

Shared memory Synchronization

mutual exclusion

- how much in hardware?
speed vs. flexibility

sol'n is typically a form of atomic read-modify-write

- components of sync. event
 - acquire event - permission to proceed
 - wait alg
 - release

waiting algs

- block

deschedule to OS

- busy-wait

spin wait on variable

- hybrid

Cray XMP - lock registers

Blue Gene line from IBM - low latency #LW
read-modify-write inst. are common

simple software lock

```
lock:  ld    r0, loc          /* copy loc to reg #0 */
        cmp  r0, #0         /* if not 0, try again */
        bnz  lock          /* mark as locked */
        st  r0, loc, #1
        ret

unlock: st  loc, #0        /* mark unlocked */
        ret
```

atomic exchange inst.

specifies loc. & reg.

value in loc \rightarrow reg.

another val (maybe func. of value read) \rightarrow loc.

simple example test & set

(value in loc. \rightarrow reg.)

(const 1 \rightarrow loc.)

success if value loaded to reg is 0

test & set lock

lock: `lds reg, loc.`

`bnz lock`

// if not 0, try again

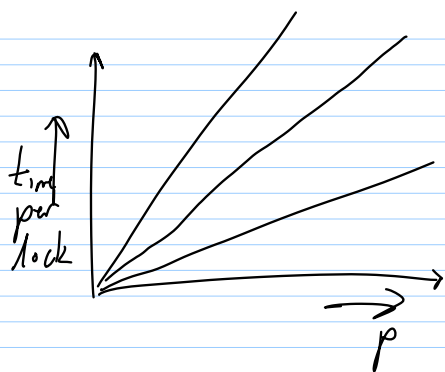
`ret`

unlock `st loc, #0`

`ret`

other atomic inst.

swap
fetch & op
compare & swap



loop
lock
delay (d)
unlock

reduce freq. of issuing tbs while waiting
busy-wait with reads (loads)

test and tbs

dimensions of performance

low uncontested latency
traffic with contention

storage
fairness

