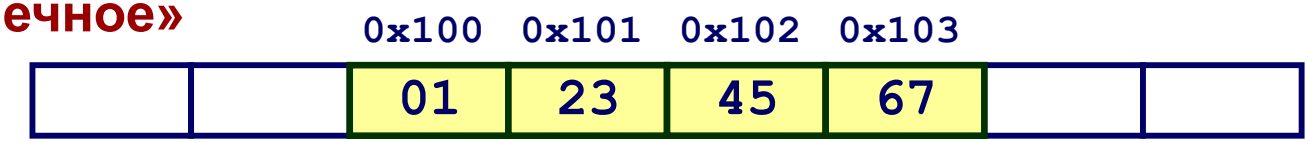
Порядок байт в слове

- **В каком порядке располагаются в памяти байты многобайтового слова?**
- **Соглашения**
  - «Тупоконечники»: Sun, PPC Mac, Internet
    - Наименее значимый байт имеет наибольший адрес
  - «Остроконечники»: x86
    - Наименее значимый байт имеет наименьший адрес

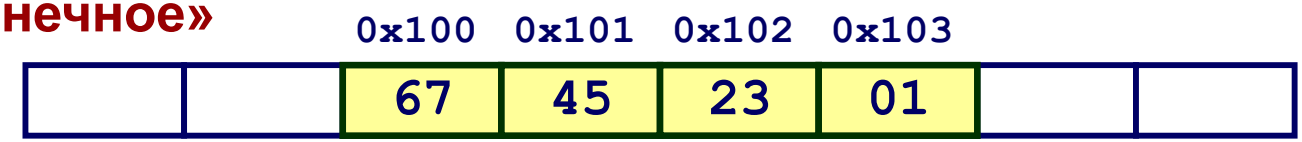
Примеры упорядочения байт

- **«Тупоконечное»**
  - Наименее значимый байт имеет наибольший адрес
- **«Остроконечное»**
  - Наименее значимый байт имеет наименьший адрес
- **Пример**
  - Переменная x
    - имеет 4-байтовое представление 0x01234567
    - Расположена по адресу &x - 0x100

«Тупоконечное»



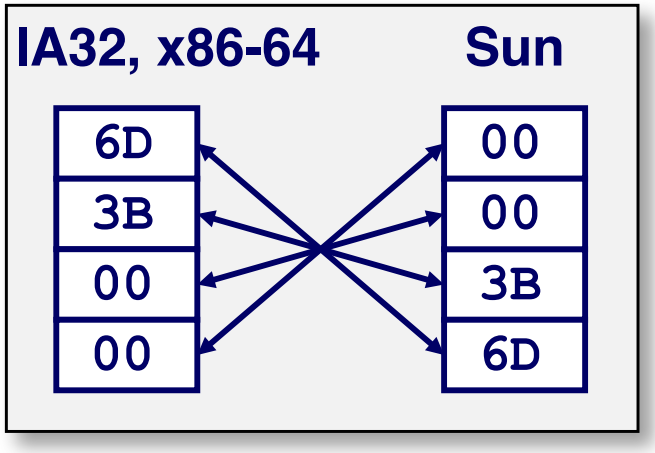
«Остроконечное»



