1. 程序的动态加载和执行

1140310606 张茗帅

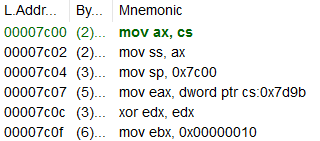
这一章通过一个命名为内核的小型操作系统，演示了如何进行保护模式下的用户程序动态加载。

内核分头部数据、初始化代码、内核代码段、内核数据段和公共例程段5部分，其中初始化代码中包含着主引导程序。

内核加载的内存地址范围为0x40000~0x9ffff（取决于内核多大），起始逻辑扇区号为0x01。

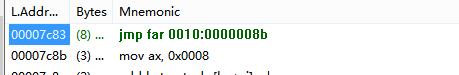
1. **初始化gdt表，然后跳转至保护模式jmp 0x0010:flush**

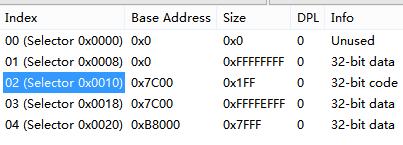
首先BIOS将主加载程序加载到0x7c00处：



随后进行gdt表的初始化工作，以及通过南桥芯片内的端口打开A20，关中断，设置CR0控制寄存器的PE标志位，为进入保护模式做准备。

代码第55行 jmp dword 0x0010:flush如图，通过选择子在GDT表中找到0x10项，获得要跳转的代码基地址，然后与偏移量相加得到最终跳转地址。





1. **巧妙计算加载的扇区数：**

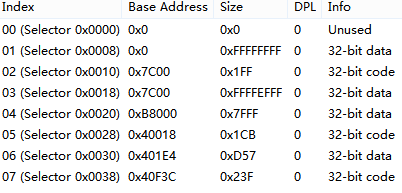
加载内核程序过程中，由于DS描述符在高速缓存中的基地址为0x00，因此我们用edi的值就可以表示加载结束后的最终地址，也就是内核的代码长度，用这个值除以512就可以得到要加载的扇区数，值得注意的是，这个除法结果可能是整数也可能有余数，我们要求的是剩余没有被加载的内核代码所占扇区数，则整除的结果需要减1（下图，81行），非整除则所得的商即为剩余扇区数。根据存储在eax中的剩余扇区数，我们使用read\_hard\_disk\_0过程就可以循环加载所有内核代码。



1. **设置内核的段描述符**

因为保护模式下，不能通过代码段描述符修改段中内容，我们采用数据段描述符在GDT表中增加公共例程的描述符。内核数据段和内核代码段描述符增加同理。

最后的更新过后的gdt表如下：（新增三个段描述符）



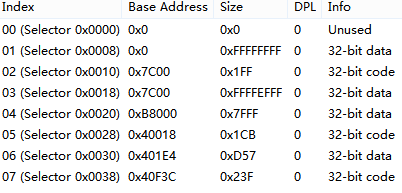
1. **135行，jmp far [edi+0x10]，如下图**



寄存器ds=0x08，在GDT表中得到数据段基地址为0x0。



此时的gdt表（与上一个相同）



偏移地址edi+0x10，edi的值为0x40000，即内核被加载到的内存地址。



0x10处存储的是核心代码入口点，也就是执行完jmp要跳转到的目的地。



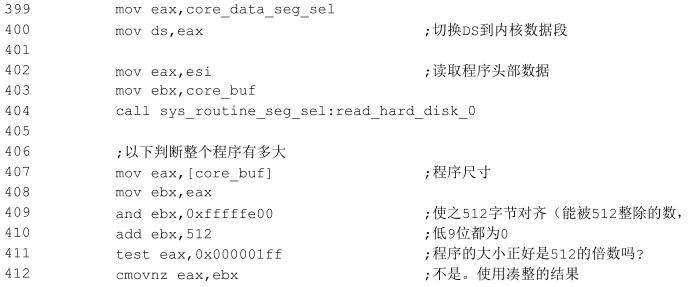
跳转结果，进入内核。

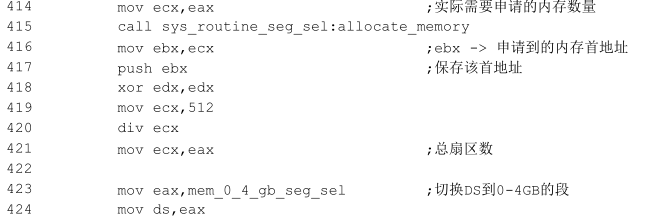


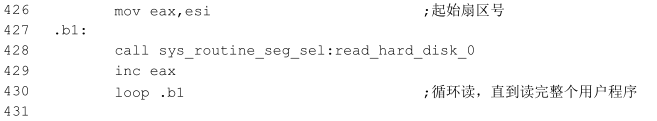
接下来进行的是用户程序的加载和重定位，保护模式下的加载和重定位相较于实模式下要复杂一点。

1. **类似于主引导程序加载内核**

计算扇区数同样是先加载第一个扇区，再做除法计算剩余扇区数，通过过程allocate\_memory计算所分配内存的起始地址，用过程read\_hard\_disk\_0和扇区数进行加载。

