CS2610
Lab 6 : Timer Interrupts and Context Switching

Problem 1: Timer Interrupt (40 points)

The timer interrupt in RISCV occurs when the value in *mtime* register becomes greater than the value in *mtimecmp* register (The value in *mtime* register is increasing at a constant frequency). Both the registers are memory-mapped (have a memory address) and can thus be accessed using the same instructions used for accessing memory. Write an assembly program that initializes the value of *mtimecmp* register (say, to 10000), switches to user-mode, runs a continuous loop until a timer interrupt is triggered, handles the timer interrupt in machine mode (by increasing the value in *mtimecmp* by 10000 everytime a timer interrupt occurs) and switches back to user-mode. The program should continuously switch between machine mode and user mode, triggering the timer interrupt in user mode and handling the timer interrupt in machine mode. A template for the same program is given below:

```
.section .text
.global main

main:
    #set the csrs and initialize mtimecmp
    mret

mtrap:
    #handle the timer interrupt and return to user-space

user:
    #implement a continuous loop
```

Read the values in *mtime* and *mtimecmp* registers while looping in user-space and when the timer interrupt occurs and submit the screenshots. Also note the value in *mcause*.

Note: *mtimecmp* can be accessed at the address 0x2004000 and *mtime* can be accessed at the address 0x200bff8.

Problem 2: Context Switching (60 points)

You are tasked with simulating a simple cooperative multitasking environment in RISC-V assembly language. Implement a program that involves two tasks running concurrently, and utilize a basic context switching mechanism triggered by a timer interrupt. Requirements:

1. Define two tasks, TaskA and TaskB, each containing a series of instructions. TaskA should increment the value of the register starting from 0 till it reaches 0x0fffffff and TaskB should decrement the value of the register starting from 0x03ffffff till it reaches 0. You should use the same set of registers in both tasks. When you are done with the tasks start a infinite loop and check the values of your respective registers in both the tasks.

2. Your program should start in machine mode by initializing required registers to enable timer interrupts (similar to what you did in Problem 1). It should jump to user mode and execute the TaskA.
3. Utilize a timer interrupt to trigger the context switch between tasks.
4. Implement a context-switching mechanism which saves the context of an interrupted task (like registers, PC) and switches to another task. Allocate separate stacks (in machine mode) for each task to store their respective contexts.
5. While switching back to the task make sure you are jumping to the correct address in the text section and all the relevant registers are restored to their correct values.
6. Make use of the given template file for writing your code.
7. Compile your code with the given linker descriptor file and execute it as shown below.

```
$riscv64-unknown-elf-gcc -nostartfiles -T link.ld <your_program>.s
$spike -d a.out
```

What you need to submit:

1. Code files
2. Screenshots demonstrating the context switching between the tasks and the values of registers after finishing the increment/decrement operations in respective tasks.

Note:

1. All the files should be submitted in a zipped folder.
2. The zipped folder should be named < *Roll_No* >_Lab6.zip.

Resources

https://people.eecs.berkeley.edu/~krste/papers/riscv-privileged-v1.9.pdf