Sufficient Conditions for Robustness of RDMA Programs

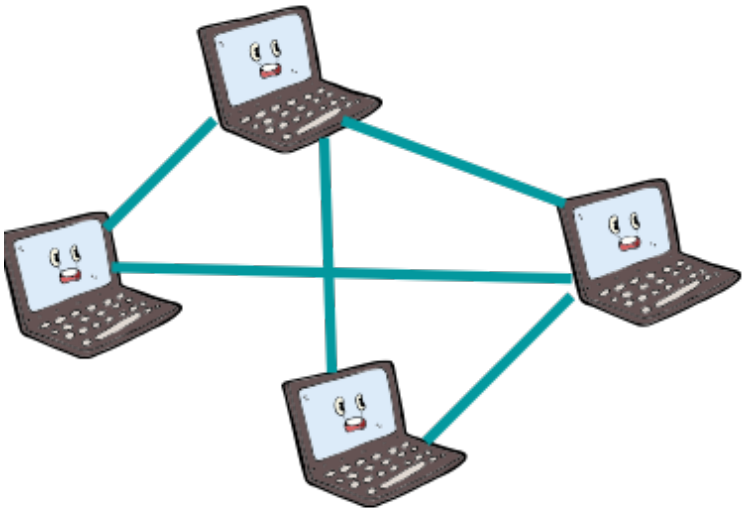
ESOP 2025

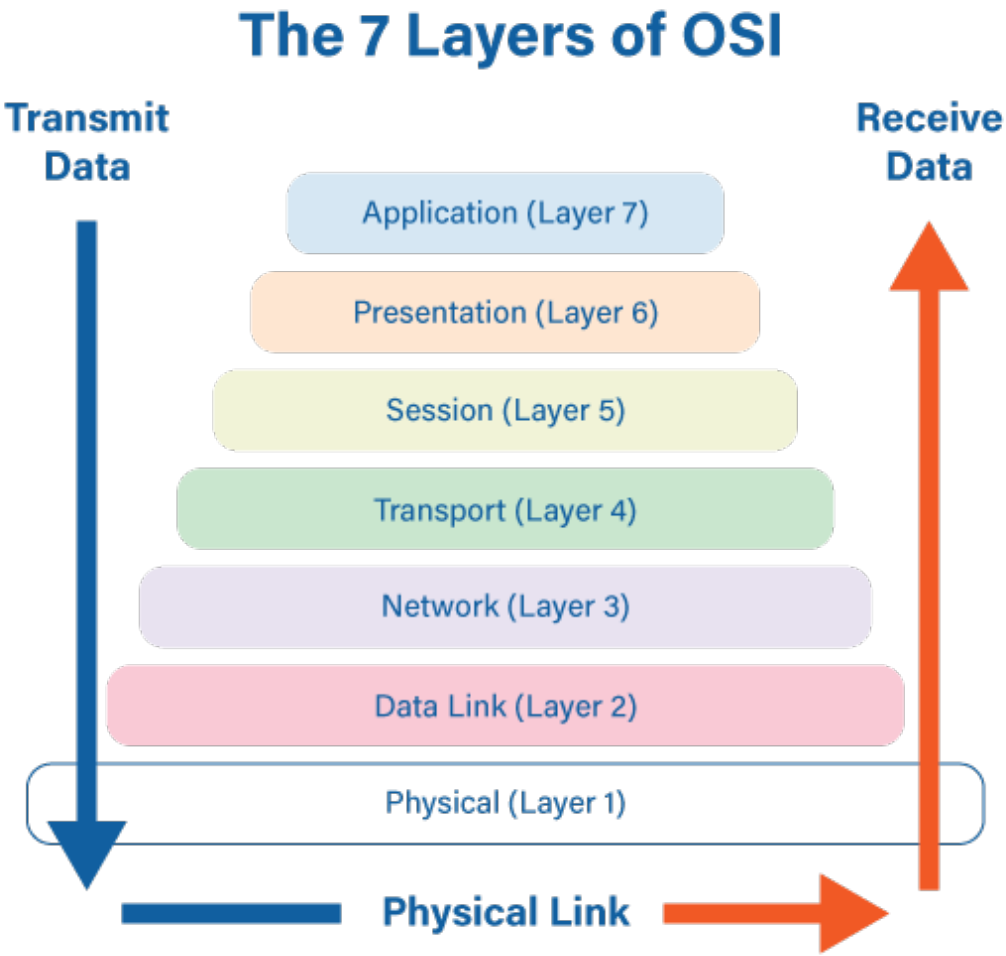
Guillaume Ambal[1], Ori Lahav[2], Azalea Raad[1]

[1]Imperial College London, [2]Tel Aviv University

May 8, 2025

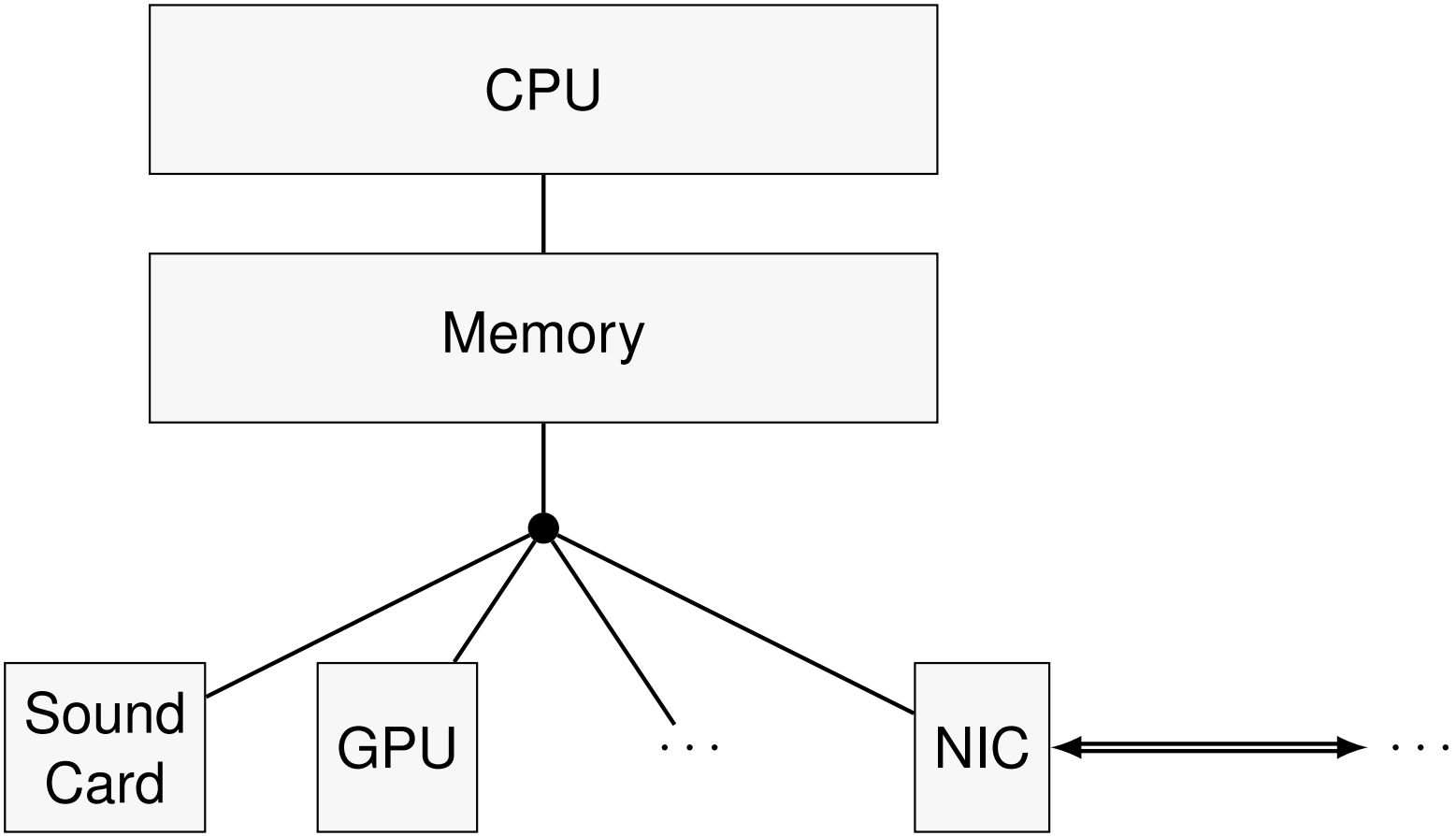
HPC/database: slow communication



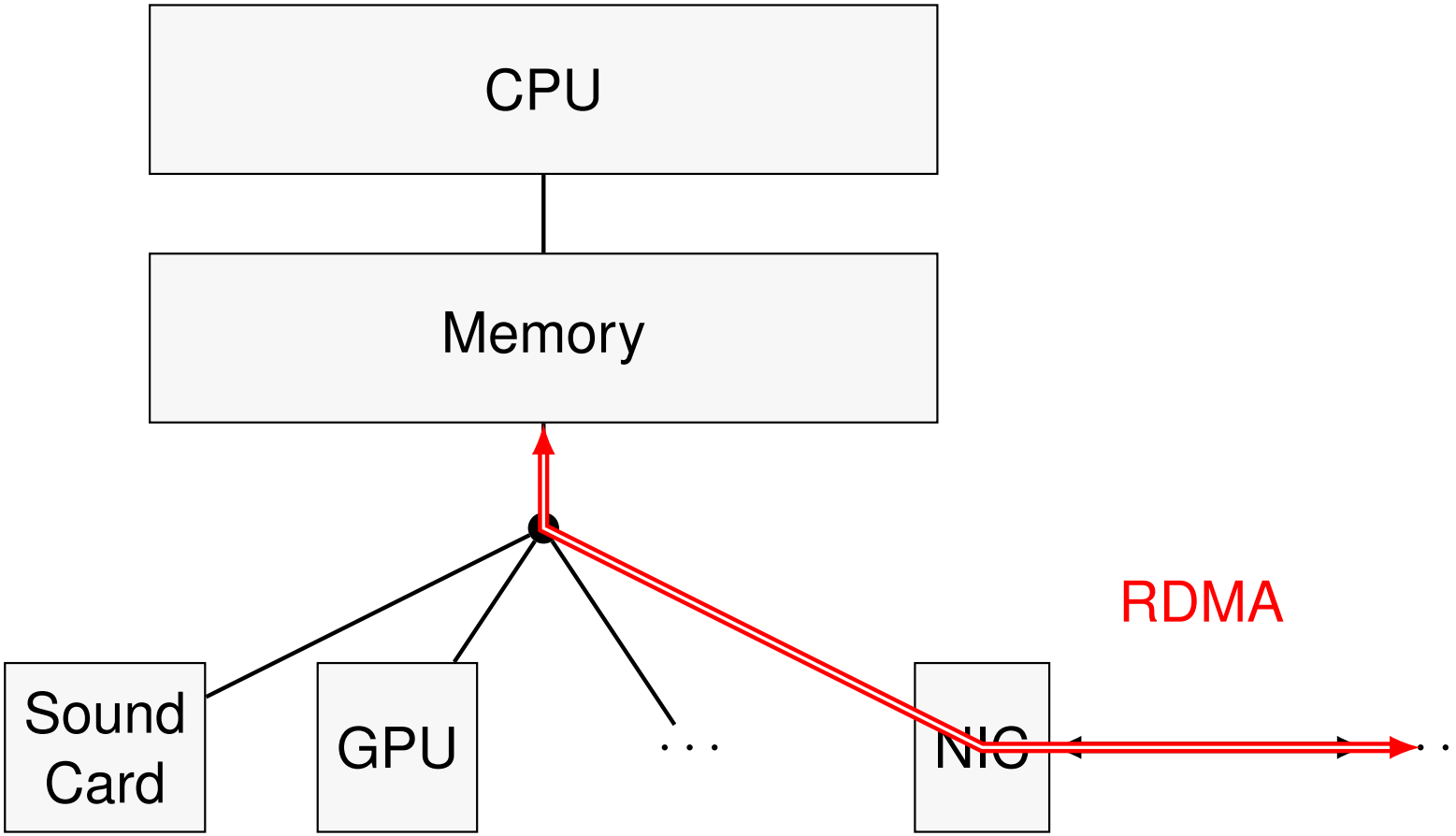


Latency : $\sim$ ms

Direct Memory Access



Direct Memory Access



Remote Direct Memory Access

RDMA

- Data-transfer protocol for High-Performance Computing
- Latency : $\sim \mu$s
- Recently becoming widespread
- Very weak and unintuitive semantics[1]

Our work: robustness as a path for verification

- Making an existing program robust
- Guidelines for writing robust programs

[1] Ambal et al., OOPSLA 2024

Remote Direct Memory Access

RDMA

- Data-transfer protocol for High-Performance Computing
- Latency : $\sim \mu$s
- Recently becoming widespread
- Very weak and unintuitive semantics[1]

Our work: robustness as a path for verification

- Making an existing program robust
- Guidelines for writing robust programs

[1] Ambal et al., OOPSLA 2024

Model and Syntax

Model:

- network with several nodes (n_1, n_2, ...)
- each node can have several threads

Syntax

$$
\begin{array}{rll}
\text{Mem Locs} & x, y, z, \ldots & \\
\text{Expressions} & e ::= \; k \in \mathbb{Z} \mid x \mid e_1 + e_2 \mid \ldots & \\
\text{Commands} & c ::= \; x := e \mid \texttt{mfence} & \leftarrow \text{usual x86-TSO} \\
 & \quad \mid x^n := y \mid x := y^n \mid \texttt{poll}(n) & \leftarrow \text{RDMA}
\end{array}
$$