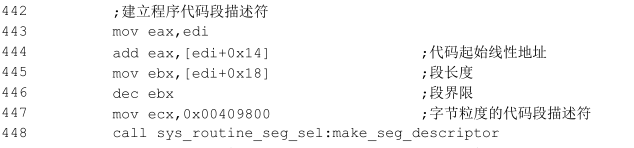




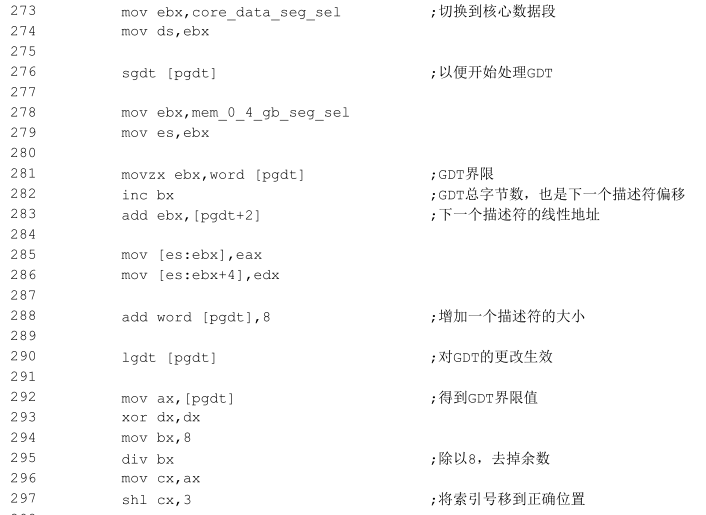


1. **重定位首先要创建新的段描述符**

如下图，通过make\_seg\_descriptor过程形成了EDX：EAX中的64位段描述符，edx：eax=0x409210:0x327

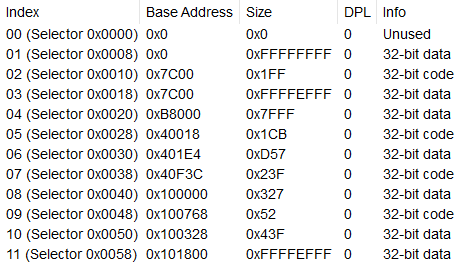


过程set\_up\_gdt\_descriptor中，通过段界限值和gdt表基地址，计算新描述符的位置，并计算该索引号。



新的程序头部段描述符设置完毕，如下图，代码段、数据段、堆栈段的描述符设置方式相同。

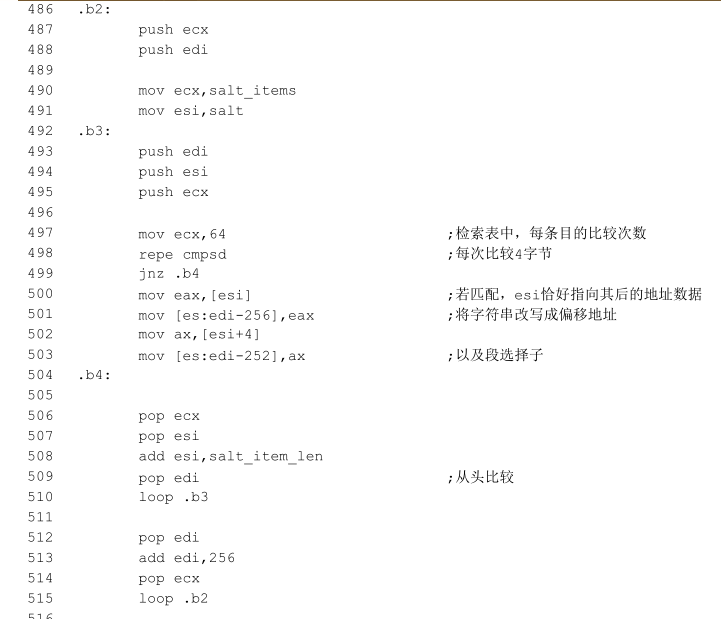
增添完所有描述符后，gdt表中内容如下：



1. **重定位SALT：**

内核中包含全部的公共例程，用户程序0x28处有它将会用到的公共例程，将用户程序加载之后，公共例程的标签地址发生改变，需要重定位与内核公共例程指向相同位置，才能使用。

这里我们使用了双重循环，外循环选择一个用户要用到的公共例程，在内循环中与每一个内核公共例程进行repe比较，找到相同的公共例程后将用户salt表中的公共例程改写成偏移地址，完成重定位。



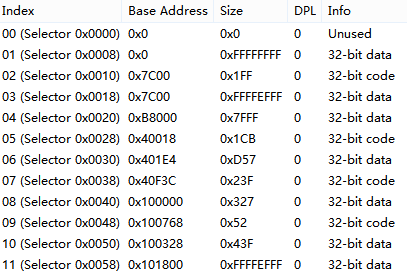
1. 跳转到用户程序：



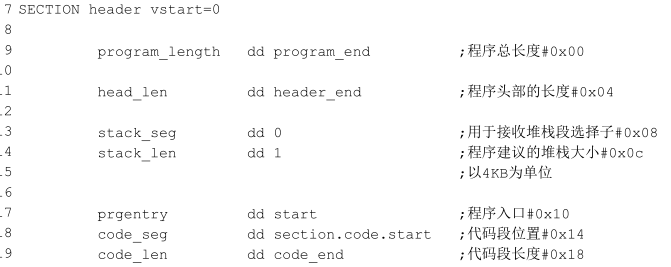
Ds=0x40



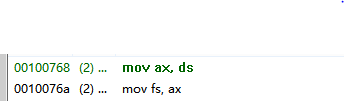
GDT表中对应的段描述符为指向用户程序基地址的0x100000：



根据偏移0x10，找到用户程序中start位置，即为要跳转的用户程序入口。

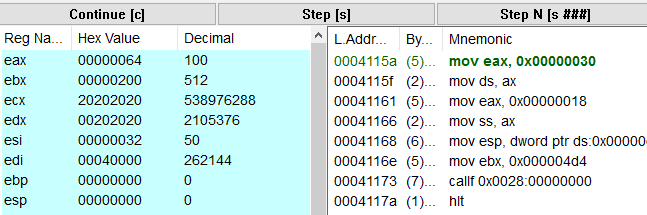


跳转结果：



在用户程序中执行（其间调用了内核的PrintString, ReadDiskData）,最后将控制权返回到系统（jmp far [fs:TerminateProgram] ）





最后展示相关信息后停机。



结束