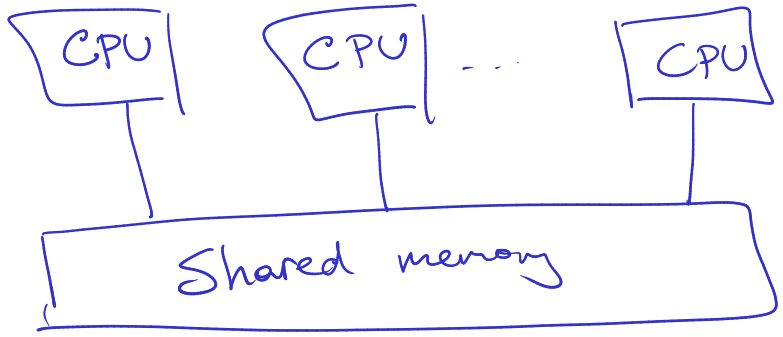
x86-TSO

Shared-mem concurrency.
Weak memory complexity
Clarifies model for x86.
Not just proofs.

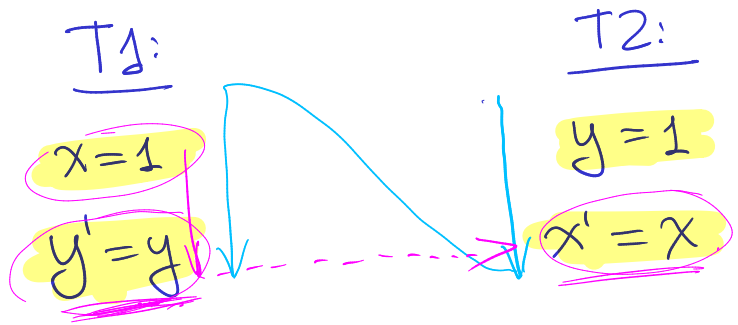
Sequential consistency (SC)



Atomic actions: Mem read(a) → v
Mem write(a,v) → ✓

- SC used in real HW.
- ☆ — Clean abstraction for devs.

Example



main:

x = 0
y = 0

Spawn T1, T2

wait

print (x', y').

