<u>HW Assignment 4</u>

Q1)

The task set τ for the processor is given below:

τ1(r0 = 0, C = 1, D = 3, T = 3)

τ2(r0 = 0, C = 1, D = 4, T = 4)

τ3(r0 = 0, C = 2, D = 3, T = 6)

a) Processor utilization is defined as the summation of number of cycles (c_i) divided by time period (p_i) of all the tasks performed by that processor. This factor should always be less or equal to one only then the processor can be scheduled. Mathematically it is shown below:

$$\mathbf{U = \sum c_i/p_i}$$

So for the given task set the processor utilization U = (1/3) + (1/4) + (2/6) = 0.9167 which is equivalent to 91.67%

Major cycle is defined as time duration [0, LCM of time period of all the tasks of the processor].

In this case the LCM of the time period of all the tasks is 12. Hence, the major cycle of this processor is [0,12].

b)

Rate Monotonic (RM) Schedule for the major cycle for the given task set of the processor is given below:

In RM algorithm higher priority is given to the task with lower time period. Also, for rate monotonic scheduling, $n(2^{1/n}-1) >= U$,

where, n= number of tasks

Time	0-1cycle	1-2cycle	2-3cycle	3-4cycle	4-5cycle	5-6cycle	6-7cycle	7-8cycle	8-9cycle	9-10cycle	10-11cycle	11-12cycle
Tasks Arriving	τ1, τ2, τ3			τ1	τ2		τ1, τ3		τ2	τ1		
Task Deadline			τ1, τ3	τ2		τ1, τ3		τ2	τ1, τ3			
RM Scheduling	τ1	τ2	τ3	τ1	τ2	τ3	τ1	τ3	τ2	τ1	τ3	

Hence, for three tasks is $3*(2^{1/3} -1) = 0.779$ which is less than 0.9167. So Rate Monotonic Scheduling Algorithm fails in this case.

Earliest Deadline First (EDF) Schedule for the major cycle of the given task set of the processor is given below:

In EDF the task with the earliest deadline is given the highest priority.

Time	0-1cycle	1-2cycle	2-3cycle	3-4cycle	4-5cycle	5-6cycle	6-7cycle	7-8cycle	8-9cycle	9-10cycle	10-11cycle	11-12cycle
Tasks Arriving	τ1, τ2, τ3			τ1	τ2		τ1, τ3		τ2	τ1		
Task Deadline			τ1, τ3	τ2		τ1		τ2	τ1, τ3			τ1, τ2
EDF Scheduling	τ1	τ3		τ2	τ1	τ2	τ1	τ3		τ1	τ2	

As seen above EDF is able to meet all the deadlines and hence is a good choice for scheduling this processor.

Least Laxity First (LLF) Schedule for the major cycle of the given task set of the processor is given below:

In LLF the task priority changes dynamically after each cycle, according to the laxity of each task after each cycle. The task with least laxity is given the highest priority in this algorithm.

The formula for calculating laxity is laxity = deadline – time left of that task.

Time	0-1cycle	1-2cycle	2-3cycle	3-4cycle	4-5cycle	5-6cycle	6-7cycle	7-8cycle	8-9cycle	9-10cycle	10-11cycle	11-12cycle
Tasks Arriving	τ1, τ2, τ3			τ1	τ2		τ1, τ3		τ2	τ1		
Task Deadline			τ1, τ3	τ2		τ1		τ2	τ1, τ3			
LLF Scheduling	τ3		τ1	τ2	τ1	τ2	τ3		τ1	τ2	τ1	
L1	2	1	0	2	1		2	1	0	2		
L2	3	2	1	0	3	2			3	2		
L3	1	1					1	1				

Here, L1 corresponds to laxity of task 1, L2 to laxity of task 2 and L3 to laxity of task 3. Again this is an optimum scheduling algorithm for this processor as all the task meets it deadline.

Q2)

The task set τ for the processor is given below:

$\tau 1(r0 = 0, C = 2, D = 8, T = 8)$

$\tau 2(r0 = 0, C = 4, D = 12, T =12)$

$\tau 3(r0 = 0, C = 4, D = 16, T = 16)$

The task of the given periodic server is τs $(r0 = 0, C = 2, D = 12, T = 12)$

Now, $\tau' = \tau + \{\tau s\}$. Hence the new task set τ' is given below:

$\tau 1(r0 = 0, C = 2, D = 8, T = 8)$

$\tau 2(r0 = 0, C = 4, D = 12, T =12)$

$\tau 3(r0 = 0, C = 4, D = 16, T = 16)$

τs $(r0 = 0, C = 2, D = 12, T = 12)$

This periodic server, serves the aperiodic task request. The two periodic tasks given here are

case a: $\tau 4(r = 18, C = 4, D = 12)$

case b: $\tau 4(r = 18, C = 4, D = 20)$

These aperiodic task occur, at the τs instance as the sever provides service to these aperiodic tasks.

a)

The processor utilization factor U for task τ' of the processor is:

$U = (2/8) + (4/12) + (4/16) + (2/12) = 1$ which is equal to 100% utilization.

