Assignment 1 Report

SYSC 4001-A: Operating Systems

Github Project: https://github.com/sidj21/SYSC4001_A1

Ajay Srirankan (101313173)

Siddarth Jain (101304051)

Objective

The purpose of this part of the assignment is to simulate CPU handling interrupts from system calls and I/O devices. The timing of various parts of the interrupt process is tweaked and analyzed to see the impact on performance.

Methodology

The project has three trace files, one for a CPU-bound program, one for an I/O-bound program, and the last file is a mix of a CPU and I/O intensive program.

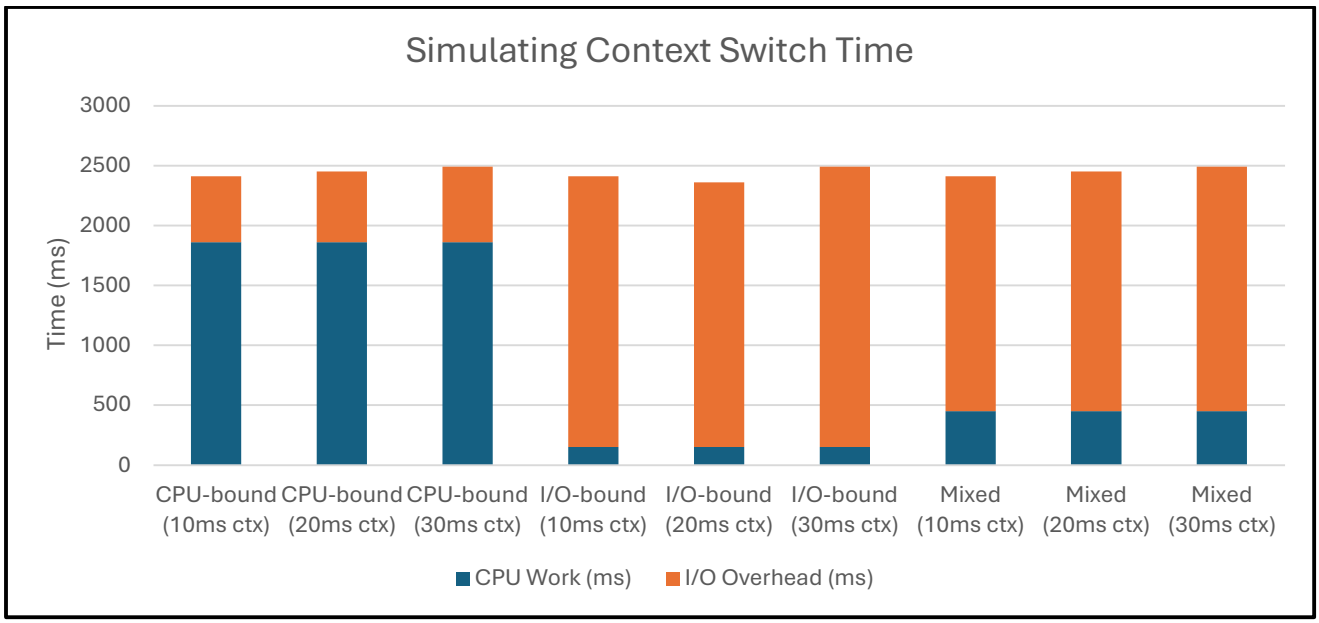
The following seven parameters were changed per trace file, for a total of 21 output files.

- Context switch times (10ms, 20ms, and 30ms)
- Time for an ISR's first activity (40ms, 80ms, and 160ms)
- CPU Speed (200Hz)

Each SYSCALL's ISR adds up to the corresponding index in **device_table.txt**. The first activity is hard coded to 40ms. The second activity fills in the remaining time for the SYSCALL. The third activity is only added if there is a specified delay and it is used for simulation cases. It can be seen as changing the duration of the first activity to something between 40ms-200ms. END_IO is just one line in the execution file.

