Chapter 7

Input/Output

William Stallings , Computer Organization and Architecture, 9th Edition

Objectives

- Why are peripherals not connected directly to the system bus?
- Why IO module is needed?
- How to control IO devices?
- How to increase IO operations?
- After studying this chapter, you should be able to:
  - Explain the use of I/O modules as part of a computer organization.
  - Understand the difference between programmed I/O and interrupt-driven I/O and discuss their relative merits.
  - Present an overview of the operation of direct memory access (DMA).
  - Explain the function and use of I/O channels.

Contents

Why are devices not connected to system bus?

- There are a wide variety of peripherals with various methods of operation. It would be impractical to incorporate the necessary logic within the processor to control a range of devices.
- The data transfer rate of peripherals is **often much slower** than that of the memory or processor. Thus, it is impractical to use the high-speed system bus to communicate directly with a peripheral.
- The data transfer rate of some peripherals **can be faster** than that of the memory or processor. Again, the mismatch would lead to inefficiencies if not managed properly.
- Peripherals often use **different data formats** and **word lengths** than the computer to which they are attached.

Generic Model of an I/O Module

Why an IO module is needed?

- Interface to the processor and memory via the system bus or central switch

- Interface to one or more peripheral devices by tailored data links

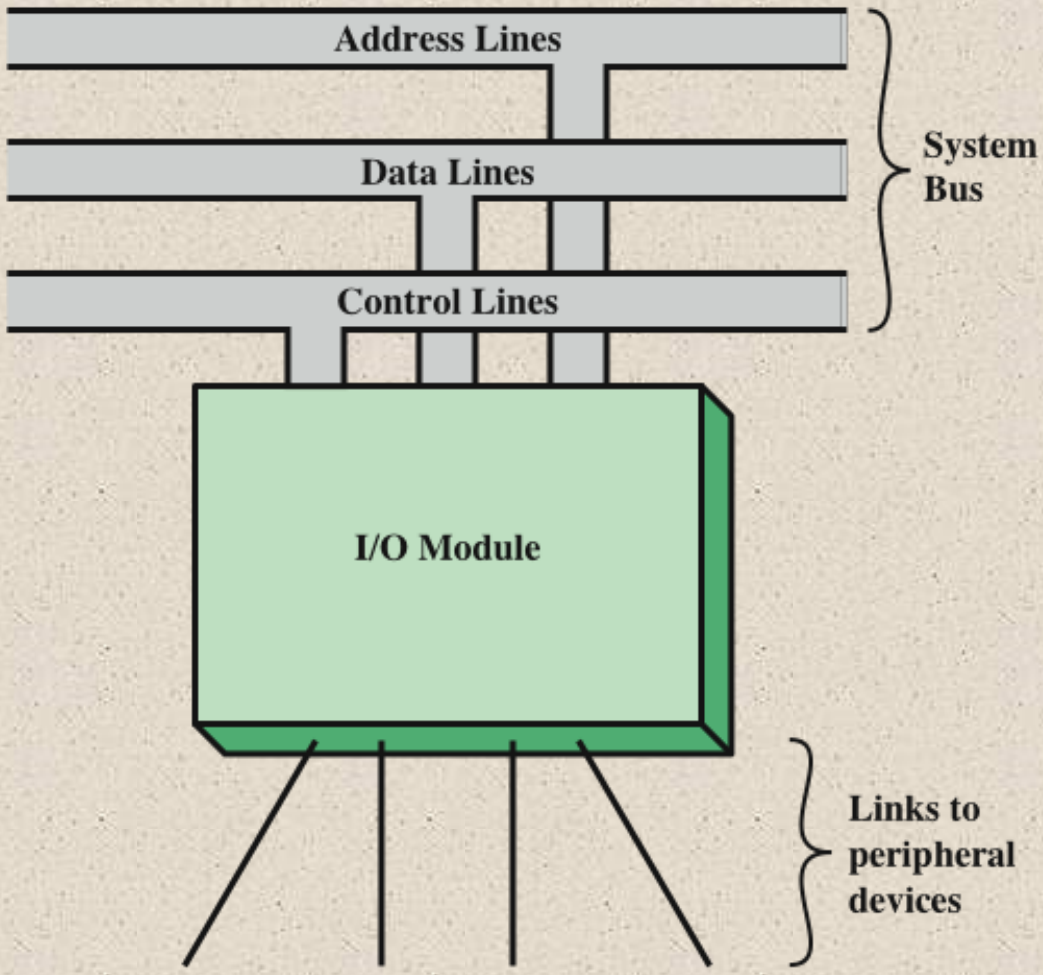


Figure 7.1 Generic Model of an I/O Module

7.1- External Devices

- Provide a means of exchanging data between the external environment and the computer
- Attach to the computer by a link to an I/O module
  - The link is used to exchange control, status, and data between the I/O module and the external device
- *peripheral device*
  - An external device connected to an I/O module

- Three categories:
- Human readable
  - Suitable for communicating with the computer user
  - Video display terminals (VDTs), printers
- Machine readable
  - Suitable for communicating with equipment
  - Magnetic disk and tape systems, sensors and actuators (thiết bị khởi phát)
- Communication
  - Suitable for communicating with remote devices such as a terminal, a machine readable device, or another computer

External Device Block Diagram

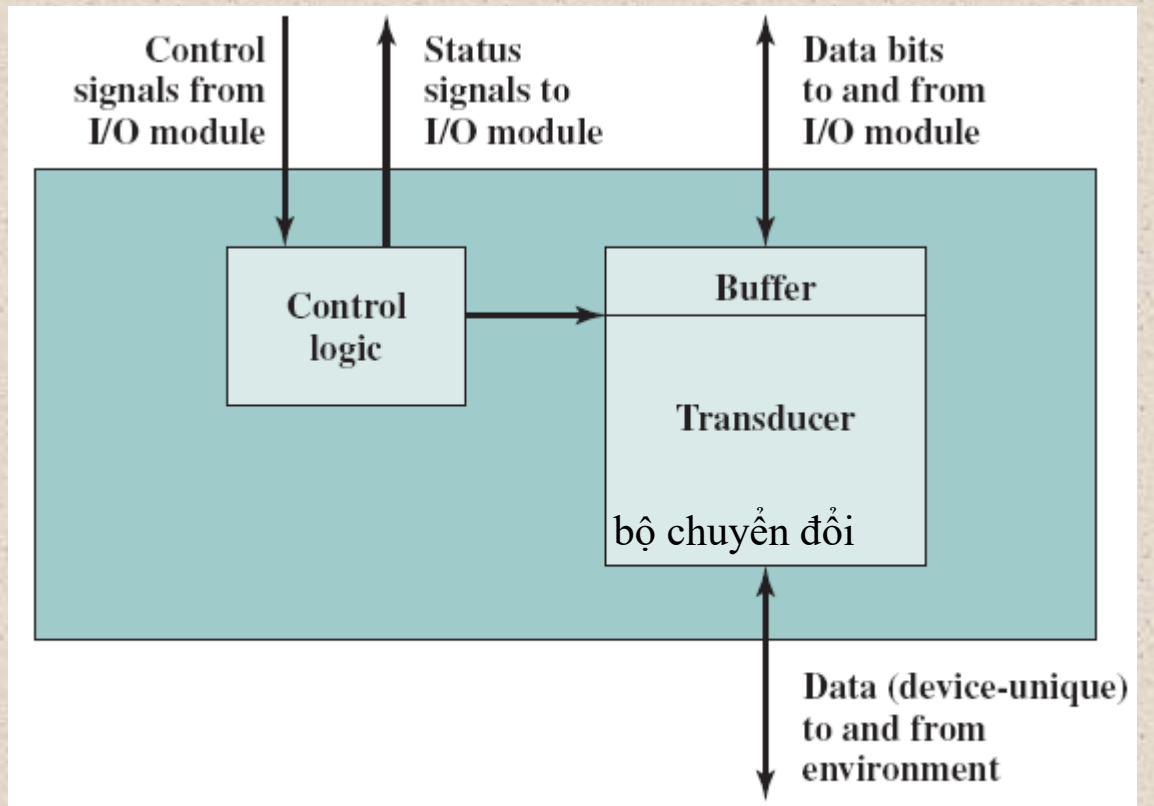


Figure 7.2 Block Diagram of an External Device

Keyboard/Monitor

Most common means of computer/user interaction

User provides input through the keyboard

The monitor displays data provided by the computer

International Reference Alphabet (IRA)

