

Міністерство освіти і науки, молоді та спорту України
Національний університет “Львівська політехніка”

Кафедра ЕОМ



Звіт

з лабораторної роботи №6 з дисципліни:
“Системного програмування”

на тему: «Використання макрокоманд та процедур.
Ввід даних з клавіатури та вивід результату на екран.»
Варіант: № 24.

Виконав:
ст. групи КІ-203
Ширий Б. І.
Перевірила:
стар. вик. кафедри ЕОМ
Ногаль М. В.

МЕТА РОБОТИ:

Набути навичок написання макрокоманд та процедур на Асемблері, освоїти способи передачі параметрів. Реалізувати ввід даних з клавіатури та вивід даних на екран.

ЗАВДАННЯ:

УМОВА ЗАВДАННЯ:

1. Створити *.exe програму, яка реалізовує обчислення, задані варіантом над даними, введеними з клавіатури і результат виводить на екран.

Програма повинна складатися з трьох основних підпрограм:

- ✚ процедура вводу – забезпечує ввід даних з клавіатури;
- ✚ процедура безпосередніх обчислень – здійснює всі необхідні арифметичні дії;
- ✚ процедура виводу – забезпечує вивід на екран результату.

Передача параметрів може здійснюватися довільним чином. Кожна з перерахованих процедур може містити довільну кількість додаткових підпрограм.

2. Переконалися у правильності роботи кожної процедури зокрема та програми загалом.
3. Скласти звіт про виконану роботу з приведенням тексту програми та коментарів до неї.
4. Дати відповідь на контрольні запитання.

Завдання для 24 варіанту наведене нижче:

- ✓ Задані число A і масив чисел X. Визначити, яких чисел в масиві більше (1 – більших за A, 0 – менших або рівних A).

ВИКОНАННЯ:

Отож, написав текст Асемблера згідно свого варіанту, який навів у лістингу 1.

Лістинг 1. Текст програми.

```
.686
.model flat, stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\user32.inc
include \masm32\include\msvcrt.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\msvcrt.lib

.data
A dd 0
```

```

X dd 10 dup(0) ; масив
N equ ($-X) / type X ; розмір масиву
hConsoleInput dd 0 ; дескриптор введення консолі
hConsoleOutput dd 0 ; дескриптор виведення консолі
NumberOfChars dd 0 ; кількість прочитаних або написаних символів
ReadBuf db 32 dup(0) ; буфер для введення
MessageInputA db 'Enter the value of A: ', 10, 13
NumberOfChToWMessageA dd $-MessageInputA
MessageInputX db 'Input elements of array: ', 10, 13
NumberOfChToWMessageX dd $-MessageInputX
Greater db '1 - There are more numbers greater than A', 13, 10, 0
Smaller db '0 - There are more numbers smaller than A', 13, 10, 0
Equal db '0 - The number of smaller and larger numbers for A is the same', 13, 10, 0

.code
start:
    call InputA
    call InputX
    push A
    push X
    push N
    call Calculation
    add esp, 12 ; очищаю стек після виклику функції
    push ebx
    call Output
    pop ebx
    invoke ExitProcess, 0

; процедура введення A
InputA proc
    invoke GetStdHandle, -11
    mov hConsoleOutput, eax
    invoke WriteConsoleA, hConsoleOutput, addr MessageInputA,
NumberOfChToWMessageA, addr NumberOfChars, 0
    invoke GetStdHandle, -10
    mov hConsoleInput, eax
    invoke ReadConsoleA, hConsoleInput, addr ReadBuf, 32, addr NumberOfChars,
0
    invoke crt_atoi, addr ReadBuf
    mov A, eax
    ret
InputA endp

; процедура введення - параметри, що передаються через глобальні змінні
InputX proc
    invoke GetStdHandle, -11
    mov hConsoleOutput, eax
    invoke WriteConsoleA, hConsoleOutput, addr MessageInputX,
NumberOfChToWMessageX, addr NumberOfChars, 0
    invoke GetStdHandle, -10
    mov hConsoleInput, eax
    mov ecx, N
    lea ebx, X
    mov edi, 0

    L_input:
        push ecx
        invoke ReadConsoleA, hConsoleInput, addr ReadBuf, 32, addr
NumberOfChars, 0
        invoke crt_atoi, addr ReadBuf
        pop ecx
        mov [ebx][edi], eax
        add edi, 4
    loop L_input

