МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №5

по дисциплине: Архитектура вычислительных систем
тема: «Команды сопроцессора»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
ст. пр. Осипов Олег Васильевич

Белгород 2024 г.

Лабораторная работа №5
Команды передачи управления
Вариант 8

Цель работы: изучение команд сопроцессора для выполнения арифметических операций.

Задания для выполнения к работе:

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. В отчёт включить текст программы, блок-схему алгоритма функции pow и набор тестовых данных. Функция pow должна удовлетворять соглашениям о вызовах. Аргументы должны передаваться ей через стек.
2. Составить таблицу состояния регистров для всех вещественных операций: для начальных операций перед циклом, для первой и последней итераций, для завершающих операций после цикла.
3. 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 = …
   …
   Убедиться, что результаты вычислений совпадают с аналогичной программой на языке Python.

Задание:

$$S = \sum_{n=1}^{\infty} \operatorname{arctg}\left(\frac{1}{n}\right) + \frac{n-1}{n^2+q^n},\ q = 15$$

Первая программа:

```
.686
.model flat, stdcall
option casemap: none

include windows.inc
include kernel32.inc
include msvcrt.inc
includelib  kernel32.lib
includelib  msvcrt.lib

; Здесь Бога нет

.data
    x dd 0.00001
    y dd 0.00001
    print_val db "%.6f", 13, 10, 0
.code

; Осторожно! В стеке для FPU должно быть свободно 4 элемента для вычислений.
; pow (float x, float y)
pow proc
    pushad

    fld dword ptr [esp + 8 + 8 * 4] ; y получаем из параметров. S(0) = y
    fabs ; y = |y|
    fld dword ptr [esp + 4 + 8 * 4] ; x получаем из параметров. S(0) = x, S(1) = y
    fabs ; x = |x|

    ; Вычислим t = ylog_2(x)
    fyl2x ; S(0) = ylog_2(x)
    fxam
    fstsw ax
    sahf
    jz  pow_ok
    jpe pow_C2is1    ; Если C2 = 1
    jc  pow_is_nan   ; Если получили isNan, то x и y = 0, а значит нужно вернуть 1.
    jmp pow_ok

    pow_C2is1:
    jc    pow_infinity
    jmp pow_ok

    pow_is_nan:
    ; У нас infinity может случиться только если у нас x = 0. В этом случае надо бы возвращать 0.
    ffree ST(0)
    fincstp
    fld1
    popad
    ret 8

    pow_infinity:
    ; У нас infinity может случиться только если у нас x = 0. В этом случае надо бы возвращать 0.
    ffree ST(0)
    fincstp
    fldz
    popad
    ret 8

pow_ok:

    fld1 ; S(0) = 1; S(1) = ylog_2(x)
    fscale ; S(0) = S(0) * 2 ^ S(1); S(1) = ylog_2(x)

    fld1 ; S(0) = 1; S(1) = 2 ^ a; S(2) = ylog_2(x)
```

```
    fld ST(2) ; S(0) = ylog_2(x); S(1) = 1; S(2) = 2 ^ a; S(3) = ylog_2(x)
    fprem ; S(0) = ylog_2(x) % 1 = b; S(1) = 1; S(2) = 2 ^ a; S(3) = ylog_2(x)
    f2xm1 ; S(0) = 2 ^ b - 1; S(1) = 1; S(2) = 2 ^ a; S(3) = ylog_2(x)
    faddp ST(1), ST(0) ; S(0) = 2 ^ b; S(1) = 2 ^ a; S(2) = ylog_2(x)
    fmulp ST(1), ST(0) ; S(0) = 2 ^ a * 2 ^ b; S(1) = ylog_2(x)

    fldz                            ; ST(0) = 0; S(1) = 2 ^ a * 2 ^ b; S(2) = ylog_2(x)
    fld dword ptr [esp + 4 + 8 * 4] ; ST(0) = x, ST(1) = 0; S(2) = 2 ^ a * 2 ^ b; S(3) =
ylog_2(x)
    fcompp   ; Сравнение x с 0 и выталкивание переменных из стека
    fstsw ax ; Загрузка swr в ax
    sahf     ; Загрузка ah в eflags. Теперь можем сравнивать.
    fld1     ; S(0) = sign_x
    jb pow_x_less_zero
    jmp pow_x_end
pow_x_less_zero:
    fchs   ; S(0) = -S(0) = sign_x
pow_x_end:

    ; ST(0) = sign_x; S(1) = 2 ^ a * 2 ^ b; S(2) = ylog_2(x)

    fmulp ST(1), ST(0) ; S(0) = sign_x * 2 ^ a * 2 ^ b; S(1) = ylog_2x
    fxch               ; S(0) = ylog_2x; S(1) = sign_x * 2 ^ a * 2 ^ b
    ffree ST(0)
    fincstp            ; S(0) = sign_x * 2 ^ a * 2 ^ b
    fldz                            ; S(0) = 0; S(1) = sign_x * 2 ^ a * 2 ^ b
    fld dword ptr [esp + 8 + 8 * 4] ; S(0) = y; S(1) = 0; S(2) = sign_x * 2 ^ a * 2 ^ b
    fcompp ; S(0) = sign_x * 2 ^ a * 2 ^ b
    fstsw ax ; Загрузка swr в ax
    sahf     ; Загрузка ah в eflags. Теперь можем сравнивать.
    jb pow_y_less_zero
    jmp pow_y_end
pow_y_less_zero:
    fld1                 ; S(0) = 1; S(1) = x ^ |y|
    fdivrp ST(1), ST(0)  ; S(0) = 1 / x ^ |y| = x ^ y.
pow_y_end:

    popad
    ret 8
pow endp

start:
    finit
    push y
    push x
    call pow

    sub esp, 8
    fstp qword ptr [esp]
    push offset print_val
    call crt_printf
    add esp, 12

    call crt__getch     ; Задержка ввода, getch()
    ; Вызов функции ExitProcess(0)
    push 0      ; Поместить аргумент функции в стек
    call ExitProcess    ; Выход из программы
end start
```

