|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №9**

**«Программирование математического сопроцессора. Вызов ассемблерных функций из программ C/C++»**

**ДИСЦИПЛИНА: «Машинно-зависимые языки программирования»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-32Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Амеличева К.А. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

**Цель:** практическое овладение навыками разработки программного кода на языке Ассемблер. Изучить особенности программирования математического сопроцессора. Научиться стыковать модули, написанные с помощью языка Ассемблер и С++.

**Вариант 8-7**

Составить программу для вычисления значения функций

**Использованные команды:**

finit – инициализация сопроцессора

fld src – загрузка операнда в вершину стека

fst dst – копирование вершины стека dst = ST(0)

fxch ST(i) – обмен значений ST(0) и ST(i)

fld1 – загрузка 1 в вершину стека

fldpi – загрузка числа Пи в вершину стека

fadd dst, src – вещественное сложение dst = dst + src

fsub dst, src – вещественное вычитание dst = dst – src

fmul dst, src – вещественное умножение dst = dst \* src

fdivr dst, src – реверсивное вещественное деление dst = src / dst

frndint – округление ST(0) = (ST(0))

fsqrt – вычисление квадратного корня ST(0) = √ST(0)

fsin – вычисление синуса ST(0) = sin ST(0)

fsincos – вычисляет синус ST(0) = sin ST(0), загружает косинус cos ST(0) в вершину стека

fptan – вычисляет тангенс ST(0) = tg ST(0), загружает 1 в вершину стека

fscale – вычисление выражения ST(0) = ST(0) \* 2ST(1), где ST(1) – целое число

f2xm1 – вычисление выражения ST(0) = 2ST(0) – 1 при ST(0) ∈ [-1; 1]

fyl2x – выталкивает ST(0) = x и ST(1) = y, загружает y\*log2 x в вершину стека

fcom – вещественное сравнение ST(0) с ST(1)

fstsw dst – считывание слова состояния сопроцессора в память

fnstcw dst – считывание слова управления сопроцессора в память

sahf – загрузка кода из AX во флаговый регистр

fldcw src – загрузка слова управления сопроцессора

**Листинг:**

Assembler

.586

.MODEL FLAT, C

.DATA

two dd 2.0

six dd 6.0

lowerBound dd 1.1

upperBound dd 5.1

nine dd 9.0

fsttmp dw 0

.CODE

calc proc C x:qword

finit

fld x ; ST(0) = x

fld x ; ST(0) = x, ST(1) = x

fmul ST(0), ST(1) ; ST(0) = x^2, ST(1) = x

fmul ST(0), ST(1) ; ST(0) = x^3, ST(1) = x

fmul ST(0), ST(1) ; ST(0) = x^4, ST(1) = x

fld1 ; ST(0) = 1, ST(1) = x^4, ST(2) = x

fsub ST(1), ST(0) ; ST(0) = 1, ST(1) = x^4 - 1, ST(2) = x

fxch ST(1) ; ST(0) = x^4 - 1, ST(1) = 1, ST(2) = x

fsqrt ; ST(0) = sqrt(x^4 - 1), ST(1) = 1, ST(2) = x

fld two ; ST(0) = 2, ST(1) = sqrt(x^4 - 1), ST(2) = 1, ST(3) = x

fxch ST(1) ; ST(0) = sqrt(x^4 - 1), ST(1) = 2, ST(2) = 1, ST(3) = x

fyl2x ; ST(0) = 2 \* log2(sqrt(x^4 - 1)), ST(1) = 1, ST(2) = x

fxch ST(2) ; ST(0) = x, ST(1) = 1, ST(2) = 2 \* log2(sqrt(x^4 - 1))

fsin ; ST(0) = sin x, ST(1) = 1, ST(2) = 2 \* log2(sqrt(x^4 - 1))

fldpi ; ST(0) = pi, ST(1) = sin x, ST(2) = 1, ST(3) = 2 \* log2(sqrt(x^4 - 1))

fld six ; ST(0) = 6, ST(1) = pi, ST(2) = sin x, ST(3) = 1,

ST(4) = 2 \* log2(sqrt(x^4 - 1))

fdivr ST(0), ST(1) ; ST(0) = pi / 6, ST(1) = pi, ST(2) = sin x, ST(3) = 1,

ST(4) = 2 \* log2(sqrt(x^4 - 1))

fadd ST(0), ST(2) ; ST(0) = sin x + pi / 6, ST(1) = pi, ST(2) = sin x, ST(3) = 1,

