

СОДЕРЖАНИЕ

Лабораторная работа 1 «Арифметические и логические команды в ассемблере».....	4
1.1 Теоретические основы лабораторной работы	4
1.2 Задание.....	5
1.3 Схема алгоритма.....	5
1.4 Решение	6
1.5 Ответ	6
Лабораторная работа 2 «Арифметические команды и команды переходов в ассемблере».....	8
2.1 Теоретические основы лабораторной работы	8
2.2 Задание.....	9
2.3 Схема алгоритма.....	9
2.4 Решение	10
2.5 Ответ	12
Лабораторная работа 3 «Команды работы с массивами и стеком»	15
3.1 Теоретические основы лабораторной работы	15
3.2 Задание.....	16
3.3 Схема алгоритма.....	16
3.4 Решение	18
3.5 Ответ	18
Лабораторная работа 4 «Работа с математическим сопроцессором в среде ассемблер».....	21
4.1 Теоретические основы лабораторной работы	21
4.2 Задание.....	22
4.3 Схема алгоритма.....	23
4.4 Решение	24
4.5 Ответ	24
Лабораторная работа 5 «Нахождения функций с помощью математического сопроцессора».....	28
5.1 Теоретические основы лабораторной работы	28

5.2	Задание.....	28
5.3	Схема алгоритма.....	29
5.4	Решение	30
5.5	Ответ	31
Лабораторная работа 6 «Нахождение суммы ряда с помощью математического сопроцессора»		31
6.1	Теоретические основы лабораторной работы	32
6.2	Задание.....	33
6.3	Схема алгоритма.....	33
6.4	Решение	34
6.5	Ответ	36
Список использованных источников		Ошибка! Закладка не определена.

ЛАБОРАТОРНАЯ РАБОТА 1

«Арифметические и логические команды в ассемблере»

1.1 Теоретические основы лабораторной работы

Команды, использованные в программе, приведены в таблице 1.1.

Таблица 1.1 – Описание команд

Команда	Назначение
ADD	Выполняет сложение байтов или слов, содержащих двоичные данные.
IMUL	Выполняет операцию умножения для знаковых данных.
IDIV	Выполняет операцию деления для знаковых данных.
NEG	Команда вычисляет двоичное дополнение операнда.
DEC	Уменьшает значение операнда в памяти или регистре на 1.
CDQ	Расширяет двойное слово со знаком до размера учетверенного слова (64 бита) со знаком. Используется для подготовки к операции деления.
MOV	Команда пересылки данных (из источника в приемник).
PUSH	Перемещает данные в стек.
POP	Считывает данные из стека.
SHR	Арифметический сдвиг вправо.
SHL	Арифметический сдвиг влево.

Использованные в программе регистры приведены в таблице 1.2.

Таблица 1.2 – Описание регистров

Регистр	Назначение
EAX	Регистр общего назначения. Аккумулятор.
EBX	Регистр общего назначения. База.
ECX	Регистр общего назначения. Счетчик.
EDX	Регистр общего назначения. Регистр данных.

Использованные библиотеки приведены в таблице 1.3.

Таблица 1.3 – Описание библиотек

Библиотека	Назначение
"stdafx.h"	Служит для генерации файла предкомпилированных заголовков.
<stdio.h>	Используется для организации стандартного ввода/вывода.
<iostream>	Используется для организации потокового ввода/вывода.

1.2 Задание

1) В программе необходимо реализовать функцию вычисления целочисленного выражения $(c - d/2 + 33)/(2 * a * a - 1)$ на встроенном ассемблере MASM в среде Microsoft Visual Studio на языке C++.

2) Значения переменных передаются в качестве параметров функции.

3) Результат выводить в консольном приложении (проект консольное приложение Win32).

4) В программе реализовать ввод переменных из командной строки и вывод результата на экран.

5) Все параметры функции 32 битные числа (знаковые и беззнаковые).

6) Первые строки функции вычисления выражения заносят значения аргументов функции в соответствующие регистры.

7) Где необходимо реализовать проверки вводимых данных и вычисления отдельных операций. Например, проверка деления на 0.

8) В качестве комментария к каждой строке необходимо указать, какой промежуточный результат, в каком регистре формируется.

9) По возможности использовать команды сдвига.

1.3 Схема алгоритма

Для того чтобы вычислить результат данного выражения, необходимо организовать ввод с клавиатуры необходимых параметров (a, b, c). После ввода значений первым считается знаменатель $(2 * a * a - 1)$, после чего он помещается в стек. Затем считается числитель $(c - d/2 + 33)$. После этого

извлекаем посчитанный знаменатель из стека и делим числитель на него. Полученный результат записываем в переменную result, а затем возвращаем result как значение функции.

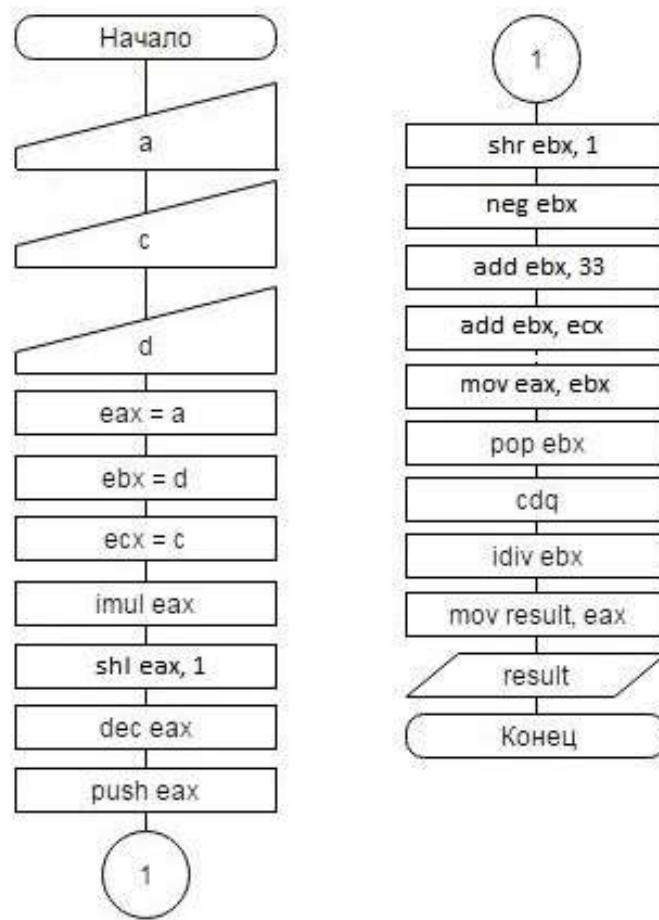


Рисунок 1.1 – Схема алгоритма вычисления исходного выражения

1.4 Решение

```

#include "stdafx.h"
#include <stdio.h> // стандартный ввод/вывод
#include <iostream> // потоковый ввод/вывод
using namespace std;

// функция вычисления выражения (c - d/2 + 33)/(2 * a * a - 1)
int Calc(int a, int d, int c)
{
    int result = 0;

    __asm {
        mov     eax, a;
        mov     ebx, d;
        mov     ecx, c;
        imul    eax;
        shl     eax, 1;
        dec     eax;
        push    eax;
        shr     ebx, 1;
        neg     ebx;
        add     ebx, 33;
        add     ebx, ecx;

        <edx:eax> = a * a
        2 * a * a
        <eax> = 2 * a * a - 1
        в стеке 2 * a * a - 1
        d/2
        -(d/2)
        + 33
    }
}

