What Happens if the execution of the ISR is long?

The ISR is responsible for handling interrupts and can have a significant impact on the program and system performance if its execution is long. After running simulation tests and analyzing the data, it is clear that long ISR’s increase the total overhead. Longer ISR’s increase the amount of time the system spends in kernel mode handling interrupts, leading to less CPU time being available for user-mode tasks. One of the tests that was run had an overhead ratio of 67.95%, it was clear that the ISR took longer than other simulation tests, causing overhead activities that take place in kernel mode to contribute to extended overhead times.

How do the different steps of the ISR affect overall execution time?

Saving information in the PCB: This step involves saving information concerning the state of the interrupted process, which can take time. If this takes long, it leads to greater overhead time as it will delay the execution of the interrupt and slow down the system.

Calling the scheduler: The scheduler determines which process will have access to the CPU and run next, a slow scheduler will cause delays, especially if multiple processes are ready concurrently.

Executing the scheduler: If the scheduling algorithm takes an extended amount of time to determine the next process to access the CPU, this increases overhead time and takes away time from CPU usage.

Saving/Restoring context: Saving/restoring the process state before/after an interrupt can contribute to extended execution times due to the possibility of having to save or restore large amounts of data. This was seen in test9 and test18 which had overhead percentages greater than 66%.

Identifying all sources of overhead

To process the data of 20 simulation tests that were run, a python script was developed.

Overhead refers to any additional time that is not spent on CPU usage or I/O operations. This includes the following that were seen in the simulation output.

**Switching to and from kernel mode:** Each switch between user and kernel mode is not CPU usage nor is it an I/O operation, thus it is overhead. In the case of the simulation it introduces a small but negligible delay.

**Saving/restoring context:** Context switching is a source of overhead due to the time it takes to save/restore the process state, and as previously stated the size of the data concerning the state such as memory associated with the process can contribute to larger overhead times.

**ISR Execution:** The execution of the interrupt service routine includes multiple non CPU and non I/O tasks such as saving state information to the PCB, calling the scheduler, executing the scheduler and updating the interrupt controller.

**Finding vector in memory and obtaining the ISR address:** Each interrupt requires looking up the vector table in memory, then obtaining the memory address of the interrupt service routine, which adds to overhead as it takes time.

**Interrupt Return (IRET):** After the ISR is complete, the IRET returns control to the interrupted process through a branch operation to the address stored in the link register. This takes time and contributes to overhead.

Calculating ratio between actual CPU use, I/O activities, and overhead

**CPU Usage Ratio:**

Across all 20 tests, CPU usage ranges from 5.50% (test18) to 28.77% (test11). On average the CPU usage ratio was 15%, meaning that actual computation time is relatively low compared to total system time.

**I/O Activities Ratio:**

I/O activities account for between 13.27% (test14) and 33.14% (test5) of the total time, with an average of around 24%.

This indicates that while I/O tasks are significant, they are still overshadowed by overhead.

**Overhead Ratio:**

Overhead ratios range from 51.52% (test16) to 69.16% (test9), with an average overhead ratio of about 60%.

This confirms that overhead (context switching, ISR execution, and kernel/user mode switches) is the primary consumer of system resources, limiting the time available for CPU usage and I/O activities.

The following is a stacked bar chart displaying the ratio between CPU Usage, I/O Activities, and Overhead for all 20 simulation tests. As discussed and now seen in this figure, overhead makes up the majority of time spent for all simulation tests. This is largely due to the system’s heavy reliance on interrupts, which introduce overhead because the system must complete tasks such as switching to kernel mode, saving the context, executing the ISR and restoring the context.

