1.5. Basics of How Operating Systems Work

1.5.1. Role of Interrupts

Interrupts are signals sent to the CPU by external devices, normally I/O devices. They tell the CPU to stop its current activities and execute the appropriate part of the operating system.

There are three types of interrupts:

1. **Hardware Interupts** are generated by hardware devices to signal that they need some attention from the OS. They may have just received some data (e.g., keystrokes on the keyboard or an data on the ethernet card); or they have just completed a task which the operating system previous requested, such as transfering data between the hard drive and memory.
2. **Software Interupts** are generated by programs when they want to request a system call to be performed by the operating system.
3. **Traps** are generated by the CPU itself to indicate that some error or condition occured for which assistance from the operating system is needed.

Interrupts are important because they give the user better control over the computer. Without interrupts, a user may have to wait for a given application to have a higher priority over the CPU to be ran. This ensures that the CPU will deal with the process immediately.

1.5.2. CPU Execution Mode

There are two modes of execution, known as user mode and kernel or supervisor mode. User mode is restricted in that certain instructions cannot be executed, certain registers cannot be accessed, and I/O devices can not be accessed. Kernel mode has none of these restrictions. A system call will set the CPU to kernel mode, as will traps and interrupts. Application programs cannot do this.

Mode bit: **Supervisor** or **User** mode

- *Supervisor mode*
    - Can execute all machine instructions
    - Can reference all memory locations
- *User mode*
    - Can only execute a subset of instructions
    - Can only reference a subset of memory locations

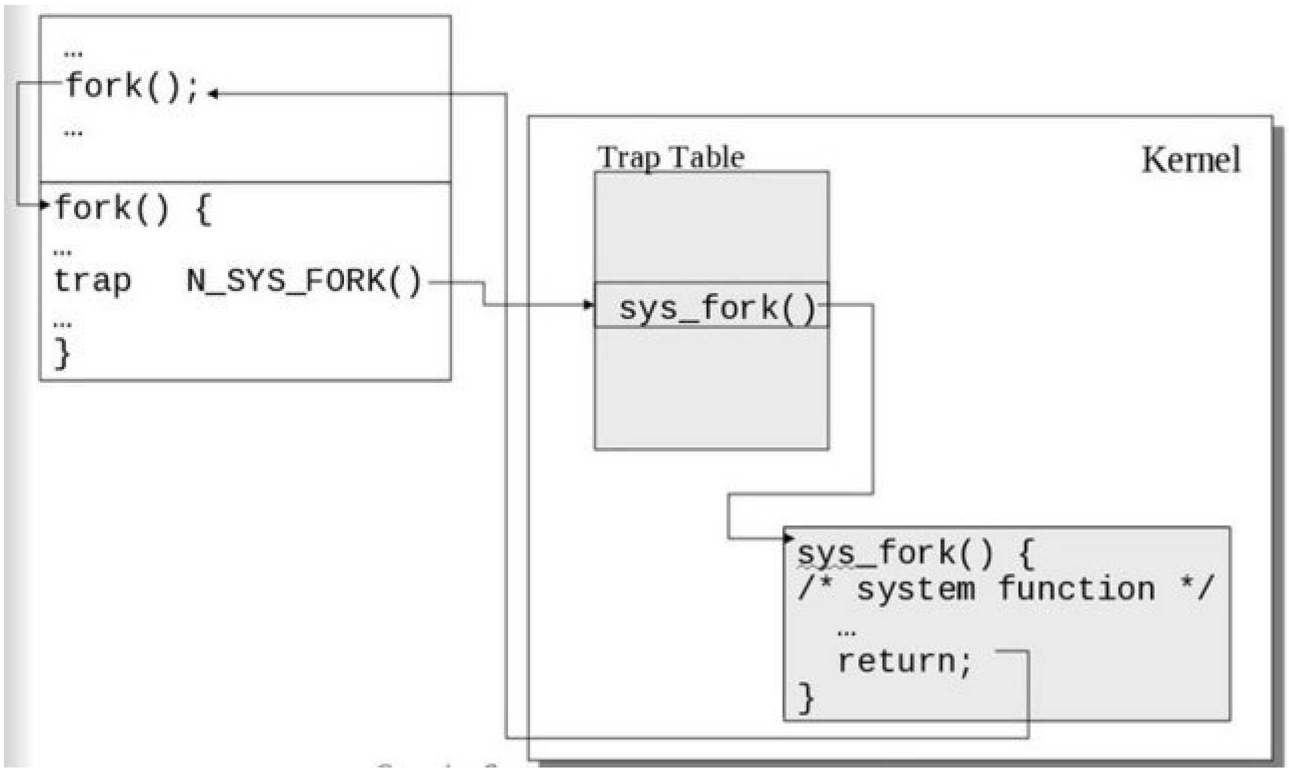
1.5.3. CPU Response to Interrupts

A key point towards understanding how operating systems work is to understand what the CPU does when an interrupt occurs. The hardware of the CPU does the exact same thing for each interrupt, which is what allows operating systems to take control away from the current running user process. The switching of running processes to execute code from the OS kernel is called a context switch.

