Семинар 09.04.2021

Перевод дробных чисел из десятичной с/с в двоичную

Пример: перевести десятичную дробь 0,1875 в двоичную, восьмеричную и шестнадцатеричную системы.

двоичная

0,	1875
	x 2
0,	3750
	x 2
0,	7500
	x 2
1,	5000
	x 2
1,	0000

восьмеричная

0,	1875
	x 8
1,	5000
	x 8
4,	0000

шестнадцатеричная

0,	1875
	x 16
1,	1250
+1,	875
3,	0000

Отсюда:

$0,1875_{10}=0,0011_{2}=0,14_{8}=0,3_{16}$

Перевод дробных чисел из двоичной с/с в десятичную

$$101{,}011_2 = 1\cdot 2^2 + 1\cdot 2^0 + 1\cdot 2^{-2} + 1\cdot 2^{-3}$$
$$= 4 + 1 + 0{,}25 + 0{,}125 = 5{,}375$$

Преставление чисел с плавающей точкой по стандарту IEEE 754

- Численное представление

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

  - Знаковый бит s определяет, является число положительным или отрицательным
  - Мантисса M – дробное число в полуинтервале [1.0,2.0).
  - Порядок E определяет степень 2 в третьем множителе

- Кодировка
  - Наибольший значащий бит s – знаковый бит s
  - Поле exp кодирует порядок E
  - Поле frac кодирует мантиссу M

Нормализованные числа

- Значение: `float f = 15213.0;`

  $15213_{10} = 11101101101101_2$

  $= 1.1101101101101_2 \times 2^{13}$

- Мантисса

  M = 1.1101101101101_2

  frac = $11011011011010000000000_2$

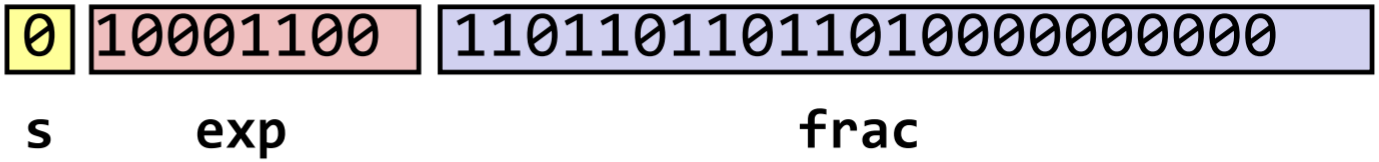
- Порядок

  E = 13

  Смещение = 127

  Exp = E + Смещение = 140 = 10001100_2

- Итого:



Денормализованные числа

- Условие: exp = 000…0
- Значение порядка: `E = –Смещение + 1` (вместо `E = 0 – Смещение`)
- Мантисса кодируется с ведущим `0`: `M = 0.xxx…`x_2
  - xxx…x: биты поля `frac`
- Примеры
  - exp = 000…0, frac = 000…0
    - Представляет число ноль
    - Различные кодировки для +0 и –0
  - exp = 000…0, frac ≠ 000…0
    - Кодируются числа близкие к 0.0
    - Распределены по числовой прямой с равным шагом

Особые числа

- Условие: exp = 111…1
- Пример: exp = 111…1, frac = 000…0
  - Представляет бесконечно большое число ∞
    (как положительное, так и отрицательное)
  - Требуются для операций в которых может произойти переполнение

    `1.0/0.0 = −1.0/−0.0 = +∞`

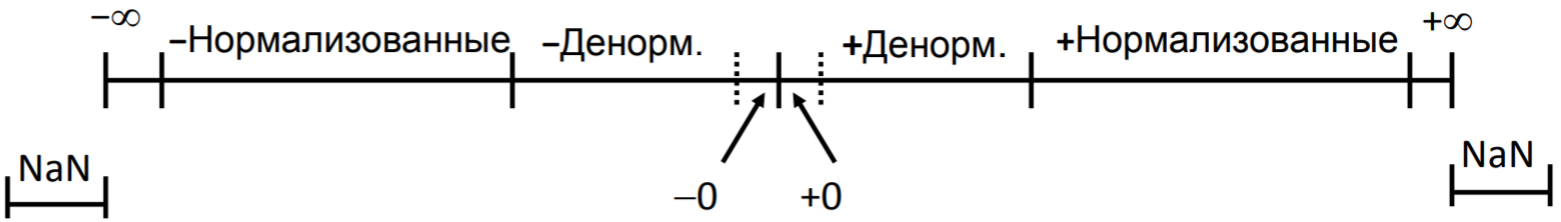
    `1.0/−0.0 = −∞`
- Пример: exp = 111…1, frac ≠ 000…0
  - Not-a-Number (NaN)
  - Используется в ситуациях, когда значение операции не определено

