Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Системне програмування

Лабораторна робота №2

«Створення модульних проектів на Асемблері та вивчення форматів представлення чисел»

Виконав:
студент групи ІО-24
Довгань М. С.

Перевірив:
Порєв В. М.

Київ - 2024

Тема: Створення модульних проектів на Асемблері 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 Microsoft Visual Studio проект з ім'ям **Lab2**.

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 **main2.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 виконуваний 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.

Індивідуальний варіант завдання:

Номер варіанту (N) згідно списку студентів у журналі. Виконати завдання для числових значень X та Y, які обчислюються за формулами:

$$X = N + 10, \text{ отже: } X = 9 + 10 = 19,$$

$$Y = 2 \cdot X, \text{ отже: } Y = 2 \cdot 19 = 38.$$

Запрограмувати на асемблері вивід шістнадцяткових значень для всіх типів даних згідно таблиці. Надати таблицю, заповнену кодами-результатами.

Виконання завдання:

Роздруківка коду програми:

module.inc:

```
EXTERN StrHex_MY : proc
```

module.asm:

```
.586
.model flat, c

.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
      push ebp
      mov ebp, esp

      mov ecx, [ebp+8]  ;кількість бітів числа
      cmp ecx, 0
      jle @exitp
      shr ecx, 3              ;кількість байтів числа
      mov esi, [ebp+12] ;адреса числа
```

```
      mov ebx, [ebp+16] ;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:
      pop ebp
      ret 12
      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

end
```

main2.asm:

```
.386
.model flat, stdcall
option casemap : none

include module.inc

include \masm32\include\user32.inc
include \masm32\include\kernel32.inc

includelib \masm32\lib\user32.lib
includelib \masm32\lib\kernel32.lib

.const
	mainWindowTitle db "Лабораторна робота №2", 0
	mainWindowText db "Здоровенькі були!", 13, 10, 13, 10,
				"Лабораторну роботу виконав:", 13, 10,
				"студент групи ІО-24,", 13, 10,
				"Довгань М. С.", 0
	windowTask1 db "Цілий 8-бітовий тип", 0
	windowTask2 db "Цілий 16-бітовий тип", 0
	windowTask3 db "Цілий 32-бітовий тип", 0
	windowTask4 db "Цілий 64-бітовий тип", 0
	windowTask5 db "Плаваюча точка 32-бітовий тип", 0
	windowTask6 db "Плаваюча точка 64-бітовий тип", 0
	windowTask7 db "Плаваюча точка 80-бітовий тип", 0
```

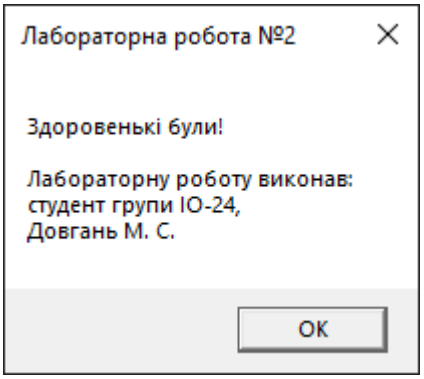
```
	lastWindowTitle db "Програма завершила роботу", 0
	lastWindowText db "Дякую за увагу!", 0

.data
	textRes db 64 dup(?)
	num1 db 19
	num2 db -19
	num3 dw 19
	num4 dw -19
	num5 dd 19
	num6 dd -19
	num7 dq 19
	num8 dq -19
	num9 dd 19.0
	num10 dd -38.0
	num11 dd 19.19
	num12 dq 19.0
	num13 dq -38.0
	num14 dq 19.19
	num15 dt 19.0
	num16 dt -38.0
	num17 dt 19.19

.code

main:
	invoke MessageBoxA, 0, ADDR mainWindowText, ADDR mainWindowTitle, 0

	push offset textRes
	push offset num1
	push 8
	call StrHex_MY
	invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask1, 0

	push offset textRes
	push offset num2
	push 8
	call StrHex_MY
	invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask1, 0
```

