

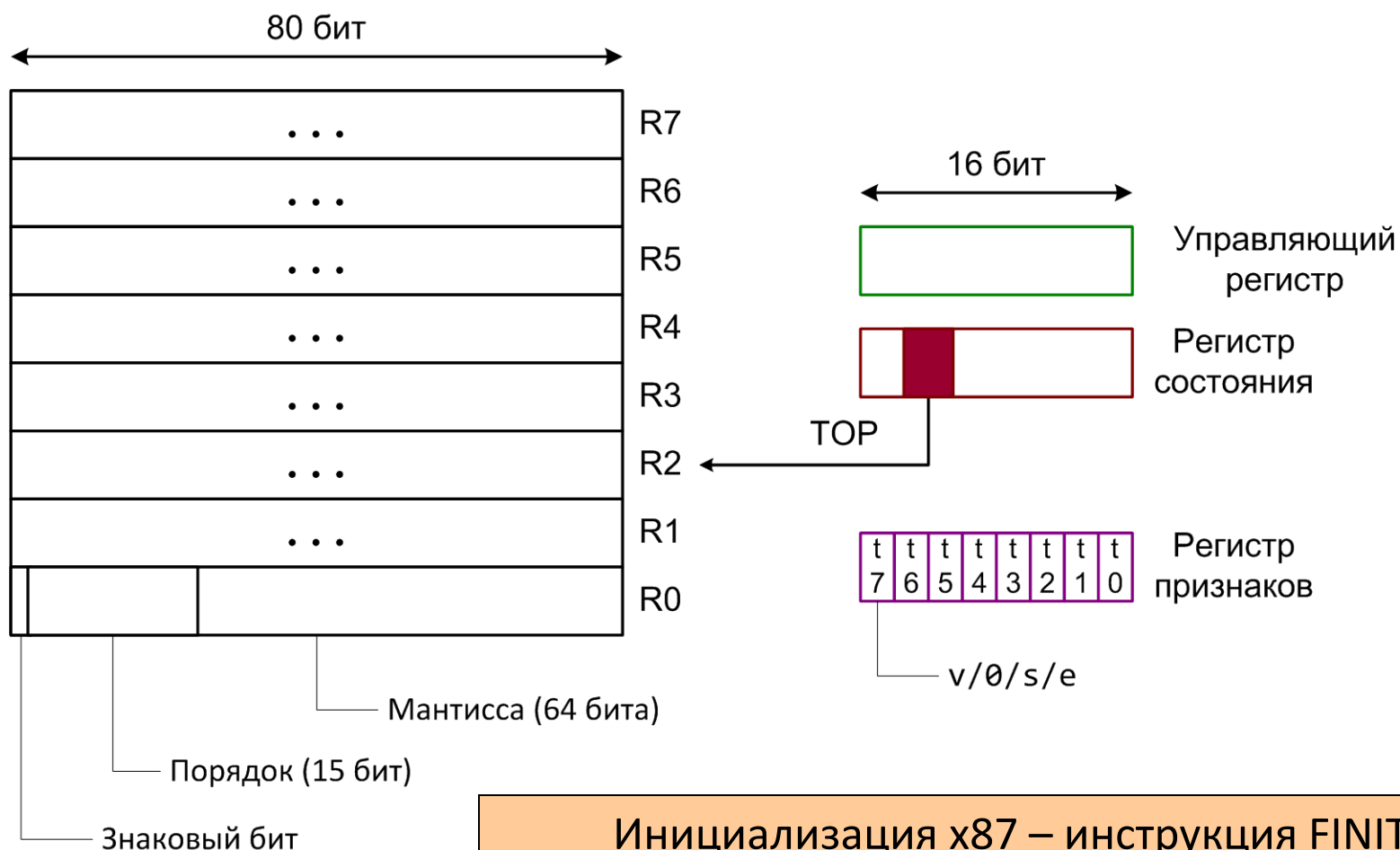
# Лекция 0x12

8 апреля

# Числа с плавающей точкой: промежуточные итоги

- IEEE 754 – четкое определение математических свойств
- Представляются числа вида  $M \times 2^E$
- Семантика операций не зависит от особенностей аппаратуры
  - Сперва точное вычисление, затем округление
- Отличия от «настоящей» арифметики
  - Нарушаются свойства ассоциативности и дистрибутивности
  - Создаются сложности для компилятора и серьезных математических вычислений

