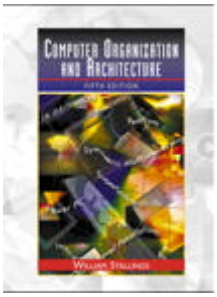
Campus de Gualtar
4710-057 Braga

UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA

Departamento de
Informática



Computer Organization and Architecture

5th Edition, 2000

by William Stallings

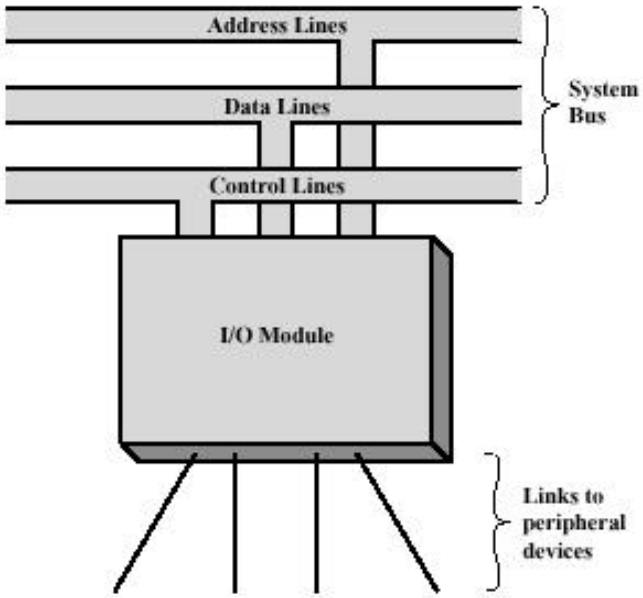
Table of Contents

II. THE COMPUTER SYSTEM.

3. ...
4. ...
5. ...
6. Input/Output. (23-Mar-98)

Introduction

- Why not connect peripherals directly to system bus?
    - Wide variety w/ various operating methods
    - Data transfer rate of peripherals is often much slower than memory or CPU
    - Different data formats and word lengths than used by computer
- Major functions of an I/O module
    - Interface to CPU and memory via system bus or central switch
    - Interface to one or more peripheral devices by tailored data links



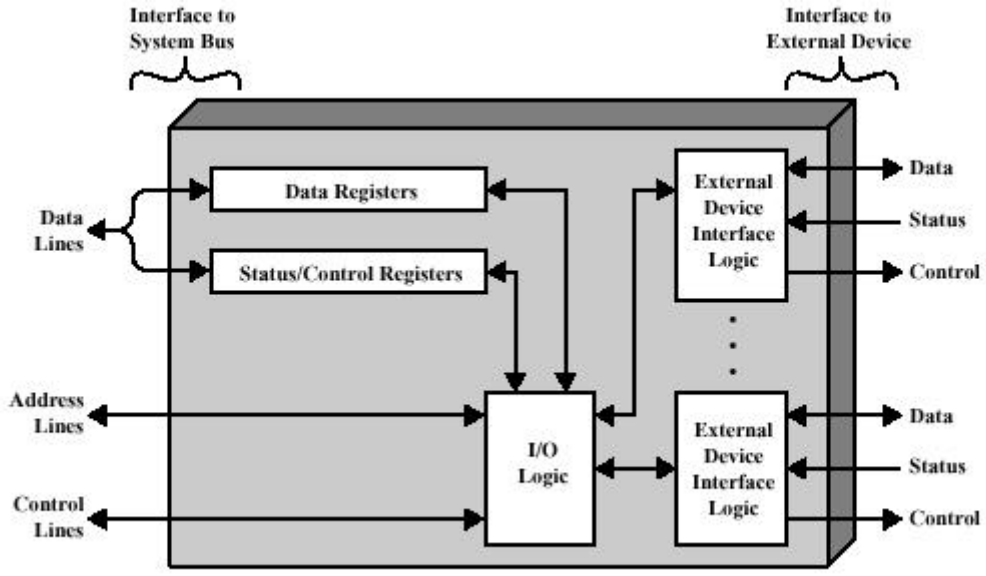
External Devices (6.1)

- External devices, often called peripheral devices or just peripherals, make computer systems useful.
- Three broad categories of external devices:
    - Human-Readable (ex. terminals, printers)
    - Machine-Readable (ex. disks, sensors)
    - Communication (ex. modems, NIC's)
- Basic structure of an external device:
    - Data - bits sent to or received from the I/O module
    - Control signals - determine the function that the device will perform
    - Status signals - indicate the state of the device (esp. READY/NOT-READY)
    - Control logic - interprets commands from the I/O module to operate the device
    - Transducer - converts data from computer-suitable electrical signals to the form of energy used by the external device
    - Buffer - temporarily holds data being transferred between I/O module and the external device

I/O Modules (6.2)

- An I/O Module is the entity within a computer responsible for:
  - control of one or more external devices
  - exchange of data between those devices and main memory and/or CPU registers
- It must have two interfaces:
  - internal, to CPU and main memory
  - external, to the device(s)
- Major function/requirement categories
  - Control and Timing
    - Coordinates the flow of traffic between internal resources and external devices
    - Cooperation with bus arbitration
  - CPU Communication
    - Command Decoding
    - Data
    - Status Reporting
    - Address Recognition.
  - Device Communication (see diagram under External Devices)
    - Commands
    - Status Information
    - Data
  - Data Buffering
    - Rate of data transfer to/from CPU is orders of magnitude faster than to/from external devices
    - I/O module buffers data so that peripheral can send/receive at its rate, and CPU can send/receive at its rate
  - Error Detection
    - Must detect and correct or report errors that occur
    - Types of errors
      - Mechanical/electrical malfunctions
      - Data errors during transmission

- I/O Module Structure
  - Basic Structure



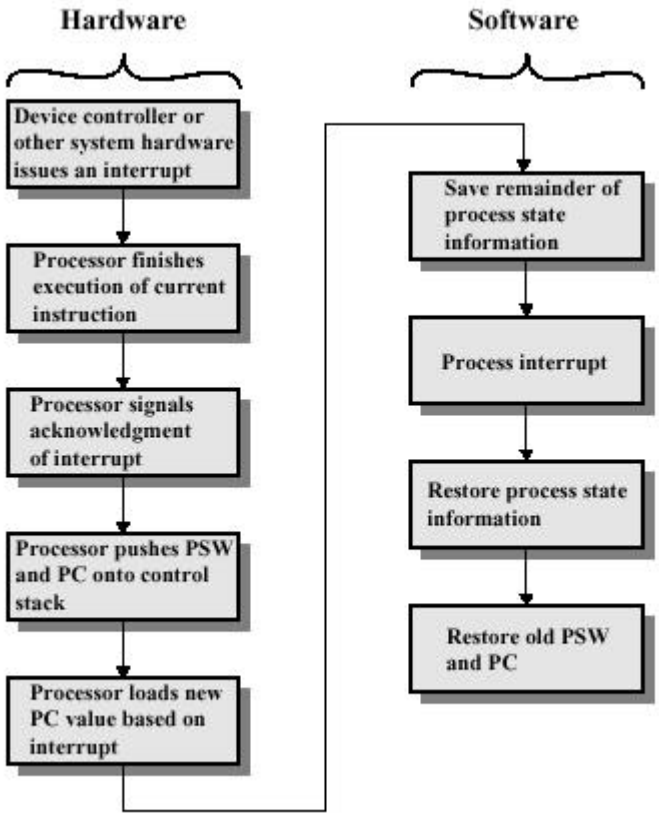
- An I/O module functions to allow the CPU to view a wide range of devices in a simple-minded way.
- A spectrum of capabilities may be provided
  - I/O channel or I/O processor - takes on most of the detailed processing burden, presenting a high-level interface to CPU
  - I/O controller or device controller - quite primitive and requires detailed control
  - I/O module - generic, used when no confusion results

Programmed I/O (6.3)

