

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

*Звіт до комп'ютерного практикуму з дисципліни*  
*«Системне програмне забезпечення»*

**Прийняв**  
**асистент кафедри ІІІ**  
**Пархоменко А.В.**  
**“06” червня 2023 р.**

**Виконав**  
**Студент групи ІІІ-14**  
**Хільчук А.В.**

## Комп'ютерний практикум №5

**Тема:** макрозасоби мови асемблер.

### Завдання:

Скласти програму на нижче наведені завдання:

- 1) переписати програму комп'ютерного практикуму No 2 з використанням одного макроса;
- 2) переписати програму комп'ютерного практикуму No 3 з використанням макросів та передачею параметрів в них;
- 3) переписати одну програму (на вибір студента) комп'ютерного практикуму No 4 з використанням макросів та залученням міток в тілі макросу.

### Виконання:

*Текст програм:*

#### Перша програма:

```
PRINT_VALUE MACRO VALUE
```

```
    LOCAL m1,m2,m3
```

```
    ;making a new line
```

```
    MOV DL, 13
```

```
    MOV AH, 02h
```

```
    INT 21h
```

```
    MOV DL, 10
```

```
    MOV AH, 02h
```

```
    INT 21h
```

```
    MOV bx, VALUE
```

```
    or bx,bx
```

```
    jns m1
```

```
    mov al,'-'
```

```
    int 29h
```

```
    neg bx
```

```
m1:
```

```
    mov ax,bx
```

```
    xor cx,cx
```

```
    mov bx,10
```

```
m2:
```

```
    xor dx,dx
```

```
    div bx
```

```
    add dl,'0'
```

```
    push dx
```

```
    inc cx
    test ax,ax
    jnz m2
m3:
    pop ax
    int 29h
    loop m3
ENDM
```

```
STSEG SEGMENT PARA STACK 'STACK'
    DB 256 DUP ('STACK')
```

```
STSEG ENDS
```

```
DSEG SEGMENT PARA PUBLIC 'DATA'
```

```
    inmes DB 7, 0, 7 dup (0) ; variable to store the input number
    prompt DB 'Enter a number [-32734;32767] : $'
    number dw 0
    CR DB 13, '$'
    validation_failed DB 'Wrong input. Ending the execution...$'
```

```
DSEG ENDS
```

```
CSEG SEGMENT PARA PUBLIC 'CODE'
```

```
    EXTRN atoi : FAR
```

```
MAIN PROC FAR
```

```
    ASSUME CS:CSEG, DS:DSEG, SS:STSEG
    MOV AX, DSEG
    MOV DS, AX
```

```
    call read_input
    call validate_input
    call inmes_chars_to_digits
    call inmes_digits_to_number
    call number_sub
    PRINT_VALUE number
    call stop_exec
```

```
MAIN ENDP
```

```
read_input proc
```

```
    ; display the prompt
    MOV DX, OFFSET prompt
    MOV AH, 9
    INT 21h
```

```
; read the input
lea dx, inmes
MOV ah, 0Ah
int 21h
ret
```

```
read_input endp
```

```
inmes_chars_to_digits PROC
```

```
MOV CL, [inmes+1]
```

```
LEA SI, inmes+2
```

```
cmp byte ptr [si],2Dh
```

```
jne NO_MINUS
```

```
sub CL,1
```

```
inc si
```

```
NO_MINUS:
```

```
FOR_LOOP:
```

```
sub byte ptr [si],48
```

```
inc si;
```

```
LOOP FOR_LOOP
```

```
ret
```

```
inmes_chars_to_digits ENDP
```

```
inmes_digits_to_number PROC
```

```
LEA SI, inmes+2;
```

```
mov ax,0
```

```
mov cx,10
```

```
cmp byte ptr[inmes+2], '-'
```

```
jne FOR_LOOP2
```

```
INC SI
```

```
FOR_LOOP2:
```

```
MUL CX
```

```
ADD AL, [SI]
```

```
INC SI
```

```
cmp byte ptr [si],13
```

```
jne FOR_LOOP2
```

```
    cmp byte ptr[inmes+2], '-'
    jne TO_NUMBER_END
    neg ax
TO_NUMBER_END:
    mov number,AX
    ret
inmes_digits_to_number ENDP
```

```
number_sub proc
    sub number,34
    ret
number_sub endp
```

```
stop_exec proc
    MOV AH, 4Ch
    INT 21h
stop_exec endp
```

```
validate_input proc
```

```
    MOV CL, [inmes+1]
    LEA SI, inmes+2
```

```
    cmp byte ptr [si],2Dh
    jne FOR_LOOP_VALIDATION
    sub CL,1
    inc si
```

```
FOR_LOOP_VALIDATION:
```

```
    cmp byte ptr [si],48
    JL VALIDATION_FAILURE
    cmp byte ptr [si],57
    JGE VALIDATION_FAILURE
```

```
    inc si;
    LOOP FOR_LOOP_VALIDATION
```

```
    ret
VALIDATION_FAILURE:
```

```
    MOV DL, 13
    MOV AH, 02h
```

INT 21h

MOV DL, 10  
MOV AH, 02h  
INT 21h

MOV DX, OFFSET validation\_failed  
MOV AH, 9  
INT 21h

call stop\_exec  
validate\_input endp

CSEG ENDS

END MAIN

**Друга програма:**

CALCULATE\_FUNC MACRO macro\_a,macro\_b,macro\_x  
local x\_is\_zero,x\_less\_than\_zero,func\_overflow,calc\_func\_end  
cmp macro\_x,0  
JE x\_is\_zero  
JS x\_less\_than\_zero

mov ax, macro\_x  
imul macro\_x  
JO func\_overflow  
imul macro\_x  
JO func\_overflow  
imul macro\_a  
JO func\_overflow  
add ax,macro\_b  
JO func\_overflow  
mov chiselnik, ax  
jmp calc\_func\_end

x\_is\_zero:  
mov ax,macro\_a  
add ax,macro\_b  
JO func\_overflow  
add ax,macro\_b  
JO func\_overflow

mov chiselnik, ax  
jmp calc\_func\_end  
x\_less\_than\_zero:

```

    mov ax,macro_a
    imul macro_x
    jo func_overflow
    sub ax,macro_b
    imul macro_x
    jo func_overflow
    mov chiselnik, ax
    jmp calc_func_end
func_overflow:
    MOV DX,offset prompt_overflow
    MOV AH, 9
    INT 21h
    call stop_exec
    calc_func_end:
ENDM

STSEG SEGMENT PARA STACK 'STACK'
    DB 256 DUP ('STACK')
STSEG ENDS
DSEG SEGMENT PARA PUBLIC 'DATA'
    bufer DB 7, 0, 7 dup (0)
    prompt DB 'Enter a number: $'
    validation_failed DB 13,10,'Wrong input. Ending the execution...$'
    result_zalishok db ' zalishok $'
    prompt_x db 'Please, enter x $'
    prompt_a db 13,10,'Please, enter a $'
    prompt_b db 13,10,'Please, enter b $'
    prompt_overflow db 13,10,'Overflow has occurred! Ending program execution
$'
    newline db 13,10,$'
    a dw 0
    b dw 0
    x dw 0
    chiselnik dw 0
DSEG ENDS
CSEG SEGMENT PARA PUBLIC 'CODE'

MAIN PROC FAR
    ASSUME CS:CSEG, DS:DSEG, SS:STSEG
    MOV AX, DSEG
    MOV DS, AX

    call input_values
    CALCULATE_FUNC a,b,x

```

```
    call print_result
    call stop_exec
MAIN ENDP
```

```
input_values proc
```

```
    MOV DX, offset prompt_x
    MOV AH, 9
    INT 21h
```

```
    call single_read_input
    mov x,AX
```

```
    MOV DX,offset prompt_a
    MOV AH, 9
    INT 21h
```

```
    call single_read_input
    mov a,AX
```

```
    MOV DX,offset prompt_b
    MOV AH, 9
    INT 21h
```

```
    call single_read_input
    mov b,AX
```

```
    ret
```

```
input_values endp
```

```
single_read_input proc
```

```
    ; read the input
    lea dx, bufer
    MOV ah, 10
    int 21h
    call validate_input
    call bufer_chars_to_digits
    call bufer_digits_to_number
    ret
```

```
single_read_input endp
```



```

bufer_chars_to_digits PROC
    MOV CL, [bufer+1]
    LEA SI, bufer+2

    cmp byte ptr [si],2Dh
    jne NO_MINUS
    sub CL,1
    inc si
NO_MINUS:
    FOR_LOOP_TO_DIGITS:
        sub byte ptr [si],48
        inc si;
        LOOP FOR_LOOP_TO_DIGITS    ; decrement CX and jump to
FOR_LOOP if CX is not zero
    ret
bufer_chars_to_digits ENDP

```

```

bufer_digits_to_number PROC
    LEA SI, bufer+2;
    mov ax,0
    mov cx,10
    cmp byte ptr[bufer+2], '-'
    jne FOR_LOOP_TO_NUMBERS
    INC SI
    FOR_LOOP_TO_NUMBERS:
        MUL CX

        ADD AL, [SI]
        INC SI

        cmp byte ptr [si],13
        jne FOR_LOOP_TO_NUMBERS

        cmp byte ptr[bufer+2], '-'
        jne TO_NUMBER_END
        neg ax
    TO_NUMBER_END:
        ret
bufer_digits_to_number ENDP

```

```

bx_result_print proc

```

```

    or bx,bx
    jns m1
    mov al,'-'
    int 29h
    neg bx
m1:
    mov ax,bx
    xor cx,cx
    mov bx,10
m2:
    xor dx,dx
    div bx
    add dl,'0'
    push dx
    inc cx
    test ax,ax
    jnz m2
m3:
    pop ax
    int 29h
    loop m3
ret
bx_result_print endp

```

```

stop_exec proc
    MOV AH, 4Ch
    INT 21h
stop_exec endp

```

```

validate_input proc

    MOV CL, [bufer+1]
    LEA SI, bufer+2

    cmp byte ptr [si],2Dh
    jne FOR_LOOP_VALIDATION
    sub CL,1
    inc si
    FOR_LOOP_VALIDATION:

    cmp byte ptr [si],48
    JL VALIDATION_FAILURE

```

```
cmp byte ptr [si],57
JGE VALIDATION_FAILURE
```

```
inc si;
LOOP FOR_LOOP_VALIDATION
ret
```

VALIDATION\_FAILURE:

```
MOV DX, OFFSET validation_failed
MOV AH, 9
INT 21h
```

```
call stop_exec
validate_input endp
```

```
print_result proc
MOV DX,offset newline
MOV AH, 9
INT 21h
mov bx,chiselnik
call bx_result_print
cmp byte ptr [x], 0
jg drobom
ret
```

drobom:

```
MOV DL, '/'
MOV AH, 02h
INT 21h
```

```
mov bx,x
call bx_result_print
```

```
MOV DL, '='
MOV AH, 02h
INT 21h
```

```
mov dx,0
mov ax,chiselnik
div x
mov bx,ax
push dx
call bx_result_print
pop dx
```

```

    cmp dx,0
    je print_end
    push dx
    MOV DX,offset result_zalishok
    MOV AH, 9
    INT 21h
    pop bx
    call bx_result_print
print_end:
    ret
print_result endp

```

CSEG ENDS

END MAIN

**Третья программа:**

FIND\_SUM\_OF\_ARRAY MACRO ARRAY\_REF,ARRAY\_COUNT

```

    local sum_macro_end,finding_sum_loop,sum_overflow
    mov cx, ARRAY_COUNT
    mov di, ARRAY_REF
    mov ax,0
    mov bx,0
finding_sum_loop:
    mov al,byte ptr[di]
    cbw
    add bx,ax
    jo sum_overflow
    inc di
    loop finding_sum_loop
    JMP sum_macro_end

```

```

sum_overflow:
    MOV AH, 09h
    LEA DX, overflow_msg
    INT 21h
    call stop_exec
    sum_macro_end:

```

ENDM

STSEG SEGMENT PARA STACK 'STACK'

DB 256 DUP ('STACK')

STSEG ENDS

DSEG SEGMENT PARA PUBLIC 'DATA'

validation\_failed DB 'Wrong input. Ending the execution...\$'

PROMPT\_MSG DB 'Enter the size of the array: \$'

```

msg_ask_for_input DB 0Ah, 0Dh, 'Enter the members of the array [-128;127]:
$'
msg1 DB 'Enter the size of the array: $'
msg2 DB 0Ah, 0Dh, 'Sum of all elements of array is:$'
msg3 DB 0Ah, 0Dh, 'Maximum element of array is: $'
msg4 DB 0Ah, 0Dh, 'The contents of the array before sort are:$'
msg5 DB 0Ah, 0Dh, 'The contents of the array after sort are:$'
overflow_msg DB 0Ah, 0Dh, 'Overflow has occurred!Stopping the
execution...$'
msg6 DB 0Ah, 0Dh, 'Array dimension cant be zero or negative! Ending
program execution...$'
msg7 DB 0Ah, 0Dh, 'Array member value must be in range [-128;127]! Ending
program execution...$'
newline DB 0Dh, 0Ah, '$'
bufer DB 7, 0, 7 dup (0) ; variable to store the input number
ARRAY_SIZE DW ?
my_array Dw ?
DSEG ENDS
CSEG SEGMENT PARA PUBLIC 'CODE'