RDMA example 1/3

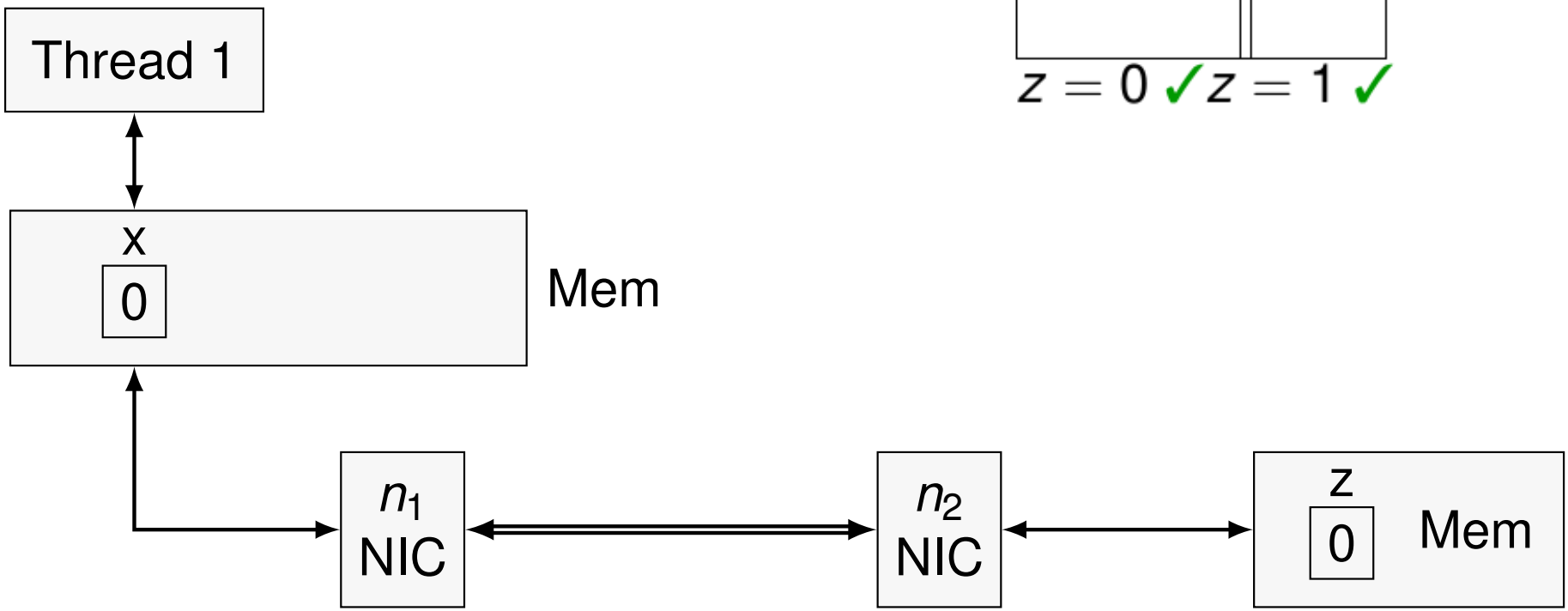
$x=0$	$z=0$
$z^{n_2} := x$ $x := 1$	

$z = 0$ ✓ $z = 1$ ✓

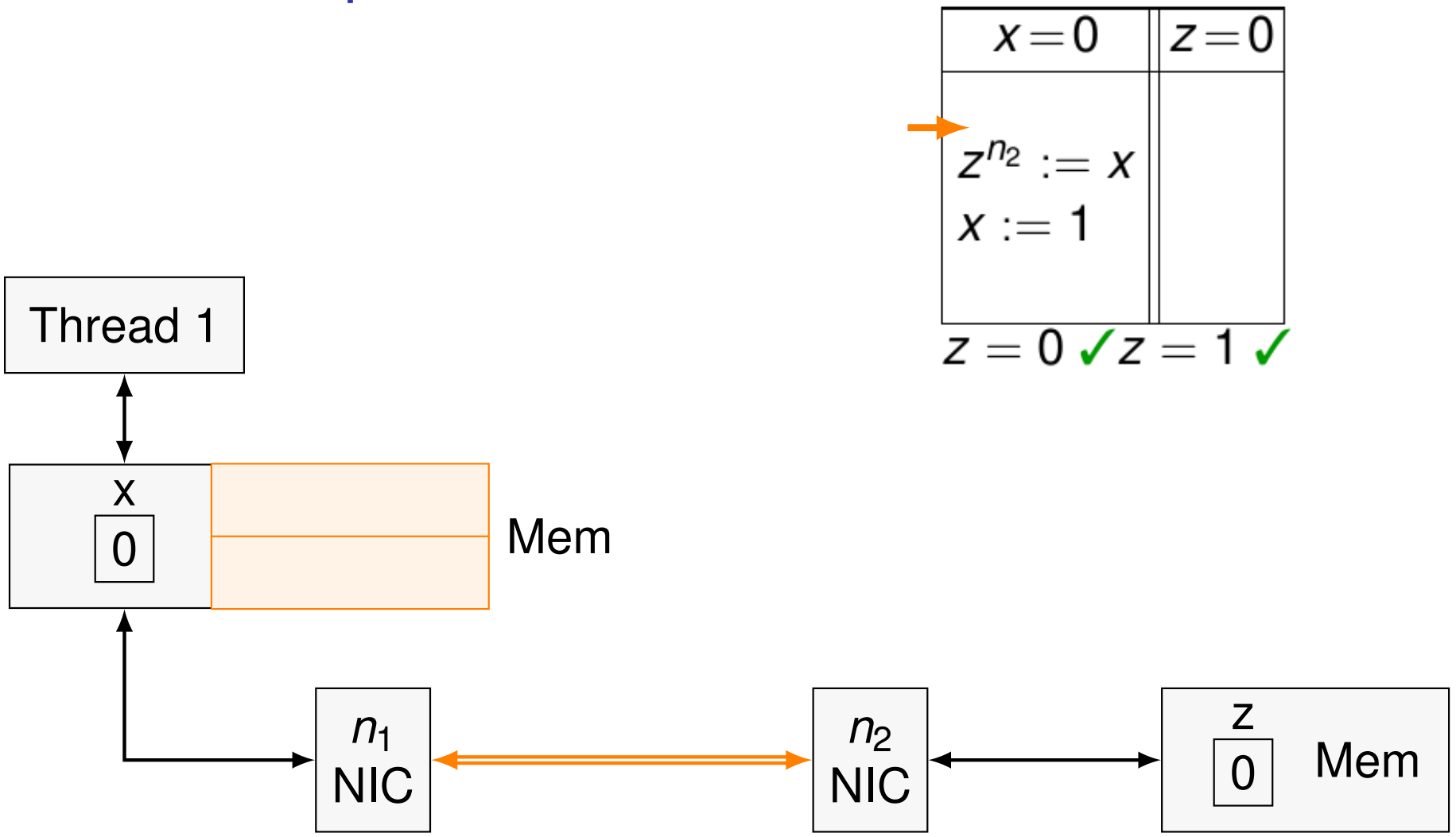
RDMA example 1/3

$x = 0$	$z = 0$
$z^{n_2} := x$ $x := 1$	

$z = 0$ ✓ $z = 1$ ✓



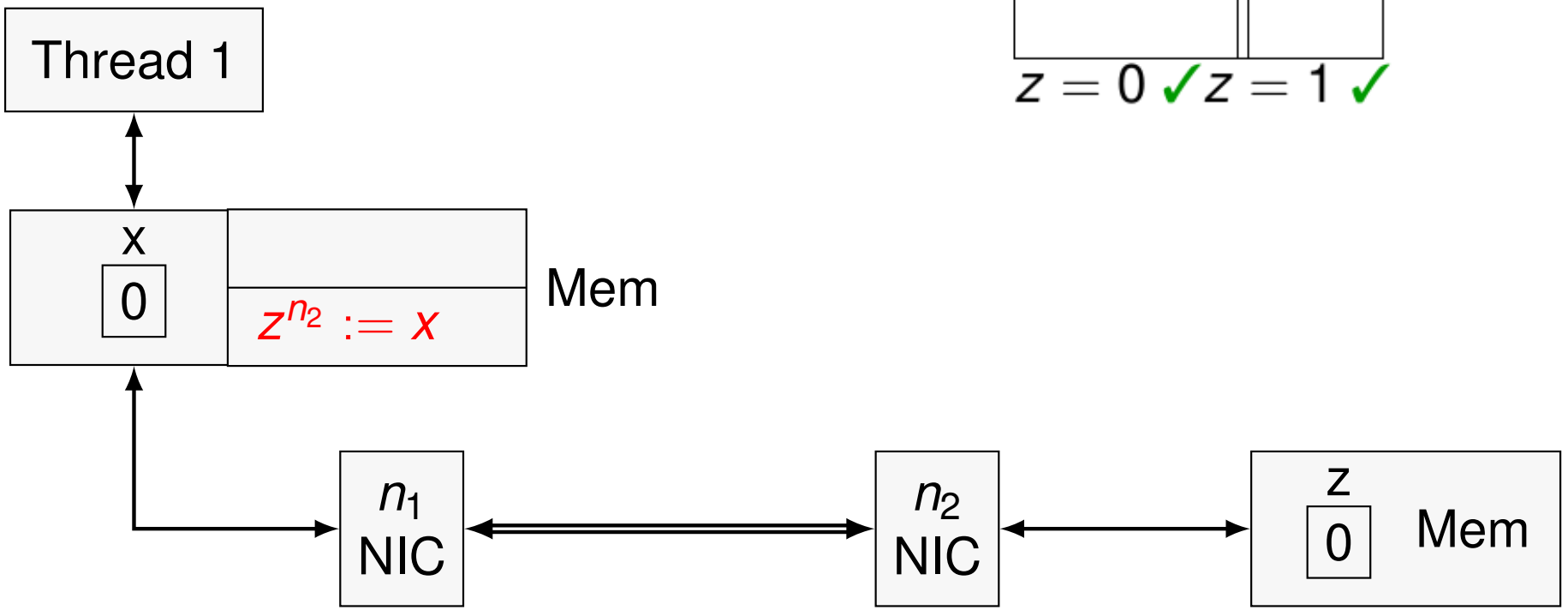
RDMA example 1/3



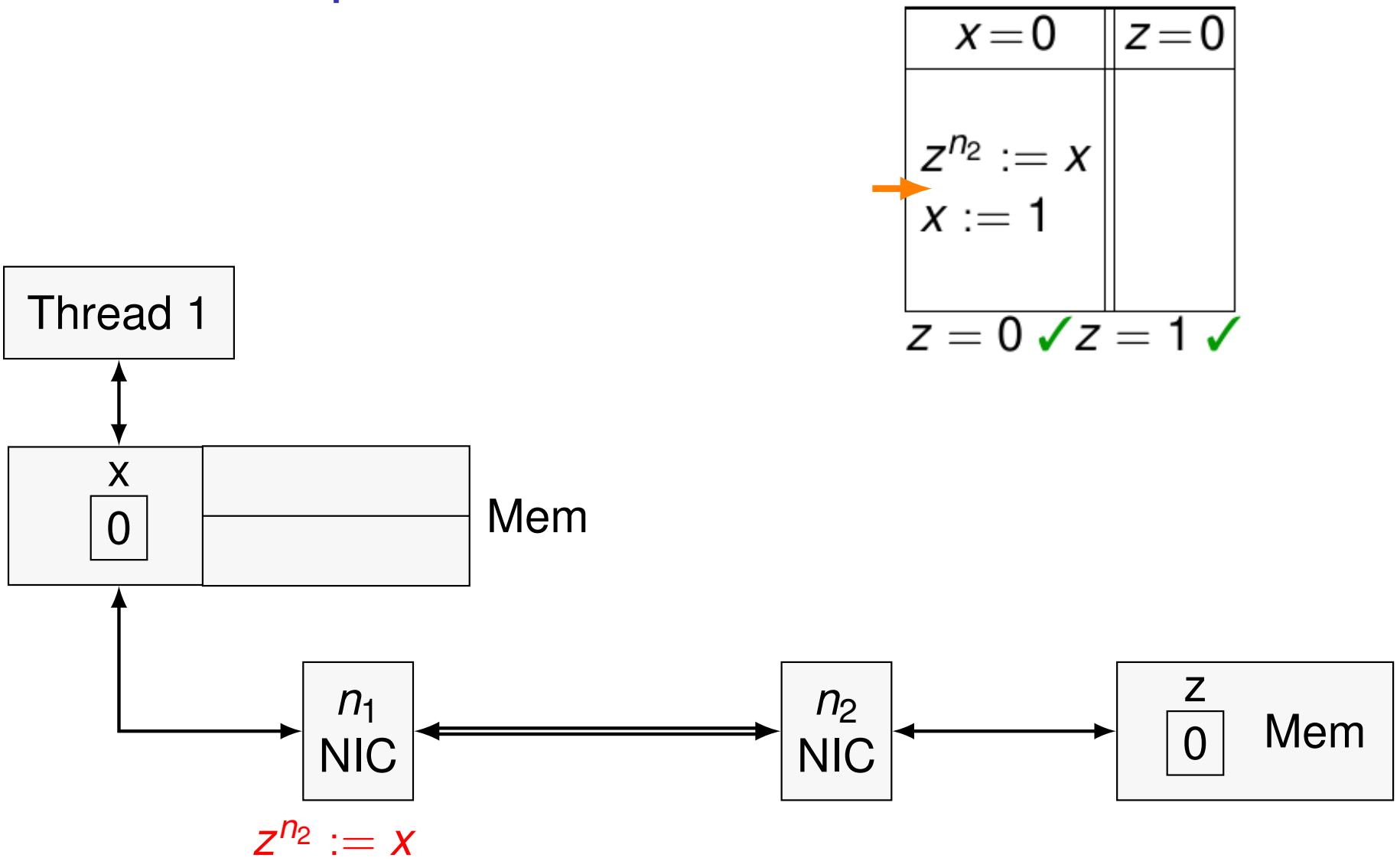
RDMA example 1/3

$x=0$	$z=0$
$z^{n_2} := x$ $x := 1$	

$z = 0$ ✓ $z = 1$ ✓



