Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информатики

Отчет по лабораторной работе №7-8

Программирование арифметического сопроцессора

Выполнил:

студент гр. 153504

Тиханёнок И. А.

Проверила:

Калиновская А. А.

Минск 2023

1. **Цель работы**

Ознакомиться с предназначением арифметического сопроцессора. Рассмотреть строение и основные принципы работы сопроцессора. Изучить команды, доступные при использовании сопроцессора. Научиться работать с сопроцессором в ходе выполнения лабораторного задания.

1. **Постановка задачи**

***Задание к лабораторной работе 7***

Написать программу, находящую решение квадратного уравнения с помощью сопроцессора.

***Задание к лабораторной работе 8***

Значение аргумента изменяется от до с шагом . Для каждого найти значения функции , суммы и число итераций , при котором достигается требуемая точность . Результат вывести в виде таблицы. Значения и вводятся с клавиатуры.

1. **Теоретические сведения**

Использование сопроцессора  позволяет значительно ускорить работу программ, выполняющих расчеты с высокой точностью, тригонометрические вычисления и обработку информации, которая должна быть представлена в виде действительных чисел. Сопроцессор подключается к системной шине параллельно с центральным процессором (CPU) и может работать только совместно с ним. Все команды попадают в оба процессора, а выполняет каждый свои. Сопроцессор не имеет своей программы и не может осуществлять выборку команд и данных. Это делает центральный процессор. Сопроцессор перехватывает с шины данные и после этого реализует конкретные действия по выполнению команды. Два процессора работают параллельно, что повышает эффективность системы.  Но возникают ситуации, когда их работа требует синхронизации (из-за разницы во времени выполнения команд).

1. **Программная реализация**

В качестве средств для написания программы используются язык C++ и среда разработки Microsoft Visual Studio.

***Результаты:***

*Лабораторная работа 7.*

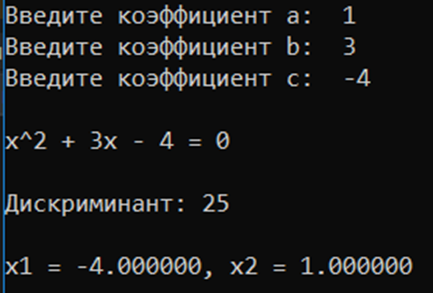


Рисунок 1. Квадратное уравнение с двумя корнями

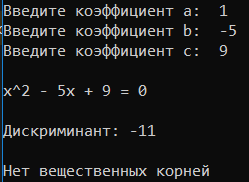


Рисунок 2. Квадратное уравнение, не имеющее корней

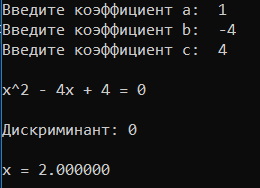


Рисунок 3. Квадратное уравнение с одним корнем

*Лабораторная работа 8.*

Была выбрана функция .

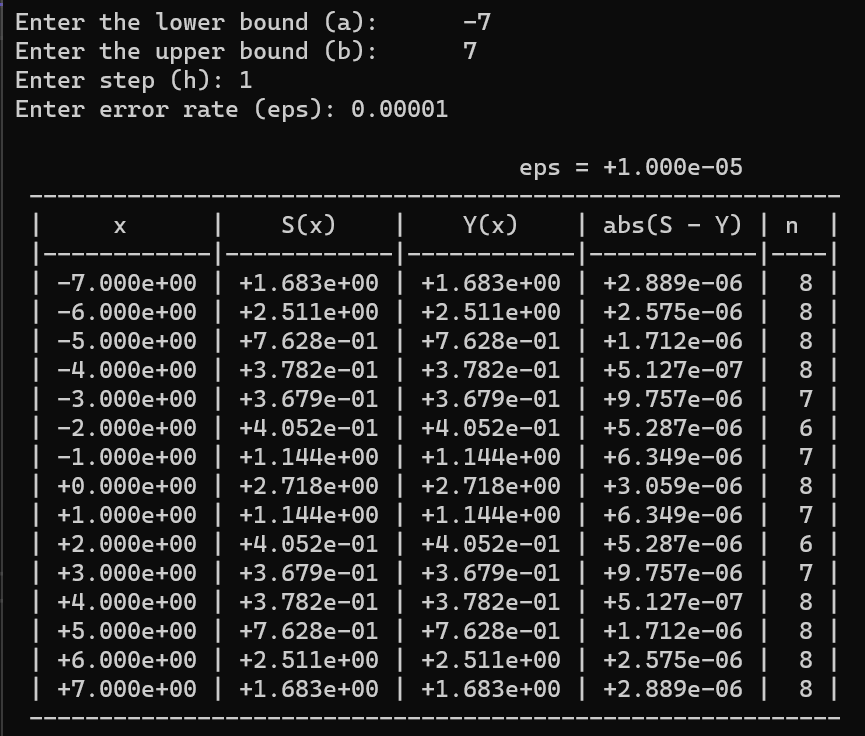


Рисунок 4. Результат выполнения программы

1. **Выводы**

В ходе выполнения лабораторной работы были изучены понятия арифметического сопроцессора, сопроцессорных конфигураций, программной модели сопроцессора. Выяснены режимы работы и состояния арифметического сопроцессора, система команд сопроцессора, особенности задания команд, различные группы команд сопроцессора (команды управления, команды передачи данных, команды загрузки, команды запоминания, арифметические команды).

**Список литературы**

Волорова Н. А. Лабораторный практикум по курсу «Архитектура вычислительных систем» для студентов специальности «Информатика» /985-444-487-2 – Мн.: БГУИР, 2003. — 32 с.: ил.

**Приложение 1. Исходный код программы (ЛР 7)**

#include <iostream>

#include <string>

#include <windows.h>

#include <functional>

#include <regex>

using namespace std;

string Solve(long double a, long double b, long double c)

{

const long double LD\_4 = 4.0L;

if (a == 0) {

return "Not square";

}

long double D;

\_\_asm

{

FLD b

FLD b

FMUL

FLD LD\_4

FLD a

FLD c

FMUL

FMUL

FSUB

FSTP D

}

cout << "\nDescr: " << D << '\n';

if (!isfinite(D)) {

return "Descr not final number";

}

if (D < 0.0L) {

return "No weshestv korney";

}

if (D == 0.0L) {

long double x;

\_\_asm

{

FLD b

FCHS

FLD a

FLD a

FADD

FDIV

FSTP x

}

return "x = " + to\_string(x);

}

if (D > 0.0L) {

long double x1, x2;

\_\_asm

{

FLD b

FCHS

FLD D

FSQRT

FSUB

FLD a

FLD a

FADD

FDIV

FSTP x1

}

\_\_asm

{

FLD b

FCHS

FLD D

FSQRT

FADD

FLD a

FLD a

FADD

FDIV

FSTP x2

}

return "x1 = " + to\_string(x1) + ", " +

"x2 = " + to\_string(x2);

}

return "Error";

}

template<typename ReturnValue>

ReturnValue CorrectInput(const string& variable\_name, const std::regex& pattern,

std::function<ReturnValue(string& input)> converter) {

string value;

do {

cout << "Enter " << variable\_name << ":\t";

cin >> value;

} while (!regex\_match(value, pattern));

return converter(value);

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

long double a = CorrectInput<long double>(std::string("koaf а"),

std::regex("-?[0-9]+,?[0-9]\*"),

[](std::string& s) {return atof(s.c\_str()); });

long double b = CorrectInput<long double>(std::string("koaf b"),

std::regex("-?[0-9]+,?[0-9]\*"),

[](std::string& s) {return atof(s.c\_str()); });

long double c = CorrectInput<long double>(std::string("koaf c"),

std::regex("-?[0-9]+,?[0-9]\*"),

[](std::string& s) {return atof(s.c\_str()); });

cout << '\n';

if (a != 1)

cout << ((a == -1) ? '-' : a);

cout << "x^2 " << ((b < 0) ? '-' : '+') << ' ' << abs(b) << "x "

<< ((c < 0) ? '-' : '+') << ' ' << abs(c) << " = 0\n";

cout << '\n' << Solve(a, b, c) << '\n';

return 0;

}

**Приложение 2. Исходный код программы (ЛР 8)**

#include <iostream>

#include <iomanip>

#include <windows.h>

#include <math.h>

#include <string>

#include <regex>

#include <functional>

using namespace std;

void Solve(long double a, long double b, long double h, long double eps)

{

cout << '\n';

cout << setprecision(3) << scientific << showpos <<

" eps = " << eps << " " << '\n';

cout << " ---------------------------------------------------------- " << '\n';

cout << " | x | S(x) | Y(x) | abs(S - Y) | n | " << '\n';

cout << " |------------|------------|------------|------------|----| " << '\n';

for (long double x = a; x < b + h / 2; x += h) {

int n = 0;

long double Y, S, f, p, intp, fracp, trash;

//Y = pow(exp, cos(x)) \* cos(sin(x));

\_\_asm

{

FLD x

FCOS

FLDL2E

FMUL

FSTP p

}

\_\_asm

{

FLD p

FLD1

FSCALE

FSTP intp

FSTP trash

}

\_\_asm

{

FLD1

FLD p

FPREM

F2XM1

FADD

FSTP fracp

}

//Y = pow(exp, cos(x)) \* cos(sin(x));

\_\_asm

{

FLD intp

FLD fracp

FMUL

FLD x

FSIN

FCOS

FMUL

FSTP Y

}

//S = 1.0, f = 1.0;

\_\_asm

{

FLD1

FSTP S

FLD1

FSTP f

}

while (abs(S - Y) > eps) {

n++;

//f \*= n;

\_\_asm

{

FLD f

FILD n

FMUL

FSTP f

}

//S += cos(n \* x) / f;

\_\_asm

{

FLD S

FILD n

FLD x

FMUL

FCOS

FLD f

FDIV

FADD

FSTP S

}

}

cout << setprecision(3) << scientific << showpos <<

" | " << x << " | " << S << " | " << Y << " | " << abs(S - Y) << " | " <<

noshowpos << right << setw(2) << n << " | \n";

}

cout << " ---------------------------------------------------------- \n";

}

template<typename ReturnValue>

ReturnValue CorrectInput(const string& variable\_name, const std::regex& pattern,

std::function<ReturnValue(string& input)> converter) {

string value;

do {

cout << "Enter " << variable\_name << ":\t";

cin >> value;

} while (!regex\_match(value, pattern));

return converter(value);

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

float a = CorrectInput<float>(std::string("the lower bound (à)"),

std::regex("-?[0-9]+,?[0-9]\*"),

[](std::string& s) {return atof(s.c\_str()); });

float b = CorrectInput<float>(std::string("the upper bound (b)"),

std::regex("-?[0-9]+,?[0-9]\*"),

[](std::string& s) {return atof(s.c\_str()); });

float h = CorrectInput<float>(std::string("step (h)"),

std::regex("-?[0-9]+,?[0-9]\*"),

[](std::string& s) {return atof(s.c\_str()); });

float eps = CorrectInput<long double>(std::string("error rate (eps)"),

std::regex("-?[0-9]+\.?[0-9]\*"),

[](std::string& s) {return atof(s.c\_str()); });

Solve(a, b, h, eps);

return 0;

}