

# Introduction to Minicomputers

Bus Structures



digital

1st Printing, June 1976  
2nd Printing (Rev), October 1977  
3rd Printing, August 1979

Copyright © 1976, 1977, 1979 by Digital Equipment Corporation

The reproduction of this workbook, in part or whole, is strictly prohibited. For copy information contact the Educational Services Department, Digital Equipment Corporation, Bedford, Massachusetts 01730.

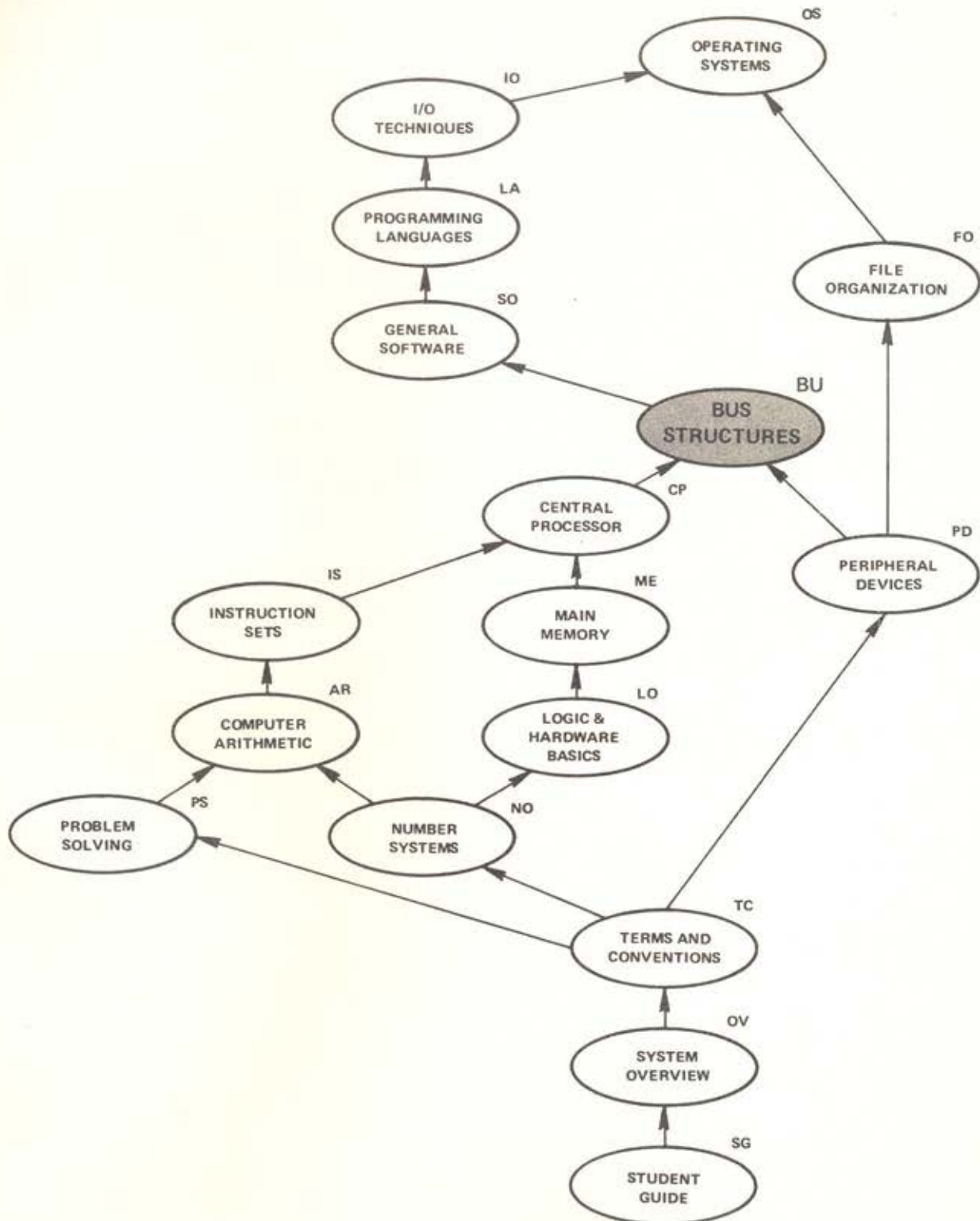
Printed in U.S.A.

**INTRODUCTION TO MINICOMPUTERS**

**Bus Structures**

**Student Workbook**

# COURSE MAP



## CONTENTS

|  |    |
|--|----|
| <b>Introduction</b> .....                    | 1  |
| <b>Bus Structures</b> .....                  | 3  |
| Objectives and Sample Test Items.....        | 3  |
| 3-Bus Configuration.....                     | 8  |
| Single-Bus System.....                       | 14 |
| Bus Structure Implications.....              | 18 |
| Maximum Memory Size.....                     | 19 |
| Instruction Set.....                         | 20 |
| Information Flow.....                        | 20 |
| Throughput.....                              | 20 |
| Exercises and Solutions.....                 | 23 |
| <b>Serial vs Parallel Transmission</b> ..... | 31 |
| Objective and Sample Test Item.....          | 31 |
| Serial Transmission.....                     | 33 |
| Parallel Transmission.....                   | 34 |
| Exercises and Solutions.....                 | 37 |
| <b>Interfaces</b> .....                      | 41 |
| Objective and Sample Test Item.....          | 41 |
| Exercise and Solution.....                   | 51 |

# Bus Structures

## Introduction

A digital computer is composed of several different units that operate together to form a complete computer system. These units include the central processor, a main memory, and any number of peripheral devices such as tape transports, magnetic disks, and line printers. Operation of the computer system involves transferring addresses, data, and control information among these major system elements.

The usefulness of any digital computer depends largely on the range of peripherals that can be connected to it. Unfortunately, peripherals cannot simply be plugged into the CPU. Most peripherals are standard devices that have been designed for use with many types of computers made by different manufacturers. Because each computer has its own method of handling information, there must be some method of connecting a peripheral to a specific computer so that the two devices can operate properly. This is accomplished by using an "interface." An interface may be thought of as a converter between a peripheral and a specific computer. All connections from the computer system to the peripheral are made through this interface.

Connections between the central processing unit, the main memory, and peripheral devices are made by electric cables called "busses." Busses are simply bundles of wires that carry addresses, data, and control information among the central processing unit, memory, and peripheral devices. However, the way these wires are arranged, or configured, and connected to the various units plays an important role in determining the characteristics of the computer system.

This module contains three lessons. The first lesson describes some typical *bus configurations* used in computer systems. Common methods of connecting the CPU to memory and peripheral interfaces are discussed.

The second lesson discusses the differences between *serial* and *parallel transfers* of binary information. Every peripheral device uses one of these two methods for transferring data to its associated interface.

Finally, the third lesson explains the basic *functions of interfaces* as they relate to the information carried on the busses in computer systems.

## Bus Structures

### OBJECTIVES

1. Given the term "bus" and six descriptive statements, be able to label those statements that provide correct information about bus structures.
2. Given 4 bus-related statements, be able to label those statements that refer to a single-bus and those that refer to a 3-bus configuration.
3. Given blank block diagrams of both 3-bus and single-bus configurations; a list of 6 bus components; and 3 bus types, be able to: (1) match each component with its position in the diagram, and (2) label each bus type in the 3-bus configuration.
4. Given the abbreviation "DMA" and seven descriptive statements, be able to label the statements that provide correct information about DMA.

### SAMPLE TEST ITEMS

1. For each statement below, write a T in the space provided if the statement correctly describes a bus structure. Write an F in the space provided if the statement does not correctly describe a bus structure.

**A Bus Structure. . .**

**T or F**

is a cable containing a bundle of wires.

\_\_\_\_\_

is a group of cables, each containing a single transmission wire.

\_\_\_\_\_

•  
•  
•

•  
•  
•

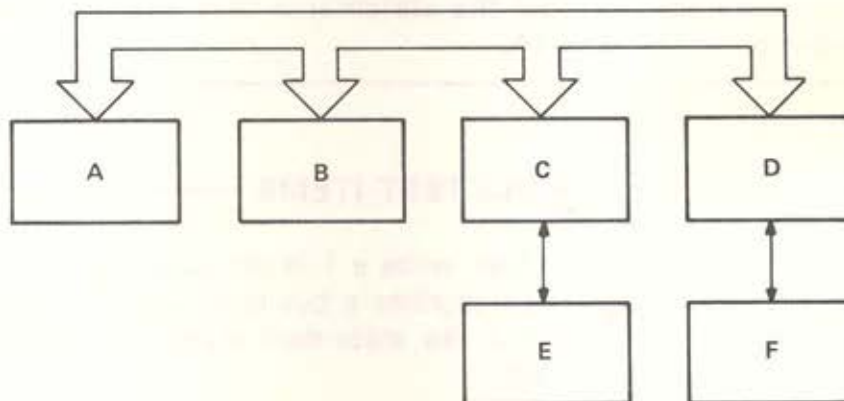
## SAMPLE TEST ITEMS

2. For each statement, indicate that it refers to a single-bus (SB) or 3-bus (3B) configuration by writing the correct abbreviation in the blank space.

| Statement                   | Bus Type |
|-----------------------------|----------|
| More versatile data flow.   | _____    |
| Larger maximum memory size. | _____    |
| .                           | .        |
| .                           | .        |
| .                           | .        |

3. Match each of the components below with its position in the single-bus configuration.

### Single-Bus Configuration



| Components         | Position |
|--------------------|----------|
| Disk Interface     | _____    |
| Moving Head Disk   | _____    |
| CPU                | _____    |
| Terminal Interface | _____    |
| Main Memory        | _____    |
| Terminal           | _____    |

**SAMPLE TEST ITEMS**

4. Indicate that each of the statements below provides true or false information about DMA by writing either a T or F in the space provided.

| <b>DMA...</b>   | <b>T or F</b> |
|---|---------------|
| means data and memory access.                             | _____         |
| moves information to and from<br>main memory via the CPU. | _____         |

Mark your place in this workbook and view Lesson 1 in the A/V program, "Bus Structures."

