1. 在如下一段 C 语言程序的 for 循环中，

```
void cycle(double* a) {
  int i;
  double b[65536];
  for(i=0; i<3; i++) {
   memcpy(a, b, sizeof(b));
  }
}
```

程序memcpy函数执行过程中可能发生哪些例外，各多少次。

答：

【说明：此题中试图问的是与页处理相关的例外（忽略实际系统中可能发生的中断），以MIPS体系结构为例，包括三类共四种例外：

(1) TLB refill 例外；

(2) TLB invalid Load例外 / TLB invalid Store例外；

(3) TLB modify 例外。

其中(2)和(3)两类例外仅与程序对于页的访问需求相关，因此回答次数时需要假设操作系统中页的大小；而(1)还与处理器中TLB的容量相关，因此还需要假设处理器中TLB的项数和TLB的替换策略】。

下面给出一种常用假设条件下的回答：

假设操作系统页大小为4KB，同时系统的物理内存足够大，即页表分配的物理内存在程序执行过程中不会被交换到swap区上。并且在后续的分析中忽略代码和局部变量i所在的页的影响。

a、b两数组大小均为65536×8=512KB。再假设a、b两数组的起始地址恰为一页的起始地址。那么a、b两数组各自均需要512KB/4KB=128个页表项。

所以a、b的访问造成的TLB invalid 例外次数为128+128=256次；

假设TLB表项为128项，每项映射连续的偶数奇数两个页，采用LRU算法。那么第一次次循环中，a、b两数组每访问相邻偶数奇数两页的首地址时，均会触发一次TLB refill例外。后续各次循环TLB中均命中。那么共触发的TLB refill例外次数为128/2 + 128/2 = 128次。

2. 对于指令 Cache 是否有必要考虑 Cache 别名问题？

答：

有必要。

1、当程序中有自修改代码时，可能会由于 cache 别名导致取错指令。

2、由于 Last Level Cache 通常是用物理地址索引的，那么当一级指令 cache 存在别名问题

时，整个处理器无法维护一级指令 cache 和 Last Level Cache 之间相同物理地址的数据备份之间的关系，导致处理器进入混乱状态，最终导致程序执行错误甚至死机。

3. 假定在某一个 CPU 的 Cache 中需要 64 位虚拟地址，8 位的进程标识，而其支持的物理内存最多有 64GB。请问，使用虚拟地址索引比使用物理地址作为索引的 Tag 大多少？这个值是否随着 Cache 块大小的变化而发生改变?

答：

1、由于虚拟地址有 64 位，进程标识为 8 位；而物理地址是 64GB，即需要 36 位物理地址。这样，使用虚拟地址比使用物理地址总共增加的位数为 8＋（64－36）＝8＋28＝36 位。
2、改变块的大小，亦或改变 cache 的其它参数，cache 占据的地址的低位的长度对于虚地址索引和物理地址索引都是一样的，所以对于 cache 的 tag 而言，两中索引方式相差位数保持不变。

4. 考虑一个包含 TLB 的当代处理器。（1）请阐述 TLB、TLB 失效例外、页表和 Page fault 之间的关系。（2）如果有这样一个机器设计：对于同样的虚拟地址，TLB 命中和 Page fault 同时发生，这样的设计合理么？为什么？（3）现代计算机页表普遍采用层次化的方式，请解释原因。

答：

1) **TLB**：Translation lookaside buffer，即旁路转换缓冲，或称为页表缓冲；里面存放的是一些页表文件（虚拟地址到物理地址的转换表）的一个子集。

   TLB 失效例外：假如某个虚地址 VA，在 TLB 中没有找到对应的页表项，则发生 TLB 失效例外(TLB miss exception)。

   页表：简单的说，页表是内存块的目录文件，作用是实现从逻辑页号到物理块号的地址映射。即用指定大小（称之为页）来划分程序逻辑地址空间和处理机的物理内存空间，并建立起从逻辑页到物理页的**页面映射**关系，这些页面映射就是一张张页表（Page Table）。