```

```

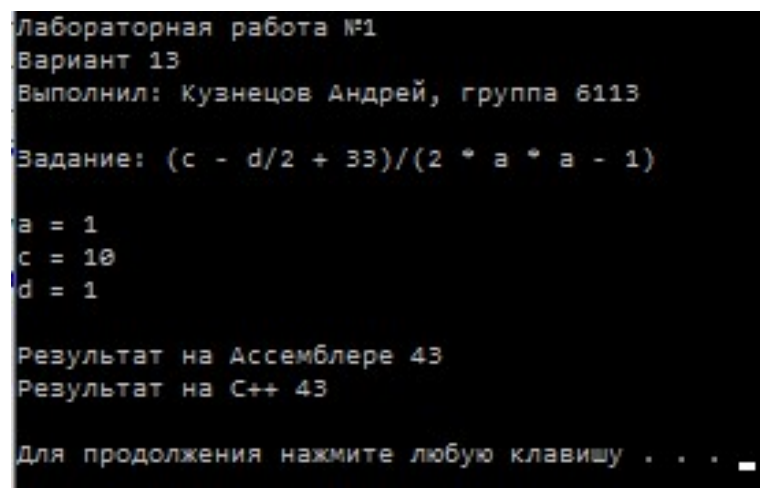
        mov eax, ebx;           (c - d / 2 + 33)
        pop ebx;               (2 * a * a - 1)
        cdq;                   <eax> => <edx:eax>
        idiv ebx;
        mov result, eax;      result = eax
    }
    return result; // возвращаем результат вычисления выражения
}

int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_ALL, "RUS");
    int a, d, c;
    printf("Лабораторная работа №1\n");
    printf("Вариант 13\n");
    printf("Выполнил: Кузнецов Андрей, группа 6113\n\n");
    printf("Задание: (c - d/2 + 33)/(2 * a * a - 1)\n\n");
    printf("a = ");
    cin >> a; // потоковый ввод
    printf("c = ");
    cin >> c;
    printf("d = ");
    cin >> d;
    int resAsm = Calc(a, d, c); // вычисление выражения
    printf("\nРезультат на Ассемблере %d\n", resAsm); // вывод результата вычисления
    // выражения
    int resC = (c - d / 2 + 33) / (2 * a * a - 1);
    printf("Результат на C++ %d\n\n", resC);
    system("PAUSE");
    return 0;
}

```

1.5 Ответ

На рисунке 1.1 приведено выполнение программы при вводе верных исходных параметров.



```

Лабораторная работа №1
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание: (c - d/2 + 33)/(2 * a * a - 1)

a = 1
c = 10
d = 1

Результат на Ассемблере 43
Результат на C++ 43

Для продолжения нажмите любую клавишу . . .

```

Рисунок 1.2 – Работа программы в случае верного ввода исходных данных

ЛАБОРАТОРНАЯ РАБОТА 2

«Арифметические команды и команды переходов в ассемблере»

2.1 Теоретические основы лабораторной работы

Команды, использованные в программе, приведены в таблице 2.1.

Таблица 2.1 – Описание команд

Команда	Назначение
ADD	Сложение байтов или слов, содержащих двоичные данные.
SUB	Вычитание байтов или слов, содержащих двоичные данные.
IDIV	Выполняет операцию деления для знаковых данных.
CDQ	Расширяет двойное слово со знаком до размера учетверенного слова (64 бита) со знаком. Используется для подготовки к операции деления, для которой размер делимого должен быть в два раза больше размера делителя.
OR	Операция логического ИЛИ над битами операнда назначения.
CMP	Сравнивает два операнда.
JE	Инструкция условного перехода «перейти, если равно».
JG	Инструкция условного перехода «перейти, если больше чем...»
JL	Инструкция условного перехода «перейти, если меньше чем...»
JMP	Используется в программе для организации безусловного перехода.
MOV	Команда пересылки данных (из источника в приемник).

Использованные в программе регистры приведены в таблице 2.2.

Таблица 2.2 – Описание регистров

Регистр	Назначение
EAX	Регистр общего назначения. Аккумулятор.
EBX	Регистр общего назначения. База.
ECX	Регистр общего назначения. Счетчик.
EDX	Регистр общего назначения. Регистр данных.

Использованные библиотеки приведены в таблице 2.3.

Таблица 2.3 – Описание библиотек

Библиотека	Назначение
"stdafx.h"	Служит для генерации файла предкомпилированных заголовков; в него включено большинство стандартных и используемых в каждом приложении включаемых файлов. Сделано это для того, чтобы ускорить компиляцию проекта.
<stdio.h>	Используется для организации стандартного ввода/вывода.
<iostream>	Используется для организации потокового ввода/вывода.

2.2 Задание

1) В программе необходимо реализовать функцию вычисления заданного условного целочисленного выражения, используя команды сравнения, условного и безусловного переходов на встроенном ассемблере.

$$X = \begin{cases} b / a + 10, & \text{если } a > b; \\ -20, & \text{если } a = b; \\ (b - a) / b, & \text{если } a < b; \end{cases}$$

2) Результат X – целочисленный, возвращается из функции регистра еах.

3) Значения переменных передаются в качестве параметров функции.

4) В программе реализовать вывод результата на экран.

5) Все параметры функции 32 битные числа.

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

7) В качестве комментария к каждой строке необходимо указать, какой промежуточный результат, в каком регистре формируется.

8) По возможности использовать команды сдвига.

2.3 Схема алгоритма

На рисунке 2.1 приведена схема алгоритма вычисления функции

$$X = \begin{cases} b / a + 10, & \text{если } a > b; \\ -20, & \text{если } a = b; \\ (b - a) / b, & \text{если } a < b; \end{cases}$$

Для того чтобы вычислить результат необходимо организовать ввод с клавиатуры исходных данных (параметров a, b). После ввода сравниваются значения введенных параметров. Если $a > b$, то результат есть значение функции $(b/a + 10)$, если $a < b$, то результат есть значение функции $((b - a)/b)$, иначе результат равен -20.

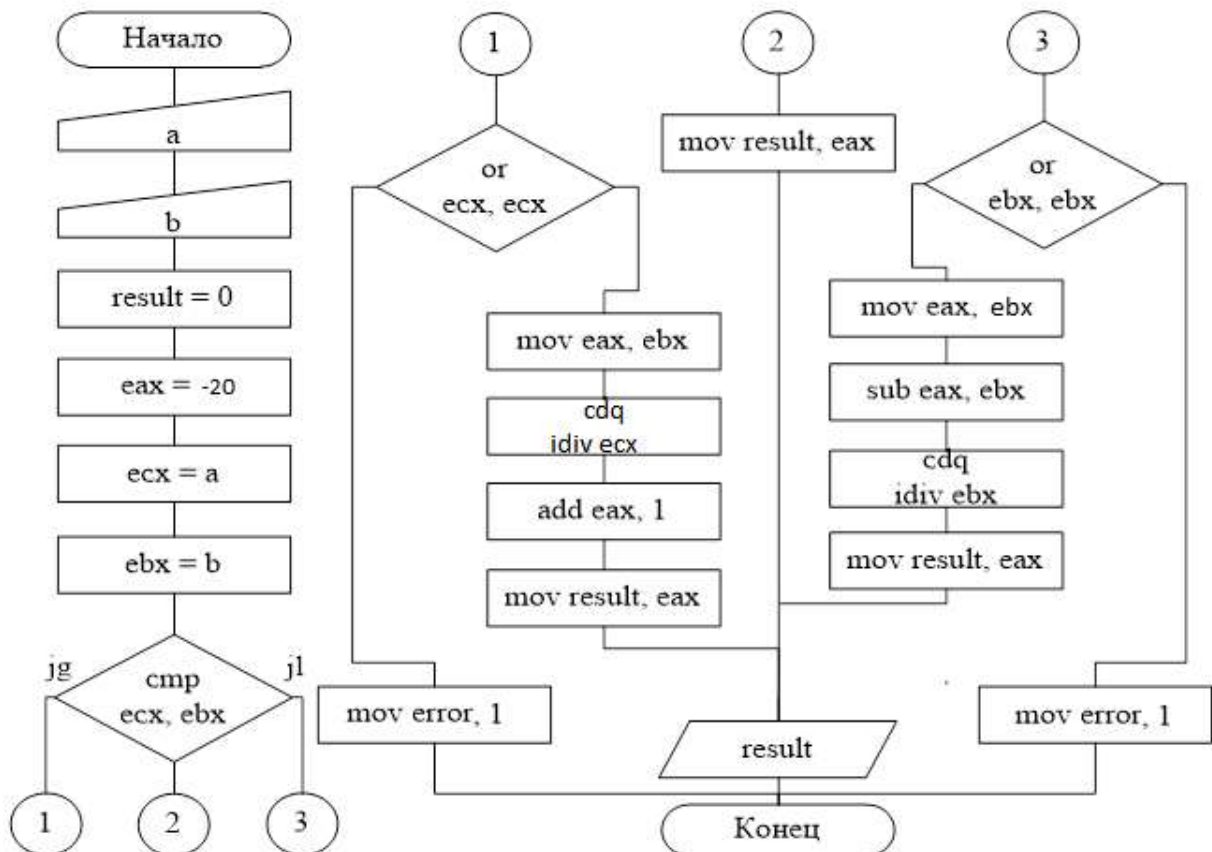


Рисунок 2.1 – Схема алгоритма вычисления исходного выражения

2.4 Решение

```

#include "stdafx.h"
#include <stdio.h> // стандартный ввод/вывод
#include <iostream> // потоковый ввод/вывод

using namespace std;

// функция вычисления выражения
// * (b/a) + 10, если a > b
// * -20, если a = b
// * (b - a)/b, если a < b

bool error = 0;
int CalcAsm(int a, int b)
{
    int result = 0;
    __asm {

