3.2 PIC16F84 Yazılımı

3.2.1 PIC Assembly

3.2.1.1 Assembler Nedir?

Assembler,bir text editöründe assembly dili kurallarına göre yazılmış olan komutları pıc'in anlayabileceği heksadesimal kodlara çeviren (derleyen) bir programdır.Microchip firmasının hazırladığı MPASM bu işi yapan assembler programıdır.Assembler'e çoğu zaman compiler (derleyici) denilir.

3.2.1.2 PIC Assembly Dili Nedir?

Assembly dili,bir PIC'e yaptırılması istenen işlerin belirli kurallara göre yazılmış komutlar dizisidir.Assembly dili komutları İngilizce dilindeki bazı kısaltmalardan meydana gelir.Bu kısaltmalar genellikle bir komutun çalışmasını ifade eden cümlenin baş harflerinden oluşur.Böylece elde edilen komut,bellekte tutulması kolay (mnemonic) bir hale getirilmiştir.Örneğin; BTFSC (Bit Test F Skip if Clear) –File registerdeki bit'i test et,eğer sıfırsa bir sonraki komutu atla,anlamında kullanılan İngilizce cümlenin kısaltmasıdır.

3.2.1.3 PIC Assembly Dili Yazım Kuralları

PIC assembly programlarının yazılması için kullanılan text editörlerinden daha önce bahsedilmişti.Bu editörler arasında Windows altında çalışan NOTPAD veya DOS altında çalışan EDIT en uygunlarıdır.Bunların dışında printer kontrol komutları içermeyen ve ASCII kodunda dosya üretebilen herhangi bir editör de kullanılabilir.MPLAB kullanıldığında ayrıca bir editör kullanılmasına gerek yoktur.Çünkü MPLAB'ın içinde hem bir text editörü hem de MPASM bulunmaktadır.

MPASM assembler programının,yazılan komutları doğru olarak algılayıp,PIC'in anlayabileceği heksadesimal kodlara dönüştürebilmesi için şu bilgiler program içinde özel formatta yazılması gerekir;

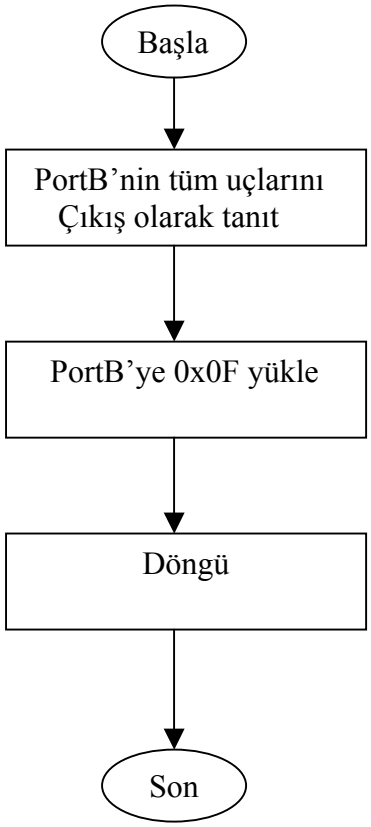
*Komutların hangi PIC16XX için yazıldığı,

*Programın bellekteki hangi adresten başlayacağı,

*Komutların ve etiketlerin neler olduğu,

*Programın bitiş yeri.

Basit bir örnekle bu bilgilerin program içinde nasıl yazıldığını gösterirsek;
Program ilk olarak PIC16F84'e B portunun 8 ucunu da çıkış olarak tanıtacak.Daha sonra bu porttaki ilk dört bitini lojik 1,sonraki dört bitini de lojik 0 yapacak.Son olarak program sonsuz bir döngüye girecektir.Bu işlemleri yapacak olan programın akış diyagramı ve komutları aşağıdaki gibi olacaktır.




```
;==============PICTEST1.ASM============================
                        LIST P=16F84
;-------------------------------------------------------------------------------
;       Adres tanımlama bloğu
STATUS      EQU     0x03
PORTB       EQU     0x06
TRISB       EQU     0x86
;-------------------------------------------------------------------------------
                ORG     0x00                ; programı 0x00 'dan başlat
```

