10A] Write an assembly language program to implement Bubble Sort.

**Program:**

%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

%macro endl 0

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 80h

%endmacro

%macro return 0

mov eax, 1

int 80h

%endmacro

section .data

asknum db 'Enter the number of elements: ';

asknumlen equ $-asknum

ask db 'Enter the elements'

asklen equ $-ask

cont db 'Array: '

contlen equ $-cont

srtd db 'The Sorted Array is'

srtedlen equ $-srtd

pass db 'Pass '

passlen equ $-pass

arrow db ': '

arlen equ $-arrow

space db ' '

splen equ $-space

nl db '', 10

nllen equ $-nl

array times 10 dw 0

len equ 10

section .bss

n resb 4

arr resb 10

i resb 4

j resb 9

trash resb 1

section .text

global \_start

\_start:

write asknum, asknumlen

read n, 1

; sub [n], byte '0'

; input

; mov eax, arr

; mov edx, [n]

call input

endl

mov eax, '0'

mov [j], eax

call bubblesort

write nl, nllen

write srtd, srtedlen

endl

; display

; mov eax, arr

; mov edx, [n]

call display

return

input:

write ask, asklen

write nl, nllen

mov [i], dword '0'

loop1:

mov esi, [i]

cmp esi, [n]

jge after1

sub esi, '0'

add esi, arr

read esi, 1

inc dword[i]

jmp loop1

after1:

ret

display:

write pass, passlen

write j, 9

write arrow, arlen

mov [i], dword '0'

loop2:

mov esi, [i]

cmp esi, [n]

jge after2

sub esi, '0'

add esi, arr

write esi, 1

write space, splen

inc dword[i]

jmp loop2

after2:

write nl, nllen

ret

bubblesort:

mov al, 0

mov bl, [n]

sub bl, '0'

sub bl, 1

loop3:

cmp al, bl

jge after3

pushad

mov eax, arr

mov edx, [n]

call display

popad

mov ecx, 0

mov dl, bl

sub dl, al

loop4:

cmp cl, dl

jge after4

mov esi, arr

add esi, ecx

mov ah, [esi]

mov bh, [esi + 1]

cmp ah, bh

jle after5

; Swap arr[ecx] and arr[ecx+1]

mov [esi + 1], ah

mov [esi], bh

after5:

inc cl

jmp loop4

after4:

inc al

inc byte[j]

jmp loop3

after3:

ret

**Output:**



10B] Write an assembly language program to implement Insertion Sort.

**Program:**

%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

%macro endl 0

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 80h

%endmacro

%macro return 0

mov eax, 1

int 80h

%endmacro

section .data

asknum db 'Enter the number of elements: ';

asknumlen equ $-asknum

ask db 'Enter the elements'

asklen equ $-ask

cont db 'Array: '

contlen equ $-cont

srtd db 'The Sorted Array is'

srtedlen equ $-srtd

pass db 'Pass '

passlen equ $-pass

arrow db ': '

arlen equ $-arrow

space db ' '

splen equ $-space

nl db '', 10

nllen equ $-nl

array times 10 dw 0

len equ 10

section .bss

n resb 4

arr resb 10

i resb 4

j resb 9

trash resb 1

section .text

global \_start

\_start:

write asknum, asknumlen

read n, 1

mov eax, '0'

mov [j], eax

call input

call insertionsort

endl

write srtd, srtedlen

endl

call display

return

; procedures

input:

write ask, asklen

write nl, nllen

mov [i], dword '0'

loop1:

mov esi, [i]

cmp esi, [n]

jge after1

sub esi, '0'

add esi, arr

read esi, 1

inc dword[i]

jmp loop1

after1:

ret

display:

write pass, passlen

write j, 9

write arrow, arlen

mov [i], dword '0'

loop2:

mov esi, [i]

cmp esi, [n]

jge after2

sub esi, '0'

add esi, arr

write esi, 1

write space, splen

inc dword[i]

jmp loop2

after2:

write nl, nllen

ret

insertionsort:

mov eax, 1

mov bl, [n]

sub bl, '0'

loop3:

cmp al, bl

jge after3

pushad

call display

popad

mov ecx, 0

mov cl, al

sub cl, 1

mov dl, [arr + eax]

loop4:

cmp cl, 0

jl after4

cmp dl, [arr + ecx]

jge after4

mov dh, [arr + ecx]

mov [arr + ecx + 1], dh

dec ecx

jmp loop4

after4:

mov [arr + ecx + 1], dl

inc al

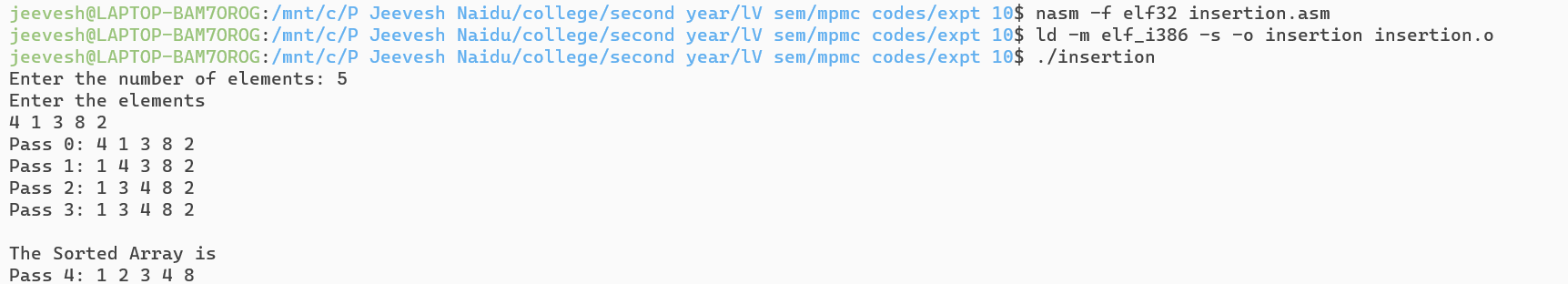
inc byte[j]

jmp loop3

after3:

ret

**Output:**



10C] Write an assembly language program to implement Selection Sort.

**Program:**

%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

%macro endl 0

mov eax, 4

mov ebx, 1

mov ecx, nl

mov edx, nllen

int 80h

%endmacro

%macro return 0

mov eax, 1

int 80h

%endmacro

section .data

asknum db 'Enter the number of elements: ';

asknumlen equ $-asknum

ask db 'Enter the elements'

asklen equ $-ask

cont db 'Array: '

contlen equ $-cont

srtd db 'The Sorted Array is'

srtedlen equ $-srtd

pass db 'Pass '

passlen equ $-pass

arrow db ': '

arlen equ $-arrow

space db ' '

splen equ $-space

nl db '', 10

nllen equ $-nl

array times 10 dw 0

len equ 10

section .bss

n resb 4

arr resb 10

i resb 4

j resb 9

trash resb 1

section .text

global \_start

\_start:

write asknum, asknumlen

read n, 1

sub byte[n], '0'

mov eax, '0'

mov [j], eax

call input

call selectionsort

endl

write srtd, srtedlen

endl

call display

return

; procedures

input:

write ask, asklen

write nl, nllen

mov [i], dword 0

loop1:

mov esi, [i]

cmp esi, [n]

jge after1

add esi, arr

read esi, 1

inc dword[i]

jmp loop1

after1:

ret

display:

write pass, passlen

write j, 9

write arrow, arlen

mov [i], dword 0

loop2:

mov esi, [i]

cmp esi, [n]

jge after2

add esi, arr

write esi, 1

write space, splen

inc dword[i]

jmp loop2

after2:

write nl, nllen

ret

selectionsort:

mov eax, 0

mov bl, [n]

sub bl, 1

loop3:

cmp al, bl

jge after3

pushad

call display

popad

mov ecx, 0

mov cl, al

add cl, 1

mov edi, arr

add edi, eax

loop4:

cmp cl, [n]

jge after4

mov esi, arr

add esi, ecx

mov bh, [esi]

mov dh, [edi]

cmp bh, dh

jge after5

mov edi, arr

add edi, ecx

after5:

inc cl

jmp loop4

after4:

; Swap arr[eax] and [edi]

mov bh, [arr + eax]

mov dh, [edi]

mov [arr + eax], dh

mov [edi], bh

inc al

inc byte[j]

jmp loop3

after3:

ret

**Output:**

