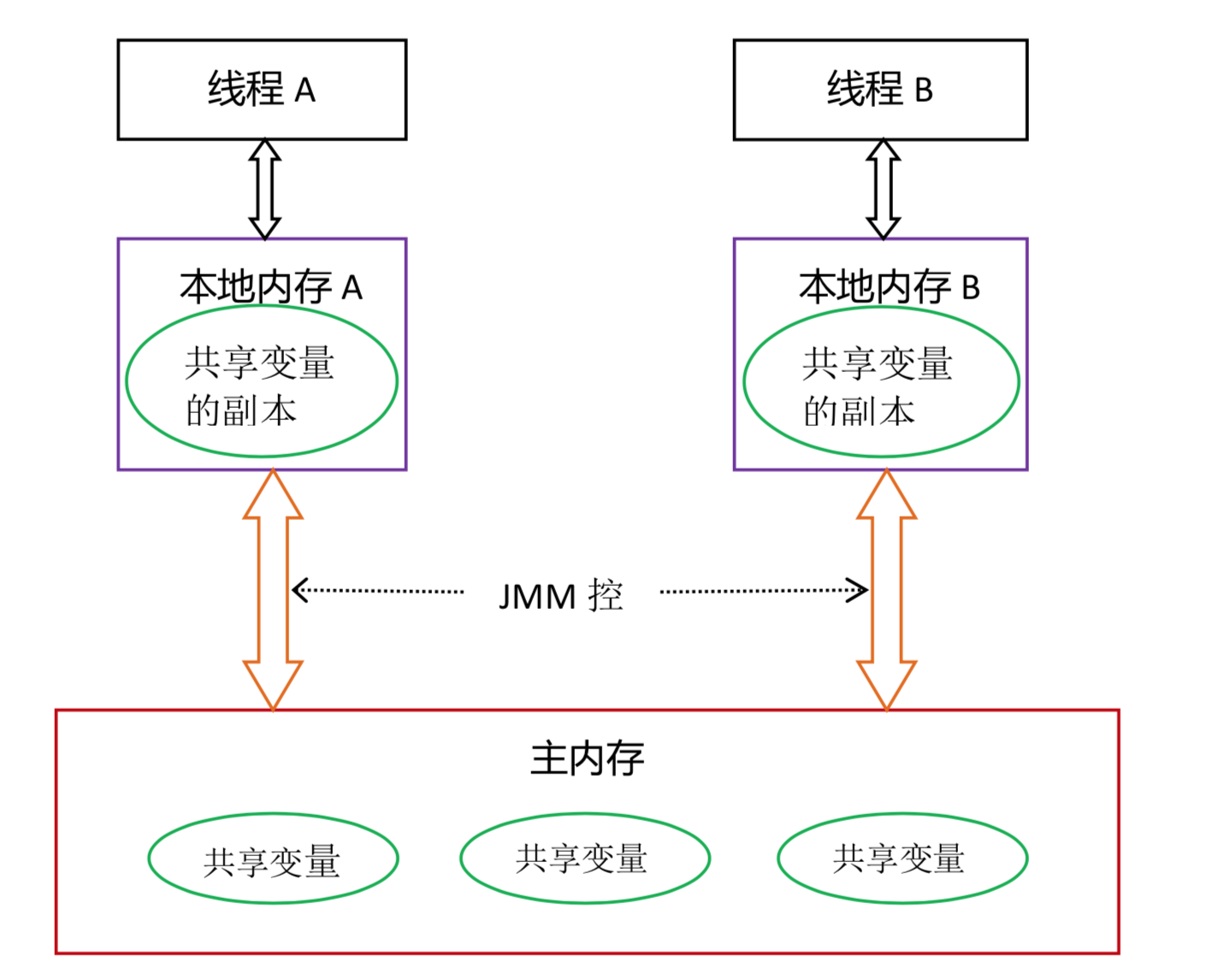
1. volatile: JVM提供的轻量级的同步机制。
   1. 保证可见性
   2. 不保证原子性
   3. 禁止指令重排序
2. 内存可见性



