# 课堂笔记

这节课讲了物理地址扩展PAE，也就是29912分页模式下，线性地址是如何拆的。

PAE 最多能支持36根地址总线。物理地址最大范围扩大到了2^36=64GB, PDE和PTE 每项扩大到8字节。

### 1.拆线性地址

拆一个典型的，gdtr



80b99 000

10

00 0000 101

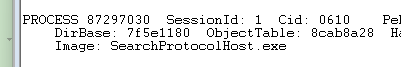
1 1001 1001

Directory Pointer : 2

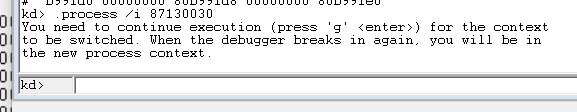
Directory: 5

Table: 199

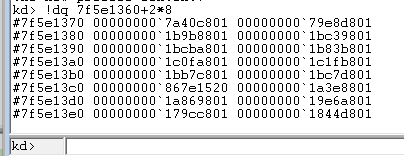
随便找一个进程的CR3，也就是页目录指针表基址：



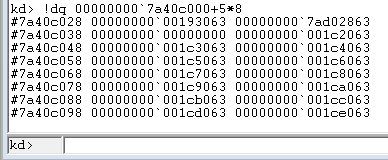
附加上去



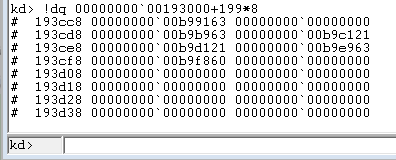
看看这个进程的gdt表



找到了页目录指针表项PDPTE，清空低12位就是页目录表的基址

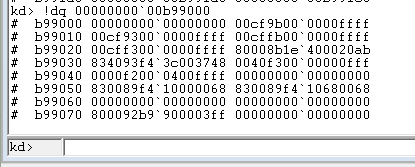


通过页目录表基址+Directory\*8找到页目录项PDE，清空低12位就是页表基址



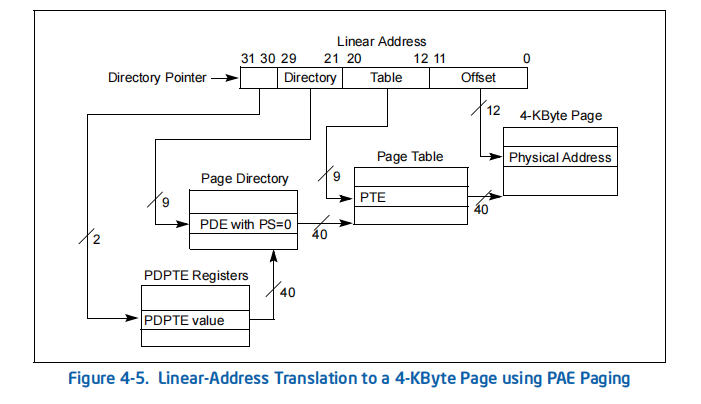
通过页表基址+Table\*8就能找到页表项PTE，清空低12位就是物理页基址

刚才拆得页内偏移是0，直接!dq看看是不是GDT表？



没毛病。

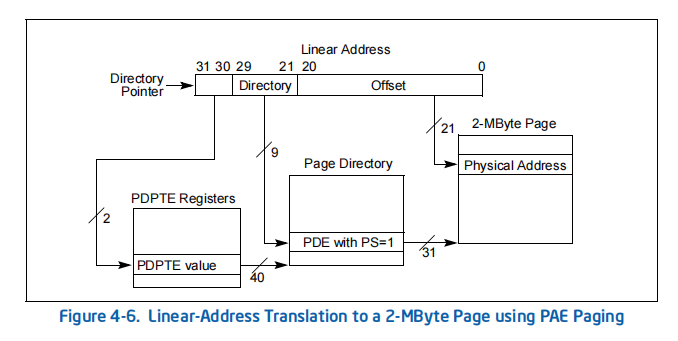
### 2.29912分页管理



29912的Directory 和 Table 变成了9位，高2位变成了Directory Pointer，也就是页目录指针表PDPT的索引。在29912分页下，CR3指向的就是PDPT.