A typical bus used in a computer system is a flat, flexible cable with connectors attached to both ends of the cable. The cable (bus) contains many individual transmission lines, or wires. Each line, or group of lines, is designed to carry some specific information. For example, the bus connecting the CPU to a magnetic tape interface contains separate groups of lines for transmitting addresses, data, and control information.

*Address lines* (sometimes called device selection lines) direct data to and from the desired devices. That is, address lines are used to select the devices that are to engage in a transfer of information. For example, if the CPU wants to load data into a buffer register within a magnetic tape interface, it places the address of that register on the bus. The magnetic tape interface would respond; all other devices would ignore the address. Typically, there is one address line for each bit in the computer's word length.

*Data lines* are used to transfer information in both directions between the two devices using the bus. The data lines between the CPU and memory, for instance, allow data to be sent from memory to the CPU or from the CPU to memory.

*Control lines* are used for both control and monitor functions. Assume, for example, that there are a number of control lines in the bus that connects the CPU to a magnetic tape interface. The CPU would perform control functions by using one control line to issue a "start" command to the tape unit, another control line to issue a "read" command, and still another line to issue a "write" command. When performing monitoring functions, the CPU would monitor the control lines used by the tape unit interface to indicate the status of the tape unit. Thus, the interface might use one control line to transmit a "busy" signal to the CPU and a different control line to transmit a "done" signal.

Most busses are constructed in the same manner. They differ only in the number of data and address lines in each bus and in the number and function of their control lines. However, in minicomputers there are *two fundamental arrangements* of busses: a 3-bus configuration and a single-bus configuration.

### 3-Bus Configuration

Figure 1 shows how a 3-bus configuration interconnects the various computer system components. As its name implies, the 3-bus configuration uses three completely separate and independent busses to serve as information paths between computer components. These three busses are: *input-output (I/O) bus*, *memory bus*, and *direct memory access (DMA) bus*.

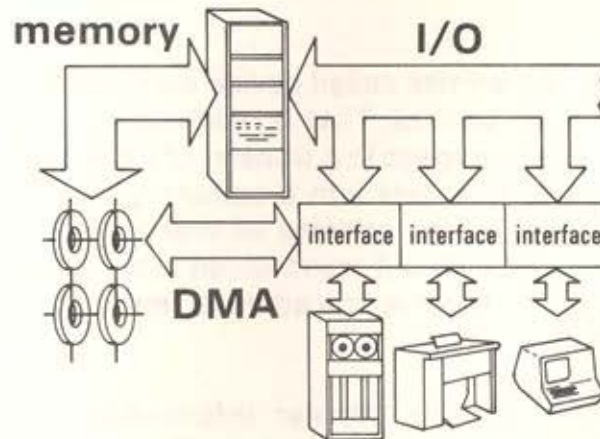


Figure 1 3-Bus Configuration

The input-output (or I/O) bus provides communication between the central processor and the various peripheral devices on the system. Note, however, that the peripheral devices themselves are not connected *directly* to this I/O bus. Instead, bus connections are made through individual "interfaces." In other words, each peripheral device is connected to its own interface. The interface, in turn, is attached to the I/O bus. The prime purpose of the interface is to compensate for differences between a "standard" peripheral device and a specific computer system. The interface may perform such functions as code conversion, signal translation, and speed compensation.

The second type of bus in a 3-bus configuration is the *memory bus*. The purpose of the memory bus is to carry addresses, data, and control information between the central processing unit and the main memory.

The third type of bus used in a 3-bus configuration is the *direct memory access* (or DMA) bus. The DMA bus interconnects main memory with one or more high-speed, mass storage devices such as magnetic tape units. Notice, as shown in Figure 1, that the DMA bus is connected to the tape unit *interface*; it is not connected directly to the tape unit itself. Also notice that the DMA bus completely *bypasses* the central processing unit. In order to understand why the DMA bus bypasses the CPU, it is necessary to look at the bus connections more closely.

Let's examine the information that flows over each bus during the operation of a computer system that includes a central processor, main memory, and a peripheral device.

As the computer system performs its operations, the central processor executes programs of instructions. Each time the central processor is ready for another instruction, it sends the address from the program counter and control information down the memory bus and receives the bits of a new instruction in return (Figure 2).

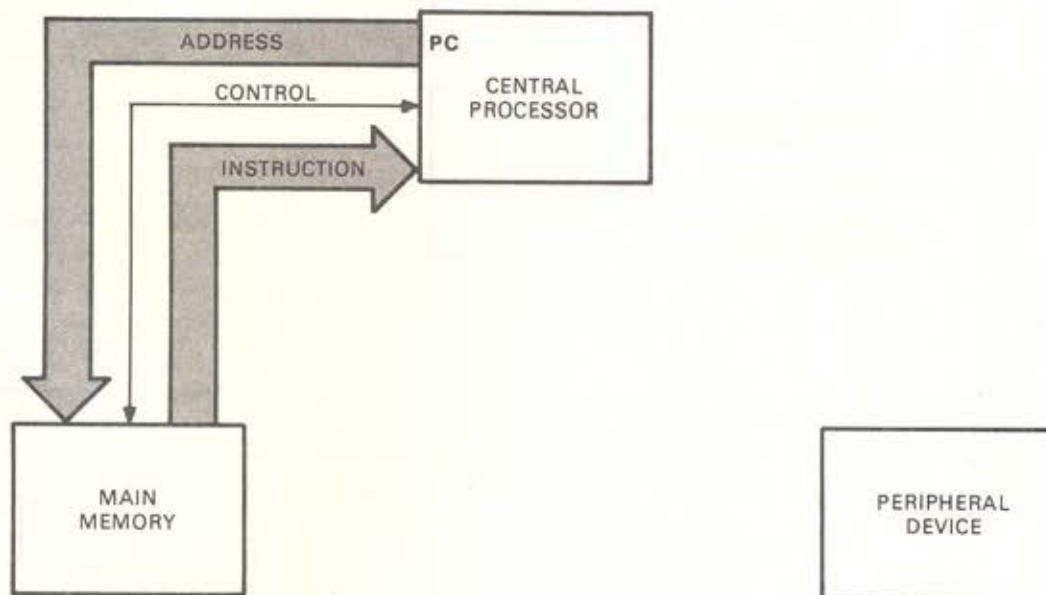


Figure 2 Use of Memory Bus to Fetch Instruction

Each instruction has two parts: an op code to tell the processor what to do next; and an address to indicate where the information that is required by that instruction is located (Figure 3).

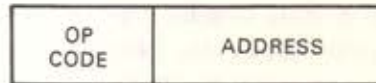


Figure 3 Parts of an Instruction

In a 3-bus system, the op code tells the processor which bus to use. Some instruction op codes require the processor to reference a memory location. To perform these memory reference instructions, the processor sends the address from the instruction and control information down the memory bus. Then, data is moved over the memory bus in the direction required by the instruction op code (Figure 4).

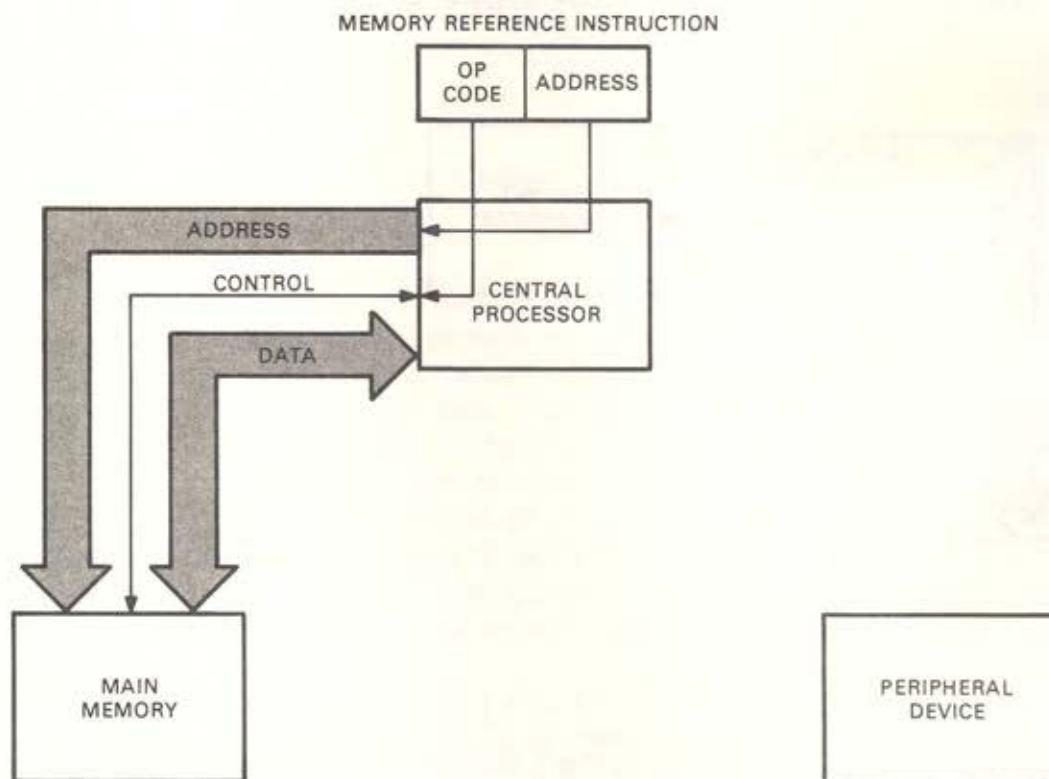


Figure 4 Use of Memory Bus to Move Data Between Memory and Processor

However, some op codes in a 3-bus system specify operations involving peripheral devices. In instructions with these op codes, the address portion of the instruction is a numeric device code that identifies the particular peripheral device that is being referenced. When the processor executes one of these instructions, it sends the device code from the address portion of the instruction and control information down the I/O bus to the interface of the peripheral device. Then data is transferred over the I/O bus between the processor and the device that corresponds to the device code (Figure 5).

