Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Системне програмування**

**Лабораторна робота №10**

«Використання у проекті С++ з використанням модулів на асемблері»

Виконав:

студент групи ІО-82

Шендріков Є. О.

Залікова № 8227

Перевірив Порєв В. М.

Київ – 2020

**Мета**

Навчитися створювати програми на С++ з використанням модулів на асемблері.

**Завдання**

1. Створити у середовищі MS Visual Studio проект C++ з ім’ям Lab10.

2. Написати вихідний текст програми згідно варіанту завдання. У проекті мають бути такі файли вихідного тексту:

- головний файл: lab10.cpp

- файли двох модулів на асемблері: module.asm та longop.asm.

3. У цьому проекті кожний модуль може окремо компілюватися.

4. Скомпілювати вихідний текст і отримати виконуємий файл програми.

5. Перевірити роботу програми. Налагодити програму.

6. Отримати результати – кодовані значення чисел згідно варіанту завдання.

7. Проаналізувати та прокоментувати результати, вихідний текст та машинний код програми.

**Мій варіант:**

|  |  |  |
| --- | --- | --- |
| № варіанту | Вираження | Розрядність |
| 25 |  | 352 |

**Текст програми**

*lab10.cpp*

// Lab10.cpp : Defines the entry point for the application.

//

#include "stdafx.h"

#include "Lab10.h"

#include "longop.h"

#include "module.h"

#define MAX\_LOADSTRING 100

// Global Variables:

HINSTANCE hInst; // поточний екземпляр TCHAR szTitle[MAX\_LOADSTRING]; // текст рядка заголовка TCHAR szWindowClass[MAX\_LOADSTRING]; // ім’я класу головного вікна

// Форвардні оголошення функцій, включених в цей модуль коду:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY \_tWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPTSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Place code here.

MSG msg;

HACCEL hAccelTable;

// Ініціалізація глобальних рядків

LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadString(hInstance, IDC\_LAB10, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Ініціалізація додатку

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB10));

// Цикл основного повідомлення:

while (GetMessage(&msg, NULL, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// FUNCTION: MyRegisterClass()

//

// PURPOSE: реєструє клас вікна.

//

// КОМЕНТАР:

//

// Ця функція і її використання необхідні тільки в разі, якщо потрібно, щоб даний код

// був сумісний з системами Win32, що не мають функції RegisterClassEx

// яка була додана в Windows 95. Виклик цієї функції важливий для того,

// щоб додаток отримав "якісні" дрібні значки і встановив зв'язок з ними.

//

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LAB10));

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LAB10);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassEx(&wcex);

}

//

// FUNCTION: InitInstance(HINSTANCE, int)

//

// PURPOSE: зберігає обробку примірника і створює головне вікно.

//

// КОМЕНТАРІ:

//

// У даній функції дескриптор екземпляра зберігається в глобальній змінній, а також

// створюється і виводиться на екран головне вікно програми.

//

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

HWND hWnd;

hInst = hInstance; // Зберегти дескриптор екземпляру в глобальній змінній

hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

void MyWork1(HWND hWnd)

{

long oA[11] = { 0x00000004, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

long oB[11] = { 0x00000005, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

long oC[11] = { 0x00000002, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

long oO[11] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

long result1[22] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

long result2[22] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

long result3[22] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

long result4[22] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

long result5[22] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

long result6[22] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,

0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000 };

char TextBuf[352];

char TextBuf10[44];

Mul\_NxN\_LONGOP(result1, oB, oB);

Add\_LONGOP(11, result2, result1, oA);

Mul\_NxN\_LONGOP(result3, oC, oC);

Mul\_NxN\_LONGOP(result4, result3, oC);

Sub\_LONGOP(22, result5, result4, result2);

Add\_LONGOP(22, result6, result5, result6);

StrToDec(44, 11, TextBuf10, result6);

MessageBox(hWnd, TextBuf10, "Результат A+B^2-C^3", MB\_OK);

StrHex\_MY(352, result6, TextBuf);

MessageBox(hWnd, TextBuf, "Результат A+B^2-C^3", MB\_OK);

}

//

// FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// PURPOSE: обробляє повідомлення в головному вікні.

//

// WM\_COMMAND - обробка меню програми

// WM\_PAINT - закрасити головне вікно

// WM\_DESTROY - ввести повідомлення про вихід і повернутися.