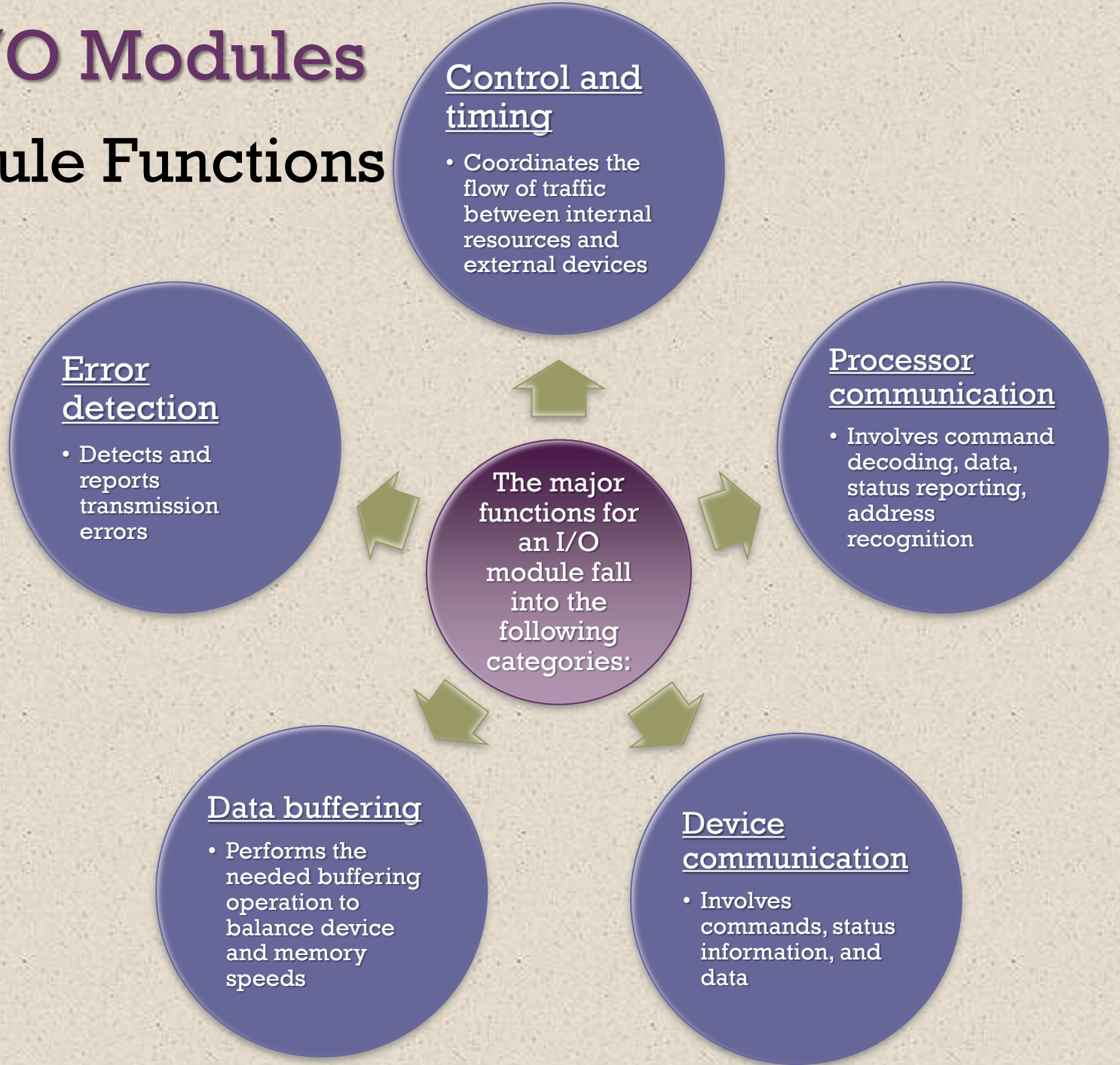
- **Basic unit of exchange is the character**
  - Associated with each character is a code
  - Each character in this code is represented by a unique 7-bit binary code
    - 128 different characters can be represented
- **Characters are of two types:**
  - **Printable**
    - Alphabetic, numeric, and special characters that can be printed on paper or displayed on a screen
  - **Control**
    - Have to do with controlling the printing or displaying of characters
    - Example is carriage return
    - Other control characters are concerned with communications procedures

Keyboard Codes

- When the user depresses a key it generates an electronic signal that is interpreted by the transducer in the keyboard and translated into the bit pattern of the corresponding IRA code
- This bit pattern is transmitted to the I/O module in the computer
- On output, IRA code characters are transmitted to an external device from the I/O module
- The transducer interprets the code and sends the required electronic signals to the output device either to display the indicated character or perform the requested control function

7.2-I/O Modules

Module Functions



I/O Module Structure

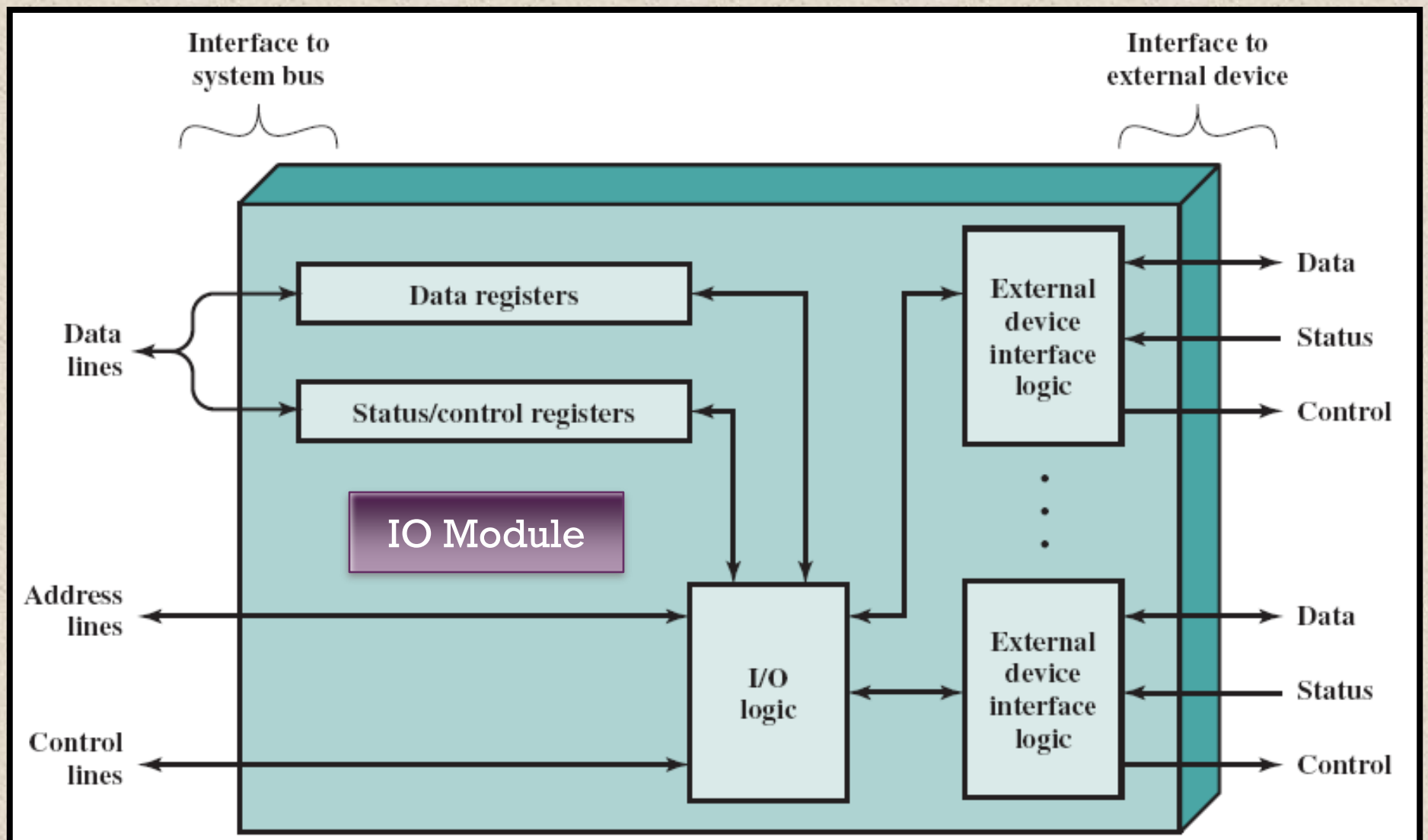


Figure 7.3 Block Diagram of an I/O Module

7.3- Programmed I/O

- **Three techniques** are possible for I/O operations:

Table 7.1 I/O Techniques

	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

- **Programmed I/O**
  - Data are exchanged between the processor and the I/O module
  - Processor executes a program that gives it direct control of the I/O operation
  - When the processor issues a command 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
- **Interrupt-driven I/O**
  - Processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work
- **Direct memory access (DMA)**
  - The I/O module and main memory exchange data directly without processor involvement

