**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

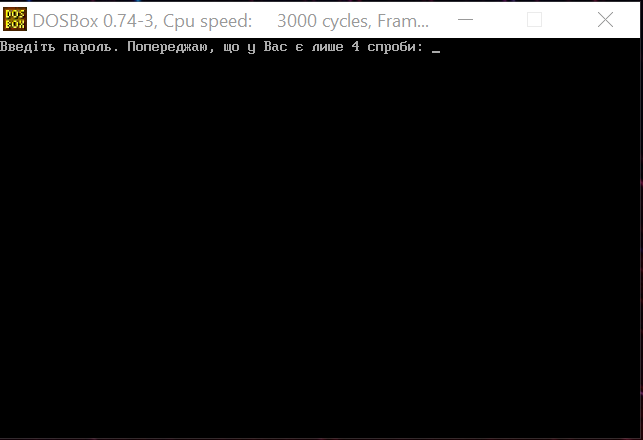
:: "keyb none 866" - to connect the appropriate language

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

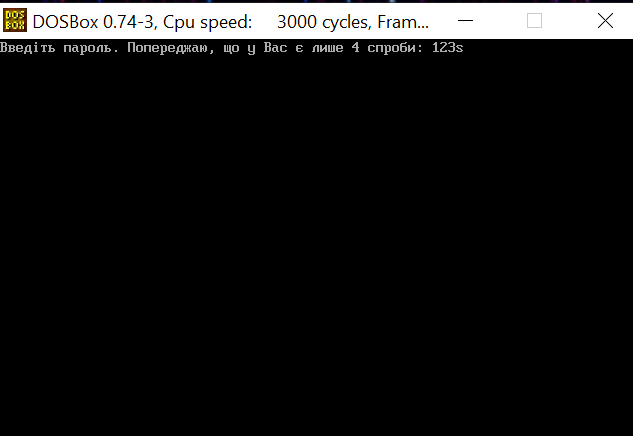
**start** D:\masm32\DosBox\DOSBox-0.74-3\DOSBox.exe -c "mount c **%cd%** " -c c: -c "keyb none 866" -c "**%NameOfTheFileAsParametr:~0,-4%**.**com**"

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

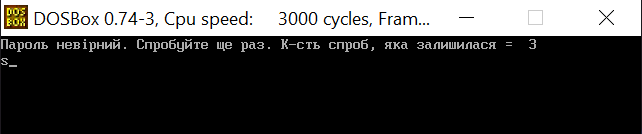
Початкове вікно:



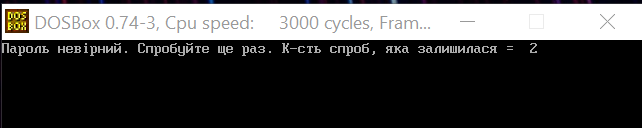
Введення неправильного паролю:

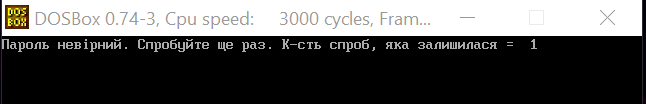


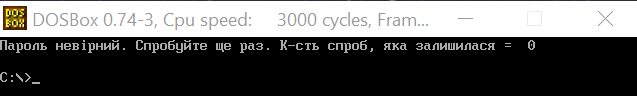
Вивід при такому паролі:



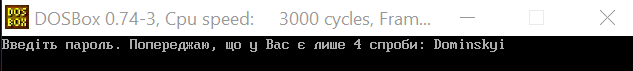
Вивід при введенні неправильного паролю кілька разів:



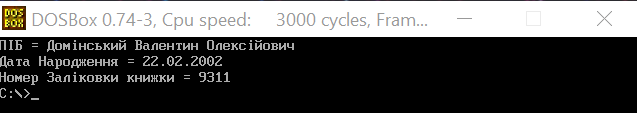




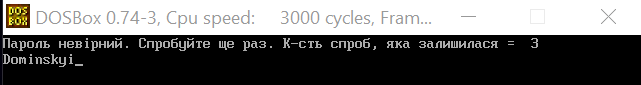
Введення правильно паролю з першого разу:



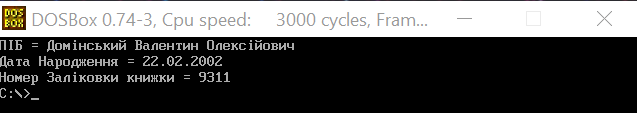
Вивід:



Введення правильно паролю не з першого разу:

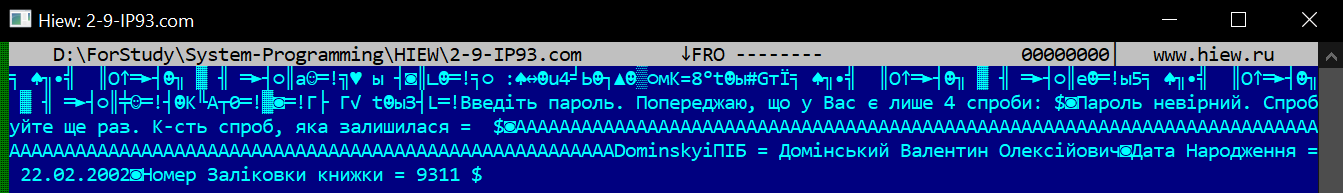


Вивід:

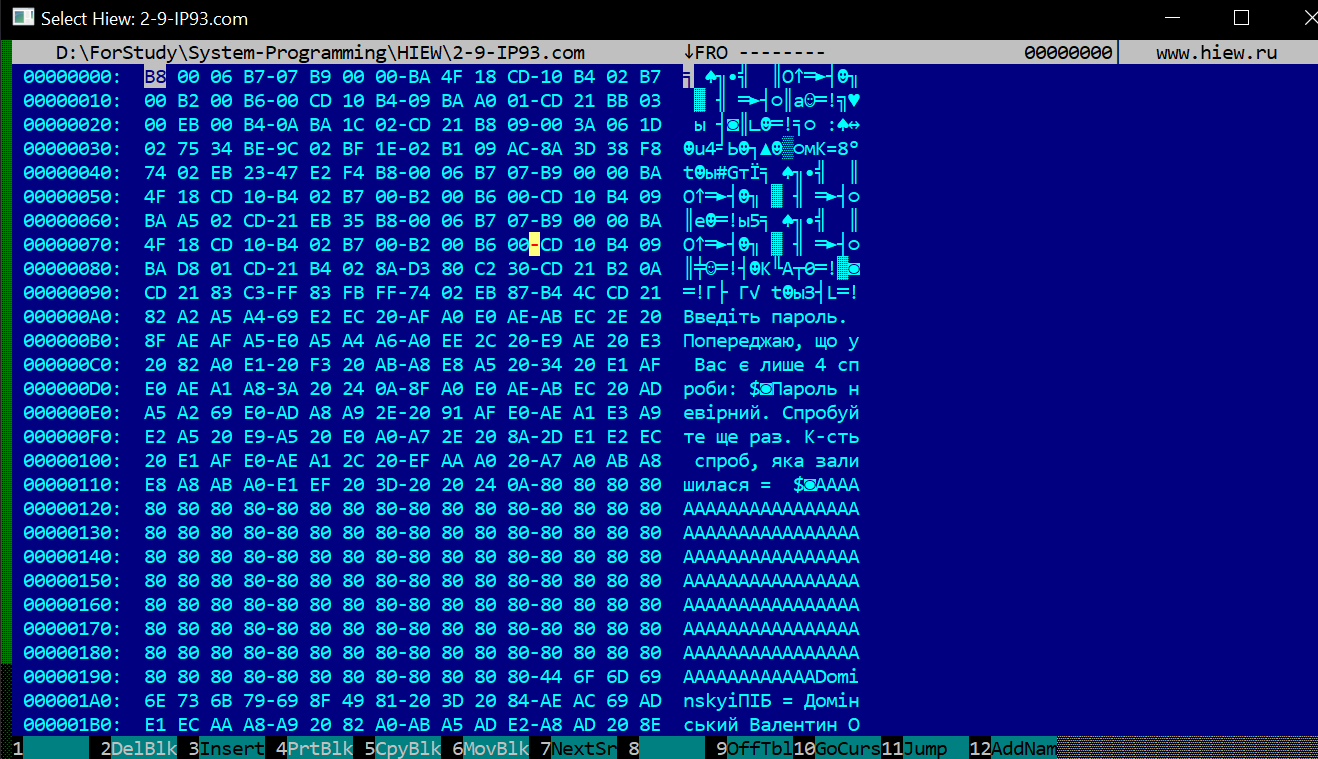


1. Отриманий виконуваний файл дослідити за допомогою програми HEX-редактору HIEW32 або HIEW. У останньому випадку для запуску програми сформувати ярлик, де виконати налаштування параметрів сумісності з використовуваною операційною системою
2. Перемикаючи послідовно режими перегляду (Text – Hex – Decode),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