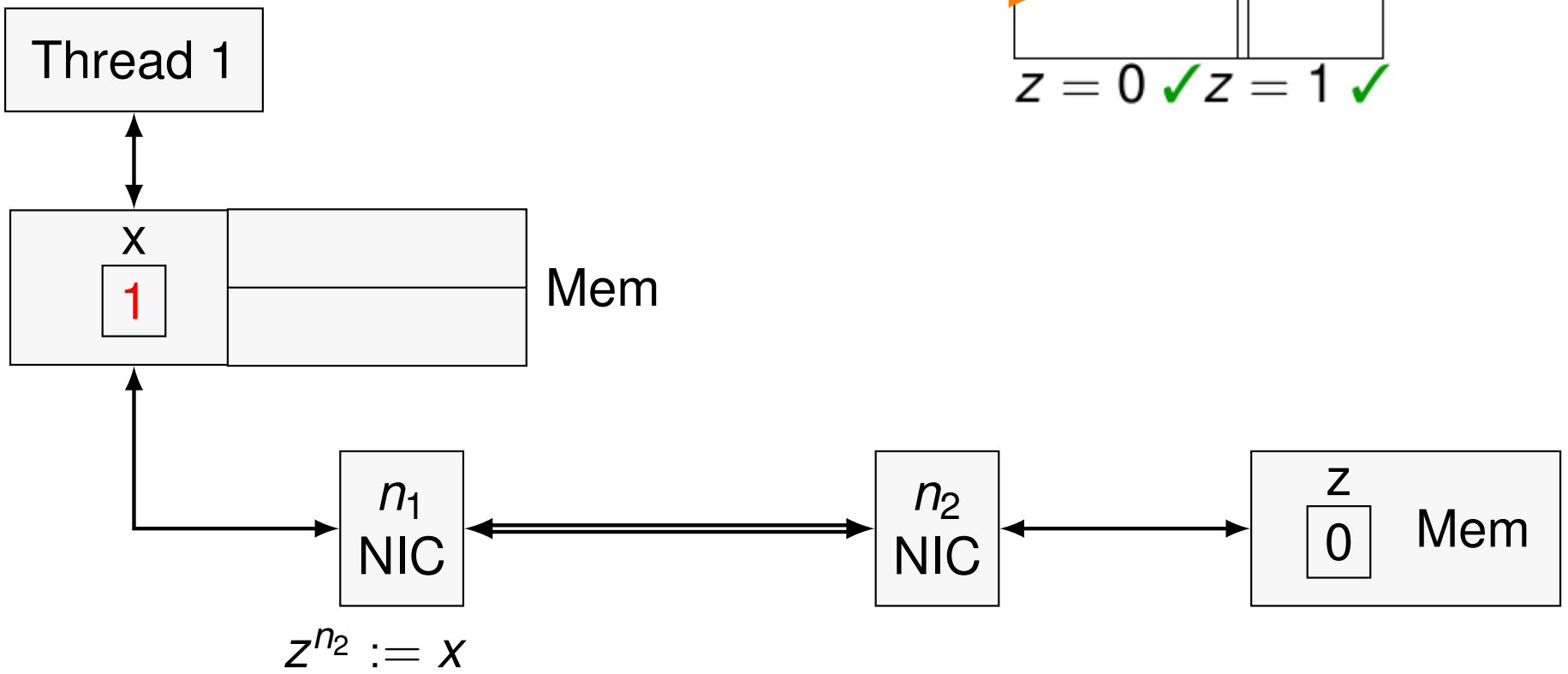
RDMA example 1/3



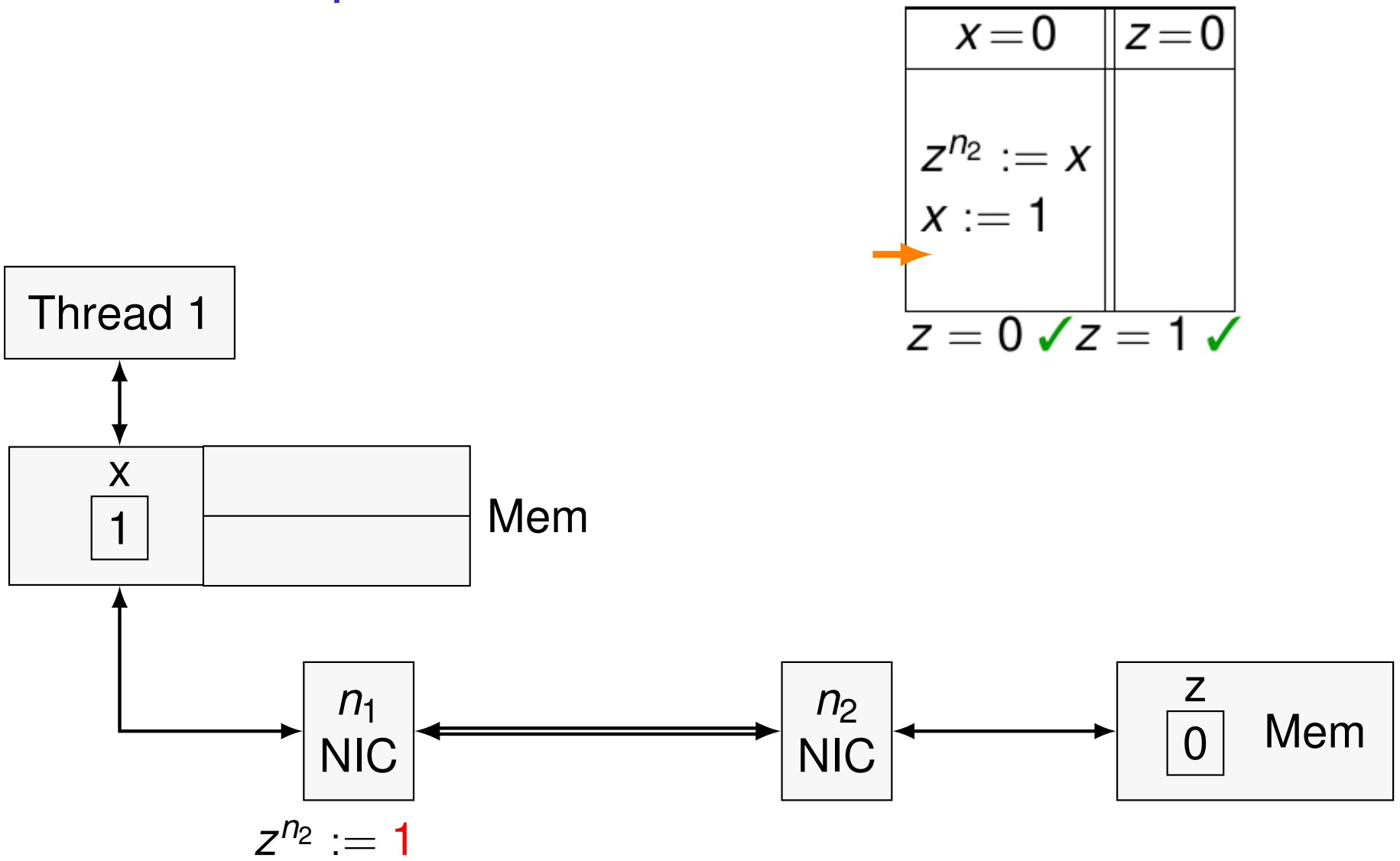
RDMA example 1/3

$x=0$	$z=0$
$z^{n_2} := x$ $x := 1$	

$z = 0$ ✓ $z = 1$ ✓



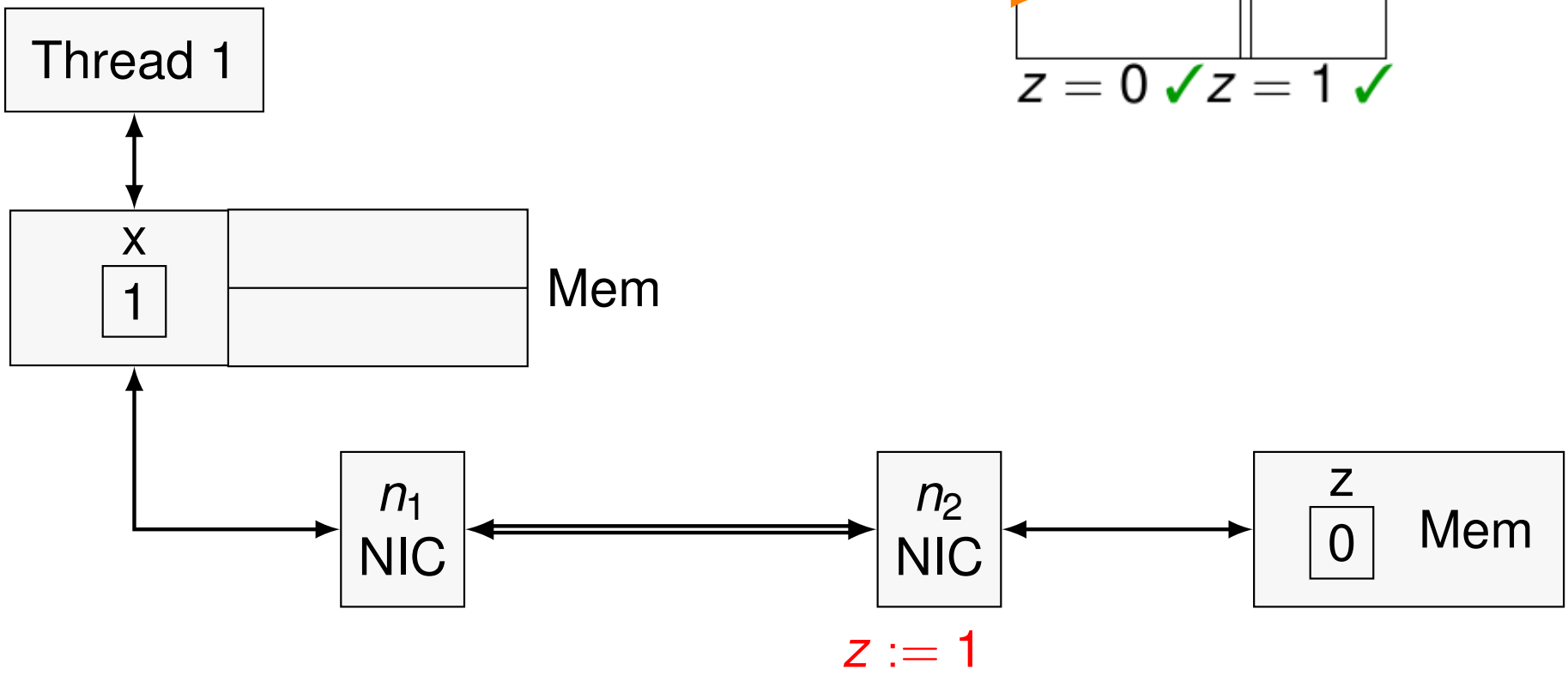
RDMA example 1/3



RDMA example 1/3

$x=0$	$z=0$
$z^{n_2} := x$ $x := 1$	

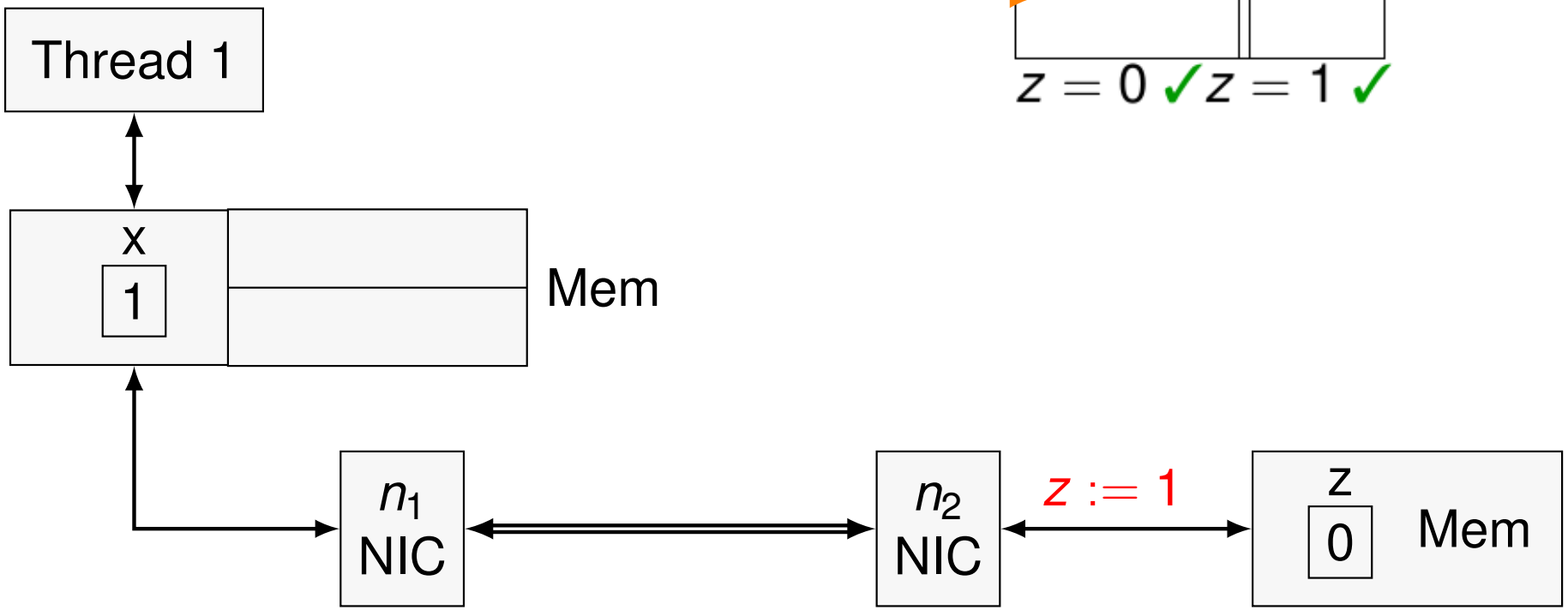
$z = 0$ ✓ $z = 1$ ✓



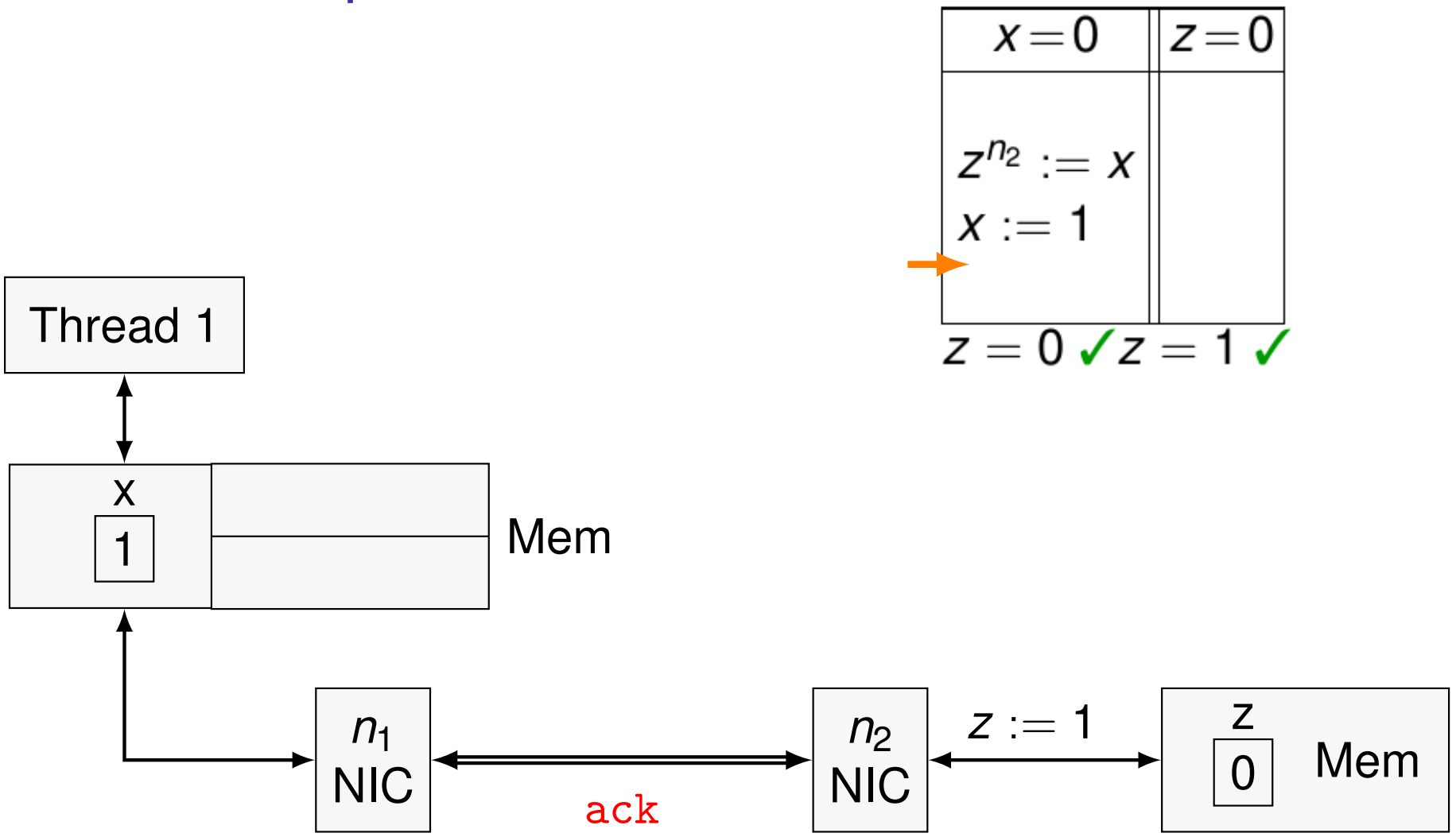
RDMA example 1/3

$x = 0$	$z = 0$