```

```

    ret
InputX endp

; процедура обчислення - параметри, передані через стек, результат в eax
Calculation proc
    push ebp
    mov ebp, esp
    mov ecx, [ebp + 8] ; ecx = N
    mov ebx, 0 ; кількість чисел більших за A
    mov edx, 0 ; кількість чисел менших за A

L:
    mov eax, [X + ecx * type X - type X]
    cmp eax, A
    jg GreaterCount
    jl SmallerCount
    jmp Next
GreaterCount:
    inc ebx
    jmp Next
SmallerCount:
    inc edx
Next:
loop L

    mov eax, ebx ; повернення кількості чисел, більших за A
    pop ebp
    ret 10 ; очищення стеку і повернутися
Calculation endp

; процедура виведення - параметри, що передаються через глобальні змінні
Output proc
    ; Порівняння кількості елементів, більших і менших за A
    cmp ebx, edx
    je EqualCount
    jl SmallerThanA
    jg GreaterThanA

EqualCount:
    ; Якщо дорівнює, відобразити повідомлення Equal
    invoke GetStdHandle, -11
    mov hConsoleOutput, eax
    invoke WriteConsoleA, hConsoleOutput, addr Equal, 62, addr
NumberOfChars, 0
    jmp EndOfComparation

SmallerThanA:
    ; Якщо більше елементів менші за A, відобразити повідомлення Smaller
    invoke GetStdHandle, -11
    mov hConsoleOutput, eax
    invoke WriteConsoleA, hConsoleOutput, addr Smaller, 43, addr
NumberOfChars, 0
    jmp EndOfComparation

GreaterThanA:
    ; Якщо більше елементів більше за A, відобразити повідомлення Greater
    invoke GetStdHandle, -11
    mov hConsoleOutput, eax
    invoke WriteConsoleA, hConsoleOutput, addr Greater, 43, addr
NumberOfChars, 0

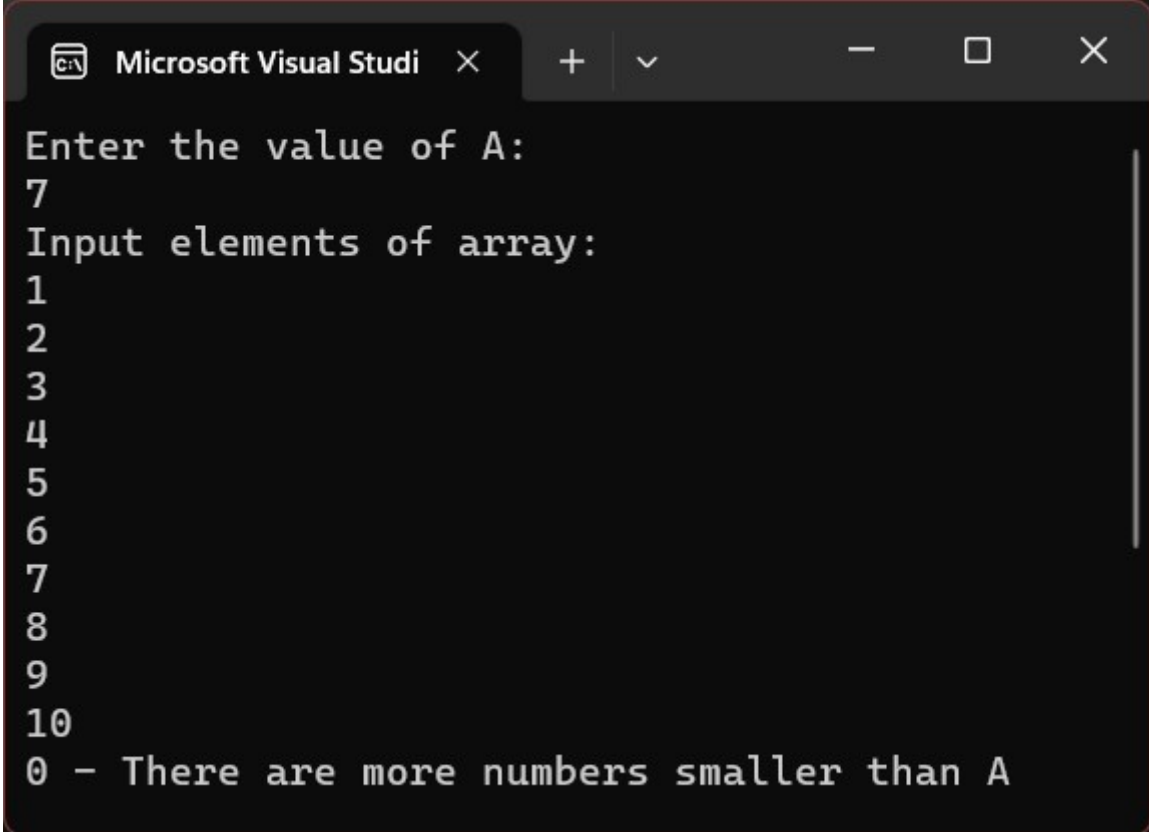
EndOfComparation:
    ret
Output endp

end start