```

```

MAIN PROC FAR

```

```

    ASSUME CS:CSEG, DS:DSEG, SS:STSEG
    MOV AX, DSEG
    MOV DS, AX ; set DS to point to DSEG

```

```

    call init_array

```

```

    FIND_SUM_OF_ARRAY my_array, ARRAY_SIZE
    MOV AH, 09h
    LEA DX, msg2
    INT 21h
    call print_bx_from_new_line

```

```

    call find_max
    MOV AH, 09h
    LEA DX, msg3
    INT 21h
    call print_bl_from_new_line

```

```

    MOV AH, 09h
    LEA DX, msg4
    INT 21h
    call print_array

```

```

call bubble_sort
MOV AH, 09h
LEA DX, msg5
INT 21h
call print_array
; free the memory allocated for the array
MOV AH, 49h ; free memory function
MOV BX, DI
INT 21h

MOV AH, 4Ch
INT 21h
main endp

```

```

bubble_sort proc
    mov cx, array_size
    dec cx

    outer_loop:
        mov si, my_array
        mov dx, cx
        inner_loop:
            mov al, [si]
            cmp al, [si+1]
            jle skip_swap
            xchg al, [si+1]
            mov [si], al
        skip_swap:
            inc si
            dec dx
            jnz inner_loop
        loop outer_loop
    ret
bubble_sort endp

```

```

find_max proc
    mov cx, array_size
    mov di, my_array
    mov bl, byte ptr [di]
    inc di
    dec cx
    comparison_loop:
        cmp byte ptr [di], bl

```

```
    jl comparison_loop_end
    mov bl,byte ptr [di]
comparison_loop_end:
    inc di
    loop comparison_loop
    ret
find_max endp
```

```
print_array proc
    MOV AH, 09h
    LEA DX, newline
    INT 21h

    mov cx, array_size
    mov di,my_array
    mov bx,0
printing_array_loop:
    push cx
    mov bl,byte ptr [di]
    call print_bl
    inc di
    mov dl, ' '
    mov ah, 02h
    int 21h
    pop cx
    loop printing_array_loop
    ret
print_array endp
```

```
init_array proc

    MOV AH, 09h
    LEA DX, msg1
    INT 21h

    call single_dimension_input
    MOV ARRAY_SIZE,AX

    MOV AH, 48h
    MOV BX, ARRAY_SIZE
    INT 21h
    MOV my_array, AX
```

```
LEA DX, msg_ask_for_input
MOV AH, 09h
INT 21h
```

```
    mov cx, array_size
    mov di, my_array
array_member_input_loop:
    MOV AH, 09h
    LEA DX, newline
    INT 21h
    push cx
    call single_value_input
    mov [di], AX
    pop cx
    inc di
    loop array_member_input_loop
    ret
init_array endp
```

```
single_value_input proc
    call single_read_input
    cmp ax, -129
    jle value_validation_error
    cmp ax, 128
    jge value_validation_error
    ret
```

```
value_validation_error:
    mov ah, 09h
    lea dx, msg7
    int 21h
    call stop_exec
single_value_input endp
```

```
single_dimension_input proc
    call single_read_input
    cmp ax, 0
    jle dimension_validation_error
    ret
```

```
dimension_validation_error:
    mov ah, 09h
    lea dx, msg6
    int 21h
    call stop_exec
single_dimension_input endp
```



single\_read\_input proc

```
; read the input
lea dx, bufer
MOV ah, 10
int 21h

call validate_input
call bufer_chars_to_digits
call bufer_digits_to_number
ret
```

single\_read\_input endp

validate\_input proc

```
MOV CL, [bufer+1]
LEA SI, bufer+2

cmp byte ptr [si],2Dh
jne FOR_LOOP_VALIDATION
sub CL,1
inc si
```

FOR\_LOOP\_VALIDATION:

```
cmp byte ptr [si],48
JL VALIDATION_FAILURE
cmp byte ptr [si],58
JGE VALIDATION_FAILURE
```

```
inc si;
LOOP FOR_LOOP_VALIDATION
ret
```

VALIDATION\_FAILURE:

```
MOV DL, 13
MOV AH, 02h
INT 21h
```

```
MOV DL, 10 ; Line Feed
MOV AH, 02h
INT 21h
```

```
; display the prompt
MOV DX, OFFSET validation_failed
MOV AH, 9
INT 21h
```

```
    call stop_exec
validate_input endp
bufer_chars_to_digits PROC
    MOV CL, [bufer+1]
    LEA SI, bufer+2

    cmp byte ptr [si],2Dh
    jne NO_MINUS
    sub CL,1
    inc si
NO_MINUS:
    FOR_LOOP_TO_DIGITS:
        sub byte ptr [si],48
        inc si;
        LOOP FOR_LOOP_TO_DIGITS
    ret
bufer_chars_to_digits ENDP
```

```
bufer_digits_to_number PROC

    LEA SI, bufer+2;

    mov ax,0
    mov cx,10
    cmp byte ptr[bufer+2], '-'
    jne FOR_LOOP_TO_NUMBERS
    INC SI
    FOR_LOOP_TO_NUMBERS:
        MUL CX

        ADD AL, [SI]
        INC SI

        cmp byte ptr [si],13
        jne FOR_LOOP_TO_NUMBERS

    cmp byte ptr[bufer+2], '-'
```

```
jne TO_NUMBER_END  
neg ax
```

```
TO_NUMBER_END:  
ret  
bufer_digits_to_number ENDP
```

```
stop_exec proc  
MOV AH, 4Ch  
INT 21h  
stop_exec endp
```

```
print_bx proc  
or bx,bx  
jns m1  
mov al,'-'  
int 29h  
neg bx
```

```
m1:  
mov ax,bx  
xor cx,cx  
mov bx,10
```

```
m2:  
xor dx,dx  
div bx  
add dl,'0'  
push dx  
inc cx  
test ax,ax  
jnz m2
```

```
m3:  
pop ax  
int 29h  
loop m3
```

```
ret  
print_bx endp
```

```
print_bx_from_new_line proc  
MOV AH, 09h  
LEA DX, newline  
INT 21h  
call print_bx  
ret  
print_bx_from_new_line endp
```

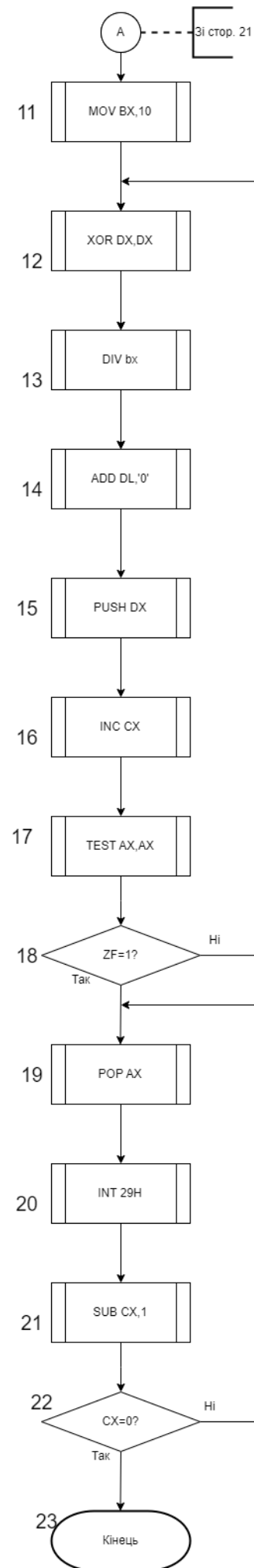
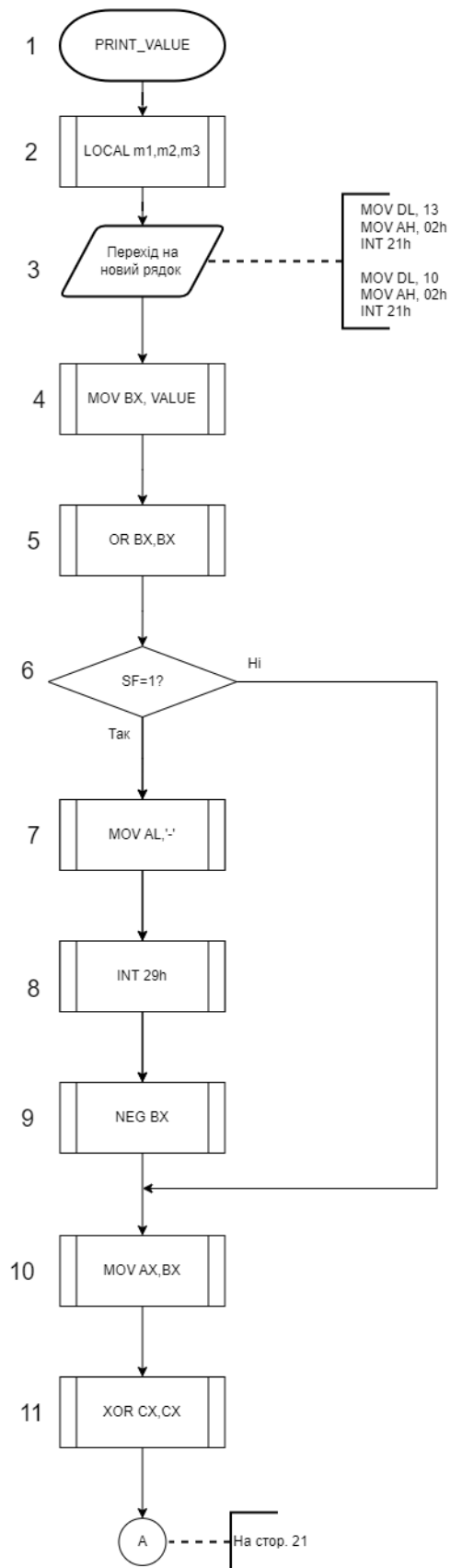
```

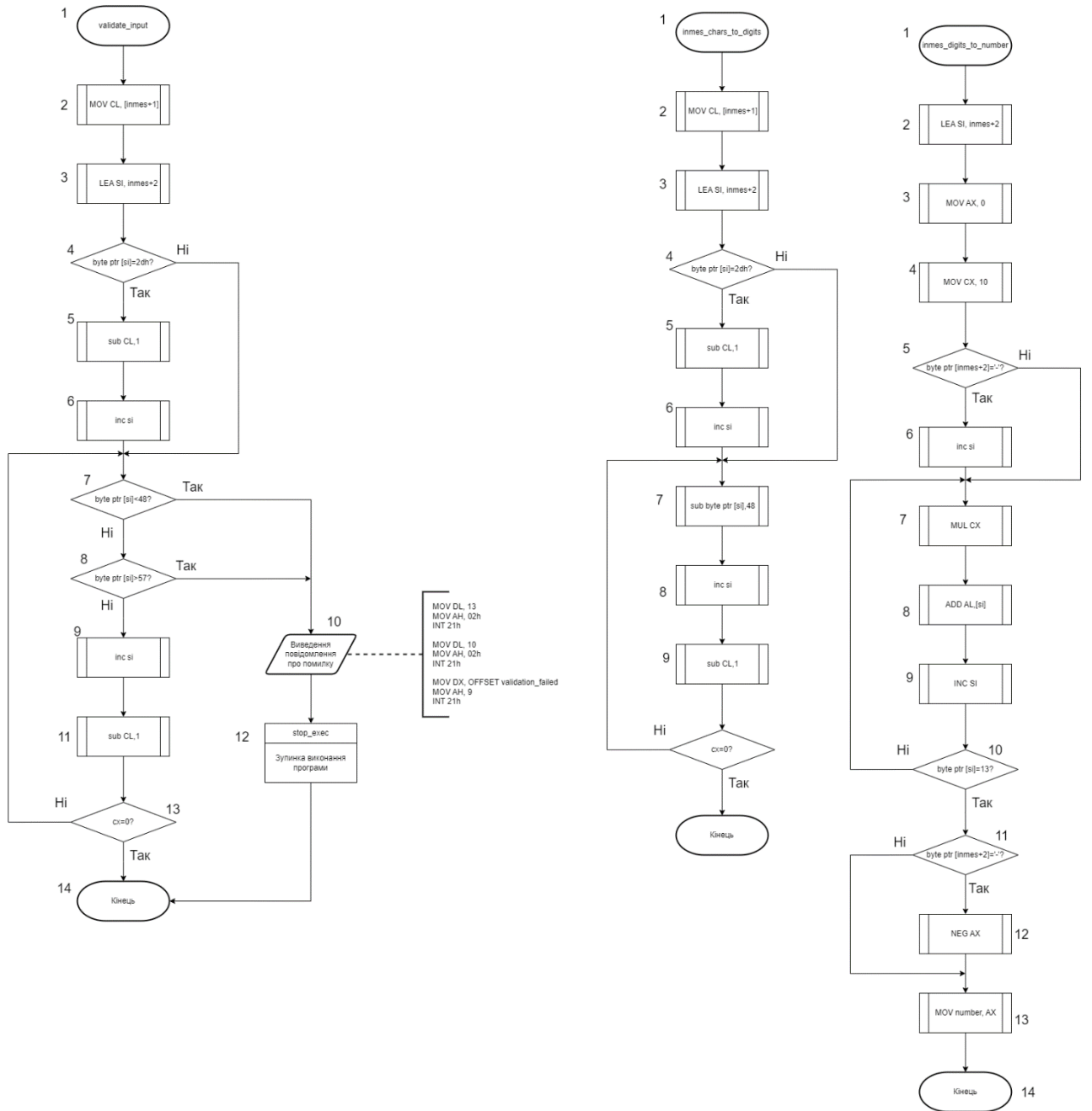
print_bl proc
    or bl,bl
    jns m1_bl
    mov al,'-'
    int 29h
    neg bl
m1_bl:
    mov ax,bx
    xor cx,cx
    mov bx,10
m2_bl:
    xor dx,dx
    div bx
    add dl,'0'
    push dx
    inc cx
    test ax,ax
    jnz m2_bl
m3_bl:
    pop ax
    int 29h
    loop m3_bl
ret
print_bl endp
print_bl_from_new_line proc
    MOV AH, 09h
    LEA DX, newline
    INT 21h
    call print_bl
    ret
    print_bl_from_new_line endp
CSEG ENDS
END MAIN

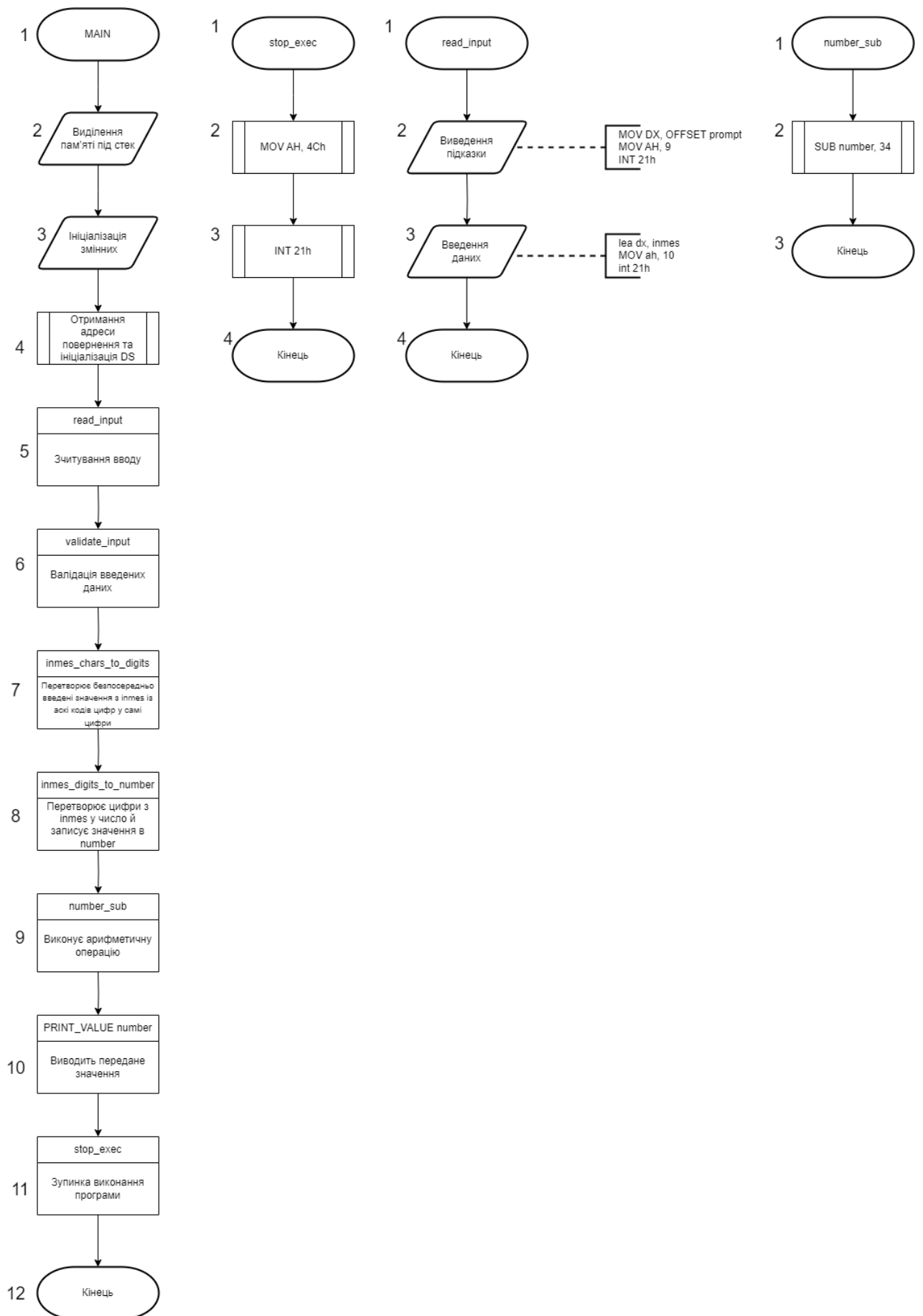
```