$z^{n_2} := x$ $x := 1$	

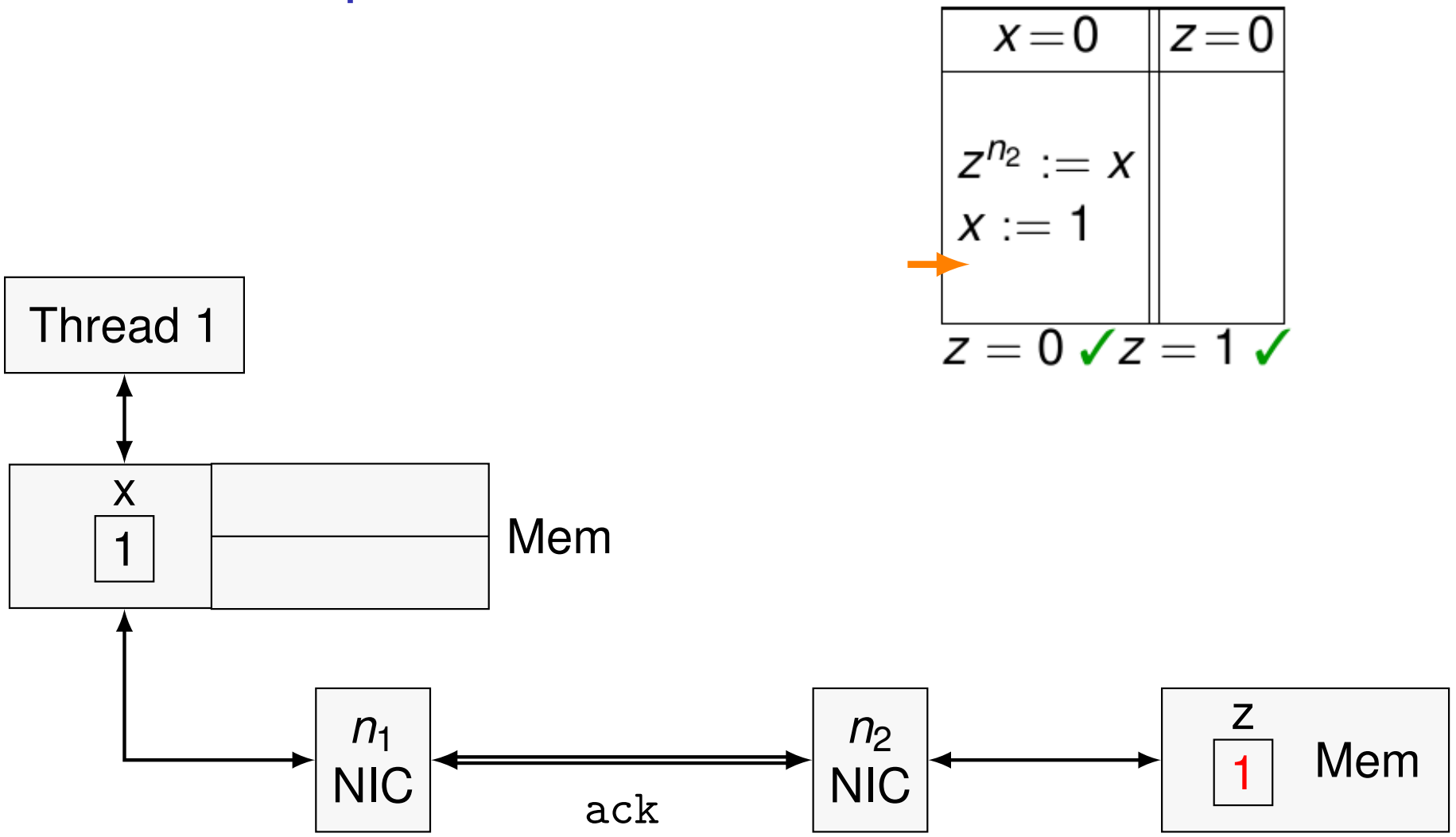
$z = 0$ ✓ $z = 1$ ✓



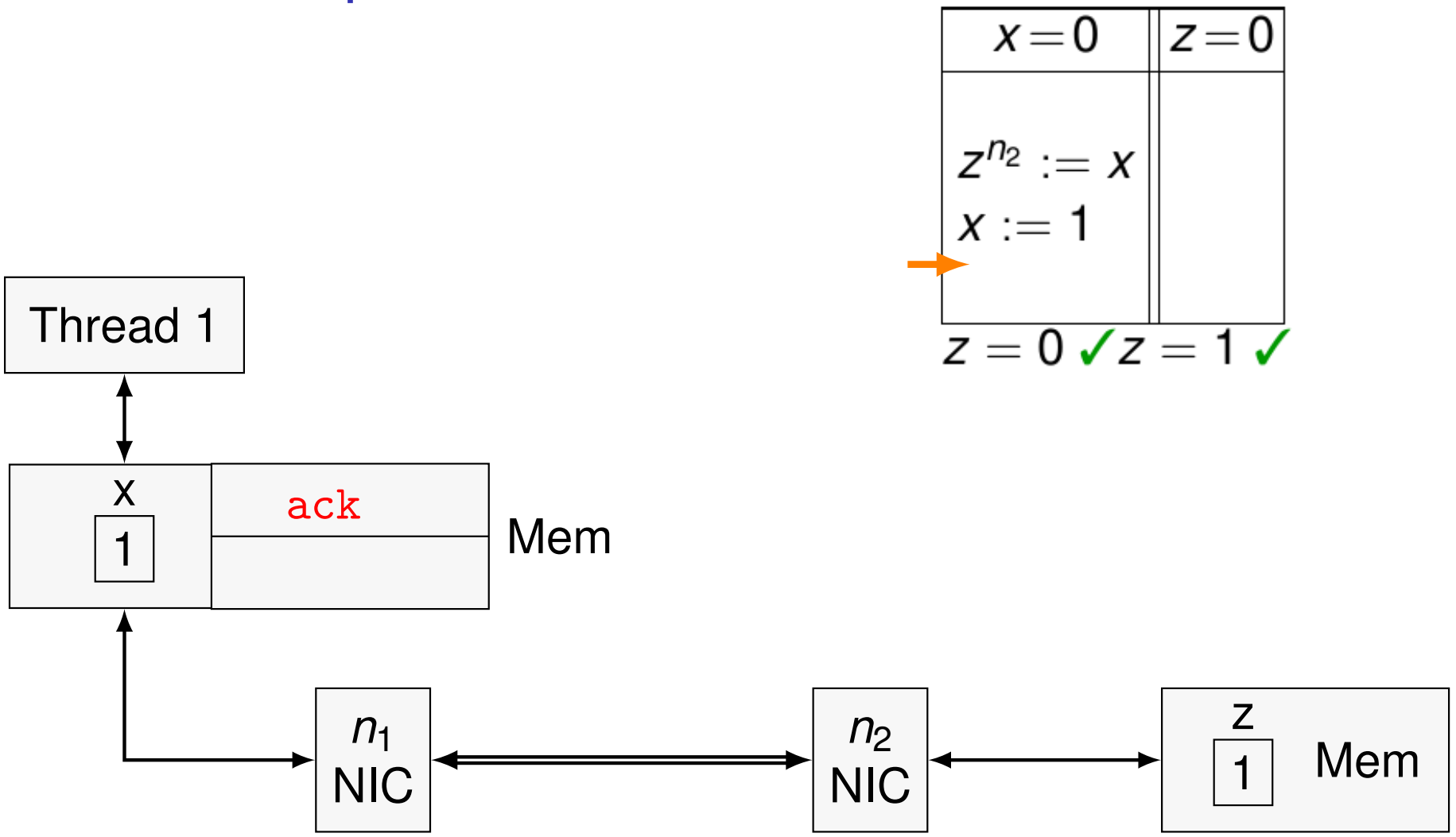
RDMA example 1/3



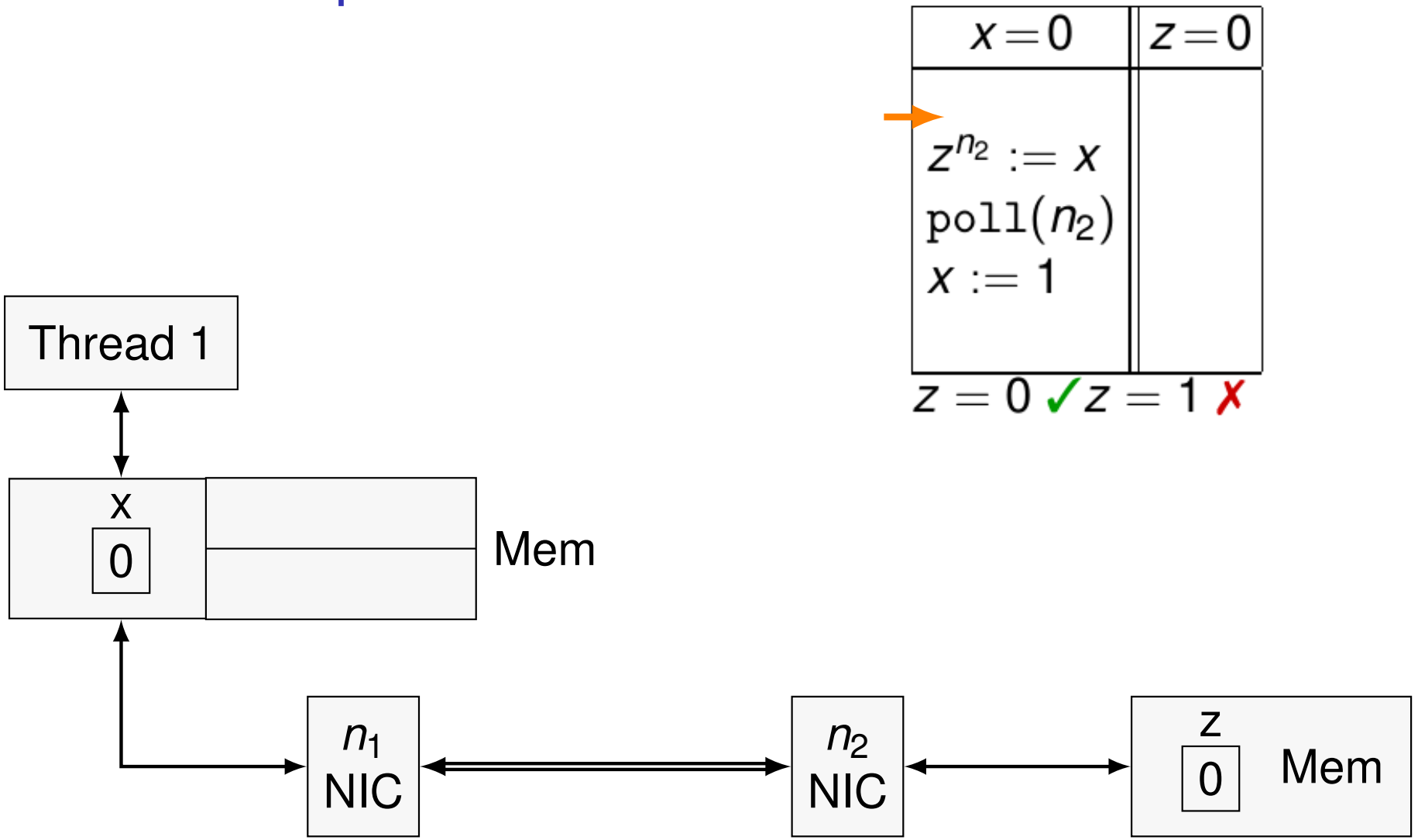
RDMA example 1/3



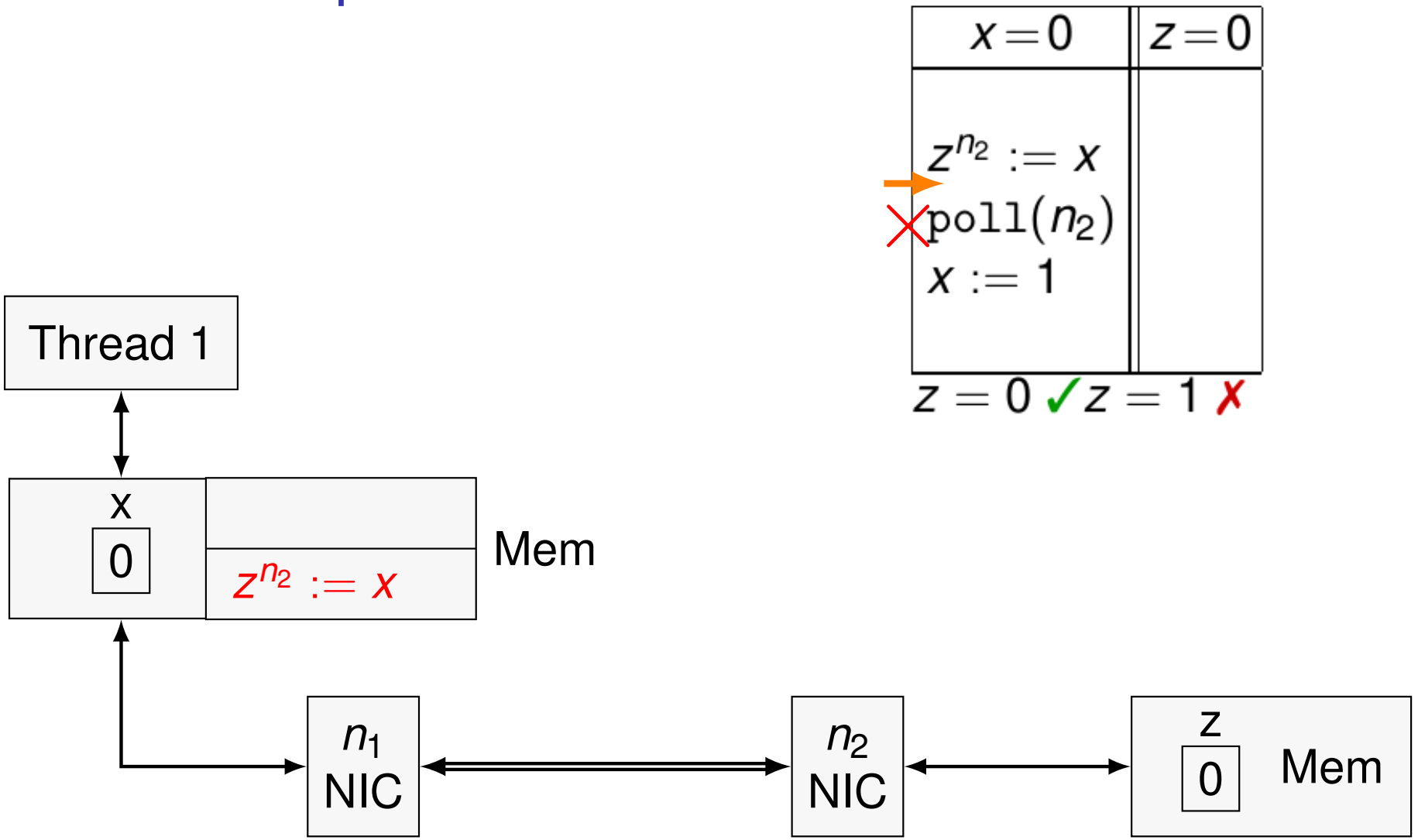
RDMA example 1/3



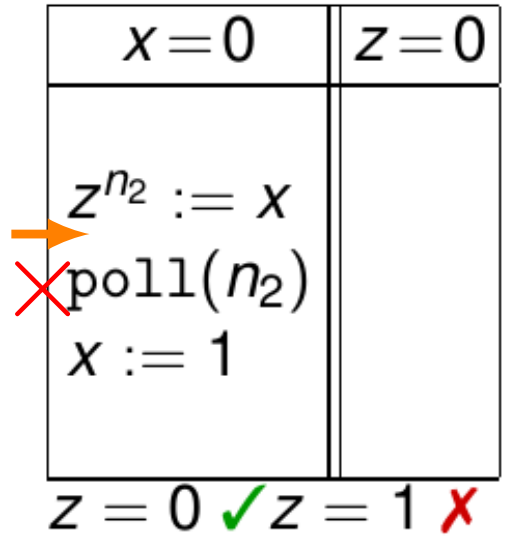
RDMA example 2/3

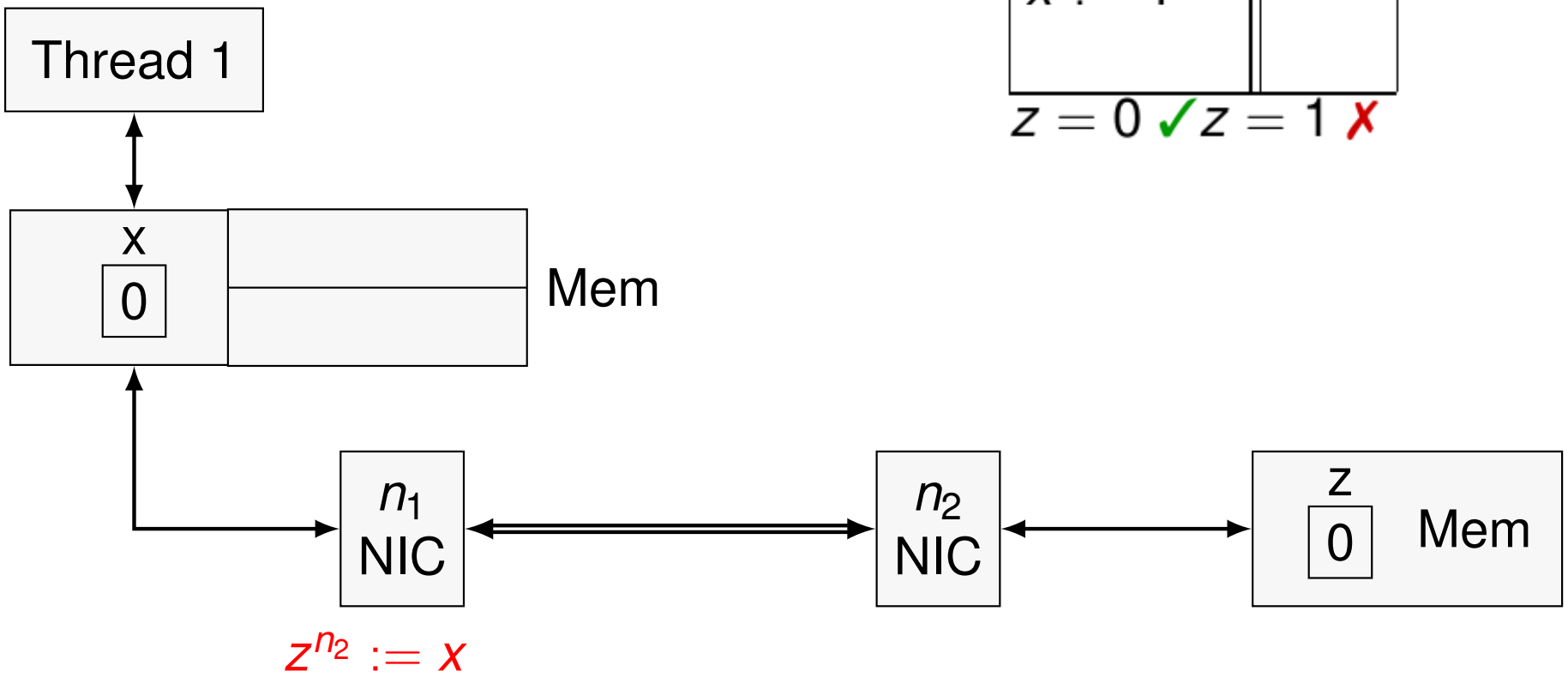


RDMA example 2/3

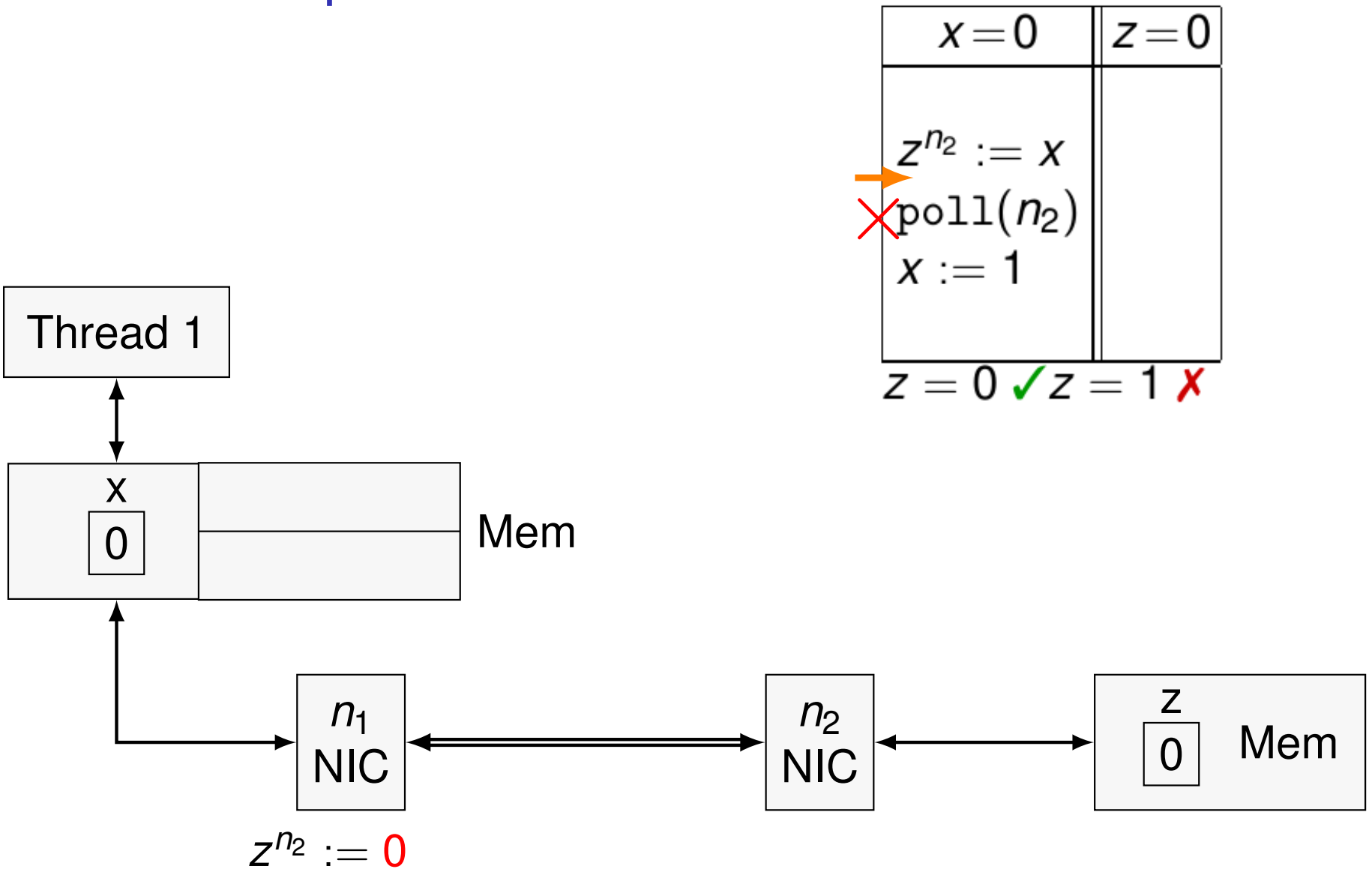


RDMA example 2/3

