет
 - Переполнение и округление!
 - Ноль – нейтральный элемент ? Да
 - Каждый элемент имеет обратный ? Почти
 - Кроме бесконечности и NaN
- Монотонность
 - $a \geq b \Rightarrow a+c \geq b+c$? Почти
 - Кроме бесконечности и NaN

Математические свойства умножения с ПТ

- Сравним со свойствами коммутативного кольца
 - Замкнут относительно умножения ? Да
 - Может выдавать бесконечность или нечисло
 - Умножение коммутативно ? Да
 - Умножение ассоциативно ? Нет
 - Переполнение и округление!
 - Единица нейтральный элемент умножения ? Да
 - Умножение дистрибутивно относительно сложения ? Нет
 - Переполнение и округление!

- Монотонность
 - $a \geq b \ \& \ c \geq 0 \Rightarrow a * c \geq b * c$? Почти
 - Кроме бесконечностей и нечисел

Плавающая точка в Си

- Си гарантирует как минимум два уровня точности
 - `float` одиночная
 - `double` двойная
- Преобразования
 - Преобразование между `int`, `float`, и `double` меняет набор битов
 - `double/float` → `int`
 - Отсекает дробную часть
 - Подобно округлению к нулю
 - Не определено для запредельных или NaN: обычно TMin
 - `int` → `double`
 - Точное преобразование, т.к. `int` имеет разрядность ≤ 53 бит
 - `int` → `float`
 - Округляется в соответствии в режимом
- Промежуточные вычисления в расширенном формате (80 бит)

Головоломки плавающей точки

- Для каждого следующего выражения Си :
 - Либо объясните почему оно верно для любых значений аргументов
 - Либо - почему неверно

```
int x = ...;  
float f = ...;  
double d = ...;
```

Принять, что
ни d ни f не есть NaN

- $x == (\text{int})(\text{float}) x$
- $x == (\text{int})(\text{double}) x$
- $f == (\text{float})(\text{double}) f$
- $d == (\text{float}) d$
- $f == -(-f)$;
- $2/3 == 2/3.0$
- $d < 0.0 \Rightarrow ((d*2) < 0.0)$
- $d > f \Rightarrow -f > -d$
- $d * d \geq 0.0$
- $(d+f)-d == f$