**Схема функціонування програми:**

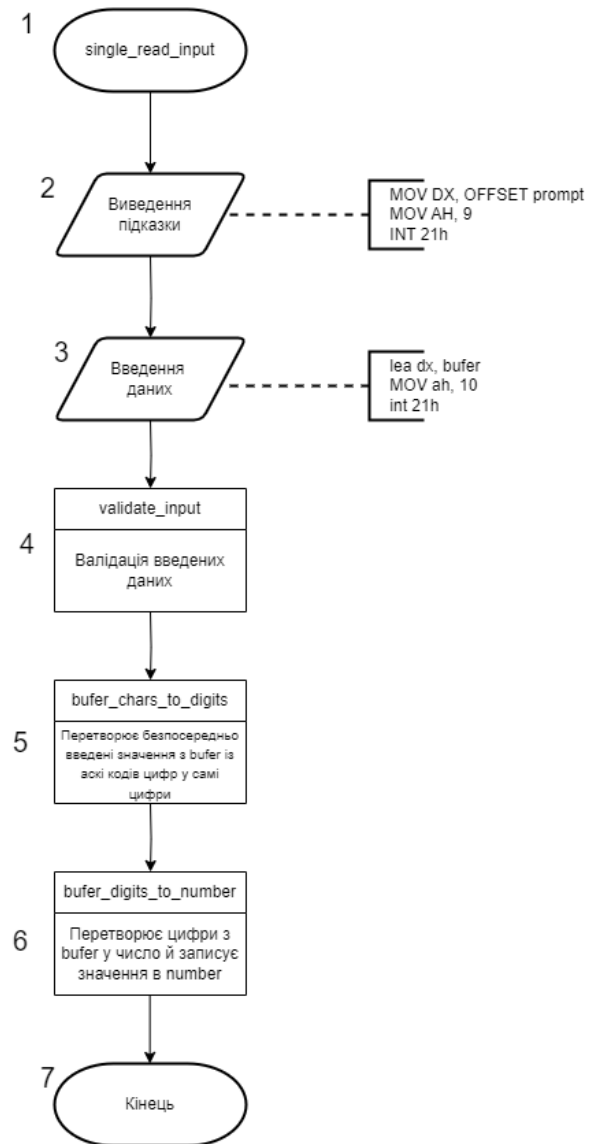
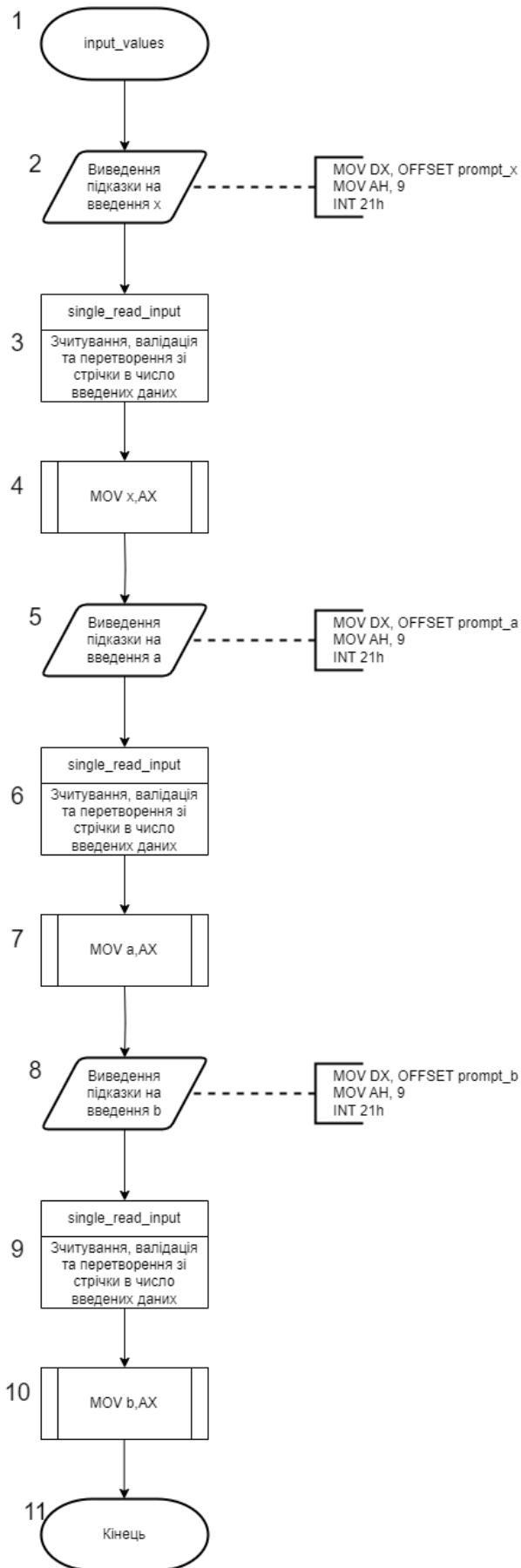
**Перша програма:**



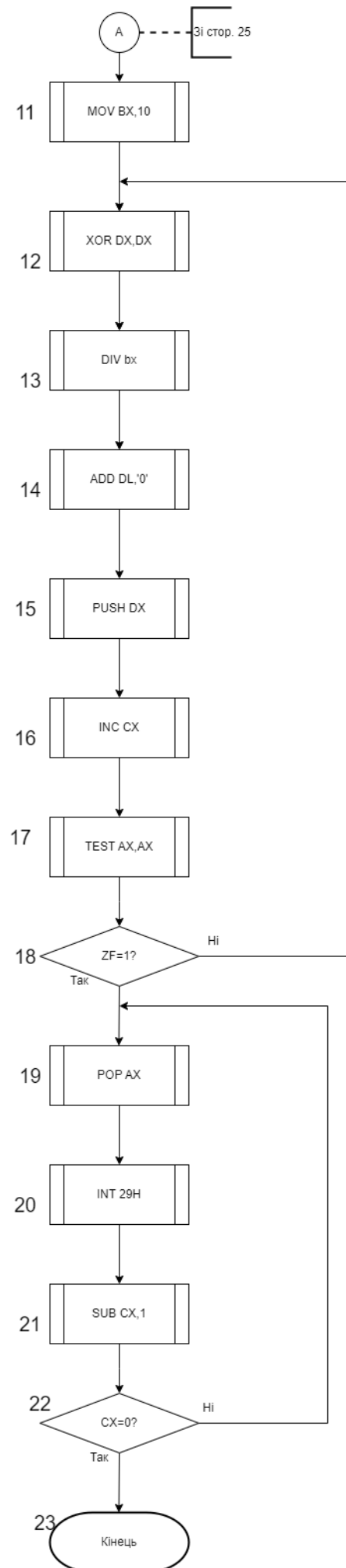
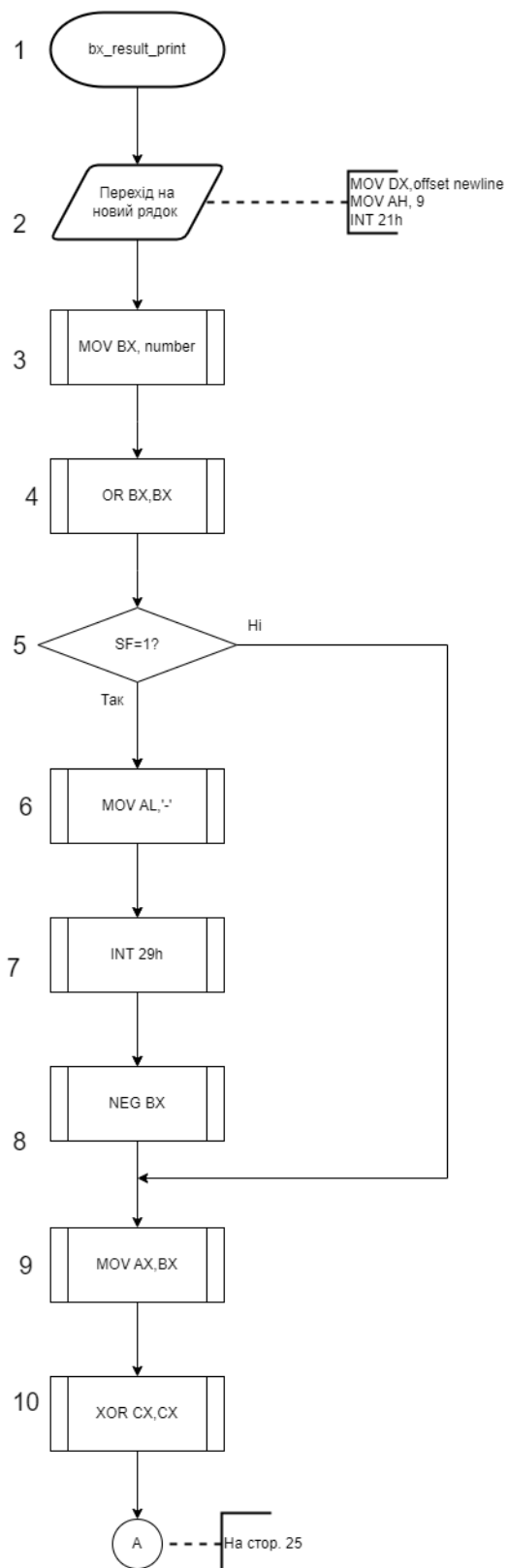


