Precept 1: Bootloader

COS 318: Fall 2019

Project 1 Schedule

- **Design Review:** Monday 9/23 & Tuesday 9/24, 3:00pm - 7:00pm
- **Precept:** Monday 9/23 & Tuesday 9/24, 7:30pm - 8:20pm
- **Due:** Sunday 9/29, 11:55pm

Project 1 Overview

- Write a bootloader: **bootblock.s**
  - How to set up and start running the OS
  - Written in x86 assembly (AT&T syntax)
- Implement a tool to create a bootable OS image: **createimage.c**
  - Bootable OS image contains bootloader and kernel
  - How are executable files structured?
  - Become familiar with ELF format

General Suggestions

- Read **assembly_example.s** in start code pkg *\/u/318/code/project1*
- Get **bootblock.s** working before starting on **createimage.c**
- Read documentation on ELF format
- If you haven’t already started, *start now*

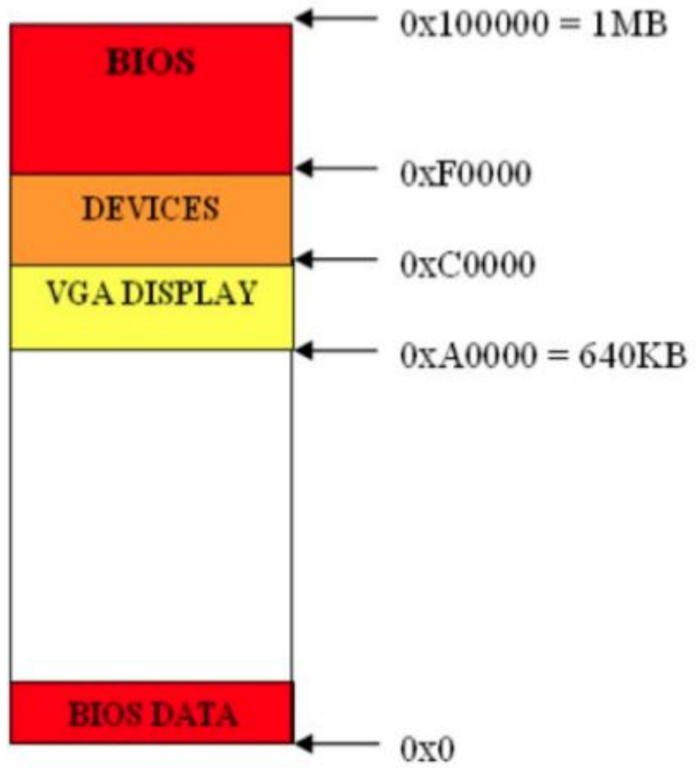
Segment Registers

- Set **%cs** as needed in BL, zero it for the kernel
- Bootloader linked with offset of 0x0
- Kernel linked with offset of 0x1000
    - **%ds** must be set to 0x0

