

Міністерство освіти і науки України  
Національний технічний університет України  
“Київський політехнічний інститут імені Ігоря Сікорського”  
Факультет інформатики та обчислювальної техніки  
Кафедра автоматики та управління в технічних системах

## Лабораторна робота 7

### Архітектура комп'ютера

*“Підпрограми архітектури IA-32 (x86) у Real Adress Mode”*

Виконали:  
Студенти групи IT-01

Тимошенко Олексій  
Тонкий Михайло

Перевірів:

Бердник Ю.М.

Київ 2021

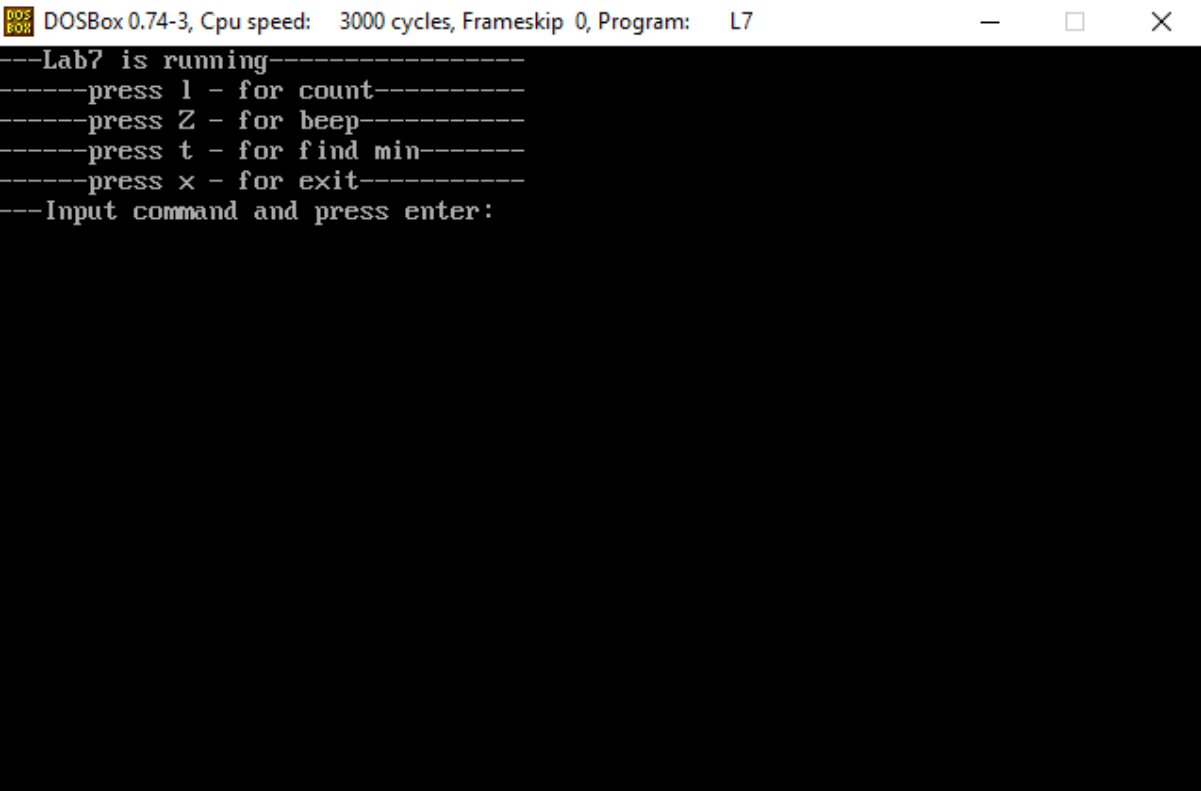
**Тема:** Підпрограми архітектури IA-32 (x86) у Real Adress Mode.

**Мета:** отримати основні навички розробки підпрограм.

**Github:** <https://github.com/OlexiiT/asembler/tree/main/L7>  
<https://github.com/Mikuyoki/Asembler/tree/main/L7>

### Хід роботи:

У даній роботі ми створили інтерфейс користувача відповідно до варіанту, та виглядає він так:

A screenshot of a DOSBox window. The title bar reads "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: L7". The window contains a black terminal area with white text. The text reads: "--Lab7 is running-----", "--press l - for count-----", "--press Z - for beep-----", "--press t - for find min-----", "--press x - for exit-----", and "--Input command and press enter:". The rest of the terminal area is empty.

```
--Lab7 is running-----
--press l - for count-----
--press Z - for beep-----
--press t - for find min-----
--press x - for exit-----
--Input command and press enter:
```

