Zachary Nawrocki (nawroz), Bingsheng Yao (yaob), and Salman Aldughaither (aldugs)

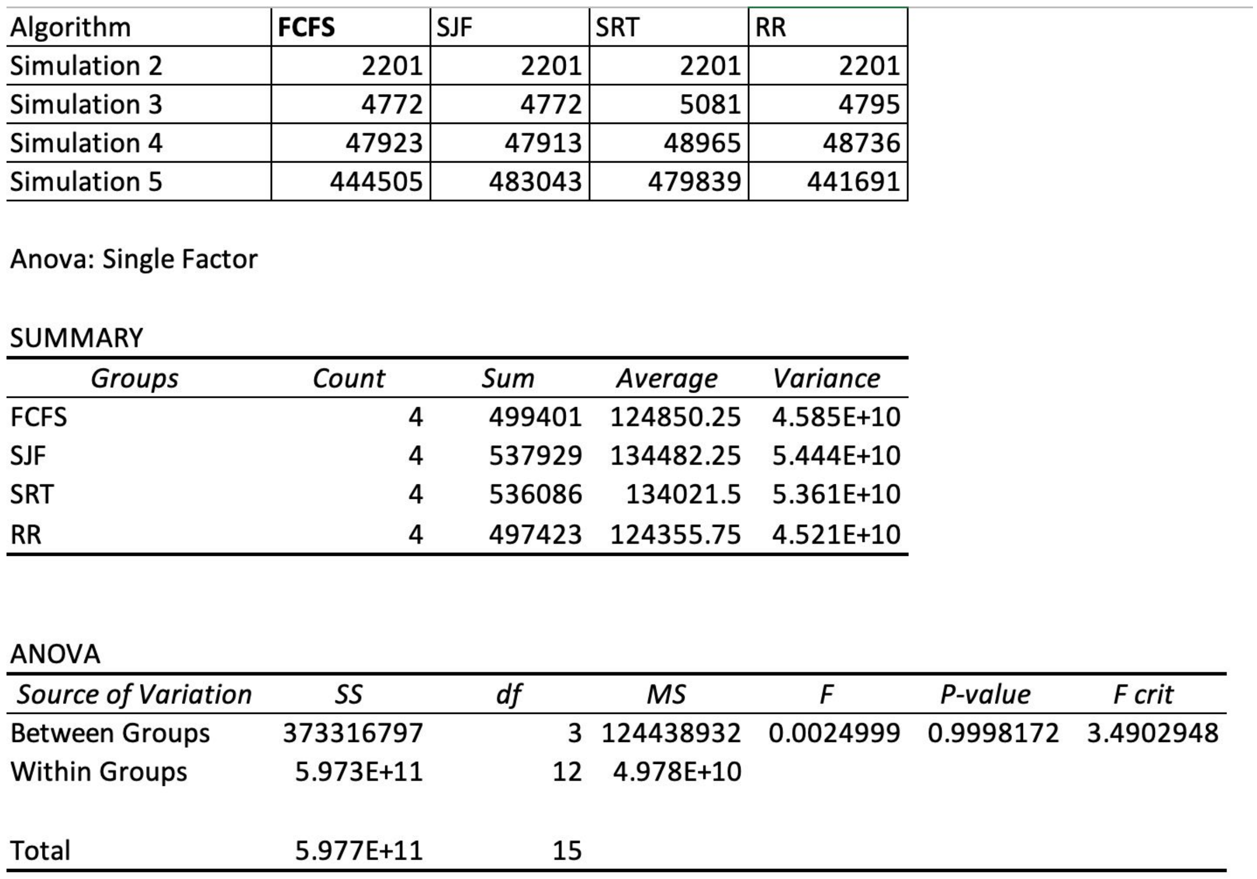
Project Analysis

1. **Of the four simulated algorithms, which algorithm is the "best" algorithm? Which algo- rithm is best-suited for CPU-bound processes? Which algorithm is best-suited for I/O-bound processes? Support your answer by citing specific simulation results.**

We attempted to determine the "best" algorithm by comparing the time it took all processes to complete for each scheduling algorithm. Our first observation was that the variance between the timings of each algorithm in a given simulation was relatively small. Therefore, we performed an analysis to determine if the difference in the variations of the duration of the algorithms were statistically significant. For this, we used an ANOVA: Single Factor test. The results indicated that no statistical significance was found.

