```
;QUESTION NUMBER 2

[org 0x0100]

jmp start

arrayunsort: db 5,7,-3,2,66,45,22,90,33,-9




swap: db 0
bubblesort:
        dec cx ; last element not compared
        shl cx, 1 ; turn into byte count
mainloop:
        mov si, 0 ; initialize array index to zero
        mov byte [swap], 0 ; reset swap flag to no swaps
innerloop:
        mov ax, [bx+si] ; load number in ax
        cmp ax, [bx+si+2] ; compare with next number
        jbe noswap ; no swap if already in order

        mov dx, [bx+si+2] ; load second element in dx
        mov [bx+si], dx ; store first number in second
        mov [bx+si+2], ax ; store second number in first
        mov byte [swap], 1 ; flag that a swap has been done
noswap:
        add si, 2 ; advance si to next index
        cmp si, cx ; are we at last index
        jne innerloop ; if not compare next two

        cmp byte [swap], 1 ; check if a swap has been done
        je mainloop ; if yes make another pass
        ret ; go back to where we came from
clrscr:
       mov ax, 0xb800 ; load video base in ax
       mov es, ax ; point es to video base
       mov di, 0 ; point di to top left column
nextchar:
       mov word [es:di], 0x0720 ; clear next char on screen
       add di, 2 ; move to next screen location
       cmp di, 4000 ; has the whole screen cleared
       jne nextchar ; if no clear next position

printnum: push bp
 mov bp, sp
 push es
 push ax
 push bx
 push cx
 push dx
 push di
 mov ax, 0xb800
 mov es, ax ; point es to video base
```

```
 mov ax, [bp+4] ; load number in ax
 mov bx, 10 ; use base 10 for division
 mov cx, 0 ; initialize count of digits

nextdigit:
        mov dx, 0 ; zero upper half of dividend
        div bx ; divide by 10
        add dl, 0x30 ; convert digit into ascii value
        push dx ; save ascii value on stack
        inc cx ; increment count of values
        cmp ax, 0 ; is the quotient zero
        jnz nextdigit ; if no divide it again
        mov di, 0 ; point di to top left column


nextpos: pop dx ; remove a digit from the stack
       mov dh, 0x07 ; use normal attribute
       mov [es:di], dx ; print char on screen
       add di, 2 ; move to next screen location
       loop nextpos ; repeat for all digits on stack
 pop di
 pop dx
 pop cx
 pop bx
 pop ax
 pop es
 pop bp
 ret 2




start:
       mov bx, arrayunsort ; send start of array in bx
       mov cx, 10 ; send count of elements in cx
       call bubblesort ; call our subroutine
       call clrscr ; call the clrscr subroutine
       looper:
       mov di, 0
       mov ax, [arrayunsort+di]
       add di ,2
       push ax ; place number on stack
       call printnum ; call the printnum subroutine
       cmp di , 20
       jne looper

       mov ax, 0x4c00 ; terminate program
       int 0x21
```

```
;QUESTION NUMBER 1
[org 0x0100]
jmp start
clrsrc:
mov ax, 0xb800 ; load video base in ax
 mov es, ax ; point es to video base
 mov di, 0 ; point di to top left column
nextchar: mov word [es:di], 0x0720 ; clear next char on screen
 add di, 2 ; move to next screen location
 cmp di, 4000 ; has the whole screen cleared
 jne nextchar ; if no clear next position



;left scrolling

scrollleft:
mov ax,0xb800
mov es,ax
mov ds,ax
mov cx,1
l00:
push cx
mov di,0
mov si,2
mov cx,25
l11:
push cx
mov cx,80
cld
rep movsw

pop cx
loop l11
```

```
push di
mov di,0
mov cx,22
mov ax,0x0723
l22:
mov [es:di],ax
add di,2

loop l22
pop di
pop cx
loop l00
ret
```

```
 sleep:
push cx
mov cx,0xFFFF
delay:
loop delay
pop cx
ret

scrollright:
mov ax,0xb800
mov es,ax
mov ds,ax
mov cx,1 ; number of dashes



l0:
push cx


mov si,3996
```

```
mov di,3998
mov cx,25
l1:
push cx


mov cx,80
std
rep movsw

pop cx
loop l1


push di
mov di,0
mov cx,1;number of dashes hahaha
mov ax,0x0728
l2:
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep


mov [es:di],ax
add di,2 ; number of lines hahahaha
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep
call sleep


loop l2
pop di


pop cx
loop l0
```

```
ret
start:
call clrsrc
call scrollright
mov ah,0x1
int 0x21
call scrollleft
mov ah,0x1
int 0x21
mov ax,0x4c00
int 0x21
```