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 ПВ-223

# Дмитриев Андрей Александрович

Проверил:

# Осипов Олег Васильевич

Белгород 2024 г.

# Вариант 2

Цель работы: изучение команд сопроцессора для выполнения арифметических операций

# Задание №1.

Для возведения числа 2 в произвольную степень также нет команды. Команда **F2XM1** корректно считает только тогда, когда в ST(0) находиться число в интервале (-1, 1). Поэтому число *t* = *y*log2*x* нужно разбить на целую (*a*) и дробную части (*b*). Дробную часть можно получить в результате вычисления командой **FPREM** частичного остатка от деления *t* на единицу. Есть другой способ. Если сначала получить целую часть *a* с помощью округления до ближайшего меньшего целого, то *b* = *t* - *a*. Тогда 2𝑡 = 2𝑎+𝑏 = 2𝑎2𝑏.

2*a* нужно вычислять помощью команды **FSCALE**. Целую часть можно специально не вычислять, так как команда **FSCALE** сама округляет число в ST(1) до ближайшего меньшего целого. 2*b* теперь можно вычислить командой **F2XM1**, но после выполнения этой команды к результату нужно прибавить единицу. Если *x* < 0, то нужно просто вынести знак: (-*x*)*y* = *xy*. Если *y* < 0, то *x*-*y* = 1 / *xy*.

Код:

; x ^ y = 2 ^ (y \* log2x)

; 2 ^ t

; t = y log2 x

; дробная часть = b

; целая часть = a( с округлением до целого меньшего)

; b = t - a

; 2 ^ a \* 2 ^ b

pow proc

    fld qword PTR [ESP + 4] ;st: x

    fldz ;st: 0 x

    db 0DBh , 0F0h+1 ;FCOMI 0, x (сравнение х и 0)

    je ret1 ;x = 0 -> выход, st0 = 0

    fld qword PTR [ESP + 12] ;st: y 0 x

    db 0DBh , 0F0h+1 ;FCOMI y, 0 (сравнение y и 0)

    fld1 ;st: 1 y 0 x

    je ret1 ;y = 0 -> выход st0 = 1

    finit ;отчистка+инициализация

    fld qword PTR [ESP + 4] ;st: x -2

    fld ST(0) ;st: x x -2 -2

    fabs ;st: abs(x) x 2 -2

    fdiv ST(0), ST(1) ;st: sign(x) x -1 -20

    fxch ST(1) ;st: x, sign(x) -2 -1

    fld qword PTR [ESP + 12] ;st: y, x, sign(x) 2 -2 -1

    fld ST(0) ;st: y, y, x, sign(x) 2 -2 -1

    frndint ; округляем y

    db 0dfh, 0f0h+1; сравнение y округленного и не округленного

    jne raise\_to\_degree

    raise\_to\_degree:

        FXCH ST(1) ;st: x, y, sign(x) 2 2 -1

        FABS ;st: abs(x), y, sign(x) 2 2 -1

        FYL2X ;st: t, sign(x) 2 -1

        FLD1 ;st: 1, t, sign(x) 1 2 -1

        FSCALE ; 2^[t], t, sign(x) 4 2 -1

        FLD1

        FLD ST(2)

        FPREM

        F2XM1

        FADDp ST(1), ST(0)

        FMULp ST(1), ST(0)

        FMUL ST(0), ST(2); умножение 2^(целая часть) \* 2^(дробная часть)

    ret1:

        RET 16

pow endp

START:

    push dword ptr y[4]

    push dword ptr y[0]

    push dword ptr x[4]

    push dword ptr x[0]

    call pow

    ; Вывод результата в консоль

    sub esp, 8

    fstp qword ptr [esp]

    push  offset format

    call  crt\_printf

    ; Завершаем программу

    push 0

    call ExitProcess

END START

Тестовые данные:

|  |  |  |
| --- | --- | --- |
| x | y | res |
| 2.0 | 3.0 | 8.0 |
| -2.0 | 3.0 | -8.0 |
| 2.0 | -3.0 | 0.125 |
| -2.0 | -3.0 | -0.125 |
| 16.0 | 0.5 | 4.0 |
| 2.0 | 0.0 | 1 |
| 0.0 | 2.0 | 0 |

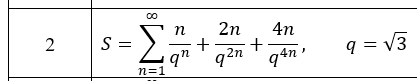
Задание №2.