////

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

int wmId, wmEvent;

PAINTSTRUCT ps;

HDC hdc;

switch (message)

{

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

// Розібрати вибір в меню:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case ID\_32771:

MyWork1(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

// TODO: Add any drawing code here...

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// Оброблювач повідомлень для вікна "Про програму".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

*longop.asm*

.586

.model flat, c

.data

x dd 1h

bitNumber dd ?

a dd 0

b dd 0

r dd 0

.code

Add\_LONGOP proc bits:DWORD, dest:DWORD, pB:DWORD, pA:DWORD

mov esi, pA ;ESI = адреса A

mov ebx, pB ;EBX = адреса B

mov edi, dest ;EDI = адреса результату

mov ecx, bits; ECX = потрібна кількість повторень

mov edx, 0

clc ; обнулює біт CF регістру EFLAGS

mov edx, 0

cycle:

mov eax, dword ptr[esi+4\*edx]

adc eax, dword ptr[ebx+4\*edx] ; додавання групи з 32 бітів

mov dword ptr[edi+4\*edx], eax

inc edx; модифікація зсуву

dec ecx ; лічильник зменшуємо на 1

jnz cycle

ret

Add\_LONGOP endp

Sub\_LONGOP proc bits:DWORD, dest:DWORD, pB:DWORD, pA:DWORD

mov esi, pA ;ESI = адреса A

mov ebx, pB ;EBX = адреса B

mov edi, dest ;EDI = адреса результату

mov ecx, bits; ECX = потрібна кількість повторень

clc ; обнулює біт CF регістру EFLAGS

mov edx, 0

cycle:

mov eax, dword ptr[esi+4\*edx]

sbb eax, dword ptr[ebx+4\*edx] ; додавання групи з 32 бітів

mov dword ptr[edi+4\*edx], eax

inc edx; модифікація зсуву

dec ecx ; лічильник зменшуємо на 1

jnz cycle

ret

Sub\_LONGOP endp

Mul\_NxN\_LONGOP proc dest:DWORD, p2:DWORD, p1:DWORD

mov esi, p1

mov edi, p2

mov ebx, dest

mov dword ptr[a],0

mov dword ptr[b],0

mov dword ptr[r],0

mov ecx, 18

@cycle:

push ecx

mov ecx, 8

@cycleInner:

push ecx

mov ecx, a

mov eax, dword ptr[esi + 4 \* ecx]

mov ecx, b

mul dword ptr[edi + 4 \* ecx]

mov ecx, r

clc

adc eax, dword ptr[ebx + 4 \* ecx]

mov dword ptr[ebx + 4 \* ecx], eax

mov eax, dword ptr[ebx + 4 \* ecx]

adc edx, dword ptr[ebx + 4 \* ecx + 4]

mov dword ptr[ebx + 4 \* ecx + 4], edx

mov eax, dword ptr[ebx + 4 \* ecx + 4]

inc a

inc r

pop ecx

dec ecx

jnz @cycleInner

inc b

xor eax, eax

mov a, eax

mov eax, b

mov r, eax

pop ecx

dec ecx

jnz @cycle

ret

Mul\_NxN\_LONGOP endp

End

*module.asm*

.586

.model flat, c

.data

x dd 0h

x1 dd 0h

x2 dd 0h

b dd 0

fractionalPart db ?

two dd 2

buf dd 80 dup(0)

decCode db ?

buffer dd 128 dup(?)

.code

;процедура StrHex\_MY записує текст шістнадцятькового коду

;перший параметр - адреса буфера результату (рядка символів)

;другий параметр - адреса числа

;третій параметр - розрядність числа у бітах (має бути кратна 8)

StrHex\_MY proc proc bits:DWORD, src:DWORD, dest:DWORD

mov ecx, bits ;кількість бітів числа

cmp ecx, 0

jle @exitp

shr ecx, 3 ;кількість байтів числа

mov esi, src ;адреса числа

mov ebx, dest ;адреса буфера результату

@cycle:

mov dl, byte ptr[esi+ecx-1] ;байт числа - це дві hex-цифри

mov al, dl

shr al, 4 ;старша цифра

call HexSymbol\_MY

mov byte ptr[ebx], al

mov al, dl ;молодша цифра

call HexSymbol\_MY

mov byte ptr[ebx+1], al

mov eax, ecx

cmp eax, 4

jle @next

dec eax

and eax, 3 ;проміжок розділює групи по вісім цифр

cmp al, 0

jne @next

mov byte ptr[ebx+2], 32 ;код символа проміжку

inc ebx

@next:

add ebx, 2

dec ecx

jnz @cycle

mov byte ptr[ebx], 0 ;рядок закінчується нулем

@exitp:

ret

StrHex\_MY endp

;ця процедура обчислює код hex-цифри

;параметр - значення AL

;результат -> AL

HexSymbol\_MY proc

and al, 0Fh

add al, 48 ;так можна тільки для цифр 0-9

cmp al, 58

jl @exitp

add al, 7 ;для цифр A,B,C,D,E,F

@exitp:

ret

HexSymbol\_MY endp

StrToDec proc bytesOnScreen: dword, numberOfDd: dword, decCodeLocal: dword, strCodeLocal: dword

mov esi, strCodeLocal ;[ebp + 20] ;str code

mov edi, decCodeLocal ;[ebp + 16] ;dec code

mov eax, numberOfDd ;[ebp + 12]

mov x1, eax ; number of dd

mov eax, bytesOnScreen ;[ebp + 8]

mov x2, eax ; bytes on screan

push esi

push edi

push esi

push offset buffer

push x1

call COPY\_LONGOP

pop edi

pop esi

mov b, 0

xor ecx, ecx

xor ebx, ebx

cycle:

push ecx

push edi

push esi

push offset buf

push offset decCode

push x1

call DIV\_LONGOP

pop edi

mov ebx, b

mov al, byte ptr[decCode]

add al, 48

mov byte ptr[edi + ebx], al

xor ecx, ecx

cycleInner:

mov eax, dword ptr[buf + 4 \* ecx]

mov dword ptr[esi + 4 \* ecx], eax

mov dword ptr[buf + 4 \* ecx], 0

inc ecx

cmp ecx, x1

jl cycleInner

pop ecx

inc ecx

inc b

cmp ecx, x2

jl cycle

mov ebx, x2

mov eax, x2

xor edx, edx

div two

mov x2, eax

dec ebx

xor ecx, ecx

cycle1:

mov al, byte ptr[edi + ecx]

mov ah, byte ptr[edi + ebx]

mov byte ptr[edi + ecx], ah

mov byte ptr[edi + ebx], al

dec ebx

inc ecx

cmp ecx, x2

jl cycle1

push offset buffer

push esi

push x2

call COPY\_LONGOP

ret

StrToDec endp

DIV\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 20] ; number

mov edi, [ebp + 16] ;integer

mov ebx, [ebp + 12] ;fractional

mov eax, [ebp + 8] ; bytes

mov x, eax

push ebx

xor edx, edx

mov ecx, x

dec x

mov ebx,x

cycle :

push ecx

mov ecx, 10

mov eax, dword ptr[esi + 4 \* ebx]

div ecx

mov fractionalPart, dl

mov dword ptr[edi + 4 \* ebx], eax

dec ebx

pop ecx

dec ecx

jnz cycle

pop ebx

mov al, fractionalPart

mov byte ptr[ebx], al

pop ebp

ret 16

DIV\_LONGOP endp

COPY\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 16]

mov edi, [ebp + 12]

mov edx, [ebp + 8]

mov ecx, 0

@cycle:

mov eax, dword ptr[esi + 4 \* ecx]

