CSE 213
Computer Architecture

Lecture 3
A Top-Level View of Computer Function and Interconnection

Military Institute of Science and Technology

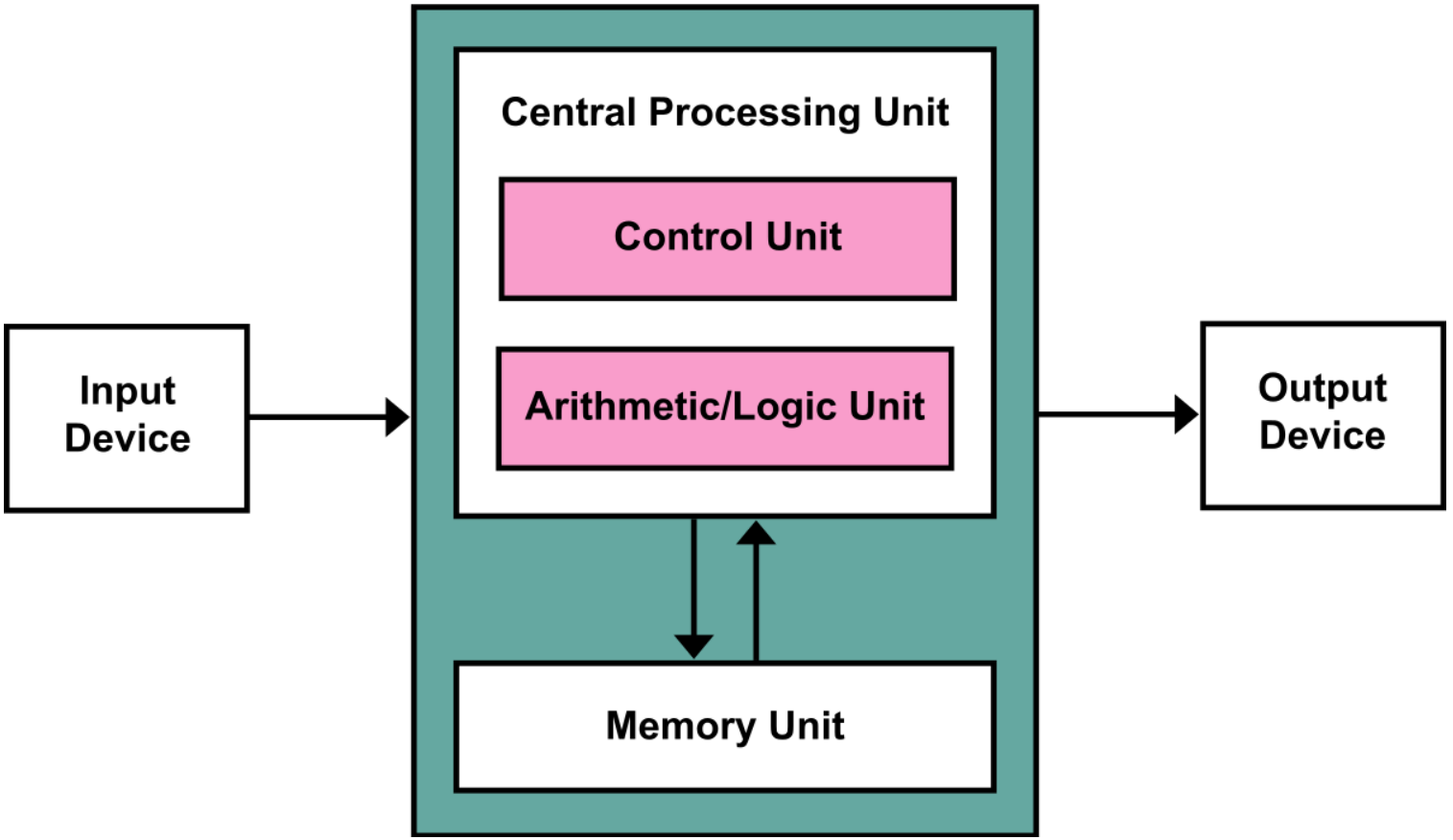
Outline

- Computer Components
- Computer Functions
- Interconnection Structures
- Bus Interconnection

Computer Components

- Contemporary computer designs are based on concepts developed by John von Neumann at the Institute for Advanced Studies, Princeton, referred to as the *von Neumann architecture* and is based on three key concepts:
  - Data and instructions are stored in a single read-write memory
  - The contents of this memory are addressable by location, without regard to the type of data contained there
  - Execution occurs in a sequential fashion (unless explicitly modified) from one instruction to the next
- *Hardwired program*
  - If there is a particular computation to be performed, a configuration of logic components designed specifically for that computation could be constructed. The process of connecting the various components in the desired configuration as a form of programming. The resulting "program" is in the form of hardware and is termed a hardwired program.

von Neumann architecture



Hardware and Software Approaches

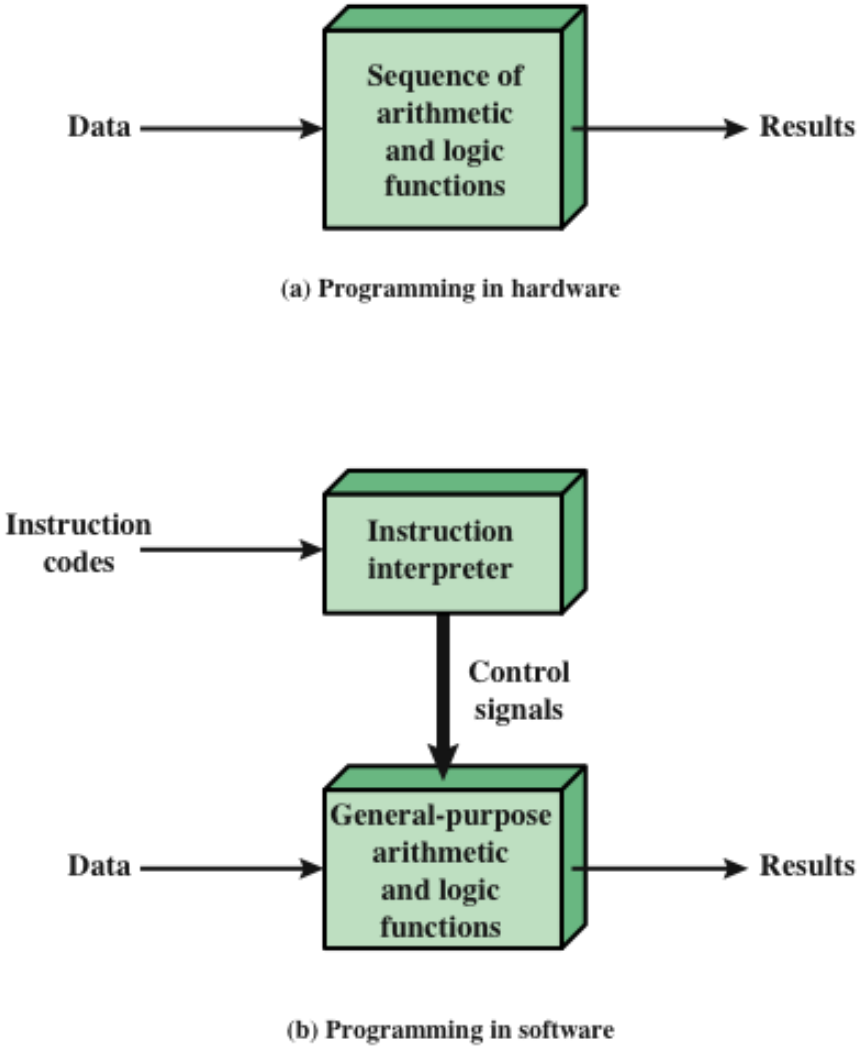


Figure 3.1 Hardware and Software Approaches

Software

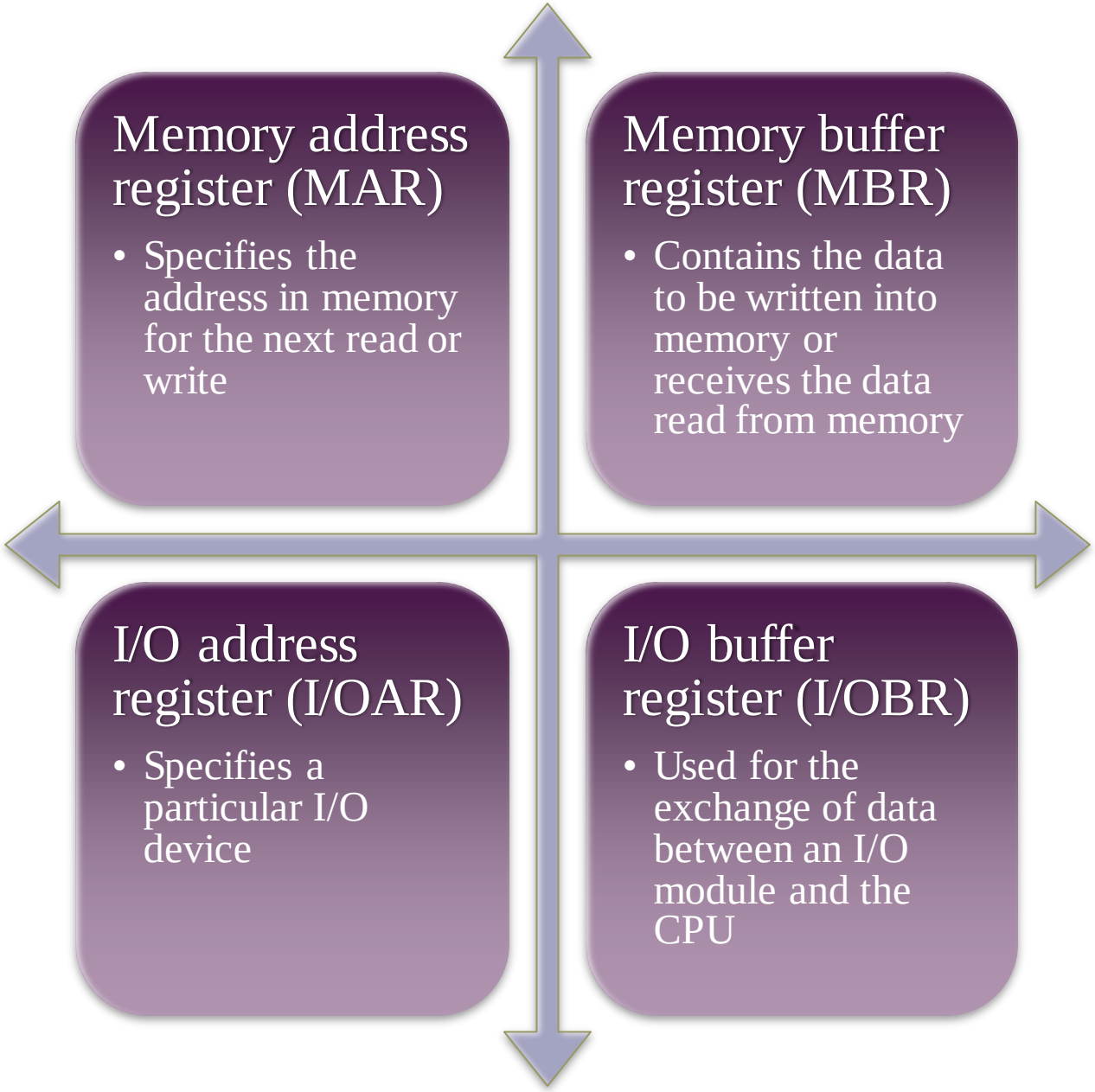
- A sequence of codes or instructions
- Part of the hardware interprets each instruction and generates control signals
- Provide a new sequence of codes for each new program instead of rewiring the hardware

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

Software

I/O Components



MEMORY

MAR

MBR

Computer Components: Top Level View

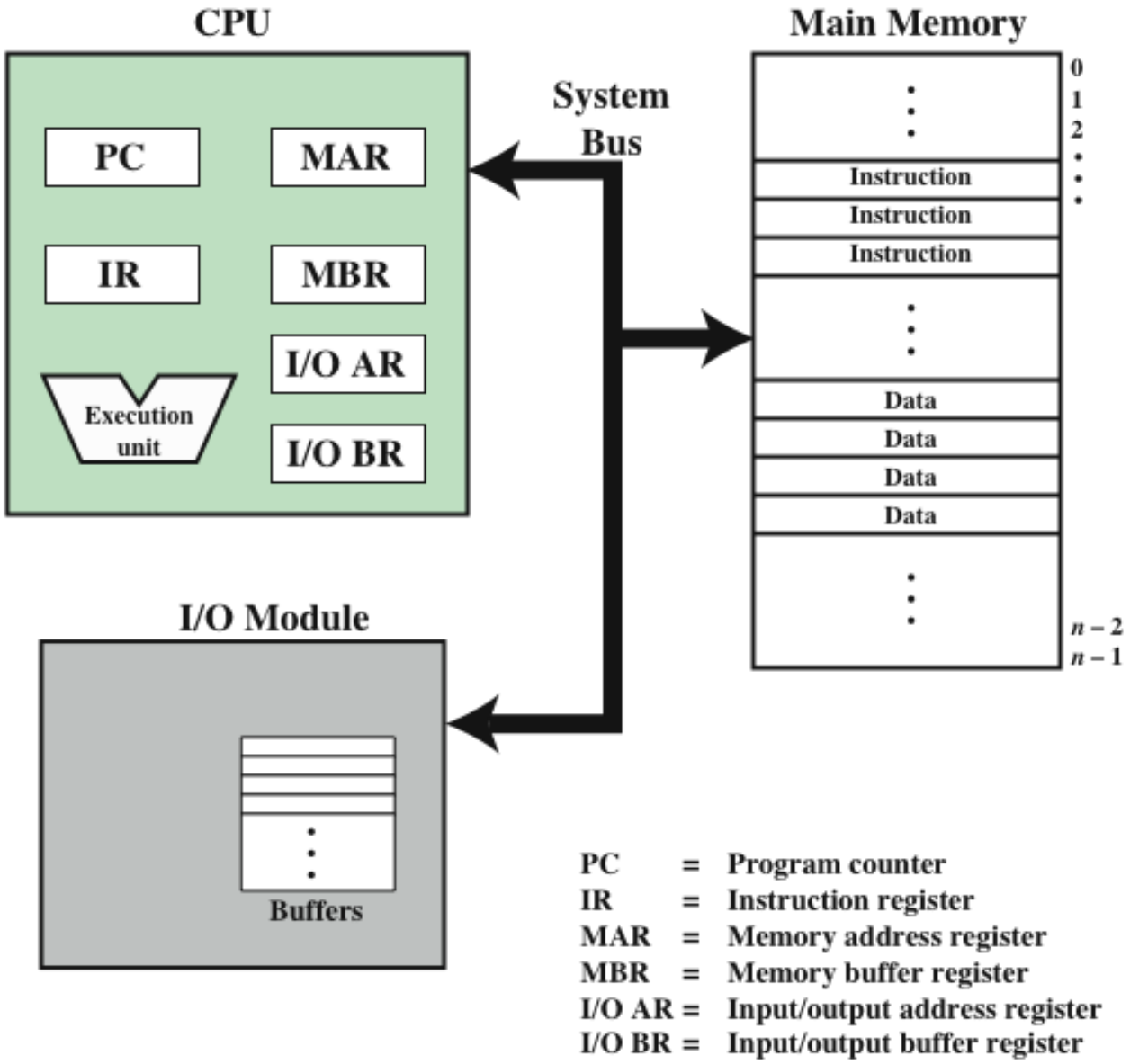


Figure 3.2 Computer Components: Top-Level View

Computer Function

- The basic function performed by a computer is execution of a program, which consists of a set of instructions stored in memory.
- Instruction processing consists of two steps:
  - The processor reads (fetches) instructions from memory one at a time and
  - executes each instruction

