**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)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №5

Дисциплина: Архитектура вычислительных систем

по теме Команды сопроцессора

Выполнил: ст. группы ВТ-31  
Новожен Н.В

Проверил: Осипов.О.В

**Белгород 2019**

**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 или функции Math.Pow языка C# совпадают. В отчёт включить текст программы, блок-схему алгоритма функции pow и набор тестовых данных.
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 = …

n = 2; S = …

…

|  |  |
| --- | --- |
| Вариант | Выражение |
| 11 |  |

**Задание 1**

float \_Pow(const float x, const float y) //x>0!

{

float z = 0;

\_\_asm {

fld y

fld x

FYL2X; Стек FPU теперь содержит : ST(0) = z = y \* log2(x) :

; Теперь считаем 2 \* \*z:

FLD st(0); Создаем еще одну копию z

FRNDINT; Округляем

; FXCH st(1)

; FSUBR st(1), st

FSUBR st(0), st(1); ST(1) = z, ST(0) = z - trunc(z)

F2XM1; ST(1) = z, ST(0) = 2 \* \*(z - trunc(z)) - 1

FLD1

FADDP st(1), st(0); ST(1) = z, ST(0) = 2 \* \*(z - trunc(z))

FXCH st(1)

FRNDINT

FXCH st(1)

FSCALE; ST(1) = z, ST(0) = (2 \* \*trunc(z))\*(2 \* \*(z - trunc(z))) = 2 \* \*t

FXCH st(1)

FSTP st; Результат остается на вершине стека ST(0)

//MOV in steck

SUB ESP, 4

FSTP dword ptr[ESP]

//MOV in reg EAX

MOV EAX, [ESP]

ADD ESP, 4

//MOV in z

MOV z,EAX

}

return z;