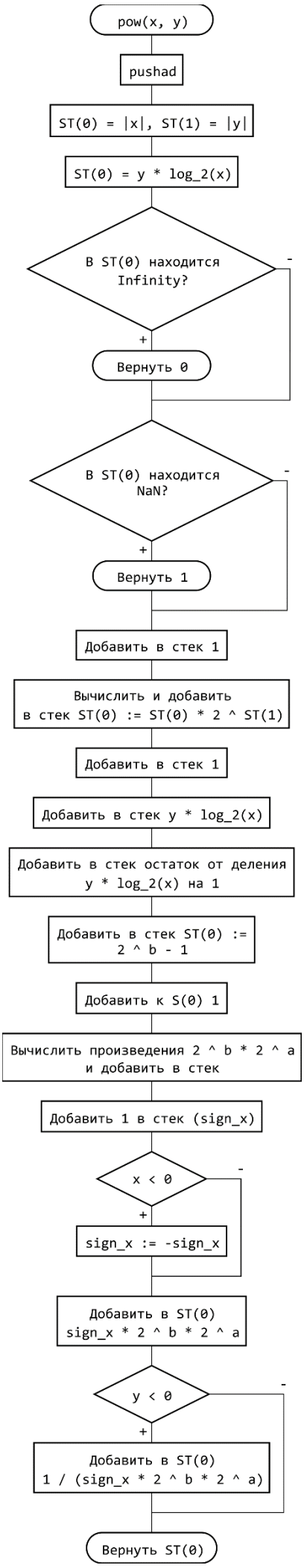
Тестовые данные:

x	y	Результат
9	1.5	27
-9	1.5	-27
9	-1.5	0.037037037037

-9	-1.5	-0.037037037037
0	1.5	0
9	0	1
123.2928	0.3212312	4.695419
-32921	-0.3211	-0.035436
0	0	1
0.00001	0.00001	0.999885