mov dword ptr[edi + 4 \* ecx], eax

inc ecx

cmp ecx, edx

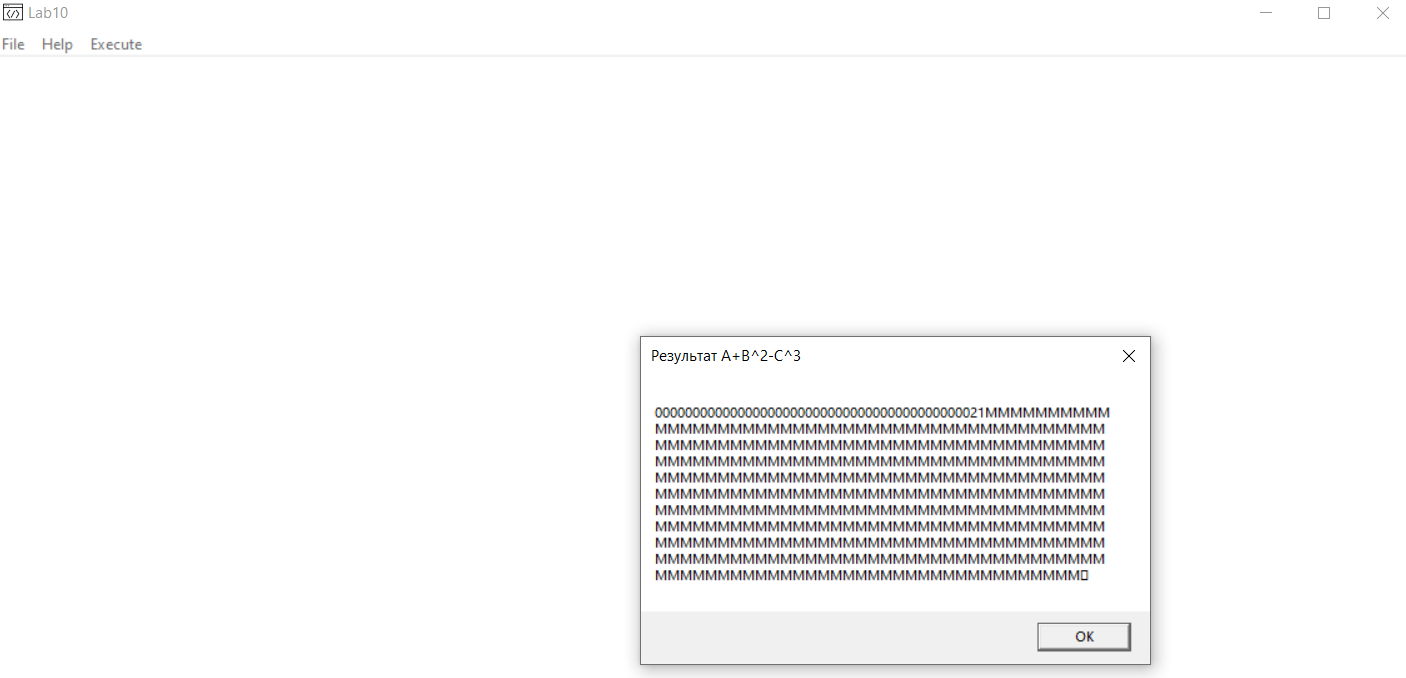
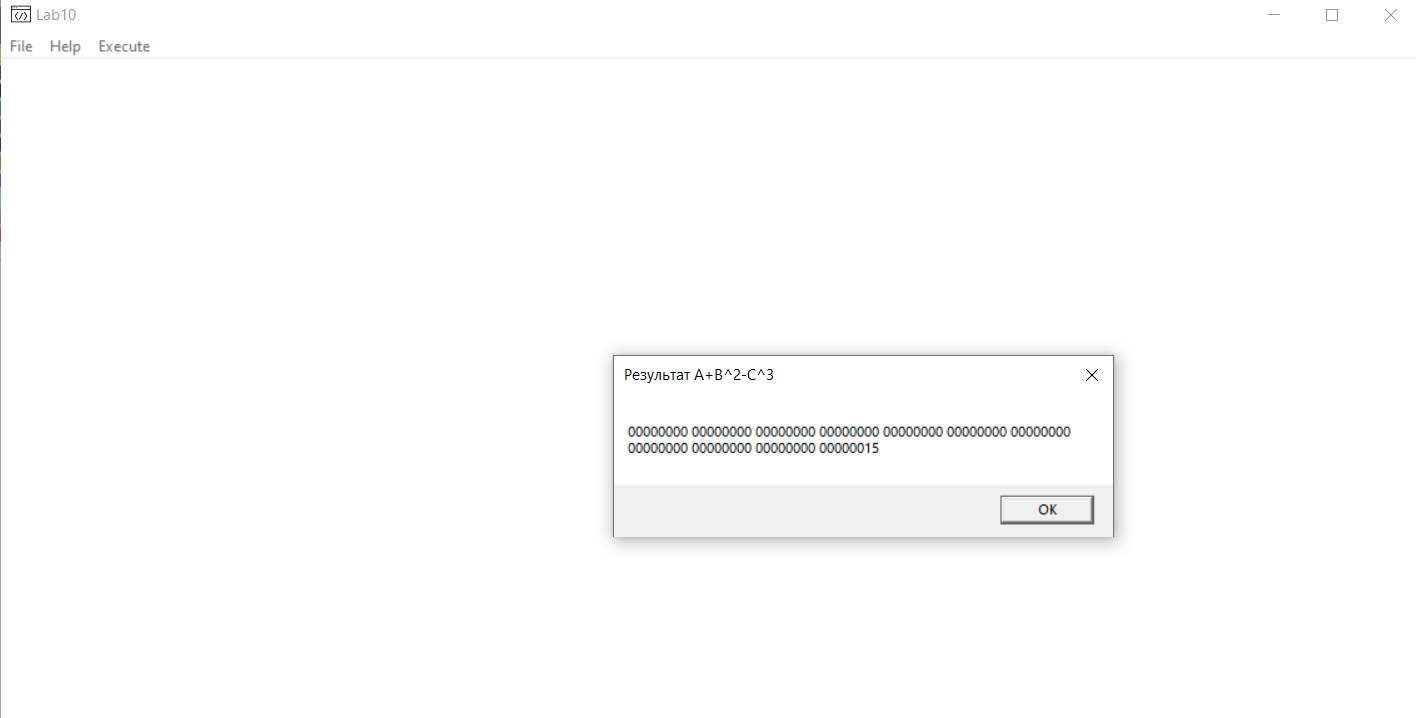
jne @cycle

