Interrupt Simulator Analysis — SYSC 4001 Assignment 1

SYSC4001_A1

Mahdi Bouakline
Timor Grigoryev

101257788
101276841

This report presents and analyzes the results of 21 simulation cases executed using our Interrupt Simulator. The simulator models CPU and I/O interrupt handling using different context save times and ISR activity durations to analyze system performance, overhead, and efficiency.

Our approach was to develop 3 different trace files, the first being the example from the professor. With those 3 files, we executed tests using all 3 as inputs, while varying the ISR activity durations and context save times. We modified those values seven times, resulting in a total of 21 unique simulations.
To summarize, we simulated with the following variations:

- Context Save Time: 10, 20, 30 ms
- ISR Activity Time: 40, 100, 150, 200 ms
- Different trace files (varying CPU bursts & device delays)

As mentioned in a comment in our interrupts.cpp: There is a discrepancy in the assignment description and what was posted by the professor as an example. The assignment notes take into account and print IRET, get ISR address, etc... For simplicity, and based on what we found to be most logical, we have decided to go with the most recent instructions/examples we were given. Generally, whenever we were uncertain with a requirement, we decided to comply with what the professor's example implemented.

Now, we must analyze our results. Below are our three trace files.
Trace 1:
CPU, 51
SYSCALL, 14
CPU, 39
END_IO, 14
CPU, 72
SYSCALL, 19
CPU, 28
END_IO, 19

Trace 2:
CPU, 80
SYSCALL, 5
END_IO, 5
CPU, 120
SYSCALL, 10
END_IO, 10
CPU, 30

Trace 3:
CPU, 60
SYSCALL, 3
END_IO, 3
CPU, 90
SYSCALL, 15
END_IO, 15
CPU, 25

We initially started by testing every trace with a modified context save time, and the initial ISR at 40, our results are below:

Test Number	Context Save Time (ms)	Total Exec. Time (ms)
1	10	2458
2	10	1832
3	10	963
4	20	2498
5	20	1872
6	20	1003
7	30	2538
8	30	1912
9	30	1043

The results showed a clear and predictable pattern: for every 10 ms increase in context save time, total execution time increased by exactly 40 ms across all traces. This linear relationship makes sense, as there are four interrupts in each trace, and each additional 10 ms of context save time accumulates to a total increase of 40 ms. The effect was consistent across all trace files, demonstrating that context save time has a small but predictable impact on overall execution.

Next, we conducted similar tests, by simulating every trace with a modified ISR Activity Time, and the initial context save time at 10, our results are below:

Test Number	ISR Activity Time (ms)	Total Exec. Time (ms)
1	80	1042
2	80	1082
3	80	1027
4	120	1442
5	120	1482
6	120	1427
7	160	1842
8	160	1882
9	160	1827
10	200	2242
11	200	2282
12	200	2227

Next, we examined the impact of varying ISR activity time, while keeping the context save time constant at 10 ms. Unlike the context save tests, these results showed a dramatic increase in total execution time as ISR activity increased. For example, raising the ISR activity from 40 ms to 200 ms increased execution time by approximately 1200 ms, depending on the trace file. This is because ISR work contributes more heavily to execution time: each SYSCALL interrupt runs three ISR segments, and each END_IO interrupt runs two ISR segments. As a result, ISR activity dominates the overhead, while context save contributes relatively little in comparison. From these results, several key conclusions can be drawn:

1. **Impact of Context Save Time:** Although increasing context save time consistently increases total execution, the magnitude of the effect is relatively small. The linear pattern is predictable because context save occurs once per interrupt.

2. **Impact of ISR Activity:** ISR duration has a far larger effect on total execution time. Longer ISR routines substantially increase system overhead because the ISR runs multiple segments for each interrupt.

3. **Performance Implications:** The difference in the influence of context save versus ISR execution highlights the importance of efficient ISR routines. Optimizing ISR activity, even marginally, can have a significant effect on overall system performance.

Some things to note:

- Increasing CPU speed would reduce the effective time for CPU bursts, but ISR and context save overhead would remain a limiting factor for total execution.
- Address Size: Using larger addresses (e.g., 4 bytes instead of 2) would slightly increase memory access times in the vector lookup step, marginally increasing overhead.

Overall, our simulation data demonstrates that ISR execution time is the dominant contributor to interrupt overhead, while context save time has a smaller but linear effect. These results highlight the importance of optimizing ISR routines in real-time and embedded systems. The tables and simulations provide a coherent and analyzable picture of interrupt processing overhead, enabling informed discussion of performance tradeoffs.