Bootloader

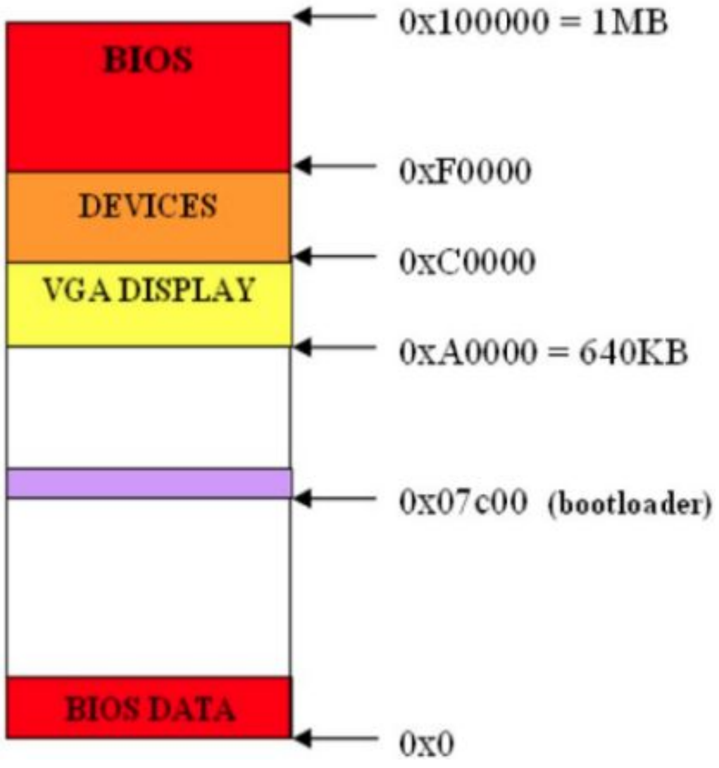
Boot Process

- Nothing in RAM on startup:
  - Load BIOS from ROM
  - BIOS loads bootloader from disk
  - Bootloader loads the rest



Loading the Bootloader

- Find bootable storage device (HDD, USB, etc.)
- Load first disk sector (MBR) into RAM at 0x7c00
- Switch control to this location

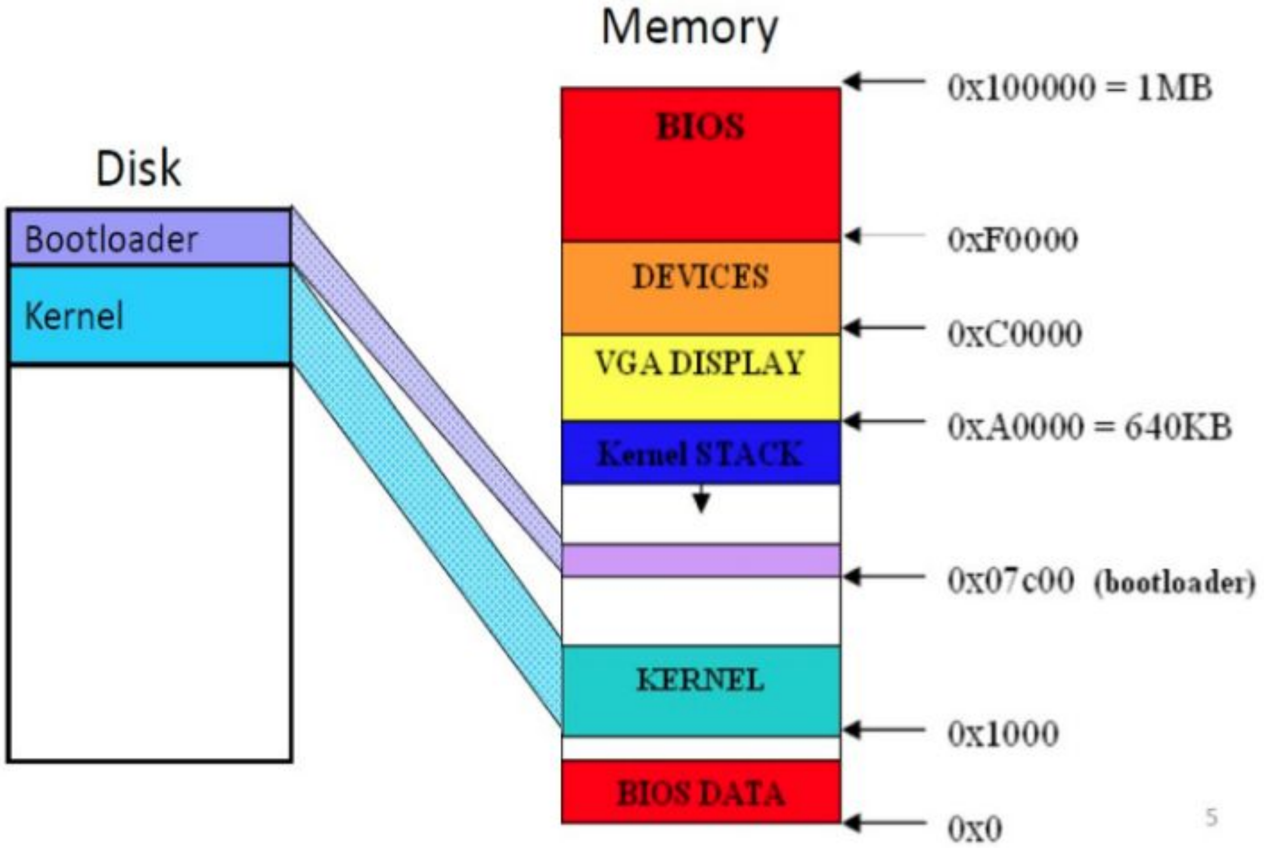


Master Boot Record

- First sector of a hard disk
    - Beginning: bootloader code
    - Remaining part: partition table
- BIOS sets %dl to the drive number
- For more info: see MBR and Partition Table

