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 ІПІ
Пархоменко А.В.
"09" червня 2023 р.

Виконав
Студент групи ІП-14
Хільчук А.В.

Київ – 2023

Комп’ютерний практикум №4

Тема: масиви.

Завдання:

1. Написати програму, яка повинна мати наступний функціонал:
   1. Можливість введення користувачем розміру одномірного масиву.
   2. Можливість введення користувачем значень елементів одномірного масиву.
   3. Можливість знаходження суми елементів одномірного масиву.
   4. Можливість пошуку максимального (або мінімального) елемента одномірного масиву.
   5. Можливість сортування одномірного масиву цілих чисел загального вигляду.
2. Написати програму, яка буде мати наступний функціонал:
   1. Можливість введення користувачем розміру двомірного масиву.
   2. Можливість введення користувачем значень елементів двомірного масиву.
   3. Можливість пошуку координат всіх входжень заданого елемента в двомірному масиві, елементи масиву та пошуковий елемент вводить користувач.
3. Програма повинна мати захист від некоректного введення вхідних даних (символи,
переповнення і т.і.)

Виконання:

Текст програм:

Перша програма:

```
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 memebers 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:$'
   overflow_msg DB 0Ah, 0Dh, 'Overflow has occured!Stopping the
execution...$'
```

```
    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

    call find_sum
    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

    MOV AH, 49h
    MOV BX, my_array
    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

find_sum proc
```

```
    mov cx, array_size
    mov di,my_array
    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
    ret
sum_overflow:
    MOV AH, 09h
    LEA DX, overflow_msg
    INT 21h
    call stop_exec
find_sum 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
    mov al,bl
    cbw
    mov bx,ax
    call print_bx
    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
```

Друга програма:

```
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 number of rows of the matrix: $'
    msg_ask_for_input DB 0Dh, 0Ah,'Enter the memebers of the matrix [-
128;127]: $'
    msg1 DB 'Enter the count of rows of the matrix: $'
    msg2 DB 0Ah, 0Dh,'Enter the number of columns of the matrix: $'
    msg3 DB 0Ah, 0Dh, 'The contents of the matrix are:$'
    msg4 DB 0Ah, 0Dh, 'Enter the number to look for all occurences of:$'
    msg5 DB 0Ah, 0Dh, 'Search begins:$'
    msg6 DB 0Ah, 0Dh, 'Search has ended$'
```

```
    msg7 DB 0Ah, 0Dh, 'Matrix dimension cant be zero or negative! Ending
program execution...$'
    msg8 DB 0Ah, 0Dh, 'Resulting matrix:$'
    msg9 DB 0Ah, 0Dh, 'Member value must be in range [-128;127]! Ending
program execution...$'
    msg10 DB 0Ah, 0Dh, 'No occurences found!$'
    newline DB 0Dh, 0Ah,'$'
    row_count DW ?
    column_count dw ?
    total_size dw ?
    bufer DB 7, 0, 7 dup (0) ; variable to store the input number
    my_array Dw ?
    check_offset dw ?
    check_value db 1
    found_flag 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 ; set DS to point to DSEG

    call init_matrix

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

    call single_value_input
    mov check_value,al
    call find_all_occurences

    ; free the memory allocated for the array
    MOV AH, 49h  ; free memory function
    MOV BX, my_array
    INT 21h

    MOV AH, 4Ch
    INT 21h
main endp

find_all_occurences proc
```

```
    MOV AH, 09h
    LEA DX, msg5
    INT 21h

    mov check_offset,0
    mov cx, total_size
    mov si,my_array
occurences_search_loop:

    mov al,check_value
    cmp al,[si]
    jne occurences_search_loop_end
    push cx
    mov found_flag,1
    MOV AH, 09h
    LEA DX, newline
    INT 21h
    mov ax,check_offset
    mov dx,0
    div column_count
    call print_pair_ax_dx
    pop cx
occurences_search_loop_end:
    inc si
    inc check_offset
    loop occurences_search_loop
    MOV AH, 09h
    LEA DX, msg6
    INT 21h
    cmp found_flag, 1
    je occurence_end
    MOV AH, 09h
    LEA DX, msg10
    INT 21h
occurence_end:
    ret
find_all_occurences endp

print_pair_ax_dx proc
    push dx
    push ax
    mov al,'(' ; ???? ????????, ?? ?????
    int 29h ; ????? '?????'
    pop bx
```

```
    call print_bx
    mov al,',' ; ???? ????????, ?? ?????
    int 29h ; ????? '?????'
    pop bx
    call print_bx
    mov al,')' ; ???? ????????, ?? ?????
    int 29h ; ????? '?????'
    ret
print_pair_ax_dx endp

print_matrix proc
    MOV AH, 09h
    LEA DX, msg8
    INT 21h
    mov di,my_array
    mov cx, row_count
print_rows_loop:
    MOV AH, 09h
    LEA DX, newline
    INT 21h
    mov ax, column_count
    push cx
    print_columns_row:
        push ax
        mov bl,byte ptr [di]
        call print_bl
        inc di
        mov dl, ' '
        mov ah, 02h
        int 21h
        pop ax
        dec ax
        jnz print_columns_row
    pop cx
    loop print_rows_loop
    ret
    print_matrix endp

init_matrix proc
    ; ask the user for the count of rows
    MOV AH, 09h
    LEA DX, msg1
    INT 21h
    call single_dimension_input
```

```
    MOV row_count,AX
    ; ask the user for the count of columns
    MOV AH, 09h
    LEA DX, msg2
    INT 21h

    call single_dimension_input
    MOV column_count,AX

    mul row_count
    mov total_size,ax


    MOV AH, 48h
    MOV BX, total_size
    INT 21h
    MOV my_array, AX

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

    mov cx, total_size
    mov di,my_array
matrix_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 matrix_member_input_loop
    ret
init_matrix 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, msg9
    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, msg7
    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
    ;NO_MINUS:
    FOR_LOOP_VALIDATION:
```

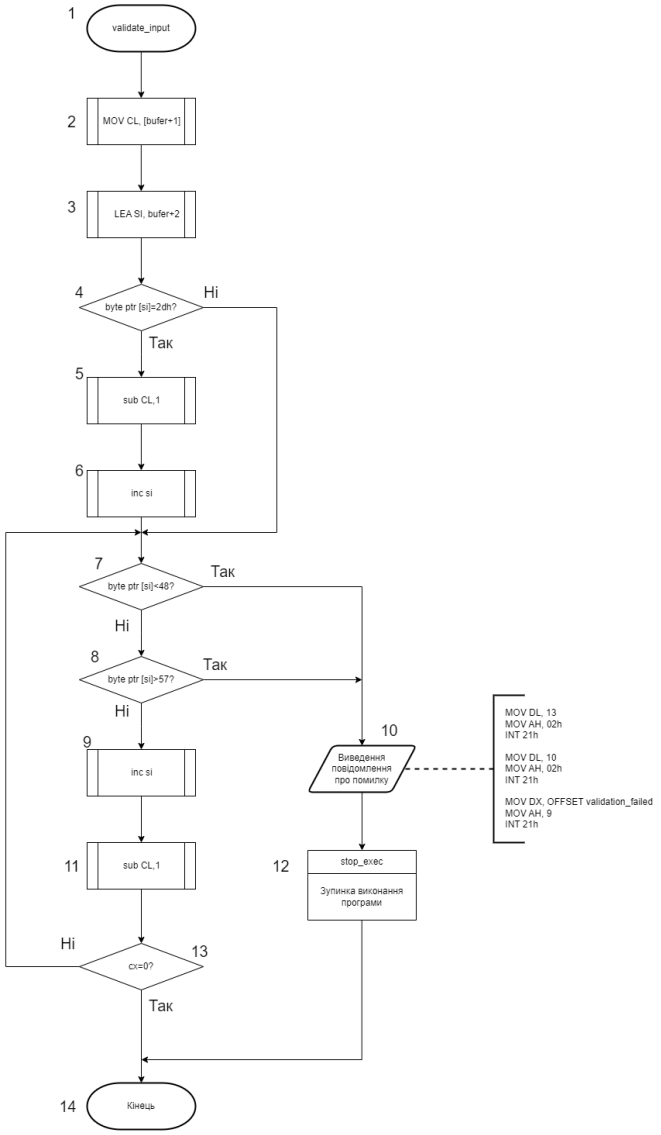
```
    cmp byte ptr [si],48
    JL VALIDATION_FAILURE
    cmp byte ptr [si],58
    JGE VALIDATION_FAILURE

    inc si;
    LOOP FOR_LOOP_VALIDATION    ; decrement CX and jump to
FOR_LOOP if CX is not zero
    ret
VALIDATION_FAILURE:

    MOV DL, 13   ; Carriage Return
    MOV AH, 02h  ; Function code for printing character
    INT 21h

    MOV DL, 10   ; Line Feed
    MOV AH, 02h  ; Function code for printing character
    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    ; decrement CX and jump to
FOR_LOOP if CX is not zero
    ret
bufer_chars_to_digits ENDP
```

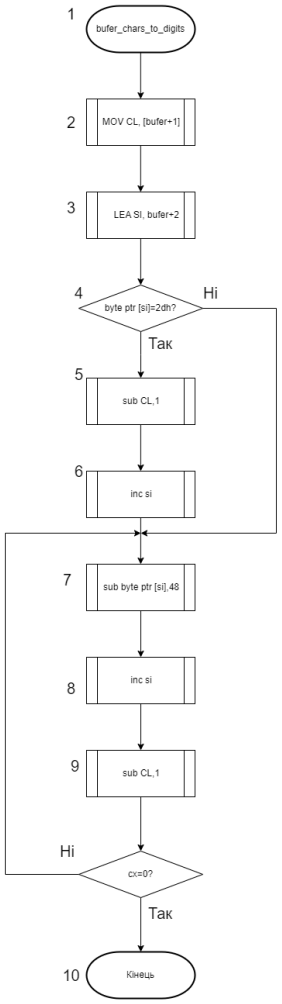
```
bufer_digits_to_number PROC

    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

stop_exec proc
    MOV AH, 4Ch   ; return to DOS function
    INT 21h       ; interrupt 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
```

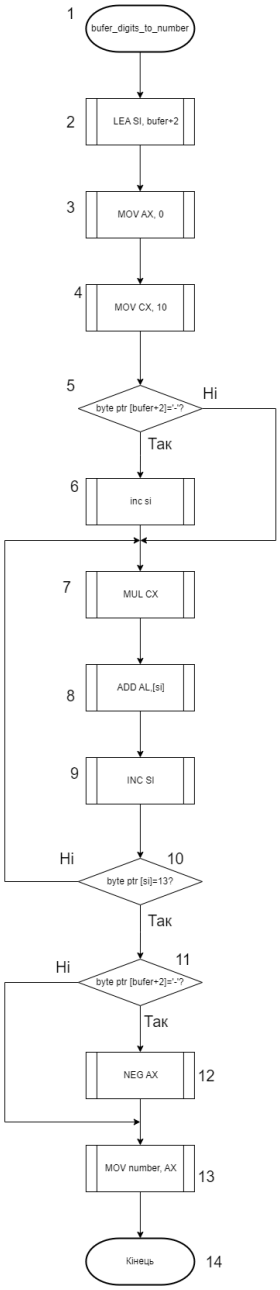
```
    div bx
    add dl,'0'
    push dx
    inc cx
    test ax,ax
    jnz m2
m3:
    pop ax
    int 29h
    loop m3
ret
print_bx endp

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
    mov al,bl
    cbw
    mov bx,ax
    call print_bx
    ret
print_bl endp

CSEG ENDS
END MAIN
```

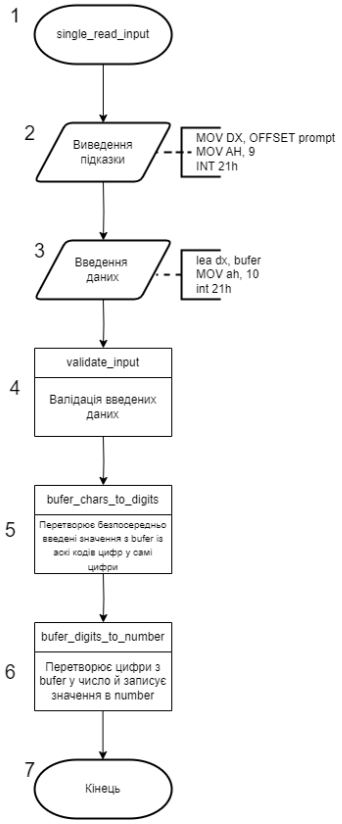
<u>Схема функціонування програми:</u>

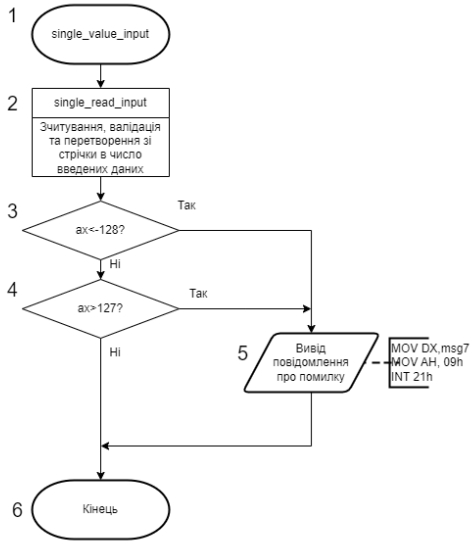
Перша програма:


1
validate_input
2
MOV CL, [bufer+1]
3
LEA SI, bufer+2
4
byte ptr [si]=2dh?
Ні
Так
5
sub CL,1
6
inc si
7
byte ptr [si]<48?
Так
Ні
8
byte ptr [si]>57?
Так
Ні
9
inc si
10
Виведення повідомлення про помилку
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
11
sub CL,1
12
stop_exec
Зупинка виконання програми
Ні
13
cx=0?
Так
14
Кінець

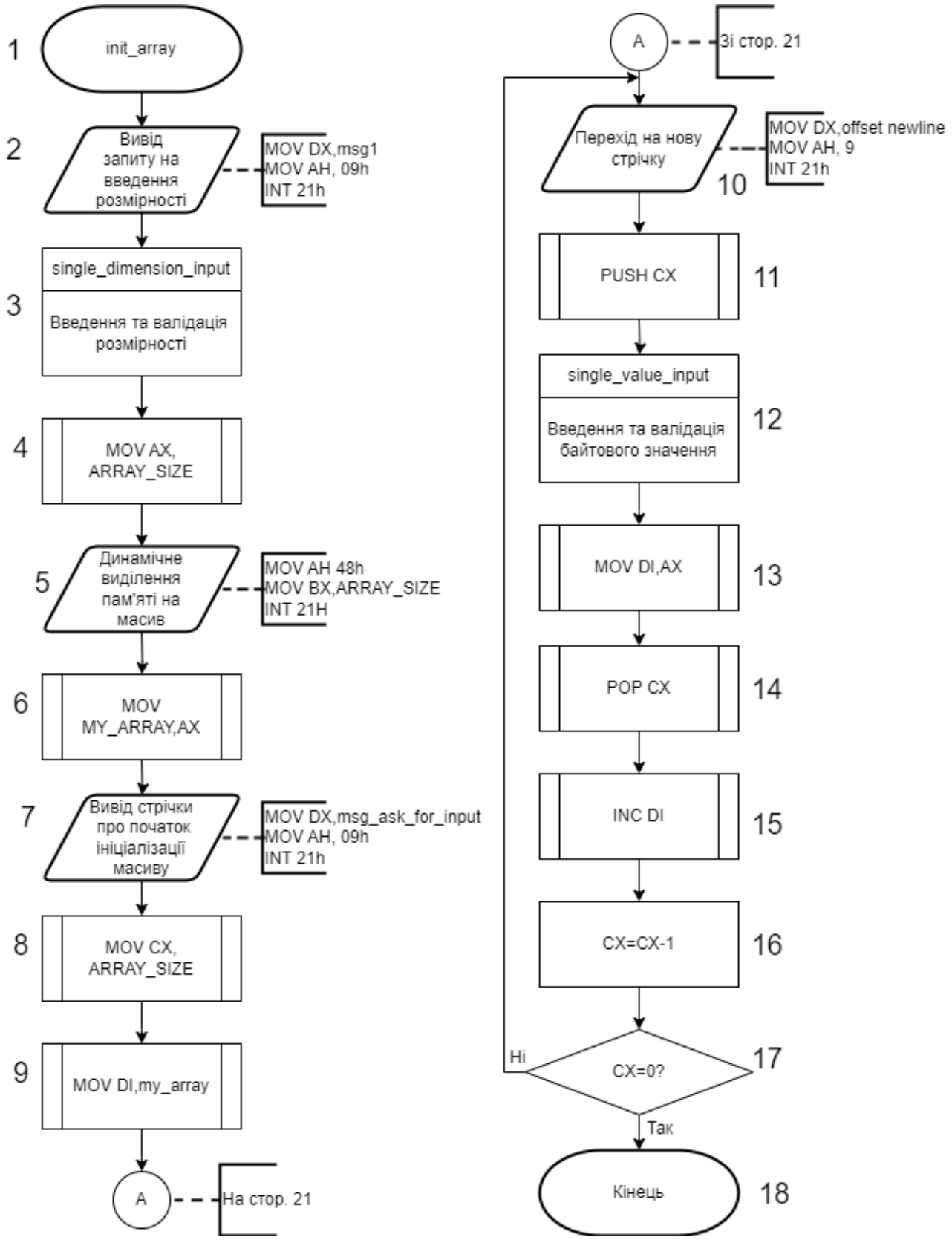

1
bufer_chars_to_digits
2
MOV CL, [bufer+1]
3
LEA SI, bufer+2
4
byte ptr [si]=2dh?
Ні
Так
5
sub CL,1
6
inc si
7
sub byte ptr [si],48
8
inc si
9
sub CL,1
Ні
cx=0?
Так
10
Кінець

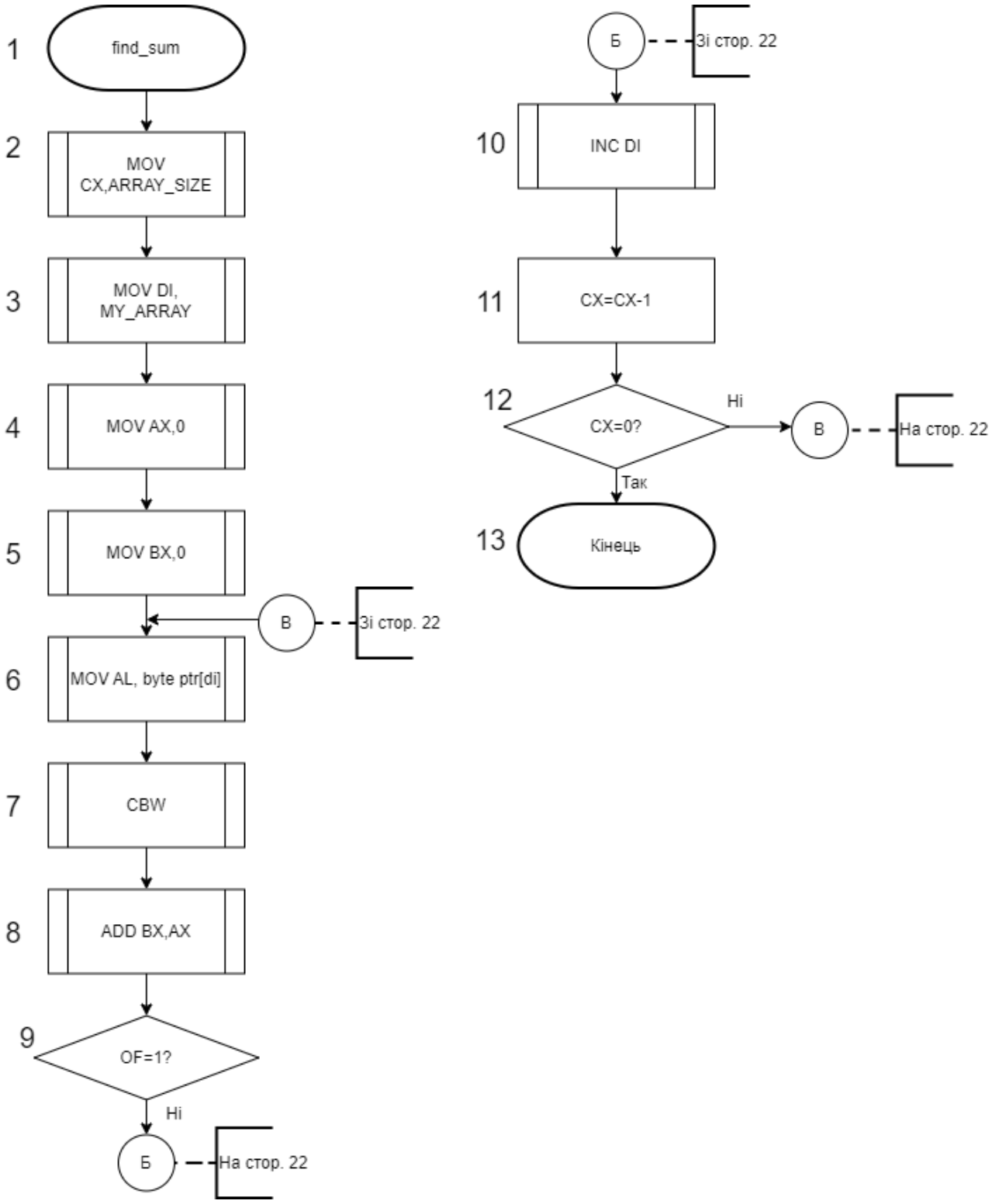

1
bufer_digits_to_number
2
LEA SI, bufer+2
3
MOV AX, 0
4
MOV CX, 10
5
byte ptr [bufer+2]='-'?
Ні
Так
6
inc si
7
MUL CX
8
ADD AL,[si]
9
INC SI
Ні
10
byte ptr [si]=13?
Так
Ні
11
byte ptr [bufer+2]='-'?
Так
NEG AX
12
MOV number, AX
13
Кінець
14

