**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №8**

**по дисциплине «Организация ЭВМ и систем»**

Тема: Обработка вещественных чисел. Программирование математического сопроцессора.

|  |  |  |
| --- | --- | --- |
| Студент гр. 8383 |  | Ларин А. |
| Преподаватель |  | Ефремов М.А, |

Санкт-Петербург

2019

**Цель работы.**

Научится обрабатывать вещественные числа при помощи математического сопроцессора на языке ассемблера. Научится рассчитывать математические функции из многих действий при помощи математического сопроцессора.

**Основные теоретические положения.**

Важной частью архитектуры микропроцессоров Intel является наличие устройства для обработки числовых данных в формате с плавающей точкой, называемого математическим сопроцессором. Архитектура компьютеров на базе микропроцессоров вначале опиралась исключительно на целочисленную арифметику. С ростом мощи стали появляться устройства для обработки чисел с плавающей точкой. В архитектуре семейства микропроцессоров Intel 8086 устройство для обработки чисел с плавающей точкой появилось в составе компьютера на базе микропроцессора i8086/88 и получило название математический сопроцессор или просто сопроцессор. Выбор такого названия был обусловлен тем, что,

* во-первых, это устройство было предназначено для расширения вычислительных возможностей основного процессора;
* во-вторых, оно было реализовано в виде отдельной микросхемы, то есть его присутствие было необязательным. Микросхема сопроцессора для микропроцессора i8086/88 имела название i8087.

Основные возможности математического сопроцессора:

* полная поддержка стандартов IEEE-754 и 854 на арифметику с плавающей точкой. Эти стандарты описывают как форматы данных, с которыми должен работать сопроцессор, так и набор реализуемых им функций;
* поддержка численных алгоритмов для вычисления значений тригонометрических функций, логарифмов и т. п.;
* обработка десятичных чисел с точностью до 18 разрядов, что позволяет сопроцессору выполнять арифметические операции без округления над целыми десятичными числами со значениями до 1018;
* обработка вещественных чисел из диапазона ±3.37х10-4932…1.18х10+4932.

Программная модель сопроцессора представляет собой совокупность регистров, каждый из которых имеет свое функциональное назначение. В программной модели сопроцессора можно выделить три группы регистров:

* Восемь регистров r0…r7, составляющих основу программной модели сопроцессора — стек сопроцессора. Размерность каждого регистра 80 битов. Такая организация характерна для устройств, специализирующихся на обработке вычислительных алгоритмов.
* Три служебных регистра:
  + регистр состояния сопроцессора swr (Status Word Register — регистр слова состояния) — отражает информацию о текущем состоянии сопроцессора;
  + управляющий регистр сопроцессора cwr (Control Word Register — регистр слова управления) — управляет режимами работы сопроцессора;
  + регистр тегов twr (Tags Word Register — слово тегов) — используется для контроля за состоянием каждого из регистров стека.
* Два регистра указателей — данных dpr (Data Point Register) и команд ipr (Instruction Point Register). Они предназначены для запоминания информации об адресе команды, вызвавшей исключительную ситуацию и адресе ее операнда. Эти указатели используются при обработке исключительных ситуаций (но не для всех команд).

Все указанные регистры являются программно доступными. Однако к одним из них доступ получить достаточно легко, для этого в системе команд сопроцессора существуют специальные команды. К другим регистрам получить доступ сложнее, так как специальных команд для этого нет, поэтому необходимо выполнить дополнительные действия.

* **Регистр состояния swr –** отражает текущее состояние сопроцессора после выполнения последней команды. В регистре swr содержатся поля, позволяющие определить: какой регистр является текущей вершиной стека сопроцессора, какие исключения возникли после выполнения последней команды, каковы особенности выполнения последней команды (некий аналог регистра флагов основного процессора).
* **Регистр управления работой сопроцессора cwr** – определяет особенности обработки числовых данных. С помощью полей в регистре cwr можно регулировать точность выполнения численных вычислений, управлять округлением, маскировать исключения
* Регистр тегов **twr** – представляет собой совокупность двухбитовых полей. Каждое поле соответствует определенному физическому регистру стека и характеризует его текущее состояние. Команды сопроцессора используют этот регистр, например, для того, чтобы определить возможность записи значений в эти регистры. Изменение состояния любого регистра стека отражается на содержимом соответствующего этому регистру 2-битового поля регистра тега.

Регистровый стек сопроцессора организован по принципу кольца. Cреди восьми регистров, составляющих стек, нет такого, который является вершиной стека. Все регистры стека с функциональной точки зрения абсолютно одинаковы и равноправны. Вершина в кольцевом стеке сопроцессора является плавающей. Контроль текущей вершины осуществляется аппаратно с помощью 3-битового поля top регистра swr.

В поле top фиксируется номер регистра стека r0…r7, являющегося в данный момент текущей вершиной стека.

Команды сопроцессора оперируют не физическими номерами регистров стека r0…r7, а их логическими номерами st(0)…st(7). C помощью логических номеров реализуется относительная адресация регистров стека сопроцессора.

**Задание**

Разработать подпрограмму на языке Ассемблера, обеспечивающую вычисление заданной математической функции с использованием математического сопроцессора. Подпрограмма должна вызываться из головной программы, разработанной на языке С. При этом должны быть обеспечены заданный способ вызова и обмен параметрами. Альтернативный вариант реализации: разработать на языке Ассемблера фрагмент программы, обеспечивающий вычисление заданной математической функции с использованием математического сопроцессора, который включается по принципу in-line в программу, разработанную на языке С. Проверить корректность выполнения вычислений для нескольких наборов исходных данных.

ВАРИАНТ 6.

\* function

Name Acos - compute acos

Usage double Acos (double \*xP);

Prototype in math.h

Description Computes acos of the number pointed to by xP.

Arguments to acos must be in the range -1 to 1, acos returns a value in the range 0 to pi.

Use the trig identities acos (x) = atan (sqrt (1-x^2) / x) \*/

**Выполнение**

Вариант 6

Фрагмент программы написан на языке ассемблера и включен в основную программу написанную на языке C по принципу in-line.

В головной программе происходит считывание аргумента *x* из стандартного потока ввода. С его помощью вычисляется эталонное значение функции при помощи стандартной библиотеки math.h, а затем аргумент передается в функцию Acos, написанную при помощи in-line вставки языка ассемблера.

double Acos(double \*xP)

Данная функция принимает на вход аргумент *x,* возвращает значение функции Acos(x).

В начале программы инициализируются переменные, которые будут использованы из языка ассемблера. Затем начинается in-line вставка.

В данной функции происходит вычисление значения функции:

atan(sqrt(1-x^2) / x).

Расчет начинается с глубины функции. В начале в стек сопроцессора дважды помещается аргумент *x* (в st(0) и st(1)) коммандой fld, затем перемножается командой, что дает нам значение d на вершине стека(st(0)). Затем командой fld1 на вершину стека помещается единица, затем значения st(0) и st(1) меняются местами командой fxch что дает нам значение на вершине стека и значение 1 сразу после. Далее командой fsub вычисляется значение и кладется на вершину стека. Затем командой fsqrt из этого значения берется корень(результат в st(0)), на вершину стека добавляется значение *x* (st(0)→st(1);x→st(0)) и командой fdiv происходит деление двух верхних значений в стеке. Имеем вычисленным значение в st(0). Осталась операция взятия арктангенса, однако для ее вычисление используется два значения со стек, то есть вычисляется значение . По этому предварительно кладем на вершину стека значение 1, и производим вычисление командой fpatan. На вершине стека лежит значение atan(sqrt(1-x^2) / x), что соответствует значению acos(x). Командой fstp достаем значение из стека и кладем его в переменную y.

Однако данное равенство справедливо только для неотрицательных *x.* Потому в конце работы программы делаем проверку на знак числа *x*, и в случае, если он отрицателен добавляем к значению функции pi. Теперь функция справедлива и для отрицательных аргументов.

**Тестирование.**

|  |  |  |
| --- | --- | --- |
| Значение X | Эталонное значение acos(x) | Рассчитанное значение Acos(x) |
| 1 | 0 | 0 |
| 0.5 | 1.04198 | 1.04198 |
| 0 | 1.570796 | 1.570796 |
| -0.42 | 2.004242 | 2.004242 |
| -0.5 | 2.094395 | 2.094395 |
| -1 | 3.141593 | 3.141593 |

**Выводы.**

В результате работы были разобраны некоторые базовые концепции языка ассемблера. Была изучены методы работы с числами с плавающей точкой. Был изучен принцип работы математического сопроцессора, и с его использованием написана функция расчета математического выражения со значениями с плавающей точкой двойной точности.

Приложение

LR8.Cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <math.h>

double Acos(double \*xP);

/\*function

Name Acos - compute acos

Usage double Acos(double \*xP);

Prototype in math.h

Description Computes acos of the number pointed to by xP.

Arguments to acos must be in the range - 1 to 1, acos returns a value in the range 0 to pi.

Use the trig identities acos(x) = atan(sqrt(1 - x ^ 2) / x) \*/

int main()

{

double x;

printf("Enter x: ");

scanf("%lf", &x);

printf("x: \t\t%lf\n", x);

if (x < -1 || x>1) {

printf("Argument should be within [-1;1] range!\n");

return 0;

}

printf("math.h acos(x): %lf\n", acos(x));

printf("asm Acos(x): \t%lf\n", Acos(&x));

return 0;

}

double Acos(double \*xP)

{

double x = \*xP;

double y = -1;

\_asm {

fld x; //x -> st(0);

fld x; //x -> st(1)

fmul; //x^2 -> st(0)

fld1; //1 -> st(0); x^2 -> st(1)

fxch st(1); //x^2 -> st(0); 1 -> st(1);

fsub; //1-x^2 -> st(0)

fsqrt; //sqrt(1-x^2) -> st(0)

fld x; //x -> st(0); sqrt(1-x^2) -> st(1)

fdiv; //sqrt(1-x^2) / x -> st(0)

fld1; //1 -> st(0); sqrt(1-x^2) / x -> st(1)

fpatan; //atan(st(1)/st(0)) -> st(0) == atan(sqrt(1-x^2) / x) -> st(0)

fstp y; //pop st(0) -> y

fldz; //0 -> st(0); x-> st(1)

fld x; //x -> st(0);

fcom; //cmp 0,x

fstsw ax; //забираем результат из сопроцессора

sahf //заносим в регистр флагов

jae c\_end; //if x >= 0

fld y; //y -> st(0);

fldpi; //pi -> st(0); y -> st(1);

fadd; //pi + y -> st(0)

fstp y; //pop st(0) -> y

c\_end:

}

return y;

}