Introduction to Minicomputers

I/O Techniques

digital

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

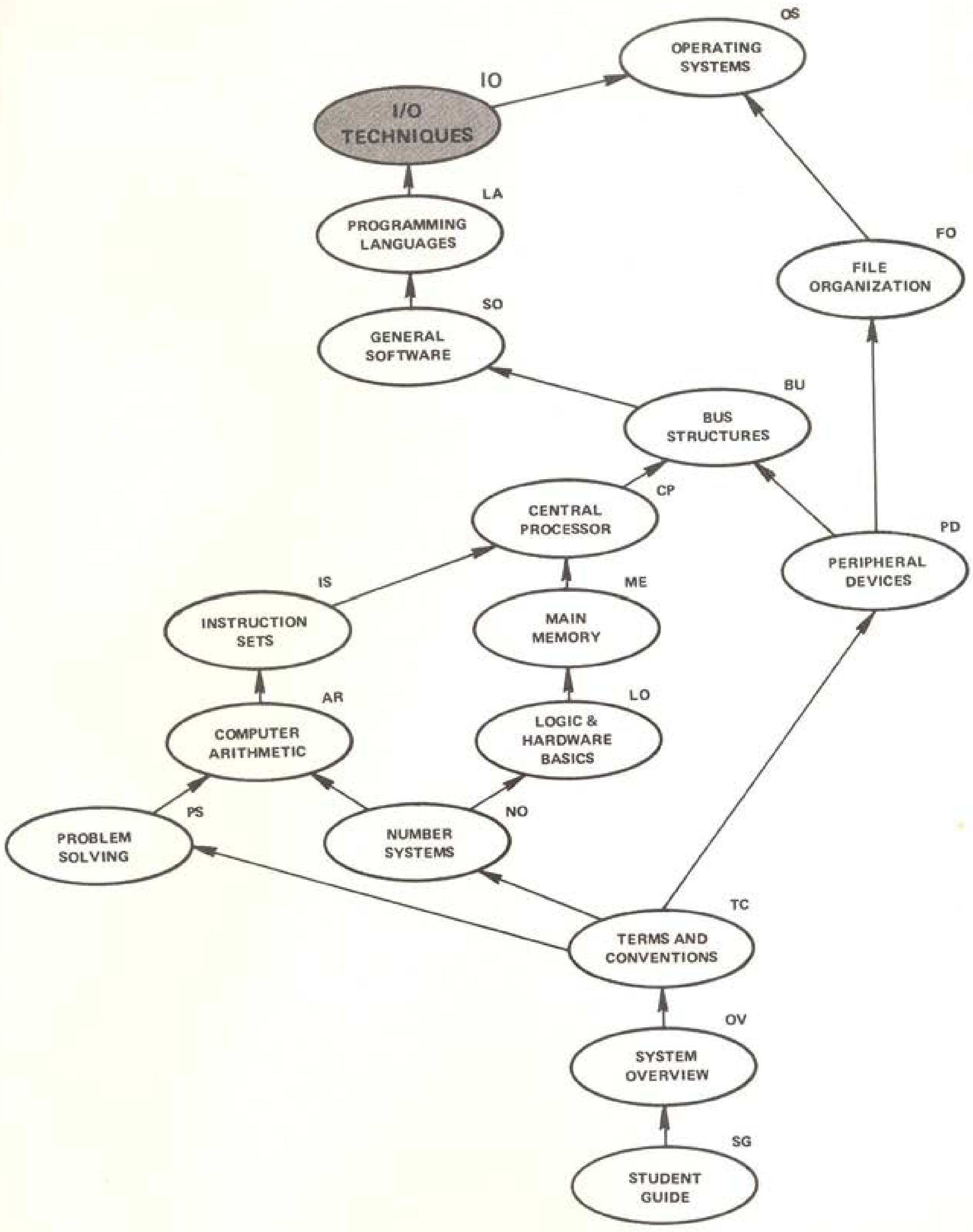
INTRODUCTION TO MINICOMPUTERS

I/O Techniques

Student Workbook

Audio-Visual Course by Digital Equipment Corporation

COURSE MAP



OS
OPERATING SYSTEMS
IO
I/O TECHNIQUES
LA
PROGRAMMING LANGUAGES
FO
FILE ORGANIZATION
SO
GENERAL SOFTWARE
BU
BUS STRUCTURES
CP
CENTRAL PROCESSOR
PD
PERIPHERAL DEVICES
IS
INSTRUCTION SETS
ME
MAIN MEMORY
AR
COMPUTER ARITHMETIC
LO
LOGIC & HARDWARE BASICS
PS
PROBLEM SOLVING
NO
NUMBER SYSTEMS
TC
TERMS AND CONVENTIONS
OV
SYSTEM OVERVIEW
SG
STUDENT GUIDE