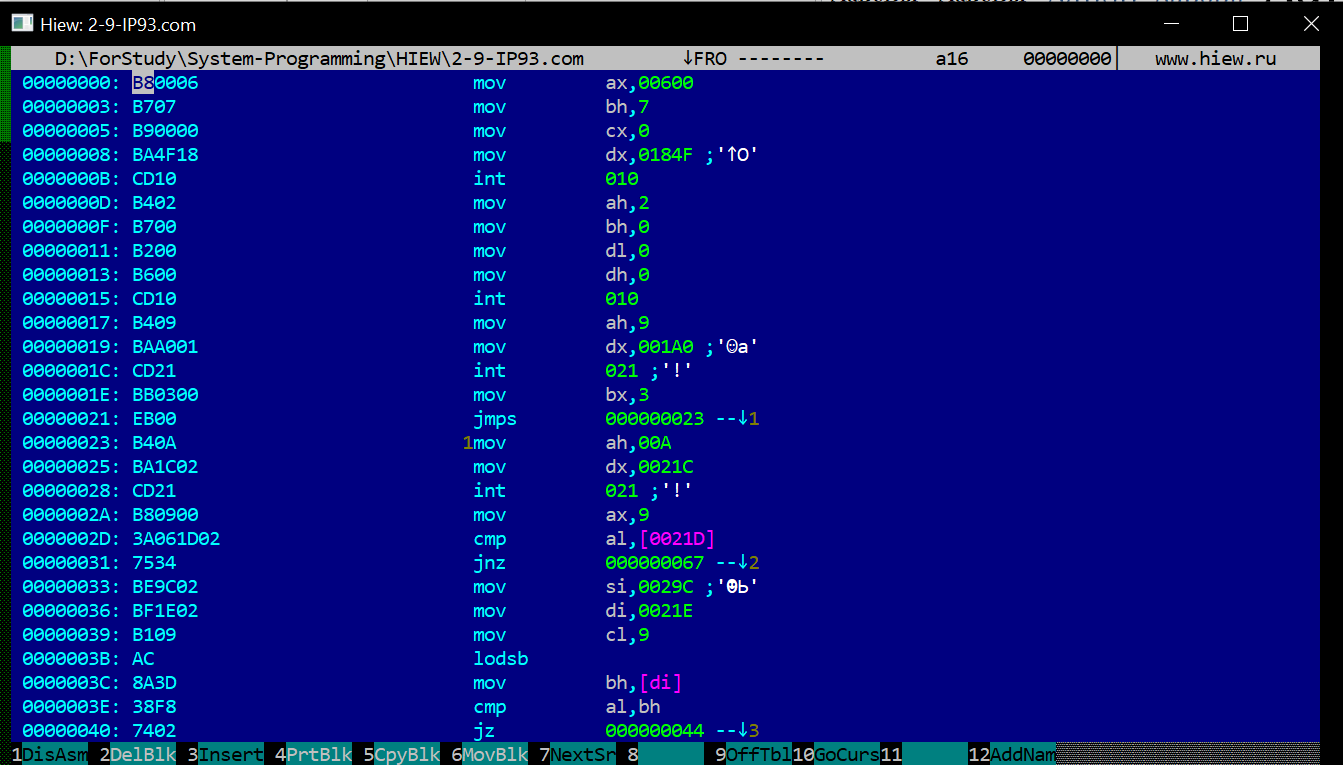
Text:



Hex:

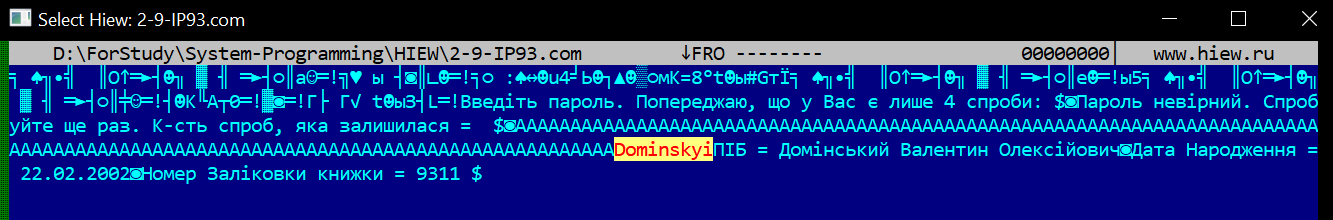


Decode:

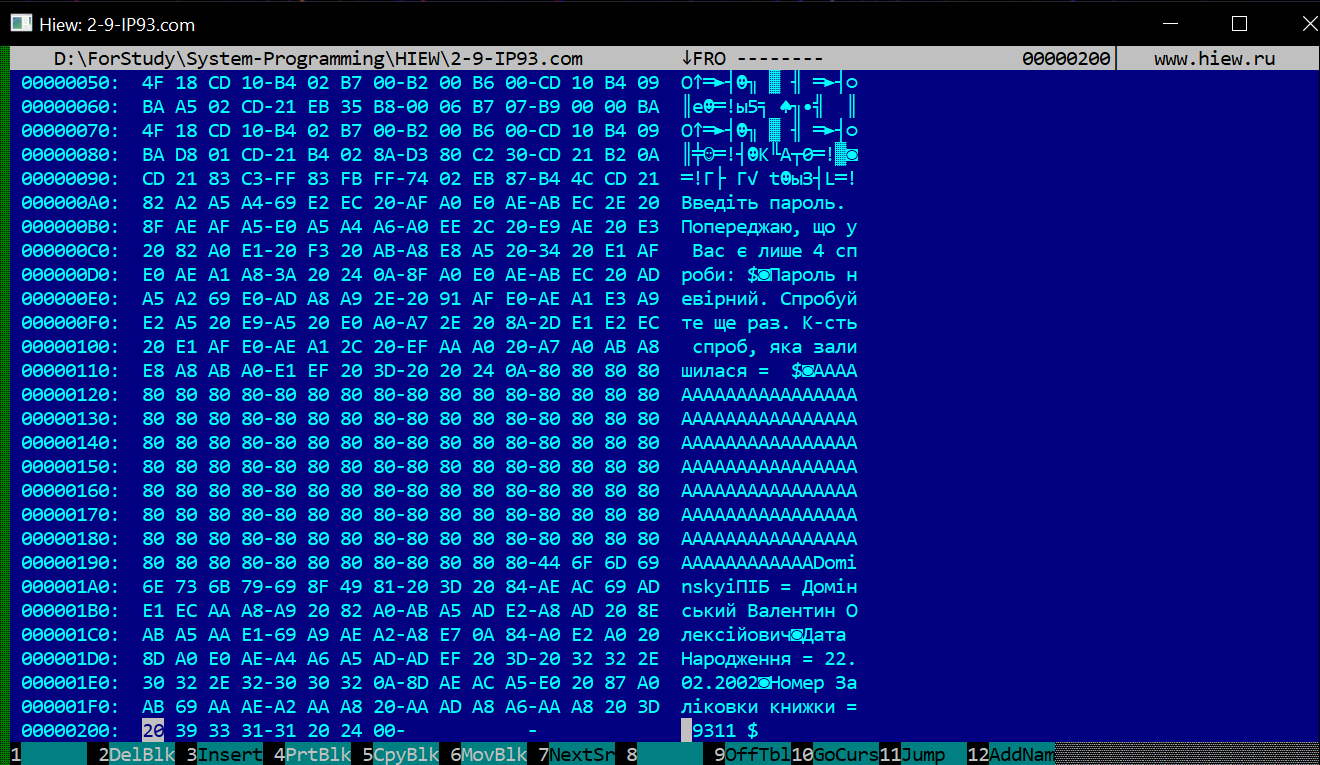


1. Переконатися, що текст оригінала пароля, який міститься в тексті програми, може бути легко виявлений за допомогою HEX-редактора

Пароль у Text:

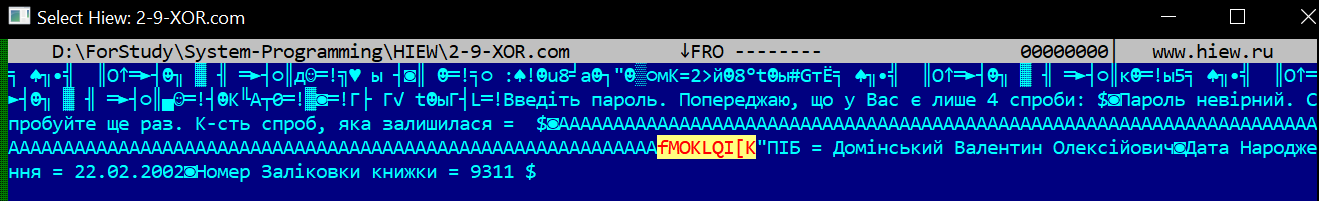


Пароль у Hex:

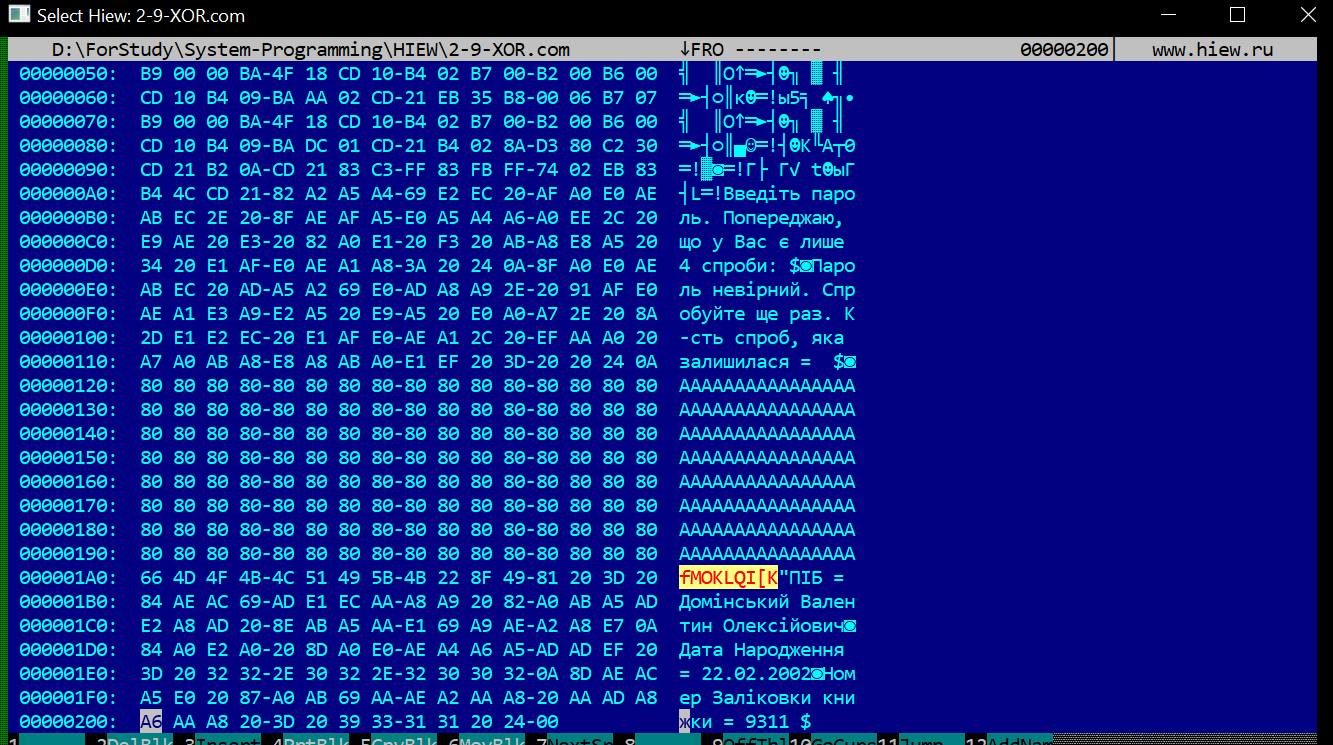


1. Виконати шифрування пароля за допомогою функції XOR, знову скомпілювати COMфайл і переконатися, що тепер вони не виявляються явним чином в тексті виконуваного COMфайлу. Привести скріншоти цієї програми в режимах «Text» та «Decode» у звіті по лабораторній роботі

