https://mp.weixin.qq.com/s/Pd6dOXaMQFUHfAUnOhnwtw

为什么synchronized无法禁止指令重排，却能保证有序性？

这是个好问题，这个问题我曾经也思考过，也查阅过很多资料，甚至还去看过hotsopt的源码。

为了进一步提升计算机各方面能力，在硬件层面做了很多优化，如处理器优化和指令重排等，但是这些技术的引入就会导致有序性问题。

先告诉面试官你知道什么是有序性问题，也知道是什么原因导致的有序性问题

我们也知道，最好的解决有序性问题的办法，就是禁止处理器优化和指令重排，就像volatile中使用内存屏障一样。

表明你知道啥是指令重排，也知道他的实现原理

但是，虽然很多硬件都会为了优化做一些重排，但是在Java中，不管怎么排序，都不能影响单线程程序的执行结果。这就是as-if-serial语义，所有硬件优化的前提都是必须遵守as-if-serial语义。

重点！解释下什么是as-if-serial语义，因为这是这道题的第一个关键词，答上来就对了一半了

再说下synchronized，他是Java提供的锁，可以通过他对Java中的对象加锁，并且他是一种排他的、可重入的锁。

装X项，不留痕迹的展示自己对锁了解的比较多

所以，当某个线程执行到一段被synchronized修饰的代码之前，会先进行加锁，执行完之后再进行解锁。在加锁之后，解锁之前，其他线程是无法再次获得锁的，只有这条加锁线程可以重复获得该锁。

介绍synchronized的原理，这是本题的第二个关键点，到这里基本就可以拿满分了。

synchronized通过排他锁的方式就保证了同一时间内，被synchronized修饰的代码是单线程执行的。所以呢，这就满足了as-if-serial语义的一个关键前提，那就是**单线程**，因为有as-if-serial语义保证，单线程的有序性就天然存在了。