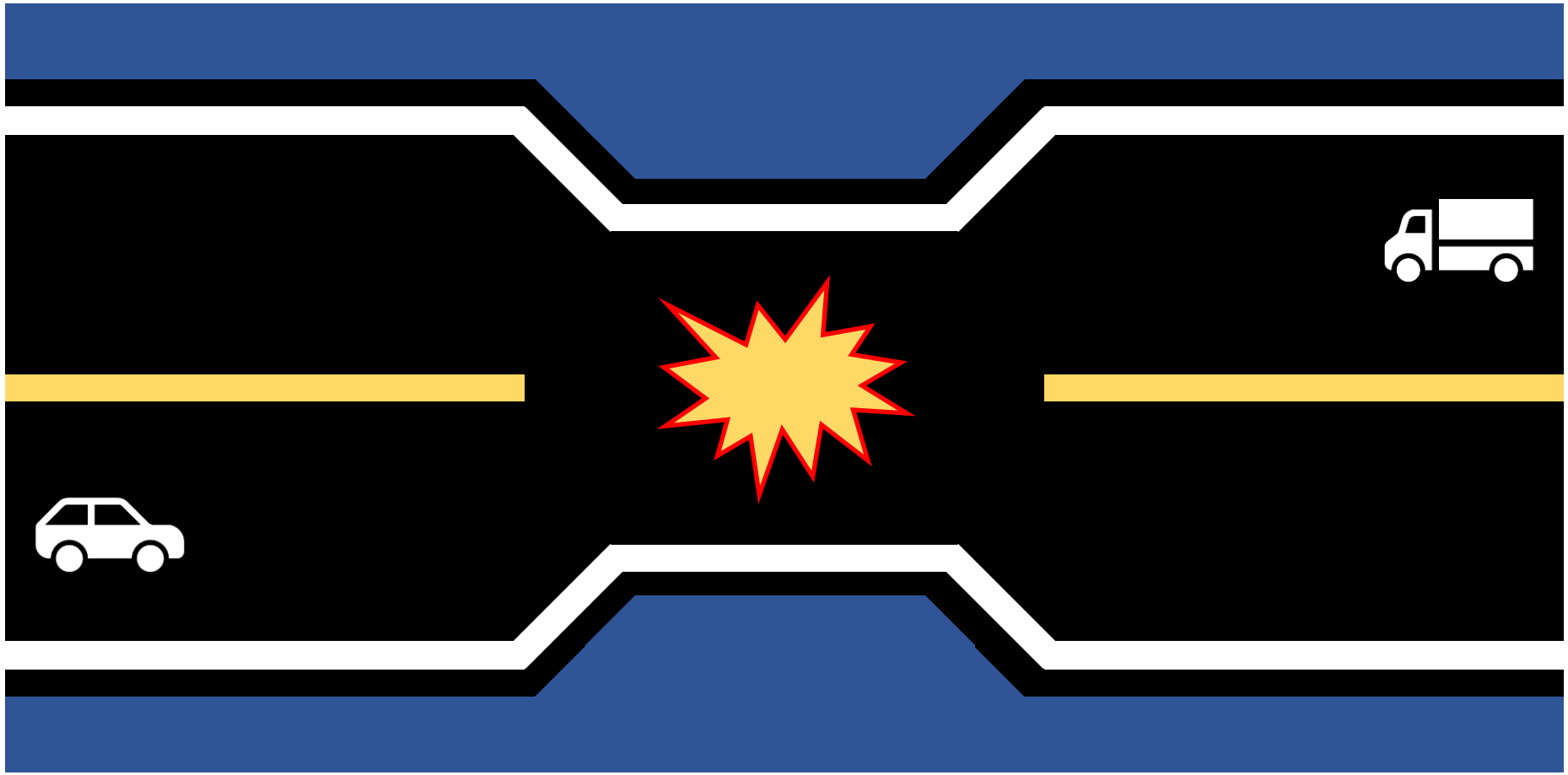
Topic V1C

Synchronization

Reading: (Section 2.11)