CONTENTS

I/O Techniques

Introduction

An essential feature of any computer is the ability to perform input and output operations. Without this ability, the computer would require data to be manually stored and retrieved from specific memory locations using the console switches and lights. This laborious, primitive, and error-prone method can be eliminated by employing various input/output (I/O) techniques or methods.

I/O techniques provide the programmer with relatively easy ways of transferring data between the CPU, main memory, and the computer's peripheral devices. Communication among these various hardware items is accomplished over one or more bus structures. A typical configuration of busses and hardware items is shown in Figure 1. Notice that there is always at least one pathway for data transfer between any two computer components. Some devices (such as terminals and line printers) require the intervention of the CPU in all data transfers involving main memory, while other devices (such as disk and magnetic tape units) may be able to access main memory directly without CPU intervention.

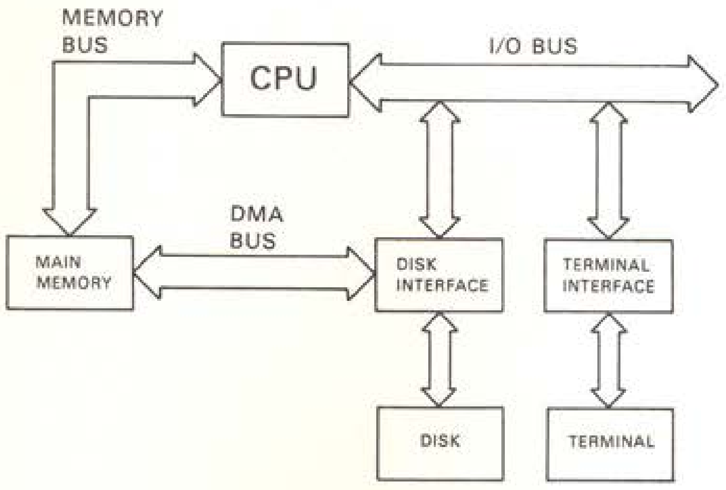


Figure 1 A Typical Three-Bus Computer Configuration

This module discusses three major I/O techniques. Each technique involves at least the following operations:

- *Initiation of the process* by either hardware or software
- *Control of the process* by either hardware or software
- *Specification of the data* including addresses, types, and counts

Each technique will be discussed in a separate lesson. Before proceeding to Lesson 1, you may wish to review information on the following subjects:

- Main Memory
- Central Processor
- Peripheral Devices
- Bus Structures
- General Software

Programmed Data Transfers

OBJECTIVES

1. Given a list of the five steps of a programmed data transfer and five definitions, be able to: (1) match each step with its definition, and (2) label the position of each step in chronological order.

2. Given four criteria for a programmed data transfer and four descriptions, be able to match each criterion with its description.

SAMPLE TEST ITEMS

1. The five steps of a programmed data transfer and five definitions are given below. Match each step with its definition. Then, in the column labeled Chronological Order, write a "1" next to the first step to be completed, a "2" next to the second step, etc., until the last step is reached.

Steps	Definitions	Chronological Order
Ready Test	_____	_____
Ready Flag Reset	_____	_____
•	•	•
•	•	•
•	•	•

Definitions

a. Set to 0 immediately after the INPUT instruction has been exhausted.

b. Necessary because the CPU is much faster than peripheral devices.

•
•
•

Mark your place in this workbook and view Lesson 1 in the A/V program, "I/O Techniques."

Programmed data transfers are the simplest form of I/O technique because the software both *initiates and controls* the process of transfer. The hardware's responsibilities are then confined to merely performing the necessary operations. A sample programmed data transfer is shown in Figure 2.

