Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Experiment 3:
Sorting the elements of an Array in Ascending and Descending order

Programme	:	**BTech. CSE Core**	Semester	:	**Win 2021-22**
Course	:	**Microprocessor and Interfacing**	Code	:	**CSE2006**
Faculty	:	**Dr. Florence Gnana Poovathy J**	Slot	:	**L15+L16**
Name	:	**Hariket Sukesh Kumar Sheth**	Register No.	:	**20BCE1975**

Sort Array Elements in Ascending Order

Aim: To Perform Sorting (Ascending Order) for the elements of the Array

Tool Used: Assembler – MASM611

Algorithm:

Step 1: First of all, mount the c drive using the command: **mount c c:\masm611\bin**

Step 2: After pressing **enter,** type **c:** and press enter.

Step 3: Now give a command, **ascend.asm** for writing/editing the code and the write the code.

Step 4: A pop window appears; there we have to write out code(instructions) following the logic given below.

1. Start the Data Segment.
2. Declare Array Numbers and Store five Hexadecimal Numbers.
3. Data Segment Ends.
4. Start Code Segment.
5. Assume Ds is Data and Cs is Code.
6. Move Data into AX.
7. Move AX into DS.
8. Move 04H into CH.
9. Start LOOP2 and Mov 04H into CL.
10. Load the Effective Address (LEA) into SI from Numbers.
11. Start LOOP1 o Move [SI] into AL and [SI+1] into BL.
12. Compare BL and AL.
13. JC Down.
14. Move [SI+1] into DL.
15. Exchange [SI] and DL.
16. Move DL into [SI+1].
17. Start Down.

18. Increment SI.
19. Decrement CL.
20. JNZ LOOP1.
21. Decrement CH.
22. JNZ LOOP2.

HLT and End of Code Segment

Step 5: Now give a command, **masm ascend.asm** for running the code. The object file is created.

Step 6: Now give a command, **link ascend.obj** to link the object file to library file present in the bin folder.

Step 7: Press **ENTER** four times.

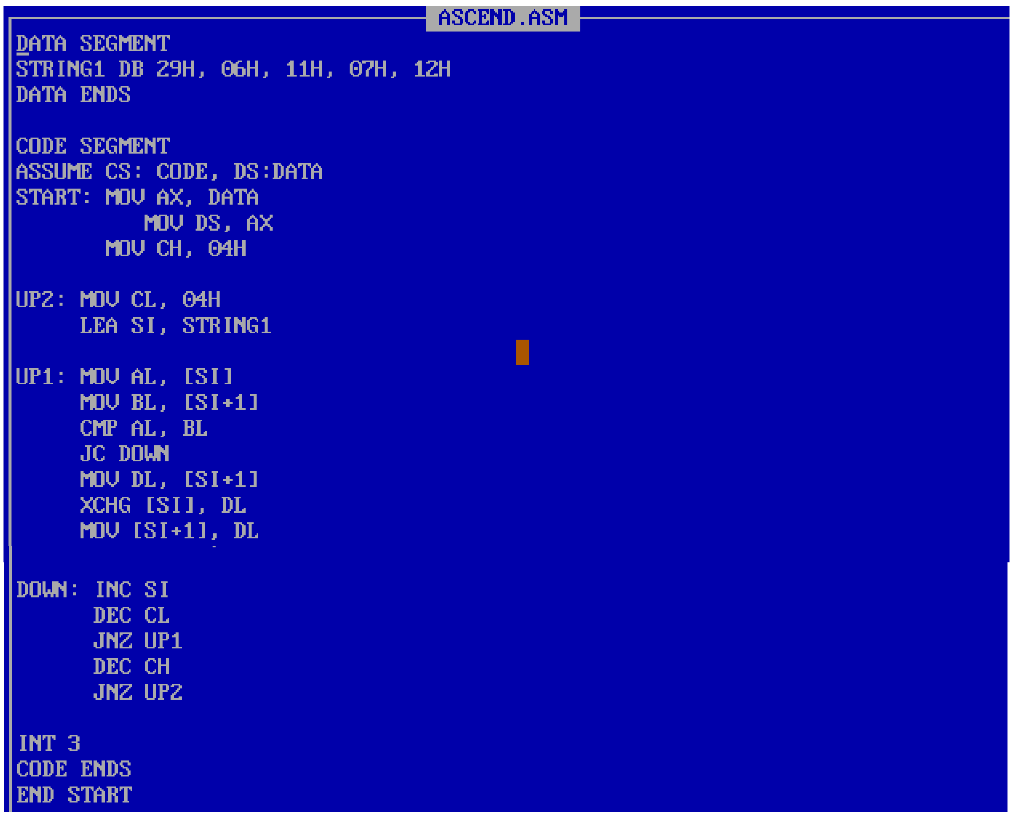
Step 8: Write debug **ascend.exe**

-u

-g (followed by the **address of HLT or INT** to view the values in registers).

