### A Critical Section

1. Reaches critical section at same time

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | flag[0] | flag[1] | lock[0] | lock[1] |
| initial | false | false | false | false |
| A1 | true | “ | “ | “ |
| A2 | “ | “ | “ | “ |
| A9 | “ | “ | “ | “ |
| B1 | “ | true | “ | “ |
| B2 | “ | “ | “ | “ |
| B4 | “ | “ | true | “ |
| B5 | “ | false | “ | “ |
| A10 | false | “ | “ | “ |
| A11 | “ | “ | false | “ |
| B9 | Critical | Critical | Critical | Critical |
| A1 | true | “ | “ | “ |
| A2 | “ | “ | “ | “ |
| A9 | Critical | Critical | Critical | Critical |

1. Deadlock

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | flag[0] | flag[1] | lock[0] | lock[1] |
| initial | false | false | false | false |
| A1 | true | “ | “ | “ |
| A2 | “ | “ | “ | “ |
| B1 | “ | true | “ | “ |
| B2 | “ | “ | “ | “ |
| A4 | “ | “ | “ | true |
| B4 | “ | “ | true | “ |
| A5 | false | “ | “ | “ |
| A6 | stuck | stuck | stuck | stuck |
| B5 | “ | false | “ | “ |
| B6 | stuck | stuck | stuck | stuck |

Now both Thread A and Thread B cannot come out of the while loop because lock[0] and lock[1] is always TRUE.

1. No starvation exists because A10 and A11 will make flag[0] and lock[0] false which will help Thread B come out of the while loop and B10 and B11 will make flag[1] and lock[1] false which will help Thread A to come out of while loop.

### B Interleaving

1. We couldn’t find a way how x = 2.
2. The lowest value we could find was x = 100

|  |  |
| --- | --- |
| Thread 1 | Thread 2 |
| Load R1 x |  |
|  | Load S1 x |
| Inc R1 x |  |
| Store R1 x |  |
|  | Inc S1 x |
|  | Store S1 x |

|  |  |
| --- | --- |
| Example of first iteration Thread 1 | Example of first iteration Thread 2 |
| Load x = 0 |  |
|  | Load x = 0 |
| Inc x = 1 |  |
| Store x = 1 |  |
|  | Inc x = 1 |
|  | Store x = 1 |

At the end of iteration 1, the value of x = 1 instead of x = 2, when this happens a hundred times (in the for loop), the final value of x is x = 100 instead of x = 200.

As the loading of the variable x in thread 2 is doing it right after the loading of R1 x, and the increment and store of S1 is right after the increment and store of R1, the variable x gets overridden by Thread 2. So, it would give the impression that only thread 2 is working.