Real-Time Systems — EDA222/DIT161

Solutions to final exam March 10, 2014

PROBLEM 1

a) True: No scheduling algorithm can schedule a task set that requires more 100% capacity of the platform.

b) True: Due to fixed-priority based scheme, an upper bound on queuing delay can be computed.

c) False: No such test is known so far since the critical instant is not yet known.

d) False: Hard real-time guarantee can be provided for sporadic tasks since the inter-arrival time of consecutive jobs has a lower bound.

e) False: There is no deadlock in non-preemptive scheduling since we cannot have a circular wait.

f) False: Interrupts are local to each processor, i.e., applicable in uniprocessor system.

PROBLEM 2

a) See lecture notes for Lecture 5 (slide 3-4).

b) See lecture notes for Lecture 4 (slide 18).

PROBLEM 3

a) The WCET of main is dependent on the WCET of functions "FuncA", "FuncB", and "FuncC". "FuncA" calculates the Greatest Common Devisor of two values. "FuncC" looks for a value in an array of sorted elements using binary search algorithm.

WCET of "main":

$$
\begin{aligned}
WCET(main) &= \{Dec, Flag\} + \{Dec, result\} + \{Dec, P\} + \{Dec, Q\} + \{Dec, find\} + \{Dec, count\} \\
&+ \{Dec, data\} + \{Assign, data[6]\} + \{Assign, count = 6\} + \{Assign, P = -16\} + \{Assign, Q = 12\} \\
&+ \{Assign, Flag = `F'\} + \{abs, abs(P)\} + \{mod, (abs(P)\%Q)\} + \{Comp, (abs(P)\%Q)! = 0\} \\
&+ \{abs, abs(P)\} + \{call, FuncA(abs(P), Q)\} + WCET(FuncA(16, 12)) \\
&+ \{Assign, find = FuncA(abs(P), Q)\} + \{sub, count - 1\} + \{call, FuncC(data, find, 0, count - 1)\} \\
&+ WCET(FuncC(data, 4, 0, 5)) + \{assign, result = FuncC(data, find, 0, count - 1)\} \\
&+ \{comp, result == -1\} + \{comp, result <= 16\} + \{assign, Flag = T\} + \{return, 2\} \\
&= 45 + WCET(FuncA(16, 12)) + WCET(FuncC(data, 4, 0, 5))
\end{aligned}
$$

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

$$Case(i) WCET(FuncA(x, y == 0)) = \{Comp, y == 0\} + \{return, x\} = 2 + 2 = 4$$
$$Case(ii) WCET(FuncA(x = 16, y = 12)) = \{Comp, y == 0\} + \{mod, x\%y\} + \{call, FuncA(y, x\%y)\}$$
$$+ WCET(FuncA(12, 16\%12)) + \{return, FuncA(y, x\%y)\}$$
$$= 11 + WCET(FuncA(12, 16\%12))$$

$$WCET(FuncA(x = 12, y = 16\%12)) = \{Comp, y == 0\} + \{mod, x\%y\} + \{call, FuncA(y, x\%y)\}$$
$$+ WCET(FuncA(4, 12\%4)) + \{return, FuncA(y, x\%y)\}$$
$$= 11 + WCET(FuncA(4, 12\%4))$$

==>
$$WCET(FuncA(16, 12)) = 11 + 11 + 4 = 26$$

WCET of "FuncC":

$$WCET(FuncC(data, x = 4, y = 0, z = 5)) = \{Dec, start\} + \{Dec, end\} + \{Dec, mid\} + \{assign, start = 0\}$$
$$+ \{assign, end = 5\} + \{sub, end - start\} + \{div, (end - start)/2\} + \{add, start + (end - start)/2\}$$
$$+ \{assign, mid = start + (end - start)/2)\} + \{Comp, start > end\} + \{Comp, data[mid] == x\}$$
$$+ \{Comp, data[mid] > x\} + \{add, mid + 1\} + \{call, FuncC(data, x, mid + 1, end)\}$$
$$+ WCET(FuncC(data, 4, 3, 5)) + \{return, FuncC(data, x, mid + 1, end)\}$$
$$= 29 + WCET(FuncC(data, 4, 3, 5))$$