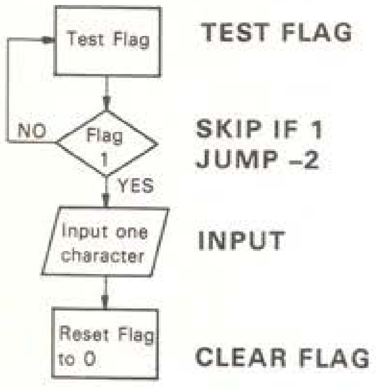


Figure 2 Sample Programmed Data Transfer

A typical programmed data transfer consists of five steps:

1. **Ready Test** – A flip-flop, called the *ready flag* or ready bit, is tested to determine if the appropriate device is ready. A value of 0 indicates that the device is *not* ready; a value of 1 indicates the device is ready for the next operation. Device readiness must be tested because the CPU is much faster than peripheral devices. Hence the CPU must frequently check to see if the device has "caught up."

2. **Conditional Loop** – The test is followed by a *conditional skip instruction.* If the ready flag is 1 (device ready), the skip is taken, and the program proceeds to the next step. If the ready flag is 0, a branch causes the test instruction to be re-executed. The computer, therefore, waits for a slow device by continually testing the readiness of the device until it reports itself ready.

3. **I/O Operation** – When the I/O device is ready, the actual operation can be performed. In the example of Figure 2, a single character is being INPUT from the teleprinter.

4. **Ready Flag Reset** – Immediately after the INPUT instruction has been executed, the CPU resets the ready flag to zero (not ready). Thus, the CPU resets a device flag to *not ready,* and the device *interface* sets it to one (ready) after the device has completed its I/O operation.

5. **Process Recycle** – Finally, if the I/O operation is to be immediately repeated, the program loops back to the test instruction to wait until the device is again ready.

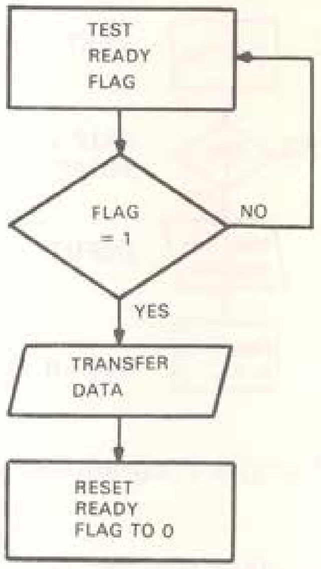


Figure 3 Sequence of a Programmed Data Transfer Technique

The steps listed above and illustrated in Figure 3 must be executed each time the operation is to be performed. In the example in Figure 2, each character typed on the teleprinter will require one execution of the steps.

When a sequence of steps, such as in Figure 2, is repeated at different points in a program, the sequence is frequently written as a subroutine in order to reduce programming time and memory usage. In general, there will be one subroutine for each different I/O operation required.

Advantages and Disadvantages

Programmed data transfers have one primary advantage – they allow simple hardware interfaces because most of the management of the I/O operations is performed by the software. Under some circumstances, a subsidiary benefit may be the greatly enhanced ability of the programmer to exercise precise control of the I/O operations. However, this is seldom an important consideration.

There are two significant disadvantages to the programmed data transfer technique. First, valuable CPU time is wasted while the CPU waits for the device to become ready. Secondly, this technique is not efficient because three memory cycles are required to move a word between a peripheral device and main memory. Therefore, programmed data transfers are generally used only with slow devices (such as teleprinters, paper tape equipment, and slow card readers), which normally handle one character or byte at a time. Higher speed devices (such as disks, magnetic tapes, and fast card readers), which handle large amounts of data, transfer data using more efficient methods described in the following lessons.

EXERCISES

1. List one advantage and one disadvantage of the programmed data transfer method.

2. Does the hardware or the software *initiate* a programmed data transfer?

3. Does the hardware or the software *control* a programmed data transfer?

4. List the major steps in a programmed data transfer sequence.

SOLUTIONS

1. List one advantage and one disadvantage of the programmed data transfer method.

   Advantage: Allows simpler hardware interfaces.

   Disadvantages:

   a. Wastes CPU time.

   b. Inefficient for large data transfers between memory and high-speed storage devices.

2. Does the hardware or the software *initiate* a programmed data transfer?

   Software

3. Does the hardware or the software *control* a programmed data transfer?

   Software

4. List the major steps in a programmed data transfer sequence.

   a. **Test Ready Flag** (1 = device ready; 0 = device not ready)

   b. **Conditional Loop** (continue looping until device is ready)

   c. **I/O Operation** (perform the actual data transfer)

   d. **Ready Flag Reset** (CPU marks the device not ready)

   e. **Process Recycle** (go back to test the ready flag if another transfer is immediately required)

Program Interrupt Data Transfers

OBJECTIVES

1. Given a list of the five steps of a program interrupt data transfer and five definitions, be able to: (1) match each step with its definition, and (2) label the position of each step in chronological order.