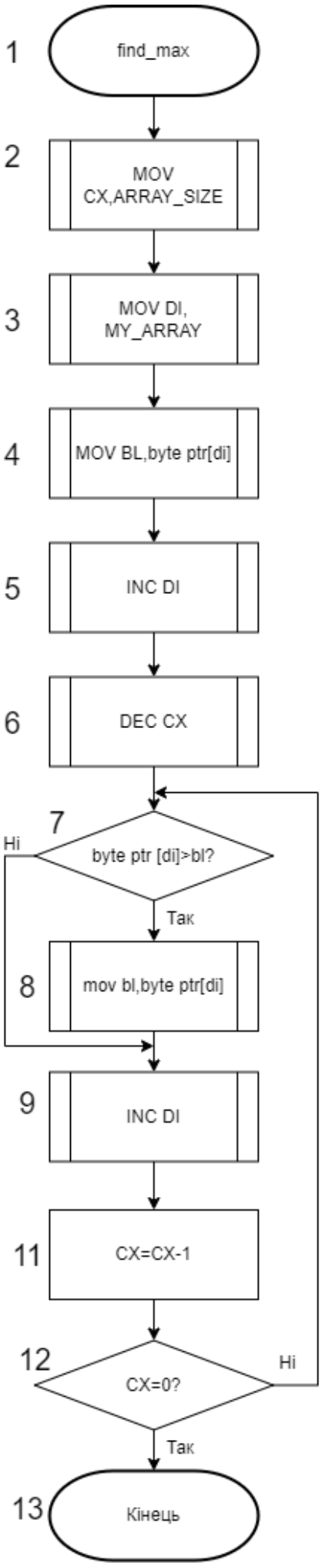

1
single_read_input
2
Виведення підказки
MOV DX, OFFSET prompt
MOV AH, 9
INT 21h
3
Введення даних
lea dx, bufer
MOV ah, 10
int 21h
4
validate_input
Валідація введених даних
5
bufer_chars_to_digits
Перетворює безпосередньо введені значення з bufer із аскі кодів цифр у самі цифри
6
bufer_digits_to_number
Перетворює цифри з bufer у число й записує значення в number
7
Кінець

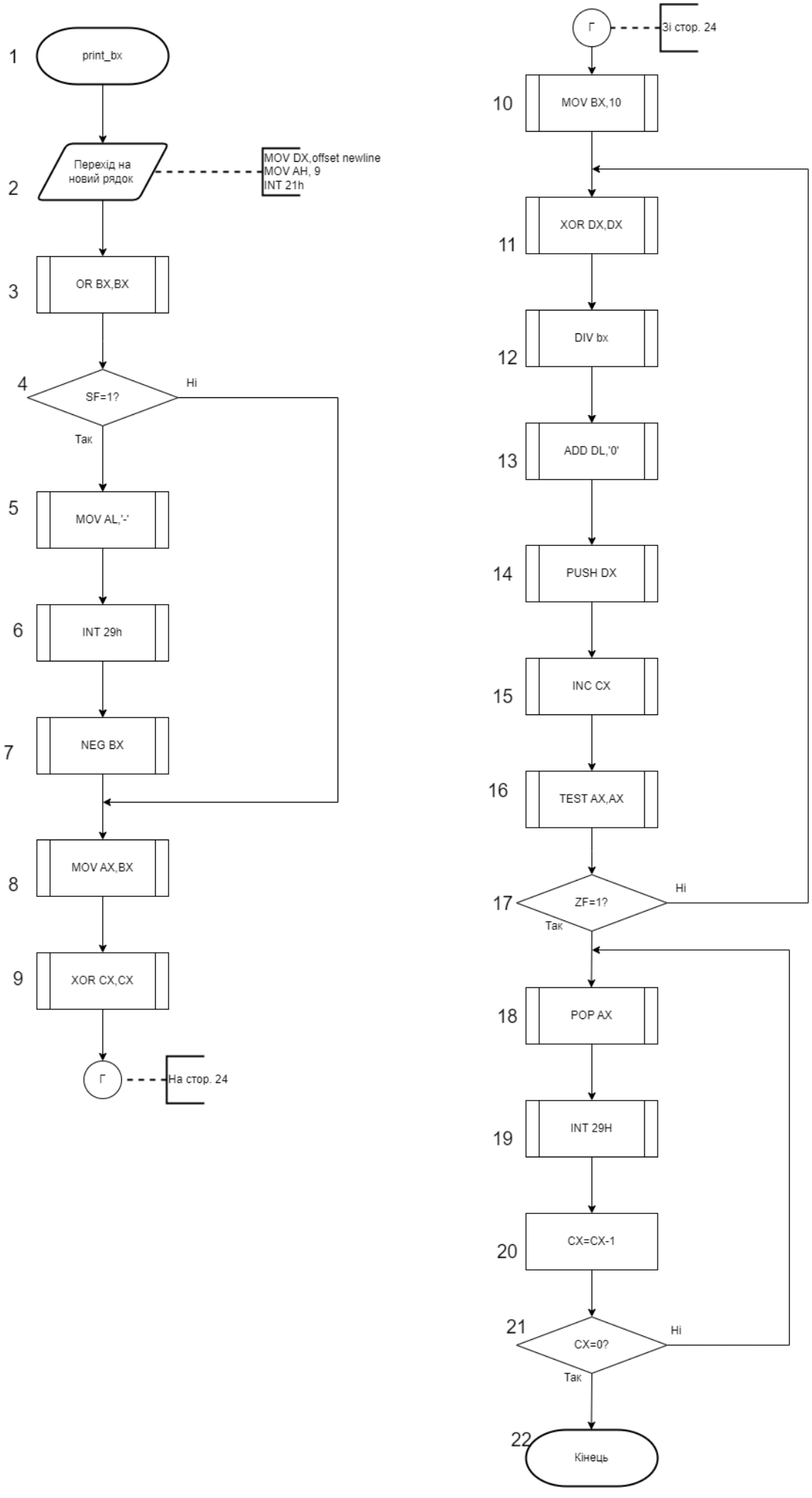

1
single_value_input
2
single_read_input
Зчитування, валідація та перетворення зі стрічки в число введених даних
3
ax<-128?
Так
Ні
4
ax>127?
Так
Ні
5
Вивід повідомлення про помилку
MOV DX,msg7
MOV AH, 09h
INT 21h
6
Кінець

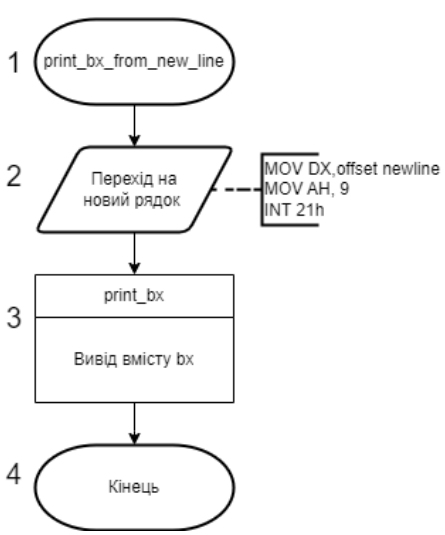

1
single_dimension_input
2
single_read_input
Зчитування, валідація та перетворення зі стрічки в число введених даних
3
ax<0?
Так
Ні
4
Вивід повідомлення про помилку
MOV DX,msg6
MOV AH, 09h
INT 21h
5
Кінець

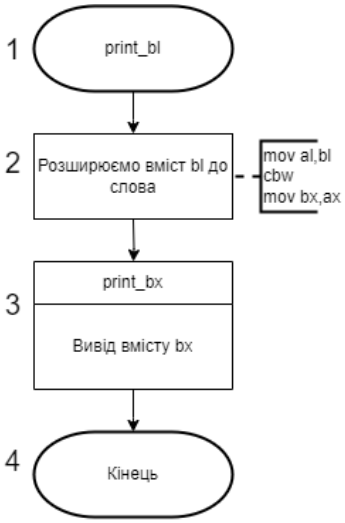

1
init_array
2
Вивід запиту на введення розмірності
MOV DX,msg1
MOV AH, 09h
INT 21h
3
single_dimension_input
Введення та валідація розмірності
4
MOV AX, ARRAY_SIZE
5
Динамічне виділення пам'яті на масив
MOV AH 48h
MOV BX,ARRAY_SIZE
INT 21H
6
MOV MY_ARRAY,AX
7
Вивід стрічки про початок ініціалізації масиву
MOV DX,msg_ask_for_input
MOV AH, 09h
INT 21h
8
MOV CX, ARRAY_SIZE
9
MOV DI,my_array
A
На стор. 21
A
З і стор. 21
Перехід на нову стрічку
MOV DX,offset newline
MOV AH, 9
INT 21h
10
PUSH CX
11
single_value_input
Введення та валідація байтового значення
12
MOV DI,AX
13
POP CX
14
INC DI
15
CX=CX-1
16
Ні
CX=0?
17
Так
Кінець
18

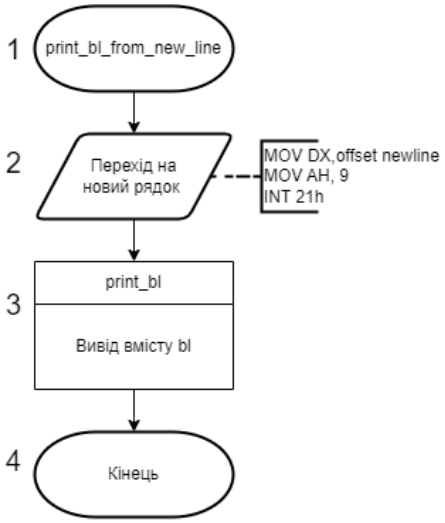

1
find_sum
2
MOV CX,ARRAY_SIZE
3
MOV DI, MY_ARRAY
4
MOV AX,0
5
MOV BX,0
В
3і стор. 22
6
MOV AL, byte ptr[di]
7
CBW
8
ADD BX,AX
9
OF=1?
Ні
Б
На стор. 22
Б
3і стор. 22
10
INC DI
11
CX=CX-1
12
CX=0?
Ні
В
На стор. 22
Так
13
Кінець

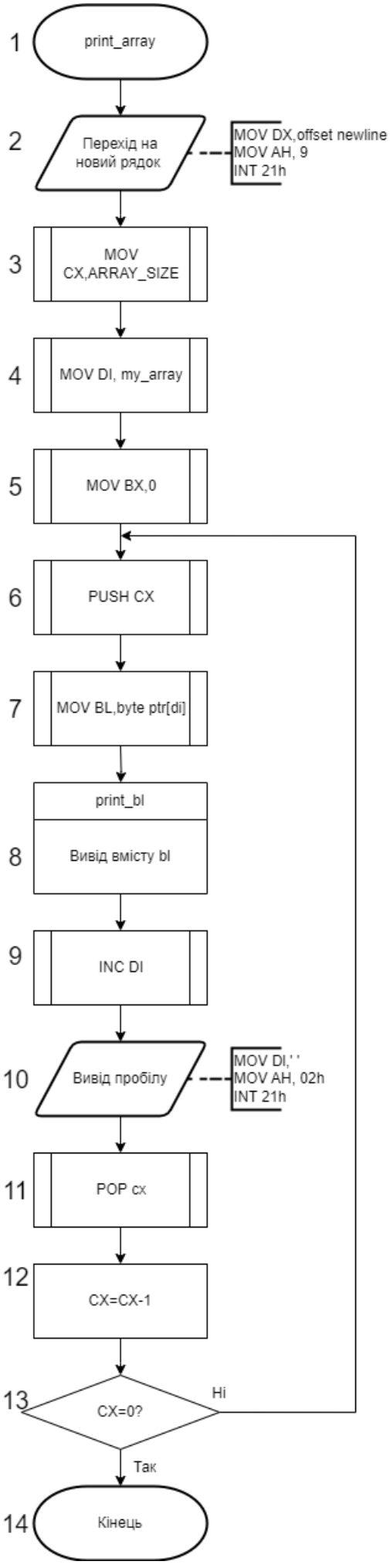

1
find_max
2
MOV CX,ARRAY_SIZE
3
MOV DI, MY_ARRAY
4
MOV BL,byte ptr[di]
5
INC DI
6
DEC CX
7
byte ptr [di]>bl?
Ні
Так
8
mov bl,byte ptr[di]
9
INC DI
11
CX=CX-1
12
CX=0?
Ні
Так
13
Кінець

