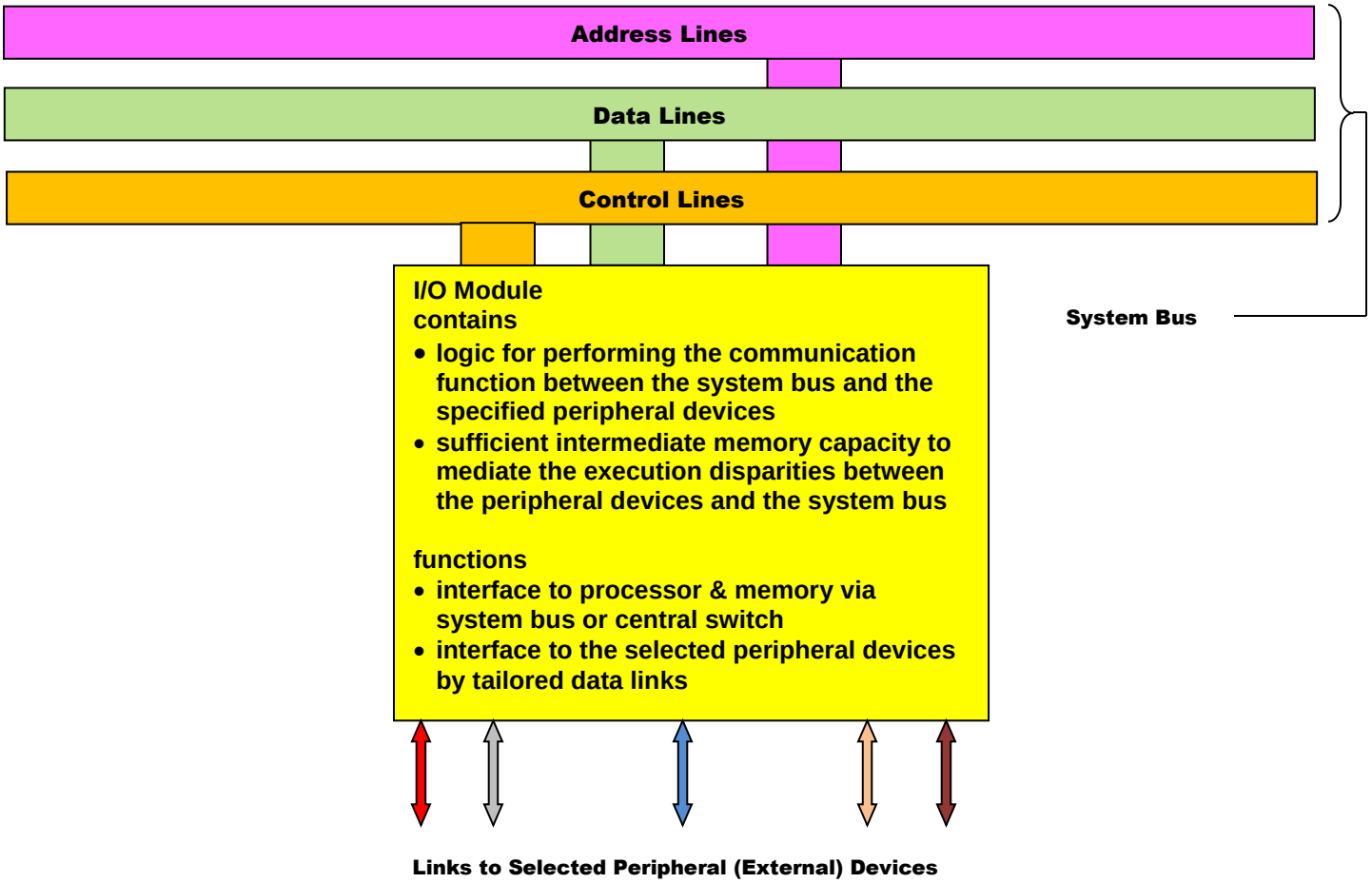
Input/Output

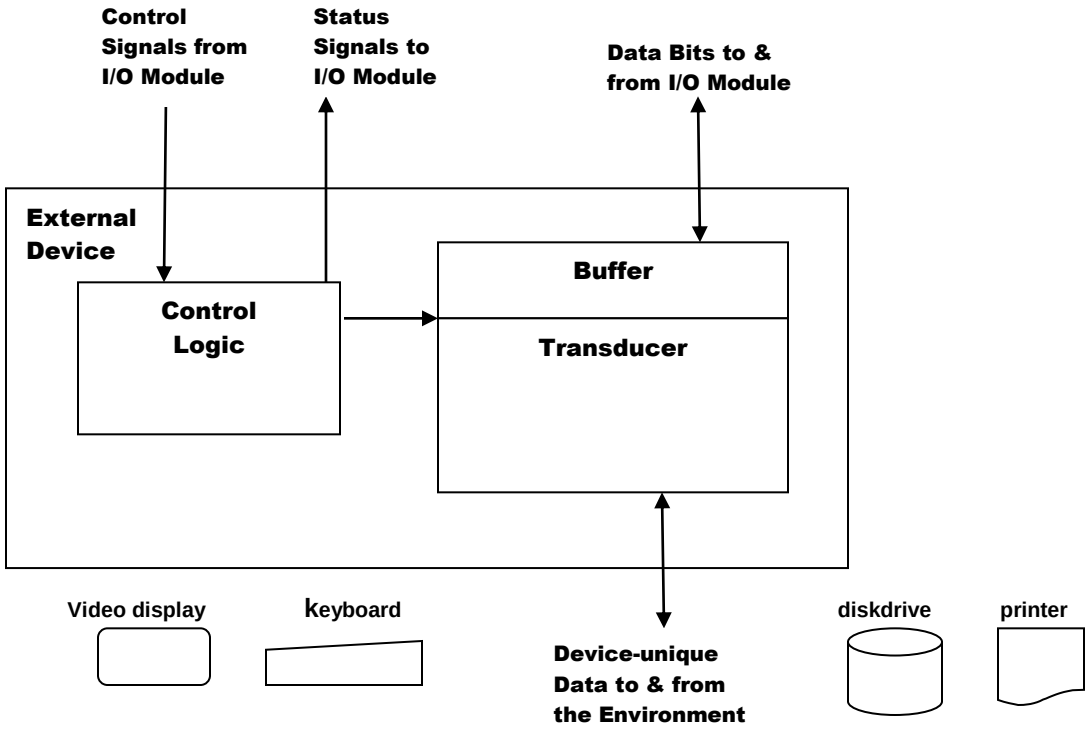
I/O Techniques
- Programmed I/O
- Interrupt-driven I/O
- Direct memory access (DMA)
  - specialized I/O Processor

External I/O Interfaces
- FireWire
- Infiniband



External Devices (Peripheral Devices)

- attached to the system via an I/O module designed for that particular type of device
- link mediates the exchange of control, status & data between the I/O module and the device
- human readable devices – video display units & printers
- machine readable – magnetic disk units, tape units, DVD units, sensors, activators, etc.
- communication – exchange data with remote devices, either human or machine readable



Control Signals
- Determine function that the device will perform
  - Send data to I/O module
  - Receive data from I/O module
  - Report status
  - Perform control operation

Data
- Data bits sent to or received from the device

Status Signals
- Report the state of the device to perform tasks

Control Logic
- Transforms the control signals & status signals into active directions for the I/O module to execute

Transducer
- Converts data to & from electrical form to diverse forms required by the various external devices

Keyboard
- key code – left shift vs. right shift
- character code – IRA code -- 7 bit code
- user depresses key – electronic signal translated by transducer to IRA code
- IRA code is transmitted to I/O Module
- Text is transmitted to the computer

Video Display Unit
- Text is transmitted to the I/O Module
- IRA code is transmitted to the transducer
- Transducer sends the appropriate signal to the display unit to either paint the desired character on the screen or to perform the required control action

Disk Drive
- Fixed-head disk – transducer converts between magnetic patterns on disk to electronic bits in the device buffer
- Moving-head disk – also cause he disk arm to move radially in & out across the disk surface

