

# Машинно-зависимые языки программирования

Савельев Игорь Леонидович

- Плавающая точка
- Обратная польская запись
- Сопроцессор

# 1. Плавающая точка

float	4	dd
double	8	dq
long double	10	dt

```
section .data
    mvar4    dd 12.14
    mvar8    dq 12.14
    mvar10   dt 12.14
section .text
global CMAIN
CMAIN:
    mov ebp, esp; for correct debugging
    ;write your code here
    finit
    fld dword [mvar4]
    fld qword [mvar8]
    fld tword [mvar10]
```

mvar4 0x41423d71  
mvar8 0x402847ae147ae148  
mvar10 0x4002c23d70a3d70a3d71


st0	12.1400000000000001	(raw 0x4002c23d70a3d70a3d71)
st1	12.1400000000000001	(raw 0x4002c23d70a3d70a4000)
st2	12.140000343322754	(raw 0x4002c23d710000000000)

1. Плавающая точка

$123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$

$123.456 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2} + 6 \cdot 10^{-3}$

	Делитель	Час.	Ост.
<b>123</b>	10	12	3
12	10	1	2
1	10	0	1



123

	Делитель	Час.	Ост.
<b>123</b>	2	61	1
61	2	30	1
30	2	15	0
15	2	7	1
7	2	3	1
3	2	1	1
1	2	0	1
			0



01111011 = 0x7b

1. Плавающая точка

123 = 1\*10^2 + 2\*10^1 + 3\*10^0

123.456 = 1\*10^2 + 2\*10^1 + 3\*10^0 + 4\*10^-1 + 5\*10^-2 + 6\*10^-3

	Множ	Произ	Целое
0,456	2	0,912	0
0,912	2	1,824	1
0,824	2	1,648	1
0,648	2	1,296	1
0,296	2	0,592	0
0,592	2	1,184	1
0,184	2	0,368	0
0,368	2	0,736	0
0,736	2	1,472	1
0,472	2	0,944	0
0,944	2	1,888	1
0,888	2	1,776	1
0,776	2	1,552	1
...	...	...	...



123.456 = 0111011.0111010010111..... = 1.1110110111010010111.... \* 2^6

## 1. Плавающая точка

32 бита (float)

S (1 бит) для хранения знака

E (8 бит) для экспоненты

M (23 бита) для мантиссы



*Внутренности числа с плавающей запятой.*



*Три части числа с плавающей запятой.*

$$(-1)^S * 1, M * 2^{(E-127)}$$

## 1. Плавающая точка

$$(-1)^S * 1, M * 2^{(E-127)}$$

Экспонента как окно (Window) или интервал между двумя соседними целыми степенями двойки.

Мантисса - смещение (Offset) в этом окне.



*Три части числа с плавающей запятой.*

## 1. Плавающая точка

S	WINDOW	OFFSET
---	--------	--------

*Три части числа с плавающей запятой.*

Окно сообщает нам, между какими двумя последовательными степенями двойки будет число:  $[0,1]$ ,  $[1,2]$ ,  $[2,4]$ ,  $[4,8]$  и так далее (вплоть до  $[2^{127}, 2^{128}]$ ). Смещение разделяет окно на  $2^{23} = 8388608$  сегментов. С помощью окна и смещения можно аппроксимировать число. Окно — это отличный механизм защиты от выхода за границы. Достигнув максимума в окне (например, в  $[2,4]$ ), можно «переплыть» вправо и представить число в пределах следующего окна (например,  $[4,8]$ ). Ценой этого будет только небольшое снижение точности, потому что окно становится в два раза больше.

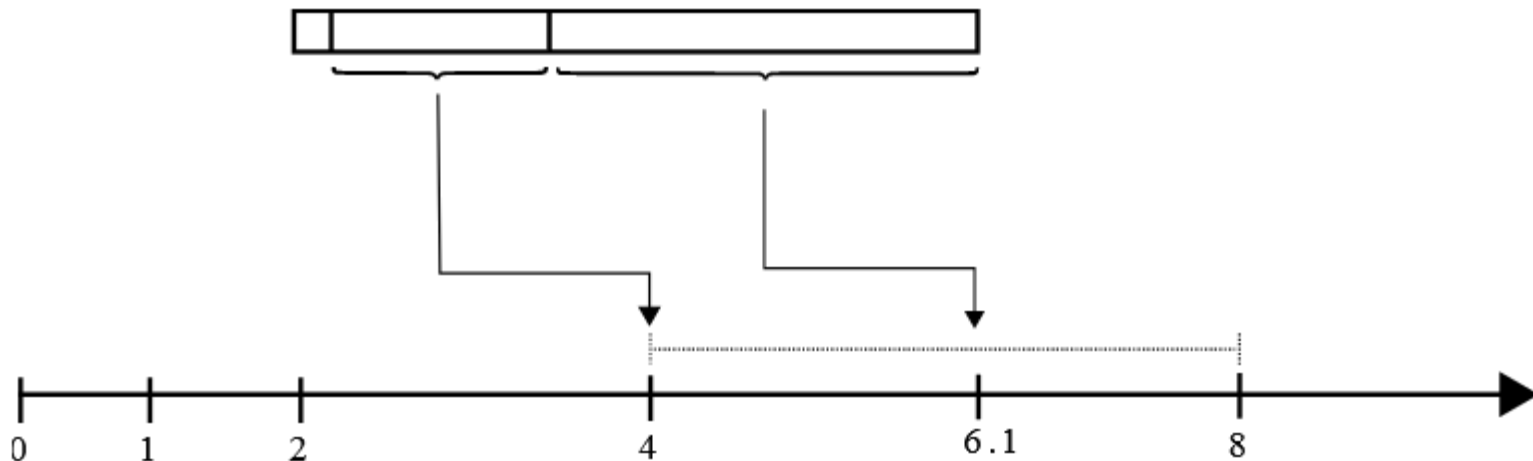


## 1. Плавающая точка

Сколько точности теряется, когда окно закрывает больший интервал?