Our conclusion, based on these results, is that the different simulations either were not long enough or otherwise did not include enough examples of situations that highlighted the benefits/drawbacks of the different algorithms.

On review of the simulation outputs, it seems that simulation 5 (really the fourth simulation) is the most IO-intensive, where 48.661% of the burst time is spent in IO bursts. By contrast, simulation 2 (really the first simulation) is the most CPU-intensive with only 39.586% of the burst time spent on IO. That said, as seen in Figure 1, the differences are relatively small and it's hard to draw any conclusions as to which algorithms might be best for specific use cases, due to the previously stated lack of statistical significance determined.



Algorithm	**FCFS**	SJF	SRT	RR
Simulation 2	2201	2201	2201	2201
Simulation 3	4772	4772	5081	4795
Simulation 4	47923	47913	48965	48736
Simulation 5	444505	483043	479839	441691

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
FCFS	4	499401	124850.25	4.585E+10
SJF	4	537929	134482.25	5.444E+10
SRT	4	536086	134021.5	5.361E+10
RR	4	497423	124355.75	4.521E+10

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	373316797	3	124438932	0.0024999	0.9998172	3.4902948
Within Groups	5.973E+11	12	4.978E+10			
Total	5.977E+11	15				

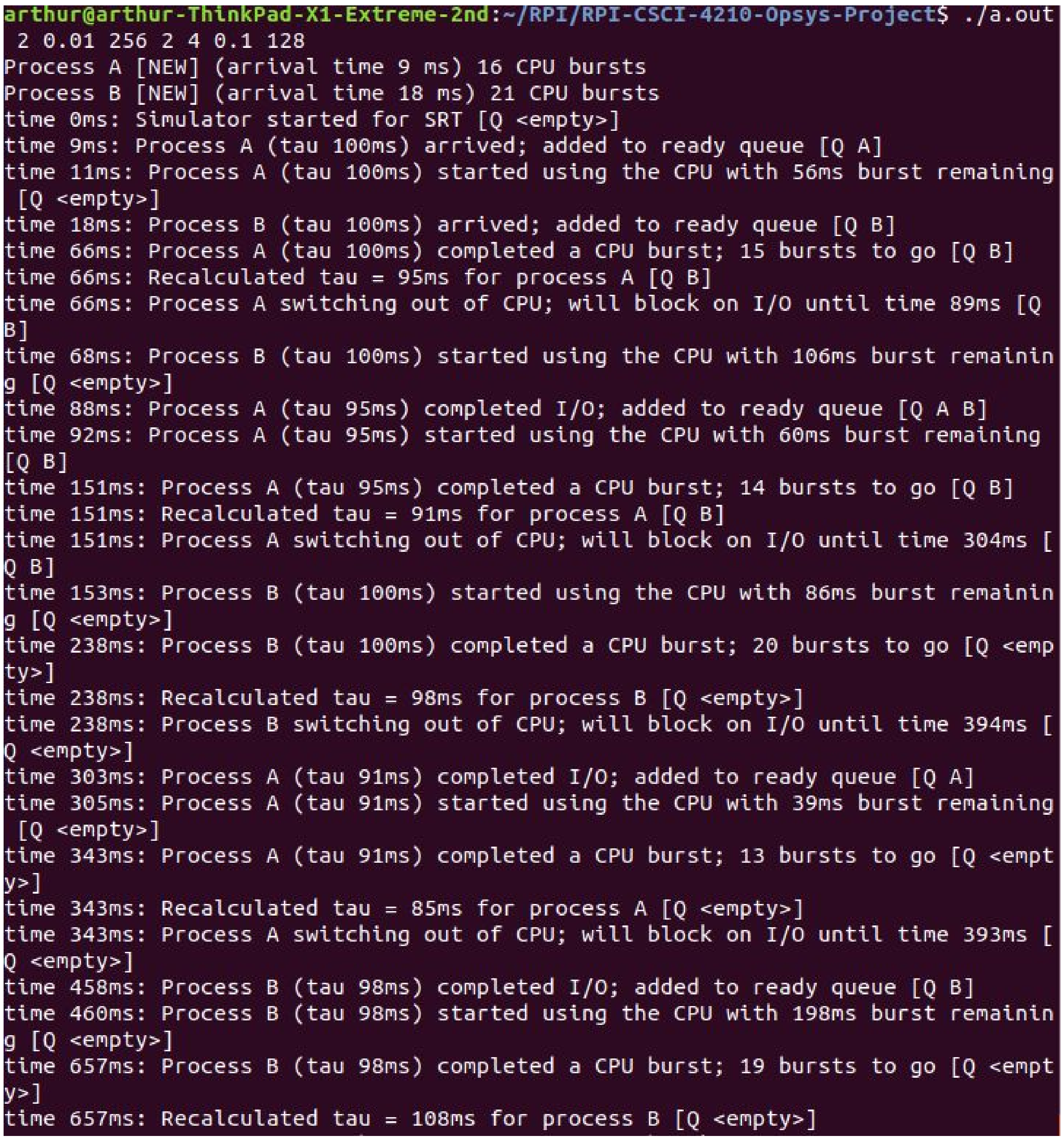
Figure 1

2. **For the RR algorithm, how does changing rr_add from END to BEGINNING impact your results? Which approach is better?**

   By changing rr_add from END (the default) to BEGINNING, we determined that all the timings improved and, on average, the simulations took half as long. This is because the first thing in the queue is always in the middle of running or context switching. This did not apply, however, to tests 2 and 3 (the first and second simulations), where test 2 showed no differences (expected because there was only one process and the queue was therefore never used) and only minimal improvements were seen on test 3. As a result, rr_add with BEGINNING is certainly a much better approach.

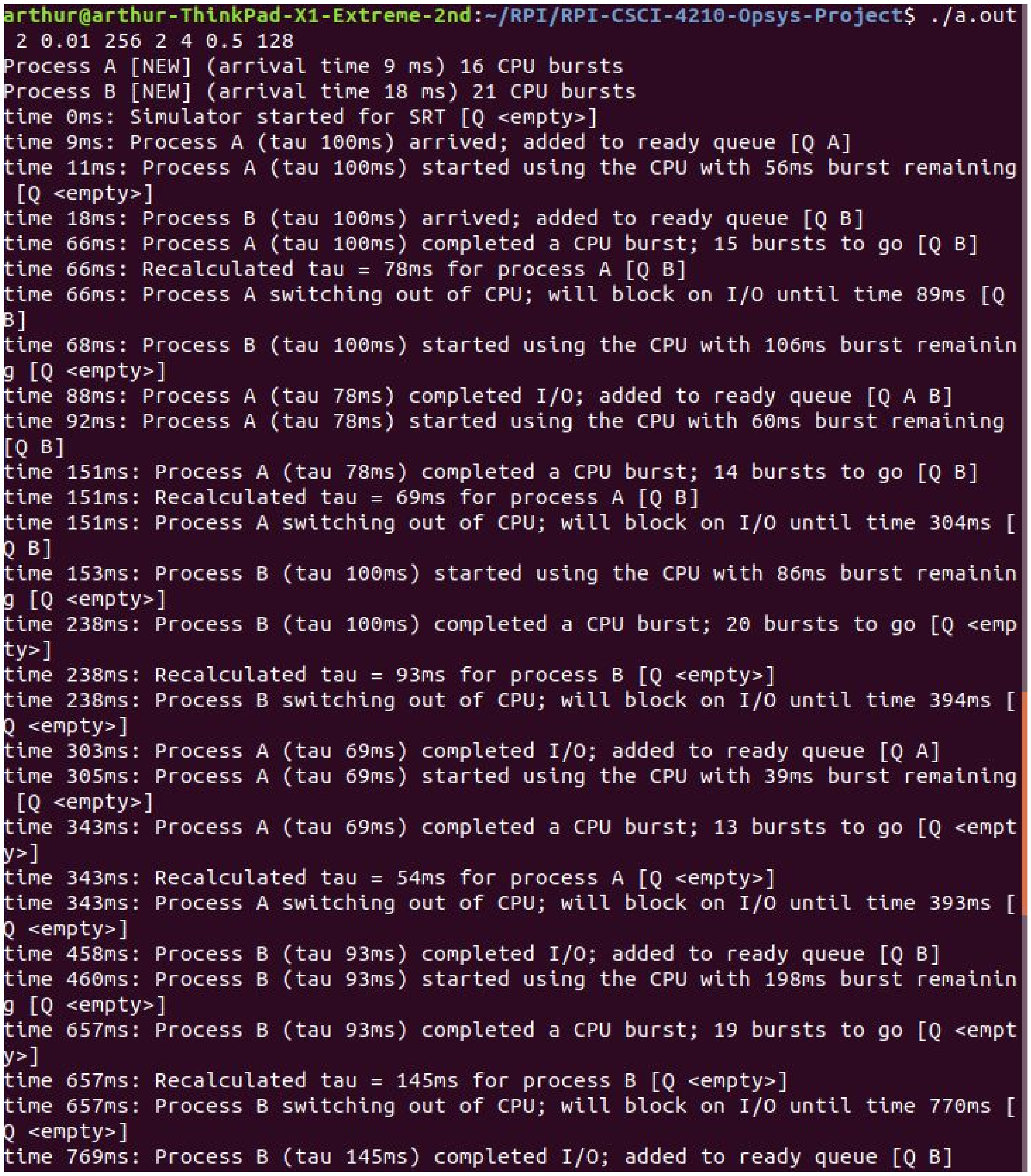
3. **For the SJF and SRT algorithms, what value of α seemed to produce the best results? Support your answer by citing specific simulation results.**