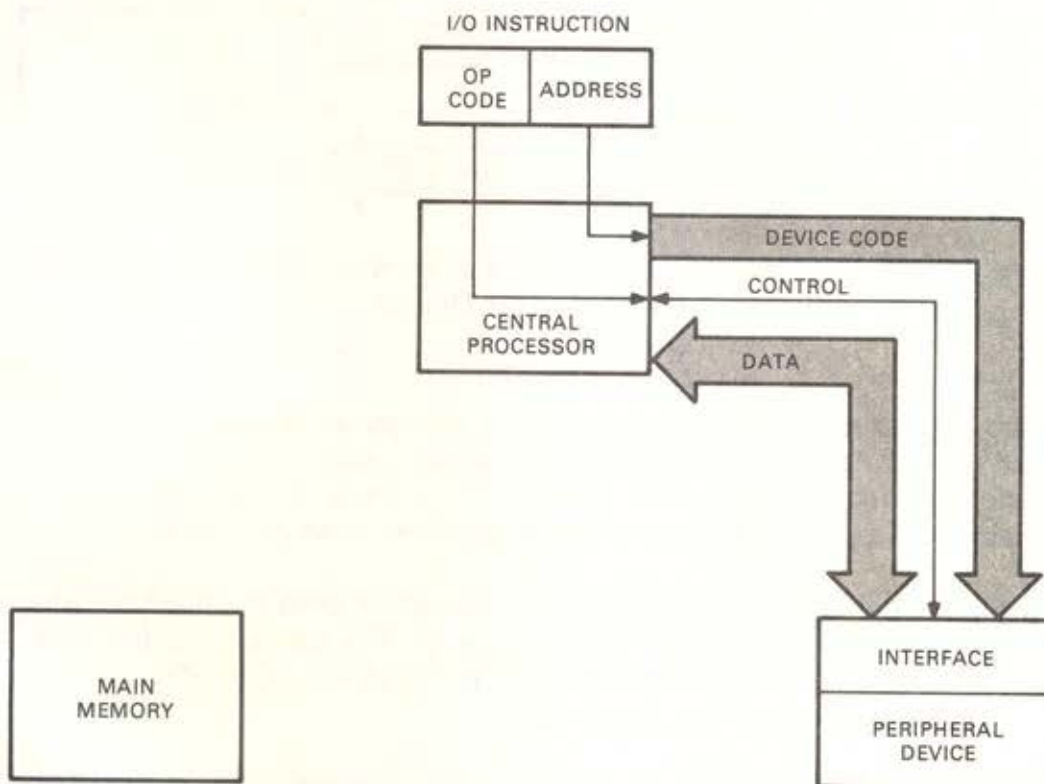


Figure 5 Use of the I/O Bus to Transfer Data Between the Central Processor and a Peripheral Device

The DMA bus is used to transfer data directly between main memory and high-speed peripheral devices. When the DMA bus is used, the peripheral device sends memory address and control information down the DMA bus. The data is transferred between main memory and the interface of the peripheral device (Figure 6).

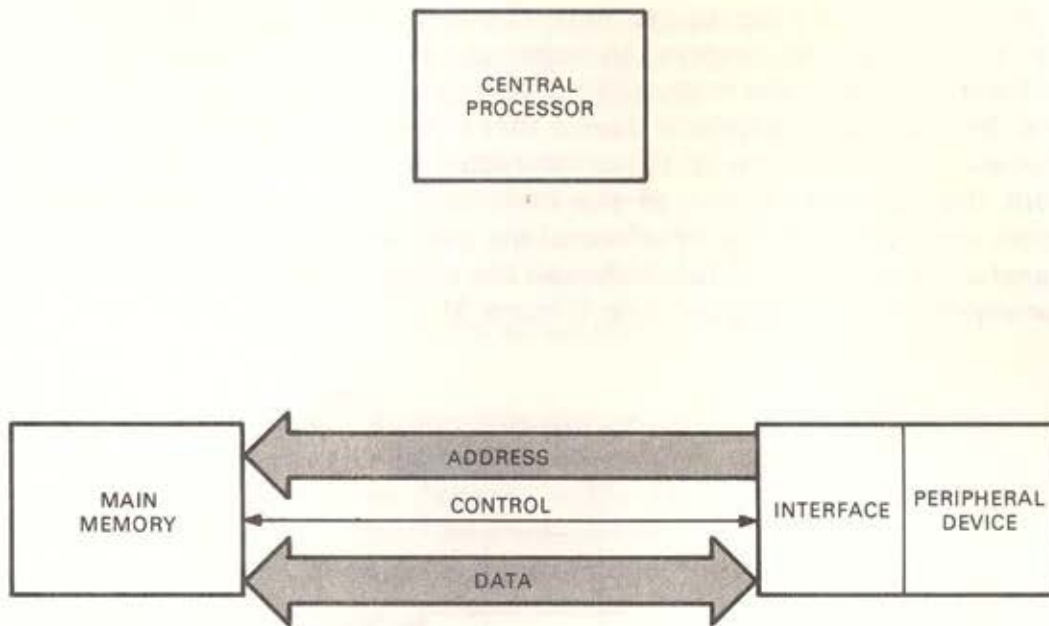


Figure 6 Use of DMA Bus to Move Data Between Main Memory and a Fast Peripheral Device

The use of the DMA bus allows the computer system to operate faster than it can by using only the memory bus and the I/O bus. To transfer a word of information from main memory *through the central processor* to a peripheral device requires *three memory cycles*:

1. During the first cycle, the central processor uses the memory bus to fetch an instruction from memory. The instruction tells the processor to use the memory bus to move the word of data from main memory to the processor.
2. During the second memory cycle, the processor uses the memory bus to execute the instruction and move the word from main memory to an accumulator in the processor.
3. During the third memory cycle, the processor uses the memory bus to fetch another instruction. This instruction tells the processor to send the data in the accumulator down the I/O bus to a specific device (Figure 7). While the core memory is restoring the contents of the memory location, the processor uses the I/O bus to send out the data.

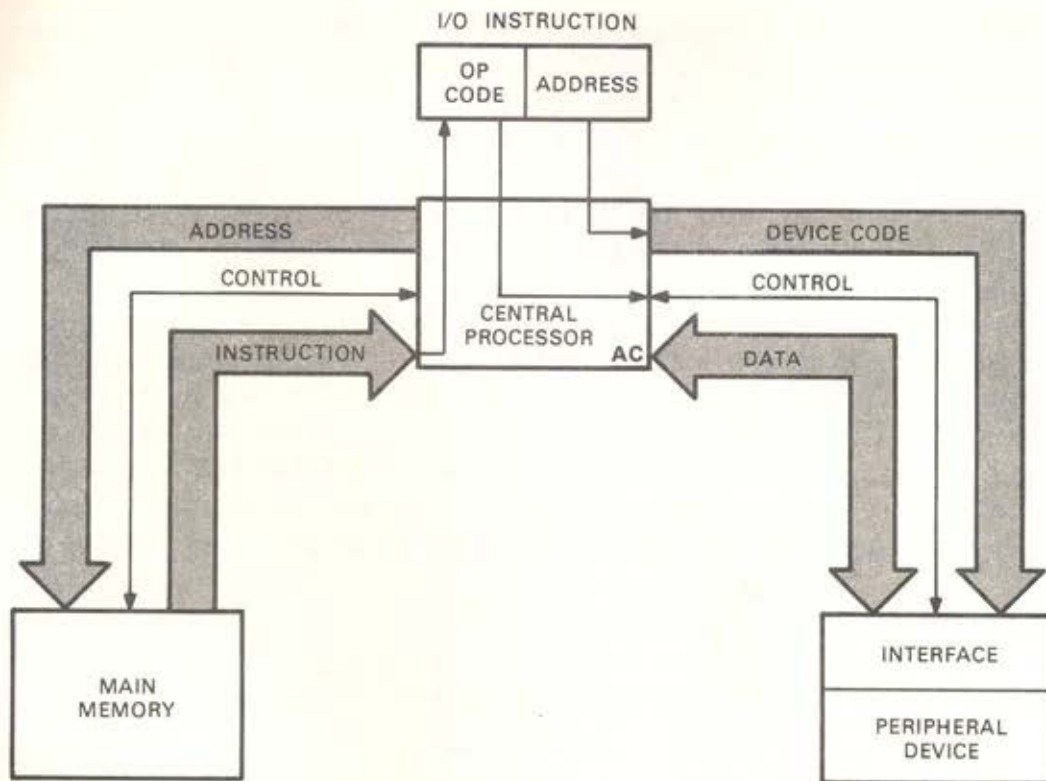


Figure 7 Use of Both Memory and I/O Busses During One Memory Cycle for an I/O Instruction

When the DMA bus is used, only one memory cycle is required to move data between main memory and the peripheral device. This is important because main memory is often the time-limiting part of a computer system.

In minicomputers, this limit is made more critical by the fact that the memory bus and the DMA bus cannot be used at the same time. In general, on a 3-bus system only one bus is used at any one instant. The memory bus and the DMA bus cannot access memory at the same time; the use of the I/O bus occurs only after the processor receives an I/O instruction from the memory bus.

Therefore, some computers have been designed using a single bus to perform the functions of all three busses (memory reference, I/O, DMA). The single-bus structure requires that the logic circuits connected to the bus operate with great speed so that each new bus use can rapidly follow the last. Otherwise, the fact that one bus must be shared for many functions might result in extra delays. Figure 8, which is a simplified diagram of a 3-bus configuration, illustrates actual bus connections. Notice that both the memory bus and the DMA bus are connected to main memory *at the same point*. This point, called a port, has connections to the location select and read/write circuits of the memory. In other words, entry into memory is *shared* by the two busses. Either the *CPU* can use the *memory bus* to transfer data to or from memory, or a *peripheral device* can use the *DMA bus* to transfer data to or from memory. However, only *one bus* can access the location select and read/write circuits of main memory at any given time.

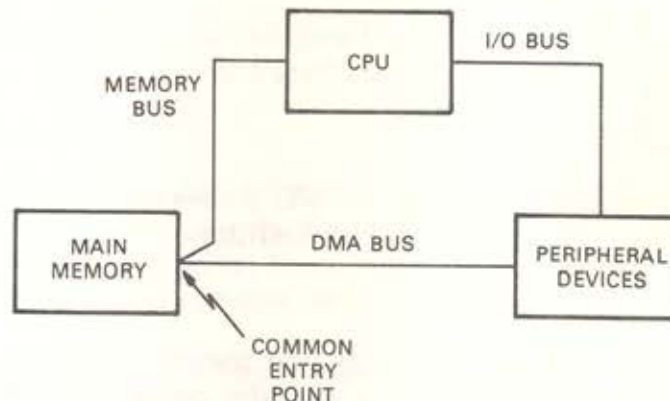


Figure 8 Bus Connections

### Single-Bus System

On a single-bus system, when the central processor fetches an instruction, it puts the memory address in the program counter on the bus along with control information. The address goes down the bus to main memory and to all the peripheral devices on the bus (Figure 9).

