МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ТЕОРЕТИЧЕСКОЙ И ПРИКЛАДНОЙ ИНФОРМАТИКИ

Лабораторная работа № 4

по дисциплине «Низкоуровневое программирование»

на тему «ПРОГРАММИРОВАНИЕ СОПРОЦЕССОРА»

Факультет: ФПМИ

Группа: ПМИ-32

Студенты: Весёлый Д., Ворончук И., Лыкова М.

Бригада №5

Преподаватели: Сивак М.А.

НОВОСИБИРСК 2025

Цель работы: Изучить архитектуру и средства программирования сопроцессора (модуля операций с плавающей точкой) на языке ассемблера и приобрести практические навыки работы с ними.

Задание: Написать программу, реализующую вычисление функции в точке (вводится пользователем) в соответствии с заданным вариантом. Программа должна состоять из модулей на C++ и ассемблере, причем в модуле на C++ осуществляется ввод-вывод, а все вычисления – в модуле на ассемблере.

Вариант 5: $f(x) = (x + 2*x^2)/(x - 1)$

Алгоритм:

1. Анализ задачи: Цель работы - разработать программу, вычисляющую значение функции $f(x) = (x + 2*x^2)/(x - 1)$ в заданной пользователем точке x. Вычисления должны выполняться на языке ассемблера с использованием команд сопроцессора, а ввод и вывод данных - в модуле на C++. Программа должна корректно обрабатывать ошибочные ситуации (например, деление на ноль при x = 1) и выводить результат в удобном виде.
2. В C++ реализуется:
    - Ввод и проверка данных пользователя (если введено не число - выводится сообщение об ошибке, если x близко к 1, программа сообщает о невозможности деления на ноль.).
    - После получения результата программа формирует строку “Результат: (x + 2*x^2) / (x - 1) = <значение>”, после чего выводит её пользователю.
    - Используются функции Windows API (SetConsoleCP, SetConsoleOutputCP, CharToOemA) для корректного отображения русских символов в кодировке OEM 866.

   Ассемблер:
    - Получение аргумента функции x из стека (соглашение вызова cdecl).
    - Вычисление числителя.
    - Вычисление знаменателя.
    - Выполнение деления
    - Возврат результата в виде значения double в регистре st(0).

Все операции выполняются средствами сопроцессора: fld, fmul, fadd, fsub, fdivr, fstp.

3. Алгоритм ассемблерной процедуры compute_func:
    - Сохраняется базовый регистр стека ebp.
    - Загружается аргумент x.
    - Вычисляется числитель: x копируется в st(0) с помощью fld, возвышение в квадрат происходит через fmul st(0), в st(0) после этого хранится x^2. $2x^2$ получается с помощью fadd st(0) ($x^2 + x^2$), в st(0) хранится $2x^2$. С помощью fadd st(0), st(1) прибавляем x к $2x^2$, в st(0) хранится $x + 2x^2$.
    - Вычисляем знаменатель: снова загружаем x (fld qword ptr [ebp + 8]). Загружаем константу 1.0 с помощью fld1. Вычитаем 1 с помощью fsub st(1), st(0), получаем в st(1) x - 1. Удаляем 1.0 из стека (fstp st(0)). Получаем теперь st(0) = x - 1, st(1) = числитель.
    - Выполняем деление: fdivr st(0), st(1) ; st(0) = (числитель) / (знаменатель).
    - В st(0) остаётся результат функции.
    - В конце восстанавливаем стек и завершаем.

Результат автоматически возвращается в вызывающий код C++ в регистре st(0).

4. Алгоритм вызывающей функции на C++:
    - Устанавливаем кодировку консоли (OEM 866).
    - Выводим приглашение на ввод: "Введите значение x:".
    - Считаем введённое пользователем значение x.
    - Проверяем корректность: если ввод некорректен, то выводим сообщение об ошибке. Если же $x \approx 1$, то выводим сообщение о делении на ноль.
    - Вызываем ассемблерную функцию: double result = compute_func(x);
    - Получаем результат вычислений из ассемблерной процедуры.
    - Формируем строку с результатом и выводим её.
    - Завершаем программу.

