

T.C.
MİLLÎ EĞİTİM BAKANLIĞI



MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

ELEKTRİK ELEKTRONİK TEKNOLOJİSİ

MİKRODENETLEYİCİ PROGRAMLAMA

ANKARA 2007

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).
- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.
- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.
- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşabilirler.
- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.
- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR	iii
GİRİŞ	1
ÖĞRENME FAALİYETİ-1	3
1. MİKRODENETLEYİCİ PROGRAMI YAZMA.....	3
1.1. Akış Diyagram Oluşturma ve Sembolleri.....	3
1.2. Mikrodenetleyici Assembler Programı ve Yazım Kuralları	4
1.2.1. Noktalı Virgül	5
1.2.2. Başlık.....	5
1.2.3. Sabitler	9
1.2.4. Org Deyimi	9
1.2.5. Girintiler ve Program Bölümleri	9
1.3. Mikrodenetleyici Komutları	11
1.3.1. Byte Yönlendirmeli Komutlar	12
1.3.2. Bit Yönlendirmeli Komutlar	22
1.3.3. Sabit İşleyen Komutlar	24
1.3.4. Kontrol Komutları.....	27
1.4. Sayı ve Karakterlerin Yazılışı	30
1.4.1. Heksadesimal Sayılar	30
1.4.2. Binary Sayılar	30
1.4.3. Desimal Sayılar.....	31
1.4.4. ASCII Karakterler	31
1.5. Mikrodenetleyici İçin Gerekli Yazılımın Kullanımı	31
1.5.1. Programın Kurulması	32
1.5.2. Menülerin Tanıtılması	34
1.5.3. Mikrodenetleyici ve Diğer Donanımların Seçilmesi	34
1.6. Programlama Tekniği	35
1.6.1. Bank Değiştirme	36
1.6.2. Portların Giriş ve Çıkış Olarak Yönlendirilmesi.....	38
1.6.3. Her Adım İçin Akış Diyagramı Çizme.....	38
1.6.4. Konfigürasyon Bitlerinin Yazılması	39
1.6.5. W Kayıtçısının Kullanımı.....	40
1.6.6. Bitleri Test Ederek İşlem Yapma.....	41
1.6.7. Sayaç Kullanarak Döngü Düzenlemek.....	42
1.6.8. Karşılaştırma Yapararak Döngü Düzenlemek	43
1.6.9. Status Kayıtçısı	44
1.6.10. Zaman Geciktirme Döngüleri	46
1.6.11. Altprogramlar.....	51
1.6.12. Bit Kaydırma.....	52
1.6.13. Mantıksal İşlemler.....	55
1.6.14. Aritmetik İşlemler	58
1.6.15. Çevrim Tabloları.....	61
1.6.16. Kesmeler	65
1.6.17. Donanım Sayıcıları.....	72
1.6.18. D/A ve A/D Çevirme.....	82
UYGULAMA FAALİYETİ.....	91
ÖLÇME VE DEĞERLENDİRME	92

ÖĞRENME FAALİYETİ-2	94
2. MİKRODENETLEYİCİ KONTROL PROGRAMININ MAKİNE DİLİNE ÇEVİRİLMESİ	94
2.1. Programın Derlenmesi.....	95
2.1.1. Derleme İşleminin Yapılması	95
2.1.2. Derleme Sonucu Elde Edilen Dosyalar	97
UYGULAMA FAALİYETİ.....	104
ÖLÇME VE DEĞERLENDİRME	105
MODÜL DEĞERLENDİRME.....	107
ÖNERİLEN KAYNAKLAR	110
KAYNAKÇA	111

AÇIKLAMALAR

KOD	523E00020
ALAN	Elektrik Elektronik Teknolojisi
DAL/MESLEK	Otomasyon Sistemleri
MODÜLÜN ADI	Mikrodenetleyici Programlama
MODÜLÜN TANIMI	Mikrodenetleyici programının yazılması ve derlenmesi ile ilgili bilgi ve becerilerin kazandırıldığı öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Mikroişlemci ve mikrodenetleyiciler modülünü tamamlamış olmak.
YETERLİK	Mikrodenetleyiciyi programlamak.
MODÜLÜN AMACI	Genel Amaç Bu modül ile “eğitim öğretim ortamları ve donanımları”nda belirtilen ortam sağlandığında , yapılacak işleme göre gerekli teknikleri kullanarak mikrodenetleyici programını hatasız yapabileceksiniz. Amaçlar 1. Kurulacak devre için mikrodenetleyici programlama yazılımını eksiksiz olarak kullanabileceksiniz. 2. Yapılan mikrodenetleyici programını makine (heksadesimal kodlara) diline hatasız olarak çevirebileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Atölye ortamı, mikrodenetleyici programlama kartı, mikrodenetleyici programlama kartı yazılımı, mikrodenetleyici uygulama kartı, bilgisayar , AVOMETRE, aktif ve pasif devre elemanları, lehimleme malzemeleri, diğer faydalı el ve güç araçları donanımları .
ÖLÇME VE DEĞERLENDİRME	Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendi kendinizi değerlendireceksiniz. Öğretmen modül sonunda size ölçme aracı (uygulama, soru-cevap) uygulayarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek değerlendirecektir.



GİRİŞ

Sevgili Öğrenci,

Mikrodenetleyiciler günlük hayatta kullandığımız pek çok cihaz içerisine girmiş durumdadır. Öyle ki, mikrodenetleyicileri otomobillerde, cep telefonlarında, kameralarda, faks-modem cihazlarında, fotokopi, radyo, TV ve bazı oyuncaklar gibi sayılamayacak kadar pek çok alanda kullanımını görmek mümkündür.

Mikrodenetleyici kullanımının bu kadar yaygın olmasının çok önemli nedenleri vardır. Daha önce mikroişlemci kullanımı gereken yerlerde, ayrıca hafıza üniteleri (RAM, ROM) ve giriş/çıkış arabirim devrelerine (I/O) ihtiyaç duyuluyordu. Bu da sistemin maliyetini yükselmesi, devrenin karmaşık olması ve fazla yer kaplaması, programlamanın zorluğu gibi problemler oluşturuyordu. Fakat mikrodenetleyicilerle bu sorunlar ortadan kaldırıldı. Çünkü fiyatları son derece makul olan bu entegreler, ayrıca hafıza üniteleri (RAM, ROM) ve giriş/çıkış arabirim devrelerine (I/O) ihtiyaç duyulmadan istenilen tasarımın yapılabilmesine imkân tanımaktadır. Bununla beraber programlanması da oldukça kolay ve programlama dili olan assembly için de bir bedel gerekmektedir. Ayrıca flash bellekleri sayesinde bir denetleyicinin onlarca defa programlanabilip silinebilmesine imkân tanır. İşte bu gibi özelliklerinden dolayı mikrodenetleyiciler pek çok alanda mikroişlemcilerin yerini almıştır.

Günümüzde Texas ins., NSC, Atmel ve Microchip gibi birçok firma mikrodenetleyici üretmekte ve bunları donanım tasarımcılarının kullanımına sunmaktadır. Biz, bundan önceki modüldeki olduğu gibi, bu modülde de Microchip firmasının ürettiği mikrodenetleyicileri anlatacağız. Konular genelde bu firmanın ürettiği PIC16F84 mikrodenetleyicisi üzerinden işlenecektir. Bu şekilde, modül sonunda artık siz de bir mikrodenetleyiciyi tanımış ve amaca göre programını yazabilme seviyesine çıkmış olacaksınız.



Bir PIC mikrodenetleyicisi programlayıcı ve deneme kartı



ÖĞRENME FAALİYETİ-1

AMAÇ

Uygun ortam sağlandığında kurulacak devre için mikrodenetleyici programını eksiksiz olarak yapabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde yapmanız gereken öncelikli araştırmalar şunlardır:

- Ø Mikrodenetleyici çeşitlerini (Microchip firmasının ürettiği PIC serisi) araştırınız.
- Ø Mikrodenetleyici komutlarını araştırınız.
- Ø Mikrodenetleyicilerin donanımsal farklılıklarını araştırınız.

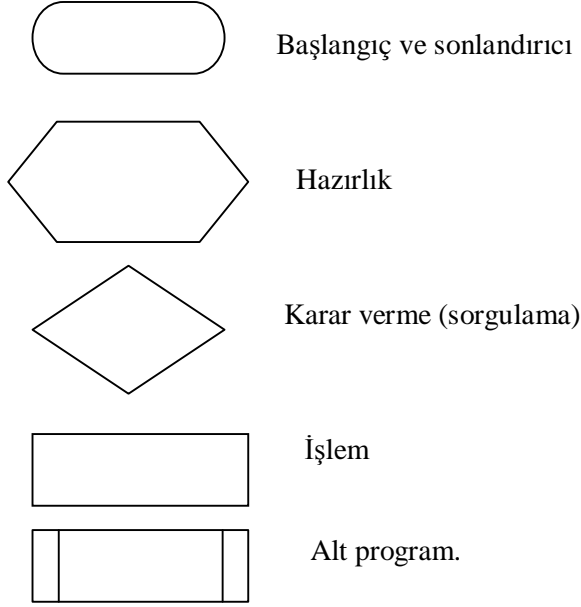
Araştırma işlemleri için internet ortamı ve mesleki kitapların bulunduğu kütüphaneler ile bu sektör üzerinde çalışan teknik elemanlardan faydalanabilirsiniz.

1. MİKRODENETLEYİCİ PROGRAMI YAZMA

Assembly dili bir mikrodenetleyiciden (bundan sonra sadece denetleyici veya PIC olarak bahsedeceğiz) beklenen veya denetleyicinin yapması istenen işlemlerin belirli kurallara uygun olarak yazılmış komutlar dizisidir. Assembly komutları İngilizcedeki manalarının kısaltmalarından meydana gelmektedir. Bu komutlar genellikle bir komutun yaptığı işi ifade eden İngilizce sözcüklerin baş harflerinden meydana gelmektedir. Örneğin BTFSC Bit Test F Skip if Clear (File kayıtçındaki (kaydedici) bit'i test et). Şimdi bir denetleyicinin assembly programı yapmadan önce , çıkarılacak akış diyagramında kullanılan sembolleri görelim.

1.1. Akış Diyagram Oluşturma ve Sembolleri

Assembler ile bir program yazmadan evvel, denetleyicinin hangi adımlarla programı işleyeceğini planlamamız gerekmektedir. Bu planlama işlemi akış diyagramları ile yapılır. Akış diyagramı işlenecek komutların sırasıdır. Uzun ve karmaşık programlarda, akış diyagramları, hangi seviyeden programın ayrılacağını ve hangi komutları yerine getireceğini, sonra tekrar nereye döneceğini göstermesi sebebiyle faydalıdır. Akış diyagramlarının kendine has sembolleri de vardır. Şekil 1.1'de akış diyagramı sembolleri görülmektedir;



Şekil 1.1: Akış diyagramı sembolleri

Bu modül içerisindeki akış diyagramlarını inceleyiniz.

1.2. Mikrodenetleyici Assembler Programı ve Yazım Kuralları

PIC assembly programlarının yazılabilmesi için kullanılan metin editörlerinden Windows altında çalışan NOTPAD veya dosya altında çalışan EDIT en uygun olanlarından bir kaç tanesidir. Bunun dışında printer kontrol komutları içermeyen ve ASCII kodunda dosya üretebilen herhangi bir editör de kullanılabilir. MPLAB kullanıldığında ayrıca editör kullanmaya gerek yoktur. Çünkü MPLAB'ın içinde hem bir text editörü hem de MPASM bulunmaktadır.

Bunun dışında PIC mikrodenetleyicisi için program yazmanın başka yolları da vardır. Pic Basic Pro, C diliyle yazılmış programı PIC için derleyici programlar veya PLC mantığıyla görsel olarak PIC programlayıcılar (PICBIT) ile de PIC denetleyicileri için program yazılabilir. Fakat bu programlar da arka planda kendi ASM uzantılı kütüklerini kullanırlar. Biz bu yüzden bütün bu programlara da temel teşkil eden assembly dilini kullanacağız.

MPASM assembler programının yazılan komutları doğru olarak algılayıp PIC'in anlayabileceği heksadesimal kodlara dönüştürülmesi, şu bilgilerin program içerisinde özel formatta yazılması gerekir;

- Ø Komutların hangi PIC için yazıldığı
- Ø Programların bellekteki hangi adresten başlayacağı

Ø Komutların ve etiketlerin neler olduğu

Ø Programın bitiş yeri

Şimdi bir assembly programında bunların nasıl yazıldığına bakalım.

1.2.1. Noktalı Virgül

Baş tarafına (;) konulan satır, assembler tarafından heksadesimal kodlara dönüştürülemez. Bu satırlar programın geliştirilmesi esnasında hatırlatıcı açıklamaların yazılmasında kullanılır. Aşağıda örnekte CLRF ile başlayan satırda “portB’nin içeriğini sıfırla” cümlesi, CLRF komutunun ne iş yaptığını açıklar. Programın bölümlerini birbirinden ayırmak için (----- veya ===) çizgileri kullanmak, programı görsel olarak daha okunur hâle getirdiği gibi bu çizgiler arasına uyarılar ve açıklamalar da yazılabilir.

```
=====bu satır derleyici tarafından dikkate alınmaz.=====
CLRF PORTB          ; portB’nin içeriğini sıfırla.
=====bu satır da derleyici tarafından dikkate alınmaz.=====
```

1.2.2. Başlık

Programın başındaki bilgilere başlık bölümü denilir.

```
===== PROGRAM1. ASM =====
LIST P=16F84
;-----
INCLUDE          “P16F84.INC”
```

Başlık bölümünde program dosyasının adı ve hazırlandığı tarih, istenirse hazırlayanın adı ya da farklı bir ad yazılabilir. İlk satır, bir açıklama satırıdır ve assembler tarafından derlenmez.

LIST P=16F84 satırı, programın hangi PIC için yazıldığını belirtir. LIST bir derleyici bildirisidir. Yani derleyiciyi yönlendiren bir komuttur ve yegane kullanım amacı ve yeri burasıdır.

Başlık bölümünde INCLUDE komutu da kullanılabilir. Bir program yazılırken tüm kayıtçıları tek tek tanımlamak oldukça zordur. Bu yüzden include dosyası denilen bu tanımların içinde bulunduğu ve assembler tarafından tanınan bir dosya kullanırız. Assembler tarafından tanınması için program girişinde tanımlanır.

Örneğin PORTB, STATUS ve TRISB kayıtçılarını kullanan bir program yazdığınızı varsayalım. Eğer programınızın başlık kısmında

```
INCLUDE          “P16F84.INC”
```

Şeklinde bir ifade kullanırsanız programınızda kayıtçıları kendiniz tanıtmak zorunda kalmazsınız. Aksi hâlde aşağıdaki gibi bellek adreslerini kendiniz tanıtmalısınız.

```
PORTB EQU          h'06'
                STATUS h'03'
                TRISB  h'86'
```

Aşağıda PIC16F84 mikrodenetleyicisi için, Microchip Technology, Inc. tarafından hazırlanmış INC dosyasını görüyorsunuz. Burada kayıtçılarının bellek adreslerinin ve kayıtçı bitlerinin tek tek tanımlandığına dikkat ediniz.

```
LIST
```

```
; P16F84.INC Standard Header File, Version 2.00 Microchip Technology, Inc.
```

```
NOLIST
```

```
; This header file defines configurations, registers, and other useful bits of
; information for the PIC16F84 microcontroller. These names are taken to match
; the data sheets as closely as possible.
; Note that the processor must be selected before this file is
; included. The processor may be selected the following ways:
```

- ```
; 1. Command line switch:
; C:\MPASM MYFILE.ASM /PIC16F84
; 2. LIST directive in the source file
; LIST P=PIC16F84
; 3. Processor Type entry in the MPASM full-screen interface
```

```
=====
=====
;
; Revision History
;
;=====
=====
```

```
;Rev: Date: Reason:
```

```
;2.00 07/24/96 Renamed to reflect the name change to PIC16F84.
;1.01 05/17/96 Corrected BADRAM map
;1.00 10/31/95 Initial Release
```

```
=====
=====
```

```

;
; Verify Processor
;
;=====
=====

 IFNDEF __16F84
 MESSG "Processor-header file mismatch. Verify selected processor."
 ENDIF

;=====
=====
;
; Register Definitions
;
;=====
=====

W EQU H'0000'
F EQU H'0001'
;----- Register Files-----

INDF EQU H'0000'
TMR0 EQU H'0001'
PCL EQU H'0002'
STATUS EQU H'0003'
FSR EQU H'0004'
PORTA EQU H'0005'
PORTB EQU H'0006'
EEDATA EQU H'0008'
EEADR EQU H'0009'
PCLATH EQU H'000A'
INTCON EQU H'000B'

OPTION_REG EQU H'0081'
TRISA EQU H'0085'
TRISB EQU H'0086'
EECON1 EQU H'0088'
EECON2 EQU H'0089'

;----- STATUS Bits -----

IRP EQU H'0007'
RP1 EQU H'0006'
RP0 EQU H'0005'
NOT_TO EQU H'0004'
NOT_PD EQU H'0003'
Z EQU H'0002'
DC EQU H'0001'

```

```
C EQU H'0000'
;---- INTCON Bits -----
```

```
GIE EQU H'0007'
EEIE EQU H'0006'
TOIE EQU H'0005'
INTE EQU H'0004'
RBIE EQU H'0003'
TOIF EQU H'0002'
INTF EQU H'0001'
RBIF EQU H'0000'
```

```
;---- OPTION Bits -----
```

```
NOT_RBPU EQU H'0007'
INTEDG EQU H'0006'
TOCS EQU H'0005'
TOSE EQU H'0004'
PSA EQU H'0003'
PS2 EQU H'0002'
PS1 EQU H'0001'
PS0 EQU H'0000'
```

```
;---- EECON1 Bits -----
```

```
EEIF EQU H'0004'
WRERR EQU H'0003'
WREN EQU H'0002'
WR EQU H'0001'
RD EQU H'0000'
```

```
=====
=====
```

```
;
; RAM Definition
;
```

```
=====
=====
```

```
__MAXRAM H'CF'
__BADRAM H'07', H'50'-H'7F', H'87'
```

```
=====
=====
```

```
;
; Configuration Bits
;
```

```

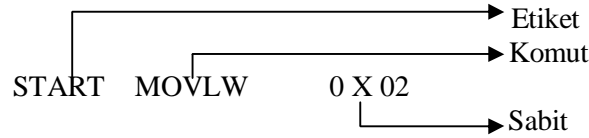
=====
=====
_CP_ON EQU H'000F'
_CP_OFF EQU H'3FFF'
_PWRTE_ON EQU H'3FF7'
_PWRTE_OFF EQU H'3FFF'
_WDT_ON EQU H'3FFF'
_WDT_OFF EQU H'3FFB'
_LP_OSC EQU H'3FFC'
_XT_OSC EQU H'3FFD'
_HS_OSC EQU H'3FFE'
_RC_OSC EQU H'3FFF'

```

LIST

### 1.2.3. Sabitler

PIC assembly dilinde heksadesimal sayılar birer sabittir. Sabitler MOVLW ve bazı mantıksal ve aritmetik işlem komutlarında kullanılırlar.



### 1.2.4. Org Deyimi

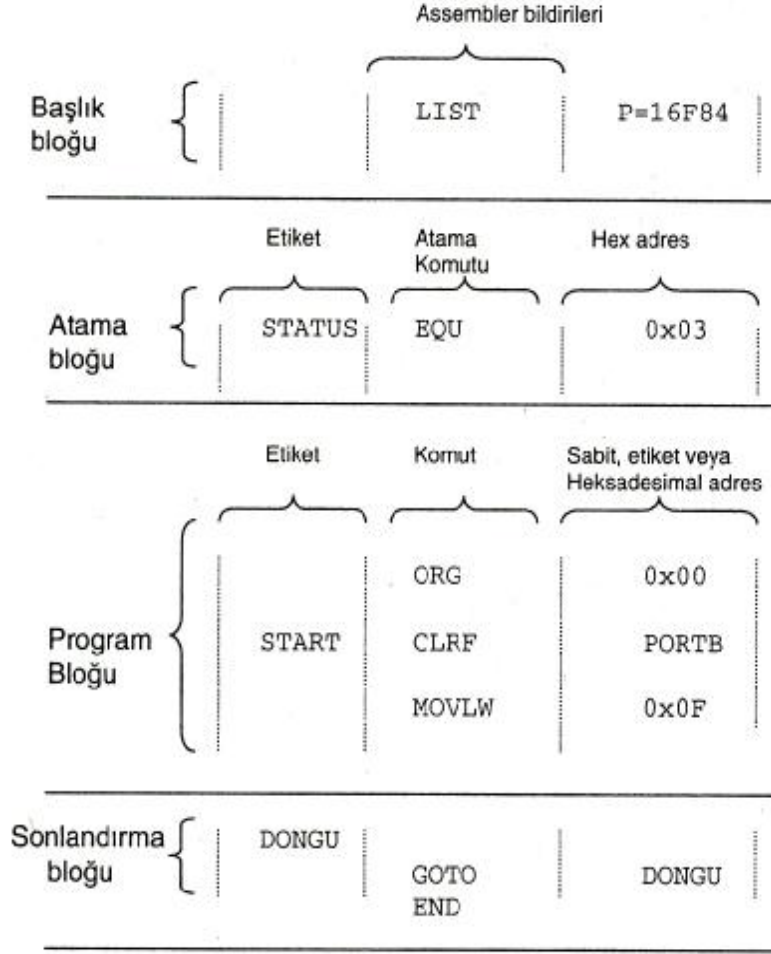
ORG İngilizce'deki "origin" kelimesinden gelmektedir. ORG deyimi iki amaç için kullanılır; program komutlarının hangi adresten itibaren başladığını ve PIC'in kesme alt programının başlangıç adresini.

ORG 0 x 000 ; ilk program komutunun bellek adresi

ORG 0 x 004 ; h'004 adresi, PIC16F84 denetleyicisinin kesme alt programının başlangıç adresi

### 1.2.5. Girintiler ve Program Bölümleri

Metin editörlerinde birbirinden farklı uzunlukta girintiler veren TAB özelliği vardır. Bu özellikten yararlanarak assembly komutları üç kolona bölünerek yazılır. Bir assembly programı temel olarak dört bölüme ayrılır. Bunlar: Başlık, atama, program ve sonuç bölümleridir.



**Şekil 1.2: Girintiler ve program bölümlerinin görünüşü**

Assembler, yukarıda Şekil 1.2’de belirtildiği gibi komutların üç kolona bölünerek yazılmış olduğunu varsayar. Belirtilen kolona yazılmayan bir komut olduğunda ise bunu da kabul eder. Ancak, heksadesimal kodlara dönüştürme esnasında bu tip hataları bir uyarı olarak belirtir. Assembly komutları yazılırken kolonlar arasında verilen TAB’ların uzunluğu önemli değildir. Boşluk tuşu ile verilen aralık da assembler tarafından TAB olarak algılanır. PIC’ e yaptırılacak herhangi bir işlem için, özellikle uzun ve zor programlar yazmaya başladıktan sonra yapılacak açıklamalar, bölümlere ayrılmış programların daha kullanışlı olduğu görülür. Çünkü programlar bu şekilde yazıldığında daha sonraki geliştirmelere açıktır. Aradan zaman geçse bile bir programı geliştirmek için tekrar ele aldığımızda, program içerisine yazılan açıklamalar ihtiyacınız olan hatırlatmaları yapacaktır. Şimdi de bu bölümleri görelim.

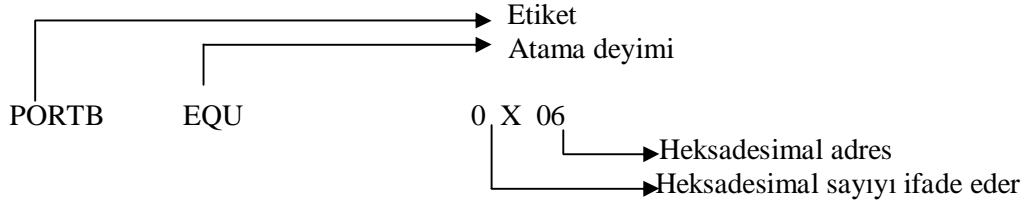
### 1.2.5.1. Başlık Bloğu

Programının başındaki bilgilere başlık bölümü denilir. Bu bölümü daha önce “başlık” konusunda gördüğümüz için geçiyoruz.



### 1.2.5.2. Atama Bloğu

EQU deyimi PIC'in belleğindeki bir heksadesimal adresi belirlenen bir etikete atamak için kullanılır. Aşağıda atama deyimine bir örnek gösterilmiştir.



### 1.2.5.3. Program Bloğu

Programcının assembly komutları kullanarak işi yaptıracak programı yazdığı, içinde altprogramlarında bulunabileceği ana bölümdür. Modül içindeki programları inceleyerek program bloğunu bulmaya çalışınız.

### 1.2.5.4. Sonlandırma Bloğu

PIC 16F84'ün duraklama (halt) komutu yoktur. Programı belirli bir yerde duraklatmak için bazen sonsuz döngü kullanılır.

```
DONGU
 GOTO DONGU
END
```

Yukarıdaki sonsuz döngüde DONGU etiketine assembler otomatik olarak bir adres verir. GOTO DONGU komutu ise program akışını devamlı olarak aynı adrese gönderir. Bu durumda program belirlenen adreste duraklatılmış olur.

END deyimi ise program komutlarının sona erdiğini assemblerlere bildirir. Her program sonunda END deyimi muhakkak kullanılmalıdır. Aksi hâlde program devam ederken dosya sonunun belirtilmediğini gösteren bir hata mesajı verecektir.

## 1.3. Mikrodenetleyici Komutları

PIC16F84 denetleyicisinin toplam 35 tane komutu vardır. Diğer PIC serisinin başka komutlarında olmasına rağmen bu 35 komut temel teşkil ettiği için biz de bu komutları göreceğiz. Bu komutların yazılış biçimini dört grupta toplayabiliriz .

- Ø Byte-yönlendirmeli komutlar.
- Ø Bit-yönlendirmeli komutlar.
- Ø Sabit işleyen komutlar.
- Ø Kontrol komutları.

Komutların yazılış biçimlerini açıklarken bazı tanımlama harfleri kullanacağız. Önce bu harflerin anlamlarını verelim:

f = Dosya kayıtçısı (kaydedicisi) (File Register)  
d = Gönderilen yer (destination) iki durum söz konusudur ;  
d=0 .....Hedef W kayıtçısıdır.  
d=1 .....Hedef dosya (F) kayıtçısıdır .  
k = Sabit veya adres etiketi  
b = Bit tanımlayıcı  
b = Binary sayıları belirleyen harf (Örneğin b'00001111' gibi)  
d = Desimal sayıları belirleyen harf (Örneğin d '16' gibi)  
h = Hexadesimal sayıları belirleyen harf (Örneğin h '0A' gibi)

### 1.3.1. Byte Yönlendirmeli Komutlar

Bu grup komutlarda hedef kayıtçı W veya bir dosya kayıtçısı (f) olabilir. Kayıtçılar arası veri transferleri ve işlemleri bu komutlarla gerçekleşir. Şimdi bu komutları görelim:

|                      |                                                                                       |      |      |      |      |
|----------------------|---------------------------------------------------------------------------------------|------|------|------|------|
| <b>ADDWF</b>         | W ve f kayıtçılarını topla.                                                           |      |      |      |      |
| Dizim:               | [etiket] ADDWF f,d                                                                    |      |      |      |      |
| Operandlar:          | $0 \leq f \leq 127$ $d \in [0,1]$                                                     |      |      |      |      |
| İşlem:               | $(W) + (f) \rightarrow (\text{hedef})$                                                |      |      |      |      |
| Etkilenen bayraklar: | C,DC,Z                                                                                |      |      |      |      |
| Kodlama:             | <table border="1"><tr><td>00</td><td>0111</td><td>dfff</td><td>ffff</td></tr></table> | 00   | 0111 | dfff | ffff |
| 00                   | 0111                                                                                  | dfff | ffff |      |      |

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.  
Tanım: W kayıtçısının içeriğini f kayıtçısına ekler. Eğer d=0 ise, sonuç W kayıtçısının içerisinde depolanır. d=1 ise sonuç f kayıtçısının içerisinde geri saklanır.  
Hafıza alanı (Bayt): 1  
Saat palsı: 1

**Örnek:** ADDWF FSR, 0

Komuttan önce;  
W = h'17'  
FSR = h'C2' olsun.  
Komuttan sonra;  
d=0 olduğundan sonucun W de olduğuna dikkat ediniz.  
W = h'D9'  
FSR = h'C2' olur.

|              |                                          |
|--------------|------------------------------------------|
| <b>ANDWF</b> | W'yi f ile mantıksal AND'le.             |
| Dizim:       | [etiket] ANDWF f,d                       |
| Operandlar:  | $0 \leq f \leq 127$ $d \in [0,1]$        |
| İşlem:       | $(W).AND.(f) \rightarrow (\text{hedef})$ |

Etkilenen bayraklar: Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0101 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: W kayıtçısını f kayıtçısı ile mantıksal AND'ler. AND mantığında girişlerden birinin 0 olması sonucu 0 yapacaktır. Sonucun 1 olması için tüm girişlerin 1 olması gerekir. Eğer d=0 ise, sonuç W kayıtçısı içinde saklanır. Eğer d=1 ise, sonuç f kayıtçısı içinde geri saklanır.

Hafıza alanı (Bayt): 1

Saat palsı: 1

Örnek: ANDWF FSR, 1

Komuttan önce;

W = h'17'

FSR = h'C2' olsun.

Komuttan sonra;

h'17' = b'0001 0111'

AND h'C2' = b'1100 0010'

b'0000 0010' = h'02' ve ;

d=1 olduğundan sonucun f (burada FSR) de olduğuna dikkat ediniz.

W = h'17'

FSR = h'02'

**CLRF** F kayıtçısını sil. ( Clear f )

Dizim: [etiket] CLRF f

Operandlar:  $0 \leq f \leq 127$

İşlem: 00h → (f) 1 → Z

Etkilenen bayraklar: Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0001 | 1fff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: f kayıtçısının içeriği silinir ve Z biti kurulur.

Hafıza alanı (Bayt): 1

Saat palsı: 1

Örnek: CLRF SAY

Komuttan önce;

SAY = h'5A'

Komuttan sonra;

SAY = h'00'

Z = 1

**CLRW** W Kayıtçısını Sil.

Dizim: [etiket] CLRW

Operandlar: Yok

İşlem: 00h → (W) 1 → Z

Etkilenen bayraklar: Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0001 | 0000 | 0011 |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım:W kayıtçısı silinir. Z biti set edilir.

Hafıza alanı (Bayt):1

Saat palsı: 1

Örnek: CLRW

Komuttan önce;

W = h'5A'

Komuttan sonra;

W = h'00'

Z = 1

**COMF** F kayıtçısını tümle ( tersini al ).

Dizim: [etiket] COMF f,d

Operandlar:  $0 \leq f \leq 127$

$d \in [0,1]$

İşlem:

(f)  $\rightarrow$  (hedef)

Etkilenen bayraklar: Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 1001 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım:F kayıtçısının içeriği tümlenmiştir. Eğer d=0 ise, sonuç W kayıtçısının içerisinde depolanır, d=1 ise, sonuç f kayıtçısının içerisinde geri saklanır.

Hafıza alanı (Bayt):1

Saat palsı: 1

**Örnek:** COMF test,0

Komuttan önce;

test = h'13' olsun.

Komuttan sonra;

d=0 olduğundan sonucun W'de olduğuna dikkat ediniz.

test = h'13'=b'0001 0011'

terslenirse; b'1110 1100'=h'EC' ve test = h'13' W= h'EC' olur.

**DECF** f kayıtçısının değerini 1 azalt.

Dizim: [etiket] DECF f,d

Operandlar:  $0 \leq f \leq 127$   $d \in [0,1]$

İşlem: (f) - 1  $\rightarrow$  (hedef)

Etkilenen bayraklar: Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0011 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: f kayıtçısının değerini 1 azaltır. Eğer d=0 ise, sonuç W kayıtçısının içerisinde depolanır, d=1 ise, sonuç f kayıtçısının içerisinde geri saklanır.

Hafıza alanı (Bayt):1

Saat pılsı: 1

Örnek: DECF SAY, 1

Komuttan önce;

SAY = h'01'

Z = 0 olsun.

Komuttan sonra;

d=1 olduğundan sonucun f (burada SAY) de olduğuna dikkat ediniz.

SAY = h'00'

Z = 1

**DECFSZ** f kayıtçısının değerini 1 azalt , sonuç 0 ise bir sonraki komuta atla.

Dizim: [etiket] DECFSZ f,d

Operandlar:  $0 \leq f \leq 127$   $d \in [0,1]$

İşlem:  $(f)-1 \rightarrow (\text{hedef});$  sonuç=0 ise atla

Etkilenen bayraklar: Yok

Kodlama: 00 1011 dfff ffff

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: f kayıtçısının içeriği 1 azaltılır.

Aynı zamanda sorgulama

yapar. Kayıtçı içeriği 0 olursa program bir sonraki komuta atlar, değilse alttaki komut işlenir.

Eğer d=0 ise, sonuç W kayıtçısının içerisinde depolanır, d=1 ise, sonuç f kayıtçısının içerisinde geri saklanır.

Hafıza alanı (Bayt):1

Saat pılsı: 1(2)

Sonuç 0'sa 2 saat pılsı , değilse 1 saat pılsı alır.Yani atlama olmadığı durumda uygulanması 1 , atlama durumunda 2 saat pılsı alır.

Örnek: DON DECFSZ CNT,1

GOTO LOOP

DEVAM

• } programın devamı  
•  
•

Komuttan önce;

PC = Adres DON

Komuttan sonra;

CNT= CNT-1

Eğer CNT= 0

PC= {Adres DEVAM}

Eğer CNT≠0,

PC={ Adres DON+1 } yani GOTO komutunun olduğu satır işlenir.

