


**Федеральное государственное бюджетное образовательное
учреждение высшего образования "Белгородский государственный
технологический университет им. В.Г. Шухова"**

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем.

Лабораторная работа № 5
Команды сопроцессора.
Вариант 13

Выполнил:
Студент группы КБ-211


_____ Коренев Д.Н.

Принял:

_____ Осипов О.В.

Цель работы: изучение команд сопроцессора для выполнения арифметических операций.

Задание

1. Написать функцию `pow(x, y)` для возведения числа x в степень y . Числа x , y могут быть произвольными, в том числе отрицательными. Рассмотреть случаи, когда $x = 0$ и/или $y = 0$. Аргументы передавать подпрограмме через стек. Если алгоритм требует выгрузки чисел из сопроцессора в память или регистры, использовать для этого стек. Подобрать набор тестовых данных для проверки работы функции `pow` (не менее 10). Убедиться в том, что результаты работы написанной функции `pow` и стандартной функции `pow` библиотеки `math.h` языка C или функции `Math.Pow` языка C# совпадают. В отчёт включить текст программы, блок-схему алгоритма функции `pow` и набор тестовых данных.

```
func_pow_double proc
    sub ESP, 24                ; доп.слоты в стеке
                                ;  $x^y = 2^{(y \cdot \log_2(x))}$ 

    mov DWORD PTR [ESP+0+0], 0
    mov DWORD PTR [ESP+0+4], 0
    mov DWORD PTR [ESP+0+8], 0
    mov DWORD PTR [ESP+0+12], 0
    mov DWORD PTR [ESP+0+16], 0
    mov DWORD PTR [ESP+0+20], 0

    fld QWORD PTR [ESP+28+0]    ; загрузка x в сопроцессор
    fldz                       ; загрузка 0.0 в сопроцессор
    db 0dbh, 0f0h+1            ; сравнение ST(0), ST(1)
    ja pow_double_x_less_0     ; если  $x < 0$ 
    jb pow_double_begin        ; если  $x > 0$ 

    fld QWORD PTR [ESP+28+8]    ; загрузка y в сопроцессор
    fldz                       ; загрузка 0.0 в сопроцессор
    db 0dbh, 0f0h+1            ; сравнение ST(0), ST(1)
    jne pow_double_end         ; загрузка 1.0 в сопроцессор
    fld1                       ; загрузка 1.0 в сопроцессор
    fstp QWORD PTR [ESP+0+8]    ; результат в 2-й доп.слот
    jmp pow_double_end

pow_double_x_less_0:          ;  $x < 0$ 
    mov DWORD PTR [ESP+0+16], 1 ; 3-й доп.слот (4 из 8 байта) = 1
    fld QWORD PTR [ESP+28+0]    ; загрузка x в сопроцессор
    fabs                       ;  $ST(0) = |x|$ 
    fstp QWORD PTR [ESP+28+0]    ; результат на место арг. x

pow_double_begin:
    fld QWORD PTR [ESP+28+8]    ;  $ST(0) = y$ 
```

```

fld QWORD PTR [ESP+28+0]      ; ST(0) = x, ST(1) = y
fyl2x                        ; ST(0) = y*log2(x)
fstp QWORD PTR [ESP+0+8]     ; [ESP+8] = y*log2(x)

fld QWORD PTR [ESP+0+8]      ; ST(0) = y*log2(x)
fld1                        ; ST(0) = 1.0, ST(1) = y*log2(x)
fscale                       ; ST(0) = 2^(целое)[y*log2(x)]
fstp QWORD PTR [ESP+0+0]     ; [ESP+0] = 2^(целое)[y*log2(x)]

fld1                        ; ST(0) = 1.0, ST(1) = 2^(целое)[y*log2(x)]
fld QWORD PTR [ESP+0+8]      ; ST(0) = y*log2(x), ST(1) = 1.0, ST(2) =
2^(целое)[y*log2(x)]
fprem                       ; ST(0) = (дробное)[y*log2(x)], ST(1) =
2^(целое)[y*log2(x)]

f2xm1                        ; ST(0) = 2^(дробное)[y*log2(x)] - 1, ST(1)
= 2^(целое)[y*log2(x)]
fld1                        ; ST(0) = 1.0, ST(1) =
2^(дробное)[y*log2(x)] - 1, ST(2) = 2^(целое)[y*log2(x)]
fadd                        ; ST(0) = 2^(дробное)[y*log2(x)], ST(1) =
2^(целое)[y*log2(x)]
fmul QWORD PTR [ESP+0+0]     ; ST(0) = 2^(y*log2(x))
fstp QWORD PTR [ESP+0+8]     ; [ESP+8] = 2^(y*log2(x))

cmp DWORD PTR [ESP+0+16], 1
jne pow_double_end          ; если x > 0, иначе:

mov DWORD PTR [ESP+0+20], 2   ; [ESP+20] = 2
fld DWORD PTR [ESP+0+20]     ; ST(0) = 2, ST(1) = 2^(y*log2(x))
fld QWORD PTR [ESP+28+8]     ; ST(0) = y, ST(1) = 2, ST(2) =
2^(y*log2(x))
fprem                       ; ST(0) = (дробное)[y/2]
fldz                        ; загрузка 0.0 в сопроцессор
db 0dbh, 0f0h+1             ; сравнение ST(0), ST(1)
je pow_double_end           ; если y == 0

fld QWORD PTR [ESP+0+8]      ; загрузка 2^(y*log2(x)) в сопроцессор
fchs
fstp QWORD PTR [ESP+0+8]     ; результат на место 2-й доп.слот

pow_double_end:
mov EAX, DWORD PTR [ESP+0+8]
mov EDX, DWORD PTR [ESP+0+12]
mov DWORD PTR [ESP+28+8], EAX
mov DWORD PTR [ESP+28+12], EDX

; Очищаем сопроцессор
fstp QWORD PTR [ESP+28+0]
fstp QWORD PTR [ESP+28+0]
fstp QWORD PTR [ESP+28+0]

