Computer Organization and Architecture

Introduction to Computer Organization

- Computer architecture refers to those attributes of a system visible to a programmer or, put another way, those attributes that have a direct impact on the logical execution of a program.

- Computer organization refers to the operational units and their interconnections that realize the architectural specifications.

At each level, the designer is concerned with structure and function:

- Structure: The way in which the components are interrelated
- Function: The operation of each individual component as part of the structure

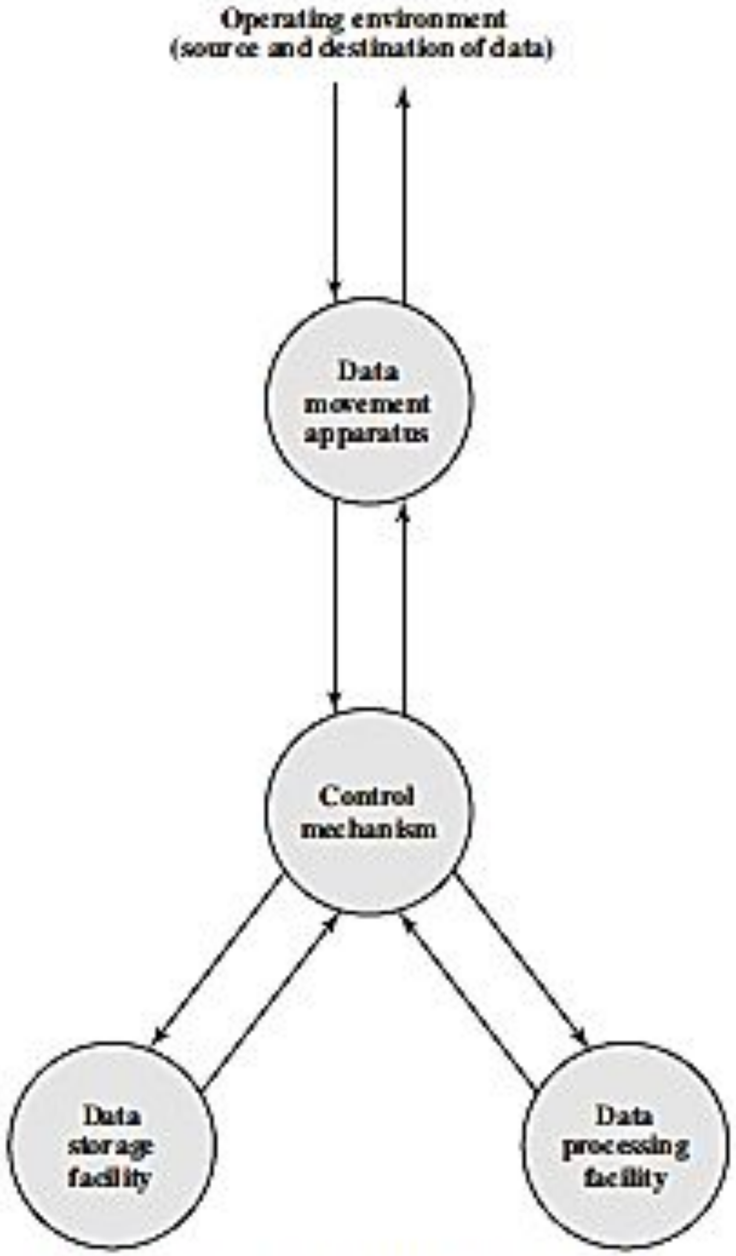
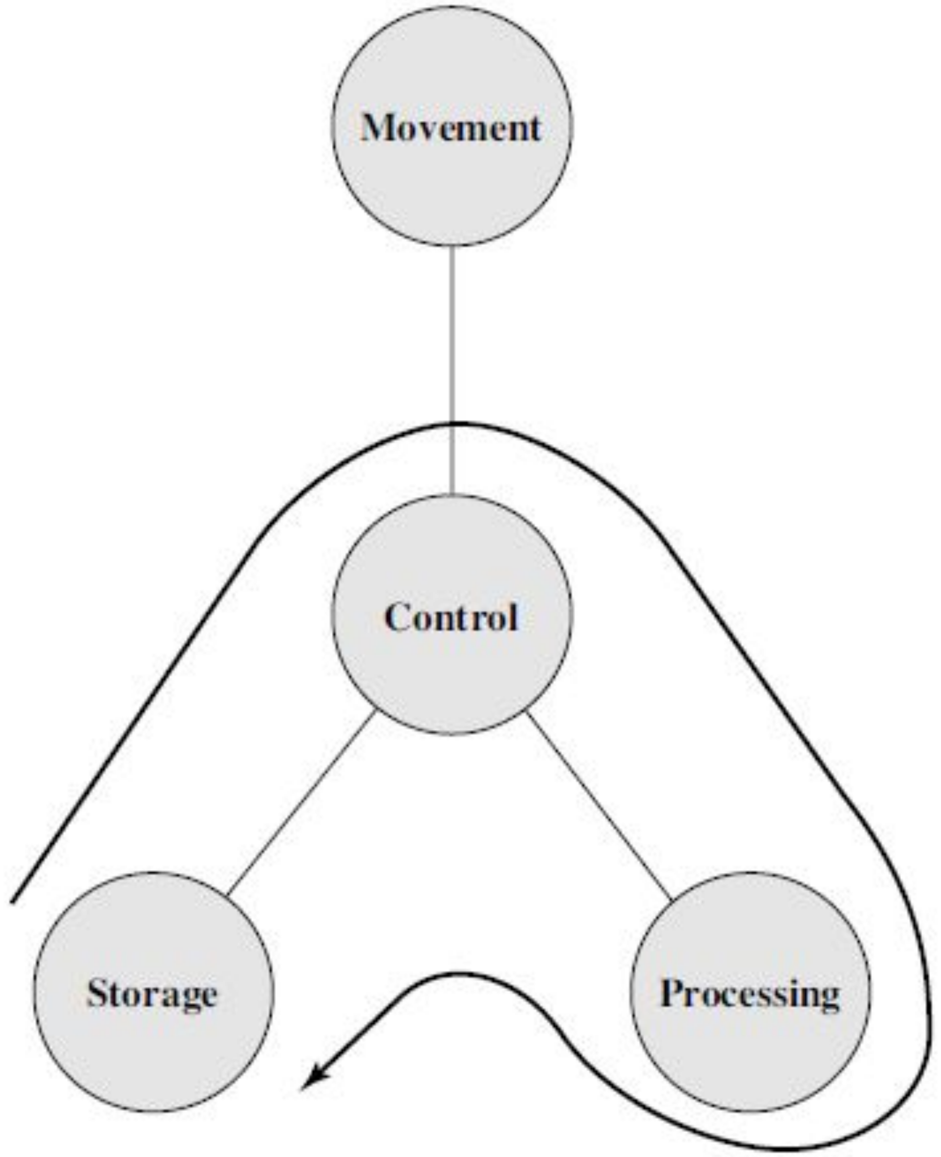