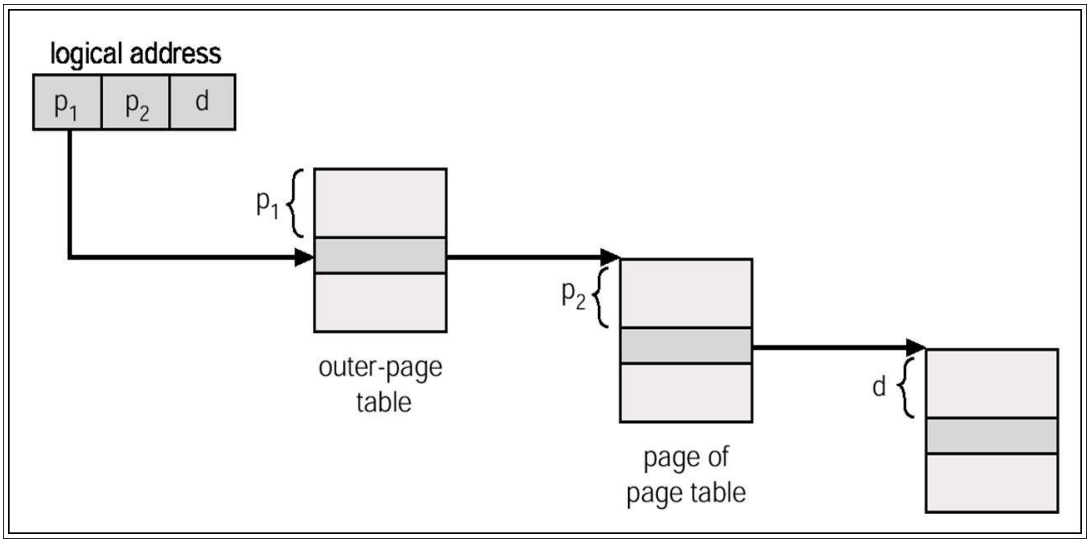
   Page Fault：缺页，就是 TLB 中没有虚地址 VA 对应的页表，主存中也没有 VA 对应的物理地址。这时候发生 Page Fault Exception（缺页例外），需要交给操作系统处理，通常需要消耗很多的时间来处理该例外。

2) 不合理，因为 Page Fault 发生时意味着主存中没有虚地址 VA 对应的页表，这是 TLB 中也应该没有，因为 TLB 里的页表是主存中页表的子集，应该发生 TLB miss。倘若没有发生，说明 TLB 不是主存页表的子集，这会导致存储不一致，也违背了层次化设计的基本要求。

3) 多级页表可以减少页表所占用的存储空间。比如：

一个 32 位逻辑地址空间的计算机系统，页大小为 4KB，其虚地址可以被分为：20 位的页号 + 12 位的偏移。那么单级页表的条目有 2^{20} 条。假设每个条目占 4B，则需要 4MB 连续物理地址空间来存储页表本身。

如果采用两级页表，假设将 20 位的页号分解为：10 位的外部页表号 + 10 位的外部页表偏移。如下图所示，逻辑地址中：p1 用来访问外部页表的索引，p2 是外部页表的页偏移。其中第一级页表需要 4KB 连续物理地址空间，第二级页表每张表也需要 4KB 连续物理地址空间，由于通常情况下第一级页表中并非每一项都需要相应的第二级页表，所以总的存储空间要少于单级页表。



5. 已知一台计算机的虚地址为 48 位，物理地址为 40 位，页大小为 16KB，TLB 为 64 项全相联，TLB 的每项包括一个虚页号 vpn，一个物理页号 pfn，以及一个有效位 valid，请根据如下模块接口写出一个 TLB 的地址查找部分的 Verilog 代码。

```
module tlb_cam(vpn_in, pfn_out, hit,...);
```

其中 vpn_in 为输入的虚页号，pfn_out 为输出的物理页号，hit 为表示是否找到的输出信号，“...”表示与该 TLB 输入输出有关的其他信号。重复的代码可以用“...”来简化。

答：

```
module tlb_cam(vpn_in, pfn_out, hit, valid_out);

input [33:0] vpn_in;
output [25:0] pfn_out;
output hit;
output valid_out;

reg[60:0] cam_content [63:0];//[60:60] valid [59:34] pfn [33:0] vpn
wire [63:0] entry_hit;
```

```
assign entry_hit[0] = (vpn_in==cam_content[0][33:0]);
assign entry_hit[1] = (vpn_in==cam_content[1][33:0]);
……
assign entry_hit[63] = (vpn_in==cam_content[63][33:0]);

assign hit = |entry_hit;

assign pfn_out =  {26{entry_hit[ 0]}} & cam_content[0][59:34] |
                  {26{entry_hit[ 1]}} & cam_content[1][59:34] |
……
                  {26{entry_hit[63]}} & cam_content[63][59:34];

assign valid_out = entry_hit[ 0] && cam_content[ 0][60] ||
                    entry_hit[ 1] && cam_content[ 1][60] ||
                    …
                    entry_hit[63] && cam_content[63][60];
endmodule
```