Unit – 6

Input/output Organization

Introduction to Bus Architecture:

What is Computer Bus?

- The electrically conducting path along which data is transmitted inside any digital electronic device.
- A **Computer bus** consists of a set of parallel conductors, which may be conventional wires, copper tracks on a PRINTED CIRCUIT BOARD, or microscopic aluminum trails on the surface of a silicon chip.
- A **bus** is a **common pathway** through which information flows from one computer component to another. This pathway is used for communication purpose and it is established between two or more computer components. We are going to check different **computer bus architectures** that are found in computers.
- Each wire carries just one bit, so the number of wires determines the largest data WORD the bus can transmit: a bus with eight wires can carry only 8-bit data words, and hence defines the device as an 8-bit device.
- A bus is a subsystem that is used to connect computer components and transfer data between them. For example, an internal bus connects computer internals to the motherboard.
- A bus may be parallel or serial. Parallel buses transmit data across multiple wires. Serial buses transmit data in bit-serial format.
- A bus was originally an electrical parallel structure with conductors connected with identical or similar CPU pins, such as a 32-bit bus with 32 wires and 32 pins.
- The earliest buses, often termed electrical power buses or bus bars, were wire collections that connected peripheral devices and memory,

with one bus designated for peripheral devices and another bus for memory.

- Each bus included separate instructions and distinct protocols and timing.
- Parallel bus standards include advanced technology attachment (ATA) or small computer system interface (SCSI) for printer or hard drive devices.

- Serial bus standards include universal serial bus (USB), FireWire or serial ATA with a daisy-chain topology or hub design for devices, keyboards or modem devices.

- A *computer bus* normally has a single word memory **circuit called a LATCH** attached to either end, which briefly stores the word being transmitted and ensures that each bit has settled to its intended state before its value is transmitted.
- *The **Computer bus** helps the various parts of the PC communicate*. If there was no bus, you would have an unwieldy number of wires connecting every part to every other part. It would be like having separate wiring for every light bulb and socket in your house.

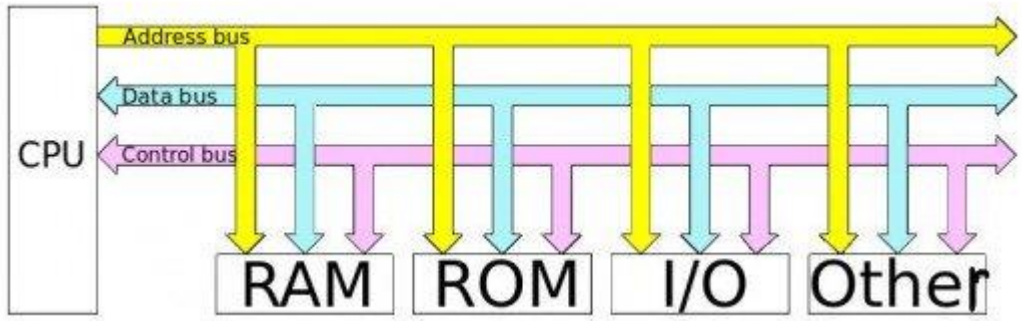
Types of Computer Bus

There are a variety of buses found inside the computer:

- **Data Bus**: The data bus allows data to travel back and forth between the microprocessor (CPU) and memory (RAM).
- **Address Bus**: The address bus carries information about the location of data in memory.
- **Control Bus:** The control bus carries the control signals that make sure everything is flowing smoothly from place to place.

- **Expansion Bus:** If your computer has expansion slots, there's an **expansion bus.** Messages and information pass between your computer and the add-in boards you plug in over the expansion bus.

Different Types of Computer Buses



Functions of Buses in Computers

1. Data sharing - All types of buses found in a computer transfer data between the computer peripherals connected to it.

The buses transfer or send data either in the serial or parallel method of data transfer. This allows for the exchange of 1, 2, 4 or even 8 bytes of data at a time. (A byte is a group of 8 bits). Buses are classified depending on how many bits they can move at the same time, which means that we have 8-bit, 16-bit, 32-bit or even 64-bit buses.

2. Addressing - A bus has address lines, which match those of the processor. This allows data to be sent to or from specific memory locations.

3. Power - A bus supplies power to various peripherals connected to it.

4. Timing - The bus provides a **system clock** signal to synchronize the peripherals attached to it with the rest of the system.

The expansion bus facilitates easy connection of more or additional components and devices on a computer such as a TV card or sound card.

Bus Terminologies

Computers have two major types of buses:

1. System bus:- This is the bus that connects the CPU to the main memory on the motherboard. The system bus is also called the front-side bus, memory bus, local bus, or host bus.

