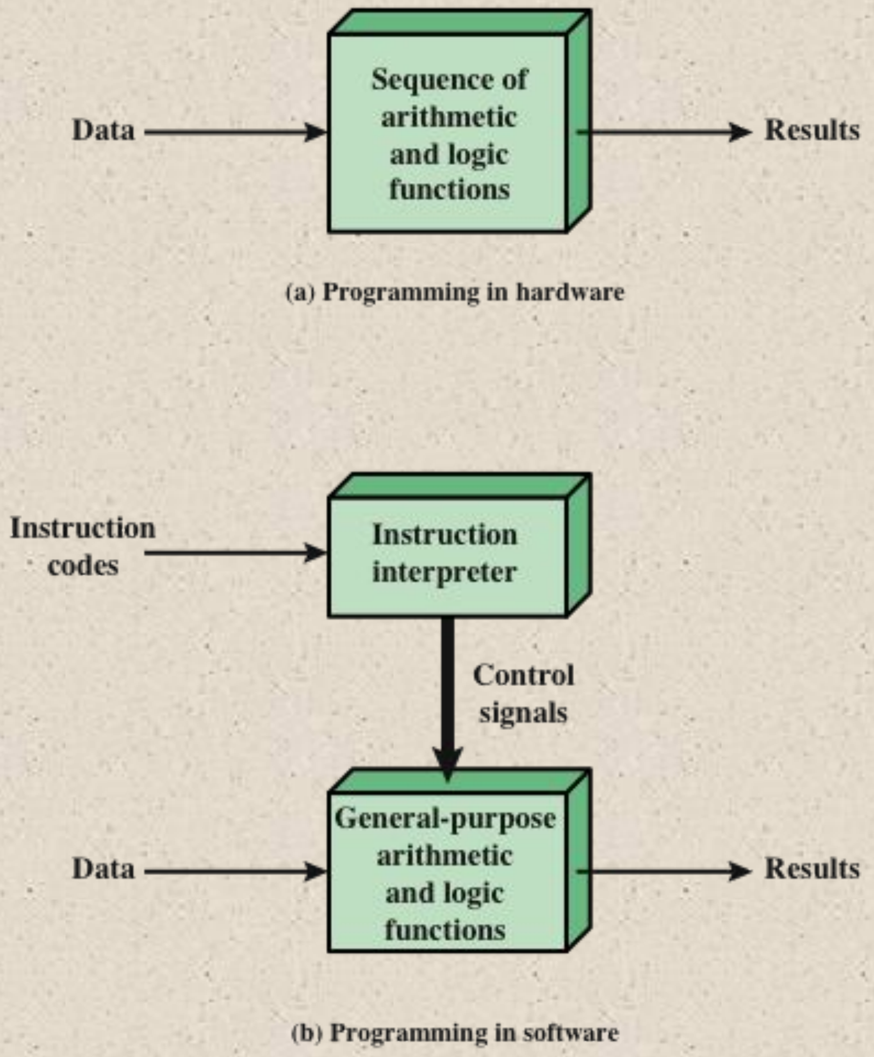
+ # Chapter 3

A Top-Level View of Computer Function and Interconnection

+

## Design a Computer System: Hardware and Software Approaches



Data
Sequence of arithmetic and logic functions
Results
(a) Programming in hardware
Instruction codes
Instruction interpreter
Control signals
Data
General-purpose arithmetic and logic functions
Results
(b) Programming in software

Figure 3.1 Hardware and Software Approaches

+
## Major components

- CPU
  - Instruction interpreter
  - Module of general-purpose arithmetic and logic functions
- I/O Components
  - Input module
    - Contains basic components for accepting data and instructions and converting them into an internal form of signals usable by the system
  - Output module
    - Means of reporting results
- Memory

+

## Memory

### Memory address register (MAR)

- Specifies the address in memory for the next read or write

### Memory buffer register (MBR)

- Contains the data to be written into memory or receives the data read from memory

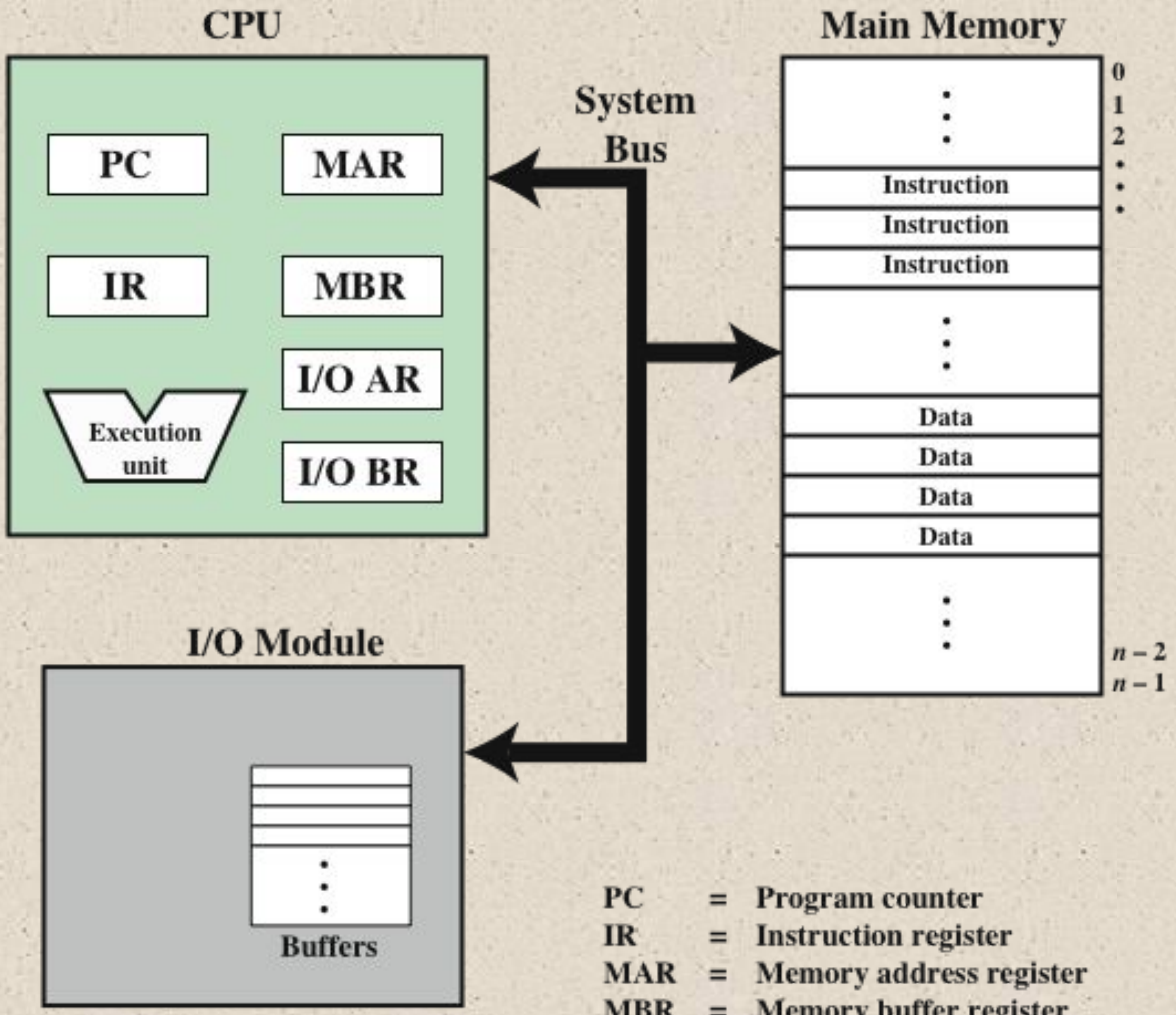
### I/O address register (I/OAR)

- Specifies a particular I/O device

### I/O buffer register (I/OBR)

- Used for the exchange of data between an I/O module and the CPU

## Computer Components: Top Level View



| | | |
|---|---|---|
| PC | = | Program counter |
| IR | = | Instruction register |
| MAR | = | Memory address register |
| MBR | = | Memory buffer register |
| I/O AR | = | Input/output address register |
| I/O BR | = | Input/output buffer register |

