William Stallings
Computer Organization and Architecture
7th Edition

Chapter 7

Input/Output

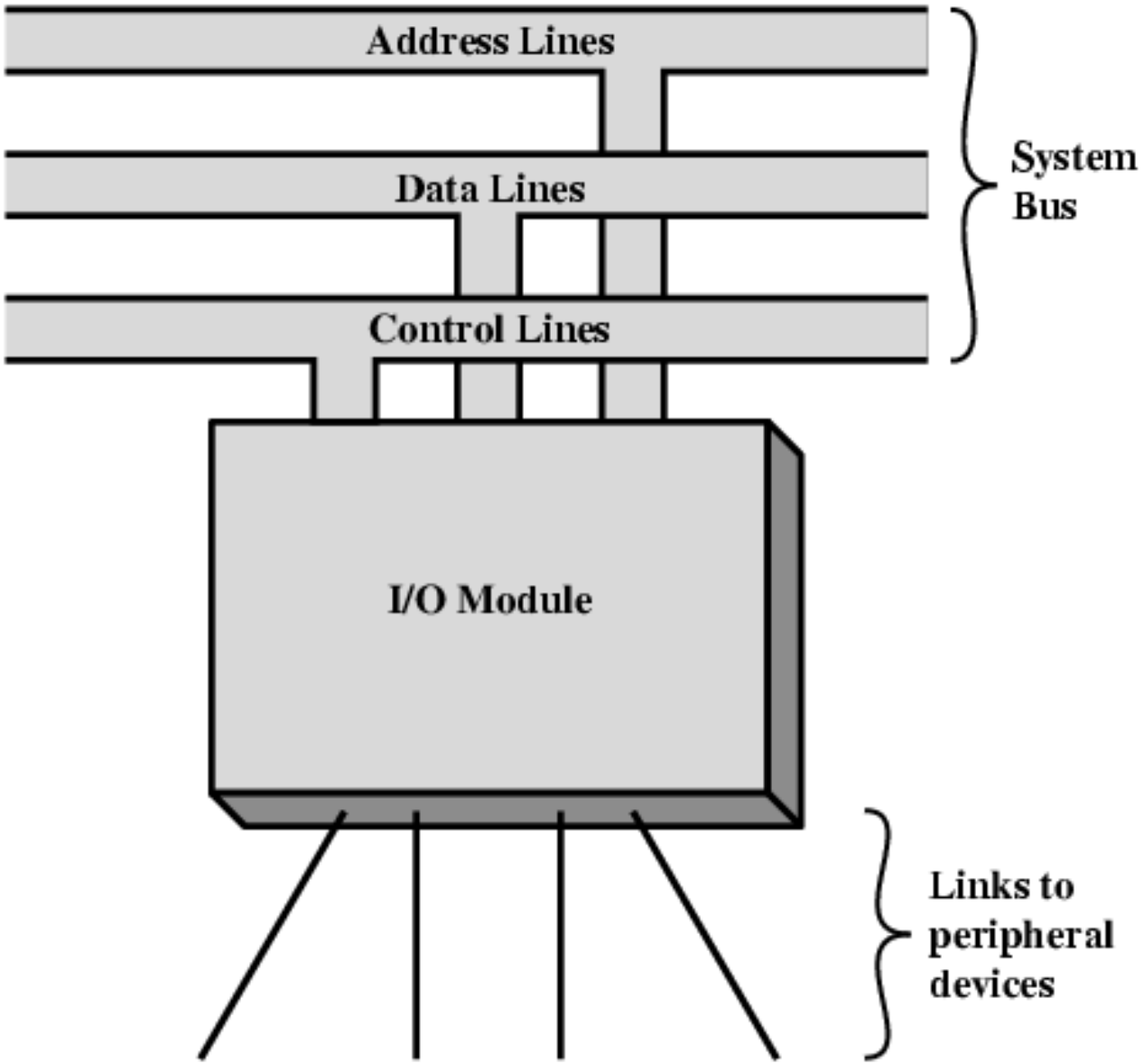
Input/Output Problems

- I/O contains some “intelligence”.
- Wide variety of peripherals
  - Delivering different amounts of data
  - At different speeds
  - In different formats
- All slower than CPU and RAM
- Need I/O modules

Input/Output Module

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

Generic Model of I/O Module



External Devices

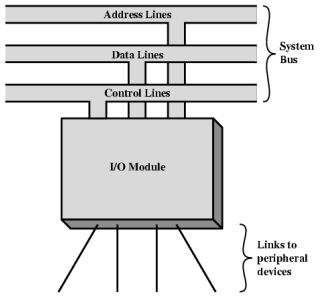
- Human readable
  - —Screen, printer, keyboard
- Machine readable
  - —Monitoring and control
  - —Magnetic disk and tape, sensors and actuators used in robotics application.
- Communication
  - —Modem,Network Interface Card (NIC)

I/O Module Functions

- Control and timing
- Processor communication
- Device communication
- Data buffering
- Error detection

1.I/O Steps(control of the transfer of data from an external device to the processor)

