

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра технічної кібернетики

Звіти до комп'ютерних практикумів з кредитного модуля

«Програмування, ч. III»

«Системне програмування»

Прийняв

доцент кафедри ТК

Лісовиченко О.І.

“29” січня 2021 р.

Виконав:

студент групи ЗПІ-зп02

Кононов Максим Анатолійович

Київ 2021

Комп'ютерний практикум № 1. Тема: Створення програм на асемблері.

Завдання:

1. Для програми, наведеної вище, створити файл типу .asm. Ця програма не має засобів виводу даних, тому правильність її виконання треба перевірити за допомогою td.exe.
2. Скомпілювати програму, включивши потрібні опції для налагоджувача та створення файлу лістингу типу .lst.
3. Ознайомитись зі структурою файлу .lst. За вказівкою викладача, для певної команди асемблера розглянути структуру машинної команди і навести її у звіті.
4. Скомпонувати .obj-файл програми. Включити опції для налагодження та створення .map-файлу.
5. Занести до звіту адреси початку та кінця всіх сегментів з .map-файлу.
6. Завантажити до налагоджувача td.exe одержаний .exe-файл програми.
7. У вікні CPU у полі DUMP знайти початкову адресу сегмента даних та записати його до звіту. Знайти масиви SOURCE та DEST. Дані у масиві SOURCE подаються у шістнадцятковій системі.
8. У покроковому режимі за допомогою клавіші F7 виконати програму. Одержані результати у масиві DEST показати викладачеві.

Текст програми:

```
STSEG SEGMENT PARA STACK "STACK"
DB 64 DUP ( "STACK" )
STSEG ENDS
DSEG SEGMENT PARA PUBLIC "DATA"
SOURCE DB 10, 20, 30, 40
DEST DB 4 DUP ( "?" )
DSEG ENDS
CSEG SEGMENT PARA PUBLIC "CODE"
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG, SS: STSEG
; адреса повернення
PUSH DS
MOV AX, 0 ; або XOR AX, AX
PUSH AX
; ініціалізація DS
MOV AX, DSEG
MOV DS, AX
; обнуляємо масив
MOV DEST, 0
MOV DEST+1, 0
MOV DEST+2, 0
MOV DEST+3, 0
; пересилання
MOV AL, SOURCE
MOV DEST+3, AL
MOV AL, SOURCE+1
MOV DEST+2, AL
MOV AL, SOURCE+2
MOV DEST+1, AL
```

```

MOV AL, SOURCE+3
MOV DEST, AL
RET
MAIN ENDP
CSEG ENDS
END MAIN

```

Для подальшого тестування програма компілюється та завантажується у налагоджувач за допомогою короткого скрипту, написаного у bat-файлі:

```

tasm /la code.asm
tlink code.obj
td code.exe

```

Введені та отримані результати:

Вміст lst-файлу:

Turbo Assembler Version 4.0 28/01/21 20:43:24 Page 1

code.asm

```

1 0000 STSEG SEGMENT PARA STACK "STACK"
2 0000 40*(53 54 41 43 4B) DB 64 DUP ( "STACK" )
3 0140 STSEG ENDS
4 0000 DSEG SEGMENT PARA PUBLIC "DATA"
5 0000 0A 14 1E 28 SOURCE DB 10, 20, 30, 40
6 0004 04*(3F) DEST DB 4 DUP ( "?" )
7 0008 DSEG ENDS
8 0000 CSEG SEGMENT PARA PUBLIC "CODE"
9 0000 MAIN PROC FAR
10 ASSUME CS: CSEG, DS: DSEG, SS: STSEG
11 ; адреса повернення
12 0000 1E PUSH DS
13 0001 B8 0000 MOV AX, 0 ; або XOR AX, AX
14 0004 50 PUSH AX
15 ; ініціалізація DS
16 0005 B8 0000s MOV AX, DSEG
17 0008 8E D8 MOV DS, AX
18 ; обнуляємо масив
19 000A C6 06 0004r 00 MOV DEST, 0
20 000F C6 06 0005r 00 MOV DEST+1, 0
21 0014 C6 06 0006r 00 MOV DEST+2, 0
22 0019 C6 06 0007r 00 MOV DEST+3, 0
23 ; пересилання
24 001E A0 0000r MOV AL, SOURCE
25 0021 A2 0007r MOV DEST+3, AL
26 0024 A0 0001r MOV AL, SOURCE+1
27 0027 A2 0006r MOV DEST+2, AL
28 002A A0 0002r MOV AL, SOURCE+2
29 002D A2 0005r MOV DEST+1, AL
30 0030 A0 0003r MOV AL, SOURCE+3
31 0033 A2 0004r MOV DEST, AL
32 0036 CB RET
33 0037 MAIN ENDP
34 0037 CSEG ENDS
35 END MAIN

```

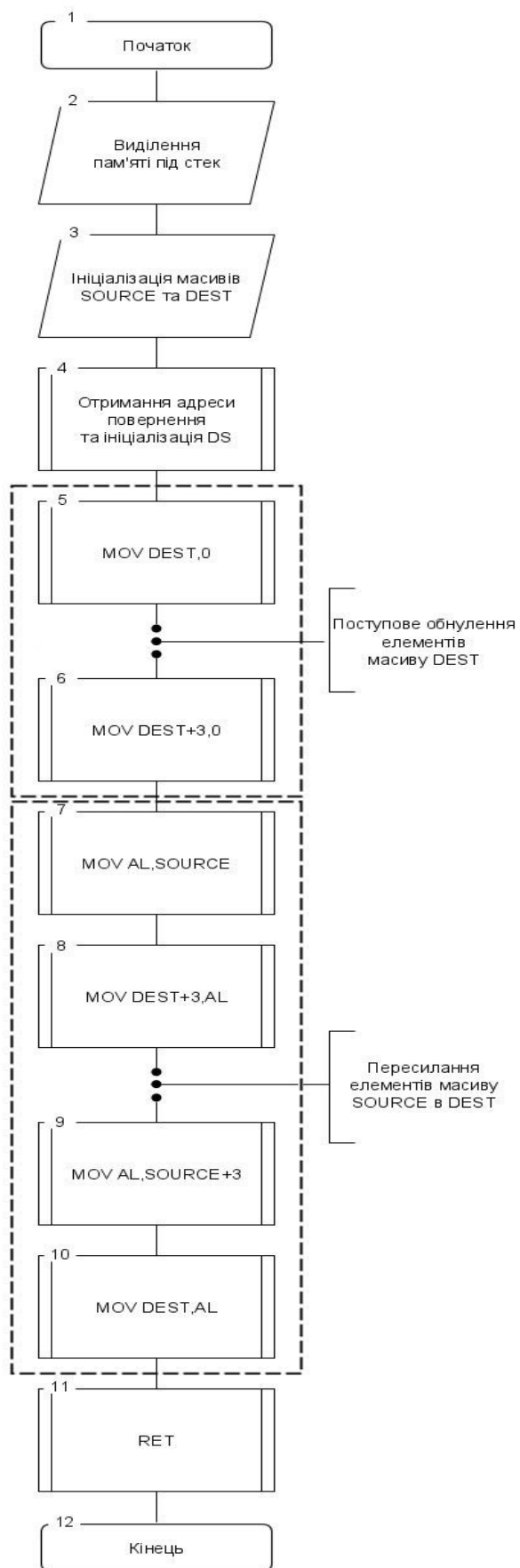
Symbol Table

Symbol Name	Type	Value
??DATE	Text	"28/01/21"
??FILENAME	Text	"code "
??TIME	Text	"20:43:24"
??VERSION	Number	0400
@CPU	Text	0101H
@CURSEG	Text	CSEG
@FILENAME	Text	CODE
@WORDSIZE	Text	2
DEST	Byte	DSEG:0004
MAIN	Far	CSEG:0000
SOURCE	Byte	DSEG:0000
Groups & Segments	Bit Size Align	Combine Class
CSEG	16 0037 Para	Public CODE
DSEG	16 0008 Para	Public DATA
STSEG	16 0140 Para	Stack STACK

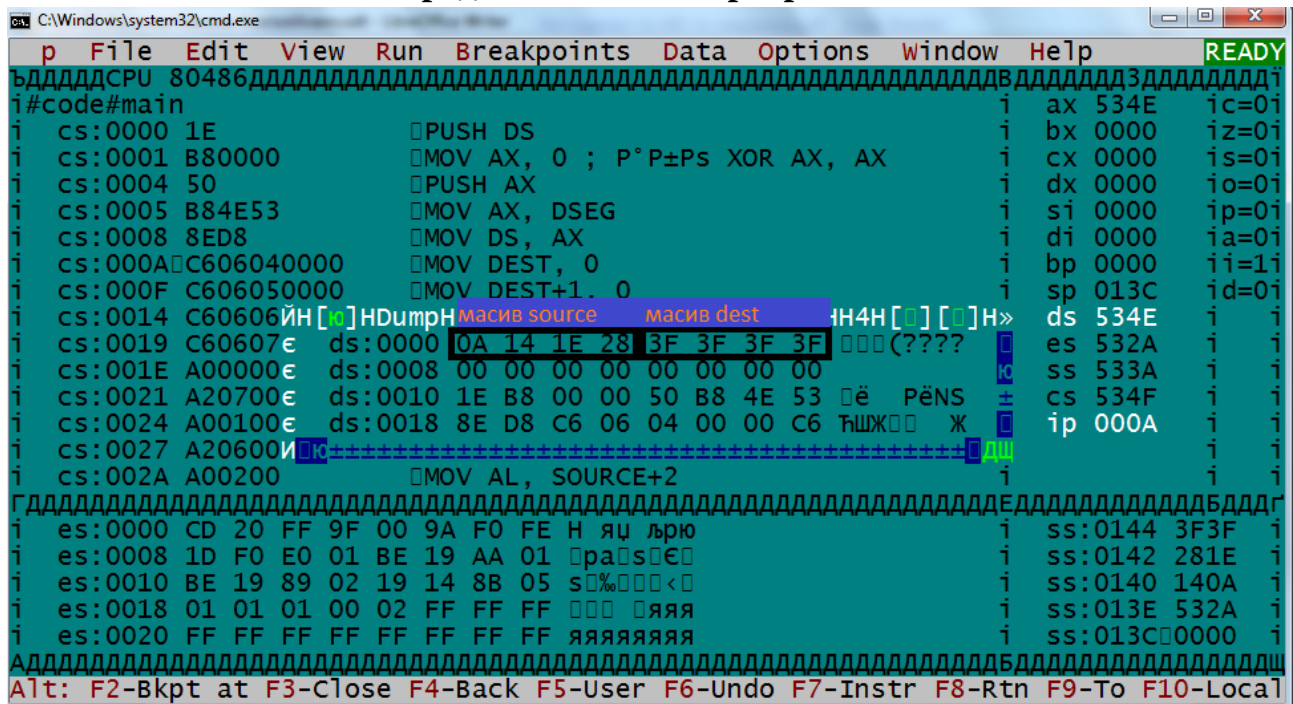
Вміст мар-файлу:

Start	Stop	Length	Name	Class
00000H	0013FH	00140H	STSEG	STACK
00140H	00147H	00008H	DSEG	DATA
00150H	00186H	00037H	CSEG	CODE
Program entry point at 0015:0000				

