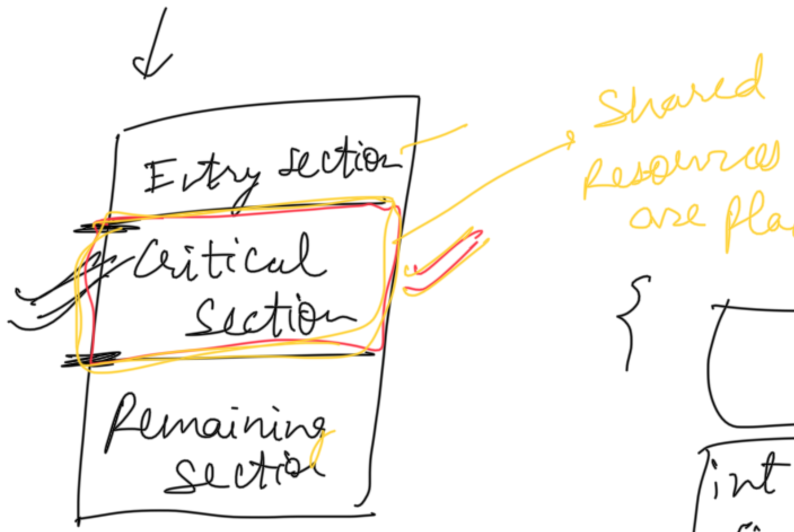
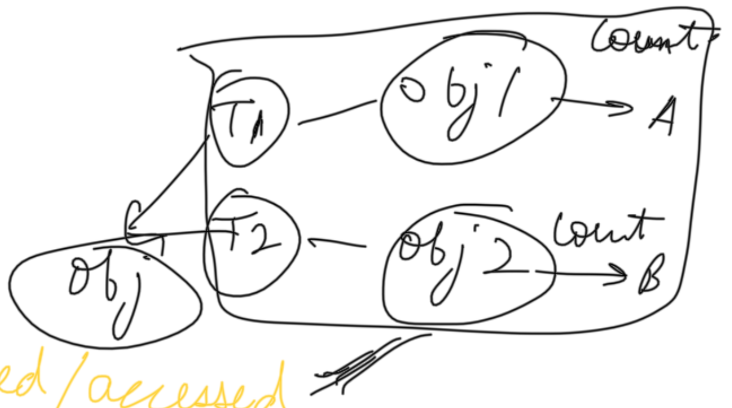
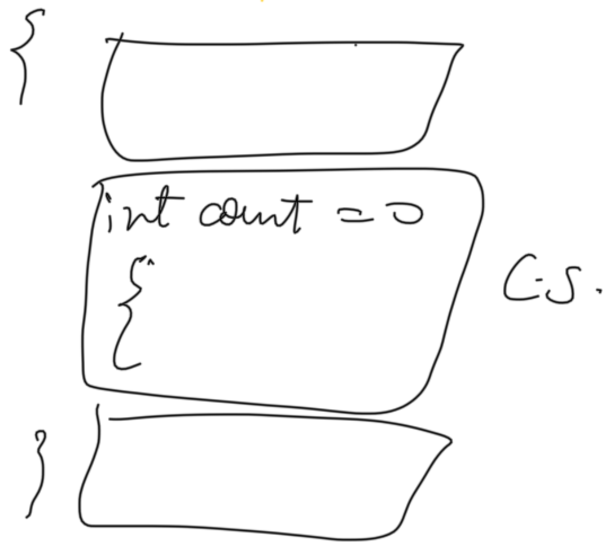
