**Name: Khan Arshad Abdulla** **Date – 26/03/2022**
**Roll No: 20CO24**

# Experiment No. 7

**AIM -** Write an assembly language program to display the contents of 16 bit flag register.

## Description:

To display the contents of flag register pushf and popf instruction are used.

Each bit of flag register is then masked off with 1 and all 0's (i.e. 1000 0000 0000 0000BOR 8000h) and based on the result of masking either 0 (30h) or 1 (31h) is get displayed on the screen.

Each bit of the above 16 bit number gets shifted in right direction by 1 position before masking to obtain the next bit position of flag register.

This whole procedure gets repeated 16 times.

## Algorithm:

1. Start
2. Initialize data segment through AX register in the DS register.
3. Display the flag bit names as "X X X X O D I T SF ZF x AF X PF X CF "
4. Push the contents of flag register to a stack
5. Pop the contents of stack to register to any 16 bit register (say BX =0000 0100 1000 1001)
6. Move the contents of BX to temporary variable say t
7. Move the 8000h number to AX.(AX 8000h)
8. Move the count as 16(in decimal) to CX register (as 16 bit flag register)
9. Move the contents of temporary variable t to BX.
10. And the contents of BX and AX.
11. If zero flag is set then goto the step no 14 otherwise goto step no. 12
12. Move the 31h to DL register.
13. Make the unconditional jump to a step no. 15
14. Move the 30h to DL register.
15. Preserve the (8000h ) number from AX in t1 temporary variable. (As while displaying 30h or 31 h AH register get modified as 02h function is moved of INT 21h).
16. Display the contents of DL register.
17. Move the contents of t1 to AX register back (As while displaying 30h or 31 h AH register get modified as 02h function is moved of INT 21h).
18. Rotate the contents of AX by 1 positions in right direction.
19. Repeat step no 5 to 17 till count CX reaches to 0.
20. Stop.

## ROR Instruction :

Syntax - **ROR destination, count.**

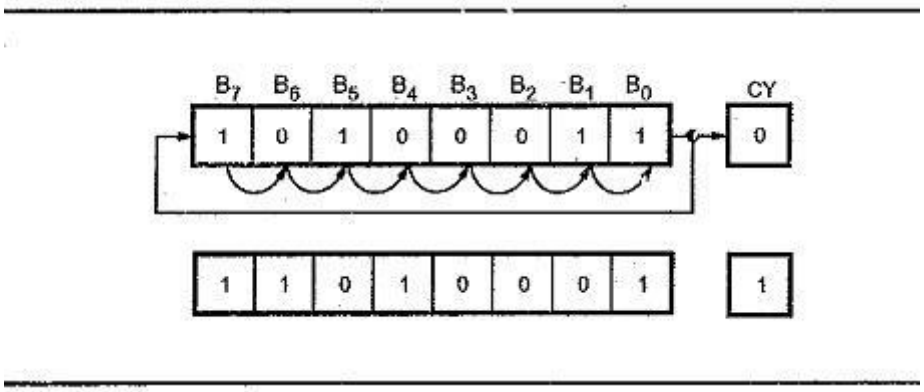
This Rotate Instruction in 8086 with example all bits in a specified byte or word to the right some number of bit positions. LSB is placed as a new MSB and a new CF.

The destination can be a byte or a word. It can be in a register or in a memory location. The number of shifts are indicated by count. If number of shifts required is one, you can place 1 in the count position.

**Name: Khan Arshad Abdulla** **Date – 26/03/2022**
**Roll No: 20CO24**

If number of shifts are greater than 1 then shift count must be loaded in CL register and CL must be placed in the count position of the instruction.

Diagram shows ROR instruction for byte rotation.




