How do I generate code for real mode through an assembler?

When the CPU starts in Real Mode (16-bit), all we can do while booting from a device is to utilize the built in functions provided by the BIOS to proceed further. What I mean here is we can utilize the functions of BIOS to write our own boot loader code, and then dump into onto the boot sector of the device, and then boot it. Let us see how to write a small piece of code in assembler that generates 16-bit CPU code through GNU Assembler.

Collapse | Copy Code

```
Example: test.S
```

Collapse | Copy Code

```
.code16                 #generate 16-bit code
.text                   #executable code location
     .globl _start;
_start:                 #code entry point
     . = _start + 510   #mov to 510th byte from 0 pos
     .byte 0x55         #append boot signature
     .byte 0xaa         #append boot signature
```

Let me explain each statement in the code above.

- .code16: It is a directive or a command given to an assembler to generate 16-bit code rather than 32-bit ones. Why is this hint necessary? Remember that you will be using an operating system to utilize an assembler and a compiler to write boot loader code. However, I have also mentioned that an operating system works in 32 bit protected mode. So when you utilize assembler on a protected mode operating system, it's configured by default to produce 32-bit code rather than 16-bit code, which does not serve the purpose, as we need 16-bit code. To avoid assembler and compilers generating 32-bit code, we use this directive.
- .text: The .text section contains the actual machine instructions, which make up your program.
- .globl _start: .global <symbol> makes the symbol visible to linker. If you define symbol in your partial program, its value is made available to other partial programs that are linked with it. Otherwise, symbol takes its attributes from a symbol of the same name from another file linked into the same program.
- _start: Entry to the main code and _start is the default entry point for the linker.
- . = _start + 510: traverse from beginning through 510th byte
- .byte 0x55: It is the first byte identified as a part of the boot signature.(511th byte)
- .byte 0xaa: It is the last byte identified as a part of the boot signature.(512th byte)

How to compile an assembly program?

Save the code as *test.S* file. On the command prompt type the below:

- as test.S -o test.o
- ld –Ttext 0x7c00 --oformat=binary test.o –o test.bin

What does the above commands means to us anyway?

- as test.S –o test.o: this command converts the given assembly code into respective object code which is an intermediate code generated by the assembler before converting into machine code.
- The --oformat=binary switch tells the linker you want your output file to be a plain binary image (no startup code, no relocations, ...).
- The –Ttext 0x7c00 tells the linker you want your "text" (code segment) address to be loaded to 0x7c00 and thus it calculates the correct address for absolute addressing.

What is a boot signature?

Remember earlier I was briefing about boot record or boot sector loaded by BIOS program. How does BIOS recognize if a device contains a boot sector or not? To answer this, I can tell you that a boot sector is 512 bytes long and in 510th byte a symbol 0x55 is expected and in the 511th byte another symbol 0xaa is expected. So I verifies if the last two bytes of a boot sector are 0x55 and 0xaa and if it is then it identifies that sector as a boot sector and proceeds execution of the boot sector code or else it throws an error that the device is not bootable. Using a hexadecimal editor you can view the contents of the binary file in a more readable way and below is the snapshot for your reference when you view the file using the hexedit tool.

How to copy the executable code to a bootable device and then test it?

To create a floppy disk image of 1.4mb size, type the following on the command prompt.

- dd if=/dev/zero of=floppy.img bs=512 count=2880

To copy the code to the boot sector of the floppy disk image file, type the following on the command prompt.

- dd if=test.bin of=floppy.img