NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Coal

LAB TASK 7

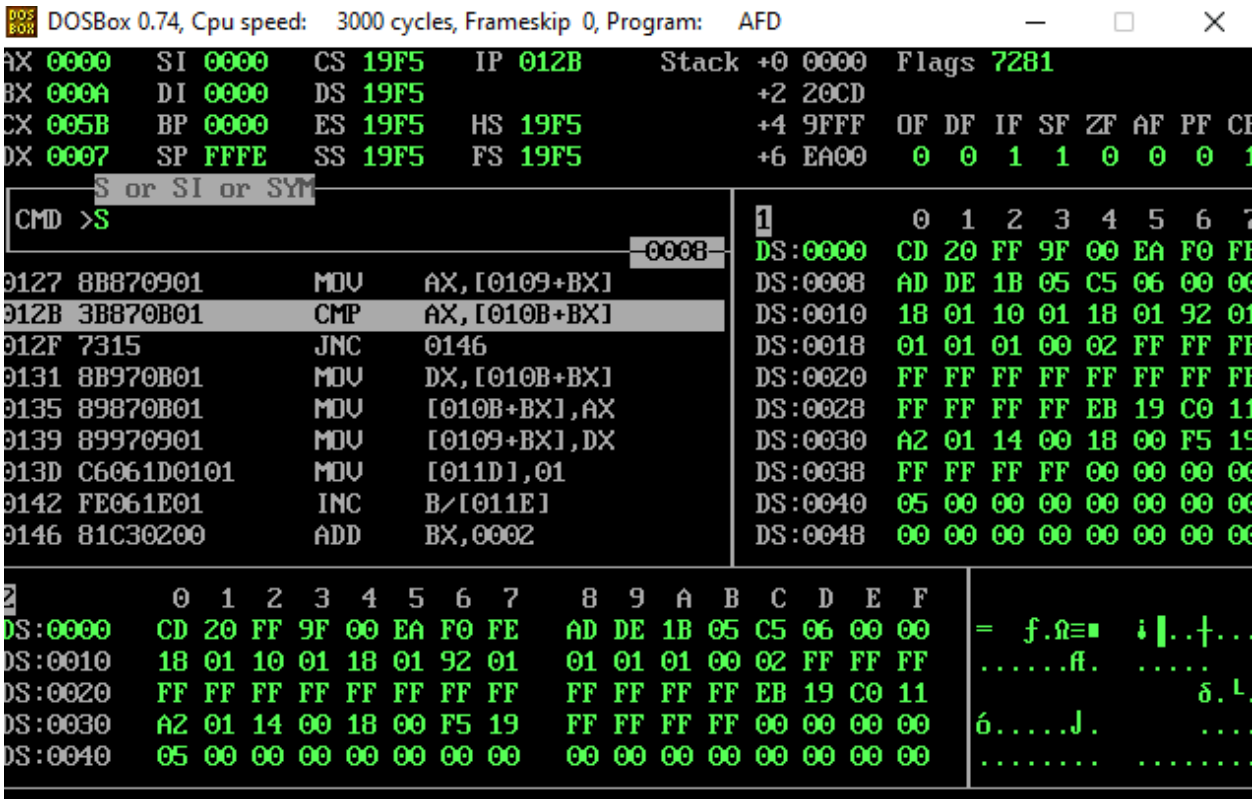
NAME SAJID ISLAM ROLLNO 24p-0745 INSTRUCTOR Muhammad Hassan

