Lab-8

Arrays in Assembly Language – CS2523

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
```

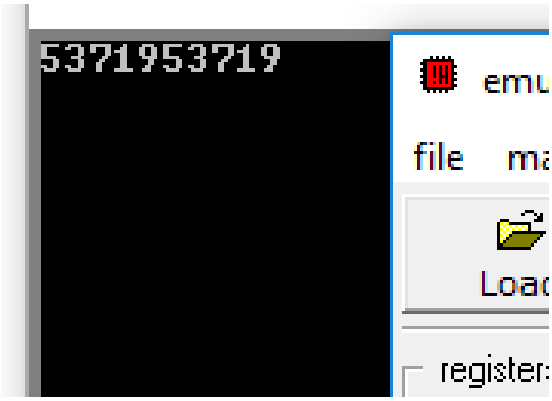
3.2. Example

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

```
01
02
03  .model small
04  .data
05      num1 db ?
06
07
08  .code
09      mov ax,@data
10      mov ds,ax
11
12  ;Taking input in an array
13
14      mov cx,5
15      mov si,0
16      l2:
17
18          mov ah,01
19          int 21h
20          mov [num1+si],al
21          sub [num1+si],48
22          inc si
23
24      loop l2
25
26  ;Show the values stored in array
27
28      mov cx,5
29      mov si,0
30
31      l1:
32          mov bl,[num1+si]
33          inc si
34
35          mov dl,bl
36          add dl,48
37          mov ah,02
38          int 21h
39
40      loop l1
41
42
```

The output of the program is shown below.

Output:

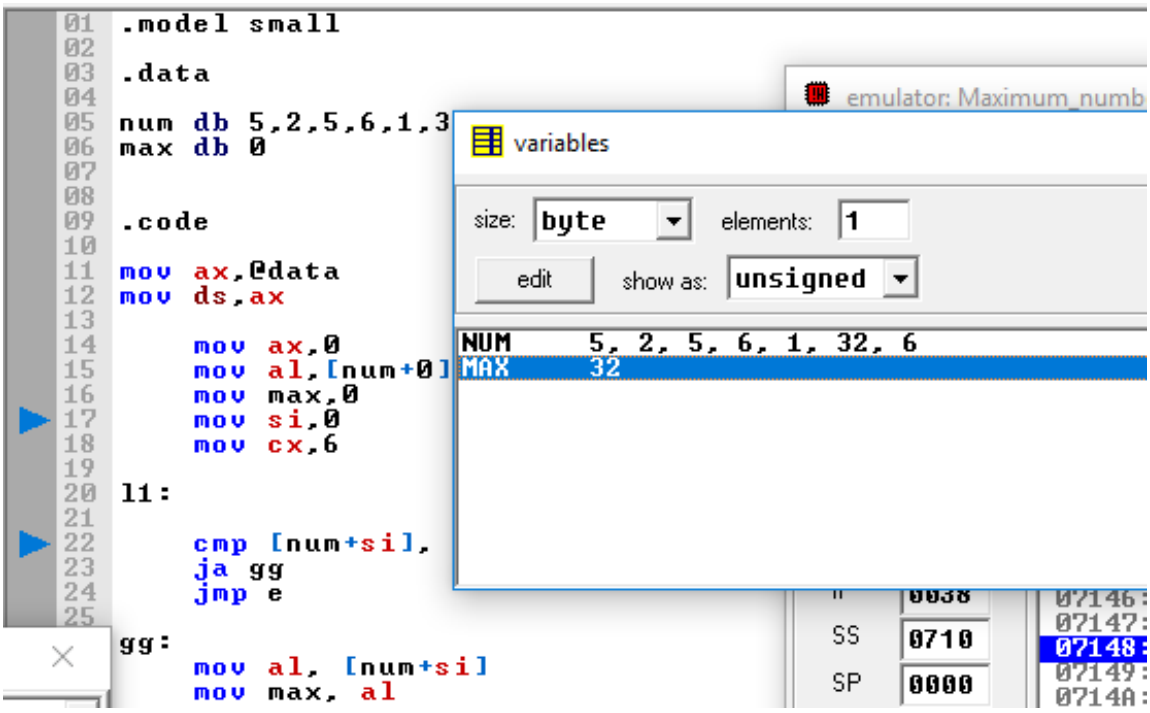


1. Walkthrough Task

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

```
.model small

.data

num db 5,2,5,6,1,32,6
max db 0


.code

mov ax,@data
mov ds,ax

    mov ax,0
    mov al,[num+0]
    mov max,0
    mov si,0
    mov cx,6

l1:

    cmp [num+si], al
    ja gg
    jmp e

gg:
    mov al, [num+si]
    mov max, al

e:

    inc si
    loop l1
```

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



4. Procedure& Tools

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

4.1. Tools

- Open emu8086.
- Make new project.
- Enter your code.
- Click on Emulate.
- Click on Run

5. 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
following folder:
\\fs\assignments$

5.1. 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.

5.2. 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

5.3. 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.

5.4. Practice Task 4 [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 in descending order.

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.

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

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

7. 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: Evaluation of the Lab

Sr. No.	Task No	Description	Marks
1		Understanding of Problem	20
2		Program Logic	20
3		Program Implementation	20
4		Program Correctness	10
5		Use of Tool	10
6		Viva	20