**Question 4**

Xv6 manages interrupts by using a set of interrupt vectors that are needed to manage various interrupt types and are stored in the dedicated Interrupt Descriptor Table (IDT) of the x86 CPU. The x86 processor saves its current state when an interrupt happens and then jumps to the address of the relevant interrupt vector in the IDT. The address of the interrupt handler code for that specific interrupt, which is executed to handle the interrupt, is contained in the interrupt vector. The interrupt handler code takes the actions required to manage the interrupt. The interrupt handler code completes its tasks, restores the x86 processor's preserved state, and hands control back to the interrupted process.

Examples of interrupts in xv6 include the following: Timer interrupts are used to perform tasks like scheduling or handling timeouts. Keyboard interrupts are used to store keyboard input in a buffer for later processing. Network interrupts to read and store packet data in a buffer for processing.

Xv6 manages system calls by transferring control from user-level processes to kernel-level code via a software interrupt instruction. When a user-level process makes a system call, the processor saves the current state of the process and transfers control to the system call's interrupt vector address. The xv6 kernel then uses a dispatcher function to determine which system call was requested and calls the appropriate system call handler function to perform the requested operation. The handler function executes the required operations and returns any relevant data or error codes to the user-level process. After the handler function completes its task, the processor restores the user-level process's saved state and returns control to it.

Examples of system calls in xv6 include the following: File system calls are used to manage files, like open(), read(), and write(). Process Management System calls to manage processes like fork(), exec(), and wait().