CS2106 Tutorial 5

AY 25/26 Sem 1 — github/omgeta

Q1. We can decompose Task A into `load x→store x→load x→store x`
We can decompose Task B into `load x→store x`

So, in there are 5 values $x = 0, 1, 2, 3, 4$

Q2. Yes within a core as only one process could run at once, but in IPC different processes could still causes race conditions accessing shared memory.

Q3. Code:

```c
int atomic_increment( int* t )
{
  // retry if lost race, if another core updates *t before set
  do {
    int temp = *t;
  } while (!_sync_bool_compare_and_swap(t, temp, temp+1));
  return temp+1;
}
```

Q4. Code:

```c
/* Define a pipe-based lock */
struct pipelock {
  int fd[2];
};

/* Initialize lock */
void lock_init(struct pipelock *lock) {
  pipe(lock->fd);
  write(lock->fd[1], "a", 1);
  // init lock so exactly one thread can acquire the lock.
}

/* Function used to acquire lock */
void lock_acquire(struct pipelock *lock) {
  char c;
  read(lock->fd[0], &c, 1);
  // read blocks if there is no byte in the pipe.
}

/* Release lock */
void lock_release(struct pipelock * lock) {
  write(lock->fd[1], "a", 1);
  // write byte back in
}
```