CECS 526 Assignment 2 (2 points) **Name:Varun Lingabathini**

Due : February 6, 2020, by class time on BeachBoard

Two concurrent threads need to access a shared variable count as follows:

```
Observer: do forever                Reporter: do forever
          observe an event;               print count;
          count := count + 1;             count := 0;
end                                 end
```

The above two concurrent threads are designed to monitor the occurrence of a particular event and report its occurrences. The two threads are executed asynchronously, with Observer keeping track of the event's occurrence by incrementing variable "count" each time the event occurs, and Reporter accessing "count", outputting its current value, and resetting it to zero. Hence the cumulative output produced by Reporter is meant to reflect the number of times that the event has occurred. However, the output may be erroneous due to race condition. Using the tables on the following page, show a scenario that results in an over-reporting (e.g., a scenario where two occurrences of the event have occurred, but the cumulative output shows a total greater than 2.)

Note that computer executes a program (thread) by instructions, and it is due to interrupt between instructions that erroneous results occur. Therefore, you will need to define the instructions for the critical sections of Observer and Reporter below and show sequences of the execution of these instructions in the tables.

Identify critical section in Observer, and define instructions (in assembly language) that implement this critical section:

Critical Section in Observer:

Instructions used to implement the above critical section (label each instruction):

01: R1<- Count (Storing count value in Register R1).
02: R1<- R1+1 (Incrementing the value of R1 by 1).
03: Count <- R1 (Restoring the updated R1 into Count).

Identify critical section in Reporter and define instructions (in assembly language) that implement this critical section:

Critical Section in Reporter:

Instruction used to implement the above critical section (label each instruction):

04: R2<- Count (Storing the count value into the register R2).
05: Count <- 0 (Resetting the value of Count to zero).

Over-reporting

Event occurrence	Initially	#1*				#2				#3																
Instruction executed (show instruction label)		01	02	03	04	01	05	02	03	01	04	05	02	03	04	05										
Value of variable count (initially 0) due to instruction execution	0	0	0	1	1	1	0	0	2	2	2	0	0	3	3	0										
Value of variable (or register) internal to Observer due to instruction execution		0	1	1	1	1	1	2	2	2	2	2	3	3	3	3										
Value output by Reporter, if any, due to instruction execution					1	1	1	1	1	1	2	2	2	2	3	3										

* Each number on this row refers to the occurrence of an event that triggers the observer to increment the count.

Summary of above result (i.e., how many events have occurred, and how many have been reported): 3 events occurred and total of 6 have been reported.