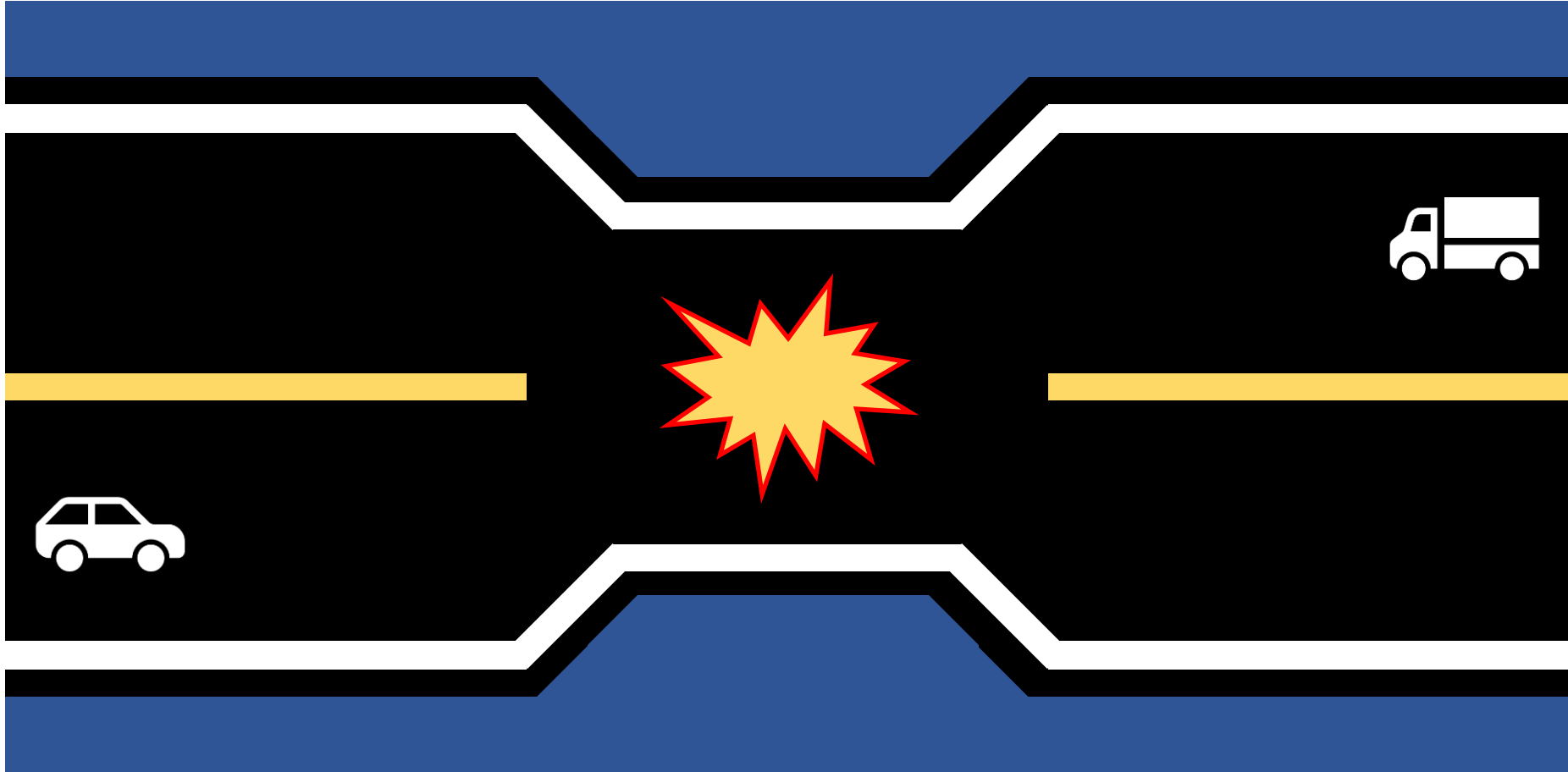


# Topic V1C

Synchronization

