HomeWork 1

RISHI KUMA SONI *1001774020*

REFERENCE : LECTURE CS VIDEO AND CS SLIDES.

QUESTION 2.1

In the Given Scenario , We consider turn = 1 .

(I)Consider A ,

T0 to be I = 0 , j = 1;

T1 to be I = 1 , J = 0;

A)

(a)

```
while (true)

{ flag[i] = true; (1)
while (flag[j]) { (2)
 if (turn != i) { (3)
flag[i] = false; (4)
while (turn == j) {;} (5)
 flag[i] = true; (6)
 break; (7)
 } }
 critical section
 turn = j; (8)
flag[i] = false; (9)
 noncritical section }
```

1. MUTUALLY EXCLUSIVE

There are two cases to be consider.

CASE 1 :

When T0 thread run , it may run after Step2 and after that it change the context to Thread T1.

In this case the Thread T1 , will run it's critical Section and as a result of that

T1 Critical Section will be in active mode.

After running the critical Section , T1 will switch it's context and T0 will runs its Critical Section.

In the above scenario,

Both the threads are in the Critical Section.

Thus Mutual Exclusiveness is Failed.

CASE2 :

When T0 doesn't change it's context.

In this case, only the T0 critical Section will run and as a result.

We can achieve ***Mutually exclusiveness.***

2. PROGRESS

There are two cases to be consider

CASE 1 :

When there is Mutual Exclusiveness then , the Single thread will only Progress , while other Thread will be in the Starvation.

CASE2:

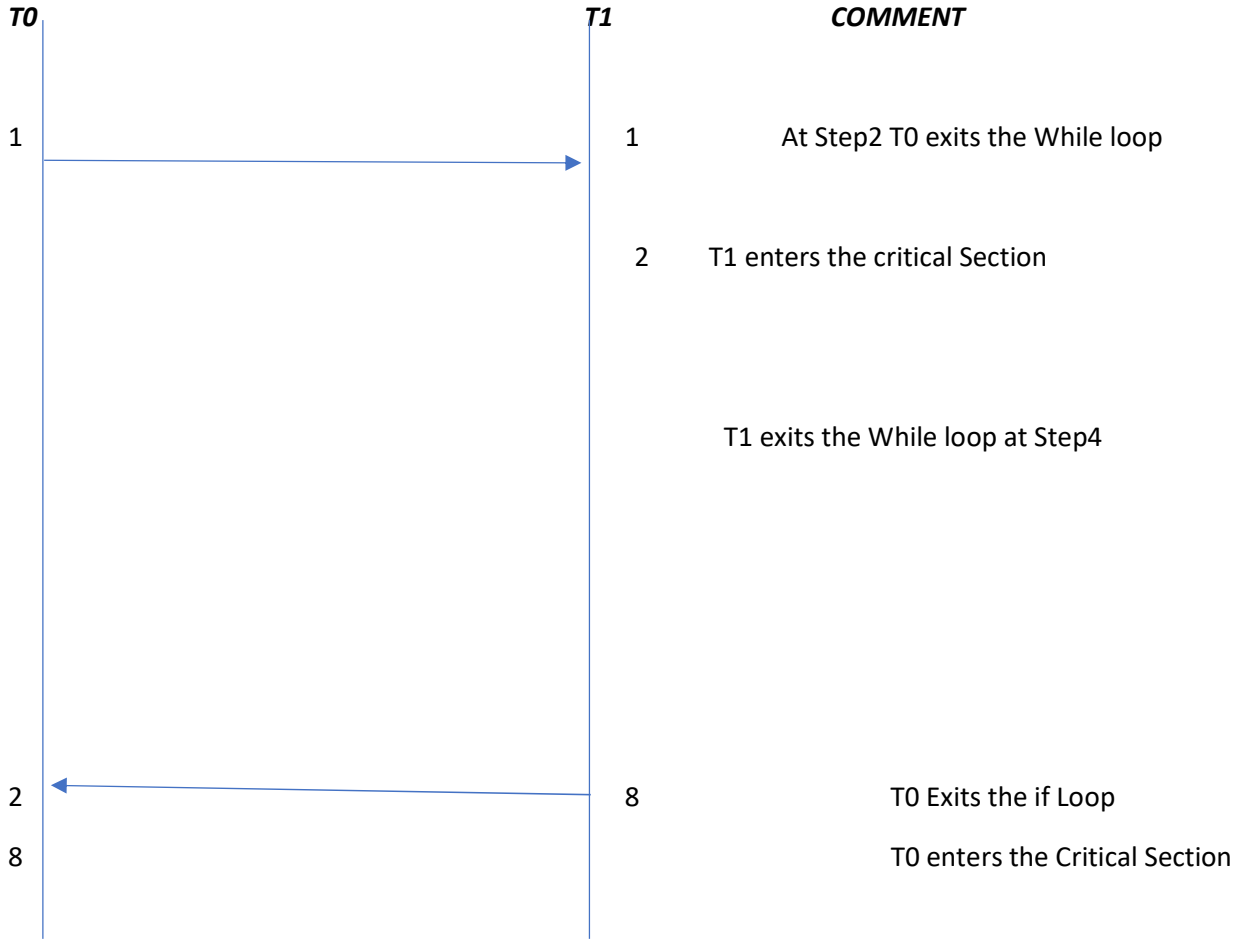
If threads are not mutually exclusive then , progress can be achieved.

3. BOUNDED WAITNING :

In the case of Mutual Exclusiveness, the Bounded time for the another thread is ***infinite.***

But when It is not Bounded then it would be normal .

REPRESENTATION:



(II)CONSIDER B

T0

I = 0 J = 1

T1

I = 1 J =0

(a)

while (true)

{ flag[i] = true; (1)

while (flag[j]) { (2)

if (turn != i) { (3)

flag[i] = false; (4)

while (turn == j) {;} (5)

flag[i] = true; (6)

break; (7)

} }

critical section

turn = j; (8)

flag[i] = false; (9)

noncritical section }

1.MUTUALLY EXCLUSIVE

There are two cases to be consider.

CASE 1 :

When T0 thread run , it may run after Step2 and after that it change the context to Thread T1.

In this case the Thread T1 , will run it's critical Section and as a result of that

T1 Critical Section will be in active mode.

After running the critical Section , T1 will switch it's context and T0 will runs its Critical Section.

In the above scenario,

Both the threads are in the Critical Section.

Thus Mutual Exclusiveness is Failed.

CASE2 :

When T0 doesn't change it's context.

In this case, only the T0 critical Section will run and as a result.

We can achieve ***Mutually exclusiveness.***

2.PROGRESS

There are two cases to be consider

CASE 1 :

When there is Mutual Exclusiveness then , the Single thread will only Progress , while other Thread will be in the Starvation.