**再次强调，PDPTE,PDE,PTE每项都是8字节。**

以上是4KB小页的情况，如果是大页，则还简单些，Table没有了：

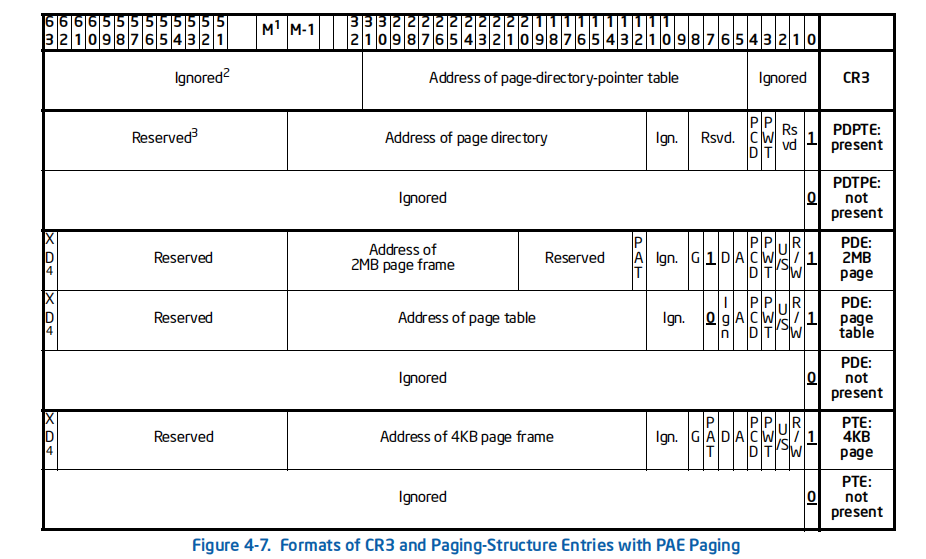


这个大页和101012不一样，它只有2MB了，因为2+9=11，Offset只剩下21位了，就是2MB了。

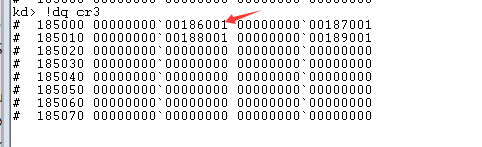
这是Intel白皮书给的定义，操作系统实现上遵循这个规定，但是在内存中实际上长得跟这个稍微有点差别，因为页目录表PDT其实是一个特殊的页表。

不过只要理解白皮书这两张示意图就ok了，操作系统怎么映射线性地址其实也没啥关系。

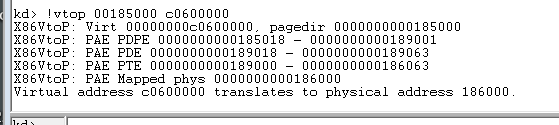
至于这个CR3指向页目录指针表，白皮书是说了的：



昨天逆向MmIsAddressValid时发现计算PDE时用到了线性地址 0xC0600000，这个是第一张页目录表的线性地址，为什么这么说呢？



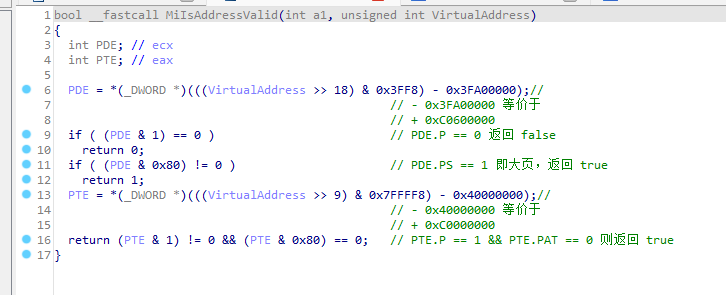
可以看到第一个PDPTE是 00000000`00186001 ，去掉低12位就是第一张页目录表的物理地址 186000



确实就是线性地址 C0600000对应的物理地址。

# 作业

### 1.如果没有做完昨天的分析题目的 请继续



刚才已经证明了 线性地址 0xC0600000就是第一个页目录表的基址。具体的编译器优化和101012没啥区别，就不赘述了。

### 2.如果游戏公司 使用MmIsAddressValid 扫描你的内存。你怎么对抗MmisAddressValid函数扫描

排除法，让函数返回false，要么P=0,要么PAT=1。P=0肯定是不行的，CPU直接产生异常，只能是让PTE.PAT=1了。