2. Given seven statements referring to methods of establishing device priorities, be able to label those that refer to the polling method and those that refer to the multiple interrupt levels method.

3. Given four criteria for a program interrupt data transfer and four descriptions, be able to match each criterion with its description.

SAMPLE TEST ITEMS

1. The five steps of a program interrupt data transfer and five definitions are given below. Match each step with its definition. Then, in the column labeled Chronological Order, write a "1" next to the first step to be completed, a "2" next to the second, etc., until the last step is reached.

Step	Definition	Chronological Order
Register Restored	____	____
Register Saved	____	____
•	•	•
•	•	•
•	•	•

Definitions

a. Current program data is put aside in memory so that the CPU registers may be used during the data transfer.

b. CPU registers returned to their status at the time of the program interrupt.

•
•
•

2. Indicate that each of these statements refers to the polling method (P) or the multiple interrupt levels method (M) of establishing device priorities by writing the correct letter in the space provided.

Statement	Method
Implemented by hardware via I/O bus.	____
Involves more costly hardware than the other method.	____
•	•
•	•
•	•

Mark your place in this workbook and view Lesson 2 in the A/V program, "I/O Techniques."

Program interrupts are a more efficient I/O technique than programmed data transfers because the CPU does not waste time waiting for a device to become ready. Rather, the CPU executes the current program until the device interface signals that the device is ready. Teleprinter mechanisms process each character and signal that they are ready for the next in approximately one tenth of a second. Computers, on the other hand, move a character between memory and the teleprinter interface in several microseconds. This means that when using programmed data transfers, the processor may execute the wait loop approximately 10,000 times between characters unless the programmer cleverly arranges other instructions for the processor to do. The use of program interrupts makes it easy for the programmer to provide useful instructions for the processor to execute while it is waiting. Remember that program interrupt data transfers are initiated by the hardware. Control of the transfer is maintained by a special program called a device handler. Thus, a program interrupt data transfer is controlled by software.

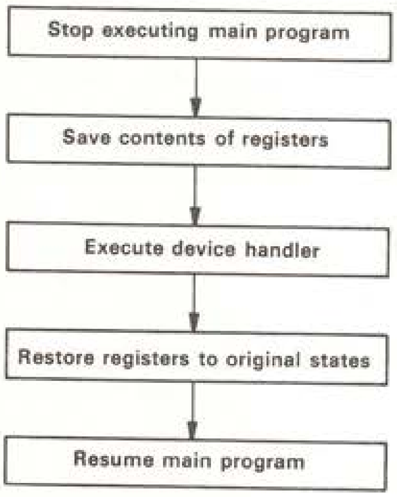

Stop executing main program
Save contents of registers
Execute device handler
Restore registers to original states
Resume main program

Figure 4 Sample Program Interrupt Data Transfer

A typical program interrupt data transfer is shown in Figure 4, and consists of five straightforward steps, which are listed below and illustrated in Figure 5.

1. **Program Interrupted** – The CPU detects an interrupt signal on the I/O bus from some device interface. Following the completion of the current instruction, the CPU stops executing the program.

2. **Register Saved** – The current contents of the CPU's registers are stored in memory so that the registers may be used during the data transfer, without destroying data from the program.

3. **Device Handler Executed** – The CPU goes to a fixed location in memory to obtain the address of the starting location of the device handler. This fixed location pointing to the device handler is called an *interrupt vector.* The CPU then jumps to the indicated address and begins executing the device handler instructions.

4. **Register Restored** – Upon completion of the device handler, the CPU's registers are restored to their status at the time of the interrupt.

5. **Normal Execution Resumed** – Finally, execution is returned to the point where is was interrupted. Thus, the program is executed as if nothing had happened during the execution of the data transfer.

