Real-Time Systems — EDA222/DIT161

Solutions to final exam March 14, 2013

PROBLEM 1

a) False: No global or partitioned multiprocessor fixed-priority scheduling can have a utilization bound greater than 50%. But the *pfair* scheduling guarantees 100% resource scheduling.

b) False: Monitors use condition variables to implement synchronization.

c) False: The response-time test for global fixed-priority scheduling is a sufficient feasibility test since one extra instance of each higher-priority task must be (pessimistically) accounted for in the interference analysis.

d) False: Mutual exclusion is automatically ensured in non-preemptive scheduling algorithms; hence, does not need any special resource sharing protocol.

e) True: Deterministic queuing policies guarantees an upper bound on queuing delay.

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

PROBLEM 2

a) See lecture notes for Lecture 4 (slide 7).

b) See lecture notes for Lecture 4 (slide 22).

PROBLEM 3

a) The WCET of main is dependent on the WCET of functions "FuncA" and "FuncB". So, we first calculate the WCET of functions "FuncA" and "FuncB".

WCET of "FuncA": There are two cases for calculating the WCET of FuncA. Case(i), $x == 0$ case(ii) $x! = 0$.

$$
\begin{aligned}
&Case(i) WCET(FuncA(x == 0)) = \{compare, x == 0\} + \{return\} = 2 + 2 = 4\\
&Case(ii) WCET(FuncA(x)) = \{compare, x == 0\} + \{sub, x - 1\} + \{call, FuncA(x - 1)\}\\
&+ WCET(FuncA(x - 1)) + \{multiply, x * FuncA(x - 1)\} + \{return\}\\
&= 2 + 3 + 2 + WCET(FuncA(x - 1)) + 4 + 2\\
&= 13 + WCET(FuncA(x - 1))
\end{aligned}
$$

WCET of "FuncB": There are two cases for calculating the WCET of FuncB: Case(i) z is a multiple of 3, Case(ii) z is not a multiple of 3

$$Case(i) WCET(FuncB(y)) = \{Dec, z\} + \{abs, y\} + \{Assign, z = abs(y)\}$$
$$+ \{mod, z\%3\} + \{Compare, (z\%3)! = 0\} + \{return\}$$
$$= 1 + 5 + 1 + 5 + 2 + 2 = 16$$

$$Case(ii) WCET(FuncB(y)) = \{Dec, z\} + \{abs, y\} + \{Assign, z = abs(y)\}$$
$$+ \{mod, z\%3\} + \{Compare, (z\%3)! = 0\} + \{mod, z\%3\} + \{mod, (z\%3)\%2\}$$
$$+ \{Compare, ((z\%3)\%2)! = 0\} + \{add\ or\ sub, (z - 1)\ or\ (z + 1)\} + \{return\}$$
$$= 1 + 5 + 1 + 5 + 2 + 5 + 5 + 2 + 3 + 2 = 31$$

WCET of "main": Now we can calculate the WCET of main.

$$WCET(main) = \{Dec, Flag\} + \{Dec, result\} + \{Dec, P\} + \{Dec, Q\}$$
$$+ \{Assign, P = -3\} + \{Assign, Q = -1\} + \{mod, FuncB(-1)\%3\}$$
$$+ \{call, FuncB(-1)\} + WCET(FuncB(-1)) + \{Compare, (FuncB(-1)\%3)! = 0\}$$
$$+$$

CASE1($(FuncB(-1)\%3)! = 0$): {abs, abs(-3)}+{Assign, result}

Or,

CASE2($(FuncB(-1)\%3) ==$ 0):

$$\{abs, abs(-3)\} + \{call, FuncB(-1)\} + WCET(FuncB(-1))$$
$$+ \{add, abs(-3) + FuncB(-1)\} + \{call, FuncA(abs(-3) + FuncB(-1))\}$$
$$+ WCET(FucA(abs(-3) + FuncB(-1))) + \{Assign, result\}$$
$$+ \{Compare, result\} + \{Assign, Flag =' T' ORFlag =' F'\} + \{return\}$$

