НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №3

з дісципліни **«**Системне програмування**»**

Виконав:

студент 2 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Перевірив:

Порєв В.М.

Київ 2014 р.

**Тема:** Створення модульних проектів на асемблері у середовищі Visual Studio

та вивчення форматів представлення чисел

**Мета:** Навчитися створювати модульні проекти на асемблері, а також

закріпити знання основних форматів представлення чисел у комп’ютері.

**Завдання:**

1. Створити у середовищі MS Visual Studio проект з ім’ям Lab3.

2. Написати вихідний текст програми згідно варіанту завдання. Вихідний текст

повинен бути у вигляді двох модулів на асемблері:

- головний модуль, у якому описується загальний хід виконання програми

від початку і до завершення. Цей модуль містить точку входу у програму,

впродовж роботи викликає процедури з інших модулів. Вихідний текст

головного модуля записати у файл main3.asm;

- другий модуль, який містить процедуру, яка викликається з головного

модуля. Цей модуль записати у файл module.asm.

3. Додати файли модулів у проект. 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 та вихідний текст.

**Варіант завдання:** N=13  
X = N+10 = 23; Y = 2×X = 46

**Результати:**

|  |  |  |  |
| --- | --- | --- | --- |
| Тип даних | Значення | Результати виконання програми | |
| шістнадцятиковий код | двійковий код |
| Ціле 8-бітове | 23 | 17 | 0001 0111 |
| -23 | E8 | 1110 1000 |
| Ціле 16-бітове | 23 | 0017 | 0000 0000 0001 0111 |
| -23 | FFE8 | 11111111 11101000 |
| Ціле 32-бітове | 23 | 00000017 | 10111 |
| -23 | FFFFFE8 | 1111 1111 1111 1111 1111 1110 1000 |
| Ціле 64-бітове | 23 | 00000000 00000017 | 10111 |
| -23 | FFFFFFF FFFFFE9 | 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1110 1001 |
| Число у 32-бітовому форматі | 23.0 | 41B80000 | 0100 0001 1011 1000 0000 0000 0000 0000 |
| -46.0 | C2380000 | 1100 0010 0011 1000 0000 0000 0000 0000 |
| 23.23 | 41B9D70A | 0100 0001 1011 1001 1101 0111 0000 1010 |
| Число у 64-бітовому форматі  з плаваючою точкою | 23.0 | 40370000 00000000 | 0100 0000 0011 0111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| -46.0 | C0470000 00000000 | 1100 0000 0100 0111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| 23.23 | 40373AE1 47AE147B | 0100 0000 0011 0111 0011 1010 1110 0001 0100 0111 1010 1110 0001 0100 0111 1011 |
| Число у 80-бітовому форматі  з плаваючою точкою | 23.0 | 4003 B8000000 00000000 | 0100 0000 0000 0011 1011 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| -46.0 | C004 B8000000 00000000 | 1100 0000 0000 0100 1011 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| 23.23 | 4003 B9D70A3D 70A3D70A | 0100 0000 0000 0011 1011 1001 1101 0111 0000 1010 0011 1101 0111 0000 1010 0011 |

Аналіз результатів:

32 біти 1 біт-знак, 2-9 експонента, 10-32 мантисса

64 біти 1 біт-знак, 2-12 експонента, 13-64 мантисса

80 біт 1 біт-знак, 2-16 експонента,17-ціла част., 18-80 мантисса

Програмний код:  
.586

.model flat, stdcall

option casemap :none

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

include \masm32\include\windows.inc

include module.inc

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\user32.lib

.const

.data

TextBuf db 64 dup(?)

Value1 db 00010111b

Value2 db 11101000b

Value3 dw 0000000000010111b

Value4 dw 1111111111101000b

Value5 dd 00000000000000000000000000010111b

Value6 dd 11111111111111111111111111101000b

Value7 dq 23

Value8 dq -23

Value9 dd 23.0

Value10 dd -46.0

Value11 dd 23.23

Value12 dq 23.0

Value13 dq -46.0

Value14 dq 23.23

Value15 dt 23.0

Value16 dt -46.0

Value17 dt 23.23

Caption db "Lab 3", 0

.code

main:

push offset TextBuf

push offset Value1

push 8

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value2

push 8

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value3

push 16

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value4

push 16

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value5

push 32

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value6

push 32

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value7

push 64

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value8

push 64

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value9

push 32

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value10

push 32

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value11

push 32

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value12

push 64

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value13

push 64

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value14

push 64

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value15

push 80

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value16

push 80

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset TextBuf

push offset Value17

push 80

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

invoke ExitProcess, 0

end main