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

по дисциплине «Архитектура вычислительных систем»

тема: «Команды сопроцессора»

Выполнил: ст. группы ВТ-32

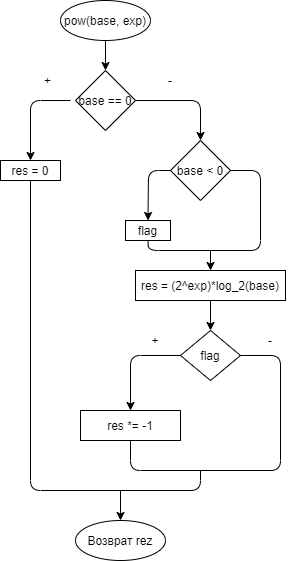
Воскобойников И. С.

Проверил: Осипов О. В.

Белгород 2020 г.

**Цель работы**: изучение команд сопроцессора для выполнения арифметических операций.

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.



pow proc

SUB ESP, 116 ; выделяем память

FSAVE [ESP] ; записать состояние сопроцессора

FLD qword ptr [ESP+4+116] ; помещаем на вершину стека exp

FLD qword ptr [ESP+12+116]; помещаем на вершину стека base

FLDZ ; помещаем на вершину стека 0

FCOMIP ST(0), ST(1) ; сравнить основание с нулем, вытолкнуть ST(0)

JE baseIs0 ; если основание 0

JA baseIsNeg ; если основание < 0

MOV EAX, 0 ; EAX = 0

processing:

FYL2X ; помещаем на вершину стека логарифм

FLD1 ; помещаем на вершину стека 1

FLD ST(1) ; помещаем на вершину стека копию ST(1)

FPREM ; вычисляем остаток от деления d в ST(0)

F2XM1 ; вычисление 2^x - 1, результат в ST(0)

FADDP ST(1),ST(0); ST(1) += ST(0), выталкивание ST(0)

FSCALE

CMP EAX, 1

JE changeSign

endProc:

FSTP qword ptr [ESP+108] ; Сохранение вещественного числа из вершины стека в память

FRSTOR [ESP] ;Восстанавливаем состояние сопроцессора

FLD qword ptr [ESP+108] ; Помещаем на вершину стека результат

ADD ESP, 116 ; Освобожаем память

RET 16

changeSign:

FLDZ ; помещаем на вершину стека 0

FLD1 ; помещаем на вершину стека 1

FSUBP ST(1),ST(0) ; ST(1) -= ST(0), выталкивание из стека ST(0)

FMULP ST(1),ST(0) ; ST(1) \*= ST(0), выталкивание из стека ST(0)

JMP endProc ; Переход на возврат результата

baseIsNeg:

MOV EAX, 1 ; EAX = 1

FLDZ ; помещаем на вершину стека -1

FLD1

FSUBP ST(1),ST(0)

FMULP ST(1),ST(0) ; ST(1) \*= ST(0), выталкивание из стека ST(0)

JMP processing ; Переход на вычисление

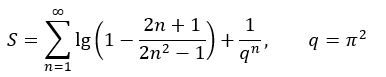
baseIs0:

FLDZ ; Заталкиваем на вершину 0

JMP endProc ; Переходим на вывод

pow 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:



.686 ; Тип процессора

.model flat, stdcall ; Модель памяти и стиль вызова подпрограмм

option casemap: none ; Чувствительность к регистру

; --- Подключение файлов с кодом, макросами, константами, прототипами функций и т.д.

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

; --- Подключаемые библиотеки ---

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

; --- Сегмент данных ---

.DATA

fout db "n = %d, S = %f", 10, 0

maxN dd 50

two dq 2.0

q dq ?

