Lab1
Group 8
Cheng Chen
Zejian Zhong

CPU Scheduling

Introduction:

In this lab, we use such a C program that emulates different CPU scheduling policies to measure their turnaround time, response time, CPU busy time, throughput and waiting time. Then, we can use the data to compare their differences and have a better understanding of them. There are three different CPU scheduling policies: FCFS, SJF and Round Robin with different quantums.

FCFS is First Come First Served, which is simple because the process that arrives firstly will be allocated CPU firstly, and it never stop until it finishes it CPU burst time. SJF is a policy that needs to check the length of the next CPU burst time, then use these lengths from all processes in ready queue and schedule the process with the shortest time. Compared with FCFS, SJF provides shortest job and the optimal turnaround time. However, SJF could not solve the problem when remaining time is unknown and starvation problem. We need another CPU scheduling that make shorter jobs service faster and avoid starvation without knowledge of remaining of CPU time needed, that is Round Robin. Round Robin order processes in ready queue in FCFS, assign CPU to the head of the list and CPU is also assigned for a limited amount of time, quantum. We execute our program and collect for each policy the average turnaround time (TAT), the average response time (RT), the CPU Busy time (%) (CBT), the throughput (T) and the average waiting time (AWT). For round robin, we collect them for quantum 1ms, 5ms, 10ms, 15ms, 20ms, 25ms, 50ms.

Workflow:

Processes are added to Job queue through NewJobIn() function, and LongtermScheduler() function decides whether there is enough memory and multi-programming limit so that decides which processes could be moved to ready queue. The

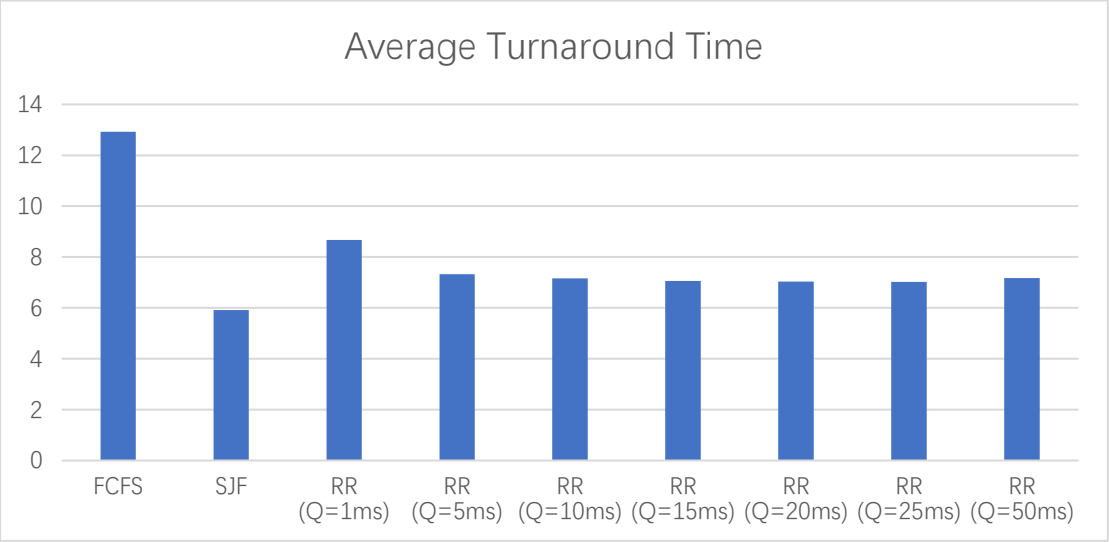
CPUScheduler() function moves job from ready queue to running queue according to Policy Number. If CPU burst done, IO() function moves process on CPU to waiting queue, for RR it will return processes to ready queue. Process with complete IO will be moved to ready queue. Processes running queue needs computation will be put on CPU through Dispatcher(). If it does not need computation, it will be moved to exit queue through Dispatcher. The program will automatically stop after generated 250 processes, then it will print out the result data.

Data:

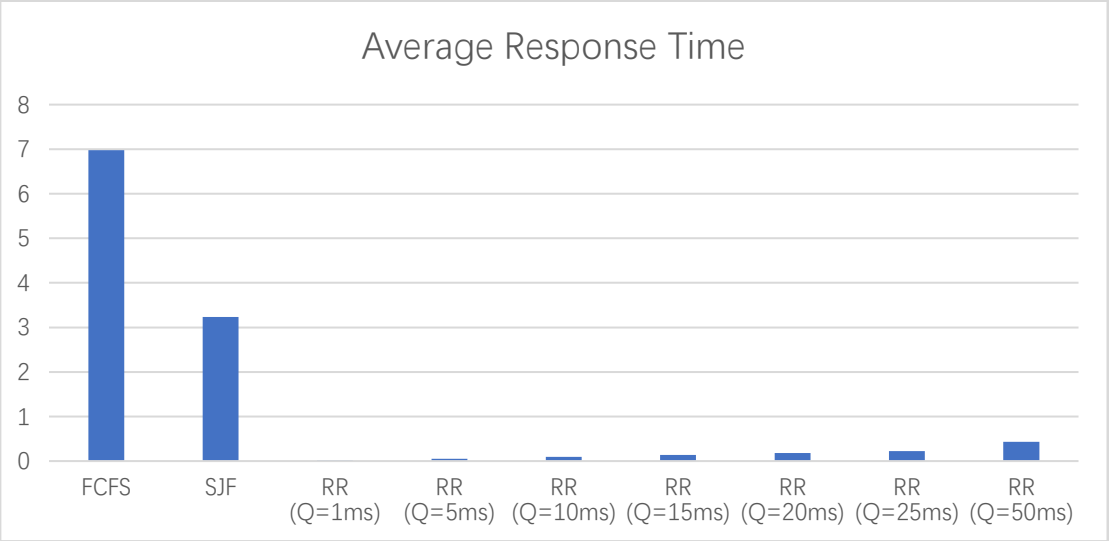
Below is the data representation of FCFS, SJF and RR with different quantum.

Policy	Policy Number	TAT	RT	CBT	T	AWT
FCFS	1	12.930555	6.979344	0.892121	0.896394	13.523179
SJF	2	5.918121	3.233726	0.901845	0.913460	12.258798
RR(Q=1ms)	3	8.666103	0.014058	0.859346	0.927531	11.767030
RR(Q=5ms)	3	7.329445	0.049590	0.893354	0.961993	8.790906
RR(Q=10ms)	3	7.158800	0.093592	0.898693	0.970590	8.409499
RR(Q=15ms)	3	7.056052	0.136939	0.901180	0.970621	8.303577
RR(Q=20ms)	3	7.031887	0.181038	0.902184	0.970634	8.274830
RR(Q=25ms)	3	7.021255	0.220823	0.902970	0.970575	8.259962
RR(Q=50ms)	3	7.173001	0.431890	0.904085	0.970498	8.387153

Analysis:

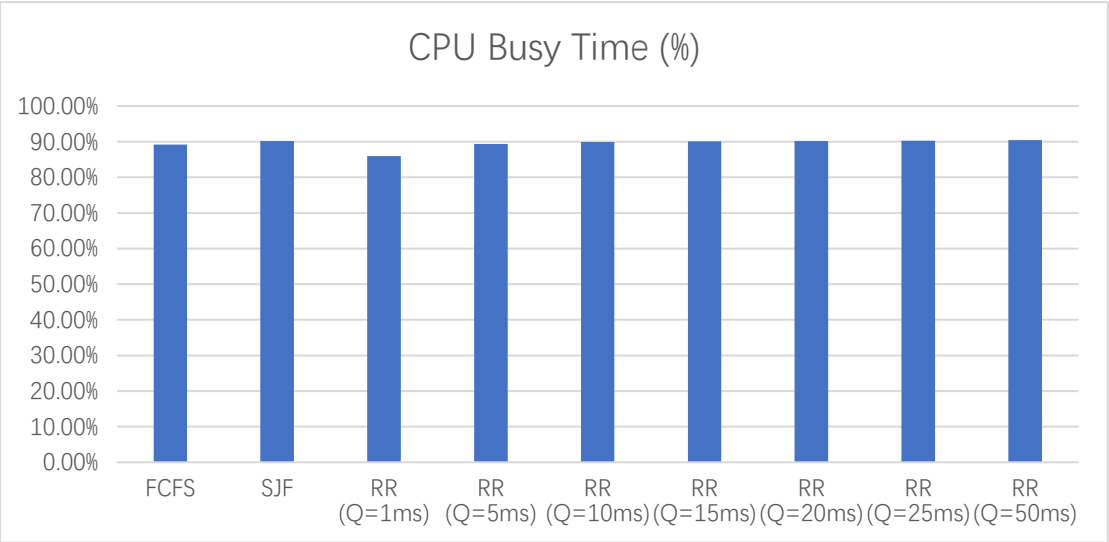


From this Average Turnaround Time table, we can see that FSFC has the longest turnaround time, followed by RR with different quantum, and the SJF has the best turnaround time in average. This result is expected because SJF chooses processes with shorter CPU burst time to run in the next burst period. RR with smaller quantums may result in having longer turnaround time because it spends too much time on context switching.