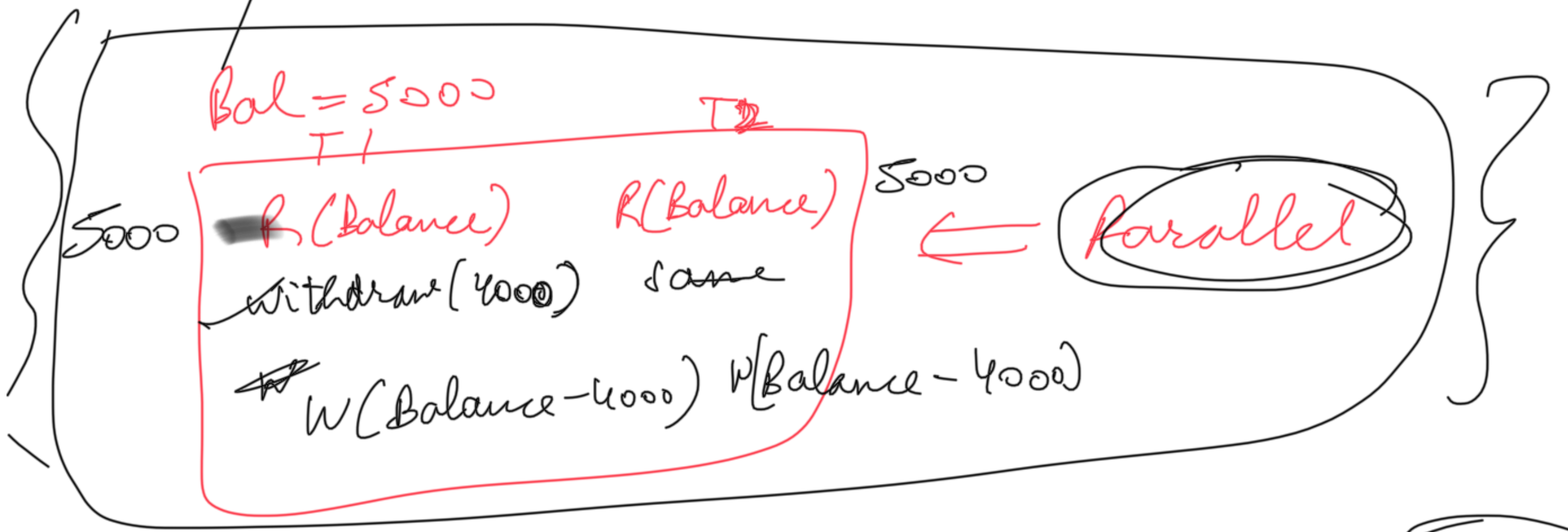
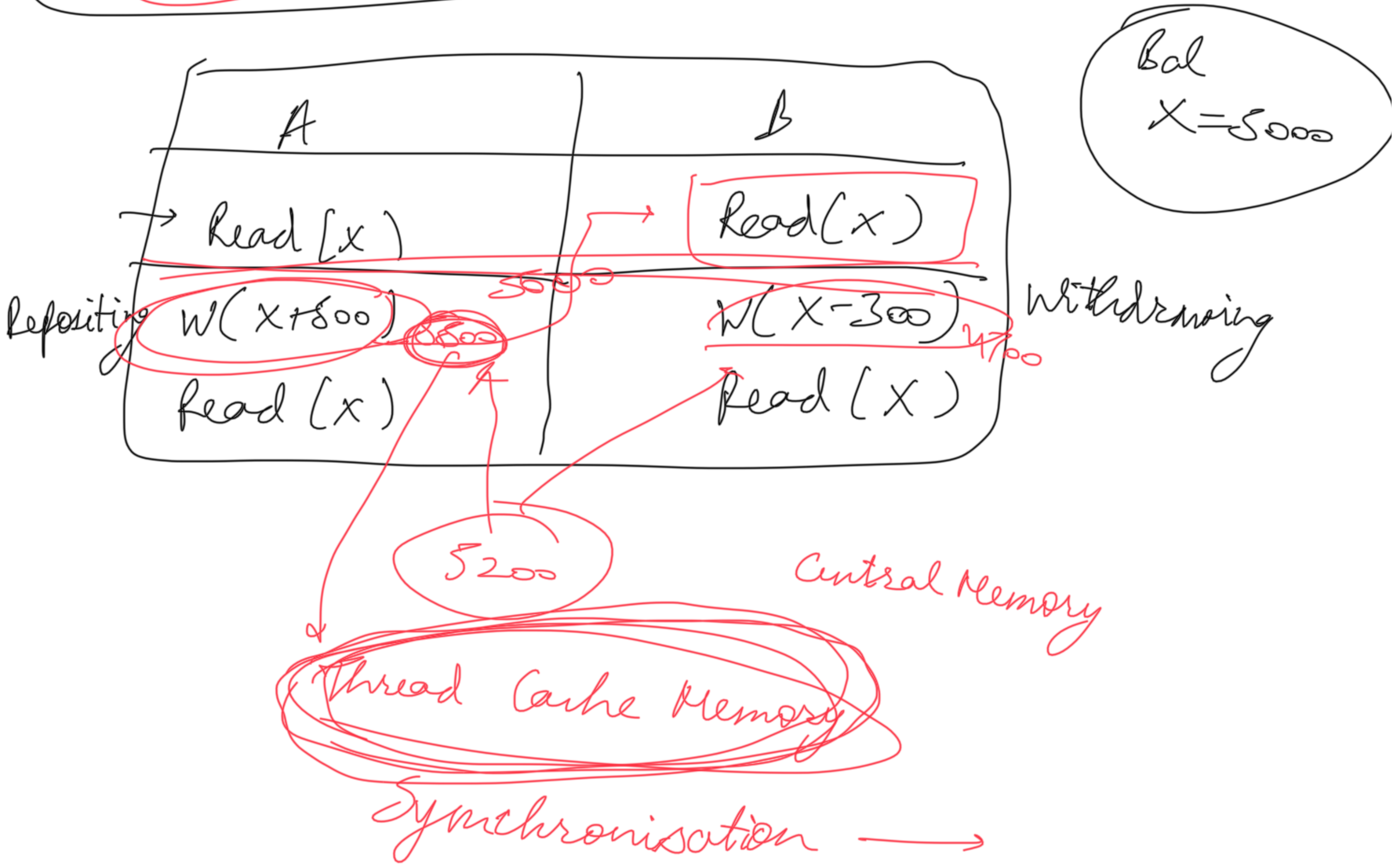
Critical Section



