**1. What happens if the execution of the ISR is long?**

If the ISR execution gets extended, it will directly increase the total amount of time taken for the interrupt handling process. This is because by having a longer ISR, the delay before returning back to normal program execution will be greater. As a result, this means that the following interrupts must wait until the current ISR completes, leading to creating a bottleneck in systems with such interrupts that are time sensitive.

For instance, in our simulation, if the ISR is to randomly be set to a high duration such as 400-500 ms, this would cause the overhead to increase significantly, especially given there are many frequent system calls. Therefore, this delay would lead to delayed responses in real-time systems.

**Q2. How about the different steps of the ISR? (saving information in the PCB, calling the scheduler, executing the scheduler, save/restore context). How does the difference in speed in these steps affect the overall execution time of the process?**

Based on the given steps mentioned, saving information in the PCB, calling the scheduler, executing the scheduler, and save/restore context, each step in such overhead has its impact on execution time. When speaking upon:

Saving Information in the PCB (Process Control Block):

* The speed of this step will vary based on how fast the system can save the context due to acknowledging faster context-saving operations results in less overhead. In contrast, if this step is slow, the overhead increases, causing delays in executing the ISR and in returning to the interrupted process.

Calling the Scheduler:

* Knowing that after an interrupt the OS might call the scheduler to decide which process should run next this leads to a time-consuming step. As a result, if the scheduler is slow, this introduces significant delays, adding to the overall overhead of the ISR. The faster the scheduler can make its decision, the quicker the system can return to executing processes.

Executing the Scheduler:

* The efficiency of this step directly affects the response time of the system. This is due to a slower context switch resulting in longer periods of overhead. Once the scheduler selects the next process to run, it must be executed. The OS will initiate a context switch, loading the saved state of the chosen process and placing it on the CPU. Therefore, the faster the OS can execute the scheduler, the quicker the CPU can return to actual work.

Save/Restore Context:

* This is generally one of the most significant sources of overhead as this step involves saving the state of the interrupted process before the ISR starts and restoring it once the ISR finishes. If saving and restoring context takes a long time, the CPU spends more time in overhead, leading to reducing the efficiency of the system.

**3. Identify in your program ALL sources of overhead, and calculate the ratio between actual CPU use, I/O activities and overhead**

The sources of overhead present in this lab are the:

* Context saving/restoration: This outputs to 1-3 ms per interrupt.
* Switching to kernel mode: This outputs to 1 ms.
* ISR execution: This outputs to 100-400 ms however this is dependent on the specific ISR being called.
* Scheduler and PCB management: This outputs to 1-2 ms per interrupt.

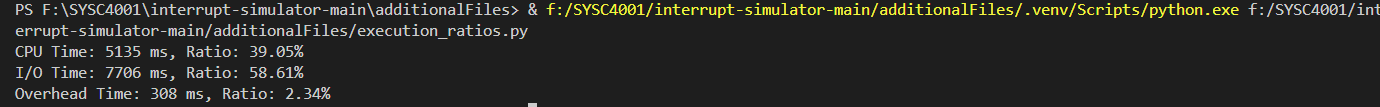
For example: Execution1.txt from trace1.txt

CPU Time: 5135 ms, Ratio: 39.05%

I/O Time: 7706 ms, Ratio: 58.61%

Overhead Time: 308 ms, Ratio: 2.34%

\*This was calculated from our short python script.



WHEN USING THE SCRIPT, SIMPLY COPY AND PASTE THE CONTENTS OF THE EXECUTION FILE YOUR LOOKING TO FIND THE RATIOS OF IN **log\_dada** VARIABLE