Question 1

```
[org 0x0100]


xor ax, ax
xor bx, bx
xor dx, dx

jmp start    ; jump to code start (skip data section)

; --- Data Section ---
data:  dw 0, 3, 24, 5, 6, 7, 8, 2, 1, 1   ; array of 10 words (2 bytes each)
swap:  db 0                               ; flag to track if any swap occurred
count: db 0                               ; to count total swaps


start:
    mov bx, 0              ; start index
    mov byte [swap], 0     ; reset swap flag

loop1:
    mov ax, [data + bx]        ; load current element
    cmp ax, [data + bx + 2]    ; compare with next
    jae noswap                 ; if current >= next → already correct (descending)

    ; --- perform swap ---
    mov dx, [data + bx + 2]    ; next element → DX
    mov [data + bx + 2], ax    ; current → next
    mov [data + bx], dx        ; DX → current (swap done)

    mov byte [swap], 1         ; mark swap happened
    inc byte [count]           ; increase swap counter
```

```
add bx, 2                   ; move to next word
cmp bx, 18                  ; stop after comparing 9 pairs (10 elements)
jne loop1

cmp byte [swap], 1          ; any swap this pass?
je start                    ; yes → another pass


mov ah, 0x4C
int 0x21
```

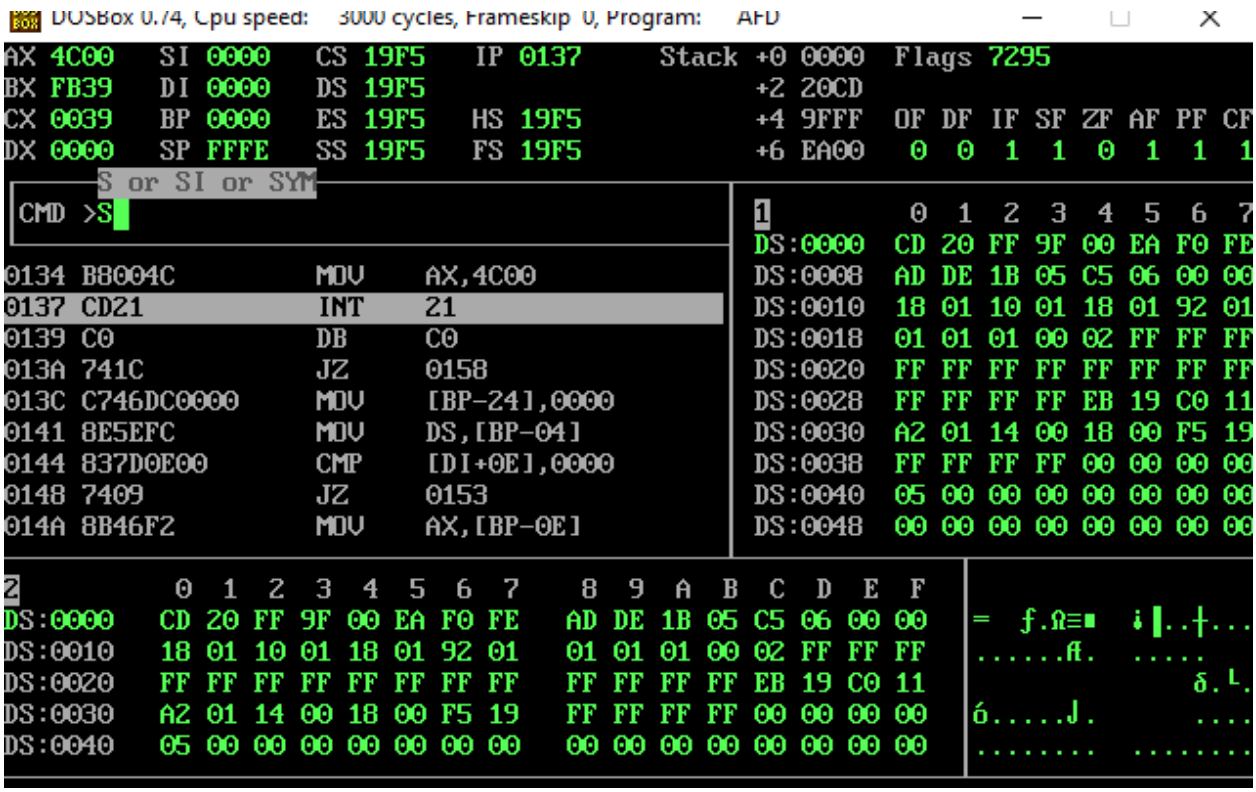
Output



Question 2

```
[org 0x0100]
jmp begin

value1:    dw 0B39Eh          ; initial 16-bit value to work on
top8:      dw 0               ; to store top 8 bits of value1
bottom8:   dw 0               ; to store bottom 8 bits of value1
shiftleft: dw 0               ; to store result after left shift
shiftrite: dw 0               ; to store result after right shift

begin:
    mov ax, [value1]          ; load value1 into AX register

    mov bx, ax                ; copy AX to BX
    shr bx, 8                 ; shift right by 8 bits → isolate top 8 bits
    mov [top8], bx            ; store top 8 bits

    mov bx, ax                ; reload AX into BX
    shl bx, 8                 ; shift left by 8 bits → isolate bottom 8 bits
    mov [bottom8], bx         ; store bottom 8 bits

    mov bx, ax                ; reload AX into BX
    shl bx, 4                 ; shift left by 4 bits → multiply by 16
    mov [shiftleft], bx       ; store left-shifted value

    mov bx, ax                ; reload AX into BX
    sar bx, 4                 ; arithmetic right shift by 4 bits → divide by 16 (sign preserved)
    mov [shiftrite], bx       ; store right-shifted value

    mov ax, 0x4C00            ; terminate program
    int 0x21
```

