# 01 - Introduction, Means of I/O

**What is Operating Systems?**

Computer programming can be categorized into two categories .i.e.

1

While designing software the programmer may determine the required inputs for that program, the wanted outputs and the processing the software would perform in order to give those wanted outputs. The implementation of the processing part is associated with application programming. Application programming facilitates the implementation of the required processing that software is supposed to perform; everything that is left now is facilitated by system programming.

Systems programming is the study of techniques that facilitate the acquisition of data from input devices, these techniques also facilitates the output of data which may be the result of processing performed by an application.

**Three Layered Approach**

A system programmer may use a three layered approach for systems programming. As you can see in the figure the user may directly access the programmable hardware in order to perform I/O operations. The user may use the trivial BIOS (Basic Input Output System) routines in order to perform I/O in which case the programmer need not know the internal working of the hardware and need only the knowledge BIOS routines and their parameters. 2

In this case the BIOS programs the hardware for required I/O operation which is hidden to the user. In the third case the programmer may invoke operating systems (DOS or whatever) routines in order to perform I/O operations. The operating system in turn will use BIOS routines or may program the hardware directly in order to perform the operation.

Methods of I/O

In the three layered approach if we are following the first approach we need to program the hardware. The hardware can be programmed to perform I/O in three ways i.e.

* Programmed I/O
* Interrupt driven I/O
* Direct Memory Access

In case of programmed I/O the CPU continuously checks the I/O device if the I/O operation can be performed or not. If the I/O operations can be performed the CPU performs the computations required to complete the I/O operation and then again starts waiting for the I/O device to be able to perform next I/O operation. In this way the CPU remains tied up and is not doing anything else besides waiting for the I/O device to be idle and performing computations only for the slower I/O device.

In case of interrupt driven the flaws of programmed driven I/O are rectified. The processor does not check the I/O device for the capability of performing I/O operation rather the I/O device informs the CPU that it’s idle and it can perform I/O operation, as a result the execution of CPU is interrupted and an Interrupt Service Routine (ISR) is invoked which performs the computations required for I/O operation. After the execution of ISR the CPU continues with whatever it was doing before the interruption for I/O operation. In this way the CPU does not remain tied up and can perform computations for other processes while the I/O devices are busy performing I/O and hence is more optimal.

Usually it takes two bus cycles to transfer data from some I/O port to memory or vice versa if this is done via some processor register. This transfer time can be reduced bypassing the CPU as ports and memory device are also interconnected by system bus. This is done with the support of DMA controller. The DMA (direct memory access) controller can controller the buses and hence the CPU can be bypassed data item can be transferred from memory to ports or vice versa in a single bus cycle.

**I/O controllers**

3

No I/O device is directly connected to the CPU. To provide control signals to the I/O device a I/O controller is required. I/O controller is located between the CPU and the I/O device. For example the monitor is not directly collected to the CPU rather the monitor is connected to a VGA card and this VGA card is in turn connected to the CPU through busses. The keyboard is not directly connected to CPU rather its connected to a keyboard controller and the keyboard controller is connected to the CPU. The function of this I/O controller is to provide

* I/O control signals
* Buffering
* Error Correction and Detection

We shall discuss various such I/O controllers interfaced with CPU and also the techniques and rules by which they can be programmed to perform the required I/O operation.

Some of such controllers are

* DMA controller
* Interrupt controller
* Programmable Peripheral Interface (PPI)
* Interval Timer
* Universal Asynchronous Receiver Transmitter

We shall discuss all of them in detail and how they can be used to perform I/O operations.

**Operating systems**

Systems programming is not just the study of programmable hardware devices. To develop effective system software one needs to the internals of the operating system as well. Operating systems make use of some data structures or tables for management of computer resources. We will take up different functions of the operating systems and discuss how they are performed and how can the data structures used for these operations be accessed.

**File Management**

File management is an important function of the operating systems. DOS/Windows uses various data structures for this purpose. We will see how it performs I/O management and how the data structures used for this purpose can be directly accessed. The various data structures are popularly known as FAT which can be of 12, 16 and 32 bit wide, Other data structures include BPB(BIOS parameter block), DPB( drive parameter block) and the FCBs(file control block) which collectively forms the directory structure. To understand the file structure the basic requirement is the understanding of the disk architecture, the disk formatting process and how this process divides the disk into sectors and clusters.

**Memory management**

Memory management is another important aspect of operating systems. Standard PC operate in two mode in terms of memory which are

* Real Mode
* Protected Mode

In real mode the processor can access only first one MB of memory to control the memory within this range the DOS operating system makes use of some data structures called

* FCB (File control block )
* PSP (Program segment prefix)

We shall discuss how these data structures can be directly accessed, what is the significance of data in these data structures. This information can be used to traverse through the memory occupied by the processes and also calculate the total amount of free memory available.

Certain operating systems operate in protected mode. In protected mode all of the memory interfaced with the processor can be accessed. Operating systems in this mode make use of various data structures for memory management which are

* Local Descriptor Table
* Global Descriptor Table
* Interrupt Descriptor Table

We will discuss the significance these data structures and the information stored in them. Also we will see how the logical addresses can be translated into physical addresses using the information these tables

**Viruses and Vaccines**

Once an understanding of the file system and the memory Management is developed it is possible to understand the working of viruses. Virus is a simple program which can embed itself within the computer resources and propagate itself. Mostly viruses when activated would perform something hazardous.

We will see where do they embed themselves and how can they be detected. Moreover we will discuss techniques of how they can be removed and mostly importantly prevented to perform any infections.

There are various types of viruses but we will discuss those which embed themselves within the program or executable code which are

Executable file viruses

Partition Table or boot sector viruses

**Device Drivers**

Just connecting a device to the PC will not make it work unless its device drivers are not installed. This is so important because a device driver contains the routines which perform I/O operations on the device. Unless these routines are provided no I/O operation on the I/O device can be performed by any application.

We will discuss the integrated environment for the development of device drivers for DOS and Windows.

We shall begin our discussion from means of I/O. On a well designed device it is possible to perform I/O operations from three different methods

* Programmed I/O
* Interrupt driven I/O
* DMA driven I/O

5

In case of programmed I/O the CPU is in a constant loop checking for an I/O opportunity and when its available it performs the computations operations required for the I/O operations. As the I/O devices are generally slower than the CPU, CPU has to wait for I/O operation to complete so that next data item can be sent to the device. The CPU sends data on the data lines. The device need to be signaled that the data has been sent this is done with the help of STROBE signal. An electrical pulse is sent to the device by turning this signal to 0 and then 1. The device on getting the strobe signal receives the data and starts its output. While the device is performing the output it’s busy and cannot accept any further data on the other and CPU is a lot faster device and can process lot more bytes during the output of previously sent data so it should be synchronized with the slower I/O device. This is usually done by another feed back signal of BUSY which is kept active as long as the device is busy. So the CPU is only waiting for the device to get idle by checking the BUSY signal as long as the device is busy and when the device gets idle the CPU will compute the next data item and send it to the device for I/O operation.

Similar is the case of input, the CPU has to check the DR (data Ready) signal to see if data is available for input and when its not CPU is busy waiting for it.

**Interrupt Driven I/O**

Picture15

The main disadvantage of programmed I/O as can be noticed is that the CPU is busy waiting for an I/O opportunity and as a result remain tied up for that I/O operation. This disadvantage can be overcome by means of interrupt driven I/O. In Programmed I/O CPU itself checks for an I/O opportunity but in case of interrupt driven I/O the I/O controller interrupts the execution of CPU when ever and I/O operation is required for the computation of the required I/O operation. This way the CPU can perform other computation and interrupted to perform and interrupt service routine only when an I/O operation is required, which is quite an optimal technique.

**DMA driven I/O**

Picture12

In case data is needed to transferred from main memory to I/O port this can be done using CPU which will consume 2 bus cycles for a single word, one bus cycle from memory to CPU and other from CPU to I/O port in case of output and the vice versa in case of input. In case no computation on data is required CPU can be bypassed and another device DMA (direct memory access) controller can be used. Its possible to transfer a data word directly from memory to CPU and vice versa in a single bus cycle using the DMA, this technique is definitely faster.

We shall start our discussion with the study of interrupt and the techniques used to program them. We will discuss other methods of I/O as required.

What are interrupts?

6

Literally to interrupt means to break the continuity of some on going task. When we talk of computer interrupt we mean exactly the same in terms of the processor. When an interrupt occurs the continuity of the processor is broken and the execution branches to an interrupt service routine. This interrupt service routine is a set of instruction carried out by the CPU to perform or initiate an I/O operation generally. When the routine is over the execution of the CPU returns to the point of interruption and continues with the on going process.

Interrupts can be of two types

* Hardware interrupts
* Software interrupts

Only difference between them is the method by which they are invoked. Software interrupts are invoked by means of some software instruction or statement and hardware interrupt is invoked by means of some hardware controller generally.

**Interrupt Mechanism**

Interrupts are quite similar to procedures or function because it is also another form temporary execution transfer, but there some differences as well. Note that when procedures are invoked by there names which represents their addresses is specified whereas in case of interrupts their number is specified. This number can be any 8 bit value which certainly is not its address. So the first question is what is the significance of this number? Another thing should also be noticed that procedures are part of the program but the interrupts invoked in the program are no where declared in the program. So the next question is where do these interrupts reside in memory and if they reside in memory then what would be the address of the interrupt?

Firstly lets see where do interrupts reside. Interrupts certainly reside somewhere in memory, the interrupts supported by the operating system resides in kernel which you already know is the core part of the operating system. In case of DOS the kernel is io.sys which loads in memory at boot time and in case of windows the kernel is kernel32.dll or kernel.dll. these files contain most of the I/O routines and are loaded as required. The interrupts supported by the ROM BIOS are loaded in ROM part of the main memory which usually starts at the address F000:0000H. Moreover it is possible that some device drivers have been installed these device drivers may provide some I/O routines so when the system boots these I/O routines get memory resident at interrupt service routines. So these are the three possibilities.

Secondly a program at compile time does not know the exact address where the interrupt service routine will be residing in memory so the loader cannot assign addresses for interrupt invocations. When a device driver loads in memory it places the address of the services provided by itself in the interrupt vector table. Interrupt Vector Table (IVT) in short is a 1024 bytes sized table which can hold 256 far addresses as each far address occupies 4 bytes. So its possible to store the addresses of 256 interrupts hence there are a maximum of 256 interrupt in a standard PC. The interrupt number is used as an index into the table to get the address of the interrupt service routine.