SYSC4001-A/L3 | Assignment #1

Gabriel Bugarija – Student ID: 101262776

Umniyah Mohammed – Student ID: 101158792

This report presents a series of simulations designed to analyze the impact of interrupt handling on CPU execution. Using a custom simulator, we examine how varying context save/restore times and ISR activity times affect the total execution time of processes.

Each simulation used one of 20 input traces and varied context save/restore times and ISR activity times. Execution logs were collected and analyzed using Python scripts to separate CPU bursts from ISR and context-switch overhead.

Simulation Analysis of a Single Input Trace

Input Trace:

CPU, 45

SYSCALL, 4

CPU, 30

END_IO, 4

SYSCALL, 12

CPU, 25

END_IO, 12

Output (Execution Log):

0, 45, CPU burst

45, 1, switch to kernel mode

46, 10, context saved

56, 1, find vector 4 in memory position 0x0008

57, 1, load address 0X0292 into the PC

58, 40, ISR body executed for device 4

98, 1, IRET

...

CPU Burst vs. Interrupt Handling

CPU bursts (first column = timestamp, second column = duration):

45 ms, 30 ms, 25 ms | Total CPU execution = 100 ms

Interrupt overhead (context save, switching, finding vector, loading PC, ISR, IRET):

- SYSCALL (device 4) = 1 + 10 + 1 + 1 + 40 + 1 = 54 ms
- END_IO (device 4) = same 54 ms
- SYSCALL (device 12) = 1 + 10 + 1 + 1 + 40 + 1 = 54 ms
- END_IO (device 12) = same 54 ms
- Total overhead = **216 ms**

The interrupt handling overhead more than doubles the CPU execution time for this trace. Even small devices with short I/O can add significant delays.

Context Save/Restore Time Effect

- Current context_save_time = 10 ms
- Every interrupt spends 10 ms saving context.
- If we increase context save time to 20 ms or 30 ms, each interrupt would increase by 10-20 ms i.e. SYSCALL for device 4 would take 64-74 ms instead of 54 ms.

Larger context save times directly increase total execution time, even if we have constant CPU bursts.

ISR Activity Time Effect

- Current ISR execution time = 40 ms
- ISR takes 40 ms, context and switch take 13 ms -> ISR is around 75% of overhead.
- Increasing ISR time to 100–200 ms:
  - Interrupt handling takes longer than the CPU bursts themselves. This causes delays in processing subsequent CPU bursts.

The longer the ISR executes, the more it delays the main process. This shows that slow device handling or heavy interrupts can reduce CPU responsivenes.

Sequence and Timing Effects

CPU start -> CPU end -> kernel switch -> context save -> find vector -> load ISR -> ISR execution -> IRET -> next CPU. The overhead is additive for multiple interrupts i.e. CPU burst of 30 ms (from 99–129) is delayed because prior interrupt handling added 54 ms.

When it comes to execution time, optimizing ISR and context save/restore times improves overall throughput.

Changing vector_size from 2 bytes to 4 bytes increases “find vector in memory”. It’s not a huge impact, but is more noticeable with a lot of interrupts.