Nov 20, 2024

CS 350 DISCUSSION 11

Midterm 2 Review

Let's Practice

Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS			
VS			
CC			
AD			

Question 1 [4 points] You are considering a high-performance single-processor machine on which you have measured the runtime of the tasks. For HS, it is between 3 ms and 4 ms. For VS it is 1±0.5 ms. For CC it falls in the range [0.5, 3] ms. Finally, for AD the runtime is between 4 ms and 7.5 ms. Complete the missing task parameters in Table 2.

Problem 19.1 from the book

Let's Practice

Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

Question 1 [4 points] You are considering a high-performance single-processor machine on which you have measured the runtime of the tasks. For HS, it is between 3 ms and 4 ms. For VS it is 1±0.5 ms. For CC it falls in the range [0.5, 3] ms. Finally, for AD the runtime is between 4 ms and 7.5 ms. Complete the missing task parameters in Table 2.

Problem 19.1 from the book

Let's Practice

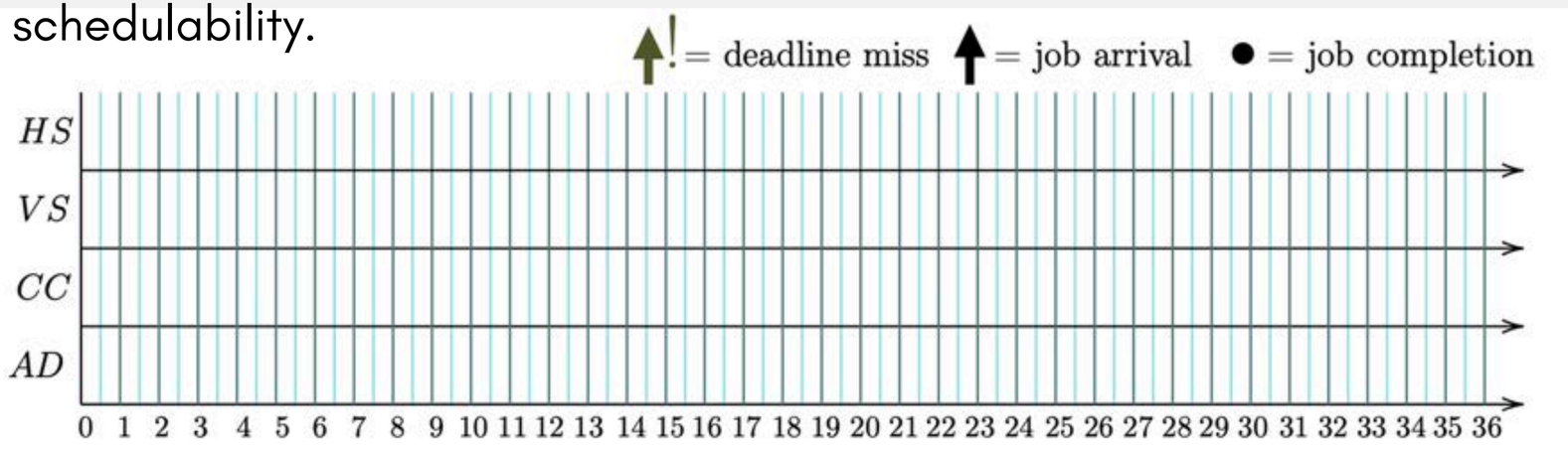
Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

Question 2 [6 points] Is the system schedulable using Shortest Job Next? Motivate your answer. Use the grid provided below only if needed and use only what you see up to time 36 to conclude about schedulability.