Basic Instruction Cycle

The processing required for a single instruction is called an **instruction cycle.**

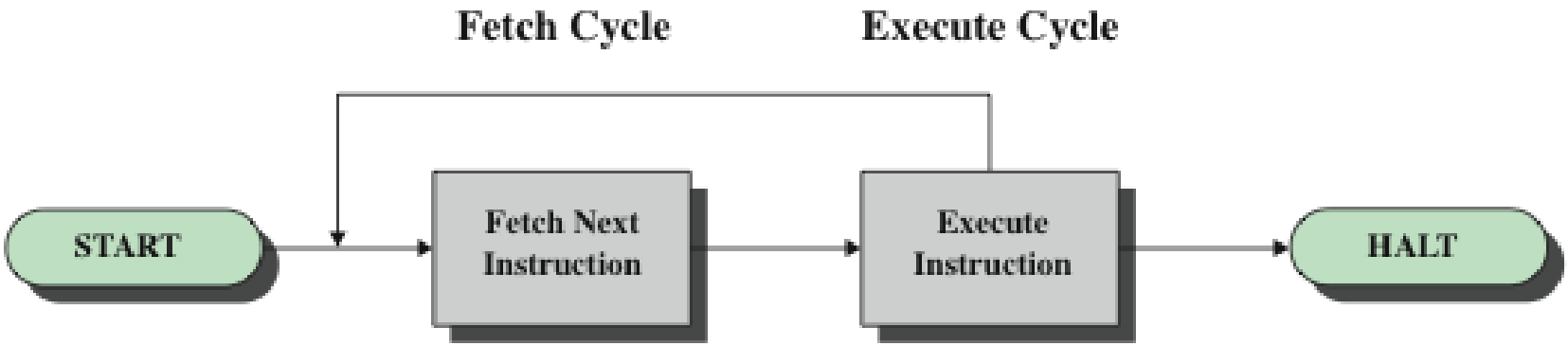


Figure 3.3 Basic Instruction Cycle

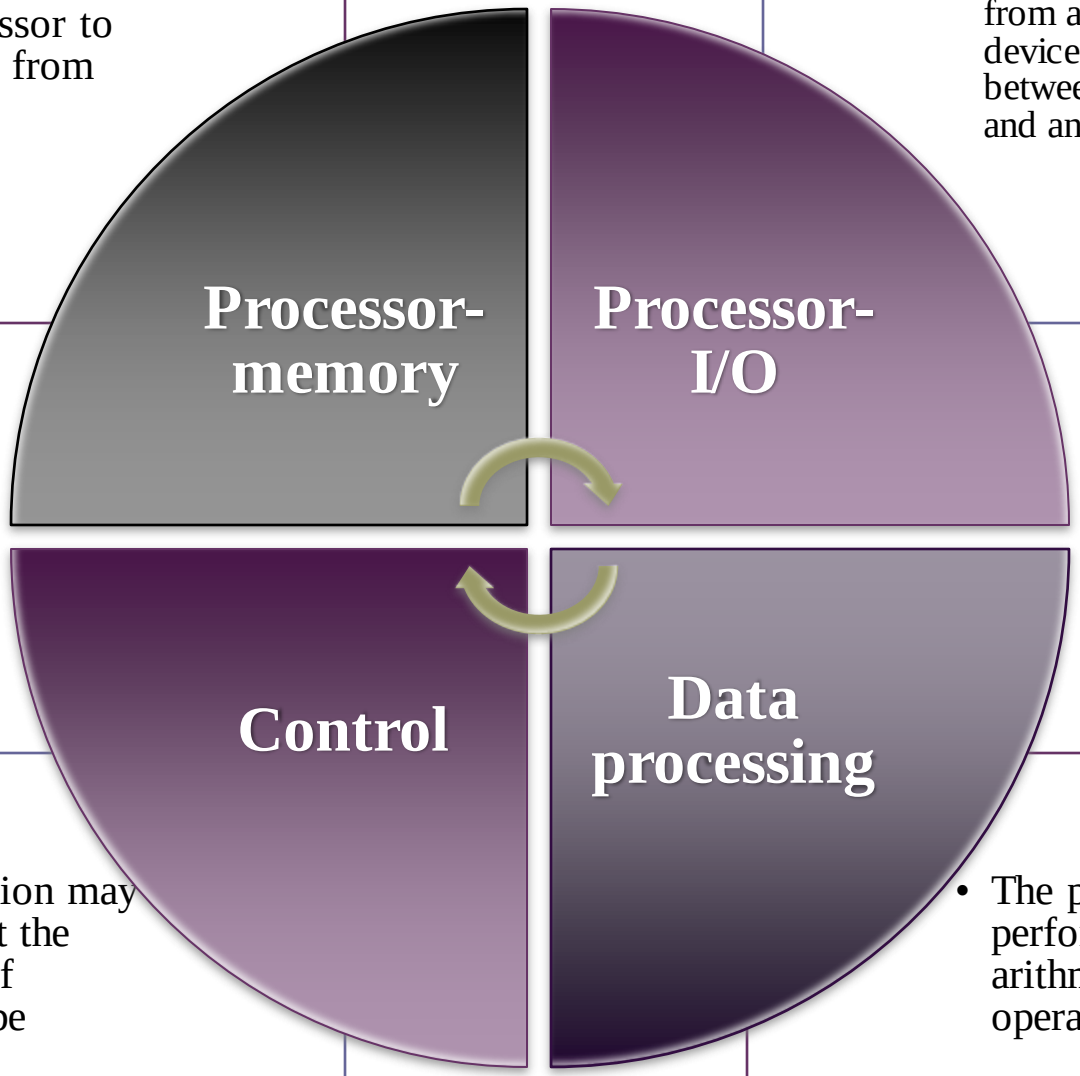
Fetch Cycle

- 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)
- The processor interprets the instruction and performs the required **action**

Action Categories

- Data transferred from processor to memory or from memory to processor

- Data transferred to or from a peripheral device by transferring between the processor and an I/O module



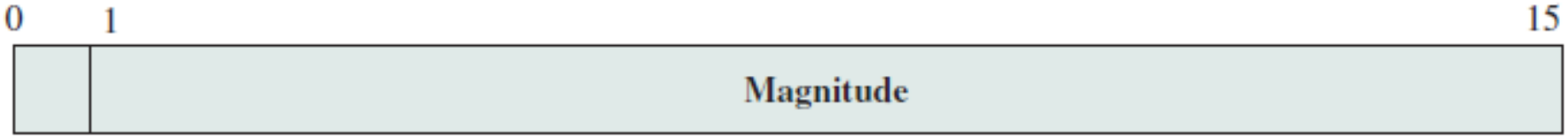
- An instruction may specify that the sequence of execution be altered

- The processor may perform some arithmetic or logic operation on data

Hypothetical machine: Example

- Consider a simple example using a hypothetical machine that includes the
  - The processor contains a single **data register**, called an accumulator (AC).
  - Both **instructions and data** are 16 bits long. Thus, it is convenient to organize memory using 16-bit words.
  - The instruction format provides **4 bits** for the **opcode**, so that there can be as many as 2^4 = 16 different opcodes, and up to 2^12 = 4096 (4K) words of memory can be directly addressed.

(a) Instruction format



(b) Integer format

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

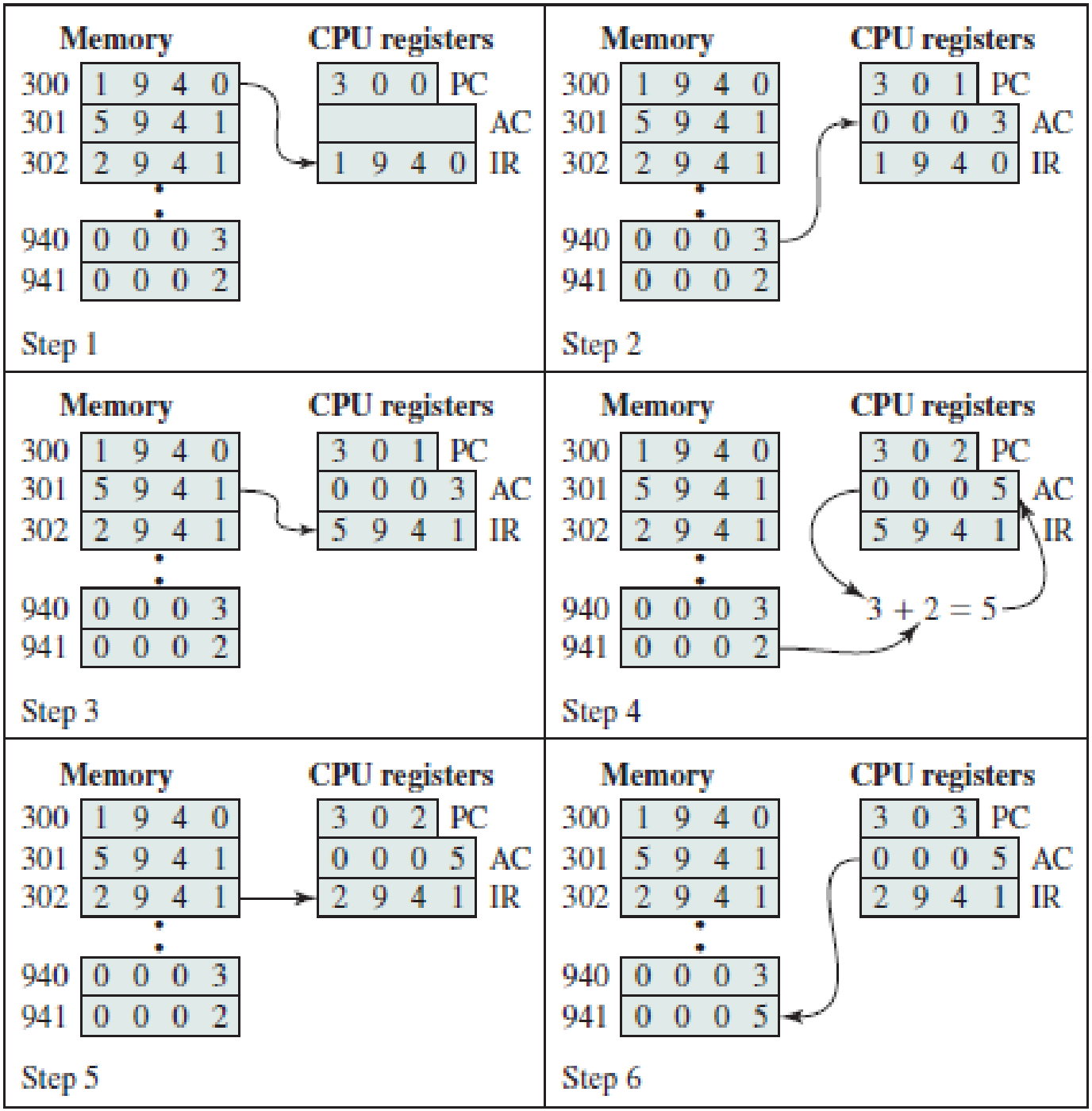
(c) Internal CPU registers

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



Instruction Cycle State Diagram

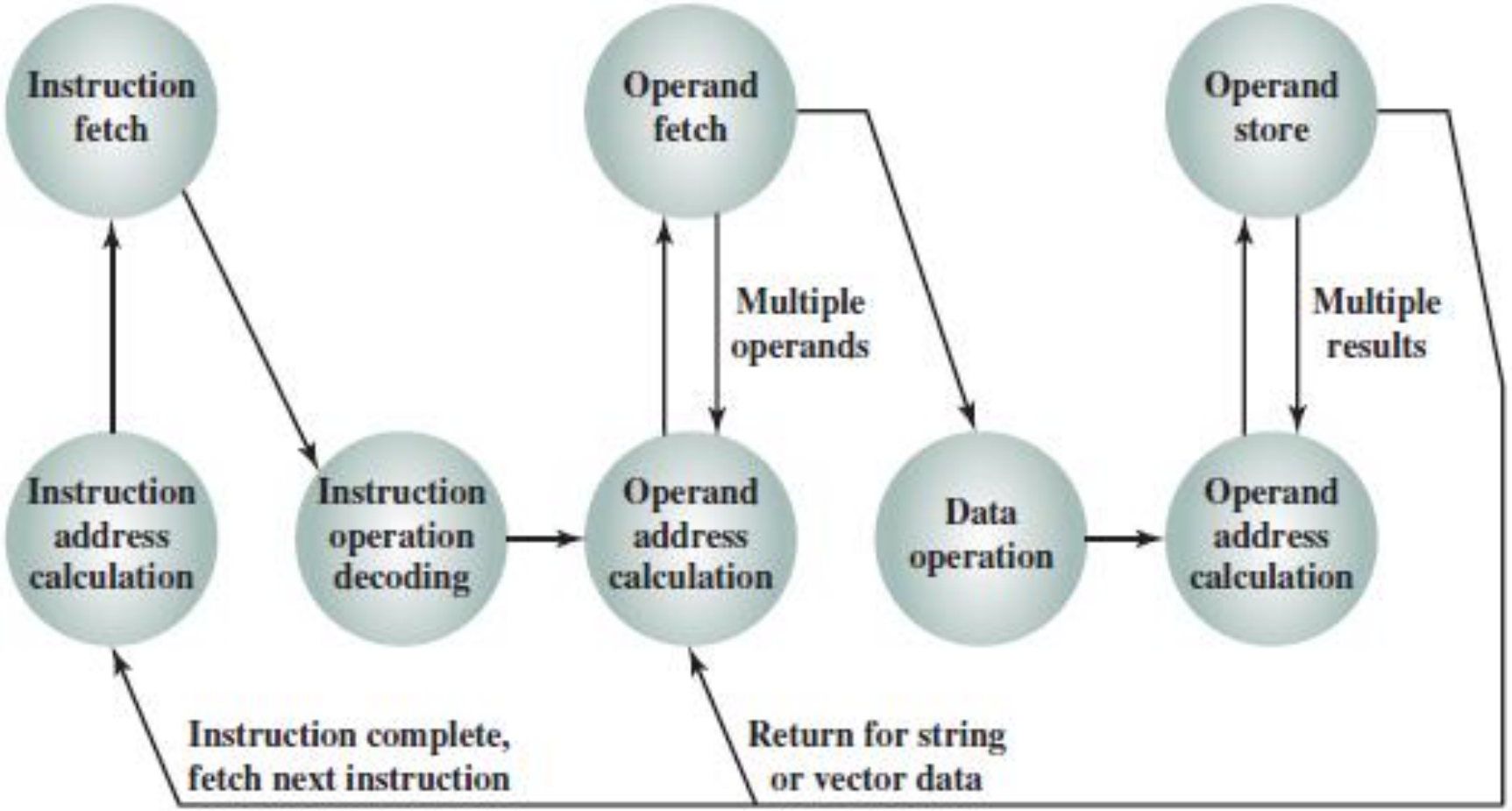


Figure 3.6 Instruction Cycle State Diagram

Instruction Cycle State Diagram

- **Instruction address calculation (iac):** Determine the address of the next instruction to be executed. Usually, this involves adding a fixed number to the address of the previous instruction.
- **Instruction fetch (if):** Read instruction from its memory location into the processor.
- **Instruction operation decoding (iod):** Analyze instruction to determine type of operation to be performed and operand(s) to be used.
- **Operand address calculation (oac):** If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.