I/O Modules

- Functional Requirements

  - Control & Timing
    - coordinate flow of information between internal & external devices
    - e.g., control of data from an external device to the processor
      - processor requests I/O module status report for attached device
      - I/O module returns the requested status report
      - If device is ready to transmit, processor requests transfer of data from I/O module
      - I/O module obtains a unit of data from external device
      - data is transferred from I/O module to the processor

    each action requires one or more bus arbitrations

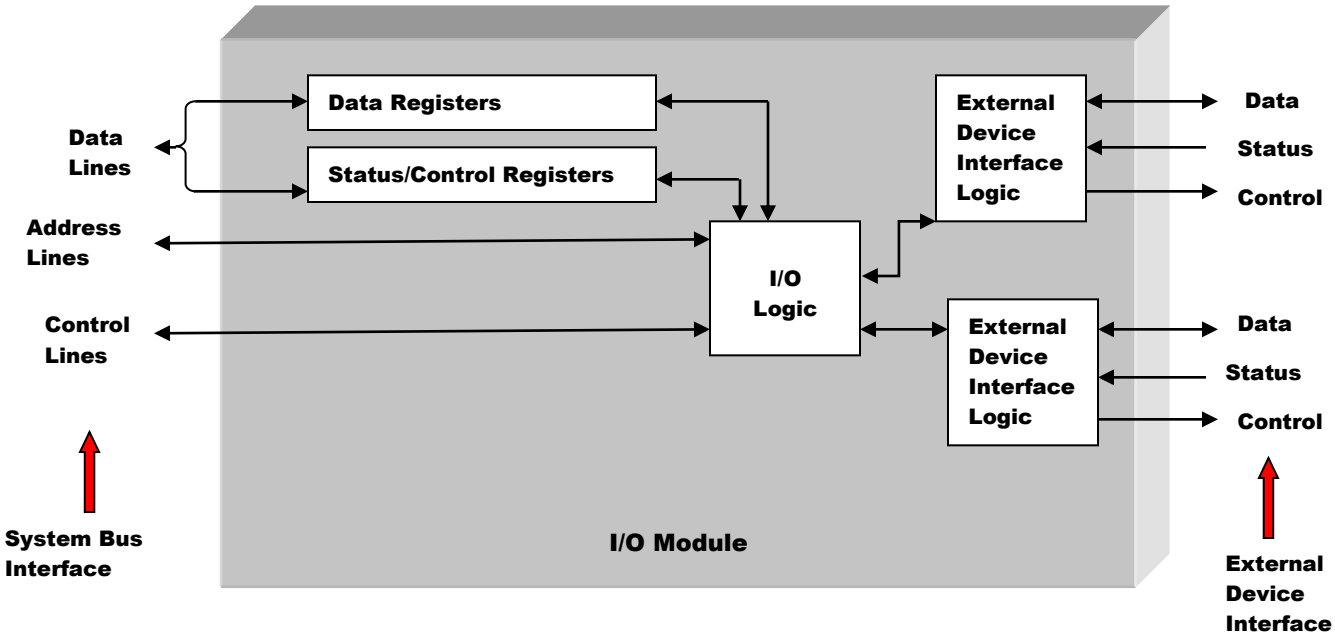
  - Processor Communication
    - Command Decoding
      - Signals sent on Control Bus
      - Disk Drive I/O Module Commands
        - ✓ **READ SECTOR**
        - ✓ **WRITE SECTOR**
        - ✓ **SEEK** <track number>
        - ✓ **SCAN** <record ID>
    - Data

    sent on data bus

    - Status Reporting
    - Address recognition
      - each I/O device attached to an I/O module has a unique address which must be transmitted along with any command directed to that device

  - Device Communication

  - Data Buffering
    - data sent from main memory arrives as a rapid burst
    - data is stored in the buffer & sent to an external device at its data rate
    - data arrives from an external device at its data rate and accumulated in the buffer
    - data is sent in a rapid burst from the buffer to main memory &/or the processor
  - Error Detection
    - mechanical & electrical malfunctions reported by the external devices
    - unintentional changes in the bit patterns which occur during transmission



I/O Channel or **I/O Processor** **(Main Frames)**

- **I/O Module**
  - **that assumes detailed control and**
  - **which provides a high-level interface to the processor**

I/O Controller or **Device Controller** **(Micro Computers)**

- **a primitive I/O Module that leaves detailed control to the processor**

Overview

Programmed I/O

- **processor is in direct control of I/O operations, i.e., responsible for**
  - **sending read, write commands**
  - **sensing device status**
  - **accessing data from memory for output**
  - **storing data in memory from input**
- **processor must wait for I/O completion, i.e., waste CPU processing cycles**

Interrupt-Driven I/O

- **processor is responsible for**
  - **sending read, write commands**
  - **accessing data from memory for output**
  - **storing data in memory from input**
- **processor**
  - **may execute other instructions while I/O operations are in progress**
  - **is interrupted by the I/O module when the I/O transfer is complete**

Direct Memory Access

- **processor sends memory base address, range, and I/O operation information to DMA unit**
- **under the control of the DMA, the I/O Module & Memory exchange data directly without further processor involvement**
- **after the I/O operation is complete, the DMA issues an interrupt for the processor**

Programmed I/O

- processor encounters an I/O instruction
  → issues an address specifying a particular I/O module & external device
  →issues an I/O command
- I/O module performs the requested action
  → sets appropriate bits in the I/O status register
- processor must periodically check the I/O module status bits for indication of completion

- I/O Commands
  - Control – activate a specific device & request a specific action
  - Test – check on specific status conditions, e.g., ***errors, ready***, etc.
  - Read – causes I/O module to obtain a data item & place it in a the buffer; processor can obtain the data item by requesting the I/O module to place it on the data bus
  - Write – causes the I/O module to take an item of data from the data bus and transmit that data item via the buffer space to the specified peripheral device

I/O related instructions that the processor fetches from memory ← 1 to 1→ I/O commands that the processor issues to the I/O module

Processor, Main Memory, & I/O
on a
Common Bus

Memory Mapped I/O

- single address space for Memory Locations & I/O Devices
- status & data registers are treated as memory locations by processor
- uses same machine instructions to access both Memory & I/O Devices
- single read statement & single write statement transfers data through the I/O Module
- 10 address lines support 2^{10} = 1024 memory locations
- LARGE SET OF INSTRUCTIONS FOR REFERENCING MEMORY → EFFICIENT PROGRAMMING

Isolated I/O

- separate Read & Write commands to access memory
- separate Input & Output commands to access the devices
  - address space for I/O is isolated from the address space for memory
- 10 address lines support 1024 memory locations & 1024 I/O addresses
- FEW I/O INSTRUCTIONS → INEFFICIENT PROGRAMMING

Interrupt-Driven I/O

- **I/O Module Viewpoint**
  - **Input – I/O Module**
    - **receives a READ command from the processor**
    - **reads data in from the peripheral device, when it is done, it**
    - **signals an interrupt to the processor over a control line**
    - **when the processor requests the data,**
      the I/O Module places the data on the data bus
- **Processor Viewpoint**
  - **Input – Processor**
    - **issues a READ command**
    - **works on other things**
    - **checks for interrupts at the end of each instruction cycle,**
    - **when an interrupt occurs,**
      saves the context of the current program in a process control block
    - **processes the interrupt, i.e., reads the word of data from the I/O Module & writes it to memory**
    - **restores the context of a selected process and continues execution**

> **Every word of data that transfers between memory & an I/O module must pass through the processor**

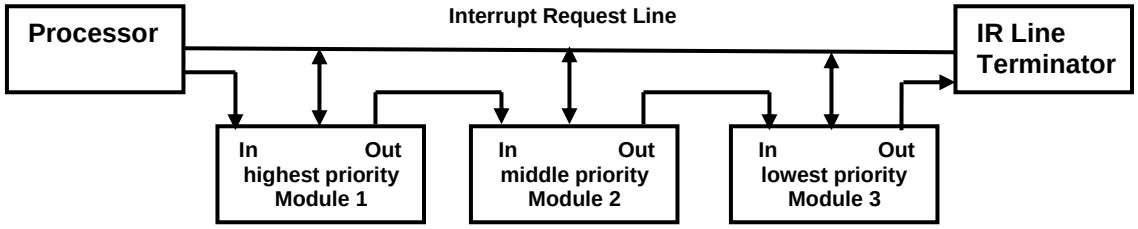
- **Interrupt Processing**
  - **Device issues an interrupt to the processor (sets status bits)**
  - **Processor finishes execution of current instruction**
  - **Processor tests for an interrupt (checks status bits)**
  - **If interrupt as occurred, processor sends acknowledgement signal to the device**
  - **Device removes interrupt signal**
  - **Processor saves context of current process on the system stack**
    - **PSW register (program status word)**
    - **PC register (program counter – location of next instruction in current process)**
    - **Stack & Frame pointer registers**
    - **Contents of all other major registers**
    - **Information on all open files, file pointers, etc**
  - **If there is more than one interrupt pending, processor must determine which one to load**
  - **Processor loads PC with the address of the appropriate interrupt-handling program**
  - **Control passes to the interrupt-handling routine**
  - **Interrupt-Handling Routine processes the interrupt**
  - **If other interrupts are pending, the processor determines which one to load and repeats the last three steps**
  - **If there are no further interrupts to process, a selected process control block is retrieved from the stack, all the registers, etc. are reset and the interrupted process is allowed to continue from the point at which it was interrupted**

- **Multiple Interrupt Processing Techniques**
  - **Multiple Interrupt Lines**
    - **limited number of lines/pins**
    - **each line may have multiple interrupts pending**
    - **for multiple interrupts, the processor picks the line with the highest priority**

- **Software Polling**
  - **Interrupt Service Routine polls each I/O module to determine which caused the interrupt**
    - ⇨ **If there is a separate command line, TESTI/O; processor raises TESTI/O line and places the address of a particular I/O Module on the address line; I/O Module reacts positively if it set the interrupt**
    - ⇨ **If each I/O Module has an addressable status register; processor reads the content of each status register to identify the interrupting module**
  - **Once the I/O Module is identified, the processor branches to a device-service routine specific to that I/O Module**
  - **The software polling order determines the priority for processing multiple interrupts**

software polling is time-consuming, i.e., inefficient

- **Daisy Chaining (Vectored Hardware Polling)**



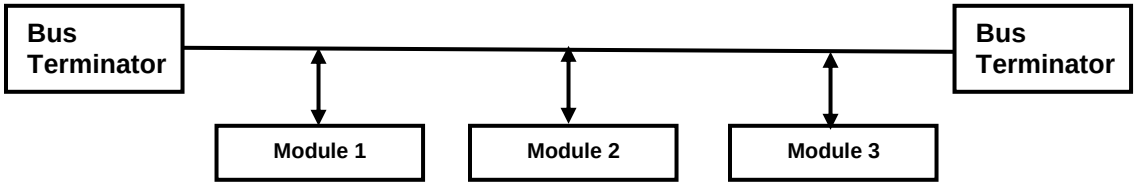
Interrupt Acknowledgement Line is daisy chained through the modules

vectored interrupt

- **When the processor senses an interrupt, it sends an interrupt acknowledgment on the Interrupt Acknowledgement Line which propagates through the modules until it reaches the requesting module, i.e., the module that sent the interrupt**
- **The requesting module places a word (vector) containing the address of the I/O Module or other unique identifier on the data lines**
- **The processor uses the content of the word (vector) as a pointer to the appropriate interrupt processing device-service routine**

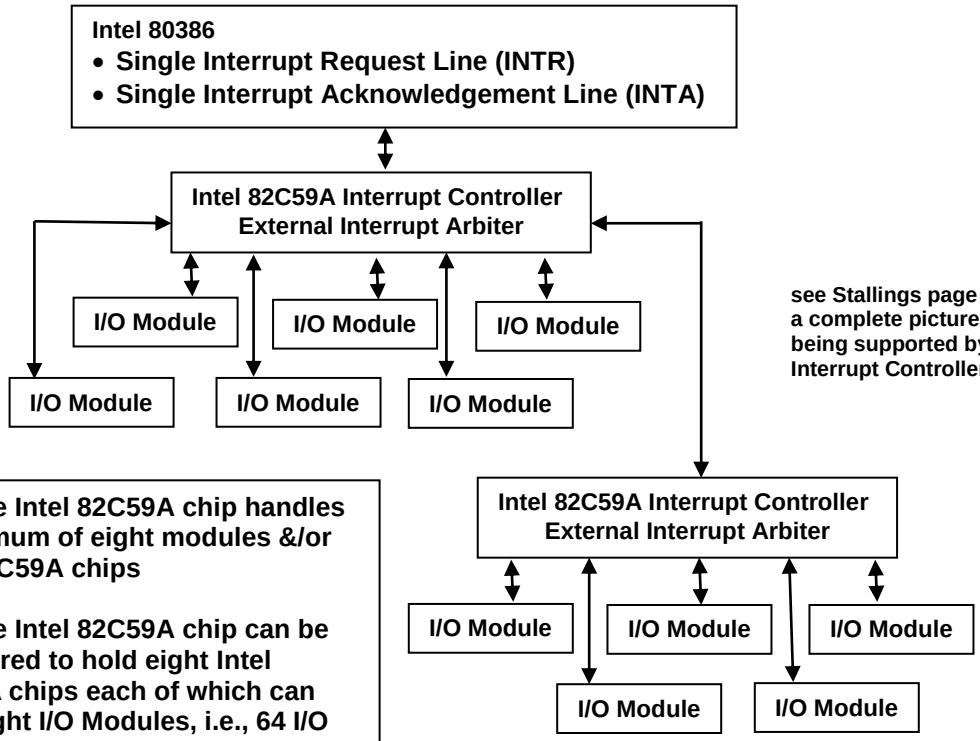
- **Multiple interrupt processing priorities are determined by the daisy chain order**

- **Bus Arbitration**



- **I/O Module must gain control of the bus before it can raise an interrupt, i.e., only one Module can be on the line at a time**
- **When processor detects an interrupt, it responds on the interrupt acknowledgement line**
- **The requesting I/O Module places its vector on the data lines**
- **Multiple interrupt priorities can be determined by a priority scheme for gaining control of the bus**

Intel 82C59A Interrupt Controller



see Stallings page 233, figure 7.8 for a complete picture of 64 devices being supported by nine 82C59A Interrupt Controllers

A single Intel 82C59A chip handles a maximum of eight modules &/or Intel 82C59A chips

A single Intel 82C59A chip can be configured to hold eight Intel 82C59A chips each of which can hold eight I/O Modules, i.e., 64 I/O Modules

Intel 82C59A
- accepts interrupts from the attached modules,
- determines which interrupt has the highest priority and
- signals the processor by raising the INTR line;

Processor
- acknowledges via the INTA line

Intel 82C59A
- places the appropriate vector information on the data bus

Processor
- precedes to process the interrupt and
- interact directly with the I/O Module to read the data

Intel 82C59A is programmable; i.e., Intel 80386 determines the priority scheme by setting a control word in the 82C59A

Interrupt Modes
- Fully Nested (Linear)
  - Interrupt requests are ordered in priority from 0 (IR0) to 7 (IR7)
- Rotating
  - Device, after being serviced for an interrupt, receives the lowest priority of all the other devices
- Special Mask
  - Allows the processor to inhibit interrupts from certain devices

Intel 82C55A Programmable Peripheral Interface

< see Stallings page 234 >

Drawbacks of Programmed and interrupt-Driven I/O

1. I/O transfer rate is limited by processors ability to test & service the device
2. Processor is required to manage the I/O transfer where multiple instructions must be executed for each I/O transfer

Results in an adverse impact on both processor activity and I/O transfer rate

Direct Memory Access

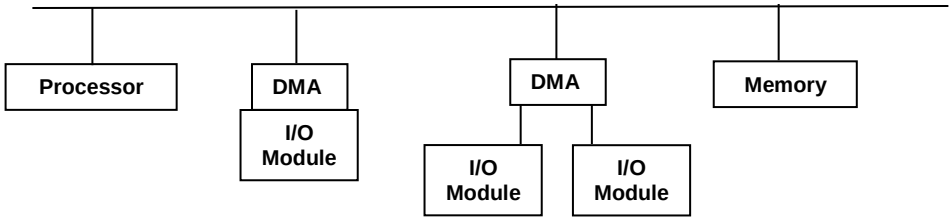
- **DMA Module**
  - **subservient processor**
    - **under certain circumstances, takes control of the system**
      - ⇨ **use the bus only when processor does not need it**
      - **or**
      - ⇨ **force the processor to suspend activity temporarily, i.e., cycle stealing**

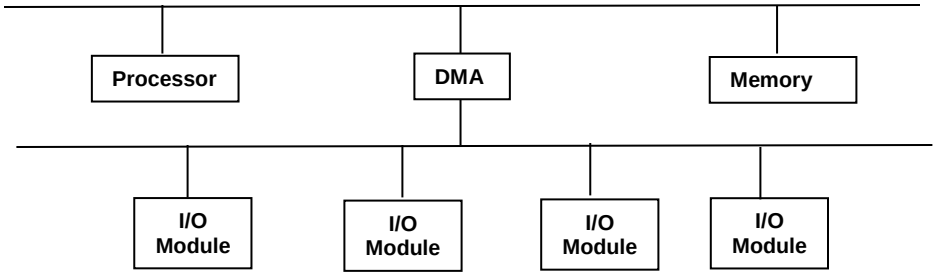
  - **operational activities**
    - **Processor issues a command to the DMA**
      - ⇨ **issues a read or write command on the control line**
      - ⇨ **address of I/O device is communicated on the data lines**
      - ⇨ **starting address in memory is transferred via data lines and stored in the DMA's address register**
      - ⇨ **number of words to be read/written is transferred via data lines and stored in the DMA's data count register**

    - **DMA module**
      - ⇨ **transfers entire block of data between selected device and memory without involving the processor**
      - ⇨ **when transfer is complete, sends an interrupt signal to the processor**

      Notice: the processor may be forced to pause for one bus cycle while the DMA accesses memory; hence the processor executes more slowly due to cycle stealing!

    - **DMA configurations**





Intel 8237A DMA Controller

- **Configuration <see Stallings page 239>**
- **Operational Activities**
  - **DMA sends an HRQ, i.e., a HOLD signal to Processor**
  - **Processor responds with an HLDA, i.e., a hold acknowledgement, signal**

DMA Active & Using the Bus←→ Processor is Idle
Processor is Active & Using the Bus ←→ DMA is Idle

Intel 8237A is a fly-by DMA Controller

- **Data being move**
  - **does not pass thru the DMA and**
  - **is not stored in the DMA**
- **DMA can only transfer data between**
  - **an I/O Port and**
  - **Memory**
- **DMA cannot transfer data between**
  - **two I/O Ports nor between**
  - **two memory locations**
- **DMA can effect a transfer of data between two memory locations by using a register**

- **8237 Architecture**
  - **each chip contains four independent DMA Channels**
  - **each chip has five control/command registers to control DMA operations**
    - ⇨ **Command**
      - **D0 enables memory-to-memory transfer**
        - ✓ **channel 0 transfers a byte to a temporary register**
        - ✓ **channel 1 transfers the byte from the temporary register to memory**
        - ✓ **D1 disables increment/decrement on channel 0 → write fixed value to a block of memory**
      - **Enable/Disable the DMA chip**
    - ⇨ **Status**
      - **used to inform the processor of DMA status**
    - ⇨ **Mode**
      - **Processor sets the Mode Register to determine the DMA's mode of operation**
    - ⇨ **Single Mask**
      - **Programmer can enable/disable individual channels**
    - ⇨ **All Mask**
      - **Programmer can enable/disable all four channels with one command**

Evolution of I/O Functions

1. **CPU directly controls all aspects of the I/O operation**
2. **CPU uses an I/O module to allow**
   a. **programmed I/O without interrupts**
   b. **programmed I/O with interrupts**
3. **DMA controls a block of data with minimal involvement of the CPU**
4. **I/O Channel**
   - **I/O Module becomes a processor**
   - **executes an I/O program residing in memory**
   - **CPU executes a sequence of activities; interrupted only when sequence is completed**
5. **I/O Processor**
   - **I/O Module becomes a computer with local memory**
   - **used to control communication with interactive terminals**