Figure 1.1 A Functional View of the Computer

Function Both the structure and functioning of a computer are, in essence, simple. Figure 1.1 depicts the basic functions that a computer can perform.

In general terms, there are only four:

- Data processing
- Data storage
- Data movement : peripheral devices, data communication, I/O
- Control: computer’s resources and orchestrates the performance of its functional parts



Movement
Control
Storage
Processing

structure

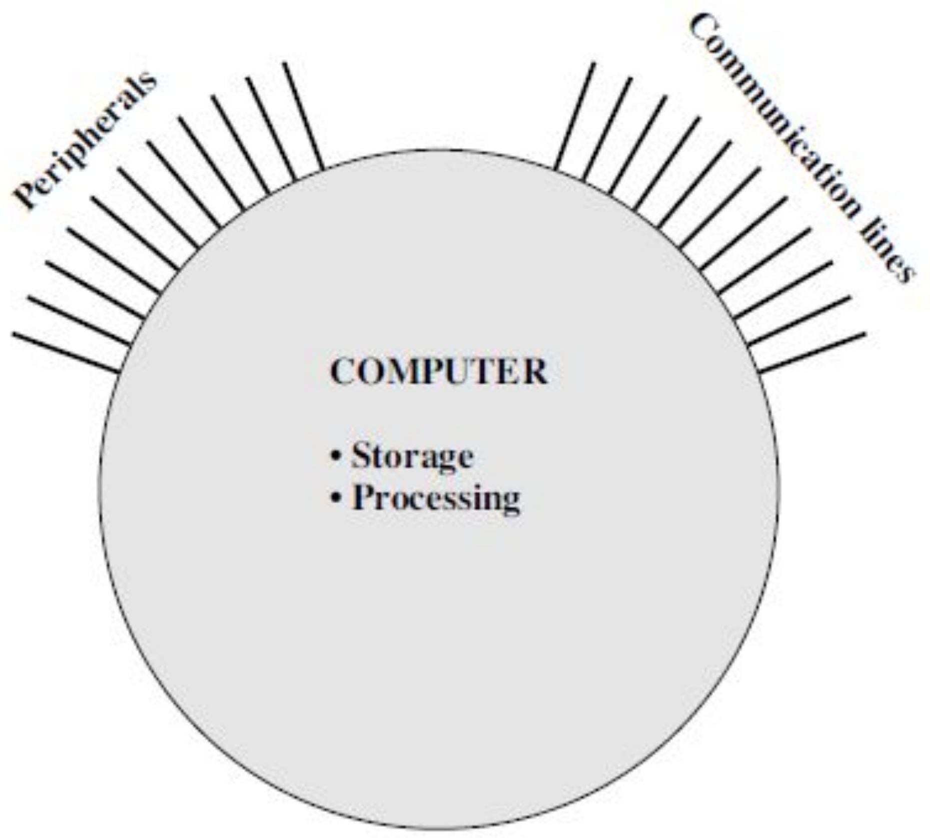


Figure 1.3 The Computer

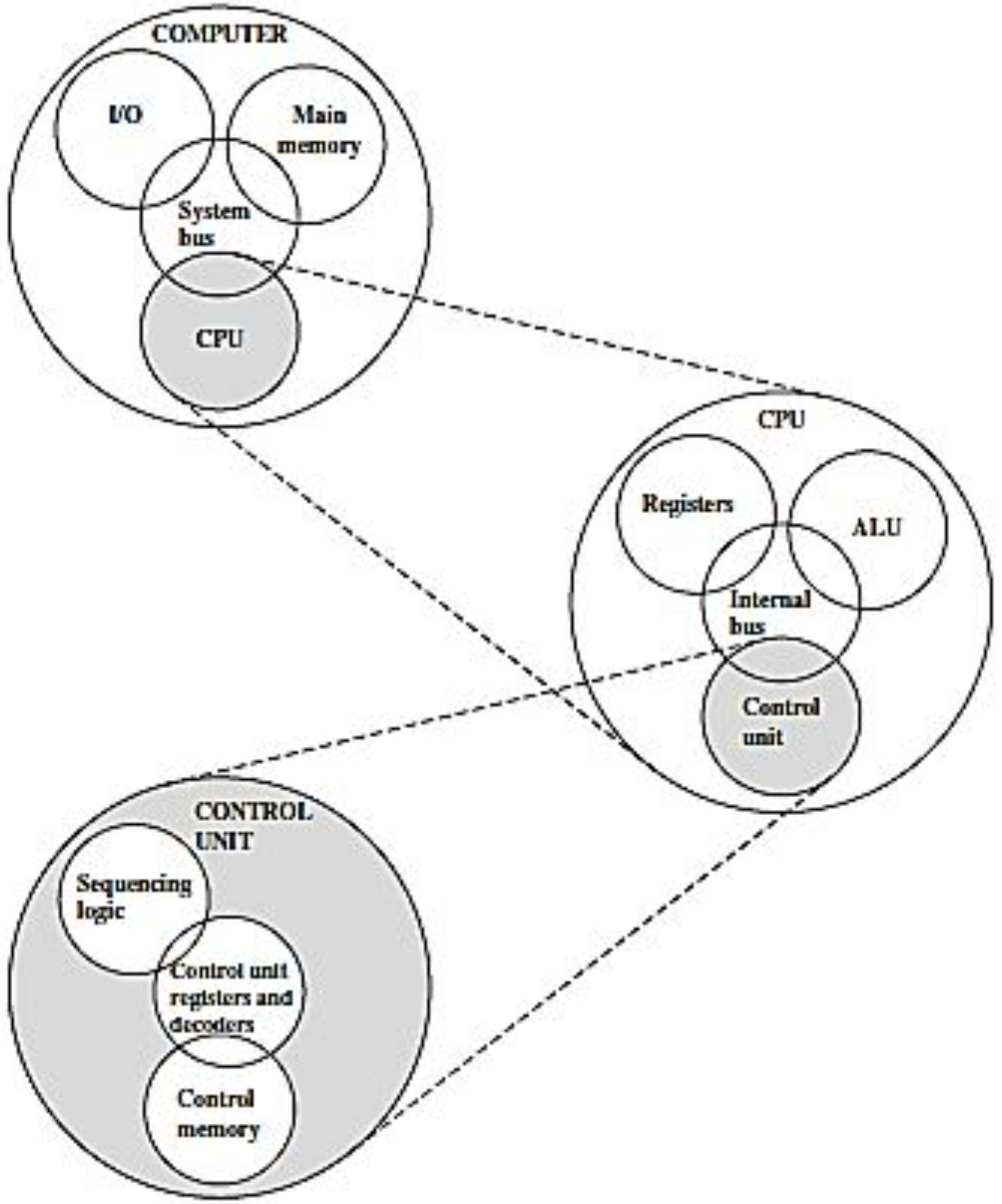


Figure 1.4 The Computer: Top-Level Structure

- There are four main structural components:

- **Central processing unit (CPU):** Controls the operation of the computer and

performs its data processing functions; often simply referred to as **processor**.

- **Main memory:** Stores data.
- **I/O:** Moves data between the computer and its external environment.
- **System interconnection:** Some mechanism that provides for communication among CPU, main memory, and I/O.

- **system bus**, consisting of a number of connecting wires to which all the other components attach.

the most complex component is the CPU. Its major structural components are as follows:

- **Control unit:** Controls the operation of the CPU and hence the computer
- **Arithmetic and logic unit (ALU):** Performs the computer’s data processing

functions

- **Registers:** Provides storage internal to the CPU
- **CPU interconnection:** Some mechanism that provides for communication
- mong the control unit, ALU, and registers

A Brief History of Computers

The evolution of computers has been characterized by

- increasing processor speed,
- decreasing component size,
- increasing memory size,
- and increasing I/O capacity and speed.

- One factor responsible for the great increase in processor speed is the shrinking size of microprocessor components; this reduces the distance between components and hence increases speed.

- **A critical issue** in computer system design is balancing the performance of the various elements so that gains in performance in one area are not handicapped by a lag in other areas.

- **The First Generation: Vacuum Tubes**

- ***ENIAC*** The ENIAC (Electronic Numerical Integrator And Computer), designed and constructed at the University of Pennsylvania, was the world’s first general-purpose electronic digital computer.

- The major drawback of the ENIAC was that it had to be programmed manually by setting
- switches and plugging and unplugging cables.

- von Neumann and his colleagues began the design of a new stored program computer, referred to as the IAS computer, at the Princeton Institute for Advanced Studies.