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

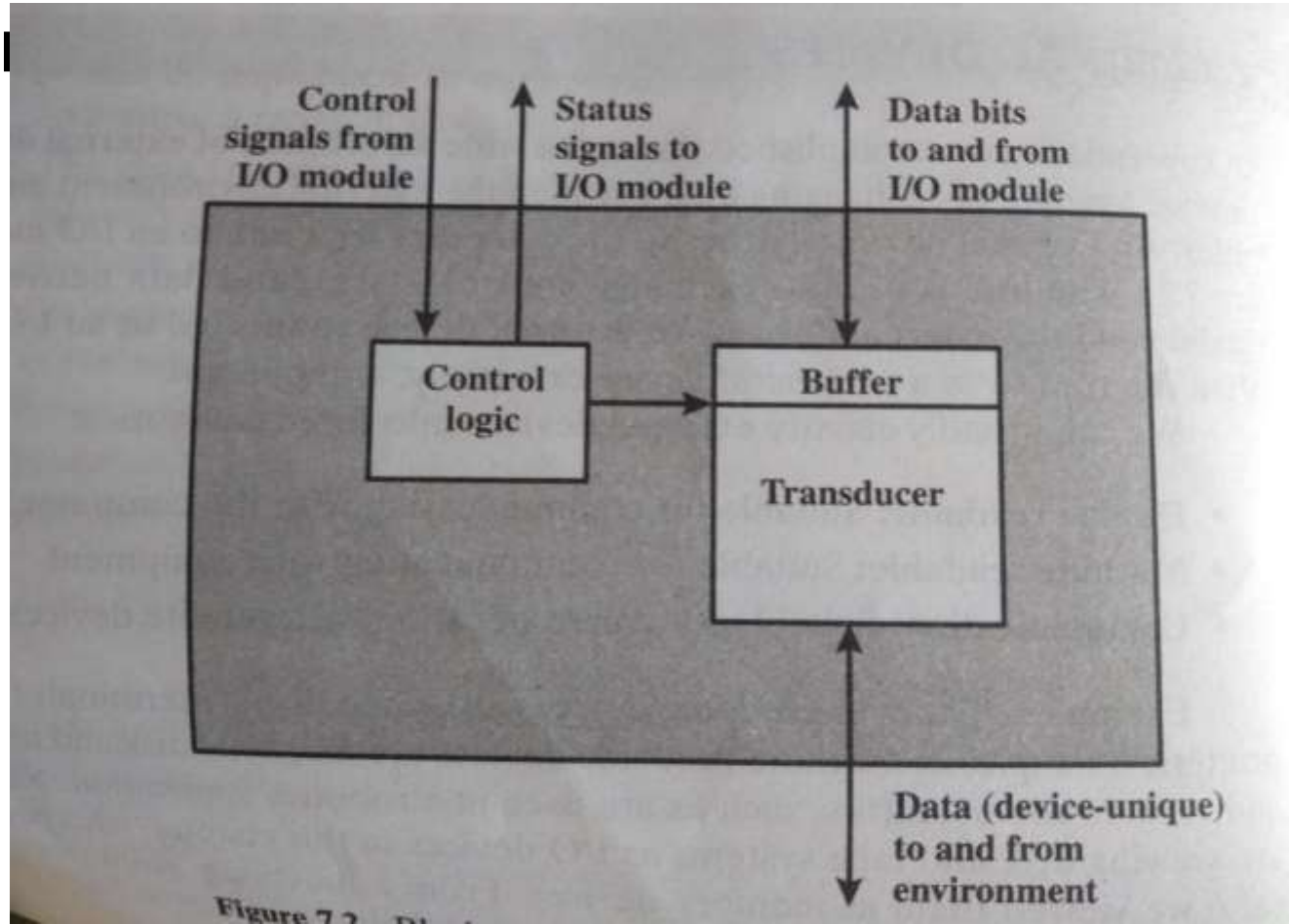


2. processor communication

- Command decoding
- Data
- Status reporting(busy/ready)
- Address recognition

3. Device communication

- Commands
- Status infor
- data



4. Data buffering

- Transfer rate of processor and memory is high whereas I/O device is low.
- Data are buffered in I/O module and then sent to the peripheral device at its data rate and vice versa.

5. Error detection

- Reporting errors to the processor.
- Mechanical and electrical(paper jam,bad disk track)
- Transmission errors.(use parity bit)

