Real-Time Systems — EDA223/DIT161

Final exam, March 12, 2018 at 08:30 – 12:30 in the M building

---

Examiner:
Professor Jan Jonsson, Department of Computer Science and Engineering

Responsible teacher:
Jan Jonsson, phone: 031–772 5220
Visits the exam at 09:30 and 11:30

Aids permitted during the exam:
J. Nordlander, *Programming with the TinyTimber kernel*
Chalmers-approved calculator

Content:
The written exam consists of 6 pages (including cover and hand-in sheet), containing 7 problems worth a total of 60 points.

Grading policy:

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 posted in PingPong under 'Objectives & Progress'.

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 might be 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.
6. A hand-in sheet is available at the end of the exam script. Do not forget to submit it together with your other solution sheets!

Good Luck!

PROBLEM 1

State whether the following propositions are TRUE or FALSE. Each correct statement will give 0.5 points; each erroneous statement will give -0.5 points; an omitted statement gives 0 points. Although a motivation for a correct answer is not required, a convincing one gives another 0.5 points, while an erroneous/weak one gives another -0.5 points. **Quality guarantee**: The total result for this problem cannot be less than 0 points. (6 points)

a) By *priority inversion*, we mean a situation where the static priority assigned to a task is inversely proportional to the period of the task.

b) A *systematic time skew* in the execution of a task is caused by interference from other tasks with higher priority.

c) For an NP-complete problem to have *pseudo-polynomial* time complexity the largest number in the problem cannot be bounded by the input length (size) of the problem.

d) The GCC cross compiler that is used in the laboratory assignment generates code for the 68HC12 processor core.

e) The RM-US priority-assignment policy has a utilization guarantee bound that converges towards 41% as the number of processors become very large.

f) If a given task set is known to be not schedulable, a *necessary* feasibility test will always report the answer "no" when applied to that task set.

PROBLEM 2

The following questions are related to network communication.

a) Describe how message *transmission delay* is a function of the properties of the message being sent and the network medium used. (2 points)

b) Describe why the Ethernet protocol is not suitable for use with hard real-time systems. (1 point)

c) Describe the *binary countdown* algorithm as used in CAN (Controller Area Network). Your answer should include an example illustrating how the algorithm works with three nodes with unique message identifiers. (5 points)

PROBLEM 3

Most scheduling analysis techniques assume the worst-case execution time (WCET) to model the computational demand of a real-time task. One of the earliest methods for WCET analysis was presented by Shaw in the end of the 1980s.

Assume that the function `main` (see below) is used as part of a real-time program and that the function, when called, is allowed to take at most 65 μs to execute.

- Each declaration and assignment statement costs 1 μs to execute.
- A function call costs 2 μs plus WCET for the function in question.
- Each compare and return statement costs 2 μs.
- Each addition and subtraction operation costs 3 μs.
- Each multiply operation costs 5 μs.

- All other language constructs can be assumed to take 0 μs to execute.

```
int calculateA(int x) {
   int temp;

   if (x == 0)
      temp = 1;
   else
      if (x == 1)
         temp = 1;
      else
         temp = x * calculateA(x-1);

   return temp;
}

int calculateB(int x) {
   int temp;

   temp = x * calculateA(x) + calculateA(x-1);

   return temp;
}

void main() {
   int N;
   int ans;

   N = 2;

   ans = calculateB(N);
}
```

a) Derive WCET for function `main` by using Shaw's method and check whether the deadline of the function (65 μs) will be met or not. (6 points)

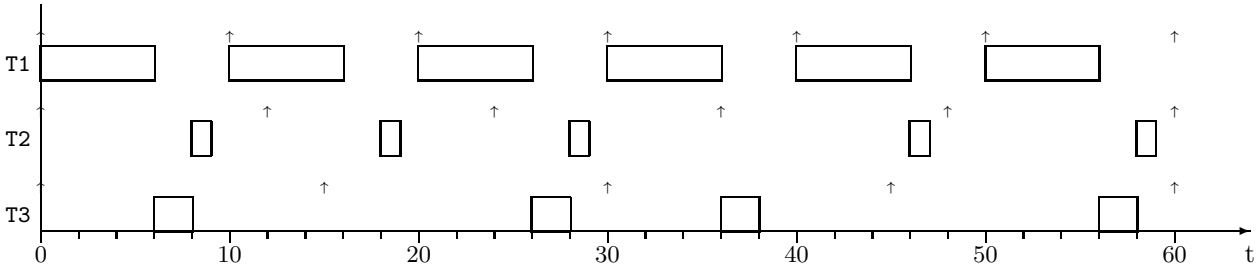
b) Derive the maximum cost that the multiply operation can have without causing the deadline of the `main` function (65 μs) to be missed. (3 points)

c) Explain why is it preferred that WCET estimates for tasks in a real-time system are *pessimistic* as well as *tight*. (3 points)

PROBLEM 4

With the TinyTimber kernel it is possible to implement periodic activities in a C program. Consider a real-time system with three independent periodic tasks: `T1`, `T2` and `T3`, whose execution are controlled by a cyclic time table with a hyper-period (LCM) of 60 ms. Within each hyper-period the instances of the tasks are scheduled according to the following:

- Task `T1` has a period of 10 ms and an execution time of 6 ms. Its first execution is scheduled during time interval [0,6), and the subsequent executions at time intervals [10,16), [20,26), [30,36), [40,46), and [50,56).
- Task `T2` has a period of 12 ms and an execution time of 1 ms. Its first execution is scheduled during time interval [8,9), and the subsequent executions at time intervals [18,19), [28,29), [46,47), and [58,59).
- Task `T3` has a period of 15 ms and an execution time of 2 ms. Its first execution is scheduled during time interval [6,8), and the subsequent executions at time intervals [26,28), [36,38), and [56,58).



You should write a TinyTimber program implementing the execution of three methods `T1()`, `T2()` and `T3()` corresponding to the cyclic time table described above. On a separate sheet at the end of this exam paper you find a C-code template. Add the missing program code directly on that sheet and hand it in for grading together with the rest of your solutions. The code for the functions `Action6ms()`, `Action1ms()` and `Action2ms()` is assumed to already exist. (6 points)

PROBLEM 5

Consider a real-time system with three independent periodic tasks and a run-time system that uses 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)

PROBLEM 6

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 of each other but share three exclusive resources protected by the binary semaphores S_a, S_b, and S_c. The tasks use the resources in the following way: Task τ_1 uses the resources protected by S_a and S_c. Task τ_2 uses the resources protected by S_a and S_b. Task τ_3 only uses the resource protected by S_b. Task τ_4 only uses the resource protected by S_c. The table below shows $H_{i,j}$, the maximum time that task τ_i may lock semaphore S_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. Assume that the Immediate Ceiling Priority Protocol (ICPP) is used for handling the shared resources. (12 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 time $t = 0$.

	C_i	T_i
τ_1	10	25
τ_2	18	40
τ_3	2	10
τ_4	$T_4/3$	T_4
τ_5	1	8

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_4 such that (i) C_4 is also an integer and (ii) all the task deadlines are met. Show the assignment of the tasks to the processors. (6 points)

```
#include "TinyTimber.h"

typedef struct {
    Object super;


} PeriodicTask;

Object app = initObject();

PeriodicTask ptask1 = { initObject()                                    };
PeriodicTask ptask2 = { initObject()                                    };
PeriodicTask ptask3 = { initObject()                                    };


void T1(PeriodicTask *self, int u) {
    Action6ms(); // procedure doing time-critical work


}

void T2(PeriodicTask *self, int u) {
    Action1ms(); // procedure doing time-critical work







}

void T3(PeriodicTask *self, int u) {
    Action2ms(); // procedure doing time-critical work







}

void kickoff(Object *self, int u) {




}

main() {
    return TINYTIMBER(&app, kickoff, 0);
}
```