**INCF** f kayıtçısının değerini 1 artır.

Dizim: [etiket] INCF f,d

Operandlar:  $0 \leq f \leq 127$   $d \in [0,1]$

İşlem:  $(f) + 1 \rightarrow (\text{hedef})$

Etkilenen bayraklar: Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 1010 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: f kayıtçısının değerini 1 artırır. Eğer d=0 ise, sonuç W kayıtçısının içerisinde depolanır. d=1 ise, sonuç f kayıtçısının içerisinde geri saklanır.

Hafıza alanı (Bayt):1

Saat palsı: 1

Örnek: INCF CNT,1

Komuttan önce;

CNT= h'FF'

Z =0

Komuttan sonra;

d=1 olduğundan sonucun f (burada CNT) de olduğuna dikkat ediniz. Ayrıca FF+1=00 olacağına da dikkat ediniz.

CNT= h'00'

Z= 1

**INCFSZ** f kayıtçısının değerini 1 artır, sonuç 0 ise bir sonraki komuta atla.

Dizim: [etiket] INCFSZ f,d

Operandlar:  $0 \leq f \leq 127$   $d \in [0,1]$

İşlem:  $(f)+1 \rightarrow (\text{hedef}), \text{sonuç}=0$  ise atla

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 1111 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: f kayıtçısının içeriği 1 arttırılır. Aynı zamanda sorgulama yapar. Kayıtçı içeriği 0 olursa program bir sonraki komuta atlar, değilse alttaki komut işlenir. Eğer d=0 ise, sonuç W kayıtçısının içerisinde depolanır, d=1 ise, sonuç f kayıtçısının içerisinde geri saklanır.

Hafıza alanı (Bayt):1

Saat palsı: 1(2)

Sonuç 0'sa 2 saat palsı, değilse 1 saat palsı alır. Yani atlama olmadığı durumda uygulanması 1 , atlama durumunda 2 saat palsı alır.

Örnek: DON DECFSZ CNT,1

GOTO LOOP

DEVAM

• } programın devamı  
• }  
• }

Komuttan önce;

PC = Adres DON

Komuttan sonra;

CNT = CNT+1

Eğer CNT= 0 ise,

PC= {Adres DEVAM}

Eğer CNT≠0 ise,

PC={Adres DON+1} Yani GOTO komutunun bulunduğu satır işleme sokulur.

**IORWF** f ile W kayıtçılarını mantıksal OR işlemine tabi tut.

---

Dizim: [etiket] IORWF f,d

Operandlar:  $0 \leq f \leq 127$   $d \in [0,1]$

İşlem: (W).OR.(f) → (W)

Etkilenen bayraklar:Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0100 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: f ile W kayıtçılarını mantıksal OR işlemine tabi tutar. OR mantığında girişlerden birinin 1 olması sonucu 1 yapacaktır. Sonucun 0 olması için tüm girişlerin 0 olması gerekir. Eğer d=0 ise, sonuç W kayıtçısının içerisinde depolanır, d=1 ise, sonuç f kayıtçısının içerisinde geri saklanır.

Hafıza alanı (Bayt):1

Saat palsı: 1

Örnek: IORWF RESULT, 0

Komuttan önce;

RESULT = h'13'

W = h'91'

Komuttan sonra;

d=0 olduğundan sonucun W'de olduğuna dikkat ediniz.

h'13'=b'0001 0011'

OR h'93'=b'1001 0011'

b '1001 0011'=h'93' ve ; RESULT = h'13' W = h'93' olur.

**MOVF** F kayıtçı içeriğini hedefe taşı.

---

Dizim: [etiket] MOVF f,d

Operandlar:  $0 \leq f \leq 127$   $d \in [0,1]$

İşlem: (f) → (hedef)

Etkilenen bayraklar: Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 1000 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: F kayıtçı içeriğini belirtilen hedefe taşınır. Eğer d=0 ise, f içeriği W kayıtçısına taşınır. d=1 ise, f içeriği f kayıtçısına taşınır. d=1 durumu, status kayıtçısında Z bayrağını etkileyeceğinden dolayı, bir file kayıtçısının içeriğinin 0 olup olmadığının testinde kullanılabilir.

Hafıza alanı (Bayt):1

Saat palsı: 1  
Örnek: MOVF FSR, 0  
Komuttan sonra;  
d=0 olduğundan sonucun W’de olduğuna dikkat ediniz.

W=FSR olur.

**MOVWF** W’nin içeriğini f kayıtçısına taşı.

---

Dizim: [etiket] MOVWF f

Operandlar:  $0 \leq f \leq 127$

İşlem: (W) → (f)

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0000 | 1fff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: W kayıtçısının içeriği f kayıtçısına taşınır.

Hafıza alanı (Bayt): 1

Saat palsı: 1

Örnek: MOVWF OPTION

Komuttan önce;

OPTION = h’FF’

W = h’4F’

Komuttan sonra;

Bu komutta hedefin (d) belirtilmediğine dikkat ediniz.

OPTION = h’4F’

W = h’4F’

**NOP** İşlem yok

---

Dizim: [etiket] NOP

Operandlar: Yok

İşlem: Yok

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0000 | 0xx0 | 0000 |
|----|------|------|------|

Tanım: 1 saat palsı zaman gecikmesi sağlar.

Hafıza alanı (Bayt): 1

Saat palsı: 1

Örnek: NOP

1 saat palsı zaman gecikmesi sağlar.

**RLF** f kayıtçısını elde biti (Carry) aracılığıyla sola döndür

---

Dizim: [etiket] RLF f,d

Operandlar:  $0 \leq f \leq 127$   $d \in [0,1]$

İşlem: Aşağıdaki tanıma bakınız.

Etkilenen bayraklar: C

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 1101 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.



Tanım: f kayıtçısının içeriği elde biti (C) içinden bir bit sola döndürülür. C biti f kayıtçısının MSB bitini taşır. Eğer d=0 ise, sonuç W kayıtçısının içerisine yerleştirilir. Eğer d=1 ise, sonuç f kayıtçısının içerisinde geri depolanır.



Hafıza alanı (Bayt):1

Saat pılsı: 1

Örnek: RLF REG1,0

Komuttan önce;

REG1 = b'1110 0110'

C = 0

Komuttan sonra;

REG1 = b'1110 0110'

W = b'1100 1100'

C = 1 olur.

**RRF** f kayıtçısını elde biti (Carry) aracılığıyla sağa döndür

Dizim: [etiket] RRF f,d

Operandlar:  $0 \leq f \leq 127$   $d \in [0,1]$

İşlem: Aşağıdaki tanıma bakınız.

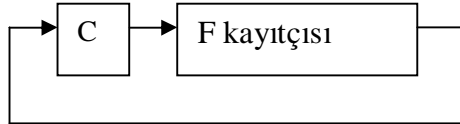
Etkilenen bayraklar: C

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 1100 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım:f kayıtçısının içeriği elde biti (C) içinden bir bit sağa döndürülür. C biti f kayıtçısının LSB bitini taşır. Eğer d=0 ise, sonuç W kayıtçısının içerisine yerleştirilir. Eğer d=1 ise, sonuç f kayıtçısının içerisinde geri depolanır.



Hafıza alanı (Bayt):1

Saat pılsı: 1

Örnek: RRF REG1,0

Komuttan önce;

REG1 = b'1110 0110'

C = 0

Komuttan sonra;

REG1 = b'1110 0110'

W = b'0111 0011'

C = 0

**SUBWF** F kayıtcısından W kayıtcısını çıkart.

Dizim: [etiket] SUBWF f,d

Operandlar:  $0 \leq f \leq 127$   $d \in [0,1]$

İşlem:  $(f) - (W) \rightarrow (\text{hedef})$

Etkilenen bayraklar: C,DC,Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0010 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtcı adresi) tanımlar.

Tanım: W kayıtcısı f kayıtcısından çıkartılır. d=1 ise, sonuç f kayıtcısı içerisinde geri saklanır.

Hafıza alanı (Bayt): 1

Saat pılsı: 1

Örnek 1: SUBWF REG1,1

Komuttan önce;

REG1 = 3

W = 2

C = ?

Z = ?

Komuttan sonra;

REG1 = 1

W = 2

C = 1, sonuç pozitif

Z = 0

**Örnek 2:** Komuttan önce;

REG1 = 2

W = 2

C = ?

Z = ?

Komuttan sonra;

REG1 = 0

W = 2

C = 1, sonuç sıfır

Z = 1

**Örnek 3:** Komuttan önce;

REG1 = 1

W = 2

C = ?

Z = ?

Komuttan sonra;

REG1 = FF

W = 2

C = 0, sonuç negatif

Z = 1

## **SWAPF** f'yi takas et

Dizim: [etiket] SWAPF f,d

Operandlar:  $0 \leq k \leq 255$   $d \in [0,1]$

İşlem:  $(f \langle 3:0 \rangle) \rightarrow (\text{hedef} \langle 7:4 \rangle)$ ,  $(f \langle 7:4 \rangle) \rightarrow (\text{hedef} \langle 3:0 \rangle)$

Yani f kayıtcısının LSB tarafındaki 4 biti (1.Dijiti) ile MSB tarafındaki 4 biti (2.dijiti) yer değiştirir. Eğer d=0 ise, sonuç W kayıtcısının içerisine yerleştirilir. Eğer d=1 ise, sonuç f kayıtcısının içerisinde geri depolanır.

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 1110 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtcı adresi) tanımlar.

Tanım:f kayıtcısının yüksek ve alçak baytları dönüştürülür. d=0 ise,sonuç W kayıtcısı içine yerleştirilir. Eğer d=1 ise, sonuç f kayıtcısı içine yerleştirilir.

Hafıza alanı (Bayt):1

Saat palsı: 1

Örnek: SWAP F ISIK, 0

Komuttan önce;

ISIK = h'A5' olsun .

Komuttan sonra;

ISIK = h'A5'

W = h'5A' olur.

## **XORWF** f ile W'nin içeriğini mantıksal EXOR'la.

Dizim: [etiket] XORWF f,d

Operandlar:  $0 \leq k \leq 127$   $d \in [0,1]$

İşlem: (W) .XOR. (f)  $\rightarrow$  (hedef)

Etkilenen bayraklar: Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0110 | dfff | ffff |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , d biti hedef biti , f ile belirtilen LSB tarafındaki 5 bit ise dosya adresini (f kayıtcı adresi) tanımlar.

Tanım:W kayıtcısının içeriği, f kayıtcısı ile mantıksal EXOR'lanır. EXOR mantığında aynı bitlerde çıkış 0, farklı bitlerde çıkış 1 olur. Eğer d=0 ise, sonuç W kayıtcısı içinde depolanır, d=1 ise sonuç f kayıtcısı içine geri yerleştirilir.

Hafıza alanı (Bayt):1

Saat palsı: 1

Örnek: XORWF REG , 1

Komuttan önce;

REG = h'AF

W = h'B5

Komuttan sonra;

h'AF' = b'1010 1111'

EXOR h'B5' = b'1011 0101'

b '0001 1010' = h'1A' ve ;

REG = h'1A

W = h'B5

### 1.3.2. Bit Yönlendirmeli Komutlar

Bit Yönlendirmeli komutlarda dikkat edilirse hedef (d) belirtilmemiştir. Sonuç her zaman f kayıtcısının içersindedir. Şimdi bu komutları görelim;

|                                                                                                                                                                        |                                                                                       |      |      |      |      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|------|------|------|------|
| <b>BCF</b>                                                                                                                                                             | F kayıtcısının belirtilen bitini sıfırla.                                             |      |      |      |      |
| Dizim:                                                                                                                                                                 | [etiket] BCF f,b                                                                      |      |      |      |      |
| Operandlar:                                                                                                                                                            | $0 \leq f \leq 127$                                                                   |      |      |      |      |
| İşlem:                                                                                                                                                                 | $0 \rightarrow (f \langle b \rangle)$                                                 |      |      |      |      |
| Etkilenen bayraklar:                                                                                                                                                   | Yok                                                                                   |      |      |      |      |
| Kodlama:                                                                                                                                                               | <table border="1"><tr><td>01</td><td>00bb</td><td>bfff</td><td>ffff</td></tr></table> | 01   | 00bb | bfff | ffff |
| 01                                                                                                                                                                     | 00bb                                                                                  | bfff | ffff |      |      |
| MSB tarafındaki 4 bit komut kodu (opcode) , 3 tane b biti hedef biti belirler , f ile belirtilen LSB tarafındaki 7 bit ise dosya adresini (f kayıtcı adresi) tanımlar. |                                                                                       |      |      |      |      |
| Tanım:f içindeki 'b.' biti silinir.                                                                                                                                    |                                                                                       |      |      |      |      |
| Hafıza alanı (Bayt):                                                                                                                                                   | 1                                                                                     |      |      |      |      |
| Saat palsı:                                                                                                                                                            | 1                                                                                     |      |      |      |      |

**Örnek:** BCF TEST,7

Komuttan önce;

TEST = h'C7' olsun.

Komuttan sonra;

h'C7' = b' 1100 0111'



0 olur ve;

TEST = h'47' olur.

|                                                                                                                                                                        |                                                                                       |      |      |      |      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|------|------|------|------|
| <b>BSF</b>                                                                                                                                                             | F kayıtcısının belirtilen bitini kur (birle).                                         |      |      |      |      |
| Dizim:                                                                                                                                                                 | [etiket] BSF f,b                                                                      |      |      |      |      |
| Operandlar:                                                                                                                                                            | $0 \leq f \leq 127$ $0 \leq b \leq 7$                                                 |      |      |      |      |
| İşlem:                                                                                                                                                                 | $1 \rightarrow (f \langle b \rangle)$                                                 |      |      |      |      |
| Etkilenen bayraklar:                                                                                                                                                   | Yok                                                                                   |      |      |      |      |
| Kodlama:                                                                                                                                                               | <table border="1"><tr><td>01</td><td>01bb</td><td>bfff</td><td>ffff</td></tr></table> | 01   | 01bb | bfff | ffff |
| 01                                                                                                                                                                     | 01bb                                                                                  | bfff | ffff |      |      |
| MSB tarafındaki 4 bit komut kodu (opcode) , 3 tane b biti hedef biti belirler , f ile belirtilen LSB tarafındaki 7 bit ise dosya adresini (f kayıtcı adresi) tanımlar. |                                                                                       |      |      |      |      |
| Tanım:f kayıtcısı içindeki 'b.' Bit 1'e kurulur.                                                                                                                       |                                                                                       |      |      |      |      |
| Hafıza alanı (Bayt):                                                                                                                                                   | 1                                                                                     |      |      |      |      |
| Saat palsı:                                                                                                                                                            | 1                                                                                     |      |      |      |      |

**Örnek:** BSF TEST, 7

Komuttan önce;

TEST = h'0A' olsun.

Komuttan sonra;

h'0A' = b' 0000 1010'



1 olur ve; TEST = h'1A' olur

**BTFS** f kayıtçısının belirlenen biti 0 ise bundan sonraki komutu atla.

---

Dizim: [etiket] BTFS f,b

Operandlar:  $0 \leq f \leq 127$   $0 \leq b \leq 7$

İşlem:  $F(\langle b \rangle) = 0$  ise atla

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 01 | 01bb | bfff | ffff |
|----|------|------|------|

MSB tarafındaki 4 bit komut kodu (opcode), 3 tane b biti hedef biti belirler, f ile belirtilen LSB tarafındaki 7 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: f kayıtçısı içindeki 'b.' Bitin 0 olup olmadığı test edilir. Eğer 'b.' bit 0 ise, bu komutun altındaki komut işlenmez bir sonraki komuta sapılır. Aksi durumda ise sıradaki komut işlenir.

Hafıza alanı (Bayt): 1

Saat palsı:

1(2)

Test edilen bit 1 ise 1 saat palsı, 0 ise 2 saat palsı alır.

Örnek: BASLA BTFS PORTB,1 ;B portunun 1.bitini test et.

GOTO BASLA ;0 değilse bu satır işlenir, BASLA etiketine dallanır.  
;Yani 0 olana kadar test etmeye devam edecek.

BSF PORTB,1 ;0 ise bu satır işlenir ve B portunun 1. biti bu komut ile  
;1 yapılır. Eğer bu pine bir led bağlı ise yanar.

**BTFS** f kayıtçısının belirlenen biti 1 ise bundan sonraki komutu atla.

---

Dizim: [etiket] BTFS f,b

Operandlar:  $0 \leq f \leq 127$   $0 \leq b \leq 127$

İşlem: Eğer  $(f \langle b \rangle) = 1$  ise atla

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 01 | 11bb | bfff | ffff |
|----|------|------|------|

MSB tarafındaki 4 bit komut kodu (opcode), 3 tane b biti hedef biti belirler, f ile belirtilen LSB tarafındaki 7 bit ise dosya adresini (f kayıtçı adresi) tanımlar.

Tanım: f kayıtçısı içindeki 'b.' Bitin 1 olup olmadığı test edilir. Eğer 'b.' bit 1 ise, bu komutun altındaki komut işlenmez bir sonraki komuta sapılır. Aksi durumda ise sıradaki komut işlenir.

Hafıza alanı (Bayt): 1

Saat palsı: 1(2)

Test edilen bit 0 ise 1 saat palsı, 1 ise 2 saat palsı alır.

Örnek: BASLA BTFS PORTA,1 ;A portunun 1.bitini test et.

GOTO BASLA ;1 değilse bu satır işlenir, BASLA etiketine dallanır.  
;Yani 1 olana kadar test etmeye devam edecek.

BSF PORTB,1 ;1 ise bu satır işlenir ve B portunun 1. biti bu komut ile 1  
;yapılır. Eğer bu pine bir led bağlı ise yanar.

### 1.3.3. Sabit İşleyen Komutlar

Sabit yönlendirmeli komutlarda dikkat edilirse hedef (d) belirtilmemiştir. Sonuç her zaman W kayıtçısının içersindedir. Ayrıca bu gruptaki komutların W dışındaki kayıtçılar ile kullanılmadığına dikkat ediniz. Şimdi bu komutları görelim;

|                      |                                                                                       |      |      |      |      |
|----------------------|---------------------------------------------------------------------------------------|------|------|------|------|
| <b>ANDLW</b>         | W ile birlikte 8 bitlik k sabitini mantıksal AND'le.                                  |      |      |      |      |
| Dizim:               | [etiket] ANDLW k                                                                      |      |      |      |      |
| Operandlar:          | $0 \leq k \leq 255$                                                                   |      |      |      |      |
| İşlem:               | $(W).AND.(k) \rightarrow (W)$                                                         |      |      |      |      |
| Etkilenen bayraklar: | Z                                                                                     |      |      |      |      |
| Kodlama:             | <table border="1"><tr><td>11</td><td>1001</td><td>kkkk</td><td>kkkk</td></tr></table> | 11   | 1001 | kkkk | kkkk |
| 11                   | 1001                                                                                  | kkkk | kkkk |      |      |

MSB tarafındaki 5 bit komut kodu , k ile belirtilen LSB tarafındaki 8 bit ise 1 baytlık sabit değeri tanımlar. Burada hedef (d) in kodlanmadığına dikkat ediniz.

Tanım:W kayıtçısının içeriği 8-bitlik sabit 'k' ile mantıksal AND'lenir. Sonuç,W kayıtçısına geri konur.

Hafıza alanı (Bayt): 1  
Saat palsi: 1  
Örnek: ANDLW h'5F'

Komuttan önce;  
 $W = h'A3'$   
Komuttan sonra;  
 $h'A3' = b'1010\ 0011'$   
AND h'5F' = b'0101\ 1111'  
 $b'0000\ 0011' = h'03'$  ve ;  
 $W = h'03'$  olur.

|                      |                                                                                       |      |      |      |      |
|----------------------|---------------------------------------------------------------------------------------|------|------|------|------|
| <b>ADDLW</b>         | W ile birlikte 8 bitlik k sabitini topla.                                             |      |      |      |      |
| Dizim:               | [etiket] ADDLW k                                                                      |      |      |      |      |
| Operandlar:          | $0 \leq k \leq 255$                                                                   |      |      |      |      |
| İşlem:               | $(W)+k \rightarrow (W)$                                                               |      |      |      |      |
| Etkilenen bayraklar: | C,DC,Z                                                                                |      |      |      |      |
| Kodlama:             | <table border="1"><tr><td>11</td><td>111x</td><td>kkkk</td><td>kkkk</td></tr></table> | 11   | 111x | kkkk | kkkk |
| 11                   | 111x                                                                                  | kkkk | kkkk |      |      |

MSB tarafındaki 5 bit komut kodu , k ile belirtilen LSB tarafındaki 8 bit ise 1 baytlık sabit değeri tanımlar.Burada hedef (d) in kodlanmadığına dikkat edin.

Hafıza alanı (Bayt): 1  
Saat palsi: 1  
Örnek: ADDLW h'15'

Komuttan önce;  
 $W = h'10'$   
Komuttan sonra;  $W = h'25'$  olur.

**IORLW** W ile birlikte kapsayan OR yazımı

Dizim: [etiket] IORLW k

Operandlar:  $0 \leq k \leq 255$

İşlem: (W).OR.(k) → (W)

Etkilenen bayraklar: Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 11 | 1000 | kkkk | kkkk |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , k ile belirtilen LSB tarafındaki 8 bit ise 1 baytlık sabit değeri tanımlar. Burada hedef (d) in kodlanmadığına dikkat ediniz.

Tanım: W kayıtçısının içeriği 8-bitlik sabit 'k' ile OR'lanır. Sonuç, W kayıtçısına geri konur.

Hafıza alanı (Bayt): 1

Saat palsı: 1

Örnek: IORLW h'35'

Komuttan önce;

W = h'9A'

Komuttan sonra;

h'9A' = b'1001 1010'

OR h'35' = b'0011 0101'

b'1011 1111' = h'BF' ve ; W = h'BF' ve Z=1 olur.

**MOVLW** W kayıtçısına bir sayı/sabit yükle.

Dizim: [etiket] MOVLW k

Operandlar:  $0 \leq k \leq 255$

İşlem: k → (W)

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 11 | 00xx | kkkk | kkkk |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , k ile belirtilen LSB tarafındaki 8 bit ise 1 baytlık sabit değeri tanımlar. Burada hedef (d) in kodlanmadığına dikkat ediniz.

Tanım: 8-bitlik sabit 'k', W kayıtçısına yüklenir.

Hafıza alanı (Bayt): 1

Saat palsı: 1

Örnek: MOVLW h'5A'

Komuttan önce;

W = h'9A' olsun ;

Komuttan sonra;

W = h'5A' olur.

**RETLW** Altprogramdan W'ye bir sayı/sabit yükle ve geri dön.

Dizim: [etiket] RETLW k

Operandlar:  $0 \leq k \leq 255$

İşlem: (k) → (W)

TOS → (PC)

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 11 | 01xx | kkkk | kkkk |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , k ile belirtilen LSB tarafındaki 8 bit ise 1 baytlık sabit değeri tanımlar. Burada hedef (d) in kodlanmadığına dikkat ediniz.

Tanım:Altprogramdan W'ye bir k sayı/sabiti yükleyerek dönmek için kullanılır. Dönüş TOS (yığının üstü- yani geri dönüş adresinin bulunduğu hafıza alanı) değerinin PC (program sayıcı) ye aktarılmasıyla yapılır. Bu komut daha çok çevrim tablolarında kullanılır. İlerde detaylı işlenecektir.

Hafıza alanı (Bayt): 1

Saat palsı: 2

Örnek: RETLW h'3F'

Komuttan önce W'nin içeriği ne olursa olsun komuttan sonra W=h'3F' yapılır ve anaprograma geri dönlür.

**SUBLW** Bir k sayı/sabitten W'yi çıkart.

Dizim: [etiket] SUBLW k

Operandlar:  $0 \leq k \leq 255$

İşlem:  $k - (W) \rightarrow (W)$

Etkilenen bayraklar: C,DC,Z

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 11 | 110x | kkkk | kkkk |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , k ile belirtilen LSB tarafındaki 8 bit ise 1 baytlık sabit değeri tanımlar. Burada hedef (d) in kodlanmadığına dikkat ediniz.

Tanım:W kayıtçısı (2'nin tümleyen metodu ile) sekiz bitlik 'k' yazımından çıkartılır. Sonuç, W kayıtçısı içerisine yerleştirilir.

Hafıza alanı (Bayt):1

Saat palsı: 1

Örnek 1:

SUBLW h'02'

Komuttan önce;

W = 1

C = ?

Z = ?

Komuttan sonra;

W = 1

C = 1, sonuç pozitif

Z = 0

Örnek 2:Komuttan önce;

W = 2

C = ?

Z = ?

Komuttan sonra;

W = 0

C = 1, sonuç sıfır

Z = 1

Örnek 3:Komuttan önce;

W = 3

C = ?

Z = ?

Komuttan sonra;

W = FF



C = 0, sonuç negatif  
Z = 1

**XORLW** Bir k sayı/sabit ile W'yi mantıksal EXOR'la.

Dizim: [etiket] XORLW k  
Operandlar:  $0 \leq k \leq 255$   
İşlem: (W) .XOR. k  $\rightarrow$  (W)  
Etkilenen bayraklar: Z  
Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 11 | 1010 | kkkk | kkkk |
|----|------|------|------|

MSB tarafındaki 5 bit komut kodu , k ile belirtilen LSB tarafındaki 8 bit ise 1 baytlık sabit değeri tanımlar. Burada hedef (d) in kodlanmadığına dikkat ediniz.

Tanım: W kayıtçısının içeriği, sekiz bitlik 'k' yazımı ile mantıksal EXOR'lanır. EXOR mantığında aynı bitlerde çıkış 0, farklı bitlerde çıkış 1 olur. Sonuç W kayıtçısı içine yerleştirilir.

Hafıza alanı (Bayt): 1  
Saat palsı: 1  
Örnek: XORLW h'AF'

Komuttan önce;  
W = h'B5' olsun.  
Komuttan sonra;  
h'AF' = b'1010 1111'  
EXOR h'B5' = b'1011 0101'  
b '0001 1010' = h'1A' ve ;  
W = h'1A' olur.

#### 1.3.4. Kontrol Komutları

Kontrol komutları program akışını belirleyen komutlardır. Altprogram çağırma, şartsız dallanma ve uyku moduna geçme gibi olaylar bu komutlarla gerçekleşir. Bu komutlarda hedef genelde bir adres değeridir. Şimdi bu komutları görelim;

**CALL** Altprogram çağır .

Dizim: [etiket] CALL k  
Operandlar:  $0 \leq k \leq 2047$   
İşlem: (PC)+1  $\rightarrow$  TOS,  
PC: Program sayıcı (program counter)  
TOS: Yığının üstü (Top Of Stack)  
k  $\rightarrow$  (PC<10:0>),  
(PCLATH<4:3>)  $\rightarrow$  (PC<12:11>)  
Etkilenen bayraklar: Yok  
Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 10 | 0kkk | kkkk | kkkk |
|----|------|------|------|

MSB tarafındaki 3 bit komut kodu , k ile belirtilen LSB tarafındaki 11 bit ise altprogram başlangıç adresini tanımlar. Burada hedef (d) in kodlanmadığına dikkat ediniz.

Tanım: Altprogramı çağırır. İlk olarak, geri-dönüş adresi (PC+1) yığına itilir. Onbir bitlik altprogramın başlangıç adresi , PC bitleri <10:0> içerisine yüklenir. PC'nin üst bitleri PC<12:11> , PCLATH'in <4:3> bitlerinden yüklenir. CALL mutlaka RETURN komutu ile birlikte kullanılmalıdır. RETURN komutu ilerde anlatılacaktır.

Hafıza alanı (Bayt):1

Saat palsı: 2

Örnek: DON CALL SAYAC

DON'un bir etiket olduğuna aynı zamanda CALL komutunun şu anki adresini tuttuğuna dikkat ediniz. SAYAC ise altprogramın başlangıç adresini tutan etikettir. Bu durumda ;

Komuttan önce;

PC = {Adres DON}

Komuttan sonra;

PC = {Adres SAYAC}

TOS = {Adres DON} olur.

**CLRWDT** Bekçi köpeği zamanlayıcısını (Watchdog Timer) sil.

---

Dizim: [etiket] CLRWDT

Operandlar: Yok

İşlem: 00h → WDT

0 → WDT ön-ölçücüsü

1 →  $\overline{T0}$

1 →  $\overline{PD}$

Etkilenen bayraklar:  $\overline{T0}$  ,  $\overline{PD}$

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0000 | 0110 | 0100 |
|----|------|------|------|

Tanım: CLRWDT komutu, Watchdog Timer'ı reset eder. Bu komut aynı zamanda WDT'nin ön bölücüsünün de resetlenmesine sebep olur. T0 ve PD durum bitleri de set edilir. Bu bitlerin işlevleri için status kayıtçısı konusuna bakınız.

Hafıza alanı (Bayt):1

Saat palsı: 1

Örnek: CLRWDT

Komuttan önce;

WDT sayacı = ?

Komuttan sonra;

WDT sayacı = h'00'

WDT ön-bölücü değeri = 0

$\overline{T0}$  = 1

$\overline{PD}$  = 1

**GOTO** Adres'e git

---

Dizim: [etiket] GOTO k

Operandlar: 0≤k≤2047

İşlem:  $k \rightarrow (PC<10:0>)$   
(PCLATH <4:3>) $\rightarrow$  (PC<12:11>)

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 10 | 1kkk | kkkk | kkkk |
|----|------|------|------|

Tanım: GOTO, koşulsuz bir sapma komutudur. K nın belirlediği 11-bit'lik adres PC bitlerinin <10:0> içerisine yüklenir. PC'm üst bitleri (PC<12:11>), PCLATH<4:3>'ten yüklenir ve oluşan adrese sapılır.

Hafıza alanı (Bayt):1

Saat palsı: 2

Örnek: GOTO SAYAC

Komuttan sonra;

PC = {Adres SAYAC} olur.

**RETFIE** Kesme altprogramından geri dön.

---

Dizim: [etiket] RETFIE

Operandlar: Yok

İşlem: TOS  $\rightarrow$  (PC) ; 1  $\rightarrow$  GIE TOS da geri dönüş adresi bulunur.

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0000 | 0000 | 1001 |
|----|------|------|------|

Tanım: Kesme altprogramından geri dönmek için kullanılır. Yığında (TOS) bulunan geri dönüş adresi PC'ye yüklenir. INTCON kesme kayıtçısının GIE (Genel kesme yetkisi) biti set edilir. Bu, iki döngülük bir komuttur.

Hafıza alanı (Bayt):1

Saat palsı: 2

Örnek: RETFIE

Komuttan sonra;

PC = TOS

GIE = 1

**RETURN** Altprogramdan geri dön.

---

Dizim: [etiket] RETURN

Operandlar: Yok

İşlem: TOS  $\rightarrow$  (PC)

Etkilenen bayraklar: Yok

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0000 | 0000 | 0000 |
|----|------|------|------|

Tanım: Altprogramdan geri dönmeyi sağlar. Yığında (TOS) bulunan geri dönüş adresi PC'ye yüklenir. Bu, iki döngülük bir komuttur.

Hafıza alanı (Bayt):1

Saat palsı: 2

Örnek: RETURN

Komuttan sonra;

PC= TOS; yani PC'ye geri dönüş adresi yüklenir ve ana programa yüklenir.

**SLEEP** Uyku (Standby) moduna gir.

Dizim: [etiket] SLEEP

Operandlar: Yok

İşlem: 00h → WDT

0 →  $\overline{\text{WDT}}$  ön-bölücüsü

1 → TO

0 →  $\overline{\text{PD}}$

Etkilenen bayraklar:  $\overline{\text{TO}}$ ,  $\overline{\text{PD}}$

Kodlama: 

|    |      |      |      |
|----|------|------|------|
| 00 | 0000 | 0110 | 0011 |
|----|------|------|------|

Tanım: Güç yok durum biti PD (Power down) sıfırlanır. Süre aşımı TO (Time-out) durum biti ise 1'e kurulur. Watchdog zamanlayıcısı ve ön-bölücüsü silinir. Osilatörün durmasıyla işlemci SLEEP moduna girer. PIC bu durumda çok az güç harcar. Arada bir kontrol gereken güvenlik işlerinde, ya da belirli sürelerde yapılacak işler bittiğinde PIC uyuma moduna sokulur.

Hafıza alanı (Bayt): 1

Saat palsı: 1

Örnek: SLEEP

## 1.4. Sayı ve Karakterlerin Yazılışı

PIC assembly komutlarında sayılar heksadesimal, binary veya desimal formda kullanılabilir. Değişik kaynaklarda kullanılan sayı ve karakter gösteriliş biçimleriyle karşılaştığınızda bunları okuyabilmeniz için aşağıda örnekler verilmiştir.

### 1.4.1. Heksadesimal Sayılar

Heksadesimal sayılar '0x', '0' veya 'h' harfleriyle başlamalıdır. Örneğin, STATUS kayıtçısına 03 adresini atamak için aşağıda gösterilen yazılış biçimleri kullanılabilir.

|        |     |       |
|--------|-----|-------|
| STATUS | EQU | 0x03  |
|        | EQU | 03h   |
|        | EQU | h'03' |

MOVLW komutu ile W kayıtçısı içerisine yüklenecek olan FF heksadesimal sabiti ise aşağıdaki gibi yazılabilir;

MOVLW 0 x FF veya MOVLW h ' FF'

Biz hexadesimal formatı h ' xx' şeklinde kullanacağız.

### 1.4.2. Binary Sayılar

Binary sayılar b harfi ile başlamalıdır. Örneğin 00001010 binary sayısını W kayıtçısı içerisine yüklemek için aşağıdaki gibi yazılmalıdır.

MOVLW b'00001010'

### 1.4.3. Desimal Sayılar

Desimal sayıların başına d harfi konularak tırnak içerisinde yazılırlar. Örneğin 15 desimal sayısı W kayıtcısı içerisinde yüklemek için aşağıdaki gibi yazılmalıdır.

```
MOVLW d'15'
```

### 1.4.4. ASCII Karakterler

Genellikle RETLW komutu ile birlikte kullanılan ASCII karakterler tırnak içerisinde alınarak aşağıdaki gibi yazılırlar.

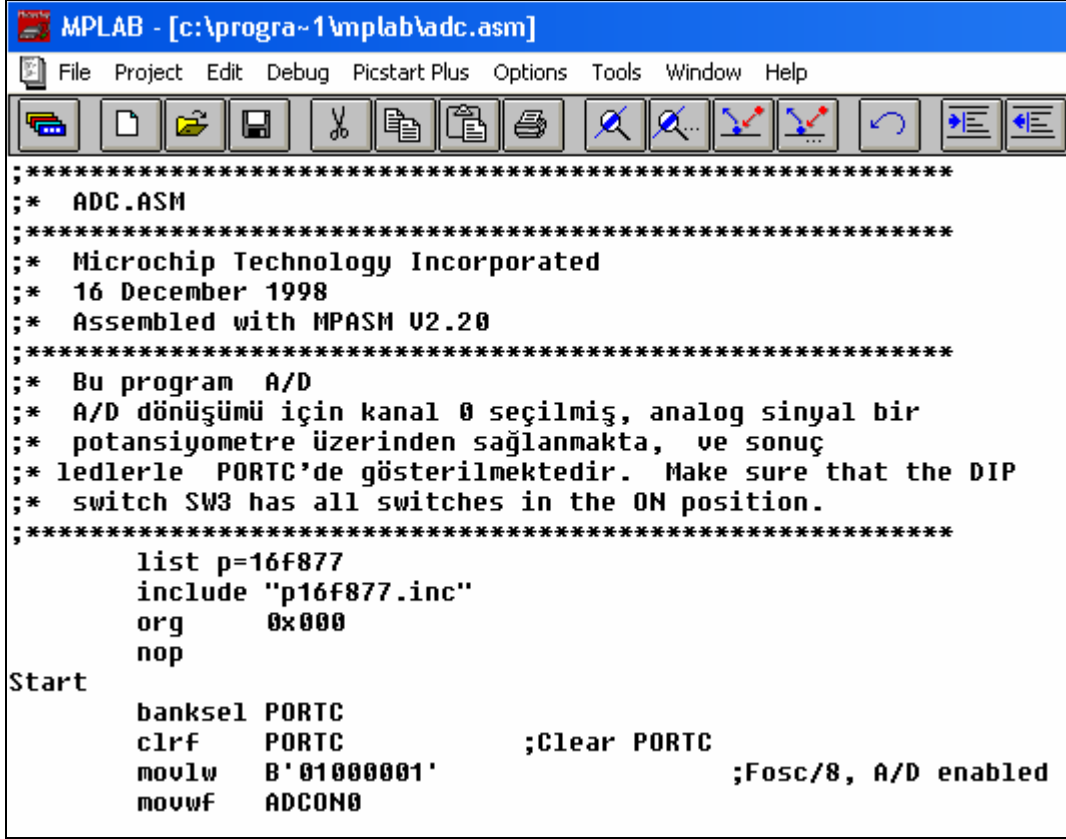
```
RETLW 'A'
RETLW 'T'
```

## 1.5. Mikrodenetleyici İçin Gerekli Yazılımın Kullanımı

Mikrodenetleyici programlamak için (Burada PIC16F84) yazılım ve donanım ihtiyaçlarımız olacaktır. Yazılım olarak;

- Ø ASM uzantılı dosyamızı yazabilmek için bir metin editörü,
- Ø ASM dosyamızı makine koduna (HEX dosyası) çevirecek derleyici program
- Ø HEX dosyasını bir programlama kartına yükleyecek PIC programlayıcı yazılımı ihtiyacı olacaktır.

Metin editörü olarak Windows'un kendi NOTPAD programını veya MPLAP içerisindeki kendi editörünü de kullanabilirsiniz. Resim 1.1'de MPLAP'nin ASM editör arabirimini görüyorsunuz.



```
MPLAB - [c:\progra~1\mplab\adc.asm]
File Project Edit Debug Picstart Plus Options Tools Window Help

;* ADC.ASM

;* Microchip Technology Incorporated
;* 16 December 1998
;* Assembled with MPASM U2.20

;* Bu program A/D
;* A/D dönüşümü için kanal 0 seçilmiş, analog sinyal bir
;* potansiyometre üzerinden sağlanmakta, ve sonuç
;* ledlerle PORTC'de gösterilmektedir. Make sure that the DIP
;* switch SW3 has all switches in the ON position.

list p=16f877
include "p16f877.inc"
org 0x000
nop
Start
banksel PORTC
clrf PORTC ;Clear PORTC
movlw B'01000001' ;Fosc/8, A/D enabled
movwf ADCON0
```

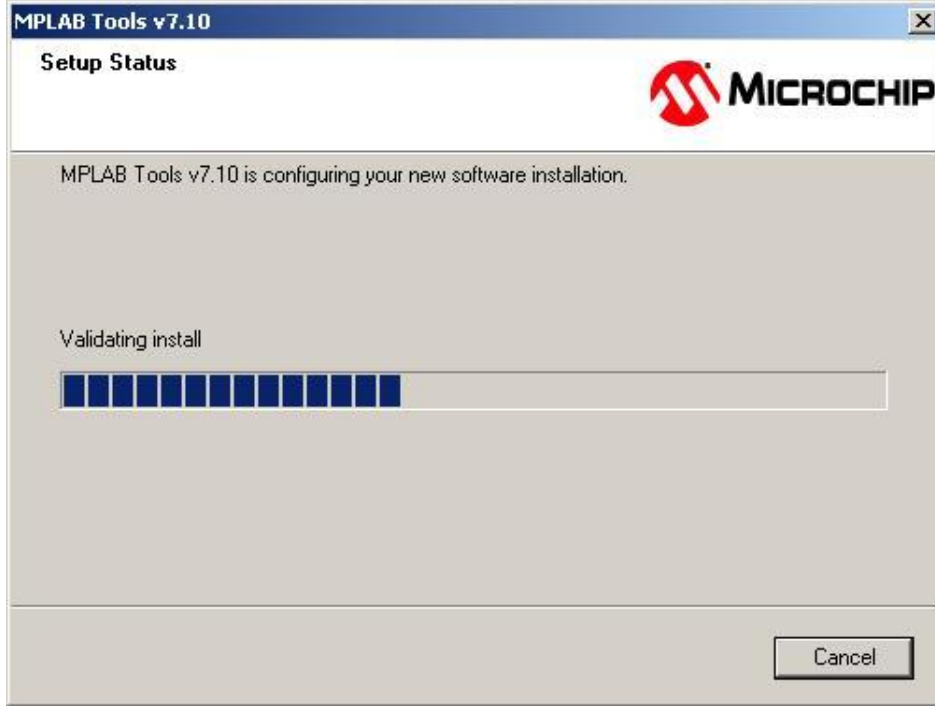
Resim 1.1: MPLAB'nin ASM editör arabirimi

Donanım olarak ise IBM uyumlu bir bilgisayara ve kullandığımız mikrodenetleyiciyi programlayabilecek programlayıcı karta ihtiyaç vardır. Bu kartın yapılması ve programlayıcının nasıl kullanacağını “mikroişlemci ve mikrodenetleyiciler” modülünde gördüğümüzden bu modülde bunlardan bahsetmeyeceğiz. Biz burada sadece derleme (compiler) olayının nasıl yapıldığına değineceğiz.

### 1.5.1. Programın Kurulması

Microchip firmasının MPASM derleyicisi ile yazmış olduğumuz assembly dosyalarımızı , mikrodenetleyiciye yüklemek üzere .HEX uzantılı dosyalara çevirebiliriz. Bu programı Microchip firmasının internet sitesinden ücretsiz dağıttığı MPLAB programının içinde bulabilirsiniz. Bu program yaklaşık 30MB'lık dosyadır. (MPLAB710.zip)

Gelen zip dosyasını açın ve içindeki Setup.exe dosyasını çalıştırarak yükleyin. Resim 1.2'de kurulumu görüyorsunuz.



Resim 1.2: MPLAB kurulumu

Yükleme işlemi bittikten sonra, başlat menüsünden programlarım içinde MPASM programını açtığınızda karşınıza Resim 1.3'deki pencere gelecek .



Resim 1.3: MPASM derleyici programı arayüzü

### 1.5.2. Menülerin Tanıtılması

Şimdi de MPASM derleyicisinin ayarlarını görelim;

- Ø “Source File Name ” ile derlenecek ASM uzantılı kaynak dosya seçilir. Bunun için Browse tıklanarak kaynak dosya seçilir.
- Ø “Radix” radyo düğmelerinden “hexadecimal” seçilmelidir.
- Ø “Warning level” , ikaz seviyesini ayarlayabileceğimiz bir ayar grubudur. Programın hangi durumlarda ikaz vereceğini belirler.Tercihen ‘Warning and errors’ seçilmelidir.
- Ø “Hex Output”, derleme sonucunda oluşacak hex dosyasının çıkış formatını belirler. Tercihen ‘INHX8M’ seçilmelidir.
- Ø “Generated Files”, derleme sonucunda hangi dosyaların oluşturulacağını belirler. Tercihen ‘Error File’ ve ‘List File’ seçilmelidir. Bir problem durumunda bu dosyalar incelenerek hatalar giderilebilir.
- Ø “Macro expansions”, makro uzantılarının kabul edilip/edilmemesini belirleyen bu ayarı tercihen default olmalıdır.
- Ø “Processor”, ile kullanılacak mikrodenetleyici seçilir.
- Ø “Tab size” , sekme uzunluğu değeri 8 seçilebilir.
- Ø “Extra Option”, özel parametreler girmek için kullanılır. Örneğin buraya /LTEST.LST yazarsanız derleme sonucunda HEX dosyası ile beraber TEST.LST dosyasının da oluştuğunu görürsünüz.

### 1.5.3. Mikrodenetleyici ve Diğer Donanımların Seçilmesi

Mikrodenetleyiciler ile bir proje tasarlanırken seçilecek mikrodenetleyicinin seçimi çok önemlidir. Mikrodenetleyicilerin komut sayıları, kullandıkları kayıtçılar, bellek değerleri vs. birbirinden farklıdır. Bu yüzden öncelikle kullanılacak mikrodenetleyici seçilmelidir. Bunun için mikrodenetleyicilerin katalogları incelenmelidir. Microchip firmasının sitesinde üretilen tüm mikrodenetleyicilerin (PIC serisi) kataloglarına ulaşabilirsiniz. Biz bu modülde daha çok PIC16F84 mikrodenetleyicisinin programlanmasını göreceğiz. Yalnız Analog/dijital dönüşümü konusunda, PIC16F84 denetleyicisinin dâhil ADC modülü olmadığından, bunun için 16F877 denetleyicisini kullanacağız. Şekil 1.3 ve Şekil 1.4’te en çok kullanılan denetleyicilerin bazı teknik verileri verilmiştir.



|                          | Clock                                | Memory |        |     |                     | Peripherals         | Features        |                   |          |                       |                  |
|--------------------------|--------------------------------------|--------|--------|-----|---------------------|---------------------|-----------------|-------------------|----------|-----------------------|------------------|
|                          | Maximum Frequency of Operation (MHz) | Flash  | EEPROM | ROM | Data Memory (bytes) | Data EEPROM (bytes) | Timer Module(s) | Interrupt Sources | I/O Pins | Voltage Range (Volts) | Packages         |
| PIC16C84                 | 10                                   | —      | 1K     | —   | 36                  | 64                  | TMR0            | 4                 | 13       | 2.0-6.0               | 18-pin DIP, SOIC |
| PIC16F84 <sup>(1)</sup>  | 10                                   | 1K     | —      | —   | 68                  | 64                  | TMR0            | 4                 | 13       | 2.0-6.0               | 18-pin DIP, SOIC |
| PIC16CR84 <sup>(1)</sup> | 10                                   | —      | —      | 1K  | 68                  | 64                  | TMR0            | 4                 | 13       | 2.0-6.0               | 18-pin DIP, SOIC |
| PIC16F83 <sup>(1)</sup>  | 10                                   | 512    | —      | —   | 36                  | 64                  | TMR0            | 4                 | 13       | 2.0-6.0               | 18-pin DIP, SOIC |
| PIC16CR83 <sup>(1)</sup> | 10                                   | —      | —      | 512 | 36                  | 64                  | TMR0            | 4                 | 13       | 2.0-6.0               | 18-pin DIP, SOIC |

Şekil 1.3: 18 pinli bazı denetleyicilerin teknik verileri

| Key Features<br>PICmicro™ Mid-Range Reference<br>Manual (DS33023) | PIC16F873               | PIC16F874               | PIC16F876               | PIC16F877               |
|-------------------------------------------------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| Operating Frequency                                               | DC - 20 MHz             | DC - 20 MHz             | DC - 20 MHz             | DC - 20 MHz             |
| Resets (and Delays)                                               | POR, BOR<br>(PWRT, OST) | POR, BOR<br>(PWRT, OST) | POR, BOR<br>(PWRT, OST) | POR, BOR<br>(PWRT, OST) |
| FLASH Program Memory<br>(14-bit words)                            | 4K                      | 4K                      | 8K                      | 8K                      |
| Data Memory (bytes)                                               | 192                     | 192                     | 368                     | 368                     |
| EEPROM Data Memory                                                | 128                     | 128                     | 256                     | 256                     |
| Interrupts                                                        | 13                      | 14                      | 13                      | 14                      |
| I/O Ports                                                         | Ports A,B,C             | Ports A,B,C,D,E         | Ports A,B,C             | Ports A,B,C,D,E         |
| Timers                                                            | 3                       | 3                       | 3                       | 3                       |
| Capture/Compare/PWM modules                                       | 2                       | 2                       | 2                       | 2                       |
| Serial Communications                                             | MSSP, USART             | MSSP, USART             | MSSP, USART             | MSSP, USART             |
| Parallel Communications                                           | —                       | PSP                     | —                       | PSP                     |
| 10-bit Analog-to-Digital Module                                   | 5 input channels        | 8 input channels        | 5 input channels        | 8 input channels        |
| Instruction Set                                                   | 35 Instructions         | 35 Instructions         | 35 Instructions         | 35 Instructions         |

Şekil 1.4: 40 pinli bazı denetleyicilerin teknik verileri

Yapılacak işe, kullanılan denetleyicinin teknik özelliklerine göre kullanılacak diğer donanımlar (ADC /DAC dönüştürücüler, LCD display, sensörler vs. ) seçilmelidir.

## 1.6. Programlama Tekniği

Program yazarken, 4 temel kural izlenmelidir.

- Ø Yazılım hakkında daima açıklamalar yapılmalıdır. Yoksa kaçınılmaz olarak geri dönüldüğünde, neyin, neden yapıldığını anlamak çok zor olur.
- Ø Programlar için evrensel bir (Header) başlık kullanılmalıdır. Bu, iş yükünü hafifletir, tutarlı bir format yaratır ve hatırlanması gereken değişken sayısını azaltır.

- Ø Tüm alt rutinler tek bölgede toplanmalıdır. PIC'le çalışırken bu evrensel bölge, her bellek sayfasının üstünde (00-FFh) olmalıdır.
- Ø Yazılımın ne yapmasını gerektiğini hatırlamak için, bir akış diyagramı çizilmelidir.

### 1.6.1. Bank Değişirme

STATUS kayıtçısının 5. ve 6. bitleri (RP0,RP1) bank değiştirmek için kullanılır. Bank değiştirme işlemi o banktaki kayıtçıları kullanabilmek için yapılır. Şekil 1.5'te PIC16F84 için Bank 0 ve Bank 1'de bulunan kayıtçılar ve bellekteki adresleri görülmektedir. PIC16F84'ün sadece 2 bankı bulunduğundan bank seçimi sadece STATUS kayıtçısının 5. biti ( RP0) ile yapılır.Yani 6. bitin değeri daima 0 olmalıdır. Zaten PIC enerjilendiği anda power-On-Reset (POR) gerçekleşir ve bu iki bit 0 olur. Bu bitler aynı zamanda MCLR ucundan yapılan hâricî reset ve WDT reseti durumunda da 0 olur. Bank geçişleri için aşağıdaki komut ikilisi kullanılır (PIC16F84 için);

BSF STATUS,5 ; Bank 1 seçilir.  
BCF STATUS,5 ; Bank 0 seçilir.

| File Address |                                                 | File Address                      |     |
|--------------|-------------------------------------------------|-----------------------------------|-----|
| 00h          | Indirect addr. <sup>(1)</sup>                   | Indirect addr. <sup>(1)</sup>     | 80h |
| 01h          | TMR0                                            | OPTION                            | 81h |
| 02h          | PCL                                             | PCL                               | 82h |
| 03h          | STATUS                                          | STATUS                            | 83h |
| 04h          | FSR                                             | FSR                               | 84h |
| 05h          | PORTA                                           | TRISA                             | 85h |
| 06h          | PORTB                                           | TRISB                             | 86h |
| 07h          |                                                 |                                   | 87h |
| 08h          | EEDATA                                          | EECON1                            | 88h |
| 09h          | EEADR                                           | EECON2 <sup>(1)</sup>             | 89h |
| 0Ah          | PCLATH                                          | PCLATH                            | 8Ah |
| 0Bh          | INTCON                                          | INTCON                            | 8Bh |
| 0Ch          |                                                 |                                   | 8Ch |
|              | 68<br>General<br>Purpose<br>registers<br>(SRAM) | Mapped<br>(accesses)<br>in Bank 0 |     |
| 4Fh          |                                                 |                                   | CFh |
| 50h          |                                                 |                                   | D0h |
|              |                                                 |                                   |     |
|              |                                                 |                                   |     |
| 7Fh          |                                                 |                                   | FFh |
|              | Bank 0                                          | Bank 1                            |     |

Şekil 1.5: PIC 16F84 denetleyicisinin bank ve kayıtçı yapısı

## 1.6.2. Portların Giriş ve Çıkış Olarak Yönlendirilmesi

Portlara bağlı bulunan bir giriş/çıkış elemanını kullanabilmek için portların giriş/çıkış olarak yönlendirilmesi gerekir. PIC16F84'de A portunu TRISA kayıtçısı, B portunu TRISB kayıtçısı yönlendirir.

PortA/PortB'nin hangi bitleri giriş yapılacaksa TRISA/TRISB kayıtçılarının o bite karşılık gelen bitleri '1' yapılır. Aynı şekilde çıkış yapılacaksa TRISA/TRISB kayıtçılarının o bite karşılık gelen bitleri '0' yapılır.

**Örnek 1:** Port A'nın ilk 4 bitini giriş 5. bitini çıkış olarak kuralım (PIC16F84'de A portu 5 bitlik bir porttur.)

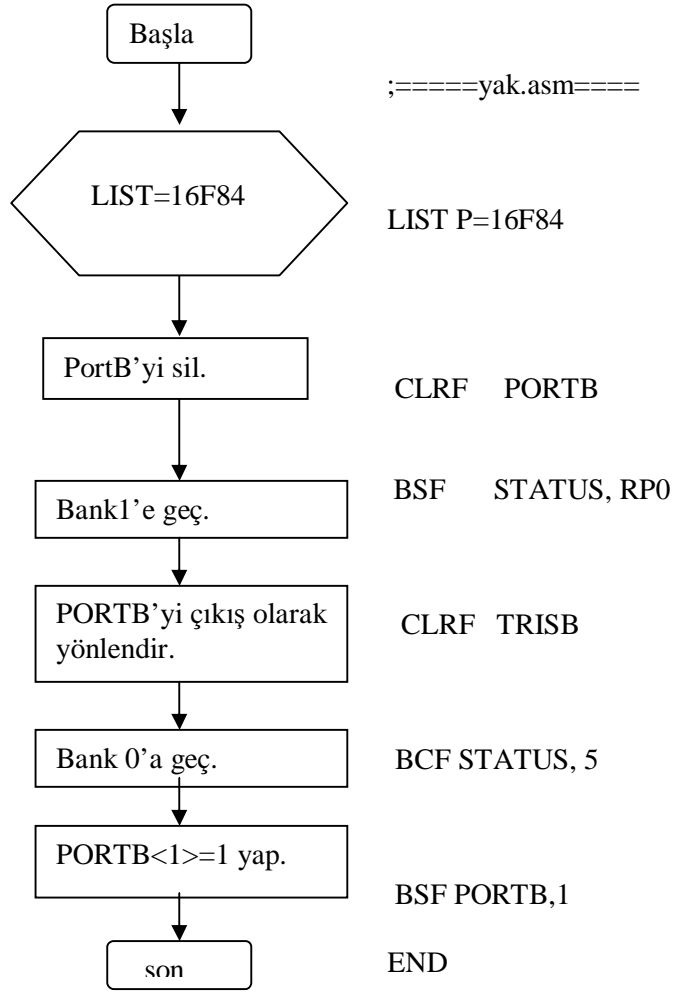
```
CLRF PORTA ; PORTA'yı sıfırla.
BSF STATUS, 5 ; Bank 1'e geç. Çünkü TRISA bank1'de.
MOVLW h'0F' ; TRISA'ya yüklemek için değer hazırla.
MOVWF TRISA ; TRISA'ya değeri yükle. TRISA<0:3> giriş, TRISA<4> çıkış için
 ; ayarlandı.
BCF STATUS, 5 ; Bank 0'e geç. Çünkü PORTA bank1'de ve I/O işlemleri; portlardan
 gerçekleşir.
```

**Örnek 2:** PortA'nın tamamı giriş, Port B'nin tamamını çıkış olarak kuralım (PIC16F84'de B portu 8 bitlik bir porttur.)

```
CLRF PORTB ; PORTB'yı sıfırla.
BSF STATUS, RP0 ; Bank1'e geç. Çünkü TRISA ve TRISB bank1'de.
CLRF TRISB ; TRISB'ya değeri yükle. TRISB<0:7> '0' olacağından tüm bitler
 ; çıkış olarak kuruldu.
MOVLW h'FF' ; A portunun tamamı
MOVWF TRISA ; giriş olarak ayarlandı.
BCF STATUS, 5 ; Bank 0'e geç. Çünkü PORTA ve PORTB bank 1'de ve I/O
 işlemleri; portlardan gerçekleşir.
```

## 1.6.3. Her Adım İçin Akış Diyagramı Çizme

Akış diyagram oluşturma ve sembolleri konusunda akış diyagramları hakkında bilgi verilmişti. Burada Port yönlendirmesine bir örnek olarak, PIC'e enerji verildiğinde PORTB'nin 1. bitine bağlı ledi yakacak programın akış diyagramını ve her sembolde işlenecek assembly komutları yazalım;



Akış diyagramı çizmenin amacı karmaşık ve/veya çok uzun programlar yazarken işlem sırası oluşturup düşünme kolaylığı sağlamasıdır.

#### 1.6.4. Konfigürasyon Bitlerinin Yazılması

Konfigürasyon bitleri, PIC'e gerilim uygulandığında PIC'in uyması gereken koşulları belirlemede kullanılır. Bu bitler aşağıdaki koşulları belirlemede kullanılır:

- Ø Osilatör tipini ,
- Ø WDT'in aktif/pasif yapma,
- Ø POR'i (Power-On-Reset) aktif/pasif yapma,
- Ø Kod korumayı aktif/pasif yapma.

Bu konfigürasyon bitlerini program içinde yazabileceğimiz gibi PIC programlayıcının kendi programındaki “fuses” penceresinden de yapabiliriz.

Ø **Şimdi konfigürasyon bitlerini belirleyen tanımları görelim;**

\_CP\_ON / OFF ; kod koruma var/yok.  
\_WDT\_ON / OFF ; WDT aktif /pasif.  
\_PWRT\_ON / OFF ; Power-On-Reset var/yok.  
\_RC\_OSC ; RC osilatör kullanılıyor.  
\_LP\_OSC ; Alçak güç (low power) osilatör kullanılıyor.  
\_XT\_OSC ; Kristal osilatör kullanılıyor.  
\_HS\_OSC ; Yüksek hızlı (High speed) osilatör kullanılıyor.

Ø Konfigürasyon cümlesini yazarken şunlara dikkat etmeliyiz:

- CONFIG ifadesinin yanındaki alt çizgi çift olmalıdır ‘\_\_’.
- Daha sonra yazılan ifadeler tek alt çizgili olmalıdır ‘\_’.
- Konfigürasyonlar arasında bir boşluk olmalıdır.
- Boşluktan sonra & işareti kullanılmalıdır.
- & işaretinden sonra ara vermeden konfigürasyon yazılır.

Ø **Aşağıdaki örneği inceleyiniz;**

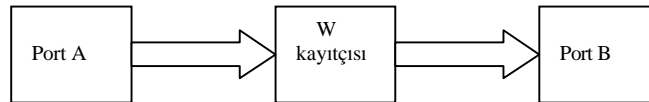
`_CONFIG _CP_OFF & _WDT_ON & _PWRT_ON & _RC_OSC`

Bu ifadeye ;

\_CP\_OFF ; kod koruma yok.  
\_WDT\_ON ; WDT aktif  
\_PWRT\_ON ; Power-On-Reset var.  
\_RC\_OSC ; RC osilatör kullanılıyor.

### 1.6.5. W Kayıtçısının Kullanımı

PIC16F84'ün RAM bellek alanında görülmeyen bir de W registeri vardır. W register bir akümülatördür. W registerine direkt olarak ulaşabiliriz. Ancak diğer registerlerin içerisindeki verileri aktarırken erişmek mümkündür. Bir PIC'te gerçekleşen tüm aritmetik işlemler ve atama işlemleri için W register kullanılmak zorunluluğu vardır. Örneğin iki register içindeki veriler toplanmak istendiğinde, ilk olarak registerlerden birinin içeriği W registre aktarılır (Şekil 1.6). Daha sonra da diğer registerin içerisindeki veri W registeri içerisindekiyle toplanır. Bu registerin kullanım özellikleri yine programlama konusunda detaylı olarak ele alınacaktır.



Şekil 1.6: W kayıtçısı kullanımı

### 1.6.6. Bitleri Test Ederek İşlem Yapma

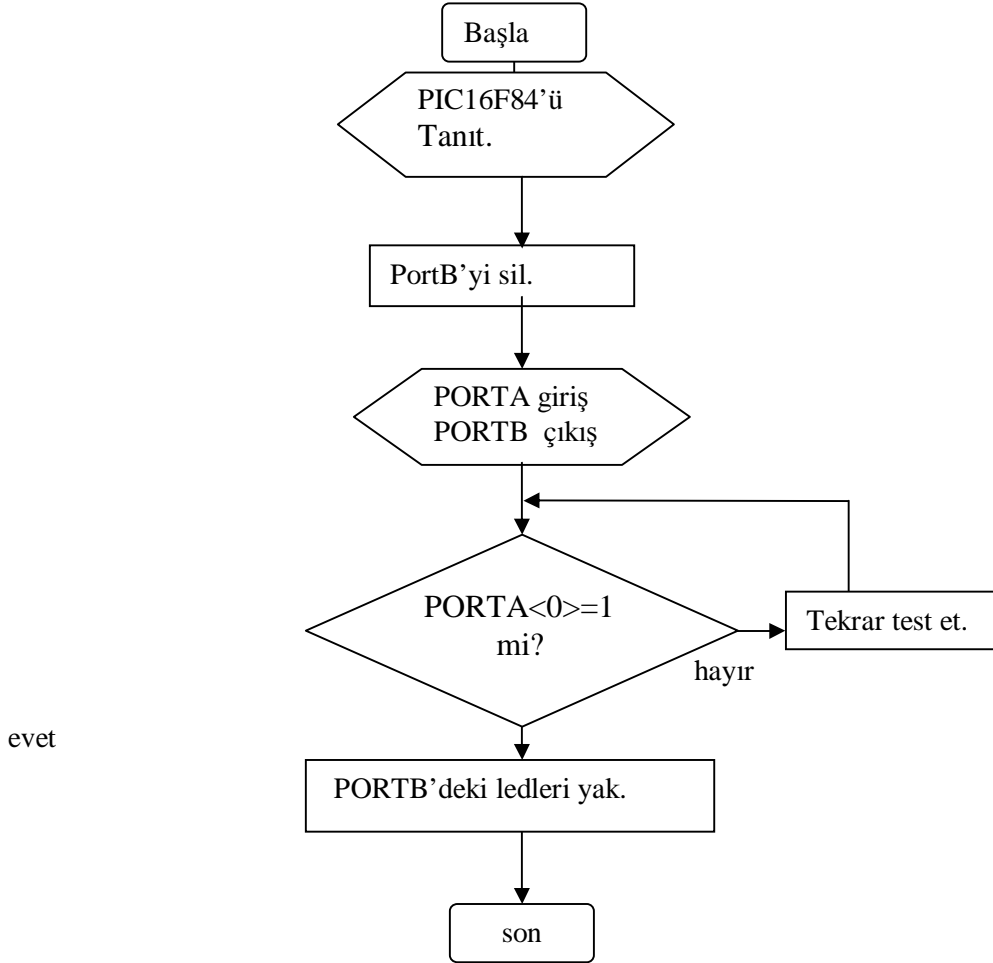
Bir kayıtçı içerisindeki herhangi bir bit BTFSS veya BTFSC komutları ile test edilebilir. “Mikrodenetleyici komutları” konusunda işlenen bu komutların işlevini kısaca hatırlarsak ; BTFSS komutu belirtilen f kayıtçısındaki biti test eder, eğer 1 ise bir sonraki komuta geçer.

BTFSC komutu ise belirtilen f kayıtçısındaki biti test eder , eğer 0 ise bir sonraki komuta geçer.

Şimdi bir örnekle bu komutun kullanımını görelim;

**Örnek :** A portunun 0.bitine bağlı olan bir butona basıldığında B portuna bağlı 8 ledi yakacak programı yazalım.

**Çözüm:** Önce programın akış diyagramını çıkaralım;



Şekil 1.7: Örnek programın algoritması

Şimdi de programımızı yazalım;

```
=====led.asm=====
```

```
LIST P=16F84
INCLUDE "P16F84.INC"
```

} tanımlamaların yapıldığı bölüm

```
CLRF PORTB
BSF STATUS,5
TRISB
MOVLW h'FF'
MOVWF TRISA
BCF STATUS,5
```

} A portu giriş, B portu çıkış olarak CLRF ayarlandı.

```
TEST_PORTA
```

```
BTFSS PORTA,0
```

; PORTA<0>=1 mi?

```
GOTO TEST_PORTA
```

; hayır.Tekrar test et.

```
MOVLW h'FF'
```

; evet. W=h'ff' yükle.

```
MOVWF PORTB
```

; B portundaki tüm ledleri yak.

```
SON
```

```
GOTO SON
```

} PIC16F84'de duraklama komutu olmadığı için burada yazılım ile sonsuz döngü oluşturup PIC'i aldatıyoruz. RESET olana kadar PIC sonsuz döngüde kalır.

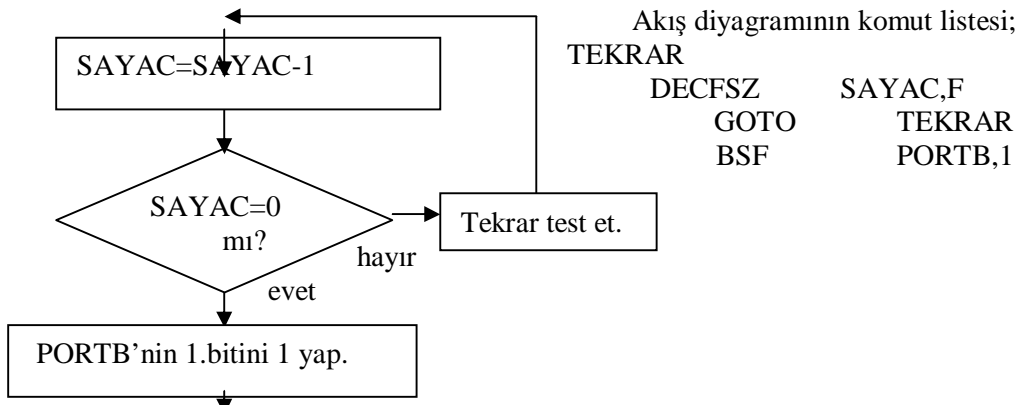
```
END
```

### 1.6.7. Sayaç Kullanarak Döngü Düzenlemek

Bazı işlemlerin önceden belirlenen sayıda tekrarlanması gerekebilir. Bu durumda programcı tarafından belirlenen bir kayıtçı sayaç olarak kullanılabilir. Sayaç şu şekilde hazırlanır:

- Ø Önce sayaç içerisine tekrar sayısını belirleyen sayı yüklenir.
- Ø Her işlem tekrarında sayaç 1 azaltılır. (DECFSZ komutu kullanılır.)
- Ø Sayaç 0 olduğunda program ya bitirilir ya da başka bir yere dallandırılır.

Şimdi de bu olayların akış diyagramını çizelim.



Şekil 1.8: DECFSZ komutu ile yapılan sayacın algoritması



Bilindiği gibi DECFSZ komutu , “kayıtçıdan 1 çıkart eğer sonuç 0 ise bir sonraki komuta dallan” işlevini yapmaktadır. Detaylı bilgi için “mikrodenetleyici komutları ” konusunda DECFSZ komutunu inceleyiniz.

“Zaman geciktirme döngüleri” konusunda sayaç kullanarak yapılan döngüleri inceleyiniz.