Synchronization



Synchronization

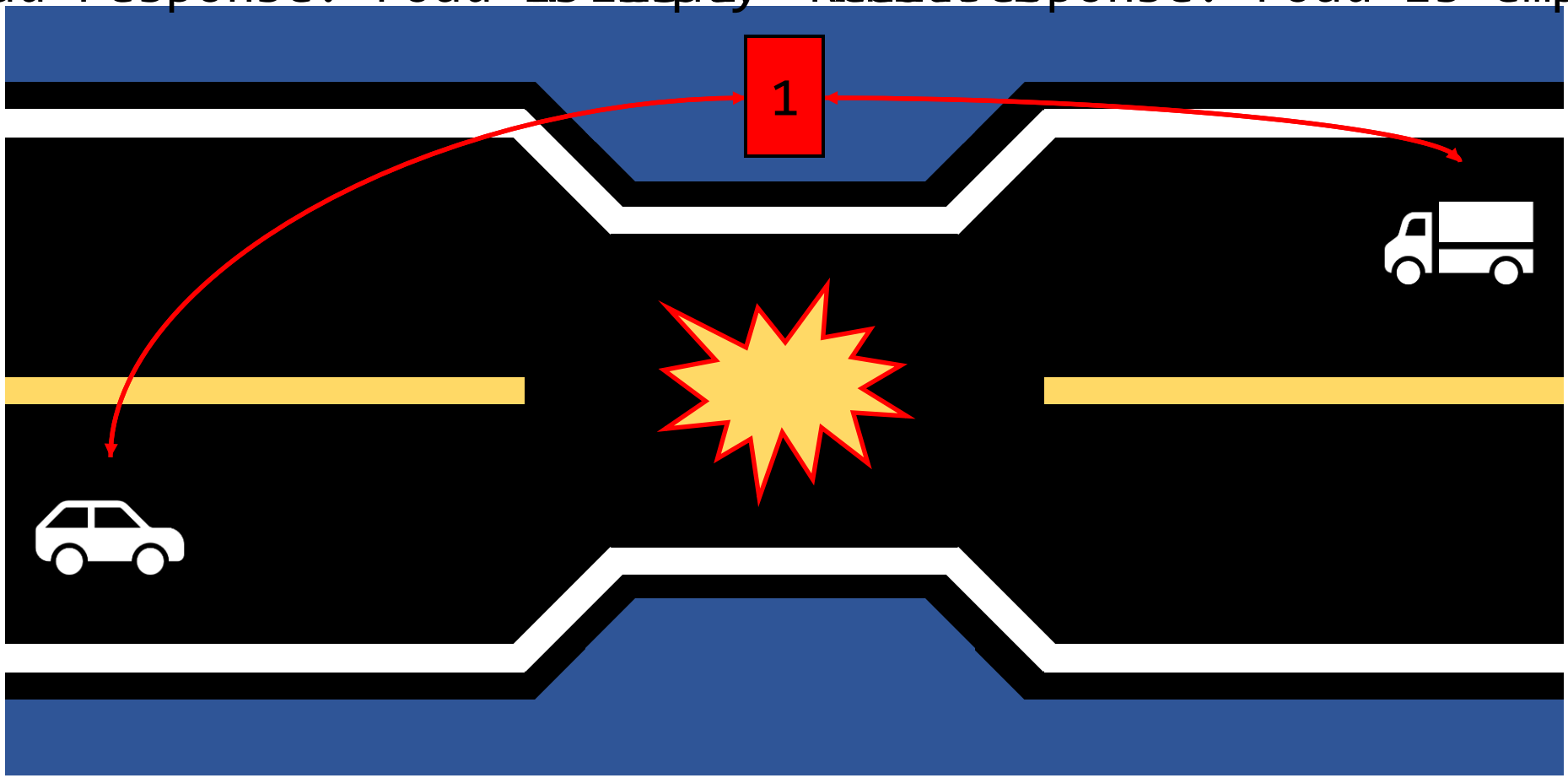
Read response: road is empty

Write 1

Ready

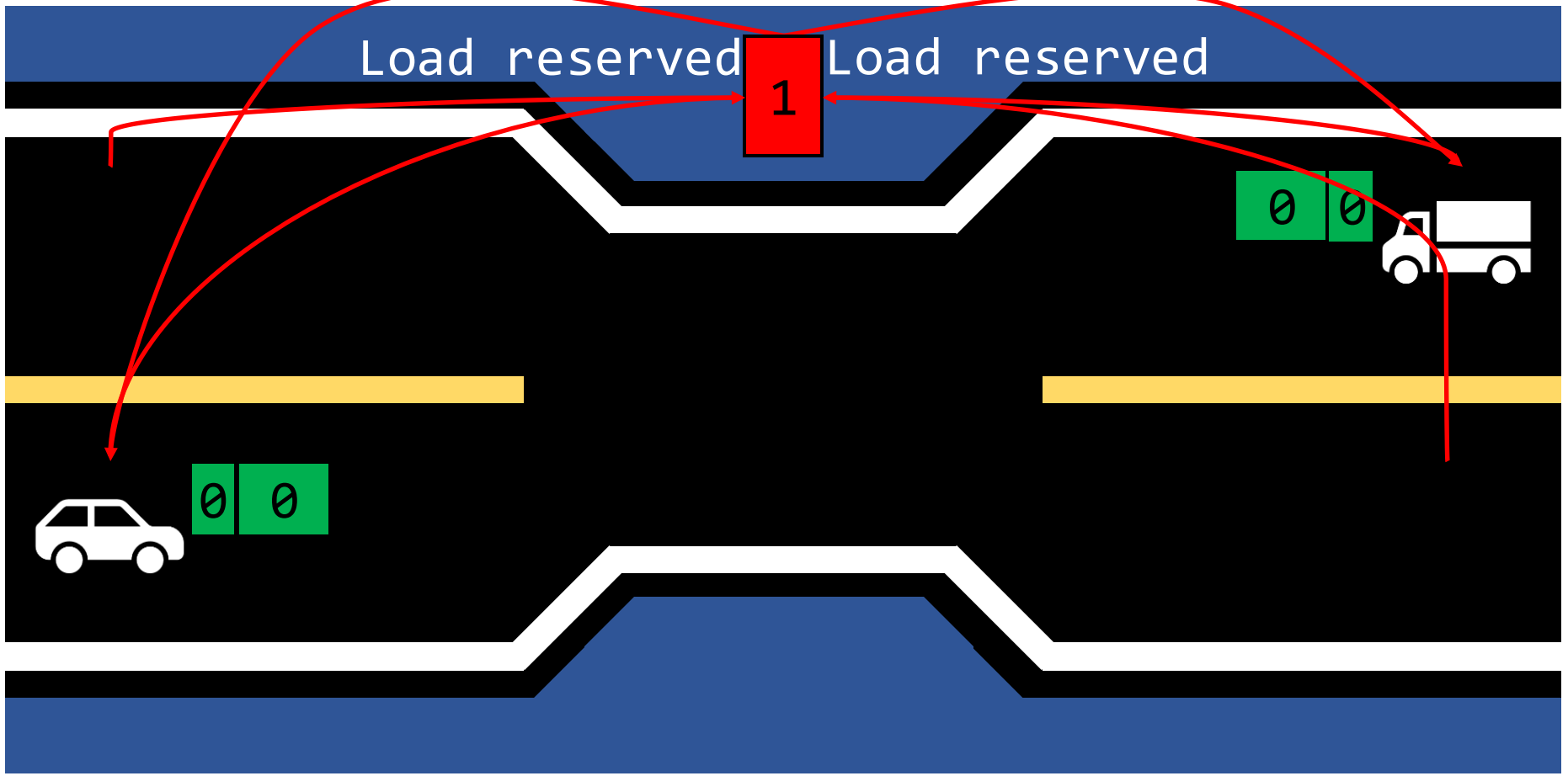
