МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Лабораторна робота №2

з предмету “Системне програмування”

Виконав:

студент 2-го курсу ФІОТ

групи ІО-41

Смішний Дмитро Миколайович

Залікова - **4126**

Варіант – **25**

ПЕРЕВІРИВ:

Порєв Віктор Миколайович

Київ – 2016

Лабораторна робота №2

Знайомство із середовищем розробки програм Microsoft Visual Studio

Мета: Отримати перші навички роботи з Microsoft Visual Studio для

створення програм, написаних мовою асемблера, а також вивчити команди

MOV та CPUID.

Завдання:

1. Створити у середовищі MS Visual Studio проект з ім’ям Lab2. Встановити

необхідні параметри проекту – опції середовища розробки програм.

2. Написати вихідний текст програми на асемблері, додати файл вихідного

тексту у проект. Зміст вихідного тексту згідно з варіантом завдання.

3. Скомпілювати вихідний текст і отримати виконуючий файл програми.

4. Перевірити роботу програми. Налагодити програму.

5. Отримати дизасембльований текст машинного коду і проаналізувати його.

Усім студентам необхідно запрограмувати:

- початкове діалогове вікно-вітання від автора програми;

- виконання команди CPUID з параметрами 0, 1, 2 а також 80000000h,

80000001h, 80000002h, 80000003h, 80000004h, 80000005h та 80000008h.

Кожний результат виконання CPUID команди потрібно виводити у окремому

діалоговому вікні. Якщо результати CPUID утворюють текстові дані, то

виводити їх як рядки тексту.

Отримати дизасембльований код і проаналізувати його.

Пояснити значення N-го біту кожного результату команди CPUID, де N –

номер студента у списку у журналі. Для пояснення використати документ

"Intel® 64 and IA-32 Architectures Software Developer’s Manual Volume 2A:

Instruction Set Reference", доступний на сайті фірми Intel.

Роздруківка тексту програми

.586

.model flat, stdcall

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\user32.lib

.data

CaptionM db "Я програма на асемблері",0

TextM db "Здоровеньки були!", 10, 13, "Автор: Смішний Денис", 0

res dd 256 dup(0)

Text db 'EAX=xxxxxxxx',13,10,

'EBX=xxxxxxxx',13,10,

'ECX=xxxxxxxx',13,10,

'EDX=xxxxxxxx',0

Caption0 db "Результат CPUID 0",0

Caption1 db "Результат CPUID 1",0

Caption2 db "Результат CPUID 2",0

Caption00 db "Результат CPUID 80000000h",0

Caption01 db "Результат CPUID 80000001h",0

Caption02 db "Результат CPUID 80000002h",0

Caption03 db "Результат CPUID 80000003h",0

Caption04 db "Результат CPUID 80000004h",0

Caption05 db "Результат CPUID 80000005h",0

Caption08 db "Результат CPUID 80000008h",0

Vendor db 16 dup(0)

CaptionVendor db "CPUID 0 Vendor",0

.code

;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

main:

invoke MessageBoxA, 0, ADDR TextM, ADDR CaptionM, 0

mov eax, 0

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

invoke MessageBoxA, 0, ADDR Text, ADDR Caption0, 0

mov eax, 1

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

invoke MessageBoxA, 0, ADDR Text, ADDR Caption1, 0

mov eax, 2

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

invoke MessageBoxA, 0, ADDR Text, ADDR Caption2, 0

mov eax, 80000000h

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

invoke MessageBoxA, 0, ADDR Text, ADDR Caption00, 0

mov eax, 80000001h

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

invoke MessageBoxA, 0, ADDR Text, ADDR Caption01, 0

mov eax, 80000002h

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

invoke MessageBoxA, 0, ADDR Text, ADDR Caption02, 0

mov eax, 80000003h

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

invoke MessageBoxA, 0, ADDR Text, ADDR Caption03, 0

mov eax, 80000004h

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

invoke MessageBoxA, 0, ADDR Text, ADDR Caption04, 0

mov eax, 80000005h

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

invoke MessageBoxA, 0, ADDR Text, ADDR Caption05, 0

mov eax, 80000008h

cpuid

mov dword ptr[res], eax

mov dword ptr[res+4], ebx

mov dword ptr[res+8], ecx

mov dword ptr[res+12], edx

push [res] ;значення регістру EAX з масиву res

push offset [Text+4] ;адреса, куди записуються 8 символів

call DwordToStrHex

push [res+4] ;значення регістру EBX з масиву res

push offset [Text+18]

call DwordToStrHex

push [res+8] ;значення регістру ECX з масиву res

push offset [Text+32]

call DwordToStrHex

push [res+12] ;значення регістру EDX з масиву res

push offset [Text+46]

call DwordToStrHex

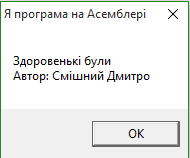
invoke MessageBoxA, 0, ADDR Text, ADDR Caption08, 0

invoke ExitProcess, 0

end main

Роздруківка результатів виконання програми





Аналіз результатів

Дана лабораторна робота є прикладом програмування на мові Асемблер, використовуючи Microsoft Visual Studio та MASM. Інсталяція IDE пройшла успішно, хоч і при лінкуванні програмного коду і виникали конфлікти з антивірусом Avira з помилкою LNK1104.

Написаний програмний код скомпілювався в результаті були отримані діалогові вікна з інформацією про регістри.

Висновки

Було встановлено Microsoft Visual Studio 2015 та вивчено інтерфейс даного програмного забезпечення. Написана програма, яка успішно запустилась та відобразила інформацію про стани регістрів та процесор комп’ютера. Освоєно команди MOV та CPUID. Дизасембльовано початковий код, що дозволило переглянути порядок виконання програми у пам’яті.