## 1.2 Intel X86系列CPU寻址方式

8086考虑到内存地址空间不足，采用了分段的方式；采用“实地址模式”，段寄存器与逻辑地址相“加”得到20位的实际地址；

80286开始实现其“保护模式”；

80386 段寄存器由单纯的基地址变为指向段描述符的指针，完成了越界和越权检查。伴随着段寄存器有对应的“影子”描述符，每当段寄存器改变时随之变化；

段式存储管理可以通过检查相应段描述项中的p标志位，来实现段式虚存管理

## 1.3 I386页式内存管理机制

CR3是指向当前页面的指针

为什么使用两个层次？

出于空间效率的考虑，线性地址格式为dir+page+offset 如果将dir和page合在一起，页面表就有20位即1M个表项，又每个页面4K大小，总空间仍为4G大小，但很难想象一个进程需要4G存储空间。采用页目录时，若目录项为空，不必设立相应的页表，节省空间

页面表和页面的起始地址总是在4K字节的边界上，这些指针的低12位永远是0，这样在目录项和页表项中都只要有20位用于指针就够了，余下12位用于其他目的。

页面表项或目录项中最低位p，表示页面或页面表是否在内存中，实现页式虚存

PSE页面大小扩充机制：当目录项中ps位为1时，页面大小为4M，页面表不再使用，线性地址中的低22位全部用作4M字节页面的位移。

## 1.4 Linux内核代码中的C语言代码

由于在GUN的C语言中引入了C++的保留字如“inline”，老代码中可能存在变量名为inline，因此在其前后都加上“\_\_”变为“\_\_inline\_\_”，也正是如此，Linux只能使用gcc编译，并且相互之间有依赖关系

Linux中把链表中的前向和后向指针抽象成数据结构list\_head，并将list\_head寄宿在需要连接件的数据结构中，如page

struct list\_head{

struct list\_head \*next,\*prev;

}

可以通过memlist\_entry()即list\_entry()得到宿主的地址