```

```

        mov     eax, -20;           <eax> = -20;
        mov     ecx, a;             <ecx> = a
        mov     ebx, b;             <ebx> = b
        cmp     ecx, ebx;           сравнение a и b
        jg      l_bigger;           переход, если a > b
        jl      l_smaller;         переход если a < b
        mov     result, eax;        result = eax
        jmp     end_1;             переход на конец программы
l_bigger:
        or      ecx, ecx;           сравнение a и 0
        je      error1;            ошибка деление на ноль
        mov     eax, ebx;           <eax> = b
        cdq;                         подготовка деления <edx : eax> = a
        idiv    ecx;                <eax> = b / a
        add     eax, 10;            <eax> = b / a + 10
        mov     result, eax;        result = eax
        jmp     end_1;             переход на конец программы
l_smaller:
        or      ebx, ebx;           сравнение b и 0
        je      error1;            ошибка деление на ноль
        mov     eax, ebx;           <eax> = b
        sub     eax, ecx;           <eax> = b - a
        cdq;                         подготовка деления <edx:eax>
        idiv    ebx;                <eax> = (b - a) / b
        mov     result, eax;        result = eax
        jmp     end_1;             переход на конец программы
error1:
        mov     error, 1;
end_1:
}
return result; // возвращение результата
}

int CalcC(int a, int b) // [a > b] (b/a) + 10; [a = b] -20; [a < b] (b - a)/b
{
    if (a > b) {
        return ((b / a) + 10);
    }
    else {
        if (a < b) {
            return ((b - a) / b);
        }
        else {
            return (-20);
        }
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_CTYPE, "rus");
    int a, b;

    printf("Лабораторная работа № 2\n");
    printf("Вариант 13\n");
    printf("Выполнил: Кузнецов Андрей, группа 6113\n\n");
    printf("Задание:\n");
    printf("Вычисление выражения с условием:\n* (b/a) + 10, если a > b\n* -20, если a
= b\n* (b - a)/b, если a < b\n\n");

    printf("Введите a = "); // стандартный вывод
    std::cin >> a; // потоковый ввод
    printf("Введите b = ");
    std::cin >> b;

    int res = CalcAsm(a, b); // вычисление выражения

```

```

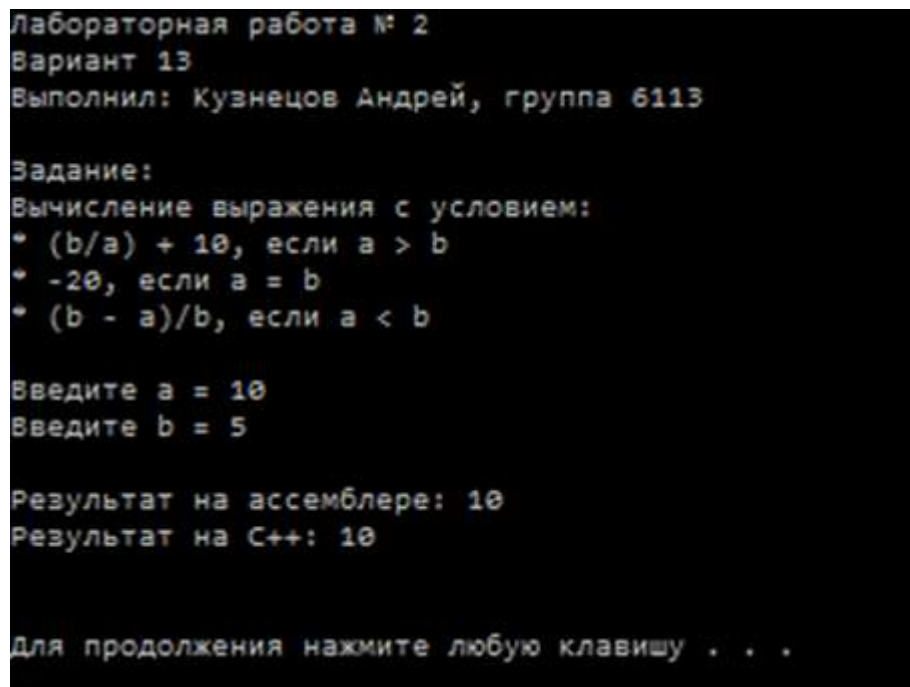
if (error == 1) {
    std::cout << "Ошибка" << std::endl;
}
else {
    printf("\nРезультат на ассемблере: ");
    printf("%d", res); // вывод результата вычисления выражения
    printf("\nРезультат на C++: ");
    printf("%d\n", CalcC(a, b));
}

printf("\n\n"); // стандартный вывод
system("PAUSE");
return 0;
}

```

2.5 Ответ

На рисунках 2.2, 2.3, 2.4 приведены тестовые примеры выполнения программы при вводе корректных исходных данных. На рисунке 2.5 приведен результат работы программы в случае ввода недопустимых значений, когда делитель равен 0. Программа выполняет проверку данного значения и выдает предупредительное сообщение.



```

лабораторная работа № 2
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
Вычисление выражения с условием:
* (b/a) + 10, если a > b
* -20, если a = b
* (b - a)/b, если a < b

Введите a = 10
Введите b = 5

Результат на ассемблере: 10
Результат на C++: 10

для продолжения нажмите любую клавишу . . .

```

Рисунок 2.2 – Работа программы в случае $a > b$

```
Лабораторная работа № 2
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
Вычисление выражения с условием:
*  $(b/a) + 10$ , если  $a > b$ 
*  $-20$ , если  $a = b$ 
*  $(b - a)/b$ , если  $a < b$ 

Введите a = 10
Введите b = 10

Результат на ассемблере: -20
Результат на C++: -20

Для продолжения нажмите любую клавишу . . .
```

Рисунок 2.3 – Работа программы в случае $a = b$

```
Лабораторная работа № 2
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
Вычисление выражения с условием:
*  $(b/a) + 10$ , если  $a > b$ 
*  $-20$ , если  $a = b$ 
*  $(b - a)/b$ , если  $a < b$ 

Введите a = 0
Введите b = 20

Результат на ассемблере: 1
Результат на C++: 1

Для продолжения нажмите любую клавишу . . .
```

Рисунок 2.4 – Работа программы в случае $a < b$

```
Лабораторная работа № 2
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
Вычисление выражения с условием:
*  $(b/a) + 10$ , если  $a > b$ 
*  $-20$ , если  $a = b$ 
*  $(b - a)/b$ , если  $a < b$ 

Введите a = 0
Введите b = 20

Результат на ассемблере: 1
Результат на C++: 1

Для продолжения нажмите любую клавишу . . .
```

Рисунок 2.5 – Работа программы в случае некорректного ввода данных

ЛАБОРАТОРНАЯ РАБОТА 3

«Команды работы с массивами и стеком»

3.1 Теоретические основы лабораторной работы

Команды, использованные в программе, приведены в таблице 3.1.

Таблица 3.1 – Описание команд

Команда	Назначение
INC	Увеличивает значение операнда в памяти или регистре на 1.
CMP	Сравнивает два операнда.
JLE	Инструкция условного перехода «перейти, если меньше или равно».
JCXZ	Инструкция перехода внутри текущего сегмента команд в зависимости от некоторого условия.
JG	Инструкция условного перехода «перейти, если больше и не равно».
JL	Инструкция условного перехода «перейти, если меньше и не равно».
LOOP	Используется для организации цикла со счетчиком в регистре ECX.
MOV	Команда пересылки данных (из источника в приемник).
XOR	Операция логического исключающего ИЛИ над двумя операндами размерностью байт, слово или двойное слово (побитовое сравнение).

