William Stallings
Computer Organization and Architecture
7th Edition

Chapter 7
Input/Output

5.0	**I/O Organization**		03	CO4
	5.1	External Devices, I/ O Modules		
	5.2	Programmed I/O, Interrupt driven I/O, DMA		

Input/Output Problems

- I/O module stands for Input/Output module, which is a device that acts as the **connective bridge between a computer system at one end and an I/O or peripheral device of some kind at the other, such as a printer, webcam or scanner.**
- I/O module contains logic for performing a communication function between the peripheral and the bus
- Peripherals not connected directly to bus because

  Wide variety of peripherals

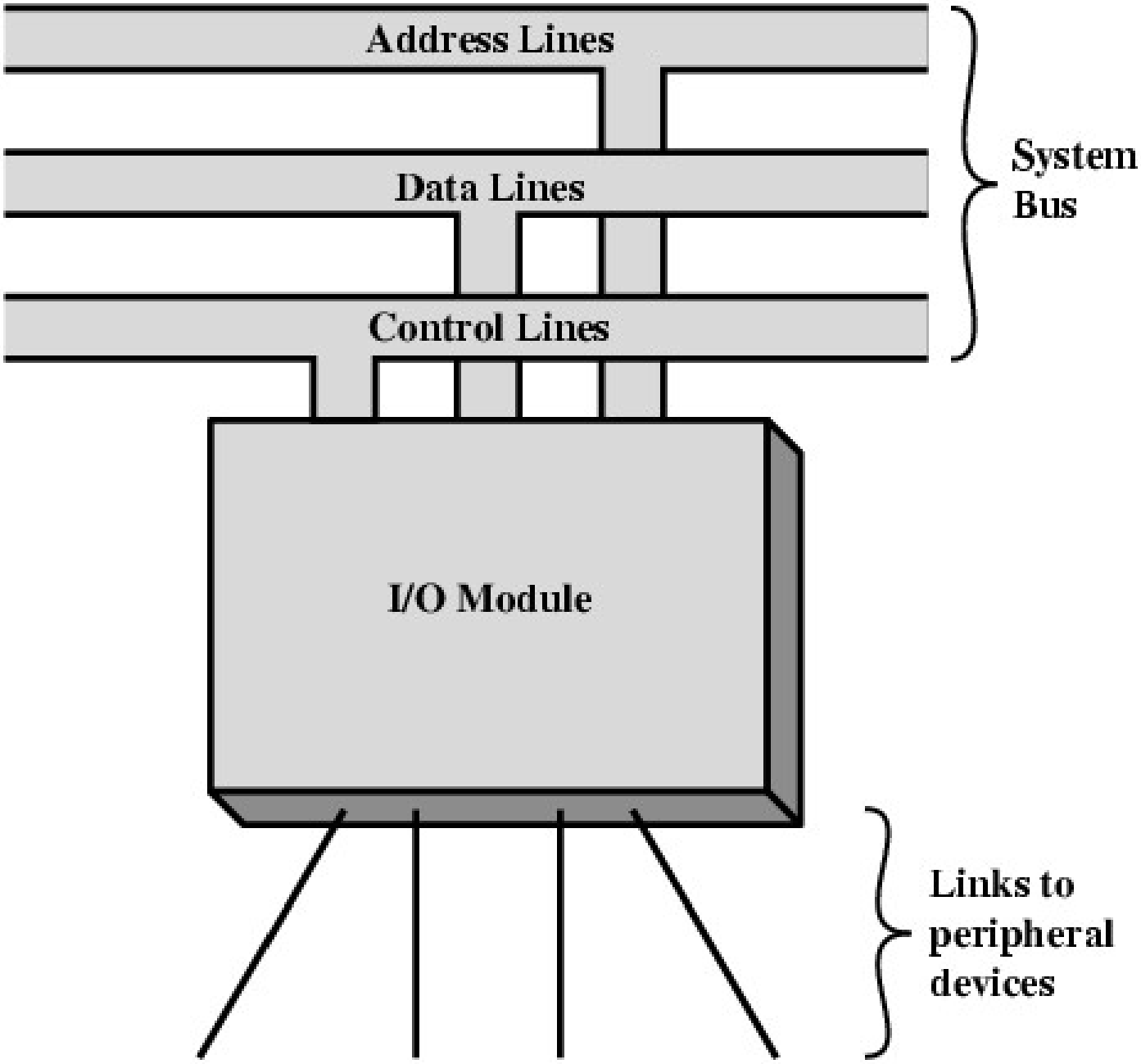
1. —Delivering different amounts of data
2. —At different speeds
3. —In different formats

- All slower than CPU and RAM
- Peripherals use different data formats and address length
- Need I/O modules

Main Functions of Input/Output Module

- Interface to CPU and Memory
- Interface to one or more peripherals

Generic Model of I/O Module



I/O Module Functions

- Control and timing-status reporting, commands and information transfer.
- Processor communication-decoding and accepting commands, reporting status updates and recognizing its own address.
- Device communication-status reporting, commands and information transfer.
- Data buffering-manage the transfer speed between the memory, processor and other peripheral devices that are connected.
- Error detection-data-based issues during transmission.