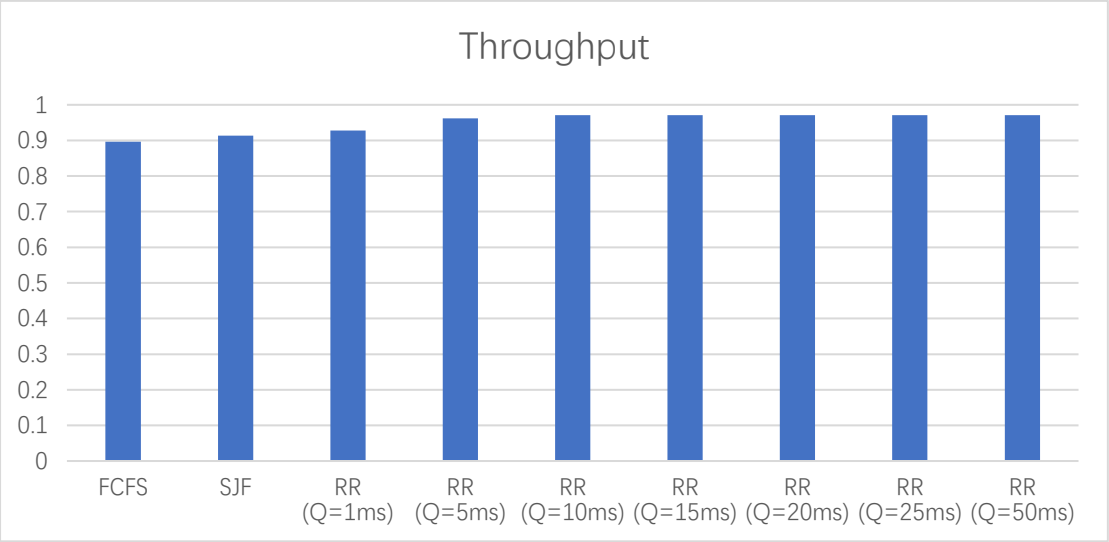


From this Average Response Time table, we can see that FCFS has the longest response time and Round Robin has the best response time with the smallest quantum. This result is expected because Round Robin forced processes to do context switch after each quantum. Thus, smaller quantum will create a more responsive scheduling. As the quantum goes up,

RR will eventually degrade to FCFS.

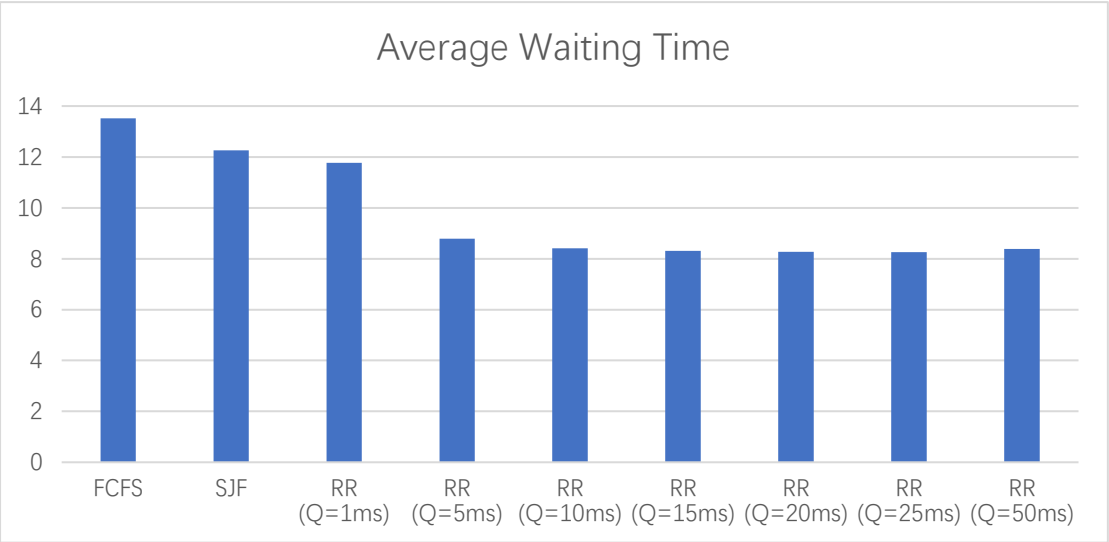


From this CPU Busy Time table, we can see that all tested scheduling policies have similar CPU busy time. This is expected because CPU busy time is measured as how much CPU is being used in a time unit. Since all three policies were running under the same environment and all of them were able to complete their runs, the CPU busy time is similar.



From this Throughput table, we can see that the different of these three CPU scheduling policies is not obvious. The reason may be that our program is only asked to measure the throughput after generating a certain amount of jobs rather than completing them. So, the throughput data in this laboratory assignment is not very useful for learning the differences

among the three policies.



In this Average Waiting Time table, we can see that FCFS has the worst waiting time because it is non-preemptive, all processes behind must wait previous one to finish. SJF should have a better waiting time compare to FCFS since it chooses shorter jobs to run in the next. In general, RR performs better during this test.

Summary:

According to our code and data, we find that CPU busy time (CBT) and throughput (T) are similar for FCFS, SJF and RR with different quantums. FCFS has the worst average turnaround time (TAT), average response time (RT) and average waiting time (AWT). SJF has the best average turnaround time, good average response time and good average waiting time. For RR, the average turnaround time is a little worse when quantum is small but becomes better as quantum going up. The average response time for RR is quite short and increases as quantum increasing. The average waiting time for RR with different quantum remains at a similar value.