OS Homework 1

10302010023 Wang Xin

## Exercise 2

From the single step trace into the BIOS instruction, we can assume that the BIOS instruction could probably execute the following steps:

1. When QEMU is powered on, the CS is set to 0xF000, the IP is set to 0xFFF0, the address 0xFFFF0 is at the top of the region of BIOS, and hence the BIOS need to jump to the front area of the BIOS ROM region.
2. From the first several steps, we can get some typical steps like “in 0x71 %al”(Figure 1), with reference to ***The more common I/O address assignments,*** we can guess that the BIOS will configure system devices such as real time clock, I/O port, page registers, interrupt descriptor table, etc..

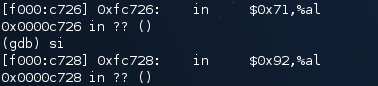


Figure 1

1. After initializing the PCI bus and important devices, the BIOS would search the bootable devices, in this case, the hard disk, the BIOS loads the boot program from the first sector of hard disk into memory at physical address 0x7C00 and give control to the boot loader program.

## Exercise 3

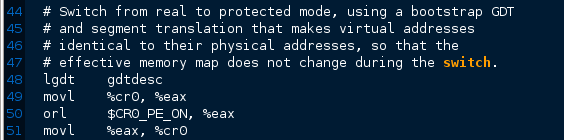
1. When the processor switches from BIOS to Boot Loader, check the assembly code in boot/boot.S (Figure 2)

Figure 2

In Line 49-51, the register %cr0 was changed to $CR0\_PE\_ON (protected mode enable flag), which indicates that the processor has entered 32-bit protected mode.

1. The last instruction the boot loader executes is to call the entry point from the ELF header. This instruction does not return for the control was moved to kernel (Figure 3).



Figure 3

The first instruction of the kernel is “movw $0x1234,0x472”(Figure 4)



Figure 4

1. In the dump file of kernel.asm, we can find that the first instruction is at the address 0xf010000C (Figure 4)
2. With reference to the dump file of obj/kern/kernel, we can get a section header table, which tells the boot loader the number of sections (Figure 5).

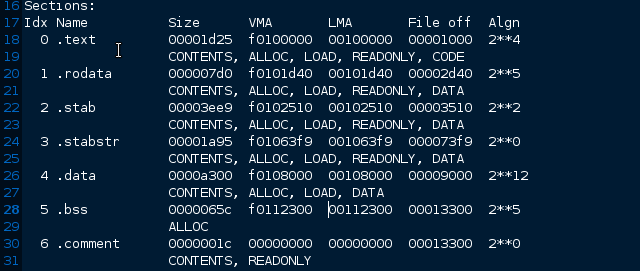
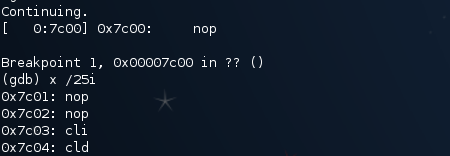


Figure 5

## Exercise 5



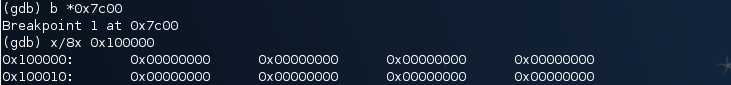
Figure 6

We have changed the –Ttext link address from 0x7C00 to 0x7C04, and then we boot the recompiled boot loader. As is shown, the loader meets an interrupt at address 0x7C2D (Figure 6). Then we check the dump file of the boot/boot.out and find that the VMA and LMA would be changed to the nearest upper 4-alignment address:.For instance, if we change the –Ttext address to 0x7C01, the boot loader would insert three “nop” instructions at the beginning of the boot sector(0x7C00) to make the instruction 4-aligned(Figure 7). Hence the link address and the load address of the boot loader would be different. Therefore any position-dependent instruction would raise a segmentation fault. Check the boot/boot.S, we can find that the first address-relative instruction was at address 0x7C2D, which means to jump to the next instruction (in 32-bit mode). Hence this instruction would fetch a wrong instruction and raise a fault. 

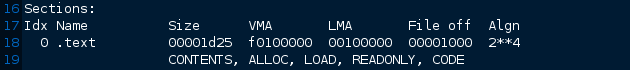
Figure

## Exercise 6

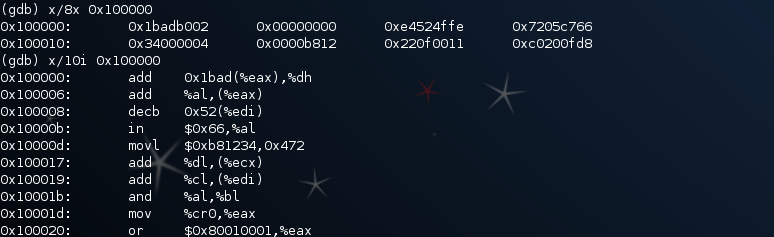
When the BIOS enter the boot loader, the kernel has not yet been loaded, thus the 8 words at 0x100000 are zero (Figure 8). When the boot loader enters the kernel, it would load the kernel at the address 0x100000. Check the dump file of the kernel (Figure 9), we can find that at the beginning of the kernel ELF file is the .text section, hence the eight words at 0x100000 would be the code text of the executable instructions (Figure 10).



Figure



Figure



Figure