SYSC4001 - Assignment 1 Part II

Sohaila Haroun - 101297624
Zaineb Ben Hmida - 101302936

In this report we will put the reference of the execution file and relate it to the input trace file and discuss our observations from that simulation/test case.

Principal Example

0 - Used example from git (trace_6.txt)

The first execution file we included in our submission is execution_0.txt, which was the result of using the example input file from GitHub, which we called trace_6.txt in our input files. There is nothing noteworthy about this example except that it works as outlined in the assignment description, the output is slightly different compared to the example on GitHub which seems to be due to a difference in understanding of the assignment guidelines. For example, our program includes IRET and the time it takes for the ISR activities to execute actually adds up to the

Tested and simulated the effect of varying ISR activity time

Using trace.txt

1.1 - Program with each ISR activity taking 40ms

1.2 - Same but 80ms ms intervals/chunks

1.3 - Same but 200ms intervals/chunks

1.4 - Randomized time per activity

Observations:

It is important to note that our understanding of this requirement is that for the duration of the delay respective to the device, there is an undetermined number of ISR activities that will run and be recorded in our simulation that must add up to that delay. Meaning if a device had a delay of 200ms, in case 1 there would be 5 40ms ISR activities, case 2 there would be 2 80ms activities and 1 40ms activity, and in case 3 there would just be 1 200ms activity. Despite the varying number of activities and their durations, the ISR body would always execute in 200ms. In case 4, the number of activities is completely random but IRET still executes 200ms after they start, and this case aligns best with actual ISR behaviour as activities often have varying durations for each device (not constant like 40-40-40..), but each device's ISR will usually take the same amount of time to execute. These variations in ISR activity durations did not make a real difference to the interrupt simulation.

It is important to note that we decided to implement this portion as a loop where it repeated the same description of the activities "ISR SYSCALL/END_IO Activity", not because we did not have an understanding of what kind of activities are commonly done in ISRs but because it wasn't feasible to program an efficient simulation that would vary the activities based on their devices and their overall delays, especially since our implementation (as recommended by the prof) was using randomized activity times, meaning sometimes an ISR could have 1 activity or 10 activities. We decided to demonstrate our knowledge of common ISR activities in

this section of the report. Some common examples are: calling device drivers, processing the interrupt (checking priority, if masked), transferring data (like storing information into memory or vice versa), updating system clock, and many more.

Of course if the time it took for an activity increased, it would make the ISR take longer and would make the whole program slower, affecting both the throughput and the turnaround time, especially if the increase was significant or if the device was used a lot.

Tested and simulated the effect of increasing save/restore context time:
Short trace file used: trace_6.txt
Long trace file used: trace_5.txt
2.1.1 - Value of the save/restore context time = 10 using short trace file
2.1.2 - Value of the save/restore context time = 10 using long trace file
2.2.1 - Changed to 20ms using short trace file
2.2.2 - Changed to 20ms long trace file
2.3.1 - Changed to 30ms short trace file
2.3.2 - Changed to 30 ms long trace file

Overall observations:

As the save/restore context time increases, each interrupt takes longer to handle and the ISR activity takes longer. As a result, the CPU has less time to execute the main programs and the throughput of the system decreases overall.
These differences and observations are more evident with the longer traces vs the shorter trace file, as the longer trace files have more interrupts, which means the CPU spends more time handling the interrupts. In the shorter trace files, the CPU is less affected as less interrupts exist. Therefore, increasing the save/restore context results in a slower execution time since interrupt handling time increases, giving the CPU less execution time, especially in heavier workloads or longer traces.

Tested and simulated the effect of changing address size:
Using trace_4.txt
3.0 - Byte 2
3.1 - change the address of bytes to 4 from 2
Doubling the byte size from 2 to 4, `#define VECTOR_SIZE 2` -> `#define VECTOR_SIZE 4`, doubles the size of each vector entry in memory. Instructions referenced by byte 2 are now referenced by byte 4.

Observations:

In some situations timing of the ISR activities decreased with a byte size of 4. By doubling the vector size, memory addresses were shifted, so now some ISR operations that required indirect access with a byte size of 2 no longer need to with a byte size of 4. This reduces some delays, specifically in ISR operations that involve fetching vector addresses. Overall, there are no functional changes to the program by doubling the byte size, and most of the execution remains the same.

Tested and simulated the effect of CPU speed on different kinds of input files:
4.0 Normal CPU time
4.0.1 with short file (trace 2) : 4052 ms
4.0.2 with long file (trace 3) : 28130 ms
4.0.3 with more i/o (trace 7) : 4703 ms
4.0.4 with more cpu (trace 8) : 3219 ms
4.1 make CPU 2x faster
4.1.1 with short file : 3690 ms
4.1.2 with long file : 24662 ms
4.1.3 with more i/o : 4483 ms
4.1.4 with more cpu : 639 ms
4.2 make CPU 2x slower
4.2.1 with short file : 4414 ms
4.2.2 with long file : 31598 ms
4.2.3 with more i/o : 4923 ms
4.2.4 with more cpu : 5799 ms

Observations:

It is often the goal to have a “faster computer”, and that is often translated as having a faster CPU, however, here we tested the extent of which CPU speed affects the time it takes for a program to be completed and we compared different kinds of programs/input/trace files. Namely we wanted to see the effect of different CPU speeds on long vs short files and programs that are input/output bound vs CPU bound. We did this by making the CPU speed only affect the CPU bursts and not the syscall/end_io portions as although their speed would be affected, that effect is likely negligible compared to the amount of time it takes for I/O, especially due to the limitations of this very simplified interrupt simulator.

It was very apparent that a faster CPU does yield in faster executing programs, however it is noteworthy to mention that the difference it made for the short vs long files was almost the exact same, this means that although the difference should be more drastic (net difference) for long files and more significant (relational difference) for short files, that is not the main factor that is affected by CPU speed. In fact, CPU speed, when varied for two very short (num of instructions = 12) programs that are differently bounded, yielded quite different results. In the example of an I/O bounded program, increasing and decreasing CPU speed do not make a considerable difference, because the CPU does not spend a lot of time processing and the amount of time it spend on I/O is negligible compared to the time it takes for the I/O devices to finish. Expectedly, the opposite is true for a CPU-bound program, where CPU speed has a very big impact on the time it takes to complete a program.

Git Hub link:
https://github.com/zaineb-bh/SYSC4001_A1.git