Схема функціонування програми:

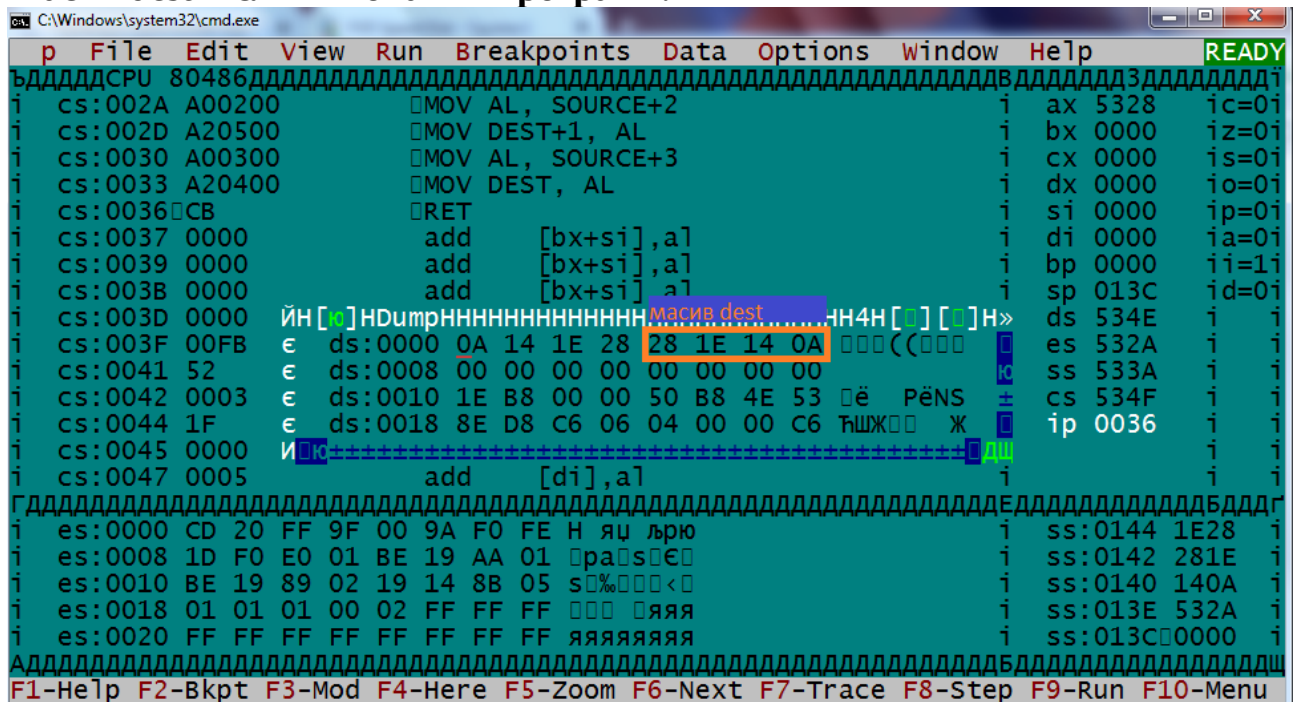


Масиви source та dest перед виконанням програми:



```
p File Edit View Run Breakpoints Data Options Window Help
CPU 80486
i #code#main
i cs:0000 1E          PUSH DS
i cs:0001 B80000      MOV AX, 0 ; P°P±Ps XOR AX, AX
i cs:0004 50          PUSH AX
i cs:0005 B84E53      MOV AX, DSEG
i cs:0008 8ED8        MOV DS, AX
i cs:000A C606040000    MOV DEST, 0
i cs:000F C606050000    MOV DEST+1, 0
i cs:0014 C606060000    MOV AL, SOURCE+2
i cs:0019 C606070000    MOV DEST+1, AL
i cs:001E A00000      MOV AL, SOURCE+3
i cs:0021 A20700      MOV AL, SOURCE+3
i cs:0024 A00100      MOV AL, SOURCE+3
i cs:0027 A20600      MOV AL, SOURCE+3
i cs:002A A00200      MOV AL, SOURCE+2
i es:0000 CD 20 FF 9F 00 9A F0 FE H яц лрю
i es:0008 1D F0 E0 01 BE 19 AA 01 паs€
i es:0010 BE 19 89 02 19 14 8B 05 s%□□□<□
i es:0018 01 01 01 00 02 FF FF FF □□□ яяя
i es:0020 FF FF FF FF FF FF FF FF яяяяяяя
Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-Local
```

Масив dest після виконання програми:



```
p File Edit View Run Breakpoints Data Options Window Help
CPU 80486
i cs:002A A00200      MOV AL, SOURCE+2
i cs:002D A20500      MOV DEST+1, AL
i cs:0030 A00300      MOV AL, SOURCE+3
i cs:0033 A20400      MOV AL, SOURCE+3
i cs:0036 CB          RET
i cs:0037 0000      add [bx+si], al
i cs:0039 0000      add [bx+si], al
i cs:003B 0000      add [bx+si], al
i cs:003D 0000      add [bx+si], al
i cs:003F 00FB      add [di], al
i cs:0041 52          add [di], al
i cs:0042 0003      add [di], al
i cs:0044 1F          add [di], al
i cs:0045 0000      add [di], al
i cs:0047 0005      add [di], al
i es:0000 CD 20 FF 9F 00 9A F0 FE H яц лрю
i es:0008 1D F0 E0 01 BE 19 AA 01 паs€
i es:0010 BE 19 89 02 19 14 8B 05 s%□□□<□
i es:0018 01 01 01 00 02 FF FF FF □□□ яяя
i es:0020 FF FF FF FF FF FF FF FF яяяяяяя
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

Висновок:

У текстовому редакторі було створено файл типу .asm, за допомогою вище наведеного скрипту в bat-файлі його було скопійовано та відкрито в налагоджувачі Turbo Debugger. Крім цього було створено lst-файл, який містить коди команд та адреси змінних програми у 16-й системі. Програма була виконана покроково шляхом натискання клавіші F8, було зафіксовано результати її роботи на двох ключових етапах: стан цільового масиву dest до та після зміни його вмісту.

Комп'ютерний практикум № 2. Тема: засоби обміну даними.

Завдання:

1. Написати процедуру введення і перетворення цілого числа.
2. Виконати математичну дію над числом (номер завдання вибирати за останніми двома числами номеру в заліковій книжці).
3. Перевести число в рядок та вивести його на екран. Варіант №8: -23.

Текст програми:

```
dseg segment para public 'data'
MyNumber dw 23
NumberFromKeyboard dw 0
ten dw 1
sign db 1,'+'
message1 db 'Enter an integer or press ENTER to exit: $'
result db 10,'Result: $'
warning db 10,'Input correct integer or press ENTER to exit: $'
operation db ' - '
input db 5,5,10 dup('$');
dseg ends
CSEG SEGMENT PARA PUBLIC "CODE"
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG
PUSH DS
MOV AX,0
PUSH AX
MOV AX, DSEG
MOV DS, AX
mov ah,09h
mov dx, offset message1
int 21h
KeyboardInput:
mov ah,0ah
mov dx, offset input
int 21h
mov cl,input[1]
mov di,2
cmp input[di], '-'
jne CheckNumber
mov sign[0], '-'
mov input[di], '0'
inc di
dec cl
CheckNumber:
cmp input[di], '0'
jb ShowWarning
cmp input[di], '9'
ja ShowWarning
inc di
dec cl
jnz CheckNumber
dec di
```

```

jmp ConvertStringToNumber
ShowWarning:
cmp input[2],13
je exit
mov ah,09h
mov dx, offset warning
int 21h
jmp KeyboardInput
exit:
ret
ConvertStringToNumber:
sub input[di],'0'
xor ax,ax
mov al,input[di]
mov bx,ten
mul bx
add ax,NumberFromKeyboard
mov NumberFromKeyboard,ax
mov ax,10
mov bx,ten
mul bx
mov ten,ax
dec di
cmp di,1
jne ConvertStringToNumber
cmp sign[0], '-'
jne ShowResult
xor ax,ax
sub ax,NumberFromKeyboard
mov NumberFromKeyboard,ax
ShowResult:
mov ax,NumberFromKeyboard
mov ten,ax
mov ah,09h
mov dx, offset result
int 21h
call ShowNumber
mov ah,09h
mov dx, offset operation
int 21h
mov dx,MyNumber
mov NumberFromKeyboard,dx
mov dx,ten
mov MyNumber,dx
call ShowNumber
mov operation[1], '='
mov ah,09h
mov dx, offset operation
int 21h
mov ax,NumberFromKeyboard
sub ax,ten

```

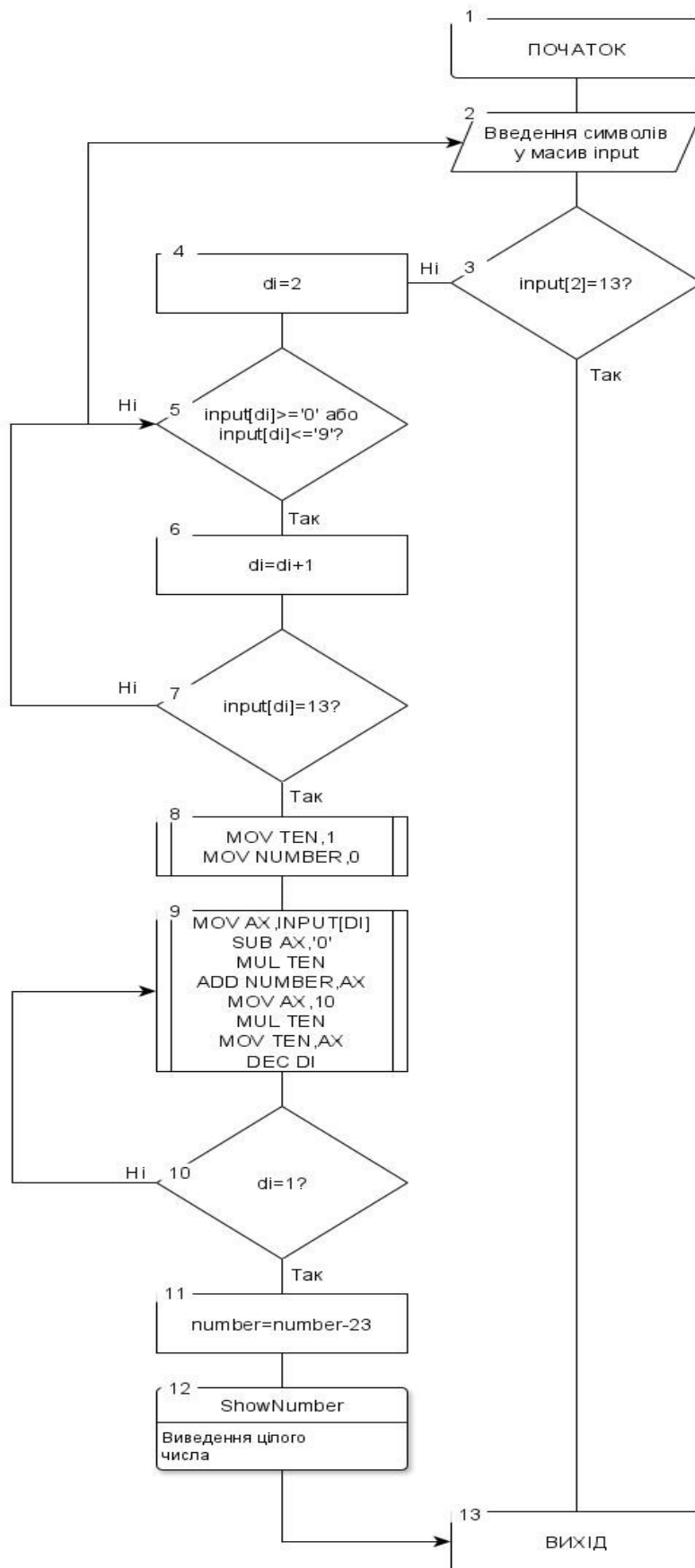


