Міністерство освіти і науки України
Національний університет “Запорізька політехніка”
Кафедра програмних засобів

ЗВІТ

З лабораторної роботи №2
З дисципліни “Архітектура Комп’ютера та Низькорівневе Програмування”
З теми “Структури програм на мові асемблера”
Варіант №20

Виконав:
Студент групи КНТ-122 О.А. Онищенко

Прийняли:
Ст. Викладач О. І. Качан
Доцент КТН А. Є. Казурова

2023

1 Завдання

1.1 Текст завдання

1. Провести практичне ознайомлення з компілятором TASM та лінкувальником TLINK і ознайомитися з їх аргументами, параметрами та способом використання.

2. У текстовому редакторі створити початковий код на мові асемблера для exe-програм з наступними вимогами до коду:

- реалізувати виведення до консолі будь-якого друкованого символу;
- на етапі завершення виконання програми реалізувати повернення до операційної системи коду помилки зі значенням згідно варіанту як у ЛР №1 (регістр BX);
- обов'язково у потрібних місцях коду прописувати коментарі;
- у якості імен сегментів та точки входу використовувати зрозумілі, не довгі але оригінальні назви.

3. У текстовому редакторі створити початковий код на мові асемблера для com-програм з наступними вимогами до коду:

- створити дві версії коду com-програм:
  а) без використання модульних процедур;
  б) з використанням модульної процедури;
- реалізувати виведення до консолі будь-якого друкованого символу;
- на етапі завершення виконання програми реалізувати повернення до операційної системи коду помилки зі значенням згідно варіанту як у ЛР №1 (регістр BX);
- обов'язково у потрібних місцях коду прописувати коментарі;
- у якості імен сегментів та точки входу використовувати зрозумілі, не довгі але оригінальні назви.

4. Над створеними початковими кодами програм виконати процеси компіляції, лінкування та позбавитись від помилок (якщо вони є).

5. Виконати демонстрацію структури програми у відлагоджувачі DEBUG:

- дизасемблювати виконуваний код програми;
- вказати на місця початку/кінця наступних частин програми: PSP, CODE Segment, DATA Segment, STACK Segment;
- знайти у виконуваному коді символ, який повинен бути виведено до консолі;
- знайти у виконуваному коді значення, яке має бути передано до змінної коду помилки операційної системи.

6. Виконати запуск програм у консолі з демонстрацією виведеного символу та кодом помилки. Перевірити коректність значення коду помилки згідно варіанту.

7. Оформити звіт згідно ДСТУ 3008:2015 та завантажити поряд з ZIP-архівом власноруч створених початкових кодів програм. Документ звіту та архів повинні мати у назві Група_Прізвище_НомерЛР.

2 Хід роботи

2.1 EXE Програма

2.1.1 Код програми

```
.model small    ; set program model as small
.stack 100h     ; set stack size to 100h

sseg segment para stack 'stack'     ; declare stack segment
    db 256 dup(?)       ; reserve memory for stack
sseg ends   ; end stack segment

dseg segment para public 'data'     ; declare data segment
    symbol db 'X'   ; declare symbol variable
dseg ends   ; end data segment

cseg segment para public 'code'     ; declare code segment
    assume cs:cseg, ss:sseg, es:nothing ; set segment register to
corresponding ones

start:  ; declare program entry point
    assume ds:dseg  ; set data segment register
    mov bx, dseg    ; add data segment to bx register
    mov ds, bx  ; set ds register to bx register

    call main   ; call main function

    mov ah, 4Ch     ; exit to OS
    mov bl, 6Ch     ; set error code to 108 in hex
    int 21h     ; call interrupt

main proc near  ; declare main function
    mov dl, symbol  ; load symbol into dl register
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt
```

```
    ret     ; stop function execution
main endp   ; end main function

cseg ends   ; end code segment
end start   ; end program execution
```