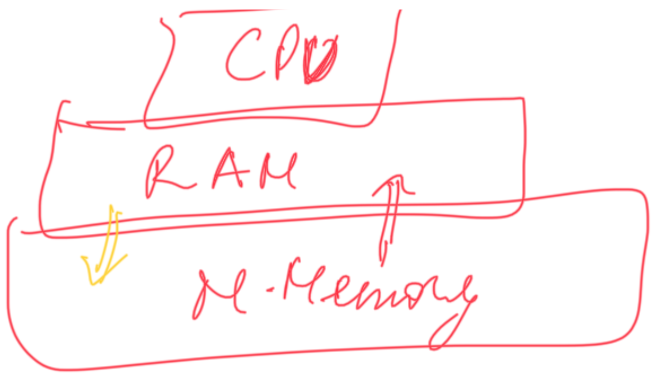


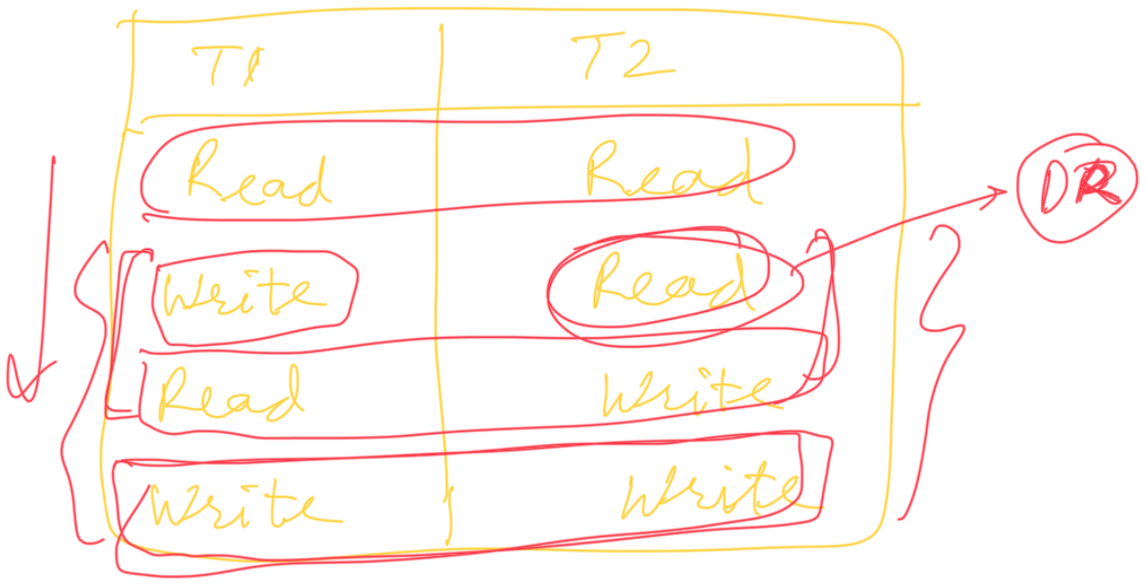






Synchronisation →





2 ways of making class thread Safe

⇓

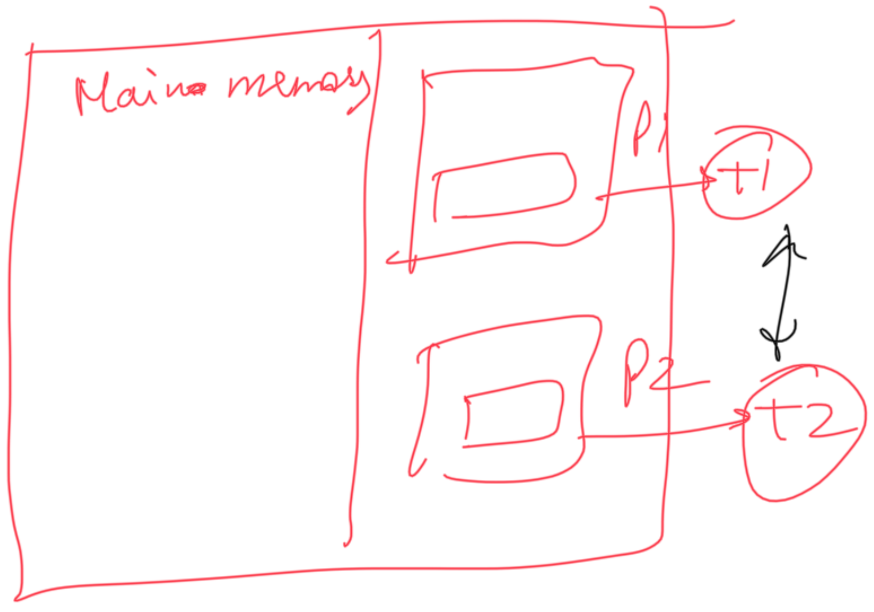
method or class instance can be used by multiple threads without any problem

① Synchronized ~~Key~~