```

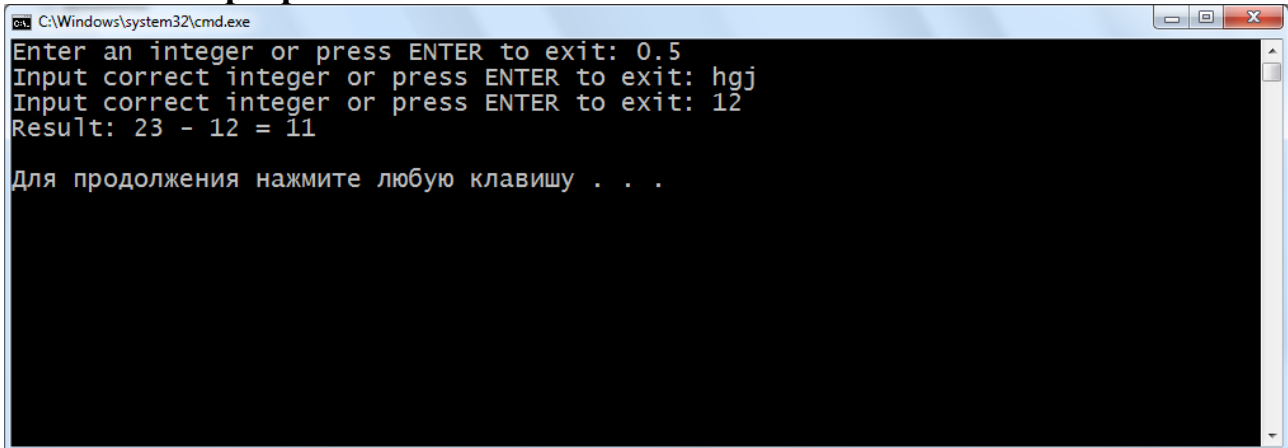
mov MyNumber,ax
call ShowNumber
RET
MAIN ENDP
ShowNumber proc
mov bx,MyNumber
or bx, bx
jns m1
mov al, '-'
int 29h
neg bx
m1:
mov ax, bx
xor cx, cx
mov bx, 10
m2:
xor dx, dx
div bx
add dl, '0'
push dx
inc cx
test ax, ax
jnz m2
m3:
pop ax
int 29h
loop m3
ret
ShowNumber endp
CSEG ENDS
END MAIN

```

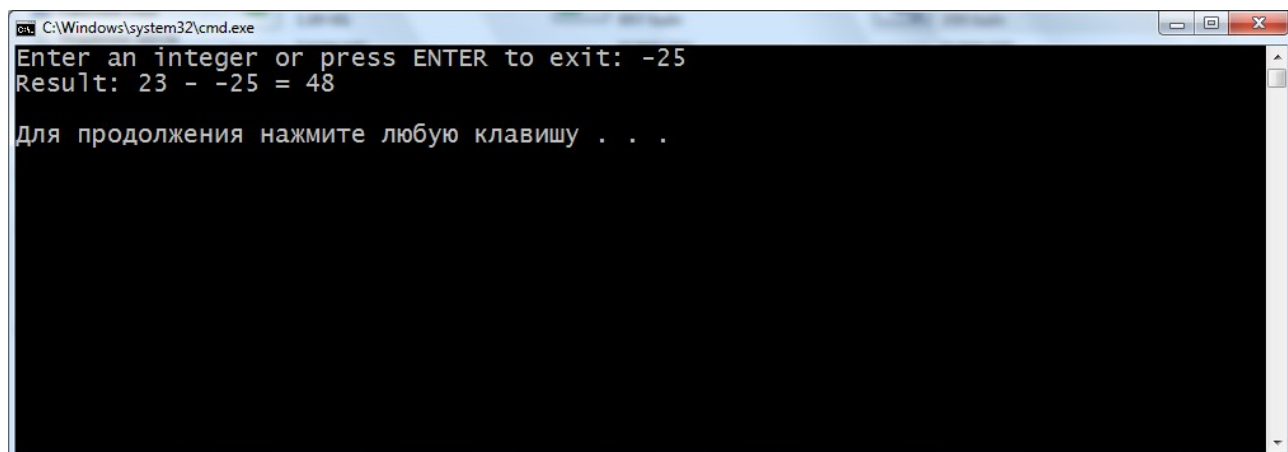
Схема функціонування програми:



Виконання програми:



```
C:\Windows\system32\cmd.exe
Enter an integer or press ENTER to exit: 0.5
Input correct integer or press ENTER to exit: hgj
Input correct integer or press ENTER to exit: 12
Result: 23 - 12 = 11
Для продовження натисніть будь-яку клавішу . . .
```



```
C:\Windows\system32\cmd.exe
Enter an integer or press ENTER to exit: -25
Result: 23 - -25 = 48
Для продовження натисніть будь-яку клавішу . . .
```

Висновок:

При виконанні даної роботи ми створили програму введення цілого числа з клавіатури та реалізували перевірку його коректності (число може бути додатним або від'ємним і не повинно містити нечислові символи окрім знаку - на початку клавіатурного вводу). Після цього була виконана арифметична дія: від числа 23 відняли введене та вивели результат на екран. Код виведення цілого числа був узятий із попередньої роботи, розміщений у вигляді процедури в кінці основного коду.

Комп'ютерний практикум № 3. Тема: програмування розгалужених алгоритмів.

Завдання:

Написати програму, яка буде обчислювати значення функції. Номер завдання вибирати за останніми двома числами номеру в заліковій книжці.

Функція:

$$8. \quad Z = \begin{cases} 8x^2 + 36 / x & \text{якщо } x > 0 \\ (1+x) / (1-x) & \text{якщо } -5 \leq x \leq 0 \\ 10x^2 & \text{якщо } x < -5 \end{cases}$$

Текст програми:

```
dseg segment para public 'data'
input db 5,5 dup('$');
NumberX dw 0
ten dw 1
sign db 1,'+'
result db 10,'x = $'
func1 db 10,'z = 8x^2 + 36 / x = $'
func1_1 db '8 * $'
func1_2 db ' ^ 2 + 36 / $'
func1_3 db ' = $'
func2 db 10,'z = (1 + x) / (1 - x) = $'
func2_1 db '(1 + $'
func2_2 db ') / (1 - $'
func2_3 db ') = $'
func3 db 10,'z = 10x^2 = $'
func3_1 db '10 * $'
func3_2 db ' ^ 2 = $'
message1 db 10,'Enter an integer or press ENTER to exit: $'
warning db 10,'Input correct integer or press ENTER to exit: $'
operation db ' - $'
overload1 db 10,'the x value must be smaller than 64$'
overload2 db 10,'the x value must be larger than -58$'
dseg ends
CSEG SEGMENT PARA PUBLIC "CODE"
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG
PUSH DS
MOV AX,0
PUSH AX
MOV AX, DSEG
MOV DS, AX
start:
call GetInteger
cmp NumberX,1
jge L1
cmp NumberX,-5
jge L2
jmp L3
L1:
cmp NumberX,64
jb equation1
mov ah,9h
mov dx, offset overload1
int 21h
jmp start
equation1:
mov ah,9h
mov dx, offset func1
int 21h
mov ah,9h
```

```

mov dx, offset func1_1
int 21h
call ShowInteger
mov ah,9h
mov dx, offset func1_2
int 21h
call ShowInteger
mov ah,9h
mov dx, offset func1_3
int 21h
mov ax,NumberX
mov ten,ax
mul ax
mov ten,8
mul ten
mov ten,ax
mov ax,36
div NumberX
add ax,ten
mov NumberX,ax
call ShowInteger
jmp start
L2:
mov ah,9h
mov dx, offset func2
int 21h
mov ah,9h
mov dx, offset func2_1
int 21h
call ShowInteger
mov ah,9h
mov dx, offset func2_2
int 21h
call ShowInteger
mov ah,9h
mov dx, offset func2_3
int 21h
mov ax,1
sub ax,NumberX
mov ten,ax
inc NumberX
cmp NumberX,0
jne divide
mov ah,2
mov dl,'0'
int 21h
jmp start
divide:
xor dx,dx
div NumberX
mov NumberX,ax

```

```

call ShowInteger
jmp start
L3:
cmp NumberX,-58
jg equation3
mov ah,9h
mov dx, offset overload2
int 21h
jmp start
int 21h
equation3:
mov ah,9h
mov dx, offset func3
int 21h
mov ah,9h
mov dx, offset func3_1
int 21h
call ShowInteger
mov ah,9h
mov dx, offset func3_2
int 21h
mov ax,NumberX
mul NumberX
mov ten,10
mul ten
mov NumberX,ax
call ShowInteger
jmp start
RET
MAIN ENDP
GetInteger proc
mov ten,1
mov NumberX,0
mov sign,'+'
mov ah,9h
mov dx, offset message1
int 21h
KeyboardInput:
mov ah,0ah
mov dx, offset input
int 21h
mov cl,input[1]
mov di,2
cmp input[di], '-'
jne CheckNumber
mov sign[0], '-'
mov input[di], '0'
inc di
dec cl
CheckNumber:
cmp input[di], '0'

```

```

jb ShowWarning
cmp input[di],'9'
ja ShowWarning
inc di
dec cl
jnz CheckNumber
dec di
jmp ConvertStringToNumber
ShowWarning:
cmp input[2],13
je exit
mov ah,09h
mov dx, offset warning
int 21h
jmp KeyboardInput
exit:
mov ah,4ch
int 21h
ConvertStringToNumber:
sub input[di],'0'
xor ax,ax
mov al,input[di]
mov bx,ten
mul bx
add ax,NumberX
mov NumberX,ax
mov ax,10
mov bx,ten
mul bx
mov ten,ax
dec di
cmp di,1
jne ConvertStringToNumber
cmp sign[0], '-'
jne return
xor ax,ax
mov ax,0
sub ax,NumberX
mov NumberX,ax
return:
ret
GetInteger endp
ShowInteger proc
mov bx,NumberX
or bx, bx
jns m1
mov al, '-'
int 29h
neg bx
m1:
mov ax, bx

