Binary lock

Lock (item)
Unlock (item)

Lock (item):
Start: if (Lock(item) == 0) {
    // item is unlocked
    Lock(item) = 1
} else {
    wait (until Lock(item) == 0) // process goes to sleep
    Lock manages wakes up the transaction when the lock is released.
} goto start

Unlock (item):
[Lock(item) = 0]
[if any transaction is waiting for the item, wake up the transaction]
Lost update problem.

For locking a DB-item:
- read the lock state
- if it is unlocked, set the lock

This entire operation must be executed in one cycle.

New opcode is introduced in the ISA
EXCH (exchanges two values) in one cycle
Swaps (lock state and user lock request (I))
Locks

Binary: 0 unlocked, 1 locked

S: Shared: 0, 1, 2, 3, ... no. of processes share that item
(Read) unlocked

E: Exclusive: 0, 1 (Single process)
(write) occupied

Intention locks

Concurrence of locks

- Server level
- Database level
  - Object (table) level
  - Row level

IS (Intention Shared) on Company DB → IS → P3
- Employee Table → Shared

P5 write ?IS

P6 IS - Company DB
IS - Department Table
S - row

P8 write Department
Company DB - Department Table - row
?IS
?S wait unlocked
Read & Write Locks

**State:**
- Unlocked (RL=0, WL=0)
- Read locked (RL=int, WL=0)
- Write locked (RL=0, WL=1)

**Read lock(item):**

```
start:  if (Lock(item) == 'Unlocked')
{
  Lock(item) = 'Readlocked'
  no_of_readers = 1
}
else if (Lock(item) == 'readlocked')
{
  no_of_readers++
}
else { wait (until Lock(item) == 'Unlocked')
      and
      Lock manager wakes up the transaction
      goto start
}
```

**Write lock(item):**

```
start:  if (Lock(item) == 'Unlocked')
{
  Lock(item) = 'Write locked'
}
else { wait (until Lock(item) == 'Unlocked')
      and
      Lock manager wakes up the transaction
      goto start
}
```

**Unlock (item):**

```
if (Lock(item) == 'Write locked')
{
  Lock(item) = 'Unlocked'
  Wake up Lock Manager to pickup a waiting Transaction
}
else { no_of_readers --
      if (No_of_readers == 0)
          Wake up Lock
      }
```