НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіти до комп’ютерних практикумів дисципліни

«Системне програмне забезпечення»

Прийняв
доцент кафедри ІПІ
Лісовиченко О.І.
“14” _лютого_ 2025 р.

Виконав
Студент групи ІП-35
Адаменко А.Б.

Київ – 2025

Комп'ютерний практикум №1

Тема: Створення програм на асемблері

Завдання:

1. Для програми, наведеної вище, створити файл типу .asm. Ця програма не має засобів виводу даних, тому правильність її виконання треба перевірити за допомогою td.exe.
2. Скомпілювати програму, включивши потрібні опції для налагоджувача та створення файлу лістингу типу .lst.
3. Ознайомитись зі структурою файлу .lst. За вказівкою викладача, для певної команди асемблера розглянути структуру машинної команди і навести її у звіті.
4. Скомпонувати .obj-файл програми. Включити опції для налагодження та створення .map-файлу.
5. Занести до звіту адреси початку та кінця всіх сегментів з .map-файлу.
6. Завантажити до налагоджувача td.exe одержаний .exe-файл програми.
7. У вікні CPU у полі DUMP знайти початкову адресу сегмента даних та записати його до звіту. Знайти масиви Source та Destination. Дані у масиві Source подаються у шістнадцятковій системі.
8. У покроковому режимі за допомогою клавіші F7 виконати програму. Одержані результати у масиві Destination показати викладачеві.

Текст програми

```
; declare a stack segment
STSEG segment para stack "STACK"

    ; a space for a stack
    db 64 dup("STACK")

STSEG ends

; declare a data segment
DSEG segment para public "DATA"

    ; A data buffer to be cloned from
    Source db 10, 20, 30, 40

    ; A data buffer to be cloned to
    Destination db 4 dup("?")

DSEG ends

; declare a code segment
CSEG segment para public "CODE"

    ; an entry point
```

```
    MAIN proc far

        ; core initialization
        ; initialize segment registers
        assume cs: CSEG, ds: DSEG, ss: STSEG

        ; as it is a procedure, push all registers which
will be overwritten
        push ds
        mov ax, 0
        push ax

        ; DS initialization
        mov ax, DSEG
        mov ds, ax

        ; zeroing of a destination
        mov Destination + 0, 0   ; store a byte 0 by an
address Destination + 1
        mov Destination + 1, 0
        mov Destination + 2, 0
        mov Destination + 3, 0

        ; cloning from a source to a destination
(reversed)
      mov al, Source + 0       ; take a byte by an
address Source + 0 to a register AL
        mov Destination + 3, al  ; take a register AL and
store a byte by an address Destination + 3

        mov al, Source + 1
        mov Destination + 2, al

        mov al, Source + 2
        mov Destination + 1, al

        mov al, Source + 3
        mov Destination + 0, al

        ; exit
        ret

    MAIN endp
    CSEG ends
```

```
end MAIN
; end of proc "main" and a code segment in order which
idk why it is
```

Введені та отримані результати

Вміст .lst файлу:

```
Turbo Assembler   Version 4.0     02/10/25 13:32:52
Page 1
lab1\main.asm



      1                     ; declare a stack segment
      2 0000                STSEG segment para  stack
"STACK"
      3
      4                     ; a space for a stack
      5 0000  40*(53 54  41 43 4B)     db 64 dup("STACK")
      6
      7 0140                STSEG ends
      8
      9                     ; declare a data segment
     10 0000                DSEG segment para public "DATA"
     11
     12                      ; A data buffer to be cloned
from
     13 0000  0A 14 1E 28          Source db 10, 20, 30,
40
     14
     15                     ; A data buffer to be cloned to
     16 0004  04*(3F)              Destination db  4
dup("?")
     17
     18 0008                DSEG ends
     19
     20                     ; declare a code segment
     21 0000                CSEG segment para public "CODE"
     22
     23                     ; an entry point
     24 0000                MAIN proc far
     25
     26                         ; core initialization
```

