Real-Time Systems

Solutions to final exam March 16, 2015

PROBLEM 1

a) True: TinyTimber's AFTER() construct allows the programmer to call a method after a delay relative to the calling method's baseline, thereby eliminating any systematic time skew.

b) True: Too large over-estimation of the execution time of a task will leave less slack for the execution of other tasks in the schedulability analysis.

c) False: The utilization guarantee bound for RM-US converges towards 33.3% as the number of processors become very large.

d) False: The critical instant refers to a point in time when the response time of an analyzed task is maximized. In single-processor system the critical instant occurs when the task arrives at the same time as all tasks with higher priority.

e) False: By following certain guidelines of how to request resources it is possible to avoid deadlock.

f) True: For a sufficient feasibility test a positive answer guarantees that the task set is schedulable. Therefore, if the task set is not schedulable the answer from the test must have been a "no".

PROBLEM 2

a) See lecture notes for Lecture 4 (slide 24).

b) See lecture notes for Lecture 4 (slide 26).

c) See lecture notes for Lecture 4 (slide 27).

PROBLEM 3

a) The WCET of main is dependent on the WCET of functions "FuncA" and "FuncB".

WCET of "main":

$$
\begin{aligned}
&WCET(main) = \{Dec, Flag\} + \{Dec, result\} + \{Dec, P\} + \{Dec, Q\} + \{Assign, Q = -4\} \\
&+ \{mod, (FuncA(Q)\%2)\} + \{call, FuncA(Q)\} + WCET(FuncA(-4)) \\
&+ \{Comp, (FuncA(Q)\%2) == 0\} + \{call, FuncA(Q)\} + WCET(FuncA(-4)) + \{sub, FuncA(Q) - 1\} \\
&+ \{Assign, FuncA(Q) - 1\} + \{abs, abs(Q)\} + \{mod, P\%abs(Q))\} + \{Comp, P\%abs(Q)) == 0\} \\
&+ \{abs, abs(Q)\} + \{call, FuncB(P, abs(Q))\} + WCET(FuncB(23, 4)) + \{Assign, FuncB(P, abs(Q))\} \\
&+ \{comp, result! = 1\} + \{assign, Flag = T\} + \{return, result\} \\
&= 1 + 1 + 1 + 1 + 1 + 5 + 2 + 2 + 2 + 3 + 1 + 5 + 5 + 2 + 5 + 2 + 1 + 2 + 1 + 2 + 2 * WCET(FuncA(-4)) \\
&+ WCET(FuncB(23, 4)) = 45 + 2 * WCET(FuncA(-4)) + WCET(FuncB(23, 4))
\end{aligned}
$$

WCET of "FuncA": There are two cases for calculating the WCET of FuncA:
Case(i) x == 1, Case(ii) x!=1.

$Case(i) WCET(FuncA(x)) = \{Comp, x == 1\} + \{return, x\} = 2 + 2 = 4$
$Case(ii) WCET(FuncA(x = -4)) = \{Comp, x == 1\} + \{abs, abs(x)\} + \{abs, abs(x)\} + \{sub, abs(x) - 1\}$
$+ \{call, FuncA(abs(x) - 1)\} + WCET(FuncA(3)) + \{mul, abs(x) * FuncA(abs(x) - 1)\}$
$+ \{return, abs(x) * FuncA(abs(x) - 1)\} = 2 + 5 + 5 + 3 + 2 + 4 + 2 + WCET(FuncA(3))$
$= 23 + WCET(FuncA(3))$
$WCET(FuncA(x = 3)) = \{Comp, x == 1\} + \{abs, abs(x)\} + \{abs, abs(x)\} + \{sub, abs(x) - 1\}$
$+ \{call, FuncA(abs(x) - 1)\} + WCET(FuncA(2)) + \{mul, abs(x) * FuncA(abs(x) - 1)\}$
$+ \{return, abs(x) * FuncA(abs(x) - 1)\} = 23 + WCET(FuncA(2))$