I/O Commands

- There are four types of I/O commands that an I/O module may receive when it is addressed by a processor:

1) **Control**

- used to activate a peripheral and tell it what to do

2) **Test**

- used to test various status conditions associated with an I/O module and its peripherals

3) **Read**

- causes the I/O module to obtain an item of data from the peripheral and place it in an internal buffer

4) **Write**

- causes the I/O module to take an item of data from the data bus and subsequently transmit that data item to the peripheral

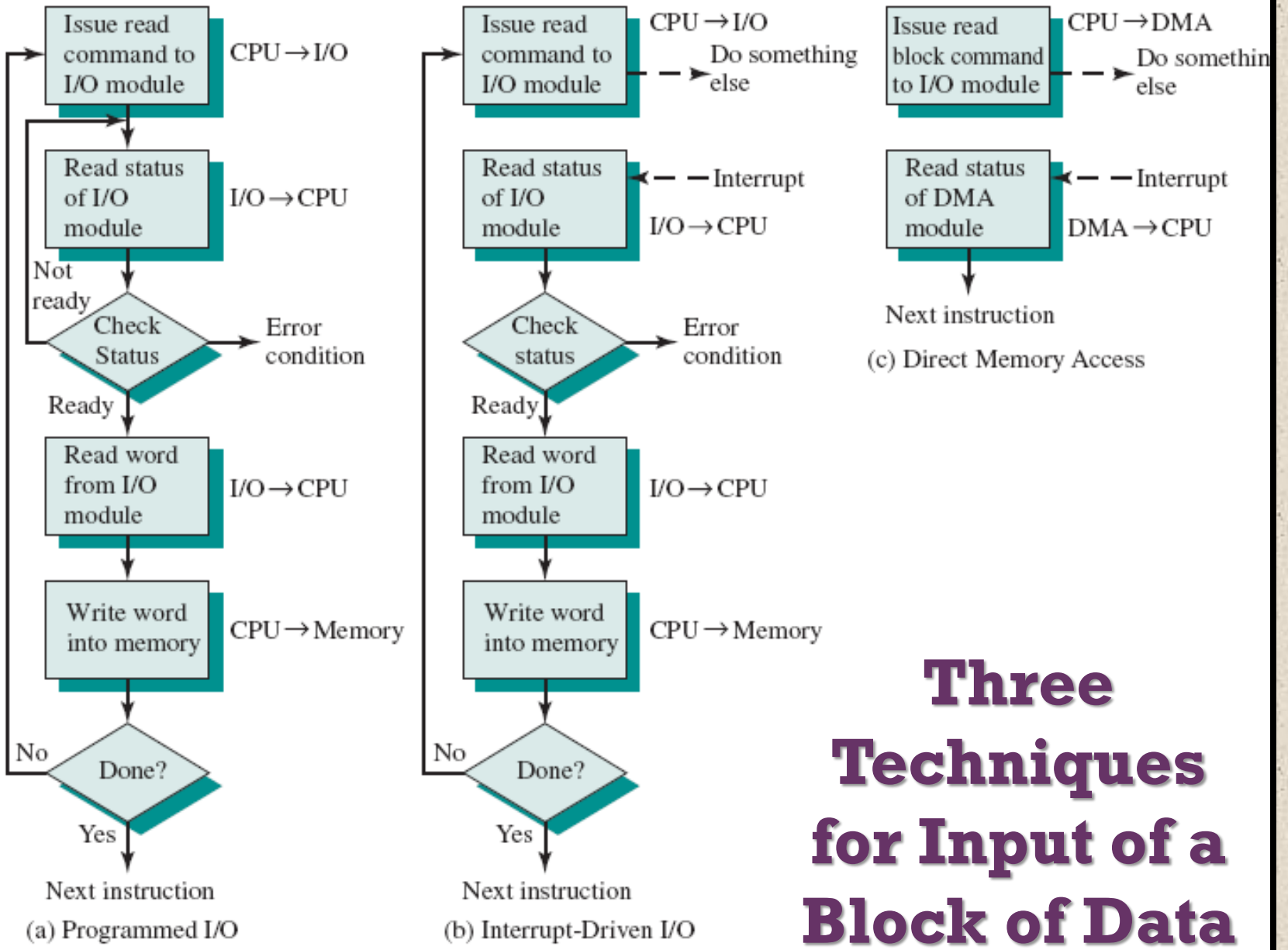


Figure 7.4 Three Techniques for Input of a Block of Data

Three Techniques for Input of a Block of Data

I/O Instructions

With programmed I/O there is a close correspondence between the I/O-related instructions that the processor fetches from memory and the I/O commands that the processor issues to an I/O module to execute the instructions

The form of the instruction depends on the way in which external devices are addressed

Each I/O device connected through I/O modules is given a unique identifier or address

When the processor issues an I/O command, the command contains the address of the desired device

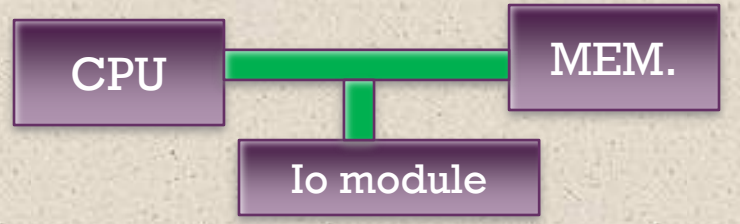
Thus each I/O module must interpret the address lines to determine if the command is for itself

Memory-mapped I/O

There is a single address space for memory locations and I/O devices

A single read line and a single write line are needed on the bus

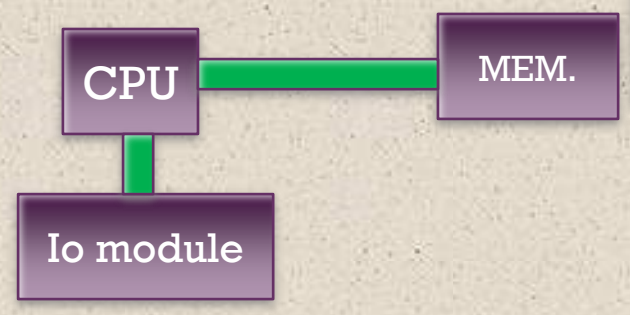
I/O Mapping Summary



- Memory mapped I/O
  - Devices and memory share an address space
  - I/O looks just like memory read/write
  - No special commands for I/O
    - Large selection of memory access commands available

Memory and IO devices share a common system bus

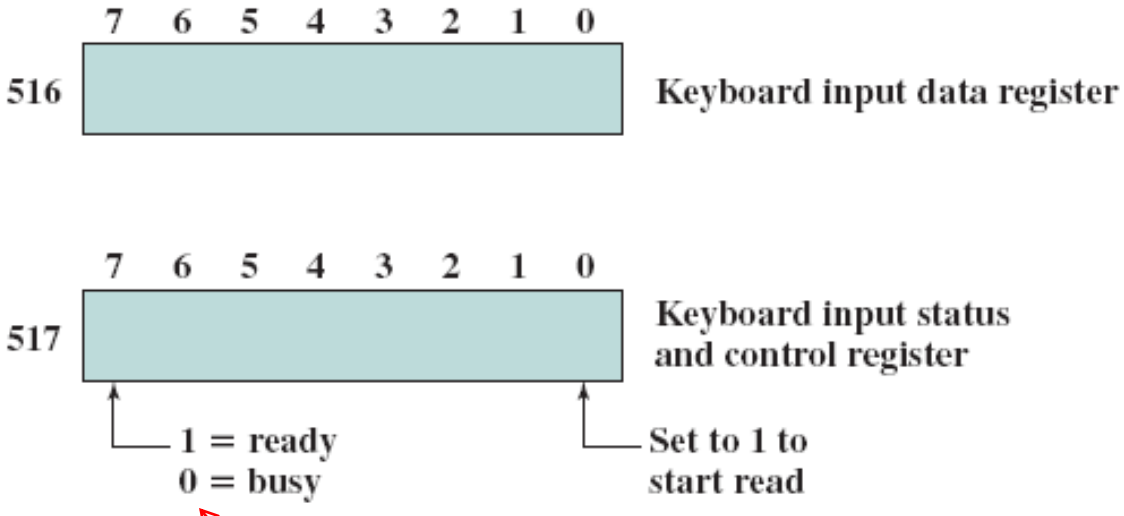
- Isolated I/O
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O
    - Limited set



2 different system buses for Memory and IO devices

Memory Mapped I/O

Isolated I/O Example



ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

Figure 7.5 Memory-Mapped and Isolated I/O

7.4- Interrupt-Driven I/O

The problem with **programmed I/O** is that the **processor has to wait** a long time for the I/O module to be ready for either reception or transmission of data