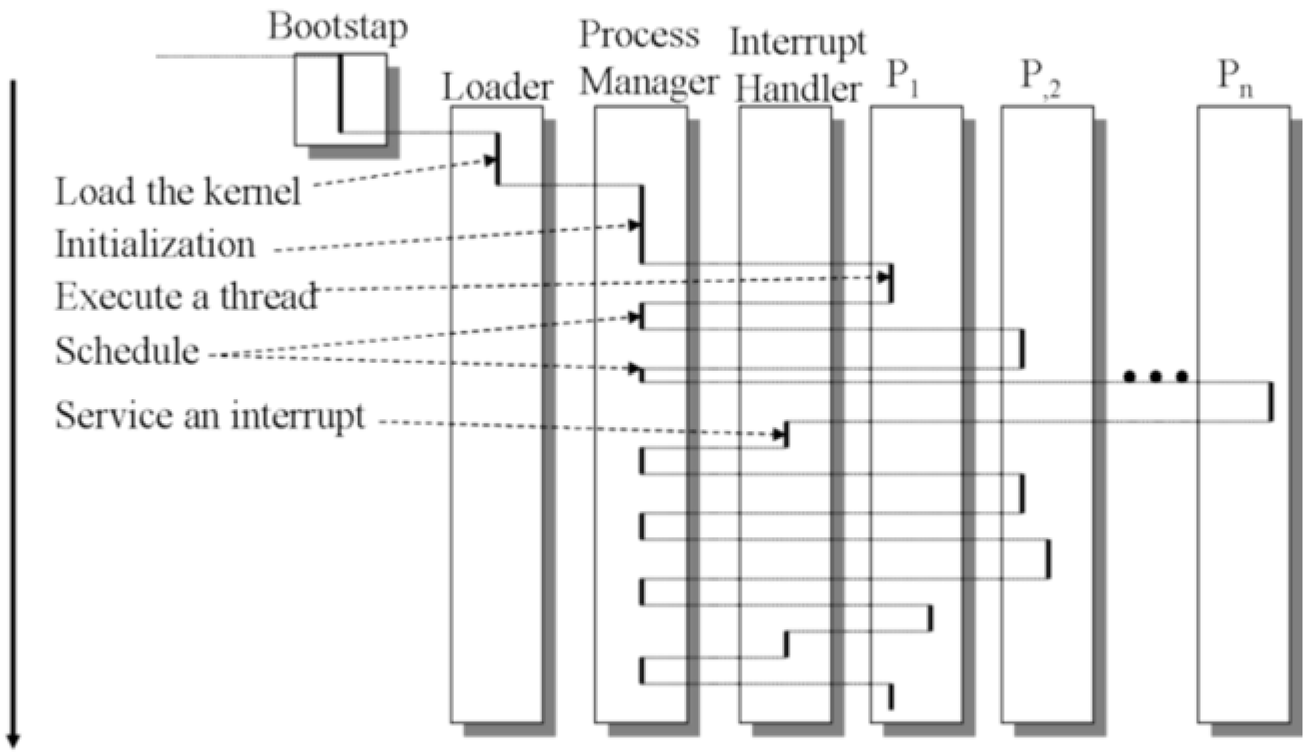
CPUs rely on the data contained in a couple registers to correctly handle interrupts. One register holds a pointer to the process control block of the current running process. This register is set each time a process is loaded into memory. The other register holds a pointer to a table containing pointers to the instructions in the OS kernel for interrupt handlers and system calls. The value in this register and contents of the table are set when the operating system is initialized at boot time.

The CPU performs the following actions in response to an interrupt:

1. Using the pointer to the current process control block, the state and all register values for the process are saved for use when the process is later restarted.
2. The CPU mode bit is switched to *supervisory* mode.
3. Using the pointer to the interrupt handler table and the interrupt vector, the location of the kernel code to execute is determined. The interrupt vector is the IRQ for hardware interrupts (read from an interrupt controller register) and an argument to the interrupt assembly language instruction for software interrupts.
4. Processing is switched to the appropriate portion of the kernel.



The CPU uses a table and the interrupt vector to find OS the code to execute in response to interrupts. A software interrupt is shown here.



As the computer runs, processing switches between user processes and the operating system as hardware and software interrupts are received.