МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпропетровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №6**

**з дисципліни «архітектура комп’ютера»**

**на тему: «Організація введення та виведення інформації в консолі на рівні Ассемблера. Обробка масивів чисел в мові Ассемблера.»**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Нежуміра О. І.

Дніпро, 2019

**Лабораторна робота №6.2**

**Тема**. Організація введення та виведення інформації в консолі на рівні Ассемблера. Обробка масивів чисел в мові Ассемблера.

**Мета**. Вивчити засоби мови Ассемблера для введення та виведення числової інформації. Набути практичних навичок з обробки двомірних числових масивів мовою Ассемблера.

**Постановка задачи**

1. Загальне завдання

Дано одновимірний масив. Елементи масиву – цілі числа зі знаком з діапазону слово. Довжина масиву від 5 до 10 елементів. Обробити масив згідно індивідуального завдання з таблиці 1.

2. Індивідуальне завдання

|  |  |
| --- | --- |
| 7 | Підрахувати суму та кількість елементів, які більші за задане число. |

3. Вимоги до програми:

- вхідні дані вводяться з клавіатури;

- результати роботи програми виводяться на екран.

4. Вимоги до вхідних даних:

- рядок символів довільної довжини, але не більше 80 символів, який може містити пробіли, великі та малі літери англійського алфавіту, цифри (перевіряти зміст рядка не треба, рядок завжди задається правильно);

- рядок починається з символа;

- словом вважається будь-яка послідовність символів, що відрізняється від пробілу; - слова відділяються одним пробілом;

- окрім рядка введенню з клавіатури підлягає додаткова інформація (символи).

5. Вимоги до тексту програми:

- коментарі щодо призначення програми, її вхідних і вихідних даних;

- коментарі щодо призначення кожного блоку програми, дій окремих команд для пояснення алгоритму;

- самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних і міток.

**Розробка тестів**

1. Тести для перевірки на правильність вводу даних

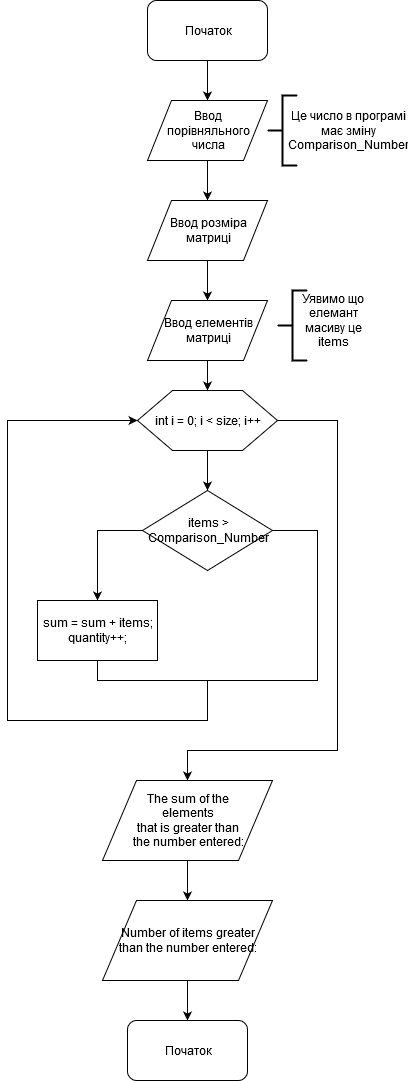
|  |  |  |  |
| --- | --- | --- | --- |
| № | Вхідні дані | Очікувані результати | Підтвердження результату |
| 1 | Порівнювальне число (Comparison\_Number): |  | + |
|  | нормальне число = 15 | буде записане в зміну | + |
|  | символ = dsgv | error | + |
|  | (пусто) | error | + |
| 2 | Розмір матриці: |  | + |
|  | менше мінімального значення = 4 | error | + |
|  | більше максимального значення = 11 | error | + |
|  | символ = dsgv | error | + |
|  | нормальне число = 6 | буде записане в зміну | + |
| 3 | Елементи матриці: |  | + |
|  | нормальне число = 1, 4, 47 | буде записане в зміну | + |
|  | символ = dsgv | error | + |
|  | (пусто) | error | + |

2. Тести для результату

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вхідні дані | Очікувані результати | Підтвердження результату |
| 1 | Нульовий результат:  порівнювальне число = 15  розмір матриці = 6  елем. матриці = 1, 2, 3, 4, 5, 6 | The sum of the elements that is greater than the number entered: 0  Enter the number of elements in the array N (5 <= N <= 10): 0 | + |
| 2 | Всі елементи пройшли провірку:  порівнювальне число = 15  розмір матриці = 6  елем. матриці = 16, 81, 36, 18, 98, 135 | The sum of the elements that is greater than the number entered: 379  Enter the number of elements in the array N (5 <= N <= 10): 6 | + |

**Алгоритм програм**

1.Блок-схема



2.Текст програми

;-------------------------------------------------------------------------------

;Програма роботи з масивом чисел.

;Елементи масиву – цілі числа зі знаком з діапазону слово.

;Вхід: розмір масиву та його елементи вводяться з клавіатури, число с яким будемо порівнювати елементи масива.

;Вихід: оновлений масив виводиться на консоль, кількість та сума елементів масиву які більше за задане число.

