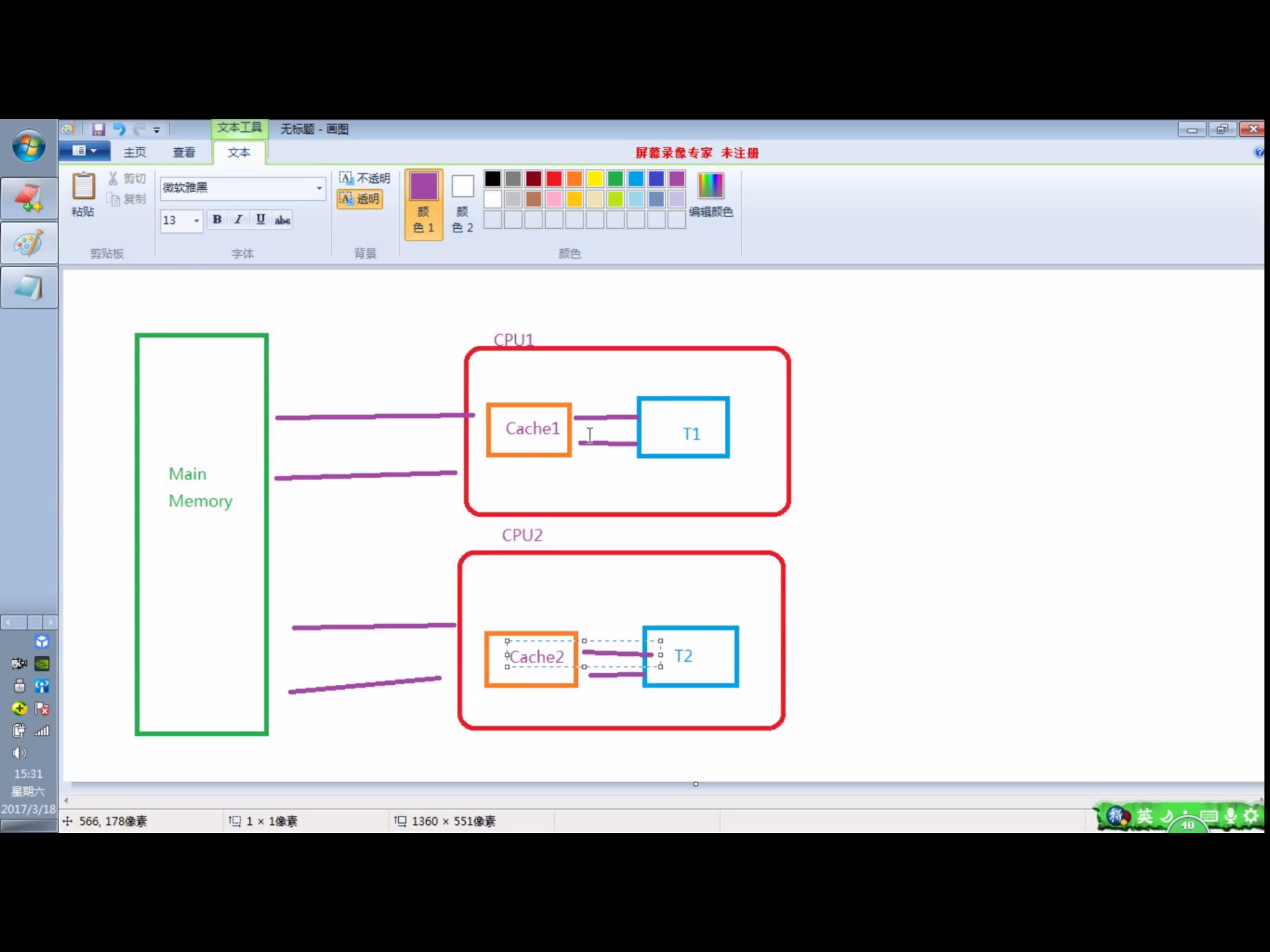
java的内存模型



java每次读取局部变量的时候，如果没有volatile修饰，并且这个数据只读不写，那么该变量在新建线程的时候会从单cpu的cach里面读取，如果其他线程对这个变量进行修改，但是在该线程不会被发现，如果使用了volatile的话，将这个变量放在主内存里面。对于不同线程同步