# Упрощенная схема x87



Инициализация x87 – инструкция FINIT

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

# Размер чисел с плавающей точкой

Регистры данных

Знак	79	78	64	63	0
R7					
R6					
R5					
R4					
R3					
R2					
R1					
R0					

Порядок      Мантисса

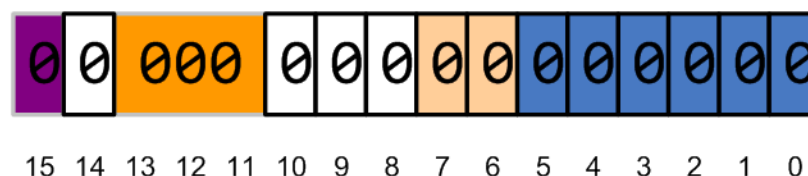
**Обмен данными только с памятью!**

dd 1.234567e20 ; Константы одинарной точности  
 dq 1.234567e20 ; Двойной точности  
 dt 1.234567e20 ; Расширенной точности

# Слово (регистр) состояния

- SF – переполнение стека (C1 показывает направление)
- Исключительные ситуации: точность, переполнение, деление на ноль, денормализованный операнд, «неправильные» данные

После инициализации



Коды условий

Общий признак ошибки

Сбой стека

Флаги

исключительных  
ситуаций

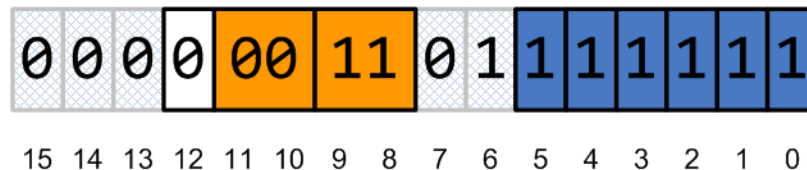
Инициализация x87  
инструкция FINIT

SW = 0x0000

# Управляющий регистр

- Точность: одинарная, двойная, расширенная
- Округление: к ближайшему четному, к нулю, к +/- бесконечности
- Флаг X – совместимость с 287
- Маски соответствуют исключениям в слове состояния

После инициализации



Округление

Точность

Маски флагов  
исключительных  
ситуаций

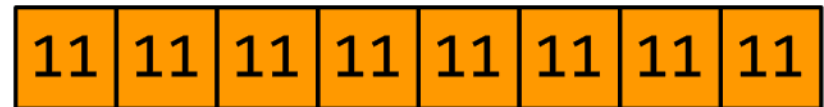
Инициализация x87  
инструкция FINIT

CW = 0x037F

# Регистр признаков (тагов)

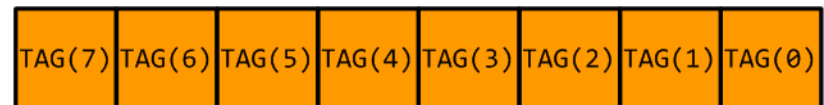
- Состояние регистров
  - 0 – нормализованное число с плавающей точкой
  - 1 – число ноль
  - 2 – особые числа (NaN,  $\pm \infty$ , денормализованное число)
  - 3 – регистр свободен
- Нумерация соответствует физическим регистрам

После инициализации



15

0



Инициализация x87  
инструкция FINIT

Tag = 0xFFFF

# NASM и числа с плавающей точкой

```

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 3.141592653589793238462 ; число Пи
do 1.e+4000      ; IEEE 754r четверная точность

```

__float8__	__Infinity__
__float16__	__NaN__
__float32__	__QNaN__
__float64__	__SNaN__
__float80m__	
__float80e__	
__float128l__	
__float128h__	

IEEE 754r – опубликован в 2008 году

```

dq +1.5, -__Infinity__, __NaN__
mov eax, __float32__(3.1415926)

```



# Сложение двух чисел

```
%include 'io.inc'
```

```
section .data
```

x dd 11.2

y dd 0.7

```
section .bss
```

z resd 1

```
section .text
```

# global CMAIN

## CMAIN:

# finit

fld dword [x]

```
fld dword [y]
```

# faddp

fstop dword [z]

```
PRINT_HEX 4, z
```

NEWLINE

```
xor  eax,  eax
```

ret

X

11.2 ~ 1.119999980926513671875E1

0x41333333 = 01000001 00110011 00110011 00110011

# Сложение двух чисел

```
%include 'io.inc'
```

```
section .data
```

```
  x dd 11.2
```

```
  y dd 0.7
```

```
section .bss
```

```
  z resd 1
```

```
section .text
```

```
global CMAIN
```

```
CMAIN:
```

```
  finit
```

```
  fld dword [x]
```

```
  fld dword [y]
```

```
  faddp
```

```
  fstp dword [z]
```

```
  PRINT_HEX 4, z
```

```
  NEWLINE
```

```
  xor eax, eax
```

```
  ret
```

y

0.7 ~ 6.99999988079071044921875E-1

0x3F333333 = 

00	111111	00	110011	00	110011	00	110011
----	--------	----	--------	----	--------	----	--------

# Сложение двух чисел

```
%include 'io.inc'
```

```
section .data
```

```
  x dd 11.2
```

```
  y dd 0.7
```

```
section .bss
```

```
  z resd 1
```

```
section .text
```

```
global CMAIN
```

```
CMAIN:
```

```
  finit
```

```
  fld dword [x]
```

```
  fld dword [y]
```

```
  faddp
```

```
  fstp dword [z]
```

```
  PRINT_HEX 4, z
```

```
  NEWLINE
```

```
  xor eax, eax
```

```
  ret
```

z

0x413E6666 = 

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

1.18999996185302734375E1 ~ 11.9

# Печать числа

```
%include 'io.inc'

section .data
x dd 11.2
y dd 0.7

section .bss
z resd 1

section .rodata
lc db '%f', 10, 0

section .text
CEXTERN printf
global CMAIN
```

```
CMAIN:
; ...
; пролог функции
sub     esp, 20

fld     dword [x]
fld     dword [y]
faddp
fst     dword [z]
fstp    dword [esp + 4]
mov     dword [esp], lc
call    printf

add     esp, 20
; эпилог функции
; ...
```

# Печать числа

```
%include 'io.inc'

section .data
x dd 11.2
y dd 0.7

section .bss
z resd 1

section .rodata
lc db '%f', 10, 0

section .text
CEXTERN printf
global CMAIN
```

CMAIN:

```
; ...
; пролог функции
sub     esp, 20

fld     dword [x]
fld     dword [y]
faddp
fst     dword [z]
fstp    dword [esp + 4]
mov     dword [esp], lc
call    printf

add     esp, 20
; эпилог функции
; ...
```

printf печатает мусор  
где ошибка?!?!1

# Печать числа

```
%include 'io.inc'

section .data
x dd 11.2
y dd 0.7

section .bss
z resd 1

section .rodata
lc db '%f', 10, 0

section .text
CEXTERN printf
global CMAIN
```

CMAIN:

```
; ...
; пролог функции
sub     esp, 20

fld     dword [x]
fld     dword [y]
faddp

fst     dword [z]
fstp    qword [esp + 4]
mov     dword [esp], lc
call    printf

add     esp, 20
; эпилог функции
; ...
```

ISO/IEC 9899:1999  
§ 6.5.2.2 абзац №6

# Польская обратная запись

$(w + x + y + z) / 4$



$w \ x \ + \ y \ + \ z \ + \ 4 \ /$

```
section .data
```

```
w dq 1e10
```

```
x dq 1e10
```

```
y dq 1e10
```

```
z dq 1e10
```

```
d dd 4
```

```
section .text
```

```
;...
```

```
fld qword [w]
```

```
fld qword [x]
```

```
faddp
```

```
fld qword [y]
```

```
faddp
```

```
fld qword [z]
```

```
faddp
```

```
fild dword [d]
```

```
fdivp ; st1 / st0
```

```
;...
```

