Subject: Assignment 01 Date:

Parallel + Distributed Computing

Monima Atif Dar
P18-0030

Code Snippet: (For CPU1 only)

```
for (int k=0; k<3; k++) {
    response.clear()
    wait.clear();
    CPU1.clear();
    for (int i=0; i<ProcQueue.size(); i++) {
        CPU1.loadProcess(ProcQueue.get(i), 1); ← Dispatcher
    }
    CPU1.runProcessRR(HQ);
    CPU1.runProcessRR(LQ);
    CPU1.runProcessHPN();
    CPU1.runProcessFCFS();
    avgs.clear();
    stats.clear();
    for (int i=0; i<CPU1.ProcessAL.size(); i++ ){
        stats.add(CPU1.stats.get(i));
    }
    avgs = StatsGen(stats, responseTime, contentSwitches, turnAround, waitTi
    responseTime = avgs.get(0);
    contentswitches = avgs.get(1);
    turnAround = avgs.get(2);
    waitTime = avgs.get(3);
```

Subject:

Date:

```
graph.start(response, "1 CPU response");
graph.start(content, "1 CPU content switches");
graph.start(turnAround, "1 CPU turnaround");
graph.start(wait, "1 CPU wait");
:
```

Explanation:

Dispatcher is a component of scheduler which takes process to CPU. In this example, 100 processes are divided among CPU1, CPU2, CPU3 and CPU4 (processors). In first three graphs only CPU1 is working with 100 processes. In next three graphs, CPU1 and CPU2 are working with 50 processes each. Graphs show CPU response, CPU turnAround and CPU wait. Basically, this example calculates average response, turnAround and wait times for all CPUs. Algorithms that are used by Scheduler are Round Robin and First Come First Serve. Graphs are plotted on the basis of response, turnAround and wait averages.

Code of dispatcher: `CPU1.loadProcess(ProcQueue.get(i),1)`

(Graphs, console output, whole code screenshots are in the document).