# Interrupts Simulator Analysis Report

## Github Repository

This report presents the results of running the Interrupts Simulator on a series of test cases. The simulator models the execution of CPU bursts, system calls, and I/O interrupts based on trace files. The goal was to evaluate how different parameters—such as context save/restore time, ISR body execution time, vector table configurations, CPU speed, and memory limits—impact system performance.

**Control Test Trace:**
CPU, 50 → SYSCALL, 7 → END_IO, 7 → CPU, 100 → SYSCALL, 12 → END_IO, 12 → CPU, 30
With context_save_time = 10 and ISR body = 40 ms, the execution trace showed predictable overheads: 1 ms kernel switch, 10 ms context save, 1 ms vector lookup, 1 ms ISR fetch, 40 ms ISR body, and 1 ms IRET. The I/O delay matched the values in device_table.txt. This test served as a baseline for comparisons.

**Varying Context Save Time**
Increasing the context save time from 10 → 20 → 30 ms showed a direct linear increase in interrupt overhead. The system spent more time switching between user and kernel mode. Longer save times also caused higher latency in handling I/O completions, confirming that context switch cost is critical in overall responsiveness.

**Varying ISR Body Time**
When the ISR execution time increased from 40 ms up to 200 ms, the interrupt handling overhead grew significantly. This caused delayed IRET execution and postponed CPU resumption. In scenarios with back-to-back interrupts, a long ISR body created a bottleneck, showing that ISR efficiency is essential to avoid starving user processes.

**Vector Table Parameters**
Changing ADDR_BASE from 0 to 1000 and VECTOR_SIZE from 2 to 4 resulted in different memory position lookups for ISR addresses. While the effect on total execution time was negligible (since lookup takes 1 ms), it highlighted how the simulator correctly adapts address resolution based on table configuration.

**CPU Speed and Memory Limits**
Increasing CPU_SPEED from 100 to 200 made CPU bursts finish sooner, reducing total execution time for CPU-bound traces. Adjusting MEM_LIMIT simulated constraints in memory access; when multiple I/O operations were scheduled, the system's ability to handle them depended on available memory. Although the simulation assumed devices were always available, this parameter can influence future extensions.

**Overall Observations**
1. Interrupt handling overhead grows linearly with context save/restore time.
2. Long ISR bodies (≥ 200 ms) create bottlenecks and degrade responsiveness.
3. CPU_SPEED affects only CPU bursts, not interrupt handling overhead.
4. Vector table base and size alter address lookups but have negligible runtime cost.
5. A balance between CPU processing, ISR execution, and I/O delays is essential for system efficiency.

**Conclusion**
The tests confirm that interrupt handling introduces measurable overhead. Optimizing context switching and ISR execution time is crucial for maintaining system responsiveness.