CASE2:

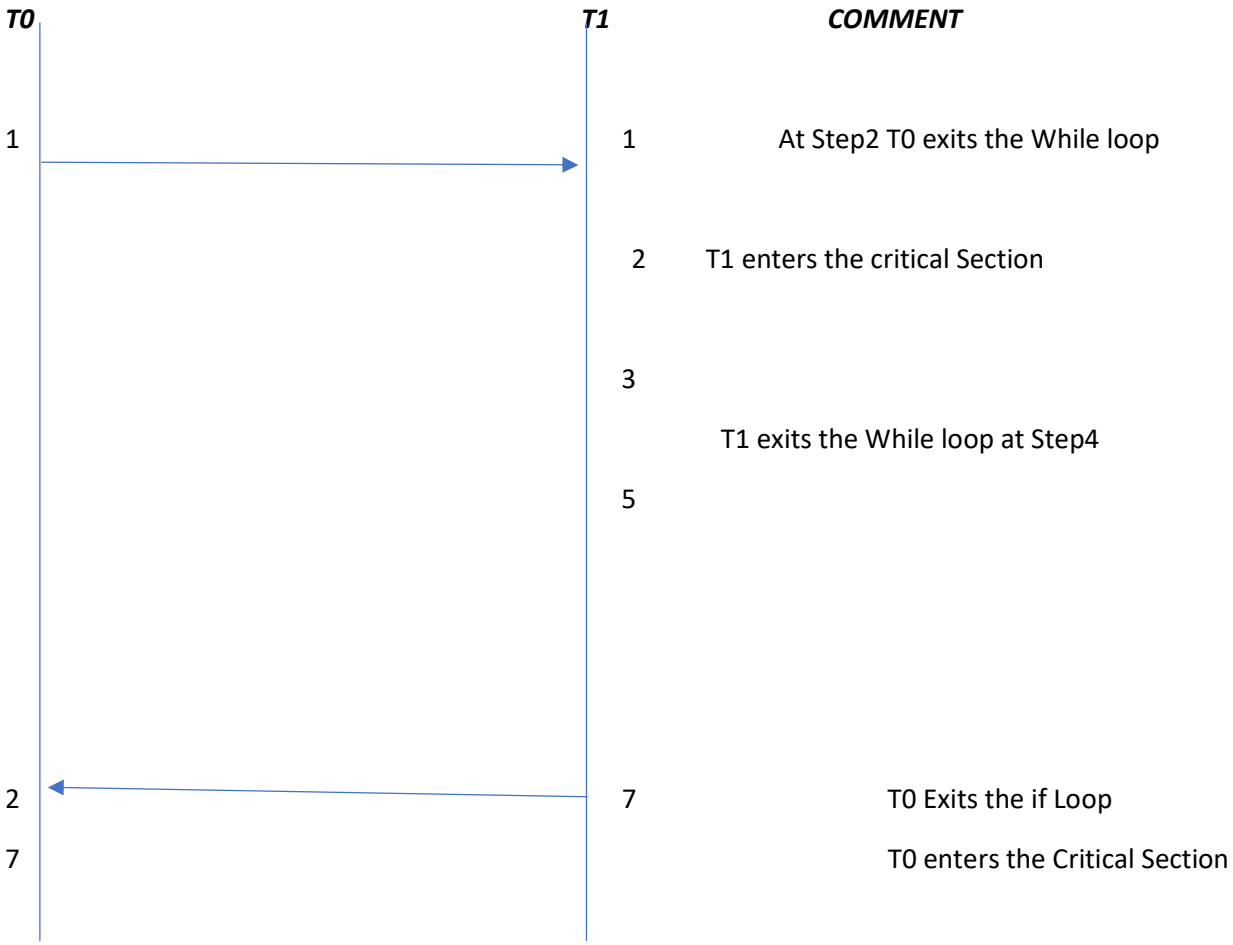
If threads are not mutually exclusive then , progress can be achieved.

3.BOUNDED WAITNING :

In the case of Mutual Exclusiveness, the Bounded time for the another thread is ***infinite.***

But when It is not Bounded then it would be normal .

REPRESENTATION:



(III)CONSIDER C

- ***T0 I = 0 J = 1***
  T1 I = 1 J = 0

1.MUTUALLY EXCLUSIVE

There are two cases to be consider.

CASE 1 :

When T0 thread run , it may run after Step2 and after that it change the context to Thread T1.

In this case the Thread T1 , will run it's critical Section and as a result of that

T1 Critical Section will be in active mode.

After running the critical Section , T1 will switch it's context and T0 will runs its Critical Section.

In the above scenario,

Both the threads are in the Critical Section.

Thus Mutual Exclusiveness is Failed.

CASE2 :

When T0 doesn’t change it’s context.

In this case, only the T0 critical Section will run and as a result.

We can achieve ***Mutually exclusiveness.***

2.PROGRESS

There are two cases to be consider

CASE 1 :

When there is Mutual Exclusiveness then , the Single thread will only Progress , while other Thread will be in the Starvation.

CASE2:

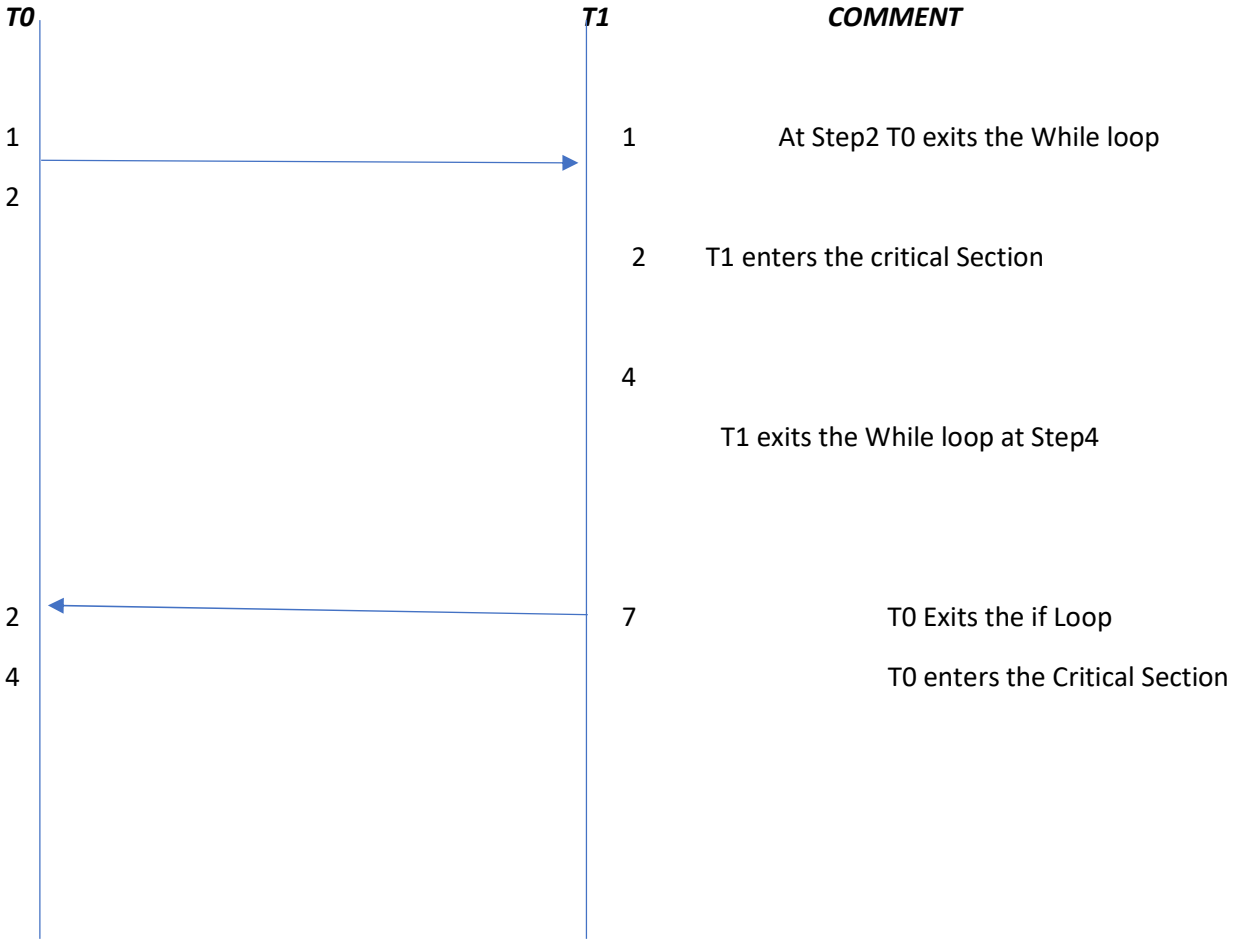
If threads are not mutually exclusive then , progress can be achieved.