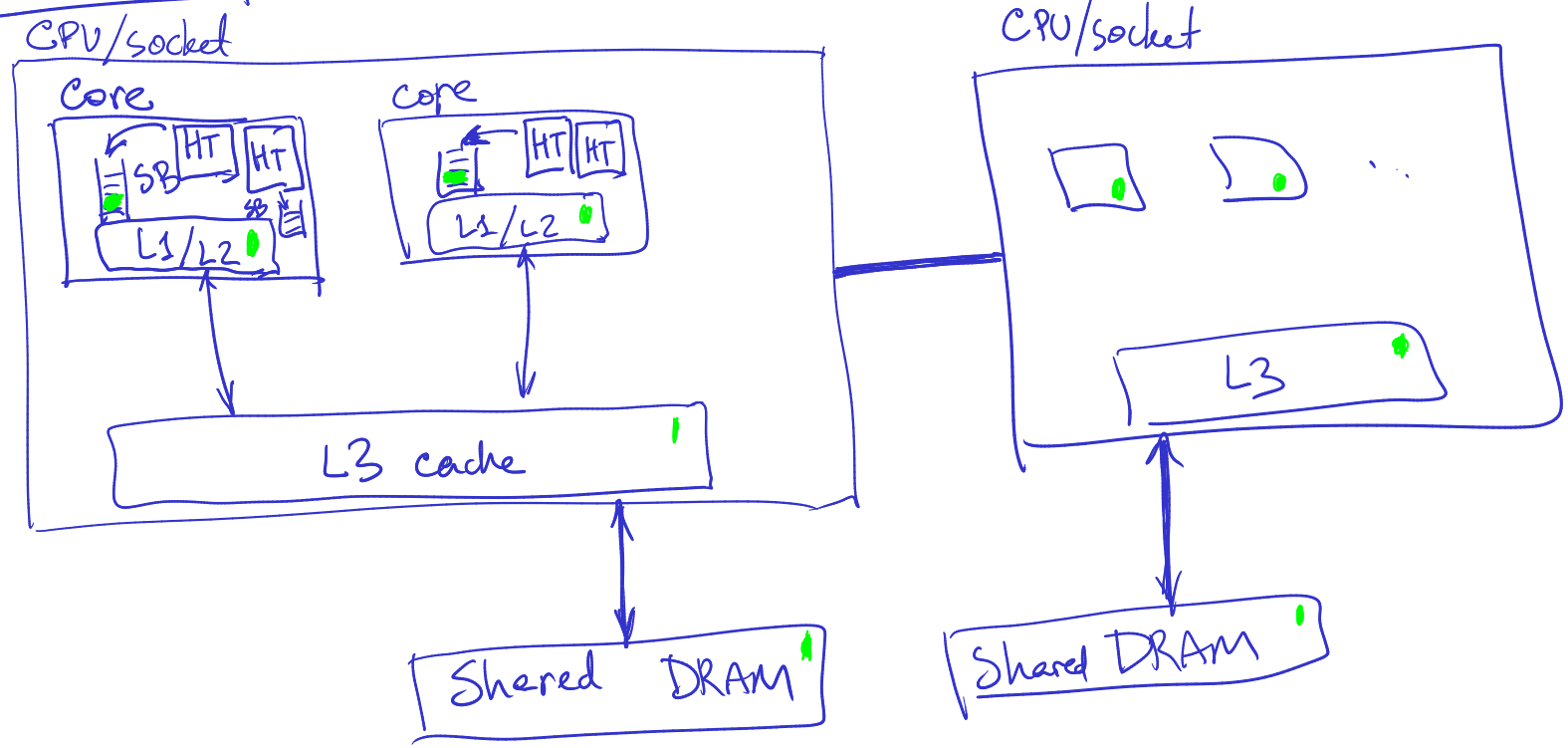
Results:

OK in SC.: (1, 1), (0, 1), (1, 0)

× (0, 0)

OK in weak mem.

Real HW for shared mem.



Out-of-order exec

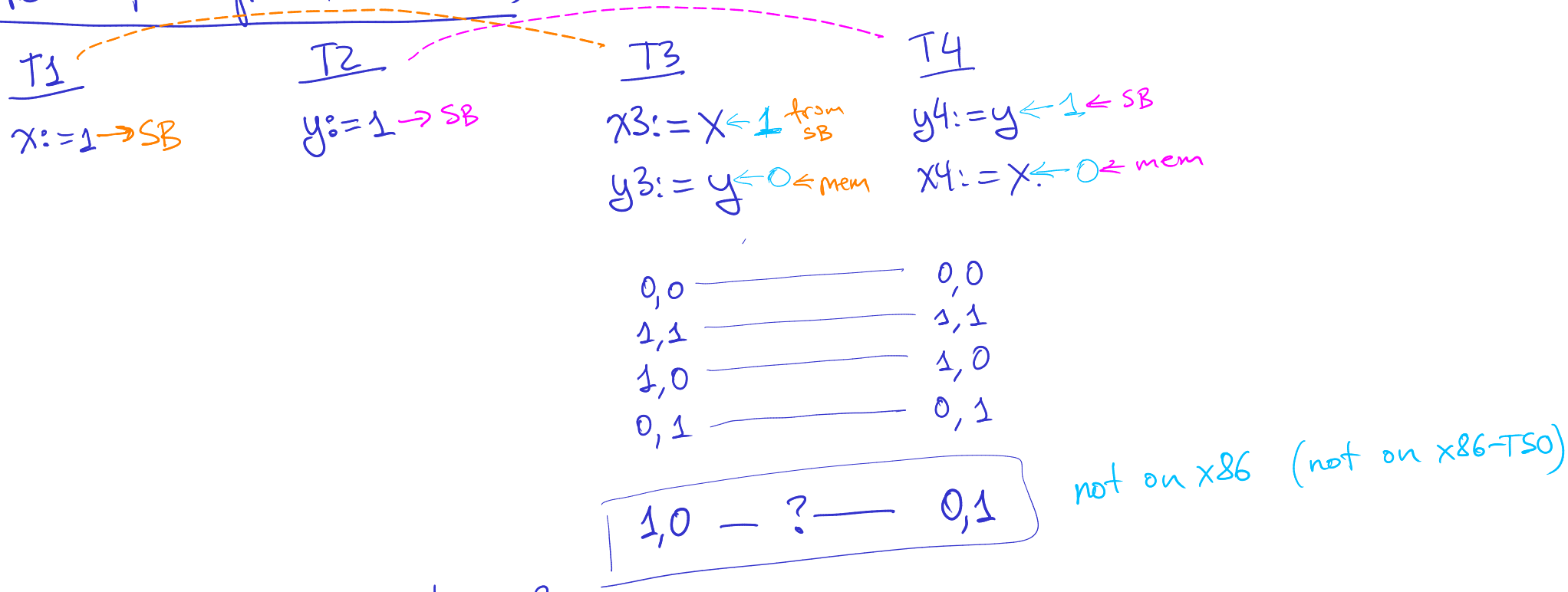
- store(x, 1)
- t := fetch(y)
  store(y', t)