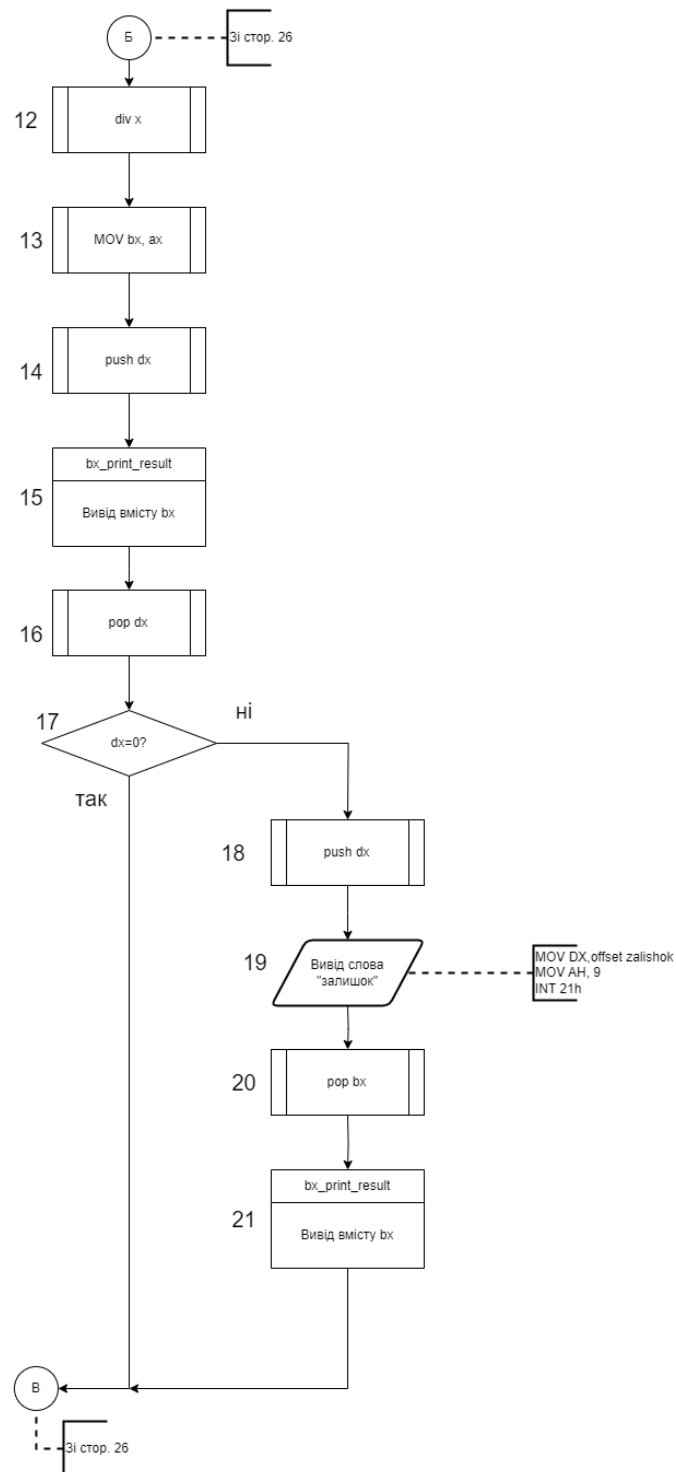
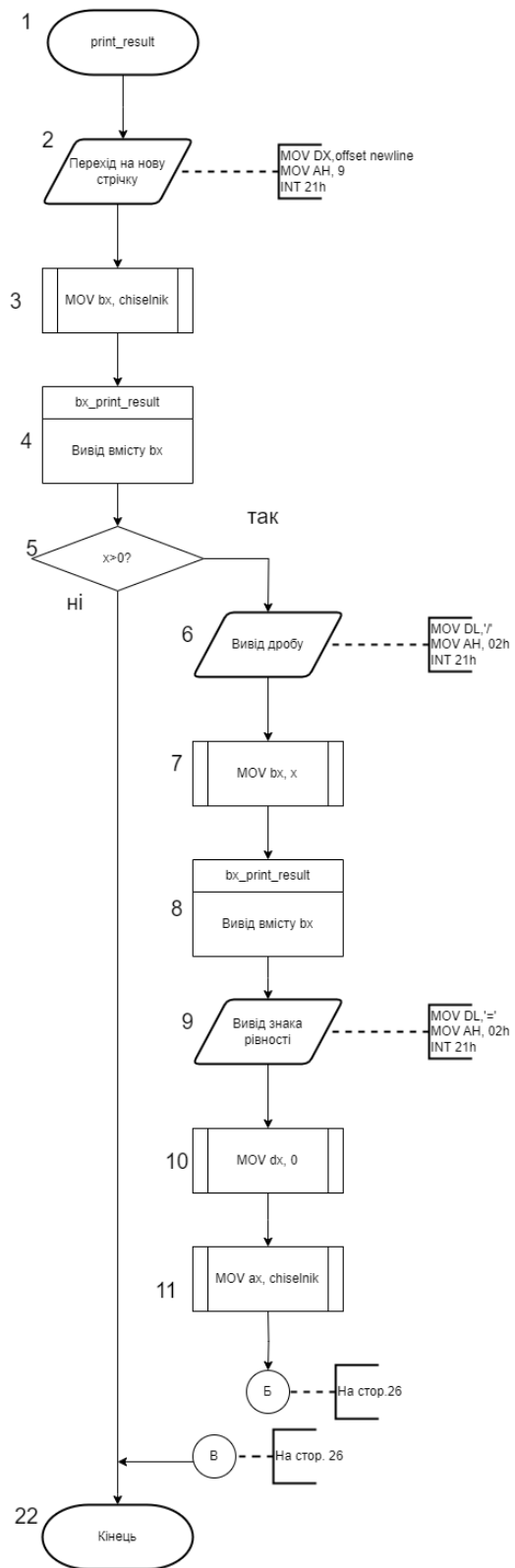


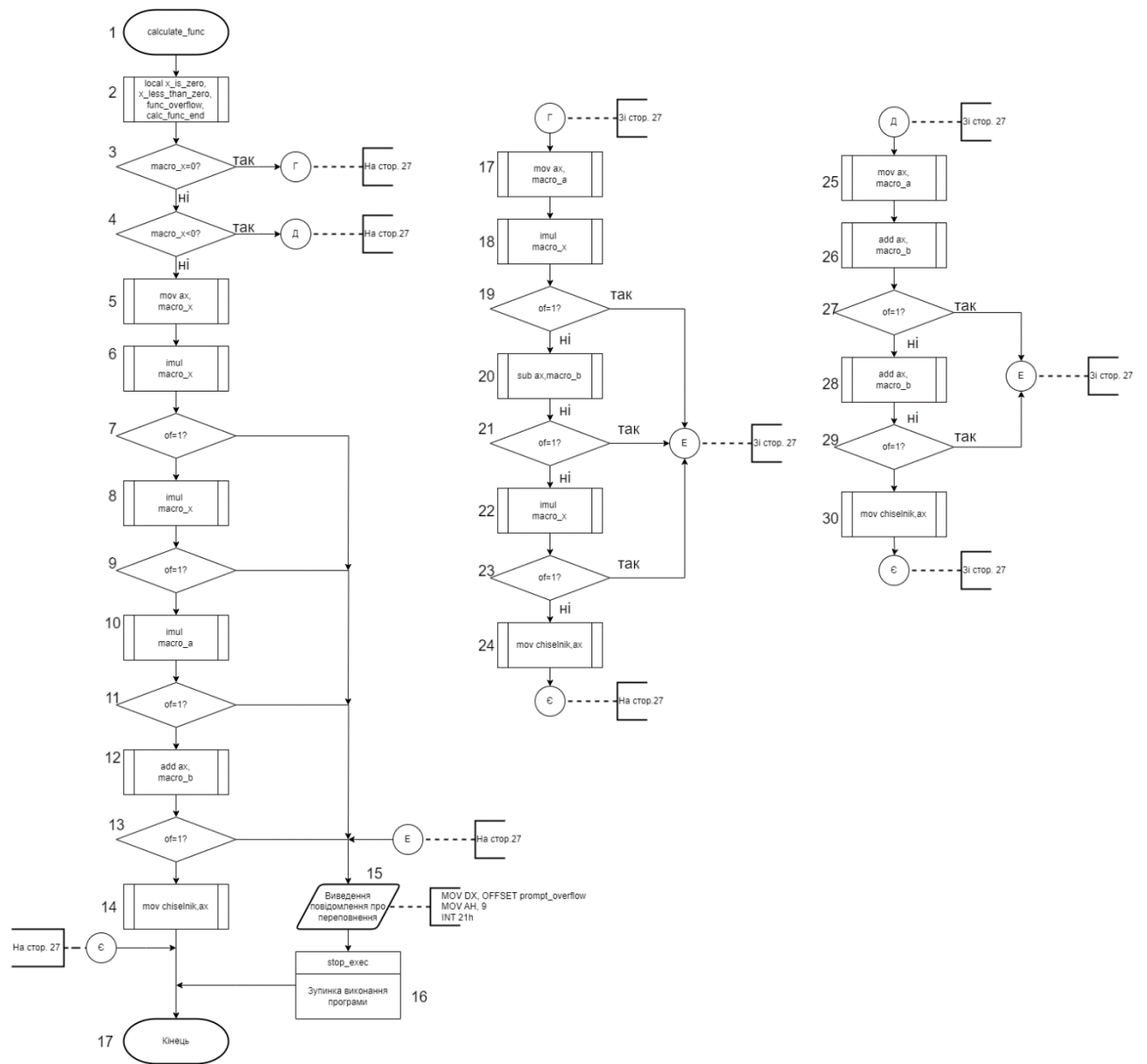
Друга програма:

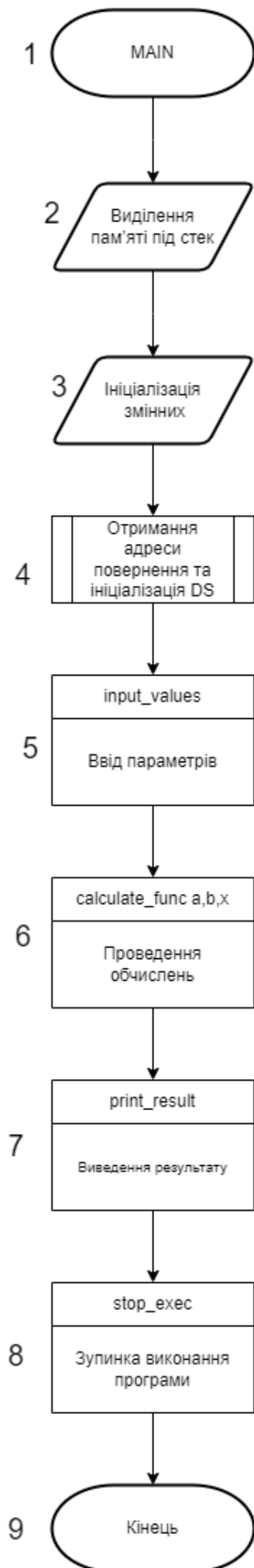
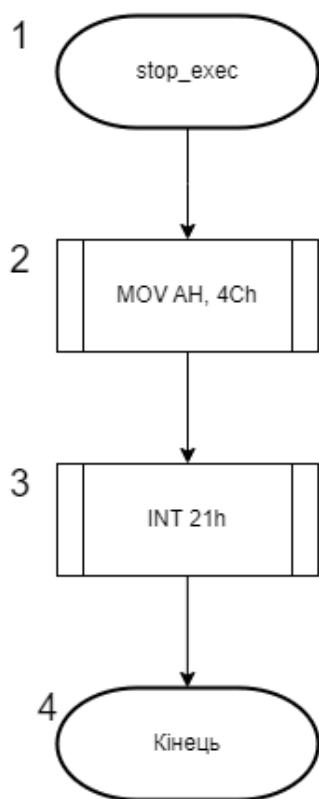




