Mikroişlemciler, I. Uygulama Yönergesi

Amaç: Assembly dili programların komut satırından (CLI) yapılması.
Hedef: Çalıştırıldığında, "Merhaba Dunya" mesajını ekrana yazdıran bir programın, Assembly dilinde kodlanması için, hata ayıklayıcı (DEBUG) aracı ve komutlarını kullanmayı öğrenme.

Bu uygulamada, çalıştırılan programların beklenen sonuçları üretmemesi durumunda, hangi noktada ne tip bir sorun ile karşılaştığını anlamak için MicroSoft tarafından geliştirilen ve işletim sistemi ile birlikte dağıtılan, **debug** hata ayıklama aracı kullanılacaktır. Debug'ın komut satırından kullanımı kısaca açıklandıktan sonra bu araç kullanılarak Assembly dilindeki basit programların Assembler veya başka bir derleyiciye gerek duymadan komut satırından yazılmasına bir örnek verilecektir.

Uyarı: Bu oturumda kullanılan debug.exe, MicroSoft tarafından 16-bit DOS işletim sisteminden başlayarak 32-bit Windows 8 sürümü de dahil olmak üzere tüm işletim sistemlerinde bulunduğu için doğrudan çalıştırılabilir olmasına karşılık, 64-bit işletim sistemleri için MicroSoft'un bu aracı dağıtmaması nedeniyle çalıştırılması olanaksızdır. 64-bit işletim sistemi kurulu bilgisayarlarda bu yönergede anlatılan debug'ı çalıştırabilmek için, açık kaynak kodlu QEMU işlemci emulatörüne grafik kullanıcı arayüzü sağlayan **QemuSimpleBoot** (qsib) kullanabilirsiniz. Qsib çalıştırıldığında, disk imajı (IMG) seçeneği işaretlendikten sonra, içinde 8-bit M68xx ve 16-bit x86 Assembly dilleri için gereken araçların bulunduğu çalıştırılabilir (bootable) disk imaj dosyasından işletim sistemini başlatarak debug'ı çalıştırabilirsiniz. Yazdığınız programın oluşturulduğu/çalıştırıldığı 16-bit DOS ortamından, bilgisayarınızın dosya sistemine aktarılması için **MagicISO** adlı ISO kalıp hazırlama/düzenleme programını kullanabilirsiniz.

Kullanılacak araç, komut satırında yalnız bir – işareti ile (prompt) sizden tek karakter ile kısaltılmış komut beklemekte ve her girdiğiniz komuttan sonra bir enter ile komutu girdiğinizi belirtmeniz gerekmektedir. Komut istemini **cmd** programını çalıştırarak başlatınız. Bu sırada dosya_adı ve klasör_adı tamamlama seçeneği için **/f** seçeneğini de kullanabilirsiniz. Debug programını komut satırından **debug** yazarak çalıştırınız. Debug çalışacak ve sizden bir komut beklediğini – işareti ile ifade edecektir.

```
C:\Users\Ali>debug (enter)
-
```

Assembly dilinde bir kod yazacağımız için **A**ssemble kısaltması olarak **a** harfi girip enter'a vurunca;

```
C:\Users\Ali>debug
-a (enter)
17C2:0100
```

Programınızın başlayacağı bellek adresinin Segment:Offset biçimindeki değeri, Assembly kodu yazmanızın beklendiği satırın önünde gösterilecektir. Bilgisayarınızda bu işlemleri yapmadan önce çalışan programlar nedeniyle, bu örnekte görüntülenen Segment değeri (17C2), sizin kendi bilgisayarınızda göreceğiniz değerden farklı olacaktır. Programımızın ilk satırını **mov ah,9** yazarak enter'a vurduğunuzda ikinci satırı girmeniz için adres gerektiği şekilde artırılarak (17C2:0102) satır önünde gösterilecektir. Bu satırı girdiğiniz gibi aşağıdaki gösterilen diğer satırları da enter ile sonlandırarak giriniz.

```
17C2:0100 mov ah,9
17C2:0102 mov dx,0109
17C2:0105 int 21
17C2:0107 int 20
17C2:0109
```

