深入Java底层：内存屏障与JVM并发详解Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/645/2021_2022__E6_B7_B1_E5_85_A5Java_c104_645014.htm 内存屏障，又称内存栅栏，是一组处理器指令，用于实现对内存操作的顺序限制。本文假定读者已经充分掌握了相关概念和Java内存模型，不讨论并发互斥、并行机制和原子性。内存屏障用来实现并发编程中称为可见性（visibility）的同样重要的作用。内存屏障为何重要？对主存的一次访问一般花费硬件的数百次时钟周期。处理器通过缓存（caching）能够从数量级上降低内存延迟的成本。这些缓存为了性能重新排列待定的内存操作的顺序。也就是说，程序的读写操作不一定会按照它要求处理器的顺序执行。当数据是不可变的，同时/或者数据限制在线程范围内，这些优化是无害的。如果把这些优化与对称多处理（symmetric multi-processing）和共享可变状态（shared mutable state）结合，那么就是一场噩梦。当基于共享可变状态的内存操作被重新排序时，程序可能行为不定。一个线程写入的数据可能被其他线程可见，原因是数据写入的顺序不一致。适当的放置内存屏障通过强制处理器顺序执行待定的内存操作来避免这个问题。内存屏障的协调作用 内存屏障不直接由JVM暴露，相反它们被JVM插入到指令序列中以维持语言层并发原语的语义。我们研究几个简单Java程序的源代码和汇编指令。首先快速看一下Dekker算法中的内存屏障。该算法利用volatile变量协调两个线程之间的共享资源访问。请不要关注该算法的出色细节。哪些部分是相关的？每个线程通过发信号试图进入代码第一行的关键区域。如果线程在第三行意识到冲突（

两个线程都要访问)，通过turn变量的操作来解决。在任何时刻只有一个线程可以访问关键区域。 1. // code run by first thread 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com