Figure 3.2 Computer Components: Top-Level View

+

## Basic Instruction Cycle

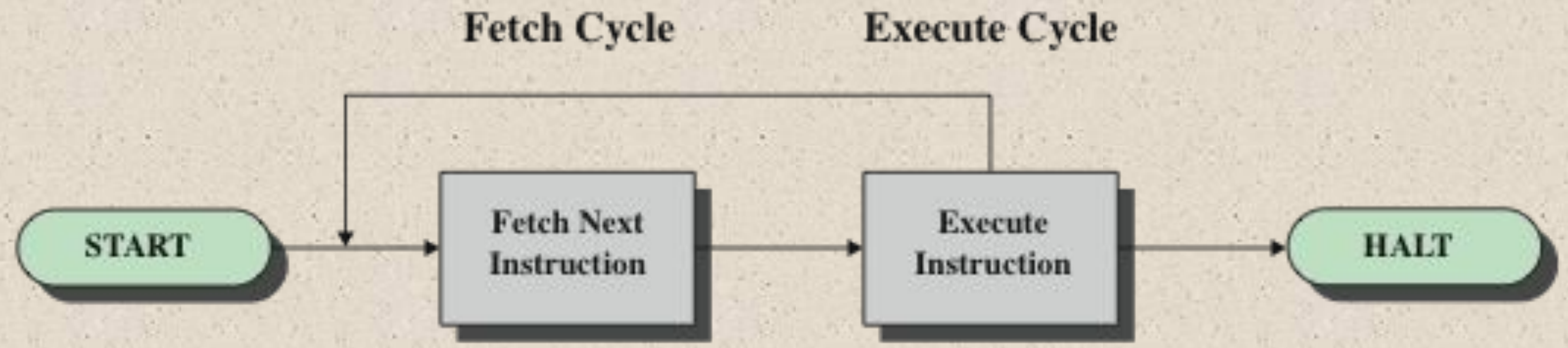


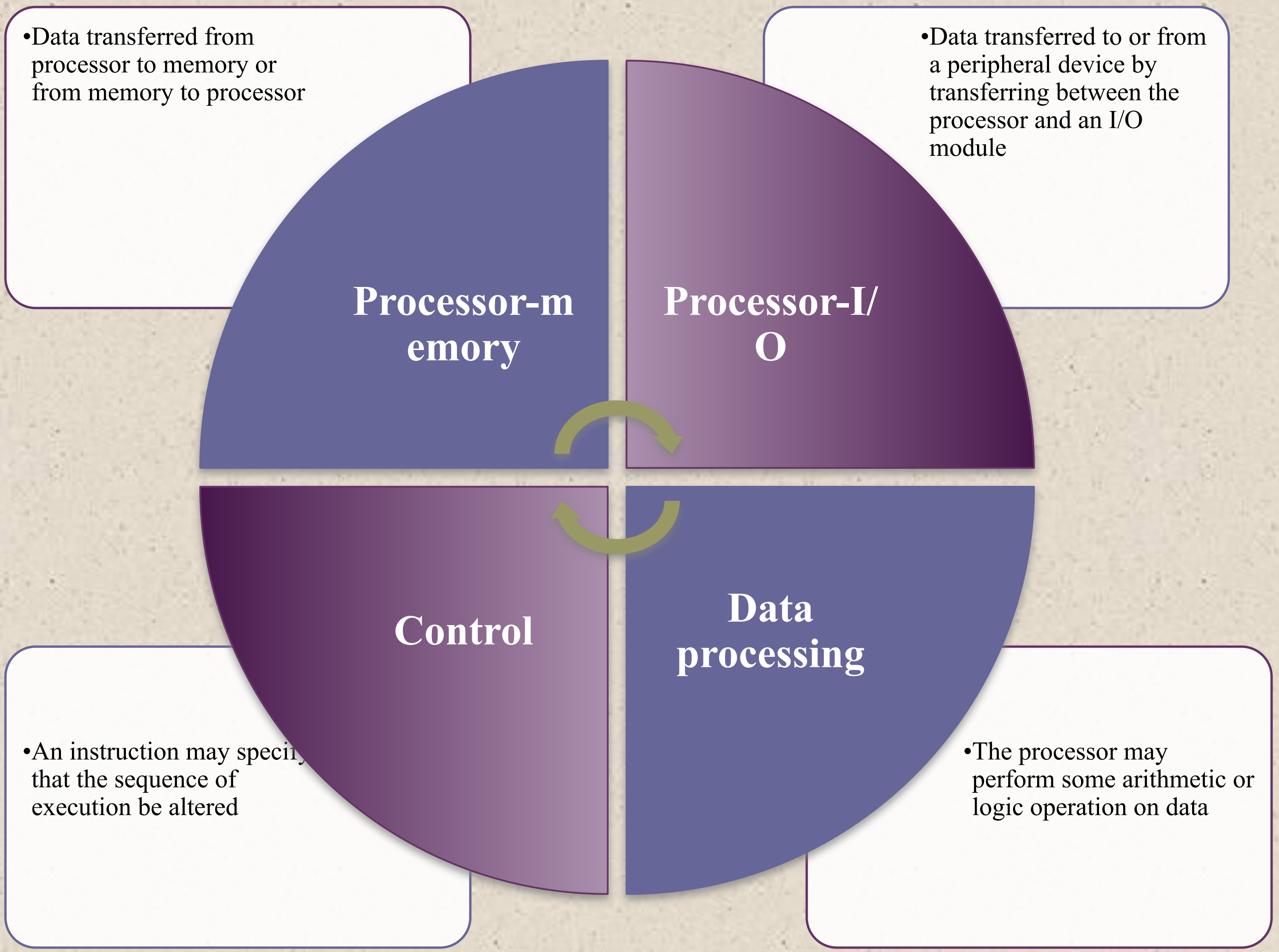
Figure 3.3 Basic Instruction Cycle

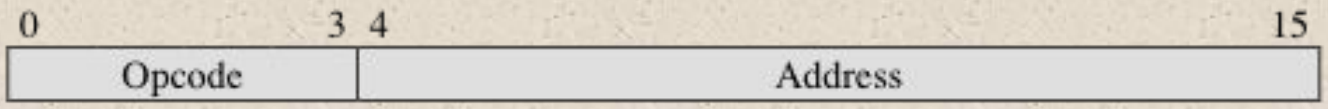
+

# Instruction Cycle

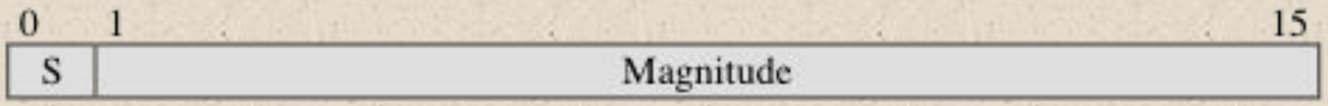
- Fetch Cycle
  - At the beginning of each instruction cycle the processor fetches an instruction from memory
  - The program counter (PC) holds the address of the instruction to be fetched next
  - The processor increments the PC after each instruction fetch so that it will fetch the next instruction in sequence
  - The fetched instruction is loaded into the instruction register (IR)
- Execute Cycle
  - The processor interprets the instruction and performs the required action

# Action Categories


•Data transferred from processor to memory or from memory to processor
Processor-memory
•Data transferred to or from a peripheral device by transferring between the processor and an I/O module
Processor-I/O
•An instruction may specify that the sequence of execution be altered
Control
•The processor may perform some arithmetic or logic operation on data
Data processing



