НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №3

з дисципліни «Системне програмування»

Виконала:

студентка 2 курсу

ФІОТ гр. ІО-32

Руденко Т.А.

Перевірив:

Порєв В.М.

Київ 2015

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.

Варіант завдання: N=25  
X = N+10 = 35; Y = 2×X = 70

Результати виконання:

|  |  |  |  |
| --- | --- | --- | --- |
| Тип даних | Значення | Результати виконання програми | |
| шістнадцятиковий код | двійковий код |
| Ціле 8-бітове | 35 | 23 | 0010 0011 |
| -35 | DC | 1101 1100 |
| Ціле 16-бітове | 35 | 0023 | 0000 0000 0010 0011 |
| -35 | FFDC | 1111 1111 1101 1100 |
| Ціле 32-бітове | 35 | 00000023 | 0000 0000 0000 0000 0000 0010 0011 |
| -35 | FFFFFDC | 1111 1111 1111 1111 1111 1101 1100 |
| Ціле 64-бітове | 35 | 00000000 00000023 | 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010 0011 |
| -35 | FFFFFFF FFFFFDD | 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1101 1100 |
| Число у 32-бітовому форматі | 35.0 | 420C0000 | 0100 0010 0000 1100 0000 0000 0000 0000 |
| -70.0 | C28C0000 | 1100 0010 1000 1100 0000 0000 0000 0000 |
| 35. 35 | 420D6666 | 0100 0010 0000 1101 0110 0110 0110 0110 |
| Число у 64-бітовому форматі  з плаваючою точкою | 35.0 | 40418000 00000000 | 0100 0000 0100 0001 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| -70.0 | C0518000 00000000 | 0101 0001 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| 35. 35 | 4041ACCCCCCCCCCD | 0100 0000 0100 0001 1010 1100 1100 1100 1100 1100 1100 1100 1100 1100 1100 1101 |
| Число у 80-бітовому форматі  з плаваючою точкою | 35.0 | 4004 8C000000 00000000 | 0100 0000 0000 0100 1000 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| -70.0 | C005 8C000000 00000000 | 1100 0000 0000 0101 1000 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| 35. 35 | 4004 8D66666666666666 | 0100 0000 0000 0100 1000 1101 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 |

Аналіз результатів:

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

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

80 біт 1 біт-знак, 2-16 експонента,17-ціла частина, 18-80 мантиса

Текст програми:

Main3.asm

.586

.model flat, stdcall

option casemap :none

include D:\\masm32\include\kernel32.inc

include D:\\masm32\include\user32.inc

include \masm32\include\windows.inc

include module3.inc

includelib D:\\masm32\lib\kernel32.lib

includelib D:\\masm32\lib\user32.lib

.data

TextBuf db 64 dup(?)

Value1 db 00100011b

Value2 db 11011100b

Value3 dw 0000000000100011b

Value4 dw 1111111111011100b

Value5 dd 00000000000000000000000000100011b

Value6 dd 11111111111111111111111111011100b

Value7 dq 35

Value8 dq -35

Value9 dd 35.0

Value10 dd -70.0

Value11 dd 35.35

Value12 dq 35.0

Value13 dq -70.0

Value14 dq 35.35

Value15 dt 35.0

Value16 dt -70.0

Value17 dt 35.35

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

module3.asm

.586

.model flat, c

.code

;процедура StrHex\_MY записує текст шістнадцяткового 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

Висновок:

Під час виконання даної лабораторної роботи ми навчилися створювати модульні проекти на асемблері, і повторили основні формати представлення чисел у комп'ютері.