**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

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**

**Дисциплина: Архитектура вычислительных систем**

**Тема: Команды передачи управления**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: Осипов Олег Васильевич

**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 13**

|  |  |
| --- | --- |
| 13 |  |

**Ход работы**

|  |  |  |
| --- | --- | --- |
| степень | число | результат |
| 7 | 15.785882283611476 | 244276685.97512513 |
| 4 | 12.397380786333002 | 23622.16850557444 |
| 5 | 1.5547164190911074 | 9.083556575710105 |
| 10 | 8.152607336313801 | 1297075493.1452634 |
| 2 | 8.149783655406079 | 66.41897362992407 |
| 8 | 10.432880896495366 | 140356973.6460661 |
| 8 | 2.98295215209596 | 6268.596229167466 |
| 6 | 1.538308962780702 | 13.251389593189709 |
| 3 | 1.6246359823133498 | 4.288132568321104 |

**Изображение выглядит как снимок экрана, компьютер, ноутбук, монитор

Автоматически созданное описание**

.386

.model flat, stdcall

option casemap: none

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

.DATA

x dd 10

y dd 2

format db "n = %d, val = %G",13, 10, 0

.CODE

; возвадит x в степень y

; результат в EAX

pow proc

; 4 байта аргумент num

; 4 байта аргумент pow

; 4 байта - адрес возврата

; 12 байт - сохранение состояний регистров

; 4 байта - буфер в стеке, для передачи в сопроцессор

PUSH ECX ; сохраняем регистры в стеке

PUSH EDX

PUSH EBX

SUB ESP, 4 ; выделяем в стеке 4 байта

MOV EAX, [ESP+24] ; записываем x в EAX

MOV [ESP], EAX ; записываем EAX в стек

fild dword ptr [ESP] ; записываем вершину стека в ST(0)

MOV ECX, [ESP+20] ; записываем y в EC

DEC ECX ; вычитаем 1 из ECX

pow\_cycle:

fimul dword ptr [ESP] ; умножаем ST(0) на вершину стека, результат в ST(0)

loop pow\_cycle

fistp dword ptr [ESP] ; выталкиваем ST(0) в стек

MOV EAX, [ESP] ; записываем вершину стека в EAX

MOV EBX, EAX ; записываем EAX в EBX

ADD ESP, 4 ; удаляем из стека 4 байта

MOV EAX, EBX ; записываем EBX в EAX

POP EBX ; возвращием регистры из стека

POP EDX

POP ECX

RET 8

pow endp

; Получает итерацию n основного цикла

; Возвращает результат в EAX

row proc

PUSH ECX ; сохраняем ECX в стек

MOV EAX, [ESP+8] ; записываем n в EAX

SUB ESP, 4 ; пропускаем ECX в стеке

MOV [ESP], EAX ; записываем в стек EAX

;f1

fild dword ptr [ESP] ; загружаем в ST(0) вершину стека ; n

MOV EAX, 2 ; записываем 2 в EAX

MOV [ESP], EAX ; записываем в стек EAX

fimul dword ptr [ESP] ; умножаем ST(0) на вершину стека, результат в ST(0) ; 2\*n

MOV EAX, 1 ; записываем 1 в EAX

MOV [ESP], EAX ; записываем EAX в стек

fiadd dword ptr [ESP] ; складываем ST(0) и вершину стека, результат в ST(0) ; 2\*n+1

;f2

MOV EAX, [ESP+12] ; записываем n в EAX

MOV [ESP], EAX ; записываем EAX в стек

fild dword ptr [ESP] ; записываем в ST(0) вершину стека ; n

fimul dword ptr [ESP] ; умножаем ST(0) на вершину стека ; n^2

MOV EAX, 2 ; записываем 2 в EAX

MOV [ESP], EAX ; записываем EAX в вершину стека

fimul dword ptr [ESP] ; умножаем ST(0) на вершину стека, результат в ST(0) ; 2\*n^2

MOV EAX, 1 ; записываем 1 в EAX

MOV [ESP], EAX ; записываем EAX в вешину стека

fisub dword ptr [ESP] ; вычитаем из ST(0) вершину стека, результат в ST(0) ; 2\*n^2-1

;f3

fdiv ST(1), ST(0) ; делим ST(1) на ST(0), результат в ST(1) ; (2\*n+1)/(2\*n^2-1)

fstp dword ptr [ESP] ; выталкиваем ST(0) в стек

MOV EAX, 1 ; записываем 1 в EAX

MOV [ESP], EAX ; записываем EAX в стек

fild dword ptr [ESP] ; записываем в ST(0) вершину стека

fsubr ST(1), ST(0) ; вычитаем ST(1) из ST(0), результат в ST(1) ; 1 - (2\*n+1)/(2\*n^2-1)

;f4

fstp dword ptr [ESP] ; выталкиваем ST(0) в стек

MOV EAX, [ESP+12] ; записываем n в EAX

MOV ECX, 2 ; записываем 2 в ECX

MUL ECX ; умножаем EAX на ECX, результат в EDX:EAX ; 2\*n

MOV ECX, EAX ; записываем EAX в ECX

fldpi ; записываем в ST(0) число pi

fldpi ; записываем в ST(0) число pi

loop\_start:

DEC ECX ; вычитаем 1 из ECX ;

fmul ST(1), ST(0) ; умножаем ST(1) на ST(0), результат в ST(1) ; pi^2

CMP ECX, 0 ; сравниваем ECX и 0

jne loop\_start ; если ECX меньше или равен, повторяем цикл

fmulp ST(1), ST(0) ; умножаем ST(1) на ST(0), результат в ST(1), выталкиваем ST(0) в стек ; (pi^2)^n

;f5

MOV EAX, 1 ; записываем 1 в EAX

MOV [ESP], EAX ; записываем EAX в стек

fild dword ptr [ESP] ; записываем в ST(0) вершину стека

fdivr ST(1), ST(0) ; делим ST(0) на ST(1), результат в ST(1) ; 1/(pi^2)^n

fstp dword ptr [ESP] ; выталкиваем ST(0) в стек

fadd ST(0), ST(1) ; складываем ST(0) и ST(1), результат в ST(0)

fstp dword ptr [ESP] ; выталкиваем ST(0) в стек

MOV EAX, [ESP] ; записываем в EAX вершину стека

fstp dword ptr [ESP] ; выталкиваем ST(0) в стек

ADD ESP, 4 ; пореходим к ECX в стеке

POP ECX ; возвращием ECX из стека

RET 8 ; выходим из подпрограммы

row endp

start:

XOR EAX, EAX ; очищаем регистры

XOR EBX, EBX

XOR EDX, EDX

XOR ECX, ECX

loop\_start:

INC ECX ; увеличиваем счетчик основного цикла

PUSH ECX ; записываем ECX в стек

CALL row

PUSH EAX ; записываем EAX в стек

PUSH 0 ; записываем 0 в стек

PUSH ECX ; записываем ECX в стек

PUSH offset format

CALL crt\_printf

ADD ESP, 4 ; удаляем из стека 4 байта

POP ECX ; возвращаем ECX из стека

CMP ECX, 1000 ; сравниваем ECX и 50

JL loop\_start ; если ECX меньше, повторить цикл

END start