Meltdown 与 Spectre 攻击

H.1. 多次读取同一内存地址时，第二次访问通常比第一次快，原因是什么？

H.2. 在 Flush+Reload 技术中，为什么使用大小为 $256 * 4096$ 的数组，而不是大小为 256 的数组？

H.3. 如何利用 CPU 缓存侧信道传递数值 89？

H.4. 已定义一个大小为 1024 字节的数组。假设 CPU 缓存块大小为 128 字节（访问地址 x 时，x 到 $x+127$ 的内存会进入缓存）。使用该数组和 Flush+Reload 技术最多能传递多少个不同的值？

H.5. 一个秘密数存储在内核地址 `0xfb102000` 处。以下用户态程序尝试访问并打印它。程序会发生什么？

```
#include <stdio.h>
int main()
{
  char *kernel_data_addr = (char*)0xfb102000;  ①
  char kernel_data = *kernel_data_addr;        ②
  printf("I have reached here.\n");            ③
  return 0;
}
```

H.6. 一个秘密数存储在内核地址 `0xfb102000` 处。请用自己的语言描述如何利用 Meltdown 读出这个秘密数。

H.7. 为提高 Meltdown 攻击的成功率，最好让目标内核数据事先进入 CPU 缓存。为什么？

H.8. 如果 CPU 不支持乱序执行，还能实施 Meltdown 攻击吗？禁用乱序执行有什么代价？

H.9. KAISER 可以阻止 Meltdown。其核心思想是在用户态页表中不映射内核内存。请解释其原理。

H.10. 在 Spectre 攻击中，为什么需要训练 CPU？

H.11. 如果 CPU 不进行乱序执行，还能实施 Spectre 攻击吗？

H.12. 在 Spectre 攻击场景中，受保护的内存与攻击者程序位于同一进程，为什么攻击者仍不能直接访问该内存？

H.13. 运行教材中的 `SpectreAttack.c` 时，为什么总能在缓存中发现 `array[0*4096 + DELTA]`？

H.14. 你的程序运行在沙箱中，只允许通过以下沙箱 API 访问受保护缓冲区的前 100 个元素。缓冲区地址为 `0xbfff0200`。地址 `0xbfff0100` 处存有一个秘密数，但沙箱禁止程序直接访问它。能否获得这个秘密数？如果可以，请说明方法。

```
int low  = 0;
int high = 100;
int restrictedAccess(int x)
{
  if (x > low && x < high) {
     return buffer[x];
  } else {
     return 0;
  }
}
```