**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\*log2(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

        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

        fild 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]

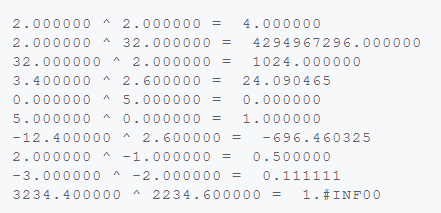
        fstp QWORD PTR [ESP+28+0]

        fstp QWORD PTR [ESP+28+0]

        ADD ESP, 24

        ret 8

func\_pow\_double endp



1. 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 = …

n = 2; S = …

…

|  |  |
| --- | --- |
| Вариант | Выражение |
| 13 |  |

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]

    fdiv

    fst QWORD PTR [ESP + 0 + 0]

    push EBP

    push offset row\_out

    call crt\_printf

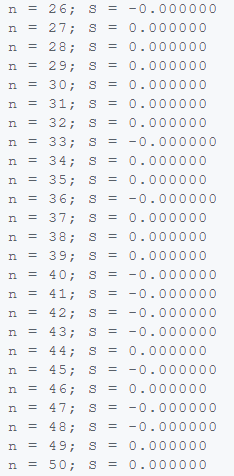
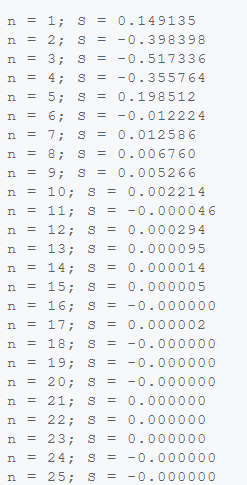
    add esp, 4

    fstp QWORD PTR [ESP+28+0]

    cmp EBP, 50

    jne loop\_2

    add ESP, 32                             ; Чистим стек от переменных



**Вывод**: в ходе лабораторной работы мы изученили 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\*log2(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

        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

        fild 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

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]

    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

    invoke ExitProcess, 0                   ; Выход из программы

end start