NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

TASK 1:

Assembly Code:

lb8q1.asm

```
1    [org 0x0100]
2    jmp start
3        num1: db 10111010b ; 8-bit number
4    start:
5        mov al, [num1]
6        rol al, 3
7        ror al, 3
8        rcl al, 3
9        rcr al, 3
10   mov ax, 0x4c00
11   int 0x21
```

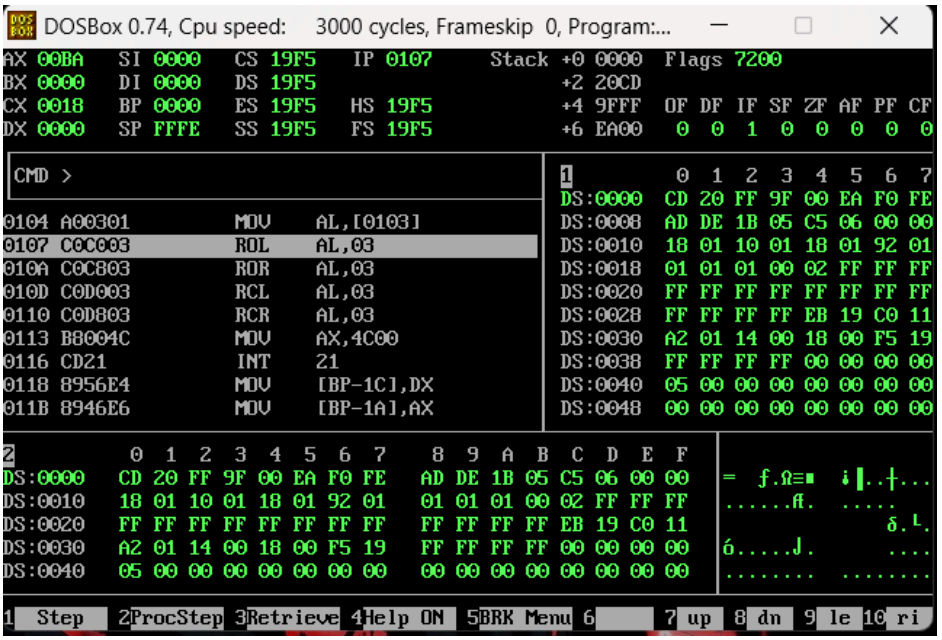
- **num1** will store the **8**-bit value.
- **rol** function will rotate the number **3**-bit to the left.
- **ror** function will rotate the number **3**-bit to right changing to original number.
- Similarly, **rcl** will rotate the number to **3**-bit left and add carry every time at the **LSB**.
- Then the **rcr** will rotate the number to **3**-bit right and add carry every time at the **MSB**.

NATIONAL UNIVERSITY
of Computer & Emerging Sciences

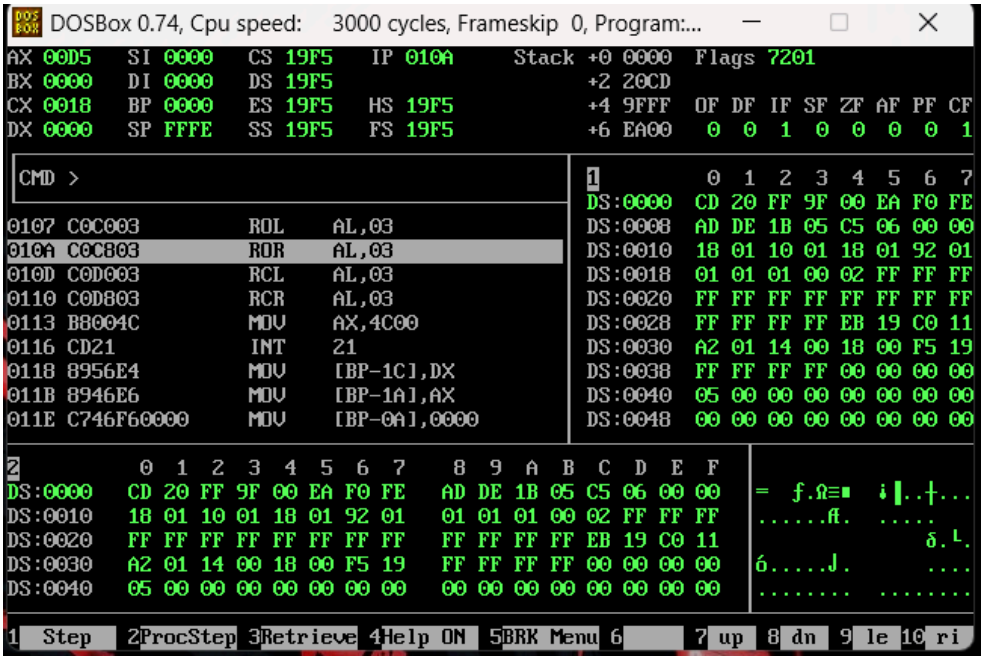
COAL LABTASK 8

SAJID ISLAM
24P-0745

Original:



ROL:

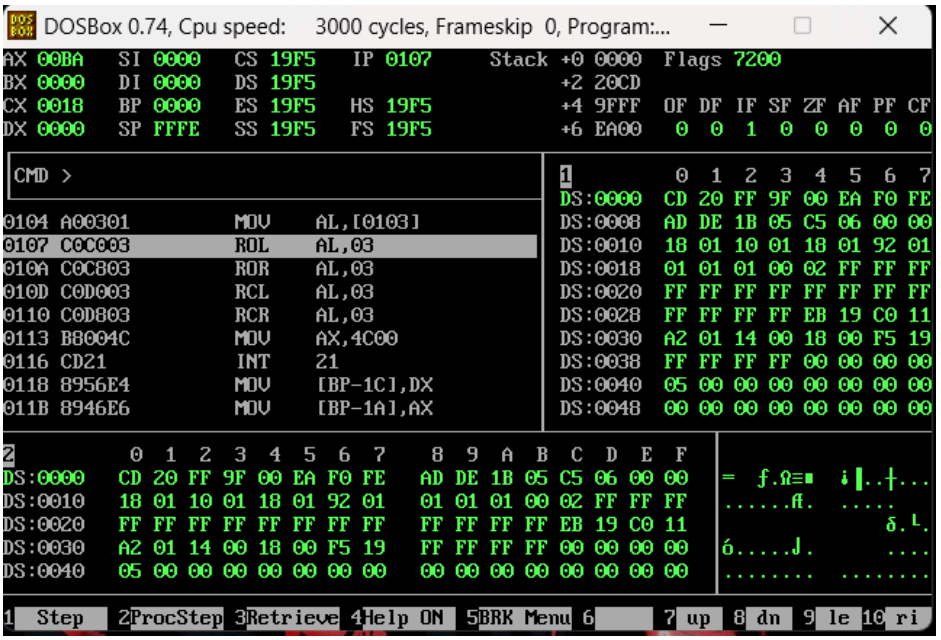


NATIONAL UNIVERSITY
of Computer & Emerging Sciences

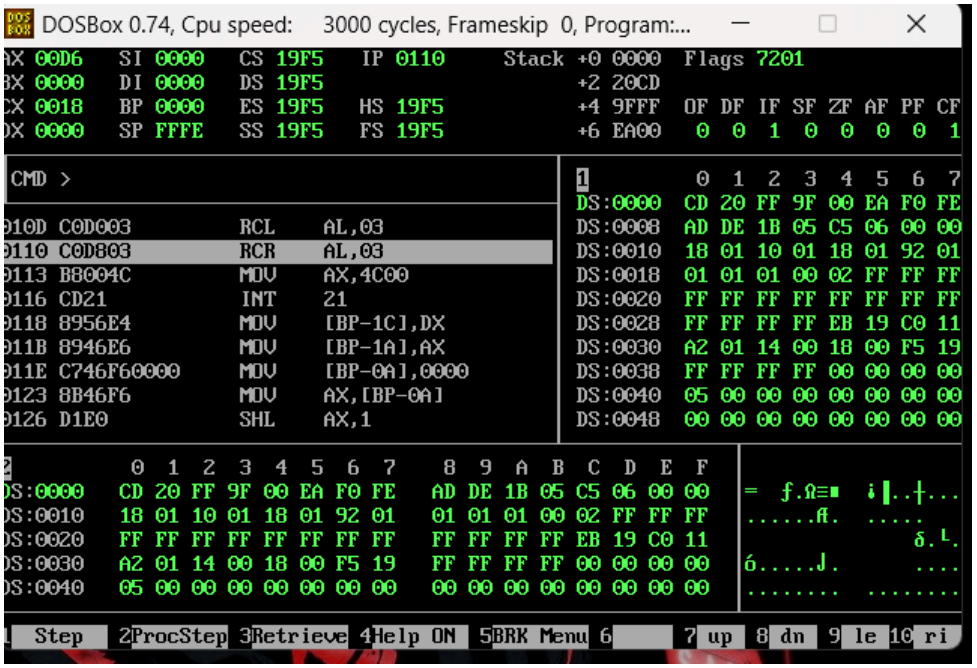
COAL LABTASK 8

SAJID ISLAM
24P-0745

ROR:



RCL:



NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

TASK 2:

Assembly Code:

lb8q2.asm

```
1   [org 0x0100]
2   jmp start
3       multiplicand: db 12 ; 4-bit number
4       multiplier: db 6 ; 4-bit
5       result: db 0 ; 8-bit result
6   start:
7       mov cl, 5 ; how many times we need to run the loop
8       mov al, [multiplicand]
9       mov bl, [ multiplier]
10  bit_checker:
11      shr bl, 1
12      jnc move
13      add [result], bl ; only add if CF IS 1
14  move:
15      shl al, 1
16      dec cl
17  jnz bit_checker
18
19  mov ax, 0x4c00
20  int 0x21
```

- The multiplicand and multiplier are both 4-bit values, producing an 8-bit result.
- We load 5 into CL because one addition is already performed during initialization.
- The multiplicand and multiplier are moved into AL and BL, respectively.
- In the bit_checker loop, the multiplier is rotated right, and the carry flag is checked.
- If the carry flag is set, the multiplicand is added to the result;

NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

- If it is clear, the program jumps to the move loop, where the multiplicand is shifted left before returning to bit_checker for the next iteration.

TASK 3(1):

Assembly Code:

lb8q31.asm

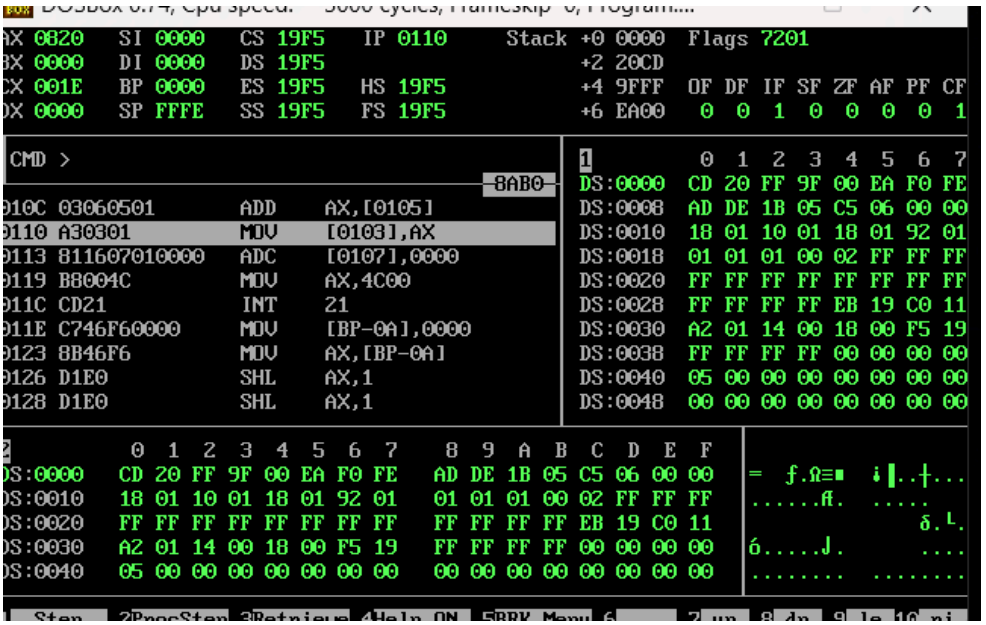
```
1   [org 0x0100]
2   jmp start
3       num1: dw 8AB0h
4       num2: dw 7D70h
5       high: dw 0000h
6   start:
7       mov ax, [num1]
8       add ax, [num2]        ; AX = 8AB0h + 7D70h = 10820h → AX=0820h, CF=1
9       mov word [num1], ax
10      adc word [high], 0    ; Add carry (1) to next word → high = 0001h
11
12  mov ax, 0x4c00
13  int 0x21
14
```

The values are stored in num1 and num2.
When num1 is loaded into AX and added to num2, an overflow (OF) occurs because the result exceeds the 16-bit limit.
The extra (17th) bit generated from the addition is stored in the Carry Flag (CF).
Afterward, the result in AX is moved back to num1, and the instruction ***adc word [high], 0*** adds the carry to high, effectively performing high = high + CF.