i = i+1

mem -> i->cache i+1 ->cache ->main memory

main mem

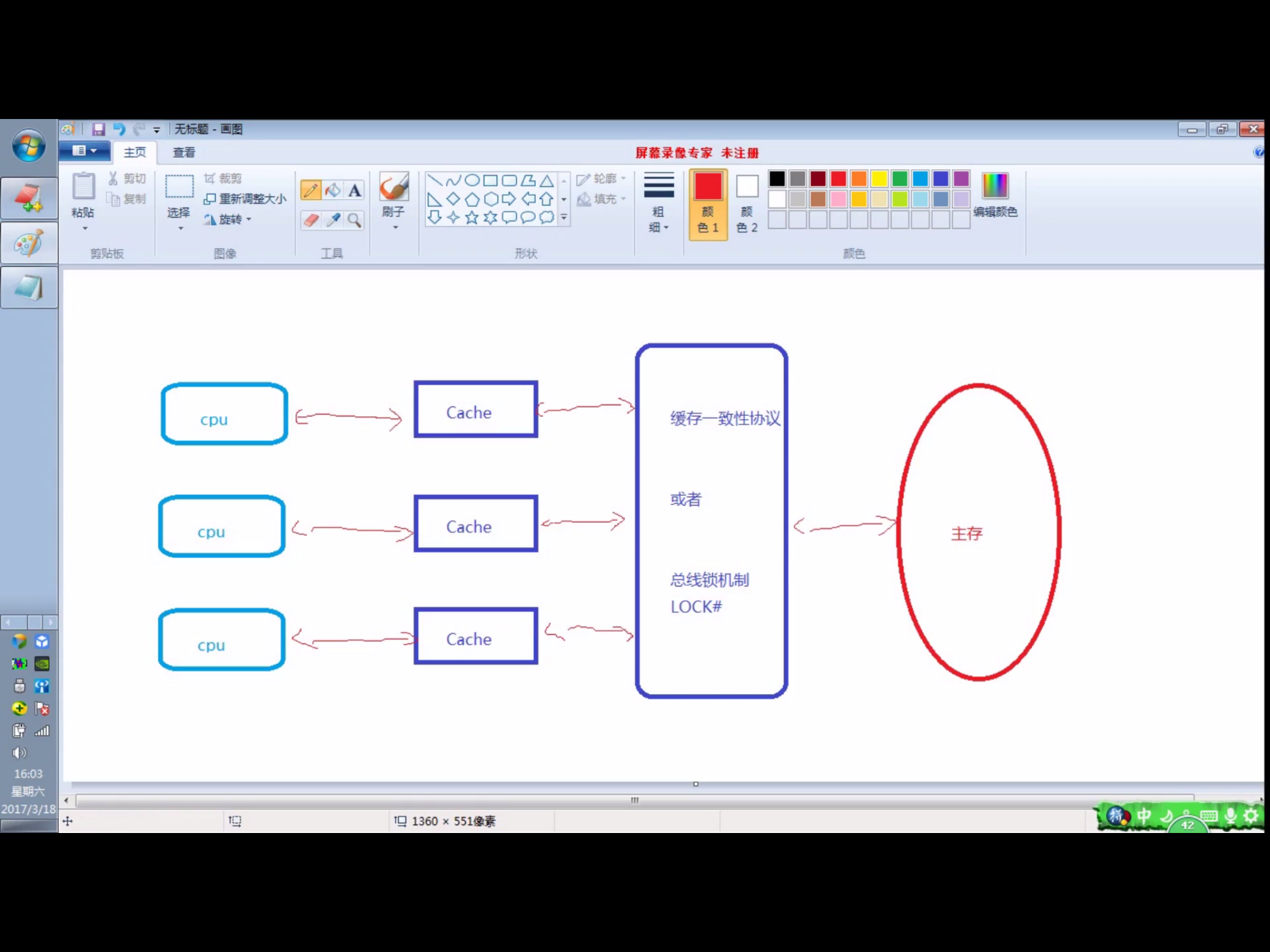
1 给数据总线加锁 或导致串行化

2.给cpu高速缓存一致性协议

Intel mesi

核心思想

1.当cpu写入数据时，如果发现该变量被共享（也就是说，在其它cpu中也存在该变量的副本），会发出一个信号，通知其他cpu该变量的缓存无效



1.原子性 ： 存钱同多同少 ，一个操作要不都成功，要不都失败，中间不能因为任何因素中断

java 对基本类型的数据读取和赋值保证了原子性，要么都成功，要么都失败，这些操作不可被中断

a = 10 原子性

b =a 不满足。1.read 2assign b

c++; 不满足 1.read 2 add 3 assign to c；

2.可见性 ：每个数据对于所有线程的可见的，相同的

使用volatile保证可见性，保证变量被修改之后会立刻被写入主存当中，而不是到缓存里面拿数据

3.有序性 ： java 有依赖关系的具有有序性

happens-before原则（happens-before relationship）

3.1代码的执行顺序，编写在前面的发生在编写的后面的

3.2 unlock必须发生在lock之后

3.3volatile修饰的变量，对该变量的写操作必须先于在该变量读操作

3.4传递规则，操作A先于B,B先于C，那么A肯定先于C

3.5线程启动规则，start方法肯定先于线程run

3.6线程中断规则，interrupt这个动作，必须发生在捕获该动作之前

3.7对象销毁规则，初始化必须发生在finalize之前

3.8现成的终结规则，

--------Thread 1-----------

init = true

obj = createObj()

--------Thread 2------------

while(!init)

sleep();

useTheObj(obj);

volatile 关键字

一旦一个变量被volatile修饰，具备两层语意

1.保证了不同线程间的可见性

2.禁止对其进行重重排序，保证了有序性，保证重排序时候不会把后面的指令放到屏障的前面，也不会把前面的放到后面

3，并未保证原子性

4.强制对缓存的修改操作，立刻写入主存

5.如果是写操作，会导致其他cpu中的缓存失效