A picture containing text

Description automatically generated

**Name: Saad Rehman  
Student ID: 21F-9640  
Section: 3A**  
**Degree: BS-CS  
Teacher Name: Sir Abdul Qadeer Bilal  
Course: EL-2003 (COAL)  
Lab Number: 08  
……………………………………………………………………………**

**Note: Perform each Question on AFD Debugger.**

**Qno.1**

Bubble sort subroutine called twice

data: dw 60, 55, 45, 50, 40, 35, 25, 30, 10, 0

data2: dw 328, 329, 898, 8923, 8293, 2345, 10, 877, 355, 98

dw 888, 533, 2000, 1020, 30, 200, 761, 167, 90, 5

**Answer**

[org 0x0100]

jmp start

data: dw 60, 55, 45, 50, 40, 35, 25, 30, 10, 0

data2: dw 328, 329, 898, 8923, 8293, 2345, 10, 877, 355, 98

dw 888, 533, 2000, 1020, 30, 200, 761, 167, 90, 5

swap: db 0

bubblesort: dec cx ; working as a counter, decrement the value of cx one by one

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] ; ; copy 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 ; it wil get to the main as c++ from where we started to call function

start: mov bx, data ; send start of array in bx

mov cx, 10 ; send count of elements in cx

call bubblesort ; 1st call of our subroutine

mov bx, data2 ; send start of array in bx

mov cx, 20 ; send count of elements in cx

call bubblesort ; 2nd call of our subroutine again

mov ax, 0x4c00 ; terminate program

int 0x21

A screenshot of a computer

Description automatically generated with medium confidence

**Qno.2**

Bubble sort subroutine taking parameters from stack

data: dw 88,47,0,-5,83,10,29,-30,2,64

data2: dw 328, 329, 898, 8923, 8293, 2345, 10, 877, 355, 98

dw 1024,88,351,743,994,125,753,852,1,81

**Answer**

[org 0x0100]

jmp start

data: dw 88,47,0,-5,83,10,29,-30,2,64

data2: dw 328, 329, 898, 8923, 8293, 2345, 10, 877, 355, 98

dw 1024,88,351,743,994,125,753,852,1,81

swapflag: db 0

bubblesort:

push bp ; save old value of bp

mov bp, sp ; make bp our reference point

push ax ; save old value of ax

push bx ; save old value of bx

push cx ; save old value of cx

push si ; save old value of si

mov bx, [bp+6] ; load start of array in bx

mov cx, [bp+4] ; load count of elements in cx

dec cx ; last element not compared

shl cx, 1 ; turn into byte count

mainloop:

mov si, 0 ; initialize array index to zero

mov byte [swapflag], 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

jl noswap ; no swap if already in order

xchg ax, [bx+si+2] ; exchange ax with second number

mov [bx+si], ax ; store second number in first

mov byte [swapflag], 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 [swapflag], 1 ; check if a swap has been done

je mainloop ; if yes make another pass

pop si ; restore old value of si

pop cx ; restore old value of cx

pop bx ; restore old value of bx

pop ax ; restore old value of ax

pop bp ; restore old value of bp

ret 4 ; go back and remove two params

start:

mov ax, data

push ax ; place start of array on stack

mov ax, 10

push ax ; place element count on stack

call bubblesort ; call our subroutine

mov ax, data2

push ax ; place start of array on stack

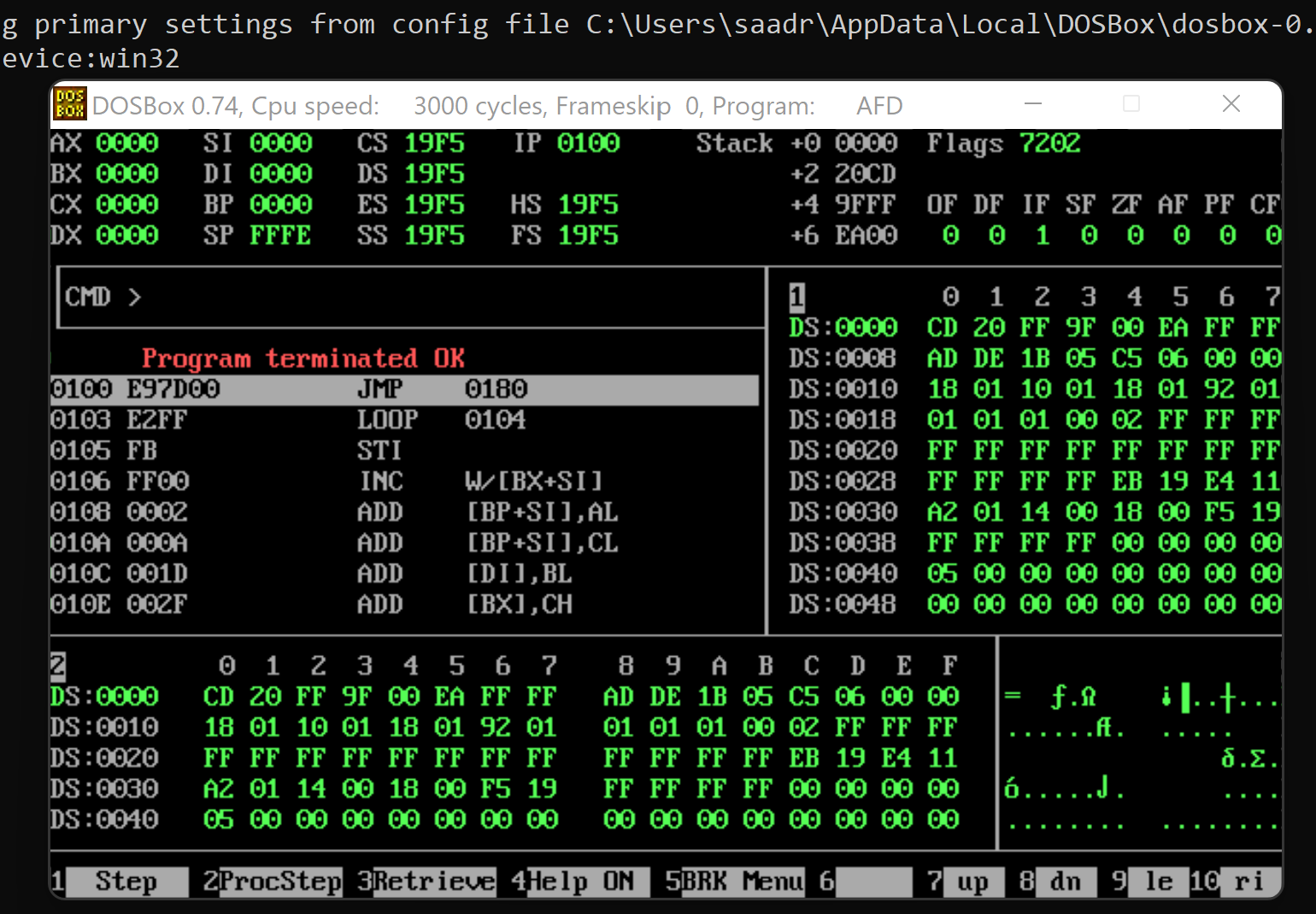
mov ax, 20

push ax ; place element count on stack

call bubblesort ; call our subroutine again

mov ax, 0x4c00 ; terminate program

int 0x21



**Qno.3**

Implement extended multiplication of the following numbers:

2430 and 674.

**Answer**

[org 0x0100]

jmp start

multiplicand: dd 2430 ; 16bit multiplicand 32bit space

multiplier: dw 674 ; 16bit multiplier

result: dd 0 ; 32bit result

start: mov cl, 16 ; initialize bit count to 16

mov dx, [multiplier] ; load multiplier in dx

checkbit:

shr dx, 1 ; move right most bit in carry

jnc skip ; skip addition if bit is zero

mov ax, [multiplicand]

add [result], ax ; add less significant word

mov ax, [multiplicand+2]

adc [result+2], ax ; add more significant word

skip:

shl word [multiplicand], 1

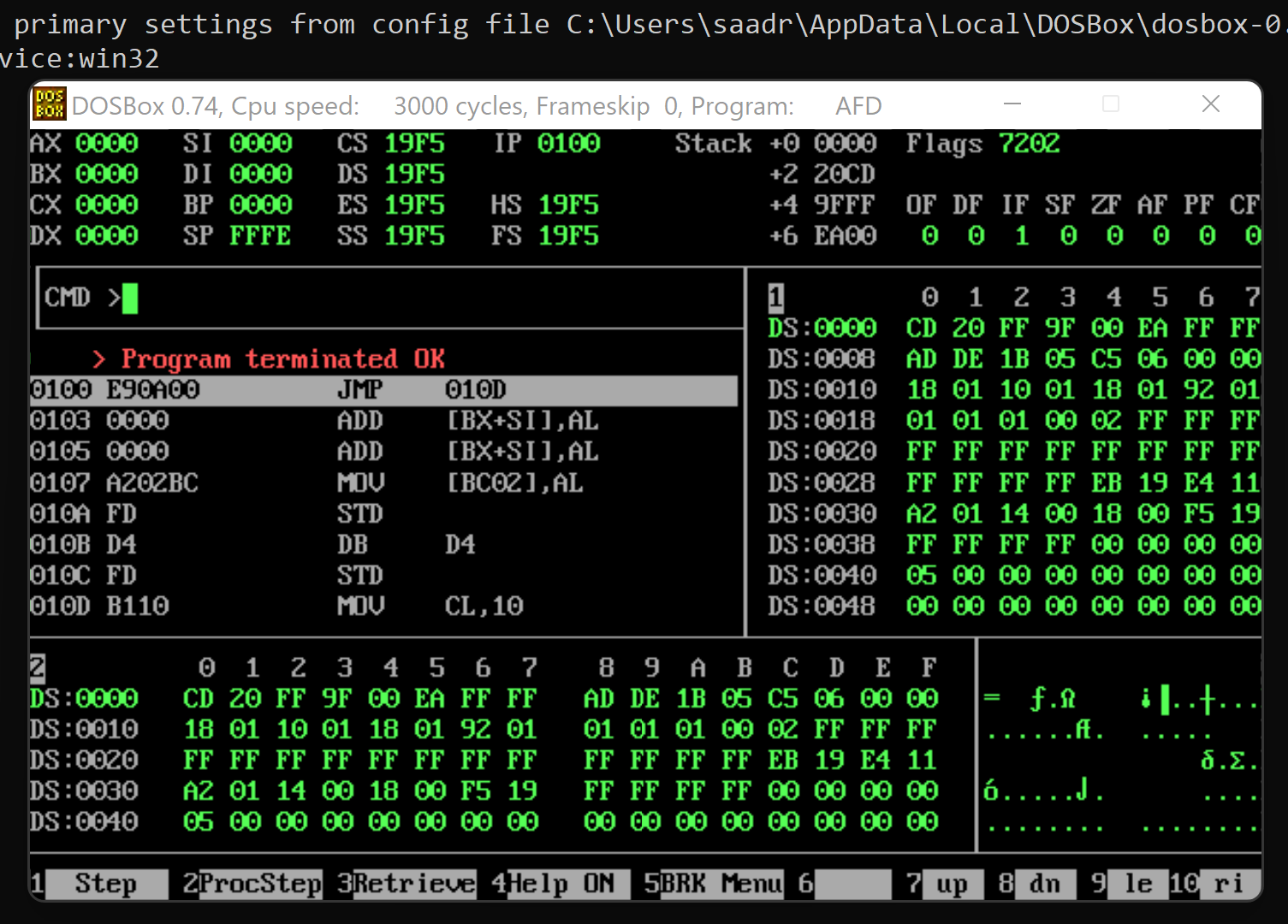
rcl word [multiplicand+2], 1 ; shift multiplicand left

dec cl ; decrement bit count

jnz checkbit ; repeat if bits left

mov ax, 0x4c00 ; terminate program

int 0x21

****

**The End**