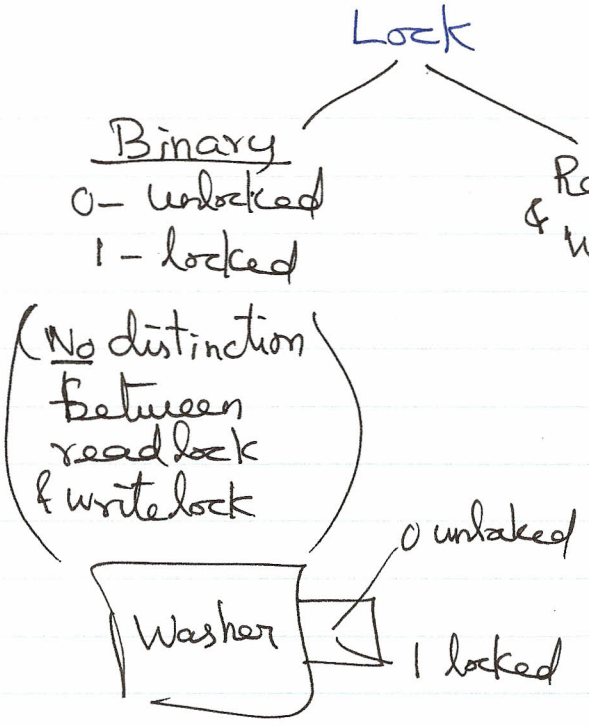
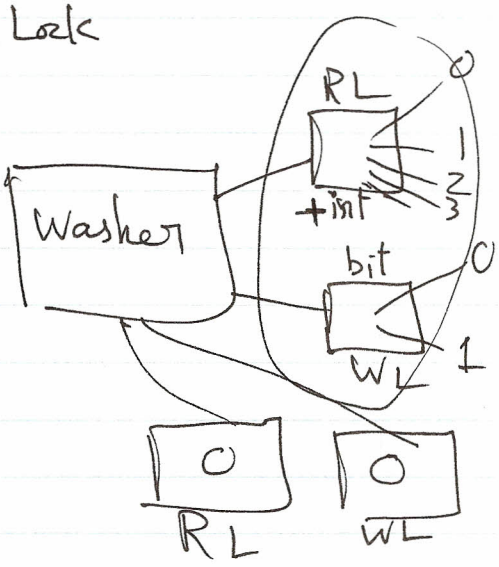
Lock





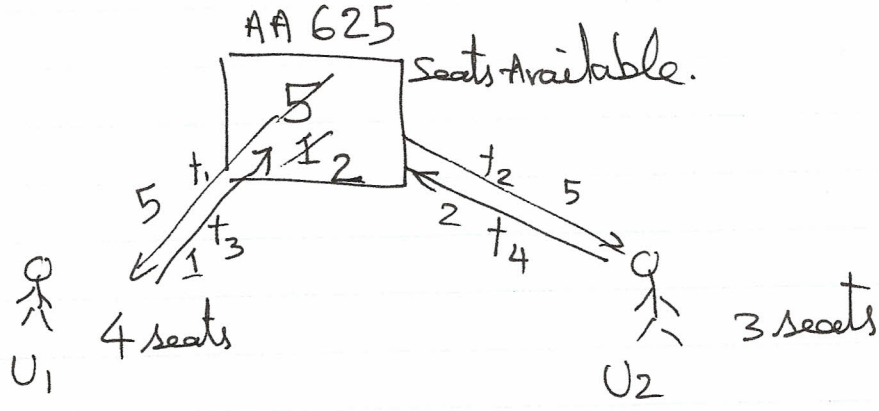
Binary Lock

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
       }
```

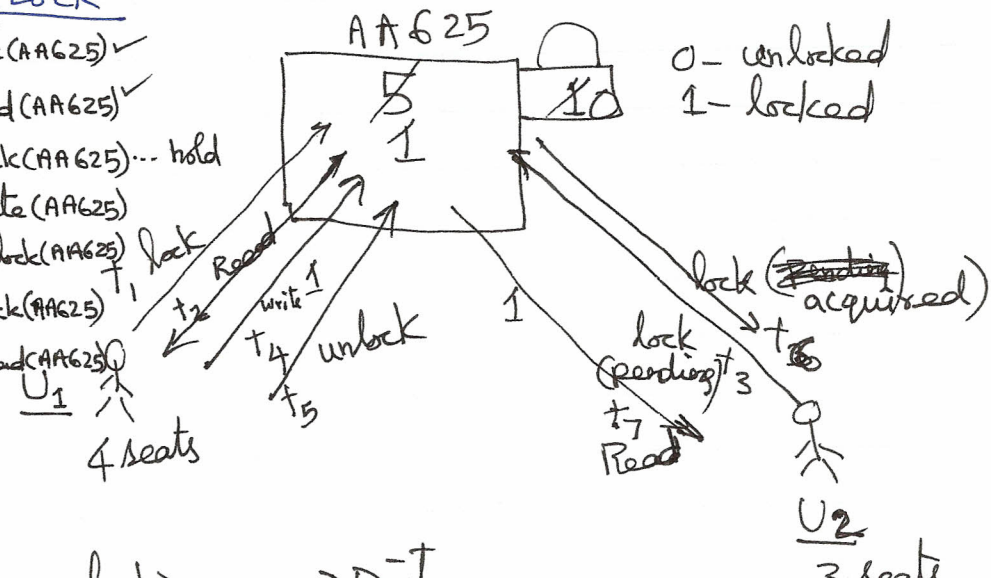
```
unlock(item):
    [Lock(item) = 0
    [if any transaction is waiting for the item,
        wake up the transaction.
```

Lost update problem.



Use of Lock

t_1 U_1: Lock(AA625) ✓
t_2 U_1: Read(AA625) ✓
t_3 U_2: Lock(AA625) ... hold
t_4 U_1: Write(AA625)
t_5 U_1: Unlock(AA625)
t_6 U_2: Lock(AA625)
t_7 U_2: Read(AA625)



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 (1))

Locks

no distinction between read/write

- Binary: 0 unlocked, 1 locked
- S: Shared: 0, 1, 2, 3, ... (Read) unlocked — no. of processes share that item
- E: Exclusive: 0, 1 (Single process) (Write) (or update) Unoccupied ~~unoccupied~~.

Intention locks

Granularity of locks

- Server — level
  - Database level
    - Object (Table) level
      - row level

IS (Intention Shared) — Early warning sign of Shared lock ~~≠~~ below process 3

on Company DB — IS — P_3

— Employee Table — Shared

P_5 write Company — Project (E)

? IS

P_6 IS — Company DB

IS — Department table

S — row.

P8 write Department.

Company DB — IS

Department table — row

?IS

?S — wait

unblocked ✓

Read & Write Locks

(int) (binary)

State: unlocked (RL=0, WL=0)

readlocked (RL=int WL=0)

Writelocked (RL=0 WL=1)

Readlock(item), writelock(item), unlock(item)

Readlock(item):

```
start: if (Lock(item)=='unlocked')
       {
          Lock(item) = 'Readlocked'
          no-of-readers = 1
       }
       else if (Lock item == 'readlocked')
          { nosf_readers++ }
       else{ wait (until lock(item)=='unlocked')
          and
             Lock manager wakes up the transaction
             goto start
          }
```

Writelock(item):

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

Unlock(item):

```
if (Lock(item)=='writelocked')
   { Lock(item) = 'unlocked'
     Wake up Lock Manager to pick up a waiting
                                    transaction
else{ No-of-readers--
     if(No_of_readers ==0)
```