Note that FuncB always returns a value that is a multiple of 3. Therefore, the condition for CASE1 is never true and we can calculate the WCET(main) by only considering CASE2.

$$WCET(main) = 1 + 1 + 1 + 1 + 1 + 1 + 5 + 2 + WCET(FuncB(-1)) + 2 + 5 + 2$$
$$+ WCET(FuncB(-1)) + 3 + 2 + WCET(FucA(abs(-3) + FuncB(-1))) + 1 + 2 + 1 + 2$$
$$= 13 + WCET(FuncB(-1)) + 9 + WCET(FuncB(-1))$$
$$+ 5 + WCET(FucA(abs(-3) + FuncB(-1))) + 6$$
$$= 33 + 2 * WCET(FuncB(-1)) + WCET(FucA(abs(-3) + FuncB(-1)))$$
$$= 33 + 2 * 31 + 43 = 138 \text{(deadline is met)}$$

where

$$WCET(FuncB(-1)) = WCET(0) = 31$$

and

$$WCET(FucA(abs(-3) + FuncB(-1))) = WCET(FucA(3 + 0)) = 43$$

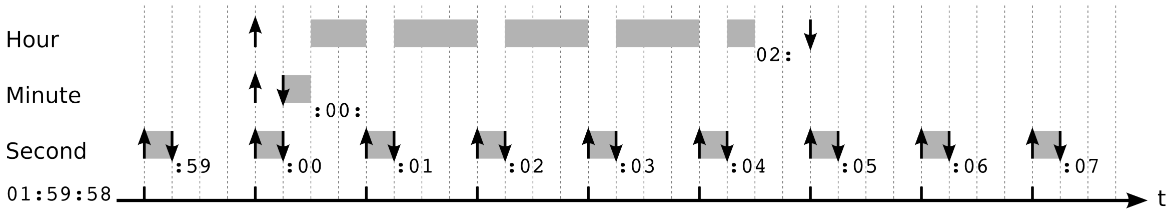
b) The false paths are:

- The condition "if((FuncB(Q)%3)!=0)" in the main function is never true since the return value of the "FuncB" is always a multiple of 3.
- The condition "if(result)" in the main function is always true and the statement in the else is never executed. This is because the value for "result" is never 0 due to the characteristic of "FuncA" which is a factorial function.

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

a)
```
#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) {

       if (self->value == 59) {
          self->value = 0;
       } else {
          self->value += 1;
       }

       SEND(SEC(period), MSEC(250), self, advance1, period);

  }

  void advance2(PeriodicTask *self, int period) {

       if (self->value == 11) {
          self->value = 0;
       } else {
          self->value += 1;
       }

       bell();

       SEND(SEC(period), SEC(5), self, advance2, period);

  }

  void kickoff(PeriodicTask *self, int unused) {

       SEND(SEC(1),    MSEC(250), &pt1, advance1, 1);
       SEND(SEC(60),   MSEC(250), &pt2, advance1, 60);
       SEND(SEC(3600), SEC(5),    &pt3, advance2, 3600);
  }

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

b) The timing diagram is shown below:



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

a) Assume a hardware port `IO_port` represented using a type, for example, `BITFIELD8` that has 8 fields `b0 ...b7` for 8 different IO pins of the port.

