Lecture 26

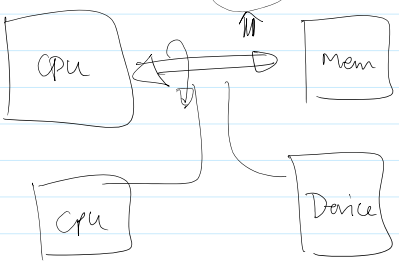
24 September 2018 09:34

spinlocks

what is a lock? — memory object — all or nothing!

atomic instructions

- multiple 'C' statements — non atomic
- single 'C' statement — non-atomic
- single h/w instruction — may not always!
  - inc #mem/reg
    - read — modify — update



design 0

```
getlock (lock_t *l) {
//A    while (l->locked == 1) {};
//B    l->locked = 1;
}
```

x86. lock; prefix instruction which ensures atomicity.

lock; inc #mem.

atomic

— ~~disabling~~ locking memory bus access till completion of instruction.

spin lock? — spin till lock obtained.

~ TSL — test-and-set (lock)

TSL #dest, #src — #olddest = dest; #dest = #src; return #olddest ~ return to set old ZF / SRC ≠ #olddest

atomic (h/w instruction).

```
getlock:  mov R0, 1      <- mov value 1 to reg R0
          TSL LOCK, R0   <- LOCK is current state of LOCK
          CMP R0, 0
          jne getlock
          ret

release lock:  mov LOCK, 0
               RET
```

LOCK -> 0 // free

LOCK -> 1 // locked

(Q) does usage of cpu register R0 bothering you?

(2) x86

xchg dest, src — copies src on dest. returns old dest.

```
getlock;  xchg LOCK, 1
          jnz getlock
          ret

release;  mov LOCK, 0
```

Check in xv6 impln. of spinlock.

lock; xchg #dest, #src

lock ; xchg #dest, #src

cmpxchg — atomic & similar to exchange
uses a implicit register.

cmpxchg #dest, #src

atomic

if eax == dest
    dest = src
    ZF = 1
else
    eax = dest
    ZF = 0

```
L:
    mov eax, 0
    mov ecx, 1
    cmpxchg lock, ecx
    jnz L
    ret
```

(3) acquire_spinlock | spin on CPU & consume it till lock obtained.

release_spinlock

(4) non-spinning lock !

```
getlock:
    TSL LOCK, 1
    JNZ getlock
```

```
getlock:
    TSL LOCK, 1
    jz hooray // get lock
    CALL YIELD // yield the cpu
    jmp getlock
hooray:
```