2.1.2 Дизасемблювання

```
Z:\ASM\TASM>debug exe.exe
-u
0B4F:0000 BB510B          MOV     BX,0B51
0B4F:0003 8EDB            MOV     DS,BX
0B4F:0005 E80600          CALL    000E
0B4F:0008 B44C            MOV     AH,4C
0B4F:000A B36C            MOV     BL,6C
0B4F:000C CD21            INT     21
0B4F:000E 8A160000        MOV     DL,[0000]
0B4F:0012 B402            MOV     AH,02
0B4F:0014 CD21            INT     21
0B4F:0016 C3              RET
0B4F:0017 0000            ADD     [BX+SI],AL
0B4F:0019 0000            ADD     [BX+SI],AL
0B4F:001B 0000            ADD     [BX+SI],AL
0B4F:001D 0000            ADD     [BX+SI],AL
0B4F:001F 0058FF          ADD     [BX+SI-01],BL
```

2.1.3 Вказівки

2.1.3.1 PSP

```
Z:\ASM\TASM>debug exe.exe
-d 0 100
0B3F:0000  CD 20 FF 9F 00 9A F0 FE-1D F0 4F 03 54 05 8A 03   . ........O.T...
0B3F:0010  54 05 17 03 54 05 43 05-01 01 01 00 02 FF FF FF   T...T.C.........
0B3F:0020  FF FF FF FF FF FF FF FF-FF FF FF FF 01 0B 4C 01   ..............L.
0B3F:0030  14 0A 14 00 18 00 3F 0B-FF FF FF FF 00 00 00 00   ......?.........
0B3F:0040  05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0B3F:0050  CD 21 CB 00 00 00 00 00-00 00 00 00 00 20 20 20   .!...........
0B3F:0060  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20            .....
0B3F:0070  20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00            ........
0B3F:0080  00 0D 65 78 65 2E 65 78-65 0D 4F 42 4A 0D 42 4A   ..exe.exe.OBJ.BJ
0B3F:0090  0D 41 30 0D 64 64 72 65-73 73 2E 20 20 46 6F 72   .A0.ddress.  For
0B3F:00A0  20 65 78 61 6D 70 6C 65-3A 0D 20 6F 6E 20 4E 54    example:. on NT
0B3F:00B0  56 44 4D 2C 20 73 70 65-63 69 66 79 20 61 6E 20   VDM, specify an
0B3F:00C0  69 6E 76 61 6C 69 64 0D-20 6F 6E 6C 79 2E 0D 00   invalid. only...
0B3F:00D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0B3F:00E0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0B3F:00F0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0B3F:0100  BB                                                .
```

2.1.3.2 Segments

```
Z:\ASM\TASM>debug exe.exe
-u
0B4F:0000 BB510B        MOV     BX,0B51
0B4F:0003 8EDB          MOV     DS,BX
0B4F:0005 E80600        CALL    000E
0B4F:0008 B44C          MOV     AH,4C
0B4F:000A B36C          MOV     BL,6C
0B4F:000C CD21          INT     21
0B4F:000E 8A160000      MOV     DL,[0000]
0B4F:0012 B402          MOV     AH,02
0B4F:0014 CD21          INT     21
0B4F:0016 C3            RET
0B4F:0017 0000          ADD     [BX+SI],AL
0B4F:0019 0000          ADD     [BX+SI],AL
0B4F:001B 0000          ADD     [BX+SI],AL
0B4F:001D 0000          ADD     [BX+SI],AL
0B4F:001F 0058FF        ADD     [BX+SI-01],BL
```

2.1.3.3 Символ

```
0B4F:000E 8A160000      MOV     DL,[0000]
```

2.1.3.4 Код помилки

```
0B4F:000A B36C            MOV     BL,6C
```

2.1.4 Результат виконання

```
Z:\ASM\TASM>TASM.EXE exe.asm
Turbo Assembler  Version 3.2i  Copyright (c) 1988, 1992 Borland International
Serial No:   Tester:

Assembling file:   exe.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  455k


Z:\ASM\TASM>TLINK.EXE EXE.OBJ
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International

Z:\ASM\TASM>EXE.EXE
X
```