Problem 19.1 from the book

Let's Practice

Exercise 1

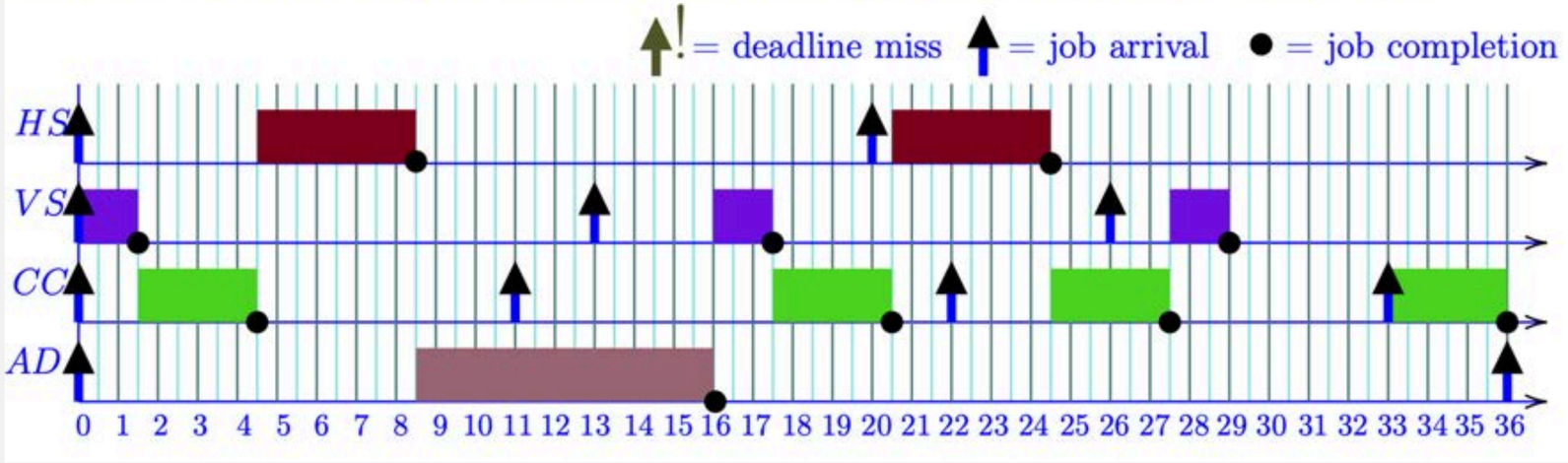
The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

We have no schedulability test for SJN, so we must try to draw the schedule that would be produced by SJN.

From the schedule visualized below, it seems that there is no deadline miss from 0 to time 36. We could therefore conclude that the system is schedulable uinder SJN.



Problem 19.1 from the book

Let's Practice

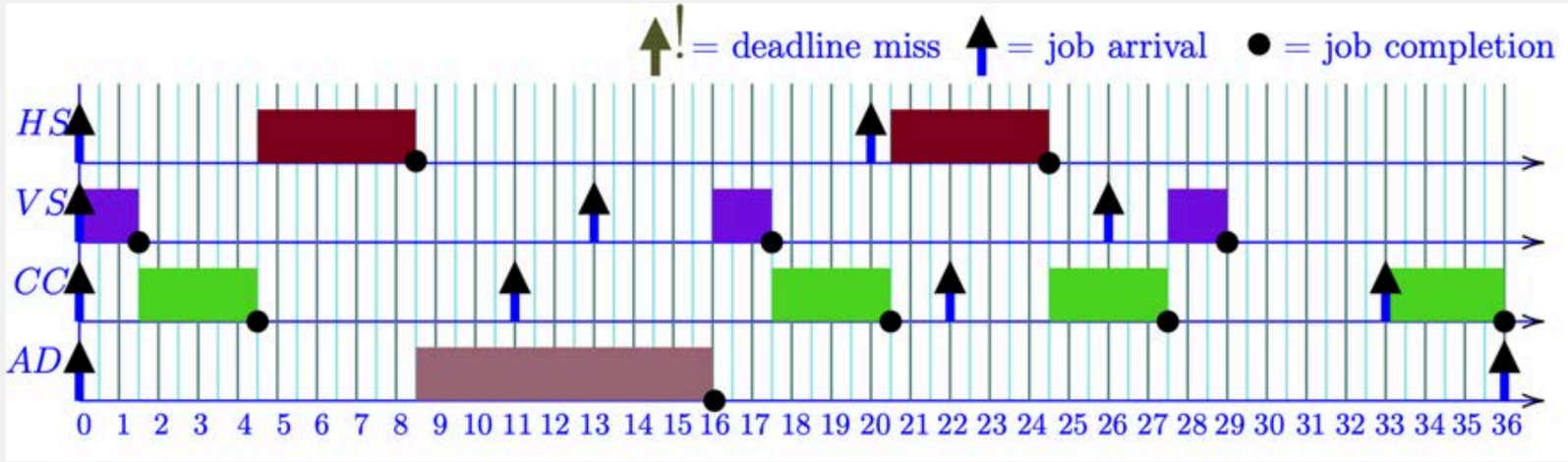
Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

NOTE: In this case, to be perfectly sure we would need to draw all the way until the hyperperiod of the tasks, which is the L.C.M. of all the task periods. In this case, that would be time 25,740.



