

Assignment No. 1

Real time system

SUBMITTED BY:

NAME: MUKUND VISHWAS CHAVAN

STUDENT ID: 01011

STUDENT E-MAIL: 2024HT01011@WILP.BITS-PILANI.AC.IN

PART A

Question 1: Rate Monotonic Scheduling (RMS) Analysis

Calculate the CPU utilization for each task and determine if the set is schedulable using RMS.

Given task set:

- **T1:** Period = 4 ms, Execution time = 1 ms >> **Utilization UT1 = 0.25**
- **T2:** Period = 5 ms, Execution time = 2 ms >> **Utilization UT2 = 0.4**
- **T3:** Period = 10 ms, Execution time = 3 ms >> **Utilization UT3 = 0.3**

Total CPU Utilization

Total Utilization $U = UT1 + UT2 + UT3 = 0.25 + 0.4 + 0.3 = 0.95$

RMS Utilization Bound

Handwritten calculation of the RMS Utilization Bound:

$$\begin{aligned} \text{RMS Utilization Bound: } & \\ U\text{-Bound} &= n (2^{1/n} - 1) \\ \text{here } n &\text{ is the number of task ie. } 3. \\ U\text{-Bound} &= 3 (2^{1/3} - 1) = \\ &= 3 (1.259 - 1) \\ \underline{U\text{-Bound}} &= \underline{0.7797} \end{aligned}$$

Comparison

- **Total Utilization:** 0.95
- **RMS Bound:** 0.779763

Since the total CPU utilization (0.95) is greater than the RMS bound (0.779763), the task set is **not schedulable** using RMS.

This means under worst-case conditions one or more tasks may miss their deadlines.

Question 2: Earliest Deadline First (EDF) Scheduling

Calculate the response time for each task and determine if they are schedulable under EDF.

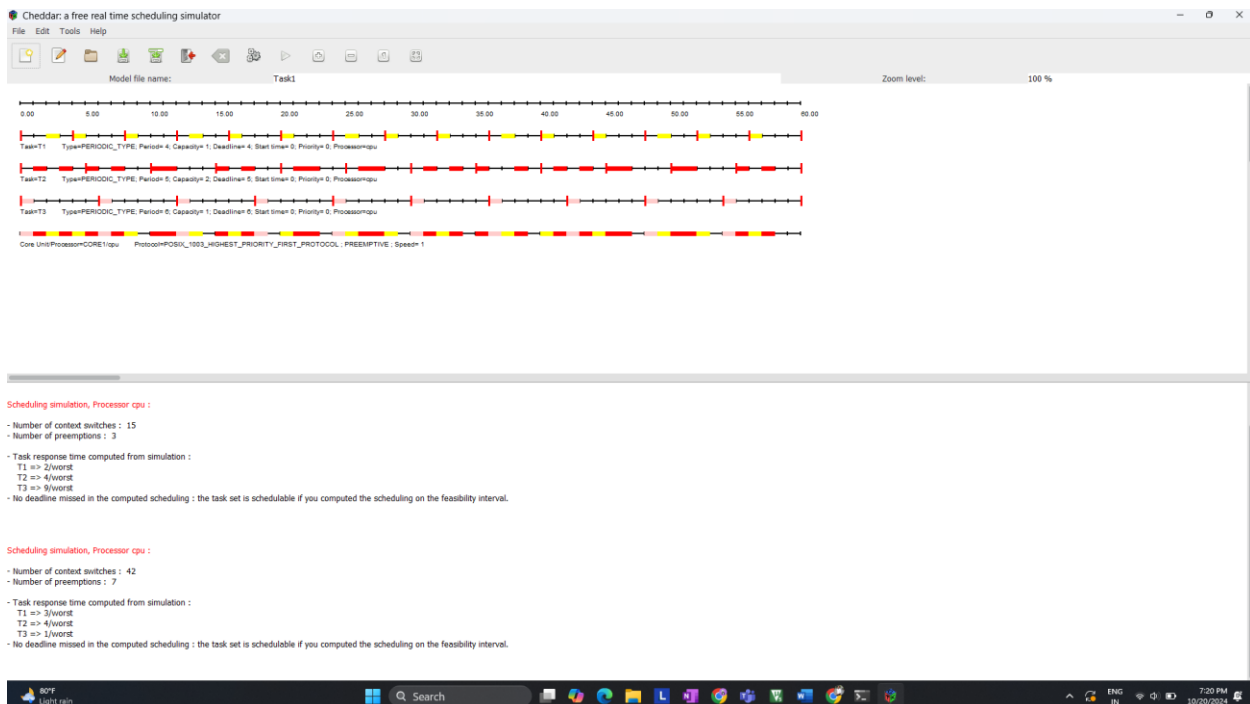
Task Set Description:

In this task set, we are given three periodic tasks, each with its own period, execution time, and deadline:

Task	Period (ms)	Execution Time (ms)	Deadline (ms)
T1	4	1	4
T2	5	2	5
T3	6	1	6

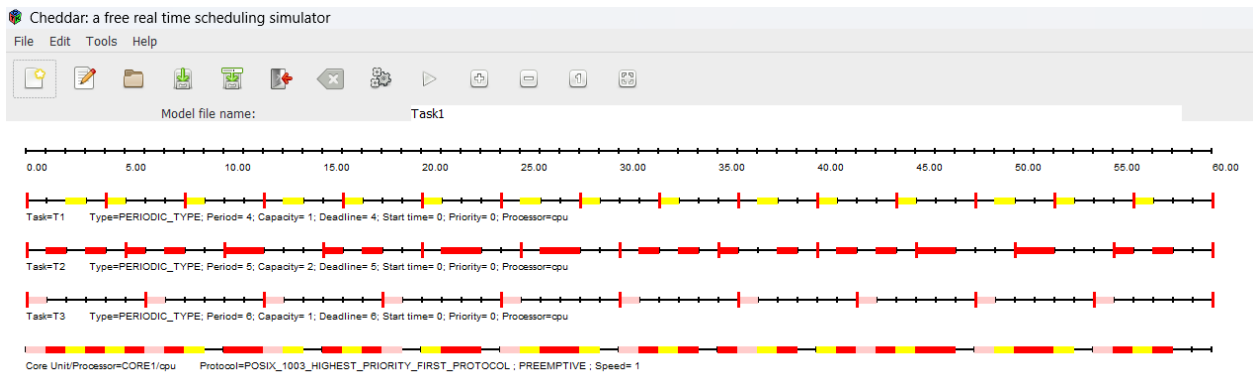
Cheddar real-time scheduling simulator to schedule the task set using EDF.

The task parameters (period, execution time, and deadlines) were input into the simulator, and the processor was configured to use the **Earliest Deadline First (EDF)** scheduling policy. The simulation was run over a hyperperiod of 60 ms (the least common multiple of the task periods).



Gantt Chart:

Below is the Gantt chart generated by the Cheddar simulator for the EDF scheduling:



The Gantt chart shows how the tasks are scheduled overtime based on their deadlines:

- **Task T1:** Since it has the shortest period (4 ms), it is scheduled frequently and completes its execution before its 4 ms deadline.
- **Task T2:** T2 is scheduled less frequently than T1 but still completes its execution before its 5 ms deadline.
- **Task T3:** T3 has the longest period (6 ms) but is scheduled in between T1 and T2 as per its deadline.

Response Time and Schedulability:

Scheduling simulation, Processor cpu :

- Number of context switches : 15
- Number of preemptions : 3
- Task response time computed from simulation :
 - T1 => 2/worst
 - T2 => 4/worst
 - T3 => 9/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

Scheduling simulation, Processor cpu :

- Number of context switches : 42
- Number of preemptions : 7
- Task response time computed from simulation :
 - T1 => 3/worst
 - T2 => 4/worst
 - T3 => 1/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

The simulator calculated the worst-case response times for each task. The response time is defined as the time from the task's release to the time when it completes its execution. The worst-case response times observed are as follows:

Task	Worst-Case Response Time (ms)
T1	3
T2	4
T3	9 (in one run) / 1 (in another run)

- **T1:** The worst-case response time for T1 is 3 ms, which is less than its deadline of 4 ms, meaning T1 meets its deadline.
- **T2:** T2's worst-case response time is 4 ms, less than its deadline of 5 ms, so it also meets its deadline.
- **T3:** T3's response time varies between 9 ms and 1 ms, depending on the scheduling scenario, but it still meets its deadline of 6 ms in both cases.

