گزارش روند برنامه نویسی

x86 "Hello world!nameSTDnumber" bootloader with assembly

نام و نام خانوادگی : مریم سعیدمهر

ش.د. : ۹۶۲۹۳۷۳

استاد مربوطه : دکتر زالی

تکلیف اول برنامه نویسی درس سیستم عامل

ترم اول ۹۸-۹۹

دانشگاه صنعتی اصفهان

بوت لودر اولین کدی هست که توی یه کامپیوتر بعد از روشن شدن اجرا میشود. در ادامه روند برنامه نویسی یک بوت لودر ساده به زبان اسمبلی را در یک سیستم ۶۴ بیتی توضیح خواهم داد.

* پیشنیاز

من دارم از سیستم عامل "Ubuntu 180.4.3 LTS" استفاده میکنم. برنامه هایی که بهش نیاز داریم، GCC یا سری کامپایلر های گنو هستش ؛ به nasm یعنی اسمبلر نیاز داریم و همچنین به ld یعنی لینوکس لینکر. اینها برای اسمبل کردن و کامپایل بوت لودر کافی هستن اما برای شبیه سازی محیط یک کامپیوتر به نرم افزار qemu نیاز داریم.

****

برای نصب qemu مراحل زیر رو دنبال کنید :

ابتدا کد زیر رو توی محیط ترمینال وارد کنید :

```
~$ sudo apt-get install qemu
```

بعد از اون برای اطمینان از نصب qemu این کد رو اجرا کنید اگر پنجره ای باز شد یعنی مراحل نصب رو درست طی کردید :

```
~$ qemu-system-x86_64
```

و برای نصب nasm هم از کد زیر استفاده کنید :

```
~$ sudo apt install nasm
```

****

* وقتی یه سیستم x86 رو روشن می کنیم چی میشه ؟

حالا که تمام نیاز ها رو برای ساخت یه بوت لودر برطرف کردیم توضیح خواهم داد که یک بوت لودر چطوری کار میکنه. یه بوت لودر 512 بیت از حافظه ی bootable را میخونه ، اگر که ۲ بایت آخر از اون برابر عدد جادویی بود ، اون رو به عنوان یک بوت لودر شناسایی میکنه. برنامه ی ما هم باید در نقطه ی 0x7c00 از حافظه باشه.

عدد جادویی برابر 0x55aa هست تو مبنای hex. اما همیشه ۲ بیت آخر با هم جا به جا میشن ، پس ما باید ۲ بیت آخر رو برابر 0xaa55 قرار بدیم تا به عنوان عدد جادویی شناخته بشه.

* توضیحات کد به فارسی

در کد زیر ، اومدیم و تک تک کاراکتر ها رو توی al گذاشتیم و CPU رو با اینتراپت 0x10 صدا زدیم و گفتیم اگر که به صفر رسید این چرخه برنامه رو ببند.

```
[bits 16]                ; داریم توی مدل ۱۶ بیت کد میزنیم
[org 0x7c00]             ; نقطه شروع برنامه را مشخص میکند
init:
  mov si, msg            ;  لود آدرس پیغام در رجیستر داده شده
  mov ah, 0x0e           ; مقدار هگز را در رجیستر داده شده میریزد که مخصوص رنگ سفید است برای نمایش خروجی
print_char:
  lodsb                  ; بایت فعلی را از رجیستر اِس آی خوانده و در رجیستر آ اِل میریزد
  cmp al, 0              ; محتوای رجیستر آ اِل را با صفر مقایسه میکند
  je done                ; if AL == 0, go to "done"
  int 0x10               ; چاپ در خروجی با استفاده از اینتراپت ۱۶ ( به فرم هگزان)
  jmp print_char         ; با بایت بعدی ، حلقه تکرار شود
```

```
done:
  hlt ; stop execution

msg: db "Hello world!Maryam 9629373", 0          ; گذاشتن صفر انتهایی فراموش نشود

times 510-($-$$) db 0          ; پر کردن ۵۱۰ بیت آخر فایل با صفر
dw 0xaa55                      ; عدد جادویی که به بایوس میگوید برنامه بوتیبل است
```

در آخر هم ، از اونجایی که مهم نیست اصلا اون ۵۱۰ بیت دیگه چی باشن ، ما هم با صفر پرشون میکنیم :

* کامپایل و اجرای کد در اِمولاتور

```
~$ nasm -o boot.bin boot.asm
~$ qemu-system-x86_64 boot.bin
```

* منابع

از این سایت استفاده کردم (یافتن سایت هم با سرچ کردن عبارت "How to write a bootloader" در گوگل بوده است)

**** توجه : این کد با Intel syntax زده شده تا بتوان آن را با nasm اسمبل کرد.**

**** توصیه میشود از این لینک بازدید کنید :)**

**** تاریخ تحویل : ۱۳۹۸/۷/۲۶**