|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Теоретической и прикладной математики | | |
|  | | |
| Расчетно-графическое задание | | |
| по дисциплине «Низкоуровневое программирование» | | |
|  | | |
| **ПРОГРАММИРОВАНИЕ СОПРОЦЕССОРА** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-02 |
| Вариант: | 7 |
| Студент: | Сидоров Даниил, |
|  | Дюков Богдан |
| Преподаватель: | Еланцева И. Л., |
|  | Сивак М. А.  . |
|
|  |  |
|  | | |
| Новосибирск | | |
| 2022 | | |

1. **Цель работы**

Изучить архитектуру и средства программирования сопроцессора (модуля

операций с плавающей точкой) на языке ассемблера и приобрести практические навыки работы с ними.

1. **Содержание работы**

* Изучить архитектуру и средства программирования сопроцессора на языке ассемблера. Изучить численный метод решения нелинейного уравнения.
* Написать программу, вычисляющую значение функции f(x) в произвольной точке x. В нашем случае f(x) = x\*ln(x).
* Программа должна состоять из модулей на С++ и ассемблере, причем в модуле на С++ осуществляется ввод-вывод, а все вычисления – в модуле на ассемблере.
* На вход ассемблеровской процедуре подается вещественное число одинарной точности. Выходными данными является вычисленное значение функции (двойная точность).
* Отладить программу, убедиться в правильности ее работы на тестовых примерах.

1. **Алгоритм**

Программа содержит в себе два модуля. Несмотря на то, что обязательные проверки введенных данных сделаны в модуле на ассемблере, модуль на С++, кроме ввода вещественного числа x и вывода результата вычисления f(x), содержит в себе дополнительные проверки для полного исключения ошибочного ввода. Модуль на С++ работает по следующему алгоритму:

1. Предложение ввода в консоли строки, которая будет содержать в себе вещественное число x;
2. Проверка введенной строки на пустоту, а также на то, чтобы строка не содержала в себе лишних символов;
3. Передача в качестве параметра ассемблеровской процедуры числа, преобразованного из строки в тип float, и получение решения f(x);
4. Анализ полученного результата, если он равен 0 или экспоненте, а число x не может давать таких результатов, то ввод был некорректен;
5. В остальных случаях выводим результат f(x) на экран.

Модуль на ассемблере содержит в себе процедуру xlnx, которая работает по следующему алгоритму:

1. Полученное из модуля C++ число x проверяется на вхождение в область определения функции натурального логарифма (x > 0), в случае не прохождения проверки процедура возвращает 0;
2. Проверка числа x на отсутствие погрешности (число не должно содержать в себе более 7 знаков, в связи с одинарной точностью float), в случае не прохождения проверки процедура возвращает экспоненту;
3. Вычисление f(x) и возврат результата.

Следует отметить, что все проверки и вычисления на ассемблере осуществлялись с помощью команд сопроцессора.

1. **Текст програ****ммы**

Файл xlnx.asm:

.386

.model flat

.data

x real4 ?

max real4 10000000.0

exp real4 2.718281

.code

@xlnx@4 proc

;Проверка полученного x на положительность

mov eax,[ecx]

mov x,eax ;Кладем число в переменную x

finit

fld x ;ST0=x

fldz ;ST0=0 ST1=x

fcompp ;Сравниваем ST0 и ST1

fstsw ax ;Сохраняем флаги в AX

sahf ;Загружаем флаги в регистр флагов EFLAGS

jnc error ;Если ST0>=ST1, то флаг CF=0 => Обрабатываем ошибку

;Проверка полученного x на отсутствие погрешности

fld max ;ST0=10000000.0

fld x ;ST0=x ST1=10000000.0

fcompp ;Сравниваем ST0 и ST1

fstsw AX ;Сохраняем флаги в AX

sahf ;Загружаем флаги в регистр флагов EFLAGS

jnc error2 ;Если ST0>=ST1, то флаг CF=0 => Обрабатываем ошибку

;Вычисление f(x)