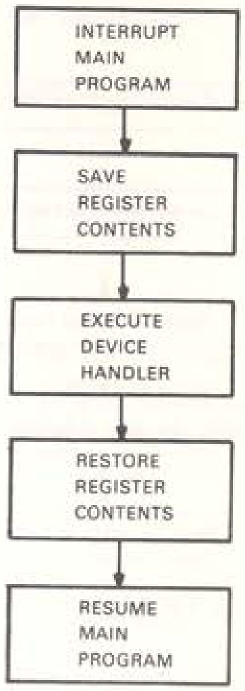


Figure 5 Sequence of an Interrupt Data Transfer Technique

Advantages and Disadvantages

There are two disadvantaqes to the program interrupt data transfer technique. First, this technique, as with the programmed data transfer technique, is not efficient for large blocks of data because too much time is spent controlling the transfer. Remember that although the hardware initiates the interrupt, the software (and hence the CPU) still controls the transfer operation. Secondly, the device interfaces are more complex for this technique because additional logic must be built to initiate the interrupt process. Hardware costs, therefore, tend to rise.

There are, however, two significant advantages to counterbalance these drawbacks. Of primary importance is the fact that CPU time is more efficiently used because no time is wasted waiting for a device to become ready. A program is interrupted only when a device is ready. At all other times, the CPU can be devoted to internal processing. A second advantage is the ability to establish individual priorities for peripheral devices. This aspect of the program interrupt data transfer technique is discussed next.

Polling Tables and Multiple Interrupt Levels

The ability to establish device priorities is an important one. High-speed storage devices such as disk and magnetic tape units require very quick response to their interrupt request or data may be lost. In contrast, low-speed devices can usually wait *without* losing data. Therefore, devices such as teleprinters and card readers are assigned low priorities. Each device, regardless of type, is assigned a unique interrupt priority in order to eliminate a conflict when simultaneous interrupt requests occur. With unique priorities, the device with the higher priority will always be serviced first.

One method of establishing priorities is through the use of a *polling table* (Figure 6). This is simply a list of all the peripheral devices in descending order by priority. When an interrupt request is detected, the CPU polls or questions each device to find out which one needs servicing. Because the table is ordered by priorities, those devices with the highest priorities will always be polled first. Therefore, interrupt servicing of them will be the most rapid. Devices (such as teleprinters) at the low end of the polling table will be serviced only if no other device (of higher priority and earlier in the list) requires service. When polling tables are used, they identify the interrupting device and call the appropriate device handler. Therefore *the priority structure is part of the software* and may be changed easily.

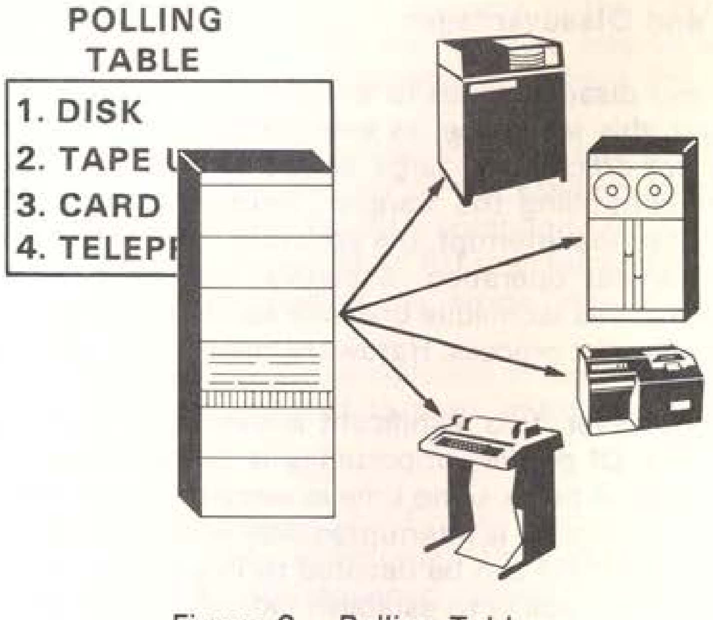


Figure 6 Polling Table

The second common method of establishing priorities is the use of *multiple interrupt levels* (Figure 7). In this case, the I/O bus has several interrupt lines rather than one, and each interrupt line is associated with a unique priority level. The priority of an individual peripheral device is then determined by which interrupt line is connected to the device's interface. The CPU also has an interrupt level, but unlike the peripheral devices (which have fixed, hardware priority levels), *the CPU has a variable priority level which can be controlled by the software.* There is a simple reason for this. Once the CPU begins servicing an interrupt request, it should not be interrupted unless a higher priority device needs service. Therefore, when multiple interrupt levels are used, the device handler assigns the CPU a priority level, enabling it to ignore requests from devices of equal or lower priority than the one currently being serviced. When the operation is complete, the CPU's priority level is restored to its previous level.