2.2 COM Програма без модульних процедур

2.2.1 Код програми

```
.model tiny     ; set program model as tiny

cseg segment para public 'code'     ; declare code segment
    assume cs:cseg, ds:cseg, ss:cseg, es:nothing    ; set each code
segment to code seg as it is the only segment in the program
    org 100h    ; start loading the first instruction at 100h

start:  ; declare program entry point
    symbol db 'U'   ; declare a symbol to output

    xor ax, ax   ; clear ax register
    mov dl, symbol      ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'k'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'r'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'a'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'i'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'n'     ; load symbol to stdout
```

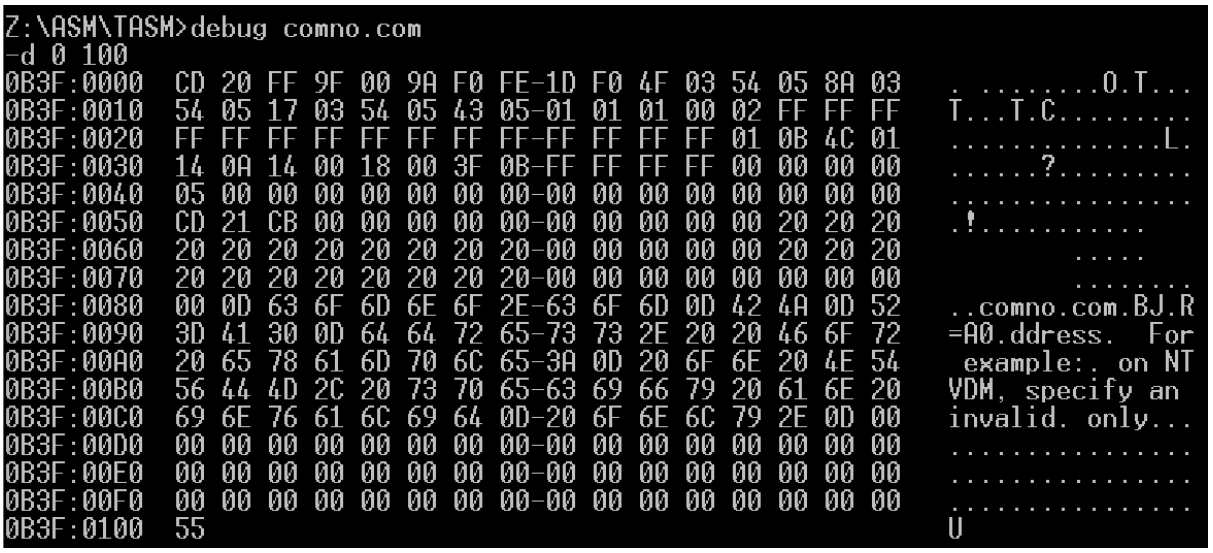
```
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'e'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 10      ; set dl register to new line
    mov ah, 02h     ; output it to stdout
    int 21h     ; call interrupt
    mov dl, 13      ; set dl register to carret return
    mov ah, 02h     ; output it to stdout
    int 21h     ; call interrupt

    mov ah, 04Ch    ; exit to OS
    mov bl, 6Ch     ; set error code to 108 in hex
    int 21h     ; call interrupt

cseg ends   ; close segment
end start   ; end program execution
```

2.2.2 Дизасемблювання

```
Z:\ASM\TASM>debug comno.com
-u
0B3F:0100 55              PUSH    BP
0B3F:0101 33C0            XOR     AX,AX
0B3F:0103 8A160001        MOV     DL,[0100]
0B3F:0107 B402            MOV     AH,02
0B3F:0109 CD21            INT     21
0B3F:010B B26B            MOV     DL,6B
0B3F:010D B402            MOV     AH,02
0B3F:010F CD21            INT     21
0B3F:0111 B272            MOV     DL,72
0B3F:0113 B402            MOV     AH,02
0B3F:0115 CD21            INT     21
0B3F:0117 B261            MOV     DL,61
0B3F:0119 B402            MOV     AH,02
0B3F:011B CD21            INT     21
0B3F:011D B269            MOV     DL,69
0B3F:011F B402            MOV     AH,02
-u
0B3F:0121 CD21            INT     21
0B3F:0123 B26E            MOV     DL,6E
0B3F:0125 B402            MOV     AH,02
0B3F:0127 CD21            INT     21
0B3F:0129 B265            MOV     DL,65
0B3F:012B B402            MOV     AH,02
0B3F:012D CD21            INT     21
0B3F:012F B20A            MOV     DL,0A
0B3F:0131 B402            MOV     AH,02
0B3F:0133 CD21            INT     21
0B3F:0135 B20D            MOV     DL,0D
0B3F:0137 B402            MOV     AH,02
0B3F:0139 CD21            INT     21
0B3F:013B B44C            MOV     AH,4C
0B3F:013D B36C            MOV     BL,6C
0B3F:013F CD21            INT     21
```

2.2.3 Вказівки

2.2.3.1 PSP



2.2.3.2 Segments

```
Z:\ASM\TASM>debug comno.com
-u
0B3F:0100 55              PUSH    BP
0B3F:0101 33C0            XOR     AX,AX
0B3F:0103 8A160001        MOV     DL,[0100]
0B3F:0107 B402            MOV     AH,02
0B3F:0109 CD21            INT     21
0B3F:010B B26B            MOV     DL,6B
0B3F:010D B402            MOV     AH,02
0B3F:010F CD21            INT     21
0B3F:0111 B272            MOV     DL,72
0B3F:0113 B402            MOV     AH,02
0B3F:0115 CD21            INT     21
0B3F:0117 B261            MOV     DL,61
0B3F:0119 B402            MOV     AH,02
0B3F:011B CD21            INT     21
0B3F:011D B269            MOV     DL,69
0B3F:011F B402            MOV     AH,02
-u
0B3F:0121 CD21            INT     21
0B3F:0123 B26E            MOV     DL,6E
0B3F:0125 B402            MOV     AH,02
0B3F:0127 CD21            INT     21
0B3F:0129 B265            MOV     DL,65
0B3F:012B B402            MOV     AH,02
0B3F:012D CD21            INT     21
0B3F:012F B20A            MOV     DL,0A
0B3F:0131 B402            MOV     AH,02
0B3F:0133 CD21            INT     21
0B3F:0135 B20D            MOV     DL,0D
0B3F:0137 B402            MOV     AH,02
0B3F:0139 CD21            INT     21
0B3F:013B B44C            MOV     AH,4C
0B3F:013D B36C            MOV     BL,6C
0B3F:013F CD21            INT     21
```

2.2.3.3 Символ

```
0B3F:0103 8A160001      MOV     DL,[0100]
```

2.2.3.4 Код помилки

```
0B3F:013D B36C          MOV     BL,6C
```

2.2.4 Результат виконання

```
Z:\ASM\TASM>TASM.EXE comNo.asm
Turbo Assembler  Version 3.2i  Copyright (c) 1988, 1992 Borland International
Serial No:   Tester:

Assembling file:   comNo.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  455k


Z:\ASM\TASM>TLINK.EXE /t COMNO.OBJ
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International

Z:\ASM\TASM>COMNO.COM
Ukraine
```

