Exercise 5. Trace through the first few instructions of the boot loader again and identify the first instruction that would "break" or otherwise do the wrong thing if you were to get the boot loader's link address wrong. Then change the link address in boot/Makefrag to something wrong, run make clean, recompile the lab with make, and trace into the boot loader again to see what happens. Don't forget to change the link address back and make clean again afterward!

When you jump to the next instruction at ljmp $PROT\_MODE\_CSEG, $protcseg

Notes / vocab:

ELF(Executable and Linkable Format)

Linker computes the memory layout of a program

Link address (VMA): The link address of a section is the memory address from which the section expects to execute

Load address(LMA): The load address of a section is the memory address at which that section should be loaded into memory.

Typically, the link and load addresses are the same.

The boot loader uses the ELF *program headers* to decide how to load the sections.

The BIOS loads the boot sector into memory starting at address 0x7c00, so this is the boot sector's load address. This is also where the boot sector executes from, so this is also its link address. We set the link address by passing -Ttext 0x7C00 to the linker inboot/Makefrag, so the linker will produce the correct memory addresses in the generated code.

**Exercise 6.** We can examine memory using GDB's **x** command. The [GDB manual](https://sourceware.org/gdb/current/onlinedocs/gdb/Memory.html) has full details, but for now, it is enough to know that the command **x/*N*x *ADDR*** prints *N* words of memory at *ADDR*. (Note that both 'x's in the command are lowercase.) *Warning*: The size of a word is not a universal standard. In GNU assembly, a word is two bytes (the 'w' in xorw, which stands for word, means 2 bytes).

Reset the machine (exit QEMU/GDB and start them again).

Examine the 8 words of memory at 0x00100000 at the point the BIOS enters the boot loader, and then again at the point the boot loader enters the kernel.

Why are they different? What is there at the second breakpoint? (You do not really need to use QEMU to answer this question. Just think.)

The 8 words of memory when the BIOS enters the boot loader are all zeroes (nothing loaded yet). When the boot loader enters the kernel the memory contains the first few instructions from the kernel.

Notes / Vocab:

e\_entry. This field holds the link address of the *entry point* in the program: the memory address in the program's text section at which the program should begin executing. You can see the entry point: athena% **objdump -f obj/kern/kernel (0x0010000c)**