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 №13

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИНББ-3301-01-00 | \_\_\_\_\_\_\_\_\_\_\_\_ | А.Н. Пономарев |
| 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.

**Ход работы:**

1. Изначально вводятся значение пределов в файле glav.cpp, в функции main()

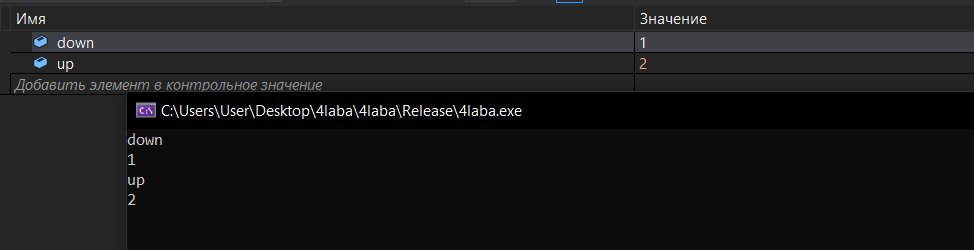


Рисунок 1 – Ввод переделов определенного интеграла

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 down.



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

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



Рисунок 5 - Значение результата функции языка Си из регистра ST0 записывается в переменную up, 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()

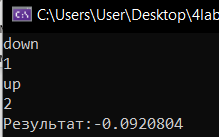


Рисунок 7 – Вывод результата в основном модуле

1. Верификация программы.



**Код программы:**

**glav.cpp:**

#include <iostream>

using namespace std;

extern "C" float sumr(int down, int up);

int main()

{

int down, up;

cout << "down" << endl; cin >> down;

cout << "up" << endl; cin >> up;

double Result = sumr(down, up);

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

return 0;

}

**Source.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

**dop.cpp:**

#include "math.h"

extern float fun\_el(int z)

{

float f = 0;

f = 1 / tan(z);

f = cbrt(f);

return f;

}

**Вопрос для защиты:**

**Код на С++:**#include <iostream>

#include <cmath>

using namespace std;

float fun\_el(int z) {

float f = 0;

f = 1 / tan(z);

f = cbrt(f);

return f;

}

float sumr(int down, int up) {

float downValue = fun\_el(down);

float upValue = fun\_el(up);

float difference = fabs(upValue) - fabs(downValue);

return difference;

}

int main() {

int down, up;

cout << "down: ";

cin >> down;

cout << "up: ";

cin >> up;

float Result = sumr(down, up);

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

return 0;

}

Для точной работы программы, значение down,up – зададим через код.

Время работы программы на ассемблере,c++,c: **14,33** ms  
  
  


Время работы программы на с++:**17,33 ms**  
  




Вывод: Многомодульный проект работает быстрее на 2 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.