3.BOUNDED WAITNING :

In the case of Mutual Exclusiveness, the Bounded time for the another thread is ***infinite.***

But when It is not Bounded then it would be normal .

REPRESENTATION :


T0
T1
COMMENT
1
1
At Step2 T0 exits the While loop
2
2
T1 enters the critical Section
4
T1 exits the While loop at Step4
2
7
T0 Exits the if Loop
4
T0 enters the Critical Section

IV) CONSIDER WHEN A = T0 AND B = T1.

Thread T0

(b)

while (true)

{ flag[i] = true; (1)

while (flag[j]) { (2)

if (turn != i) { (3)

flag[i] = false; (4)

while (turn == j) {;} (5)

flag[i] = true; (6)

break; (7)

} }

critical section

turn = j; (8)

flag[i] = false; (9)

noncritical section }

Thread T1

(c) while (true
{

flag[i] = true; (1)

while (flag[j]) { (2)

flag[i] = false; (3)

while (turn == j) {;} (4)

flag[i] = true; (5)

} (6)

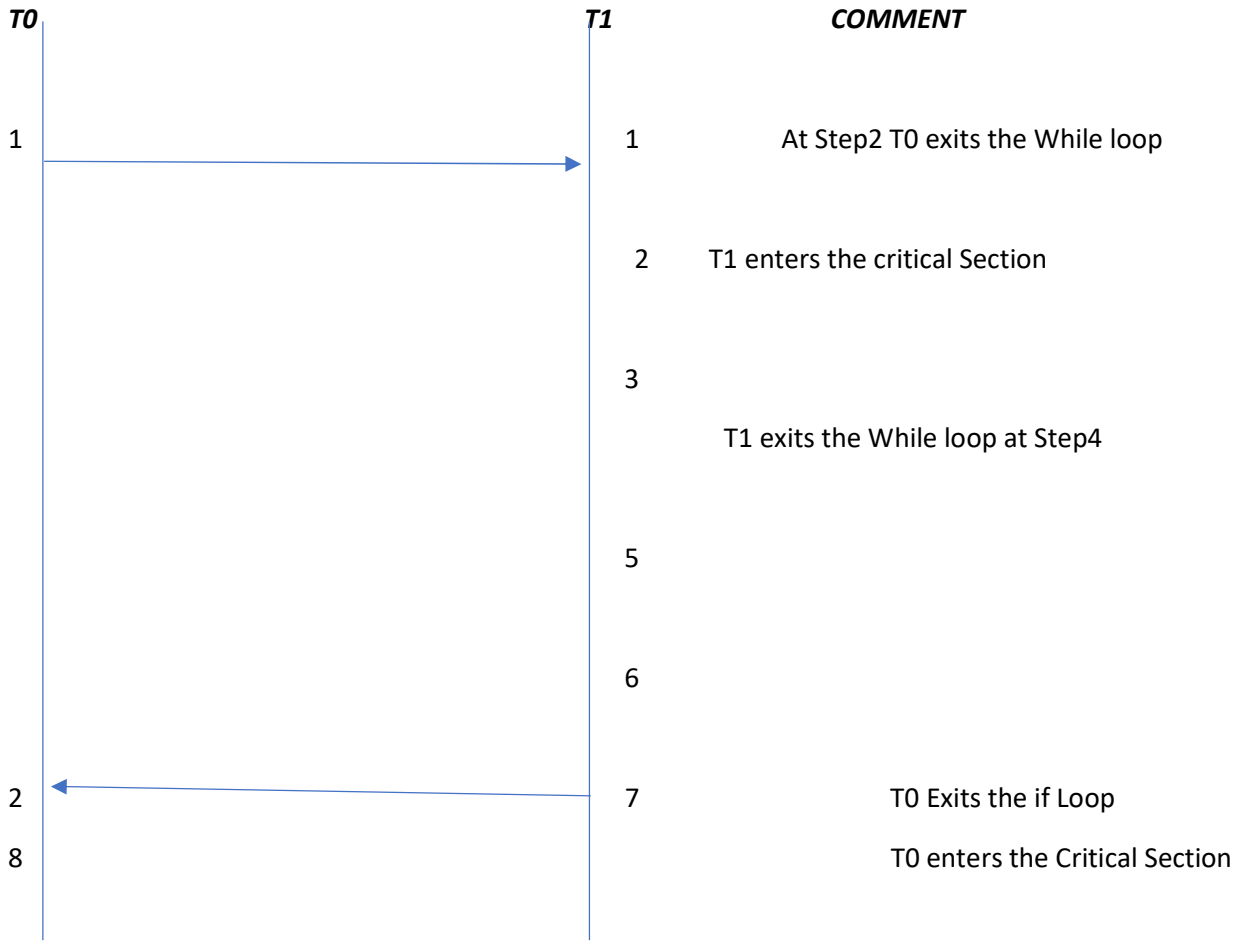
critical section

turn = j; (7)

flag[i] = false; (8)

noncritical section }

REPRESENTATION


T0
T1
COMMENT
1
1
At Step2 T0 exits the While loop
2
T1 enters the critical Section
3
T1 exits the While loop at Step4
5
6
2
7
T0 Exits the if Loop
8
T0 enters the Critical Section

Consider Turn = 1 ;

SOLUTIONS :

1.Mutually Exclusive :

THERE ARE TWO CASES

Case 1 .

