INTERRUPTS

- An **interrupt** is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing.
- The processor responds by suspending its current activities, saving its state, and executing a function called an *interrupt handler* (or an interrupt service routine,ISR) to deal with the event. This interruption is temporary, and, after the interrupt handler finishes, the processor resumes normal activities.

There are two types of interrupts: hardware interrupts and software interrupts.

- **Hardware interrupts** are used by devices to communicate that they require attention from the operating system. Internally, hardware interrupts are implemented using electronic alerting signals that are sent to the processorfrom an external device, which is either a part of the computer itself, such as a disk controller, or an external peripheral.
- For example, pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position. The act of initiating a hardware interrupt is referred to as an interrupt request (IRQ).
- A **software interrupt** is caused either by an exceptional condition in theprocessor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. The former is often called a trap or exception(For example a divide- by-zero exception) and is used for errors or events occurring during program execution.
- Each interrupt has its own interrupt handler. The number of hardware interruptsis limited by the number of interrupt request (IRQ) lines to the processor, but there may be hundreds of different software interrupts. Interrupts are a commonly used technique for computer multitasking,especially in real-time computing. Such a systemis said to be interrupt-driven.

Interrupts Handling

- The ***interrupt mechanism allows devices to signal the CPU and to force execution*** of a particular piece of code.

- When an interrupt occurs, the program counter's value is changed to point toan ***interrupt handler routine (also commonly known*** as a ***device driver) that takes care of the device.***
- The interface between the CPU and I/O device includes the following signalsfor interrupting:
- The I/O device asserts the ***interrupt request signal when it wants service*** from the CPU; and
- The CPU asserts the ***interrupt acknowledge signal when it is ready to handle*** the I/O device's request.

The interrupt mechanism:

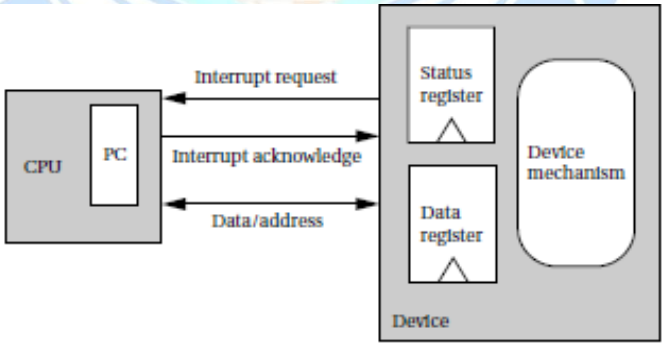

FIGURE 3.2
The interrupt mechanism.

- The *interrupt mechanism allows devices to signal the CPU and to force execution* of a particular piece of code.
- When an interrupt occurs, the program counter's value is changed to pointto an *interrupt handler routine (also commonly known* as a *device driver) thattakes care*

of the device.

- The interface between the CPU and I/O device includes the following signalsfor interrupting:
  - the I/O device asserts the *interrupt request signal when it wants service* from the CPU; and
  - The CPU asserts the *interrupt acknowledge signal when it is ready to handle* the I/O device's request.

Priorities and Vectors

- ***Interrupt priorities*** allow the CPU to recognize some interrupts as more important than others, and
- ***Interrupt vectors*** allow the interrupting device to specify its handler.

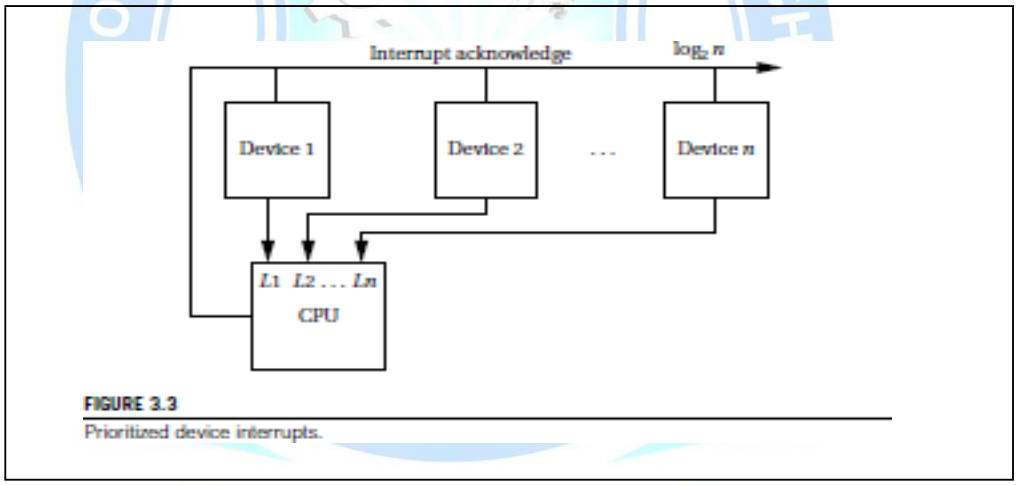

FIGURE 3.3
Prioritized device interrupts.

