Two Elementary Instructions make Compare-and-Swap
Compare-and-Swap
Shared Memory Model
Shared Memory Model

write(6)
read()

write(5)
read()
Atomic Registers

fetch-and-add(x)

f&a(1)  f&a(2)

3
Atomic Registers

fetch-and-add(x)

f&a(1)=3

f&a(2)

4
Atomic Registers

fetch-and-add(x)

f&a(1) = 3
f&a(2) = 4

6
Atomic Registers

fetch-and-add(x)

\[
\begin{align*}
\text{f&a(1)} &= 3 \\
\text{f&a(2)} &= 4
\end{align*}
\]

\[
6
\]

c&swap(x, y)

\[
\begin{align*}
\text{c&s(0,1)} \\
\text{c&s(0,2)}
\end{align*}
\]

\[
0
\]
Atomic Registers

fetch-and-add(x)

f&a(1)=3  
f&a(2)=4  
6

compare-and-swap(x,y)

c&s(0,1)  
c&s(0,2)  
2
Atomic Registers

fetch-and-add(x)

f&a(1)=3  \rightarrow  f&a(2)=4

\[ \text{6} \]

compare-and-swap(x,y)

c&s(0,1)

\[ \times \]

\[ \text{2} \]

c&s(0,2)

\[ \checkmark \]
Best Atomic Registers?

Consensus Numbers [Herlihy 1991]

\[ \text{compare-and-swap} = \infty \]
\[ \text{...} \]
\[ \text{n-register assignment} = 2n-2 \]
\[ \text{...} \]
\[ \text{fetch-and-add} = 2 \]
\[ \text{read/write} = 1 \]
Best Atomic Registers?

[Ruppert 1997]

\[
\begin{align*}
\text{compare-and-swap} &= \infty \\
... \\
\text{n-register assignment} &= 2^{n-2} \\
... \\
\text{fetch-and-add} &= 2 \\
\text{read/write} &= 1
\end{align*}
\]
Best Atomic Registers?

[Ellen et al. 2016]

**Assumption**

- fetch-and-add(x)
- fetch-and-multiply(y)

**Reality**

- fetch-and-add(x)
- fetch-and-multiply(y)
Best Atomic Registers?

[Ellen et al. 2016]

Assumption

\[\text{C.N.}(\text{fetch-and-add, fetch-and-multiply}) = 2\]

Reality
Best Atomic Registers?

[Ellen et al. 2016]

Assumption

![Diagram of assumptions with fetch-and-add and fetch-and-multiply operations, showing 2 contexts.]

Reality

![Diagram of reality with fetch-and-add and fetch-and-multiply operations, showing infinite contexts.]

C.N.(fetch-and-add, fetch-and-multiply)
Best Atomic Registers?

- Queues
- Stacks
- ...
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

half-max(3) → (3,5)

half-max(1) → (2,5)

(2,5) → (3,5)

(2,5) → (2,5)
This Work: half-max, max-write
This Work: half-max, max-write

\[ c&s(5,6) \rightarrow 6 \quad \text{✓} \]

\[ 5 \]

\[ c&s(3,4) \rightarrow 5 \quad \text{✗} \]
Simulation Idea

Announce Array

1 2 3 4

Return Values Array

1 2 3 4

(seq., winner)

(seq., value)
Simulation Idea

Announce Array

Read/Write

Return Values Array

(seq., winner)

half-max/max-write

(seq., value)

max-write

1             2             3             4
1             2             3             4
Simulation Idea

Announce Array

<table>
<thead>
<tr>
<th>(b,z)</th>
<th>(a,x)</th>
<th>(a,y)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Return Values Array

<table>
<thead>
<tr>
<th>f</th>
<th>f</th>
<th>f</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

(seq., winner)

(0, ⊥)

(seq., value)

(0,a)
Simulation Idea

Announce Array

<table>
<thead>
<tr>
<th></th>
<th>(b,z)</th>
<th>(a,x)</th>
<th>(a,y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Return Values Array

<table>
<thead>
<tr>
<th></th>
<th>f</th>
<th>f</th>
<th>f</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(seq., winner)

(0, ⊥)

(seq., value)

(0, a)

false
Simulation Idea

Announce Array

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(b, z)</td>
<td>(a, x)</td>
<td>(a, y)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Return Values Array

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

(seq., winner)

(0, ⊥)

(seq., value)

(0, a)
Simulation Idea

Announce Array

(\(b, z\)) (\(a, x\)) (\(a, y\))

1  2  3  4

Return Values Array

\(f\) \(f\) \(f\) \(f\)

1  2  3  4

max-write(1,3) (seq., winner) (0, \(\perp\))

max-write(1,2) (seq., value) (0, a)
Simulation Idea

Announce Array

max-write(1,3)

Return Values Array

(1,2)

(b,z) (a,x) (a,y) 
1 2 3 4

(1,2) (seq., winner)

(seq., value)

(0,a)

(1,2) (seq., winner)

(seq., value)

(0,a)
Simulation Idea

Announce Array

<table>
<thead>
<tr>
<th></th>
<th>(b,z)</th>
<th>(a,x)</th>
<th>(a,y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Return Values Array

<table>
<thead>
<tr>
<th></th>
<th>f</th>
<th>f</th>
<th>f</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- max-write(1,3)
- half-max(2)

(seq., winner)

(1,2)

(seq., value)

(0,a)
Simulation Idea

Announce Array

Return Values Array

max-write(1,3)

(seq., winner)

(seq., value)

(2,2)

(0,a)
Simulation Idea

Announce Array

<table>
<thead>
<tr>
<th>(b,z)</th>
<th>(a,x)</th>
<th>(a,y)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Return Values Array

<table>
<thead>
<tr>
<th>f</th>
<th>f</th>
<th>f</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

(seq., winner)

(2,2)

(seq., value)

(0,a)
Simulation Idea

Announce Array

<table>
<thead>
<tr>
<th>(b,z)</th>
<th>(a,x)</th>
<th>(a,y)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Return Values Array

<table>
<thead>
<tr>
<th>f</th>
<th>f</th>
<th>f</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

(seq., winner)

(2,2)

(seq., value)

(0,a)
Simulation Idea

Announce Array

<table>
<thead>
<tr>
<th>(b,z)</th>
<th>(a,x)</th>
<th>(a,y)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

(2,2)

(seq., winner)

Return Values Array

<table>
<thead>
<tr>
<th>f</th>
<th>f</th>
<th>f</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

(0,a)

(seq., value)
Simulation Idea

Announce Array

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(b,z)</td>
<td>(a,x)</td>
<td>(a,y)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Return Values Array

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

(seq., winner)
(2,2)

(seq., value)
(0,a)
Simulation Idea

Announce Array

<table>
<thead>
<tr>
<th>(b,z)</th>
<th>(a,x)</th>
<th>(a,y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Return Values Array

| f     | f     | f     | f     |
|-------|-------|-------|
| 1     | 2     | 3     | 4     |

(seq., winner)

(2,2)

(seq., value)

(2,x)
Simulation Idea

Announce Array

<table>
<thead>
<tr>
<th>(b,z)</th>
<th>(a,x)</th>
<th>(a,y)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Return Values Array

<table>
<thead>
<tr>
<th>f</th>
<th>t</th>
<th>f</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

(seq., winner)

(2,2)

(seq., value)

(2,x)
Consensus: Each process has an input. The processes agree on an input of a single process and output that input.
Consensus Number is the maximum number of processes $n$ for which the instruction can solve consensus.
C.N.\(\text{\textup{\textit{compare-and-swap}}}\) = \infty
C.N. (compare-and-swap) = ∞
C.N.(half-max) = 1
C.N. (half-max) = 1

Initial Config. : A = 0, B = 1

Critical Config.

R. half-max(a)  b ≥ a  R. half-max(b)

R. half-max(b)
Are they better?

Memory
Are they better?

half-max(5)  half-max(4)  half-max(1)  half-max(2)

Memory
Are they better?

half-max(5)  half-max(2)

Memory
Are they better?
Are they better?
Are they better?

max-write(3,1)  
half-max(5)  

max-write(4,2)  
half-max(4)  

max-write(8,5)  
half-max(1)  

max-write(9,4)  
half-max(2)  

Memory
Are they better?

max-write(3,1)  max-write(4,2)  max-write(8,5)  max-write(9,4)

half-max(5)  half-max(2)

Memory
Are they better?

```
half-max(5)
max-write(4,2)
max-write(9,4)
```
Are they better?

Memory

half-max(5)
max-write(9,4)
Are they better?

c&s(1,2)  c&s(3,4)  c&s(5,6)  c&s(7,8)
c&s(2,7)  c&s(4,5)  c&s(6,3)  c&s(8,1)
Are they better?

c&s(2,7)  c&s(4,5)  c&s(6,3)  c&s(8,1)

Memory
Are they better?

c&s(2,7)  c&s(4,5)  c&s(6,3)  c&s(8,1)

Memory

c&s(1,2)  c&s(3,4)  c&s(7,8)  c&s(5,6)
Throughput

For c&s

\[ \frac{n}{t_b} \]
Throughput

For c&s
\[ \frac{n}{t_b} \]

For half-max + max-write
\[ \frac{n}{6 + \frac{t_b}{m}} \approx O(n) \]