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 |

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

* F = 100100111101.0010100011

|  |  |
| --- | --- |
| 0.16 | \*2 |
| 0.32 | 0 |
| 0.64 | 0 |
| 1.28 | 1 |
| 0.56 | 0 |
| 1.12 | 1 |
| 0.24 | 0 |
| 0.48 | 0 |
| 0.96 | 0 |
| 1.92 | 1 |
| 1.84 | 1 |
| 1.68 | 1 |

1. За допомогою розрядної сітки показати в звіті представлення цілих чисел в наступних форматах:

* “ddmmyyy” у вигляді символьного рядка;
* числа A и –A у однобайтовому форматі Byte;
* числа A, B, -A и -B у двобайтовому формат Word;
* числа A, B, C, -A , -B и -C у чотирьохбайтовому форматі Shortlnt;
* числа A, B, C, -A , -B и -C у восьмибайтовому форматі Longlnt;
  1. “2202200” у вигляді символьного рядка

|  |  |
| --- | --- |
| Число | Символ |
| 2 | 32 |
| 2 | 32 |
| 0 | 30 |
| 2 | 32 |
| 2 | 32 |
| 0 | 30 |
| 0 | 30 |

* 1. числа A = 22 й –A = -22 у однобайтовому форматі Byte:

|  |  |
| --- | --- |
| Число | Двійковий вигляд |
| А | 0001 0110 |
| -А | 1110 1010 |

* 1. числа A = 22, B = 2202, -A = -22 й –B = -2202 у двобайтовому форматі Word

|  |  |
| --- | --- |
| Число | Двійковий вигляд |
| А | 0000 0000 0001 0110 |
| -А | 1111 1111 1110 1010 |
| B | 0000 1000 1001 1010 |
| -B | 1111 0111 0110 0110 |

* 1. числа A = 22, B = 2202, C = 22022002, -A = -22 , -B = -2202 й –C = -22022002 у чотирьохбайтовому форматі Shortlnt

|  |  |
| --- | --- |
| Число | Двійковий вигляд |
| А | 0000 0000 0000 0000 0000 0000 0001 0110 |
| -А | 1111 1111 1111 1111 1111 1111 1110 1010 |
| B | 0000 0000 0000 0000 0000 1000 1001 1010 |
| -B | 1111 1111 1111 1111 1111 0111 0110 0110 |
| C | 0000 0001 0101 0000 0000 0111 0111 0010 |
| -C | 1111 1110 1010 1111 1111 1000 1000 1101 |

* 1. числа A = 22, B = 2202, C = 22022002, -A = -22 , -B = -2202 й –C = - 22022002 у восьмибайтовому форматі Longlnt

|  |  |
| --- | --- |
| Число | Двійковий вигляд |
| А | 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0110 |
| -А | 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1110 1010 |
| B | 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1000 1001 1010 |
| -B | 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 0111 0110 0110 |
| C | 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0101 0000 0000 0111 0111 0010 |
| -C | 1111 1111 1111 1111 1111 1111 1111 1111 1111 1110 1010 1111 1111 1000 1000 1101 |

1. Представити числа D = 0.002, E = 0.236 и F = 2365.16 у нормалізованому вигляді:

Щоб використовувати дійсні числа їх треба нормалізувати. Робиться це за допомогою такої формули:

A = ZN \* M \* Nq

Де: ZN – знак числа;

M – мантіса числа;

N – основа системи числення;

q – показник;

Візьмемо для початку число D = 0.002, переведемо його в двійкову систему числення:

0 – це 0, а 0.002 – це 0.0000000010.

Отже 0.002 = 0.0000000010b;

Тепер Нам треба здвинути всі числа зліва окрім одної направо або ж навпаки:

0.0000000010b = 1.0 \* 2-9

Таким чином отримаємо:

ZN = 0, M = 0.0, N = 2, q = -9

Робимо те саме з числами E = 0.236 та F = 2365.16:

E = 0.236:

0 – це 0, а 0.236 – це:

0.0011110001.

Отже 0.236 = 0.0011110001b

0.0011110001b = 1.1110001 \* 2-3

ZN = 0, M = 1.1110001, N = 2, q = -3

F = 2365.16:

2365 – це 100100111101, а 0.16 – це:

0.0010100011.

