Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 3

З дисципліни «Системне програмування»

ВИКОНАВ:

студент 2 курсу ФІОТ

групи ІО-41

Логвинчук А. І.

ПЕРЕВІРИВ:

ст. вик. Порєв В. М.

Київ – 2016

**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=20 X = N+10 = 30; Y = 2×X = 60

**Результати:**

|  |  |  |  |
| --- | --- | --- | --- |
| Тип даних | Значення | Результати виконання програми | |
| шістнадцятиковий код | двійковий код |
| Ціле 8-бітове | 30 | 1E | 0001 1110 |
| -30 | E2 | 0010 |
| Ціле 16-бітове | 30 | 001E | 0000 0000 0001 1110 |
| -30 | FFE2 | 1111 1111 1110 0010 |
| Ціле 32-бітове | 30 | 0000 001E | 0000 0000 0000 0000 0000 0000 0001 1110 |
| -30 | FFFF FFE2 | 1111 1111 1111 1111 1111 1111 1110 0010 |
| Ціле 64-бітове | 30 | 0000 0000  0000 0001E | 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 1110 |
| -30 | FFFF FFFF  FFFF FFFE2 | 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1110 1011 |
| Число у 32-бітовому форматі з плаваючою точкою | 30.0 | 41F0 0000 | 0100 0001 1111 0000 0000 0000 0000 0000 |
| -60.0 | C270 0000 | 1100 0010 0111 0000 0000 0000 0000 0000 |
| 30.30 | 41F2 6666 | 0100 0001 1111 0010 0110 0110 0110 0110 |
| Число у 64-бітовому форматі  з плаваючою точкою | 30.0 | 403E 0000  0000 0000 | 0100 0000 0011 1110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| -60.0 | C04E 0000  0000 0000 | 1100 0000 0100 0101 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| 30.30 | 403E 4CCC  CCCC CCCD | 0100 0000 0011 0101 0011 0101 1100 0010 1000 1111 0101 1100 0010 1000 1111 0110 |
| Число у 80-бітовому форматі  з плаваючою точкою | 30.30 | 4003 F000  0000 0000  0000 | 0100 0000 0000 0011 1010 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| -60.0 | C004 F000  0000 0000  0000 | 1100 0000 0000 0100 1010 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| 30.30 | 4003 F266  6666 6666  6666 | 0100 0000 0000 0011 1010 1001 1010 1110 0001 0100 0111 1010 1110 0001 0100 0111 1010 1110  0001 0100 |

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**

main3.asm:

.586

.model flat, stdcall

option casemap: none

include \DEV\masm32\include\kernel32.inc

include \DEV\masm32\include\user32.inc

include \DEV\masm32\include\windows.inc

include module.inc

includelib \DEV\masm32\lib\kernel32.lib

includelib \DEV\masm32\lib\userl32.lib

.data

Greeting db "Лаб. робота №3", 0

GreetingText db "Лабораторна робота №3", 13, 10,

"вик. Логвинчук А.І.", 13, 10,

"гр. ІО-41", 0

TextBuf db 64 dup(?)

BinBuf db 32 dup(32), 0

Value1 db 30

Value2 db -30

Value3 dw 30

Value4 dw -30

Value5 dd 30

Value6 dd -30

Value7 dq 30

Value8 dq -30

Value9 dd 30.0

Value10 dd -60.0

Value11 dd 30.30

Value12 dq 30.0

Value13 dq -60.0

Value14 dq 30.30

Value15 dt 30.0

Value16 dt -60.0

Value17 dt 30.30

CaptionBin db "BIN", 0

CaptionHex db "HEX", 0

Caption db "Lab3", 0

.code

main:

invoke MessageBoxA, 0, ADDR GreetingText, ADDR Greeting, MB\_ICONINFORMATION

push offset TextBuf

push offset Value1

push 8

call StrHex\_MY

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

push offset BinBuf

push offset Value1

push 8

call ToBinStr

invoke MessageBox, 0, ADDR BinBuf, ADDR CaptionBin, MB\_ICONINFORMATION

push offset TextBuf

push offset Value2

push 8

call StrHex\_MY

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

push offset BinBuf

push offset Value2

push 8

call ToBinStr

invoke MessageBox, 0, ADDR BinBuf, ADDR CaptionBin, MB\_ICONINFORMATION

push offset TextBuf

push offset Value3

push 16

call StrHex\_MY

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

push offset BinBuf

push offset Value3

push 16

call ToBinStr

invoke MessageBox, 0, ADDR BinBuf, ADDR CaptionBin, MB\_ICONINFORMATION

push offset TextBuf

push offset Value4

push 16

call StrHex\_MY

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

push offset BinBuf

push offset Value4

push 16

call ToBinStr

invoke MessageBox, 0, ADDR BinBuf, ADDR CaptionBin, MB\_ICONINFORMATION

push offset TextBuf

push offset Value5

push 32

call StrHex\_MY

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

push offset BinBuf

push offset Value5

push 32

call ToBinStr

invoke MessageBox, 0, ADDR BinBuf, ADDR CaptionBin, MB\_ICONINFORMATION

push offset TextBuf

push offset Value6

push 32

call StrHex\_MY

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

push offset BinBuf

push offset Value6

push 32

call ToBinStr

invoke MessageBox, 0, ADDR BinBuf, ADDR CaptionBin, MB\_ICONINFORMATION

push offset TextBuf

push offset Value7

push 64

call StrHex\_MY

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

push offset TextBuf

push offset Value8

push 64

call StrHex\_MY

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

push offset TextBuf

push offset Value9

push 32

call StrHex\_MY

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

push offset BinBuf

push offset Value9

push 32

call ToBinStr

invoke MessageBox, 0, ADDR BinBuf, ADDR CaptionBin, MB\_ICONINFORMATION

push offset TextBuf

push offset Value10

push 32

call StrHex\_MY

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

push offset BinBuf

push offset Value10

push 32

call ToBinStr

invoke MessageBox, 0, ADDR BinBuf, ADDR CaptionBin, MB\_ICONINFORMATION

push offset TextBuf

push offset Value11

push 32

call StrHex\_MY

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

push offset BinBuf

push offset Value11

push 32

call ToBinStr

invoke MessageBox, 0, ADDR BinBuf, ADDR CaptionBin, MB\_ICONINFORMATION

push offset TextBuf

push offset Value12

push 64

call StrHex\_MY

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

push offset TextBuf

push offset Value13

push 64

call StrHex\_MY

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

push offset TextBuf

push offset Value14

push 64

call StrHex\_MY

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

push offset TextBuf

push offset Value15

push 80

call StrHex\_MY

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

push offset TextBuf

push offset Value16

push 80

call StrHex\_MY

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

push offset TextBuf

push offset Value17

push 80

call StrHex\_MY

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

invoke ExitProcess, 0

end main

module.asm:

ToBinStr proc ; Перетворення у рядок двійкових цифр

push ebp

mov ebp, esp

mov ecx, [ebp+8] ; Кількість розрядів

mov ebx, ecx

cmp ecx, 0

jle @exit

mov esi, [ebp+12] ; Адреса числа

mov edi, [ebp+16] ; Адреса рядка

mov eax, [esi] ; Записуемо число в регістр еах

@loop:

mov dl, al

and dl, 1b ; Залишаємо тільки молодший біт

cmp dl, 0

je @zero

mov byte ptr[edi+ecx], 49 ; записуємо код одиниці в ASCII

jmp @next

@zero:

mov byte ptr[edi+ecx], 48 ; або нуля

@next:

shr eax, 1 ; зсуваємо число на 1 розряд вправо

dec ecx ; декремент лічильника

cmp ecx, 0

je @exit

jmp @loop

@exit:

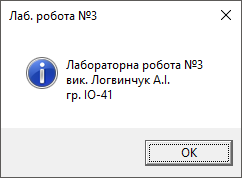
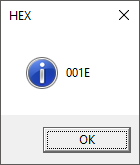
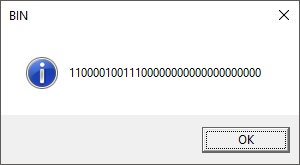
mov byte ptr[edi+ebx+1], 0

pop ebp

ret 12

ToBinStr end

**Результати роботи програми:**



**Висновок**

Числа в комп’ютерах представлені у двійковій системі числення. Таке кодування зумовлене апаратними вимогами.

Для роботи з цілими числами у архітектурі x86 використовуються формати розрядністю 8, 16, 32 та 64-біти. У кожному з цих форматів передбачена робота як з числами без знаку, так і з числами зі знаком.

Числа без знаку кодуються звичайним двійковим кодом. Для чисел зі знаком використовується додатковий код. Для зручності сприйняття машинних кодів людиною широко використовують шістнадцяткове кодування.