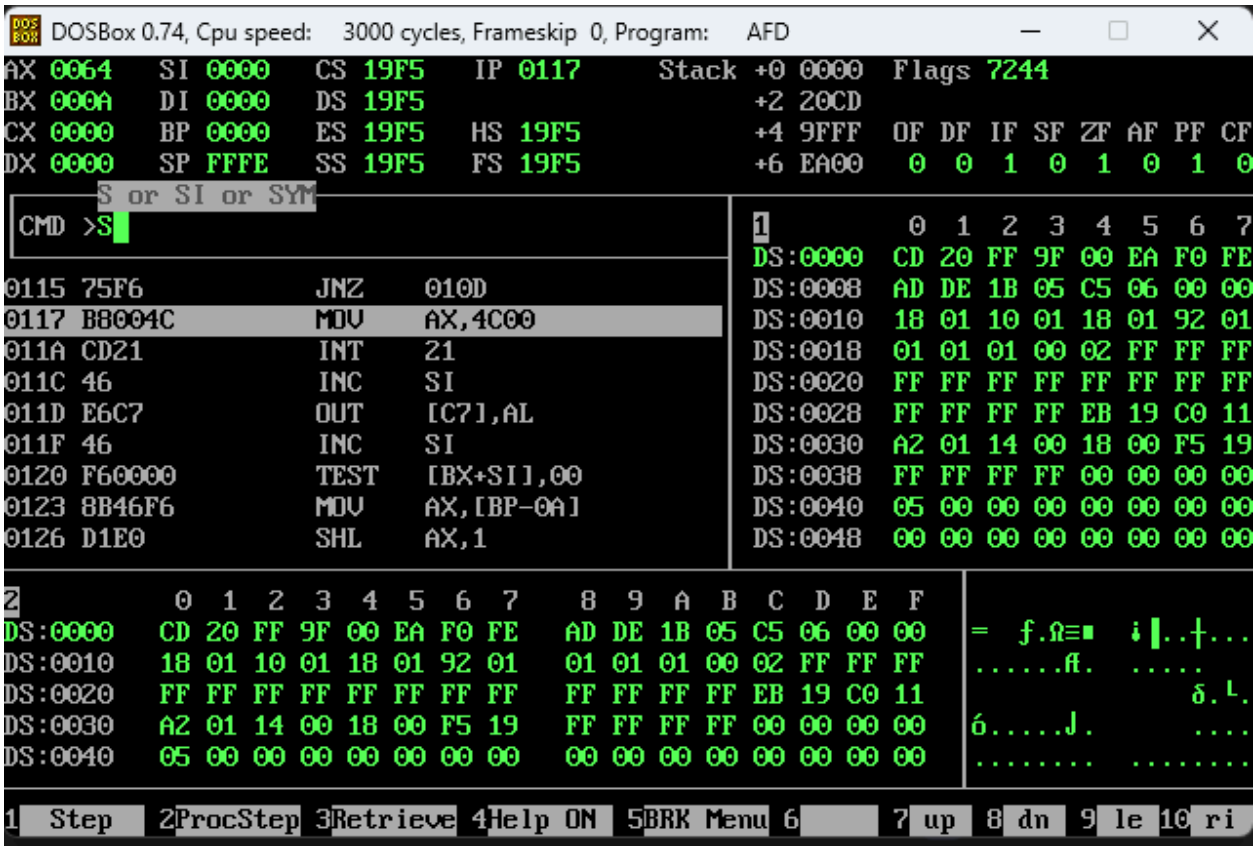
ACTIVITY 1: Give the value of the zero flag, the carry flag, the sign flag, and the overflow flag after each of the following instructions:

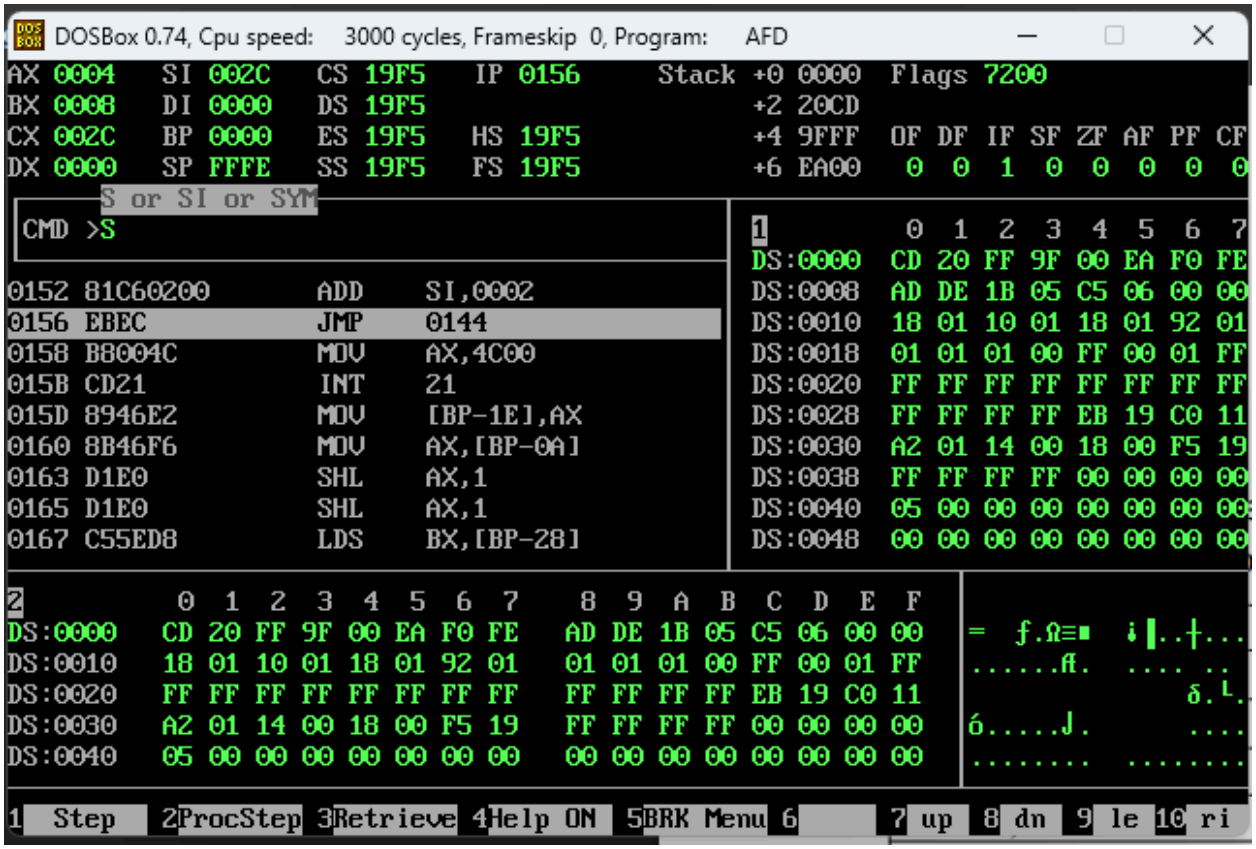
	ZF	CF	SF	OF
mov ax, 0x1254	-	-	-	-
mov bx, 0x0FFF	-	-	-	-
add ax, 0xEDAB	0	0	1	0
add ax, bx	0	1	0	0
add bx, 0xF001	1	1	0	0

ACTIVITY 2: Write a program which calculates the square of a number in memory variable. Display the result in accumulator (AX).



```
[org 0x0100]

jmp start

num dw 10

start:
        mov bx, [num]
        mov cx, [num]

l1:
        add ax, [num]
        sub cx, 1
        jnz l1

mov ax, 0x4c00
int 0x21
```

