volatile底层实现

volatile修饰的共享变量进行写操作的时候会多出**Lock前缀的指令**，该指令在多核处理器下会引发两件事情：

1. 将当前处理器缓存行数据刷写到系统主内存。
2. 这个刷写回主内存的操作会使其他CPU缓存的该共享变量内存地址的数据无效。

这样就保证了多个处理器的缓存是一致的，对应的处理器发现自己缓存行对应的内存地址被修改，就会将当前处理器缓存行设置无效状态，当处理器对这个数据进行修改操作的时候会重新从主内存中把数据读取到缓存里。

> 指令重排序对单线程没有什么影响，它不会影响程序的运行结果，反而会优化执行性能，但会影响多线程的正确性。
>
> Java因为指令重排序，优化我们的代码，让程序运行更快，也随之带来了多线程下，指令执行顺序的不可控。
>
> **volatile的底层是通过lock前缀指令、内存屏障来实现的。**

lock前缀指令实际上相当于一个内存屏障（也成内存栅栏），内存屏障会提供3个功能：

- 它确保指令重排序时不会把其后面的指令排到内存屏障之前的位置，也不会把前面的指令排到内存屏障的后面；即在执行到内存屏障这句指令时，在它前面的操作已经全部完成；
- 它会强制将对缓存的修改操作立即写入主存；
- 如果是写操作，它会导致其他CPU中对应的缓存行无效。