

Лабораторна робота №3

Тема: «Команди умовного та безумовного переходу. Цикли»

Мета роботи: навчитися використовуючи команди умовного та безумовного переходу, а також навчитися будувати циклічні структури.

Хід роботи

Завдання на 3 бали

Приклад 1

Написати програму складання двох чисел зі знаком з перевіркою переповнення. У разі переповнення буде вивести повідомлення про помилку.

Рішення

1. Відкомпілювати та відлагодити приклади представлені у роботі. Пояснити їх роботу, стани реєстрів та принципи роботи команд.

```

use16; Генерувати 16-бітний код
org 100h; Програма починається з адреси 100h
    mov al, [x]; AL = x
    add al, [y]; AL = x + y
    jo error; Перехід, якщо переповнення
    mov ah, 09h; \
    mov dx, ok_msg; > Виведення рядка 'OK'
    int 21h; /
exit:
    mov ah, 09h; \
    mov dx, pak; > Виведення рядка 'Press any key ...'
    int 21h; /
    mov ah, 08h; \
    int 21h; / Введення символу
    mov ax, 4C00h; \
    int 21h; / Завершення програми
error:
    mov ah, 09h; \
    mov dx, err_msg; >повідомлення про помилку
    int 21h; /
    jmp exit; Перехід на мітку exit
;
-----  

x db -89
y db -55
err_msg db 'Error: overflow detected.', 13,10, '$'
ok_msg db 'OK', 13,10, '$'
pak db 'any', 13,10, '$'

```

```
C:\>td first_project.com
Turbo Debugger Version 5.0 Copyright (c) 1988,96 Borland International
Error: overflow detected.
any
```

Робота програми

Приклад 2

Створити програму, яка виводить меню і пропонує користувачеві зробити вибір. Для введення символу використовувати функцію DOS 01h (при введенні символ відображається на екрані).

Рішення

```
use16; Генерувати 16-бітний код
org 100h; Програма починається з адреси 100h
jmp start; Безумовний перехід на мітку start
; -----
menu db '1 - Print hello', 13,10
        db '2 - Print go away', 13,10
        db '0 - Exit', 13,10, '$'
select db 13,10, 'Select> $'
hello db 13,10, 'Hello!', 13,10,13,10, '$'
go_away db 13,10, 'Go away!', 13,10,13,10, '$'
; -----
start:
    mov ah, 09h; \
    mov dx, menu; > Виведення меню
    int 21h; /
select_loop:
    mov ah, 09h; \
    mov dx, select; > Виведення рядка 'Select> '
    int 21h; /

    mov ah, 01h; Функція DOS 01h - введення символу
    int 21h; Введений символ поміщається в AL

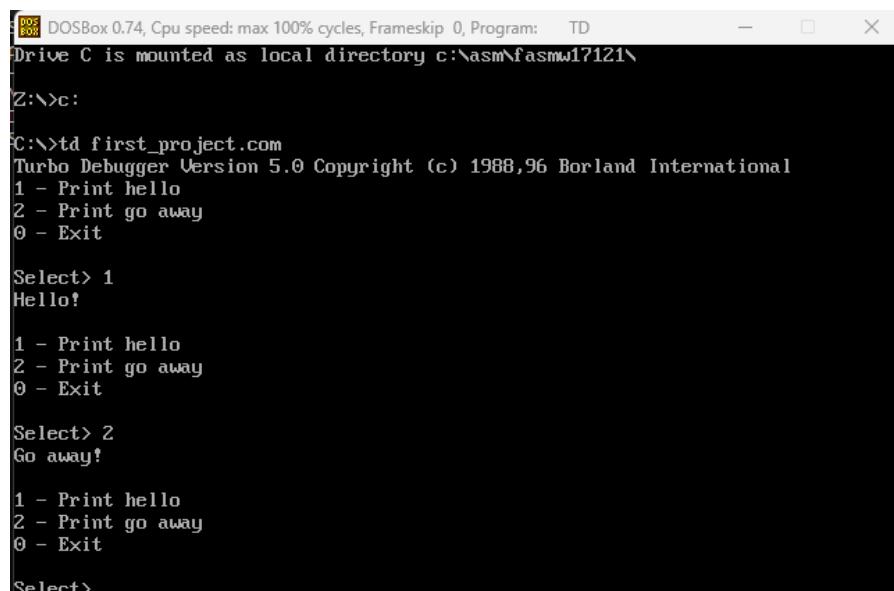
    cmp al, '1'; Порівняння введеного символу з '1'
    je c1; Перехід, якщо дорівнює
    cmp al, '2'; Порівняння введеного символу з '2'
    je c2; Перехід, якщо дорівнює
    cmp al, '0'; Порівняння введеного символу з "0"
    je exit; Перехід, якщо дорівнює
    jmp select_loop; Безумовний перехід

c1:
    mov ah, 09h; \
```

```

    mov dx, hello; > Виведення рядка 'Hello'
    int 21h; /
    jmp start; Безумовний перехід
c2:
    mov ah, 09h; \
    mov dx, go_away; > Виведення рядка 'Go away'
    int 21h; /
    jmp start; Безумовний перехід
exit:
    mov ax, 4C00h; \
    int 21h; / Завершення програми