Результаты выполнения программы:

```
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
27.000000
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
-27.000000
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
0.037037
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
-0.037037
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
0.000000
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
1.000000
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
4.695420
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
-0.035436
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
1.000000
C:\Users\vladi\Workspace\Assembler\computing_systems_architecture\lab5>task1.exe
0.999885
```


pow(x, y)
pushad
ST(0) = |x|, ST(1) = |y|
ST(0) = y * log_2(x)
В ST(0) находится Infinity?
-
+
Вернуть 0
В ST(0) находится NaN?
-
+
Вернуть 1
Добавить в стек 1
Вычислить и добавить в стек ST(0) := ST(0) * 2 ^ ST(1)
Добавить в стек 1
Добавить в стек y * log_2(x)
Добавить в стек остаток от деления y * log_2(x) на 1
Добавить в стек ST(0) := 2 ^ b - 1
Добавить к S(0) 1
Вычислить произведения 2 ^ b * 2 ^ a и добавить в стек
Добавить 1 в стек (sign_x)
x < 0
-
+
sign_x := -sign_x
Добавить в ST(0) sign_x * 2 ^ b * 2 ^ a
y < 0
-
+
Добавить в ST(0) 1 / (sign_x * 2 ^ b * 2 ^ a)
Вернуть ST(0)