```

```

fstp QWORD PTR [ESP+28+0]
fstp QWORD PTR [ESP+28+0]
fstp QWORD PTR [ESP+28+0]
fstp QWORD PTR [ESP+28+0]
fstp QWORD PTR [ESP+28+0]
fstp QWORD PTR [ESP+28+0]

ADD ESP, 24
ret 8
func_pow_double endp

```

```

2.000000 ^ 2.000000 = 4.000000
2.000000 ^ 32.000000 = 4294967296.000000
32.000000 ^ 2.000000 = 1024.000000
3.400000 ^ 2.600000 = 24.090465
0.000000 ^ 5.000000 = 0.000000
5.000000 ^ 0.000000 = 1.000000
-12.400000 ^ 2.600000 = -696.460325
2.000000 ^ -1.000000 = 0.500000
-3.000000 ^ -2.000000 = 0.111111
3234.400000 ^ 2234.600000 = 1.#INF00

```

2. Численно исследовать на сходимость ряд. Аргументы тригонометрических функций считать в радианах. Для возведения чисел в степень использовать написанную функцию pow. В отчёт включить текст программы и значения суммы ряда при n от 1 до 50. Вывести результат на экран в виде:

$n = 1; S = \dots$

$n = 2; S = \dots$

...

Вариант	Выражение
13	$S = \sum_{n=1}^{\infty} \frac{n^3 \cos(n^2)}{q^n}, \quad q = \sqrt[3]{5} + \sqrt[3]{7}$

```

mov EBP, 0 ; !n
loop_2:
    inc EBP

    sub ESP, 32 ; Локальные переменные
    mov DWORD PTR [ESP + 0 + 0], 0 ; [ESP+0]
    mov DWORD PTR [ESP + 0 + 4], 0
    mov DWORD PTR [ESP + 0 + 8], 0 ; [ESP+8]
    mov DWORD PTR [ESP + 0 + 12], 0
    mov DWORD PTR [ESP + 0 + 16], 0 ; [ESP+16]
    mov DWORD PTR [ESP + 0 + 20], 0
    mov DWORD PTR [ESP + 0 + 24], 0 ; [ESP+24]
    mov DWORD PTR [ESP + 0 + 28], 0