(a) Instruction format



(b) Integer format

Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

**Figure 3.4 Characteristics of a Hypothetical Machine**

+

## Example of Program Execution

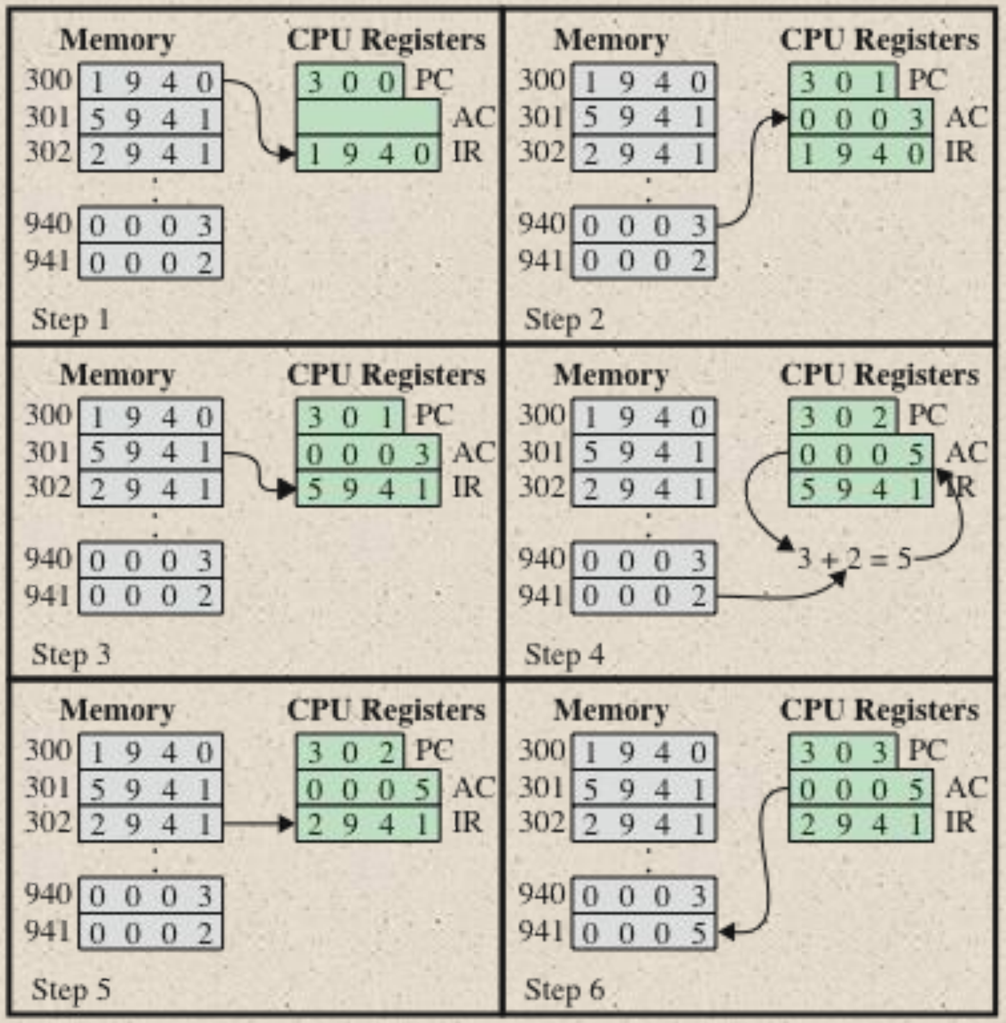
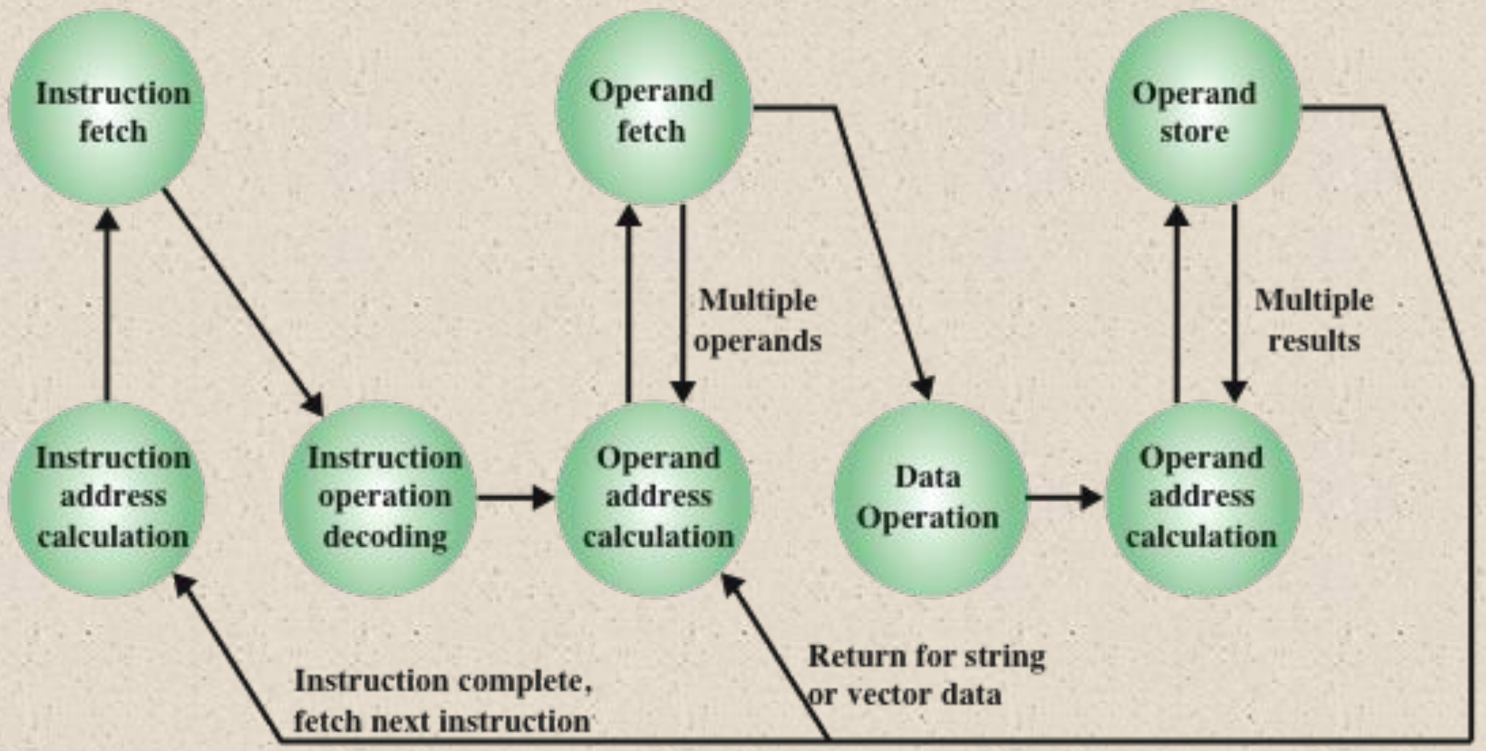


**Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)**

+

## Instruction Cycle State Diagram

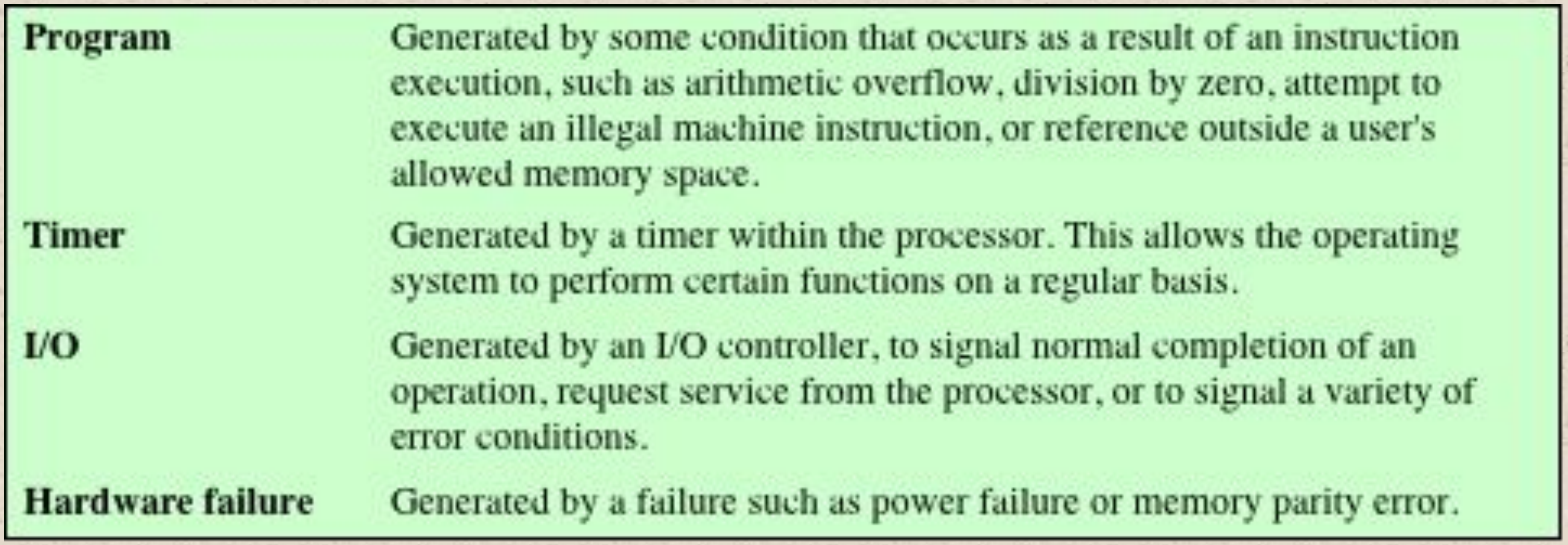


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Instruction complete, fetch next instruction
Return for string or vector data

**Figure 3.6 Instruction Cycle State Diagram**

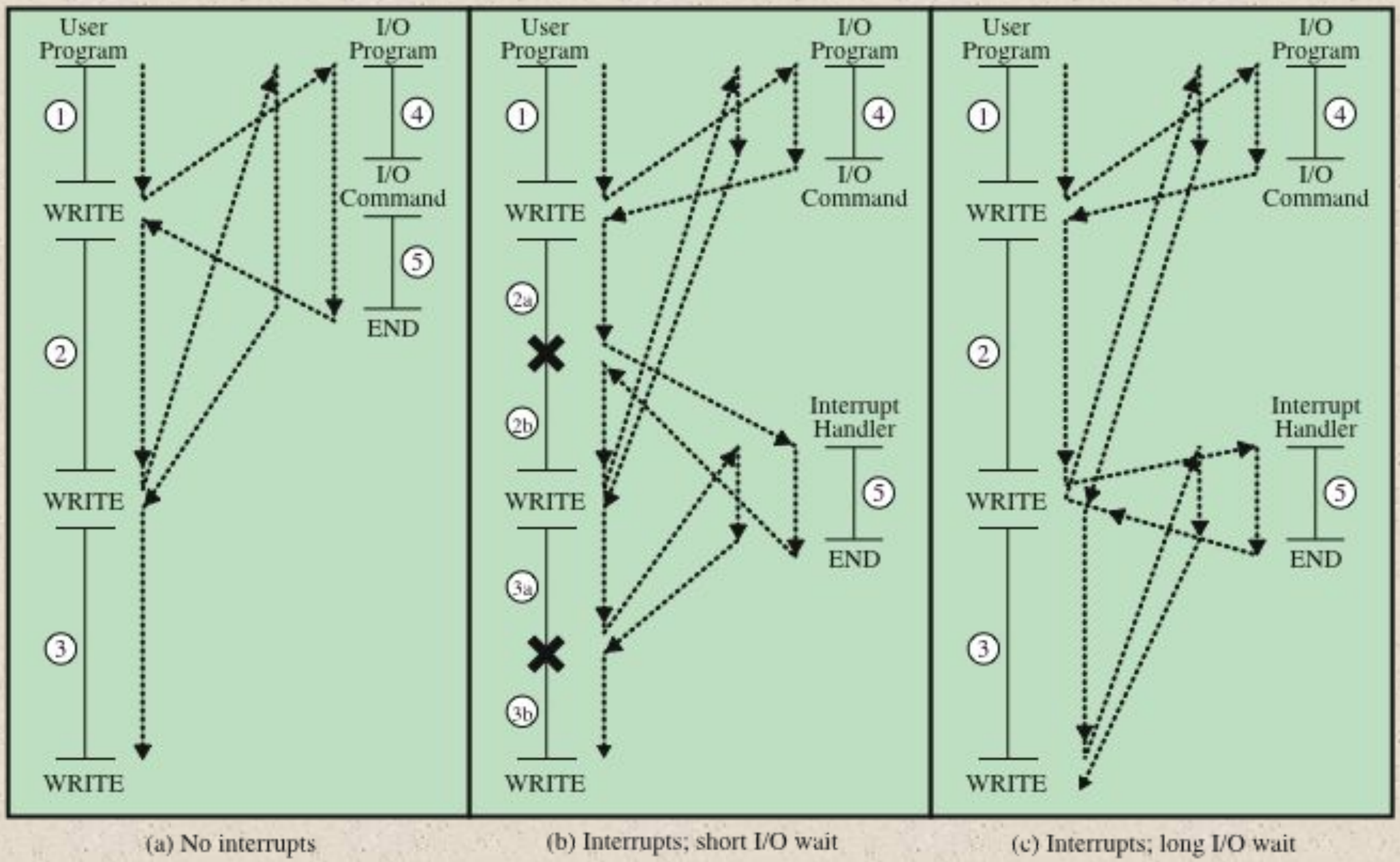
+

## Classes of Interrupts



| | |
|---|---|
| **Program** | Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, or reference outside a user's allowed memory space. |
| **Timer** | Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis. |
| **I/O** | Generated by an I/O controller, to signal normal completion of an operation, request service from the processor, or to signal a variety of error conditions. |
| **Hardware failure** | Generated by a failure such as power failure or memory parity error. |

