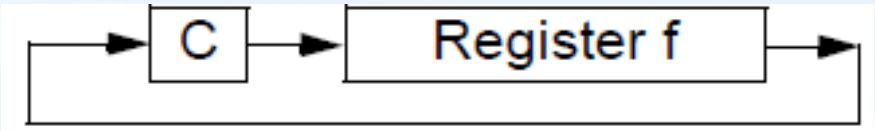
Εντολές PIC16F887

20. rrf <όνομα καταχωρητή>,a
a=f ή a=w
Η εντολή πραγματοποιεί ολίσθηση προς τα δεξιά του καταχωρητή που ακολουθεί μέσω κρατουμένου (Carry) και αν a=f αποθηκεύει το αποτέλεσμα στον καταχωρητή ενώ αν a=w αποθηκεύει το αποτέλεσμα στον W. Επηρεάζει το Carry flag.
π.χ.: Αρχικές τιμές:
Reg1=b'00001011', W=b'00001111', C=0
rrf Reg1,f
τελικές τιμές:
Reg1=b'00000101', W=b'00001111', C=1
Η ολίσθηση προς τα δεξιά έχοντας C=0 ισοδυναμεί με διαίρεση με το 2.



Εντολές PIC16F887

21. rlf <όνομα καταχωρητή>,a
a=f ή a=w
Η εντολή πραγματοποιεί ολίσθηση προς τα αριστερά του καταχωρητή που ακολουθεί μέσω κρατουμένου (Carry) και αν a=f αποθηκεύει το αποτέλεσμα στον καταχωρητή ενώ αν a=w αποθηκεύει το αποτέλεσμα στον W. Επηρεάζει το Carry flag.
π.χ.: Αρχικές τιμές:
Reg1=b'10001011', W=b'00001111', C=0
rlf Reg1,f
τελικές τιμές:
Reg1=b'00010110', W=b'00001111', C=1
Η ολίσθηση προς τα αριστερά έχοντας C=0 ισοδυναμεί με πολλαπλασιασμό με το 2.

Εντολές PIC16F887

22. bcf <όνομα καταχωρητή>,d
d=0 ή 1 ή 2 ή 3 ή 4 ή 5 ή 6 ή 7
Η εντολή πραγματοποιεί μηδενισμό bit(d) από τον καταχωρητή που ακολουθεί.
π.χ.: Αρχική τιμή: Reg1=b'11111111'
bcf Reg1,0
τελική τιμή: Reg1=b'11111110'

23. bsf <όνομα καταχωρητή>,d
d=0 ή 1 ή 2 ή 3 ή 4 ή 5 ή 6 ή 7
Η εντολή ορίζει στο λογικό '1' το bit(d) από τον καταχωρητή που ακολουθεί.
π.χ.: Αρχική τιμή: Reg1=b'00000000'
bsf Reg1,0
τελική τιμή: Reg1=b'00000001'

Εντολές PIC16F887

24. btfsc <όνομα καταχωρητή>,d
d=0 ή 1 ή 2 ή 3 ή 4 ή 5 ή 6 ή 7
Η εντολή πραγματοποιεί παράκαμψη της επόμενη εντολής αν το bit(d) του καταχωρητή που ακολουθεί είναι μηδέν, διαφορετικά δεν γίνεται παράκαμψη της επόμενης εντολής.
π.χ.: Reg1=b'10101010'
btfsc Reg1,0
εντολή 1
εντολή 2

Μετά την εκτέλεση της εντολής btfsc Reg1,0 η επόμενη εντολή που θα εκτελεστεί θα είναι η 'εντολή 2' καθώς το bit0 (Reg1<0>) είναι ίσο με μηδέν.

Εντολές PIC16F887

25. btfss <όνομα καταχωρητή>,d
d=0 ή 1 ή 2 ή 3 ή 4 ή 5 ή 6 ή 7
Η εντολή πραγματοποιεί παράκαμψη της επόμενη εντολής αν το bit(d) του καταχωρητή που ακολουθεί είναι '1', διαφορετικά δεν γίνεται παράκαμψη της επόμενης εντολής.
π.χ.: Reg1=b'10101010'
btfsc Reg1,7
εντολή 1
εντολή 2

Μετά την εκτέλεση της εντολής btfss Reg1,7 η επόμενη εντολή που θα εκτελεστεί θα είναι η 'εντολή 2' καθώς το bit7 (Reg1<7>) είναι ίσο με '1'.

