Exercise 2

January 17, 2014

An atomic operation is an operation where the processor both reads and writes to the same bus, meaning no other processor or I/O device can write or read memory until that operation is complete.

A semaphore is a variable for access control. The problem in exercise 1 could be solved by setting wait(A) before the increment/decrement and signal(A) after, where A is the semaphore. wait() would decrement the value, preventing any other wait() to be allowed to run. When it's done with the memory, it would call signal(A) and increment A again. A could switch between 0 and 1.

A **mutex** is a flag, when "held" by a thread, that thread can access a certain piece of code in the program and no other. The other threads can only ask and wait to be given the flag, and the thread with the flag releases it when it's done. This is to ensure that the specific piece of code will only be executed by a single thread at a time.

A **critical section** is a specific piece of code that accesses a shared resource. That piece of code must not be accesses by more than one thread at a time. You should lock it and use a mutex.