When we switch the Context, after Critical Section in Thread T0 , then we cannot achieve Mutually Exclusion and we fell into the Incomplete Solution 1 or Incomplete Solution 2 Category.

The reason being ,

For the Solution 1 ; T0 AND T1 is not **Mutually Exclusive** because after the context change take place from the Tread T1 after running step 7 , T0 will run its step 2 and step 8 . Which shows that T0 AND T1 both are in ***the CRITICAL SECTION*** which FAILS our first condition.

Case 2 .

When We don't Switch the context after the step 7 but let we finish the Thread T1 completely and make to run it's non-critical section. And once again when we run the Thread T1, at this point we can switch the context with the completion of the Thread T1.

In this case we can achieve ***Mutually Exclusiveness***.

2.PROGRESS :

Thus we can observe for the above the code that

Both the codes share the common the variables , but the variable doesn't affect the progress of the both the Thread T0 AND T1.

3.BOUNDED WAITING :

There is ***no Starvation*** in the two above threads i.e T0 AND T1.

(V) WHEN B = T0 AND A = T1

1 . MUTUALLY EXCLUSIVE

CASE 1 :

When we Consider Thread T0 as B and Thread T1 as A,

When the Thread T0 , after the execution of the step1 , the context change will take place.

Now the thread T1 will come into picture.

T1 will run it' Critical Section.

In the above case the Mutual Exclusiveness is achieved.

CASE2 :

When we Consider Thread T0 as B and Thread T1 as A,

When the Thread T0 , after the execution of the step1 , the context change will take place.

Now the thread T1 will come into picture.

T1 will run it' Critical Section.

After that context change will take place and as a result , Thread T0 will also in the Critical Section.

Thus Mutually Exlusiveness is failed.

2 .PROGRESS

When If Mutual Exclusive is achieved then , Both the threads make the progress because of the critical section are running.

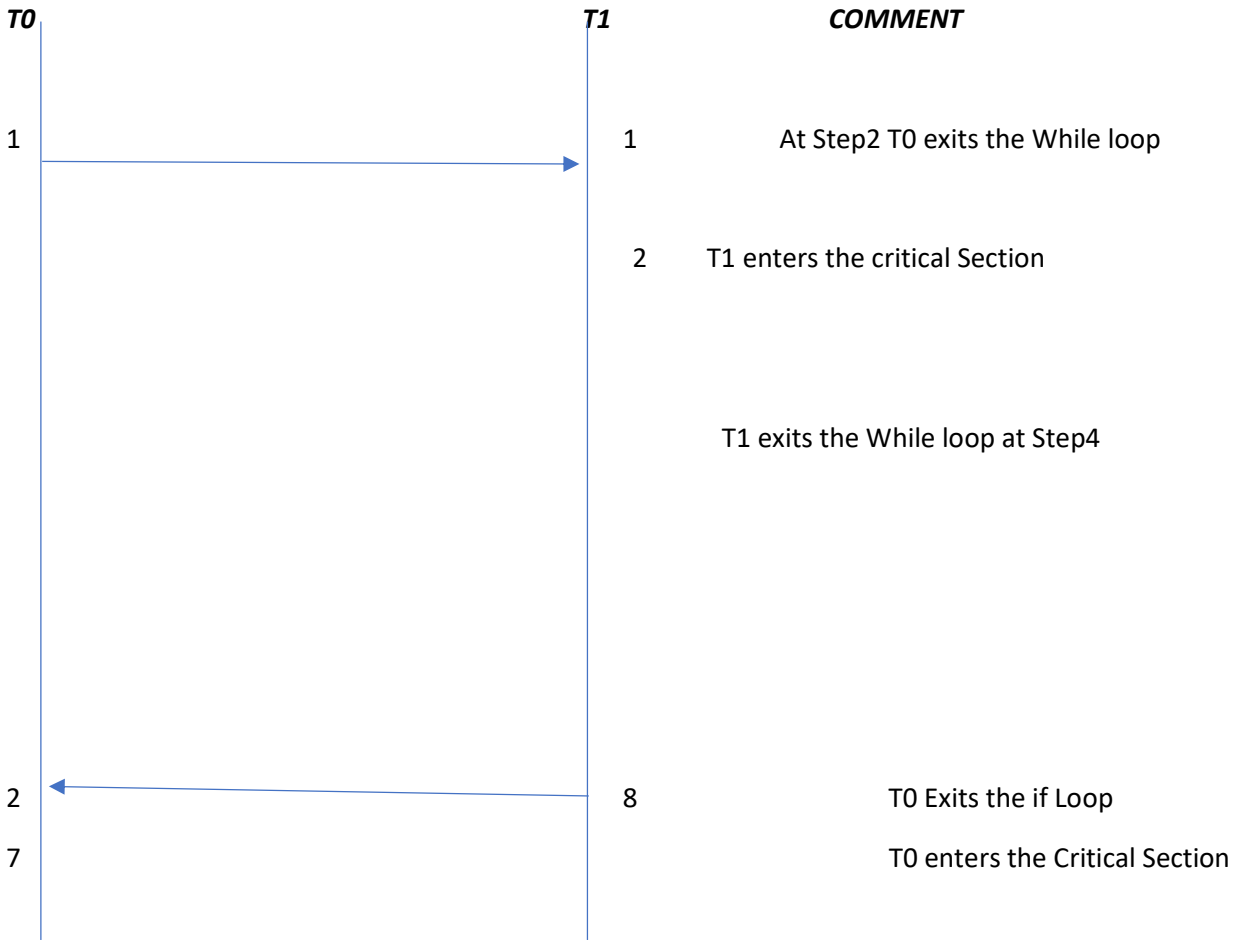
If the Mutual Exclusiveness is not achieved , then only one Thread keep running and as a result progress is not achieved.

3. BOUNDED WAITING

When the threads are not mutually exclusive, then in that case, one of the thread is in the ***Starvation***.

If the Threads are mutually exclusive then , the switching between the threads takes places.