Speculative exec.

T1	T2
*ptr = x	p = shared
shared = ptr	print(*p)

```
if (a) {
  ... <-
} else {
  ... <-
}
```

Indep. reads of indep. writes (IRIW)



not on x86 (not on x86-TSO).

How to program on weak mem?

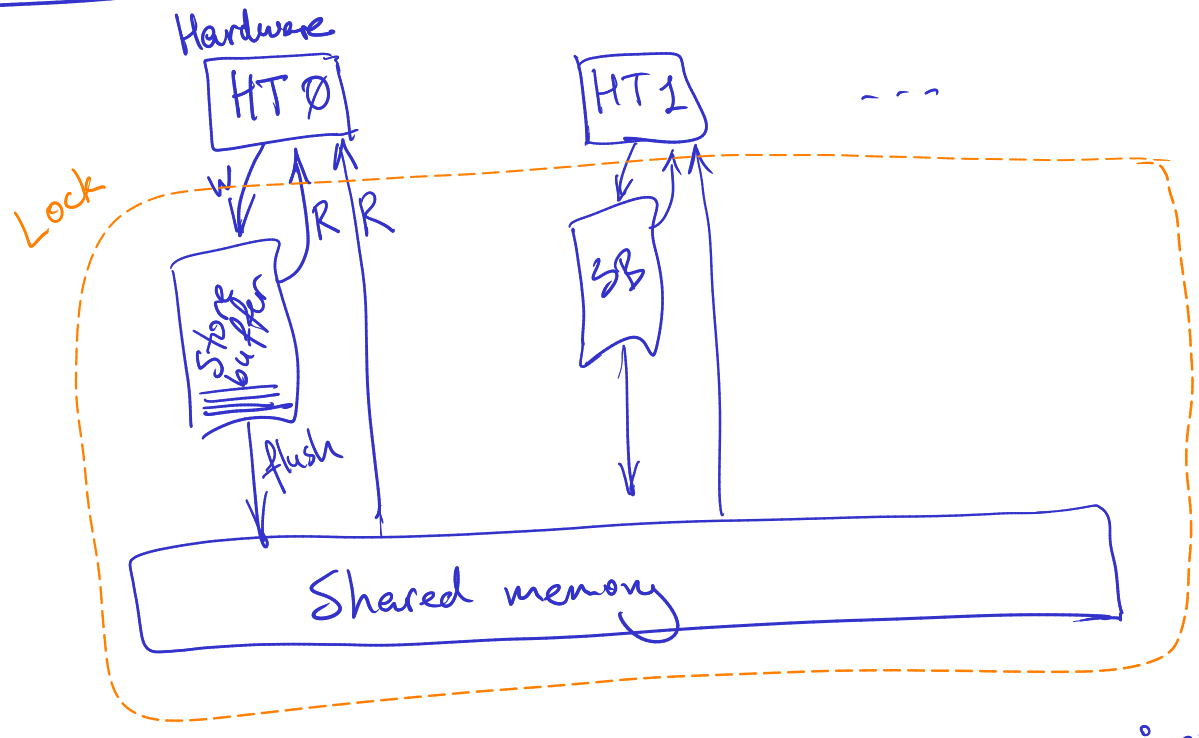
Simulate SC: locks, etc.