Использованные в программе регистры приведены в таблице 3.2.

Таблица 3.2 – Описание регистров

Регистр	Назначение
EAX	Регистр общего назначения. Аккумулятор.
EBX	Регистр общего назначения. База.

Продолжение таблицы 3.2

ECX	Регистр общего назначения. Счетчик.
EDX	Регистр общего назначения. Регистр данных.
ESI	Регистр общего назначения. Индекс источника.
EDI	Регистр общего назначения. Индекс приёмника.

Использованные библиотеки приведены в таблице 3.3.

Таблица 3.3 – Описание библиотек

Библиотека	Назначение
"stdafx.h"	Служит для генерации файла предкомпилированных заголовков.
<stdio.h>	Используется для организации стандартного ввода/вывода.
<iostream>	Используется для организации потокового ввода/вывода.

3.2 Задание

1) В программе необходимо реализовать функцию обработки элементов массива используя команды сравнения, переходов и циклов на встроенном ассемблере.

2) Результат целочисленный, возвращается из функции в регистре еах.

3) Массив передаётся в качестве параметра функции.

4) В программе реализовать вывод результата на экран.

5) В качестве комментария к каждой строке необходимо указать, какое действие выполняет команда относительно массива.

В одномерном массиве $A=\{a[i]\}$ целых чисел вычислить количество положительных элементов массива, которые удовлетворяют условию:

$$b \leq a[i] \leq d.$$

3.3 Схема алгоритма

На рисунке 3.1 приведена схема алгоритма поиска положительных элементов массива А. Для этого организовывается ввод с клавиатуры длины

массива (size). Затем поэлементно заполняется массив (mas[]). Каждый i-тый элемент сравнивается с 0. Если элемент равен 0, то значение счетчика увеличивается на единицу, иначе цикл переходит к следующему элементу (i + 1). В качестве результата на экран выводится значение счетчика.

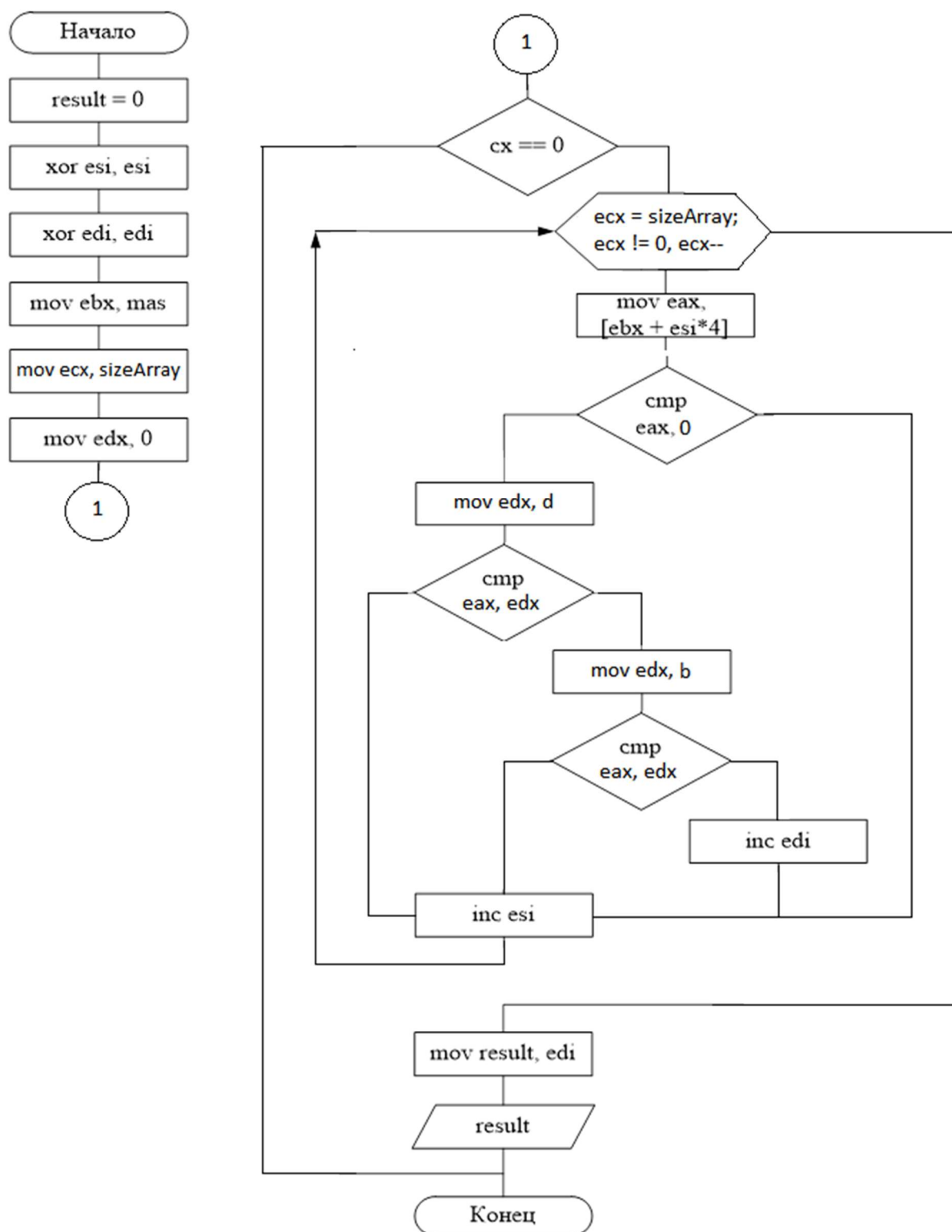


Рисунок 3.1 – Схема алгоритма подсчета нулевых элементов массива

3.4 Решение

```
#include "stdafx.h"
#include <stdio.h> // стандартный ввод/вывод
#include <iostream> // потоковый ввод/вывод
#include <locale.h>
using namespace std;

// В одномерном массиве A = {a[i]} целых чисел вычислить количество положительных
элементов массива, которые удовлетворяют условию: b <= a[i] <= d.

int AsmArray(int mas[], int sizeArray, int b, int d)
{
    int result = 0;

    __asm {
        xor     esi, esi;                подготовим регистр индекса в
    массиве
        xor     edi, edi;                счётчик количества элементов
        mov     ebx, mas;                ebx указывает на начало массива
        mov     ecx, sizeArray;          счётчик цикла по всем элементам
    массива
        jcxz    exit_1;                  завершить если длина массива 0
    begin_loop:
        mov     eax, [ebx + esi * 4];    определяем текущий элемент
        cmp     eax, 0;                  сравниваем текущий элемент с
    нулём
        jle     end_loop;                определяем положительный ли (больше нуля)
        mov     edx, d;                  подготовка сравнения с d
        cmp     eax, edx;                сравнение a[i] и d
        jg      end_loop;                если больше, то завершаем цикл
        mov     edx, b;                  подготовка сравнения с b
        cmp     eax, edx;                сравнение a[i] и b
        jl      end_loop;                если меньше, то завершаем цикл
        inc     edi;                     элемент удовлетворяет условию,
    увеличиваем счётчик
    end_loop:
        inc     esi;                     переходим к следующему элементу
        loop    begin_loop;              повторяем цикл для всех элементов массива
    exit_1:
        mov     result, edi;             возвращаем количество элементов
    }
    return result; // возвращаем результат вычисления выражения
}

int CArray(int mas[], int sizeArray, int b, int d)
{
    int k = 0;
    for (int i = 0; i < sizeArray; i++)
    {
        if (mas[i] > 0)
            if (mas[i] >= b && mas[i] <= d) k++;
    }
    return (k);
}

int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_CTYPE, "rus");

    printf("Лабораторная работа № 3\n");
    printf("Вариант 13\n");
    printf("Выполнил: Кузнецов Андрей, группа 6113\n\n");
    printf("Задание:\n");
}
```

```

printf("В одномерном массиве A={a[i]} целых чисел вычислить количество
положительных элементов массива, \пкоторые удовлетворяют условию: b <= a[i] <= d.\n\n");

int sizeArray;
do
{
    cout << "Введите размер массива не больше 10: " << endl; // потоковый ввод
    cin >> sizeArray;
} while (sizeArray > 10 || sizeArray < 0);

cout << "\nВведите массив: " << endl;
int mas[10];
for (int i = 0; i < sizeArray; i++)
{
    cin >> mas[i];
}

int b, d;
cout << "\nВведите условия для задачи: " << endl;
do
{
    cout << "b = ";
    cin >> b;
    cout << "d = ";
    cin >> d;
    if (d < b)
    {
        cout << "\nОшибка! d не может быть меньше b." << endl;
        cout << "Введите значение снова: " << endl;
    }
} while (d < b);

int res = AsmArray(mas, sizeArray, b, d); // вычисление выражения

printf("\nКоличество элементов больше или равно b, но меньше или равно d: ");//
стандартный вывод
printf("\nАссемблер: ");
printf("%d", res); // вывод результата вычисления выражения
printf("\nC++: ");
printf("%d", CArray(mas, sizeArray, b, d));
printf("\n"); // стандартный вывод
system("PAUSE");
return 0;
}

