(bad) implementation of a spin lock

```
struct lock {
    int held = 0;
}

void acquire (lock) {
    while (lock->held);
    lock->held = 1;
}

void release (lock) {
    lock->held = 0;
}
```

What is the context switch happens in between?

➡ We have a race condition

(bad) implementation of a spin lock

```
struct lock {
     int held = 0;
}

void acquire (lock) {
     while (lock->held);
     lock->held = 1;
}

void release (lock) {
     lock->held = 0;
}
```

What is the context switch happens in between?

➡ We have a race condition

The hardware to the rescue

- `test-and-set`(`TAS` x86 CPU instruction)
  atomically writes to the memory location
  and returns its old value in a **single indivisible step**

➡ the caller is responsible for testing if the operation has succeeded or not

```
bool test_and_set(bool *flag) {
  bool old = *flag;
  *flag = True;
  return old;
}
```

This is pseudo-code!
The hardware execute this atomically