# 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 Бутыло Е.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 5.

Вернуть значение выражения *value ·* 2*exp*, для заданные value и exp.

# Выполнение работы.

С помощью головной программы на языке Си производится считывание исходных данных: значения value, exp.

После ввода числа данные передаются в модуль, написанный на языке Ассемблера. На вход процедуре ldexp подаются данные: в регистре

xmm0 содержится value, в регистре rdi — exp.

Для выполнения поставленной задачи нужно записать необходимые данные в стек математического сопроцессора. Первое значение из rdi

вносим на стек, далее с помощью инструкции fild значение передаём на стек мат. сопроцессора. Последнее значение из xmm0 нельзя напрямую внести на стек, поэтому сначала перемещаем 64 бита из xmm0 в регистр rax, после чего уже записываем значение, откуда, с помощью инструкции fld помещаем его на стек математического сопроцессора.

Значения value и exp хранятся в операндах st(0) и st(1) соответственно. Инструкция fscale обеспечивает выполнение поставленной задачи, то есть, st(0) умножает на 2 в степени st(1). Результат помещает в st. После чего, с помощью инструкции fst помещаем результат выполнения на стек, откуда снимаем его в регистр rax, далее перемещаем в xmm0 и возвращаем результат выполнения функции.

В конце головная программа выводит значение выражения. Входные данные:

*value* = *−*3*.*72837

*exp* = 7

Таблица 1 – Результат прогона программы main в отладчике, начиная с момента вызова ассемблерного модуля.

|  |  |  |
| --- | --- | --- |
| Символический код команды | Содержимое регистров  и ячеек памяти | |
| До выполнения | После выполнения |
| push rdi | rip = 0x555555555226 | rip = 0x555555555227 |
|  | rsp = 0x7fffffffdc78 | rsp = 0x7fffffffdc70 |
|  | Stack: | Stack: |
|  | +0 0x00005555555551e5 | +0 0x0000000000000007 |
|  | +8 0x0000000700000000 | +8 0x00005555555551e5 |

|  |  |  |
| --- | --- | --- |
| fild dword ptr [rsp] | rip = 0x555555555227  st0 = 0 fstat = 0x0  ftag = 0xffff | rip = 0x55555555522a  st0 = 7  fstat = 0x3800 ftag = 0x3fff |
| movq rax, xmm0 | rip = 0x55555555522a  rax = 0xc00dd3b3a68b19a4 | rip = 0x55555555522f  rax = 0xc00dd3b3a68b19a4 |
| mov qword ptr [rsp], rax | rip = 0x55555555522f  Stack:  +0 0x0000000000000007 | rip = 0x555555555233  Stack:  +0 0xc00dd3b3a68b19a4 |
| fld qword ptr [rsp] | rip = 0x555555555233  st0 = 7  st1 = 0  fstat = 0x3800 ftag = 0x3fff | rip = 0x555555555236  st0 =  -3.72836999999999996191  st1 = 7  fstat = 0x3000 ftag = 0xfff |
| fscale | rip = 0x555555555236  st0 =  -3.72836999999999996191 | rip = 0x555555555238  st0 =  -477.231359999999995125 |
| fst qword ptr [rsp] | rip = 0x555555555238  Stack:  +0 0xc00dd3b3a68b19a4 | rip = 0x55555555523b  Stack:  +0 0xc07dd3b3a68b19a4 |
| pop rax | rip = 0x55555555523b  rax = 0xc00dd3b3a68b19a4 rsp = 0x7fffffffdc70  Stack:  +0 0xc07dd3b3a68b19a4  +8 0x00005555555551e5 | rip = 0x55555555523c  rax = 0xc07dd3b3a68b19a4 rsp = 0x7fffffffdc78  Stack:  +0 0x00005555555551e5  +8 0x0000000700000000 |
| movq xmm0, rax | rip = 0x55555555523c  xmm0 = 0xc00dd3b3a68b19a4, 0x0 | rip = 0x555555555241  xmm0 = 0xc07dd3b3a68b19a4, 0x0 |

|  |  |  |
| --- | --- | --- |
| ret | rip = 0x555555555241 | rip = 0x5555555551e5 |
|  | rsp = 0x7fffffffdc78 | rsp = 0x7fffffffdc80 |
|  | Stack: | Stack: |
|  | +0 0x00005555555551e5 | +0 0x0000000700000000 |
|  | +8 0x0000000700000000 | +8 0xc00dd3b3a68b19a4 |

# Выводы

Получены навыки работы со специальными инструкциями Ассемблера для чисел с плавающей запятой. Разработана программа на ЯВУ Си, которая с помощью модуля написанного на языке Ассемблера и использованием математического сопроцессора, вычисляет, а после и отображает значение выражения для заданных value и exp.

# ПРИЛОЖЕНИЕ A КОД ПРОГРАММ

Название файла: main.c

#include <stdio.h>

#include <stdlib.h>

double ldexp(double value, int exp);

int main() {

printf("Enter value for composition(<double> \* ...), value: ");

double val;

scanf("%lf", &val);

printf("Enter exp for pow(2, <int>), exp: ");

int exp;

scanf("%d", &exp);

double result = ldexp(val, exp);

printf("Result of expression(value \* 2^exp):\t%.9lf\n", result);

return 0;

}

Название файла: source.s

.global ldexp

ldexp:

push rdi

fild dword ptr [rsp]

movq rax, xmm0

movq [rsp], rax

fld qword ptr [rsp]

fscale

fst qword ptr [rsp]

pop rax

movq xmm0, rax

ret

Название файла: Makefile

all: main

main: main.o source.o

gcc main.o source.o -o main -z noexecstack -lm

main.o: main.c

gcc -c main.c

source.o: source.s

as source.s -msyntax=intel -mnaked-reg -mmnemonic=intel -o source.o

clean:

rm -f \*.o main