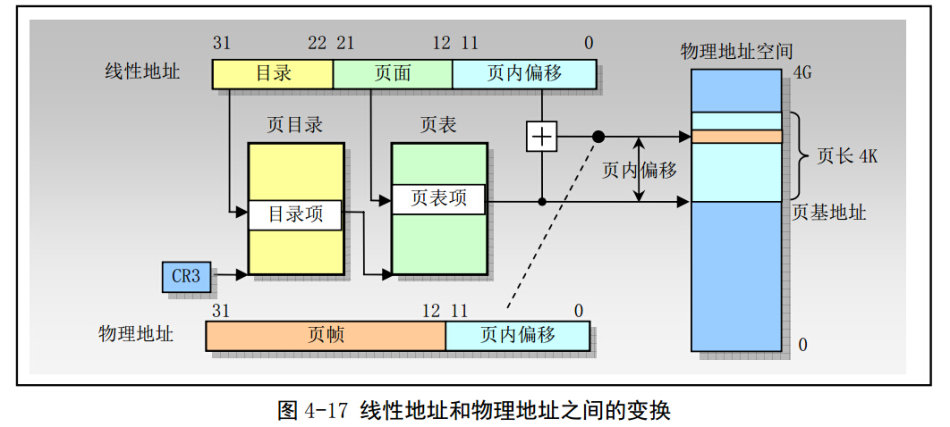
近日看到大疆14年嵌入式的笔试题是这样的：简述处理器在读取内存过程中，CPU核，Cache，MMU是如何协同工作的？（用自己熟悉的处理解释）不明觉厉啊，捯饬了一段时间有点明白了，就写下来记录一下。

首先不得不说大疆是个“心机婊”​，大家很熟悉的STM32因为没有MMU（Memory ManagementUnit）所以不能举栗子了，STM32存储器映射用的是“位带”。这个先不讲了，相信网上有很多关于VA（virtualaddress）转化为PA（physicaladdress）的文章了，我借鉴大家的看法和自己的理解写了这篇博客，可能有错误，欢迎大家指正。用在学院派的S3C2440处理器。

话不多说，先上图：（这个图来自《Linux内核完全注释》赵炯；以下用《Linux注释》）

*填写图片摘要（选填）*

CPU核要访问内存，发出一个地址，此地址为虚拟地址，因32位机器上，地址线为32条，可以访问的的地址范围是0x0000\_0000~0xffff\_ffff，这里多说一句访问相同的虚拟地址，但不同的进程给映射到不同的物理地址，小于32M的地址MVA = VA | (PID << 25);MMU先访问TLB（Translation LookasideBuffers），如果TLB中含有能转换这个虚拟地址的描述符，则直接利用此描述符进行地址转换和权限检查，访问到的内存数据又通过MMU返回给CPU内核；如何TLB中没有这个描述符，就MMU就会像上图进行查找；

之前看过多篇文档发现开始检查有是从TLB中开始的，《Linux注释》用的是CR3，开始我也不明白，所以只能Google了。查到的贴下来：

The TLB is not transparently informed of changes made to pagingstructures. Therefore the TLB has to be flushed upon such a change.On x86 systems, this can be done by writing to the page directorybase register (CR3):

movl%cr3,%eax；

movl%eax,%cr3；

(⊙o⊙)…这让我吹嘘可以熟练阅读英文文档的人敲了一个警钟；大体意思是：TLB不知就会被改变，所以没办法，一改变就会立即写给CR3，保持TLB与CR3相同；我是根据flused函数猜的；

利用二级表查找；第一级表称页目录（page directory），含有2^10（1K）个4字节的表项，这个表项的值为第二级表项的地址；第二级表项称为页表（pagetable），也是含有2^10（1K）个4字节，这表项就是存放物理地址所在区域的基地址，再加上[11..0]页面偏移就对应所映射的物理地址。

那Cache怎样工作的呢？其实TLB本身就是Cache；当找到所映射的物理地址后返回给CPU内核的同时也会将这个描述符写入TLB中，方便下次高速的访问。所以实验的时候你就会发现相同的程序开启MMU比没有开启MMU的执行的速度快，有led灯指示啊（当时不晓得为啥会闪的比直接访问内存快）

https://weibo.com/p/23041897851b3d0102wl97