② Using volatile keyword

Shared object

static int count = 0



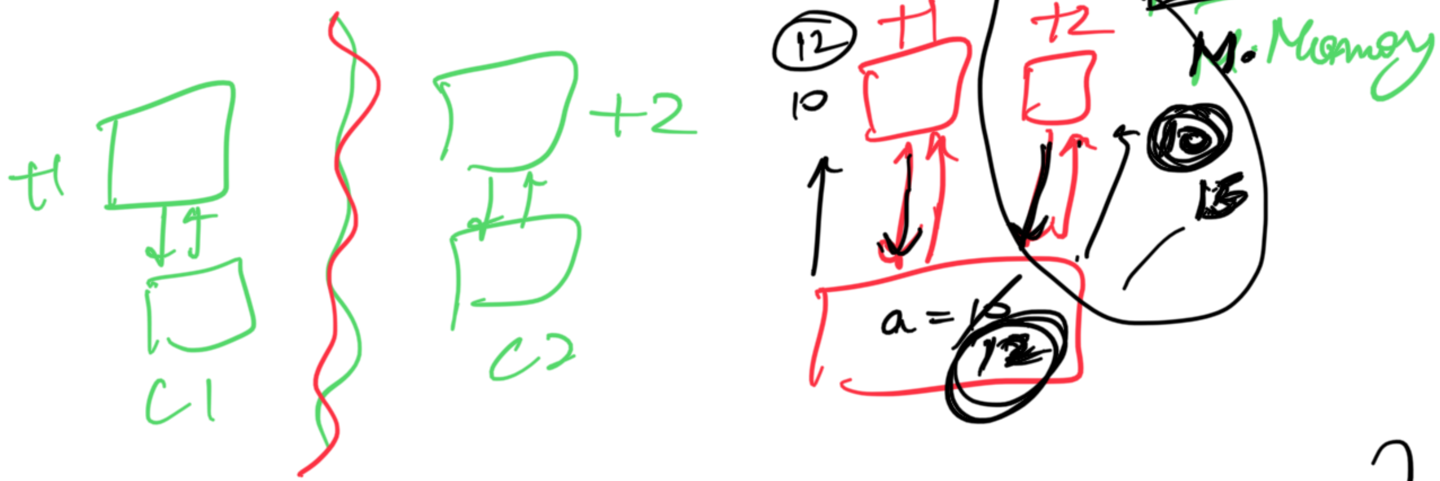
data inconsistency

① Mutual Exclusion → Only one thread can execute a piece of code ~~ate~~ at a time [C·Section]

② Visibility → Changes made by t1 are visible to other threads

Volatile keyword.