$$WCET(FuncC(data, x = 4, y = 3, z = 5)) = \{Dec, start\} + \{Dec, end\} + \{Dec, mid\} + \{assign, start = 3\}$$
$$+ \{assign, end = 5\} + \{sub, end - start\} + \{div, (end - start)/2\} + \{add, start + (end - start)/2\}$$
$$+ \{assign, mid = start + (end - start)/2)\} + \{Comp, start > end\} + \{Comp, data[mid] == x\}$$
$$+ \{Comp, data[mid] > x\} + \{sub, mid - 1\} + \{call, FuncC(data, x, start, mid - 1)\}$$
$$+ WCET(FuncC(data, 4, 3, 3)) + \{return, FuncC(data, x, start, mid - 1)\}$$
$$= 29 + WCET(FuncC(data, 4, 3, 3))$$

$$WCET(FuncC(data, x = 4, y = 3, z = 3)) = \{Dec, start\} + \{Dec, end\} + \{Dec, mid\} + \{assign, start = 3\}$$
$$+ \{assign, end = 5\} + \{sub, end - start\} + \{div, (end - start)/2\} + \{add, start + (end - start)/2\}$$
$$+ \{assign, mid = start + (end - start)/2)\} + \{Comp, start > end\} + \{Comp, data[mid] == x\}$$
$$+ \{call, FuncB(data[mid])\} + WCET(FuncB(data[3])) + \{return, FuncB(data[mid])\}$$
$$= 24 + WCET(FuncB(data[3]))$$

==>
$$WCET(FuncC(data, 4, 0, 5)) = 29 + 29 + 24 + WCET(FuncB(data[3]))$$
$$= 82 + WCET(FuncB(data[3]))$$

WCET of "FuncB":

$$WCET(FuncB(y=4)) = \{Dec, z\} + \{Assign, z=2\} + \{comp, 4 == 0\} + \{comp, 4>1\} + \{mul, 2*2\}$$
$$+ \{assign, 4 = 2*2\} + \{sub, 4-1\} + \{assign, 3 = 4-1\} + \{comp, 3>1\} + \{mul, 4*4\}$$
$$+ \{assign, 16 = 4*4\} + \{sub, 3-1\} + \{assign, 2 = 3-1\} + \{comp, 2>1\} + \{mul, 16*16\}$$
$$+ \{assign, 256 = 16*16\} + \{sub, 2-1\} + \{assign, 1 = 2-1\} + \{comp, 1>1\} + \{return, 2\}$$
$$= 41$$

WCET of "main":

$$WCET(main) = 45 + WCET(FuncA(16, 12)) + WCET(FuncC(data, 4, 0, 5))$$
$$= 45 + 26 + 82 + 41 = 194$$

The deadline is missed

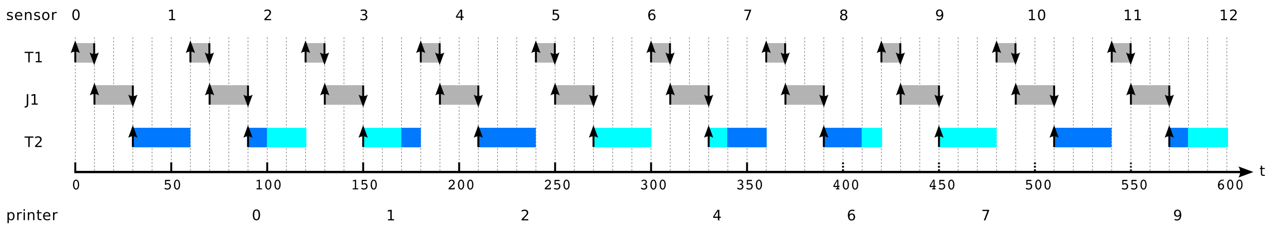
b) According to the inputs provided here, the false paths are:

- The condition "if(abs(P)%Q!=0)" in the main function is always true, so "find=Q" is a false path.
- In the main function, "result" is equal to 256, so the condition "if(result==-1)" is always false which makes "return -1" a false path.
- In the main function, result is equal to 256 so the condition "if(result¡=16)" is always false which makes "Flag = T and return 1" two false paths.
- In FuncB, the initial value for y is 4, so the condition "if(y==0)" is always false which makes "return 1" a false path.
- In FuncC, the value of start is never greater than end, thus "return -1" is a false path.

