Real-Time Systems — EDA222/DIT161

Final Exam, March 14, 2013 at 8:30 – 12:30 in the M building

Examiner:
Professor Jan Jonsson

Questions:
Risat Pathan, phone: 076 214 8509

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

Content:
The written exam consists of 8 pages (including cover), containing 7 problems worth a total of 60 points.

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

Solution:
Posted on the course home page on Friday, March 15, 2013 at 09:00

Results:
Posted on the course home page on Thursday, April 4, 2013 at 09:00.

Inspection:
Room 5128, Rännvägen 6 B, on Thursday, April 4, 2013 at 13:00–15:00. Inspection at another occasion could be arranged by contacting the course examiner.

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 I cannot read your solution, I will assume that it is wrong.

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) There is *no* multiprocessor scheduling algorithm with 100% resource utilization.

b) Ada protected objects use *condition variables* to implement synchronization.

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

d) Special *resource sharing protocols* are used for mutual exclusion in both preemptive and non-preemptive scheduling algorithms.

e) Switched Ethernet with deterministic queuing policies for input and output buffers in the switches guarantees *bounded* queuing delay.

f) Priority ceiling protocol (PCP) and immediate ceiling priority protocol (ICPP) are both deadlock free protocols.

PROBLEM 2

A fundamental prerequisite for correct concurrent execution of multiple tasks with shared resources is that the run-time system can guarantee mutual exclusion.

a) State the four conditions for deadlock to occur? (4 points)

b) State the techniques offered at the machine level to provide mutual exclusion in single processor and multiprocessors? (2 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 140 μs to execute where.

- 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 statement costs 2 μs.
- Each addition and subtraction operation costs 3 μs.
- Each multiplication operation costs 4 μs.
- Each return statement costs 2 μs.
- Each modulo operation costs 5 μs.
- The function `abs()` is predefined and costs 5 μs. This function computes and returns the absolute value of the number given as its parameter. For example, `abs(-5)` returns 5. Assume that the overhead to call function `abs()` is already included in its WCET estimation.
- All other language constructs can be assumed to take 0 μs to execute.

```
int FuncA(int x){

    if(x == 0)
        return 1;
    else
        return (x * FuncA (x - 1));
}


int FuncB(int y){

    int z;
    z = abs(y);

    if((z % 3) != 0){
        if (((z % 3) % 2) != 0)
            return z - 1;
        else
            return z + 1;
    }

    return z;
}


int main(){

    char Flag;
    int result;
    int P;
    int Q;
    P = -3;
    Q = -1;

    if((FuncB(Q) % 3) != 0)
        result = abs(P);
    else
        result = FuncA(abs(P) + FuncB(Q));

    if (result)
        Flag = 'T';
    else
        Flag = 'F';
    return 1;
}
```

a) Derive WCET for function `main` by using Shaw's method and check whether the function's deadline will be met or not. (8 points)

b) Identify all the false paths in the problem given in subproblem a) (2 points)

PROBLEM 4
(This question is only for students registered in academic year 2011/2012 or 2012/2013)

With the TinyTimber kernel it is possible to implement periodic activities in a C program. Consider a clock with three hands: second, minute, and hour. Use the 12-hour `hh:mm:ss` time format (i.e., the minimum value is `00:00:00` and the maximum is `11:59:59`). Each time the hour hand advances, a `bell` mechanism executes, and it takes *three* seconds to finish it. The table below shows the full timing specification for the clock hands.

Hand	Offset	Period	Deadline	Execution Time
Second	1s	1s	250ms	250ms
Minute	1m	1m	250ms	250ms
Hour	1h	1h	5s	3s

(ms – millisecond, s – second, m – minute, h – hour)

On a separate sheet at the end of this exam paper you find a C-code template. Write your solutions to the following subproblems directly on that sheet and hand it in for grading together with the rest of your solutions. The code for the function `bell()` is assumed to already exist.

a) Implement the clock using the template given at the end of this exam paper. Each hand is an independent task complying with the specification in the table given above. The clock starts at `00:00:00`. (4 points)

b) Draw a schedule (i.e., timing diagram) for the three tasks assuming a starting point at `01:59:58` and an end point at `02:00:07`. (4 points)

