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

Отчёт по дисциплине

«Цифровые устройства и микропроцессоры»

Лабораторная работа №4

Вариант-16

Модульное программирование

| Выполнил: | студент группы ИНБб-31 |  | \_\_\_\_\_\_\_\_\_\_\_ |  | А.И. Парфенов |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 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.

**Исходные данные:**

| Функция C | Функция Ассемблера |
| --- | --- |
|  |  |

**Ход работы:**

Для смешанного программирования на языке высокого уровня и ассемблере используются многомодульные проекты, структура вызова функций которых изображена на рисунке 1.

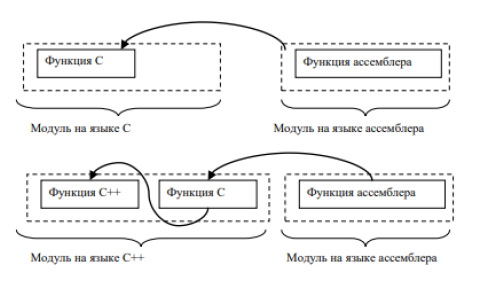


Рисунок 1 - Структура вызова функций

При этом должны выполняться некоторые условия, например, для вызова функций из модуля C на ассемблере они должны быть объявлены как extern, а в модуле ассемблера должна использоваться совместимая модель памяти и функция должна быть объявлена в сегменте .CODE с директивой EXTERN.

На рисунке 2 представлен код для вычисления элемента ряда на языке C.

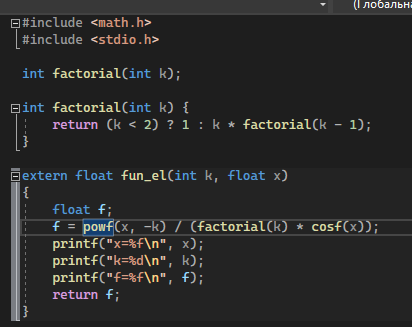


Рисунок 2 - Вычисление элемента ряда

Функция fun\_el объявлена с помощью ключевого слова extern и может быть вызвана в модуле ассемблера. На рисунке 3 представлен фрагмент ассемблерного кода для расчета суммы элементов ряда.

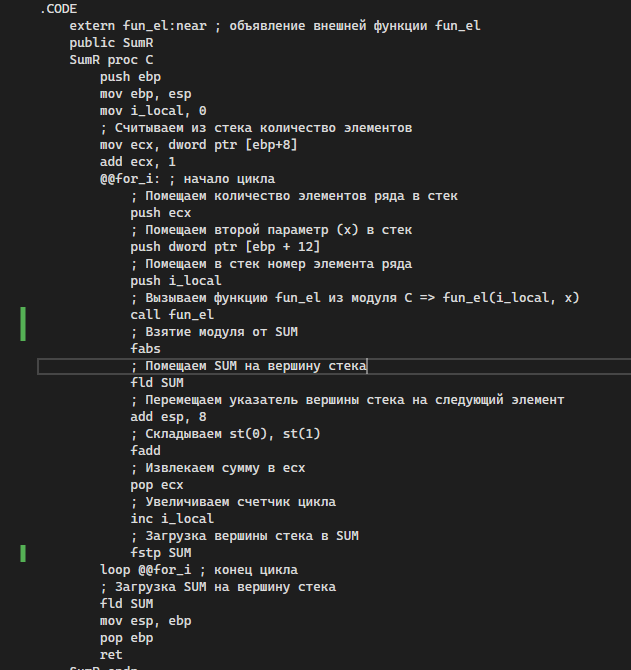


Рисунок 3 - Ассемблерный модуль

Как видно, в сегменте .CODE функция fun\_el объявлена с директивой extern. Результат вызова данной функции помещается в стек, после чего можно взять модуль от вычисленного значения. Данная функция для вычисления суммы элементов ряда SumR объявлена с помощью директивы public для того, чтобы ее можно было вызвать из модуля на C++. Код основного модуля на С++ представлен на рисунке 4.

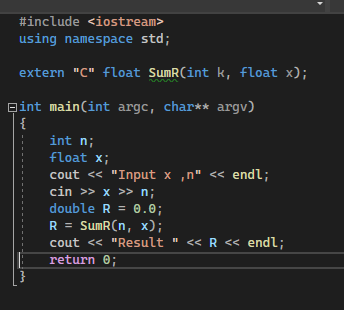


Рисунок 4 - Модуль С++

Вначале объявляется прототип функции SumR, описывающий ее как внешнюю. Затем идет ввод параметров x, n и вывод результата в консоль.

