SYSC4001 Assignment 1 Interrupt Simulator Report
Authors: Ajan Balaganesh, Kyle Deng

Introduction:

We implemented a mini interrupts simulator in C++ which gives all the major steps of the interrupt process which helped reinforce our overall understanding of interrupts in general. Each step(line) of the interrupt simulator listed three things. The first value listed was the current system time in ms, the second value was the description of the step and the third values was the duration of the step in ms. We added the code for the actual simulation part in interrupts.cpp and we stored our simulation traces/input in the input_files directory. The output files or the execution files are stored in the output_files directory and it corresponds to the trace with the same number (0 through 20). Not all the traces in the input_files directory are unique because there were certain simulations that were testing the effect that a code change would have on the same trace. An example of this is having the different context save times, where we changed the context save time from 10ms to 20ms to 30ms in the code and we simulated it with the same trace file so that we can see the difference in execution time. The analysis of the changing parameters is below.

Analysis:

There were many different simulations ran to look at the effect of the three parameters. The first parameter that was looked at was the context save time. We varied it between 10ms, 20ms and 30 ms. The second parameter that was looked at was the ISR activity time. To observe this parameter we made trace files that would call both SYSCALL and ENDIO on different devices that had ISR activity times between 40ms and 200ms and looked at the results. The third parameter that was looked at was the CPU speed. To observe the effect of CPU speed we had trace files that had the same fundamental steps but the only difference between them was the time for CPU burst between SYSCALL steps and ENDIO steps.

Context Save Time Results:

When we increased the context save time from 10 ms to 20 ms to 30 ms, the total execution time grew pretty linearly. This is because there is exactly one context save step in each of the interrupt steps(SYSCALL and ENDIO). You can see in execution number 6(context save time of 10 ms), the total time for execution is 31851 ms. When we increased the context save time in execution number 14(context save time of 20 ms) the total time for execution was 32451 ms and then when we finally increased the context save time to 30 ms in execution 15, the total execution time became 33051 ms. From this we saw that each 10 ms increase in context save time caused a 600 ms increase for the total execution time of the trace because there were 60 SYSCALL and ENDIO steps. This proves that increasing the context save time will increase the total execution time proportionally and linearly.

ISR Activity Time Results:

During execution 7, 8, 9, 10, 11, 12 and 13 we wrote traces that would call ISRs that had a duration between 40 and 200 ms. During these simulations we noticed longer ISRs would increase the total execution time because the CPU would have to deal with the interrupt fully first before returning back to its original context and carrying out the rest of the work. For example, in execution 7 the ISR that was called had a duration of 68 ms and the total execution time of the trace was 220 ms(256 ms with the final CPU burst at the end). On the other hand, in execution 12 an ISR with a duration of 145 ms was called and it had a total execution time of 374 ms (410 ms with the final CPU burst). This total execution time increased by 154 ms despite having all the same steps shown in the execution file (only difference is the duration for ISR procedure-related steps) which further proves that longer ISR activity times will lead to longer total execution times.

CPU Processing Time Results:

During execution 16, 17, 18, 19 and 20 we tested the interrupt simulator with CPU processing/burst times of 5 ms, 10 ms, 20 ms, 40 ms and 80 ms respectively. The CPU's speed linearly decreased the total execution time of the trace(higher speed means less duration during the CPU burst step). For example during execution 16 every CPU burst was given a duration of 5 ms and the total execution time was 2026 ms. On the other hand, in execution 20 the speed was less because the CPU burst duration was longer (80 ms) and the total execution time was 2476 ms. There is a difference of 450 ms due to CPU speed between the two executions which proves that CPU speed is disproportional to total execution time(higher speed means less execution time).

Effect of Having a Two Byte address vs Four Byte address:

In the current implementation of our interrupt simulator the look up time is constant and doesn't change based on the number addresses there are to search so changing the address to 4 bytes would not have any effect. On the other hand, in a real computer system a four byte address would mean more addresses to search and more memory for the vector table. This would make the look up time a bit slower in real life. However, the impact on the total execution time would still be very little and can be neglected in most cases.

Conclusion:

Overall there are many things that affect the total execution time of the trace. Higher context save times, longer ISR activity times and slower CPU speeds can all contribute to increasing the total execution time and should be optimized so that the computer system can process and

accomplish tasks quickly. On the other hand, the amount of bytes in a memory address does not really have any impact on the system that is worth considering. It should be dictated by how many addresses are needed. For example if a system has many peripheral devices and many ISRs then there should be enough addresses to accommodate them.

GitHub Repo of Interrupt Simulator: AjanzzSkool/SYSC4001_A1: Sysc 4001 Assignment 1 Code