**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

**Лабораторна робота №2**

з дисципліни

«Системне програмування»

на тему

«Дослідження структури програм формату COM»

Виконав:

Перевірив:

студент групи ІП-93

Павлов Валерій Георгійович

Домінський Валентин Олексійович

номер залікової книжки: 9311

номер у списку: 9

Київ 2021

**Мета:**

Вивчення прийомів написання, компіляції і відладки програм формату COM в середовищі Masm32. Здобуття навичок читання лістингу і розшифрування кодів команд

**Порядок виконання роботи:**

1. Вивчити структуру програм формату COM і застосування переривань BIOS та MS-DOS при роботі з консоллю
2. Розробити програму на мові Асемблер, за допомогою якої на екран системної консолі по введеному паролю виводяться персональні дані студента – ПІБ, дата народження, номер залікової книжки тощо кожне з нового рядка з попереднім очищенням вікна системної консолі. Для правильного відображення при виведенні символів кирилиці підключити відповідну кодову сторінку

; Processors

.model **tiny**

.386

; Data Segment

.data

StartingText DB "Введiть пароль. Попереджаю, що у Вас є лише 4 спроби: $"**,** 10

FailureText DB "Пароль невiрний. Спробуйте ще раз. К-сть спроб, яка залишилася = $"**,** 10

StringFromUser DB 128 dup**(**128**)**

; We can write password in two ways:

Password DB "Dominskyi"

; And another one is:

; Password DB 31h 32h 33h

PasswordCount **=** **$-**Password

; Text To Show

InformationText DB "ПIБ = Домiнський Валентин Олексiйович"**,** 10**,**

"Дата Народження = 22.02.2002"**,** 10**,**

"Номер Залiковки книжки = 9311 $"**,** 0

; Code Segment

.code

org 100h ; this is offset for com programs:

; It defines where the machine code (translated

; assembly program) is to place in memory.

; As for org 100h this deals with 80x86 COM

; program format (COMMAND) which consist

; of only one segment of max. 64k bytes.

; 100h says that the machine code starts from

; address (offset) 100h in this segment.

