Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 3**

з дисципліни «Системне програмування-1» на тему

«Створення модульних проектів на асемблері у середовищі Visual Studio та вивчення форматів представлення чисел»

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-82

Шендріков Євгеній

Залікова - 8227

ПЕРЕВІРИВ:

ст.вик. Порєв В. М.

Київ – 2020

**ЛАБОРАТОРНА РОБОТА №3**

*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.

**I. 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***: 25.

**Х =** 25 + 10 = **35**

**Y =** X \* 2 = 35 \* 2 = **70**

**ІІ. Код програми**

**main3.asm**

.586

.model flat, stdcall

option casemap : none; розрізнювати великі та маленькі букви

include \masm32\include\windows.inc

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

include module.inc

includelib \lib\kernel32.lib

includelib \lib\user32.lib

.data

TextBuf db 64 dup(? )

Caption db "Лабораторна робота №3", 0

Text db "Hello!", 10, 13, "Автор: Шендріков Євгеній, ІО-82", 0

value1 db 35; ціле 8 - бітове

value2 db - 35; ціле 8 - бітове

value3 dw 35; ціле 16 - бітове

value4 dw - 35; ціле 16 - бітове

value5 dd 35; ціле 32 - бітове

value6 dd - 35; ціле 32 - бітове

value7 dq 35; ціле 64 - бітове

value8 dq - 35; ціле 64 - бітове

value9 dd 35.0; Число у 32 - бітовому форматі з плаваючою точкою

value10 dd - 35.0; Число у 32 - бітовому форматі з плаваючою точкою

value11 dd 35.35; Число у 32 - бітовому форматі з плаваючою точкою

value12 dq 35.0; Число у 64 - бітовому форматі з плаваючою точкою

value13 dq - 70.0; Число у 64 - бітовому форматі з плаваючою точкою

value14 dq 35.35; Число у 64 - бітовому форматі з плаваючою точкою

value15 dt 35.0; Число у 80 - бітовому форматі з плаваючою точкою

value16 dt - 70.0; Число у 80 - бітовому форматі з плаваючою точкою

value17 dt 35.35; Число у 80 - бітовому форматі з плаваючою точкою

.code

main :

invoke MessageBoxA, 0, ADDR Text, ADDR Caption, 0

push offset TextBuf

push offset value1

push 8

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value2

push 8

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value3

push 16

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value4

push 16

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value5

push 32

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value6

push 32

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value7

push 64

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value8

push 64

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value9

push 32

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value10

push 32

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value11

push 32

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value12

push 64

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value13

push 64

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value14

push 64

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value15

push 80

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value16

push 80

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

push offset TextBuf

push offset value17

push 80

call StrHex\_MY

invoke MessageBoxA, 0, ADDR TextBuf, ADDR Caption, 0

invoke ExitProcess, 0

end main

**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

; ця процедура записує 8 символів HEX коду числа

; перший параметр - 32 - бітове число

; другий параметр - адреса буфера тексту

DwordToStrHex proc

push ebp

mov ebp, esp

mov ebx, [ebp + 8]; другий параметр

mov edx, [ebp + 12]; перший параметр

xor eax, eax

mov edi, 7

@next:

mov al, dl

and al, 0Fh; виділяємо одну шістнадцяткову цифру

add ax, 48; так можна тільки для цифр 0 - 9

cmp ax, 58

jl @store

add ax, 7; для цифр A, B, C, D, E, F

@store:

mov[ebx + edi], al

shr edx, 4

dec edi

cmp edi, 0

jge @next

pop ebp

ret 8

DwordToStrHex endp

End

**module.inc**

EXTERN StrHex\_MY : proc

EXTERN DwordToStrHex : proc

**III. Результат**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Типи даних*** | ***Зн-я*** | ***Шістнадцятковий код*** | ***Двійковий код*** |
| Ціле 8-бітове | 35 | 23 |  |
| -35 | DD |  |
| Ціле 16-бітове | 35 | 0023 | 0000 0000 0010 0011 |
| -35 | FFDD | 1111 1111 1101 1101 |
| Ціле 32-бітове | 35 | 0000 0023 | 0000 0000 0000 0000 0000 0000 0010 0011 |
| -35 | FFFF FFDD | 1111 1111 1111 1111 1111 1111 1101 1101 |
| Ціле 64-бітове | 35 | 0000 0000 0000 0023 | 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0010 0011 |
| -35 | FFFF FFFF FFFF FFDD | 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1101 1101 |
| Число у 32-бітовому форматі з плаваючою точкою | 35.0 | 420C 0000 |  |
| -70.0 | C20C 0000 |  |
| 35.35 | 420D 6666 |  |
| Число у 64-бітовому форматі з плаваючою точкою | 35.0 | 4041 8000 0000 0000 |  |
| -70.0 | С051 8000 0000 0000 |  |
| 35.35 | 4041 ACCC CCCC CCCD |  |
| Число у 80-бітовому форматі з плаваючою точкою | 35.0 | 4004 8C00 0000 0000 0000 | 0100 0000 0000 0100 1000 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| -70.0 | C005 8C00 0000 0000 0000 | 1100 0000 0000 0101 1000 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| 35.35 | 4004 8D66 6666 6666 6666 |  |

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

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

***80 бітів****: 1 біт - знак, 2-16 біти - експонента, 17 біт- ціла частина, 18-80 біти - мантиса*

**ІV. Висновок**

У ході виконання лабораторної роботи було закріплено на практиці навички створення модульних проектів у середовищі Microsoft Visual Studio 2017 та застосовано знання про представлення чисел у комп’ютері.