2. A number of I/O Buses, (I/O is an acronym for input/output), connecting various peripheral devices to the CPU. These devices connect to the system bus via a 'bridge' implemented in the processors' chipset. Other names for the I/O bus include "expansion bus", "external bus" or "host bus".

Expansion Bus Types

These are some of the common expansion bus types that have ever been used in computers:

- **ISA -** Industry Standard Architecture
- **EISA -** Extended Industry Standard Architecture
- **MCA -** Micro Channel Architecture
- **VESA -** Video Electronics Standards Association
- **PCI -** Peripheral Component Interconnect
- **PCI Express** (PCI-X)
- **PCMCIA -** Personal Computer Memory Card Industry Association (Also called PC bus)
- **AGP -** Accelerated Graphics Port
- **SCSI -** Small Computer Systems Interface.

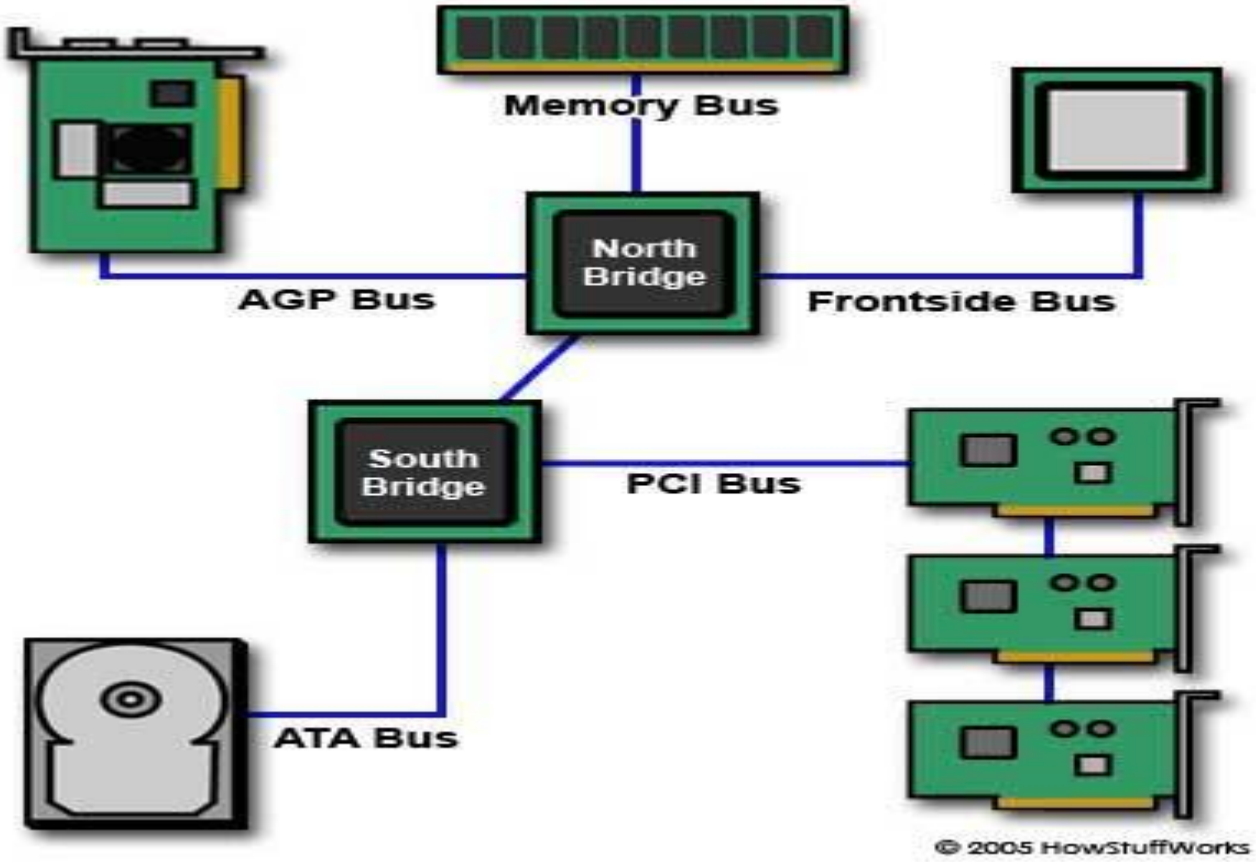
Comparison between 8 and 16 Bit ISA Bus

8-Bit ISA card (XT-Bus)	16-Bit ISA (AT –Bus card)
8-bit data interface	16-bit data interface
4.77 MHZ bus	8-MHZ bus
62-pin connector	62-pin connector
	36-pin AT extension connection

Effect of bus widths:

A bus is simply a circuit that connects one part of the motherboard to another. The more data a bus can handle at one time, the faster it allows information to travel. The **speed** of the bus, measured in megahertz (MHz), refers to how much data can move across the bus simultaneously.