Read 1

Read response: road is empty



Better Synchronization

Read response: 1 is empty

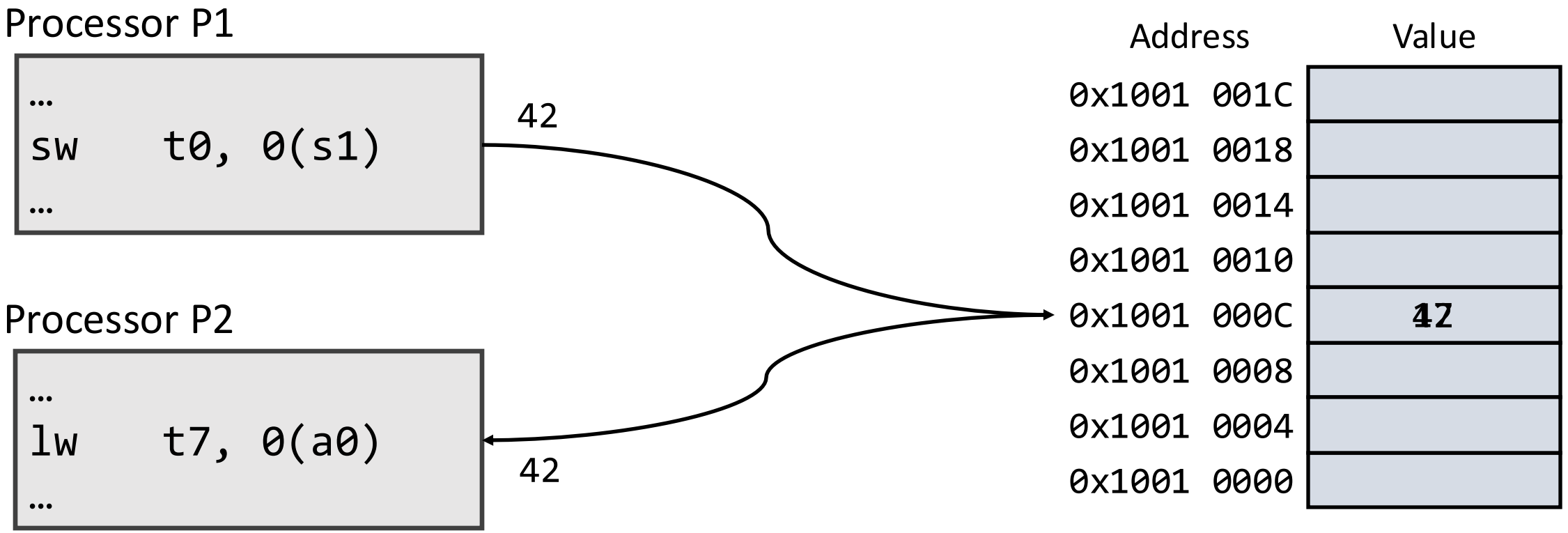


Synchronization

Two processors sharing an area of memory

P1 writes then P2 reads

If P1 and P2 don't synchronize, what will P2 read?