External Devices

- Human readable-human

  —Screen, printer, keyboard

- Machine readable-equipment

  —magnetic disk and tape systems- Monitoring and control

- Communication-remote devices

  —human-readable device, such as a terminal, a machine-readable device, or even another computer

  —Network Interface Card (NIC-connects to N/w)

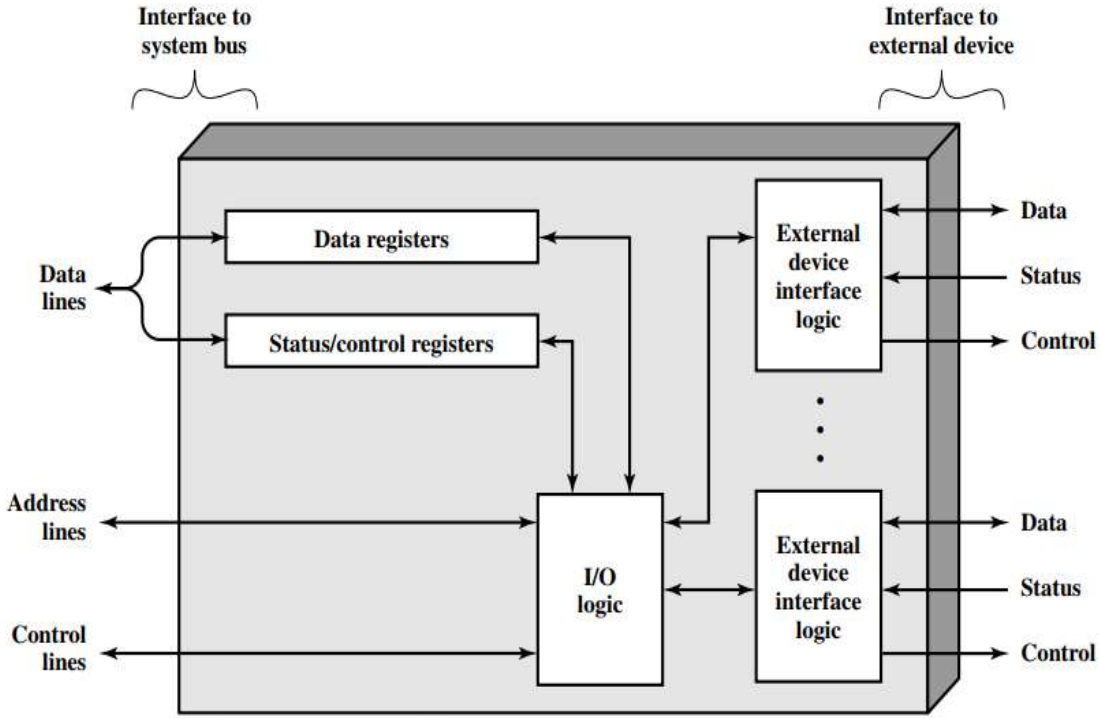
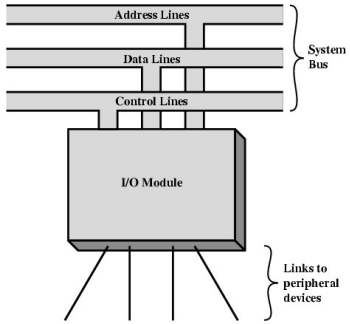


Figure 7.3 Block Diagram of an I/O Module

The control of the transfer of data from an **external device to the processor** might involve the following sequence of steps: I/O Steps

- CPU checks I/O module device status
- I/O module returns status
- If device is ready, CPU requests data transfer
- I/O module gets data from device
- I/O module transfers data to CPU
- Variations for output, DMA, etc.



Input Output Techniques

- Programmed
- Interrupt driven
- Direct Memory Access (DMA)

- **Programmed**-data are exchanged between the processor and the I/O module.
- The processor executes a program that gives it direct control of the I/O operation
- When the processor issues a command to the I/O module, it must wait until the I/O operation is complete

- With *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 latter has completed its work.

- **DMA**-I/O module and main memory exchange data directly, without processor involvement.

