Introduction to Real-Time Systems — LET626

Final exam, May 31, 2017 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 at 15:00, and then at several occasions.

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

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

Grading policy:
24–35 points ⇒ grade 3
36–47 points ⇒ grade 4
48–60 points ⇒ grade 5

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 can be written in Swedish or 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) For a *sporadic* task there is a maximum time interval between two, subsequent, arrivals that is guaranteed to never be exceeded.

c) The response-time test for global fixed-priority scheduling is an exact feasibility test.

d) The RM-US priority-assignment policy has a utilization guarantee bound that converges towards 50% as the number of processors become very large.

e) TinyTimber's AFTER() construct enables scheduling of periodic tasks without systematic time skew.

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

In real-time systems that employ concurrent execution of multiple tasks with shared resources there is a potential risk that *deadlock* may occur.

a) State the four conditions for deadlock to occur in such systems. (4 points)

b) In such systems where tasks are assigned static priorities it is possible to avoid deadlock by using a run-time protocol that supports *ceiling priorities*. Describe the basic idea of a priority ceiling protocol (such as ICPP). (4 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 `Control` (see below) is used as part of a real-time program and that the function, when called, is allowed to take at most 73 μs to execute.

The system has one input port, located at address 0x40021010, and one output port, located at address 0x40020C14. The input port is connected to a sensor that delivers 8-bit values in the range $[-9, +9]$.

```
#define Inport  (*((volatile signed char *)(0x40021010)))
#define Outport (*((volatile signed char *)(0x40020C14)))
```

```
int Calc(int x) {
   int i;
   int r;
   i = 0;
   r = x;
   while (i < 3) {
      r = r * x;
      i = i + 1;
   }
   r = r - 1;
   return r;
}

void Control() {
   int c;
   int r;
   c = Inport;
   r = Calc(c) / 3;
   if (r <= 800)
       r = r >> 3;
   else
       r = (3 * r) / 289 + 2;
   Outport = r;
}
```

Use Shaw's method to solve sub-problems a), b) and c). To that end, assume the following costs:

- Each declaration and assignment statement costs 1 μs to execute.
- A function call costs 2 μs plus WCET for the function in question.
- Each evaluation of the logical condition in an `if`-statement or a `while`-statement costs 2 μs.
- Each shift operation ('>>') costs 2 μs.
- Each add and subtract operation costs 3 μs.
- Each multiply operation costs 5 μs.
- Each divide operation costs 8 μs.
- Each return statement costs 2 μs.
- All other language constructs can be assumed to take 0 μs to execute.

a) Show that function `Control` will not meet its deadline of 73 μs with the code given above and an input port value range of $[-9, +9]$. (6 points)

b) What is the largest input port data range for which function `Control` will be able to meet its deadline of 73 μs with the code given above? (3 points)

c) A colleague of yours suggests that you replace the code for the subroutine `Calc` with the code given below, which is functionally equivalent to the old code. Is the new code a good replacement from a timing point of view? (3 points)

```
int Calc(int x) {
   int r;
   r = x * x - 1;
   r = r * (r + 2);
   return r;
}
```

PROBLEM 4

With the TinyTimber kernel it is possible to implement periodic activities in a C program. Consider a real-time system with two independent periodic tasks: `T1` and `T2`, with the following properties:

- Task `T1` has a period of 95 ms, and should start executing for the first time at time t = 0 ms at the earliest. The relative deadline for task `T1` is 40 ms.
- Task `T2` has a period of 160 ms, and should start executing for the first time at time t = 70 ms at the earliest. The relative deadline for task `T2` is 85 ms.

You should write a TinyTimber program implementing the execution of two methods `T1()` and `T2()` corresponding to the tasks mentioned above. On a separate sheet at the end of this exam paper you find a C-code template. For sub-problem a) below 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 `Action1()` and `Action2()` is assumed to already exist.

a) Implement the code for the execution of the two methods `T1()` and `T2()` so that they have the same timing behavior as the corresponding tasks mentioned above. Add the missing TinyTimber statements directly in the provided C-code template. (3 points)

Some general questions relating to the TinyTimber run-time system:

b) Describe how priorities are assigned to scheduled activities (such as the two periodic tasks above) in TinyTimber. (2 points)

c) Describe how mutual exclusion is achieved in TinyTimber. (2 points)

d) State what type of shared-resource protocol is used by the TinyTimber run-time system. (1 point)

PROBLEM 5

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

a) Explain the principle behind the RM priority assignment approach. (1 point)

b) Describe under what conditions RM is an optimal priority assignment for preemptive scheduling on one processor. (2 points)

For each task τ_i it applies that its deadline D_i is equal to the period T_i. Both tasks arrive the first time at time 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.

c) 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. (4 points)

d) 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 three periodic tasks and a run-time system that employs preemptive earliest-deadline-first (EDF) scheduling. The table below shows C_i (WCET), D_i (deadline) and T_i (period) for each task τ_i. All tasks arrive the first time at time 0.

	C_i	D_i	T_i
τ_1	3	4	4
τ_2	1	4	10
τ_3	3	15	20

a) Explain why Liu & Layland's utilization-based test for EDF scheduling may not be used for the given task set. (1 points)

b) Perform processor-demand analysis to determine the schedulability of the task set. (5 points)

c) Verify the outcome of the analysis in sub-problem b) by constructing a timing diagram for the execution of the three tasks that spans from time $t = 0$ to time $t = LCM(T_1, T_2, T_3)$. (2 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 the first time at time 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)

List of useful expressions and equations.

$$n\left(2^{1/n}-1\right)$$

$$m\left(2^{1/2}-1\right)$$

$$\frac{m^2}{3m-2}$$

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j$$

$$C_P(0,L) = \sum_{i=1}^{n} \left(\left\lfloor \frac{L-D_i}{T_i} \right\rfloor + 1 \right) C_i$$

$$R_i^{n+1} = C_i + \frac{1}{m} \sum_{\forall j \in hp(i)} \left(\left\lceil \frac{R_i^n}{T_j} \right\rceil \cdot C_j + C_j \right)$$

```
#include TinyTimber.h

typedef struct {
    Object super;
    char *id;



} PeriodicTask;

Object app = initObject();

PeriodicTask ptask1 = { initObject(), "Task 1"                    };
PeriodicTask ptask2 = { initObject(), "Task 2"                    };



void T1(PeriodicTask *self, int u) {
    Action1(); // procedure doing time-critical work



}


void T2(PeriodicTask *self, int u) {
    Action2(); // procedure doing time-critical work



}

void kickoff(PeriodicTask *self, int u) {




}

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