An alternative is for the processor to issue an I/O command to a module and then go on to do some other useful work

The I/O module will then **interrupt** the processor to request service **when it is ready** to exchange data with the processor

The processor executes the data transfer and resumes its former processing

Simple Interrupt Processing

PSW: Process Status Word

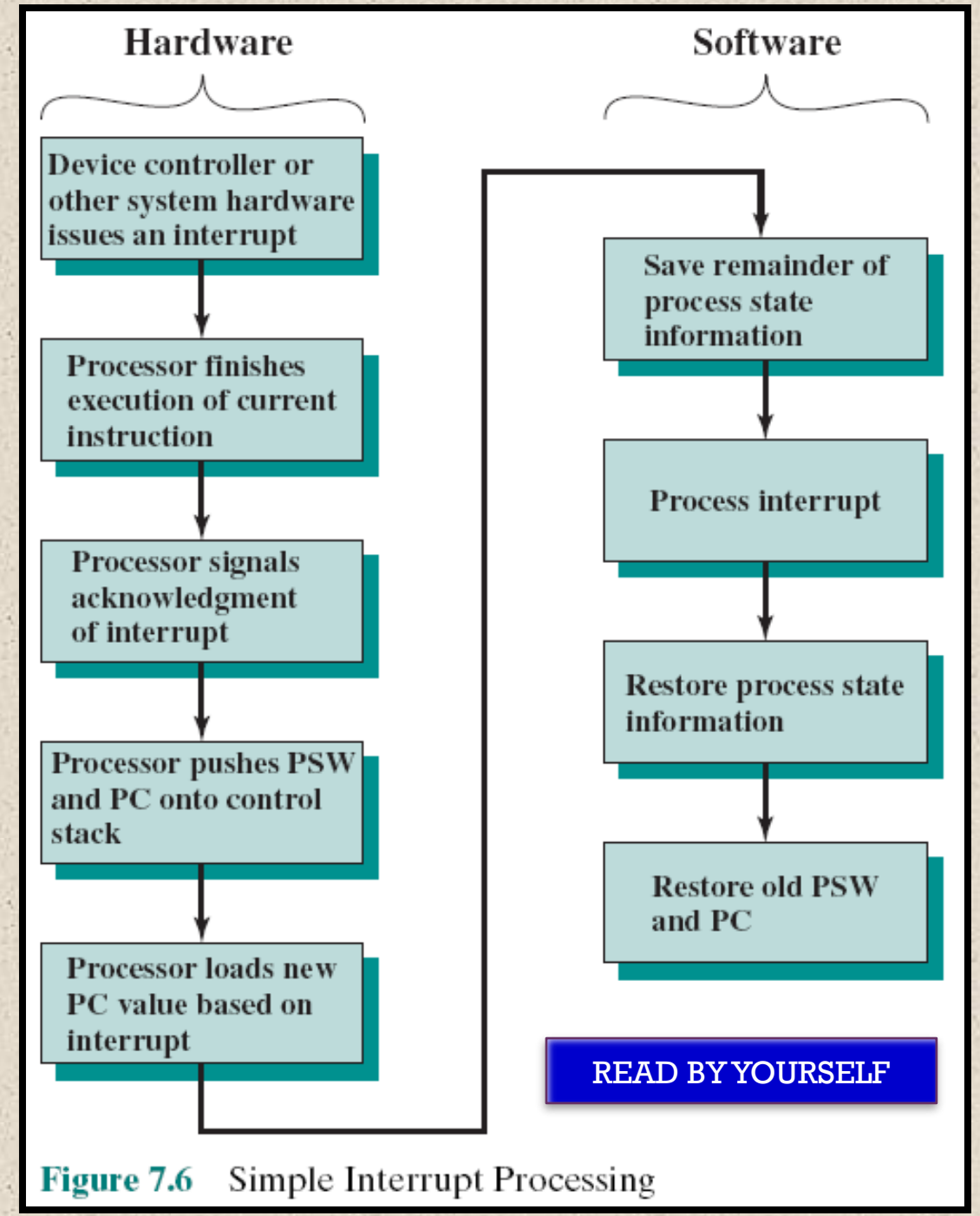


Figure 7.6 Simple Interrupt Processing

Changes in Memory and Registers for an Interrupt

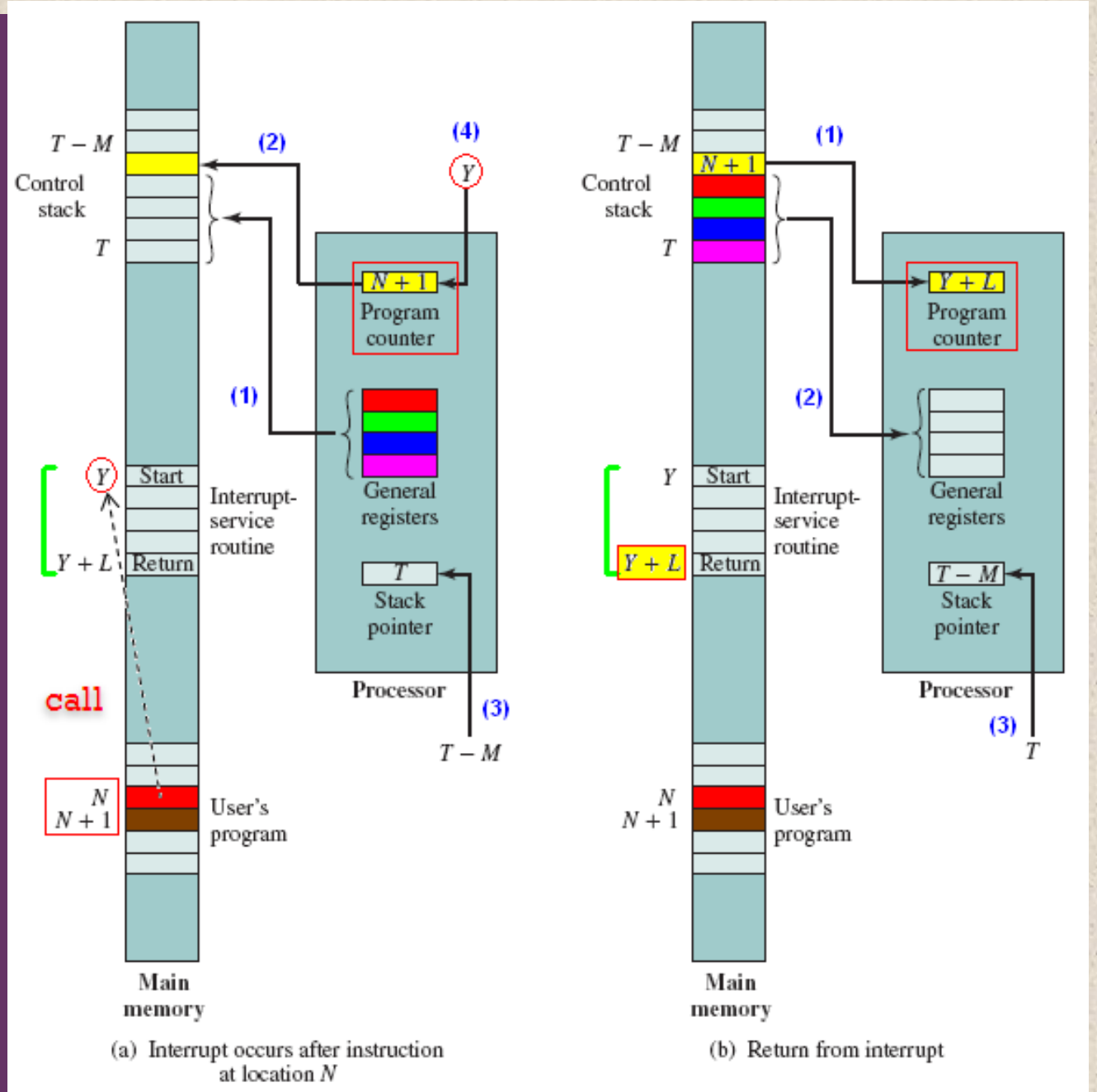


Figure 7.7 Changes in Memory and Registers for an Interrupt

Design Issues

Two design issues arise in implementing interrupt I/O:

- Because there will be multiple I/O modules how does the processor determine which device issued the interrupt?
- If multiple interrupts have occurred how does the processor decide which one to process?