На рисунке 5 показан вывод результатов при значении x = 2 и n = 5.

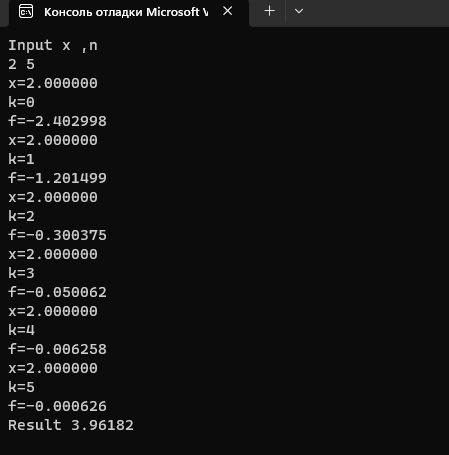


Рисунок 5 - Пример

На калькуляторе можно проверить полученный результат (рисунок 6).

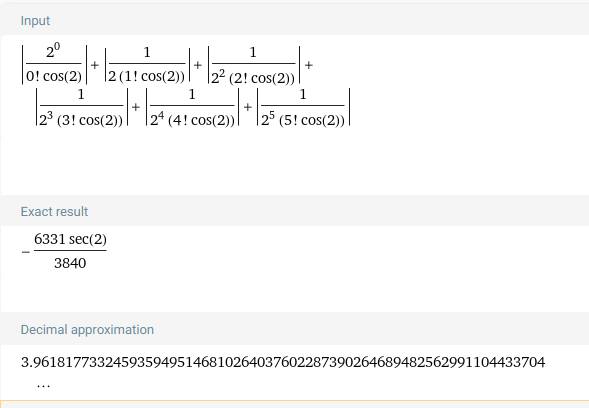


Рисунок 6 - Проверка результата

Результаты совпали.

**Вывод:** в ходе выполнения лабораторной работы были изучены принципы организации многомодульных проектов на языках высокого уровня с применением ассемблерных модулей.

**Код программы:**

Source.cpp

#include <iostream>

using namespace std;

extern "C" float SumR(int k, float x);

int main(int argc, char\*\* argv)

{

int n;

float x;

cout << "Input x ,n" << endl;

cin >> x >> n;

double R = 0.0;

R = SumR(n, x);

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

return 0;

}

file1.c

#include <math.h>

#include <stdio.h>

int factorial(int k);

int factorial(int k) {

return (k < 2) ? 1 : k \* factorial(k - 1);

}

extern float fun\_el(int k, float x)

{

float f;

f = powf(x, -k) / (factorial(k) \* cosf(x));

printf("x=%f\n", x);

printf("k=%d\n", k);

printf("f=%f\n", f);

return f;

}

file2.asm

.586

.MODEL flat, C

.DATA

; Промежуточная сумма

SUM DD 0.0

; Номер элемента ряда

i\_local DD 0

.CODE

extern fun\_el:near ; объявление внешней функции fun\_el

public SumR

SumR proc C

push ebp

mov ebp, esp

mov i\_local, 0

; Считываем из стека количество элементов

mov ecx, dword ptr [ebp+8]

add ecx, 1

@@for\_i: ; начало цикла

; Помещаем количество элементов ряда в стек

push ecx

; Помещаем второй параметр (x) в стек

push dword ptr [ebp + 12]

; Помещаем в стек номер элемента ряда

push i\_local

; Вызываем функцию fun\_el из модуля C => fun\_el(i\_local, x)

call fun\_el

; Взятие модуля от SUM

fabs

; Помещаем SUM на вершину стека

fld SUM

; Перемещаем указатель вершины стека на следующий элемент

add esp, 8

; Складываем st(0), st(1)

fadd

; Извлекаем сумму в ecx

pop ecx

; Увеличиваем счетчик цикла

inc i\_local

; Загрузка вершины стека в SUM

fstp SUM

loop @@for\_i ; конец цикла

; Загрузка SUM на вершину стека

fld SUM

mov esp, ebp

pop ebp

ret

SumR endp

End