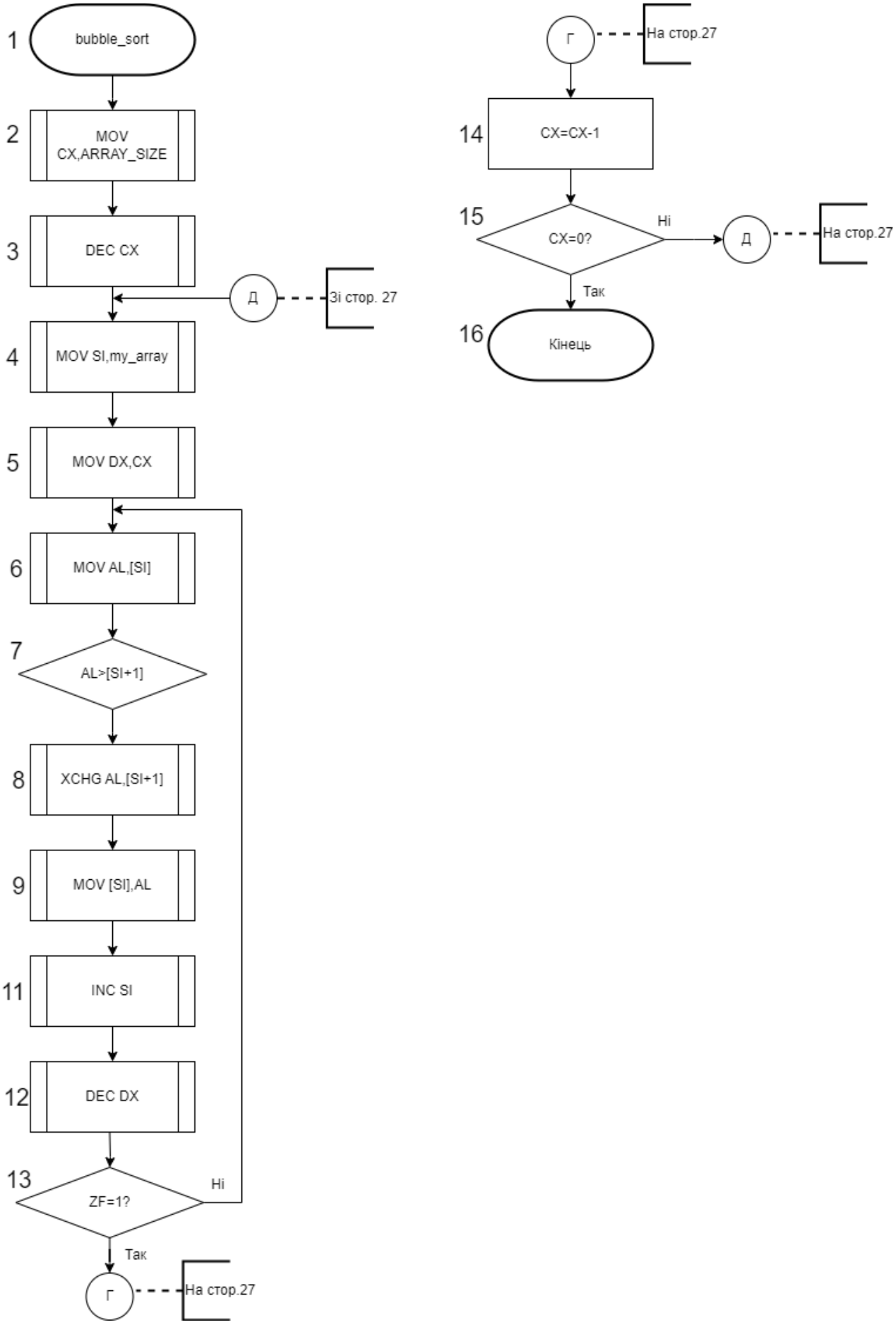

1
print_bx
2
Перехід на новий рядок
MOV DX,offset newline
MOV AH, 9
INT 21h
3
OR BX,BX
4
SF=1?
Ні
Так
5
MOV AL,'-'
6
INT 29h
7
NEG BX
8
MOV AX,BX
9
XOR CX,CX
Г
На стор. 24
Г
Зі стор. 24
10
MOV BX,10
11
XOR DX,DX
12
DIV bx
13
ADD DL,'0'
14
PUSH DX
15
INC CX
16
TEST AX,AX
17
ZF=1?
Ні
Так
18
POP AX
19
INT 29H
20
CX=CX-1
21
CX=0?
Ні
Так
22
Кінець

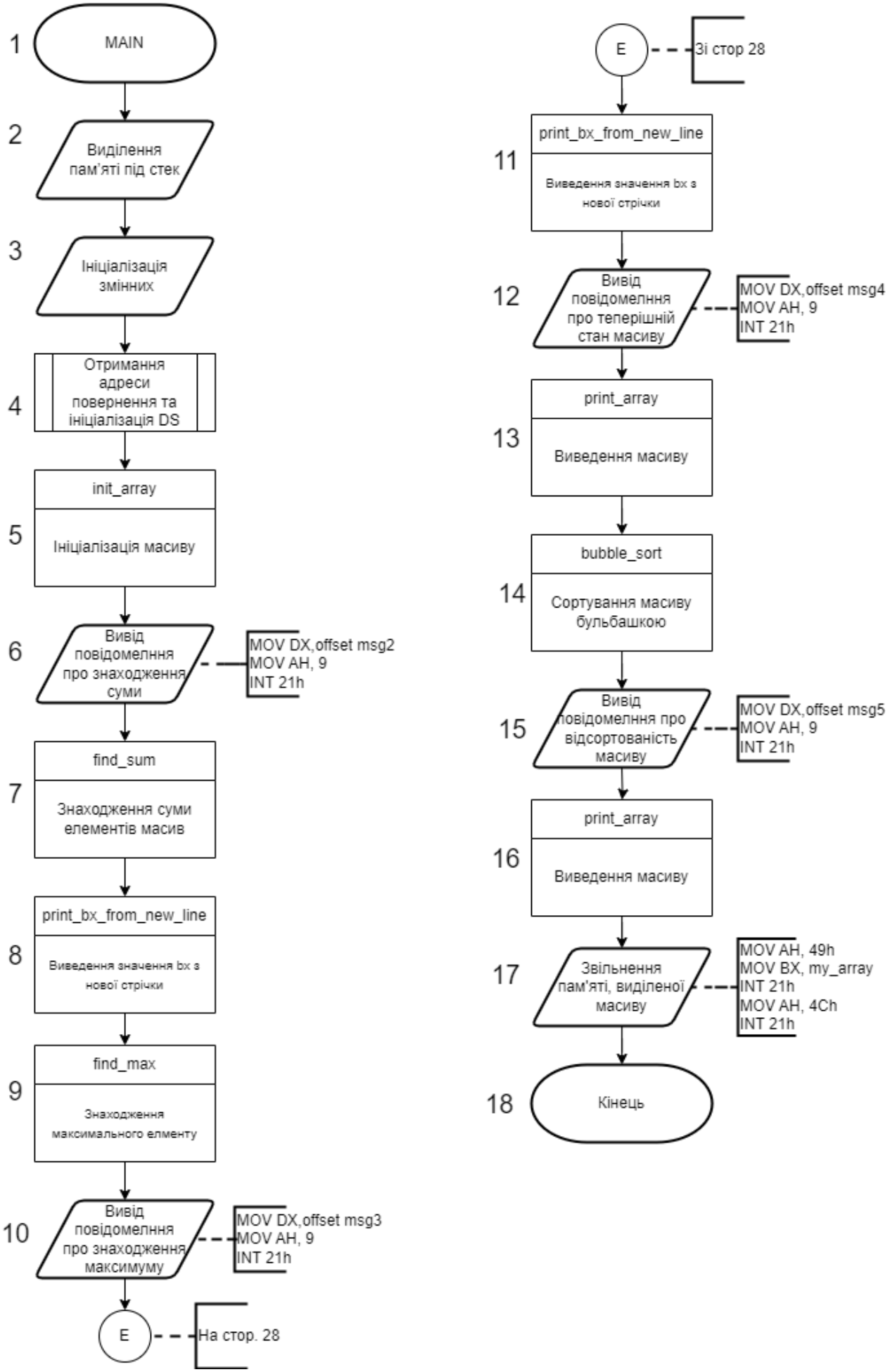

1
print_bx_from_new_line
2
Перехід на новий рядок
MOV DX,offset newline
MOV AH, 9
INT 21h
3
print_bx
Вивід вмісту bx
4
Кінець


1
print_bl
2
Розширюємо вміст bl до слова
mov al,bl
cbw
mov bx,ax
3
print_bx
Вивід вмісту bx
4
Кінець

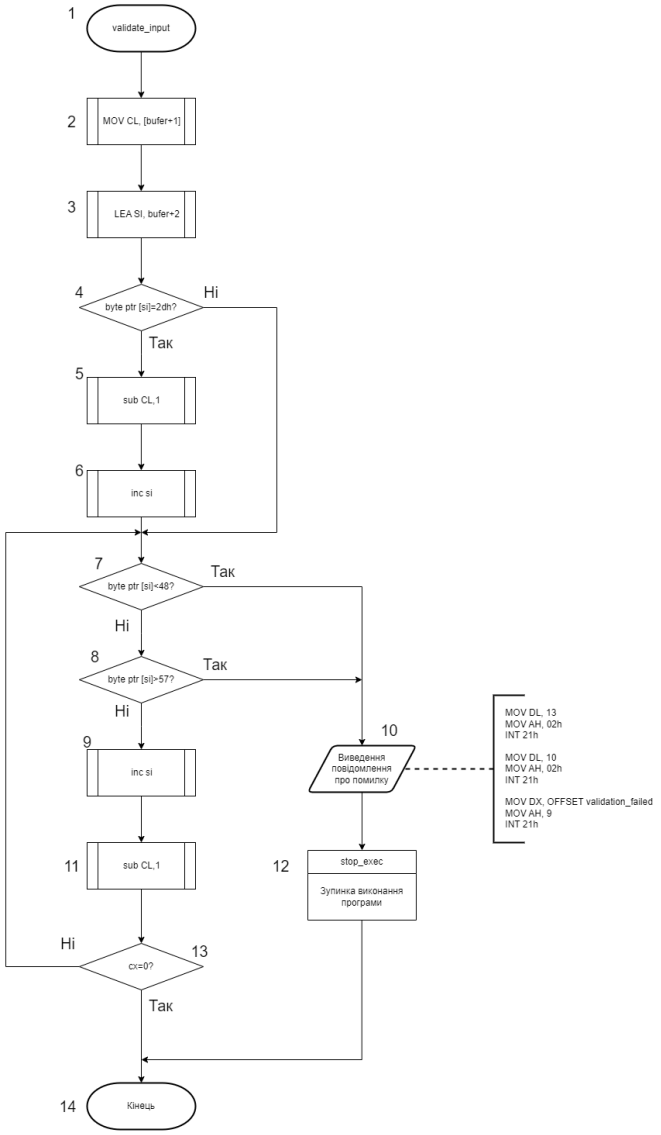

1
print_bl_from_new_line
2
Перехід на новий рядок
MOV DX,offset newline
MOV AH, 9
INT 21h
3
print_bl
Вивід вмісту bl
4
Кінець