Ця програма виконується у загальному циклі. У разі, коли користувач вводить одну з запропонованих йому літер, програма здійснює виклик відповідної функції та після її виконання повертається у початок циклу (крім команди exit), якщо ж буква є невірною цикл повторюється одразу.

```

;-----
Main_cycle:
    call display_menu
    call input
    cmp ax, 06ch
    je Count
    cmp ax, 05ah
    je Beep
    cmp ax, 74h
    je Find
    cmp ax, 078h
    je Exit
    jmp Main_cycle

Count:
    call calc
    jmp Main_cycle

Beep:
    mov dx, offset display_message_wait
    call display_foo
    call sound
    jmp Main_cycle

Find:
    call search
    jmp Main_cycle

Exit:
    mov dx, offset display_message_finish
    call display_foo
    mov ax, 04c00h
    int 21h

```

;Головний цикл, що виконується вродовж всього часу роботи програми  
;Виклик функції (процедури) для друкування меню  
;Виклик функції, що приймає значення символу з клавіатури  
;Порівнюємо значення введеного символу з літерою l  
;Порівнюємо значення введеного символу з літерою Z  
;Порівнюємо значення введеного символу з літерою t  
;Порівнюємо значення введеного символу з літерою x  
;Перехід до коду виходу з програми  
;Коли жодна буква не відповідає команді, повертаємося в початок циклу  
;Виклик функції, що виконує обчислення виразу  
;Виклик функції, що відтворює звуковий сигнал  
;Виклик функції, що шукає найменше значення у масиві

Також у циклі можна помітити ще три допоміжні функції. display\_foo відповідає за вивід даних з регістру dx на екран. Виглядає ж вона так:

```

PROC display_foo
    mov ah, 09h
    int 21h
    xor dx, dx
    ret
ENDP display_foo

```

Функція display\_menu виводить на екран меню, що пропонує обрати певну команду.

```

;-----
newline db 13,10,'$'
result db "Result: $"
min_message db "The min value is: $"

system_message_1 db "----Input command and press enter: ", '$'
system_message_2 db "-----Press any key to continue-----", 13,10,'$'

display_message_start db "----Lab7 is running-----", 13,10,'$'
display_message_count db "-----press 1 - for count-----", 13,10,'$'
display_message_beep db "-----press Z - for beep-----", 13,10,'$'
display_message_find db "-----press t - for find min-----", 13,10,'$'
display_message_exit db "-----press x - for exit-----", 13,10,'$'
display_message_finish db "----Lab7 finished-----", 13,10,'$'
display_message_wait db "-----", 13,10,'$'

```

;Повідомлення для виводу інформації

```

PROC display_menu                                     ;функція виводу меню на екран користувача
mov ax, 00003h                                       ;Команда очищення екрану
int 10h                                              ;Переривання BIOS
mov dx, offset display_message_start               ;Виводимо основний текст
call display_foo
mov dx, offset display_message_count
call display_foo
mov dx, offset display_message_beep
call display_foo
mov dx, offset display_message_find
call display_foo
mov dx, offset display_message_exit
call display_foo
mov dx, offset system_message_1
call display_foo
ret
ENDP display_menu

```

Додатково ця функція очищає екран кожний раз під час свого виклику.

І, нарешті, функція input повертає нам у регістр ax значення введеної літери у вигляді її коду ASCII.

```

PROC input
mov ah, 0ah
mov dx, offset string
int 21h

xor ax, ax
mov bx, offset string
mov ax, [bx+1]
shr ax, 8
ret
ENDP input

```

Розгляньмо тепер поближче функції, які може викликати користувач. Почнемо з функції count. Її завданням є підрахунок заданого нам виразу із заданими значеннями, а також вивід відповіді на екран. Її реалізація виглядає так:

```
PROC calc                                     ;(((a1 - a2) + a3) / a4 * a5)      ; -2, 3, 1, 2, 3
mov dx, offset newline                       ;Переносимо каретку на новий рядок
call display_foo
mov dx, offset result                         ;Виводимо на екран рядок
call display_foo
mov ax, [a1]                                 ;ax <- a1          (ax <- -2)
mov cx, [a2]                                 ;cx <- a2          (ax <- 3)
sub ax, cx                                   ;ax <- ax - cx     (ax <- -2 - 3)
mov cx, [a3]                                 ;cx <- a3          (ax <- 1)
add ax, cx                                   ;ax <- ax + cx     (ax <- -5 + 1)
mov cx, [a4]                                 ;cx <- a4          (ax <- 2)
idiv cx                                      ;al <- ax / cx     (al <- -4 / 2)
mov cx, [a5]                                 ;cx <- a5          (ax <- 3)
imul cx                                      ;ax <- al * cx     (ax <- -2 * 3)
mov bx, ax                                   ;bx <- ax
neg bx                                       ;bx <- -bx
cmp ax, bx                                  ;Порівнюємо значення ax і bx, таким чином дізнаємося знак результату
jb print

mov ax, bx                                   ;Якщо результат негативний записуємо у регістр ax модуль результату
mov ah, '-'                                  ;У старший біт записуємо знак "-"
add al, 30h                                  ;Щоб отримати з числа код числа (у ASCII) додаємо 30h

print:
mov dl, ah                                   ;Зберігаємо знак
mov dh, al                                   ;Зберігаємо число
mov ah, 02h                                  ;Виводимо на екран знак
int 21h
mov dl, dh                                   ;Виводимо на екран число
int 21h

mov dx, offset newline                       ;Переносимо каретку на новий рядок
call display_foo

mov dx, offset system_message_2              ;Виводимо повідомлення
call display_foo

mov ah, 08h                                  ;Чекаємо натискання клавіші
int 21h

ret
ENDP calc
```