Block diagram of I/O module

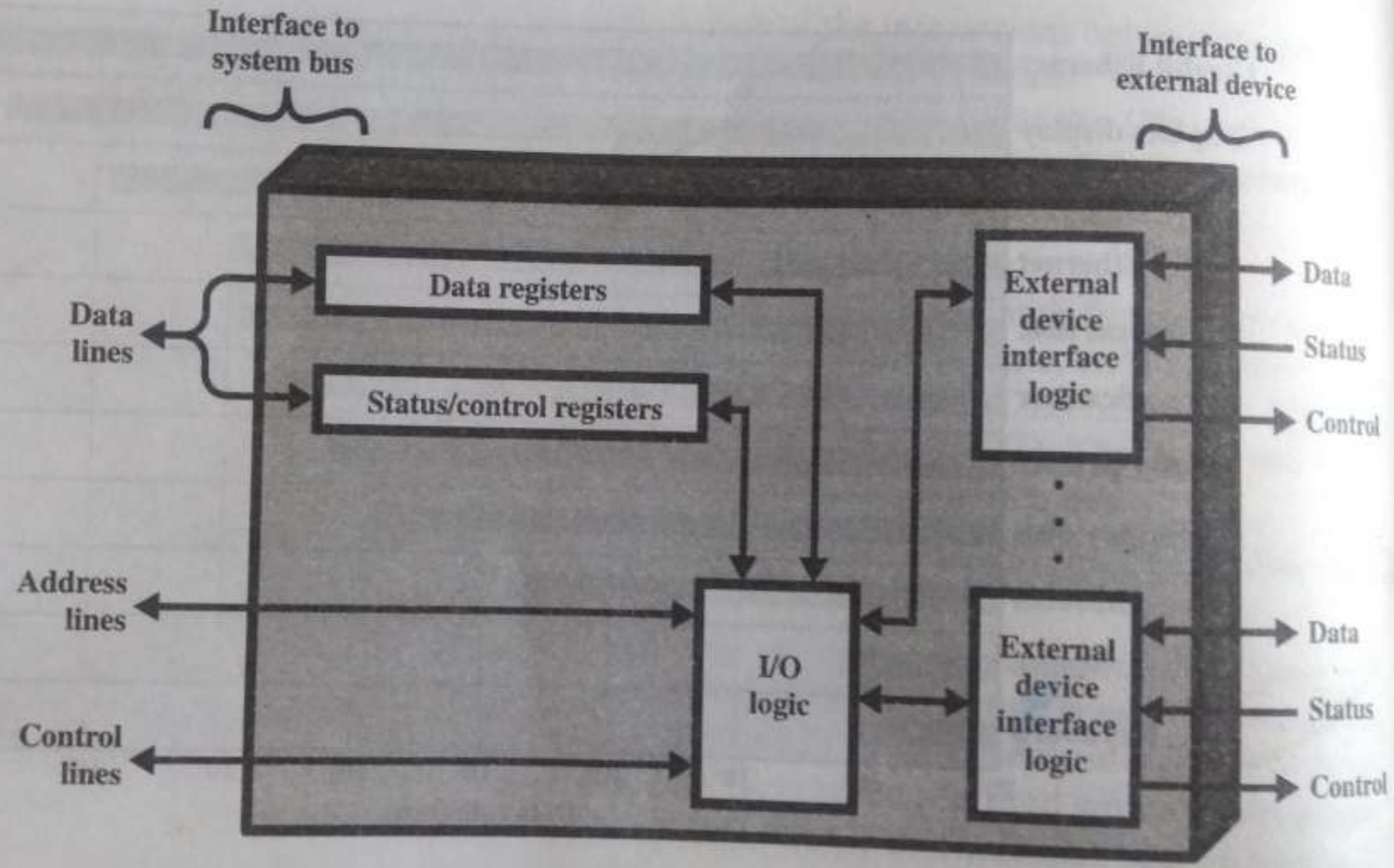


Figure 7.4 Block Diagram of an I/O Module

Input Output Techniques

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

PROGRAMMED I/O

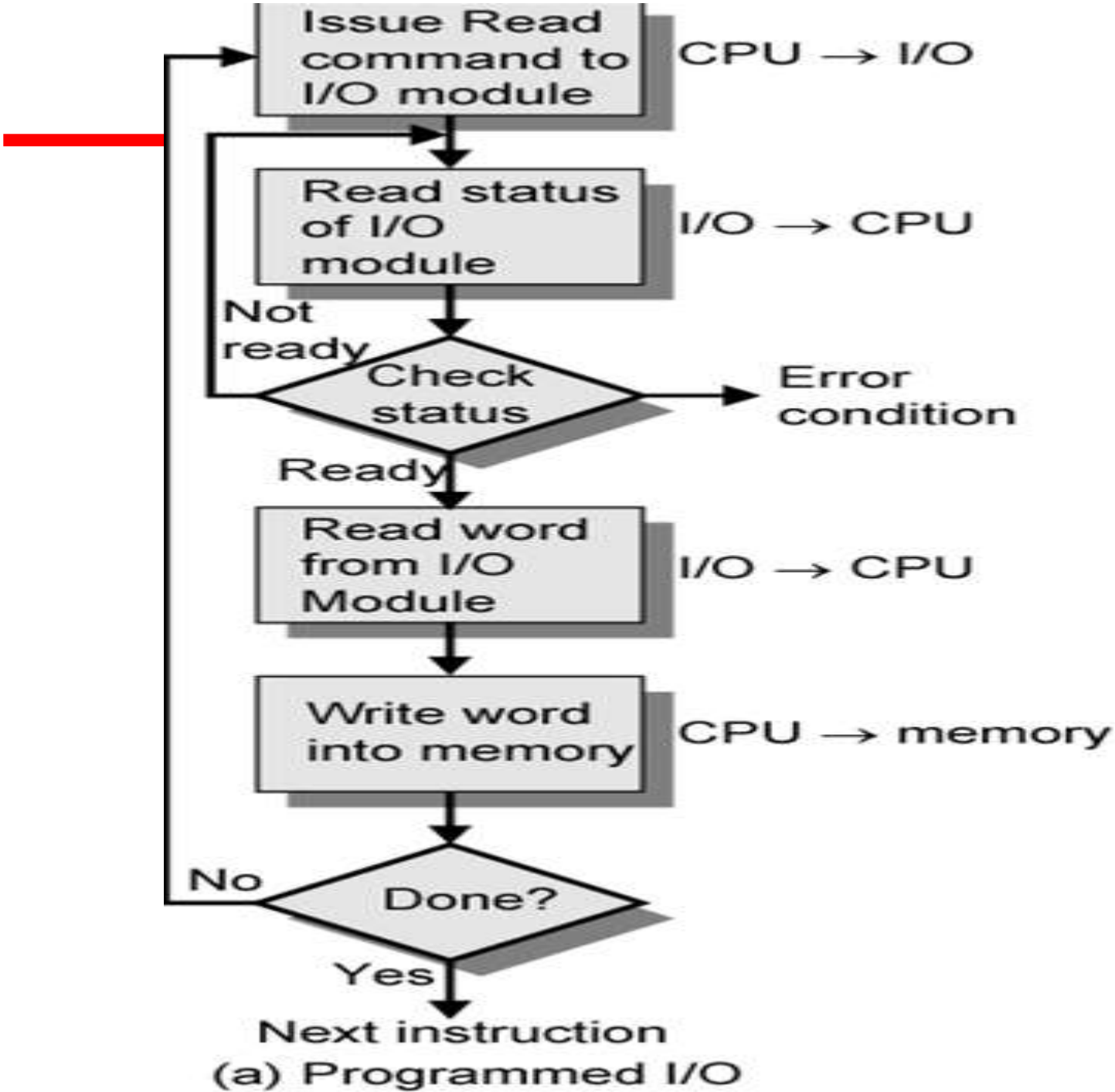
- Data transfer operations are completely controlled by **CPU** i.e. CPU executes a program that initiates, directs and terminates the I/O operation