2.3 COM Програма з модульними процедурами

2.3.1 Код програми

```
.model tiny     ; set program model as tiny

cseg segment para public 'code'     ; Declare code segment
    assume cs:cseg, ds:cseg, ss:cseg, es:nothing    ; set each code
segment to code seg as it is the only segment in the program
    org 100h    ; start loading the first instruction at 100h

start:  ; declare program entry point
    call main   ; call main function

    mov ah, 04Ch    ; exit to OS
    mov bl, 6Ch     ; set error code to 108 in hex
    int 21h     ; call interrupt

    symbol db 'U'   ; declare a symbol to output
```

```
main proc near  ; declare main procedure
    call newline    ; call newline function

    xor ax, ax   ; clear ax register
    mov dl, symbol      ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'k'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'r'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'a'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'i'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'n'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    mov dl, 'e'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    call newline    ; call newline function

    mov dl, '<'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
```

```
    int 21h     ; call interrupt

    mov dl, '3'     ; load symbol to stdout
    mov ah, 02h     ; output symbol to stdout
    int 21h     ; call interrupt

    call newline    ; call newline function

    ret ; end function execution
main endp   ; end procedure

newline proc near   ; declare modular procedure
    mov dl, 10      ; set dl register to new line
    mov ah, 02h     ; output it to stdout
    int 21h     ; call interrupt
    mov dl, 13      ; set dl register to carret return
    mov ah, 02h     ; output it to stdout
    int 21h     ; call interrupt

    ret ; end function execution
newline endp    ; end procedure


cseg ends   ; close segment
end start   ; end program execution
```

2.3.2 Дизасемблювання

```
Z:\ASM\TASM>debug comyes.com
-u
0B3F:0100 E80700          CALL    010A
0B3F:0103 B44C            MOV     AH,4C
0B3F:0105 B36C            MOV     BL,6C
0B3F:0107 CD21            INT     21
0B3F:0109 55              PUSH    BP
0B3F:010A E84100          CALL    014E
0B3F:010D 33C0            XOR     AX,AX
0B3F:010F 8A160901        MOV     DL,[0109]
0B3F:0113 B402            MOV     AH,02
0B3F:0115 CD21            INT     21
0B3F:0117 B26B            MOV     DL,6B
0B3F:0119 B402            MOV     AH,02
0B3F:011B CD21            INT     21
0B3F:011D B272            MOV     DL,72
0B3F:011F B402            MOV     AH,02
-u
0B3F:0121 CD21            INT     21
0B3F:0123 B261            MOV     DL,61
0B3F:0125 B402            MOV     AH,02
0B3F:0127 CD21            INT     21
0B3F:0129 B269            MOV     DL,69
0B3F:012B B402            MOV     AH,02
0B3F:012D CD21            INT     21
0B3F:012F B26E            MOV     DL,6E
0B3F:0131 B402            MOV     AH,02
0B3F:0133 CD21            INT     21
0B3F:0135 B265            MOV     DL,65
0B3F:0137 B402            MOV     AH,02
0B3F:0139 CD21            INT     21
0B3F:013B E81000          CALL    014E
0B3F:013E B23C            MOV     DL,3C
0B3F:0140 B402            MOV     AH,02
```

```
-u
0B3F:0142 CD21          INT     21
0B3F:0144 B233          MOV     DL,33
0B3F:0146 B402          MOV     AH,02
0B3F:0148 CD21          INT     21
0B3F:014A E80100        CALL    014E
0B3F:014D C3            RET
0B3F:014E B20A          MOV     DL,0A
0B3F:0150 B402          MOV     AH,02
0B3F:0152 CD21          INT     21
0B3F:0154 B20D          MOV     DL,0D
0B3F:0156 B402          MOV     AH,02
0B3F:0158 CD21          INT     21
0B3F:015A C3            RET
0B3F:015B 3C02          CMP     AL,02
0B3F:015D 7506          JNZ     0165
0B3F:015F 26            ES:
0B3F:0160 895504        MOV     [DI+04],DX
```

