Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное   
образовательное учреждение высшего образования   
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №6

Дисциплина: «Системное программирование и архитектура ЭВМ»

Выполнила: Иванова

Анна Алексеевна

Группа: 6206-010302D

№ варианта:4

Самара, 2024

**Задание на лабораторную работу**

1) В программе необходимо реализовать функцию вычисления заданного

условного выражения на языке ассемблера с использованием команд

арифметического сопроцессора.

2) Результат для возврата из функции необходимо формировать в вершине

стека сопроцессора.

3) Необходимо провести анализ полученного результата и особых состояний

возникающих при вычислении.

4) Значения переменных передаются в качестве параметров функции.

5) В программе реализовать вывод результата на экран и вывод сообщения о

состоянии результата.

6) Все параметры функции имеют тип double.

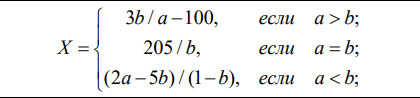
7) Проверку деления на 0 реализовать также на встроенном ассемблере.

8) В качестве комментария к каждой строке необходимо указать, какой

промежуточный результат, в каком регистре формируется.

9) В качестве комментария к строкам, содержащим команды сопроцессора

необходимо указать состояние регистров сопроцессора.

Вариант 4: 

**Текст программы**

#include<iostream>

#include<string>

using namespace std;

double calculate(double a, double b) {

double X;

if (a > b) X = 3 \* b / a - 100;

else if (a == b) X = 205 / b;

else X = (2 \* a - 5 \* b) / (1 - b);

return X;

}

int main() {

double a, b;

cout << "Input a: ";

cin >> a;

cout << "Input b: ";

cin >> b;

double res = 0;

int err = 0;

const double const\_no1 = -1, const\_1 = 1, const\_2 = 2, const\_3 = 3, const\_5 = 5, const\_20 = 20, const\_100 = 100, const\_205 = 205;

// X = 3\*b/a - 100, a > b

// X = 205 / b, a = b

// X = (2\*a-5\*b) / (1 - b), a < b

\_\_asm {

finit

fld qword ptr[b]; st0 = b

fld qword ptr[a]; st0 = a, st1 = b

fcom; сравниваем a и b

fnstsw ax

sahf

ja a\_bigger

jb b\_bigger

ftst; сравнение b с 0

fnstsw ax; сохраняем регистр флагов сопроцессора

sahf; записываем в регистр флагов процессора

je error; переход если b = 0

fld qword ptr[const\_205]; ST(0) = 205; ST(1) = a ST(2) = b

fdiv qword ptr[b]; ST(0) = 205 / c ST(1) = a ST(2) = b

jmp m3; выводим результат

a\_bigger :

; X = 3 \* b / a - 100, a > b

; ST(0) = a; ST(1) = b

ftst; сравнение a с 0

fnstsw ax; сохраняем регистр флагов сопроцессора

sahf; записываем в регистр флагов процессора

je error; переход если a = 0

fld qword ptr[const\_3]; ST(0) = 3, ST(1) = a, ST(2) = b

fmulp st(2), st; ST(0) = a, ST(1) = 3 \* b

fdivp st(1), st; ST(0) = 3 \* b / a,

fld qword ptr[const\_100]; ST(0) = 100, ST(1) = 3 \* b / a

fsubp st(1), st; ST(0) = 3 \* b / a - 100

jmp m3; выводим результат

b\_bigger :

; X = (2 \* a - 5 \* b) / (1 - b), a < b

; ST(0) = a; ST(1) = b

; вычисление числителя

fld qword ptr[const\_5]; ST(0) = 5, ST(1) = a, ST(2) = b

fmulp st(2), st; ST(0) = a, ST(1) = 5\*b

fld qword ptr[const\_2]; ST(0) = 2, ST(1) = a, ST(2) = 5\*b

fmulp st(1), st; ST(0) = 2 \* a, ST(1) = 5\*b

fsubp st(1), st; ST(0) = 5 \* b - 2 \* a

fld qword ptr[const\_no1]; ST(0) = -1, ST(1) = 5 \* b - 2 \* a

fmulp st(1), st; ST(0) = 2 \* a - 5 \* b

; вычисление знаменателя

fld qword ptr[const\_1]; ST(0) = 1, ST(1) = 2 \* a - 5 \* b

fld qword ptr[b]; ST(0) = b, ST(1) = 1, ST(2) = 2 \* a - 5 \* b

fsubp st(1), st; ST(0) = 1 - b, ST(1) = 2 \* a - 5 \* b

; проверка знаменателя и деление

ftst; сравниваем st(0) и 0

fstsw ax; записываем содержимое SWR в регистр АХ

sahf; запись флагов из ax в flags

je error; если знаменатель = 0

fdivp st(1), st; st(0) = (2 \* a - 5 \* b) / (1 - b)

jmp m3

error :

mov err, 1

jmp m3

m3 :

fstp res

}

if (err == 0) {

cout << "Asm result: " << res << endl;

cout << "C++ result: " << calculate(a, b);;

}

else if (err == 1) {

cout << "Division by zero error" << endl;

}

return 0;

}