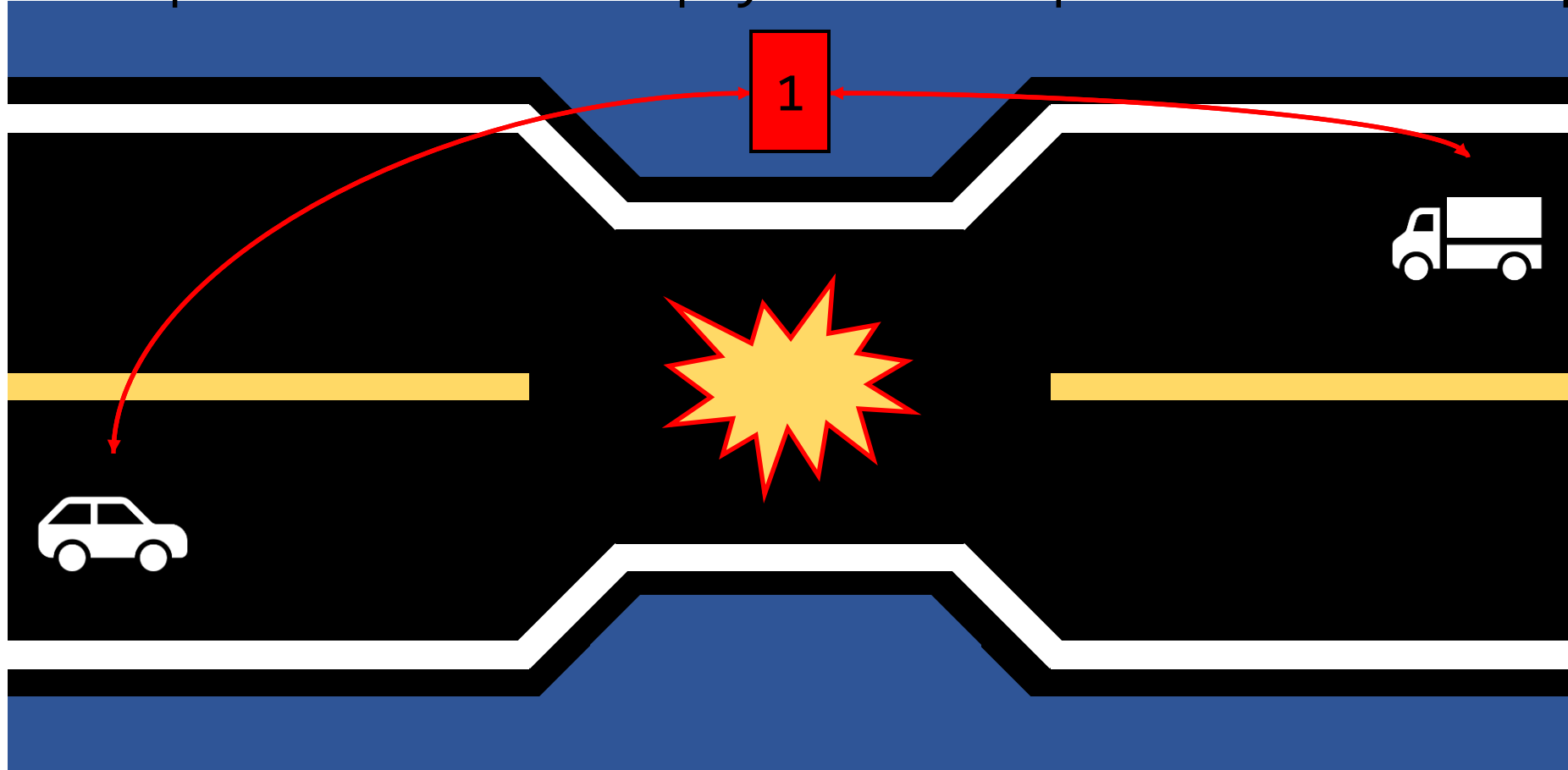
Reading: (Section 2.11)