; For com format the offset is always 100h

.startup ; Generates program start-up code

InvitePoint**:** ; Starting Code

; Clear the screen

**mov** **ax,** 0600h

**mov** **bh,** 7h

**mov** **cx,** 0000h

**mov** **dx,** 184fh

**int** 10h

; Set the position of cursor

**mov** **ah,** 02h

**mov** **bh,** 00h

**mov** **dl,** 00h

**mov** **dh,** 00h

**int** 10h

; Display The Text

**mov** **ah,** 9h

**mov** **dx,** offset StartingText

**int** 21h

**mov** **bx,** 03h ; counter for tries

**jmp** InputOfTheUser ; Unconditional jump

; Responsible For Input

InputOfTheUser**:**

**mov** **ah,** 0Ah

**mov** **dx,** offset StringFromUser

**int** 21h

**mov** **ax,** PasswordCount

**cmp** **al,** StringFromUser**+**1 ; Compare

**jne** WrongPasswordByUser ; Jump Not Equal

**mov** **si,** offset Password

**mov** **di,** offset StringFromUser**+**2 ; low-order 16 bits of 32-bit registers

**mov** **cl,**PasswordCount ; counter register

; Responsible For Checking, if password and input string are the same

IsPasswordCorrect**:**

**lodsb** ; loads 1 byte into the AL register

**mov** **bh,** **byte** ptr **[di]**

**cmp** **al,** **bh** ; Compare

; ptr = The first operator forces the expression to be treated as having

; the specified type. The second operator specifies a pointer to type

**je** LoopItself ; Jump Equal

**jmp** WrongPasswordByUser ; Unconditional jump

LoopItself**:**

**inc** **di** ; incrementing

**loop** IsPasswordCorrect

; Responsible For Correct Input

CorrectPasswordByUser**:**

; Clear the screen

**mov** **ax,** 0600h ; register si 32-bit general-purpose register, used for temporary data storage and memory access

**mov** **bh,** 7h ; register represent the high-order 8 bits of the corresponding register

**mov** **cx,** 0000h

**mov** **dx,** 184fh

**int** 10h

; Set the position of cursor

**mov** **ah,** 02h

**mov** **bh,** 00h

**mov** **dl,** 00h

**mov** **dh,** 00h

**int** 10h

**mov** **ah,** 09h

**mov** **dx,** offset InformationText

**int** 21h

**jmp** ExitCode ; Unconditional jump

; Responsible For Wrong Input

WrongPasswordByUser**:**

; Clear the screen

**mov** **ax,** 0600h

**mov** **bh,** 7h

**mov** **cx,** 0000h

**mov** **dx,** 184fh

**int** 10h

; Set the position of cursor

**mov** **ah,** 02h

**mov** **bh,** 00h

**mov** **dl,** 00h

**mov** **dh,** 00h

**int** 10h

; display text

**mov** **ah,** 09h

**mov** **dx,** offset FailureText

**int** 21h

**mov** **ah,** 02h ; Display Counter

**mov** **dl,** **bl**

**add** **dl,** "0" ; Integer to single-digit ASCII character

**int** 21h

**mov** **dl,** 0ah ; New Line

**int** 21h

; counting tries

**add** **bx,** **-**01h ; decrementing

**cmp** **bx,** **-**01h ; negative possible tries

**je** ExitCode ; Jump Equal

**jmp** InputOfTheUser ; Unconditional jump

; Responsible For Exit

ExitCode**:**

; For exiting program We can use this code...

;mov ah, 4Ch

;mov al, 00h

;int 21h

; ... or this one

.exit

end

1. Вивчити опції компілятора і лінковщика і сформувати BAT-файл, в якому передбачити завдання назви вихідного файлу .asm, як параметра. Шлях до файлу має бути визначений в результаті сканування логічного диску і служити для вказівки розміщення відповідних йому об'єктного і виконуваного файлів

:: Creating variable

**set** NameOfTheFileAsParametr**=%1**

:: Creating COM File

**start** D:\masm32\bin\ml /Bl D:\masm32\bin\link16.exe **%NameOfTheFileAsParametr%**

:: I'm using DOSBox for emulating DOS. That's why I launch this program there

:: Launching COM File

:: mount c %cd% - connect physical folders and drives to virtual drives inside DOSBox

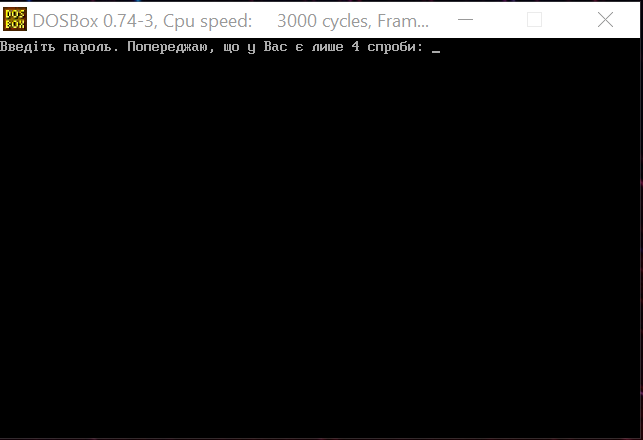
:: "keyb none 866" - to connect the appropriate language