fld1 ;ST0=1

fld x ;ST0=X ST1=1

fyl2x ;ST0=log2x

fldl2e ;ST0=log2e ST1=log2x

fdivp ;ST0=lnx

fld x ;ST0=x ST1=lnx

fmulp ;ST0=xlnx

jmp finish

;Обработка ситуации x<=0, возвращаем 0

error:

fldz

jmp finish

;Обработка ситуации с появлением погрешности (x>=10000000), возвращаем e

error2:

fld exp

finish:

ret

@xlnx@4 endp

end

Файл Main.cpp:

// Для экспоненты M\_E

#define \_USE\_MATH\_DEFINES

#include <math.h>

#include <iostream>

#include <iomanip>

#include <string>

// Функция получает число x, возвращает x\*lnx

extern "C" double \_\_fastcall xlnx(float\* x);

int main()

{

// Строка для сохранения введенного числа

std::string xStr;

std::cout << "Enter the x: ";

getline(std::cin, xStr);

// Проверка введенной строки на пустоту

if (xStr == "")

{

std::cout << "Error! An empty string has been entered";

return 0;

}

// Счетчик точек в числе

int pointsNumber = 0;

// Проверяем введенную строку (чтобы она могла содержать в себе только числа, точку и знак минуса)

for (int i = 0; i < xStr.length(); i++)

{

// Пропускаем минус в самом начале

if (i == 0 && xStr[i] == '-')

{

continue;

}

// Пропускаем точку внутри числа, но проверяем, чтобы она встречалась только один раз

if (i > 0 && i < xStr.length() - 1 && xStr[i] == '.')

{

pointsNumber++;

// Проверяем количество знаков в числе

if (xStr.length() - 1 > 7)

{

std::cout << "Error! x contains more than 7 characters, accuracy is not guaranteed";

return 0;

}

continue;

}

// Проверяем очередной индекс

if (xStr[i] < '0' || xStr[i] > '9' || pointsNumber > 1)

{

std::cout << "Error! A non-numeric value has been entered";

return 0;

}

}

// Конвертируем строку в тип float

float x = std::stof(xStr);

// Сохраняем результат f(x)

double result = xlnx(&x);

// Машинный ноль для сравнения чисел типа double

double eps = 1e-06;

// Если результат равен 0, а x не равен 1, то введенный x <= 0

// Если результат равен экспоненте, а x не равен экспоненте, то x содержит в себе более 7 знаков, а значит обладает погрешностью

if (fabs(result - 0.0) < eps && fabs(x - 1.0f) > eps)

{

std::cout << "Error! x <= 0";

return 0;

}

else if (fabs(result - M\_E) < eps && fabs(x - M\_E) > eps)

{

std::cout << "Error! x contains more than 7 characters, accuracy is not guaranteed";

return 0;

}

// Выводим на экран результат f(x)

std::cout << "Answer: " << std::setprecision(16) << result;

return 0;

}

1. **Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Введенная строка** | **Результат** | **Комментарий** |
| 1 | 1 | 0 | Простой тест x = 1, 1\*ln1=0 |
| 2 | 10 | 23.02585092994046 | Еще один тест с получением численного ответа. Так как в результате получается число двойной точности, выводим первые 16 знаков |
| 3 | 0 | Error! x <= 0 | Значения 0 и -152 не входит в область определения логарифма (x > 0) |
| 4 | -152 |
| 5 | 10000000000 | Error! x contains more than 7 characters, accuracy is not guaranteed | Ввод слишком большого числа для типа float, у которого одинарная точность. Для исключения погрешности считаем такой ввод ошибочным |
| 6 | Пустая строка | Error! An empty string has been entered | Ошибочный ввод, обработанный в C++ |
| 7 | 151faga1521gf | Error! A non-numeric value has been entered |

1. **Вывод**

Контрольные вопросы проработаны.