COS318 Project 1
Bootloader
Project Overview

• Bootloader: bootblock.s
  – Understand how the PC boots
  – BIOS, X86 assembly language (tutorial on Thu)

• Create an image for booting: createimage.c
  – Understand ELF format
  – Read files in ELF format and extract the necessary information

• Test: Bochs first, then boot off a USB disk

• Extra credit: loading larger kernels
Bootloader

- There’s nothing in the memory once turning on the machine
  - Resort to the hardware
- The BIOS is loaded
  - Typically doesn’t know anything about the OS
  - Minimal functionality
- Everything starts from here
Bootloader

• The BIOS starts at 0xFFFFF0
  – Self check, initialization, search for boot devices
  – Load the first sector (512 bytes) of a boot device to 0x7C00
  – Jump to 0x7C00
Bootloader

- Load the kernel
- Set up kernel stack
- Jump to the kernel
Read from Disk to Memory

• BIOS Interrupt 0x13, function 2: Disk - Read sector(s) into memory
  – %ah=0x02, function 2
  – %al=#sectors to read (must be nonzero)
  – %ch+bits 6 7 of %cl=cylinder number
  – %cl=sector number (bits 0-5)
  – %dh=head number
  – %dl=drive number (has been set on entering)
  – %es:%bx->data buffer
  – int $0x13

• Details will be covered in the tutorial on Thursday
Createimage

• After compiling and linking

• What we want

![Diagram of ELF format file and disk image]

- ELF header
- Program header table
  - Program header 1
  - Program header 2
  - Program header 3

An ELF format file 1

Disk image

- Segment 1
- Segment 2
- Segment 3

Bootloader
Kernel
Createimage

• Study the ELF format
  – ELF header: Elf32_Ehdr
  – Program header: Elf32_Phdr
• Padding up to a complete sector (512 bytes)
• Mark the image to be bootable
  – Write 0x55 0xAA to the end of the first sector
• Compare your implementation to the given createimage.given
  – Implement --extend to print information
  – Ignore --vm
Test

• Use Bochs to do the simulation
  – Installed in the fishbowl machines
  – bochs: run
  – /u/318/bin/bochsdbg: debug

• USB boot off
  – On the fishbowl machines: cat image > /dev/sdf
  – Boot from a USB disk on the fishbowl machines
Extra Credit

• Load larger kernels
  – Relocate the bootloader
  – Read data from more than one head/cylinder
    • Get the device parameters: #max head, #max sector
  – Deal with the cross physical segment reading
    • Data are read to %es:%bx

• Read the kernel sector by sector from the disk
Get Drive Parameters

- BIOS Interrupt 0x13, function 8: Disk – Get drive parameters
  - %ah=0x08, function 8
  - %dl=drive number
  - int $0x13, then if successful you will get:
    - %ch+(bits 6 7 of %cl)=maximum cylinder number, 0-based
    - %cl=maximum sector number (bits 0-5), 1-based
    - %dh=maximum head number, 0-based
Design Review

• Monday, 9/24 from 10:30am to 10:30pm, signup online

• Answer questions listed in the project description briefly. No more than 10 mins!
Print characters and strings

- Code based on the given bootblock.s
- Refer to bootblock_example.s
- Use BIOS Interrupt 0x10 function 14
  - %ah=0x0E, function 14
  - %al=character to be printed
  - %bh=active page number (use 0x00)
  - %bl=foreground color (use 0x02)
  - int $0x10
Something else...

- Tutorial of assembly language and bochs debugging will be on Thursday, 9/20
- Use Piazza to ask question (except personal or private issues), I will be generally monitoring through the whole project
- We are working on an OS image that can be used via VirtualBox, so as to relieve the workload of fishbowl machines