```
IO_port : BITFIELD8;
```

Consider the following loop code:

```
-- wait for device ready ...
while (IO_Port.b7 /= 0 ) loop
NULL;
end loop;
```

The code segment above runs the risk of being completely removed by the compiler optimization, unless the compiler is told that the value of IO Port may change outside the program control.

On the other hand, adding the following pragma statement to the code tells the compiler that IO Port might change and thus inhibits any optimization:

```
pragma Volatile(IO_Port);
```

b) `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);
 I,J : Integer range 1..N := 1;
 Count : Integer range 0..N := 0;
end Circular_Buffer;

protected body Circular_Buffer is

entry Put(D : in Data) when Count < N is begin

 A(I) := D;
 I := (I mod N) + 1;
 Count := Count + 1;

end Put;
```

```
entry Get(D : out Data) when Count > 0 is begin

 D := A(J);
 J := (J mod N) + 1;
 Count := Count - 1;

end Get;

end Circular_Buffer;
```

PROBLEM 5

a) The total utilization U of the task set is computed as follows:

$$\begin{aligned} U &= u_1 + u_2 + u_3 + u_4 + u_5 + u_6 \\ &= \frac{1}{10} + \frac{2}{20} + + \frac{2}{4} + \frac{30}{100} + \frac{8}{25} + \frac{6}{19} \\ &= 0.1 + 0.1 + 0.5 + 0.3 + 0.32 + 0.316 = 1.636 \end{aligned}$$

As a system designer, the decision can be made in different ways. One approach is to guide the decision by cost factor, for example, minimize the number of processors. So, we will verify if the task set is schedulable on Type-A processor having $m = 4$ single processors hosted on the same platform. To guarantee that all the deadlines are met, we have to find the utilization bound of RM-US and RMFF scheduling for $m = 4$.

Utilization bound for RM-US for $m = 4$ is $U_{RM-US} = \frac{m^2}{3m-2} = \frac{4^2}{3\cdot4-2} = \frac{16}{10} = 1.6$. Since $U > U_{RM-US}$, the task set can not be guaranteed to be schedulable on $m = 4$ processors using RM-US scheduling on Type-A multiprocessor. However, the utilization bound for RM-US for $m = 8$ is $U_{RM-US} = \frac{m^2}{3m-2} = \frac{8^2}{3\cdot8-2} = \frac{64}{22} \approx 2.91$. Since $U \leq U_{RM-US}$, the task set is guaranteed to be schedulable on $m = 8$ processors using RM-US scheduling on Type-B multiprocessor.

Utilization bound for RMFF for $m = 4$ is $U_{RMFF} = m \cdot (\sqrt{2} - 1) = m \cdot (\sqrt{2} - 1) = 1.656$. Since $U \leq U_{RMFF}$, the task set is guaranteed to be schedulable on $m = 4$ processors using RMFF scheduling on Type-A multiprocessor.

So, if minimizing the cost is the only objective for the design of the system, then RMFF scheduling on Type-A processor is the best choice.

b) See lecture notes for Lecture 15 (slide 15).

PROBLEM 6

a) See lecture notes for Lecture 11 (slides 21-23).

b) This task set is guaranteed to be schedulable using the earliest deadline first (EDF) scheduling since EDF has a guarantee utilization bound of 1.

Now consider a tie breaking policy for EDF scheduling as follows: if different instances of multiple tasks have the same absolute deadlines, then ties are broken by EDF giving higher priority to the instance of the task having smallest period, i.e., according to rate-monotonic (RM) priority. EDF scheduling with such tie breaking policy also has an utilization bound of 1 since the exact test for EDF is not based on any assumption regarding the tie breaking policy (please see lecture 12 slide 21).

Since the period of the four tasks are integer multiple of one another, the rate-monotonic scheduling is exactly same as the EDF scheduling with the tie breaking policy discussed in last paragraph. In other words, the schedule generated by EDF (with the above mentioned tie breaking policy) and the schedule generated by RM are the same if each larger period is integer multiple of each smaller period. Therefore, a task set having periods integer multiple of one another if scheduled using rate-monotonic scheduling has an utilization bound of 1. Consequently, the task set is schedulable using RM.

c) No. When the offsets of all the tasks are zero, it represents the critical instant (i.e., the worst-case) for uniprocessor DM scheduling. Therefore, a task in which no task has positive offset may not be schedulable using DM scheduling even if the same task set is schedulable using DM where at least one task has positive offset. (3 points)

PROBLEM 7

Define priority as follows: H = highest priority, M=medium priority and L = lowest priority. Since DM is used and $D_3 > 20 > D_1 > D_2$, the task priorities are as follows: task τ_2 has highest priority H, task τ_1 has medium priority M and task τ_3 has lowest priority L.