PROGRAMMED I/O: commands

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

Programmed I/O

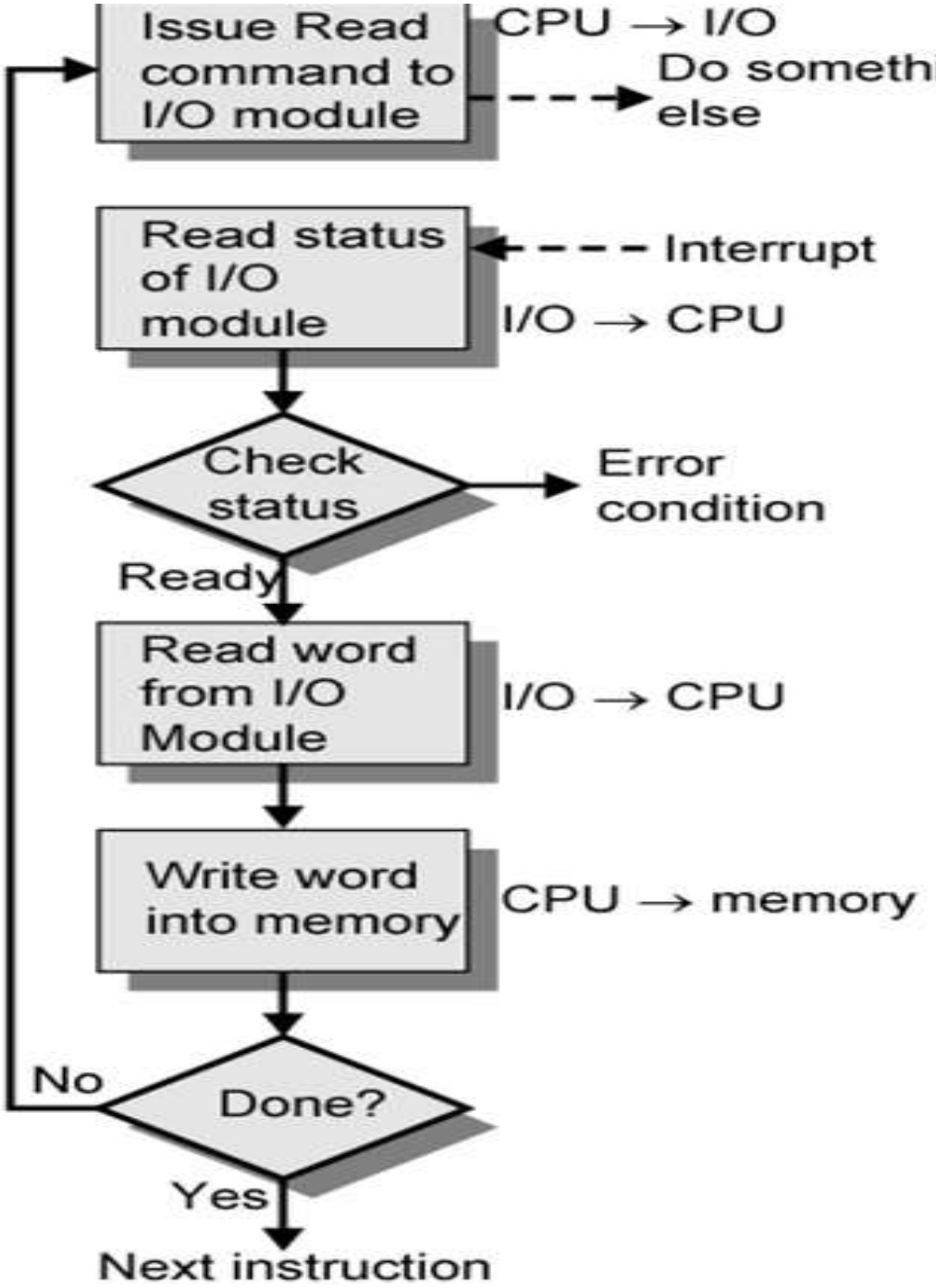
- CPU has direct control over I/O
  - Sensing status
  - Read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time

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


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
Done?
No
Yes
Next instruction

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 while 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

From I/O module Viewpoint

- I/O module receives a READ command from the processor.
- Read data from associated peripheral.
- Once data is in the module's data register, the module signals an interrupt over control lines.

INTERRUPT DRIVEN I/O

- An interrupt causes CPU to temporarily transfer control from its current program to another program(an interrupt handler).
- Types of interrupts:

1. Program interrupts(e.g. divide by zero)
2. Timer interrupts(within the processor)
3. I/O interrupts(generated for initiation/completion of I/O)
4. Hardware failure interrupts(power failure,memory parity error)

INTERRUPT DRIVEN I/O

- Pentium interrupt structure

1. Hardware interrupts:

   NMI(Non Maskable interrupts : cannot be disabled)

   INTR(Interrupt Request: can be disabled)

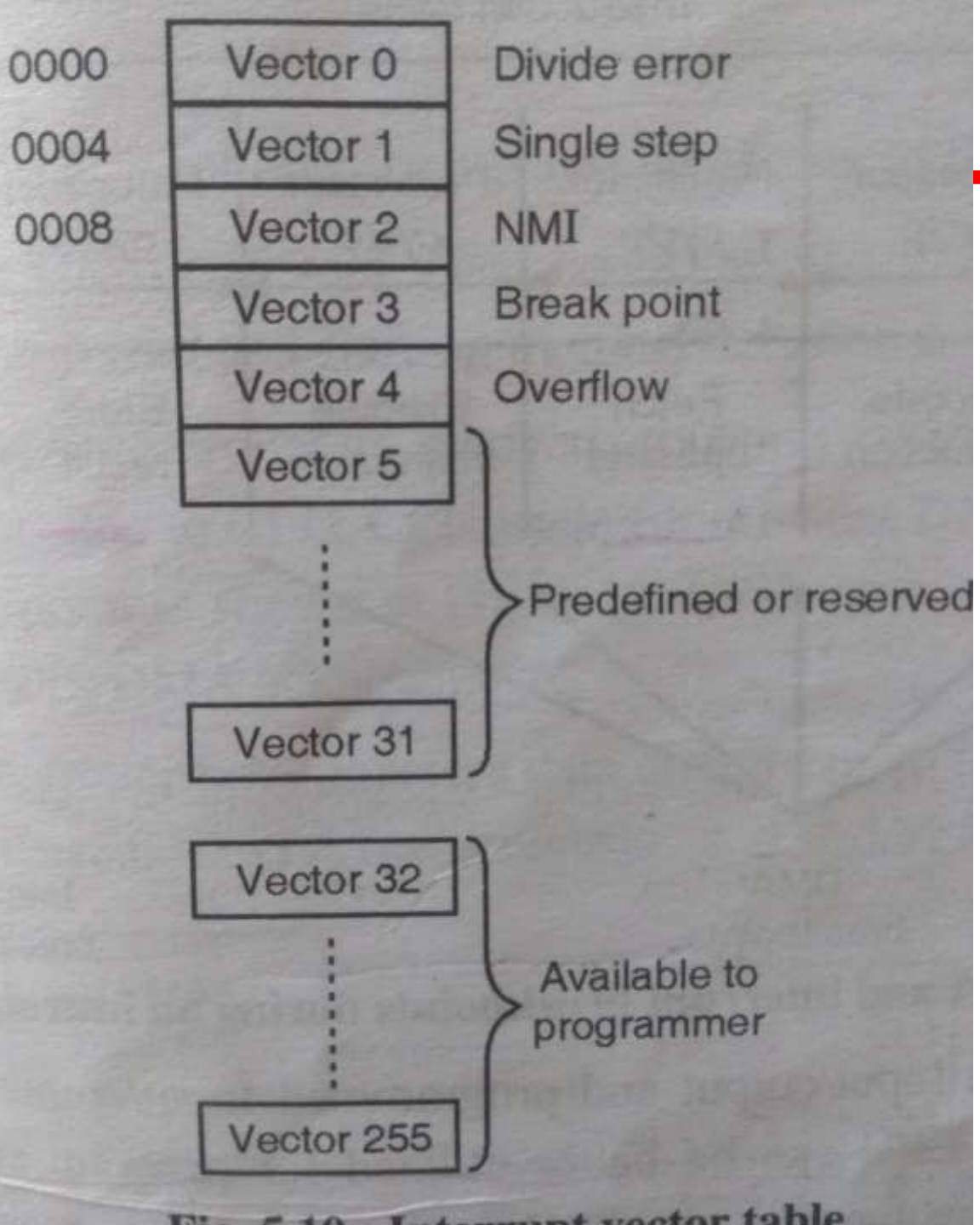
2. Software interrupts(exceptions):