```

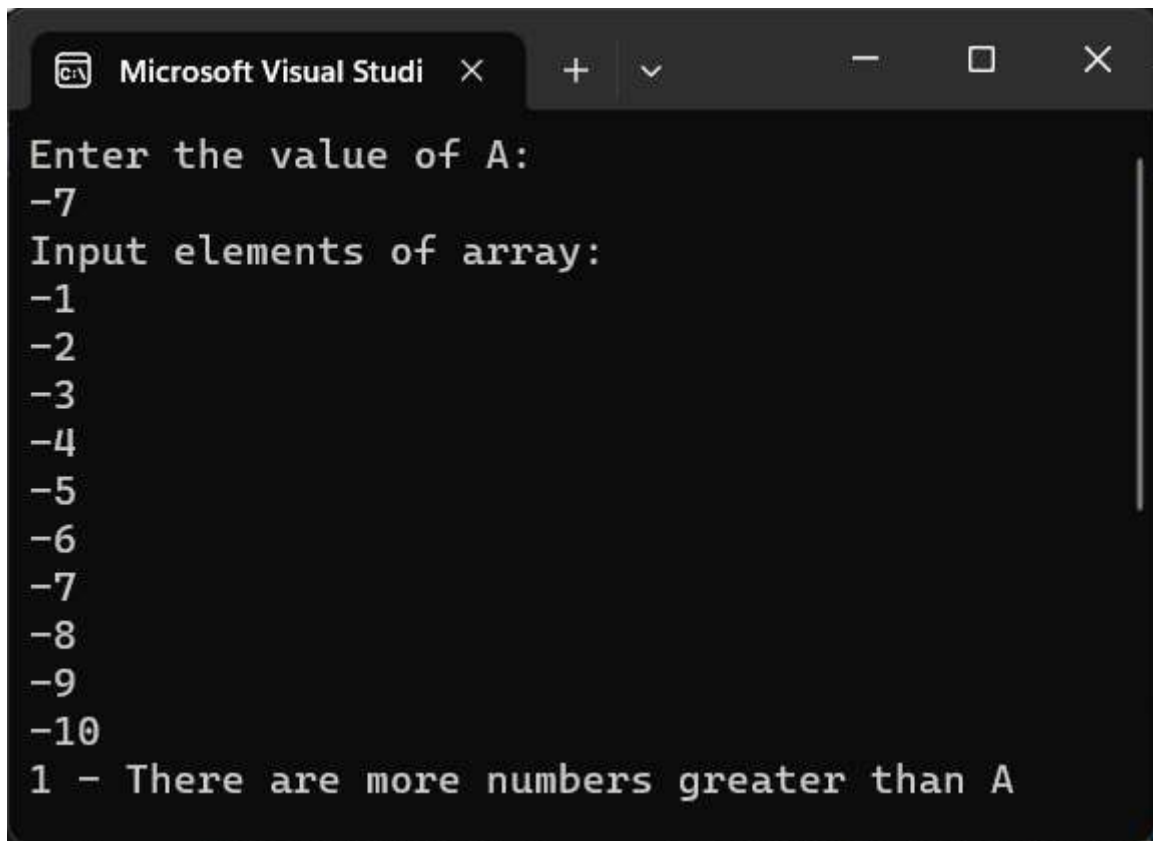
ВИВІД КОНСОЛІ:

Отож, запускав програму з різними вхідними даними, щоб перевірити її на коректність. Вивід консолі зображений на рисунках 1, 2, 3, 4 та 5.



