**Lab 09: implementation of Arrays and Strings in Assembly.**

**OBJECT:**

Arrays and Strings in Assembly.

**THEORY:**

An array is a collection of elements (usually of same type). A single variable contains the array elements, each with its own index in the array.

String is a special type of array that is specially used for storing a series of characters. The string constants are defined within single starting and ending quote.

**Pre-Lab Task**

**Program:**

The program defines array and string in the data segment. It then uses these array variables in the program by using their starting offset address.

**Note**: Strings in C++ end with ‘\0’ character. Similarly strings in assembly end with a ‘$’ character.

. MODEL SMALL

. STACK 100H

.DATA

NUMBERS DB 0, 2, 4, 6, 8

NUM\_LENGTH = $ - OFFSET NUMBERS

UNKNOWN DB 5 DUP (0)

I DB 0

GREETING DB 'ARRAY ELEMENTS ARE', '$'

NEWLINE DB 0AH, 0DH, '$'

. CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV AH, 9 ; TO OUTPUT STRING

; MOV DX, OFFSET GREETING

LEA DX, GREETING

INT 21H

; PRINT NEW LINE STRING, AH ALREADY CONTAINS STRING OUTPUT FUNCTION 9

MOV DX, OFFSET NEWLINE

INT 21H

MOV AH, 2

MOV CL, 0

LEA SI, NUMBERS ; MOV THE ARRAY TO ONE OF THE INDEXING REGISTERS

LBL:

MOV DX, [SI]

ADD DL, 48 ; CONVERT THE NUMBER TO ITS ASCII VALUE

INT 21H ; OUTPUT THE NUMBER

INC SI

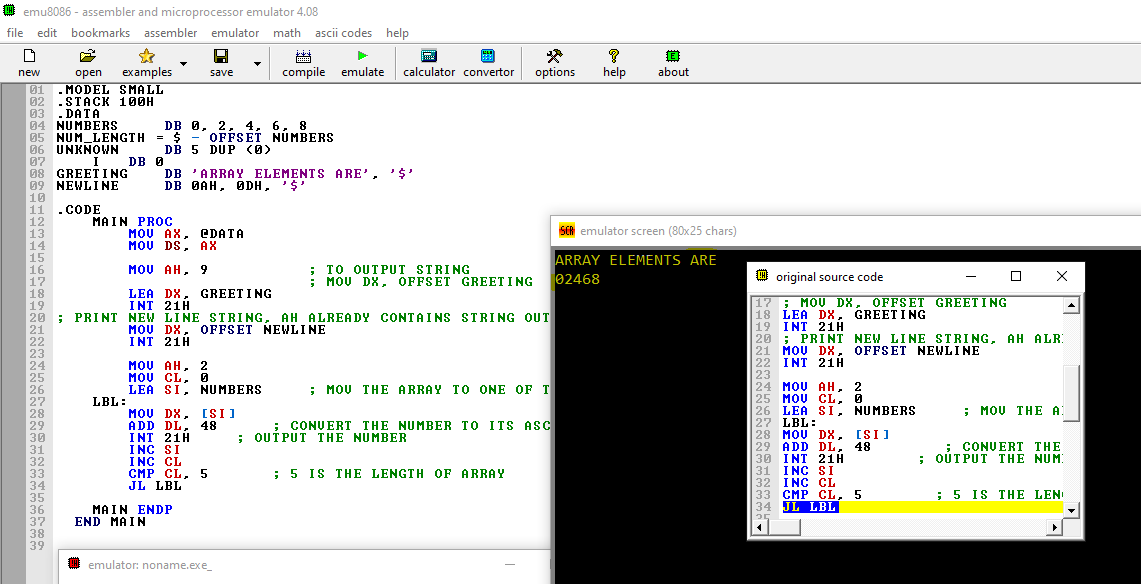
INC CL

CMP CL, 5 ; 5 IS THE LENGTH OF ARRAY

JL LBL

MAIN ENDP

END MAIN



LEA or OFFSET keywords are used for arrays because they load the offset address of the first element of the array. Array elements in the memory are arranged in continuous memory locations. So, if system gets the address of first element, it can access rest of the elements. The interrupt function number 9 is used to output string.