Вторая программа:

```
.686
.model flat, stdcall
option casemap: none

include windows.inc
include kernel32.inc
include msvcrt.inc
includelib  kernel32.lib
includelib  msvcrt.lib

; Здесь Бога нет тем более

.data
    q dd 15.0
    num_pow dd 2.0
    print_step db "S = %.6f, n = %.6f", 13, 10, 0
    print_fmt db "%.6f", 13, 10, 0
.code

; Осторожно! В стеке для FPU должно быть свободно 4 элемента для вычислений.
; pow (float x, float y)
pow proc
    pushad

    fld dword ptr [esp + 8 + 8 * 4] ; y получаем из параметров. S(0) = y
    fabs ; y = |y|
    fld dword ptr [esp + 4 + 8 * 4] ; x получаем из параметров. S(0) = x, S(1) = y
    fabs ; x = |x|

    ; Вычислим t = ylog_2(x)
    fyl2x ; S(0) = ylog_2(x)
    fxam
    fstsw ax
    sahf
    jz  pow_ok
    jpe pow_C2is1    ; Если C2 = 1
    jc  pow_is_nan   ; Если получили isNan, то x и y = 0, а значит нужно вернуть 1.
    jmp pow_ok

    pow_C2is1:
    jc    pow_infinity
    jmp pow_ok

    pow_is_nan:
    ; У нас infinity может случиться только если у нас x = 0. В этом случае надо бы возвращать 0.
    ffree ST(0)
    fincstp
    fld1
    popad
    ret 8

    pow_infinity:
    ; У нас infinity может случиться только если у нас x = 0. В этом случае надо бы возвращать 0.
    ffree ST(0)
    fincstp
    fldz
    popad
    ret 8

pow_ok:

    fld1 ; S(0) = 1; S(1) = ylog_2(x)
    fscale ; S(0) = S(0) * 2 ^ S(1); S(1) = ylog_2(x)
```

```
    fld1 ; S(0) = 1; S(1) = 2 ^ a; S(2) = ylog_2(x)
    fld ST(2) ; S(0) = ylog_2(x); S(1) = 1; S(2) = 2 ^ a; S(3) = ylog_2(x)
    fprem ; S(0) = ylog_2(x) % 1 = b; S(1) = 1; S(2) = 2 ^ a; S(3) = ylog_2(x)
    f2xm1 ; S(0) = 2 ^ b - 1; S(1) = 1; S(2) = 2 ^ a; S(3) = ylog_2(x)
    faddp ST(1), ST(0) ; S(0) = 2 ^ b; S(1) = 2 ^ a; S(2) = ylog_2(x)
    fmulp ST(1), ST(0) ; S(0) = 2 ^ a * 2 ^ b; S(1) = ylog_2(x)

    fldz                            ; ST(0) = 0; S(1) = 2 ^ a * 2 ^ b; S(2) = ylog_2(x)
    fld dword ptr [esp + 4 + 8 * 4] ; ST(0) = x, ST(1) = 0; S(2) = 2 ^ a * 2 ^ b; S(3) =
ylog_2(x)
    fcompp   ; Сравнение x с 0 и выталкивание переменных из стека
    fstsw ax ; Загрузка swr в ax
    sahf     ; Загрузка ah в eflags. Теперь можем сравнивать.
    fld1     ; S(0) = sign_x
    jb pow_x_less_zero
    jmp pow_x_end
pow_x_less_zero:
    fchs   ; S(0) = -S(0) = sign_x
pow_x_end:

    ; ST(0) = sign_x; S(1) = 2 ^ a * 2 ^ b; S(2) = ylog_2(x)

    fmulp ST(1), ST(0) ; S(0) = sign_x * 2 ^ a * 2 ^ b; S(1) = ylog_2x
    fxch               ; S(0) = ylog_2x; S(1) = sign_x * 2 ^ a * 2 ^ b
    ffree ST(0)
    fincstp            ; S(0) = sign_x * 2 ^ a * 2 ^ b
    fldz                            ; S(0) = 0; S(1) = sign_x * 2 ^ a * 2 ^ b
    fld dword ptr [esp + 8 + 8 * 4] ; S(0) = y; S(1) = 0; S(2) = sign_x * 2 ^ a * 2 ^ b
    fcompp ; S(0) = sign_x * 2 ^ a * 2 ^ b
    fstsw ax ; Загрузка swr в ax
    sahf     ; Загрузка ah в eflags. Теперь можем сравнивать.
    jb pow_y_less_zero
    jmp pow_y_end
pow_y_less_zero:
    fld1                 ; S(0) = 1; S(1) = x ^ |y|
    fdivrp ST(1), ST(0)  ; S(0) = 1 / x ^ |y| = x ^ y.
pow_y_end:

    popad
    ret 8
pow endp

start:
    finit ; Инициализация сопроцессора
    fld1 ; S(1) = n = 1
    fldz ; S(0) = 0 = S (1)

    mov ecx, 50

    cycle:
        sub esp, 16
        fst qword ptr [esp]
        fxch
        fst qword ptr [esp + 8]
        fxch
        mov esi, ecx
        push offset print_step
        call crt_printf
        add esp, 20
        mov ecx, esi

        fld1                ; ST(0) = 1, ST(1) = S, ST(2) = n
        fld ST(2)           ; ST(0) = n, ST(1) = 1, ST(2) = S, ST(3) = n
        fpatan              ; ST(0) = atan(1 / n), ST(1) = S, ST(2) = n
        faddp ST(1), ST(0)  ; ST(0) = S + atan(1 / n) = S, ST(1) = n
        fld ST(1)           ; ST(0) = n, ST(1) = S + atan(1 / n) = S, ST(2) = n
        sub esp, 8
```

```
        mov eax, dword ptr num_pow
        mov ebx, dword ptr q
        fstp dword ptr [esp] ; ST(0) = S + atan(1 / n) = S, ST(1) = n
        mov dword ptr [esp + 4], eax
        call pow             ; ST(0) = n ^ 2, ST(1) = S + atan(1 / n) = S, ST(2) = n
        fld ST(2)
        sub esp, 8
        mov dword ptr [esp], ebx
        fstp dword ptr [esp + 4]
        call pow             ; ST(0) = q^2, ST(1) = n ^ 2, ST(2) = S + atan(1 / n) = S, ST(3) = n
        faddp ST(1), ST(0)   ; ST(0) = n ^ 2 + q ^ 2, ST(1) = S + atan(1 / n) = S, ST(2) = n

        fld1                 ; ST(0) = 1, ST(1) = n ^ 2 + q ^ 2, ST(2) = S + atan(1 / n) = S,
ST(3) = n
        fld ST(3)            ; ST(0) = n, ST(1) = 1, ST(2) = n ^ 2 + q ^ 2, ST(3) = S + atan(1 /
n) = S, ST(4) = n
        fsubrp ST(1), ST(0)  ; ST(0) = n - 1, ST(1) = n ^ 2 + q ^ 2, ST(2) = S + atan(1 / n) = S,
ST(3) = n
        fdivrp ST(1), ST(0)  ; ST(0) = (n - 1) / (n ^ 2 + q ^ 2), ST(1) = S + atan(1 / n) = s,
ST(2) = n
        faddp ST(1), ST(0)   ; ST(0) = S + atan(1 / n) + (n - 1) / (n ^ 2 + q ^ 2), ST(1) = n
        fxch                 ; ST(0) = n, ST(1) = S + atan(1 / n) + (n - 1) / (n ^ 2 + q ^ 2)
        fld1                 ; ST(0) = 1, ST(1) = n, ST(2) = S + atan(1 / n) + (n - 1) / (n ^ 2 +
q ^ 2)
        faddp ST(1), ST(0)   ; ST(0) = n + 1, ST(1) = S + atan(1 / n) + (n - 1) / (n ^ 2 + q ^ 2)
        fxch                 ; ST(0) = S, ST(1) = n + 1
    loop cycle

    fxch
    ffree ST(0)
    fincstp

    sub esp, 8
    fstp qword ptr [esp]
    push offset print_fmt
    call crt_printf
    add esp, 8

    call crt__getch     ; Задержка ввода, getch()
    ; Вызов функции ExitProcess(0)
    push 0      ; Поместить аргумент функции в стек
    call ExitProcess    ; Выход из программы
end start
```

