CR3 寄存器

Virtual Memory 的实现是需要硬件支持的！

依稀记得 n 久在学校里学《操作系统原理》的时候，曾经介绍过 OS 中为了实现内存分页的各个表项啊什么 XX 的，用来将某逻辑地址，通过某种恶心的查询策略，翻译成 CPU 可寻址的物理地址。现在在工作中由于经常接触汇编级别的调试，迂回于 Process 的 4G 地址空间中，猛然回想起之前老师所说的，现在看起来是如此的清晰和真实|||

不知道还有多少人记得 CPU 的保护模式？呵呵，狞笑…

- **实模式：**特征是实地址模式，地址的转换就是很简单的：段+偏移=物理地址。那时老师总会强调一句，DOS 就是在这个模式下运行的。汗~
- **保护模式：**（重点来了）就是它的提出，为多任务和 Virtual Memory 提供了硬件上的支持。所以有了现在的 Process Space 这个东东。此外加上了特权的概念。Intel CPU 提供了 Ring0 到 Ring3 的 4 种模式，Windows 使用了其中的两种 Ring0 和 Ring1，分别对应我们耳熟能详的 Kernel 和 User 模式。
- **系统管理模式：**BIOS 执行电源管理，安全检查等等特定任务。

Virtual Memory 和 Process Space 在保护模式中实现。接下来比较重要的问题在于，他们是相对于一个进程的。不同的进程拥有自己的所属。所以当 CPU 在执行不同应用程序的时候，他们是要切换的！！那怎么切换呢？呵呵，狞笑…其实答案已经在 Title 中给出。CR3 寄存器！！！

从一个 Virtual Address 转换成一个 Physical Address，需要经历“段+偏移”产生的线性地址这么一个中间产物。一般格式如下（32 位线性地址）：

1. **31~22 位：页目录地址偏移；**
2. **21~12：页表偏移。它含有一个 Flag，用以标识地址是在 RAM 中还是已经被交换到硬盘中了，如果在硬盘中，OK 那么 Page Fault，SWAP；**
3. **11~0：Offset，相对于该页的偏移；**
4. **页目录，也就是当前进程的内存页，它的基地址就保存在 CR3 寄存器中！**

所以我们把线性地址翻译成物理地址，需要 CR3，然后获得对应的页目录项、页表项，然后将他们累加就可以了。所以 CR3 很关键，它的值改变了，那就不能在当前 Process Space 中寻址了。所以如果 CR3 设置为另一个进程的页目录基地址，那么 CPU 就是在切换地址空间。

Hope this can help…