Окно  $[0,1]$ , в котором 8388608 смещений накладываются на интервал размером 1, что даёт нам точность  $(1-0) / 8388608 = 0,00000011920929$ .

Окно  $[2048,4096]$  8388608 смещений накладываются на интервал  $(4096-2048) = 2048$ , что даёт нам точность  $(4096-2048)/8388608 = 0,0002$ .



1. Плавающая точка

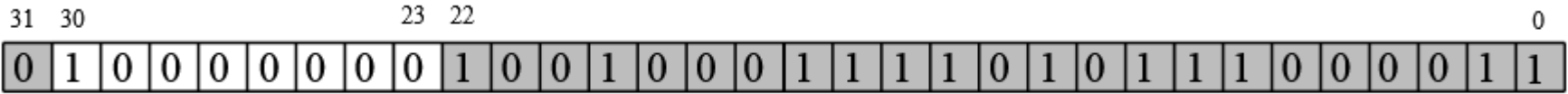
3,14.

- Число 3,14 положительно  $\rightarrow S = 0$ .
- Число 3,14 находится между степенями двойки 2 и 4, то есть окно числа с плавающей запятой должно начинаться с  $2^1 \rightarrow E = 128$  (см. формулу, где окно — это  $2^{(E-127)}$ ).
- Наконец, есть  $2^{23}$  смещений, которыми можно выразить расположение 3,14 внутри интервала [2-4]. Оно находится в  $\frac{3,14 - 2}{4 - 2} = 0,57$  внутри интервала, что даёт нам смещение  $M = 2^{23} * 0,57 = 4781507$

$S = 0 = 0b$

$E = 128 = 10000000b$

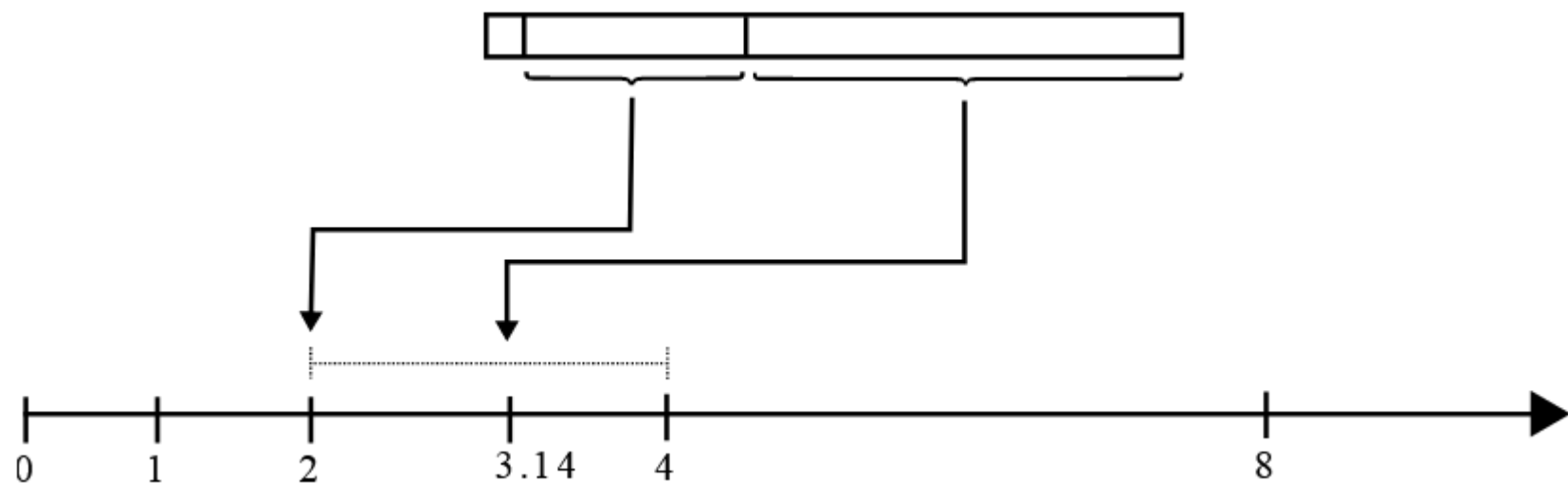
$M = 4781507 = 10010001111010111000011b$



3,14 аппроксимируется как 3,1400001049041748046875

## 1. Плавающая точка

$$3,14 = (-1)^0 * 1,57 * 2^{(128-127)}$$



## 1. Плавающая точка

Степень	E-127	Экспонента (E)
-2	$125 - 127 = -2$	125 (01111101)
-1	$126 - 127 = -1$	126 (01111110)
0	$127 - 127 = 0$	127 (01111111)
1	$128 - 127 = 1$	128 (10000000)
2	$129 - 127 = 2$	129 (10000001)



### Экспонента

- 1) позволяет указывать отрицательные значения
- 2) увеличивается, при увеличении степени

Если float представить как int, то увеличивая число на единицу, мы получим следующее допустимое число float, уменьшая – предыдущее, с учетом погрешности

```
float f = 3.14;  
int i = 0;  
memcpy(&i, &f, 4);  
i++;  
memcpy(&f, &i, 4);
```

## 1. Плавающая точка

$$(-1)^S * 1, M * 2^{(E-127)}$$

Нормализованное:  $1 \leq M < 10$

Пример: 1,57; -1,99

Денормализованное:  $0,1 \leq M < 1$

Пример: 0,57; -0,99

Числа простой и двойной точности (float (DD) и double (DQ) соответственно) могут быть представлены только в нормированной форме. **При этом бит целой части числа является скрытым и подразумевает логическую 1.** Остальные 23 (52) разряда хранят двоичную мантиссу числа.