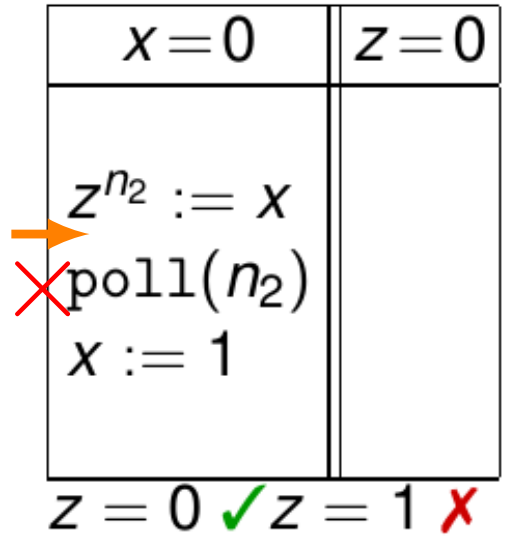


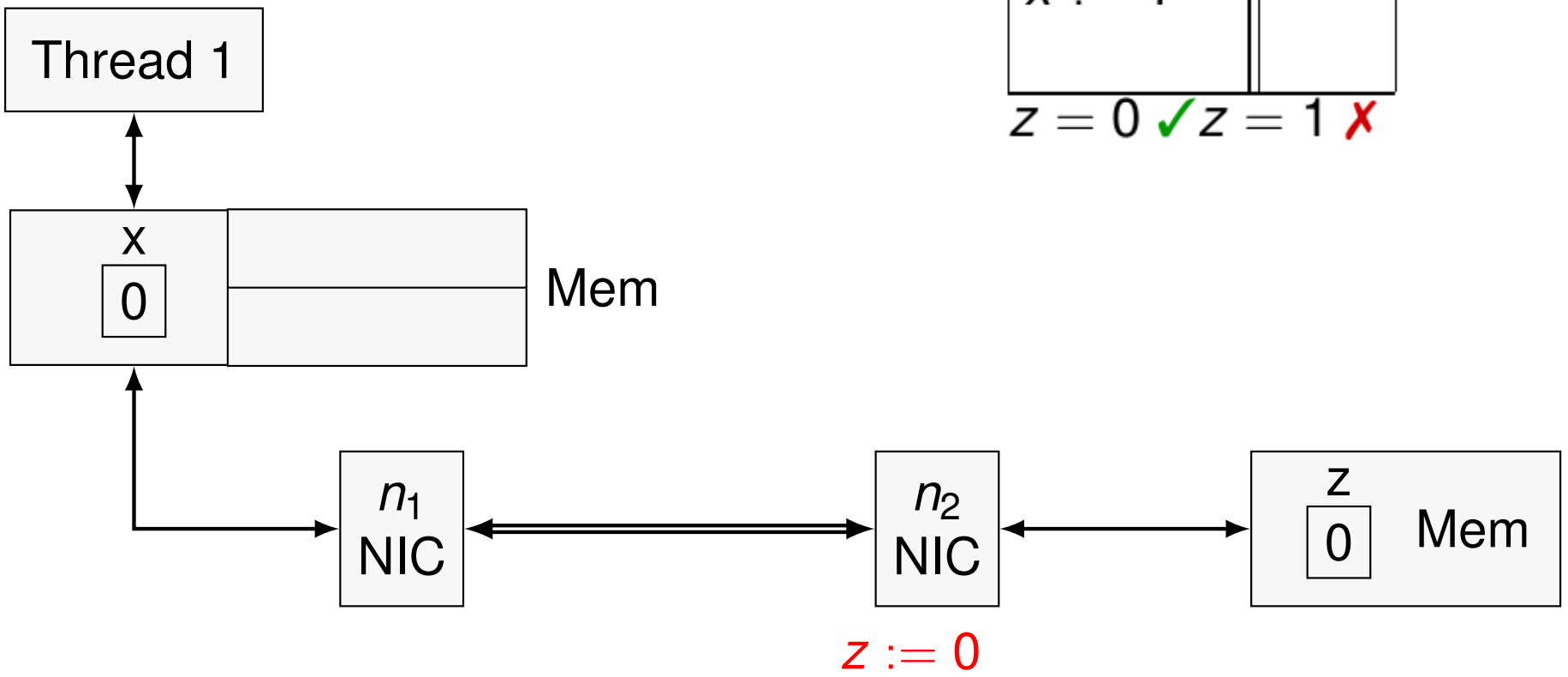


RDMA example 2/3

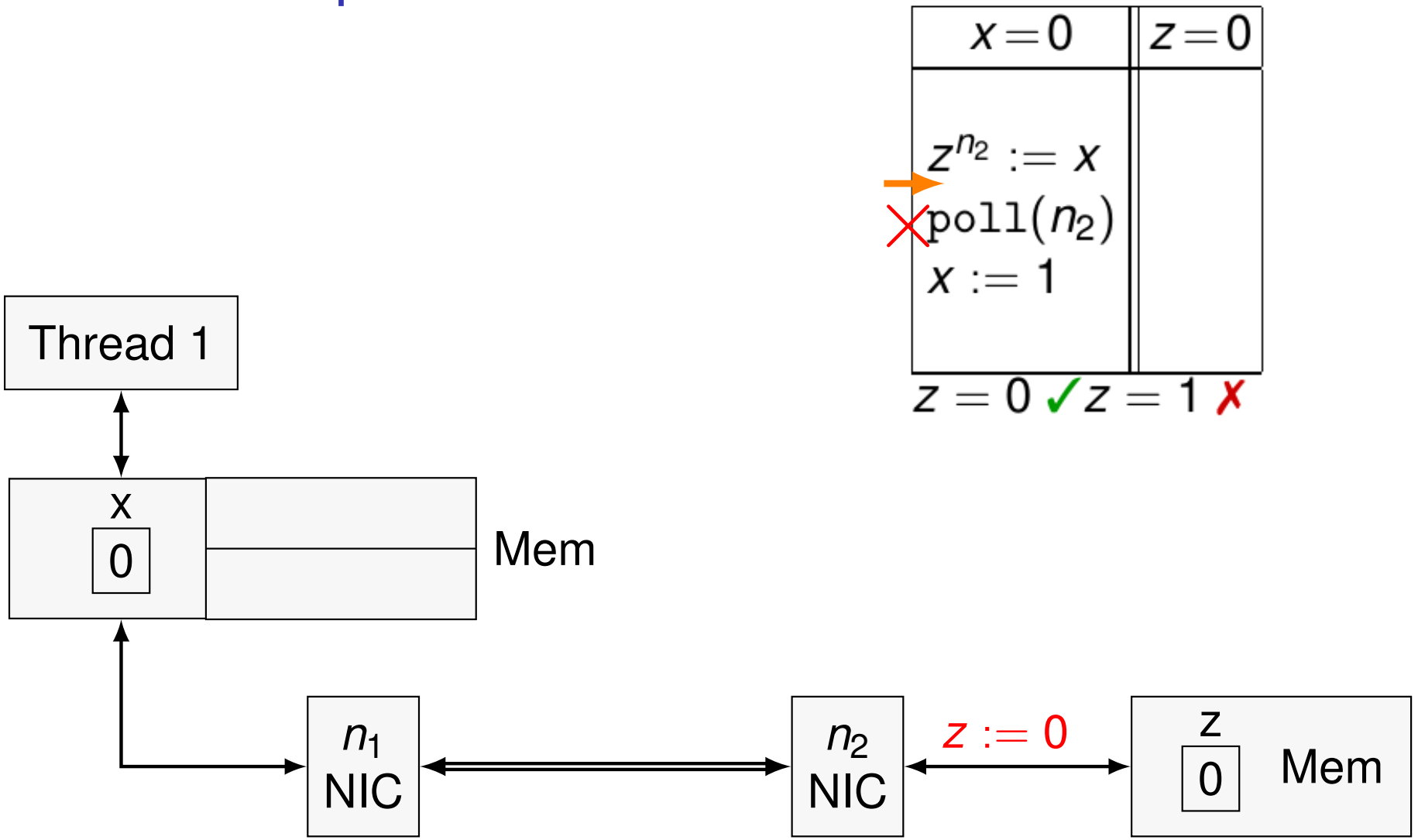


RDMA example 2/3

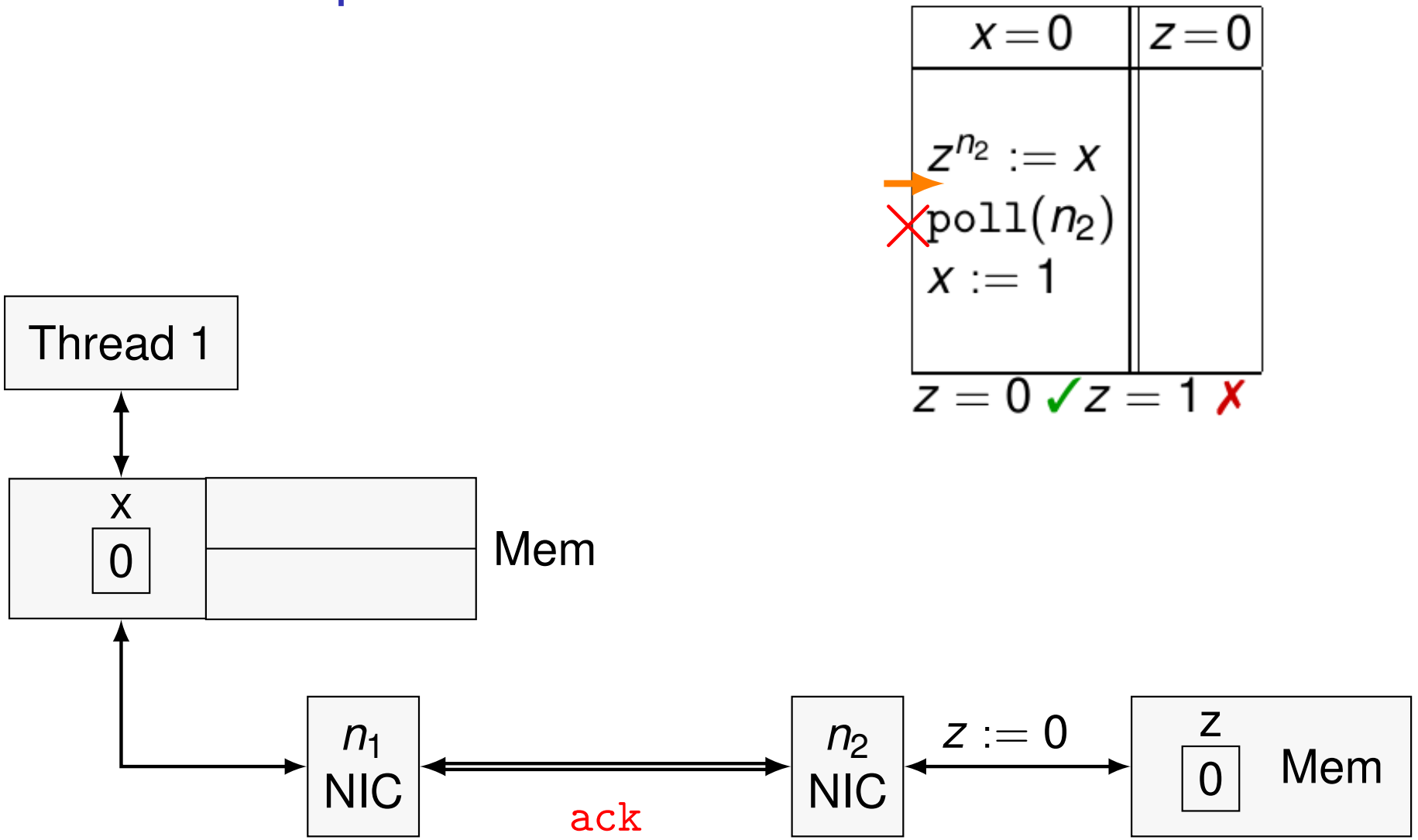




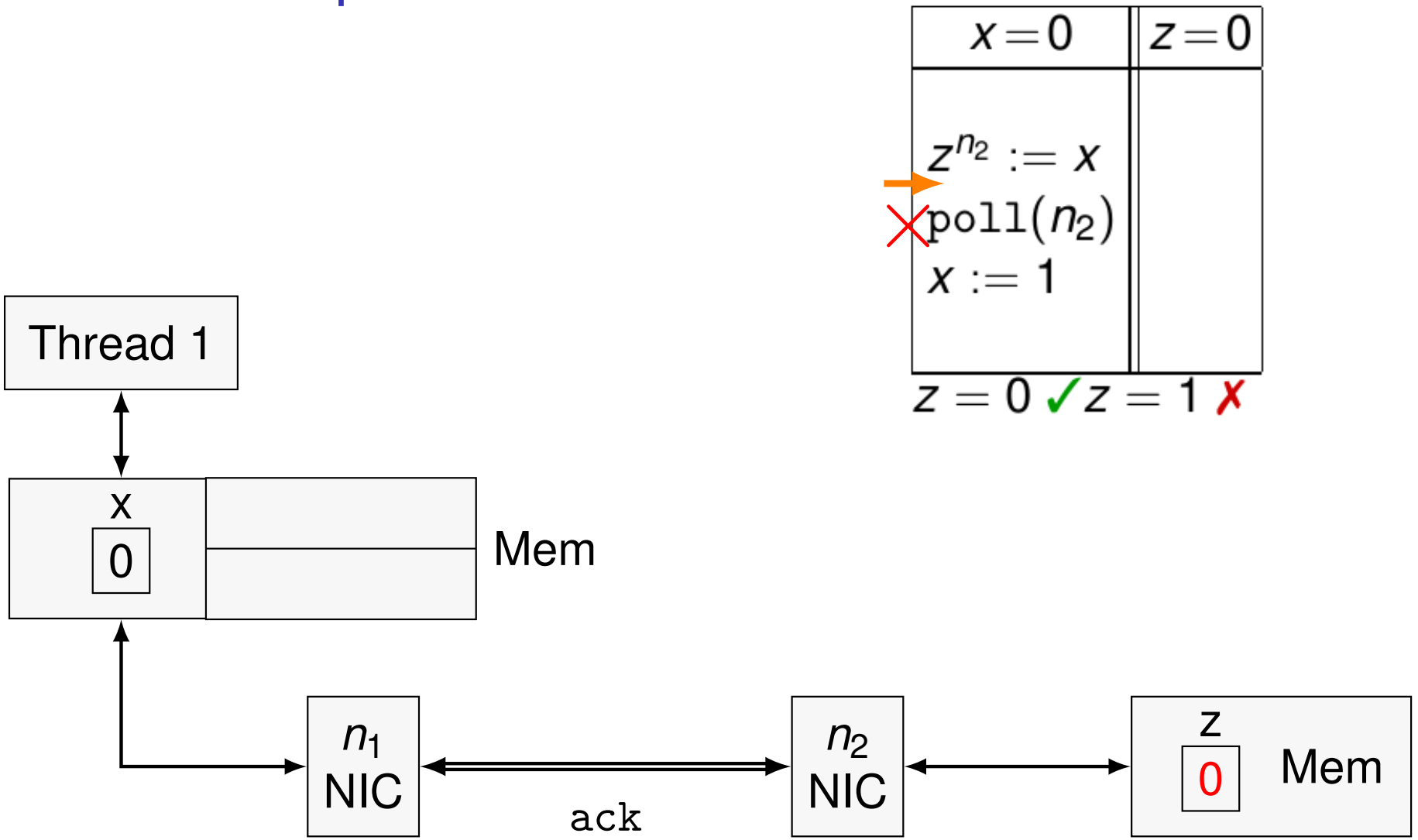
RDMA example 2/3



RDMA example 2/3



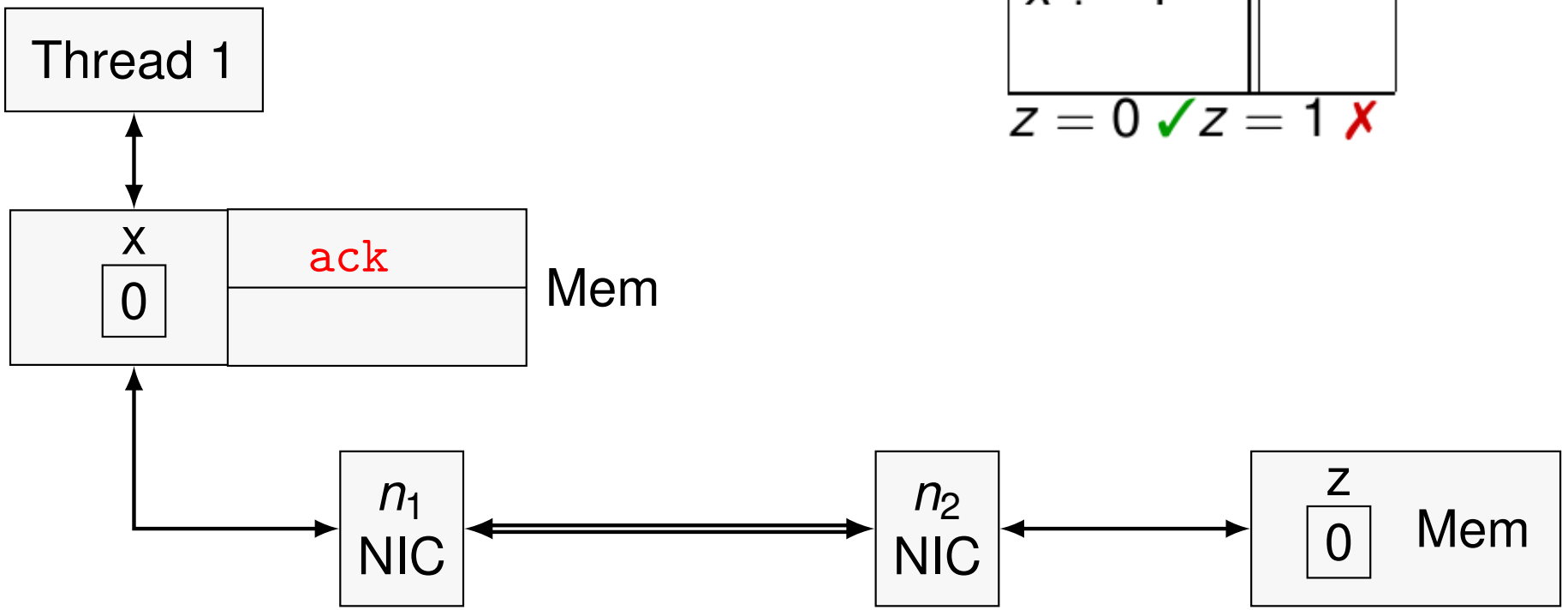
RDMA example 2/3



RDMA example 2/3

$x = 0$	$z = 0$
$z^{n_2} := x$ poll(n_2) $x := 1$	