The device whose address appears on the bus performs a data transfer. In this case the address is a memory address, so the memory responds with the contents of the addressed location, which is an instruction.

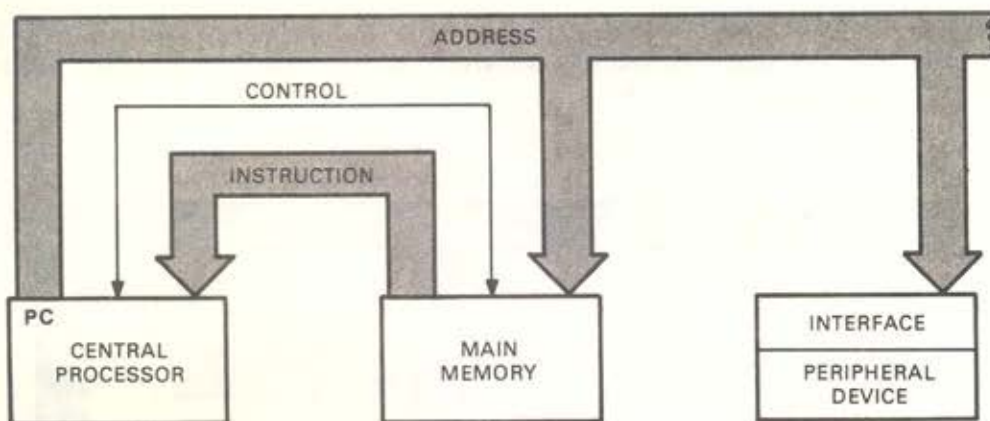


Figure 9 Use of Single Bus to Fetch Instructions

Once the instruction is fetched, the processor examines the op code to find out what to do next. On a single-bus computer, the op code specifies what operation is to be performed on the data and whether the data is flowing *to* or *from* the processor. However, the op code does not specify whether the instruction references memory or a peripheral device.

On a single-bus system, the difference between a memory reference instruction and an I/O instruction is the numeric value of the address. Therefore, the central processor simply puts the address portion of the instruction on the bus (Figure 10) and relies on the appropriate device to respond.

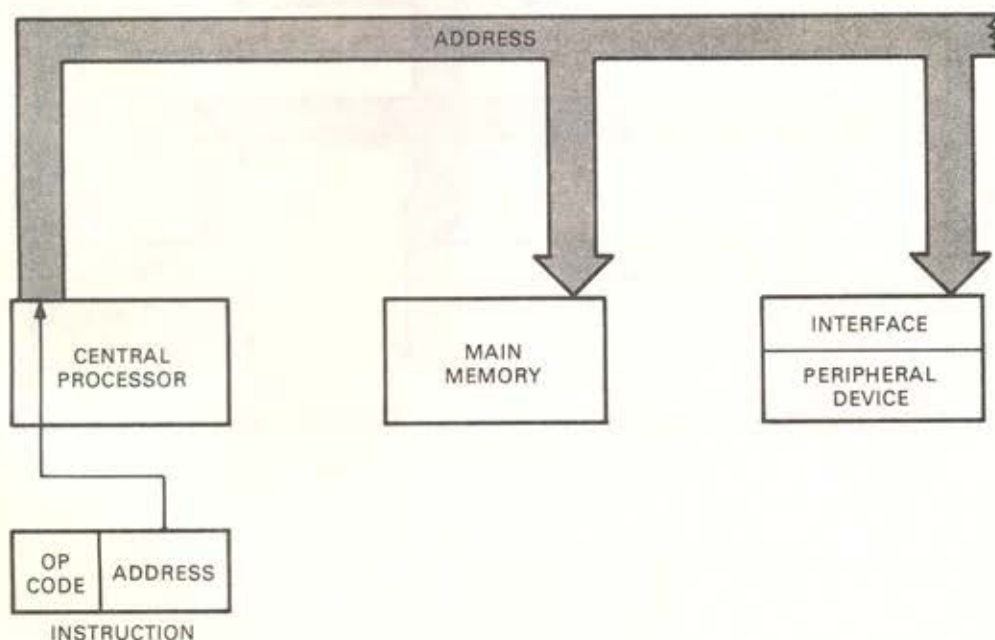


Figure 10 Processor Puts the Instruction Address on the Bus

If the address corresponds to a memory address, main memory responds. A data transfer occurs between main memory and the processor (Figure 11).

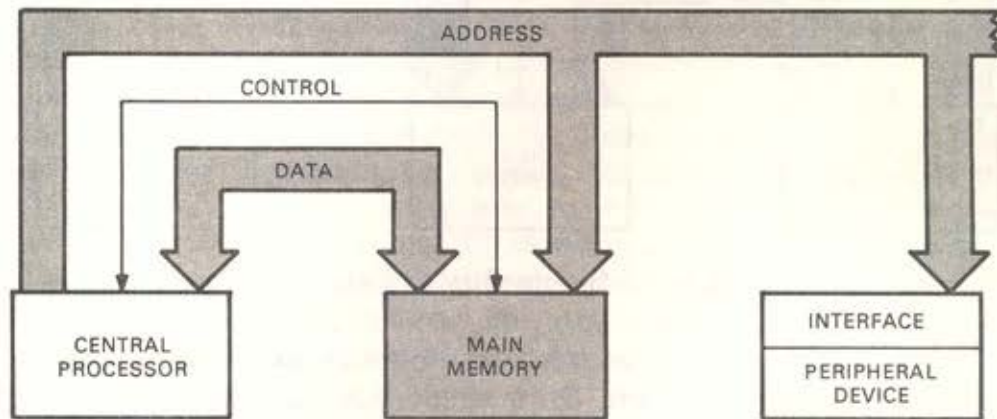


Figure 11 Use of Single Bus to Transfer Data Between the Processor and Main Memory

On the other hand, if the address from the instruction corresponds to a peripheral device, that device responds. A data transfer occurs between the processor and the addressed device (Figure 12).

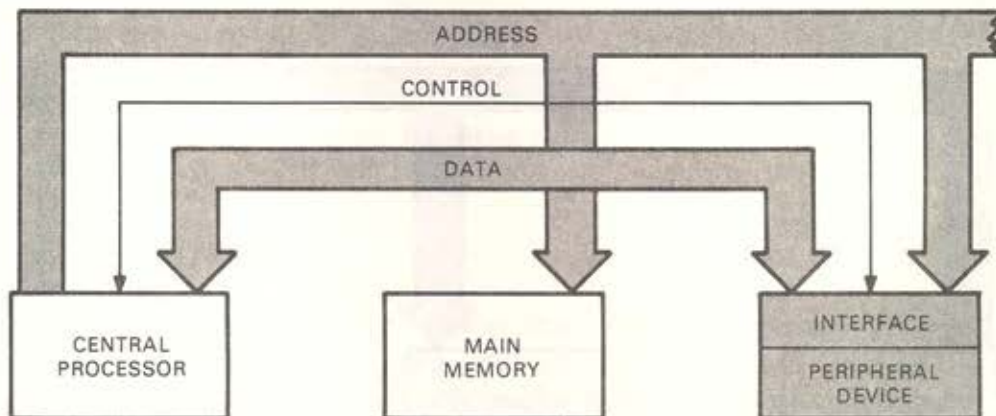


Figure 12 Use of Single Bus to Transfer Data Between the Processor and a Peripheral Device

Single-bus systems can also perform the equivalent of DMA. When this happens, a high-speed peripheral device takes over the bus by exchanging control signals with a bus control circuit that is normally located in the central processor. Once the peripheral device is granted control of the bus, the peripheral asserts an address on the bus (sets the address lines on the bus to the appropriate 1s and 0s) to specify the device with which it wishes to exchange information (Figure 13).

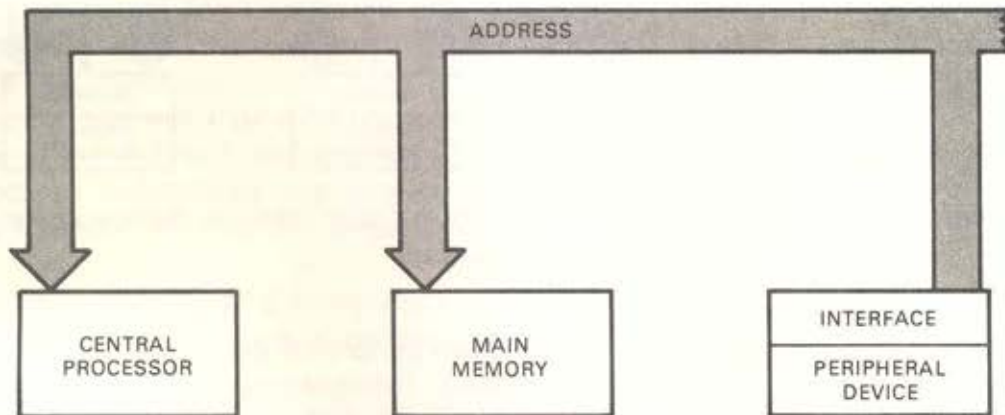


Figure 13 Some Peripheral Devices Can Also Assert Addresses

If the address asserted by the peripheral device corresponds to a memory location, main memory responds on the bus. The result is a direct transfer of data between main memory and the peripheral device (DMA)(Figure 14).