   For SRT, α ~ 0.5 tends to be better than the other settings, because of the following reasons. First, if the processes have very similar cpu burst times, an extreme large tau (~0.9) will lead to a situation where one process has a very small calculated tau value, so that it is always occupying the CPU and preempting other processes. For example, in figure 4, process A is always occupying the CPU because the first short CPU burst gave itself a very small tau. However, if the α is too small, it does not provide significant scheduling advantages, since the tau for each process will be similar in some cases, for instance, in figure 2, you may see the tau for process A and B are very close, which does not guarantee the best result. For randomly generated CPU bursts, using various seeds, there is no single best α for them, but α = 0.5 might be the best balance value among all the choices.



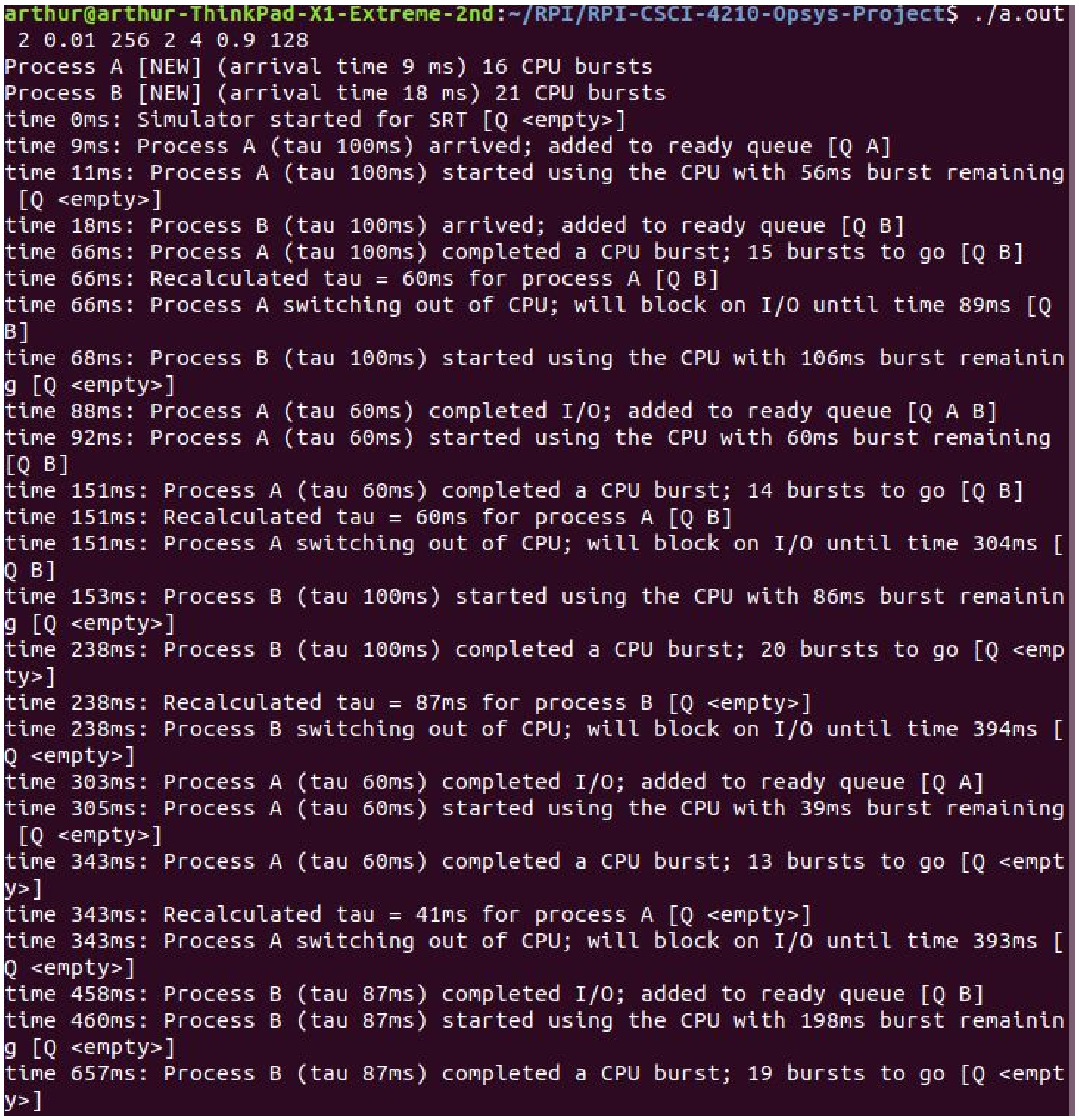
```
arthur@arthur-ThinkPad-X1-Extreme-2nd:~/RPI/RPI-CSCI-4210-Opsys-Project$ ./a.out
 2 0.01 256 2 4 0.1 128
Process A [NEW] (arrival time 9 ms) 16 CPU bursts
Process B [NEW] (arrival time 18 ms) 21 CPU bursts
time 0ms: Simulator started for SRT [Q <empty>]
time 9ms: Process A (tau 100ms) arrived; added to ready queue [Q A]
time 11ms: Process A (tau 100ms) started using the CPU with 56ms burst remaining
 [Q <empty>]
time 18ms: Process B (tau 100ms) arrived; added to ready queue [Q B]
time 66ms: Process A (tau 100ms) completed a CPU burst; 15 bursts to go [Q B]
time 66ms: Recalculated tau = 95ms for process A [Q B]