```

3.5 Ответ

На рисунке 3.2 приведен тестовый пример выполнения программы при вводе корректных данных. На рисунке 3.3 приведен результат работы программы в случае ввода недопустимых значений (например, ввод отрицательной длины массива). Программа выполняет проверку на корректность и выдает предупредительное сообщение.

```

Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
В одномерном массиве  $A=\{a[i]\}$  целых чисел вычислить количество положительных
элементов массива,
которые удовлетворяют условию:  $b \leq a[i] \leq d$ .

Введите размер массива не больше 10:
4

Введите массив:
3
-5
0
1

Введите условия для задачи:
b = 1
d = 10

Количество элементов больше или равно b, но меньше или равно d:
Ассемблер: 2
C++: 2
Для продолжения нажмите любую клавишу . . .

```

Рисунок 3.2 – Работа программы в случае корректного ввода длины массива

```

Задание:
В одномерном массиве  $A=\{a[i]\}$  целых чисел вычислить количество положительных
элементов массива,
которые удовлетворяют условию:  $b \leq a[i] \leq d$ .

Введите размер массива не больше 10:
11
Введите размер массива не больше 10:
-5
Введите размер массива не больше 10:
3

Введите массив:
-11
4
3

Введите условия для задачи:
b = 0
d = 10

Количество элементов больше или равно b, но меньше или равно d:
Ассемблер: 2
C++: 2
Для продолжения нажмите любую клавишу . . .

```

Рисунок 3.3 – Работа программы при попытке ввода некорректной длины массива

ЛАБОРАТОРНАЯ РАБОТА 4

«Работа с математическим сопроцессором в среде ассемблер»

4.1 Теоретические основы лабораторной работы

Команды, использованные в программе, приведены в таблице 4.1.

Таблица 4.1 – Описание команд

Команда	Назначение
FINIT	Инструкция инициализации сопроцессора.
FSTSW	Сохраняет текущее значение регистра SR в приемник.
SAHF	Загружает флаги из регистра AH.
FTST	Сравнивает текущее значение с нулем.
FLD	Загружает из памяти в вершину стека вещественное число.
FILD	Загружает из памяти в вершину стека ST(0) целое число.
FSTP	Извлекает из вершины стека ST(0) в память вещественное число.
FADDP	Сложение с «выбросом» результата в стек.
FSUBP	Вычитание с «выбросом» результата в стек.
FDIVP	Деление с «выбросом» результата в стек.
JA	Инструкция условного перехода «перейти, если больше чем...»
JB	Инструкция условного перехода «перейти, если меньше чем...»
JE	Инструкция условного перехода «перейти, если равно».
JMP	Используется в программе для организации безусловного перехода.
MOV	Команда пересылки данных (из источника в приемник).
FCOM	Сравнивает два операнда.
FCOM	Сравнивает с «выбросом» два операнда.
FLDZ	Поместить в стек +0,0

Использованные в программе библиотеки приведены в таблице 4.2.

Таблица 4.2 – Описание библиотек

Библиотека	Назначение
"stdafx.h"	Служит для генерации файла предкомпилированных заголовков; в него включено большинство стандартных и используемых в каждом приложении включаемых файлов. Сделано это для того, чтобы ускорить компиляцию проекта.
<stdio.h>	Используется для организации стандартного ввода/вывода.
<iostream>	Используется для организации потокового ввода/вывода.

4.2 Задание

1) В программе необходимо реализовать функцию вычисления заданного условного выражения на языке ассемблера с использованием команд арифметического сопроцессора.

$$X = \begin{cases} b / a + 10, & \text{если } a > b; \\ -20, & \text{если } a = b; \\ (b - a) / b, & \text{если } a < b; \end{cases}$$

2) Значения переменных передаются в качестве параметров функции.

3) В программе реализовать вывод результата на экран.

4) Все параметры функции имеют тип double.

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

6) В качестве комментария к каждой строке необходимо указать, какой промежуточный результат, в каком регистре формируется.

7) В качестве комментария к строкам, содержащим команды сопроцессора необходимо указать состояние регистров сопроцессора.

8) Результат можно возвращать из функции в вершине стека сопроцессора.

4.3 Схема алгоритма

На рисунке 4.1 приведена схема алгоритма вычисления функции

$$X = \begin{cases} b/a + 10, & \text{если } a > b; \\ -20, & \text{если } a = b; \\ (b-a)/b, & \text{если } a < b; \end{cases}$$

Для того чтобы вычислить результат необходимо организовать ввод с клавиатуры исходных данных (параметров а, b). После ввода сравниваются значения введенных параметров. Если $a > b$, то результат есть значение функции $(b/a + 10)$, если $a < b$, то результат есть значение функции $((b - a)/b)$, иначе результат равен -20.