REPRESENTATION:


T0
T1
COMMENT
1
1
At Step2 T0 exits the While loop
2
T1 enters the critical Section
T1 exits the While loop at Step4
2
8
T0 Exits the if Loop
7
T0 enters the Critical Section

(VI)

WHEN A = T0 AND C = T1

Thread T0

(a)

while (true)

{ flag[i] = true; (1)

while (flag[j]) { (2)

if (turn != i) { (3)

flag[i] = false; (4)

while (turn == j) {;} (5)

flag[i] = true; (6)

break; (7)

} }

critical section

turn = j; (8)

flag[i] = false; (9)

noncritical section }

Thread T1

(c)while (true) { flag[i] = true; (1)

turn = j; (2)

while (flag[j] && turn== i) {;} (3)

critical section flag[i] = false; (4)

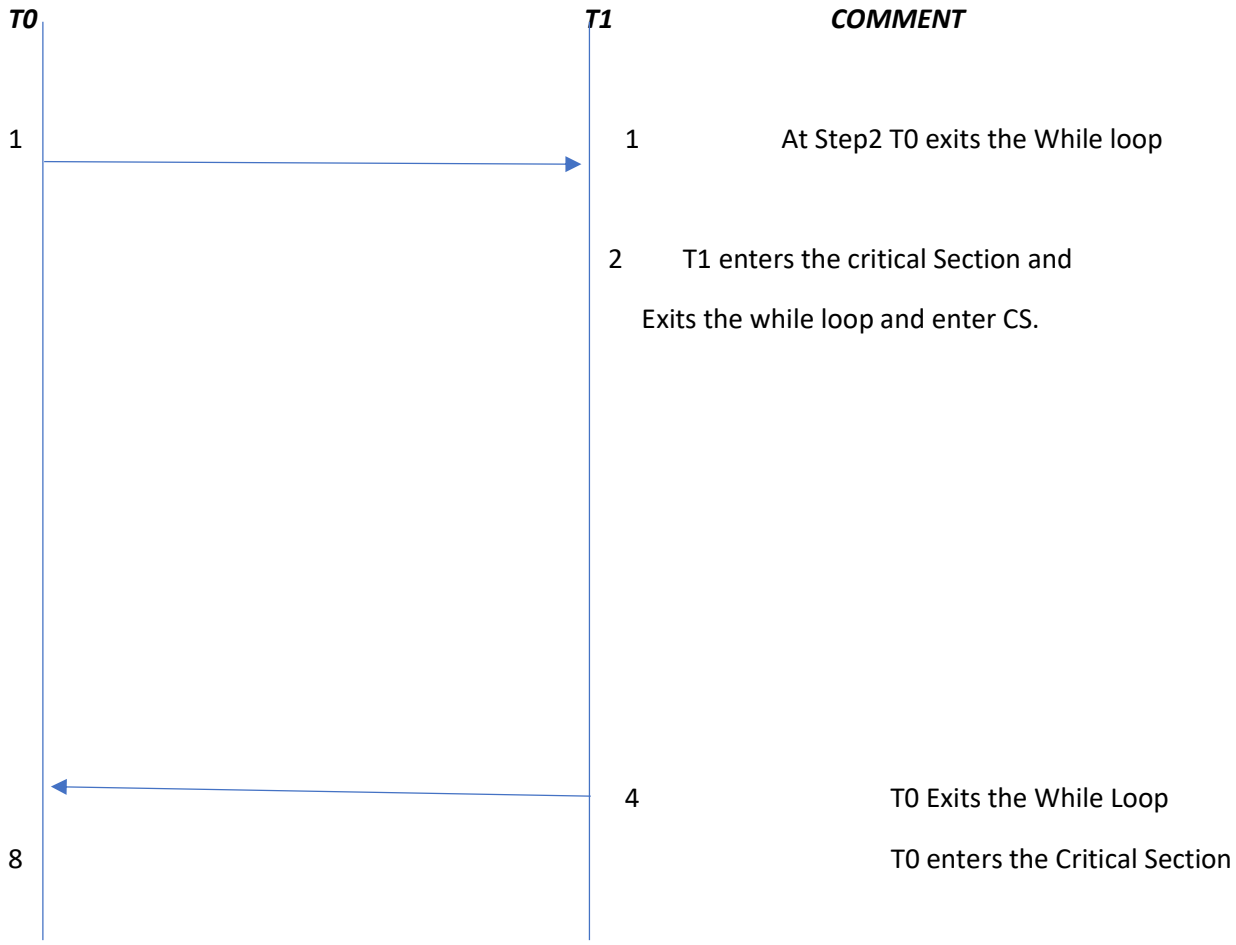
noncritical section}

The Below diagram is when we change the context from .

Like it mention in the Incorrect Solution 1 and 2.

.

REPRESENTATION



SOLUTION :

1. ***Mutually Exclusive***

There are two cases to be consider.

CASE 1.

When the Thread T0 execute , after Step1 , context is change and Thread T1 start executing.

T1 enters into its critical section. After Completing CS, it again changed it's context and Thread T0 starts executing and enters it's Critical Section.

In this Scenario , both the Threads T0 AND T1 are in the Critical Section.

Mutually Exclusive Fails.

CASE 2 .

What if the context is not changed ?

Answer :

In that the Thread will execute infinite time and other Thread cannot enter its critical Section .

Mutually Exclusive is achieved !!

2. PROGRESS

There are two case to be consider :

CASE 1 .

In the above case 1 . Both the Thread execute and make ***progress .***

CASE 2 .

When we don't change the context and let the Thread T0 run , in that case there ***no progress*** for the Another Thread i.e T1.

3 .BOUNDED WAITING :

When we consider the Case2 From the Mutually Exclusive or from the Progress .

The bounded time is ***INFINTE .***

One of the Thread is in the ***STARVATION.***

(VI) WHEN A = T1 AND C = T0

1 . MUTUALLY EXCLUSIVE

CASE1 :

When the Thread T0 will execute then, after executing step1 to step3 ,

Context switch will take place and another Thread i.e T1 , A will execute.

A will reach it's critical section.

Thus in this case , Mutual exclusive is achieved.

CASE2:

When the thread T0 doesn't change it context then the Thread T0 will ll in loop and as a result

No other thread will be in critical section.

Mutual Exclusive is achieved .

2 . PROGRESS

CASE 1 .

T0 AND T1 will be in the critical Section .

Thus both the threads will do progess.