PROGRAMMED I/O

- Data transfer operations are completely controlled by **CPU** i.e. CPU executes a program that
  - initiates,
  - directs and
  - terminates the I/O operation

	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)

Overview of 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.
- With programmed I/O, the I/O module will perform the requested action and then set the appropriate bits in the I/O status register
- Processor periodically checks the status of the I/O module until it finds that the operation is complete
-

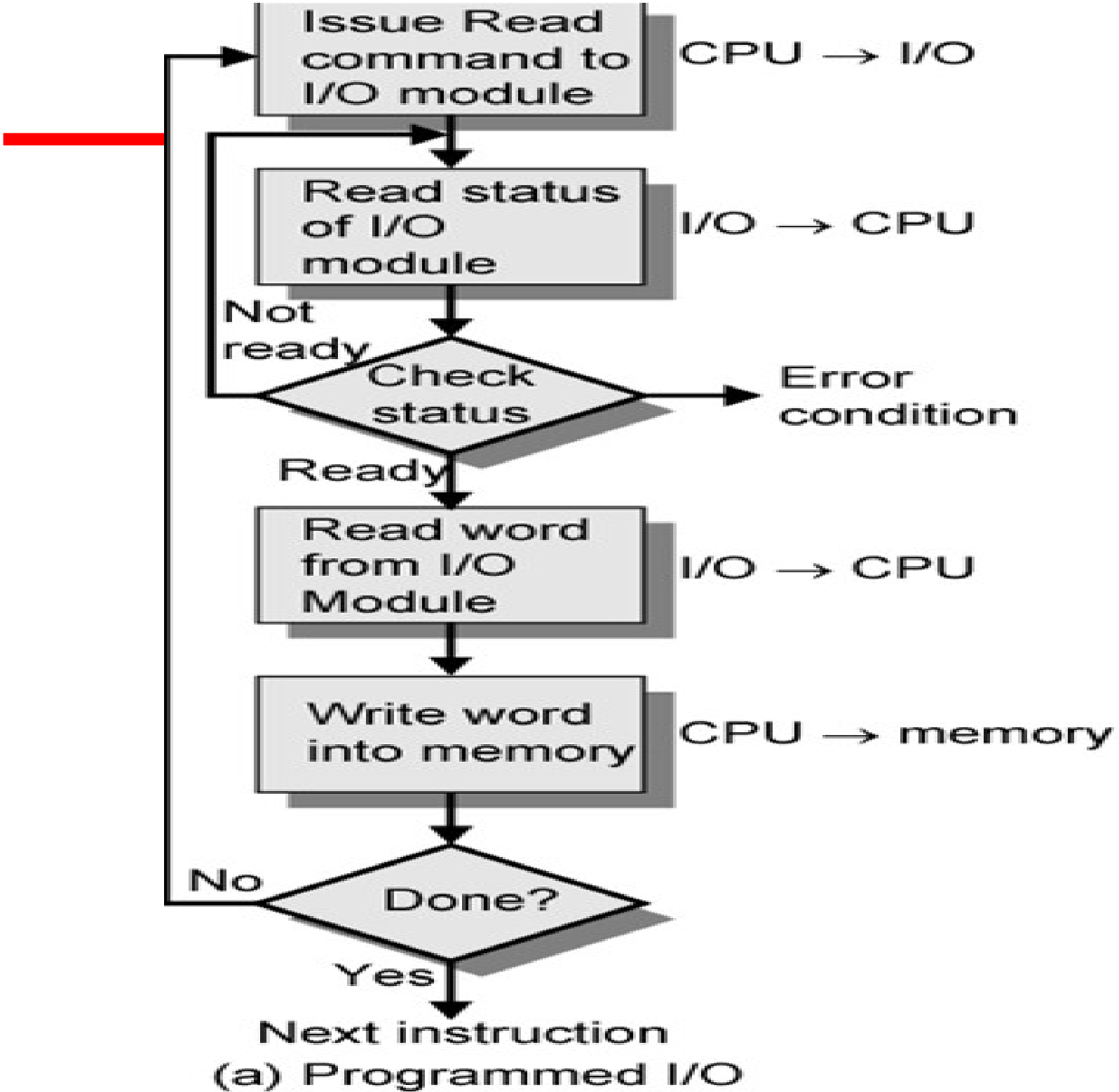
I/O Commands

- Processor issues an address, specifying the particular I/O module and external device, and an I/O command to execute an I/O-related instruction

- **Control:** Used to activate a peripheral and tell it what to do.
- **Test:** Used to test various status conditions associated with an I/O module and its peripherals.
- **Read:** Causes the I/O module to obtain an item of data from the peripheral and place it in an internal buffer
- **Write:** Causes the I/O module to take an item of data (byte or word) from the data bus and subsequently transmit that data item to the peripheral.

PROGRAMMED I/O

- Useful where h/w costs need to be minimized.
- Entire I/O is handled by CPU
- **STEPS**
  - 1. Read I/O devices status bit
  - 2. Test status bit to determine if device is ready
  - 3. If device not ready return to step 1
  - 4. During interval when device is not ready CPU simply wastes its time until device is ready



(a) Programmed I/O

- Figure-use of programmed I/O to read in a block of data from a peripheral device (e.g., a record from tape) into memory.
- Data are read in one word (e.g., 16 bits) at a time.
- For each word that is read in, the processor must remain in a status-checking cycle until it determines that the word is available in the I/O module's data register.
- When the processor issues an I/O command, the **command contains the address of the desired device.**
- Thus, each **I/O module** must **interpret the address lines to determine if the command is for itself**

- When the processor, main memory, and I/O share a common bus, two modes of addressing are possible: **memory mapped and isolated.**
- With memory-mapped I/O, there is a **single address space** for memory locations and I/O devices.
- the address space for I/O is isolated from that for memory, this is referred to as **isolated I/O**.

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

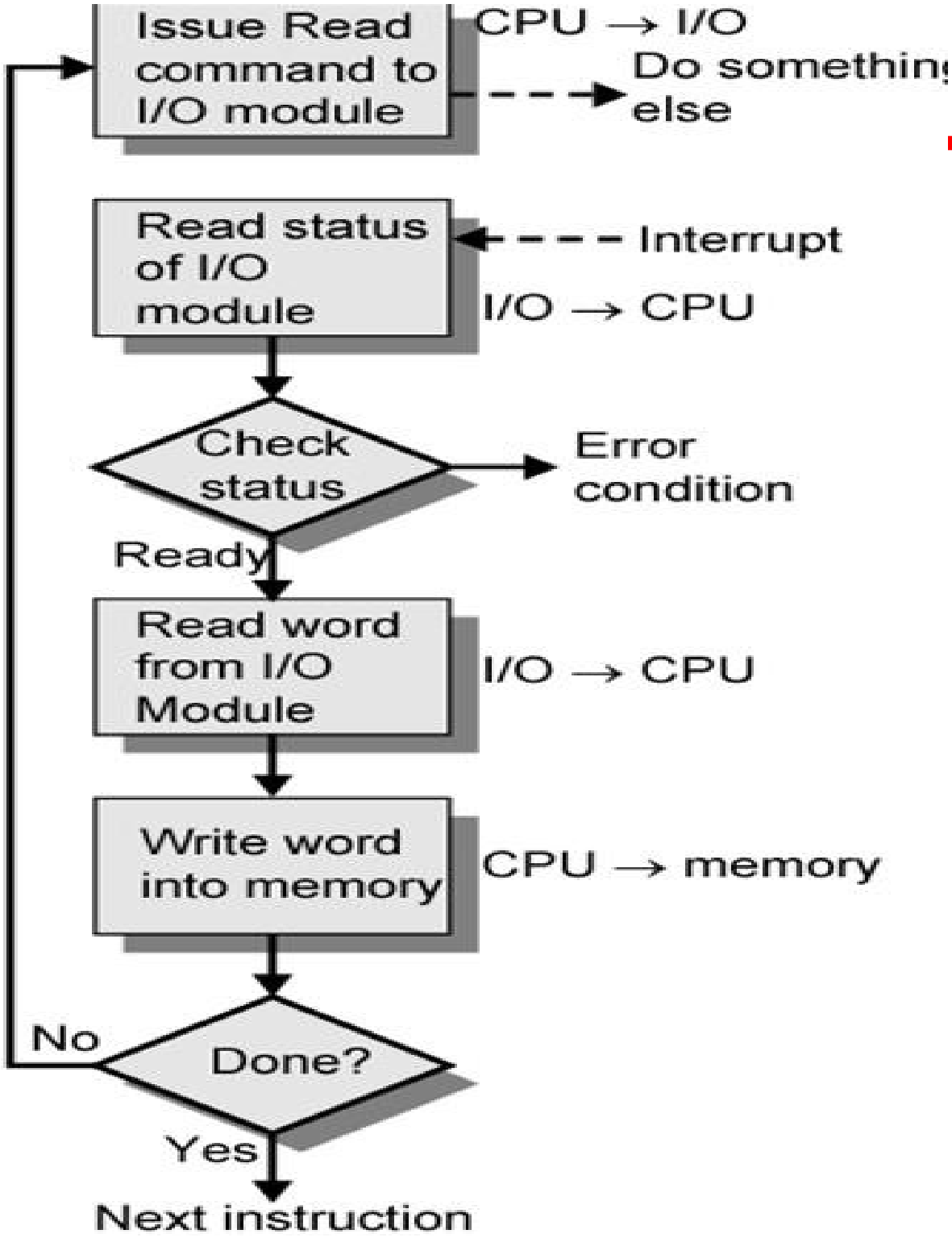
Programmed I/O - detail

- 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

INTERRUPT DRIVEN I/O

- Major drawback of programmed I/O is **busy waiting**

- Speed of **I/O devices is much slower than CPU**
- Performance of **CPU** goes down as it has to **repeatedly check for device status**
- ***Solution****: Switch to another task if device is not ready and thus use interrupt mechanism*