```



Робота програми

Приклад 3

Написати програму для порівняння двох змінних зі знаком а і б. Залежно від результатів порівняння вивести « $a < b$ », « $a > b$ » або « $a = b$ ».

Рішення

```

use16
org 100h
jmp start
;_____
    a db -55
    b db -23
    bolshe db 'a>b$'
    menshe db 'a<b$'
    ravno db 'a=b$'
;_____
start:

```

```

    mov ah, [a]
    mov al, [b]
    cmp ah, al
    jg a_bolshe_b
    jl a_menshe_b
    jz a_ravno_b

a_bolshe_b:
    mov ah, 09
    mov dx, bolshe
    int 21h
    jmp exit

a_menshe_b:
    mov ah, 09
    mov dx, menshe
    int 21h
    jmp exit

a_ravno_b:
    mov ah, 09
    mov dx, ravno
    int 21h

exit:
    mov ah, 08h
    int 21h
    mov ax, 4c00h
    int 21h

```

```
C:\>td first_project.com
Turbo Debugger Version 5.0 Copyright (c) 1988,96 Borland International
a<b_
```

```
C:\>td first_project.com
Turbo Debugger Version 5.0 Copyright (c) 1988,96 Borland International
a>b
```

```
C:\>td first_project.com
Turbo Debugger Version 5.0 Copyright (c) 1988,96 Borland International
a=b_
```

Робота програми

Приклад 4

Вивести всі букви англійського алфавіту.

Рішення

ASCII-коди цих символів розташовані послідовно, тому можна виводити їх в циклі. Для виведення символу на екран використаємо функцію DOS 02h (виведений байт повинен знаходитися в реєстрі DL).

```

use16; Генерувати 16-бітний код
org 100h; Програма починається з адреси 100h
    mov ah, 02h; Для виклику функції DOS 02h - виведення
    символу
        mov dl, 'A'; Перший виведений символ
        mov cx, 26; Лічильник повторень циклу
metka:
    int 21h; Звернення до функції DOS
    inc dl; Наступний символ
    loop metka; Команда циклу
    mov ah, 09h; Функція DOS 09h - виведення рядка
    mov dx, press; В DX адреса рядка
    int 21h; Звернення до функції DOS
    mov ah, 08h; Функція DOS 08h - введення символу без
    луни
    int 21h; Звернення до функції DOS
    mov ax, 4C00h; \
    int 21h; / Завершення програми
;
-
press:
    db 13,10, 'Press any key ... $'

```

```

C:\>td first_project.com
Turbo Debugger Version 5.0 Copyright (c) 1988,96 Borland International
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Press any key ... _