   - generated by an executing program

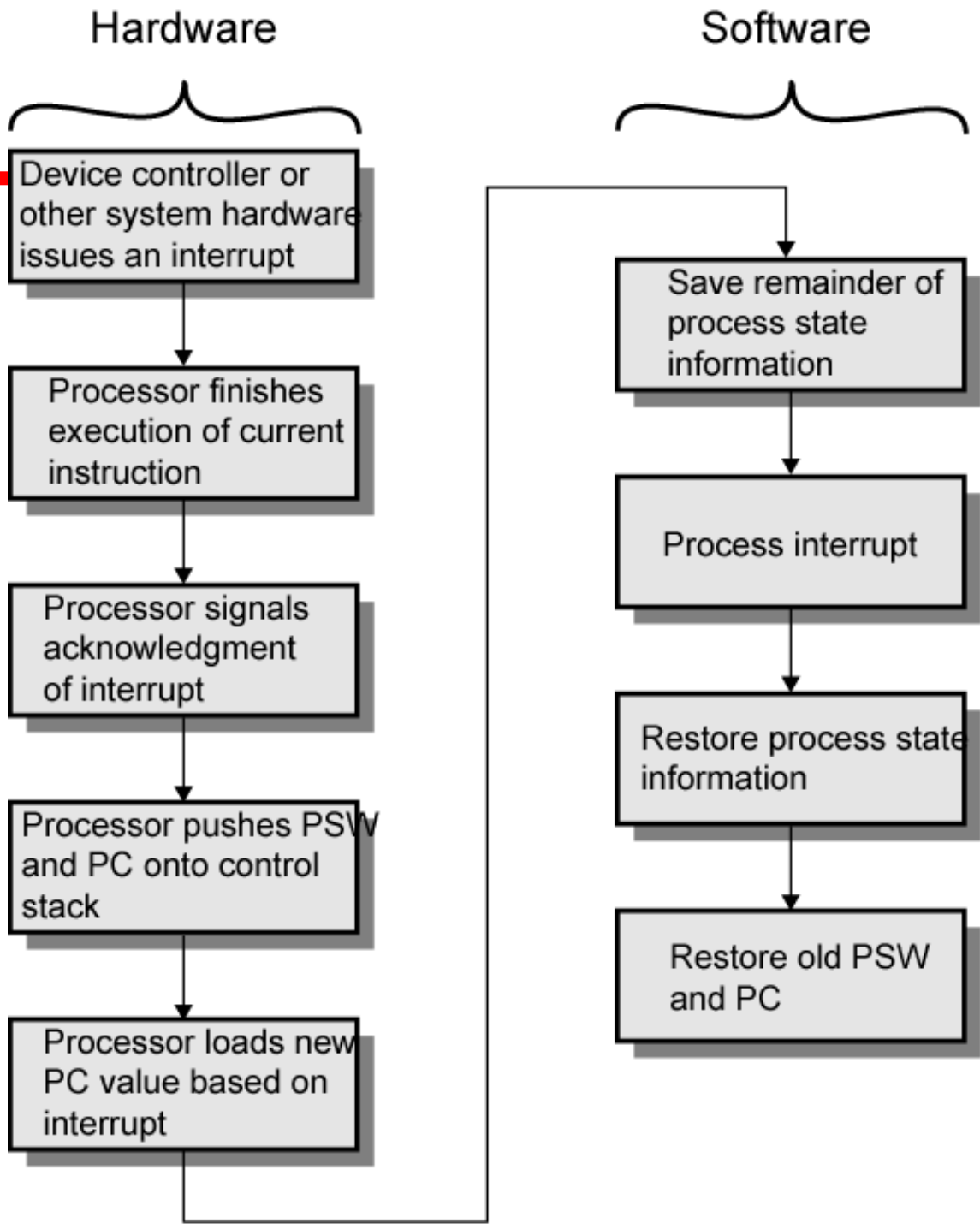
IN

2.	NMI
3.	Break point
4.	INTO
5.	BOUND range exceeded
6.	Invalid opcode
7.	Device not available
8.	Double fault
9.	Reserved
10.	Invalid task state segment
11.	Segment not present
12.	Stack exception
13.	General protection
14.	Page fault
15.	Reserved
16.	Floating point error
17.	Alignment check
18.	Machine check
19-31	Reserved
32-255	Maskable interrupts

INTE


0000
Vector 0
Divide error
0004
Vector 1
Single step
0008
Vector 2
NMI
Vector 3
Break point
Vector 4
Overflow
Vector 5
Predefined or reserved
Vector 31
Vector 32
Available to programmer
Vector 255

Simple Interrupt Processing



Design issues:

- Multiple I/O devices: How does the processor determine which device issued the interrupt?

  -Use multiple interrupt lines: independent request,software poll, daisy chain

- If multiple interrupts have occurred, how does the processor decide which one to process?

  -assign priorities to interrupts.

Multiple Interrupts

- Each interrupt line has a priority
- Higher priority lines can interrupt lower priority lines
- If bus mastering only current master can interrupt

Direct Memory Access

- Interrupt driven and programmed I/O require active CPU intervention
  - Transfer rate is limited
  - CPU is tied up
- DMA is the answer
- 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).

Direct Memory Access

acknowledge.