Bootloader Tasks

1. Load kernel into memory
2. Setup kernel stack
3. Transfer control to kernel



BIOS Services

- Use BIOS services through INT instruction
  - Store the parameters in the registers
  - Triggers a software interrupt
- **int $INT_NUM**
  - int $0x10 # video services
  - int $0x13 # disk services
  - int $0x16 # keyboard services

BIOS INT 0x13

- Function 2 reads from disk
  - %ah: 2
  - %al: Number of sectors to read
  - %ch: Cylinder number (bits 0-7)
  - %cl: Sector number (bits 0-5); bits 6-7 are bits 8-9 of the cylinder number
  - %dh: Starting head number
  - %dl: Drive number
  - %es:%bx: Pointer to memory region to place data read from disk
- Returns
  - %ah: Return status (0 if successful)
  - Carry flag = 0 if successful, 1 if error occurred
- For more information:
  - https://en.wikipedia.org/wiki/Cylinder-head-sector

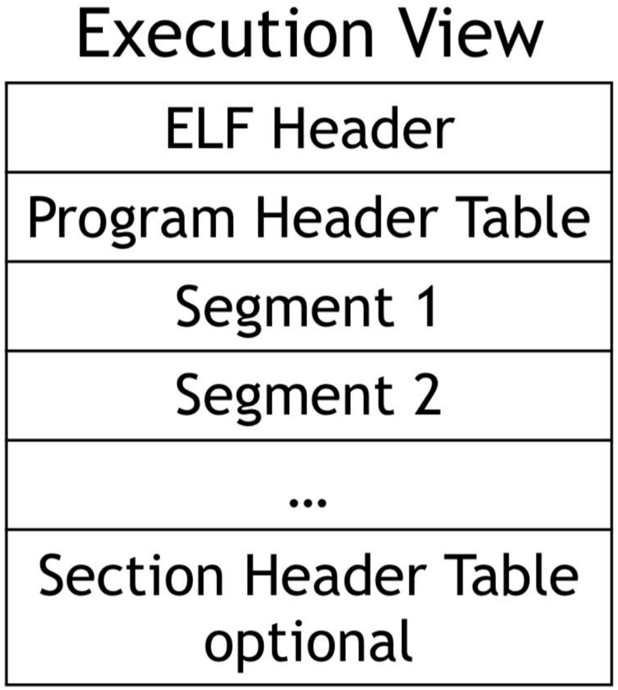
Createimage + ELF

ELF Format

- Executable and linking format
- Created by assembler and link editor
- Object file: Binary representation of programs intended to execute directly on a processor
- Supports various processors/architectures
- Represents control data in a machine-independent format

ELF Object File Format

- Header (pp. 1-3 to 1-5)
  - Beginning of file
  - Roadmap, file organization
- Program Header Table (p. 2-2)
  - Array, each element describes a segment
  - Tells system how to create the process image
  - Files used to create an executable program must have a program header



p. 1-1 in the ELF manual

ELF Useful Tools

- **objdump**: Display information from object files
  - Read manual page (*man objdump*)
- **hexdump**: Display file contents in hexadecimal, decimal, octal, or ascii
  - Read manual page (*man hexdump*)

Questions?

USB Live Booting

- Bootloader needs to work on USB as well
  - Bochs provides cleaner environment than USB - don’t make any assumptions!
- Rooms on the left side of Friend 010 Lab - for COS 318
  - 011 Code: 4620,   012 Code: 46283
  - Put in USB, Power on machine, Mash F12, Hit F1 to Continue, Select USB from boot devices, Celebrate (or cry)