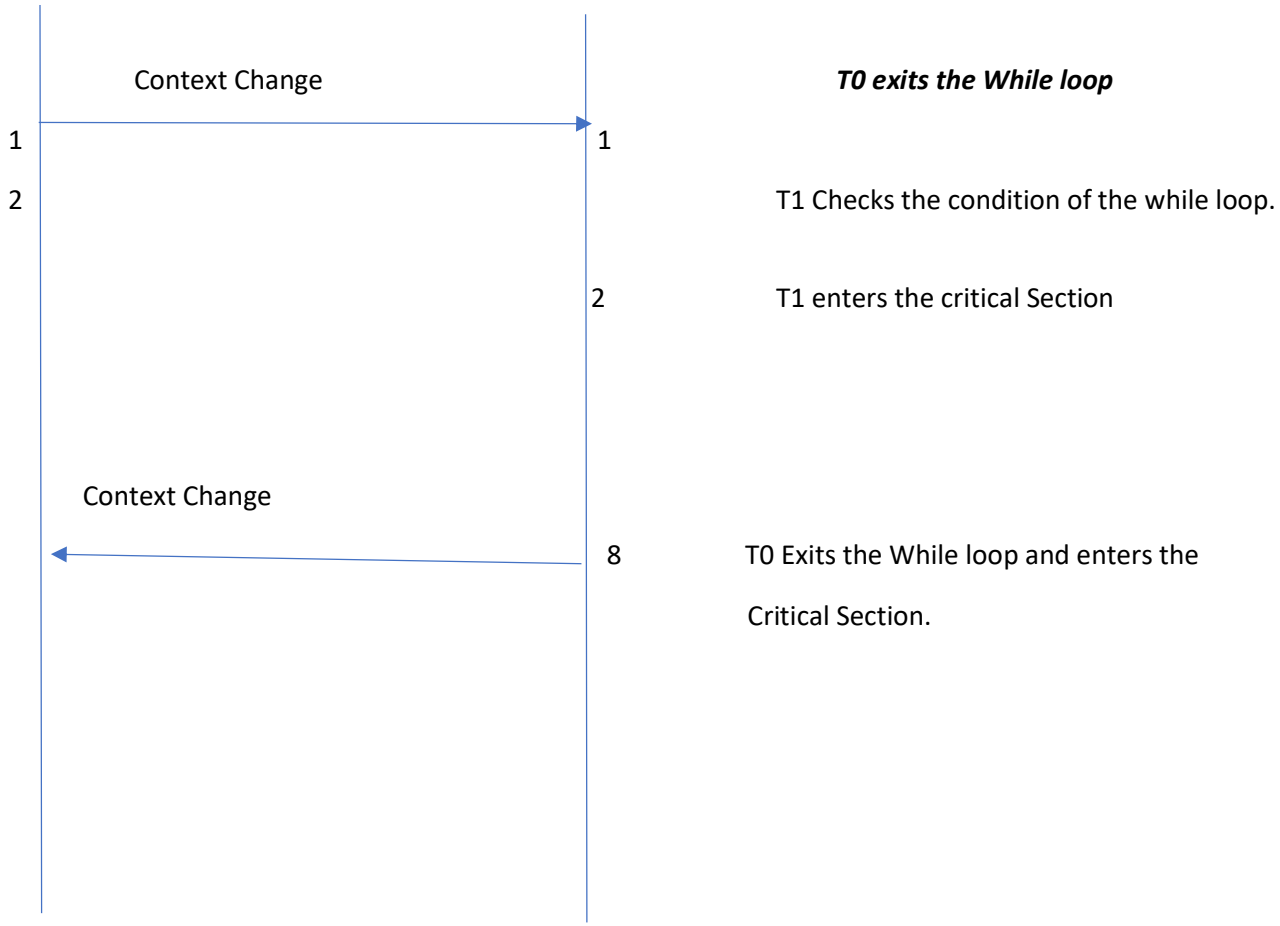
CASE 2:

If the T0 doesn't make the context change then, only one thread will execute and as result No progress for the Another Thread.

3. BOUNDED THREAD

For the mutual exclusiveness , other thread T1 bounded time is infinite.

It goes in the ***Starvation.***



VIII . WHEN B = T0 AND C = T1

Thread T0

(b)while (true)

{ flag[i] = true; (1)

while (flag[j]) { (2)

flag[i] = false; (3)

while (turn == j) {;} (4)

flag[i] = true; (5)

} (6)

critical section

turn = j; (7)

flag[i] = false; (8)

noncritical section}}

Thread T1

(c)while (true) { flag[i] = true; (1)

turn = j; (2)

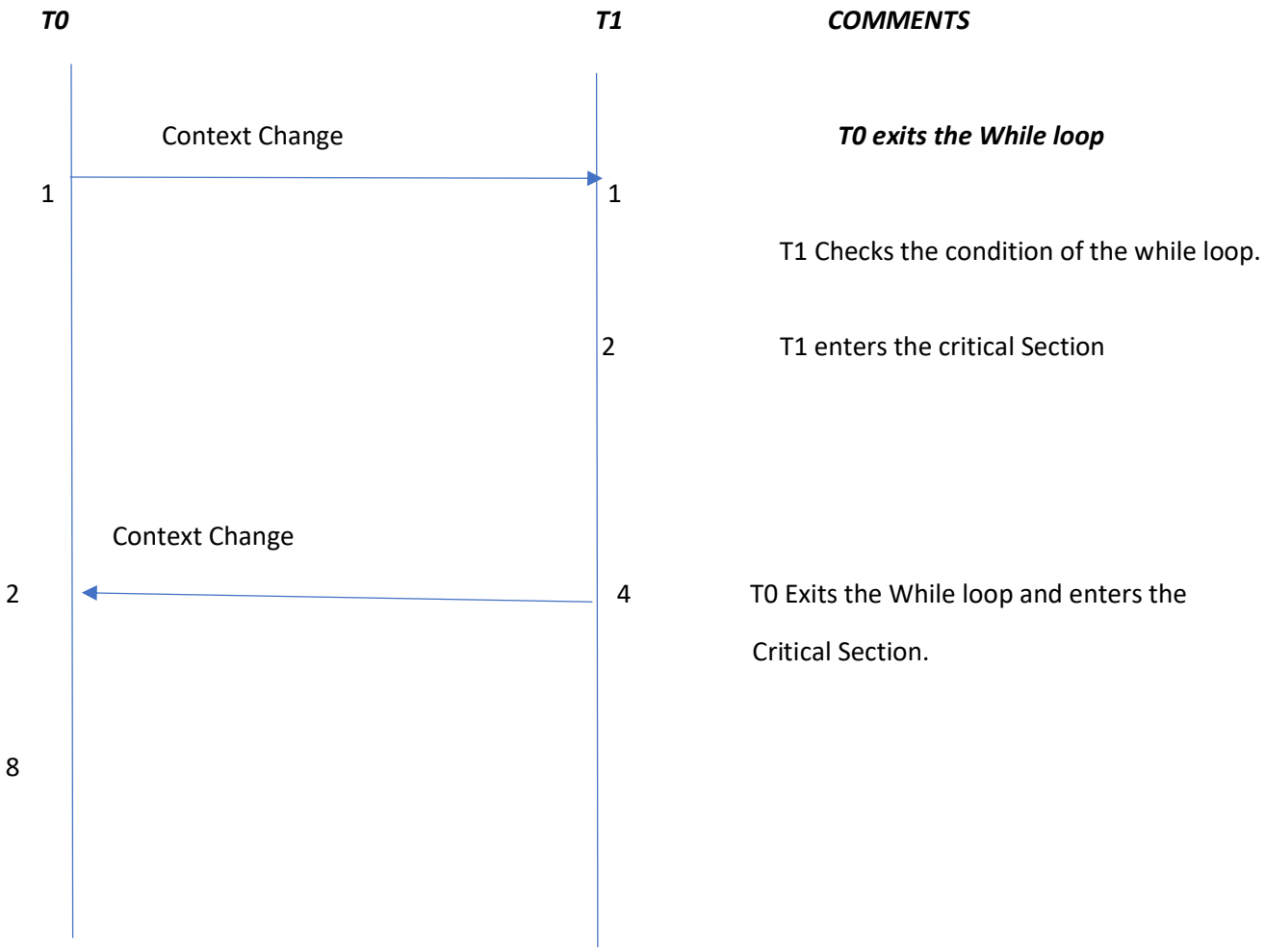
while (flag[j] && turn== i) {;} (3)

critical section flag[i] = false; (4)

noncritical section

}

REPRESENTATION



SOLUTION :

1. Mutually Exclusive

There are two cases to be consider.