3. The DMA controller now transfers data directly to or from main mem
word is transferred, 'DATA count register' and 'Address register' are

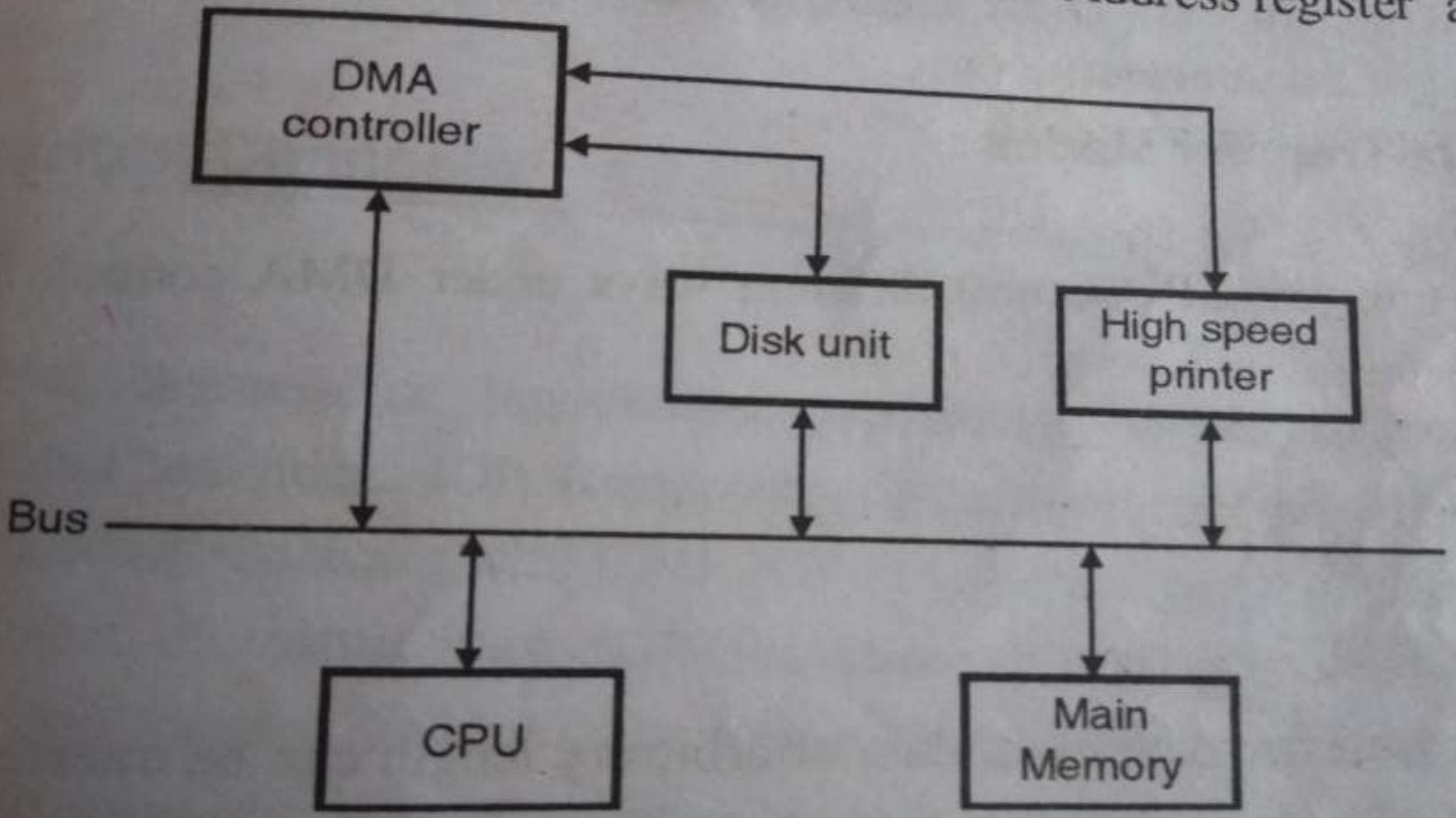


Fig. 5.13 : A two channel DMA controller

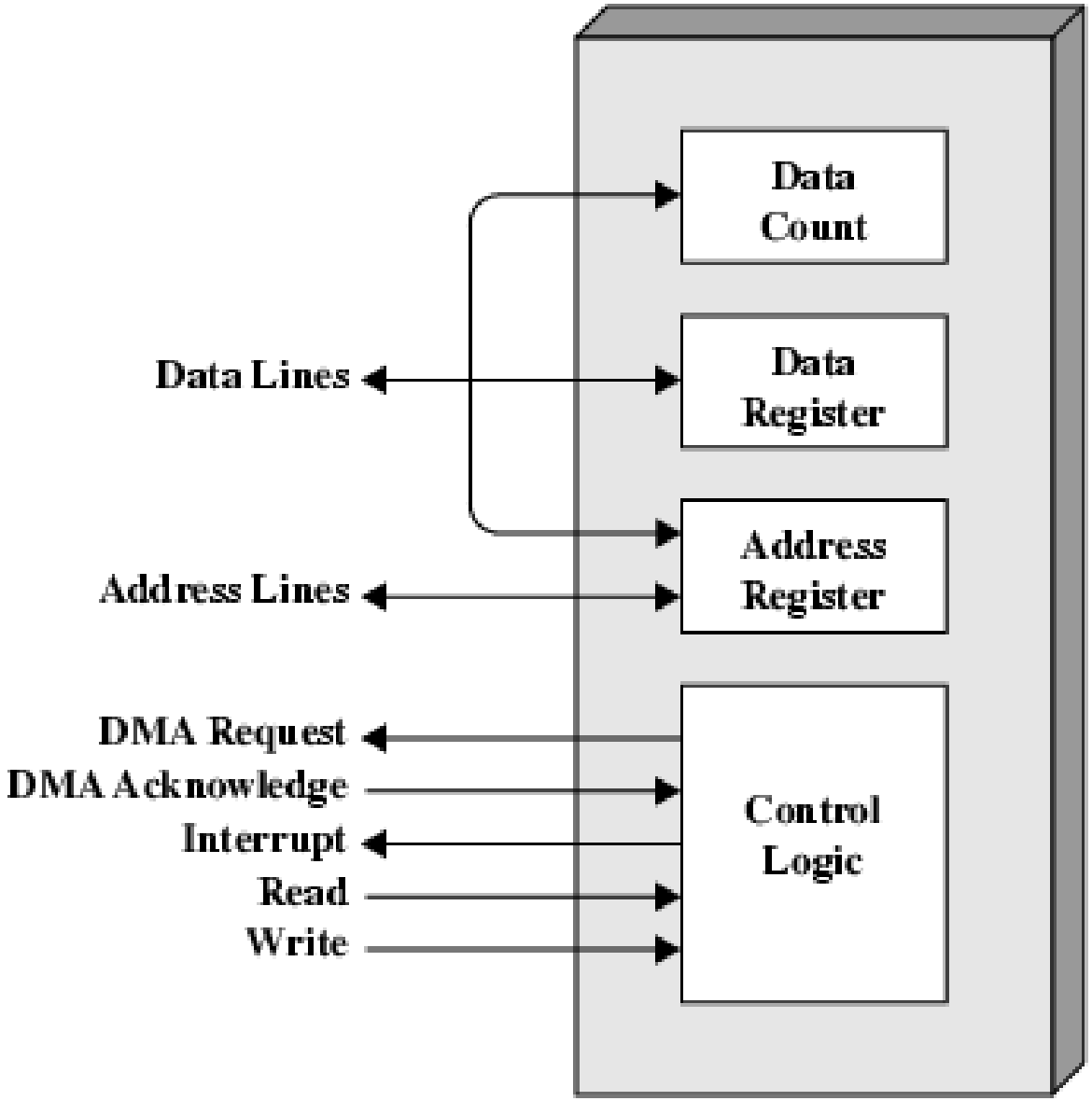
Direct Memory Access

- When an input/output is requested, the CPU instructs the DMA module about the operation