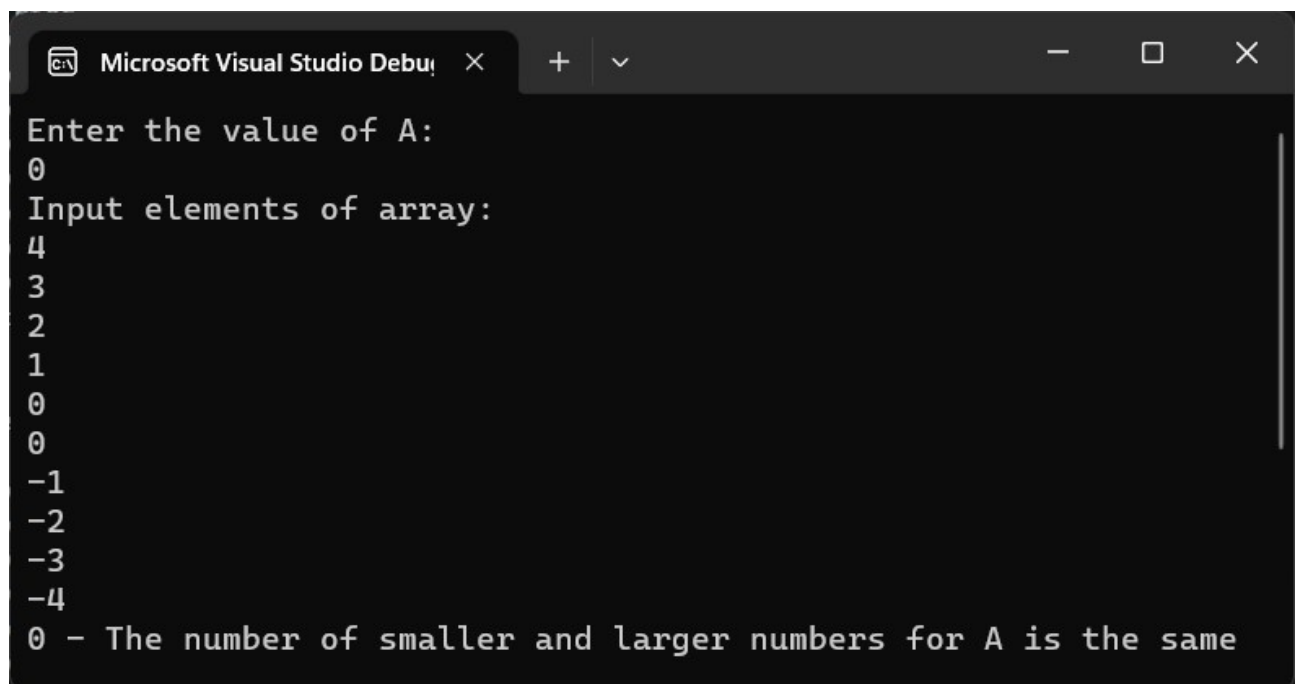
```
Microsoft Visual Studi × + ▾ - □ ×  
Enter the value of A:  
7  
Input elements of array:  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
0 - There are more numbers smaller than A
```

Рис. 1. Вивід, коли більшість елементів масиву менші за A.



```
Microsoft Visual Studi X + v - □ X
Enter the value of A:
-7
Input elements of array:
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
1 - There are more numbers greater than A
```

Рис. 2. Вивід, коли більшість елементів масиву більші за A.



```
Microsoft Visual Studio Debug X + v - □ X
Enter the value of A:
0
Input elements of array:
4
3
2
1
0
0
-1
-2
-3
-4
0 - The number of smaller and larger numbers for A is the same
```

Рис. 3. Вивід, коли кількість елементів масиву, що менші та більші за A - однакова.

ВИСНОВКИ:

У процесі виконання лабораторної роботи набув вмінь і навичок написання макрокоманд та процедур на мові асемблера, а також освоїв способи передачі параметрів і зміг реалізувати ввід даних з клавіатури та вивід даних на екран.

Також ознайомився з тим, як використовувати макрокоманди та процедури для полегшення написання коду та виконання однотипних дій. Зрозумів, як передавати параметри між процедурами та макрокомандами, що дозволяє використовувати їх для виконання різних завдань.

У результаті виконання лабораторної роботи зміг реалізувати ввід даних з клавіатури та вивід даних на екран. Це дозволяє використовувати отримані знання та навички в майбутньому для розробки програмних продуктів на мові асемблера.

КОНТРОЛЬНІ ПИТАННЯ:

ЩО В АСЕМБЛЕРІ РОБЛЯТЬ КОМАНДИ CALL I RET?

Команда CALL в Асемблері використовується для того, щоб здійснити виклик підпрограми. Підпрограма - це фрагмент коду, який може бути використаний у багатьох місцях програми. Коли виконується команда CALL, адреса повернення зберігається у стеку, а виконання програми переходить до початку підпрограми.