Εντολές PIC16F887

26a. call <ετικέτα>

26b. return

Η εντολή call πραγματοποιεί διακλάδωση στην ετικέτα που ακολουθεί (κλήση υποροutίνας) ενώ η εντολή return πραγματοποιεί επιστροφή από υποροutίνα, π.χ.:

```
      Εντολή 1
      Εντολή 2
      call SUBROUTINE1
      Εντολή 3
      Εντολή 4
LOOP  goto LOOP
SUBROUTINE1
      Εντολή 5
      Εντολή 6
      return
```

Εντολές PIC16F887

27. sleep

Με την εντολή αυτή ο μικροελεγκτής εισέρχεται σε λειτουργία χαμηλής κατανάλωσης ισχύος (Low Power Mode). Επανέρχεται σε κανονική λειτουργία μετά από reset ή από αίτηση διακοπής.

Στην κατάσταση χαμηλής κατανάλωσης ισχύος απενεργοποιούνται τα περισσότερα κυκλώματα του μικροελεγκτή. Παύει να λειτουργεί ο ταλαντωτής χρονισμού οπότε δεν εκτελούνται εντολές.

Η λειτουργία αυτή είναι πολύ σημαντική καθώς σε εφαρμογές που βασίζονται σε μπαταρίες θα πρέπει να έχουμε τη χαμηλότερη δυνατή κατανάλωση ισχύος για τη μεγιστοποίηση της διάρκειας της μπαταρίας. Τυπικά ο μικροελεγκτής καταναλώνει κάποια mA στην κανονική λειτουργία ενώ σε sleep mode μπορεί να καταναλώνει λίγα nA.

Καταχωρητές έμμεσης διευθυνσιοδότησης

Ο PIC16F887 διαθέτει έναν καταχωρητή δείκτη με τη χρήση του οποίου μπορούμε να πραγματοποιήσουμε έμμεση (ή δεικτοδοτούμενη) διευθυνσιοδότηση.

Ο καταχωρητής αυτός είναι ο FSR.

Για την προσπέλαση στη διεύθυνση μνήμης που "δείχνει" ο FSR χρησιμοποιείται το συμβολικό όνομα καταχωρητή INDF.

π.χ.

```
Reg1      equ    h'20'
Reg2      equ    h'21'
  movlw h'20'  (W <= h'20')
  movwf FSR    (FSR <= h'20')
  movlw d'10'  (W <= d'10')
  movwf INDF   (Reg1 <= d'10')
  incf FSR,f   (FSR <= FSR+1 ή FSR <= h'21')
  movwf INDF   (Reg2 <= d'10')
```

Παραδείγματα – Ασκήσεις

1. Να δηλωθεί καταχωρητής Reg1 σε θέση μνήμης γενικής χρήσης της RAM του μικροελεγκτή PIC16F887. Να αποθηκευτεί στον Reg1 ο αριθμός d'50' και να γίνει ο πολλαπλασιασμός 50x4. Το αποτέλεσμα να αποθηκευτεί στον καταχωρητή Reg1.

```
Reg1   equ   h'20'
   movlw d'50'    ; W <= d'50'
   movwf Reg1     ; Reg1 <= d'50'
   rlf Reg1,f     ; Reg1 <= Reg1 * 2 = d'100'
   bcf Reg1,0     ; Μηδενισμός του bit0
   rlf Reg1,f     ; Reg1 <= Reg1 * 2 = d'200'
   bcf Reg1,0     ; Μηδενισμός του bit0
```

Παραδείγματα – Ασκήσεις

2. Να δηλωθεί καταχωρητής Reg1 σε θέση μνήμης γενικής χρήσης της RAM του μικροελεγκτή PIC16F887. Να αποθηκευτεί στον Reg1 ο αριθμός d'250' και να γίνει η διαίρεση 250/4. Το αποτέλεσμα να αποθηκευτεί στον καταχωρητή Reg1.

```
Reg1    equ   h'20'
    movlw d'250'   ; W <= d'250'
    movwf Reg1     ; Reg1 <= d'250'
    rrf Reg1,f     ; Reg1 <= Reg1 / 2 = d'125'
    bcf Reg1,7     ; Μηδενισμός του bit7
    rrf Reg1,f     ; Reg1 <= Reg1 / 2 = d'62' (C=1)
    bcf Reg1,7     ; Μηδενισμός του bit7
```