time 66ms: Process A switching out of CPU; will block on I/O until time 89ms [Q
B]
time 68ms: Process B (tau 100ms) started using the CPU with 106ms burst remainin
g [Q <empty>]
time 88ms: Process A (tau 95ms) completed I/O; added to ready queue [Q A B]
time 92ms: Process A (tau 95ms) started using the CPU with 60ms burst remaining
[Q B]
time 151ms: Process A (tau 95ms) completed a CPU burst; 14 bursts to go [Q B]
time 151ms: Recalculated tau = 91ms for process A [Q B]
time 151ms: Process A switching out of CPU; will block on I/O until time 304ms [
Q B]
time 153ms: Process B (tau 100ms) started using the CPU with 86ms burst remainin
g [Q <empty>]
time 238ms: Process B (tau 100ms) completed a CPU burst; 20 bursts to go [Q <emp
ty>]
time 238ms: Recalculated tau = 98ms for process B [Q <empty>]
time 238ms: Process B switching out of CPU; will block on I/O until time 394ms [
Q <empty>]
time 303ms: Process A (tau 91ms) completed I/O; added to ready queue [Q A]
time 305ms: Process A (tau 91ms) started using the CPU with 39ms burst remaining
 [Q <empty>]
time 343ms: Process A (tau 91ms) completed a CPU burst; 13 bursts to go [Q <empt
y>]
time 343ms: Recalculated tau = 85ms for process A [Q <empty>]
time 343ms: Process A switching out of CPU; will block on I/O until time 393ms [
Q <empty>]
time 458ms: Process B (tau 98ms) completed I/O; added to ready queue [Q B]
time 460ms: Process B (tau 98ms) started using the CPU with 198ms burst remainin
g [Q <empty>]
time 657ms: Process B (tau 98ms) completed a CPU burst; 19 bursts to go [Q <empt
y>]
time 657ms: Recalculated tau = 108ms for process B [Q <empty>]
```

