# Computer Architecture

Course Code: CSE360

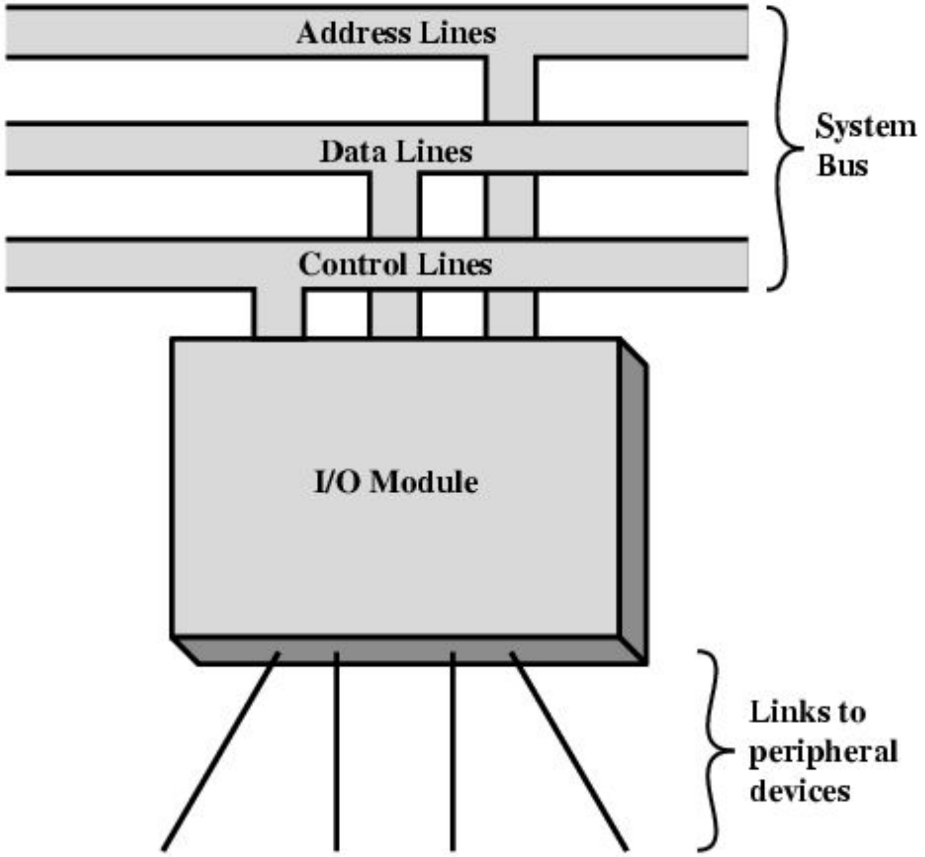
## Lecture 6

## Input/Output

## Input/Output Problems

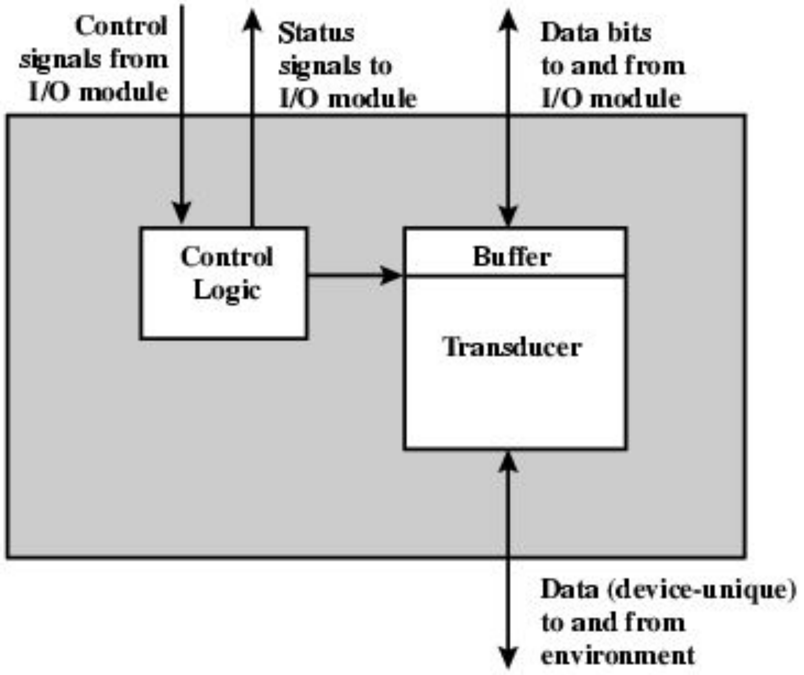
- Wide variety of peripherals with various methods of operations
  - Delivering different amounts of data
  - At different speeds
  - In different formats
- All the devices are slower than CPU and RAM
- Need I/O modules

# Generic Model of I/O Module



- Each module interfaces the CPU and memory via system buses and controls, and one or more peripheral devices
- The I/O module contains **logic for performing a communication** function between the peripheral and the bus.

# External Device Block Diagram



Control signals from I/O module
Status signals to I/O module
Data bits to and from I/O module
Control Logic
Buffer
Transducer
Data (device-unique) to and from environment

