Computer Architecture & Organization

BCA 2nd Sem. Study Material
Paper: C4T

Input Output

Anupam Pattanayak[1]
Assistant Professor,
Department of Computer Science,
Raja N. L. Khan Women's College (Autonomous),
Midnapore, West Bengal

May 3, 2020

[1]anupam.pk@gmail.com

Contents

1

Input Output

In previous two study materials, we have discussed control unit and memory unit. Hope, you have understood the concepts introduced there. In case of any problem, please send me your queries to my email[1].

In this study material, we will discuss various issues of input and output unit of a computer system. In particular, we will discuss the following:

I. Communication Bus,

II. I/O-mapped I/O and Memory-mapped I/O,

III. Parallel I/O and Serial I/O,

IV. Synchronous I/O and Asynchrnous I/O,

V. Different modes of data transfer: Programmed-I/O, Interrupt-driven I/O, and Direct memory access (DMA).

We will refer the books by M. Mano[2], by J. P. Hayes[3], and book by Patterson and Hennessy[4]. Read any standard textbook available with you for more detailed discussion.

1.1 Introduction

Input device is the device from which CPU receives input. Examples of input devices are *keyboard*, *mouse*, *web cam* or (web camera), *touch-pad* in laptop,

[1]anupam.pk@gmail.com

[2]Computer System Architecture, M. Mano, PHI Publication

[3]Computer Architecture and Organization, J P Hayes, TMH Publication,

[4]Computer Organization and Design, Patterson and Hennessy, Morgan Kaufmann Publication

joystick used in gaming console. Input is often referred as i/p which we will use mostly. Examples of output devices are *monitor*, *printer*, *speaker*. Some device can act as both input and output device such as *touch-screen* and *modem*. Output is often referred as o/p in short and we will use this notation mostly. As a whole when a general device which is either input device or output device or both is referred, then the term *I/O device* is often used. In our discussion we use this notation heavily. I/O devices are also sometimes referred as *peripheral devices*. I/O devices are indispensable unit of a computer system. Different I/O devices can be of different nature. In short, they can be of heterogenous types. CPU needs to communicate wih these heterogeneous devices seamlessly using standard bus protocols.

1.2 Bus

CPU needs communication medium to interact with the I/O devices attached with it. This communication medium can be *wired* or *wireless*. However, we will not discuss anything about wireless communiction medium here. We will describe briefly about the wired communication medium, *bus*. Bus is a group of wires that work together to act as transmission medium between different sub-systems of a computer system. Multiple devices can be attached to a bus, and a signal transmitted by a device can be received by one or more devices attached to this bus. Generally, there can be three different types of bus for carrying three different types of messages: *data bus*, *address bus*, and *control bus*. Data bus carries data CPU and memory or I/O devices. Address bus transmit address information intended for memory or I/O devices. Control bus carries control signal from CPU to peripheral devices. The more generic term *system bus* is used to refer all the three types of buses together.

There can be different type of buses that connect CPU and memory or that connects CPU and I/O devices. *Backplane bus* is a type of bus that allows CPU, memory and I/O devices to be connected together through it. Generally, *processor-memory bus* is used to interconnect CPU and memory and these support high speed data transfer than *I/O bus* that connects I/O devices. *Synchronous bus* is a kind of bus that includes a clock and has a fixed protocol for communication amongst sub-systems with respect to the clock. In *asynchronous bus*, there is no clock. Rather it uses handshaking protocol for co-ordinating the communication amongst the sub-systems.

Two main advantages of bus are *low cost* and *versatility*. One disadvantage is that it puts a limit on *I/O throughput*. Throughput is a metric that refers to the amount of work done in certain unit of time. A very important term in communication is *bandwidth* which refers to the data transfer rate. Bandwidth of the bus limits the I/O throughput. Maximum achievable bus speed is mainly limited by certain physical factors: bus length is one factor, number of devices connected is another factor. Clock skew and reflection also deters the speed limit when mant parallel wires are configured to operate at high-speed. This is why industry has transitioned mostly from parallel bus to high speed serial bus such as universal bus (USB).

There are several bus standards such as SCSI, PCI, and USB.

1.2.1 SCSI

SCSI stands for Small Computer System Interface. It is a set of standards for physical connection and data transfer between computer and I/O devices. This standard defines commands, protocols, electrical interfacing, optical interfacing and logical interfacing. SCSI uses master-slave model. One device initiates communication with another device by using commands. There are four types of commands: N (non-data), R (read), W (write), and B (bi-directional). SCSI protocol is *half-duplex*. Half-duplex means data can flow in only one direction at one time. SCSI is commonly used for hard disks and tape drives.

1.2.2 PCI

PCI stands for Peripheral Component Interconnect. This standard was developed by Intel. It has *plug-and-play* architecture which means any I/O devices can be plugged to the socket or I/O port without worrying about compatibilty or unplugged from the system interface at wish without any issue. It has higher data transfer rate. Any device here can be master and can initiate transactions. This standard allows block-wise data transfer. It's electrical specification provides low-power consumption.

1.2.3 USB

We use the term *USB* or *USB cable* commonly in our daily life at present for transfering files from smart phones to laptop/PC or such usage. USB or Universal Serial Device standard has been developed for simple and improved interface. USB interface is self-configuring. It means end users need not worry about adjusting the system parameters. From the term *serial* we can understand that USB uses serial transmission. It uses master-slave protocol. USB connectors are standardized at host device. So, any I/O device or portable memory devices can be plugged to the USB connector of the host machine. Using this standdard, small devices donot need to possess power unit. These devices get operating power from the USB interface of the host machine. USB interface is *hot-pluggable*. This means I/O peripheral devices can be attached to the system and can interacting with the system without the need of re-booting the host machine. USB provides very high speed data transfer capability. To maintain this high data transfer speed, the length of USB cable is made short. This shortened cable length is the disadvantage of USB.

1.3 I/O-mapped I/O & Memory-mapped I/O

Like CPU accesses memory content by providing memory addresses, CPU needs to assign addresses to I/O devices for accessing them. There are two approaches for addressing I/O devices. First one is to have different addressing scheme for I/O devices and use separate processor instructions like IN, OUT for reading from i/p devices and for writting to o/p devices. This approach is referred as *I/O-mapped I/O* or *peripheral-mapped I/O* or *isolated I/O*. Second approach is to treat I/O device as a memory loation. This is called *memory-mapped I/O*. Advantage of memory-mapped I/O is the uniform procedure for accessing memory location and I/O device. However, there should not be any conflict between I/O address and actual memory location. That is some address space of entire memory address range is set aside exclusively for I/O addresses. The internal hardware decoding circuit should map to particular I/O devices correctly when referring an I/O device by memory-mapped I/O. Another advantage of memory-mapped I/O is that processor instructions available for reading from memory or writing into memory and other memory related instructions can be used to interact with I/O devices in memory-mapped I/O. Generally, this topic is learnt in more depth while studying microprocessors and it's interfacing covered in some higher semester.

1.4 Parallel I/O and Serial I/O

In general, data transfer from i/p device to o/p device is done through processor. That is, processor first reads the data from i/p device and then the processor sends this data to the o/p device. Data transmission can be done in parallel mode or in serial mode. In parallel mode, data is transmitted in paralle. That is, if data bus width is 8-bit, then 8-bit data will be transmitted at the same time parallely. Whereas in serial mode, the data is transmitted serially: bit-by-bit. While discussing the bus, we have discussed the disadvantages of high speed transmission parallely. To overcome this, serial transmissions are sometimes preferred. Some I/O devices are serial in nature such as modem. Modem stands for modulator demodulator. We use modem for accessing Internet via broadband or so through telephone lines. Generally, CPU processes data word-wide and those processed data is suitable for parallel transmission. We have seen bus standards such as SCSI and PCI which are used for parallel I/O. But some devices such as modem, keyboard, mouse are generally serial device. Processor needs to have serial communication with these devices.

There are some disadvantages of parallel communication. Firstly, it is comapratively expensive as it requires more hardware with one connection per bit. Secondly, as the transfer speed increases, data loss / crosstalk / clock skew are experienced that deters reliability and performance of the communication. With respect to IC design, parallel data transfer needs more ICs pins that complicates IC design and increases cost. Although low-speed parallel data transfers can be made adequately within short distance. Today, most parallel buses are used in short communication between ICs on a PC board or through short cables within subsystems attached to motherboard. Otherwise, data transfers are generally made serially.

Serial I/O data transimission rate is expressed in terms of bits per second (bps). It has a special name, *baud rate*. Baud rate is the data transfer rate in bits per second. Suppose, a system can transmit 15 characters per second where each character is represented by 8-bits. Then, baud rate of the system is 15×8 bps $= 120$ *bps*.

1.5 Synchronous I/O and Asynchrnous I/O

Both parallel I/O and serial I/O can be classified into two categories: *synchronous* and *asynchronous*. In parallel synchronous I/O, the synchronization in the bus is done by a clock signal in the control line. But it imposes the restriction that every device must share a common clock frequency. Also, there can be clock skew problem. General, the CPU-memory data transfer bus is parallel and synchronous.

In parallel asynchronous I/O, as the devices need not share same clock frequency, wide range of devices can communicate to each other. Generally, the communicating devices uses *handshaking signals* to agreee on starting the data transfer.

