Computer Organization and Architecture

Lecture notes

SHAMBHUNATH
Group of Institutions
... Shaping the future

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Unit -5

Input / Output:

Prepared by: Rajan Mani Tripathi

Input/output Subsystem

The I/O subsystem of a computer provides an efficient mode of communication between the central system and the outside environment. It handles all the input-output operations of the computer system.

Peripheral Devices

Input or output devices that are connected to computer are called **peripheral devices**. These devices are designed to read information into or out of the memory unit upon command from the CPU and are considered to be the part of computer system. These devices are also called **peripherals**.

For example: *Keyboards*, *display units* and *printers* are common peripheral devices.

There are three types of peripherals:

1. **Input peripherals**: Allows user input, from the outside world to the computer. Example: Keyboard, Mouse etc.
2. **Output peripherals**: Allows information output, from the computer to the outside world. Example: Printer, Monitor etc.
3. **Input-Output peripherals**: Allows both input (from outside world to computer) as well as, output (from computer to the outside world). Example: Touch screen etc.

Interfaces

Interface is a shared boundary between two separate components of the computer system which can be used to attach two or more components to the system for communication purposes.

There are two types of interface:

1. CPU Interface
2. I/O Interface

Let's understand the I/O Interface in detail,

Input-Output Interface

Input-output interface provides a method for transferring information between internal storage and external I/0 devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit. The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral. The major differences are:

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.

2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be needed.

3. Data codes and formats in peripherals differ from the word format in the CPU and memory.

4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface between the processor bus and the peripheral device.

Modes of Transfer

Binary information received from an external device is usually stored in memory for later processing. Information transferred from the central computer into an external device originates in the memory unit. The CPU merely executes the 110 instructions and may accept the data temporarily, but the ultimate source or destination is the memory unit. Data transfer between the central computer and 110 devices may be handled in a variety of modes. Some modes use the CPU as an intermediate path; others transfer the data directly to and from the memory unit. Data transfer to and from peripherals may be handled in one of three possible modes:

1. Programmed I/O

2. Interrupt-initiated I/O

3. Direct memory access (DMA)

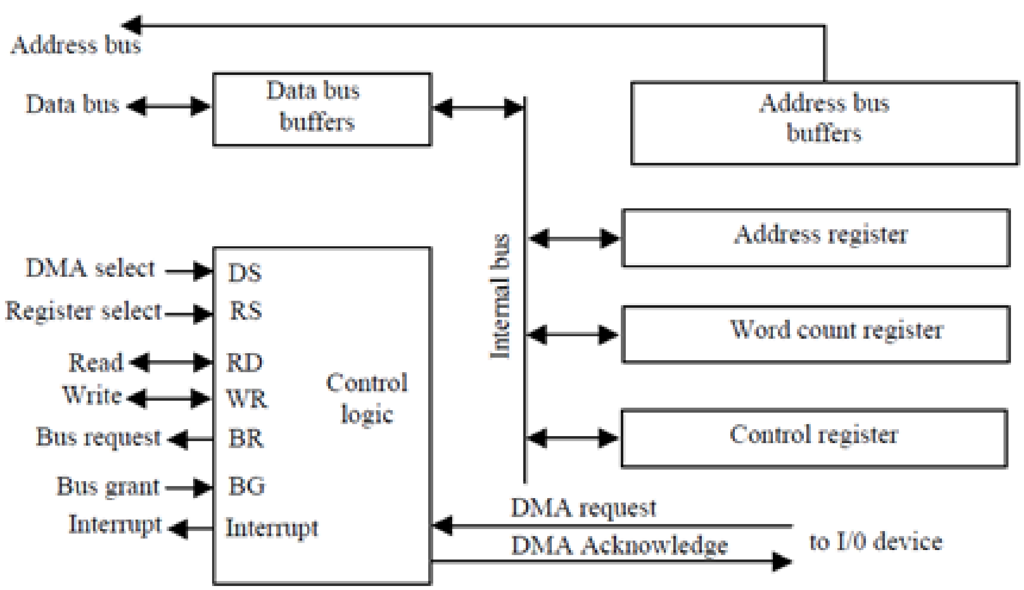
Programmed I/O operations are the result of I/O instructions written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually, the transfer is to and from a CPU register and peripheral. Other instructions are needed to transfer the data to and from CPU and memory. Transferring data under program control requires constant mon-itoring of the peripheral by the CPU. Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made. It is up to the programmed instructions executed in the CPU to keep close tabs on everything that is taking place in the interface unit and the I/O device.