Составить таблицу состояния регистров для всех вещественных операций: для начальных операций перед циклом, для первой и последней итераций, для завершающих операций после цикла.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Этап выполнения | Операция | ST(0) | ST(1) | ST(2) | ST(3) |
| Проверка x = 0 | fld x | x |  |  |  |
|  | fldz | 0 | x |  |  |
|  | fcomi x, 0 | 0 | x |  |  |
| Проверка y = 0 | fld y | y | 0 | x |  |
|  | fcomi y, 0 | y | 0 | x |  |
|  | fldz | 0 | y | 0 | x |
|  | fld1 | 1 | 0 | y | 0 |
| Вычисление | finit |  |  |  |  |
|  | fld x | x |  |  |  |
|  | fld st(0) | x | x |  |  |
|  | fabs | abs(x) | x |  |  |
|  | fdiv st(0), st(1) | sign(x) | x |  |  |
|  | fxch st(1) | x | sign(x) |  |  |
|  | fld y | y | x | sign(x) |  |
|  | fld st(0) | y | y | x | sign(x) |
|  | frdint | round(y) | y | x | sign(x) |
|  | fcomip round(y), y | y | x | sign(x) |  |
|  | fxch st(1) | x | y | sign(x) |  |
|  | fabs | abs(x) | y | sign(x) |  |
|  | fyl2x | ylog2(x) | sign(x) |  |  |
|  | fld1 | 1 | ylog2(x) | sign(x) |  |
|  | fscale | 2ylog2(x) | ylog2(x) | sign(x) |  |
|  | fld1 | 1 | 2ylog2(x) | ylog2(x) | sign(x) |
|  | fld st(2) | ylog2(x) | 1 | 2ylog2(x) | ylog2(x) |
|  | fprem | rem(ylog2(x)) | 1 | 2ylog2(x) | ylog2(x) |
|  | f2xm1 | 2rem(ylog2(x))-1 | 1 | 2ylog2(x) | ylog2(x) |
|  | faddp st(1), st(0) | 2rem(ylog2(x)) | 2ylog2(x) | ylog2(x) | sign(x) |
|  | fmulp st(1), st(0) | 2rem(ylog2(x))+ ylog2(x) | ylog2(x) | sign(x) |  |
|  | fmul st(0), st(2) | result | sign(x) |  |  |

Задание №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.



Код:

 .686

 .model flat, stdcall

 option casemap: none

include windows.inc

include kernel32.inc

include user32.inc

include msvcrt.inc

includelib kernel32.lib

includelib msvcrt.lib

includelib user32.lib

.DATA

q dq 3.0

n dq 1.0

n\_end dq 50.0

sum dq 0.0

part\_degree\_2\_23 dq 2.0

part\_degree\_3\_2 dq 3.0

part\_degree\_5\_3 dq 5.0

degree dq 1.0

str\_fmt db "n = %.1f, S = %.2f ", 13, 10, 0

.CODE

; proc pow

calculating\_denominator proc

    finit

    xor eax, eax

    xor edx, edx

    mov eax, dword ptr [degree]

    mov edx, dword ptr [degree + 4]

    push edx

    push eax

    xor eax, eax

    xor edx, edx

    mov eax, dword ptr [q]

    mov edx, dword ptr [q + 4]

    push edx

    push eax

    call pow

    ret 0

calculating\_denominator endp

START:

    finit

    fld q

    fsqrt

    fstp q

    j1:

        mov eax, dword ptr [n]

        mov edx, dword ptr [n + 4]

        mov dword ptr [degree + 4], edx

        mov dword ptr [degree], eax

        call calculating\_denominator

        fld n

        fdiv st(0), st(1)

        fld sum

        fadd

        fst sum

        finit

        fld const\_2

        fld n

        fmulp st(1), st(0)

        fst degree

        call calculating\_denominator

        fld degree

        fdiv st(0), st(1)

        fld sum

        fadd

        fst sum

        finit

        fld const\_4

        fld n

        fmulp st(1), st(0)

        fst degree

        call calculating\_denominator

        fld degree

        fdiv st(0), st(1)

        fld sum

        fadd

        fst sum

        sub ESP, 8 ; Выделение памяти в стеке

        fstp qword ptr [ESP] ; Вытолкнуть число типа float на вершину стека

        mov eax, dword ptr [n]

        mov edx, dword ptr[n + 4]

        push edx

        push eax

        push offset str\_fmt

        call crt\_printf

        add esp, 4\*5

        fld n\_end

        fld n

        db 0DBh , 0F0h+1

        je ret2

        fld1

        fadd st(0), st(1)

        fst n

        jmp j1

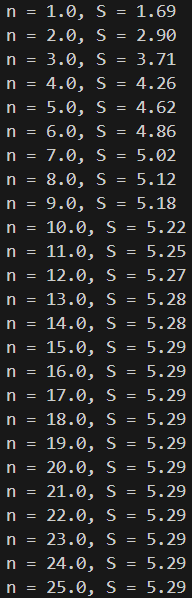
    ret2:

        push 0

        call ExitProcess

END START

Пример работы программы:



Контрольный код на Python:

def func(n: float, q: float):  
 return \  
 n / q \*\* n \  
 + 2 \* n / q \*\* (2 \* n) \  
 + 4 \* n / q \*\* (4 \* n)  
  
  
S = 0  
q = sqrt(3)  
for n in range(1, 25):  
 S += func(n, q)  
 print(f"n = {n}, S = {S}")

Вывод: В ходе лабораторной работы изучены команды сопроцессора для выполнения арифметических операций.