CSSE 332 -- Operating Systems

Interrupts and Traps

Name: Solution Key

Question 1. (5 points) Why are timer interrupts important from an operating system's perspective?

> **Solution:** The OS will use timer interrupts to know when it is time to swap out a process in favor of another. Timers ensure that all processes get their share of the CPU, allowing us to run multiple processes and a small number of processors.

Question 2. (5 points) What are the steps taken by the processor when an exception or an interrupt occurs?

> **Solution:**
>
> 1. Record the address of the currently executing instruction in `sepc`.
> 2. Record the cause of the exception or interrupt in the `scause` register.
> 3. If needed, store the address of the faulty instruction or memory address in the `stval` register.
> 4. Jump to a specified location in the kernel that handles exceptions. If the exception occurred when in user space, that address is stored in the `stvec` register.

Question 3. (5 points) In the case of a segmentation fault, register `sepc` is used to store the address of the instruction executing when the fault occurred. The cause of the exception is stored in the `scause` register and the violating address is stored in the `stval` register. When the fault is detected, the hardware will jump to the kernel at the address stored in the `stvec` register.

Question 4. (5 points) Consider a process `P` with the following code snippet

```
1 int x, rc;
2
3 y = x + 2;
4 rc = fork();
5 /* ... */
```

`fork()` is a system call that will issue the `ecall` instruction, causing a trap into the kernel. How does the kernel ensure that the state of `P` before the call to `fork` is the same as the one after `fork` returns.

Solution: The OS will save all of the registers of the currently running process in memory so that they can later on be recovered when it is time to put the process into execution mode.