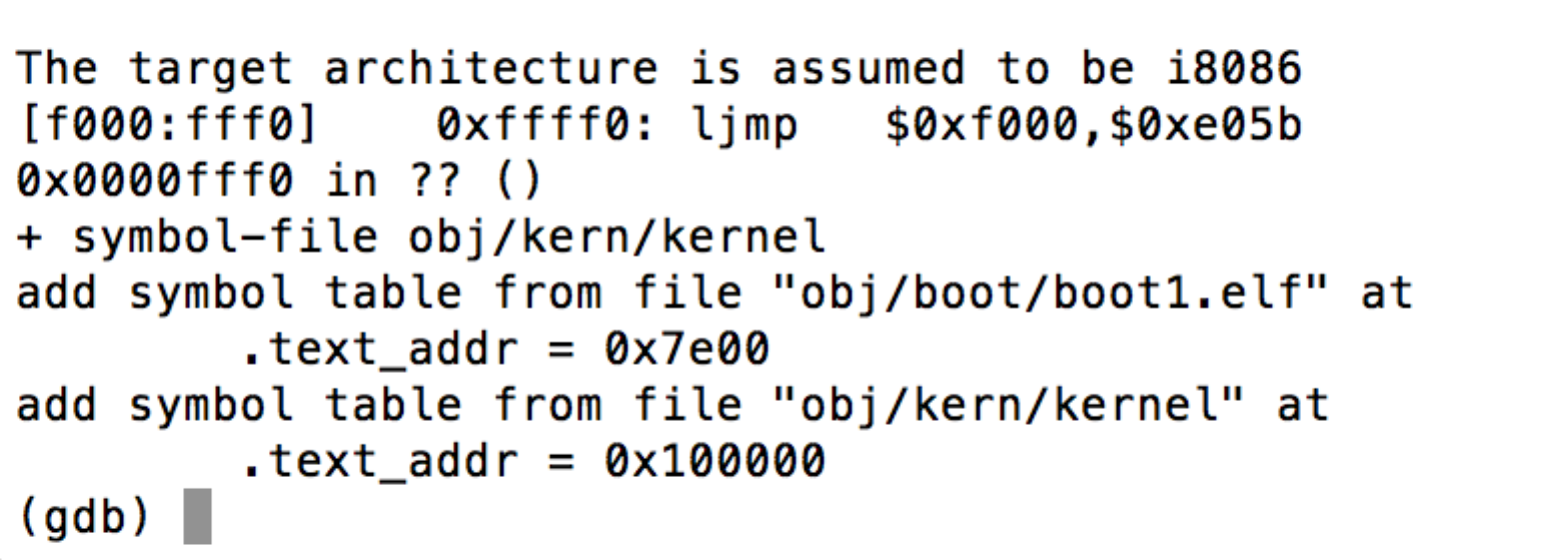
1. Testcase: MATIntro.c -> pageindex >= (1<<20)



e05b = 57435

fff0 = 65520

**Question A**

1. At what point does the processor start executing 32-bit code? What exactly causes the switch from 16- to 32-bit mode?

->When the real **boot loader** (in lab1, boot1 (load from sector 2 - 63 by boot0 ???)) start to function and switch the process from real mode to 32-bit protected mode.

\*Instruction:

movl %eax, %cr0 (eax = 1)

ljmp $PROT\_MODE\_CSEG, $protcseg

1. What is the *last* instruction of the boot loader executed, and what is the *first* instruction of the kernel it just loaded?

* last instruction:in function exec\_kernel
* jmp \*%edx (kernal’s entry point)
* first: /\* tell BIOS to warmboot next time \*/
* movw $0x1234,0x472

1. *Where* is the first instruction of the kernel

.globl start

Vma: (link address) 00100000

1. How does the boot loader decide how many sectors it must read in order to fetch the entire kernel  from disk? Where does it find this information?

#define ELFHDR ((elfhdr \*) 0x20000)

readsection((uint32\_t) ELFHDR, SECTOR\_SIZE \* 8, 0, dkernel);

-> #define ELFHDR ((elfhdr \*) 0x20000) in boot1main.c

**Question B**

1. Explain the interface between dprintf.c and console.c . Specifically, what function does console.c export? How is this function used by dprintf.c ?

2. Explainthefollowingfromvideo.c: