НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №3
з дісципліни «Системне програмування»

Виконав:
студент 2 курсу
ФІОТ гр. ІО-32
Попенко Руслан

Перевірив:
Порєв В.М.

Київ 2015 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=24
X = N+10 = 34; Y = 2×X = 68

Результати:

Тип даних	Значення	Результати виконання програми	
		шістнадцятиковий код	двійковий код
Ціле 8-бітове	34	22	0010 0010
	-34	DE	1101 1110
Ціле 16-бітове	34	0022	0000000000100010
	-34	FFDE	1111111111011110
Ціле 32-бітове	34	00000022	00000000000000000000000000100010
	-34	FFFFFDE	11111111111111111111111111011110
Ціле 64-бітове	34	00000000 00000022	0000000000000000000000000000000000000000000000000000000000100010
	-34	FFFFFFF FFFFFDE	1111111111111111111111111111111111111111111111111111111111011110
Число у 32-	34.0	42080000	0100 0010 0000 1000 0000 0000

бітовому форматі з плаваючою точкою			0000 0000
	-68.0	C2880000	1100 0010 1000 1000 0000 0000 0000 0000
	34.34	42095C29	0100 0010 0000 1001 0101 1100 0010 1000
Число у 64-бітовому форматі з плаваючою точкою	34.0	40410000 00000000	0100 0000 0100 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
	-68.0	C0510000 00000000	1100 0000 0101 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
	34.34	40412B85 1EB851EC	0100 0000 0100 0001 0010 1011 1000 0101 0001 1110 1011 1000 0101 0001 1110 1011
Число у 80-бітовому форматі з плаваючою точкою	34.0	4004 88000000 00000000	0100 0000 0100 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
	-68.0	C005 88000000 00000000	1100 0000 0101 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
	34.34	4004 895C28F5 C28 F5C29	0100 0000 0100 0001 0010 1011 1000 0101 0001 1110 1011 1000 0101 0001 1110 1011

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 00100010b
Value2 db 11011110b
Value3 dw 0000000000100010b
Value4 dw 1111111111011110b
Value5 dd 00000000000000000000000000100010b
Value6 dd 11111111111111111111111111011110b
Value7 dq 34
Value8 dq -34
```

```
Value9 dd 34.0
Value10 dd -68.0
Value11 dd 34.34
Value12 dq 34.0
Value13 dq -68.0
Value14 dq 34.34
Value15 dt 34.0
Value16 dt -68.0
Value17 dt 34.34

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

MODULE.INC
EXTERN StrHex_MY : proc
```

Результат роботи