ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)»

Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе № 8

по дисциплине «Организация ЭВМ и систем»

Тема: Обработка вещественных чисел. Программирование математического сопроцессора.

Студент гр. 1303 _____ Ягодаров М.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Разработать подпрограмму на языке Ассемблера, обеспечивающую вычисление заданной математической функции с использованием математического 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.

Выполнить трансляцию программы с подготовкой ее ассемблерной версии и отладочной информации. Для выбранного контрольного набора исходных данных прогнать программу под управлением отладчика. При этом для каждой команды сопроцессора следует фиксировать содержимое используемых ячеек памяти, регистров ЦП и численных регистров сопроцессора до и после выполнения этой команды.

Проверить корректность выполнения вычислений для нескольких наборов исходных данных.

Вариант 1.

Вернуть значение многочлена для заданного x.

Выполнение работы.

С помощью головной программы на языке Си производится считывание исходных данных: значения x, количества констант, а также из значения; значение x и констант принимается в виде чисел с плавающей запятой двойной точности.

После ввода числа даннные передаются в математический сопроцессор, написанный на языке Ассемблера.

В сопроцессоре вычисляется значение многочлена с помощью схемы

Горнера. На вход процедуре `poly` подаются данные: в регистре xmm0 содержится x, в регистре rdi — количество констант, в регистре rsi — массив констант.

Значение x перемещается в регистр xmm1 (инструкция `movsd`), а регистр xmm0 обнуляется (инструкция `subsd`). Далее проверяется количество констант: если их число равно нулю, то процедура завершается. Иначе в регистр rcx записывается значение rdx (число констант в массиве), после чего начинается цикл: значение в регистре xmm0 умножается (инструкция `mulsd`) на xmm1 (значение x), затем к регистру xmm0 добавляется (`addsd`) число из массива констант, получаемое обращением по адресу регистра rsi, смещённого на текущий номер константы, хранящийся в регистре rcx.

По итогу цикла в регистре xmm0 будет находиться значение результата вычисления исходного многочлена.

В конце головная программа выводит значение многочлена в консоль.

Выводы

Получены навыки работы со специальными инструкциями Ассемблера для чисел с плавающей запятой. Разработана программа на ЯВУ Си, которая с помощью математического сопроцессора, написанного на языке Ассемблера отображает значение многочлена для заданного x.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>

extern double poly(double x, int n, double *c);

int main() {
 double x;
 printf("Enter x: ");
 scanf("%lf", &x);

 int n;
 printf("Enter number of constants: ");
 scanf("%d", &n);

 double *constants = malloc(n * sizeof(double));
 char c;
 printf("Enter constants: ");
 for (int i = 0; i < n; ++i) {
  scanf("%lf%c", &constants[i], &c);
 }

 double result = poly(x, n, constants);

 printf("(asm) Result is:\n\t%lf\n", result);
 free(constants);

 return 0;
}
```

Название файла: lib.s

```
.global poly

#; Input:
#; x: double  → xmm0
#; n: int     → rdi
#; c: double* → rsi
poly:
 movsd xmm1, xmm0
 subsd xmm0, xmm0
 test  rdi, rdi
 jz   poly_end
 mov  rcx, rdi
horner:
```

```
  mulsd xmm0, xmm1
  addsd xmm0, [rsi + rcx * 8 - 8]
  loop  horner
poly_end:
 ret
```

Название файла: Makefile

```
all: main

main: main.o lib.o
	gcc main.o lib.o -o main -z noexecstack -lm

main.o: main.c
	gcc -c main.c

lib.o: lib.s
	as lib.s -msyntax=intel -mnaked-reg -mmnemonic=intel -o lib.o

clean:
	rm -f *.o main
```