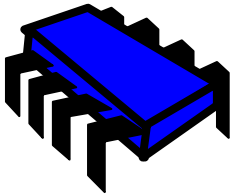
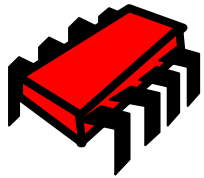
Two Elementary Instructions make Compare-and-Swap

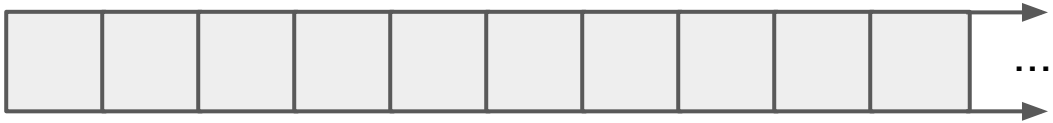


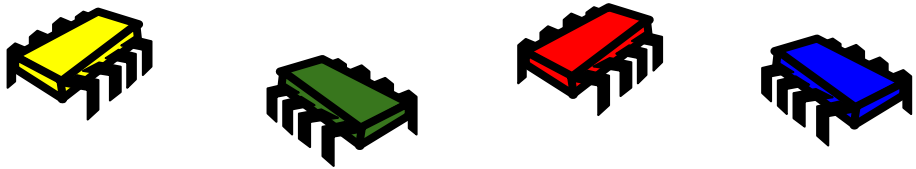
Pankaj Khanchandani, Roger Wattenhofer
ETH Zurich - Distributed Computing Group (DISCO)



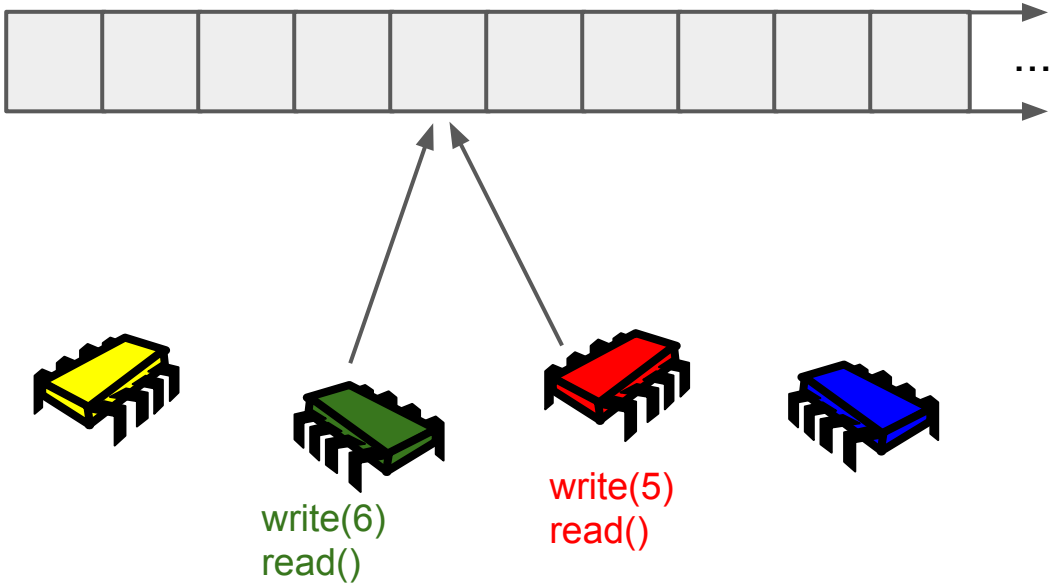
Compare-and-Swap

Shared Memory Model

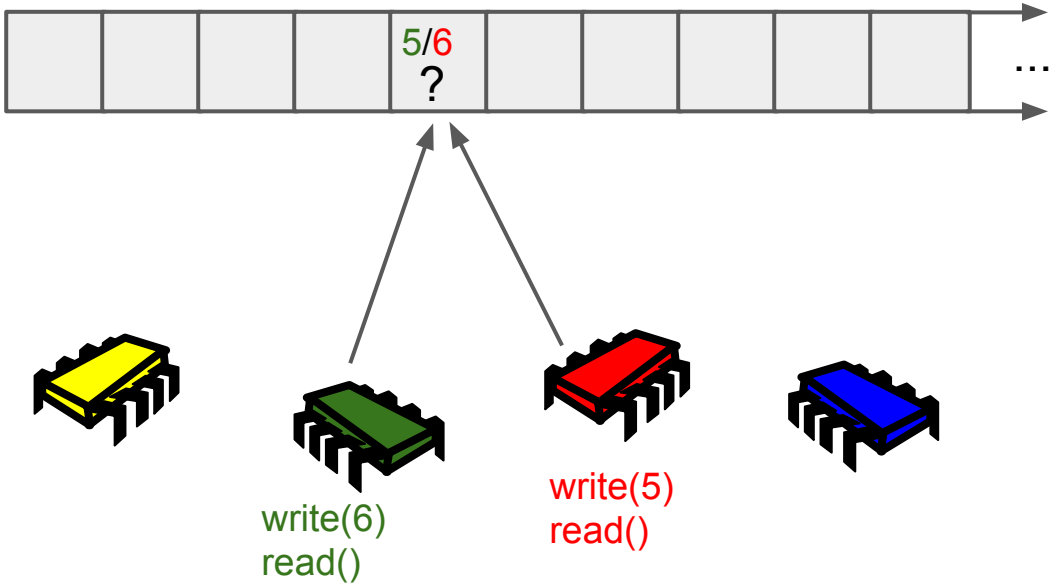




Shared Memory Model

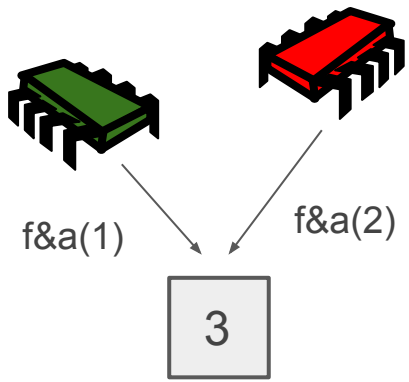


Shared Memory Model



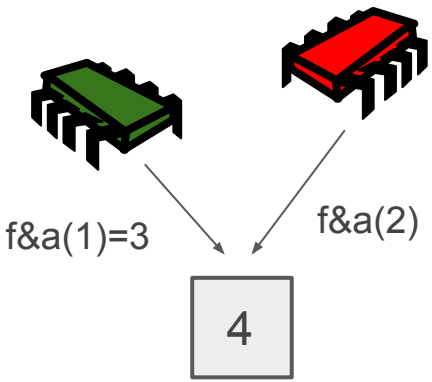
Atomic Registers

fetch-and-add(x)

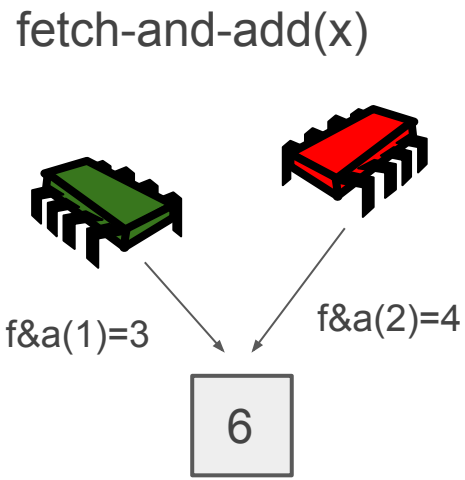


Atomic Registers

fetch-and-add(x)

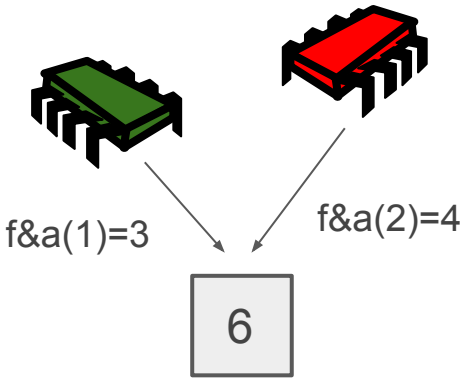


Atomic Registers

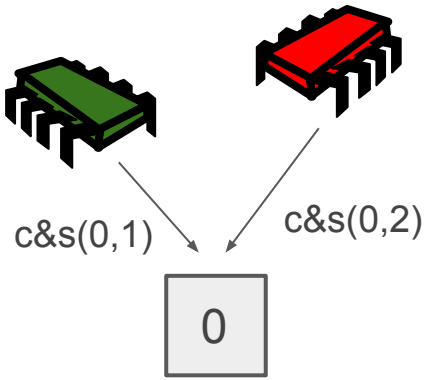


Atomic Registers

fetch-and-add(x)

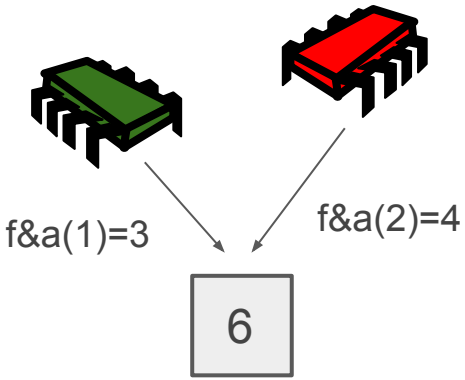


compare-and-swap(x,y)

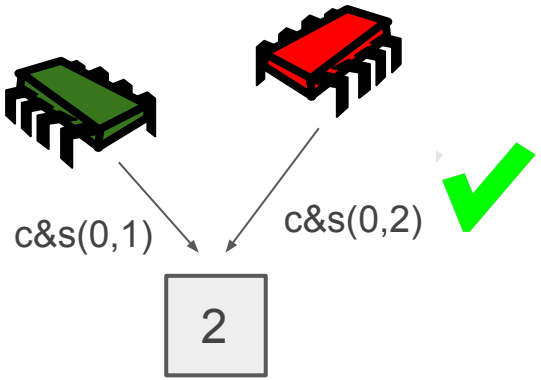


Atomic Registers

fetch-and-add(x)

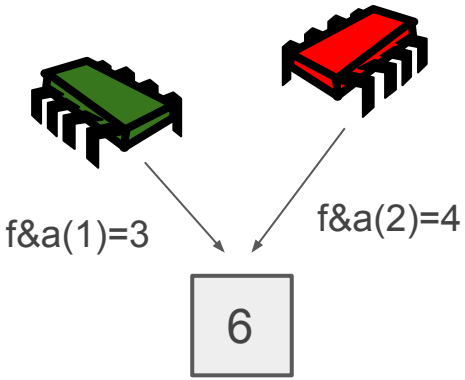


compare-and-swap(x,y)

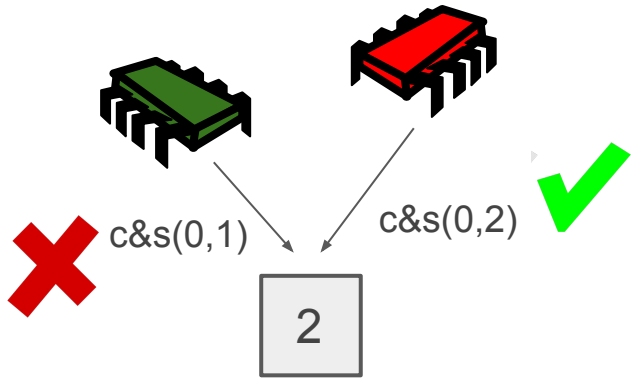


Atomic Registers

fetch-and-add(x)



compare-and-swap(x,y)



Best Atomic Registers?

Consensus Numbers [Herlihy 1991]

compare-and-swap = ∞

…

n-register assignment = 2n-2

...

fetch-and-add = 2

read/write = 1

Best Atomic Registers?

