57 什么是指令重排序？为什么要重排序？

本课时我们主要介绍什么是重排序？为什么要重排序？

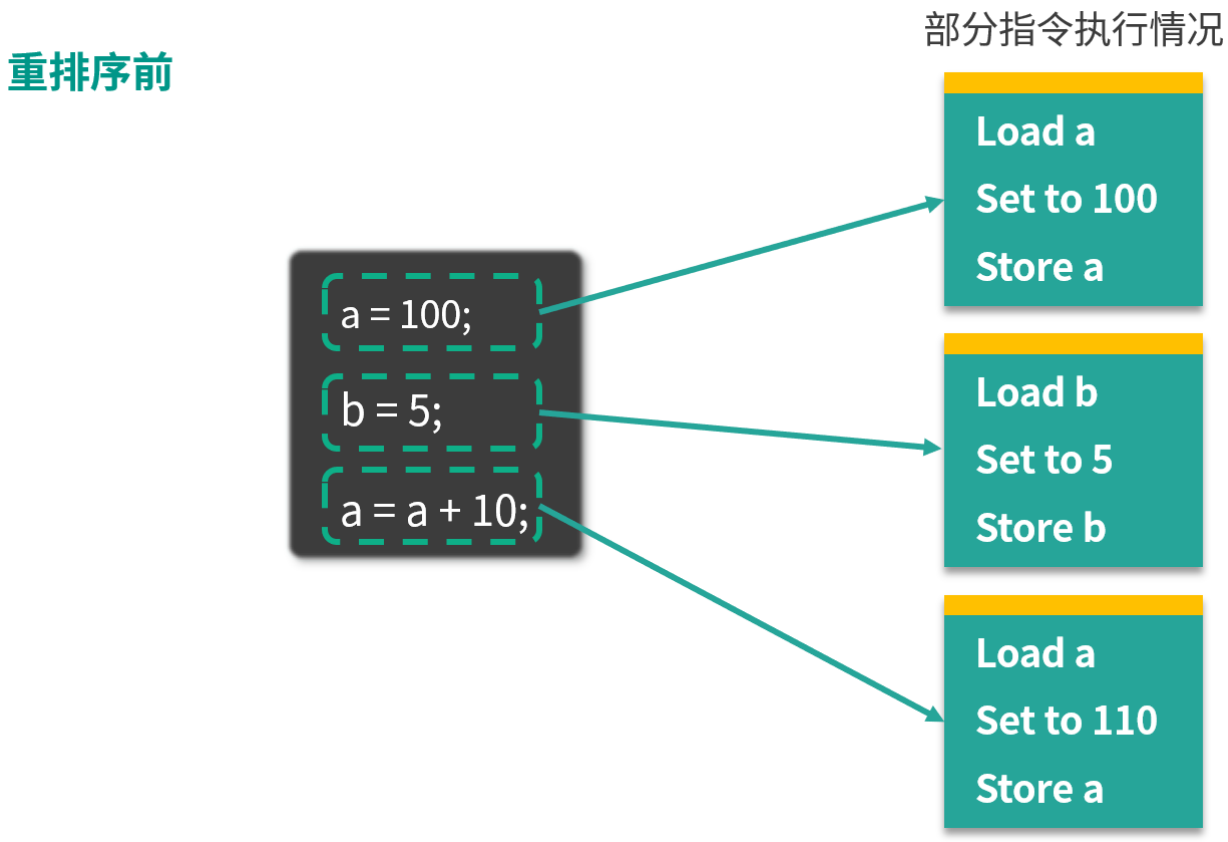
什么是重排序

假设我们写了一个 Java 程序，包含一系列的语句，我们会默认期望这些语句的实际运行顺序和写的代码顺序一致。但实际上，编译器、JVM 或者 CPU 都有可能出于优化等目的，对于实际指令执行的顺序进行调整，这就是**重排序**。