№	Команда		ST(0)	ST(1)	ST(2)	ST(3)	ST(4)	ST(5)	ST(6)	ST(7)
Инициализация регистров:										
1	fld1		1 (n)							
2	fldz		0 (S)	1 (n)						
Первая итерация:										
3	fld1		1	0 (S)	1 (n)					
4	fld ST(2)		1 (n)	1	0 (S)	1 (n)				
5	fpatan	←	0.785398	0 (S)	1 (n)					
6	faddp ST(1), ST(0)	←	0.785398 (S)	1 (n)						
7	fld st(0), st(1)		1 (n)	0.785398 (S)	1 (n)					
8	fstp esp, st(0)	←	0.785398 (S)	1 (n)						
9	call pow		n ^ 2	0.785398 (S)	1 (n)					
10	fld st(0), st(2)		1 (n)	1 (n ^ 2)	0.785398 (S)	1 (n)				
11	fstp esp + 4, st(0)	←	1 (n ^ 2)	0.785398 (S)	1 (n)					
12	call pow		15 (q ^ n)	1 (n ^ 2)	0.785398 (S)	1 (n)				
13	faddp	←	16 (q^n + n^2)	0.785398 (S)	1 (n)					
14	fld1		1	16 (q^n + n^2)	0.785398 (S)	1 (n)				
15	fld st(0), st(3)		1 (n)	1	16 (q^n + n^2)	0.785398 (S)	1 (n)			
16	fsubrp	←	0 (n – 1)	16 (q^n + n^2)	0.785398 (S)	1 (n)				
17	fdivrp st(1), st(0)	←	0 (n – 1) / (q^n + n^2)	0.785398 (S)	1 (n)					
18	faddp st(1), st(0)		0.785398 (S)	1 (n)						
19	fxch		1(n)	0.785398 (S)						
20	fld1		1	1(n)	0.785398 (S)					
21	faddp ST(1), ST(0)	←	2 (n + 1)	0.785398 (S)						
22	fxch		0.785398 (S)	2 (n + 1)						
Вторая итерация:										
3	fld1		1	0.785398 (S)	2 (n)					
4	fld ST(2)		2 (n)	1	0.785398 (S)	2 (n)				
5	fpatan	←	0.46364760	0.785398 (S)	2 (n)					
6	faddp ST(1), ST(0)	←	1.249046 (S)	2 (n)						
7	fld st(0), st(1)		2 (n)	1.249046 (S)	2 (n)					
8	fstp esp, st(0)	←	1.249046 (S)	2 (n)						
9	call pow		n ^ 2	1.249046 (S)	2 (n)					
10	fld st(0), st(2)		2 (n)	4 (n ^ 2)	1.249046 (S)	2 (n)				
11	fstp esp + 4, st(0)	←	4 (n ^ 2)	1.249046 (S)	2 (n)					
12	call pow		225 (q ^ n)	4 (n ^ 2)	1.249046 (S)	2 (n)				
13	faddp	←	229 (q^n + n^2)	1.249046 (S)	2 (n)					
14	fld1		1	229 (q^n + n^2)	1.249046 (S)	2 (n)				
15	fld st(0), st(3)		2 (n)	1	229 (q^2 + n^2)	1.249046 (S)	2 (n)			
16	fsubrp	←	1 (n – 1)	229 (q^n + n^2)	1.249046 (S)	2 (n)				
17	fdivrp st(1), st(0)	←	0.00436681 (n – 1) / (q^n + n^2)	1.249046 (S)	2 (n)					
18	faddp st(1), st(0)		1.25341258 (S)	2 (n)						
19	fxch		2 (n)	1.25341258 (S)						
20	fld1		1	2 (n)	1.25341258 (S)					

