CSE 237A – Winter 2018 – Homework 2
For Problem 1 and 2, you are expected to do some research beyond the given lecture materials.

Problems 1 & 2

Answers vary

Some information for Problem 1:

- ARM BUS: Parallel, I2C & PCI-E: Serial

Some information for Problem 2:

1. Linux: The current Linux kernel supports multiple schedulers for processors, e.g., FIFO and Round Robin (RR). The most popular choice (default scheduler) is CFS (Completely Fair Scheduler). For the IO, it also uses other scheduling policies e.g., CFQ (Complete fair queue). However, since it is designed for general-purpose computing devices, real-time schedulers are not supported by default.

2. RTLinux: This supports real-time scheduling policies with priority, e.g., EDF and RM. Such schedulers in this OS also support preemption.

3. FreeRTOS: The baseline OS offers some useful features & APIs to design any custom scheduler along with RR scheduler. There are many variants of FreeRTOS, and some of them also support popular schedulers such as EDF and RM covered in the class.
Problem 3 [10 pts]
Consider the following list of operations.

\[
\begin{align*}
o_1 &= i_1 + i_2 \\
o_2 &= o_1 * i_3 \\
o_3 &= i_4 + i_5 \\
o_4 &= o_2 + o_3 \\
o_5 &= i_6 * i_7 \\
o_6 &= i_8 + i_9 \\
o_7 &= o_5 + o_6
\end{align*}
\]

The variables starting with “i” are inputs and the ones starting with “o” are the outputs. That is, there are 9 inputs and 5 outputs. The operations are computed by adders and multipliers.
- Multipliers and adders can operate on 2 operands at once.
- An adder consumes 0.4 \(W\), while a multiplier consumes 1 \(W\).
- Power consumption of any idle unit is 0.1 \(W\).
- A multiplier takes 2 cycles, whereas an adder takes 1 cycle to produce the output.
- One cycle takes 1 \(ns\).

Use this notation style
(Note that this example does not correspond to the current problem)

*Example:*
\[
\begin{align*}
o_1 &= i_1 * i_2 \\
o_2 &= i_3 * i_4
\end{align*}
\]
a) Give the ASAP schedule assuming no resource constraints, (i.e., you can use as many adders and multipliers as you need at a time.)

b) Give the ALAP schedule assuming no resource constraints
c) Give the List schedule assuming that you have only 1 adder and 1 multiplier. If multiple units have the same priority, use numerical order of the outputs to break the tie. e.g., if two units computing o1 and o2 have the same priority, run task o1 first.

```
  i1   i2   i3   i4   i5   i6   i7   i8   i9
  +    +    *    +    +    *    +    +    +
  o1   o2   o3   o4   o5   o6   o7
```

- What is the total energy consumption according to the LIST schedule you found in part c?
  \[ \text{Total energy (\text{nJ})} = 6.1 \]
  \[ = (1.0 + 0.4) \times 4 + (0.4 + 0.1) \times 1 \]

- What is the minimum power consumption?
  \[ \text{Maximum power (\text{W})} = 0.5 \]
  Happens at the last cycle.
Problem 4 [10 pt]
Consider that a single processor runs 3 periodic tasks which are defined in the table below.

<table>
<thead>
<tr>
<th>Task</th>
<th>Worst case execution time (WCET)</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>x</td>
</tr>
</tbody>
</table>

a. In EDF (Earliest Deadline First) scheduling, what is the minimum period of task C, x, to ensure all the tasks can be scheduled?

\[ x = \text{10} \]

\[ \because \frac{1}{5} + \frac{2}{4} + \frac{3}{x} \leq 1 \]

- Note: this gives “x>=10” not “x<=10”.

b. Using x found in part a, fill tasks scheduled by EDF in the table below. Assume the following:
   - All tasks arrive at the time t = 0.
   - If multiple tasks have the same priority, use alphabetic order to break the tie
     e.g., if task A and task B have the same priority, run task A first

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>
Problem 5 [10 pt]
You are given five tasks (T1 – T5) and one each of five different hardware implementations: HW1, HW2, HW3, HW4, and a general-purpose processor P (each costing $10, $15, $5, $25, and $30 respectively). The table below shows the number of seconds it takes to run each task on one of the possible hardware implementations. Each hardware element can only run one task at a time, but different hardware elements may work in parallel. You may not use multiple elements of the same hardware (e.g., you can use only one HW1 for the entire scheduling.) The task graph has a deadline of 60 seconds.

<table>
<thead>
<tr>
<th>T</th>
<th>HW1</th>
<th>HW2</th>
<th>HW3</th>
<th>HW4</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>-</td>
<td></td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>15</td>
<td>-</td>
<td>10</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>30</td>
<td>-</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>25</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>25</td>
</tr>
</tbody>
</table>

For each of the following schedules, provide a feasible partitioning of tasks among hardware elements, the total system cost, and the total execution time of all tasks for the task graph:

a) Minimum execution time schedule.
   Hardware partitioning: T1: HW1, T2: HW2, T3: HW4, T4: HW4, T5: HW2
   Execution time = 15 + max(15, 15) + max(15, 10) = 45
   Cost = 10 + 15 + 25 = $ 50

b) Minimum cost schedule. Note that the minimum cost schedule may miss the deadline.
   Both answers are right:
   i) Hardware partitioning: T1: HW1, T2: HW1, T3: HW2, T4: HW3, T5: HW2
      Execution time = 15 + max(max(20 + 25), max(20, 30) + 10) = 60
      Cost = 10 + 15 + 5 = $ 30
   ii) Hardware partitioning: T1: HW1, T2: HW1, T3: HW2, T4: HW3, T5: HW1
      Execution time = 15 + max(max(20 + 25), max(20, 30) + 30) = 75
      Cost = 10 + 15 + 5 = $ 30

c) Cheapest schedule that meets the deadline of the graph.
   (The answer b-i) meets the deadline. So it’s the answer for this subproblem.)
   i) Hardware partitioning: T1: HW1, T2: HW1, T3: HW2, T4: HW3, T5: HW2
      Execution time = 15 + max(max(20 + 25), max(20, 30) + 10) = 60
      Cost = 10 + 15 + 5 = $ 30