:: %NameOfTheFileAsParametr:~0,-4%.com - launch %1.com file

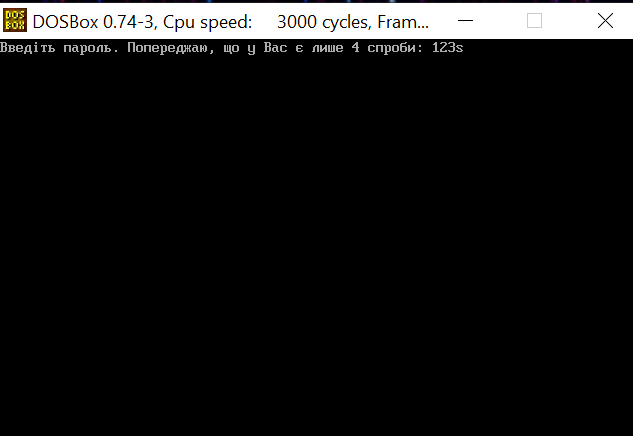
**start** D:\masm32\DosBox\DOSBox-0.74-3\DOSBox.exe -c "mount c **%cd%** " -c c: -c "keyb none 866" -c "**%NameOfTheFileAsParametr:~0,-4%**.**com**"

1. Виконати компіляцію розробленого файлу у формат COM
2. Перевірити роботу програми шляхом введення як правильного, так і невірного паролів.

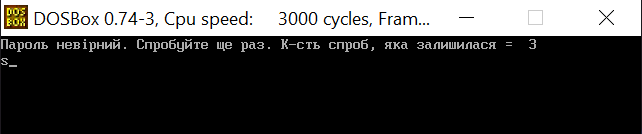
Початкове вікно:



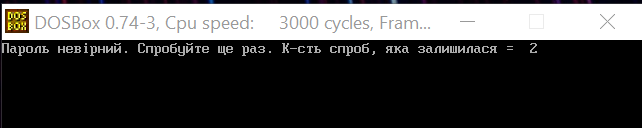
Введення неправильного паролю:

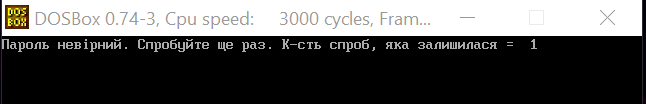


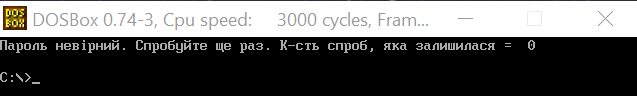
Вивід при такому паролі:



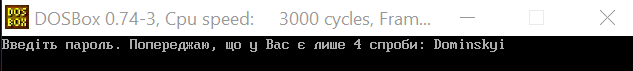
Вивід при введенні неправильного паролю кілька разів:



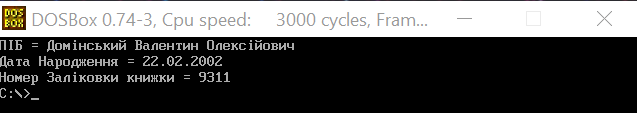




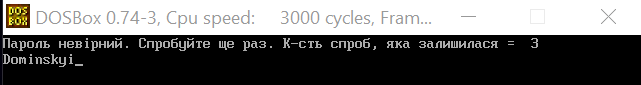
Введення правильно паролю з першого разу:



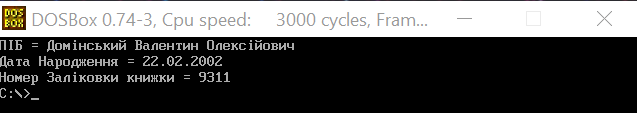
Вивід:



Введення правильно паролю не з першого разу:

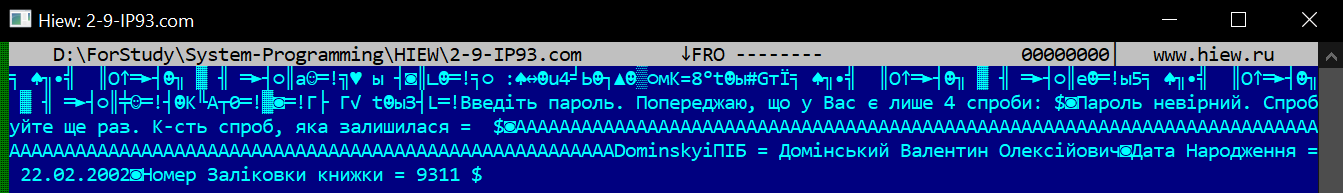


Вивід:

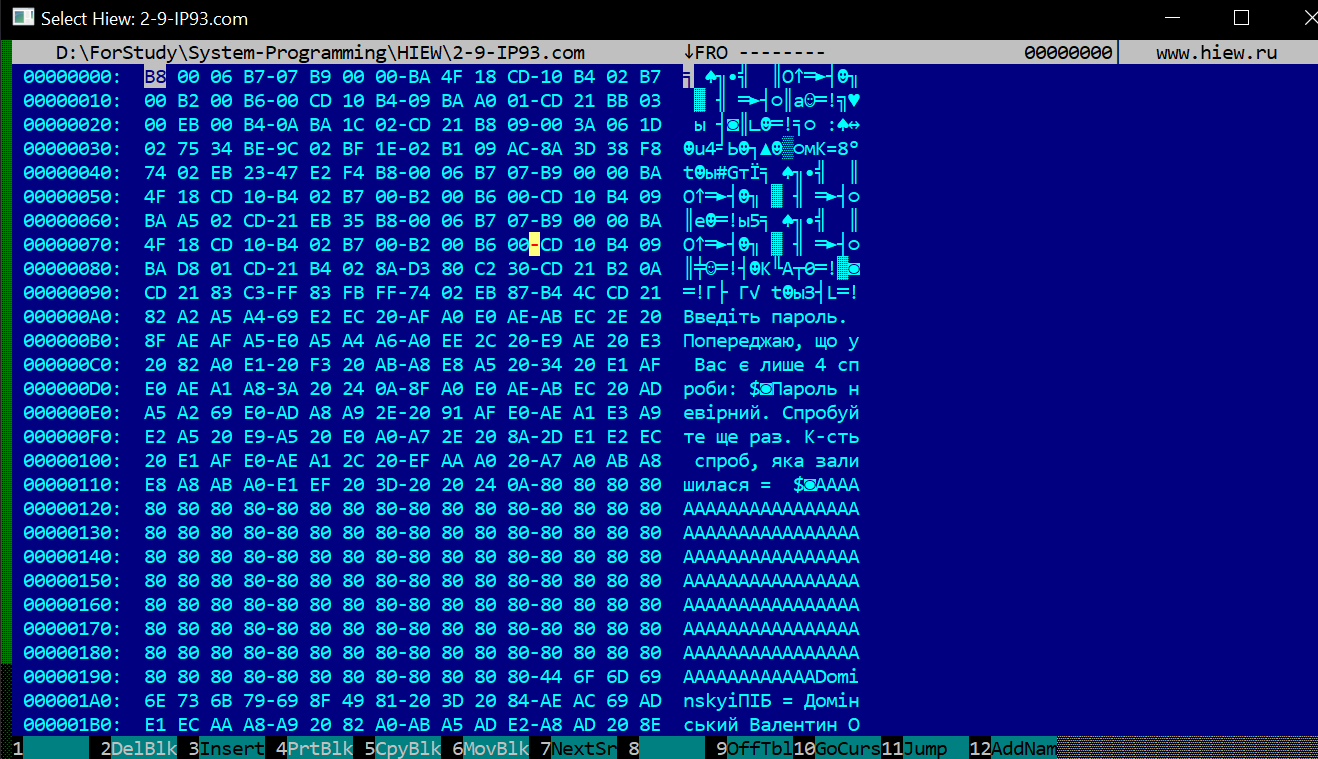


1. Отриманий виконуваний файл дослідити за допомогою програми HEX-редактору HIEW32 або HIEW. У останньому випадку для запуску програми сформувати ярлик, де виконати налаштування параметрів сумісності з використовуваною операційною системою
2. Перемикаючи послідовно режими перегляду (Text – Hex – Decode), зняти три відповідних скріншоти програми і привести їх в звіті по лабораторній роботі.