-d (followed by the address of the Data Segment and index of the Array 0 to 4)

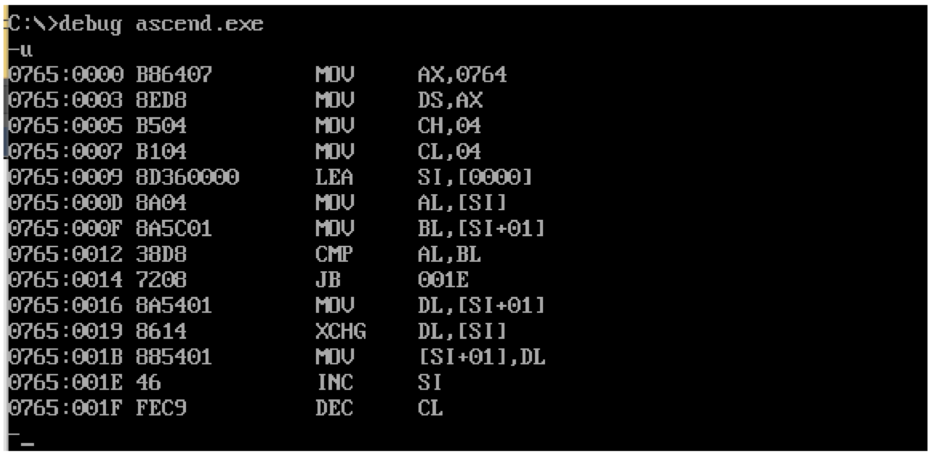
Program:



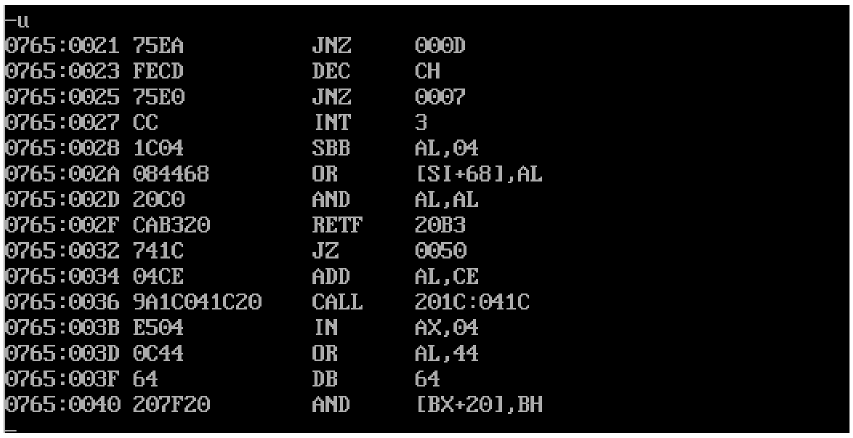
```
DATA SEGMENT
STRING1 DB 29H, 06H, 11H, 07H, 12H
DATA ENDS

CODE SEGMENT
ASSUME CS: CODE, DS:DATA
START: MOV AX, DATA
          MOV DS, AX
       MOV CH, 04H

UP2: MOV CL, 04H
     LEA SI, STRING1

UP1: MOV AL, [SI]
     MOV BL, [SI+1]
     CMP AL, BL
     JC DOWN
     MOV DL, [SI+1]
     XCHG [SI], DL
     MOV [SI+1], DL

DOWN: INC SI
      DEC CL
      JNZ UP1
      DEC CH
      JNZ UP2

INT 3
CODE ENDS
END START
```

Sample Input:	**Sample Output:**
String1: 29H, 06H, 11H, 07H, 12H	String1: 06H, 07H, 11H, 12H, 29H Maximum Element: 29H Minimum Element: 06H

Register / Memory Contents for I/O:

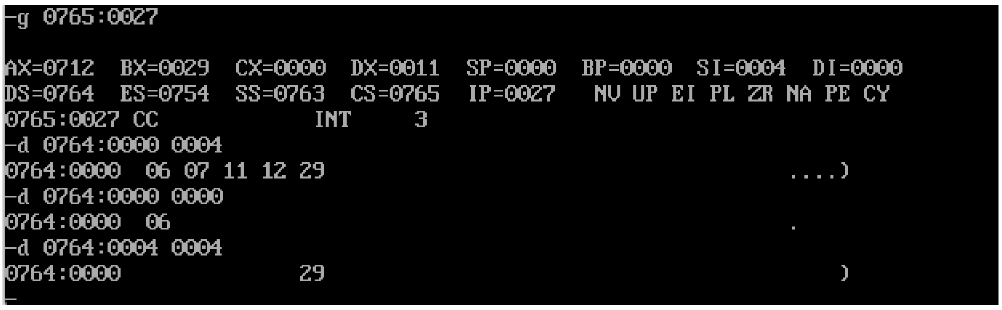


```
C:\>debug ascend.exe
-u
0765:0000 B86407        MOV     AX,0764
0765:0003 8ED8          MOV     DS,AX
0765:0005 B504          MOV     CH,04
0765:0007 B104          MOV     CL,04
0765:0009 8D360000      LEA     SI,[0000]
0765:000D 8A04          MOV     AL,[SI]
0765:000F 8A5C01        MOV     BL,[SI+01]
0765:0012 38D8          CMP     AL,BL
0765:0014 7208          JB      001E
0765:0016 8A5401        MOV     DL,[SI+01]
0765:0019 8614          XCHG    DL,[SI]
0765:001B 885401        MOV     [SI+01],DL
0765:001E 46            INC     SI
0765:001F FEC9          DEC     CL
-_
```



```
-u
0765:0021 75EA          JNZ     000D
0765:0023 FECD          DEC     CH
0765:0025 75E0          JNZ     0007
0765:0027 CC            INT     3
0765:0028 1C04          SBB     AL,04
0765:002A 084468        OR      [SI+68],AL
0765:002D 20C0          AND     AL,AL
0765:002F CAB320        RETF    20B3
0765:0032 741C          JZ      0050
0765:0034 04CE          ADD     AL,CE
0765:0036 9A1C041C20    CALL    201C:041C
0765:003B E504          IN      AX,04
0765:003D 0C44          OR      AL,44
0765:003F 64            DB      64
0765:0040 207F20        AND     [BX+20],BH
-
```

Output:



```
-g 0765:0027

AX=0712  BX=0029  CX=0000  DX=0011  SP=0000  BP=0000  SI=0004  DI=0000
DS=0764  ES=0754  SS=0763  CS=0765  IP=0027   NV UP EI PL ZR NA PE CY
0765:0027 CC            INT     3
-d 0764:0000 0004
0764:0000  06 07 11 12 29                                    ....)
-d 0764:0000 0000
0764:0000  06                                                .
-d 0764:0004 0004
0764:0000              29                                        )
-
```

Sort Array Elements in Descending Order

Aim: To Perform Sorting (Descending Order) for the elements of the Array

Tool Used: Assembler – MASM611

Algorithm:

Step 1: First of all, mount the c drive using the command: **mount c c:\masm611\bin**

Step 2: After pressing **enter,** type **c:** and press enter.

Step 3: Now give a command, **descend.asm** for writing/editing the code and the write the code.

Step 4: A pop window appears; there we have to write out code(instructions) following the logic given below.

1. Start the Data Segment.
2. Declare Array Numbers and Store five Hexadecimal Numbers
3. Data Segment Ends.
4. Start Code Segment.
5. Assume Ds is Data and Cs is Code.
6. Move Data into AX.
7. Move AX into DS.
8. Move 04H into CH.
9. Start LOOP2 and Mov 04H into CL.
10. Load the Effective Address (LEA) into SI from Numbers.
11. Start LOOP1.
12. Move [SI] into AL and [SI+1] into BL.
13. Compare BL and AL.
14. JNC Down.
15. Move [SI+1] into DL.
16. Exchange [SI] and DL.

17. Move DL into [SI+1].
18. Start Down.
19. Increment SI.
20. Decrement CL.
21. JNZ LOOP1.
22. Decrement CH.
23. JNZ LOOP2.
24. HLT and End of Code Segment.
25. End of Program

Step 5: Now give a command, **masm descend.asm** for running the code. The object file is created.

Step 6: Now give a command, **link descend.obj** to link the object file to library file present in the bin folder.

Step 7: Press **ENTER** four times.