# Program Flow Control



✖ = interrupt occurs during course of execution of user program

**Figure 3.7 Program Flow of Control Without and With Interrupts**

+ 
## Transfer of Control via Interrupts

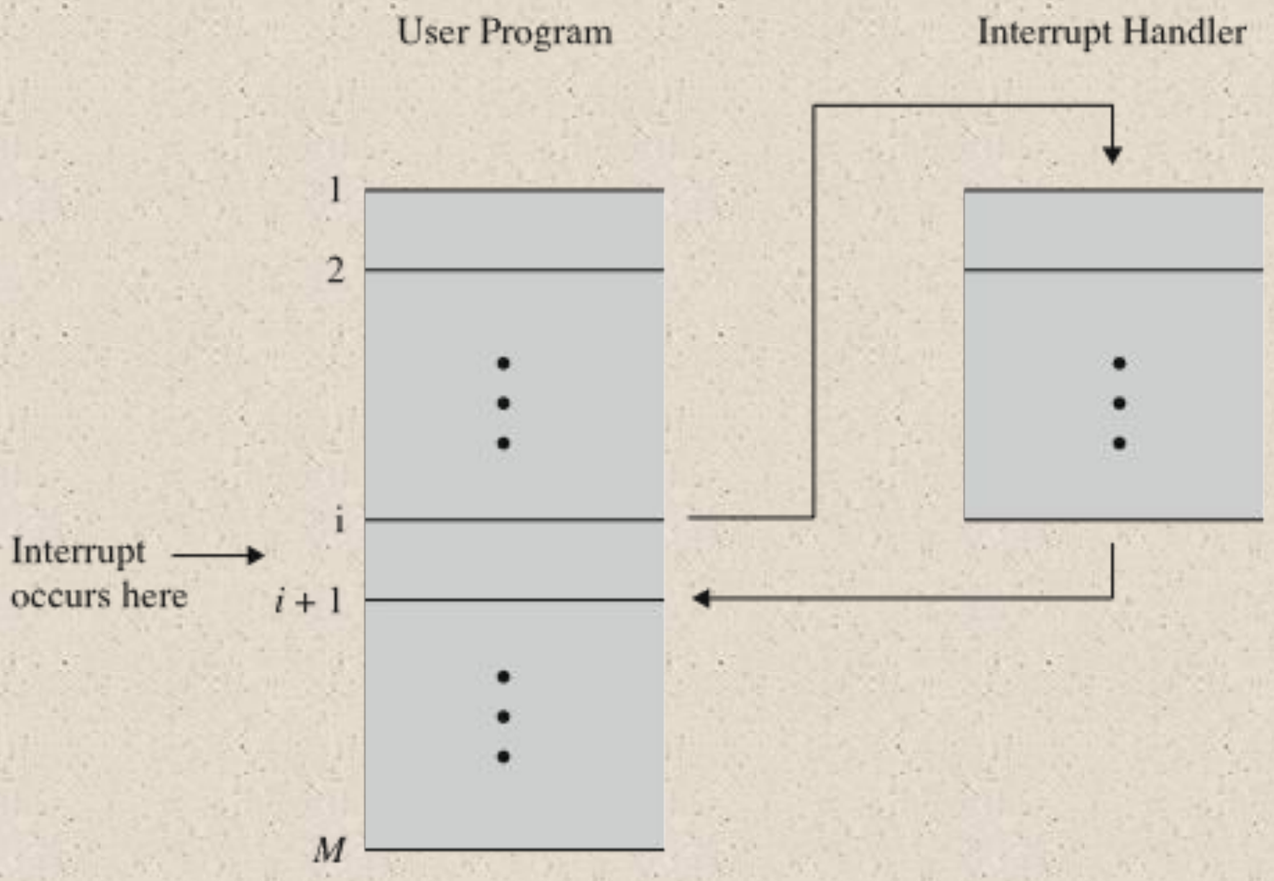


**Figure 3.8 Transfer of Control via Interrupts**

# Program Timing: Short I/O Wait

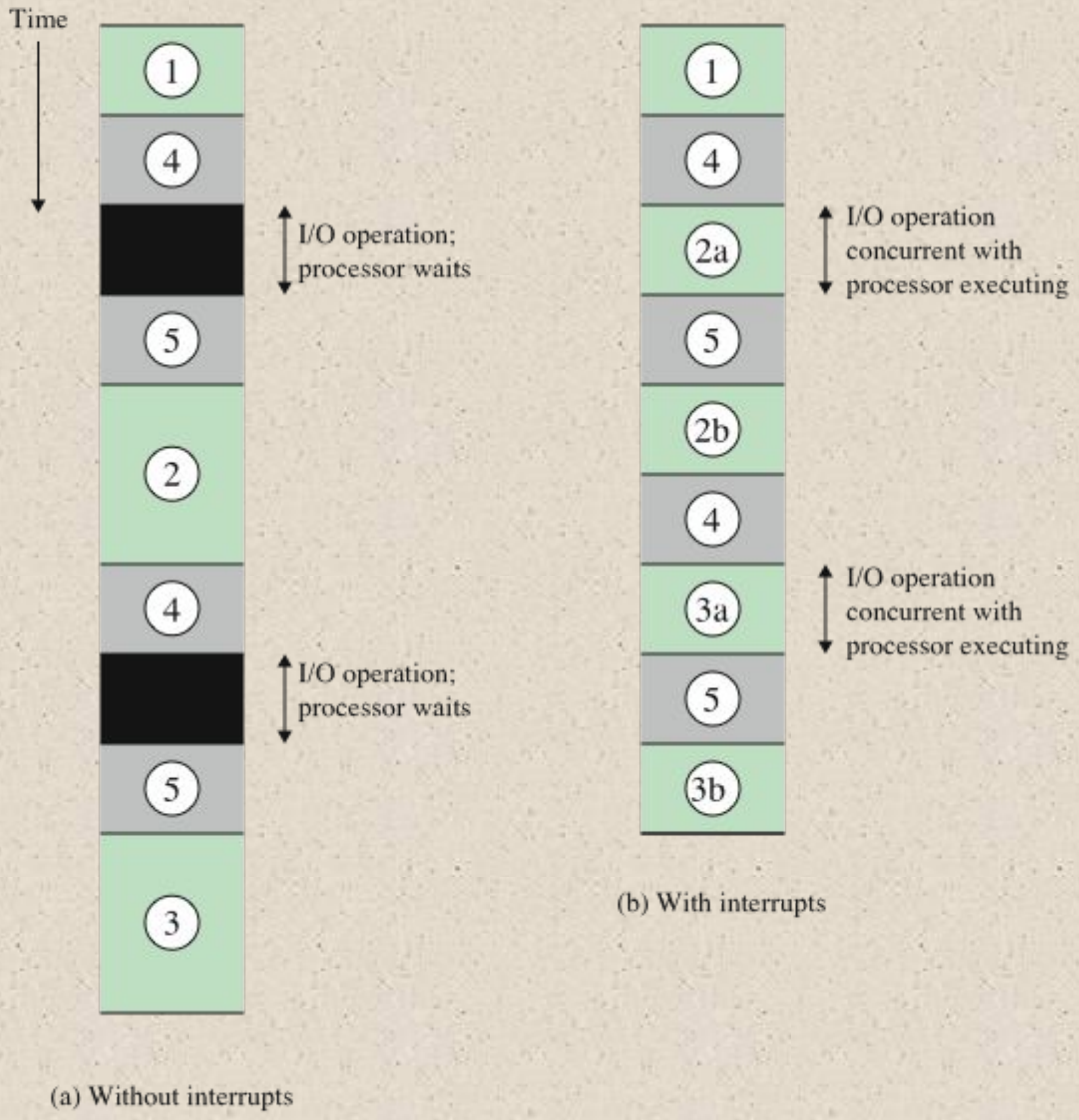

Time
1
4
I/O operation; processor waits
5
2
4
I/O operation; processor waits
5
3
(a) Without interrupts
1
4
2a
I/O operation concurrent with processor executing
5
2b
4
3a
I/O operation concurrent with processor executing
5
3b
(b) With interrupts