Отже 2365.16 = 100100111101.0010100011b

100100111101.0010100011b = 1.001001111010010100011 \* 211

ZN = +1, M = 1.001001111010010100011, N = 2, q = 11

1. За допомогою розрядної сітки показати в звіті представлення дійсних чисел в наступних форматах:

* числа D = 0.002 и –D = -0.002 в у чотирьохбайтовому форматі Single (float);
* числа E = 0.236 и –E = -0.236 у восьмибайтовому форматі Double (double);
* числа F = 2365.16 и –F = -2365.16 у десятибайтовому форматі Extended (long double);
  + 1. Числа D = 0.002 и –D = -0.002 в у чотирьохбайтовому форматі Single (float):

Додатне число D = 0.002

Нормалізоване = 1.0 \* 2-9

Знак: 0 – додатне

Порядок: До показника q додаємо 127 – -9 + 127 = 11810 = 11101102

Мантіса: 1.0 = 0

Тепер розміщуємо це у 32 бітах за стандартом IEEE 754:

|  |  |  |
| --- | --- | --- |
| 31 | 30 23 | 22 0 |
| 0 | 0111 0110 | 000 0000 0000 0000 0000 0000 |
| Знак | Порядок | Мантіса |

Відповідь = 0011 1011 0000 0000 0000 0000 0000 0000 = 3B000000

Від’ємне число -D = -0.002

Знак: 1 – від’ємне

Усе інше залишається тим самим:

|  |  |  |
| --- | --- | --- |
| 31 | 30 24 | 23 0 |
| 1 | 1111 111 | 0000 0000 1000 0000 0000 0000 |
| Знак | Порядок | Мантіса |

Відповідь = 1011 1011 0000 0000 0000 0000 0000 0000 = BB000000

* + 1. Числа E = 0.236 и –E = -0.236 у восьмибайтовому форматі Double (double):

Додатне число E = 0.236

Нормалізоване = 1.1110001\* 2-3

Знак: 0 – додатне

Порядок: До показника q додаємо 1023 – -3 + 1023 = 102010 = 11111111002

Мантіса: 1.1110001 = 1110001

Тепер розміщуємо це у 64 бітах за стандартом IEEE 754:

|  |  |  |
| --- | --- | --- |
| 63 | 62 52 | 51 0 |
| 0 | 0111 1111 100 | 1110 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| Знак | Порядок | Мантіса |

Відповідь = 0011 1111 1100 1110 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 = 3FCE200000000000

Від’ємне число -E = -0.236

Знак: 1 – від’ємне

Усе інше залишається тим самим:

|  |  |  |
| --- | --- | --- |
| 63 | 62 51 | 50 0 |
| 1 | 1111 1111 11 | 0011 1100 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0 |
| Знак | Порядок | Мантіса |

Відповідь = 1011 1111 1100 1110 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 = BFCE200000000000

* + 1. Числа F = 2365.16 и –F = -2365.16 у десятибайтовому форматі Extended (long double):

Додатне число F = 2365.16

Нормалізоване = 1.001001111010010100011 \* 211

Знак: 0 – додатне

Порядок: До показника q додаємо 16383 – 11 + 16383 = 1639410 = 1000000000010102

Мантіса: 1.001001111010010100011 = 1001001111010010100011 (у цьому форматі записується повністю)

Тепер розміщуємо це у 80 бітах за стандартом IEEE 754:

|  |  |  |
| --- | --- | --- |
| 79 | 78 64 | 63 0 |
| 0 | 1000 0000 0001 010 | 1001 0011 1101 0010 1000 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| Знак | Порядок | Мантіса |

Відповідь = 0100 0000 0000 1010 1001 0011 1101 0010 1000 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 = 400A93D28C0000000000

Від’ємне число -F = -2365.16

Знак: 1 – від’ємне

Усе інше залишається тим самим:

|  |  |  |
| --- | --- | --- |
| 79 | 78 64 | 63 0 |
| 1 | 1000 0000 0001 010 | 1001 0011 1101 0010 1000 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 |
| Знак | Порядок | Мантіса |

Відповідь = 1100 0000 0000 1010 1001 0011 1101 0010 1000 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 = C00A93D28C0000000000

1. 220220010 = 219A5816

2210 = 1616

-2210 = EA16

220210 = 089A16

-220210 = F76616

2202200210 = 0150077216

-2202200210 = FEAFF88E16

0.00210 = 3B00000016

-0.00210 = BB00000016

0.23610 = 3FCE20000000000016

-0.23610 = BFCE20000000000016

2365.1610 = 400A93D28C000000000016

-2365.1610 = C00A93D28C000000000016



; Processors

.386

.model **flat,** **stdcall**

option **CaseMap:None**

; Libraries And Macroses

include **/**masm32**/**include**/**masm32rt.inc

.data?