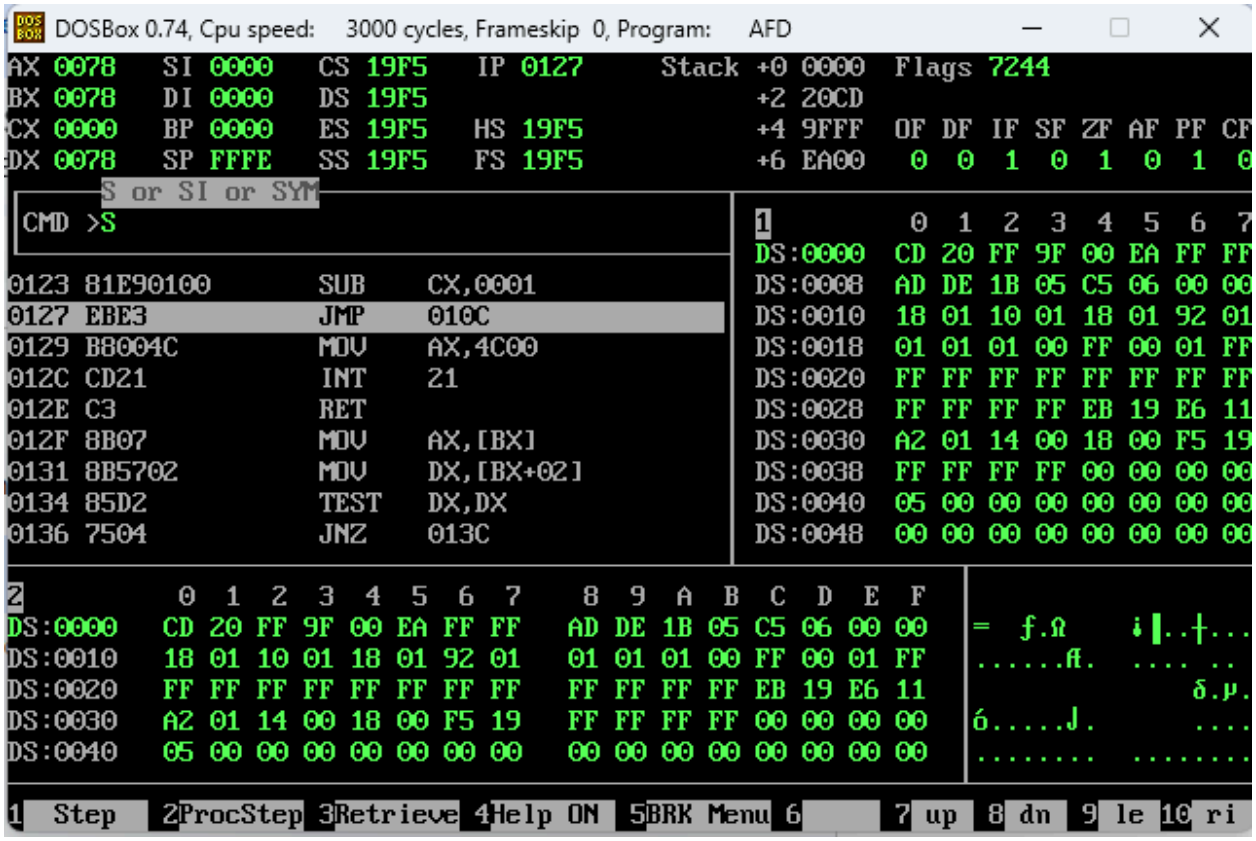
ACTIVITY 3: Write a program which finds the frequency of a specific number from the given array. array: dw 1, 9, 9,9, 8, 8,8, 8, 8,8, 1, 1, 9, 9, 8, 8, 8, 8, 1, 9, 8, 8




```
[org 0x0100]

jmp start

array: dw 1, 9, 9,9, 8, 8,8, 8, 8,8, 1, 1, 9, 9, 8, 8, 8, 8, 1, 9, 8, 8
find dw 1

incr:
        add ax, 1
        add si, 2
        jmp l1
start:
        mov cx, 44
        mov si, 0
        mov ax, 0
l1:
        cmp cx, si
        je term
        mov bx, [array+si]
```

```
        cmp bx, [find]
        je incr
        add si, 2
        jmp l1

term:

mov ax, 0x4c00
int 0x21
```