;-------------------------------------------------------------------------------

;сегмент стеку

stk segment stack

db 256 dup (?)

stk ends

;сегмент даних

data segment para public 'data'

len dw 10 ;максимальний розмір масиву

counter dw ? ;розмір масиву

division db 5

sum dw 0

quantity dw 0

A dw 10 dup (?) ;масив

Comparison\_Number dw ? ;число с яким будемо порівнювати елементи масива(число должен вводить пользеватель)

;буфер для введення числа в ASCII-формі (максимум 6 символів (-32768) + кінець рядка (0dh))

buffer db 7, ?, 7 dup (?)

buffer2 db 7, ?, 7 dup (?)

msgQuantity db 'Number of items greater than the number entered: $'

msgPrintsum db 'The sum of the elements that is greater than the number entered: $'

msgCounter db 'Enter the number of elements in the array N (5 <= N <= 10): $'

msgArray db 'Enter array elements:', 0ah, 0dh, '$'

msgPrompt1 db 'A [ $'

msgPrompt2 db ' ] = $'

msgError db 0ah, 0dh, 'ERROR!', 0ah, 0dh, '$'

msgPrint db 'Array :', 0ah, 0dh, '$'

msgPause db 'Press any key ...', '$'

CrLf db 0ah, 0dh, '$'

data ends

;сегмент команд

code segment para public 'code'

assume cs:code, ds:data, ss:stk

main:

mov ax, data

mov ds, ax

mInCounter1:

lea dx, msgQuantity ;виведення запрошення для введення числа

call outString

lea dx, buffer2 ;DX = адреса буфера

mov ah, 0ah ;введення рядка в буфер

int 21h

cmp buffer2 + 1, 0 ;перевірка довжини рядка

call convertStrToInt ;перетворення рядка в слово (число зі знаком)

jc mErrorCounter1 ;якщо помилка, то виведення повідомлення

mov Comparison\_Number, ax

lea dx, CrLf

call outString

jmp mEndLoop1

mErrorCounter1:

lea dx, msgError

call outString ;виведення повідомлення 'ERROR!'

jmp mInCounter1 ;перехід до початку циклу

mEndLoop1:

;введення кількості елементів масиву (цикл повторюється до введення "правильного" числа)

mInCounter:

lea dx, msgCounter ;виведення запрошення для введення числа

call outString

lea dx, buffer ;DX = адреса буфера

mov ah, 0ah ;введення рядка в буфер

int 21h

cmp buffer + 1, 0 ;перевірка довжини рядка

jz mInCounter ;якщо довжина = 0, перехід до початку циклу

call convertStrToInt ;перетворення рядка в слово (число зі знаком)

jc mErrorCounter ;якщо помилка, то виведення повідомлення

cmp ax, len ;якщо кількість елементів > розміру масиву,

jg mErrorCounter ;то перехід на повідомлення про помилку

cmp ax, 4 ;якщо кількість елементів <= 4,

jle mErrorCounter ;то перехід на повідомлення про помилку

mov counter, ax ;збереження кількості елементів масиву

lea dx, CrLf ;виведення управляючих символів

call outString

jmp mEndLoop ;вихід з циклу введення кількості елементів

mErrorCounter:

lea dx, msgError

call outString ;виведення повідомлення 'ERROR!'

jmp mInCounter ;перехід до початку циклу

mEndLoop:

;введення елементів масиву

lea dx, msgArray ;виведення запрошення для елементів масиву

call outString

mov cx, counter ;лічильник циклу введення елементів масиву

xor si, si ;індекс елементів масиву

;введення елементу масиву

mInArray:

call outPrompt ;виведення запрошення для елемента масиву

lea dx, buffer ;DX = адреса буфера

mov ah, 0ah ;введення рядка в буфер

int 21h

cmp buffer + 1, 0 ;перевірка довжини рядка

jz mInArray ;якщо довжина = 0, перехід до початку циклу

call convertStrToInt ;перетворення рядка в число зі знаком (слово)

jnc mElementOk ;якщо немає помилки, то збереження елементу

lea dx, msgError ;інакше виведення повідомлення 'ERROR!'

call outString

jmp mInArray ;перехід до початку циклу

mElementOk:

mov di, si ;DI = індекс поточного елементу масиву

add di, di ;DI = зміщення поточного елементу

mov A[di], ax ;збереження поточного елементу масиву

inc si ;перехід до наступного елементу масиву

lea dx, CrLf ;виведення управляючих символів

call outString

loop mInArray ;якщо не всі елементи введені - до початку циклу

;виведення масиву на консоль

lea dx, msgPrint ;виведення повідомлення

call outString

mov cx, counter ;лічильник циклу виведення елементів масиву

xor si, si ;індекс поточного елементу масиву A

mPrintArray:

call outPrompt ;виведення "імені" елемента масиву

mov di, si ;DI = індекс поточного елементу масиву

add di, di ;DI = зміщення поточного елементу

mov ax, A[di] ;AX = поточний елемент

call printInt ;виведення поточного елементу

lea dx, CrLf ;виведення управляючих символів

call outString

inc si ;перехід до наступного елементу масиву

loop mPrintArray ;не всі елементи виведені - на початок циклу

xor di, di

mov cx, counter

mov bx, 0

mCondition:

mov ax, A[di]

cmp ax, Comparison\_Number

jl mNext

add bx, A[di]

inc quantity

mNext:

inc di

inc di

loop mCondition

mPrintsum:

lea dx, msgPrintsum

call outString

mov ax, bx

call printInt

lea dx, CrLf ;виведення управляючих символів

call outString

lea dx, msgQuantity

call outString

mov ax, quantity

call printInt

lea dx, CrLf ;виведення управляючих символів

call outString

;затримка виконання та вихід з програми

lea dx, msgPause

call outString ;виведення повідомлення 'Press any key...'