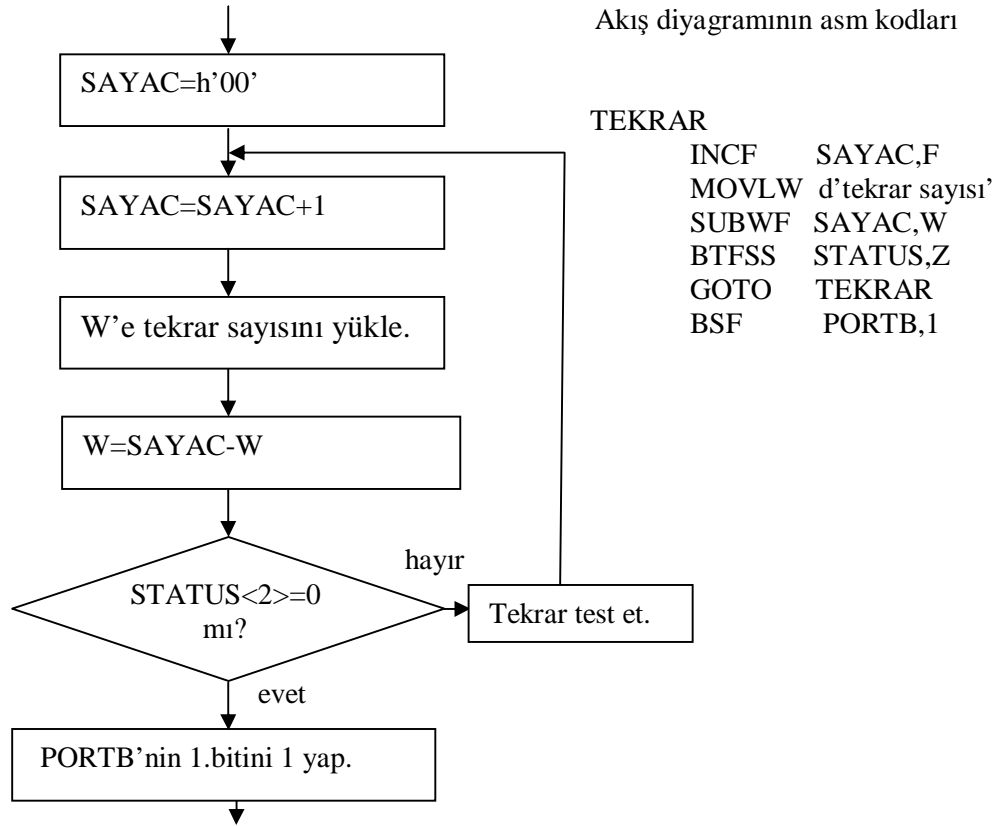
### 1.6.8. Karşılaştırma Yaparak Döngü Düzenlemek

SUBLW, SUBWF, INCF, DECF komutları kullanılarak da sayacı düzenlenebilir. Bu komutlar hakkında detaylı bilgi için “mikrodenetleyici komutları ” konusuna bakınız.

Sayaç şu şekilde hazırlanır:

- Ø Önce sayaç içerişi sıfırlanır.
- Ø Her işlem tekrarında sayaç 1 arttırılır. (INCF komutu kullanılır.)
- Ø Sayaç istenilen sayıya ulaştığında STATUS kayıtçısının Z biti BTFSS komutu ile sorgulanarak program ya bitirilir yada başka bir yere dallandırılır.

Şimdi de bu olayların akış diyagramını çizelim.



Şekil 1.9: SUBWF komutu ile yapılan sayacın algoritması

Yukarıdaki program parçasında SAYAC değeri W'ye yüklenen 'tekrar sayısına' ulaştığında SUBWF SAYAC,W komut satırı icrasında STATUS kayıtçısının Z bayrağı 1 olacaktır, ve B portunun 1.biti 1 yapılarak buna bağlı led yanacaktır. Burada dikkat edilmesi gereken konu, GOTO TEKRAR komut satırı, W'ye yüklenen d'tekrar sayısı'ndan 1 eksik sayıda icra edilecektir.Yani MOVLW d'10' yapıldıysa, GOTO TEKRAR komut satırı 9 defa icra edilecektir.

### 1.6.9. Status Kayıtçısı

STATUS kayıtçısı ALU'nun aritmetik statüsünü, RESET statüsünü ve veri belleğinin küme seçim bitlerini içermektedir.

Herhangi bir kayıtle olduğu gibi, STATUS kaydı herhangi bir komut için hedef olabilir. Eğer STATUS kaydı, Z, DC, veya C bitini etkileyen komutun hedefiyse, o zaman bu üç bit üzerine yazma etkin değildir.Yani sadece okunabilir durumdadır. Bu bitler aygıtın lojik durumuna göre uygun olarak kurulmuş veya silinmiştir. Daha ötesinde, TO ve PD bitleri sadece okunabilir bitlerdir.

Örneğin, CLRF STATUS komutu yukarı üç biti temizleyerek Z bitini kuracaktır. Bu STATUS kaydından 000u uluu olarak çıkacaktır. (u=değişmeyen). STATUS kaydını değiştirmek için yalnızca BCF, BSF, SWAPF ve MOVWF kullanılmaktadır. Çünkü bu komutlar hiçbir statüs bitini etkilememektedir.

STATUS kaydı, Z, DC ve C bitini etkileyen komutun hedefi olduğu durumlarda, bu üç bitin üzerine yazma etkinsizdir.

Çıkartmalarda, C ve DC bitleri ödünç alan bitler olarak çalışmaktadır.

Aşağıda STATUS kayıtçısının her bir bitinin hangi durumlarda 1 ve 0 olacağı gösterilmiştir;

|       |       |       |                 |                 |       |       |       |
|-------|-------|-------|-----------------|-----------------|-------|-------|-------|
| R/W-0 | R/W-0 | R/W-0 | R-1             | R-1             | R/W-x | R/W-x | R/W-x |
| IRP   | RP1   | RP0   | $\overline{TO}$ | $\overline{PD}$ | Z     | DC    | C     |
| bit7  |       |       |                 |                 | bit0  |       |       |

Şekil 1.10: Status kaydı (03h, 83h adresi)

R= Okunabilir Bit

W= Yazılabilir Bit

N= POR resetindeki değer.

-n=Bilinmez 1 ya da 0 olabilir.

Örneğin 2.bit olan Z biti R/W-n ifadesiyle, hem okunabilir hem de yazılabilir bir bit olduğunu, ayrıca güç reseti durumunda lojik seviyesi için bir şey söylenemeyeceği anlamını taşır. Ama 3.bit in güç reseti durumundaki lojik değeri 1'dir.

**Bit 7:** IRP: Bank kümesi seçim biti (dolaylı adresleme için kullanılır) .

0: Bank 0,1 (h'00'-h'ff')

1:Bank 2,3 (h'100'-h'1ff')

IRP biti PIC16F87X serisinde kullanılmaktadır. 16CXX, 16F8X serisi mikrodenetleyicilerde kullanılmaz.

**Bit 6-5:** RP1:RPO: Bank seçim bitleri (doğrudan adresleme için kullanılır)

00:Bank 0

01:Bank 1

10:Bank 2

11:Bank 3

PIC16F84'te sadece bank 0 ve bank1 bulunduğu için bank seçimini sadece RPO bitini kullanarak yaparız. Her bir bank 128 bayttır.

**Bit 4:**TO: Zaman aralığı biti (Time-out)

1= PIC'e enerji verildiğinde, CLRWDT komutuyla veya SLEEP'den güç verme durumuna geçirildiğinde 1 olur.

0= WDT süre aşımı işlemi gerçekleşmiş ise 0 olur.

**Bit 3:**PD: Güç kesme biti (Power down).

1= PIC'e enerji verildiğinde veya CLRWDT komutu ile olur.

0= SLEEP komutunun yürütülmesi ile olur.

**Bit 2 :**Z: Sıfır Bit (zero).

1= Aritmetik veya mantıksal işlemin sonucu sıfırdır.

0= Aritmetik veya mantıksal işlemin sonucu sıfır değildir.

**Bit 1:**DC: Dijit elde (Digit Carry/Borrow) biti. (ADDWF ve ADDLW komutları için)

1= Alt 4 bitin (1.dijit) 4. bitinde taşma meydana geldiğinde olur.

0= Alt 4 bitin (1.dijit) 4. bitinde taşma meydana gelmediğinde olur.

**Bit 0 :**C: Elde (Digit Carry/Borrow) biti. (ADDWF ve ADDLW komutları için)

1= Meydana gelen sonucun en önemli bitinden (7.bit) taşma olursa 1 olur.

0= Meydana gelen sonucun en önemli bitinden (7.bit) taşma olmazsa 0 olur.

RRF, RLF komutların yürütülmesi sırasında , bu bit, kaynak kaydındaki düşük veya yüksek değerlikli ( MSB veya LSB) biti ile yüklenmektedir.

## 1.6.10. Zaman Geciktirme Döngüleri

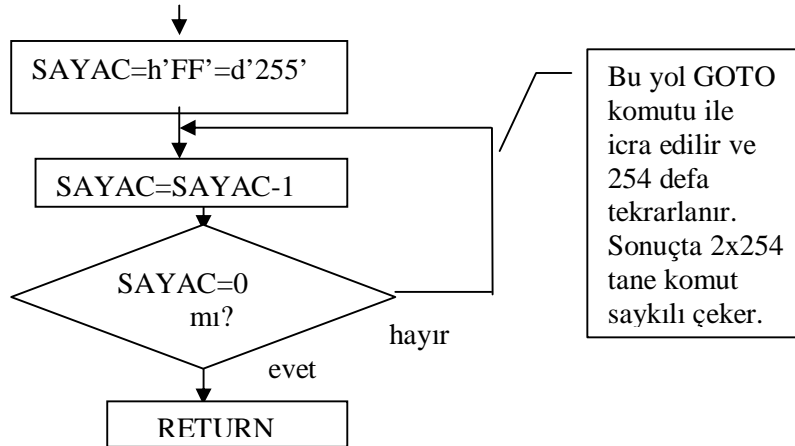
Bu konumuzda döngü gecikmelerinde geçen zamanın hesaplanmasını ve diğer komutları öğreneceğiz. Bunun için öncelikle dâhilî komut saykılı kavramını bilmemiz gerekir.

PIC16F84'e uygulanan 4 Mhz'lik osilatör sinyalinin 1 komutu icrası için 4 palsı gerekir. Dolayısıyla PIC'in 1 komutu icra etmek için kullandığı frekans, dışarıdan uygulanan sinyalin ¼ üne düşer ve 1 Mhz olur. İşte Fosc/4 değerindeki bu değere dâhilî komut saykılı denir. 1Mhz'lik sinyalin dâhilî komut periyodu da 1µS olacaktır. Bazı komutların icrası ise 2 komut saykılık zaman alır. Bu komutlar tablo 1.1'de verilmiştir

| Komut                     | Komut saykılı                                                            |
|---------------------------|--------------------------------------------------------------------------|
| GOTO                      | 2                                                                        |
| RETURN                    | 2                                                                        |
| CALL                      | 2                                                                        |
| PC'ye veri yazan komutlar | 2                                                                        |
| DECFSZ                    | 1 (Kayıtçı içindeki değer 0 değilse)<br>2 (Kayıtçı içindeki değer 0 ise) |
| RETLW                     | 2                                                                        |
| RETFIE                    | 2                                                                        |
| INCFSZ                    | 1 (Kayıtçı içindeki değer 0 değilse)<br>2 (Kayıtçı içindeki değer 0 ise) |
| BTFSC                     | 1 (Kayıtçı içindeki değer 0 değilse)<br>2 (Kayıtçı içindeki değer 0 ise) |
| BTFSS                     | 1 (Kayıtçı içindeki değer 1 değilse)<br>2 (Kayıtçı içindeki değer 1 ise) |

Tablo 1.1: 2 komut saykılı çeken komutlar

Şimdi de tek döngü ile gecikme altprogramının akış diyagramını çizip programını yazalım ve maksimum ne kadarlık bir zaman gecikmesi yaptığını hesaplayalım.



Şekil 1.11: Tek döngülük zaman gecikmesi algoritması

|                            |       |        |         |                 |
|----------------------------|-------|--------|---------|-----------------|
| GECIKME                    | MOVLW | h'FF'  |         | ;1 saykıl       |
|                            |       | MOVWF  | SAYAC   | ;1 saykıl       |
| SAY                        |       | DECFSZ | SAYAC,F | ;1x254+2 saykıl |
|                            |       | GOTO   | SAY     | ;2x254 saykıl   |
|                            |       | RETURN |         | ;2 saykıl       |
| <hr/>                      |       |        |         |                 |
| TOPLAM: 766 komut saykılı. |       |        |         |                 |

Yukarıdaki hesaplamada dikkat edilmesi gereken nokta DECFSZ komutunun icrasdır. SAYAC değeri '0' olana kadar bu komut bir komut saykılı çekecektir. SAYAC değeri '0' olunca ise 2 komut saykılı çeker. SAYAC sıfır olana kadar 254 defa komut icra edileceğinden toplamda bu komut (1x254+2=256) komut saykılı çekecektir.

Aynı şekilde GOTO komutu icrası 2 saykıl alır (Tablo 1.1). Bu komut proram sonuna kadar 254 defa icra edileceğinden toplam 2x254=508 komut saykılı çeker.

GECIKME alt programı sonunda ise toplan 766 komut saykılı süresi kadar bir gecikme olur . Şimdi de bu kadar saykılın 4 Mhz kristal osilatör kullanan bir PICC16F84'de ne kadar sürelik bir gecikme yaptığını bulalım.

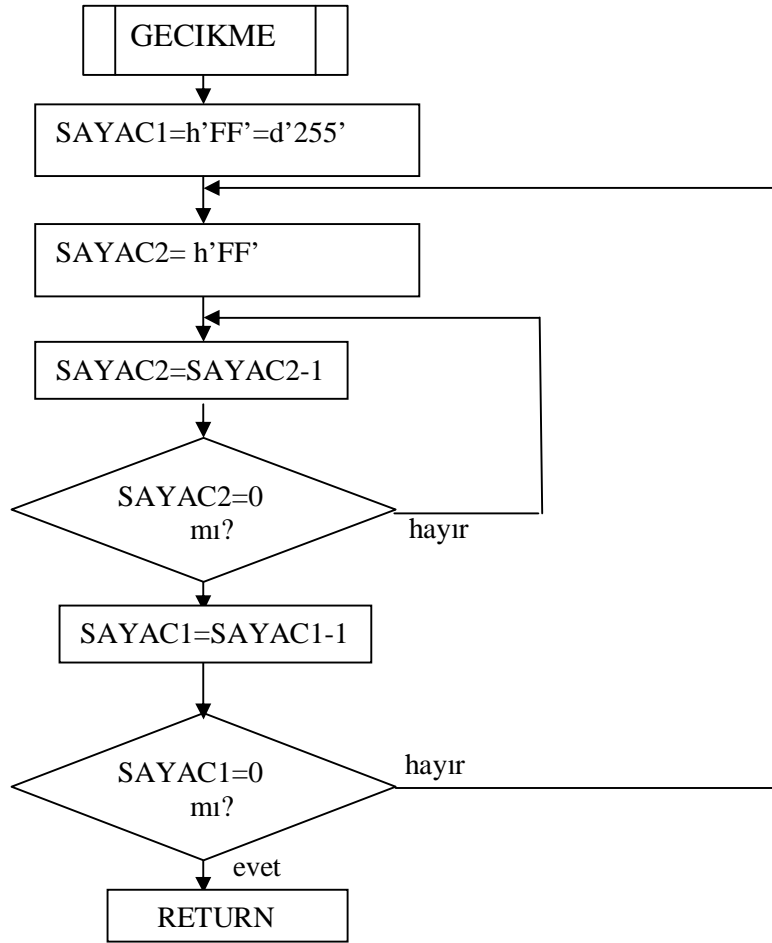
Osilatör frekansı 4Mhz ise , dâhilî komut frekansı  $F=F_{osc}/4$  olduğundan 1 Mhz olacaktır. O hâlde bir komut saykıl süresi ;

$$t=1/f=1/1Mhz= 1\mu S \text{ olacaktır.}$$

GECIKME altprogramının süresi ise  $1\mu S \times 766 = 766\mu S = 0.766mS$  olacaktır. Bu süre tek döngü ile yapılabilecek maksimum zaman gecikmesidir.

Uygulamalarda tek döngü ile yeteri kadar gecikme yapılamadığı için, en az iki döngü yapmak gerekmektedir.

Aşağıda iki döngü ile yapılmış gecikme altprogramının akış diyagramı görülmektedir.



Şekil 1.12: Çift döngülük zaman gecikmesi algoritması

Şimdi de GECİKME altprogramının assembly programını yazıp geçen zamanı bulalım.

| Etiket  | Komut  | Sayı     |
|---------|--------|----------|
| GECİKME | MOVLW  | D'255'   |
|         | MOVWF  | SAYAC1   |
| DONGU1  |        |          |
|         | MOVLW  | D'255'   |
|         | MOVWF  | SAYAC2   |
| DONGU2  |        |          |
|         | DECFSZ | SAYAC2,F |
|         | GOTO   | DONGU2   |
|         | DECFSZ | SAYAC1,F |
|         | GOTO   | DONGU1   |
|         | RETURN |          |

Şekil 1.13: Çift döngülük zaman gecikmesi assembly programı

Burada SAYAC1 değerine M, SAYAC2 değerine N dersek toplam saykıl sayısını yaklaşık  $TSS=3MN+5M+4$  kadar buluruz. Buradaki 4 değeri rahatlıkla ihmal edilebilir. Hatta çok hassas bir zamandan bahsedilmiyorsa 5M değeri de ihmal edilebilir. Eğer  $M=N$  alınırsa  $TSS=3M^2$  olarak sadeleştirilebiliriz. Maksimum zaman gecikmesi yapmak için sayaçlara d'225' yüklenmiştir. Yani  $M=N=255$ 'dir. Bu durumda toplam komut saykıl sayısı yaklaşık olarak;

$$TSS=3 \times 255 \times 255 + 5 \times 255 + 4 = 196.354 \text{ çıkar.}$$

Eğer sadeleşmiş formülü kullanırsak;

$$TSS=3M^2=3 \times 255^2=195.075 \text{ çıkar.}$$

Bu da 4 Mhz frekans için  $196.35 \text{ mS} = 0.196$  saniye yaklaşık 0.2 saniye eder. Bu değer 2 döngü ile yapabileceğimiz maksimum gecikme süresidir. Şimdide istenen bir gecikme değeri için sayaçlara yüklenecek değerin nasıl bulunacağını görelim. Örneğin 2 döngülü bir gecikme ile 40mS'lik bir zaman gecikmesi yapmak için kayıtlara hangi değerleri yükleyeceğimizi bulalım. Kolaylık bakımından sayaç değerlerinin aynı sayı ile yükleyelim. (PIC 16F84'in osc frekansı 4Mhz kabul edilecek.) Çözüm:  $40\text{mS}=40000\mu\text{S}$  yapar. Bu da dâhilî komut saykıl süresi  $1\mu\text{S}$  olan bir PIC için  $TSS=40000$  yapar.  $TSS=3M^2$ ,  $40000=3M^2$  ve  $M=\sqrt{(4000/3)=116}$  desimal değeri bulunur.

Eğer PIC'imizin osc frekansı 10Mhz olsaydı , ne değışirdi?

Çok şey. Bu durumda; dâhilî komut saykılı  $f=Fosc/4=2,5 \text{ Mhz}$  ve  $t=1/f=0.4 \mu\text{S}$  olurdu.  $40\text{mS}=40000\mu\text{S}$  olduğundan bu sürenin geçmesi için gereken saykıl sayısı  $TSS=40000/0.4=100000$  saykıl gerekir.  $TSS=3M^2$ ,  $100000=3M^2$  ve  $M=\sqrt{(100000/3)=183}$  desimal değeri bulunur.

Bu bilgiler ışığında M'in bulunması için genel bir formül yazarsak;

$$M=\sqrt{\frac{GS.Fosc}{12}}$$

bulunur. Burada; GS=Gecikme süresi( $\mu\text{S}$ ), süresi

$Fosc$ =PIC'in osilatör frekansı (Mhz)

M=Sayaç değerini (desimal) ifade eder.

Yukarıdaki örnekleri bir bu formülle çözerek sonuçların aynı olduğunu görünüz. Şimdi de 3 döngülü bir gecikme altprogramı yazalım.  $Fosc=4 \text{ Mhz}$  için toplam 50.070.529 mikrosaniye ve yaklaşık 50 saniye eder.

|         |        |          |               |            |
|---------|--------|----------|---------------|------------|
| GECIKME |        |          |               |            |
|         | MOVLW  | D*255*   | 1             | 1          |
|         | MOVWF  | SAYAC1   | 1             | 1          |
| DONGU1  |        |          |               |            |
|         | MOVLW  | D*255*   | 255           | 255        |
|         | MOVWF  | SAYAC2   | 255           | 255        |
| DONGU2  |        |          |               |            |
|         | MOVLW  | D*255*   | 255x255       | 65.025     |
|         | MOVWF  | SAYAC3   | 255x255       | 65.025     |
| DONGU3  |        |          |               |            |
|         | DECFSZ | SAYAC3,F | 255x255x255   | 16.581.375 |
|         | GOTO   | DONGU3   | 2x255x255x255 | 33.162.750 |
|         | DECFSZ | SAYAC2,F | 255x255       | 65.025     |
|         | GOTO   | DONGU2   | 2x255x255     | 130.050    |
|         | DECFSZ | SAYAC1,F | 255           | 255        |
|         | GOTO   | DONGU1   | 2x255         | 510        |
|         | RETURN |          | 2             | 2          |
|         |        |          |               | 50.070.529 |

Şekil 1.14: Çift döngülük zaman gecikmesi assembly programı

Demek ki 3'lü döngü ile yeteri kadar zaman elde edebiliriz.

Eğer 4'lü bir döngü düzenlersek o zaman saatlerce sürecek bir zaman gecikmesi elde etmek mümkün olur. Şimdi de bir tablo çizerek dörtlü bir döngü hâlinde ne olabileceğini görelim.

|         |        |          |                   |                |
|---------|--------|----------|-------------------|----------------|
| GECIKME |        |          |                   |                |
|         | MOVLW  | D*255*   | 1                 | 1              |
|         | MOVWF  | SAYAC1   | 1                 | 1              |
| DONGU1  |        |          |                   |                |
|         | MOVLW  | D*255*   | 255               | 255            |
|         | MOVWF  | SAYAC2   | 255               | 255            |
| DONGU2  |        |          |                   |                |
|         | MOVLW  | D*255*   | 255x255           | 65.025         |
|         | MOVWF  | SAYAC3   | 255x255           | 65.025         |
| DONGU3  |        |          |                   |                |
|         | MOVLW  | D*255*   | 255x255x255       | 16.581.375     |
|         | MOVWF  | SAYAC4   | 255x255x255       | 16.581.375     |
| DONGU4  |        |          |                   |                |
|         | DECFSZ | SAYAC4,F | 255x255x255x255   | 4.228.250.600  |
|         | GOTO   | DONGU4   | 2x255x255x255x255 | 8.456.501.200  |
|         | DECFSZ | SAYAC3,F | 255x255x255       | 16.581.375     |
|         | GOTO   | DONGU3   | 2x255x255x255     | 33.162.750     |
|         | DECFSZ | SAYAC2,F | 255x255           | 65.025         |
|         | GOTO   | DONGU2   | 2x255x255         | 130.050        |
|         | DECFSZ | SAYAC1,F | 255               | 255            |
|         | GOTO   | DONGU1   | 2x255             | 510            |
|         | RETURN |          | 2                 | 2              |
|         |        |          |                   | 12.767.985.129 |

Şekil 1.15. Çift döngülük zaman gecikmesi assembly programı

Sonuç olarak 12.767 bir gecikme elde edilir. Bu da 212 dakika ve yaklaşık 3.5 saatlik bir gecikme olur. Bu yaklaşık 255 X50 saniye demektir, burada ilk 255 yerine decimal 100 girerseniz 100X50 saniye yani 5000 saniye elde edersiniz bu da 83 dakika eder. 1 girilirse 50 sn. 10 girilirse 8.3 dakika elde edilir. İlk 255 sayısı yerine 1'den 255'e kadar değişen rakamlar girildiğinde 4'lü bir döngü için 50 sn.den 3.5 saate kadar uzanan bir zaman



gecikmesi elde edilir. Tek ve iki döndüdeki gibi istenirse 3'lü ve 4'lü döngülerde formülize edilebilir. 5'li bir döngü hâlinde ise bu zaman bir ay (900 saat) civarındadır.

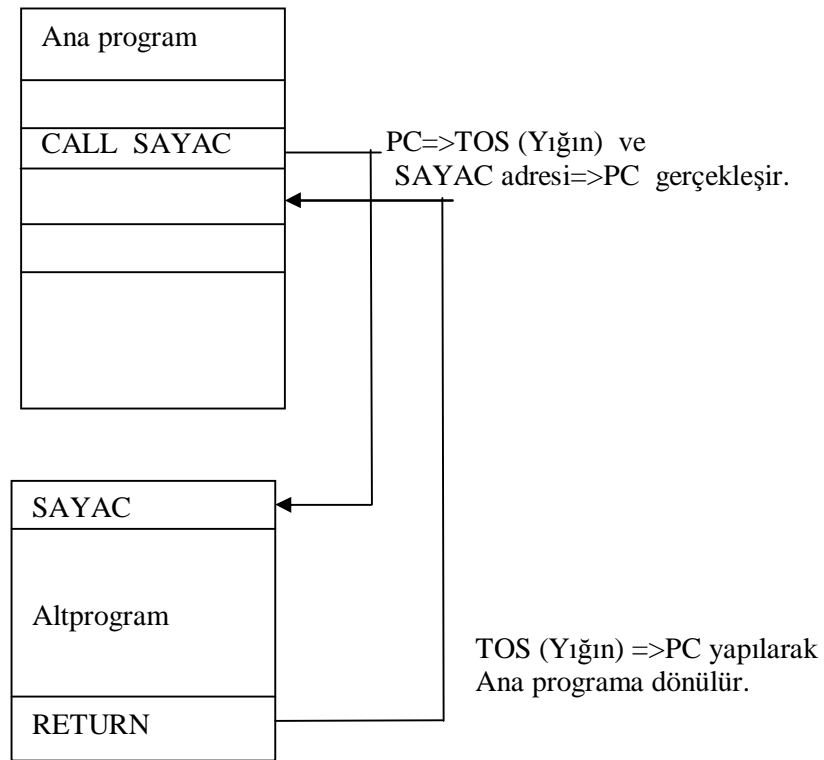
### 1.6.11. Altprogramlar

Program içerisinde defalarca tekrar edilmesi gereken komut dizilerini sürekli yazmak yerine, bu komut dizilerini altprogramlar şeklinde yazarak hem bellek alanından tasarruf edilmiş olur, hem de program sadelik kazanır. Altprogramlar ana programdan CALL komutu ile çağrılır ve altprogramın çalışması bittikten sonra ana programa RETURN komutu ile dönüş sağlanır.

Altprogramlar iki çeşittir:

- Ø CALL komutu ile çağrılan altprogramlar,
- Ø Kesme altprogramları. Kesme altprogramları , “Kesme alt programlarının düzenlenmesi” konusunda işlendiğinden burada işlenmemiştir.

Bir altprogramın çalışmasını şematik olarak şekil 1.16' daki gibi gösterebiliriz;



Şekil 1.16: Altprogram kurgusu

Yukarıdaki şekilde şunlar dikkatinizi çekmiş olmalıdır:

- Ø CALL komutundan sonra altprogramın isimini yazarak altprogramı çağırırız. Burada altprogramımızın ismi SAYAC'tır.
- Ø Altprogram çağrıldığında program sayıcı (PC) kayıtçısının içeriği (yani geri dönüş adresi) yığına (TOS) itilir ve SAYAC altprogram adresi PC'ye yazılır. Bu olay mikrodenetleyici tarafından otomatik olarak gerçekleşir. Böylece altprograma dallanma gerçekleşmiş olur.
- Ø Altprogramın çalıştırılması bittikten sonra RETURN komutu ile ana programa dönüş sağlanır. Çünkü RETURN komutu, daha önce yığına itilmiş geri dönüş adresini PC'ye tekrar yükler.
- Ø Ayrıca altprogram içinde başka altprogramlar da çağrılabilir.

Sıradaki "Bit pozisyonlarını sola kaydırma" konumuzda led.asm programında bir gecikme altprogramı kullanılmıştır. İnceleyiniz.

## 1.6.12. Bit Kaydırma

### 1.6.12.1. Bit Pozisyonlarını Sola Kaydırma

Bit pozisyonlarını sola kaydırmak için RLF komutunu kullanırız. Bu komut belirlenen bir kayıtçı içerisindeki bitleri her defasında 1 bit sola kaydırır. Detaylı bilgi için mikrodenetleyici komutları konusunda RLF komutunu inceleyiniz.

Şimdi bu komutun kullanımına bir örnek verelim.

**Örnek:** PORTB'ye bağlı 8 led sırayla periyodik olarak yansın. Şöyle ki; PORTB<0>'dan PORTB<7>'ya doğru tek bit sırayla yansın ve 7. bitten sonra işlem başa dönsün.

```

;====led.asm=====
LIST P=16F84
INCLUDE "P16F84.INC"
SAYAC1 EQU h'0C'
SAYAC2 EQU h'0D'
CLRF PORTB
BSF STATUS,5
CLRF TRISB
BCF STATUS,5

BASLA
BCF STATUS,0 ;Elde biti sıfırlandı.
MOVLW h'01' ;ilk değeri yükle ve
MOVWF PORTB ;B portundan çıkar.

TEKRAR
CALL GECIKME ;2.değer için bekle.
RLF PORTB,F ;sola kaydır
BTFSS STATUS,0 ;C=1 mi?
GOTO TEKRAR ;hayır.TEKRAR'a dallan.
GOTO BASLA ;evet.BASLA'ya dallan.

GECIKME
MOVLW h'FF'
MOVWF SAYAC1
DON1
MOVLW h'FF'
MOVWF SAYAC2
DON2
DECFSZ SAYAC2,F
GOTO DON2
DECFSZ SAYAC1,F
GOTO DON1
RETURN
END

```

tanımlamaların yapıldığı bölüm

B portu çıkış olarak ayarlandı.

Ana Program

GECİKME altprogramı.

### 1.6.12.2. Bit Pozisyonlarını Sağa Kaydırma

Bit pozisyonlarını sağa kaydırmak için RRF komutunu kullanırız. Bu komut belirlenen bir kayıtçı içerisindeki bitleri her defasında 1 bit sağa kaydırır. Detaylı bilgi için mikrodenetleyici komutları konusunda RRF komutunu inceleyiniz.

Şimdi bu komutun kullanımına bir örnek verelim.

**Örnek:** PORTB'ye bağlı 8 led kara şimşek düzeninde yansın. Yani PORTB<0>'dan PORTB<7>'ya doğru tek bit sola doğru sırayla yansın ve 7. bitten sonra işlem sağa doğru dönsün ve bu işlemler periyodik olarak tekrarlınsın.

```

;====karaşimşek.asm=====
LIST P=16F84
INCLUDE "P16F84.INC"
SAYAC1 EQU h'0C'
SAYAC2 EQU h'0D'
CLRF PORTB
BSF STATUS,5
CLRF TRISB
BCF STATUS,5
BCF STATUS,0
MOVLW h'01'
MOVWF PORTB

SOL
CALL GECIKME ;Yeni deęer için bekle.
RLF PORTB,F ;sola kaydır
BTFSS PORTB,7 ;PORB<7>=1 mi?
GOTO SOL ;hayır.SOL'a dallan.

SAG
CALL GECIKME ;evet. Yeni deęer için bekle.
RRF PORTB,F ; saęa kaydır.
BTFSS PORTB,0 ; PORB<0>=1 mi?
GOTO SAG ;hayır.SAG'a dallan.
GOTO SOL ;evet.SOL'a dallan,

GECIKME
MOVLW h'FF'
MOVWF SAYAC1
DON1
MOVLW h'FF'
MOVWF SAYAC2
DON2
DECFSZ SAYAC2,F
GOTO DON2
DECFSZ SAYAC1,F
GOTO DON1
RETURN
END

```

tanımlamaların yapıldığı bölüm

B portu çıkış olarak ayarlandı.

;Elde biti sıfırlandı.  
;ilk değeri yükle ve  
;B portundan çıkar.

GECİKME altprogramı.

### 1.6.12.3. Bit Pozisyonlarını Tersleme

COMF komutu bir kayıtçı içerisindeki bitleri tersine çevirir (complement) . Detaylı bilgi için mikrodenetleyici komutları konusunda COMF komutunu inceleyiniz. Bu komut, 2'li tümleme yöntemiyle çıkarma işlemi yaparken , bazı mantıksal devre çözümlerine ,ışık şovları tasarlarken işimize yarayabilir.

Şimdi bu komutun kullanımına bir örnek verelim.

**Örnek:** PORTB'ye bağlı 8 ledleri dönüşümlü olarak ilk önce ilk 4 bitindeki, sonrada son 4 bitindeki ledleri yakan programı yazalım.

```
;=====TERSLE.asm=====
```

```
LIST P=16F84
INCLUDE "P16F84.INC"
SAYAC1 EQU h'0C'
SAYAC2 EQU h'0D'
CLRF PORTB
BSF STATUS,5
CLRF TRISB
BCF STATUS,5
MOVLW h'0F'
MOVWF PORTB
;ilk değeri yükle ve
;B portundan çıkar.

TERSLE
CALL GECIKME ;Yeni değer için bekle.
COMF PORTB,F ;PORTB'yi tersle.
GOTO TERSLE ; sağa kaydır.

GECIKME
MOVLW h'FF'
MOVWF SAYAC1

DON1
MOVLW h'FF'
MOVWF SAYAC2

DON2
DECFSZ SAYAC2,F
GOTO DON2
DECFSZ SAYAC1,F
GOTO DON1
RETURN
END
```

tanımlamaların yapıldığı bölüm

B portu çıkış olarak ayarlandı.

GECIKME altprogramı.

### 1.6.13. Mantıksal İşlemler

Mantıksal işlem komutları bir kayıtçadaki istenilen bitleri değiştirmek (maskelemek veya kurmak), test etmek amacıyla kullanılırlar. Bu komutlar ANDLW, ANDWF, IORLW, IORWF, XORLW, XORWF ve COMF komutlarıdır. Detaylı bilgi için mikrodenetleyici komutları konusunda bu komutları inceleyiniz. Şimdi bu komutları görelim.

