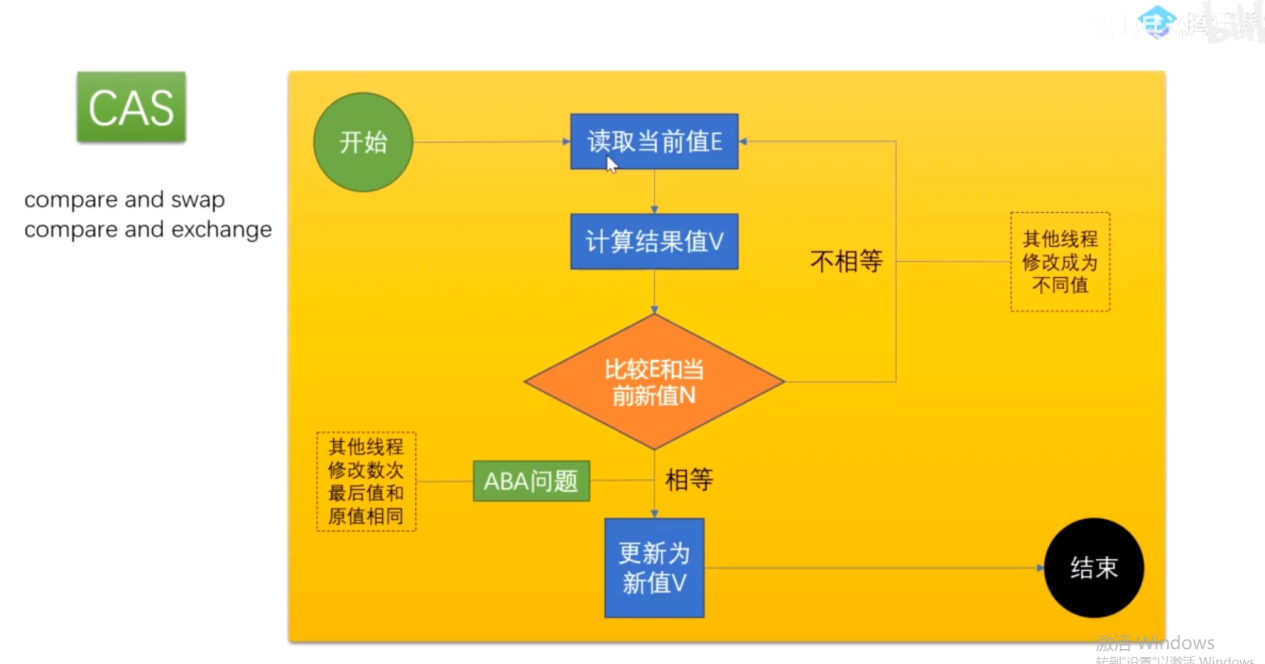
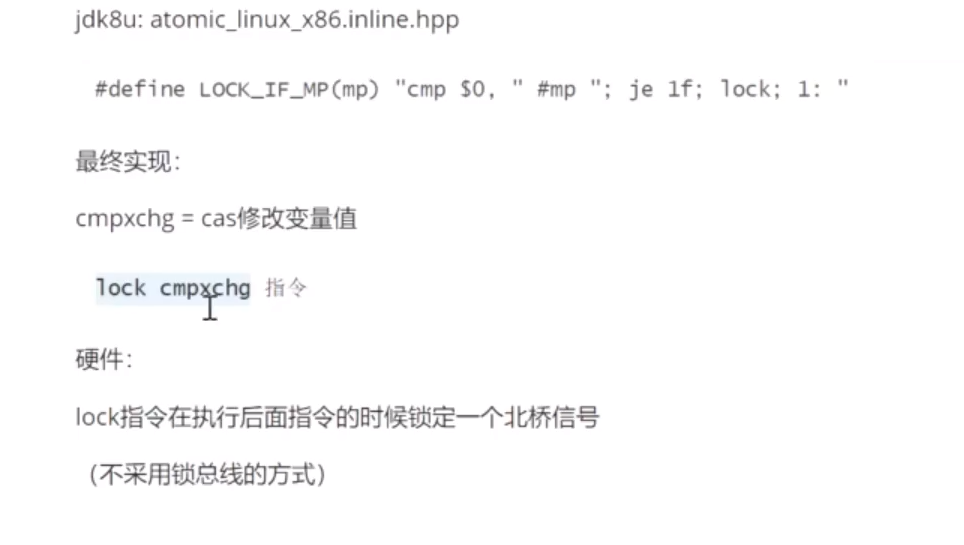
1、CAS顶层原理（自旋）----->lock cmpxchg





cmpxchg不保证原子性

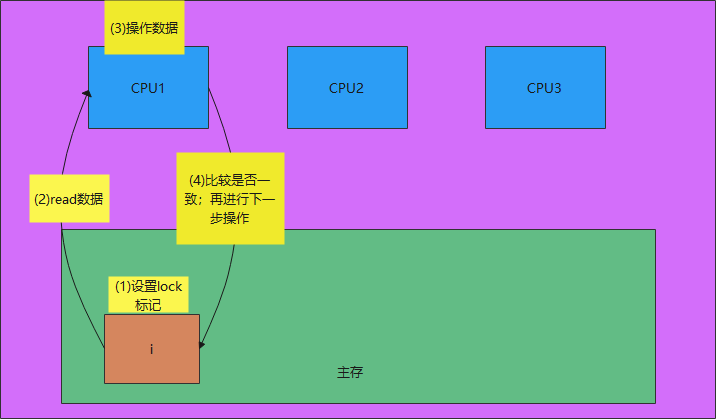
通过lock锁总线保证原子性

从内存中读取值V到CPU中为E，进行运算得到N，然后写回内存，在此之前判断V与E是否还相等，相等（没被人改过）则写回内存，不相等（被人改过）重新读取值运算。

1. CAS底层原理，lock cmpxchg汇编指令

Lock：锁总线，其他线程写不进去

原子指令在软件上看来逻辑并不复杂，但在微架构上看，成本是很高的。如果我们把CPU 和内存都看做是总线上的一个个独立的实体，有一个CPU要做CAS指令，这个CPU需要先从内存中读一个值到CPU，运算完成后，进行比较，比较完成，锁总线（保证其他线程写不进去）写回数据。



1. CAS问题：ABA问题

例子1： 假设有一个变量 A ，修改为B，然后又修改为了 A，实际已经修改过了，但 CAS 可能无法感知，造成了不合理的值修改操作。

例子2： 引用R指向一个对象，一个线程把R指向的对象改了，另外一个线程发现引用R没变，CAS就无法感知到R指向对象的变化。

如果是基本类型，无所谓

如果是引用类型，可能就会出现问题

解决方法：

1. 加版本号
2. 加标签