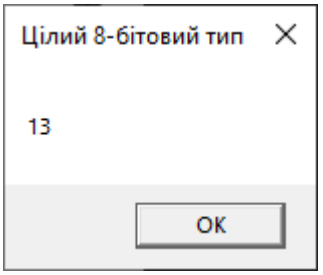
```
push offset textRes
push offset num3
push 16
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask2, 0

push offset textRes
push offset num4
push 16
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask2, 0

push offset textRes
push offset num5
push 32
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask3, 0

push offset textRes
push offset num6
push 32
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask3, 0

push offset textRes
push offset num7
push 64
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask4, 0

push offset textRes
push offset num8
push 64
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask4, 0

push offset textRes
push offset num9
```

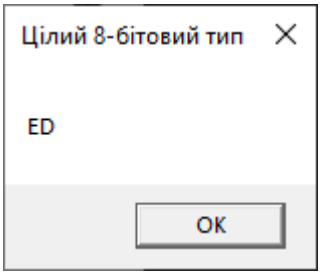
```
push 32
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask5, 0

push offset textRes
push offset num10
push 32
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask5, 0

push offset textRes
push offset num11
push 32
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask5, 0

push offset textRes
push offset num12
push 64
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask6, 0

push offset textRes
push offset num13
push 64
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask6, 0

push offset textRes
push offset num14
push 64
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask6, 0

push offset textRes
push offset num15
push 80
call StrHex_MY
invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask7, 0
```

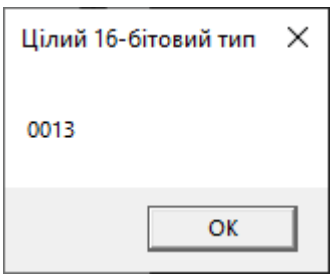
```
        push offset textRes
        push offset num16
        push 80
        call StrHex_MY
        invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask7, 0

        push offset textRes
        push offset num17
        push 80
        call StrHex_MY
        invoke MessageBoxA, 0, ADDR textRes, ADDR windowTask7, 0

        invoke MessageBoxA, 0, ADDR lastWindowText, ADDR lastWindowTitle, 0

        invoke ExitProcess, 0

end main
```

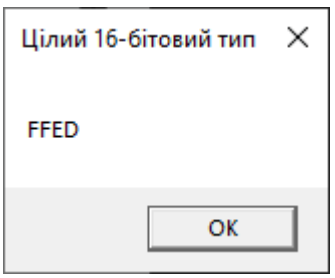
Результати виконання програми:

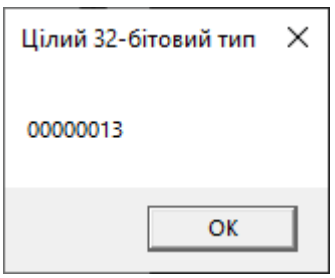


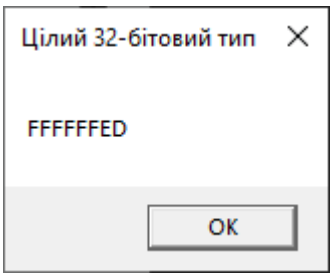


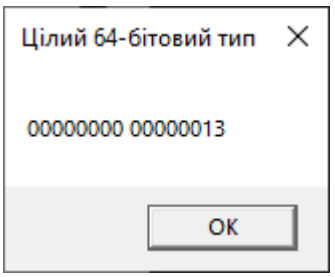


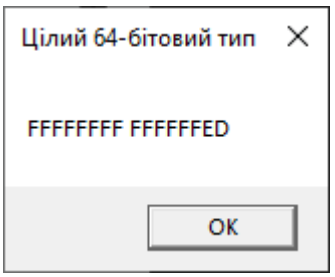


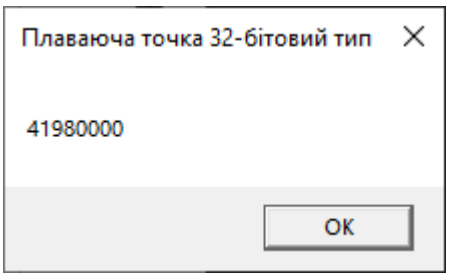


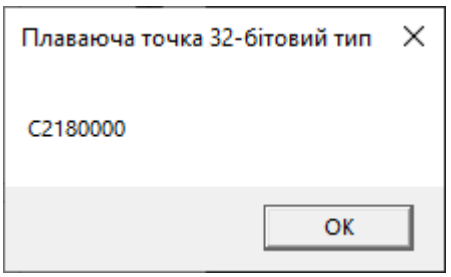


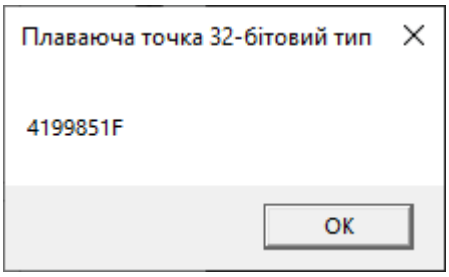

Цілий 32-бітовий тип
FFFFFFED
OK

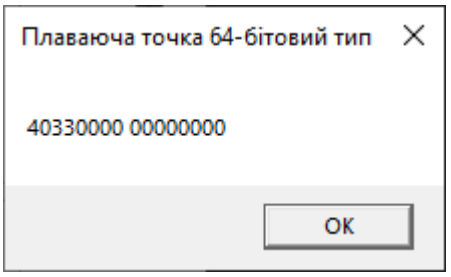

Цілий 64-бітовий тип
00000000 00000013
OK

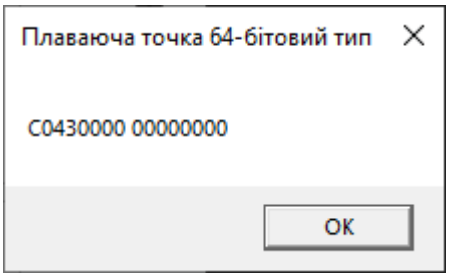

Цілий 64-бітовий тип
FFFFFFFF FFFFFFED
OK

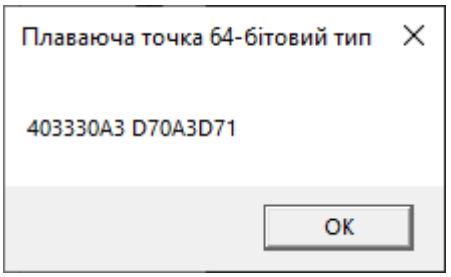

Плаваюча точка 32-бітовий тип
41980000
OK

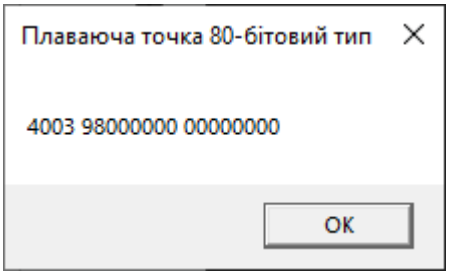

Плаваюча точка 32-бітовий тип
C2180000
OK

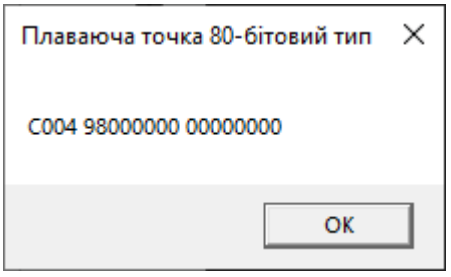

Плаваюча точка 32-бітовий тип
4199851F
OK

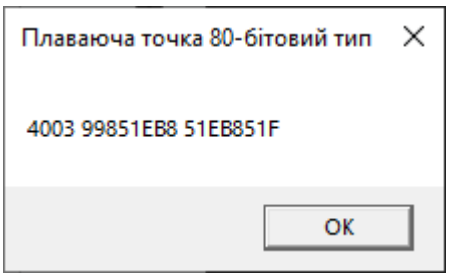

Плаваюча точка 64-бітовий тип
40330000 00000000
OK

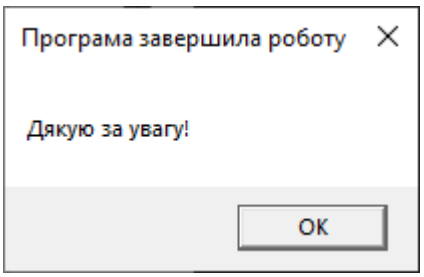

Плаваюча точка 64-бітовий тип
C0430000 00000000
OK


Плаваюча точка 64-бітовий тип
403330A3 D70A3D71
OK


Плаваюча точка 80-бітовий тип
4003 98000000 00000000
OK


Плаваюча точка 80-бітовий тип
C004 98000000 00000000
OK


Плаваюча точка 80-бітовий тип
4003 99851EB8 51EB851F
OK


Програма завершила роботу
Дякую за увагу!
OK

Аналіз виконання роботи:

Значення кодованих чисел

(Таблиця переведення з шістнадцяткового коду в двійковий)

Тип даних, які обробляє програма	Значення	Результати виконання програми	
		Шістнадцятковий код	Двійковий код
Цілий 8-бітовий	19	13	0001 0011
	-19	ED	1110 1101
Цілий 16-бітовий	19	0013	0000 0000 0001 0011
	-19	FFED	1111 1111 1110 1101
Цілий 32-бітовий	19	0000 0013	0000 0000 0000 0000 0000 0000 0001 0011
	-19	FFFF FFED	1111 1111 1111 1111 1111 1111 1110 1101
Цілий 64-бітовий	19	0000 0000 0000 0013	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0011
	-19	FFFF FFFF FFFF FFED	1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111

			1111 1111 1110 1101
32-бітовий із плаваючою точкою	19.0	4198 0000	0100 0001 1001 1000 0000 0000 0000 0000
