Communicating with the Processor

The process of periodically checking status bits to see if it is time for the next I/O operation, as in the previous example, is called *polling*. Polling is the simplest way for an I/O device to communicate with the processor. The I/O device simply puts the information in a Status register, and the processor must come and get the information. The processor is totally in control and does all the work.

The disadvantage of polling is that it can waste a lot of processor time because processors are so much faster than I/O devices. The processor may read the Status register many times, only to find that the device has not yet completed a comparatively slow I/O operation, or that the mouse has not budged since the last time it was polled. When the device has completed an operation, we must still read the status to determine whether it was successful.

Polling can be used in several different ways, depending on the I/O device and whether the I/O device can initiate I/O independently. For example, a mouse is an input-only device that initiates I/O independently, when a user moves the mouse or clicks a button. Because a mouse has a low I/O rate, polling is often used to interface to a mouse. Many other I/O devices, such as a floppy disk or a printer, initiate I/O only under control of the operating system. Thus we need only poll such devices when the OS knows that the device is active. As we will see, this allows polling to be used even when the I/O rate is somewhat higher.

Overhead of Polling in an I/O System

Example

Let's determine the impact of polling overhead for three different devices. Assume that the number of clock cycles for a polling operation—including transferring to the polling routine, accessing the device, and restarting the user program—is 400 and that the processor executes with a 500-MHz clock.

Determine the fraction of CPU time consumed for the following three cases, assuming that you poll often enough so that no data is ever lost and assuming that the devices are potentially always busy:

1. The mouse must be polled 30 times per second to ensure that we do not miss any movement made by the user.
2. The floppy disk transfers data to the processor in 16-bit units and has a data rate of 50 KB/sec. No data transfer can be missed.
3. The hard disk transfers data in four-word chunks and can transfer at 4 MB/sec. Again, no transfer can be missed.

Transferring the Data between a Device and Memory

We have seen two different methods that enable a device to communicate with the processor. These two techniques, polling and I/O interrupts, form the basis for two methods of implementing the transfer of data between the I/O device and memory. Both these techniques work best with lower-bandwidth devices, where we are more interested in reducing the cost of the device controller and interface than in providing a high-bandwidth transfer. Both polling and interrupt-driven transfers put the burden of moving data and managing the transfer on the processor. After looking at these two schemes, we will examine a scheme more suitable for higher-performance devices or collections of devices.

We can use the processor to transfer data between a device and memory based on polling. Consider our mouse example. The processor can periodically read the mouse counter values and the position of the mouse buttons. If the position of the mouse or one of its buttons has changed, the operating system can notify the program associated with interpreting the mouse changes.

An alternative mechanism is to make the transfer of data interrupt driven. In this case, the OS would still transfer data in small numbers of bytes from or to the device. But because the I/O operation is interrupt driven, the OS simply works on other tasks while data is being read from or written to the device. When the OS recognizes an interrupt from the device, it reads the status to check for errors. If there are none, the OS can supply the next piece of data, for example, by a sequence of memory-mapped writes. When the last byte of an I/O request has been transmitted and the I/O operation is completed, the OS can inform the program. The processor and OS do all the work in this process, accessing the device and memory for each data item transferred. Let's see how an interrupt-driven I/O interface might work for the floppy disk.

Overhead of Interrupt-Driven I/O

Example Suppose we have the same hard disk and processor we used in the example on page 676, but we use interrupt-driven I/O. The overhead for each transfer, including the interrupt, is 500 clock cycles. Find the fraction of the processor consumed if the hard disk is only transferring data 5% of the time.

Answer The interrupt rate when the disk is busy is the same as the polling rate. Hence,

$$\begin{aligned}\text{Cycles per second for disk} &= 250\text{K} \times 500 \\ &= 125 \times 10^6 \text{ cycles per second}\end{aligned}$$

3. Once the DMA transfer is complete, the controller interrupts the processor, which can then determine by interrogating the DMA device or examining memory whether the entire operation completed successfully.

There may be multiple DMA devices in a computer system. For example, in a system with a single processor-memory bus and multiple I/O buses, each I/O bus controller will often contain a DMA processor that handles any transfers between a device on the I/O bus and the memory. Let's see how much of the processor is consumed using DMA to handle our hard-disk example.

Overhead of I/O Using DMA

Example Suppose we have the same processor and hard disk as our earlier example on page 676. Assume that the initial setup of a DMA transfer takes 1000 clock cycles for the processor, and assume the handling of the interrupt at DMA completion requires 500 clock cycles for the processor. The hard disk has a transfer rate of 4 MB/sec and uses DMA. If the average transfer from the disk is 8 KB, what fraction of the 500-MHz processor is consumed if the disk is actively transferring 100% of the time? Ignore any impact from bus contention between the processor and DMA controller.

Answer Each DMA transfer takes

$$\frac{8 \text{ KB}}{4\dfrac{\text{MB}}{\text{second}}} = 2 \times 10^{-3} \text{ seconds}$$

So if the disk is constantly transferring, it requires

$$\frac{1000 + 500\dfrac{\text{cycles}}{\text{transfer}}}{2 \times 10^{-3}\ \dfrac{\text{seconds}}{\text{transfer}}} = 750 \times 10^{3}\,\frac{\text{clock cycles}}{\text{second}}$$

Since the processor runs at 500 MHz,

$$\text{Fraction of processor consumed} = \frac{750 \times 10^{3}}{500 \times 10^{6}}$$
$$= 1.5 \times 10^{-3} = 0.2\%$$