А результат роботи так:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: L7
Lab? is running-----
press 1 - for count-----
press Z - for beep-----
press t - for find min-----
press x - for exit-----
Input command and press enter: 1
Result: -6
Press any key to continue-----
```

Далі йде функція sound, що відтворює звук:

```
PROC sound                                     ;функція, що відтворює звуковий сигнал

    in al, 61h                                ;Одержуємо стан динаміка
    or al, 3                                  ;Змінюємо стан на увімкнений
    out 61h, al                               ;Зберігаємо цей стан

    mov al, 10110110B                         ;Встановлюємо частоту звука
    out 43h, al                               ;Вмикаємо звук

    mov ax, 2700                              ;Викликаємо функцію затримки часу
    out 42h, al

    call wait_t

    in al, 61h                                ;Одержуємо стан динаміка
    and al, 11111100B                         ;Змінюємо стан на вимкнений
    out 61h, al                               ;Зберігаємо цей стан
    ret
ENDP sound
```

В ній також використовується метод wait\_t, що створює затримку на необхідний проміжок часу.

```
PROC wait_t                                   ;функція, що створює затримку у часі
                                           ;за допомогою вкладених циклів
    push cx
    mov cx, TIME
    w1:                                     ;Перший цикл
        push cx
        mov cx, TIME
        w2:                               ;Другий цикл
            loop w2
        pop cx
        loop w1
    pop cx
    ret
ENDP wait_t
```

Результат роботи: функція починає відтворювати звук)

Функція пошуку найменшого елементу search проходиться по всьому масиву, записуючи у регістр al найменше значення та виводить його на екран:

```

array2Db      db 2h, 2h, 9h, 9h, 5h, 5h, 3h, 8h, 4h, 7h, 2h, 5h, 6h, 4h, 6h, 8h
               db 2h, 2h, 6h, 9h, 8h, 9h, 9h, 6h, 7h, 7h, 2h, 7h, 6h, 3h, 9h, 7h
               db 2h, 3h, 6h, 8h, 2h, 3h, 5h, 3h, 9h, 5h, 4h, 6h, 9h, 8h, 3h, 3h
               db 3h, 5h, 4h, 4h, 3h, 5h, 8h, 5h, 2h, 9h, 2h, 4h, 2h, 9h, 2h, 9h
               db 5h, 3h, 6h, 5h, 2h, 9h, 8h, 9h, 9h, 6h, 2h, 7h, 6h, 4h, 2h, 6h
               db 2h, 6h, 4h, 8h, 8h, 6h, 9h, 5h, 4h, 4h, 6h, 4h, 4h, 3h, 5h, 7h
               db 9h, 9h, 7h, 9h, 5h, 6h, 9h, 5h, 4h, 4h, 4h, 5h, 4h, 3h, 7h, 4h
               db 6h, 9h, 5h, 6h, 8h, 7h, 6h, 3h, 2h, 9h, 3h, 4h, 5h, 2h, 6h, 5h
               db 3h, 6h, 4h, 9h, 6h, 5h, 3h, 9h, 2h, 2h, 3h, 7h, 3h, 2h, 4h, 3h
               db 5h, 6h, 2h, 9h, 3h, 5h, 6h, 9h, 6h, 8h, 6h, 4h, 8h, 3h, 7h, 5h
               db 9h, 2h, 4h, 4h, 3h, 6h, 3h, 3h, 4h, 9h, 5h, 3h, 2h, 6h, 6h, 5h
               db 7h, 9h, 4h, 9h, 6h, 8h, 4h, 7h, 2h, 8h, 7h, 4h, 8h, 3h, 5h, 5h
               db 4h, 5h, 3h, 2h, 5h, 5h, 9h, 2h, 4h, 4h, 2h, 7h, 4h, 8h, 6h, 7h
               db 3h, 3h, 2h, 3h, 4h, 4h, 3h, 2h, 6h, 8h, 6h, 7h, 2h, 9h, 8h, 4h
               db 4h, 6h, 4h, 4h, 7h, 9h, 4h, 8h, 3h, 5h, 4h, 6h, 8h, 8h, 8h, 8h
               db 9h, 4h, 4h, 9h, 8h, 4h, 9h, 3h, 9h, 8h, 5h, 8h, 5h, 8h, 8h, 5h