# Порядок действий имеет значение

```
%include 'io.inc'

section .data
x dq 3.14
y dq 1e50
z dq -1e50

section .bss
r resq 1

section .rodata
lc db '%f', 10, 0

section .text
CEXTERN printf
global CMAIN
```

```
CMAIN:
; ... пролог функции
sub     esp, 20

fld     qword [x]
fld     qword [y]
fld     qword [z]
faddp
faddp
fst     qword [r]
fstp    qword [esp + 4]
mov     dword [esp], lc
call    printf

add     esp, 20
; ... эпилог функции
```



# Порядок действий имеет значение

CMAIN:

```
; ... пролог функции  
sub     esp, 20
```

```
fld     qword [y]  
fld     qword [z]  
fld     qword [x]  
faddp  
faddp  
fst     qword [r]  
fstp    qword [esp + 4]  
mov     dword [esp], lc  
call    printf
```

```
add     esp, 20  
; ... эпилог функции
```

CMAIN:

```
; ... пролог функции  
sub     esp, 20
```

```
fld     qword [x]  
fld     qword [y]  
fld     qword [z]  
faddp  
faddp  
fst     qword [r]  
fstp    qword [esp + 4]  
mov     dword [esp], lc  
call    printf
```

```
add     esp, 20  
; ... эпилог функции
```

# Распределение слагаемых

```
section .data
```

```
x dq 1e200
```

```
y dq 1e200
```

```
z dq 1e200
```

```
section .bss
```

```
r resq 1
```

```
section .rodata
```

```
lc db '%lf', 10, 0
```

```
CMAIN:
```

```
; ... пролог функции  
sub     esp, 20
```

```
fld     qword [x]
```

```
fld     qword [y]
```

```
fsubp
```

```
fld     qword [z]
```

```
fmulp
```

```
fst     qword [r]
```

```
fstp    qword [esp + 4]
```

```
mov     dword [esp], lc
```

```
call    printf
```

```
add     esp, 20
```

```
; ... эпилог функции
```

# Распределение слагаемых

```
section .data
```

```
x dq 1e200
```

```
y dq 1e200
```

```
z dq 1e200
```

```
section .bss
```

```
r resq 1
```

```
section .rodata
```

```
lc db '%lf', 10, 0
```

```
CMAIN:
```

```
; ... пролог функции  
sub     esp, 20
```

```
fld     qword [x]
```

```
fld     qword [z]
```

```
fmulp
```

```
fld     qword [y]
```

```
fld     qword [z]
```

```
fmulp
```

```
fsubp
```

```
; вызов printf
```

```
add     esp, 20
```

```
; ... эпилог функции
```

# Распределение слагаемых

```
section .data
```

```
x dq 1e200
```

```
y dq 1e200
```

```
z dq 1e200
```

```
section .bss
```

```
r resq 1
```

```
cw resw 1
```

```
section .rodata
```

```
lc db '%lf', 10, 0
```

```
CMAIN:
```

```
; ... пролог функции
```

```
sub     esp, 20
```

```
fstcw   word [cw]
```

```
and     word [cw], 11111111_11000000b
```

```
fldcw   word [cw]
```

```
fld     qword [x]
```

```
fld     qword [z]
```

```
fmulp
```

```
fld     qword [y]
```

```
fld     qword [z]
```

```
fmulp
```

```
fsubp
```

```
; вызов printf
```

```
add     esp, 20
```

```
; ... эпилог функции
```

# Предопределенные константы

- На «верхушку» стека регистров ( $St0$ ) помещается определенная константа
  - FLD1       $+1.0$
  - FLDL2T    $\log_2 10$
  - L2E        $\log_2 e$
  - FLDPI      $\pi$
  - FLDLG2    $\log_{10} 2$
  - FLDLN2    $\log_e 2$
  - FLDZ       $+0.0$

# Сравнение чисел

```
_Bool isLe(double x, float y) {
    return x <= y;
}
```

Результат сравнения	C3	C2	C0
St0 > St1	0	0	0
St0 < St1	0	0	1
St0 == St1	1	0	0
неопределенно	1	1	1

```
isLe:
    push    ebp
    mov     ebp, esp
    fld     dword [ebp+16]
    fld     qword [ebp+8]
    fucompp ; St0 vs. St1
    fnstsw  ax
    sahf
    setbe   al
    pop     ebp
    ret
```

# Извлечение результатов сравнения

- C3 → ZF, C0 → CF
- Можно использовать условные коды, применяемые при сравнении беззнаковых чисел

isLe:

```

...
fld      dword [ebp+16]
fld      qword [ebp+8]
fucompp  ; St0 vs. St1
fnstsw   ax
sahf
setbe    al
...

