Name: **Tushar Nankani** Roll No: **1902112** Batch: **C23**

**Microprocessor: Experiment 5**

**Aim**: Assembly program to sort numbers in ascending/descending order.

**Theory:**

A Sorting Algorithm is used to rearrange a given elements according to a comparison operator on the elements.

Sorted elements can be easy to interact with and it is very easy to perform various operations on them like adding more elements, adding more elements at a specific position, deleting elements at a specific position, deleting from start or end, etc.

**ALGORITHM: -**

1.Start

2.Initialize data segment through AX register in the DS register.

3.Display Array Limit

4.Initialize Si to 4000 and Cx to 10

5.Display message to enter Number

6.Take input of 1 digit into AL Register

7.Rotate AL Register to left by 4 bits

8.Copy contents of AL to BL

9.Take input of second digit into AL

10.Add AL to BL

11.Copy BL to Stack

12.Increment SI 13.Repeat from step 5 to step 12 10 Times

14.Initialise CX to 9

15.Initialise BX to 0.

16.Set SI to 4000.

17.Copy Data at location of SI to BL

18.Increment SI

19.Compare BL and contents at SI

20.IF Contents at SI are greater than BL Jump to 27

21.Move contents BL to DL

22.Move contents at SI to BL

23.Decrement SI

24.Move BL into Contents at SI

25.Increment SI

26.Move contents in DL to contents at SI

27.Move contents at SI to BL

28.Increment SI

29.Repeat from step 17 to step 28 9 Times

30.Set SI to 4000 and Cx to 10

31.Print Output Message

32.Copy contents at SI to BL

33.Print first Digit of BL

34.Print Second Digit of BL

35.Increment SI

36.Repeat from step 31 to step 35 10 Times

37.Stop

**Code:**

1. Ascending order.

DATA SEGMENT

MSG1 DB 10,13,"ARRAY LIMIT IS 10 $"

MSG2 DB 10,13,"Enter the number: $"

MSG3 DB 10,13,"Sorted Array: $"

MSG4 DB 10,13,"$"

COUNT DB ?

TEMP DB ?

DATA ENDS

CODE SEGMENT

ASSUME DS:DATA,CS:CODE

START:

MOV AX,DATA

MOV DS,AX

LEA DX,MSG1

MOV AH,09H

INT 21H

MOV SI,4000H

MOV CX,10

LABEL1:

LEA DX,MSG2

MOV AH,09H

INT 21H

MOV AH,01H

INT 21H

CALL INPUT

ROL AL,04H

MOV BL,AL

MOV AH,01H

INT 21H

CALL INPUT

MOV AH,00H

ADD BL,AL

MOV [SI],BL

INC SI

LEA DX,MSG4

MOV AH,09H

INT 21H

LOOP LABEL1

MOV CX,9

mov bx,0000h

LABEL2:

MOV SI,4000H

MOV BL,[SI]

INC SI

MOV BH,9

LABEL3:

DEC BH

CMP BL,[SI]

JC LABEL4

MOV DL,BL

MOV BL,[SI]

DEC SI

MOV [SI],BL

INC SI

MOV [SI],DL

LABEL4:

MOV BL,[SI]

INC SI

CMP BH,00H

JNZ LABEL3

LOOP LABEL2

MOV SI,4000H

MOV CX,10

d:

LEA DX,MSG3

MOV AH,09H

INT 21H

MOV BL,00H

MOV BL,[SI]

AND BL,0F0H

ROR BL,04H

CALL OUTPUT

MOV BL,[SI]

AND BL,0FH

CALL OUTPUT

INC SI

LOOP d

MOV AH,4CH

INT 21H

INPUT PROC

CMP AL,41H

JC LABEL6

SUB AL,07H

LABEL6:

SUB AL,30H

RET

ENDP

OUTPUT PROC

CMP BL,0AH

JC LABEL7

ADD BL,07H

LABEL7:

ADD BL,30H

MOV DL,BL

MOV AH,02H

INT 21H

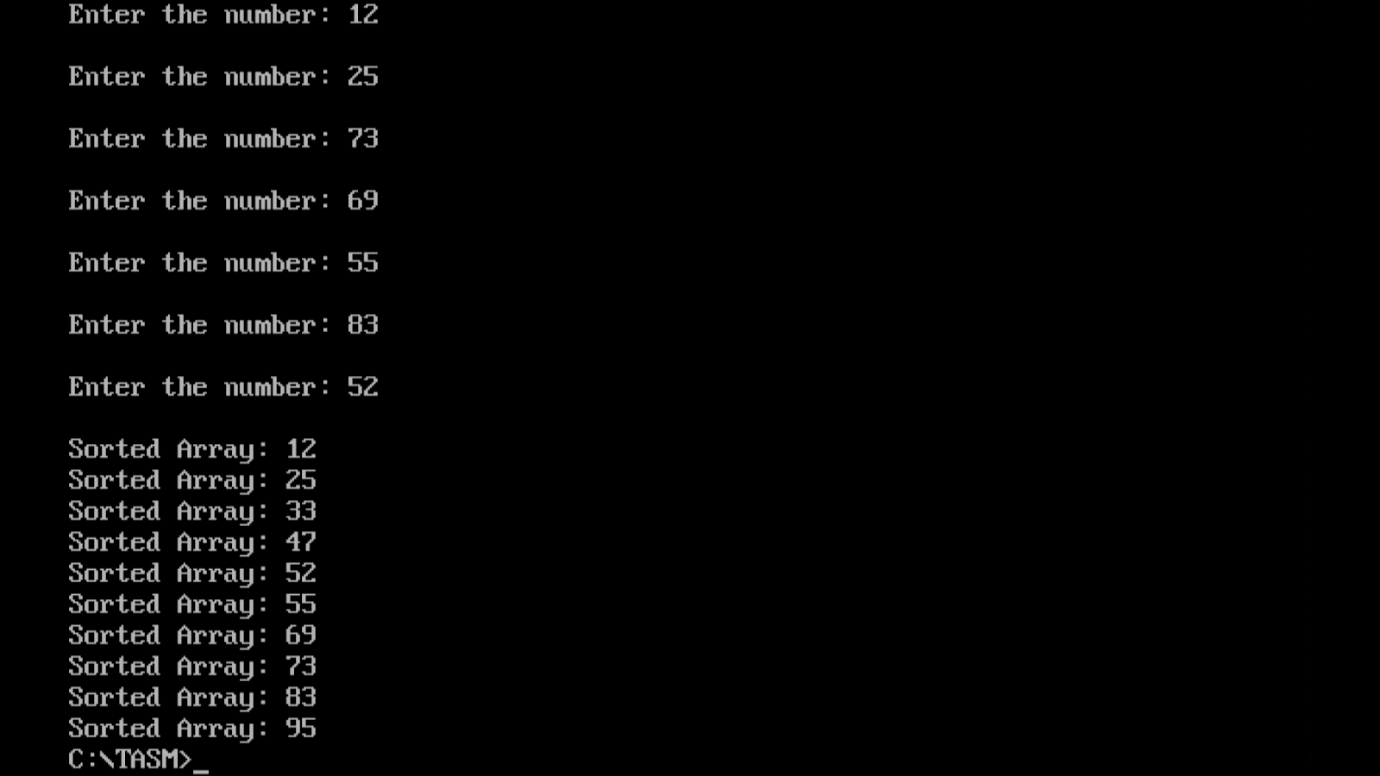
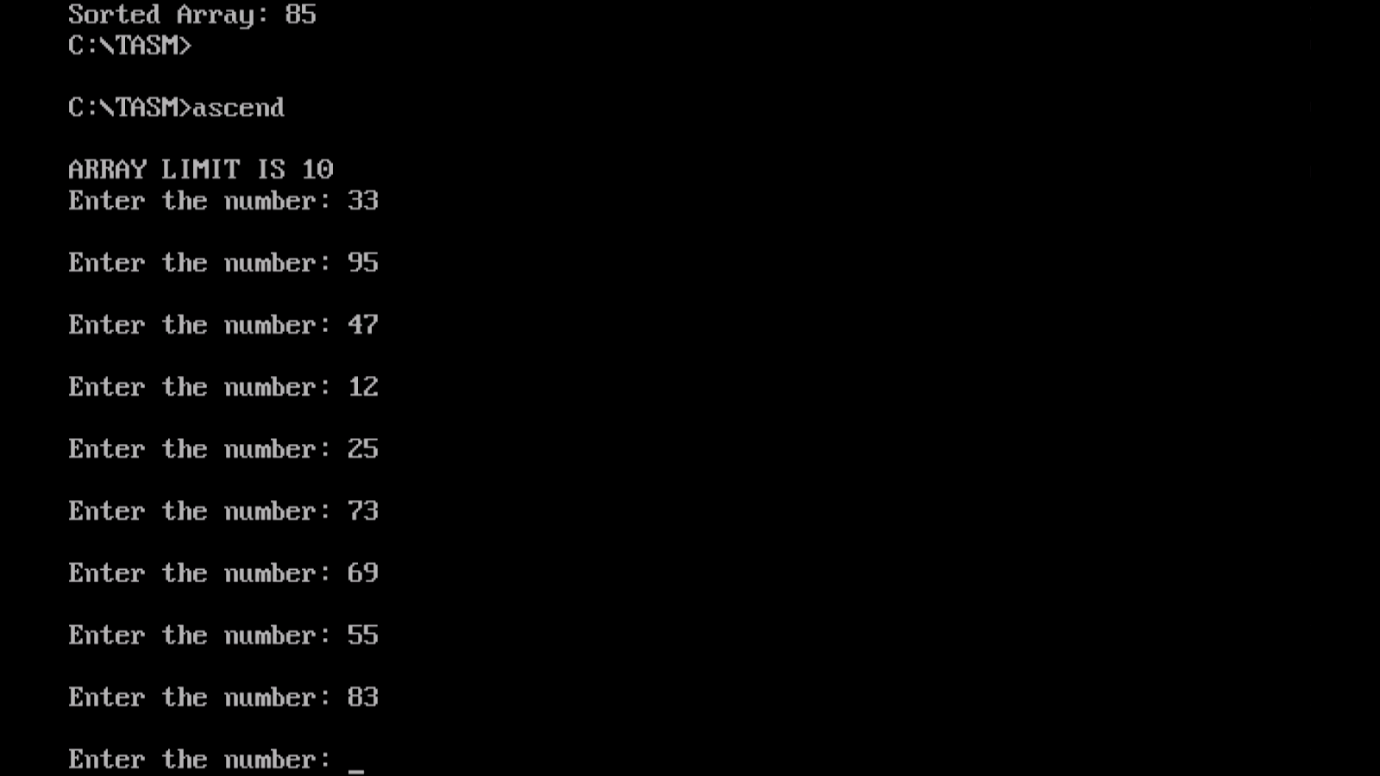
RET