We first determine the ceiling priority for each resource:

$$ceil\{R_a\} = max\{M, H, L\} = H \quad \text{(since } \tau_1\text{, } \tau_2 \text{ or } \tau_3 \text{ uses resource } R_a\text{)} \tag{1}$$
$$ceil\{R_b\} = max\{M, H\} = H \quad \text{(since } \tau_1 \text{ or } \tau_2 \text{ uses resource } R_b\text{)} \tag{2}$$

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. Note that nested blocking is used by task τ_1. This could lead to accumulated critical region blocking times in the final blocking factor B_2.

$B_1 = max\{H_{3,a}\} = max\{3\} = 3$ (since τ_1 may be blocked only by τ_3 who uses resource R_a whose ceiling priorities are higher than the priority of τ_1)

$B_2 = max\{H_{1,a} + H_{1,b}, H_{1,a}, H_{3,a}\} = max\{2 + 3, 2, 3\} = 5$ (since τ_2 may be blocked by τ_1 and τ_3 that uses resources whose ceiling priorities are higher than or equal to the priority of τ_2; note accumulated blocking time due to the nested blocking by τ_1)

$B3 = 0$ (since τ_3 has the lowest priority among all tasks, and thereby per definition cannot suffer blocking)

We now calculate the response time of each task and compare it against the corresponding task deadline:

$R_2 = C_2 + B_2 = 6 + 5 = 11 \leq D_1 = 17$ (task τ_2 meets its deadline)

Assume that $R_1^0 = C_1 = 8$.

$$R_1^1 = C_1 + B_1 + \lceil \frac{R_1^0}{T_2} \rceil \cdot C_2 = 8 + 3 + \lceil \frac{8}{28} \rceil \cdot 6 = 17$$

$$R_1^2 = C_1 + B_1 + \lceil \frac{R_1^1}{T_2} \rceil \cdot C_2 = 8 + 3 + \lceil \frac{17}{28} \rceil \cdot 6 = 17.$$

Since $R_1^1 = R_1^2 = 17 \leq D_1 = 18,$ we have $R_2 = 18$. (task τ_1 meets its deadline)

Assume that $R_3^0 = C_3 = 5$.

$$R_3^1 = C_3 + B_3 + \lceil \frac{R_3^0}{T_2} \rceil \cdot C_2 + \lceil \frac{R_3^0}{T_1} \rceil \cdot C_1 = 5 + 0 + \lceil \frac{5}{28} \rceil \cdot 6 + \lceil \frac{5}{18} \rceil \cdot 8 = 19$$

$$R_3^2 = C_3 + B_3 + \lceil \frac{R_3^1}{T_2} \rceil \cdot C_2 + \lceil \frac{R_3^1}{T_1} \rceil \cdot C_1 = 5 + 0 + \lceil \frac{19}{28} \rceil \cdot 6 + \lceil \frac{19}{18} \rceil \cdot 8 = 27.$$

$$R_3^3 = C_3 + B_3 + \lceil \frac{R_3^2}{T_2} \rceil \cdot C_2 + \lceil \frac{R_3^2}{T_1} \rceil \cdot C_1 = 5 + 0 + \lceil \frac{27}{28} \rceil \cdot 6 + \lceil \frac{27}{18} \rceil \cdot 8 = 27.$$

Since $R_3^2 = R_3^3,$ the minimum value of D_3 is 27 and $R_3 = 27$. So, if $D_3 = 27$, then task τ_3 meets its deadline.