3.11 Vectored Interrupts

When a device interrupts, how does the hardware know the location of the code that handles the interrupt? The hardware on most processors uses a mechanism known as *vectored interrupts*. The basic idea is straightforward: each device is assigned a small integer number: 0, 1, 2, and so on. The integers are known as *interrupt level numbers* or *interrupt request numbers*. The operating system creates an array of pointers in memory known as an *interrupt vector*, where the i^{th} entry in the interrupt vector array points to the code that handles interrupts for the device with vector number *i*. When it interrupts, a device sends its vector number over the bus to the processor. Depending on the processor details, either the hardware or the operating system uses the vector number as an index into the interrupt vector, obtains a pointer, and uses the pointer as the address of the code to run.

Because it must be configured before any interrupts occur, an operating system initializes the interrupt vector at the same time devices are assigned addresses on the bus. The assignment of interrupt level numbers usually employs the same paradigm as address assignment. A manual assignment means a human assigns a unique interrupt level number to each device and then configures the interrupt vector addresses accordingly. An automatic approach requires bus and device hardware that can assign interrupt levels at runtime. To use the automatic approach, an operating system polls devices at startup, assigns a unique interrupt level number to each device, and initializes the interrupt vector accordingly. Automatic assignment is safer (i.e., less prone to human error), but requires more complex hardware in both the devices and the bus. We will see examples of static and automatic interrupt vector assignment.

3.12 Exception Vectors And Exception Processing

Many processors follow the same vectored approach for *exceptions* as they use for interrupts. That is, each exception is assigned a unique number: 0, 1, 2, and so on. When an exception occurs, the hardware places the exception number in a register. The operating system extracts the exception number, and uses the number as the index into an *exception vector*. A minor difference occurs between the way processor hardware handles interrupts and exceptions. We think of an interrupt as occurring *between* two instructions. Thus, one instruction has completed and the next instruction has not begun. However, an exception occurs *during* an instruction. Thus, when the processor returns from the exception, the program counter has not advanced, and the instruction can be restarted. Restarting is especially important for page fault exceptions — when a page fault occurs, the operating system must read the missing page from disk into memory, set the page table, and then execute the instruction that caused the fault a second time.