- With programmed I/O, data is exchanged under complete control of the CPU
  - CPU encounters an I/O instruction
  - CPU issues a command to appropriate I/O module
  - I/O module performs requested action and sets I/O status register bits
  - CPU must wait, and periodically check I/O module status until it finds that the operation is complete
- To execute an I/O instruction, the CPU issues:
  - an address, specifying I/O module and external device
  - a command, 4 types:
    - control - activate a peripheral and tell it what to do
    - test - querying the state of the module or one of its external devices
    - read - obtain an item of data from the peripheral and place it in an internal buffer (data register from preceding illustration)
    - write - take an item of data from the data bus and transmit it to the peripheral
- With programmed I/O, there is a close correspondence between the I/O instructions used by the CPU and the I/O commands issued to an I/O module
- Each I/O module must interpret the address lines to determine if a command is for itself.
- Two modes of addressing are possible:
  - Memory-mapped I/O
    - there is a single address space for memory locations and I/O devices.
    - allows the same read/write lines to be used for both memory and I/O transactions
  - Isolated I/O
    - full address space may be used for either memory locations or I/O devices.
    - requires an additional control line to distinguish memory transactions from I/O transactions
    - programmer loses repertoire of memory access commands, but gains memory address space

Interrupt-Driven I/O (6.4)

- Problem with programmed I/O is CPU has to wait for I/O module to be ready for either reception or transmission of data, taking time to query status at regular intervals.
- Interrupt-driven I/O is an alternative
  - It allows the CPU to go back to doing useful work after issuing an I/O command.
  - When the command is completed, the I/O module will signal the CPU that it is ready with an interrupt.
- Simple Interrupt Processing Diagram

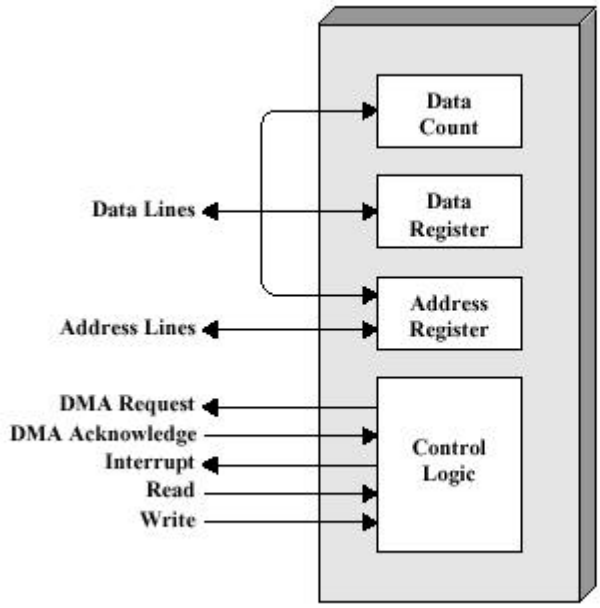


- Design issues
    - How does the CPU determine which device issued the interrupt?
        - Multiple Interrupt Lines
            - most straightforward solution
            - impractical to dedicate many lines
                - multiple I/O modules are likely attached to each line
                - one of other 3 techniques must be used on each line
        - Software Poll
            - interrupt service routine polls each device to see which caused the interrupt
                - using a separate command line on the system bus (TESTI/O) ? raise TESTI/O ? place I/O module address on address lines ? check for affirmative response
                - each I/O module contains an addressable status register, which CPU reads
            - time consuming
        - Daisy Chain (hardware poll, vectored)
            - interrupt occurs on interrupt request line which is shared by all I/O modules
            - CPU senses interrupt
            - CPU sends interrupt acknowledge, which is daisy-chained through all I/O modules
            - When it gets to requesting module, it places its vector (either an I/O address, or an ID which the CPU uses as a pointer to the appropriate device-service routine) on the data lines
            - No general interrupt-service routine needed (still need specific ones)
        - Bus Arbitration (vectored)

        - requires an I/O module to first gain control of the bus before it can interrupt
        - thus only one module can interrupt at a time
        - when CPU detects the interrupt, it ACK's
        - requesting module then places its vector on the data lines
        - another type of vectored interrupt
- If multiple interrupts have occurred, how does the CPU decide which one to process?
    - Multiple lines - assign priorities to lines, and pick the one with highest priority
    - Software polling - order in which modules are polled determines priority
    - Daisy chain - order of modules on chain determines priority
    - Bus arbitration can employ a priority scheme through the arbiter or arbitration algorithm

Direct Memory Access (6.5)