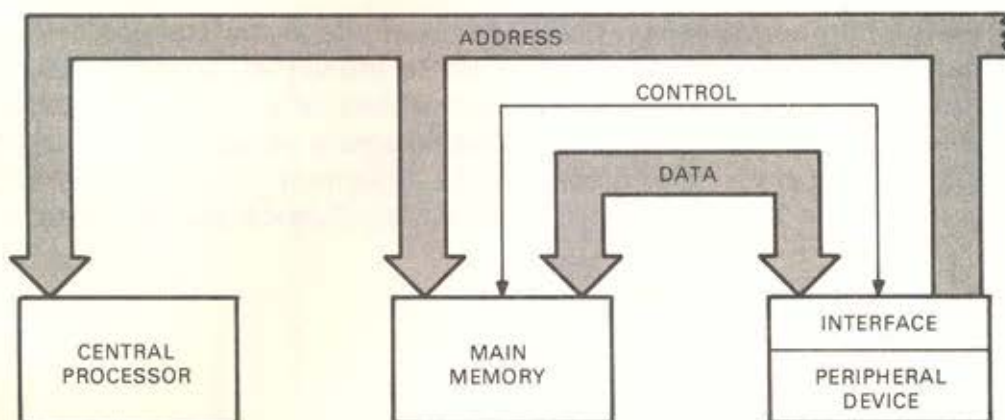


Figure 14 Use of Single Bus to Perform DMA Operation

However, if the address asserted by the peripheral device corresponds to *another* peripheral device, the second device responds, and a data transfer occurs directly *between peripheral devices* (Figure 15).

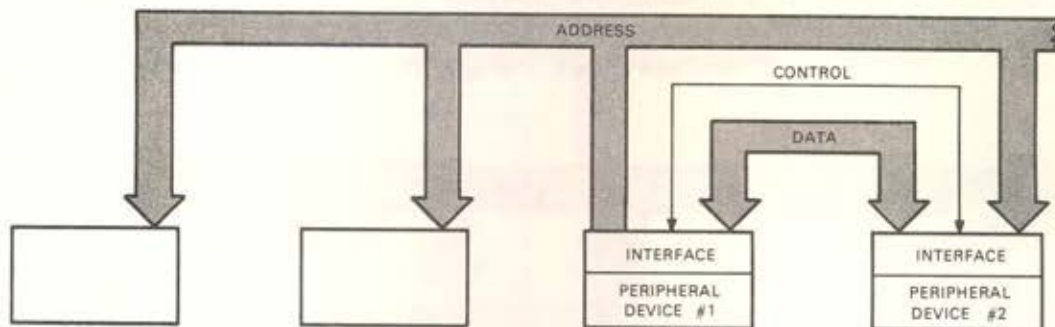


Figure 15 Use of Single Bus to Perform Data Transfer Between Peripheral Devices

Direct transfer of information between peripheral devices is not done on 3-bus systems. In 3-bus systems, information going between devices is first moved to main memory. Main memory serves as an intermediate buffer area. Then the information is moved from main memory to the second device.

The use of main memory as a buffer between peripheral devices is also common in single-bus systems when the peripheral devices involved in the transfer operate at very different speeds.

### Bus Structure Implications

A large number of design decisions are made in the creation of new computers. Often it is difficult to point to individual decisions as the primary determinant in the final specifications of a computer system. For example, the choice of a single-bus structure versus a 3-bus structure is not the only determinant of the maximum memory size of a system. But the bus structure chosen does influence memory size.

With this qualification in mind, let's look at the implications of the bus structure of a computer system. Four major areas will be discussed:

1. Maximum Memory Size
2. Instruction Set
3. Information Flow
4. Throughput

**Maximum Memory Size** – At first glance it might seem that the maximum size of main memory on a computer system is not related to the bus structure but to the size of the computer word. Typically, the largest address that a computer can have is a full word address that is used for indirect addressing. Therefore, the number of bits in this address determines the largest possible address value, and memory size cannot exceed the largest address on the system.

The word size of a computer *is* the *primary* determinant of maximum memory size, but bus structure also has an effect (Figure 16).

In a single-bus system, some address values must be reserved for peripheral devices. The difference between accessing main memory and accessing a peripheral device is the value of the address that is placed on the bus. Since some addresses must be used for peripheral devices, the maximum amount of memory on a single bus system must be *less than* the total number of possible addresses determined by the computer's word length.

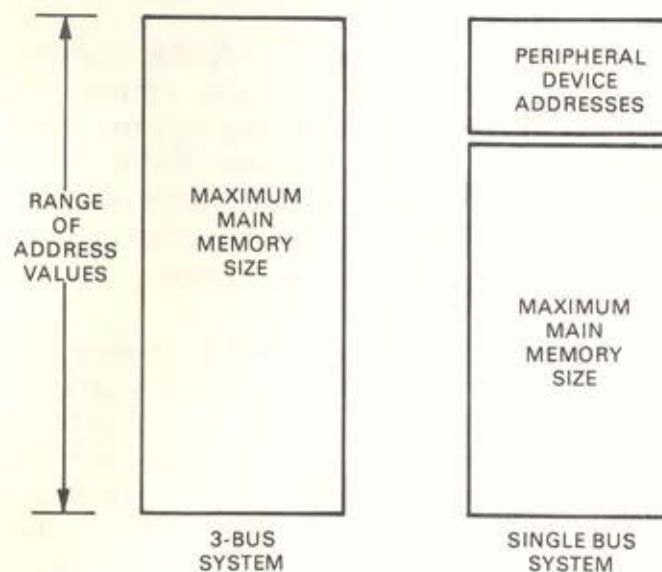


Figure 16 Maximum Memory Size Versus Bus Structure

In 3-bus systems, the difference between accessing main memory and a peripheral device is the particular op code in the instruction. Since memory addresses and device addresses apply to different buses, the presence of peripheral devices does not restrict memory size.

**Instruction Set** – In 3-bus systems, the processor must have a different set of instructions for each bus because the op code specifies which bus to use with the address. So in these systems there are separate instructions for memory reference and I/O.

Single-bus systems, on the other hand, use the same op codes for memory reference and I/O transfers. These systems use different addresses for memory reference and I/O.

Therefore, programmers who write programs that use the instruction set of a single-bus machine have fewer op codes to remember. This makes machine-level programming much easier on single-bus systems without limiting what the programmer can do.

**Information Flow** – On single-bus systems, information flow between peripheral devices can be direct (device to device) or indirect (device to memory to device). Also, the processor can move the information indirectly through itself (device to accumulator to device).

On 3-bus systems, direct device-to-device transfers are not possible. Therefore, in situations where this kind of transfer would be possible on single-bus systems, 3-bus systems operate less efficiently by moving information indirectly through memory or the processor.

**Throughput** – The throughput of a computer system is the rate at which information can be moved into the system, processed, and moved out. This rate depends how fast the information can be moved into main memory; how fast the processor can fetch instructions and data from main memory and replace them with results; and how fast the results can be sent from main memory to an output device. Main memory is involved in each one of these steps.

Since processors and the logic attached to busses typically perform their functions faster than main memory, the primary measure of system throughput is the cycle time of the memory. For this reason, memory interleaving and cache memory are important to improve the performance (throughput) of a system. But there is another way to increase the effective speed of the memory through the use of more than one bus. That way is to use multi-ported memory.

Multi-ported memory is memory with connections for more than one bus. We noted earlier that most 3-bus systems only have one port or connection to their memory; this port is shared between the memory bus and the DMA bus. The fact that this port is shared prevents the DMA bus and the memory bus from operating at the same time.

However, multi-ported memory has a different set of location select and read/write circuits for each port. This means that if different busses are connected to each port, the memory can perform a read or write operation at each port at the same time. This allows more information to flow in and out of memory than is possible with only one port.

Therefore, if 3-bus systems use multi-ported memory, they can operate faster than single-bus systems. However, in *minicomputers*, the use of multi-ported memories is rare.

#### NOTE

One could argue that multi-ported memory might be used with single-bus architecture. However, this would require another bus, and the computer system could not be categorized as a single-bus system any longer.

The following chart summarizes the advantages of single and 3-bus systems:

| Single-Bus System   | 3-Bus System  |
|---|---|
| 1. Fewer machine instructions for programmers to remember | 1. Larger maximum memory size                                   |
| 2. More versatile data flow                               | 2. More potential for throughput if multi-ported memory is used |

Before proceeding to the next lesson, do the exercises for this lesson.

---

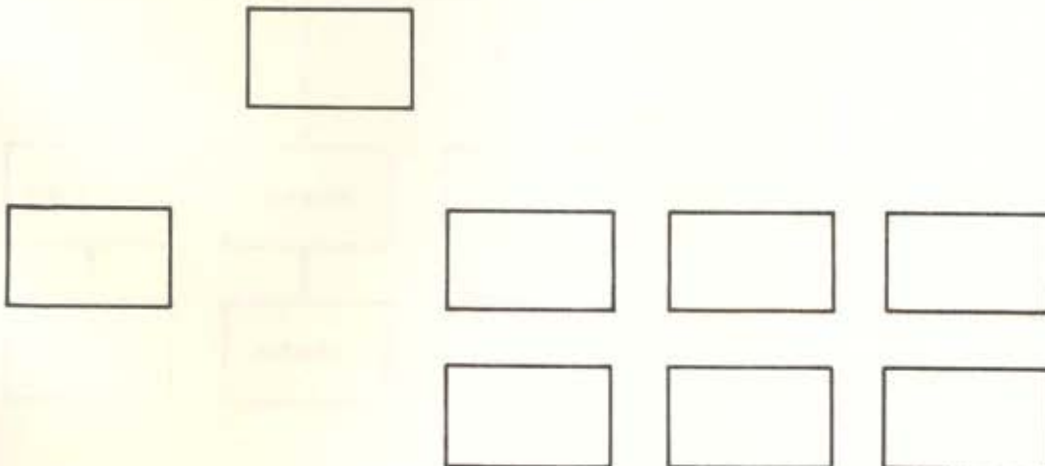
## EXERCISES

---

1. In the space provided below, draw a simple block diagram of a computer system that has three components (CPU, main memory, and one peripheral device) connected in a 3-bus configuration. Label each bus.

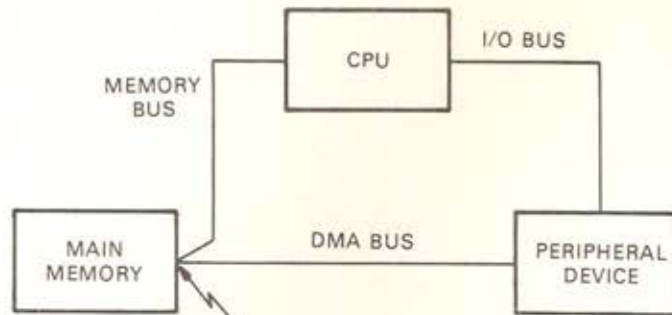
2. The computer system shown below has a CPU, main memory, and three peripheral devices (tape unit, printer, display terminal) with associated interfaces.

