练习3：分析bootloader进入保护模式的过程（要求在报告中写出分析）。

BIOS将通过读取硬盘主引导扇区到内存，并转跳到对应内存中的位置执行bootloader。请分析bootloader是如何完成从实模式进入保护模式的。

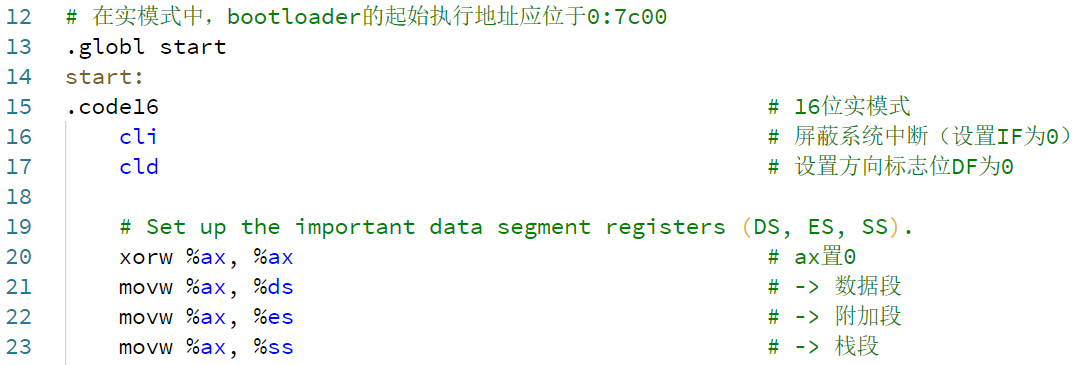
提示：需要阅读小节“保护模式和分段机制”和lab1/boot/bootasm.S源码，了解如何从实模式切换到保护模式。

答：

bootloader 工作的实现文件在lab1/boot 目录下，包含以下三个文件：

* bootasm.S —— 定义并实现了 bootloader 最先执行的函数 start ，此函数进行了一定的初始化，完成了从实模式到保护模式的转换，并调用 bootmain.c 中的 bootmain 函数；
* bootmain.c —— 定义并实现了 bootmain 函数，实现了通过屏幕、串口和并口显示字符串。 bootmain 函数加载 uCore 操作系统到内存，然后跳转到 uCore 的入口处执行；
* asm.h —— 是 bootasm.S 汇编文件所需要的头文件，主要是一些与 x86 保护模式的段访问方式相关的宏定义。

经过练习2可以知道，CPU加电后执行的0x7c00处的代码即为bootasm.S中的代码，因此该文件中的代码被最先执行。阅读代码的源文件，可以看到，在代码的第23行之前，进行了一些初始化置0的操作，如下图所示：

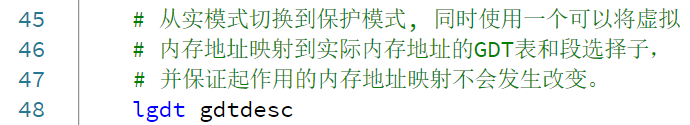


实模式下（16位模式），只能访问1MB物理内存空间。必须通过修改A20地址线（就是指地址总线中的第20根）才能从实模式转换到保护模式。这里涉及到早期CPU为了节省硬件设计成本而使用8042键盘控制器来控制A20 Gate。

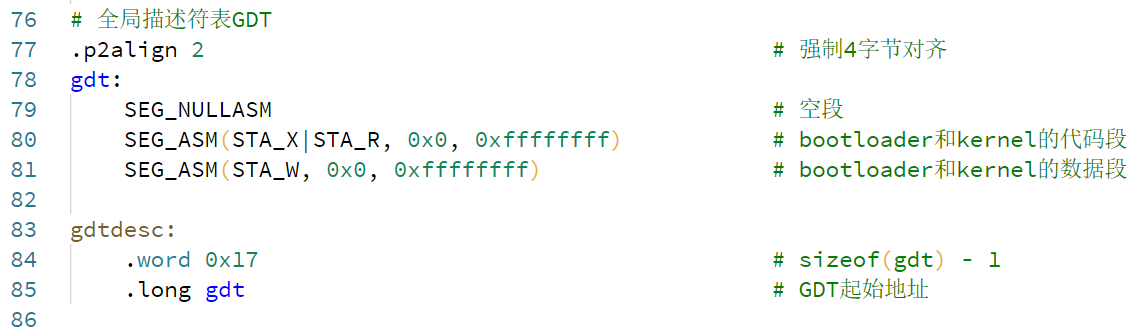


这段代码先从8042 Input Buffer中读取数据，使用testb判断它的第2位（索引为1）是否为1，如果为1说明缓冲器中还有数据没有被处理，直到缓冲器空，继续写数据0xd1（11010001）到0x64端口，发送一个键盘控制命令，从而禁止键盘的操作。然后再次等待直到缓冲器空，写数据0xdf（11011111）到0x60端口，即设置P2的A20位为1，自此，全部32条地址线均可以使用，能够访问4G的内存空间。

接下来需要将CPU从实模式切换到保护模式，在此之前先使用lgdt命令载入已经存在的GDT全局描述符表，并将GDT表的首地址加载到GDTR：



其中gdtdesc是在文件的末尾定义的：

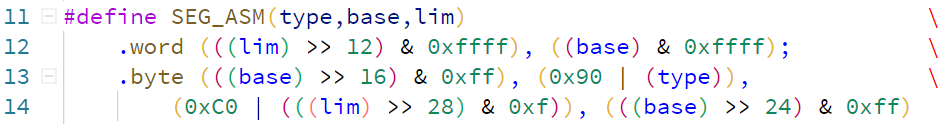


可以看出GDT表的存放位置是4字节对齐的，即GDT表的物理首地址是4的整数倍，gdt部分标识了3个GDT表项，他们使用了asm.h中的两个宏SEG\_NULLASM和SEG\_ASM，其中SEG\_NULLASM的定义如下：