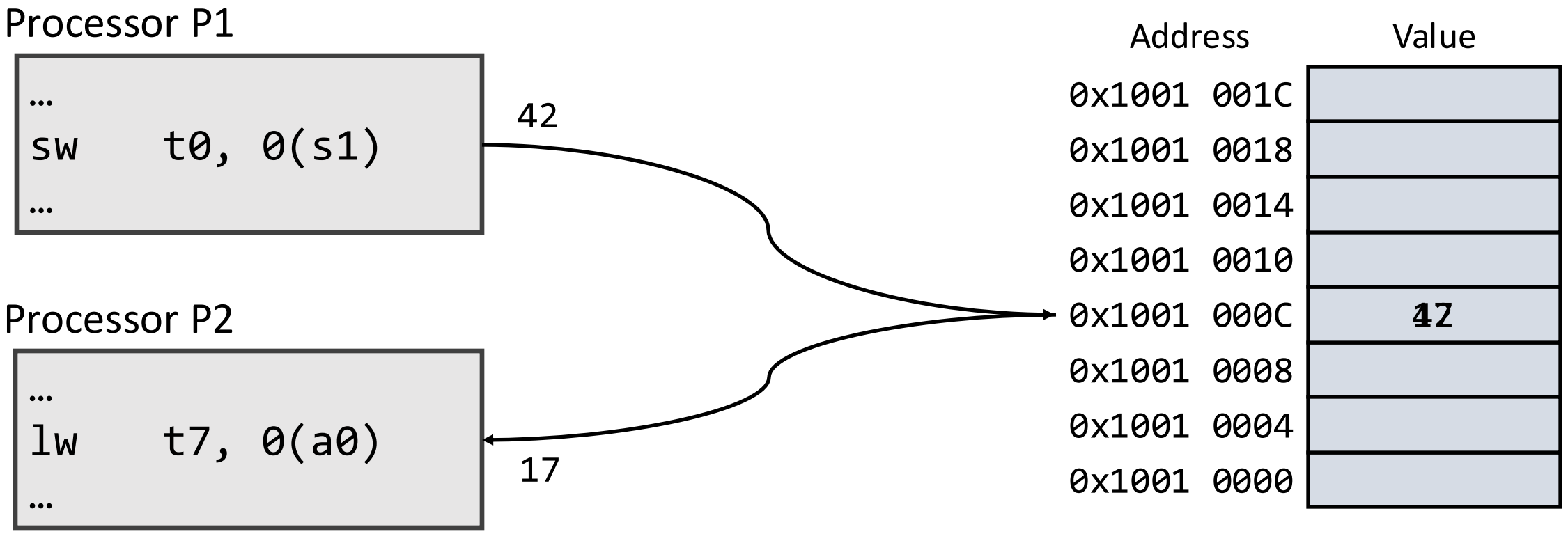
Synchronization

Two processors sharing an area of memory

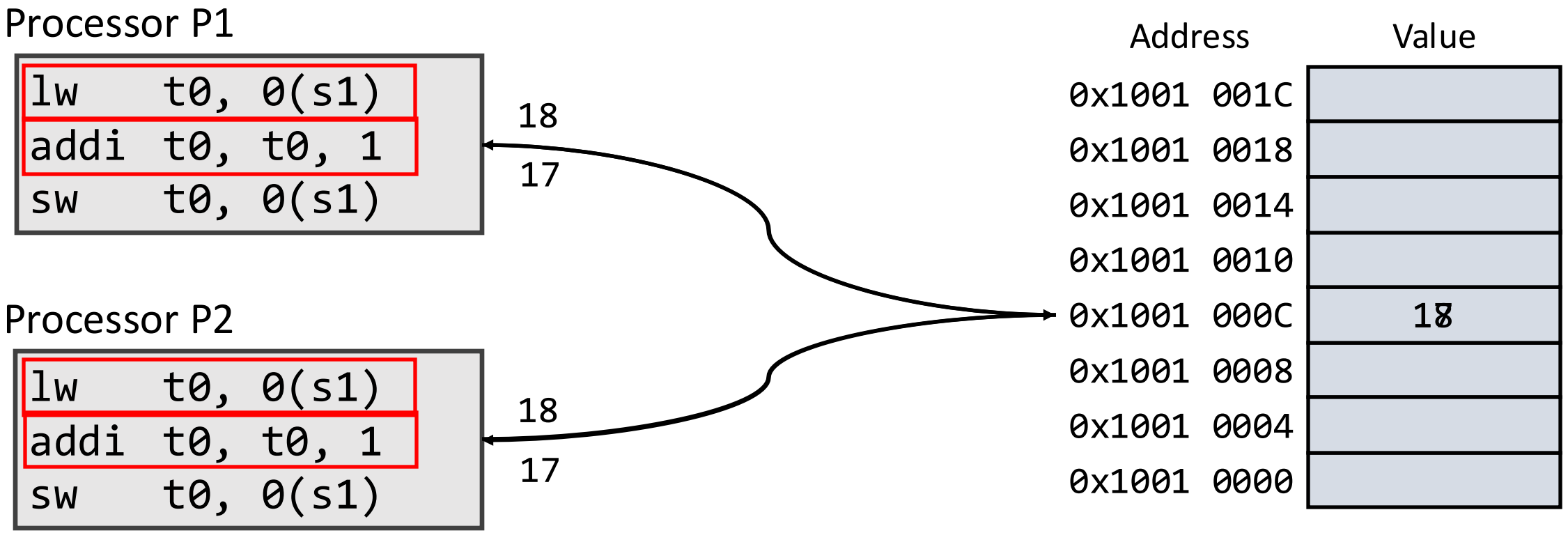
P1 writes then P2 reads

If P1 and P2 don't synchronize, what will P2 read?

Result depends on order of accesses ↔ *Data Race*



Counter-Update (example)



Synchronization

Hardware support required

- Atomic read/write memory operation
- No other access to the location allowed between read and write

Could be a single instruction

- E.g., atomic swap of register ↔ memory

Or an atomic pair of instructions

Synchronization in RISC-V

Load reserved: `lr.w rd, (rs1)`

Store conditional: `sc.w rd, rs2, (rs1)`