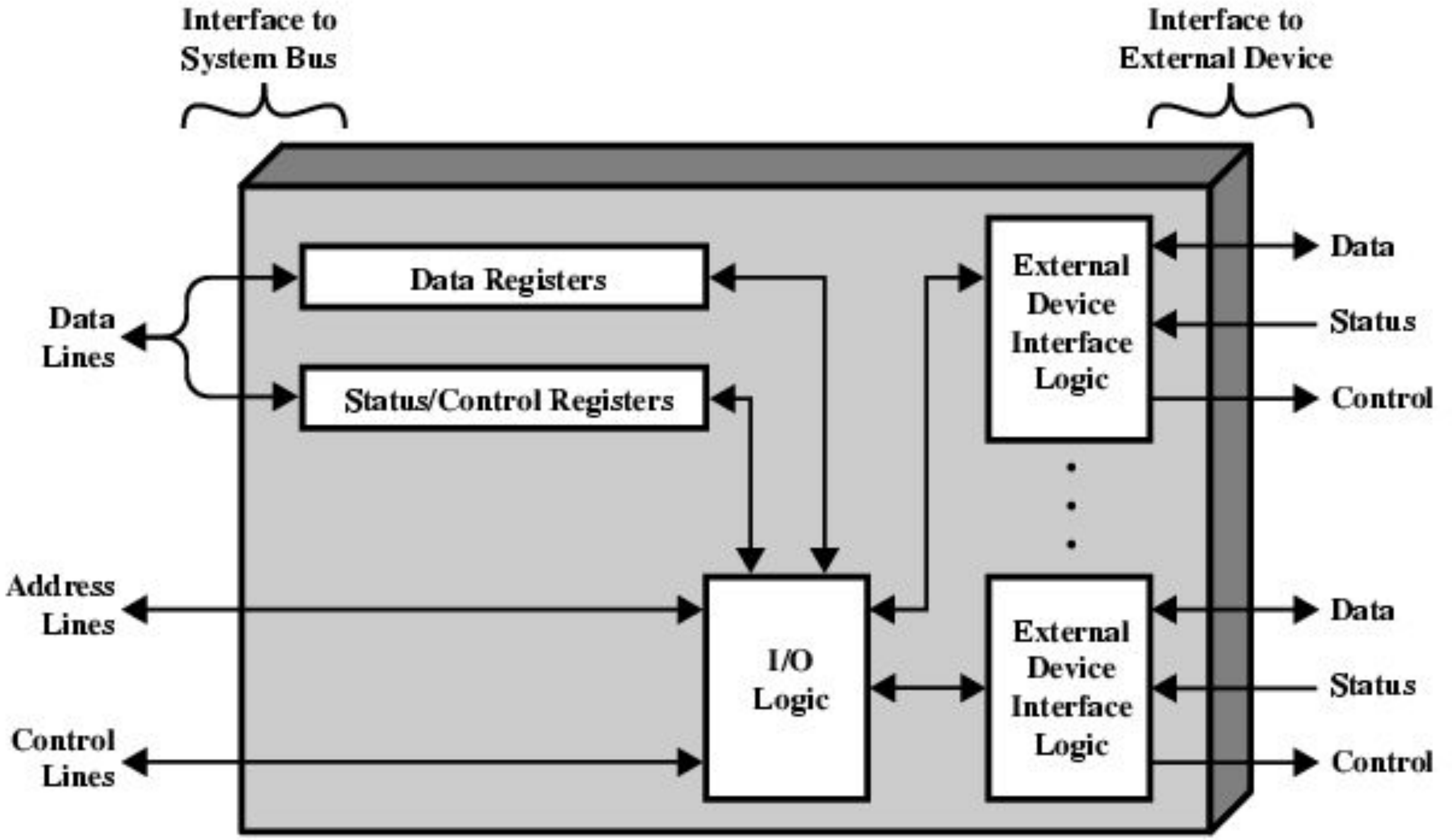
- Interface to the I/O module in the form of control, data, and status signals.
- Control signals determine function that device will perform, such as send data to I/O module (INPUT or READ), accept data from I/O module (OUTPUT or WRITE), report status, or perform control function (e.g. position a disk head)
- Data are in the form a set of bits to be sent to or received from I/O module
- Status signals indicate (READY/NOT-READY) during data transfer
- Control logic <u>controls external device's operation</u> in response to direction from I/O module
- The transducer converts data from electrical to other forms of energy during output and from other forms to electrical during input
- A buffer is associated with transducer to temporarily hold data being transferred between I/O module and external environments (a buffer size of 8 to 16 bits is common)

## I/O Steps

The control of the transfer of data from an external device to the processor involves following steps:

- CPU interrogates I/O module to check the status of the attached external device
- I/O module returns the device status
- If the device is ready, CPU requests data transfer
- I/O module gets data (e.g. 8 or 16 bits) from external device
- I/O module transfers data to CPU

## I/O Module Diagram



## I/O Module Structure

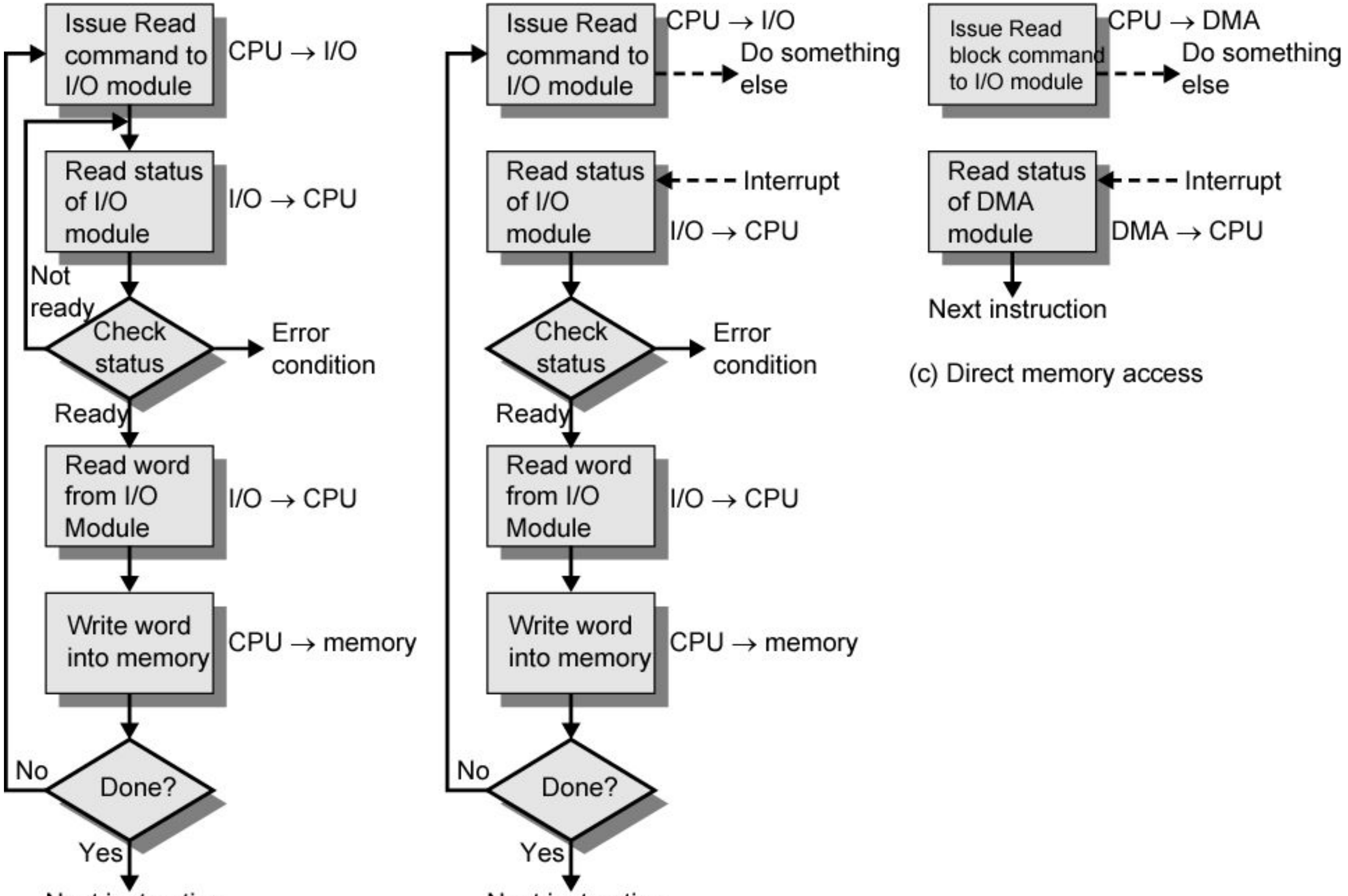
- **Data transferred to and from the module are buffered in one or more <u>data registers</u>**
- There may also one or more status registers that provide current status information
- **A <u>status register</u> may also function as control register, to accept detailed control information from the processor**
- **The <u>logic</u>** within the module **interacts with the processor via a set of <u>control lines</u>.**
- The processor uses <u>control lines</u> to issue commands to the I/O module
- Some of the control lines may be used by the I/O module
- The module must also be able to recognize and generate <u>addresses</u> associated with the devices it controls
- Each I/O module has a unique address or, if it controls more than one external device, a unique **set of addresses**
- Finally, the I/O module contains <u>logic specific to the interface with each external device</u> that it controls

# Input Output Techniques

- **Three techniques are possible for I/O operations**
- **Programmed I/O**
  - Data are exchanged between processor and I/O module
  - **The processor executes a program** that gives direct control of I/O operation, including sensing **device status**, **sending a read or write command**, and **transferring data**
  - **When processor issues a command to I/O module, it must wait until I/O operation is complete**
  - **If the processor is faster than I/O module, this is wasteful of processor time.**
- **Interrupt driven I/O**
  - The processor issues an I/O command, continues to execute other instructions, and **is interrupted by the I/O module** when the later has completed its work

  ☐ <u>With both programmed and interrupt I/O, **the processor is responsible** for extracting data from main memory for output and storing data in main memory for input</u>
- **Direct Memory Access (DMA)**
  - The I/O module and main memory exchange data directly without processor involvement.

# Three Techniques for Input of a Block of Data



(a) Programmed I/O

(b) Interrupt-driven I/O

(c) Direct memory access

## Programmed I/O

- CPU has direct control over I/O
  - Sensing status
  - Read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time

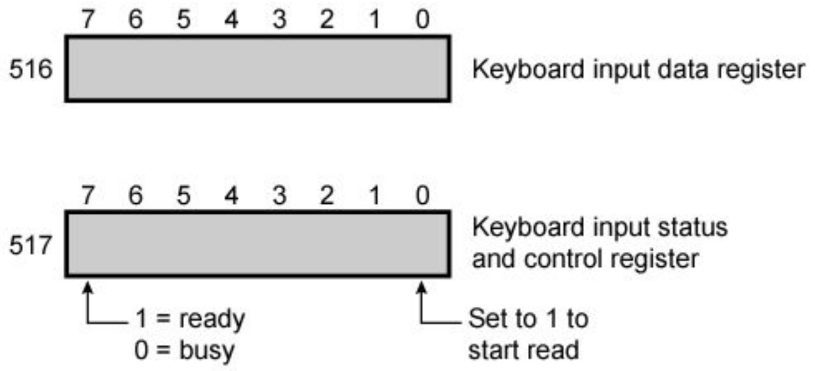
## Programmed I/O - detail

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU
- CPU may wait or come back later

## I/O Mapping

- **Memory mapped I/O**
  - Single address space for memory locations and I/O devices
  - The **processor treats the status and data registers of I/O modules as memory locations** and uses same machine instructions to access both memory and I/O devices
  - **Devices and memory share an address space (e.g. with 10 address lines, a combined total of 1024 memory locations and I/O address can be supported, in any combination)**
  - **No special commands for I/O**
- **Isolated I/O**
  - The bus may be equipped with memory read and write **plus input and output command**
  - Now the **command line specifies whether the address refers to a memory location or an I/O device**
  - **Separate full range of address spaces for both** (e.g. with 10 address lines, **the system may support both 1024 memory locations and 1024 I/O addresses**
  - **Special commands for I/O**
  - The address space for I/O is isolated from that of memory

# Memory Mapped and Isolated I/O



| 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

- **Keyboard input appears to a programmer**
- **Assume a 10 bit address, with a 512-bit memory (locations 0-511) and up to I/O addresses (locations 512-1023).**
- **Two addresses are dedicated to keyboard input from a particular terminal**
- **Address 516 refers to data register and address 517 refers to status register**
- **The program will read 1 byte of data from keyboard into an accumulator register in the processor. The processor loops until the data byte is available.**

- **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.**

## I/O Mapping (Advantages and Disadvantages)

- Memory mapped I/O
  - The advantage is that a **large selection of memory access commands available**, allowing more efficient programming
  - **A disadvantage is that valuable memory address is used up**
- Isolated I/O
  - **If isolated I/O is used, there are only a few I/O instructions**; which is treated as major advantage
  - **Limited set of special commands for I/O**; which is the limitation

## Interrupt Driven I/O

- Overcomes CPU waiting
- No repeated CPU checking of device
- I/O module interrupts when ready