PROBLEM 4

a)
```
#include "TinyTimber.h"

typedef struct{
    Object super;
    char *id;
} RTprocess;

Object app = initObject();
RTprocess rtp1 = {initObject(), "T1"};
RTprocess rtp2 = {initObject(), "J1"};
RTprocess rtp3 = {initObject(), "T2"};


void exec1(RTprocess *self, int u) {
    SEND(MSEC(60), MSEC(10),  self, exec1, 0);
    SEND(MSEC(10), MSEC(20), &rtp2, exec2, 0);
    work1(); // executes for 10ms
}

void exec2(RTprocess *self, int u) {
    work2(); // executes for 20ms
}

void exec3(RTprocess *self, int u) {
    SEND(MSEC(60), MSEC(0), self, exec3, 0); // or AFTER(MSEC(60), self, exec3, 0);
    work3(); // executes for 40ms
}


void kickoff(RTprocess *self, int u) {
    SEND(MSEC(0),  MSEC(10), &rtp1, exec1, 0);
    SEND(MSEC(30), MSEC(0),  &rtp3, exec3, 0); // or AFTER(MSEC(30), &rtp3, exec3, 0);
}

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

b) The timing diagram is shown below:



PROBLEM 5

a) The DM priority ordering is as follows: τ_1 is highest, τ_2 is the medium, and τ_1 is the lowest priority task. By simulating the DM schedule of the tasks, if $C_3 = 3$, then there is a deadline miss at time $t = 14$. By simulating the DM schedule of the tasks, if $C_3 = 2$, then all deadlines are met.

Since the same schedule in time 0 to 12 is repeated again from time $t = 12$, we have to generate the table from time $t = 0$ to $t = 12$. Within $[0, 12)$, the three tasks are executed as follows: task τ_1 executes in (1,2), (4,5), (7,8), and (10,11); task τ_2 executes in (0,1), (5,6), and (8,9); and task τ_3 executes in (2,4), (9,10), (11,12).

b) See lecture notes for Lecture 10 (slide 21-22).

c) See lecture notes for Lecture 10 (slide 7).

PROBLEM 6

a) Yes. The utilization bound of EDF is 100% when $D_i = T_i$ for all tasks. A DM-schedulable task set must have total utilization not larger than 100%.

b) We have to apply processor demand analysis. The least common multiple of the periods is 20. The set of control points for task τ_1 is $K_1 = \{5, 10, 15, 20\}$. The set of control points for task τ_2 is $K_2 = \{6, 16\}$. Finally, the set of control points for task τ_3 is $K_3 = \{2C_3\}$ since $2C_3 + T_3 > 20$. Therefore, the set of all the control points for all the tasks is

$$K = K_1 \cup K_2 \cup K_3 = \{5, 6, 10, 15, 16, 20\} \cup \{2C_3\}$$

Since $C_3 \geq 5$, the value of the control point $2C_3$ must satisfy $2C_3 \geq 10$.

The processor demand for $L = 5$ is

$$\left(\left\lfloor \frac{5 - D_1}{T_1} \right\rfloor + 1\right) \cdot C_1 + \left(\left\lfloor \frac{5 - D_2}{T_2} \right\rfloor + 1\right) \cdot C_2 + \left(\left\lfloor \frac{5 - D_3}{T_3} \right\rfloor + 1\right) \cdot C_3$$