- The priority mechanism must ensure that a lower-priority interrupt does not occur when a higher-priority interrupt is being handled. The decision processis known as ***masking.***
- Asking an I/O device whether it is finished by reading its status register is often called ***polling.***
- The highest-priority interrupt is normally called the ***non maskable interrupt*** **(NMI).**
- Most CPUs provide a relatively small number of interrupt priority levels, such as

eight. When several devices naturally assume the same priority. You can **combine polling with prioritized interrupts** to efficiently **handle** the devices.

- Interrupt request signals

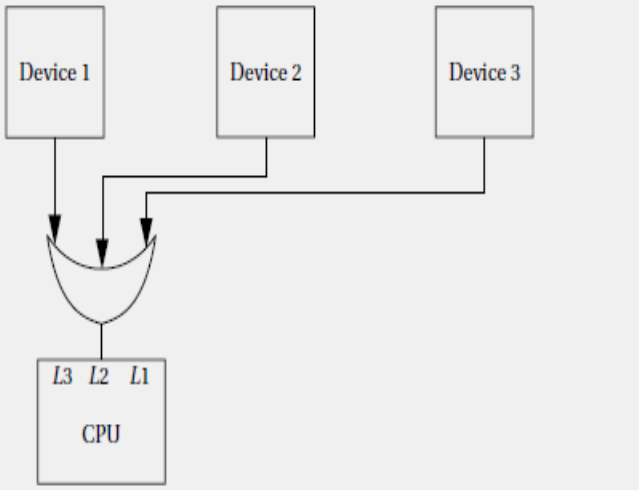

FIGURE 3.4
Using polling to share an interrupt over several devices.

- The CPU will call the interrupt handler associated with this priority; that handler does not know which of the devices actually requested the interrupt. The handler uses software polling to check the status of each device: In this example, it would read the status registers of 1, 2, and 3 to see which of them ready and requesting service is. The given example illustrates how priorities affect the order in which I/O requests are handled.

- **Vectors** provide flexibility in a different dimension, namely, the ability to definethe interrupt handler that should service a request from a device. Figure shows the hardware structure required to support interrupt vectors. In additionto the interrupt request and acknowledge lines, additional interrupt vector lines run from the devices to the CPU.

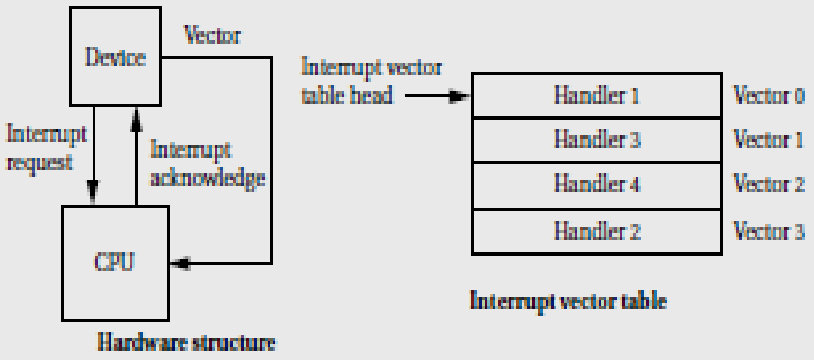

FIGURE 3.5
Interrupt vectors.

- After a device's request is acknowledged, it sends its interrupt vector over thoselines to the CPU. The CPU then uses the vector number as an index in a table stored in memory as shown in Figure 3.5. The location referenced in the interruptvector table by the vector number gives the address of the handler.
- There are two important things to notice about the interrupt vector mechanism. First, the device, not the CPU, stores its vector number. In this way, a devicecan be given a new handler simply by changing the vector number it sends, without modifying the system software. For example, vector numbers can be changed by programmable switches.
- The second thing to notice is that there is no fixed relationship between vector numbers and interrupt handlers. The interrupt vector table allows arbitrary relationships between devices and handlers. The vector mechanism provides great flexibility in the coupling of hardware devices and the software routines that service them.
- Most modern CPUs implement both prioritized and vectored interrupts. Priorities determine which device is serviced first, and vectors determine what routine is used to service the interrupt. The combination of the two provides a rich interface between hardware and software.