mov ah, 08h ;затримка виконання програми

int 21h

mov ax, 4C00h ;завершення програми

int 21h

;-------------------------------------------------------------------------------

;процедура виведення рядка символів, обмежених символом '$' на консоль

;вхід: DX - адреса рядка

;-------------------------------------------------------------------------------

outString proc near

mov ah, 09h

int 21h

ret

outString endp

;-------------------------------------------------------------------------------

;процедура виведення на консоль імені масиву та індексу елемента

;-------------------------------------------------------------------------------

outPrompt proc near

lea dx, msgPrompt1 ;виведення імені масиву

call outString

mov ax, si ;AX = індекс елементу масиву

call printInt ;виведення індексу

lea dx, msgPrompt2

call outString

ret

outPrompt endp

;-------------------------------------------------------------------------------

;процедура перетворення рядка в число зі знаком в форматі слова

;(діапазон значення числа від -32768 до 32767)

;вхід: DX - адреса буфера

;вихід: AX - число в форматі слова (в разі помилки AX = 0)

; CF = 1 - помилка

;-------------------------------------------------------------------------------

convertStrToInt proc near

;збереження регістрів

push bx

push cx

push dx

push si

push di

push bp

mov si, dx ;SI - адреса буфера

xor cx, cx

mov cl, [si + 1] ;CL = довжина введеного рядка

;перевірка знаку числа

add si, 2 ;SI - адреса рядка

mov bl, [si] ;BL = перший символ рядка

cmp bl, '-' ;порівняння першого символу з '-'

jne mNoSign ;якщо не '-', то перетворюємо рядок як число без знаку

inc si ;інкремент адреси рядка

dec cl ;декремент довжини рядка

;перетворення рядка у слово без знаку

mNoSign:

mov bp, bx ;зберігаємо регістр

xor bx, bx

mov di, 10 ;DI = множник 10 (база системи числення)

xor ax, ax ;AX = 0, для обчислення числа, яке введене з клавіатури

mScanString:

mov bl, [si] ;завантаження в BL чергового символу рядка

inc si ;інкремент адреси

cmp bl, '0' ;якщо код символу менше коду '0'

jb mErrorValue ;повертаємо помилку

cmp bl, '9' ;якщо код символу більше коду '9'

ja mErrorValue ;повертаємо помилку

sub bl, '0' ;перетворення символу-цифри в число

mul di ;AX = AX \* 10

jc mErrorValue ;якщо результат більше 16 бітів - помилка

add ax, bx ;додаємо цифру

jc mErrorValue ;якщо переповнення - помилка

loop mScanString ;продовжуємо сканувати рядок

;обробка знаку числа та перевірка діапазону для від'ємного числа

mov bx, bp ;відновлюємо регістр

cmp bl, '-' ;знову перевіряємо знак

jne mPlus ;якщо перший символ не '-', то число додатне

cmp ax, 32768 ;модуль від'ємного числа повинен бути не більш 32768

ja mErrorValue ;якщо більше (без знаку), повертаємо помилку

neg ax ;інвертуємо число (число - від'ємне)

jmp mOk ;переходимо до нормального завершення процедури

;перевірка діапазону для додатного числа

mPlus:

cmp ax, 32767 ;додатне число повинно бути не більше 32767

ja mErrorValue ;якщо більше, повертаємо помилку

;успішне перетворення рядка у число в форматі слова

mOk:

clc ;CF = 0 ознака успішного перетворення рядка у число

jmp mExit ;переходимо до виходу з процедури

;помилка перетворення рядка у число в форматі слова

mErrorValue:

xor ax, ax ;AX = 0

stc ;CF = 1 повертаємо помилку

mExit:

;відновлення регістрів

pop bp

pop di

pop si

pop dx

pop cx

pop bx

ret

convertStrToInt endp

;-------------------------------------------------------------------------------

;процедура виведення цілого числа у форматі слова на консоль

;вхід: AX - ціле число

;-------------------------------------------------------------------------------

printInt proc near

;збереження регістрів

push ax

push bx

push cx

push dx

;обробка знаку числа AX

cmp ax, 0 ;якщо число додатне

jge mPositive ;перехід до його обробки

neg ax ;інакше - перетворення від'ємного числа на додатне

push ax

mov dl, '-' ;виведення на консоль

mov ah, 02h ;знаку '-'

int 21h

pop ax

;обробка додатного числа

mPositive:

mov bx, 10

xor cx, cx ;лічильник цифр числа

;отримуємо цифри числа AX

mDivisionNumber:

cwd

div bx ;ділимо з остачею

push dx ;зберігаємо остачу в стеку

inc cx ;збільшуємо лічильник цифр числа

or ax, ax ;якщо число ще не 0,

jnz mDivisionNumber ;продовжуємо цикл

;виведення на консоль цифр числа AX

mPrintChar:

pop dx ;DX = цифра числа

add dl, 30h ;DL = символ цифри числа

mov ah, 02h ;виведення символу-цифри на консоль

int 21h

loop mPrintChar ;лічильник цифр числа не 0 - продовжуємо цикл

;відновлення регістрів

pop dx

pop cx

pop bx

pop ax

ret

printInt endp

;-------------------------------------------------------------------------------

code ends

end main

3.Лістинг

Turbo Assembler Version 3.0 01/25/20 02:23:40 Page 1

lab61.asm

1 ;-------------------------------------------------------------------------------

2 ;Програма роботи з масивом чисел.