# Synchronization



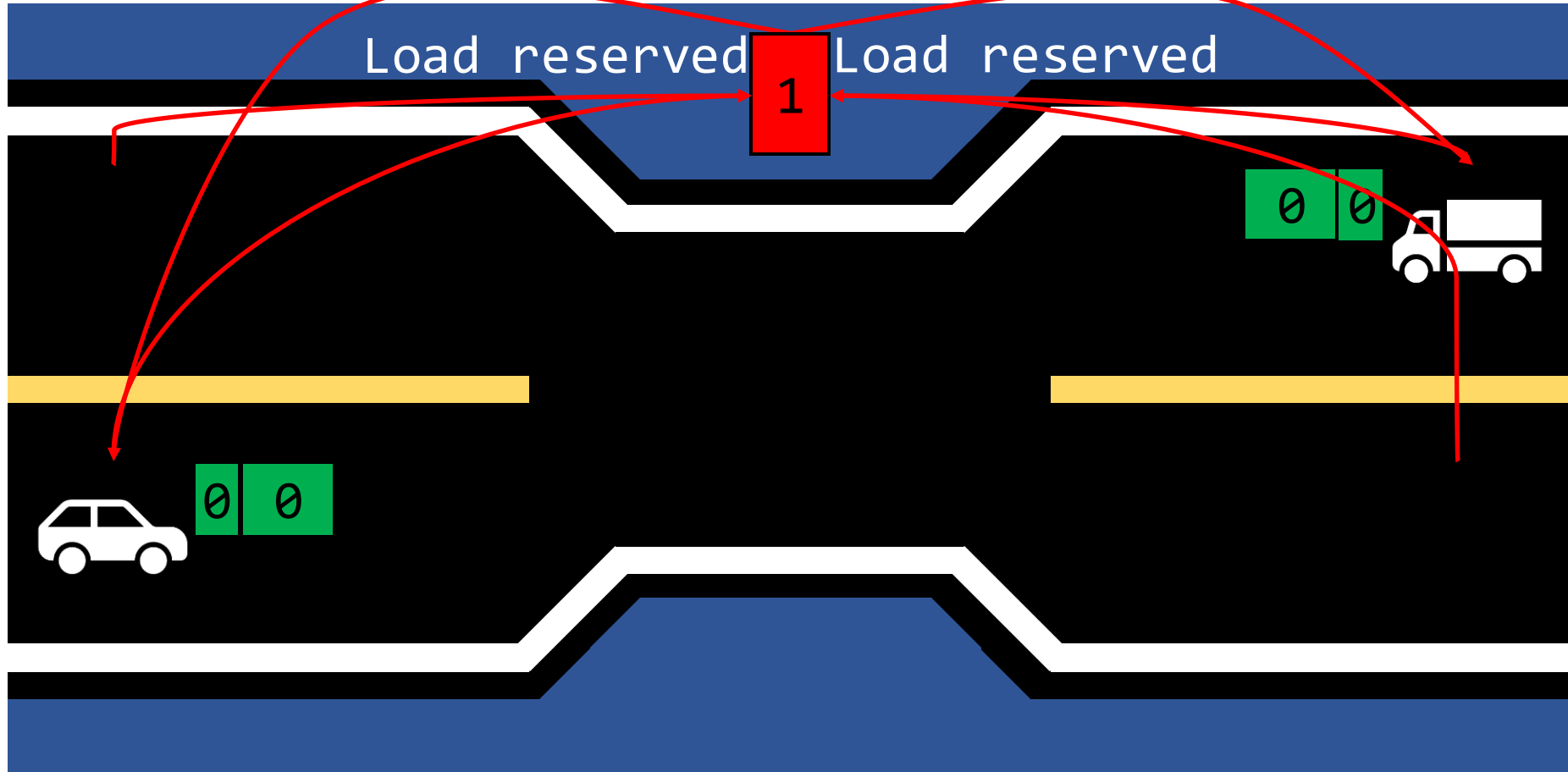
# Synchronization

Read response: road is busy      Write response: road is empty



# Better Synchronization

Read response condition is empty  
Write condition is empty



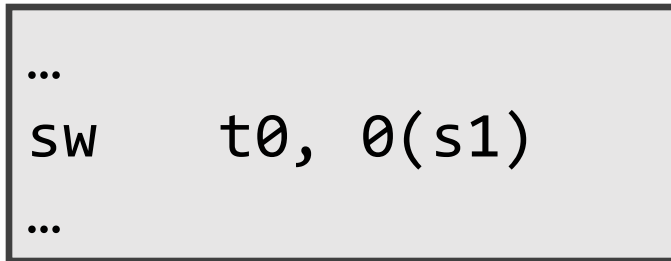
# Synchronization

Two processors sharing an area of memory

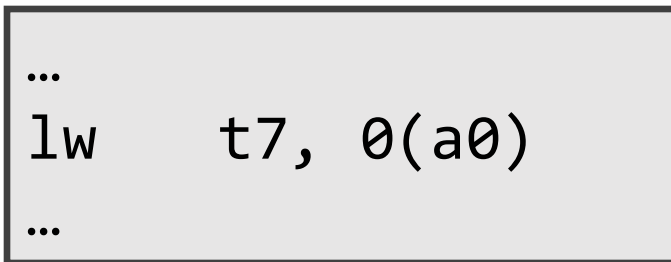
P1 writes then P2 reads

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

Processor P1



Processor P2



Address		Value
0x1001	001C	
0x1001	0018	
0x1001	0014	
0x1001	0010	
0x1001	000C	<b>42</b>
0x1001	0008	
0x1001	0004	
0x1001	0000	