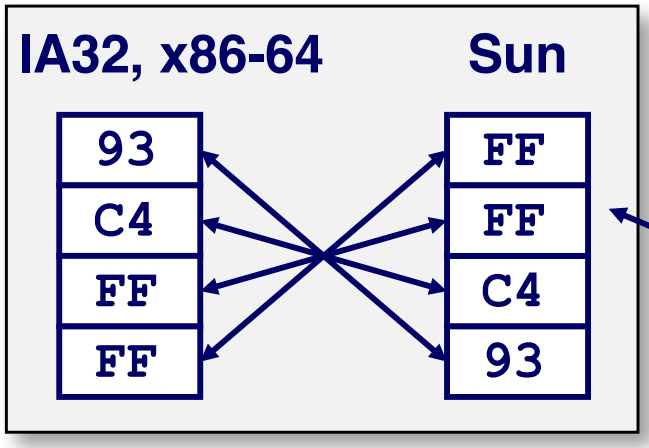
Представление Целочисленное

Десятичное:	15213
Двоичное:	0011 1011 0110 1101
Шестнадцатиричное:	3 B 6 D

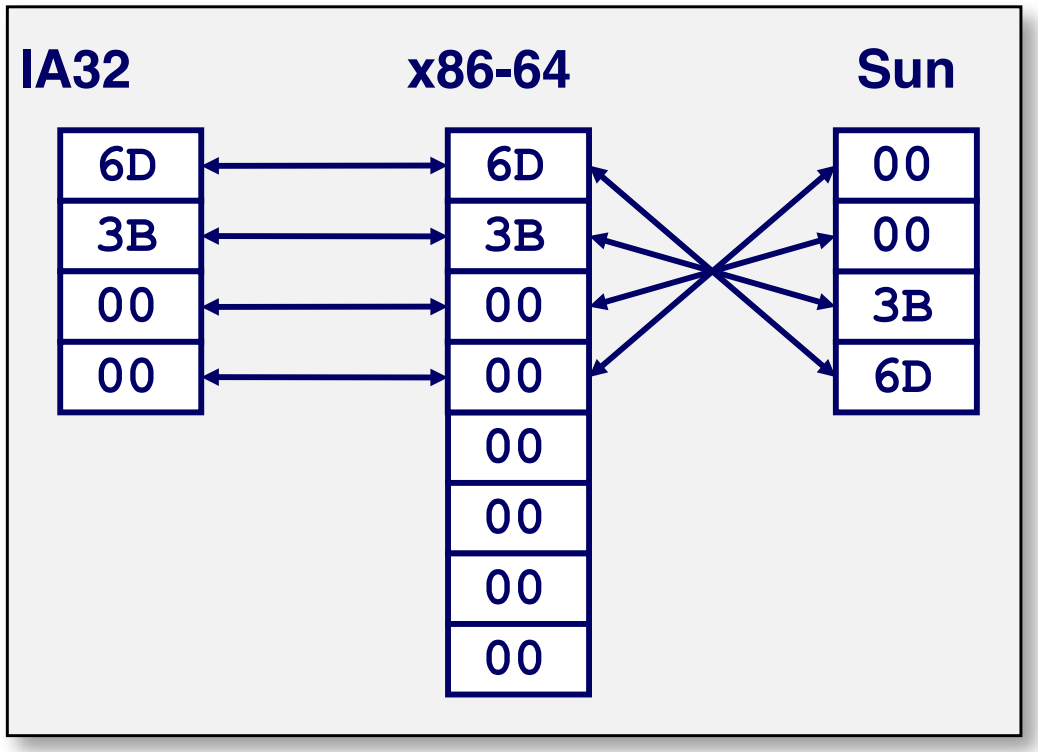
```
int A = 15213;
```




```
int B = -15213;
```




```
long int C = 15213;
```



Два представления в дополнительном коде (пояснения последуют)

Изучение представления данных

- **Вывод байтового представления данных**
  - Представление указателя как массива **`unsigned char *`**

```
typedef unsigned char *pointer;

void show_bytes(pointer start, int len){
  int i;
  for (i = 0; i < len; i++)
    printf(”%p\t0x%.2x\n",start+i, start[i]);
  printf("\n");
}
```

Спецификации преобразования:

%p: Вывод указателя

%x: Вывод шестнадцатиричного

Пример исполнения show_bytes

```
int a = 15213;
printf("int a = 15213;\n");
show_bytes((pointer) &a, sizeof(int));
```

Результат (x86, Linux):

```
int a = 15213;
0x11ffffcb8 0x6d
0x11ffffcb9 0x3b
0x11ffffcba 0x00
0x11ffffcbb 0x00
```

Плавающая точка

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

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

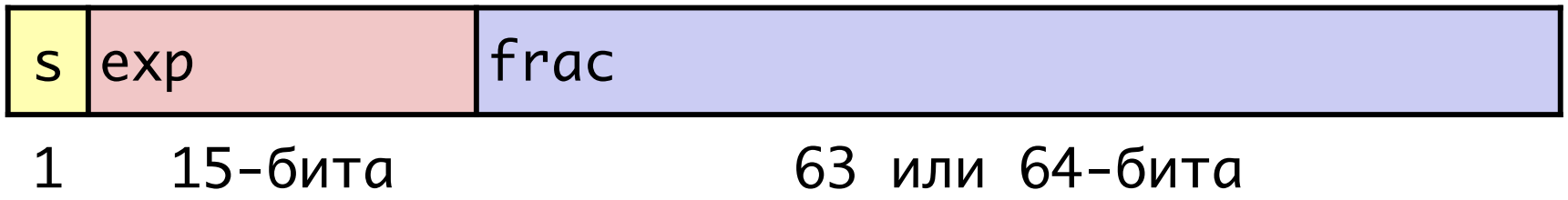
- **Кодирование**
  - Наиболее значимый бит **s** – знаковый
  - Поле **exp** кодирует ***E***, но не совпадает с двоичным значением ***E***
  - Поле **frac** кодирует ***M,*** но не совпадает с двоичным значением ***M***

