Real-Time Systems — EDA223/DIT162

Final exam, March 14, 2022 at 08:30 – 12:30 in Lecture Halls HA

Examiner:

Professor Jan Jonsson, Department of Computer Science and Engineering

Responsible teacher:

Risat Pathan, phone: 070–9415951

Visits the exam room at 09:30 and 11:30.

Aids permitted during the exam:

Chalmers-approved calculator

NOTE: Electronic dictionaries may not be used.

Content:

The **Basic part** of the exam consists of 3 problems worth a total of 30 points.

The **Advanced part** of the exam consists of 4 problems worth a total of 30 points.

Grading policy:

Obtaining less than 24 points in the Basic part leads to a **Fail** grade.

If you obtain at least 24 points in the Basic part, then your grade is determined by the total number of points in the Basic and Advanced parts combined as follows:

24–35 points ⇒ **grade 3**	24–43 points ⇒ **grade G** (GU)
36–47 points ⇒ **grade 4**	
48–60 points ⇒ **grade 5**	44–60 points ⇒ **grade VG** (GU)

Results:

When the grading is completed overall result statistics, and a time and location for inspection, will be announced on the course home page. Individual results will be available in Ladok.

Language:

Your solutions should be written in English

IMPORTANT ISSUES

1. Use separate sheets for each answered problem, and mark each sheet with the problem number.
2. Justify all answers. Lack of justification can lead to loss of credit even if the answer is correct.
3. Explain all calculations thoroughly. If justification and method is correct then simple calculation mistakes do not necessarily lead to loss of credit.
4. If some assumptions in a problem are missing or you consider that the made assumptions are unclear, then please state explicitly which assumptions you make in order to find a solution.
5. Write clearly! If we cannot read your solution, we will assume that it is wrong.

Good Luck!

Basic part

PROBLEM 1

The following sub-problem concerns processor-utilization analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the rate-monotonic (RM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	18	18
τ_2	3	27	27
τ_3	19	54	54

a) Perform Liu & Layland's utilization-based analysis for the given task set. (2 points)

The following sub-problem concerns response-time analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the deadline-monotonic (DM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	12	18
τ_2	3	10	27
τ_3	19	31	54

b) Perform response-time analysis for the given task set. The final (converged) response time should be calculated for each task in the task set, regardless of whether the response-time analysis for that task fails or not. (4 points)

PROBLEM 1 (cont'd)

The following sub-problems concern processor-demand analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the earliest-deadline-first (EDF) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	12	18
τ_2	3	10	27
τ_3	19	31	54

c) Show that the largest interval to examine is $\mathrm{L_{max}} = 54$ for the given task set. The solution should include calculations of the hyper-period upper bound as well as the upper bound proposed by Baruah, Rosier and Howell. (2 points)

d) Calculate the complete set of control points within $\mathrm{L_{max}} = 54$ for the given task set. (2 points)

e) Perform processor-demand analysis for the given task set, and the complete set of control points calculated in sub-problem d). The analysis should be performed for every control point in the set, regardless of whether the analysis in another control point fails or not. (4 points)

PROBLEM 2

The following sub-problems concern hyper-period analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling. The table below shows O_i (offset), C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i.

	O_i	C_i	D_i	T_i
τ_1	0	6	12	18
τ_2	3	3	10	27
τ_3	0	19	31	54

a) Simulate the execution of the task set using the earliest-deadline-first (EDF) priority-assignment policy, and construct a timing diagram that spans the entire hyper period. The arrival and deadline of each task instance should be clearly indicated in the diagram in the form of an up-arrow (arrival) and down-arrow (deadline), respectively. All task executions must be completed, regardless of whether a task instance misses its deadline or not. (5 points)

b) Derive a time table, corresponding to the timing diagram constructed in sub-problem a), that clearly states the start and stop times for each task instance (or task segment, if a task instance is preempted). (1 point)

PROBLEM 3

The following sub-problems concern analysis of worst-case execution time (WCET).

Consider the following C program code for the functions `FuncA` and `FuncB`:

```
int FuncA(int a, int b) {
   int p;
   int i;

   p = a;
   i = b;

   if (b == 0)
      return 1;

   while (i > 1) {
      p = p * a;
      i = i - 1;
   }

   return p;
}

int FuncB(int c, int d) {
   int p;

   if (d == 1)
      p = c;
   else
      p = c * FuncB(c, d-1);

   return p;
}
```

Assume the following costs for the language statements:

- Each declaration and assignment statement costs 1 μs to execute.
- Each function call costs 2 μs plus WCET for the function in question.
- Each evaluation of the logical condition in an `if`- or `while`-statement costs 2 μs.
- Each add and subtract operation costs 3 μs.
- Each multiply operation costs 5 μs.
- Each return statement costs 2 μs.
- All other language constructs can be assumed to take 0 μs to execute.

a) Derive the WCET of function `FuncA` by using the analysis method proposed by Shaw. The WCET should be expressed in the form $\mathbf{k_1 \cdot a + k_2 \cdot b + k_3}$, where k_1, k_2 and k_3 are integer constants, and **a** and **b** are the function parameters. Assume that parameter $\mathbf{b \geq 0}$. (3 points)

b) Derive the WCET of function `FuncB` by using the analysis method proposed by Shaw. The WCET should be expressed in the form $\mathbf{k_1 \cdot c + k_2 \cdot d + k_3}$, where k_1, k_2 and k_3 are integer constants, and **c** and **d** are the function parameters. Assume that parameter $\mathbf{d \geq 1}$. (3 points)

PROBLEM 3 (cont'd)

The following sub-problems concern scheduling concepts.

c) Describe the meaning of a *priority ceiling* in access-control protocols for critical regions. (1 point)

d) State the major difference in the behaviour of the *immediate ceiling priority protocol* (ICPP) and the original *priority ceiling protocol* (PCP). (1 point)

The following sub-problems concern real-time programming.

e) In the TinyTimber kernel there is always a *baseline* associated with a method execution. Describe the purpose of the baseline in the TinyTimber run-time scheduler. (1 point)

f) The TinyTimber run-time scheduler takes into account any *priority* that may be associated with a method execution. Describe how the application programmer instructs the TinyTimber kernel that a method execution should be associated with a priority. (1 point)

Advanced part

PROBLEM 4

The following questions are related to feasibility testing and time complexity of scheduling algorithms.

a) The Traveling Salesman decision problem (TSP) is known to be NP-complete, and therefore also (per definition) will belong to class NP. Explain why the complement Traveling Salesman decision problem (TSP^{c}) cannot belong to class NP. (3 points)

b) Consider a synchronous (all offsets O_i are identical) task set for which it applies that the deadline of each task does not exceed its period (that is, constrained-deadline tasks with $D_i \leq T_i$). What is the largest value of the common offset O_i for which pseudo-polynomial time complexity can be guaranteed for feasibility testing with response-time analysis on a single-processor system? Motivate your answer! (3 points)

c) Consider a synchronous (all offsets O_i are identical) task set for which it applies that the deadline of each task does not exceed its period (that is, constrained-deadline tasks with $D_i \leq T_i$). Assume that the task set is known to be schedulable on a single-processor system using preemptive scheduling and static task priorities. If one of the tasks in the task set would increase its offset by one time unit, keeping its other task parameters intact, would the task set still be schedulable on the single-processor system? Motivate your answer! (2 points)

PROBLEM 5

Consider a real-time system with three independent periodic tasks and a run-time system that uses preemptive single-processor scheduling using the rate-monotonic (RM) priority-assignment approach. The table below shows O_i (offset), C_i (WCET) and U_i (utilization) for the three tasks. The relative deadline of each periodic task is equal to its period. The value of the parameter C is not known.

	O_i	C_i	U_i
τ_1	0.4C	0.2C	1/3
τ_2	0	0.3C	3/8
τ_3	0	0.3C	1/4

Use a suitable analysis method to determine the schedulability of the tasks in the system. (8 points)

PROBLEM 6

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the earliest-deadline-first (EDF) priority-assignment approach. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	3	4	8
τ_2	4	12	16
τ_3	8	D_3	32

The task set is known to be schedulable for $D_3 = 30$. Use a suitable analysis method to determine the smallest integer value of the deadline D_3 for which the task set is still schedulable. (6 points)

PROBLEM 7

Consider a real-time system with five independent periodic tasks that should be scheduled on $m = 2$ processors using the rate-monotonic first-fit (RMFF) partitioned scheduling algorithm.

The table below shows C_i (WCET) and T_i (period) for the five tasks. The relative deadline of each periodic task is equal to its period. All tasks arrive at $t = 0$.

	C_i	T_i
τ_1	2	10
τ_2	2	27
τ_3	$T_3/4$	T_3
τ_4	15	37
τ_5	15	30

a) Show that Oh & Baker's test for RMFF does not provide enough information to determine if the tasks can successfully be scheduled on $m = 2$ processors. (2 points)

b) Determine, by assigning the tasks to the $m = 2$ processors according to the RMFF algorithm, the smallest integer value of T_3 such that (i) T_3 is evenly divisible by 4 and (ii) all the task deadlines are met. Show the assignment of the tasks to the processors. (6 points)

Real-Time Systems — EDA223/DIT162

Re-exam, August 16, 2022 at 14:00 – 18:00 in the Saga building

---

Examiner:

Professor Jan Jonsson, Department of Computer Science and Engineering

Responsible teacher:

Jan Jonsson, phone: 031–772 5220

Visits the exam room at 15:00 and 17:00.

Aids permitted during the exam:

Chalmers-approved calculator

NOTE: Electronic dictionaries may not be used.

Content:

The **Basic part** of the exam consists of 3 problems worth a total of 30 points.

The **Advanced part** of the exam consists of 4 problems worth a total of 30 points.

Grading policy:

Obtaining less than 24 points in the Basic part leads to a **Fail** grade.

If you obtain at least 24 points in the Basic part, then your grade is determined by the total number of points in the Basic and Advanced parts combined as follows:

24–35 points ⇒ **grade 3**	24–43 points ⇒ **grade G** (GU)
36–47 points ⇒ **grade 4**	
48–60 points ⇒ **grade 5**	44–60 points ⇒ **grade VG** (GU)

Results:

When the grading is completed overall result statistics, and a time and location for inspection, will be announced on the course home page. Individual results will be available in Ladok.

Language:

Your solutions should be written in English

IMPORTANT ISSUES

1. Use separate sheets for each answered problem, and mark each sheet with the problem number.
2. Justify all answers. Lack of justification can lead to loss of credit even if the answer is correct.
3. Explain all calculations thoroughly. If justification and method is correct then simple calculation mistakes do not necessarily lead to loss of credit.
4. If some assumptions in a problem are missing or you consider that the made assumptions are unclear, then please state explicitly which assumptions you make in order to find a solution.
5. Write clearly! If we cannot read your solution, we will assume that it is wrong.

Good Luck!

Basic part

PROBLEM 1

The following sub-problem concerns processor-utilization analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the rate-monotonic (RM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	15	15
τ_2	2	25	25
τ_3	21	75	75

a) Perform Liu & Layland's utilization-based analysis for the given task set. (2 points)

The following sub-problem concerns response-time analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the deadline-monotonic (DM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	10	15
τ_2	2	8	25
τ_3	21	43	75

b) Perform response-time analysis for the given task set. The final (converged) response time should be calculated for each task in the task set, regardless of whether the response-time analysis for that task fails or not. (4 points)

PROBLEM 1 (cont'd)

The following sub-problems concern processor-demand analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the earliest-deadline-first (EDF) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	10	15
τ_2	2	8	25
τ_3	21	43	75

c) Show that the largest interval to examine is $L_{\max} = 52$ for the given task set. The solution should include calculations of the hyper-period upper bound as well as the upper bound proposed by Baruah, Rosier and Howell. (2 points)

d) Calculate the complete set of control points within $L_{\max} = 52$ for the given task set. (2 points)

e) Perform processor-demand analysis for the given task set, and the complete set of control points calculated in sub-problem d). The analysis should be performed for every control point in the set, regardless of whether the analysis in another control point fails or not. (4 points)

PROBLEM 2

The following sub-problems concern hyper-period analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling. The table below shows O_i (offset), C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i.

	O_i	C_i	D_i	T_i
τ_1	0	6	10	15
τ_2	4	2	8	25
τ_3	0	21	43	75

a) Simulate the execution of the task set using the deadline-monotonic (DM) priority-assignment policy, and construct a timing diagram that spans the entire hyper period. The arrival and deadline of each task instance should be clearly indicated in the diagram in the form of an up-arrow (arrival) and down-arrow (deadline), respectively. All task executions must be completed, regardless of whether a task instance misses its deadline or not. (5 points)

b) Derive a time table, corresponding to the timing diagram constructed in sub-problem a), that clearly states the start and stop times for each task instance (or task segment, if a task instance is preempted). (1 point)

PROBLEM 3

The following sub-problems concern analysis of worst-case execution time (WCET).

Consider the following C program code for the function `FuncY`:

```
int FuncY(int y) {
   int t;

   if (y == 0) {
      t = 0;
   } else {
      if (y == 1) {
         t = 1;
      } else {
         t = y + FuncY(y-1);

         t = t * t;
      }
   }

   return t;
}
```

Assume the following costs for the language statements:

- Each declaration and assignment statement costs 1 μs to execute.
- Each function call costs 2 μs plus WCET for the function in question.
- Each evaluation of the logical condition in an `if`- or `while`-statement costs 2 μs.
- Each add and subtract operation costs 3 μs.
- Each multiply operation costs 5 μs.
- Each return statement costs 2 μs.
- All other language constructs can be assumed to take 0 μs to execute.

a) Derive the WCET of function `FuncY` by using the analysis method proposed by Shaw. The WCET should be expressed in the form $\mathbf{k_1 \cdot y + k_2}$, where k_1 and k_2 are integer constants, and y is the function parameter. Assume that `FuncY` is always called with parameter value $\mathbf{y > 0}$. (3 points)

b) Explain why is it preferred that WCET estimates for the program code of tasks in a real-time system are *tight*. (1 points)

PROBLEM 3 (cont'd)

The following sub-problems concern scheduling concepts.

c) Describe the meaning of *Dhall's effect* in the context of global multiprocessor scheduling, and also explain why the effect occurs. (2 points)

d) Describe the difference between of a *periodic task* and a *sporadic task*. (1 point)

The following sub-problems concern real-time programming.

e) Describe the meaning of *systematic time skew* in the context of implementation of periodic task executions. (2 points)

f) State the prominent feature of the `AFTER()` operation in the TinyTimber kernel that facilitates implementation of periodic tasks without systematic time skew. (1 point)

Advanced part

PROBLEM 4

The following questions are related to feasibility testing and time complexity of scheduling algorithms.

a) The rate-monotonic (RM) and earliest-deadline-first (EDF) priority-assignment policies are both optimal under similar assumptions regarding the task model on a single-processor system. Does this mean that the policies are equally good in terms of schedulability, or does one policy dominate the other? (2 points)

b) Consider a synchronous (all offsets O_i are identical) task set for which it applies that the deadline of each task does not exceed its period (that is, constrained-deadline tasks with $D_i \leq T_i$). Assume that the task set is known to be schedulable on a multiprocessor system using preemptive global scheduling and static task priorities. If one of the tasks in the task set would increase its offset by one time unit, keeping its other task parameters intact, would the task set still be schedulable on the multiprocessor system? Motivate your answer! (3 points)

c) Consider a synchronous (all offsets O_i are identical) task set for which it applies that the deadline of each task is equal to its period (that is, implicit-deadline tasks with $D_i = T_i$). Assume that the task set is known to be schedulable on a single-processor system using the rate-monotonic (RM) priority-assignment policy. State for each of the feasibility tests listed below what outcome ('True' or 'False') they would report if applied to the task set. Motivate your answer! (3 points)

- Response-time analysis
- Liu and Layland's utilization-based test for RM
- Liu and Layland's utilization-based test for EDF

PROBLEM 5

Consider a real-time system with two periodic tasks and a run-time system that employs preemptive rate-monotonic (RM) scheduling.

The relative deadline of each periodic task is equal to its period. Both tasks arrive at $t = 0$.

It is known that the tasks' utilizations $U_i = C_i/T_i$ are $U_1 = 0.75$ och $U_2 = 0.25$, respectively. However, the individual execution times C_i for the two tasks are not known.

a) If the tasks' periods T_i are not known either, is it possible to decide if the tasks are schedulable with RM? Motivate your answer. (3 points)

b) If we know that $T_2 = 2T_1$, is it then possible to decide whether the tasks are schedulable or not with RM? Motivate your answer. (3 points)

PROBLEM 6

Consider a real-time system with five tasks, whose timing properties are listed in the table below. Tasks τ_3 and τ_4 cooperate in the sense that τ_4 uses the result produced by τ_3, and they must both complete within a common end-to-end deadline of $D_{global} = 75$. The global deadline should be divided between these two tasks, and X represents the part of the deadline that is assigned to τ_3. To guarantee that valid data is available at the beginning of its execution, the offset for τ_4 is set to X. The same offset is also used for task τ_5. All other tasks are supposed to be independent, with an offset of 0.

	C_i	O_i	D_i	T_i
τ_1	5	0	6	15
τ_2	6	0	11	25
τ_3	9	0	X	75
τ_4	15	X	$75 - X$	75
τ_5	15	X	25	25

The tasks are assigned to two separate processors that are connected through a shared-bus network. Processor 1 executes tasks τ_1, τ_2 and τ_3, and employs preemptive earliest-deadline-first (EDF) scheduling (that is, dynamic priorities.) Processor 2 executes tasks τ_4 and τ_5, and employs preemptive deadline-monotonic (DM) scheduling (that is, static priorities.) The overhead for sending data between the processors over the network is assumed to be negligible.

Derive the range of allowed (maximum and minimum) values of X for which all tasks will meet their deadlines. (10 points)

PROBLEM 7

The following sub-problems are related to multiprocessor scheduling of independent periodic tasks.

a) Consider a task set with four independent periodic tasks and a run-time system that uses preemptive *global scheduling* on $m = 3$ processors. The task priorities are given according to the rate-monotonic (RM) policy. The table below shows C_i (WCET) and T_i (period) for each task in the task set. The relative deadline of each task is equal to its period. All tasks arrive at time $t = 0$.

	C_i	T_i
τ_1	10	100
τ_2	10	100
τ_3	10	100
τ_4	181	200

Show that is it is not possible to schedule the given task set on $m = 3$ processors such that all task deadlines are met, when using global scheduling with RM priorities. (3 points)

b) Show, by using a suitable *processor utilization test for global scheduling*, that there exists a better approach to assign static priorities to the tasks in sub-problem a) such that all task deadlines will be met when using global scheduling on $m = 3$ processors. (3 points)

Real-Time Systems — EDA223/DIT162

Final exam, March 13, 2023 at 08:30 – 12:30 in the SB2 and M buildings

---

Examiner:

Professor Jan Jonsson, Department of Computer Science and Engineering

Responsible teacher:

Jan Jonsson, phone: 031–772 5220

Visits the exam room at 09:30 and 11:30.

Aids permitted during the exam:

Chalmers-approved calculator

NOTE: Electronic dictionaries may not be used.

Content:

The **Basic part** of the exam consists of 3 problems worth a total of 30 points.

The **Advanced part** of the exam consists of 4 problems worth a total of 30 points.

Grading policy:

Obtaining less than 24 points in the Basic part leads to a **Fail** grade.

If you obtain at least 24 points in the Basic part, then your grade is determined by the total number of points in the Basic and Advanced parts combined as follows:

24–35 points ⇒ **grade 3**	24–43 points ⇒ **grade G** (GU)
36–47 points ⇒ **grade 4**	
48–60 points ⇒ **grade 5**	44–60 points ⇒ **grade VG** (GU)

Results:

When the grading is completed overall result statistics, and a time and location for inspection, will be announced on the course home page. Individual results will be available in Ladok.

Language:

Your solutions should be written in English

IMPORTANT ISSUES

1. Use separate sheets for each answered problem, and mark each sheet with the problem number.
2. Justify all answers. Lack of justification can lead to loss of credit even if the answer is correct.
3. Explain all calculations thoroughly. If justification and method is correct then simple calculation mistakes do not necessarily lead to loss of credit.
4. If some assumptions in a problem are missing or you consider that the made assumptions are unclear, then please state explicitly which assumptions you make in order to find a solution.
5. Write clearly! If we cannot read your solution, we will assume that it is wrong.

Good Luck!

Basic part

PROBLEM 1

The following sub-problem concerns processor-utilization analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the rate-monotonic (RM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	2	6	6
τ_2	4	14	14
τ_3	4	20	20

a) Perform Liu & Layland's utilization-based analysis for the given task set. (2 points)

The following sub-problem concerns response-time analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the deadline-monotonic (DM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	2	4	6
τ_2	4	11	14
τ_3	4	12	20

b) Perform response-time analysis for the given task set. The final (converged) response time should be calculated for each task in the task set, regardless of whether the response-time analysis for that task fails or not. (4 points)

PROBLEM 1 (cont'd)

The following sub-problems concern processor-demand analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the earliest-deadline-first (EDF) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	2	4	6
τ_2	4	11	14
τ_3	4	12	20

c) Show that the largest interval to examine is $L_{\max} = 18$ for the given task set. The solution should include calculations of the hyper-period upper bound as well as the upper bound proposed by Baruah, Rosier and Howell. (2 points)

d) Calculate the complete set of control points within $L_{\max} = 18$ for the given task set. (2 points)

e) Perform processor-demand analysis for the given task set, and the complete set of control points calculated in sub-problem d). The analysis should be performed for every control point in the set, regardless of whether the analysis in another control point fails or not. (4 points)

PROBLEM 2

The following sub-problems concern hyper-period analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling. The table below shows O_i (offset), C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i.

	O_i	C_i	D_i	T_i
τ_1	3	2	4	6
τ_2	0	4	11	14
τ_3	0	4	12	20

a) Simulate the execution of the task set using the earliest-deadline-first (EDF) priority-assignment policy, and construct a timing diagram that spans an interval of the hyper period, ranging from $t = 0$ to $t = 18$. The arrival and deadline of each task instance should be clearly indicated in the diagram in the form of an up-arrow (arrival) and down-arrow (deadline), respectively. All task executions that fall within the interval must be completed, regardless of whether a task instance misses its deadline or not. (5 points)

b) Derive a time table, corresponding to the timing diagram constructed in sub-problem a), that clearly states the start and stop times for each task instance (or task segment, if a task instance is preempted). (1 point)

PROBLEM 3

The following sub-problems concern analysis of worst-case execution time (WCET).

Consider the following C program code for the functions `CalcXY`, `CalcWZ` and `main`:

```
int CalcXY(int x, int y) {
   int p;
   int i;
   p = x;
   i = y;
   if (y == 0)
      return 1;
   while (i > 1) {
      p = p * x;
      i = i - 1;
   }
   return p;
}

int CalcWZ(int w, int z) {
   int p;
   if (z == 1)
      p = w;
   else
      p = w * CalcWZ(w, z-1);
   return p;
}

void main() {
   int a;
   int b;
   a = CalcXY(2, N);
   b = CalcWZ(2, L);
   if (b != a)
      b = CalcWZ(3, L);
   else
      b = b - 3;
}
```

Function `main` uses two constants, `N` and `L`, whose values are both positive integers in the range $[2, 5]$.

Assume the following costs for the language statements:

- Each declaration and assignment statement costs 1 μs to execute.
- Each function call costs 2 μs plus WCET for the function in question.
- Each evaluation of the logical condition in an `if`- or `while`-statement costs 2 μs.
- Each add and subtract operation costs 3 μs.
- Each multiply operation costs 5 μs.
- Each return statement costs 2 μs.
- All other language constructs can be assumed to take 0 μs to execute.

PROBLEM 3 (cont'd)

a) Derive the WCET of `CalcXY` by using the analysis method proposed by Shaw. More specifically, show that the WCET can be expressed as $\mathbf{12 \cdot N - 2}$ when `CalcXY` is called from `main` with parameter $\mathbf{y = N}$, where $\mathbf{N}$ is an integer constant in the range $[2, 5]$. (1.5 points)

b) Derive the WCET of `CalcWZ` by using the analysis method proposed by Shaw. More specifically, show that the WCET can be expressed as $\mathbf{16 \cdot L - 10}$ when `CalcWZ` is called from `main` with parameter $\mathbf{z = L}$, where $\mathbf{L}$ an integer constant in the range $[2, 5]$. (1.5 points)

c) Derive the WCET of `main` by using the analysis method proposed by Shaw. The WCET should be expressed as a function of the integer constants $\mathbf{N}$ and $\mathbf{L}$. (2 points)

d) Assuming that the value of integer constant $\mathbf{L = 4}$, identify the value of integer constant $\mathbf{N}$ (in the range $[2, 5]$) that results in the lowest WCET of `main`. Also state the resulting WCET of `main` for the identified value of integer constant $\mathbf{N}$. (1 point)

The following sub-problems concern scheduling concepts.

e) Describe the meaning of *Dhall's effect* in the context of global multiprocessor scheduling, and also explain why the effect occurs. (2 points)

f) Describe the difference between of a *periodic task* and a *sporadic task*. (1 point)

The following sub-problem concerns real-time programming.

g) The TinyTimber run-time scheduler uses a *system clock* to support timing-aware programming. Describe how the application programmer instructs the TinyTimber kernel to read the current value of the system clock. Also state the *resolution* (that is, the length of a time unit) of the TinyTimber system clock in the laboratory system used in the course. (1 point)

Advanced part

PROBLEM 4

The following questions are related to feasibility testing and time complexity of scheduling algorithms.

a) Explain the relevance of co-NP-complete problems in real-time scheduling theory. (1 point)

b) Consider a synchronous (all offsets O_i are identical) task set for which it applies that the deadline of each task does not exceed its period (that is, constrained-deadline tasks with $D_i \leq T_i$). Assume that the task set is known to be schedulable on a multiprocessor system using preemptive global scheduling and static task priorities. If one of the tasks in the task set would increase its offset by one time unit, keeping its other task parameters intact, would the task set still be schedulable on the multiprocessor system? Motivate your answer! (2 points)

c) Consider a synchronous (all offsets O_i are identical) task set for which it applies that the deadline of each task is equal to its period (that is, implicit-deadline tasks with $D_i = T_i$). Assume that the task set is known to be schedulable on a single-processor system using the rate-monotonic (RM) priority-assignment policy. State for each of the feasibility tests listed below what outcome ('True' or 'False') they would report if applied to the task set. Motivate your answer! (3 points)

- Response-time analysis
- Liu and Layland's utilization-based test for RM
- Liu and Layland's utilization-based test for EDF

PROBLEM 5

Consider a real-time system with four periodic tasks and a run-time system that employs preemptive single-processor scheduling using the deadline-monotonic (DM) priority-assignment approach. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for the four tasks. All tasks arrive at time $t = 0$.

	C_i	D_i	T_i
τ_1	2	6	6
τ_2	3	12	16
τ_3	3	15	20
τ_4	4	28	28

The four tasks are not independent, but share three exclusive resources R_a, R_b, and R_c. Assume that the run-time system employs the Immediate Ceiling Priority Protocol (ICPP) to resolve resource request conflicts. Each task uses the resources in the following way during each of its periodic invocations:

- Task τ_1 first requests R_a; and after releasing R_a, requests R_c; and then releases R_c.
- Task τ_2 first requests R_a; and after releasing R_a, requests R_b; and then releases R_b.
- Task τ_3 requests R_b, and then releases R_b.
- Task τ_4 requests R_c, and then releases R_c.

The table below shows $H_{i,j}$, the maximum time that task τ_i may lock resource R_j during its execution. Note: each resource use $H_{i,j}$ is assumed to be included in the normal execution time, C_i.

	$H_{i,a}$	$H_{i,b}$	$H_{i,c}$
τ_1	1	-	?
τ_2	1	2	-
τ_3	-	3	-
τ_4	-	-	?

As seen in the table above, the values for two of the parameters, $H_{1,c}$ and $H_{4,c}$, are not specified. Using *response-time analysis*, derive the largest integer values for these two parameters for which the task set is still schedulable. (10 points)

PROBLEM 6

Consider a real-time system with four independent periodic tasks and a run-time system that employs preemptive single-processor scheduling using the earliest-deadline-first (EDF) priority-assignment approach. The table below shows C_i (WCET), D_i (relative deadline) and T_i (period) for the four periodic tasks. All tasks arrive at time $t = 0$.

	C_i	D_i	T_i
τ_1	5	15	20
τ_2	5	55	60
τ_3	10	25	30
τ_4	5	20	20

It is known that the task set is schedulable on the given run-time system.

Using a suitable analysis method, perform a robustness analysis of the given task by assuming that a task τ_i can increase its execution time to $C_i \cdot \alpha_i$. The value of the scaling factor α_i is a positive real number. Derive, for each task τ_i, the maximum scaling factor α_i for which the task set is still schedulable when that, and only that, task increases its execution time (while assuming the original execution times for the other three tasks). (6 points)

PROBLEM 7

Consider a real-time system with three independent periodic tasks and a run-time system that employs preemptive *global scheduling* on $m = 2$ processors. The task priorities are given according to the deadline-monotonic (DM) priority-assignment approach. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for the three tasks. All tasks arrive at time $t = 0$.

	C_i	D_i	T_i
τ_1	3	5	6
τ_2	6	9	15
τ_3	6	7	9

Use a suitable analysis method to determine whether the deadlines are met or not for the three tasks in the system. (8 points)

Real-Time Systems — EDA223/DIT162

Re-exam, August 15, 2023 at 14:00 – 18:00 in the Saga building

---

Examiner:

Professor Jan Jonsson, Department of Computer Science and Engineering

Responsible teacher:

Jan Jonsson, phone: 031–772 5220

Visits the exam room at 15:00 and 17:00.

Aids permitted during the exam:

Chalmers-approved calculator

NOTE: Electronic dictionaries may not be used.

Content:

The **Basic part** of the exam consists of 3 problems worth a total of 30 points.

The **Advanced part** of the exam consists of 4 problems worth a total of 30 points.

Grading policy:

Obtaining less than 24 points in the Basic part leads to a **Fail** grade.

If you obtain at least 24 points in the Basic part, then your grade is determined by the total number of points in the Basic and Advanced parts combined as follows:

24–35 points ⇒ **grade 3**	24–43 points ⇒ **grade G** (GU)
36–47 points ⇒ **grade 4**	
48–60 points ⇒ **grade 5**	44–60 points ⇒ **grade VG** (GU)

Results:

When the grading is completed overall result statistics, and a time and location for inspection, will be announced on the course home page. Individual results will be available in Ladok.

Language:

Your solutions should be written in English

IMPORTANT ISSUES

1. Use separate sheets for each answered problem, and mark each sheet with the problem number.
2. Justify all answers. Lack of justification can lead to loss of credit even if the answer is correct.
3. Explain all calculations thoroughly. If justification and method is correct then simple calculation mistakes do not necessarily lead to loss of credit.
4. If some assumptions in a problem are missing or you consider that the made assumptions are unclear, then please state explicitly which assumptions you make in order to find a solution.
5. Write clearly! If we cannot read your solution, we will assume that it is wrong.

Good Luck!

Basic part

PROBLEM 1

The following sub-problem concerns processor-utilization analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the rate-monotonic (RM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	15	15
τ_2	2	25	25
τ_3	21	75	75

a) Perform Liu & Layland's utilization-based analysis for the given task set. (2 points)

The following sub-problem concerns response-time analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the deadline-monotonic (DM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	10	15
τ_2	2	8	25
τ_3	21	43	75

b) Perform response-time analysis for the given task set. The final (converged) response time should be calculated for each task in the task set, regardless of whether the response-time analysis for that task fails or not. (4 points)

PROBLEM 1 (cont'd)

The following sub-problems concern processor-demand analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the earliest-deadline-first (EDF) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	10	15
τ_2	2	8	25
τ_3	21	43	75

c) Show that the largest interval to examine is $L_{\max} = 52$ for the given task set. The solution should include calculations of the hyper-period upper bound as well as the upper bound proposed by Baruah, Rosier and Howell. (2 points)

d) Calculate the complete set of control points within $L_{\max} = 52$ for the given task set. (2 points)

e) Perform processor-demand analysis for the given task set, and the complete set of control points calculated in sub-problem d). The analysis should be performed for every control point in the set, regardless of whether the analysis in another control point fails or not. (4 points)

PROBLEM 2

The following sub-problems concern hyper-period analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling. The table below shows O_i (offset), C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i.

	O_i	C_i	D_i	T_i
τ_1	0	6	10	15
τ_2	4	2	8	25
τ_3	0	21	43	75

a) Simulate the execution of the task set using the deadline-monotonic (DM) priority-assignment policy, and construct a timing diagram that spans the entire hyper period. The arrival and deadline of each task instance should be clearly indicated in the diagram in the form of an up-arrow (arrival) and down-arrow (deadline), respectively. All task executions must be completed, regardless of whether a task instance misses its deadline or not. (5 points)

b) Derive a time table, corresponding to the timing diagram constructed in sub-problem a), that clearly states the start and stop times for each task instance (or task segment, if a task instance is preempted). (1 point)

PROBLEM 3

The following sub-problems concern analysis of worst-case execution time (WCET).

Consider the following C program code for the function `FuncY`:

```
int FuncY(int y) {
   int t;

   if (y == 0) {
      t = 0;
   } else {
      if (y == 1) {
         t = 1;
      } else {
         t = y + FuncY(y-1);
         t = t * t;
      }
   }

   return t;
}
```

Assume the following costs for the language statements:

- Each declaration and assignment statement costs 1 μs to execute.
- Each function call costs 2 μs plus WCET for the function in question.
- Each evaluation of the logical condition in an `if`- or `while`-statement costs 2 μs.
- Each add and subtract operation costs 3 μs.
- Each multiply operation costs 5 μs.
- Each return statement costs 2 μs.
- All other language constructs can be assumed to take 0 μs to execute.

a) Derive the WCET of `FuncY` by using the analysis method proposed by Shaw. More specifically, show that the WCET can be expressed as $\mathbf{22 \cdot y - 14}$ where $\mathbf{y}$ is the function parameter. Assume that `FuncY` is always called with parameter value $\mathbf{y > 0}$. (2 points)

b) Explain why is it preferred that WCET estimates for the program code of tasks in a real-time system are *pessimistic* as well as *tight*. (2 points)

PROBLEM 3 (cont'd)

The following sub-problems concern scheduling concepts.

c) Describe the meaning of *priority inversion* in the context of single-processor scheduling. (1.5 points)

d) Describe the difference between of a *periodic task* and a *sporadic task*. Your answer should include a timing diagram to illustrate the differences between the task types. (1.5 points)

The following sub-problems concern real-time programming.

e) Describe the meaning of *systematic time skew* in the context of implementation of periodic task executions. In addition, state the underlying reason why systematic time skew occurs, and also give an example of how it can be avoided. (1.5 points)

f) The TinyTimber run-time scheduler supports the concept of *concurrent programming*.

Name two different operations, one without any timing information and one containing timing information, in the TinyTimber kernel that the application programmer can use to create a concurrent execution of a given piece of program code. Also state what information is provided to the operations to identify the program code that should run concurrently. (1.5 points)

Advanced part

PROBLEM 4

The following questions are related to feasibility testing and time complexity of scheduling algorithms.

a) Describe the general meaning of a *critical instant* in the context of real-time scheduling. Also state the importance of the critical instant in preemptive single-processor scheduling. (1 point)

b) Consider a synchronous (all offsets O_i are identical) task set for which it applies that the deadline of each task does not exceed its period (that is, constrained-deadline tasks with $D_i \leq T_i$). Assume that the task set is known to be schedulable on a multiprocessor system using preemptive global scheduling and static task priorities. If one of the tasks in the task set would increase its offset by one time unit, keeping its other task parameters intact, would the task set still be schedulable on the multiprocessor system? Motivate your answer! (2 points)

c) The response-time analysis is known to be an NP-complete problem with pseudo-polynomial time complexity. Refer to the conditions that must apply for such a problem to have pseudo-polynomial time complexity, and show that response-time analysis fulfils these conditions, when assuming periodic task sets where all tasks arrive at time $t = 0$ and the deadline of a task does not exceed its period. (3 points)

PROBLEM 5

Consider a real-time system with two periodic tasks and a run-time system that employs preemptive rate-monotonic (RM) scheduling.

The relative deadline of each periodic task is equal to its period. Both tasks arrive at $t = 0$.

It is known that the tasks' utilizations $U_i = C_i/T_i$ are $U_1 = 0.75$ och $U_2 = 0.25$, respectively. However, the individual execution times C_i for the two tasks are not known.

a) If the tasks' periods T_i are not known either, is it possible to decide if the tasks are schedulable with RM? Motivate your answer. (3 points)

b) If we know that $T_2 = 2T_1$, is it then possible to decide whether the tasks are schedulable or not with RM? Motivate your answer. (3 points)

PROBLEM 6

Consider a real-time system with five tasks, whose timing properties are listed in the table below. Tasks τ_3 and τ_4 cooperate in the sense that τ_4 uses the result produced by τ_3, and they must both complete within a common end-to-end deadline of $D_{global} = 75$. The global deadline should be divided between these two tasks, and X represents the part of the deadline that is assigned to τ_3. To guarantee that valid data is available at the beginning of its execution, the offset for τ_4 is set to X. The same offset is also used for task τ_5. All other tasks are supposed to be independent, with an offset of 0.

	C_i	O_i	D_i	T_i
τ_1	5	0	6	15
τ_2	6	0	11	25
τ_3	9	0	X	75
τ_4	15	X	$75 - X$	75
τ_5	15	X	25	25

The tasks are assigned to two separate processors that are connected through a shared-bus network. Processor 1 executes tasks τ_1, τ_2 and τ_3, and employs preemptive earliest-deadline-first (EDF) scheduling (that is, dynamic priorities.) Processor 2 executes tasks τ_4 and τ_5, and employs preemptive deadline-monotonic (DM) scheduling (that is, static priorities.) The overhead for sending data between the processors over the network is assumed to be negligible.

Derive the range of allowed (maximum and minimum) values of X for which all tasks will meet their deadlines. (12 points)

PROBLEM 7

The following sub-problems are related to multiprocessor scheduling of independent periodic tasks.

a) Consider a task set with four independent periodic tasks and a run-time system that uses preemptive *global scheduling* on $m = 3$ processors. The task priorities are given according to the rate-monotonic (RM) policy. The table below shows C_i (WCET) and T_i (period) for each task in the task set. The relative deadline of each task is equal to its period. All tasks arrive at time $t = 0$.

	C_i	T_i
τ_1	10	100
τ_2	10	100
τ_3	10	100
τ_4	181	200

Show that is it is not possible to schedule the given task set on $m = 3$ processors such that all task deadlines are met, when using global scheduling with RM priorities. (3 points)

b) Show, by using a suitable *processor utilization test for global scheduling*, that there exists a better approach to assign static priorities to the tasks in sub-problem a) such that all task deadlines will be met when using global scheduling on $m = 3$ processors. (3 points)

Real-Time Systems — EDA223/DIT162

Final exam, March 11, 2024 at 08:30 – 12:30 in the SB2 building

---

Examiner:

Professor Jan Jonsson, Department of Computer Science and Engineering

Responsible teacher:

Jan Jonsson, phone: 031–772 5220

Visits the exam room at 09:30 and 11:30.

Aids permitted during the exam:

Chalmers-approved calculator

NOTE: Electronic dictionaries may not be used.

Content:

The **Basic part** of the exam consists of 3 problems worth a total of 30 points.

The **Advanced part** of the exam consists of 4 problems worth a total of 30 points.

Grading policy:

Obtaining less than 24 points in the Basic part leads to a **Fail** grade.

If you obtain at least 24 points in the Basic part, then your grade is determined by the total number of points in the Basic and Advanced parts combined as follows:

24–35 points ⇒ **grade 3**	24–43 points ⇒ **grade G** (GU)
36–47 points ⇒ **grade 4**	
48–60 points ⇒ **grade 5**	44–60 points ⇒ **grade VG** (GU)

Results:

When the grading is completed overall result statistics, and a time and location for inspection, will be announced on the course home page. Individual results will be available in Ladok.

Language:

Your solutions should be written in English

IMPORTANT ISSUES

1. Use separate sheets for each answered problem, and mark each sheet with the problem number.
2. Justify all answers. Lack of justification can lead to loss of credit even if the answer is correct.
3. Explain all calculations thoroughly. If justification and method is correct then simple calculation mistakes do not necessarily lead to loss of credit.
4. If some assumptions in a problem are missing or you consider that the made assumptions are unclear, then please state explicitly which assumptions you make in order to find a solution.
5. Write clearly! If we cannot read your solution, we will assume that it is wrong.

Good Luck!

Basic part

PROBLEM 1

The following sub-problem concerns processor-utilization analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the rate-monotonic (RM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	1	4	4
τ_2	3	15	15
τ_3	6	18	18

a) Perform Liu & Layland's utilization-based analysis for the given task set. (2 points)

The following sub-problem concerns response-time analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the deadline-monotonic (DM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	1	4	4
τ_2	3	10	15
τ_3	6	14	18

b) Perform response-time analysis for the given task set. The final (converged) response time should be calculated for each task in the task set, regardless of whether the response-time analysis for that task fails or not. (4 points)

PROBLEM 1 (cont'd)

The following sub-problems concern processor-demand analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the earliest-deadline-first (EDF) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	1	4	4
τ_2	3	10	15
τ_3	6	14	18

c) Show that the largest interval to examine is $L_{\max} = 14$ for the given task set. The solution should include calculations of the hyper-period upper bound as well as the upper bound proposed by Baruah, Rosier and Howell. (2 points)

d) Calculate the complete set of control points within $L_{\max} = 14$ for the given task set. (2 points)

e) Perform processor-demand analysis for the given task set, and the complete set of control points calculated in sub-problem d). The analysis should be performed for every control point in the set, regardless of whether the analysis in another control point fails or not. (4 points)

PROBLEM 2

The following sub-problems concern hyper-period analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling. The table below shows O_i (offset), C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i.

	O_i	C_i	D_i	T_i
τ_1	2	1	4	4
τ_2	4	3	10	15
τ_3	0	6	14	18

a) Simulate the execution of the task set using the deadline-monotonic (DM) priority-assignment policy, and construct a timing diagram that spans an interval of the hyper period, ranging from $t = 0$ to $t = 14$. The arrival and deadline of each task instance should be clearly indicated in the diagram in the form of an up-arrow (arrival) and down-arrow (deadline), respectively. All task executions that fall within the interval must be completed, regardless of whether a task instance misses its deadline or not. (5 points)

b) Derive a time table, corresponding to the timing diagram constructed in sub-problem a), that clearly states the start and stop times for each task instance (or task segment, if a task instance is preempted). (1 point)

PROBLEM 3

The following sub-problems concern analysis of worst-case execution time (WCET).

Consider the following C program code for the function `ShiftXY`:

```
int ShiftXY(int x, int y) {
   int result;
   int xbit0;

   result = 1;

   while (x != 0) {
      xbit0 = x & 1;

      if (xbit0 == 1)
          result = result * y;

      x = x >> 1;
      y = y << 1;
   }

   return result;
}
```

Assume the following costs for the language statements:

- Each declaration and assignment statement costs 1 μs to execute.
- Each evaluation of the logical condition in an `if`- or `while`-statement costs 2 μs.
- Each bitwise–right–shift operation ('`>> 1`') costs 2 μs.
- Each bitwise–left–shift operation ('`<< 1`') costs 2 μs.
- Each bitwise–and operation ('`&`') costs 1 μs.
- Each multiply operation costs 5 μs.
- Each return statement costs 2 μs.
- All other language constructs can be assumed to take 0 μs to execute.

a) Derive the WCET of function `ShiftXY` by using the analysis method proposed by Shaw. More specifically, show that the WCET is **73** μs, when the function is called with parameter **x** in the value range [1, 14] and parameter **y** in the value range [7, 63]. (3 points)

b) What is the largest allowed value range [1, N] of parameter **x** if the WCET of function `ShiftXY` may not exceed **67** μs? Again, assume that parameter **y** has the value range [7, 63]. State clearly the value of the upper bound N of parameter **x**, and motivate your answer. (1 point)

PROBLEM 3 (cont'd)

The following sub-problems concern scheduling concepts.

c) Explain the concept of *priority ceiling* as used in access-control protocols for critical regions. Your answer should include a description of how the priority ceiling is derived as well as how it it used by the access-control protocol to achieve the desired protocol behavior. (1.5 points)

d) Describe the general meaning of a *critical instant* in the context of real-time scheduling. Also describe to what extent the critical instant can be *a priori* predicted for preemptive static-priority single-processor scheduling and global multiprocessor scheduling, respectively. (1.5 points)

The following sub-problem concerns real-time programming.

e) The TinyTimber run-time scheduler uses a *system clock* to support timing-aware programming.

Name one operation in the TinyTimber kernel that the application programmer can use to read the value of the system clock, and explain what the read clock value represents when using that operation. Also state the resolution (that is, the length of a time unit) of the system clock in the laboratory system used in the course. (1.5 points)

f) The TinyTimber run-time scheduler supports the concept of *concurrent programming*.

Name two different operations, one without any timing information and one containing timing information, in the TinyTimber kernel that the application programmer can use to create a concurrent execution of a given piece of program code. Also state what information is provided to the operations to identify the program code that should run concurrently. (1.5 points)

Advanced part

PROBLEM 4

The following questions are related to feasibility testing and time complexity of scheduling algorithms.

a) List the four steps in the general procedure used for proving that a decision problem, such as the NON-PREEMPTIVE SCHEDULING OF PERIODIC TASKS problem, is NP-complete. (2 points)

b) Describe the meaning of *Dhall's effect* in the context of global multiprocessor scheduling. In addition, state the underlying reason why Dhall's effect occurs, and also give an example of how the effect can be avoided. (3 points)

c) Consider a real-time system with independent periodic tasks and a run-time system that employs preemptive single-processor scheduling with static priorites using the deadline-monotonic (DM) priority-assignment approach. The task set is synchronous (all offsets O_i are identical), has constrained deadlines ($D_i \leq T_i$), and has the special property that $\forall i, j : D_i \neq D_j$.

Now assume that response-time analysis (RTA) is applied to the real-time system described above. Can we guarantee that a higher-priority task τ_j in the task set always meets its deadline according to RTA if a lower-priority task τ_i in the same task set meets its deadline according to RTA?

If your answer is 'Yes', provide a convincing argument supporting your standpoint. If your answer is 'No', provide a counter-example using a concrete task set. (1.5 points)

d) Consider three special cases of preemptive scheduling of periodic tasks with static priorities. For each of the special cases (**S1**, **S2**, **S3**) state the time complexity for the corresponding schedulability decision problem, by selecting one of the alternatives (**C1**, **C2**, **C3**, **C4**, **C5**).

Note: only consider the special cases as they are formulated in this sub-problem; that is, do not assume any further restrictions regarding priority assignment or task parameters. (1.5 points)

Special cases of preemptive scheduling of periodic tasks with static priorities:

- **S1:** Global multiprocessor scheduling of synchronous, implicit-deadline tasks
- **S2:** Single-processor scheduling of synchronous, arbitrary-deadline tasks
- **S3:** Single-processor scheduling of synchronous, constrained-deadline tasks

Available time complexity alternatives:

- **C1:** Exponential time
- **C2:** NP-complete in the strong sense
- **C3:** Co-NP-complete in the strong sense
- **C4:** NP-complete with pseudo-polynomial time
- **C5:** Polynomial time

PROBLEM 5

Consider a real-time system in a control application, with three independent periodic tasks. The table below shows C_i (WCET) and T_i (period) for the three tasks. The relative deadline of each periodic task is equal to its period. All tasks arrive at time $t = 0$. The tasks are allowed to preeempt each other.

	C_i	T_i
τ_1	2	5
τ_2	4	13
τ_3	6	29

a) Assume a run-time system that employs preemptive single-processor scheduling using the rate-monotonic (RM) priority-assignment approach. Use a suitable analysis method to show that all deadlines of the tasks are met. (4 points)

b) Now, assume a run-time system with a cyclic executive that uses a time table. In order to generate a compact cyclic time table (that is, with as few task instances per cycle as possible) for the task set it would be preferred to be able to adjust one or more of the task periods. Luckily, the control properties of the application allows the periods of tasks τ_2 and τ_3 to deviate at most ± 3 time units from the original value given in the table above. The control properties of the application also dictate that the execution of task τ_1 must not experience any jitter. Therefore, the period of task τ_1 cannot be changed, and the task must always execute as soon as it arrives.

Choose a suitable version of the task set given above (possibly modifying the period for tasks τ_2 and/or τ_3 within the given bounds), and construct a compact cyclic time table for the execution of the task set version. Make sure that the chosen task set version is still schedulable if the original periods are being modified. Your solution should clearly indicate the start and stop times for each task instance (or task segment, if a task is preempted). In addition, the total length of your time table (in time units) should be given. (4 points)

PROBLEM 6

Consider a real-time system with three independent periodic tasks and a run-time system that employs preemptive single-processor scheduling using the earliest-deadline-first (EDF) priority-assignment approach. The table below shows C_i (WCET), D_i (relative deadline) and T_i (period) for the three tasks. For each task τ_i it applies that $C_i \leq D_i \leq T_i$. All tasks arrive at time $t = 0$.

	C_i	D_i	T_i
τ_1	4	5	10
τ_2	2	D_2	20
τ_3	4	25	40

Use a suitable analysis method to determine the smallest integer value of the deadline D_2 for which all tasks meet their deadlines. (6 points)

PROBLEM 7

The following sub-problems are related to multiprocessor scheduling of independent periodic tasks.

a) Consider a task set with six independent periodic tasks and a run-time system that uses preemptive *partitioned scheduling* on $m = 3$ processors. The task priorities and task-to-processor assignments are given according to the the rate-monotonic first-fit (RMFF) algorithm. The table below shows C_i (WCET) and T_i (period) for each task in the task set. The relative deadline of each task is equal to its period. All tasks arrive at time $t = 0$.

	C_i	T_i
τ_1	10	100
τ_2	10	100
τ_3	10	100
τ_4	141	200
τ_5	141	200
τ_6	141	200

Show that is it is not possible to schedule the given task set on $m = 3$ processors such that all task deadlines are met, when using the RMFF partitioned scheduling algorithm. (5 points)

b) Show that there exists a better task-to-processor assignment for the tasks in sub-problem a) such that all task deadlines will be met when using partitioned scheduling on $m = 3$ processors and giving task priorities according to the rate-monotonic (RM) policy. (3 points)

Real-Time Systems — EDA223/DIT162

Re-exam, August 20, 2024 at 14:00 – 18:00 in the Saga building

---

Examiner:

Professor Jan Jonsson, Department of Computer Science and Engineering

Responsible teacher:

Jan Jonsson, phone: 031–772 5220

Visits the exam room at 15:00 and 17:00.

Aids permitted during the exam:

Chalmers-approved calculator

NOTE: Electronic dictionaries may not be used.

Content:

The **Basic part** of the exam consists of 3 problems worth a total of 30 points.

The **Advanced part** of the exam consists of 4 problems worth a total of 30 points.

Grading policy:

Obtaining less than 24 points in the Basic part leads to a **Fail** grade.

If you obtain at least 24 points in the Basic part, then your grade is determined by the total number of points in the Basic and Advanced parts combined as follows:

24–35 points ⇒ **grade 3**	24–43 points ⇒ **grade G** (GU)
36–47 points ⇒ **grade 4**	
48–60 points ⇒ **grade 5**	44–60 points ⇒ **grade VG** (GU)

Results:

When the grading is completed overall result statistics, and a time and location for inspection, will be announced on the course home page. Individual results will be available in Ladok.

Language:

Your solutions should be written in English

IMPORTANT ISSUES

1. Use separate sheets for each answered problem, and mark each sheet with the problem number.
2. Justify all answers. Lack of justification can lead to loss of credit even if the answer is correct.
3. Explain all calculations thoroughly. If justification and method is correct then simple calculation mistakes do not necessarily lead to loss of credit.
4. If some assumptions in a problem are missing or you consider that the made assumptions are unclear, then please state explicitly which assumptions you make in order to find a solution.
5. Write clearly! If we cannot read your solution, we will assume that it is wrong.

Good Luck!

Basic part

PROBLEM 1

The following sub-problem concerns processor-utilization analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the rate-monotonic (RM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	15	15
τ_2	2	25	25
τ_3	20	75	75

a) Perform Liu & Layland's utilization-based analysis for the given task set. (2 points)

The following sub-problem concerns response-time analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the deadline-monotonic (DM) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	10	15
τ_2	2	8	25
τ_3	20	38	75

b) Perform response-time analysis for the given task set. The final (converged) response time should be calculated for each task in the task set, regardless of whether the response-time analysis for that task fails or not. (4 points)

PROBLEM 1 (cont'd)

The following sub-problems concern processor-demand analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling using the earliest-deadline-first (EDF) priority-assignment policy. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive at $t = 0$.

	C_i	D_i	T_i
τ_1	6	10	15
τ_2	2	8	25
τ_3	20	38	75

c) Show that the largest interval to examine is $L_{\max} = 53$ for the given task set. The solution should include calculations of the hyper-period upper bound as well as the upper bound proposed by Baruah, Rosier and Howell. (2 points)

d) Calculate the complete set of control points within $L_{\max} = 53$ for the given task set. (2 points)

e) Perform processor-demand analysis for the given task set, and the complete set of control points calculated in sub-problem d). The analysis should be performed for every control point in the set, regardless of whether the analysis in another control point fails or not. (4 points)

PROBLEM 2

The following sub-problems concern hyper-period analysis.

Consider a real-time system with three periodic tasks and a run-time system that employs preemptive single-processor scheduling. The table below shows O_i (offset), C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i.

	O_i	C_i	D_i	T_i
τ_1	0	6	10	15
τ_2	4	2	8	25
τ_3	0	20	38	75

a) Simulate the execution of the task set using the earliest-deadline-first (EDF) priority-assignment policy, and construct a timing diagram that spans the entire hyper period. The arrival and deadline of each task instance should be clearly indicated in the diagram in the form of an up-arrow (arrival) and down-arrow (deadline), respectively. All task executions must be completed, regardless of whether a task instance misses its deadline or not. (5 points)

b) Derive a time table, corresponding to the timing diagram constructed in sub-problem a), that clearly states the start and stop times for each task instance (or task segment, if a task instance is preempted). (1 point)

PROBLEM 3

The following sub-problems concern analysis of worst-case execution time (WCET).