# Synchronization

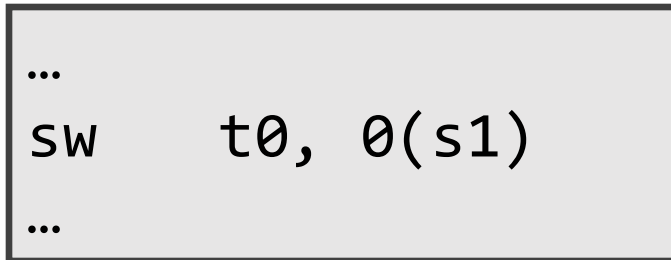
Two processors sharing an area of memory

P1 writes then P2 reads

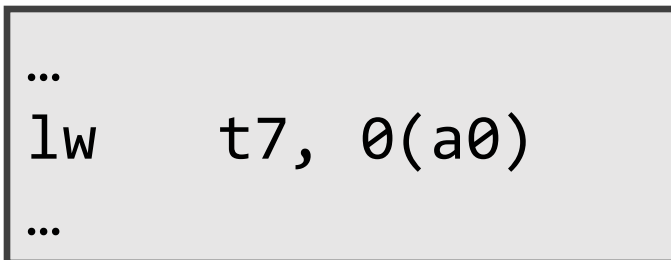
If P1 and P2 don't synchronize, what will P2 read?

Result depends on order of accesses  $\leftrightarrow$  *Data Race*

Processor P1



Processor P2



Address		Value
0x1001	001C	
0x1001	0018	
0x1001	0014	
0x1001	0010	
0x1001	000C	<b>42</b>
0x1001	0008	
0x1001	0004	
0x1001	0000	

# Counter-Update (example)

Processor P1

```
lw    t0, 0(s1)
addi  t0, t0, 1
sw    t0, 0(s1)
```

Processor P2

```
lw    t0, 0(s1)
addi  t0, t0, 1
sw    t0, 0(s1)
```

Address

Value

0x1001 001C

0x1001 0018

0x1001 0014

0x1001 0010

0x1001 000C

0x1001 0008

0x1001 0004

0x1001 0000

18

18

17

18

17

# 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  $\leftrightarrow$  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



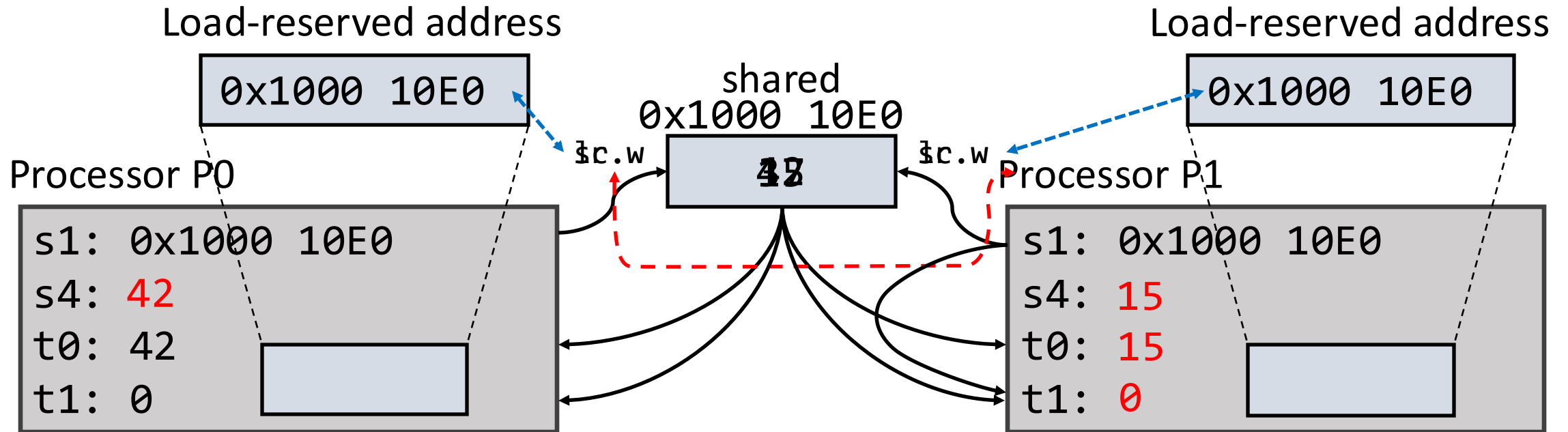
Processor P0

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
```