2.3.3 Вказівки

2.3.3.1 PSP

```
-d 0 100
0B3F:0000  CD 20 FF 9F 00 9A F0 FE-1D F0 4F 03 54 05 8A 03   . ........O.T...
0B3F:0010  54 05 17 03 54 05 43 05-01 01 01 00 02 FF FF FF   T...T.C.........
0B3F:0020  FF FF FF FF FF FF FF FF-FF FF FF FF 01 0B 4C 01   ..............L.
0B3F:0030  14 0A 14 00 18 00 3F 0B-FF FF FF FF 00 00 00 00   ......?.........
0B3F:0040  05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0B3F:0050  CD 21 CB 00 00 00 00 00-00 00 00 00 00 20 20 20   .!...........
0B3F:0060  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20            .....
0B3F:0070  20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00            ........
0B3F:0080  00 0D 63 6F 6D 79 65 73-2E 63 6F 6D 0D 54 45 52   ..comyes.com.TER
0B3F:0090  3D 41 30 0D 64 64 72 65-73 73 2E 20 20 46 6F 72   =A0.ddress.  For
0B3F:00A0  20 65 78 61 6D 70 6C 65-3A 0D 20 6F 6E 20 4E 54    example:. on NT
0B3F:00B0  56 44 4D 2C 20 73 70 65-63 69 66 79 20 61 6E 20   VDM, specify an
0B3F:00C0  69 6E 76 61 6C 69 64 0D-20 6F 6E 6C 79 2E 0D 00   invalid. only...
0B3F:00D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0B3F:00E0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0B3F:00F0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0B3F:0100  E8                                                .
```

2.3.4.2 Segments

```
Z:\ASM\TASM>debug comyes.com
-u
0B3F:0100 E80700          CALL    010A
0B3F:0103 B44C            MOV     AH,4C
0B3F:0105 B36C            MOV     BL,6C
0B3F:0107 CD21            INT     21
0B3F:0109 55              PUSH    BP
0B3F:010A E84100          CALL    014E
0B3F:010D 33C0            XOR     AX,AX
0B3F:010F 8A160901        MOV     DL,[0109]
0B3F:0113 B402            MOV     AH,02
0B3F:0115 CD21            INT     21
0B3F:0117 B26B            MOV     DL,6B
0B3F:0119 B402            MOV     AH,02
0B3F:011B CD21            INT     21
0B3F:011D B272            MOV     DL,72
0B3F:011F B402            MOV     AH,02
-u
0B3F:0121 CD21            INT     21
0B3F:0123 B261            MOV     DL,61
0B3F:0125 B402            MOV     AH,02
0B3F:0127 CD21            INT     21
0B3F:0129 B269            MOV     DL,69
0B3F:012B B402            MOV     AH,02
0B3F:012D CD21            INT     21
0B3F:012F B26E            MOV     DL,6E
0B3F:0131 B402            MOV     AH,02
0B3F:0133 CD21            INT     21
0B3F:0135 B265            MOV     DL,65
0B3F:0137 B402            MOV     AH,02
0B3F:0139 CD21            INT     21
0B3F:013B E81000          CALL    014E
0B3F:013E B23C            MOV     DL,3C
0B3F:0140 B402            MOV     AH,02
```

2.3.3.3 Символ

```
0B3F:010F 8A160901        MOV     DL,[0109]
```

2.3.3.4 Код помилки

```
0B3F:0105 B36C            MOV     BL,6C
```

2.3.4 Результат виконання

```
Z:\ASM\TASM>TASM.EXE comYes.asm
Turbo Assembler  Version 3.2i  Copyright (c) 1988, 1992 Borland International
Serial No:   Tester:

Assembling file:   comYes.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  455k


Z:\ASM\TASM>TLINK.EXE /t COMYES.OBJ
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International

Z:\ASM\TASM>COMYES.COM

Ukraine
<3
```

2.4 Виведення кодів помилок

```
Z:\ASM\TASM>test.bat
EXE Program:
X88
---------------
COM Program without Function:
Ukraine
13
---------------
COM Program with Function:

Ukraine
<3
13
```

2.5 BAT файл

```
@echo off
echo EXE Program:
exe.exe
echo %ERRORLEVEL%
echo ---------------
echo COM Program without Function:
comNo.com
echo %ERRORLEVEL%
echo ---------------
echo COM Program with Function:
comYes.com
echo %ERRORLEVEL%
```

3 Висновки

Таким чином, ми ознайомились зі структурою програм на мові асемблера і навели приклад її застосування