}

int main(){

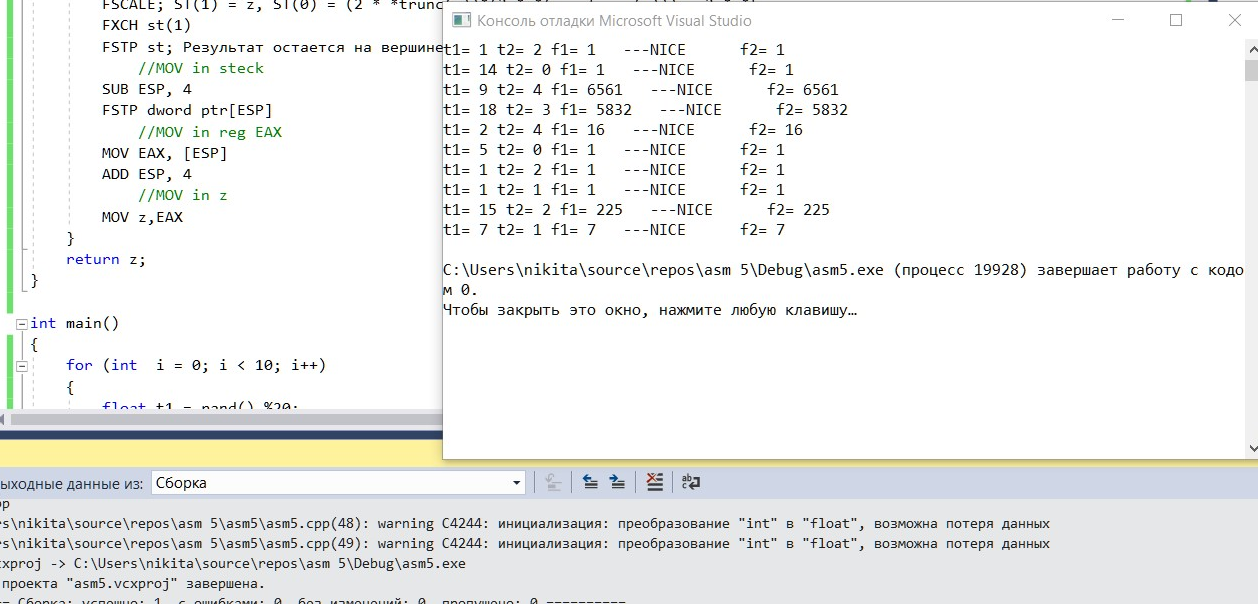
for (int i = 0; i < 10; i++){

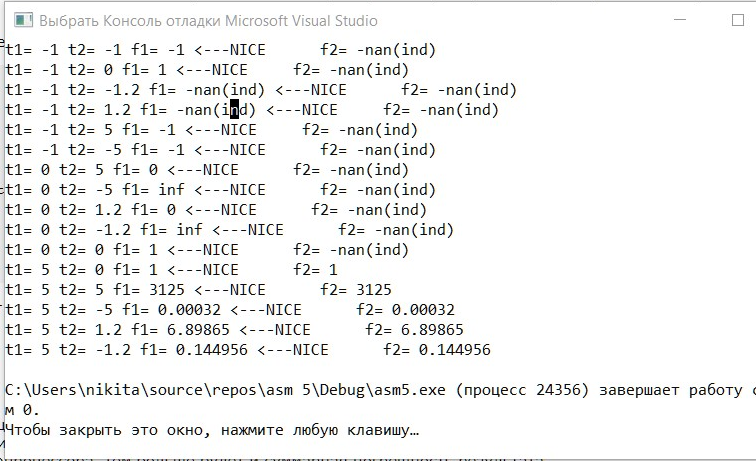
float t1 = rand() %20; float t2 = rand() % 5;

std::cout << "t1= " << t1 << " t2= " << t2 << " f1= " << pow(t1,t2) << " ---NICE f2= " << MyPow(t1, t2) << std::endl;

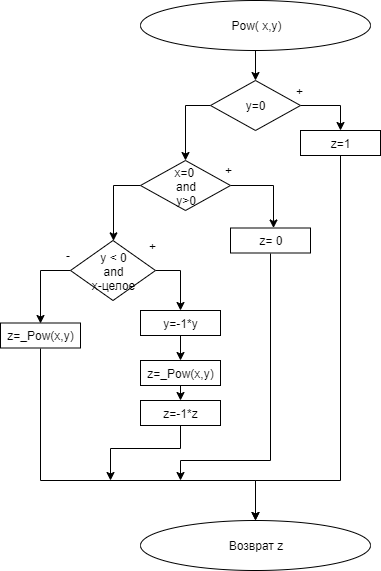
}

}





**Блок-схема функций Pow**



**Код функций Pow с тестами на asm**

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

include E:\masm32\include\windows.inc

include E:\masm32\include\kernel32.inc

include E:\masm32\include\user32.inc

includelib E:\masm32\lib\user32.lib

includelib E:\masm32\lib\kernel32.lib

include E:\masm32\include\msvcrt.inc

includelib E:\masm32\lib\msvcrt.lib

.DATA

x dd -1.0,-1.0,-1.0,-1.0,-5.0,-1.0,0.0,0.0,0.0,0.0,0.0,5.0,5.0,5.0,5.0,5.0

y dd -1.0,0.0,-1.2,1.2,5.0,-5.0,5.0,-5.0,1.2,-1.2,0.0,0.0,5.0,-5.0,1.2,-1.2;16

s db " X= Y= Z=%f ",13, 10, 0

.CONST

stp dd 1.4

.CODE

Pow proc

FINIT

FLD dword ptr[ESP + 8] ;x

FLD dword ptr[ESP + 4] ;y

FLDZ

;0, Y, X

db 0dbh, 0f0h+1;FCOMIP st(0),st(1)

je j66; Y==0

db 0dbh, 0f0h+2;FCOMIP st(0),st(1)

je j2; X==0

JMP j3

j2:

db 0dbh, 0f0h+1;

jb j20 ;0<Y

j3:

db 0dbh, 0f0h+2;

ja j4 ;0>X

JMP j5

j4:

FLD st(1); Создаем еще одну копию

;Y,0,Y,X

FRNDINT; Округляем

FSUBR st(0), st(2); ST(0) = z - trunc(z)

db 0dbh, 0f0h+1;

je j6 ;(ST0-ST1==0)X-ЦЕЛОЕ

JMP j5

;ERR 5 6

j6:

;FLD1

FLD dword ptr[ESP + 4] ;Y

;FDIVP st,st(1)

FLD dword ptr[ESP + 8] ;x

FCHS

FYL2X; Стек FPU теперь содержит : ST(0) = z = y \* log2(x) :

; Теперь считаем 2 \* \*z:

FLD st(0); Создаем еще одну копию z

FRNDINT; Округляем

FSUBR st(0), st(1); ST(1) = z, ST(0) = z - trunc(z)

F2XM1; ST(1) = z, ST(0) = 2 \* \*(z - trunc(z)) - 1

FLD1

FADDP st(1), st(0); ST(1) = z, ST(0) = 2 \* \*(z - trunc(z))

FXCH st(1)

FRNDINT

FXCH st(1)

FSCALE; ST(1) = z, ST(0) = (2 \* \*trunc(z))\*(2 \* \*(z - trunc(z))) = 2 \* \*t

FXCH st(1)

FSTP st; Результат остается на вершине стека ST(0)

FCHS

JMP j20

j5:

FLD dword ptr[ESP + 4] ;y

FLD dword ptr[ESP + 8] ;x

FYL2X; Стек FPU теперь содержит : ST(0) = z = y \* log2(x) :

; Теперь считаем 2 \* \*z:

FLD st(0); Создаем еще одну копию z

FRNDINT; Округляем

FSUBR st(0), st(1); ST(1) = z, ST(0) = z - trunc(z)

F2XM1; ST(1) = z, ST(0) = 2 \* \*(z - trunc(z)) - 1

FLD1

FADDP st(1), st(0); ST(1) = z, ST(0) = 2 \* \*(z - trunc(z))

FXCH st(1)

FRNDINT

FXCH st(1)

FSCALE; ST(1) = z, ST(0) = (2 \* \*trunc(z))\*(2 \* \*(z - trunc(z))) = 2 \* \*t

FXCH st(1)

FSTP st; Результат остается на вершине стека ST(0)

JMP j20

j66:

FLD1 ;1 ,0, Y ,X

j20:

SUB ESP, 8 ; Выделение памяти в стеке

FSTP qword ptr [ESP] ; Вытолкнуть число типа Dfloat на вершину стека

MOV EAX, [ESP] ; Скопировать в регистр

MOV EDX, [ESP+4] ; Скопировать в регистр

ADD ESP, 8 ; Вернуть стек в исходное состояние

RET 8

Pow endp

fun proc

fun endp

START:

XOR EDX,EDX

XOR ESI,ESI

XOR EDI,EDI

XOR ECX,ECX

MOV ECX,16

j55:

PUSH ECX

PUSH [offset x + EDI]

PUSH [offset y +EDI]

CALL Pow

PUSH EDX

PUSH EAX

;PUSH ESI

;PUSH [offset x + EDI]

;PUSH ESI

;PUSH [offset y +EDI]

PUSH offset s

CALL crt\_printf

ADD ESP,12

ADD EDI,4

POP ECX

LOOP j55

CALL crt\_\_getch

push NULL

call ExitProcess

END START

**Задание 2**

Численно исследовать на сходимость ряд

=>Ряд сходится

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 = …

**Код основной программы**

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

include E:\masm32\include\windows.inc

include E:\masm32\include\kernel32.inc

include E:\masm32\include\user32.inc

includelib E:\masm32\lib\user32.lib

includelib E:\masm32\lib\kernel32.lib

include E:\masm32\include\msvcrt.inc

includelib E:\masm32\lib\msvcrt.lib

.DATA

s db " N=%d Z=%10.7f ",13, 10, 0

.CONST

stp dd 1.4

.CODE

Pow proc

FINIT

FLD qword ptr[ESP + 8] ;x

FLD dword ptr[ESP + 4] ;y

FLDZ

;0, Y, X

db 0dbh, 0f0h+1;FCOMIP st(0),st(1)

je j66; Y==0

db 0dbh, 0f0h+2;FCOMIP st(0),st(1)

je j2; X==0

JMP j3

j2:

db 0dbh, 0f0h+1;

jb j20 ;0<Y

j3:

db 0dbh, 0f0h+2;

ja j4 ;0>X

JMP j5

j4:

FLD st(1); Создаем еще одну копию

;Y,0,Y,X

FRNDINT; Округляем

FSUBR st(0), st(2); ST(0) = z - trunc(z)

db 0dbh, 0f0h+1;

je j6 ;(ST0-ST1==0)X-ЦЕЛОЕ

JMP j5

;ERR 5 6

j6:

;FLD1

FLD dword ptr[ESP + 4] ;Y

;FDIVP st,st(1)

FLD qword ptr[ESP + 8] ;x

FCHS

FYL2X; Стек FPU теперь содержит : ST(0) = z = y \* log2(x) :

; Теперь считаем 2 \* \*z:

FLD st(0); Создаем еще одну копию z

FRNDINT; Округляем

FSUBR st(0), st(1); ST(1) = z, ST(0) = z - trunc(z)

F2XM1; ST(1) = z, ST(0) = 2 \* \*(z - trunc(z)) - 1

FLD1

FADDP st(1), st(0); ST(1) = z, ST(0) = 2 \* \*(z - trunc(z))

FXCH st(1)

FRNDINT

FXCH st(1)

FSCALE; ST(1) = z, ST(0) = (2 \* \*trunc(z))\*(2 \* \*(z - trunc(z))) = 2 \* \*t

FXCH st(1)

FSTP st; Результат остается на вершине стека ST(0)

FCHS

JMP j20

j5:

FLD dword ptr[ESP + 4] ;y

FLD qword ptr[ESP + 8] ;x

FYL2X; Стек FPU теперь содержит : ST(0) = z = y \* log2(x) :

; Теперь считаем 2 \* \*z:

FLD st(0); Создаем еще одну копию z

FRNDINT; Округляем

FSUBR st(0), st(1); ST(1) = z, ST(0) = z - trunc(z)

F2XM1; ST(1) = z, ST(0) = 2 \* \*(z - trunc(z)) - 1

FLD1

FADDP st(1), st(0); ST(1) = z, ST(0) = 2 \* \*(z - trunc(z))

FXCH st(1)

FRNDINT

FXCH st(1)

FSCALE; ST(1) = z, ST(0) = (2 \* \*trunc(z))\*(2 \* \*(z - trunc(z))) = 2 \* \*t

FXCH st(1)

FSTP st; Результат остается на вершине стека ST(0)

JMP j20

j66:

FLD1 ;1 ,0, Y ,X

j20:

SUB ESP, 8 ; Выделение памяти в стеке

FSTP qword ptr [ESP] ; Вытолкнуть число типа Dfloat на вершину стека

MOV EAX, [ESP] ; Скопировать в регистр

MOV EDX, [ESP+4] ; Скопировать в регистр

ADD ESP, 8 ; Вернуть стек в исходное состояние

RET 12

Pow endp

funrad proc

PUSH EBX

MOV EAX ,[ESP+8]

MOV EBX,EAX

IMUL EAX,5

ADD EAX,3

ADC EDX,0

IMUL EBX

POP EBX

RET 4

funrad endp

funarctg proc

FILD qword ptr[ESP+4]

FLD1

FPATAN

SUB ESP, 8 ; Выделение памяти в стеке

FSTP qword ptr [ESP] ; Вытолкнуть число типа Dfloat на вершину стека

MOV EAX, [ESP] ; Скопировать в регистр

MOV EDX, [ESP+4] ; Скопировать в регистр

ADD ESP, 8 ; Вернуть стек в исходное состояние

RET 8

funarctg endp

arctg proc;ARCTG -PI/2

FLD1

FADD st(0),st(0)

FLDPI

FDIV st(0) , st(1)

FLD qword ptr[ESP+4] ; PI/2 ,ST(0)=X 4 А НЕ 8

FSUBP st(1),st

SUB ESP, 8 ; Выделение памяти в стеке

FSTP qword ptr [ESP] ; Вытолкнуть число типа Dfloat на вершину стека

MOV EAX, [ESP] ; Скопировать в регистр

MOV EDX, [ESP+4] ; Скопировать в регистр

ADD ESP, 8 ; Вернуть стек в исходное состояние

RET 8

arctg endp

START:

XOR EDX,EDX

XOR ESI,ESI

XOR EDI,EDI

XOR ECX,ECX

MOV ECX,50

j55:

PUSH ECX

ADD ESI,1

PUSH ESI

CALL funrad

PUSH EDX

PUSH EAX

CALL funarctg

PUSH EDX

PUSH EAX

CALL arctg

PUSH EDX;OR EAX

PUSH EAX;;;;;;;;;;;;;;

PUSH stp

CALL Pow

PUSH EDX

PUSH EAX

PUSH ESI

PUSH offset s

CALL crt\_printf

ADD ESP,12+4

ADD EDI,4

POP ECX

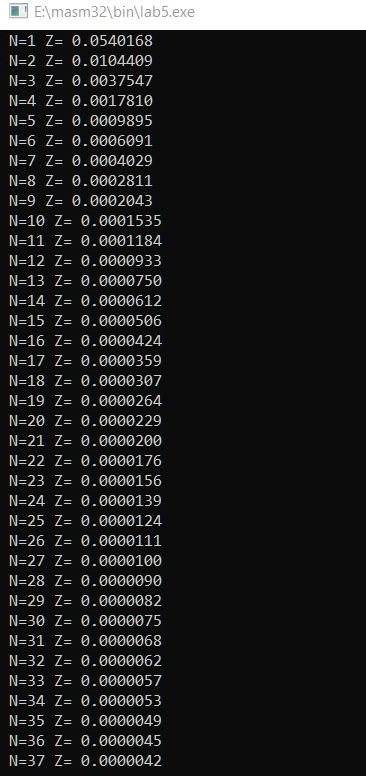
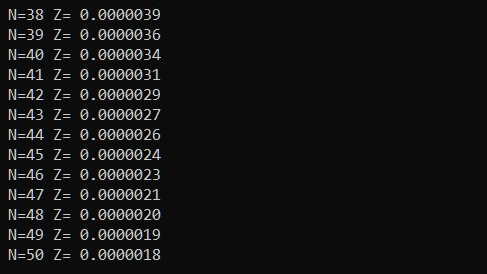
LOOP j55

CALL crt\_\_getch

push NULL

call ExitProcess

END START

**Вывод**: В ходе лабораторной работы мы изучили команды сопроцессора для выполнения арифметических операций с целыми и вещественными числами, Применяли команды вычисления трансцендентных функций и другие ,Также рассмотрели структуру сопроцессора