Bus speed usually refers to the speed of the **front side bus** (FSB), which connects the CPU to the north bridge. FSB speeds can range from 66 MHz to over 800 MHz. Since the CPU reaches the memory controller though the north bridge, FSB speed can dramatically affect a computer's performance.



Here are some of the other busses found on a motherboard:

- The **back side bus** connects the CPU with the level 2 (L2) cache, also known as secondary or external cache. The processor determines the speed of the back side bus.
- The **memory bus** connects the north bridge to the memory.
- The **IDE** or **ATA** bus connects the south bridge to the disk drives.
- The **AGP** bus connects the video card to the memory and the CPU. The speed of the AGP bus is usually 66 MHz.
- The **PCI** bus connects PCI slots to the south bridge. On most systems, the speed of the PCI bus is 33 MHz. Also compatible with PCI is **PCI Express**, which is much faster than PCI but is still compatible with current software and operating systems. PCI Express is likely to replace both PCI and AGP busses.

The faster a computer's bus speed, the faster it will operate -- to a point. A fast bus speed cannot make up for a slow processor or chipset.

Data transfer to and from the peripherals may be done in any of the three possible ways

1. Programmed I/O.
2. Interrupt- initiated I/O.
3. Direct Memory Access(DMA).

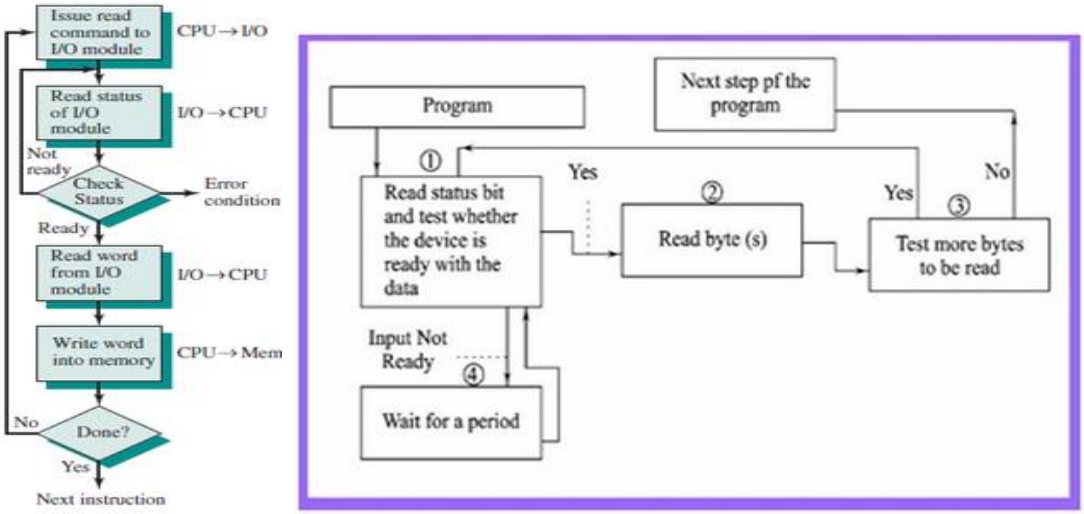
Programmed I/O

- Programmable I/O is one of the I/O techniques other than the interrupt-driven I/O and direct memory access (DMA).
- The programmed I/O was the simplest type of I/O technique for the exchanges of data or any types of communication between the processor and the external devices.
- 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 wasteful of processor time.
- The overall operation of the programmed I/O can be summaries as follow:

1. The processor is executing a program and encounters an instruction relating to I/O operation.
2. The processor then executes that instruction by issuing a command to the appropriate I/O module.
3. The I/O module will perform the requested action based on the I/O command issued by the processor (READ/WRITE) and set the appropriate bits in the I/O status register.
4. The processor will periodically check the status of the I/O module until it finds that the operation is complete.