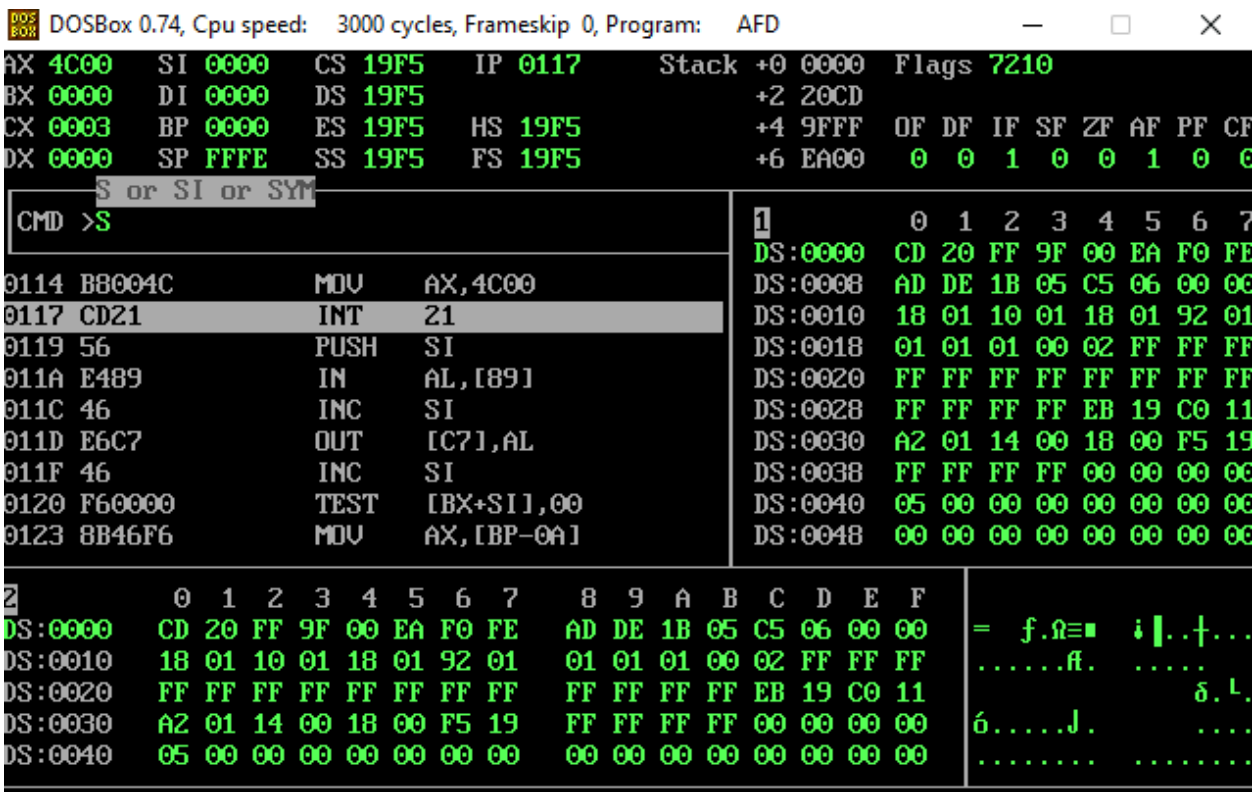
Output



Question 2 part II

```
[org 0x0100]
jmp main

bitmask: dw 0

main:
    mov ax, 1
    mov cl, 3

    shl ax, cl
    mov [bitmask], ax

    shr ax, cl
    mov [bitmask], ax

    mov ax, 0x4C00
    int 0x21
```

Output


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD
AX 4C00 SI 0000 CS 19F5 IP 0117 Stack +0 0000 Flags 7210
BX 0000 DI 0000 DS 19F5 +2 20CD
CX 0003 BP 0000 ES 19F5 HS 19F5 +4 9FFF OF DF IF SF ZF AF PF CF
DX 0000 SP FFFE SS 19F5 FS 19F5 +6 EA00 0 0 1 0 0 1 0 0
S or SI or SYM
CMD >S
0114 B8004C MOV AX,4C00
0117 CD21 INT 21
0119 56 PUSH SI
011A E489 IN AL,[89]
011C 46 INC SI
011D E6C7 OUT [C7],AL
011F 46 INC SI
0120 F60000 TEST [BX+SI],00
0123 8B46F6 MOV AX,[BP-0A]
DS:0000 CD 20 FF 9F 00 EA F0 FE
DS:0008 AD DE 1B 05 C5 06 00 00
DS:0010 18 01 10 01 18 01 92 01
DS:0018 01 01 01 00 02 FF FF FF
DS:0020 FF FF FF FF FF FF FF FF
DS:0028 FF FF FF FF EB 19 C0 11
DS:0030 A2 01 14 00 18 00 F5 19
DS:0038 FF FF FF FF 00 00 00 00
DS:0040 05 00 00 00 00 00 00 00
DS:0048 00 00 00 00 00 00 00 00
DS:0000 CD 20 FF 9F 00 EA F0 FE AD DE 1B 05 C5 06 00 00
DS:0010 18 01 10 01 18 01 92 01 01 01 01 00 02 FF FF FF
DS:0020 FF FF FF FF FF FF FF FF FF FF FF FF EB 19 C0 11