#### 1.6.13.1. İstenen Bitleri Sıfırlamak

ANDLW komutu W kayıtçı içerisinde bulunan istenilen bitleri sıfırlamak (maskelemek) için kullanılır. Burada sonuç daima W kayıtçısındadır.

Örneğin W kayıtçısındaki 3. ve 4. bitlerini diğer bitleri değiştirmeden sıfırlamak (meskelemek) istiyoruz Bunun için kullanacağımız komut satırı;

ANDLW h'E7' ; olmalıdır.

Burada sabiti h'E7' olarak kurduk. Çünkü;

W =b(xxxx xxxx)  
AND h'E7'=b'1110 0111'  
W=b'xxx0 0xxx' olur.

Sonuçta W kayıtçısındaki 3. ve 4. bitlerini diğer bitleri değiştirmeden sıfırlamış (meskelemek) olduk. Burada dikkat edilmesi gereken , maskelenecek bitler 0, diğer bitler 1 ile AND'lenecek şekilde sayı seçilmelidir (h'E7').

ANDWF komutu da maskelemek için kullanılabilir. Ancak bu komutta maskeleme için kullanılan değer sabit bir sayıdan değil, bir dosya kayıtçısından (f) alınır ve sonuç değer hedef (d) bitine göre W yada F kayıtçısındadır.

“Çevrim tablolarının kullanım yerleri ve kullanımı” konusunda verilen hex.asm programında da bu komutun kullanımını görebilirsiniz.

#### 1.6.13.2. İstenen Bitleri Bire Çevirmek

IORLW komutu W kayıtçı içerisinde bulunan istenilen bitleri 1 yapmak (kurmak) için kullanılır. Burada sonuç daima W kayıtçısındadır.

Örneğin W kayıtçısındaki 3. ve 4. bitlerini diğer bitleri değiştirmeden 1 yapmak istiyoruz Bunun için kullanacağımız komut satırı;

IORLW h'18' ; olmalıdır.

Burada sabiti h'18' olarak kurduk. Çünkü;

W =b(xxxx xxxx)  
OR h'18'=b'0001 1000'  
W=b'xxx1 1xxx' olur.

Sonuçta W kayıtçısındaki 3. ve 4. bitlerini diğer bitleri değiştirmeden sıfırlamış (meskelemek) olduk. Burada dikkat edilmesi gereken , kurulacak bitler 1, diğer bitler 0 ile OR'lanacak şekilde sayı seçilmelidir (h'18').

IORWF komutu da maskelemek için kullanılabilir. Ancak bu komutta maskeleme için kullanılan değer sabit bir sayıdan değil, bir dosya kayıtçısından (f) alınır ve sonuç değer hedef (d) bitine göre W ya da F kayıtçısındadır.

### 1.6.13.3. İstenen Bitleri Terslemek

XORLW komutu W kayıtçı içerisinde bulunan istenilen bitleri terslemek için kullanılır. Burada sonuç daima W kayıtçısındadır.

Örneğin W kayıtçısındaki değer h'4C' olsun. W'nin 3. ve 4. bitlerini diğer bitleri değiştirmeden terslemek istiyoruz. Bunun için kullanacağımız komut satırı;

```
XORLW h'18' ; olmalıdır.
```

Burada sabiti h'18' olarak kurduk. Çünkü;

```
 W = b'0100 1100'
XOR h'18'=b'0001 1000'
 W= b'0101 0100' olur.
```

Sonuçta W kayıtçısındaki 3. ve 4. bitlerini diğer bitleri değiştirmeden sıfırlamış (maskelemek) olduk. Burada dikkat edilmesi gereken, terslenecek bitler 1, diğer bitler 0 ile XOR'lanacak şekilde sayı seçilmelidir (h'18').

XORWF komutu da maskelemek için kullanılabilir. Ancak bu komutta maskeleme için kullanılan değer sabit bir sayıdan değil, bir dosya kayıtçısından (f) alınır ve sonuç değer hedef (d) bitine göre W ya da F kayıtçısındadır.

### 1.6.13.4. Bir Baytlık İki Veriyi Karşılaştırmak

W kayıtçısının, bir baytlık sabit bir veriyle aynı olup olmadığını test etmek için XORLW komutu kullanılabilir. Benzer şekilde W kayıtçısının, bir dosya kayıtçısı içeriği ile aynı olup olmadığını test etmek için de XORWF komutu kullanılabilir. Eğer veriler eşit ise STATUS kayıtçısındaki sıfır bayrağı (Z) 1 olur. İşte bu biti test ederek verilerin aynı olup olmadığını kontrol edebiliriz.

Örneğin B portundaki değer h'4C' olup olmadığını test etmek için gereken program parçasını yazalım. Bunun için kullanacağımız komut satırı;

```
 MOVLW h'4C' ;W= h'4C'
TEST_PORTB
 XORWF PORTB,F ;PORTB=(PORTB XOR W)
 BTFSS STATUS,2 ;STATUS<2>=1 mi? (Z=1 mi?)
 GOTO TEST_PORTB; hayır.Tekrar test et.
DEVAM ;evet. Sonraki komutlar...
 MOV


```

XORLW ve XORWF komutlarıyla sadece verilerin eşit olup olmadığını testini yapabiliriz.

Bu işlem için istenirse SUBLW ve SUBWF komutları da kullanılabilir. Ayrıca bu iki komutla, büyük / küçük mü? Testleride yapılabilir.

### 1.6.13.5. Bir Baytlık Veriyi Sıfır ile Karşılaştırmak

W kayıtçısının içeriğindeki 1 baytlık verinin '0' olup olmadığını anlamak için , h'00' sabit verisiyle OR'lanır. Bu işlem için IORLW komutu kullanılır.

Bir kayıtçının (f) içeriğindeki 1 baytlık verinin '0' olup olmadığını anlamak için , önce W kayıtçısına h'00' sabit verisi yüklenir. Sonrada W ile F kayıtçılarını OR'lanır. Bu işlem için ise IORWF komutu kullanılır.

OR'lama sonucunda sonuç 0 ise STATUS'un 2.biti (Z) 1 , değilse 0 olacaktır. Bu biti kontrol ederek programı yönlendirebiliriz.

Örneğin W kayıtçısının 0 olup olmadığını aşağıdaki gibi test edebiliriz;

```
TEST_W IORLW h'00' ;W=(W OR h'00')
 BTFSS STATUS,Z ;Sonuç sıfır mı? (Z=1 mi?)
 GOTO TEST_W ;hayır.Tekrar kontrol et.
DEVAM ;evet.Programın devamı.....
```

### 1.6.14. Aritmetik İşlemler

PIC'ler bazı aritmetik işlemlerin yapılmasına da izin verirler. Biz bu modülde 8 ve 16 bitlik sayıların toplanması ve çıkarılmasını göreceğiz. Bunun için ADDLW, ADDWF, SUBLW, SUBWF komutlarını kullanacağımızdan, "mikrodenetleyici komutları" konusunda ilgili komutları incelemeniz yerinde olacaktır.

#### 1.6.14.1. 8 Bit Toplama

8 bitlik ( 1 Bayt ) lik iki veri iki şekilde toplanabilir;

Ø W kayıtçısı ile sabit bir sayı toplanır, sonuç W kayıtçısındadır (ADDLW).

Ø W kayıtçısı ile dosya kayıtçısı (f) toplanır, sonuç hedef kayıtçısındadır (ADDWF).

İki sayının toplamından elde edilen sonuç h'FF'den büyükse taşma olacağı için STATUS kayıtçısındaki elde bayrağı (C) 1 olacaktır. Şimdi her iki toplamaya da örnekler verelim.

**Örnek:** Önce W kayıtçısına h'25' değerini yükleyip daha sonrada W ile h'A3' sabitini toplayan program parçasını yazınız. Elde bayrağının durumunu bulunuz.

```
MOVLW h'25' ;W= h'25'
ADDLW h'A3' ;W= h'25'+ h'A3'= h'C8' olur. Taşma olmağından
C=0'da kalır.
```



**Örnek:** Önce W kayıtçısına h'7A' değerini, TOPLAM kayıtçısına da h'DD' yükleyip daha sonrada W ile TOPLAM kayıtçılarını toplayan ve sonucu TOPLAM kayıtçısına kaydeden program parçasını yazınız. Elde bayrağının durumunu bulunuz.

```

MOVLW h'DD' ;W= h'DD'
MOVWF TOPLAM ;TOPLAM= h'DD'
MOVLW h'7A' ;W= h'7A'
ADDWF TOPLAM,1 ;TOPLAM= h'DD'+ h'7A'= h'57'

```

Taşma olduğundan C=1 olur.Sonucun C bitinin dikkate alınmaması durumunda yanlış olacağına dikkat ediniz.

### 1.6.14.2. 16 Bit Toplama

Kayıtçılar 1 baytlık oldukları için, 16 bitlik (2 bayt) verilerin toplanması için 2 kayıtçı kullanmak gerekecektir. 2 baytlık veri 0-65535 onlu sayıları ifade edebilecektir. Bu işlemi yapmak için;

- Ø Toplanacak iki sayının önce alt (LSB) baytları yoplanır.
- Ø Eğer alt baytların toplamından elde oluşursa üst baytlardan birine 1 eklenir.
- Ø Üst baytlar toplanır.

Bunu bir örnekte görelim;

**Örnek:** A= h'33A5' ve B= h'8EDE' sayılarını toplayan sonucu C adında bir kayıtçıya atan program parçasını yazalım.

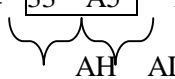
Çözüm: A= 

|    |    |
|----|----|
| 33 | A5 |
|----|----|

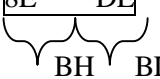
 B= 

|    |    |
|----|----|
| 8E | DE |
|----|----|



AH AL



BH BL

şeklinde sayıları yüklemek için 4 kayıtçı

**gerekir. Ayrıca sonucun bulunacağı kayıtçı da CH ve CL olarak 2 tane olmalıdır.**

```

MOVLW h'A5' ;W= h'A5'
MOVWF AL ;AL= h'A5'
MOVLW h'33' ;W= h'33'
MOVWF AH ;AH= h'33'
MOVLW h'DE' ;W= h'DE'
MOVWF BL ;BL= h'DE'
MOVLW h'8E' ;W= h'8E'
MOVWF BH ;BH= h'8E'
TOPLA MOVF AL,0 ;W=AL
ADDWF BL,0 ;W=BL+W(AL)
MOVWF CL ;Sonucun düşük baytı
BTFSC STATUS,0 ;C=0 mi?
INCF BH,1 ;Hayır(C=1) BH'ı 1 arttır.
MOVF AH,0 ;Evet. W=AH
ADDWF BH,0 ;W=BH+W(AH)
MOVWF CH ;Sonucun yüksek baytı.

```

### 1.6.14.3. 8 Bit Çıkarma

8 bitlik ( 1 Bayt ) lik iki veri iki şekilde çıkarılabilir;

- Ø Sabit bir sayıdan W kayıtçısı çıkarılır, sonuç W kayıtçısındadır (SUBLW).
- Ø Dosya kayıtçısı (f) içeriğinden W kayıtçısı çıkarılır , sonuç hedef kayıtçısındadır (SUBWF).
- Ø Çıkarma işlemi yapılırken çıkarılacak sayının tersi alınıp 1 eklenir (2'li tümleyeni alınır). Daha sonra her iki sayı da toplanır. PIC bu olayı otomatik olarak kendi yapar. Bu nedenle küçük sayıdan büyük sayı çıkarılırsa C=0 (sonuç negatif) , büyük sayıdan küçük sayı çıkarılırsa veya sayılar eşit ise C=1 (sonuç pozitif) olur. Yalnız küçük sayıdan büyük sayı çıkarılırsa gerçek sonucu bulmak için , sonucun 2'li tümleyeni alınmalıdır. Bu programcının sorumluluğundadır. Şimdi her iki çıkarmaya da örnekler verelim;

**Örnek:** Önce W kayıtçısına h'25' değerini yükleyip daha sonrada h'A3'sabitinden W'yi çıkaran program parçasını yazınız. Elde bayrağının durumunu bulunuz.

```
MOVLW h'25' ;W= h'25'
SUBLW h'A3' ;W= h'A3'-h'25' = h'7E' olur.
```

```
h'A3'= b'1010 0011' 1010 0011
SUB h'25'= b'0010 0101'>>>2'li tümleyeni>>> 1101 1011
 0111 1110
 ← 1
W = h'7E' ve C=1 Sonuç pozitif olur.
```

**Örnek:** Önce W kayıtçısına h'40' değerini, FARK kayıtçısına da h'25' yükleyip daha sonrada FARK kayıtçısından W'yi çıkaran ve sonucu FARK kayıtçısına yazan program parçasını yazınız. Elde bayrağının durumunu bulunuz.

```
MOVLW h'40' ;W= h'40'
MOVWF FARK ;FARK= h'40'
MOVLW h'35' ;W= h'35'
SUBWF FARK,1 ;FARK= h'40'- h'35'= h'F5' ;Yukarıdaki örnekteki
 ;gibi PIC 2'li tümleyen ile
 ;h'F0'sonucunu buldu.
 ;(C=0 sonuç negatif).
```

#### 1.6.14.4. 16 Bit Çıkarma

Kayıtçılar 1 baytlık oldukları için, 16 bitlik (2 bayt) verilerin çıkarılması için 2 kayıtçı kullanmak gerekecektir. 2 baytlık veri 0-65535 onlu sayıları ifade edebilecektir. Bu işlemi yapmak için;

- Ø Çıkarılacak iki sayının önce alt (LSB) baytları birbirinden çıkarılır.
- Ø Eğer alt baytların çıkarılmasından borç varsa (C=0) , üst baytlardan birisinden 1 çıkarılır.
- Ø Üst baytlar birbirinden çıkarılır.

Bunu bir örnekte görelim;

**Örnek:** A= h'33A5' ve B= h'8EDE' sayılarını çıkaran (B-A) sonucu C adında bir kayıtçıya atan program parçasını yazalım.

**Çözüm:** A= 

|    |    |
|----|----|
| 33 | A5 |
|----|----|

 B= 

|    |    |
|----|----|
| 8E | DE |
|----|----|

AH AL BH BL şeklinde sayıları yüklemek için 4 kayıtçı gerekiyor. Ayrıca Sonucun bulunacağı kayıtçı da CH ve CL olarak 2 tane olmalıdır.

|       |       |             |                                   |
|-------|-------|-------------|-----------------------------------|
| MOVLW | h'A5' | ;W= h'A5'   | } sayılar kayıtçılara yükleniyor. |
| MOVWF | AL    | ;AL= h'A5'  |                                   |
| MOVLW | h'33' | ; W= h'33'  |                                   |
| MOVWF | AH    | ; AH= h'33' |                                   |
| MOVLW | h'DE' | ; W= h'DE'  |                                   |
| MOVWF | BL    | ; BL= h'DE' |                                   |
| MOVLW | h'8E' | ; W= h'8E'  |                                   |
| MOVWF | BH    | ; BH= h'8E' |                                   |
| CIKAR | MOVF  | AL,0        | ;W=AL                             |
|       | SUBWF | BL,0        | ;W=BL-W(AL)                       |
|       | MOVWF | CL          | ;Sonucun düşük baytı              |
|       | BTFSS | STATUS,0    | ;C=1 mi?                          |
|       | DECF  | BH,1        | ;Hayır (C=0) BH'ı 1 Azalt.        |
|       | MOVF  | AH,0        | ;Evet. W=AH                       |
|       | SUBWF | BH,0        | ;W=BH-W(AH)                       |
|       | MOVWF | CH          | ;Sonucun yüksek baytı.            |

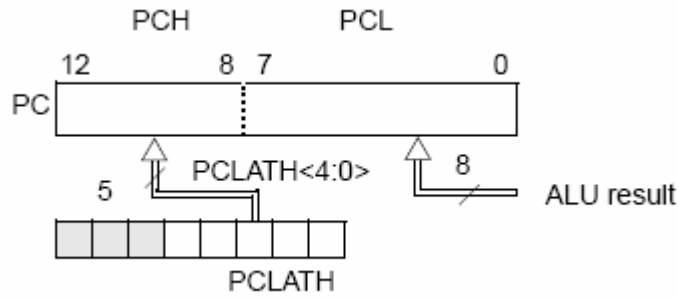
#### 1.6.15. Çevrim Tabloları

##### 1.6.15.1. Çevrim Tabloları

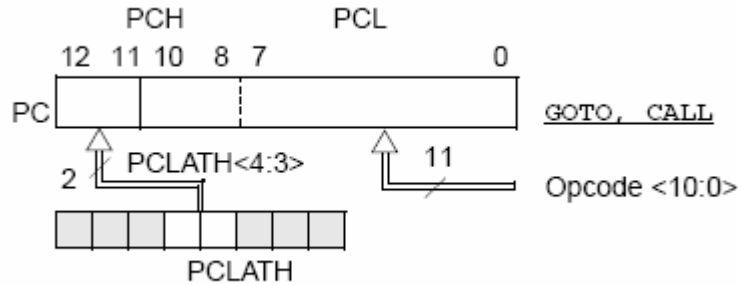
Çevrim tabloları sıralı değer alma işlemleri gereken yerlerde kullanılır. Bu şekilde bir kodu başka bir koda çevirebiliriz. Örneğin B portuna bağladığımız 7 segment display üzerinde hexadesimal karakterleri görmek istiyoruz. Çevrim tablosunda hexadesimal koda karşılık gelen uygun kodu B portuna göndermemiz gerekir. Ya da bir adım motorunu belirli bir şekilde döndürmek için porta bağlanmış step motoruna sıralı ikili değerleri göndermemiz gerekir. Bu tip durumlarda çevrim tablolarından faydalanılır.

### 1.6.15.2. Çevrim Tablolarının Kullanım Yerleri ve Kullanımı

Çevrim tablolarının kullanımı için PC'nin (program counter) yapısının iyice anlaşılmasında fayda olacaktır. Program sayacı (PC), 13 bit genişliğindedir. Düşük byte PCL registeri , okunabilir ve yazılabilir bir kayıtçı olup belleğin h'02' adresinde yer alır. PC'nin yüksek byte'ı , PCLATH registerinden gelir ve direkt olarak okunamaz ya da yazılamaz. PCLATH (PC latch high) registeri PC<12:8> için bir tutma registeridir ve PIC'e gerilim uygulandığında (power on reset) tüm bitleri '0'dır. PC yeni bir değerle yüklendiği zaman PCLATH'ın içeriği program sayacının üstteki byte'ına transfer edilir. Bu olay PCL'ye yazılımda, CALL veya GOTO esnasında olur. PC'nin yüksek bitleri PCLATH'dan yüklenmiştir. Bu olaylar şekil 1.17 ve şekil 1.18'de gösterilmiştir.



Şekil 1.17: Hedefi PCL kayıtçısı olan komutların PC'yi yüklemesi



Şekil 1.18: GOTO ve CALL komutlarının PC'yi yüklemesi

GOTO ve CALL komutlarıyla kullanılan 11 bitlik adreslemeler 2KB'lık program belleği bulunan PIC'lerde (16FCxxx ve 16Fxx) herhangi bir adrese erişim için yeterlidir ( $2^{11}=2KB$ ). 1KB belleği olanlarda (16F84) ise 10 bit yeterlidir ( $2^{10}=1KB$ ).

Çevrim tablolarının kullanımı için ayrıca RETLW komutunun da iyice anlaşılması gerekir. Bu komut RETURN komutu gibi anaprograma dönüşü sağlar. Tek fark dönüş esnasında W kayıtçısına bir sabit sayı yükler.

Şimdi de bir örnek ile çevrim tablolarının nasıl kullanıldığını görelim;

**Örnek:** B portunun uçlarına bağlı 7 segment display in 0-F arasında saydıran programı yazınız.

**Çözüm:** Öncelikle Hexadesimal koddan 7 segment display koduna dönüşüm tablosunu çıkarmalıyız;

| Hexadesimal sayı | 7 segmentte görülecek | 7 segment Sayı | 7 segment kodu uçlarındaki veri |
|------------------|-----------------------|----------------|---------------------------------|
| 00               | 0                     | 00111111       | 3F                              |
| 01               | 1                     | 00000110       | 06                              |
| 02               | 2                     | 01011011       | 5B                              |
| 03               | 3                     | 01001111       | 4F                              |
| 04               | 4                     | 01100110       | 66                              |
| 05               | 5                     | 01101101       | 6D                              |
| 06               | 6                     | 01111101       | 7D                              |
| 07               | 7                     | 00000111       | 07                              |
| 08               | 8                     | 01111111       | 7F                              |
| 09               | 9                     | 01101111       | 6F                              |
| 0A               | A                     | 01110111       | 77                              |
| 0B               | B                     | 01111100       | 7C                              |
| 0C               | C                     | 00111001       | 39                              |
| 0D               | D                     | 01011110       | 5E                              |
| 0E               | E                     | 01111001       | 79                              |
| 0F               | F                     | 01110001       | 71                              |
| Nokta            | .                     | 10000000       | 80                              |

Şimdi de programı yazalım. Programın “tablodan değer alma ve gecikme altprogramına dallan” bölümünde şu olaylar gerçekleşir;

AND komut satırındaki ifade ile W'nin üst 4 biti maskelendi. Dolayısıyla SAYAC değeri h'F'=15 değerini geçemez [0-15] arası sayar. Ve her sayıda tablodan bir değer alıp , B portuna bağlı 7 segment display'de göstererek GECİKME altprogramına dallanır.

;=====hex.asm=====

```

LIST P=16F84
INCLUDE "P16F84.INC"
SAYAC1 EQU h'0C'
SAYAC2 EQU h'0D'
SAYAC EQU h'0E'
CLRF PORTB
BSF STATUS,5
CLRF TRISB
BCF STATUS,5
MOVLW h'00'
MOVWF SAYAC

```

tanımlamaların yapıldığı bölüm

B portu çıkış olarak ayarlandı.

SAYAC sıfırlandı.

TEKRAR

```
MOVW SAYAC,W
ANDLW b'00001111'
CALL HEX_7SEGMENT
MOVWF PORTB
INCF SAYAC,F
CALL GECIKME
GOTO TEKRAR
```

} tablodan deęer al ve gecikme altprogramına dallan.

HEX\_7SEGMENT

```
ADDWF PCL,F ;PCL= W(SAYAC)
RETLW h'3F' ;0
RETLW h'06' ;1
RETLW h'5B' ;2
RETLW h'4F' ;3
RETLW h'66' ;4
RETLW h'6D' ;5
RETLW h'7D' ;6
RETLW h'07' ;7
RETLW h'7F' ;8
RETLW h'6F' ;9
RETLW h'77' ;A
RETLW h'7C' ;B
RETLW h'39' ;C
RETLW h'5E' ;D
RETLW h'79' ;E
RETLW h'71' ;F
```

} hexadesimalden 7 segment display'e dönüşüm tablosu.

GECIKME

```
MOVLW h'FF'
MOVWF SAYAC1
DON1
MOVLW h'FF'
MOVWF SAYAC2
DON2
DECFSZ SAYAC2,F
GOTO DON2
DECFSZ SAYAC1,F
GOTO DON1
RETURN
END
```

} GECİKME altprogramı.

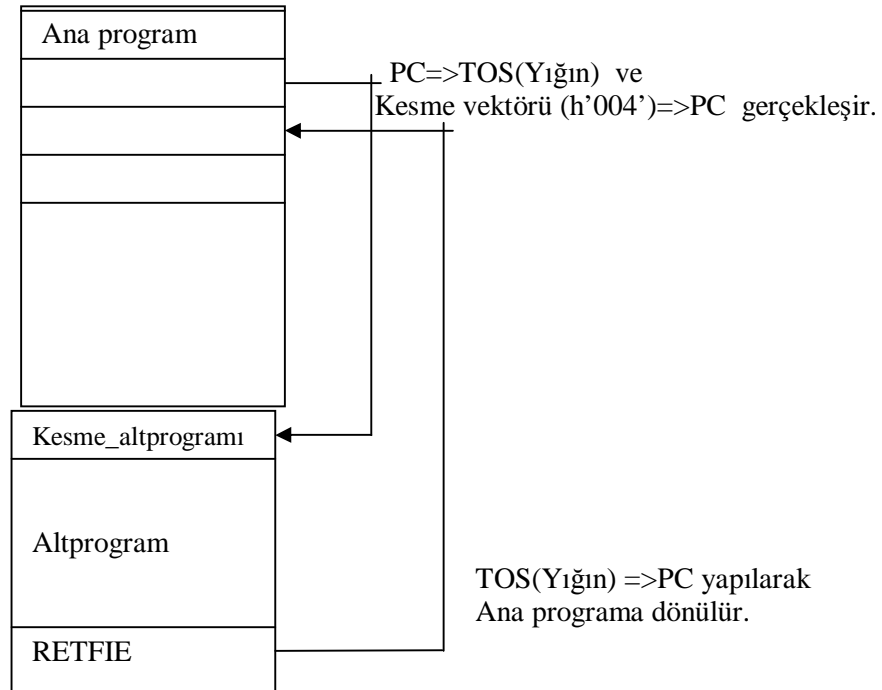
## 1.6.16. Kesmeler

Mikrodenetleyicilerle yeni çalışmaya başlayanlar kesme (interrupt) kelimesini duymalarına rağmen, kullanımlarının zor olduğu düşüncesiyle programları içerisinde kullanmaktan çekinirler. Oysa öğrenilmesi ve uygulaması pek de zor olmayan kesme altprogramları kullanarak , program içerisinde kullanılacak komut sayısı azaltılabilir ve bir çok mantıksal karışıklıklar önlenir.

### 1.6.16.1. Kesme Nedir?

Kesme işlemini günlük hayattan bir örnek vererek açıklayalım; mesela yemek yiyorsunuz ve telefon çaldı. Ne yaparsınız? Yemek yemeyi bir süre için bırakır, telefona bakar sonra dönüp yemek yemeye devam edersiniz . İşte günlük hayatta karşılaştığımız bir kesme. Asıl işiniz yemek yeme olayı iken, ihtiyaçtan (telefon çalması) dolayı asıl işinizi bırakıp, diğer bir işi yaptınız (telefona bakma) ve geri dönüp asıl işinizde kaldığınız yerden devam ettiniz.

Mikrodenetleyicilerdeki kesme olayı yukarıdaki örneğe oldukça benzerdir. Kesme, mikrodenetleyicilerin herhangi bir kesme kaynağından gelen sinyal nedeniyle çalışmakta olan programı bırakması ve önceden hazırlanan kesme altprogramını çalıştırması olayıdır. Altprogramın çalışması bittikten sonra ana program kaldığı yerden itibaren tekrar çalışmasına devam eder. Şekil 1.19'u inceleyiniz.



Şekil 1.19: Kesme altprogram kurgusu

### 1.6.16.2. INTCON Kayıtçısı (Kaydedicisi)

INTCON registerı, bütün kesme kaynakları için olan çeşitli yetkilendirici bitleri içeren, okunabilir ve de yazılabilir türde bir registerdır.

Aşağıda PIC16F8X mikrodenetleyicisi için INTCON kayıtçısının her bir bitinin hangi durumlarda 1 ve 0 olacağı gösterilmiştir;

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| GIE   | EEIE  | TOIE  | INTE  | RBIE  | TOIF  | INTF  | RBIF  |

bit7

bit0

R = Okunabilir bit.

W = Yazılabilir bit.

-n = Güç reseti (POR reset) durumundaki bit değeri.

- bit 7: **GIE**:Global Kesme biti yetkisi.  
1=Tüm kesmelere yetki verir.  
0=Tüm kesmeleri yetkisizleştirir.
- bit 6: **EEIE**:EEPROM belleğe yazma işlemi tamamlama kesim yetki biti.  
1= EEPROM yazma tamamlama kesmesine yetki verir.  
0= EEPROM yazma tamamlama kesmesine yetkiyi kaldırır .
- bit 5: **TOIE**:TMR0 sayıcı taşma (Overflow) kesmesi yetki biti.  
1=TMR0 kesmesini yetkilendirir.  
0=TMR0 kesmesine yetkiyi kaldırır.
- bit 4: **INTE**: RB0/INT hâricî kesmesi yetki biti.  
1=RB0/INT kesmesine yetki verir.  
0=RB0/INT kesmesine yetkiyi kaldırır.
- bit 3: **RBIE**: Port B (4,5,6ve 7. bitleri) değişiklik kesmesi yetki biti.  
1= Port B değişiklik kesmesine yetki verir. Yani B portundaki değişiklikler kesme oluşturur.  
0= Port B değişiklik kesmesine yetkiyi kaldırır.
- bit 2: **TOIF**: TMR0 sayıcısı taşması durum bayrağı.  
1=TMR0 taşmıştır (yazılımın içinden silinmesi gerekiyor).  
0=TMR0 taşmamıştır. Zaman aşımı yok.
- bit 1: **INTF**: RB0/INT hâricî kesme durum bayrağı.  
1=RB0/INT kesmesi meydana gelmiştir.  
0=RB0/INT kesmesi meydana gelmemiştir.
- bit 0: **RBIF**: Port B değişiklik bayrağı.  
1=RB7:RB4 pinlerinden en az bir tanesinin durumu değişti (yazılım içinden silinmesi gerekiyor).  
0=RB7:RB4 pinlerinden hiç bir tanesinde durum değişikliği yoktur.



### 1.6.16.3. Kesme Kaynakları

Kesme kaynakları mikrokontrolür ailesine bağılı olarak farklılık gösterir. Aşağıda PIC 16F8X ailesi için olabilecek kesme kaynakları sıralanmıştır.

- Ø Hâricî (External) kesme: PIC16F84'ün RB0/INT ucundan gelen sinyal ile oluşur.
- Ø Hâricî kesmelerin kullanabilmesi için 2 şey gereklidir; yazılım ve donanım. Yazılım aracılığı ile B portunun RB0 ucu, dışarıdan gelebilecek kesmeyi alabilecek şekilde hazırlanmalıdır. Bunun için iki işlem yapılır;
  - RB0 ucu giriş olarak yönlendirilmeli.
  - INTCON kayıtçısı içerisinde ilgili bayrak (INTE bayrağı) kullanılarak hâricî kesme işlemi için yetkilendirilmeli.

Hâricî kesmenin kullanılması için bir de donanım gereksinimi vardır. Bu da RB0 ucundan kesme sinyalini verecek elektronik devredir.

Ayrıca RB0 ucundan uygulanacak sinyalin kenar tetiklemesi önemlidir. OPTION kayıtçısının 6. biti kesmeyi yapacak sinyalin kenar tetiklemesini belirler. Eğer bu bit 0 ise kesme sinyalin düşen kenarında , 1 ise yükselen kenarında gerçekleşir.

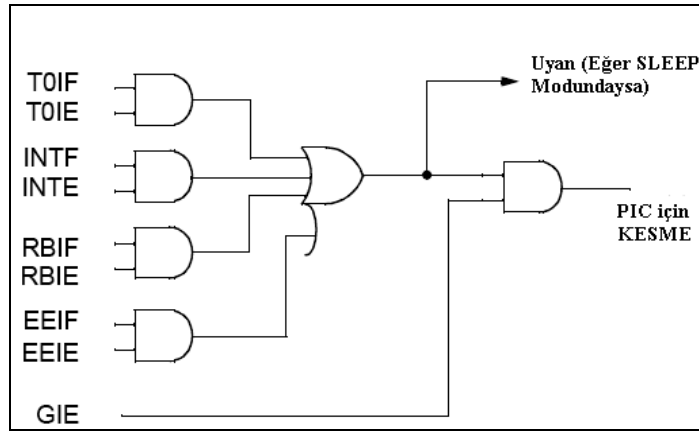
Kesme altprogramın çalışması esnasında gelebilecek yeni kesmeleri engellemek için INTCON kayıtçısındaki INTE biti 0 yapılarak hâricî kesme yetkisi kaldırılmalıdır. Bununla beraber bir hâricî kesme olursa INTCON kayıtçısı içindeki INTF 1 olur. Bu bayrak kesme altprogramı içerisinde tekrar 0 yapılmalıdır. Aksi durumda tekrarlanan kesmelerle karşılaşılır.

- Ø TMR0 sayıcısında oluşan zaman aşımı kesmesi; yani TMR0 sayıcısının h'ff' den h'00'a gelmesi olayında oluşur. Bu konu ileride TMR0 sayıcısı konusunda işlenecektir.
- Ø B Portunun 4,5,6 veya 7. bitlerindeki lojik seviye değişikliği;
- Ø B portunun 4,5,6 ve 7. bitlerindeki lojik değişim INTCON kayıtçısının RBIF bayrağını 1 yapacaktır. Bu kesme INTCON kayıtçısının RBIE biti ile aktif ya da pasif yapılabilir.
- Ø B portundaki değişikliği algılamak için bu portaki en son değer, RB4-RB7 uçlarından okunan veri ile karşılaştırılır. Eski ve yeni okunan veri OR'lanır. Farklılık varsa RBIF bayrağı 1 olur.

B Portu kesmesi aşağıdaki gibi silinebilir;

- RBIE biti silinmek suretiyle,
  - B Portunu okuduktan sonra RBIF bitini silmek suretiyle.
- Ø EEPROM belleğe yazma işleminin tamamlanması; EEPROM veri yazma zamanı yaklaşık 10 ms kadardır. Yazma işlemi bittiğinde EEIF bayrağı 1 olur ve yazmanın bittiğini gösterir. EEIF bayrağı EECON1 kayıtçısında bulunur.

Yukarıda anlatılan kesme kaynakları mantığı şekil 1.20’de verilmiştir.



Şekil 1.20: PIC16F84'ün kesme kaynakları ve bayrakları

#### 1.6.16.4. Kesme Altprogramlarının Düzenlenmesi

Bir kesme olayının meydana gelmesi esnasında INTCON kayıtçısının GIE biti 0 olur. Bu işlem kesme alt programının çalışması esnasında yeni bir kesmenin olmaması için PIC tarafından otomatik olarak yapılır. Kesme altprogramı çalışmasını RETFIE komutu ile sona erdirip ana programa döndüğü anda ise sonraki kesmelerin geçerli olabilmesi için tekrar otomatik olarak 1 yapılır. Kesme işleminde meydana gelen olayları şekil 1.21’den inceleyiniz.

