**Operating Systems**

**CS4348**

**Project #3 (Alternative): Exploring the Booting Procedure**

**Due Date: Saturday, November 30, 2019**

## I. Project Organization

In this project you will explore the booting procedure of an X86 system.

You should do the following pieces to complete your project. Each piece is explained below:

* Code 60 points
* Output 30 points
* Summary 10 points

# Code

Your code should be nicely formatted with plenty of comments. The code should be easy to read, properly indented, employ good naming standards, good structure, etc.

# Output

Output will be graded by running your program.

# Summary

The summary section should discuss your project experience. You should discuss how you approached the project, including anything difficult or interesting, what was learned, and the end result. This should be at least 1 page in length.

## II. Project Description

**Language/Platform/Approach**

This project requires X86 assembly language for all of its code.

You will use the NASM assembler which is already found on cs3.  
You will use the QEMU emulator which is also found on cs3.  
Both of these can be downloaded and run on your local machine, although they are large.

### Problem Overview

The project involves writing the code necessary to boot the system into 32-bit protected mode. The first step is to write code for the boot sector. This code will load another sector and jump to it. This code will transition the system from 16-bit real mode to 32-bit protected mode. It should output to the screen to demonstrate that 32-bit mode has been reached, and it should activate a timer interrupt which will also output to the screen to demonstrate that interrupts are working.

**Details**

You will need to research the steps required to fulfill the requirements. Here are some important details.

You should assume you are booting from a floppy disk.

The boot sector is always loaded to address 0x7C00.

You can set the stack pointer to 0x9000 for 4k in size (it grows down towards address zero).  
The next sector should be loaded at address 0x7E00. Note that sectors are 512 bytes.  
The system begins in 16-bit real mode. To read a disk sector, use BIOS call 13h.  
To print something, use BIOS call 10h.

The code at 0x7E00 should setup a GDT and IDT.   
You can place these anywhere (for example, address 0x500 and 0x1000).  
The GDT needs at least 3 entries: null, kernel code, and kernel data.

You will need to initialize the PIC and perform PIC remapping.  
Setup the timer interrupt at entry 32. The timer handler should just output a message to show it ran.  
After jumping to 32-bit mode, output something to the screen to show this was successful.  
In 32-bit mode, the BIOS calls are not available, so you will need to output to the video memory.

**Expected Output**

(Note: the comments beside each line are not part of the output)

Boot sector ; this line is printed by the boot sector code

Next sector ; this line is printed by the code that the boot sector loads

32-bit mode ; this is printed after jumping successfully to 32-bit mode

Interrupt handler ; this line is printed by the timer interrupt

## III. Project Guidelines

### Submitting

Submit your project on eLearning. Include in your submission the following files:

1. readme.txt. A readme file describing how to run your project
2. summary.doc A Word document for the summary
3. Your source files

### Academic Honesty

All work must be your own. If cheating is suspected, you will be referred to the Judicial Affairs Office for further discussion. Copying may be detected in a number of ways, including by software which compares your code with all other students’ source code, by comparison with code on the Internet, or by a visual inspection of your source code.

### Resources

This link is a very good description of how to write the boot sector code:

### <http://mikeos.sourceforge.net/write-your-own-os.html>

This tutorial is a very good resource for booting into 32-bit protected mode:

<http://www.independent-software.com/operating-system-development.html>

You will need to research how to do some things that not found in these tutorials.