There is another more intuitive way of accessing the array elements.

The program below shows an example.

. MODEL SMALL

. STACK 100H

.DATA

ARR DB 1,2,3,4

ARR\_LEN = $ - OFFSET ARR ; ARR\_LEN WILL CONTAIN THE LENGTH OF THE ARRAY ELEMENTS

. CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV AH, 2

MOV BX, 0

NEXT:

MOV DL, ARR[BX]

ADD DL, 48

INT 21H

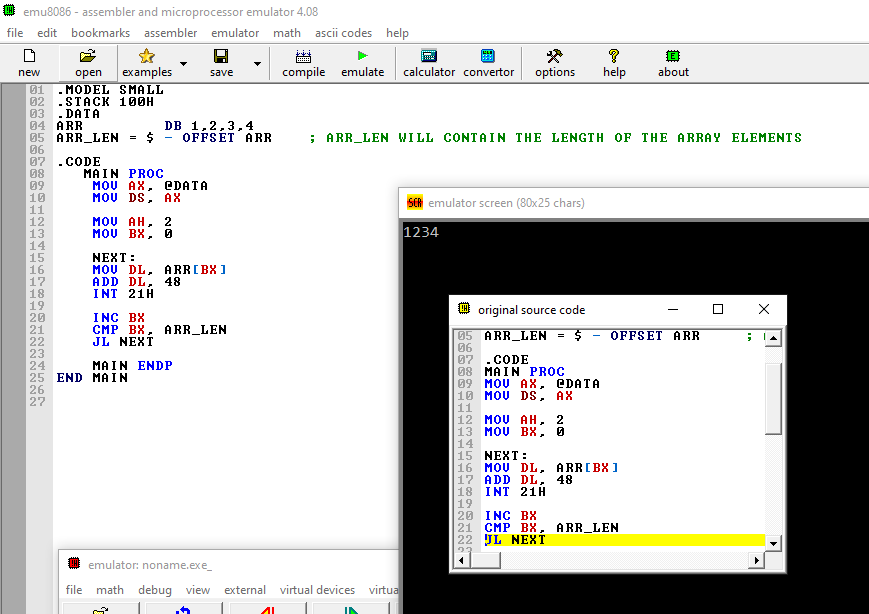
INC BX

CMP BX, ARR\_LEN

JL NEXT

MAIN ENDP

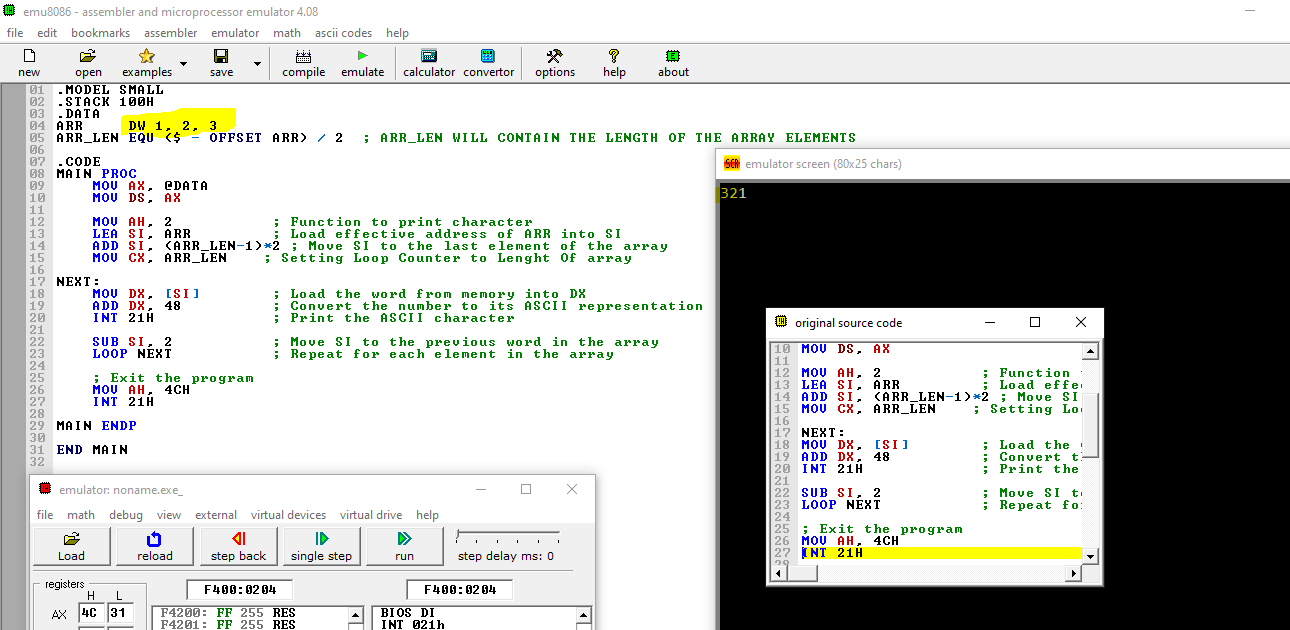
END MAIN



**In-Lab Task:**

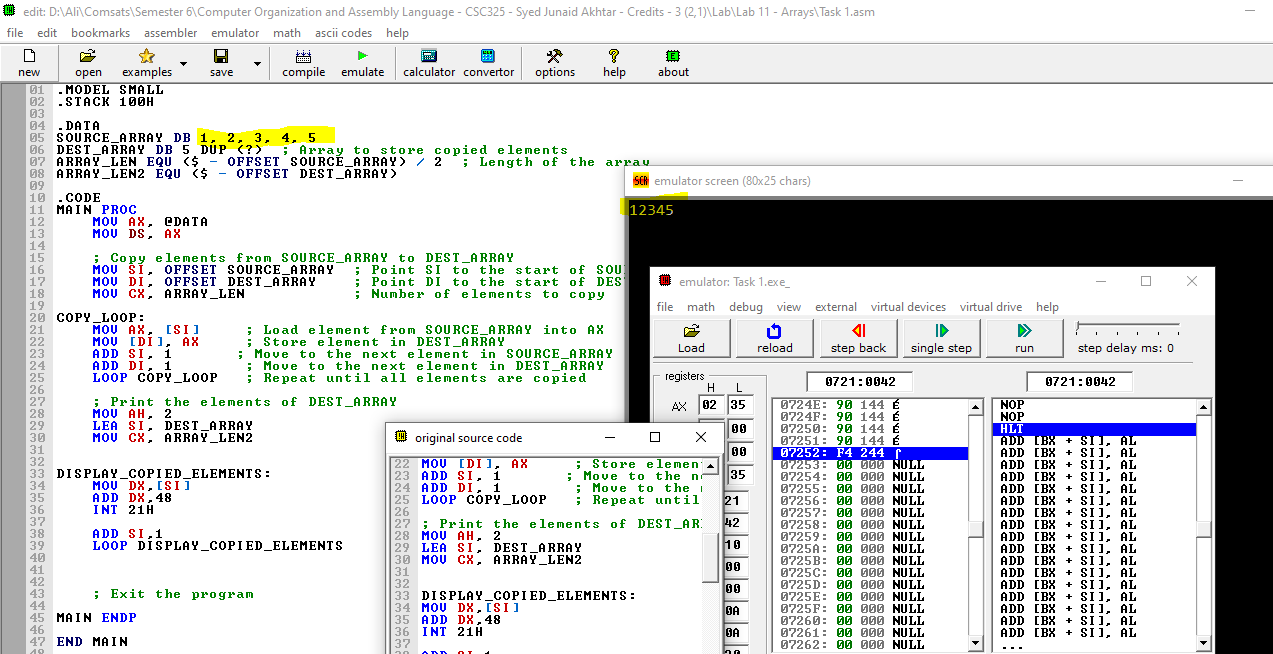
**Task1**

Define an array of words (with DW) and print its elements in reverse order.