No deadlines were missed in the computed scheduling, confirming that the task set is schedulable under EDF.

The task set was successfully scheduled under EDF without missing any deadlines, as confirmed by the Gantt chart and the response time analysis. Despite the dynamic nature of EDF, which results in context switches and preemptions, the scheduling algorithm was able to meet all task deadlines, proving that the task set is **schedulable under EDF**.

Question 3 Calculate the worst-case response time for each task using a priority scheduling algorithm.

Given Task Set:

Task	Priority (Higher number = Lower priority)	Period (ms)	Execution Time (ms)	Response Time (ms)
T1	1 (Highest Priority)	6	2	2
T2	2	8	3	5
T3	3 (Lowest Priority)	10	1	6

- **T1:** Priority = 1 (Highest), Period = 6 ms, Execution Time = 2 ms
- **T2:** Priority = 2, Period = 8 ms, Execution Time = 3 ms
- **T3:** Priority = 3 (Lowest), Period = 10 ms, Execution Time = 1 ms

Worst-Case Response Time Calculation:

Task T1 (Highest Priority):

- **Execution Time:** 2 ms
- **Blocking Time:** 0 ms (No higher priority tasks)
- **Response Time:** $R1=C1=2$ ms

Task T2:

- **Execution Time:** 3 ms
- **Interference from Higher Priority Task (T1):** 1 instance of T1's execution time during T2's period (2 ms)
- **Response Time:** $R2=C2+R1=3+2=5$ ms

Task T3 (Lowest Priority):

- **Execution Time:** 1 ms
- **Interference from Higher Priority Tasks (T1 and T2):**
 - T1: 2 instances of T1's execution time during T3's period (2×2 ms = 4 ms)
 - T2: 1 instance of T2's execution time during T3's period (3 ms)
- **Response Time:** $R3=C3+R1+R2=1+2+3=6$ ms

Final Worst-Case Response Times:

Task	Priority	Period (ms)	Execution Time (ms)	Response Time (ms)
T1	1	6	2	2
T2	2	8	3	5
T3	3	10	1	6

- T1 completes its execution in 2 ms, which is less than its period of 6 ms.
- T2 completes its execution in 5 ms, which is less than its period of 8 ms.
- T3 completes its execution in 6 ms, which is less than its period of 10 ms.

Conclusion:

- T1: Meets its deadline.
- T2: Meets its deadline.
- T3: Meets its deadline.

Since all tasks complete within their periods, the task set is **schedulable** using the given priority scheduling.

Question 4: Context Switching Impact.

Given Task Set:

Task	Period (ms)	Execution Time (ms)	Context Switches	Effective Execution Time (ms)
T1	5	2	1	3
T2	10	3	1	4
T3	15	4	2	6

calculate the **effective execution time** for each task, which is the task's execution time plus the overhead caused by context switching.

For each task, the **effective execution time** is the sum of the task's **execution time** and the product of the **context switch time** and the number of **context switches**.

Given that the context switch time is **1 ms**, the formula becomes:

Effective Execution Time = Execution Time + (Context Switches × Context Switch Time)

Where the **Context Switch Time** is **1 ms**.

Calculation:

1. For Task T1:

- Execution Time = 2 ms
- Context Switches = 1
- Effective Execution Time:

$$\text{Effective Execution Time} = 2 + (1 \times 1) = 2 + 1 = 3 \text{ ms}$$

2. For Task T2:

- Execution Time = 3 ms
- Context Switches = 1
- Effective Execution Time:

$$\text{Effective Execution Time} = 3 + (1 \times 1) = 3 + 1 = 4 \text{ ms}$$

3. For Task T3:

- Execution Time = 4 ms
- Context Switches = 2
- Effective Execution Time:

$$\text{Effective Execution Time} = 4 + (2 \times 1) = 4 + 2 = 6 \text{ ms}$$

- The **effective execution time** for each task is the sum of the actual execution time and the overhead caused by context switching.
- **T1** has an effective execution time of **3 ms**, **T2** has **4 ms**, and **T3** has **6 ms**, considering the 1 ms context switch overhead for each context switch.

Question 5: Missed Deadlines in a Hyperperiod.

Given Task Set:

Task	Period (ms)	Execution Time (ms)	Number of Deadlines in Hyperperiod (30 ms)	Deadlines Missed
T1	6	2	?	?
T2	10	3	?	?
T3	12	4	?	?

We need to calculate the **number of deadlines missed** for each task in a hyperperiod of **30 ms**.

Step 1: Number of Deadlines in the Hyperperiod:

The **number of deadlines** for each task within the **30 ms hyperperiod** can be calculated by dividing the hyperperiod by the task's period.

Number of Deadlines=Hyperperiod/Task Period

1. For Task T1:

- Period = 6 ms
- Hyperperiod = 30 ms

Number of Deadlines for T1 = $30 / 6 = 5$

2. For Task T2:

- Period = 10 ms
- Hyperperiod = 30 ms

Number of Deadlines for T2 = $30 / 10 = 3$

3. For Task T3:

- Period = 12 ms
- Hyperperiod = 30 ms

Number of Deadlines for T3 = $30 / 12 = 2.5 \Rightarrow 2$ deadlines

Step 2: Check for Missed Deadlines:

We will check whether any deadlines are missed by considering whether the task can finish execution before its next deadline, assuming no preemption for this calculation.

1. For Task T1:

- Execution Time = 2 ms

- Since T1 has a period of 6 ms and an execution time of 2 ms, it will easily meet each of its 5 deadlines in the 30 ms hyperperiod.
- **Deadlines Missed = 0**

2. For Task T2:

- Execution Time = 3 ms
- T2 has a period of 10 ms and an execution time of 3 ms. It will also meet all 3 of its deadlines within its periods without missing any deadlines.
- **Deadlines Missed = 0**

3. For Task T3:

- Execution Time = 4 ms
- T3 has a period of 12 ms and an execution time of 4 ms. Since it completes execution before each deadline within its period, it will meet both of its deadlines within the 30 ms hyperperiod.
- **Deadlines Missed = 0**

Final Table:

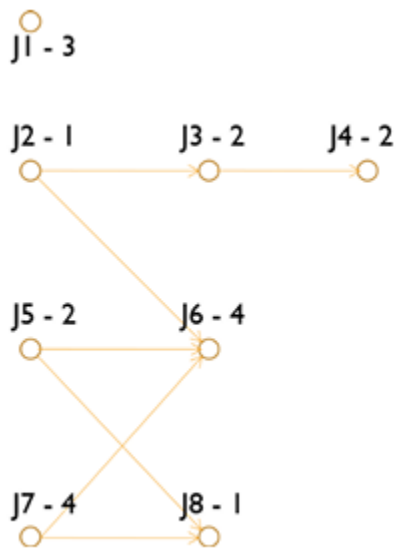
Task	Period (ms)	Execution Time (ms)	Number of Deadlines in Hyperperiod (30 ms)	Deadlines Missed
T1	6	2	5	0
T2	10	3	3	0
T3	12	4	2	0

- **No deadlines are missed** for any of the tasks in the 30 ms hyperperiod.
- All tasks complete their execution within their periods, ensuring they meet their deadlines.

PART B

Question 1 and set up the tasks in **Cheddar** step by step for both **non-preemptive** and **preemptive priority-driven scheduling**.