$WCET(FuncA(x = 3)) = \{Comp, x == 1\} + \{abs, abs(x)\} + \{abs, abs(x)\} + \{sub, abs(x) - 1\}$
$+ \{call, FuncA(abs(x) - 1)\} + WCET(FuncA(1)) + \{mul, abs(x) * FuncA(abs(x) - 1)\}$
$+ \{return, abs(x) * FuncA(abs(x) - 1)\} = 23 + WCET(FuncA(1))$

$==>$
$WCET(FuncA(-4)) = 23 + 23 + 23 + 4 = 73$

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

$Case(i) WCET(FuncB(a, b)) = \{Comp, b == 0\} + \{return, a\} = 2 + 2 = 4$
$Case(ii) WCET(FuncB(a = 23, b = 4)) = \{Comp, b == 0\} + \{mod, (a\%b)\} + \{call, FuncB(b, a\%b)\}$
$+ WCET(FuncB(4, 3)) + \{return, FuncB(b, a\%b)\} = 11 + WCET(FuncB(4, 3))$

$WCET(FuncB(a = 4, b = 3)) = \{Comp, b == 0\} + \{mod, (a\%b)\} + \{call, FuncB(b, a\%b)\}$
$+ WCET(FuncB(3, 1)) + \{return, FuncB(b, a\%b)\} = 11 + WCET(FuncB(3, 1))$

$WCET(FuncB(a = 3, b = 1)) = \{Comp, b == 0\} + \{mod, (a\%b)\} + \{call, FuncB(b, a\%b)\}$
$+ WCET(FuncB(1, 0)) + \{return, FuncB(b, a\%b)\} = 11 + WCET(FuncB(1, 0))$

$==>$
$WCET(FuncB(23, 4)) = 11 + 11 + 11 + 4 = 37$

WCET of "main":

$$WCET(main) = 45 + 2 * WCET(FuncA(-4)) + WCET(FuncB(23, 4))$$
$$= 45 + 2 * 73 + 37 = 228$$

The deadline is missed

b) There are eight abs operations. The cost of each abs operation should be 3 μs so that the deadline is met.

PROBLEM 4

a) The two design flaws are the following:

```
TaskObj C = { initObject() }; // separate object for background task

...

void T2(TaskObj *self, int u) {
    ...
    SEND(USEC(200), USEC(50), self, T2, 0); // period of 200 us needed for 2500 Hz tone
}

...

void kickoff(TaskObj *self, int u) {
    SEND(USEC(0), USEC(1300), &C, BG, 0); // separate object needed for background task
    ...                                   // to allow T1 to produce a clean 1000 kHz tone
}

...
```

b) If shared data is stored within an object, TinyTimber guarantees that mutual exclusion applies for the methods defined with the object if called with SYNC or ASYNC.

c) TinyTimber uses the Deadline Inheritance Protocol, combined with deadlock detection via the return value of the SYNC call.

PROBLEM 5

We start by observing that task τ_2 has a first arrival time that differs from that of the other tasks. This means that the use of a utilization-based or response-time-based schedulability test may become overly pessimistic IF there exists no point in time in the schedule where all tasks arrive at the same time. This, in turn, could mean that, should the test fail, the task set could potentially still be schedulable.

Luckily, by observing the given periods and offsets, we can see that there does exist a point in time where all tasks arrive at the same time, namely at $t = 60$ (where the arrivals of the 4th instance of τ_2, the 4th instance of τ_3 and the 3rd instance of τ_1 coincide). We can then use this as the critical instant in our analysis.

Our first candidate method for schedulability analysis is Liu and Layland's classic utilization-based test. For three tasks, the schedulability bound is $U_{lub} = 3(2^{1/3} - 1) \approx 0.780$. Unfortunately, the accumulated task utilization, $U = 4/30 + 4/16 + 9/20 = \approx 0.833$, exceeds the guarantee bound, and the test does not provide any useful information.

We must, consequently, resort to response-time analysis. Since RM is used, the task priorities are determined by the task periods. To that end, task τ_2 has highest priority (shortest period) and process τ_1 has lowest priority.

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

$R_2 = C_2 = 4 < D_2 = 16$.

