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.

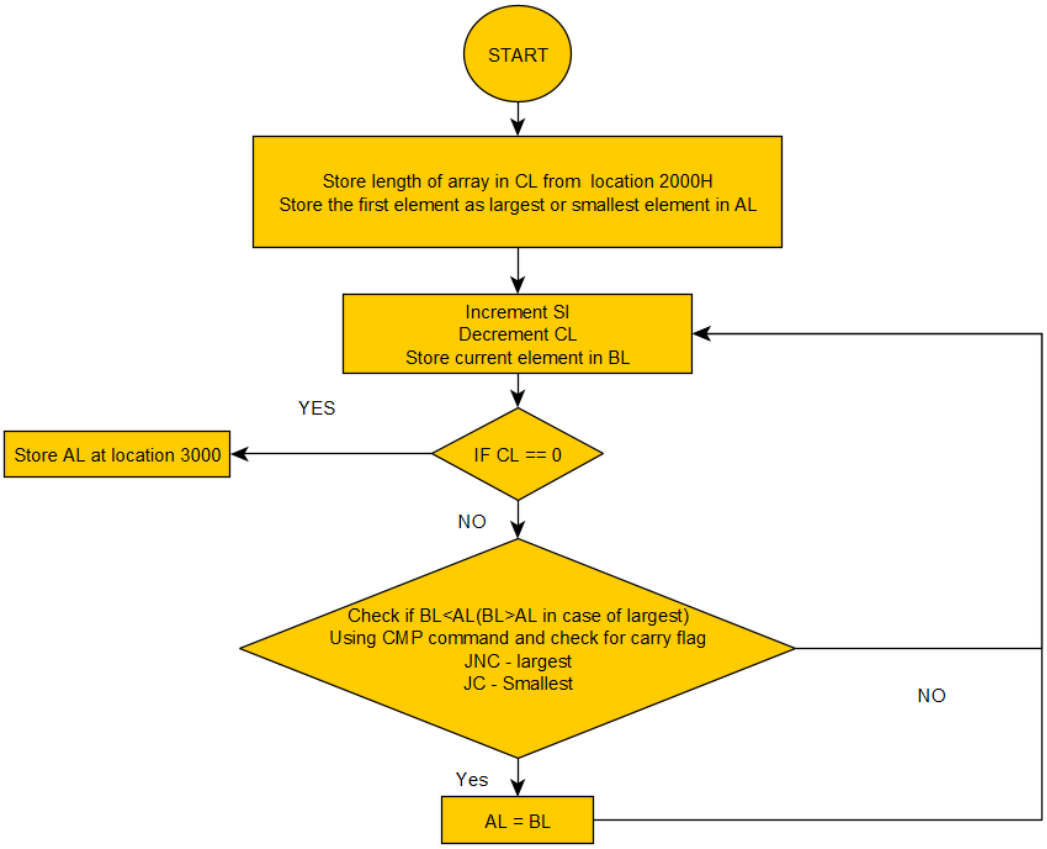
In Assembly Language:

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.

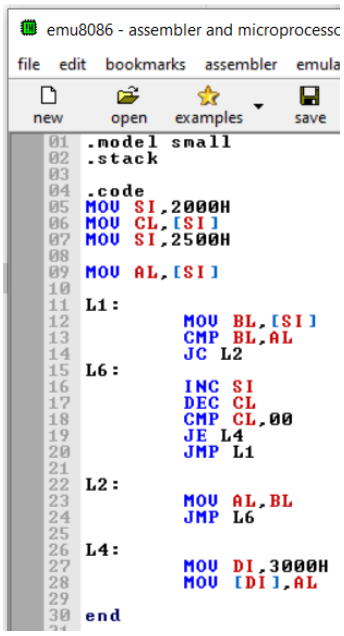
End

FlowChart:



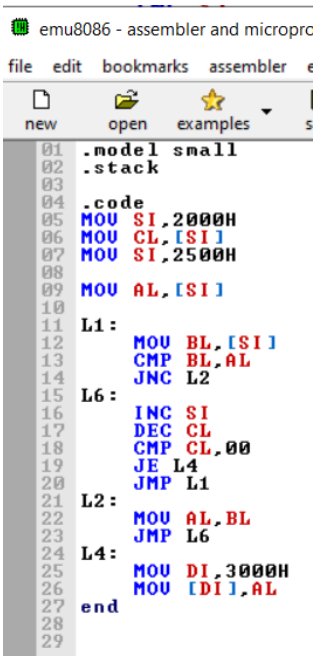
Code Screenshot:

Smallest




```
.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



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
```

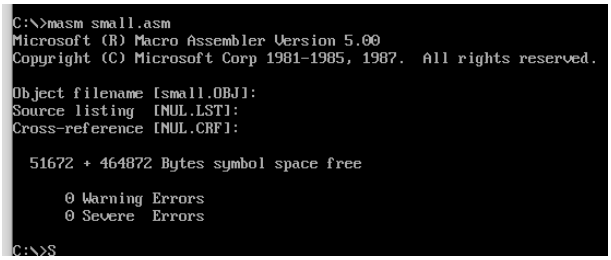
Largest:

```
.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
```

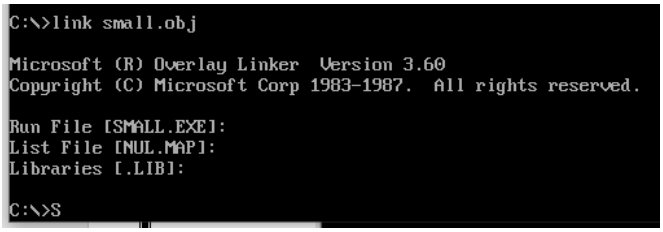
Execution Proof:

Command: masm small.asm



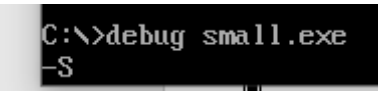
```
C:\>masm small.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [small.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51672 + 464872 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>S
```

Command: link small.obj



```
C:\>link small.obj

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [SMALL.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\>S
```

Command: debug small.exe



```
C:\>debug small.exe
-S
```

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**)

```
C:\>debug small.exe
-e ds:2000
075A:2000  7F.07

-e ds:2500
075A:2500  FF.04   A1.05   10.06   19.AF   24.89   FE.02   3D.78
```

The Output is(Smallest):

```
-d ds:3000
075A:3000  02 06 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   ...P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
```

```
-d ds:3000
075A:3000  02
```

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

The Output is(Largest):

```
-d ds:3000
075A:3000  AF 06 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   ...P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
```

```
-d ds:3000
075A:3000  AF
```

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 location

Algorithm:

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
```

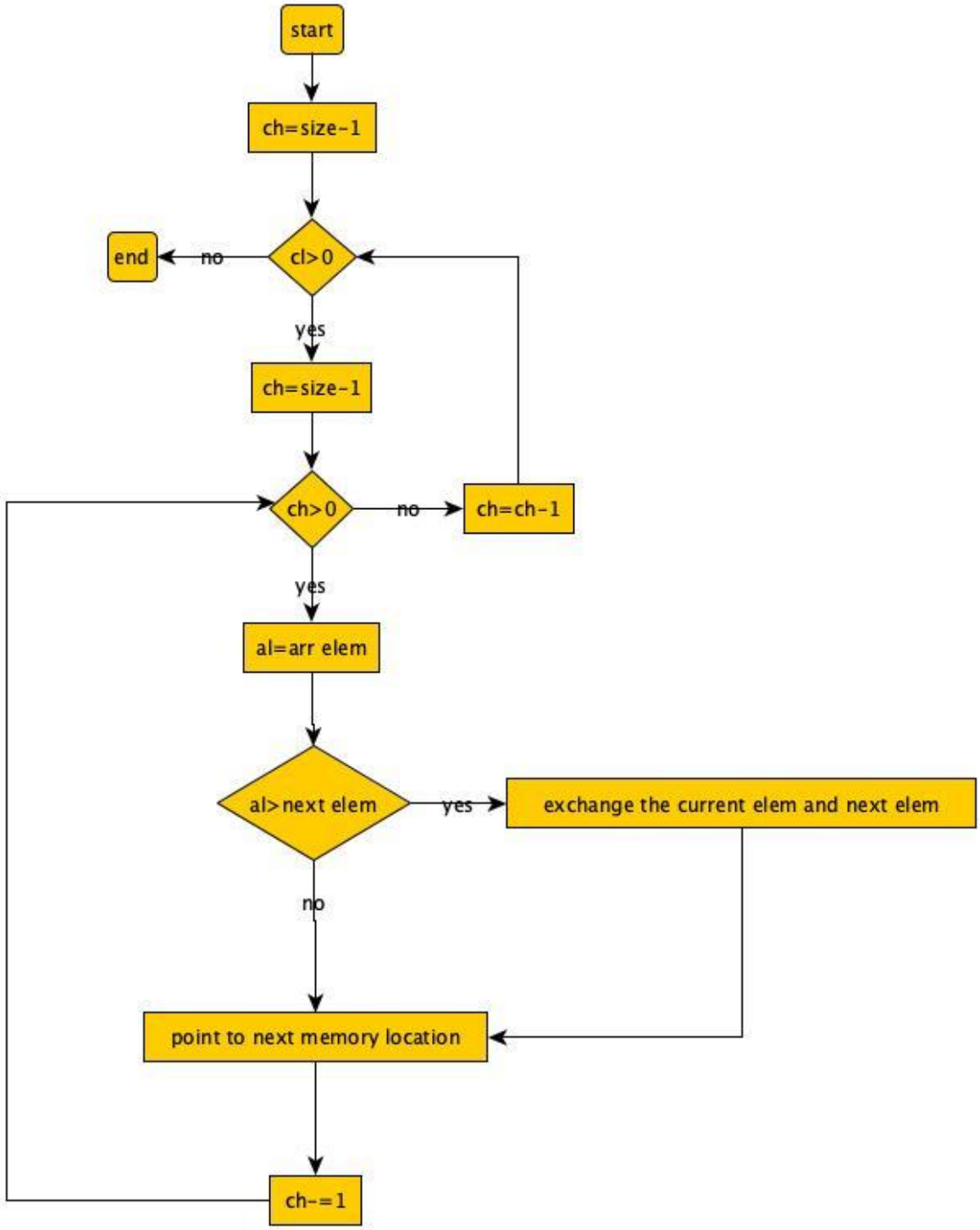
In Assembly Language:

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:


start
ch=size-1
cl>0
no
end
yes
ch=size-1
ch>0
no
ch=ch-1
yes
al=arr elem
al>next elem
yes
exchange the current elem and next elem
no
point to next memory location
ch-=1

Code Screenshot:

Ascending

emu8086 - assembler and microprocessor emulator 4.08

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor opt