- CPU issues 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 then executes the data transfer, as before, and then resumes its former processing.


Issue Read command to I/O module
CPU → I/O
Do something else
Read status of I/O module
Interrupt
I/O → CPU
Check status
Error condition
Ready
Read word from I/O Module
I/O → CPU
Write word into memory
CPU → memory
No
Done?
Yes
Next instruction

From I/O module's point

- I/O module receives a READ command from the processor for input
- The I/O module then proceeds to read data in from an associated peripheral.
- Once the data are in the module's data register, the module signals an interrupt to the processor over a control line.
- The module then waits until its data are requested by the processor.
- When the request is made, the module places its data on the data bus and is then ready for another I/O operation.

From CPU point

- The **processor issues a READ command**.
- It then goes off and does **something else**
- At the end of each instruction cycle, the **processor checks for interrupts**
- When the **interrupt** from the I/O module occurs, the **processor saves the context** (e.g., program counter and processor registers) of the current program and **processes the interrupt.**
- In this case, the **processor reads the word of data from the I/O module and stores it in memory**.
- It then **restores the context of the program** it was working on (or some other program) and resumes execution.

Interrupt Processing-Occurrence of interrupt-H/W & s/w

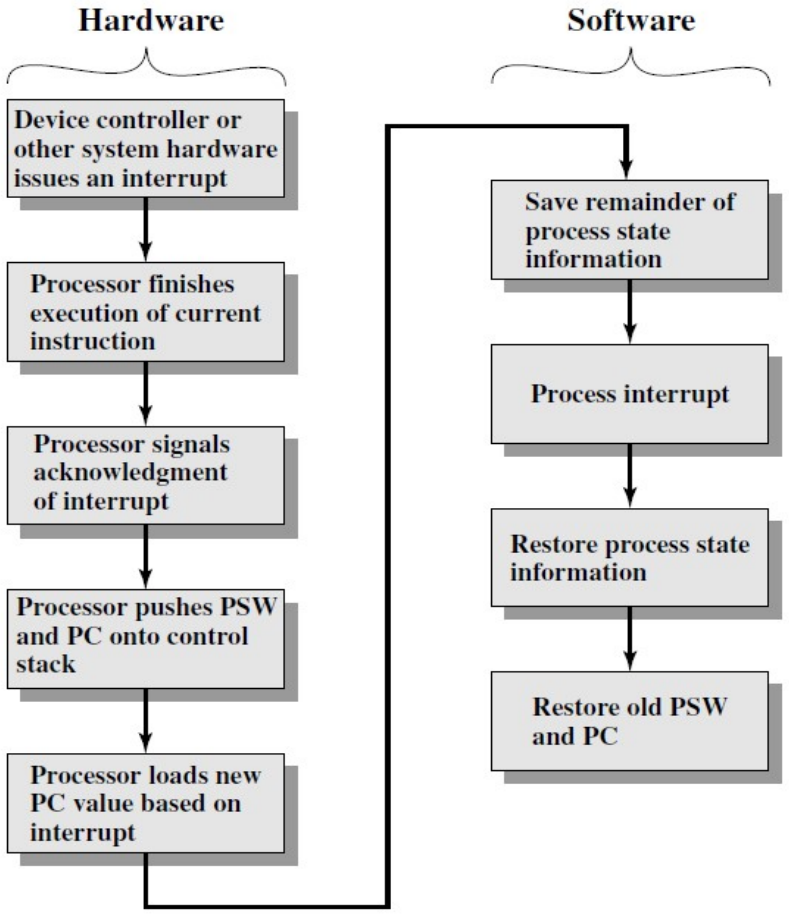


Figure 7.6 Simple Interrupt Processing

it is important to save all the state information about the interrupted program for later resumption