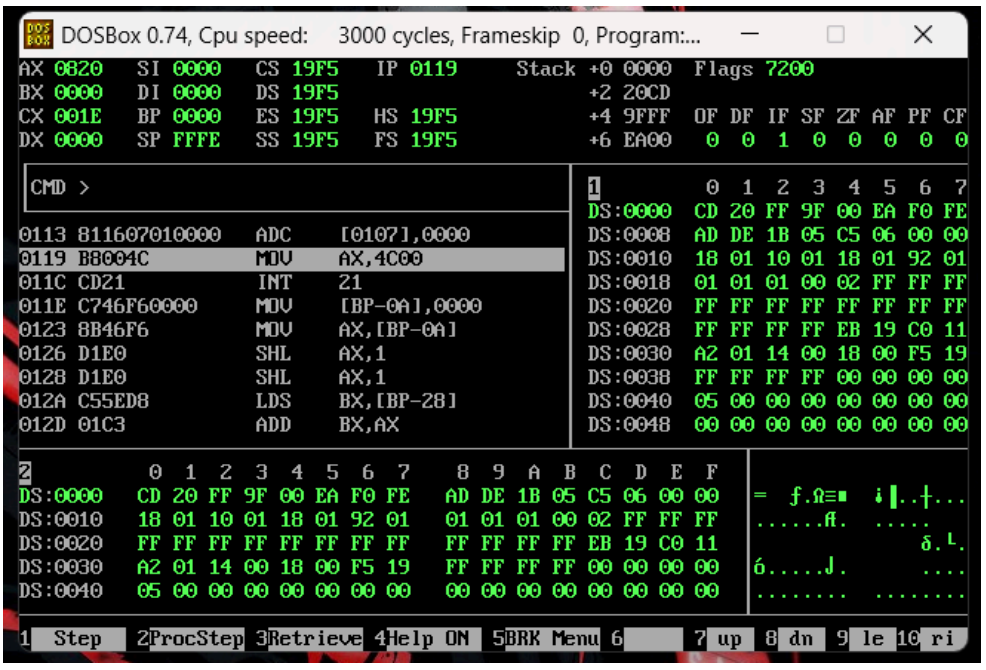
NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745



When CF is Added:



NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

TASK 3(2):

Assembly Code:

lb8q32.asm

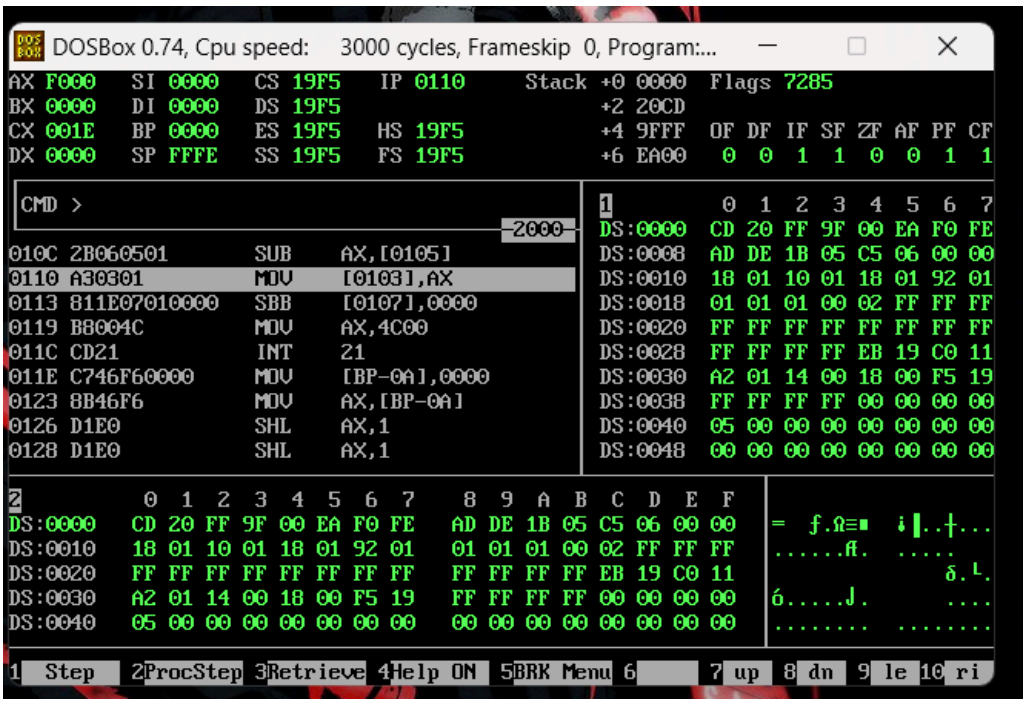
```
1    [org 0x0100]
2    jmp start
3        num1: dw 2000h
4        num2: dw 3000h
5        high: dw 0001h
6    start:
7        mov ax, [num1]
8        sub ax, [num2]
9        mov word [num1], ax
10       sbb word [high], 0   ; Subtract with borrow
11
12
13   mov ax, 0x4c00
14   int 0x21
15
```

