CS 310 Paging Lab

Spring 2020

February 25, 2020

1 Getting Started

This activity will prepare you for doing Homework 4. The code that you write in this activity will run **on the bare metal, not in Linux**, meaning that you can't call C library functions (unless you write them yourself).

1. Go to the course website and clone the homework 4 repo.
2. It contains some start code and an IDE disk driver.
3. Create a new C file in the `src` directory called `main.c` and put a blank `main()` function in there.
4. Open up the `Makefile` and add `main.c` to the `OBJS` list so that it will compile.

You may also want to copy your terminal driver from Homework 2 into the `src` directory and add it to the `OBJS` list. This will allow you to print stuff to the screen.

2 Getting `printf` Working

Your terminal driver supports only printing single characters and unformatted strings. You probably also want to print formatted strings (including numbers, etc.). On the course website there's a file called `rprintf.c` which works just like the regular C `printf` function, except you also have to pass a pointer to your `putc` function. You can use it like this:

```
esp_printf(putc, "the value of i is %d\n", i);
```

The first argument to `esp_printf` is a pointer to your `putc` function which prints a single character to the screen. If you want to use it, just add `rprintf.o` to your `OBJS` list in your Makefile.

3 Reading FAT Data Structures from Disk

Below I have pasted a data structure that lists all the fields in the boot sector and the BIOS information block. Your job is to read one disk sector into memory (by calling the IDE driver) and parse the fields. You can read blocks from the disk drive by calling the IDE driver included in `ide.s`:

```
#include "ide.h" // See this file for function prototype of ata_lba_read()

char bootSector[512]; // Allocate a global array to store boot sector

int main() {
  ata_lba_read(0, bootSector, 1); // Read sector 0 from disk drive into bootSector array

  // Print out some of the elements of the BIOS information block using rprintf...
}
```

```
// Boot sector data struct. Put this either above main() or in its own header file.

struct boot_sector {
    char code[3];
    char oem_name[8];
    uint16_t bytes_per_sector;
    uint8_t num_sectors_per_cluster;
    uint16_t num_reserved_sectors;
    uint8_t num_fat_tables;
    uint16_t num_root_dir_entries;
    uint16_t total_sectors;
    uint8_t media_descriptor;
    uint16_t num_sectors_per_fat;
    uint16_t num_sectors_per_track;
    uint16_t num_heads;
    uint32_t num_hidden_sectors;
    uint32_t total_sectors_in_fs;
    uint8_t logical_drive_num;
    uint8_t reserved;
    uint8_t extended_signature;
    uint32_t serial_number;
    char volume_label[11];
    char fs_type[8];
    char boot_code[448];
    uint16_t boot_signature;
}__attribute__((packed));
```