```

```

mov DWORD PTR [ESP + 0 + 0], 1          ; [ESP+0] = 1
fld QWORD PTR [ESP + 0 + 0]            ; ST(0) = 1f
mov DWORD PTR [ESP + 0 + 0], 3          ; [ESP+0] = 3
fld QWORD PTR [ESP + 0 + 0]            ; ST(0) = 3f, ST(1) = 1f
fdiv
fstp QWORD PTR [ESP + 0 + 8]           ; [ESP+8] = 1/3f

mov DWORD PTR [ESP + 0 + 0], 5          ; [ESP+0] = 5
fld QWORD PTR [ESP + 0 + 0]            ; ST(0) = 5f
mov DWORD PTR [ESP + 0 + 0], 1          ; [ESP+0] = 1
fld QWORD PTR [ESP + 0 + 0]            ; ST(0) = 1f, ST(1) = 5f
fdiv
fstp QWORD PTR [ESP + 0 + 0]           ; [ESP+0] = 5f
call func_pow_double                   ; [ESP+0] = pow(5, 1/3)

mov EBX, DWORD PTR [ESP + 0 + 0]        ; EBX = pow(5, 1/3)[0]
mov EDI, DWORD PTR [ESP + 0 + 0 + 4]    ; EDI = pow(5, 1/3)[4]

mov DWORD PTR [ESP + 0 + 0], 0
mov DWORD PTR [ESP + 0 + 4], 0
mov DWORD PTR [ESP + 0 + 8], 0
mov DWORD PTR [ESP + 0 + 12], 0

mov DWORD PTR [ESP + 0 + 0], 1          ; [ESP+0] = 1
fld QWORD PTR [ESP + 0 + 0]            ; [ESP+0] = 1f
mov DWORD PTR [ESP + 0 + 8], 3          ; [ESP+0] = 3
fld QWORD PTR [ESP + 0 + 8]            ; [ESP+0] = 3f, [ESP+8] = 1f
fdiv
fst QWORD PTR [ESP + 0 + 8]            ; [ESP+8] = 1/3f

mov DWORD PTR [ESP + 0 + 0], 7          ; [ESP+0] = 7
fld QWORD PTR [ESP + 0 + 0]            ; [ESP+0] = 7f
mov DWORD PTR [ESP + 0 + 0], 1          ; [ESP+0] = 1
fld QWORD PTR [ESP + 0 + 0]            ; [ESP+0] = 1f, [ESP+8] = 7f
fmul
fst QWORD PTR [ESP + 0 + 0]            ; [ESP+0] = 7f
call func_pow_double                   ; [ESP+0] = pow(7, 1/3)

mov DWORD PTR [ESP + 0 + 8], EBX        ; [ESP+8] = pow(5, 1/3)[0]
mov DWORD PTR [ESP + 0 + 8 + 4], EDI    ; [ESP+12] = pow(5, 1/3)[4]
; [ESP+8] = pow(5, 1/3)
fld QWORD PTR [ESP + 0 + 0]            ; [ESP+0] = pow(7, 1/3)
fld QWORD PTR [ESP + 0 + 8]            ; [ESP+8] = pow(5, 1/3)
fadd
fst QWORD PTR [ESP + 0 + 0]            ; [ESP+0] = pow(5,1/3)+pow(7,1/3)
; ![ESP+0] = q = const

mov DWORD PTR [ESP + 0 + 8], EBP        ; n