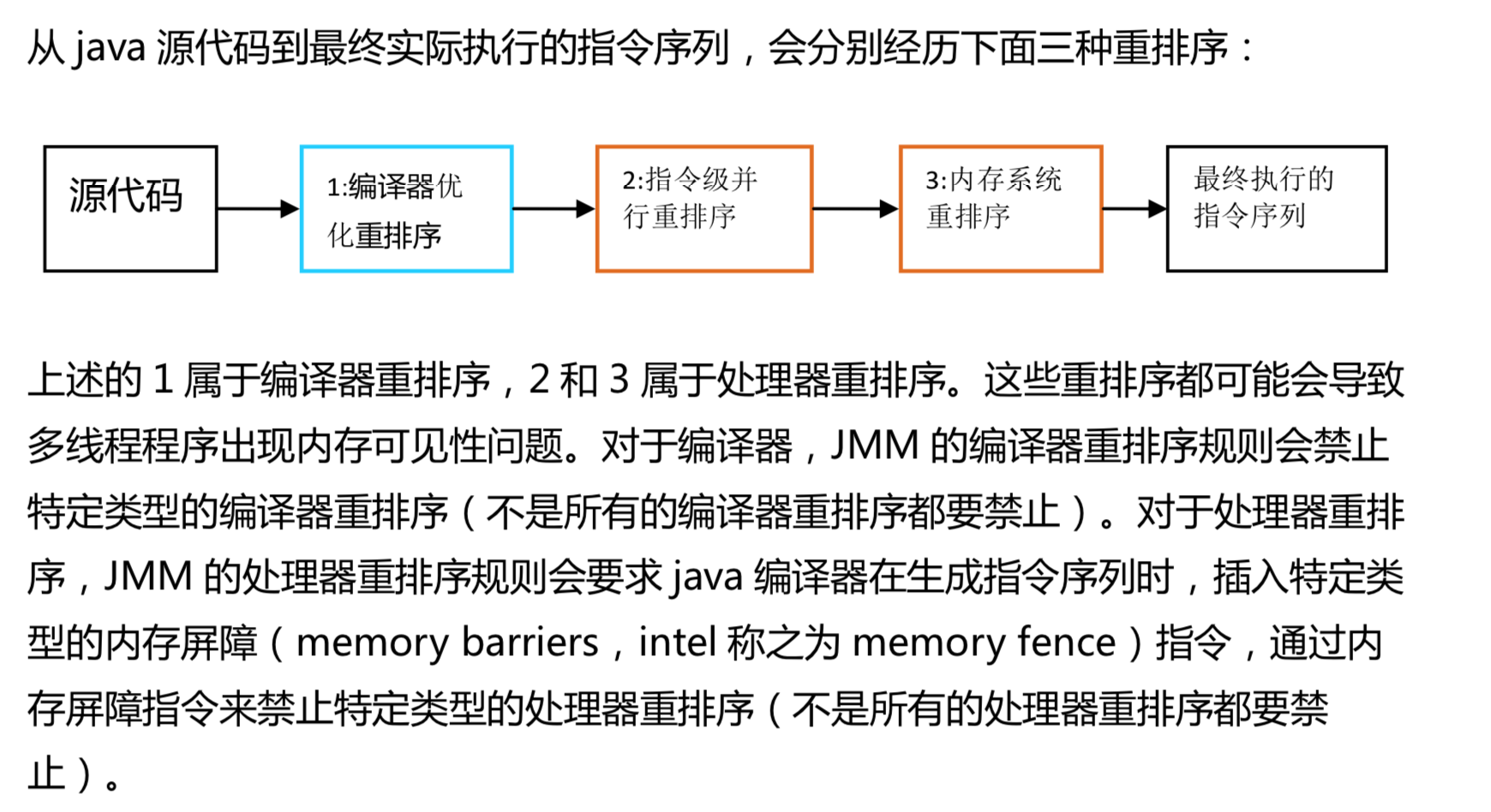
在当前的Java内存模型下，线程可以把变量保存在本地内存(比如机器的寄存器)中，而不是直接在主存中进行读写。这就可能造成一个线程在主存中修改了一个变量的值，而另外一个线程还继续使用它在寄存器中的变量值的拷贝，造成数据的不一致。要解决这个问题，只需要把该变量声明为volatile（不稳定的）即可，这就指示JVM，这个变量是不稳定的，每次使用它都到主存中进行读取。一般说来，多任务环境下各任务间共享的标志都应该加volatile修饰。

volatile修饰的成员变量在每次被线程访问时，都强迫从共享内存中重读该成员变量的值。而且，当成员变量发生变化时，强迫线程将变化值回写到共享内存。这样在任何时刻，两个不同的线程总是看到某个成员变量的同一个值。

1. 原子性

number++操作在多线程环境下是非线程安全的，需要使用AtomicInteger或者加锁解决

1. 重排序



1. 案例
   1. 单例模式Double Check Lock写法instance必须采用volatile修饰防止重排序造成的类初始化未完成就使用的问题