Instruction Cycle State Diagram

- **Operand fetch (of):** Fetch the operand from memory or read it in from I/O.
- **Data operation (do):** Perform the operation indicated in the instruction.
- **Operand store (os):** Write the result into memory or out to I/O.

Example: PDP-11 instruction **ADD A,B**

Sequence of states: iac, if, iod, oac, of, oac, of, do, oac, os

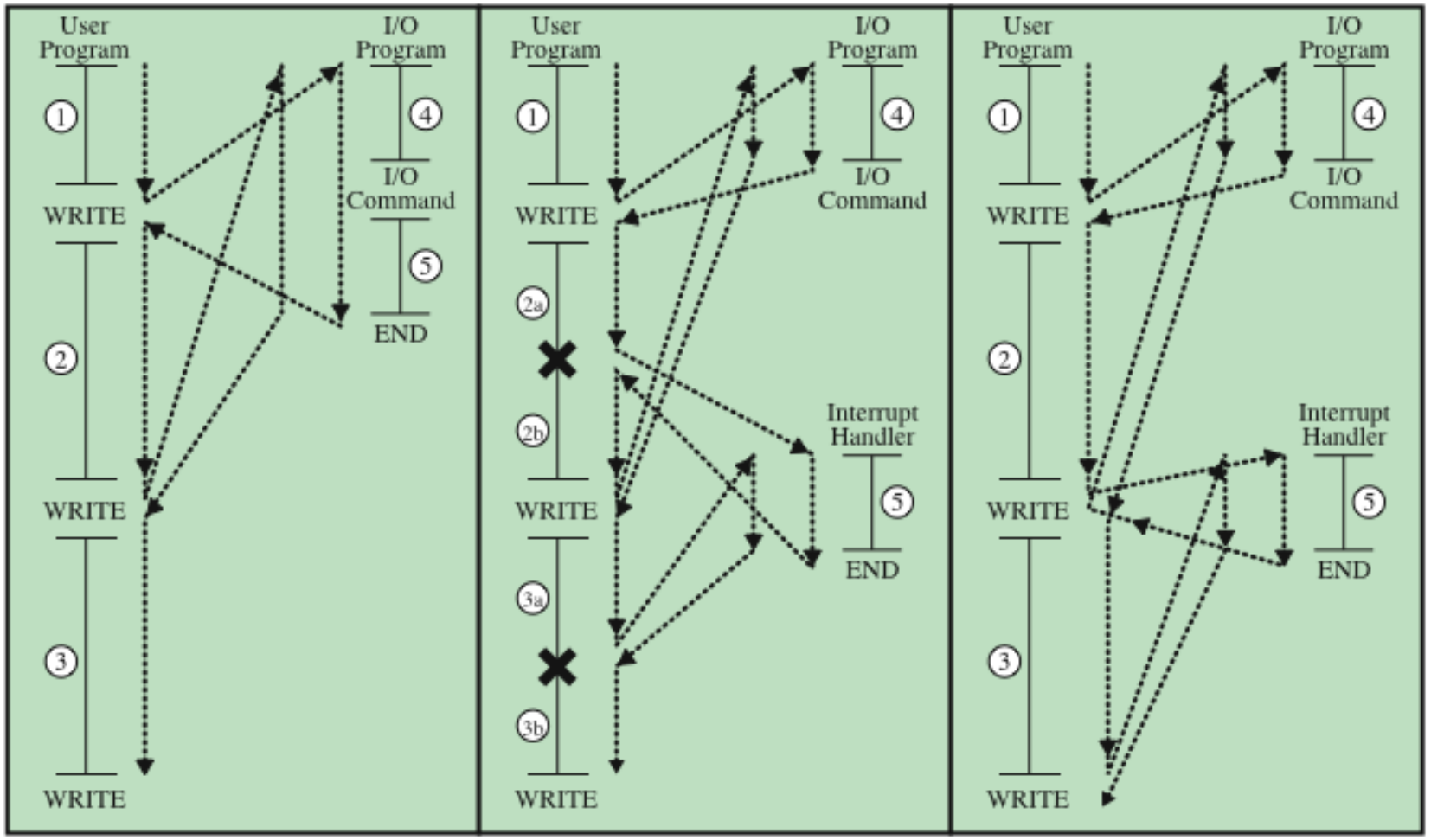
Classes of Interrupts

All computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal processing of the processor.