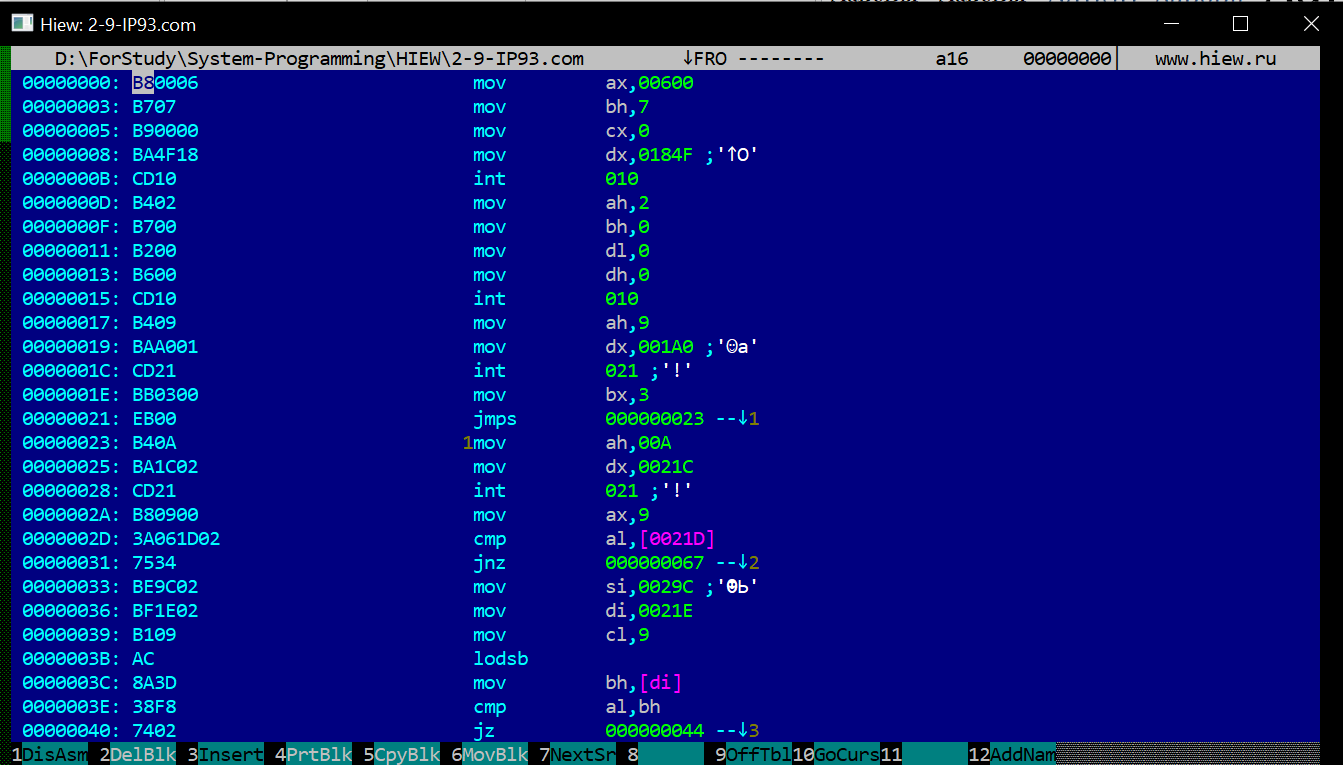
Text:



Hex:

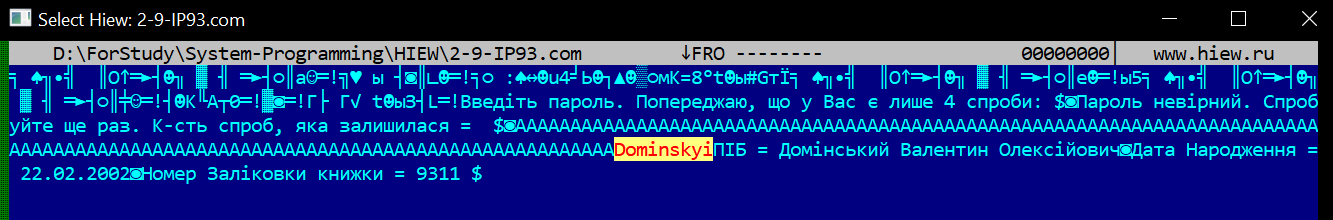


Decode:

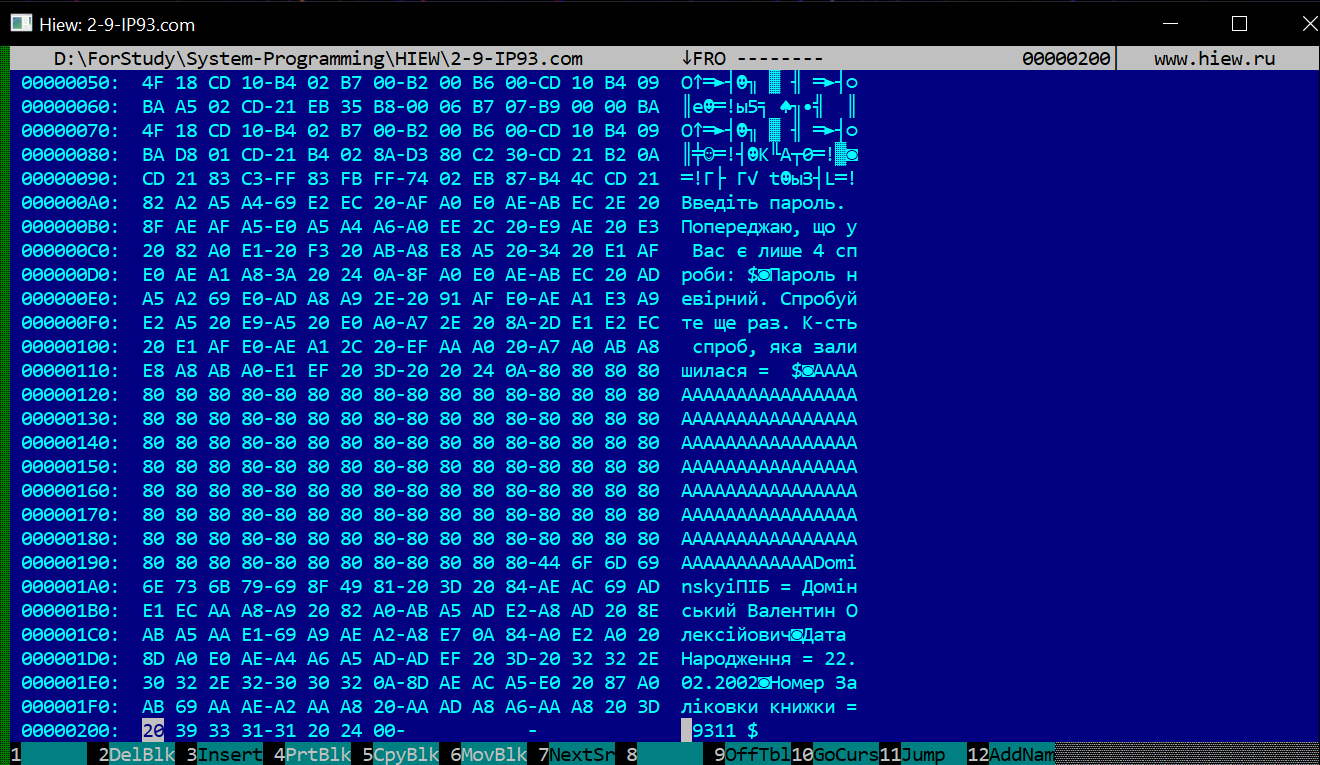


1. Переконатися, що текст оригінала пароля, який міститься в тексті програми, може бути легко виявлений за допомогою HEX-редактора