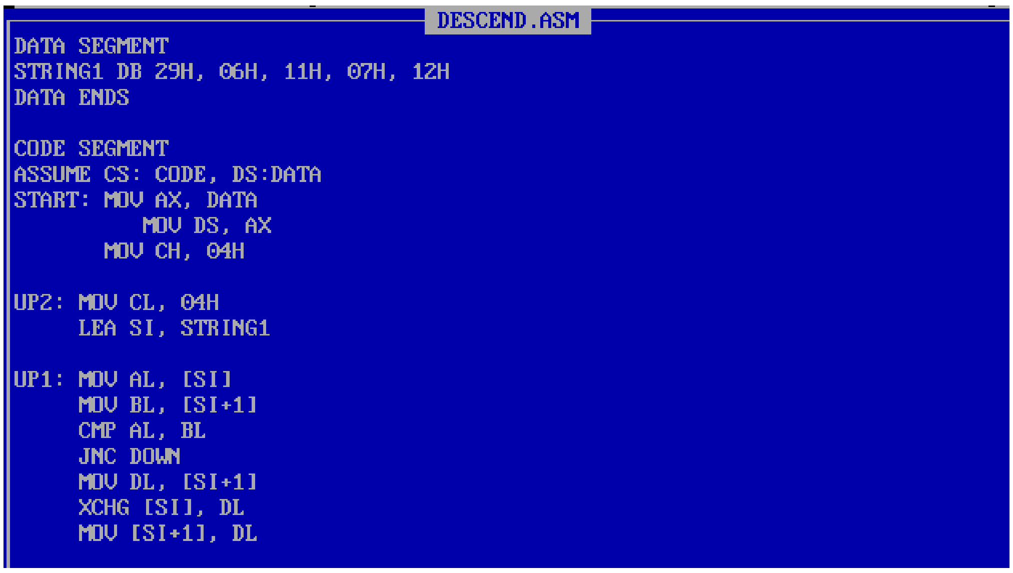
Step 8: Write debug **descend.exe**

-u

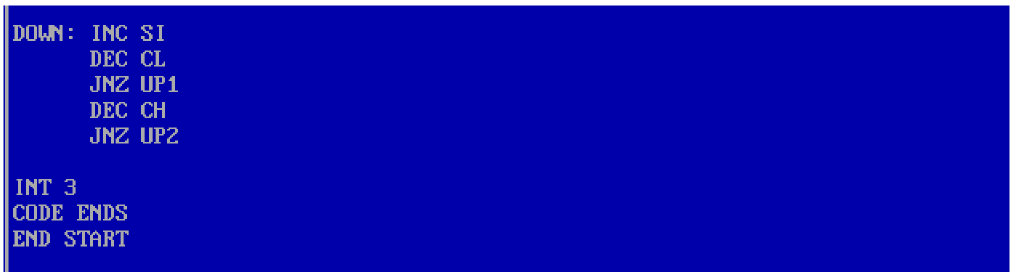
-g (followed by the **address of HLT or INT** to view the values in registers).

-d (followed by the address of the Data Segment and index of the Array 0 to 4)

Program:



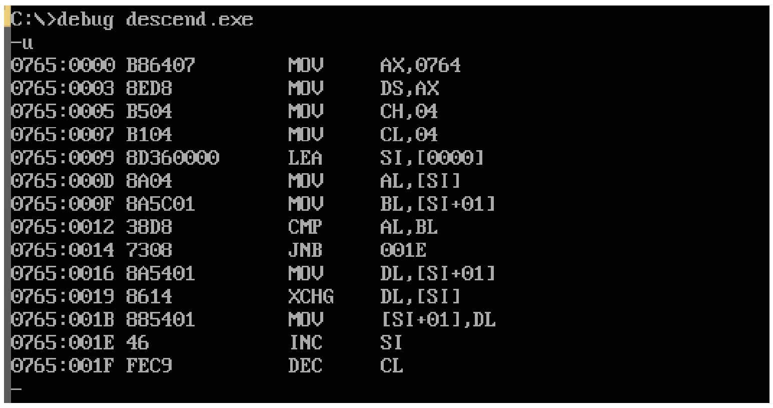
```
DATA SEGMENT
STRING1 DB 29H, 06H, 11H, 07H, 12H
DATA ENDS

CODE SEGMENT
ASSUME CS: CODE, DS:DATA
START: MOV AX, DATA
          MOV DS, AX
       MOV CH, 04H

UP2: MOV CL, 04H
     LEA SI, STRING1

UP1: MOV AL, [SI]
     MOV BL, [SI+1]
     CMP AL, BL
     JNC DOWN
     MOV DL, [SI+1]
     XCHG [SI], DL
     MOV [SI+1], DL
```