Table 3.1 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



(a) No interrupts

(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait

✖ = interrupt occurs during course of execution of user program

Figure 3.7 Program Flow of Control Without and With Interrupts

Program Timing: Short I/O Wait

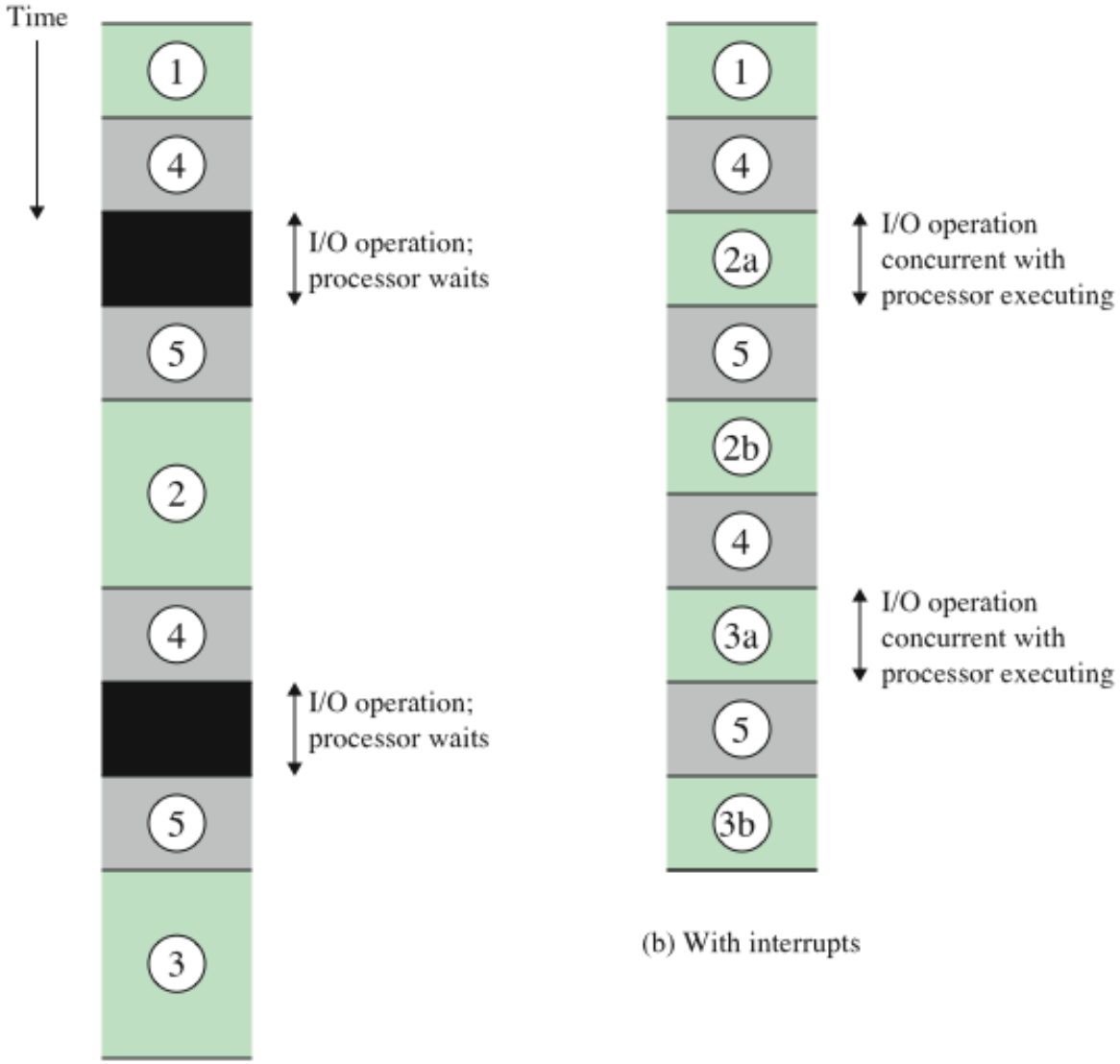


Figure 3.10 Program Timing: Short I/O Wait

Program Timing: Long I/O Wait

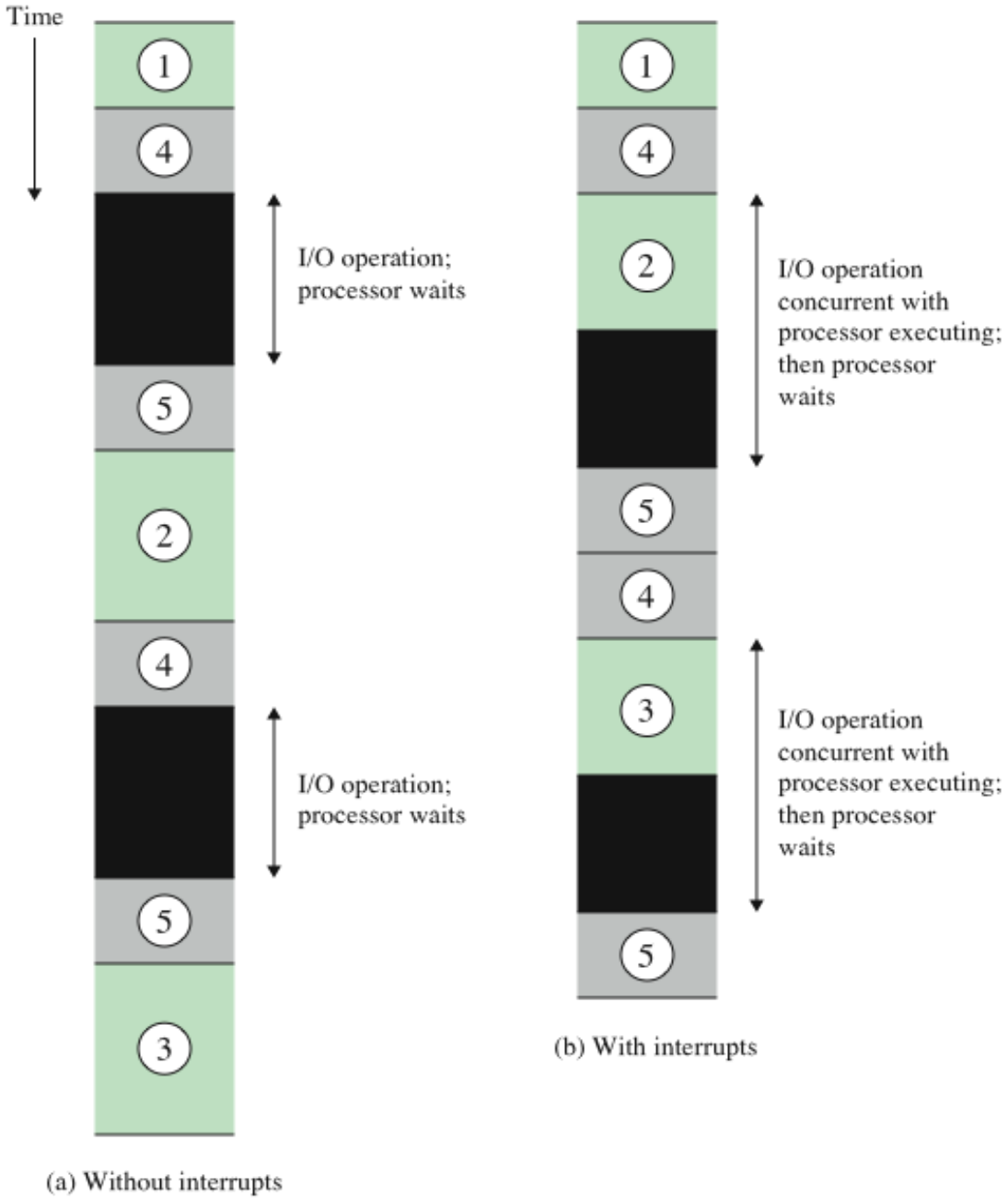


Figure 3.11 Program Timing: Long I/O Wait

Instruction Cycle With Interrupts

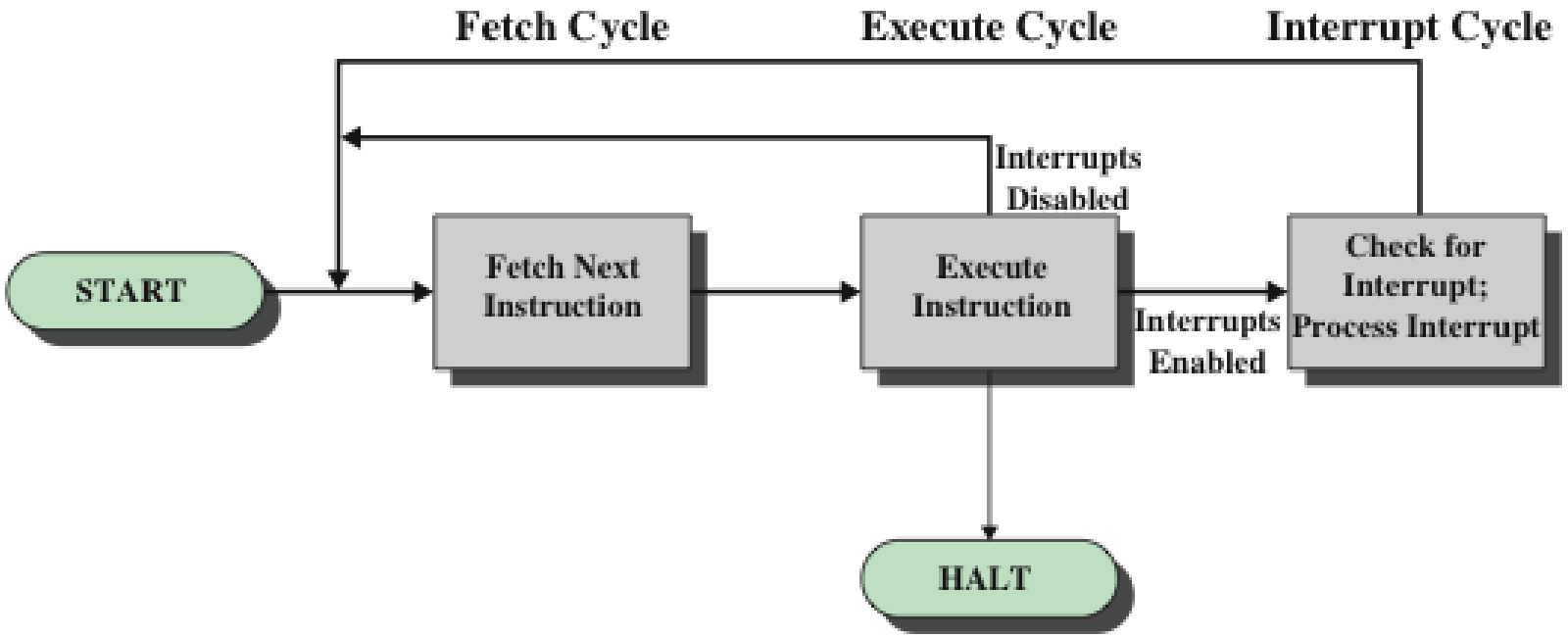


Figure 3.9 Instruction Cycle with Interrupts

Instruction Cycle State Diagram With Interrupts

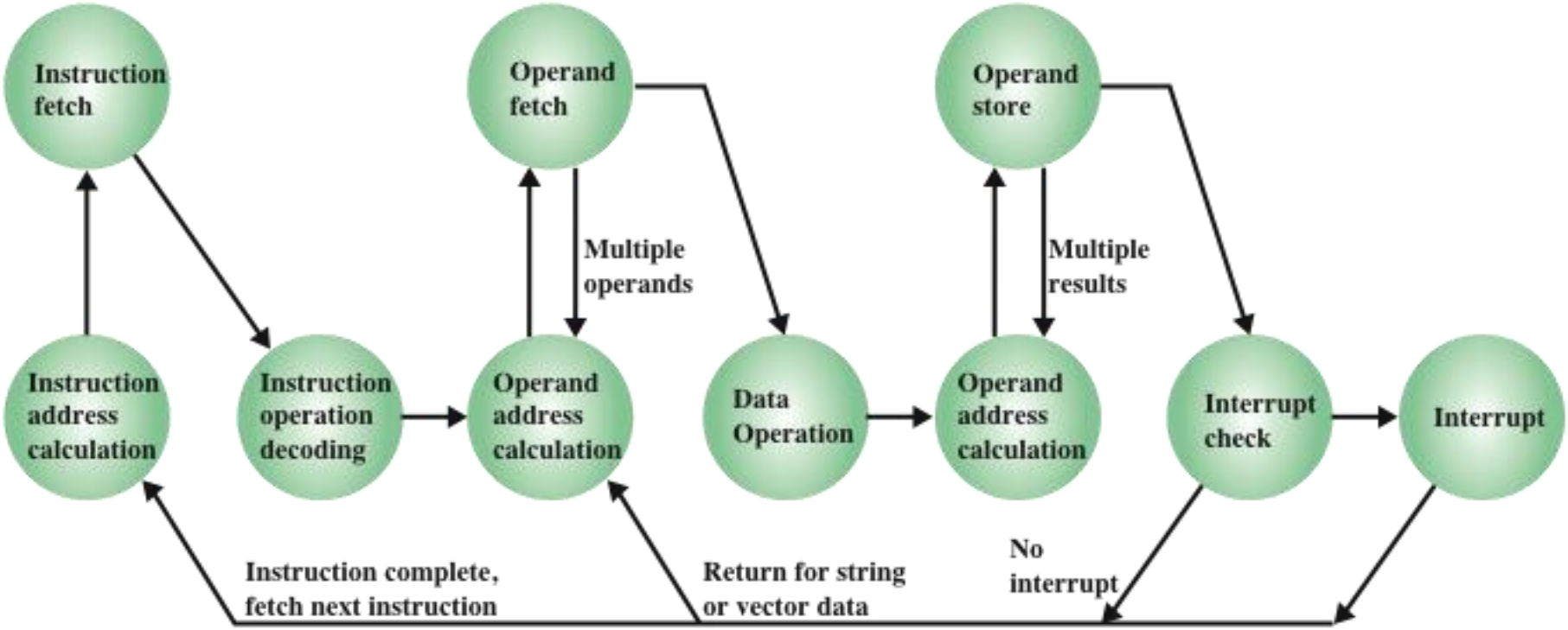


Figure 3.12 Instruction Cycle State Diagram, With Interrupts

Transfer of Control via Interrupts

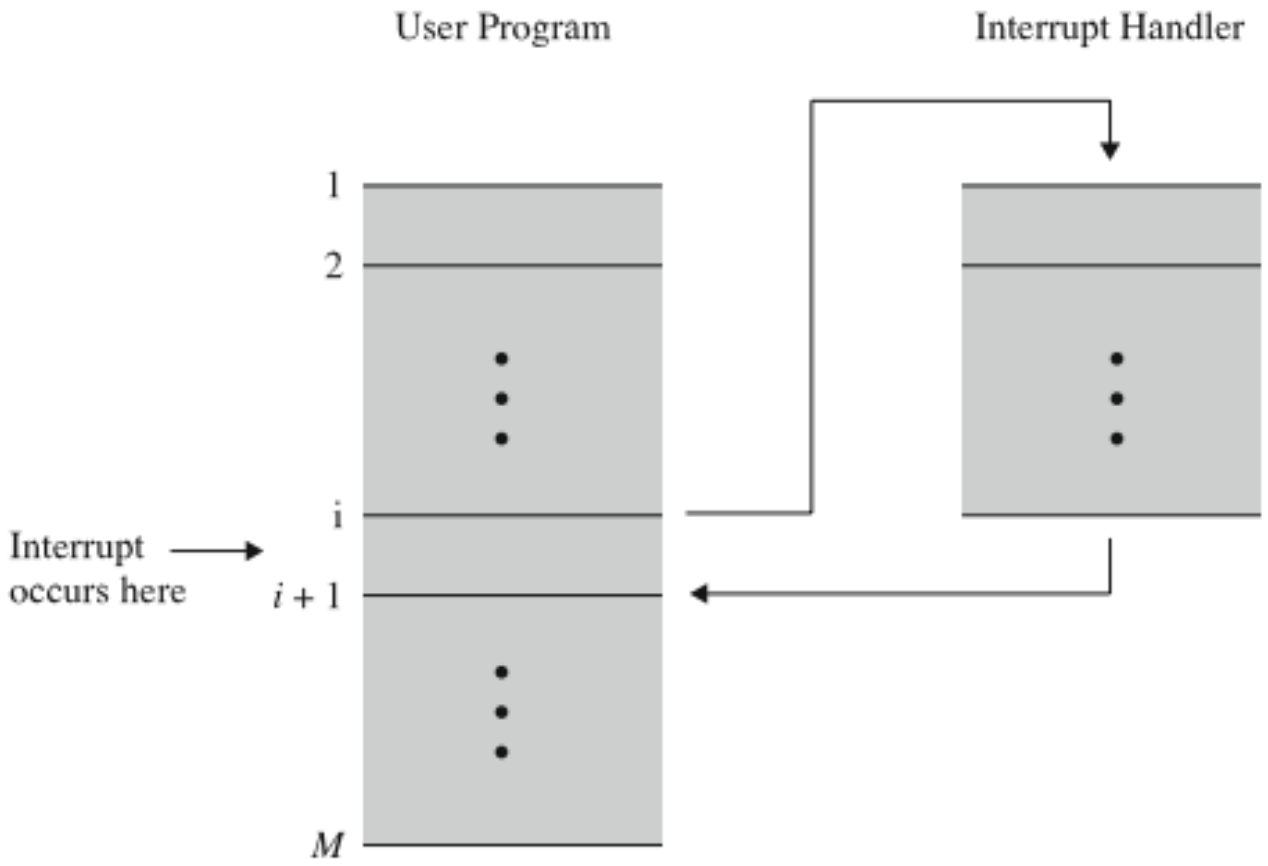
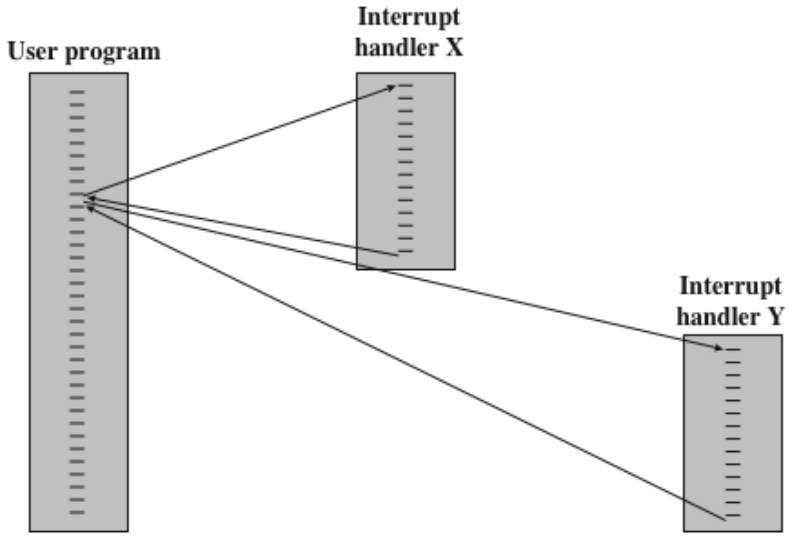


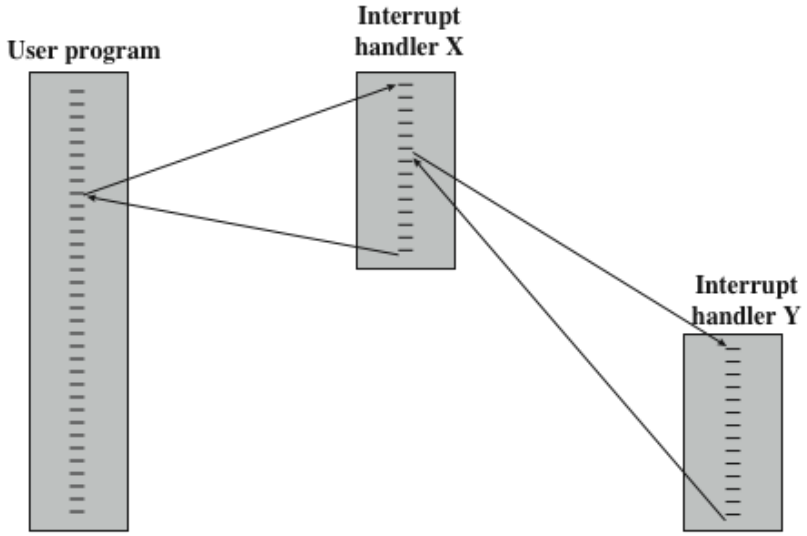
Figure 3.8 Transfer of Control via Interrupts

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

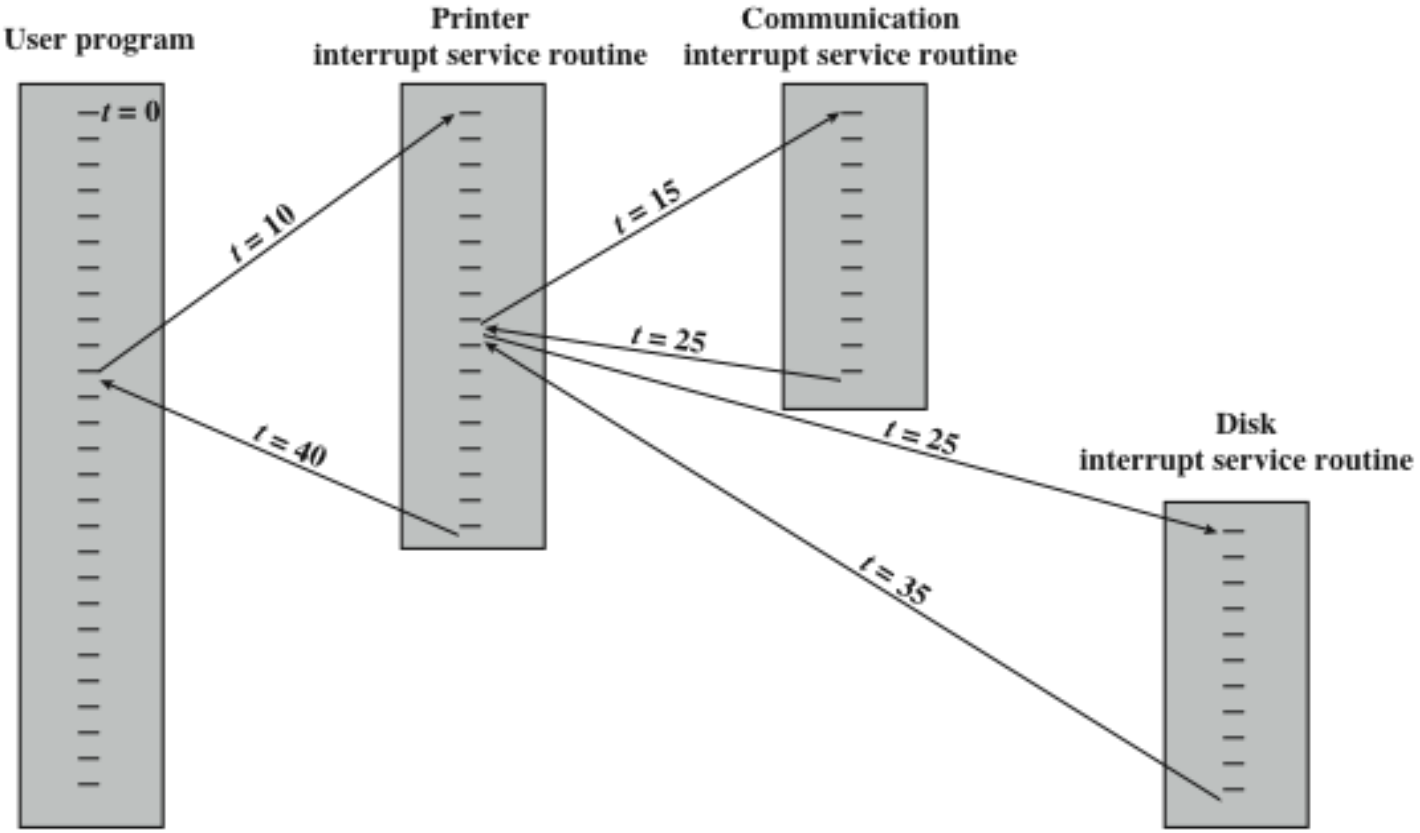


Figure 3.14 Example Time Sequence of Multiple Interrupts

Example

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

The collection of paths connecting the various modules is called the **interconnection structure.**

