Introduction to Real-Time Systems

Solutions to final exam June 3, 2019 (version 20190603)

PROBLEM 1

a) True: No scheduling algorithm can schedule a task set that requires more than 100% capacity of a single-processor system.

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

c) False: For an NP-complete problem to have pseudo-polynomial time complexity the largest number in the problem cannot be bounded by the input length (size) of the problem.

d) False: The utilization guarantee bound for RMFF is 41.4% of the total processor capacity.

e) False: Interrupts are local to each processor, i.e., only applicable for single-processor system.

f) False: If we know that the task set is not schedulable then a necessary test can either result in either the outcome 'True' or the outcome 'False'. This is because a necessary test can result in the outcome 'True' even though the task set is not schedulable.

PROBLEM 2

a) The four conditions for deadlock is:

- Mutual exclusion – only one task at a time can use a resource
- Hold and wait – there must be tasks that hold one resource at the same time as they request access to another resource
- No preemption – a resource can only be released by the task holding it
- Circular wait – there must exist a cyclic chain of tasks such that each task holds a resource that is requested by another task in the chain

b) The basic idea of a priority ceiling protocol is as follows:

- Each resource is assigned a priority ceiling equal to the priority of the highest-priority task that can lock it.
- A task τ_i is allowed to enter a critical region only if its priority is higher than all priority ceilings of the resources currently locked by tasks other than τ_i.
- When task τ_i blocks one or more higher-priority tasks, it temporarily inherits the highest priority of the blocked tasks.

PROBLEM 3

a) In order to find the WCET of `main`, we need to find the WCET for the `times`, `methA` and `methB` functions.

The WCET of `times(a,b)` is as follows:

$$WCET(times(a,b)) = \{multiply, a*b\} + \{return, a*b\} = 5 + 2 = 7$$

The WCET of `methA(a,b)` is derived based on two cases of the value of parameter b.

Case 1: $b = 0$:

$$\begin{aligned}&WCET(methA(a, b = 0)) =\\ &\{declare, p\} + \{declare, i\} + \{assign, p\} + \{assign, i\} + \{compare, b == 0\} + \{return, 1\}\\ &= 1 + 1 + 1 + 1 + 2 + 2 = 8\end{aligned}$$

Case 2: $b > 0$:

The WCET of `methA` for this case largely depends on the number of times the `while` loop executes.

Let $WCET(whileLoop, b)$ denote the WCET of the `while` loop as a function of parameter b. For the given start value of i the logical condition of the loop will be evaluated b times, which means that the body of the loop will execute $(b - 1)$ times. The value of $WCET(whileLoop, b)$ is thus

$$\begin{aligned}&WCET(whileLoop, b) =\\ &b \cdot \{compare, i < b\}+\\ &(b-1) \cdot [\ \{call, times(p, a)\} + WCET(times(p, a)) + \{assign, p\} + \{add, i + 1\} + \{assign, i\}\ ]\\ &=\ b \cdot 2 + (b - 1) \cdot [\ 2 + 7 + 1 + 3 + 1\ ] = 16 \cdot b - 14\end{aligned}$$

The WCET of `methA` for the case $b > 0$ is then:

$$\begin{aligned}&WCET(methA(a, b > 0)) =\\ &\{declare, p\} + \{declare, i\} + \{assign, p\} + \{assign, i\}+\\ &\{compare, b == 0\} + WCET(whileLoop, b) + \{return, p\}\\ &= 1 + 1 + 1 + 1 + 2 + (16 \cdot b - 14) + 2 = 16 \cdot b - 6\end{aligned}$$

The WCET of `methB(a,b)` is derived based on two cases of the value of parameter b.

Case 1: $b = 1$:

$$\begin{aligned}&WCET(methB(a, b = 1)) =\\ &\{compare, b == 1\} + \{return, a\} = 2 + 2 = 4\end{aligned}$$

Case 2: $b > 0$:

$$\begin{aligned}&WCET(methB(a, b > 1)) =\\ &\{compare, b == 1\}+\\ &\{sub, b-1\} + \{call, methB(a, b-1)\} + WCET(methB(a, b - 1))+\\ &\{call, times(a, methB(a, b-1))\} + WCET(times(a, methB(a, b-1))+\\ &\{return, times(a, methB(a, b-1)\} = 2 + 3 + 2 + WCET(methB(a, b - 1)) + 2 + 7 + 2\\ &= 18 + WCET(methB(a, b - 1))\end{aligned}$$