```

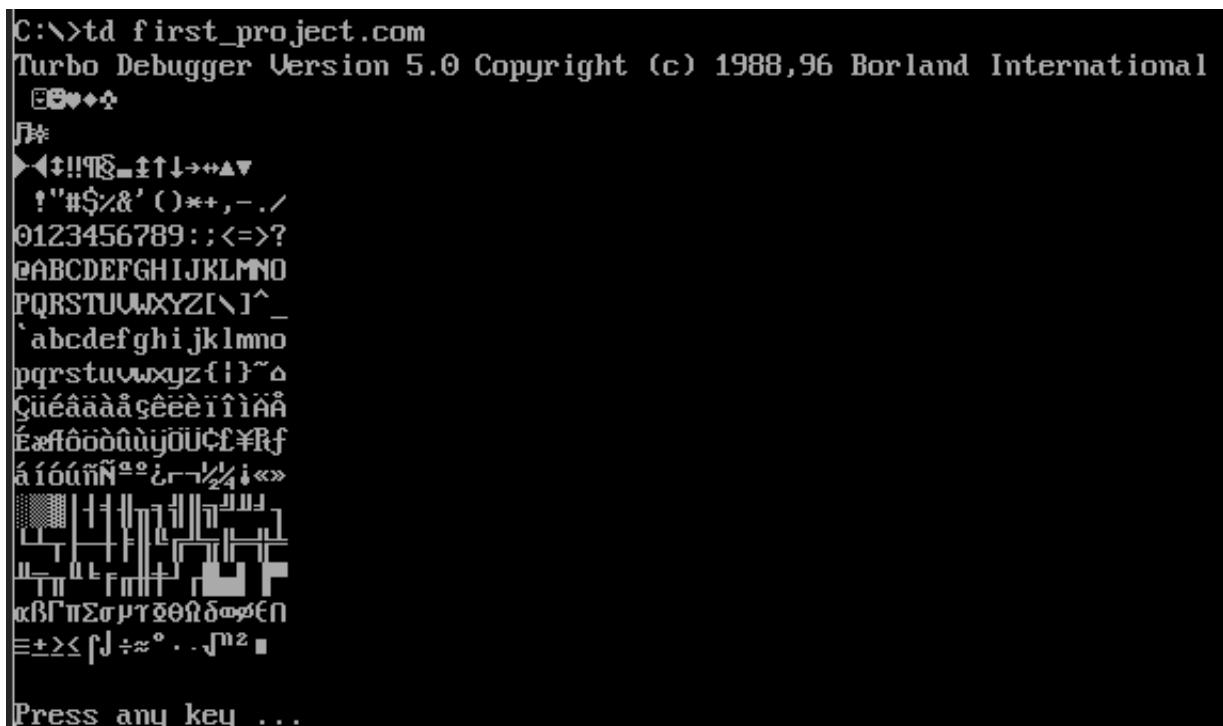
Робота програми

```

use16; Генерувати 16-бітний код
org 100h; Програма починається з адреси 100h
    mov ah, 02h; Для виклику функції DOS 02h - виведення
    символу
        sub dl, dl; Перший виведений символ
        mov cx, 16; Лічильник зовнішнього циклу (по рядках)
lp1:
    mov bx, cx; Зберігаємо лічильник в BX
    mov cx, 16; Лічильник внутрішнього циклу (по
    стовпцях)
lp2:
    int 21h; Звернення до функції DOS
    inc dl; Наступний символ
    loop lp2; Команда внутрішнього циклу

```

```
mov dh, dl; Зберігаємо значення DL в DH
mov dl, 13; \
int 21h; \
mov dl, 10; / Перехід на наступний рядок
int 21h; /
mov dl, dh; Відновлюємо значення DL
mov cx, bx; Відновлюємо значення лічильника
loop lpl; Команда зовнішнього циклу
mov ah, 09h; Функція DOS 09h - висновок рядка
mov dx, press; В DX адреса рядка
int 21h; Звернення до функції DOS
mov ah, 08h; Функція DOS 08h - введення символу без
луни
int 21h; Звернення до функції DOS
mov ax, 4C00h; \
int 21h; / Завершення програми
;
-----  
press db 13,10, 'Press any key ... $'
```



Робота програми

Завдання на 4 бали

1. Виконати завдання на 3 бали.
 2. Виконати завдання згідно свого варіанту із таблиці 3.1