    `sqrt(−1)`

    `∞ − ∞`

    `∞ × 0`

Распределение чисел



Округление чисел

- Двоичные дробные числа
  - “Четные” числа у которых младший значащий бит 0
  - “Середина” – когда биты справа от позиции к которой происходит округление = $100\ldots_2$

- Примеры
  - Округление до ближайшей 1/4 (2 бита справа от бинарной точки)

Число	Двоичное	Окр.	Действие	Окр. число
2 3/32	10.00011_2	10.00_2	(<1/2—down)	2
2 3/16	10.00110_2	10.01_2	(>1/2—up)	2 1/4
2 7/8	10.11100_2	11.00_2	(1/2—up)	3
2 5/8	10.10100_2	10.10_2	(1/2—down)	2 1/2

Сложение чисел

- Выполняются ли свойства Абелевых групп
  - Замкнутость? — Да
    - Результатом может быть бесконечность или NaN
  - Коммутативность? — Да
  - Ассоциативность? — Нет
    - Переполнения и изменение результата при округлении
  - `0.0` – нейтральный элемент? — Да
  - Каждый элемент имеет обратный — Почти всегда
    - За исключением бесконечности и NaN

Умножение чисел

- Выполняются ли свойства коммутативных колец
  - Замкнуто ли относительно умножения? Да
    - Результат может быть бесконечность или NaN
  - Умножение коммутативно? Да
  - Умножение ассоциативно? Нет
    - Возможность переполнения, неточности округления
  - 1.0 – мультипликативная единица? Да
  - Умножение дистрибутивно над сложением? Нет
    - Возможность переполнения, неточности округления

Пример

Используется 10-битный формат, удовлетворяющий требованиям стандарта IEEE 754: знаковый бит, 4 бита – порядок, 5 битов – мантисса. Требуется представить в данном формате числа -⅕ и 105.

Решение

$-\frac{1}{5} = (-1)^1 \times \frac{8}{5} \times 2^{-3}$, таким образом `S = 1`, M = $\frac{8}{5}$, `E = -3`. Поскольку для кодировки порядка выделено 4 бита $bias = 2^{4-1} - 1 = 7$, а `EXP = -3 + bias|7 = 4 =` 0100_2.

Переводим $\frac{8}{5}$ в периодическую двоичную дробь разложением по степеням двойки.

$$\frac{8}{5} = 1 \times 2^0 + \frac{6}{5} \times 2^{-1} = 1 \times 2^0 + 1 \times 2^{-1} + \frac{2}{5} \times 2^{-2} = 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + \frac{4}{5} \times 2^{-3} =$$

$$= 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + \frac{8}{5} \times 2^{-4} = 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + \frac{6}{5} \times 2^{-5} = \ldots$$

Собираем коэффициенты перед степенями двоек и получаем периодическую дробь: $\frac{8}{5} = 1.100(1100)_2$. Т. к. ведущая 1 в поле FRAC кодироваться не будет, записаны будут следующие, выделенные серым цветом, биты `1.1001100(1100)`$_2$. Не вмещающаяся в поле последовательность битов `00(1100)`$_2$ меньше `1(0)`$_2$ поэтому округление выполняется к меньшему число, т. е. к `1.10011`$_2$.

Таким образом, искомая кодировка для -⅕ имеет вид `1_0100_10011`.

Пример

Представляем 105 в виде суммы степеней двойки и переводим в двоичное представление $105_{10} = 64 + 32 + 8 + 1 = 1101001_2$. В виде трех множителей число представится как $105_{10} = (-1)^0 \times 1.101001_2 \times 2^6$. EXP = 6 + $bias|_7$ = 13 = 1101_2.

В записи мантиссы последняя единица не помещается в поле FRAC, что означает необходимость искать ближайшее четное число. Выпишем (в порядке убывания), какие представимые заданной кодировкой числа окружают мантиссу.

