Assignment 2

ADVANCE OPERATING SYSTEMS, FALL 2016

Time: One weeks

Instructions: Do not copy material from other sources, if it is necessary, then provide the references. In case of plagiarism, negative marks will be given to the plagiarizing person.

*Rules for assignment grading: The time for all assignments for the course of AOS will be 1 week. If there is at least one assignment submitted in time, then the time will exceed to 2 weeks. Of course, the late assignments will never have the full grade. If there is no assignment submission in one week, then there will be zero for all the class in that assignment. The grade A will only be given to a person who will submit **all** the assignments (no matter, in first week or second). The grade B will not be given to a person who has less than **75%** of submissions.*

You have a hardware with **two cores**, and it has to serve only the real time jobs. You are given the information about the arriving tasks, and you have to generate a schedule out of it. Following table is a sample table, you will copy the same table into a file named `inputAssign-2.txt` and use it as a sample input file. The file should be represented in comma separated fashion. The output file will be named `outputAssign-2.txt`, which will contain the generated schedule. Ofcourse, the code should run if the table values are changed or new rows are added.

The fields in **table 1** are self explanatory except for second field, named **Threads**. This tells you that how many threads this process wants to run on your machine. The threads share the same starting and ending deadlines and execution times. A process cannot have more than **5** threads. You can use the **Earliest deadline first** algorithm to generate the schedule. If it is impossible to meet all deadlines, then the algorithm should try to meet maximum number of deadlines. The output file will contain a comma separated table as shown in **table 2**. Here 1.2 shows the 2^{nd} thread of the 1^{st} process.

Addendum:

Looking more closely at the solution in **table 2** it reveals that the solution is not unique. The solution depends upon how do you prepare the solution. In the given example it is assumed that all the deadlines are first converted into **Ending deadlines**, by adding execution time into the starting deadlines. In this case the **table 2** is correct. If you convert all deadlines into **Starting deadlines**, by subtracting execution times from ending deadlines, then the results will be slightly different. Also note that in **table 2** the deadline for **Process 7** is missed.

How to read the table 2:

It should be understood that a row in table is added as soon as a process finishes, or starts. It is obvious that in many cases two processes will stop and two will start simultaneously on both processors. In that case instead of adding two lines, one for stop and one for start, only one line is added, and that is for stopping of the processes. This is why although, **3.1** and **3.2** have started on cycle number 30, but the context switch time is written when they both have finished. For ease you could convert **table 2** into **table 3**. In **table 3** the context switch time is written twice, once for the processes which stopped, and second for the processes which started.

Table 1: Input file

Process	Threads	Arrival Time	Execution Time	Starting Deadline	Ending Deadline
1	2	0	15	NA	50
2	1	10	20	10	NA
3	4	20	10	60	NA
4	1	30	50	NA	200
5	2	30	30	50	NA
6	4	40	10	NA	100
7	5	50	10	90	NA

Table 2: Output file

CPU 1 Executing Process	CPU 2 Executing Process	Context Switch Time
1.1	idle	0
1.1	2	10
1.2	2	15
1.2	2	30
3.1	3.2	40
3.3	3.4	50
5.1	5.2	80
6.1	6.2	90
6.3	6.4	100
7.1	7.2	110
7.3	7.4	120
7.5	4	130
idle	4	140
idle	idle	170

Table 3: Output file

CPU 1 Executing Process	CPU 2 Executing Process	Context Switch Time
1.1	idle	0
1.1	2	10
1.1	2	15
1.2	2	15
1.2	2	30
3.1	3.2	30
3.1	3.2	40
3.3	3.4	40
3.3	3.4	50
5.1	5.2	50
5.1	5.2	80
6.1	6.2	80
6.1	6.2	90
6.3	6.4	90
6.3	6.4	100
7.1	7.2	100
7.1	7.2	110
7.3	7.4	110
7.3	7.4	120
7.5	4	120
idle	4	130
idle	idle	170