[Ruppert 1997]

compare-and-swap = ∞

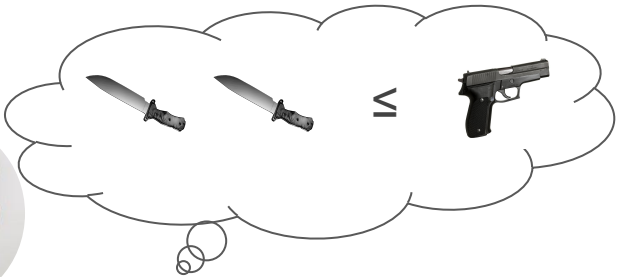
…

n-register assignment = 2n-2

…

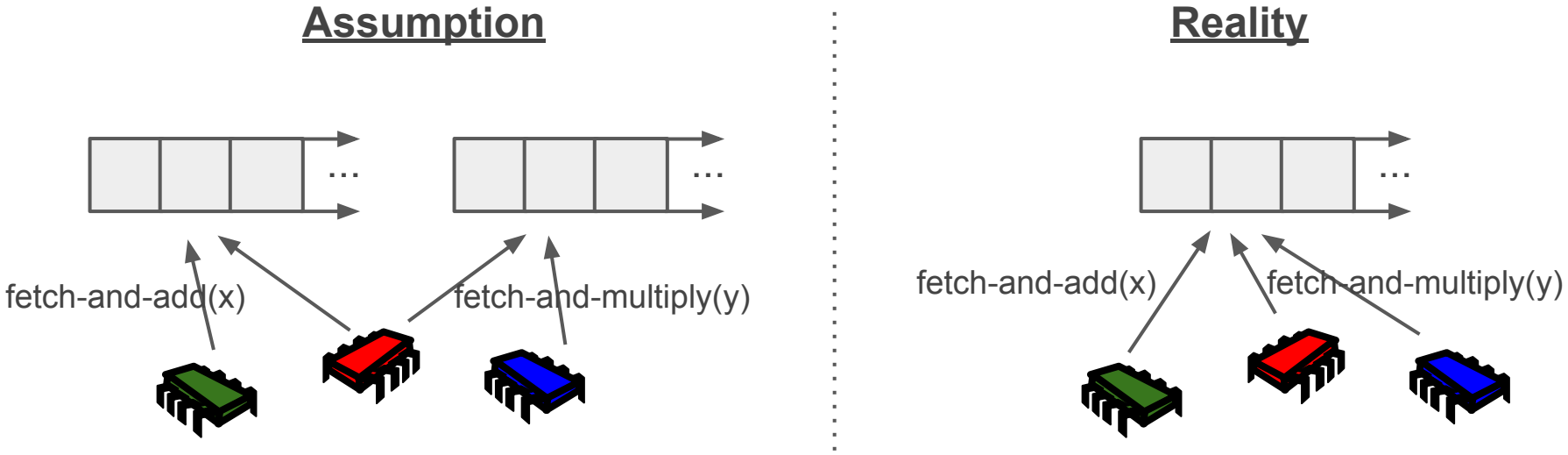
fetch-and-add = 2

read/write = 1



Best Atomic Registers?

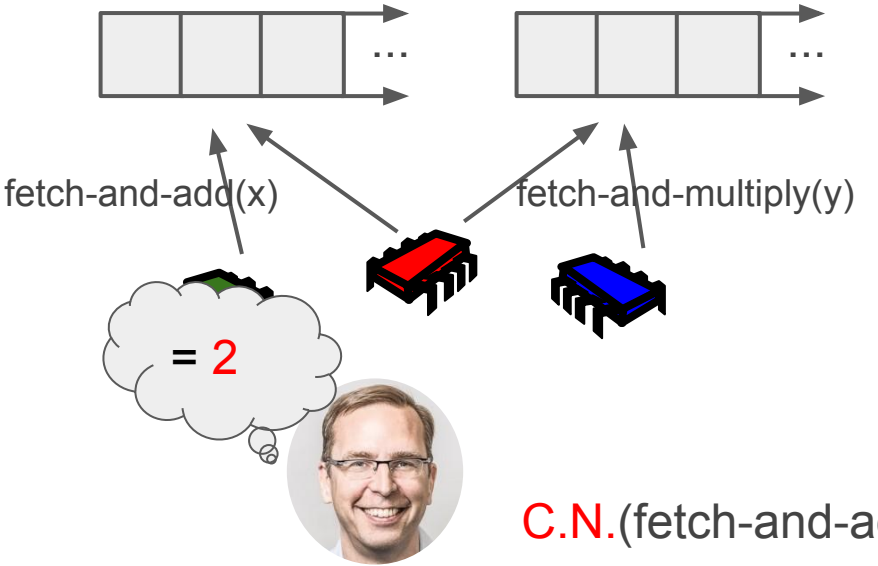
[Ellen et al. 2016]



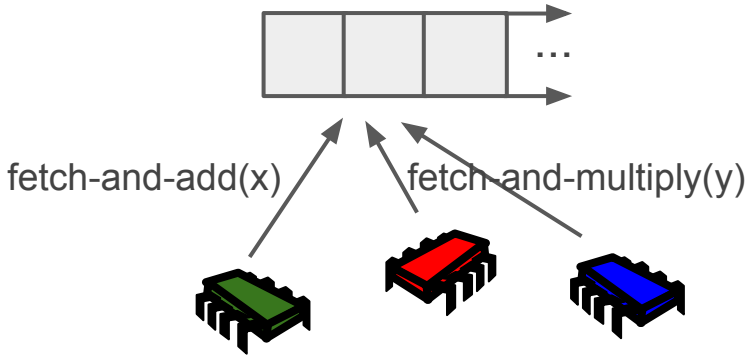
Best Atomic Registers?

[Ellen et al. 2016]

Assumption



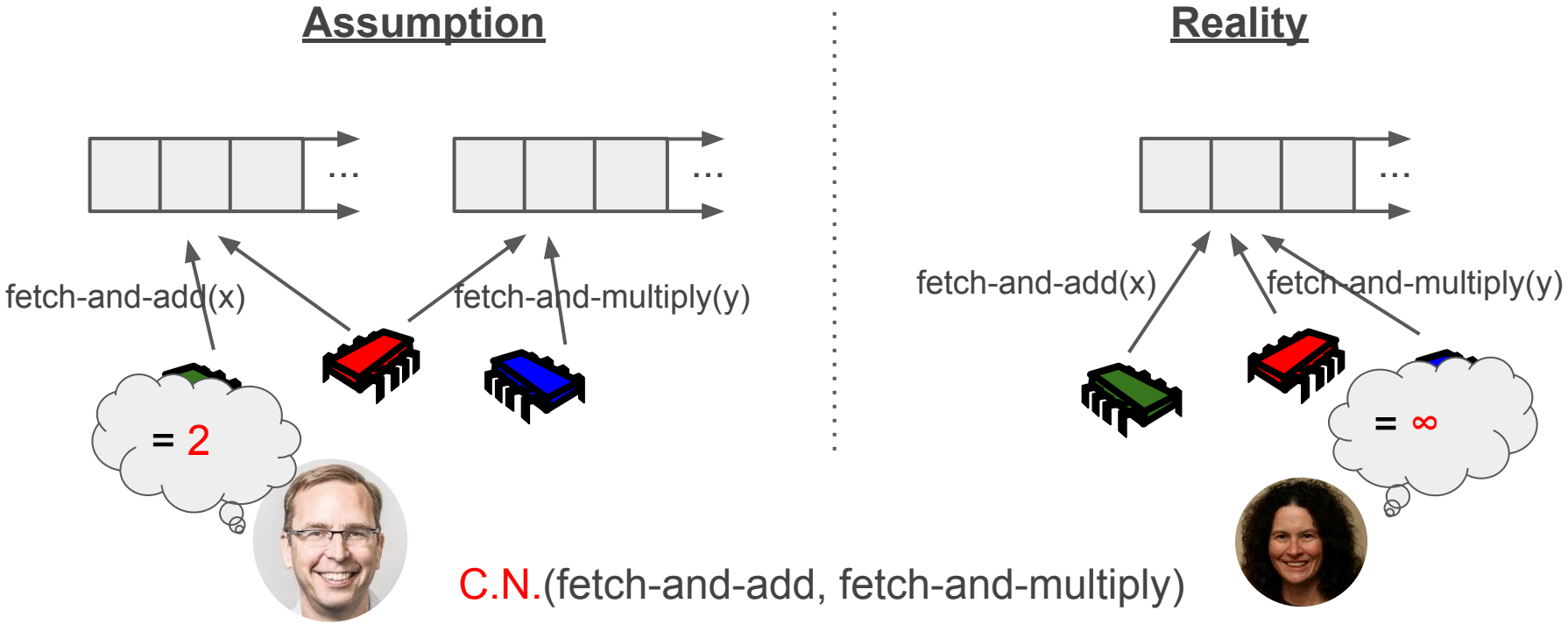
Reality



C.N.(fetch-and-add, fetch-and-multiply)

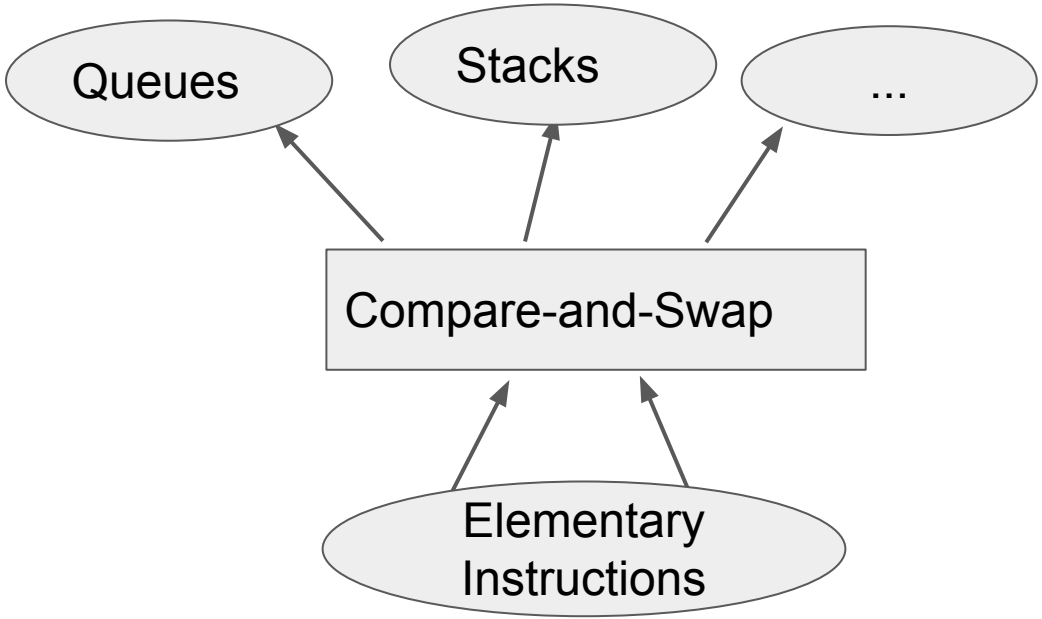
Best Atomic Registers?

[Ellen et al. 2016]



C.N.(fetch-and-add, fetch-and-multiply)

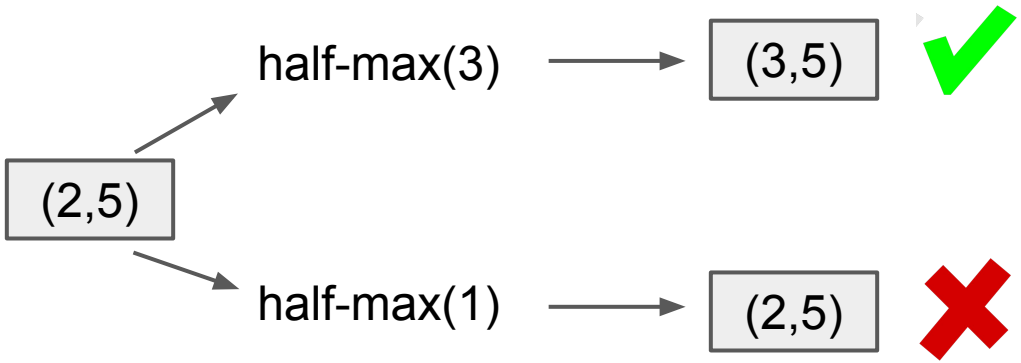
Best Atomic Registers?



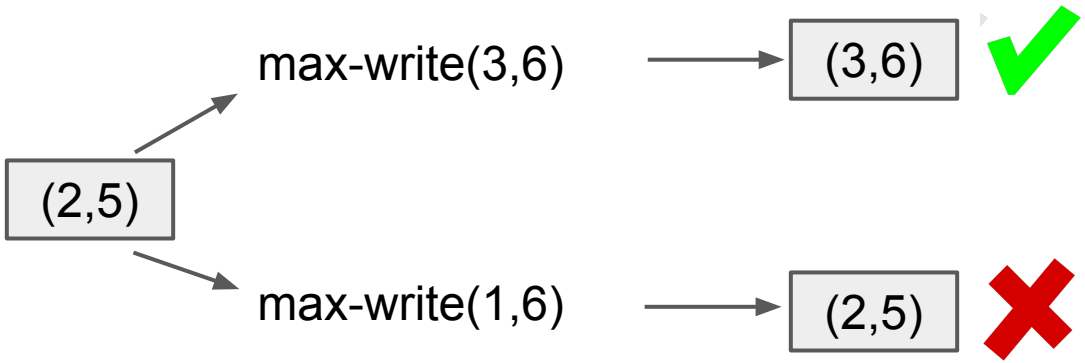
This Work

Half-max and max-write operations can simulate Compare-and-Swap in O(1) steps, in linearizable and wait-free manner.

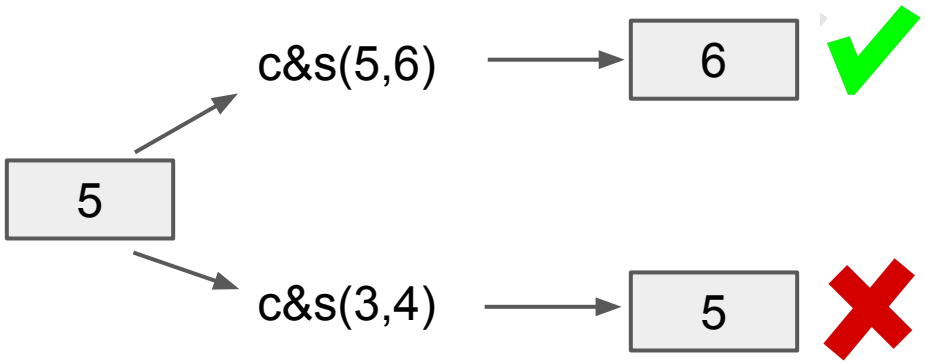
This Work: half-max, max-write



This Work: half-max, max-write



This Work: half-max, max-write



Simulation Idea

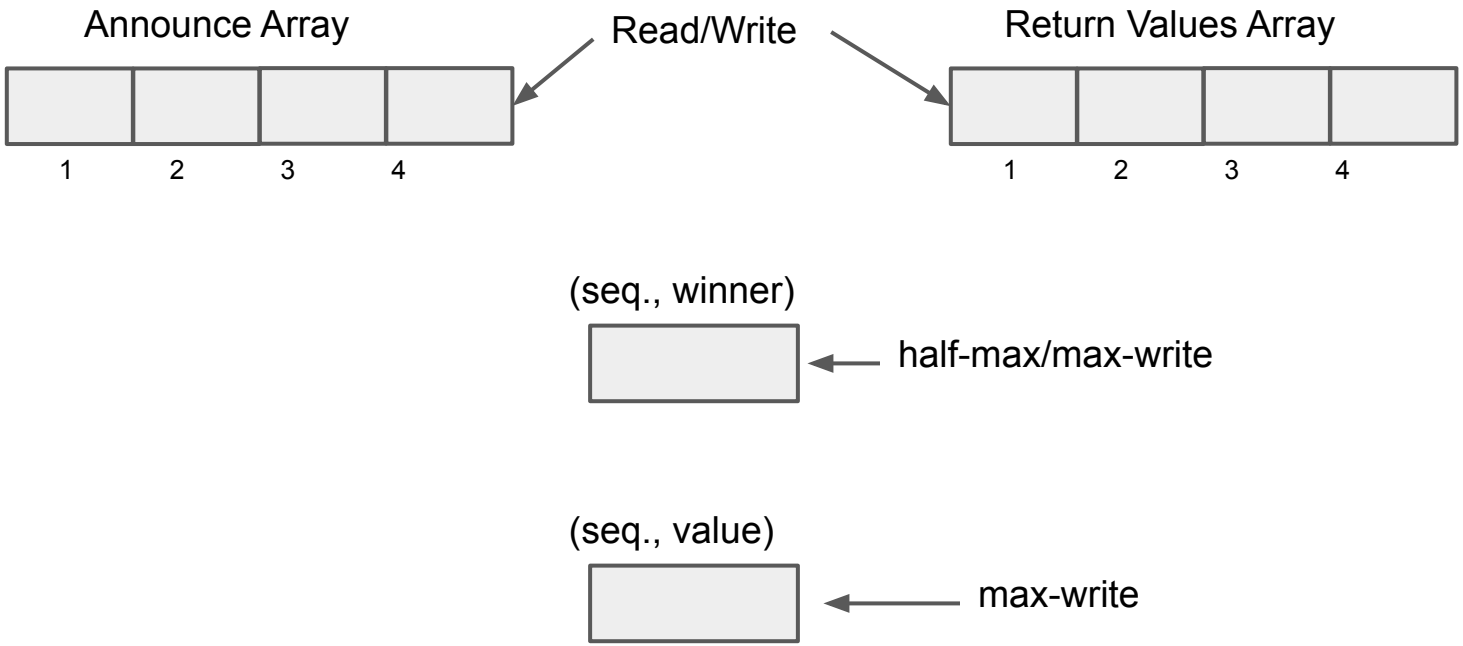
Announce Array

Return Values Array

(seq., winner)

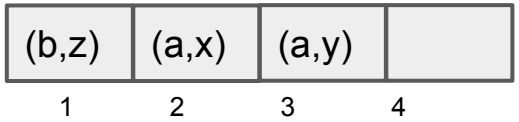
(seq., value)

Simulation Idea

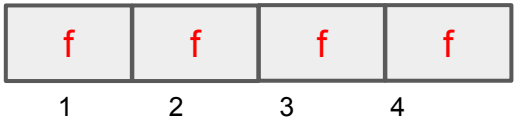


Simulation Idea

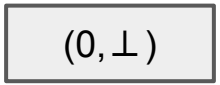
Announce Array



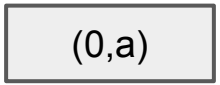
Return Values Array



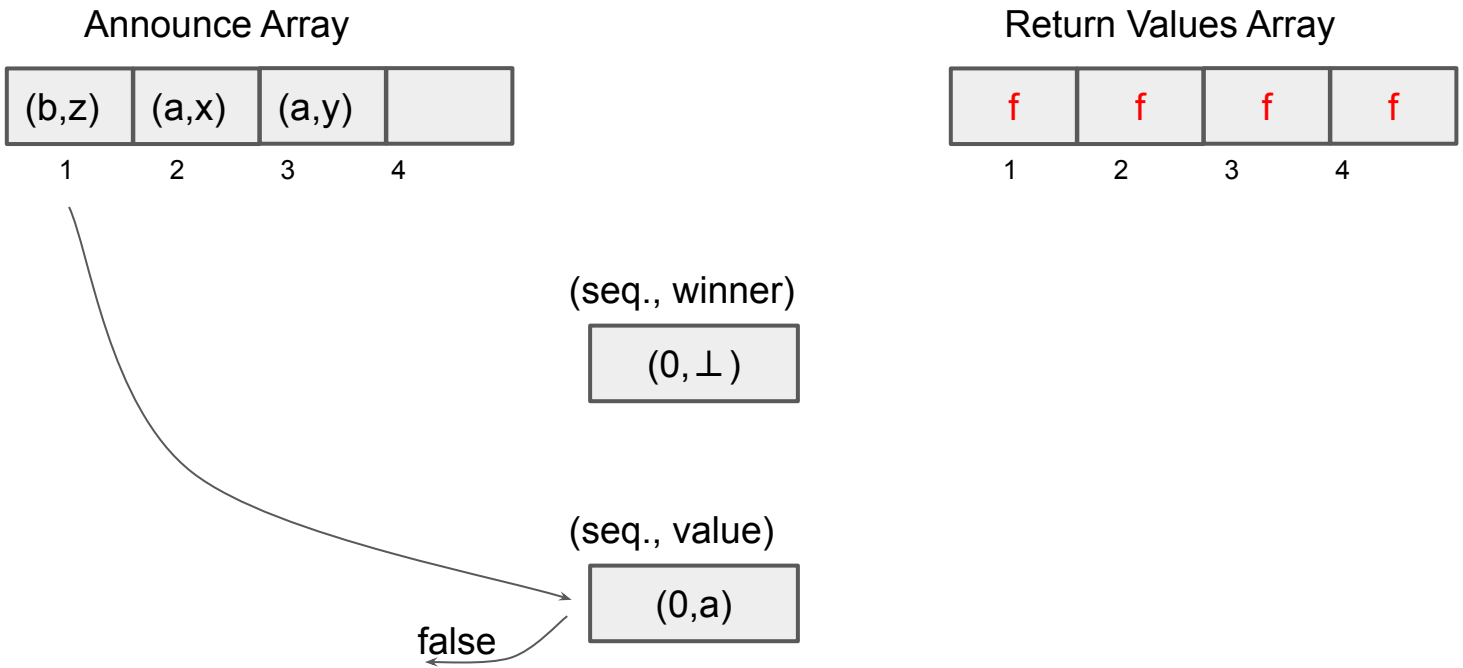
(seq., winner)



(seq., value)

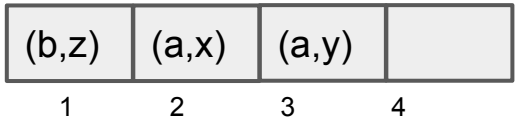


Simulation Idea

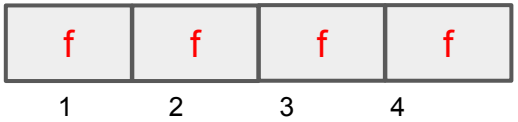


Simulation Idea

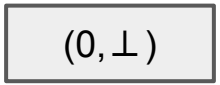
Announce Array



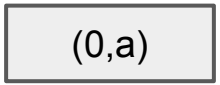
Return Values Array



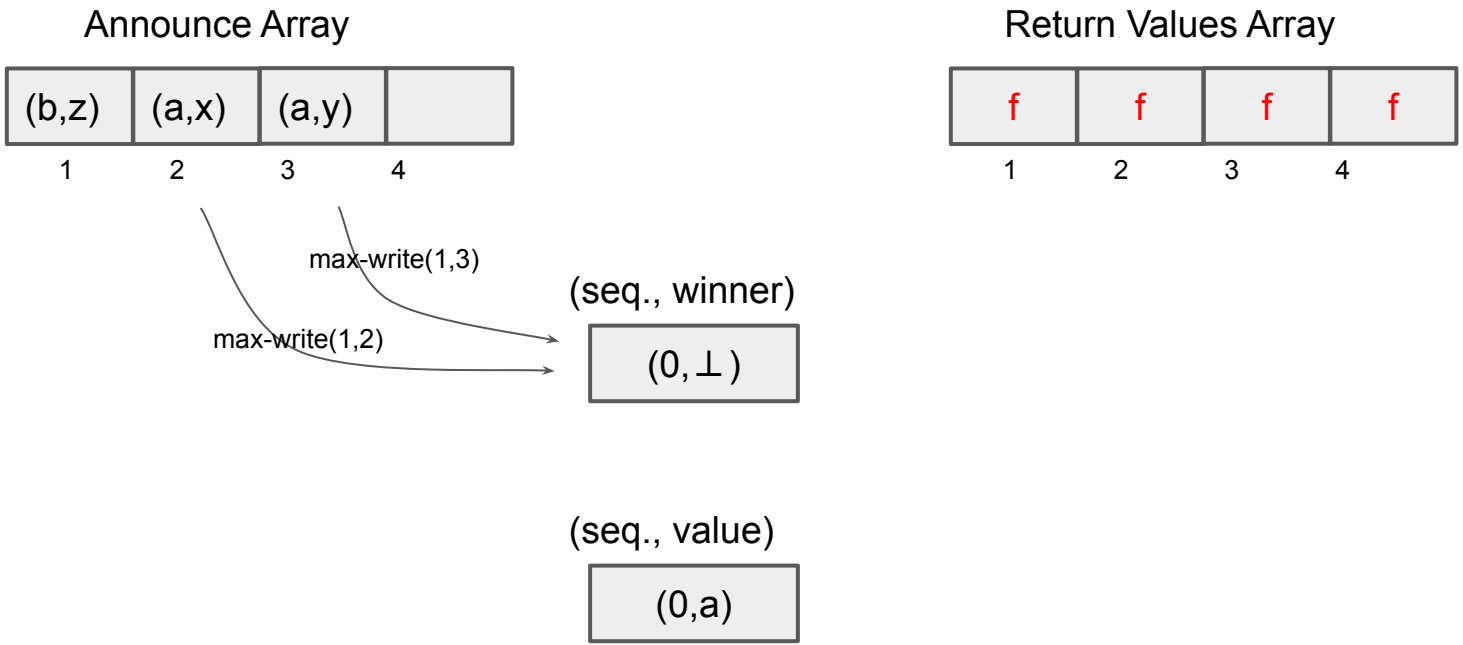
(seq., winner)



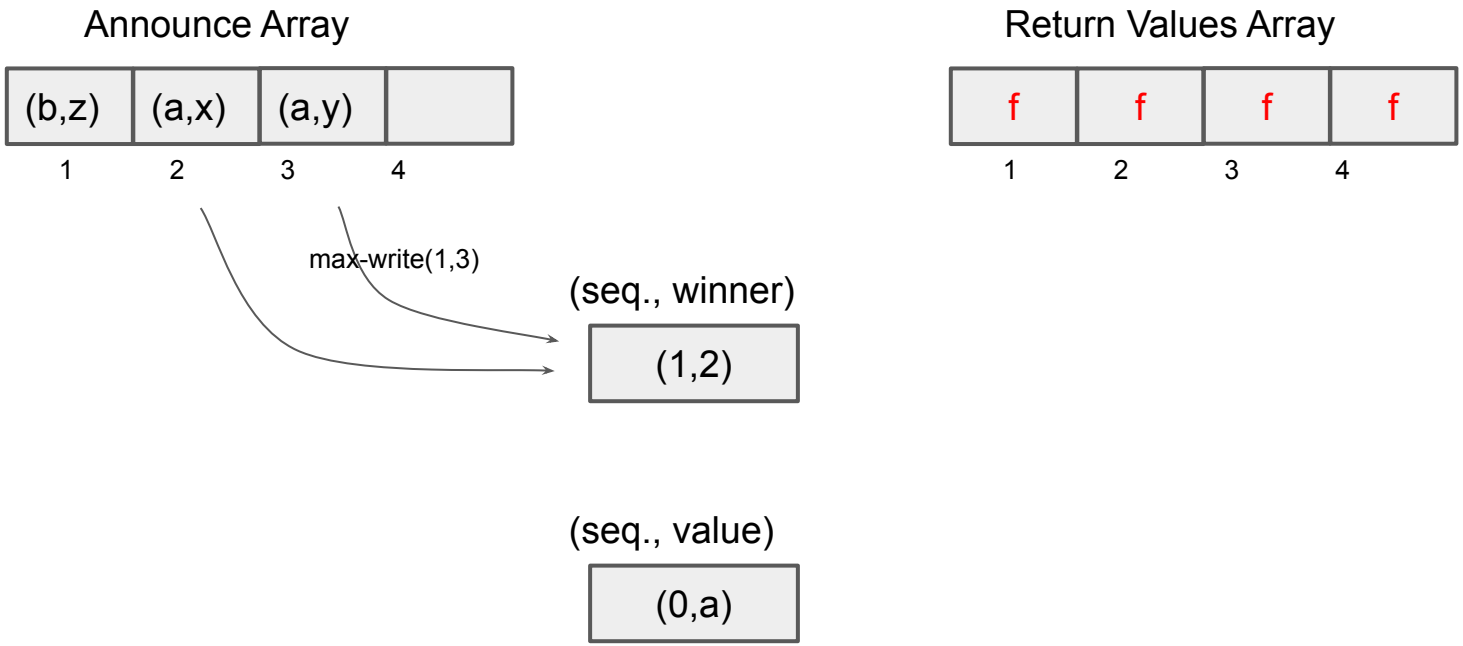
(seq., value)



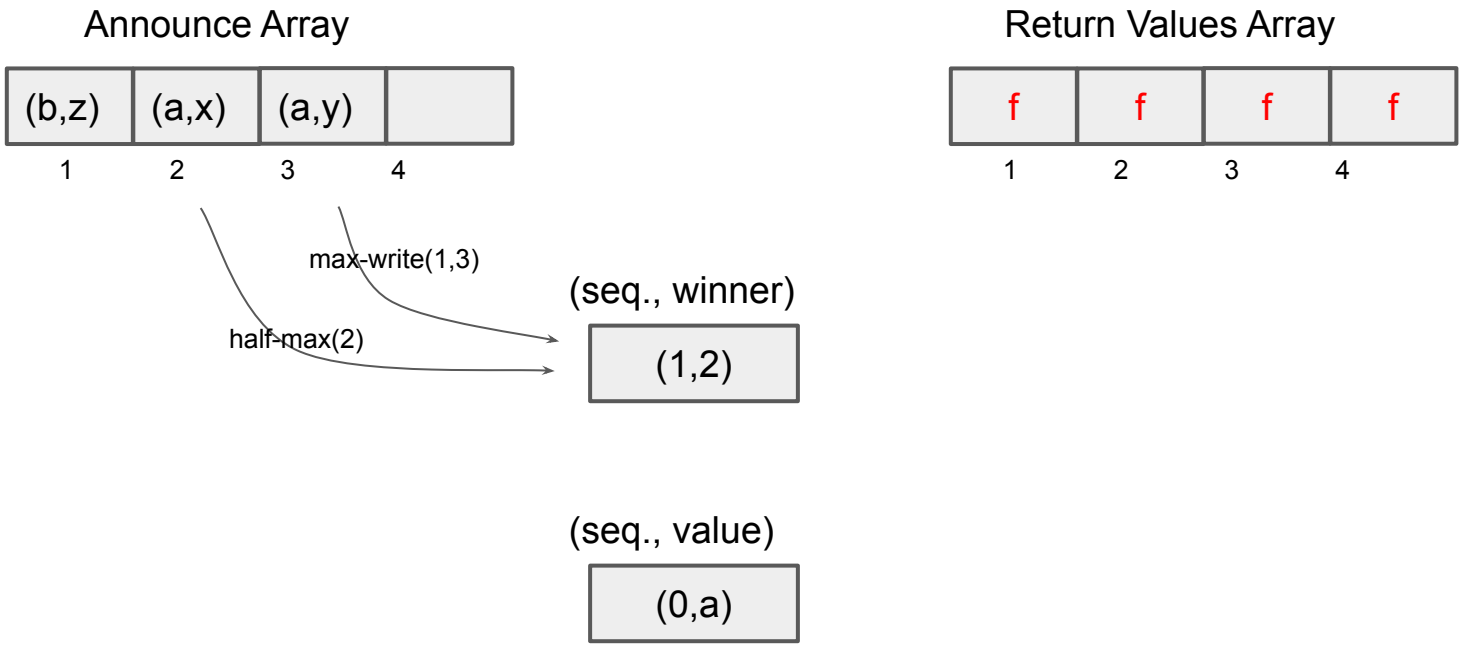
Simulation Idea



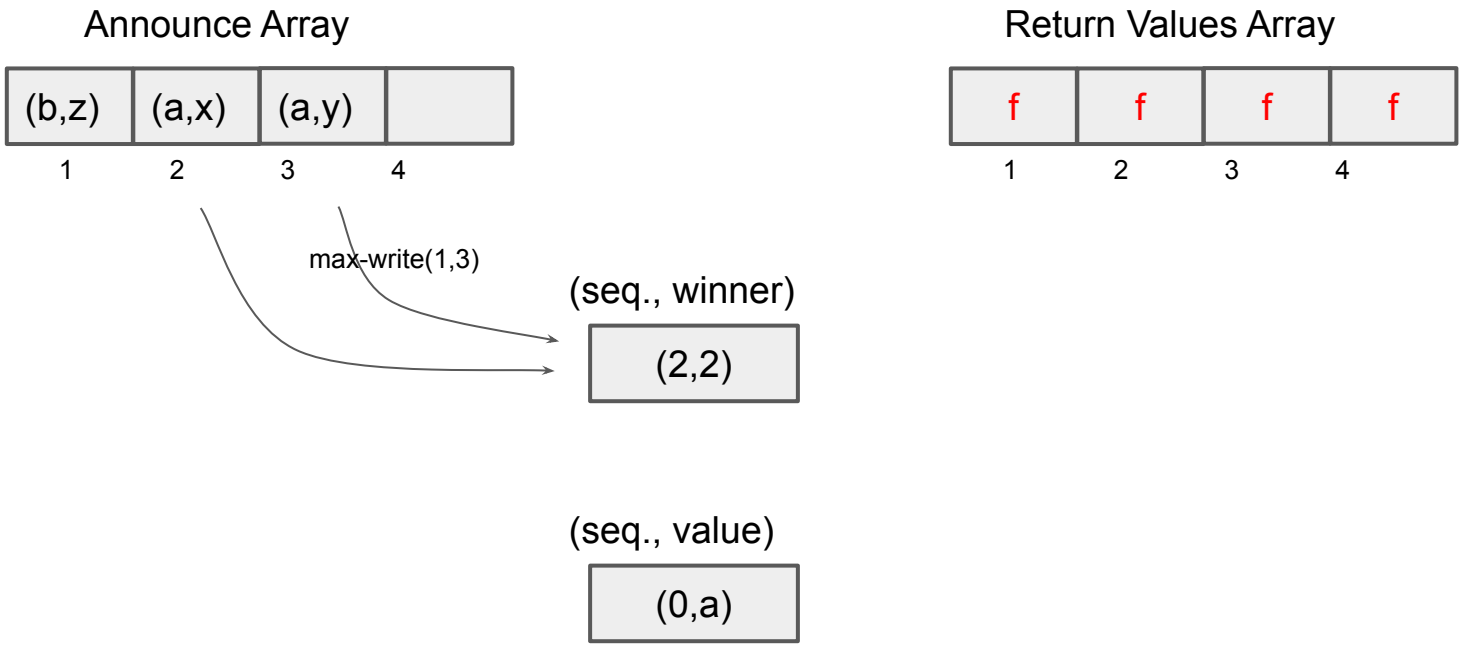
Simulation Idea



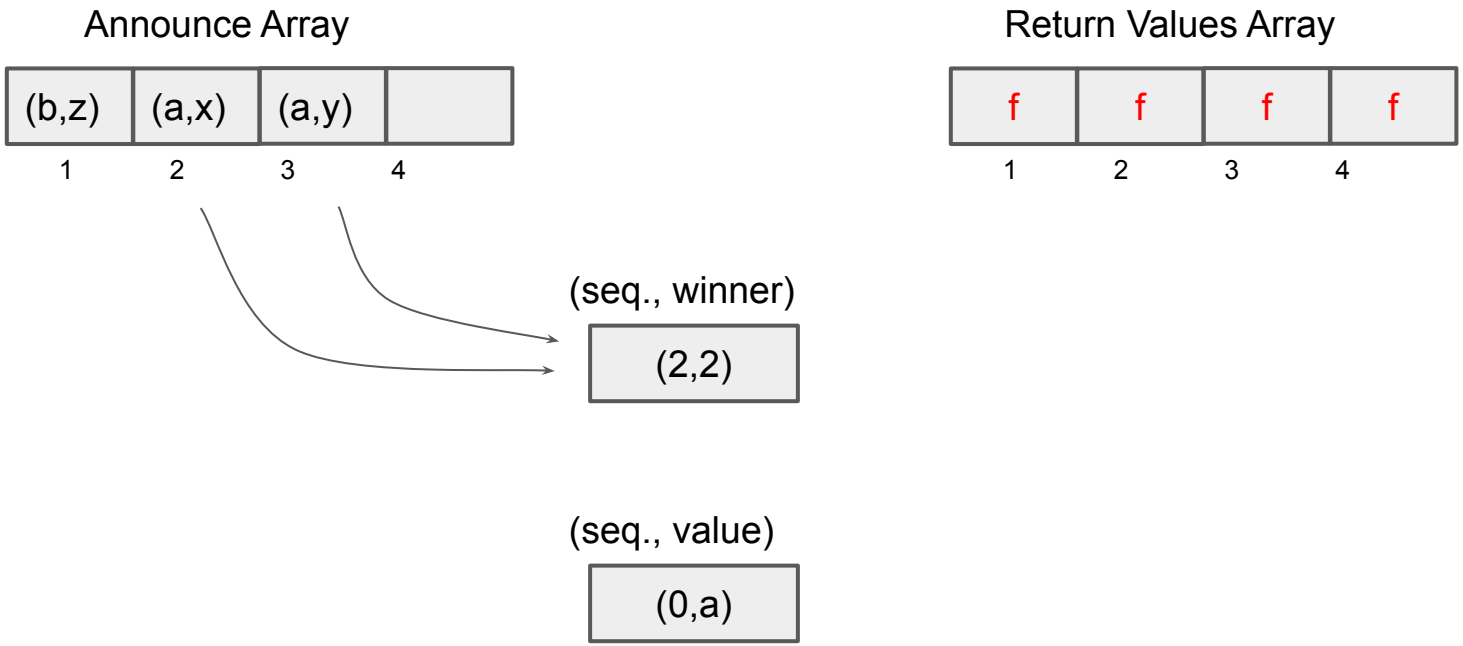
Simulation Idea



Simulation Idea

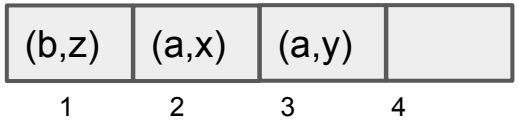


Simulation Idea

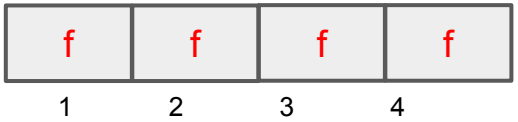


Simulation Idea

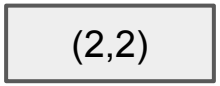
Announce Array



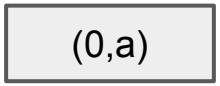
Return Values Array



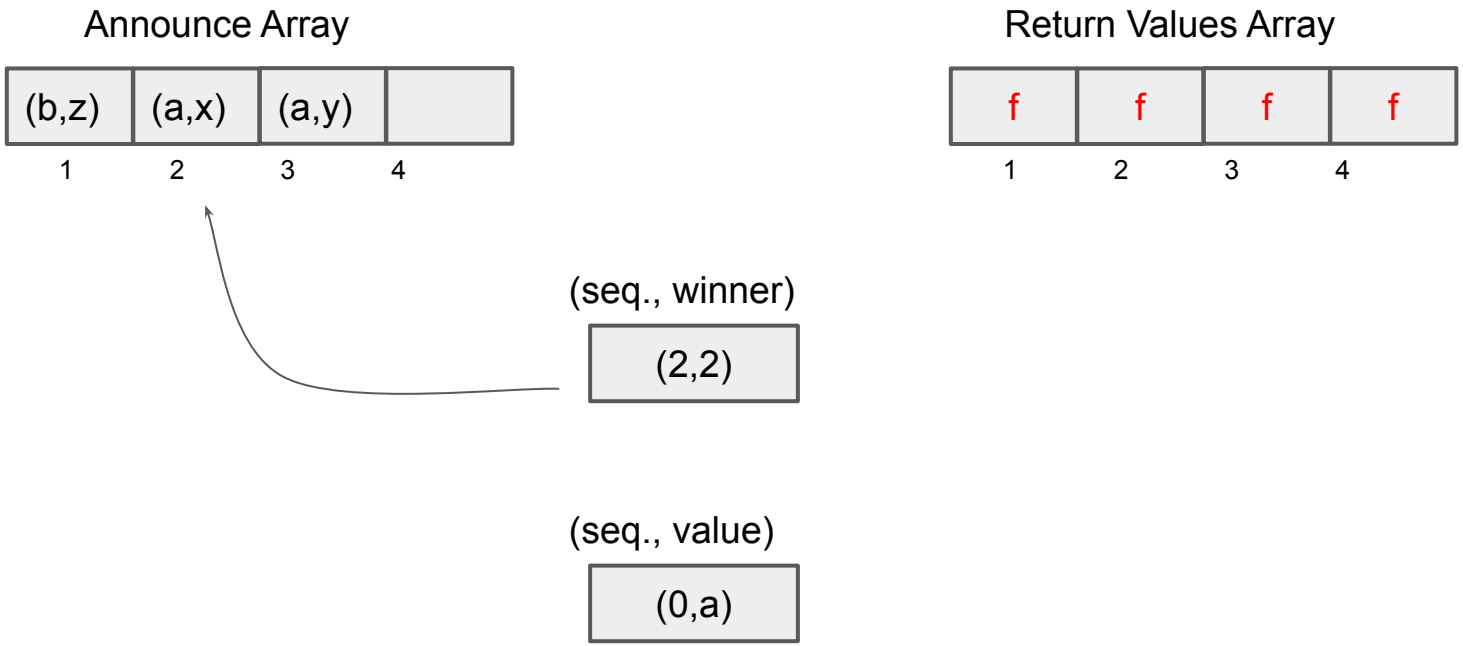
(seq., winner)



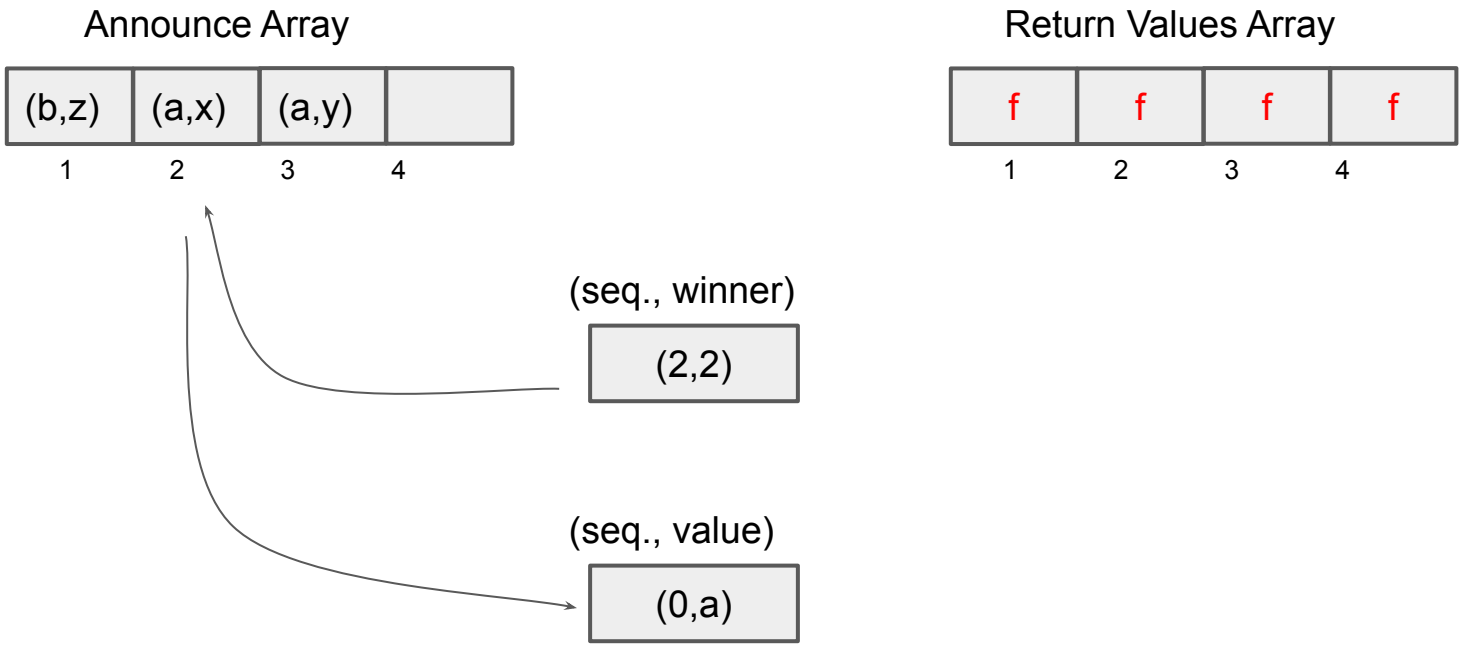
(seq., value)



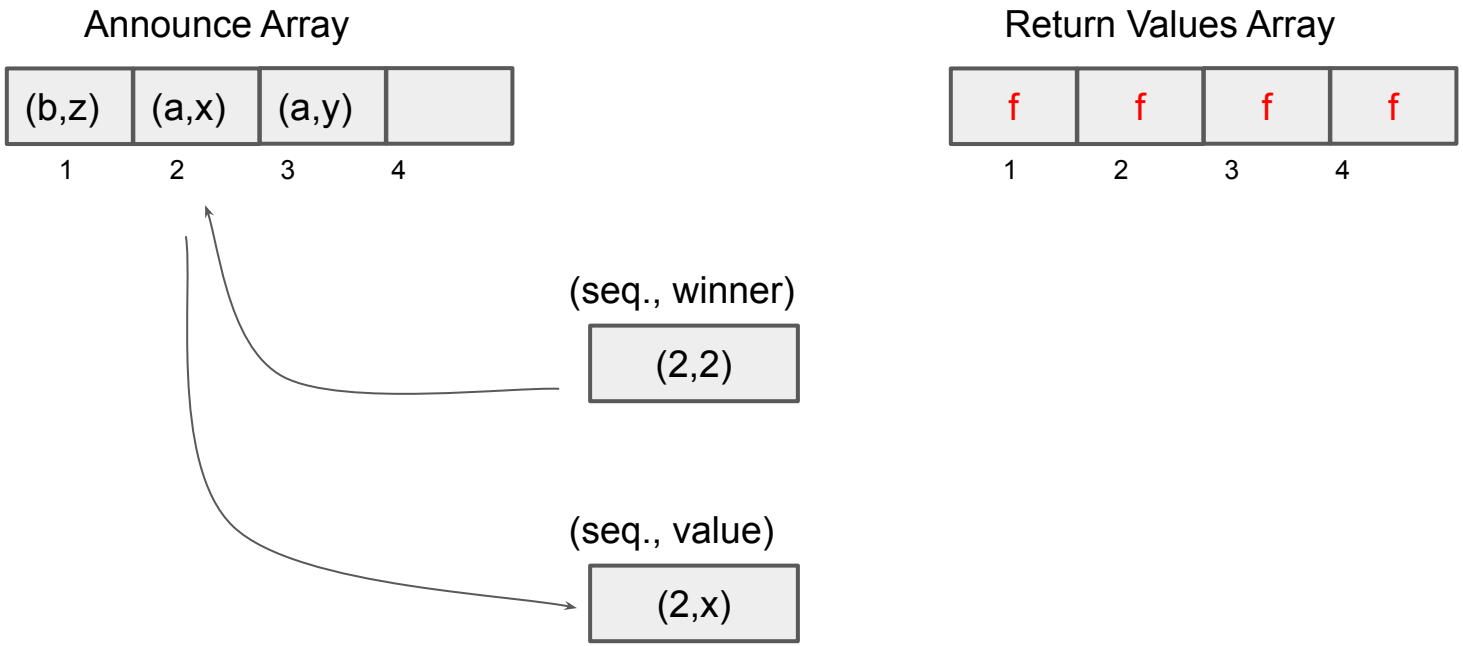
Simulation Idea



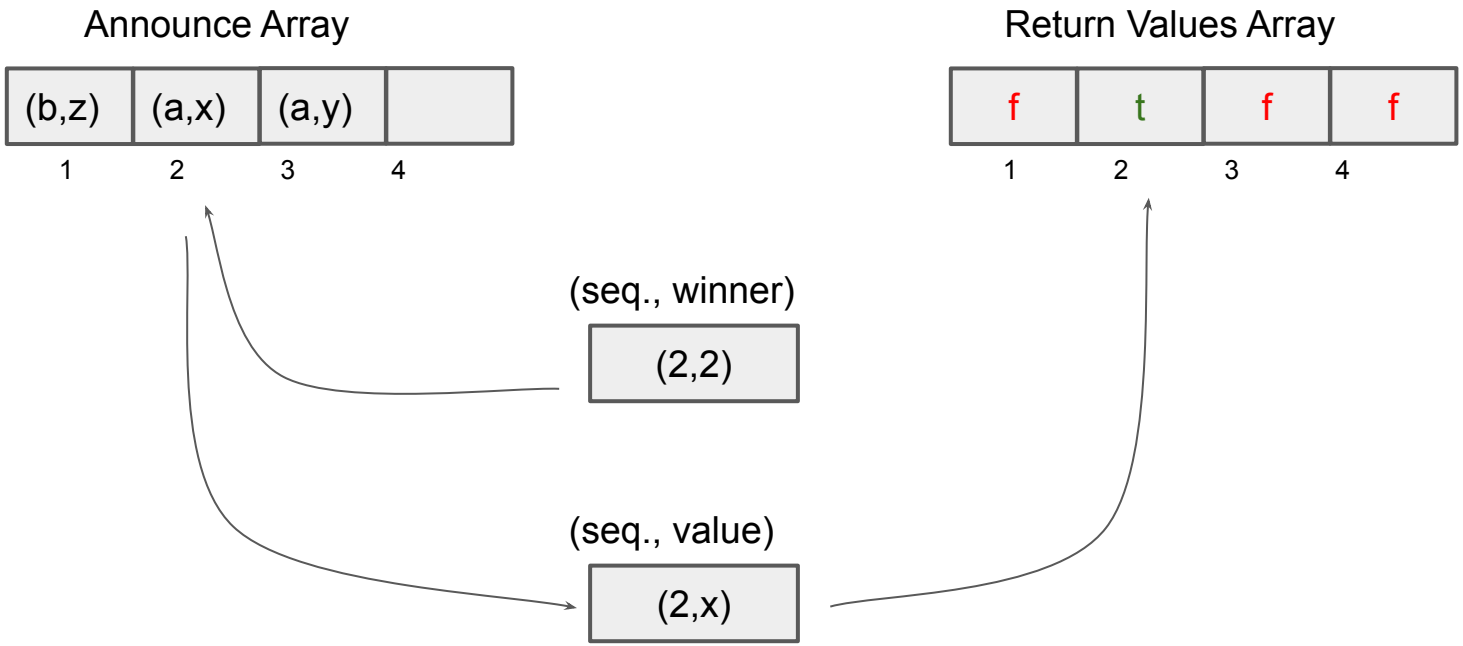
Simulation Idea



Simulation Idea



Simulation Idea



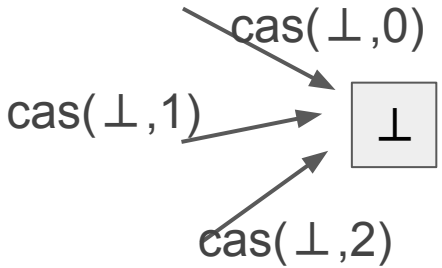
Consensus Numbers

Consensus: Each process has an input. The processes agree on an input of a single process and output that input.

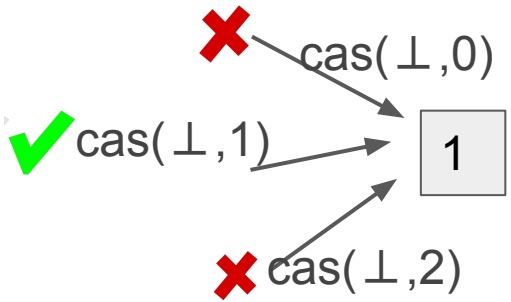
Consensus Numbers

Consensus Number is the maximum number of processes n for which the instruction can solve consensus.

C.N.(compare-and-swap) = ∞

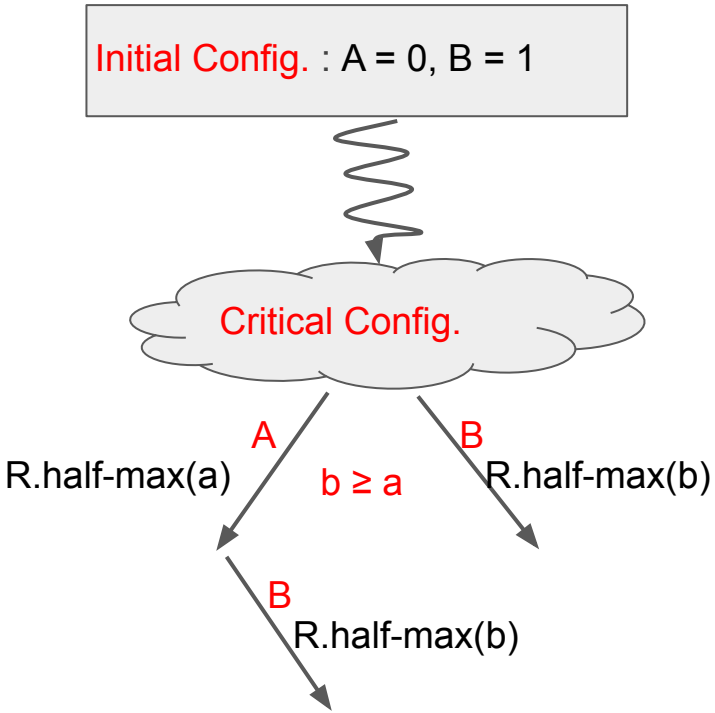


C.N.(compare-and-swap) = ∞

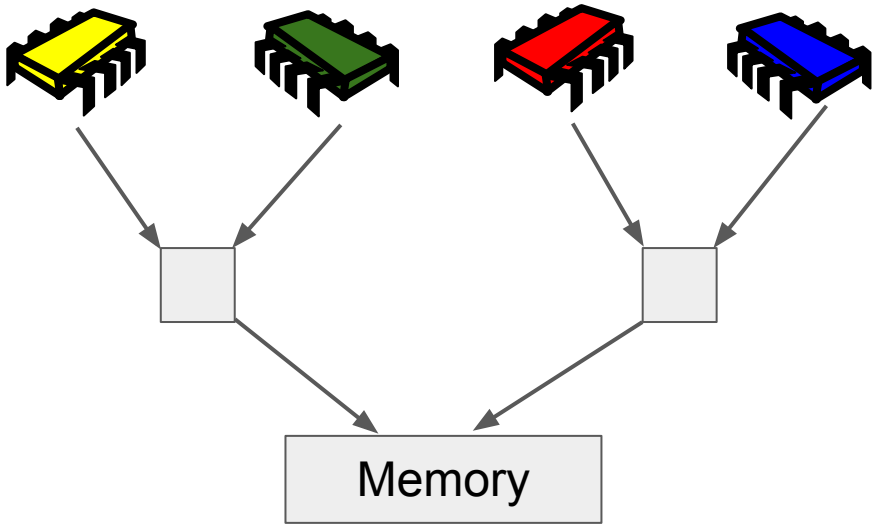


C.N.(half-max) = 1

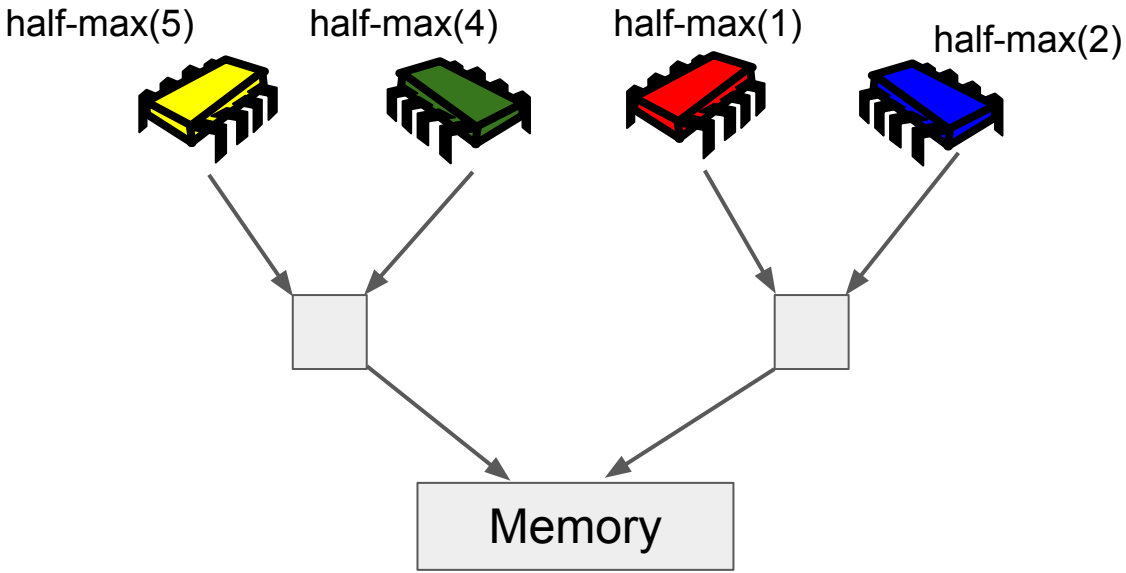
C.N.(half-max) = 1


Initial Config. : A = 0, B = 1
Critical Config.
A
R.half-max(a)
b ≥ a
B
R.half-max(b)
B
R.half-max(b)

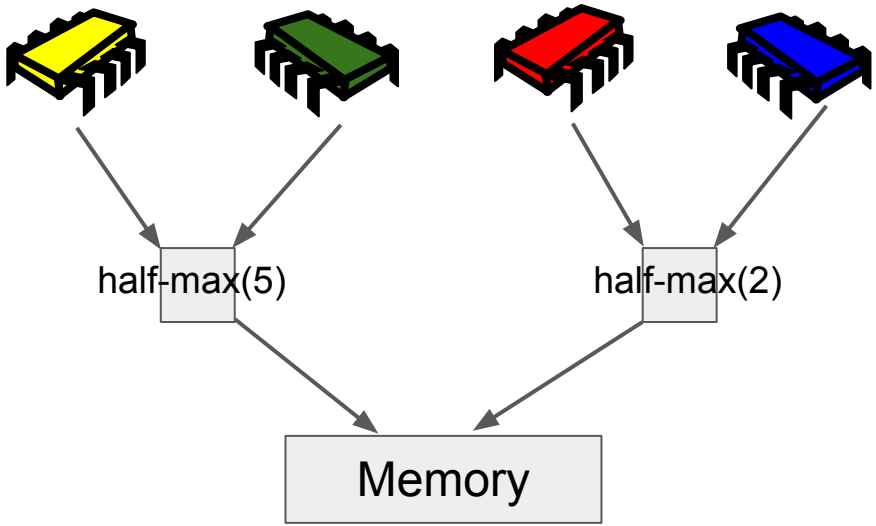
Are they better?



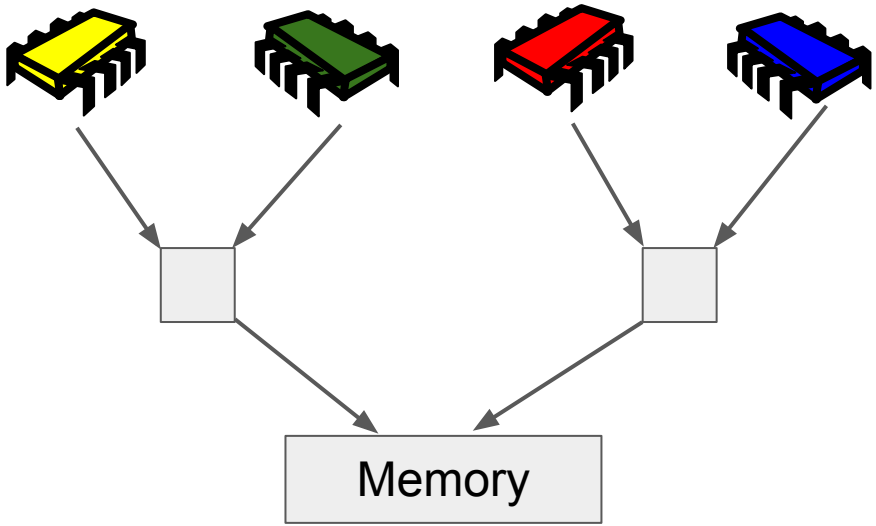
Are they better?



Are they better?

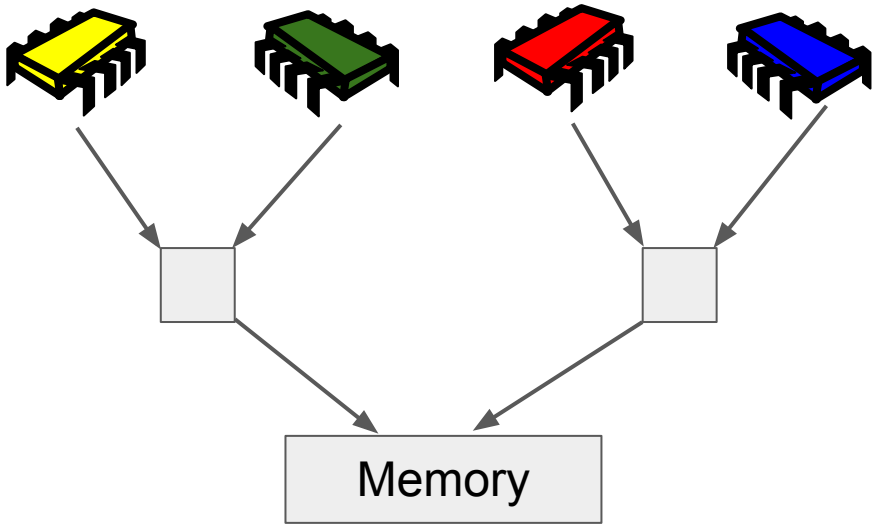


Are they better?

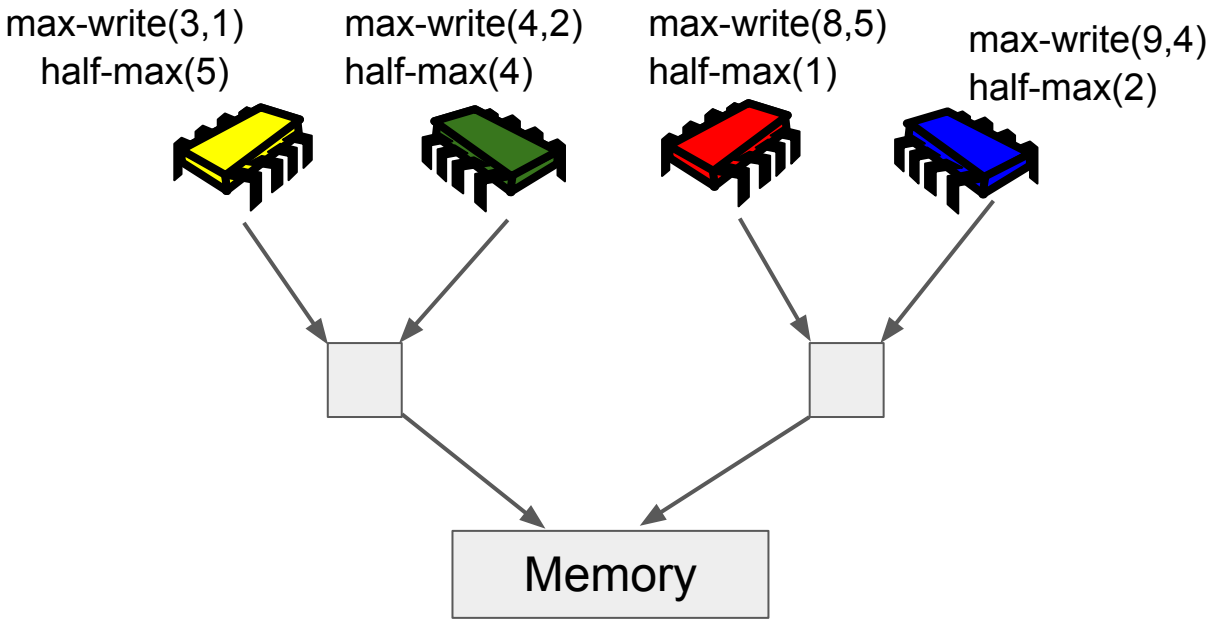


half-max(5)

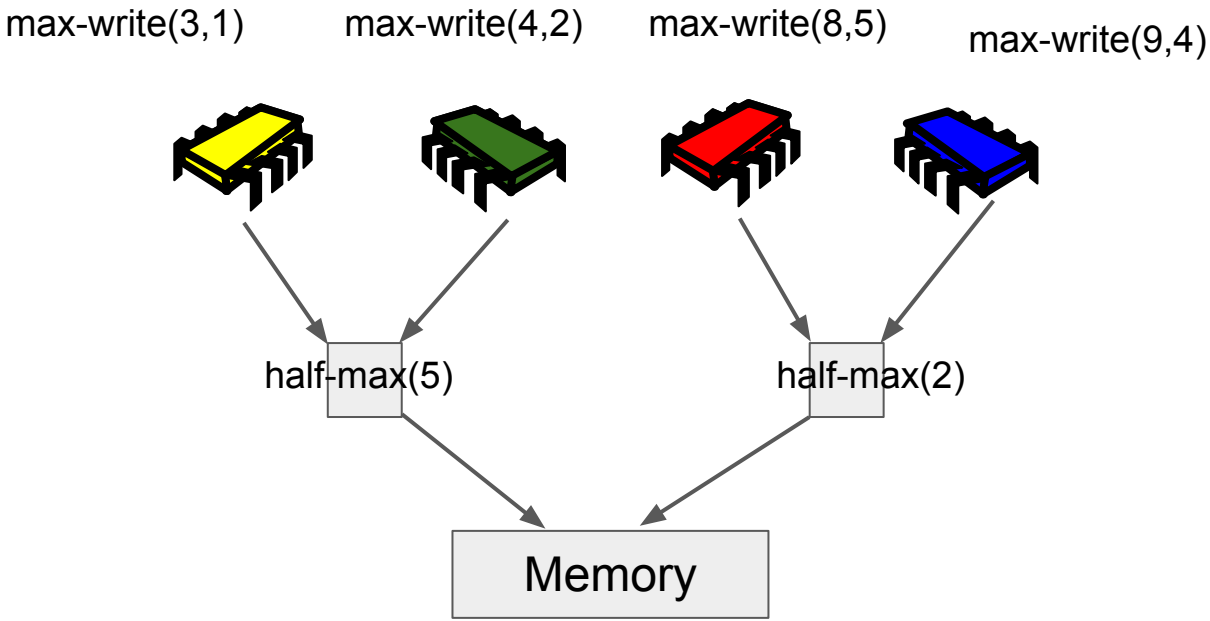
Are they better?



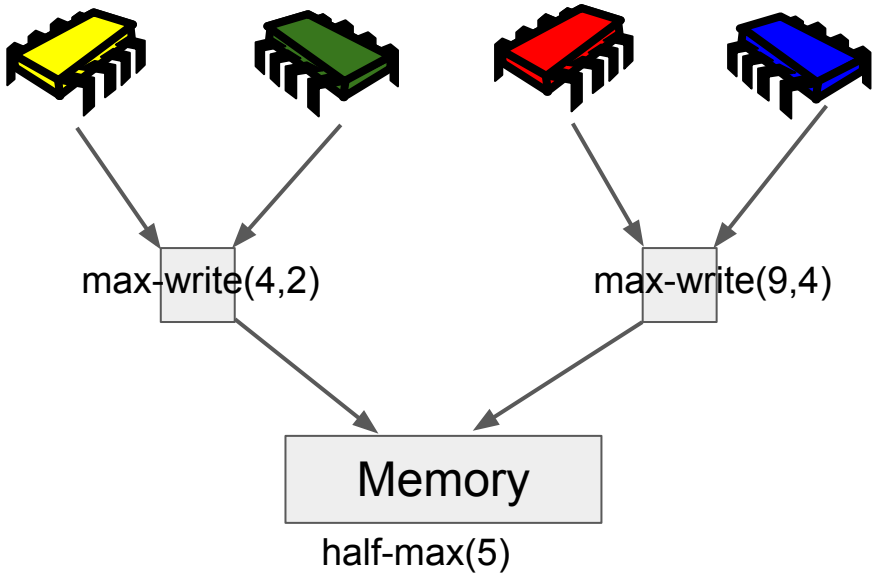
Are they better?



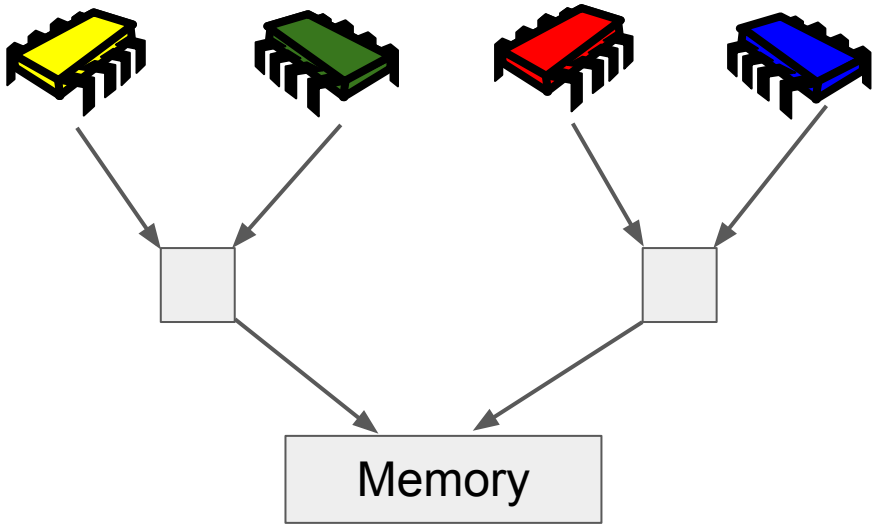
Are they better?



Are they better?

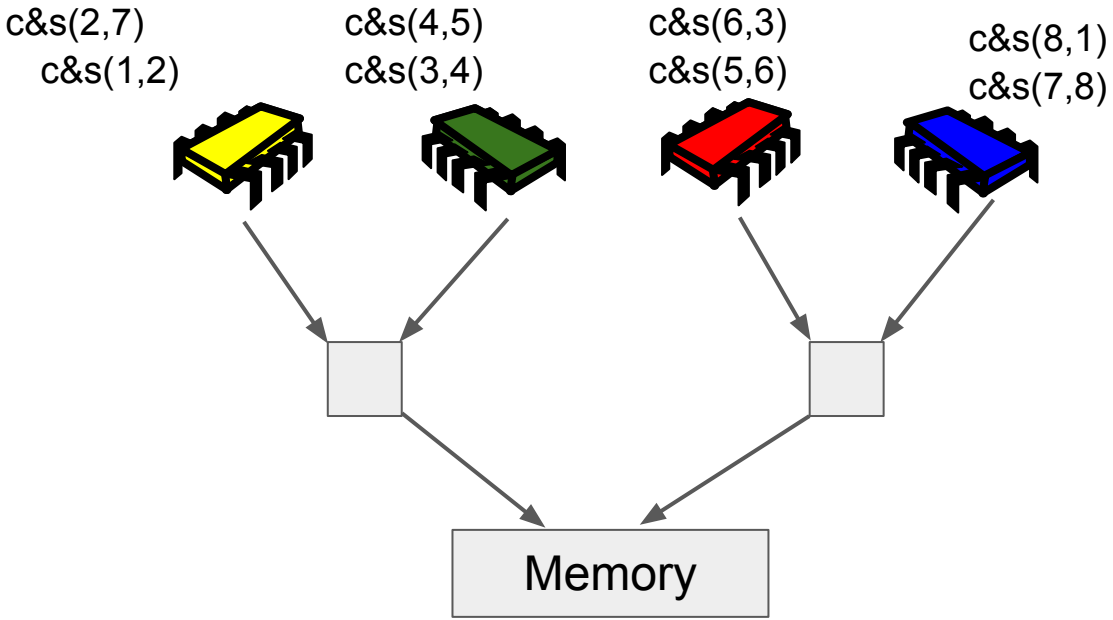


Are they better?

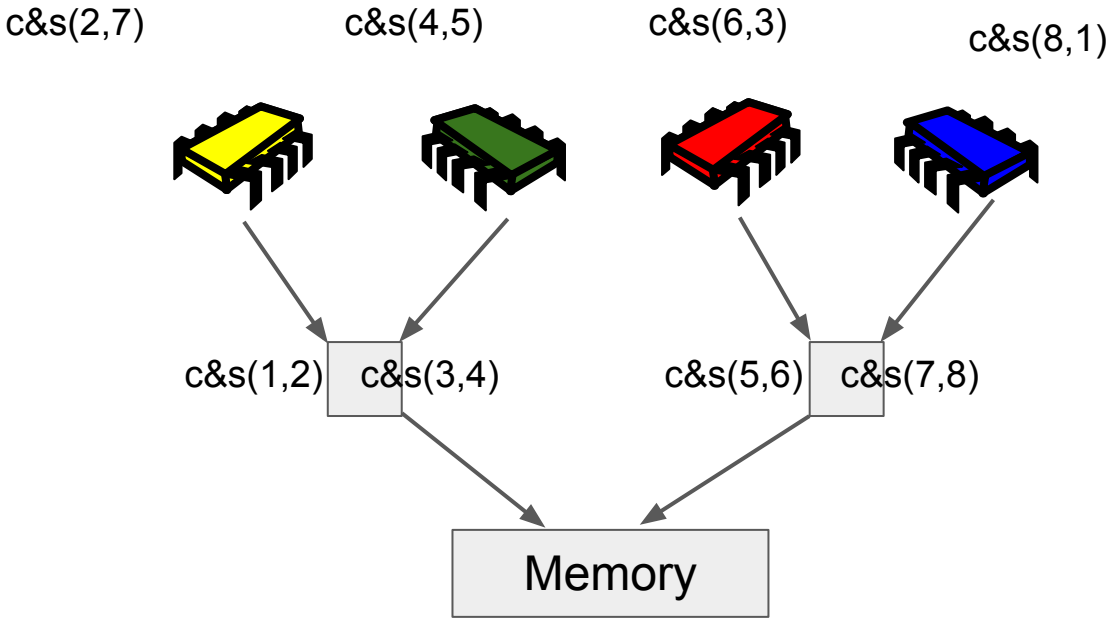


half-max(5)
max-write(9,4)

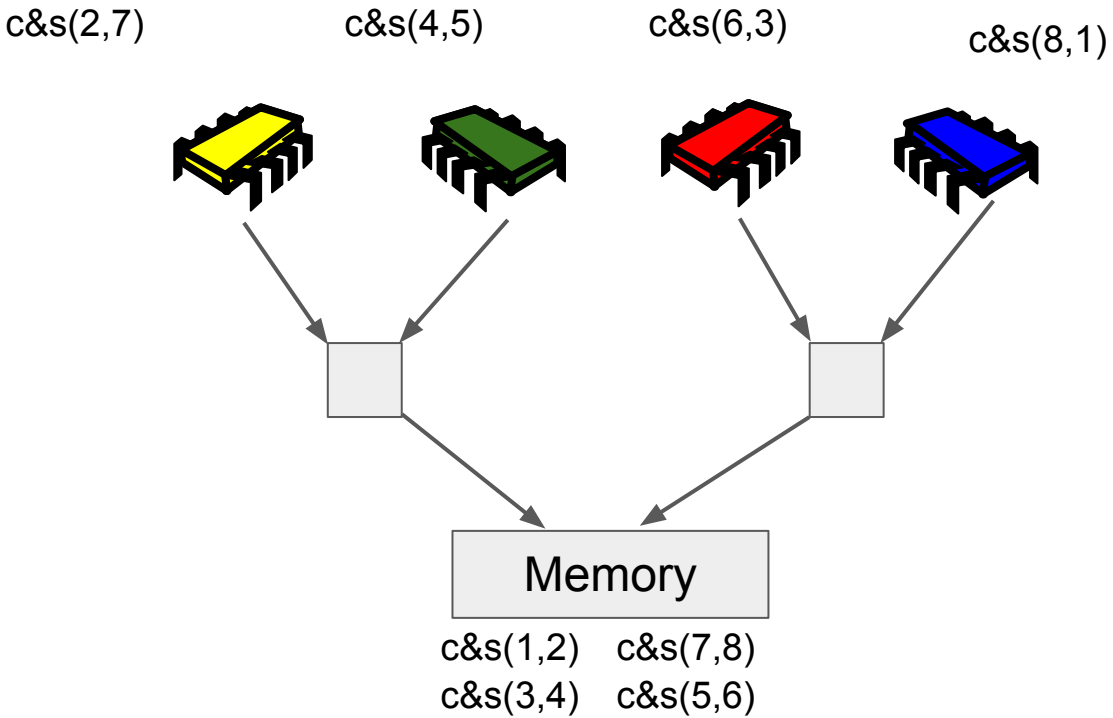
Are they better?



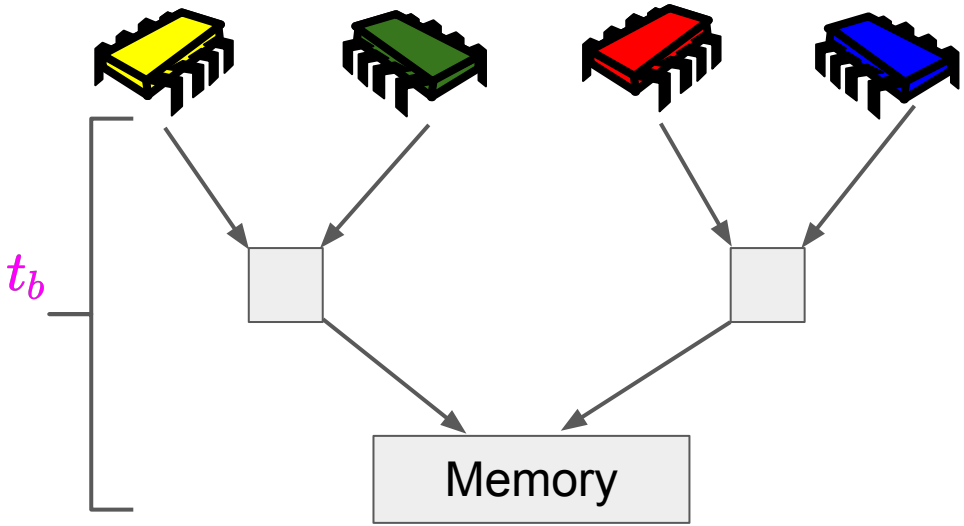
Are they better?



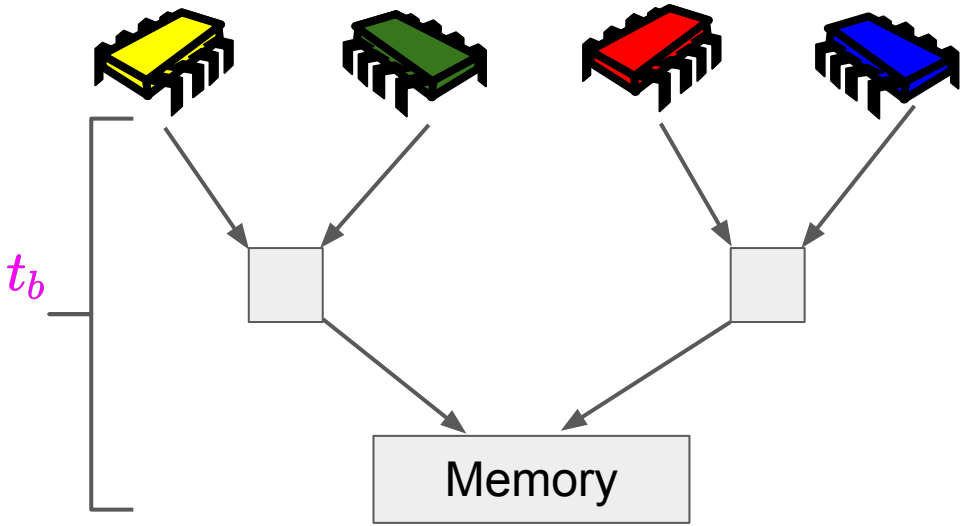
Are they better?



Throughput



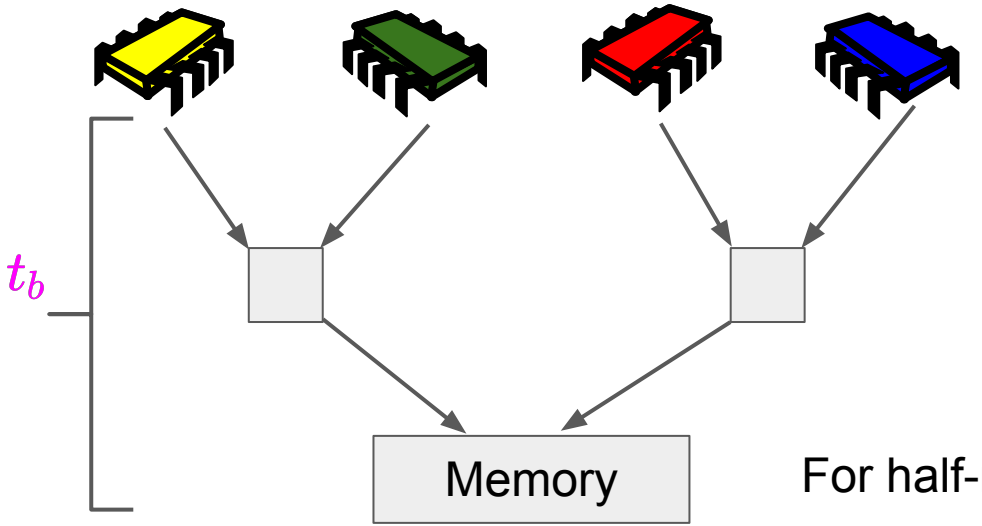
Throughput



For c&s

$$\frac{n}{t_b}$$

Throughput



For c&s

$$\frac{n}{t_b}$$

For half-max + max-write

$$\frac{n}{6+t_b/m} \approx O(n)$$

kpankaj@ethz.ch

https://disco.ethz.ch/members/kpankaj