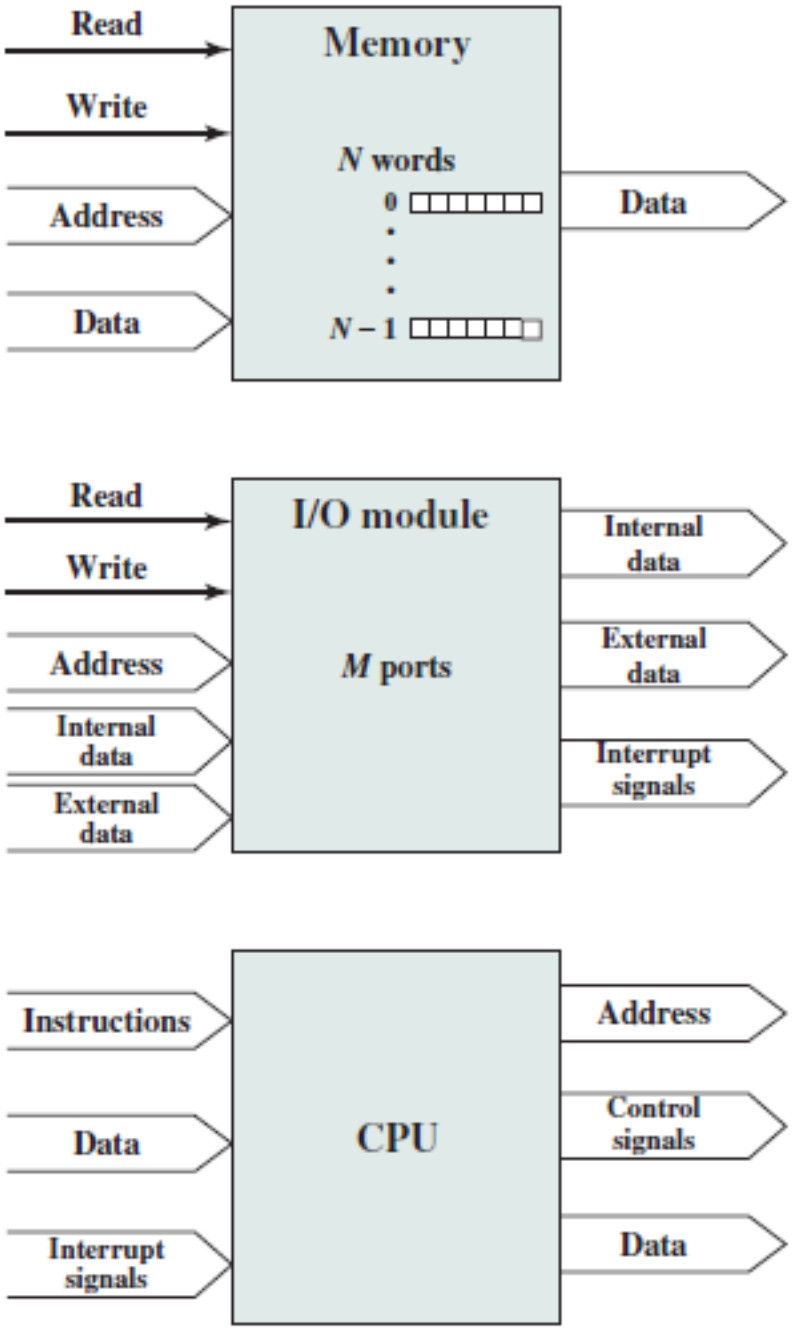


Figure 3.15 Computer Modules

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

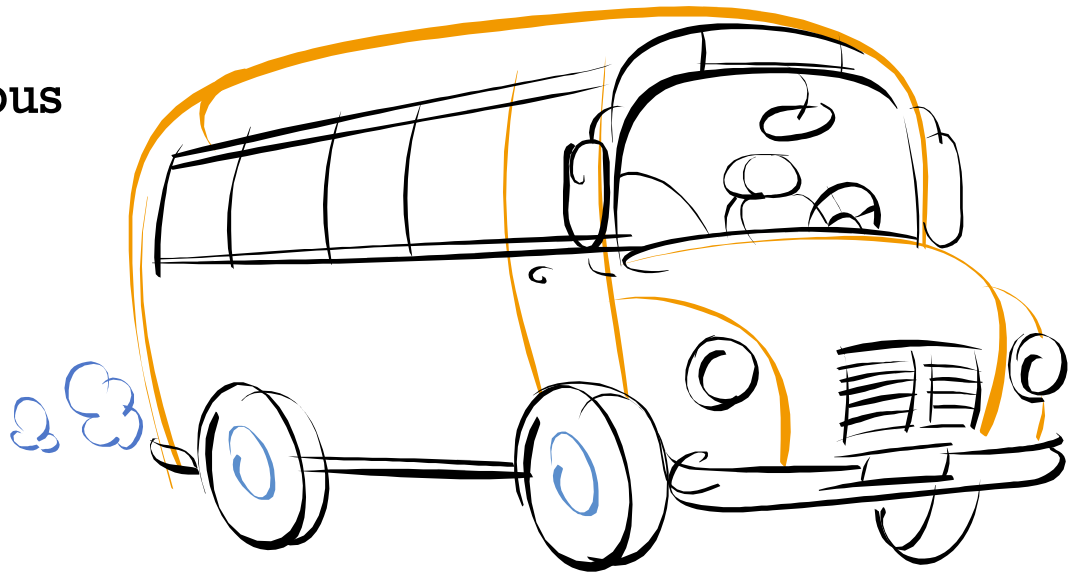
System bus

- 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

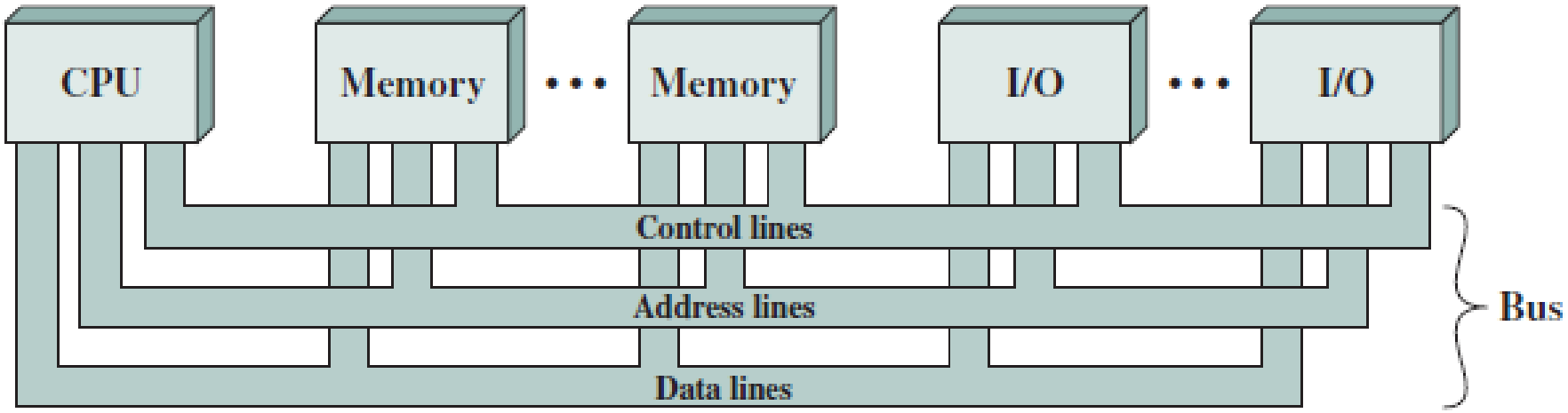
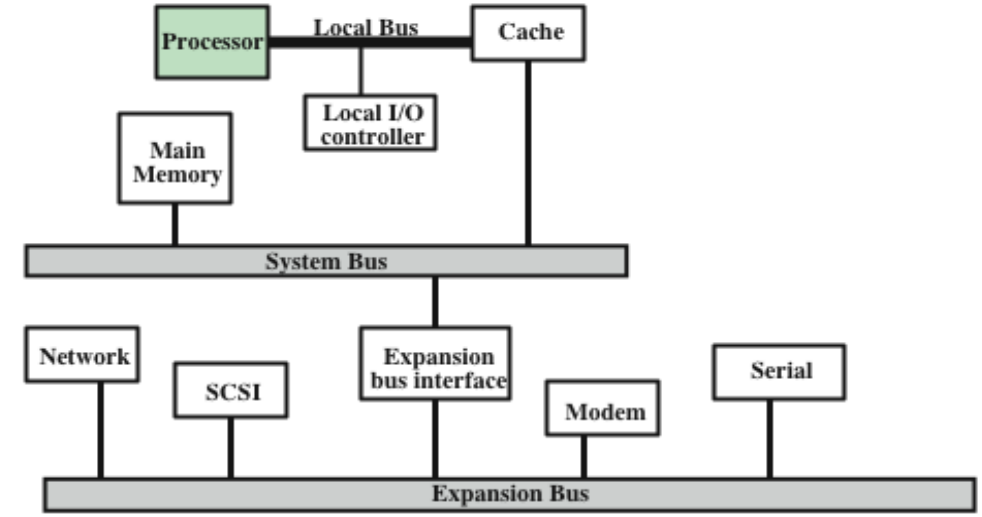
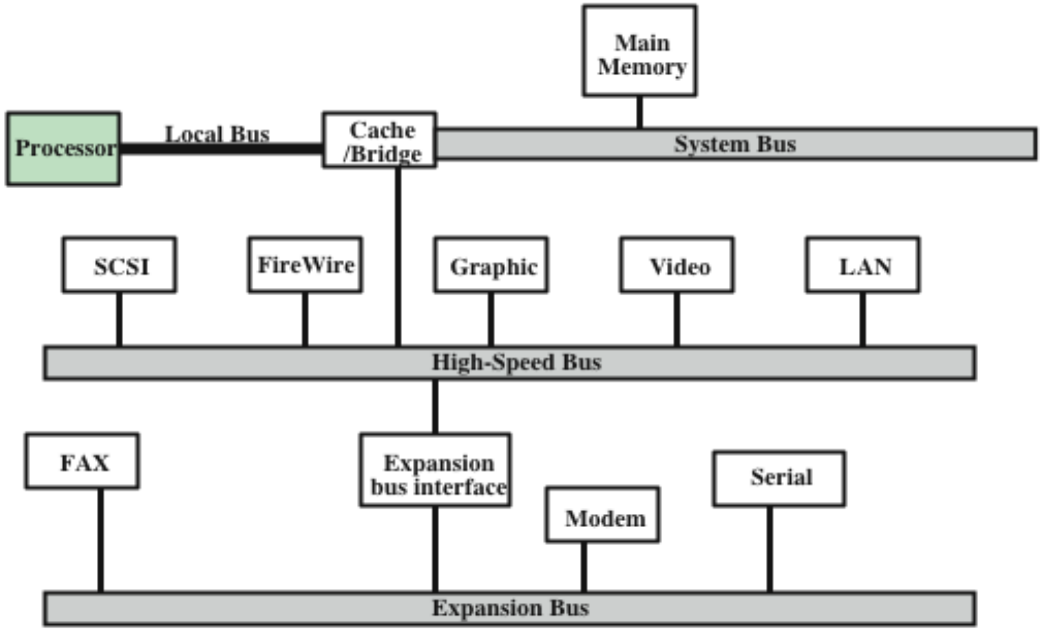


Figure 3.16 Bus Interconnection Scheme



(a) Traditional Bus Architecture



(b) High-Performance Architecture

Figure 3.17 Example Bus Configurations

Elements of Bus Design

Type	**Bus Width**
Dedicated	Address
Multiplexed	Data
Method of Arbitration	**Data Transfer Type**
Centralized	Read
Distributed	Write
Timing	Read-modify-write
Synchronous	Read-after-write
Asynchronous	Block