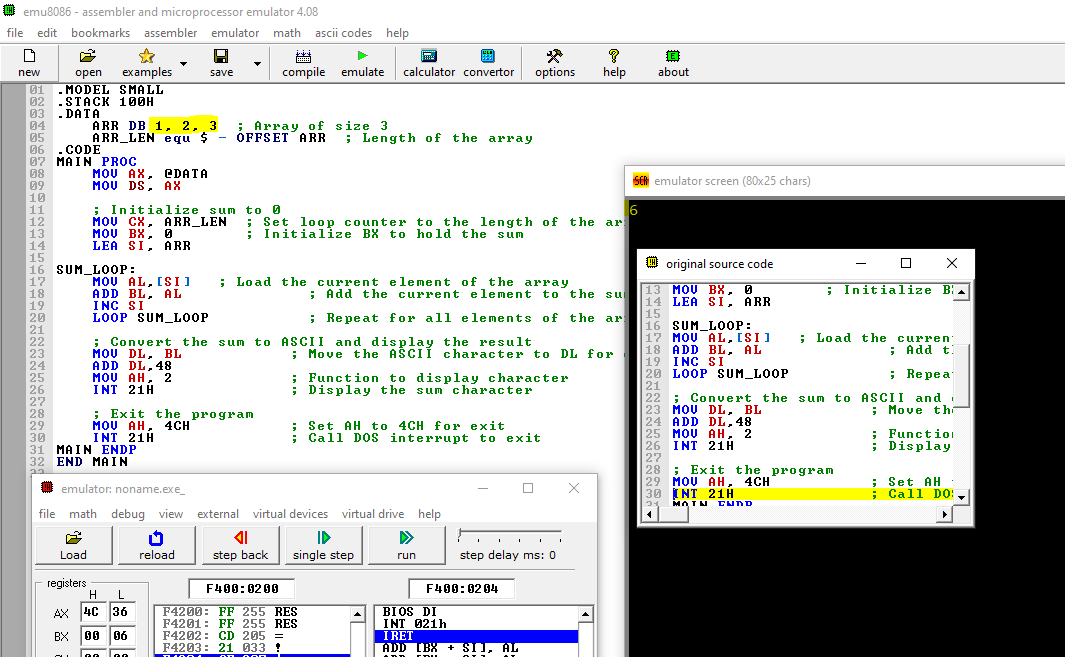
**Task 2**

Write a program to one array to another.



**Task 3**

Write a program that finds sum of elements of an array of size three.



**Post-Lab Assignment:**

**REVIEW QUESTIONS**

1. How are arrays and strings arranged in the memory?

Arrays and strings are arranged in memory as contiguous blocks of bytes, with each element occupying consecutive memory locations.

1. What will happen if a string is not terminated with a ‘$’ character?

It can result in Undefined Behavior, Memory Overrun or Incorrect Input/Output.

1. What will be the output of printing string if in the middle of the string comes the character ‘$’?

String will stop printing when $ is encountered.

1. What will be the effect on output if we print an increment array index further off its original size (total number of elements)?

It can result in Error, Undefined output, Garbage Values being Printed.

1. What is the purpose of using **offset** or **lea** for loading an array or string in register? What will happen if array/string name is directly used without using offset?

It will result in Error.

**Critical Analysis /Conclusion**

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Performance (15Marks)** | | **Viva**  **(5 Marks)** | **Total/20** |
| **Pre-Lab Exercise** | **/3** |  |  |
| **Performance** | **/4** |  |  |
| **Results** | **/3** |
| **Critical Analysis** | **/2** |
| **Post Lab Exercise** | **/3** |
| **Comments** | | | |