BufferForText DB 256 DUP**(?)**

BufferDPlus DB 32 DUP**(?)**

BufferDMinus DB 32 DUP**(?)**

BufferEPlus DB 32 DUP**(?)**

BufferEMinus DB 32 DUP**(?)**

BufferFPlus DB 32 DUP**(?)**

BufferFMinus DB 32 DUP**(?)**

; Data Segment

.data

; Name Of Message Box

MsgBoxName DB "1-9-IP93-Dominskyi"**,** 0

; Symbols

Symbols DB "Number is 22022002"**,** 0

; Text Of Message Box

Form DB "Symbols - %s"**,** 10**,**

"A plus = %d"**,** 10**,** "A minus = %d"**,** 10**,**

"B plus = %d"**,** 10**,** "B minus = %d"**,** 10**,**

"C plus = %d"**,** 10**,** "C minus = %d"**,** 10**,**

"D plus = %s"**,** 10**,** "D minus = %s"**,** 10**,**

"E plus = %s"**,** 10**,** "E minus = %s"**,** 10**,**

"F plus = %s"**,** 10**,** "F minus = %s"**,** 0

; A Byte Numbers

APlusByte DB **+**22

AMinusByte DB **-**22

; A Word Numbers

APlusWord DW **+**22

AMinusWord DW **-**22

; A Shortlnt Numbers

APlusShortlnt DD **+**22

AMinusShortlnt DD **-**22

; A Longlnt Numbers

APlusLonglnt DQ **+**22

AMinusLonglnt DQ **-**22

; B Word Numbers

BPlusWord DW **+**2202

BMinusWord DW **-**2202

; B Shortlnt Numbers

BPlusShortlnt DD **+**2202

BMinusShortlnt DD **-**2202

; B Longlnt Numbers

BPlusLonglnt DQ **+**2202

BMinusLonglnt DQ **-**2202

; C Shortlnt Numbers

CPlusShortlnt DD **+**22022002

CMinusShortlnt DD **-**22022002

; C Longlnt Numbers

CPlusLonglnt DQ **+**22022002

CMinusLonglnt DQ **-**22022002

; D Single (Float) Numbers

DPlusSingle DD **+**0.002

DMinusSingle DD **-**0.002

; D Double Numbers

DPlusDouble DQ **+**0.002

DMinusDouble DQ **-**0.002

; E Double Numbers

EPlusDouble DQ **+**0.236

EMinusDouble DQ **-**0.236

; F Double Numbers

FPlusDouble DQ **+**2365.16

FMinusDouble DQ **-**2365.16

; F Extended (Long Double) Numbers

FPlusExtended DT **+**2365.16

FMinusExtended DT **-**2365.16

; Code Segment

.code

; Enter point

Main**:**

invoke FloatToStr2**,** DPlusDouble**,** **addr** BufferDPlus

invoke FloatToStr2**,** DMinusDouble**,** **addr** BufferDMinus

invoke FloatToStr2**,** EPlusDouble**,** **addr** BufferEPlus

invoke FloatToStr2**,** EMinusDouble**,** **addr** BufferEMinus

invoke FloatToStr2**,** FPlusDouble**,** **addr** BufferFPlus

invoke FloatToStr2**,** FMinusDouble**,** **addr** BufferFMinus

invoke wsprintf**,** **addr** BufferForText**,** **addr** Form**,**

**addr** Symbols**,**

APlusShortlnt**,** AMinusShortlnt**,**

BPlusShortlnt**,** BMinusShortlnt**,**

CPlusShortlnt**,** CMinusShortlnt**,**

**addr** BufferDPlus**,** **addr** BufferDMinus**,**

**addr** BufferEPlus**,** **addr** BufferEMinus**,**

**addr** BufferFPlus**,** **addr** BufferFMinus

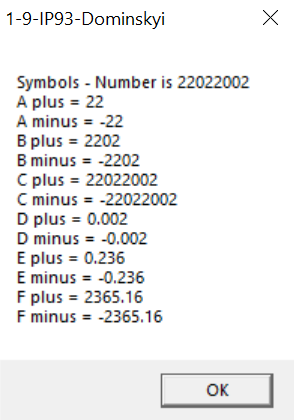
invoke MessageBox**,** 0**,** offset BufferForText**,** offset MsgBoxName**,** MB\_OK

invoke ExitProcess**,** 0

; End of a program

end Main

1. Скріншот програми:



9.

ml /Fl 1-9-IP93-Dominskyi.asm

10.

