思考题部分：

**思考题1.请思考 cache 用虚拟地址来查询的可能性，并且给出这种方式对访存带来的好处和坏处。另外，你能否能根据前一个问题的解答来得出用物理地址来查询的优势?**

答：cache是存放主存中最常使用的数据的地址，如果采用虚拟地址来查询，则查询到还需要转换为物理地址映射到主存中。这一情况的好处是可以省去之前的TLB查询，直接对cache进行查询，如果命中转换为物理地址直接映射到主存中，这样更高效；坏处就是如果不命中，替换比较麻烦，需要将物理地址转换为虚拟地址替换到cache中，还有相应的映射地址也要替换，并且虚拟地址容易造成混乱，当两个程序同时运行的时候，虚拟地址就会发生冲突。用物理地址查询，实际上TLB查询的失效率不会太高，程序地址映射后的物理地址不会冲突，从而保证了程序的并行运行。

**思考题2**.请查阅相关资料，针对我们提出的疑问，给出一个上述流程的优化版本，新的版本需要有更快的访存效率。（提示：考虑并行执行某些步骤）

答：考虑虚拟地址访问cache，物理地址比较Tag。首先利用虚拟地址访问TLB，进行虚实地址转换；同时在这个周期中，利用虚拟地址的低位访问cache，取出cache中存储的tag和data，此时的tag其实对应的是物理地址的tag；假设TLB和cache都命中，则利用上个步骤得到的物理地址进行tag比较，并选择合适的cache data数据。

**思考题3.**在我们的实验中，有许多对虚拟地址或者物理地址操作的宏函数 (详见 include/mmu.h ), 那么我们在调用这些宏的时候需要弄清楚需要操作的地址是物理地址还是虚拟地址，阅读下面的代码，指出 x 是一个物理地址还是虚拟地址。

答：指针是一个虚拟地址。

**思考题4.**我们注意到我们把宏函数的函数体写成了 do { /\* ... \*/ } while(0)的形式，而不是仅仅写成形如 { /\* ... \*/ } 的语句块，这样的写法好处是什么？

答：为了使得我们需要定义多语句宏的时候依然能够按照定义顺序来执行。且避免使用goto语句，巧妙使用do{ }while(0)和break可以起到这个作用。

**思考题5.**注意，我们定义的 Page 结构体只是一个信息的载体，它只代表了相应物理内存页的信息，它本身并不是物理内存页。那我们的物理内存页究竟在哪呢？Page 结构体又是通过怎样的方式找到它代表的物理内存页的地址呢？请你阅读include/pmap.h 与 mm/pmap.c 中相关代码，给出你的想法。

答：获得page的物理地址：首先右移22位得到第几个页表，(pp-pages)<<PGSHIFT，左移12位得到页表内偏移。

**思考题6.请阅读 include/queue.h 以及 include/pmap.h, 将 Page\_list 的结构梳理清楚, 选择正确的展开结构 (请注意指针)。**

答：C

**思考题7.在 mmu.h 中定义了bzero(void \*b, size\_t)这样一个函数, 请你思考，此处的 b 指针是一个物理地址，还是一个虚拟地址呢？**

答：此处b是一个void函数类型的指针，在此处作为bzero函数的参数，对应的是虚拟地址。

**思考题8.了解了二级页表页目录自映射的原理之后，我们知道，Win2k 内核的虚存管理也是采用了二级页表的形式，其页表所占的 4M 空间对应的虚存起始地址为 0xC0000000，那么，它的页目录的起始地址是多少呢？**

答：(0xC0000000 >> 12 )\*4= 0x30000，为相对页表起始地址的偏移，0x30000 + 0xC0000000 = 0xC0030000

思考题9.思考一下 tlb\_out 汇编函数，结合代码阐述一下跳转到 NOFOUND的流程？从 MIPS 手册中查找 tlbp 和 tlbwi 指令，明确其用途，并解释为何第 10行处指令后有 4 条 nop 指令。

答：首先读取寄存器CP0的值存在k1里，把a0写入CP0；找到TLB入口，并设置寄存器Index为入口；再读取CP0到k0里，如果k0也就是Index小于0则跳转到NOFOUND标签。Tlbp是寻找TLB目录匹配的入口并加载到index，tlbwi是通过Index写入TLB目录中。

由于之后立即用到index寄存器，所以需要添加4个nop保证没有数据冒险。

思考题10.显然，运行后结果与我们预期的不符，va 值为 0x88888，相应的 pa中的值为 0。这说明我们的代码中存在问题，请你仔细思考我们的访存模型，指出问题所在。

答：本次实验中不存在TLB缺失的情况，也就是访问TLB一定有对应的项存在。而当va=0x88888的时候，TLB中不存在对应的项，无法进行中断处理。

思考题11.在 X86 体系结构下的操作系统，有一个特殊的寄存器 CR4，在其中有一个 PSE 位，当该位设为 1 时将开启 4MB 大物理页面模式，请查阅相关资料，说明当 PSE 开启时的页表组织形式与我们当前的页表组织形式的区别。  
 答：原本每个页表是4KB大小，共有1024个页表。当需要扩大为4MB页面时，PSE设为1，线性地址高10位为页目录索引，低22位是页内偏移地址。

实验难点图示：

1. Lab2开始操作系统的真正实验，难度不容小觑，在原理代码理解上花了很大功夫。首先编写一些宏函数，对页面链表进行操作。
2. 第一次主要是物理内存管理，空闲页面的申请和释放。
3. 接着第二次是虚拟内存管理，通过两级页表实现。

STEP

03

虚拟内存，二级页表STEP

虚拟内存，二级页表STEP

02

物理内存，页面的释放与

创建STEP

创建STEP

01

宏函数的编写÷

体会与感想：

本次lab2作业增加了很大的难度，开始进入操作系统的管理部分，关于看代码的理解，自己还是有很大的难度，有的时候完全不知道如何下手，自己编写代码的时候找不到头绪，也不知道自己写的正确与否，往往这时就必须与写好的同学讨论，我想问他们为什么会做，而我不会？实验基本是自己摸索，阅读指导书，阅读代码，通过课上讲解的内存管理知识联系到代码中，感觉还是有一点抽象。在以后的实验中还得加强学习了。