Problem 19.1 from the book

Let's Practice

Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

Question 3 Is the system schedulable using Rate Monotonic? Motivate your answer. Use the grid provided below only if needed and use only what you see up to time 36 to conclude about schedulability.

Problem 19.1 from the book

Let's Practice

Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

The first thing to try is to apply the RM schedulability test. The total utilization $U = U_{HS} + U_{VS} + U_{CC} + U_{AD} = 4/20 + 1.5/13 + 3/11 + 7.5/36 = 0.7964$

The RM utilization bound for $m = 4$ tasks is $4(2^{1/4} - 1) = 0.7568 < U$, thus the test is inconclusive.

Since $U < 1$, we attempt to draw the schedule produced by RM in the grid below. No deadline is observed until the end of the period of the lowest-priority task, therefore we can conclude that the taskset is schedulable under RM.

Problem 19.1 from the book

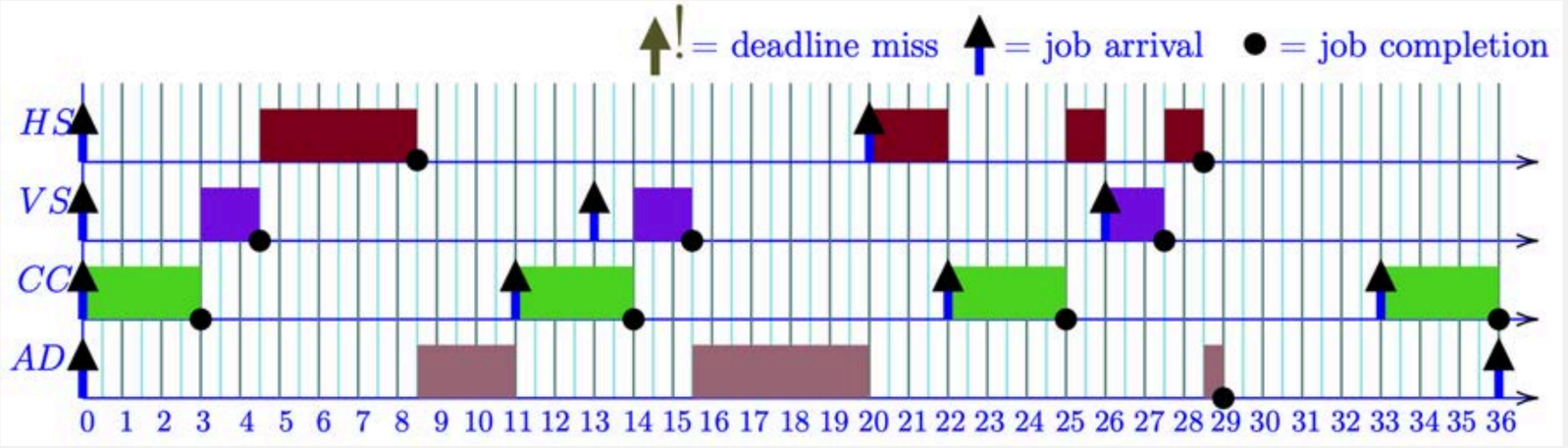
Let's Practice

Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.



Problem 19.1 from the book

Let's Practice

Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

Question 4 In an attempt to reduce overall power consumption, you decide to in- vestigate the possibility of using a slower 2-processors system instead of the original machine. The clock frequency is exactly half compared to the original machine, mean- ing that all the runtimes are exactly twice as long. Is the system schedulable if we apply RM-FF with the tasks in the following order: CC, HS, VS, AD?

Problem 19.1 from the book

Let's Practice

Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

The 4 plasma control tasks on the new machine would have the following WCETs: $C_{CC} = 6$ ms, $C_{HS} = 8$ ms, $C_{VS} = 3$ ms, $C_{AD} = 15$ ms.

Problem 19.1 from the book

Let's Practice

Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

The 4 plasma control tasks on the new machine would have the following WCETs: $C_{CC} = 6$ ms, $C_{HS} = 8$ ms, $C_{VS} = 3$ ms, $C_{AD} = 15$ ms.