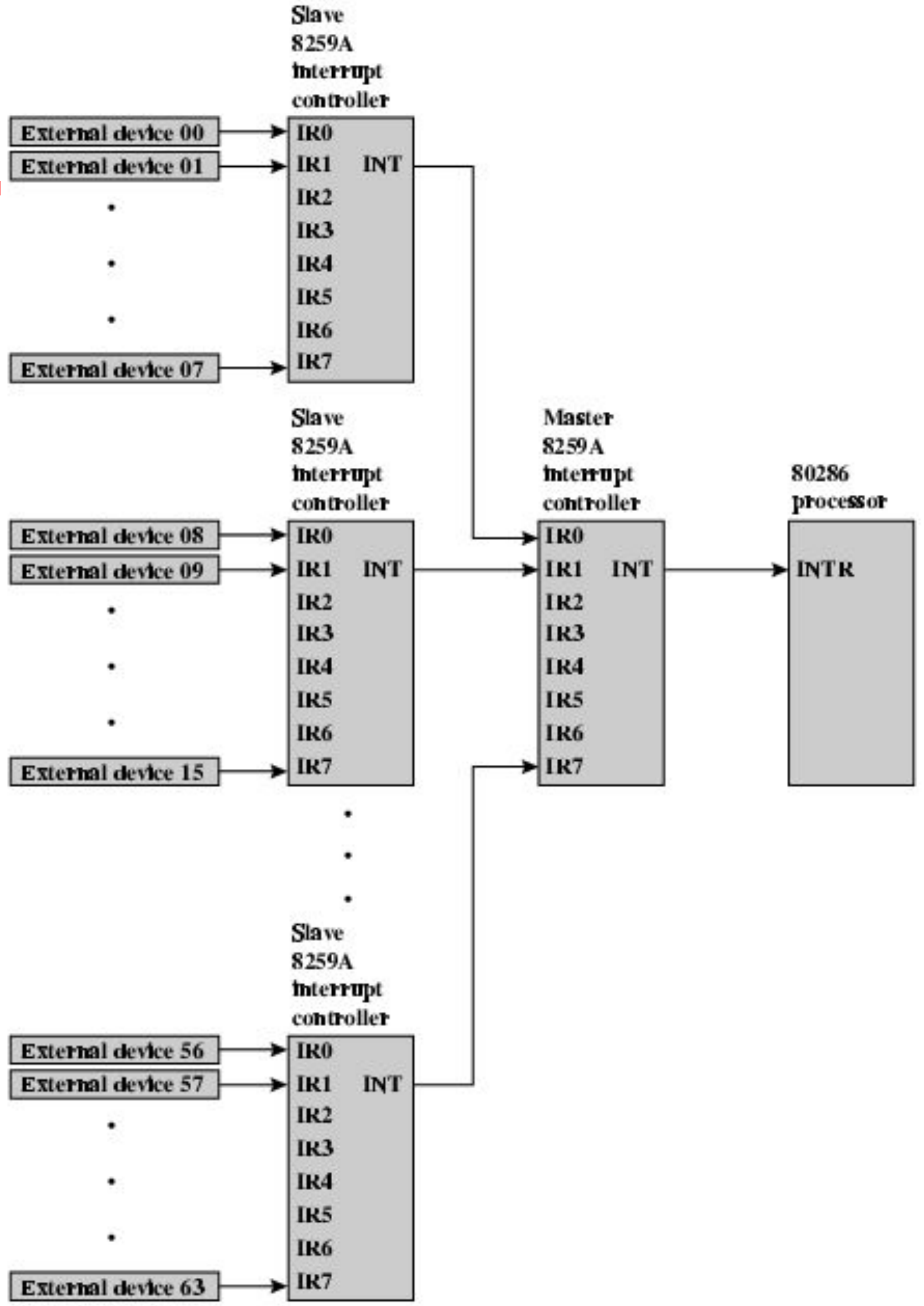
## Interrupt Driven I/O
## Basic Operation

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

## 82C59A Interrupt Controller

- Figure shows the use of the 82C59A to connect multiple I/O modules
- A single 82C59A can handle CPU upto 8 modules
- The 82C59A's sole responsibility in the management of interrupt
- It accepts requests from attached modules, determines which interrupt has the highest priority and then signals the processor by raising the INTR line
- The processor acknowledges via INTA line
- This prompts the 82C59A to place the appropriate vector information on data bus

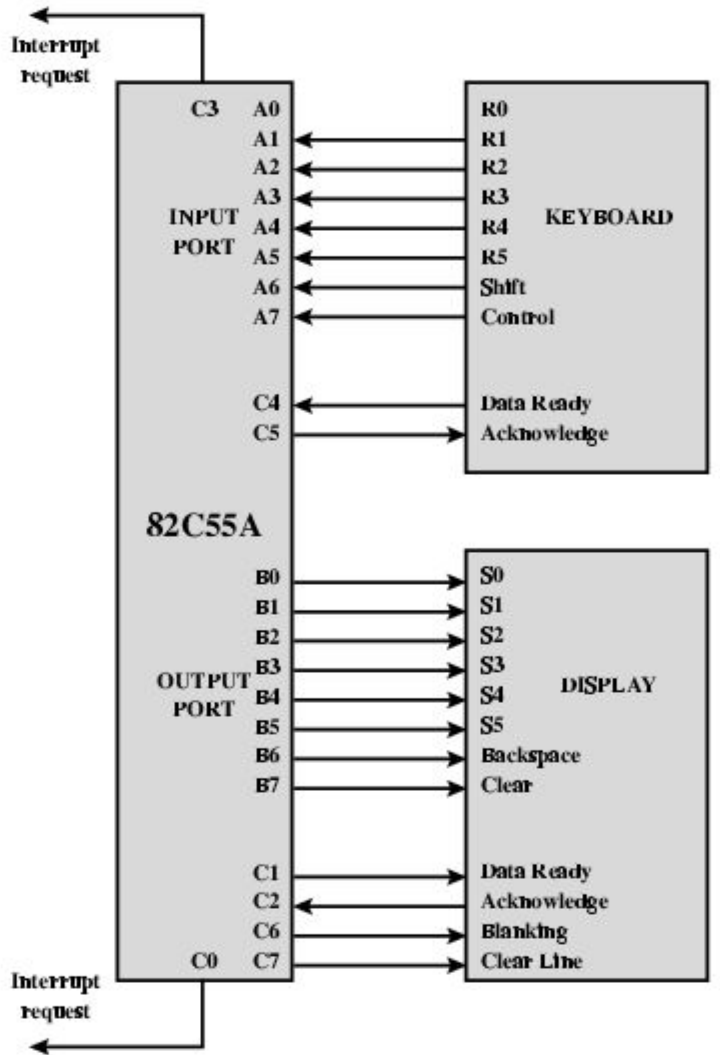
# 82C59A Interrupt Controller



## Keyboard/Display Interfaces to 82C55A

- The figure illustrates the use of 82C55 PPI to control and keyboard/display terminal
- The keyboard provides 8-bit of input
- Two of these bits, SHIFT and CONTROL, have special meaning to the keyboard handling program executing in the processor
- Two handshaking control line are provided for use with the keyboard
- The display is also linked by an 8-bit data port
- Two of the bits have special meanings that are transparent to the 82C55A
- In addition to two handshaking lines, two lines provide additional control functions.

# Keyboard/Display Interfaces to 82C55A



## Direct Memory Access

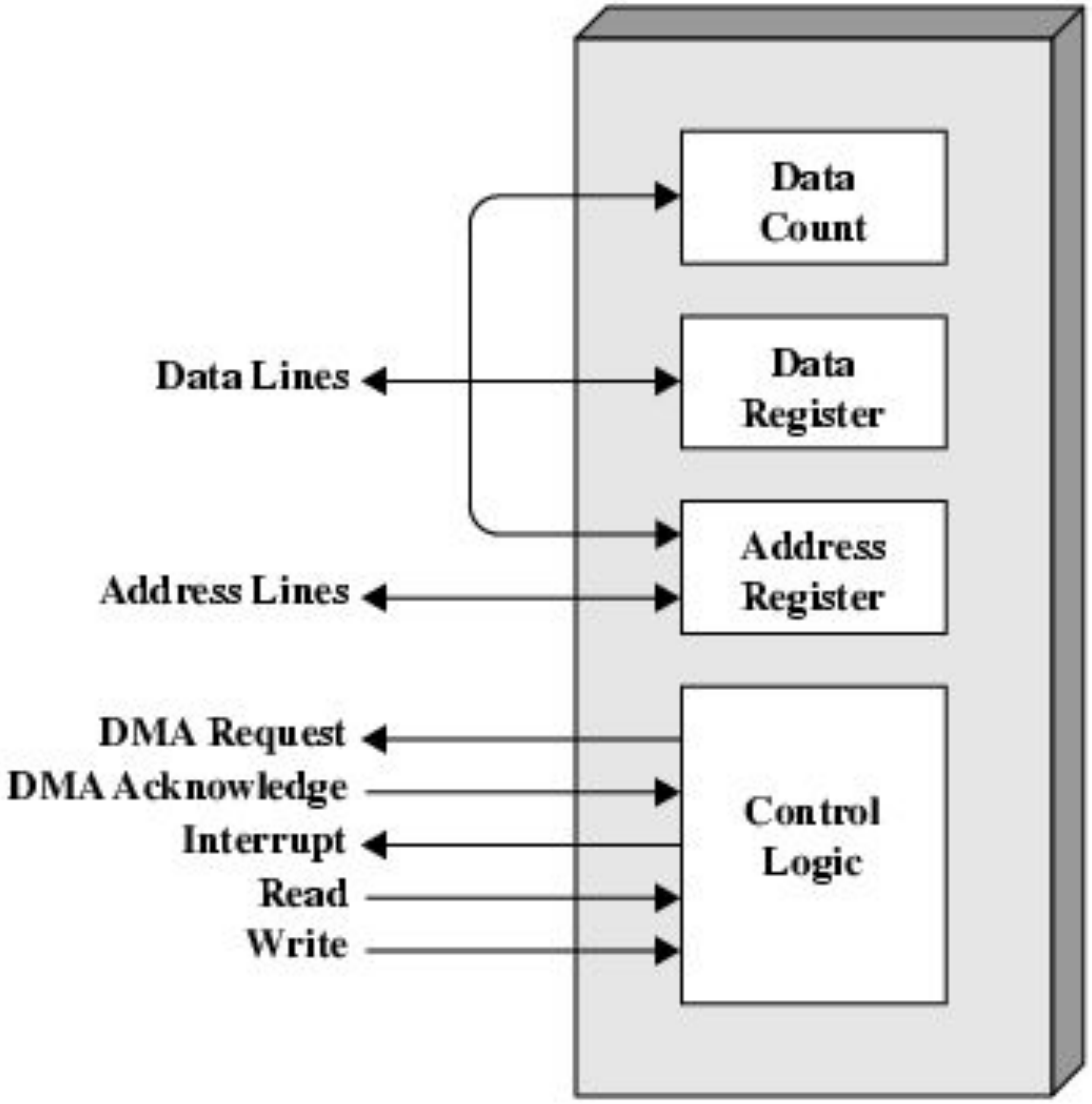
- **Interrupt driven and programmed I/O require active CPU intervention to transfer data between memory and an I/O module**
  - The **I/O transfer rate is limited by the speed**
  - **CPU is tied up in managing an I/O transfer**; a number of instructions must be executed for each I/O transfer

- **Using simple programmed I/O**, the processor is dedicated to the task of I/O and **can move data at high rate at the cost of doing nothing else**
- **Interrupt I/O frees up the processor to some extent at the expense of the I/O transfer rate**.
- Nevertheless, **both methods have an averse impact on both processor activity and I/O transfer rate**
- DMA is the answer

## DMA Function

- DMA involves an additional Module (hardware) on the system bus
- **DMA controller takes over control of the system bus from processor**
- It needs to do this to transfer data to and from memory over the system bus
- **The DMA module use the bus only when processor does not need it, or it must force the processor to suspend operation temporarily**
- **The later technique is referred to as cycle stealing, because DMA module steals a bus cycle**