Device Identification

Four general categories of techniques are in common use:

- **Multiple interrupt lines**
  - Between the processor and the I/O modules
  - Most straightforward approach to the problem
  - Consequently even if multiple lines are used, it is likely that each line will have multiple I/O modules attached to it
- **Software poll**
  - When processor detects an interrupt it branches to an interrupt-service routine whose job is to poll each I/O module to determine which module caused the interrupt
  - Time consuming
- **Daisy chain (hardware poll, vectored)**
  - The interrupt acknowledge line is daisy chained through the modules
  - Vector – address of the I/O module or some other unique identifier
  - Vectored interrupt – processor uses the vector as a pointer to the appropriate device-service routine, avoiding the need to execute a general interrupt-service routine first
- **Bus arbitration (vectored)**
  - An I/O module must first gain control of the bus before it can raise the interrupt request line
  - When the processor detects the interrupt it responds on the interrupt acknowledge line
  - Then the requesting module places its vector on the data lines

Intel 82C59A Interrupt Controller

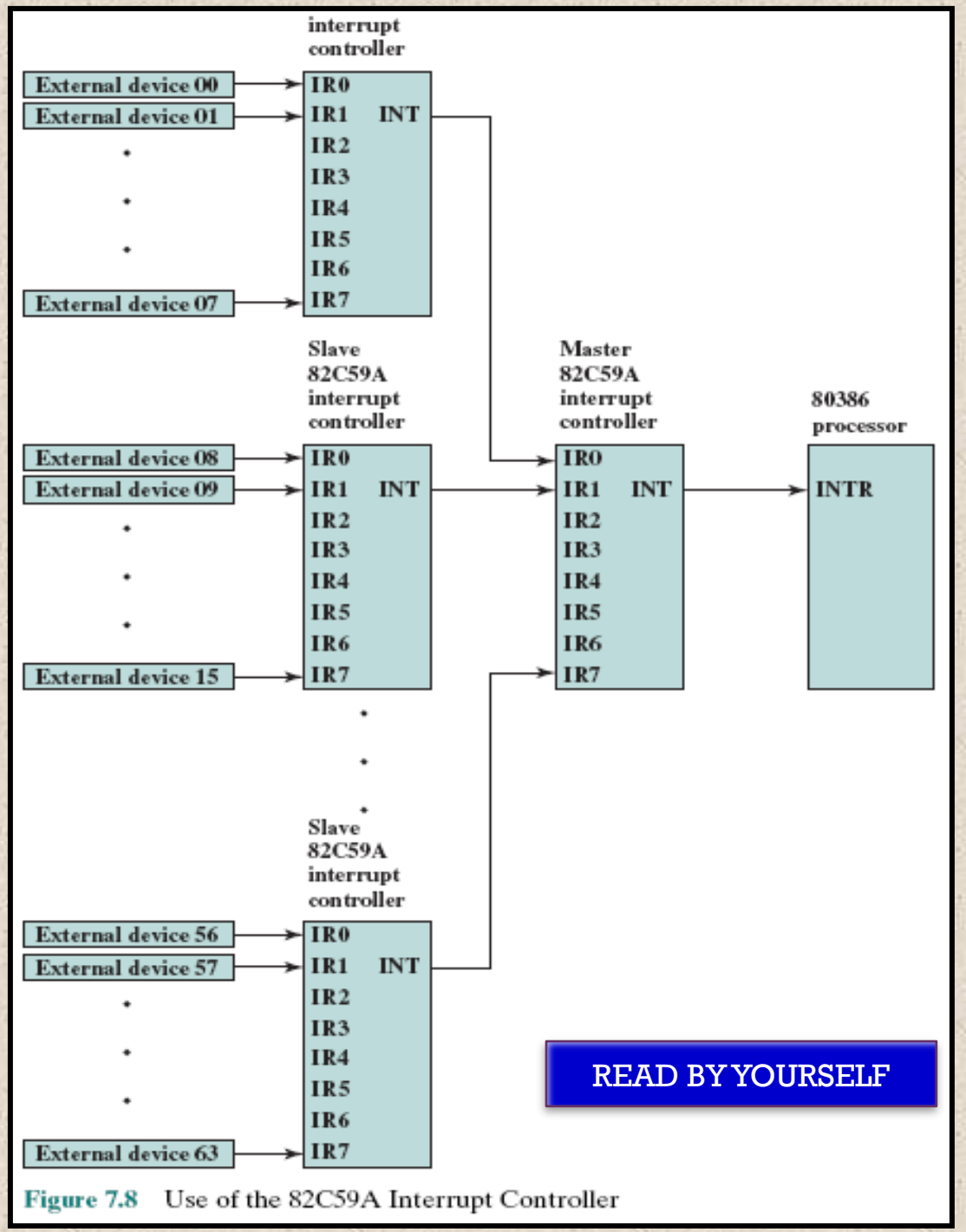
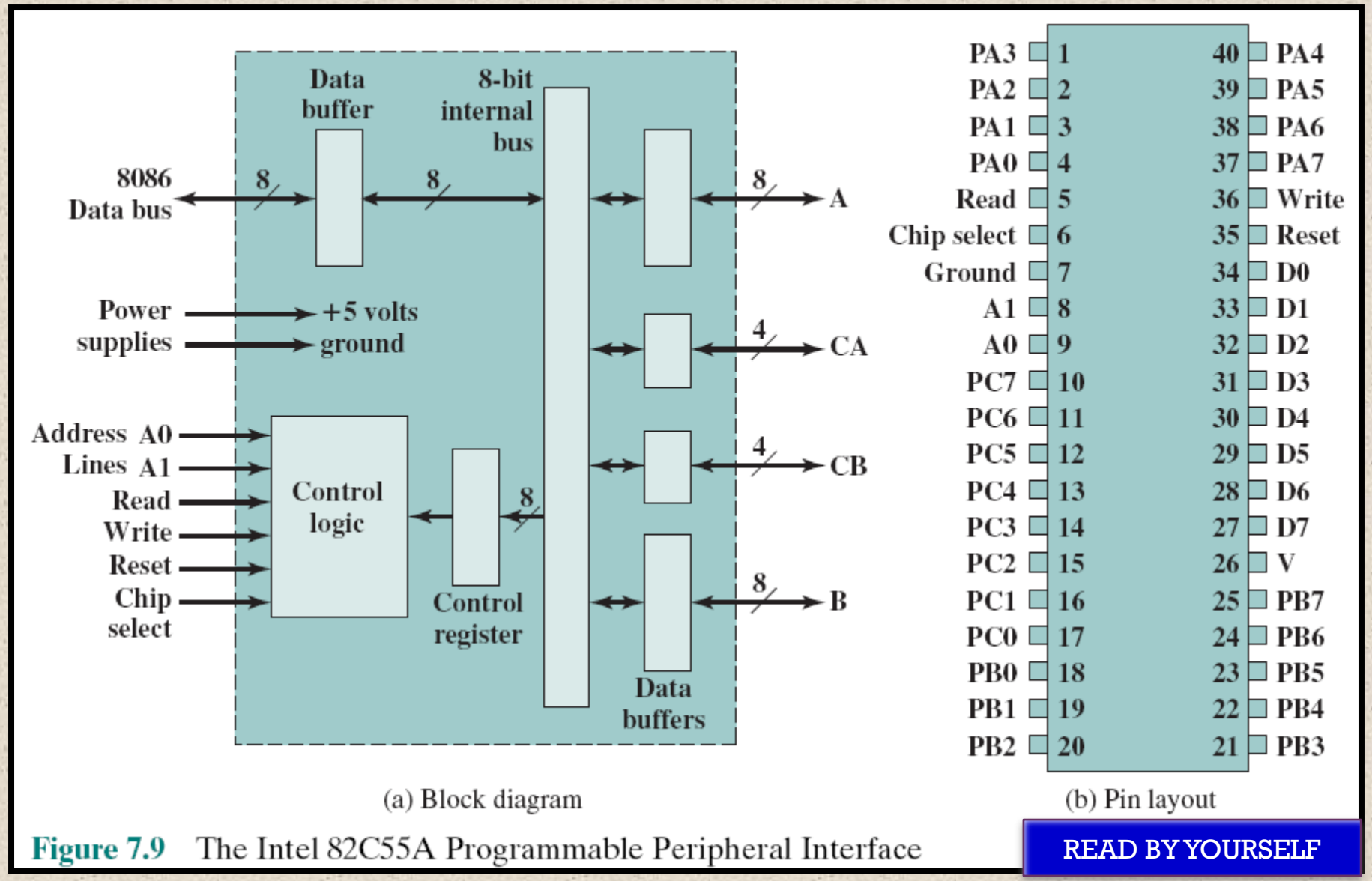


Figure 7.8 Use of the 82C59A Interrupt Controller

Intel 82C55A Programmable Peripheral Interface

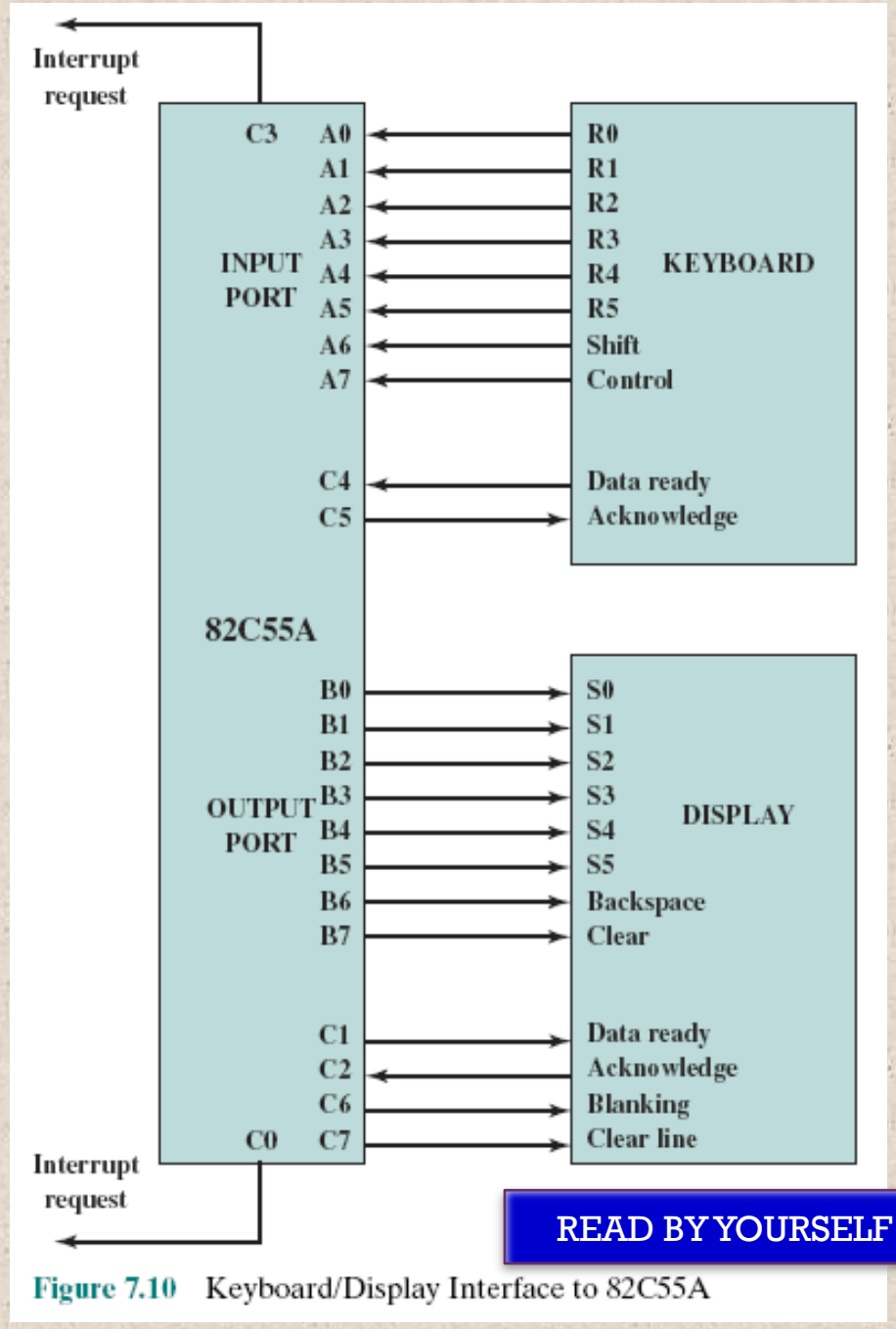


(a) Block diagram (b) Pin layout

Figure 7.9 The Intel 82C55A Programmable Peripheral Interface

READ BY YOURSELF

Keyboard/ Display Interfaces to 82C55A



READ BY YOURSELF

Figure 7.10 Keyboard/Display Interface to 82C55A

Drawbacks of Programmed and Interrupt-Driven I/O

- Both forms of I/O suffer from two inherent drawbacks:

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

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

- When large volumes of data are to be moved a more efficient technique is *direct memory access* (DMA)

