Assignment 1 Part 2: Group Report
SYSC 4001 A, L4

Justin Kim
101298662

Dorian Bansoodeb
101309988

Monday, October 6th, 2025

Github Repository Link:https://github.com/Dorianbansoodeb/SYSC4001_A1

Objective: The goal of this assignment is to implement a simulator in C++ that models how the interrupt handling process works in a batch operating system. The simulator tracks input/output operations, CPU activity, and all events involved in the interrupt cycle. The system's goal is to track and analyze the performance by changing different parameters such as ISR execution duration, if a faster/slower CPU was used and context switch time.

Methodology: The interrupt handling simulator was implemented using the provided files in C++ in the repository (interrupts.cpp + interrupts.hpp). The program reads a trace.txt file line by line to simulate the events in Batch OS such as CPU running, and I/O events.

Each line in the trace file contains an activity (CPU, SYSCALL, END_IO), which are all treated differently and a number, which is treated differently based on the activity. The corresponding actions from each line are logged in an output file (execution.txt for example).

The simulation is able to model the stages of the interrupt process by keeping track and incrementing a running time clock (now_ms), keeping track of time as more actions are performed. Some of these actions are:

- Switching to kernel mode (1ms)
- Vector and ISR Lookup (1ms)
- Data Transfer (40ms)
- IRET (1ms)

For each interrupt event, the simulator retrieves the timing and vector information from the device_table.txt and vector_table.txt files. These files contain the delays and ISR addresses to be searched up. The results are saved line by line for each step, into an output file (execution.txt for example). A line would contain entries, separated by commas showing the timestamp, duration of the event, and the description of the event. (0, 51, CPU burst) for example.

This simulation allows us to observe how changing timings of different parameters affects the overall performance of the execution time and system .

Simulation #	Change	Effect
1	Base case	
2	Base case	
3	Base case	
4	Base case	
5	Base case	
6	Context change 30ms	CPU waits for 60ms each int. cycle
7	ISR cost 100ms	Increases program time
8	ISR cost 150ms	Increases program time
9	Base case	
10	Base case	
11	Base case	
12	Base case	
13	Base case	
14	Base case	
15	Base case	
16	Base case	
17	Base case	
18	Context change 20ms	CPU waits for 40ms each int. cycle
19	ISR cost 200ms	Increases program time
20	Context change 10ms	CPU waits for 20ms each int. cycle