

8086 REGISTERLARI ve SEGMENTLERİ:

Mikro işlemci, register adı verilen 14 adet özel lokasyon içerir. Herbiri 16 bit genişliğinde olan bu lokasyonları özel bellek birimleri olarak düşünebiliriz. Registerlar, mikro işlemci tarafından özel amaçlar için kullanılır. Ayrıca, çeşitli verilerin geçici süreler için saklanması ve bu veriler üzerinde işlem yapılması için registerlardan yararlanır. Registerlar CPU chip'inin üzerinde yer aldıklarından, registerla yapılan işlemler, bellek bölgeleri üzerinde yapılan işlemlere nazaran çok daha hızlıdır.

8086/8088 mikro işlemcisinin sahip olduğu registerları aşağıdaki gibi guruplara ayırmamız mümkün;

1.Genel amaçlı Registerlar

- a) Data Registerları (4 adet)
- b) Pointer ve Index Registerları (4 adet)

2.Instruction Pointer Registeri(1 adet)

3.Program Status Word Registeri (1 adet)

4.Segment Registerları (4 adet)

Yukarıda gösterilen 14 adet registerın 4 tanesi **CODE, DATA, STACK VE EXTRA** segmentlerin başlangıç adreslerini saklayan CS,DS,SS ve ES registerlarıdır.Bu dört registra genel olarak segment registerları denilmektedir. Geriye kalanlardan 8 tanesi genel amaçlı register, 1 tanesi instruction pointer (komut göstergesi) registeri ve sonuncusu da program status word (durum göstergesi) registeri olarak bilinmektedir. Program status word registerına flag registeri adını vereceğiz. Genel amaçlı registerlar, kendi aralarında iki guruba ayrılırlar. Her iki gurup, 4 adet registerdan oluşur. Bunlardan ilk dört registra data, diğer 4 adet registra ise pointer ve index registerları adı verilir.

1.Genel amaçlı registerlar:

a) Data Registerları :

Daha çok, aritmetiksel işlemler için kullanılan DATA registerleri AX,BX,CX ve DX'dir. Herbir DATA registeri diğer registerlerden farklı olarak, bir bütün halinde veya iki parça halinde kullanılabilir. Bir DATA registerini iki alt alana bölebilir, böylece registeri 16 bitlik birtek register halinde veya 8 bitlik iki ayrı register halinde kullanma imkanına sahip olabiliriz. Bu özellik şekil üzerinde gösterilmiştir. Örneğin AX registerini 16 bitlik işlemler için AX'i oluşturan AH ve AL yi ise 8 bitlik işlemler için ele alırız. 16 bit uzunluğundaki data registerlerinden herhangi birisinin yüksek seviyeli kısmı (high order) register isminin ve 'High' kelimesinin ilk harfi ile , alçak seviyeli kısmı (Low order) ise ,yine register isminin ve 'Low' kelimesinin ilk harfi ile gösterilir. Böylece AX için AH ve AL, BX için BH ve BL ,CX için CH ve CL, DX için DH ve DL olarak isimlendirilen 8-bitlik registerlar elde edilir. Bu 8 bit registerlerden birinin değerinin değiştirilmesi ile, kendisini ihtiva eden 16 bit registerinin değerinin de değişeceği ve bunun tersinin de mümkün olacağı açıktır. Fakat bir 16-bit registerin herhangi bir yarımı üzerinde yapılan işlem diğer yarımı üzerinde değişikliğe yol açmayacaktır. Data registerlerinin dışında kalan registerlarının dışında kalan registerların, bu şekilde alt alanlara bölünemeyeceği şekil üzerinde de gösterilmiştir. Ancak flag registerleri için daha da farklı bir kullanım söz konusudur.

Akümülatör olarak isimlendirilebilen AX registeri ; çarpa, bölme, giriş, çıkış, bazı string operasyonlarında word uzunluklu olarak kullanılır. AL registeri byte uzunluklu olup, bazı operasyonların sayaç registeri olarak veya desimal amaçlı aritmetiksel işlemlerde transfer alanı (alıcı alan registeri) olarak kullanılır. Ah registeri byte uzunluklu bölme ve çarpmalarda kullanılır. Base registerolarak isimlendirilebilen BX , daha çok bellek içindeki verilerin adreslanmesinde kullanılır. CX sayaç registeridir, LOOP komutlarının veya tekrarlı! ; string operasyonlarının sayaç registeri olarak kullanılır. CL registeri shift ve rotate sayacı olarak kullanılabilir. Data registeri ismi verilen DX , word uzunluklu çarpma ve bölme operasyonlarında kullanılır.

b) Pointer ve Index Registerları :

Yukardaki şekil üzerinde de gösterildiği gibi; pointer ve indeks registerları SP, BP, SI ve DI 'dir. Ancak, IP ve BX registerlerini de birer pointer ya da indeks registerı olarak kabul etmek mümkündür. Pointer ve indeks registerleri de, data registerleri gibi aritmetiksel işlemler için kullanılabilirler. Fakat bu registerlerin kullanımındaki asıl amaç çeşitli bellek bölgelerindeki verilere ulaşabilmek için adres göstericisi olarak ele alınmalarıdır. Bellek içindeki bir lokasyonun adresini gösterebilmek için, iki ayrı adres değeri kullanılır. İlk değer, işaret edilen lokasyonu içinde barındıran segmentin adresidir . bu noktada segment registerlerinden yararlanılır. Lokasyonun segment içindeki yerini gösteren ikinci adres değeri ise offset adresi olarak bilinir. Offset adreslerinin gösterilmesi için de , pointer ve indeks registerlarından yararlanılır.

Mikroişlemci, code segment içindeki lokasyonlara erişebilmek için CS ve IP registerlarını kullanır. Stack segment içinde bulunan bir veriye ulaşmak için SS registerı ile birlikte SP ya da BP registerlarından birini kullanmak gerekir. Data segment içindeki lokasyonlara erişmek için, data segmentin başlangıcı olan DS registerından, lokasyonun offset adresini göstermesi için de BX, SI veya DI registerlarından yararlanılır. Ekstra segment ya da gelişigüzel seçilen bir segment içerisindeki lokasyonlara erişebilmek için CS, DS, SS'ten yararlanabiliriz.

2. Instruction Pointer Registeri :

Komut göstergesi olarak isimlendirebileceğimiz IP registeri, 8088'in icra ünitesi tarafından icra ettirilecek olan bir sonraki komutun offset adresini içerir. Offset adersi IP ile gösterilen bellek bölgesi code segmenti içindedir. Çünkü programları oluşturan komutlar bu segment içinde bulunur. IP registeri özel amaçlı bir register olması nedeniyle, programcı tarafından direkt olarak kullanılamaz. Ancak bu registerin değerini, stack üzerindeki bilgiler ile dolaylı olarak değiştirmek mümkündür.

3. Program Status Word Registeri :

Program status word registeri, kısaca PSW şeklinde gösterebiliriz. Bu registere flag registeri adını vereceğiz. Flag registeri diğer registerlerden oldukça farklıdır. Herşeyden önce, 'PSW' gösterimi bir operand değil, sadece bu registerden bahsetmek için kullanılan bir kısaltmadır. Flag registerinin bir bütün olarak veya 16 bitlik, 8 bitlik bir sayı olarak değerlendirilmesi söz konusu değildir. Çünkü flag registerinin sayısal değeri değil, kendisini oluşturan bitlerin herbirinin değeri önem taşımaktadır. Ancak, burada da bir kısıtlama olacak ve registeri oluşturan 16 bitten sadece 9 tanesi kullanılacaktır. Bu 9 bitin herbirinin mevcut değeri çeşitli kontrol komutların icra şekline etki etmekte ya da aritmetiksel veya mantıksal işlemlerin sonucu hakkında bilgi vermektedir.