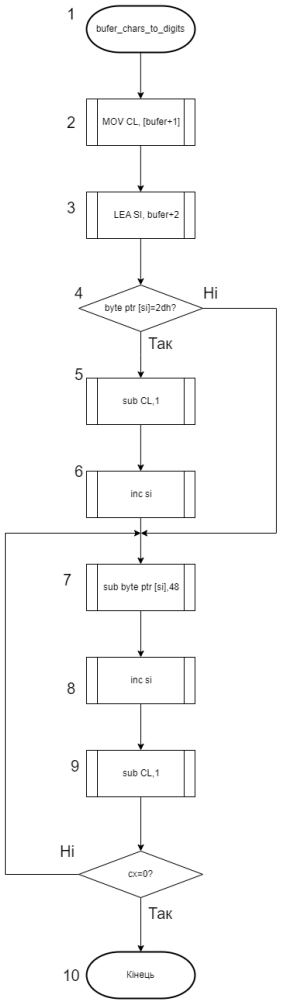

1
print_array
2
Перехід на новий рядок
MOV DX,offset newline
MOV AH, 9
INT 21h
3
MOV CX,ARRAY_SIZE
4
MOV DI, my_array
5
MOV BX,0
6
PUSH CX
7
MOV BL,byte ptr[di]
print_bl
8
Вивід вмісту bl
9
INC DI
10
Вивід пробілу
MOV DI,' '
MOV AH, 02h
INT 21h
11
POP cx
12
CX=CX-1
13
CX=0?
Ні
Так
14
Кінець

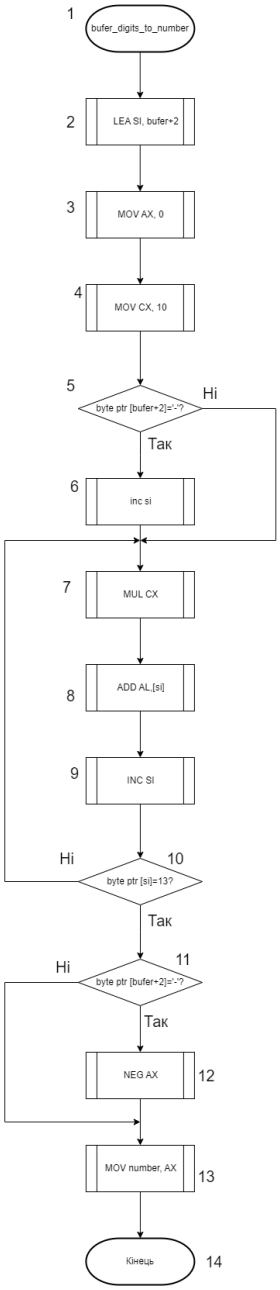

1
bubble_sort
2
MOV CX,ARRAY_SIZE
3
DEC CX
Д
Зі стор. 27
4
MOV SI,my_array
5
MOV DX,CX
6
MOV AL,[SI]
7
AL>[SI+1]
8
XCHG AL,[SI+1]
9
MOV [SI],AL
11
INC SI
12
DEC DX
13
ZF=1?
Ні
Так
Г
На стор.27
Г
На стор.27
14
CX=CX-1
15
CX=0?
Ні
Д
На стор.27
Так
16
Кінець

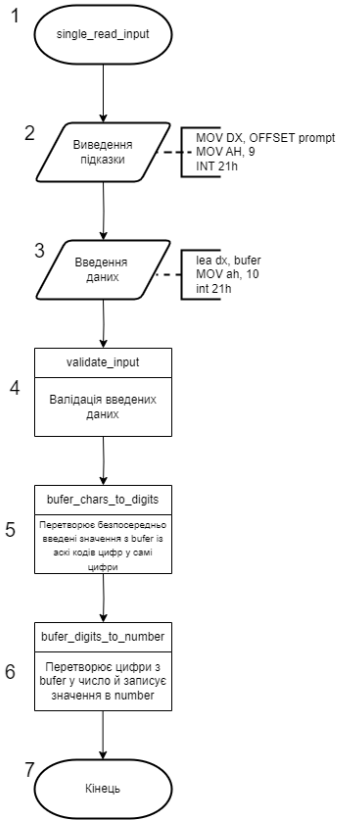


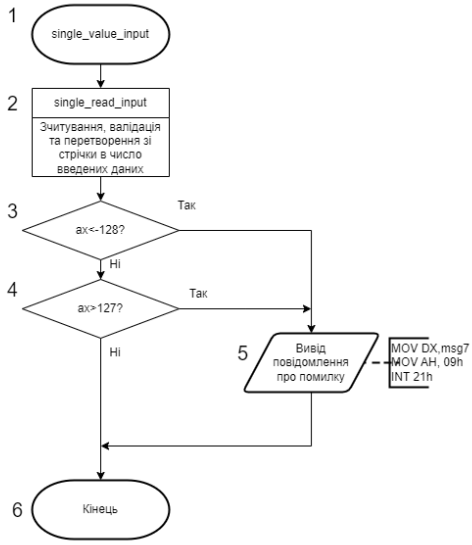
Друга програма:


1
validate_input
2
MOV CL, [bufer+1]
3
LEA SI, bufer+2
4
byte ptr [si]=2dh?
Ні
Так
5
sub CL,1
6
inc si
7
byte ptr [si]<48?
Так
Ні
8
byte ptr [si]>57?
Так
Ні
9
inc si
10
Виведення повідомлення про помилку
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
11
sub CL,1
12
stop_exec
Зупинка виконання програми
Ні
13
cx=0?
Так
14
Кінець

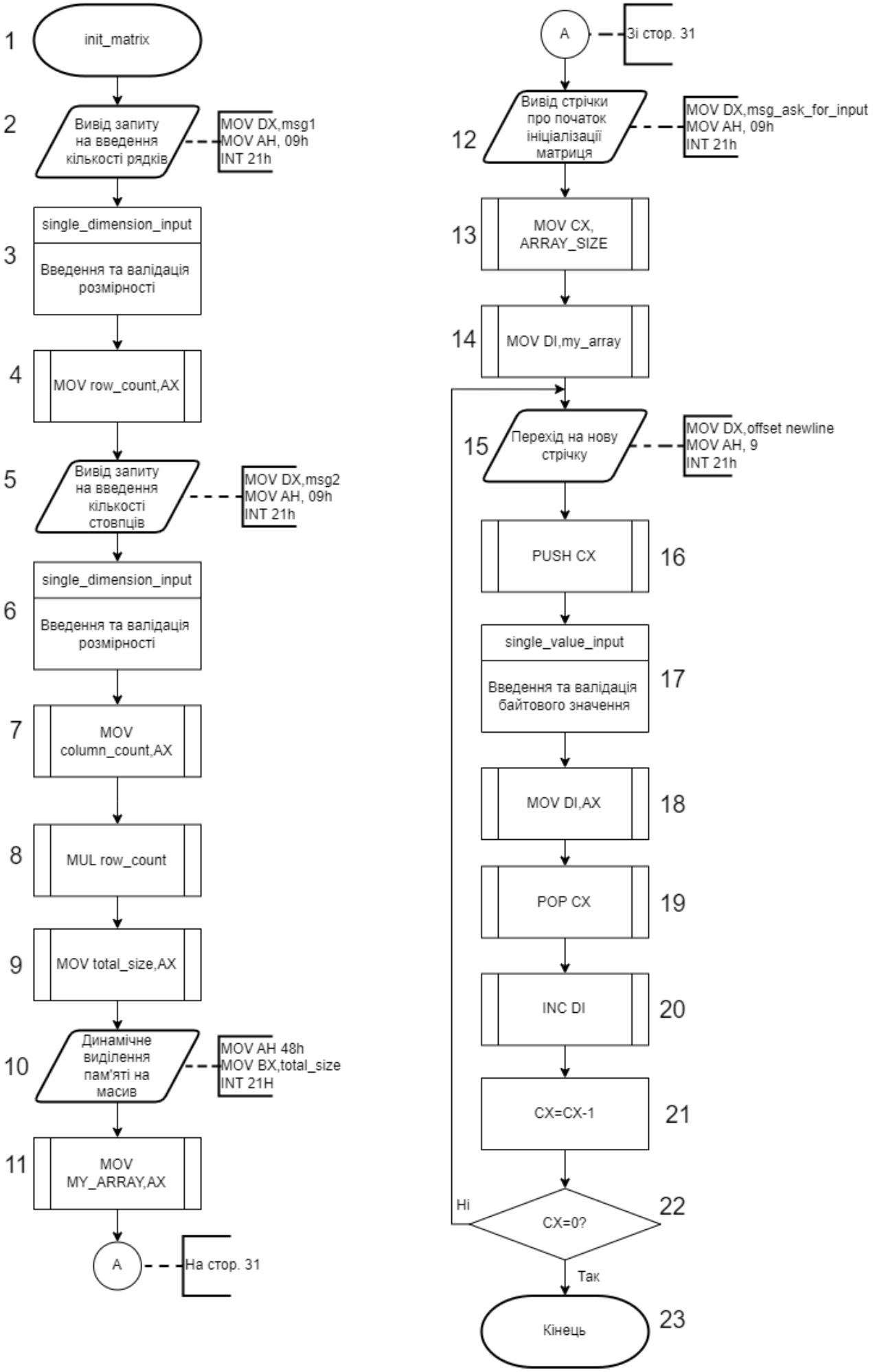

1
bufer_chars_to_digits
2
MOV CL, [bufer+1]
3
LEA SI, bufer+2
4
byte ptr [si]=2dh?
Ні
Так
5
sub CL,1
6
inc si
7
sub byte ptr [si],48
8
inc si
9
sub CL,1
Ні
cx=0?
Так
10
Кінець

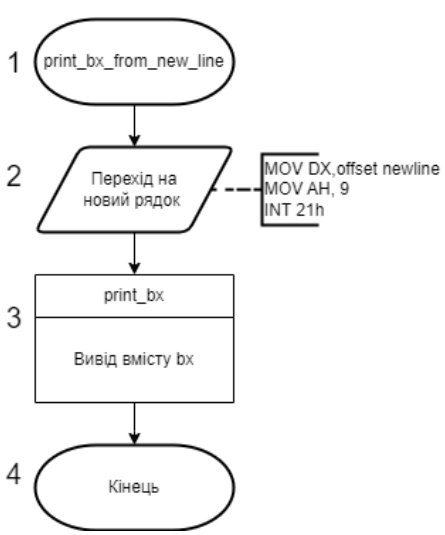

1
bufer_digits_to_number
2
LEA SI, bufer+2
3
MOV AX, 0
4
MOV CX, 10
5
byte ptr [bufer+2]='-'?
Ні
Так
6
inc si
7
MUL CX
8
ADD AL,[si]
9
INC SI
Ні
10
byte ptr [si]=13?
Так
Ні
11
byte ptr [bufer+2]='-'?
Так
NEG AX
12
MOV number, AX
13
Кінець
14

