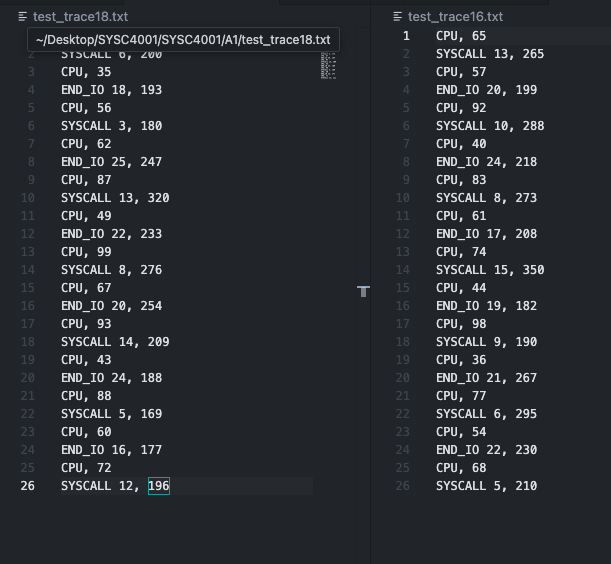
**SYSC4001 - Assignment 1 Part 2 : Interrupt Simulator**

**Authors:** Aj Donald 101259149, Jayven Larsen 101260364

**Simulation Analysis**

1. Long ISR Execution

* In the context of the simulator a longer ISR execution occurs by modifying the duration of SYSCALL or END\_IO instructions. Especially since, the typical duration for these is confined in a range of 100-400ms within the trace files that were used for testing / input. For instance, through running a variety of randomized trace files with varying executions times for SYSCALL and END\_IO, there can be large fluctuations in total elapsed time. This is true even for files of similar length and comparable number of instructions. Through the various testing of files, these examples of ISRs running long can significantly increase elapsed time of a program. A short example to visualize will follow below.



*Figure 1: Test Trace files*

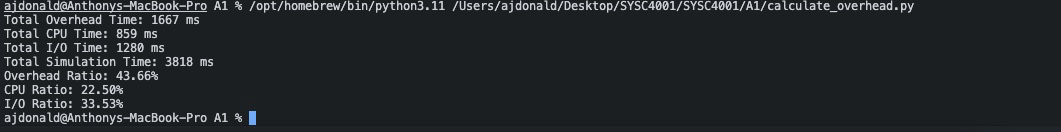
As seen in figure above, traces of identical length can have drastically different average execution time. For instance, test\_trace18 has an average execution duration of 217.84 whereas, test\_trace16 has an average execution duration of 244.23, which may not seem incredibly significant but will have a large impact with longer trace files.

2. ISR Steps

* Different steps throughout the interrupt process will be more strenuous than others. For instance, saving information in the Process Control Block or even calling and executing the scheduler will cause a variety of delays. In our context, saving or restoring the context of registers and previous states will introduce time delays. Lastly, some of the randomness associated with some of these actions can add up quickly, contributing to longer total elapsed time, especially for actions that reoccur and can vary uncontrollably (ex, context saving).

3. Overhead

* In terms of addressing the overall overhead of the simulator program, a python script was developed to read from the output (execution.txt) file and then display some key information. For instance, ratios of CPU, END\_IO or other actions and their consumption percentage.



*Figure 2: Output of calculate\_overhead.py script.*

As seen above, the python script took data from the output of the simulator, performed some brief calculation on the total amount time CPU, IO and overhead took as well as displayed a summary of the results.

4. Miscellaneous Questions

* Q: What happens if we have addresses of 4 bytes instead of 2?
* Answer: By changing the size of the addresses from 2 to 4 bytes, we can assume memory usage will increase and potentially the number of times memory is accessed. Additionally, increasing the number of bytes used for addresses means we can have a large number of event IDs.
* Q: What if we have a faster CPU and now all processing takes ½ of the time?
* Answer: When it comes to simulating a faster cpu, we can modify trace files to reduce total duration of CPU execution instructions. As a result, we can also cut the execution time of I/O events by half. Afterwards, we can analyze the results of these changes that should be reflected in the output of the simulator. All in all, these changes can help reduce overall time consumption but may not have an impact on the program overhead.