Числа двойной расширенной точности (long double (DT)) могут быть представлены как в нормированной, так и в ненормированной форме, поскольку бит целой части числа **не является скрытым** и может принимать значения как 0, так и 1.

Основным типом данных, которыми оперирует математический сопроцессор, являются 10-байтные данные (DT).

## 1. Плавающая точка

# 32 бита

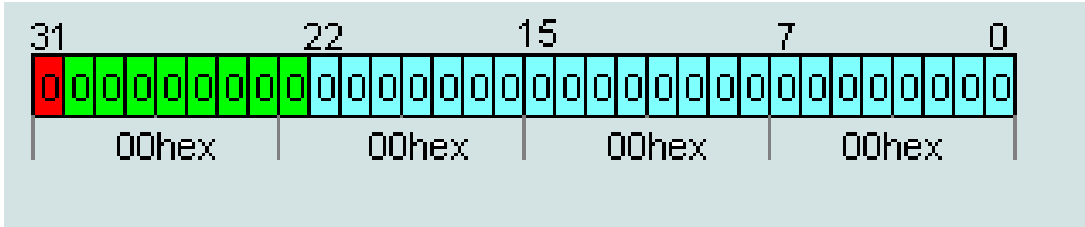
00 00 00 00 hex = 5,87747175411144e-39 (мин. положительное число) 0-127 = -127

80 00 00 00 hex = -5,87747175411144e-39 (мин. отрицательное число) 0-127 = -127

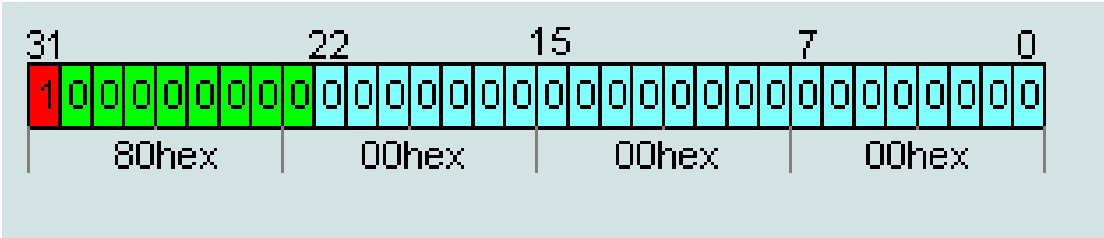
**7f ff ff ff** hex = 6,80564693277058e+38 (макс. положительное число) 255-127 = 128

ff ff ff ff hex = -6,80564693277058e+38 (макс. отрицательное число) 255-127 = 128

число IEEE754 = 00 00 00 00hex считается числом +0



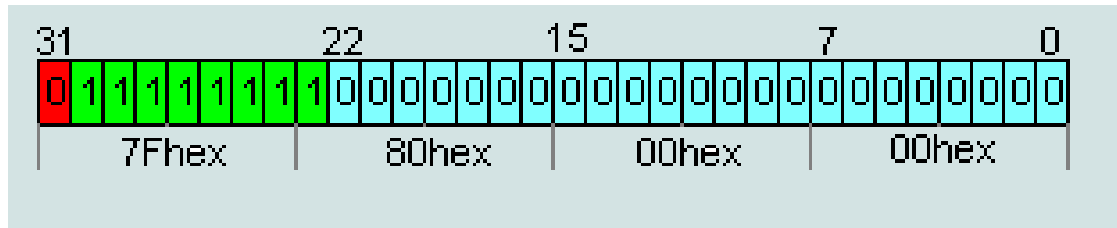
число IEEE754 = 80 00 00 00hex считается числом -0



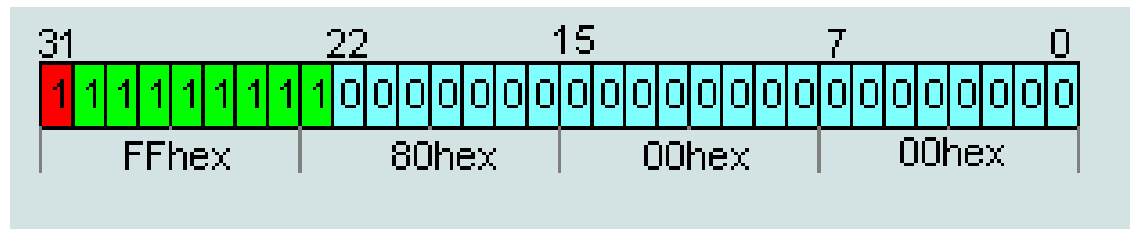
22 бит == 0 для чисел!

число IEEE754=7F **7F** FF FF hex максимальное число (3,40282347 e+38) (**E**=254)

число IEEE754=7F 80 00 00hex считается числом  $+\infty$



число IEEE754=FF **7**F FF FF hex минимальное число (-3,40282347 e+38) (**E**=254)