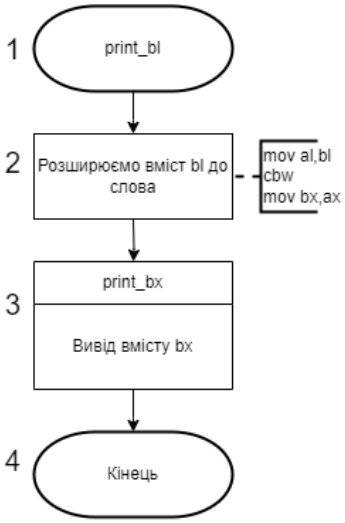

1
single_read_input
2
Виведення підказки
MOV DX, OFFSET prompt
MOV AH, 9
INT 21h
3
Введення даних
lea dx, bufer
MOV ah, 10
int 21h
4
validate_input
Валідація введених даних
5
bufer_chars_to_digits
Перетворює безпосередньо введені значення з bufer із аскі кодів цифр у самі цифри
6
bufer_digits_to_number
Перетворює цифри з bufer у число й записує значення в number
7
Кінець

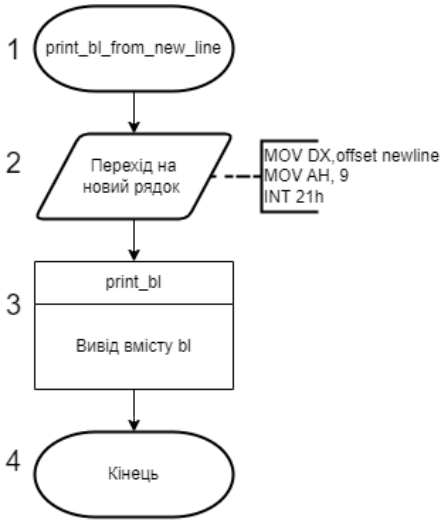

1
single_value_input
2
single_read_input
Зчитування, валідація та перетворення зі стрічки в число введених даних
3
ax<-128?
Так
Ні
4
ax>127?
Так
Ні
5
Вивід повідомлення про помилку
MOV DX,msg7
MOV AH, 09h
INT 21h
6
Кінець

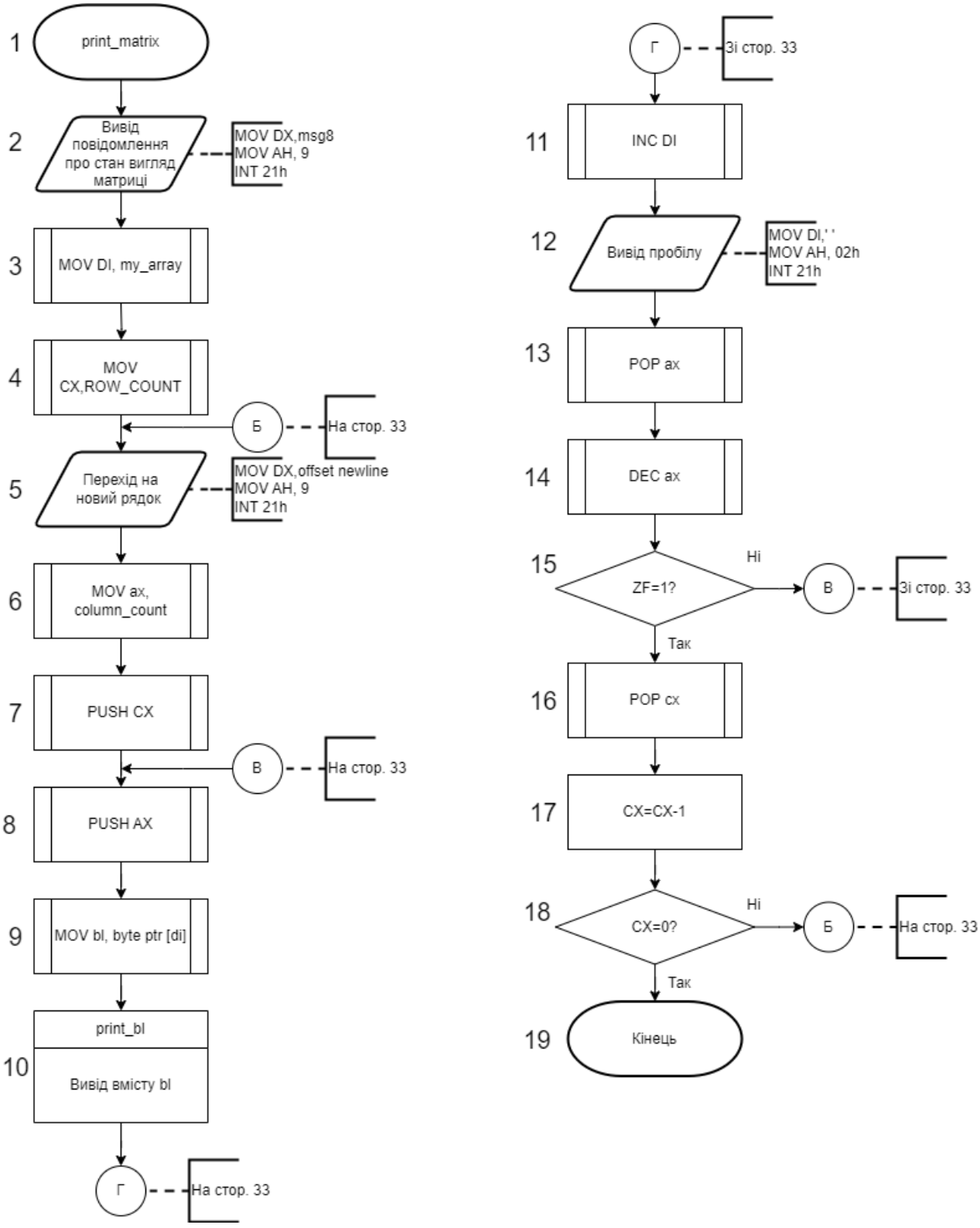

1
single_dimension_input
2
single_read_input
Зчитування, валідація та перетворення зі стрічки в число введених даних
3
ax<0?
Так
Ні
4
Вивід повідомлення про помилку
MOV DX,msg6
MOV AH, 09h
INT 21h
5
Кінець

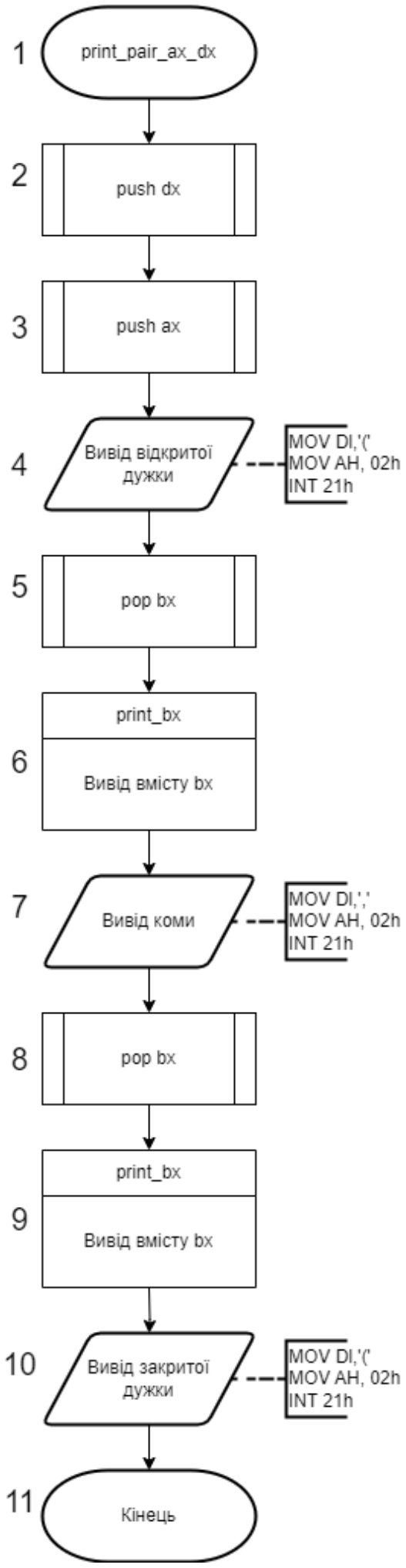

1
init_matrix
2
Вивід запиту на введення кількості рядків
MOV DX,msg1
MOV AH, 09h
INT 21h
3
single_dimension_input
Введення та валідація розмірності
4
MOV row_count,AX
5
Вивід запиту на введення кількості стовпців
MOV DX,msg2
MOV AH, 09h
INT 21h
6
single_dimension_input
Введення та валідація розмірності
7
MOV column_count,AX
8
MUL row_count
9
MOV total_size,AX
10
Динамічне виділення пам'яті на масив
MOV AH 48h
MOV BX,total_size
INT 21H
11
MOV MY_ARRAY,AX
A
На стор. 31
A
З і стор. 31
12
Вивід стрічки про початок ініціалізації матриця
MOV DX,msg_ask_for_input
MOV AH, 09h
INT 21h
13
MOV CX, ARRAY_SIZE
14
MOV DI,my_array
15
Перехід на нову стрічку
MOV DX,offset newline
MOV AH, 9
INT 21h
16
PUSH CX
17
single_value_input
Введення та валідація байтового значення
18
MOV DI,AX
19
POP CX
20
INC DI
21
CX=CX-1
22
CX=0?
Ні
Так
23
Кінець

