

**DIGITAL ASSIGNMENT 4**

**MICROPROCESSOR AND INTERFACING(D1)[MRS SHOBHA REKH]**



**October 19, 2022**

**ANIRUDH VADERA**

**20BCE2940**

**QUESTION 1:**

**Find the smallest and largest number in an array. The size of array can be stored in a memory location. Store the smallest/largest number in a memory location**

**Algorithm:**

1. Take n as input
2. Initialize first element as the smallest or the largest element.
3. Run a loop from i = n to 1. For each iteration, check if the current element is larger ( or smaller ) than the current largest(or smallest) element that we are storing and if the current element is larger(or smaller) then update our current largest(or smallest) element to that element
4. Simply output the current largest (or smallest) element at the end of for loop.

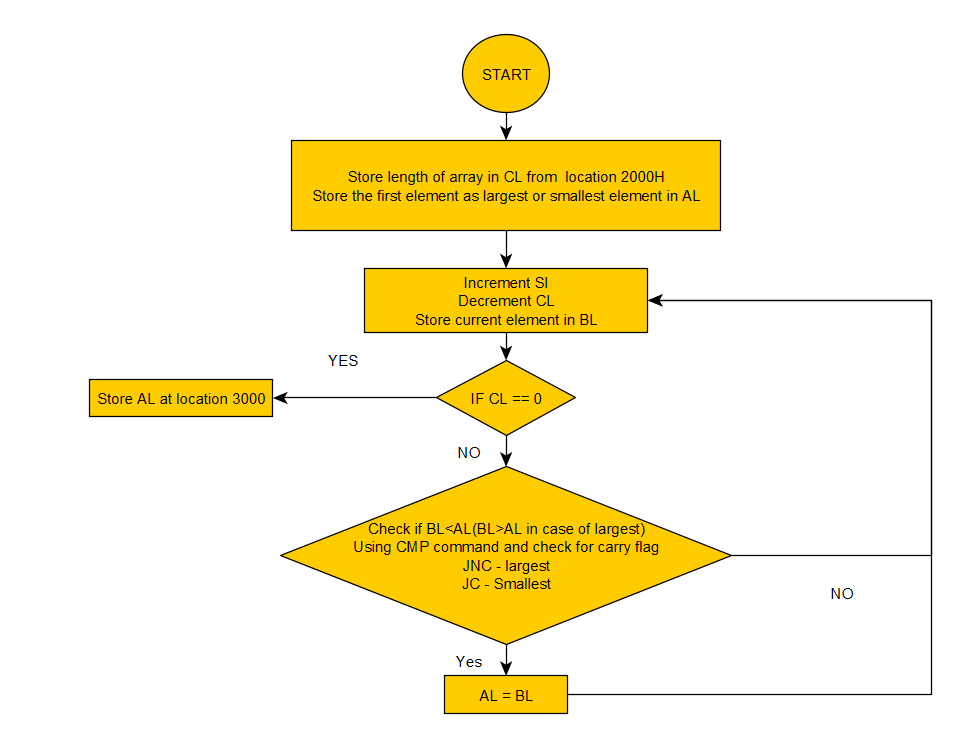
**In Assembly Language:**

Start

1. Load the number of elements in the CL register stored at location 2000H
2. The array starts from location 2500H
3. Load the first element as the largest(Smallest) Element in AL
4. Now move the current element to BL
5. Compare the BL and AL register (BL has the current element and AL has the supposedly the largest / smallest element)
6. If the carry flag has 1 – BL < AL (if Smallest, then we set AL = BL) If largest then we ignore (if carry flag has 0 we set AL = BL in largest case)
7. We then decrement CL and check if its equal to 0. If it is equal, then we end our loop and store the result in AL to location 3000
8. Else we increment SI and continue the same from step 4 to 7.