1.101010_2

M = 1.101001_2

1.101000_2

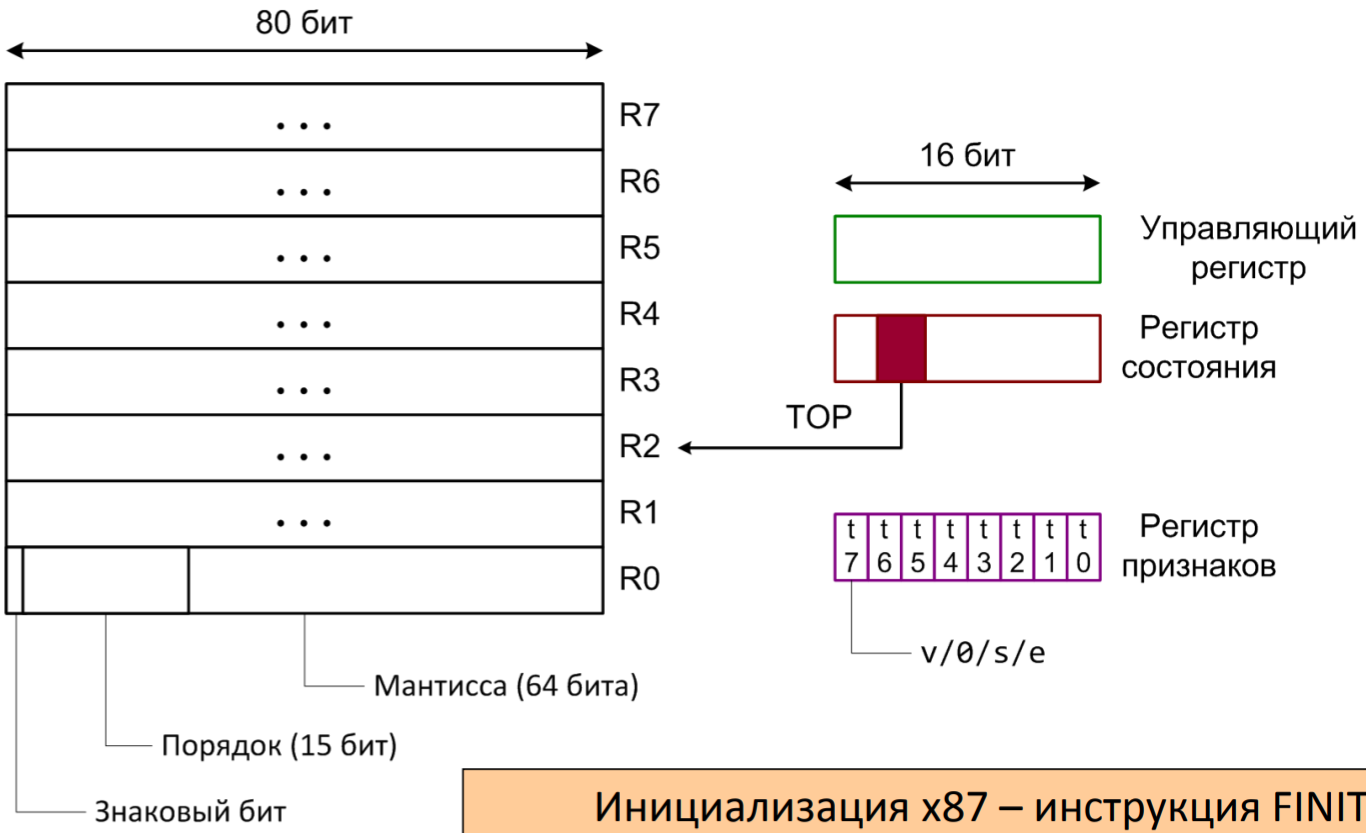
Как видно, после приведения к предоставленным размерам большее число оказывается оканчивающимся на 1, а меньшее – на 0. Таким образом, округляем мантиссу к 1.101000_2 и заносим последовательность выделенных серым цветом битов в поле FRAC.

Собрав все поля, получаем для числа 105 искомую кодировку $0\_1101\_10100_2$.

Задача

Используется 9-битный формат, удовлетворяющий требованиям стандарта IEEE 754: знаковый бит, 4 бита – порядок, 4 бита - мантисса. Требуется представить в данном формате числа ⅝ и -89.