4. Segment Registerları :

8088 ve 8086 işlemcisi kullanan bilgisayarlar, programların veya dataların belleğin iki ayrı alanına yerleştirirler. Bu program ve data alanları maximum 64 K büyüklüğünde olup segment olarak isimlendirilir. 8088 işlemcisi aynı anda 4 ayrı segmente sahip olabilir. Bu segmentler CODE, DATA, STACK ve EXTRA segment olarak isimlendirilir. Bu segmentlere ait ilk byte'ın diğer bir deyişle segmentin başlangıç adresinin gösterilmesi için 4 ayrı register kullanılır. Bunlar; sırası ile CS, DS, SS ve ES olup segment registerleri olarak isimlendirilirler. Code segment registeri olan CS icra gören programı içeren segmentin başlangıç adresini gösterir. Diğer bir deyişle, CS ile adreslenen segment içinde makine dili komutları yer almaktadır. Stack segment registeri olan SS, geçerli stack segmentin başlangıç adresini içerir. Stack segmenti t,çeşitli verilerin geçici olarak saklandığı bir bellek bloğudur. Data segment registeri olan DS ,geçerli data segmentinin başlangıç adresidir. Bu segment genel olarak program değişkenleri tarafından kullanılır. Diğer bir deyişle, bilgi işlem operasyonlarına tabi tutulan bilgiler, data segment içindeki lokasyonlarda yer alırlar. Ekstra segment registeri olan ES geçerli ekstra segmentin başlangıç adresini içerir. Ekstra segment, genel olarak string operasyonları için kullanılır. Yeterli bellek büyüklüğüne sahip olan sistemlerde bu 4 segment, belleğin farklı parçalarıdır. Geniş belleğe sahip olmayan sistemlerde ise, bu segmentler üstüste gelebilir.

8086 SEGMENTLERİ:

*Yukarıda registerlerden bahsederken, 8086 segmentlerinin de detaylı olarak bahsi geçmişti.Ancak yeniden segmentleri özetleyecek olursak:

Code Segment:

Bu segment çalışan programı içeren segmentin adresini gösterir. Diğer bir deyişle CS ile adreslenen segment içinde makine dili komutları yer almaktadır. Segmentin ilk baytından itibaren program parçası çalıştırılmaya başlanır.

Data Segment:

Data segmenti register'i olan DS geçerli data segmentinin başlangıç adresini içerir. Bu segment genel olarak program değişkenlerini tarafından kullanılır. Diğer bir deyişle bilgi işlem operasyonlarında tabii tutulan bilgiler Data Segment içindeki lokasyonlara (bölümlerde) yer alır.

Stack Segment:

Bu segment alt programlara giderken dönüş adreslerini saklamak program bitiminde işletim sistemine dönmek için gerekli bölgelerin saklanması için kullanılır.

Extra Segment:

Genel olarak string operasyonları için kullanılır.

DEBUG KOMUTLARI

Assembly komutlarının nasıl çalıştıklarını deneyerek görmek ve yazılan programların amacına uygun çalışıp çalışmadığını anlamak varsa hatalarını düzeltmek için DOS'un **DEBUG.COM** isimli programı kullanılmaktadır. Her ne kadar daha gelişmiş, kullanıcıya hitap eden debugger'lar olsa da kolay bulunulabilirliği ve basit uygulamalarda sağladığı rahatlık sayesinde tercih nedenidir. DEBUG programını kullanabilmek için komutlarını bilmek gereklidir. Burada DEBUG programının **bazı komutlarını** basit örnekler yardımıyla açıklamaya çalışacağız.

C:\>**DEBUG**

-

Çizgi işareti artık debug programının komut kabul etmeye hazır olduğunu belirtmektedir.

- **R (Register)** : Bu komut o andaki register değerlerini ve bayrakların durumunu belirlemekte kullanılır.

Bu komut yardımı ile register'lara istenilen değerler (hexadecimal olmak şartıyla) verilebilir.

Register'ları takip eden **NV UP EI PL NZ NA PO NC** değerleri ise bayrakların durumunu belirtmektedir.

-R F

NV UP EI PL NZ AC PO CY -

komutu ile sadece bayrakların durumu da görülerek gerekenler değiştirilebilir. Eğer flag değeri değiştirilmeyecek ise ara çubuğuna basarak bir sonraki flag değerine geçilebilir.

