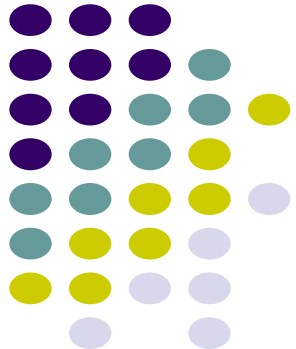
ECE469: Booting

Aravind Machiry

1/11/2024



Some content copied from: OSE 544

What is booting?

Booting an OS

- Start the OS and give it control:
  - Where is OS?
  - What is OS?
    - a computer program.
  - Giving control:
    - Start executing.

Booting an OS

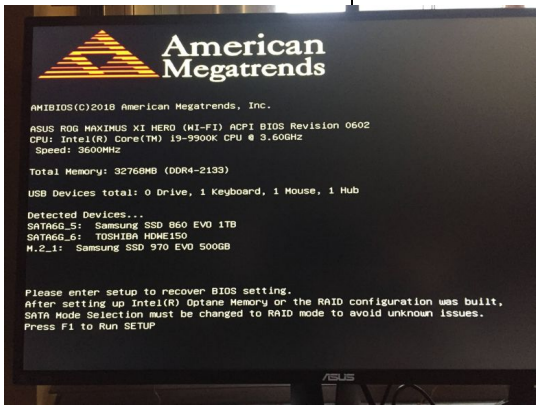
1. Initialize disk/peripherals.

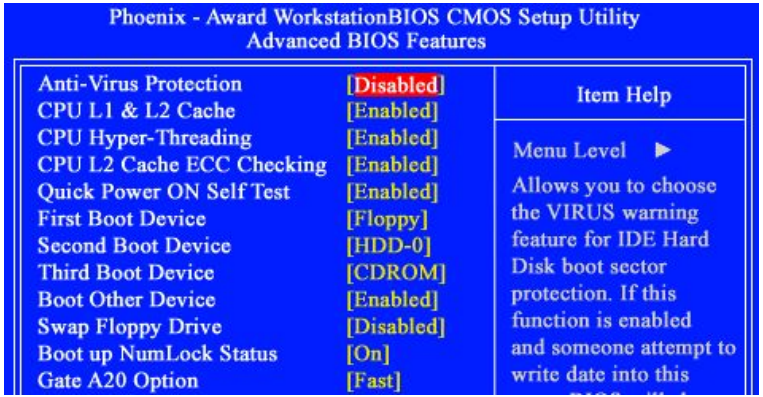
2. Read the OS code (i.e., binary) and load it into main memory.

3. Start executing from the first instruction.

What happens, when we turn on the machine?

1. BIOS:
   a. Basic Input Output System.
   b. Enables basic device access.





What happens, when we turn on the machine?

1. Power up.
2. BIOS initializes basic devices.
3. After initializing peripheral devices, it will put some initialization code to
   a. DRAM physical address 0xffff0 ([f000:fff0])
   b. Copy the code from ROM to RAM
   c. Run from RAM

```
The target architecture is assumed to be i8086
[f000:fff0]    0xffff0: ljmp   $0xf000,$0xe05b
0x0000fff0 in ?? ()
```

4. What does the code do? Load and run the boot sector from disk
   b. Read the 1st sector from the boot disk (512 bytes)
   c. Put the sector at 0x7c00
   d. Run it! (set the instruction pointer = 0x7c00)

What!!? Why?

1. What is ROM?
    a. Read Only Memory: Memory that contains read-only data -> code for BIOS.

1. What is i8086 and why is the address 0xffff0 ([f000:fff0])?

    b. Intel 8086 (1978, ~44 years old) -> The seed for Intel x86 processors.

    c. 16-bit processor; all registers are 16-bits.

    d. The processor starts at address 0xffff0 -> Hardcoded!

    e. **BIOS assumes that our processor is i8086! Why!?**

What is this [f000:fff0]?

- i8086 has 16-bit registers:
  a. We can access 1 MB of memory (i.e., 0x0 - 0xfffff) - 20 bit address space
  b. How?

- **Memory Segmentation**: Allows 16-bit processor to access 20-bit address space.

  a. Address Format = [Segment:Offset]

  b. Final address = (Segment * 16 + Offset) or (Segment << 4 + Offset)

  c. [f000:fff0] => 0xffff0

Real Mode v/s Protected Mode

- Real mode:
  a. Mode that uses physical memory directly.
  b. No memory protection.
  c. MS-DOS (1981 ~ 2000) runs in this mode.

- Protected mode (Modern processors):
  a. Uses virtual memory -> gets translate to physical memory by page tables.
  b. Memory protection through MMU.
  c. All modern operating systems run in this mode.

- Booting always occur in real mode. Why?