Succeeds if location has not changed since the `lr.w`

Returns zero in `rd`

Fails if location has changed

Returns non-zero

Example: atomic swap (to test/set lock variable)

Atomic Swap (example)

Used to test/set a variable

shared

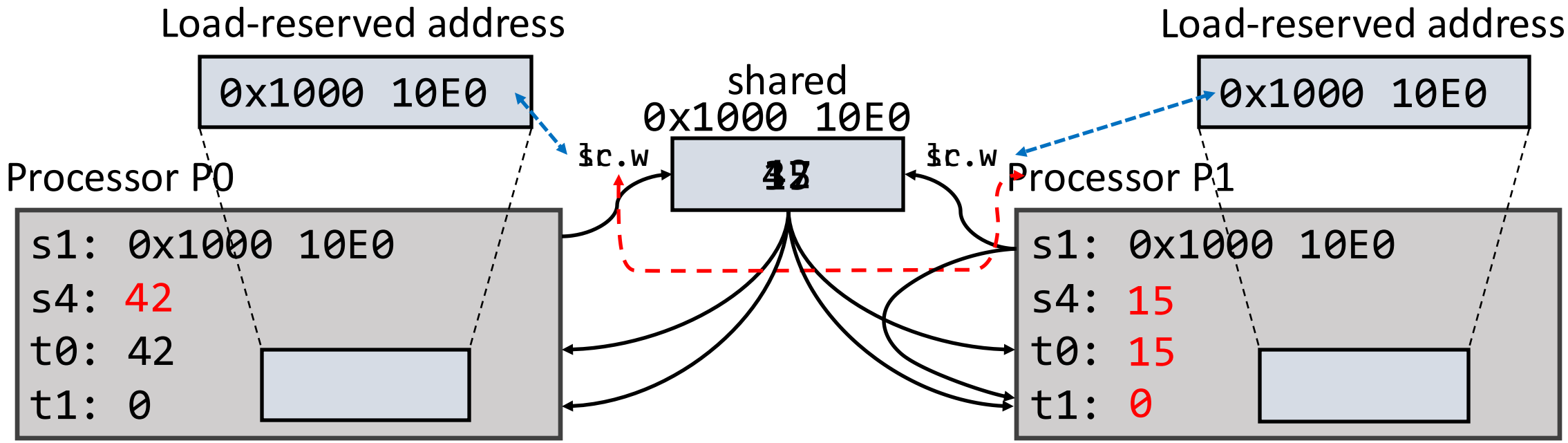
Processor P0

```
s1: address of shared

s4: local value to be
    swapped
```

Processor P1

```
s1: address of shared

s4: local value to be
    swapped
```

Atomic Swap (example)




```
try:  lr.w  t0, (s1)          # load reserved
      sc.w  t1, s4, (s1)      # store conditional
      bne   t1, zero, try     # branch if store fails
      add   s4, zero, t0      # put loaded value in s4
```