Тесты:

1. **Простое положительное число**

Входное значение: 2

Ожидаемый результат выражения: $(2 + 2*2^2) / (2 - 1) = (2 + 8) / 1 = 10.00$

Результат программы:

```
Введите значение x: 2
Результат: (2 + 2*2^2) / (2 - 1) = 10
```

2. **Ноль в числителе**

Входное значение: 0

Ожидаемый результат выражения: $(0 + 0) / (0 - 1) = 0 / (-1) = 0$

Результат программы:

```
Введите значение x: 0
Результат: (0 + 2*0^2) / (0 - 1) = 0
```

3. **Отрицательное число**

Входное значение: −1

Ожидаемый результат выражения: $(-1 + 2*(-1)^2) / (-1 - 1) = (-1 + 2) / (-2) = 1 / (- 2) = - 0.5$

Результат программы:

```
Введите значение x: -1
Результат: (-1 + 2*-1^2) / (-1 - 1) = -0.5
```

4. **Проверка работы с дробными числами**

Входное значение: 1.5

Ожидаемый результат: $(1.5 + 2*2.25) / (1.5 - 1) = (1.5 + 4.5) / 0.5 = 6 / 0.5 = 12.00$

Результат программы:

```
Введите значение x: 1.5
Результат: (1.5 + 2*1.5^2) / (1.5 - 1) = 12
```

5. **Проверка защиты от деления на ноль:**

Входное значение: 1

Ожидаемый результат:

Результат программы:

```
Введите значение x: 1
Ошибка: деление на ноль (x не может быть равно 1).
```