Timing of Synchronous Bus Operations

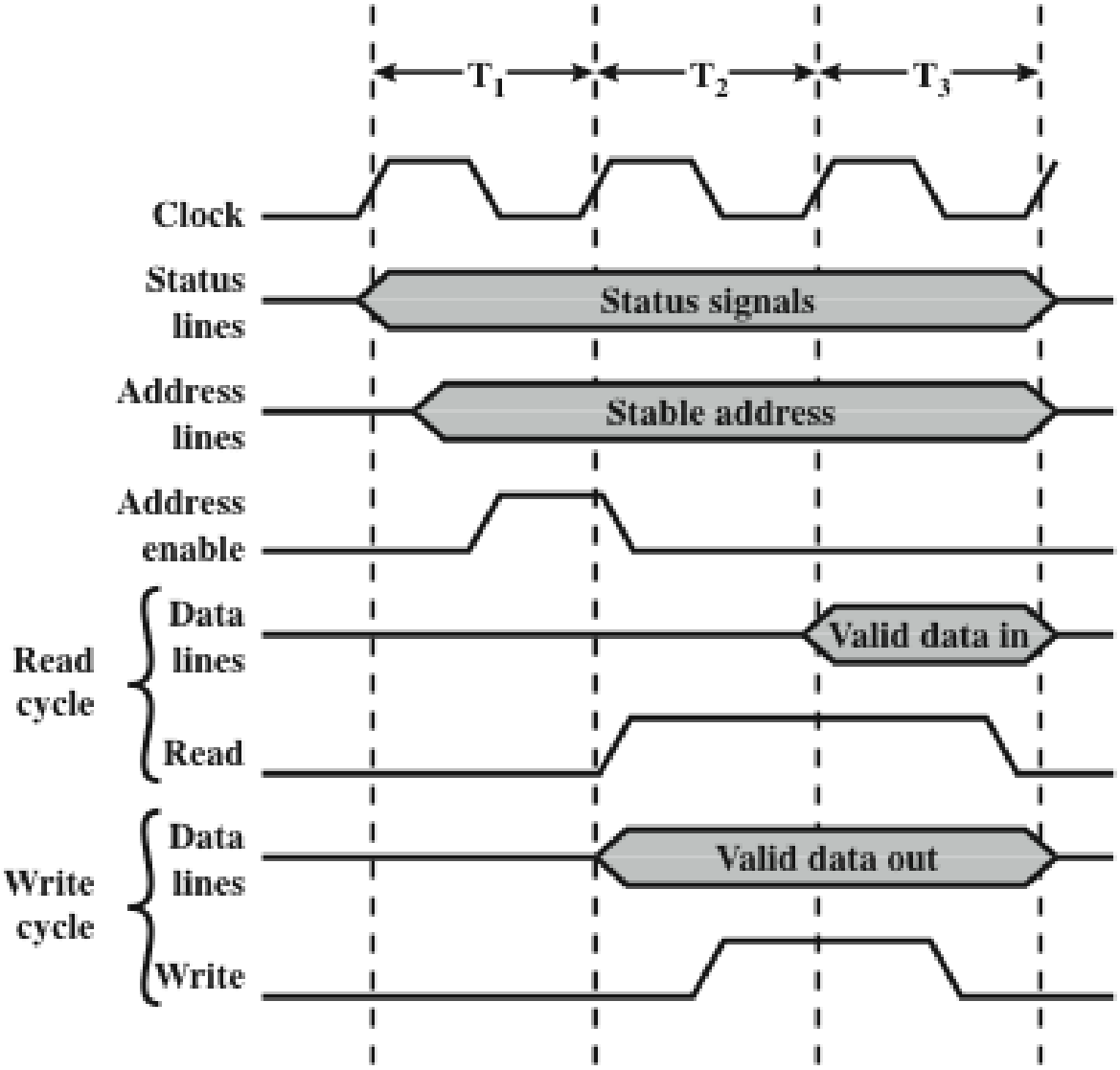
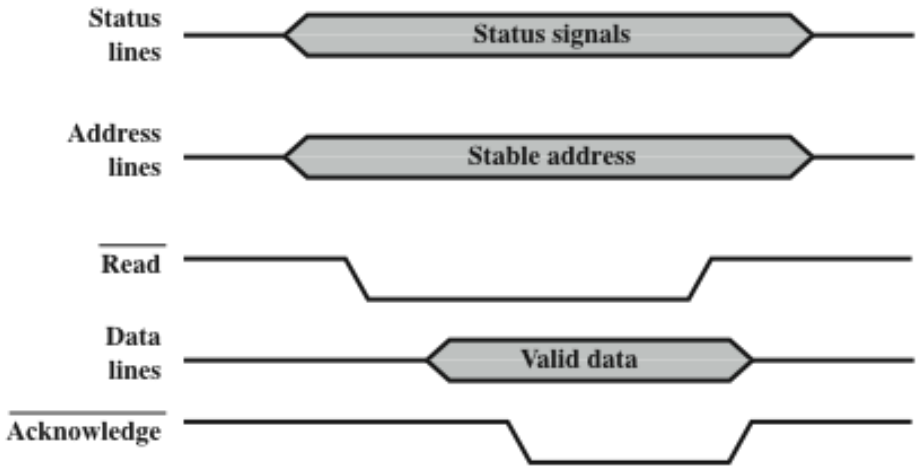
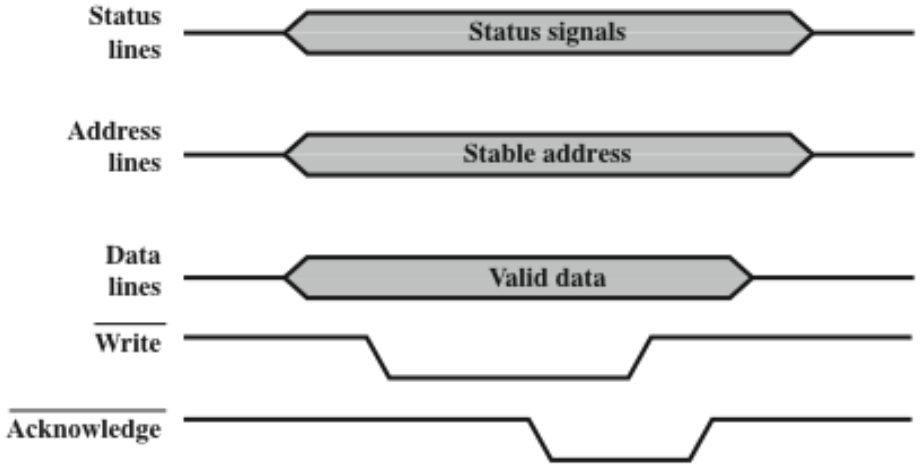


Figure 3.18 Timing of Synchronous Bus Operations

Timing of Asynchronous Bus Operations



(a) System bus read cycle



(b) System bus write cycle

Figure 3.19 Timing of Asynchronous Bus Operations

Point-to-Point Interconnect

Principal reason for driving the change from bus to point-to-point interconnect change was the electrical constraints encountered with increasing the frequency of wide synchronous buses

At higher and higher data rates it becomes increasingly difficult to perform the synchronization and arbitration functions in a timely fashion

A conventional shared bus on the same chip magnified the difficulties of increasing bus data rate and reducing bus latency to keep up with the processors

Has lower latency, higher data rate, and better scalability

Quick Path Interconnect

QPI

- Introduced in 2008

Significant characteristics of QPI:

- Multiple direct connections
  - Direct pairwise connections to other components eliminating the need for arbitration found in shared transmission systems
- Layered protocol architecture
  - These processor level interconnects use a layered protocol architecture rather than the simple use of control signals found in shared bus arrangements
- Packetized data transfer
  - Data are sent as a sequence of packets each of which includes control headers and error control codes