```
.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

emu8086 - assembler and microprocessor emulator 4.08

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor

```
.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
```

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

```
C:\>masm bubble_sort.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [bubble_sort.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51642 + 464902 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>S
```

Command: link bubble_sort.obj

```
C:\>link bubble_sort.obj

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [BUBBLE_SORT.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\>S_
```

Command: debug bubble_sort.exe

```
C:\>debug bubble_sort.exe
-S
```

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**)

```
C:\>debug bubble_sort.exe
-e ds:2000
075A:2000  03.03   04.

-e ds:2500
075A:2500  04.03   03.04   01.01

-S_
```

The Output is(Ascending):

```
-d ds:2500
075A:2500  01 03 04 19 24 FE 3D 90-00 75 15 8A 86 72 FF 2A   ....$.=..u...r.*
075A:2510  E4 50 8D 86 73 FF 50 E8-20 1E 83 C4 04 EB 0C 90   .P..s.P. .......
075A:2520  8D 86 72 FF 50 E8 02 26-83 C4 02 80 BE 72 FF 00   ..r.P..&.....r..
075A:2530  74 12 80 BE 73 FF 20 76-0B 80 BE 73 FF 7E 77 04   t...s. v...s.~w.
075A:2540  B0 01 EB 02 2A C0 88 46-F8 0A C0 74 04 FF 06 98   ....*..F...t....
075A:2550  07 F6 06 10 19 01 74 06-E8 BF 1D EB 04 90 E8 9F   ......t.........
075A:2560  1D 89 46 F2 B8 FF 7F 50-2B C0 50 E8 32 1E 83 C4   ..F....P+.P.2...
075A:2570  04 89 46 FA 80 3E AC 07-00 75 05 C7 46 FA 00 00   ..F..>...u..F...
-S_
```

```
-d ds:2500
075A:2500  01 03 04
```

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

The Output is(Descending):

```
-d ds:2500
075A:2500  04 03 01 19 24 FE 3D 90-00 75 15 8A 86 72 FF 2A   ....$.=..u...r.*
075A:2510  E4 50 8D 86 73 FF 50 E8-20 1E 83 C4 04 EB 0C 90   .P..s.P. .......
075A:2520  8D 86 72 FF 50 E8 02 26-83 C4 02 80 BE 72 FF 00   ..r.P..&.....r..
075A:2530  74 12 80 BE 73 FF 20 76-0B 80 BE 73 FF 7E 77 04   t...s. v...s.~w.
075A:2540  B0 01 EB 02 2A C0 88 46-F8 0A C0 74 04 FF 06 98   ....*..F...t....
075A:2550  07 F6 06 10 19 01 74 06-E8 BF 1D EB 04 90 E8 9F   ......t.........
075A:2560  1D 89 46 F2 B8 FF 7F 50-2B C0 50 E8 32 1E 83 C4   ..F....P+.P.2...
075A:2570  04 89 46 FA 80 3E AC 07-00 75 05 C7 46 FA 00 00   ..F..>...u..F...
-S
```

```
-d ds:2500
075A:2500  04 03 01
```

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