The philosophy of serial synchronous I/O and serial asynchronous I/O are also the same as of parallel counterparts. In asynchronous serial I/O, data is sent in a group of one character at a time usually. There are some overheads for every character such as use of *start bits*, *stop bits*, *parity bit*. Communicating device agree on number of characters to place between start bits and stop bits. Start/stop bits are a group of consecutive bits that are dedicatedly used to indicate start of communication and end of communication. Sunchronous serial I/O reduces the overhead of start bits and stop bits for every character by merging several characters into one frame which is placed between header of the frame and at the tail of the frame. These header and tail field contain information like start bits, stop bits, error detecting code like parity bit etc. Many of these concepts you will come across in depth when you study data link layer protocols in computer network in higher semester.

1.6 Different Modes of Data Transfer

Having seen the basics of bus and different types of data transmission, now we are ready to see different modes of data transfer between CPU and I/O devices. There are mainly three modes of data transfer:

I. Programmed-I/O,

II. Interrupt-driven I/O, and

III. Direct Memory Access (DMA).

1.6.1 Programmed I/O

Here, the transfer between CPU and I/O devices takes place under constant supervision of CPU through programs. CPU must check periodically to see if there is any data for input/output. If I/O-mapped I/O is used, then CPU uses I/O instructions like IN or OUT. If CPU finds that there is a data from input device to be sent to an output device then it first reads this data using IN instruction, 1-word at a time, places the data into accummulator register, and then sends the content to output device by using OUT instruction. Similar steps are used when memory-mapped I/O is used.

So, one disadvatnage is that CPU needs to periodically check if there is any data for performing I/O. This wastes some precious processor cycles. Even if there is no data for I/O, CPU needs to examine if there is any I/O data in regular interval. To avoid wasting the valuable processor cycle, an alternative approach for I/O data transfer mode is introduced next.

1.6.2 Interrupt-Driven I/O

In this mode of data transfer, CPU does not need to check periodically for any I/O data. When any I/O device has data to be serviced by CPU then it raises an *interrupt* for the CPU. Upon receiving the interrrupt, CPU completes the presently executing instruction and then serves the interrupt request. It stops executing further instruction, saves the content of PC and processor status word (PSW) in stack and executes the *interrupt service routine* (ISR) corresponding to the interrrupt raised by the I/O device. After completion of ISR execution, CPU restores its state by re-initializing its PC and PSW to the state it was before executing the ISR and resumes the normal program execution.

However, there is one issue that has to be adressed. What happens if there is data for I/O from multiple devices at the same time. CPU should serve request of which device? So, to resolve this issue, CPU should assign priorities to the I/O devices. This can be achieved in two ways: first one is by using software approach, known as *polling*. Second approach uses hardware circuit to resolve the priority, known as *daisy-chaining*.

In polling, a common branching address is assigned to all the interrupts. The ISR code then *polls* every possible source of interrupt in order of priority of the I/O devices. This sequence of checking interrupt source establishes the level of priority of the I/O device.

In daisy-chaining, priority of I/O devices is established by a serial connection to all the devices. CPU checks the devices one-by-one in order of their position in the chain of serial connection. The highest priority device is placed at the first probing point in the serial connection, second-highest priority device is placed next in the serial connection, and so on.

1.6.3 Direct Memory Access

So far we have seen two modes of data transfer: programmed-I/O and interrupt-driven I/O. These are O.K. for working with low-bandwidth devices. But for I/O devices to access a block of memory, we need some alternative approach. Because still, if we use the above mentioned approaches, then we need to transfer data through CPU. Every word of data from i/p device is sent to memory through CPU register. Or, every word of memory is sent to o/p device through CPU register. This makes the data transfer process between memory and I/O device slower. Even, if it is required to copy a block of data consisting of multiple words from one part of memory to another part of memory, this intermediate role of CPU degrades the data transfer rate. To address this, direct memory access (DMA) is used.

In DMA, when the CPU receives DMA request from the I/O device, CPU completes it's currently executing instruction, saves it's current state and then CPU relinquishes the control of system bus to the requesting device. The DMA requesting device now becomes master of system bus. It places the address of memory location which is source/destiation of data and number of bytes to be transferred in approapriate registers. Generally, a special I/O controller device called DMA controller takes the responsibility of implemting the DMA operation and reduces DMA operation overhead burden from I/O devices. Once the data transfer begins, DMA controller keeps track of number of bytes remaining for data transfer. When the block of data transfer is finished, then DMA controller gives back the control of system bus to the CPU. This mode of DMA operation is called *burst transfer mode* where entire block of memory words transferred continuously. Another mode of DMA operation is *cycle stealing mode* where CPU does not need to wait for entire data block to be traansferred. In this mode, DMA controller *steals* one CPU cylce, gets control of system bus and transfers one memory word/byte and then returns back control to the CPU. DMA controller continues this untill the entire memory block gets read/written.