Local-Spin Mutex
without Strong Primitives

Local-Spin Tournament-Tree Mutex

$O(\log n)$ RMR complexity for both DSM and CC systems (this is optimal)
$O(n \log n)$ registers

Key is to find the right 2-process mutex
Local-Spin 2-Process Mutex: 1st Try

Shared variables:
Want[0], Want[1]: initially ⊥
Spin[0], Spin[1]: initially ⊥

acquire-lock(side)
Want[side] = 1    // announce
Spin[side] = 0
opponent = Want[1-side]  // read other side
if ( opponent <> ⊥ )
    wait until ( Spin[side] <> 0 )  // spin

release-lock(side)
Want[side] = ⊥    // cancel announcement
Spin[1-side] = 1 // release other

Ensures mutual exclusion
But may deadlock
Local-Spin 2-Process Mutex: Avoid Deadlock

Shared variables:
- Tie, Want[0], Want[1]: initially ⊥
- Spin[0], Spin[1]: initially ⊥

acquire-lock(side)
- Want[side] = 1 // announce
- Tie = i // tie breaker
- Spin[side] = 0
- opponent = Want[1-side] // read other side
- if ( opponent <> ⊥ ) and ( Tie == i )
  - if ( Spin[1-side] == 0 ) Spin[1-side] = 1
  - wait until ( Spin[side] <> 0 ) // spin
  - if ( Tie == i ) wait until ( Spin[side] > 1 )

release-lock(side)
- Want[side] = ⊥ // cancel announcement
- if ( Tie <> i ) Spin[1-side] = 2 // release other

Is this local spinning in DSM?

236755 (2014) Mutual exclusion
Local-Spin 2-Process Mutex

Shared variables:
- Tie, Want[0], Want[1]: initially ⊥
- Spin[0,…,n-1]: initially ⊥

```plaintext
acquire-lock(side)
Want[side] = i  // announce your identity
Tie = i  // tie breaker
Spin[i] = 0
opponent = Want[1-side]  // who's competing
if ( opponent <> ⊥ ) and ( Tie == i )
  if ( Spin[opponent] == 0 ) Spin[opponent] = 1
  wait until ( Spin[i] <> 0 )
  if ( Tie == i ) wait until ( Spin[i] > 1 )
```

```plaintext
release-lock(side)
Want[side] = nil  // who's competing
opponent = Tie  // who's competing
if ( opponent <> i ) Spin[opponent] = 2
```

Example (for processes 3 and 7)

```
Want[0] = 3  Want[1] = 7
Tie = 3
Spin[3] = 0  Tie = 7
opponent = 7  Spin[7] = 0
opponent <> ⊥ and Tie <> 3  opponent = 3
CRITICAL  CRITICAL
opponent <> ⊥ and Tie == 7
CRITICAL  WAIT until Spin[7] <> 0
CRITICAL  WAIT
CRITICAL  WAIT
CRITICAL
```
Local-Spin 2-Process Mutex

Shared variables:
- Tie, Want[0], Want[1]: initially 1
- Spin[0,...,n-1]: initially 1

acquire-lock(side)
- Want[side] = i
- Tie = i
- Spin[i] = 0
- opponent = Want[1-side]
  - if (opponent <> nil) and (Tie == i)
    - if (Spin[opponent] == 0) Spin[opponent] = 1
    - wait until (Spin[i] <> 0)
    - if (Tie == i) wait until (Spin[i] > 1)

release-lock(side)
- Want[side] = nil
- opponent = Tie
  - if (opponent <> i) Spin[opponent] = 2

An array for each level
O(n log n) total

©Hagit Attiya