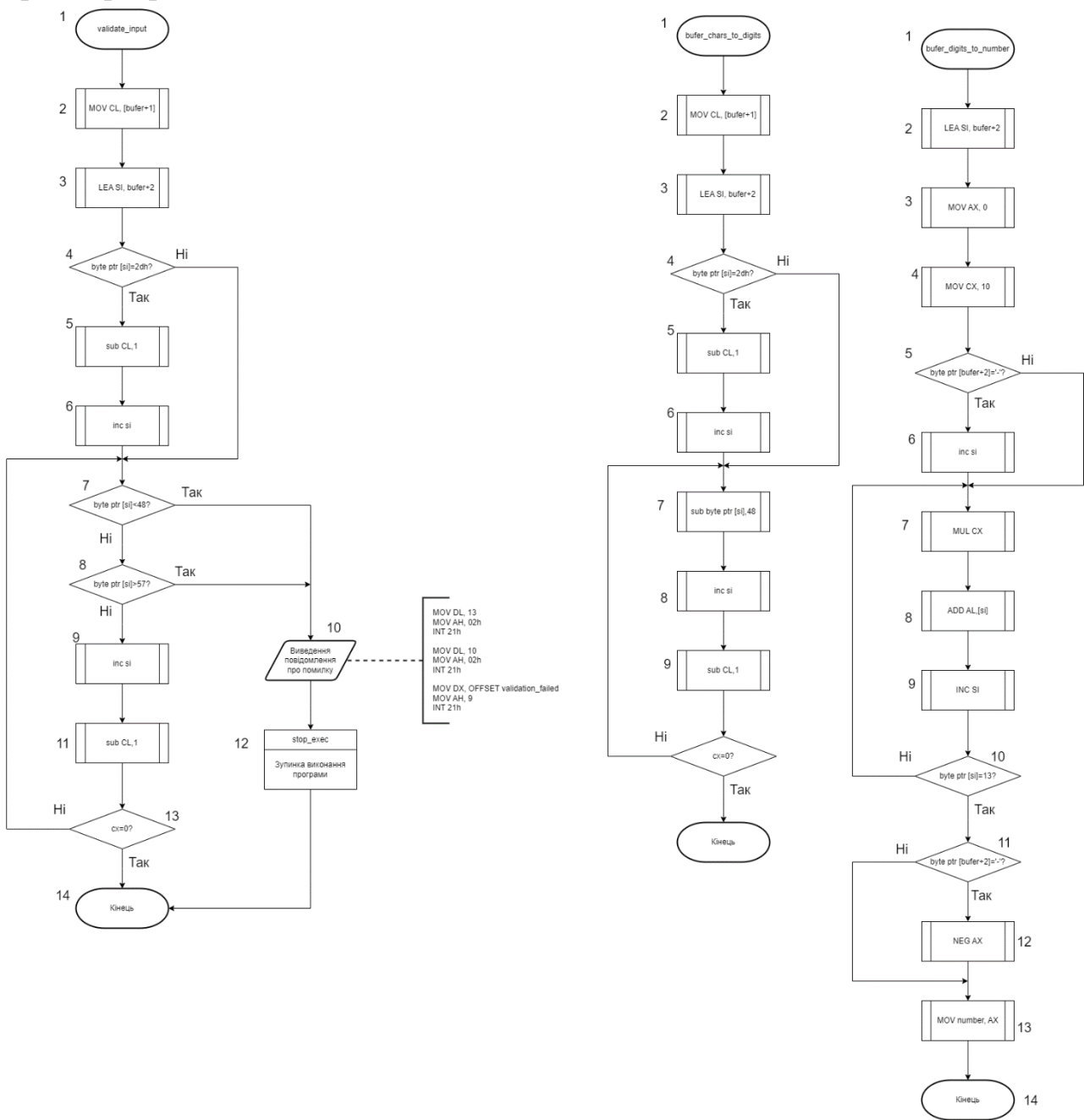


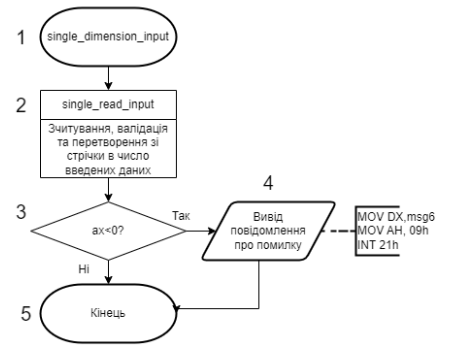
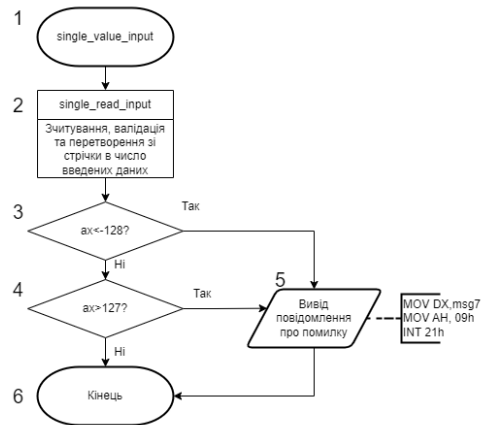
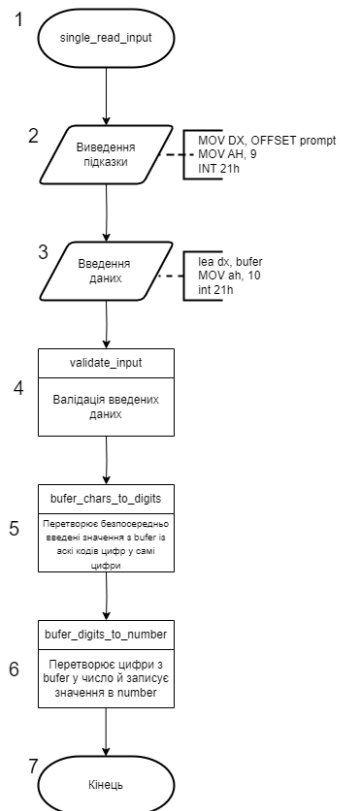