Also, the LCM of all the periods of all the tasks in the set τ' is 48.

Hence the major cycle is [0,48].

b)

Rate Monotonic (RM) Schedule for the major cycle for the given task set of the processor is given below:

As discussed earlier, in RM algorithm higher priority is given to the task with lower time period. Also, for rate monotonic scheduling, $n(2^{1/n}-1) >= U$,

where, n= number of tasks

The following is the aperiodic task for which the scheduling has to be done along with former tasks.

case a: τ4(r = 18, C = 4, D = 12)

Time	0-1cycle	1-2cycle	2-3cycle	3-4cycle	4-5cycle	5-6cycle	6-7cycle	7-8cycle	8-9cycle	9-10cycle	10-11cycle	11-12cycle
Tasks Arriving	τ1, τ2, τ3								τ1			
Aperiodic Task Arrival												
Periodic Task Server	τs											
Task Deadline								τ1				τ2, τs
RM Scheduling	τ1		τ2				τs		τ1		τ3	

Time	12-13cycle	13-14cycle	14-15cycle	15-16cycle	16-17cycle	17-18cycle	18-19cycle	19-20cycle	20-21cycle	21-22cycle	22-23cycle	23-24cycle
Tasks Arriving	τ2				τ1, τ3							
Aperiodic Task Arrival							τ4					
Periodic Task Server	τs											
Task Deadline				τ1, τ3								τ1, τ2, τs
RM Scheduling	τ2				τ1		τ4		τ3		τ3	

Time	24-25cycle	25-26cycle	26-27cycle	27-28cycle	28-29cycle	29-30cycle	30-31cycle	31-32cycle	32-33cycle	33-34cycle	34-35cycle	35-36cycle
Tasks Arriving	τ1, τ2								τ1, τ3			
Aperiodic Task Arrival												
Periodic Task Server	τs											
Task Deadline						τ4		τ1, τ3				τ2, τs
RM Scheduling	τ1		τ2				τ4		τ1		τ3	

Time	36-37cycle	37-38cycle	38-39cycle	39-40cycle	40-41cycle	41-42cycle	42-43cycle	43-44cycle	44-45cycle	45-46cycle	46-47cycle	47-48cycle
Tasks Arriving	τ2				τ1							
Aperiodic Task Arrival												
Periodic Task Server	τs											
Task Deadline				τ1								τ1, τ2, τ3, τs
RM Scheduling	τ2				τ1		τs		τ3			

In above case using RM scheduling, τ1 pre-empts τ2 and τ2 Pre-empts τ3 because of their periods. τ2 and τs do not pre-empt each other, but τs do pre-empts τ3.

Also, as discussed earlier for RM $n(2^{1/n}-1)$ for four tasks including the server task is 0.757 which is less than U. Hence RM scheduling fails.

Therefore, as seen from the table RM scheduling fails to complete tasks τ4 before its deadline at 30 and task τ3 before its deadlines at 16 and 32.

c) Earliest Deadline First (EDF) Schedule for the major cycle for the given task set of the processor is given below:

In EDF the task with the earliest deadline is given the highest priority.

The following is the aperiodic task for which the scheduling has to be done along with former tasks.

case b: τ4(r = 18, C = 4, D = 20)

<table>
<tr><td>Time</td><td>0-1cycle</td><td>1-2cycle</td><td>2-3cycle</td><td>3-4cycle</td><td>4-5cycle</td><td>5-6cycle</td><td>6-7cycle</td><td>7-8cycle</td><td>8-9cycle</td><td>9-10cycle</td><td>10-11cycle</td><td>11-12cycle</td></tr>
<tr><td>Tasks Arriving</td><td>τ1, τ2, τ3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>τ1</td><td></td><td></td><td></td></tr>
<tr><td>Aperiodic Task Arrival</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Periodic Server Task</td><td>τs</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Task Deadline</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>τ1</td><td></td><td></td><td></td><td>τ2, τs</td></tr>
<tr><td>EDF Scheduling</td><td colspan="2">τ1</td><td colspan="4">τ2</td><td colspan="2">τs</td><td colspan="2">τ1</td><td colspan="2">τ3</td></tr>
<tr><td colspan="13"></td></tr>
<tr><td colspan="13"></td></tr>
<tr><td>Time</td><td>12-13cycle</td><td>13-14cycle</td><td>14-15cycle</td><td>15-16cycle</td><td>16-17cycle</td><td>17-18cycle</td><td>18-19cycle</td><td>19-20cycle</td><td>20-21cycle</td><td>21-22cycle</td><td>22-23cycle</td><td>23-24cycle</td></tr>
<tr><td>Tasks Arriving</td><td>τ2</td><td></td><td></td><td></td><td>τ1, τ3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Aperiodic Task Arrival</td><td></td><td></td><td></td><td></td><td></td><td>τ4</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Periodic Server Task</td><td>τs</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Task Deadline</td><td></td><td></td><td></td><td>τ1, τ3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>τ1, τ2, τs</td></tr>
<tr><td>EDF Scheduling</td><td colspan="2">τ3</td><td colspan="2">τ2</td><td colspan="2">τ1</td><td colspan="2">τ2</td><td colspan="2">τ4</td><td colspan="2">τ3</td></tr>
<tr><td colspan="13"></td></tr>
<tr><td colspan="13"></td></tr>
<tr><td>Time</td><td>24-25cycle</td><td>25-26cycle</td><td>26-27cycle</td><td>27-28cycle</td><td>28-29cycle</td><td>29-30cycle</td><td>30-31cycle</td><td>31-32cycle</td><td>32-33cycle</td><td>33-34cycle</td><td>34-35cycle</td><td>35-36cycle</td></tr>
<tr><td>Tasks Arriving</td><td>τ1, τ2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>τ1, τ3</td><td></td><td></td><td></td></tr>
<tr><td>Aperiodic Task Arrival</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Periodic Server Task</td><td>τs</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Task Deadline</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>τ1, τ3</td><td></td><td></td><td></td><td>τ2, τs</td></tr>
<tr><td>EDF Scheduling</td><td colspan="2">τ1</td><td colspan="2">τ3</td><td colspan="4">τ2</td><td colspan="2">τ1</td><td colspan="2">τ4</td></tr>
<tr><td colspan="13"></td></tr>
<tr><td>Time</td><td>36-37cycle</td><td>37-38cycle</td><td>38-39cycle</td><td>39-40cycle</td><td>40-41cycle</td><td>41-42cycle</td><td>42-43cycle</td><td>43-44cycle</td><td>44-45cycle</td><td>45-46cycle</td><td>46-47cycle</td><td>47-48cycle</td></tr>
<tr><td>Tasks Arriving</td><td>τ2</td><td></td><td></td><td></td><td>τ1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Aperiodic Task Arrival</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Periodic Server Task</td><td>τs</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Task Deadline</td><td></td><td>τ4</td><td>τ1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>τ1, τ2, τ3, τs</td></tr>
<tr><td>EDF Scheduling</td><td colspan="4">τ2</td><td colspan="2">τ1</td><td colspan="2">τs</td><td colspan="4">τ3</td></tr>
</table>

Note: task τ3 is continuous same task in cycle 11-13 and not two different tasks.

As seen from the table, the EDF scheduling algorithm is able to meet all the tasks before its deadline and hence, is the optimum choice for scheduling this processor.

Q3)

Task	r_i	C_i	D_i	T_i
τ_1	1	2 (R)	6	6
τ_2	1	2	8	8
τ_3	0	5 (R)	12	12

Priority inversion is a scenario in scheduling in which a high priority task is indirectly preempted by a lower priority task effectively inverting the relative priorities of the two tasks. This happens when two or more tasks share the same resources at the same time.

a)

The processor utilization factor for the given task set of the processor is:

U = (2/6) + (2/8) + (5/12) = 1 which is equivalent to 100%

Time	0-1cycle	1-2cycle	2-3cycle	3-4cycle	4-5cycle	5-6cycle	6-7cycle	7-8cycle	8-9cycle	9-10cycle	10-11cycle	11-12cycle	12-13cycle
Tasks Arriving	τ3	τ1, τ2						τ1		τ2			τ3
Task Deadline							τ1		τ2			τ3	τ1
EDF Scheduling	τ3	τ2		τ3				τ1		τ1		τ2	
			Priority inversion occurs here										
Time	13-14cycle	14-15cycle	15-16cycle	16-17cycle	17-18cycle	18-19cycle	19-20cycle	20-21cycle	21-22cycle	22-23cycle	23-24cycle	24-25cycle	
Tasks Arriving	τ1				τ2		τ1					τ3	
Task Deadline				τ2		τ1					τ3		
EDF Scheduling	τ1		τ3		τ2		τ3			τ1		τ3	

As indicated at time unit 1 priority inversion takes place. Hence at the 1st time unit after task τ3, task τ2 is executed instead of task τ1 because task τ3 is using the critical resource. This causes task τ1 to miss its deadline. To avoid this, we use priority inheritance protocol.

b)