**Figure 3.10 Program Timing: Short I/O Wait**

# Program Timing: Long I/O Wait

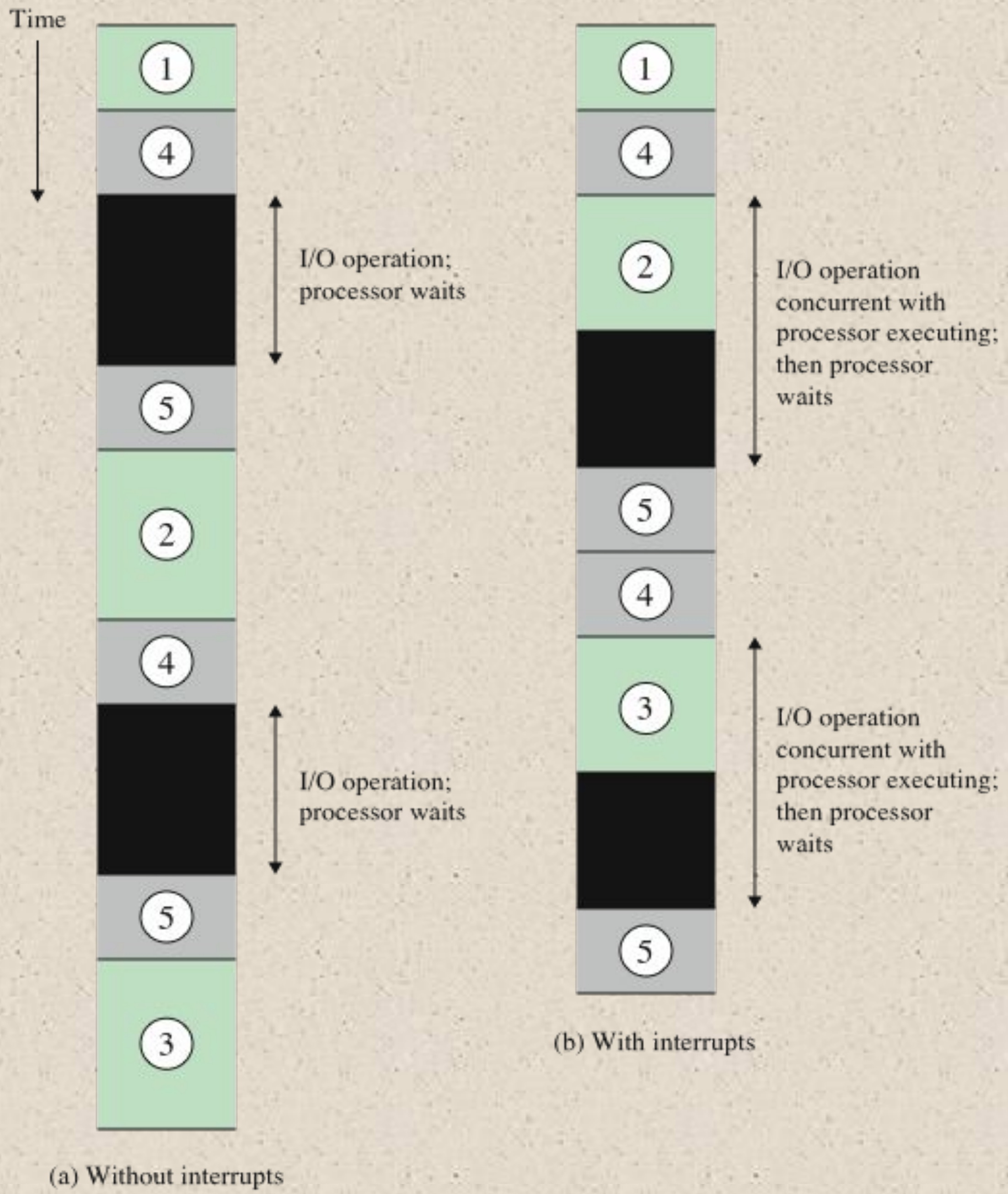

Time
1
4
I/O operation; processor waits
5
2
4
I/O operation; processor waits
5
3
(a) Without interrupts
1
4
2
I/O operation concurrent with processor executing; then processor waits
5
4
3
I/O operation concurrent with processor executing; then processor waits
5
(b) With interrupts

**Figure 3.11 Program Timing: Long I/O Wait**

+

# Instruction Cycle With Interrupts

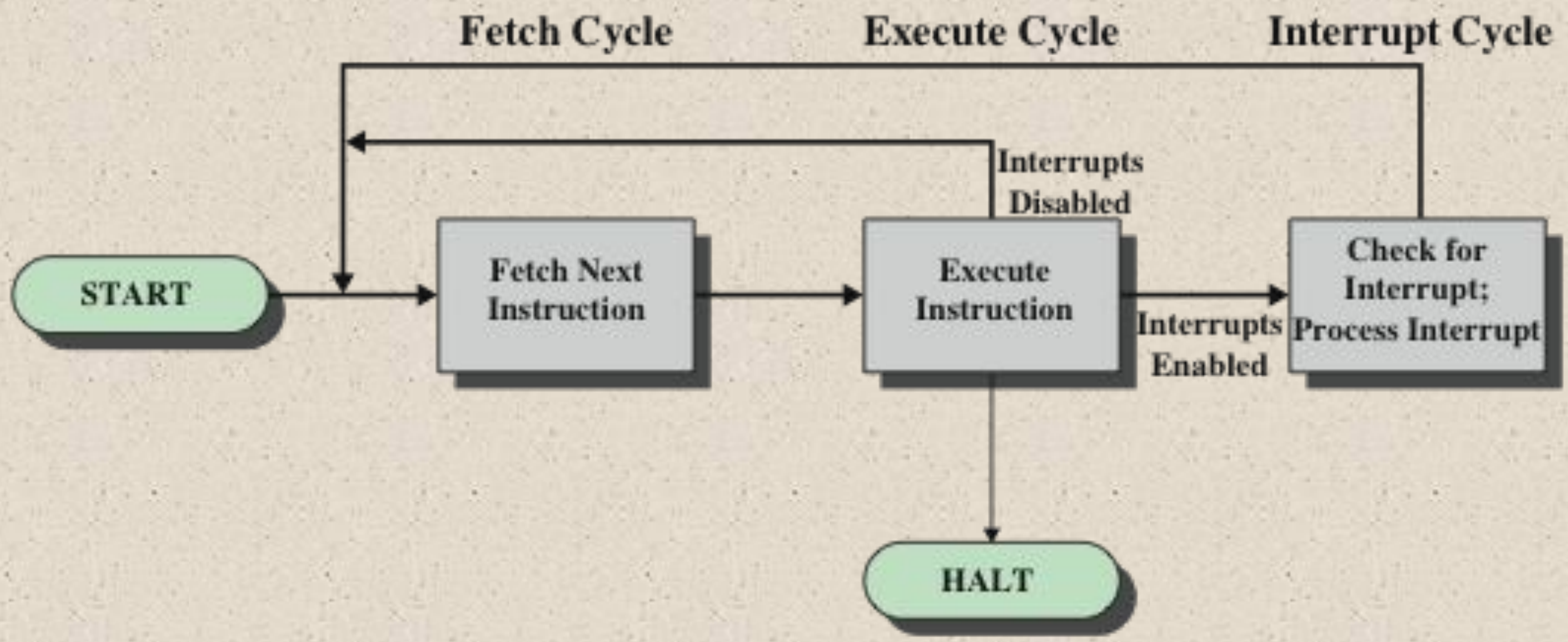


Figure 3.9 Instruction Cycle with Interrupts

# Instruction Cycle State Diagram With Interrupts

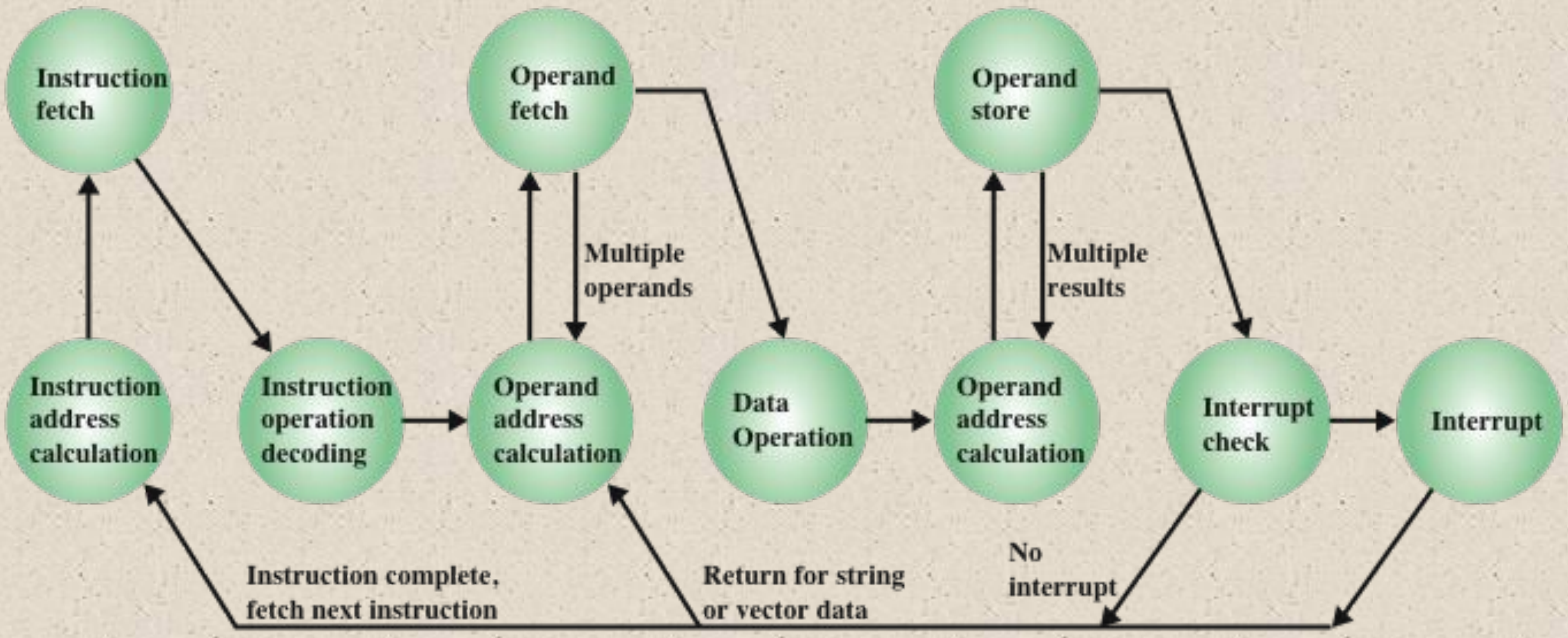
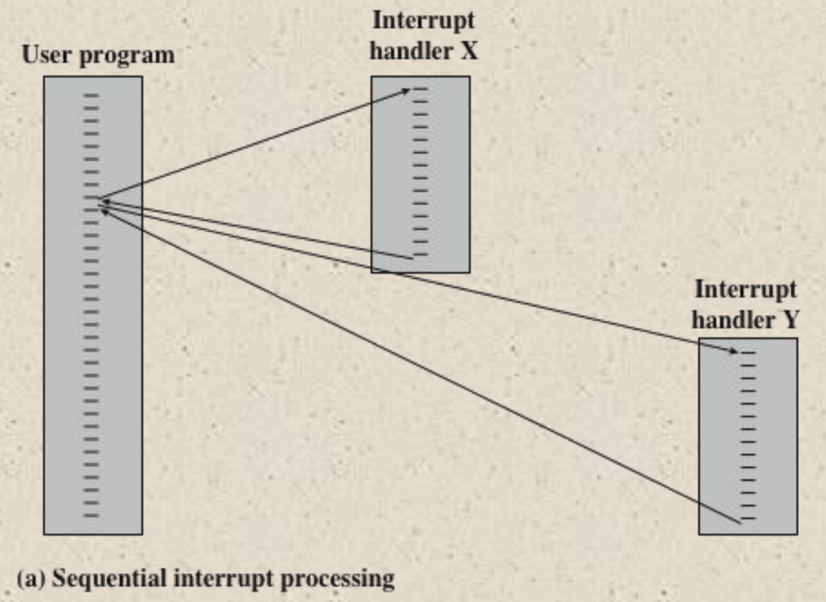