6,12,18,24	Створити міні калькулятор на 4 основні дії.
-------------------	--

```
use16
org 100h

jmp start
; -----
first_menu db 13,10,'a=', '$'
second_menu db 13,10,'b=', '$'
; -----
menu db 13,10,'+ - Add numbers', 13,10
      db ' - Subtract numbers', 13,10
      db '* - Multiply numbers', 13,10
      db '/ - Divide the numbers', 13,10
      db '0 - Exit', 13,10, '$'
select db 13,10, 'Select> $'
result db 13,10, 'Result= $'
err_div0 db 13,10,'Error: division by zero!$'
; -----
a db ?
b db ?
res dw ?
buffer db 6 dup(?)
; -----
start:
    mov ah, 09h
    mov dx, first_menu
    int 21h

    mov ah, 01h
    int 21h
    sub al, '0'
    mov [a], al

    mov ah, 09h
    mov dx, second_menu
    int 21h

    mov ah, 01h
    int 21h
    sub al, '0'
    mov [b], al

show_menu:
```

```

        mov ah, 09h
        mov dx, menu
        int 21h

select_loop:
        mov ah, 09h
        mov dx, select
        int 21h

        mov ah, 01h
        int 21h

        cmp al, '+'
        je c1
        cmp al, '-'
        je c2
        cmp al, '*'
        je c3
        cmp al, '/'
        je c4
        cmp al, '0'
        je exit
        jmp select_loop

; додавання
c1:
        mov al, [a]
        add al, [b]
        movzx ax, al
        mov [res], ax

        mov ah, 09h
        mov dx, result
        int 21h

        mov ax, [res]
        mov bx, 10
        mov si, 0

c1_conv:
        xor dx, dx
        div bx

```

```

        add dl, '0'
        mov [buffer+si], dl
        inc si
        test ax, ax
        jnz c1_conv

c1_print:
        dec si
        mov dl, [buffer+si]
        mov ah, 02h
        int 21h
        test si, si
        jnz c1_print
        jmp start

; віднімання
c2:
        mov al, [a]
        sub al, [b]
        mov [res], ax
        mov ah, 09h
        mov dx, result
        int 21h
        mov ax, [res]
        add al, '0'
        mov dl, al
        mov ah, 02h
        int 21h
        jmp start

; множення
c3:
        mov al, [a]
        mov bl, [b]
        mul bl
        mov [res], ax

        mov ah, 09h
        mov dx, result
        int 21h

        mov ax, [res]

```

```

        mov bx, 10
        mov si, 0

c3_conv:
        xor dx, dx
        div bx
        add dl, '0'
        mov [buffer+si], dl
        inc si
        test ax, ax
        jnz c3_conv

c3_print:
        dec si
        mov dl, [buffer+si]
        mov ah, 02h
        int 21h
        test si, si
        jnz c3_print
        jmp start

; ділення
c4:
        mov al, [b]
        cmp al, 0
        je div_error

        mov al, [a]
        mov bl, [b]
        xor ah, ah
        div bl
        mov [res], ax
        mov ah, 09h
        mov dx, result
        int 21h
        mov ax, [res]
        add al, '0'
        mov dl, al
        mov ah, 02h
        int 21h
        jmp start

```

```

div_error:
    mov ah, 09h
    mov dx, err_div0
    int 21h
    jmp start

exit:
    mov ax, 4C00h
    int 21h

```

```

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD
C:\>td first_project.com
Turbo Debugger Version 5.0 Copyright (c) 1988,96 Borland International

a=8
b=9
+ - Add numbers
- - Subtract numbers
* - Multiply numbers
/ - Divide the numbers
0 - Exit

Select> +
Result= 17
a=4
b=1
+ - Add numbers
- - Subtract numbers
* - Multiply numbers
/ - Divide the numbers
0 - Exit

Select> -
Result= 3
a=6
b=7
+ - Add numbers
- - Subtract numbers
* - Multiply numbers
/ - Divide the numbers
0 - Exit

Select> *
Result= 42
a=9
b=3
+ - Add numbers
- - Subtract numbers
* - Multiply numbers
/ - Divide the numbers
0 - Exit

Select> /
Error: division by zero!
a=6
b=0
+ - Add numbers
- - Subtract numbers
* - Multiply numbers
/ - Divide the numbers
0 - Exit

Select> *
Result= 0

