K-OS Boot Sector

Kai Chen

CS134c Spring 2003

Overview

- Test CPU
- Load GDT
- Enable A20 Address Line
- Load Kernel
- Switch to protected mode
- Flush registers and execute the kernel

Boot process

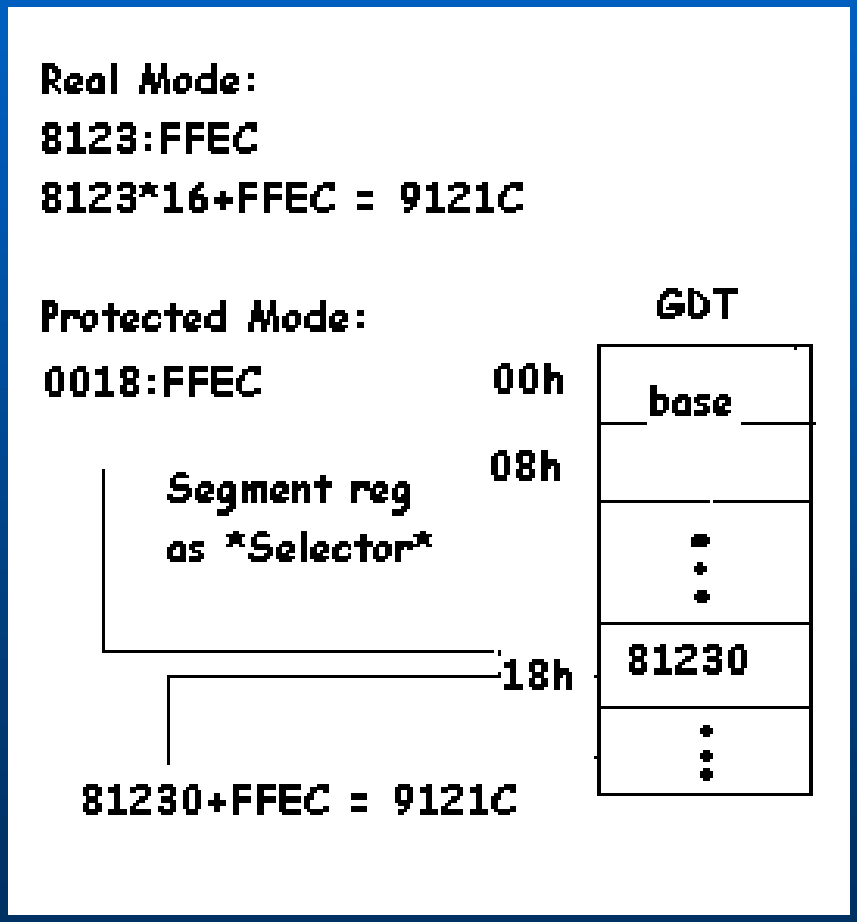
- **BIOS Power-On Self Test (POST)**
- **Select boot device**
  - **Floppy, hard drive, CD …**
  - **512 bytes, ending with 0xAA55**
- **Copies to memory location 0x7C00**

Test CPU

- **Test the flags**
- **Bits 12 - 15 set : 8086/8088**
- **Bits 12 - 15 clear : 80286**

Protected Mode Basics

- **Real Mode**
  - 16-bit addressable
  - seg:off =seg*16+off
- **Protected Mode**
  - 32-bit addressable
  - segment register as “selector”



Segment Register Basics

- **Really 4 registers**
  - Selector
  - Base
  - Limit
  - Attributes
- **Real mode**
  - value -> selector
  - value*16 -> base
- **Protected mode**
  - a descriptor is fetched
  - unpacked into 4 registers

Proteced mode segment register:

-- A Simple view

Used as *selector*

b15 ---------- b2 b1 b0

Index	L	RPL

Index: Selects the GDT descriptor
L: 0 - use GDT
1 - use LDT
RPL: Requestor Priviledge Level (Ring)

Global Descriptor Table

- A vector of 8-byte descriptors
  - 32-bit base
  - 20-bit limit
  - 12-bit segment type
- Protected mode segments
- Protection
  - Faults if off limit

Setting up GDT

- **GDT Structure:**

```
gdtr:
    dw GDTLimit          ; gdt size - 1
    dd GDTLBase          ; starting address
null_sel:                ; null selector, each entry 0
    …                    ; 8 bytes for each descriptor
code_sel:                ; code segment, read/exec
    …
data_sel:                ; data segment, read/write
    …
```

- **Loading GDT Register:**
  - **LGDT [gdtr]**

A20 Address Line

- **A20 refers to the 21st address line.**
- **For programs with 1 Mbytes memory or less, 20 address lines (A0 - A19) are sufficient. (Real mode is only 16-bit addressable.)**
- **32-bit addressable in protected mode. Wrap around without A20.**

Enable A20

- **Talk to Keyboard controller**
- **Communicate at port 0x60, 0x64**

Loading Kernel

- BIOS interrupt 0x13, on error CF set.
- Sets up registers properly
  - AH -- BIOS function. (0x02)
  - AL -- Number of sectors to read
  - ES:BX -- segment:offset, memory location
  - CH -- Track number
  - CL -- Starting sector
  - DH -- Head number
  - DL -- Drive number

Switching to Protected Mode

- **Easy! Set bit 1 in CR0**

```
MOV     EAX, CR0
OR      AL, 1
MOV     CR0, EAX
```

- **Now in 32 bit protected mode**
  - **Do a few jumps to empty prefetched queue**
  - **Do a far jump on CODE_SEL to set CS/IP/EIP**
  - **Use DATA_SEL to flush other segment registers.**

Execute the Kernel

- **Compile kernel into plain binary**
- **Our boot sector loads kernel to some memory location (0x10000 for K-OS). Need to specify this during linking/loading.**
- **Far jump CODE_SEL:KERNEL_ADDR to run!**