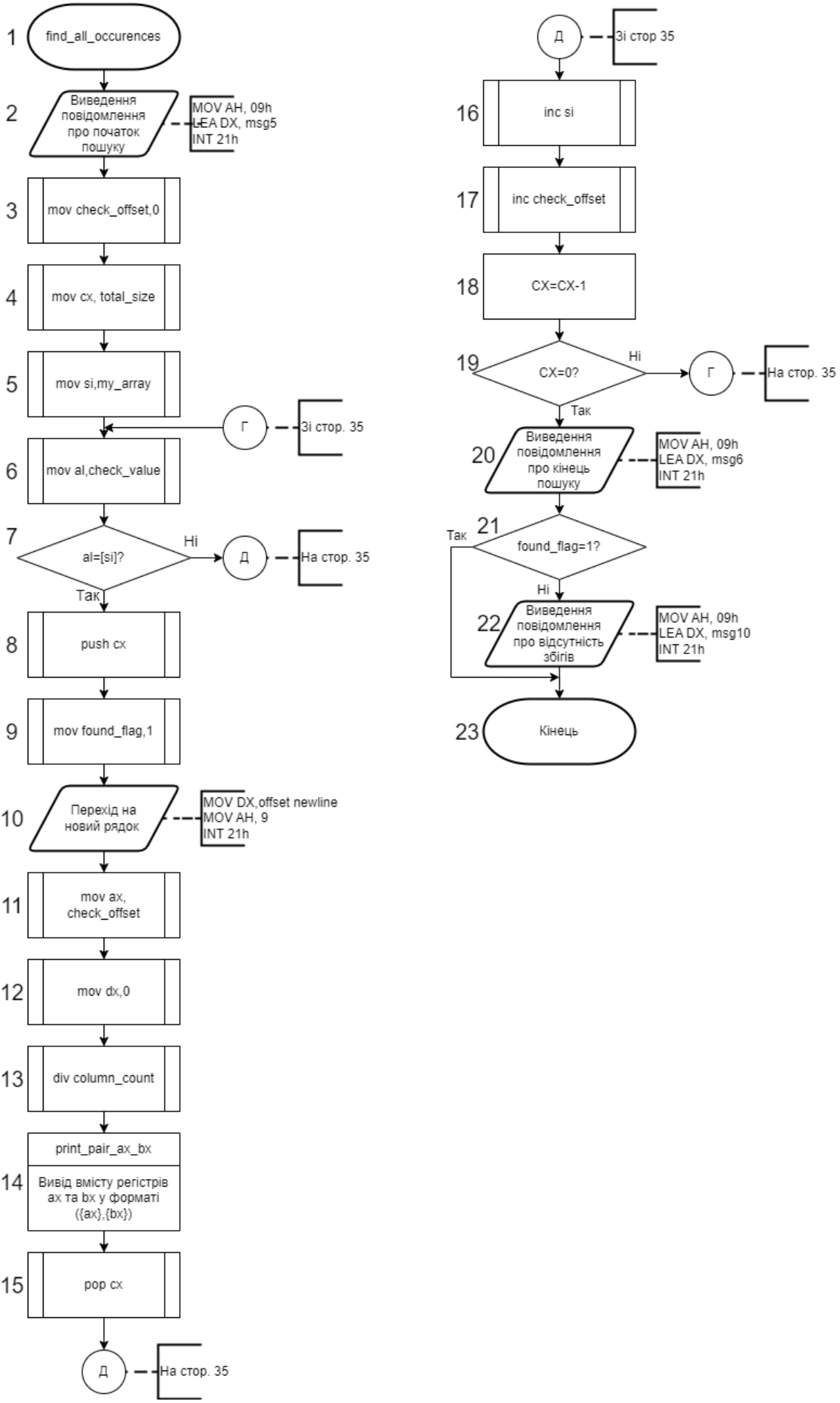

1
print_bx_from_new_line
2
Перехід на новий рядок
MOV DX,offset newline
MOV AH, 9
INT 21h
3
print_bx
Вивід вмісту bx
4
Кінець

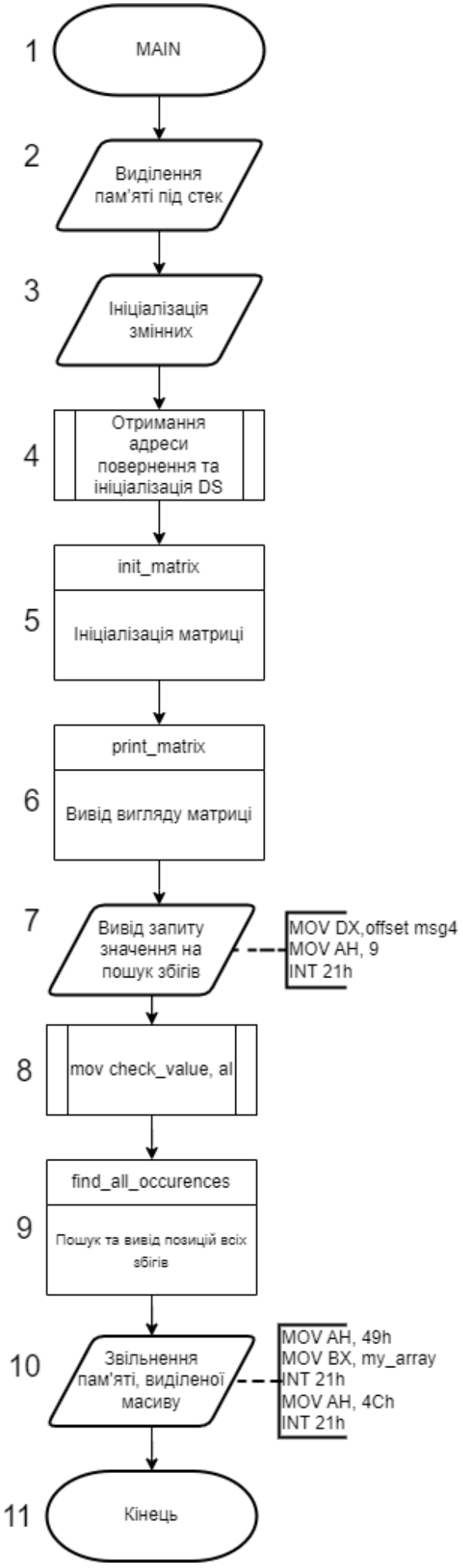

1
print_bl
2
Розширюємо вміст bl до слова
mov al,bl
cbw
mov bx,ax
3
print_bx
Вивід вмісту bx
4
Кінець


1
print_bl_from_new_line
2
Перехід на новий рядок
MOV DX,offset newline
MOV AH, 9
INT 21h
3
print_bl
Вивід вмісту bl
4
Кінець


1
print_matrix
2
Вивід повідомлення про стан вигляд матриці
MOV DX,msg8
MOV AH, 9
INT 21h
3
MOV DI, my_array
4
MOV CX,ROW_COUNT
Б
На стор. 33
5
Перехід на новий рядок
MOV DX,offset newline
MOV AH, 9
INT 21h
6
MOV ax, column_count
7
PUSH CX
В
На стор. 33
8
PUSH AX
9
MOV bl, byte ptr [di]
10
print_bl
Вивід вмісту bl
Г
На стор. 33
Г
3і стор. 33
11
INC DI
12
Вивід пробілу
MOV DI,' '
MOV AH, 02h
INT 21h
13
POP ax
14
DEC ax
15
ZF=1?
Ні
В
3і стор. 33
Так
16
POP cx
17
CX=CX-1
18
CX=0?
Ні
Б
На стор. 33
Так
19
Кінець

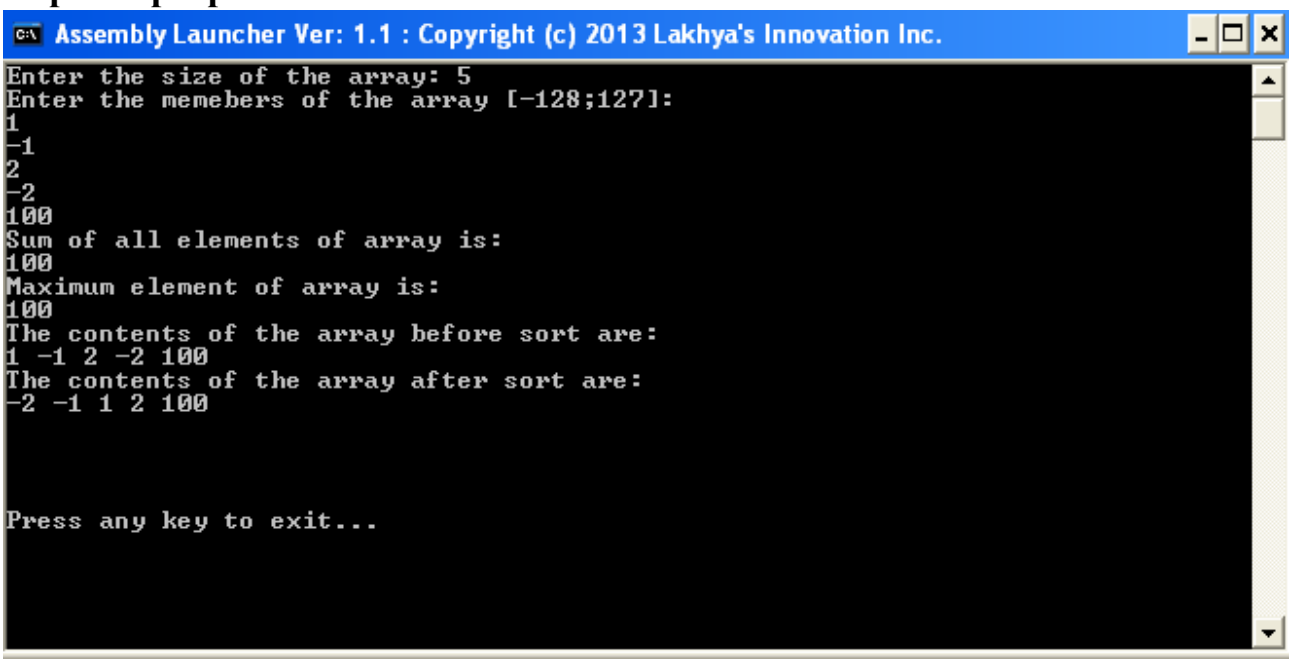

1
print_pair_ax_dx
2
push dx
3
push ax
4
Вивід відкритої дужки
MOV DI,'('
MOV AH, 02h
INT 21h
5
pop bx
6
print_bx
Вивід вмісту bx
7
Вивід коми
MOV DI,','
MOV AH, 02h
INT 21h
8
pop bx
9
print_bx
Вивід вмісту bx
10
Вивід закритої дужки
MOV DI,'('
MOV AH, 02h
INT 21h
11
Кінець


1
find_all_occurences
2
Виведення повідомлення про початок пошуку
MOV AH, 09h
LEA DX, msg5
INT 21h
3
mov check_offset,0
4
mov cx, total_size
5
mov si,my_array
Г
3і стор. 35
6
mov al,check_value
7
al=[si]?
Ні
Д
На стор. 35
Так
8
push cx
9
mov found_flag,1
10
Перехід на новий рядок
MOV DX,offset newline
MOV AH, 9
INT 21h
11
mov ax, check_offset
12
mov dx,0
13
div column_count
print_pair_ax_bx
14
Вивід вмісту регістрів ax та bx у форматі ({ax},{bx})
15
pop cx
Д
На стор. 35
Д
3і стор 35
16
inc si
17
inc check_offset
18
CX=CX-1
19
CX=0?
Ні
Г
На стор. 35
Так
20
Виведення повідомлення про кінець пошуку
MOV AH, 09h
LEA DX, msg6
INT 21h
Так
21
found_flag=1?
Ні
22
Виведення повідомлення про відсутність збігів
MOV AH, 09h
LEA DX, msg10
INT 21h
23
Кінець