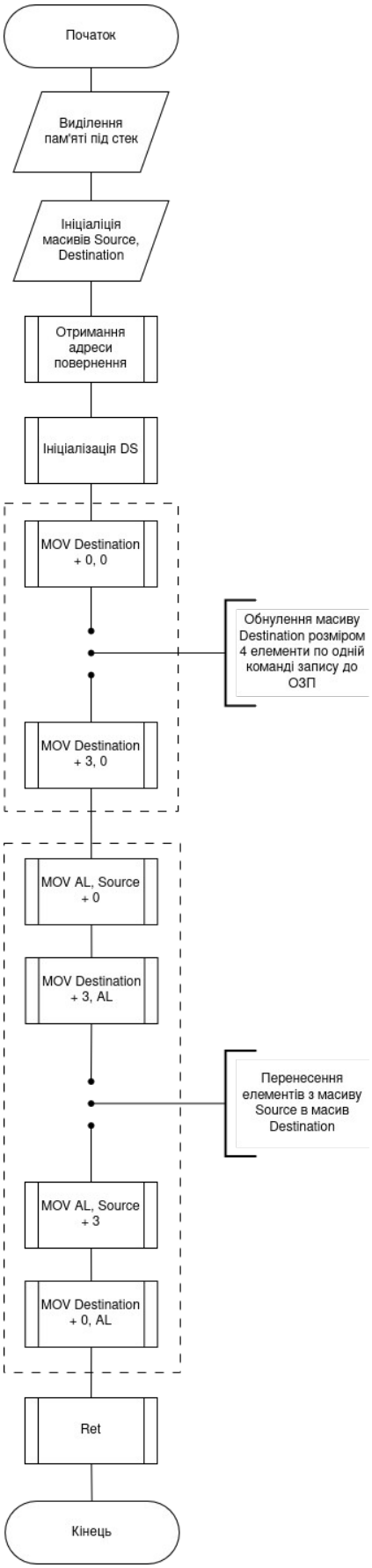
```
     27                              ;  initialize  segment
registers
     28                         assume cs:  CSEG, ds: DSEG,
    ss: STSEG
     29
     30                         ; as it is  a    procedure,
push all registers   which will be overwritten
     31  0000  1E              push ds
     32  0001  B8 0000                  mov ax, 0
     33  0004  50              push ax
     34
     35                         ; DS initialization
     36  0005  B8 0000s                 mov ax, DSEG
     37  0008  8E D8                mov ds, ax
     38
     39                         ; zeroing of a destination
     40  000A  C6 06 0004r 00           mov Destination +
0, 0   ; store a  byte 0 by an address Destination + 1
     41  000F  C6 06 0005r 00           mov Destination +
1, 0
     42  0014  C6 06 0006r 00           mov Destination +
2, 0
     43  0019  C6 06 0007r 00           mov Destination +
3, 0
     44
     45                          ; cloning from a source to
    a destination (reversed)
     46  001E  A0 0000r                 mov al, Source + 0
          ;  take a byte by an address Source + 0 to   a
register AL
     47  0021  A2 0007r                 mov Destination +
3, al  ; take a register  AL  and  store  a  byte  by  an
address     +
     48                   Destination + 3
     49
     50  0024  A0 0001r                mov al, Source + 1
     51  0027  A2 0006r                 mov Destination +
2, al
     52
     53  002A  A0 0002r                mov al, Source + 2
     54  002D  A2 0005r                 mov Destination +
1, al
     55
     56  0030  A0 0003r                mov al, Source + 3
```

```
     57  0033  A2 0004r                 mov Destination +
0, al

Turbo Assembler   Version 4.0     02/10/25 13:32:52
Page 2
lab1\main.asm



     58
     59                          ; exit
     60  0036  CB               ret
     61
     62  0037                MAIN endp
     63  0037                CSEG ends
     64
     65                      end MAIN

Turbo Assembler   Version 4.0     02/10/25 13:32:52
Page 3
Symbol Table



Symbol Name             Type    Value

??DATE                  Text    "02/10/25"
??FILENAME              Text    "main    "
??TIME                  Text    "13:32:52"
??VERSION               Number 0400
@CPU                Text    0101H
@CURSEG                 Text    CSEG
@FILENAME               Text    MAIN
@WORDSIZE               Text    2
DESTINATION             Byte    DSEG:0004
MAIN                Far     CSEG:0000
SOURCE                  Byte    DSEG:0000

Groups & Segments           Bit Size Align  Combine Class

CSEG                16  0037 Para    Public  CODE
DSEG                16  0008 Para    Public  DATA
STSEG                   16  0140 Para    Stack   STACK
```

