Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №5  
по курсу «Программирование»  
на тему «Процедуры»

Выполнили студенты группы 22ВВП1:  
Беляев Д.

Ипполитов И.

Приняли:  
Слепцов Н.В  
Голотенков Н.О

Пенза 2023

**Название**

Процедуры

**Цель работы**

Получение навыков разработки структурной организации ассемблерных программ и ее реализации на основе аппарата процедур, изучение способов организации связи по данным между процедурами

**Лабораторное задание**

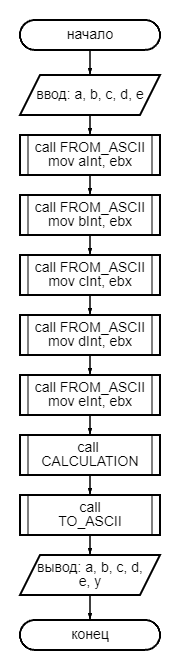
Разработать программу вычисления функции *y=f(a,b,c,d,e).* Значения аргументов *a, b, c, d, e* должны вводиться с клавиатуры ПЭВМ. Значение функции *y* следует вывести на экран дисплея.

**Метод решения задачи**

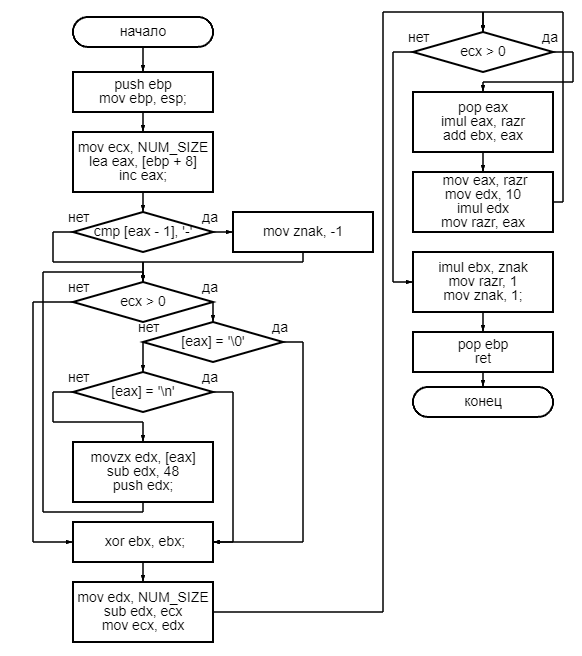
Объявили и инициализировали переменные в С, с помощью ассемблерной вставки выполнили алгоритм в соответствии с условием

**Блок-схема программы**

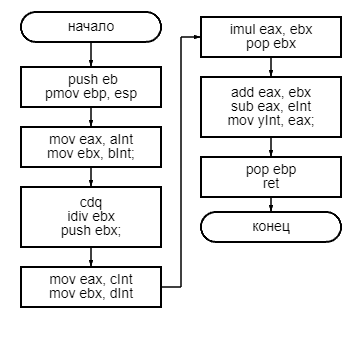
* **Main**



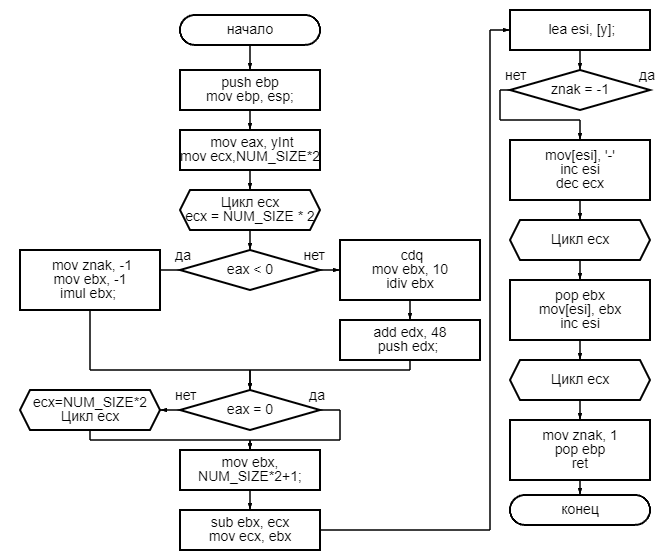
* **FROM\_ASCII**

****

* **CALCILATION**

****

* **TO\_ASCII**

****

**Листинг**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#define NUM\_SIZE 3

// y=a/b+c\*d-e

char a[NUM\_SIZE] = { 0 };

char b[NUM\_SIZE] = { 0 };

char c[NUM\_SIZE] = { 0 };

char d[NUM\_SIZE] = { 0 };

char e[NUM\_SIZE] = { 0 };

int aInt = 0, bInt = 0, cInt = 0, dInt = 0, eInt = 0;

char y[NUM\_SIZE \* 2] = { 0 };

int yInt = -15;

int znak = 1;

int razr = 1;

int main()

{

#pragma region Read Values

printf("Insert a (-99, +99): ");

fgets(a, 4, stdin);

getchar();

printf("Insert b (-99, +99): ");

fgets(b, 4, stdin);

getchar();

printf("Insert c (-99, +99): ");

fgets(c, 4, stdin);

getchar();

printf("Insert d (-99, +99): ");

fgets(d, 4, stdin);

getchar();

printf("Insert e (-99, +99): ");

fgets(e, 4, stdin);

getchar();

#pragma endregion

\_asm

{

#pragma region From ASCII

push a;

call FROM\_ASCII;

mov aInt, ebx;

pop a;

push b;

call FROM\_ASCII;

mov bInt, ebx;

pop b;

push c;

call FROM\_ASCII;

mov cInt, ebx;

pop c;

push d;

call FROM\_ASCII;

mov dInt, ebx;

pop d;

push e;

call FROM\_ASCII;

mov eInt, ebx;

pop e;

#pragma endregion

call CALCULATION;

call TO\_ASCII;

// Выход из программы

jmp EXIT;

#pragma region Functions

// ------------------- //

FROM\_ASCII:

push ebp;

mov ebp, esp;

mov ecx, NUM\_SIZE; // Кладем в ecx размер строки

lea eax, [ebp + 8]; // кладем в eax ссылку на строку

inc eax;

cmp [eax - 1], '-'; // сравниваем, не является ли текущий символ "-"

jz MINUS\_F; // если да то преходим по метке MINUS

jmp SEPARATE;

MINUS\_F:

mov znak, -1; // меняем знак на отрицательный

SEPARATE:

cmp[eax], '\0'; // смотрим, не закончилась ли строка

je END\_FA; //

cmp[eax], '\n'; //

je END\_FA; // если да, то вызодим из цикла

movzx edx, [eax]; // конвертируем символ из ASCII в нормальный вид

sub edx, 48; // вычитаем 48, чтобы полчить число в привычном виде

push edx;

END\_FA\_LOOP:

inc eax;

loop SEPARATE;

END\_FA:

xor ebx, ebx; // Итоговое число

mov edx, NUM\_SIZE; // считаем счетчик

sub edx, ecx; //

mov ecx, edx; //

TO\_INT:

pop eax; // Забираем цифру из массива

imul eax, razr; // Умножаем ее на разряд

add ebx, eax; // Приавляем к итоговому числу

mov eax, razr; // повышаем разряд

mov edx, 10; //

imul edx; //

mov razr, eax; //

loop TO\_INT;

cdq;

imul ebx, znak; // умножаем число на знак

mov razr, 1;

mov znak, 1;

pop ebp;

ret;

// ------------------- //

CALCULATION:

push ebp;

mov ebp, esp;

mov eax, aInt;

mov ebx, bInt;

cdq;

idiv ebx;

push ebx;

mov eax, cInt;

mov ebx, dInt;

imul eax, ebx;

pop ebx;

add eax, ebx;

sub eax, eInt;

mov yInt, eax;

pop ebp;

ret;

// ------------------- //

TO\_ASCII:

push ebp;

mov ebp, esp;

mov eax, yInt;

mov ecx, NUM\_SIZE \* 2;

DIVIDE:

cmp eax, 0; // смотрим, отрицательное ли число или нет

jl MINUS\_T; // если да, то перезодим на метку MINUS\_T

PLUS\_T:

cdq;

mov ebx, 10; // делим число на 10,

idiv ebx; // чтобы получить отдельную цифру

add edx, 48;

push edx;

jmp END\_TA\_LOOP;

MINUS\_T:

mov znak, -1;

mov ebx, -1; // меняем знак числа

imul ebx; //

END\_TA\_LOOP:

cmp eax, 0; // смотрим не закончилось ли число

je END\_DIVIDE; // если закончилось то выходим из цикла

loop DIVIDE;

END\_DIVIDE:

mov ebx, NUM\_SIZE \* 2 + 1; // считаем счетчик

sub ebx, ecx; //

mov ecx, ebx; //

lea esi, [y]; // кладем в eax ссылку на начало y

cmp znak, -1; // смотрим не отрицательное ли число

jne TO\_ASCII\_LOOP; // если да

mov[esi], '-'; // то в начало y кладем -

inc esi; //

dec ecx; //

TO\_ASCII\_LOOP:

pop ebx; // берем остаток числ из стека

mov[esi], ebx; // кладем его в y

inc esi; //

loop TO\_ASCII\_LOOP;

mov znak, 1;

pop ebp;

ret;

#pragma endregion

EXIT:

}

printf("Ishod dannye: %d, %d, %d, %d, %d\n", aInt, bInt, cInt, dInt, eInt);

printf("Result: y = a/b+c\*d-e = %s", y);

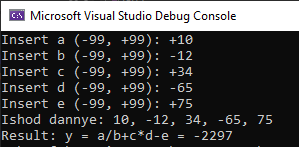
return 0;

}

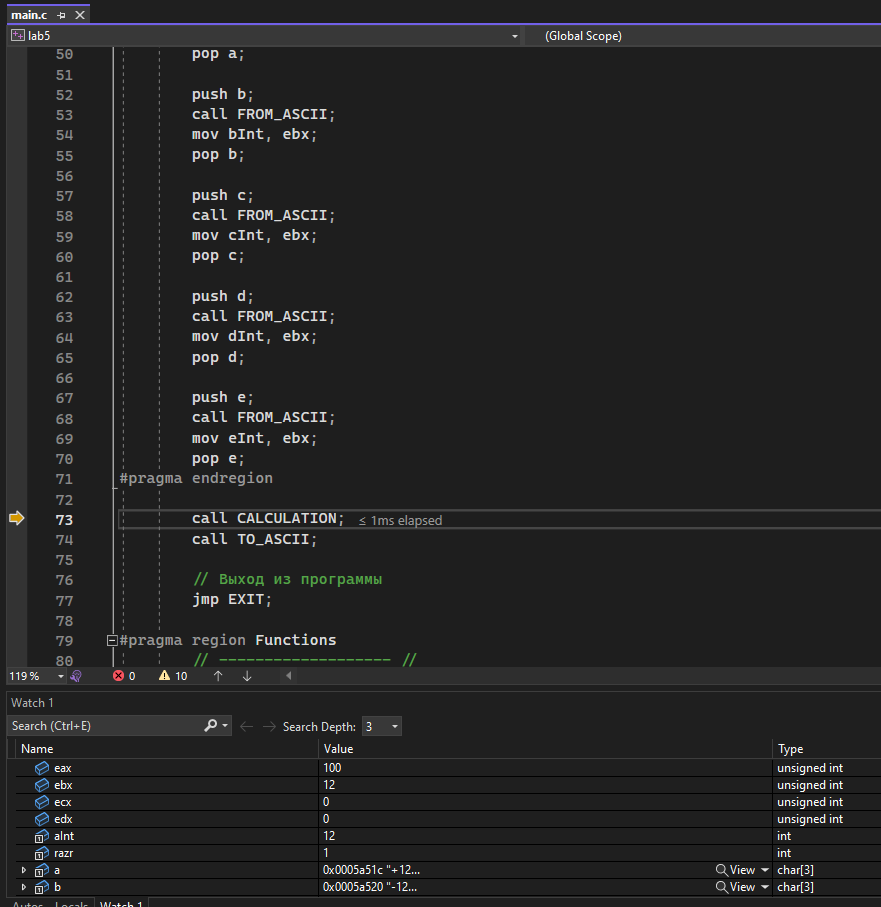
**Пояснительный текст к программе**

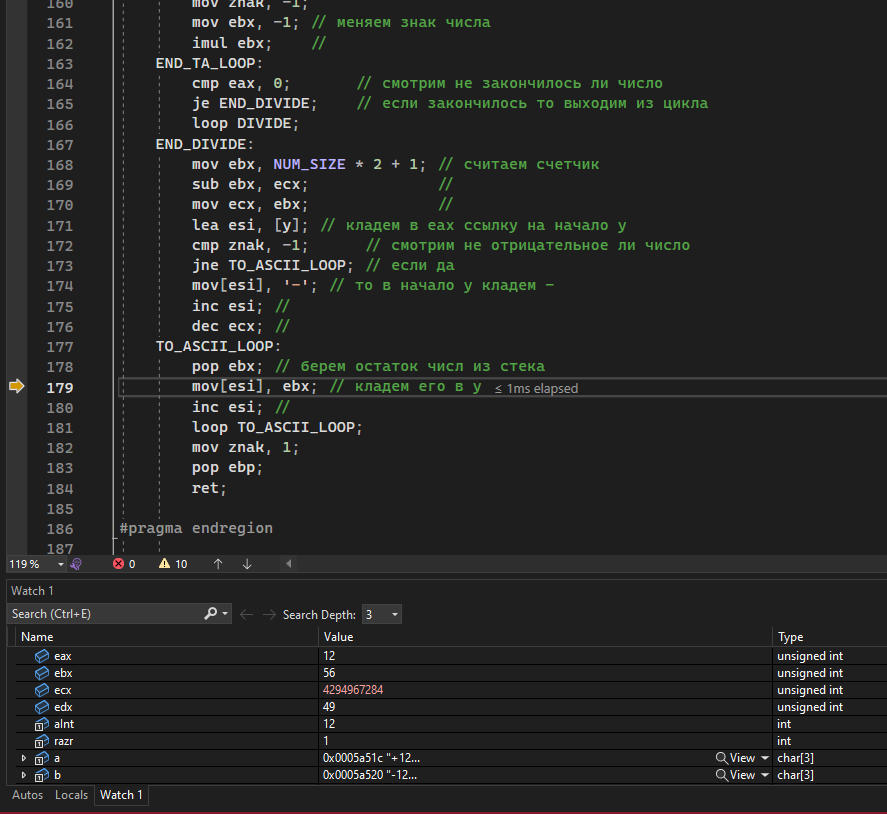
У пользователя запрашиваются на вход числа a, b, c, d, e, где первый символ знак. Далее программа помещает каждую переменную в стек и переводит их в нормальный вид. После программа считает результат функции *y=a/b+c\*d-e.* Далее программа переводит результат в код ASCII и выводит результат

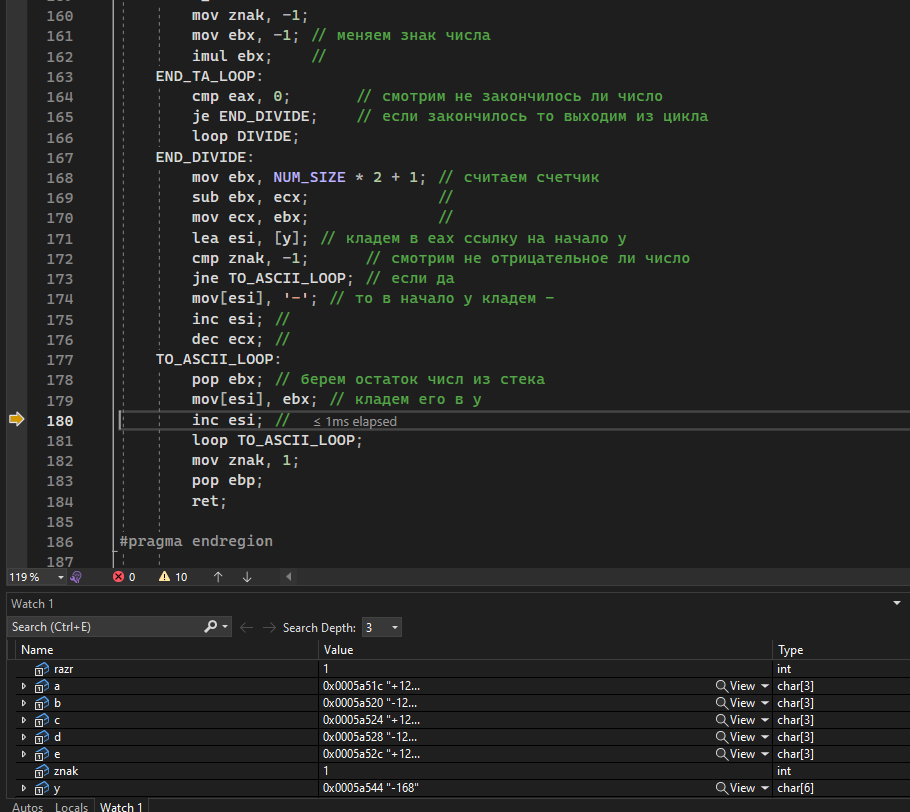
**Результат работы программы**

****

**Протокол трассировки программы**

****





**Вывод**

Получили навыки разработки структурной организации ассемблерных программ и ее реализации на основе аппарата процедур, изучили способы организации связи по данным между процедурами