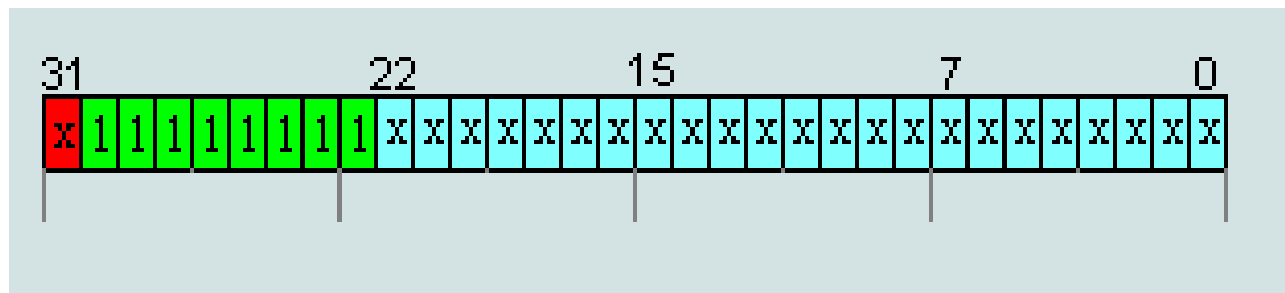
## 1. Плавающая точка

32 бита

числа IEEE754=FF (1xxx)X XX XX hex не считаются числами (NaN)

числа IEEE754=7F (1xxx)X XX XX hex не считаются числами (NaN)

Числа представленные в битах с 0...22 могут быть любым числом кроме 0 (т.е.  $+\infty$  и  $-\infty$ ).



Бит 22 == 1 -> SNaN, сигнальный NaN, вызывает генерацию исключения

x111.1111.1xx.xxxx.xxxx.xxxx.xxxx.xxxx

Бит 22 == 0 -> QNaN, тихий NaN, не вызывает генерацию исключения

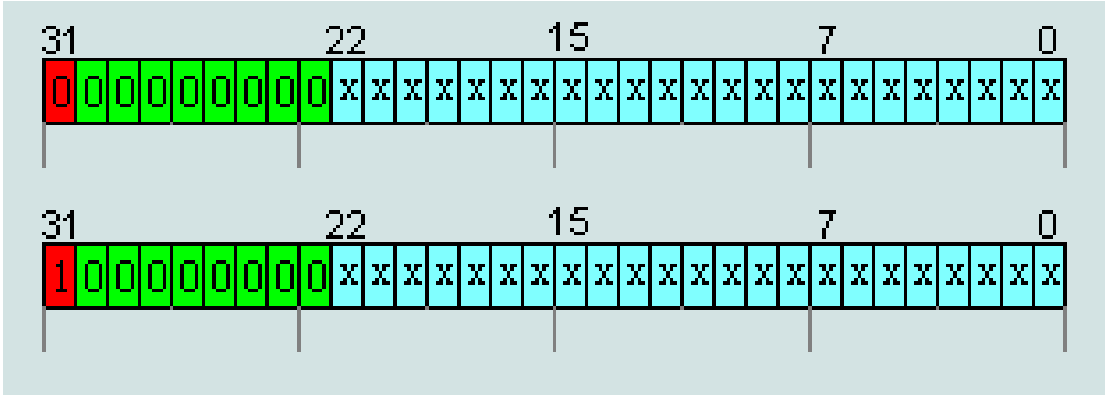
x111.1111.0xx.xxxx.xxxx.xxxx.xxxx.xxxx



# 1. Плавающая точка

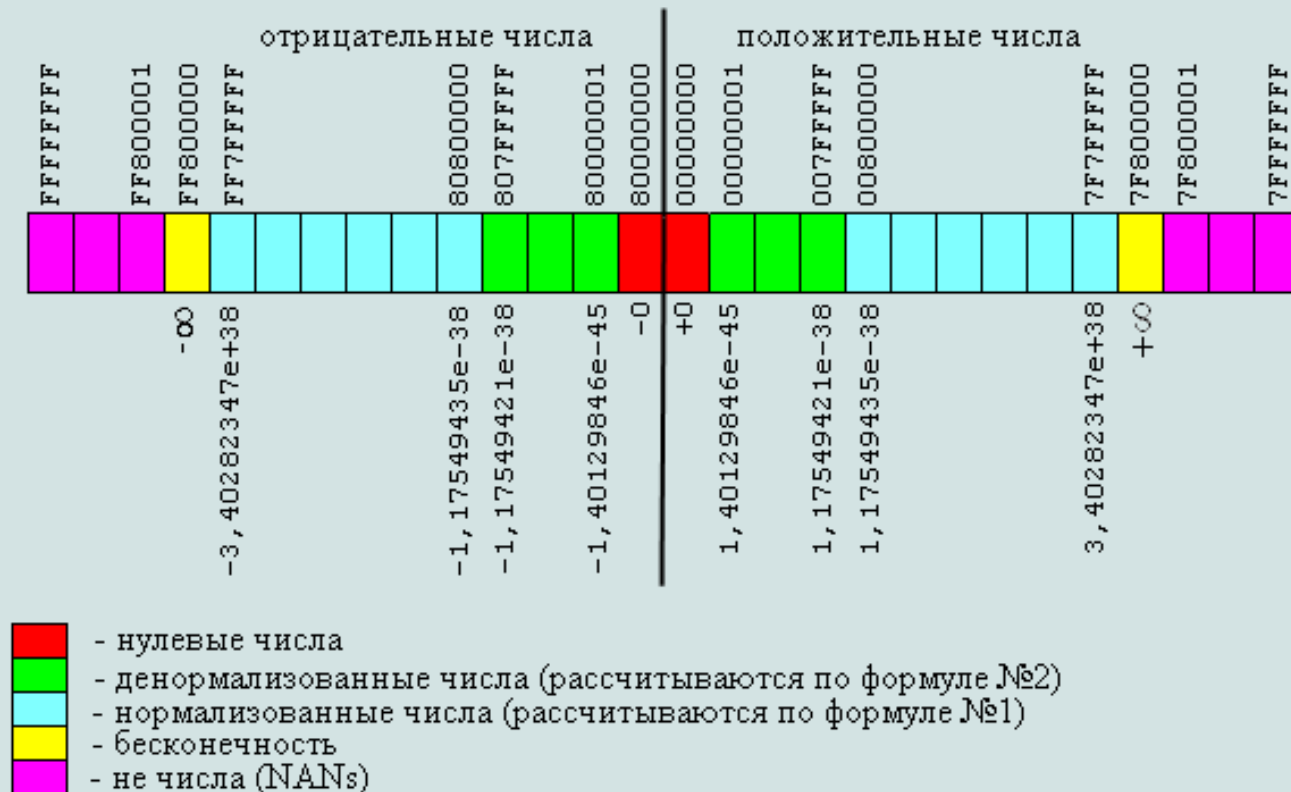
32 бита

числа IEEE754=(x000) (0000) (0xxx)X XX XX hex считаются денормализованными числами