重排序的好处：提高处理速度

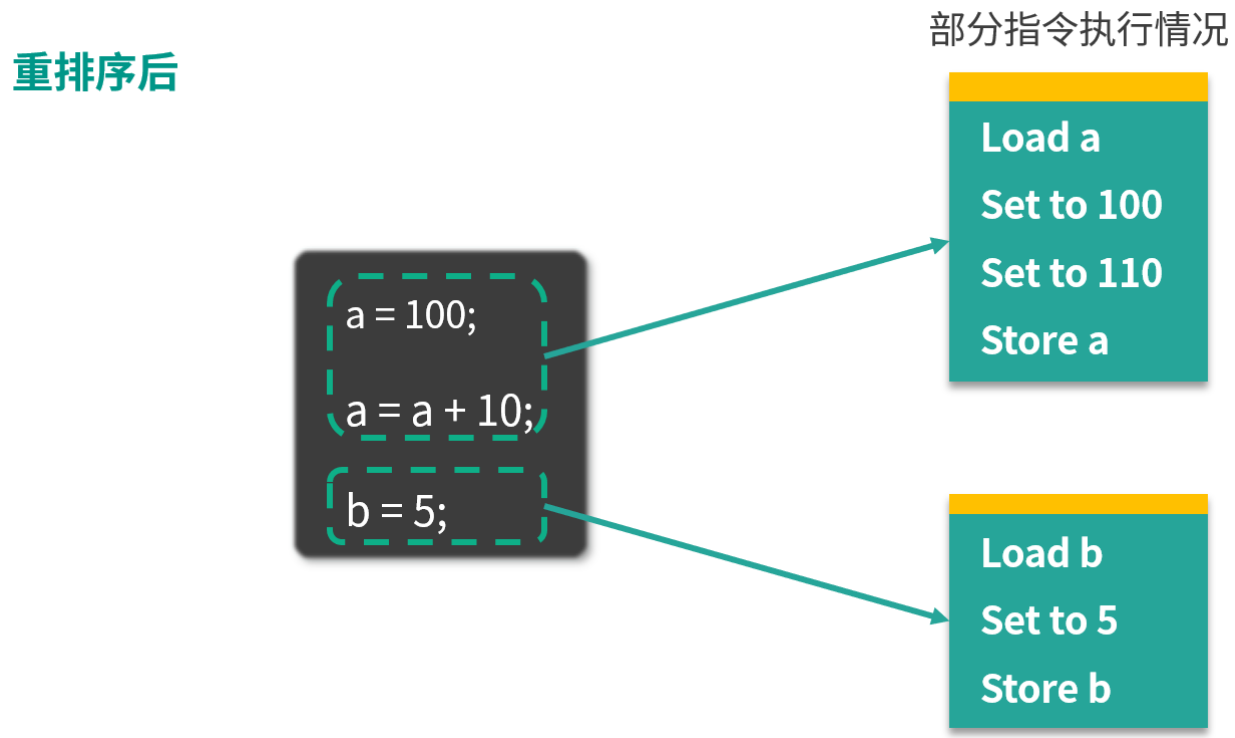
你可能感到很困惑，为什么要重排序？这样做有什么好处呢？

我们来举一个具体的例子。



图中左侧是 3 行 Java 代码，右侧是这 3 行代码可能被转化成的指令。可以看出 a = 100 对应的是 Load a、Set to 100、Store a，意味着从主存中读取 a 的值，然后把值设置为 100，并存储回去，同理， b = 5 对应的是下面三行 Load b、Set to 5、Store b，最后的 a = a + 10，对应的是 Load a、Set to 110、Store a。如果你仔细观察，会发现这里有两次“Load a”和两次“Store a”，说明存在一定的重排序的优化空间。

经过重排序之后，情况如下图所示：



重排序后， a 的两次操作被放到一起，指令执行情况变为 Load a、Set to 100、Set to 110、 Store a。下面和 b 相关的指令不变，仍对应 Load b、 Set to 5、Store b。

可以看出，重排序后 a 的相关指令发生了变化，节省了一次 Load a 和一次 Store a。重排序通过减少执行指令，从而提高整体的运行速度，这就是重排序带来的优化和好处。

重排序的 3 种情况

下面我们来看一下重排序的 3 种情况。

（1）编译器优化

编译器（包括 JVM、JIT 编译器等）出于优化的目的，例如当前有了数据 a，把对 a 的操作放到一起效率会更高，避免读取 b 后又返回来重新读取 a 的时间开销，此时在编译的过程中会进行一定程度的重排。不过重排序并不意味着可以任意排序，它需要需要保证重排序

后，不改变单线程内的语义，否则如果能任意排序的话，程序早就逻辑混乱了。

（2）CPU 重排序

CPU 同样会有优化行为，这里的优化和编译器优化类似，都是通过乱序执行的技术来提高整体的执行效率。所以即使之前编译器不发生重排，CPU 也可能进行重排，我们在开发中，一定要考虑到重排序带来的后果。

（3） 内存的“重排序”

内存系统内不存在真正的重排序，但是内存会带来看上去和重排序一样的效果，所以这里的“重排序”打了双引号。由于内存有缓存的存在，在 JMM 里表现为主存和本地内存，而主存和本地内存的内容可能不一致，所以这也会导致程序表现出乱序的行为。

举个例子，线程 1 修改了 a 的值，但是修改后没有来得及把新结果写回主存或者线程 2 没来得及读到最新的值，所以线程 2 看不到刚才线程 1 对 a 的修改，此时线程 2 看到的 a 还是等于初始值。但是线程 2 却可能看到线程 1 修改 a 之后的代码执行效果，表面上看起来像是发生了重顺序。

总结

以上就是本课时的内容。本课时我们首先用一个例子介绍了什么是重排序，然后分析了重排序所能带来的好处，并介绍了可能发生重排序的 3 种情况

上一页 下一页