ENDP

CODE ENDS

END START

**OUTPUT:**



2. Descending order.

DATA SEGMENT

MSG1 DB 10,13,"ARRAY LIMIT IS 10 $"

MSG2 DB 10,13,"Enter the number: $"

MSG3 DB 10,13,"Sorted Array: $"

MSG4 DB 10,13,"$"

COUNT DB ?

TEMP DB ?

DATA ENDS

CODE SEGMENT

ASSUME DS:DATA,CS:CODE

START:

MOV AX,DATA

MOV DS,AX

LEA DX,MSG1

MOV AH,09H

INT 21H

MOV SI,4000H

MOV CX,10

LABEL1:

LEA DX,MSG2

MOV AH,09H

INT 21H

MOV AH,01H

INT 21H

CALL INPUT

ROL AL,04H

MOV BL,AL

MOV AH,01H

INT 21H

CALL INPUT

MOV AH,00H

ADD BL,AL

MOV [SI],BL

INC SI

LEA DX,MSG4

MOV AH,09H

INT 21H

LOOP LABEL1

MOV CX,9

mov bx,0000h

LABEL2:

MOV SI,4000H

MOV BL,[SI]

INC SI

MOV BH,9

LABEL3:

DEC BH

CMP BL,[SI]

JNC LABEL4

MOV DL,BL

MOV BL,[SI]

DEC SI

MOV [SI],BL

INC SI

MOV [SI],DL

LABEL4:

MOV BL,[SI]

INC SI

CMP BH,00H

JNZ LABEL3

LOOP LABEL2

MOV SI,4000H

MOV CX,10

d:

LEA DX,MSG3

MOV AH,09H

INT 21H

MOV BL,00H

MOV BL,[SI]

AND BL,0F0H

ROR BL,04H

CALL OUTPUT

MOV BL,[SI]

AND BL,0FH

CALL OUTPUT

INC SI

LOOP d

MOV AH,4CH

INT 21H

INPUT PROC

CMP AL,41H

JC LABEL6

SUB AL,07H

LABEL6:

SUB AL,30H

RET

ENDP

OUTPUT PROC

CMP BL,0AH

JC LABEL7

ADD BL,07H

LABEL7:

ADD BL,30H

MOV DL,BL

MOV AH,02H

INT 21H

RET

ENDP

CODE ENDS

END START

**OUTPUT:**