Flag İsmi	Set(1)	Clear (0)
-----------	--------	-----------

Overflow - OF	OV (overflow)	NV (not overflow)
Direction - DF	DN (decrement - down)	UP (increment - up)
Interrupt - IF	EI (enable)	DI (disable)
Sign - SF	NG (negative)	PL (plus)
Zero - ZF	ZR (zero)	NZ (not zero)
Auxiliary Carry- AF	AC (auxiliary carry)	NA (not auxiliary)
Parity - PF	PE (even parity)	PO (odd parity)
Carry - CF	CY (carry yes)	NC (not carry)

Debugger’da kullanılan Flag kısaltmaları

- **D (Dump / Display):** Bu komut ile belleğin istediğimiz segment'inin istediğimiz offset'inden itibaren bilgileri görebiliriz. İstersek görmek istediğimiz alanı sınırlama imkanına da sahibiz.Eğer D komutunun yanına gösterilmesi istenen aralığın adresi verilecek olursa sadece belirlenen adres aralığındaki değerler ekranda görüntülenecektir.

- **A (Assembly) :** Bu komut Debug yardımıyla basit programlar yazmayı sağlar. Debug içinde yazılan programlar COM uzantılı programlarda olduğu gibi 100H offset adresinden başlarlar. Bu neden ile program çalıştırılacağı zaman mutlaka IP register'ının değerinin **R** komutu ile 100H e getirilmesi gerekmektedir.

-A

2AF2:0100 MOV AX,5

2AF2:0103 ADD AX,4

2AF2:0106 SHR AX,1

2AF2:0108 ^C

-

A komutu verildikten sonra CS'in 100H adresinden itibaren assembly program yazmanıza imkan sağlanmış olur. Burada verilen her türlü sayı mutlaka hexadecimal formda olmalıdır. Komut yazmanız bittiğinde Ctrl-C tuşlarına basarak işlemi sonlandırabilirsiniz.

- **T (Trace) :** Yazılmış olan assembly komutları teker teker çalıştırılarak işlem sonunda register ve bayrakların aldıkları değerleri ekranda gösterir. T komutu çalışmaya IP nin gösterdiği noktadan itibaren başlar ve her komut işlendikten sonra durur.

Eğer kullanıcı ister ise T komutunun yanına bir sayı da verebilir. Bu durumda verilen sayı kadar komut satırı arka arkaya çalıştırılır, her seferinde register ve flag değerleri ekrana yazılır.

- **U (Unassembly) :** CS’de yazılı olan makine kodunu kullanıcının anlayacağı formdaki mnemonic'lere çevirmek için kullanılan bir komuttur. Her çağırılışında 32 byte uzunluğunda makine kodunun mnemonic'lerini kullanıcıya gösterir.

- **E (Edit) :** Bellekte herhangi bir yerdeki değerleri değiştirmek amacıyla kullanılan bir komuttur.

- **F (Fill) :** Bellek başlangıç ve bitiş adresleri belirlenen bir alanı belirlenen bir sayı ile doldurmak amacıyla kullanılır.

- **C (Compare) :** Belirlenen aralıktaki bilgileri karşılaştırmak amacıyla kullanılan bir komuttur. Bu komut 100H-110H adresindeki bilgiler ile 10H adresinden başlayan bilgileri karşılaştırır ve farklı olanları listeler.

- **G (Go):** Programın verilen bir adrese kadar ya da bitene kadar çalışmasını sağlamak amacıyla kullanılır.

- **H (Hex):** Verilen iki sayının hexadecimal toplamını ve farkını hesaplamak için kullanılır.

- **L (Load)** : Belleğe programı tekrar yüklemek için kullanılır.

- **M (Move)** : Bir yerdeki bir blok bilgiyi başka bir yere taşımak için kullanılır

- **P (Proceed)** : Bu komut T komutu ile benzer özellikler göstermektedir. Komutları çalıştırdıktan sonra register ve flag değerlerini ekranda gösterir. Ancak T komutundan en büyük farklılığı özellikle LOOP, CALL, INT gibi komutların işlenmesindedir. Bu komutlardan herhangi biri P komutu ile çalıştırılacak olursa o işlemin detaylarına girmeden sonucu üretir.

- **Q (Quit)**: Debugger dan çıkmak için kullanılan bir komuttur.

-Q

C:\>