The choice between polling tables and multiple interrupt levels is one of weighing the somewhat slower polling table procedure against the more costly hardware required for multiple interrupt levels. Each method is useful, and some machines even combine both methods to establish a more complex system of assigning priority levels to peripheral devices.

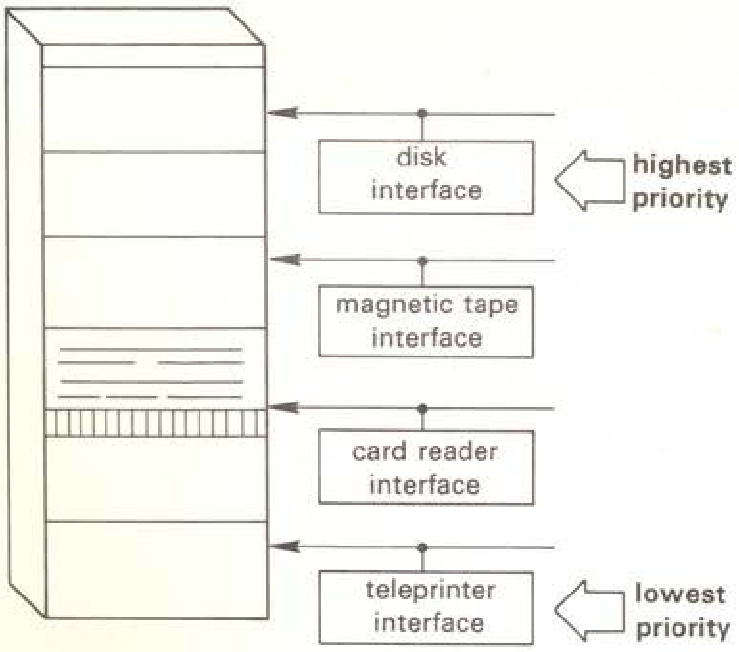


Figure 7 Multiple Interrupt Levels

EXERCISES

1. List one advantage and one disadvantage of the program interrupt data transfer method.

2. Does the hardware or the software *initiate a program interrupt data transfer?*

3. Does the hardware or the software *control a program interrupt data transfer?*

SOLUTIONS

1. List one advantage and one disadvantage of the program interrupt data transfer method.

   Advantages:

   a. Is less wasteful of CPU time because the CPU does not have to wait for a device to become ready.

   b. Allows the establishment of priorities for the servicing of peripheral devices.

   Disadvantages:

   a. Uses more expensive hardware because the interfaces must have more logic.

   b. Not an efficient technique for transferring large blocks of data.

2. Does the hardware or the software *initiate a program interrupt data transfer?*

   Hardware (the device interface)

3. Does the hardware or the software *control a program interrupt data transfer?*

   Software (the device handler)

EXERCISES

4. List the major steps in a program interrupt data transfer sequence.

5. Briefly describe the differences between *polling* and *multiple interrupt levels* as methods of establishing priorities.

4. List the major steps in a program interrupt data transfer sequence.

   a. **Program Interrupted**
   (CPU detects interrupt request from a device interface and stops executing the present program.)

   b. **Register Saved**
   (The present program's register contents are stored in memory for later use.)

   c. **Device Handler Executed**
   (The interrupting device is identified, and the appropriate device handler services the request.)

   d. **Registers Restored**
   (The CPU's registers are restored to their status at the moment of the interrupt.)

   e. **Normal Execution Resumed**
   (The original program begins executing at the point where it was interrupted.)

5. Briefly describe the differences between *polling* and *multiple interrupt levels* as methods of establishing priorities.

   Polling is a method using a software table in which each device is listed in descending order of priority. When an interrupt request is detected, the CPU tests each device in order to see which one needs service.

   Multiple interrupt levels are accomplished by sharing several interrupt lines on the I/O bus. A device's priority is fixed by connecting a specific interrupt line to the device's interface. The CPU has a variable priority level to enable it to ignore low-priority interrupt requests when servicing a higher priority one.

   Both methods achieve the same results – *providing the quickest service to the devices that need it most.*

EXERCISES

6. Define:

   a. **Interrupt vector**

   b. **Device handler**

6. Define:

   a. **Interrupt vector** – A fixed location in memory which contains the starting address of a device handler.

   b. **Device handler** – A special program that identifies which peripheral device initiated the program interrupt and then services that device.