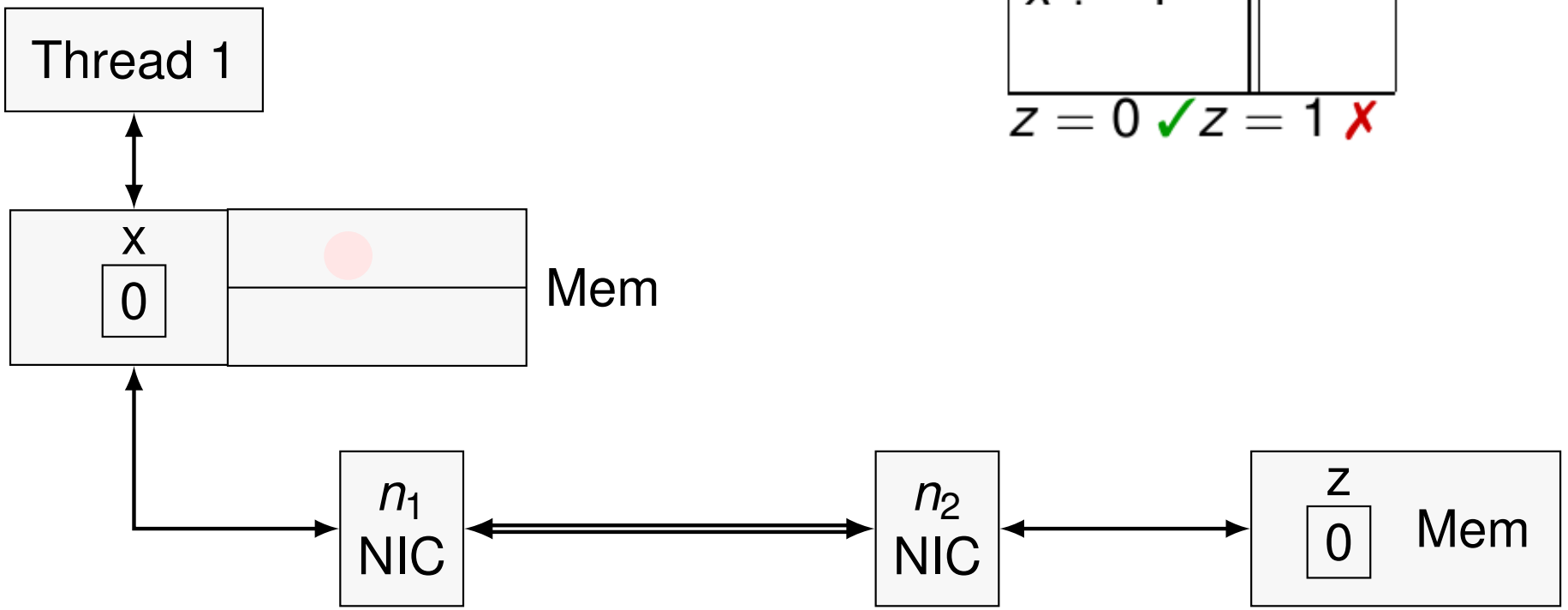
$z = 0$ ✓ $z = 1$ ✗



RDMA example 2/3

$x = 0$	$z = 0$
$z^{n_2} := x$ $\texttt{poll}(n_2)$ $x := 1$	

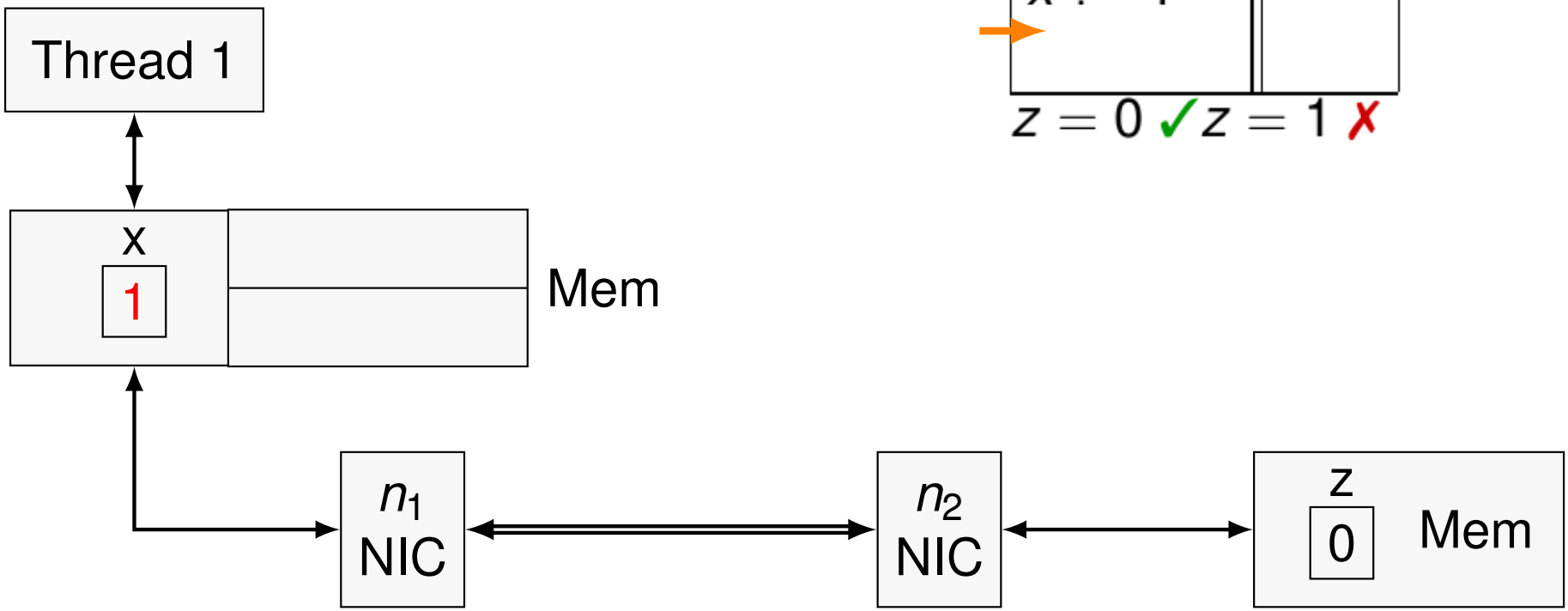
$z = 0$ ✓ $z = 1$ ✗



RDMA example 2/3

$x = 0$	$z = 0$
$z^{n_2} := x$ $\texttt{poll}(n_2)$ $x := 1$	

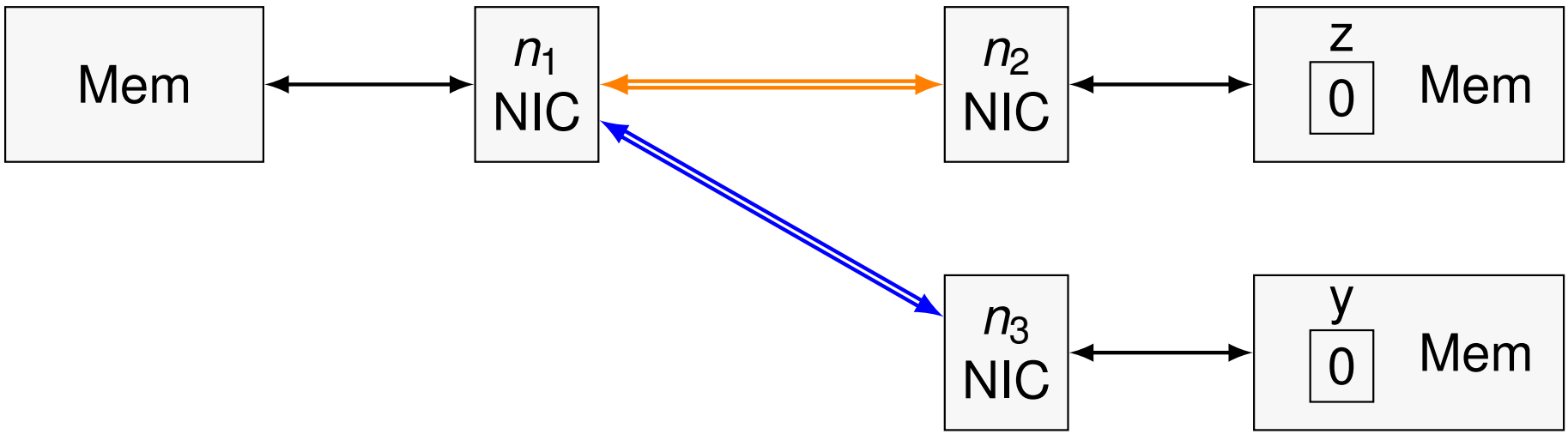
$z = 0$ ✓ $z = 1$ ✗



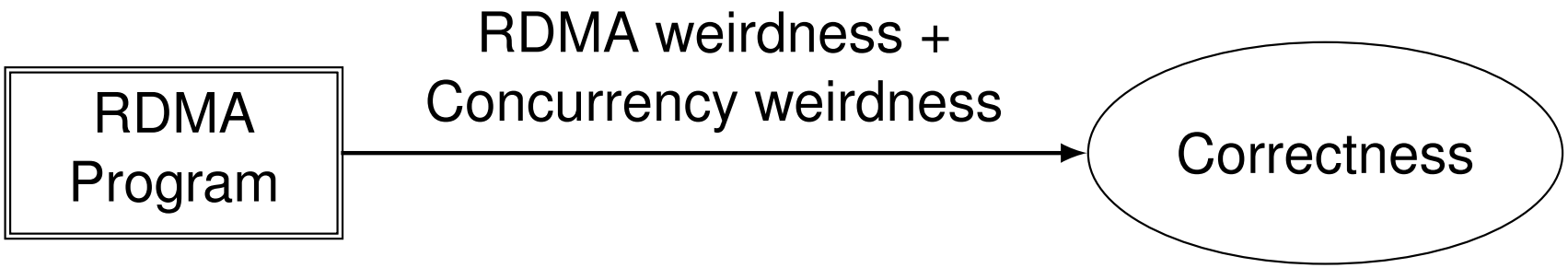
RDMA example 3/3

Independent paths are independent

