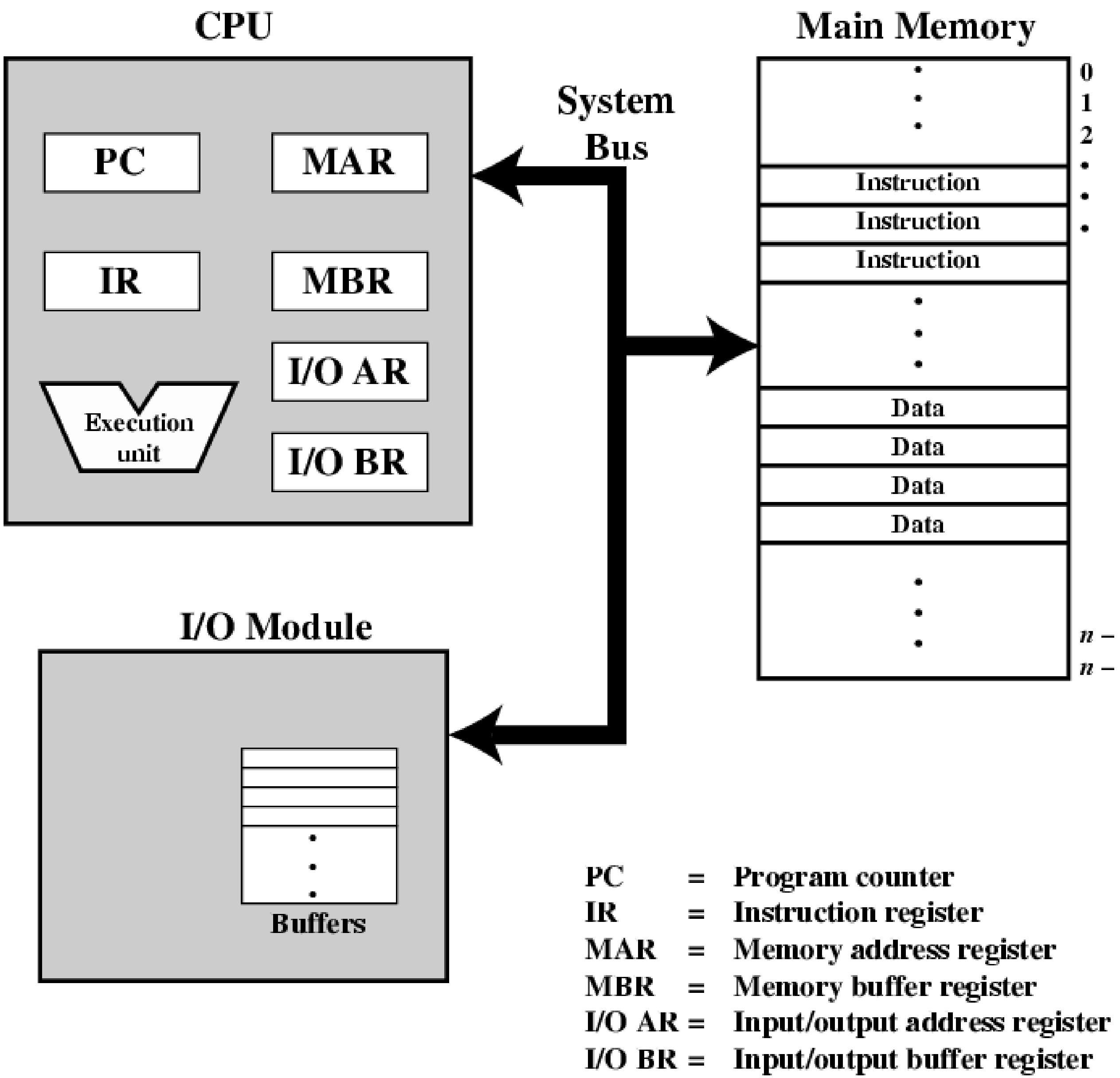
Chapter #7
Input/Output

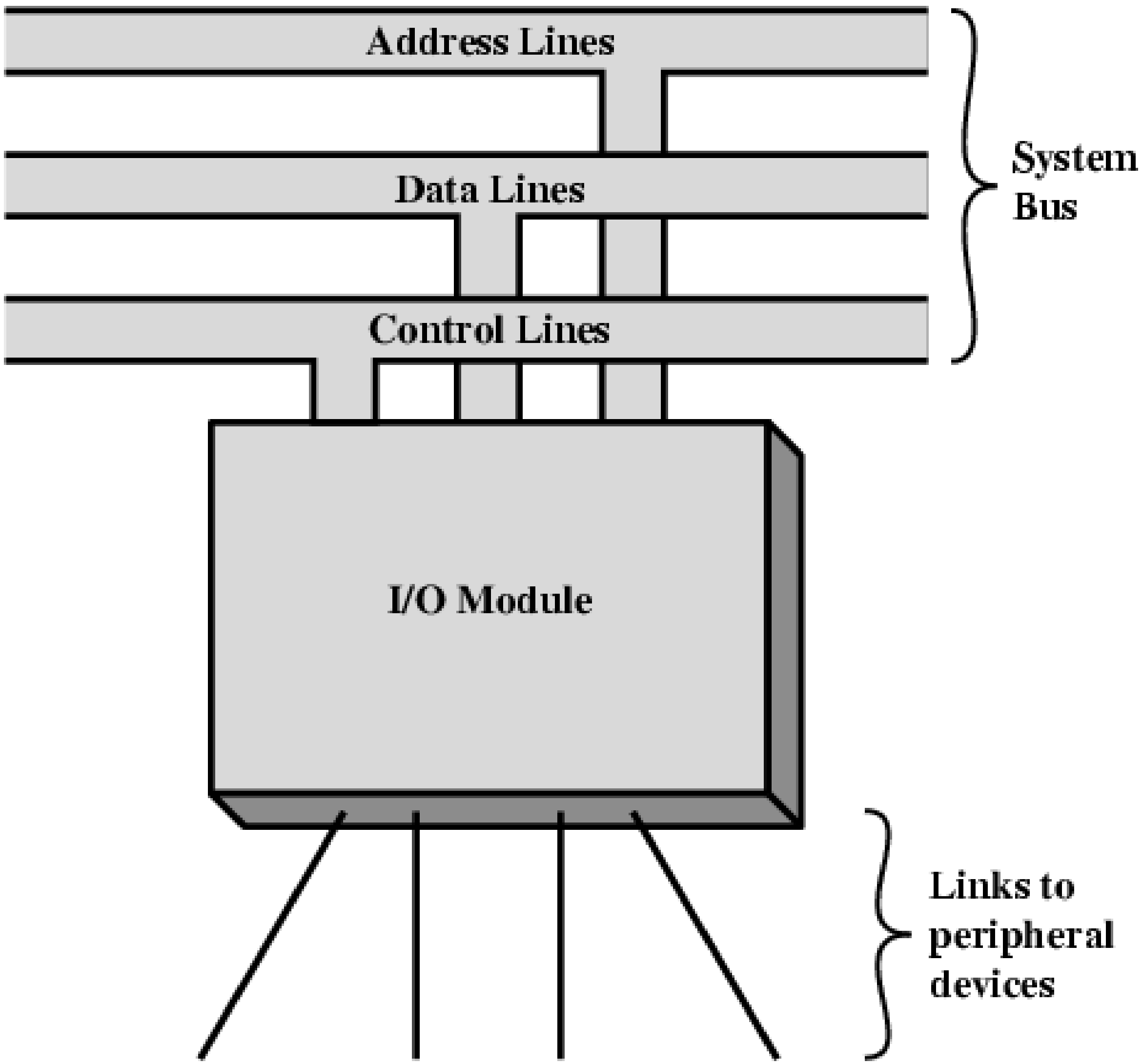
Input/Output Issues

- Managing wide variety of peripherals w/
  - —varying data transfer amounts
  - —varying data transfer speeds
  - —varying data transfer formats & word lengths
- I/O is significantly slower than CPU & Memory
- Need I/O modules for interfacing
  - —to CPU and Memory via system bus
  - —among multiple I/O peripherals