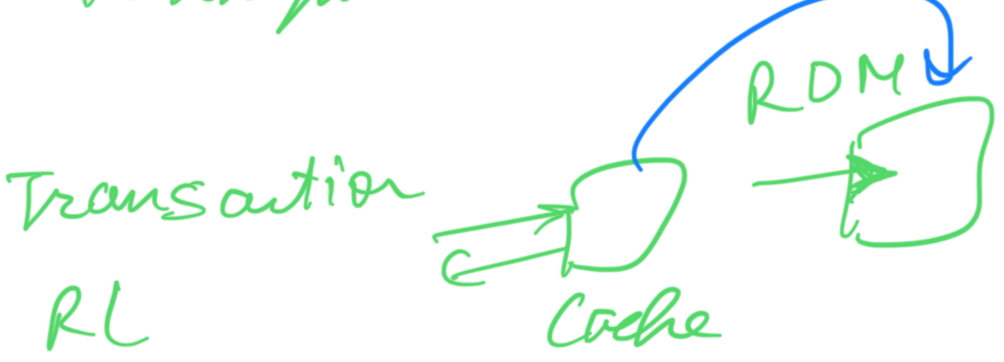
① variable won't be cached



[Tells the compiler that variable must not be cache]

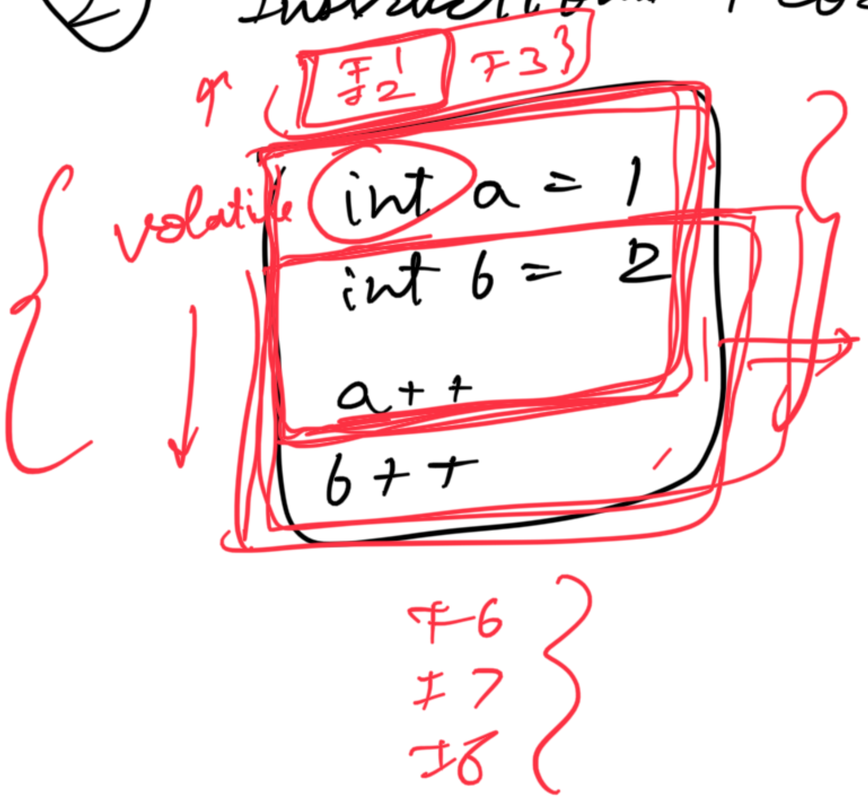
device drivers / embedded systems

Cache → Optimisation technique



Don't do the optimisation

② Instructions Reordering



```
int a = 1
int b = 2

a++
b++
```

JVM and CPU

```
int a=1
a++

int b=1
b++
```