6. **Проверка на буквы:**

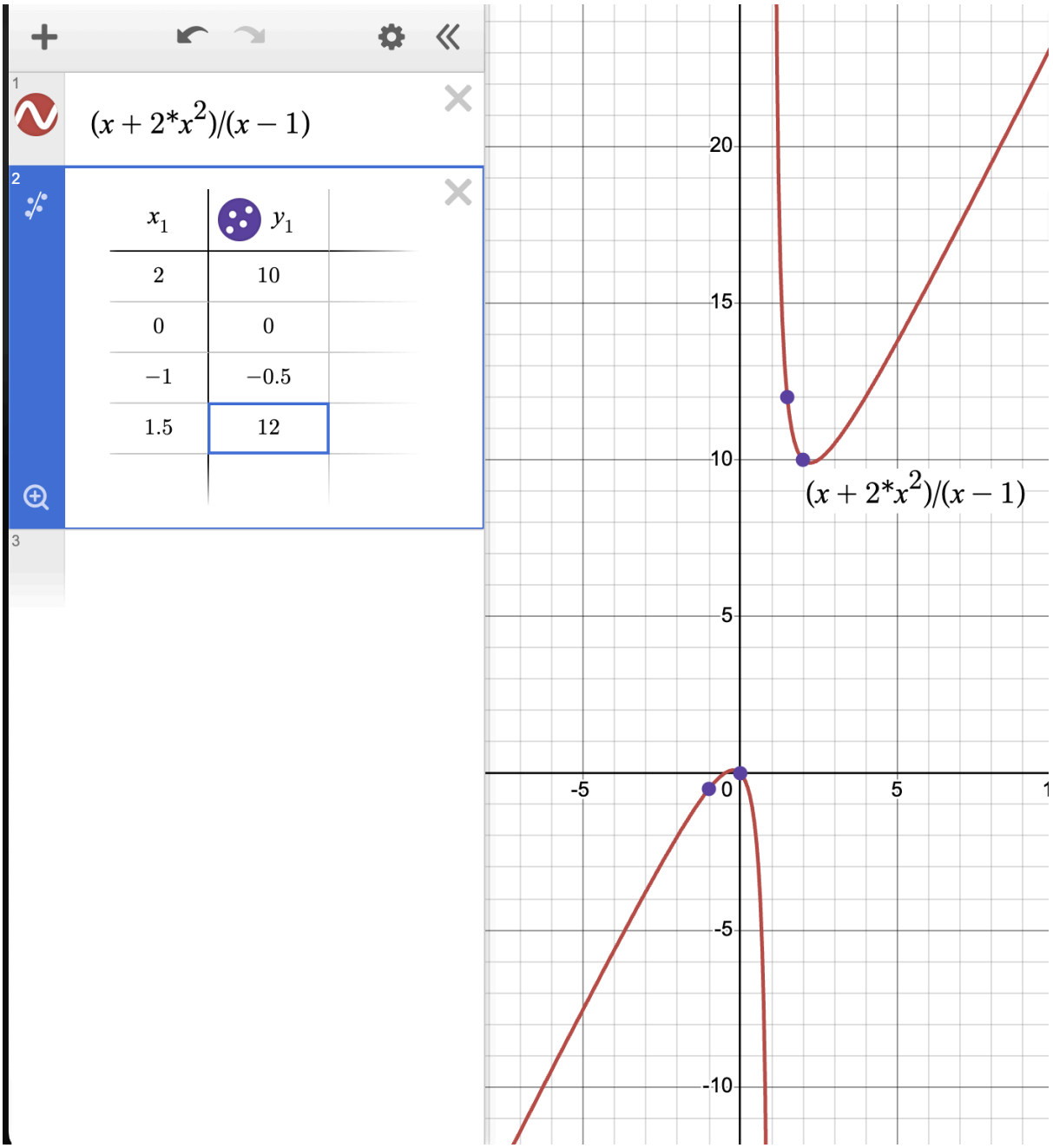
Входное значение: qwerty

Ожидаемый результат: сообщение об ошибке.

Результат программы:

```
Введите значение x: qwerty
Ошибка: введены некорректные данные (требуется число).
```

График функции:



main.cpp

```
#include <iostream>
#include <windows.h>
#include <cmath>
#include <limits>

extern "C" double compute_func(double x);

void PrintOemString(const char* ansiStr) {
    char oemStr[512];
    CharToOemA(ansiStr, oemStr);
    std::cout << oemStr;
}

int main() {
    // Настройка OEM-консоли
    SetConsoleCP(866);
    SetConsoleOutputCP(866);

    double x;
    PrintOemString("Введите значение x: ");

    // Считываем число
    if (!(std::cin >> x)) {
        PrintOemString("Ошибка: введены некорректные данные (требуется
число).\n");
        return 1;
    }

    // Проверка на деление на ноль
    if (std::abs(x - 1.0) < 1e-12) {
        PrintOemString("Ошибка: деление на ноль (x не может быть равно
1).\n");
        return 1;
    }
    double result = compute_func(x);
    // Формируем строку результата
    char buffer[256];
    int len = std::snprintf(buffer, sizeof(buffer),
        "Результат: (%g + 2*%g^2) / (%g - 1) = %g\n", x, x, x, result); // %g -
автоматически выбирает между десятичным и
//экспоненциальным представлением + лишние нули не выводит
    if (len > 0 && len < static_cast<int>(sizeof(buffer))) {
        PrintOemString(buffer);
    }
    else {
        PrintOemString("Ошибка: не удалось сформировать строку результата.\n");
    }

    return 0;
```

```
}
```

math.asm

```
; Вычисляем (x + 2*x^2) / (x - 1)
; cdecl, возвращает double в st(0)

.386
.model flat, C

.code

compute_func PROC
    push    ebp
    mov     ebp, esp    ; Получаем указатель на аргумент через ebx
; ====================================
;                       Стек
;       [ebp]       -> сохранённый ebp
;       [ebp+4]     -> возвратный адрес
;       [ebp+8]     -> аргумент x (double)
; ====================================

    fld     qword ptr [ebp + 8]     ; st(0) = x, fld берёт 8 байт по адресу [ebp+8]

; ====================================
; Числитель: x + 2*x2
    fld     st(0)                   ; st(0) = x, st(1) = x
    fmul    st(0), st(0)            ; st(0) = x2, st(1) = x
    fadd    st(0), st(0)            ; st(0) = 2*x2, st(1) = x (x2+x2)
    fadd    st(0), st(1)            ; st(0) = 2*x2 + x

; st(0) содержит числитель 2*x2 + x
; ====================================

; ====================================
; Знаменатель: x - 1
    fld     qword ptr [ebp + 8]     ; st(0) = x,   st(1) = числитель
    fld1                            ; st(0) = 1.0, st(1) = x,        st(2) = числитель
    fsub    st(1), st(0)            ; st(0) = 1.0, st(1) = x - 1,    st(2) = числитель
    fstp st                         ; st(0) = x-1, st(1) = числитель
; fstp - загрузить вершину в вершину и удалить её. элементы сдвинуться на i-1
; ====================================

; ====================================
; Деление: числитель / знаменатель
    fdivr   st(0), st(1)            ; st(0) = st(1) / st(0)
; r - реверсивный
; ====================================

; Очищаем стек (остался только результат в st(0))
    pop ebp       ; Восстанавливаем старый EBP
    ret
compute_func ENDP

END
```