```

```

fild DWORD PTR [ESP + 0 + 8]          ; [ESP+8] = nf
mov  DWORD PTR [ESP + 0 + 16], 1      ; [ESP+16] = 1
fild DWORD PTR [ESP + 0 + 16]        ; [ESP+16] = 1f
fmul
fst  QWORD PTR [ESP + 0 + 8]          ; [ESP+8] = 1f
call func_pow_double                 ; ![ESP+0] = pow(q, n)

mov  EAX, DWORD PTR [ESP + 0 + 0]
mov  ESI, DWORD PTR [ESP + 0 + 0 + 4]

mov  DWORD PTR [ESP + 0 + 8], 3      ; 3
fild DWORD PTR [ESP + 0 + 8]        ; [ESP+8] = 3f
fst  QWORD PTR [ESP + 0 + 8]
mov  DWORD PTR [ESP + 0 + 0], EBP    ; n
fild DWORD PTR [ESP + 0 + 0]        ; [ESP+8] = nf
fst  QWORD PTR [ESP + 0 + 0]
call func_pow_double                 ; [ESP+0] = pow(n, 3)

mov  EBX, DWORD PTR [ESP + 0 + 0]    ; EBX = pow(n, 3)[0]
mov  EDI, DWORD PTR [ESP + 0 + 0 + 4] ; EDI = pow(n, 3)[4]

mov  DWORD PTR [ESP + 0 + 8], 2      ; 2
fild DWORD PTR [ESP + 0 + 8]        ; [ESP+8] = 2f
fst  QWORD PTR [ESP + 0 + 8]
mov  DWORD PTR [ESP + 0 + 0], EBP    ; n
fild DWORD PTR [ESP + 0 + 0]        ; [ESP+8] = nf
fst  QWORD PTR [ESP + 0 + 0]
call func_pow_double                 ; [ESP+0] = pow(n, 2)
fld  QWORD PTR [ESP + 0 + 0]
fcos
fst  QWORD PTR [ESP + 0 + 0]          ; [ESP+0] = cos(pow(n, 2))

mov  DWORD PTR [ESP + 0 + 8], EBX    ; [ESP+8] = pow(n, 3)[0]
mov  DWORD PTR [ESP + 0 + 8 + 4], EDI ; [ESP+12] = pow(n, 3)[4]
                                         ; [ESP+8] = pow(n, 3)

fld  QWORD PTR [ESP + 0 + 0]
fld  QWORD PTR [ESP + 0 + 8]
fmul
fst  QWORD PTR [ESP + 0 + 8]          ; ![ESP+8] = pow(n,3)*cos(pow(n,2))

mov  DWORD PTR [ESP + 0 + 0], EAX
mov  DWORD PTR [ESP + 0 + 0 + 4], ESI
fld  QWORD PTR [ESP + 0 + 8]
fld  QWORD PTR [ESP + 0 + 0]
fdiv
fst  QWORD PTR [ESP + 0 + 0]

push EBP
push offset row_out

```

```

call crt_printf
add esp, 4

fstp QWORD PTR [ESP+28+0]

cmp EBP, 50
jne loop_2

add ESP, 32 ; Чистим стек от переменных

```

n = 1; s = 0.149135	n = 26; s = -0.000000
n = 2; s = -0.398398	n = 27; s = 0.000000
n = 3; s = -0.517336	n = 28; s = 0.000000
n = 4; s = -0.355764	n = 29; s = 0.000000
n = 5; s = 0.198512	n = 30; s = 0.000000
n = 6; s = -0.012224	n = 31; s = 0.000000
n = 7; s = 0.012586	n = 32; s = 0.000000
n = 8; s = 0.006760	n = 33; s = -0.000000
n = 9; s = 0.005266	n = 34; s = 0.000000
n = 10; s = 0.002214	n = 35; s = 0.000000
n = 11; s = -0.000046	n = 36; s = -0.000000
n = 12; s = 0.000294	n = 37; s = 0.000000
n = 13; s = 0.000095	n = 38; s = 0.000000
n = 14; s = 0.000014	n = 39; s = 0.000000
n = 15; s = 0.000005	n = 40; s = -0.000000
n = 16; s = -0.000000	n = 41; s = -0.000000
n = 17; s = 0.000002	n = 42; s = -0.000000
n = 18; s = -0.000000	n = 43; s = -0.000000
n = 19; s = -0.000000	n = 44; s = 0.000000
n = 20; s = -0.000000	n = 45; s = -0.000000
n = 21; s = 0.000000	n = 46; s = 0.000000
n = 22; s = 0.000000	n = 47; s = -0.000000
n = 23; s = 0.000000	n = 48; s = -0.000000
n = 24; s = -0.000000	n = 49; s = 0.000000
n = 25; s = -0.000000	n = 50; s = 0.000000

Вывод: в ходе лабораторной работы мы изучили команды сопроцессора для выполнения арифметических операций.