We can first check the schedulability test for RM-FF. The new overall task utilization is $U = \frac{C_{CC}}{T_{CC}} + \frac{C_{HS}}{T_{HS}} + \frac{C_{VS}}{T_{VS}} + \frac{C_{AD}}{T_{AD}} = 6/11 + 8/20 + 3/13 + 15/36 = 1.5929$.

The inequality $U \leq N(\sqrt{2} - 1)$ is therefore $1.5929 \leq 2(\sqrt{2} - 1) = 0.8284$ which does not hold. Because $U < 2$ and the test is inconclusive, we attempt RM-FF assignment.

Problem 19.1 from the book

Let's Practice

Exercise 1

The ITER Tokamak nuclear fusion reactor needs to maintain stable plasma within the re- actor's vacuum vessel where hydrogen atoms are fused to create clean energy! To do this, each section of the vacuum vessel is surrounded by an electromagnetic coil that needs to be continuosly controlled by a dedicated Plasma Control System (PCS). You are in charge of picking a processor to deploy the PCS software. The key software modules of the PCS are 4 periodic tasks: 1. Plasma Horizontal Stabilization (HS) which must run 50 times per sec- ond, 2. Plasma Vertical Stabilization (VS) which has a period of 13 ms, 3. Plasma Current Calculation (CC) which needs to run every 11 ms, and 4. Plasma Anomaly Detection (AD) with a period of 36 ms. The ITER is a very expensive one-of-a-kind machine and failure to keep the plasma stable inside the vacuum vessel can seriously damage the reactor.

Task ID	WCET (ms)	Period (ms)	Utilization (%)
HS	4	20	20%
VS	1.5	13	11.54%
CC	3	11	27.27%
AD	7.5	36	20.83%

Table 2: Task Parameters.

When applying RM-FF to the tasks in the considered order, we follow the steps beow.

Step 1: $U_{CC} = 0.5454 < 0.69 \rightarrow$ CC assigned to Proc 1.

Step 2: $U_{CC} + U_{HS} = 0.9454 > 2(2^{1/2} - 1) = 0.8284 \rightarrow$ HS does not fit on Proc 1.

Step 3: $U_{HS} = 0.4 < 0.69 \rightarrow$ HS assigned to Proc 2.

Step 4: $U_{CC} + U_{VS} = 0.7762 < 2(2^{1/2} - 1) = 0.8284 \rightarrow$ VS assigned to Proc 1.

Step 5: $U_{CC} + U_{VS} + U_{AD} = 1.1929 \rightarrow$ AD does not fit on Proc 1.

Step 6: $U_{HS} + U_{AD} = 0.8167 < 2(2^{1/2} - 1) = 0.8284 \rightarrow$ AD assigned to Proc 2.

RM-FF succeeds and produces the assignment below.

Proc 1: {CC, VS}; Proc 2: {HS, AD}.

Problem 19.1 from the book

Let's Practice

Exercise 2

A colleague of yours has just discovered the power of multi-threading. They have imple- mented the following concurrent threads (T1 and T2) to solve some mysterious problem of theirs. The two threads are executed concurrently on a single-processor system with an unknown scheduler.

```c
/* Shared Variables - START */
uint32_t result = 1;
uint8_t  lock = 0;
/* Shared Variables - END */
```

Listing 5: Global and shared variables definition.

```
Thread T1:
   int thread1_main (void) {
      [T1 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result + 10;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]

      printf("Result = %d\n", result);
   }
```

Listing 6: Code of T1.

```
Thread T2:
   int thread2_main (void) {
      [T2 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result * 4;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]
   }
   //
   //
```

Listing 7: Code of T2.

Problem 19.1 from the book

Let's Practice

Exercise 2

A colleague of yours has just discovered the power of multi-threading. They have imple- mented the following concurrent threads (T1 and T2) to solve some mysterious problem of theirs. The two threads are executed concurrently on a single-processor system with an unknown scheduler.

```c
/* Shared Variables - START */
uint32_t result = 1;
uint8_t  lock = 0;
/* Shared Variables - END */
```

Listing 5: Global and shared variables definition.

```
Thread T1:
   int thread1_main (void) {
      [T1 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result + 10;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]

      printf("Result = %d\n", result);
   }
```

Listing 6: Code of T1.

```
Thread T2:
   int thread2_main (void) {
      [T2 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result * 4;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]
   }
   //
   //
```

Listing 7: Code of T2.