Consider the following C program code for the function `FuncZ`:

```
int FuncZ(int z) {
   int p;

   if (z == 0) {
      p = 0;
   } else {
      if (z == 1) {
         p = 1;
      } else {
         p = z + FuncZ(z-1);

         p = p * p;
      }
   }

   return p;
}
```

Assume the following costs for the language statements:

- Each declaration and assignment statement costs 1 μs to execute.
- Each function call costs 2 μs plus WCET for the function in question.
- Each evaluation of the logical condition in an `if`- or `while`-statement costs 2 μs.
- Each add and subtract operation costs 3 μs.
- Each multiply operation costs 5 μs.
- Each return statement costs 2 μs.
- All other language constructs can be assumed to take 0 μs to execute.

a) Derive the WCET of function `FuncZ` by using the analysis method proposed by Shaw. The WCET should be expressed in the form $\mathbf{k_1 \cdot z + k_2}$, where k_1 and k_2 are integer constants, and z is the function parameter. Assume that `FuncZ` is always called with parameter value $\mathbf{z > 0}$. (3 points)

b) Explain why is it preferred that WCET estimates for the program code of tasks in a real-time system are *tight*. (1 points)

PROBLEM 3 (cont'd)

The following sub-problems concern scheduling concepts.

c) Describe the meaning of *Dhall's effect* in the context of global multiprocessor scheduling, and also explain why the effect occurs. (2 points)

d) Consider a task set with independent periodic tasks, for which it applies that the deadline of each task is equal to its period. Assume that the task set is known to be schedulable on a single-processor system using static task priorities. Would a *necessary* feasibility test report the answer 'True' or 'False' when applied to that task set? Motivate your answer! (1 point)

The following sub-problems concern real-time programming.

e) Describe the meaning of *systematic time skew* in the context of implementation of periodic task executions. In addition, state the underlying reason why systematic time skew occurs, and also give an example of how it can be avoided. (1.5 points)

f) The TinyTimber run-time scheduler uses a *system clock* to support timing-aware programming.

Name <u>one</u> operation in the TinyTimber kernel that the application programmer can use to read the value of the system clock, and explain what the read clock value represents when using that operation. Also state the resolution (that is, the length of a time unit) of the system clock in the laboratory system used in the course. (1.5 points)

Advanced part

PROBLEM 4

The following questions are related to feasibility testing and time complexity of scheduling algorithms.

a) The rate-monotonic (RM) and earliest-deadline-first (EDF) priority-assignment policies are both optimal under similar assumptions regarding the task model on a single-processor system. Does this mean that the policies are equally good in terms of schedulability, or does one policy dominate the other? (1 point)

b) Consider a synchronous (all offsets O_i are identical) task set for which it applies that the deadline of each task does not exceed its period (that is, constrained-deadline tasks with $D_i \leq T_i$). What is the largest value of the common offset O_i for which pseudo-polynomial time complexity can be guaranteed for feasibility testing with response-time analysis on a single-processor system? Motivate your answer! (3 points)

PROBLEM 5

Consider a real-time system with three independent periodic tasks and a run-time system that uses preemptive single-processor scheduling using the rate-monotonic (RM) priority-assignment approach. The table below shows O_i (offset), C_i (WCET) and U_i (utilization) for the three tasks. The relative deadline of each periodic task is equal to its period. The value of the parameter C is not known.

	O_i	C_i	U_i
τ_1	0.4C	0.2C	1/3
τ_2	0	0.3C	3/8
τ_3	0	0.3C	1/4

Use a suitable analysis method to determine the schedulability of the tasks in the system. (8 points)

PROBLEM 6

Consider a real-time system with sporadic tasks and a run-time system that employs preemptive earliest-deadline-first (EDF) scheduling.

a) At a certain stage in the execution of the system, four sporadic tasks, τ_1, τ_2, τ_3 and τ_4, arrive according to the following scenario:

- At time $t = t_a$, task τ_2 arrives with an execution time $C_2 = 6$ and a relative deadline $D_2 = 14$.
- At time $t = t_a + 1$, task τ_4 arrives with an execution time $C_4 = 3$ and a relative deadline $D_4 = 18$.
- At time $t = t_a + 3$, task τ_3 arrives with an execution time $C_3 = 3$ and a relative deadline $D_3 = 13$.
- At time $t = t_a + 4$, task τ_1 arrives with an execution time $C_1 = 6$ and a relative deadline $D_1 = 8$.

Show, by constructing a timing diagram, that the task instances given above are schedulable using EDF. Assume, for simplicity, that the system is not currently executing any other task at $t = t_a$ and that no new instances of the four tasks arrive before $t = t_a + 30$. (4 points)

b) Show, by adding one more sporadic task arrival, τ_0, to the schedule, that all of the original four tasks ($\tau_1, \tau_2, \tau_3, \tau_4$) can be made to miss their deadlines when EDF scheduling is being used. Assume that the run-time system executes a task to its completion even if it has missed its deadline. (4 points)

c) Give an example of another priority assignment that would behave more stable than EDF in the situation described in sub-problem b). By 'more stable' we mean, in this context, that the priority assignment guarantees that at least one of the original tasks ($\tau_1, \tau_2, \tau_3, \tau_4$) meets its deadline despite the arrival of the new sporadic task τ_0. Again, assume that the run-time system executes a task to its completion even if it has missed its deadline. (4 points)

PROBLEM 7

The following sub-problems are related to multiprocessor scheduling of independent periodic tasks.

a) Consider a task set with four independent periodic tasks and a run-time system that uses preemptive *global scheduling* on $m = 3$ processors. The task priorities are given according to the rate-monotonic (RM) policy. The table below shows C_i (WCET) and T_i (period) for each task in the task set. The relative deadline of each task is equal to its period. All tasks arrive at time $t = 0$.

	C_i	T_i
τ_1	10	100
τ_2	10	100
τ_3	10	100
τ_4	181	200

Show that is it is not possible to schedule the given task set on $m = 3$ processors such that all task deadlines are met, when using global scheduling with RM priorities. (3 points)

b) Show, by using a suitable *processor utilization test for global scheduling*, that there exists a better approach to assign static priorities to the tasks in sub-problem a) such that all task deadlines will be met when using global scheduling on $m = 3$ processors. (3 points)

Real-Time Systems

Solutions to final exam March 14, 2022 (version 20240507)

Basic part

PROBLEM 1

a) The total utilization of the task set is $U = 6/18 + 3/27 + 19/54 \approx 0.796$.

Liu and Layland's utilization bound for 3 tasks is $\mathrm{U}_{\mathrm{RM}(3)} = 3 \cdot (2^{1/3} - 1) \approx 0.780$

The feasibility test fails, as $U > \mathrm{U}_{\mathrm{RM}(3)}$. Because the feasibility test is only sufficient, the schedulability of the task set cannot be determined.

---

b) Assuming the DM priority-assignment policy task τ_3 will have the lowest priority, since $D_3 = 31$ is larger than both $D_1 = 12$ and $D_2 = 10$. Task τ_2 will have highest priority, since $D_2 < D_1$

$R_2 = C_2 = 3 \leq D_2 = 10$

$R_1 = C_1 + \lceil \frac{R_1}{T_2} \rceil \cdot C_2$ [Assume that $R_1^0 = C_1 = 6$]

$R_1^1 = 6 + \lceil \frac{6}{27} \rceil \cdot 3 = 6 + 1 \cdot 3 = 9$

$R_1^2 = 6 + \lceil \frac{8}{27} \rceil \cdot 3 = 6 + 1 \cdot 3 = 9 \leq D_1 = 12$

$R_3 = C_3 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2$ [Assume that $R_3^0 = C_3 = 19$]

$R_3^1 = 19 + \lceil \frac{19}{18} \rceil \cdot 6 + \lceil \frac{19}{27} \rceil \cdot 3 = 19 + 2 \cdot 6 + 1 \cdot 3 = 34$

$R_3^2 = 19 + \lceil \frac{34}{18} \rceil \cdot 6 + \lceil \frac{34}{27} \rceil \cdot 3 = 19 + 2 \cdot 6 + 2 \cdot 3 = 37$

$R_3^3 = 19 + \lceil \frac{37}{18} \rceil \cdot 6 + \lceil \frac{37}{27} \rceil \cdot 3 = 19 + 3 \cdot 6 + 2 \cdot 3 = 43$

$R_3^4 = 19 + \lceil \frac{43}{18} \rceil \cdot 6 + \lceil \frac{43}{27} \rceil \cdot 3 = 19 + 3 \cdot 6 + 2 \cdot 3 = 43 > D_3 = 31$

The feasibility test for task τ_3 fails as the calculated response time of the task exceeds its deadline. Because the feasibility test is exact, the task set is <u>not</u> schedulable.

PROBLEM 1 (cont'd)

c) The total utilization of the task set is: $U = 6/18 + 3/27 + 19/54 \approx 0.796$

Since $U < 1$ the largest interval to examine is: $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,)$

For the given task set we have:

	U_i	$T_i - D_i$	$(T_i - D_i)\,U_i$
τ_1	6/18	6	2
τ_2	3/27	17	1.89
τ_3	19/54	23	8.09

$$L^* = \frac{\sum (T_i - D_i)\,U_i}{1 - U} = \frac{2 + 1.89 + 8.09}{1 - 0.796} = \frac{11.98}{0.204} \approx 58.7 \leq 59$$

The upper bound proposed by Baruah, Rosier and Howell:

$L_{\text{BRH}} = \max(D_1, D_2, D_3, L^*) = \max(12, 10, 31, 59) = 59$

The upper bound based on the hyper period:

$L_{\text{LCM}} = \text{LCM}\{18, 27, 54\} = 54$

Consequently, $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,) = \min(59, 54) = 54$

d) Derive the set K of control points in the interval $[\,0, 54\,]$

$K_1 = \{12, 30, 48\}$, $K_2 = \{10, 37\}$ and $K_3 = \{31\}$

Thus, $K = K_1 \cup K_2 \cup K_3 = \{10, 12, 30, 31, 37, 48\}$.

e) Perform processor-demand analysis for all tasks and all control points in K:

$N_i^L \cdot C_i = (\lfloor \frac{(L - D_i)}{T_i} \rfloor + 1) \cdot C_i$

L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
10	$(\lfloor \frac{(10-12)}{18} \rfloor + 1) \cdot 6 = 0$	$(\lfloor \frac{(10-10)}{27} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(10-31)}{54} \rfloor + 1) \cdot 19 = 0$	3	OK
12	$(\lfloor \frac{(12-12)}{18} \rfloor + 1) \cdot 6 = 6$	$(\lfloor \frac{(12-10)}{27} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(12-31)}{54} \rfloor + 1) \cdot 19 = 0$	9	OK
30	$(\lfloor \frac{(30-12)}{18} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(30-10)}{27} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(30-31)}{54} \rfloor + 1) \cdot 19 = 0$	15	OK
31	$(\lfloor \frac{(31-12)}{18} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(31-10)}{27} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(31-31)}{54} \rfloor + 1) \cdot 19 = 19$	34	FAIL
37	$(\lfloor \frac{(37-12)}{18} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(37-10)}{27} \rfloor + 1) \cdot 3 = 6$	$(\lfloor \frac{(37-31)}{54} \rfloor + 1) \cdot 19 = 19$	37	OK
48	$(\lfloor \frac{(48-12)}{18} \rfloor + 1) \cdot 6 = 18$	$(\lfloor \frac{(48-10)}{27} \rfloor + 1) \cdot 3 = 6$	$(\lfloor \frac{(48-31)}{54} \rfloor + 1) \cdot 19 = 19$	43	OK

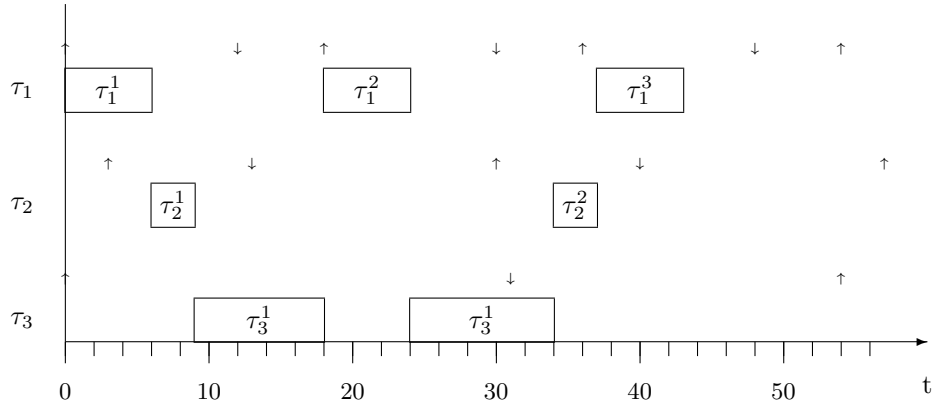
The feasibility test for $L = 31$ fails as the calculated processor demand in the interval exceeds the length of the interval. Because the feasibility test is exact, the task set is <u>not</u> schedulable.

The implication of this is that the first instance of task τ_3 will not meet its deadline at $t = 31$.

PROBLEM 2

The hyper period for the task set is $\mathrm{L_{LCM}} = 54$.

a) A simulation of the task set using EDF scheduling, in the interval $[\,0, 54\,]$, gives the following timing diagram.



The first instance of task τ_3 fails to meet its deadline at $t = 31$, and completes its execution at $t = 34$.

Note that the missed deadline at $t = 31$ corresponds to the failure in the processor-demand analysis for control point $L = 31$ in sub-problem 1e).

b) Time table corresponding to the timing diagram in sub-problem a):

$(\tau_1^1, 0, 6)$ $(\tau_2^1, 6, 9)$ $(\tau_3^1, 9, 18)$ $(\tau_1^2, 18, 24)$ $(\tau_3^1, 24, 34)$ $(\tau_2^2, 34, 37)$ $(\tau_1^3, 37, 43)$

PROBLEM 3

a) The WCET of `FuncA(a,b)` is derived based on two cases of the value of parameter b.

Case 1: $b = 0$:

$$
\begin{aligned}
&WCET(FuncA(a, b = 0)) = \\
&\{declare, p\} + \{declare, i\} + \{assign, p\} + \{assign, i\} + \{compare, b == 0\} + \{return, 1\} \\
&= 1 + 1 + 1 + 1 + 2 + 2 = \mathbf{8}
\end{aligned}
$$

Case 2: $b > 0$:

The WCET of `FuncA` for this case largely depends on the number of times the `while` loop executes.

Let $WCET(whileLoop, b)$ denote the WCET of the `while` loop as a function of parameter b. For the given start value of i the logical condition of the loop will be evaluated b times, which means that the body of the loop will execute $(b - 1)$ times. The value of $WCET(whileLoop, b)$ is thus

$$
\begin{aligned}
&WCET(whileLoop, b) = \\
&b \cdot \{compare, i > 1\} + \\
&(b - 1) \cdot [\ \{multiply, p * a)\} + \{assign, p\} + \{sub, i - 1\} + \{assign, i\}\ ] \\
&= b \cdot 2 + (b - 1) \cdot [\ 5 + 1 + 3 + 1\ ] \\
&= 12 \cdot b - 10
\end{aligned}
$$

The WCET of `FuncA` for the case $b > 0$ is then:

$$
\begin{aligned}
&WCET(FuncA(a, b > 0)) = \\
&\{declare, p\} + \{declare, i\} + \{assign, p\} + \{assign, i\} + \\
&\{compare, b == 0\} + WCET(whileLoop, b) + \{return, p\} \\
&= 1 + 1 + 1 + 1 + 2 + (12 \cdot b - 10) + 2 \\
&= \mathbf{12 \cdot b - 2}
\end{aligned}
$$

b) The WCET of `FuncB(c,d)` is derived based on two cases of the value of parameter d.

Case 1: $d = 1$:

$$
\begin{aligned}
&WCET(FuncB(c, d = 1)) = \\
&\{declare, p\} + \{compare, d == 1\} + \{assign, p\} + \{return, p\} \\
&= 1 + 2 + 1 + 2 = \mathbf{6}
\end{aligned}
$$

Case 2: $d > 1$:

$$
\begin{aligned}
&WCET(FuncB(c, d > 1)) = \\
&\{declare, p\} + \{compare, d == 1\} + \\
&\{sub, d - 1\} + \{call, FuncB(c, d - 1)\} + WCET(FuncB(c, d - 1)) + \\
&\{multiply, c * FuncB(c, d - 1)\} + \{assign, p\} + \{return, p\} \\
&= 1 + 2 + 3 + 2 + WCET(FuncB(c, d - 1)) + 5 + 1 + 2 \\
&= 16 + WCET(FuncB(c, d - 1)) = \{\text{recursive expansion}\} = (d - 1) \cdot 16 + 6 \\
&= \mathbf{16 \cdot d - 10}
\end{aligned}
$$

PROBLEM 3 (cont'd)

c) A priority ceiling is associated with a shared resource, and represents the priority of the task with highest static priority that may use that resource. By means of a resource access protocol, that accounts for the priority ceilings of shared resources, it is possible to avoid problems such as priority inversion and deadlock.

d) In the original priority-ceiling protocol (PCP), a task X's priority is raised only when a higher-priority task tries to acquire a resource that X has locked. In the immediate ceiling priority protocol (ICPP), a task X's priority is immediately raised to the ceiling priority of a resource when X locks the resource.

---

e) The baseline of a method execution in the TinyTimber kernel represents the earliest possible start time of that method execution. If a method is called with its baseline set to a time later than the current time of the method call, e.g. by means of an `AFTER()` or `SEND()` operation, the execution of the method will be delayed by the TinyTimber run-time scheduler until the time specified by the baseline at the earliest.

f) The application programmer may instruct the TinyTimber kernel that a method execution should be associated with a priority by specifying a deadline in a `BEFORE()` or `SEND()` operation. Since the TinyTimber run-time scheduler uses the earliest-deadline-first (EDF) policy to select among the ready tasks specifying a deadline for a method execution corresponds directly to assigning a priority to that execution.

Advanced part

PROBLEM 4

a) In order for a decision problem to belong to class NP it must be possible to verify the correctness of a solution to the problem in polynomial time.

The Traveling Salesman decision problem (TSP) asks: "Does there exist one tour among the cities in C having a total length of no more than B?". With this problem formulation it is enough to provide one solution that allows us to answer 'Yes' to the question by verifying, in polynomial time, that the provided solution fulfils the criteria.

The complement Traveling Salesman decision problem (TSP^c), on the other hand, asks: "Does every tour among the cities in C have a total length that exceeds B?". With this problem formulation it is necessary to verify that all possible solutions fulfil the criteria before we are allowed to answer 'Yes' to the question. Consequently, the verification can only be done in exponential time.

b) Any positive value of O_i will work, as long as the task set is synchronous. To see this, assume that all O_i would have the same positive (non-zero) value. Since the tasks are synchronous we are analysing the critical-instant scenario, even if the offsets are positive. Without loss of generality we can therefore analyse a task set with the same timing parameters, except that all offsets are $O_i = 0$. The time complexity of the response-time behaviour would be exactly the same since we are still analysing the critical-instant case. Therefore, since RTA has been shown to have pseudo-polynomial time complexity for the case where $O_i = 0$ (see Lecture #15), it will also have pseudo-polynomial time complexity for a task set with an arbitrary positive value of O_i (as long as the task set remains synchronous).

c) Yes, the task set will still be schedulable even if it becomes asynchronous. The reason is that the synchronous case is known to be the worst-case scenario (the critical instant) for the single-processor case, that is, the situation where the task response times are maximized. An asynchronous case then means that one or more tasks may have shorter response times than in the critical instant case.

PROBLEM 5

We start by observing that task τ_1 has a first arrival time that differs from that of the other tasks. This means that the use of a utilization-based or response-time-based schedulability test may become overly pessimistic IF there exists no point in time in the schedule where all tasks arrive at the same time. This, in turn, could mean that, should the test fail, the task set could potentially still be schedulable.

Our first candidate method for schedulability analysis is Liu and Layland's classic utilization-based test. For three tasks, the schedulability bound is $\mathrm{U}_{\mathrm{RM(3)}} = 3\,(2^{1/3} - 1) \approx 0.780$. Unfortunately, the accumulated task utilization, $U = 1/3 + 3/8 + 1/4 \approx 0.958$, significantly exceeds the guarantee bound, and the test (being only sufficient) does not provide any useful information.

Our second candidate method is response-time analysis. Since task periods are required by the analysis, we begin by deriving the period of each task: $T_i = C_i/U_i = C_i \cdot (U_i)^{-1}$

$T_1 = C_1 \cdot (U_1)^{-1} = 0.2C \cdot 3 = 0.6C$

$T_2 = C_2 \cdot (U_2)^{-1} = 0.3C \cdot 8/3 = 0.8C$

$T_3 = C_3 \cdot (U_3)^{-1} = 0.3C \cdot 4 = 1.2C$

Assuming RM scheduling, task τ_1 has highest priority (shortest period) and task τ_3 has lowest priority. We then calculate the response time of each task and compare it against the corresponding task deadline:

$R_1 = C_1 = 0.2C < D_1 = T_1 = 0.6C.$

$R_2 = C_2 + \lceil \frac{R_2}{T_1} \rceil \cdot C_1$. Assume that $R_2^0 = C_2 = 0.3C$:

$R_2^1 = 0.3C + \lceil \frac{0.3C}{0.6C} \rceil \cdot 0.2C = 0.3C + 1 \cdot 0.2C = 0.5C$

$R_2^2 = 0.3C + \lceil \frac{0.5C}{0.6C} \rceil \cdot 0.2T = 0.3C + 1 \cdot 0.2C = 0.5C < D_2 = T_2 = 0.8C$

$R_3 = C_3 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2$. Assume that $R_3^0 = C_3 = 0.3C$:

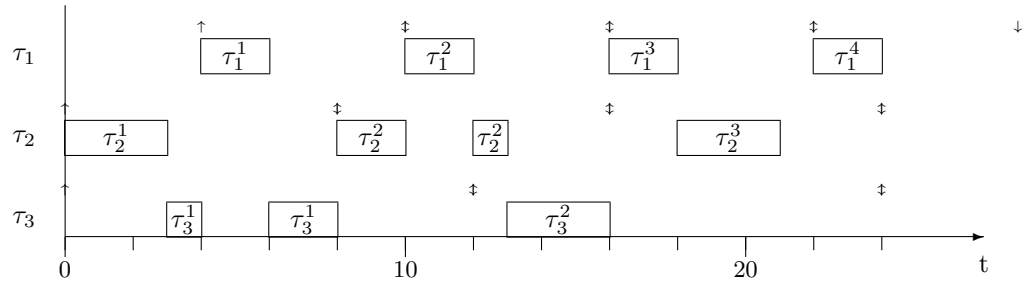
$R_3^1 = 0.3C + \lceil \frac{0.3C}{0.6C} \rceil \cdot 0.2C + \lceil \frac{0.3C}{0.8C} \rceil \cdot 0.3C = 0.3C + 1 \cdot 0.2C + 1 \cdot 0.3C = 0.8C$

$R_3^2 = 0.3C + \lceil \frac{0.8C}{0.6C} \rceil \cdot 0.2C + \lceil \frac{0.8C}{0.8C} \rceil \cdot 0.3C = 0.3C + 2 \cdot 0.2C + 1 \cdot 0.3C = 1.0C$

$R_3^3 = 0.3C + \lceil \frac{1.0C}{0.6C} \rceil \cdot 0.2C + \lceil \frac{1.0C}{0.8C} \rceil \cdot 0.3C = 0.3C + 2 \cdot 0.2C + 2 \cdot 0.3C = 1.3C > D_3 = T_3 = 1.2C$

The response-time analysis thus indicates that the worst-case response-time for τ_3 exceeds the task deadline. However, by observing the given periods and offsets (see time diagram below), we can see that there does NOT exist a point in time where all tasks arrive at the same time. This means that the worst-case response-time for τ_3 calculated by the response-time analysis will in fact never occur, and that it still is possible that τ_3 will meet its deadline. We will find out by using hyper-period analysis.

By simulating the RM schedule for one hyper period[1] we will see that <u>all tasks indeed meet their deadlines</u>. The hyper period for the given task set is LCM = 2.4C. Let C = 10 to get LCM = 24:



[1] Although we have an asynchronous task set, all task executions are completed within the first hyper period. This means that we only need to check schedulability within that interval.

PROBLEM 6

Perform processor-demand analysis:

For this problem we will use the largest interval $\mathrm{L_{max}} = \mathrm{L_{LCM}}$. The other bound, $\mathrm{L_{BRH}}$, would not be possible to determine as we will be looking for an unknown value of D_3 less than 30.

$\mathrm{L_{LCM}} = \mathrm{LCM}\{T_1, T_2, T_3\} = \mathrm{LCM}\{8, 16, 32\} = 32$.

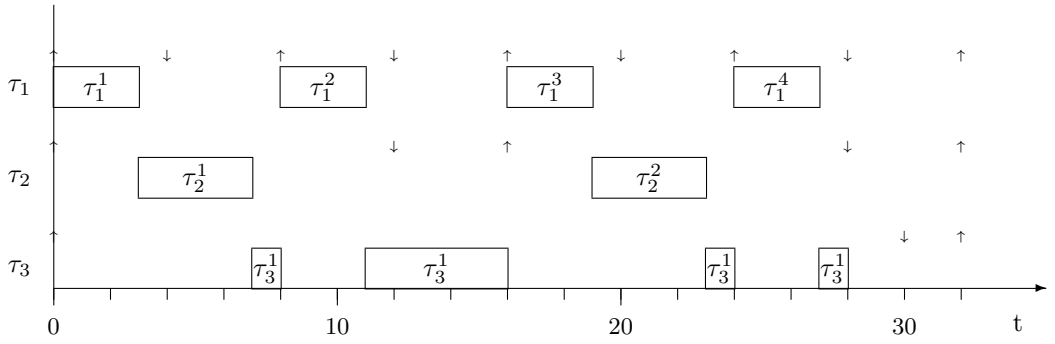
Then, derive the set K of control points: $K_1 = \{4, 12, 20, 28\}$, $K_2 = \{12, 28\}$ and $K_3 = \{30\}$ which gives us $K = K_1 \cup K_2 \cup K_3 = \{4, 12, 20, 28, 30\}$.

Schedulability analysis now gives us:

L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
4	$(\lfloor \frac{(4-4)}{8} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(4-12)}{16} \rfloor + 1) \cdot 4 = 0$	$(\lfloor \frac{(4-30)}{32} \rfloor + 1) \cdot 8 = 0$	3	OK
12	$(\lfloor \frac{(12-4)}{8} \rfloor + 1) \cdot 3 = 6$	$(\lfloor \frac{(12-12)}{16} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(12-30)}{32} \rfloor + 1) \cdot 8 = 0$	10	OK
20	$(\lfloor \frac{(20-4)}{8} \rfloor + 1) \cdot 3 = 9$	$(\lfloor \frac{(20-12)}{16} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(20-30)}{32} \rfloor + 1) \cdot 8 = 0$	13	OK
28	$(\lfloor \frac{(28-4)}{8} \rfloor + 1) \cdot 3 = 12$	$(\lfloor \frac{(28-12)}{16} \rfloor + 1) \cdot 4 = 8$	$(\lfloor \frac{(28-30)}{32} \rfloor + 1) \cdot 8 = 0$	20	OK
30	$(\lfloor \frac{(30-4)}{8} \rfloor + 1) \cdot 3 = 12$	$(\lfloor \frac{(30-12)}{16} \rfloor + 1) \cdot 4 = 8$	$(\lfloor \frac{(30-30)}{32} \rfloor + 1) \cdot 8 = 8$	28	OK

As expected all tasks meet their deadlines.

A simulation of the tasks using EDF scheduling in the interval $[0, \mathrm{LCM}]$ gives the following timing diagram. We see that, also here, all task meet their deadlines.

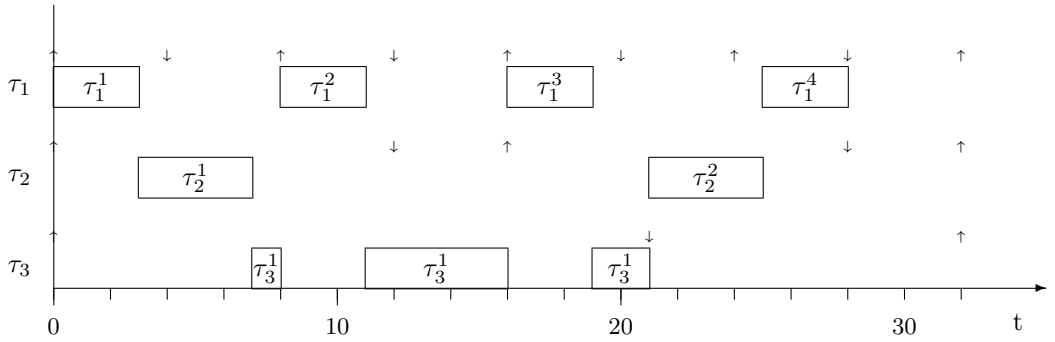


Both the processor-demand analysis and the timing diagram indicate that task τ_3 may decrease its deadline to $D_3 = 28$ without any task missing its deadline. Re-applying the processor-demand analysis after merging the new D_3 with the already-existing control point at $L = 28$ (and removing $L = 30$) verifies this:

L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
4	$(\lfloor \frac{(4-4)}{8} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(4-12)}{16} \rfloor + 1) \cdot 4 = 0$	$(\lfloor \frac{(4-28)}{32} \rfloor + 1) \cdot 8 = 0$	3	OK
12	$(\lfloor \frac{(12-4)}{8} \rfloor + 1) \cdot 3 = 6$	$(\lfloor \frac{(12-12)}{16} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(12-28)}{32} \rfloor + 1) \cdot 8 = 0$	10	OK
20	$(\lfloor \frac{(20-4)}{8} \rfloor + 1) \cdot 3 = 9$	$(\lfloor \frac{(20-12)}{16} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(20-28)}{32} \rfloor + 1) \cdot 8 = 0$	13	OK
28	$(\lfloor \frac{(28-4)}{8} \rfloor + 1) \cdot 3 = 12$	$(\lfloor \frac{(28-12)}{16} \rfloor + 1) \cdot 4 = 8$	$(\lfloor \frac{(28-28)}{32} \rfloor + 1) \cdot 8 = 8$	28	OK

This is, however, not the smallest possible value of D_3 that still makes the task set schedulable.

It is, in fact, possible for task τ_3 to decrease its deadline to $D_3 = 21$ without any task missing its deadline, as can be seen in the timing diagram below:



Re-applying the original processor-demand analysis with the new control point at $L = 21$ (replacing $L = 30$) verifies this:

L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
4	$(\lfloor \frac{(4-4)}{8} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(4-12)}{16} \rfloor + 1) \cdot 4 = 0$	$(\lfloor \frac{(4-21)}{32} \rfloor + 1) \cdot 8 = 0$	3	OK
12	$(\lfloor \frac{(12-4)}{8} \rfloor + 1) \cdot 3 = 6$	$(\lfloor \frac{(12-12)}{16} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(12-21)}{32} \rfloor + 1) \cdot 8 = 0$	10	OK
20	$(\lfloor \frac{(20-4)}{8} \rfloor + 1) \cdot 3 = 9$	$(\lfloor \frac{(20-12)}{16} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(20-21)}{32} \rfloor + 1) \cdot 8 = 0$	13	OK
21	$(\lfloor \frac{(21-4)}{8} \rfloor + 1) \cdot 3 = 9$	$(\lfloor \frac{(21-12)}{16} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(21-21)}{32} \rfloor + 1) \cdot 8 = 8$	21	OK
28	$(\lfloor \frac{(28-4)}{8} \rfloor + 1) \cdot 3 = 12$	$(\lfloor \frac{(28-12)}{16} \rfloor + 1) \cdot 4 = 8$	$(\lfloor \frac{(28-21)}{32} \rfloor + 1) \cdot 8 = 8$	28	OK

It is not possible to further decrease D_3. For example, choosing $D_3 = 20$ would cause either τ_1 or τ_3 to miss its deadline. The last processor-demand analysis above verifies this: there is only $20 - 13 = 7$ time units of slack available in control point $L = 20$, but adding one instance of τ_3 would require $C_3 = 8$ time units.

PROBLEM 7

We begin by calculating the utilization U_i for each task:

	C_i	T_i	U_i
τ_1	2	10	0.2
τ_2	2	27	0.074
τ_3	$T_3/4$	T_3	0.25
τ_4	15	37	0.405
τ_5	15	30	0.5

a) The Oh & Baker utilization guarantee bound for partitioned scheduling is $\mathrm{U_{RMFF}} = m(2^{1/2} - 1)$, where m is the number of processors.

For the given system, with $m = 2$, we have: $\mathrm{U_{RMFF}} = m \cdot (2^{1/2} - 1) = 2 \cdot (2^{1/2} - 1) \approx 0.828$

The total utilization of the task set is:

$\mathrm{U_{Total}} = 0.2 + 0.074 + 0.25 + 0.405 + 0.5 \approx 1.429$

Clearly, $\mathrm{U_{Total}} > \mathrm{U_{RMFF}}$ which means that the Oh & Baker test fails. However, since the test is only sufficient we cannot determine the schedulability of the task set.

b) Start by numbering the two processors μ_1 and μ_2.

According to the RMFF partitioning algorithm the tasks (temporarily excluding task τ_3, whose period is still unknown) should be assigned to the processors in the following order: $\tau_1, \tau_2, \tau_5, \tau_4$.

Based on this assignment order, we can see that three tasks can be assigned to processor μ_1, regardless of whether or not τ_3 is among those three tasks. The Liu & Layland utilization guarantee bound for three tasks is $\mathrm{U_{RM(3)}} = n \cdot (2^{1/n} - 1) = 3 \cdot (2^{1/3} - 1) \approx 0.780$.

We have two cases, based on the relation between the periods of τ_3 and τ_5:

Case 1 ($T_3 < T_5$): Task τ_3 is assigned to μ_1. The utilization of the three assigned tasks is then $\mathrm{U} = U_1 + U_2 + U_3 = 0.2 + 0.074 + 0.25 \approx 0.524$, which is less than $\mathrm{U_{RM(3)}}$.

Case 2 ($T_3 > T_5$): Task τ_5 is assigned to μ_1. The utilization of the three assigned tasks is then $\mathrm{U} = U_1 + U_2 + U_5 = 0.2 + 0.074 + 0.5 \approx 0.774$, which is also less than $\mathrm{U_{RM(3)}}$.

It is not possible to add a fourth task to processor μ_1 as the utilization of the assigned tasks would then exceed the corresponding Liu & Layland utilization guarantee bound. The remaining two tasks must therefore be assigned to processor μ_2. The Liu & Layland utilization guarantee bound for two tasks is $\mathrm{U_{RM(2)}} = n \cdot (2^{1/n} - 1) = 2 \cdot (2^{1/2} - 1) \approx 0.828$.

Again, looking at the two cases:

Case 1: The remaining two tasks to be assigned to μ_2 are tasks τ_5 and τ_4. The utilization of these tasks is $\mathrm{U} = U_5 + U_4 = 0.5 + 0.405 \approx 0.905$, which exceeds $\mathrm{U_{RM(2)}}$. This is consequently an infeasible assignment.

Case 2: The remaining two tasks to be assigned to μ_2 are tasks τ_3 and τ_4. The utilization of these tasks is $\mathrm{U} = U_3 + U_4 = 0.25 + 0.405 \approx 0.655$, which is less than $\mathrm{U_{RM(2)}}$. This is a feasible assignment.

Based on the reasoning above we saw that the task set can only be successfully scheduled on two processors if the period of task τ_3 is larger than the period of task τ_5, that is $T_3 > T_5 = 30$. The smallest integer value of $T_3 > 30$, that also satisfies the additional constraint that T_3 is evenly divisible by 4, is $T_3 = 32$.

The resulting assignment of tasks to processors is:

Processor μ_1: tasks τ_1, τ_2, and τ_5

Processor μ_2: tasks τ_3 and τ_4

Real-Time Systems

Solutions to re-exam August 16, 2022 (version 20240507)

Basic part

PROBLEM 1

a) The total utilization of the task set is $U = 6/15 + 2/25 + 21/75 = 0.760$.

Liu and Layland's utilization bound for 3 tasks is $\mathrm{U}_{\mathrm{RM}(3)} = 3 \cdot (2^{1/3} - 1) \approx 0.780$

The feasibility test succeeds, as $U < \mathrm{U}_{\mathrm{RM}(3)}$. Consequently, the task set is schedulable.

---

b) Assuming the DM priority-assignment policy task τ_3 will have the lowest priority, since $D_3 = 43$ is larger than both $D_1 = 10$ and $D_2 = 8$. Task τ_2 will have highest priority, since $D_2 < D_1$

$R_2 = C_2 = 2 \leq D_2 = 8$

$R_1 = C_1 + \lceil \frac{R_1}{T_2} \rceil \cdot C_2 \quad$ [Assume that $R_1^0 = C_1 = 6$]

$R_1^1 = 6 + \lceil \frac{6}{25} \rceil \cdot 2 = 6 + 1 \cdot 2 = 8$

$R_1^2 = 6 + \lceil \frac{8}{25} \rceil \cdot 2 = 6 + 1 \cdot 2 = 8 \leq D_1 = 10$

$R_3 = C_3 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2 \quad$ [Assume that $R_3^0 = C_3 = 21$]

$R_3^1 = 21 + \lceil \frac{21}{15} \rceil \cdot 6 + \lceil \frac{21}{25} \rceil \cdot 2 = 21 + 2 \cdot 6 + 1 \cdot 2 = 35$

$R_3^1 = 21 + \lceil \frac{35}{15} \rceil \cdot 6 + \lceil \frac{35}{25} \rceil \cdot 2 = 21 + 3 \cdot 6 + 2 \cdot 2 = 43$

$R_3^1 = 21 + \lceil \frac{43}{15} \rceil \cdot 6 + \lceil \frac{43}{25} \rceil \cdot 2 = 21 + 3 \cdot 6 + 2 \cdot 2 = 43 \leq D_3 = 43$

The feasibility test for each task succeeds, as the calculated response time for each task never exceeds its deadline. Consequently, the task set is schedulable.

PROBLEM 1 (cont'd)

c) The total utilization of the task set is: $U = 6/15 + 2/25 + 21/75 = 0.76$

Since $U < 1$ the largest interval to examine is: $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,)$

For the given task set we have:

	U_i	$T_i - D_i$	$(T_i - D_i)\,U_i$
τ_1	6/15	5	2
τ_2	2/25	17	1.36
τ_3	21/75	32	8.96

$$L^* = \frac{\sum (T_i - D_i)\,U_i}{1 - U} = \frac{2 + 1.36 + 8.96}{1 - 0.76} = \frac{12.32}{0.24} \approx 51.33 \leq 52$$

The upper bound proposed by Baruah, Rosier and Howell:

$L_{\text{BRH}} = \max(D_1, D_2, D_3, L^*) = \max(10, 8, 43, 52) = 52$

The upper bound based on the hyper period:

$L_{\text{LCM}} = \text{LCM}\{15, 25, 75\} = 75$

Consequently, $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,) = \min(52, 75) = 52$

d) Derive the set K of control points in the interval $[\,0, 52\,]$

$K_1 = \{10, 25, 40\}$, $K_2 = \{8, 33\}$ and $K_3 = \{43\}$

Thus, $K = K_1 \cup K_2 \cup K_3 = \{8, 10, 25, 33, 40, 43\}$

e) Perform processor-demand analysis for all tasks and all control points in K:

$N_i^L \cdot C_i = (\lfloor \frac{(L - D_i)}{T_i} \rfloor + 1) \cdot C_i$

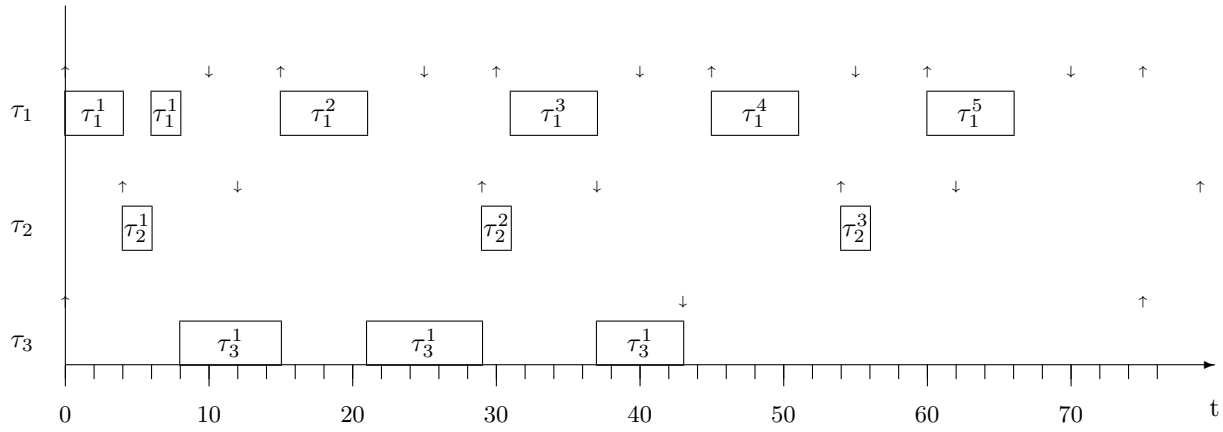
L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
8	$(\lfloor \frac{(8-10)}{15} \rfloor + 1) \cdot 6 = 0$	$(\lfloor \frac{(8-8)}{25} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(8-43)}{75} \rfloor + 1) \cdot 21 = 0$	2	OK
10	$(\lfloor \frac{(10-10)}{15} \rfloor + 1) \cdot 6 = 6$	$(\lfloor \frac{(10-8)}{25} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(10-43)}{75} \rfloor + 1) \cdot 21 = 0$	8	OK
25	$(\lfloor \frac{(25-10)}{15} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(25-8)}{25} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(25-43)}{75} \rfloor + 1) \cdot 21 = 0$	14	OK
33	$(\lfloor \frac{(33-10)}{15} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(33-8)}{25} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(33-43)}{75} \rfloor + 1) \cdot 21 = 0$	16	OK
40	$(\lfloor \frac{(40-10)}{15} \rfloor + 1) \cdot 6 = 18$	$(\lfloor \frac{(40-8)}{25} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(40-43)}{75} \rfloor + 1) \cdot 21 = 0$	22	OK
43	$(\lfloor \frac{(43-10)}{15} \rfloor + 1) \cdot 6 = 18$	$(\lfloor \frac{(43-8)}{25} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(43-43)}{75} \rfloor + 1) \cdot 21 = 21$	43	OK

The feasibility test for each control point succeeds, as the calculated processor demand for each interval never exceeds the length of the interval. Consequently, the task set is schedulable.

PROBLEM 2

The hyper period for the task set is $L_{\text{LCM}} = 75$.

a) A simulation of the task set using DM scheduling, in the interval $[\,0, 75\,]$, gives the following timing diagram.



Note that the worst-case response times of the tasks correspond to those derived in the response-time analysis in sub-problem 1b). Thus, for this task set, the offset of task τ_2 does not affect the worst-case response times of the other tasks.

b) Time table corresponding to the timing diagram in sub-problem a):

$(\tau_1^1, 0, 4)$ $(\tau_2^1, 4, 6)$ $(\tau_1^1, 6, 8)$ $(\tau_3^1, 8, 15)$ $(\tau_1^2, 15, 21)$ $(\tau_3^1, 21, 29)$ $(\tau_2^2, 29, 31)$ $(\tau_1^3, 31, 37)$
$(\tau_3^1, 37, 43)$ $(\tau_1^4, 45, 51)$ $(\tau_2^3, 54, 56)$ $(\tau_1^5, 60, 66)$

PROBLEM 3

a) The WCET of `FuncY(y)` is derived based on three cases of the value of parameter y.

Case 1: $y = 0$:

$$WCET(FuncY(y = 0)) =$$
$$\{declare, t\} + \{compare, y == 0\} + \{assign, t\} + \{return, t\}$$
$$= 1 + 2 + 1 + 2 = \mathbf{6}$$

Case 2: $y = 1$:

$$WCET(FuncY(y = 1)) =$$
$$\{declare, t\} + \{compare, y == 0\} + \{compare, y == 1\} +$$
$$\{assign, t\} + \{return, t\}$$
$$= 1 + 2 + 2 + 1 + 2 = \mathbf{8}$$

Case 3: $y > 1$:

$$WCET(FuncY(y > 1)) =$$
$$\{declare, t\} + \{compare, y == 0\} + \{compare, y == 1\} +$$
$$\{sub, y - 1\} + \{call, FuncY(y - 1)\} + WCET(FuncY(y - 1)) +$$
$$\{add, y + FuncY(y - 1)\} + \{assign, t\} + \{multiply, t * t\} + \{assign, t\} + \{return, t\}$$
$$= 1 + 2 + 2 + 3 + 2 + 3 + 1 + 5 + 1 + 2 + WCET(FuncY(y - 1))$$
$$= 22 + WCET(FuncY(y - 1)) = \{\text{recursive expansion}\} = (y - 1) \cdot 22 + 8$$
$$= \mathbf{22 \cdot y - 14}$$

b) WCET estimates must be pessimistic to make sure assumptions made in the schedulability analysis of hard real-time tasks also apply at run time. WCET estimates must be tight to avoid unnecessary pessimism in the schedulability analysis, which could cause feasibility tests to be too inaccurate to be useful.

PROBLEM 3 (cont'd)

c) Dhall's effect in the context of global multiprocessor scheduling refers to the following phenomenon. If a traditional single-processor priority-assignment approach, such as RM, is used with global scheduling it is possible to construct a task set, that is not schedulable although it has a very low utilization, regardless of how many processors are used.

The underlying reason for this phenomenon is that, with the RM approach, it is possible to construct a task set where the lowest priority is given to a task that have a long period while at the same time have a very high computational requirement in relation to its period (the "heavy" task). If there are as many highest-priority tasks, with short periods and very low computational requirement in relation to their periods (the "light" tasks), as there are processors the "heavy" task will not be able to fully utilize the available processor capacity in the best way, and will therefore miss its deadline.

d) The difference between the two types of tasks is as follows:

- For a periodic task τ_i the time interval between two subsequent arrivals of the task is guaranteed to be exactly T_i (the period parameter).
- For a sporadic task τ_i the time interval between two subsequent arrivals of the task is guaranteed to be higher than, or equal, to T_i (the period parameter).

---

e) When using a (relative) delay statement in the software to implement periodic executions the actual period becomes longer than the intended one, which in turn causes the each subsequent task arrival to drift further and further away from its intended arrival time. This drift is referred to systematic skew. The reason for this behaviour is that, with relative delay statements, the execution time of the calling method's code executed before/after the delay statement will be added to the requested delay.

f) The `AFTER()` operation takes advantage of the baseline mechanism in the TinyTimber kernel. The baseline of a method execution represents the earliest possible start time of that method execution. When a method calls itself by means of the `AFTER()` operation and an offset (the period), the next execution of the method will be delayed by the TinyTimber run-time scheduler until the time specified by the current baseline of the calling method plus the provided offset. In this way the execution time of the method's code does not influence the periodic behaviour.

Advanced part

PROBLEM 4

a) While the RM and EDF priority-assignment policies both make similar assumptions regarding the task model, they differ in the way they affect how the tasks are sorted in the ready queue in the run-time system. Depending on the system load the outcome will differ, as shown below:

Under normal load conditions it can be shown that there exist task sets that can be scheduled with dynamic priorities but cannot be scheduled with static priorities. This is because static priorities are a restricted subset of dynamic priorities. Thus, EDF dominates RM in terms of schedulability.

Under overload, however, not all task will be able to meet their deadlines. In such situations it can be shown that more tasks may miss their deadlines if scheduled with dynamic priorities than if they are scheduled with static priorities. To that end, RM dominates EDF when the performance goal is to maximise the number of tasks that meet their deadlines.

b) The task set may no longer be schedulable when it becomes asynchronous. The reason is that the synchronous case is not necessarily the worst-case scenario (the critical instant) for global scheduling on a multiprocessor system. In fact, it may turn out that the asynchronous case constitutes the critical instant, which means that one or more tasks may get longer response times in the asynchronous case than in the synchronous case. This, in turn, may cause one or more deadlines to be missed.

c) If we know that the task set is schedulable then the outcome from applying a feasibility test depends on the exactness of the test.

Response-time analysis: This is an exact feasibility test, which means that it will always report the outcome 'True' if the task set is known to be schedulable.

Liu and Layland's utilization-based test for RM: This is a sufficient test, which means that it can report either the outcome 'True' or the outcome 'False'. This is because a sufficient test can result in the outcome 'False' even though the task set is schedulable.

Liu and Layland's utilization-based test for EDF: This is an exact test, which means that it will always report the outcome 'True' if the task set is known to be schedulable.

Comment: although it is in general not a good idea to use a feasibility test designed for one priority-assignment policy (EDF) on a task set scheduled using another priority-assignment policy (RM), the question here was what outcome the EDF test would report if applied to the given task set. Since the assumptions are identical for both of Liu and Layland's feasibility tests (synchronous task set, deadline of each task equal to its period) the EDF test can be applied to the task set, and will also report the correct outcome for the given task set (since any task set that is schedulable on a uniprocessor must have a utilization that does not exceed 100%).

PROBLEM 5

a) If the exact values of the task periods are not known, we only have access to the individual task utilizations. It may then seem natural to try to apply Liu & Layland's sufficient utilization-based test. Unfortunately, the test's utilization bound for two tasks ($\approx 83\%$) is significantly lower than the actual utilization of the tasks ($= 100\%$), which means that the test does not tell us anything regarding schedulability. Neither can we apply any type of detailed analysis within a hyper-period since we do know know which task has the highest priority. And even if we make the assumption that τ_1 has the highest priority (and thereby meets all of its deadlines), we still cannot answer our question because $C_1/T_1 + C_2/T_2 = 1 \rightarrow T_2C_1 + T_1C_2 = T_1T_2 \rightarrow T_2 = T_2/T_1C_1 + C_2$. Here, we can see that, in order to decide whether τ_2 is schedulable or not, we must know how many instances of τ_1 that interferes with the execution of τ_2 within T_2. If T_2/T_1 is not an integer number there is a risk that task τ_2 misses a deadline (e.g. for $T_2 = 10$ and $T_1 = 4$).

b) With the new information that $T_2 = 2T_1$, it is now possible to decide schedulability. First, we now know with certainty that task τ_1 has highest priority according to RM and thereby meets its deadlines. In addition, we know from sub-problem (a) that $T_2 = T_2/T_1C_1 + C_2$. Therefore, we can conclude that $T_2 = 2C_1 + C_2$. That is, we now know that, within T_2, there is room for exactly two instances of task τ_1 and one instance of task τ_2. It is then clear that it is possible to do an analysis within one hyper-period ($= T_2$) since the system is not overloaded, and the behavior of the tasks will be repeated for each new hyper-period. And, since we already know that $T_2 = T_2/T_1C_1 + C_2$, we also know that task τ_2 will be able to execute C_2 time units within its period. The system is consequently schedulable.

PROBLEM 6

The easiest approach to solving this problem is to take advantage of the fact that the two processors are "isolated" from each other with respect to time. Since τ_4 and τ_5 have a common arrival time at $t = X$, this could be seen as the origin of these tasks' life cycles. And since τ_3 is assumed to have completed its execution no later than $t = X$, and the time to transfer its data to the other processor is assumed to be negligible, τ_3 and τ_4 need not synchronize its executions.

Processor 1 (EDF scheduling):

Since the task deadlines are shorter than the periods, we apply processor-demand analysis. We first derive the LCM for the tasks: $\text{LCM}\{\tau_1, \tau_2, \tau_3\} = \text{LCM}\{15, 25, 75\} = 75$.

We then calculate the set of control points K: $K_1 = \{6, 21, 36, 51, 66\}$, $K_2 = \{11, 36, 61\}$ och $K_3 = \{X\}$ which gives us $K = K_1 \cup K_2 \cup K_3 = \{6, 11, 21, 36, 51, 61, 66, X\}$.

Processor-demand analysis, including unknown control point X:

L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
6	$(\lfloor\frac{(6-6)}{15}\rfloor + 1) \cdot 5 = 5$	$(\lfloor\frac{(6-11)}{25}\rfloor + 1) \cdot 6 = 0$	$(\lfloor\frac{(6-X)}{75}\rfloor + 1) \cdot 9 = ?$	$5 + ?$	OK if $X > 6$
11	$(\lfloor\frac{(11-6)}{15}\rfloor + 1) \cdot 5 = 5$	$(\lfloor\frac{(11-11)}{25}\rfloor + 1) \cdot 6 = 6$	$(\lfloor\frac{(11-X)}{75}\rfloor + 1) \cdot 9 = ?$	$11 + ?$	OK if $X > 11$
21	$(\lfloor\frac{(21-6)}{15}\rfloor + 1) \cdot 5 = 10$	$(\lfloor\frac{(21-11)}{25}\rfloor + 1) \cdot 6 = 6$	$(\lfloor\frac{(21-X)}{75}\rfloor + 1) \cdot 9 = ?$	$16 + ?$	OK if $X > 21$
36	$(\lfloor\frac{(36-6)}{15}\rfloor + 1) \cdot 5 = 15$	$(\lfloor\frac{(36-11)}{25}\rfloor + 1) \cdot 6 = 12$	$(\lfloor\frac{(36-X)}{75}\rfloor + 1) \cdot 9 = ?$	$27 + ?$	OK
51	$(\lfloor\frac{(51-6)}{15}\rfloor + 1) \cdot 5 = 20$	$(\lfloor\frac{(51-11)}{25}\rfloor + 1) \cdot 6 = 12$	$(\lfloor\frac{(51-X)}{75}\rfloor + 1) \cdot 9 = ?$	$32 + ?$	OK
61	$(\lfloor\frac{(61-6)}{15}\rfloor + 1) \cdot 5 = 20$	$(\lfloor\frac{(61-11)}{25}\rfloor + 1) \cdot 6 = 18$	$(\lfloor\frac{(61-X)}{75}\rfloor + 1) \cdot 9 = ?$	$38 + ?$	OK
66	$(\lfloor\frac{(66-6)}{15}\rfloor + 1) \cdot 5 = 25$	$(\lfloor\frac{(66-11)}{25}\rfloor + 1) \cdot 6 = 18$	$(\lfloor\frac{(66-X)}{75}\rfloor + 1) \cdot 9 = ?$	$43 + ?$	OK

As can be seen from the table, the processor demand never exceeds the length of the interval for the known control points if $X > 21$. While $X = 36$ is a safe choice, it is possible to find a smaller value of X by observing that the processor demand $C_P(0, \text{X}) = 25$ for $21 < X < 36$. This means that we can add $D_3 = X \geq 25$ as the last control point. Hence, tasks τ_1, τ_2 and τ_3 will meet their deadlines if $X \geq 25$.

Processor 2 (DM scheduling):

We first observe that $t = X$ is the critical instant for tasks τ_4 and τ_5. We can therefore apply response-time analysis to determine a suitable value for X.

We first check whether $D_4 < 25$, which would mean that τ_4 has the highest priority. However, the response time for τ_5 would then become $R_5 = C_4 + C_5 = 15 + 15 = 30 > D_5$, causing τ_5 to miss its deadline.

Consequently, $D_4 \geq 25$, and we use response time analysis to find the smallest value of D_4. Assume $R_4^0 = C_4 = 15$.

$R_4^1 = 15 + \lceil\frac{15}{25}\rceil \cdot 15 = 15 + 1 \cdot 15 = 30$

$R_4^2 = 15 + \lceil\frac{30}{25}\rceil \cdot 15 = 15 + 2 \cdot 15 = 45$

$R_4^3 = 15 + \lceil\frac{45}{25}\rceil \cdot 15 = 15 + 2 \cdot 15 = 45$ (convergence)

Therefore, we have that $D_4 \geq 45$.

Since $D_4 = 75 - X$, we can now see that $X \leq 75 - 45 = 30$. Hence, tasks τ_4 and τ_5 will meet their deadlines if $X \leq 30$.

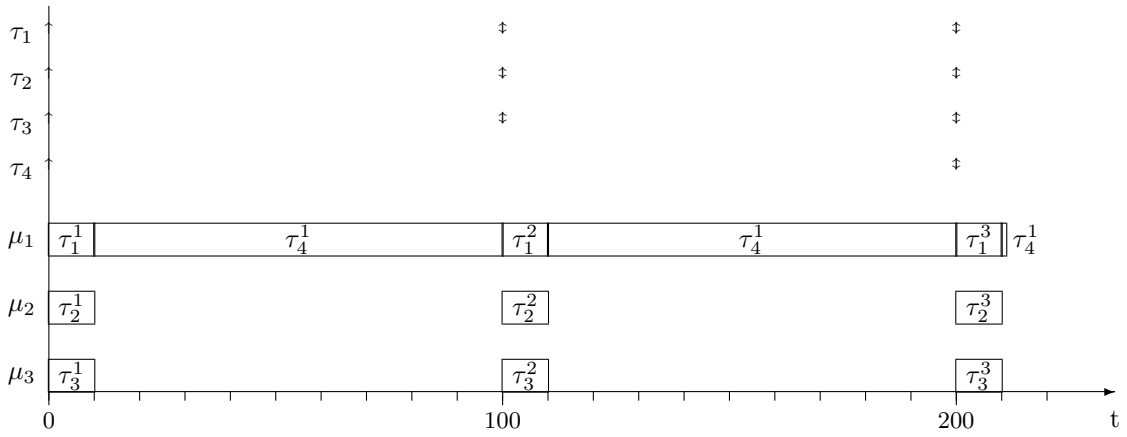
Conclusion: By combining the results from the analysis of processors 1 and 2, we see that all tasks will meet their deadlines if $25 \leq X \leq 30$

PROBLEM 7

a) Since rate-monotonic (RM) scheduling is used, the task priorities are as follows:

$prio(\tau_1) = \text{H}$, $prio(\tau_2) = \text{H}$, $prio(\tau_3) = \text{H}$, $prio(\tau_4) = \text{L}$.

We generate a multiprocessor schedule with tasks τ_1, τ_2 and τ_3 (having the highest priorities) running on one processor each. Task τ_4 is scheduled in the remaining time slots according to the following diagram (covering the first execution of τ_4):



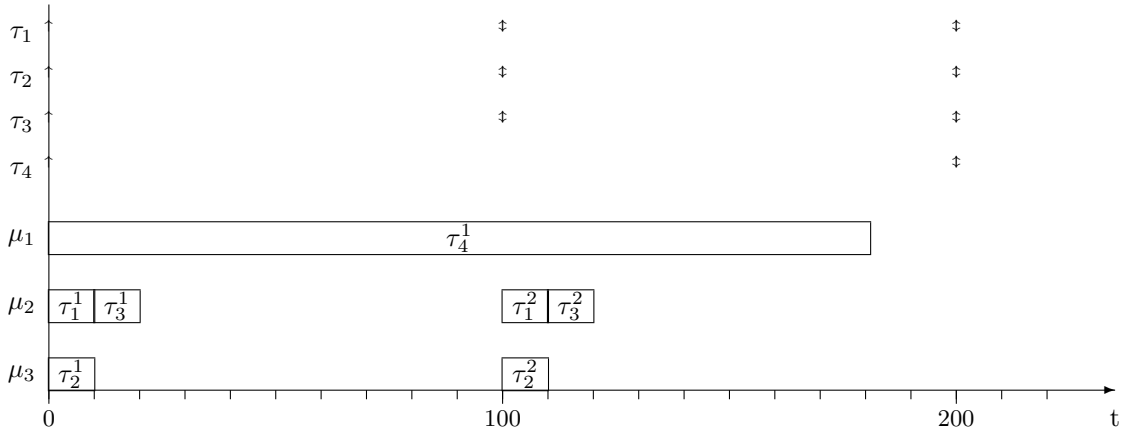
We observe that the first instance of task τ_4 completes its execution at $t = 211$ on processor μ_1, thereby missing its deadline at $t = T_4 = 200$. This happens despite there being significant processor capacity available on processors μ_2 and μ_3. A clear case of Dhall's effect!

b) If task priorities are given according to the rate-monotonic utilization-separation (RM-US) approach it may be possible to circumvent Dhall's effect, since that approach gives highest priority to "heavy" tasks in the task set. In order to make sure that task deadlines are met using the RM-US approach we need to verify that the total task utilization does not exceed the guarantee bound for RM-US.

The total utilization of the task set is $U_{\text{Total}} = U_1 + U_2 + U_3 + U_4 = 0.3 + 0.905 = 1.205$

The guarantee bound for RM-US is $U_{\text{RM-US}} = \frac{m^2}{3m-2}$. Since $m = 3$, $U_{\text{RM-US}} = 9/7 \approx 1.285$.

Consequently, by using the RM-US approach, all task deadlines for the task set in sub-problem a) will be met since $1.205 < 1.285$. The successful schedule in the hyper period $[0, 200]$ can be seen in the timing diagram below.



Real-Time Systems

Solutions to final exam March 13, 2023 (version 20240507)

Basic part

PROBLEM 1

a) The total utilization of the task set is $U = 2/6 + 4/14 + 4/20 \approx 0.819$.

Liu and Layland's utilization bound for 3 tasks is $\mathrm{U}_{\mathrm{RM}(3)} = 3 \cdot (2^{1/3} - 1) \approx 0.780$

The feasibility test fails, as $U > \mathrm{U}_{\mathrm{RM}(3)}$. Because the feasibility test is only sufficient, the schedulability of the task set cannot be determined.

---

b) Assuming the DM priority-assignment policy task τ_3 will have the lowest priority, since $D_3 = 12$ is larger than both $D_1 = 4$ and $D_2 = 11$. Task τ_1 will have highest priority, since $D_1 < D_2$

$R_1 = C_1 = 2 \leq D_1 = 4$

$R_2 = C_2 + \lceil \frac{R_2}{T_1} \rceil \cdot C_1$ [Assume that $R_2^0 = C_2 = 4$]

$R_2^1 = 4 + \lceil \frac{4}{6} \rceil \cdot 2 = 4 + 1 \cdot 2 = 6$

$R_2^2 = 4 + \lceil \frac{6}{6} \rceil \cdot 2 = 4 + 1 \cdot 2 = 6 \leq D_2 = 11$

$R_3 = C_3 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2$ [Assume that $R_3^0 = C_3 = 4$]

$R_3^1 = 4 + \lceil \frac{4}{6} \rceil \cdot 2 + \lceil \frac{4}{14} \rceil \cdot 4 = 4 + 1 \cdot 2 + 1 \cdot 4 = 10$

$R_3^1 = 4 + \lceil \frac{10}{6} \rceil \cdot 2 + \lceil \frac{10}{14} \rceil \cdot 4 = 4 + 2 \cdot 2 + 1 \cdot 4 = 12$

$R_3^1 = 4 + \lceil \frac{12}{6} \rceil \cdot 2 + \lceil \frac{12}{14} \rceil \cdot 4 = 4 + 2 \cdot 2 + 1 \cdot 4 = 12 \leq D_3 = 12$

The feasibility test for each task succeeds, as the calculated response time for each task never exceeds its deadline. Consequently, the task set is schedulable.

PROBLEM 1 (cont'd)

c) The total utilization of the task set is: $U = 2/6 + 4/14 + 4/20 \approx 0.819$

Since $U < 1$ the largest interval to examine is: $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,)$

For the given task set we have:

	U_i	$T_i - D_i$	$(T_i - D_i)\,U_i$
τ_1	2/6	2	0.667
τ_2	4/14	3	0.857
τ_3	4/20	8	1.6

$$L^* = \frac{\sum (T_i - D_i)\,U_i}{1 - U} = \frac{0.667 + 0.857 + 1.6}{1 - 0.819} = \frac{3.124}{0.181} \approx 17.26 \leq 18$$

The upper bound proposed by Baruah, Rosier and Howell:

$$L_{\text{BRH}} = \max(D_1, D_2, D_3, L^*) = \max(4, 11, 12, 18) = 18$$

The upper bound based on the hyper period:

$$L_{\text{LCM}} = \text{LCM}\{6, 14, 20\} = 420$$

Consequently, $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,) = \min(18, 420) = 18$

d) Derive the set K of control points in the interval $[\,0, 18\,]$

$K_1 = \{4, 10, 16\}$, $K_2 = \{11\}$ and $K_3 = \{12\}$

Thus, $K = K_1 \cup K_2 \cup K_3 = \{4, 10, 11, 12, 16\}$

e) Perform processor-demand analysis for all tasks and all control points in K:

$N_i^L \cdot C_i = (\lfloor \frac{(L - D_i)}{T_i} \rfloor + 1) \cdot C_i$

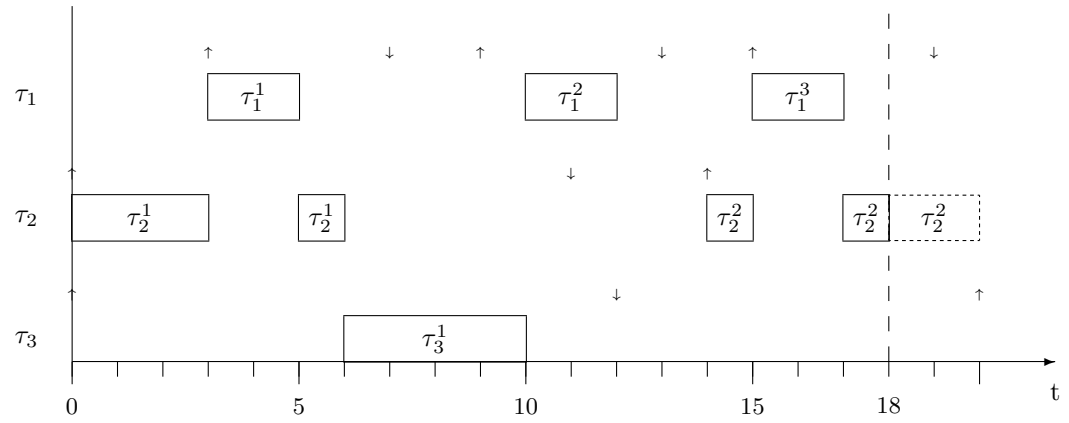
L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
4	$(\lfloor \frac{(4-4)}{6} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(4-11)}{14} \rfloor + 1) \cdot 4 = 0$	$(\lfloor \frac{(4-12)}{20} \rfloor + 1) \cdot 4 = 0$	2	OK
10	$(\lfloor \frac{(10-4)}{6} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(10-11)}{14} \rfloor + 1) \cdot 4 = 0$	$(\lfloor \frac{(10-12)}{20} \rfloor + 1) \cdot 4 = 0$	4	OK
11	$(\lfloor \frac{(11-4)}{6} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(11-11)}{14} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(11-12)}{20} \rfloor + 1) \cdot 4 = 0$	8	OK
12	$(\lfloor \frac{(12-4)}{6} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(12-11)}{14} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(12-12)}{20} \rfloor + 1) \cdot 4 = 4$	12	OK
16	$(\lfloor \frac{(16-4)}{6} \rfloor + 1) \cdot 2 = 6$	$(\lfloor \frac{(16-11)}{14} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(16-12)}{20} \rfloor + 1) \cdot 4 = 4$	14	OK

The feasibility test for each control point succeeds, as the calculated processor demand for each interval never exceeds the length of the interval. Consequently, the task set is schedulable.

PROBLEM 2

The hyper period for the task set is $L_{\text{LCM}} = 420$. However, we will only consider the interval $[\,0, 18\,]$.

a) A simulation of the task set using EDF scheduling, in the interval $[\,0, 18\,]$, gives the following timing diagram.



Note that task instance τ_1^3 preempts task instance τ_2^2 at time t=15, because the absolute deadline of τ_1^1 (at t=19) is earlier in time than the absolute deadline of τ_2^2 (at t=25, outside of the diagram). Also note that task instance τ_2^2 would continue executing beyond the boundaries of the interval, and complete at t=20.

b) Time table for the interval $[\,0, 18\,]$, corresponding to the timing diagram in sub-problem a):

$(\tau_2^1, 0, 3)$ $(\tau_1^1, 3, 5)$ $(\tau_2^1, 5, 6)$ $(\tau_3^1, 6, 10)$ $(\tau_1^2, 10, 12)$ $(\tau_2^2, 14, 15)$ $(\tau_1^3, 15, 17)$ $(\tau_2^2, 17, 18)$

PROBLEM 3

a) The WCET of `CalcXY(x,y)` is derived based on two cases of the value of parameter y.

Case 1: $y = 0$: (not applicable, since `CalcXY` is only called from `main` with values of $y > 0$)

$$
\begin{aligned}
&WCET(CalcXY(x, y = 0)) = \\
&\{declare, p\} + \{declare, i\} + \{assign, p\} + \{assign, i\} + \{compare, y == 0\} + \{return, 1\} \\
&= 1 + 1 + 1 + 1 + 2 + 2 = 8
\end{aligned}
$$

Case 2: $y > 0$:

The WCET of `CalcXY` for this case largely depends on the number of times the `while` loop executes.

Let $WCET(whileLoop, y)$ denote the WCET of the `while` loop as a function of parameter y. For the given start value of i the logical condition of the loop will be evaluated y times, which means that the body of the loop will execute $(y - 1)$ times. The value of $WCET(whileLoop, y)$ is thus

$$
\begin{aligned}
&WCET(whileLoop, y) = \\
&y \cdot \{compare, i > 1\} + \\
&(y - 1) \cdot [\ \{multiply, p * x)\} + \{assign, p\} + \{sub, i - 1\} + \{assign, i\}\ ] \\
&= y \cdot 2 + (y - 1) \cdot [\ 5 + 1 + 3 + 1\ ] \\
&= 12 \cdot y - 10
\end{aligned}
$$

The WCET of `CalcXY` for the case $y > 0$ is then:

$$
\begin{aligned}
&WCET(CalcXY(x, y > 0)) = \\
&\{declare, p\} + \{declare, i\} + \{assign, p\} + \{assign, i\} + \\
&\{compare, y == 0\} + WCET(whileLoop, y) + \{return, p\} \\
&= 1 + 1 + 1 + 1 + 2 + (12 \cdot y - 10) + 2 \\
&= 12 \cdot y - 2
\end{aligned}
$$

Consequently: when called from `main` with `y = N` the WCET of `CalcXY` is $\mathbf{12 \cdot N - 2}$.

b) The WCET of `CalcWZ(w,z)` is derived based on two cases of the value of parameter z.

Case 1: $z = 1$:

$$
\begin{aligned}
&WCET(CalcWZ(w, z = 1)) = \\
&\{declare, p\} + \{compare, z == 1\} + \{assign, p\} + \{return, p\} \\
&= 1 + 2 + 1 + 2 = 6
\end{aligned}
$$

Case 2: $z > 1$:

$$
\begin{aligned}
&WCET(CalcWZ(w, z > 1)) = \\
&\{declare, p\} + \{compare, z == 1\} + \\
&\{sub, z - 1\} + \{call, CalcWZ(w, z - 1)\} + WCET(CalcWZ(w, z - 1)) + \\
&\{multiply, c * CalcWZ(w, z - 1)\} + \{assign, p\} + \{return, p\} \\
&= 1 + 2 + 3 + 2 + WCET(CalcWZ(w, z - 1)) + 5 + 1 + 2 \\
&= 16 + WCET(CalcWZ(w, z - 1)) = \{\text{recursive expansion}\} = (z - 1) \cdot 16 + 6 \\
&= 16 \cdot z - 10
\end{aligned}
$$

Consequently: when called from `main` with `z = L` the WCET of `CalcWZ` is $\mathbf{16 \cdot L - 10}$.

c) The WCET of `main()` is derived based on the results from sub-problems a) and b).

$$
\begin{aligned}
&WCET(main()) = \\
&\{declare, x\} + \{declare, y\} + \\
&\{call, CalcXY(2, N)\} + WCET(CalcXY(2, N)) + \{assign, x\} + \\
&\{call, CalcWZ(2, L)\} + WCET(CalcWZ(2, L)) + \{assign, y\} + \\
&\{compare, y \neq x\} + \\
&\max(\{call, CalcWZ(3, L)\} + WCET(CalcWZ(3, L)) + \{assign, y\},\ \{sub, y - 3\} + \{assign, y\}) \\
&= 1 + 1 + 2 + WCET(CalcXY(2, N)) + 1 + 2 + WCET(CalcWZ(2, L)) + 1 + \\
&2 + \max(2 + WCET(CalcWZ(3, L)) + 1,\ 3 + 1) \\
&= 10 + \max(3 + WCET(CalcWZ(3, L)),\ 4) + WCET(CalcXY(2, N)) + WCET(CalcWZ(2, L))
\end{aligned}
$$

Knowing that `CalcXY` is called from `main` with $y = N$, where N is in the range $[2, 5]$, we have

$$WCET(CalcXY(2, N)) = WCET(CalcXY(x, y > 0)) = 12 \cdot y - 2 = \{y = N\} = 12 \cdot N - 2$$

Knowing that `CalcWZ` is called from `main` with $z = L$, where L is in the range $[2, 5]$, we have

$$WCET(CalcWZ(2, L)) = WCET(CalcWZ(w, z > 1)) = 16 \cdot z - 10 = \{z = L\} = 16 \cdot L - 10$$

$$WCET(CalcWZ(3, L)) = WCET(CalcWZ(w, z > 1)) = 16 \cdot z - 10 = \{z = L\} = 16 \cdot L - 10$$

The WCET of `main` can then be re-written as:

$$
\begin{aligned}
&WCET(main()) = \\
&10 + \max(3 + WCET(CalcWZ(3, L)), 4) + WCET(CalcXY(2, N)) + WCET(CalcWZ(2, L)) = \\
&10 + \max(3 + 16 \cdot L - 10, 4) + 12 \cdot N - 2 + 16 \cdot L - 10 = \\
&10 + \max(16 \cdot L - 7, 4) + 12 \cdot N - 2 + 16 \cdot L - 10 = \\
&\max(16 \cdot L - 7, 4) + 12 \cdot N + 16 \cdot L - 2
\end{aligned}
$$

Consequently: the WCET of `main` is $\max(\mathbf{16 \cdot L - 7}, \mathbf{4}) + \mathbf{12 \cdot N + 16 \cdot L - 2}$

d) Assuming `L = 4` means that the WCET of `main` can be expressed as

$$WCET(main()) = \max(16 \cdot 4 - 7, 4) + 12 \cdot N + 16 \cdot 4 - 2 = \max(57, 4) + 12 \cdot N + 62 = \mathbf{119 + 12 \cdot N}$$

However, for the special case `N = L`, there is actually a false path in function `main`. When `N = L` the condition `b != a` will never become true, and consequently the statement `b = CalcWZ(3, L)` will never be executed! To see this we observe that `CalcXY(x,y)` and `CalcWZ(w,z)` both implement the same mathematical function (exponentiation), albeit using different algorithmic approaches (iterative vs recursive) and different parameter names (x^y and w^z, respectively).

With this new knowledge the lowest value of the WCET of `main` will be

$$WCET(main()) = \max(0, 4) + 12 \cdot 4 + 16 \cdot 4 - 2 = 4 + 48 + 64 - 2 = \mathbf{114}$$

PROBLEM 3 (cont'd)

e) Dhall's effect in the context of global multiprocessor scheduling refers to the following phenomenon. If a traditional single-processor priority-assignment approach, such as RM, is used with global scheduling it is possible to construct a task set, that is not schedulable although it has a very low utilization, regardless of how many processors are used.

The underlying reason for this phenomenon is that, with the RM approach, it is possible to construct a task set where the lowest priority is given to a task that have a long period while at the same time have a very high computational requirement in relation to its period (the "heavy" task). If there are as many highest-priority tasks, with short periods and very low computational requirement in relation to their periods (the "light" tasks), as there are processors the "heavy" task will not be able to fully utilize the available processor capacity in the best way, and will therefore miss its deadline.

f) The difference between the two types of tasks is as follows:

- For a periodic task τ_i the time interval between two subsequent arrivals of the task is guaranteed to be exactly T_i (the period parameter).
- For a sporadic task τ_i the time interval between two subsequent arrivals of the task is guaranteed to be higher than, or equal, to T_i (the period parameter).

---

g) To read the current value of the system clock the programmer has two options:

- The `CURRENT_OFFSET` operation returns the time duration from the current baseline to the current time.
- The `T_SAMPLE` operation returns the time duration from a bookmarked baseline to the current baseline. A baseline of an earlier event is bookmarked by means of the `T_RESET` operation. Both operations should refer to a common object of type `Timer`.

The resolution of the TinyTimber system clock in the laboratory system is 10 μs.

Advanced part

PROBLEM 4

a) Many uniprocessor scheduling problems, such as scheduling of asynchronous tasks or scheduling of synchronous tasks using dynamic task priorities, cannot be shown to be NP-complete. Instead, their *complement* problem formulations can be shown to be reducible to known NP-complete problems, thus making the aforementioned uniprocessor scheduling problems co-NP-complete.

b) The task set may no longer be schedulable when it becomes asynchronous. The reason is that the synchronous case is not necessarily the worst-case scenario (the critical instant) for global scheduling on a multiprocessor system. In fact, it may turn out that the asynchronous case constitutes the critical instant, which means that one or more tasks may get longer response times in the asynchronous case than in the synchronous case. This, in turn, may cause one or more deadlines to be missed.

c) If we know that the task set is schedulable then the outcome from applying a feasibility test depends on the exactness of the test.

Response-time analysis: This is an exact feasibility test, which means that it will always report the outcome 'True' if the task set is known to be schedulable.

Liu and Layland's utilization-based test for RM: This is a sufficient test, which means that it can report either the outcome 'True' or the outcome 'False'. This is because a sufficient test can result in the outcome 'False' even though the task set is schedulable.

Liu and Layland's utilization-based test for EDF: This is an exact test, which means that it will always report the outcome 'True' if the task set is known to be schedulable.

Comment: although it is in general not a good idea to use a feasibility test designed for one priority-assignment policy (EDF) on a task set scheduled using another priority-assignment policy (RM), the question here was what outcome the EDF test would report if applied to the given task set. Since the assumptions are identical for both of Liu and Layland's feasibility tests (synchronous task set, deadline of each task equal to its period) the EDF test can be applied to the task set, and will also report the correct outcome for the given task set (since any task set that is schedulable on a uniprocessor must have a utilization that does not exceed 100%).

PROBLEM 5

Since deadline-monotonic scheduling is used, the static task priorities are as follows:

$prio(\tau_1) = \mathrm{H}$, $prio(\tau_2) = \mathrm{M_H}$, $prio(\tau_3) = \mathrm{M_L}$, $prio(\tau_4) = \mathrm{L}$.

We can now determine the ceiling priority for each resource:

$\mathrm{ceil}\{S_a\} = \max\{\mathrm{H}, \mathrm{M_H}\} = \mathrm{H}$ (since τ_1 och τ_2 may lock the resource)

$\mathrm{ceil}\{S_b\} = \max\{\mathrm{M_H}, \mathrm{M_L}\} = \mathrm{M_H}$ (since τ_2 och τ_3 may lock the resource)

$\mathrm{ceil}\{S_c\} = \max\{\mathrm{H}, \mathrm{L}\} = \mathrm{H}$ (since τ_1 och τ_4 may lock the resource)

We then identify, for each task τ_i, what tasks with lower priority may block τ_i and thereby cause the corresponding blocking factor B_i:

$B_1 = \max\{H_{2,a}, H_{4,c}\} = \max\{1, H_{4,c}\}$ (since τ_1 may be blocked by τ_2 and τ_4 who lock resources whose ceiling priorities are higher than or equal to the priority of τ_1)

$B_2 = \max\{H_{3,b}, H_{4,c}\} = \max\{3, H_{4,c}\}$ (since τ_2 may be blocked by τ_3 and τ_4 who lock resources whose ceiling priorities are higher than or equal to the priority of τ_2)

$B_3 = \max\{H_{4,c}\} = H_{4,c}$ (since τ_3 may be blocked by τ_4 who locks a resource whose ceiling priority is higher than or equal to the priority of τ_3)

$B_4 = 0$ (since τ_4 has lowest priority of all tasks, and thereby per definition cannot be subject to blocking)

We can now observe that the time $H_{1,c}$ does not affect the scheduling of the other tasks except through the execution time of τ_1. Since the time for using a resource is included in the normal execution time, we have $H_{1,c} \leq C_1 = 2$ (or, to be even more accurate, we have $H_{1,c} \leq C_1 - H_{1,a} = 2 - 1 = 1$).

On the other hand, the time $H_{4,c}$ does affect the schedulability of τ_1, τ_2 and τ_3 since that time is included in the blocking factors of these tasks. We therefore calculate the task response times and check whether they are less than or equal to the corresponding deadline:

$R_1 = C_1 + B_1 = 2 + H_{4,c} \leq D_1 = 6$.

This means that $H_{4,c} \leq D_1 - C_1 = 4$.

$R_2 = C_2 + B_2 + \lceil \frac{R_2}{T_1} \rceil \cdot C_1$. Assume that $B_2 = \max\{3, H_{4,c}\} = 4$ and $R_2^0 = C_2 = 3$:

$R_2^1 = 3 + 4 + \lceil \frac{3}{6} \rceil \cdot 2 = 3 + 4 + 1 \cdot 2 = 9$

$R_2^2 = 3 + 4 + \lceil \frac{9}{6} \rceil \cdot 2 = 3 + 4 + 2 \cdot 2 = 11$

$R_2^3 = 3 + 4 + \lceil \frac{11}{6} \rceil \cdot 2 = 3 + 4 + 2 \cdot 2 = 11 < D_2 = 12$.

This means that it still applies that $H_{4,c} \leq 4$.

$R_3 = C_3 + B_3 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2$. Assume that $B_3 = H_{4,c} = 4$ and $R_3^0 = C_3 = 3$:

$R_3^1 = 3 + 4 + \lceil \frac{3}{6} \rceil \cdot 2 + \lceil \frac{3}{16} \rceil \cdot 3 = 3 + 4 + 1 \cdot 2 + 1 \cdot 3 = 12$

$R_3^2 = 3 + 4 + \lceil \frac{12}{6} \rceil \cdot 2 + \lceil \frac{12}{16} \rceil \cdot 3 = 3 + 4 + 2 \cdot 2 + 1 \cdot 3 = 14$

$R_3^3 = 3 + 4 + \lceil \frac{14}{6} \rceil \cdot 2 + \lceil \frac{14}{16} \rceil \cdot 3 = 3 + 4 + 3 \cdot 2 + 1 \cdot 3 = 16 > D_3 = 15$

This means that our assumption that $H_{4,c} = 4$ does not hold. Instead, we try $H_{4,c} = 3$:

$R_3^1 = 3 + 3 + \lceil \frac{3}{6} \rceil \cdot 2 + \lceil \frac{3}{16} \rceil \cdot 3 = 3 + 3 + 1 \cdot 2 + 1 \cdot 3 = 11$

$R_3^2 = 3 + 3 + \lceil \frac{11}{6} \rceil \cdot 2 + \lceil \frac{11}{16} \rceil \cdot 3 = 3 + 3 + 2 \cdot 2 + 1 \cdot 3 = 13$

$R_3^3 = 3 + 3 + \lceil \frac{13}{6} \rceil \cdot 2 + \lceil \frac{13}{16} \rceil \cdot 3 = 3 + 3 + 3 \cdot 2 + 1 \cdot 3 = 15$

$R_3^4 = 3 + 3 + \lceil \frac{15}{6} \rceil \cdot 2 + \lceil \frac{15}{16} \rceil \cdot 3 = 3 + 3 + 3 \cdot 2 + 1 \cdot 3 = 15 \leq D_3 = 15$

This means that it must apply that $H_{4,c} \leq 3$.

$R_4 = C_4 + B_4 + \lceil \frac{R_4}{T_1} \rceil \cdot C_1 + \lceil \frac{R_4}{T_2} \rceil \cdot C_2 + \lceil \frac{R_4}{T_3} \rceil \cdot C_3$. Assume that $R_4^0 = C_4 = 4$:

$R_4^1 = 4 + \lceil \frac{4}{6} \rceil \cdot 2 + \lceil \frac{4}{16} \rceil \cdot 3 + \lceil \frac{4}{20} \rceil \cdot 3 = 4 + 1 \cdot 2 + 1 \cdot 3 + 1 \cdot 3 = 12$

$R_4^2 = 4 + \lceil \frac{12}{6} \rceil \cdot 2 + \lceil \frac{12}{16} \rceil \cdot 3 + \lceil \frac{12}{20} \rceil \cdot 3 = 4 + 2 \cdot 2 + 1 \cdot 3 + 1 \cdot 3 = 14$

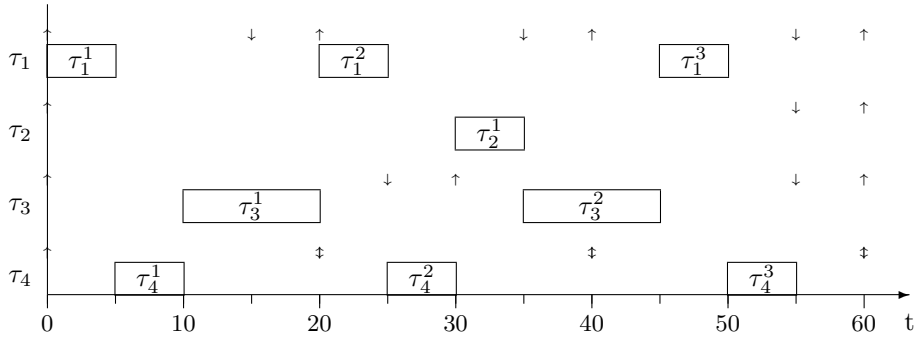
$R_4^3 = 4 + \lceil \frac{14}{6} \rceil \cdot 2 + \lceil \frac{14}{16} \rceil \cdot 3 + \lceil \frac{14}{20} \rceil \cdot 3 = 4 + 3 \cdot 2 + 1 \cdot 3 + 1 \cdot 3 = 16$

$R_4^4 = 4 + \lceil \frac{16}{6} \rceil \cdot 2 + \lceil \frac{16}{16} \rceil \cdot 3 + \lceil \frac{16}{20} \rceil \cdot 3 = 4 + 3 \cdot 2 + 1 \cdot 3 + 1 \cdot 3 = 16 < D_4 = 28$

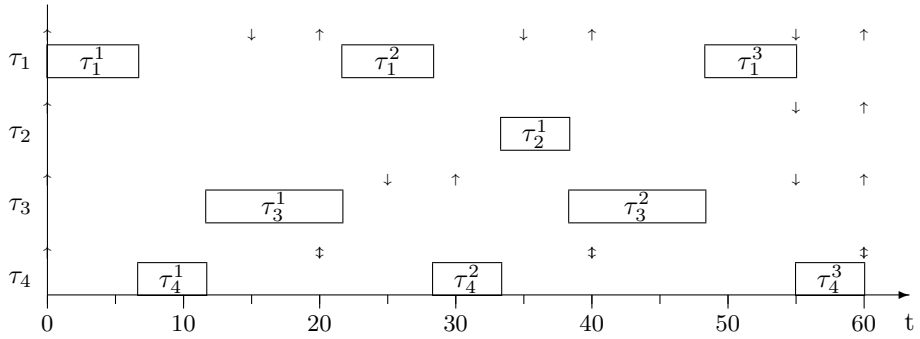
The system consequently meets all of its deadlines if $H_{1,c} \leq 1$ and $H_{4,c} \leq 3$.

PROBLEM 6

Simulating the execution of the tasks using earliest-deadline-first (EDF) scheduling within the hyper-period produces the following timing diagram:

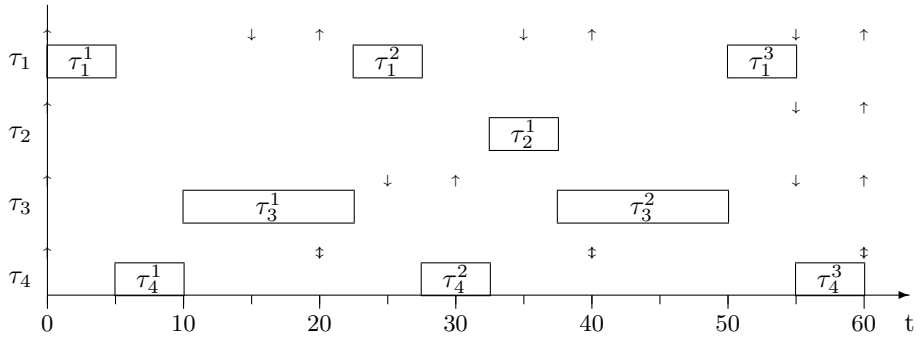


It is clear from this diagram that no task is closer to its deadline than five time units. This means that task τ_1 can scale up its execution time with a total of five time units distributed over the three instances of the task, that is, with $5/3 \approx 1.67$ time unit per task instance. This means that $\alpha_1 = (5 + 5/3)/5 = 20/15 = 4/3$.



Exactly the same reasoning applies for task τ_4, that is, $\alpha_4 = 4/3$.

It is also clear that task τ_3 can scale up its execution time with a total of five time units distributed over the two instances of the task, that is, with $5/2 = 2.5$ time unit per task instance. This means that $\alpha_3 = (10 + 5/2)/10 = 25/20 = 5/4$.



Finally, it is clear that task τ_2 can scale up its execution time with a total of five time units for its single instance. This means that $\alpha_2 = (5 + 5)/5 = 2$.

PROBLEM 7

There is only one utilization-based feasibility test for global scheduling, namely the one for RM-US. Since we assume scheduling using deadline-monotonic (DM) priority assignment the RM-US test cannot be used. We thus need to make a more detailed schedulability analysis.

We proceed by performing response-time analysis for global scheduling.

With deadline-monotonic (DM) priority assignment the static task priorities are as follows:

$prio(\tau_1) = \text{H}$, $prio(\tau_2) = \text{L}$, $prio(\tau_3) = \text{M}$.

With the given priorities tasks τ_1 and τ_3 will run undisturbed on one processor each, and are obviously schedulable since $C_i < D_i$ for these tasks. The response time of τ_2 is then derived using the following formula:

$R_2 = C_2 + \frac{1}{m}((\lceil \frac{R_2}{T_1} \rceil \cdot C_1 + C_1) + (\lceil \frac{R_2}{T_3} \rceil \cdot C_3 + C_3))$. Assume that $R_2^0 = C_2 = 6$:

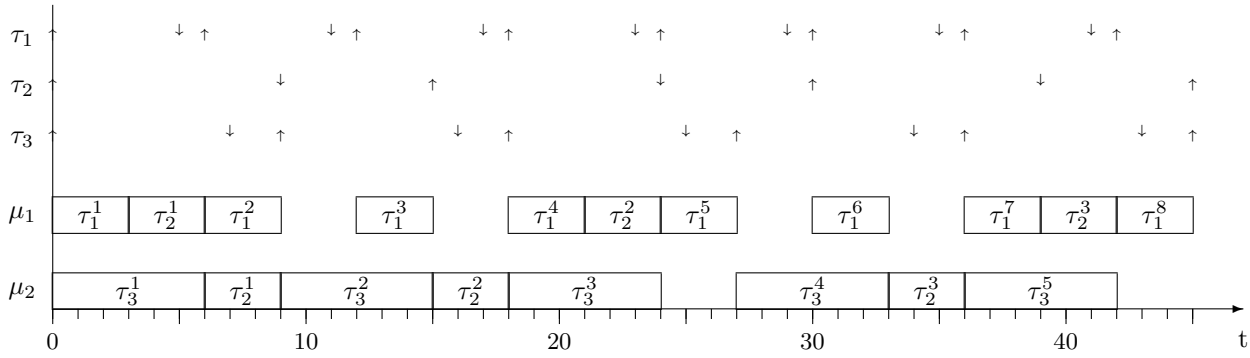
$R_2^1 = 6 + \frac{1}{2}((\lceil \frac{6}{6} \rceil \cdot 3 + 3) + (\lceil \frac{6}{9} \rceil \cdot 6 + 6)) = 6 + \frac{1}{2}((1 \cdot 3 + 3) + (1 \cdot 6 + 6)) = 6 + \frac{1}{2}(6 + 12) = 6 + 9 = 15$

The derived response-time of τ_2 exceeds the deadline D_2 $(= 9)$, so the test fails. Since the response-time test for global scheduling is only sufficient, we cannot yet determine the schedulability of the task set.

Our final option is then to use hyper-period analysis, which is an exact feasibility test:

$\text{LCM}\{T_1, T_2, T_3\} = \text{LCM}\{6, 15, 9\} = 90$.

We generate a multiprocessor schedule with tasks τ_1 and τ_3 (having the highest priorities) running on one processor each. Task τ_2 is scheduled in the remaining time slots according to the following timing diagram (covering the first half of the hyper period, that is $t = 0$ to $t = 45$):



We observe that there exists a critical instant at $t = 30$ in the diagram, relating to the third instance of task τ_2. Here, the response time of τ_2 is maximised, with $R_2 = 12$. Since $R_2 > D_2$ the task set is not schedulable.

Real-Time Systems

Solutions to re-exam August 15, 2023 (version 20240507)

Basic part

PROBLEM 1

a) The total utilization of the task set is $U = 6/15 + 2/25 + 21/75 = 0.760$.

Liu and Layland's utilization bound for 3 tasks is $\mathrm{U}_{\mathrm{RM}(3)} = 3 \cdot (2^{1/3} - 1) \approx 0.780$

The feasibility test succeeds, as $U < \mathrm{U}_{\mathrm{RM}(3)}$. Consequently, the task set is schedulable.

---

b) Assuming the DM priority-assignment policy task τ_3 will have the lowest priority, since $D_3 = 43$ is larger than both $D_1 = 10$ and $D_2 = 8$. Task τ_2 will have highest priority, since $D_2 < D_1$

$R_2 = C_2 = 2 \leq D_2 = 8$

$R_1 = C_1 + \lceil \frac{R_1}{T_2} \rceil \cdot C_2 \quad$ [Assume that $R_1^0 = C_1 = 6$]

$R_1^1 = 6 + \lceil \frac{6}{25} \rceil \cdot 2 = 6 + 1 \cdot 2 = 8$

$R_1^2 = 6 + \lceil \frac{8}{25} \rceil \cdot 2 = 6 + 1 \cdot 2 = 8 \leq D_1 = 10$

$R_3 = C_3 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2 \quad$ [Assume that $R_3^0 = C_3 = 21$]

$R_3^1 = 21 + \lceil \frac{21}{15} \rceil \cdot 6 + \lceil \frac{21}{25} \rceil \cdot 2 = 21 + 2 \cdot 6 + 1 \cdot 2 = 35$

$R_3^1 = 21 + \lceil \frac{35}{15} \rceil \cdot 6 + \lceil \frac{35}{25} \rceil \cdot 2 = 21 + 3 \cdot 6 + 2 \cdot 2 = 43$

$R_3^1 = 21 + \lceil \frac{43}{15} \rceil \cdot 6 + \lceil \frac{43}{25} \rceil \cdot 2 = 21 + 3 \cdot 6 + 2 \cdot 2 = 43 \leq D_3 = 43$

The feasibility test for each task succeeds, as the calculated response time for each task never exceeds its deadline. Consequently, the task set is schedulable.

PROBLEM 1 (cont'd)

c) The total utilization of the task set is: $U = 6/15 + 2/25 + 21/75 = 0.76$

Since $U < 1$ the largest interval to examine is: $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,)$

For the given task set we have:

	U_i	$T_i - D_i$	$(T_i - D_i)\,U_i$
τ_1	6/15	5	2
τ_2	2/25	17	1.36
τ_3	21/75	32	8.96

$$L^* = \frac{\sum (T_i - D_i)\,U_i}{1 - U} = \frac{2 + 1.36 + 8.96}{1 - 0.76} = \frac{12.32}{0.24} \approx 51.33 \leq 52$$

The upper bound proposed by Baruah, Rosier and Howell:

$L_{\text{BRH}} = \max(D_1, D_2, D_3, L^*) = \max(10, 8, 43, 52) = 52$

The upper bound based on the hyper period:

$L_{\text{LCM}} = \text{LCM}\{15, 25, 75\} = 75$

Consequently, $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,) = \min(52, 75) = 52$

d) Derive the set K of control points in the interval $[\,0, 52\,]$

$K_1 = \{10, 25, 40\}$, $K_2 = \{8, 33\}$ and $K_3 = \{43\}$

Thus, $K = K_1 \cup K_2 \cup K_3 = \{8, 10, 25, 33, 40, 43\}$

e) Perform processor-demand analysis for all tasks and all control points in K:

$N_i^L \cdot C_i = (\lfloor \frac{(L - D_i)}{T_i} \rfloor + 1) \cdot C_i$

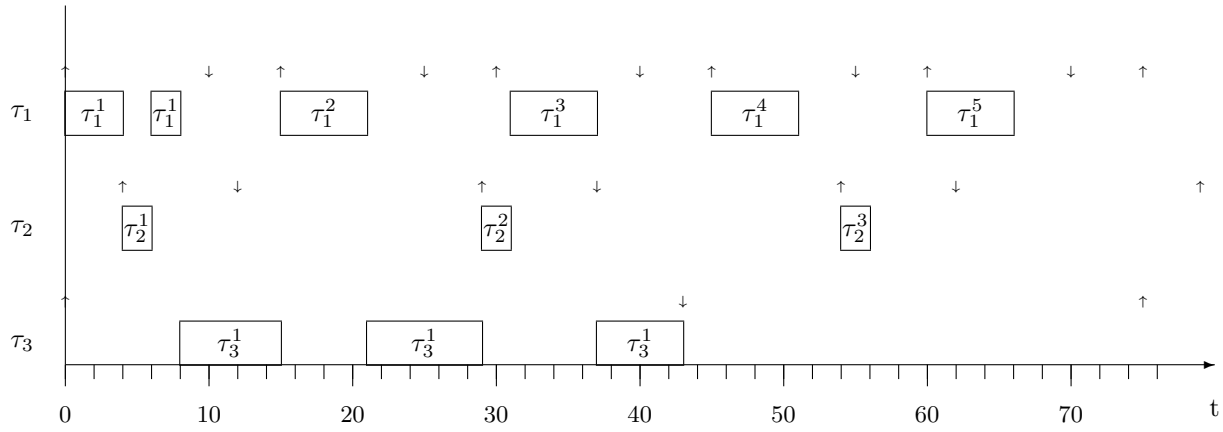
L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
8	$(\lfloor \frac{(8-10)}{15} \rfloor + 1) \cdot 6 = 0$	$(\lfloor \frac{(8-8)}{25} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(8-43)}{75} \rfloor + 1) \cdot 21 = 0$	2	OK
10	$(\lfloor \frac{(10-10)}{15} \rfloor + 1) \cdot 6 = 6$	$(\lfloor \frac{(10-8)}{25} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(10-43)}{75} \rfloor + 1) \cdot 21 = 0$	8	OK
25	$(\lfloor \frac{(25-10)}{15} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(25-8)}{25} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(25-43)}{75} \rfloor + 1) \cdot 21 = 0$	14	OK
33	$(\lfloor \frac{(33-10)}{15} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(33-8)}{25} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(33-43)}{75} \rfloor + 1) \cdot 21 = 0$	16	OK
40	$(\lfloor \frac{(40-10)}{15} \rfloor + 1) \cdot 6 = 18$	$(\lfloor \frac{(40-8)}{25} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(40-43)}{75} \rfloor + 1) \cdot 21 = 0$	22	OK
43	$(\lfloor \frac{(43-10)}{15} \rfloor + 1) \cdot 6 = 18$	$(\lfloor \frac{(43-8)}{25} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(43-43)}{75} \rfloor + 1) \cdot 21 = 21$	43	OK

The feasibility test for each control point succeeds, as the calculated processor demand for each interval never exceeds the length of the interval. Consequently, the task set is schedulable.

PROBLEM 2

The hyper period for the task set is $L_{\text{LCM}} = 75$.

a) A simulation of the task set using DM scheduling, in the interval $[\,0, 75\,]$, gives the following timing diagram.



Note that the worst-case response times of the tasks correspond to those derived in the response-time analysis in sub-problem 1b). Thus, for this task set, the offset of task τ_2 does not affect the worst-case response times of the other tasks.

b) Time table corresponding to the timing diagram in sub-problem a):

$(\tau_1^1, 0, 4)$ $(\tau_2^1, 4, 6)$ $(\tau_1^1, 6, 8)$ $(\tau_3^1, 8, 15)$ $(\tau_1^2, 15, 21)$ $(\tau_3^1, 21, 29)$ $(\tau_2^2, 29, 31)$ $(\tau_1^3, 31, 37)$
$(\tau_3^1, 37, 43)$ $(\tau_1^4, 45, 51)$ $(\tau_2^3, 54, 56)$ $(\tau_1^5, 60, 66)$

PROBLEM 3

a) The WCET of `FuncY(y)` is derived based on three cases of the value of parameter y.

Case 1: $y = 0$:

$$WCET(FuncY(y = 0)) =$$
$$\{declare, t\} + \{compare, y == 0\} + \{assign, t\} + \{return, t\}$$
$$= 1 + 2 + 1 + 2 = \mathbf{6}$$

Case 2: $y = 1$:

$$WCET(FuncY(y = 1)) =$$
$$\{declare, t\} + \{compare, y == 0\} + \{compare, y == 1\} +$$
$$\{assign, t\} + \{return, t\}$$
$$= 1 + 2 + 2 + 1 + 2 = \mathbf{8}$$

Case 3: $y > 1$:

$$WCET(FuncY(y > 1)) =$$
$$\{declare, t\} + \{compare, y == 0\} + \{compare, y == 1\} +$$
$$\{sub, y - 1\} + \{call, FuncY(y - 1)\} + WCET(FuncY(y - 1)) +$$
$$\{add, y + FuncY(y - 1)\} + \{assign, t\} + \{multiply, t * t\} + \{assign, t\} + \{return, t\}$$
$$= 1 + 2 + 2 + 3 + 2 + 3 + 1 + 5 + 1 + 2 + WCET(FuncY(y - 1))$$
$$= 22 + WCET(FuncY(y - 1)) = \{\text{recursive expansion}\}$$
$$= (y - 1) \cdot 22 + WCET(FuncY(1)) = (y - 1) \cdot 22 + 8$$
$$= \mathbf{22 \cdot y - 14}$$

b) WCET estimates must be *pessimistic* (that is, equal to or larger than the real WCET) to make sure assumptions made in the schedulability analysis of hard real-time tasks also apply at run time.

WCET estimates must be *tight* (that is, close to the real WCET) to avoid unnecessary pessimism in the schedulability analysis, which could cause feasibility tests to be too inaccurate to be useful.

PROBLEM 3 (cont'd)

c) By priority inversion we refer to a scenario where a higher-priority task cannot execute because another task holds a resource that the higher-priority task needs, and a lower-priority task (that does not use the resource) is able to execute instead.

d) The difference between the two types of tasks is as follows:

- For a periodic task τ_i the time interval between two subsequent arrivals of the task is guaranteed to be exactly T_i (the period parameter).
- For a sporadic task τ_i the time interval between two subsequent arrivals of the task is guaranteed to be higher than, or equal, to T_i (the period parameter).

---

e) Systematic skew is a phenomenon relating to periodic task executions, where the actual period becomes longer than the intended one, causing each subsequent task arrival to drift further and further away from its intended arrival time.

Systematic skew typically occurs in systems where periodic task executions have been implemented using the *relative* delay statement. With such a delay statement the execution time of the calling code before the delay statement will be added to the requested delay.

By using an *absolute* delay statement, specifying an actual time instant on the timeline for the next execution (instead of a delay relative to the current point of execution), the execution time of the calling code does not influence the periodic behaviour. This can be achieved by means of, for example, the `AFTER()` operation in TinyTimber, where a method's baseline is used to calculate a time instant on the timeline for the next execution of the method.

f) To create a concurrent execution the programmer should use the `ASYNC` operation or one of its associated timing-aware versions (`AFTER`, `SEND`, `BEFORE`), and provide as a parameter an object method identifying the program code to execute concurrently.

Note that the `SYNC` operation cannot be used to create a concurrent execution, as it causes the calling task to wait for the completion of the `SYNC` operation.

Advanced part

PROBLEM 4

a) A critical instant refers to a task arrival scenario in which the response time of a given task is maximized. This definition is valid for both single-processor and multiprocessor scheduling.

For preemptive single-processor scheduling using static priorities it can be shown that the critical instant occurs when the task arrives at the same time as all tasks with higher priority. This knowledge is a fundamental assumption in feasibility testing for single-processor scheduling using processor-utilization and response-time analysis.

b) The task set may no longer be schedulable when it becomes asynchronous. The reason is that the synchronous case is not necessarily the worst-case scenario (the critical instant) for global scheduling on a multiprocessor system. In fact, it may turn out that the asynchronous case constitutes the critical instant, which means that one or more tasks may get longer response times in the asynchronous case than in the synchronous case. This, in turn, may cause one or more deadlines to be missed.

c) The following conditions apply for pseudo-polynomial time complexity of an NP-complete algorithm:

- The algorithm must be a number problem, that is, the largest number (parameter value) in a problem instance cannot be bounded by the input length (size) of the problem.
- The time complexity of the number problem can be shown to be a polynomial-time function of both the input length and the largest number.

Assume a task set with n tasks. For task τ_i to be schedulable its WCET C_i must not exceed its deadline D_i. Consequently, C_i and D_i are both bounded above by the period T_i, which means that $\max_{\forall i}(T_i)$ is the largest number in the problem. Since each T_i can be chosen arbitrarily $\max_{\forall i}(T_i)$ is not bounded by n (the size of the problem).

Calculating the response-time for task τ_i requires at most D_i iterations of the algorithm. Each D_i is in turn bounded above by T_i. The time it takes to calculate the response times for <u>all</u> tasks is then bounded above by $n \cdot \max_{\forall i}(T_i)$, which is a polynomial-time function of both the input length and the largest number. Consequently, response-time analysis has pseudo-polynomial time complexity.

PROBLEM 5

a) If the exact values of the task periods are not known, we only have access to the individual task utilizations. It may then seem natural to try to apply Liu & Layland's sufficient utilization-based test. Unfortunately, the test's utilization bound for two tasks ($\approx 83\%$) is significantly lower than the actual utilization of the tasks ($= 100\%$), which means that the test does not tell us anything regarding schedulability. Neither can we apply any type of detailed analysis within a hyper-period since we do know know which task has the highest priority. And even if we make the assumption that τ_1 has the highest priority (and thereby meets all of its deadlines), we still cannot answer our question because $C_1/T_1 + C_2/T_2 = 1 \rightarrow T_2C_1 + T_1C_2 = T_1T_2 \rightarrow T_2 = T_2/T_1C_1 + C_2$. Here, we can see that, in order to decide whether τ_2 is schedulable or not, we must know how many instances of τ_1 that interferes with the execution of τ_2 within T_2. If T_2/T_1 is not an integer number there is a risk that task τ_2 misses a deadline (e.g. for $T_2 = 10$ and $T_1 = 4$).

b) With the new information that $T_2 = 2T_1$, it is now possible to decide schedulability. First, we now know with certainty that task τ_1 has highest priority according to RM and thereby meets its deadlines. In addition, we know from sub-problem (a) that $T_2 = T_2/T_1C_1 + C_2$. Therefore, we can conclude that $T_2 = 2C_1 + C_2$. That is, we now know that, within T_2, there is room for exactly two instances of task τ_1 and one instance of task τ_2. It is then clear that it is possible to do an analysis within one hyper-period ($= T_2$) since the system is not overloaded, and the behavior of the tasks will be repeated for each new hyper-period. And, since we already know that $T_2 = T_2/T_1C_1 + C_2$, we also know that task τ_2 will be able to execute C_2 time units within its period. The system is consequently schedulable.

PROBLEM 6

The easiest approach to solving this problem is to take advantage of the fact that the two processors are "isolated" from each other with respect to time. Since τ_4 and τ_5 have a common arrival time at $t = X$, this could be seen as the origin of these tasks' life cycles. And since τ_3 is assumed to have completed its execution no later than $t = X$, and the time to transfer its data to the other processor is assumed to be negligible, τ_3 and τ_4 need not synchronize its executions.

Processor 1 (EDF scheduling):

Since the task deadlines are shorter than the periods, we apply processor-demand analysis. We first derive the LCM for the tasks: $\text{LCM}\{\tau_1, \tau_2, \tau_3\} = \text{LCM}\{15, 25, 75\} = 75$.

We then calculate the set of control points K: $K_1 = \{6, 21, 36, 51, 66\}$, $K_2 = \{11, 36, 61\}$ och $K_3 = \{X\}$ which gives us $K = K_1 \cup K_2 \cup K_3 = \{6, 11, 21, 36, 51, 61, 66, X\}$.

Processor-demand analysis, including unknown control point X:

L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
6	$(\lfloor\frac{(6-6)}{15}\rfloor + 1) \cdot 5 = 5$	$(\lfloor\frac{(6-11)}{25}\rfloor + 1) \cdot 6 = 0$	$(\lfloor\frac{(6-X)}{75}\rfloor + 1) \cdot 9 = ?$	$5 + ?$	OK if $X > 6$
11	$(\lfloor\frac{(11-6)}{15}\rfloor + 1) \cdot 5 = 5$	$(\lfloor\frac{(11-11)}{25}\rfloor + 1) \cdot 6 = 6$	$(\lfloor\frac{(11-X)}{75}\rfloor + 1) \cdot 9 = ?$	$11 + ?$	OK if $X > 11$
21	$(\lfloor\frac{(21-6)}{15}\rfloor + 1) \cdot 5 = 10$	$(\lfloor\frac{(21-11)}{25}\rfloor + 1) \cdot 6 = 6$	$(\lfloor\frac{(21-X)}{75}\rfloor + 1) \cdot 9 = ?$	$16 + ?$	OK if $X > 21$
36	$(\lfloor\frac{(36-6)}{15}\rfloor + 1) \cdot 5 = 15$	$(\lfloor\frac{(36-11)}{25}\rfloor + 1) \cdot 6 = 12$	$(\lfloor\frac{(36-X)}{75}\rfloor + 1) \cdot 9 = ?$	$27 + ?$	OK
51	$(\lfloor\frac{(51-6)}{15}\rfloor + 1) \cdot 5 = 20$	$(\lfloor\frac{(51-11)}{25}\rfloor + 1) \cdot 6 = 12$	$(\lfloor\frac{(51-X)}{75}\rfloor + 1) \cdot 9 = ?$	$32 + ?$	OK
61	$(\lfloor\frac{(61-6)}{15}\rfloor + 1) \cdot 5 = 20$	$(\lfloor\frac{(61-11)}{25}\rfloor + 1) \cdot 6 = 18$	$(\lfloor\frac{(61-X)}{75}\rfloor + 1) \cdot 9 = ?$	$38 + ?$	OK
66	$(\lfloor\frac{(66-6)}{15}\rfloor + 1) \cdot 5 = 25$	$(\lfloor\frac{(66-11)}{25}\rfloor + 1) \cdot 6 = 18$	$(\lfloor\frac{(66-X)}{75}\rfloor + 1) \cdot 9 = ?$	$43 + ?$	OK

As can be seen from the table, the processor demand never exceeds the length of the interval for the known control points if $X > 21$. While $X = 36$ is a safe choice, it is possible to find a smaller value of X by observing that the processor demand $C_P(0, \text{X}) = 25$ for $21 < X < 36$. This means that we can add $D_3 = X \geq 25$ as the last control point. Hence, tasks τ_1, τ_2 and τ_3 will meet their deadlines if $X \geq 25$.

Processor 2 (DM scheduling):

We first observe that $t = X$ is the critical instant for tasks τ_4 and τ_5. We can therefore apply response-time analysis to determine a suitable value for X.

We first check whether $D_4 < 25$, which would mean that τ_4 has the highest priority. However, the response time for τ_5 would then become $R_5 = C_4 + C_5 = 15 + 15 = 30 > D_5$, causing τ_5 to miss its deadline.

Consequently, $D_4 \geq 25$, and we use response time analysis to find the smallest value of D_4. Assume $R_4^0 = C_4 = 15$.

$R_4^1 = 15 + \lceil\frac{15}{25}\rceil \cdot 15 = 15 + 1 \cdot 15 = 30$

$R_4^2 = 15 + \lceil\frac{30}{25}\rceil \cdot 15 = 15 + 2 \cdot 15 = 45$

$R_4^3 = 15 + \lceil\frac{45}{25}\rceil \cdot 15 = 15 + 2 \cdot 15 = 45$ (convergence)

Therefore, we have that $D_4 \geq 45$.

Since $D_4 = 75 - X$, we can now see that $X \leq 75 - 45 = 30$. Hence, tasks τ_4 and τ_5 will meet their deadlines if $X \leq 30$.

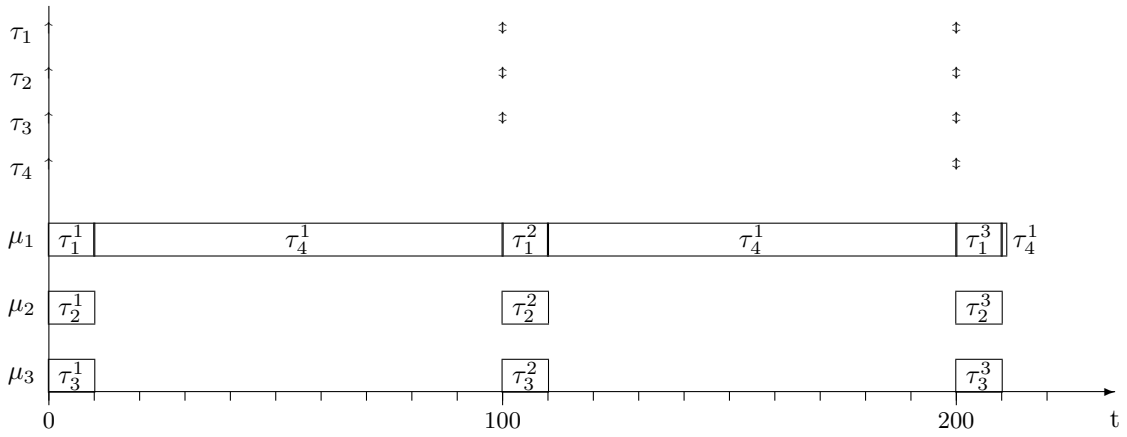
Conclusion: By combining the results from the analysis of processors 1 and 2, we see that all tasks will meet their deadlines if $25 \leq X \leq 30$

PROBLEM 7

a) Since rate-monotonic (RM) scheduling is used, the task priorities are as follows:

$prio(\tau_1) = \text{H}$, $prio(\tau_2) = \text{H}$, $prio(\tau_3) = \text{H}$, $prio(\tau_4) = \text{L}$.

We generate a multiprocessor schedule with tasks τ_1, τ_2 and τ_3 (having the highest priorities) running on one processor each. Task τ_4 is scheduled in the remaining time slots according to the following diagram (covering the first execution of τ_4):



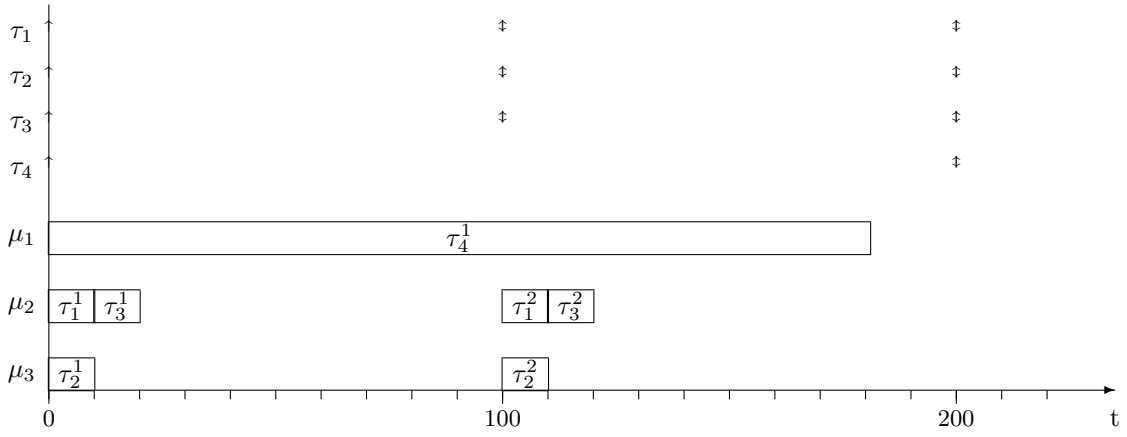
We observe that the first instance of task τ_4 completes its execution at $t = 211$ on processor μ_1, thereby missing its deadline at $t = T_4 = 200$. This happens despite there being significant processor capacity available on processors μ_2 and μ_3. A clear case of Dhall's effect!

b) If task priorities are given according to the rate-monotonic utilization-separation (RM-US) approach it may be possible to circumvent Dhall's effect, since that approach gives highest priority to "heavy" tasks in the task set. In order to make sure that task deadlines are met using the RM-US approach we need to verify that the total task utilization does not exceed the guarantee bound for RM-US.

The total utilization of the task set is $U_{\text{Total}} = U_1 + U_2 + U_3 + U_4 = 0.3 + 0.905 = 1.205$

The guarantee bound for RM-US is $U_{\text{RM-US}} = \frac{m^2}{3m-2}$. Since $m = 3$, $U_{\text{RM-US}} = 9/7 \approx 1.285$.

Consequently, by using the RM-US approach, all task deadlines for the task set in sub-problem a) will be met since $1.205 < 1.285$. The successful schedule in the hyper period $[0, 200]$ can be seen in the timing diagram below.



Real-Time Systems

Solutions to final exam March 11, 2024 (version 20240311)

Basic part

PROBLEM 1

a) The total utilization of the task set is $U = 1/4 + 3/15 + 6/18 \approx 0.783$.

Liu and Layland's utilization bound for 3 tasks is $\mathrm{U}_{\mathrm{RM}(3)} = 3 \cdot (2^{1/3} - 1) \approx 0.780$

The feasibility test fails, as $U > \mathrm{U}_{\mathrm{RM}(3)}$. Because the feasibility test is only sufficient, the schedulability of the task set cannot be determined.

---

b) Assuming the DM priority-assignment policy task τ_3 will have the lowest priority, since $D_3 = 14$ is larger than both $D_1 = 4$ and $D_2 = 10$. Task τ_1 will have highest priority, since $D_1 < D_2$

$R_1 = C_1 = 1 \leq D_1 = 4$

$R_2 = C_2 + \lceil \frac{R_2}{T_1} \rceil \cdot C_1$ [Assume that $R_2^0 = C_2 = 3$]

$R_2^1 = 3 + \lceil \frac{3}{4} \rceil \cdot 1 = 3 + 1 \cdot 1 = 4$

$R_2^2 = 3 + \lceil \frac{4}{4} \rceil \cdot 1 = 3 + 1 \cdot 1 = 4 \leq D_2 = 10$

$R_3 = C_3 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2$ [Assume that $R_3^0 = C_3 = 6$]

$R_3^1 = 6 + \lceil \frac{6}{4} \rceil \cdot 1 + \lceil \frac{6}{15} \rceil \cdot 3 = 6 + 2 \cdot 1 + 1 \cdot 3 = 11$

$R_3^1 = 6 + \lceil \frac{11}{4} \rceil \cdot 1 + \lceil \frac{11}{15} \rceil \cdot 3 = 6 + 3 \cdot 1 + 1 \cdot 3 = 12$

$R_3^1 = 6 + \lceil \frac{12}{4} \rceil \cdot 1 + \lceil \frac{12}{15} \rceil \cdot 3 = 6 + 3 \cdot 1 + 1 \cdot 3 = 12 \leq D_3 = 14$

The feasibility test for each task succeeds, as the calculated response time for each task never exceeds its deadline. Consequently, the task set is schedulable.

PROBLEM 1 (cont'd)

c) The total utilization of the task set is: $U = 1/4 + 3/15 + 6/18 \approx 0.783$

Since $U < 1$ the largest interval to examine is: $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,)$

For the given task set we have:

	U_i	$T_i - D_i$	$(T_i - D_i)\,U_i$
τ_1	1/4	0	0
τ_2	3/15	5	1
τ_3	6/18	4	1.333

$$L^* = \frac{\sum (T_i - D_i)\,U_i}{1 - U} = \frac{0 + 1 + 1.333}{1 - 0.783} = \frac{2.333}{0.217} \approx 10.75 \leq 11$$

The upper bound proposed by Baruah, Rosier and Howell:

$$L_{\text{BRH}} = \max(D_1, D_2, D_3, L^*) = \max(4, 10, 14, 11) = 14$$

The upper bound based on the hyper period:

$$L_{\text{LCM}} = \text{LCM}\{4, 15, 18\} = 180$$

Consequently, $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,) = \min(14, 180) = 14$

d) Derive the set K of control points in the interval $[\,0, 14\,]$

$K_1 = \{4, 8, 12\}$, $K_2 = \{10\}$ and $K_3 = \{14\}$

Thus, $K = K_1 \cup K_2 \cup K_3 = \{4, 8, 10, 12, 14\}$

e) Perform processor-demand analysis for all tasks and all control points in K:

$N_i^L \cdot C_i = (\lfloor \frac{(L - D_i)}{T_i} \rfloor + 1) \cdot C_i$

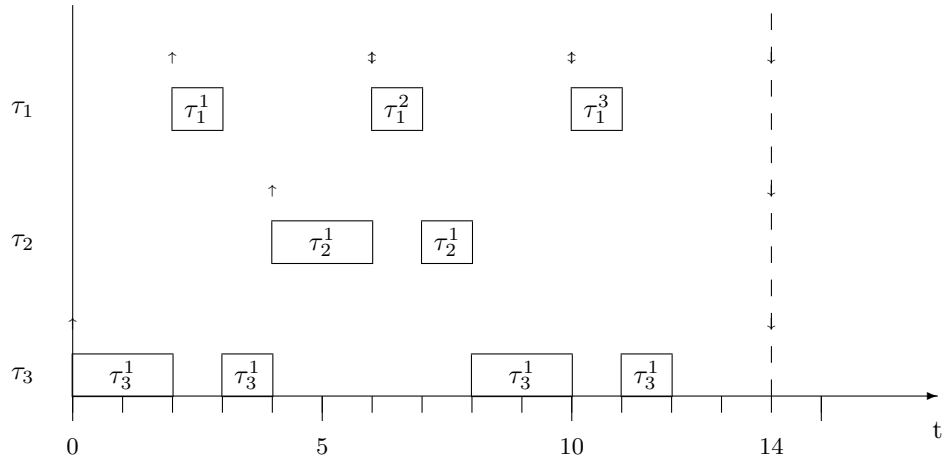
L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
4	$(\lfloor \frac{(4-4)}{4} \rfloor + 1) \cdot 1 = 1$	$(\lfloor \frac{(4-10)}{15} \rfloor + 1) \cdot 3 = 0$	$(\lfloor \frac{(4-14)}{18} \rfloor + 1) \cdot 6 = 0$	1	OK
8	$(\lfloor \frac{(8-4)}{4} \rfloor + 1) \cdot 1 = 2$	$(\lfloor \frac{(8-10)}{15} \rfloor + 1) \cdot 3 = 0$	$(\lfloor \frac{(8-14)}{18} \rfloor + 1) \cdot 6 = 0$	2	OK
10	$(\lfloor \frac{(10-4)}{4} \rfloor + 1) \cdot 1 = 2$	$(\lfloor \frac{(10-10)}{15} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(10-14)}{18} \rfloor + 1) \cdot 6 = 0$	5	OK
12	$(\lfloor \frac{(12-4)}{4} \rfloor + 1) \cdot 1 = 3$	$(\lfloor \frac{(12-10)}{15} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(12-14)}{18} \rfloor + 1) \cdot 6 = 0$	6	OK
14	$(\lfloor \frac{(14-4)}{4} \rfloor + 1) \cdot 1 = 3$	$(\lfloor \frac{(14-10)}{15} \rfloor + 1) \cdot 3 = 3$	$(\lfloor \frac{(14-14)}{18} \rfloor + 1) \cdot 6 = 6$	12	OK

The feasibility test for each control point succeeds, as the calculated processor demand for each interval never exceeds the length of the interval. Consequently, the task set is schedulable.

PROBLEM 2

The hyper period for the task set is $L_{\text{LCM}} = 180$. However, we will only consider the interval $[\,0, 14\,]$.

a) A simulation of the task set using DM scheduling, in the interval $[\,0, 14\,]$, gives the following timing diagram.



Note that the worst-case response times of the tasks correspond to those derived in the response-time analysis in sub-problem 1b). Thus, for this task set, the offsets of tasks τ_1 and τ_2 do not affect the worst-case response times of the other tasks.

b) Time table for the interval $[\,0, 14\,]$, corresponding to the timing diagram in sub-problem a):

$(\tau_3^1, 0, 2)\,(\tau_1^1, 2, 3)\,(\tau_3^1, 3, 4)\,(\tau_2^1, 4, 6)\,(\tau_1^2, 6, 7)\,(\tau_2^1, 7, 8)\,(\tau_3^1, 8, 10)\,(\tau_1^3, 10, 11)\,(\tau_3^1, 11, 12)$

PROBLEM 3

a) The WCET of `ShiftXY` largely depends on the bit pattern of the value of parameter x.

$$
\begin{aligned}
&WCET(ShiftXY(x,y)) = \\
&\{declare, result\} + \{declare, xbit0\} + \{assign, result\} + (b+1)\cdot\{compare, x \neq 0\} + \\
&b\cdot[\,\{and, x\ \&\ 1\} + \{assign, xbit0\} + \{compare, xbit0 == 1\} + \\
&\quad \{rshift, x\} + \{assign, x\} \ + \{lshift, y\} + \{assign, y\}\,] + \\
&u\cdot[\,\{multiply, result * y\} + \{assign, result\}\,] + \\
&\{return, result\} \\
&= 1 + 1 + 1 + (b+1)\cdot 2 + b\cdot[\,1+1+2+2+1+2+1\,] + u\cdot[\,5+1\,] + 2 \\
&= 12\cdot b + 6\cdot u + 7
\end{aligned}
$$

The maximum number of `while` loops, b, and updates of variable `result`, u, are determined by the binary bit pattern of the value of parameter x. To that end it is known that `ShiftXY` is called with values of x that can be at most $14_{10} = 1110_2$.

The `while` loop runs as long as the current value of x is not zero, and shifts x one step to the right (divides x by 2) at the end of each iteration. The number of iterations is maximized when bit 3 of the initial value of x is 1. For example, if the initial value of $x = 1110_2$ then x will be shifted as follows: $1110_2, 0111_2, 0011_2, 0001_2$, and 0000_2. Hence, the loop will run at most $b = 4$ iterations.

In each iteration of the `while` loop variable `result` is updated if variable `xbit0`, which represents bit 0 of the current value of x, is 1. The number of updates is therefore equal to the number of bits that are 1 in the initial value of x. As shown in the example above (with $x = 1110_2$) variable `xbit0` will be 1 in three of the four iterations of the `while` loop. Hence, variable `result` will be updated $u = 3$ times.

Consequently, the WCET of `ShiftXY` is: $12 \cdot 4 + 6 \cdot 3 + 7 = 48 + 18 + 7 = 73\ \mu$s

b) From sub-problem a) we know that the WCET of `ShiftXY` is:

$$WCET(ShiftXY(x,y)) = 12\cdot b + 6\cdot u + 7$$

Here, b is the total number of `while` loops, and u the total number of updates of variable `result`.

Assuming that $b = 4$, (that is, bit 3 of the initial value of x is 1) we get the following inequality:

$$12\cdot 4 + 6\cdot u + 7 \leq 67 \implies u \leq 2$$

Recall that u, the number of updates, is equal to the number of bits that are 1 in the initial value of x. Then, only values with at most two bits set to 1 will yield a WCET ≤ 67. Since $b = 4$ is assumed we know that bit 3 must be set to 1. This leaves four possible values:

$12_{10}\,(1100_2), 10_{10}\,(1010_2), 9_{10}\,(1001_2)$ and $8_{10}\,(1000_2)$.

The value $12_{10}\,(1100_2)$ cannot be the largest value N in the value range $[1, \mathrm{N}]$ of parameter x, since there would then exist a smaller value in the value range yielding a WCET > 67. That value, $11_{10}\,(1011_2)$, has three bits set to 1 and would yield a WCET of 73.

The largest value range $[1, \mathrm{N}]$ of parameter x is consequently achieved when $\mathrm{N} = \mathbf{10}$.

PROBLEM 3 (cont'd)

c) A priority ceiling is associated with a shared resource, and represents the priority of the task with highest static priority that may use that resource.

During run-time the access-control protocol makes sure that a task X can enter the critical region of a shared resource only if the priority of task X is higher than all priority ceilings of the resources currently locked by tasks other than task X.

d) A critical instant refers to a task arrival scenario in which the response time of a given task is maximized. This definition is valid for both single-processor and multiprocessor scheduling.

For preemptive single-processor scheduling using static priorities it can be shown that the critical instant occurs when the task arrives at the same time as all tasks with higher priority. For preemptive global multiprocessor scheduling using static priorities, however, this is not necessarily the case as the response time of a task may actually be maximized at some other (asynchronous) task arrival pattern.

---

e) To read the current value of the system clock the programmer has two options:

- The `CURRENT_OFFSET` operation returns the time duration from the current baseline to the current time.
- The `T_SAMPLE` operation returns the time duration from a bookmarked baseline to the current baseline. A baseline of an earlier event is bookmarked by means of the `T_RESET` operation. Both operations should refer to a common object of type `Timer`.

The resolution of the TinyTimber system clock in the laboratory system is 10 μs.

f) To create a concurrent execution the programmer should use the `ASYNC` operation or one of its associated timing-aware versions (`AFTER`, `SEND`, `BEFORE`), and provide as a parameter an object method identifying the program code to execute concurrently.

Note that the `SYNC` operation cannot be used to create a concurrent execution, as it causes the calling task to wait for the completion of the `SYNC` operation.

Advanced part

PROBLEM 4

a) The four steps of proving NP-completeness for a decision problem Π:

1. Show that Π is in NP
2. Select a known NP-complete problem Π'
3. Construct a transformation α from Π' to Π
4. Prove that α is a (polynomial) transformation

b) Dhall's effect in the context of global multiprocessor scheduling refers to the following phenomenon. If a traditional single-processor priority-assignment approach, such as RM, is used with global scheduling it is possible to construct a task set, that is not schedulable although it has a very low utilization, regardless of how many processors are used.

The underlying reason for this phenomenon is that, with the RM approach, it is possible to construct a task set where the lowest priority is given to a task that have a long period while at the same time have a very high computational requirement in relation to its period (the "heavy" task). If there are as many highest-priority tasks, with short periods and very low computational requirement in relation to their periods (the "light" tasks), as there are processors the "heavy" task will not be able to fully utilize the available processor capacity in the best way, and will therefore miss its deadline.

To avoid Dhall's effect the "heavy" tasks in the task set should be given higher priority than the "light" tasks, for example by using the RM-US priority-assignment policy.

c) No, we cannot provide a guarantee. This is because the schedulability of task τ_i based on RTA only considers the amount of interference by the higher-priority task τ_j regardless of whether τ_j meets its deadline or not. The following task set demonstrates such a situation (using DM priorities).

	C_i	D_i	T_i
τ_1	2	3	5
τ_2	4	7	10
τ_3	2	12	20

For this task set the response time of the lowest-priority task τ_3 is $R_3 = 10$. However, task τ_2 (which has higher priority than τ_3) misses its deadline ($R_2 = 8$). Therefore, the lower-priority task τ_3 meets its deadline while the higher-priority task τ_2 misses its deadline.

d) Time complexity of deciding schedulability for each special case of preemptive scheduling of periodic tasks with static priorities:

- Global multiprocessor scheduling of synchronous, implicit-deadline tasks (**S1**): a decision problem that is NP-complete in the strong sense (**C2**).
- Single-processor scheduling of synchronous, arbitrary-deadline tasks (**S2**): a decision problem with exponential time complexity (**C1**).
- Single-processor scheduling of synchronous, constrained-deadline tasks (**S3**): an NP-complete problem with pseudo-polynomial time complexity (**C4**). [Using response-time analysis]

PROBLEM 5

Since RM scheduling is used, the task priorities are determined by the task periods. To that end, using the original task periods, the static priorities are as follows:

$prio(\tau_1) = \text{H}$, $prio(\tau_2) = \text{M}$, $prio(\tau_3) = \text{L}$.

a) Our first candidate method for schedulability analysis is Liu and Layland's utilization-based test. For three tasks, the guarantee bound for RM is $\text{U}_{\text{RM(3)}} = 3(2^{1/3} - 1) \approx 0.780$. Unfortunately, the accumulated task utilization, $U = 2/5 + 4/13 + 6/29 \approx 0.915$, exceeds the guarantee bound, and the test does not provide any useful information.

We therefore calculate the response time of each task and compare it against the corresponding task deadline (= period):

$R_1 = C_1 = 2 \leq T_1 = 5$.

$R_2 = C_2 + \lceil \frac{R_2}{T_1} \rceil \cdot C_1$. Assume that $R_2^0 = C_2 = 4$:

$R_2^1 = 4 + \lceil \frac{4}{5} \rceil \cdot 2 = 4 + 1 \cdot 2 = 6$

$R_2^2 = 4 + \lceil \frac{6}{5} \rceil \cdot 2 = 4 + 2 \cdot 2 = 8$

$R_2^3 = 4 + \lceil \frac{8}{5} \rceil \cdot 2 = 4 + 2 \cdot 2 = 8 \leq T_2 = 13$

$R_3 = C_3 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1$. Assume that $R_3^0 = C_3 = 6$:

$R_3^1 = 6 + \lceil \frac{6}{13} \rceil \cdot 4 + \lceil \frac{6}{5} \rceil \cdot 2 = 6 + 1 \cdot 4 + 2 \cdot 2 = 6 + 4 + 4 = 14$

$R_3^2 = 6 + \lceil \frac{14}{13} \rceil \cdot 4 + \lceil \frac{14}{5} \rceil \cdot 2 = 6 + 2 \cdot 4 + 3 \cdot 2 = 6 + 8 + 6 = 20$

$R_3^3 = 6 + \lceil \frac{20}{13} \rceil \cdot 4 + \lceil \frac{20}{5} \rceil \cdot 2 = 6 + 2 \cdot 4 + 4 \cdot 2 = 6 + 8 + 8 = 22$

$R_3^4 = 6 + \lceil \frac{22}{13} \rceil \cdot 4 + \lceil \frac{22}{5} \rceil \cdot 2 = 6 + 2 \cdot 4 + 5 \cdot 2 = 24$

$R_3^5 = 6 + \lceil \frac{24}{13} \rceil \cdot 4 + \lceil \frac{24}{5} \rceil \cdot 2 = 6 + 2 \cdot 4 + 5 \cdot 2 = 24 \leq T_3 = 29$

Conclusion: all tasks meet their deadlines!

b) An obvious version of the task set that has more appropriate periods (within the given limits) is where $T_2 = 15$ and $T_3 = 30$. Since the original task set is schedulable, and neither the new T_2 nor the new T_3 is shorter than the original period, the new task set must also be schedulable. The length of this (cyclic) schedule is 30 time units. If we generate the time table by simulating an RM scheduler we get the following start and stop times for the tasks:

τ_1: 6 instances: (0,2), (5,7), (10,12), (15,17), (20,22) and (25,27)

τ_2: 2 instances: (2,5)(7,8) and (17,20)(22,23)

τ_3: 1 instance: (8,10)(12,15)(23,24)

There is also a version of the task set with $T_2 = 10$ and $T_3 = 30$ that, despite a total task utilization of 100%, is RM schedulable. The length of this (cyclic) schedule is also 30 time units, but has one more instance of τ_2 and thus less compact.

PROBLEM 6

We apply processor-demand analysis to determine the minimum value of the deadline D_2.

For our analysis we will assume the largest interval $L_{\max} = L_{\text{LCM}}$.

$L_{\text{LCM}} = \text{LCM}\{T_1, T_2, T_3\} = \text{LCM}\{10, 20, 40\} = 40.$

Since $C_2 = 2$, we must have $D_2 \geq 2$. Assuming $2 \leq D_2 \leq 5$ it is easy to see that the analysis in control point $L = 5$ would fail. The first instances of tasks τ_1 and τ_2 would need to complete a total of $(2 + 4) = 6$ units of execution since their deadlines fall within the interval $[0, 5]$. However, completing 6 units of execution within interval $[0, 5]$ is not possible. Therefore, we must have $D_2 \geq 6$.

We tentatively assume that $D_2 = 6$, and continue by checking schedulability under that assumption.

With $D_2 = 6$, the set K of control points in the interval $[\,0, L_{\max}\,]$ is as follows:

$K_1 = \{5, 15, 25, 35\}$, $K_2 = \{6, 26\}$ and $K_3 = \{25\}$

Thus, $K = K_1 \cup K_2 \cup K_3 = \{5, 6, 15, 25, 26, 35\}$

L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
5	$(\lfloor \frac{(5-5)}{10} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(5-6)}{20} \rfloor + 1) \cdot 2 = 0$	$(\lfloor \frac{(5-25)}{40} \rfloor + 1) \cdot 4 = 0$	4	OK
6	$(\lfloor \frac{(6-5)}{10} \rfloor + 1) \cdot 4 = 4$	$(\lfloor \frac{(6-6)}{20} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(6-25)}{40} \rfloor + 1) \cdot 4 = 0$	6	OK
15	$(\lfloor \frac{(15-5)}{10} \rfloor + 1) \cdot 4 = 8$	$(\lfloor \frac{(15-6)}{20} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(15-25)}{40} \rfloor + 1) \cdot 4 = 0$	10	OK
25	$(\lfloor \frac{(25-5)}{10} \rfloor + 1) \cdot 4 = 12$	$(\lfloor \frac{(25-6)}{20} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(25-25)}{40} \rfloor + 1) \cdot 4 = 4$	18	OK
26	$(\lfloor \frac{(26-5)}{10} \rfloor + 1) \cdot 4 = 12$	$(\lfloor \frac{(26-6)}{20} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(26-25)}{40} \rfloor + 1) \cdot 4 = 4$	20	OK
35	$(\lfloor \frac{(35-5)}{10} \rfloor + 1) \cdot 4 = 16$	$(\lfloor \frac{(35-6)}{20} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(35-25)}{40} \rfloor + 1) \cdot 4 = 4$	24	OK

The processor demand in each strategic time interval never exceeds the length of the interval, so all tasks meet their deadlines. Consequently, the minimum value of D_2 is 6.

PROBLEM 7

a) We begin by calculating the utilization U_i for each task:

	C_i	T_i	U_i
τ_1	10	100	0.1
τ_2	10	100	0.1
τ_3	10	100	0.1
τ_4	141	200	0.705
τ_5	141	200	0.705
τ_6	141	200	0.705

Then, number the three processors μ_1, μ_2 and μ_3.

According to the RMFF partitioning algorithm the tasks should be assigned to the processors in the following (RM) order: $\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6$.

Tasks τ_1, τ_2, and τ_3 can all be assigned to processor μ_1, since

$U_1 + U_2 + U_3 = 0.1 + 0.1 + 0.1 = 0.3 < \mathrm{U}_{\mathrm{RM(3)}} = 3 \cdot (2^{1/3} - 1) \approx 0.780$

Task τ_4 cannot assigned to μ_1, since

$U_1 + U_2 + U_3 + U_4 = 0.3 + 0.705 > 1.0$

Task τ_4 can be assigned to μ_2, since there are no task assigned to that processor.

Task τ_5 cannot assigned to μ_1, since

$U_1 + U_2 + U_3 + U_5 = 0.3 + 0.705 > 1.0$

Task τ_5 cannot assigned to μ_2, since

$U_4 + U_5 = 0.705 + 0.705 > 1.0$

Task τ_5 can be assigned to μ_3, since there are no task assigned to that processor.

Task τ_6 cannot assigned to μ_1, since

$U_1 + U_2 + U_3 + U_6 = 0.3 + 0.705 > 1.0$

Task τ_6 cannot assigned to μ_2, since

$U_4 + U_6 = 0.705 + 0.705 > 1.0$

Task τ_6 cannot assigned to μ_3, since

$U_5 + U_6 = 0.705 + 0.705 > 1.0$

Task τ_6 can, consequently, not be assigned to any processor. This means that the given task set cannot be scheduled using the RMFF algorithm.

b) The partitioned scheduling scenario in sub-problem a) has many similarities with Dhall's effect for global multiprocessor scheduling. For the given task set the RMFF algorithm will assign the "light" tasks first and the "heavy" tasks last, and because of the special task properties one of the "heavy" tasks cannot be assigned to any processor.

Similar to how Dhall's effect is circumvented for global multiprocessor scheduling (using RM-US) there is a simple remedy to the problem in sub-problem a). If the "heavy" tasks are assigned to the processors before the "light" tasks, it is possible to find a task-to-processor assignment for which RM-priority scheduling will meet all task deadlines. In such an assignment each processor will contain one "heavy" and one "light" task. The task utilization on each processor is then $0.1 + 0.705 = 0.805 < \mathrm{U}_{\mathrm{RM(2)}} = 2 \cdot (2^{1/2} - 1) \approx 0.828$, which means that all task deadlines are met on each processor.

Consequently, there exists a task-to-processor assignment for the task set in sub-problem a), such that all task deadlines will be met when task priorities are given according to the RM policy.

Real-Time Systems

Solutions to re-exam August 20, 2024 (version 20240820)

Basic part

PROBLEM 1

a) The total utilization of the task set is $U = 6/15 + 2/25 + 20/75 \approx 0.747$.

Liu and Layland's utilization bound for 3 tasks is $\mathrm{U}_{\mathrm{RM}(3)} = 3 \cdot (2^{1/3} - 1) \approx 0.780$

The feasibility test succeeds, as $U < \mathrm{U}_{\mathrm{RM}(3)}$. Consequently, the task set is schedulable.

---

b) Assuming the DM priority-assignment policy task τ_3 will have the lowest priority, since $D_3 = 38$ is larger than both $D_1 = 10$ and $D_2 = 8$. Task τ_2 will have highest priority, since $D_2 < D_1$

$R_2 = C_2 = 2 \leq D_2 = 8$

$R_1 = C_1 + \lceil \frac{R_1}{T_2} \rceil \cdot C_2 \quad$ [Assume that $R_1^0 = C_1 = 6$]

$R_1^1 = 6 + \lceil \frac{6}{25} \rceil \cdot 2 = 6 + 1 \cdot 2 = 8$

$R_1^2 = 6 + \lceil \frac{8}{25} \rceil \cdot 2 = 6 + 1 \cdot 2 = 8 \leq D_1 = 10$

$R_3 = C_3 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2 \quad$ [Assume that $R_3^0 = C_3 = 20$]

$R_3^1 = 20 + \lceil \frac{20}{15} \rceil \cdot 6 + \lceil \frac{20}{25} \rceil \cdot 2 = 20 + 2 \cdot 6 + 1 \cdot 2 = 34$

$R_3^1 = 20 + \lceil \frac{34}{15} \rceil \cdot 6 + \lceil \frac{34}{25} \rceil \cdot 2 = 20 + 3 \cdot 6 + 2 \cdot 2 = 42$

$R_3^1 = 20 + \lceil \frac{42}{15} \rceil \cdot 6 + \lceil \frac{42}{25} \rceil \cdot 2 = 20 + 3 \cdot 6 + 2 \cdot 2 = 42 > D_3 = 38$

The feasibility test for task τ_3 fails as the calculated response time of the task exceeds its deadline. Because the feasibility test is exact, the task set is <u>not</u> schedulable.

PROBLEM 1 (cont'd)

c) The total utilization of the task set is: $U = 6/15 + 2/25 + 20/75 \approx 0.75$

Since $U < 1$ the largest interval to examine is: $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,)$

For the given task set we have:

	U_i	$T_i - D_i$	$(T_i - D_i)\,U_i$
τ_1	6/15	5	2
τ_2	2/25	17	1.36
τ_3	20/75	37	9.87

$$L^* = \frac{\sum (T_i - D_i)\,U_i}{1 - U} = \frac{2 + 1.36 + 9.87}{1 - 0.75} = \frac{13.23}{0.25} \approx 52.92 \leq 53$$

The upper bound proposed by Baruah, Rosier and Howell:

$L_{\text{BRH}} = \max(D_1, D_2, D_3, L^*) = \max(10, 8, 38, 53) = 53$

The upper bound based on the hyper period:

$L_{\text{LCM}} = \text{LCM}\{15, 25, 75\} = 75$

Consequently, $L_{\max} = \min(\, L_{\text{BRH}}, L_{\text{LCM}}\,) = \min(53, 75) = 53$

d) Derive the set K of control points in the interval $[\,0, 53\,]$

$K_1 = \{10, 25, 40\}$, $K_2 = \{8, 33\}$ and $K_3 = \{38\}$

Thus, $K = K_1 \cup K_2 \cup K_3 = \{8, 10, 25, 33, 38, 40\}$

e) Perform processor-demand analysis for all tasks and all control points in K:

$N_i^L \cdot C_i = (\lfloor \frac{(L - D_i)}{T_i} \rfloor + 1) \cdot C_i$

L	$N_1^L \cdot C_1$	$N_2^L \cdot C_2$	$N_3^L \cdot C_3$	$C_P(0, L)$	$C_P(0, L) \leq L$
8	$(\lfloor \frac{(8-10)}{15} \rfloor + 1) \cdot 6 = 0$	$(\lfloor \frac{(8-8)}{25} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(8-38)}{75} \rfloor + 1) \cdot 20 = 0$	2	OK
10	$(\lfloor \frac{(10-10)}{15} \rfloor + 1) \cdot 6 = 6$	$(\lfloor \frac{(10-8)}{25} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(10-38)}{75} \rfloor + 1) \cdot 20 = 0$	8	OK
25	$(\lfloor \frac{(25-10)}{15} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(25-8)}{25} \rfloor + 1) \cdot 2 = 2$	$(\lfloor \frac{(25-38)}{75} \rfloor + 1) \cdot 20 = 0$	14	OK
33	$(\lfloor \frac{(33-10)}{15} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(33-8)}{25} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(33-38)}{75} \rfloor + 1) \cdot 20 = 0$	16	OK
38	$(\lfloor \frac{(38-10)}{15} \rfloor + 1) \cdot 6 = 12$	$(\lfloor \frac{(38-8)}{25} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(38-38)}{75} \rfloor + 1) \cdot 20 = 20$	36	OK
40	$(\lfloor \frac{(40-10)}{15} \rfloor + 1) \cdot 6 = 18$	$(\lfloor \frac{(40-8)}{25} \rfloor + 1) \cdot 2 = 4$	$(\lfloor \frac{(40-38)}{75} \rfloor + 1) \cdot 20 = 20$	42	**FAIL**

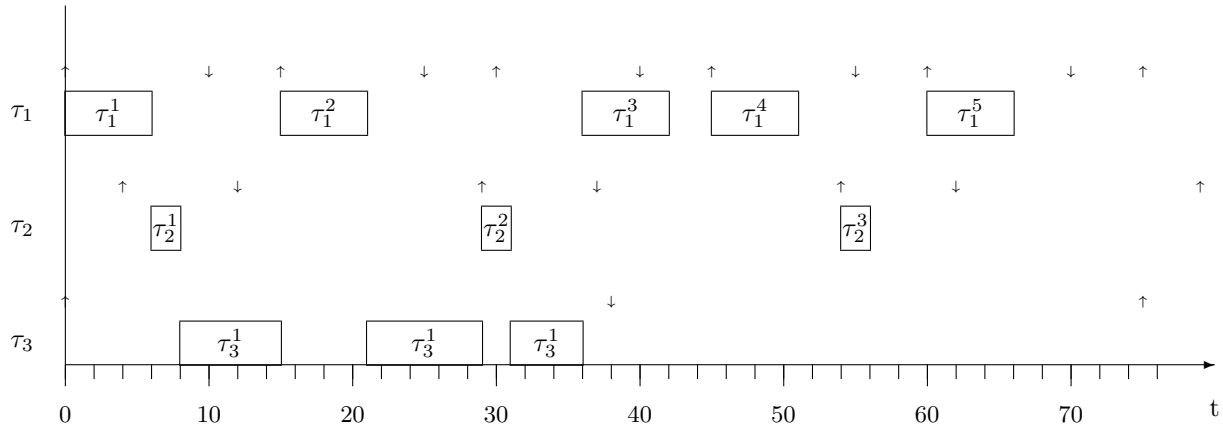
The feasibility test for $L = 40$ fails as the calculated processor demand in the interval exceeds the length of the interval. Because the feasibility test is exact, the task set is <u>not</u> schedulable.

The implication of this is that the third instance of task τ_1 will not meet its deadline at $t = 40$.

PROBLEM 2

The hyper period for the task set is $L_{\text{LCM}} = 75$.

a) A simulation of the task set using EDF scheduling, in the interval $[\,0, 75\,]$, gives the following timing diagram.



The third instance of task τ_1 fails to meet its deadline at $t = 40$, and completes its execution at $t = 42$.

Note that the missed deadline at $t = 40$ corresponds to the failure in the processor-demand analysis for control point $L = 40$ in sub-problem 1e).

b) Time table corresponding to the timing diagram in sub-problem a):

$(\tau_1^1, 0, 6)$ $(\tau_2^1, 6, 8)$ $(\tau_3^1, 8, 15)$ $(\tau_1^2, 15, 21)$ $(\tau_3^1, 21, 29)$ $(\tau_2^2, 29, 31)$ $(\tau_3^1, 31, 36)$
$(\tau_1^3, 36, 42)$ $(\tau_1^4, 45, 51)$ $(\tau_2^3, 54, 56)$ $(\tau_1^5, 60, 66)$

PROBLEM 3

a) The WCET of `FuncZ(z)` is derived based on three cases of the value of parameter z.

Case 1: $z = 0$:

$$WCET(FuncZ(z = 0)) =$$
$$\{declare, p\} + \{compare, z == 0\} + \{assign, p\} + \{return, p\}$$
$$= 1 + 2 + 1 + 2 = \mathbf{6}$$

Case 2: $z = 1$:

$$WCET(FuncZ(z = 1)) =$$
$$\{declare, p\} + \{compare, z == 0\} + \{compare, z == 1\} +$$
$$\{assign, p\} + \{return, p\}$$
$$= 1 + 2 + 2 + 1 + 2 = \mathbf{8}$$

Case 3: $z > 1$:

$$WCET(FuncZ(z > 1)) =$$
$$\{declare, p\} + \{compare, z == 0\} + \{compare, z == 1\} +$$
$$\{sub, z - 1\} + \{call, FuncZ(z - 1)\} + WCET(FuncZ(z - 1)) +$$
$$\{add, z + FuncZ(z - 1)\} + \{assign, p\} + \{multiply, p * p\} + \{assign, p\} + \{return, p\}$$
$$= 1 + 2 + 2 + 3 + 2 + 3 + 1 + 5 + 1 + 2 + WCET(FuncZ(z - 1))$$
$$= 22 + WCET(FuncZ(z - 1)) = \{\text{recursive expansion}\}$$
$$= (z - 1) \cdot 22 + WCET(FuncZ(1)) = (z - 1) \cdot 22 + 8$$
$$= \mathbf{22 \cdot z - 14}$$

b) WCET estimates must be tight to avoid unnecessary pessimism in the schedulability analysis, which could cause feasibility tests to be too inaccurate to be useful.

PROBLEM 3 (cont'd)

c) Dhall's effect in the context of global multiprocessor scheduling refers to the following phenomenon. If a traditional single-processor priority-assignment approach, such as RM, is used with global scheduling it is possible to construct a task set, that is not schedulable although it has a very low utilization, regardless of how many processors are used.

The underlying reason for this phenomenon is that, with the RM approach, it is possible to construct a task set where the lowest priority is given to a task that have a long period while at the same time have a very high computational requirement in relation to its period (the "heavy" task). If there are as many highest-priority tasks, with short periods and very low computational requirement in relation to their periods (the "light" tasks), as there are processors the "heavy" task will not be able to fully utilize the available processor capacity in the best way, and will therefore miss its deadline.

d) If we know that the task set is schedulable then a necessary test must have resulted in the outcome 'True'. This is because, for necessary tests, the outcome 'False' always means that the task set is not schedulable.

---

e) Systematic skew is a phenomenon relating to periodic task executions, where the actual period becomes longer than the intended one, causing each subsequent task arrival to drift further and further away from its intended arrival time.

Systematic skew typically occurs in systems where periodic task executions have been implemented using the *relative* delay statement. With such a delay statement the execution time of the calling code before the delay statement will be added to the requested delay.

By using an *absolute* delay statement, specifying an actual time instant on the timeline for the next execution (instead of a delay relative to the current point of execution), the execution time of the calling code does not influence the periodic behaviour. This can be achieved by means of, for example, the `AFTER()` operation in TinyTimber, where a method's baseline is used to calculate a time instant on the timeline for the next execution of the method.

f) To read the current value of the system clock the programmer has two options:

- The `CURRENT_OFFSET` operation returns the time duration from the current baseline to the current time.
- The `T_SAMPLE` operation returns the time duration from a bookmarked baseline to the current baseline. A baseline of an earlier event is bookmarked by means of the `T_RESET` operation. Both operations should refer to a common object of type `Timer`.

The resolution of the TinyTimber system clock in the laboratory system is 10 μs.

Advanced part

PROBLEM 4

a) While the RM and EDF priority-assignment policies both make similar assumptions regarding the task model, they differ in the way they affect how the tasks are sorted in the ready queue in the run-time system.

It can be shown that there exist task sets that can be scheduled with dynamic priorities but cannot be scheduled with static priorities. This is because static priorities are a restricted subset of dynamic priorities. Thus, EDF dominates RM in terms of schedulability.

b) Any positive value of O_i will work, as long as the task set is synchronous. To see this, assume that all O_i would have the same positive (non-zero) value. Since the tasks are synchronous we are analysing the critical-instant scenario, even if the offsets are positive. Without loss of generality we can therefore analyse a task set with the same timing parameters, except that all offsets are $O_i = 0$. The time complexity of the response-time behaviour would be exactly the same since we are still analysing the critical-instant case. Therefore, since RTA has been shown to have pseudo-polynomial time complexity for the case where $O_i = 0$ (see Lecture #15), it will also have pseudo-polynomial time complexity for a task set with an arbitrary positive value of O_i (as long as the task set remains synchronous).

PROBLEM 5

We start by observing that task τ_1 has a first arrival time that differs from that of the other tasks. This means that the use of a utilization-based or response-time-based schedulability test may become overly pessimistic IF there exists no point in time in the schedule where all tasks arrive at the same time. This, in turn, could mean that, should the test fail, the task set could potentially still be schedulable.

Our first candidate method for schedulability analysis is Liu and Layland's classic utilization-based test. For three tasks, the schedulability bound is $\mathrm{U}_{\mathrm{RM(3)}} = 3\,(2^{1/3} - 1) \approx 0.780$. Unfortunately, the accumulated task utilization, $U = 1/3 + 3/8 + 1/4 \approx 0.958$, significantly exceeds the guarantee bound, and the test (being only sufficient) does not provide any useful information.

Our second candidate method is response-time analysis. Since task periods are required by the analysis, we begin by deriving the period of each task: $T_i = C_i/U_i = C_i \cdot (U_i)^{-1}$

$T_1 = C_1 \cdot (U_1)^{-1} = 0.2C \cdot 3 = 0.6C$

$T_2 = C_2 \cdot (U_2)^{-1} = 0.3C \cdot 8/3 = 0.8C$

$T_3 = C_3 \cdot (U_3)^{-1} = 0.3C \cdot 4 = 1.2C$

Assuming RM scheduling, task τ_1 has highest priority (shortest period) and task τ_3 has lowest priority. We then calculate the response time of each task and compare it against the corresponding task deadline:

$R_1 = C_1 = 0.2C < D_1 = T_1 = 0.6C.$

$R_2 = C_2 + \lceil \frac{R_2}{T_1} \rceil \cdot C_1$. Assume that $R_2^0 = C_2 = 0.3C$:

$R_2^1 = 0.3C + \lceil \frac{0.3C}{0.6C} \rceil \cdot 0.2C = 0.3C + 1 \cdot 0.2C = 0.5C$

$R_2^2 = 0.3C + \lceil \frac{0.5C}{0.6C} \rceil \cdot 0.2T = 0.3C + 1 \cdot 0.2C = 0.5C < D_2 = T_2 = 0.8C$

$R_3 = C_3 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2$. Assume that $R_3^0 = C_3 = 0.3C$:

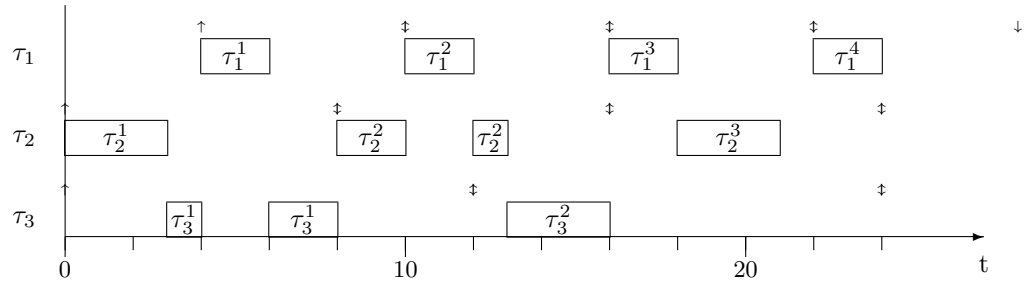
$R_3^1 = 0.3C + \lceil \frac{0.3C}{0.6C} \rceil \cdot 0.2C + \lceil \frac{0.3C}{0.8C} \rceil \cdot 0.3C = 0.3C + 1 \cdot 0.2C + 1 \cdot 0.3C = 0.8C$

$R_3^2 = 0.3C + \lceil \frac{0.8C}{0.6C} \rceil \cdot 0.2C + \lceil \frac{0.8C}{0.8C} \rceil \cdot 0.3C = 0.3C + 2 \cdot 0.2C + 1 \cdot 0.3C = 1.0C$

$R_3^3 = 0.3C + \lceil \frac{1.0C}{0.6C} \rceil \cdot 0.2C + \lceil \frac{1.0C}{0.8C} \rceil \cdot 0.3C = 0.3C + 2 \cdot 0.2C + 2 \cdot 0.3C = 1.3C > D_3 = T_3 = 1.2C$

The response-time analysis thus indicates that the worst-case response-time for τ_3 exceeds the task deadline. However, by observing the given periods and offsets (see time diagram below), we can see that there does NOT exist a point in time where all tasks arrive at the same time. This means that the worst-case response-time for τ_3 calculated by the response-time analysis will in fact never occur, and that it still is possible that τ_3 will meet its deadline. We will find out by using hyper-period analysis.

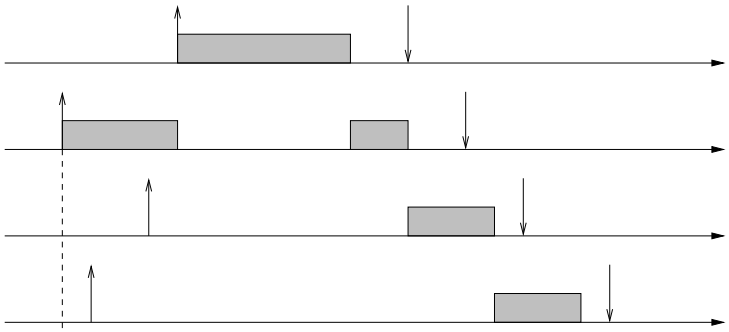
By simulating the RM schedule for one hyper period[1] we will see that <u>all tasks indeed meet their deadlines</u>. The hyper period for the given task set is LCM = 2.4C. Let C = 10 to get LCM = 24:



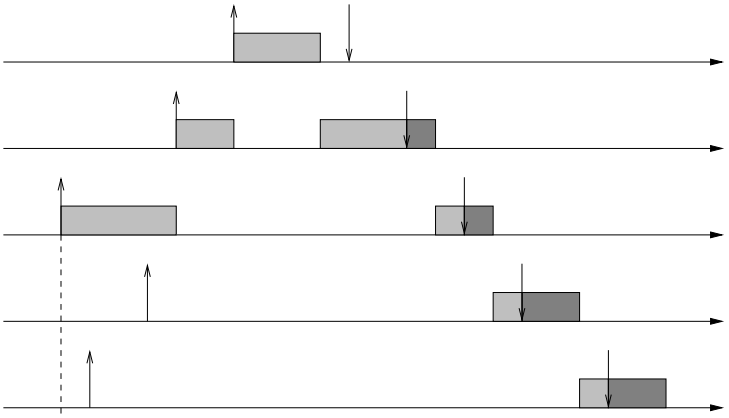
[1] Although we have an asynchronous task set, all task executions are completed within the first hyper period. This means that we only need to check schedulability within that interval.

PROBLEM 6

a) The timing diagram for the execution of the four original tasks, τ_1, τ_2, τ_3 och τ_4, looks as follows:



b) Task τ_0 can, for example, be chosen in the following way: At $t = t_a + 6$, task τ_0 with execution time $C_0 = 3$ and relative deadline $D_0 = 4$. The arrival of τ_0 then has the requested consequence that all of the tasks τ_1 – τ_4 miss their deadlines. See figure below.



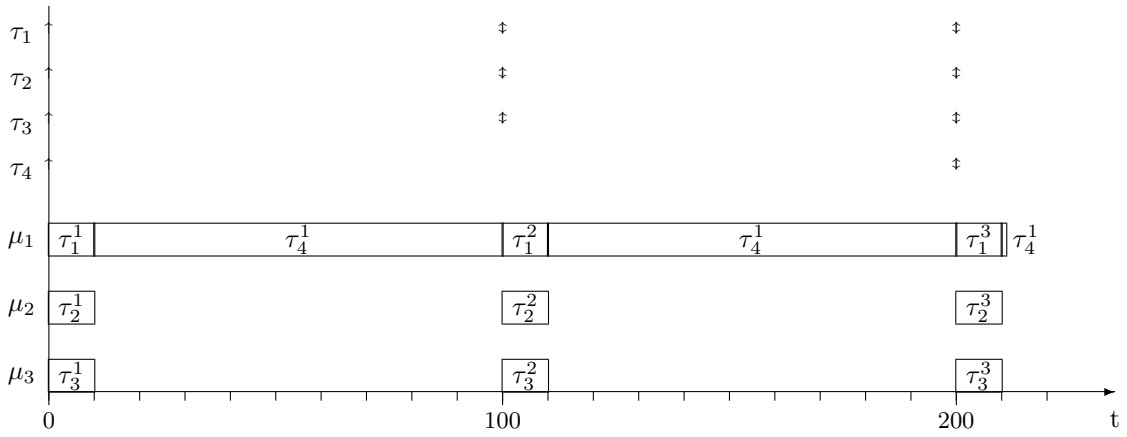
c) An example of a scheduling method that would work better that EDF is deadline-monotonic (DM) scheduling. With DM priorities, the tasks would be ordered as follows (highest priority first): $\tau_0, \tau_1, \tau_3, \tau_2$ and τ_4. With the arrival scenario assumed in sub-problem b), tasks τ_0 and τ_3 will meet their deadlines. τ_3 has a shorter relative deadline than τ_2, and τ_3 will therefore execute 1 time units of its execution already at time $t = t_a + 3$ and then resume its execution when τ_1 has completed its execution at time $t = t_a + 13$. Since the absolute deadline for τ_3 is at time $t = t_a + 16$ and there's only one time unit of execution left, τ_3 meets it deadline. However, all the other tasks miss their deadlines. DM is therefore (although not very much) better than EDF in this case.

PROBLEM 7

a) Since rate-monotonic (RM) scheduling is used, the task priorities are as follows:

$prio(\tau_1) = \text{H}$, $prio(\tau_2) = \text{H}$, $prio(\tau_3) = \text{H}$, $prio(\tau_4) = \text{L}$.

We generate a multiprocessor schedule with tasks τ_1, τ_2 and τ_3 (having the highest priorities) running on one processor each. Task τ_4 is scheduled in the remaining time slots according to the following diagram (covering the first execution of τ_4):



We observe that the first instance of task τ_4 completes its execution at $t = 211$ on processor μ_1, thereby missing its deadline at $t = T_4 = 200$. This happens despite there being significant processor capacity available on processors μ_2 and μ_3. A clear case of Dhall's effect!

b) If task priorities are given according to the rate-monotonic utilization-separation (RM-US) approach it may be possible to circumvent Dhall's effect, since that approach gives highest priority to "heavy" tasks in the task set. In order to make sure that task deadlines are met using the RM-US approach we need to verify that the total task utilization does not exceed the guarantee bound for RM-US.

The total utilization of the task set is $U_{\text{Total}} = U_1 + U_2 + U_3 + U_4 = 0.3 + 0.905 = 1.205$

The guarantee bound for RM-US is $U_{\text{RM-US}} = \frac{m^2}{3m-2}$. Since $m = 3$, $U_{\text{RM-US}} = 9/7 \approx 1.285$.

Consequently, by using the RM-US approach, all task deadlines for the task set in sub-problem a) will be met since $1.205 < 1.285$. The successful schedule in the hyper period $[0, 200]$ can be seen in the timing diagram below.