CASE 1.

When the Thread T0 execute , after Step1 , context is change and Thread T1 start executing.

T1 enters into its critical section. After Completing CS, it again changed it's context and Thread T0 starts executing and enters it's Critical Section.

In this Scenario , both the Threads T0 AND T1 are in the Critical Section.

Mutually Exclusive Fails.

CASE 2 .

What if the context is not changed ?

Answer :

In that the Thread will execute infinite time and other Thread cannot enter its critical Section .

Mutually Exclusive is achieved !!

2. PROGRESS

There are two case to be consider :

CASE 1 .

In the above case 1 . Both the Thread execute and make progress .

CASE 2 .

When we don't change the context and let the Thread T0 run , in that case there no progress for the Another Thread i.e T1.

3 .BOUNDED WAITING :

When we consider the Case2 From the Mutually Exclusive or from the Progress .

The bounded time is ***INFINTE .***

One of the Thread is in the ***STARVATION.***

(IX) WHEN B = T1 AND C = T0

1 . MUTUALLY EXCLUSIVE

There are two cases to follow :

CASE1:

WhenThread T0 will run , after checking the While loop, It will change its context and Thread T1 will run .

As a result T1 will be in the Critical Section.

But when the Thread T1 will run again.

It will change the context. And the T0 will run.

So from the above case it is No mutual exclusive.

CASE2 :

WHAT IF NO CHANGE OF CONTEXT ?

Answer :

Mutual Exclusiveness is achieved.

2 .PROGRESS

CASE 1 :

If there is no mutual exclusiveness , then progress of both the threads will take place.

CASE2 :

If there is mutual exclusive the another thread can progress.

And it will be starved.

3 . BOUNDED WAITING

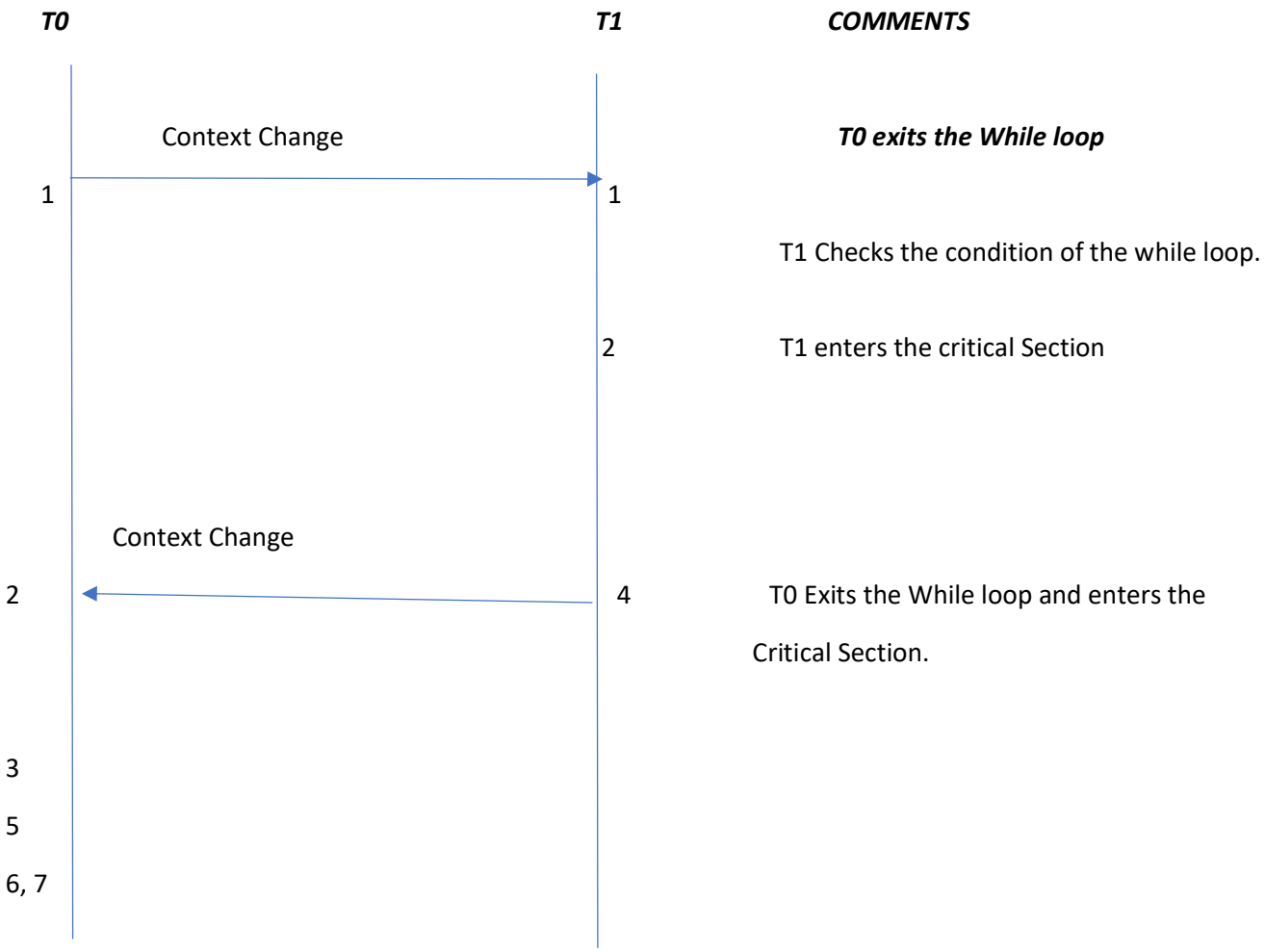
Case 1 :

When there is no mutual exclusiveness then the bounded waiting is normal

CASE 2:

When there is mutual exclusiveness the Bounded Waiting will be Infinite.

REPRESENTATION:



- ***2.5***

Suppose that we switch the order of the first two statements in Peterson's algorithm:

boolean intendToEnter0 = false, intendToEnter1 = false; int turn; // no initial value for turn is needed.

T0

```
while (true) {
turn = 1(1)
intendToEnter0 = true(2)
while (intendToEnter1 &&(3)
turn == 1) {;}
critical section(4)
intendToEnter0 = false;(5)
noncritical section(6)
}
```

T1

```
while (true) {
turn = 0;(1)
intendToEnter1 = true;(2)
while (intendToEnter0 &&(3)
turn == 0) {;}
critical section(4)
intendToEnter1 = false;(5)
noncritical section (6) }
```

SOLTUION :

1 .MUTUALLY EXCLUSIVE

CASE 1 :

Suppose that the Thread T0 is running.