Multicore Configuration Using QPI

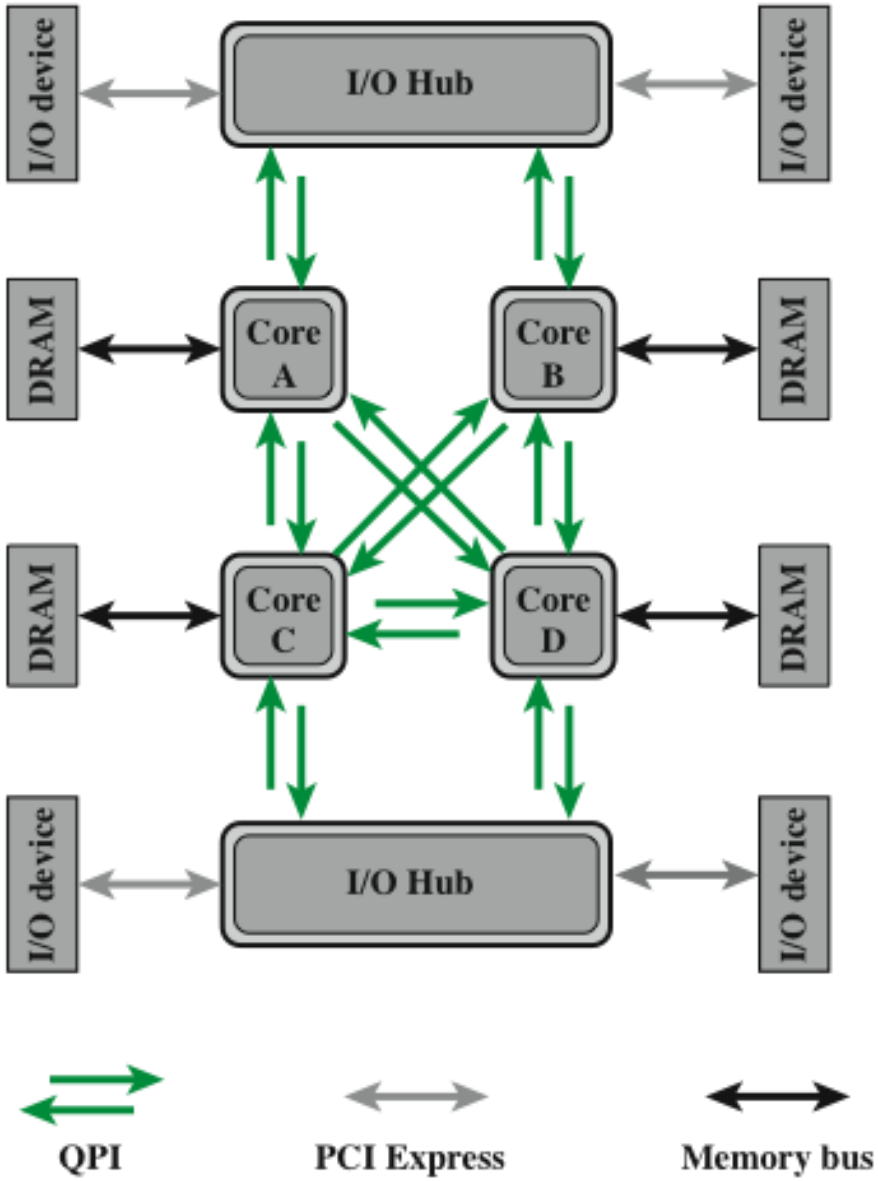


Figure 3.20 Multicore Configuration Using QPI

QPI Layers

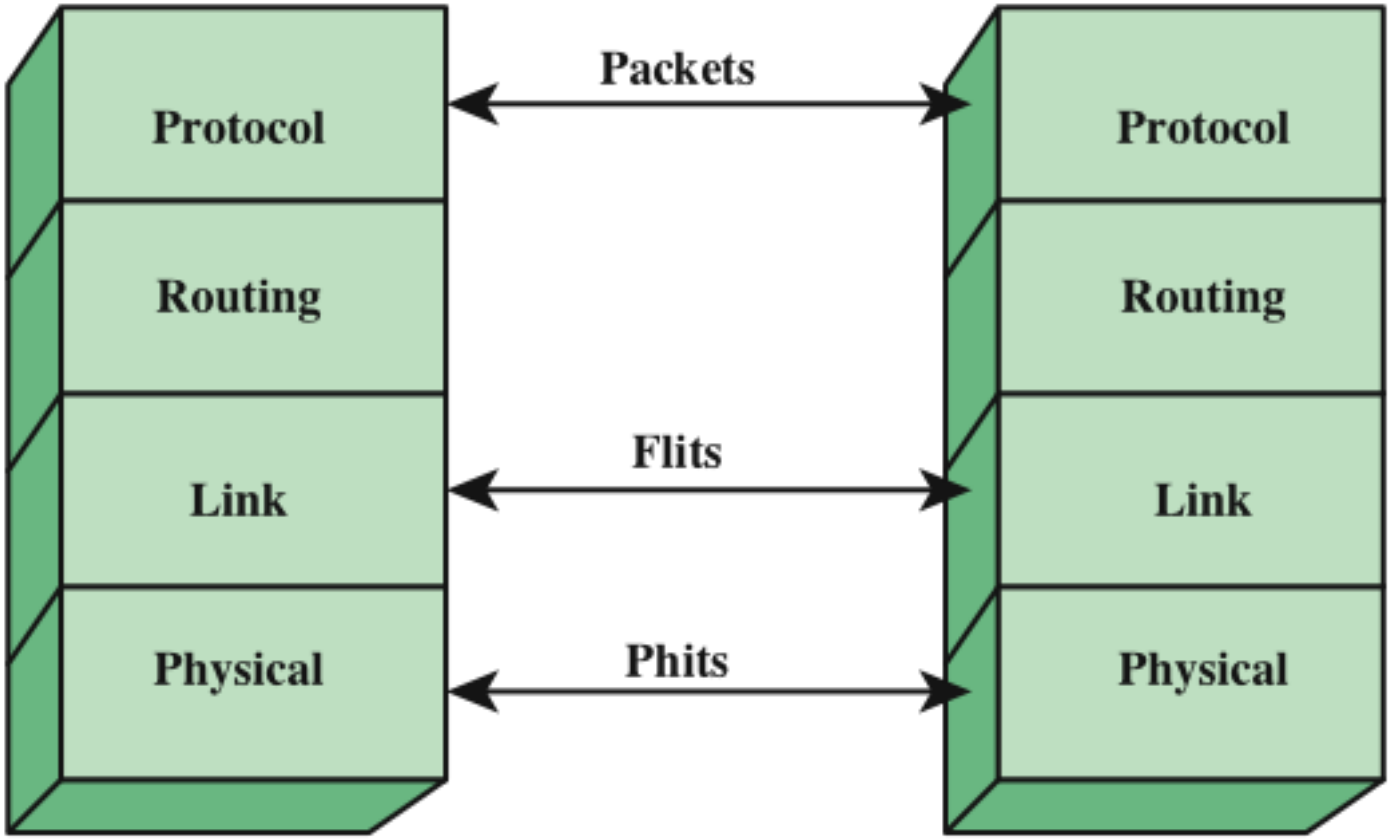


Figure 3.21 QPI Layers

Physical Interface of the Intel QPI Interconnect

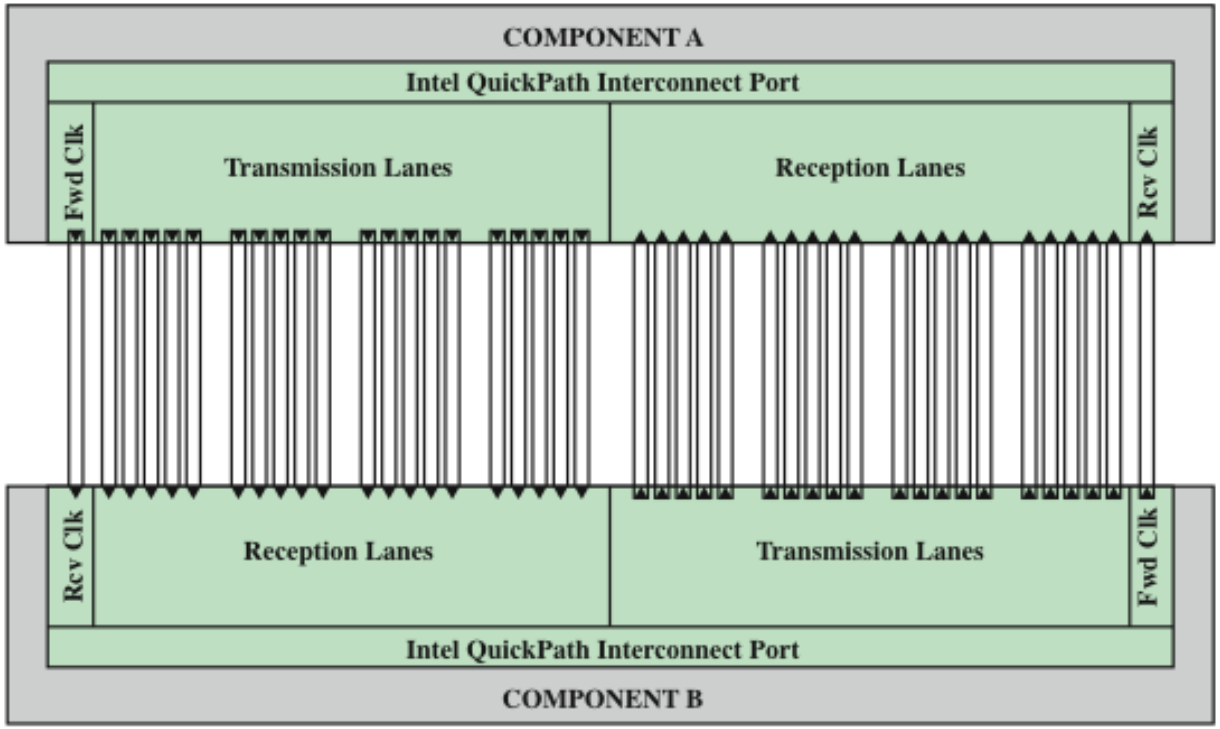


Figure 3.22 Physical Interface of the Intel QPI Interconnect

QPI Multilane Distribution

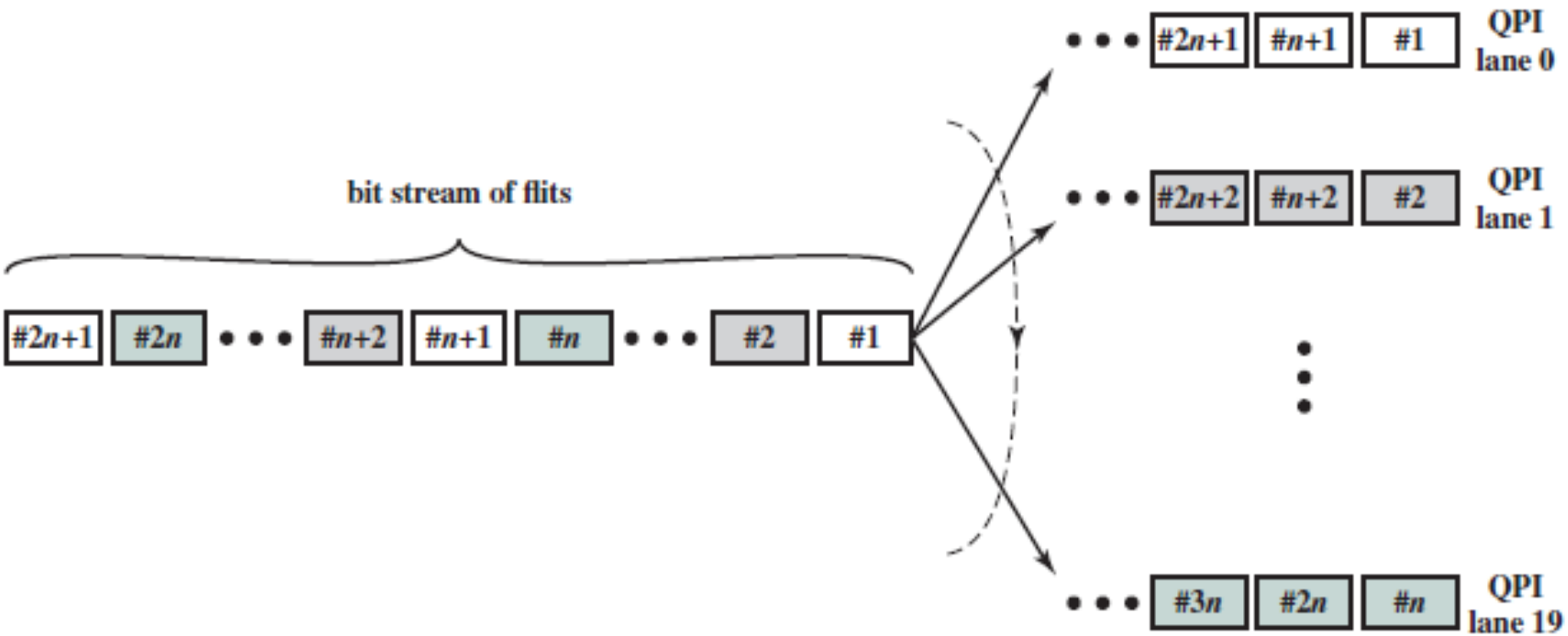


Figure 3.20 QPI Multilane Distribution

QPI Link Layer

- Performs two key functions: ***flow control*** and ***error control***
  - Operate on the level of the flit (flow control unit)
  - Each flit consists of a 72-bit message payload and an 8-bit error control code called a *cyclic redundancy check* (CRC)

- Flow control function
  - Needed to ensure that a sending QPI entity does not overwhelm a receiving QPI entity by sending data faster than the receiver can process the data and clear buffers for more incoming data

- Error control function
  - Detects and recovers from bit errors, and so isolates higher layers from experiencing bit errors

QPI Routing and Protocol Layers

Routing Layer

- Used to determine the course that a packet will traverse across the available system interconnects
- Defined by firmware and describe the possible paths that a packet can follow

Protocol Layer

- Packet is defined as the unit of transfer
- One key function performed at this level is a cache coherency protocol which deals with making sure that main memory values held in multiple caches are consistent
- A typical data packet payload is a block of data being sent to or from a cache

Peripheral Component Interconnect (PCI)

- A popular high bandwidth, processor independent bus that can function as a mezzanine or peripheral bus
- Delivers better system performance for high speed I/O subsystems
- PCI Special Interest Group (SIG)
  - Created to develop further and maintain the compatibility of the PCI specifications
- PCI Express (PCIe)
  - Point-to-point interconnect scheme intended to replace bus-based schemes such as PCI
  - Key requirement is high capacity to support the needs of higher data rate I/O devices, such as Gigabit Ethernet
  - Another requirement deals with the need to support time dependent data streams

PCIe Configuration

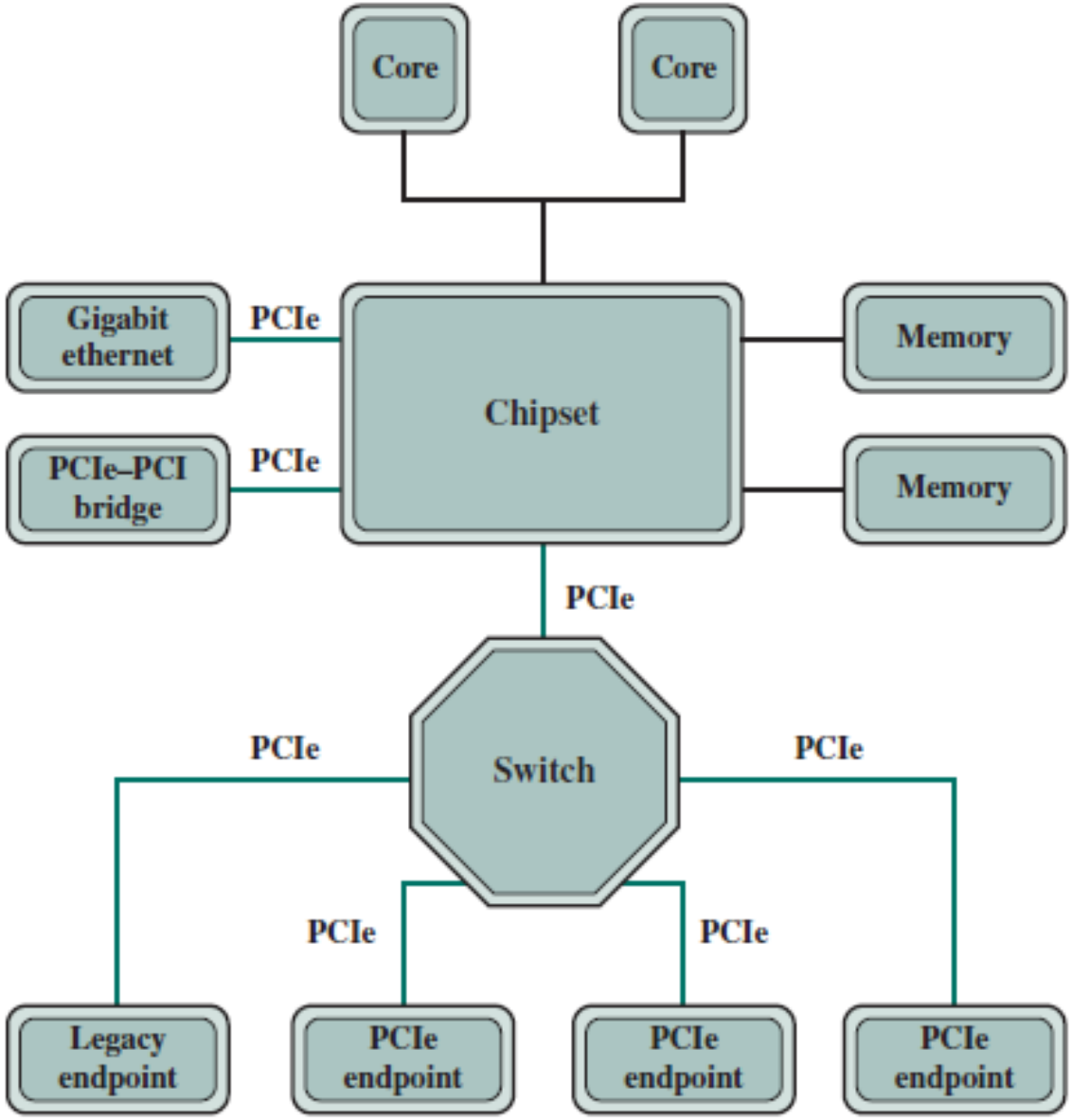


Figure 3.21 Typical Configuration Using PCIe

PCIe Protocol Layers

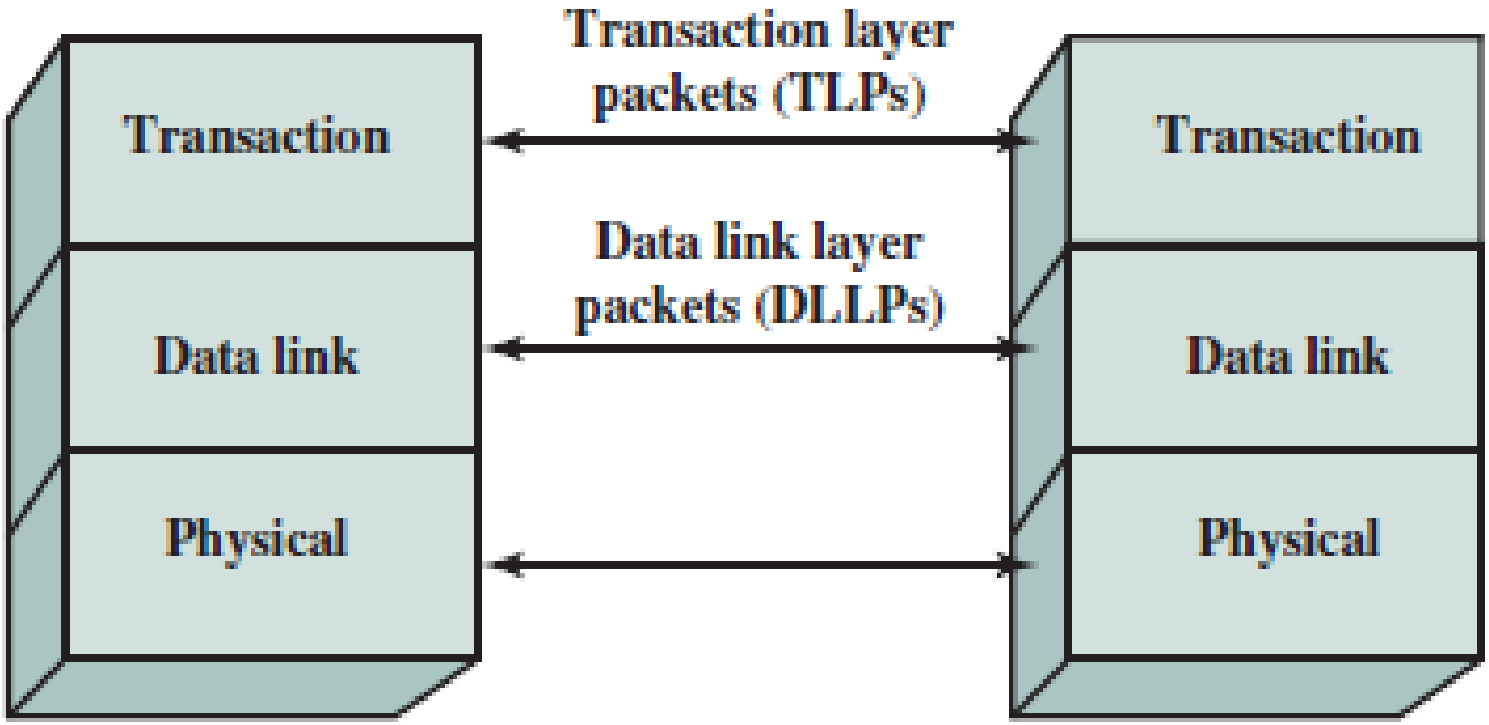


Figure 3.22 PCIe Protocol Layers

PCIe Protocol Layers

Transaction Layer : Responsible for converting requests or completion data from device core to a valid PCIe transaction
Data Link Layer : Integrity of transactions across the link.
Physical Layer : Actually transmit and receive transactions across a PCIe link. On power-on the physical layer initializes the number of layers to be used , link speed etc. The TL and DLL are oblivious to how the data is transmitted, it is taken care by the PL.

PCIe

Transaction Layer (TL)

- Receives read and write requests from the software above the TL and creates request packets for transmission to a destination via the link layer
- Most transactions use a *split transaction* technique
  - A request packet is sent out by a source PCIe device which then waits for a response called a *completion* packet
  - TL messages and some write transactions are posted transactions (meaning that no response is expected)
  - TL packet format supports 32-bit memory addressing and extended 64-bit memory addressing

Transaction Layer –Overview

There are major types of transaction layer packets :

1. **Memory -** Transaction to and from a memory mapped location. They can use 32 bit or 64 bit addressing.
2. **I/O -** Transactions targeting to and from a I/O location. PCIe mainly supports this for backward compatibility of address space. Uses only 32 bit addressing.
3. **Configuration -** Transactions targeting the configuration space targeted for device config and setup during enumeration
4. **Message (new type specific to PCIe) -** This is a new class of transaction in PCIe. Since there are no side band signals as in PCI. The interrupts, error and power management signals are mapped and transmitted as message transactions.

PCIe TLP Transaction Types

Address Space	TLP Type	Purpose
Memory	Memory Read Request	Transfer data to or from a location in the system memory map.
	Memory Read Lock Request	
	Memory Write Request	
I/O	I/O Read Request	Transfer data to or from a location in the system memory map for legacy devices.
	I/O Write Request	
Configuration	Config Type 0 Read Request	Transfer data to or from a location in the configuration space of a PCIe device.
	Config Type 0 Write Request	
	Config Type 1 Read Request	
	Config Type 1 Write Request	
Message	Message Request	Provides in-band messaging and event reporting.
	Message Request with Data	
Memory, I/O, Configuration	Completion	Returned for certain requests.
	Completion with Data	
	Completion Locked	
	Completion Locked with Data	

Data Link Layer - overview

The data link layer assigns a 12 bit sequence number to each TLP as it is passed from the transmit to receive side. Sequence number checks is on a per link basis i.e., Sequence number can be different across links

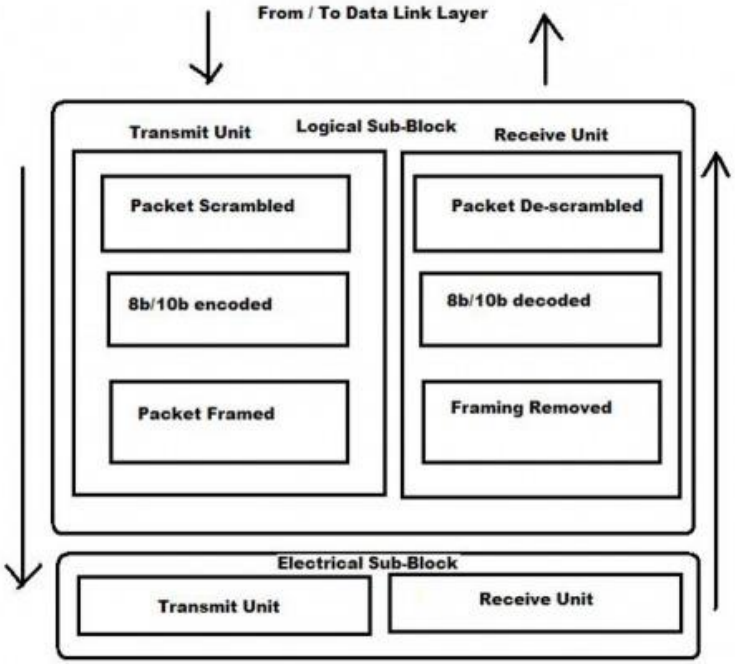
1. Data Link Layer is the gatekeeper for each individual link in the PCI Express system
2. Ensures data sent across the link is correct and received in the same order it was sent
3. Link Management functions are accomplished with the DLLPs (Data Link Layer Packets) which are used by the DLL for error notification, power management, flow control etc.