Programmed I/O Mode Input Data Transfer

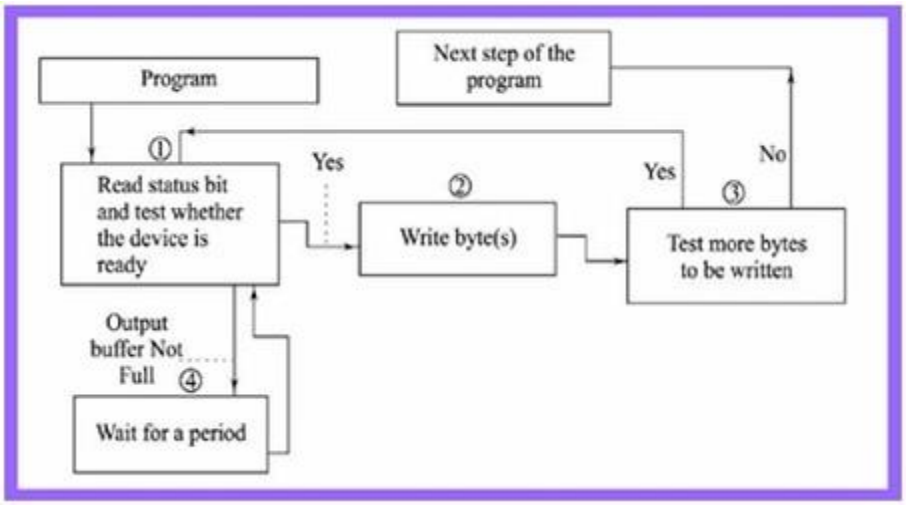


1. Each input is read after first testing whether the device is ready with the input (a state reflected by a bit in a status register).

2. The program waits for the ready status by repeatedly testing the status bit and till all targeted bytes is read from the input device.

3. The program is in busy (non-waiting) state only after the device gets ready else in wait state.

Programmed I/O Mode Output Data Transfer



1. Each output written after first testing whether the device is ready to accept the byte at its output register or output buffer is empty.
2. The program waits for the ready status by repeatedly testing the status bit(s) and till all the targeted bytes are written to the device.
3. The program in busy (non-waiting) state only after the device gets ready else waits state.

I/O Commands

To execute an I/O-related instruction, the processor issues an address, specifying the particular I/O module and external device, and an I/O command. There are four types of I/O commands that an I/O module may receive when it is addressed by a processor:

- **Control**: Used to activate a peripheral and tell it what to do. For example, a magnetic-tape unit may be instructed to rewind or to move forward one record. These commands are tailored to the particular type of peripheral device.

- **Test**: Used to test various status conditions associated with an I/O module and its peripherals. The processor will want to know that the peripheral of interest is powered on and available for use. It will also want to know if the most recent I/O operation is completed and if any errors occurred.

- **Read**: Causes the I/O module to obtain an item of data from the peripheral and place it in an internal buffer. The processor can then obtain the data item by requesting that the I/O module place it on the data bus.
- With **memory mapped I/O**, there is a single address space for memory locations and I/O devices and the processor treats the status and data registers of I/O modules as memory locations and uses the same machine instructions to access both memory and I/O devices. So, for example, with 10 address lines, a combined total of = 1024 memory locations and I/O addresses can be supported, in any combination. With memory-mapped I/O, a single read line and a single write line are needed on the bus.
- With **isolated I/O,** the bus may be equipped with memory read and write plus input and output command lines. Now, the command line specifies whether the address refers to a memory location or an I/O device. The full range of addresses may be available for both. Again, with 10 address lines, the system may now support both 1024 memory locations and 1024 I/O addresses.

  For most types of processors, there is a relatively large set of different instructions for referencing memory. If isolated I/O is used, there are only a few I/O instructions. Thus, an advantage of memory-mapped I/O is that this large repertoire of instructions can be used, allowing more efficient programming. A disadvantage is that valuable memory address space is used up.

Differences between Isolated I/O and Memory Mapped I/O:

Isolated I/O	No.	Memory Mapped I/O
Isolated I/O uses separate memory space.	01	Memory mapped I/O uses memory from the main memory.
Limited instructions can be used. Those are IN, OUT, INS, OUTS.	02	Any instruction which references to memory can be used.
The address for Isolated I/O devices are called ports	03	Memory mapped I/O devices are treated as memory locations on the memory map.

Advantages & Disadvantages

Advantages	-	simple to implement
	-	very little hardware support
Disadvantages	-	busy waiting
	-	ties up CPU for long period with no useful work

What is the difference between programmed-driven I/O and interrupt-driven I/O?

No.	Programmed I/O	Interrupt Driven I/O
1.	In programmed I/O, processor has to check each I/O device in sequence and in effect 'ask' each one if it needs communication with the processor. This checking is achieved by continuous polling cycle and hence processor can not execute other instructions in sequence.	External asynchronous input is used to tell the processor that I/O device needs its service and hence processor does not have to check whether I/O device needs it service or not.
2.	During polling processor is busy and therefore, have serious and decremental effect on system throughput.	In interrupt driven I/O, the processor is allowed to execute its instructions in sequence and only stop to service I/O device when it is told to do so by the device itself. This increases system throughput.
3.	It is implemented without interrupt hardware support.	It is implemented using interrupt hardware support.
4.	It does not depend on interrupt status.	Interrupt must be enabled to process interrupt driven I/O.
5.	It does not need initialization of stack.	It needs initialization of stack.
6.	System throughput decreases as number of I/O devices connected in the system increases.	System throughput does not depend on number of I/O devices connected in the system.