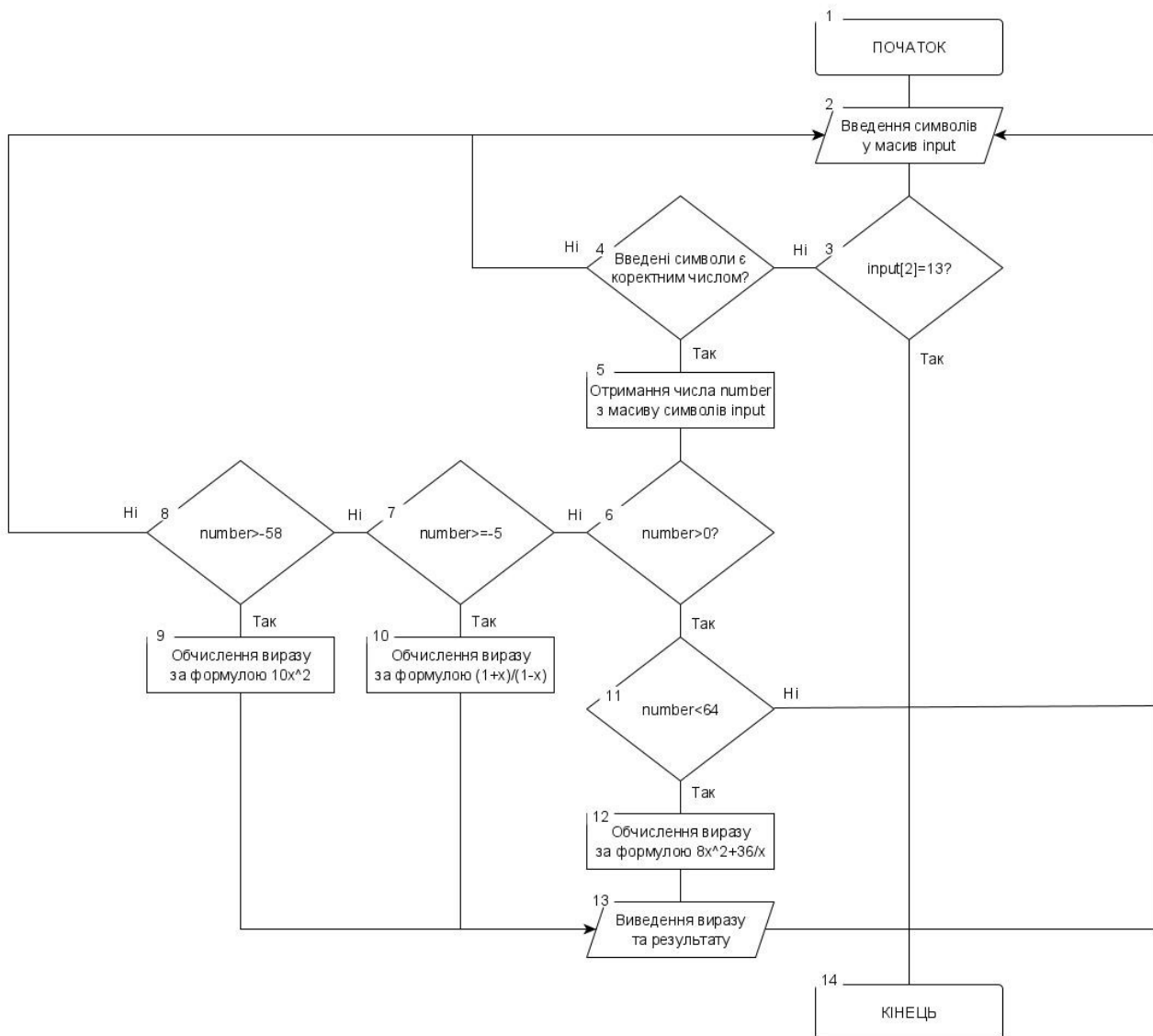
```

```

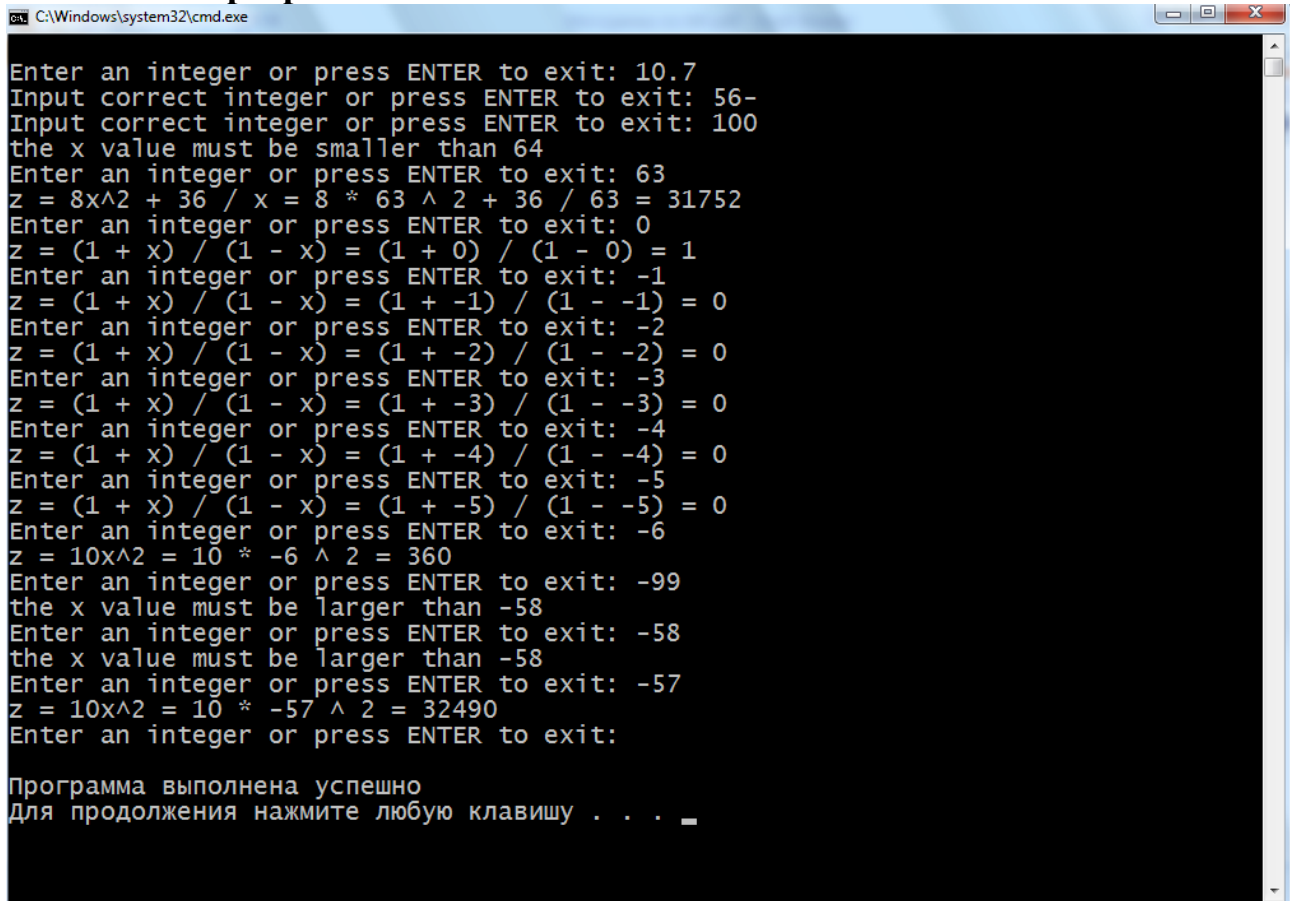
xor cx, cx
mov bx, 10
m2:
xor dx, dx
div bx
add dl, '0'
push dx
inc cx
test ax, ax
jnz m2
m3:
pop ax
int 29h
loop m3
ret
ShowInteger endp
CSEG ENDS
END MAIN

```

Схема функціонування програми:



Виконання програми:



```
C:\Windows\system32\cmd.exe
Enter an integer or press ENTER to exit: 10.7
Input correct integer or press ENTER to exit: 56-
Input correct integer or press ENTER to exit: 100
the x value must be smaller than 64
Enter an integer or press ENTER to exit: 63
z = 8x^2 + 36 / x = 8 * 63 ^ 2 + 36 / 63 = 31752
Enter an integer or press ENTER to exit: 0
z = (1 + x) / (1 - x) = (1 + 0) / (1 - 0) = 1
Enter an integer or press ENTER to exit: -1
z = (1 + x) / (1 - x) = (1 + -1) / (1 - -1) = 0
Enter an integer or press ENTER to exit: -2
z = (1 + x) / (1 - x) = (1 + -2) / (1 - -2) = 0
Enter an integer or press ENTER to exit: -3
z = (1 + x) / (1 - x) = (1 + -3) / (1 - -3) = 0
Enter an integer or press ENTER to exit: -4
z = (1 + x) / (1 - x) = (1 + -4) / (1 - -4) = 0
Enter an integer or press ENTER to exit: -5
z = (1 + x) / (1 - x) = (1 + -5) / (1 - -5) = 0
Enter an integer or press ENTER to exit: -6
z = 10x^2 = 10 * -6 ^ 2 = 360
Enter an integer or press ENTER to exit: -99
the x value must be larger than -58
Enter an integer or press ENTER to exit: -58
the x value must be larger than -58
Enter an integer or press ENTER to exit: -57
z = 10x^2 = 10 * -57 ^ 2 = 32490
Enter an integer or press ENTER to exit:

Програма виконана успішно
Для продовження натисніть будь-яку клавішу . . . _
```

Висновок:

При виконанні даного завдання ми реалізували ввід чисел з клавіатури та вивід арифметичних виразів, використовуючи код з попередніх робіт. Ми навчились широко використовувати умовні розгалуження та вирішувати проблему умовного переходу на відстань більшу ніж 128 байт. Це стане нам в нагоді при розробці ПЗ, де неможливо обійтись без циклів та умовних виразів. Крім цього ми зрозуміли як працюють умовні оператори та циклічні конструкції в мовах програмування високого рівня.

Комп'ютерний практикум № 4. Тема: масиви.

Завдання:

1. Написати програму знаходження суми елементів одномірного масиву, елементи вводить користувач.
2. Написати програму пошуку максимального (або мінімального) елемента одномірного масиву, елементи вводить користувач.
3. Написати програму сортування одномірного масиву цілих чисел загального вигляду.
4. Написати програму пошуку координат всіх входжень заданого елемента в двомірному масиві, елементи масиву та пошуковий вводить користувач.

Текст програми:

```
dseg segment para public 'data'
NumbersCount dw 10
MaxInd dw 0
arr dw 10 dup(0);5177,6364,9806,8745,9698,440,5204,6343,4890,7742
;arr2 dw 10 DUP(10 DUP(0))
arr2 dw 100 dup(0)
number dw 0
minrand dw 1
maxrand dw 1000
seed dw 0
seed2 dw 0
SumOfNumbers dw 0
MaxNumber dw 0
TargetNumber dw 0
input db 5,5 dup('$')
ten dw 1
sign db 1,'+'
message db 10,10 dup('$')
RandomArray db 10,'Random array: $'
StrSumOfNumbers db 10,'Sum of numbers: $'
StrMaxNumber db '; Maximum number: $'
StrSortedArray db 10,'Sorted array: $'
StrTwoDimensionalArray db 10,'Random two-dimensional array: $'
StrIndexFirst db 10,'Indexes: $'
StrNumberNotFound db 10,'This number was not found.$'
message1 db 'Enter an integer which must be found in the array or press ENTER to exit: $'
warning db 10,'Input correct integer or press ENTER to exit: $'
;-----variables for LST radix sort-----
dest dw 10 dup (0)
baskets dw 2048 dup (0)
key dw 0
value dw 0
dseg ends
CSEG SEGMENT PARA PUBLIC "CODE"
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG
PUSH DS
MOV AX,0
PUSH AX
MOV AX,DSEG
MOV DS,AX
mov ax,2
mul NumbersCount
mov MaxInd,ax
mov cx,NumbersCount
mov si,0
GenerateRandomNumbers:
push cx
call GetRandomNumber
pop cx
mov ax,number
mov arr[si],ax
add si,2
loop GenerateRandomNumbers
mov ah,9
mov dx,offset RandomArray
int 21h
mov cx,NumbersCount
mov si,0
mov SumOfNumbers,0
```

```

FindSumAndMaximumNumber:
mov ax,arr[si]
mov number,ax
add SumOfNumbers,ax
cmp ax,MaxNumber
jb continue
mov MaxNumber,ax
continue:
push cx
mov sign,'+'
call ShowInteger
pop cx
mov ah,2
mov dl,' '
int 21h
add si,2
cmp si,MaxInd
jb FindSumAndMaximumNumber
mov ah,9
mov dx,offset StrSumOfNumbers
int 21h
mov ax,SumOfNumbers
mov number,ax
call ShowInteger
mov ah,9
mov dx,offset StrMaxNumber
int 21h
mov ax,MaxNumber
mov number,ax
call ShowInteger
mov ah,9
mov dx,offset StrSortedArray
int 21h
call LSTRadixSort
call ShowArray
,,,,,,,,,,,,,,,,,,,,,,,,,TheTwoDimensionalArray,,,,,,,,,,,,,,,,,,,,,,,,,
mov ah,9
mov dx,offset STRTwoDimensionalArray
int 21h
mov ah,2
mov dl,10
int 21h
mov di,0
mov cx,NumbersCount
rows:
mov si,0
push cx
columns:
push di
call GetRandomNumber
call ShowInteger
pop di
mov ax,number
mov arr2[di],ax
mov ah,2
mov dl,' '
int 21h
add di,2
inc si
cmp si,NumbersCount
jb columns

```

```

mov ah,2
mov dl,10
int 21h
pop cx
loop rows
SearchNumber:
call GetInteger
mov ax,number
mov TargetNumber,ax
mov di,0
mov bx,0
mov cx,NumbersCount
RowsSearchCycle:
mov si,0
push cx
ColumnsSearchCycle:
mov ax,arr2[bx][si]
cmp ax,TargetNumber
jne NextElement
push bx
cmp di,0
ja IndexNotFirst
mov ah,9
mov dx,offset StrIndexFirst
int 21h
jmp ShowIndex
IndexNotFirst:
mov ah,2
mov dl','
int 21h
xor dx,dx
mov ax,di
mov number,9
div number
cmp dx,0
ja ShowIndex
mov ah,2
mov dl,10
int 21h
pop bx
push bx
ShowIndex:
mov ah,2
mov dl,['
int 21h
xor dx,dx
mov ax,bx
div MaxInd
mov number,ax
call ShowInteger
mov ah,2
mov dl','
int 21h
xor dx,dx
mov ax,si
mov number,2
div number
mov number,ax
call ShowInteger
mov ah,2
mov dl,']'

```

```

int 21h
inc di
pop bx
NextElement:
add si,2
cmp si,MaxInd
jb ColumnsSearchCycle
add bx,MaxInd
pop cx
dec cx
cmp cx,0
je NextRow
jmp RowsSearchCycle;loop RowsSearchCycle
NextRow:
cmp di,0
je NumberNotFound
mov ah,2
mov dl,10
int 21h
jmp SearchNumber
NumberNotFound:
mov ah,9
mov dx,offset StrNumberNotFound
int 21h
mov ah,2
mov dl,10
int 21h
jmp SearchNumber
,,,,,,,,,,,,,,,,,,,,,TheTwoDimensionalArray END,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
RET
MAIN ENDP
ShowArray proc
mov si,0
ShowArrayCycle:
mov ax,arr[si]
mov number,ax
mov sign,'+'
call ShowInteger
mov ah,2
mov dl,' '
int 21h
add si,2
cmp si,MaxInd
jb ShowArrayCycle
ret
ShowArray endp
GetRandomNumber proc;http://datadump.ru/fasm-pseudo-random-number-generator/
mov di,maxrand
mov     dx,word[seed]
or dx,dx
jnz     l2
mov ah,2ch
int 21h
xor ax,ax
mov al,dl
mov     dx,ax
mov     ax,word[seed2]
or ax,ax
in ax,40h
l2:
mul     dx

```

```

inc     ax
mov     word[seed],dx
mov     word[seed2],ax
xor     dx,dx
sub     di,minrand
inc     di
div     di
mov     ax,dx
add     ax,minrand
mov     number,ax
ret
GetRandomNumber endp
LSTRadixSort proc
; mov ah, 2
; mov dl,10
; int 21h
; int 21h
mov     si,0
CreateBaskets:
mov     di,arr[si]
and     di,255
mov     ax,di
mov     bx,2
mul     bx
mov     di,ax
inc     baskets[di]
; mov number,di;start;
; call ShowInteger
; mov ah, 2
; mov dl, ' '
; int 21h;end;
mov     di,arr[si]
shr     di,8
and     di,255
add     di,256
mov     ax,di
mov     bx,2
mul     bx
mov     di,ax
inc     baskets[di]
; mov number,di;start;
; call ShowInteger
; mov ah, 2
; mov dl, ' '
; int 21h;end;
mov     di,arr[si]
shr     di,8
shr     di,8
and     di,255
add     di,512
mov     ax,di
mov     bx,2
mul     bx
mov     di,ax
inc     baskets[di]
; mov number,di;start;
; call ShowInteger
; mov ah, 2
; mov dl, ' '
; int 21h;end;
mov     di,arr[si]

```

```

shr di,8
shr di,8
shr di,8
and di,255
add di,768
mov ax,di
mov bx,2
mul bx
mov di,ax
inc baskets[di]
; mov number,di,,,,,,,,,,,,,start,,,,,,,,,,,,,
; call ShowInteger
; mov ah, 2
; mov dl,10
; int 21h,,,,,,,,,,,,,end,,,,,,,,,,,,,
add si,2
cmp si,MaxInd
je NextStep
jmp CreateBaskets
NextStep:
; mov ah, 2
; mov dl,10
; int 21h
; mov si,0
; ShowBaskets:
; mov sign,'+'
; mov ax,baskets[si]
; mov number,ax
; call ShowInteger
; mov ah,2
; mov dl,' '
; int 21h
; add si,2
; cmp si,2048
; jne ShowBaskets
; ;////////////////checked////////////////
; mov ah,2
; mov dl,10
; int 21h
mov cx,0
FillBaskets:
mov si,0
mov key,0
FillBasketsCycle:
mov ax,cx
mov bx,256
mul bx
add ax,si
mov bx,2
mul bx
mov di,ax
; mov number,di,,,,,,,,,,,,,start,,,,,,,,,,,,,
; push cx
; call ShowInteger
; mov ah,2
; mov dl,' '
; int 21h
; pop cx
; mov di,number,,,,,,,,,,,,,end,,,,,,,,,,,,,
mov ax,baskets[di]
mov value,ax

```

```

mov ax,key
mov baskets[di],ax
add ax,value
mov key,ax
inc si
cmp si,256
jb FillBasketsCycle
; mov ah,2,,,,,,,,,,,,,start,,,,,,,,,,,,,
; mov dl,10
; int 21h
; push cx
; mov ax,key
; mov number,ax
; call ShowInteger
; mov ah,2
; mov dl,10
; int 21h
; pop cx,,,,,,,,,,,,,end,,,,,,,,,,,,,
inc cx
cmp cx,4
jb FillBaskets
;////////////////////////checking////////////////////////
; mov ah, 2
; mov dl,10
; int 21h
; mov si,0
; ShowBaskets2:
; mov sign,'+'
; mov ax,baskets[si]
; mov number,ax
; call ShowInteger
; mov ah,2
; mov dl,' '
; int 21h
; add si,2
; cmp si,2048
; jne ShowBaskets2
; mov ah, 2
; mov dl,10
; int 21h
; int 21h
;////////////////////////checked////////////////////////
mov si,0
SortCycle1:
mov ax,arr[si]
and ax,255
mov bx,2
mul bx
mov ten,ax
; mov number,ax,,,,,,,,,,,,,start,,,,,,,,,,,,,
; call ShowInteger
; mov ah,2
; mov dl,' '
; int 21h
; mov ax,number,,,,,,,,,,,,,end,,,,,,,,,,,,,
mov di,ten
mov ax,baskets[di]
mov bx,2
mul bx
mov di,ax
; mov number,di,,,,,,,,,,,,,start,,,,,,,,,,,,,

```



```

; call ShowInteger
; mov ah,2
; mov dl,10
; int 21h
; mov di,number,,,,,,,,;end,,,,,,,,;
mov ax,arr[si]
mov dest[di],ax
mov di,ten
inc baskets[di]
add si,2
cmp si,MaxInd
jb SortCycle1
; mov ah,2
; mov dl,10
; int 21h
; mov si,0
; ShowDest:
; mov sign,'+'
; mov ax,dest[si]
; mov number,ax
; mov sign,'+'
; call ShowInteger
; mov ah,2
; mov dl,' '
; int 21h
; add si,2
; cmp si,MaxInd
; jb ShowDest
; mov ah, 2
; mov dl,10
; int 21h
; int 21h
;////////////////checked////////////////
; lea si,baskets[256]
; lea di,baskets
; push dword ptr[si]
; pop dword ptr[di]
mov si,0
;////////////////checked////////////////
mov si,0
SortCycle2:
mov ax,dest[si]
shr ax,8
and ax,255
mov bx,2
mul bx
add ax,512
mov ten,ax
; mov number,ax,,,,,,,,;start,,,,,,,,;
; call ShowInteger
; mov ah,2
; mov dl,' '
; int 21h
; mov ax,number,,,,,,,,;end,,,,,,,,;
mov di,ten
mov ax,baskets[di]
mov bx,2
mul bx
mov di,ax
; mov number,di,,,,,,,,;start,,,,,,,,;
; call ShowInteger

```