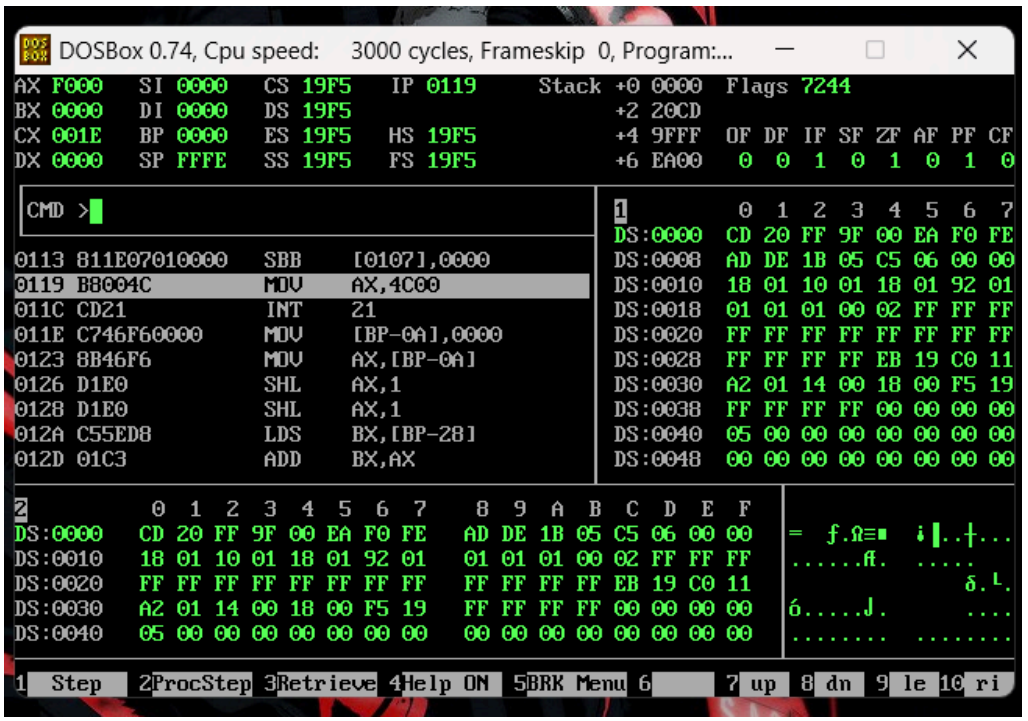
- Values are stored in **num1** and **num2**.
- When **num1** is moved to **ax** and both numbers are subtracted, borrow is taken.
- We move ax to **num1** and the **borrow** is subtracted to high by using ***sbb word [high],0*** which means **high= high - borrow.**

NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745





NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

TASK 4:

Assembly Code:

lb8q4.asm

```
1  [org 0x0100]
2      jmp start
3      val1: db 10101010b
4      val2: db 11001100b
5      result: db 0
6  start:
7      mov al, [val1]
8      mov bl, [val2]
9      and al, bl
10
11     mov [result], al     ; Store AND result
12     mov al, [val1]
13     or al, bl
14
15     mov [result], al     ; Store OR result
16     mov al, [val1]
17     xor al, bl
18
19     mov [result], al     ; Store XOR result
20     mov al, [val1]
21     not al
22
23     mov [result], al     ; Store NOT result
24
25 mov ax, 0x4c00
26 int 0x21
27
```