00000000 **.data?**

00000000 00000100 **[** BufferForText DB 256 DUP**(?)**

00

**]**

00000100 00000020 **[** BufferDPlus DB 32 DUP**(?)**

00

**]**

00000120 00000020 **[** BufferDMinus DB 32 DUP**(?)**

00

**]**

00000140 00000020 **[** BufferEPlus DB 32 DUP**(?)**

00

**]**

00000160 00000020 **[** BufferEMinus DB 32 DUP**(?)**

00

**]**

00000180 00000020 **[** BufferFPlus DB 32 DUP**(?)**

00

**]**

000001A0 00000020 **[** BufferFMinus DB 32 DUP**(?)**

00

**]**

**;** **Data** Segment

00000000 **.data**

**;** Name Of Message Box

00000000 31 2D 39 2D 49 MsgBoxName DB "1-9-IP93-Dominskyi"**,** 0

50 39 33 2D

44 6F 6D 69

6E 73 6B 79

69 00

**;** Symbols

00000013 4E 75 6D 62 65 Symbols DB "Number is 22022002"**,** 0

72 20 69 73

20 32 32 30

32 32 30 30

32 00

**;** Text Of Message Box

00000026 53 79 6D 62 6F Form DB "Symbols - %s"**,** 10**,**

6C 73 20 2D

20 25 73 0A

41 20 70 6C

75 73 20 3D

20 25 64 0A

41 20 6D 69

6E 75 73 20

3D 20 25 64

0A 42 20 70

6C 75 73 20

3D 20 25 64

0A 42 20 6D

69 6E 75 73

20 3D 20 25

64 0A 43 20

70 6C 75 73

20 3D 20 25

64 0A 43 20

6D 69 6E 75

73 20 3D 20

25 64 0A 44

20 70 6C 75

73 20 3D 20

25 73 0A 44

20 6D 69 6E

75 73 20 3D

20 25 73 0A

45 20 70 6C

75 73 20 3D

20 25 73 0A

45 20 6D 69

6E 75 73 20

3D 20 25 73

0A 46 20 70

6C 75 73 20

3D 20 25 73

0A 46 20 6D

69 6E 75 73

20 3D 20 25

73 00

"A plus = %d"**,** 10**,** "A minus = %d"**,** 10**,**

"B plus = %d"**,** 10**,** "B minus = %d"**,** 10**,**

"C plus = %d"**,** 10**,** "C minus = %d"**,** 10**,**

"D plus = %s"**,** 10**,** "D minus = %s"**,** 10**,**

"E plus = %s"**,** 10**,** "E minus = %s"**,** 10**,**

"F plus = %s"**,** 10**,** "F minus = %s"**,** 0

**;** A Byte Numbers

000000C9 16 APlusByte DB **+**22

000000CA EA AMinusByte DB **-**22

**;** A Word Numbers

000000CB 0016 APlusWord DW **+**22

000000CD FFEA AMinusWord DW **-**22

**;** A Shortlnt Numbers

000000CF 00000016 APlusShortlnt DD **+**22

000000D3 FFFFFFEA AMinusShortlnt DD **-**22

**;** A Longlnt Numbers

000000D7 APlusLonglnt DQ **+**22

0000000000000016

000000DF AMinusLonglnt DQ **-**22

FFFFFFFFFFFFFFEA

**;** B Word Numbers

000000E7 089A BPlusWord DW **+**2202

000000E9 F766 BMinusWord DW **-**2202

**;** B Shortlnt Numbers

000000EB 0000089A BPlusShortlnt DD **+**2202

000000EF FFFFF766 BMinusShortlnt DD **-**2202

**;** B Longlnt Numbers

000000F3 BPlusLonglnt DQ **+**2202

000000000000089A

000000FB BMinusLonglnt DQ **-**2202

FFFFFFFFFFFFF766

**;** C Shortlnt Numbers

00000103 01500772 CPlusShortlnt DD **+**22022002

00000107 FEAFF88E CMinusShortlnt DD **-**22022002

**;** C Longlnt Numbers

0000010B CPlusLonglnt DQ **+**22022002

0000000001500772

00000113 CMinusLonglnt DQ **-**22022002

FFFFFFFFFEAFF88E

**;** D Single **(**Float**)** Numbers

0000011B 3B03126F DPlusSingle DD **+**0**.**002

0000011F BB03126F DMinusSingle DD **-**0**.**002

**;** D Double Numbers

00000123 DPlusDouble DQ **+**0**.**002

3F60624DD2F1A9FC

0000012B DMinusDouble DQ **-**0**.**002

BF60624DD2F1A9FC

**;** **E** Double Numbers

00000133 EPlusDouble DQ **+**0**.**236

3FCE353F7CED9168

0000013B EMinusDouble DQ **-**0**.**236

BFCE353F7CED9168

**;** F Double Numbers

00000143 FPlusDouble DQ **+**2365**.**16

40A27A51EB851EB8

0000014B FMinusDouble DQ **-**2365**.**16

C0A27A51EB851EB8

**;** F Extended **(**Long Double**)** Numbers

00000153 FPlusExtended DT **+**2365**.**16

400A93D28F5C28F5C28F

0000015D FMinusExtended DT **-**2365**.**16

C00A93D28F5C28F5C28F

**;** Code Segment

00000000 **.**code

**;** Enter point

00000000 Main**:**

**invoke** FloatToStr2**,** DPlusDouble**,** addr BufferDPlus

**invoke** FloatToStr2**,** DMinusDouble**,** addr BufferDMinus

**invoke** FloatToStr2**,** EPlusDouble**,** addr BufferEPlus

**invoke** FloatToStr2**,** EMinusDouble**,** addr BufferEMinus

**invoke** FloatToStr2**,** FPlusDouble**,** addr BufferFPlus

**invoke** FloatToStr2**,** FMinusDouble**,** addr BufferFMinus

**invoke** wsprintf**,** addr BufferForText**,** addr Form**,**

addr Symbols**,**

APlusShortlnt**,** AMinusShortlnt**,**

BPlusShortlnt**,** BMinusShortlnt**,**

CPlusShortlnt**,** CMinusShortlnt**,**

addr BufferDPlus**,** addr BufferDMinus**,**

addr BufferEPlus**,** addr BufferEMinus**,**

addr BufferFPlus**,** addr BufferFMinus

**invoke** MessageBox**,** 0**,** offset BufferForText**,** offset MsgBoxName**,** MB\_OK

**invoke** ExitProcess**,** 0

**;** End of a **program**

end Main

12.

|  |  |  |
| --- | --- | --- |
| Число | Варіант у лістингу | Варіант у роботі |
| +А | 16 | 16 |
| -А | EA | EA |
| +B | 089A | 089A |
| -B | F766 | F766 |
| +C | 01500772 | 01500772 |
| -C | FEAFF88E | FEAFF88E |
| +D | 3B03126F | 3B031200 |
| -D | BB03126F | 7B031200 |
| +E | 3FCE353F7CED9168 | 3FCE200000000000 |
| -E | BFCE353F7CED9168 | BFCE200000000000 |
| +F | 400A93D28F5C28F5C28F | 400A93D28C0000000000 |
| -F | C00A93D28F5C28F5C28F | C00A93D28C0000000000 |

**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.