	-38.0	C218 0000	1100 0010 0001 1000 0000 0000 0000 000
	19.19	4199 851F	0100 0001 1001 1001 1000 0101 0001 1111
64-бітовий із плаваючою точкою	19.0	4033 0000 0000 0000	0100 0000 0011 0011 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
	-38.0	C043 0000 0000 0000	1100 0000 0100 0011 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
	19.19	4033 30A3 D70A 3D71	0100 0000 0011 0011 0011 0000 1010 0011 1101 0111 0000 1010

			0011 1101 0111 0001
80-бітовий із плаваючою точкою	19.0	4003 9800 0000 0000 0000	0100 0000 0000 0011 1001 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
	-38.0	C004 9800 0000 0000 0000	1100 0000 0000 0100 1001 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
	19.19	4003 9985 1EB8 51EB 851F	0100 0000 0000 0011 1001 1001 1000 0101 0001 1110 1011 1000 0101 0001 1110 1011 1000 0101 0001 1111

Створена мною програма виконує завдання лабораторної роботи, відповідно до мого індивідуального варіанту завдання та обчислених значень. Обчислені мною значення на початку заносяться до програми у різних форматах - db, dw, dd, dq, dt, тобто, створюються перемінні розміром від одного байту (db - define byte), до десяти байтів (dt - define ten

bytes), а також два байти (dw - define word), чотири байти (dd - define double word) та вісім байтів (dq - define quad word). На початку програма видає користувачеві стартове вікно-привітання. Після цього програма обробляє значення і вони виводяться у формі шістнадцяткових кодів, кожен в окремому підписаному вікні для усіх типів даних згідно таблиці, наведеної вище. При виведенні усіх значень, останнім вікном є вікно, яке повідомляє користувача про те, що було виведено всі значення, і програма завершує свою роботу.

Висновок: під час виконання даної лабораторної роботи я створив свій перший модульний проект на мові програмування Асемблер та вивчив, використав і закріпив знання форматів представлення чисел.