$$= \left(\left\lfloor \frac{5 - 5}{5} \right\rfloor + 1\right) \cdot 2 + \left(\left\lfloor \frac{5 - 6}{10} \right\rfloor + 1\right) \cdot 3 + \left(\left\lfloor \frac{5 - 2C3}{20} \right\rfloor + 1\right) \cdot C_3 = 2 \leq L = 5 \quad (OK!)$$

The processor demand for $L = 6$ is

$$\left(\left\lfloor \frac{6 - D_1}{T_1} \right\rfloor + 1\right) \cdot C_1 + \left(\left\lfloor \frac{6 - D_2}{T_2} \right\rfloor + 1\right) \cdot C_2 + \left(\left\lfloor \frac{6 - D_3}{T_3} \right\rfloor + 1\right) \cdot C_3$$
$$= \left(\left\lfloor \frac{6 - 5}{5} \right\rfloor + 1\right) \cdot 2 + \left(\left\lfloor \frac{6 - 6}{10} \right\rfloor + 1\right) \cdot 3 + \left(\left\lfloor \frac{6 - 2C3}{20} \right\rfloor + 1\right) \cdot C_3 = 2 + 3 = 5 \leq L = 6 \quad (OK!)$$

The processor demand for $L = 10$ (assuming that $2C_3 = 10$) is

$$\left(\left\lfloor \frac{10 - 5}{5} \right\rfloor + 1\right) \cdot 2 + \left(\left\lfloor \frac{10 - 6}{10} \right\rfloor + 1\right) \cdot 3 + \left(\left\lfloor \frac{10 - 2C3}{20} \right\rfloor + 1\right) \cdot C_3$$
$$= 4 + 3 + C_3 = 7 + C_3 = 7 + 5 = 12 > L = 10 \quad (NOT\ OK!)$$

Therefore, to guarantee schedulability, we must have $2C_3 > 10$. Since C_3 is an integer, the next possible choice of $2C_3$ is 12. Consequently, the control point for $2C_3$ is $2C_3 = 12$. The processor demand for $L = 12$ (assuming $2C_3 = 12$) is

$$\left(\left\lfloor \frac{12-5}{5} \right\rfloor + 1\right) \cdot 2 + \left(\left\lfloor \frac{12-6}{10} \right\rfloor + 1\right) \cdot 3 + \left(\left\lfloor \frac{12-2C3}{20} \right\rfloor + 1\right) \cdot C_3$$
$$= 4 + 3 + C_3 = 7 + C_3 = 7 + 6 = 13 > L = 12 \quad (NOT\ OK!)$$

Therefore, to guarantee schedulability, we must have $2C_3 > 12$. Since C_3 is an integer, the next possible choice of $2C_3$ is 14. Consequently, the control point for $2C_3$ is $2C_3 = 14$. The processor demand for $L = 14$ (assuming $2C_3 = 14$) is

$$\left(\left\lfloor \frac{14-5}{5} \right\rfloor + 1\right) \cdot 2 + \left(\left\lfloor \frac{14-6}{10} \right\rfloor + 1\right) \cdot 3 + \left(\left\lfloor \frac{14-2C3}{20} \right\rfloor + 1\right) \cdot C_3$$
$$= 4 + 3 + C_3 = 7 + C_3 = 7 + 7 = 14 <= L = 14 \quad (OK!)$$

The next control points in K is at $L = 15$. The processor demand for $L = 15$ (assuming $2C_3 = 14$) is

$$\left(\left\lfloor \frac{15-5}{5} \right\rfloor + 1\right) \cdot 2 + \left(\left\lfloor \frac{15-6}{10} \right\rfloor + 1\right) \cdot 3 + \left(\left\lfloor \frac{15-2C3}{20} \right\rfloor + 1\right) \cdot C_3$$
$$= 6 + 3 + 7 = 16 > L = 15 \quad (NOT\ OK!)$$

Since $7 \geq C_3 \geq 5$, the task set is not EDF schedulable.

c) See lecture notes for Lecture 12 (slide 17).

PROBLEM 7

a) The utilization of the tasks are

	C_i	T_i	u_i
τ_1	2	10	0.2
τ_2	10	25	0.4
τ_3	12	30	0.4
τ_4	5	10	0.5
τ_5	8	20	0.4
