

The Boot Sector

Chris Lattner

The boot sector on a disk is always the first sector on the first track on the first head. When the computer is powered on (or reset), the BIOS starts up and does the POST. It initializes all of it's data, then it looks for a valid boot sector. First it looks at the A: drive, then it looks to C:. If it doesn't find it then interrupt 18h is called, which, on original IBM PCs, started the ROM BASIC. A valid boot sector (to the BIOS) is one that has 0AA55h at offset 510 in the boot sector.

When the BIOS finds the boot sector, it reads that sector (512 bytes) off of the disk and into memory at 0:7C00h. Then it jumps to 0:7C00h and the boot sector code gets control. At this point, all that has been initialized is the BIOS data area (40h:0) and the BIOS interrupts (10h - 1Ah). At this point, memory is mostly unused, but not necessarily cleared to 0.

Below is an example shell that I use when writing boot sector code:

```
;Generic boot sector shell.  Written by Chris Lattner 1995
;Code+Data MUST be less than 510 bytes long!

_Text SEGMENT PUBLIC USE16
  assume CS:_Text, DS:_Text
  org 0

EntryPoint:
  db 0EAh  ;jmp far SEG:OFS      ;Currently we are at 0:7C00
  dw OFFSET AfterData, 7C0h    ;This makes us be at 7C0:0

;Put any data here!

AfterData:
  push CS
  pop DS      ; update DS to be 7C0 instead of 0

;Put code here!

  jmp $      ; Hang out...

org 510      ; Make the file 512 bytes long
  dw 0AA55h  ; Add the boot signature
_Text ENDS
  END
```

To use this code, you must compile it with either MASM or TASM. Link it together as a COM file if you can (Tasm v4 complains about illegal COM entry point). Then write the 512 byte file in sector 1 of a floppy (With some suitable disk tool) to test it out... If you can't compile it as a COM file, compile it as an EXE, but only write out the last 512 bytes of the file (eg. skip the EXE file header). Pop the disk in, reset you computer, and watch the magic!