# 分析bootloader进入保护模式的过程。

## 1. 基本知识

在区别I/O空间和内存空间的进程的I/O空间写入数据

outb() I/O上写入8位数据（1字节）；

outw() I/O上写入16位数据（1字节）；

outl() I/O上写入32位数据（1字节）；

inb() I/O上读出8位数据（1字节）；

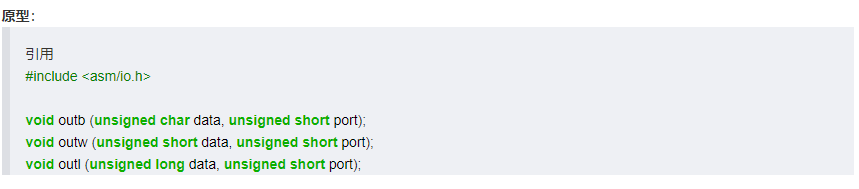
inw() I/O上读出16位数据（1字节）

intl() I/O上读出32位数据（1字节）；

movb 传送8位

movw 传送16位

movl 传送32位



in、out都是对I/O端口进行操作。ioreadb、iowriteb等是对I/O内存操作。

## 2、为何开启A20以及如何开启A20

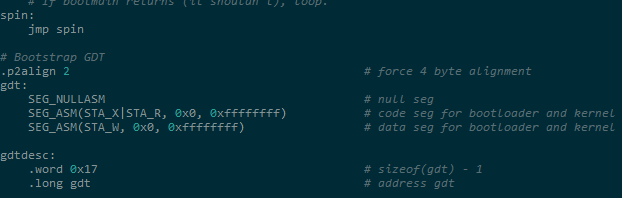
为了将1M（20位）的寻址空间拓展到4G（32位）的寻址空间。从而得进行相应处理。总共包括开启A20，建立GDT、将CR0的bit0位置成1。其中在开启A20之前，BIOS需要关中断、清除方向标志、将AX、DS、ES、SS置0。





开启A20的两个关键端口port 0x64、port 0x60.

## 3、如何初始化GDT表

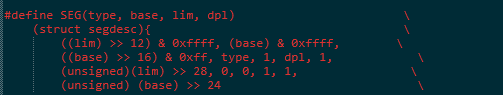


GDT表的存放位置是4字节对齐的（物理首地址是4的倍数）。gdt标示了3个GDT表项，boot.S程序使用了SEG\_NULL与SEG(type,base,lim)这两个宏，所谓的宏就是与函数类似的东西。区别在于在编译的时候，函数被编译成相应的可执行代码，而宏则被编译成对应的常量。

其中SEG\_NULL的定义为



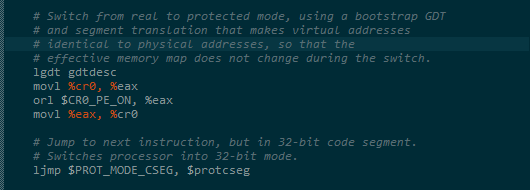
SEG被定义为



表示一个GDT表项由type（段属性）、base（段基址）、lim（段长的界限）来定义，而段属性见表。

|  |  |  |
| --- | --- | --- |
| 宏 | 值 | 属性 |
| STA\_X | 0x8 | 可执行的段 |
| STA\_E | 0x4 | 向下扩展（飞可执行段 |
| STA\_C | 0x4 | 一致性的代码段（仅限于可执行段） |
| STA\_W | 0x2 | 可写（仅限于非可执行段） |
| STA\_R | 0x2 | 可读（仅限于可执行段） |
| STA\_A | 0x1 | 可访问的 |

## 4、如何使能和进入保护模式



Lgdt gdtdesc 将GDT表的首地址加载到GDTR,剩下三个指令是将CRO寄存器的最低位置一，这标志着系统进入保护模式。跳转指令实现的是跳转到下一个指令的位置，其中$PROT\_MODE\_CSEG代表的是段选择子，也即段的基地址。