PROBLEM 4
(This question is only for students registered in academic year 2010/2011 or earlier)

a) Explain the purpose of the ADA pragma `Volatile`. Give an example of a proper use of the pragma in a program. (3 points)

b) Write a protected object `Circular_Buffer` that handles a circular buffer with room for 100 data records of type `Data` using the following template. The protected object should have two entries, `Put` and `Get`. Producer tasks should be able to insert data records in the buffer via entry `Put`. If the buffer is full, a task that calls `Put` should be blocked. Consumer tasks should be able to remove data records from the buffer via entry `Get`. If the buffer is empty, a task that calls `Get` should be blocked. (5 points)

```
type Buffer is array (Integer range <>) of Data;

protected type Circular_Buffer is
  entry Put(D : in Data);
  entry Get(D : out Data);

  private
    N : constant := 100;
    A : Buffer(1..N);
    ... ... ...

end Circular_Buffer;
```

```
protected body Circular_Buffer is
     ... ... ...
end Circular_Buffer;
```

PROBLEM 5

There are two approaches for scheduling tasks on multiprocessor platform: the partitioned approach and the global approach. The table below shows C_i (WCET) and T_i (period) for the six periodic tasks of a real-time application. The relative deadline of each task is equal to its period.

	C_i	T_i
τ_1	1	10
τ_2	2	20
τ_3	2	4
τ_4	30	100
τ_5	8	25
τ_6	6	19

Consider that there are two different choices of multiprocessor platform available in the market for implementing the application. The first choice (called Type-A multiprocessor platform) has 4 single processors hosted on the same platform while the second choice (called Type-B multiprocessor platform) has 8 single processors hosted on the same platform. The price of Type-B multiprocessor is 500 SEK higher than the price of Type-A multiprocessor. You, as the system designer, can use either `RM-US` global scheduling or `RMFF` partitioned scheduling.

a) Which scheduling algorithm and which type of multiprocessor platform you will select for implementing the system such that all the deadlines of the tasks are met? Motivate your answer. (7 points)

b) Global fixed-priority scheduling under rate-monotonic (RM) priority assignment has weak theoretical framework. Explain two underlying causes for such weak theoretical framework of global fixed-priority scheduling. (3 points)

PROBLEM 6

There are two different paradigms for scheduling tasks on uniprocessor: static scheduling and dynamic scheduling.

a) Write one advantage and one disadvantage of static scheduling? How is the schedule generated in static scheduling? (4 points)

b) Consider a task set having four periodic tasks τ_1, τ_2, τ_3 and τ_4 such that the periods T_1, T_2, T_3 and T_4 of the four tasks are related as follows:

$$T_2 = 2T_1 \qquad T_3 = 4T_1 \qquad T_4 = 8T_1$$

The relative deadline of each task is equal to its period. The WCET of the tasks are such that the total utilization of the task set is 1. Show that this task set is schedulable using uniprocessor rate-monotonic (RM) scheduling algorithm. (6 points)

c) Consider a task set that is schedulable using deadline-monotonic (DM) scheduling algorithm on uniprocessor where the offset for at least one task is positive. Can we also guarantee that this task set is schedulable using DM if the offset of each task is zero? Why or why not? (2 points)

PROBLEM 7

Consider a real-time system with three periodic tasks and a run-time system that employs uniprocessor preemptive 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 three tasks. The initial arrival time of each task is not known. All values are given in milliseconds.

	C_i	D_i	T_i
τ_1	8	18	18
τ_2	6	17	28
τ_3	5	?	30

The value of D_3 is greater than 20, i.e., $D_3 > 20$. The three tasks are not independent, but share two exclusive resources R_a and R_b. The run-time system employs the Immediate Ceiling Priority Protocol (ICPP) to resolve resource request conflicts. The tasks use the resources in the following way:

- Task τ_1 first requests R_a and then, while using R_a, requests R_b; then, after releasing the two resources, τ_1 again requests only R_a.
- Task τ_2 requests R_b; and after releasing R_b, requests R_a
- Task τ_3 requests R_a.

The table below shows $H_{i,j}$, the maximum time (in milliseconds) that task τ_i locks resource R_j during its execution.

	$H_{i,a}$	$H_{i,b}$
τ_1	2	3
τ_2	1	4
τ_3	3	-

Use a suitable analysis method to determine the minimum value of D_3 such that the tasks in the system are schedulable. (8 points)

Template Code for Problem 4
(Submit this page with rest of the solutions)

```
#include "TinyTimber.h"

typedef struct{
        Object super;
        int value;
} PeriodicTask;

Object app = initObject();
PeriodicTask pt1 = {initObject(), 0};
PeriodicTask pt2 = {initObject(), 0};
PeriodicTask pt3 = {initObject(), 0};

void advance1(PeriodicTask *self, int period) {









}

void advance2(PeriodicTask *self, int period) {









     bell();



}

void kickoff(PeriodicTask *self, int unused) {









}


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