τ_6	7	100	0.07

The order of allocation (based in increasing period) is τ_1, τ_4, τ_5, τ_2, τ_3 and τ_6. The three processors are indexed as P_1, P_2, and P_3.

Task τ_1 can be allocated to P_1 since there is no other tasks in P_1.

Task τ_4 can also be allocated to P_1 since $u_1 + u_4 = 0.2 + 0.5 <= 2 \cdot (2^{\frac{1}{2}} - 1) = 0.824$.

Task τ_5 cannot be allocated to P_1 since $u_1 + u_4 + u_5 = 0.2 + 0.5 + 0.4 > 1$. Task τ_5 can be allocated to P_2 since there is no other task in P_2.

Task τ_2 cannot be allocated to P_1 since $u_1 + u_4 + u_2 = 0.2 + 0.5 + 0.4 > 1$. Task τ_4 can be allocated to P_2 since $u_5 + u_2 = 0.4 + 0.4 <= 0.824$.

Task τ_3 cannot be allocated to P_1 since $u_1 + u_4 + u_3 = 0.2 + 0.5 + 0.4 > 1$. Task τ_3 cannot be allocated to P_2 since $u_5 + u_2 + u_3 = 0.4 + 0.4 + 0.4 > 1$. Task τ_3 can be allocated to P_3 since there is no other task in P_3.

Task τ_6 can be allocated to P_1 since $u_1 + u_4 + u_6 = 0.2 + 0.5 + 0.07 = 0.77 <= 3 \cdot (2^{\frac{1}{3}} - 1) = 0.779$.

So, the final allocation is as follows:

P_1 gets τ_1, τ_4, and τ_6.

P_2 gets τ_5 and τ_2.

P_3 gets τ_3.

b) Task τ_2 is removed from the task set. The new task τ_7 has WCET $C_7 = 3$. The smallest possible period T_7 is 3. In such case the utilization is $u_7 = 1$. We have to check if a successful allocation using RMFF exists.

The order of allocation (in order of increasing period) is τ_7, τ_1, τ_4, τ_5, τ_3 and τ_6. The three processors are indexed as P_1, P_2, and P_3.

Task τ_7 can be allocated to P_1 since there is no other tasks in P_1.

Task τ_1 cannot be allocated to P_1 since P_1 is full. Task τ_1 can be allocated to P_2 since there is no other task in P_2.

Task τ_4 cannot be allocated to P_1 since P_1 is full. Task τ_4 can be allocated to P_2 since $u_1 + u_4 = 0.2 + 0.5 <= 2 \cdot (2^{\frac{1}{2}} - 1) = 0.824$.

Task τ_5 cannot be allocated to P_1 since P_1 is full. Task τ_5 cannot be allocated to P_2 since $u_1 + u_4 + u_5 = 0.2 + 0.5 + 0.4 > 1$. Task τ_5 can be allocated to P_3 since there is no other task in P_3.

Task τ_3 cannot be allocated to P_1 since P_1 is full. Task τ_3 cannot be allocated to P_2 since since $u_1 + u_4 + u_3 = 0.2 + 0.5 + 0.4 > 1$. Task τ_3 can be allocated to P_3 since $u_5 + u_3 = 0.4 + 0.4 <= 2 \cdot (2^{\frac{1}{2}} - 1) = 0.824$.

Task τ_6 cannot be allocated to P_1 since P_1 us full. Task τ_6 can be allocated to P_2 since $u_1 + u_4 + u_6 = 0.2 + 0.5 + 0.07 = 0.77 <= 3 \cdot (2^{\frac{1}{3}} - 1) = 0.779$.

So, the final allocation is as follows:

P_1 gets τ_7.

P_2 gets τ_1, τ_4, and τ_6.

P_3 gets τ_5 and τ_3.

Since the allocation is successful, the smallest period T_7 is $T_7 = C_7 = 3$.

c) See lecture notes for Lecture 14 (slide 22).