General: lock impl?

lock-free: Linux RCU

X86-TSO

x86-TSO model : Abstract machine.



Atomic steps: R from SB or mem
* W to SB
Flush from SB: fence
↳ ≈
Locked op: +flush SB.+

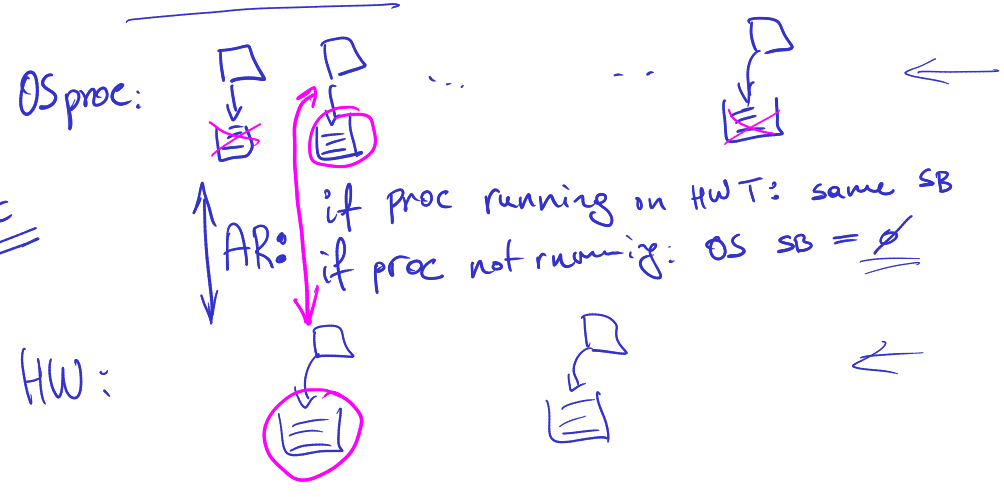
T1
x=1→SB
y'=y←mem

T2
y=1→SB
x'=x←mem

0,0: not in SC.

x=1→SB
y=x←Ø.

Breakout Q: x86-TSO for OS thread?



iret

Intel / AMD specs

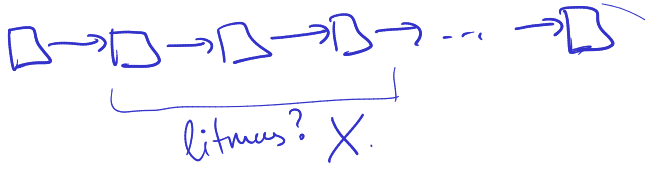
Principles: "causality"

Litmus tests.

Imprecise (prose) — Hard to understand.

Incomplete ⇐

"Axiomatic": predicates on traces.



On entry the address of spinlock is in register EAX and the spinlock is unlocked iff its value is 1			
acquire:	LOCK;DEC	[EAX]	; LOCK'd decrement of [EAX]
	JNS	enter	; branch if [EAX] was ≥ 1
spin:	CMP	[EAX],0	; test [EAX]
	JLE	spin	; branch if [EAX] was ≤ 0
	JMP	acquire	; try again
enter:	; the critical section starts here		
release:	MOV	[EAX]←1	

Locked?

→ release in mem or later?