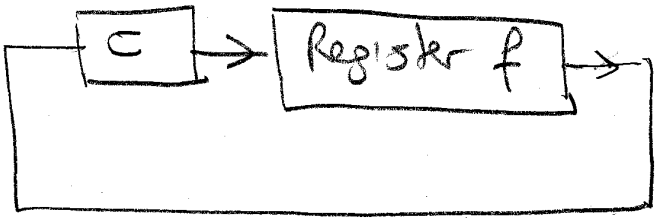
(1)

rrf f,d → Rotate Right f through Carry

↓

The content of register 'f' are rotated one bit to the right through the Carry Flag. If d=0 the result is placed into the W register. If d=1, the result is placed back in register f



rlf f,d → Rotate Left f through Carry

comf f,d → Complement f

↓

The contents of register 'f' are complemented. If d=0, the result is stored in W register. If d=1, " " " " " f register.

(2)

Carry flag in status register

C → Carry Flag is changed after these commands

addwf, addlw, sublw, subwf

OR

Borrow Flag

C: Carry / $\overline{\text{borrow}}$ flag

C = 1 → A carry out from the most significant of the result occured.

C = 0 → No carry from the Most significant bit of the result occured.

(For $\overline{\text{borrow}}$ the polarity is reversed)

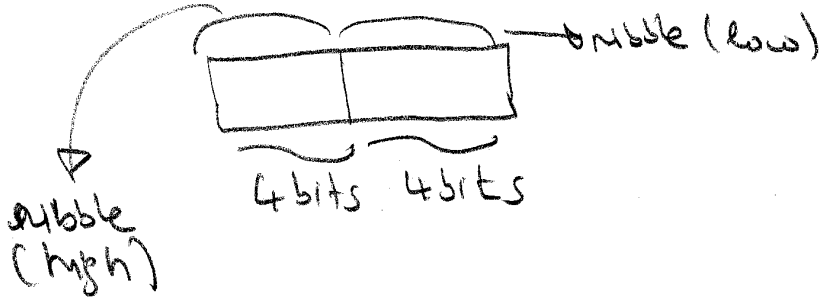
DC → Digit carry / $\overline{\text{borrow}}$ bit

Can be changed after these commands

addwf, addlw, sublw, subwf

DC = 1 if a carry out from low nibble occurs

DC = 1 " no " " " " " " "



Zero flag:

Z = 1 If the result of arithmetic or logical operation is zero

Z = 0 If the result of arithmetic or logical operation is not zero

Ex: Write a program as described below

- If the button connected to RA0 is pressed 10 times, the LED connected to RB0 is turned on.

Sln:

```
list P=16f84A
include "P16f84A.INC"
bsf STATUS, RP0;
clrf TRISB;
bcf STATUS, RP0
clrf PORTB;
clrf Counter;
loop1
loop2 btfsc PORTA,0; → check button press
      goto loop2;
      incf Counter, F; → increment counter if pressed
      movf Counter, W; (w) ← (counter)
      sublw .10; (w) ← 10-(w)
      btfss STATUS, Z; btfss STATUS,2
      goto loop1; → Go for a new check.
      bsf PORTB, 0; loop goto loop
end;
```

Counter equ 0x0C

If the result of previous subtraction is zero then Z = 1