Παραδείγματα – Ασκήσεις

3. Να μηδενιστούν τα bit0 και bit3 του καταχωρητή Reg1. Τα υπόλοιπα bit του καταχωρητή να παραμείνουν όπως ήταν πριν.

Πριν: Reg1 = b'UUUUUUUU'

```
bcf Reg1,0
bcf Reg1,3
```

Μετά: Reg1 = b'UUUU0UU0'

4. Να καθοριστούν στο λογικό '1' τα bit7 και bit2 του καταχωρητή Reg1. Τα υπόλοιπα bit του καταχωρητή να παραμείνουν όπως ήταν πριν.

Πριν: Reg1 = b'UUUU0UU0'

```
bsf Reg1,7
bsf Reg1,2
```

Μετά: Reg1 = b'1UUU01U0'

Σημ.: U=Unchanged

Παραδείγματα – Ασκήσεις

5. Να γίνει ο έλεγχος αν Reg1=4 τότε να φορτωθεί στον Reg2 η τιμή d'100' διαφορετικά να φορτωθεί η τιμή d'200'.Ποιος από τους παρακάτω είναι καλύτερος τρόπος;

```
   movlw d'4'
   subwf Reg1,w
   btfsc STATUS,Z
     goto LOAD_100
   goto LOAD_200
LOAD_100
   movlw d'100'
   movwf Reg1
   goto CONTINUE
LOAD_200
   movlw d'200'
   movwf Reg1
   goto CONTINUE
CONTINUE
   ....
```

```
movlw d'4'
subwf Reg1,w
btfsc STATUS,Z
 movlw d'100'
btfss STATUS,Z
 movlw d'200'
movwf Reg1
```

Το λεπτό σημείο εδώ είναι ότι η εντολή movlw δεν επηρεάζει το Z flag

Παραδείγματα – Ασκήσεις

6. Να γίνει πρόγραμμα για τον μικροελεγκτή PIC16F887 το οποίο πραγματοποιεί άθροιση ανάμεσα σε δύο 16-bit αριθμούς.
Να γίνει η υπόθεση ότι το υψηλής τάξης byte του 1ου προσθετέου βρίσκεται στη θέση μνήμης h'20'.
Να γίνει η υπόθεση ότι το χαμηλής τάξης byte του 1ου προσθετέου βρίσκεται στη θέση μνήμης h'21'.
Να γίνει η υπόθεση ότι το υψηλής τάξης byte του 2ου προσθετέου βρίσκεται στη θέση μνήμης h'22'.
Να γίνει η υπόθεση ότι το χαμηλής τάξης byte του 2ου προσθετέου βρίσκεται στη θέση μνήμης h'23'.
Να αποθηκευτεί το αποτέλεσμα στις θέσεις μνήμης h'30' (high byte) και h'31' (low byte). Τυχόν κρατούμενο να αποθηκεύεται στο Carry flag του καταχωρητή STATUS.

ΛΥΣΗ:

```
    include <p16f887.inc>
Reg11    equ  h'20'
Reg10    equ  h'21'
Reg21    equ  h'22'
Reg20    equ  h'23'
Sum1     equ  h'30'
Sum0     equ  h'31'
HelpReg  equ  h'32'
    org h'00'
    movlw h'27'
    movwf Reg11   ; Reg11 <= h'27'
    movlw h'35'
    movwf Reg10   ; Reg10 <= h'35' άρα h'2735'= 10037
    movlw h'44'
    movwf Reg21   ; Reg11 <= h'44'
    movlw h'27'
    movwf Reg20   ; Reg10 <= h'27' άρα h'4427'= 17447
```