Приложения

Приложение 1. Полный код программы

```
.386
.model flat, stdcall
option casemap: none

include include\windows.inc
include include\kernel32.inc
include include\user32.inc
include include\msvcrt.inc

includelib user32.lib
includelib kernel32.lib
includelib msvcrt.lib

nline MACRO
    LOCAL nline_fmt
    .data
        nline_fmt DB 13, 10, 0
    .code
        invoke crt_printf, offset nline_fmt
        add esp, 4
    ENDM

.data
    main_fmt DB "DBG: 0x%016llX", 13, 10, 0
    main_fmt_8 DB "DBG: 0x%08lX", 13, 10, 0
    main_fmt_f DB "DBG: %f", 13, 10, 0
.code

func_pow_double proc
    sub ESP, 24                ; доп.слоты в стеке
                                ;  $x^y = 2^{(y \cdot \log_2(x))}$ 

    mov DWORD PTR [ESP+0+0], 0
    mov DWORD PTR [ESP+0+4], 0
    mov DWORD PTR [ESP+0+8], 0
    mov DWORD PTR [ESP+0+12], 0
    mov DWORD PTR [ESP+0+16], 0
    mov DWORD PTR [ESP+0+20], 0

    fld QWORD PTR [ESP+28+0]    ; загрузка x в сопроцессор
    fldz                       ; загрузка 0.0 в сопроцессор
    db 0dbh, 0f0h+1            ; сравнение ST(0), ST(1)
    ja pow_double_x_less_0     ; если  $x < 0$ 
    jb pow_double_begin        ; если  $x > 0$ 

    fld QWORD PTR [ESP+28+8]    ; загрузка y в сопроцессор
    fldz                       ; загрузка 0.0 в сопроцессор
```