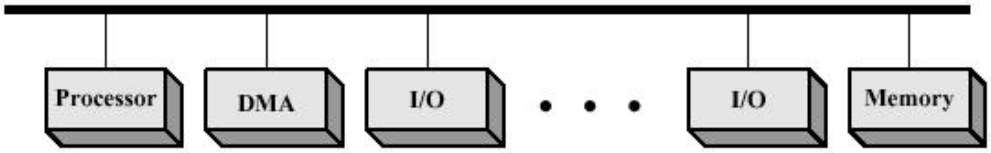
- Drawbacks of Programmed and Interrupt-Driven I/O
    - The I/O transfer rate is limited by the speed with which the CPU can test and service a device
    - The CPU is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer
- DMA Function



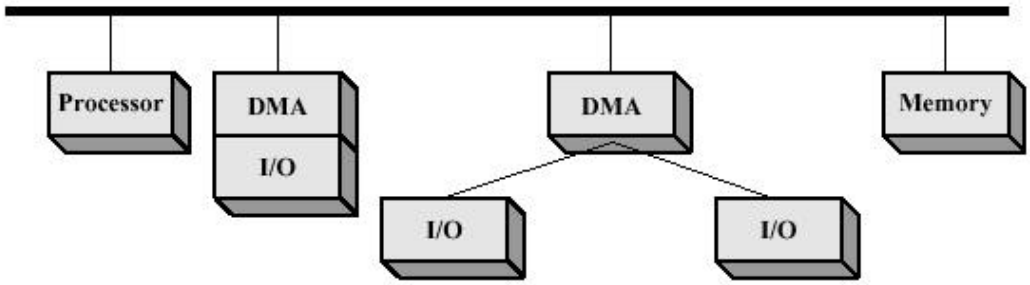
    - Involves adding a DMA module to the system bus
        - can take over system from CPU
        - can mimic certain CPU operations
    - When CPU wishes to read or write a block of data it issues a command to the DMA module containing:
        - Whether a read or write is requested
        - The address of the I/O device involved
        - The starting location in memory to read from or write to
        - The number of words to be read or written
    - CPU continues with other work

- DMA module handles entire operation. When memory has been modified as ordered, it interrupts the CPU
- CPU is only involved at beginning and end of the transfer
- DMA module can force CPU to suspend operation while it transfers a word
  - called cycle stealing
  - not an interrupt, just a wait state
  - slows operation of CPU, but not as badly as non-DMA

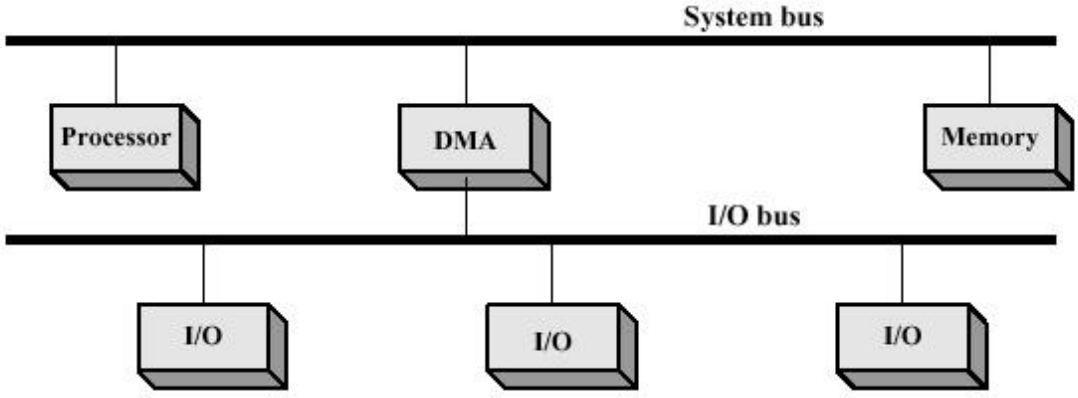
- Possible DMA Configurations
  - Single Bus, Detached DMA



    - DMA module uses programmed I/O as a surrogate CPU
    - Each transfer of a word consumes 2 bus cycles
  - Single-Bus, Integrated DMA-I/O



    - DMA module is attached to one or more I/O modules
    - Data and control instructions can move between I/O module and DMA module without involving system bus
    - DMA module must still steal cycles for transfer to memory
  - I/O Bus



    - Reduces number of I/O interfaces in the DMA module to one
    - Easily expanded
    - DMA module must still steal cycles for transfer to memory

I/O Channels and Processors (6.6)

- Past DMA, we see two evolutionary developments to the I/O module
    - The I/O Channel enhances the I/O module to become a processor in its own right
        - CPU directs I/O module to execute a sequence of special I/O instructions in memory
        - I/O channel fetches and executes these instructions without CPU intervention
        - The CPU is only interrupted when the entire sequence is completed
    - The I/O Processor has a local memory of its own, and is a computer in its own right, allowing a large set of devices to be controlled with minimal CPU involvement
- Two common types of I/O channels
    - A selector channel
        - controls multiple high-speed devices
        - is dedicated to transfer of data with one device at a time
        - each device is handled by a controller that is very similar to an I/O module
        - the I/O channel serves in place of the CPU in controlling these I/O controllers
    - A multiplexor channel
        - Handles I/O with multiple controllers at once
        - Low speed devices use a byte multiplexor
        - High speed devices use a block multiplexor

The External Interface (6.7)

- Types of interfaces
    - Parallel interface - multiple lines connect the I/O module and the peripheral, and multiple bits are transferred simultaneously
    - Serial interface - only one line is used to transmit data, one bit at a time
- Typical dialog between I/O module and peripheral (for a write operation)
    - The IO module sends a control signal requesting permission to send data
    - The peripheral ACK's the request
    - The I/O module transfers the data
    - The peripheral ACK's receiving the data
- Buffer in I/O module compensates for speed differences
- Point-to-Point and Multipoint Configurations
    - Point-to-point interface
        - provides a dedicated line between the I/O module and the external device
        - examples are keyboard, printer, modem
    - Multipoint interface
        - in effect are external buses
        - examples are SCSI, USB, IEEE 1394 (FireWire), and even IDE
- SCSI (Small Computer System Interface)
    - Standards
        - SCSI-1 (early '80's)
            - 8 data lines
            - data rate of 5Mb/s
            - up to 7 devices daisy-chained to interface

        - SCSI-2 (1991)
            - usually what is meant today by “SCSI”
            - 16 or 32 data lines
            - data rate of 20Mb/s or 40Mb/s (dep. on data lines)
            - up to 16 devices daisy-chained to interface
            - includes commands for the following device types:
                - direct-access devices
                - sequential-access devices
                - printers
                - processors
                - write-once devices
                - CD-ROM’s
                - scanners
                - optical memory devices
                - medium-changer devices
                - communication devices
        - SCSI-3 (still in development)
            - doubles speed of SCSI-2
            - includes “serial SCSI”, which is basically FireWire
    - Basically defines a high-speed external bus
        - has its own arbitration
        - communication can take place between devices (such as disk to tape) without involving the CPU or other internal buses at all
        - reselection - if a command is issued that takes some time to complete, the target can release the bus and reconnect to the initiator later

- P1394 Serial Bus (FireWire)
    - Basic features
        - very high-speed - up to 400Mbps
        - serial transmission
            - requires less wires
            - allows simple connector
            - less potential for damage
            - requires less shielding w/ no synchronization problems
            - cheaper
        - physically small - suitable for handheld computers and other small consumer electronics
    - Details
        - Uses a daisy chain configuration
            - up to 63 devices off a single port
            - up to 1022 P1394 buses can be interconnected using bridges
        - Supports hot plugging - peripherals can be connected and disconnected without power down or reconfiguration
        - Supports automatic configuration
            - no manual configuration of device ID’s required
            - relative positioning of devices unimportant
        - Strict daisy-chain not required - trees possible
        - Basically sets up a bridged bus-type network as the I/O bus
            - data and control information (such as for arbitration) is transferred in packets
            - can function asynchronously
                - using variable amounts of data and larger packet headers w/ ACK needed
                - for devices with intermittent need for the bus
                - can use fair or urgent arbitration
            - can function isochronously
                - using fixed-size packets transmitted at regular intervals
                - for devices that regularly transmit or consume data, such as digital sound or video