## Typical DMA Block Diagram



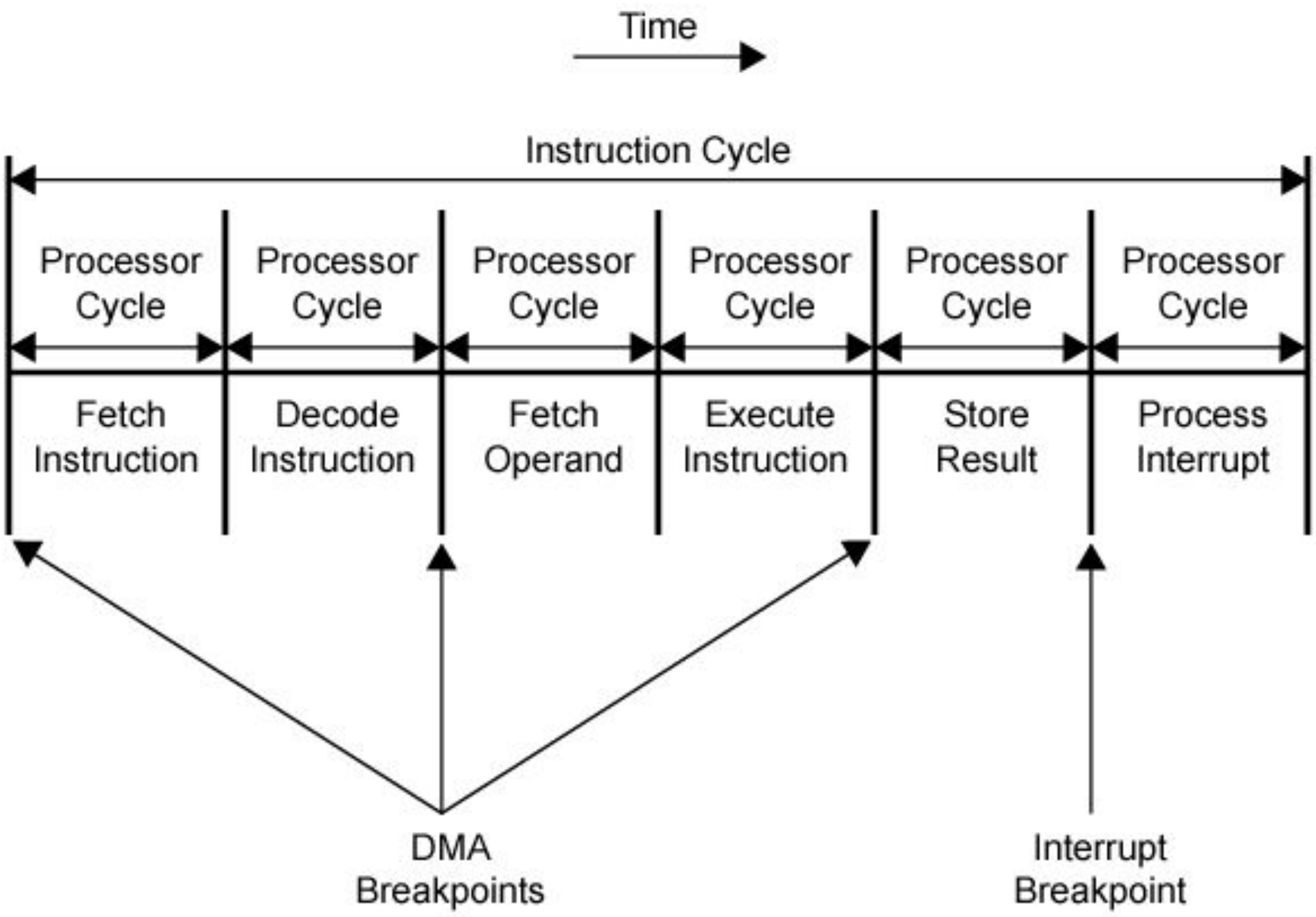
## DMA Operation

- CPU tells DMA controller:-
  - **To Read/Write data**
  - **Address the I/O Device address**
  - **The starting address of memory to read from or write to data** and **stored by the DMA module in its address register**
  - **Number of words to be read or written via the data lines and stored in data count register**
- **<u>The CPU then carries on with other work</u>**
- **<u>DMA controller deals with data transfer directly to or from memory without going through processor</u>**
- **<u>DMA controller sends interrupt when finished</u>**
- **The processor is involved only at the beginning and end of data transfer**

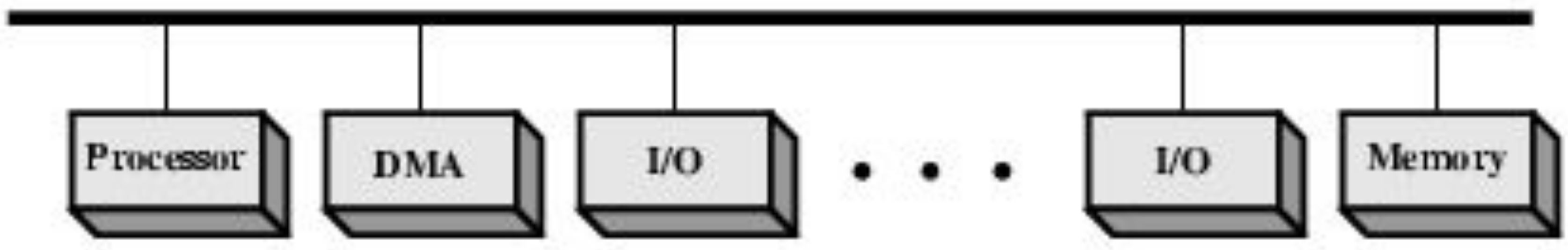
## DMA Transfer Cycle Stealing

- **DMA controller takes over bus for a cycle**
- **The DMA Transfers one word of data and returns control to the processor**
- **This is not an interrupt; rather the processor pauses for one bus cycle**
- **CPU suspended just before it accesses bus**
  - i.e. before an operand or data fetch or a data write
- **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
- Figure shows where in the instruction cycle the processor may be suspended.

# DMA and Interrupt Breakpoints During an Instruction Cycle



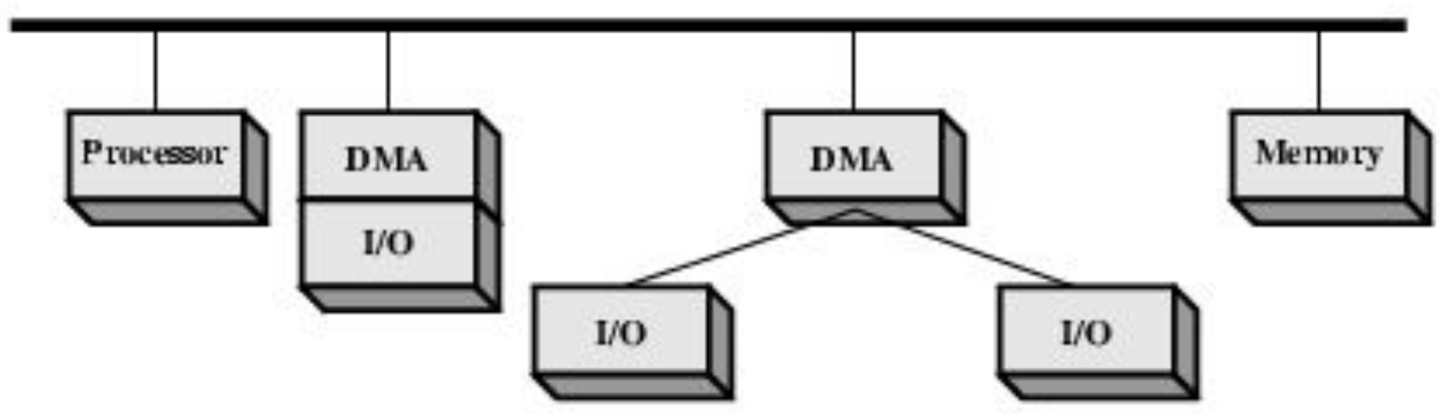
## DMA Configurations (1)