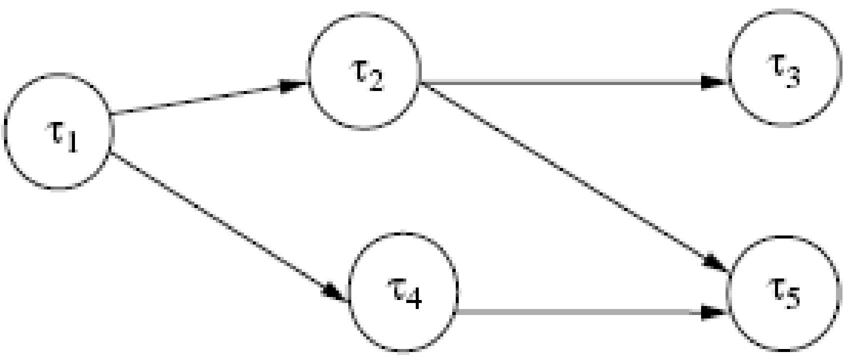
Now, EDF with priority inheritance protocol.

<table>
<tr><td>Time</td><td>0-1cycle</td><td>1-2cycle</td><td>2-3cycle</td><td>3-4cycle</td><td>4-5cycle</td><td>5-6cycle</td><td>6-7cycle</td><td>7-8cycle</td><td>8-9cycle</td><td>9-10cycle</td><td>10-11cycle</td><td>11-12cycle</td><td>12-13cycle</td></tr>
<tr><td>Tasks Arriving</td><td>τ3</td><td>τ1, τ2</td><td></td><td></td><td></td><td></td><td></td><td>τ1</td><td></td><td>τ2</td><td></td><td></td><td>τ3</td></tr>
<tr><td>Task Deadline</td><td></td><td></td><td></td><td></td><td></td><td></td><td>τ1</td><td></td><td>τ2</td><td></td><td></td><td>τ3</td><td>τ1</td></tr>
<tr><td>EDF Scheduling</td><td colspan="5">τ3</td><td colspan="2">τ1</td><td colspan="2">τ2</td><td colspan="2">τ1</td><td colspan="2">τ2</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Time</td><td>13-14cycle</td><td>14-15cycle</td><td>15-16cycle</td><td>16-17cycle</td><td>17-18cycle</td><td>18-19cycle</td><td>19-20cycle</td><td>20-21cycle</td><td>21-22cycle</td><td>22-23cycle</td><td>23-24cycle</td><td>24-25cycle</td><td></td></tr>
<tr><td>Tasks Arriving</td><td>τ1</td><td></td><td></td><td></td><td>τ2</td><td></td><td>τ1</td><td></td><td></td><td></td><td></td><td>τ3</td><td></td></tr>
<tr><td>Task Deadline</td><td></td><td></td><td></td><td>τ2</td><td></td><td>τ1</td><td></td><td></td><td></td><td></td><td>τ3</td><td></td><td></td></tr>
<tr><td>EDF Scheduling</td><td colspan="2">τ1</td><td colspan="5">τ3</td><td colspan="2">τ2</td><td colspan="2">τ1</td><td>τ3</td><td></td></tr>
</table>

As seen in the table due to priority inheritance, at time unit 1 and because of that task τ3 occurs instead of task τ2 as it is using the critical resource. But at time unit 17 and 19 task τ3 pre-empts task τ2 and task τ1 respectively because task τ3 has earlier deadline and not because of priority inheritance protocol. And as seen in the table, using priority inheritance protocol we can prevent priority inversion phenomenon and meet all the task deadlines.

Q4)

Task	r_i	C_i	D_i	T_i
τ_1	0	3	12	12
τ_2	0	2	11	11
τ_3	0	3	12	12
τ_4	0	1	11	11
τ_5	0	2	9	9



The table above shows the set of tasks which are dependent on each other and the precedence graph shows the conditions for the event or task to take place. In this case, we are supposed to use EDF scheduling algorithm. According to EDF algorithm, task τ5 should be executed first. But since all the task are dependent tasks and as shown in the precedence graph, task τ5 would occur only after task τ2 and task τ4 is executed and for task τ2 and task τ4 to be executed task τ1 must occur. This is depicted in the table below:

Time	0-1cycle	1-2cycle	2-3cycle	3-4cycle	4-5cycle	5-6cycle	6-7cycle	7-8cycle	8-9cycle	9-10cycle	10-11cycle	11-12cycle
Tasks Arriving	τ1, τ2, τ3, τ4, τ5									τ5		τ2, τ4
Task Deadline									τ5			τ1, τ3
EDF Scheduling	τ1			τ4	τ2		τ5		τ3			

Time	12-13cycle	13-14cycle	14-15cycle	15-16cycle	16-17cycle	17-18cycle	18-19cycle	19-20cycle				
Tasks Arriving	τ1, τ3											
Task Deadline						τ5						
EDF Scheduling	τ1			τ4	τ2		τ5					