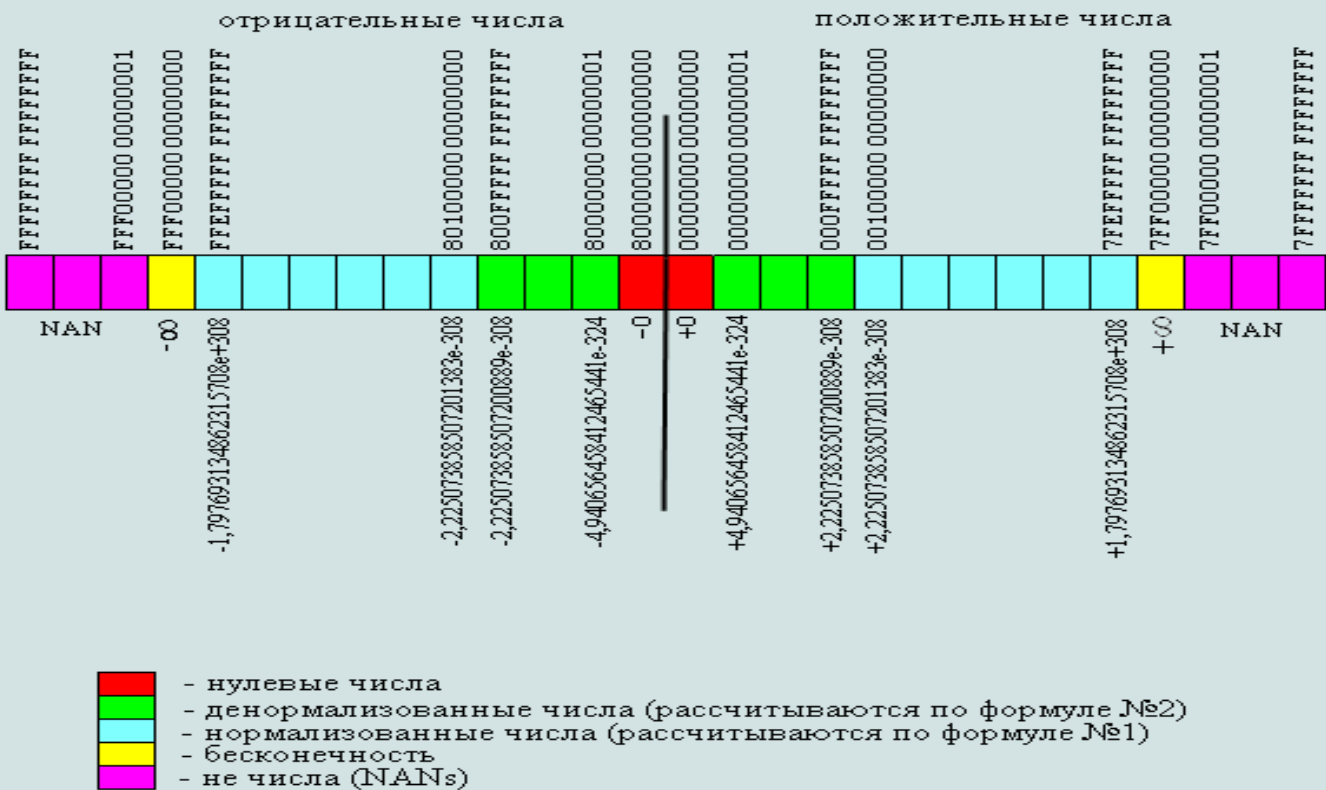
# 1. Плавающая точка

32 бита



# 1. Плавающая точка

64 бита (Е – 11 бит, М – 52 бита, смещение E=1023)



1. Плавающая точка

Тип	Размер	Е	М	Смещение
db	8	4	3	7
dw	16	5	10	15
dd	32	8	23	127
dq	64	11	53	1023
dt	80	15	63	16383