Команда RET використовується для повернення виконання програми з підпрограми. Вона видаляє адресу повернення зі стеку і переходить до цієї адреси. Якщо в підпрограмі використовується команда PUSH, то перед викликом RET може бути використана команда POP, щоб відновити значення регістрів, які були збережені у стеку перед викликом підпрограми.

Використання команд CALL і RET дозволяє побудувати складні функції та підпрограми, які можуть використовуватися у різних місцях програми. Це дозволяє підвищити читабельність та структурованість коду та зменшити кількість повторення коду.

ЯК ОПИСАТИ ПРОЦЕДУРУ?

Процедура асемблера - це фрагмент програмного коду, що виконує певну операцію і може бути викликаний з інших частин програми. Вона починається з мітки, яка ідентифікує її, а потім містить послідовність інструкцій процесора, що виконуються в заданому порядку. Після завершення процедури вона повертає управління до того місця програми, звідки була викликана. Для передачі параметрів в процедуру та повернення результату використовуються регістри та стек.

ЯКІ СПОСОБИ ПЕРЕДАЧІ ПАРАМЕТРІВ У ПРОЦЕДУРУ МОЖНА ВИКОРИСТОВУВАТИ?

Існує кілька методів передачі параметрів у процедуру:

| <i>Передача</i> | <i>Пояснення</i> |
|------------------------|--|
| <i>За значенням</i> | Значення параметра передається в процедуру, але будь-які зміни, внесені до параметра в рамках процедури, не впливають на значення вихідного аргументу. |
| <i>За посиланням</i> | Адреса пам'яті параметра передається процедурі, тому будь-які зміни, внесені до параметра в рамках процедури, впливають на вихідний аргумент. |
| <i>За вказівником</i> | Вказівник на параметр передається в процедуру, що дозволяє процедурі опосередковано отримати доступ до оригінального аргументу та змінити його значення. |
| <i>За іменем</i> | Параметр не оцінюється перед передачею в процедуру, а передається його ім'я, яке оцінюється в межах процедури. |

Вибір методу, який використовувати, залежить від мови програмування, типу та розміру параметра та бажаної поведінки процедури.

ЯК ПРАВИЛЬНО ПЕРЕДАВАТИ ПАРАМЕТРИ ЧЕРЕЗ СТЕК?

Щоб правильно передати параметри через стек, ви повинні виконати такі загальні дії:

| | |
|----------|--|
| 1 | Визначити кількість і типи параметрів, які необхідно передати в процедуру. |
| 2 | Надіслати параметри в стек у порядку, зворотному їх появі під час виклику процедури, щоб останній надісланий параметр знаходився за найнижчою адресою в стеку. |
| 3 | Зберегти покажчик поточного стека в регістр або місце пам'яті, щоб відстежувати вихідну позицію стека. |
| 4 | Виклик процедури. |
| 5 | У межах процедури витягніть параметри зі стеку в правильному порядку та збережіть їх у відповідних регістрах або місцях пам'яті для використання в процедурі. |
| 6 | Коли процедура повернеться, відновіть оригінальний вказівник стека на попереднє положення. |

Важливо відзначити, що виклик і виклик (процедура, що викликається) повинні узгодити порядок і типи параметрів, що передаються, а також розмір кожного

параметра. Крім того, абонент повинен переконатися, що в стеку достатньо місця для зберігання всіх параметрів, що передаються.

ДЕ ПРОЦЕДУРА ЗАЛИШАЄ РЕЗУЛЬТАТ СВОЄЇ РОБОТИ?

Процедура може залишати результат своєї роботи в кількох різних місцях, залежно від мови програмування та типу даних, що повертаються.

| Місце | Пояснення |
|-------------------------------|---|
| У поверненому значенні | Деякі мови програмування дозволяють процедурі повертати одне значення як результат. У цьому випадку значення зазвичай зберігається в реєстрі або місці пам'яті, яке зарезервовано для повернутих значень. |
| У вихідному параметрі | Процедура може змінити значення одного або кількох параметрів, переданих їй за посиланням. Ці параметри часто використовуються для повернення кількох значень або складних структур даних. |
| У глобальній змінній | Процедура може змінювати значення глобальної змінної, доступне з інших частин програми. |
| У динамічній пам'яті | Процедура може виділити пам'ять у динамічній пам'яті та повернути вказівник на виділену пам'ять як результат. |

Вибір місця зберігання результату роботи процедури залежить від характеру задачі, що розв'язується, мови програмування, що використовується, і дизайну програми.