**Examples :**

```
ROR  CX,  1          ; Rotated word in CX one bit position
                     ; left, LSB to MSB and CF.
MOV  CL,  03H        ; Load number of bits to rotate in CL.
ROR  BL,  CL         ; Rotate BL three positions.
```

**ALP –**

```
Data Segment
msg  db  0dh,0ah,"-- -- -- -- OF DF IF TF SF ZF -- AF -- PF -- CF $"
newl  db  0dh,0ah,"$"
flag  dw  ?
Data ends
Code Segment
assume  CS:Code , DS:Data
start:
mov ax,Data
mov DS,ax
mov dx,offset msg
mov ah,09h
int 21h
mov dx,offset newl
mov ah,09h
int 21h
cli
stc
std
pushf
pop  bx
mov  flag,bx
mov cx,16
mov  bx,8000h
BACK:
mov  ax,flag
```

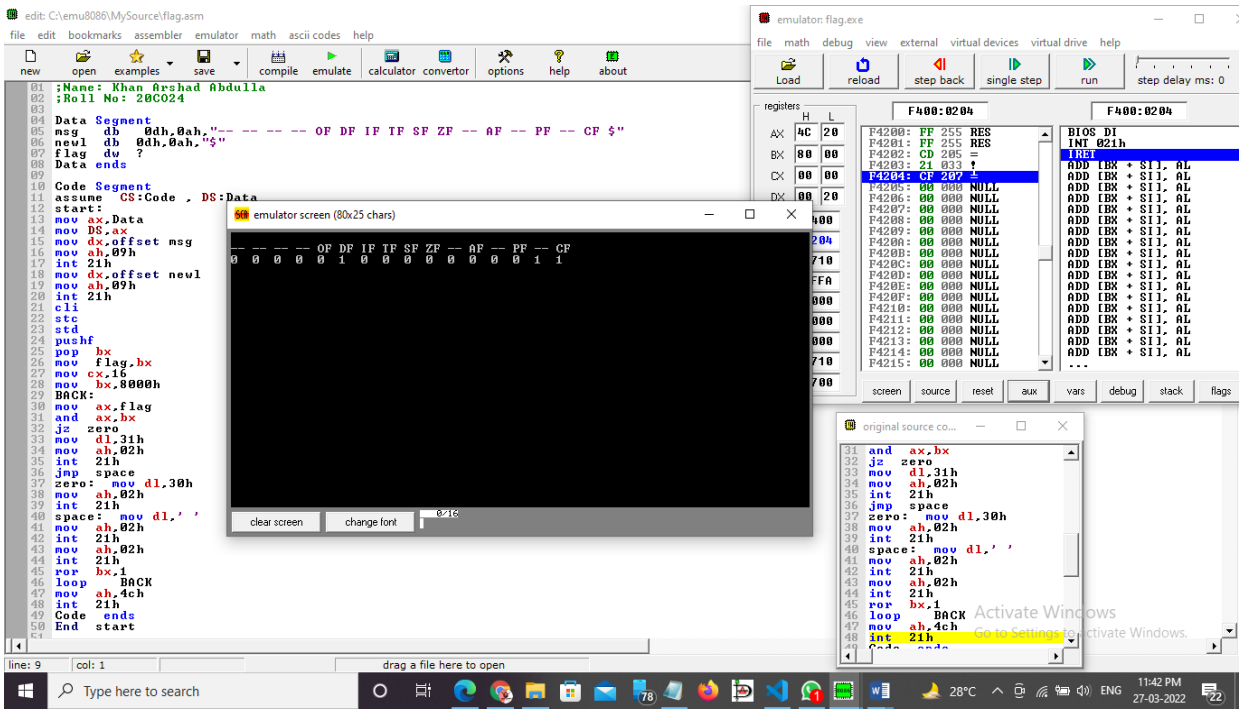
**Name: Khan Arshad Abdulla** **Date – 26/03/2022**
**Roll No: 20CO24**

```
and  ax,bx
jz  zero
mov  dl,31h
mov  ah,02h
int  21h
jmp  space
zero:  mov dl,30h
       mov  ah,02h
      int  21h
space:  mov dl,' '
       mov  ah,02h
      int  21h
      mov  ah,02h
      int  21h
ror  bx,1
loop   BACK
mov  ah,4ch
int  21h
Code  ends
End  start
```

## Procedure –

1. **Launch emu8086 IDE** from menu.
2. **Edit** your program , save as file_name.asm
3. **Compile** your program to check for syntax errors, rectify if any error is present. Save and recompile your program.
4. **Run** to observe output of your program.

## Output –



**Name: Khan Arshad Abdulla** **Date – 26/03/2022**
**Roll No: 20CO24**

**Conclusion –** To display the contents of 16 bit flag register, we use PUSHF and POPF Instructions.

----------end----------