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Interrupt check
Interrupt
No interrupt
Instruction complete, fetch next instruction
Return for string or vector data

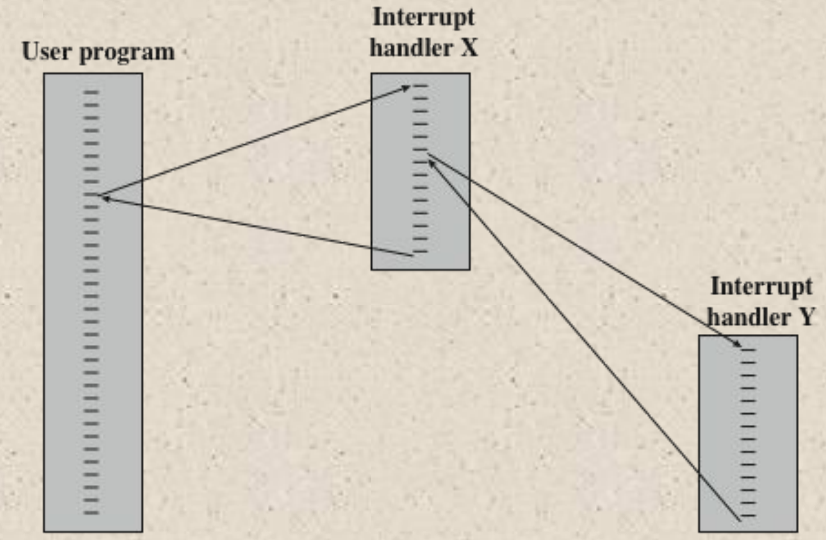
**Figure 3.12 Instruction Cycle State Diagram, With Interrupts**

## Transfer of Control

## Multiple Interrupts



(a) Sequential interrupt processing



(b) Nested interrupt processing

Figure 3.13 Transfer of Control with Multiple Interrupts

+ Time Sequence of Multiple Interrupts

Ex am ple

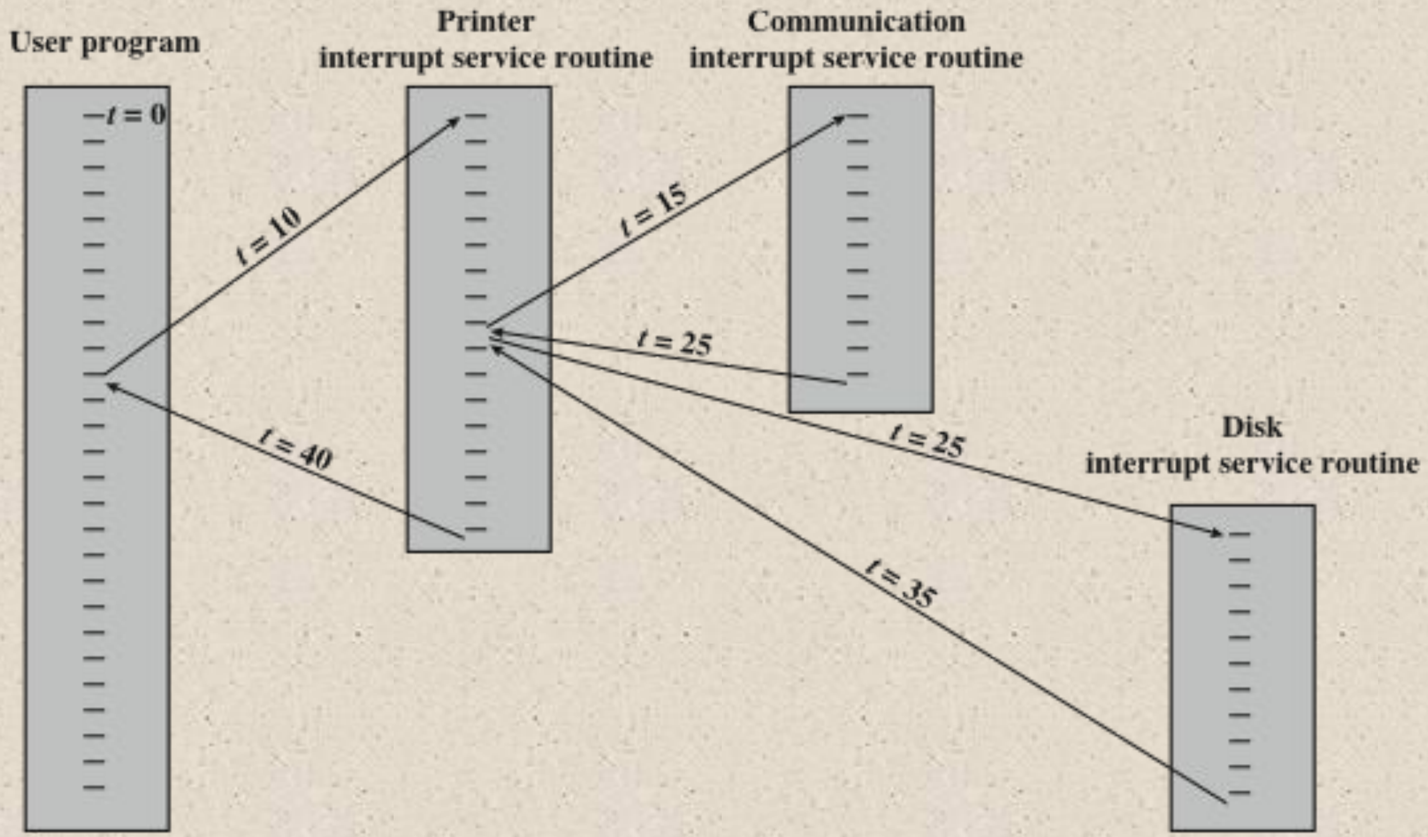

User program
Printer interrupt service routine
Communication interrupt service routine
Disk interrupt service routine
t = 0
t = 10
t = 15
t = 25
t = 25
t = 35
t = 40

Figure 3.14 Example Time Sequence of Multiple Interrupts

+

## I/O Function

- I/O module can exchange data directly with the processor
- Processor can read data from or write data to an I/O module
  - Processor identifies a specific device that is controlled by a particular I/O module
  - I/O instructions rather than memory referencing instructions
- In some cases it is desirable to allow I/O exchanges to occur directly with memory
  - The processor grants to an I/O module the authority to read from or write to memory so that the I/O memory transfer can occur without tying up the processor
  - The I/O module issues read or write commands to memory relieving the processor of responsibility for the exchange
  - This operation is known as direct memory access (DMA)