T0 will enters it's critical Section.

After running the Steps 1 , Steps 4. We change the Context and hence the Thread T1 will run.

After the running the Thread T1 , the Condition 3 is true , which makes the Thread T0 to run again.

From the above description we can consider that ,

Only T0 thread is running and only it's critical section is running.

Thus it follows the ***Mutual Exclusion.***

2.PROGRESS

Although T0 Thread is running again and again.

In the that case , T0 is making the ***Progress*** but

The thread T1 has no progress at all , because it always making the Step 3 to be True.

So , ***No Progress*** for the T1 thread.

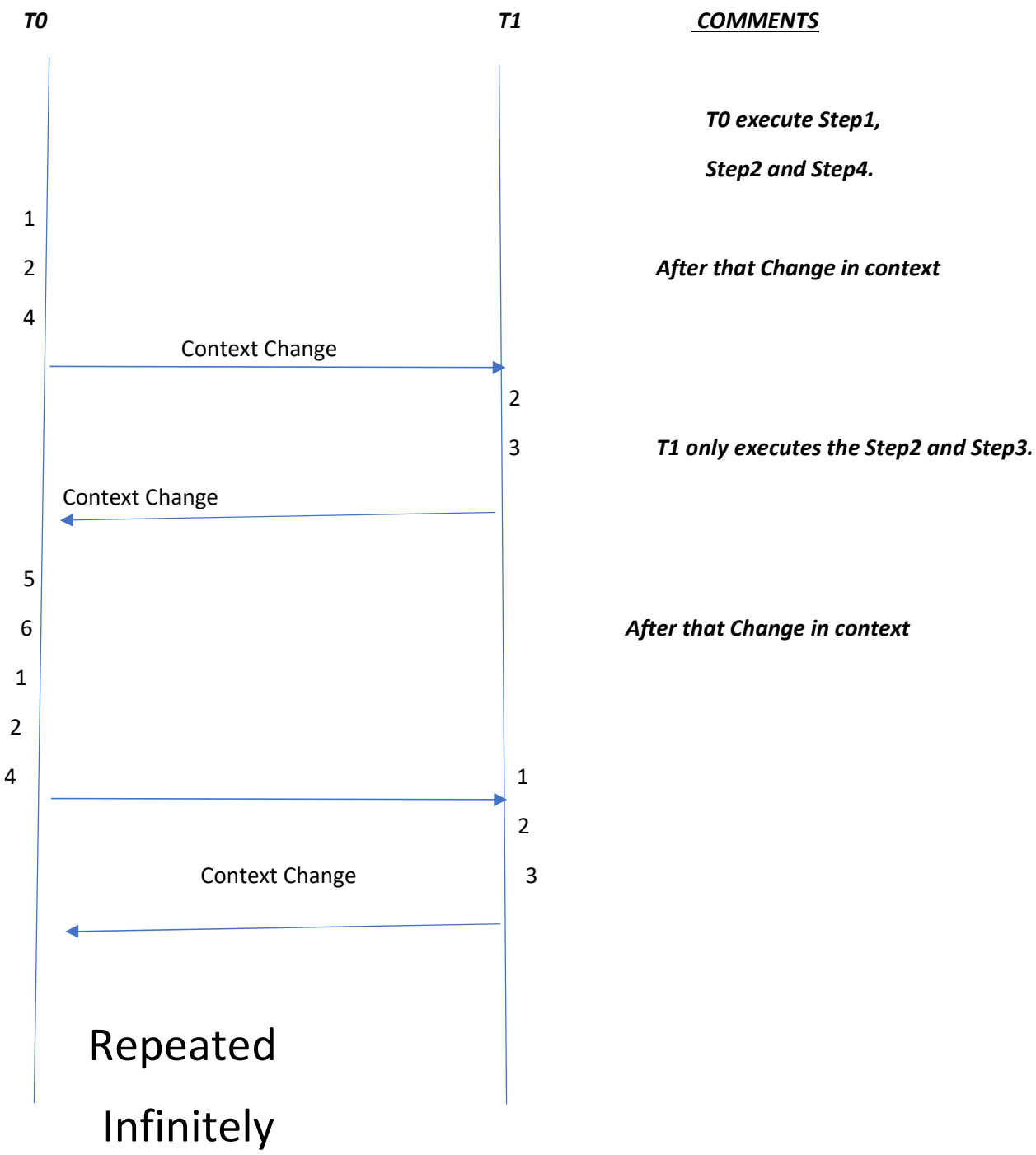
3 . Bounded Waiting

Waiting for the Thread T1 is ***infinite*** because its never entering into the critical Section.

So T0 is executing after every steps but T1 not.

NOT BOUNDED.

REPRESENTATION



No, this solution guarantee ***Mutual Exclusion*** but not the ***Progress*** and the ***Bounded Waiting.***

- ***2.7***

while (true) { number[i] = max(number) + 1; (1)

for(int j=0; j<n; j++) (2)

while (j != i && number[j] != 0 && (3)

(number[j],j) < (number[i],i)) {;} (4)

critical section (5)

number[i] = 0; (6)

noncritical section (7) }

(a) Explain how the value of number[i] grows without bound.

(b) One suggestion for bounding the value of number[i] is to replace the statement

number[i] = max(number) + 1;

with the statement

// numThreads is # of threads in the program number[i] = (max(number) + 1) % numThreads;

Is this suggestion correct? Explain.

SOLTUION :

(a) Explain how the value of number[i] grows without bound.

Answer :

When we don’t have the bound , then there would be many threads to be executed at the same time i.e all the threads hanging in the Critical Section. This causes **number[i] to keep on increasing**.

Below I consider a given scenario.

- ➔ If there infinite number of Process. Then There would be no progress with the above code . Thus the ***Mutual Exclusion*** condition fails.

Suppose when have infinite Threads;

T0, T1, T2, ……. T∞

Now consider , Two Threads T0 and T1.

T0 execute step1 , step2, step3 and step4.

Now the threads change the context and another threads T1 follow the same steps.

Thus both the threads are in the critical section.

Similarly when the number of threads keep on increasing if we don’t have the bound then it will create chaos and our process fails.

b.)

What If we replace the number[i] = (maxnumber)+1%numberofthreads

?

Answers :

Suppose we have three threads

T1, T2 , T3

Each associate with values 1 ,2 ,3 respectively.

Now consider , T1 thread is processing , in this case.

Max value is 3 .

And the number of threads is 3.

So the number[i] will be

Numberof[0] = 3+1 % 3

Numberof[0] = 4 % 3

Number[0] = 4

Thus the value of max will be same.

Similarly for the threads T1 and T2 .

Number[1] = 3+1 % 4

Numberof[1] = 4

And

Numberof[2] = 4

Thus we can observe that , the ticket number is same , for all the threads i.e 4 .

Thus this solution doesn’t create any modification.