Lab Manual for Computer Organization and Assembly Language

Lab-9

Arrays

Table of Contents

1. Introduction

To access an array in assembly language, we use a *pointer*. A pointer is simply a register or variable that contains a memory address. The value in the pointer is computed by adding the base address of the array and the offset of the desired element. The data section can also be used for defining a one-dimensional array. Let us define a one-dimensional array of numbers.

2. Objective

- To know more about Assembly language, such as how to manipulate with arrays in assembly language.

3. Concept Map

This section provides you the overview of the concepts that will be discussed and implemented in this lab.

3.1 Arrays

We can define an array in data section using the following method

```
ArrayNum Db 34, 45, 56, 67, 75, 89
```

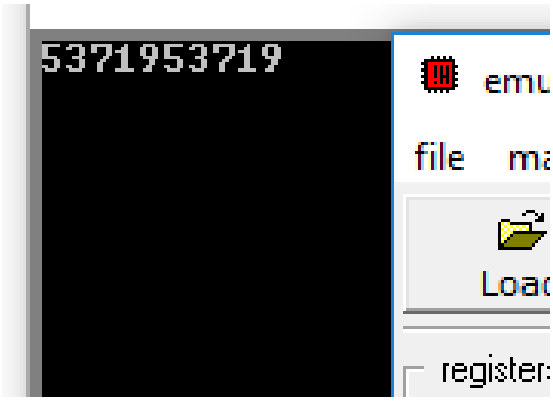
3.2 Example

The following program inputs values from user in an array and then prints on the console.

```
.model small
.data
    num1 db ?


.code
    mov ax,@data
    mov ds,ax

;Taking input in an array

    mov cx,5
    mov si,0
    l2:

        mov ah,01
        int 21h
        mov [num1+si],al
        sub [num1+si],48
        inc si

    loop l2

;Show the values stored in array

    mov cx,5
    mov si,0

    l1:
        mov bl,[num1+si]
        inc si

        mov dl,bl
        add dl,48
        mov ah,02
        int 21h

    loop l1
```

The output of the program is shown below.

Output:

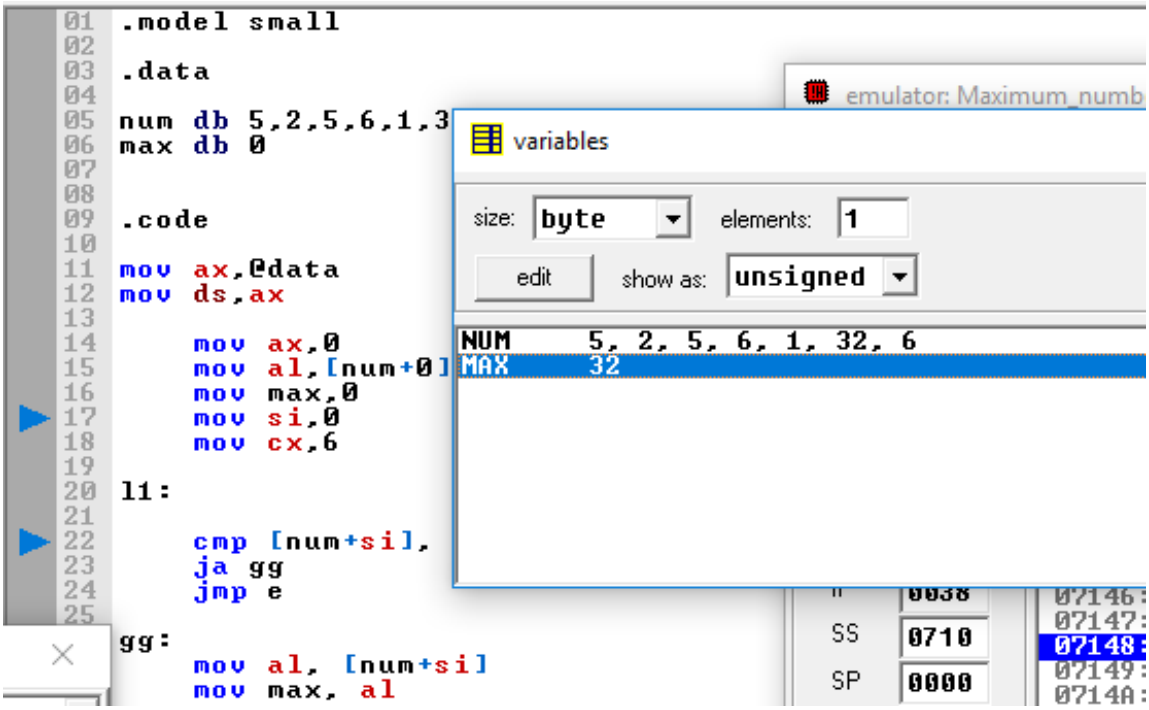


4. Walkthrough Task

Write an assembly program that can find the maximum number in an array.

```
01  .model small
02
03  .data
04
05  num db 5,2,5,6,1,32,6
06  max db 0
07
08
09  .code
10
11  mov ax,@data
12  mov ds,ax
13
14      mov ax,0
15      mov al,[num+0]
16      mov max,0
17      mov si,0
18      mov cx,6
19
20  l1:
21
22      cmp [num+si], al
23      ja gg
24      jmp e
25
26  gg:
27      mov al, [num+si]
28      mov max, al
29
30  e:
31
32      inc si
33      loop l1
34
```

When the above code is compiled and executed, it produces the following **Output.**



5. Procedure& Tools

In this section you will study how to setup and MASM Assembler.

5.1 Tools

- Download emu 8086 from (http://www.emu8086.com/files/emu8086v408r11.zip)
- Just extract the emu8086.15.zip on C
- Install emu8086

6. Practice Tasks

This section will provide more practice exercises which you need to finish during the lab. You need to finish the tasks in the required time. When you finish them, put these tasks in the following folder:
\\fs\assignments$\

6.1 Practice Task 1 [Expected time = 15mins]

Write an assembly code that can input ten numbers from user and find the maximum and minimum number in the array.

6.2 Practice Task 2 [Expected time = 15mins]

Write an assembly code that can input ten numbers from user and count the even and odd numbers in array. And show the maximum count

6.3 Practice Task 3 [Expected time = 15mins]

Write an assembly code which can take input from user and then store it in array and after storing data program should display array data in reverse order.

6.4 Out comes

After completing this lab, student will be able to setup emu 8086. He/ She will also be able to compile and run basic Assembly programs.

7. Evaluation Task (Unseen) [Expected time = 30mins for tasks]

The lab instructor will give you unseen task depending upon the progress of the class.

8. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 3: Evaluation of the Lab

Sr. No.	Task No	Description	Marks
1	4	Problem Modeling	20
2	6	Procedures and Tools	10
3	7	Practice tasks and Testing	35
4	8	Evaluation Tasks (Unseen)	20
5		Comments	5
6		Good Programming Practices	10

9. Further Reading

This section provides the references to further polish your skills.

9.1 Slides

The slides and reading material can be accessed from the folder of the class instructor available at \\fs\lectures$