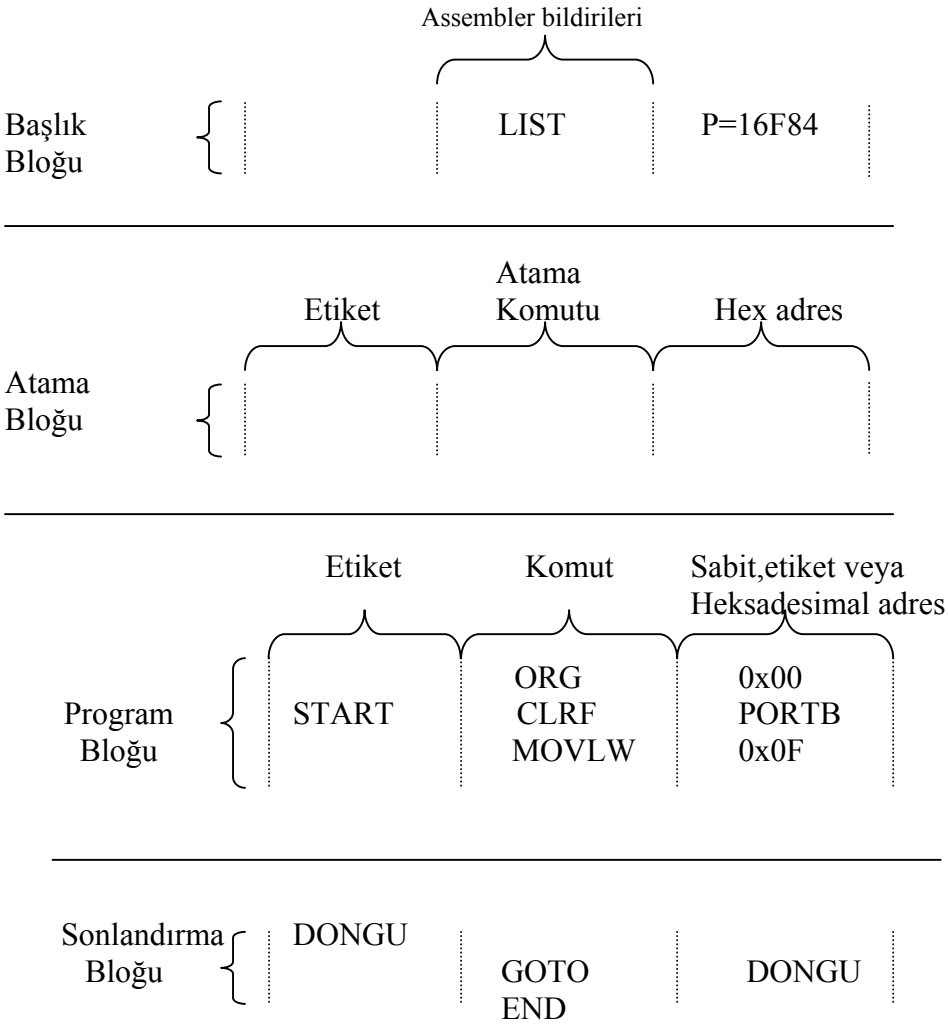
```
;-------------------------------------------------------------------------------
;Portların durumunu belirleme bloğu
START
        CLRF    PORTB           ; portB'nin içini sıfırla
        BSF     STATUS,5        ; BANK1 'e geç
        CLRF    TRISB           ;portB'nin uçlarını output yap
        BCF     STATUS,5        ;tekrar BANK0 'geç
;-------------------------------------------------------------------------------
;Program bloğu
        MOVLW   0x0F            ;W registerine 0x0F'i yükle
        MOVWF   PORTB           ;W'yi portB'ye yükle
;-------------------------------------------------------------------------------
;Sonlandırma bloğu
DONGU
        GOTO    DONGU
        END
;===============================================================
```

3.2.1.3.1 Noktalı Virgül (;)

Baş tarafına (;) konulan satır,assembler tarafından heksadesimal kodlara dönüştürülmez.Bu satırlar programın geliştirilmesi esnasında hatırlatıcı açıklamaların yazılmasında kullanılır.Örneğin CLRF ile başlayan satırda "portB'nin içini sıfırla" cümlesi,CLRF komutunun ne iş yapacağını açıklar.Programın bölümlerini birbirinden ayırmak için (-------- veya =======) çizgileri kullanmak,programı görsel olarak daha okunur hale getirdiği gibi bu çizgiler arasına uyarılar ve açıklamalar da yazılabilir.

