МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

#### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ

БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

Отчет по дисциплине

«Цифровые устройства и микропроцессоры»

Лабораторная работа №4

«МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ»

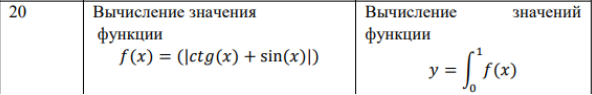
Вариант №20

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИНББ-3301-01-00 | \_\_\_\_\_\_\_\_\_\_\_\_ | Д.А. Шатов |
| Проверил: доцент кафедры РЭС | \_\_\_\_\_\_\_\_\_\_\_\_ | М.А. Земцов |

#### Киров 2023

**Цель работы:** знакомство с технологией применения языка ассемблера при разработке программного обеспечения на языках высокого уровня.

**Вариант 20**



**Ход работы:**

1. Ввод значений в файле Source.cpp, в функции main()

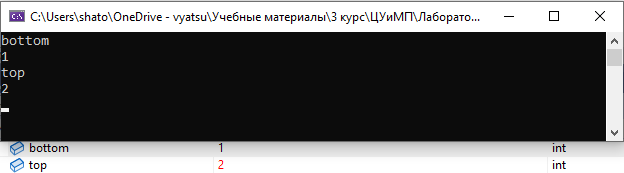


Рисунок 1 – Ввод переделов определенного интеграла

1. Вызов из файла ассемблера функции языка Cи для вычисления значения функции нижнего предела и получаем результат.



Рисунок 2 –значения функции нижнего предела  


Рисунок 3 – значения функции значение верхнего предела

1. Потом значение результата функции языка Си из регистра ST0 записывается в переменную bottom.



Рисунок 4 – Значение результата функции языка Си из регистра ST0 записывается в переменную bottom

1. Потом значение результата функции языка Си из регистра ST0 записывается в переменную top.



Рисунок 5 - Значение результата функции языка Си из регистра ST0 записывается в переменную top, после модуля.

1. Разность значений верхнего и нижнего пределов интеграла. И результат вычисления записываем в переменной razn.

Рисунок 6 – Разность операндов

1. Выводим результат в функцию main()

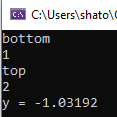
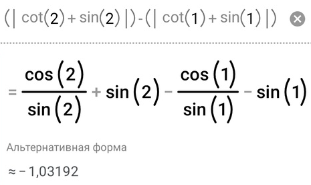


Рисунок 7 – Вывод результата в основном модуле

1. Верификация программы.



**Код программы:**

**Source.cpp:**

#include <iostream>

using namespace std;

extern "C" float sumr(int bottom, int top);

int main()

{

int bottom, top;

cout << "bottom" << endl; cin >> bottom;

cout << "top" << endl; cin >> top;

double y = sumr(bottom, top);

cout << "y = " << y << endl;

return 0;

}

**int.asm:**

.586

.MODEL flat,C

.DATA

down DD 0.0

up DD 0.0

razn DD 0.0

.CODE

extern fun\_el:near

public sumr

sumr proc C

push ebp

mov ebp,esp

push dword ptr [ebp + 8]

call fun\_el

fabs

fld down

fadd

fstp down

push dword ptr [ebp + 12]

call fun\_el

fabs

fld up

fadd

fstp up

fld up

fsub down

fstp razn

fld razn

mov esp,ebp

pop ebp

ret

sumr endp

End

**func.c:**

#include "math.h"

extern float fun\_el(int x)

{

float f = 0;

f = (1 / tan(x)) + sin(x);

if (f < 0) f = -f;

return f;

}

**Вопрос для защиты:**

**Код на С++:**#include <iostream>

#include <cmath>

using namespace std;

float fun\_el(int x) {

float f = 0;

f = (1 / tan(x)) + sin(x);

if (f < 0) f = -f;

return f;

}

float sumr(int bottom, int top) {

float bottomValue = fun\_el(bottom);

float topValue = fun\_el(top);

float difference = fabs(topValue) - fabs(bottomValue);

return difference;

}

int main() {

int bottom, top;

cout << "bottom" << endl; cin >> bottom;

cout << "top" << endl; cin >> top;

float Result = sumr(bottom, top);

cout << "R = " << Result << endl;

return 0;

}

Для точной работы программы, значение bottom,top – зададим через код.

Время работы программы на ассемблере,c++,c: **25** **ms**  
  
  


Время работы программы на C++: **21 ms**  
  




Вывод: Многомодульный проект работает быстрее на 4 мс, нужно учитывать что в ходе лабораторной работы проект на чистом с++ - написан более грамотно и оптимизирован, в отличие от многомодульного проекта. В реальности многомодульный проект должен работать чуть быстрее из-за наличия там функции на ассемблере.