Interrupt Driven I/O

- **Overcomes CPU waiting**-Interrupt I/O is more efficient
- No repeated CPU checking of device
- I/O module interrupts when ready
- interrupt I/O still **consumes a lot of processor time**, (because every word of data that goes from memory to I/O module/or from I/O module to memory must pass through the processor)

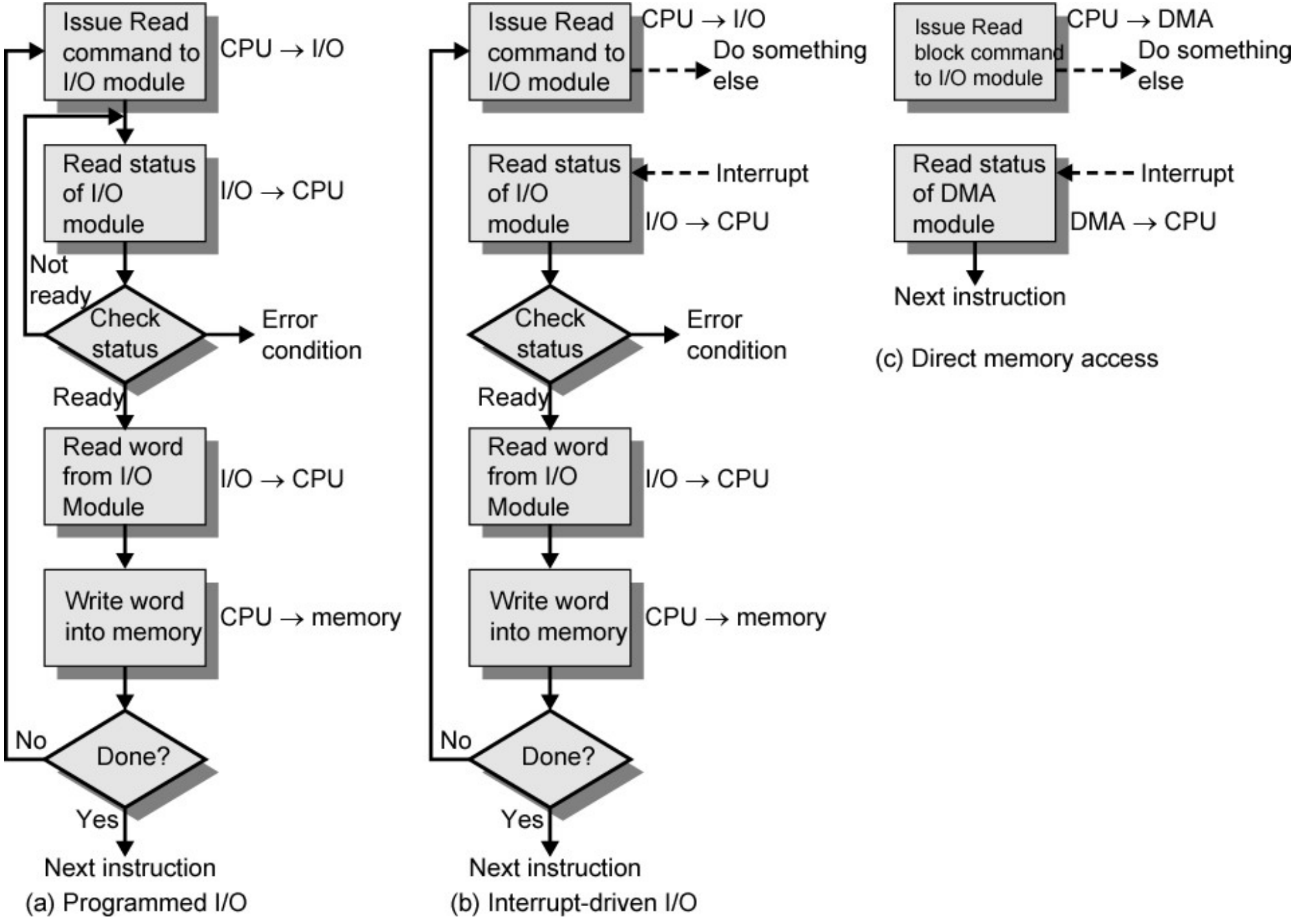
Interrupt Driven I/O
Basic Operation

- 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

CPU Viewpoint

- Issue read command
- Do other work
- Check for interrupt at end of each instruction cycle
- If interrupted:-
  - Save context (registers)
  - Process interrupt
    - Fetch data & store