(a) Single-bus, detached DMA

- **The DMA mechanism can be configured in a variety of ways**
- **Single Bus, Detached DMA controller**
- All modules share the system bus
- **Each transfer of a word consumes two bus cycles: I/O to DMA then DMA to memory**
- **CPU is suspended twice**

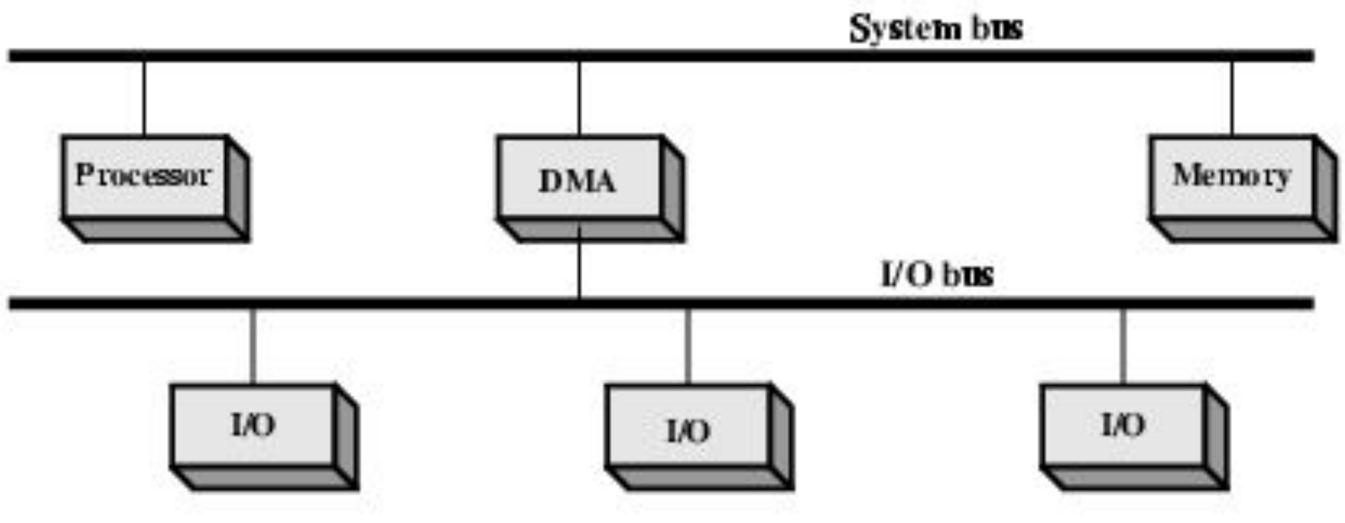
## DMA Configurations (2)



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

- **Single Bus, Integrated DMA controller**
- **Controller may support >1 device**
- **Each transfer uses bus once: DMA to memory**
- **CPU is suspended once**

## DMA Configurations (3)



(c) I/O bus

- **Separate I/O Bus**
  - Bus supports all DMA enabled devices
- **Each transfer uses bus once**
  - **DMA to memory**
- **CPU is suspended once**

## Intel 8237A DMA Controller

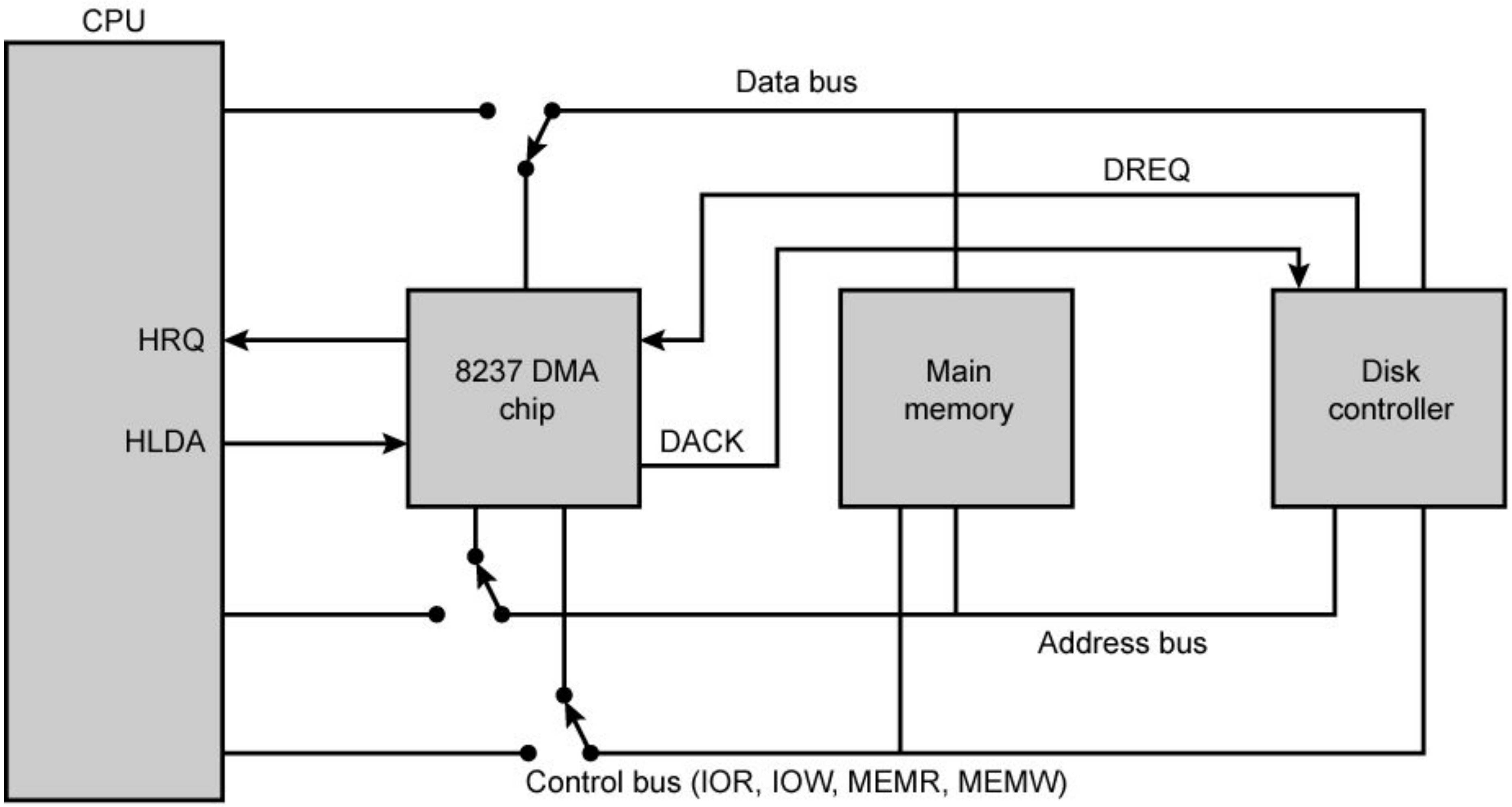
- Interfaces to 80x86 family and DRAM
- When DMA module needs buses it sends HOLD signal to processor
- CPU responds HLDA (hold acknowledge)
  - DMA module can use buses
- E.g. transfer data a block of data from memory to disk