<code>db 0dbh, 0f0h+1</code>	<code>; сравнение ST(0), ST(1)</code>
<code>jne pow_double_end</code>	
<code>fld1</code>	<code>; загрузка 1.0 в сопроцессор</code>
<code>fstp QWORD PTR [ESP+0+8]</code>	<code>; результат в 2-й доп.слот</code>
<code>jmp pow_double_end</code>	
<code>pow_double_x_less_0:</code>	<code>; x < 0</code>
<code>mov DWORD PTR [ESP+0+16], 1</code>	<code>; 3-й доп.слот (4 из 8 байта) = 1</code>
<code>fld QWORD PTR [ESP+28+0]</code>	<code>; загрузка x в сопроцессор</code>
<code>fabs</code>	<code>; ST(0) = x </code>
<code>fstp QWORD PTR [ESP+28+0]</code>	<code>; результат на место арг. x</code>
<code>pow_double_begin:</code>	
<code>fld QWORD PTR [ESP+28+8]</code>	<code>; ST(0) = y</code>
<code>fld QWORD PTR [ESP+28+0]</code>	<code>; ST(0) = x, ST(1) = y</code>
<code>fyl2x</code>	<code>; ST(0) = $y \cdot \log_2(x)$</code>
<code>fstp QWORD PTR [ESP+0+8]</code>	<code>; [ESP+8] = $y \cdot \log_2(x)$</code>
<code>fld QWORD PTR [ESP+0+8]</code>	<code>; ST(0) = $y \cdot \log_2(x)$</code>
<code>fld1</code>	<code>; ST(0) = 1.0, ST(1) = $y \cdot \log_2(x)$</code>
<code>fscale</code>	<code>; ST(0) = $2^{(\text{целое})} [y \cdot \log_2(x)]$</code>
<code>fstp QWORD PTR [ESP+0+0]</code>	<code>; [ESP+0] = $2^{(\text{целое})} [y \cdot \log_2(x)]$</code>
<code>fld1</code>	<code>; ST(0) = 1.0, ST(1) = $2^{(\text{целое})} [y \cdot \log_2(x)]$</code>
<code>fld QWORD PTR [ESP+0+8]</code>	<code>; ST(0) = $y \cdot \log_2(x)$, ST(1) = 1.0, ST(2) =</code>
<code>2^(целое) [$y \cdot \log_2(x)$]</code>	
<code>fprem</code>	<code>; ST(0) = (дробное) [$y \cdot \log_2(x)$], ST(1) =</code>
<code>2^(целое) [$y \cdot \log_2(x)$]</code>	
<code>f2xm1</code>	<code>; ST(0) = $2^{(\text{дробное})} [y \cdot \log_2(x)] - 1$, ST(1)</code>
<code>= 2^(целое) [$y \cdot \log_2(x)$]</code>	
<code>fld1</code>	<code>; ST(0) = 1.0, ST(1) =</code>
<code>2^(дробное) [$y \cdot \log_2(x)$] - 1, ST(2) = 2^(целое) [$y \cdot \log_2(x)$]</code>	
<code>fadd</code>	<code>; ST(0) = $2^{(\text{дробное})} [y \cdot \log_2(x)]$, ST(1) =</code>
<code>2^(целое) [$y \cdot \log_2(x)$]</code>	
<code>fmul QWORD PTR [ESP+0+0]</code>	<code>; ST(0) = $2^{(y \cdot \log_2(x))}$</code>
<code>fstp QWORD PTR [ESP+0+8]</code>	<code>; [ESP+8] = $2^{(y \cdot \log_2(x))}$</code>
<code>cmp DWORD PTR [ESP+0+16], 1</code>	
<code>jne pow_double_end</code>	<code>; если x > 0, иначе:</code>
<code>mov DWORD PTR [ESP+0+20], 2</code>	<code>; [ESP+20] = 2</code>
<code>fld DWORD PTR [ESP+0+20]</code>	<code>; ST(0) = 2, ST(1) = $2^{(y \cdot \log_2(x))}$</code>
<code>fld QWORD PTR [ESP+28+8]</code>	<code>; ST(0) = y, ST(1) = 2, ST(2) =</code>
<code>2^{($y \cdot \log_2(x)$)}</code>	
<code>fprem</code>	<code>; ST(0) = (дробное) [$y/2$]</code>
<code>fldz</code>	<code>; загрузка 0.0 в сопроцессор</code>
<code>db 0dbh, 0f0h+1</code>	<code>; сравнение ST(0), ST(1)</code>
<code>je pow_double_end</code>	<code>; если y == 0</code>

```

    fld QWORD PTR [ESP+0+8]          ; загрузка  $2^{(y \cdot \log_2(x))}$  в сопроцессор
    fchs
    fstp QWORD PTR [ESP+0+8]        ; результат на место 2-й доп.слот

pow_double_end:
    mov EAX, DWORD PTR [ESP+0+8]
    mov EDX, DWORD PTR [ESP+0+12]
    mov DWORD PTR [ESP+28+8], EAX
    mov DWORD PTR [ESP+28+12], EDX

    ; Очищаем сопроцессор
    fstp QWORD PTR [ESP+28+0]
    fstp QWORD PTR [ESP+28+0]
    fstp QWORD PTR [ESP+28+0]
    fstp QWORD PTR [ESP+28+0]
    fstp QWORD PTR [ESP+28+0]
    fstp QWORD PTR [ESP+28+0]
    fstp QWORD PTR [ESP+28+0]
    fstp QWORD PTR [ESP+28+0]
    fstp QWORD PTR [ESP+28+0]

    ADD ESP, 24
    ret 8
func_pow_double endp

start:
.data
    x DQ 2.0, 2.0, 32.0, 3.4, 0.0, 5.0, -12.4, 2.0, -3.0, 3234.4
    y DQ 2.0, 32.0, 2.0, 2.6, 5.0, 0.0, 2.6, -1.0, -2.0, 2234.6
    q DQ ?
    pow_fmt_f1 DB "%f ^ %f = ", 0
    pow_fmt_f DB "%f", 10, 13, 0
    row_out DB "n = %d; S = %f", 10, 13, 0
.code

    mov EDI, 0
loop_1:
    mov EAX, 0

    push dword ptr y[EDI + 4]
    push dword ptr y[EDI]
    push dword ptr x[EDI + 4]
    push dword ptr x[EDI]
    push offset pow_fmt_f1
    call crt_printf
    add esp, 5*4

    push dword ptr y[EDI + 4]
    push dword ptr y[EDI]
    push dword ptr x[EDI + 4]

```

```

push dword ptr x[EDI]

fld y[EDI]                ; загрузка double в сопроцессор
sub ESP, 8                ; выделение памяти в стеке под double
fstp QWORD PTR [ESP]      ; вытолкнуть double в стек

fld x[EDI]                ; загрузка double в сопроцессор
sub ESP, 8                ; выделение памяти в стеке под double
fstp QWORD PTR [ESP]      ; вытолкнуть double в стек
call func_pow_double

push offset pow_fmt_f
call crt_printf
add esp, 4

add EDI, 8
cmp EDI, 80
jne loop_1

nline

mov EBP, 0                ; !n
loop_2:
inc EBP

sub ESP, 32               ; Локальные переменные
mov DWORD PTR [ESP + 0 + 0], 0 ; [ESP+0]
mov DWORD PTR [ESP + 0 + 4], 0
mov DWORD PTR [ESP + 0 + 8], 0 ; [ESP+8]
mov DWORD PTR [ESP + 0 + 12], 0
mov DWORD PTR [ESP + 0 + 16], 0 ; [ESP+16]
mov DWORD PTR [ESP + 0 + 20], 0
mov DWORD PTR [ESP + 0 + 24], 0 ; [ESP+24]
mov DWORD PTR [ESP + 0 + 28], 0

mov DWORD PTR [ESP + 0 + 0], 1 ; [ESP+0] = 1
fld QWORD PTR [ESP + 0 + 0]    ; ST(0) = 1f
mov DWORD PTR [ESP + 0 + 0], 3 ; [ESP+0] = 3
fld QWORD PTR [ESP + 0 + 0]    ; ST(0) = 3f, ST(1) = 1f
fdiv
fstp QWORD PTR [ESP + 0 + 8]    ; [ESP+8] = 1/3f

mov DWORD PTR [ESP + 0 + 0], 5 ; [ESP+0] = 5
fld QWORD PTR [ESP + 0 + 0]    ; ST(0) = 5f
mov DWORD PTR [ESP + 0 + 0], 1 ; [ESP+0] = 1
fld QWORD PTR [ESP + 0 + 0]    ; ST(0) = 1f, ST(1) = 5f
fdiv
fstp QWORD PTR [ESP + 0 + 0]    ; [ESP+0] = 5f
call func_pow_double           ; [ESP+0] = pow(5, 1/3)