3 ;Елементи масиву – цілі числа зі знаком з діапазону слово.

4 ;Вхід: розмір масиву та його елементи вводяться з клавіатури, число с яким будемо порівнювати +

5 елементи масива.

6 ;Вихід: оновлений масив виводиться на консоль, кількість та сума елементів масиву які більше за+

7 задане число.

8 ;-------------------------------------------------------------------------------

9

10

11 ;сегмент стеку

12 0000 stk segment stack

13 0000 0100\*(??) db 256 dup (?)

14 0100 stk ends

15

16 ;сегмент даних

17 0000 data segment para public 'data'

18 0000 000A len dw 10 ;максимальний розмір масиву

19 0002 ???? counter dw ? ;розмір масиву

20 0004 05 division db 5

21 0005 0000 sum dw 0

22 0007 0000 quantity dw 0

23 0009 0A\*(????) A dw 10 dup (?) ;масив

24 001D ???? Comparison\_Number dw ? ;число с яким будемо порівнювати елементи масива(число должен вводить +

25 пользеватель)

26 ;буфер для введення числа в ASCII-формі (максимум 6 символів (-32768) + кінець рядка (0dh))

27 001F 07 ?? 07\*(??) buffer db 7, ?, 7 dup (?)

28 0028 07 ?? 07\*(??) buffer2 db 7, ?, 7 dup (?)

29 0031 4E 75 6D 62 65 72 20+ msgQuantity db 'Number of items greater than the number entered: $'

30 6F 66 20 69 74 65 6D+

31 73 20 67 72 65 61 74+

32 65 72 20 74 68 61 6E+

33 20 74 68 65 20 6E 75+

34 6D 62 65 72 20 65 6E+

35 74 65 72 65 64 3A 20+

36 24

37 0063 54 68 65 20 73 75 6D+ msgPrintsum db 'The sum of the elements that is greater than the number entered: $'

38 20 6F 66 20 74 68 65+

39 20 65 6C 65 6D 65 6E+

40 74 73 20 74 68 61 74+

41 20 69 73 20 67 72 65+

42 61 74 65 72 20 74 68+

43 61 6E 20 74 68 65 20+

44 6E 75 6D 62 65 72 20+

45 65 6E 74 65 72 65 64+

46 3A 20 24

47 00A5 45 6E 74 65 72 20 74+ msgCounter db 'Enter the number of elements in the array N (5 <= N <= 10) : $'

48 68 65 20 6E 75 6D 62+

49 65 72 20 6F 66 20 65+

50 6C 65 6D 65 6E 74 73+

51 20 69 6E 20 74 68 65+

52 20 61 72 72 61 79 20+

53 4E 20 28 35 20 3C 3D+

54 20 4E 20 3C 3D 20 31+

55 30 29 20 3A 20 24

56 00E3 45 6E 74 65 72 20 61+ msgArray db 'Enter array elements:', 0ah, 0dh, '$'

57 72 72 61 79 20 65 6C+

58 65 6D 65 6E 74 73 3A+

59 0A 0D 24

60 00FB 41 20 5B 20 24 msgPrompt1 db 'A [ $'

61 0100 20 5D 20 3D 20 24 msgPrompt2 db ' ] = $'

62 0106 0A 0D 45 52 52 4F 52+ msgError db 0ah, 0dh, 'ERROR!', 0ah, 0dh, '$'

63 21 0A 0D 24

64 0111 41 72 72 61 79 20 3A+ msgPrint db 'Array :', 0ah, 0dh, '$'

65 0A 0D 24

66 011B 50 72 65 73 73 20 61+ msgPause db 'Press any key ...', '$'

67 6E 79 20 6B 65 79 20+

68 2E 2E 2E 24

69 012D 0A 0D 24 CrLf db 0ah, 0dh, '$'

70 0130 data ends

71

72 ;сегмент команд

73 0000 code segment para public 'code'

74 assume cs:code, ds:data, ss:stk

75 0000 main:

76 0000 B8 0000s mov ax, data

77 0003 8E D8 mov ds, ax

78

79 0005 mInCounter1:

80 0005 BA 0031r lea dx, msgQuantity ;виведення запрошення для введення числа

81 0008 E8 010C call outString

82 000B BA 0028r lea dx, buffer2 ;DX = адреса буфера

83 000E B4 0A mov ah, 0ah ;введення рядка в буфер

84 0010 CD 21 int 21h

85 0012 80 3E 0029r 00 cmp buffer2 + 1, 0 ;перевірка довжини рядка

86 0017 E8 0114 call convertStrToInt ;перетворення рядка в слово (число зі знаком)

87 001A 72 0C jc mErrorCounter1 ;якщо помилка, то виведення повідомлення

88 001C A3 001Dr mov Comparison\_Number, ax

89 001F BA 012Dr lea dx, CrLf

90 0022 E8 00F2 call outString

91 0025 EB 09 90 jmp mEndLoop1

92 0028 mErrorCounter1:

93 0028 BA 0106r lea dx, msgError

94 002B E8 00E9 call outString ;виведення повідомлення 'ERROR!'