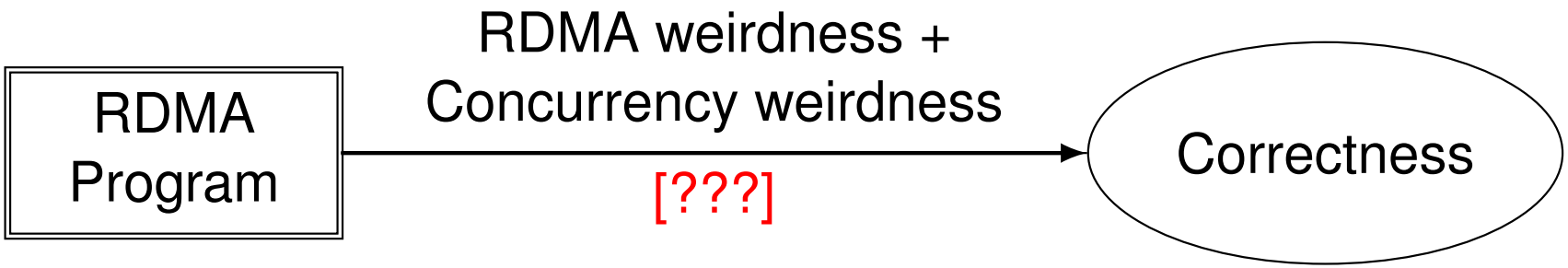
	$y = 0$	$z = 0$
$z^{n_3} := 1$ $y^{n_2} := 1$		



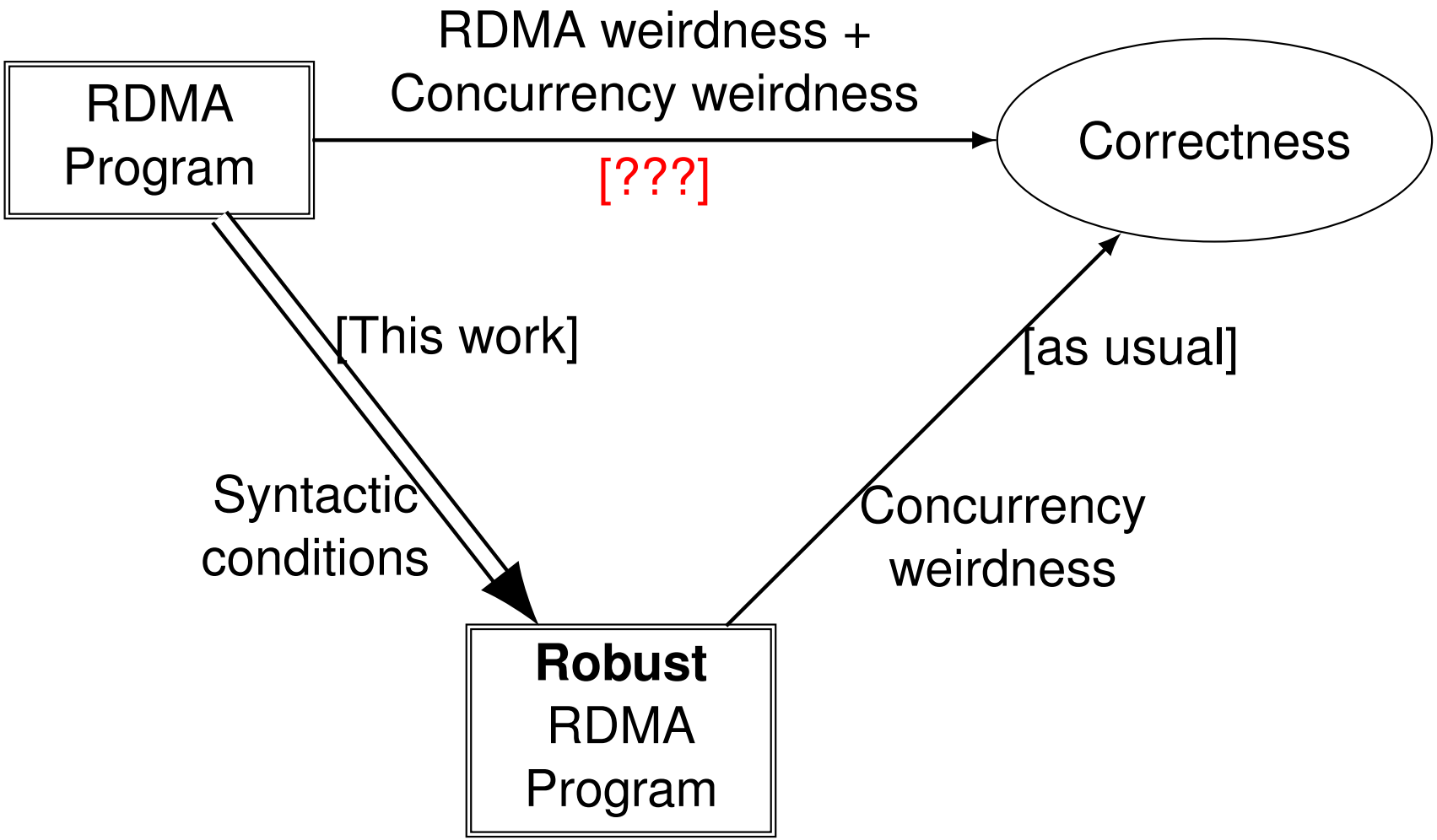
Formal Verification



Formal Verification



Formal Verification



SC and Robustness

Sequential Consistency (noun): Theoretical memory model where computers execute programs in order, line by line.

a.k.a.: **no fun allowed.**

Robust (adj.): Program whose outputs are achievable under SC.