Пароль у Text:

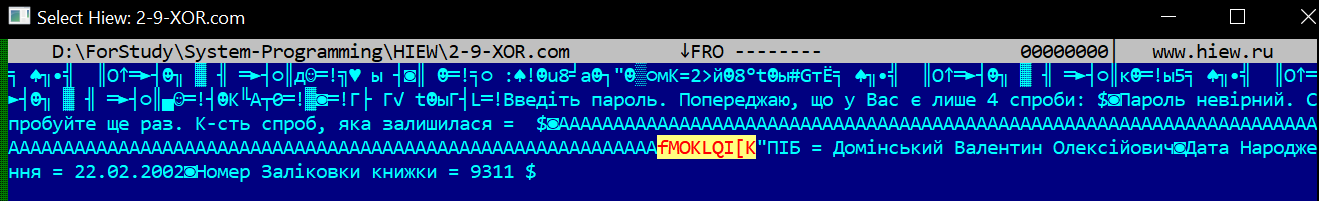


Пароль у Hex:

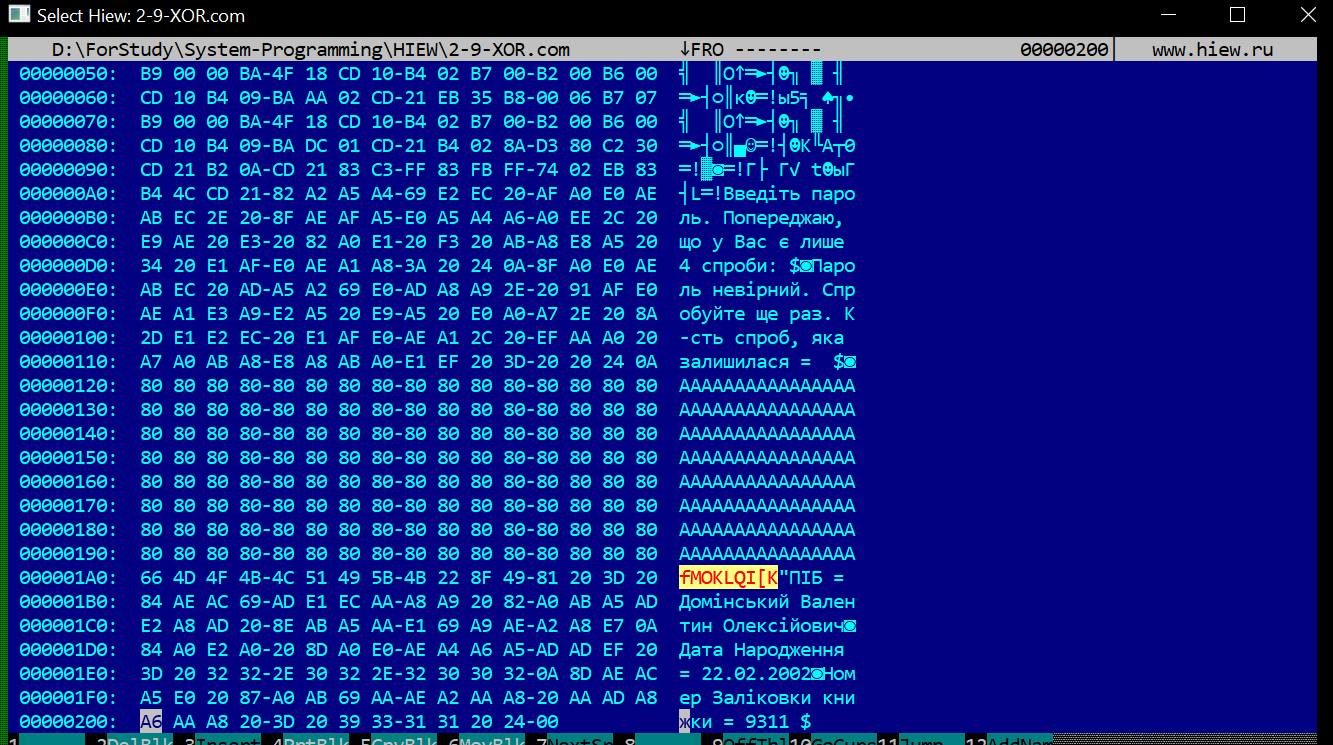


1. Виконати шифрування пароля за допомогою функції XOR, знову скомпілювати COMфайл і переконатися, що тепер вони не виявляються явним чином в тексті виконуваного COMфайлу. Привести скріншоти цієї програми в режимах «Text» та «Decode» у звіті по лабораторній роботі

Text:



Hex:



1. Порівняти текст програми, який набирався в редакторові, з текстом програми в скомпільованому вигляді, який формує HEX-редактор HIEW. Виявити розбіжності і відобразити їх в звіті по лабораторній роботі.

|  |  |
| --- | --- |
| Текст з редактору | Текст з HEX-редактору |
| ; Clear the screen  mov ax, 0600h  mov bh, 7h  mov cx, 0000h  mov dx, 184fh  int 10h  ; Set the position of cursor  mov ah, 02h  mov bh, 00h  mov dl, 00h  mov dh, 00h  int 10h  ; Display The Text  mov ah, 9h  mov dx, offset StartingText  int 21h    mov bx, 03h ; counter for tries    jmp InputOfTheUser ; Unconditional jump | 00000000: B80006 mov ax,00600 00000003: B707 mov bh,7 00000005: B90000 mov cx,0 00000008: BA4F18 mov dx,0184F ;'↑O' 0000000B: CD10 int 010  0000000D: B402 mov ah,2 0000000F: B700 mov bh,0 00000011: B200 mov dl,0 00000013: B600 mov dh,0 00000015: CD10 int 010  00000017: B409 mov ah,9 00000019: BAA401 mov dx,001A4 ;'☺д' 0000001C: CD21 int 021 ;'!'  0000001E: BB0300 mov bx,3  00000021: EB00 jmps 000000023 --↓1 |

