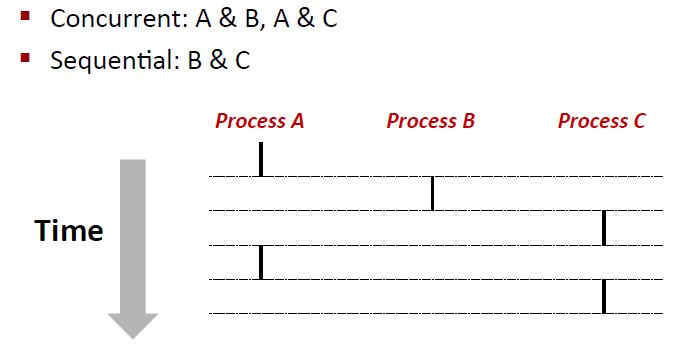
* From startup to shutdown, CPU simple reads and execute a sequence of instructions. This sequence is the CPU’s control flow.
* **Exceptional Control Flow**
  + Exists at all levels of a computer
  + Low-level Mechanisms :
    - Exceptions
  + High-level Mechanisms :
    - Process Context Switch
    - Signals
    - Non Local jumps
* **Exceptions**
  + An exception is a transfer of control to the OS kernel in response to some event
  + Exception Table
    - Each type of event has a unique exception number k
    - K = index into the exception table
    - Handler K is called each time the exception occurs
  + Types of Exceptions
    - Asynchronous Exception: caused by events external to the processor
      * Example: Timer Interrupt, I/O interrupt from an external device
    - Synchronous Exception: caused by events that occur as a result of executing an instruction
      * Example: Traps, Faults, Abort
* **System Calls**
  + Each x86-64 system call has a unique ID number
  + Example : 0 - read, 1 - write, 2 - open, 3 - close, 57 - fork, 60 - exit
* The **process** is an instance of a running program
* A process provides each program with two key abstractions

1. Logical Control Flow - provided by kernel mechanism called context switching
2. Private Address Space - provided by kernel mechanism called virtual memory

* Multi-processing
  + Single processes execute multiple processes concurrently
  + Multi-core processors are Multiple CPUs on a single chip. They share the main memory and cache memory
  + Two processes run concurrently if their flow overlaps in time. Otherwise, they are sequential



* **Context Switching**
  + In general, processes are managed by OS Kernal
  + Control flow passes from one process to another via a context switch