Task Graph:



The task graph includes 8 tasks (**J1 to J8**) with the following characteristics:

- We have eight aperiodic tasks (**J1 to J8**) with a **deadline of 12 ms**.
- All tasks except **J5** are released at **0 ms**, while **J5** is released at **4 ms**.
- Tasks have descending priority from **J1 (highest)** to **J8 (lowest)**.
- The scheduling is done on a processor with **dual identical cores**, and jobs can migrate between cores at any time.
- The goal is to compare **non-preemptive priority-driven scheduling** and **preemptive priority-driven scheduling** and determine which performs better.

Core and Processor Configuration:

I first set up the cores and then assigned them to the processor. Below are the detailed steps with screenshots.

Step 1: Core Setup

1. Core1 Configuration:

- Name: CORE1
- Scheduler Type: Posix 1003 Highest Priority First Protocol
- Preemptive Type: Set to Non-Preemptive
- Quantum: 0
- Speed: 1

Name	Scheduler	Quanti	Preemptive	Automat	Capacity	Period	Priority	Use
CORE1	Posix 1003 Highest Priority First Protocol	0	Not Preemptive		0	0	0	
CORE2	Posix 1003 Highest Priority First Protocol	0	Not Preemptive		0	0	0	

2. Core2 Configuration:

- Name: CORE2
- Scheduler Type: Same as CORE1: Posix 1003 Highest Priority First Protocol
- Preemptive Type: Set to Non-Preemptive
- Quantum: 0
- Speed: 1

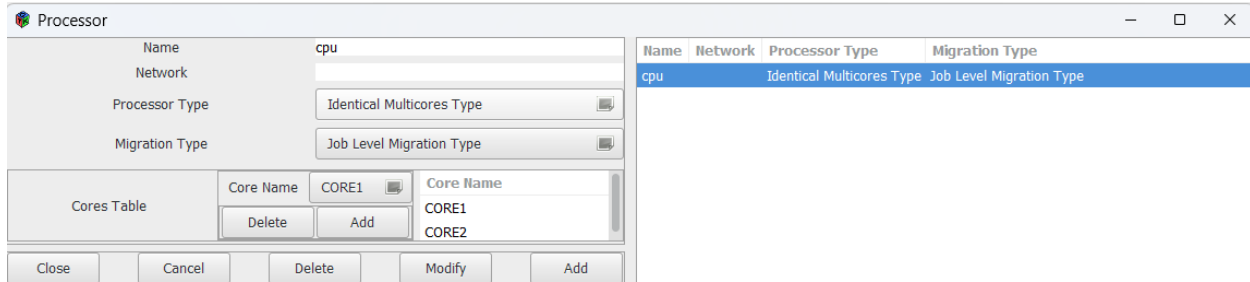
Name	Scheduler	Quanti	Preemptive	Automat	Capacity	Period	Priority	Use
CORE1	Posix 1003 Highest Priority First Protocol	0	Not Preemptive		0	0	0	
CORE2	Posix 1003 Highest Priority First Protocol	0	Not Preemptive		0	0	0	

Step 2: Processor Setup

Once both cores are created, we configure the processor to manage these cores.

- Processor Name: cpu
- Processor Type: Set to Identical Multicores Type (dual-core).
- Migration Type: Set to Job Level Migration Type (allows jobs to migrate between the two cores).

- Cores Added: Both CORE1 and CORE2 were added to the processor.

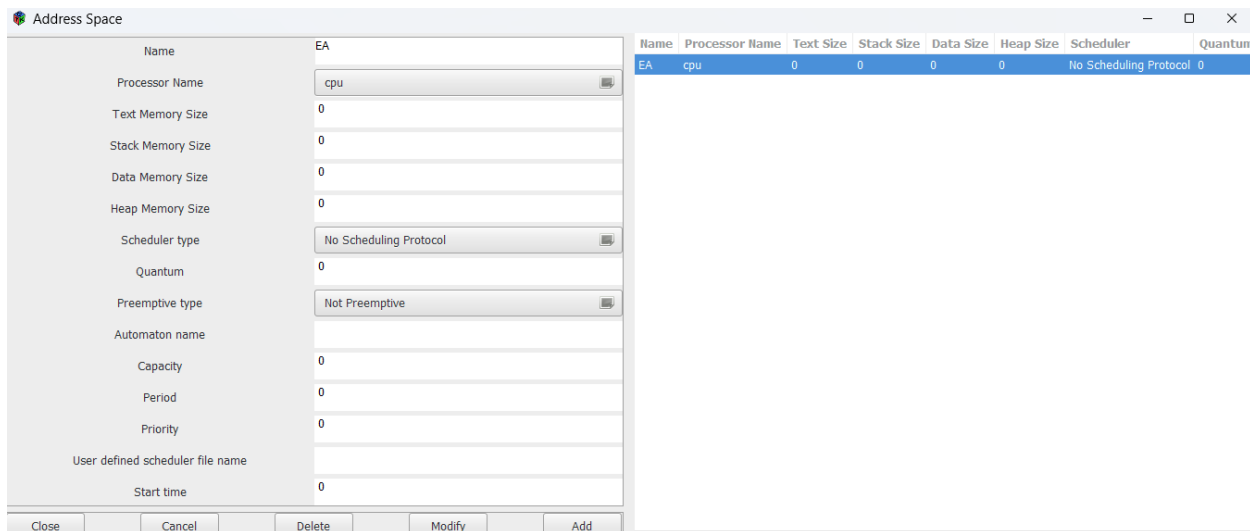


This setup ensures that tasks can run on dual cores with non-preemptive scheduling and can migrate between cores if necessary.

Step 3: Address Space Configuration:

I have configured the address space for the processor. The current configuration is as follows:

1. Processor Name: cpu
2. Scheduler Type: Set to No Scheduling Protocol since no specific memory-related scheduling is required for this task.
3. Preemptive Type: Since we are using Non-Preemptive Scheduling for this part of the task, ensure the Preemptive Type is set to Not Preemptive (to match the core settings).



This configuration ensures that the address space settings are aligned with the processor's non-preemptive priority scheduling.

Step 4: Task Setup

In this step, the tasks J1 to J8 were configured with the following parameters:

- Task Type: Aperiodic
- Processor: CPU
- Address Space: EA
- Priority Levels: Based on their respective dependencies (J1 with the highest priority, J8 with the lowest)
- Policy: FIFO (First-In, First-Out)
- Capacity and Deadlines: The capacity and deadline values were set based on the task graph provided, ensuring the deadlines were achievable.

Tasks were configured using the FIFO scheduling policy to avoid the invalid priority error encountered with other policies.

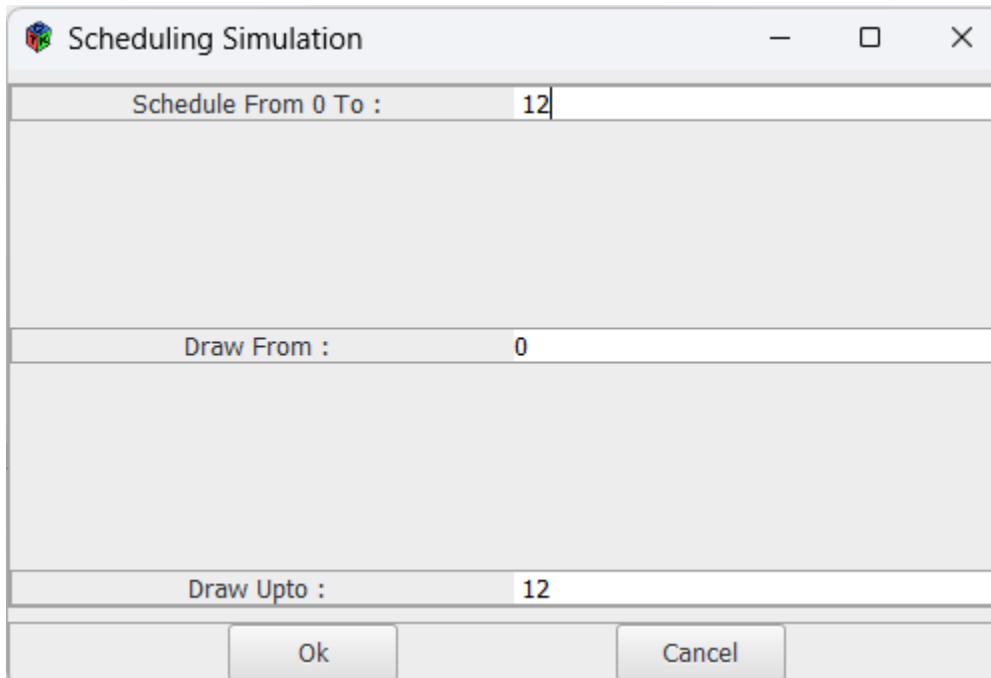
The screenshot shows the 'Task' configuration window. On the left, the configuration for task J1 is displayed. On the right, a table lists the parameters for tasks J1 through J8.