```

; mov ah,2
; mov dl,10
; int 21h
; mov di,number,,,,,,,,;end,,,,,,,,;
mov ax,dest[si]
mov arr[di],ax
mov di,ten
inc baskets[di]
add si,2
cmp si,MaxInd
jb SortCycle2
ret
LSTRadixSort endp
GetInteger proc
mov ten,1
mov sign,'+'
mov number,0
mov ah,9h
mov dx, offset message1
int 21h
KeyboardInput:
mov ah,0ah
mov dx, offset input
int 21h
mov cl,input[1]
mov di,2
cmp input[di], '-'
jne CheckNumber
mov sign[0], '-'
mov input[di], '0'
inc di
dec cl
CheckNumber:
cmp input[di], '0'
jb ShowWarning
cmp input[di], '9'
ja ShowWarning
inc di
dec cl
jnz CheckNumber
dec di
jmp ConvertStringToNumber
ShowWarning:
cmp input[2], 13
je exit
mov ah,09h
mov dx, offset warning
int 21h
jmp KeyboardInput
exit:
mov ah,4ch
int 21h
ConvertStringToNumber:
sub input[di], '0'
xor ax,ax
mov al,input[di]
mov bx,ten
mul bx
add ax,number
mov number,ax
mov ax,10

```

```

mov bx,ten
mul bx
mov ten,ax
dec di
cmp di,1
jne ConvertStringToNumber
cmp sign[0], '-'
jne return
xor ax,ax
mov ax,0
sub ax,number
mov number,ax
return:
ret
GetInteger endp
ShowInteger proc
mov bx,number
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
ShowInteger endp
CSEG ENDS
END MAIN

```

Загальна схема функціонування програми:

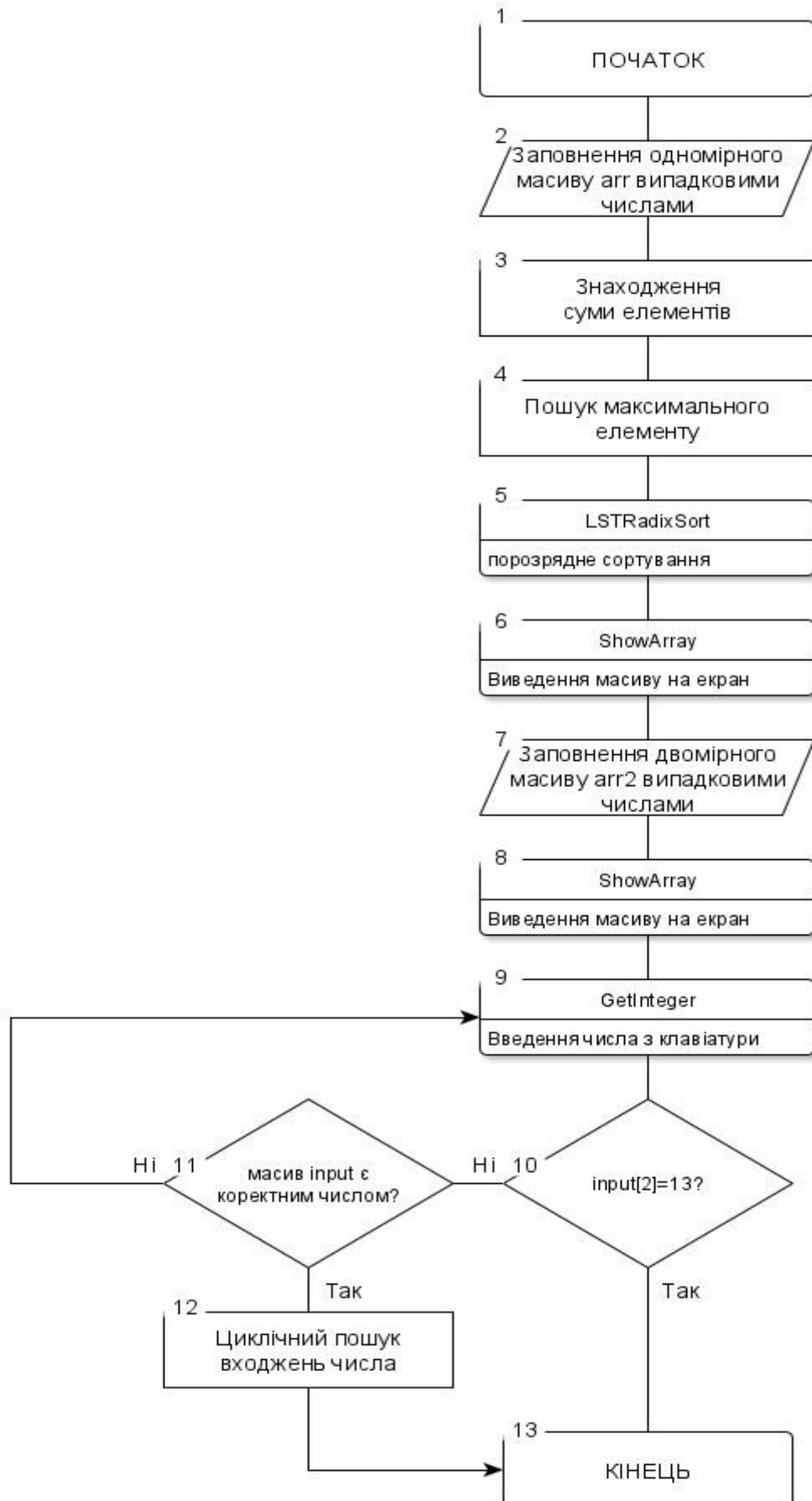
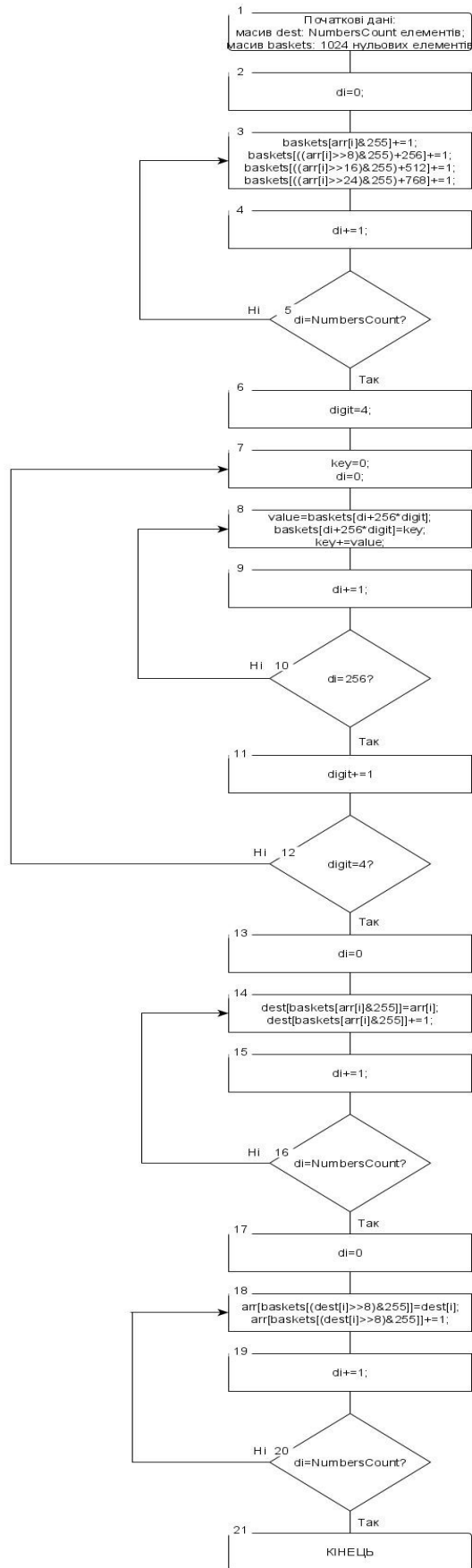


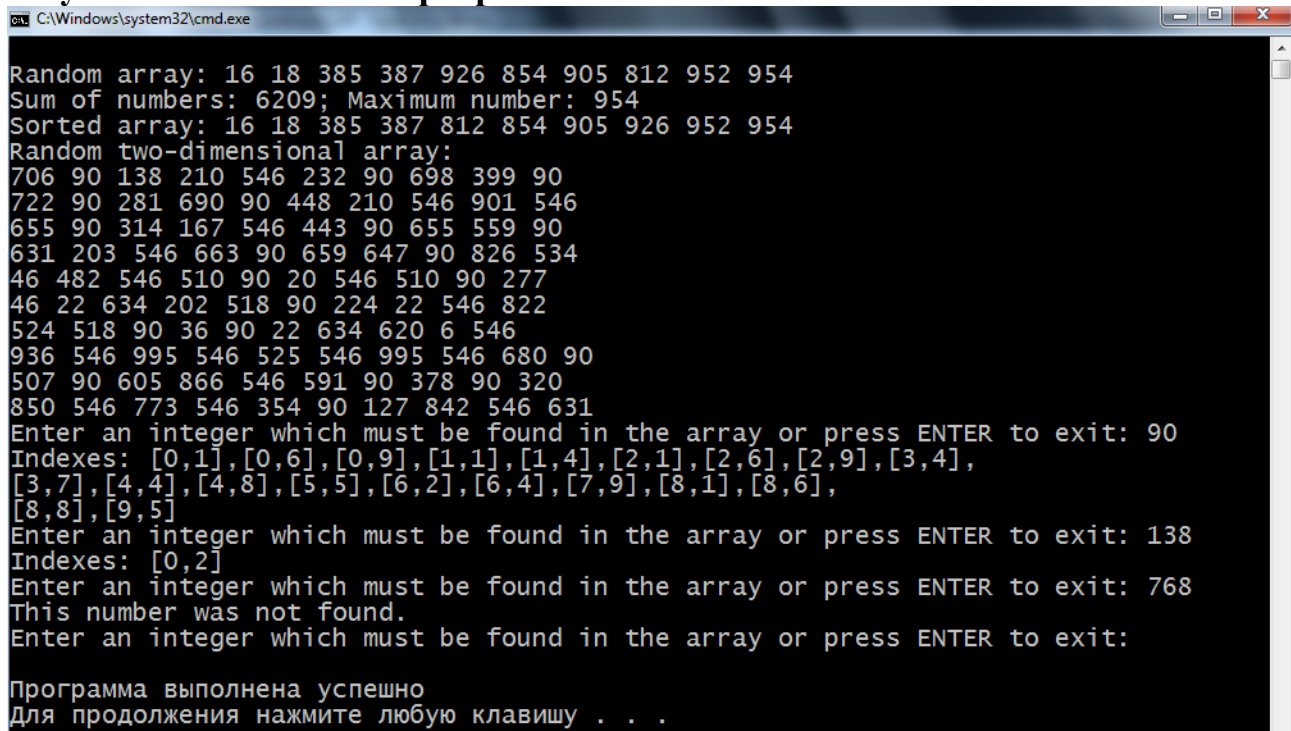
Схема оптимізованого алгоритму порозрядного сортування (LST radix sort)



Код алгоритму на мові програмування C, з якого він був портований:

```
void LSTRadixSort_optimized(int* arr,int size)
{
    int *dest=(int*)malloc(sizeof(int)*size);
    int i,key,digit,value,*baskets=(int*)calloc(sizeof(int),1024);
    for(i=0; i<size; ++i)
    {
        baskets[arr[i]&255]++;
        baskets[((arr[i]>>8)&255)+256]++;
        baskets[((arr[i]>>16)&255)+512]++;
        baskets[((arr[i]>>24)&255)+768]++;
    }
    for(digit=0; digit<4; digit++)
    {
        key=0;
        for(i=0; i<256; i++)
        {
            value=baskets[i+256*digit];
            baskets[i+256*digit]=key;
            key+=value;
        }
    }
    for(i=0; i<size; ++i)
        dest[baskets[arr[i]&255]++]=arr[i];
    // *(dest+(*(baskets+(*(arr+i)&255))))=arr[i];
    // *(baskets+0+(*(arr+i)&255))+=1;
    baskets=&baskets[256];
    for(i=0; i<size; ++i)
        arr[baskets[(dest[i]>>8)&255]++]=dest[i];
    // *(arr+(*(baskets+256+((*(dest+i)>>8)&255))))=dest[i];
    // *(baskets+256+((*(dest+i)>>8)&255))+=1;
    free(dest);
    free(baskets-256);
}
```

Результати виконання програми:



```
C:\Windows\system32\cmd.exe
Random array: 16 18 385 387 926 854 905 812 952 954
Sum of numbers: 6209; Maximum number: 954
Sorted array: 16 18 385 387 812 854 905 926 952 954
Random two-dimensional array:
706 90 138 210 546 232 90 698 399 90
722 90 281 690 90 448 210 546 901 546
655 90 314 167 546 443 90 655 559 90
631 203 546 663 90 659 647 90 826 534
46 482 546 510 90 20 546 510 90 277
46 22 634 202 518 90 224 22 546 822
524 518 90 36 90 22 634 620 6 546
936 546 995 546 525 546 995 546 680 90
507 90 605 866 546 591 90 378 90 320
850 546 773 546 354 90 127 842 546 631
Enter an integer which must be found in the array or press ENTER to exit: 90
Indexes: [0,1],[0,6],[0,9],[1,1],[1,4],[2,1],[2,6],[2,9],[3,4],
[3,7],[4,4],[4,8],[5,5],[6,2],[6,4],[7,9],[8,1],[8,6],
[8,8],[9,5]
Enter an integer which must be found in the array or press ENTER to exit: 138
Indexes: [0,2]
Enter an integer which must be found in the array or press ENTER to exit: 768
This number was not found.
Enter an integer which must be found in the array or press ENTER to exit:

Програма виконана успішно
Для продовження натисніть будь-яку клавішу . . .
```

Висновок:

Виконуючи дану роботу ми реалізували заповнення масивів різних розмірностей випадковими числами заданого діапазону замість введення їх з клавіатури. Було розроблено код знаходження суми елементів масиву, максимального елементу, а також пошуку і виведення координат усіх входжень введеного числа в двовірному масиві. Крім цього був знайдений цікавий алгоритм порозрядного сортування на сайті [habr.com](https://habr.com/en/post/533206/) (<https://habr.com/en/post/533206/>). Нами було здійснено детальну оптимізацію та тестування даного рішення на мові програмування високого рівня C, після цього дане рішення було реалізований на асемблері.

Комп'ютерний практикум № 5. Тема: макрозасоби мови асемблер.

Завдання:

Скласти програму на нижче наведені завдання:

- 1) переписати програму комп'ютерного практикуму № 2 з використанням макросів;
- 2) переписати програму комп'ютерного практикуму № 3 з використанням макросів;
- 3) переписати одну програму (на вибір викладача) комп'ютерного практикуму № 4 з використанням макросів.

Текст програми практикуму №2 з використанням макросів:

```
OutputString MACRO MyString
mov ah,09h
mov dx, offset MyString
int 21h
endm
OutputInteger MACRO number
LOCAL m1,m2,m3
```

```

mov bx,number
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
dseg segment para public 'data'
MyNumber dw 23
NumberFromKeyboard dw 0
ten dw 1
sign db 1, '+'
message1 db 'Enter an integer or press ENTER to exit: $'
result db 10, 'Result: $'
warning db 10, 'Input correct integer or press ENTER to exit: $'
operation db ' - $'
input db 5,5,10 dup('$');
dseg ends
CSEG SEGMENT PARA PUBLIC "CODE"
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG
PUSH DS
MOV AX, 0
PUSH AX
MOV AX, DSEG
MOV DS, AX
OutputString message1
KeyboardInput:
mov ah, 0ah
mov dx, offset input
int 21h
mov cl, input[1]
mov di, 2
cmp input[di], '-'
jne CheckNumber
mov sign[0], '-'
mov input[di], '0'
inc di
dec cl
CheckNumber:
cmp input[di], '0'
jb ShowWarning
cmp input[di], '9'
ja ShowWarning
inc di

```



```

dec cl
jnz CheckNumber
dec di
jmp ConvertStringToNumber
ShowWarning:
cmp input[2],13
je exit
OutputString warning
jmp KeyboardInput
exit:
ret
ConvertStringToNumber:
sub input[di],'0'
xor ax,ax
mov al,input[di]
mov bx,ten
mul bx
add ax,NumberFromKeyboard
mov NumberFromKeyboard,ax
mov ax,10
mov bx,ten
mul bx
mov ten,ax
dec di
cmp di,1
jne ConvertStringToNumber
cmp sign[0], '-'
jne ShowResult
xor ax,ax
sub ax,NumberFromKeyboard
mov NumberFromKeyboard,ax
ShowResult:
mov ax,NumberFromKeyboard
mov ten,ax
OutputString result
OutputInteger MyNumber
OutputString operation
OutputInteger ten
mov operation[1], '='
OutputString operation
mov ax,MyNumber
sub ax,ten
OutputInteger ax
RET
MAIN ENDP
CSEG ENDS
END MAIN

```

Текст програми практикуму №3 з використанням макросів:

```

OutputString MACRO MyString
mov ah,09h
mov dx, offset MyString
int 21h
endm
OutputInteger MACRO number
LOCAL m1,m2,m3
mov bx,number
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
GetInteger MACRO
LOCAL KeyboardInput, CheckNumber, ShowWarning, exit, ConvertStringToNumber, MacrosEnd
mov ten, 10
mov NumberX, 0
mov sign, '+'
OutputString message1
KeyboardInput:
mov ah, 0ah
mov dx, offset input
int 21h
mov cl, input[1]
mov di, 2
cmp input[di], '-'
jne CheckNumber
mov sign[0], '-'
mov input[di], '0'
inc di
dec cl
CheckNumber:
cmp input[di], '0'
jb ShowWarning
cmp input[di], '9'
ja ShowWarning
inc di
dec cl
jnz CheckNumber
dec di
jmp ConvertStringToNumber
ShowWarning:
cmp input[2], 13
je exit
OutputString warning
jmp KeyboardInput
exit:
mov ah, 4ch
int 21h
ConvertStringToNumber:
sub input[di], '0'
xor ax, ax
mov al, input[di]
mov bx, ten
mul bx
add ax, NumberX
mov NumberX, ax

```

```

mov ax,10
mov bx,ten
mul bx
mov ten,ax
dec di
cmp di,1
jne ConvertStringToNumber
cmp sign[0], '-'
jne MacrosEnd
xor ax,ax
mov ax,0
sub ax,NumberX
mov NumberX,ax
MacrosEnd:
endm
dseg segment para public 'data'
input db 5,5 dup('$');
NumberX dw 0
ten dw 1
sign db 1, '+'
result db 10, 'x = $'
func1 db 10, 'z = 8x^2 + 36 / x = $'
func1_1 db '8 * $'
func1_2 db '^ 2 + 36 / $'
func1_3 db '= $'
func2 db 10, 'z = (1 + x) / (1 - x) = $'
func2_1 db '(1 + $'
func2_2 db ') / (1 - $'
func2_3 db ') = $'
func3 db 10, 'z = 10x^2 = $'
func3_1 db '10 * $'
func3_2 db '^ 2 = $'
message1 db 10, 'Enter an integer or press ENTER to exit: $'
warning db 10, 'Input correct integer or press ENTER to exit: $'
operation db ' - $'
overload1 db 10, 'the x value must be smaller than 64$'
overload2 db 10, 'the x value must be larger than -58$'
dseg ends
CSEG SEGMENT PARA PUBLIC "CODE"
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG
PUSH DS
MOV AX,0
PUSH AX
MOV AX, DSEG
MOV DS, AX
start:
GetInteger
cmp NumberX,1
jge L1
cmp NumberX,-5
jl GoToStart
jmp L2
GoToStart:
jmp L3
L1:
cmp NumberX,64
jb equation1
OutputString overload1
jmp start
equation1:

```

```

OutputString func1
OutputString func1_1
OutputInteger NumberX
OutputString func1_2
OutputInteger NumberX
OutputString func1_3
mov ax,NumberX
mov ten,ax
mul ax
mov ten,8
mul ten
mov ten,ax
mov ax,36
div NumberX
add ax,ten
OutputInteger ax
jmp start
L2:
OutputString func2
OutputString func2_1
OutputInteger NumberX
OutputString func2_2
OutputInteger NumberX
OutputString func2_3
mov ax,1
sub ax,NumberX
mov ten,ax
inc NumberX
cmp NumberX,0
jne divide
mov ah,2
mov dl,'0'
int 21h
jmp start
divide:
xor dx,dx
div NumberX
OutputInteger ax
jmp start
L3:
cmp NumberX,-58
jg equation3
OutputString overload2
jmp start
equation3:
OutputString func3
OutputString func3_1
OutputInteger NumberX
OutputString func3_2
mov ax,NumberX
mul NumberX
mov ten,10
mul ten
OutputInteger ax
jmp start
RET
MAIN ENDP
CSEG ENDS
END MAIN

```

Текст програми практикуму №4 з використанням макросів:

```
OutputChar MACRO chr
mov ah,2
mov dl,chr
int 21h
endm
OutputString MACRO MyString
mov ah,09h
mov dx, offset MyString
int 21h
endm
OutputInteger MACRO number
LOCAL m1,m2,m3
mov bx,number
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
GetInteger MACRO
LOCAL KeyboardInput,CheckNumber,ShowWarning
LOCAL exit,ConvertStringToNumber,MacrosEnd
mov ten,1
mov number,0
mov sign, '+'
OutputString message1
KeyboardInput:
mov ah,0ah
mov dx,offset input
int 21h
mov cl,input[1]
mov di,2
cmp input[di], '-'
jne CheckNumber
mov sign[0], '-'
mov input[di], '0'
inc di
dec cl
CheckNumber:
cmp input[di], '0'
jb ShowWarning
cmp input[di], '9'
ja ShowWarning
inc di
```

```

dec cl
jnz CheckNumber
dec di
jmp ConvertStringToNumber
ShowWarning:
cmp input[2],13
je exit
OutputString warning
jmp KeyboardInput
exit:
mov ah,4ch
int 21h
ConvertStringToNumber:
sub input[di],'0'
xor ax,ax
mov al,input[di]
mov bx,ten
mul bx
add ax,number
mov number,ax
mov ax,10
mov bx,ten
mul bx
mov ten,ax
dec di
cmp di,1
jne ConvertStringToNumber
cmp sign[0], '-'
jne MacroEnd
xor ax,ax
mov ax,0
sub ax,number
mov number,ax
MacroEnd:
endm
OutputArray MACRO arr
LOCAL ShowArrayCycle
mov si,0
ShowArrayCycle:
mov ax,arr[si]
OutputInteger ax
OutputChar ' '
add si,2
cmp si,MaxInd
jb ShowArrayCycle
endm
GetRandomNumber MACRO minrand,mazrand
LOCAL label1
mov di,maxrand
mov dx,word[seed]
or dx,dx
jnz label1
mov ah,2ch
int 21h
xor ax,ax
mov al,dl
mov dx,ax
mov ax,word[seed2]
or ax,ax
in ax,40h
label1:

```