Порівнявши даний код я прийшов до таких висновків:

* Перший стовпчик тексту з HEX-редактору – це положення даної команди у пам’яті. Команда «mov ax, 0600h» займає такі місця:
  + B8-00000000
  + 00-00000001
  + 06-00000002

А наступна команда «mov bh,7»:

* + B7-00000003
  + 07-00000004

Це можна зрозуміти просто обравши в HIEW відповідну частину команди. При цьому програма покаже у верхньому правому куті позицію конкретної частини.

* Другий стовпчик – значення стрічки.

Як бачимо, цих даних Ми не писали, але вони є

* А третій – просто текст команди, який Ми самі писали у редакторі.
* Також інколи буває четвертий стовпчик. У ньому, наприклад, можуть показуватися стрілочками вгору або вниз функції “jmp” та її варіації. Також можна помітити символи, які стосуються «offset»

1. На отриманому у п. 9 в режимі «Decode» скріншоті знайти всі команди MOV, виписати їх коди і розібрати по окремих полях, відповідно їх формату. Результати привести в звіті по лабораторній роботі
2. Зробити висновки по лабораторній роботі

**Висновок:**

Завдяки цій лабораторній роботі Я ознайомився з Assembler, зі змінними, сегментами коду, функціями, форматуваннями, лістингом. Також перевірив цифри, які сам перевів у HEX формат з тими, які зробив Assembler. Зверху видно, що деякі дані з лістингу не співпадають (зелений фон). Це можна пояснити тим, що Ми округлювали числа до 10 знаків після коми.