Three Techniques for Input of a Block of Data



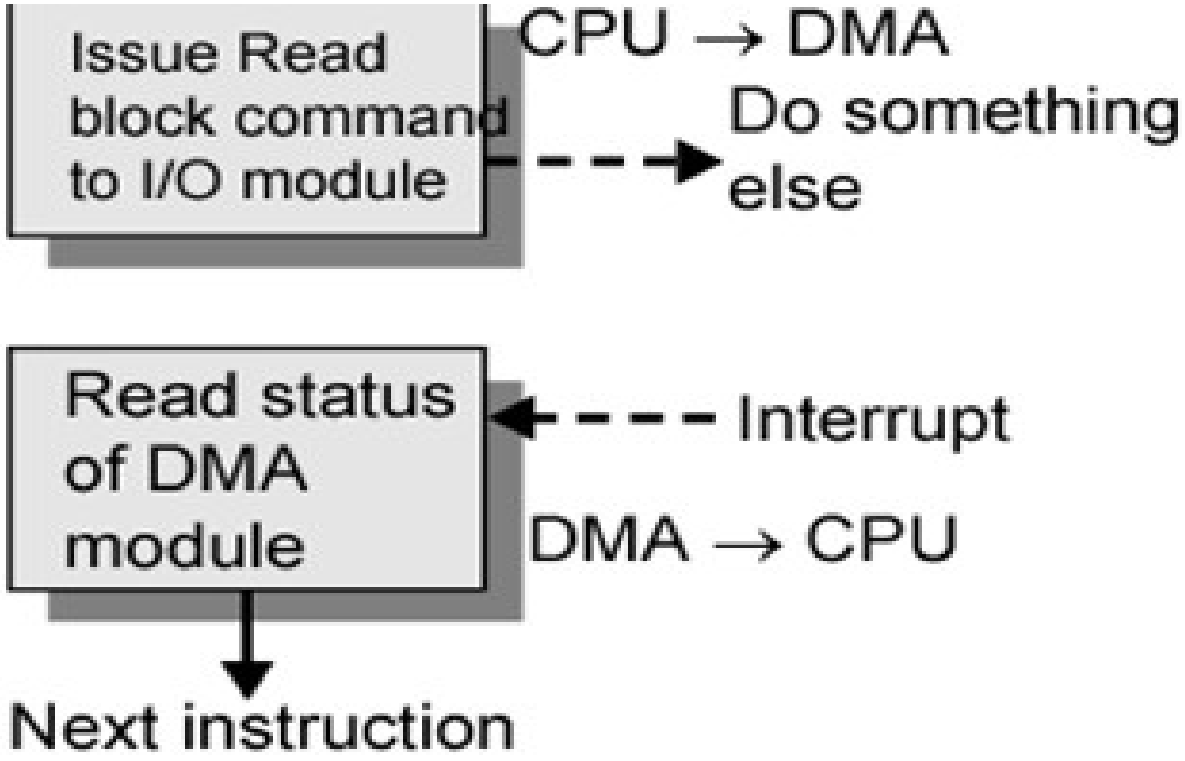
Direct Memory Access

- Interrupt driven and programmed I/O require active CPU intervention
  - —Transfer rate is limited (with which the processor can test and service a device.)
  - —CPU is tied up (managing an I/O transfer)
  - large volumes of data are to be moved-DMA

- Direct memory access (**DMA**) is a feature of computer systems that **allows certain hardware subsystems to access main system memory (RAM) independently of the central processing unit (CPU).**

DMA Function

- Involves Additional Module (hardware) on bus
- DMA controller takes over from CPU for I/O