.CODE

START:

XOR ECX, ECX

FLDPI ; помещаем на вершину стека pi

FLDPI ; помещаем на вершину стека pi

FMULP ST(1), ST(0) ; ST(1) = pi^2, ST(0) вытолкнули

FSTP q ; q = pi^2, вытолкнули ST(0)

FLDZ ; помещаем на вершину стека 0

MOV ECX, 1 ; ECX = 1 ; Это n

solve:

CMP ECX, maxN ; ECX == maxN =>

JG endCycle ; => то конец цикла

PUSH 0

PUSH ECX

FLD1 ; помещаем на вершину стека 1 ; Нужно далее для команды FYL2X

FILD qword ptr [ESP] ; помещаем на вершину стека n ; Собираем знаменатель

FLD ST(0) ; помещаем на вершину стека n ;

FMULP ST(1),ST(0) ; ST(1) = n^2; вытолкнули ST(0)

FLD two ; помещаем на вершину стека 2.0

FMULP ST(1),ST(0) ; ST(1) = 2\*n^2; вытолкнули ST(0)

FLD1 ; помещаем на вершину стека 1

FSUBP ST(1),ST(0) ; ST(1) = 2\*n^2 - 1; вытолкнули ST(0)

FILD qword ptr [ESP] ; помещаем на вершину стека n ; Собираем числитель

FLD two ; помещаем на вершину стека 2.0

FMULP ST(1),ST(0) ; ST(1) = 2\*n; вытолкнули ST(0)

FLD1 ; помещаем на вершину стека 1

FADDP ST(1),ST(0) ; ST(1) = 2\*n + 1; вытолкнули ST(0)

FDIVRP ST(1),ST(0) ; ST(1) = (2\*n + 1) / (2\*n^2 - 1); вытолкнули ST(0)

FLD1 ; помещаем на вершину стека 1

FCOM

FSTSW AX

SAHF

JBE isNegLg ; если 1 <= (2\*n + 1) / (2\*n^2 - 1), то не выводим

FSUBRP ST(1),ST(0) ; ST(1) = 1 - (2\*n + 1) / (2\*n^2 - 1); вытолкнули ST(0)

FYL2X ; ST(1) = 1 \* LOG\_2(1 - (2\*n + 1) / (2\*n^2 - 1)); вытолкнули ST(0)

FLDL2T ; Помещаем в ST(0) LOG\_2(10)

FDIVP ST(1),ST(0) ; ST(1) = LOG\_2(1 - (2\*n + 1) / (2\*n^2 - 1)) / LOG\_2(10) ; вытолкнули ST(0)

SUB ESP, 16

FLD q ; Помещаем на вершину стека число, возводимое в степень

FILD qword ptr [ESP+16] ; Помещаем на вершину стека n ; Показатель степени

FSTP qword ptr [ESP] ; Загружаем в стек из ST(0)

FSTP qword ptr [ESP+8] ; Загружаем в стек из ST(0)

call pow ; Выталкиваем ST(1) и ST(0); ST(0) = q^n

FLD1 ; Помещаем на вершину стека 1

FDIVRP ST(1),ST(0) ; ST(1) = 1 / q^n ; вытолкнули ST(0)

FADDP ST(1),ST(0) ; ST(1) = LOG\_2(1 - (2\*n + 1) / (2\*n^2 - 1)) / LOG\_2(10) + 1 / q^n ; вытолкнули ST(0)

FADDP ST(1),ST(0) ; Сумма текущей итерации с предыдущими

SUB ESP, 8 ; Выделяем место для результата в стэке

FST qword ptr [ESP] ; Помещаем ST(0) в стэк

PUSH ECX ; Помещаем в стек n

PUSH offset fout ; Формат помещаем в стек

CALL crt\_printf ; Выводим на экран n = %d, S = %f

ADD ESP, 4 ; Очищаем стек после принта (Но в стеке остается ECX)

POP ECX ; Восстанавливаем ECX (текущее значение n)

getNextSolve:

INC ECX ; n += 1

ADD ESP, 16 ; Чистим мусор из стека

JMP solve ; Новая итерация

isNegLg:

SUB ESP, 8 ; Выделяем буфер для очистки ST

FSTP qword ptr [ESP]

FSTP qword ptr [ESP]

FSTP qword ptr [ESP]

JMP getNextSolve

endCycle:

call crt\_\_getch

push 0

call ExitProcess

END START