1.保护模式入门

（1）寄存器拓展

除了段寄存器，通用寄存器、标志寄存器、指令指针寄存器都向高位拓展了16位，成为32位寄存器。为了表示拓展，加了一个e来表示。ax-->eax
寄存器的低16位为了兼容实模式，可以单独使用，但是高16位无法单独使用。

段寄存器还是16位，如果在实模式下，仍然和以后类似。
偏移地址的设置和实模式一样，仍然是IP寄存器保留一个值，CS:IP标记某一个固定的地址。
在32位保护模式内，段基址需要有很多限制条件，仅仅使用寄存器是放不下的。于是，我们使用全局描述符表来存储。
每一个表项叫段描述符，大小为64字节，描述各个内存段的起始、大小、权限等信息。
这张表存放在内存中，使用GDTR寄存器指向它。
此时，段寄存器中保留的就是选择子，这个选择子就是全局描述符表中的一个索引。

段描述符格式很奇怪，为了加速获取，加入了缓存。段描述符缓冲寄存器（DCR）。
当进入到保护模式后，段基址可以随意长度，毕竟是存储在内存中的，只要送入地址总线的数据满足要求，就可以获取目标地址的数据。

后面推出了地址总线和寄存器都是32位的80386，在段描述符中，段基址为32位，在IP段内偏移寄存器中，也为32位。此时仅仅使用IP段内偏移寄存器，就可以访问4GB空间的各个角落。

（2）保护模式寻址拓展

在实模式下，基址寻址、变址寻址、基址变址寻址只能用基址寄存器bx、bp（bx默认的段寄存器是ds数据段、bp默认的段寄存器是ss栈），变址寄存器si、di。
在保护模式下，内存寻址中，基址寄存器可以是任意32位通用寄存器，变址寄存器也一样，同时变址寄存器也可以乘上某个因子。
esp不能用作变址寄存器，但是可以用做基址寄存器。

（3）保护模式之运行模式的反转

保护模式和实模式下，命令拥有两套编码，所以在编译时，我们需要用[bits 16]或[bits 32]来明确指出编译代码的目标。
详见代码如下：

```
[bits 16]
mov ax, 0x1234
mov dx, 0x1234

[bits 32]
mov eax, 0x1234
mov edx, 0x1234
```

还有一些额外的知识点：操作数大小反转前缀0x66和寻址方式反转前缀0x67，这二者是保护模式和实模式可以相互调用的基础，增加这些反转前缀是编译器自动完成的。

（4）保护模式之指令拓展

一般的指令区别不大，可能乘法mul和除法有些区别。
压栈指令有一些小意思：

在保护模式下，压栈8位立即数，默认被拓展成32位压栈。
压栈16位立即数，添加0x66操作数大小反转前缀，压栈16位。
压栈32位立即数，直接加入即可。

对于段寄存器压栈，无论在任何模式下，都是压栈当前模式的段寄存器位宽。
实模式：段寄存器是16位。
32位保护模式下：段寄存器是32位。

2.全局描述符表

全局描述符表Global Descriptor Table GDT是保护模式下的内存段的登记表。

（1）段描述符

段基址有32位，被分成三段，陈列在不同的location。
段界限有20位，低32位由16位，高32位由4位。
标记G表示粒度，G=1表示粒度为4KB，G=0表示力度为1byte，与之对应的是2^20位的段界限，计算后的地址空间分别为4GB和1MB，分别对应于保护模式和实模式。

倘若访问的段内偏移超过了段界限，认为出现越界，CPU陷入中断异常，这是内存访问保护的底层原理之一。
系统的访问权限（读写执行）位于此处，非系统段中的代码段没有写权限，只可以读执行，非系统段中的数据段有读写执行三种权限。
段描述符的13-14位是DPL字段，描述符特权级。
CPU由实模式进入保护模式后，特权级为0，表示是操作系统代码。
用户程序通常位于特权级3，权限最小。某些指令只能在特权级0下执行，用户禁止执行，这样就确保了操作系统的安全性。
15位P表示是否在内存中，处理过程需要开发人员来写。实际上这是未开启分页时需要考虑的，开启分页之后，不需要再考虑了。