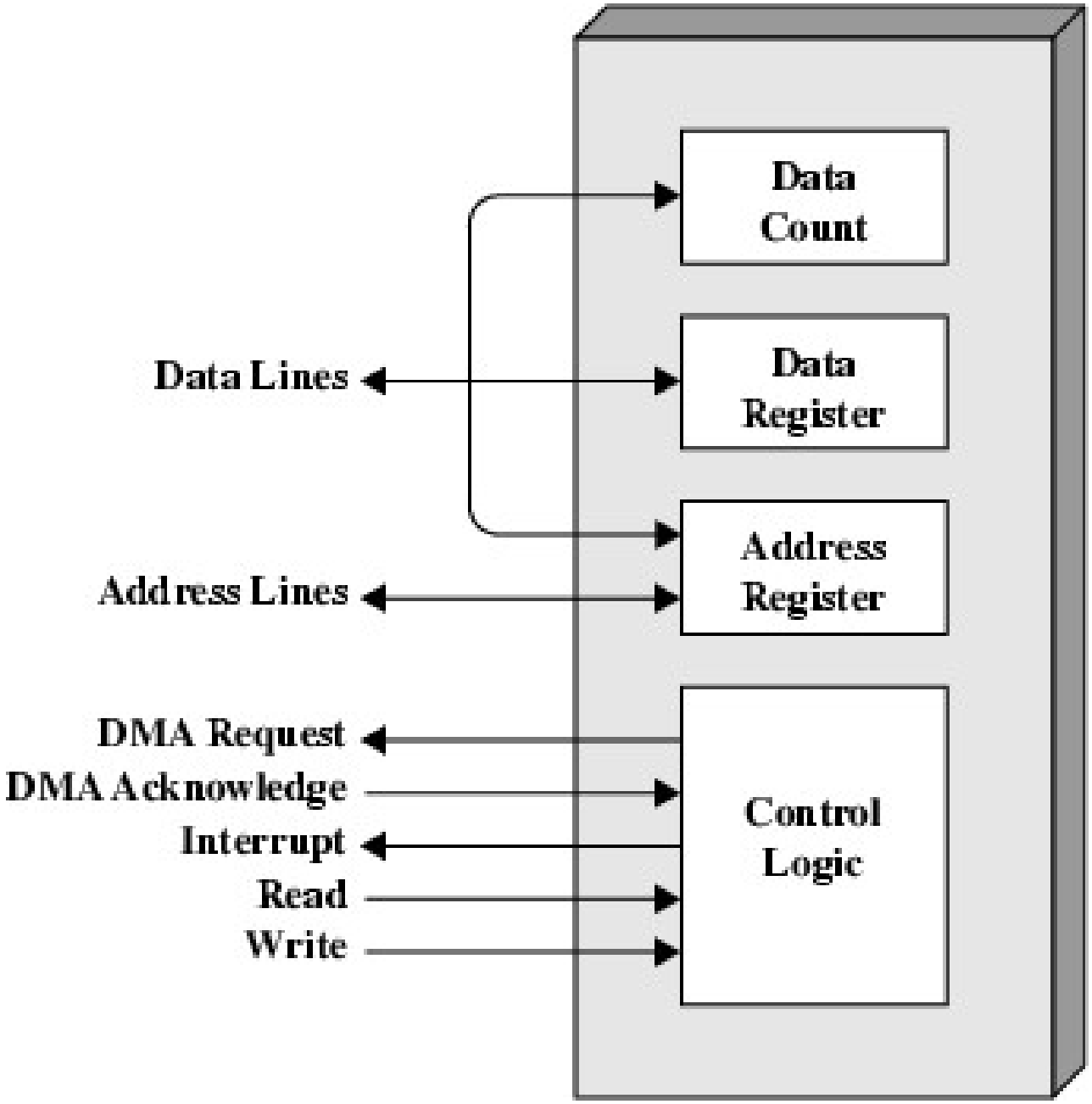
(c) Direct memory access

- The DMA module is capable of mimicking the processor-**transfer data to and from memory over the system bus**.
- DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily-more common and is referred to as ***cycle stealing***, because the DMA module in effect steals a bus cycle.

DMA Operation

- CPU tells DMA controller:-
  - Read/Write request is made using control line
  - Device address communicated on data lines
  - Starting address of memory block communicated in data line & stored by DMA
  - Amount of data to be transferred communicated in data line &stored in data count register
- CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished
- Processor involvement only at start & end of transfer

Typical DMA Module Diagram



DMA data transfer modes:

- Burst Mode
- Cycle Stealing Mode
- Transparent Mode

DMA data transfer modes

1. DMA block transfer/Burst Mode

A block of data of **arbitrary length** is transferred in a single

Burst

- Burst - Temporary high-speed data transmission mode used to facilitate sequential data transfer at maximum throughput.

2. Cycle stealing mode

- DMA controller is allowed to use system bus to transfer **one word of data at a time**, after which it must return control of the bus to the CPU.
- The DMA module uses the system bus only when the **processor does not need it**, or it must force the processor to suspend operation temporarily.
- Referred to as **cycle stealing**, because the DMA module in effect steals a bus cycle.

DMA data transfer modes:

3.Transparent DMA

DMA is allowed to steal only those cycles **when CPU is not using system bus**

Advantages of DMA

- High transfer rates
  - fewer CPU cycles for each transfer.
  - DMA speedups the memory operations by bypassing the involvement of the CPU.
- Work overload on the CPU decreases.

Disadvantages of DMA

- DMA transfer requires a **DMA controller** to carry out the operation
- More **expensive** system
- **Synchronization mechanisms** must be provided in order to avoid accessing **non**-updated information from RAM
  - **Cache coherence problem (**Cache and the main memory may have inconsistent copies of the same object.**)**