Assembly dilindeki programın kodu girişinin bittiğini debug'a belirtmek için bir kez daha enter'a vurularak tekrar sizden kısaltılmış komut bekleme durumuna geçirmelisiniz. Ekranda görülen adreslerden, yazdığımız program kodunun bellekte tutulduğu kısmın (**C**ode **S**egment, **cs**, bu örnekte 17C2) içinde, 0100_{16} adresinden başlayarak 0108_{16} adresine kadar olduğu görülmektedir. Girdiğiniz Assembly dilindeki program, DOS/BIOS çağrıları kullanarak ekrana kullanıcının belirlediği bir metin mesajı yazdırmak için kullanılacaktır. Programın en küçük boyutta tutulması amacı ile, metini program kodunun bittiği adresten başlayarak yazmak için debug'ın **E**dit komutu kısaltması **e** harfini düzenlemek istediğiniz adresin ana bellekte tutulduğu kısmın (**D**ata **S**egment, **ds**) ön ek olarak kullanıldığı şekilde **e ds:109** olarak girdiğinizde;

```
-e ds:0109 'Merhaba Dunya$' (enter)
```

ilk Assembly dili programınız çalışmaya hazır demektir. Programınızı çalıştırmadan önce, debug'ın **U**nassemble komutu kısaltması **u** ile çalıştırılabilir kodun Assembly dili emirlerinden oluşan kaynak kod şeklini görmeniz mümkündür. Program, çalıştırılabilir kodun içinde bulunduğu **cs**'nin içinde 0100_{16} adresinden başlayan bellek adresine tutulduğu için **u cs:100** yazmanız yeterlidir.

```
-u cs:100 (enter)
17C2:0100 B409          MOV     AH,09           ; DOS Int 21'in 9 işlevi
17C2:0102 BA0901        MOV     DX,0109         ; Metin başlangıç adresi
17C2:0105 CD21          INT     21              ; DOS Int 21'i çağır
17C2:0107 CD20          INT     20              ; Programı sonlandır
17C2:0109 4D            DEC     BP              ; M karakter kodu
17C2:010A 65            DB      65              ; e
17C2:010B 7268          JB      0175            ; r, h
17C2:010D 61            DB      61              ; a
17C2:010E 62            DB      62              ; b
17C2:010F 61            DB      61              ; a
17C2:0110 204475        AND     [SI+75],AL      ; White Space, D, u
17C2:0113 6E            DB      6E              ; n
17C2:0114 7961          JNS     0177            ; ya
17C2:0116 2400          AND     AL,00           ; $ (metin sonlandırma karakteri)
17C2:0118 0000          ADD     [BX+SI],AL
17C2:011A 0000          ADD     [BX+SI],AL
17C2:011C 3400          XOR     AL,00
17C2:011E B117          MOV     CL,17
-
```