The GitHub project is available here: **https://github.com/sidj21/SYSC4001_A1.**

Context Switch Time

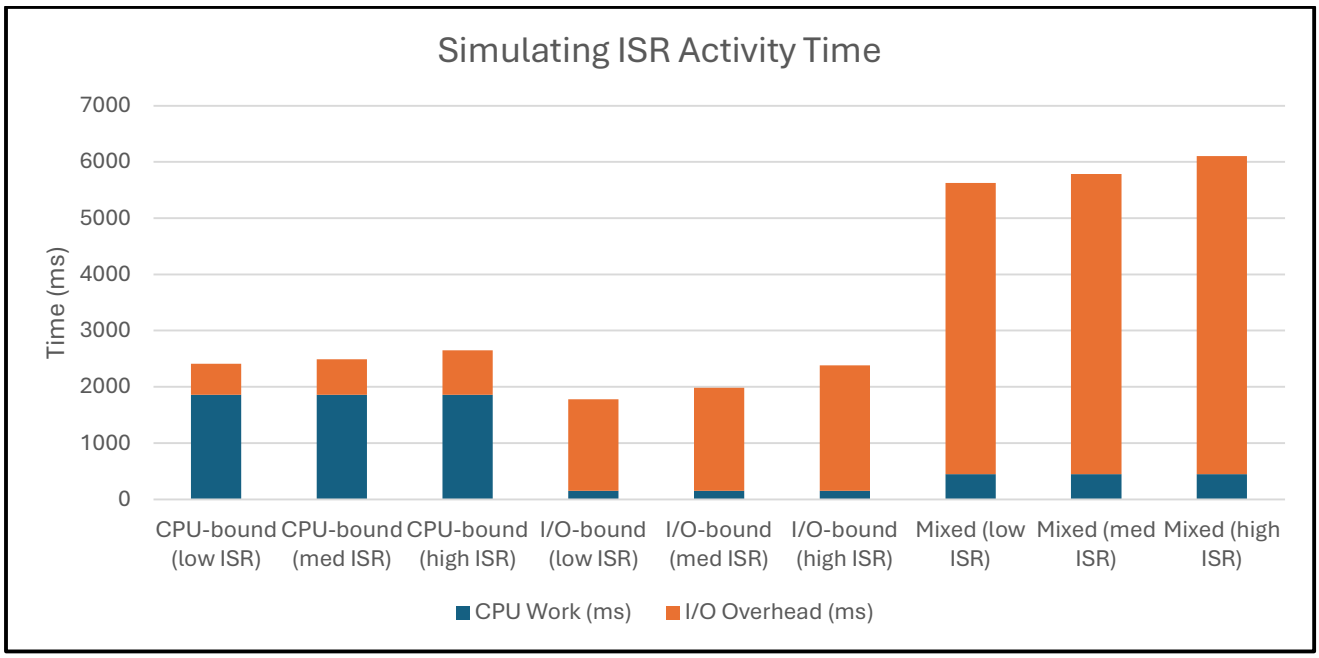


Graph 1: Simulating Context Switch Time (10ms, 20ms, 30ms)

When we are increasing the context time, the CPU spends more time switching between tasks. This leads to a higher overhead for the CPU. For example, just the I/O overhead for the CPU-bound trace file goes from 550ms to 630ms when the context switch time goes from 10ms to 30ms, which is a 13.55% increase. But for the I/O-bound trace file, there was only a 3.48% increase in the I/O overhead. This makes sense as the ISR bodies and device delays dominate when compared to incrementing the context switch time by 10ms.

Overall, there was not a huge gap when changing the context time. It represents responsiveness as in how fast the CPU can save its state to service an interrupt.