Применяемые точности

- **Одинарная: 32 бита**

- **Двойная: 64 бита**

- **Расширенная: 80 бит (только для Intel)**



Нормализованные значения

- **Признак: exp ≠ 000…0 и exp ≠ 111…1**

- **Порядок кодируется со смещением : *E* = *Exp* – *Bias***
  - *Exp*: беззнаковое значение поля **exp**
  - *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 мантиссы подразумевает старшую 1: *M* = 1.xxx…x$_2$**
  - xxx…x: биты поля **frac**
  - Минимальное значение *M* = 1.0 , когда **frac** = 000…0
  - Минимальное значение *M* = 2.0 – ε , когда **frac** = 111…1
  - Старший бит мантиссы не расходует ресурсы оборудования
  - Не подразумевается для расширенной точности 80-бит Intel !

Пример нормализованного кода

- **Значение:** `Float F = 15213.0;`
  - $15213_{10} = 11101101101101_2$
    $= 1.1101101101101_2 \times 2^{13}$

- **Мантисса**

  $M = 1.\underline{1101101101101}_2$

  $\texttt{frac} = \underline{1101101101101}0000000000_2$

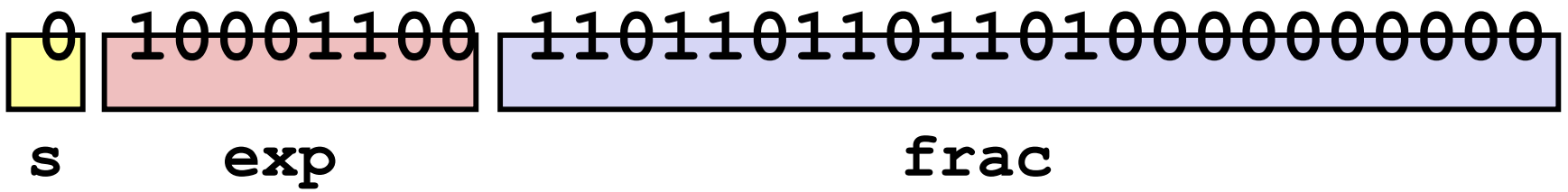
- **Порядок**

  $E = 13$

  $Bias = 127$

  $Exp = 140 = 10001100_2$

- **Результат:**



Денормализованные значения

- **Признак: exp = 000…0**

- **Значение порядка: *E* = 1–*Bias* (вместо *E* = 0 – *Bias*)**
- **Код мантиссы подразумевает старший 0: *M* = 0.xxx…x_2**
  - **xxx**…**x**: биты **frac**
- **Варианты**
  - **exp** = 000…0, **frac** = 000…0
    - Обозначает нулевое значение
    - Представление неоднозначно: +0 and –0 (почему?)
  - **exp** = 000…0, **frac** ≠ 000…0
    - числа очень близкие к 0.0
    - чем меньше, тем хуже относительная погрешность
    - равноотстоящие

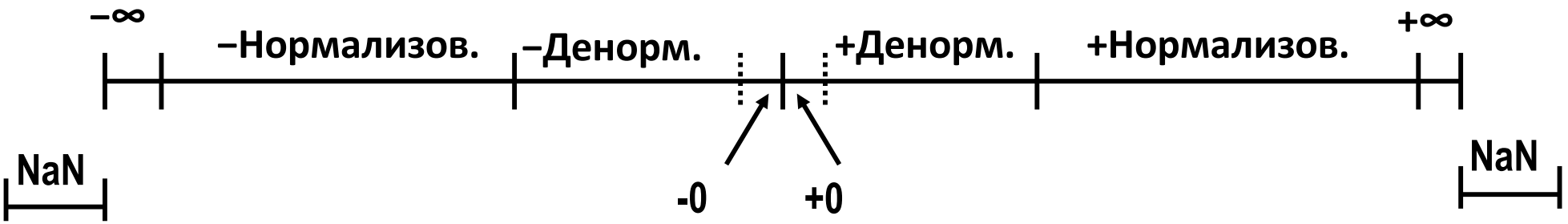
Специальные коды

- **Признак: `exp` = 111…1**

- **Вариант 1: `exp` = 111…1, `frac` = 000…0**
  - Обозначает значение ∞ (бесконечность)
  - Результат операции при переполнении
  - Есть оба варианта: отрицательная и положительная
  - E.g., 1.0/0.0 = −1.0/−0.0 = +∞, 1.0/−0.0 = −∞

- **Вариант 2: `exp` = 111…1, `frac` ≠ 000…0**
  - Не число, Not-a-Number (NaN)
  - Обозначает случаи когда невозможно определить численное значение
  - Примеры: sqrt(–1), ∞ − ∞, ∞ × 0

Коды с плавающей точкой визуально



(не)Представимые числа (2)