Yukarıdaki programın **u** komutu ile alınan tersine Assembly listesinin kolay anlaşılabilmesi için, noktalı virgül karakteri sonrasında yorum (comment) kısımları eklenerek bu yönergeye aktarılmıştır. Unassemble edilen bir kodun açıklama kısımları olmayacaktır. Assembly dili komutları makine dili kodları ile bire bir eşleştirilebildiği için Unassemble komutu ile çalıştırılabilir kodların (exe dosyalar) karşı düştüğü Assembly dili biçimine geçirmek çok basit bir işlemdir. Unassemble komutu, nasıl çalıştığı bilinmeyen kodların (malware gibi) yapısının tersine mühendislik (reverse engineering) yöntemleri ile analiz edilebilmesi için kullanılabilecek **en önemli** araçlardan birisidir. Benzer şekilde, programı debug'ın Assembly dili karşılıklarına dönüştürme (unassemble) işlemi yapmadan doğrudan göstermesi için **D**ump (dök) komutu kısaltması **d** kullanılabilir;

```
-d cs:100 (enter)
17C2:0100  B4 09 BA 09 01 CD 21 CD-20 4D 65 72 68 61 62 61   ......!. Merhaba
17C2:0110  20 44 75 6E 79 61 24 00-00 00 00 00 34 00 B1 17    Dunya$.....4...
17C2:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
17C2:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
17C2:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
17C2:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
17C2:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
17C2:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Debug aracının **W**rite komut kısaltması **w** ile yazdığımız programı bir dosyaya saklayarak gerektiğinde komut satırından doğrudan çalıştırabiliriz. Yazma işleminden önce programın uzunluğunu sayarak diske'e yazılacak byte sayısını x86 mimarisinde genellikle döngü sayacı olarak kullanılan 16-bit uzunluktaki CX kaydedicisi içerisine saklamanız gerekmektedir. Programın uzunluğu 23 byte olduğu için CX kaydedicisine $23_{10}=17_{16}$ değeri saklamak için sırası ile **R**egister komutu kısaltması **r**, **cx** parametresi ile girildiğinde CX kaydedicisinin o anki değeri (0000) gösterildikten sonra yeni değeri ":" karakteri görüldüğünde 17 olarak girilmelidir;

```
-r cx (enter)
CX 0000
:

-r cx
CX 0000
:17 (enter)
-
```

Çalıştırılabilir kodun saklanacağı dosya boyu CX kaydedicisine saklandığı için, dosya adını **N**ame komutu kısaltması **n** harfinden sonra parametre olarak belirterek;

```
-n p1.com (enter)
-
```

çalıştırılabilir kod, **W**rite komutu kısaltması **w** ile disk'te adı belirtilen dosyaya (p1.com) yazılır;

```
-w
Writing 00017 bytes
-
```

Programın çalıştırılabilir kodu bir komut dosyasına saklandığı için, artık debug aracından **Q**uit komutu kısaltması **q** ile çıkılabilir;

```
-q (enter)

C:\Users\Ali>
```

Çalıştırılabilir dosya artık diske yazıldığı için, işletim sistemi komut satırından adı verilerek doğrudan veya yine debug aracılığı ile kaydedici içerikleri de gözlemlenerek adım adım çalıştırılabilir.

```
C:\Users\Ali>p1 (enter)
Merhaba Dunya
C:\Users\Ali>
```

veya

```
C:\Users\Ali>debug p1.com
-
```

Programın kaydedici içeriklerini de izleyerek adım adım çalıştırılması için **T**race komutu kısaltması **t** kullanarak her seferinde 1 adım çalıştırılabilir;

```
-t

AX=0900  BX=0000  CX=0017  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=17D3  ES=17D3  SS=17D3  CS=17D3  IP=0102   NV UP EI PL NZ NA PO NC
17D3:0102 BA0901        MOV     DX,0109
```

Eğer her seferinde 1 adım çok yavaş (ve detaylı) görünürse **t** komutuna parametre olarak bir seferde kaç emir çalıştırılacağı da belirtilebilir;

```
-t 3

AX=0900  BX=0000  CX=0017  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=17D3  ES=17D3  SS=17D3  CS=17D3  IP=0102   NV UP EI PL NZ NA PO NC
17D3:0102 BA0901        MOV     DX,0109

AX=0900  BX=0000  CX=0017  DX=0109  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=17D3  ES=17D3  SS=17D3  CS=17D3  IP=0105   NV UP EI PL NZ NA PO NC
17D3:0105 CD21          INT     21

AX=0900  BX=0000  CX=0017  DX=0109  SP=FFF8  BP=0000  SI=0000  DI=0000
DS=17D3  ES=17D3  SS=17D3  CS=00A7  IP=107C   NV UP DI PL NZ NA PO NC
00A7:107C 90            NOP
-
```

Kaynak : http://www.intel-assembler.it/portale/5/Write-an-assembly-program-using-DEBUG/Write-an-assembly-program-using-DEBUG.asp