Математический сопроцессор x87



Инициализация x87 – инструкция FINIT

```
CW = 0x037F   SW = 0x0000   Tag = 0xFFFF
```

Вещественные константы в NASM

```
db -0.2                          ; «Четверть»
dw -0.5                          ; IEEE 754r/SSE5
                                 ; половинная точность
dd 1.2                           ; одинарная точность
dd 1.222_222_222                 ; допускается использовать
                                 ; знак подчеркивания
dd 0x1p+2                        ; 1.0x2^2 = 4.0
dq 0x1p+32                       ; 1.0x2^32 = 4 294 967 296.0
dq 1.e10                         ; 10 000 000 000.0
dq 1.e+10                        ; синоним для 1.e10
dq 1.e-10                        ; 0.000 000 000 1
dt 3.14159265358979238462 ; число Пи
do 1.e+4000                      ; IEEE 754r четверная точность
```

Схема вычисления выражения с помощью сопроцессора x87

1. `finit` - инициализация
2. `fld` - загрузка операндов
3. вычисления выражения с помощью команд сопроцессора
4. `fstp` - выгрузка результата и очистка стека x87
5. при необходимости возврата вещественного значения - оставить его на вершине стека x87 (`st0`)

Команды `fld`/`fstp` работают только с памятью.

Инструкции x87

- `fld, fld1, fldz, fldpi, fild`
- `fst, fstp, fist, fistp`
- `fabs, fsqrt`
- `fadd, faddp, fsub, fsubp, fmul, fmulp, fdiv, fdivp`
- `fucomi, fucomip`
- другие https://en.wikipedia.org/wiki/X86_instruction_listings#x87_floating-point_instructions

Один неявный операнд - st0

Два неявных операнда - левый st1, правый st0

Пример

Требуется взять по модулю число двойной точности, расположенное в статической памяти.

```
section .bss
   var resq 1          ; резервируем в статической памяти 8 байт

section .text
   finit               ; инициализируем сопроцессор
   fld   qword [var] ; помещаем на стек регистров значение из памяти
   fabs                ; берем по модулю значение из верхнего регистра и
                       ; помещаем туда же полученный результат
   fstp  qword [var] ; пересылаем в память содержимое занятого регистра
                       ; и освобождаем этот регистр
```

Пример

Требуется вычислить разность двух чисел с плавающей точкой одинарной точности. Разность необходимо расширить до двойной точности и сохранить в третьей переменной соответствующего размера. Все переменные расположены в статической памяти.

```
section .bss
   x resd 1          ; резервируем 4 байта для первой переменной x
   y resd 1          ; резервируем 4 байта для второй переменной y
   z resq 1          ; резервируем 8 байт для сохранения результата

section .text
   finit             ; инициализируем сопроцессор
   fld   dword [x]   ; st0 = x
   fld   dword [y]   ; st0 = y, st1 = x
   fsubp             ; Вычисляем st1 – st0, освобождаем один регистр,
                     ; результат записываем в верхний элемент стека
   fstp  qword [z]   ; Снимаем со стека регистров x87 верхнее значение
                     ; и записываем его в переменную z
```

Пример

Реализовать функцию `int cmp(double a, double b)`, возвращающую `1`, если `a` больше `b`, `-1`, если `b` больше `a`, и `0`, если параметры равны.

```
section .text
cmp:
   push   ebp              ; Стандартный пролог для соглашения cdecl
   mov    ebp, esp

   xor    eax, eax         ; Предварительно обнуляем возвращаемое значение
   fld    qword [ebp+16]   ; Помещаем на стек второй параметр b
   fld    qword [ebp+8]    ; Помещаем на стек первый параметр a
   fucomi st1              ; Сравниваем st0|a vs. st1|b
                           ; Результаты сравнения сразу попадают в ZF и CF
   finit                   ; Через инициализацию очищаем стек регистров

   setb   al               ; Если a больше b, помещаем в al 1
   jbe    .1               ; Только если a меньше b …
   mov    al, 0xff         ; … помещаем в al -1
 .1:
   movsx  eax, al          ; Расширяем al до всего регистра eax

   pop    ebp
   ret
```

Задача

На стандартный вход подается вещественное число, задающее длину радиуса окружности. Требуется вычислить длину окружности и напечатать ее на стандартный вывод.