Write the appropriate name in each of the boxes, draw in the busses needed to make this a 3-bus system, and label each bus.



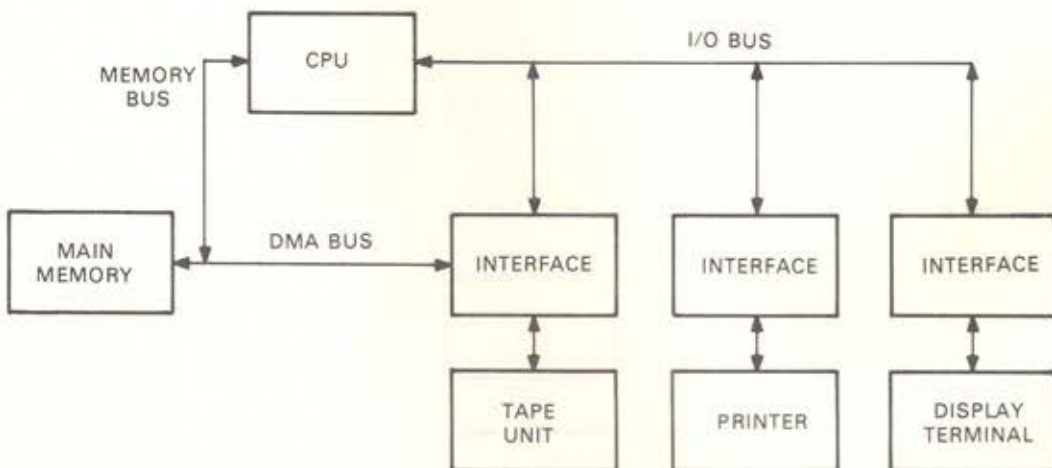
## SOLUTIONS

1. In the space provided below, draw a simple block diagram of a computer system that has three components (CPU, main memory, and one peripheral device) connected in a 3-bus configuration.



2. The computer system shown below has a CPU, main memory and three peripheral devices (tape unit, printer, display terminal) with associated interfaces.

Write the appropriate name in each of the boxes, draw in the busses needed to make this a 3-bus system, and label each bus.



---

### EXERCISES

---

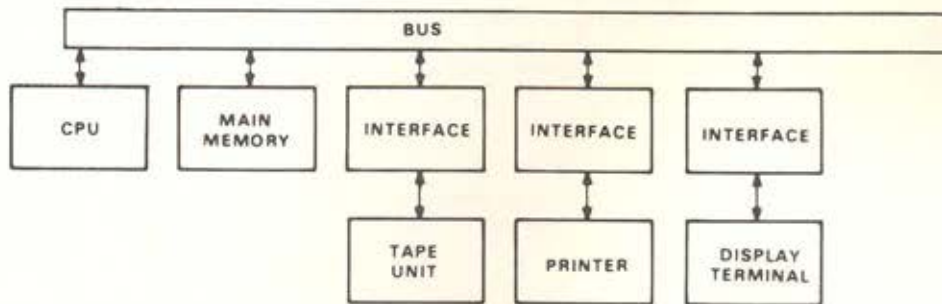
3. In the space provided below, draw a simple block diagram of a *single-bus* computer system that has a CPU, main memory, and three peripheral devices (tape unit, printer, display terminal) with associated interface units.

---

## SOLUTIONS

---

3. In the space provided below, draw a simple block diagram of a *single-bus* computer system that has a CPU, main memory and three peripheral devices (tape unit, printer, display terminal) with associated interface units.



---

## EXERCISES

---

4. Match each of the following terms with the appropriate definition by placing the letter of the term in the parentheses next to the definition.

- |                  |     |  |
|------------------|-----|--|
| a. bus           | ( ) | A method of transferring data directly between memory and a peripheral device.                     |
| b. DMA           |     |  |
| c. address lines | ( ) | Consists of a number of both bidirectional and single direction lines                              |
| d. data lines    |     |  |
| e. control lines | ( ) | Used to steer data to and from a selected device   |
|                  | ( ) | Part of the bus used to transfer information in both directions between two devices                |
|                  | ( ) | Bus lines that permit monitoring of a device's status by the CPU                                   |
|                  | ( ) | Bus lines used by the CPU to issue commands to a selected device                                   |
|                  | ( ) | An electrical cable used to interconnect computer components                                       |
|                  | ( ) | An element that provides required communication paths for addresses, data, and control information |
|                  | ( ) | Direct memory access   |

---

## SOLUTIONS

---

4. Match each of the following terms with the appropriate definition by placing the letter of the term in the parentheses next to the definition.

- a. bus
- b. DMA
- c. address lines
- d. data lines
- e. control lines

- (b) A method of transferring data directly between memory and a peripheral device.
- (a) Consists of a number of both bidirectional and single direction lines
- (c) Used to steer data to and from a selected device
- (d) Part of the bus used to transfer information in both directions between two devices
- (e) Bus lines that permit monitoring of a device's status by the CPU
- (e) Bus lines used by the CPU to issue commands to a selected device
- (a) An electrical cable used to interconnect computer components
- (a) An element that provides required communication paths for addresses, data, and control information
- (b) Direct memory access

---

## EXERCISES

---

5. List at least one major advantage of a single-bus system and one major advantage of a multiple-bus system.

a. Single-bus system advantage:

b. Multiple-bus system advantage:

6. The abbreviation "DMA" stands for \_\_\_\_\_ .

Describe what this term means in relation to bus structures.

---

## SOLUTIONS

---

5. List at least one major advantage of a single-bus system and one major advantage of a multiple-bus system.

### NOTE

**Any one of the following statements can be used.**


- a. Single-bus system advantage:
    - Fewer instructions for programmers to master
    - More versatile data flow
  - b. Multiple-bus system advantage:
    - A larger maximum memory size
    - Can use multi-port memory to get increased performance
6. The abbreviation "DMA" stands for **direct memory access**. In a 3-bus configuration, the DMA bus interconnects main memory with one or more high-speed mass storage devices.

## Serial vs Parallel Transmission

### OBJECTIVE

Given a series of binary digits (e.g., 110101), be able to draw both the serial transmission diagram and the parallel transmission diagram for the digits.

### SAMPLE TEST ITEM

You are given the binary digits 110101. The pulse (  ) represents a 1, while no pulse (—) represents a 0. Draw both the serial transmission diagram and the parallel transmission diagram for these digits.

Mark your place in this workbook and view Lesson 2 in the A/V program, "Bus Structures."

Processing information in a digital computer requires switching and storing electrical signals. Because digital computers use binary numbers, these electrical signals need represent only one of two values; that is, each signal must represent either a 0 or a 1. There are two methods used for transmitting these binary signals: *serial transmission* and *parallel transmission*. Two different methods are used because each method has distinct advantages and disadvantages.

### Serial Transmission

When information is transmitted in *serial* fashion, the signals (pulses) that represent the binary digits (bits) of a number are transmitted in *sequence*, one bit at a time, over a single wire. The bits are usually transmitted so that the first signal represents the least significant bit of the number and the last signal sent is the most significant bit of the number. Serial transmission is illustrated in Figure 17.

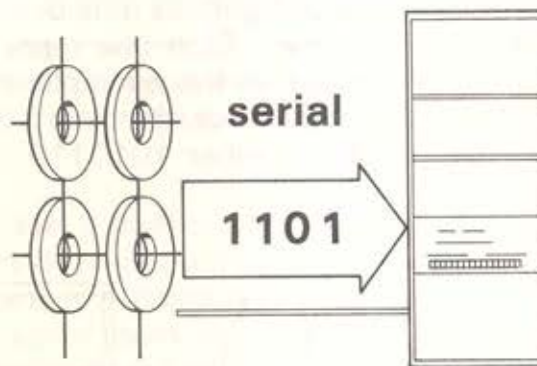


Figure 17 Serial Transmission

When using serial transmission, it is possible for a number of 0s or a number of 1s to follow one another in sequence. If a coding scheme is used in which a binary 0 is represented by the absence of a pulse, six or seven 0s in a row would be represented by no signal on the line for some period of time. In order to determine how many 0s are being transmitted, there must be some type of timing scheme that the receiving device can understand. The approach normally used is to indicate the "bit time"; that is, the length of time a bit (whether 0 or 1) is on the line.

If it is assumed that serial data consists of "bit times," then it takes eight bit times to transmit eight bits, or one byte, of binary data. If, for example, the receiving device received a pulse at the first bit time and then received no more pulses for the remaining seven bit times, it would know that the information received consisted of a binary 1 followed by seven binary 0s.

The prime advantage of serial transmission over parallel transmission is that only a single wire is needed. This is especially important for long-distance transmission. Regardless of the amount of data transferred, only one transmission line is required. On the other hand, parallel transmission requires multiple lines, but moves information faster.

### Parallel Transmission

When binary information is transmitted in *parallel* fashion, a separate line is required for each data bit. The binary 1s and 0s are transmitted *simultaneously* but on *different* lines. Information is, therefore, moved faster.

An example of parallel transmission is shown in Figure 18. If the data to be transferred consists of eight bits (one byte), then eight separate and individual lines are used. Each line represents the specific place value of the binary number. In this example, the uppermost line represents the most significant bit. Thus, the binary 1s and 0s shown in the figure indicate that binary number 10101101 (decimal 173) is being transmitted.

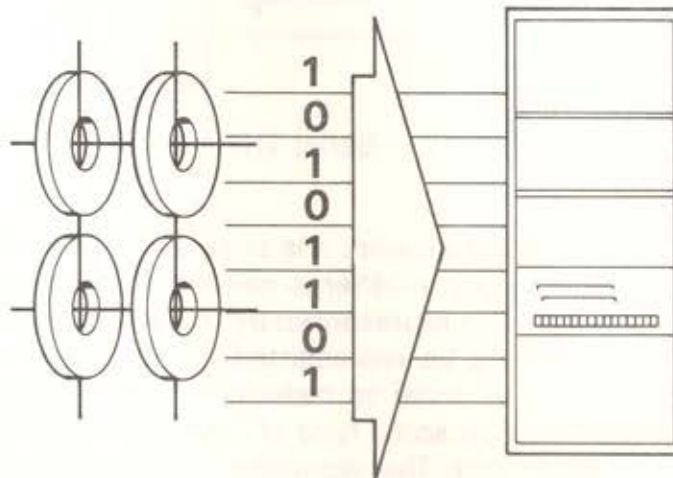


Figure 18 Parallel Transmission

Because all bits are transmitted at the same time, parallel data transmission is considerably *faster* than serial data transmission. For example, an entire 12-bit address can be sent from the CPU to main memory in a single bit time, using 12 separate bus address lines.

A comparison of serial and parallel data transmission is given in Table 1.

**Table 1    Serial Vs Parallel Transmission**

| Characteristic                          | Serial               | Parallel       |
|---|----------------------|----------------|
| No. of wires                            | One                  | Many           |
| Amount of data sent during one bit time | One bit              | Multiple bits* |
| Relative speed                          | Slow                 | Fast           |
| Typical application                     | Communication cables | Busses         |

\*The number of bits sent is limited only by the number of lines used. However, it is common to transmit either a byte or a full word during one bit time.

Because parallel transmission is faster and the cost of extra wire is small over short distances, computer busses typically operate in parallel. However, when a computer communicates with devices farther away than the same room, information is typically converted to serial form and transmitted bit by bit over a single wire or cable.

Before proceeding to the next lesson, do the exercises for this lesson.

---

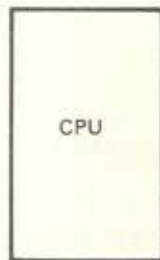
## EXERCISES

---

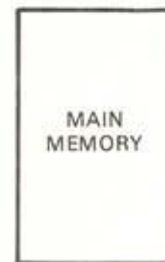
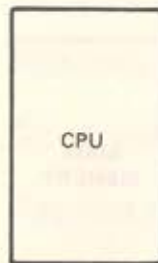
1. Assume that the binary information shown below is being transferred from the CPU to main memory. Draw the binary information in the proper form to indicate: (a) serial transmission and (b) parallel transmission. Include the required transmission lines.



- a. Serial Transmission

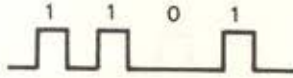


- b. Parallel Transmission

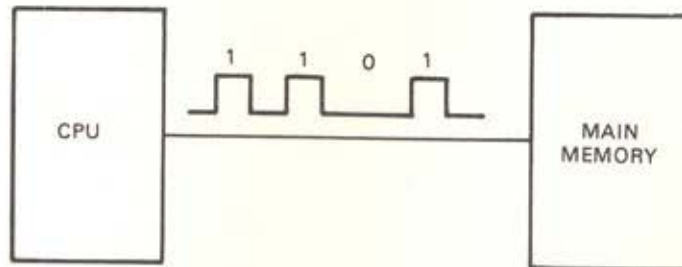


## SOLUTIONS

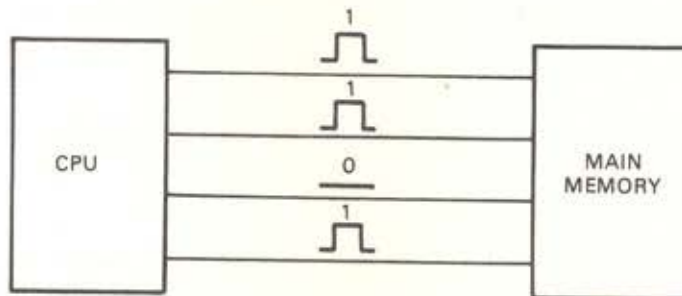
1. Assume that the binary information shown below is being transferred from the CPU to main memory. Draw the binary information in the proper form to indicate: (a) serial transmission and (b) parallel transmission. Include the required transmission lines.



- a. Serial transmission



- b. Parallel transmission

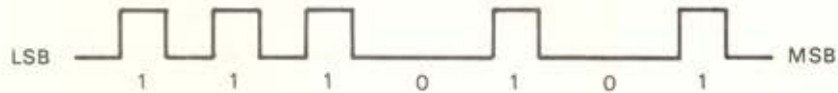


---

## EXERCISES

---

2. A binary number has been transferred out of memory in serial fashion as shown below:



Draw the same number, in serial form, as it would look if it were transferred back into memory

3. Match each of the following statements with the appropriate transmission method by placing the letter associated with the method in the parentheses next to the statement.

- a. Serial transmission
- b. Parallel transmission

- ( ) Typically used with computer busses
- ( ) Can only transfer one bit during a bit time
- ( ) Relatively slow
- ( ) Can transfer a full word during a bit time
- ( ) Requires many lines
- ( ) Typically used for communication cables
- ( ) Relatively fast

---

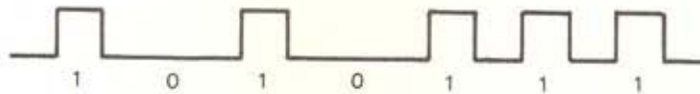
## SOLUTIONS

---

2. A binary number has been transferred out of memory in serial fashion as shown below:



Draw the same number, in serial form, as it would look if it were transferred back into memory.



**REMEMBER:** During serial transfers, the least significant bit is the first one transmitted. In effect, the number is transmitted in the reverse order when it is sent back to memory.

3. Match each of the following statements with the appropriate transmission method by placing the letter associated with the method in the parentheses next to the statement.
- a. Serial transmission
  - b. Parallel transmission
- (b) Typically used with computer busses
  - (a) Can only transfer one bit during a bit time
  - (a) Relatively slow
  - (b) Can transfer a full word during a bit time
  - (b) Requires many lines
  - (a) Typically used for communication cables
  - (b) Relatively fast

## Interfaces

### OBJECTIVE

Given the six functions of a typical interface and six descriptions, be able to match each function with its description.

### SAMPLE TEST ITEM

The six functions of a typical interface and their descriptions are given below. Match each function with its description.

| Function          | Description |
|-------------------|-------------|
| Control           | _____       |
| Buffer            | _____       |
| Status            | _____       |
| Conversion        | _____       |
| Housekeeping      | _____       |
| Program Interrupt | _____       |

#### Descriptions

- Performs specialized functions, such as updating a byte counter or current address register.
- Monitors the operational situation of the peripheral and stores the information as data. This data, such as READY and ERROR, can be acted on when the CPU is temporarily halted.
- Serves as a compensator for differences in the speeds of peripherals and the rest of the computer system.
- Governs the operation of the peripheral based on command information supplied by the software.
- Performs required data changes (e.g., serial to parallel) so that data can be transferred between the peripheral and CPU correctly.
- Halts the CPU whenever a peripheral requires some type of action from the software.

The computer user of today has a wide range of computer components to choose from when selecting a system. Options include various sizes of CPUs, different types of memories and mass storage devices, and a great range of peripheral devices ranging from terminals and printers to specialized process control and laboratory equipment. Thus, most computer systems today may be "mixed" systems, composed of a variety of different devices built by different manufacturers.

Because of the "mixed" computer system, there must be some way to make certain that all devices, even if made by different companies, are compatible and can function properly together. Normally, the mainframe (CPU and main memory) along with the associated bus configuration, dictates the characteristics required by the overall computer system. Thus, the problem becomes one of making certain that all other peripheral devices conform to these specified characteristics.

The problem is solved by using an "interface" between the bus and each peripheral device. This interface serves as a converter to ensure that the peripheral sends and receives information in the form required by the computer system characteristics.

Mark your place in this workbook and view Lesson 3 of the audio-visual program "Bus Structure."

In a typical computer system there are various types of peripheral devices and, therefore, various types of interfaces. Each interface permits the associated peripheral to communicate with the central processor over a bus shared by other devices.

A typical interface usually has *six* functions to perform in order to ensure proper operation. These functions are:

1. **Control** – The interface controls the operation of the peripheral based on command information supplied by software.
2. **Buffer** – The interface serves as a buffer between the peripheral and the rest of the computer system to compensate for differences in the speeds of various devices.
3. **Status** – The interface monitors the operational status of the peripheral and stores information as status data. Status data, such as READY, DONE, and ERROR, can be acted on when the CPU interrogates the peripheral device.

4. **Conversion** – The interface performs any required data conversions (for example, parallel to serial or serial to parallel) so that data can be transferred between the peripheral and the CPU correctly.
5. **Housekeeping** – The interface may perform specialized functions, such as updating a byte counter or current address register if necessary.
6. **Program Interrupt** – The interface issues an interrupt to the CPU whenever the peripheral requires some type of action from the software. For example, if the peripheral has completed an operation or an error condition exists in the peripheral, the interface issues an interrupt.

In general, an interface is connected to only one peripheral and translates signals between the peripheral and a bus while using commands from the program to control the operations within the peripheral. For this reason other terms such as "control unit," "controller," "adapter," or "synchronizer" are often used to describe interfaces.

There are two groups of interfaces – one for non-DMA peripherals (as shown in Figure 19) and one for DMA peripherals (as illustrated in Figures 20 and 21).