Interrupt- initiated I/O: Since in the above case we saw the CPU is kept busy unnecessarily. This situation can very well be avoided by using an interrupt driven method for data transfer. By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device. In the meantime the CPU can proceed for any other program execution. The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer. Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.

Note: Both the methods programmed I/O and Interrupt-driven I/O require the active intervention of the processor to transfer data between memory and the I/O module, and any data transfer must transverse a path through the processor. Thus both these forms of I/O suffer from two inherent drawbacks.

- The I/O transfer rate is limited by the speed with which the processor can test and service a device.

- The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer.

Direct Memory Access: The data transfer between a fast storage media such as magnetic disk and memory unit is limited by the speed of the CPU. Thus we can allow the peripherals directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access. During DMA the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

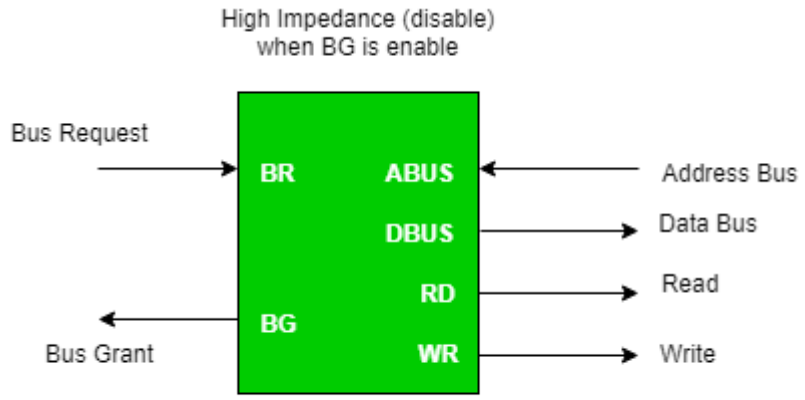


Figure - CPU Bus Signals for DMA Transfer

Bus Request: It is used by the DMA controller to request the CPU to relinquish the control of the buses.

Bus Grant: It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.

Types of DMA transfer using DMA controller:

Burst Transfer:

DMA returns the bus after complete data transfer. A register is used as a byte count, being decremented for each byte transfer, and upon the byte count reaching zero, the DMAC will release the bus. When the DMAC operates in burst mode, the CPU is halted for the duration of the data transfer.

Steps involved are:

- Bus grants request time.
- Transfer the entire block of data at transfer rate of device because the device is usually slow than the speed at which the data can be transferred to CPU.

- Release the control of the bus back to CPU So, total time taken to transfer the N bytes = Bus grant request time + (N) * (memory transfer rate) + Bus release control time.

```
Where,
X µsec =data transfer time or preparation time (words/block)
Y µsec =memory cycle time or cycle time or transfer time (words/block)
% CPU idle (Blocked)=(Y/X+Y)*100
% CPU Busy=(X/X+Y)*100
```

Cyclic Stealing :

In this DMA controller transfers one word at a time after which it must return the control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow the direct memory I/O transfer to “steal” one memory cycle.

Steps Involved are:

- Buffer the byte into the buffer
- Inform the CPU that the device has 1 byte to transfer (i.e. bus grant request)
- Transfer the byte (at system bus speed)
- Release the control of the bus back to CPU.

Before moving on transfer next byte of data, device performs step 1 again so that bus isn’t tied up and the transfer won’t depend upon the transfer rate of device.

So, for 1 byte of transfer of data, time taken by using cycle stealing mode (T). = time required for bus grant + 1 bus cycle to transfer data + time required to release the bus, it will be N x T.

In cycle stealing mode we always follow pipelining concept that when one byte is getting transferred then Device is parallel preparing the next byte. “The fraction of CPU time to the data transfer time” if asked then cycle stealing mode is used.

```
Where,
X µsec =data transfer time or preparation time
(words/block)
Y µsec =memory cycle time or cycle time or transfer
time (words/block)
% CPU idle (Blocked) =(Y/X)*100
% CPU busy=(X/Y)*100
```

Interleaved mode: In this technique, the DMA controller takes over the system bus when the microprocessor is not using it. An alternate half cycle i.e. half cycle DMA + half cycle processor.