3.BÖLÜM

3.ASSEMBLER-DİLİ YAPISI

3.1.Giriş: Her assembler dili yapısı bir kaynak program içinde birbirini izleyen **[label:](etiket),Mnemonic,[Operand],[;comment] (yorum)** isimli 4 alandan oluşur.

Bunların içinde yalnızca mnemonic alanı daima gereklidir. Etiket ve yorum alanları daima seçimliktir. Operand alanı operand gereken yapılarda uygulanır. Aksi halde atlamalısınız. (Biz etiket, operand ve yorum alanlarının seçimlik olarak belirtmeniz için boşluklarla göstereceğiz.)

Bu alanları çizgi üzerinde herhangi bir yerde girebilirsiniz. Ancak en az bir boşluk (ya da tab) ile ayırmalısınız. Bir assembler-dili yapısında bütün dört alan böyle kullanılır.

```
GETCOUNT:   MOV   CX, DI ; sayacı başlatır
```

3.2.Etiket Alanı:

Etiket alanı assembler-dili yapısında bir isimle var olur. Bu yapı program içindeki diğer yapılarak kaynak teşkil eder. Böylece etiketler assembler-dili programı içinde aynı amaçlı ve çizgi numarası belirli temel programlara sunumu gerçekleştirir.

Bir etiket yapısı en fazla 31 karakterden uzunluğunda olabilir ve iki nokta ile sonlanmalıdır.

Böyle oluşabilir:

- A'dan Z'ye harfler (veya a'dan z'ye, assembler'de küçük harf büyük harf ayrımı yoktur)
- 0'dan 9'arasında sayısal değerler.
- Özel karakterler : ? . @ _ $

Herhangi bir karakterle etikete başlayabilirsiniz. Ancak siz bir periyod kullandıysanız (.) ilk karakteri olmalı.

AH, AL, AX, BH, BL, BP, CH, CL, CX, CS, DH, DL, DX, DI, DS, ES, SI, SP ve ST saklayıcı isimleridir. Bunları kullanamazsınız.

Etiket içinde boşluk kullanamazsınız. Fakat aynı etiketi alt çizgi karakteri ile (_) sağlayabilirsiniz. Örneğin; önceki örnek yapı gibi yazabilirsiniz.

```
GET_COUNT :     MOV     CX, DI  ; sayacı başlatır.
```

Açıkça, GET_COUNT, GETCOUNT'ta olabilir.

3.2.1.Etiket İsmi Seçme:

Çünkü assembler harflerin, basamakların, simgelerin değişken durumlarını girmenize izin verir. Hemen hemen bütün etiketler kabul edilebilir olarak düşünebilirsiniz. Zaten biz etiket seçme için yorumlar öneriyoruz.

- Kısa ve olası isim yapma, mantıklı olabilmeli. Böylece, MPH – MILES_PER_HOUR için tercih edilebilir. CUR_YR, CURRENT_YEAR için mantıklı bir kısaltmalıdır.
- Hatasız tipte isim yapma. Her zamanki tipleme sorunu birkaç özdeş harfin bir sırada (HHHH gibi) ve benzer karakterin (O harfi ile "0" sıfır, I harfiyle 1 sayısı, S harfi ile 5 sayısı gibi) Burada tanımlama hatası çağırmasının hiçbir mantığı yoktur, hepimiz için bunları başka yolla yapmak yeterlidir.
- Etiketleri karıştırabileceğiniz diğerleri ile birlikte kullanmayın. Örneğin, bunun gibi şeyleri kullanmaktan kaçının XXXX ve XXYX

3.3.Mnemonic Alanı:

Mnemonic alan (mnemonic 'in başı "m" harfi sessiz) 2'den 7'ye kadarki harfler yapı için kısaltmadır. Örneğin: MOV move yapısı için kısaltmadır. ADD add yapısı için kısaltmadır. Assembler her sayısal eşitliklerin içindeki mnemonic yapıyı çevirirken içsel tablolar kullanır.

Mnemonic'e ek, bazı yapılar gerektiriyor ki bir ya da iki operand tanımlanması (Örneğin : bir ADD yapısı eklemek için iki terim bilmek gerektiriyor.) Mnemonic assembler'a kaç tane operand, hangi tip, operand alanında ele geçirmeyi söyler.

3.4.Operand Alanı:

Operand alanı 80286'ya işlenen veriyi nerede bulabileceğini söyler. Bu bazı yapılar için gereklidir ve diğerleriyle engellenmiştir. Eğer hazırsa, operand alanı 8 tane 1 veya 2 operand içerir. Mnemonic'ten en az bir boşluk ya da tablo ayırmıştır. Eğer iki operand gerekliyse, aralarına bir virgül koymalısınız.

İki operandlı yapılarda, ilki hedef operand ve ikincisi kaynak operandtır. Kaynak operand mikroişlemcinin hedef operandın içine ekleyeceği, çıkartacağı, karşılaştıracağı veya depolayacağı değeri belirtir. Örneğin : Bu move talimatında

```
MOV     CX, DX
```

CX, DX deki : Kaynak operantının içindeki Dx register'ının içeriğini hedef operandının içindeki CX register'ına taşır.

Belki de tahmin ediyorsunuzdur. Hedef operand hemen hemen her zaman değiştirilirken; kaynak operand asla işlemle değiştirilmedi.

3.5.Yorum Alanı:

BASIC'teki REM'e benzer bir ifade, kaynak programın içinde seçimlik tanımlama ifadesidir; programı daha kolay anlaşılır yapar. Yorumdan önce noktalı virgül (;) gelmelidir. Bu alandan önce en az bir boşluk ya da tabla ayırmak iyi fikir, ancak bunu yapamazsınız. Assembler yorumları yok sayar., ancak programı listeleyeceğiniz zaman yazar.

Yorum alanına istediğiniz herhangi bir şeyi yazabilirsiniz., ama yararlı alabilecek, programda ne olduğunu açıklayan yorumlar yapmalısınız. talimatı yeniden vermeyin

Örneğin:

1) MOV CX, 0 ; sayaç tutucusunu temizler

2) MOV CX, 0 ; CX'in içine 0'ı taşır.

1. açıklama ikinci açıklamadan daha anlamlıdır.

3.5.1.Yalnız Yorumlar:

Tanım bloğu içinde talimatın kendisi tarafından bir yorum tanımlayabilirsiniz. Bunu yapmak için çizginin başına bir noktalı virgül girin; assembler yorum çizgisini tanımlayacaktır ve isleyen her şeyi yok sayar.