In the programmed 1/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time-consuming process since it keeps the processor busy needlessly. It can be avoided by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device. In the meantime the CPU can proceed to execute another program. The interface meanwhile keeps monitoring the device. When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer. Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing.

Direct Memory Access (DMA)

The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called direct memory access (DMA). During DMA transfer, CPU is idle and has no control of the memory buses. A DMA controller take over the buses to manage thee transfer directly between the I/O device and memory.

DMA Controller:



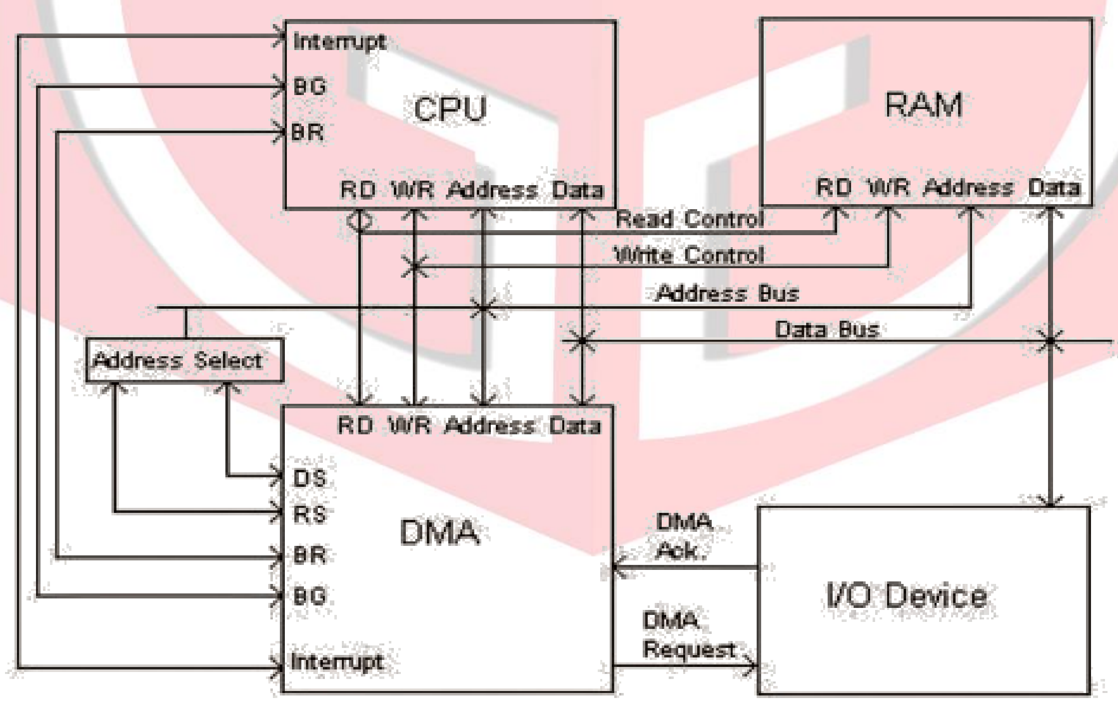
The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device. In addition, it needs an address register, a word count register, and a set of address lines. The address register and address lines are used for direct communication with the memory the word count register specifies the number of words that must be transferred. The data transfer may be done directly between the device and memory under control of the DMA. The block diagram of a typical DMA controller. The unit communicates with the CPU via the data bus and control lines. The register in the

DMA are selected by the CPU through the address bus by enabling the DS (DMA select) and RS (register select inputs.)

The RD (read) and WR (write) inputs are bidirectional. When the BG (bus grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers. When BG=1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control. The DMA communicates with the external peripheral through the request and acknowledge line by using a prescribed handshaking procedure.