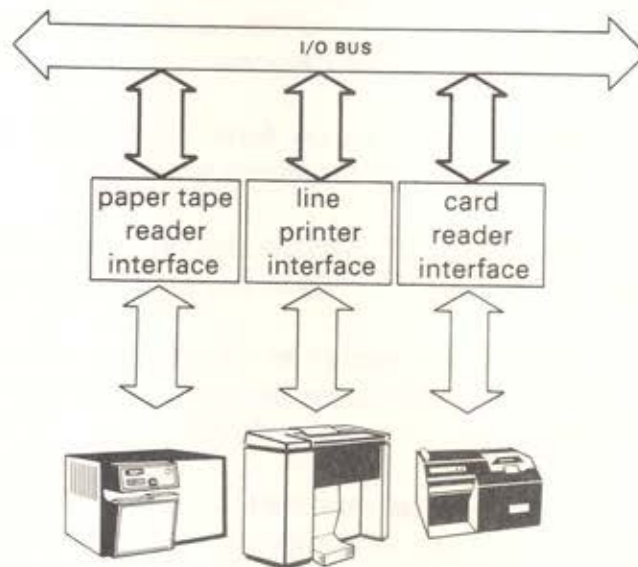


Figure 19 Interfaces for Non-DMA Peripherals

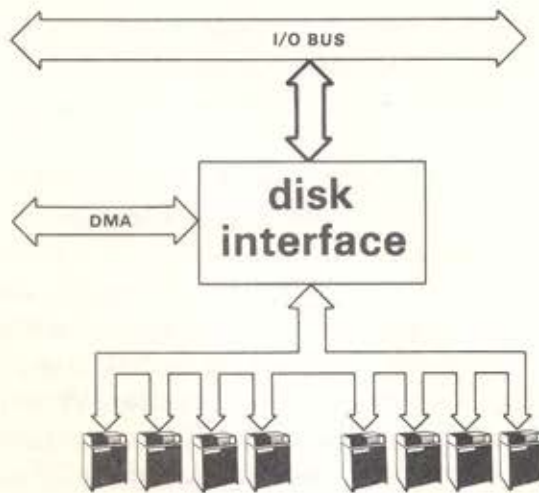


Figure 20

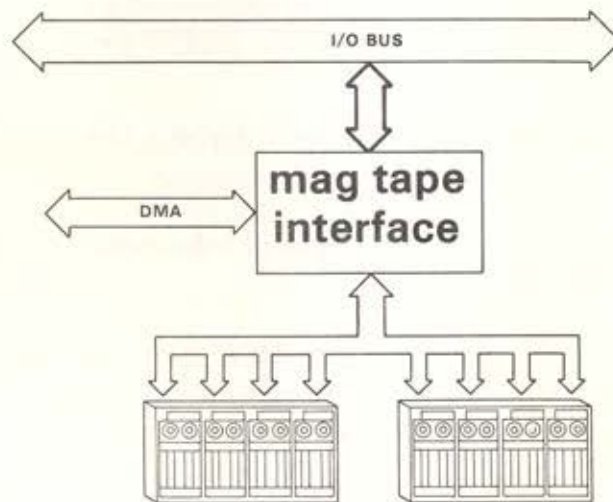


Figure 21

The non-DMA type interface is used to connect slow peripherals, such as terminals and line printers, while the DMA type interface connects high-speed magnetic tape and disk peripherals to the bus.

A non-DMA interface generally works on the principle of supply and demand. An example of this principle is the output of data from the computer to the line printer. The program commands the computer to send data to the line printer. The computer sends the data in parallel mode to the interface. Then the interface *converts* the data to the signals the line printer requires. These data signals are sent with a control signal that loads the data into the line printer. When the operation is complete, the line printer sends a READY signal to the interface that demands more data or another command. Figure 22 shows a simplified block diagram of a non-DMA interface.

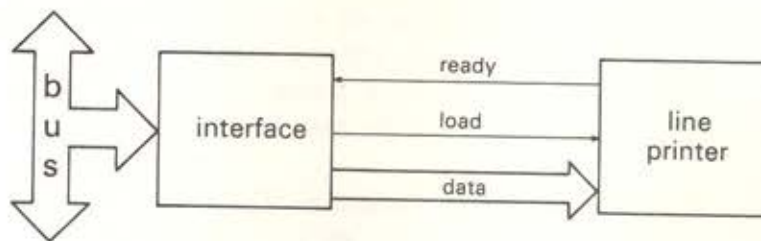


Figure 22 Non-DMA Interface Diagram

For each transfer of data, one byte of information is sent to the line printer and loaded into its buffer.

On the other hand, the DMA interface performs these functions as well as others. A DMA interface also performs housekeeping functions, such as keeping track of the address where the data is located in the peripheral and the location where the data is to be sent.

In addition, the DMA interface tells the peripheral either to read or write, while monitoring the status of the peripheral (READY, DONE, ERROR, etc.) and transferring data between the peripheral and the bus. Figure 23 shows a block diagram of a simplified DMA interface.

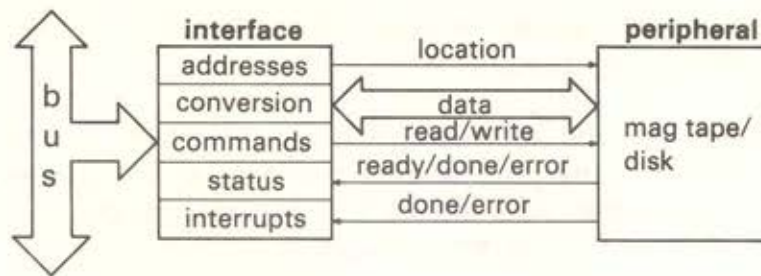


Figure 23 Simplified DMA Interface

The transfer of data between a DMA-type peripheral (magnetic tape or disk) and the bus is really quite simple if each function is examined individually. First, all peripherals are arranged along the bus much like houses along a street, each having its own address or group of addresses. When the processor requires the use of a peripheral, it sends the address of the desired peripheral down the bus, and the interface of the peripheral with the *same* address replies to that address.

Once the interface is addressed, a command word is loaded into its command register. The command word tells the control logic the type of operation to be done. Figure 24 illustrates this operation.

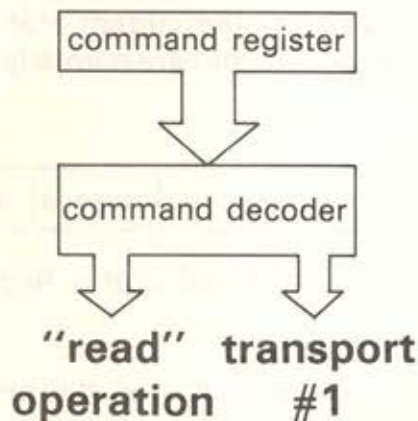


Figure 24 Command Decoder

The control logic then issues the necessary commands to the peripheral, such as move tape, read, or write.

At the same time, the addressing logic in the interface is looking for a specific location in the peripheral to move data. It also specifies the address for each byte of data it transfers. Meanwhile, the status of the peripheral is reflected in the status register.

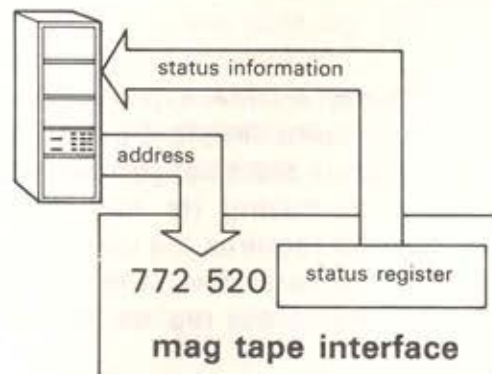


Figure 25 Status Information

In all cases, the BUSY, READ, or WRITE, and direction bits will be set along with other bits giving the current status of the peripheral. Figure 26 shows the types of bits that are normally set in a status word.

|      |       |      |      |         |         |      |       |       |
|------|-------|------|------|---------|---------|------|-------|-------|
| DONE | WRITE | READ | FAST | REVERSE | FORWARD | BUSY | READY | ERROR |
|------|-------|------|------|---------|---------|------|-------|-------|

Figure 26 Simplified Status Register

When the transfer of data is complete, the peripheral will set the DONE bit in the status register, causing the interface to generate an interrupt signal to the processor. Figure 27 shows this operation.

Keep in mind that once the necessary commands are given to the interface by the processor, it is free of the peripheral and can perform other functions.

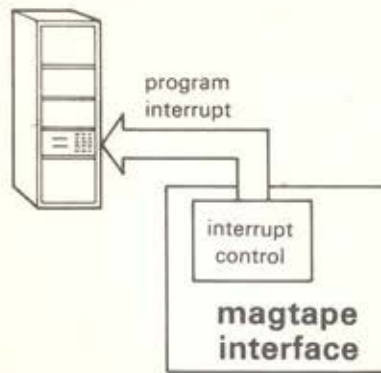


Figure 27 Program Interrupt

In summary, a DMA interface coordinates all necessary commands, addresses, data transfers, sequencing, monitoring, and error detecting required to complete an operation before it is necessary to interrupt the CPU. On the other hand, a non-DMA interface requires the CPU to issue commands for *each* operation.

---

## EXERCISE

---

Match each of the following terms with its appropriate definition. Place the letter(s) of the term next to the definition. (Note that a single definition may apply to more than one term.)

- |                      |       |   |
|----------------------|-------|---|
| a. Control unit      | _____ | Perform specialized functions such as updating counters                                 |
| b. Data buffer       |       |   |
| c. Conversion        |       |   |
| d. Status data       | _____ | Another name for "interface"  |
| e. Command           |       |   |
| f. Housekeeping      |       |   |
| g. Controller        | _____ | The process of notifying the CPU that the peripheral device needs some action performed |
| h. Adapter           |       |   |
| i. Program interrupt |       |   |
| j. Synchronizer      | _____ | Compensate for speed differences between system components                              |
|                      | _____ | Information such as READY, DONE, and ERROR  |
|                      | _____ | Changing from one type of transmission method to another                                |
|                      | _____ | Orders sent from the CPU to an interface  |

---

## SOLUTION

---

Match each of the following terms with its appropriate definition. Place the letter(s) of the term next to the definition. (Note that a single definition may apply to more than one term.)

- |                      |                |  |
|----------------------|----------------|--|
| a. Control unit      | <u>f</u>       | Perform specialized functions such as updating counters                                  |
| b. Data buffer       |                |  |
| c. Conversion        |                |  |
| d. Status data       | <u>a,g,h,j</u> | Another name for "interface"   |
| e. Command           |                |  |
| f. Housekeeping      |                |  |
| g. Controller        | <u>i</u>       | The process of notifying the CPU that the peripheral device needs some action performed. |
| h. Adapter           |                |  |
| i. Program interrupt |                |  |
| j. Synchronizer      | <u>b</u>       | Compensates for speed differences between system components                              |
|                      | <u>d.</u>      | Information such as a READY, DONE, and ERROR   |
|                      | <u>c</u>       | Changing from one type of transmission method to another                                 |
|                      | <u>e</u>       | Orders sent from the CPU to an interface   |

Take the test for this module and evaluate your answers before studying another module.