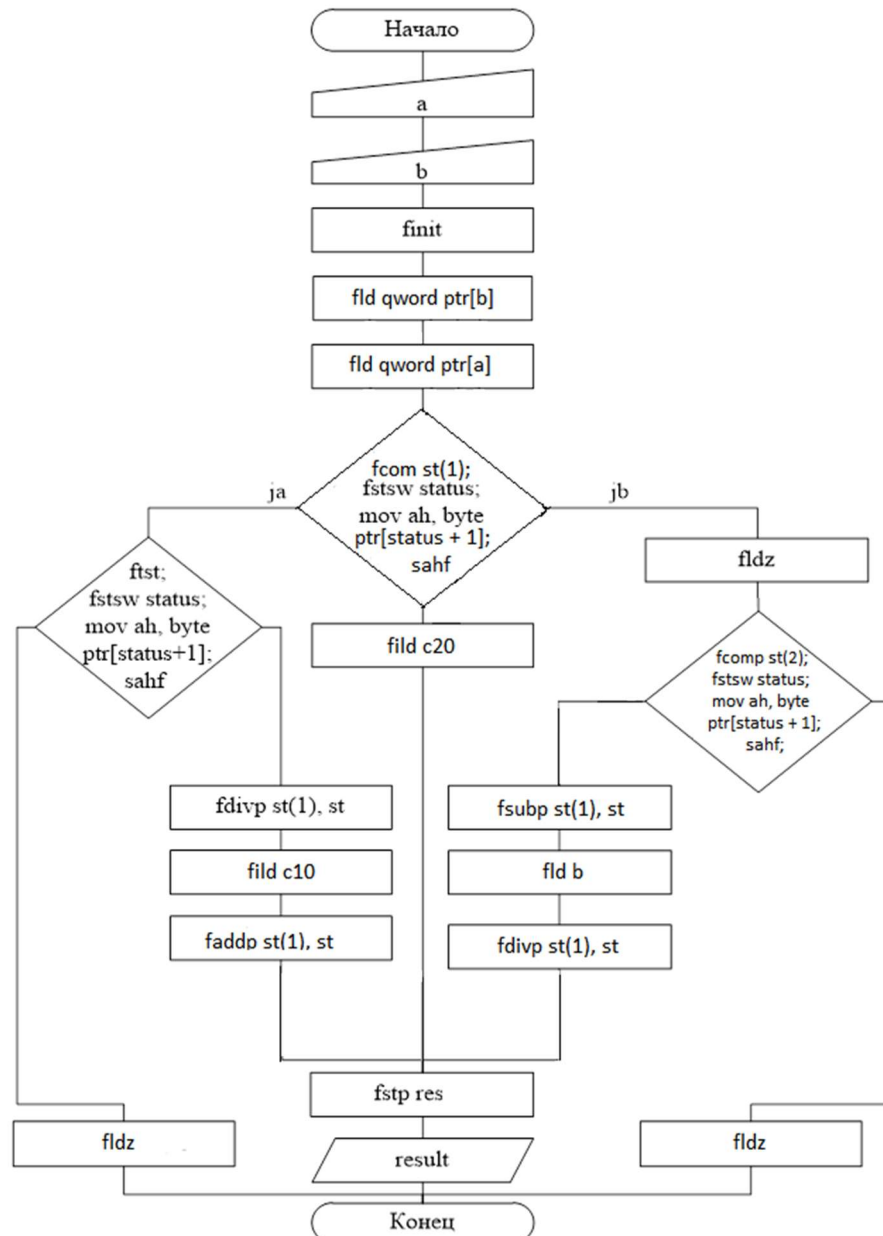


Рисунок 4.1 – Схема алгоритма вычисления исходного выражения

4.4 Решение

```
#include "stdafx.h"
#include <stdio.h> // стандартный ввод/вывод
#include <iostream> // потоковый ввод/вывод
using namespace std;

// функция вычисления выражения
// * (b/a) + 10, если a > b
// * -20, если a = b
// * (b - a)/b, если a < b

double CalcAsm(double a, double b)
{
    double res; int status;
    const int c10 = 10;
    const int c20 = -20;

    __asm {
        finit;                                инициализация сопроцессора
        fld qword ptr[b];                     b
        fld qword ptr[a];                     a, b
        fcom st(1);                            сравниваем a и b
        fstsw status;                         сохраняем регистр флагов
сопроцессора
        mov ah, byte ptr[status + 1]
        sahf;                                записываем в регистр флагов
процессора
        ja a_bigger;                          переход если a больше
        jb b_bigger;                          переход если b больше
        ;                                    если равны
        fild c20;                             -20, a, b
        jmp endcalc;
    a_bigger:
        ftst;                                сравнение a с 0
        fstsw status;                         сохраняем регистр флагов
сопроцессора
        mov ah, byte ptr[status + 1]
        sahf;                                записываем в регистр флагов
процессора
        je error;                             переход если a = 0
        fdivp st(1), st;                       b / a
        fild c10;                             4, b / a
        faddp st(1), st;                       b / a + 10
        jmp endcalc;
    b_bigger:; тут(a ^ 3 - 5) / b
        fldz;                                a у меня(b - a) / b, если a < b
        fcomp st(2);                          0, a, b
        fstsw status;                         сравнение b с 0
сопроцессора
        mov ah, byte ptr[status + 1];          сохраняем регистр флагов
        sahf;                                записываем в регистр флагов
процессора
        je error;                             переход если b = 0
        fsubp st(1), st;
        fld b;
        fdivp st(1), st;
        jmp endcalc;
    error:
        fldz;                                формируем результат ошибки
    endcalc:
        fstp res;                            сохранение результата
    }

    return res;
}
```



```

}

double CalcC(double a, double b) // [a > b] (b/a) + 10; [a = b] -20; [a < b] (b - a)/b
{
    if (a > b) {
        return ((b / a) + 10);
    }
    else {
        if (a < b) {
            return ((b - a) / b);
        }
        else {
            return (-20);
        }
    }
}

int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_CTYPE, "rus");
    double a, b;
    bool error = 0;

    printf("Лабораторная работа № 3\n");
    printf("Вариант 13\n");
    printf("Выполнил: Кузнецов Андрей, группа 6113\n\n");
    printf("Задание:\n");
    printf("Вычисление выражения с условием:\n* (b/a) + 10, если a > b\n* -20, если a  
= b\n* (b - a)/b, если a < b\n\n");

    printf("Введите a = "); // стандартный вывод
    cin >> a; // потоковый ввод
    printf("Введите b = ");
    cin >> b;

    double res = CalcAsm(a, b); // вычисление выражения
    if (error == 1) {
        cout << "Ошибка" << endl;
    }
    else {
        printf("\nРезультат на ассемблере: ");
        printf("%f", res); // вывод результата вычисления выражения
        printf("\nРезультат на C++: ");
        printf("%f\n", CalcC(a, b));
    }

    printf("\n"); // стандартный вывод
    system("PAUSE");
    return 0;
}

```

4.5 Ответ

На рисунках 4.2, 4.3, 4.4 приведены тестовые примеры выполнения программы при вводе корректных исходных данных. На рисунке 4.5 приведен результат работы программы в случае ввода недопустимых значений, когда делитель равен 0. Программа выполняет проверку данного значения и выдает предупредительное сообщение.

```
Лабораторная работа № 4
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
Вычисление выражения с условием:
*  $(b/a) + 10$ , если  $a > b$ 
*  $-20$ , если  $a = b$ 
*  $(b - a)/b$ , если  $a < b$ 

Введите a = 13
Введите b = 7

Результат на ассемблере: 10.538462
Результат на C++: 10.538462

Для продолжения нажмите любую клавишу . . .
```

Рисунок 4.2 – Работа программы в случае $a > b$

```
Лабораторная работа № 4
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
Вычисление выражения с условием:
*  $(b/a) + 10$ , если  $a > b$ 
*  $-20$ , если  $a = b$ 
*  $(b - a)/b$ , если  $a < b$ 

Введите a = 11
Введите b = 11

Результат на ассемблере: -20.000000
Результат на C++: -20.000000

Для продолжения нажмите любую клавишу . . .
```

Рисунок 4.3 – Работа программы в случае $a = b$

```
лабораторная работа № 4
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
Вычисление выражения с условием:
*  $(b/a) + 10$ , если  $a > b$ 
*  $-20$ , если  $a = b$ 
*  $(b - a)/b$ , если  $a < b$ 

Введите a = 6
Введите b = 13

Результат на ассемблере: 0.538462
Результат на C++: 0.538462

для продолжения нажмите любую клавишу . . .
```

Рисунок 4.4 – Работа программы в случае $a < b$

```
лабораторная работа № 4
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
Вычисление выражения с условием:
*  $(b/a) + 10$ , если  $a > b$ 
*  $-20$ , если  $a = b$ 
*  $(b - a)/b$ , если  $a < b$ 

Введите a = -5
Введите b = 0

Результат на ассемблере: 0.000000
Результат на C++: 0.000000

для продолжения нажмите любую клавишу . . .
```

Рисунок 4.5 – Работа программы в случае некорректного ввода данных

ЛАБОРАТОРНАЯ РАБОТА 5

«Нахождения функций с помощью математического сопроцессора»

5.1 Теоретические основы лабораторной работы

Команды, использованные в программе, приведены в таблице 5.1.

Таблица 5.1 – Описание команд

Команда	Назначение
FINIT	Инструкция инициализации сопроцессора.
FLD	Загружает из памяти в вершину стека ST(0) вещественное число.
FSIN	Вычисление значения синуса ST(0).
FMULP	Умножение с «выбросом» результата в стек.
FADDP	Сложение с «выбросом» результата в стек.
FSTP	Извлекает из вершины стека ST(0) в память вещественное число.

Использованные в программе библиотеки приведены в таблице 5.2.

Таблица 5.2 – Описание библиотек

Библиотека	Назначение
"stdafx.h"	Служит для генерации файла предкомпилированных заголовков; в него включено большинство стандартных и используемых в каждом приложении включаемых файлов. Сделано это для того, чтобы ускорить компиляцию проекта.
<stdio.h>	Используется для организации стандартного ввода/вывода.
<iostream>	Используется для организации потокового ввода/вывода.

5.2 Задание

1) В программе необходимо реализовать функцию $f(x) = 3 - 2 * x * x$, зависящую от аргумента x на языке ассемблера с использованием команд арифметического сопроцессора.

