Debug Komutları

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 debuger'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.

```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2AF2  ES=2AF2  SS=2AF2  CS=2AF2  IP=0100   NV UP EI PL NZ NA PO NC
2AF2:0100 8B24          MOV     SP,[SI]                          DS:0000=20CD
-
```

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

```
-R AX
AX 0000
:
```

':' işaretinden sonra AX register'i için istediğimiz değeri yazarak enter tuşuna basacak olursak;

```
:12AF
-R
AX=12AF  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2AF2  ES=2AF2  SS=2AF2  CS=2AF2  IP=0100   NV UP EI PL NZ NA PO NC
2AF2:0100 8B24          MOV     SP,[SI]                          DS:0000=20CD
-
```

AX registerinin değerinin 12AF olmasını sağlarız.

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)
Auxilary Carry- AF	AC (auxilary carry)	NA (not auxilary)
Parity - PF	PE (even parity)	PO (odd parity)
Carry - CF	CY (carry yes)	NC (not carry)

Tablo 0-1 Debuger'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.

```
-D DS:0
2AF2:0000  CD 20 FF 9F 00 9A EE FE-1D F0 4F 03 5A 25 8A 03   . ........O.Z%..
2AF2:0010  5A 25 17 03 5A 25 14 24-01 01 01 00 02 FF FF FF   Z%..Z%.$........
2AF2:0020  FF FF FF FF FF FF FF FF-FF FF FF FF 3B 25 4E 01   ............;%N.
2AF2:0030  16 2A 14 00 18 00 F2 2A-FF FF FF FF 00 00 CE 0B   .*.....*........
2AF2:0040  06 1E 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2AF2:0050  CD 21 CB 00 00 00 00 00-00 00 00 00 20 20 20 20   .!............
2AF2:0060  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20            .....
2AF2:0070  20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00            ........
-
```

Burada Data segment'in 0000H offset'inden itibaren olan kısmı gösterilmiştir. Eğer herhangi bir aralık verilmemiş ise 128 byte gösterir. Hexadecimal olarak belleğin dökümünü verdikten sonra yanında gösterilebilir ASCII karakter karşılıklarına da yer verilir.

```
-D DS:100,115
2AF2:0100  8B 24 17 5A 18 BA 4A 29-A2 4F F6 C9 07 AE 07 09   .$.Z..J).O......
2AF2:0110  20 8C 07 BA 4F 80                                  ...O.
-
```

Görüldüğü gibi 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.

```
-R
AX=12AF  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2AF2  ES=2AF2  SS=2AF2  CS=2AF2  IP=0100   NV UP EI PL NZ NA PO NC
2AF2:0100 B80500        MOV     AX,0005

-T

AX=0005  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2AF2  ES=2AF2  SS=2AF2  CS=2AF2  IP=0103   NV UP EI PL NZ NA PO NC
2AF2:0103 050400        ADD     AX,0004
-T

AX=0009  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2AF2  ES=2AF2  SS=2AF2  CS=2AF2  IP=0106   NV UP EI PL NZ NA PE NC
2AF2:0106 D1E8          SHR     AX,1
-T

AX=0004  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2AF2  ES=2AF2  SS=2AF2  CS=2AF2  IP=0108   NV UP EI PL NZ AC PO CY
2AF2:0108 A24FF6        MOV     [F64F],AL                       DS:F64F=11
-
```

Bu örnekte görüldüğü gibi az evvel yazdığımız assembly kodunu çalıştırmış olduk. Her işlemden sonra register ve bayraklar üzerindeki değişikleri rahat izlememiz için hazırlanmış bir komuttur.

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. Örnek;

```
- T 5
```

Ya da kullanıcı istediği bir offset'ten (adresten) itibaren kaç komut çalıştıracağını da söyleyebilir. Örnek;

```
-T=200,5
```

Bu komut 200H offset'inden itibaren 5 komut arka arkaya çalıştır demektir.

- **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.

```
-U CS:112
2AF2:0112 07            POP     ES
2AF2:0113 BA4F80        MOV     DX,804F
2AF2:0116 FC            CLD
2AF2:0117 807463BA      XOR     BYTE PTR [SI+63],BA
2AF2:011B C6418208      MOV     BYTE PTR [BX+DI-7E],08
2AF2:011F 83745BBA      XOR     WORD PTR [SI+5B],-46
2AF2:0123 D208          ROR     BYTE PTR [BX+SI],CL
2AF2:0125 847453        TEST    DH,[SI+53]
2AF2:0128 BADE04        MOV     DX,04DE
2AF2:012B 0908          OR      [BX+SI],CX
2AF2:012D 85744B        TEST    SI,[SI+4B]
2AF2:0130 BAEA08        MOV     DX,08EA
-
```

Bu komut ile CS:112 adresinden itibaren 32 byte makine kodu unassembly edilerek mnemonic karşılıkları gösterilmiştir.

İstenirse başlangıç adresi ve kaç byte unassembly edilmesi gerektiği de parametre olarak verilebilir.

```
-U CS:100L8
```

Bu komut CS:100H adresinden itibaren 8 byte'ı unassembly edecektir.