Figure 2. ./a.out 2 0.01 256 2 4 0.1 128



```
arthur@arthur-ThinkPad-X1-Extreme-2nd:~/RPI/RPI-CSCI-4210-Opsys-Project$ ./a.out
 2 0.01 256 2 4 0.5 128
Process A [NEW] (arrival time 9 ms) 16 CPU bursts
Process B [NEW] (arrival time 18 ms) 21 CPU bursts
time 0ms: Simulator started for SRT [Q <empty>]
time 9ms: Process A (tau 100ms) arrived; added to ready queue [Q A]
time 11ms: Process A (tau 100ms) started using the CPU with 56ms burst remaining
 [Q <empty>]
time 18ms: Process B (tau 100ms) arrived; added to ready queue [Q B]
time 66ms: Process A (tau 100ms) completed a CPU burst; 15 bursts to go [Q B]
time 66ms: Recalculated tau = 78ms for process A [Q B]
time 66ms: Process A switching out of CPU; will block on I/O until time 89ms [Q
B]
time 68ms: Process B (tau 100ms) started using the CPU with 106ms burst remainin
g [Q <empty>]
time 88ms: Process A (tau 78ms) completed I/O; added to ready queue [Q A B]
time 92ms: Process A (tau 78ms) started using the CPU with 60ms burst remaining
[Q B]
time 151ms: Process A (tau 78ms) completed a CPU burst; 14 bursts to go [Q B]
time 151ms: Recalculated tau = 69ms for process A [Q B]
time 151ms: Process A switching out of CPU; will block on I/O until time 304ms [
Q B]
time 153ms: Process B (tau 100ms) started using the CPU with 86ms burst remainin
g [Q <empty>]
time 238ms: Process B (tau 100ms) completed a CPU burst; 20 bursts to go [Q <emp
ty>]
time 238ms: Recalculated tau = 93ms for process B [Q <empty>]
time 238ms: Process B switching out of CPU; will block on I/O until time 394ms [
Q <empty>]
time 303ms: Process A (tau 69ms) completed I/O; added to ready queue [Q A]
time 305ms: Process A (tau 69ms) started using the CPU with 39ms burst remaining
 [Q <empty>]
time 343ms: Process A (tau 69ms) completed a CPU burst; 13 bursts to go [Q <empt
y>]
time 343ms: Recalculated tau = 54ms for process A [Q <empty>]
time 343ms: Process A switching out of CPU; will block on I/O until time 393ms [
Q <empty>]
time 458ms: Process B (tau 93ms) completed I/O; added to ready queue [Q B]
time 460ms: Process B (tau 93ms) started using the CPU with 198ms burst remainin
g [Q <empty>]
time 657ms: Process B (tau 93ms) completed a CPU burst; 19 bursts to go [Q <empt
y>]
time 657ms: Recalculated tau = 145ms for process B [Q <empty>]
time 657ms: Process B switching out of CPU; will block on I/O until time 770ms [
Q <empty>]
time 769ms: Process B (tau 145ms) completed I/O; added to ready queue [Q B]
```