Bir kesme meydana geldiğinde kesme vektörü (interrupt vector) denilen adres gösterici PC’ye program belleğinin h’0004’ adresini gösterir. Kesme altprogramın ilk komutu ( genellikle GOTO ) buraya yazılmalıdır.

Kesme kullanılmadığı zaman ana program, program belleğinin h’0000’ adresinden itibaren h’0004’ adresine doğru problem çıkarmadan çalışır. Eğer kesme kullanılıyorsa programcı tarafından başka bir çalışma sırası düzenlenmelidir. Bu düzen aşağıdaki gibi olabilir.

```
ORG h'000' ;Reset vektör adresi.
GOTO BASLA ;Ana program başlangıcı.
ORG h'004' ;Kesme vektör adresi
GOTO KESME
```

BASLA

;Ana program komutları

{ Ana program  
buraya  
yazılmalıdır. }

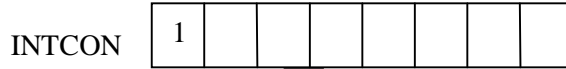
KESME

;kesme altprogram komutları

{ Kesme alt programı  
buraya yazılmalıdır. }

.

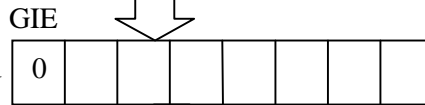
RETFIE ;Ana programa dön.



GIE



INTCON



PC= TOS  
PC (program sayıcı) değeri yığına itilir.

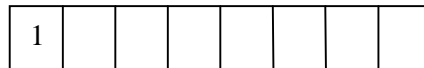
h'004' = PC  
PC reset vektörüne dallanır. Burada GOTO  
komutu kesme alt programına dallanılır.

Kesme altprogramı çalıştırılır.

İlgili kesme bayrağı kontrol edilir.

RETFIE komutu ile ana programa geçilir.

INTCON



Şekil 1.21: Kesme işleminde oluşan olaylar

Bir kesme esnasında sadece geri dönme PC değeri yığında saklanır. Çoğunlukla , kullanıcılar bir kesme esnasında anahtar register değerlerini saklamak isterler (Örneğin W registeri ve STATUS registeri). Bu, yazılımda yerine getirilir.

Örnek 1’de STATUS ve W registerlerinin değerleri saklanmış ve tekrar depolanmıştır. Kullanıcı tanımlı registerler W\_TEMP ve STATUS\_TEMP , W ve STATUS register değerleri için geçici depolama yerleridir.

Örnek 1’de aşağıdakiler yapılmaktadır:

- Ø W registerini bir değişkene (W =W\_TEMP) yükle.
- Ø STATUS registerini bir değişkene (STATUS=STATUS\_TEMP) yükle
- Ø Kesme altprogramını çalıştır .
- Ø STATUS registerini geri yükle (STATUS\_TEMP= STATUS).
- Ø W registerini geri yükle (W\_TEMP=W).
- Ø Kesme altprogramından geri dön (RETFIE).
- Ø Şimdi bu işlemleri yapacak örneğimizi yazalım.

**Örnek 1:** Kesme esnasında W ve STATUS kayıtçı değerlerini saklamak.

```
MOVWF W_TEMP ; W , TEMP registerine kopyalanır
SWAPF STATUS , W ; STATUS içeriğini SWAP’la ve W’ye yükle.
MOVWF STATUS_TEMP ; STATUS içeriğini STATUS_TEMP
;registerine yükle.
```

; { Kesme altprogramı }  
; { buraya yazılmalı. }

```
SWAPF STATUS_TEMP , W ; STATUS_TEMP içeriğini yeniden
;SWAP’la ve W’ye yükle.
```

```
MOVWF STATUS ; W’yi STATUS Registerine yükle.
```

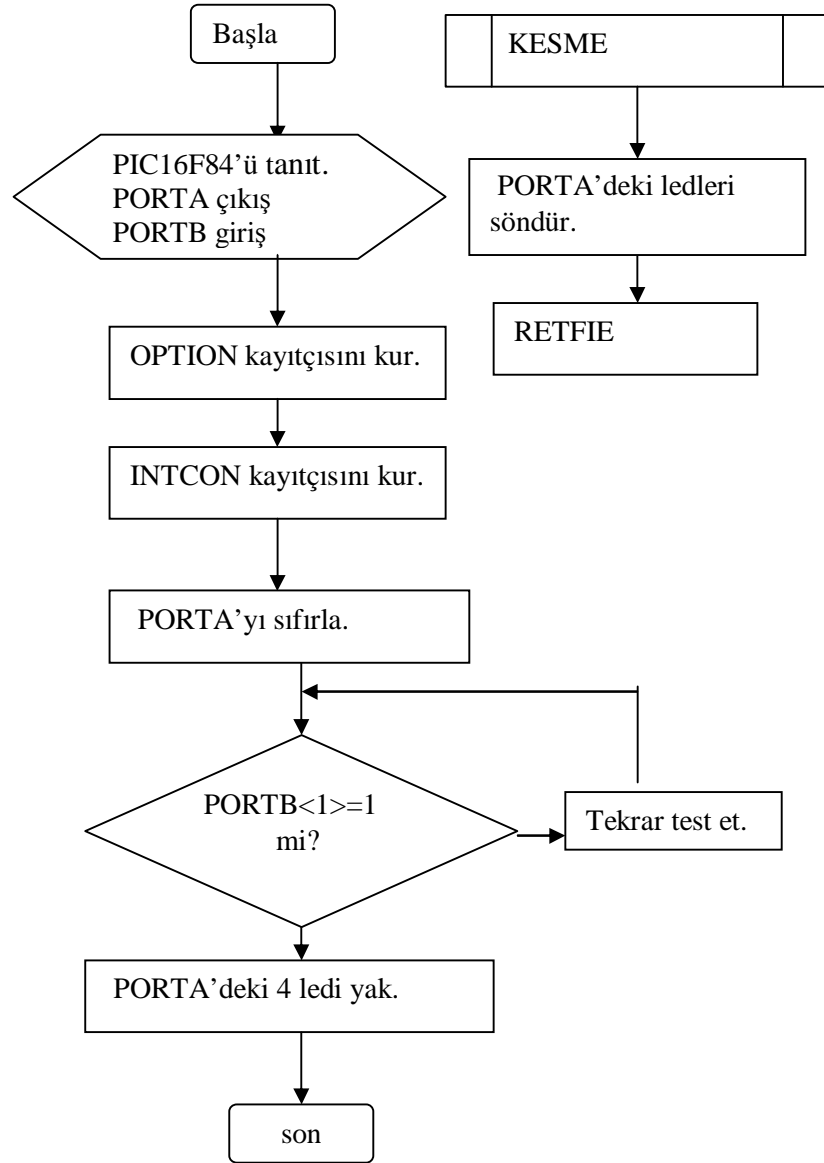
```
SWAPF W_TEMP , F ; W_TEMP içeriğini SWAP’la sonucu
;W_TEMP’e tekrar yükle.
```

```
SWAPF W_TEMP , W ; W_TEMP içeriğini SWAP’la sonucu
;W’ye tekrar ;yükle.
```

Yukarıdaki işlem için neden MOVF komutunu kullanmadığımız aklınıza gelmiş olabilir. Kullanamayız, çünkü MOVF komutu, STATUS kayıtçısındaki Z bayrağını değiştireceğinden, programda bu bayrağın kontrolü ile yapılan işlemlerde hatalara neden olacaktır.

**Örnek2:** PIC16F84'e enerji verildiğinde A portundaki tüm ledler sönmük iken B portunun 0.bitine bağlı (RB1) bir butuna basıldığında A potuna bağlı ledler yansın. Daha hâricî kesme ile (RB0/INT) ledler tekrar söndürülsün. Kesme, kesme sinyalinin düşen kenarında gerçekleşsin.

Çözüm: Problemdeki verilere göre aşağıdaki algoritmayı çizebiliriz;



Şekil 1.22: Örnek 2'deki programın algoritması

Şimdi de asm programımızı yazalım;

;====kesme.asm=====

```
 LIST P=16F84
 INCLUDE "P16F84.INC"
 ORG h'000'
 GOTO BASLA
 ORG h'004'
 GOTO KESME

BASLA BSF STATUS,5
 CLRF TRISA
 MOVLW h'FF'
 MOVWF TRISB
 MOVLW b'10111111'
 MOVWF OPTION_REG
 BCF STATUS,5
 CLRF PORTA
 BCF INTCON,1
 BSF INTCON,7
 BSF INTCON,4

 } A portu çıkış, B portu giriş olarak
 } ayarlandı.

 ;düşen kenarda kesme palsı ,
 ;gelecek şekilde OPTION_REG'i kur.
 ;Bank0'a geç.
 ;A portunu sıfırla.
 ;INF bayrağını sil.
 ;Global kesmeyi etkinleştir.
 ;Hâricî kesmeyi aktifleştire.

TEST_PORTB
 BTFSS PORTB,0 ;PORTA<0>=1 mi?
 GOTO TEST_PORTA ;hayır.Tekrar test et.
 MOVLW h'FF' ;evet. W=h'ff' yükle.
 MOVWF PORTA ;A portundaki tüm ledleri yak.
 GOTO TEST_PORTB
 END

KESME
 BCF INTCON,1 ;INTF bayrağını sil.
 MOVLW h'00' ;W=h'00' yükle.
 MOVWF PORTA ;A portundaki tüm ledleri söndür.
 RETFIE
 END
```

### 1.6.17. Donanım Sayıcıları

Zaman geciktirme döngüleri konusunda, port çıkışlarına göndereceğimiz sinyaller arasında bir gecikme olmasını istediğimiz zaman bunu yazılım (software) ile nasıl yapacağımızı gördük. Burada ise aynı işlemi donanım zamanlayıcısını (hardware) kullanarak nasıl yapacağımızı göreceğiz. Bunun için donanım zamanlayıcısını (TMR0) kullanacağız. Bu zamanlayıcı / sayıcıların sayısı kullanılan mikrokontrolöre göre değişmekle beraber, biz genel olan TMR0'ın kullanımını göreceğiz.

### 1.6.17.1. Donanım Sayıcısı ( zamanlayıcısı ) Nedir?

Peki donanım zamanlayıcısı ya da sayıcısı nedir? Belli bir değerden (genelde h'00') başlayıp değeri yukarı doğru artan bir dosya kayıtcısıdır. PIC'ler iki tip zamanlayıcıya sahiptir; ilki TMR0 adı verilen 8 bitlik bir sayıcıdır. Bu RAM belleğin h'01' adresinde bulunan özel bir kayıtcıdır. Diğeri de WDT (Watchdog timer) denilen bir zamanlayıcıdır. WDT zamanlayıcısı ileriki konularda işlenecektir.

### 1.6.17.2. TMR0 Sayıcısı ( zamanlayıcısı )

TMR0 sayıcısı RAM belleğin h'01' adresinde bulunan özel bir kayıtcıdır. TMR0 programlanabilen bir sayıcıdır. Yani saymaya istenilen bir sayıdan başlatılabilir ve herhangi bir anda içeriği silinebilir. Timer0 (TMR0) zamanlama/sayıcısı aşağıdaki özelliklere sahiptir:

- Ø 8 bit'lik zamanlama/sayıcı,
- Ø Okunabilirlik ve yazılabilirlik,
- Ø Programlanabilen frekans bölme değeri özelliği(prescaler),
- Ø Dâhilî ve hâricî saat palsı seçimi,
- Ø FFh'dan 00h'a düşerken taşma üzerinden kesme oluşturma,
- Ø Hâricî saat için düşen veya yükselen kenar seçimine (edge select) sahiptir.

TMR0 sayıcısının önemli özelliklerinden biri de ana program veya kesme altprogramları çalışırken sayma işlemini durdurmamasıdır. Sayma işleminde FFh'dan 00h'a geçişte oluşan taşma INTCON kayıtcısının TOIF bayrağında görülür ve bu bayrak 1 olur (INTCON kayıtcısı konusuna bakabilirsiniz). İstenirse bu bayrak test edilerek bir kesme altprogramı çalıştırılabilir. Şimdi de TMR0 ve WDT zamanlayıcıların kontrolünde kullanılan OPTION kayıtcısını görelim.

### 1.6.17.3. Option Kayıtcısı

OPTION registeri, TMR0/WDT için frekans bölme için gerekli bitleri bulduran, hâricî kesme sinyal tipini belirleyen, PORTB üzerindeki zayıf Pull-up dirençlerini ayarlayan çeşitli kontrol bitlerini içeren, TMR0 veya WDT'yi seçme bayrağı bulduran, yazılabilen ve de okunabilen tipte bir registerdir.

|       |        |       |       |       |       |       |       |
|-------|--------|-------|-------|-------|-------|-------|-------|
| R/W-1 | R/W-1  | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| RBPU  | INTEDG | T0CS  | T0SE  | PSA   | PS2   | PS1   | PS0   |
| bit7  |        |       |       |       |       |       | bit0  |

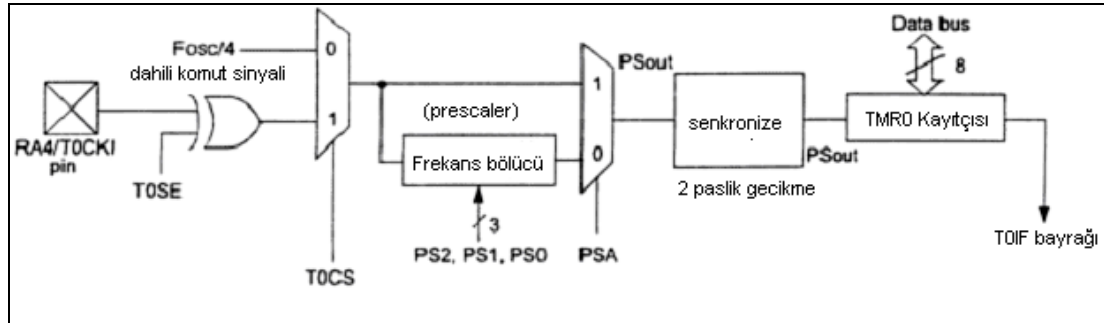
Şekil 1.23: Option kayıtcısı içeriği

- it 7: **RBPU**:PORTB Pull-up dirençleri yetki biti.  
1=PORTB Pull-up'lar pasif durumundadırlar (yetki vermeme).  
0=PORTB Pull-up'lar aktif durumundadırlar (yetki verme).  
Pull-Up dirençleri aktif yapıldığında B portu pinleri 50KΩ luk dirençlerle +Vcc'ye bağlanır.
- bit 6: **INTEDG**:Hâricî kesme sinyali kenar seçim biti.  
1=RB0/INT pini sinyalin yükselen kenarında kesme yapar.  
0= RB0/INT pini sinyalin düşen kenarında kesme yapar.
- bit 5: **TOCS**:TMR0 sinyal kaynağı seçim biti.  
1=RA4/T0CKI pini üzerinden hâricî sinyal.  
0=Dâhilî komut palsleri (CLKOUT).
- bit 4: **T0SE**: TMR0 sinyal kaynağı kenar seçim biti.  
1=RA4/T0CKI pini üzerindeki sinyalin yükselen kenarında.  
0=RA4/T0CKI pini üzerindeki sinyalin düşen kenarında.
- bit 3: **PSA**: Frekans bölücü seçim biti.  
1= Frekans bölücü sayısı ,WDT'a tayin edilir.  
0= Frekans bölücü sayısı ,TMR0'a tayin edilir.
- bit 2-0: **PS2:PS0**:Ön-ölçücü oran seçim bitleri

| Bit 2-0 Değeri | TMR0 Oranı | WDT Oranı |
|----------------|------------|-----------|
| 000            | 1 : 2      | 1 : 1     |
| 001            | 1 : 4      | 1 : 2     |
| 010            | 1 : 8      | 1 : 4     |
| 011            | 1 : 16     | 1 : 8     |
| 100            | 1 : 32     | 1 : 16    |
| 101            | 1 : 64     | 1 : 32    |
| 110            | 1 : 128    | 1 : 64    |
| 111            | 1 : 256    | 1 : 128   |

#### 1.6.17.4. TMR0 Sayıcısının Özellikleri

TMR0'ın çalışması ve özelliklerini anlamak için şekil 1.24'deki blok şemayı incelemeniz faydalı olacaktır.





### Şekil 1.24: TMR0 yapısı

- Ø Sayıcı içindeki sayının artması için gerekli saat palsı iki kaynaktan alınabilir:
  - Dâhilî komut sinyali (T0CS=0 durumu)
  - Hâricî saat palsı (T0CS=1 durumu). Bu durumda hâricî sinyal A portunun 3.bitinden (RA4) uygulanır. Hâricî sinyal PIC tarafından sayılması gereken bir sinyal de olabilir.
- Ø Sinyal kaynağından gelen sinyal direkt olarak TMR0'ı besleyebileceği gibi frekans bölücü aracılığı ile de besleyebilir.
- Ø OPTION kayıtçısının 0,1 ve 2.bitleri kullanarak 8 farklı frekans bölme değeri seçilebilir. TMR0 sayıcısını tetikleyecek sinyal, frekans bölücü kullanılmadan direkt olarak kullanılmak isteniyorsa , frekans bölme değeri WD'ye atanır. Bu işlem PSA biti ile gerçekleştirilebilir. Ayrıca frekans bölme oranlarının TMR0 ve WDT için farklı olduğuna dikkat ediniz.
- Ø Frekans bölme değerleri TMR0'a atandığında ,TMR0'a yazmak için kullanılan tüm komutlar frekans bölme değerini siler (CLRF , MOVWF, BSF....gibi).
- Ø Frekans bölme değeri kullanılmadan, direkt olarak hâricî sinyal kullanılırsa, dâhilî komut sinyali ile eşleşmeyi (senkron) sağlamak için 2 sayıklık bir gecikme sağlanır.
- Ø TMR0 sayıcısında çıkış sinyalleri iki şekilde oluşur:
  - RAM belleğin h'01' adresindeki TMR0 kayıtçısının okunmasıyla,
  - Sayıcının h'ff'den h'00'a geçişinde oluşan taşmadan dolayı INTCON kayıtçısının TOIF bayrağına 1 yazılması durumunda.
- Ø Kaynak olarak hâricî sinyal seçildiğinde istenirse bu sinyaller sayılabilir. Bu durumda TMR0 sinyal sayıcı olarak kullanılmış olur. Bu uygulama devir sayıcı, ürün saydırma gibi işlemler için kullanılabilir.
- Ø Frekans bölme değerleri, TMR0 veya WDT sayıcılarının kaç dâhilî komut sayıklında bir defa bir üst sayıya geçişini belirleyen orandır. Örneğin,  
TMR0 oranı 1/2 ise , 2 komut sayıklında bir defa üst sayıya geçiş olur.  
TMR0 oranı 1/8 ise , 8 komut sayıklında bir defa üst sayıya geçiş olur.  
TMR0 oranı 1/256 ise , 256 komut sayıklında bir defa üst sayıya geçiş olur.

Bilindiği gibi program belleğine yerleştirilen komutların çalışması için PIC'e hâricî bir osilatörden saat palsı ( $f_{osc}$ ) uygulamak gerekir. Bu frekans PIC tarafından 4'e bölünerek OSC2 ucundan dışarı verilir. İşte bu sinyalin 1 sayıklı için geçen süre (periyodu), bir komutun çalışması için geçen zamandır. Bu durumda ;

$$F_{\text{komut}} = f_{\text{osc}} / 4$$

$$T_{\text{komut}} = 1 / F_{\text{komut}}$$

$$\text{TMR0 sayma aralığı zamanı} = T_{\text{komut}} \times (\text{TMR0 oranı})^{-1}$$

Kesme olayı sayının h'ff'den h'00'a geçişinde oluştuğuna göre ve h'ff'=256 olduğundan ;

$$\text{Kesme gecikmesi} = \text{TMR0 sayma aralığı zamanı} \times 256 \text{ olur.}$$

Bütün bu formülleri tek formülde birleştirirsek;

$$\text{Kesme gecikmesi} = \frac{(256 - \text{TMR0 ilk değeri}) \times (\text{TMR0 oranı})^{-1}}{f_{\text{osc}} / 4} \mu\text{S olur.}$$

Prescaler: Frekans bölme sayısı (2,4,8,...,256)

TMR0 ilk değeri: TMR0'a atanan ilk sayı. Eğer değer atanmamışsa '0'dır .

Bu formülden, istenirse belli bir kesme süresi için, TMR0'a yüklenecek sayı formülü de çıkarılabilir. Bu durumda;

$$\text{TMR0 ilk değeri} = 256 - [(\text{Kesme gecikmesi} \times f_{\text{osc}}) / 4 \times (\text{TMR0 oranı})^{-1}] \text{ olur.}$$

**Örnek 1:** 4MHz'lik kristal osilatör kullanan bir PIC'de, frekans bölme değeri (prescaler) b'000' seçilirse, kesmenin ne zaman olacağını bulalım.

Frekans bölme değeri (prescaler) b'000' olduğundan TMR0 oranı 1/2 olur.

$$F_{\text{komut}} = f_{\text{osc}} / 4 = 4\text{Mhz} / 4 = 1 \text{ Mhz}$$

$$T_{\text{komut}} = 1 / F_{\text{komut}} = 1 / 1 \text{ Mhz} = 1\mu\text{S}$$

$$\text{TMR0 sayma aralığı zamanı} = 1\mu\text{S} \times (\text{TMR0 oranı})^{-1}$$

TMR0 sayma aralığı zamanı =  $1\mu\text{S} \times (1/2)^{-1} = 2 \mu\text{S}$  ; yani her 2  $\mu\text{S}$ 'de bir sayı yukarı doğru artıyor. Kesme olayı sayının h'ff'den h'00'a geçişinde oluştuğuna göre ve h'ff'=256 olduğundan ;

$$\text{Kesme gecikmesi} = \text{TMR0 sayma aralığı zamanı} \times 256$$

Kesme gecikmesi =  $2 \mu\text{S} \times 256 = 512 \mu\text{S}$ 'lik periyotlarla kesme olayı oluşur.

Veya;

$$\text{Kesme gecikmesi} = \frac{(256 - \text{TMR0 ilk deęeri}) \times (\text{TMR0 oranı})^{-1}}{f_{\text{osc}} / 4}$$

$$\text{Kesme gecikmesi} = \frac{(256 - 0) \times 2}{4 \text{ Mhz} / 4} = 512 \mu\text{S} \text{ olur.}$$

**Örnek 2:** 4MHz'lik kristal osilatör kullanan bir PIC'de, frekans bölme deęeri (prescaler) b'110' ise 1.28 ms'lik gecikme için TMR0'a hangi sayının atanması gerekir?

$$\begin{aligned} 1.28 \text{ ms} &= 1280 \mu\text{S} \text{ yapar ve (prescaler) b'110' için TMR0 oranı} = 1 / 128 \text{ dir.} \\ \text{TMR0 ilk deęeri} &= 256 - [(\text{Kesme gecikmesi} \times f_{\text{osc}}) / 4 \times (\text{TMR0 oranı})^{-1}] \\ \text{TMR0 ilk deęeri} &= 256 - [(1280 \times 4 \text{ Mhz}) / 4 \times 128] \\ \text{TMR0 ilk deęeri} &= 246 \text{ desimal deęeri bulunur.} \end{aligned}$$

Bunu hexadecimal formata dönüştürürsek h'F6' olur.

TMR0 '0'dan deęil de örneğimizdeki gibi başka bir sayıdan başlatılmak isteniyorsa , yapılması gereken TMR0'a MOVWF komutu ile o sayının yüklenmesidir.

Ø Prescaler deęerini, **TMR0' dan WDT' a veya WDT'dan TMR0' a** atama işlemi yapılırken, prescalerın sıfırlanması nedeniyle PIC'in çalışması esnasında istenmeyen resetleri engellemek için aşağıdaki komut sırası takip edilerek prescaler tayininin Timer0'dan WDT'ye deęiştirilmesi sağlanmalıdır. Bu sıra , WDT mümkün kılınmadığı durumlara da takip edilmelidir. Prescalerı WDT'den Timer0'a deęiştirmek için de verilen sıra takip edilmelidir.

- TMR0' dan WDT' a Prescaler deęeri atamak (TMR0→WDT) için aşağıdaki sıra takip edilmelidir.

|        |             |                                    |
|--------|-------------|------------------------------------|
| BCF    | STATUS, RP0 | ; Bank 0'a geç.                    |
| CLRF   | TMR0        | ; TMR0 ve prescaler silinir.       |
| BSF    | STATUS, RP0 | ; Bank 1'e geç.                    |
| CLRWDT |             | ; WDT silinir.                     |
| MOVLW  | b'xxxx1xxx' | ; Yeni prescaler WDT için seçilir. |
| MOVWF  | OPTION      | ; OPTION kayıtçısına yaz.          |
| BCF    | STATUS, RP0 | ; Bank 0'a geç.                    |

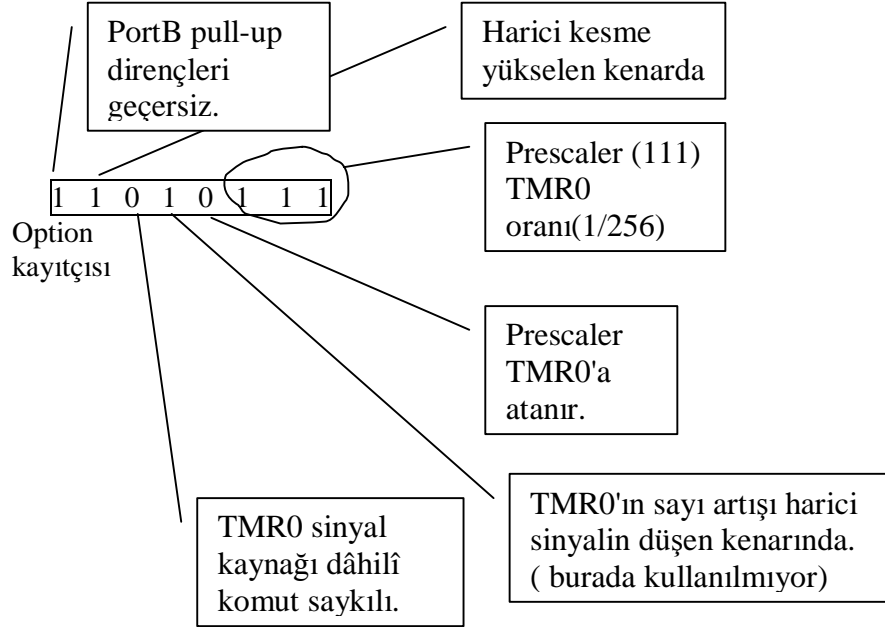
- WDT'dan TMR0'a prescaler değeri atamak (WDT→TMR0) için aşağıdaki sıra takip edilmelidir.

CLRWDT ; WDT ve prescaler silinir.  
 BSF STATUS, RP0 ; Bank 1'e geç.  
 MOVLW b'xxxx0xxx' ;TMR0, yeni prescaler değeri ve saat kaynağı seçilir.  
 MOVWF OPTION ; OPTION kayıtcısına yaz.  
 BCF STATUS, RP0 ; Bank 0'a geç.

**Örnek:** Şimdi de B portunun 1.bitinden kare dalga sinyal veren programı yazalım. Sinyal kaynağı olarak dâhilî sinyal kaynağını kullanalım ve TMR0 oranını 1 / 256 olacak şekilde prescaler değerini ayarlayalım. Çıkış sinyali PIC'in RB1 bitine osilaskobun bağlanmasıyla izlenecektir.

Şekil 1.26'deki akış diyagramını inceleyiniz. Bu diyagramda;

- Ø TMR0 <6>=1 ifadesiyle TMR0'ın 6.biti 1 mi? Sorgusu yapılmıştır.
- Ø GECIKME etiketi ile altprograma dallanılmıştır. Burada Türkçe karakterlerinin kullanılmasını hatırlayın.
- Ø Program içindeki MOVLW b'11010111' ve MOVWF OPTION\_REG komut satırları ile OPTION kayıtcısı aşağıdaki gibi ayarlanmıştır;



Şekil 1.25: Program için OPTION kayıtcısının kurulması

- Ø TMRO kayıtçısının tamamı olunabilir veya bu örnekteki gibi sadece bir biti test edilebilir. TMR0 <6>=1 olduğunda ulaşılan sayı b'0100 0000'=64'tür (onlu). PIC'in 4 Mhz'lik kristal osilatörle kullanılacağını düşünürsek GECİKME altprogramının süresini aşağıdaki gibi bulabiliriz;

$$F_{\text{komut}} = f_{\text{osc}} / 4 = 4\text{Mhz} / 4 = 1\text{ Mhz}$$

$$T_{\text{komut}} = 1 / F_{\text{komut}} = 1 / 1\text{ Mhz} = 1\mu\text{S}$$

$$\text{TMR0 sayma aralığı zamanı} = 1\mu\text{S} \times (\text{TMR0 oranı})^{-1}$$

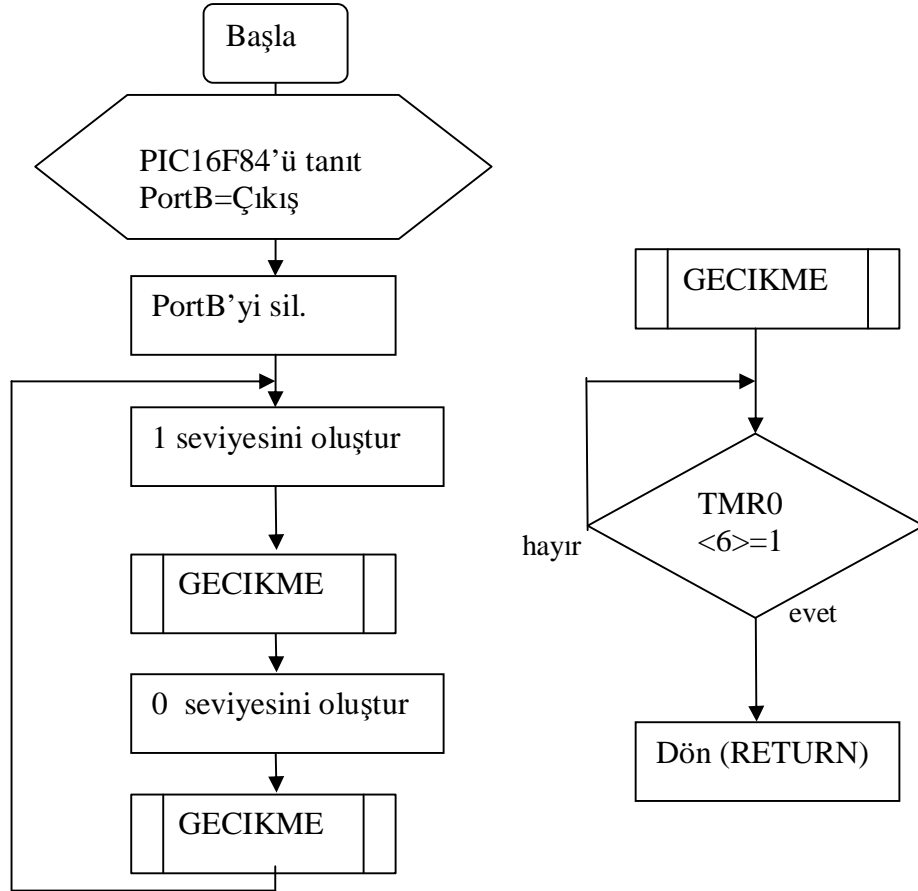
$$\text{TMR0 sayma aralığı zamanı} = 1\mu\text{S} \times (1/256)^{-1} = 256\mu\text{S}$$

TMR0 64'e kadar sayacağından;

$$\text{GECİKME} = 256\mu\text{S} \times 64 = 16384\mu\text{S} = 16.384\text{ ms olur.}$$

Bu durumda PortB<1> pininden çıkacak kare dalga sinyalin frekansı;

$$F = 1/t = 1/16.384\text{ ms} = 61\text{ Hz olur.}$$



Şekil 1.26: Programın akış diyagramı

Şimdi de algoritmayı programa dökelim;

```
LIST P=16F84
INCLUDE "P16F84.INC"
BSF STATUS,5 ;Bank1'e geç.
CLRF TRISB ;B portunun tümü çıkış.
CLRWDT ; WDT ve prescalar'ı sil.
MOVLW b'11010111' ;TMR0'ı, yeni prescalar değerini ve sinyal
; kaynağını seç ve
MOVWF OPTION_REG ;OPTION kayıtçısına yaz.
BCF STATUS,5 ;Bank0'a geç.
CLRF PORTB ;B portunun tüm uçlarını '0' yap.
PALS BSF PORTB,1 ;PortB 1.bit lojik 1 yap.
CALL GECİKME ;GECİKME altprogramını çağır.
BCF PORTB,1 ; PortB 1.bit lojik 0 yap.
CALL GECİKME ;GECİKME altprogramını çağır.
GOTO PALS
GECİKME CLRF TMR0
TEST_BIT BTFSS TMR0,6
GOTO TEST_BIT
RETURN
END
```

#### 1.6.17.5. WDT Zamanlayıcısı

PIC donanımı içerisinde bulunan ikinci bir zamanlayıcıdır. Türkçe karşılığı "bekçi köpeği"dir. Mikrodenetleyici içerisinde bulunan programın bir anlamda bekçiliğini yaptığı için bu ad verilmiştir. Peki bu işi nasıl yapmaktadır? PIC'in önceden belirlenen sürede yapması planlanan bir işi zamanında yapmadıysa, yani döngü kontrolden çıkıp da kilitlendiyse, WDT devreye girer ve PIC'i resetler.

