SORTING

Exp No.: 6 **Name:** S Vishakan

Date: 23-09-2020 **Reg. No:** 18 5001 196

AIM:

To write assembly language programs to perform the following experiments:

1. Ascending order sorting using Bubble Sort.
2. Descending order sorting using Bubble Sort.

PROGRAM – 1: ASCENDING ORDER SORT:

ALGORITHM:

1. Begin.

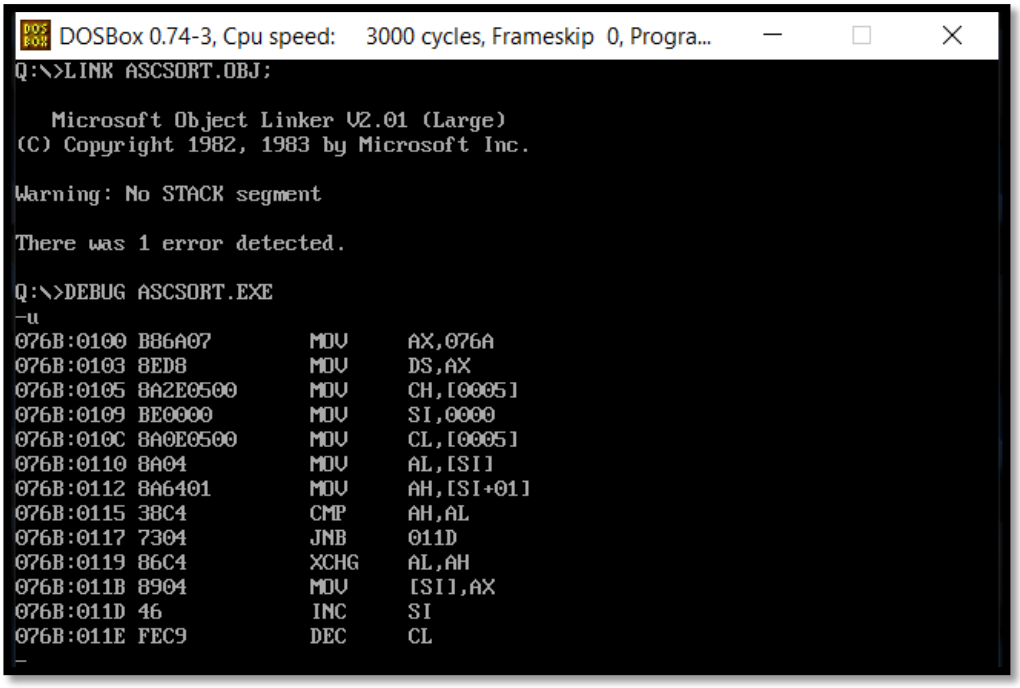
2. Declare the data segment.
3. Initialize data segment with array and its length len.
4. Close the data segment.

5. Declare the code segment.
6. Set a preferred offset (preferably 100)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Move the length len to CH register.
10. Till CH goes to zero:
    a. Load SI with offset of list.
    b. Move the length len to CL register.
    c. Till CL goes to zero:
        i. Compare values at SI and SI+1 address.
        ii. If value at SI > value at SI+1, exchange them.
        iii. Increment SI.
        iv. Decrement CL.
    d. Decrement CH.
11. Introduce an interrupt for safe exit. (INT 21h)
12. Close the code segment.

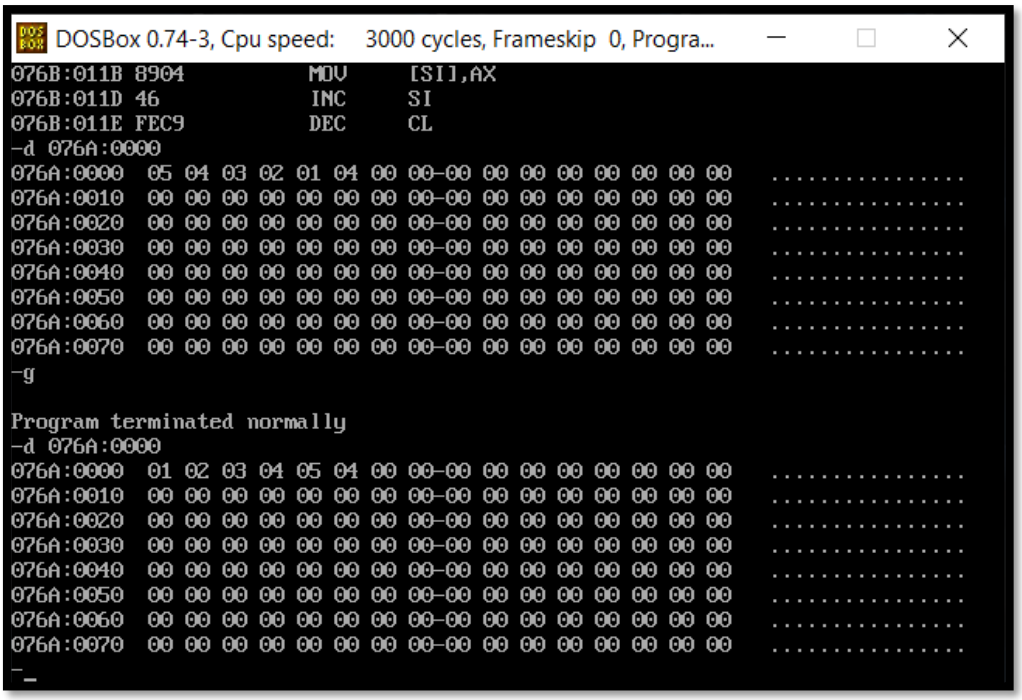
13. End.

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
list db 05h, 04h, 03h, 02h, 01h	Stores the list of elements.
len db 04h	Stores the length of the above array.
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: mov ax, data	Transfer data from “data” to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov ch, len	
outer: mov si, offset list	Pointer at first element.
mov cl, len	Inner loop count.
inner: mov al, [si]	
mov ah, [si+1]	
cmp ah, al	Compare by AL – AH.
jnc skip	Skip if no carry occurred on AL – AH.
xchg al, ah	Exchange register contents.
mov [si], ax	Copy back moved contents to data segment (AL -> [SI], AH -> [SI + 1])
skip: inc si	Go to next element.
dec cl	Decrement inner loop count.
jnz inner	Restart inner loop.
dec ch	Decrement outer loop count.
jnz outer	Restart outer loop.
mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:



SAMPLE I/O SNAPSHOT:



PROGRAM – 2: DESCENDING ORDER SORT:

ALGORITHM:

1. Begin.

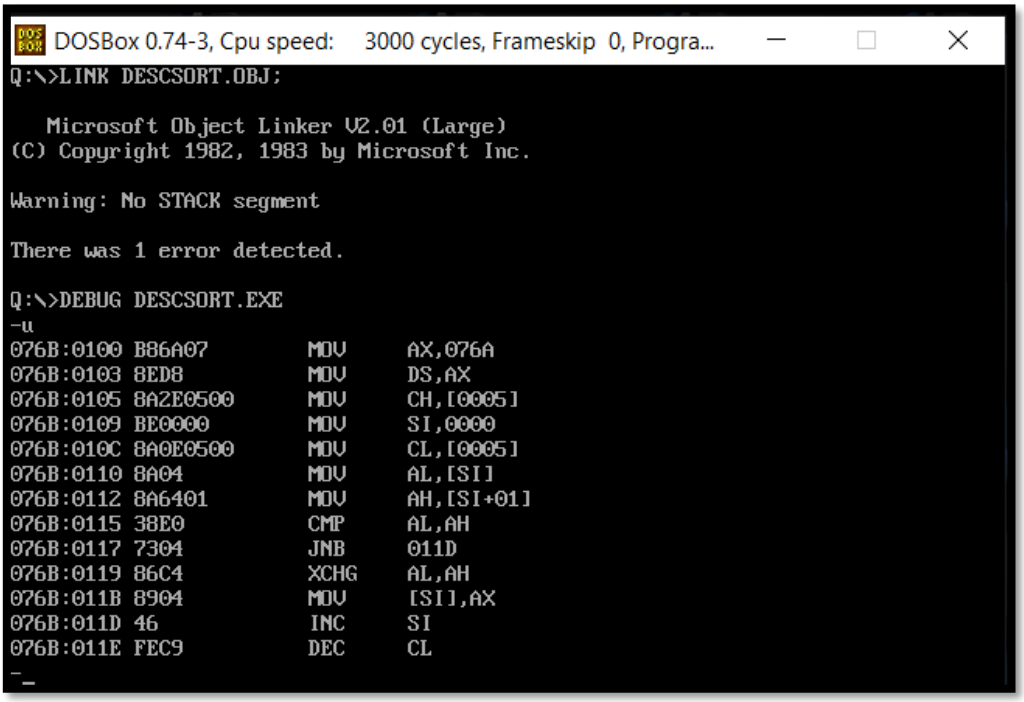
2. Declare the data segment.
3. Initialize data segment with array and its length len.
4. Close the data segment.

5. Declare the code segment.
6. Set a preferred offset (preferably 100)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Move the length len to CH register.
10. Till CH goes to zero:
    a. Load SI with offset of list.
    b. Move the length len to CL register.
    c. Till CL goes to zero:
        i. Compare values at SI and SI+1 address.
        ii. If value at SI < value at SI+1, exchange them.
        iii. Increment SI.
        iv. Decrement CL.
    d. Decrement CH.
11. Introduce an interrupt for safe exit. (INT 21h)
12. Close the code segment.

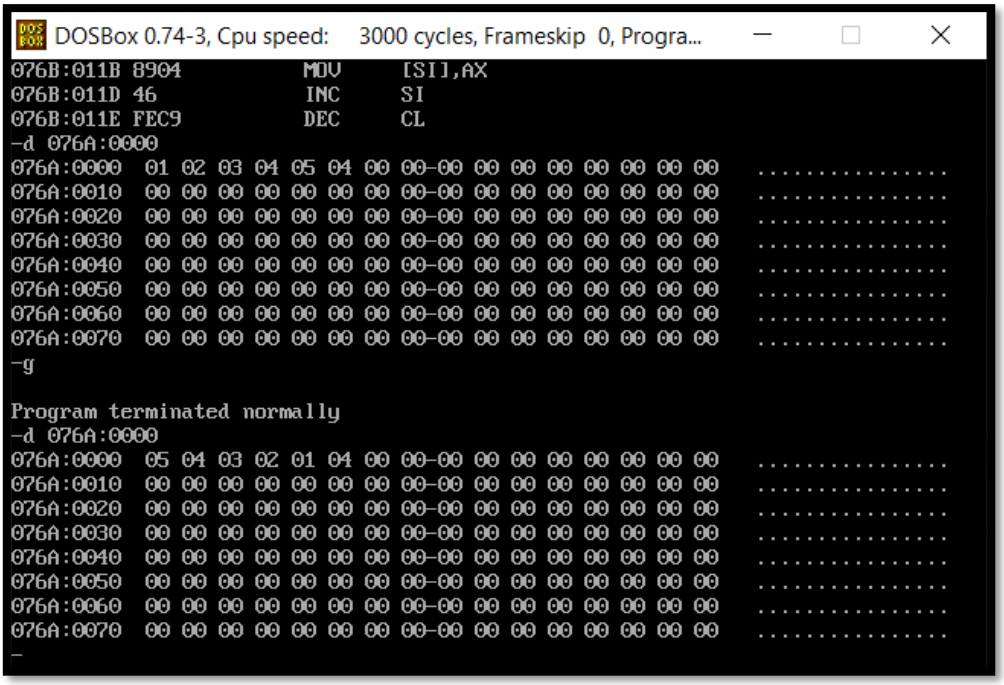
13. End.

PROGRAM	COMMENTS
assume cs:code, ds:data	Declare code and data segment.
data segment	Initialize data segment with values.
list db 05h, 04h, 03h, 02h, 01h	Stores the list of elements.
len db 04h	Stores the length of the above array.
data ends	
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: mov ax, data	Transfer data from “data” to AX.
mov ds, ax	Transfer data from memory location AX to DS.
mov ch, len	
outer: mov si, offset list	Pointer at first element.
mov cl, len	Inner loop count.
inner: mov al, [si]	
mov ah, [si+1]	
cmp al, ah	Compare by AH – AL.
jnc skip	Skip if no carry occurred on AH – AL.
xchg al, ah	Exchange register contents.
mov [si], ax	Copy back moved contents to data segment (AL -> [SI], AH -> [SI + 1])
skip: inc si	Go to next element.
dec cl	Decrement inner loop count.
jnz inner	Restart inner loop.
dec ch	Decrement outer loop count.
jnz outer	Restart outer loop.
mov ah, 4ch	
int 21h	Interrupt the process with return code and exit.
code ends	
end start	

UNASSEMBLED CODE:



SAMPLE I/O SNAPSHOT:



RESULT:

The assembly level programs were written to perform the above specified sorting functions and the output was verified.