a.k.a.: **it's not cheating unless you get caught!**

Preventing Reorderings

Result 1: existing RDMA programs can be made robust

Strategy: when in doubt, prevent reorderings

Preventing Reorderings

Result 1: existing RDMA programs can be made robust

	$y = 0$	$z = 0$
$z^{n_3} := 1$ $y^{n_2} := 1$		

ROBUST

Preventing Reorderings

Result 1: existing RDMA programs can be made robust

	$y = 0$	$z = 0$
$z^{n_3} := 1$ $y^{n_2} := 1$	$a := y$ $b := z^{n_3}$	

NOT ROBUST

$(a, b) = (1, 0)$

Preventing Reorderings

Result 1: existing RDMA programs can be made robust

	$y = 0$	$z = 0$
$z^{n_3} := 1$ `poll`(n_3) $y^{n_2} := 1$	$a := y$ $b := z^{n_3}$	

NOT ROBUST

$(a, b) = (1, 0)$

Preventing Reorderings

Result 1: existing RDMA programs can be made robust

$x = 0$	$y = 0$	$w, z = 0$
$z^{n_3} := 1$ $x := w^{n_3}$ $\texttt{poll}(n_3)$ $\texttt{poll}(n_3)$ $y^{n_2} := 1$	$a := y$ $b := z^{n_3}$	

ROBUST

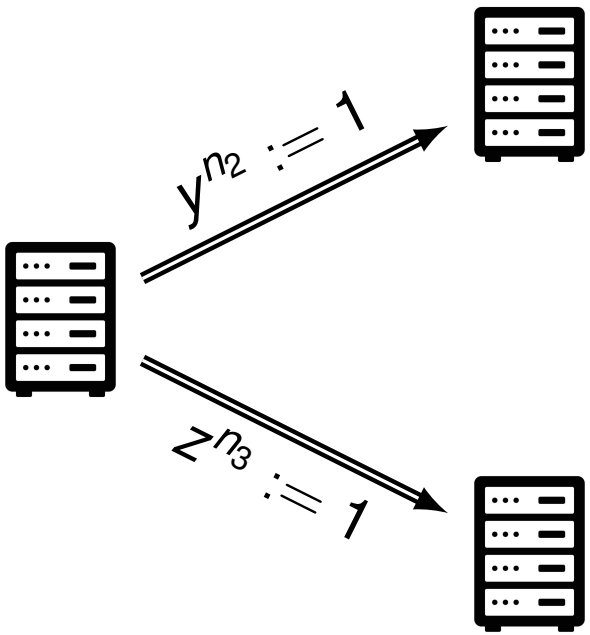
Tree topology

Result 2: programming guidelines for robustness

(a.k.a.: how to not get caught)

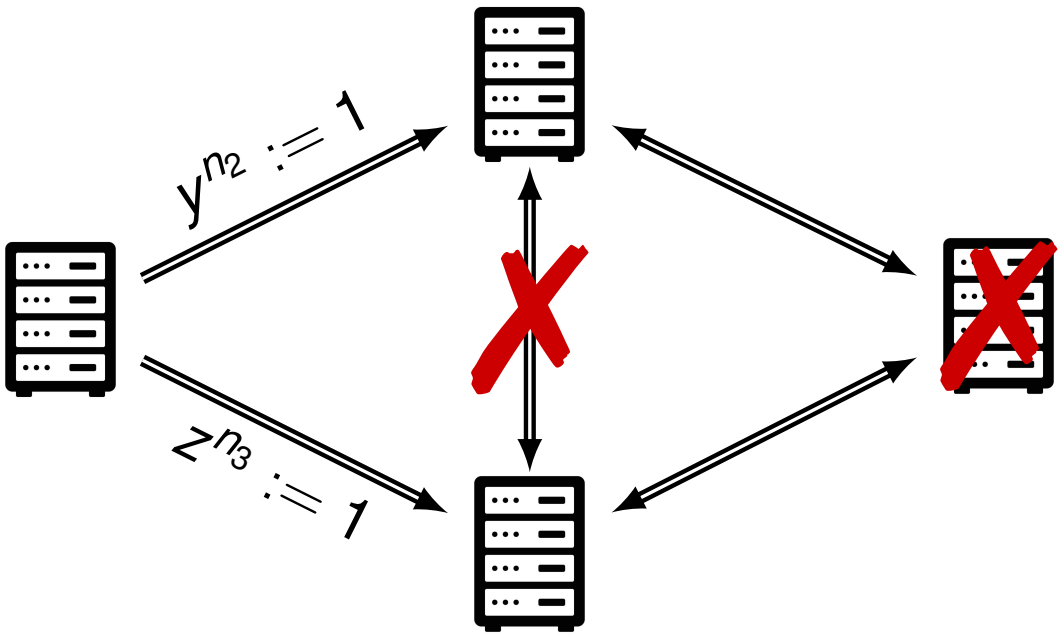
Tree topology

Result 2: programming guidelines for robustness



Tree topology

Result 2: programming guidelines for robustness



Condition 1: Tree topology

Maintaining plausible deniability

Result 2: programming guidelines for robustness

Standard RDMA usage:

$$x := y^n$$
$$\vdots$$
$$\vdots$$
$$f(x)$$

Maintaining plausible deniability

Result 2: programming guidelines for robustness

Standard RDMA usage:

$$x := y^n$$
$$\vdots$$
$$\vdots$$
$$f(x)$$

NOT ROBUST

Maintaining plausible deniability

Result 2: programming guidelines for robustness

Standard RDMA usage:

$$
\begin{array}{l}
x := y^n \\
\vdots \\
\vdots \\
\texttt{poll}(n) \\
f(x)
\end{array}
$$

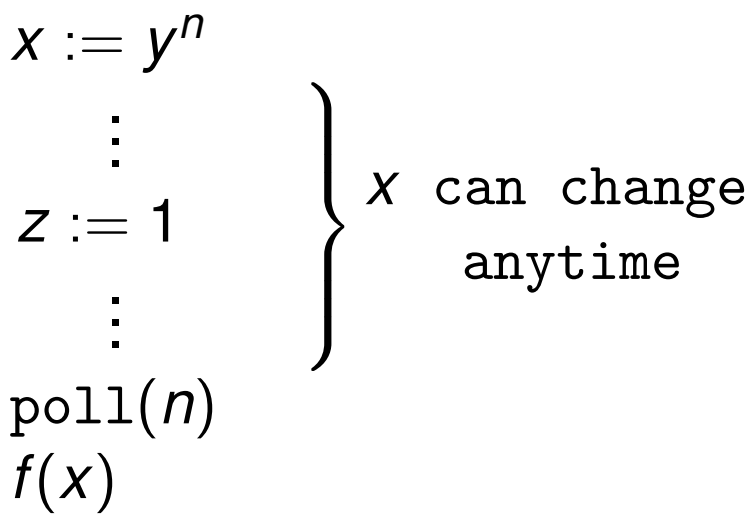
ROBUST

Condition 2: Poll before reusing

Maintaining plausible deniability

Result 2: programming guidelines for robustness

Standard RDMA usage:

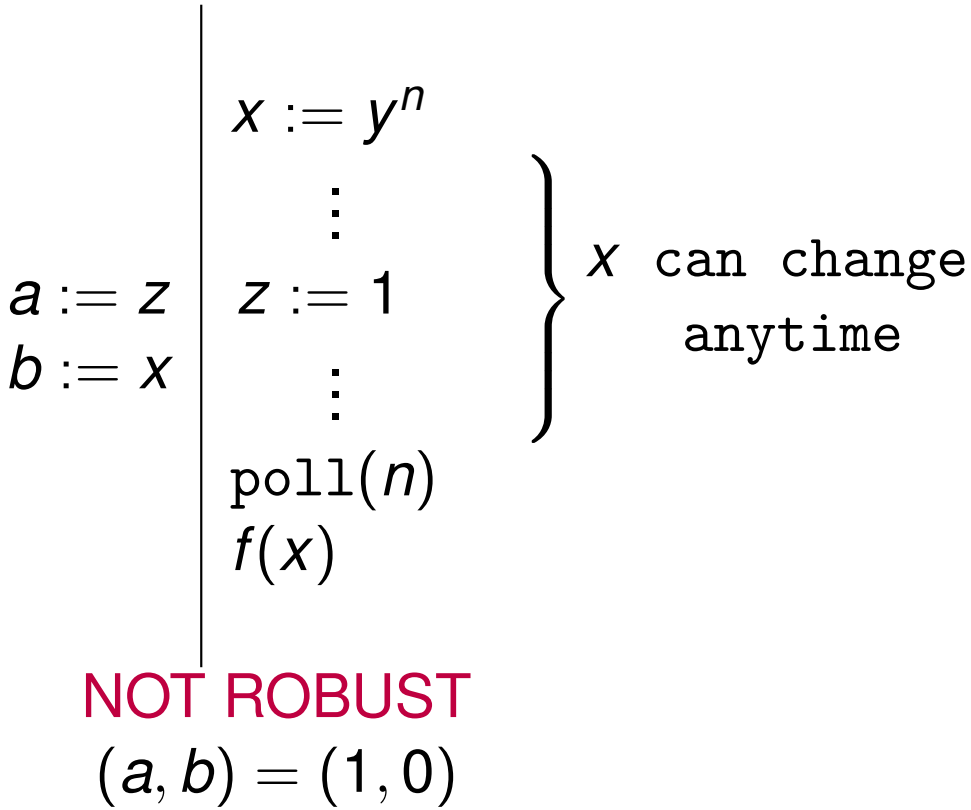


ROBUST

Maintaining plausible deniability

Result 2: programming guidelines for robustness

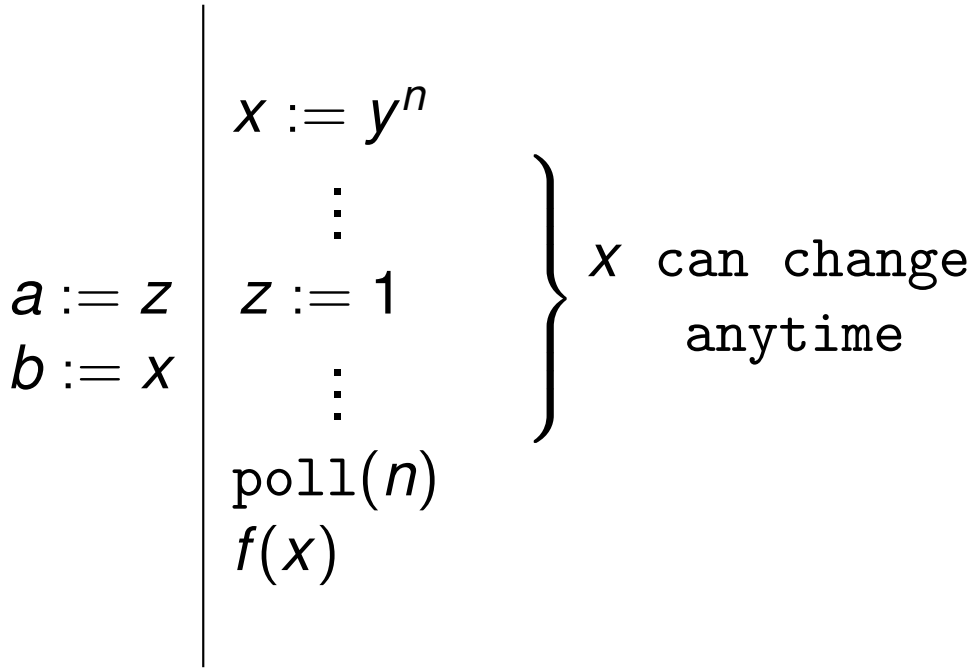
Standard RDMA usage:



Maintaining plausible deniability

Result 2: programming guidelines for robustness

Standard RDMA usage:

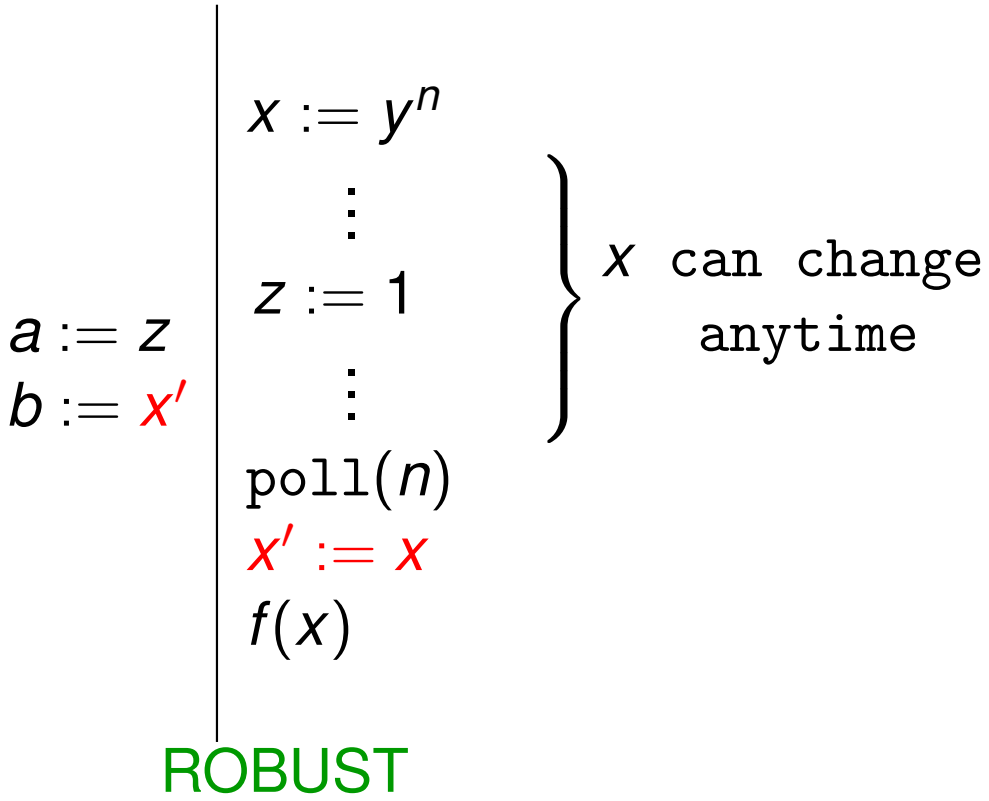


Condition 3: No snitches
(locations for RDMA operations are private)

Maintaining plausible deniability

Result 2: programming guidelines for robustness

Standard RDMA usage:



Theorem

Result 2: programming guidelines for robustness

- Tree topology
- Poll before reusing
- No snitches

$\Longrightarrow$ Robustness

Conclusion

RDMA

- Low-latency data-transfer protocol for HPC/databases
- Recently becoming widespread
- Very weak and unintuitive semantics

Our work: RDMA robustness (assuming x86-TSO for the CPU)

- Existing RDMA programs can be made robust (and probably slow)
- With tree topology, can write robust RDMA programs for cheap
- Else... good luck for your formal verification