7.5- Direct Memory Access

Typical DMA Module Diagram

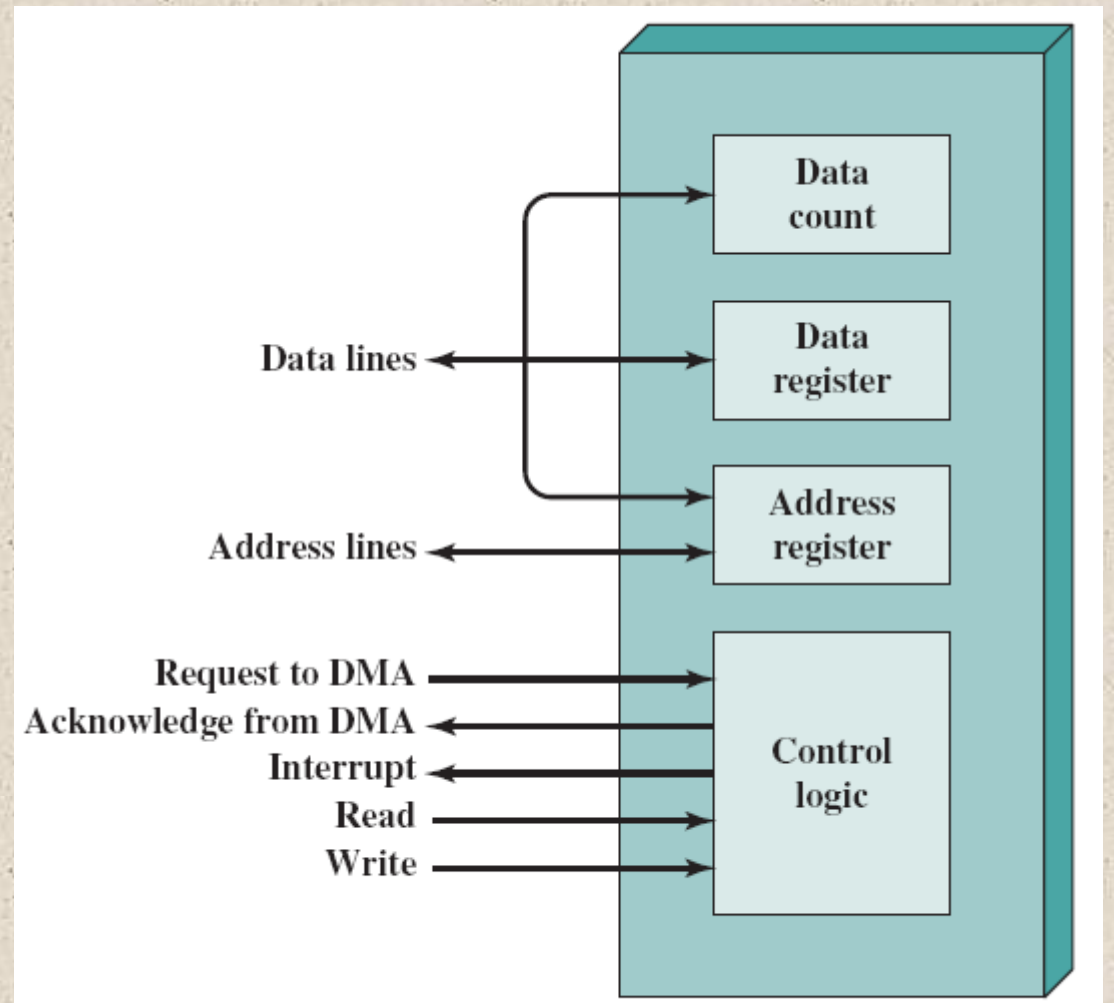


Figure 7.11 Typical DMA Block Diagram

DMA Operation

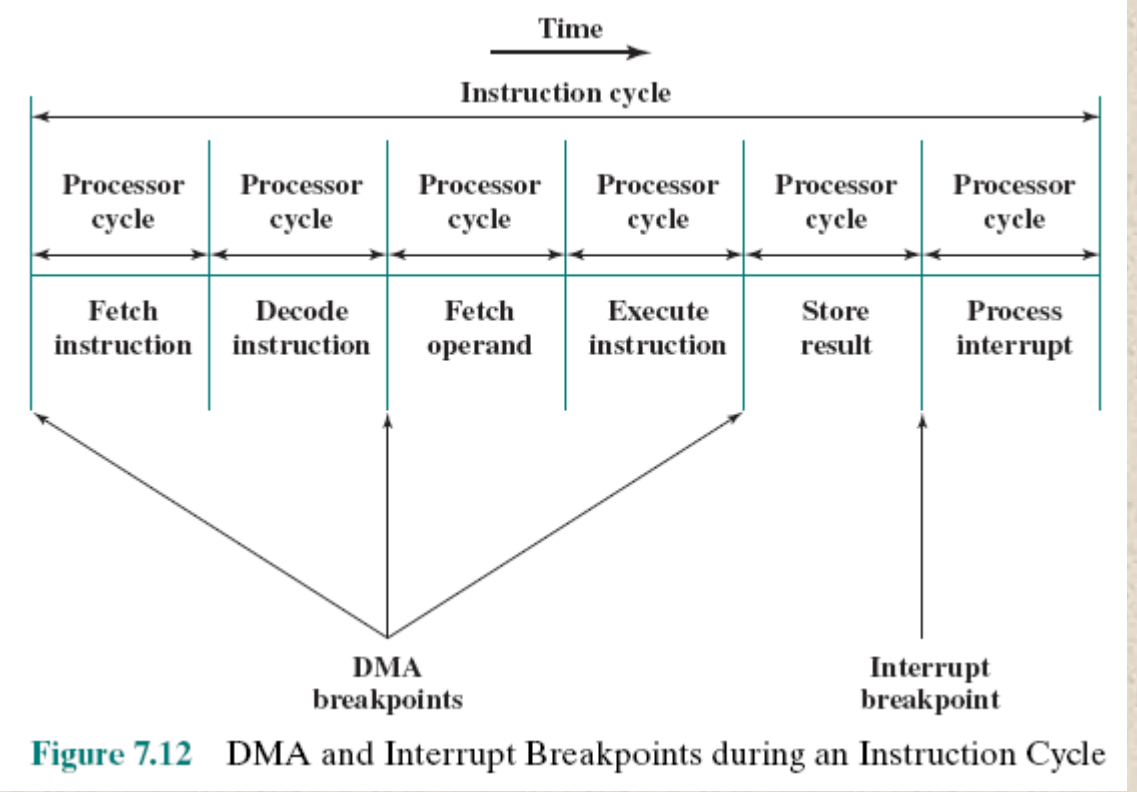


Figure 7.12 DMA and Interrupt Breakpoints during an Instruction Cycle

DMA

DMA

Figure 7.12 shows where in the instruction cycle the processor may be suspended. In each case, the processor is suspended just before it needs to use the bus. The DMA module then transfers one word and returns control to the processor. Note that this is not an interrupt; the processor does not save a context and do something else. Rather, the processor pauses for one bus cycle. The overall effect is to cause the processor to execute more slowly. Nevertheless, for a multiple-word I/O transfer, DMA is far more efficient than interrupt-driven or programmed I/O.

Alternative DMA Configurations

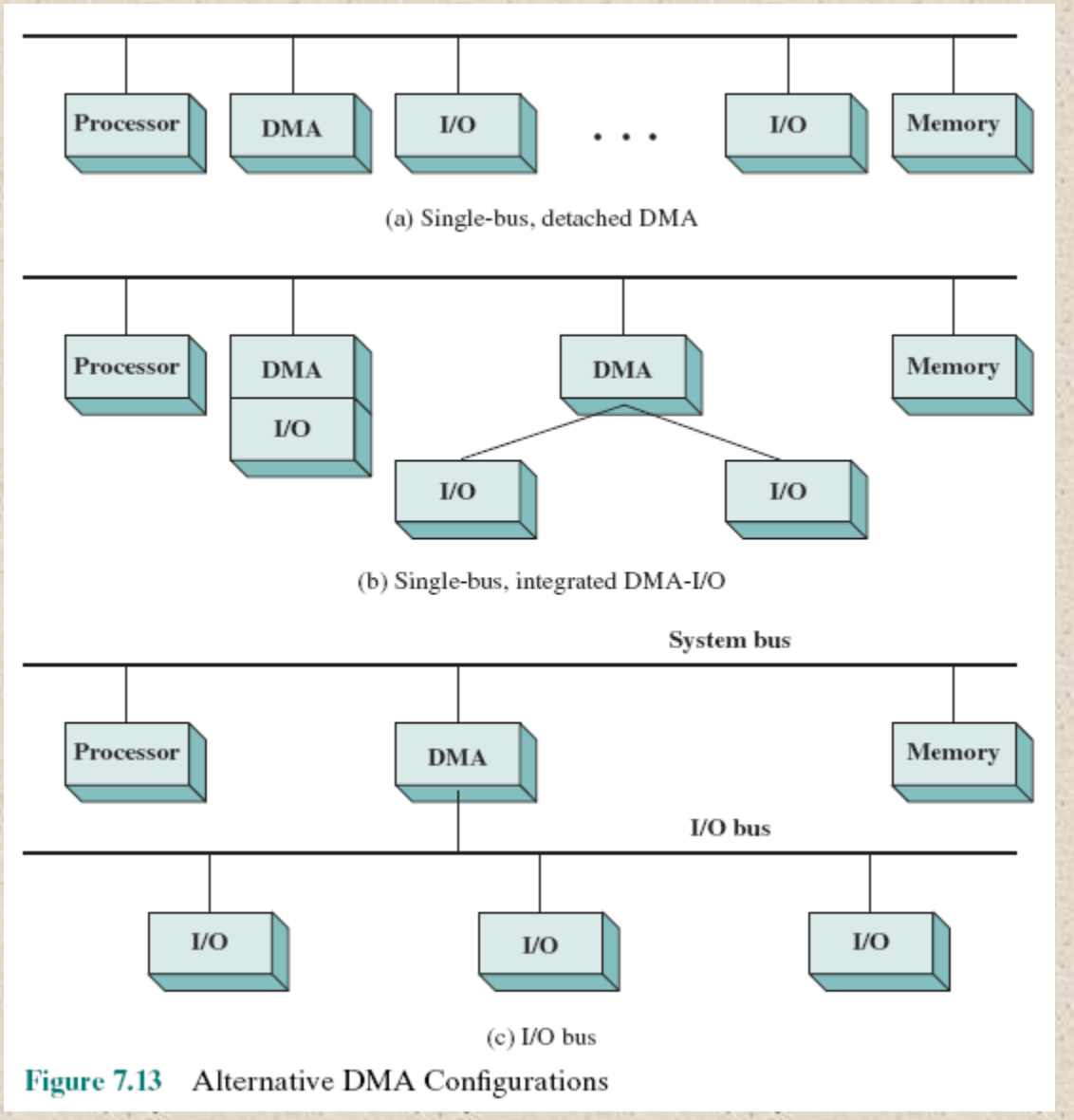
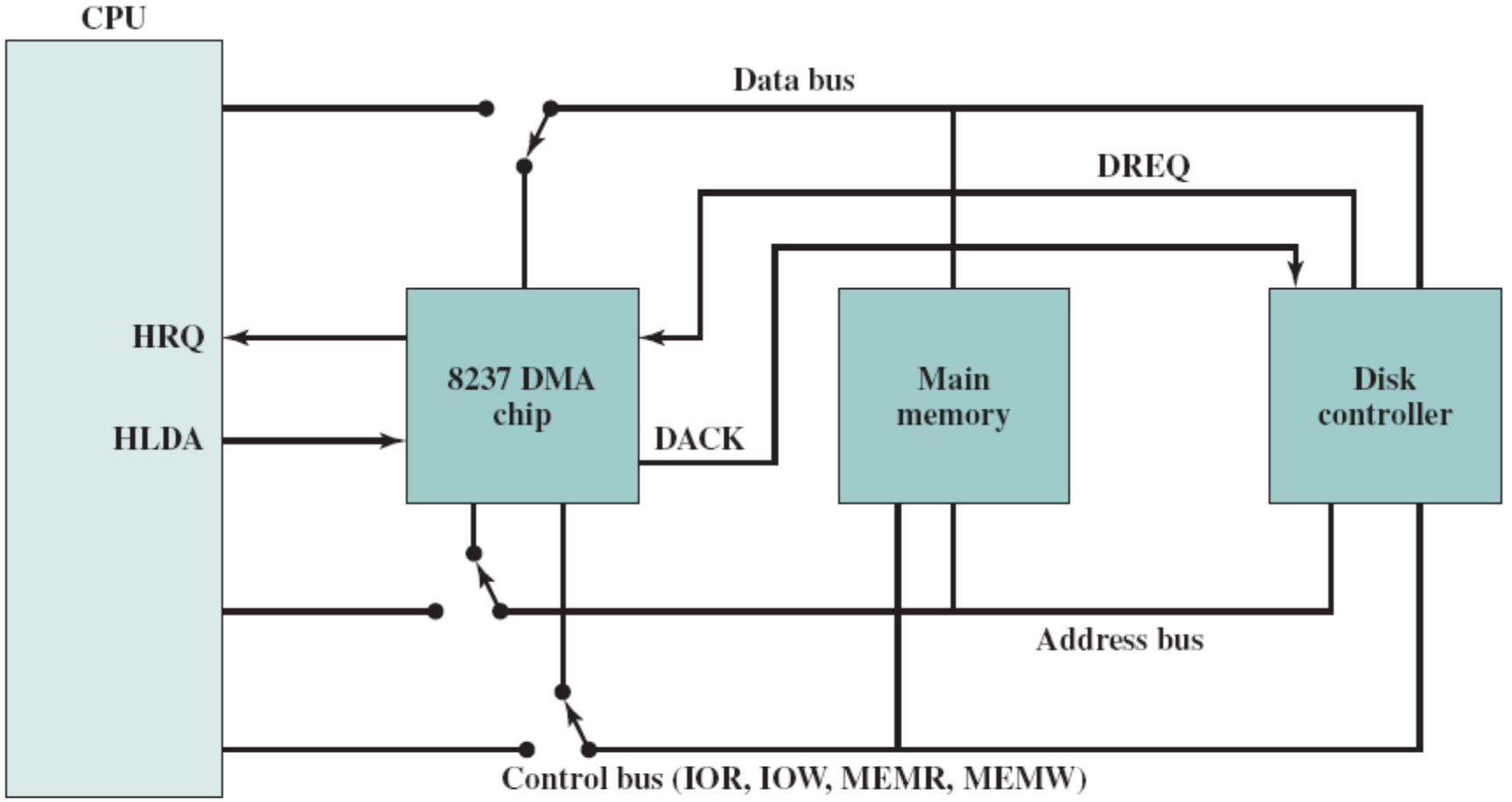