- 2) Значения переменных передаются в качестве параметров функции.
- 3) Составить таблицу значений функции на отрезке $[-1, 3]$ с шагом $h = 0,1$.
- 4) Номер вычисления №, значения x и $f(x)$ вывести на экран.
- 5) Все параметры функции имеют тип double.
- 6) В качестве комментария к каждой строке необходимо указать, какой промежуточный результат, в каком регистре формируется.
- 7) В качестве комментария к строкам, содержащим команды сопроцессора необходимо указать состояние регистров сопроцессора.
- 8) Результат можно возвращать из функции в вершине стека сопроцессора.

5.3 Схема алгоритма

На рисунке 5.1 приведена схема алгоритма вычисления функции $f(x) = 3 - 2 * x * x$. Значение параметра x поступает из цикла основной программы. Сначала вычисляется значение многочлена $2 * x * x$, затем из 3 вычитается этот многочлен.

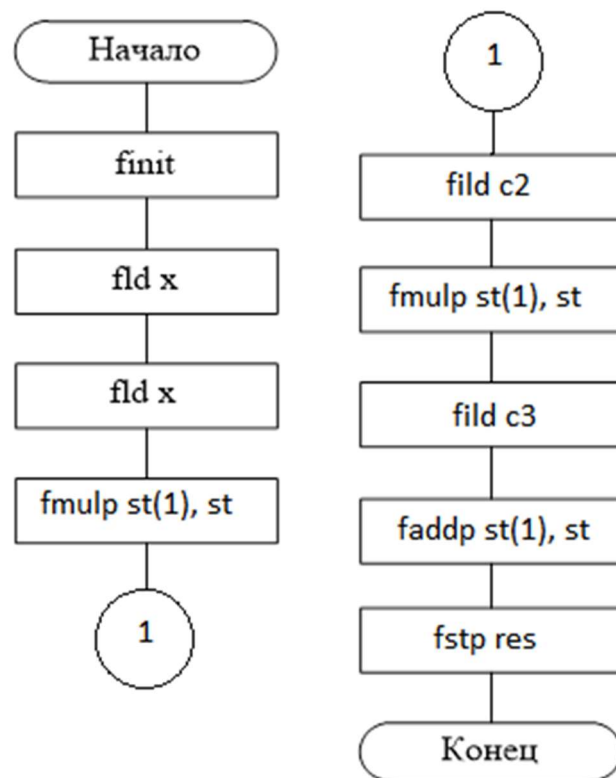


Рисунок 5.1 – Схема алгоритма вычисления выражения $f(x) = 3 - 2 * x * x$.

5.4 Решение

```
#include "stdafx.h"
#include <stdio.h> // стандартный ввод/вывод
#include <iostream> // потоковый ввод/вывод
using namespace std;

// y = 3 - 2 * x * x
double CalcAsm(double x)
{
    double res;
    int c2 = -2, c3 = 3;

    __asm {
        finit;
        fld x;
        fld x;
        fmulp st(1), st;
        fild c2;
        fmulp st(1), st;
        fild c3;
        faddp st(1), st;
        fstp res;
    }
    return res;
}

double CalcC(double x)
{
    return (3 - 2 * x * x);
}

int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_CTYPE, "rus");

    printf("Лабораторная работа № 4\n");
    printf("Вариант 13\n");
    printf("Выполнил: Кузнецов Андрей, группа 6113\n");
    printf("Задание:\n");
    printf("Вычислить выражения у = 3 - 2 * x * x с шагом 0,1\n");
    printf("\n\t\t\tТаблица ответов");
    printf("\n");
    printf("=====\n");
    std::cout << " " << " № " << "\t\t" << " x " << "\t\t" << " АСЕМБЛЕР" << "\t\t"
<< "C++";
    printf("\n");
    printf("=====\n");
    int k = 1;
    for (double i = -1; i <= 3.1; i = i + 0.1)
    {
        printf(" %d \t\t% 0.1f \t\t % 0.4f \t\t% 0.4f", k++, i, CalcAsm(i),
CalcC(i));
        printf("\n");
    }
    printf("=====\n");
    system("PAUSE");
    return 0;
}
```

5.5 Ответ

На рисунке 5.2 приведён тестовый пример работы программы.

Лабораторная работа № 4
Вариант 13
Выполнил: Кузнецов Андрей, группа 6113

Задание:
Вычислить выражения $y = 3 - 2 * x * x$ с шагом 0,1

Таблица ответов

№	x	АСЕМБЛЕР	C++
1	-1.0	1.0000	1.0000
2	-0.9	1.3800	1.3800
3	-0.8	1.7200	1.7200
4	-0.7	2.0200	2.0200
5	-0.6	2.2800	2.2800
6	-0.5	2.5000	2.5000
7	-0.4	2.6800	2.6800
8	-0.3	2.8200	2.8200
9	-0.2	2.9200	2.9200
10	-0.1	2.9800	2.9800
11	-0.0	3.0000	3.0000
12	0.1	2.9800	2.9800
13	0.2	2.9200	2.9200
14	0.3	2.8200	2.8200
15	0.4	2.6800	2.6800
16	0.5	2.5000	2.5000
17	0.6	2.2800	2.2800
18	0.7	2.0200	2.0200
19	0.8	1.7200	1.7200
20	0.9	1.3800	1.3800
21	1.0	1.0000	1.0000
22	1.1	0.5800	0.5800
23	1.2	0.1200	0.1200
24	1.3	-0.3800	-0.3800
25	1.4	-0.9200	-0.9200
26	1.5	-1.5000	-1.5000
27	1.6	-2.1200	-2.1200
28	1.7	-2.7800	-2.7800
29	1.8	-3.4800	-3.4800
30	1.9	-4.2200	-4.2200
31	2.0	-5.0000	-5.0000
32	2.1	-5.8200	-5.8200
33	2.2	-6.6800	-6.6800
34	2.3	-7.5800	-7.5800
35	2.4	-8.5200	-8.5200
36	2.5	-9.5000	-9.5000
37	2.6	-10.5200	-10.5200
38	2.7	-11.5800	-11.5800
39	2.8	-12.6800	-12.6800
40	2.9	-13.8200	-13.8200
41	3.0	-15.0000	-15.0000

для продолжения нажмите любую клавишу . . .

Рисунок 5.2 – Работа программы

ЛАБОРАТОРНАЯ РАБОТА 6

«Нахождение суммы ряда с помощью математического сопроцессора»

6.1 Теоретические основы лабораторной работы

Команды, использованные в программе, приведены в таблице 6.1.

Таблица 6.1 – Описание команд

Команда	Назначение
FINIT	Инструкция инициализации сопроцессора.
FSTSW	Сохраняет текущее значение регистра SR в приемник.
SAHF	Загружает флаги из регистра AH.
FLD	Загружает из памяти в вершину стека ST(0) вещественное число.
FLD1	Поместить в стек "1"
FSTP	Извлекает из вершины стека ST(0) в память вещественное число.
FADD	Сложение вещественных чисел.
FMUL	Умножение вещественных чисел.
JNE	Инструкция условного перехода «перейти, если не нуль или не равно...»
INC	Увеличивает значение операнда в памяти или регистре на 1.
MOV	Команда пересылки данных (из источника в приемник).
CMR	Сравнивает два операнда.

Использованные в программе библиотеки приведены в таблице 6.2.

Таблица 6.2 – Описание библиотек

Библиотека	Назначение
"stdafx.h"	Служит для генерации файла предкомпилированных заголовков.

Продолжение таблицы 6.2

<stdio.h>	Используется для организации стандартного ввода/вывода.
<iostream>	Используется для организации потокового ввода/вывода.

6.2 Задание

1) В программе необходимо реализовать функцию определения значения функции $sh\ x$, зависящей от аргумента x на языке ассемблера с использованием команд арифметического сопроцессора.

2) Функция вычисляется в виде суммы ряда: $shx = \sum_{k=0}^{\infty} \frac{x^{2k+1}}{(2k+1)!}$.

3) Значение параметров x и N передаются в качестве аргументов.

4) В программе необходимо также реализовать функцию вычисления значения элементарной функции на основе аналитического выражения. Значение функции вывести для контроля на экран.