21	faddp ST(1), ST(0)	←	3 (n + 1)	1.25341258 (S)						
22	fxch		1.25341258 (S)	3 (n + 1)						
Последняя итерация (*n*=50)										
3	fld1		1	4.2087208 (S)	50 (n)					
4	fld ST(2)		50 (n)	1	4.2087208 (S)	50 (n)				
5	fpatan	←	0.019997	4.2087208 (S)	50 (n)					
6	faddp ST(1), ST(0)	←	4.228711 (S)	50 (n)						
7	fld st(0), st(1)		50 (n)	4.228711 (S)	50 (n)					
8	fstp esp, st(0)	←	4.228711 (S)	50 (n)						
9	call pow		n ^ 2	4.228711 (S)	50 (n)					
10	fld st(0), st(2)		50 (n)	2500 (n ^ 2)	4.228711 (S)	50 (n)				
11	fstp esp + 4, st(0)	←	2500 (n ^ 2)	4.228711 (S)	50 (n)					
12	call pow		6.376215E+58 (q ^ n)	2500 (n ^ 2)	4.228711 (S)	50 (n)				
13	faddp	←	6.376215E+58 (q^n + n^2)	4.228711 (S)	50 (n)					
14	fld1		1	6.376215E+58 (q^n + n^2)	4.228711 (S)	50 (n)				
15	fld st(0), st(3)		50 (n)	1	6.376215E+58 (q^n + n^2)	4.228711 (S)	50 (n)			
16	fsubrp	←	49 (n – 1)	6.376215E+58 (q^n + n^2)	4.228711 (S)	50 (n)				
17	fdivrp st(1), st(0)	←	7.68481E-58 (n – 1) / (q^2 + n^2)	4.228711 (S)	50 (n)					
18	faddp st(1), st(0)		4.228711 (S)	50 (n)						
19	fxch		50 (n)	4.228711 (S)						
20	fld1		1	50 (n)	4.228711 (S)					
21	faddp ST(1), ST(0)	←	51 (n + 1)	4.228711 (S)						
22	fxch		4.228711 (S)	51 (n + 1)						
Заключительные операции:										
23	fxch		51 (n)	4.228711 (S)						
24	ffree ST(0)	←		0.692647 (S)						
25	fincstp		0.692647 (S)							
26	fstp DWORD PTR [ESP]	←								