```

Робота програми

Завдання на 5 балів

- Виконати завдання на 3 та на 4 бали.
- Виконати завдання згідно свого варіанту із таблиці 3.2

6,12,18,24	Перемножити кожен елемент масиву на мінімальний елемент масиву
------------	---

```
use16
org 100h

jmp start
; -----
arr db 5, 12, 3, 7, 9, 2, 8
min db ?
; -----
```

start:

```
    mov si, 0
    mov al, [arr+si]
    mov [min], al
```

find_min:

```
    inc si
    cmp si, 7
    je found_min

    mov bl, [arr+si]
    cmp bl, [min]
    jge find_min
    mov [min], bl
    jmp find_min
```

found_min:

```
    mov si, 0
```

mul_loop:

```
    cmp si, 7
    je exit
    cbw
    mov al, [arr+si]
    mov bl, [min]
    cbw
    imul bl
    mov [arr+si], al
    inc si
    jmp mul_loop
```

exit:

```
    mov ax, 4C00h
    int 21h
```

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

[CPU 80486]

cs:0100 EB08	jmp	010A ↓	ax 0000	c=0
cs:0102 050003	add	ax,0300	bx 0000	z=0
cs:0105 07	pop	es	cx 0000	s=0
cs:0106 0902	or	[bp+sil],ax	dx 0000	o=0
cs:0108 0800	or	[bx+sil],al	si 0000	p=0
cs:010A BE0000	mov	si,0000	di 0000	a=0
cs:010D 8A840201	mov	al,[si+0102]	bp 0000	i=1
cs:0111 A20901	mov	[0109],al	sp FFFE	d=0
cs:0114 46	inc	si	ds 0AB7	
cs:0115 83FE06	cmp	si,0006	es 0AB7	
cs:0118 7410	je	012A	ss 0AB7	
cs:011A 8A9C0201	mov	bl,[si+0102]	cs 0AB7	
cs:011E 3A1E0901	cmp	bl,[0109]	ip 0100	
ds:0102 05 0C 03 07 09 02 08 00 0F•••08				
ds:010A BE 00 00 8A 84 02 01 A2 ↴ èä•••6				
ds:0112 09 01 46 83 FE 06 74 10 0F•••t▶				
ds:011A 8A 9C 02 01 3A 1E 09 01 èf•••:•o			ss:0000 20CD	
			ss:FFFE>0000	

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

[CPU 80486]

0AB7:0100 EB08	jmp	010A	ax 0959	c=0
0AB7:0102 0A18	or	bl,[bx+sil]	bx 0FA8	z=0
0AB7:0104 06	push	es	cx F000	s=0
0AB7:0105 0E	push	cs	dx 3046	o=0
0AB7:0106 1204	adc	al,[sil]	si 0098	p=0
0AB7:0108 1002	adc	[bp+sil],al	di E98F	a=0
0AB7:010A BE0000	mov	si,0000	bp 0314	i=0
0AB7:010D 8A840201	mov	al,[si+0102]	sp 031A	d=0
0AB7:0111 A20901	mov	[0109],al	ds F000	
0AB7:0114 46	inc	si	es 3002	
0AB7:0115 83FE07	cmp	si,0007	ss 0959	
0AB7:0118 7410	je	012A	cs 0000	
0AB7:011A 8A9C0201	mov	bl,[si+0102]	ip 0000	
0AB7:0102 0A 18 06 0E 12 04 10 02 0t•••••••8				
0AB7:010A BE 00 00 8A 84 02 01 A2 ↴ èä•••6				
0AB7:0112 09 01 46 83 FE 07 74 10 0F•••t▶			0AB7:0000 20CD	
0AB7:011A 8A 9C 02 01 3A 1E 09 01 èf•••:•o			0AB7:FFFE 0000	

Робота програми