Data Link Layer - overview

Services of a DLL can be broadly classified into :

- ✓ Data Exchange
- ✓ Error Detection and Retry
- ✓ TLP Sequence number and LCRC generation
- ✓ Initialization and Power Management

Physical Layer- overview



The two key sub-blocks of the physical layer are : Logical sub-block and electrical sub-block.

Logical Sub-block – Decision maker for the physical layer.

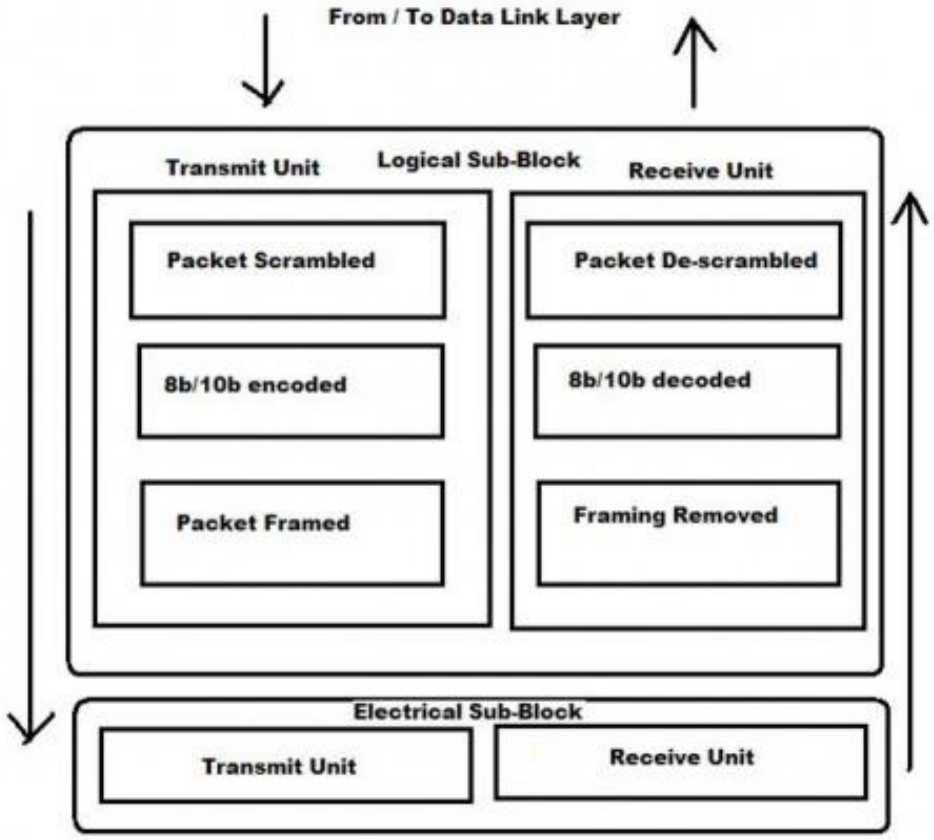
The logical sub-block has a transmit and receive unit.

Transmit Unit has three primary stages of operation :

Data scrambling

128b/130b encoding and packet framing

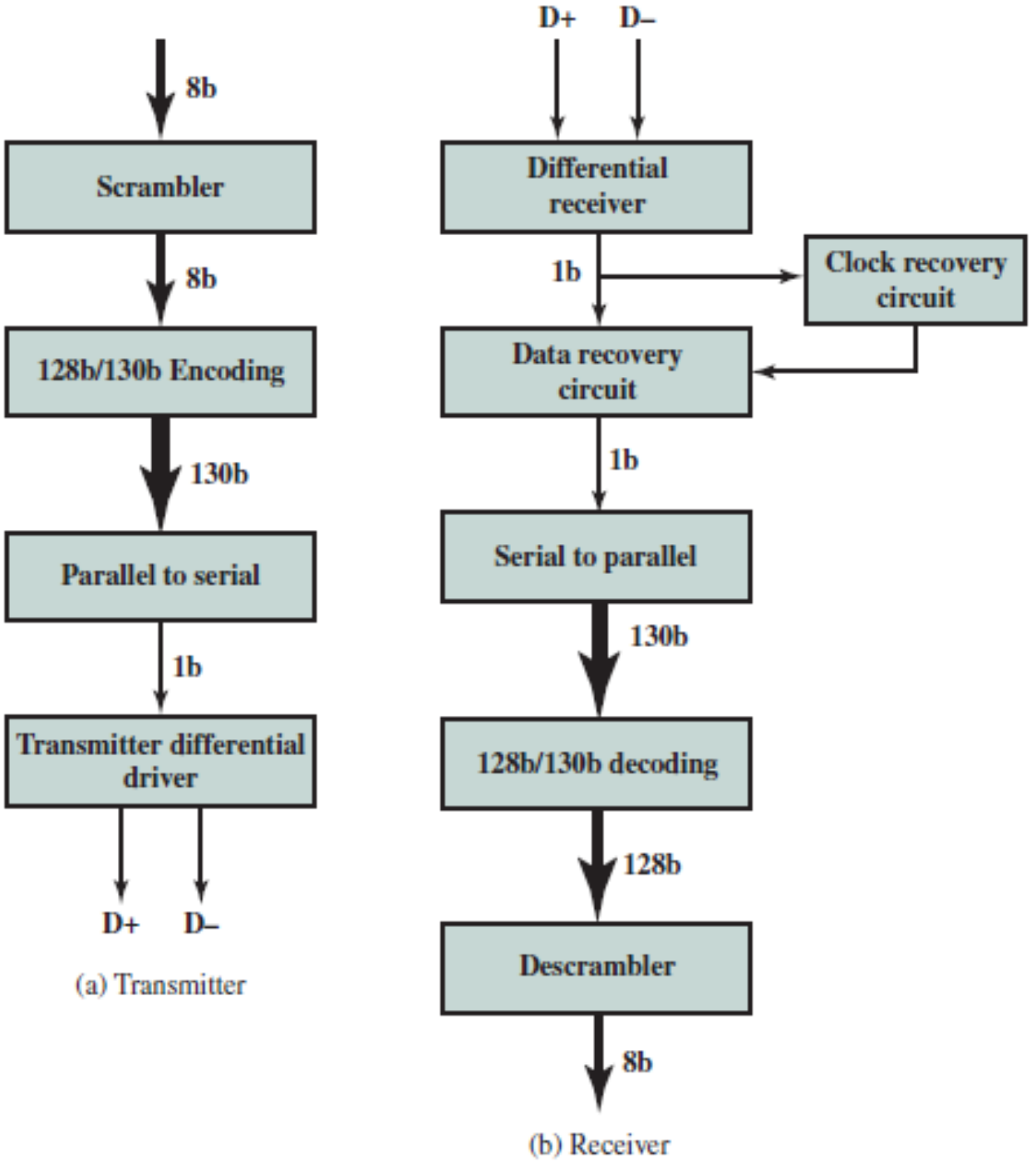
Physical Layer- overview



Receive unit takes the de-serialized physical packet from link, removes framing, decodes and descrambles the packet

The electrical sub-block functions as the delivery mechanism for the physical link. Its primary functions include: Serial- parallel conversion, Clock extraction and Lane-to-Lane de-skew.

PCIe Transmit and Receive Block Diagrams



PCIe Multilane Distribution

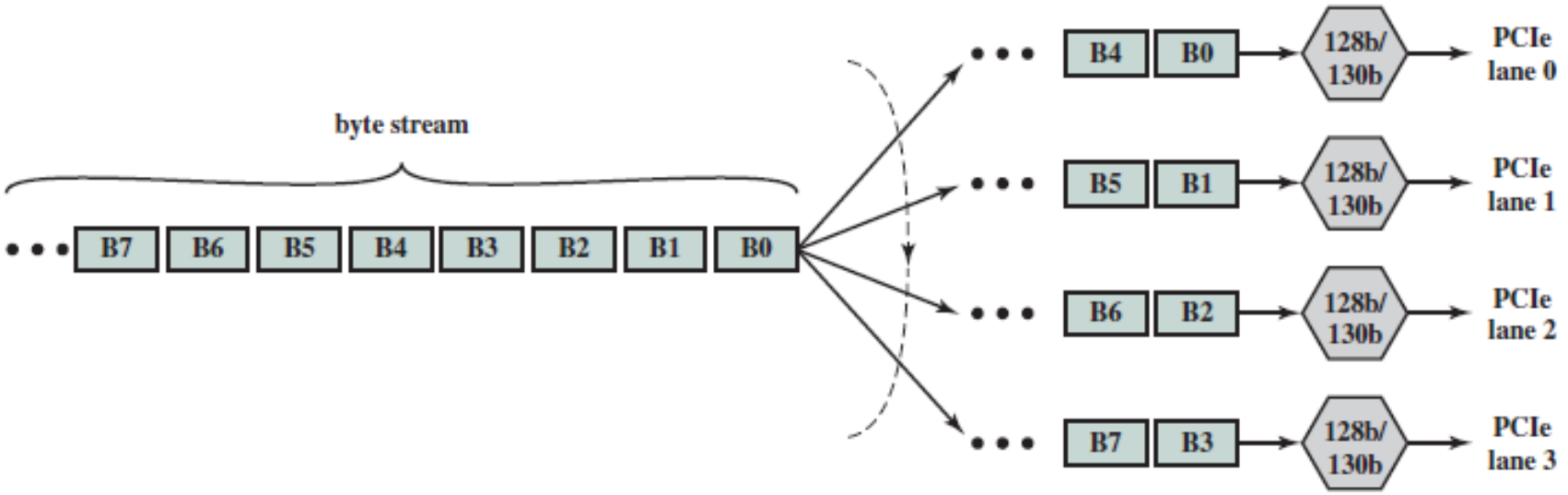
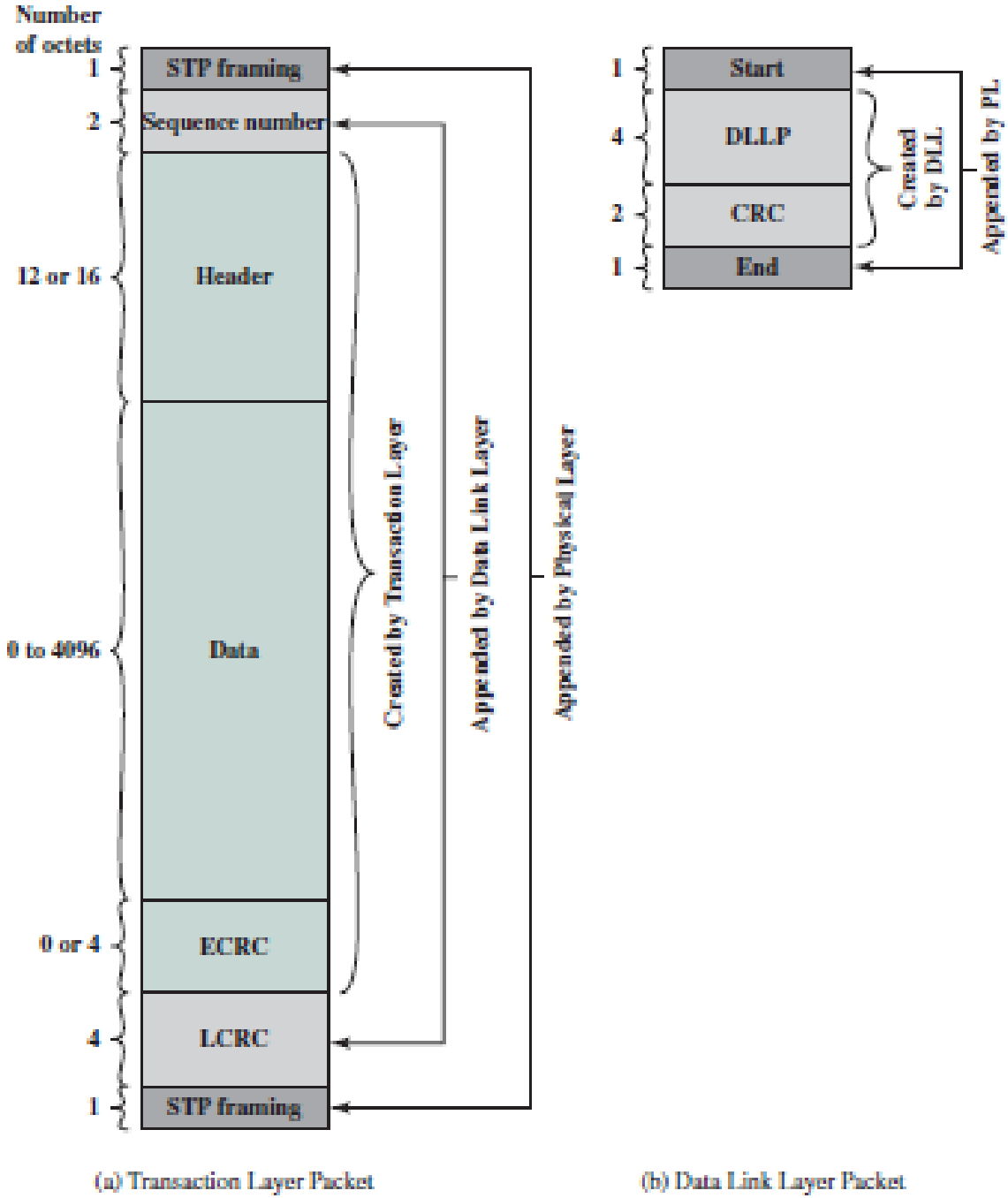


Figure 3.23 PCIe Multilane Distribution

The TL supports four address spaces:

- Memory
  - The memory space includes system main memory and PCIe I/O devices
  - Certain ranges of memory addresses map into I/O devices
- Configuration
  - This address space enables the TL to read/write configuration registers associated with I/O devices
- I/O
  - This address space is used for legacy PCI devices, with reserved address ranges used to address legacy I/O devices
- Message
  - This address space is for control signals related to interrupts, error handling, and power management

PCIe Protocol Data Unit Format



(a) Transaction Layer Packet

(b) Data Link Layer Packet

Thank You :)