SYSC 4001 Assignment 1

Justin Morrison (101302852) - Student 1

Random Hygaard (101273397) - Student 2

Github: https://github.com/Justin-Morrison-github/SYSC4001_A1

Task 2 Report

For our simulation tests we changed various parameters in the code and ran the same 7 trace files for each case. Then we made a python script to figure out the average percent of CPU usage, IO, and overhead, and also compare the total run time of the programs. All of the execution and trace files can be found in Github. We chose to analyze the average to simplify things.

The default program contains the following parameter values:

- Data Transfer Rate = 40
- Context Switch Time = 10
- ISR Execute Time = 40
- IRET Return Time = 1
- VECTOR_SIZE = 2

Parameter	Value	Avg CPU %	Avg IO %	Avg Overhead %	Total Execution Time (ms)
Default	N/A	12.1996	82.191	5.6094	9423
DATA_TRANSFER_RATE	100	11.4302	83.3142	5.2556	10057
	200	10.3431	84.9012	4.7557	11114
CONTEXT_SAVE_TIME	20	11.9011	78.476	9.6229	9726
	30	11.413	75.2573	13.3297	10142
ISR_EXECUTE_TIME	100	10.7521	84.3041	4.9438	10691
	200	8.9769	86.8955	4.1276	12805
IRET_RETURN_TIME	10	11.9582	80.5641	7.4777	9613

	30	11.4543	77.1697	11.376	10036
VECTOR_SIZE	4	12.1996	82.191	5.6094	9423

First, the DATA_TRANSFER_RATE determines an internal ISR activity time to simulate differing complexities of ISRs. This operation occurs at every I/O SYSCALL that requires data to be read from or written to the I/O device. Since the reading or writing occurs within the ISR, increasing this parameter increases the length of each SYSCALL ISR execution and increases the IO percent.

Next, looking at the results of changing CONTEXT_SAVE_TIME to be longer. This operation happens twice every time a context switch is needed, which in our example is the SYSCALL and END_IO ISRs. So it is very costly to increase its time. This was considered part of the overhead, so predictably a longer CONTEXT_SAVE_TIME leads to much more overhead time and a longer program. This means that the time the CPU and IO were using were a smaller portion of the program.

ISR_EXECUTE_TIME is used to simulate the length of time it takes for the ISR to finish its task, such as reading information from an external device. Increasing ISR_EXECUTE_TIME had very similar effects to increasing the DATA_TRANSFER_RATE. Each time either a SYSCALL or an ENDIO command is read it must execute the corresponding ISR, if this ISR takes longer then so will the whole program. The effects of increasing this over DATA_TRANSFER_RATE were greater. This is because the DATA_TRANSFER_RATE only matters when a SYSCALL is executed, while the ISR must also be executed each ENDIO command. With the same delay between the two by increasing ISR_EXECUTE_TIME the CPU usage decreases alongside the overhead, while the share of time IO takes increases.

IRET_RETURN_TIME simulates the Interrupt Return instruction which occurs at the end of each interrupt to return to the next instruction before the interrupt occurred. In our implementation, we split the restore context and user switch into separate things, so it doesn't make a lot of sense why IRET would vary in time, but it is an interesting test case. Since it is such a low duration event, making it even 10 times longer barely had an effect on the total run time, but it did increase the overhead percentage.

The value of VECTOR_SIZE has no impact on the performance and usage of the CPU, however it did change the execution.txt file. This is because this constant declares what the size of each instruction should be. This means by doubling the vector size the expected address of the first line of the ISR is exactly twice the default. This is an important constant to have correct so that the interrupt hardware can always find the correct instruction. This would have a memory cost as now each vector is twice as large, so you would need twice as much memory to store the same vector table.

All tests were done with values greater than the default parameter values, but the trends we discussed should follow if values less than the default ones were used. For example the CPU% increasing and IO% decreasing if DATA_TRANSFER_RATE was lowered, or memory addresses being halved if VECTOR_SIZE is halved.