(a) [5 points] Initially, your colleague has not implemented the `[T1 ENTRY SECTION]`, `[T2 ENTRY SECTION]`, and `[EXIT SECTION]`. You can consider all of them empty. Is it possible for the code to print the output "`Result = 4`"? If so, show the interleaving of code execution that leads to that output using the table below.

Step	Thread T1	Thread T2
1		
2		
3		
4		
5		
6		
7		

Problem 19.1 from the book

Let's Practice

Exercise 2

A colleague of yours has just discovered the power of multi-threading. They have imple- mented the following concurrent threads (T1 and T2) to solve some mysterious problem of theirs. The two threads are executed concurrently on a single-processor system with an unknown scheduler.

```c
/* Shared Variables - START */
uint32_t result = 1;
uint8_t  lock = 0;
/* Shared Variables - END */
```

Listing 5: Global and shared variables definition.

```c
Thread T1:
   int thread1_main (void) {
      [T1 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result + 10;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]

      printf("Result = %d\n", result);
   }
```

Listing 6: Code of T1.

```c
Thread T2:
   int thread2_main (void) {
      [T2 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result * 4;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]
   }
   //
   //
```

Listing 7: Code of T2.

Step	Thread T1	Thread T2
1		(5) if (result == 1)
2		(6) int loc = result * 4;
3		(7) result = loc;
4	(5) if (result == 1)	
5	(12) printf("Result = %d\n");	
6		
7		

Problem 19.1 from the book

Let's Practice

Exercise 2

A colleague of yours has just discovered the power of multi-threading. They have imple- mented the following concurrent threads (T1 and T2) to solve some mysterious problem of theirs. The two threads are executed concurrently on a single-processor system with an unknown scheduler.

```c
/* Shared Variables - START */
uint32_t result = 1;
uint8_t  lock = 0;
/* Shared Variables - END */
```

Listing 5: Global and shared variables definition.

```c
Thread T1:
   int thread1_main (void) {
      [T1 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result + 10;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]

      printf("Result = %d\n", result);
   }
```

Listing 6: Code of T1.

```c
Thread T2:
   int thread2_main (void) {
      [T2 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result * 4;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]
   }
   //
   //
```

Listing 7: Code of T2.

Question 2 With the same implementation as before, is it possible for the code to produce the output: "Result = 44"? If so, show the interleaving of code execution that leads to that output using the table below.

Problem 19.1 from the book

Let's Practice

Exercise 2

A colleague of yours has just discovered the power of multi-threading. They have imple- mented the following concurrent threads (T1 and T2) to solve some mysterious problem of theirs. The two threads are executed concurrently on a single-processor system with an unknown scheduler.

```c
/* Shared Variables - START */
uint32_t result = 1;
uint8_t  lock = 0;
/* Shared Variables - END */
```

Listing 5: Global and shared variables definition.

```c
Thread T1:
   int thread1_main (void) {
      [T1 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result + 10;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]

      printf("Result = %d\n", result);
   }
```

Listing 6: Code of T1.

```c
Thread T2:
   int thread2_main (void) {
      [T2 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result * 4;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]
   }
   //
   //
```

Listing 7: Code of T2.

Step	Thread T1	Thread T2
1		(5) if (result == 1)
2	(5) if (result == 1)	
3	(6) int loc = result + 10;	
4	(7) result = loc;	
5		(6) int loc = result * 4;
6		(7) result = loc;
7	(12) printf("Result = %d\n");	

Problem 19.1 from the book

Let's Practice

Exercise 2

A colleague of yours has just discovered the power of multi-threading. They have imple- mented the following concurrent threads (T1 and T2) to solve some mysterious problem of theirs. The two threads are executed concurrently on a single-processor system with an unknown scheduler.

```c
/* Shared Variables - START */
uint32_t result = 1;
uint8_t  lock = 0;
/* Shared Variables - END */
```

Listing 5: Global and shared variables definition.

```c
Thread T1:
   int thread1_main (void) {
      [T1 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result + 10;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]

      printf("Result = %d\n", result);
   }
```

Listing 6: Code of T1.

```c
Thread T2:
   int thread2_main (void) {
      [T2 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result * 4;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]
   }
   //
   //
```

Listing 7: Code of T2.