```

```

mov EBX, DWORD PTR [ESP + 0 + 0] ; EBX = pow(5, 1/3)[0]
mov EDI, DWORD PTR [ESP + 0 + 0 + 4] ; EDI = pow(5, 1/3)[4]

mov DWORD PTR [ESP + 0 + 0], 0
mov DWORD PTR [ESP + 0 + 4], 0
mov DWORD PTR [ESP + 0 + 8], 0
mov DWORD PTR [ESP + 0 + 12], 0

mov DWORD PTR [ESP + 0 + 0], 1 ; [ESP+0] = 1
fld QWORD PTR [ESP + 0 + 0] ; [ESP+0] = 1f
mov DWORD PTR [ESP + 0 + 8], 3 ; [ESP+0] = 3
fld QWORD PTR [ESP + 0 + 8] ; [ESP+0] = 3f, [ESP+8] = 1f
fdiv
fst QWORD PTR [ESP + 0 + 8] ; [ESP+8] = 1/3f

mov DWORD PTR [ESP + 0 + 0], 7 ; [ESP+0] = 7
fild QWORD PTR [ESP + 0 + 0] ; [ESP+0] = 7f
mov DWORD PTR [ESP + 0 + 0], 1 ; [ESP+0] = 1
fild QWORD PTR [ESP + 0 + 0] ; [ESP+0] = 1f, [ESP+8] = 7f
fmul
fst QWORD PTR [ESP + 0 + 0] ; [ESP+0] = 7f
call func_pow_double ; [ESP+0] = pow(7, 1/3)

mov DWORD PTR [ESP + 0 + 8], EBX ; [ESP+8] = pow(5, 1/3)[0]
mov DWORD PTR [ESP + 0 + 8 + 4], EDI ; [ESP+12] = pow(5, 1/3)[4]
; [ESP+8] = pow(5, 1/3)
fld QWORD PTR [ESP + 0 + 0] ; [ESP+0] = pow(7, 1/3)
fld QWORD PTR [ESP + 0 + 8] ; [ESP+8] = pow(5, 1/3)
fadd
fst QWORD PTR [ESP + 0 + 0] ; [ESP+0] = pow(5,1/3)+pow(7,1/3)
; ![ESP+0] = q = const

mov DWORD PTR [ESP + 0 + 8], EBP ; n
fild DWORD PTR [ESP + 0 + 8] ; [ESP+8] = nf
mov DWORD PTR [ESP + 0 + 16], 1 ; [ESP+16] = 1
fild DWORD PTR [ESP + 0 + 16] ; [ESP+16] = 1f
fmul
fst QWORD PTR [ESP + 0 + 8] ; [ESP+8] = 1f
call func_pow_double ; ![ESP+0] = pow(q, n)

mov EAX, DWORD PTR [ESP + 0 + 0]
mov ESI, DWORD PTR [ESP + 0 + 0 + 4]

mov DWORD PTR [ESP + 0 + 8], 3 ; 3
fild DWORD PTR [ESP + 0 + 8] ; [ESP+8] = 3f
fst QWORD PTR [ESP + 0 + 8]
mov DWORD PTR [ESP + 0 + 0], EBP ; n
fild DWORD PTR [ESP + 0 + 0] ; [ESP+8] = nf

```

```

fst QWORD PTR [ESP + 0 + 0]
call func_pow_double                ; [ESP+0] = pow(n, 3)

mov EBX, DWORD PTR [ESP + 0 + 0]    ; EBX = pow(n, 3)[0]
mov EDI, DWORD PTR [ESP + 0 + 0 + 4] ; EDI = pow(n, 3)[4]

mov DWORD PTR [ESP + 0 + 8], 2      ; 2
fild DWORD PTR [ESP + 0 + 8]        ; [ESP+8] = 2f
fst QWORD PTR [ESP + 0 + 8]
mov DWORD PTR [ESP + 0 + 0], EBP    ; n
fild DWORD PTR [ESP + 0 + 0]        ; [ESP+8] = nf
fst QWORD PTR [ESP + 0 + 0]
call func_pow_double                ; [ESP+0] = pow(n, 2)
fld QWORD PTR [ESP + 0 + 0]
fcos
fst QWORD PTR [ESP + 0 + 0]         ; [ESP+0] = cos(pow(n, 2))

mov DWORD PTR [ESP + 0 + 8], EBX    ; [ESP+8] = pow(n, 3)[0]
mov DWORD PTR [ESP + 0 + 8 + 4], EDI ; [ESP+12] = pow(n, 3)[4]
                                     ; [ESP+8] = pow(n, 3)

fld QWORD PTR [ESP + 0 + 0]
fld QWORD PTR [ESP + 0 + 8]
fmul
fst QWORD PTR [ESP + 0 + 8]         ; ![ESP+8] = pow(n,3)*cos(pow(n,2))

mov DWORD PTR [ESP + 0 + 0], EAX
mov DWORD PTR [ESP + 0 + 0 + 4], ESI
fld QWORD PTR [ESP + 0 + 8]
fld QWORD PTR [ESP + 0 + 0]
fddiv
fst QWORD PTR [ESP + 0 + 0]

push EBP
push offset row_out
call crt_printf
add esp, 4

fstp QWORD PTR [ESP+28+0]

cmp EBP, 50
jne loop_2

add ESP, 32                        ; Чистим стек от переменных

invoke ExitProcess, 0              ; Выход из программы
end start

```