ST(4) = 2 \* log2(sqrt(x^4 - 1))

fdivr ST(0), ST(1) ; ST(0) = pi / (sin x + pi / 6), ST(1) = pi, ST(2) = sin x, ST(3) = 1,

ST(4) = 2 \* log2(sqrt(x^4 - 1))

fdivr ST(0), ST(4) ; ST(0) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fld x ; ST(0) = x, ST(1) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fld x ; ST(0) = x, ST(1) = x, ST(2) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fmul ST(0), ST(1) ; ST(0) = x^2, ST(1) = x,

ST(2) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fsin ; ST(0) = sin x^2, ST(1) = x,

ST(2) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fsqrt ; ST(0) = sqrt(sin x^2), ST(1) = x,

ST(2) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fld1 ; ST(0) = 1, ST(1) = sqrt(sin x^2), ST(2) = x,

ST(3) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fxch ST(1) ; ST(0) = sqrt(sin x^2), ST(1) = 1, ST(2) = x,

ST(3) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fyl2x ; ST(0) = log2(sqrt(sin x^2)), ST(1) = x,

ST(2) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fptan ; ST(0) = 1, ST(1) = tg(log2(sqrt(sin x^2))),

ST(2) = x, ST(3) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fxch ST(1) ; ST(0) = tg(log2(sqrt(sin x^2))), ST(1) = 1, ST(2) = x,

ST(3) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fst ST(1) ; ST(0) = tg(log2(sqrt(sin x^2))), ST(1) = tg(log2(sqrt(sin x^2))),

ST(2) = x, ST(3) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fmul ST(1), ST(0) ; ST(0) = tg(log2(sqrt(sin x^2))), ST(1) = tg^2(log2(sqrt(sin x^2))),

ST(2) = x, ST(3) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6))

fxch ST(3) ; ST(0) = 2 \* log2(sqrt(x^4 - 1)) / (pi / (sin x + pi / 6)),

ST(1) = tg^2(log2(sqrt(sin x^2))), ST(2) = x, ST(3) = tg(log2(sqrt(sin x^2)))

fsub ST(0), ST(1) ; ST(0) = 2\*log2(sqrt(x^4-1)) / (pi / (sin x + pi/6)) - tg^2(log2(sqrt(sin x^2)))

ret

calc ENDP

compare proc C x:qword

finit

fld lowerBound ; ST(0) = 1.1

fld x ; ST(0) = x, ST(1) = 1.1

fcom

fstsw ax

sahf

jb CASE\_1

fld upperBound ; ST(0) = 5.1, ST(1) = x

fcom

fstsw ax

sahf

jae CASE\_2

fld nine ; ST(0) = 9, ST(1) = 5.1, ST(2) = x

fmul ST(0), ST(2) ; ST(0) = 9x, ST(1) = 5.1, ST(2) = x

fmul ST(0), ST(2) ; ST(0) = 9x^2

fld nine ; ST(0) = 9, ST(1) = 9x^2

fadd ST(0), ST(1) ; ST(0) = 9x^2 + 9

jmp CASE\_END

CASE\_2:

fxch ST(1) ; ST(0) = x, ST(1) = 5.1

fnstcw fsttmp

or [fsttmp], 0C00h

fldcw fsttmp

frndint ; ST(0) = цел(x)

fld x ; ST(0) = x, ST(1) = цел(x)

fsub ST(0), ST(1) ; ST(0) = ост(x), ST(1) = цел(x)

f2xm1 ; ST(0) = 2^ост(x) - 1, ST(1) = цел(x)

fxch ST(1) ; ST(0) = цел(x), ST(1) = 2^ост(x) - 1

fld1 ; ST(0) = 1, ST(1) = цел(x), ST(2) = 2^ост(x) - 1

fadd ST(2), ST(0) ; ST(0) = 1, ST(1) = цел(x), ST(2) = 2^ост(x)

fxch ST(2) ; ST(0) = 2^ост(x), ST(1) = цел(x), ST(2) = 1

fscale ; ST(0) = 2^x, ST(1) = цел(x), ST(2) = 1

fsub ST(0), ST(2) ; ST(0) = 2^x - 1

jmp CASE\_END

CASE\_1:

fsincos ; ST(0) = cos x, ST(1) = sin x

fadd ST(0), ST(1) ; ST(0) = sin x + cos x

CASE\_END:

ret

compare ENDP

END

C++

#define \_USE\_MATH\_DEFINES

#include <iostream>

using namespace std;

extern "C" double calc(double);

extern "C" double compare(double);

int main() {

double a, x;

cout << "x = ";

cin >> x;

cout << endl << "Task 1" << endl;

if (x > 1 && sin(x) != -M\_PI / 6 && sin(x \* x) > 0 && cos(log2(sqrt(sin(x \* x)))) != 0)

{

a = (2 \* log2(sqrt(pow(x, 4) - 1))) / (M\_PI / (sin(x) + M\_PI / 6)) - pow(tan(log2(sqrt(sin(x \* x)))), 2);

cout << "(C++) a = " << a << endl;

a = calc(x);

cout << "(Asm) a = " << a << endl;

}

else

{

cout << "Invalid x" << endl;

}

cout << endl << "Task 2" << endl;

if (x < 1.1)

{

a = sin(x) + cos(x);

}

else if (x <= 5.1)

{

a = pow(2, x) - 1;

}

else

{

a = 9\*x\*x + 9;

}

cout << "(C++) f(x) = " << a << endl;

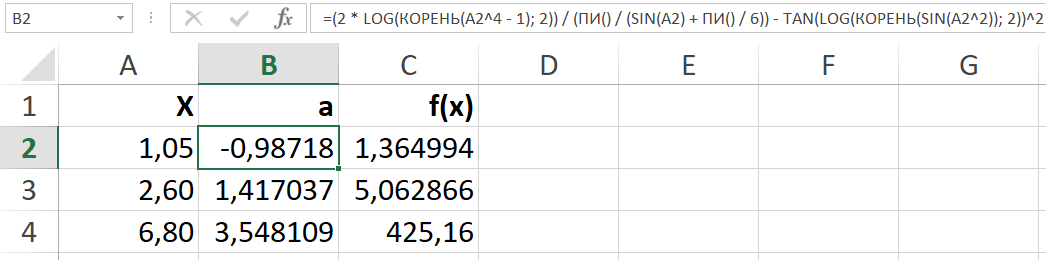
a = compare(x);

cout << "(Asm) f(x) = " << a << endl;

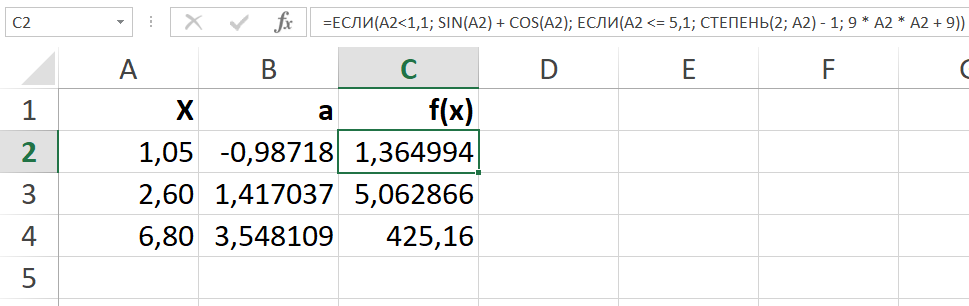
return 0;

}

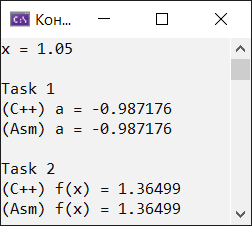
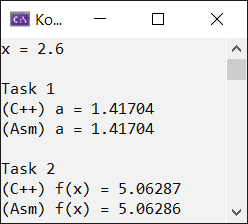
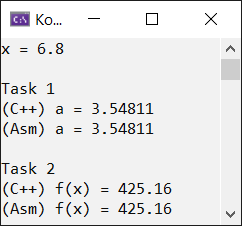
**Результаты работы:**



**Рисунок 1.** Формула первой задачи



**Рисунок 2.** Формула второй задачи

**Рисунок 3** Результат работы

**Вывод:** в ходе выполнения лабораторной работы были получены навыки работы с командами сопроцессора, реализации вычислений и сравнений с участием вещественных чисел, программирования модуля на Assembler.