Şekil 1.27'de WDT zamanlayıcısı blok diyagramı görülmektedir. Şimdi de WDT'in birkaç özelliğinden bahsedelim;

- Ø Watchdog Timer, başka hâricî bileşenler talep etmeyen serbest çalışan on-chip RC osilatördür. Bu osilatörün WDT'a sağladığı nominal zaman aşımı süresi 18 ms'dir. Prescaler değeri artırılarak bu değer 2-3 S'ye kadar çıkarılabilir.
- Ø WDT RC osilatörü, OSC1/CLKIN pininin RC osilatöründen ayrılmıştır. Yani, WDT, OSC1/CLKIN ve OSC2/CLKOUT pinlerindeki saat palsı (clock) dursa bile çalışacaktır. Örneğin, SLEEP komutu yürürlüğe konduğu durumlarda buna rastlanır. Normal işletim esnasında bir WDT zaman aşımı (time-out) PIC reset'i yaratacaktır. Eğer PIC SLEEP modunda ise bir WDT cihazın uyanmasını (wake-up) ve normal işletimine devam etmesine sebep olur.

- Ø OPTION kayıtçısının sadece ilk 4 biti (prescaler ve TMR0/WDT seçim bitleri) WDT'la ilgilidir. OPTION kayıtçısının 3.biti 1 yapılarak WDT seçilir ve prescaler bitleri (OPTION<0:2>) ile de frekans bölme değeri ayarlanır.
- Ø WDT'in, CLRWDT komutuyla reset yapıldıktan sonra, saymasını tamamlayıp baştan tekrar saymaya başladığı ana kadar geçen süreye zaman aşımı süresi (time-out) denir. WDT saymasını tamamladığı anda zaman aşımı sinyali verir. Bu sinyal STATUS kayıtçısının 3. ve 4. bitlerindeki (TO ve PD bayrakları) bayrakların durumunu değiştirir. Bu konumda STATUS kayıtçısı konusuna dönüp ilgili bitleri inceleyiniz.
- Ø Eğer WDT ile, kontrolden çıkmış bir programı tekrar resetleyerek, kontrolü ele almayı planladıysanız , ana program başında STATUS kayıtçısının bu bitlerini kontrol etmelisiniz. Daha sonra da reset gerçekleştikten hemen sonra da bu bitleri kontrol etmelisiniz. Böylece WDT'in zaman aşımı süresi dolduğunda program akışını başka bir yere dandırabilirsiniz. Örnek program parçası aşağıdaki gibi olabilir.

```

WDT_KONTROL BTFSS STATUS,TO ;TO bayrağı 1 mi?
 CALL KUR ;hayır (zaman aşımı oluşmuş) , KUR
 ;altprogramını çağır.
 MOVLW h'CC' ;Evet (zaman aşımı oluşmamış)
 . ;programa devam et.
 .
 .
 .

```

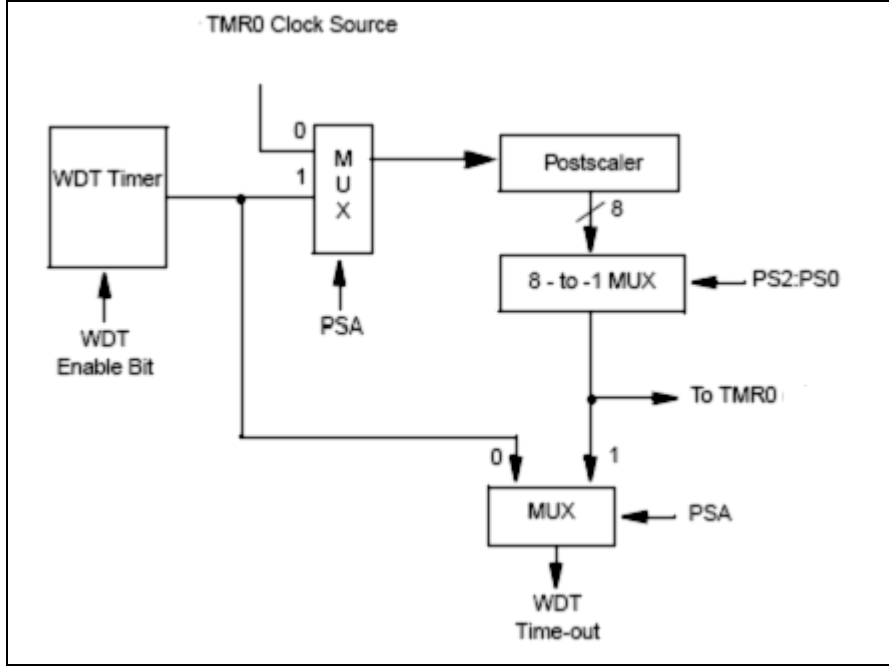
- Ø PIC içerisinde bulunan WDT zamanlayıcısının durumunu konfigürasyon bitlerini yazarken belirleyebiliriz. Bu konfigürasyon bitlerini program içerisinde aşağıdaki gibi yazabiliriz.

```

_CONFIG _CP_OFF & _WDT_ON & _PWRT_ON & _RC_OSC
Bu ifade de ;
_CP_OFF ;kod koruma yok.
_WDT_ON ;WDT aktif
_PWRT_ON ;Power-On-Reset var.
_RC_OSC ;RC osilatör kullanılıyor. (Diğer seçenekler _LP_OSC, _XT_OSC,
_HS_OSC)

```

İstenirse konfigürasyon bitlerini PIC programlayıcının kendi programındaki “fuses” penceresinden de yapabiliriz.WDT’ı programın çalışması esnasında geçersiz yapmak mümkün değildir.



Şekil 1.27: WDT Blok diyagramı

**Not :** T0CS , T0SE , PSA , PS2:PS0 bitleri OPTION registeri içindedir.

### 1.6.18. D/A ve A/D Çevirme

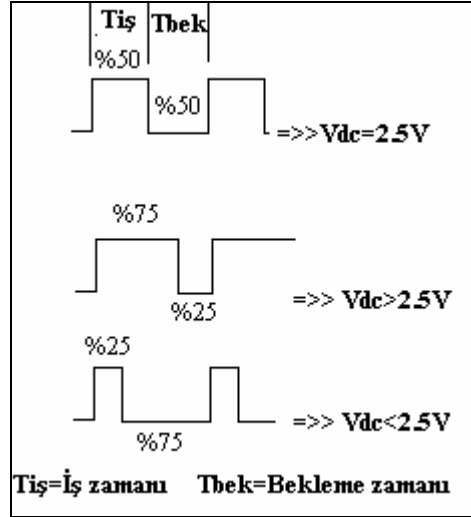
PIC16F84 mikrokontrolü sadece dijital verileri giriş olarak kabul eden ve yine sadece dijital çıkış verebilen bir entegredir. Bu yüzden analog veri girişi ve çıkışı gereken durumlarda bu işler için üretilmiş özel entegreler kullanmak gerekir. Bununla beraber Microchip firmasının katalogları incelendiğinde, PIC14C000 mikrodenetleyicisinin girişine uygulanan dijital sinyali doğrudan analog voltaj çıkışı olarak verebilmektedir. Ayrıca PIC16F877 mikrodenetleyicisi de girişine uygulanan analog sinyali doğrudan dijital çıkış olarak verebilmektedir.

#### 1.6.18.1. Dijital Analog Çevirici

PIC’ten çıkan dijital kelimeyi analog sinyale dönüştürmek için kullanılacak yöntemlerden biri AD558 entegresini kullanmaktır. Çünkü bu entegre 8 bitlik dönüşüm sağlamaktadır ve PIC için kontrol girişlerine sahiptir. Diğer bir yöntem de PWM (Puls genişlik modülasyonu) yöntemini kullanmaktır. Bu yöntemle ayrıca bir entegre kullanmak gerekmediğinden (zaten denetleyicilerin kullanma amaçlarından biri de budur. Biz de bu yöntemi kullanacağız.



Şekil 1.28’de bir kare dalga sinyalin, 3 farklı lojik “1” ve “0” süreleri için , voltmetrede ölçüm değerleri (Analog gerilim değerleri) görülmektedir. Dolayısı ile bu 3 gerilim değerleri ile ledleri sürersek led parlaklıkları birbirinden farklı olacaktır.



Şekil 1.28:PWM sinyali ile analog gerilim üretmek

Eğer iş ve bekleme sürelerini bir periyot içinde değiştirirsek çıkış derilimini de 0-5V arasında değiştirebiliriz. Aşağıdaki gibi bir gecikme alt programı ile iş ve bekleme sürelerini belirleyebiliriz.

GECIKME

MOVWF SAYAC

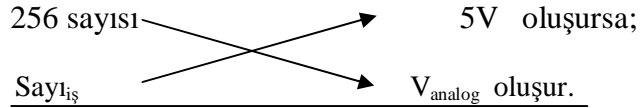
TEKRAR

DECFSZ SAYAC,1

GOTO TEKRAR

RETURN

8 bitlik bir DA çevriminde, olabilecek en büyük dijital değer h'FF' olacağından bu değeri biz bir çevrim periyodunu oluşturan sayı olarak almamız gerekir. Yani PWM çıkış periyodu SAYAC değerinin h'FF'=d'256' olması durumunda oluşur. Bu durumda iş ve bekleme sürelerini,

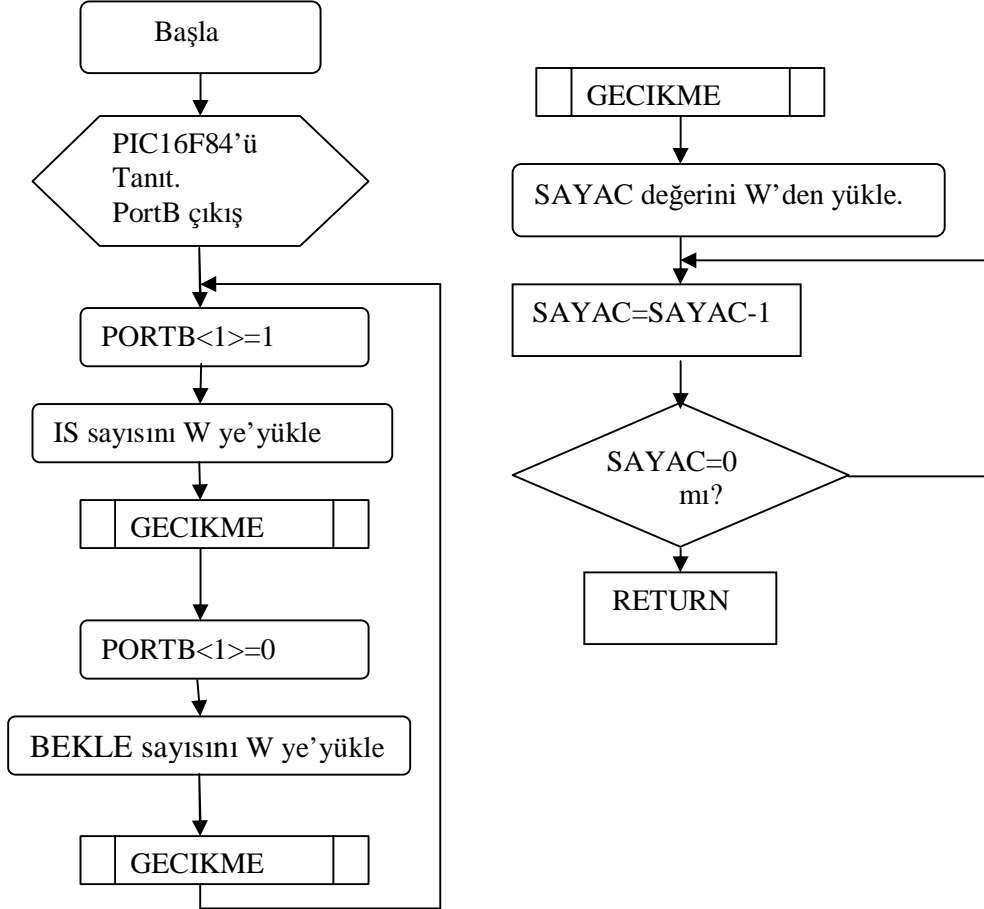


$$Sayı_{i\dot{s}} \times 5V = 256 \times V_{analog}$$

$Sayı_{i\dot{s}} = (256 \times V_{analog}) / 5$  ,  $V_{analog} = (Sayı_{i\dot{s}} \times 5) / 256$  ,  $Sayı_{b\dot{e}k} = 256 - Sayı_{i\dot{s}}$  formülleri bulunur.

Örneğin 0.5V'luk analog gerilim elde etmek için GECIKME altprogramında SAYAC kayıttısına yüklenecek iş ve bekleme sayılarını bulalım.

$Say_{i\dot{s}} = (256 \times V_{analog}) / 5 = (256 \times 0.5) / 5 = 25.6 = 26 = h'1A'$  ve  
 $Say_{bek} = 256 - Say_{i\dot{s}} = 256 - 26 = 230 = h'E6'$  olur. Bu kadar basit.  
 Şimdi de bir programla PWM yöntemi ile analog sinyal üretelim:  
 Örnek: B portunun 1.Bitinden 2V'luk analog değer üreten programı yazınız.  
 Çözüm: Önce iş ve bekleme sayılarını bulalım;  
 $Say_{i\dot{s}} = (256 \times V_{analog}) / 5 = (256 \times 2) / 5 = 102.4 = 103 = h'67'$  ve  
 $Say_{bek} = 256 - Say_{i\dot{s}} = 256 - 103 = 153 = h'99'$  olur.



Şekil 1.29: Örnek programın algoritması

Programda B portunun 1.biti önce 1 yapılıyor. Daha sonra iş zamanını belirleyen değer (h'67') W kayıtçısına yüklenip GECIKME altprogramına dallanılıyor. GECIKME altprogramında h'67' değeri 0 olana kadar B portunun 1.biti 1 olarak kalıyor. SAYAC değeri 0 olduğunda ise anaprograma dönülüyor. Böylece iş zamanı bitmiş oluyor. Anaprogramda B portunun 1.biti bu el 0 yapılıyor. Daha sonra bekleme zamanını belirleyen değer (h'99') W kayıtçısına yüklenip GECIKME altprogramına dallanılıyor. GECIKME altprogramında h'99' değeri 0 olana kadar B portunun 1.biti 0 olarak kalıyor. SAYAC değeri 0 olduğunda ise anaprograma dönülerek tekrar B portunun 1.biti 1 yapılıyor. Böylece bekleme zamanı da bitmiş oluyor. Bu olay silsilesi sonunda B portunun 1.bitine bağlı voltmetrede 2V oluşuyor.

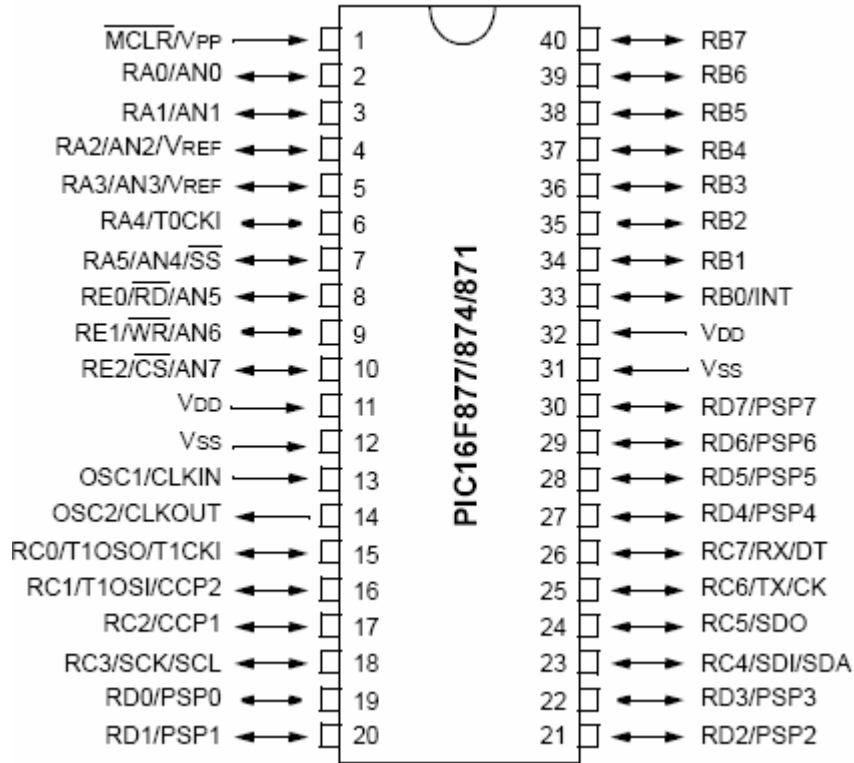
Şimdi de programımızı yazalım.

```
#####DAC.asm#####
LIST P=16F84
INCLUDE "P16F84.INC"
IS EQU h'0C'
BEKLE EQU h'0D'
SAYAC EQU h'0E'
ORG h'00'
BSF STATUS,5
CLRF TRISB
BCF STATUS,5
CLRF PORTB
MOVLW h'67'
MOVWF IS
MOVLW h'99'
MOVWF BEKLE
TEKRAR
BSF PORTB,1
MOVF IS,W
CALL GECIKME
BCF PORTB,0
MOVF BEKLE,W
CALL GECIKME
GOTO TEKRAR
GECIKME
MOVWF SAYAC
DONGU
DECFSZ SAYAC,1
GOTO DONGU
RETURN
```

### 1.6.18.2. Analog Dijital Çevirici

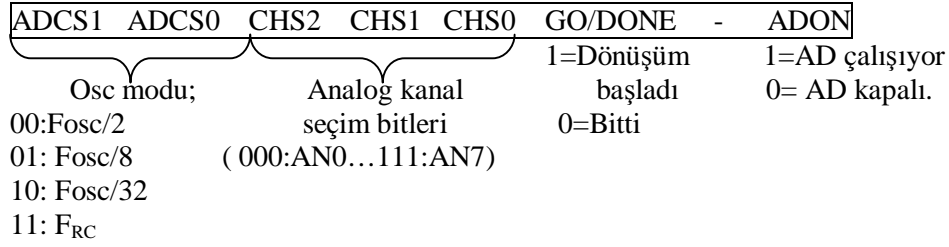
PIC16F84 kontrolcüsünde dâhilî ADC olmadığı için bu işi PIC16FF877 kullanarak gerçekleştireceğiz.

16F877 analog giriş için A portunda 5 , E portunda ise 3 pini kullanır (Şekil 1.30'u inceleyiniz). Yani 8 kanallı ADC olarak kullanılabilir. Bu pinlere bağlanacak sıcaklık, ses, ışık sensörleri veya potansiyometrelerden analog veri alınabilir. Çevrim sonucu 10 bitliktir.



Şekil 1.30: 16F877 denetleyicisi pin diyagramı

ADC uyuma modunda da çalışabilir ve kendi iç devre saatini (RC osilatörünü) kullanır. ADC modülü bu iş için 4 kayıtçı kullanır; ADRESH, ADRESL, ADCON0 ve ADCON1 .ADRESH ve ADRESL kayıtçıları, ADC dönüşüm sonucunun üst ve alt baytlarını tutar. ADCON0 , ADC işleminin kontrolünde ADCON1 ise port pinlerinin konfigürasyonunda kullanılan kayıtçılarıdır. ADCON0 ve ADCON1 kayıtçıların yapıları şekil 1.31 ve şekil 1.32'de verilmiştir.



Şekil 1.31: ADCON1 kayıtçısı



Şekil 1.32: ADCON0 kayıtçısı

| PCFG3:<br>PCFG0 | AN7 <sup>(1)</sup><br>RE2 | AN6 <sup>(1)</sup><br>RE1 | AN5 <sup>(1)</sup><br>RE0 | AN4<br>RA5 | AN3<br>RA3 | AN2<br>RA2 | AN1<br>RA1 | AN0<br>RA0 | VREF+           | VREF-           | CHAN /<br>Refs <sup>(2)</sup> |
|-----------------|---------------------------|---------------------------|---------------------------|------------|------------|------------|------------|------------|-----------------|-----------------|-------------------------------|
| 0000            | A                         | A                         | A                         | A          | A          | A          | A          | A          | V <sub>DD</sub> | V <sub>SS</sub> | 8/0                           |
| 0001            | A                         | A                         | A                         | A          | VREF+      | A          | A          | A          | RA3             | V <sub>SS</sub> | 7/1                           |
| 0010            | D                         | D                         | D                         | A          | A          | A          | A          | A          | V <sub>DD</sub> | V <sub>SS</sub> | 5/0                           |
| 0011            | D                         | D                         | D                         | A          | VREF+      | A          | A          | A          | RA3             | V <sub>SS</sub> | 4/1                           |
| 0100            | D                         | D                         | D                         | D          | A          | D          | A          | A          | V <sub>DD</sub> | V <sub>SS</sub> | 3/0                           |
| 0101            | D                         | D                         | D                         | D          | VREF+      | D          | A          | A          | RA3             | V <sub>SS</sub> | 2/1                           |
| 011x            | D                         | D                         | D                         | D          | D          | D          | D          | D          | V <sub>DD</sub> | V <sub>SS</sub> | 0/0                           |
| 1000            | A                         | A                         | A                         | A          | VREF+      | VREF-      | A          | A          | RA3             | RA2             | 6/2                           |
| 1001            | D                         | D                         | A                         | A          | A          | A          | A          | A          | V <sub>DD</sub> | V <sub>SS</sub> | 6/0                           |
| 1010            | D                         | D                         | A                         | A          | VREF+      | A          | A          | A          | RA3             | V <sub>SS</sub> | 5/1                           |
| 1011            | D                         | D                         | A                         | A          | VREF+      | VREF-      | A          | A          | RA3             | RA2             | 4/2                           |
| 1100            | D                         | D                         | D                         | A          | VREF+      | VREF-      | A          | A          | RA3             | RA2             | 3/2                           |
| 1101            | D                         | D                         | D                         | D          | VREF+      | VREF-      | A          | A          | RA3             | RA2             | 2/2                           |
| 1110            | D                         | D                         | D                         | D          | D          | D          | D          | A          | V <sub>DD</sub> | V <sub>SS</sub> | 1/0                           |
| 1111            | D                         | D                         | D                         | D          | VREF+      | VREF-      | D          | A          | RA3             | RA2             | 1/2                           |

A = Analog input

D = Digital I/O

Şekil 1.33: AD PORT konfigürasyon bitleri

ADC dönüşümü yapmak için şu sıra izlenir:

- Ø AD biriminin konfigürasyonunu yap;
  - Analog giriş portunu belirle (ADCON1).
  - Analog giriş kanalını seç (ADCON0).
  - AD dönüşüm osilatörünü seç (ADCON0).
  - AD birimini aktifleştire.

- Ø PIC16877’de ayrıca AD kesmesi yapılabilir. İstenirse AD kesme kullanılarak kesme altprogramı da yazılabilir. Bu durumda ADIF bitinin silinip ADIE, PEIE, GIE yetkilerinin 1 yapılması gerekir. Bunu yapmak için aşağıdakileri bilmemiz gerekir.
  - ADIE biti 16F877’nin PIE1 kayıtçısında bulunur. PIE1 16F877’nin kesme yetki durumları hakkında içinde kontrol bitleri barındırır. Burada biz sadece ADIE yetki biti ile ilgileniliyoruz.  
ADIE=1 AD kesmesi aktif.  
ADIE=0 AD kesmesi pasif.
  - PEIE ve GIE biti 16F877’nin INTCON kayıtçısında bulunur. INTCON 16F877’nin kesme durumları hakkında içinde kontrol bitleri barındırır. Burada biz sadece PEIE ve GIE biti bayrakları ile ilgileniliyoruz.  
PEIE=1 çevresel kesmeler aktif.  
PEIE=0 çevresel kesmeler pasif.  
GIE=1 tüm kesmeler aktif.  
GIE=0 tüm kesmeler pasif.  
Kullanılmayacaksa bu basamak geçilir.
- Ø Gereken dolum süresini bekle (çevrim zamanı).
- Ø GO/DONE bitini 1 yaparak çevrime başla.
- Ø Çevrimin bitmesini bekle. GO/DONE biti ‘0’sa çevrim bitmiştir. Ya da AD kesmesini bekle.
- Ø AD sonuç yazmaç çiftini oku (ADRESH:ADRESL). ADIF bayrağını 0 yap.  
ADIF biti 16F877’nin PIR1 kayıtçısında bulunur. PIR1 16F877’nin kesme durumları hakkında içinde kontrol bitleri barındırır. Burada biz sadece ADIF bayrağı ile ilgileniliyoruz.  
ADIF=1 ise AD çevrimi tamamlandı.  
ADIF=0 ise AD çevrimi tamamlanmadı.
- Ø Sonraki çevrim için 1 yada 2. basamağa git.

Şimdi de bir örnekle AD dönüşümünü görelim.

**Örnek:** 16F877 denetleyicisinin AN0 kanalına bağlı bir potansiyometrenin oluşturacağı analog sinyal, 8 bit dijital kelimeye dönüştürülerek C portundaki ledlerde gösterilsin.

Bunun çözümü için kayıtçıların hangi banklarda bulunduğunu gösteren 16F877’nin kayıtçı haritasına ihtiyacımız olacaktır (Şekil 1.34). Zaman gecikmesi için TMR0 donanım zamanlayıcısının kullanıldığına dikkat ediniz.

|                                      |     |                                      |     |                                      |      |                                      |      |
|--------------------------------------|-----|--------------------------------------|-----|--------------------------------------|------|--------------------------------------|------|
| Indirect addr. <sup>(1)</sup>        | 00h | Indirect addr. <sup>(1)</sup>        | 80h | Indirect addr. <sup>(1)</sup>        | 100h | Indirect addr. <sup>(1)</sup>        | 180h |
| TMR0                                 | 01h | OPTION_REG                           | 81h | TMR0                                 | 101h | OPTION_REG                           | 181h |
| PCL                                  | 02h | PCL                                  | 82h | PCL                                  | 102h | PCL                                  | 182h |
| STATUS                               | 03h | STATUS                               | 83h | STATUS                               | 103h | STATUS                               | 183h |
| FSR                                  | 04h | FSR                                  | 84h | FSR                                  | 104h | FSR                                  | 184h |
| PORTA                                | 05h | TRISA                                | 85h |                                      | 105h |                                      | 185h |
| PORTB                                | 06h | TRISB                                | 86h | PORTB                                | 106h | TRISB                                | 186h |
| PORTC                                | 07h | TRISC                                | 87h |                                      | 107h |                                      | 187h |
| PORTD <sup>(1)</sup>                 | 08h | TRISD <sup>(1)</sup>                 | 88h |                                      | 108h |                                      | 188h |
| PORTE <sup>(1)</sup>                 | 09h | TRISE <sup>(1)</sup>                 | 89h |                                      | 109h |                                      | 189h |
| PCLATH                               | 0Ah | PCLATH                               | 8Ah | PCLATH                               | 10Ah | PCLATH                               | 18Ah |
| INTCON                               | 0Bh | INTCON                               | 8Bh | INTCON                               | 10Bh | INTCON                               | 18Bh |
| PIR1                                 | 0Ch | PIE1                                 | 8Ch | EEDATA                               | 10Ch | EECON1                               | 18Ch |
| PIR2                                 | 0Dh | PIE2                                 | 8Dh | EEADR                                | 10Dh | EECON2                               | 18Dh |
| TMR1L                                | 0Eh | PCON                                 | 8Eh | EEDATH                               | 10Eh | Reserved <sup>(2)</sup>              | 18Eh |
| TMR1H                                | 0Fh |                                      | 8Fh | EEADRH                               | 10Fh | Reserved <sup>(2)</sup>              | 18Fh |
| T1CON                                | 10h |                                      | 90h |                                      | 110h |                                      | 190h |
| TMR2                                 | 11h | SSPCON2                              | 91h |                                      | 111h |                                      | 191h |
| T2CON                                | 12h | PR2                                  | 92h |                                      | 112h |                                      | 192h |
| SSPBUF                               | 13h | SSPADD                               | 93h |                                      | 113h |                                      | 193h |
| SSPCON                               | 14h | SSPSTAT                              | 94h |                                      | 114h |                                      | 194h |
| CCPR1L                               | 15h |                                      | 95h |                                      | 115h |                                      | 195h |
| CCPR1H                               | 16h |                                      | 96h |                                      | 116h |                                      | 196h |
| CCP1CON                              | 17h |                                      | 97h | General Purpose Register<br>16 Bytes | 117h | General Purpose Register<br>16 Bytes | 197h |
| RCSTA                                | 18h | TXSTA                                | 98h |                                      | 118h |                                      | 198h |
| TXREG                                | 19h | SPBRG                                | 99h |                                      | 119h |                                      | 199h |
| RCREG                                | 1Ah |                                      | 9Ah |                                      | 11Ah |                                      | 19Ah |
| CCPR2L                               | 1Bh |                                      | 9Bh |                                      | 11Bh |                                      | 19Bh |
| CCPR2H                               | 1Ch |                                      | 9Ch |                                      | 11Ch |                                      | 19Ch |
| CCP2CON                              | 1Dh |                                      | 9Dh |                                      | 11Dh |                                      | 19Dh |
| ADRESH                               | 1Eh | ADRESL                               | 9Eh |                                      | 11Eh |                                      | 19Eh |
| ADCON0                               | 1Fh | ADCON1                               | 9Fh |                                      | 11Fh |                                      | 19Fh |
|                                      | 20h |                                      | A0h |                                      | 120h |                                      | 1A0h |
| General Purpose Register<br>96 Bytes |     | General Purpose Register<br>80 Bytes |     | General Purpose Register<br>80 Bytes |      | General Purpose Register<br>80 Bytes |      |
|                                      |     |                                      | EFh |                                      | 16Fh |                                      | 1EFh |
|                                      |     | accesses<br>70h-7Fh                  | F0h | accesses<br>70h-7Fh                  | 170h | accesses<br>70h - 7Fh                | 1F0h |
|                                      |     |                                      | FFh |                                      | 17Fh |                                      | 1FFh |
| Bank 0                               |     | Bank 1                               |     | Bank 2                               |      | Bank 3                               |      |

Şekil 1.34: 16F877'nin kayıtçı haritası

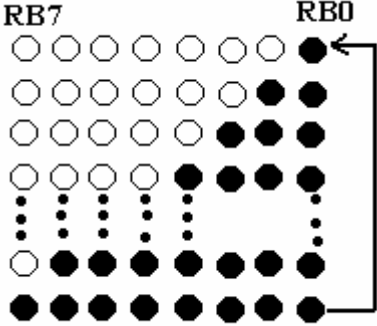
```

;*****
;* ADC.ASM
;*****
;* Microchip Technology Incorporated
;* 16 December 1998
;* Assembled with MPASM V2.20
;*****
;* Bu program A/D
;* A/D dönüşümü için kanal 0 seçilmiş, analog sinyal bir
;* potansiyometre üzerinden sağlanmakta ve sonuç
;* PORTC'deki ledlerde gösterilmektedir.
;*****
 list p=16f877
 include "p16f877.inc"
 org 0x000
 nop
 clrf PORTC ;Clear PORTC
 movlw B'01000001' ;Fosc/8, A/D etkin olacak şekilde
 movwf ADCON0 ; ADCON0'ı ayarla.
 bsf status,5 ; Bank1'e geç.
 movlw b'10000111' ; TMR0 etkin, prescaler 1:256 olacak
 movwf OPTION_REG ;şekilde OPTION_REG'i ayarla.
 clrf TRISC ;PORTC çıkış.
 movlw B'00001110' ;sola yanaşık çıkış formatı, 1 analog kanal
 movwf ADCON1 ; (AN0)girişi ve VDD ve VSS referans
 ; olacak şekilde ADCON1'i ayarla.
 bcf status,5 ;Bank0'a geç.
BASLA
 btfss INTCON,T0IF ; Timer0 taşması gerçekleşti mi?
 goto BASLA ;hayır.Taşmayı bekle.
 bcf INTCON,T0IF ;evet.T0IF bayrağını sil.
 bsf ADCON0,GO ; A/D çevrimini başlat.
BEKLE
 btfss PIR1,ADIF ;çevrim bitti mi?
 goto BEKLE ;hayır.Bitirmesini bekle.
 movf ADRESH,W ;Evet sonucu PORTC'deki
 movwf PORTC ;ledlere gönder.
 goto BASLA ;tekrarla.
 End

```



## UYGULAMA FAALİYETİ

| İşlem Basamakları                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Öneriler                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Ø B Portuna bağlı 8 led , A portunun 0.bitine bağlı bir butona basıldığında birbirine eklenerek aşağıdaki gibi 1 sn. arayla sola doğru kayarak yansın.</p>  <p>Programın;</p> <ul style="list-style-type: none"> <li>Ø Program akış diyagramını çiziniz.</li> <li>Ø Programlama dilini seçiniz.</li> <li>Ø Programlama dilinin yazım kurallarına dikkat ederek assembler komutlara göre programınızı yazınız.</li> <li>Ø Programı MPASM ile derleyiniz.</li> <li>Ø Derleme sonucunda oluşan HEX dosyasını , programlama kartı ile mikrodenetleyiciye yazınız.</li> </ul> | <ul style="list-style-type: none"> <li>Ø Akış diyagram sembollerini amacına uygun kullanınız.</li> <li>Ø Programın başlık kısmında gerekli tanımlamaları yapınız ( LIST, INCLUDE).</li> <li>Ø Programda kullandığınız mikrodenetleyici komutlarının, sayı ve karakterlerin yazılışına dikkat ediniz. Programın tamamını ya büyük ya da küçük harfle yazmak yazım hatalarını azaltacaktır.</li> <li>Ø Program başında giriş/çıkış portlarını kurduktan sonra B portunu ve C (elde) bayrağını silerek programa başlayınız.</li> <li>Ø Programın derlenmesi sonucunda oluşan HEX dosyası ASM dosyanızın bulunduğu dizinedir (MPASM programı dizininde değildir).</li> </ul> |

## ÖLÇME VE DEĞERLENDİRME

### OBJEKTİF TESTLER (ÖLÇME SORULARI)

1. LIST ve INCLUDE ifadeleri bir assabmler programın hangi bölümünde bulunur?  
A) Başlık                      B) Atama                      C) Program                      D) Sonlandırma
2. A Portu yönlendirme kayıtcısı aşağıdakilerden hangisidir?  
A) PORTA                      B) TRISA                      C) STATUS                      D) INTCON
3. Kesme bit ve bayraklarını içinde barındıran kayıtcı aşağıdakilerden hangisidir?  
A) OPTION\_REG              B) INTCON                      C) TMR0                      D) STATUS
4. Bank deęiřtirme iřlemini hangi kayıtcı saęlar?  
A) PORTB                      B) OPTION\_REG              C) STATUS                      D) ADCON0

**Ařaęıdaki program parçasına göre 5-15. sorulardaki boşlukları tamamlayınız.**

;=====örnek.asm=====

```
LIST P=16F84
INCLUDE "P16F84.INC"
ORG h'000'
GOTO BASLA
BASLA BSF STATUS,5
 CLRF TRISB
 MOVLW h'FF'
 MOVWF TRISA
 BCF STATUS,5
TEKRAR CLRF PORTB
 BCF STATUS,C
KONTROL BTFSS PORTA,0
 GOTO KONTROL
 MOVLW h'35'
 MOVWF TEST
 INCF TEST,1
 RRF TEST,1
 COMF TEST,1
 RLF TEST,0
 MOVWF PORTB
 GOTO TEKRAR
END
```

5. Programın başlangıç adresi h'.....'dır.
6. Programda .....mikrodenetleyicisi kullanılmıştır.
7. Programda .. ....portu giriş,..... portu çıkış olarak kurulmuştur.
8. A Portunun .....bitine bağlı bir buton ile programın akışı kontrol edilmektedir.
9. A Portuna bağlı buton basılmadığı sürece PORB kayıtçısının değeri h'.....'dir.
10. Butona basıldıktan sonra BCF STATUS, C komut satırında C (elde) bayrağının değeri olur.
11. Butona basıldıktan sonra, program sonunda W kayıtçısının değeri h'.....'dir.
12. Butona basıldıktan sonra, program sonunda TEST kayıtçısının değeri h'.....'dir.
13. Butona basıldıktan sonra, program sonunda PORTB kayıtçısının değeri h'.....'dir.
14. Butona basıldıktan sonra program sonunda C (elde) bayrağının değeri .....dir.
15. GOTO komutuna PC'ye(program sayıcıya) .....etiketinin adresi yüklenir.

## DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları faaliyete dönerek tekrar inceleyiniz.

Tüm sorulara doğru cevap verdiyseniz diğer faaliyete geçiniz.

## ÖĞRENME FAALİYETİ-2

### AMAÇ

Uygun ortam sağlandığında microdenetleyici programını derlemesini yapabilecek, derleme sonunda oluşan dosyaları tanıyabilecek ve oluşan dosyalardan faydalanarak programınız hakkında yorum yapabileceksiniz.

### ARAŞTIRMA

Bu faaliyet öncesinde yapmanız gereken öncelikli araştırmalar şunlar olmalıdır:

- Ø MPASM programının menülerini tanıyıp ayarlarının nasıl yapılması gerektiğini öğrenmelisiniz.
- Ø Yazdığımız programı derleme olayına neden ihtiyaç duyduğumuzu araştırmalısınız.

MPASM programının menülerini tanıyıp ayarlarının nasıl yapılması gerektiğini öğrenme faaliyetleri-1'den öğrenebilirsiniz. Ayrıca internet ortamından ve mesleki kitapların bulunduğu kütüphaneler ile bu işi yapan sektörde çalışanlardan daha ayrıntılı bilgi bulabilirsiniz. Araştırma için arkadaşlarınızla iş bölümü yapabilirsiniz.

## 2. MİKRODENETLEYİCİ KONTROL PROGRAMININ MAKİNE DİLİNE ÇEVİRİLMESİ

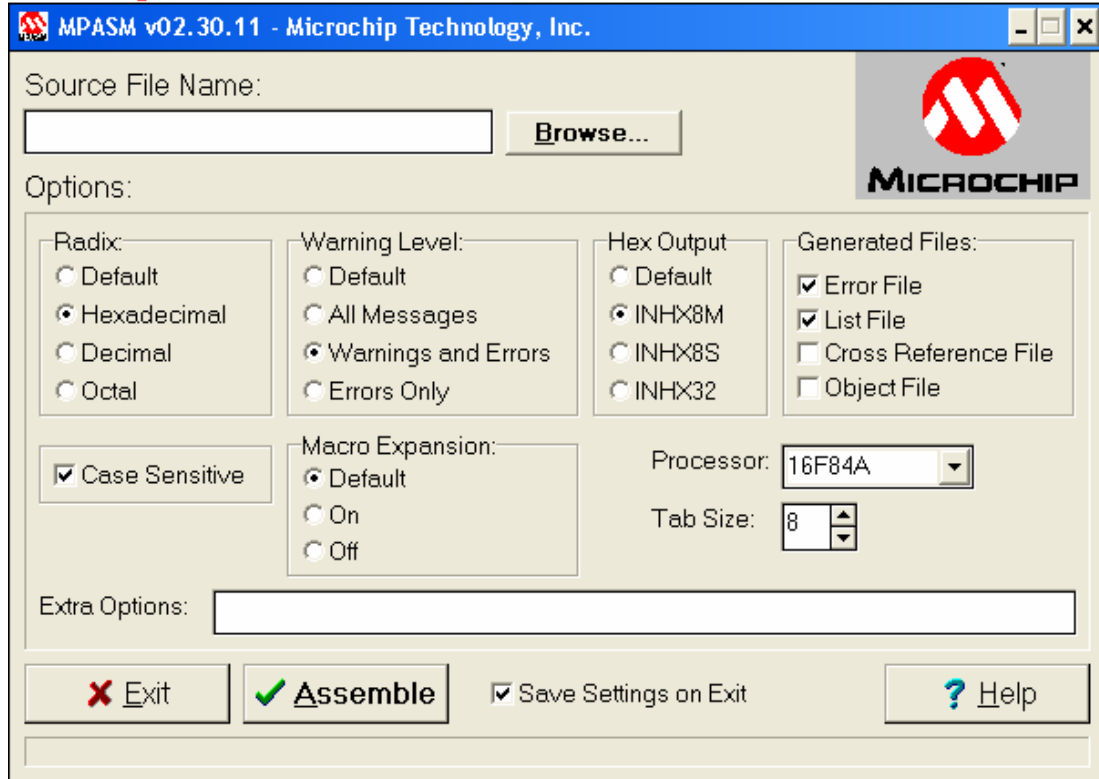
Bildiğimiz gibi MPASM derleyici programı assembly programımızın makine kodunu üretiyordu . MPASM derleyici programının nasıl kurulacağını, ayarlarının nasıl yapılacağını “Mikrodenetleyici için gerekli yazılımın kullanımı” konusunda anlatmıştık. Burada ise derleme işleminin yapılması ve derleme sonucunda oluşturulan dosyaları göreceğiz.

## 2.1. Programın Derlenmesi

### 2.1.1. Derleme İşleminin Yapılması

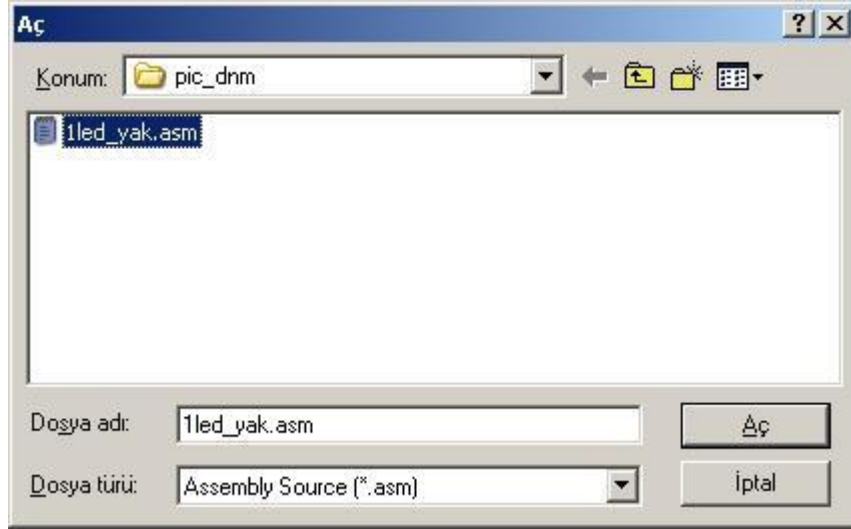
Bir assembly dosyasını derlemek için aşağıdaki sıra izlenir.

- Ø MPASM programı çalıştırılarak resim 2.1'deki gibi ayarlanır. Burada mikrodenetleyici olarak 16F84A seçilmiştir.



Resim 2.1: MPASM derleyici programı arabirimi

- Ø Daha önceden yazılıp kaydedilmiş .asm uzantılı kaynak dosyası “Browse” ile açılır (Resim 2.2).

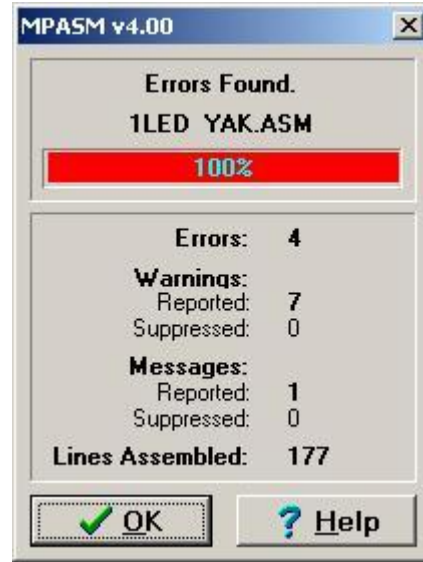


Resim 2.2: ASM dosyanın seçilmesi.

- Ø “Assemble” komut tuşuna basılarak derleme işlemi bitirilir. Sonuç, derleme başarılı olduysa resim 2.3’teki, başarılı olmadıysa resim 2.4’teki pencere ile programcıya bildirilir.



Resim 2.3: Başarılı ASM derleme



Resim 2.4: Başarısız ASM derleme

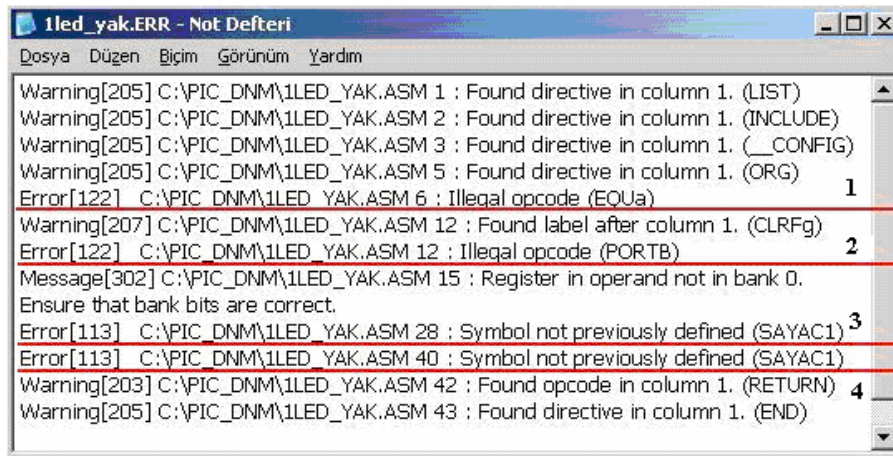
## 2.1.2. Derleme Sonucu Elde Edilen Dosyalar

Derleme sonunda LST dosyası, ERR dosyası, HEX dosyası, WAT dosyası, PJT dosyası, LST dosyası dosyaları MPASM programının bulunduğu dizinde oluşur. Şimdi de bu dosyaların ne anlama geldiğini görelim.

### 2.1.2.1. ERR Dosyası

ERR dosyası metin editörünü kullanarak açabileceğiniz bir dosyadır. ASM dosyası konusunda örnek olarak verilen tersle.asm dosyası derlendikten sonra hiç hata oluşmadığı için tersle.err dosyası açılırsa boş bir sayfa görülecektir. Fakat hata çıkması durumunda MPASM bunu programcıya şekil 2.4'teki pencere ile bildirecektir. Bu pencerede hata , uyarı ve mesaj sayılarına ait bilgiler bulunur. Errors:4 ifadesi 4 satırda hata bulunduğunu ifade etmektedir. Bu hatalar bir ERR dosyası olarak ASM dosyasının bulunduğu dizinde oluşturulur.

Şimdide 1LEDYAK.ERR dosyamızın hatalarını bulup düzelteyim. ERR uzantılı dosyayı bir metin editörü ile açıp içinde hataların olduğu satırları arıyoruz. Hataları bulduktan sonrada tekrar ASM dosyamızı açıp ilgili hataları birer birer gidereceğiz. Eğer programınızda kod satırları çoksa biraz zor olabilir, ama gene de çözüm var; dosyada metin arama özelliğini kullanarak "Error" kelimesini arayarak bulabilirsiniz. Şimdi dosyayı açalım ve hataları bulalım.



Şekil 2.5: Örnek hata dosyası

- Ø 1.satırda bir hata var, hata olan kod **EQUa** =>hâlbuki bu değişken tanımlama için kullandığımız equ olmalıydı. Gidip onu EQU yaparak düzeltiyoruz.
- Ø 2.satırda bir hata var. **Illegal opcode (PORTB)**=> Yalnız bu PORTB'nin yazılışında bir hata yok; burada başka bir hata olmalı gidip 12 inci satıra bakalım. Görüyoruz ki 12. satır CLRF PORTB olması gerekirken, CLRFg PORTB olarak girilmiş, burada ERR dosyasında CLRFg ifadesini warning olarak bir üst satırda görüyoruz.CLRF yazdığımızda bu assembler kodu olur,

ama CLRFg yazdığımızda bu ifade satırın başında olduğu için bunu ETIKET olarak tanımaya çalışır. Bu yüzden bu satırda PORTB'de hata gösterdi.

Ø 28 ve 40. satırlarda iki hata var; fakat esas hata burda değil. Error dosyasında açıklamaya bakarsak "**Symbol not previously defined (SAYAC1)**" yazıyor. Burada MPASM diyor ki: "Sayac1 değişkenini daha önce tanımlamadınız." O zaman öğreteceğiz demektir, gidip asm dosyamızda tanımlamaları yaptığımız bölüme bakıyoruz ve görüyoruz ki aslında bu hatayı az önce düzeltmişiz. Çünkü az önce 6.satırda bulunan hata , SAYAC1 değişkenini tanımlarken bulduğumuz hataydı ve düzeltmiştik.

Ø **Not: Bu hataya düşmeyiniz.** Bir hata başka bir hatanın sebebi olabilir. Yani derleme sonucunda eğer 20 hata uyarısı aldıysanız derleme yapılırken 20 defa sorunla karşılaşmış anlamına gelir, yoksa 20 tane hata var anlamına gelmez (ama olabilir de tabii). Hatta yukarıdaki örneğe benzer şekilde bir hata olup çok hata uyarısı alabilirsiniz. Örneğin bir değişkenin tanımlanmasında hata varsa ve bu değişkeni biz 20 yerde kullandıysak, hatamız bir yerde olmasına rağmen 20 tane hata uyarısı alırız. Üstteki hatayı düzelttiğinizde unutup 2 dk. sonra aşağılarda bulduğunuz hatadan dolayı tekrar aynı hatayı ararsanız bulamazsınız, en güzeli birkaç düzeltme yaptığınızda bir karmaşa olursa MPASM'la tekrar derleme yapınız ve ERROR dosyasını tekrar açınız. Bu durumda en son hataları görmüş olacağınızdan sorun yaşamazsınız, uzun uzun hata aramazsınız.



### 2.1.2.2. ASM Dosyası

ASM dosyası aslında derleme sonunda oluşan bir dosya değildir. Bizim program kaynak dosyamızdır. Fakat derleme sonunda oluşan kaynak dosyalarla aynı dizinde bulunduğu için listeye dahil edildi.

“Bit pozisyonlarını tersleme” konusundaki tersle.asm programını aşağıda görüyorsunuz.

```
=====tersle.asm=====
 LIST P=16F84
INCLUDE "P16F84.INC"
SAYAC1 EQU h'0C'
SAYAC2 EQU h'0D'
CLRF PORTB
BSF STATUS,5
CLRF TRISB
BCF STATUS,5
MOVLW h'0F' ;ilk değeri yükle ve
MOVWF PORTB ;B portundan çıkar.
TERSLE
CALL GECIKME ;Yeni değer için bekle.
COMF PORTB,F ;PORTB'yi tersle.
GOTO TERSLE ; sağa kaydır.
GECIKME ;GECIKME alt programı başlangıcı.
MOVLW h'FF'
MOVWF SAYAC1
DON1
MOVLW h'FF'
MOVWF SAYAC2
DON2
DECFSZ SAYAC2,F
GOTO DON2
DECFSZ SAYAC1,F
GOTO DON1
RETURN ;GECIKME alt programı sonu
END
```

### 2.1.2.3. HEX Dosyası

HEX dosyası metin editörünü kullanarak açabileceğiniz bir dosyadır. ASM dosyası konusunda örnek olarak verilen tersle.asm dosyası derlendikten sonra oluşan tersle.hex dosyası aşağıda görülmektedir. Denetleyiciye bu dosyadaki verileri yükleriz.

```
:1000000086018316860183120F3086000920860937
:100010000628FF308C00FF308D008D0B0D288C0BD7
:040020000B280800A1
:00000001FF
```

### 2.1.2.4. WAT Dosyası

WAT dosyası MPASM programının derlemesi sonucu oluşan bir dosya değildir. Bu yüzden bunun üzerinde durmayacağız.

### 2.1.2.5. PJT Dosyası

PJT dosyası metin editörünü kullanarak açamayacağımız bir dosyadır. Bu dosya proje dosyasıdır ve hangi program kullanarak proje oluşturulmuş ise o program tarafından açılır. Örneğin MPLAB kullanarak oluşturacağımız projeler gibi... Programcı için bilgilendirici bir dosya değildir sadece var olan çalışmayı açmak için kullanılır.

### 2.1.2.6. LST Dosyası

LST dosyası metin editörünü kullanarak açabileceğiniz bir dosyadır. Bu dosya birkaç sayfadan oluşabilir. ASM dosyası konusunda örnek olarak verilen tersle.asm dosyası derlendikten sonra oluşan tersle.lst dosyası aşağıda görülmektedir. Dikkatli incelendiği zaman aşağıdaki bilgileri içerdiği görülmektedir.

- Ø Bir lst dosyası birkaç sayfadan oluşur. Bu örnekte PAGE1, PAGE2 ve PAGE3 şeklinde 3 sayfa var.
- Ø Komutların hex kodları (PAGE1 içerisinde)
- Ø Komutların bellekteki adresleri ( PAGE1 içerisinde LOC OBJECT CODE ),
- Ø Kaynak program ve satır numaraları (PAGE1 içerisinde LINE SOURCE TEXT),
- Ø Programda kullanılan etiketler ve tanımlar (PAGE2 içerisinde SYMBOL TABLE)
- Ø OPTION\_REG tanımlamaları (PAGE3 içerisinde)
- Ø Bellek kullanım haritası (PAGE3 içerisinde, x =kullanılan, '-' kullanılmayan alanları belirtir)
- Ø Bellekte durumu (PAGE3 içerisinde kullanılan 18 , kullanılmayan 1006 bayt, toplam 1KB)

Ø Oluşan hata ve uyarı sayısı (PAGE3 sonunda hata sayısı 0, uyarı sayısı 7, uyarılar ise PAGE1 içerisinde bulunuyor. )

MPASM 02.30.11 Intermediate tersle.ASM 8-6-2005 19:02:47 PAGE 1

```
LOC OBJECT CODE LINE SOURCE TEXT
VALUE
00001 ;=====TERSLE.asm=====
00002 LIST P=16F84
Warning[205]: Found directive in column 1. (INCLUDE)
00003 INCLUDE "P16F84.INC"
00001 LIST
00002 ; P16F84.INC Standard Header File, Version 2.00 Microchip Technology, Inc.
00136 LIST
0000000C 00004 SAYAC1 EQU h'0C'
0000000D 00005 SAYAC2 EQU h'0D'
0000 0186 00006 CLRF PORTB
0001 1683 00007 BSF STATUS,5
0002 0186 00008 CLRF TRISB
0003 1283 00009 BCF STATUS,5
0004 300F 00010 MOVLW h'0F' ;ilk değeri yükle ve
Warning[203]: Found opcode in column 1. (MOVWF)
0005 0086 00011 MOVWF PORTB ;B portundan çıkar.
0006 00012 TERSLE
0006 2009 00013 CALL GECIKME ;Yeni değer için bekle.
Warning[203]: Found opcode in column 1. (COMF)
0007 0986 00014 COMF PORTB,F ;PORTB'yi tersle.
0008 2806 00015 GOTO TERSLE ; sağa kaydır.
00016
0009 00017 GECIKME
Warning[203]: Found opcode in column 1. (MOVLW)
0009 30FF 00018 MOVLW h'FF'
Warning[203]: Found opcode in column 1. (MOVWF)
000A 008C 00019 MOVWF SAYAC1
000B 00020 DON1
Warning[203]: Found opcode in column 1. (MOVLW)
000B 30FF 00021 MOVLW h'FF'
Warning[203]: Found opcode in column 1. (MOVWF)
000C 008D 00022 MOVWF SAYAC2
000D 00023 DON2
000D 0B8D 00024 DECFSZ SAYAC2,F
000E 280D 00025 GOTO DON2
000F 0B8C 00026 DECFSZ SAYAC1,F
0010 280B 00027 GOTO DON1
0011 0008 00028 RETURN
00029 END
```

| LABEL      | VALUE    |
|------------|----------|
| C          | 00000000 |
| DC         | 00000001 |
| DON1       | 0000000B |
| DON2       | 0000000D |
| EEADR      | 00000009 |
| EECON1     | 00000088 |
| EECON2     | 00000089 |
| EEDATA     | 00000008 |
| EEIE       | 00000006 |
| EEIF       | 00000004 |
| F          | 00000001 |
| FSR        | 00000004 |
| GECIKME    | 00000009 |
| GIE        | 00000007 |
| INDF       | 00000000 |
| INTCON     | 0000000B |
| INTE       | 00000004 |
| INTEDG     | 00000006 |
| INTF       | 00000001 |
| IRP        | 00000007 |
| NOT_PD     | 00000003 |
| NOT_RBPU   | 00000007 |
| NOT_TO     | 00000004 |
| OPTION_REG | 00000081 |
| PCL        | 00000002 |
| PCLATH     | 0000000A |
| PORTA      | 00000005 |
| PORTB      | 00000006 |
| PS0        | 00000000 |
| PS1        | 00000001 |
| PS2        | 00000002 |
| PSA        | 00000003 |
| RBIE       | 00000003 |
| RBIF       | 00000000 |
| RD         | 00000000 |
| RP0        | 00000005 |
| RP1        | 00000006 |
| SAYAC1     | 0000000C |
| SAYAC2     | 0000000D |
| STATUS     | 00000003 |
| T0CS       | 00000005 |
| T0IE       | 00000005 |
| T0IF       | 00000002 |

```

TOSE 00000004
TERSLE 00000006
TMR0 00000001
TRISA 00000085
TRISB 00000086
W 00000000
WR 00000001
WREN 00000002
WRERR 00000003
Z 00000002

```

MPASM 02.30.11 Intermediate ABD.ASM 8-6-2005 19:02:47 PAGE 3

SYMBOL TABLE

| LABEL      | VALUE    |
|------------|----------|
| _CP_OFF    | 00003FFF |
| _CP_ON     | 0000000F |
| _HS_OSC    | 00003FFE |
| _LP_OSC    | 00003FFC |
| _PWRTE_OFF | 00003FFF |
| _PWRTE_ON  | 00003FF7 |
| _RC_OSC    | 00003FFF |
| _WDT_OFF   | 00003FFB |
| _WDT_ON    | 00003FFF |
| _XT_OSC    | 00003FFD |
| __16F84    | 00000001 |

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

0000 : XXXXXXXXXXXXXXXXXXXX XX-----

All other memory blocks unused.

Program Memory Words Used: 18

Program Memory Words Free: 1006

Errors : 0

Warnings : 7 reported, 0 suppressed Messages : 0 reported, 1 suppressed

## UYGULAMA FAALİYETİ

Yaptığınız ve yapacağınız bir mikrodenetleyici programını makine diline çevirerek ASM, HEX, LST, COD, ERROR dosyalarını inceleyiniz.

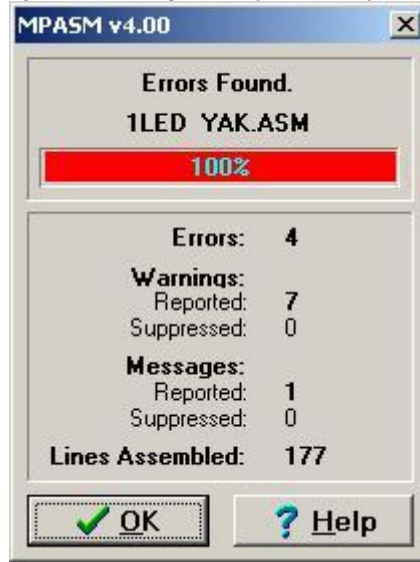
| İşlem Basamakları                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Öneriler                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>Ø Mikrodenetleyici derleme programını (MPASM) bilgisayarınıza kurunuz.</li><li>Ø MPASM program ayarlarını yapınız.</li><li>Ø “Uygulama faaliyetleri-1” Uygulamalar bölümünde yazmış olduğunuz programı derleme programında açınız.</li><li>Ø Komut dilinde yazılmış programınızı derleyerek makine diline heksadesimal kodlara çeviriniz (HEX dosyası).</li><li>Ø Eğer derleme sonucunda hatalar oluştuysa, hata dosyasını açıp hataları gideriniz.</li><li>Ø LST dosyasını açıp denetleyicinin bellek durumunu (boş ve dolu bellek alanları) öğreniniz.</li></ul> | <ul style="list-style-type: none"><li>Ø Derleme yapmadan önce mutlaka MPASM programında kullanacağınız denetleyici tipi, çıkış formatı ayarlarını yapınız.</li><li>Ø Derleme sonucunda hata oluştuysa, bütün hataları birden bulmak yerine, tek tek bulmaya çalışınız ve bir hatayı bulduktan sonra, o hatayı giderip programı tekrar derleyiniz. Çünkü bir hata başka hataların nedeni olabilir.</li></ul> |

## ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları cevaplayarak faaliyette kazandığınız bilgi ve becerileri ölçünüz.

### OBJEKTİF TESTLER (ÖLÇME SORULARI)

1. Bir assembly program MPASM ile derlendikten sonra aşağıdaki dosyalardan hangisi **oluşmaz**?  
A) ERR dosyası      B) LST dosyası      C) HEX dosyası      D) PPT dosyası
2. MPASM ile derleme sonucu oluşan hata dosyası aşağıdakilerden hangisidir?  
A) LST      B) ERR      C) PJT      D) ASM
3. MPASM ile derleme sonucu oluşan dosyalar hangi dizinde bulunur?  
A) Belgelerim      B) ASM dosyasının bulunduğu dizin      C) MPASM programı dizininde      D) C (sabit disk)
4. Aşağıdaki hata mesajı ekranına göre kaç assembly satırında hata var?



- A) 4      B) 177      C) 1      D) 7
5. Aşağıdaki dosyalardan hangisini bir metin editörü ile açıp programımız hakkında bilgi **alamayız**?  
A) ERR dosyası      B) LST dosyası      C) ASM dosyası      D) PJT dosyası
- 10 nu.lu soruları doğru/yanlış olarak cevaplandırınız.**

6. ASM dosyası MPASM programının derleme sonucunda oluşturduğu bir dosya **değildir.**
7. Denetleyiciye yükleyeceğimiz veriler ASM dosyası içindedir.
8. LST dosyasından denetleyici belleğini ne kadar kulladığımızı öğrenebiliriz.
9. “ Error [113] C:\prog\test.asm 20 :illegal opcode (PORTA)” hata satırı bize test.asm programının 20. satırında bir komut yazılım hatası olduğunu söyler.
10. Hatasız program mutlaka beklediğimiz gibi çalışır.

## **DEĞERLENDİRME**

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları faaliyete dönerek tekrar inceleyiniz.

Tüm sorulara doğru cevap verdiyseniz diğer faaliyete geçiniz.



# MODÜL DEĞERLENDİRME

## PERFORMANS TESTİ (YETERLİK ÖLÇME)

Mikrodenetleyici programlama modülü, faaliyetleri ve araştırma çalışmaları sonunda kazandığınız bilgi ve becerilerin ölçülmesi ve değerlendirilmesi için kendinizi değerlendirme ölçeğine göre değerlendiriniz. Bu değerlendirme sonucuna göre bir sonraki modüle geçebilirsiniz.

| DEĞERLENDİRME ÖLÇÜTLERİ                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Evet | Hayır |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-------|
| Akış diyagramı sembollerini bir problemin çözümünde kullanabilir misiniz?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |      |       |
| Mikrodenetleyici assembler programı ve yazım kurallarına uygun program yazabilir misiniz?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |      |       |
| Mikrodenetleyici komutlarının işlevlerini biliyor musunuz?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |       |
| Programda sayı ve karakterleri kullanmak;<br>Ø Heksadesimal sayıları kullanabilir misiniz?<br>Ø Binary sayılar kullanabilir misiniz?<br>Ø Desimal sayılar kullanabilir misiniz?<br>Ø ASCII karakterler kullanabilir misiniz?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |      |       |
| Mikrodenetleyici için gerekli yazılımın kullanmak;<br>Ø MPASM derleyici programını kurabilir misiniz?<br>Ø MPASM programının kullanacağınız denetleyiciye göre ayarlarını yapabilir misiniz?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |      |       |
| Mikrodenetleyici programını yazmak;<br>Ø Portların giriş ve çıkış olarak yönlendirebilir misiniz?<br>Ø Konfigürasyon bitlerinin yazabilir misiniz?<br>Ø W kayıtçısının kullanabilir misiniz?<br>Ø Bitleri test ederek işlem yapabilir misiniz?<br>Ø Status kayıtçısını kullanabilir misiniz?<br>Ø Zaman geciktirme döngüleri düzenleyebilir misiniz?<br>Ø Bit pozisyonlarını sola/sağa kaydırma ,tersleme işlemlerini yapabildiniz mi?<br>Ø İstenen bitleri sıfırlamak, bire çevirmek ve terslemek işlemlerini yapabilir misiniz?<br>Ø Bir bytlık iki veriyi birbiriyle veya 0 ile karşılaştırabilir misiniz?<br>Ø 8 ve 16 bitlik toplama yapabilir misiniz?<br>Ø 8 ve 16 bitlik çıkarma yapabilir misiniz?<br>Ø Çevrim tablolarını kullanabilir misiniz?<br>Ø INTCON kayıtçısını kullanabilir misiniz?<br>Ø Kesme kaynaklarını biliyor musunuz?<br>Ø Kesme alt programlarını düzenleyebildiniz mi?<br>Ø TMR0 ve WDT sayıcılarını kullanabilir misiniz? |      |       |

|                                                                                                                |  |  |
|----------------------------------------------------------------------------------------------------------------|--|--|
| <input type="checkbox"/> Option kayıtçısını kullanabilir misiniz?                                              |  |  |
| <input type="checkbox"/> Mikrodenetleyici ile dijital /analog çevirmesini yapabilir misiniz?                   |  |  |
| <input type="checkbox"/> Mikrodenetleyici ile analog /dijital / çevirmesini yapabilir misiniz?                 |  |  |
| Mikrodenetleyici kontrol programını makine diline çevirmek;                                                    |  |  |
| <input type="checkbox"/> Assembly programını derleyebilir misiniz?                                             |  |  |
| <input type="checkbox"/> Derleme sonucunda oluşan dosyalardan faydalanarak çıkan hataları giderebilir misiniz? |  |  |

## DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

Modülü tamamladınız, tebrik ederiz. Öğretmeniniz size çeşitli ölçme araçları uygulayacaktır. Öğretmeninizle iletişime geçiniz.

# CEVAP ANAHTARLARI

## ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

|    |        |
|----|--------|
| 1  | A      |
| 2  | B      |
| 3  | B      |
| 4  | C      |
| 5  | 000    |
| 6  | P16F84 |
| 7  | A,B    |
| 8  | 0      |
| 9  | 0.     |
| 10 | 00     |
| 11 | C8     |
| 12 | E4     |
| 13 | C8     |
| 14 | 1      |
| 15 | TEKRAR |

## ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

|    |   |
|----|---|
| 1  | D |
| 2  | B |
| 3  | B |
| 4  | A |
| 5  | D |
| 6  | Y |
| 7  | Y |
| 8  | D |
| 9  | D |
| 10 | Y |

Cevaplarınızı cevap anahtarları ile karşılaştırarak kendinizi değerlendiriniz.

## ÖNERİLEN KAYNAKLAR

- Ø ALTINBAŞAK Orhan, **Mikrodenetleyiciler ve PIC Programlama**, İstanbul, 2000.
- Ø <http://www.antrak.org.tr>
- Ø <http://www.Microchip.com>

## KAYNAKÇA

- Ø ALTINBAŞAK Orhan, **Mikrodenetleyiciler ve PIC Programlama**, İstanbul, 2000.
- Ø BODUR Yaşar, **Adım Adım PICmicro Programlama**, İstanbul.
- Ø Microchip Technology Inc., **PIC16F84 Data Sheet 18-Pin Enhanced Flash/Eprom 8-bit Microcontrollers**, 2001.
- Ø GÜNEŞ Abdullah, **Mikrodenetleyici Ders Notları**, Bursa, 2004.
- Ø <http://www.antrak.org.tr>
- Ø <http://www.ramay.net>
- Ø <http://www.Microchip.com>