DMA Data Transfers

OBJECTIVES

1. Given a list of the seven steps of a DMA data transfer and seven definitions, be able to: (1) match each step with its definition, and (2) label the position of each step in chronological order.

2. Given four criteria for a DMA data transfer and four descriptions, be able to match each criterion with its description.

SAMPLE TEST ITEMS

1. The seven steps of a DMA data transfer and seven definitions are given below. Match each step with its definition by writing the correct letter in the space provided. Then, in the column labeled Chronological Order, write a "1" next to the first step to be completed, a "2" next to the second step, etc.

Step	Definition	Chronological Order
Program Initiates Transfer	_____	_____
Signal Completion	_____	_____
Test Device Readiness	_____	_____
•	•	•
•	•	•
•	•	•

Definitions

a. Interface informs the CPU that the transfer has been completed and the device is free.

b. Only step *not* performed by the interface.

c. Interface performs essentially the same check as described for programmed data transfers.

•
•
•

Mark your place in this workbook and view Lesson 3 in the A/V Program, "I/O Techniques."

The I/O techniques discussed in the two previous lessons are inefficient for transferring large blocks of data. A third technique, *direct memory access* (DMA), does provide an efficient means of performing such transfers. Figure 8 depicts the DMA data transfer process.

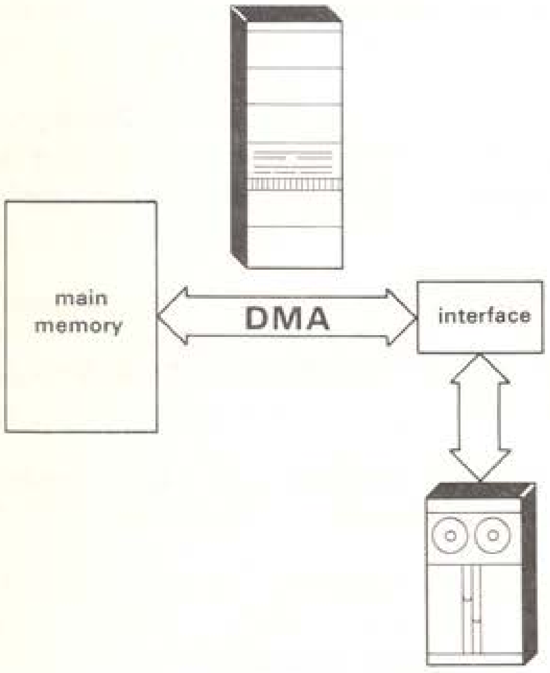


Figure 8 DMA Data Transfers

DMA data transfers occur *directly between main memory and high-speed storage devices* such as disk and magnetic tape units. Frequently the transfer occurs on a separate DMA bus and does not involve the CPU. A DMA transfer is *initiated by a program,* but the actual *control* of the transfer is exercised by the device's *interface.* The CPU is, thus, free to perform independent processing operations while the transfer is taking place. Figure 9 shows a sample of the DMA data transfer.

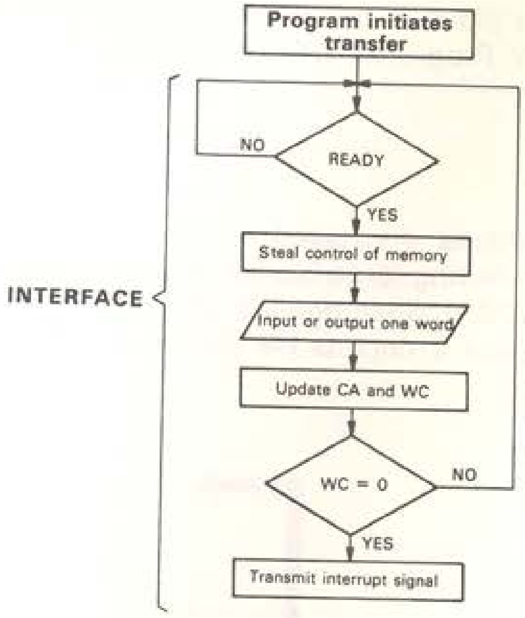


Figure 9 Sample DMA Data Transfer

The DMA Transfer Process

A typical DMA data transfer requires the following steps, which are illustrated in Figure 9:

1. **Program Initiates Transfer** – A DMA transfer is *initiated by a program when it specifies:*

   - The device to be used
   - The operation (input or output)
   - The word count
   - The current address

   The word count is the number of words to be transferred in this operation, and the current address is the location of the first word in memory to be affected by the transfer. All subsequent steps are performed by the interface.