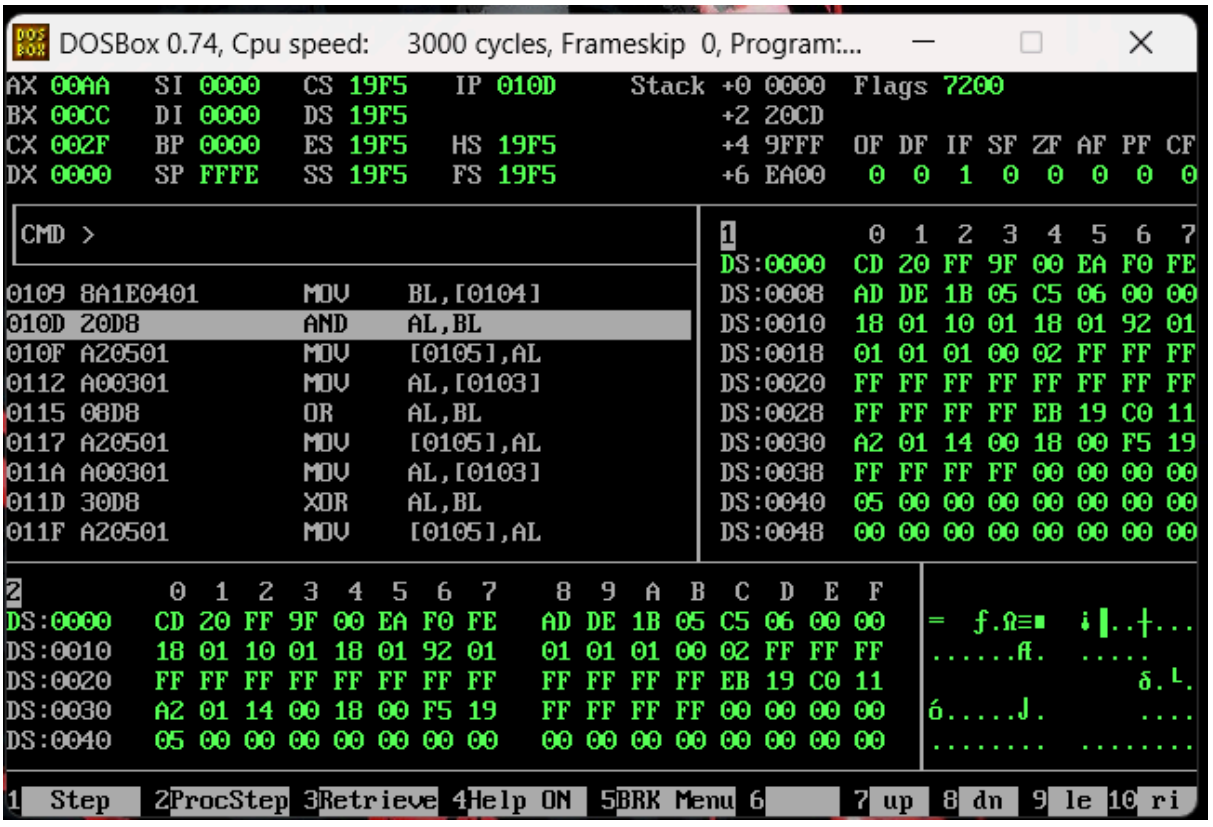
NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

- val1 and val2 are stored in al and bl.
- When ***and al ,bl*** is run it will check if both are 1 then output 1 else so in our case **10001000** will be stored in the result.
- When ***or al ,bl*** is run it will check if one bit is 1 then output 1 so **10001000** will be stored in the result.
- ***xor al,bl*** checks if opposite bits then 1 else 0 so the result will store **01100110**
- ***not al*** will just reverse the bits make 1->0 and 0->1 so the output will store **01010101**

Original:

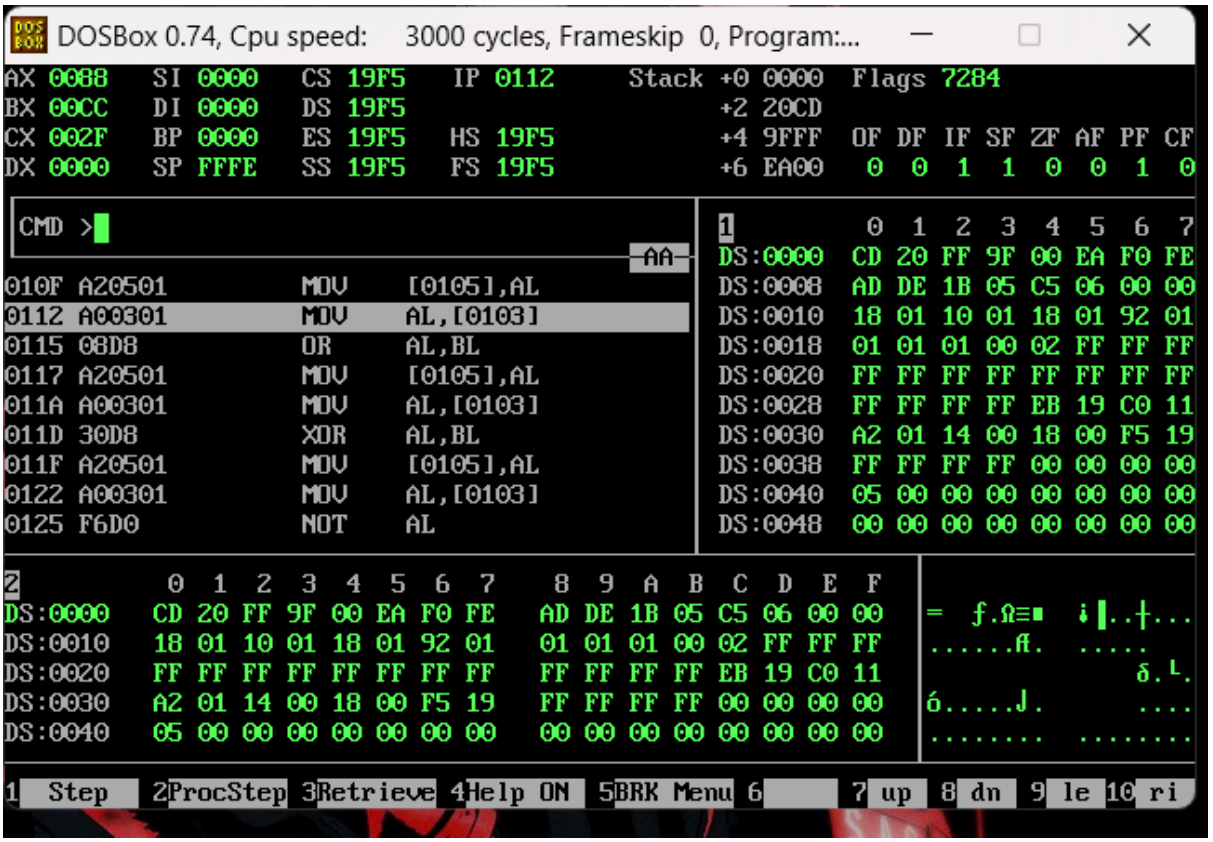


NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

AND:

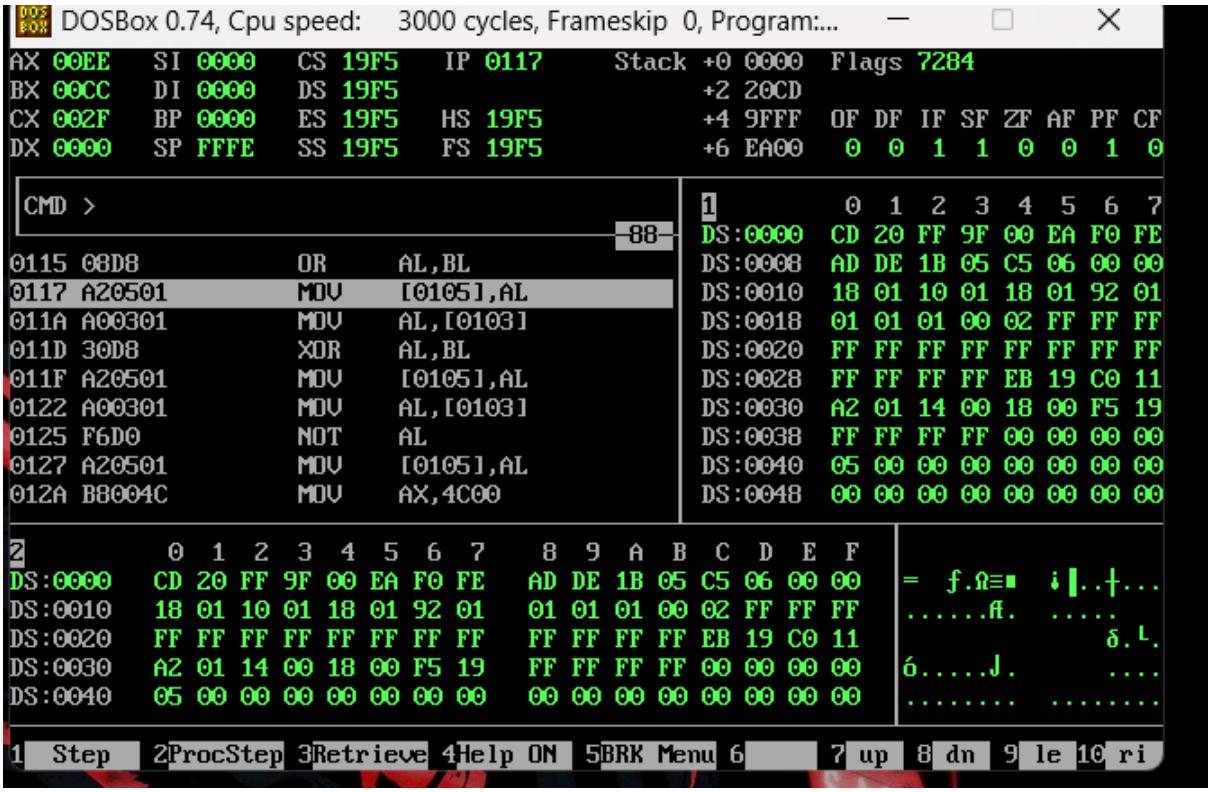


NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

OR:

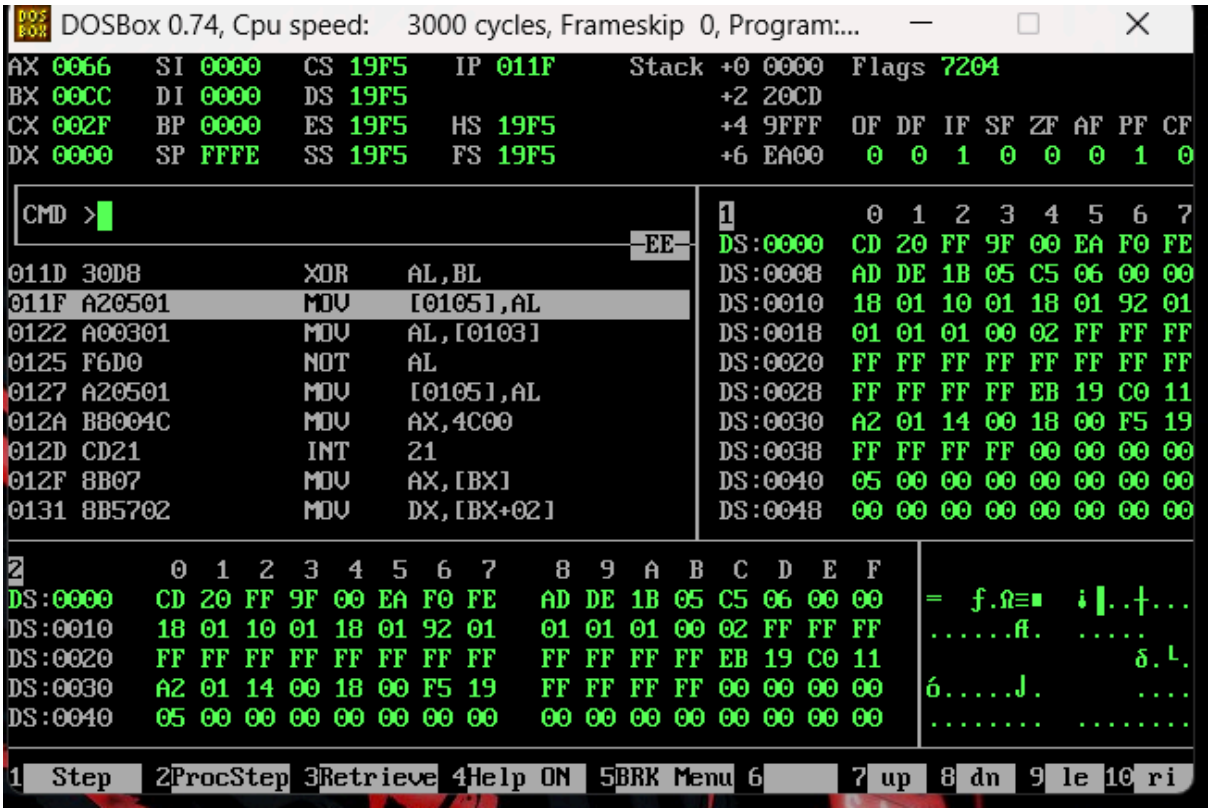


NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

XOR:



NATIONAL UNIVERSITY
of Computer & Emerging Sciences

COAL LABTASK 8

SAJID ISLAM
24P-0745

NOT: