Writing Hello World Bootloader

Introduction

Bootloader is a piece of code that is executed once the system is booted. It is 512 byte long code which resides on 1st sector of booting device (floppy, hard drive etc). I recommend to read the tutorial The Booting Process before proceeding this tutorial. We will make a small bootloader that will print Hello World on screen. We will use a floppy for booting (I know you dont want to crash your harddrive for this). We will use normal 3.5 inch floppy for this.

Getting Tools

For making a bootloader we will need some tools which will be used to compile and write the boot loader.

NASM

The assembler used here is NASM (Netwide Assembler). NASM is an 80x86 assembler designed for portability and modularity. It supports a range of object file formats, including Linux and NetBSD/FreeBSD a.out, ELF, COFF, Microsoft 16-bit OBJ and Win32. It will also output plain binary files. Its syntax is designed to be simple and easy to understand, similar to Intel's but less complex. It supports Pentium, P6, MMX, 3DNow!, SSE and SSE2 opcodes, and has macro capability.

- Nasm 0.98.35 for Windows(ZIP, 218KB)
- Nasm 0.98.35 for i386 Linux(RPM, 133KB)

Partcopy

Windows user can use Partcopy to write bootsector on floppy/hdd disks.

- Partcopy (ZIP, 10KB)

3.5 inch Floppy Disk

You dont want to damage your harddisk so it is recommended to use floppy for this work. Please use a floppy which does not have any important data (because you will be going to damage that one). So use a fresh formated floppy disk.

Creating Bootloader

Now its time to start coding our bootloader. For this you will require some prior knowledge of Assembly. I assume that you know assembly level language.

First try: Hanging Bootloader

We will start with a bootloder that will do nothing. Neither it will print anything. Our system will be just booted from this and it will be hanged. Open your favourite text editor and write following code and save the file as ourbootloader.asm.

```
[BITS 16]       ;tell the assembler that its a 16 bit code
[ORG 0x7C00]    ;Origin, tell the assembler that where the code will
;be in memory after it is been loaded

JMP $           ;infinite loop

TIMES 510 - ($ - $$) db 0       ;fill the rest of sector with 0
DW 0xAA55                       ; add boot signature at the end of bootloader
```

`[BITS 16]`: Our code starts with [BITS 16] which is an assembler directive. This will tell assembler that our code is a 16 bit code.

`[ORG 0x7C00]`: Then we have used [ORG 0x7C00] which tell assembler to assemble the instructions from Origin 0x7C00. BIOS loads bootloader at physical address 0x7C00 hence we have assemble our bootloader starting from that location.

`JMP $`: JMP at location $ means jumping to the same location. Thus this nothing but an infinite loop. We just want to hang our code here.

`TIMES 510 - ($ - $$) db 0`: As bootloader is always of length 512 bytes, our code does not fit in this size as its small. We need to use rest of memory and hence we clear it out using TIMES directive. $ stands for start of instruction and $$ stands for start of program. Thus ($ - $$) means length of our code.

`DW 0xAA55`: This is boot signature. This tells the BIOS that this is a valid bootloader. If bios does not get 0x55 and 0xAA at the end of the bootloader than it will treat bootloader as invalid. Thus we provide this two bytes at the end of our bootloader.

Compiling / Assembling

Its time to compile our bootloader using NASM. For that you need to type:

```
nasm ourbootloader.asm -f bin -o boot.bin
```

This will create a file boot.bin. Now check the size of this file. It must be 512 bytes. If its not that then there must be some problem. Our bootloader may not be compiled properly. Check the code once again.

Copying bootsector to floppy disk

Windows user can use PARTCOPY for writing their bootloader to floppy disk. For this you need to write command:

```
partcopy boot.bin 0 200 -f0
```

Linux users can use dd command for this. For this Insert a floppy but dont mount it and then write following command:

```
dd if=boot.bin bs=512 of=/dev/fd0
```

Congratulation!!!! You have just made your bootloader. Now you are ready to boot your system using this floppy.

So insert this floppy in floppy drive and boot your system. Dont forget to select floppy as your 1st boot device from BIOS. Once booted with this floppy your system will not do anything but it will get hanged.

Second try: Print a character Bootloader

We have successfully created our hanging bootloader. Now we will add some code in this bootloader to print a character 'A' on screen. For printing we will use BIOS video interrupt int 0x10.

```
INT 0x10 is a BIOS video interrupt. All the video related calls are made through this interrupt.
To use this interrupt we need to set the values of some register.
AL = ASCII value of character to display
AH = 0x0E ;Teletype mode (This will tell bios that we want to print one character on screen)
BL = Text Attribute (This will be the fore ground and background color
         of character to be displayed. 0x07 in our case.)
BH = Page Number (0x00 for most of the cases)
```

```
Once all the registers all filled with appropriate value, we can call interrupt.
```

So its time to write code for a bootloader that will print 'A' on screen and then it will be hanged. The code for this is:

```
[BITS 16]       ;Tells the assembler that its a 16 bit code
[ORG 0x7C00]    ;Origin, tell the assembler that where the code will
        ;be in memory after it is been loaded

MOV AL, 65
CALL PrintCharacter
JMP $           ;Infinite loop, hang it here.


PrintCharacter: ;Procedure to print character on screen
        ;Assume that ASCII value is in register AL
MOV AH, 0x0E    ;Tell BIOS that we need to print one charater on screen.
MOV BH, 0x00    ;Page no.
MOV BL, 0x07    ;Text attribute 0x07 is lightgrey font on black background

INT 0x10        ;Call video interrupt
RET             ;Return to calling procedure

TIMES 510 - ($ - $$) db 0       ;Fill the rest of sector with 0
DW 0xAA55                       ;Add boot signature at the end of bootloader
```

The code is pretty self explainatory. Here we have made a procedure PrintCharater which will print a character on screen. It assumes that we provide the ASCII value in AL register.

Compile this code as mention above and write the bootloader on floppy disk. Now boot your system using this floppy and voila!! There is an 'A' on the screen!!!

Third try: Hello World Bootloader

Its time to create our final Hello World bootloader. We have enough expirence now and can code it without wasting a second. So once again start your favourite text editor and start writing following code:

```
[BITS 16]       ;Tells the assembler that its a 16 bit code
[ORG 0x7C00]    ;Origin, tell the assembler that where the code will
                                ;be in memory after it is been loaded

MOV SI, HelloString ;Store string pointer to SI
CALL PrintString        ;Call print string procedure
```

```
JMP $           ;Infinite loop, hang it here.


PrintCharacter: ;Procedure to print character on screen
        ;Assume that ASCII value is in register AL
MOV AH, 0x0E    ;Tell BIOS that we need to print one charater on screen.
MOV BH, 0x00    ;Page no.
MOV BL, 0x07    ;Text attribute 0x07 is lightgrey font on black background

INT 0x10        ;Call video interrupt
RET             ;Return to calling procedure



PrintString:    ;Procedure to print string on screen
        ;Assume that string starting pointer is in register SI

next_character: ;Lable to fetch next character from string
MOV AL, [SI]    ;Get a byte from string and store in AL register
INC SI          ;Increment SI pointer
OR AL, AL       ;Check if value in AL is zero (end of string)
JZ exit_function ;If end then return
CALL PrintCharacter ;Else print the character which is in AL register
JMP next_character      ;Fetch next character from string
exit_function:  ;End label
RET             ;Return from procedure


;Data
HelloString db 'Hello World', 0 ;HelloWorld string ending with 0

TIMES 510 - ($ - $$) db 0       ;Fill the rest of sector with 0
DW 0xAA55                       ;Add boot signature at the end of bootloader
```

Now once again compile the bootloader and write it into floppy disk and boot the system using it. You will be once again surprised to see 'Hello World' on screen.

Disclaimer

Author of this article is not responsible for anything that happens due to it. Please read all the instruction carefully and then try. Dont use your hard disk for booting your bootloader.