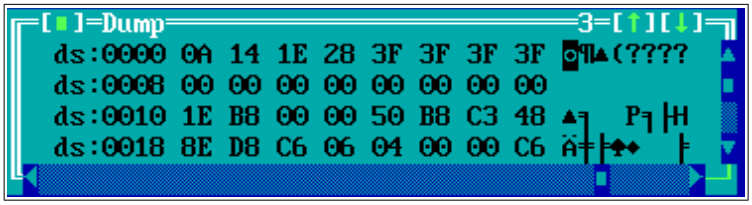
Вміст .map файлу:

```
 Start  Stop   Length Name               Class

 00000H 0013FH 00140H STSEG              STACK
 00140H 00147H 00008H DSEG               DATA
 00150H 00186H 00037H CSEG               CODE

  Address         Publics by Name


  Address         Publics by Value


Program entry point at 0015:0000
```

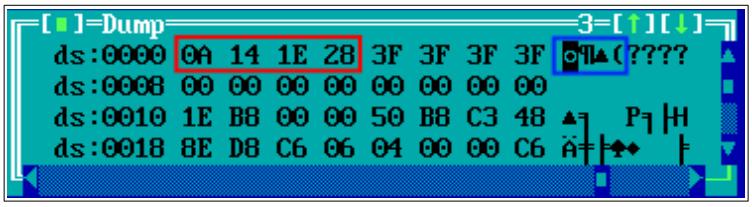
Схема функціонування програми



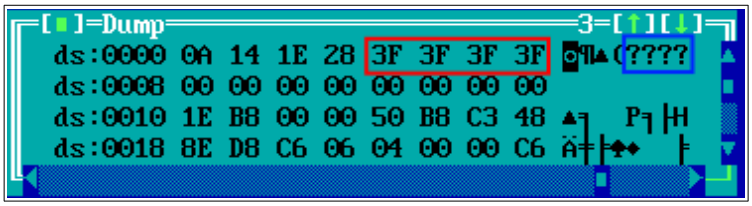
Вікно DUMP



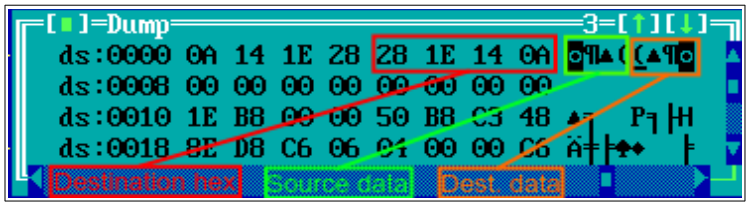
До виконання програми:
Масив Source:



Масив Destination:



Після виконання програми
Масив Destination:



Висновок:

1. В текстовому редакторі було створено файл типу .asm.
2. Скомпілював програму, включивши потрібні опції для налагоджувача та створення файлу лістингу типу .lst.
3. Ознайомився зі структурою файлу .lst. Розглянув структури машинних команд.
4. Після усунення помилок, скомпонував .obj-файл програми, включивши опції для налагодження та створення .map-файлу.
5. Відкрив файл карти памяті (.map-файл) та подивився на адреси початку та кінця всіх сегментів програми.
6. Завантажив програму налагоджувача td.exe та мій одержаний .exe-файл програми.
7. У вікні CPU у полі DUMP подивився на початкову адресу сегмента даних. В сегменті даниз знайшов масиви Source та Destination. Дані у масиві Source подаються у шістнадцятковій системі.

8. У покроковому режимі за допомогою клавіші F7 виконав програму. Програма коректно виконує поставлену задачу.app