# ECE469: Operating System Engieering

This is a course focus on Operating System, Adapted from MIT6.828, using JOS as primary lab materials.

## Note

**PC's physical address space.**

```
+------------------+  <- 0xFFFFFFFF (4GB)
|      32-bit      |
|  memory mapped   |
|     devices      |
|                  |
/\/\/\/\/\/\/\/\/\/\

/\/\/\/\/\/\/\/\/\/\
|                  |
|      Unused      |
|                  |
+------------------+  <- depends on amount of RAM
|                  |
|                  |
| Extended Memory  |
|                  |
|                  |
+------------------+  <- 0x00100000 (1MB)
|     BIOS ROM     |
+------------------+  <- 0x000F0000 (960KB)
|  16-bit devices, |
|  expansion ROMs  |
+------------------+  <- 0x000C0000 (768KB)
|   VGA Display    |
+------------------+  <- 0x000A0000 (640KB)
|                  |
|    Low Memory    |
|                  |
+------------------+  <- 0x00000000
```

**BIOS**

- 16 bit real mode, [CS:IP] format
- Basic IO initialization
- Ends with magic byte 0x55AA

**Bootloader**

- 32bit protected mode
- Use virtual memory, index using GDT (global descriptor table)

**ELF**

- Defined at `inc/elf.h`
- Contains loading information
- `objdump -h obj/boot/boot.out` and `objdump -h obj/kern/kernel` to observe section header info for boot and kernel
- `.text`: the program's executable instructions
- `.rodata`: Read-only data
- `.data`: data section holds the program's initialized data
- `.bss`: section reserved for uninitialized global variable

**Kernel**

- `kern/entrypgdir.c` map virtual memory in the kernel, once paging enabled in `kern/entry.S`

**OS initialization process**

- BIOS Phase
    - add later
- Bootloader Phase
    - boot/boot.S + boot/main.c
    - Switch processor mode
    - Read 8 disk sectors to get ELF header data
    - Load segments from disk to mem, which contains actual kernel code + data
    - Jump to kernel entry, 0x10000C

# Side Note

- `VMA`: linked address. Execution assumes section is here
- `LMA`: load address. where section is placed when loaded
- `.out` file: linked executable
- gdb: use `symbol-file` to let gdb know the function lines, calls, etc
- gdb: use `x/10x addr` to examine 10 words from adder
- `CR0`: control register 0; `CR0_PG`: paging enable; `CR0_PE`: enable protected mode; `CR0_WP`: write protect, cannot write to RO pages.

# Lab Exercise Answer

**EX6**

They are different. I think may because at bootloader, the kernel isn't being loaded to LMA, because the 8 sectors needs to be loaded in the for loop in `boot/main.c`

**EX7**

After paging enabled, 0xF0100000 also has information at LMA. I think the problem will be at relocated tag, at line 74 in `kern/entry.S`, because it will relocate execution to VMA