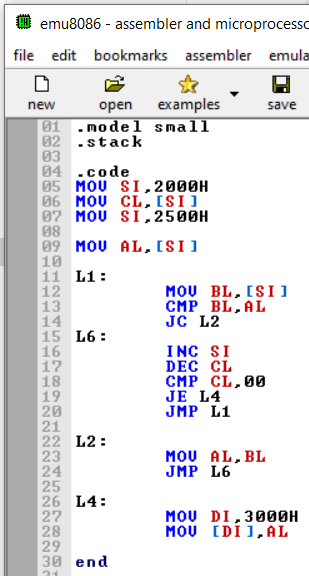
End

**FlowChart:**

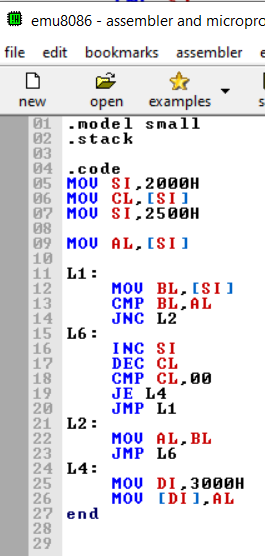


**Code Screenshot:**

**Smallest**



**Largest**

****

**Assembly Language Code:**

**Smallest**

.model small

.stack

.code

MOV SI,2000H

MOV CL,[SI]

MOV SI,2500H

MOV AL,[SI]

L1:

MOV BL,[SI]

CMP BL,AL

JC L2

L6:

INC SI

DEC CL

CMP CL,00

JE L4

JMP L1

L2:

MOV AL,BL

JMP L6

L4:

MOV DI,3000H

MOV [DI],AL

end

**Largest:**

.model small

.stack

.code

MOV SI,2000H

MOV CL,[SI]

MOV SI,2500H

MOV AL,[SI]

L1:

MOV BL,[SI]

CMP BL,AL

JNC L2

L6:

INC SI

DEC CL

CMP CL,00

JE L4

JMP L1

L2:

MOV AL,BL

JMP L6

L4:

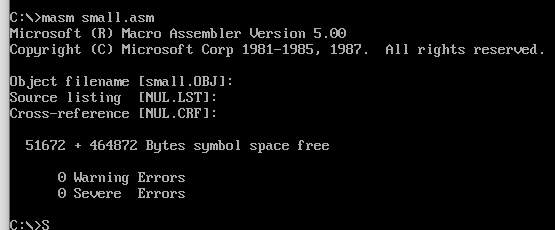
MOV DI,3000H

MOV [DI],AL

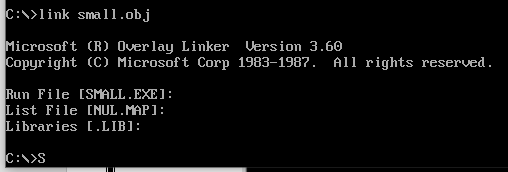
end

**Execution Proof:**

**Command: masm small.asm**



**Command: link small.obj**



**Command: debug small.exe**



**Given Input at location 2000H: The length of the array**

**The array elements start from location 2500H**

**Output at Location 3000H (Smallest or Largest Element)**

**Giving Inputs:**

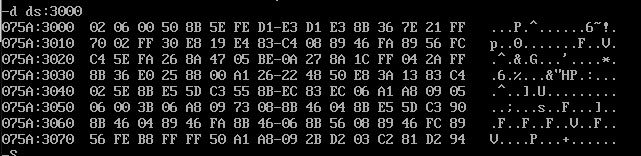
Given Number at **2000H**: **07**

Given array from **2500H**: **04 05 06 AF 89 02 78**

(Expected Output **02H** at **3000H for smallest and AFH at 3000H for largest**)



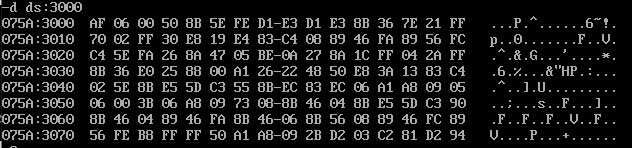
**The Output is(Smallest):**





Hence **02H** is stored at **location 3000** hence **02 is the smallest number in the given array.**

**The Output is(Largest):**





Hence **AFH** is stored at **location 3000** hence **AF is the largest number in the given array.**

**QUESTION 2:**

**Arrange the given array in ascending/Descending order. The size of array can be taken from a memory locationAlgorithm:**

We assume **list** is an array of **n** elements. We further assume that **swap** function swaps the values of the given array elements.

begin BubbleSort(list)

for all elements of list

if list[i] > list[i+1]

swap(list[i], list[i+1])

end if

end for

return list

end BubbleSort

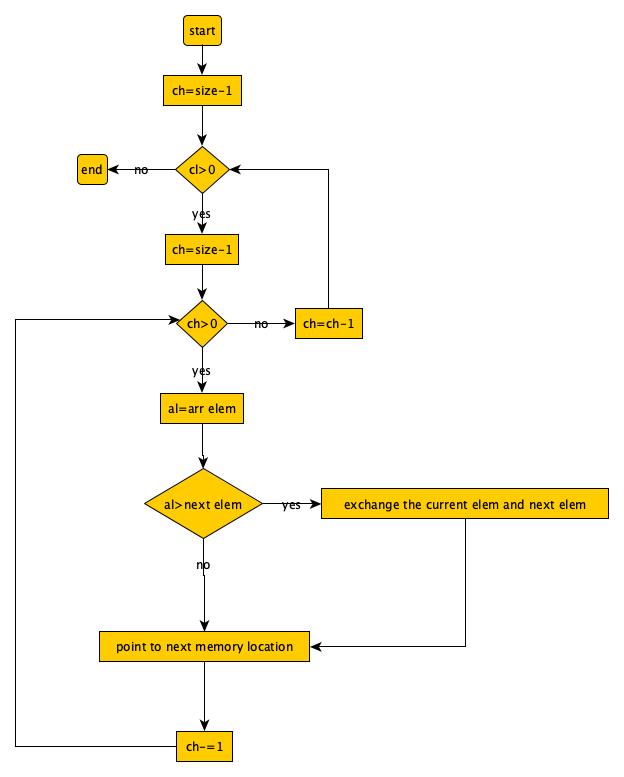
**In Assembly Language:**

Start

1. Load the number of elements in the CH register stored at location 2000H
2. Now load the number of iterations to be performed on each element in order to sort the array into CL which will be same as in CH
3. Decrement CH and CL at every step as we have to run the loop one time less.
4. Now take the element at location SI (say 2nd element from location 2501) into AL
5. Compare the element previous to it SI – 1 (i.e the first element) and the current element
6. Now we check the value of carry flag
7. IF Carry flag = 1, For ascending order follow step 8 for descending order ignore
8. IF Carry flag = 0, For descending order follow step 9 for ascending order ignore
9. For Ascending Order: If the 2nd element is smaller than the first element we swap the two elements in hand ie at location SI and at location SI-1
10. We then increment the SI in order to check the next 2 elements
11. We repeat this (step 10 and again from step 4 to 9) until CL becomes 0
12. Now we repeat the steps from 4 to 11 until CH becomes 0
13. We end the loop when CH becomes 0

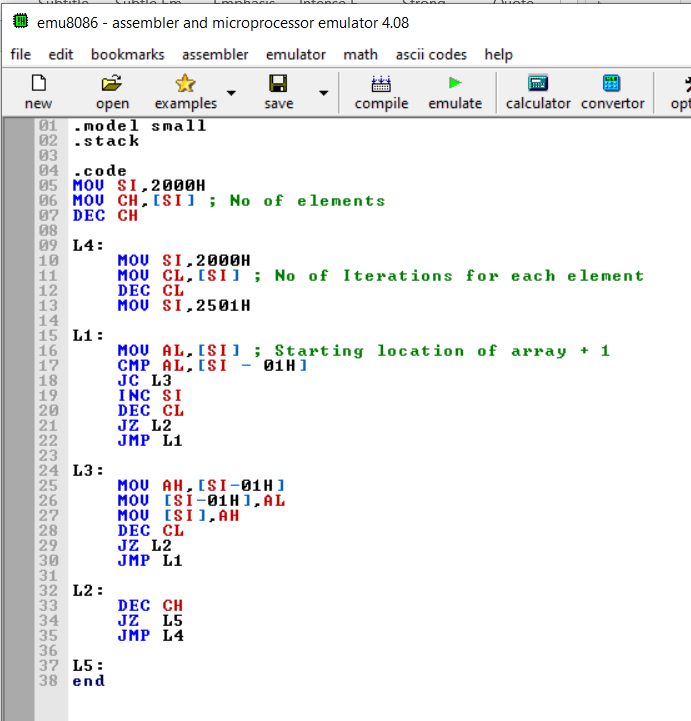
End

**FlowChart:**

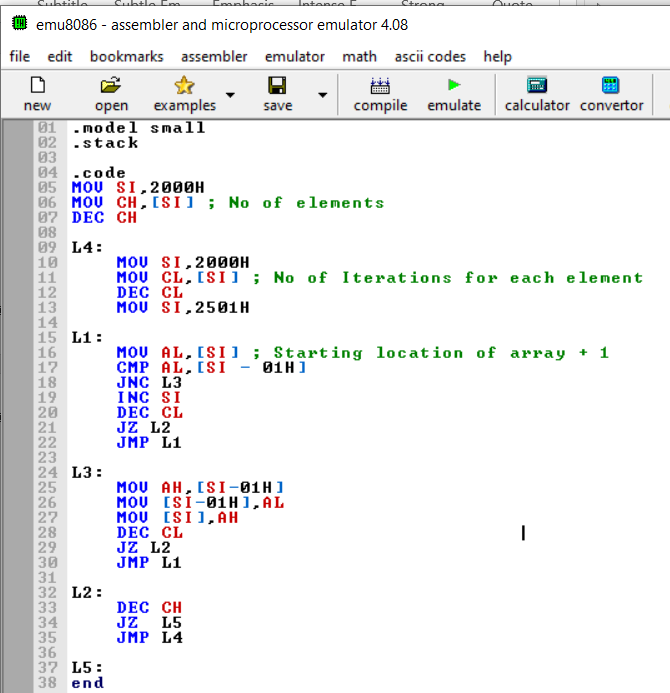


**Code Screenshot:**

**Ascending**



**Descending**

****

**Assembly Language Code:**

**Ascending**

.model small

.stack

.code

MOV SI,2000H

MOV CH,[SI] ; No of elements

DEC CH

L4:

MOV SI,2000H

MOV CL,[SI] ; No of Iterations for each element

DEC CL

MOV SI,2501H

L1:

MOV AL,[SI] ; Starting location of array + 1

CMP AL,[SI - 01H]

JC L3

INC SI

DEC CL

JZ L2

JMP L1

L3:

MOV AH,[SI-01H]

MOV [SI-01H],AL

MOV [SI],AH

DEC CL

JZ L2

JMP L1

L2:

DEC CH

JZ L5

JMP L4

L5:

End

**Descending**

.model small

.stack

.code

MOV SI,2000H

MOV CH,[SI] ; No of elements

DEC CH

L4:

MOV SI,2000H

MOV CL,[SI] ; No of Iterations for each element

DEC CL

MOV SI,2501H

L1:

MOV AL,[SI] ; Starting location of array + 1

CMP AL,[SI - 01H]

JNC L3

INC SI

DEC CL

JZ L2

JMP L1

L3:

MOV AH,[SI-01H]

MOV [SI-01H],AL

MOV [SI],AH

DEC CL

JZ L2

JMP L1

L2:

DEC CH

JZ L5

JMP L4

L5:

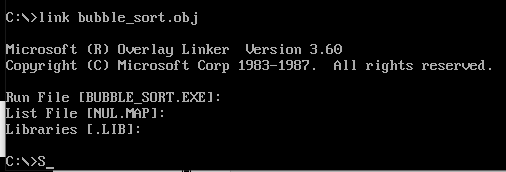
end

**Execution Proof:**

**Command: masm bubble\_sort.asm**



**Command: link bubble\_sort.obj**



**Command: debug bubble\_sort.exe**



**Given Input at location 2000H: The length of the array**

**The array elements start from location 2500H**

**Output from Location 2500H (Until the location the array is stored)**

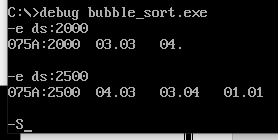
**Giving Inputs:**

Given Number at **2000H**: **03**

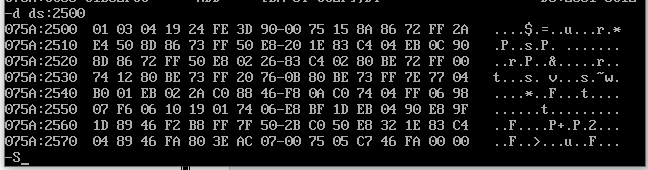
Given array from **2500H**: **03 04 01**

(Expected Output **01 03 04** from **2500H for ascending**)

(Expected Output **04 03 01** from **2500H for descending**)



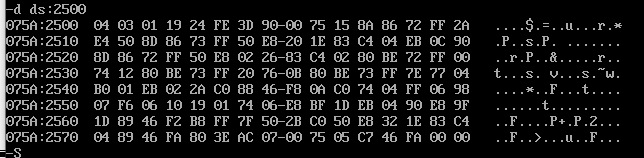
**The Output is(Ascending):**





Hence **the array is successfully sorted in ascending order.**

**The Output is(Descending):**

****



Hence **the array is successfully sorted in descending order.**