Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 10**

з дисципліни «Системне програмування-1» на тему

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

ВИКОНАЛА:

студентка ІІ курсу ФІОТ

групи ІО-64

Бровченко Анастасія

Залікова - 6403

ПЕРЕВІРИВ:

ст.вик. Порєв В. М.

Київ – 2018

**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.

**Варіант завдання:**

|  |  |  |
| --- | --- | --- |
| № варіанту | Вираження | Розрядність |
| 3 | А(В+1)(С+2) | 832 |

**Роздруківка тексту програми:**

**Lab10.cpp**

// lab10.cpp: определяет точку входа для приложения.

//

#include "stdafx.h"

#include "lab10.h"

#include "longop.h"

#include "module.h"

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

TCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

TCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

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

void MyWork1(HWND hWnd)

{

long oA[26] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

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

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

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

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

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

0x00000000, 0x00000000 };

long oB[26] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

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

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

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

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

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

0x00000000, 0xFFFFFFFF };

long o1[26] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

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

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

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

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

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

0x00000000, 0x00000001 };

long oC[52] = { 0x00000001, 0x00000000, 0x00000000, 0x00000000,

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

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

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

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

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

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

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

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

0x00000000, 0x00000000, 0x80000000, 0x00000000,

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

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

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

long o2[52] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

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

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

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

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

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

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

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

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

0x00000000, 0x00000000, 0x80000000, 0x00000000,

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

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

0x00000000, 0x00000000, 0x00000000, 0x00000002 };

long result1[52] = { 0x80000000, 0x00000000, 0x00000000, 0x00000000,

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

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

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

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

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

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

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

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

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

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

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

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

long result2[104] = { 0x00000000, 0x00000000, 0x00000000, 0x00000000,

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Add\_LONGOP(52,oB,oB,o1);

Add\_LONGOP(104,oC,oC,o2);

char TextBuf[3328];

Mul\_LONGOP(104, result1, oB, oA);

Mul\_LONGOP(208, result2, result1, oC);

StrHex\_MY(3328, result2, TextBuf);

MessageBox(hWnd, TextBuf, "Результат A(B+1)(C+2)", MB\_OK);

}

int APIENTRY \_tWinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPTSTR lpCmdLine,

int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: разместите код здесь.

MSG msg;

HACCEL hAccelTable;

// Инициализация глобальных строк

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

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

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

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

// Цикл основного сообщения:

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

{

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// НАЗНАЧЕНИЕ: регистрирует класс окна.

//

// КОММЕНТАРИИ:

//

// Эта функция и ее использование необходимы только в случае, если нужно, чтобы данный код

// был совместим с системами Win32, не имеющими функции RegisterClassEx'

// которая была добавлена в Windows 95. Вызов этой функции важен для того,

// чтобы приложение получило "качественные" мелкие значки и установило связь

// с 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);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// НАЗНАЧЕНИЕ: сохраняет обработку экземпляра и создает главное окно.

//

// КОММЕНТАРИИ:

//

// В данной функции дескриптор экземпляра сохраняется в глобальной переменной, а также

// создается и выводится на экран главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

HWND hWnd;

hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

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;

}

//

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

//

// НАЗНАЧЕНИЕ: обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработка меню приложения

// WM\_PAINT -Закрасить главное окно

// WM\_DESTROY - ввести сообщение о выходе и вернуться.

//

//

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);

// Разобрать выбор в меню:

switch (wmId)

{

case ID\_32771:

MyWork1(hWnd);

break;

case IDM\_ABOUT:

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

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

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

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

// TODO: добавьте любой код отрисовки...

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

// Обработчик сообщений для окна "О программе".

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

cote dd 0

x dd 1

n dd 0

num10 db 10

inner dd 0

num7 db 7

minn db 0

spacee db 3

three dd 3

.code

Mul\_LONGOP proc bytes:DWORD, dest:DWORD, pB:DWORD, pA:DWORD

mov esi, pA ;ESI = адреса множеного

mov ebx, pB ;EBX = адреса множника

mov edi, dest ;EDI = адреса результату

mov ecx, bytes ; кількість байтів

shr ecx, 2

mov cote, ecx

xor ecx, ecx

cycle:

push ebx

push ecx

push edi

shl ecx, 2

add ebx, ecx

add edi, ecx

shr ecx, 2

xor ecx, ecx

cycle\_p: ;для парних доданків

mov eax, [esi+4\*ecx]

mul dword ptr [ebx]

add [edi+4\*ecx], eax

adc [edi+4\*ecx+4], edx

adc byte ptr [edi+4\*ecx+8], 0

inc ecx

inc ecx

cmp ecx, cote

jl cycle\_p

xor ecx, ecx

inc ecx

cycle\_np: ;для непарних доданків

mov eax, [esi+4\*ecx]

mul dword ptr [ebx]

add [edi+4\*ecx], eax

adc [edi+4\*ecx+4], edx

adc byte ptr [edi+4\*ecx+8], 0

inc ecx

inc ecx

cmp ecx, cote

jl cycle\_np

pop edi

pop ecx

pop ebx

inc ecx

cmp ecx, cote

jl cycle

ret

Mul\_LONGOP endp

Add\_LONGOP proc bytes: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, bytes ; ECX = потрібна кількість повторень

shr ecx, 2

xor edx, edx

clc ; обнулює біт CF регістру EFLAGS

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

cmp edx, ecx

jl cycle

ret

Add\_LONGOP endp

Sub\_LONGOP proc bytes: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, bytes ; ECX = потрібна кількість повторень

shr ecx, 2

xor edx, edx

clc ; обнулює біт CF регістру EFLAGS

cycle:

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

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

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

inc edx

cmp edx, ecx

jl cycle

ret

Sub\_LONGOP endp

Div\_Column\_LONGOP proc

xor ebx, ebx

xor ecx, ecx

dec edx

cmp byte ptr[esi + edx], 0

jnz @cycleout

inc bl

@cycleout:

mov ch, byte ptr[esi + edx]

@cycleinner:

shl cl, 1

shl bh, 1

shl ch, 1

jnc @zero

inc bh

@zero:

cmp bh, num10

jc @less

inc cl

sub bh, num10

@less:

inc inner

cmp inner, 8

jnz @cycleinner

mov byte ptr[esi + edx], cl

mov inner, 0

sub edx, 1

jnc @cycleout

ret

Div\_Column\_LONGOP endp

Str\_Dec proc bits:DWORD, src:DWORD, dest:DWORD

;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)

mov edx, bits ;кількість бітів числа

shr edx, 3 ;кількість байтів числа

mov esi, src ;адреса числа

mov edi, dest ;адреса буфера результату

mov eax, edx

mul three

mov cl, byte ptr[esi + edx - 1]

and cl, 128

cmp cl, 128

jnz @plus

mov minn, 1

push edx

@minus:

not byte ptr[esi + edx - 1]

sub edx, 1

jnz @minus

inc byte ptr[esi + edx]

pop edx

@plus:

@cycle:

push edx

call Div\_Column\_LONGOP

pop edx

add bh, 48

mov byte ptr[edi + eax], bh

dec eax

cmp bl, 0

jz @cycle

dec edx

jnz @cycle

cmp minn, 1

jc @nomin

mov byte ptr[edi + eax + 1], 45

dec eax

@nomin:

inc eax

@space:

mov byte ptr[edi + eax], 32

sub eax, 1

jnc @space

ret

Str\_Dec endp

End

**module.asm**

.586

.model flat, c

.data

Nbit dd 0

counter dd 0

.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 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

call HexSymbol\_MY

mov byte ptr[ebx], al

mov al, dl

call HexSymbol\_MY

mov byte ptr[ebx+1], al

mov eax, ecx

cmp eax, 4

jle @next

dec eax

and eax, 3

cmp al, 0

jne @next

mov byte ptr[ebx+2], 32

inc ebx

@next:

add ebx, 2

dec ecx

jnz @cycle

mov byte ptr[ebx], 0

@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

StrDec proc bits:DWORD, src:DWORD, dest:DWORD

mov esi, dest

mov edi, src

mov ebx, bits

mov Nbit, ebx

shr ebx, 5 ; кількість 32-бітових елементів

dec ebx

mov ecx, ebx

@copy:

number dw 0

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

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

dec ecx

cmp ecx, 0

jge @copy

@cycle:

push ebx

push esi

push edi

push offset number

push 10

push Nbit

push offset quotient

push offset remainder

call Div10\_LONGOP

pop edi

pop esi

pop ebx

mov al, byte ptr[remainder]

add al, 48

mov byte ptr[esi], al

inc counter

mov ecx, counter

@lshift:

mov dl, byte ptr[esi + ecx - 1]

mov byte ptr[esi + ecx - 1], 48

mov byte ptr[esi + ecx], dl

dec ecx

cmp ecx, 0

jne @lshift

mov ecx, ebx

@swap:

quotient dw 0

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

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

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

dec ecx

cmp ecx, 0

jge @swap

remainder dw 0

mov dword ptr[remainder], 0

mov ecx, ebx

@check:

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

cmp eax, 0

jne @cycle

dec ecx

cmp ecx, 0

jge @check

ret

StrDec endp

Div10\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp+24] ; ділене

mov eax, [ebp+20] ; дільник

mov ebx, [ebp+16] ; розрядність діленого

mov edi, [ebp+12] ; частка

mov ecx, [ebp+8] ; остача

buf dd 0

mov buf, eax

shr ebx, 3

dec ebx

xor eax, eax

@cycle:

mov al, byte ptr[esi + ebx]

div byte ptr[buf]

mov byte ptr[edi + ebx], al

dec ebx

cmp ebx, 0

jge @cycle

mov byte ptr[ecx], ah

mov esp, ebp

pop ebp

ret 20

Div10\_LONGOP endp

end

**Висновок:**

У ході виконання лабораторної роботи було створено проект С++ та модуль до нього на асемблері. Програма працює коректно.