```

mul    dx
inc    ax
mov     word[seed],dx
mov     word[seed2],ax
xor     dx,dx
sub     di,minrand
inc     di
div     di
mov     ax,dx
add     ax,minrand
mov     number,ax
endm

LSTRadixSort MACRO arr
LOCAL CreateBaskets,NextStep,FillBaskets
LOCAL FillBasketsCycle,SortCycle1,SortCycle2
mov     si,0
CreateBaskets:
mov     di,arr[si]
and     di,255
mov     ax,di
mov     bx,2
mul     bx
mov     di,ax
inc     baskets[di]
mov     di,arr[si]
shr     di,8
and     di,255
add     di,256
mov     ax,di
mov     bx,2
mul     bx
mov     di,ax
inc     baskets[di]
mov     di,arr[si]
shr     di,8
shr     di,8
and     di,255
add     di,512
mov     ax,di
mov     bx,2
mul     bx
mov     di,ax
inc     baskets[di]
mov     di,arr[si]
shr     di,8
shr     di,8
shr     di,8
and     di,255
add     di,768
mov     ax,di
mov     bx,2
mul     bx
mov     di,ax
inc     baskets[di]
add     si,2
cmp     si,MaxInd
je      NextStep
jmp     CreateBaskets
NextStep:
mov     cx,0
FillBaskets:

```

```

mov si,0
mov key,0
FillBasketsCycle:
mov ax,cx
mov bx,256
mul bx
add ax,si
mov bx,2
mul bx
mov di,ax
mov ax,baskets[di]
mov value,ax
mov ax,key
mov baskets[di],ax
add ax,value
mov key,ax
inc si
cmp si,256
jb FillBasketsCycle
inc cx
cmp cx,4
jb FillBaskets
mov si,0
SortCycle1:
mov ax,arr[si]
and ax,255
mov bx,2
mul bx
mov ten,ax
mov di,ten
mov ax,baskets[di]
mov bx,2
mul bx
mov di,ax
mov ax,arr[si]
mov dest[di],ax
mov di,ten
inc baskets[di]
add si,2
cmp si,MaxInd
jb SortCycle1
mov si,0
mov si,0
SortCycle2:
mov ax,dest[si]
shr ax,8
and ax,255
mov bx,2
mul bx
add ax,512
mov ten,ax
mov di,ten
mov ax,baskets[di]
mov bx,2
mul bx
mov di,ax
mov ax,dest[si]
mov arr[di],ax
mov di,ten
inc baskets[di]
add si,2

```



```

cmp si,MaxInd
jb SortCycle2
endm
dseg segment para public 'data'
NumbersCount dw 10
MaxInd dw 0
arr dw 10 dup(0);5177,6364,9806,8745,9698,440,5204,6343,4890,7742
arr2 dw 10 DUP(10 DUP(0))
number dw 0
minrand dw 1
maxrand dw 1000
seed dw 0
seed2 dw 0
SumOfNumbers dw 0
MaxNumber dw 0
TargetNumber dw 0
input db 5,5 dup('$')
ten dw 1
sign db 1,'+'
message db 10,10 dup('$')
RandomArray db 10,'Random array: $'
StrSumOfNumbers db 10,'Sum of numbers: $'
StrMaxNumber db '; Maximum number: $'
StrSortedArray db 10,'Sorted array: $'
StrTwoDimensionalArray db 10,'Random two-dimensional array: $'
StrIndexFirst db 10,'Indexes: $'
StrNumberNotFound db 10,'This number was not found.$'
message1 db 'Enter an integer which must be found in the array or press ENTER to exit: $'
warning db 10,'Input correct integer or press ENTER to exit: $'
;--variables for LST radix sort--
dest dw 10 dup (0)
baskets dw 2048 dup (0)
key dw 0
value dw 0
dseg ends
CSEG SEGMENT PARA PUBLIC "CODE"
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG
PUSH DS
MOV AX,0
PUSH AX
MOV AX,DSEG
MOV DS,AX
mov ax,2
mul NumbersCount
mov MaxInd,ax
mov cx,NumbersCount
mov si,0
GenerateRandomNumbers:
push cx
GetRandomNumber minrand maxrand
pop cx
mov ax,number
mov arr[si],ax
add si,2
loop GenerateRandomNumbers
OutputString RandomArray
mov cx,NumbersCount
mov si,0
mov SumOfNumbers,0
FindSumAndMaximumNumber:

```

```

mov ax,arr[si]
mov number,ax
add SumOfNumbers,ax
cmp ax,MaxNumber
jb continue
mov MaxNumber,ax
continue:
push cx
mov sign,'+'
OutputInteger number
pop cx
OutputChar ' '
add si,2
cmp si,MaxInd
jb FindSumAndMaximumNumber
OutputString StrSumOfNumbers
OutputInteger SumOfNumbers
OutputString StrMaxNumber
OutputInteger MaxNumber
OutputString StrSortedArray
LSTRadixSort arr
OutputArray arr
,,,,,,TheTwoDimensionalArray,,,,,,
OutputString STRTwoDimensionalArray
OutputChar 10
mov di,0
mov cx,NumbersCount
rows:
mov si,0
push cx
columns:
push di
GetRandomNumber minrand,maxrand
OutputInteger number
pop di
mov ax,number
mov arr2[di],ax
OutputChar ' '
add di,2
inc si
cmp si,NumbersCount
jb columns
OutputChar 10
pop cx
dec cx
cmp cx,0
je SearchNumber
jmp rows
SearchNumber:
GetInteger
mov ax,number
mov TargetNumber,ax
mov di,0
mov bx,0
mov cx,NumbersCount
RowsSearchCycle:
mov si,0
push cx
ColumnsSearchCycle:
mov ax,arr2[bx][si]
cmp ax,TargetNumber

```

```

je NumberFound
jmp NextElement
NumberFound:
push bx
cmp di,0
ja IndexNotFirst
OutputString StrIndexFirst
jmp ShowIndex
IndexNotFirst:
OutputChar ','
xor dx,dx
mov ax,di
mov number,9
div number
cmp dx,0
ja ShowIndex
OutputChar 10
pop bx
push bx
ShowIndex:
OutputChar '['
xor dx,dx
mov ax,bx
div MaxInd
OutputInteger ax
OutputChar ' '
xor dx,dx
mov ax,si
mov number,2
div number
OutputInteger ax
OutputChar ']'
inc di
pop bx
NextElement:
add si,2
cmp si,MaxInd
je GoToTheNextRow
jmp ColumnsSearchCycle
GoToTheNextRow:
add bx,MaxInd
pop cx
dec cx
cmp cx,0
je NextRow
jmp RowsSearchCycle;
NextRow:
cmp di,0
je NumberNotFound
OutputChar 10
jmp SearchNumber
NumberNotFound:
OutputString StrNumberNotFound
OutputChar 10
jmp SearchNumber
RET
MAIN ENDP
CSEG ENDS
END MAIN

```

Тест програми з практику №2, реалізованої з використанням макросів:

```
C:\Windows\system32\cmd.exe
Enter an integer or press ENTER to exit: 0/7
Input correct integer or press ENTER to exit: 92
Result: 23 - 92 = -69
Програма виконана успішно
```

```
C:\Windows\system32\cmd.exe
Enter an integer or press ENTER to exit: -999
Result: 23 - -999 = 1022
Програма виконана успішно
```

Тест програми з практику №3, реалізованої з використанням макросів:

```
C:\Windows\system32\cmd.exe
Enter an integer or press ENTER to exit: 7777
the x value must be smaller than 64
Enter an integer or press ENTER to exit: 63
z = 8x^2 + 36 / x = 8 * 63 ^ 2 + 36 / 63 = 31752
Enter an integer or press ENTER to exit: 1
z = 8x^2 + 36 / x = 8 * 1 ^ 2 + 36 / 1 = 44
Enter an integer or press ENTER to exit: 0
z = (1 + x) / (1 - x) = (1 + 0) / (1 - 0) = 1
Enter an integer or press ENTER to exit: -4
z = (1 + x) / (1 - x) = (1 + -4) / (1 - -4) = 0
Enter an integer or press ENTER to exit: -5
z = (1 + x) / (1 - x) = (1 + -5) / (1 - -5) = 0
Enter an integer or press ENTER to exit: -20
z = 10x^2 = 10 * -20 ^ 2 = 4000
Enter an integer or press ENTER to exit: -100
the x value must be larger than -58
Enter an integer or press ENTER to exit: -57
z = 10x^2 = 10 * -57 ^ 2 = 32490
Enter an integer or press ENTER to exit:
Програма виконана успішно
```

Тест програми з практику №4, реалізованої з використанням макросів:

```
C:\Windows\system32\cmd.exe
Random array: 326 328 306 226 154 310 450 452 632 530
Sum of numbers: 3714; Maximum number: 632
Sorted array: 154 226 306 310 326 328 450 452 530 632
Random two-dimensional array:
154 90 858 90 58 178 802 546 154 90
766 90 850 634 118 172 634 8 178 140
524 84 964 90 780 178 140 546 326 90
172 634 282 634 268 546 164 90 916 634
358 546 650 90 588 634 810 178 66 778
46 292 90 602 634 776 906 90 396 634
938 178 676 778 90 912 634 940 586 546
844 90 816 634 940 694 90 796 634 516
46 562 178 14 178 834 546 232 546 786
46 196 90 690 178 676 676 634 566 178
Enter an integer which must be found in the array or press ENTER to exit: 140
Indexes: [1,9],[2,6]
Enter an integer which must be found in the array or press ENTER to exit: 546
Indexes: [0,7],[2,7],[3,5],[4,1],[6,9],[8,6],[8,8]
Enter an integer which must be found in the array or press ENTER to exit: 90
Indexes: [0,1],[0,3],[0,9],[1,1],[2,3],[2,9],[3,7],[4,3],[5,2],
[5,7],[6,4],[7,1],[7,6],[9,2]
Enter an integer which must be found in the array or press ENTER to exit: 9999
This number was not found.
Enter an integer which must be found in the array or press ENTER to exit: 89-
Input correct integer or press ENTER to exit: 778
Indexes: [4,9],[6,3]
Enter an integer which must be found in the array or press ENTER to exit:
Програма виконана успішно
```

Висновок:

Для спрощення організації коду в мові асемблер використовують макроси. Вони є майже повними аналогами функцій в мовах програмування високого рівня, дозволяючи уникнути повторення коду, реалізувати передачу змінних у вигляді параметрів, оголошувати локальні мітки за допомогою ключового слова `local`, що дає можливість скоротити частоту використання повністю глобальних змінних. На відміну від процедур, макроси можуть створювати проблеми при переході на відстань більшу ніж 128 байт, оскільки розміщуються на низькому рівні так, наче ми їх використовуємо у вигляді звичайного коду. Другим недоліком макросів, на мою думку, є неможливість повертати значення подібно до мов програмування високого рівня: у цьому випадку результати мають бути розміщені в глобальній змінній прямо в тілі макросу.