Basit Bootstrap Uygulaması

BELGE HAKKINDA

Bu belgedeki eksik, yanliş ya da geliştirilmesi gerektiğini düşündüğünüz yerleri e-posta yoluyla bildirebilirsiniz.

Belgenin ilk oluşturulma tarihi: 13 Nisan 2007

/***
* Cihan KÖMEÇOĞLU *
* *
* cihan [at] akademi.enderunix.org *
* *
* EnderUNIX Yazılım Geliştirme Takımı *
* *
* http://www.enderunix.org *
* *
* Sürüm : 1.0 *
* *
* Tarih : 15.04.2007 *
* *
* Etiketler : işletim sistemi,boot,çekirdek programlama,assembly *
* *
* Seviye : Başlangıç - Orta Seviye *
* *
* Makalenin en yeni versiyonu : http://www.enderunix.org/docs/helloboot.pdf adresinden elde edilebilir. *
* *
***/

Basit bir boot işleminin olması için temel olarak bir kaç şey gereklidir. Bunlar ise:

- BIOS sizin boot kodunuzu 07C00h adresine yerleştirecektir ve işlemler bu adresi dikkate alınarak yapılır.
- Derlediğiniz boot kodu düz ikili dosya(plain binary file) formatında olmak zorundadır. Bu formatta nasıl derleneceğini ilerleyen kısımlarda göstereceğim
- Boot kodunuz 512 byte olması zorunludur.
- Boot kodunun son iki byte'ı AA55h olmak zorundadır. Bu son iki byte EOF gibidir. Yani boot kodunun sonuna geldiği buradan anlaşılmaktadır. Bu byte'a "Boot record Signature" (boot kayıt imzası) denilmektedir. Eğer kodunuza bu son iki byte'ı eklemezseniz boot sırasında "Boot Record Signature AA55 not found" hatası alırsınız.
- Boot kodumuzu derledikten sonra diskin 0. sektörüne yazılmak zorundadır.
Çünkü bilgisayar açılmaya başladıktan sonra bios'ta belirtilen sırada disklerin 0.sektöründe boot kodu arayacaktır.

Not: Boot kodlarımızı disket ile test edersek daha iyi olur. Yoksa sisteminizi kurtarabilmek için tekrar işletim sisteminin bootstrap'ını yüklemek zorunda kalırsınız. Bu da zahmetli iş.

Bu kısa bilgiden sonra sadece temel bootstrap işlemlerini kapsayan ufak bir boot kodu yazabiliriz.

```
;Mini bootstrap
       ;mini.asm

      dongu:
            jmp dongu

      times 510-($-$$) db 0
      dw 0AA55h
```

Bu assembler kodunu derleyebilmek için nasm derleyicisini kullanacağız.
Nasm ile derleyebilmek için:

```
$ nasm -f bin mini.asm -o miniboot
```

Diskete yazmak için:

```
$ dd if=miniboot of=/dev/fd0
```

Yukarıdaki boot kodumuz bios tarafından belleğin 07C00h adresine yüklenecektir.
Boot sırasında ekrana bir çıktı almayacaksınız. Sadece "jmp dongu" satırı ile sonsuz bir döngü içine gireceksiniz. Dökümanın ilerki kısımlarında "hello world " programının nasıl boot edileceğini anlatacağım.

```
times 510-($-$$) db 0
      dw 0AA55h
```

Bu satır ile boot programımızı 512 byte'a tamamlıyoruz. Daha önceden de belirtiğim gibi boot programımız 512 byte olmak zorundadır. times sadece nasm da tanımlı bir koddur. ilk satırda 510 byte'a kadar "0" ile doldurulmaktadır.Bunu sağlayan "times" komutudur. AA55h değeri iki byte olduğu için toplam 512 byte olmaktadır.

Yukarıdaki bootstrap kodumuz değişken kullandığımızda sorun çıkacaktır.Çünkü yukarıdaki kod ile segmentlerin değerlerini bilemeyiz. Bunu yapmazsak değişkenlerimize erişemeyiz.

Bu sorunu halletmek için yukarıdaki kodumuza aşağıdaki satırları eklemek yeterli olacaktır.

```
mov ax, cs
mov ds, ax
mov es, ax
```

Burada data segment ve extra data segment değerlerini code segment'in değeriolarak ayarlıyoruz.
Yani kodlarımız aynı zamanda birer değişken gibi olacaktır. Bu kodu derleyip test ettiğinizde bir öncekinden farklı bir şey görmeyeceksiniz. Ama değişkenlere erişmek istediğimizde yapılması gereken bir işlemdir.

Şimdi Hello World programını nasıl boot edebileceğimizi görelim.

- Hello World'ü boot etmek ?

Kodları açıklamadan önce işlemlerin nasıl olduğu konusunda kısa bir bilgi vereceğim.
Önceden sadece bios 07C00h bellek bölgesine boostrapımızı kopyaladı ve kodumuz orada sonsuz bir döngü içine girmişti.Burada yapacağımız ilk olarak disketin 0. sektörüne boot kodumuzu , 1. sektörüne de hello world programını yüklemektir.Bunları yaptıktan sonra disketin 1. sektöründe bulunan hello world programını belleğin belli bir bölgesine bios interruptlarını kullanarak yerleştirmek. Daha önceden de bahsettiğim gibi 0.sektör sadece bootstrapımız diğer sektörde herhangi bir programolabilir. Diğer programlar illa 1.sektörden başlamak zorunda da değildir. Herhangi bir sektörde olsa da olur. Önemli olan programın hangi sektörde ve kaç sektör okumamız gerektiğini bilmemiz önemlidir.