ISR Activity Time

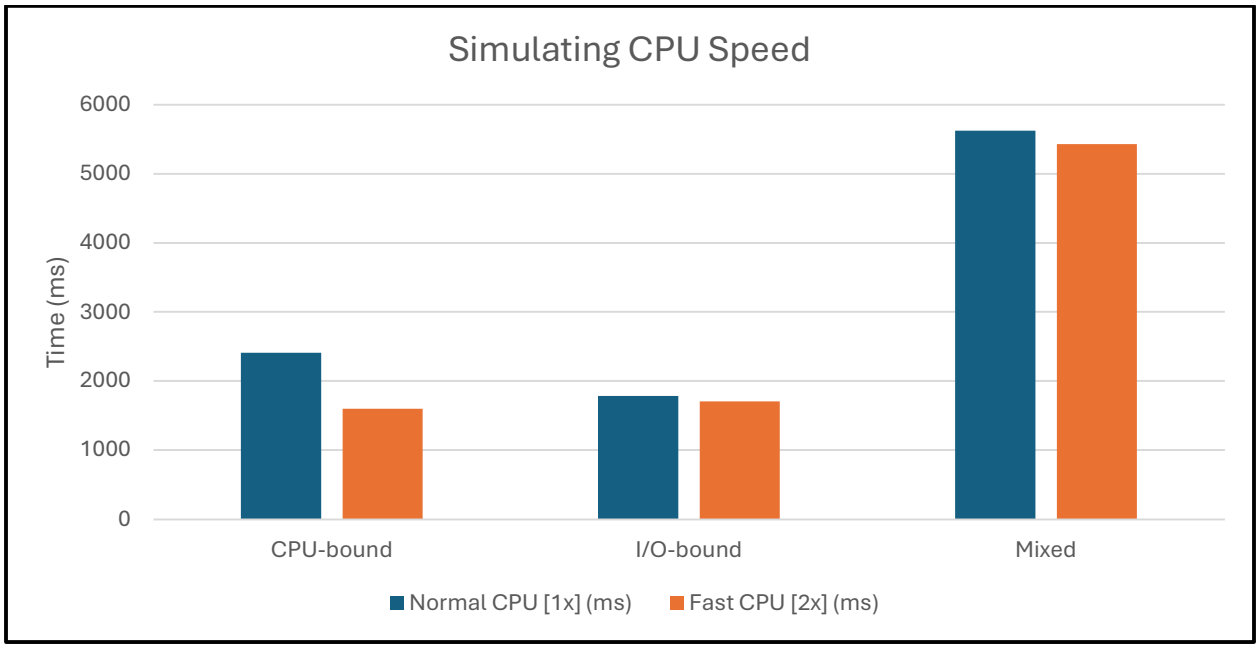


Graph 2: Simulating ISR Activity Time (40ms, 80ms, 160ms)

When the ISR activity time is increased, the CPU spends a lot more time inside the interrupt service routine rather than processing any given program. The I/O-bound program was impacted the most as more delays were added. Going from a low ISR delay (+40ms) to a high ISR delay (+120ms) increased the total execution time from 1782ms to 2382ms (28.8% increase). For CPU-bound and mixed programs, this was closer to an 8-9% increase. This is because SYSCALLs in succession trigger the interrupt service routine, leading to a bigger interrupt latency overall. *Aside:* There was no delay for the “low ISR”.

In general, this leads to longer response times, delayed execution of the next tasks, and reduced system responsiveness. In extreme cases, other interrupts may be delayed so much that they may be missed, or the system fails to meet timing constraints.

CPU Speed



Graph 3: Simulating CPU Speed (100Hz, 200Hz)

An interesting parameter to change is the CPU's execution speed. When increasing the CPU speed from 100Hz to 200Hz, the total execution time is only drastically reduced for CPU-bound processes, with a 40% drop. *Aside:* The normal CPU is one with the default context saves time (1ms) and no ISR delay. An interesting connection can be made with the I/O-bound and mixed program. Even a faster CPU does not help because the bottleneck is the slower devices (e.g. the printer from part 1 of the assignment).

Conclusion

The difference in the speed of context switching and ISR execution steps directly determines the overall execution time of the process. A longer save or switch context time increases the delay of the interrupts while the ISR activity increases the time the CPU spends on servicing the device instead of running the main program. As these times increase, the total runtime of the process grows, tasks get delayed, and the system becomes less responsive.

Also, increasing the time for the ISR body had a much greater impact on the total execution time than increasing the context switch speed. Since the switch context time mostly consists of basic MOV operations, the bottleneck is the amount of data in the CPU to save. So to reduce execution time, the effort should go towards simplifying and making ISR bodies more efficient.