(c) [8 points] After running the code many times and achieving very inconsistent results, your colleague has finally decided to implement their own protocol for mutual exclusion. Listing 8 reports the implementation for `[T1 ENTRY SECTION]`; Listing 9 reports the implementation for `[T2 ENTRY SECTION]`; and Listing 10 reports the implementation for the common `[EXIT SECTION]`;

It seems however that sometimes thread T2 remains stuck. Can you show in the provided table a possible execution interleaving that illustrates the issue? *NOTE: No need to use all the rows in the table if you can illustrate the problem with less than 12 rows.*

```c
lock = 1;
while (true) {
   if (lock == 1)
      break;
}
```

Listing 8: T1 Entry Section

```c
lock = 2;
while (true) {
   if (lock == 2)
      break;
}
```

Listing 9: T2 Entry Section

```c
lock = 0;
```

Listing 10: Common Exit Section

Problem 19.1 from the book

Let's Practice

Exercise 2

A colleague of yours has just discovered the power of multi-threading. They have imple- mented the following concurrent threads (T1 and T2) to solve some mysterious problem of theirs. The two threads are executed concurrently on a single-processor system with an unknown scheduler.

```c
/* Shared Variables - START */
uint32_t result = 1;
uint8_t  lock = 0;
/* Shared Variables - END */
```

Listing 5: Global and shared variables definition.

```c
Thread T1:
   int thread1_main (void) {
      [T1 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result + 10;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]

      printf("Result = %d\n", result);
   }
```

Listing 6: Code of T1.

```c
Thread T2:
   int thread2_main (void) {
      [T2 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result * 4;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]
   }
   //
   //
```

Listing 7: Code of T2.

(c) [8 points] After running the code many times and achieving very inconsistent results, your colleague has finally decided to implement their own protocol for mutual exclusion. Listing 8 reports the implementation for `[T1 ENTRY SECTION]`; Listing 9 reports the implementation for `[T2 ENTRY SECTION]`; and Listing 10 reports the implementation for the common `[EXIT SECTION]`;

It seems however that sometimes thread T2 remains stuck. Can you show in the provided table a possible execution interleaving that illustrates the issue? *NOTE: No need to use all the rows in the table if you can illustrate the problem with less than 12 rows.*

```c
lock = 1;
while (true) {
   if (lock == 1)
      break;
}
```

Listing 8: T1 Entry Section

```c
lock = 2;
while (true) {
   if (lock == 2)
      break;
}
```

Listing 9: T2 Entry Section

```c
lock = 0;
```

Listing 10: Common Exit Section

Problem 19.1 from the book

Let's Practice

Exercise 2

A colleague of yours has just discovered the power of multi-threading. They have imple- mented the following concurrent threads (T1 and T2) to solve some mysterious problem of theirs. The two threads are executed concurrently on a single-processor system with an unknown scheduler.

```
/* Shared Variables - START */
uint32_t result = 1;
uint8_t  lock = 0;
/* Shared Variables - END */
```

Listing 5: Global and shared variables definition.

```
Thread T1:
   int thread1_main (void) {
      [T1 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result + 10;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]

      printf("Result = %d\n", result);
   }
```

Listing 6: Code of T1.

```
Thread T2:
   int thread2_main (void) {
      [T2 ENTRY SECTION]
      /* Critical section begin */
      if (result == 1) {
         int loc = result * 4;
         result = loc;
      }
      /* Critical section end */
      [EXIT SECTION]
   }
   //
   //
```

Listing 7: Code of T2.

(c) [8 points] After running the code many times and achieving very inconsistent results, your colleague has finally decided to implement their own protocol for mutual exclusion. Listing 8 reports the implementation for `[T1 ENTRY SECTION]`; Listing 9 reports the implementation for `[T2 ENTRY SECTION]`; and Listing 10 reports the implementation for the common `[EXIT SECTION]`;

It seems however that sometimes thread T2 remains stuck. Can you show in the provided table a possible execution interleaving that illustrates the issue? *NOTE: No need to use all the rows in the table if you can illustrate the problem with less than 12 rows.*

```
lock = 1;
while (true) {
   if (lock == 1)
      break;
}
```

Listing 8: T1 Entry Section

```
lock = 2;
while (true) {
   if (lock == 2)
      break;
}
```

Listing 9: T2 Entry Section

```
lock = 0;
```

Listing 10: Common Exit Section

Step	Thread T1	Thread T2
1		`lock = 2;`
2	`lock = 1;`	
	...	...

Step	Thread T1	Thread T2
1		`lock = 2;`
2		`while (true)`
3	`lock = 1;`	
	...	...

Problem 19.1 from the book