2. **Test Device Readiness** – The interface performs essentially the same test as was described for programmed data transfers. When the device is ready, the interface proceeds to the next step.

3. **Steal Memory Control** – The interface next temporarily takes control of memory from the CPU. At this point, the CPU cannot utilize memory.

4. **Transfer One Word** – One word is transferred between memory and the device. The direction of the transfer is determined by which operation was specified by the program. The memory location involved in the transfer is indicated by the value of the current address. Memory control is immediately returned to the CPU once the transfer is complete.

5. **Update Control Parameters** – The interface updates the word count (WC) to reflect the transfer of a word. At the same time, the current address (CA) is also updated to point at the next word to be affected.

6. **Test Completion** – The word count is tested for zero, the indication that the entire transfer has been completed. If there are remaining words to be transferred, the interface goes back to step 2 to wait for the device to become ready again.

7. **Signal Completion** – When the word count is zero, indicating the transfer is complete, the interface transmits an interrupt signal to the CPU via the I/O bus to indicate that the transfer has been completed and the device is free.

Advantages and Disadvantages

The primary advantage of the DMA data transfer technique is that it provides the efficient transfer of data between storage devices and memory without involving the CPU.

However, hardware costs are greater. The interface must be both sophisticated and expensive to perform the several logical functions required of it. On many machines, use of DMA techniques also requires the addition of a DMA bus. Despite the hardware costs required by the DMA technique, it is nearly always used for large transfers as it is the most efficient alternative.

EXERCISES

1. List one advantage and one disadvantage of the direct memory access data transfer method.

2. Does the hardware or the software *initiate a DMA transfer?*

3. Does the hardware or the software *control a DMA transfer?*

SOLUTIONS

1. List one advantage and one disadvantage of the direct memory access data transfer method.

   Advantage: Only efficient way to transfer large blocks of data from high-speed storage devices to main memory.

   Disadvantages:

   a. Requires expensive interfaces because of complex logic.

   b. Not efficient for transfers involving small amounts of data.

2. Does the hardware or the software *initiate a DMA transfer?*

   Software

3. Does the hardware or the software *control a DMA transfer?*

   Hardware

EXERCISES

4. List the major steps in a DMA data transfer sequence.

4. List the major steps in a DMA data transfer sequence.

   a. **Program Initiates Transfer**

      Specifies

      - Which device
      - Which operation
      - Word count (how much data)
      - Current address (what memory location to use)

   b. **Test Device Readiness**

      Continues testing until device is ready.

   c. **Steal Memory Control**

      CPU temporarily relinquishes memory control to the interface.

   d. **Data Transfer**

      The operation specified by the program is performed on a single word.

   e. **Update Control Parameters**

      The word count and current address are updated to reflect the transfer of one word.

   f. **Test Completion**

      If the word count is zero, the transfer is done. If not, go back to step b.

   g. **Signal Completion**

      When the word count is zero, the interface transmits an interrupt signal to the CPU over the I/O bus to inform the CPU that the transfer has been completed.

EXERCISES

5. Define:

   a. **Word Count**

   b. **Current address** (in the context of DMA transfers)

5. Define:

   a. **Word Count** – A parameter initialized by the software when the DMA transfer is initiated. It indicates the number of words remaining to be transferred. The word count is updated each time a word is transferred.

   b. **Current Address** – Another parameter initialized by the software, the current address is the memory location to be used during the transfer. Like the word count, the current address is updated after each word is transferred.

Final Review

Table 1 compares the three I/O techniques discussed in this module.

Table 1 Comparison of I/O Techniques

Criterion	Programmed Data Transfers	Program Interrupts	DMA
Advantages	Allows simple hardware	CPU does not wait for device Allows priorities to be established	Only efficient way to transfer large data blocks
Disadvantages	Wastes CPU time Inefficient for large data blocks	Hardware is more expensive Still not efficient for large data blocks	Hardware is very expensive Inefficient for small amounts of data
Initiated by	Software	Hardware	Software
Controlled by	Software	Software	Hardware
CPU Utilization	High	Moderate	Minimal
CPU Efficiency	Low	Good	Excellent

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