95 002E EB D5 jmp mInCounter1 ;перехід до початку циклу

96 0030 mEndLoop1:

97

98 ;введення кількості елементів масиву (цикл повторюється до введення "правильного" числа)

99 0030 mInCounter:

100 0030 BA 00A5r lea dx, msgCounter ;виведення запрошення для введення числа

101 0033 E8 00E1 call outString

102 0036 BA 001Fr lea dx, buffer ;DX = адреса буфера

103 0039 B4 0A mov ah, 0ah ;введення рядка в буфер

104 003B CD 21 int 21h

105 003D 80 3E 0020r 00 cmp buffer + 1, 0 ;перевірка довжини рядка

106 0042 74 EC jz mInCounter ;якщо довжина = 0, перехід до початку циклу

107 0044 E8 00E7 call convertStrToInt ;перетворення рядка в слово (число зі знаком)

108 0047 72 17 jc mErrorCounter ;якщо помилка, то виведення повідомлення

109 0049 3B 06 0000r cmp ax, len ;якщо кількість елементів > розміру масиву,

110 004D 7F 11 jg mErrorCounter ;то перехід на повідомлення про помилку

111 004F 3D 0004 cmp ax, 4 ;якщо кількість елементів <= 4,

112 0052 7E 0C jle mErrorCounter ;то перехід на повідомлення про помилку

113 0054 A3 0002r mov counter, ax ;збереження кількості елементів масиву

114 0057 BA 012Dr lea dx, CrLf ;виведення управляючих символів

115 005A E8 00BA call outString

116 005D EB 09 90 jmp mEndLoop ;вихід з циклу введення кількості елементів

117 0060 mErrorCounter:

118 0060 BA 0106r lea dx, msgError

119 0063 E8 00B1 call outString ;виведення повідомлення 'ERROR!'

120 0066 EB C8 jmp mInCounter ;перехід до початку циклу

121 0068 mEndLoop:

122 ;введення елементів масиву

123 0068 BA 00E3r lea dx, msgArray ;виведення запрошення для елементів масиву

124 006B E8 00A9 call outString

125 006E 8B 0E 0002r mov cx, counter ;лічильник циклу введення елементів масиву

126 0072 33 F6 xor si, si ;індекс елементів масиву

127 ;введення елементу масиву

128 0074 mInArray:

129 0074 E8 00A5 call outPrompt ;виведення запрошення для елемента масиву

130 0077 BA 001Fr lea dx, buffer ;DX = адреса буфера

131 007A B4 0A mov ah, 0ah ;введення рядка в буфер

132 007C CD 21 int 21h

133 007E 80 3E 0020r 00 cmp buffer + 1, 0 ;перевірка довжини рядка

134 0083 74 EF jz mInArray ;якщо довжина = 0, перехід до початку циклу

135 0085 E8 00A6 call convertStrToInt ;перетворення рядка в число зі знаком (слово)

136 0088 73 08 jnc mElementOk ;якщо немає помилки, то збереження елементу

137 008A BA 0106r lea dx, msgError ;інакше виведення повідомлення 'ERROR!'

138 008D E8 0087 call outString

139 0090 EB E2 jmp mInArray ;перехід до початку циклу

140 0092 mElementOk:

141 0092 8B FE mov di, si ;DI = індекс поточного елементу масиву

142 0094 03 FF add di, di ;DI = зміщення поточного елементу

143 0096 89 85 0009r mov A[di], ax ;збереження поточного елементу масиву

144 009A 46 inc si ;перехід до наступного елементу масиву

145 009B BA 012Dr lea dx, CrLf ;виведення управляючих символів

146 009E E8 0076 call outString

147 00A1 E2 D1 loop mInArray ;якщо не всі елементи введені - до початку циклу

148 ;виведення масиву на консоль

149 00A3 BA 0111r lea dx, msgPrint ;виведення повідомлення

150 00A6 E8 006E call outString

151 00A9 8B 0E 0002r mov cx, counter ;лічильник циклу виведення елементів масиву

152 00AD 33 F6 xor si, si ;індекс поточного елементу масиву A

153 00AF mPrintArray:

154 00AF E8 006A call outPrompt ;виведення "імені" елемента масиву

155 00B2 8B FE mov di, si ;DI = індекс поточного елементу масиву

156 00B4 03 FF add di, di ;DI = зміщення поточного елементу

157 00B6 8B 85 0009r mov ax, A[di] ;AX = поточний елемент

158 00BA E8 00D2 call printInt ;виведення поточного елементу

159 00BD BA 012Dr lea dx, CrLf ;виведення управляючих символів

160 00C0 E8 0054 call outString

161 00C3 46 inc si ;перехід до наступного елементу масиву

162 00C4 E2 E9 loop mPrintArray ;не всі елементи виведені - на початок циклу

163 00C6 33 FF xor di, di

164 00C8 8B 0E 0002r mov cx, counter

165 00CC BB 0000 mov bx, 0

166 00CF mCondition:

167 00CF 8B 85 0009r mov ax, A[di]

168 00D3 3B 06 001Dr cmp ax, Comparison\_Number

169 00D7 7C 08 jl mNext

170 00D9 03 9D 0009r add bx, A[di]

171 00DD FF 06 0007r inc quantity

172 00E1 mNext:

173 00E1 47 inc di

174 00E2 47 inc di

175 00E3 E2 EA loop mCondition

176 00E5 mPrintsum:

177 00E5 BA 0063r lea dx, msgPrintsum

178 00E8 E8 002C call outString

179 00EB 8B C3 mov ax, bx

180 00ED E8 009F call printInt

181 00F0 BA 012Dr lea dx, CrLf ;виведення управляючих символів

182 00F3 E8 0021 call outString

183 00F6 BA 0031r lea dx, msgQuantity

184 00F9 E8 001B call outString

185 00FC A1 0007r mov ax, quantity

186 00FF E8 008D call printInt

187 0102 BA 012Dr lea dx, CrLf ;виведення управляючих символів

188 0105 E8 000F call outString

189 ;затримка виконання та вихід з програми

190 0108 BA 011Br lea dx, msgPause

191 010B E8 0009 call outString ;виведення повідомлення 'Press any key...'