Bu arada bios interruptları aygıtlara erişmemizi ve onları kullanmamızı sağlar. ğğin hello world programında ekrana çıktı vermek için int 10h interruptını kullandık. Bir interruptı kullanabilmek için belli registerlara belli değerler atamamız gerekmektetir. Sonra interruptı çağırarak işlemi yaptırabiliriz.

Aşağıdaki adresten Interruptların listesini ve nasıl kullanılacağını görebilirsiniz.

http://en.wikipedia.org/wiki/BIOS_call

Şimdi kodumuza geçebiliriz.

```
;*****************boot.asm ***************

[ORG 0]

      jmp reset

reset:
      mov ax, 0   ;ax ve floppy sürücüsü resetlenir
      mov dl, 0
      int 13h
      jc reset

floppy:
      mov ax, 1000h     ;hellow world programının yükleneceği bellek bölgesi
      mov es, ax
      mov bx, 0

      mov ah, 2   ;belirtilen bellek bölgesine programı yükle
      mov al, 1   ;disketten 1 sektör oku
      mov ch, 0   ;silindir = 0
      mov cl, 2   ;sektör = 2
      mov dh, 0   ;head = 0
      mov dl, 0   ; disket sürücüsü olduğunu belirtir
      int 13h

      jc floppy   ;hata varsa tekrar okumayı dene

      jmp 1000h:0000    ;hello world'un yüklü olduğu bellek bölgesine zıpla

times 510-($-$$) db 0
dw 0AA55h

;*******************************************
```

Derlemek için:

```
      $ nasm -f bin boot.asm -o progboot
```

Diskete yazmak için:

```
      # dd  if=progboot of=/dev/fd0
```

```
      mov ax, 1000h
      mov es, ax
      mov bx, 0
```

Yukarıdaki kod ile es registerına "hello world" programının yükleneceği bellek bölgesi belirtilir. Bu bellek bölgesi bios'un yüklü olduğu A000:000 - FFFF:000F adresler haricindeki herhangi bir bellek bölgesine kopyalayabiliriz. bx registerına da bu adresin offseti belirtilir. Yani programımız 1000:0000 adresinde bulunacaktır.

```
      mov ah, 2
      mov al, 1
      mov ch, 0
      mov cl, 2
      mov dh, 0
      mov dl, 0
```

```
int 13h
```

Burada ise ah registerına 2 değeri atanarak verilerin daha önceden belirtilen bellek bölgesine yükleneceği belirtilmektedir. al registerına da kaç sektör okunuluacağı belirtilmektedir. Hello world programı 512 byte'dan küçük bir program olduğu için 1 sektör okumak yeterlidir.
Burada dikkat edilecek bir nokta ise cl registerına 2 değeri atanmıştır. Bu sektör numarasını belirtmektedir. Fakat biz " Hello world" programını disketin 1. sektörüne yazacağız. Diskette ilk sektör numarası 0'dır. Ama bios rutinlerinde ilk sektör 0'dan değil 1'den başlamaktadır. bu yüzden biz 2 değerini atadık. Yani cl registerına 2 değerini atayarak disketin 1. sektörünü belirtmiş olduk ve en sonunda bios interruptını (int 13h) çağırarak işlemi gerçekleştirmiş oluyoruz.

```
jmp 1000h:0000
```

Bu jump işlemi ile de hello.asm nin yüklendiği bellek bölgesine zıplıyoruz. Boot.asm'nin "floppy" tagında es ve bx registerlarına hello.asm'nin yükleneceği bellek bölgesini belirtmiştik. Bu işlem ile de bu registerlara belirtiğimiz bellek bölgesine zıplıyoruz. "**`jmp 1000h:0000`**" komutu hello.asm'deki "**`jmp main`**" satırının bulunduğu kısma zıplatır.

```
;****************** hello.asm *********
[ORG 0]

jmp main

msg db 'Hello World'

main:
      mov si, msg ;mesajı yazdırmak için
      mov ax, cs
      mov ds, ax
      mov es, ax
write:
      lodsb       ;stringi AL registerına kopyala
      cmp al,0    ;mesajın sonuna gelip gelinmediği kontrol edilir
      je loops

      mov ah, 0Eh ;karakteri ekrana yazdırır
      mov bx, 7
      int 10h

      jmp write

loops:
      jmp loops

;***************************************
```

Derlemek için:

```
$ nasm -f bin hello.asm -o hello
```

Diskete kopyalamak için:

```
# dd seek=1 conv=sync if=hello of=/dev/fd0
```

```
mov si, msg
```

Si registerı data segmentde bulunan verilerin içeriğini göstermek için kullanılan bir index registerıdır. Böylelikle msg değişkeninin içeriği gösterilebilmektedir.

```
        mov ax, cs
        mov ds, ax
        mov es, ax
```

Burada yapılan işlemse daha önceden bahsettiğimiz gibi tanımladığımız değişkenlere erişebilmek için registerları güncelliyoruz. Dikkat ederseniz boot.asm'de bunu yapmaya gerek yoktu çünkü herhangi bir değişken tanımlamamıştık. Fakat hello.asm'de msg değişkeni tanımladığımız için registerları güncellememiz gerekmektedir.

```
Write:
                .
                .
                .
        mov ah, 0Eh
        mov bx, 7
        int 10h

        jmp write
                        .
                .
                .
```

Yukarıdaki kodda ise bios interruptını kullanarak stringimizi karakter karakter yazdırıyoruz.Stringin sonuna gelip gelmediğimizi ise al registerına bakarak karar veriyoruz.ah registerına 0Eh değeri atanarak ekrana çıktı verileceği belirtilmektedir.