1. Introduction

The purpose of this assignment was to design and implement a simulator for interrupt handling in an operating system. The simulator processes a trace file that includes CPU bursts, system calls, and device I/O completions. It reproduces the full interrupt mechanism (kernel switch, context save/restore, vector lookup, ISR execution, and IRET) and records the execution timeline into an output log.

This report presents the test case results, analyzes the impact of various parameters (context switching time, ISR length, device delays) and highlights performance bottlenecks.

2. Methodology

The simulator was implemented in CPP using the provided template.

The following assumptions and parameters were used:

- Context save/restore: 10 ms each
- Kernel/User mode switch: 1 ms
- Vector lookup and address load: 1 ms each
- ISR activity execution: fixed in 40 ms chunks
- Device delays: based on device_table.txt
- No I/O queueing

The input trace file (see below) guided the simulation:

```
CPU, 50
SYSCALL, 7
END_IO, 7
CPU, 75
SYSCALL, 5
END_IO, 10
CPU, 51
SYSCALL, 14
END_IO, 19
CPU, 75
SYSCALL, 6
END_IO, 8
CPU, 100
SYSCALL, 12
END_IO, 12
CPU, 90
SYSCALL, 13
END_IO, 15
CPU, 20
SYSCALL, 5
END_IO, 5
CPU, 30
SYSCALL, 3
END_IO, 3
```

The simulator produced a detailed execution timeline (context switches, ISR execution in slices, I/O interrupts).

3. Results

3.1 Example Output Snippet From the simulation (execution.txt):

```
50, 1, switch to kernel mode
51, 10, context saved
61, 1, find vector 7 in memory position 0x000E
62, 1, load address 0X00BD into the PC
63, 40, execute ISR activity
...
403, 1, switch to user mode
404, 152: interrupt
```

3.2 Observations

- CPU bursts executed correctly, pausing when I/O interrupts occurred.
- ISR execution time matched the device delay (chunked into 40 ms steps).
- Context save/restore overhead added measurable latency to each interrupt.
- Longer device delays (ex. device 12 or 13) led to proportionally longer ISR activity sequences.

4. Analysis

Impact of Context Switch Time

Increasing context save/restore from 10 ms → 20 ms → 30 ms adds significant overhead. For frequent interrupts, total runtime increases nearly linearly with context switching cost.

Impact of ISR Duration

When ISR execution is short (ex. 40 ms), overhead is dominated by kernel/user switches. However, when ISR execution grows large (ex. >200 ms), ISR time dominates and delays return to user space are significant.

Impact of Device Delays Devices with high latency (ex. device 12 at 600 ms) lead to long ISR sections in the trace. In such cases, CPU efficiency is reduced because the system spends excessive time inside the ISR.

Overhead vs. Useful Work

From the output, we can see that about 15–20% of total time is spent in overhead (context switching, saving state, vector lookup), while the rest is real CPU or ISR work. This aligns with expectations for interrupt-driven systems.