2. Обратная польская запись

$A = (1 + 2) * 4 + 3$

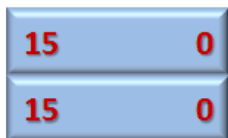
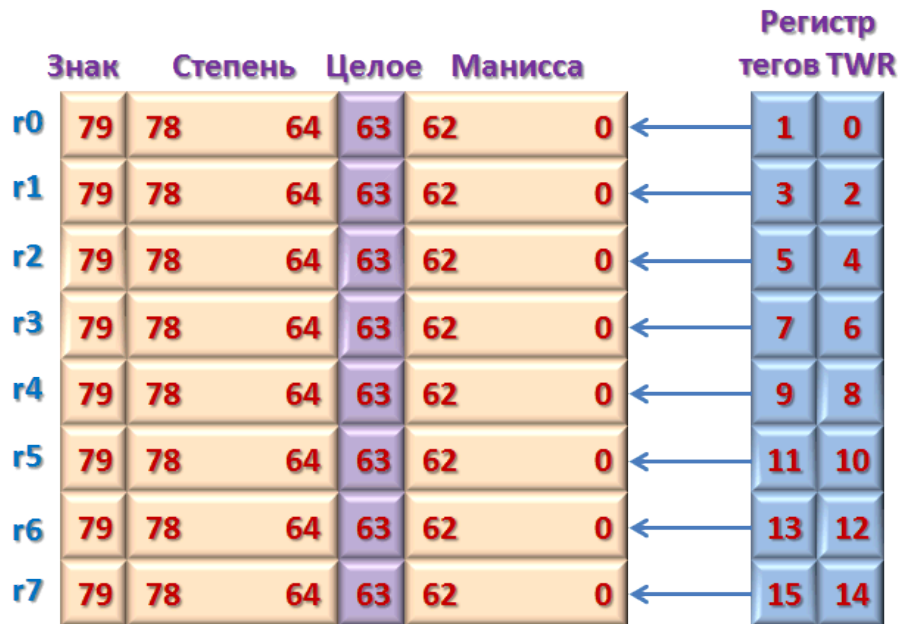
$1 \ 2 + 4 \times 3 + A =$

$A = (1 + 2) * (4 + 3)$

$1 \ 2 + 4 \ 3 + * A =$

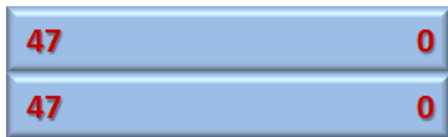
Шаг	Позиция	Инструкции	Содержимое стека
1	<b>1</b> $2 + 4 * 3 + A =$	Push 1	1
2	<b>1</b> <b>2</b> $+ 4 * 3 + A =$	Push 2	2, 1
3	<b>1</b> $2 + $ <b>4</b> $ * 3 + A =$	pop 2, pop 1, сложить, push 3	3
4	<b>1</b> $2 + $ <b>4</b> $ * $ <b>3</b> $ + A =$	Push 4	4, 3
5	<b>1</b> $2 + $ <b>4</b> $ * $ <b>3</b> $ + A =$	Pop 4, pop 3, умножить, push 12	12
6	<b>1</b> $2 + 4 * $ <b>3</b> $ + A =$	Push 3	3, 12
7	<b>1</b> $2 + 4 * $ <b>3</b> $ + $ <b>A</b> $ =$	Pop 3, pop 12 сложить, push 15	15
8	<b>1</b> $2 + 4 * 3 + $ <b>A</b> $ =$	Push A	A, 15
9	<b>1</b> $2 + 4 * 3 + A =$	Pop A, pop 15 переместить 15 в ячейку A	-
10		окончание алгоритма	

### 3. Сопроцессор



Регистр управления CWR

Регистр состояния SWR



Указатель команд IPR

Указатель данных DPR

В программной модели сопроцессора можно выделить три группы регистров:

Восемь регистров **r0...r7**, составляющих основу программной модели сопроцессора — **стек сопроцессора**. Размерность каждого регистра 80 битов. Такая организация характерна для устройств, специализирующихся на обработке вычислительных алгоритмов.

Три служебных регистра:

- регистр состояния сопроцессора **swr** (Status Word Register — регистр слова состояния) — отражает информацию о текущем состоянии сопроцессора;
- управляющий регистр сопроцессора **cwr** (Control Word Register — регистр слова управления) — управляет режимами работы сопроцессора;
- регистр тегов **twr** (Tags Word Register — слово тегов) — используется для контроля за состоянием каждого из регистров стека.

Два регистра указателей — данных **dpr** (Data Point Register) и команд **ipr** (Instruction Point Register). Они предназначены для запоминания информации об адресе команды, вызвавшей исключительную ситуацию и адресе ее операнда. Эти указатели используются при обработке исключительных ситуаций (но не для всех команд).

### 3. Сопроцессор



**Регистр состояния swr (FSTAT)** — отражает текущее состояние сопроцессора после выполнения последней команды. В регистре swr содержатся поля, позволяющие определить: какой регистр является текущей вершиной стека сопроцессора, какие исключения возникли после выполнения последней команды, каковы особенности выполнения последней команды (некий аналог регистра флагов основного процессора).

Структурно регистр swr состоит из:

- 6 флагов исключительных ситуаций **PE, OE, UE, ZE, DE, IE**.

Исключения — это разновидность прерываний, с помощью которых процессор информирует программу о некоторых особенностях ее реального исполнения. Сопроцессор также обладает способностью возбуждения подобных прерываний при возникновении определенных ситуаций (не обязательно ошибочных). Все возможные исключения сведены к би типам, каждому из которых соответствует 1 бит в регистре swr. Программисту не обязательно писать обработчик для реакции на ситуацию, приведшую к некоторому исключению.

Сопроцессор умеет самостоятельно реагировать на многие из них. Это так называемая обработка исключений по умолчанию. Для того чтобы вызвать обработку определенного типа исключения по умолчанию, необходимо это исключение оставить не маскированным. Такое действие выполняется с помощью установки в 1 соответствующего бита в управляющем регистре сопроцессора swr.

### 3. Сопроцессор



Типы исключений, фиксируемые с помощью регистра swr:

IE (Invalid operation Error) — недействительный код операция;

DE (Denormalized operand Error) — ненормированный операнд;

ZE (divide by Zero Error) — ошибка деления на ноль;

OE (Overflow Error) — ошибка переполнения. Возникает в случае выхода порядка числа за максимально допустимый диапазон;

UE (Underflow Error) — ошибка антипереполнения. Возникает, когда результат слишком мал (близок к нулю);

PE (Precision Error) — ошибка точности. Устанавливается, когда сопроцессору приходится округлять результат из-за того, что его точное представление невозможно. Так, сопроцессору никогда не удастся точно разделить 10 на 3.