+

## Computer Modules

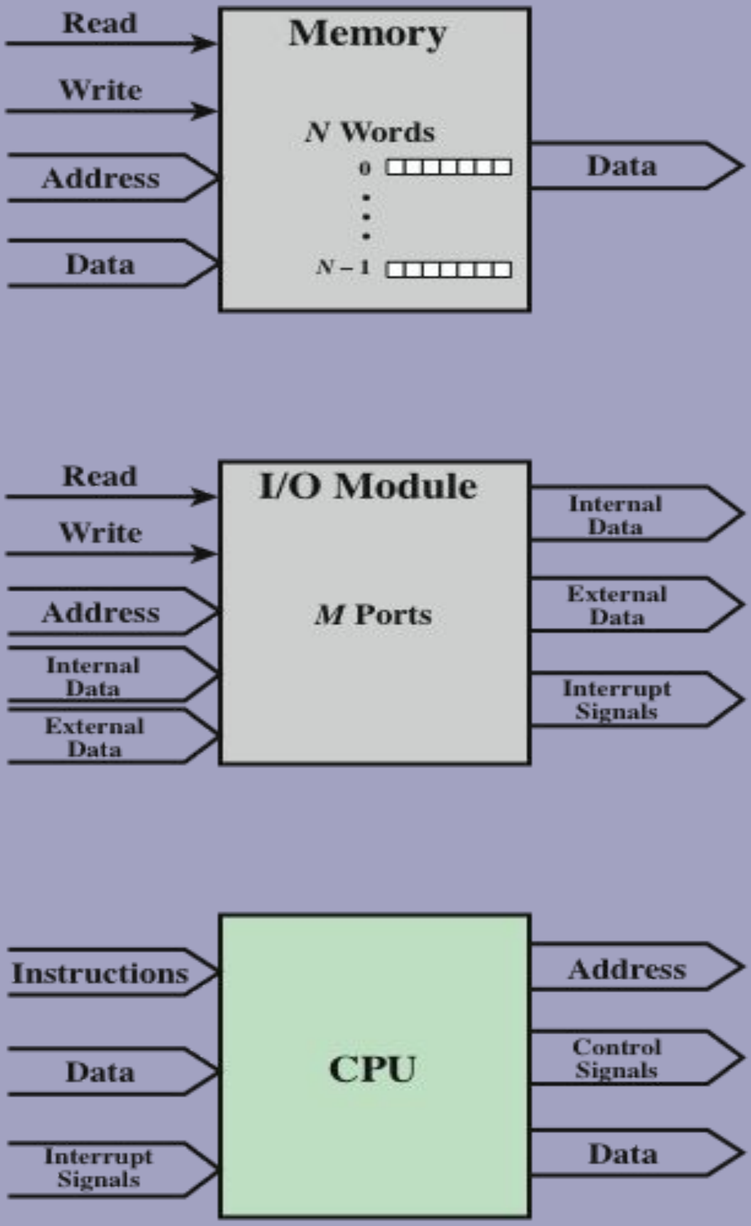


Figure 3.15 Computer Modules

## The interconnection structure must support the following types of transfers:

| Memory to processor | Processor to memory | I/O to processor | Processor to I/O | I/O to or from memory |
|---|---|---|---|---|
| Processor reads an instruction or a unit of data from memory | Processor writes a unit of data to memory | Processor reads data from an I/O device via an I/O module | Processor sends data to the I/O device | An I/O module is allowed to exchange data directly with memory without going through the processor using direct memory access |

+

## Bus Interconnection

A communication pathway connecting two or more devices

- Key characteristic is that it is a shared transmission medium

Signals transmitted by any one device are available for reception by all other devices attached to the bus

- If two devices transmit during the same time period their signals will overlap and become garbled

Typically consists of multiple communication lines

- Each line is capable of transmitting signals representing binary 1 and binary 0

Computer systems contain a number of different buses that provide pathways between components at various levels of the computer system hierarchy

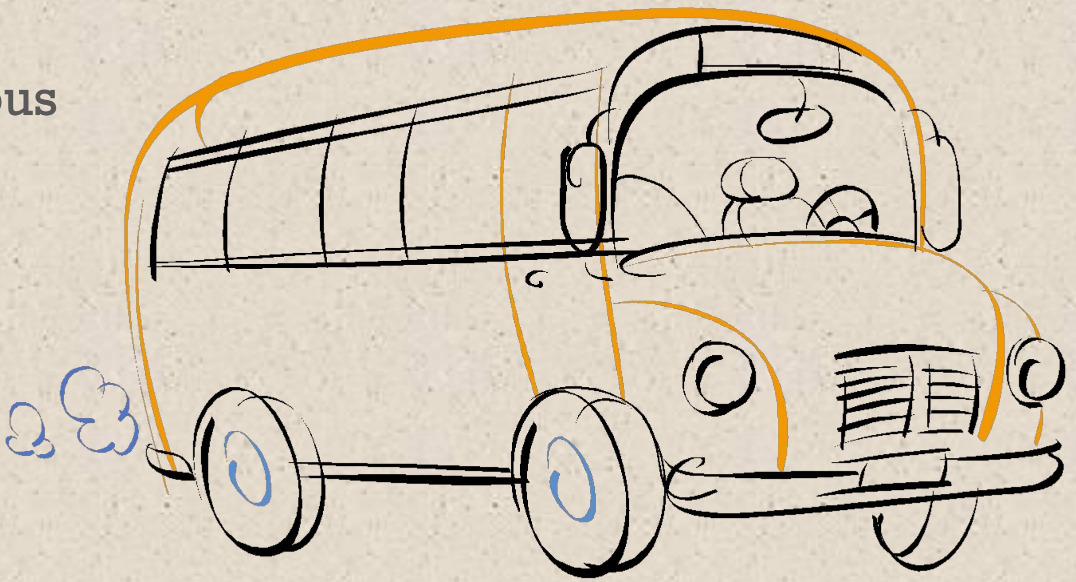
*System bus*

- A bus that connects major computer components (processor, memory, I/O)

The most common computer interconnection structures are based on the use of one or more system buses

# Data Bus

- Data lines that provide a path for moving data among system modules
- May consist of 32, 64, 128, or more separate lines
- The number of lines is referred to as the *width* of the data bus
- The number of lines determines how many bits can be transferred at a time
- The width of the data bus is a key factor in determining overall system performance



## Address Bus

- Used to designate the source or destination of the data on the data bus
- If the processor wishes to read a word of data from memory it puts the address of the desired word on the address lines
- Width determines the maximum possible memory capacity of the system
- Also used to address I/O ports
- The higher order bits are used to select a particular module on the bus and the lower order bits select a memory location or I/O port within the module

## Control Bus

- Used to control the access and the use of the data and address lines
- Because the data and address lines are shared by all components there must be a means of controlling their use
- Control signals transmit both command and timing information among system modules
- Timing signals indicate the validity of data and address information
- Command signals specify operations to be performed

## Bus Interconnection Scheme

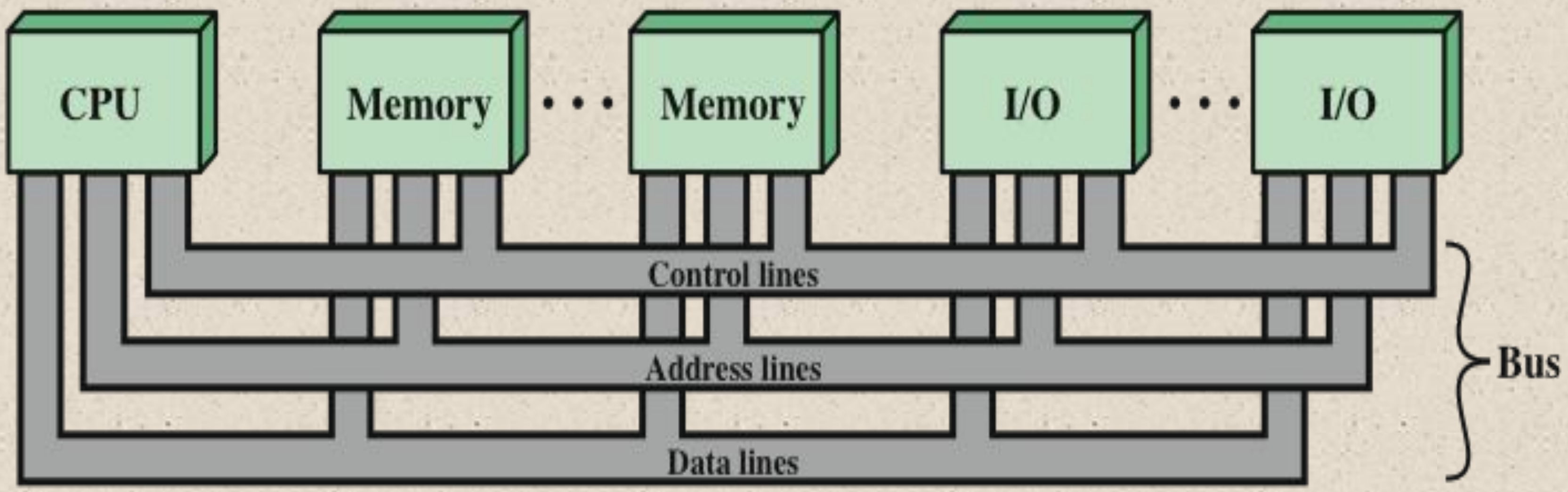
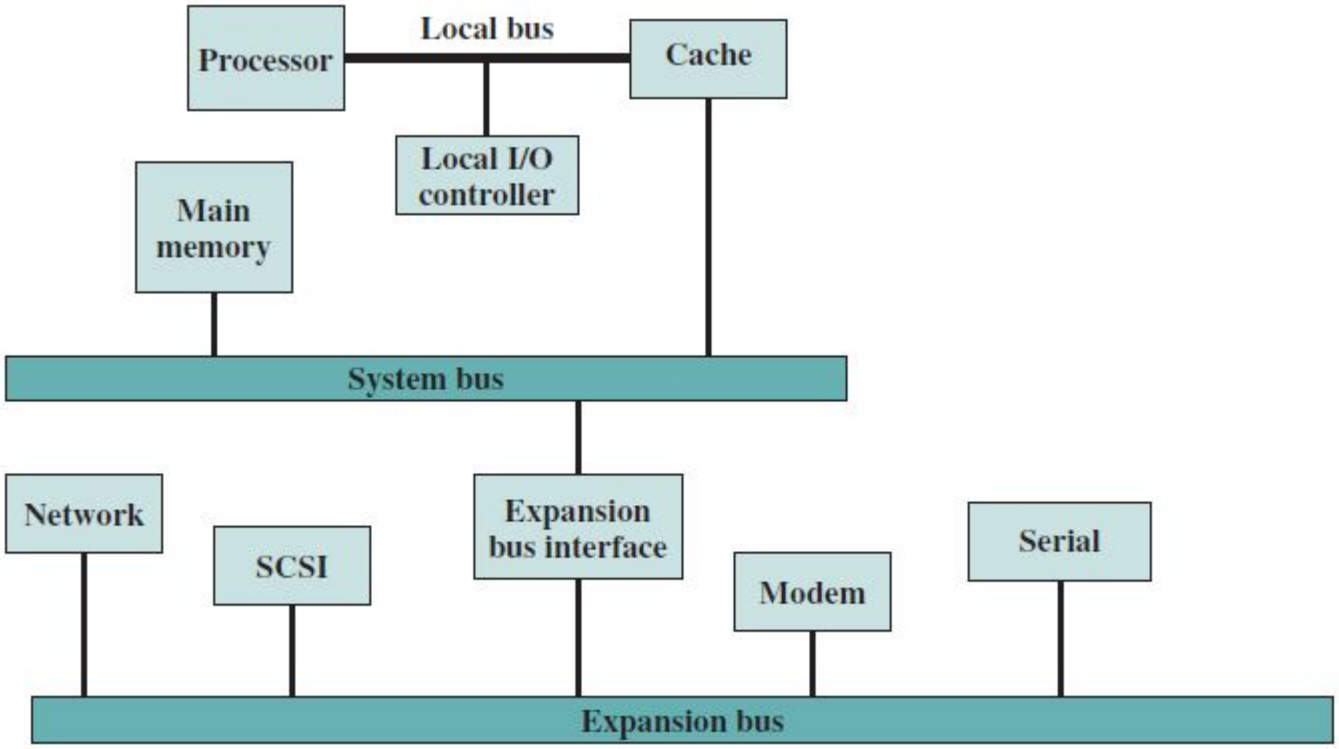