1. The peripheral device (such as disk controller) requests service of DMA by pulling DREQ (DMA request) high
2. DMA puts high on HRQ (hold request), signaling  the CPU through its HOLD pins that it needs to use bus

## Intel 8237A DMA Controller

3. CPU finishes present bus cycle (not necessarily present instruction) and respond to DMA request by putting high on HDLA (hold acknowledge), thus telling the DMA that it can go ahead and use the buses to perform its tasks. HOLD remains active high as long as DMA is performing its tasks
4. DMA activates DACK (DMA acknowledge), telling the peripheral device to start transfer
5. DMA starts transfer from memory to peripheral by putting address of first byte of the block on the address bus and activating MEMR; it then activates IOW to write to peripheral. Then DMA decrements the counter and increments the address pointer.  Repeat the process until count reaches zero and task is finished.
6. DMA deactivates HRQ, giving bus back to CPU

## 8237 DMA Usage of Systems Bus



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

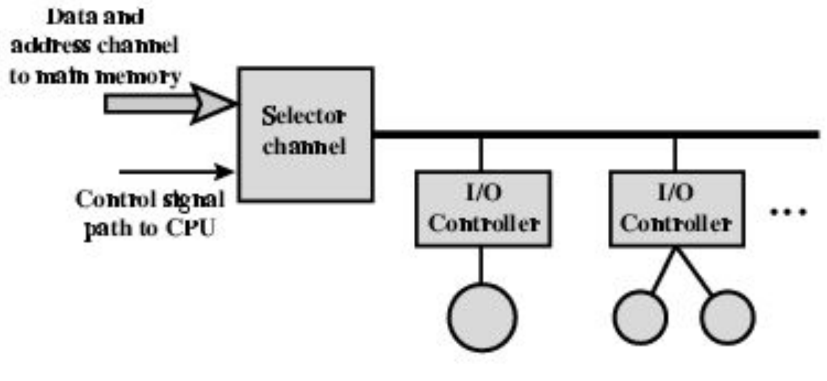
## I/O Channels

- **I/O devices getting more sophisticated**
- **e.g. 3D graphics cards**
- CPU instructs I/O controller to do transfer
- **I/O controller does entire transfer**
- **Improves speed**
  - **Takes load off CPU**
  - Dedicated processor is faster

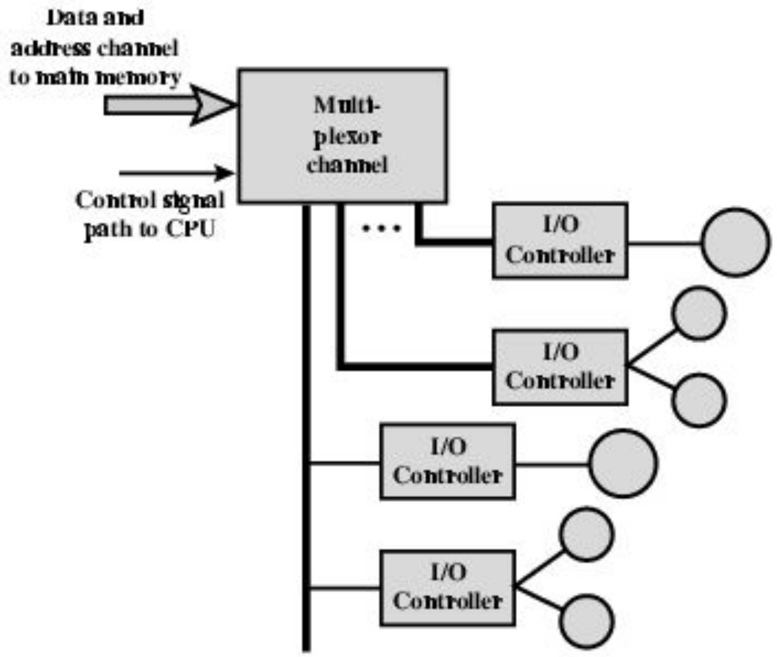
## The Evolution of the I/O function

- The CPU directly controls a peripheral device.
- A controller or I/O module is added. **The CPU uses programmed I/O without interrupts**
- **The same configurations 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 preformed, **increasing efficiency**
- **The I/O module is given direct access to memory via DMA**. **It can transfer data to or from memory without involving CPU**
- The I/O module (often referred to as **I/O channel**) is enhanced to become a processor, with a specialized instruction set for I/O. **The CPU directs I/O processor to execute I/O program in memory**. **The I/O processor fetches and executes these instructions without CPU intervention**
- **The I/O module has a local memory of its own**. With this architecture, **a large set of I/O devices can be controlled with minimal CPU involvement**

# I/O Channel Architecture


Data and address channel to main memory
Selector channel
Control signal path to CPU
I/O Controller
I/O Controller
• • •

(a) Selector



(b) Multiplexor

- Two types of I/O channels are common.
- <u>A **selector channel** controls multiple high-speed devices and at any one time, is dedicated to the transfer of data with **one** of those devices</u>
- <u>Thus, the I/O channel selects **one** device and effects the data transfer</u>
- Each device or a small set of devices, is handled by a controller or I/O module
- Thus I/O channel serves in place of the CPU in controlling these I/O controllers
- <u>A **multiplexor channel** can handle I/O with **multiple** devices at the same time.</u>
- For low-speed device, a byte multiplexor accepts or transmits characters as fast as possible to multiple devices
- For high speed devices, a block multiplexer interleaves blocks of data from several devices