При возникновении любого из этих шести типов исключений устанавливается в единицу соответствующий бит в регистре swr, вне зависимости от того, было ли замаскировано это исключение в регистре swr или нет.



### 3. Сопроцессор



- бита ошибки работы стека сопроцессора SF (Stack Fault). Бит устанавливается в 1, если возникает одна из трех исключительных ситуаций — PE, UE или IE. В частности, его установка информирует о попытке записи в заполненный стек, или, напротив, попытке чтения из пустого стека. После того как значение этого бита проанализировано, его нужно снова сбросить в 0, вместе с битами PE, UE и IE (если они были установлены);
- бита суммарной ошибки работы сопроцессора ES (Error Summary). Бит устанавливается в 1, если возникает любая из шести перечисленных выше исключительных ситуаций;
- четырех битов c0...c3 (Condition Code) — кода условия. Назначение этих битов аналогично флагам в регистре EFLAGS основного процессора — отразить результат выполнения последней команды сопроцессора.
- трехбитного поля TOP. Поле содержит указатель регистра текущей вершины стека.
- бита В занятости сопроцессора.

### 3. Сопроцессор



**Регистр управления работой сопроцессора *cwr* (FCTRL)**— определяет особенности обработки числовых данных. С помощью полей в регистре ***cwr*** можно регулировать точность выполнения численных вычислений, управлять округлением, маскировать исключения.

Регистр управления сопроцессора CWR состоит из:

- шести масок исключений PM, UM, OM, ZM, DM, IM;
- поля управления точностью PC (Precision Control);
- поля управления округлением RC (Rounding Control).

Маски исключений предназначены для маскирования исключительных ситуаций, возникновение которых фиксируется с помощью шести бит регистра *swr*. Если какие-то маскирующие биты исключений в регистре *swr* установлены в 1, то соответствующие исключения будут обрабатываться самим сопроцессором. Если для какого-либо исключения в соответствующем бите маски исключений регистра *swr* содержится 0, то при возникновении исключения этого типа будет возбуждено прерывание int 16 (10h). Операционная система должна содержать (или программист должен написать) обработчик этого прерывания. Он должен выяснить причину прерывания, после чего, если это необходимо, исправить ее, а также выполнить другие действия.

### 3. Сопроцессор



2-битовое поле управления точностью **PC** предназначено для выбора длины мантиссы. Возможные значения в этом поле означают:

PC=00 — длина мантиссы  $1 + 23 = 24$  бита;

PC=10 — длина мантиссы  $1 + 52 = 53$  бита;

PC=**11** — длина мантиссы  $1 + 63 = 64$  бита.

По умолчанию устанавливается значение поля PC=11.

### 3. Сопроцессор

```
section .data
mvar14 dt 13.515151
mvar15 dq 14.515151
tmp    dw 0
section .text
global CMAIN
CMAIN:
    fld tword [mvar14]
    fnstcw [tmp]
    and [tmp], word 1111110011111111b;
    or [tmp],  word 0000000000000000b
    fldcw [tmp]
    fld tword [mvar14]
    fadd qword [mvar15]

; or [tmp], word 0000001000000000b
; or [tmp], word 0000001100000000b
ret
```

Флаги PC регистра CWR

st0	28.030301999999999471718803434328038	(raw 0x4003e03e0ef99806f132)
st1	28.0303019999999998940893419785425067	(raw 0x4003e03e0ef99806f000)
st2	28.0303020477294921875	(raw 0x4003e03e0f0000000000)
st3	13.51515100000000000000404731803627101	(raw 0x4002d83e0ef99806f263)
st4	0	(raw 0x00000000000000000000)
st5	0	(raw 0x00000000000000000000)
st6	0	(raw 0x00000000000000000000)
st7	0	(raw 0x00000000000000000000)

### 3. Сопроцессор



Поле управления округлением **RC** позволяет управлять процессом округления чисел в процессе работы сопроцессора. Необходимость операции округления может появиться в ситуации, когда после выполнения очередной команды сопроцессора получается не представимый результат, например, периодическая дробь. Установив одно из значений в поле **RC**, можно выполнить округление в необходимую сторону. Значения поля **RC** с соответствующим алгоритмом округления:

- **00** — значение округляется к ближайшему числу, которое можно представить в разрядной сетке регистра сопроцессора;
- **01** — значение округляется в меньшую сторону;
- **10** — значение округляется в большую сторону;
- **11** — производится отбрасывание дробной части числа. Используется для приведения значения к форме, которая может использоваться в операциях целочисленной арифметики.

Бит 12 в регистре `swr` физически отсутствует и считывается равным 0.

### 3. Сопроцессор

**Регистр тегов twr (FTAGS)**— представляет собой совокупность двухбитовых полей. Каждое поле соответствует определенному физическому регистру стека и характеризует его текущее состояние. Команды сопроцессора используют этот регистр, например, для того, чтобы определить возможность записи значений в эти регистры. Изменение состояния любого регистра стека отражается на содержимом соответствующего этому регистру 2-битового поля регистра тега. Возможны следующие значения в полях регистра тега:

- 00 — регистр стека сопроцессора занят допустимым ненулевым значением;
- 01 — регистр стека сопроцессора содержит нулевое значение;
- 10 — регистр стека сопроцессора содержит одно из специальных численных значений, за исключением нуля;
- 11 — регистр пуст и в него можно производить запись. Это значение в двухбитовом поле регистра тегов не означает, что все биты соответствующего регистра стека должны быть обязательно нулевыми.