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

```
-
-E DS:10
2AF2:0010  5A.
```

Komut DS:10H adresindeki byte'in değerinin değiştirilmesi için kullanıcının bir hexadecimal değer girmesini bekler. Eğer değer değiştirilmeyecek ise ara çubuğu yardımıyla bir sonraki byte'a geçilerek onun değeri değiştirilebilir. Bu işlemi bitirmek için ENTER tuşuna basılmalıdır.

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

```
-
-F DS:100,120,AB
```

F komutundan yararlanarak DS'in 100H-120H arasındaki kısım ABH değeri ile doldurulacaktır. Bu işlemden sonra D komutu ile DS:100H adresinden itibaren incelersek işlemin olması gibi yapıldığını kontrol etmiş oluruz.

```
-D DS:100
2AF2:0100  AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB AB   ................
2AF2:0110  AB AB AB AB AB AB AB AB-AB AB AB AB AB AB AB AB   ................
2AF2:0120  AB 5B BA D2 08 84 74 53-BA DE 04 09 08 85 74 4B   .[....tS......tK
2AF2:0130  BA EA 08 86 74 43 12 28-BA F6 08 87 74 3B BA 02   ....tC.(....t;..
2AF2:0140  50 08 90 20 88 74 33 BA-0E 08 89 74 2B BA 1A 41   P.. .t3....t+..A
2AF2:0150  82 08 8A 74 23 BA 26 08-8B 74 1B BA 32 54 24 08   ...t#.&..t..2T$.
2AF2:0160  8D A2 BA 3E 08 8E 48 00-74 0B BA 86 08 FF 74 03   ...>..H.t.....t.
2AF2:0170  BA 4A 50 E9 1D A0 C0 C9-BA 56 50 E8 17 75 E8 AB   .JP......VP..u..
-
```

- **C (Compare) :** Belirlenen aralıktaki bilgileri karşılaştırmak amacıyla kullanılan bir komuttur.

```
-C 100,110,10
2AF2:0100  AB  5A  2AF2:0010
2AF2:0101  AB  25  2AF2:0011
2AF2:0102  AB  17  2AF2:0012
2AF2:0103  AB  03  2AF2:0013
2AF2:0104  AB  5A  2AF2:0014
2AF2:0105  AB  25  2AF2:0015
2AF2:0106  AB  14  2AF2:0016
2AF2:0107  AB  24  2AF2:0017
2AF2:0108  AB  01  2AF2:0018
2AF2:0109  AB  01  2AF2:0019
2AF2:010A  AB  01  2AF2:001A
2AF2:010B  AB  00  2AF2:001B
2AF2:010C  AB  02  2AF2:001C
2AF2:010D  AB  FF  2AF2:001D
2AF2:010E  AB  FF  2AF2:001E
2AF2:010F  AB  FF  2AF2:001F
2AF2:0110  AB  FF  2AF2:0020
```

Bu komut 100H-110H adresindeki bilgiler ile 10H adresinden başlayan bilgileri karşılaştırır ve farklı olanları listeler.

```
-C 100,110,100
-
```

100H-110H adresindekileri yine 100H adresindekiler ile karşılaştırmaktadır. Herhangi bir farklılık bulunmadığı için ekranda hiçbir bilgi görünmez.

```
-C DS:100,110,CS:300
2AF2:0100  AB  25  2AF2:0300
2AF2:0101  AB  44  2AF2:0301
2AF2:0102  AB  68  2AF2:0302
2AF2:0103  AB  20  2AF2:0303
2AF2:0104  AB  DD  2AF2:0304
2AF2:0105  AB  20  2AF2:0305
2AF2:0106  AB  91  2AF2:0306
2AF2:0107  AB  CF  2AF2:0307
2AF2:0108  AB  3A  2AF2:0308
2AF2:0109  AB  1C  2AF2:0309
2AF2:010A  AB  04  2AF2:030A
2AF2:010B  AB  1C  2AF2:030B
2AF2:010C  AB  6B  2AF2:030C
2AF2:010D  AB  82  2AF2:030D
2AF2:010E  AB  04  2AF2:030E
2AF2:010F  AB  1C  2AF2:030F
2AF2:0110  AB  04  2AF2:0310
```

Görüldüğü gibi farklı segment'ler üzerindeki bilgiler de karşılaştırılabilmektedir.

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

```
-G

Program terminated normally
```

Program eğer düzgün çalıştı ise yukarıdaki mesaj ekrana gelecektir. Programın dönüş adreslerinde bir hata varsa ya da programda bir mantık hatası yapılmış ise program hiçbir zaman bitmez. Bu durumda makinenin reset edilmesi gerekecektir.

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

```
-H2E,3
0031  002B
-
```

İlk gösterilen değer iki hexadecimal sayının toplamı diğeri ise farkıdır.

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

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

```
- M DS:100,110,CS:500
```

Bu komut ile DS:100H ile DS:110H arasında kalan bilgi CS:500H adresinden itibaren yerleştirilecektir.

- **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. (Eğer bir LOOP ise çevrim sonuçlanana kadar, bir CALL ise işlemler yerine getirilip geri dönülene kadar ya da bir INT ise interrupt handler'in yapması gereken tüm işlemler yapıldıktan sonra register ve Bayrakların durumlarını gösterir).

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

```
-Q

C:\>
```