ΛΥΣΗ (συνέχεια):

```
;πράξεις άθροισης
   clrf HelpReg ;μηδενισμός βοηθητικής μεταβλητής
   movf Reg20,w     ; W <= Reg20
   addwf Reg10,w    ; W <= Reg10 + W
   movwf Sum0       ; Sum0 <= W
   btfsc STATUS,C   ; Έλεγχος ύπαρξης κρατουμένου
     bsf HelpReg,0  ; Αποθήκευση κρατουμένου
   movf Reg21,w     ; W <= Reg21
   addwf Reg11,w    ; W <= Reg11 + W
   movwf Sum1       ; Sum1 <= W
   btfsc HelpReg,0  ; Έλεγχος κρατουμένου από πριν
     incf Sum1,f    ; Αύξηση κατά 1 αν είχε Carry
LOOP
   goto LOOP
   end
```

Παραδείγματα – Ασκήσεις

7. Στις θέσεις μνήμης από h'30' ώς h'37' αποθηκεύονται οι μετρήσεις από έναν αισθητήρα θερμοκρασίας οι οποίες κυμαίνονται από 0 ως 30 βαθμούς Κελσίου. Να βρεθεί ο μέσος όρος των τιμών θερμοκρασίας και να αποθηκευτεί σε καταχωρητή Reg1 που δηλώνεται στη θέση h'20'.

ΛΥΣΗ:

```
    include <p16f887.inc>
Reg1      equ   h'20'
Counter   equ   h'21'
    org h'00'
    movlw d'8'
    movwf Counter   ; Counter <= d'8'
    movlw h'30'
    movwf FSR       ; FSR <= d'30'
    ...συνεχίζεται...
```

ΛΥΣΗ (...συνέχεια...)

```
    movlw d'0'      ; Μηδενισμός του W
LOOP
    addwf INDF,w    ; W <= [FSR] + W
    incf FSR,f      ; FSR <= FSR + 1
    decfsz Counter,f ; Μείωση κατά 1 του Counter
      goto LOOP     ; Διακλάδωση αν Counter≠0
    movwf Reg1      ; Αποθήκευση αθροίσματος στο Reg1
    rrf Reg1,f      ; Διαίρεση με το 2
    bcf Reg1,7      ; Μηδενισμός του bit7 (ίσως C=1)
    rrf Reg1,f      ; Διαίρεση με το 2 (συνολικά /4)
    bcf Reg1,7      ; Μηδενισμός του bit7 (ίσως C=1)
    rrf Reg1,f      ; Διαίρεση με το 2 (συνολικά /8)
    bcf Reg1,7      ; Μηδενισμός του bit7 (ίσως C=1)
END_LOOP            ; Ο Reg1 έχει το μέσο όρο
    goto END_LOOP ; Το Carry έχει το υπόλοιπο
    end
```

ΠΡΟΑΙΡΕΤΙΚΗ ΕΡΓΑΣΙΑ

Να γίνει πρόγραμμα για τον μικροελεγκτή PIC16F887 το οποίο πραγματοποιεί αφαίρεση ανάμεσα σε δύο 16-bit αριθμούς.
Να γίνει η υπόθεση ότι το υψηλής τάξης byte του αφαιρετέου βρίσκεται στη θέση μνήμης h'20' και ότι το χαμηλής τάξης byte βρίσκεται στη θέση μνήμης h'21'.
Να γίνει η υπόθεση ότι το υψηλής τάξης byte του αφαιρέτη βρίσκεται στη θέση μνήμης h'22' και ότι το χαμηλής τάξης byte βρίσκεται στη θέση μνήμης h'23'.
Να αποθηκευτεί η διαφορά στις θέσεις μνήμης h'30' (high byte) και h'31' (low byte).
Να ερμηνευτεί το αποτέλεσμα της αφαίρεσης (πότε είναι θετικό, πότε αρνητικό, πότε μηδέν) και σε ποια μορφή βρίσκεται.

Η εργασία θα περιλαμβάνει κώδικα ο οποίος θα πρέπει να αναπτυχθεί στο περιβάλλον MPLAB IDE καθώς επίσης και ένα συνοπτικό κείμενο στο οποίο θα παρατίθεται η ερμηνεία του αποτελέσματος, το λογικό διάγραμμα και παραδείγματα αφαίρεσης με τα αποτελέσματά της καλύπτοντας κάθε δυνατή περίπτωση.