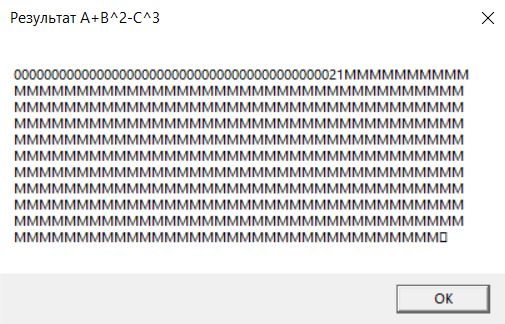
pop ebp

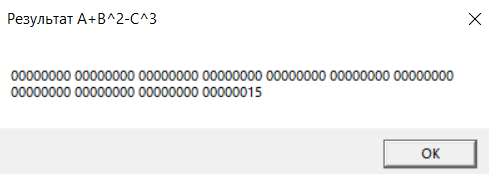
ret 12

COPY\_LONGOP endp

End

**Результати роботи програми**





**Аналіз результатів**

А = 4 = 4h

B = 5 = 5h

C = 2 = 2h

= 21 = 15h

Як бачимо результат співпадає зі значеннями з програми, як у десятковій, так і в шістнадцятковій системі. Отже, результати програми повністю збігаються з теоретичними даними. Програма працює вірно.

**Висновок**

В ході лабораторної роботи було розроблено навик створення програми на С++ з використанням модулів на асемблері. Було виконано підрахунок чисел за формулою . Кінцева мета роботи досягнута.

**Відповіді на контрольні запитання**

1. **Як налагодити підтримку асемблеру у проекті Visual C++?**

Викликаємо контекстне меню проекту. Натискаємо праву кнопку миші на назві проекту, переходимо у розділ «Зависимости сборки», потім «Настройки сборки» і там вмикаємо підтримку masm.

1. **Як у проекті вказати робочий набір символів multibyte замість Unicode?**

Відкриваємо властивості проекту у меню «Отладка». Потім вкладку «Свойства конфигурации» і «Дополнительно», а там рядок «Набор символов» та змінюємо «Использовать набор символов Юникода» на «Использовать многобайтовую кодировку».

1. **Що таке конвенція cdecl?**

Це конвенція виклику на C++, згідно з якою процедура не буде сама звільнювати стек від параметрів – це буде робитися після відпрацювання процедури та виходу з неї. Завдяки цьому дані процедури можливо використовувати у програмах на С++.

1. **Як відредагувати меню програми?**

Це можна зробити відкривши файл ресурсів (з розширенням .rc). Там потрібно обрати файл у папці «Menu» та в ньому змінювати все що забажаєте.

1. **Який машинний код додає компілятор при виклику процедури?**

Використовуючи конвенцію cdecl компілятор перед викликом процедури розмістить у стеку параметри, та після виклику здійснить звільнення стеку від них шляхом зміщення показника стеку на певну кількість значень.

1. **Чому аргумент функції Func(char \*dest), який мовою C++ має тип char\*, при програмуванні на асемблері повинен мати тип DWORD?**

У С++ char\* повертає адресу на перший елемент масиву і для доступу до масиву даних на асемблері нам також потрібна адреса першого елементу, тому ми використовуємо DWORD як розмір комірки для цієї адреси.