**LAB 5**

# Question 1:

Write a program that calculates the first 12 terms of Fibonacci sequence and stores them in an array

**PROGRAM:**

org 100h

.data

arr dw 0,1,10 dup (0)

.code

mov cl,11

lea bx,arr

lea si,arr

add si,2

top:

add ax,[bx]

add ax,[si]

mov bx,si

add si,2

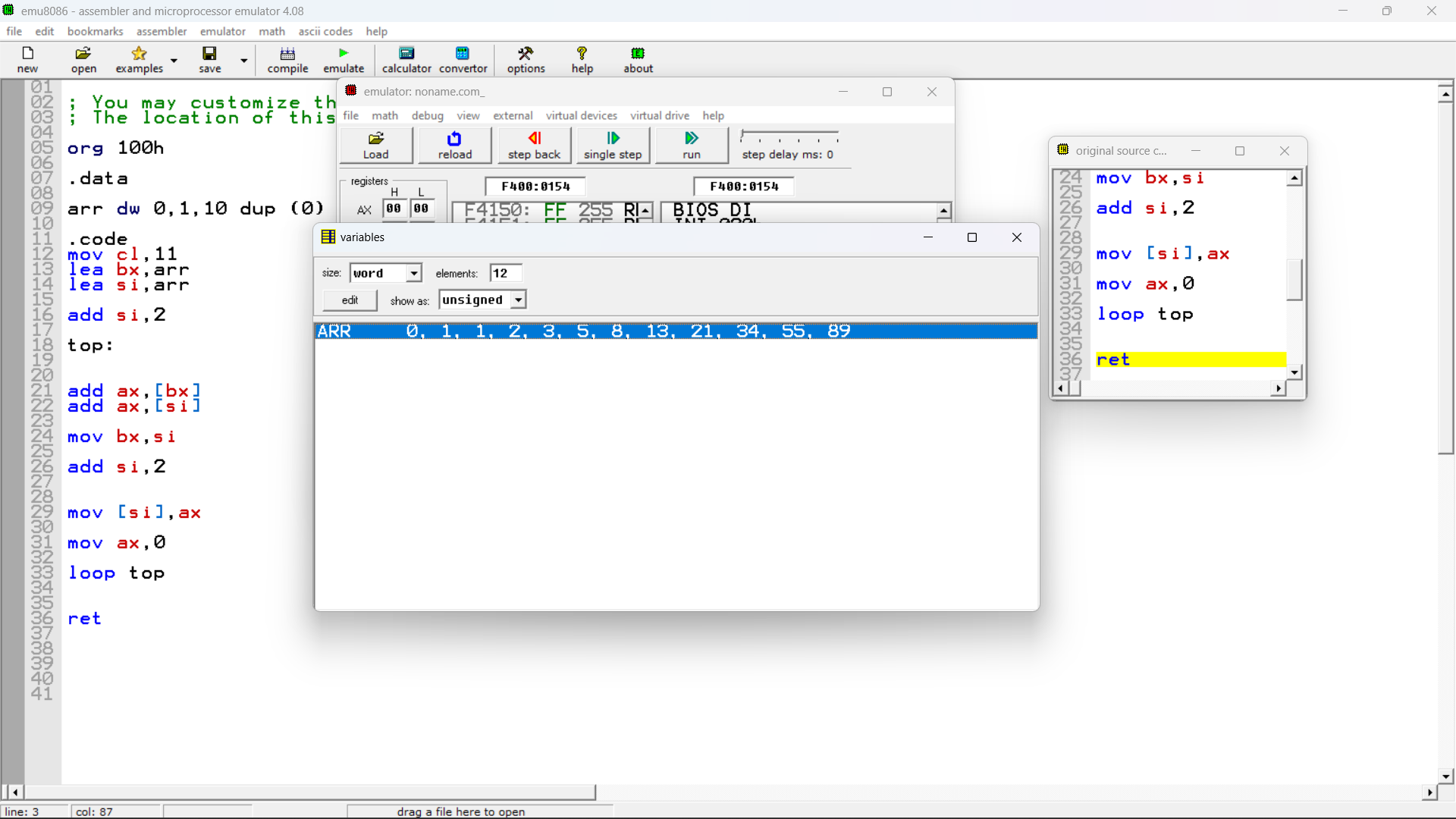
mov [si],ax

mov ax,0

loop top

ret

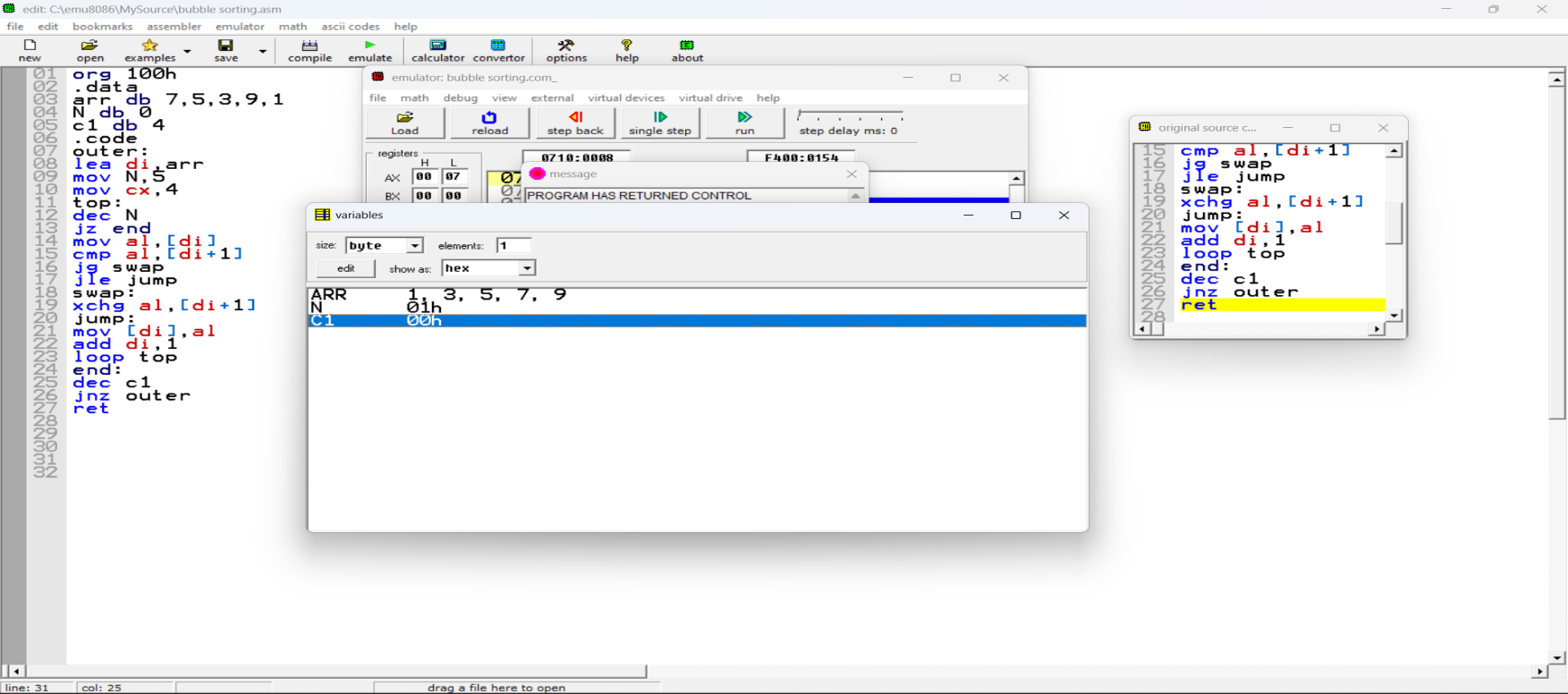
**OUTPUT:**



Question 2:  
(Modified version of exercise from chapter 10)  
To sort an array A of N elements by the bubblesort method, we  
proceed as follows:  
  
Pass 1: For j= 2 ... N, If A(j) <A(j - 1) then swap A(j) and A(j-1).  
This will place the largest element In position N.  
Pass 2: For j = 2 ... N -1, if A(j)< A(j-1) then swap A(j) and A(j -1).  
This will place the second largest element in position N -1.  
.  
.  
.  
Pass N -1: If A(2) < A(1), then swap A[2) and A(1).  
At this point the array is sorted.  
  
Demonstration :  
initial data 7 5 3 9 1  
pass 1 5 3 7 1 9  
pass 2 3 5 1 7 9  
pass 3 3 1 5 7 9  
pass 4 1 3 5 7 9  
  
Write a program segment to sort a byte array (DATA DB 7 5 3 9 1) by the bubblesort algorithm. The program should put the offset address of the array in Sl and the number of elements in variable N.

**SOLUTION:**

**OUTPUT:**



**PROGRAM:**

org 100h

.data

arr db 7,5,3,9,1

N db 0

c1 db 4

.code

outer:

lea di,arr

mov N,5

mov cx,4

top:

dec N

jz end

mov al,[di]

cmp al,[di+1]

jg swap

jle jump

swap:

xchg al,[di+1]

jump:

mov [di],al

add di,1

loop top

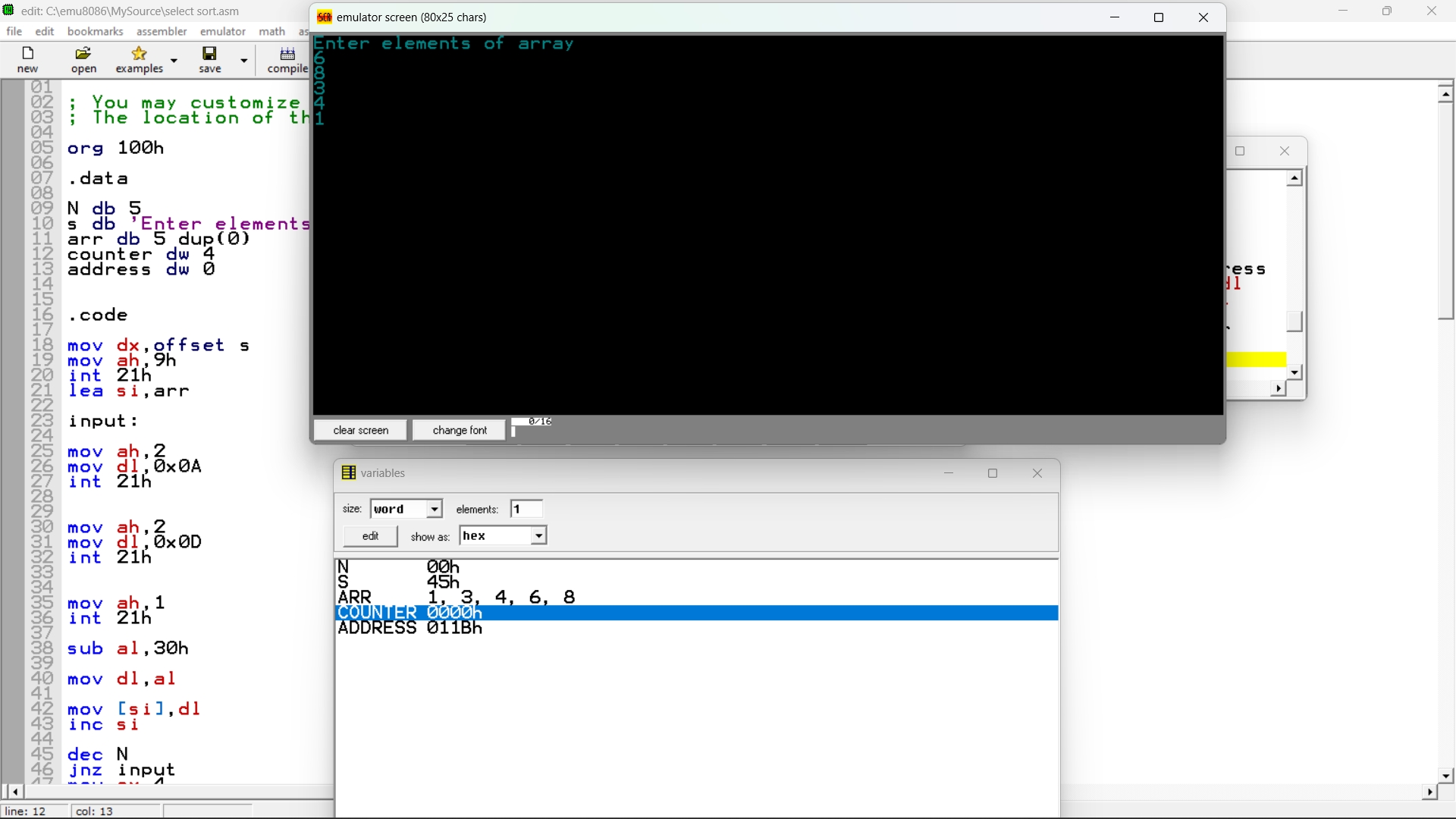
end:

dec c1

jnz outer

ret

Question 3:  
Implement the following sorting algorithm for a byte array  
  
  
i = N  
FOR i=N-1 times DO  
Find the position k of the largest element among A[1] ….A[i]  
Swap A[k] and A[i]  
i=i-1  
END FOR  
  
Usually algorithms are evaluated on speed by observing how many steps they took to sort a certain data set. For the data set given in question 1 , which algorithm you feel is faster, Bubble sort or select sort? (Hint : See which algorithm does less swapping)



org 100h

.data

N db 5

s db 'Enter elements of array$'

arr db 5 dup(0)

counter dw 4

address dw 0

.code

mov dx,offset s

mov ah,9h

int 21h

lea si,arr

input:

mov ah,2

mov dl,0x0A

int 21h

mov ah,2

mov dl,0x0D

int 21h

mov ah,1

int 21h

sub al,30h

mov dl,al

mov [si],dl

inc si

dec N

jnz input

mov cx,4

mov ah,4

outer:

lea di,arr

mov al,[di]

lea bx,arr

mov N,ah

lea si,arr

add si,counter

mov address,di

inner:

sort:

cmp al,[di+1]

jl greater

jg end1

greater:

mov al,[di+1]

mov address,di

inc address

jmp end1

end1:

inc di

dec N

jnz inner

swap:

mov dl,al

mov bx,address

xchg [si],dl

mov [bx],dl

dec ah

dec counter

loop outer

ret

Select sort is better as it swaps less times than bubble sort. Which puts less pressure on the system. Hence better speed.