Input/Output

Introduction

- In addition to the processor and a set of memory modules, the third key element of a computer system is a set of I/O modules.
- Each module interfaces to the system bus or central switch and controls one or more peripheral devices.
- An I/O module is not simply a set of mechanical connectors that wire a device into the system bus. Rather, the I/O module contains logic for performing a communication function between the peripheral and the bus.

External Devices

- I/O operations are accomplished through a wide assortment of external devices that provide a means of exchanging data between the external environment and the computer.
- An external device attaches to the computer by a link to an I/O module. The link is used to exchange control, status, and data between the I/O module and the external device.
- An external device connected to an I/O module is often referred to as a peripheral device or, simply, a peripheral.

Categories of External Devices

We can broadly classify external devices into three categories:

■ **Human readable**: Suitable for communicating with the computer user;

■ **Machine readable**: Suitable for communicating with equipment;

■ **Communication:** Suitable for communicating with remote devices.

I/O Modules

Module Function The major functions or requirements for an I/O module fall into the following categories:

■ Control and timing

■ Processor communication

■ Device communication

■ Data buffering

■ Error detection

I/O Modules cont…

During any period of time, the processor may communicate with one or more external devices in unpredictable patterns, depending on the program’s need for I/O. The internal resources, such as main memory and the system bus, must be shared among a number of activities, including data I/O. Thus, the I/O function includes a control and timing requirement, to coordinate the flow of traffic between internal resources and external devices

For example, the control of the transfer of data from an external device to the processor might involve the following sequence of steps:

1. The processor interrogates the I/O module to check the status of the attached device.

2. The I/O module returns the device status.

3. If the device is operational and ready to transmit, the processor requests the transfer of data, by means of a command to the I/O module. 4. The I/O module obtains a unit of data (e.g., 8 or 16 bits) from the external device.

5. The data are transferred from the I/O module to the processor.

If the system employs a bus, then each of the interactions between the processor and the I/O module involves one or more bus arbitrations

The preceding simplified scenario also illustrates that the I/O module must communicate with the processor and with the external device. Processor communication involves the following:

■ **Command decoding:** The I/O module accepts commands from the processor, typically sent as signals on the control bus. For example, an I/O module for a disk drive might accept the following commands: READ SECTOR, WRITE SECTOR, SEEK track number, and SCAN record ID. The latter two commands each include a parameter that is sent on the data bus.

■ **Data:** Data are exchanged between the processor and the I/O module over the data bus.

■ **Status reporting:** Because peripherals are so slow, it is important to know the status of the I/O module. For example, if an I/O module is asked to send data to the processor (read), it may not be ready to do so because it is still working on the previous I/O command. This fact can be reported with a status signal. Common status signals are BUSY and READY. There may also be signals to report various error conditions.

■ **Address recognition:** Just as each word of memory has an address, so does each I/O device. Thus, an I/O module must recognize one unique address for each peripheral it controls.

Programmed I/O

Three techniques are possible for I/O operations.

- With **programmed I/O**, data are exchanged between the processor and the I/O module. The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data. When the processor issues a command to the I/O module, it must wait until the I/O operation is complete. If the processor is faster than the I/O module, this is waste of processor time.
- With **interrupt-driven I/O,** the processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work. With both programmed and interrupt I/O, the processor is responsible for extracting data from main memory for output and storing data in main memory for input.
- The alternative is known as **direct memory access (DMA).** In this mode, the I/O module and main memory exchange data directly, without processor involvement.

What is Programmed I/O?

- In this mode the data transfer is initiated by the instructions written in a computer program.
- An input instruction is required to store the data from the device to the CPU and a store instruction is required to transfer the data from the CPU to the device.
- Data transfer through this mode requires constant monitoring of the peripheral device by the CPU and also monitor the possibility of new transfer once the transfer has been initiated. Thus CPU stays in a loop until the I/O device indicates that it is ready for data transfer.
- Thus programmed I/O is a time consuming process that keeps the processor busy needlessly and leads to wastage of the CPU cycles. This can be overcome by the use of an interrupt facility. This forms the basis for the Interrupt Initiated I/O.

Advantages of Programmed I/O

- Provides simple and easy interfaces to programs.
- Direct access to interact with the CPU without involving any peripheral devices in the process.

Disadvantages of Programmed I/O

- Synchronization is not efficient in terms of the CPU time as it will need to loop waiting for the device to be ready.
- The CPU is idle and cannot handle any other task at the time it waits for the I/O device to complete processing.

What is Interrupt Initiated I/O?

- This mode uses an interrupt facility and special commands to inform the interface to issue the interrupt command when data becomes available and interface is ready for the data transfer. In the meantime CPU keeps on executing other tasks and need not check for the flag. When the flag is set, the interface is informed and an interrupt is initiated. This interrupt causes the CPU to deviate from what it is doing to respond to the I/O transfer. The CPU responds to the signal by storing the return address from the program counter (PC) into the memory stack and then branches to service that processes the I/O request. After the transfer is complete, CPU returns to the previous task it was executing. The branch address of the service can be chosen in two ways known as vectored and non-vectored interrupt. In vectored interrupt, the source that interrupts, supplies the branch information to the CPU while in case of non-vectored interrupt the branch address is assigned to a fixed location in memory.
- Interrupt-initiated I/O, also referred to as hard-initiated I/O, is where input and output operations are done when an interrupt signal is detected by the CPU.
- Interrupt-initiated I/O is a more efficient one, as the I/O device sends a signal to the CPU that it is ready for the data transfer. The CPU performs other tasks until it receives an interrupt request.

Advantages of Interrupt-Initiated I/O

- Reduced overhead on the CPU as the same CPU waits for an interrupt while doing other tasks at the same instance.
- There is also no need for monitoring the status of a specific device.
- Reduces CPU time wastage, hence enhancing the performance of the system.

Disadvantages of Interrupt-Initiated I/O

- Difficult to implement and program in contrast to sequential file processing or compared to it when it is implemented in low-level languages.
- Needs interrupt handling routines, which can be time-consuming.

Direct Memory Access (DMA)

Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module controls exchange of data between main memory and the I/O device. Because of DMA device can transfer data directly to and from memory, rather than using the CPU as an intermediary, and can thus relieve congestion on the bus. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.

- Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.
- DMA increases system concurrency by allowing the CPU to perform tasks while the DMA system transfers data via the system and memory busses. Hardware design is complicated because the DMA controller must be integrated into the system, and the system must allow the DMA controller to be a bus master. Cycle stealing may also be necessary to allow the CPU and DMA controller to share use of the memory bus.