PROC search                                         ;функція пошуку найменшого значення у масиві
mov al, [array2Db]
mov cx, 255
xor si, si
loop1:                                             ;У циклі порівнюємо кожне наступне значення з найменшим з попередніх
inc si                                           ;та якщо наступне - менше, перезаписуємо його
mov ah, [array2Db+si]
cmp al, ah
jb outer
mov al, ah

outer:
loop loop1

mov dx, offset newline                           ;Переходимо на новий рядок
call display_foo

mov dx, offset min_message                       ;Виводимо повідомлення про результат
call display_foo

mov dl, al
add dl, 30h
mov ah, 02h
int 21h

mov dx, offset newline                           ;Переходимо на новий рядок
call display_foo

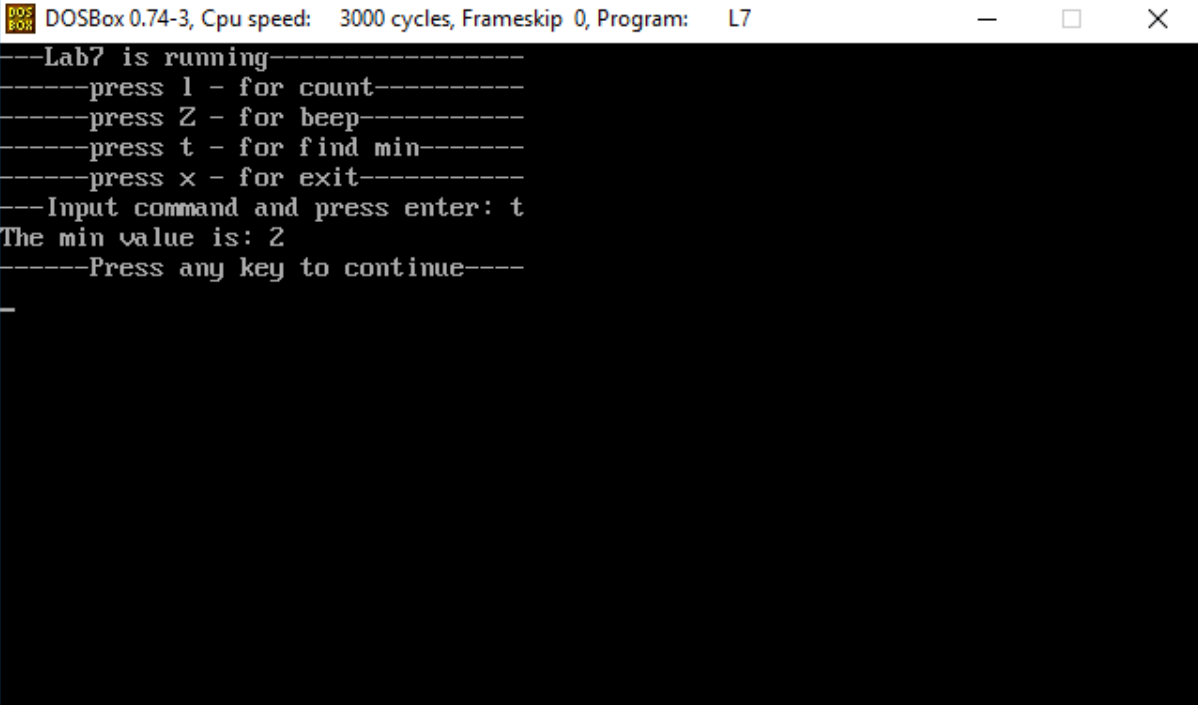
mov dx, offset system_message_2                 ;Виводимо повідомлення про очікування
call display_foo

mov ah, 08h                                     ;Чекаємо натискання клавіші
int 21h

ret
ENDP search

```

## Результат пошуку:



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: L7
---Lab7 is running-----
---press l - for count-----
---press Z - for beep-----
---press t - for find min-----
---press x - for exit-----
---Input command and press enter: t
The min value is: 2
---Press any key to continue---
```

І нарешті, частина коду з міткою Exit, що виконує вивід на екран повідомлення про завершення роботи програми та завершує її:

```
Exit:
    mov dx, offset display_message_finish
    call display_foo
    mov ax, 04c00h
    int 21h
```

## Висновки:

Під час даної роботи ми навчилися створювати підпрограми з їх викликом за допомогою системних переривань клавіатури.