Figure 3. ./a.out 2 0.01 256 2 4 0.5 128



```
arthur@arthur-ThinkPad-X1-Extreme-2nd:~/RPI/RPI-CSCI-4210-Opsys-Project$ ./a.out
 2 0.01 256 2 4 0.9 128
Process A [NEW] (arrival time 9 ms) 16 CPU bursts
Process B [NEW] (arrival time 18 ms) 21 CPU bursts
time 0ms: Simulator started for SRT [Q <empty>]
time 9ms: Process A (tau 100ms) arrived; added to ready queue [Q A]
time 11ms: Process A (tau 100ms) started using the CPU with 56ms burst remaining
 [Q <empty>]
time 18ms: Process B (tau 100ms) arrived; added to ready queue [Q B]
time 66ms: Process A (tau 100ms) completed a CPU burst; 15 bursts to go [Q B]
time 66ms: Recalculated tau = 60ms for process A [Q B]
time 66ms: Process A switching out of CPU; will block on I/O until time 89ms [Q
B]
time 68ms: Process B (tau 100ms) started using the CPU with 106ms burst remainin
g [Q <empty>]
time 88ms: Process A (tau 60ms) completed I/O; added to ready queue [Q A B]
time 92ms: Process A (tau 60ms) started using the CPU with 60ms burst remaining
[Q B]
time 151ms: Process A (tau 60ms) completed a CPU burst; 14 bursts to go [Q B]
time 151ms: Recalculated tau = 60ms for process A [Q B]
time 151ms: Process A switching out of CPU; will block on I/O until time 304ms [
Q B]
time 153ms: Process B (tau 100ms) started using the CPU with 86ms burst remainin
g [Q <empty>]
time 238ms: Process B (tau 100ms) completed a CPU burst; 20 bursts to go [Q <emp
ty>]
time 238ms: Recalculated tau = 87ms for process B [Q <empty>]
time 238ms: Process B switching out of CPU; will block on I/O until time 394ms [
Q <empty>]
time 303ms: Process A (tau 60ms) completed I/O; added to ready queue [Q A]
time 305ms: Process A (tau 60ms) started using the CPU with 39ms burst remaining
 [Q <empty>]
time 343ms: Process A (tau 60ms) completed a CPU burst; 13 bursts to go [Q <empt
y>]
time 343ms: Recalculated tau = 41ms for process A [Q <empty>]
time 343ms: Process A switching out of CPU; will block on I/O until time 393ms [
Q <empty>]
time 458ms: Process B (tau 87ms) completed I/O; added to ready queue [Q B]
time 460ms: Process B (tau 87ms) started using the CPU with 198ms burst remainin
g [Q <empty>]
time 657ms: Process B (tau 87ms) completed a CPU burst; 19 bursts to go [Q <empt
y>]
```

Figure 4. ./a.out 2 0.01 256 2 4 0.9 128

4. **For the SJF and SRT algorithms, how does changing from a non-preemptive algorithm to a preemptive algorithm impact your simulation results?**

   If there is only one process in the system, it does not make any difference, as there is no preemption. For multiple processes, SRT can significantly reduce the average waiting time among all processes, when compared to SJF, since it supports preemption. So an "expected-to-be-shorter-CPU-burst" process can always have a higher priority of running CPU bursts, thus reducing the total waiting time for all the processes.

   That being said, since we are always estimating the expected tau value of each process based on its past CPU burst performance, the estimation can go wrong if the process does not have a reliable pattern of performing CPU bursts. In other words, if the process did short CPU bursts in the past, but suddenly has a very long CPU burst, it may occupy the CPU for a long time, and no other processes can preempt.

5. **Describe at least three limitations of your simulation, in particular how the project specifications could be expanded to better model a real-world operating system.**

   There are many limitations, in terms of our simulation. First, we don't have the implementation of the IO functionality, and only focus on the CPU burst. A better model should have an IO queue and scheduling algorithms for IO. Secondly, we don't have a method that kills processes, or closes processes. It would mirror real world operations if processes could get cancelled. Third and finally, we don't consider system usage, that will block the cpu.

6. **Describe a priority scheduling algorithm of your own design (i.e., how do you calculate priority?). What are its advantages and disadvantages? Support your answer by citing specific simulation results.**

A possible scheduling algorithm, that we could develop, can be referred to as a “greedy, ‘round robin-like’ algorithm, with a ‘shortest job first consideration’.” In our proposed priority scheduling algorithm, similar to round robin scheduling, we would allot time slices, in order to complete the given processes. The difference, however, is that we would order each process’ personal time quantum, based on the priority of having the shortest time to complete, hence the term “greedy.”

The individual processes would be sorted by the time of completion, in which the smallest jobs go first. The advantages of this algorithm are that it is both fair and mindful of the time it takes to finish each process. The disadvantages are that, first, due to there being a priority, starvation is possible, and second, due to time quantum inclusion, there is a strong dependency on the time slice selection. In particular, based on simulations three and four results, this algorithm would be able to better prioritize multiple processes, unlike round robin, while still providing fairness, unlike non-round robin algorithms.