```

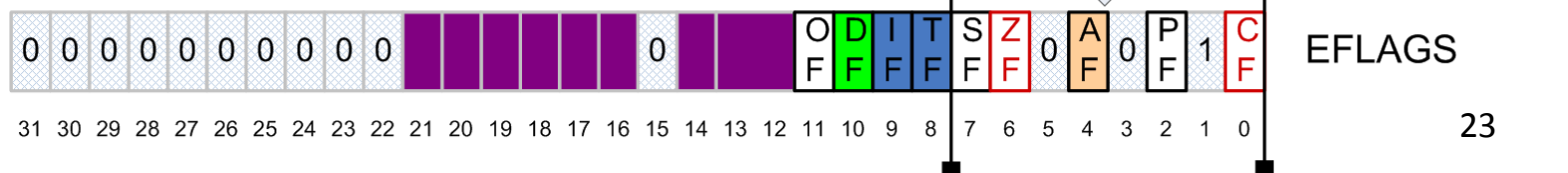
Результат сравнения	ZF	PF	CF
St0 > St(i)	0	0	0
St0 < St(i)	0	0	1
St0 == St(i)	1	0	0
неупорядочены	1	1	1

Слово (регистр) состояния

fnstsw ax

AX

sahf



# Непосредственное обновление EFLAGS

- F[U]COMI[P] St(*i*)

– появились в P6

isLe(2.0, 3.0)

isLe:

```
...
fld     dword [ebp+16]
fld     qword [ebp+8]
fucomip ; st0 vs. st1
setbe   al
...
```

BE:

CF = 1 или ZF = 1

Результат сравнения	ZF	PF	CF
St0 > St( <i>i</i> )	0	0	0
St0 < St( <i>i</i> )	0	0	1
St0 == St( <i>i</i> )	1	0	0
неупорядочены	1	1	1



# Функции: возвращаемое значение – число с плавающей точкой

```
void caller(double *p) {
    *p = inverse(*p);
}
```

```
float inverse(double x) {
    return 1/x;
}
```

```
caller:
    push    ebp
    mov     ebp, esp
    sub     esp, 8
    mov     eax, dword [ebp+8]
    fld     dword [eax]
    fstp    qword [esp]
    call    inverse
    mov     eax, dword [ebp+8]
    fstp    qword [eax]
    leave
    ret
```

На входе в функцию регистры  
St0 – St7 должны быть пустыми

```
inverse:
    push    ebp
    mov     ebp, esp
    fld1
    fld     qword [ebp+8]
    fdivp
    pop     ebp
    ret
```

На выходе из функции:

1. регистры St1 – St7 должны быть пустыми,
2. регистр St0 либо содержит возвращаемое значение, либо см. п. №1

## Некоторые команды x87

- FABS
- FCHS
- FSQRT
- FXCH [ST(*i*)]
- FTST
- FYL2X
- F2XM1
- FCOS
- FSINCOS
- FPTAN
- FPATAN
- FILD  $m[16/32/64]\text{int}$
- FIST[P]  $m[16/32/64]\text{int}$
- FICOM[P]  $m[16/32/64]\text{int}$

## Итоги второй части

- Процессор с точки зрения пользовательской программы
- Основные механизмы реализации языка Си
  - Соответствие типов
  - Указатели, адресная арифметика
  - Управляющие операторы: разные стратегии реализации
  - Размещение данных в памяти
    - Классы памяти: статическая, автоматическая, динамическая
    - Выравнивание
  - Соглашения о вызове функции
- ABI – Application Binary Interface
- IA-32: x86 + x87 + ...
  - Некоторые особенности устройства