The DMA controller has three registers: an address register, a word count register, and a control register, the address register contains an address to specify the desired location in memory. The address bits go through bus to specify the desired location in memory. The address bits go through bus buffers into the address bus. The address register is incremented after each word that is transferred to memory. The word count register holds the number of words to be transferred. This register is decremented by one after each word transfer and internally tested for zero. The control register specifies the mode of transfer. All register in the DMA appear to the CPU as i/o interface registers. Thus, the CPU CAN reads from or write into the DMA registers under program control via the data bus.

DMA TRANSFER



DMA Transfer

The position of the DMA controller among the other components in a computer system is illustrated in Fig The CPU communicates with the DMA through the address and data buses as with any interface unit. The DMA has its own address, which activates the *DS* and *RS* lines. The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory.

When the peripheral device sends a DMA request, the DMA controller activates the *BR* line, informing the CPU to relinquish the buses. The CPU responds with its *BG* line, informing the DMA that its buses are disabled. The DMA then puts the current value of its address register into the address bus, initiates the *RD* or *WR* signal, and sends a DMA acknowledge to the peripheral device. Note that the *RD* and *WR* lines in the DMA controller are bidirectional. The direction of transfer depends on the status of the *BG* line. When $BG = 0$, the *RD* and *WR* are input lines allowing the CPU to communicate with the internal DMA registers. When $BG = 1$, the *RD* and *WR* are output lines from the DMA controller to the random-access memory to specify the read or write operation for the data.

When the peripheral device receives a DMA acknowledge, it puts a word in the data bus (for write) or receives a word from the data bus (for read). Thus the DMA controls the read or write operations and supplies the address for the memory. The peripheral unit can then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled.

For each word that is transferred, the DMA increments its address register and decrements its word count register. If the word count does not reach zero, the DMA checks the request line coming from the peripheral. For a high-speed device, the line will be active as soon as the previous transfer is completed. A second transfer is then initiated, and the process continues until the entire block is transferred. If the peripheral speed is slower, the DMA request line may come somewhat later. In this case the DMA disables the bus request line so that the CPU can continue to execute its program. When the peripheral requests a transfer, the DMA requests the buses again.

If the word count register reaches zero, the DMA stops any further transfer and removes its bus request. It also informs the CPU of the termination by means of interrupt.

Interrupt

Interrupt is a signal which has highest priority from hardware or software which processor should process its signal immediately.

Types of Interrupts:

Although interrupts have highest priority than other signals, there are many types of interrupts, but basic type of interrupts are

1. **Hardware Interrupts:** If the signal for the processor is from external device or hardware is called hardware interrupts. Example: from keyboard we will press the key to do some action this pressing of key in keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts. Hardware interrupts can be classified into two types they are
    - **Maskable Interrupt:** The hardware interrupts which can be delayed when a much highest priority interrupt has occurred to the processor.
    - **Non Maskable Interrupt:** The hardware which cannot be delayed and should process by the processor immediately.
2. **Software Interrupts:** Software interrupt can also divide in to two types. They are
    - **Normal Interrupts:** the interrupts which are caused by the software instructions are called software instructions.
    - **Exception:** unplanned interrupts while executing a program is called Exception. For example: while executing a program if we got a value which should be divided by zero is called a exception.

Interrupt Handling:

We know that instruction cycle consists of fetch, decode, execute and read/write functions. After every instruction cycle the processor will check for interrupts to be processed if there is no interrupt is present in the system it will go for the next instruction cycle which is given by the instruction register.

If there is an interrupt present then it will trigger the interrupt handler, the handler will stop the present instruction which is processing and save its configuration in a register and load the program counter of the interrupt from a location which is given by the interrupt vector table. After processing the interrupt by the processor interrupt handler will load the instruction and its configuration from the saved register, process will start its processing where it's left. This saving the old instruction processing configuration and loading the new interrupt configuration is also called as context switching.

Input / Output: Peripheral devices, I/O interface, I/O ports, Interrupt type of interrupt and exceptions. Modes of Data Transfer: Program Direct Memory Access., I/O channels and processors. Serial asynchronous communication, standard communication interfaces.