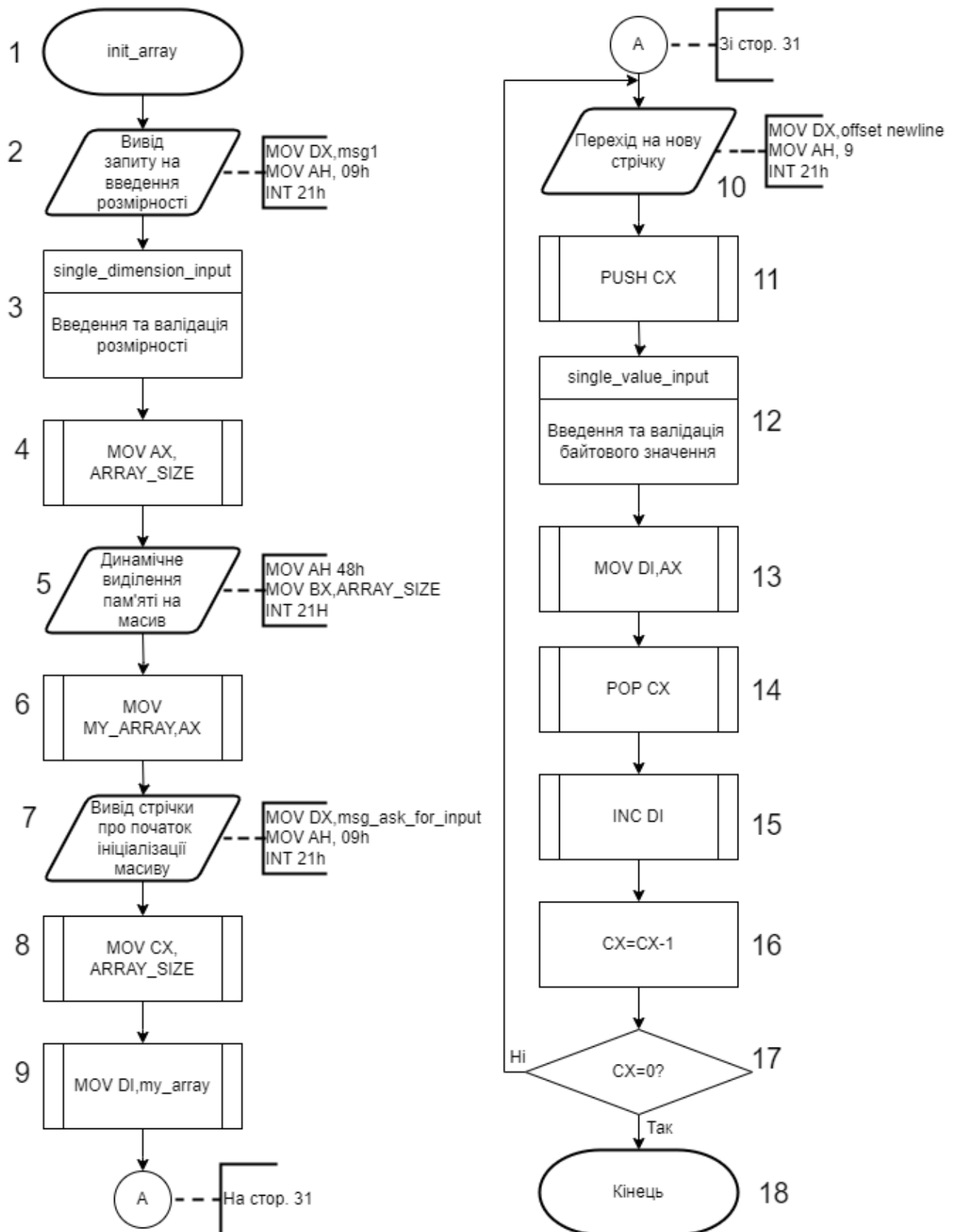


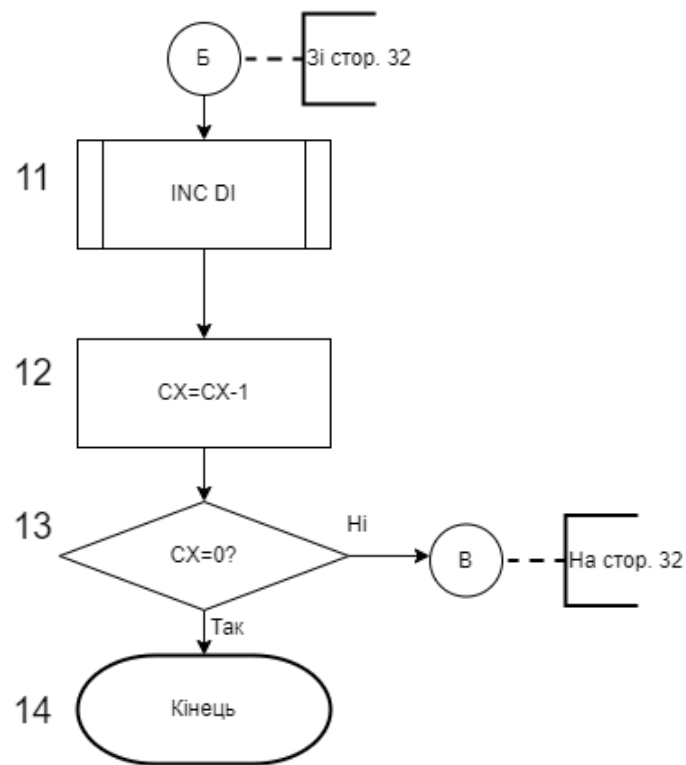
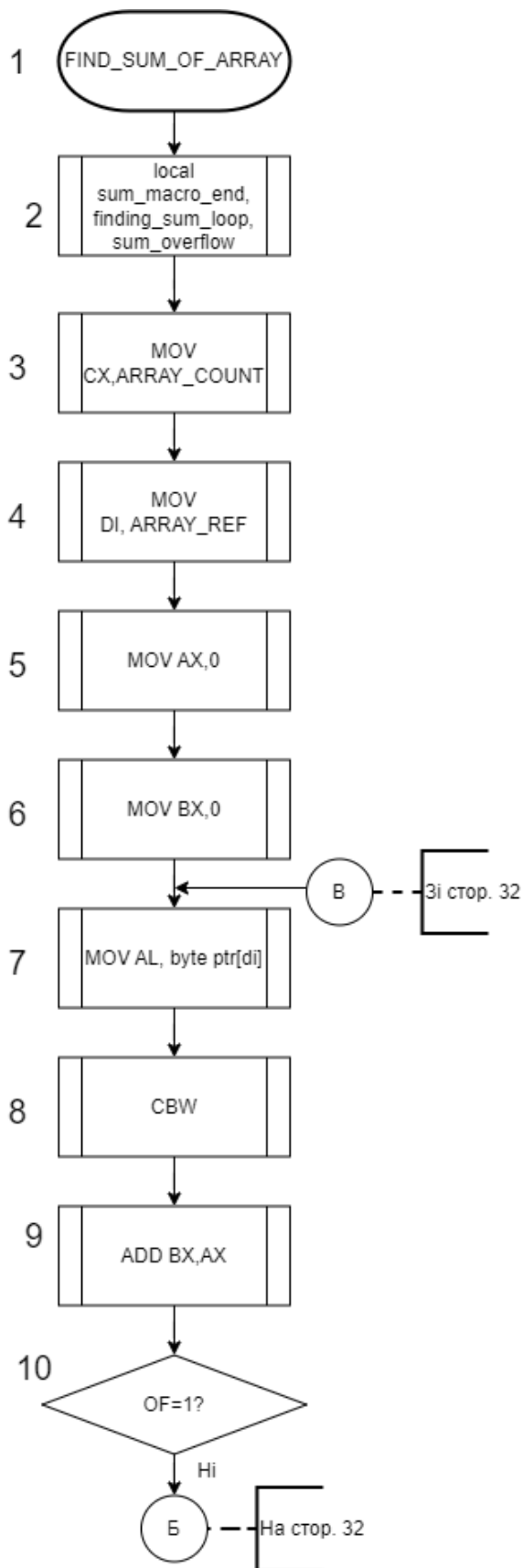


Третя програма:

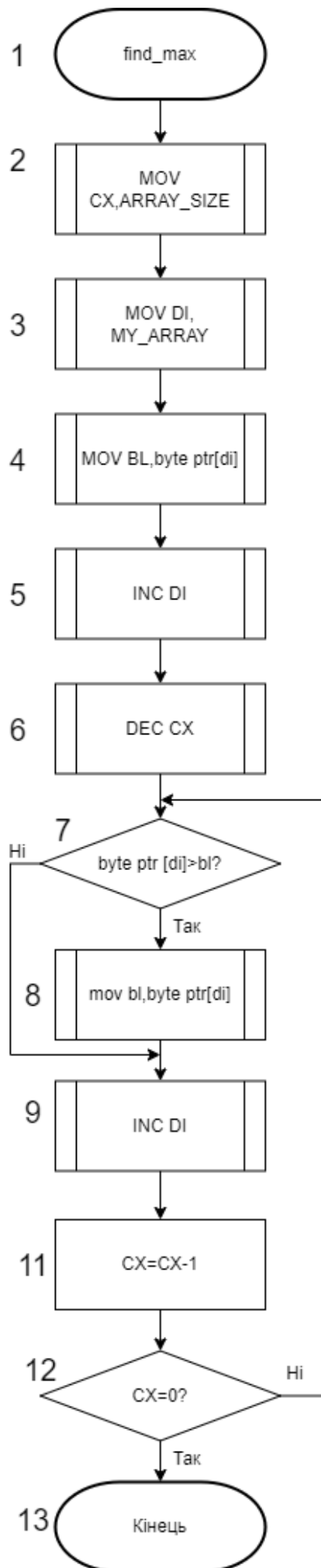


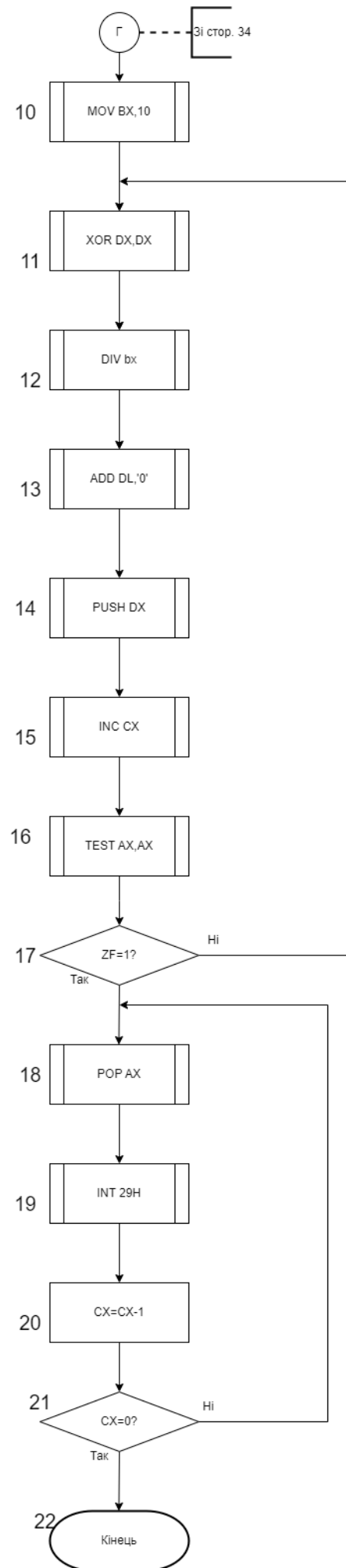
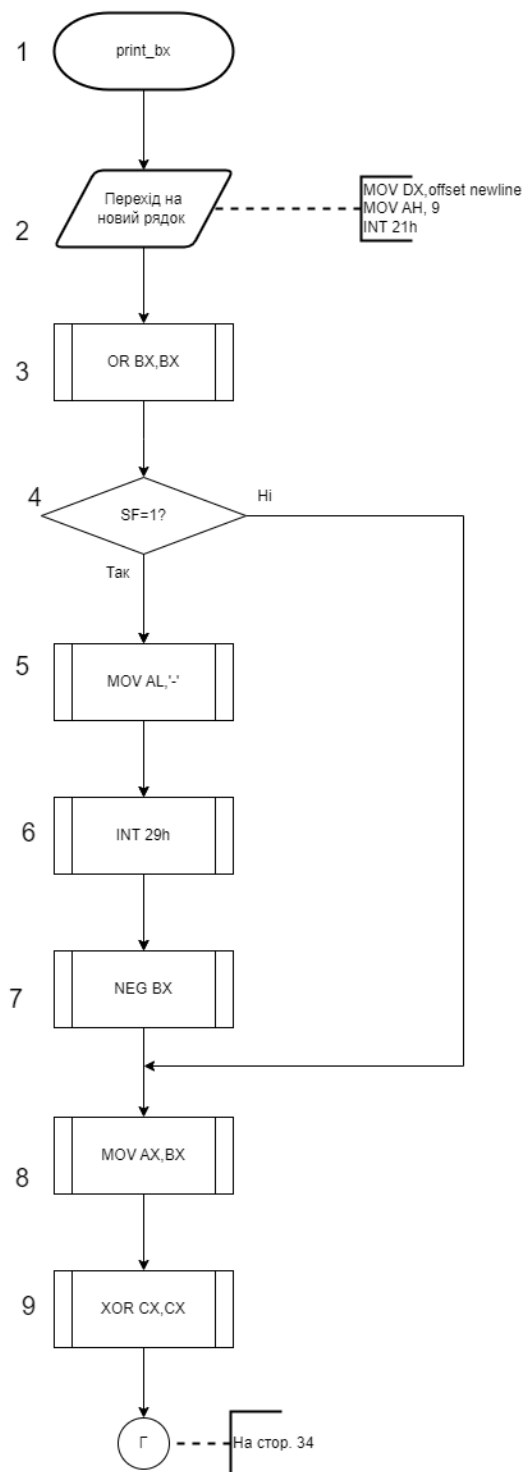


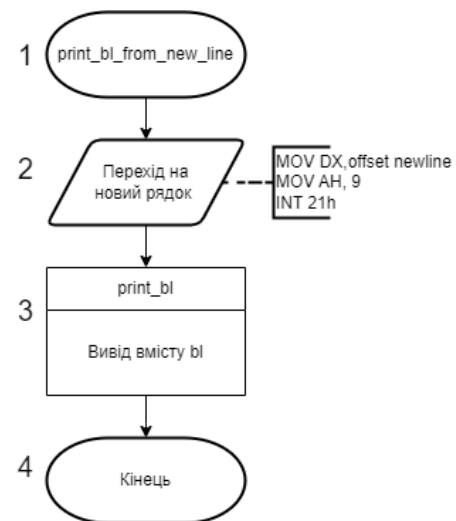
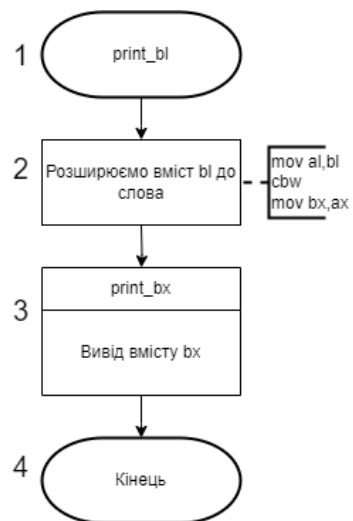
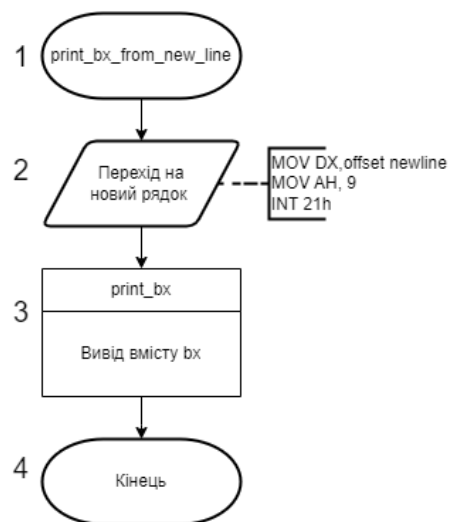