Components: Top Level View



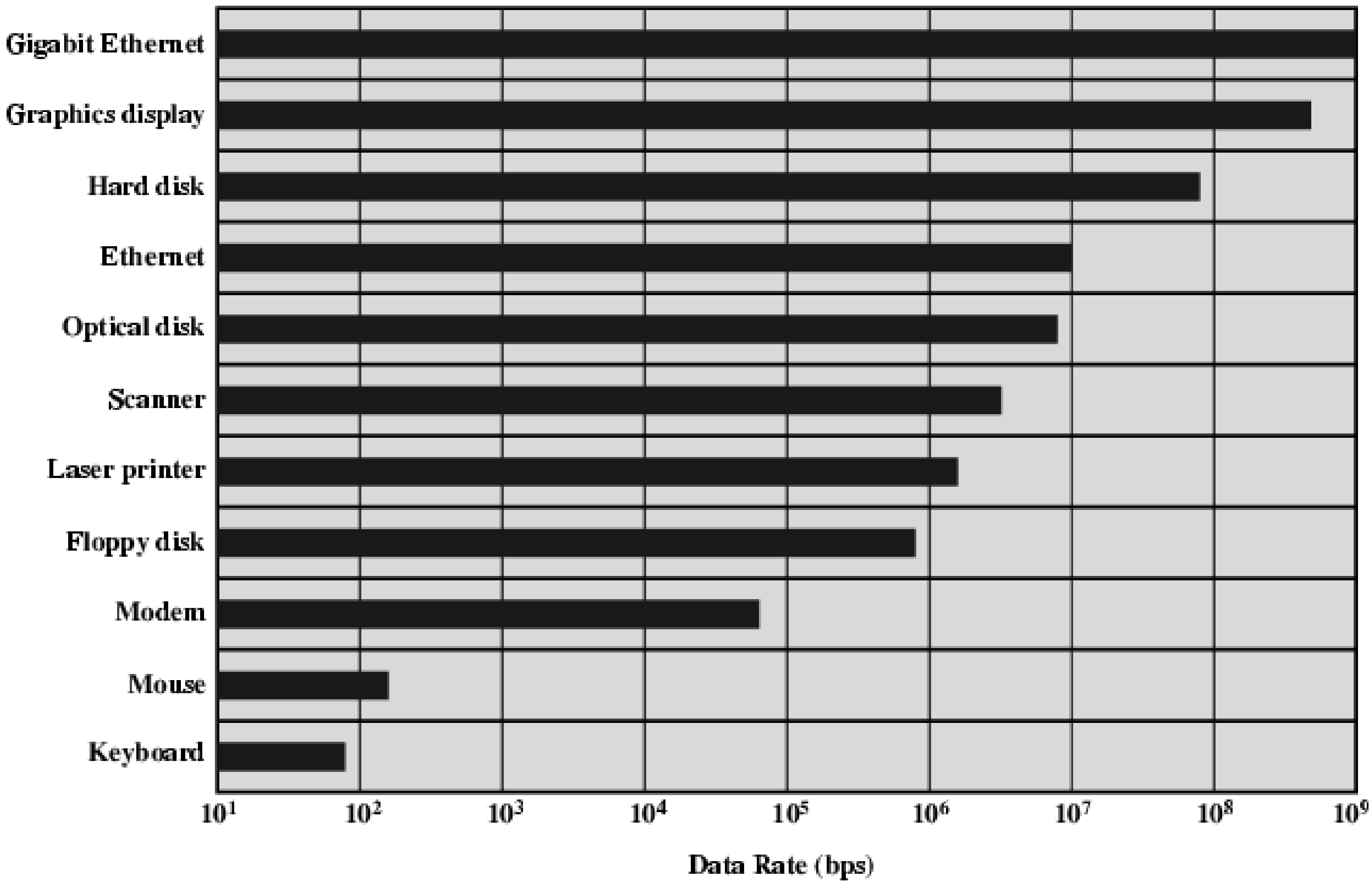
Generic Model of I/O Module



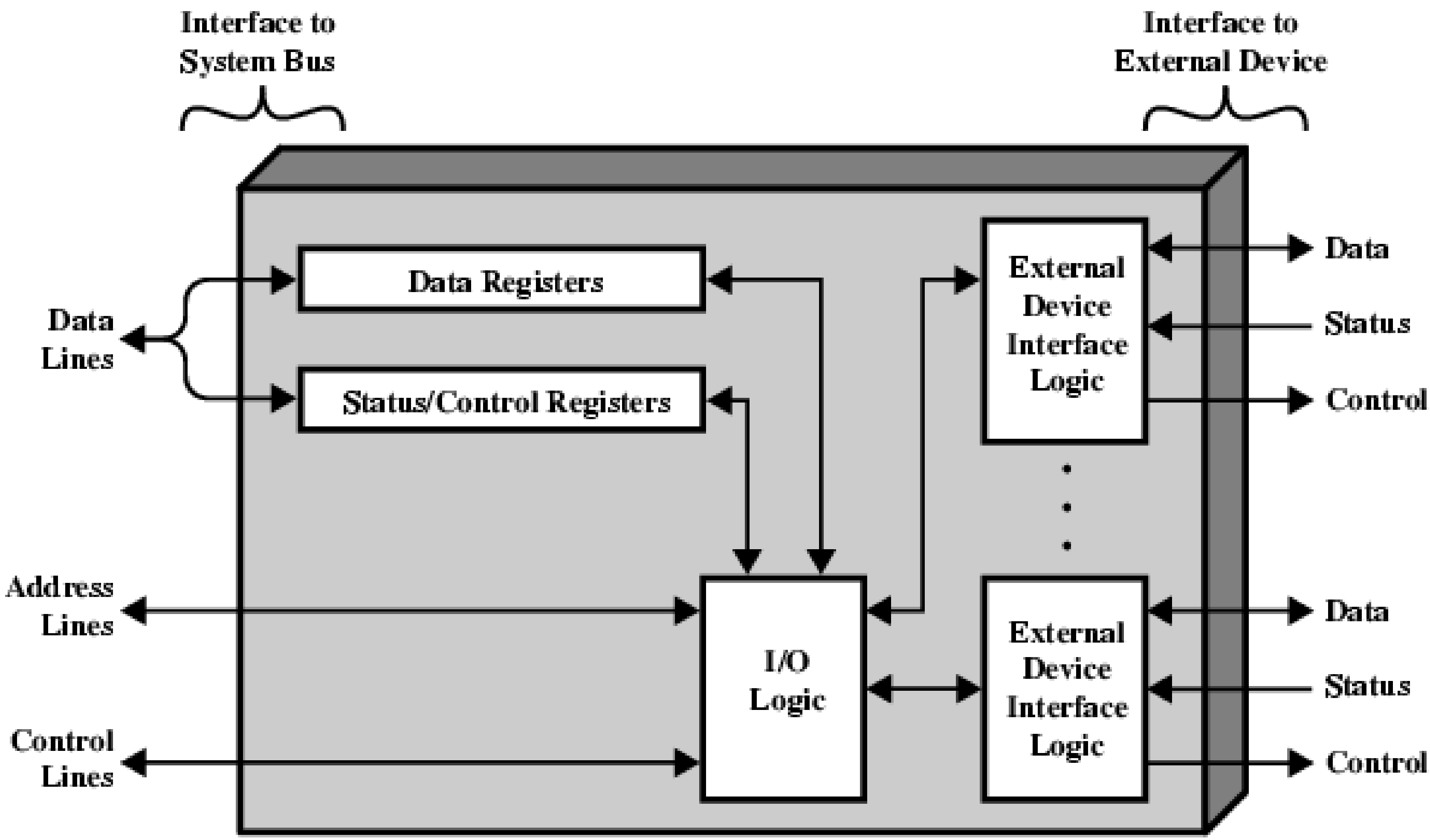
Peripheral Devices

- Human-oriented
  - Monitor
  - Printer
  - Keyboard
  - Mouse
  - Scanner
  - Graphics display
- Machine-oriented
  - Floppy disk
  - Hard disk
  - Optical disk (CD-rom, DVD-rom)
- Communication-oriented
  - Modem
  - Ethernet card