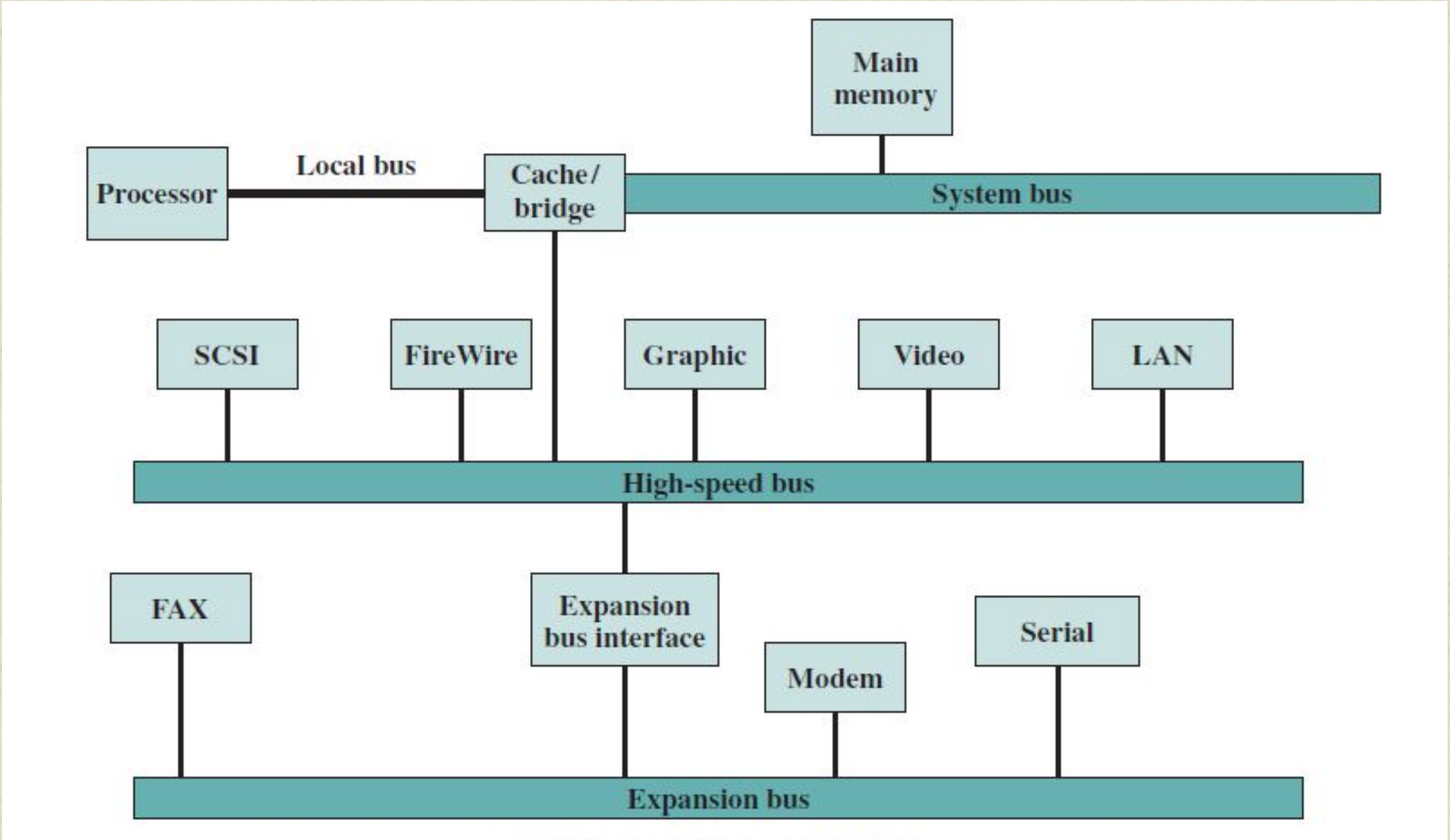
Figure 3.16 Bus Interconnection Scheme

+ 
## Bus Configurations



(a) Traditional bus architecture

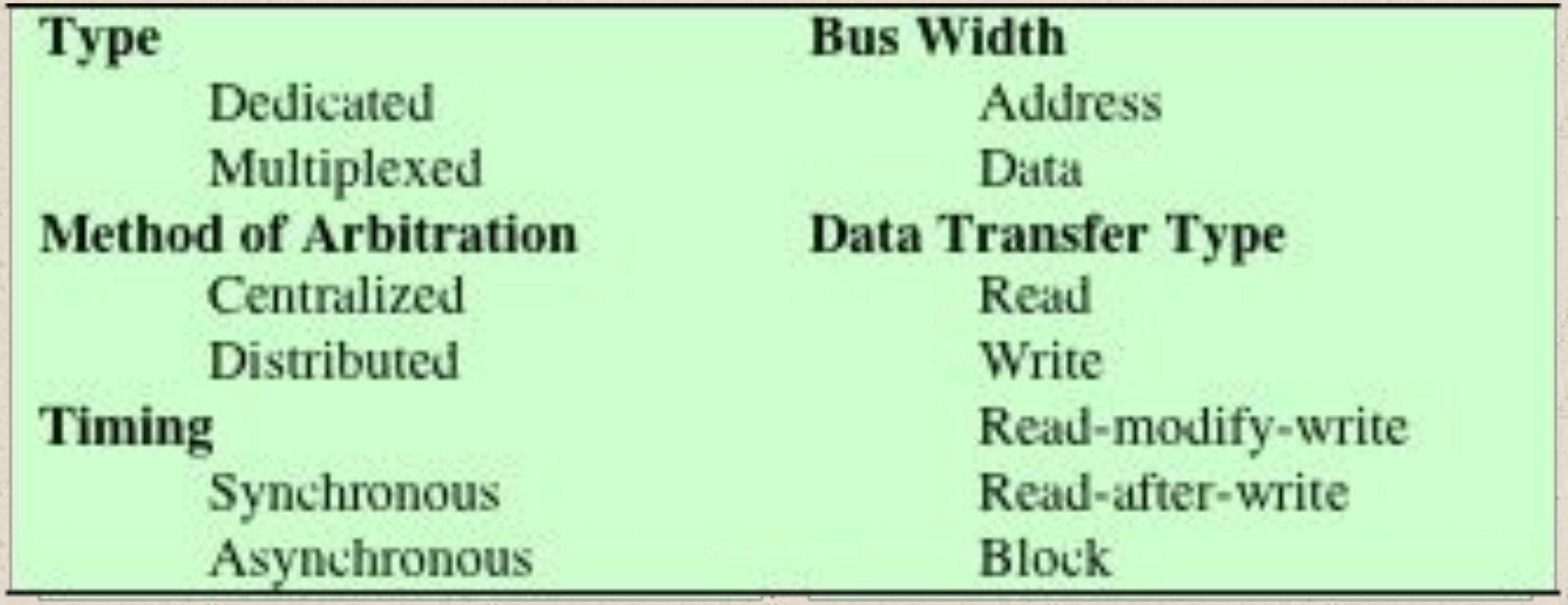
+
## Bus Configurations



(b) High-performance architecture

+

## Elements of Bus Design



**Type**
- Dedicated
- Multiplexed

**Method of Arbitration**
- Centralized
- Distributed

**Timing**
- Synchronous
- Asynchronous

**Bus Width**
- Address
- Data

**Data Transfer Type**
- Read
- Write
- Read-modify-write
- Read-after-write
- Block