Вывод программы на ассемблере:

```
S = 0.000000, n = 1.000000
S = 0.785398, n = 2.000000
S = 1.253413, n = 3.000000
S = 1.575754, n = 4.000000
S = 1.820792, n = 5.000000
S = 2.018193, n = 6.000000
S = 2.183342, n = 7.000000
S = 2.325239, n = 8.000000
S = 2.449594, n = 9.000000
S = 2.560251, n = 10.000000
S = 2.659920, n = 11.000000
S = 2.750580, n = 12.000000
S = 2.833721, n = 13.000000
S = 2.910493, n = 14.000000
S = 2.981800, n = 15.000000
S = 3.048369, n = 16.000000
S = 3.110787, n = 17.000000
S = 3.169543, n = 18.000000
S = 3.225042, n = 19.000000
S = 3.277625, n = 20.000000
S = 3.327583, n = 21.000000
```

```
S = 3.375166, n = 22.000000
S = 3.420590, n = 23.000000
S = 3.464040, n = 24.000000
S = 3.505683, n = 25.000000
S = 3.545662, n = 26.000000
S = 3.584104, n = 27.000000
S = 3.621124, n = 28.000000
S = 3.656824, n = 29.000000
S = 3.691293, n = 30.000000
S = 3.724614, n = 31.000000
S = 3.756861, n = 32.000000
S = 3.788100, n = 33.000000
S = 3.818394, n = 34.000000
S = 3.847797, n = 35.000000
S = 3.876361, n = 36.000000
S = 3.904132, n = 37.000000
S = 3.931152, n = 38.000000
S = 3.957462, n = 39.000000
S = 3.983097, n = 40.000000
S = 4.008092, n = 41.000000
S = 4.032477, n = 42.000000
S = 4.056283, n = 43.000000
S = 4.079534, n = 44.000000
S = 4.102258, n = 45.000000
S = 4.124476, n = 46.000000
S = 4.146212, n = 47.000000
S = 4.167485, n = 48.000000
S = 4.188315, n = 49.000000
S = 4.208721, n = 50.000000
4.228718
```

Вывод аналогичной программы на Питоне:

```python
from math import atan

q = 15
S = 0

for n in range(1, 50):
    print(f"S = {S}, n = {n}")
    S += atan(1 / n) + (n - 1) / (n ** 2 + q ** n)
```

```
S = 0, n = 1
S = 0.7853981633974483, n = 2
S = 1.2534125846253286, n = 3
S = 1.5757541555704342, n = 4
S = 1.8207920592336226, n = 5
S = 2.018192886399806, n = 6
S = 2.1833420027705213, n = 7
S = 2.325239092491273, n = 8
S = 2.4495940897693256, n = 9
S = 2.5602513111513194, n = 10
S = 2.659919963658089, n = 11
S = 2.75057985085999, n = 12
S = 2.833721082748516, n = 13
S = 2.9104929740183003, n = 14
S = 2.981800438803591, n = 15
S = 3.0483686025794148, n = 16
S = 3.1107874125753723, n = 17
S = 3.169543235291095, n = 18
S = 3.225041740536812, n = 19
S = 3.2776248021477534, n = 20
S = 3.3275831978696964, n = 21
S = 3.3751663011466797, n = 22
S = 3.4205895805682567, n = 23
```

```
S = 3.4640404759597874, n = 24
S = 3.505683055058376, n = 25
S = 3.545661742181666, n = 26
S = 3.5841043322028536, n = 27
S = 3.6211244480767837, n = 28
S = 3.6568235607561075, n = 29
S = 3.6912926617556154, n = 30
S = 3.7246136576338627, n = 31
S = 3.7568605400691166, n = 32
S = 3.7881003734993848, n = 33
S = 3.8183941334181597, n = 34
S = 3.8477974216221646, n = 35
S = 3.8763610794609247, n = 36
S = 3.9041317160543456, n = 37
S = 3.9311521652416106, n = 38
S = 3.957461882494533, n = 39
S = 3.9830972910162106, n = 40
S = 4.0080920846351304, n = 41
S = 4.032477493807849, n = 42
S = 4.0562825199929184, n = 43
S = 4.079534142803381, n = 44
S = 4.102257503645022, n = 45
S = 4.124476068971742, n = 46
S = 4.146211775813534, n = 47
S = 4.167485161837596, n = 48
S = 4.188315481873813, n = 49
```

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