Question 1: (20 points)
Consider a critical section of the following form where a shared variable `shvar` is updated using a local (non-shared) variable `x` as follows:

```
lock(lk);

if(shvar > 0) {
  shvar = max(shvar, x);
}

unlock(lk);
```

- **a.** (5 points) Write the MIPS assembly code for this critical section, assuming that the address of the `lk` variable is in `$a0`, the address of the `shvar` variable is in `$a1`, and the value of variable `x` is in `$a2`. Your critical section should not contain any function calls, i.e., you should include the MIPS instructions for `lock()`, `unlock()`, `max()`, and `min()`. Use `ll/sc` instructions to implement the `lock()` operation, and the `unlock()` operation is simply an ordinary store instruction.
- **b.** (5 points) Solve the above problem, but this time use `ll/sc` to perform an atomic update of the `shvar` variable directly, without using `lock()` and `unlock()`. Note that in this problem, there is no variable `lk`.
- **c.** (5 points) Compare the best-case performance of your code from parts a and b, assuming that each instruction takes one cycle to execute. Note: best-case means that `ll/sc` always succeeds, the lock is always free when we want to `lock()`. Iff there is a branch, we take the path that completes the operation with fewer executed instructions.
- **d.** (5 points) Using your code from part b as an example, explain what happens when two processors begin to execute this critical section at the same time, assuming that each processor executes exactly one instruction per cycle.