Name	Task Type	Processor	Address Space	Core	Capacity	Deadline	Start time	Priority	Blocking Time
J1	Aperiodic	cpu	EA		3	12	0	0	0
J2	Aperiodic	cpu	EA		1	12	0	1	0
J3	Aperiodic	cpu	EA		2	12	0	2	0
J4	Aperiodic	cpu	EA		2	12	0	3	0
J5	Aperiodic	cpu	EA		2	12	4	4	0
J6	Aperiodic	cpu	EA		4	12	0	5	0
J7	Aperiodic	cpu	EA		4	12	0	6	0
J8	Aperiodic	cpu	EA		1	12	0	7	0

Name	Task Type	Processor	Address Space	Core	Capacity	Deadline	Start time	Priority	Blocking Time
J1	Aperiodic	cpu	EA		3	12	0	0	0
J2	Aperiodic	cpu	EA		1	12	0	1	0
J3	Aperiodic	cpu	EA		2	12	0	2	0
J4	Aperiodic	cpu	EA		2	12	0	3	0
J5	Aperiodic	cpu	EA		2	12	4	4	0
J6	Aperiodic	cpu	EA		4	12	0	5	0
J7	Aperiodic	cpu	EA		4	12	0	6	0
J8	Aperiodic	cpu	EA		1	12	0	7	0

Step 5: Running the Scheduling Simulation

1. Scheduling Setup: Before running the simulation, we defined the time interval for scheduling (in this case, from 0 to 12).

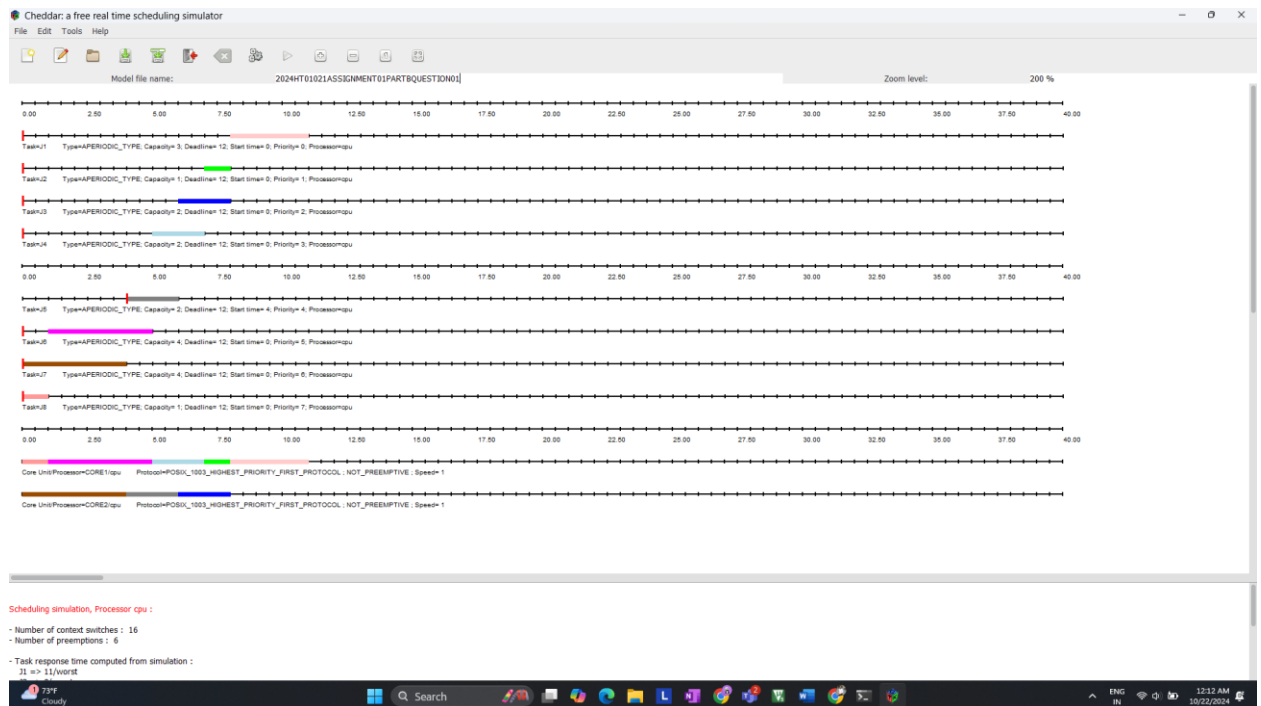
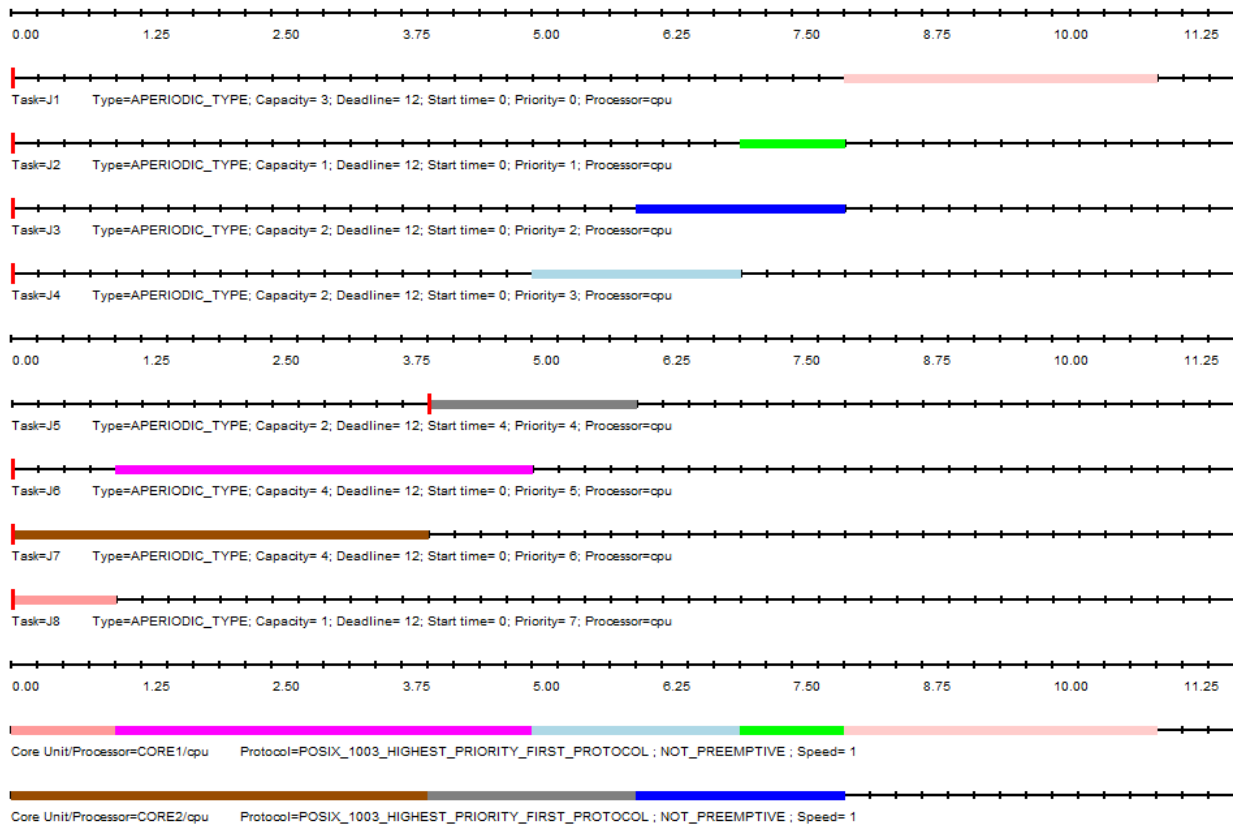


The image shows a window titled "Scheduling Simulation" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains three input fields, each with a label and a text box. The first field is labeled "Schedule From 0 To :" and has the value "12" entered. The second field is labeled "Draw From :" and has the value "0" entered. The third field is labeled "Draw Upto :" and has the value "12" entered. At the bottom of the window, there are two buttons: "Ok" and "Cancel".

2. Executing the Simulation: The Cheddar simulation tool was used to analyze task scheduling, showing the tasks distributed across the two cores (Core1 and Core2). The results of the simulation were displayed in the form of task Gantt charts for each task.

Model file name:

2024HT01021ASSIGNMENT01PARTBQUESTION01



3. Results and Analysis:

- Number of context switches: 16
- Number of preemptions: 6
- Worst-case response times for each task were recorded. These results showed that no deadlines were missed, indicating that the task set is schedulable.

Scheduling simulation, Processor cpu :

- Number of context switches : 16
- Number of preemptions : 6

- Task response time computed from simulation :

J1 => 11/worst
J2 => 8/worst
J3 => 8/worst
J4 => 7/worst
J5 => 2/worst
J6 => 5/worst
J7 => 4/worst
J8 => 1/worst

- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

Step 6: Analyze and Interpret the Results

1. **Gantt Chart Review:** After running the simulation, review the Gantt chart to ensure the tasks are scheduled without missing deadlines. Tasks should be divided appropriately between **CORE1** and **CORE2**.
2. **Response Time Analysis:** Analyze the worst-case response times of each task:
 - J1: 11ms
 - J2: 8ms
 - J3: 8ms
 - J4: 7ms
 - etc.

No deadlines were missed in this simulation.

3. **Conclusion:** Conclude that the system is schedulable within the given deadline using the **FIFO policy** and non-preemptive protocol. Suggest improvements, such as exploring different scheduling policies or priority adjustments.

Here is the GitHub link for the above solution:

https://github.com/mukundchavan251/BITS/blob/main/Real_Time_Systems/Assignments/Q1/Cheddar/2024HT01011AssignmentNo1_Q1

Question 3

Show that the given task set is schedulable using **LLF (Least Laxity First)** but not using **EDF (Earliest Deadline First)** on a dual-core processor. All tasks are aperiodic.

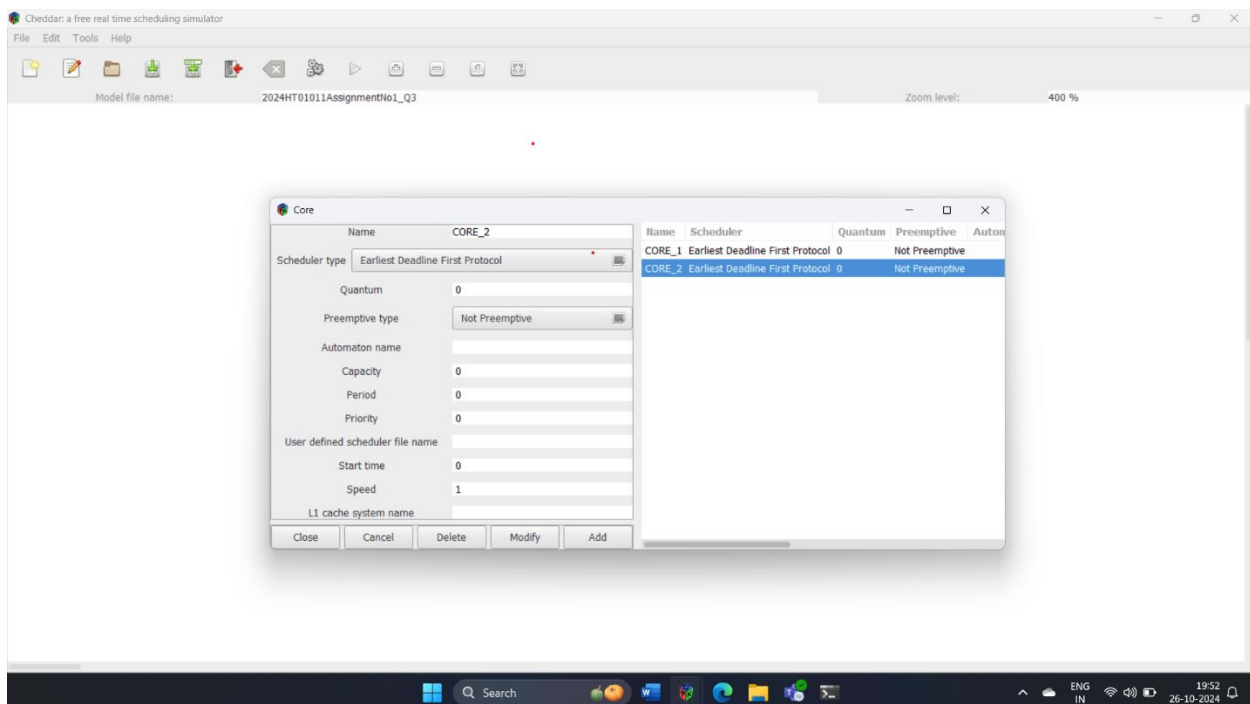
The task parameters are as follows:

	J1	J2	J3
r	0	0	0
e	1	1	5
d	1	2	5

Step 1: Processor and Core Setup

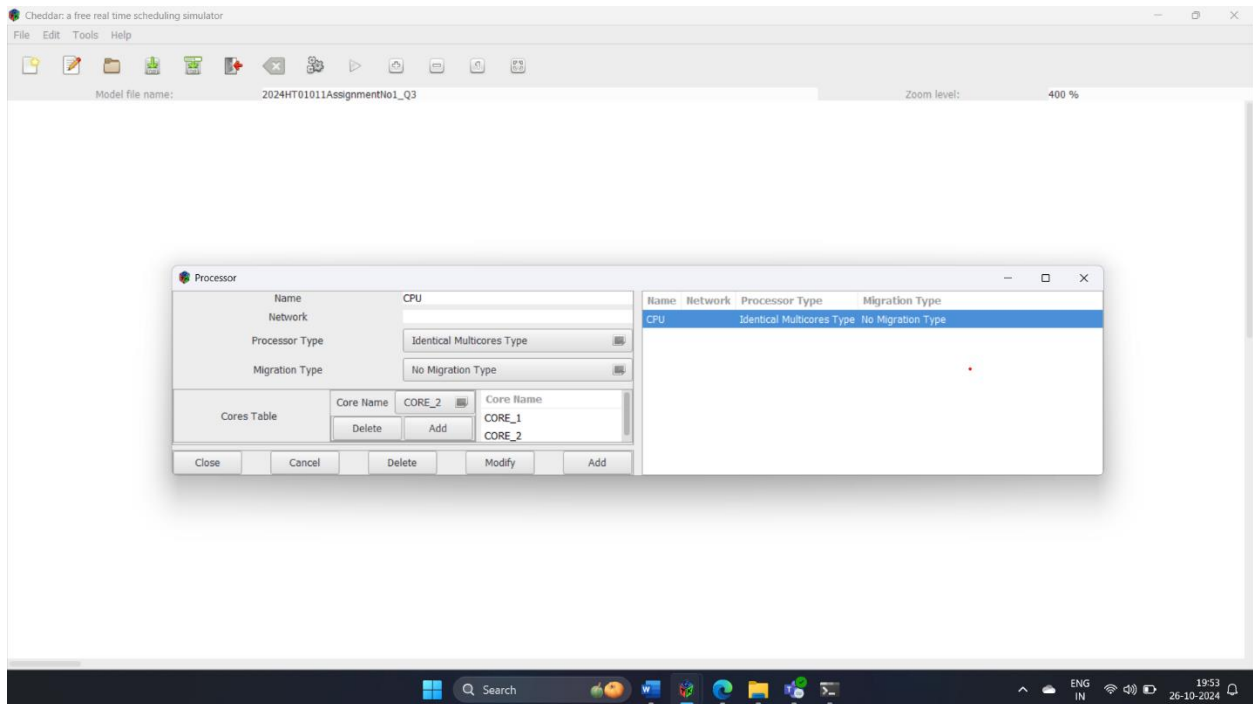
1. Core Setup:

- Both **Core 1** and **Core 2** are initially configured to use **Earliest Deadline First (EDF)** for this part of the simulation.



2. Processor Setup

- **Processor Type:** Identical Multicore Type
- **Migration Type:** No Migration

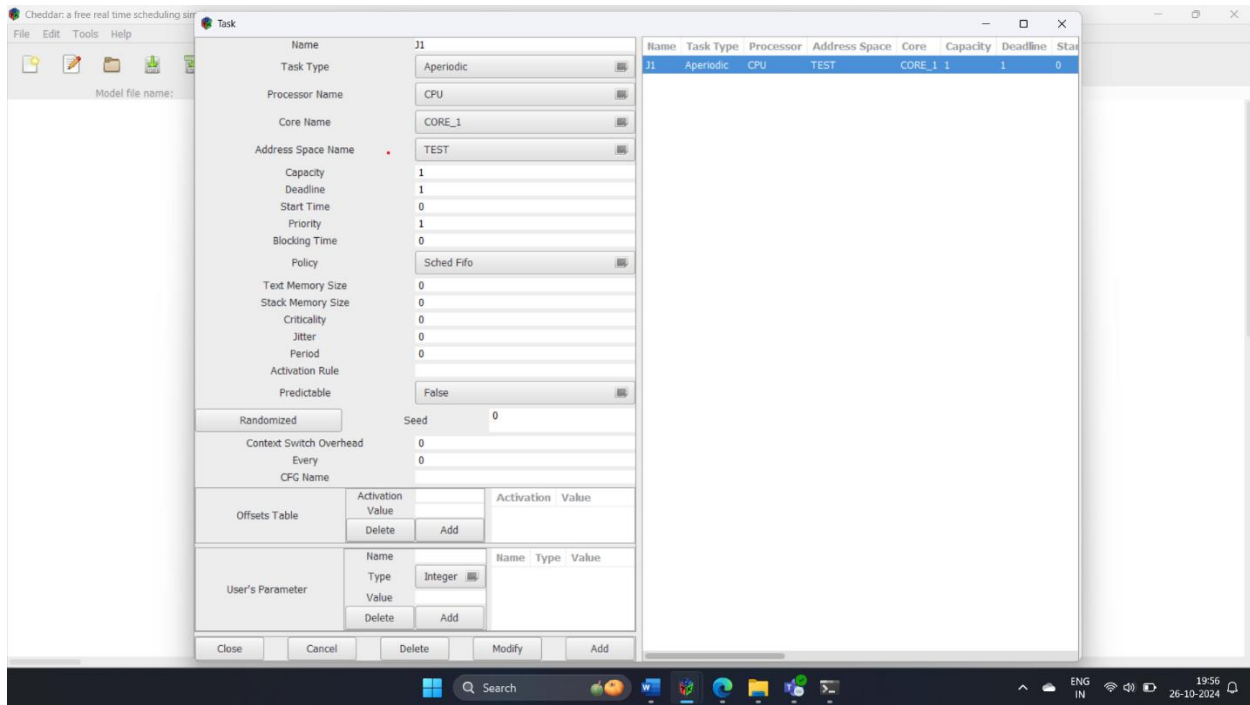


Step 2: Task Setup

Now, configure the tasks **J1**, **J2**, and **J3** in the Cheddar tool.

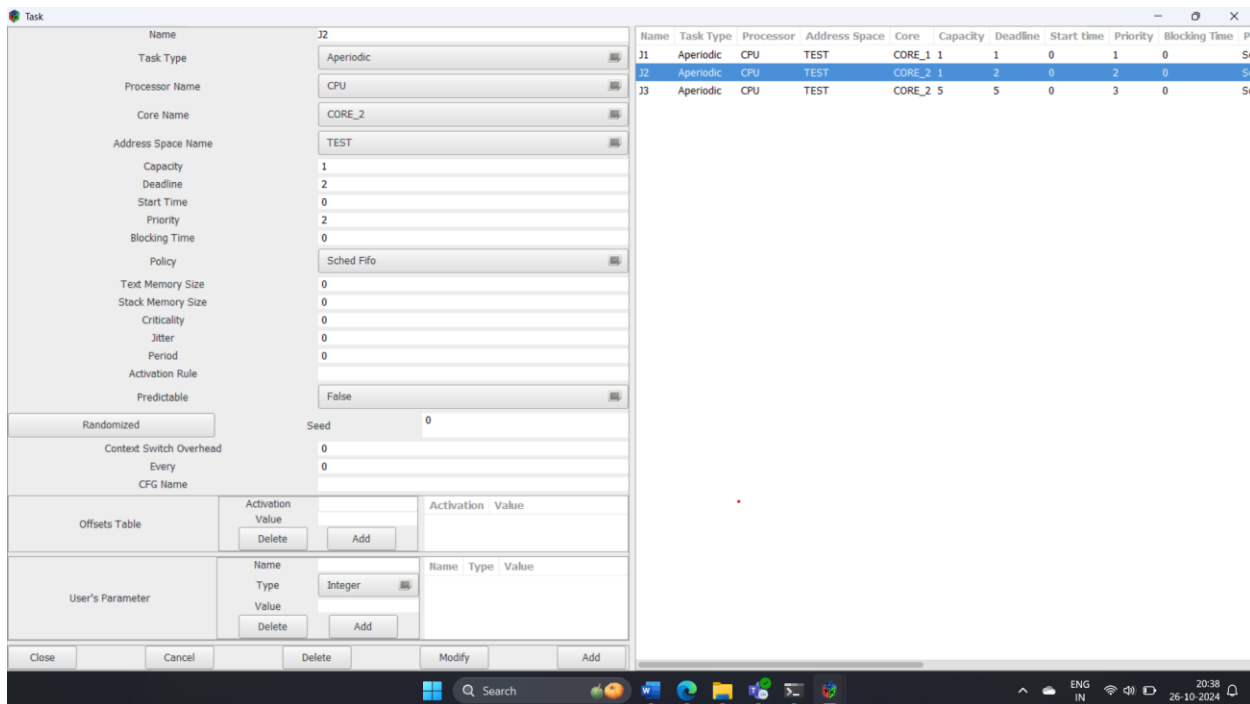
Task Configuration:

1. **Task J1:**
 - **Arrival Time (r):** 0
 - **Execution Time (e):** 1
 - **Deadline (d):** 1



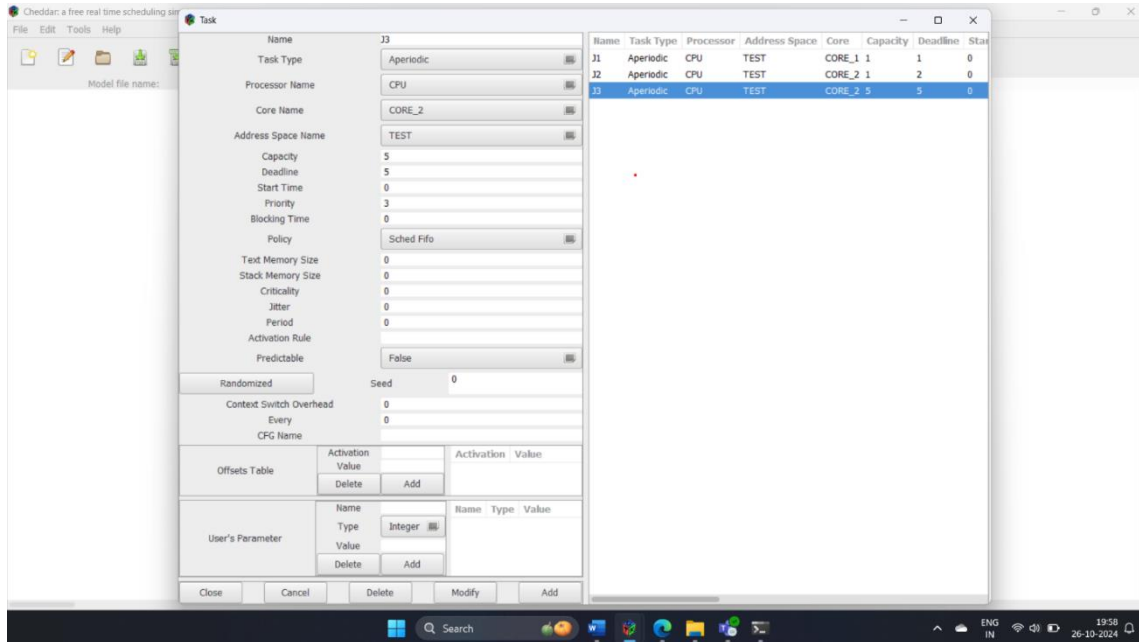
2. Task J2:

- **Arrival Time (r): 0**
- **Execution Time (e): 1**
- **Deadline (d): 2**



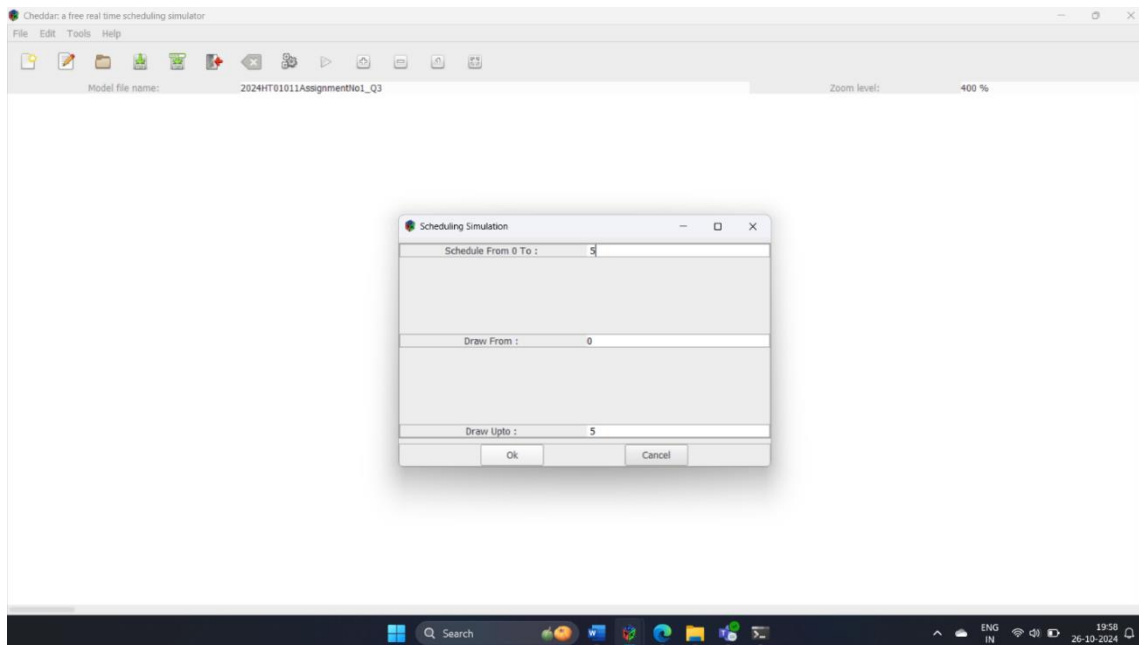
3. Task J3:

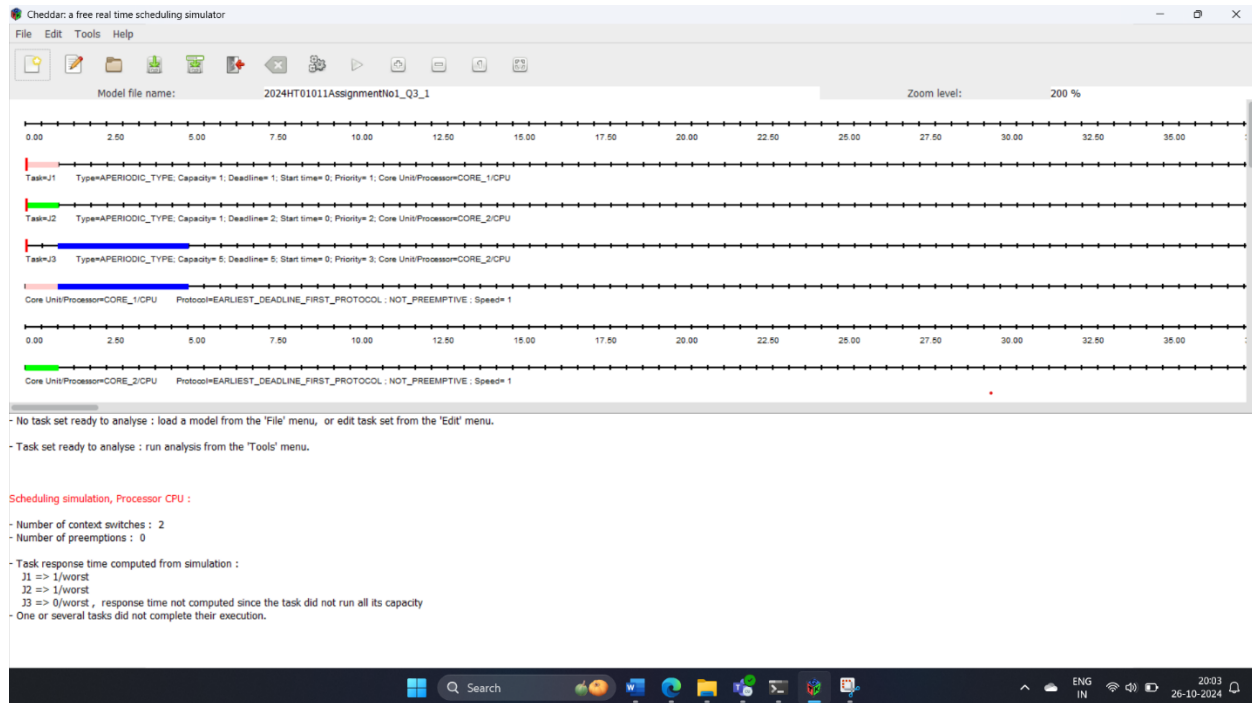
- **Arrival Time (r): 0**
- **Execution Time (e): 5**
- **Deadline (d): 5**



Step 3: Simulation with EDF Scheduler

After the core and task configuration, run the scheduling simulation using **Earliest Deadline First (EDF)** scheduling to determine whether the task set is schedulable.