$R_3 = C_3 + \lceil \frac{R_3}{T_2} \rceil \cdot C_3$. Assume that $R_3^0 = C_3 = 9$:

$R_3^1 = 9 + \lceil \frac{9}{16} \rceil \cdot 4 = 9 + 1 \cdot 4 = 13$

$R_3^2 = 9 + \lceil \frac{13}{16} \rceil \cdot 4 = 9 + 1 \cdot 4 = 13 < D_3 = 20$

$R_1 = C_1 + \lceil \frac{R_1}{T_2} \rceil \cdot C_2 + \lceil \frac{R_1}{T_3} \rceil \cdot C_3$. Assume that $R_1^0 = C_1 = 4$:

$R_1^1 = 4 + \lceil \frac{4}{16} \rceil \cdot 4 + \lceil \frac{4}{20} \rceil \cdot 9 = 4 + 1 \cdot 4 + 1 \cdot 9 = 17$

$R_1^2 = 4 + \lceil \frac{17}{16} \rceil \cdot 4 + \lceil \frac{17}{20} \rceil \cdot 9 = 4 + 2 \cdot 4 + 1 \cdot 9 = 21$

$R_1^3 = 4 + \lceil \frac{21}{16} \rceil \cdot 4 + \lceil \frac{21}{20} \rceil \cdot 9 = 4 + 2 \cdot 4 + 2 \cdot 9 = 30$

$R_1^4 = 4 + \lceil \frac{30}{16} \rceil \cdot 4 + \lceil \frac{30}{20} \rceil \cdot 9 = 4 + 2 \cdot 4 + 2 \cdot 9 = 30 \leq D_1 = 30$

Conclusion: all tasks meet their deadlines!

PROBLEM 6

Since RM is used, the task priorities are determined by the task periods. To that end, with the original task periods, task τ_1 has highest priority (shortest period) and process τ_3 has lowest priority.

a) Our first candidate method for schedulability analysis is Liu and Layland's classic utilization-based test. For three tasks, the schedulability bound is $U_{lub} = 3(2^{1/3} - 1) \approx 0.780$. Unfortunately, the accumulated task utilization, $U = 2/5 + 4/13 + 6/29 = \approx 0.915$, exceeds the guarantee bound, and the test does not provide any useful information.

We therefore calculate the response time of each task and compare it against the corresponding task deadline (= period):

$R_1 = C_1 = 2 < T_1 = 5$.

$R_2 = C_2 + \lceil \frac{R_2}{T_1} \rceil \cdot C_1$. Assume that $R_2^0 = C_2 + C_1 = 4 + 2 = 6$:

$R_2^1 = 4 + \lceil \frac{6}{5} \rceil \cdot 2 = 4 + 2 \cdot 2 = 8$

$R_2^2 = 4 + \lceil \frac{8}{5} \rceil \cdot 2 = 4 + 2 \cdot 2 = 8 < T_2 = 13$

$R_3 = C_3 + \lceil \frac{R_3}{T_2} \rceil \cdot C_2 + \lceil \frac{R_3}{T_1} \rceil \cdot C_1$. Assume that $R_3^0 = C_3 + C_2 + C_1 = 6 + 4 + 2 = 12$:

$R_3^1 = 6 + \lceil \frac{12}{13} \rceil \cdot 4 + \lceil \frac{12}{5} \rceil \cdot 2 = 6 + 1 \cdot 4 + 3 \cdot 2 = 6 + 4 + 6 = 16$

$R_3^2 = 6 + \lceil \frac{16}{13} \rceil \cdot 4 + \lceil \frac{16}{5} \rceil \cdot 2 = 6 + 2 \cdot 4 + 4 \cdot 2 = 6 + 8 + 8 = 22$

$R_3^3 = 6 + \lceil \frac{22}{13} \rceil \cdot 4 + \lceil \frac{22}{5} \rceil \cdot 2 = 6 + 2 \cdot 4 + 5 \cdot 2 = 24$

$R_3^4 = 6 + \lceil \frac{24}{13} \rceil \cdot 4 + \lceil \frac{24}{5} \rceil \cdot 2 = 6 + 2 \cdot 4 + 5 \cdot 2 = 24 \leq T_3 = 29$

Conclusion: all tasks meet their deadlines!

b) An obvious version of the task set that has more appropriate periods (within the given limits) is where $T_2 = 15$ and $T_3 = 30$. Since the original task set is schedulable, and neither the new T_2 nor the new T_3 is shorter than the original period, the new task set must also be schedulable. The length of this (repeatable) schedule is 30 time units, and the start and stop times for the tasks are as follows:

τ_1: 6 instances: (0,2), (5,7), (10,12), (15,17), (20,22) and (25,27)

τ_2: 2 instances: (2,5)(7,8) and (17,20)(22,23)

τ_3: 1 instance: (8,10)(12,15)(23,24)

There is also a version of the task set with $T_2 = 10$ and $T_3 = 30$ that, despite a total task utilization of 100%, is schedulable. The length of this (repeatable) schedule is also 30 time units, but has one more instance of τ_2 and thus less compact.

PROBLEM 7

a) See lecture notes for Lecture 14 (slide 14-17).

b) See lecture notes for Lecture 14 (slide 27-29).

c) First we compute the utilization of the tasks.

	C_i	T_i	U_i
τ_1	160	200	$160/200 = 0.8$
τ_2	10	25	$10/25 = 0.4$
τ_3	10	40	$10/40 = 0.25$
τ_4	5	20	$160/200 = 0.25$
τ_5	10	?	$10/T_5$
τ_6	5	10	$5/10 = 0.5$

Task τ_1, τ_2 and τ_6 must be assigned to different processors. None of these two tasks can be allocated on the same processor because the Liu and Layland bound for uniprocessor rate-monotonic scheduling for two tasks is $2 \cdot (2^{\frac{1}{2}-1}) \approx 0.8284$.

Task τ_3 or task τ_4 cannot be allocated to the processor on which task τ_1 is assigned. Since $U_3 = 0.25$ and $U_4 = 0.25$, allocating τ_3 or τ_4 on a processor where τ_1 is assigned would result in total utilization on that processor larger than 1.

Assigning both τ_3 and τ_4 with task τ_2 on the same processor would result in total utilization on that processor equal to $U_2 + U_3 + U_4 = 0.4 + 0.25 + 0.25 = 0.9$. Assigning both τ_3 and τ_4 with task τ_6 on the same processor would result in total utilization on that processor equal to $U_3 + U_4 + U_6 = 0.25 + 0.25 + 0.5 = 1.0$. Liu and Layland bound for uniprocessor rate-monotonic scheduling for three tasks is $3 \cdot (2^{\frac{1}{3}-1}) \approx 0.7797$. Therefore, both τ_3 and τ_4 cannot be allocated to any of the processors on which task τ_2 or task τ_6 is assigned. Only one of the τ_3 and τ_4 can be allocated to the processor on which τ_2 or τ_6 is assigned.

If the period of task $T_5 \leq 40$, there is no processor on which task τ_5 can be allocated. Therefore, $T_5 > 40$. Therefore, $\tau_2, \tau_3, \tau_4, \tau_6$ are allocated before task τ_5 is allocated (RMFF assigns task with smaller period before assigning task with larger period). Consequently, τ_4 and τ_6 is allocated to the first processor; τ_2 and τ_3 is allocated to the second processors; and, τ_1 is allocated to the third processors.

The period of task τ_5 will be smallest when allocated to the second processor since this processor has lowest utilization. All the tasks τ_2, τ_3, τ_5 will meet their deadlines if $U_2 + U_3 + \frac{10}{T_5} \leq 3 \cdot (2^{\frac{1}{3}} - 1)$. This implies

$$\frac{10}{3 \cdot (2^{\frac{1}{3}} - 1) - U_2 - U_3} \leq T_5$$

$$\text{or, } \frac{10}{3 \cdot (2^{\frac{1}{3}} - 1) - 0.4 - 0.25} \leq T_5$$

(for smallest integer T_5)

$$T_5 = \left\lceil \frac{10}{3 \cdot (2^{\frac{1}{3}} - 1) - 0.4 - 0.25} \right\rceil = \lceil 77.06 \rceil = 78$$