5) Необходимо определить достигнутую погрешность, вычислив отклонение аналитического значения от вычисленного значения.

6) В качестве комментария к строкам содержащим команды сопроцессора необходимо указать состояние регистров сопроцессора.

6.3 Схема алгоритма

На рисунке 6.1 приведена схема алгоритма вычисления значения функции $sh\ x$. Для того чтобы вычислить результат, необходимо организовать ввод с клавиатуры исходных данных (x , N). Запускается цикл разложения в ряд $sh\ x$: нахождение выражений $x^{(2k+1)}$, $(2k+1)!$ и их частного, которое затем умножается на x ; заметим, что степень числителя последующего члена отличается на 2 и что факториал знаменателя – также отличается на 2. Условием выхода из цикла является превышение заданного числа шагов (N) количеством итераций цикла. Результатом работы программы является значение суммы элементов.

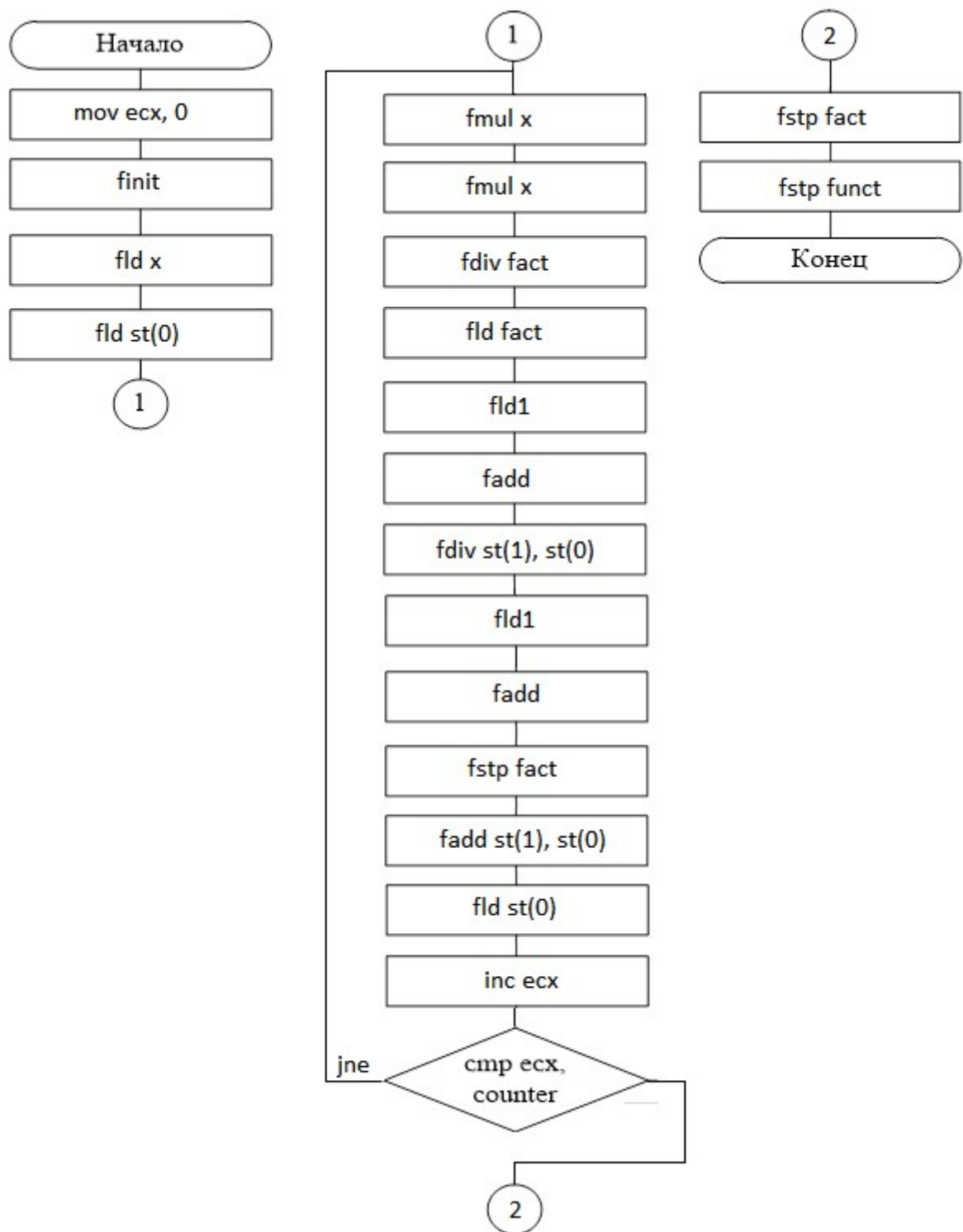


Рисунок 6.1 – Схема алгоритма разложения в ряд

6.4 Решение

```

#include "stdafx.h"
#include <stdio.h> // стандартный ввод/вывод
#include <iostream> // потоковый ввод/вывод

using namespace std;

double CalcAsm(double x, int N)

```

```
{
    double fact = 2.0;
    double funct = 0.0;
    __asm {
        mov ecx, 0;
        finit;
        fld x;
        fld st(0);
Cycle:
        fmul x;
        fmul x;
        fdiv fact;
        fld fact;
        fld1;
        fadd;
        fdiv st(1), st(0);
        fld1;
        fadd;
        fstp fact;
        fadd st(1), st(0);
        ; fld st(0);
        inc ecx
        cmp ecx, N
        jne Cycle
        fstp fact;
        fstp funct;
    }
    return funct;
}

int _tmain(int argc, _TCHAR* argv[])
{
    setlocale(LC_ALL, "RUSSIAN");
    printf("Лабораторная работа № 6\n");
    printf("Вариант 13\n");
    printf("Выполнил Кузнецов Андрей, группа 6113\n");
    printf("Задание:\n* Определить значение sh x.\n\n");

    double x;
    printf("Введите значение x: ");
    std::cin >> x;

    int N;
    do {
        printf("Введите количество итераций (количество членов в ряду): ");
        std::cin >> N;
    } while (N < 1);
    printf("\n");

    printf("\n\t\t\t\t\tТаблица ответов");
    printf("\n");

    printf("=====\n");
    std::cout << " " << " № " << "\t" << " Ассемблер" << "\t\t" << "C++" <<
"\t\t\t" << "погрешность";
    printf("\n");

    printf("=====\n");

    for (int i = 1; i <= N; i++)
    {
        printf(" %d \t %9.12f \t %9.12f \t %9.12f", i, CalcAsm(x, i),
sinh(x), abs(sinh(x)-CalcAsm(x, i)));
        printf("\n");
    }
}
```

```

    }
    printf("=====\\
n\\n");
    system("PAUSE");
    return 0;
}

```

6.5 Ответ

На рисунке 6.2 приведены тестовые примеры выполнения программы при вводе корректных исходных данных. На рисунке 6.3 приведен результат работы программы в случае попыток ввода некорректных значений. Программа выполняет проверку вводимых значений и позволяет исправить некорректный параметр.

```

лабораторная работа № 6
Вариант 13
Выполнил Кузнецов Андрей, группа 6113
Задание:
* Определить значение sh x.

Введите значение x: 3
Введите количество итераций (количество членов в ряду): -2
Введите количество итераций (количество членов в ряду): 2

Таблица ответов
=====
№      Ассемблер      C++      погрешность
=====
1      7,50000000000000    10,017874927410    2,517874927410
2      9,52500000000000    10,017874927410    0,492874927410
=====

Для продолжения нажмите любую клавишу . . .

```

Рисунок 6.2 – Работа программы в случае верного ввода исходных данных

```
лабораторная работа № 6
Вариант 13
Выполнил Кузнецов Андрей, группа 6113
Задание:
* Определить значение sh x.

Введите значение x: 3
Введите количество итераций (количество членов в ряду): -2
Введите количество итераций (количество членов в ряду): 2

                        Таблица ответов
=====
№      Ассемблер          C++          погрешность
=====
1      7,500000000000      10,017874927410      2,517874927410
2      9,525000000000      10,017874927410      0,492874927410
=====

для продолжения нажмите любую клавишу . . .
```

Рисунок 6.3 – Работа программы при попытке ввода некорректных данных