Assembly Programs:

1. write an alp to implement bubble sort

section .data

msg db "Enter number of elements: "

msglen equ $-msg

msg2 db "Enter the elements in the array: "

msg2len equ $-msg2

msg3 db "The sorted array is: "

msg3len equ $-msg3

msg4 db "Pass "

msg4len equ $-msg4

msg5 db " : "

msg5len equ $-msg5

newline db 10

space db ' '

%macro write 2

mov eax,4

mov ebx,1

mov ecx,%1

mov edx,%2

int 80h

%endmacro

%macro read 2

mov eax,3

mov ebx,2

mov ecx,%1

mov edx,%2

int 80h

mov eax,3

mov ebx,2

mov ecx,trash

mov edx,1

int 80h

%endmacro

input:

write msg2,msg2len

mov [i],dword '0'

loop1:

mov esi,[i]

cmp esi,[n]

jge end1

sub esi,'0'

add esi,arr

read esi,1

inc dword[i]

jmp loop1

end1:

ret

display:

mov [i],dword '0'

loop2:

mov esi,[i]

cmp esi,[n]

jge end2

sub esi,'0'

add esi,arr

write esi,1

write space,1

inc dword[i]

jmp loop2

end2:

write newline,1

ret

bubble\_sort:

mov al,0

mov bl,[n]

sub bl,'0'

sub bl,1

loop3:

cmp al,bl

jge end3

pushad

write msg4,msg4len

write j,9

write msg5,msg5len

call display

popad

mov ecx,0

mov dl,bl

sub dl,al

loop4:

cmp cl,dl

jge update1

mov esi,arr

add esi,ecx

mov ah,[esi]

mov bh,[esi+1]

cmp ah,bh

jle update2

mov [esi+1],ah

mov [esi],bh

update2:

inc cl

jmp loop4

update1:

inc al

inc byte[j]

jmp loop3

end3:

write msg4,msg4len

write j,9

write msg5,msg5len

call display

ret

section .bss

n resb 4

arr resb 10

i resb 4

j resb 9

trash resb 1

section .text

global \_start

\_start:

write msg,msglen

read n,1

call input

write newline,1

mov eax,'0'

mov [j],eax

call bubble\_sort

write newline,1

write msg3,msg3len

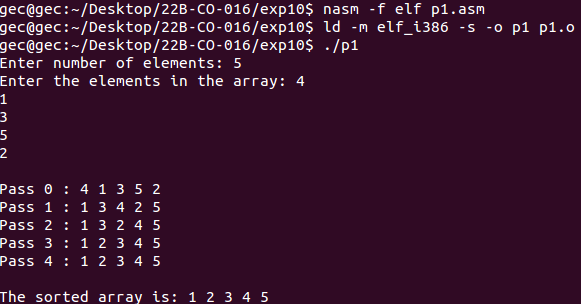
call display

mov eax,1

mov ebx,0

int 80h

**Output:**

****

2. write an alp to implement insertion sort

section .data

msg db "Enter number of elements: "

msglen equ $-msg

msg2 db "Enter the elements in the array: "

msg2len equ $-msg2

msg3 db "The sorted array is: "

msg3len equ $-msg3

msg4 db "Pass "

msg4len equ $-msg4

msg5 db " : "

msg5len equ $-msg5

newline db 10

space db ' '

%macro write 2

mov eax,4

mov ebx,1

mov ecx,%1

mov edx,%2

int 80h

%endmacro

%macro read 2

mov eax,3

mov ebx,2

mov ecx,%1

mov edx,%2

int 80h

mov eax,3

mov ebx,2

mov ecx,trash

mov edx,1

int 80h

%endmacro

input:

write msg2,msg2len

mov [i],dword '0'

loop1:

mov esi,[i]

cmp esi,[n]

jge end

sub esi,'0'

add esi,arr

read esi,1

inc dword[i]

jmp loop1

end:

ret

display:

mov [i],dword '0'

loop2:

mov esi,[i]

cmp esi,[n]

jge end2

sub esi,'0'

add esi,arr

write esi,1

write space,1

inc dword[i]

jmp loop2

end2:

write newline,1

ret

insertion\_sort:

mov eax,1

mov bl,[n]

sub bl,'0'

loop3:

cmp al,bl

jge end3

pushad

write msg4,msg4len

write j,9

write msg5,msg5len

call display

popad

mov ecx,0

mov cl,al

sub cl,1

mov dl,[arr+eax]

loop4:

cmp cl,0

jl update

cmp dl,[arr+ecx]

jge update

mov dh,[arr+ecx]

mov [arr+ecx+1],dh

dec ecx

jmp loop4

update:

mov [arr+ecx+1],dl

inc al

inc byte[j]

jmp loop3

end3:

write msg4,msg4len

write j,9

write msg5,msg5len

call display

ret

section .bss

n resb 4

arr resb 10

i resb 4

j resb 9

trash resb 1

section .text

global \_start

\_start:

write msg,msglen

read n,1

mov eax,'0'

mov [j],eax

call input

write newline,1

call insertion\_sort

write newline,1

write msg3,msg3len

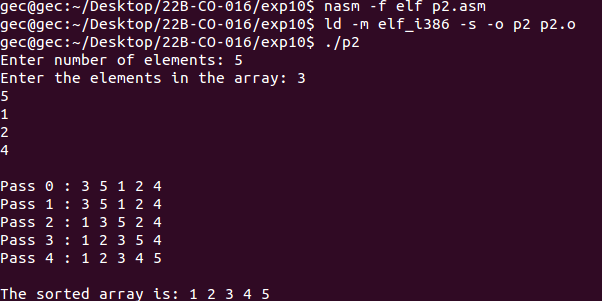
call display

mov eax,1

mov ebx,0

int 80h

**Output:**



3. write an alp to implement selection sort

section .data

msg db "Enter number of elements: "

msglen equ $-msg

msg2 db "Enter the elements in the array: "

msg2len equ $-msg2

msg3 db "The sorted array is: "

msg3len equ $-msg3

msg4 db "Pass "

msg4len equ $-msg4

msg5 db " : "

msg5len equ $-msg5

newline db 10

space db ' '

%macro write 2

mov eax,4

mov ebx,1

mov ecx,%1

mov edx,%2

int 80h

%endmacro

%macro read 2

mov eax,3

mov ebx,2

mov ecx,%1

mov edx,%2

int 80h

mov eax,3

mov ebx,2

mov ecx,trash

mov edx,1

int 80h

%endmacro

input:

write msg2,msg2len

mov [i],dword 0

loop1:

mov esi,[i]

cmp esi,[n]

jge end1

add esi,arr

read esi,1

inc dword[i]

jmp loop1

end1:

ret

display:

mov [i],dword 0

loop2:

mov esi,[i]

cmp esi,[n]

jge end2

add esi,arr

write esi,1

write space,1

inc dword[i]

jmp loop2

end2:

write newline,1

ret

selection\_sort:

mov eax,0

mov bl,[n]

sub bl,1

loop3:

cmp al,bl

jge end3

pushad

write msg4,msg4len

write j,9

write msg5,msg5len

call display

popad

mov ecx,0

mov cl,al

add cl,1

mov edi,arr

add edi,eax

loop4:

cmp cl,[n]

jge update1

mov esi,arr

add esi,ecx

mov bh,[esi]

mov dh,[edi]

cmp bh,dh

jge update2

mov edi,arr

add edi,ecx

update2:

inc cl

jmp loop4

update1:

mov bh,[arr+eax]

mov dh,[edi]

mov [arr+eax],dh

mov [edi],bh

inc al

inc byte[j]

jmp loop3

end3:

write msg4,msg4len

write j,9

write msg5,msg5len

call display

ret

section .bss

n resb 4

arr resb 10

i resb 4

j resb 9

trash resb 1

section .text

global \_start

\_start:

write msg,msglen

read n,1

sub byte[n],'0'

mov eax,'0'

mov [j],eax

call input

write newline,1

call selection\_sort

write newline,1

write msg3,msg3len

call display

mov eax,1

mov ebx,0

int 80h

**Output:**