192 010E B4 08 mov ah, 08h ;затримка виконання програми

193 0110 CD 21 int 21h

194 0112 B8 4C00 mov ax, 4C00h ;завершення програми

195 0115 CD 21 int 21h

196 ;-------------------------------------------------------------------------------

197 ;процедура виведення рядка символів, обмежених символом '$' на консоль

198 ;вхід: DX - адреса рядка

199 ;-------------------------------------------------------------------------------

200 0117 outString proc near

201 0117 B4 09 mov ah, 09h

202 0119 CD 21 int 21h

203 011B C3 ret

204 011C outString endp

205 ;-------------------------------------------------------------------------------

206 ;процедура виведення на консоль імені масиву та індексу елемента

207 ;-------------------------------------------------------------------------------

208 011C outPrompt proc near

209 011C BA 00FBr lea dx, msgPrompt1 ;виведення імені масиву

210 011F E8 FFF5 call outString

211 0122 8B C6 mov ax, si ;AX = індекс елементу масиву

212 0124 E8 0068 call printInt ;виведення індексу

213 0127 BA 0100r lea dx, msgPrompt2

214 012A E8 FFEA call outString

215 012D C3 ret

216 012E outPrompt endp

217 ;-------------------------------------------------------------------------------

218 ;процедура перетворення рядка в число зі знаком в форматі слова

219 ;(діапазон значення числа від -32768 до 32767)

220 ;вхід: DX - адреса буфера

221 ;вихід: AX - число в форматі слова (в разі помилки AX = 0)

222 ; CF = 1 - помилка

223 ;-------------------------------------------------------------------------------

224 012E convertStrToInt proc near

225 ;збереження регістрів

226 012E 53 push bx

227 012F 51 push cx

228 0130 52 push dx

229 0131 56 push si

230 0132 57 push di

231 0133 55 push bp

232 0134 8B F2 mov si, dx ;SI - адреса буфера

233 0136 33 C9 xor cx, cx

234 0138 8A 4C 01 mov cl, [si + 1] ;CL = довжина введеного рядка

235 ;перевірка знаку числа

236 013B 83 C6 02 add si, 2 ;SI - адреса рядка

237 013E 8A 1C mov bl, [si] ;BL = перший символ рядка

238 0140 80 FB 2D cmp bl, '-' ;порівняння першого символу з '-'

239 0143 75 03 jne mNoSign ;якщо не '-', то перетворюємо рядок як число +

240 без знаку

241 0145 46 inc si ;інкремент адреси рядка

242 0146 FE C9 dec cl ;декремент довжини рядка

243 ;перетворення рядка у слово без знаку

244 0148 mNoSign:

245 0148 8B EB mov bp, bx ;зберігаємо регістр

246 014A 33 DB xor bx, bx

247 014C BF 000A mov di, 10 ;DI = множник 10 (база системи числення)

248 014F 33 C0 xor ax, ax ;AX = 0, для обчислення числа, яке введене з +

249 клавіатури

250 0151 mScanString:

251 0151 8A 1C mov bl, [si] ;завантаження в BL чергового символу рядка

252 0153 46 inc si ;інкремент адреси

253 0154 80 FB 30 cmp bl, '0' ;якщо код символу менше коду '0'

254 0157 72 2C jb mErrorValue ;повертаємо помилку

255 0159 80 FB 39 cmp bl, '9' ;якщо код символу більше коду '9'

256 015C 77 27 ja mErrorValue ;повертаємо помилку

257 015E 80 EB 30 sub bl, '0' ;перетворення символу-цифри в число

258 0161 F7 E7 mul di ;AX = AX \* 10

259 0163 72 20 jc mErrorValue ;якщо результат більше 16 бітів - помилка

260 0165 03 C3 add ax, bx ;додаємо цифру

261 0167 72 1C jc mErrorValue ;якщо переповнення - помилка

262 0169 E2 E6 loop mScanString ;продовжуємо сканувати рядок

263 ;обробка знаку числа та перевірка діапазону для від'ємного числа

264 016B 8B DD mov bx, bp ;відновлюємо регістр

265 016D 80 FB 2D cmp bl, '-' ;знову перевіряємо знак

266 0170 75 0A jne mPlus ;якщо перший символ не '-', то число додатне

267 0172 3D 8000 cmp ax, 32768 ;модуль від'ємного числа повинен бути не більш +

268 32768

269 0175 77 0E ja mErrorValue ;якщо більше (без знаку), повертаємо помилку

270 0177 F7 D8 neg ax ;інвертуємо число (число - від'ємне)

271 0179 EB 06 90 jmp mOk ;переходимо до нормального завершення +

272 процедури

273 ;перевірка діапазону для додатного числа

274 017C mPlus:

275 017C 3D 7FFF cmp ax, 32767 ;додатне число повинно бути не більше 32767

276 017F 77 04 ja mErrorValue ;якщо більше, повертаємо помилку

277 ;успішне перетворення рядка у число в форматі слова

278 0181 mOk:

279 0181 F8 clc ;CF = 0 ознака успішного перетворення рядка у +

280 число

281 0182 EB 04 90 jmp mExit ;переходимо до виходу з процедури

282 ;помилка перетворення рядка у число в форматі слова

283 0185 mErrorValue:

284 0185 33 C0 xor ax, ax ;AX = 0

285 0187 F9 stc ;CF = 1 повертаємо помилку

286 0188 mExit:

287 ;відновлення регістрів

288 0188 5D pop bp

289 0189 5F pop di

290 018A 5E pop si

291 018B 5A pop dx

292 018C 59 pop cx

293 018D 5B pop bx

294 018E C3 ret

295 018F convertStrToInt endp

296 ;-------------------------------------------------------------------------------

297 ;процедура виведення цілого числа у форматі слова на консоль

298 ;вхід: AX - ціле число

299 ;-------------------------------------------------------------------------------

300 018F printInt proc near

301 ;збереження регістрів

302 018F 50 push ax

303 0190 53 push bx

304 0191 51 push cx

305 0192 52 push dx

306 ;обробка знаку числа AX

307 0193 3D 0000 cmp ax, 0 ;якщо число додатне

308 0196 7D 0A jge mPositive ;перехід до його обробки

309 0198 F7 D8 neg ax ;інакше - перетворення від'ємного числа на +

310 додатне

311 019A 50 push ax

312 019B B2 2D mov dl, '-' ;виведення на консоль

313 019D B4 02 mov ah, 02h ;знаку '-'

314 019F CD 21 int 21h

315 01A1 58 pop ax

316 ;обробка додатного числа

317 01A2 mPositive:

318 01A2 BB 000A mov bx, 10

319 01A5 33 C9 xor cx, cx ;лічильник цифр числа

320 ;отримуємо цифри числа AX

321 01A7 mDivisionNumber:

322 01A7 99 cwd

323 01A8 F7 F3 div bx ;ділимо з остачею

324 01AA 52 push dx ;зберігаємо остачу в стеку

325 01AB 41 inc cx ;збільшуємо лічильник цифр числа

326 01AC 0B C0 or ax, ax ;якщо число ще не 0,

327 01AE 75 F7 jnz mDivisionNumber ;продовжуємо цикл

328 ;виведення на консоль цифр числа AX

329 01B0 mPrintChar:

330 01B0 5A pop dx ;DX = цифра числа

331 01B1 80 C2 30 add dl, 30h ;DL = символ цифри числа

332 01B4 B4 02 mov ah, 02h ;виведення символу-цифри на консоль

333 01B6 CD 21 int 21h

334 01B8 E2 F6 loop mPrintChar ;лічильник цифр числа не 0 - продовжуємо цикл

335 ;відновлення регістрів

336 01BA 5A pop dx

337 01BB 59 pop cx

338 01BC 5B pop bx

339 01BD 58 pop ax

340 01BE C3 ret

341 01BF printInt endp

342 ;-------------------------------------------------------------------------------

343 01BF code ends

344 end main

Symbol Name Type Value

??DATE Text "01/25/20"

??FILENAME Text "lab61 "

??TIME Text "02:23:40"