3.2.1.3.2 Girintiler ve Program Bölümleri

Text 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.



Assembler,yukarıda belirtildiği gibi komutların üç kolona bölünerek yazılmış olduğunu varsayar.Belirtilen kolona yazılmayan bir komut olduğunda ise bunu da kabul eder.Ancak,heksadesimal kodlara dönüştürme (compile) esnasında bu tip hataları bir uyarı (Warning) olarak belirtir.Assembly komutları yazılırken kolonlar arasında verilen TAB uzunluğu önemli değildir.SPACE tuşu ile verilen aralık da assembler tarafından TAB olarak algılanır.

Yukarıda açıklamarla verilen program açıklamalar kaldırarak tekrar yazılırsa;

```
            LIST        P=16F84
STATUS      EQU         0x03
```

```
PORTB       EQU     0x06
TRISB       EQU     0x86
            ORG     0x00
START
            CLRF    PORTB
            BSF     STATUS,5
            CLRF    TRISB
            BCF     STATUS,5
            MOVLW   0x0F
            MOVWF   PORTB
DONGU
            GOTO    DONGU
            END
```

3.2.1.3.3 Başlık

Programın en başındaki bilgilere başlık bölümü denilir.

```
;============PICTEST1.ASM===================
            LIST    P=16F84
;-----------------------------------------------------------------------
```

Başlık bölümünde program dosyasının adı ve hazırladığı tarih ,istenirse hazırlayanın adı da yazılabilir.İlk satır,bir açıklama satırıdır ve assembler tarafından derlenmez
LIST P=16F84 satırı,programın hangi PIC için yazıldığını belirtir.LIST bir compiler bildirisidir.Yani compiler'i yönlendiren bir komuttur ve tek kullanılış amacı ve yeri burasıdır. Başlık bölümünde ayrıca verdiğimiz örnekte kullanılmayan INCLUDE komutu da kullanılabilir.INCLUDE komutu adresleri sabit olan STATUS:PORTA:PORTB,TRISA,TRISB gibi özel registerlerin "atamalar"bloğunda adreslerini her defasında belirtme zorunluluğunu ortadan kaldırmak için kullanılan bir compiler bildirisidir.

3.2.1.3.4 Etiketler

PIC belleğindeki bir adresin atandığı,hatırlamayı kolaylaştıran kısaltmalardan meydana gelen sembolik isimlere etiket denilir.Örneğin PORTB etiketi,PIC16F84'ün file register belleğindeki B portunun bulunduğu adresi temsil eden etikettir.Etiketler program içerisinde 1. kolona yazılır.

EQU eşitleme ifadesidir.Programın herhangi bir yerinde PORTB etiketi kullanıldığında,B portunun adresi olan 0x06 yazılmış gibi işlem görür.

Birinci kolona yazılan ve adres atanmayan etiketler de kullanılabilir.START ve DONGU bu tip etiketlerdir.B u etiketler program akışını istenilen bir yere dallanmasını sağlamak amacıyla kullanılır.Bu etiketleri adresleri bir özel register adresi gibi fiziksel bir adres değildir.Bu şekilde tanımlanan bir etikete assembler otomatik olarak bir adres atar.Bu adresi bizim bilmemiz gerekmez.

Etiket tanımlanırken uyulması gereken kurallar şunlardır:

*Etiketler 1. kolona yazılmalıdır.

*Etiketler bir harfle veya alt çizgi(_) ile başlamalıdır.

*Etiketler içerisinde Türkçe karakterler kullanılamaz.

*Etiketler bir assembly komutundan oluşamaz.