Debugging notes

```
[f000:e05b]    0xfe05b: cmpl   $0x0,%cs:0x6ac8
0x0000e05b in ?? ()
```

```
cs 0x0000f000
```

what are we comparing 0x0 with?

Debugging notes

```
[f000:e05b]    0xfe05b: cmpl   $0x0,%cs:0x6ac8
0x0000e05b in ?? ()
```

```
cs 0x0000f000
```

what are we comparing 0x0 with?

cs:0x6ac8

f000:6ac8 == 0xf6ac8

```
>>> x/w 0xf6ac8
0xf6ac8:        0x00000000
```

Boot from disk

- Load the boot sector (512 bytes) from the boot disk

- Boot sector (Master Boot Record)
  a. The 1st sector of the disk partition
  b. Ends with 0x55AA: **Let's check!**

- Load OS at 0x7c00, and run
  a. Now the OS takes the control!

What should boot sector do?

- Load the OS and run!
  a. Processor maximum memory in real mode: 1MB.
     i. OS size can be more than 1MB!!?

What should boot sector do?

- Load the OS and run!
  a. Processor maximum memory in real mode: 1MB.
     i. OS size can be more than 1MB!!?

- First, enable Protected Mode (virtual memory support 4GB).

- Then load the OS and run it.

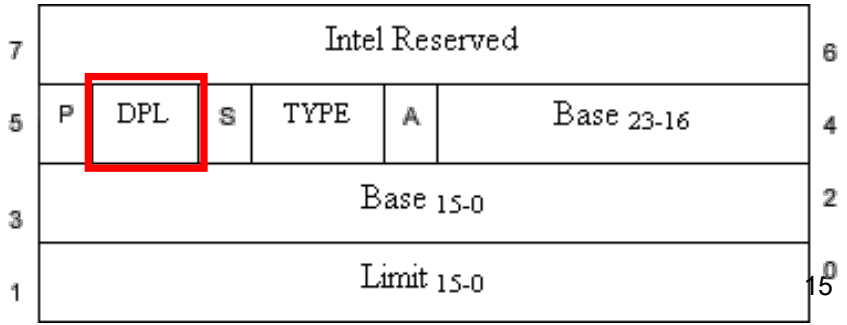
- Boot sector is 512 bytes, but we should do this in the first 510 bytes!!? Why?

Intel memory models

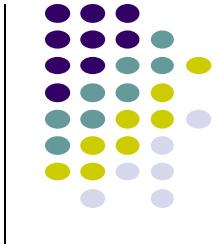
- 8086 (1978, 16-bit), 8088 (1979, 8-bit), and 80186 (1982, 16-bit)
  a. Uses 20-bit addressing via **Real Mode** segmentation

- 80286 (1982), a 16-bit computer
  a. Uses 24-bit (16MB) addressing via **Protected Mode**
  b. A different way of using segment registers (286 is also 16-bit computer)
  c. Segment register points to Global Descriptor Table, which sets base (24-bit) and limit (16-bit)

Base (24-bit) + Limit (16-bit)



i386 Protected Mode



- 80386 (1985, 32-bit)
  a. 32-bit processor, all registers are 32 bits, 2^32 = 4,294,967,295 = 4GB Space!
  b. At that time major computers were equipped only with 4~16MB RAM…
  c. Segment register now points 32bit base addressable by 32bit offset

- 32bit base + 20bit limit

- Supports paging (Lab2)

<table>
<tr><td colspan="3">31 16</td><td colspan="2">15 0</td></tr>
<tr><td colspan="3">Base 0:15</td><td colspan="2">Limit 0:15</td></tr>
<tr><td>63 56</td><td>55 52</td><td>51 48</td><td>47 40</td><td>39 32</td></tr>
<tr><td>Base 24:31</td><td>Flags</td><td>Limit 16:19</td><td>Access Byte</td><td>Base 16:23</td></tr>
</table>

i386 Protected Mode

- 80486, Pentium (P5), Pentium II (i686, P6), Pentium !!!
  a. Uses the same protected mode with 80386

- Pentium 4 (Prescott, 2004)
  a. Supports 64-bit (amd64)
  b. Address space: 48-bit (256TB)

- Latest (Coffee Lake and onward)
  a. Address space: 57-bit (128PB)

Summary!

Map code in BIOS at f000:fff0

Extended Memory (Over 1MB)

Load kernel and run!

BIOS
0xf0000 ~ 0x100000
(960KB ~ 1MB)

Devices
0xc0000 ~ 0xf0000
(768KB ~ 960KB)

VGA
0xa0000 ~ 0xc0000
(640KB ~ 768KB)

Low Memory
0x00000 ~ 0xa0000
(0 ~ 640KB)

Enabling Protected Mode

Read Master Boot Record (MBR)
from the boot disk
and load it at 0x7c00