LAU
الجامعة اللبنانية الأميركية
Lebanese American University

Input/Output

Chapter 8

Based on:
William Stallings
Computer Organization and Architecture, 11th Global Edition

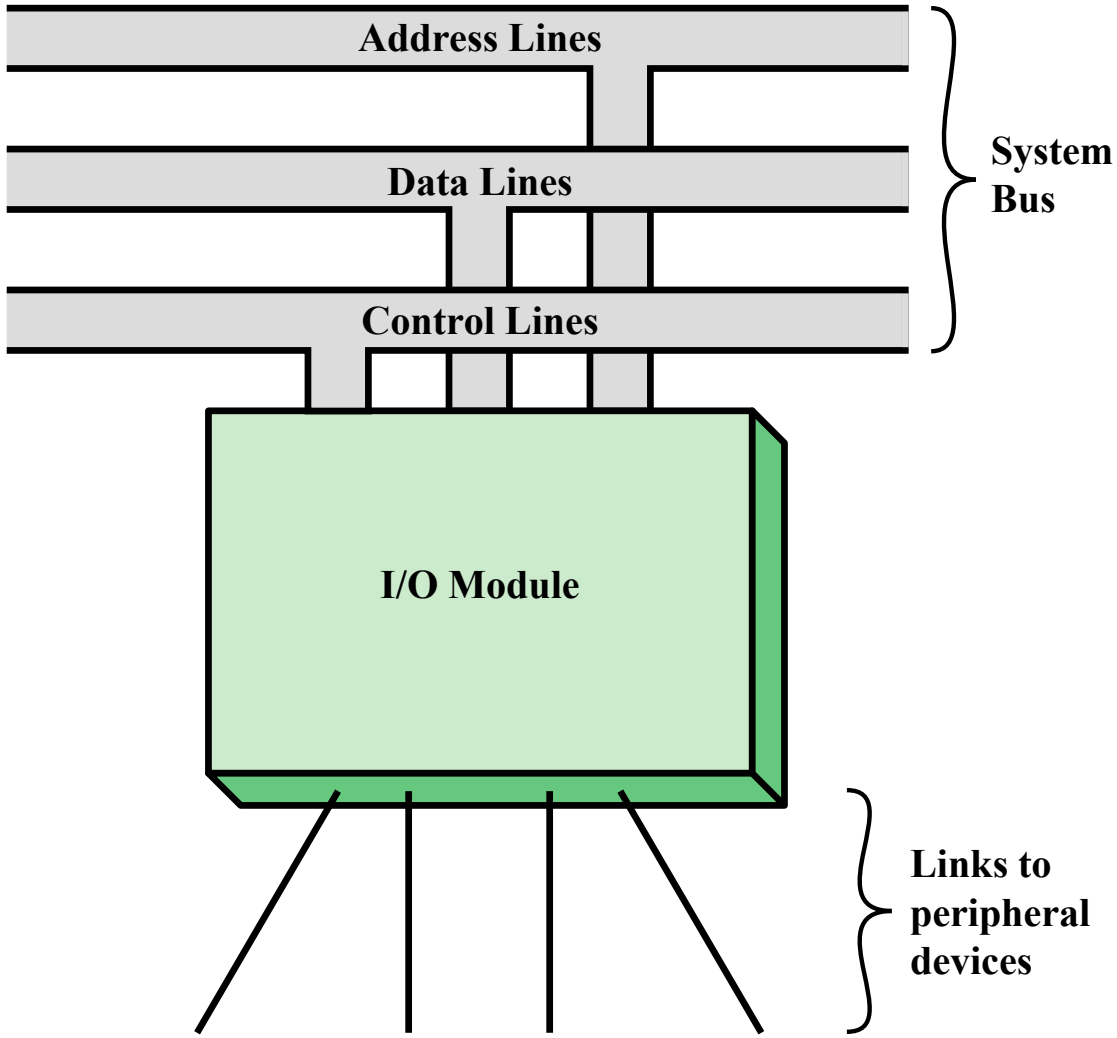
Why not Connect Peripherals Directly to System Bus?

- Wide variety of peripherals with various methods of operation
  - Impractical to incorporate the necessary logic within the processor to control a range of devices
- Data transfer rate of peripherals is often much slower than that of the memory or processor
  - Impractical to use the high-speed system bus to communicate directly with a peripheral
- Data transfer rate of some peripherals is faster than that of the memory or processor
  - 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

- An I/O module is required

Generic Model of an I/O Module

- 2 major functions:
  - Interface to the processor and memory via system bus or central switch
  - Interface to one or more peripheral devices by data links

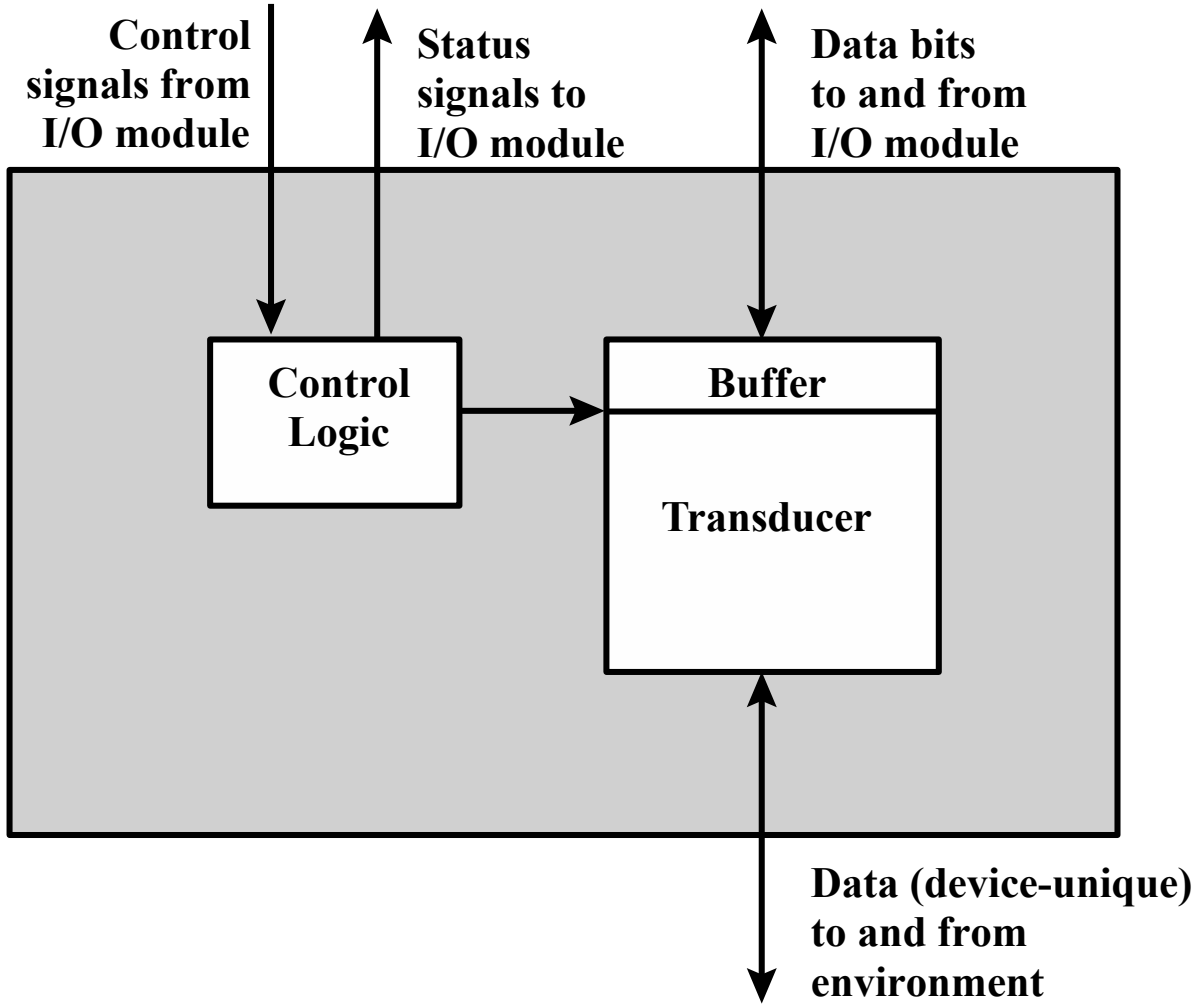


External Devices

- 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
- Categories
  - Human readable
    - Suitable for communicating with the computer user
    - Video display terminals, printers
  - Machine readable
    - Suitable for communicating with equipment
    - Magnetic disk and tape systems, sensors
  - Communication
    - Suitable for communicating with remote devices such as a terminal, a machine readable device, or another computer

Block Diagram of an External Device

- Control signals determine the function that the device will perform
- Data: set of bits to be sent to or received from the I/O module
- Status signals indicate the state of the device



Categories of the Major Functions for an I/O Module

- Control and timing
  - Coordinates the flow of traffic between internal resources and external devices
- Processor communication
  - Involves command decoding, data, status reporting, address recognition
- Device communication
  - Involves commands, status information, and data
- Data buffering
  - Performs the needed buffering operation to balance device and memory speeds
- Error detection
  - Detects and reports transmission errors

Control and Timing

- Processor may communicate with one or more external devices in unpredictable patterns, depending on the program's need for I/O
- Internal resources, such as main memory and the system bus, must be shared among a number of activities, including data I/O
- E.g.: control of transfer of data from external device to processor might involve the following sequence of steps:
  - Processor interrogates I/O module to check status of the attached device
  - I/O module returns the device status
  - If the device is operational and ready to transmit, processor requests the transfer of data, by means of a command to the I/O module
  - I/O module obtains a unit of data from external device
  - Data are transferred from the I/O module to the processor
- If the system employs a bus, then each of the interactions between the processor and the I/O module involves one or more bus arbitrations

Processor Communication

- I/O module must communicate with the processor and with the external device
- Command decoding:
  - I/O module accepts commands from the processor, typically sent as signals on the control bus.
  - E.g. an I/O module for a disk drive might accept the following commands: READ SECTOR, WRITE SECTOR, …
- Data are exchanged between the processor and the I/O module over the data bus
- Status reporting**:**
  - Because peripherals are so slow, it is important to know the status of the I/O module
  - E.g. if an I/O module is asked to send data to the processor (read), it may not be ready to do so because it is still working on the previous I/O command. This fact can be reported with a status signal
  - Common status signals are BUSY and READY
- Address recognition:
  - Just as each word of memory has an address, so does each I/O device
  - I/O module must recognize one unique address for each peripheral it controls.

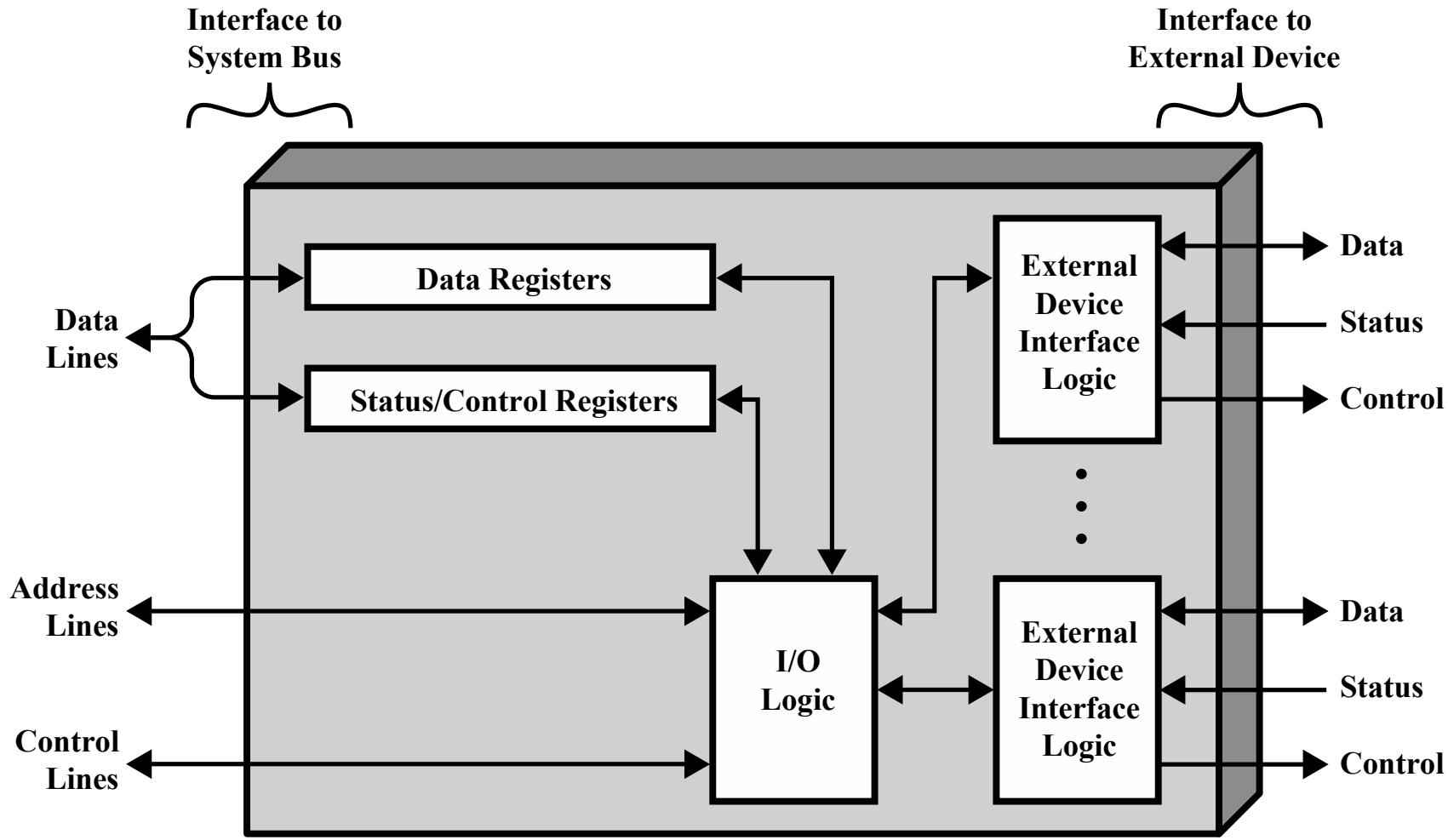
Data Buffering

- Transfer rate into and out of main memory of the processor is quite high
- Rate is orders of magnitude lower for many peripheral devices
- Data coming from main memory are sent to an I/O module in a rapid burst
  - Data are buffered in the I/O module and then sent to the peripheral device at its data rate
  - In the opposite direction, data are buffered so as not to tie up the memory in a slow transfer operation.
  - The I/O module must be able to operate at both device and memory speeds.
  - Similarly, if the I/O device operates at a rate higher than the memory access rate, then the I/O module performs the needed buffering operation.

Error Detection

- Responsible for error detection and for reporting errors to the processor
- One class of errors includes mechanical and electrical malfunctions reported by the device (e.g., paper jam, bad disk track)
- Another class consists of unintentional changes to the bit pattern as it is transmitted from device to I/O module
- Some form of error-detecting code is often used to detect transmission errors
  - Simple example is the use of a parity bit on each character of data
  - E.g. the IRA character code occupies 7 bits of a byte. The 8^{th} bit is set so that the total number of 1s in the byte is even (even parity) or odd (odd parity). When a byte is received, the I/O module checks the parity to determine whether an error has occurred.

Block Diagram of an I/O Module



3 Techniques for I/O Operations

- **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
    - Sensing device status, sending a read or a write command, transferring the data
  - 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 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

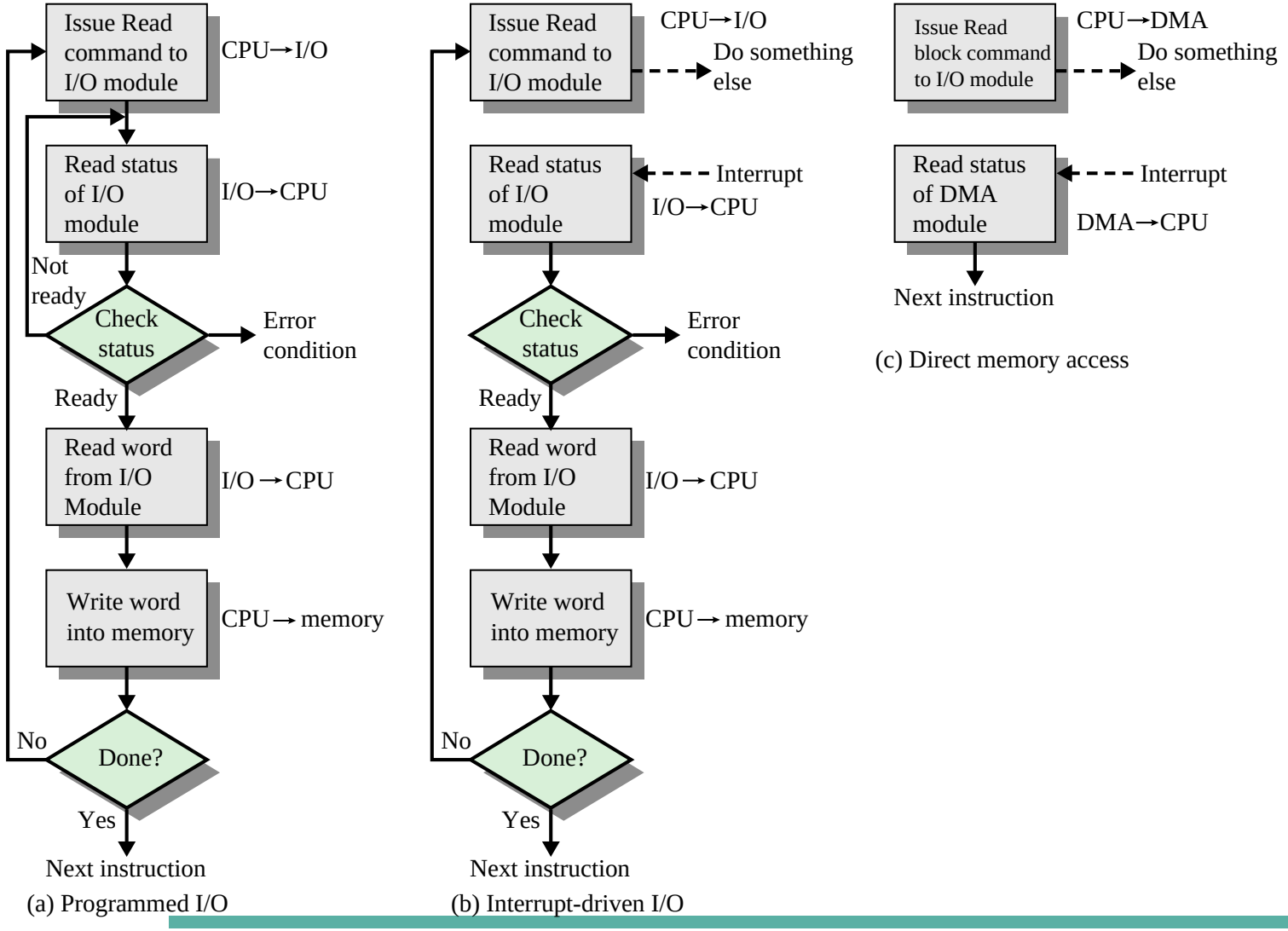
- When the processor is executing a program and encounters an instruction relating to I/O, it executes that instruction by issuing a command to the appropriate I/O module
- The I/O module will perform the requested action and then set the appropriate bits in the I/O status register
- The I/O module takes no further action to alert the processor
  - It does not interrupt the processor
- It is the responsibility of the processor to periodically check the status of the I/O module until it finds that the operation is complete
- View programmed I/O technique from the point of view of
  - I/O commands issued by the processor to the I/O module
  - I/O instructions executed by the processor

I/O Commands

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

  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

Techniques for Input of a Block of Data



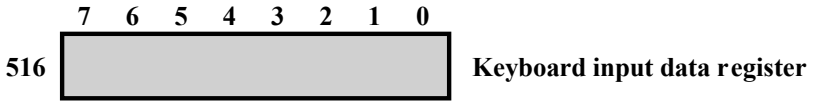
I/O Instructions

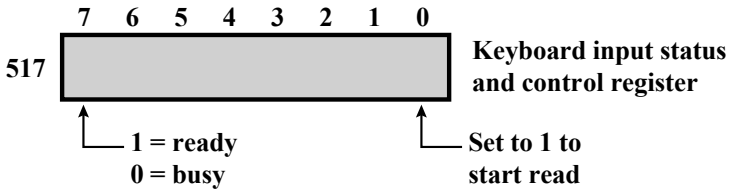
- With programmed I/O there is a close correspondence between the I/O-related instructions that the processor fetches from memory and the I/O commands that the processor issues to an I/O module to execute the instructions
- The form of the instruction depends on the way in which external devices are addressed
- Many I/O devices connected through I/O modules to the system
  - Each device is given a unique identifier or address
  - When the processor issues an I/O command, the command contains the address of the desired device
  - Each I/O module must interpret the address lines to determine if the command is for itself

I/O Mapping

- When processor, main memory, and I/O share a common bus, two modes of addressing are possible:
- **Memory mapped I/O**
  - I/O devices and memory locations share a single address space
  - I/O looks just like memory read/write
  - No special commands for I/O
    - Processor treats status and data registers of I/O modules as memory locations and uses the same machine instructions to access both memory and I/O devices
    - Large selection of memory access commands available
    - E.g. with 10 address lines, a combined total of 2^{10} = 1024 memory locations and I/O addresses can be supported, in any combination
- **Isolated I/O**
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O
    - Limited set
    - E.g. with 10 address lines, the system may now support both 1024 memory locations and 1024 I/O addresses

I/O Mapping





Assumptions:

- 10-bit address, with
- a 512- bit memory (locations 0–511)
- up to 512 I/O addresses (locations 512–1023)

Two addresses are dedicated to keyboard input from a particular terminal

- Address 516 refers to data register
- Address 517 refers to status register

The program shown will read 1 byte of data from the keyboard into an accumulator. The processor loops until the data byte is available.

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

With isolated I/O, the I/O ports are accessible only by special I/O commands, which activate the I/O command lines on the bus.

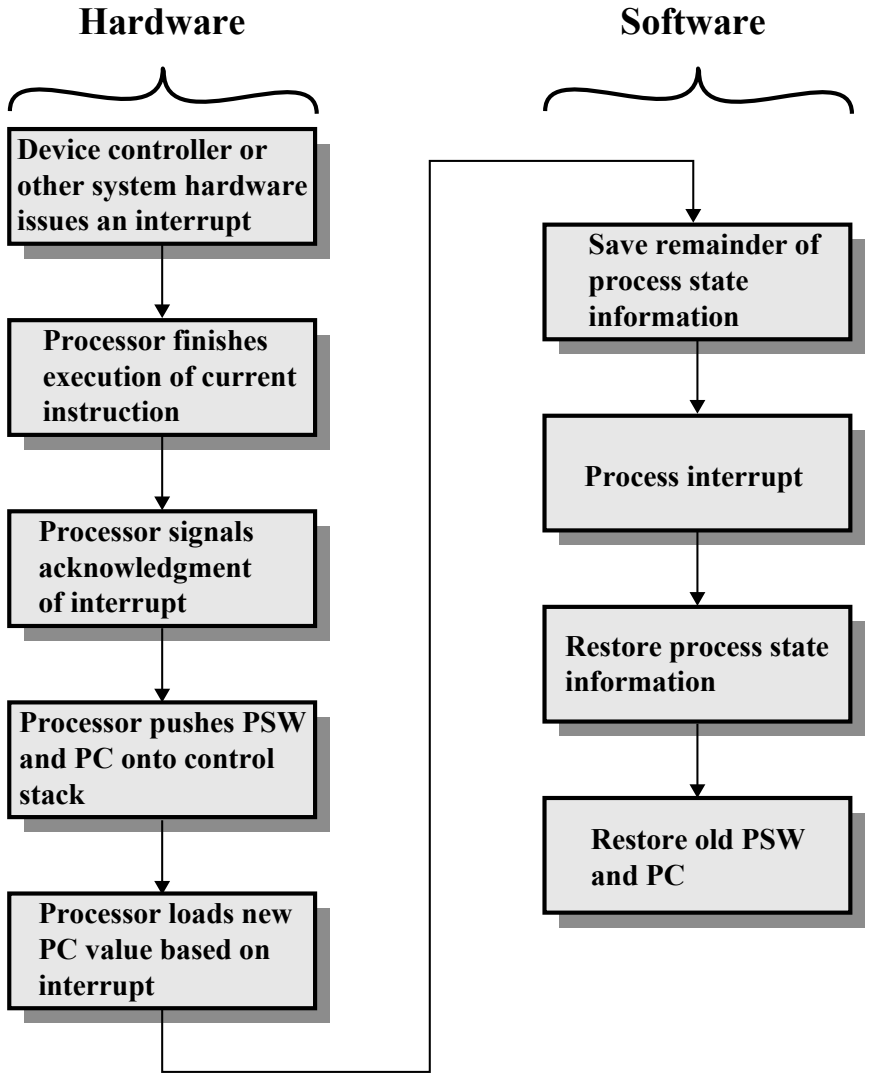
ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

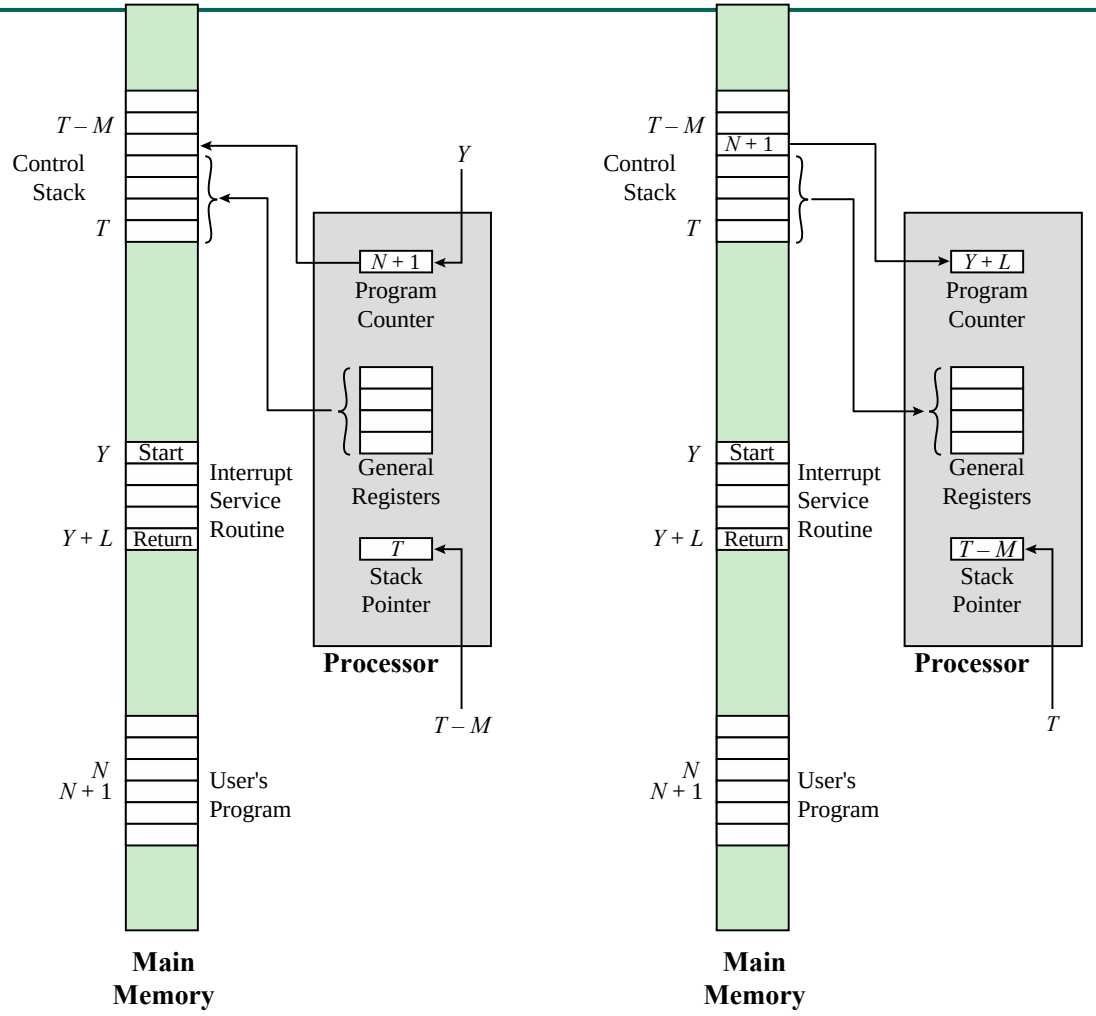
Interrupt-Driven I/O

- 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



Changes in Memory and Registers for an Interrupt



(a) Interrupt occurs after instruction at location N

(b) Return from interrupt

Design Issues

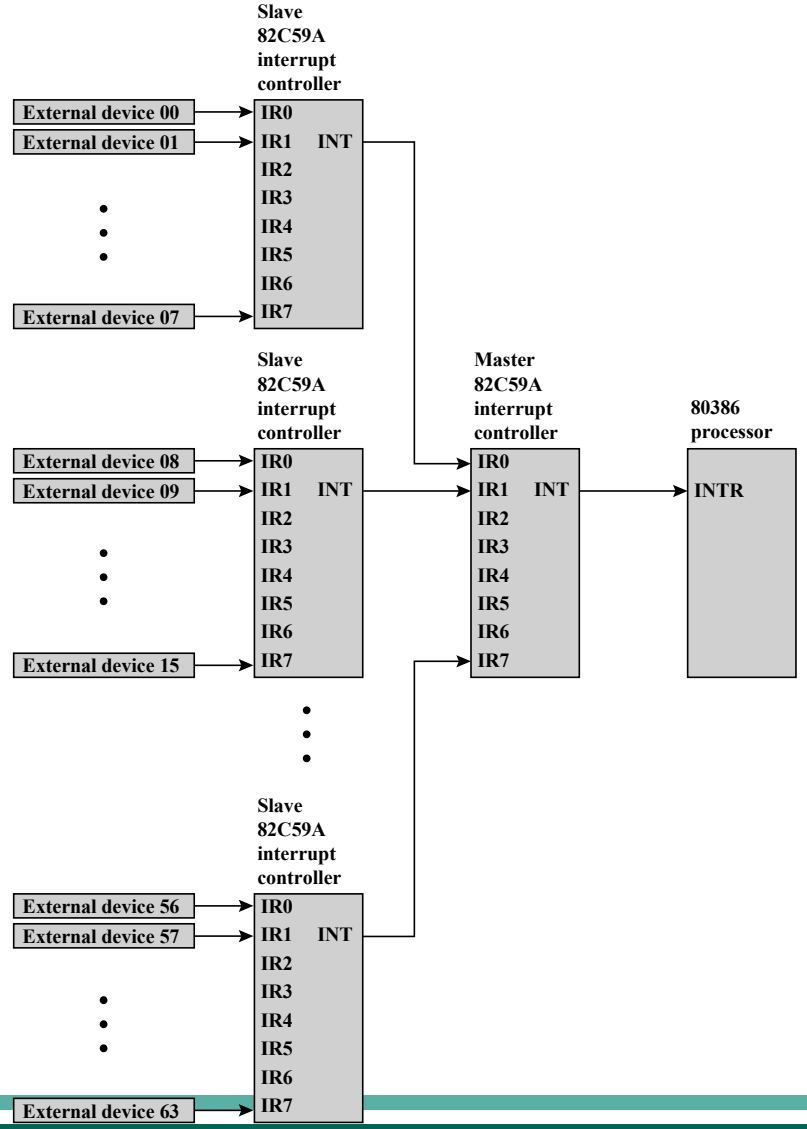
- Two design issues arise in implementing interrupt I/O
  - Because there will be multiple I/O modules, how does the processor determine which device issued the interrupt?
  - If multiple interrupts have occurred, how does the processor decide which one to process?

Device Identification

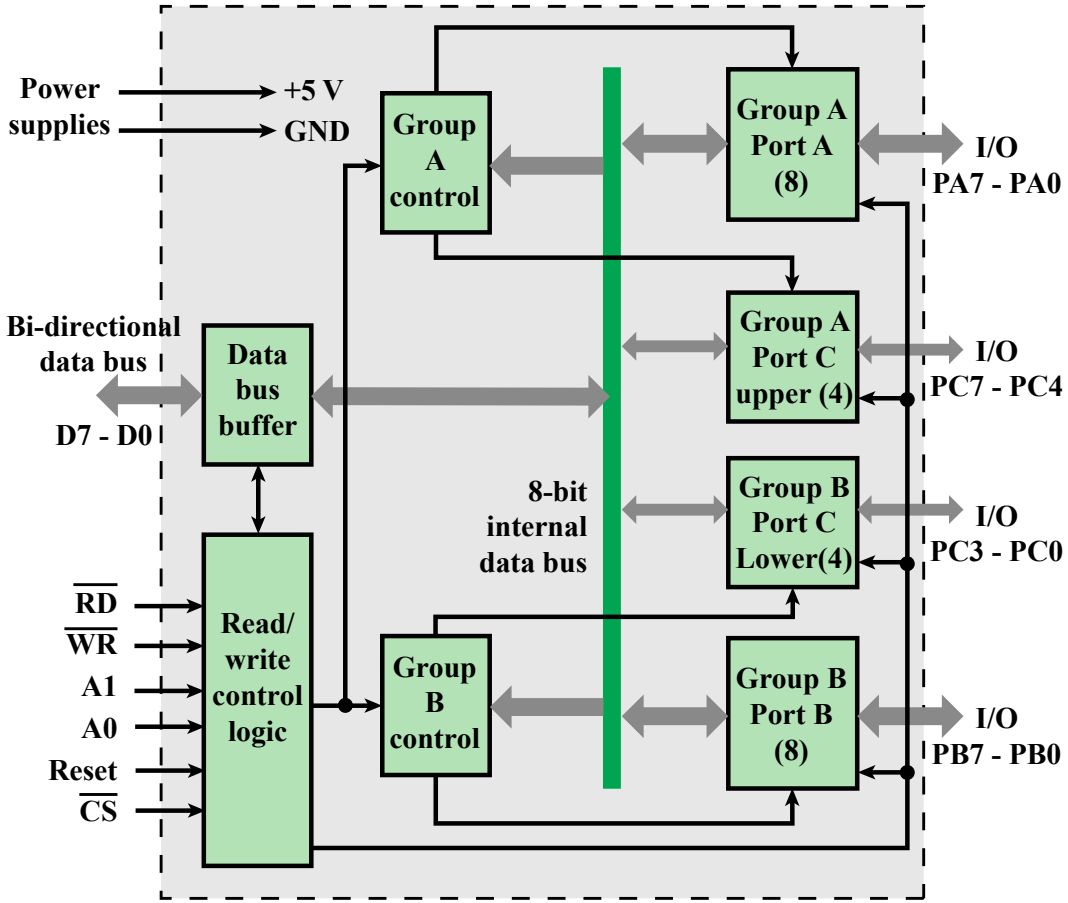
Four general categories of techniques are in common use:

- **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

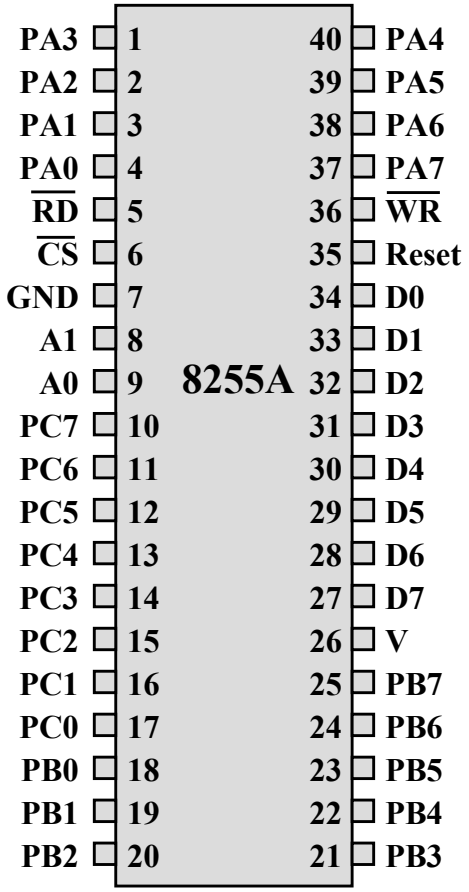
Use of the 82C59A Interrupt Controller



The Intel 8255A Programmable Peripheral Interface

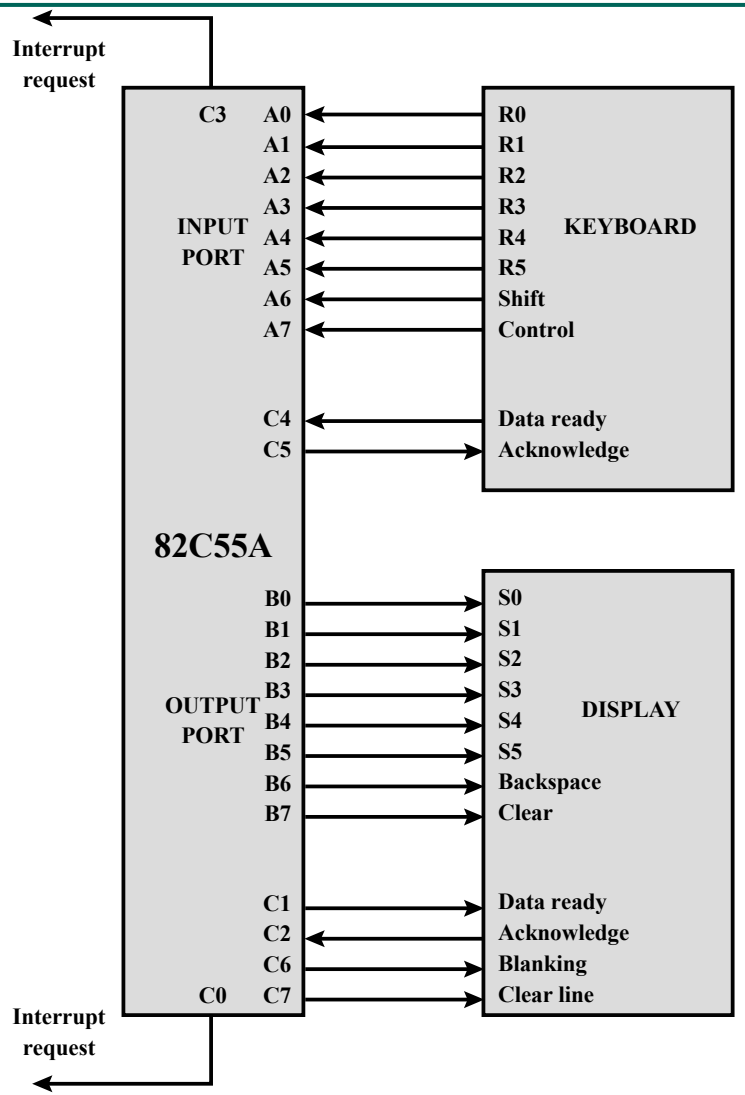


(a) Block diagram



(b) Pin layout

Keyboard/Display Interface to 8255A



Drawbacks of Programmed and Interrupt-Driven I/O

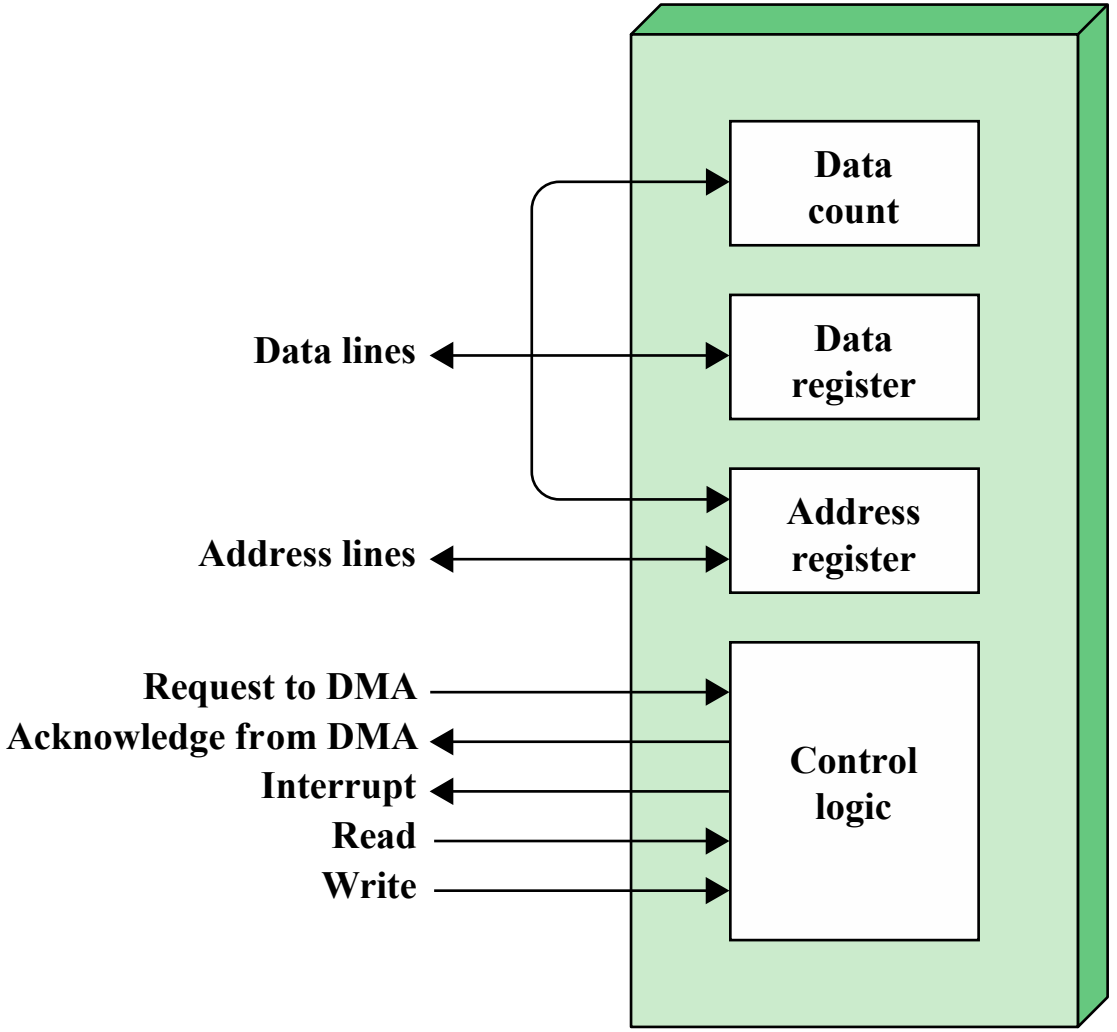
- Both forms of I/O suffer from two inherent drawbacks:

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

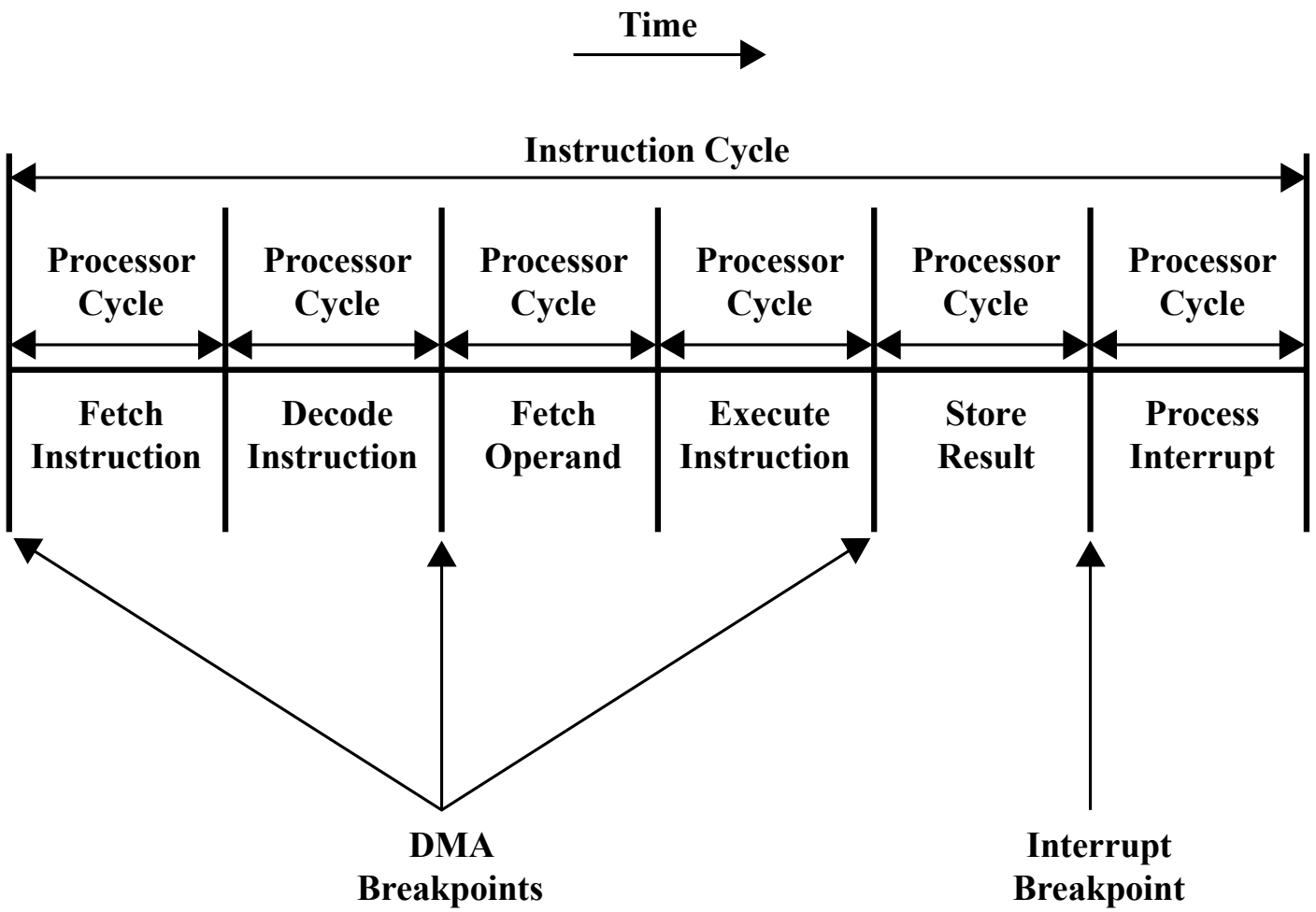
  2) The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer

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

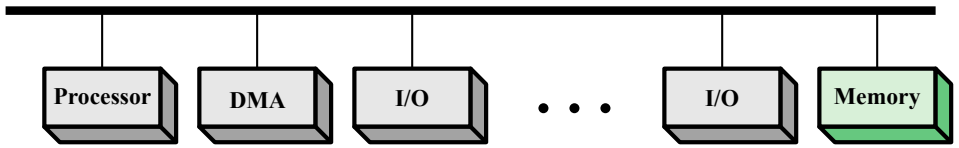
Typical DMA Block Diagram



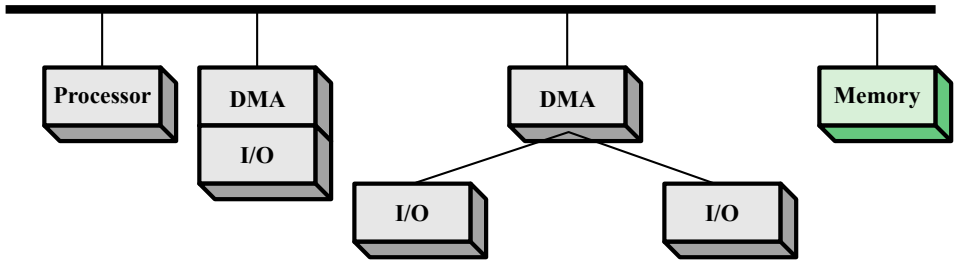
DMA and Interrupt Breakpoints During an Instruction Cycle



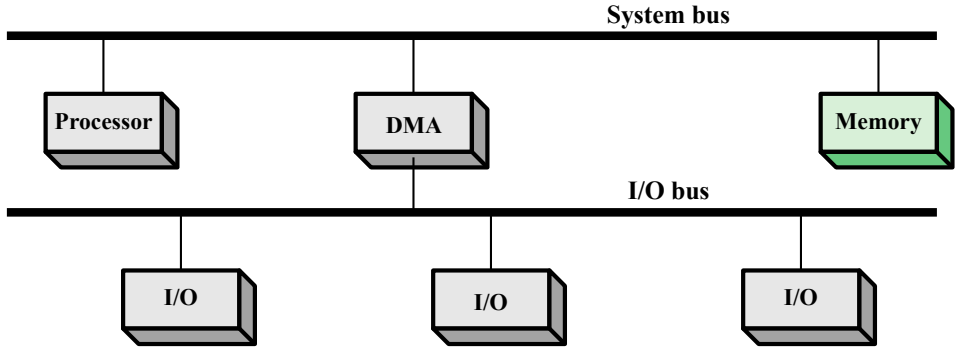
Alternative DMA Configurations



(a) Single-bus, detached DMA

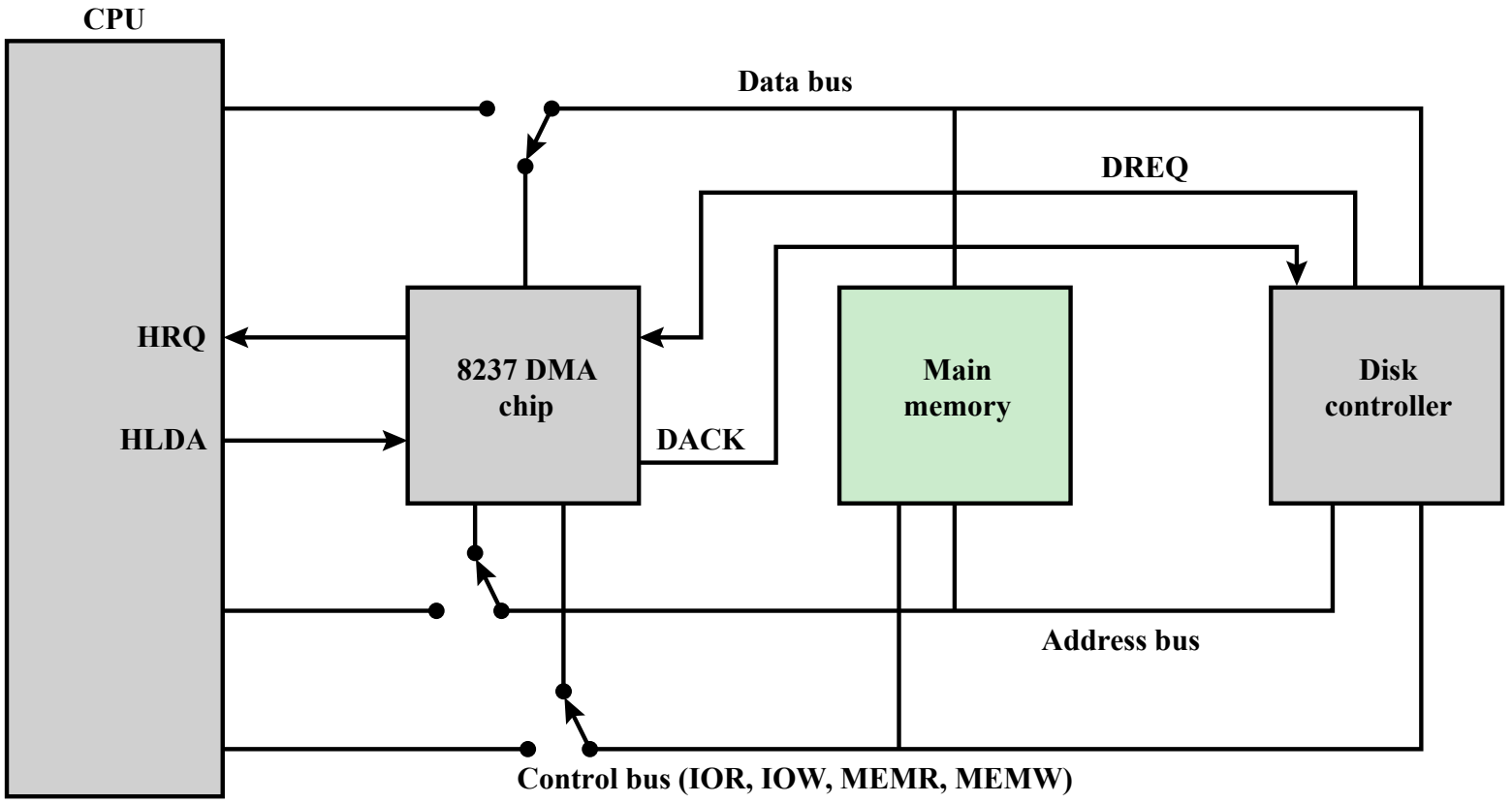


(b) Single-bus, Integrated DMA-I/O



(c) I/O bus

8237 DMA Usage of System Bus



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

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

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.

External Interconnections Standards

- Universal Serial Bus (USB)
  - Widely used for peripheral connections
  - Default interface for slower speed devices, such as keyboard
  - Commonly used high-speed I/O, including printers, disk drives, network adapters
  - Has gone through multiple generations

- FireWire Serial Bus
  - Was developed as an alternative to small computer system interface (SCSI) to be used on smaller systems, such as personal computers, workstations, and servers
  - Objective was to meet the increasing demands for high I/O rates while avoiding the bulky and expensive I/O channel technologies developed for mainframe and supercomputer systems
  - Uses a daisy chain configuration, with up to 63 devices connected off a single port
  - Provides for hot plugging which makes it possible to connect and disconnect peripherals without having to power the computer system down or reconfigure the system
  - Provides for automatic configuration

External Interconnections Standards

- SCSI: Small Computer System Interface
  - A once common standard for connecting peripheral devices (disks, modems, printers…) to small and medium-sized computers
  - Has lost popularity to USB and FireWire in smaller systems
  - High-speed versions remain popular for mass memory support on enterprise systems (some IBM mainframes offer support for SCSI)

- Thunderbolt
  - Most recent and fastest peripheral connection technology to become available for general-purpose use
  - Developed by Intel with collaboration from Apple
  - The technology combines data, video, audio, and power into a single high-speed connection for peripherals such as hard drives, RAID arrays, video-capture boxes, and network interfaces

External Interconnections Standards

- InfiniBand
  - I/O specification aimed at the high-end server market
  - Heavily relied on by IBM zEnterprise series of mainframes
  - Standard describes an architecture and specifications for data flow among processors and intelligent I/O devices
  - Has become a popular interface for storage area networking and other large storage configurations
  - Enables servers, remote storage, and other network devices to be attached in a central fabric of switches and links

- SATA: Serial Advanced Technology Attachment
  - An interface for disk storage systems
  - Provides data rates of up to 6 Gbps, with a maximum per device of 300 Mbps
  - Widely used in desktop computers and in industrial and embedded applications

External Interconnections Standards

- PCIExpress
  - High-speed bus system for connecting peripherals of a wide variety of types and speeds
- Ethernet
  - Predominant wired networking technology
  - Has evolved to support data rates up to 100 Gbps and distances from a few meters to tens of km

- WiFi
  - Predominant wireless Internet access technology
  - Public hotspots have expanded dramatically to provide free Internet access
  - As the technology of antennas, wireless transmission techniques, and wireless protocol design has evolved, the IEEE 802.11 committee has been able to introduce standards for new versions of Wi-Fi at higher speeds