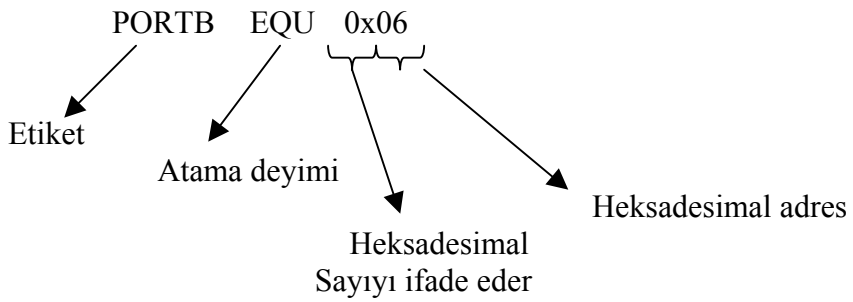
*Etiketlerin içerisinde alt çizgi,rakam,soru işareti bulunabilir.

*Etiketler en fazla 31 karakter uzunluğunda olabilir.

*Etiketlerde büyük/küçük harf duyarlılığı vardır.

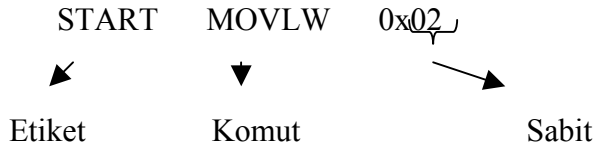
3.2.1.3.5 Atama deyimi (EQU)

EQU deyimi PIC16F84'ün belleğindeki bir heksadesimal adresi ,belirlenen bir etikete atamak için kullanılır.



3.2.1.3.6 Sabitler

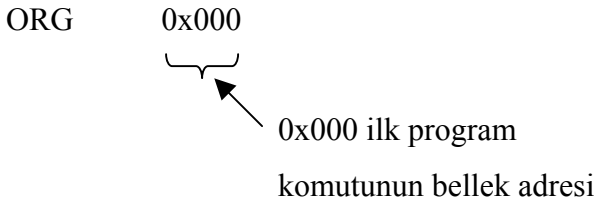
PIC assembly dilinde heksadesimal sayılar birer sabittir.Sabitler MOVLW ve bazı mantıksal ve aritmetik işlem komutlarında kullanılırlar.



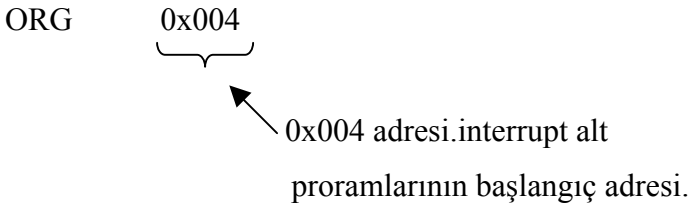
3.2.1.3.7 ORG Deyimi

ORG İngilizce'deki "origin" kelimesinden gelmektedir.ORG deyimi iki amaç için kullanılır.

*Program komutlarının hangi adresten itibaren başladığını gösterir.



*PIC16F84 Interrupt alt programının başlangıç adresini belirlemede kullanılır.



3.2.1.3.8 Sonlandırma Bloğu

PIC16F84'ün duraklama (halt) komutu yoktur.Programı belirli bir yerde duraklatmak için bazen sonsuz döngü kullanılır.

```
DONGU
        GOTO        DONGU
        END
```

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

END deyimi ise program komutlarının sona erdiğini assembler'e bildirir.Her program sonunda END deyimi kesinlikle kullanılmalıdır.Aksi halde program derlenirken dosya sonunun belirtilmediğini belirten bir hata mesajı verecektir.

3.2.1.3.9 Büyük ve Küçük Harflerin Kullanımı

PIC assembler komutlarının büyük veya küçük harfle yazılması önemli değildir.İstenirse büyük/küçük harf karışımı komutlarda kullanılabilir.Ancak etiketler büyük/küçük harfe duyarlıdır.

3.2.1.4 PIC Assembly Komutlarının Yazılış Biçimi

PIC16F84'ün toplam 35 tane komutu vardır.Bu komutların yazılış biçimi üç grupta toplanılabilir.

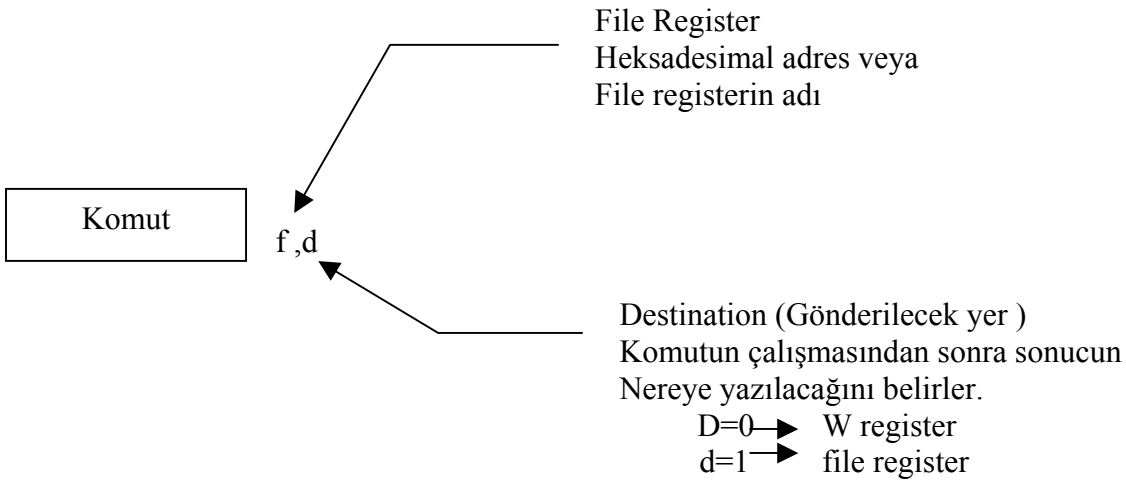
1.Byte-yönlendirmeli komutlar.
2.Bit yönlendirmeli komutlar.
3.Sabit işleyen komutlar.
4.Kontrol komutları

Komutların yazılış biçimlerini açıklarken bazı tanımlama harfleri kullanılacaktır.

F=File register
d=destination (gönderilen yer)
d=0 → W register
d=1 → file register
k=Sabit veya adres etiketi
b=Bit tanımlayıcı
b=Binary sayıları belirleyen harf (b '00001111')

d=Desimal sayıları belirleyen harf (d ‘16’)

3.2.1.4.1 Byte-Yönlendirmeli Komutlar



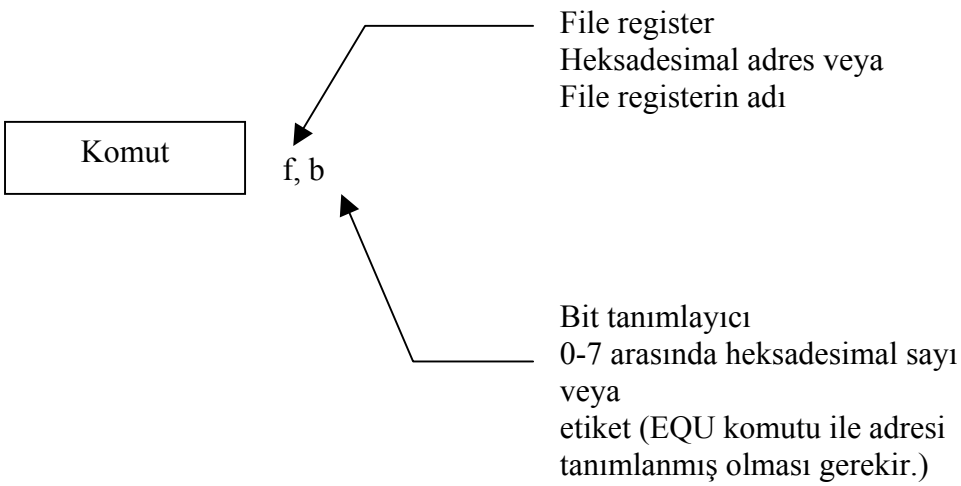

```
MOVF    0x03,0      ;0x03 adresindeki file registerin içeriğini W
                    Registeri içerisine kopyalanır.
MOVF    STATUS.0    ;STATUS registerin içeriği W registere kopyalanır.
MOVF    STATUS,1    ;STATUS registerinin içeriği yine kendi içine yazılır.
```

NOT: Byte-yönlendirmeli komutlarda destination (gönderilecek yer) belirleyen d’nin yazıldığı yere 0 veya 1 yazmak hatırlatıcı olmayabilir.MPASM bunu dikkate alarak 0 yerine W, 1 yerine f yazmaya izin verir.MPASM’nin MS-DOS versiyonunda ise W ve f harflerinin otomatik olarak kullanılmasına izin verilmez.Bu durumda her programın tanımlama bölümünde aşağıdaki eşitlikler yazılmalıdır.

```
W       EQU     0
F       EQU     1
```

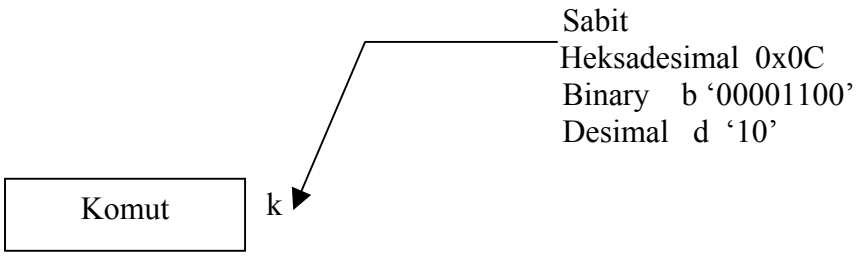
Bu eşitliklerden sonra komutlarda destination belirlemek için W ve f harfleri kullanılabilir.

3.2.1.4.2 Bit-Yönlendirmeli Komutlar



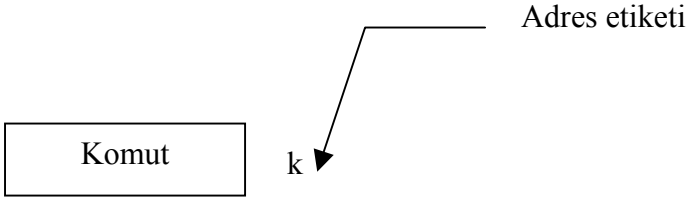

```
BCF     0x03,5          ;0x03 adresindeki registerin 5. bitini sıfırla
BSF     STATUS,BESBIT   ;STATUS registerinin BESBIT etiketiyle
                        Tanımlı olan bitini '1' yapar. (Tanımlama
                        Bloğunda BESBIT EQU 5 yazılması gerekir.)
```

3.2.1.4.2 Sabit İşleyen Komutlar




```
MOWLW   0x02            ;W registerine 2F heksadesimal
                        Sayısını yükler.
ADDLW   b '00101111'    ;W registeri içerisindeki sayıya
                         00101111 binary sayısını ekler.
```

3.2.1.4.3 Kontrol Komutları



```
GOTO    DONGU       ;Program akışı DONGU olarak belirlenen
                    Etikete dallanır.
CALL    TIMER       ;Program akışı TIMER etiketi ile belirlenen
                    Adresteki alt programa dallanır.
```

NOT:Program içerisinde yazılan etiketlere assembler otomatik olarak adres verir.

3.2.1.5 Sayı ve Karakterlerin Yazılış Biçimi

PIC assembly komutlarında sayılar heksadesimal ,binary veya desimal formda kullanılabilir.

3.2.1.5.1 Heksadesimal sayılar

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

```
STATUS      EQU     0x03
            EQU     3
            EQU     03
            EQU     03h
            EQU     h '03'
```

MOVLW komutu ile W registeri içerisine yüklenecek olan FF heksadesimal sabitler ise aşağıdaki gibi yazılabilir.

```
MOVLW       0xFF
            h 'FF'
```

3.2.1.5.2 Binary sayılar

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

```
MOVLW          b ‘00001010’
```

3.2.1.5.3 Desimal sayılar

Desimal sayıların başına d harfi konularak tırnak içerisinde yazılırlar.Örneğin 15 desimal sayısını W registeri içerisine yüklemek için aşağıdaki gibi yazılmalıdır.

```
MOVLW          d ‘15’
```

3.2.1.5.4 ASCII karakterler

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

```
RETLW          ‘A’
RETLW          ‘T’
```