Result:

The simulation output for EDF (Earliest Deadline First) scheduling shows that one or more tasks did not complete their execution. Specifically, Task J3 could not run to its full capacity, which is reflected in the following details:

1. **Number of Context Switches:** 2
2. **Number of Preemptions:** 0
3. **Task Response Time:**
 - J1: 1 (worst)
 - J2: 1 (worst)
 - J3: 0 (worst) — Task J3 did not complete execution.

Conclusion:

This means that under EDF scheduling with the given configuration, Task J3 could not meet its deadline, which leads to the conclusion that the system is not schedulable under EDF in this setup.

Step 4: Change to LLF Scheduler

Next, change the scheduler to **LLF (Least Laxity First)** on both cores to verify whether the task set is schedulable under LLF.

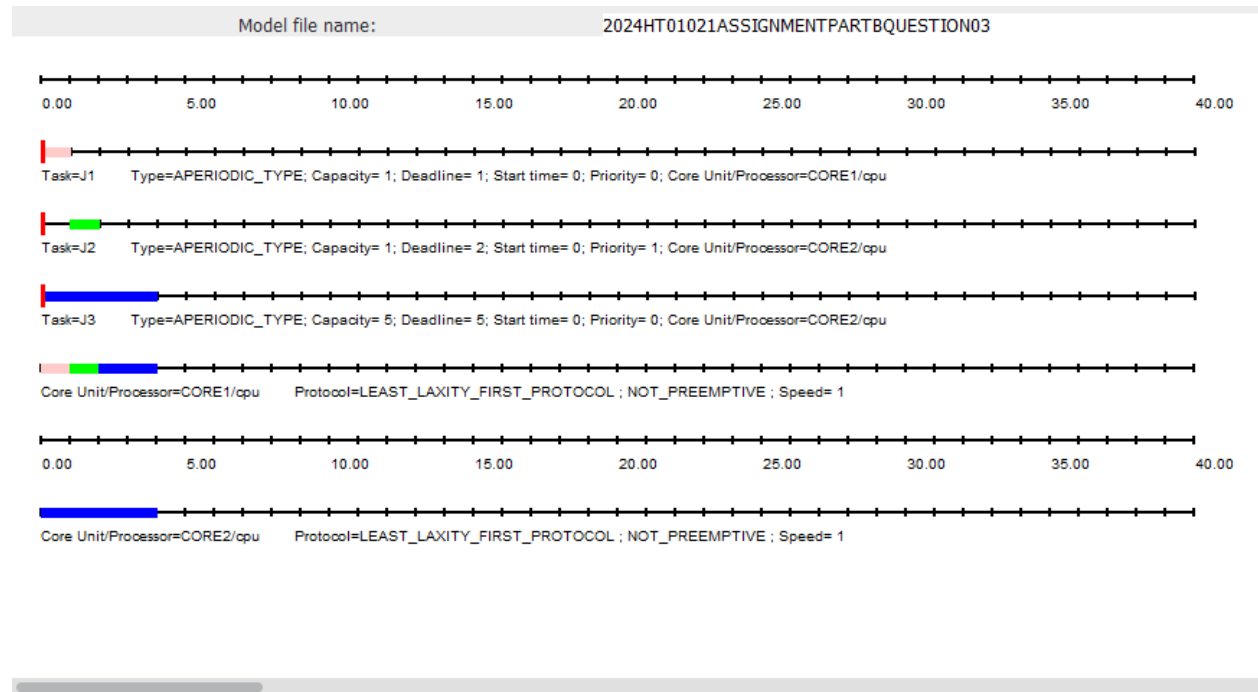
Core Configuration for Core 1 and Core 2.

Name	Scheduler	Quantum	Preemptive	Automaton
CORE1	Least Laxity First Protocol	0	Not Preemptive	
CORE2	Least Laxity First Protocol	0	Not Preemptive	

Task Setup:

Name	Task Type	Processor	Address Space	Core	Capacity	Deadline	Start time	Priority	...
J1	Aperiodic	cpu	EA	CORE1	1	1	0	0	0
J2	Aperiodic	cpu	EA	CORE2	1	2	0	1	0
J3	Aperiodic	cpu	EA	CORE2	5	5	0	0	0

Step 5: Simulation with LLF Scheduler



- No task set ready to analyse : load a model from the 'File' menu, or edit task set from the 'Edit' menu.

- Task set ready to analyse : run analysis from the 'Tools' menu.

Scheduling simulation, Processor cpu :

- Number of context switches : 3
- Number of preemptions : 2

- Task response time computed from simulation :

J1 => 1/worst

J2 => 2/worst

J3 => 4/worst

- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.

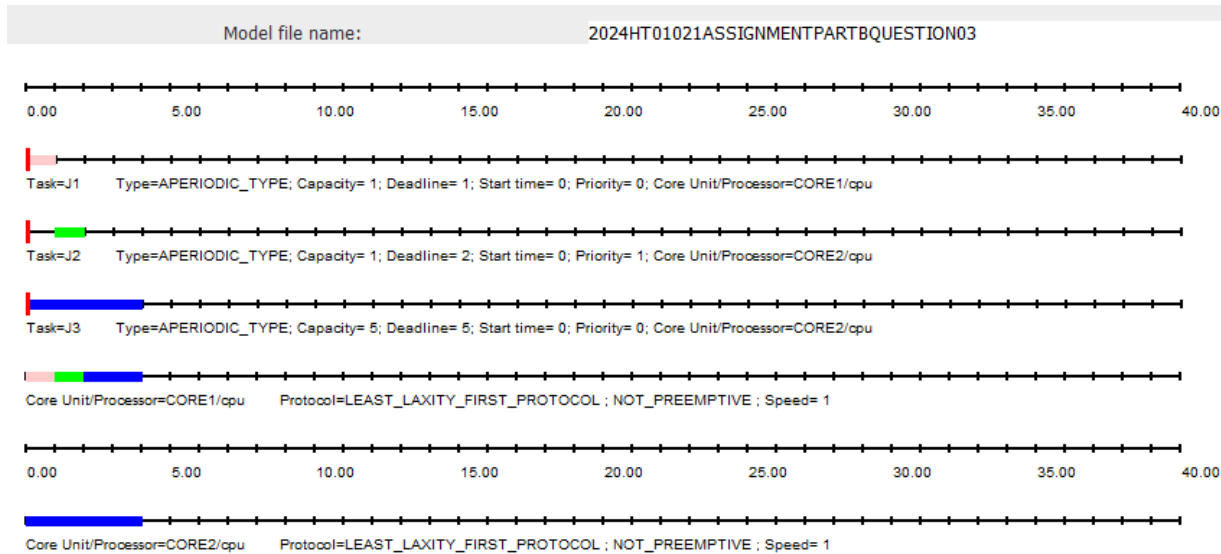
Result:

The simulation for the tasks J1, J2, and J3 has run successfully. The Least Laxity First (LLF) protocol has been applied, and the output confirms the following:

1. **Number of Context Switches:** 3
2. **Number of Preemptions:** 2
3. **Task Response Times:**
 - o J1: 1 (Worst-case response time)
 - o J2: 2 (Worst-case response time)
 - o J3: 4 (Worst-case response time)

Conclusion:

The task set is **schedulable** on the dual-core processor under the LLF protocol, and no deadlines were missed in the computed scheduling.



Step 6: Analysis and Conclusion

In this analysis, we demonstrated that **Earliest Deadline First (EDF)** scheduling fails to meet the deadline of Task J3 because it prioritizes J1 and J2, both of which have shorter deadlines. As a result, J3's execution is delayed beyond its deadline, making the task set unschedulable under EDF.

On the other hand, Least Laxity First (LLF) successfully schedules all tasks because it prioritizes tasks with the least slack, ensuring that Task J3's long execution time is accounted for without missing its deadline.

Conclusion:

- The task set is schedulable using LLF but not using EDF on a dual-core processor.

Here is the GitHub link for the above solution:

https://github.com/mukundchavan251/BITS/blob/main/Real_Time_Systems/Assignments/Q3/2024HT01011AssignmentNo1_Q3