1. Which operation(read/write) to be performed
2. The address of I/O device to be used
3. Starting location in memory where the info will be read or written.
4. No of words to be read/written.

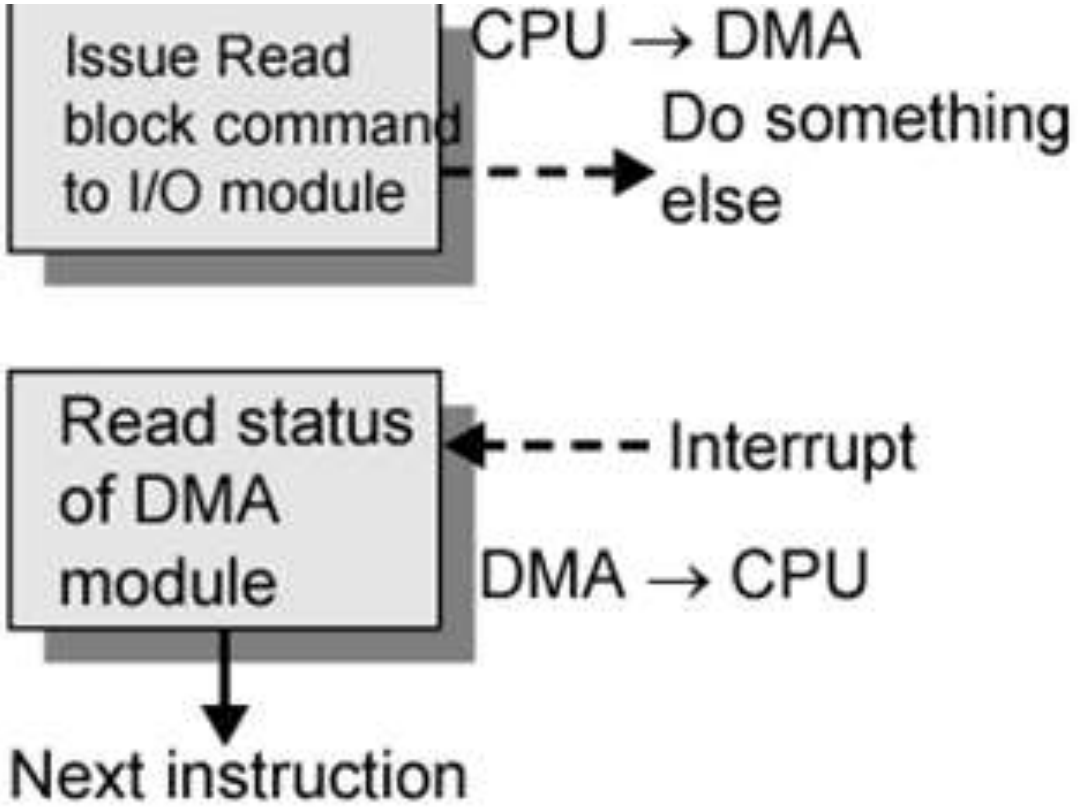
- Processor delegates I/O operation to the DMA module and continues with other work

Typical DMA Module Diagram



DMA Function

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



(c) Direct memory access

DMA Operation

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

DMA Operation

- **Transferring a block data from memory to the Printer**

1. OS writes foll info to DMA registers
- memory address
- Word count
- Read or write operation

2.DMA controller checks whether printer is ready

3.If data count has not reached zero but printer is not ready ,the DMA controller releases system bus

4.If data count register is decremented to zero, DMA controller releases the control of system bus. and sends an interrupt signal to CPU.

DMA Operation

- DMA data transfer modes:

1. DMA block transfer

A block of data of arbitrary length is transferred in a single burst.

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.
- CPU and DMA use the system bus alternatively.
- After each cycle(fetch instruction ,decode instruction, fetch operand, execute instruction, store result)