DS:0030 A2 01 14 00 18 00 F5 19 FF FF FF FF 00 00 00 00
DS:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Question 3

```
[org 0x0100]
jmp start

arr:        dw 9, 4, 7, 2, 5
count:      dw 5
flag:       db 0

leftRes:    dw 0, 0, 0, 0, 0
rightRes:   dw 0, 0, 0, 0, 0
arithRes:   dw 0, 0, 0, 0, 0

start:
outer_loop:
    mov byte [flag], 0
    mov bx, 0

inner_loop:
    mov ax, [arr + bx]
    cmp ax, [arr + bx + 2]
    jbe skip_swap

    mov dx, [arr + bx + 2]
    mov [arr + bx + 2], ax
    mov [arr + bx], dx
    mov byte [flag], 1

skip_swap:
    add bx, 2
    cmp bx, 8
    jne inner_loop

    cmp byte [flag], 1
```

```
    je outer_loop

    mov cx, [count]
    mov si, 0
    mov di, 0

shift_loop:
    mov ax, [arr + si]
    mov cl, byte di
    shl ax, cl
    mov [leftRes + si], ax

    mov ax, [arr + si]
    shr ax, 3
    mov [rightRes + si], ax

    mov ax, [arr + si]
    sar ax, 1
    mov [arithRes + si], ax

    add si, 2
    inc di
    loop shift_loop

    mov ax, 0x4C00
    int 0x21
```

What it do

This program first sorts the array in ascending order using bubble sort, then for each element it performs left, logical right, and arithmetic right bit shifts, storing the results in separate arrays before ending.