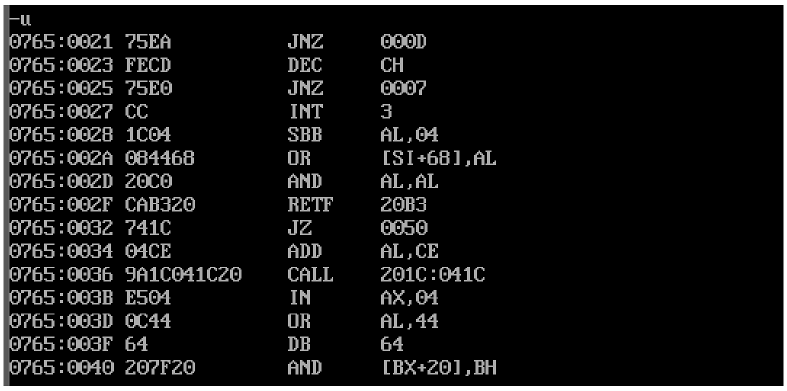
```
DOWN: INC SI
      DEC CL
      JNZ UP1
      DEC CH
      JNZ UP2

INT 3
CODE ENDS
END START
```

Sample Input:	**Sample Output:**
String1: 29H, 06H, 11H, 07H, 12H	String1: 29H, 12H, 11H, 07H, 06H Last Element: 06H First Element: 29H

Register / Memory Contents for I/O:

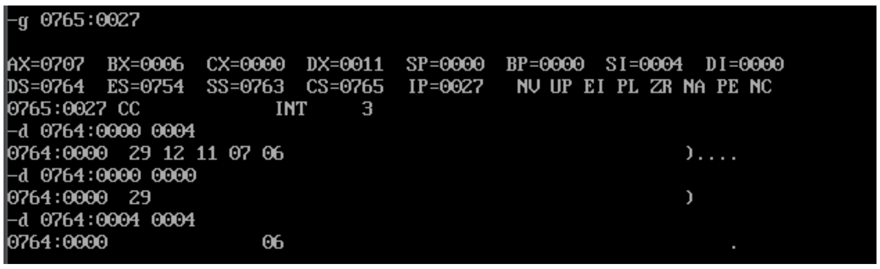


```
C:\>debug descend.exe
-u
0765:0000 B86407        MOV     AX,0764
0765:0003 8ED8          MOV     DS,AX
0765:0005 B504          MOV     CH,04
0765:0007 B104          MOV     CL,04
0765:0009 8D360000      LEA     SI,[0000]
0765:000D 8A04          MOV     AL,[SI]
0765:000F 8A5C01        MOV     BL,[SI+01]
0765:0012 38D8          CMP     AL,BL
0765:0014 7308          JNB     001E
0765:0016 8A5401        MOV     DL,[SI+01]
0765:0019 8614          XCHG    DL,[SI]
0765:001B 885401        MOV     [SI+01],DL
0765:001E 46            INC     SI
0765:001F FEC9          DEC     CL
-
```



```
-u
0765:0021 75EA          JNZ     000D
0765:0023 FECD          DEC     CH
0765:0025 75E0          JNZ     0007
0765:0027 CC            INT     3
0765:0028 1C04          SBB     AL,04
0765:002A 084468        OR      [SI+68],AL
0765:002D 20C0          AND     AL,AL
0765:002F CAB320        RETF    20B3
0765:0032 741C          JZ      0050
0765:0034 04CE          ADD     AL,CE
0765:0036 9A1C041C20    CALL    201C:041C
0765:003B E504          IN      AX,04
0765:003D 0C44          OR      AL,44
0765:003F 64            DB      64
0765:0040 207F20        AND     [BX+20],BH
```

Output:



```
-g 0765:0027

AX=0707  BX=0006  CX=0000  DX=0011  SP=0000  BP=0000  SI=0004  DI=0000
DS=0764  ES=0754  SS=0763  CS=0765  IP=0027   NV UP EI PL ZR NA PE NC
0765:0027 CC            INT     3
-d 0764:0000 0004
0764:0000  29 12 11 07 06                                    )....
-d 0764:0000 0000
0764:0000  29                                                )
-d 0764:0004 0004
0764:0000              06                                        .
```