Figure 7.13 Alternative DMA Configurations

8237 DMA Usage of System Bus



DACK = DMA acknowledge
DREQ = DMA request
HLDA = HOLD acknowledge
HRQ = HOLD request

When DMA carries out, CPU is idle. True or false?

Figure 7.14 8237 DMA Usage of System Bus

Fly-By DMA Controller

Data does not pass through and is not stored in DMA chip

- DMA only between I/O port and memory
- **Not** between two I/O ports or two memory locations

Can do memory to memory via register

8237 contains four DMA channels

- Programmed independently
- Any one active
- Numbered 0, 1, 2, and 3

Table 7.2 – Intel 8237A Registers

Table 7.2 Intel 8237A Registers

Bit	Command	Status	Mode	Single Mask	All Mask
D0	Memory-to-memory E/D	Channel 0 has reached TC	Channel select	Select channel mask bit	Clear/set channel 0 mask bit
D1	Channel 0 address hold E/D	Channel 1 has reached TC			Clear/set channel 1 mask bit
D2	Controller E/D	Channel 2 has reached TC	Verify/write/ read transfer	Clear/set mask bit	Clear/set channel 2 mask bit
D3	Normal/compressed timing	Channel 3 has reached TC		Not used	Clear/set channel 3 mask bit
D4	Fixed/rotating priority	Channel 0 request	Auto-initialization E/D		Not used
D5	Late/extended write selection	Channel 0 request	Address increment/ decrement select		
D6	DREQ sense active high/low	Channel 0 request			
D7	DACK sense active high/low	Channel 0 request	Demand/single/block/ cascade mode select		

E/D = enable/disable
TC = terminal count

READ BY YOURSELF

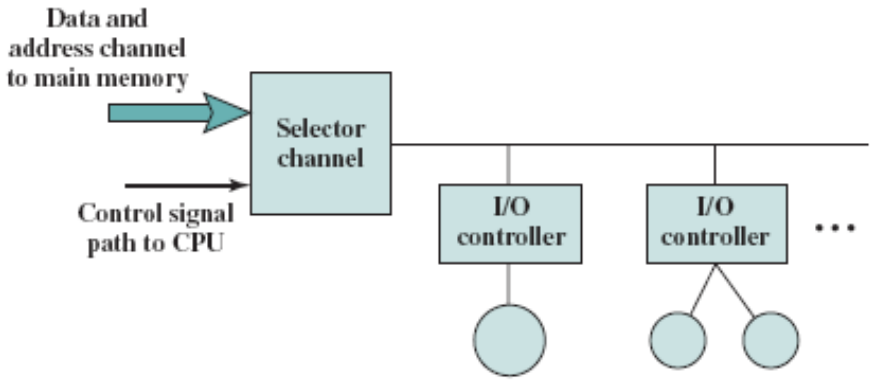
E/D = enable/disable
TC = terminal count

7.6- IO Channels and Processors

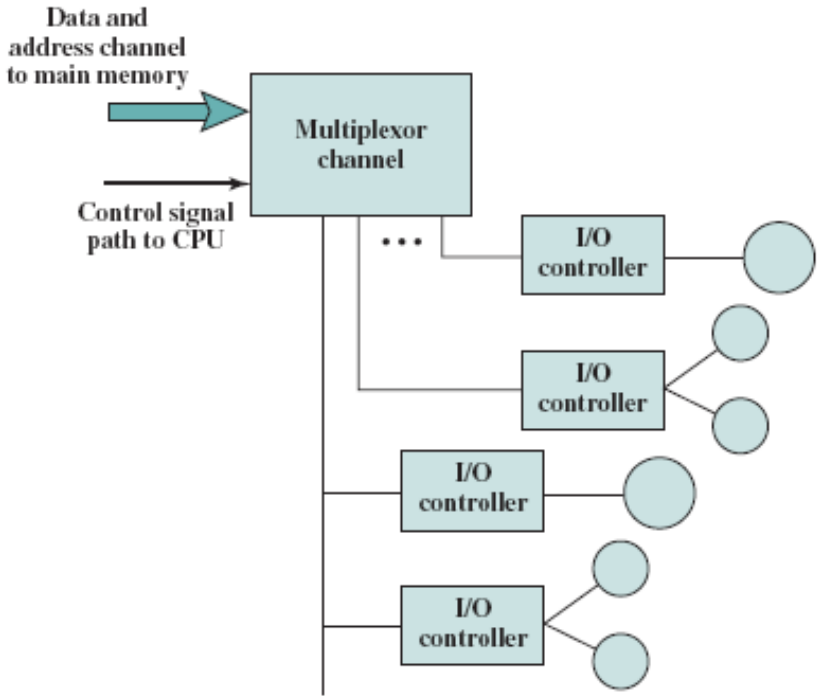
Evolution of the I/O Function

1. The CPU directly controls a peripheral device.
2. A controller or I/O module is added. The CPU uses programmed I/O without interrupts.
3. Same configuration as in step 2 is used, but now interrupts are employed. The CPU need not spend time waiting for an I/O operation to be performed, thus increasing efficiency.
4. The I/O module is given direct access to memory via DMA. It can now move a block of data to or from memory without involving the CPU, except at the beginning and end of the transfer.
5. The I/O module is enhanced to become a processor in its own right, with a specialized instruction set tailored for I/O
6. The I/O module has a local memory of its own and is, in fact, a computer in its own right. With this architecture a large set of I/O devices can be controlled with minimal CPU involvement.

I/O Channel Architecture



(a) Selector



(b) Multiplexor

Figure 7.15 I/O Channel Architecture

Exercises

- 7.1- List three broad classifications of external, or peripheral, devices.
- 7.2- What is the International Reference Alphabet?
- 7.3- What are the major functions of an I/O module?
- 7.4- List and briefly define three techniques for performing I/O.
- 7.5- What is the difference between memory-mapped I/O and isolated I/O?
- 7.6- When a device interrupt occurs, how does the processor determine which device issued the interrupt?
- 7.7- When a DMA module takes control of a bus, and while it retains control of the bus, what does the processor do?

Summary

Chapter 7

Input/Output

- External devices
  - Keyboard/monitor
  - Disk drive
- I/O modules
  - Module function
  - I/O module structure
- Programmed I/O
  - Overview of programmed I/O
  - I/O commands
  - I/O instructions
- Interrupt-driven I/O
  - Interrupt processing
  - Design issues
  - Intel 82C59A interrupt controller
  - Intel 82C55A programmable peripheral interface
- Direct memory access
  - Drawbacks of programmed and interrupt-driven I/O
  - DMA function
  - Intel 8237A DMA controller
- I/O channels and processors
  - The evolution of the I/O function
  - Characteristics of I/O channels
- The external interface
  - Types of interfaces
  - Point-to-point and multipoint configurations
  - Thunderbolt
  - InfiniBand