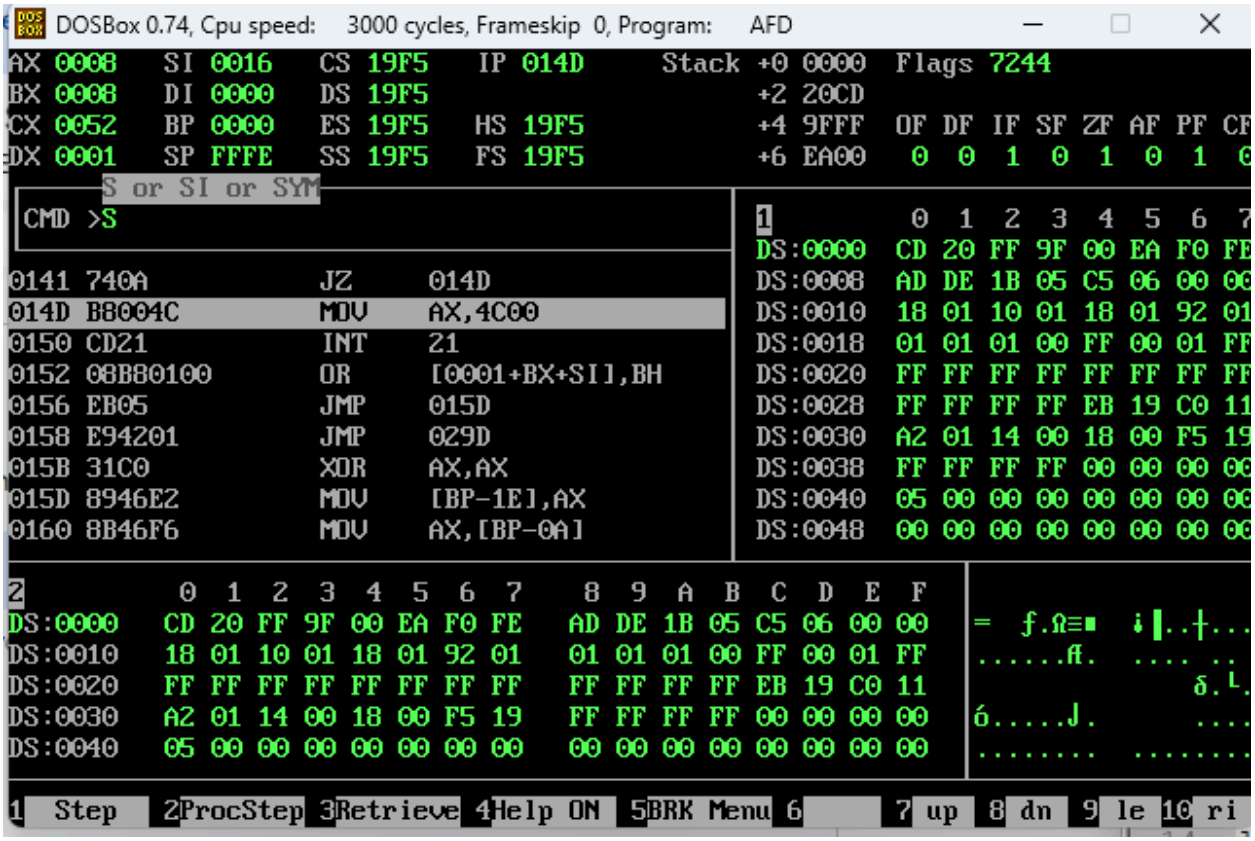
ACTIVITY 4: Write a program which finds the factorial of a given integer without the use of MUL command.




```
[org 0x0100]

jmp start
num dw 5 ; Factorial should be 120 (0078h)

start:
        mov ax, 1
        mov cx, [num]

Mult:
        cmp cx, 0
        je term
        mov bx, ax
        mov dx, 0
        mov si, cx
Adding:
        add dx, bx
        sub si, 1
        jnz Adding
```

```
        mov ax, dx
        sub cx, 1
        jmp Mult
term:

mov ax, 0x4c00
int 0x21
```

ACTIVITY 5: Write a program which determines smallest number from the given array. array: dw 111, 999, 888, 888, 11, 99, 88, 88, 1, 9, 8, 8



[org 0x0100]

jmp start
array: dw 111, 999, 888, 888, 11, 99, 88, 88, 1, 9, 8, 8

start:
	mov ax, [array]
	mov dx, ax
	mov bx, [array+2]
	mov si, 0
l11:
	mov dx, ax
	jmp l2
l12:
	mov dx, bx
	jmp l2
comp:
	cmp ax, dx
	jl l11
	cmp bx, dx

```
        jl l12
l2:
        add si, 2
        cmp si, 22
        je term
        mov ax, [array+si]
        mov bx, [array+si+2]
        jmp comp
term:

mov ax, 0x4c00
int 0x21
```