DMA Operation

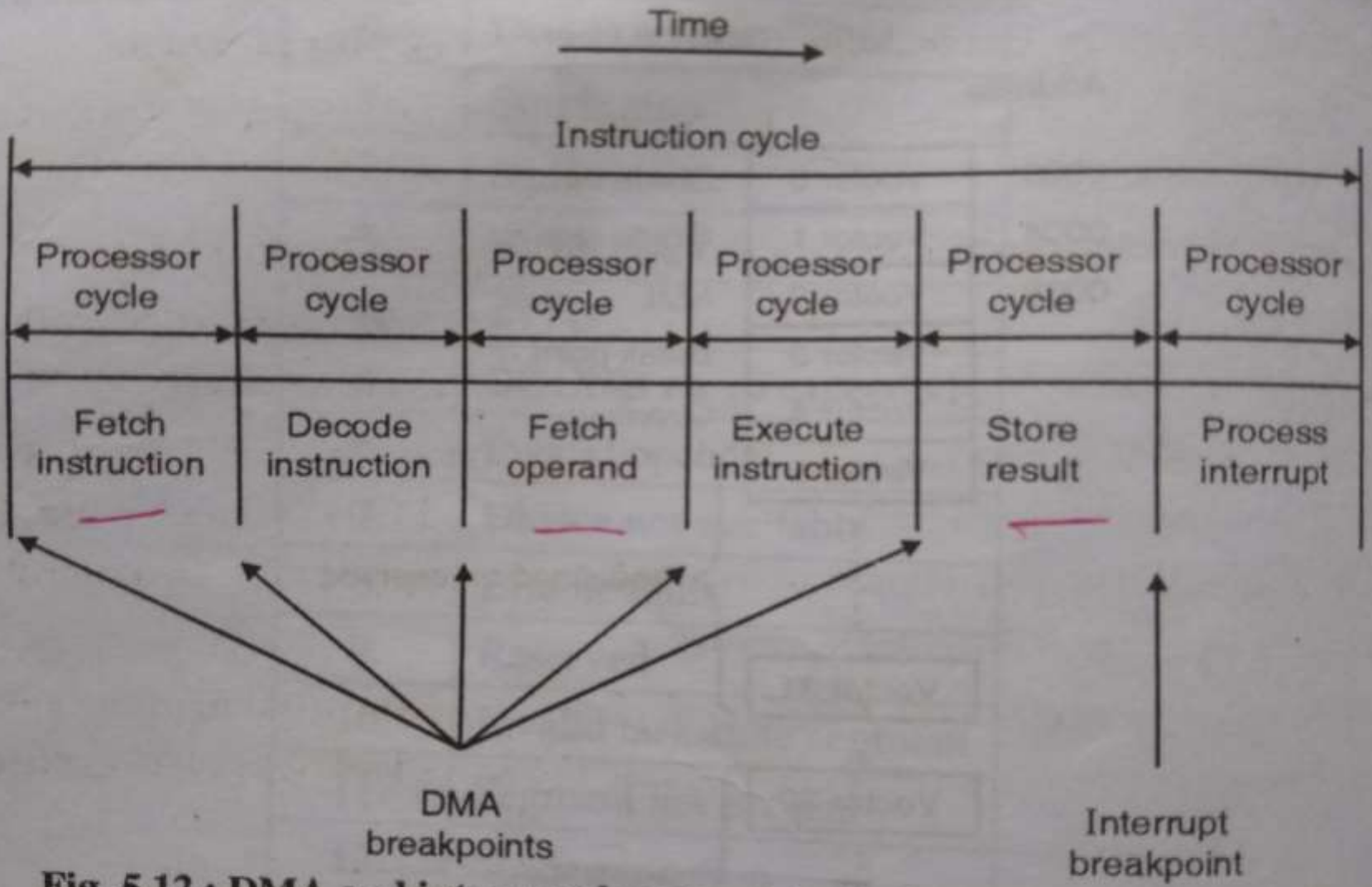


Fig. 5.12 : DMA and interrupt breakpoints during an instruction cycle

Both interrupt-driven Input/output and programmed Input/output requires inv
f CPU for data transfer. CPU can be better

DMA Operation

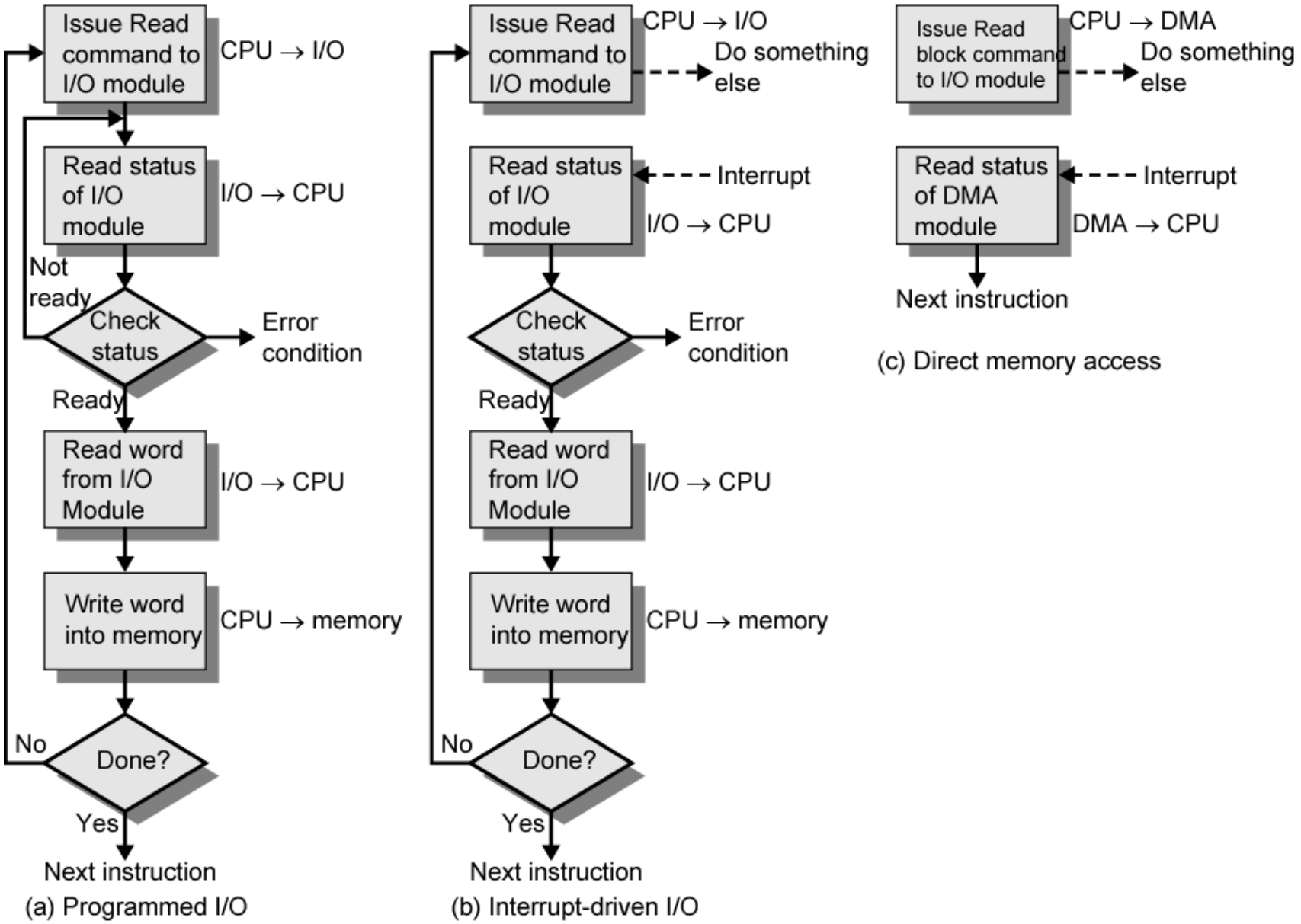
- DMA data transfer modes:

3.**Transparent DMA**

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

e.g. decode instruction and execute instruction

Three Techniques for Input of a Block of Data



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