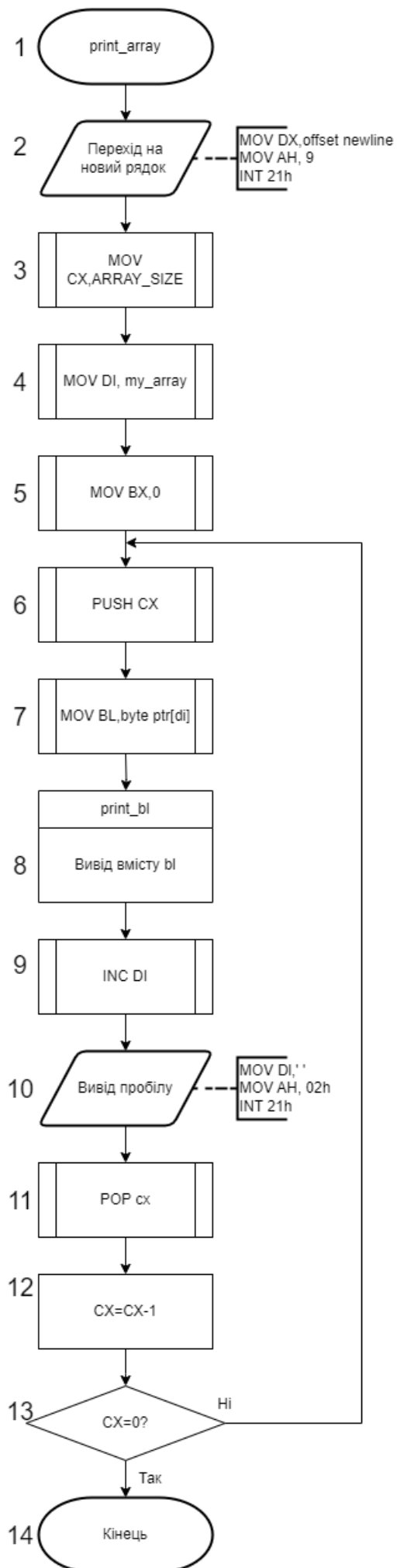


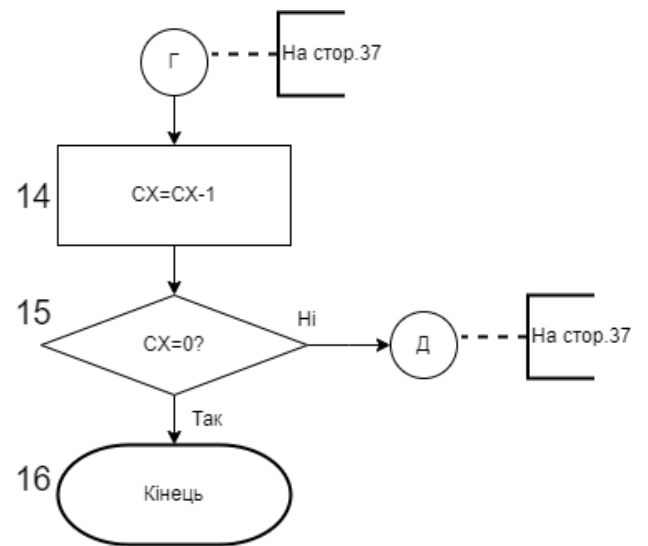
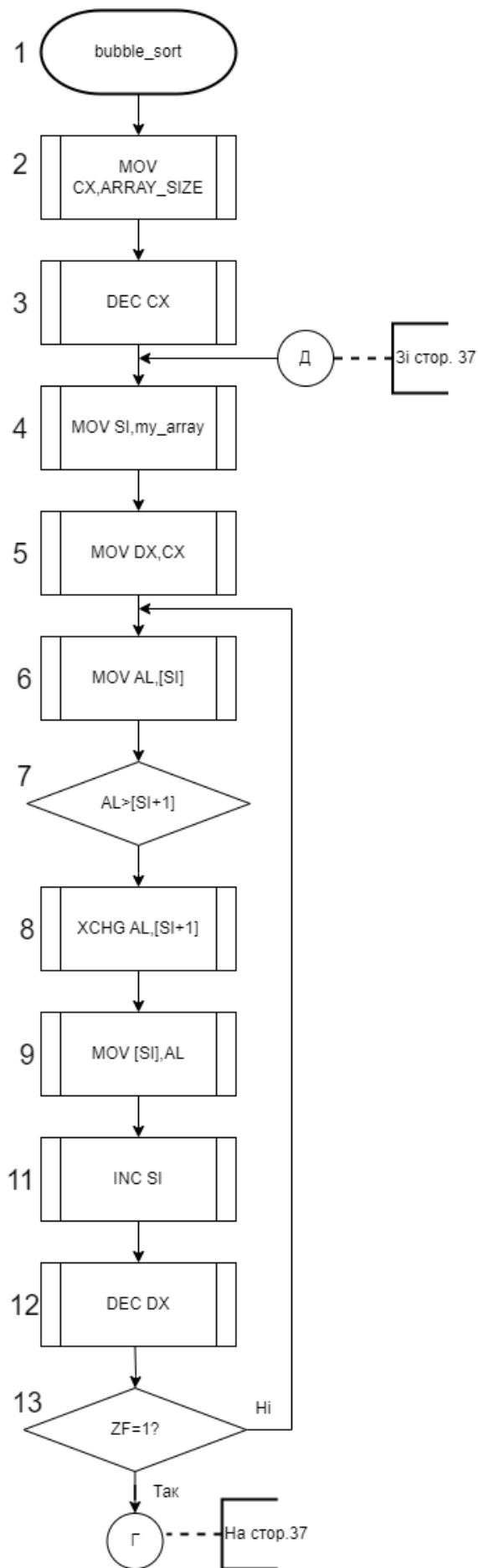


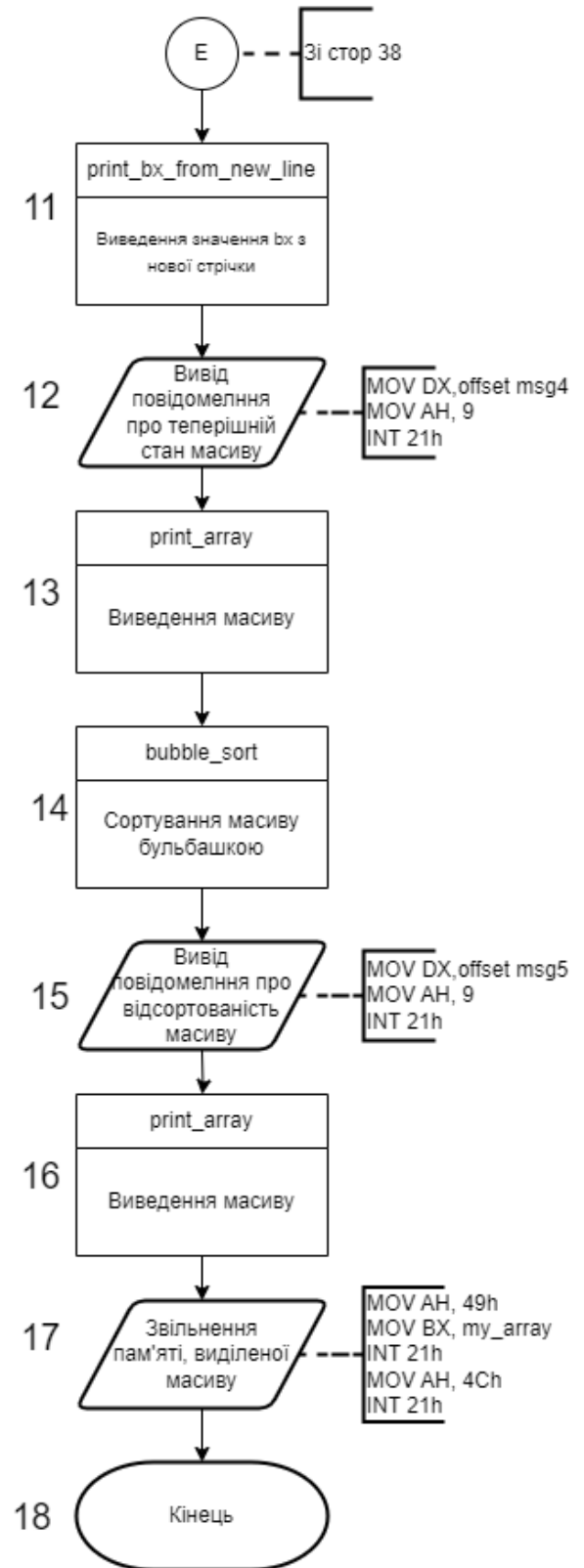
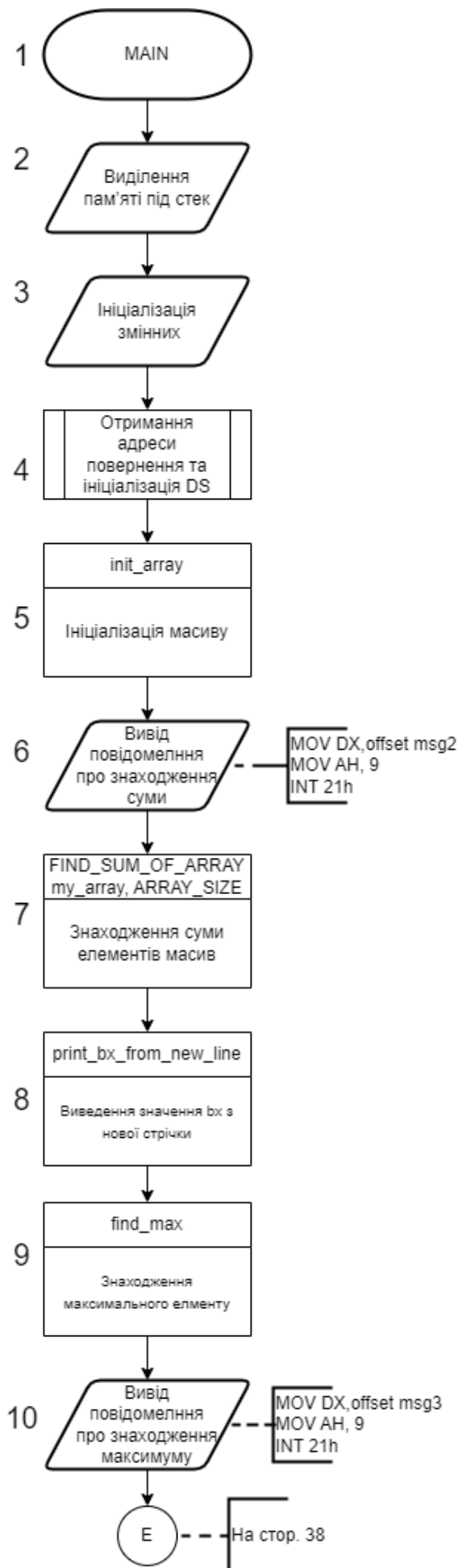












### Приклад локальних міток макросів в .lst файлах:

```
115 0003 8E D8          MOV DS, AX ; set DS to point to DSEG
116
117 0005 E8 00B4        call init_array
118
119                      FIND_SUM_OF_ARRAY my_array, ARRAY_SIZE
1 120 0008 8B 0E 01FFr   mov cx, ARRAY_SIZE
1 121 000C 8B 3E 0201r   mov di, my_array
1 122 0010 B8 0000      mov ax,0
1 123 0013 BB 0000      mov bx,0
1 124 0016              ??0001:
1 125 0016 8A 05        mov al,byte ptr[di]
1 126 0018 98          cbw
1 127 0019 03 D8        add bx,ax
1 128 001B 70 06        jo ??0002
1 129 001D 47          inc di
1 130 001E E2 F6        loop ??0001
1 131 0020 EB 0B 90     JMP ??0000
1 132 0023              ??0002:
1 133 0023 B4 09        MOV AH, 09h
1 134 0025 BA 012Cr    LEA DX, overflow_msg
1 135 0028 CD 21        INT 21h
1 136 002A E8 0174     call stop_exec
1 137 002D              ??0000:
138 002D B4 09        MOV AH, 09h
139 002F BA 0090r     LEA DX, msg2
140 0032 CD 21        INT 21h
141 0034 E8 0192     call print_bx_from_new_line
142
```

### Приклад роботи програм:

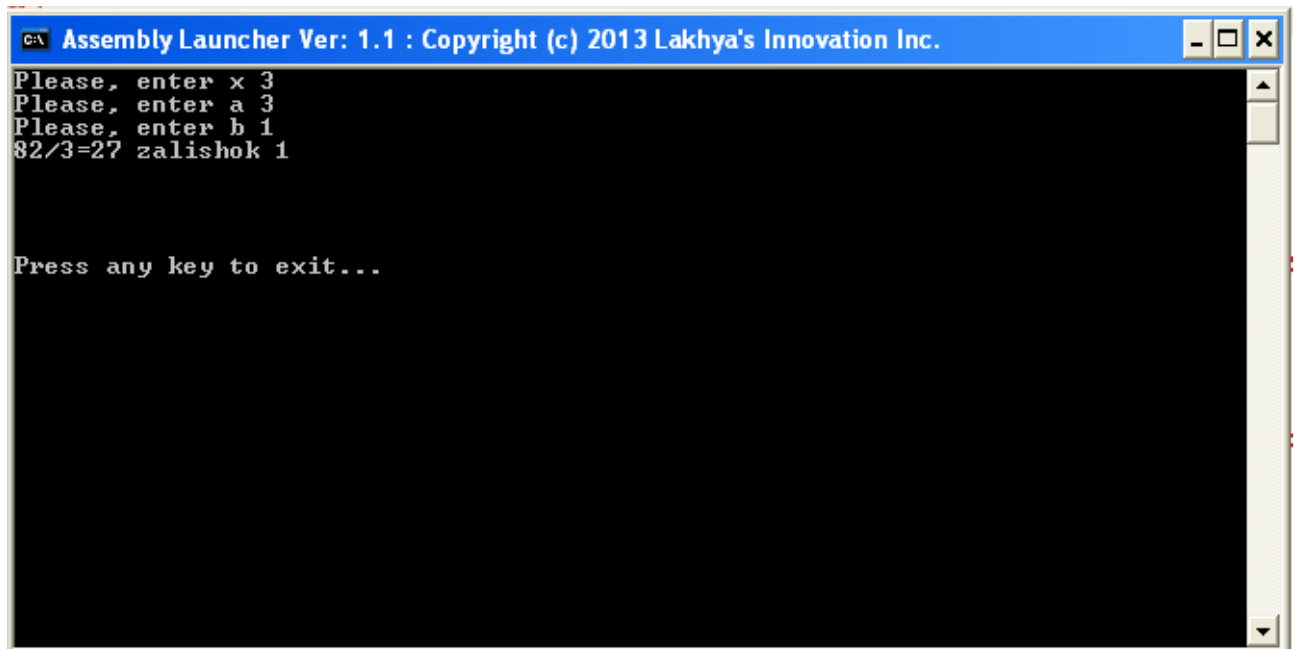
#### Перша програма:



```
C:\> Assembly Launcher Ver: 1.1 : Copyright (c) 2013 Lakhya's Innovation Inc.
Enter a number [-32734;32767] : 1234
1200

Press any key to exit...
```

#### Друга програма:

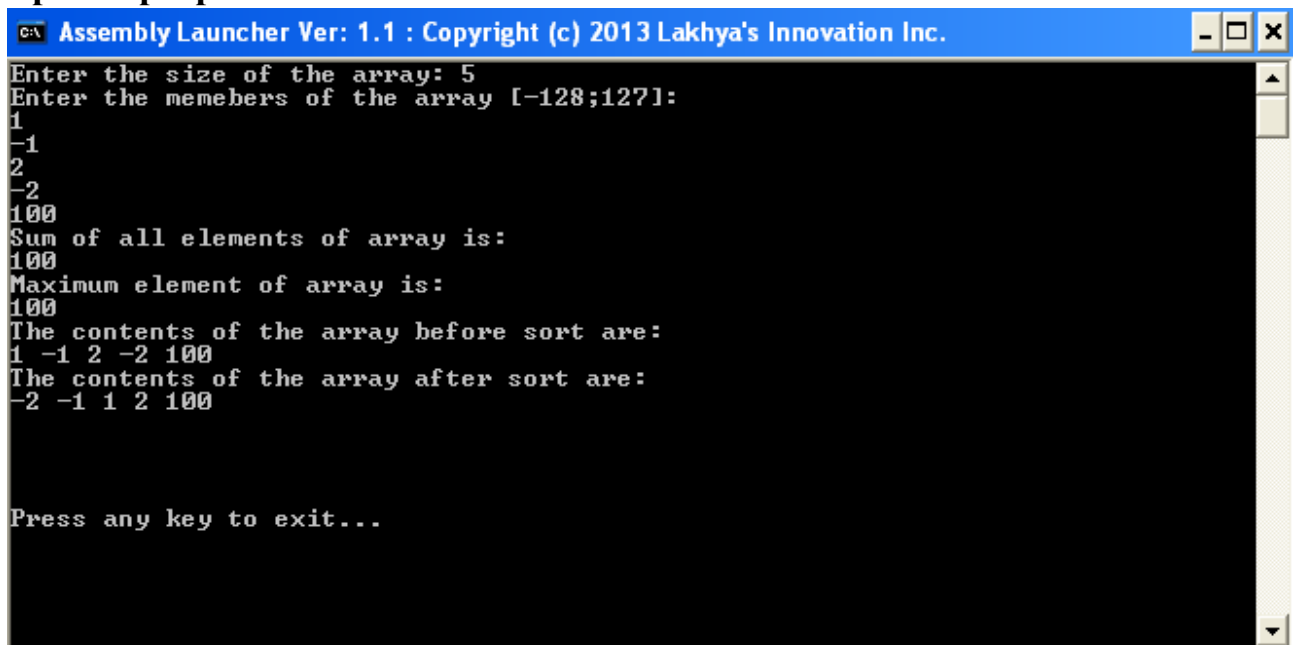


Assembly Launcher Ver: 1.1 : Copyright (c) 2013 Lakhya's Innovation Inc.

```
Please, enter x 3
Please, enter a 3
Please, enter b 1
82/3=27 zalishok 1

Press any key to exit...
```

Третя програма:



Assembly Launcher Ver: 1.1 : Copyright (c) 2013 Lakhya's Innovation Inc.

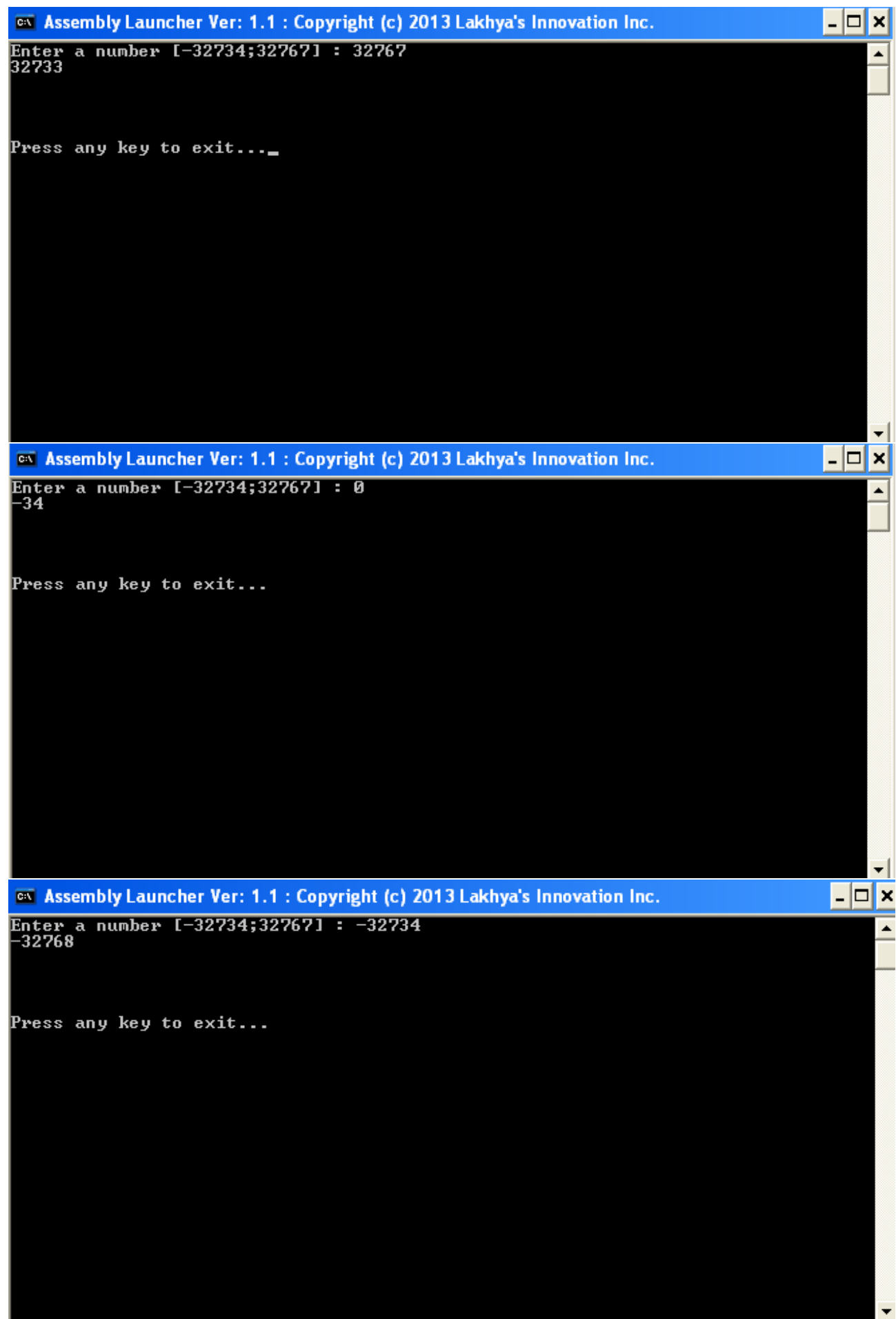
```
Enter the size of the array: 5
Enter the members of the array [-128;127]:
1
-1
2
-2
100
Sum of all elements of array is:
100
Maximum element of array is:
100
The contents of the array before sort are:
1 -1 2 -2 100
The contents of the array after sort are:
-2 -1 1 2 100

Press any key to exit...
```

Перевірочні дані:

Перша програма:





Друга програма:

```
C:\ Assembly Launcher Ver: 1.1 : Copyright (c) 2013 Lakhya's Innovation Inc.
Please, enter x 14
Please, enter a 1
Please, enter b 1
2745/14=196 zalishok 1

Press any key to exit...
```

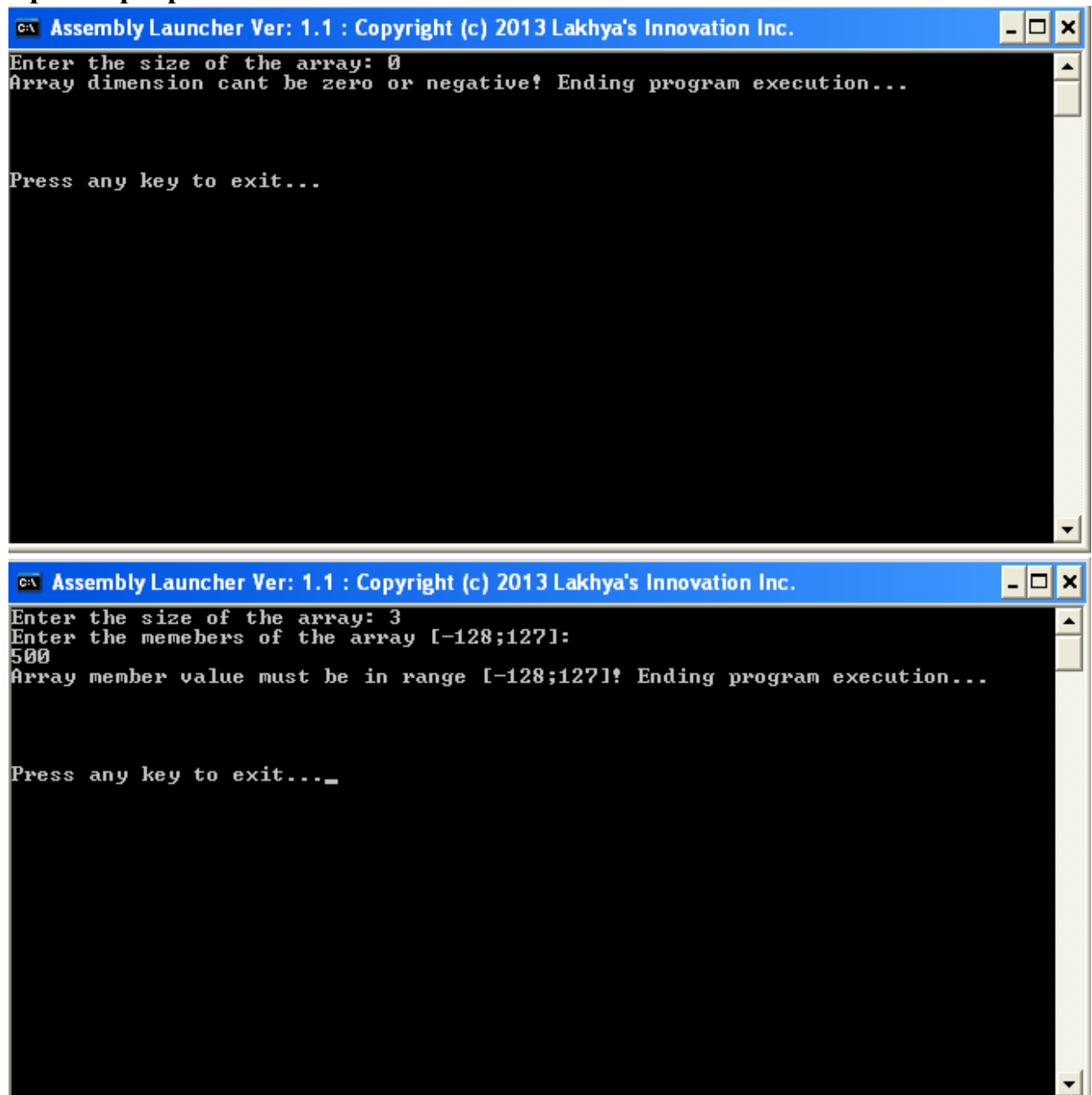
```
C:\ Assembly Launcher Ver: 1.1 : Copyright (c) 2013 Lakhya's Innovation Inc.
Please, enter x -14
Please, enter a 1
Please, enter b 1
210

Press any key to exit..._
```

```
C:\ Assembly Launcher Ver: 1.1 : Copyright (c) 2013 Lakhya's Innovation Inc.
Please, enter x 0
Please, enter a 14
Please, enter b 14
42

Press any key to exit..._
```

### Третя програма:



### Висновок:

Отож, у ході виконання лабораторної роботи було проаналізовано поставлені завдання, ознайомлено з теоретичної базою завдання та створено програмне забезпечення, що являє собою попердні комп'ютерні практикуми, модифіковані із застосуванням макросів різного характеру за функціональністю. У рамках виконання поставленого завдання було створено низку макросів: макрос виводу переданого значення для КП 1, макрос обрахунку значення функції для КП 2 та макрос для обчислення суми елементів для КП 3. Макроси для підзавдань було спроектовано з передбаченням можливого повторного застосування в програмі за допомогою оголошення локальних міток. Систему було побудовано з урахуванням найвірогідніших відмов: неправильного вводу, переповнення, ін.. Під час виконання лабораторної

роботи було набуто практичних навичок роботи з макрокомандами, специфікою їхнього застосування та функціональностей, що вони надають. Урешті-решт, було проведено тестування створеного ПЗ та побудовано на основі коду програм блок-схеми.