Typical I/O Data Rates



I/O Module Diagram



Input/Output Control 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)

Method #1--Programmed I/O

- CPU has direct control over I/O
  - —Checking status
  - —Issuing read/write commands
  - —Transferring data
- Adv:
  - —Direct and simple
- Disadv:
  - —CPU waits for I/O module to complete operation
  - —Not efficient→wastes CPU time

Programmed I/O - details

- Steps:
  - —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 Commands

- CPU issues address
  - —Identifies unique module
  - —Identifies unique device if >1 per I/O module
- CPU issues command
  - —Control - telling module what to do
  - —Test - check status
  - —Read/Write
    - – Module transfers data via buffer from/to device

I/O Mapping

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

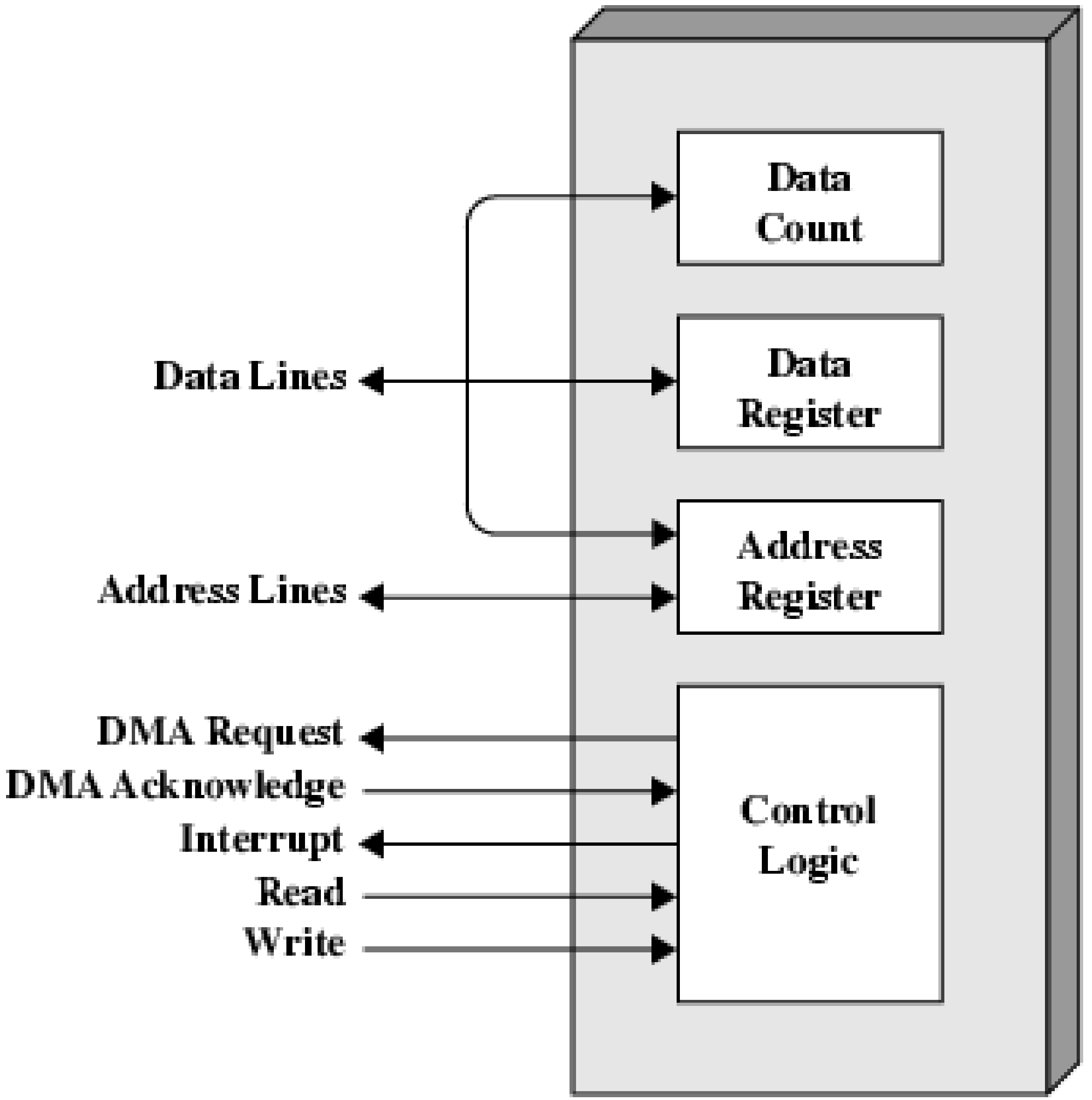
Method #2--Interrupt Driven I/O

- Characteristics:
  - —Overcomes CPU waiting
  - —No repeated CPU checking of device
  - —I/O module interrupts when ready
- Basic operation:
  - —CPU issues read command
  - —I/O module gets data from peripheral while CPU does other work
  - —I/O module interrupts CPU
  - —CPU requests data
  - —I/O module transfers data

Method #3--Direct Memory Access

- Motivation:
  - —Interrupt driven and programmed I/O require active CPU intervention
  - —Transfer rate is limited
  - —CPU is tied up
- Function:
  - —DMA controller takes over from CPU for I/O
  - —Interrupts CPU to report status

DMA Module Diagram



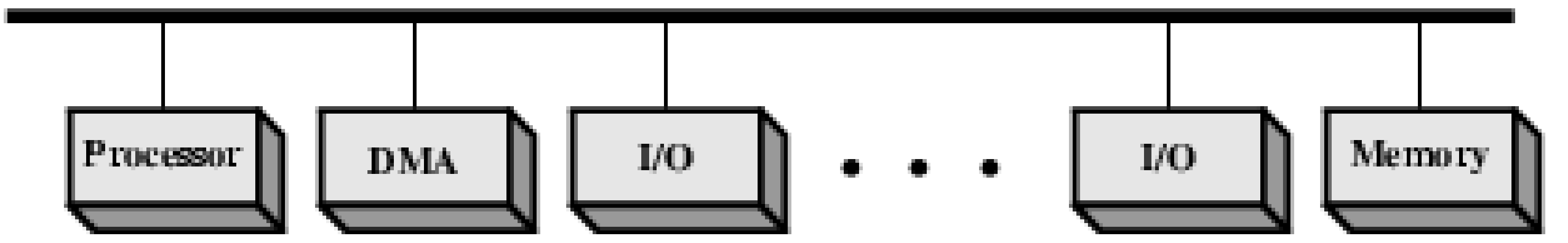
DMA Operation

- CPU tells DMA controller:
  - —Read/Write
  - —Device address
  - —Starting address of memory block for data
  - —Amount of data to be transferred
- CPU performs other tasks
- DMA controller deals with transfer
- DMA controller sends interrupt when finished

DMA Transfer: Cycle Stealing

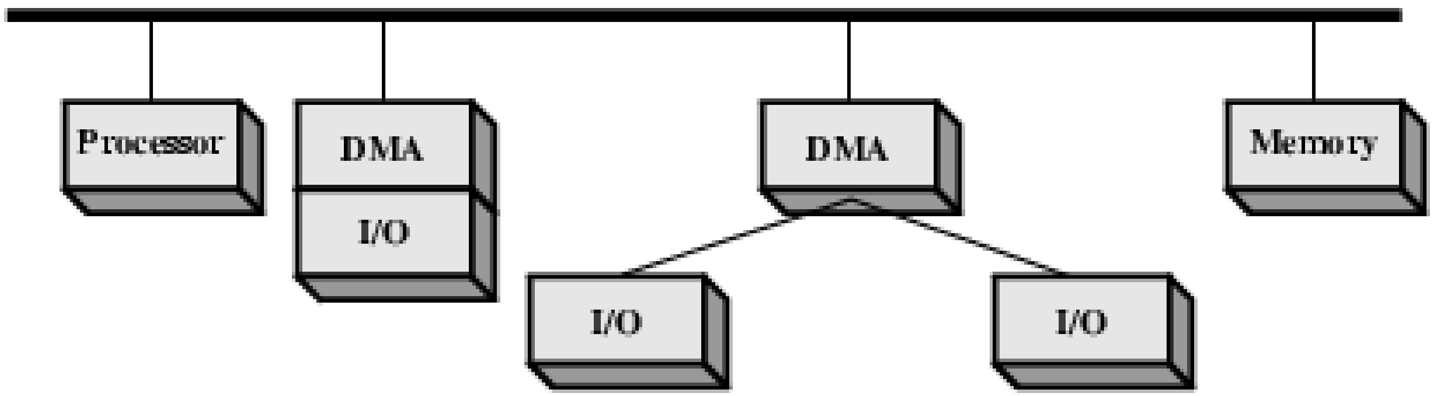
- DMA controller takes over bus for a cycle
- Transfer of one word of data
- Not an interrupt
  - —CPU does not switch context
- CPU suspended just before it accesses bus
  - —i.e. before an operand or data fetch or a data write
- Slows down CPU but not as much as CPU doing transfer

DMA Configurations (1)



- Single Bus, Detached DMA controller
- Each transfer uses bus twice
    - —I/O to DMA then DMA to memory
- CPU is suspended twice

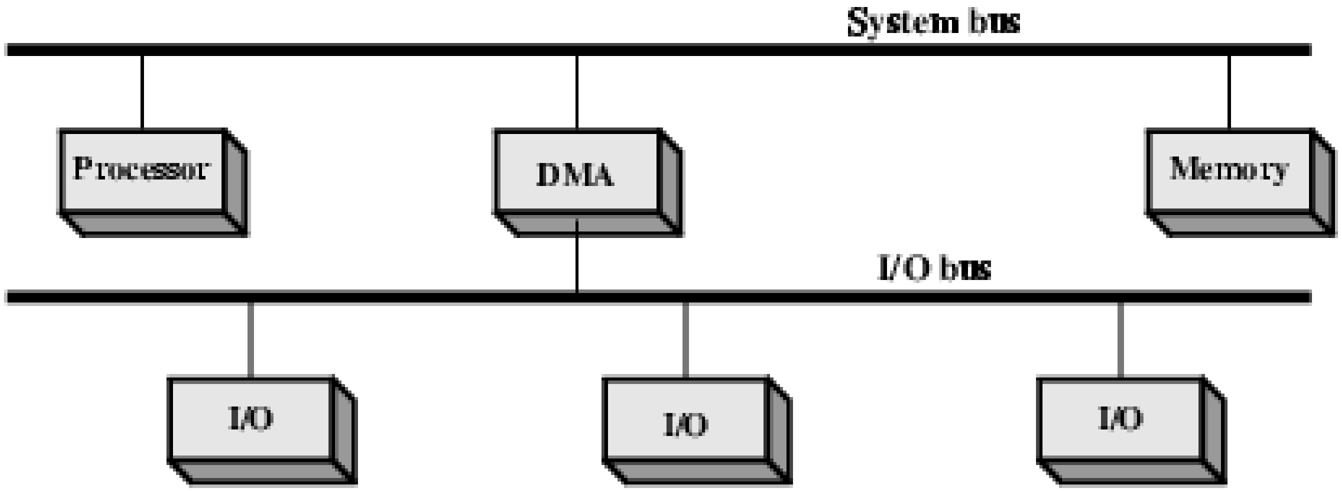
DMA Configurations (2)



(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