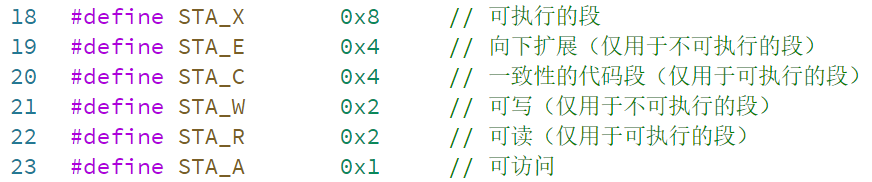
这里定义了8个字节为0，是一个空的GDT表项。

SEG\_ASM宏的定义为：



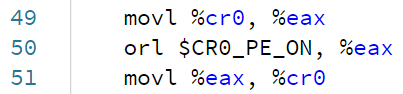
在这里，type表示段属性，base表示段基址，lim则表示段长的界限，给出这三个参数就可以用这个宏来定义一个GDT表项。

其中的type也是宏定义的：



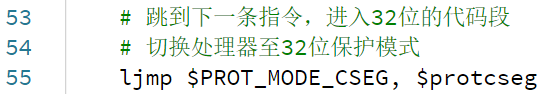
多个不同的type使用按位或（|）连接。

进入保护模式之前的所有工作到此都已经处理完毕，接下来就是设置控制寄存器CR0上的PE位为1以便使CPU进入保护模式：



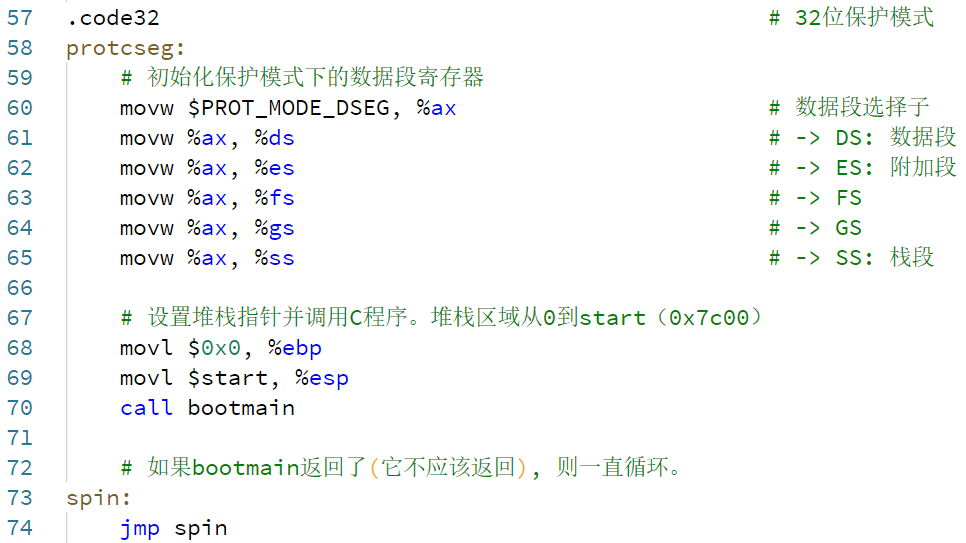
这里CR0\_PE\_ON的值定义为1，通过orl这个32位的按位或指令利用eax寄存器间接将寄存器CR0上的第0位使能设置为1。

之后，用一个跳转指令让系统开始使用32位的寻址模式：



可以看到第55行的长跳转指令实际上是在系统进入保护模式后执行的。于是在这里$PROT\_MODE\_CSEG，代表的是段选择子，从前面的GDT表中可以看到基地址是 0x0，而偏移地址是$protcseg， $protcseg实际上代表的是接下来指令的链接地址，也就是可执行程序在内存中的虚拟地址，只是刚好在这里编译生成的可执行程序boot的加载地址与链接地址是一致的，于是$protcseg就相当于指令在内存中存放位置的物理地址，所以这个长跳转可以成功的跳转到下一条指令的位置。

ljmp指令实际上改变了代码段寄存器%cs的值，让它指向了GDT中的一个代码描述符表项，将处理器真正切换到32位模式。最后，初始化除代码段外的段寄存器为0x10，然后设置堆栈指针指向start处，然后执行call bootmain调用C函数：



实际上这个函数永远都不会返回，末尾73-74行的无限循环实际上并不会被执行到。

附录1：

**本练习使用到的部分AT&T汇编指令含义**

|  |  |
| --- | --- |
| **汇编指令** | **含义** |
| xorw | 16位按位或运算 |
| movw | 16位字传送 |
| movb | 8位字节传送 |
| movl | 32位双字传送 |
| inb | 向I/O端口写入一个字节 |
| outb | 从I/O端口读取一个字节 |
| orl | 32位按位或运算 |
| testb | 字节的与运算，仅修改标志位，不返回结果 |
| lgdt | 装载全局描述符表GDT到全局描述符表寄存器GDTR |

参考资料：

[1] 自在随心．uCore实验1笔记整理[OL]．<http://qiaoin.github.io/ucore-ex1-notes.html>．