The WCET of `main()` can now be calculated.

$$\begin{aligned}
&\mathit{WCET}(\mathit{main}()) = \\
&\{\mathit{declare}, \mathit{ans}\} + \{\mathit{declare}, x\} + \{\mathit{declare}, y\} + \{\mathit{assign}, x\} + \{\mathit{assign}, y\}+ \\
&\{\mathit{call}, \mathit{methA}(2,3)\} + \mathit{WCET}(\mathit{methA}(2,3)) + \{\mathit{call}, \mathit{methB}(2,3)\} + \mathit{WCET}(\mathit{methB}(2,3))+ \\
&\{\mathit{compare}, \mathit{methA}(2,3) > \mathit{methB}(2,3)\}+ \\
&\max(\{\mathit{assign}, \mathit{ans}\} + \{\mathit{add}, x+y\} + \{\mathit{assign}, x\}, \{\mathit{assign}, \mathit{ans}\}) + \{\mathit{return}, 1\} \\
&= 1+1+1+1+1+2+ \mathit{WCET}(\mathit{methA}(2,3)) + 2 + \mathit{WCET}(\mathit{methB}(2,3))+ \\
&2 + \max(1+3+1, 1) + 2 \\
&= 13 + \max(5,1) + \mathit{WCET}(\mathit{methA}(2,3)) + \mathit{WCET}(\mathit{methB}(2,3))
\end{aligned}$$

The WCET of `methA(2,3)` is

$$\mathit{WCET}(\mathit{methA}(2,3)) = \{\text{Case } 2 : b > 0\} = 16 \cdot 3 - 6 = 42$$

The WCET of `methB(2,3)` is

$$\begin{aligned}
&\mathit{WCET}(\mathit{methB}(2,3)) = \\
&\{\text{Case } 2 : b > 0\} = 18 + \mathit{WCET}(\mathit{methB}(2,2)) = \\
&\{\text{Case } 2 : b > 0\} = 18 + 18 + \mathit{WCET}(\mathit{methB}(2,1)) = \\
&\{\text{Case } 1 : b = 1\} = 18 + 18 + 4 = 40
\end{aligned}$$

The WCET of `main` is then:

$$\begin{aligned}
&\mathit{WCET}(\mathit{main}()) = \\
&= 13 + \max(5,1) + \mathit{WCET}(\mathit{methA}(2,3)) + \mathit{WCET}(\mathit{methB}(2,3)) = \\
&= 13 + \max(5,1) + 42 + 40 = 95 + \max(5,1) > 90
\end{aligned}$$

The deadline of function `main` is <u>missed</u>, regardless of the outcome of the comparison between the results from the `methA(2,3)` and `methB(2,3)` calls.

b) The two false paths in the program are as follows.

The first false path is in function `methA`, where the case $b = 0$ will never apply. There is only one call to the function, in which case the value of parameter b is 3. Thus, the statement `return 1` is never executed.

The second false path is in function `main`, where the condition `methA(x,y)` $>$ `methB(x,y)` will never become true. This can be seen by observing that `methA(x,y)` and `methB(x,y)` both evaluate the function x^y (although using two different algorithms). Thus, the two statements `ans = 'T'` and `x = x + y` are never executed.

PROBLEM 4

a) The tasks T1, T2 and T3 should normally reside in three separate objects, but since their execution is precedence-constrained a solution where they share one object (A) is also correct. Task BG needs to reside in an object that is separate (object B) from the hard-real-time tasks.

```
void T1(TaskObj *self, int u) {

    Action300(); // Do work for 300 microseconds

    BEFORE(USEC(1200), self, T2, 0); // Keep current baseline
}

void T2(TaskObj *self, int u) {

    Action800(); // Do work for 800 microseconds

    BEFORE(USEC(2100), self, T3, 0); // Keep current baseline
}

void T3(TaskObj *self, int u) {

    Action500(); // Do work for 500 microseconds

    SEND(USEC(2400), USEC(1600), self, T1, 0);
}

void BG(TaskObj *self, int u) {

    Load700(); // Do background work for 700 microseconds

    SEND(USEC(1800), USEC(1800), self, BG, 0);
}

void kickoff(TaskObj *self, int u) {

    BEFORE(USEC(1600), &A, T1, 0);
    BEFORE(USEC(1800), &B, BG, 0);
}

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

b) By simulating the execution of the tasks, assuming the EDF scheduler in TinyTimber, we can see that task T3 will miss its deadline at t=2100 μs (having an additional 200 μs to execute).

Conclusion: It is not possible to guarantee that task T3 will meet its deadline.

PROBLEM 5

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 utilization-based test. For three tasks, the guarantee bound for RM is $\text{U}_{\text{RM(3)}} = 3(2^{1/3} - 1) \approx 0.780$. Unfortunately, the accumulated task utilization, $\text{U}_{\text{total}} = 2/5 + 4/13 + 6/29 \approx 0.915$, 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 $R_2^0 = C_2 = 4$:

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

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

$R_2^3 = 4 + \lceil \frac{8}{5} \rceil \cdot 2 = 4 + 2 \cdot 2 = 8 < T_2 = 13$

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

$R_3^1 = 6 + \lceil \frac{6}{13} \rceil \cdot 4 + \lceil \frac{6}{5} \rceil \cdot 2 = 6 + 1 \cdot 4 + 2 \cdot 2 = 6 + 4 + 4 = 14$

$R_3^2 = 6 + \lceil \frac{14}{13} \rceil \cdot 4 + \lceil \frac{14}{5} \rceil \cdot 2 = 6 + 2 \cdot 4 + 3 \cdot 2 = 6 + 8 + 6 = 20$

$R_3^3 = 6 + \lceil \frac{20}{13} \rceil \cdot 4 + \lceil \frac{20}{5} \rceil \cdot 2 = 6 + 2 \cdot 4 + 4 \cdot 2 = 6 + 8 + 8 = 22$

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

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

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. If we generate the time table by simulating an RM scheduler we get the following start and stop times for the tasks:

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 RM 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 6

We apply processor-demand analysis to determine the minimum value of D_2. The hyper-period for the given task set is $\text{LCM}\{10, 20, 40\} = 40$.

Since $C_2 = 2$, we must have $D_2 \geq 2$. If $D_2 <= 5$, then within $L = 5$ time units the first instances of both tasks τ_1 and τ_2 must complete a total of $(2+4) = 6$ units of execution since their deadlines will be within (0, 5]. However, completing 6 units of execution within $[0, 5]$ is not possible. Therefore, we must have $D_2 \geq 6$. We try with $D_2 = 6$.

If $D_2 = 6$, then the set of control points are $K = \{5, 6, 15, 25, 26, 35\}$.

Consider $L = 5$.

$N_1^L \cdot C_1 = (\lfloor \frac{5-5}{10} \rfloor + 1) \cdot C_1 = 4 \qquad N_2^L \cdot C_2 = (\lfloor \frac{5-6}{20} \rfloor + 1) \cdot C_2 = 0$

$N_3^L \cdot C_3 = (\lfloor \frac{5-25}{40} \rfloor + 1) \cdot C_3 = 0$

$C_P(0, L) = C_P(0, 5) = 4 + 0 + 0 = 4 \leq L = 5.$

Consider $L = 6$.

$N_1^L \cdot C_1 = (\lfloor \frac{6-5}{10} \rfloor + 1) \cdot C_1 = 4 \qquad N_2^L \cdot C_2 = (\lfloor \frac{6-6}{20} \rfloor + 1) \cdot C_2 = 2$

$N_3^L \cdot C_3 = (\lfloor \frac{6-25}{40} \rfloor + 1) \cdot C_3 = 0$

$C_P(0, L) = C_P(0, 6) = 4 + 2 + 0 = 6 \leq L = 6.$

Consider $L = 15$.

$N_1^L \cdot C_1 = (\lfloor \frac{15-5}{10} \rfloor + 1) \cdot C_1 = 8 \qquad N_2^L \cdot C_2 = (\lfloor \frac{15-6}{20} \rfloor + 1) \cdot C_2 = 2$

$N_3^L \cdot C_3 = (\lfloor \frac{15-25}{40} \rfloor + 1) \cdot C_3 = 0$

$C_P(0, L) = C_P(0, 15) = 8 + 2 + 0 = 10 \leq L = 15.$

Consider $L = 25$.

$N_1^L \cdot C_1 = (\lfloor \frac{25-5}{10} \rfloor + 1) \cdot C_1 = 12 \qquad N_2^L \cdot C_2 = (\lfloor \frac{25-6}{20} \rfloor + 1) \cdot C_2 = 2$

$N_3^L \cdot C_3 = (\lfloor \frac{25-25}{40} \rfloor + 1) \cdot C_3 = 4$

$C_P(0, L) = C_P(0, 25) = 12 + 2 + 4 = 18 \leq L = 25.$

Consider $L = 26$.

$N_1^L \cdot C_1 = (\lfloor \frac{26-5}{10} \rfloor + 1) \cdot C_1 = 12 \qquad N_2^L \cdot C_2 = (\lfloor \frac{26-6}{20} \rfloor + 1) \cdot C_2 = 4$

$N_3^L \cdot C_3 = (\lfloor \frac{26-25}{40} \rfloor + 1) \cdot C_3 = 4$

$C_P(0, L) = C_P(0, 26) = 12 + 4 + 4 = 20 \leq L = 26.$

Consider $L = 35$.

$N_1^L \cdot C_1 = (\lfloor \frac{35-5}{10} \rfloor + 1) \cdot C_1 = 16 \qquad N_2^L \cdot C_2 = (\lfloor \frac{35-6}{20} \rfloor + 1) \cdot C_2 = 4$

$N_3^L \cdot C_3 = (\lfloor \frac{35-25}{40} \rfloor + 1) \cdot C_3 = 4$

$C_P(0, L) = C_P(0, 35) = 16 + 4 + 4 = 24 \leq L = 35.$

The minimum value of D_2 is 6.

PROBLEM 7

a) See the lecture slides on multiprocessor scheduling.

b) See the lecture slides on multiprocessor scheduling.

c) The total utilization of the task set is $U = U_1 + U_2 + \ldots U_7 + U_8 = 2.0$.

	C_i	T_i	U_i
τ_1	2	8	0.25
τ_2	2	5	0.40
τ_3	1	10	0.10
τ_4	3	120	0.025
τ_5	15	30	0.5
τ_6	60	600	0.1
τ_7	100	200	0.5
τ_8	2	16	0.125

We find the minimum number of processors required based on the guarantee bound $m^2/(3m-2)$ for RM-US scheduling. Since $U = 2.0$, it is necessary that

$$\begin{aligned} m^2/(3m-2) &\geq 2.0 \\ \text{or,}\ \ m^2 - 6m + 4 &\geq 0 \end{aligned}$$

in order to guarantee that all the tasks are schedulable using RM-US global scheduling. By solving the quadratic equation $m^2 - 6m + 4 = 0$, we have $m = \frac{6 \pm \sqrt{36-16}}{2} = 3 \pm \sqrt{5}$. Since the number of processors $m \geq 1$, we have $m \geq 3 + \sqrt{5} = 5.23$. Since the number of processors m must be an integer, we need at least 6 processors.

d) We denote $\tau_a \succ \tau_b$ to say that task τ_a has higher priority than τ_b.

The threshold utilization for RM-US global scheduling (for $m = 6$) is

$$m/(3m-2) = 6/16 = 0.375$$

Since each of the tasks τ_1, τ_3, τ_4, τ_6 and τ_8 has utilization smaller than the threshold utilization 0.375, these five tasks will get lower priority than the remaining three tasks. The relative priority ordering of these five (lower-priority) tasks is governed by the RM priority. Therefore, $\tau_1 \succ \tau_3 \succ \tau_8 \succ \tau_4 \succ \tau_6$ because $T_1 < T_3 < T_8 < T_4 < T_6$.

Since each of the other three tasks τ_2, τ_5 and τ_7 has utilization larger than the threshold utilization 0.375, these three tasks will get the highest priority and their relative priority ordering is arbitrary. Examples of such arbitrary ordering is $\tau_5 \succ \tau_2 \succ \tau_7$, or $\tau_2 \succ \tau_5 \succ \tau_7$ or, $\tau_7 \succ \tau_5 \succ \tau_2$ (there are six such arbitrary orderings, since $3! = 6$).

Therefore, three possible priority ordering of all the tasks are

$$\tau_5 \succ \tau_2 \succ \tau_7 \succ \tau_1 \succ \tau_3 \succ \tau_8 \succ \tau_4 \succ \tau_6$$

or,

$$\tau_2 \succ \tau_5 \succ \tau_7 \succ \tau_1 \succ \tau_3 \succ \tau_8 \succ \tau_4 \succ \tau_6$$

or,

$$\tau_7 \succ \tau_5 \succ \tau_2 \succ \tau_1 \succ \tau_3 \succ \tau_8 \succ \tau_4 \succ \tau_6$$