Synchronization

- ordering of events across potentially concurrently executing code

\[ a = 0 \]
\[ a = 1 \]
\[ b = a \]

\[ a = 2 \]
\[ a = 3 \]
\[ b = a \]
Locks

- lock(a)
- unlock(a)

lock(a)
\[ a = 1 \]
\[ b = 2 \]
unlock(a)

lock(a)
\[ a = 1 \]
unlock(a)
\[ b = 3 \]

critical section

bool locked
3

fn lock(self #:) &
while self.locked &
| wait |
| 3 |
| self.locked = true; |
| |
| fn unlock(self #:) |
| & self.locked = false |
| 3 |
MESI

I = invalid
S = shared
E = exclusive
M = modified

P1 read 0x00 (CLO)
P2 reads 0x10 (CLO)
P1 write to 0x00
Upgrade CLO

RAM

P2 cache
P1 cache

Coherence (cache)

Weak consistency

TSO
A = B -> C
work throughput