（2）全局描述符表GDT、局部描述符表LDT及选择子

这里涉及到了一些代码，需要认真考虑一下?
由寄存器GDTR指向，GDT Register指向。GDTR是一个48位的寄存器，前32位是地址，后16位是界限。
对于此寄存器的访问，我们需要使用命令lgdt。
lgdt 48位内存数据
前16位是以字节为单位的界限值，后32位是GDT的起始地址。

段的选择子：
本质上是一个16位的索引。
0-1位：RPL，特权级，0,1,2,3四种特权级
2位：TI为，表示选择子在GDT中还是LDT中，TI=1，选择子在LDT中。
索引部分13位，可以定义8192的段，GDT全局描述符表也规定最多定义8192个段描述符，二者是完全吻合的。

之后，根据给出的选择子，读取GDT或者LDT中的段描述符，获取基址，再加上段内偏移量IP，即可获得最终的目标地址。

GDT的第0个描述符不可用的。若选择子未被初始化，就会选择第0个，此时CPU会报出异常。

马上就到彻底进入保护模式的部分了，看完会让人非常的开心的。
LDT在许多操作系统中的使用都非常少，并不需要特别详细地了解。

（3）打开A20地址总线

在地址总线只有20位时，在实模式中，会出现地址回绕现象。
后续地址总线位宽上升，就存在了不兼容的可能性。
为了兼容性，IBM在键盘控制器上添加了一些输出线来控制第21根地址线A20的有效性，被称为A20Gate。
若A20Gate被打开，CPU会真正访问目标地址。
若A20Gate没有被打开，CPU会访问回环地址，也就是按照8086/8088地址回绕。

如今我们需要进入保护模式，需要突破1MB内存限制，需要关闭地址回绕，也就是打开A20Gate，打开A20地址线。

打开的方式是极为简单的：将92端口的第1位置置为1即可。

```
in al, 0x92 ; 内存绝对寻址
or al, 0000_0010B
out 0x92, al
```

（4）保护模式的开关 CR0寄存器的PE位

CR0寄存器的第0位，也就是PE为，Protection Enable位，只有把这一位置为1，CPU才真正进入保护模式。

（5）进入保护模式

综上所述，进入保护模式步骤如下：
在实模式设置GDT表并且加载、打开A20地址总线、设置CR0寄存器的PE为，用[32bits]编译代码，最终可以直接进入保护模式。
当然，现在写的代码还有一些问题，需要处理掉。

小tips：
type中的e字段为0，表示向上拓展，比如0xb800向上拓展7，远处地址为0xb8000+7*4k = 0xbffff
也就是拓展方向是从低地址向高地址拓展，倘若超出地址范围，就会触发CPU中断异常。

整个编译命令串：
loader写入第二扇区四个扇区的内容
mbr写入第0个扇区一个扇区的内容

```

```

```
nasm -I include/ -o loader.bin loader.S
nasm -I include/ -o mbr.bin mbr.S
dd if=./loader.bin of=./hd60M.img bs=512 count=4 seek=2 conv=notrunc
dd if=./mbr.bin of=./hd60M.img bs=512 count=1 conv=notrunc
```

```
wzd@wzd-virtual-machine:~/bochs$ dd if=./loader.bin of=./hd60M.img bs=512 count=
4 seek=2 conv=notrunc
记录了1+1 的读入
记录了1+1 的写出
618字节已复制，0.000556634 s，1.1 MB/s
wzd@wzd-virtual-machine:~/bochs$ dd if=./mbr.bin of=./hd60M.img bs=512 count=1 c
onv=notrunc
记录了1+0 的读入
记录了1+0 的写出
512字节已复制，0.000580499 s，882 kB/s
wzd@wzd-virtual-machine:~/bochs$
```

成功进入了保护模式!