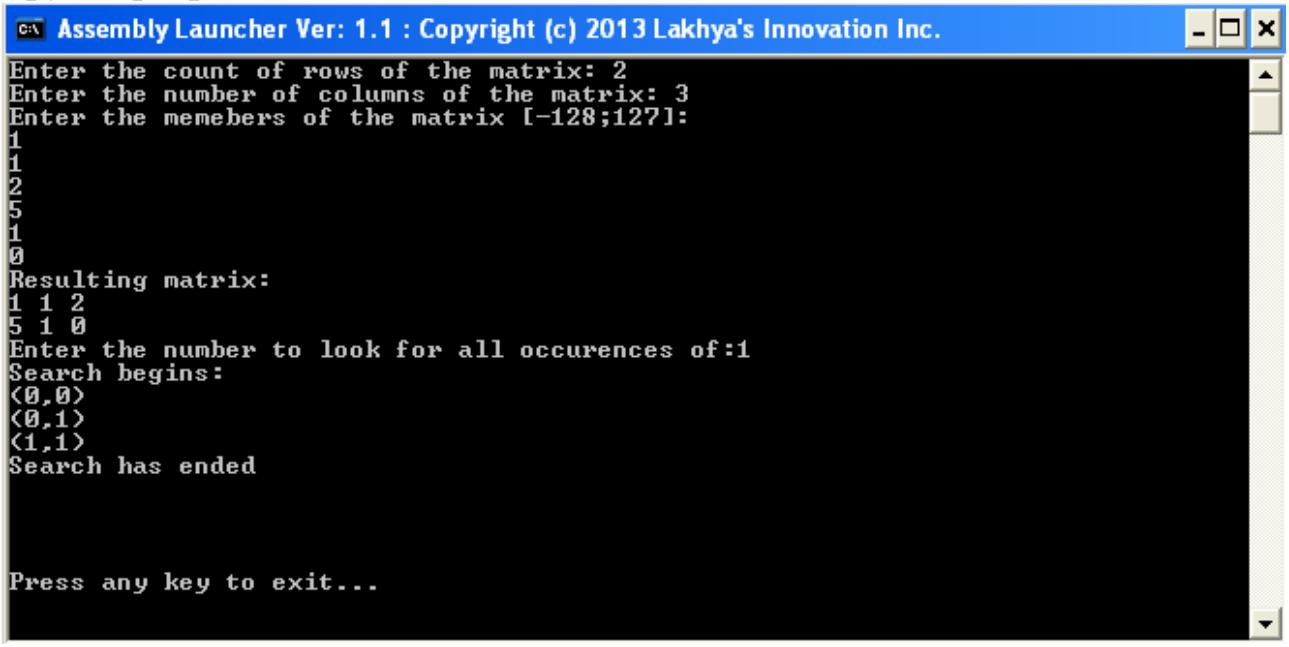

1
MAIN
2
Виділення пам'яті під стек
3
Ініціалізація змінних
4
Отримання адреси повернення та ініціалізація DS
init_matrix
5
Ініціалізація матриці
print_matrix
6
Вивід вигляду матриці
7
Вивід запиту значення на пошук збігів
MOV DX,offset msg4
MOV AH, 9
INT 21h
8
mov check_value, al
find_all_occurences
9
Пошук та вивід позицій всіх збігів
10
Звільнення пам'яті, виділеної масиву
MOV AH, 49h
MOV BX, my_array
INT 21h
MOV AH, 4Ch
INT 21h
11
Кінець

Приклад роботи програм:

Перша програма:

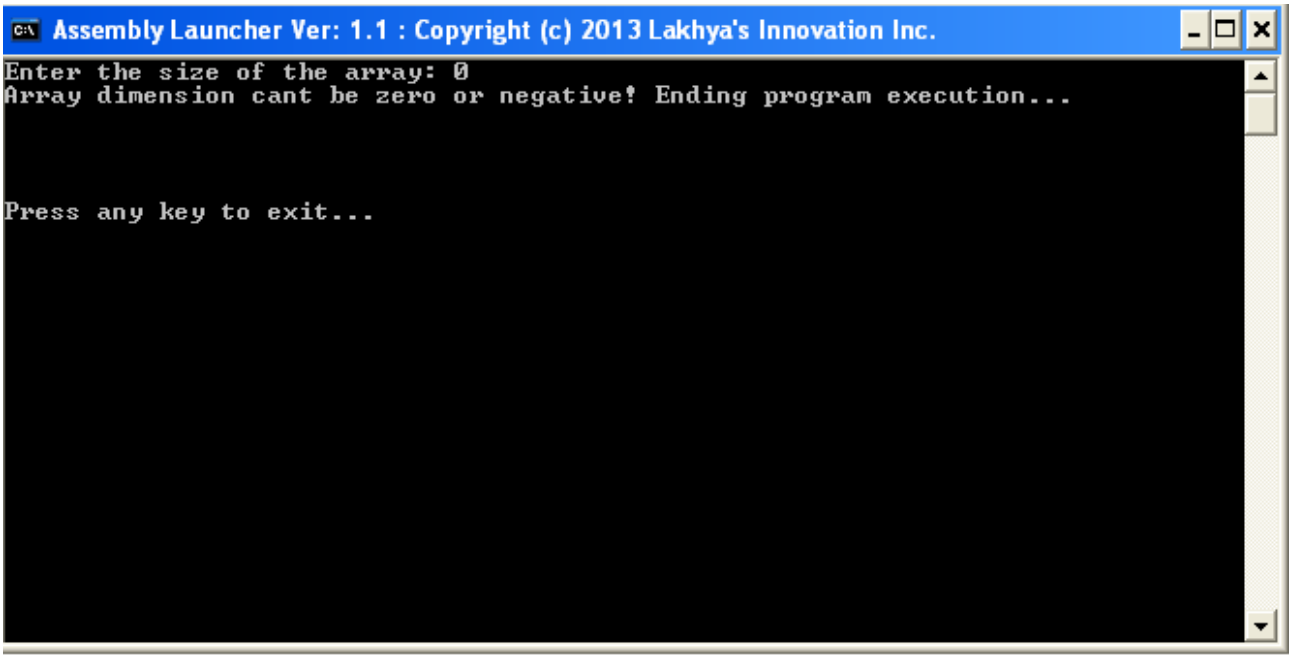


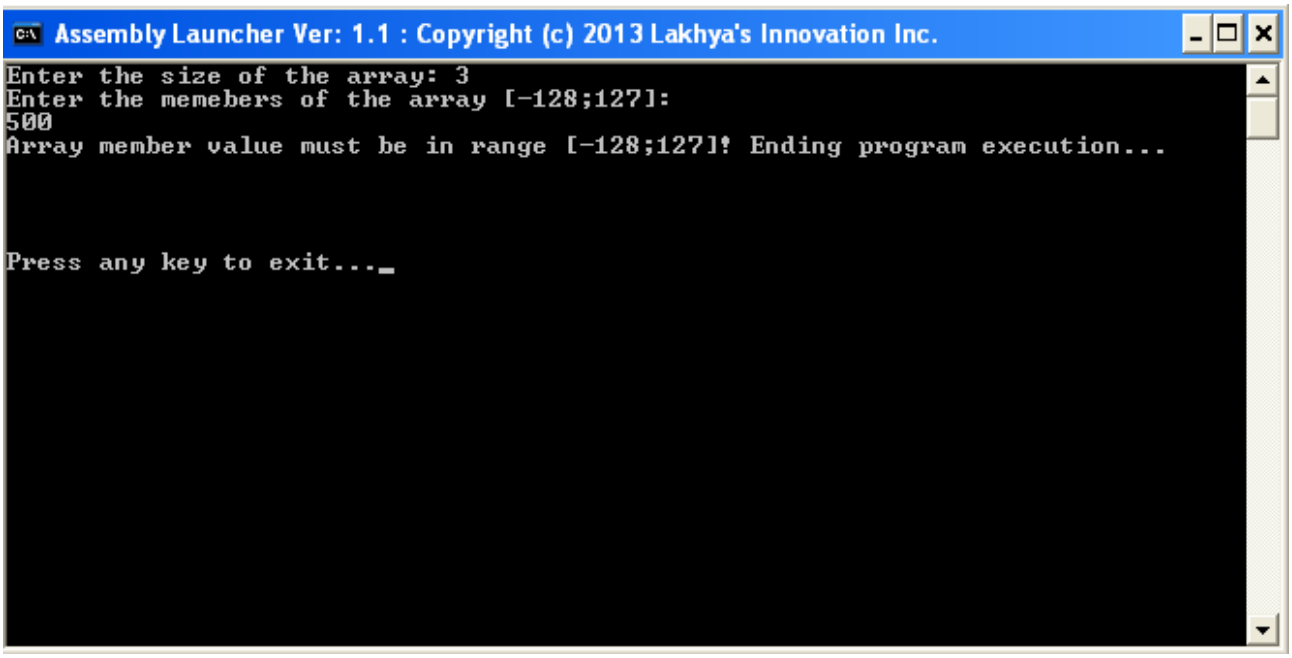
Друга програма:



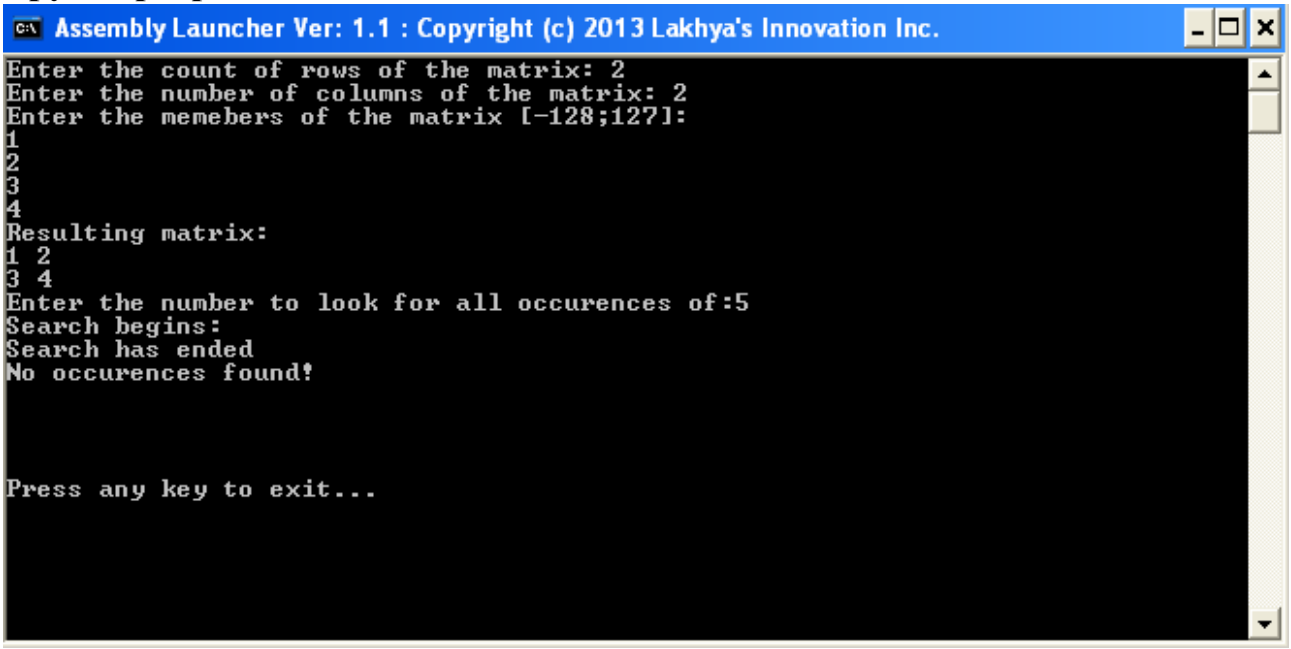
Перевірочні дані:

Перша програма:





Друга програма:



Висновок:

Отож, у ході виконання лабораторної роботи було проаналізовано поставлене завдання, ознайомлено з теоретичної базою завдання та створено програмне забезпеченння для вирішення двох задач: ініціалізація масиву, пошук найбільшого елементу, суми елементів, а також сортування; ініціалізація матриці та пошук усіх появ зазначеного користувачем значення у даній структурі. У рамках виконання даних підзавдань створено низку функцій різного вектору спрямування за своїм призначенням: функції виводу байтових та словесних значень, вводу розмірностей структур та числових значень в даних структурах, форматованого виводу значень регістрів, тощо. Систему було побудовано з урахуванням найвірогідніших відмов: неправильного вводу, переповнення, ін.. Під час виконання лабораторної роботи було набуто практичних навичок роботи з перериваннями для динамічного виділення та вивільнення пам’яті. Урешті-решт, було проведено тестування створеного ПЗ та побудовано на основі коду програм блок-схеми.