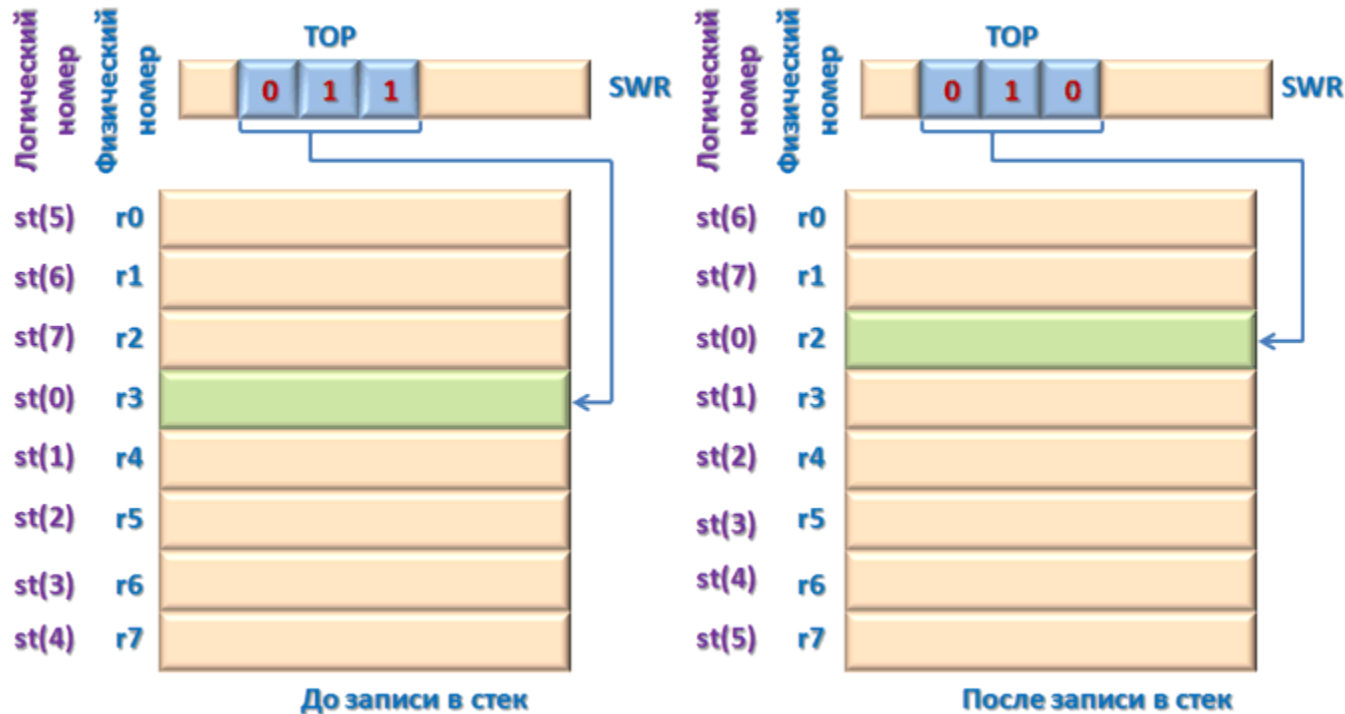
Регистр  
тегов TWR

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

### 3. Сопроцессор

#### Принцип работы сопроцессора

Регистровый стек сопроцессора организован по принципу кольца. Среди восьми регистров, составляющих стек, нет такого, который является вершиной стека. Все регистры стека с функциональной точки зрения абсолютно одинаковы и равноправны. Вершина в кольцевом стеке сопроцессора является плавающей. Контроль текущей вершины осуществляется аппаратно с помощью 3-битового поля top регистра swr.



### 3. Сопроцессор

В поле **top** фиксируется номер регистра стека **r0...r7**, являющегося в данный момент текущей вершиной стека.

Команды сопроцессора оперируют не физическими номерами регистров стека **r0...r7**, а их логическими номерами **st(0)...st(7)**. С помощью логических номеров реализуется относительная адресация регистров стека сопроцессора. Например, если текущей вершиной до записи в стек является физический регистр стека **r3**, то после записи в стек текущей вершиной становится физический регистр стека **r2**. То есть, по мере записи в стек, указатель его вершины движется по направлению к младшим номерам физических регистров (уменьшается на единицу). Если текущей вершиной является **r0**, то после записи очередного значения в стек сопроцессора его текущей вершиной станет физический регистр **r7**. Что касается логических номеров регистров стека **st(0)...st(7)**, то они перемещаются вместе с изменением текущей вершины стека.

**Логическая вершина стека всегда имеет имя st(0).**

Поскольку при написании программы разработчик манипулирует не абсолютными, а относительными номерами регистров стека, у него могут возникнуть трудности при попытке интерпретации содержимого регистра тегов **twr**, с соответствующими физическими регистрами стека. В качестве связующего звена необходимо привлекать информацию из поля **top** регистра **swr**. Таким образом реализуется принцип **кольца**.



### 3. Сопроцессор

Такая организация стека обладает большой гибкостью, в частности при передаче параметров в процедуру. Для повышения гибкости разработки и использования процедур не желательно привязывать их по передаваемым параметрам к аппаратным ресурсам (физическим номерам регистров сопроцессора). Гораздо удобнее задавать порядок следования передаваемых параметров в виде логических номеров регистров. Такой способ передачи был бы однозначным и не требовал от разработчика знания лишних подробностей об аппаратных реализациях сопроцессора. Логическая нумерация регистров сопроцессора, поддерживаемая на уровне системы команд, идеально реализует эту идею. При этом не имеет значения, в какой физический регистр стека сопроцессора были помещены данные перед вызовом подпрограммы, определяющим является только порядок следования параметров в стеке. По этой причине подпрограмме важно знать только порядок размещения передаваемых параметров в стеке.

Спасибо