??VERSION Number 0300

@CPU Text 0101H

@CURSEG Text CODE

@FILENAME Text LAB61

@WORDSIZE Text 2

A Word DATA:0009

BUFFER Byte DATA:001F

BUFFER2 Byte DATA:0028

COMPARISON\_NUMBER Word DATA:001D

CONVERTSTRTOINT Near CODE:012E

COUNTER Word DATA:0002

CRLF Byte DATA:012D

DIVISION Byte DATA:0004

LEN Word DATA:0000

MAIN Near CODE:0000

MCONDITION Near CODE:00CF

MDIVISIONNUMBER Near CODE:01A7

MELEMENTOK Near CODE:0092

MENDLOOP Near CODE:0068

MENDLOOP1 Near CODE:0030

MERRORCOUNTER Near CODE:0060

MERRORCOUNTER1 Near CODE:0028

MERRORVALUE Near CODE:0185

MEXIT Near CODE:0188

MINARRAY Near CODE:0074

MINCOUNTER Near CODE:0030

MINCOUNTER1 Near CODE:0005

MNEXT Near CODE:00E1

MNOSIGN Near CODE:0148

MOK Near CODE:0181

MPLUS Near CODE:017C

MPOSITIVE Near CODE:01A2

MPRINTARRAY Near CODE:00AF

MPRINTCHAR Near CODE:01B0

MPRINTSUM Near CODE:00E5

MSCANSTRING Near CODE:0151

MSGARRAY Byte DATA:00E3

MSGCOUNTER Byte DATA:00A5

MSGERROR Byte DATA:0106

MSGPAUSE Byte DATA:011B

MSGPRINT Byte DATA:0111

MSGPRINTSUM Byte DATA:0063

MSGPROMPT1 Byte DATA:00FB

MSGPROMPT2 Byte DATA:0100

MSGQUANTITY Byte DATA:0031

OUTPROMPT Near CODE:011C

OUTSTRING Near CODE:0117

PRINTINT Near CODE:018F

QUANTITY Word DATA:0007

SUM Word DATA:0005

Groups & Segments Bit Size Align Combine Class

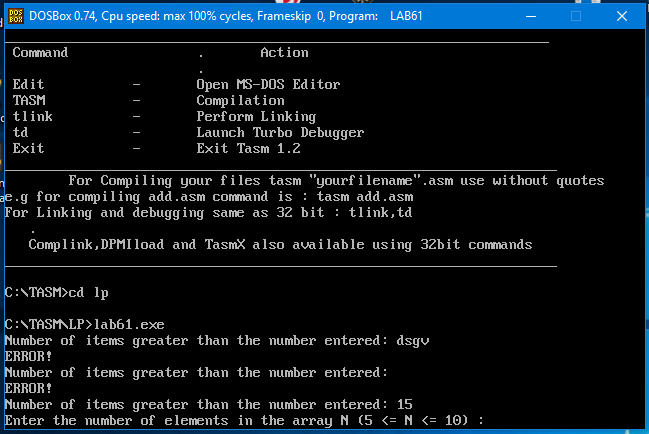
CODE 16 01BF Para Public CODE

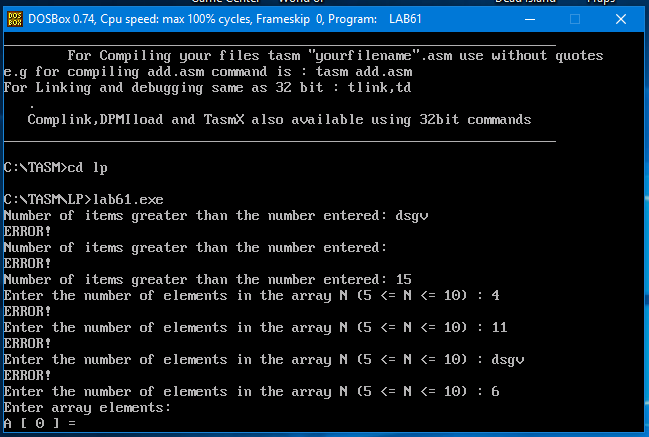
DATA 16 0130 Para Public DATA

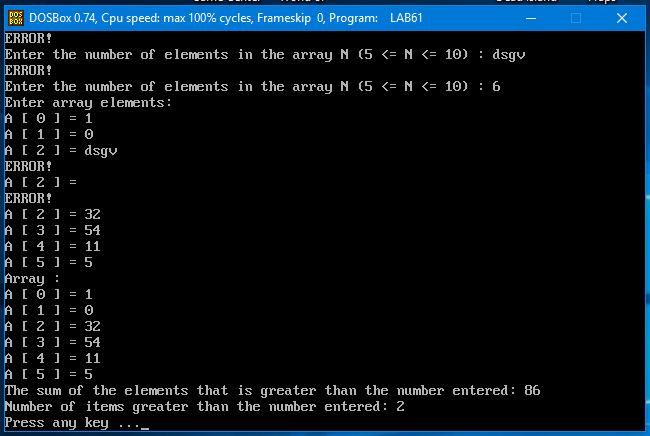
STK 16 0100 Para Stack

4. Результати виконання програми

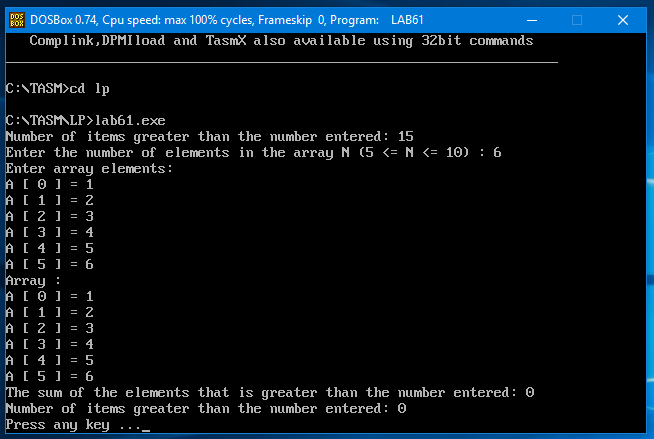
1. Тести для перевірки на правильність вводу даних

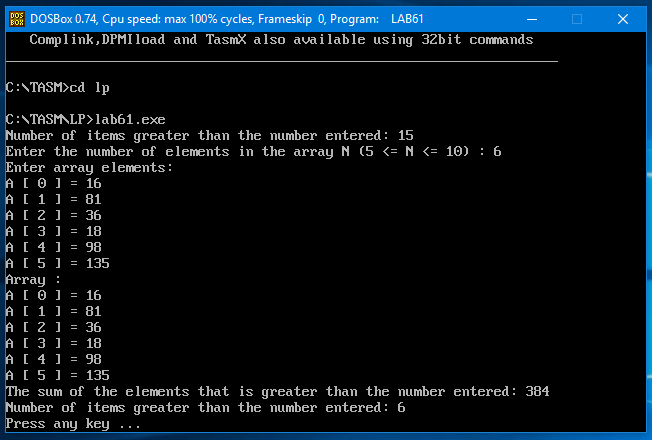






2. Тести для результату





Висновок: у даній лабораторній роботі я ознайомився організацією введення та виведення інформації в консолі та обробка масивів чисел в мові Ассемблера. Була створена блок-схема та розроблена програма до індивідуального завдання. Розробленні все варіанти тестів і всі вони були пройдені вдало.