《内存管理》章节练习

ChCore运行在64位的机器上，并且配置了4级页表，第四级为PGD，第三级为PUD，第二级为PMD，第一级为PTE。请回答以下问题：

1. 在32位的机器中，巨页的大小是4M；在64位机器中，巨页的大小是2M，为什么巨页的大小不一致。

页表页的大小是4k，64位机的pte是8byte，一个页里面有512个pte，可以管理2M大小的地址空间。64位机的pte是4byte，一个页里面有1024个pte，可以管理4M大小的地址空间

2. 为什么OS/MMU使用多级页表映射虚拟地址到物理地址中？使用4级页表的最大内存空间消耗是多少？

多级页表允许页表中出现空洞，可以减少空间占用。对于64位地址空间，占用最大空间为33620096.25GB

3. 在ARM-mmu架构中，mmu是如何区分页条目是否被使用。

页表页L3中的页表项的第0位表示该项是否被使用

4. 使用巨页的优点和缺点分别是什么？

- 优点是可以减少TLB缓存项的使用，提高TLB命中率；减少页表的级数，提升遍历效率
- 缺点是如果不能使用整个大页的话会造成物理内存资源浪费，增加管理内存的复杂度。对于数据库应用程序，占用内存很大，不使用大页的话页表会变得很大，将数据库的共享内存放在大页可以减小开销。大页面不能用于文件缓存

5. 内存的属性位AP和UXN已经能够隔离用户态和内核态，为什么还需要两个ttbr寄存器？

如果内核和用户页表放在一起，仅由属性来判断，这会导致权限判断延迟，从而在CPU冒险冲突中得以访问无权限数据，而使用两个ttbr寄存器也就是使用两个页表，可以让权限判断与CPU计算解耦

6. TLB能够缓存虚拟地址到物理地址的映射，当发生进程间的上下文切换的时候，需要刷掉所有的TLB条目，这是为什么？你能想出一种解决方式，使得TLB条目在上下文切换的时候不需要被刷掉吗？

- 因为每个进程都有自己独立的虚拟地址空间，即使是相同的虚拟地址，在不同进程中也可能对应不同的物理地址，因此在进程切换的时候，缓存在TLB中的条目需要刷新

- 操作系统可以为不同的应用程序分配不同的身份标签，再将标签写入应用程序的页表基地址寄存器中的空闲位。同时，TLB的缓存项也会包含这个标签，从而使得TLB中属于不同应用的缓存项可以被分开。因此在切换页表时不再需要清空TLB

7. 在ARMv8结构之前，内存属性中没有DBM（Dirty Bit Modifier）位。这意味着硬件不支持脏页。所以OS需要如何模拟并且记录脏页呢？给出一种可行的解决办法。

可以通过设置访问权限加异常处理来模拟。要追踪一个可写的页是否为脏页，在建立对应的页表时，可以先设置为readonly。当要写这个页的时候会产生异常，kernel检查异常原因时发现是访问权限的问题，于是kernel把权限改为可读写并且记录其为脏页