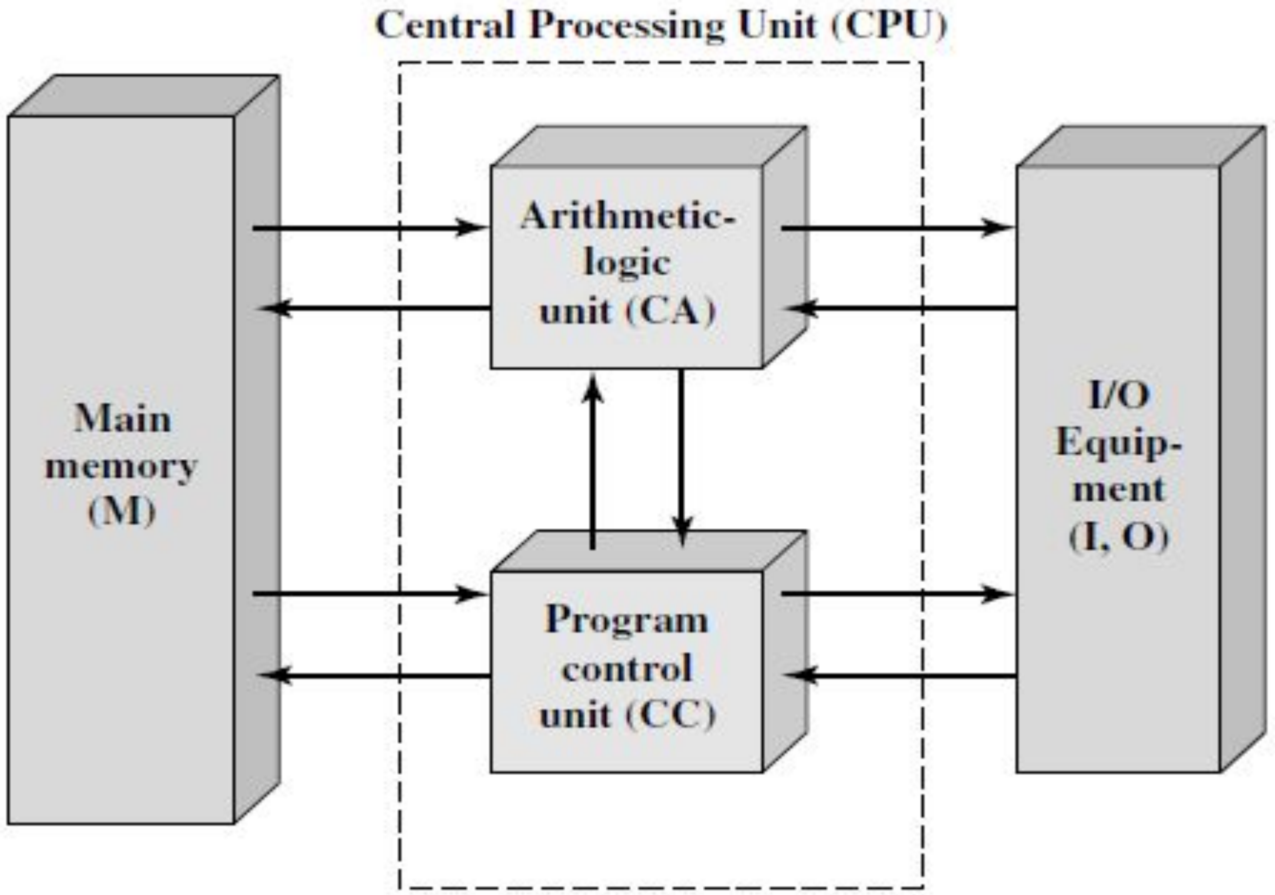

Central Processing Unit (CPU)
Main memory (M)
Arithmetic-logic unit (CA)
Program control unit (CC)
I/O Equipment (I, O)

Figure 2.1 Structure of the IAS Computer

The Second Generation: Transistors

- *solid-state device,* made from silicon.
- The second generation saw the introduction of **more complex arithmetic and logic units and control units, the use of high level**

programming languages, and the provision of *system software* with the computer.

The Third Generation: Integrated Circuits

- A single, self-contained transistor is called a *discrete component.*

- Microprocessor: all the components of the CPU on the single chip

- Fourth and fifth generation

PERFORMANCE ASSESSMENT

- **Clock Speed and Instructions per Second:**

clock signals are generated by a quartz crystal, which generates a constant signal wave while power is applied. This wave is converted into a digital voltage pulse stream that is provided in a constant flow to the processor. For example, a 1-GHz processor receives 1 billion pulses per second. The rate of pulses is known as the **clock rate**, or **clock speed.**

- ***INSTRUCTION EXECUTION RATE***
- ***MIPS and MFLOPS***

DESIGNING FOR PERFORMANCE

- **Microprocessor Speed:**
  - **pipelining**
  - **Branch prediction:**
  - **Data flow analysis:**
  - **Speculative execution:**
  - **Performance Balance:**

Improvements in Chip Organization and Architecture

- There are three approaches to achieving increased processor speed:
- Increase the hardware speed of the processor:
- With gates closer together, the propagation time for signals is significantly reduced, enabling a speeding up of the processor.
- An increase in clock rate means that individual operations are executed more rapidly.

Improvements in Chip Organization and Architecture

- Increase the size and speed of caches that are interposed between the processor and main memory:
- In particular, by dedicating a portion of the processor chip itself to the cache, cache access times drop significantly.
- Make changes to the processor organization and architecture that increase the effective speed of instruction execution: Typically, this involves using parallelism in one form or another.

Improvements in Chip Organization and Architecture

- as **clock speed and logic density increase**, a number of **obstacles** become more significant
- **Power:** As the density of logic and the clock speed on a chip increase, so does the power density (Watts/cm2). The difficulty of **dissipating the heat generated on high-density,** high-speed chips is becoming a serious design issue.
- **RC delay:** The **speed at which electrons can flow on a chip between transistors is limited by the resistance and capacitance** of the metal wires connecting them; specifically, delay increases as the RC product increases. As components on the chip decrease in size, the wire interconnects become thinner, increasing resistance. Also, the wires are closer together, increasing capacitance.

Improvements in Chip Organization and Architecture

- as clock speed and logic density increase, a number of obstacles become more significant
- **Memory latency and throughput:** Memory access speed (latency) and transfer speed (throughput) lag processor speeds.

Multicore, Mics, and GPGPUs

- **Multicore:** placing multiple processors on the same chip, with a large shared cache.
- Studies indicate that, within a processor, **the increase in performance is roughly proportional to the square root of the increase in complexity.**
- But if the software can support the effective use of multiple processors, then doubling the number of processors almost doubles performance.
- Thus, the strategy is to **use two simpler processors on the chip rather than one more complex processor**.
- the **power consumption of memory logic on a chip is much less than that of processing logic**.

Multicore, Mics, and GPGPUs

- As the caches became larger, it made performance sense to create two and then three levels of cache on a chip, with initially, **the first-level cache dedicated to an individual processor and levels two and three being shared by all the processors.**
- It is now common for the second-level cache to also be private to each core.
- **many integrated core (MIC)**: more than 50 cores per chip.
- The multicore and MIC strategy involves a homogeneous collection of general purpose processors on a single chip.

Multicore, Mics, and GPGPUs

- GPUs perform parallel operations on multiple sets of data, they are increasingly being used as vector processors for a variety of applications that require repetitive computations.

multicore computer structure

- **Central processing unit (CPU):** That portion of a computer that **fetches and executes instructions.** It consists of an **ALU, a control unit, and registers.** In a system with a single processing unit, it is often simply referred to as a *processor*.
- **Core:** An individual processing unit on a processor chip. A core may be equivalent in functionality to a CPU on a single-CPU system. Other specialized processing units, such as one optimized for vector and matrix operations, are also referred to as cores.
- **Processor: A physical piece of silicon containing one or more cores.** The processor is the computer component that interprets and executes instructions. If a processor contains multiple cores, it is referred to as a **multicore processor**.

multicore computer structure

- A **printed circuit board (PCB)** is a **rigid, flat board that holds and interconnects chips and other electronic components.** The board is made of layers, typically two to ten, that interconnect components via copper pathways that are etched into the board.
- The main printed circuit board in a computer is called a system board or **motherboard**, while smaller ones that plug into the slots in the main board are called expansion boards.
- A **chip** is **a single piece of semiconducting material, typically silicon, upon which electronic circuits and logic gates are fabricated.** The resulting product is referred to as an **integrated circuit**.

multicore computer structure

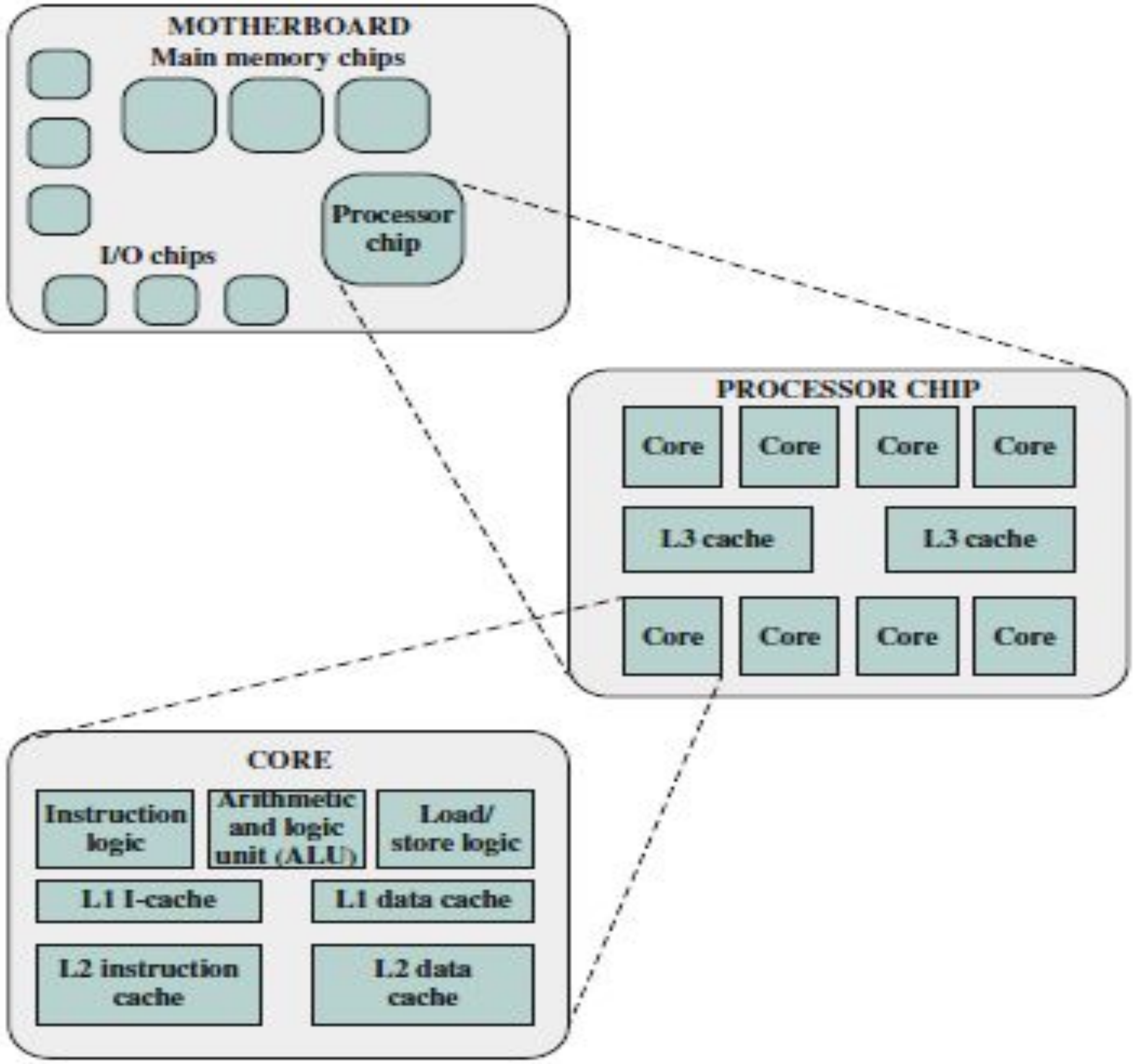


Figure 1.2 Simplified View of Major Elements of a Multicore Computer

Amdahl’s Law

$$\text{Speedup} = \frac{\text{time to execute program on a single processor}}{\text{time to execute program on N parallel processors}} = \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$

- Example:

$$\text{Speedup} = \frac{1}{0.6 + \frac{0.4}{K}}$$

Little’s Law

- we have a steady state system to which **items arrive at an average rate of λ items per unit time. The items stay in the system an average of *W* units of time.** Finally, there is an **average of *L* units** in the system at any one time. Little’s Law relates these three variables as
    - $L = \lambda W$.
- The server in this model can represent anything that performs some function or service for a collection of items.
- Since items arrive at a rate of λ, we can reason that in the time *w*, a total of λ *W* items must have arrived. Thus $w = \lambda W$.
- To summarize, under steady state conditions, **the average number of items in a queuing system equals the average rate at which items arrive multiplied by the average time that an item spends in the system.**

Little’s Law

- Example:
- Consider a multicore system, with each core supporting multiple threads of execution. At some level, the cores share a common memory. The cores share a common main memory and typically share a common cache memory as well.

Little’s Law

- Example:
- If the system is being used as a server, an analyst can determine the demand on the system in terms of the rate of user requests, and then translate that into the rate of requests for data from the threads generated to respond to an individual user request.

Little’s Law

- Example:
- For this purpose, each user request is broken down into subtasks that are implemented as threads.
- We then have λ = the average rate of total thread processing required after all members’ requests have been broken down into whatever detailed subtasks are required.
- Define L as the average number of stopped threads waiting during some relevant time.
- Then W = average response time.
- This simple model can serve as a guide to designers as to whether user requirements are being met and, if not, provide a quantitative measure of the amount of improvement needed.

Basic Measures of Computer Performance

- Clock speed
- Instruction execution mode

Example on CPI and MIPS

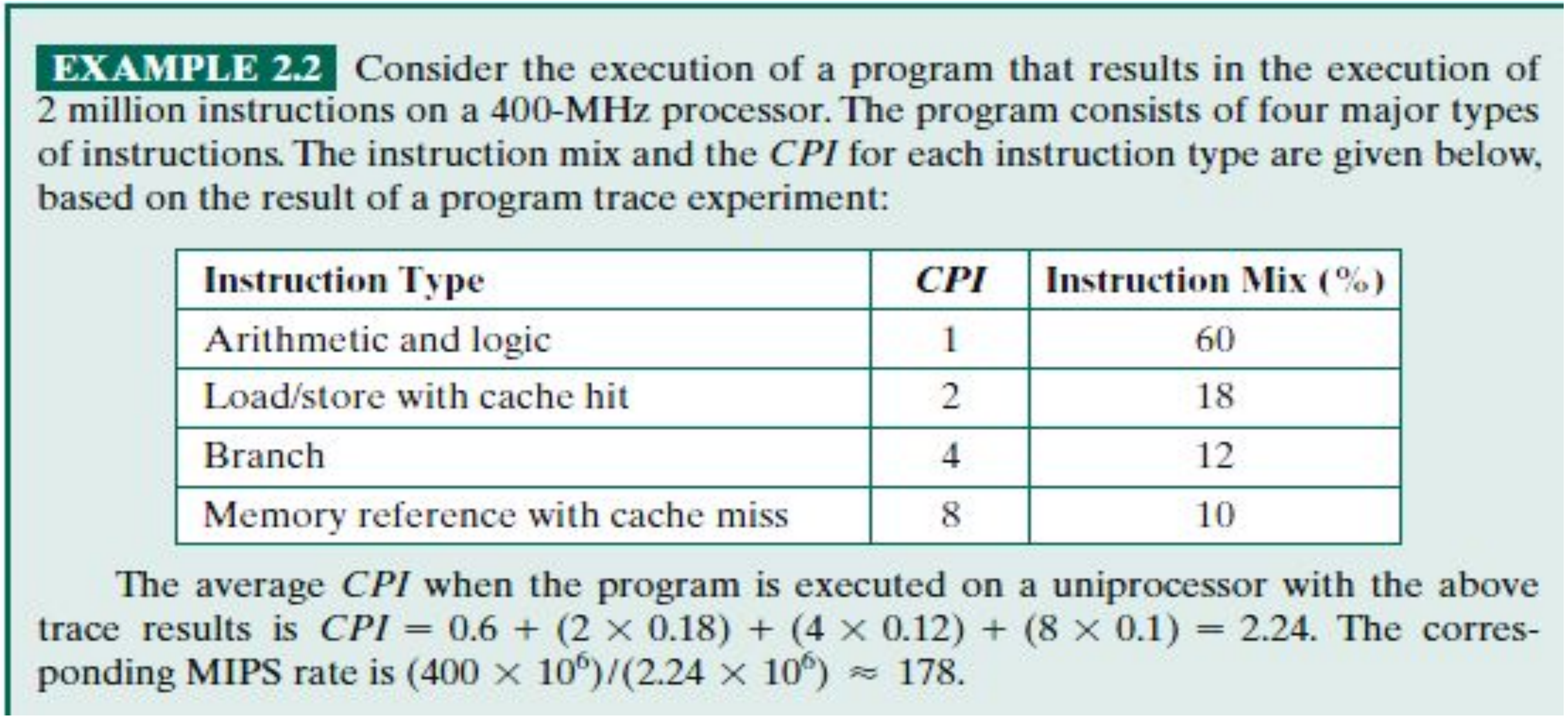
$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

The processor time T needed to execute a given program can be expressed as

$$T = I_c \times CPI \times \tau$$

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

Example on CPI and MIPS



EXAMPLE 2.2 Consider the execution of a program that results in the execution of 2 million instructions on a 400-MHz processor. The program consists of four major types of instructions. The instruction mix and the *CPI* for each instruction type are given below, based on the result of a program trace experiment:

Instruction Type	*CPI*	Instruction Mix (%)
Arithmetic and logic	1	60
Load/store with cache hit	2	18
Branch	4	12
Memory reference with cache miss	8	10

The average *CPI* when the program is executed on a uniprocessor with the above trace results is $CPI = 0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24$. The corresponding MIPS rate is $(400 \times 10^6)/(2.24 \times 10^6) \approx 178$.

Top level view of computer function and evolution

- a computer system by

(1) describing the external behaviour of each component—that is, the data and control signals that it exchanges with other components;

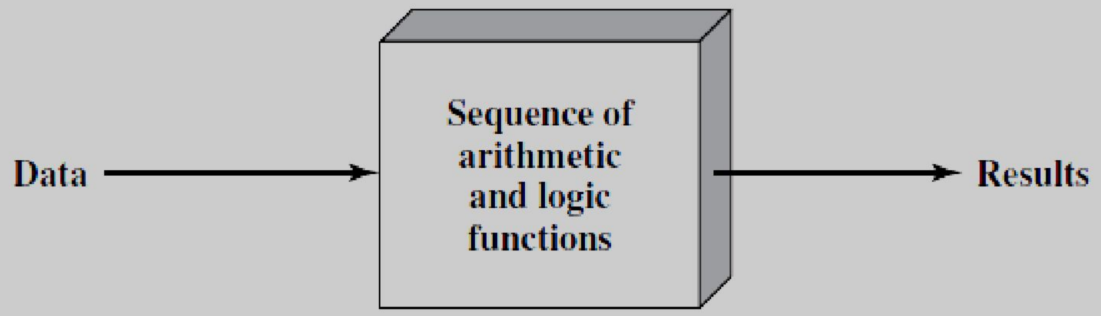
(2) describing the interconnection structure and the controls required to manage the use of the interconnection structure.

Computer components

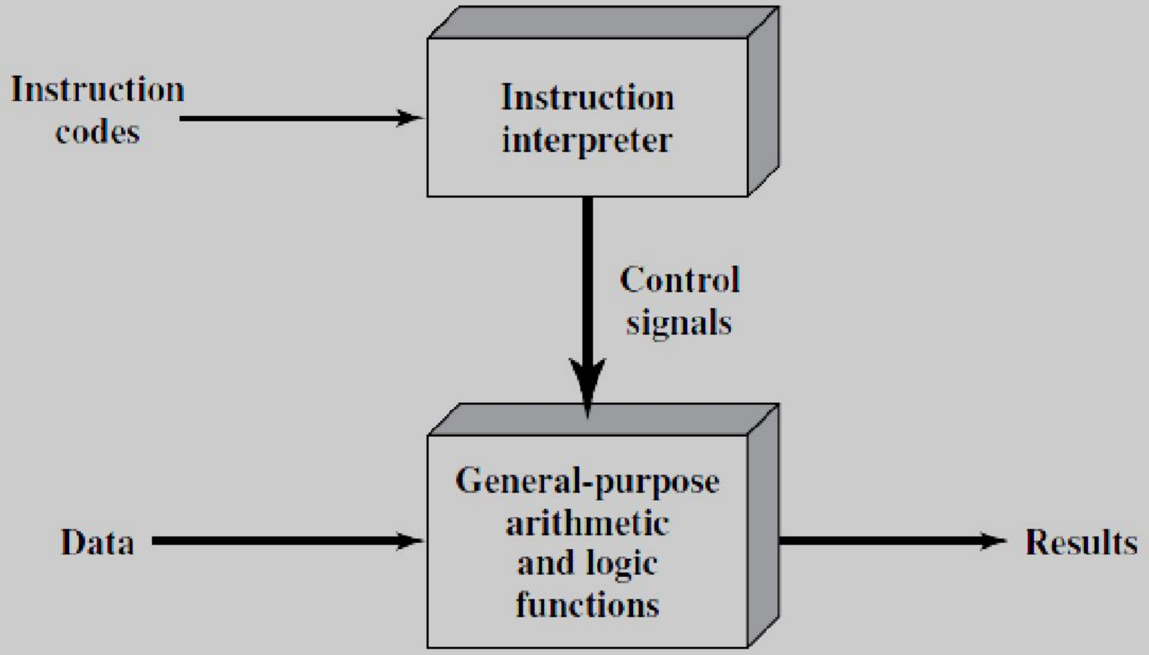
- *von Neumann architecture* and is based on three key concepts:

  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.

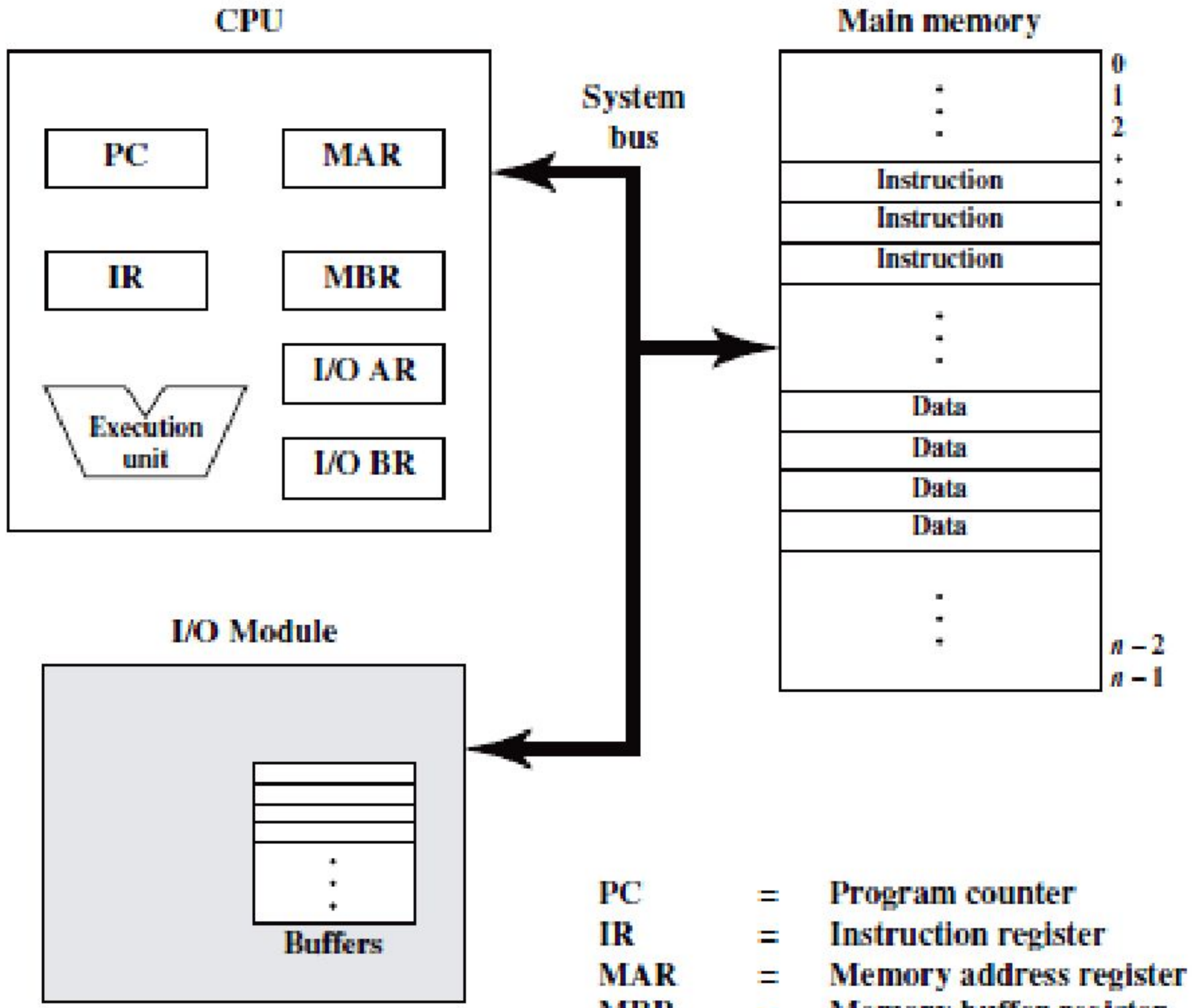


(a) Programming in hardware



(b) Programming in software

Computer function



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

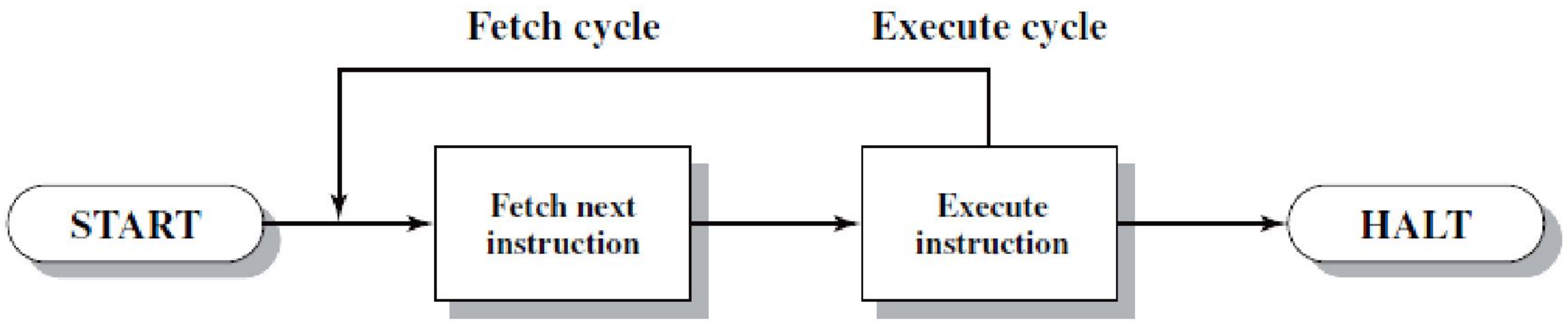


Figure 3.3 Basic Instruction Cycle

Basic instruction cycle

- The fetched instruction is loaded into a register in the processor known as the instruction register (IR). The instruction contains bits that specify the action the processor is to take. The processor interprets the instruction and performs the required action. In general, these actions fall into four categories:

• **Processor-memory:** Data may be transferred from processor to memory or

from memory to processor.

• **Processor-I/O:** Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module.

• **Data processing:** The processor may perform some arithmetic or logic operation on data.

• **Control:** An instruction may specify that the sequence of execution be altered. For example, the processor may fetch an instruction from location 149, which specifies that the next instruction be from location 182. The processor will remember this fact by setting the program counter to 182.

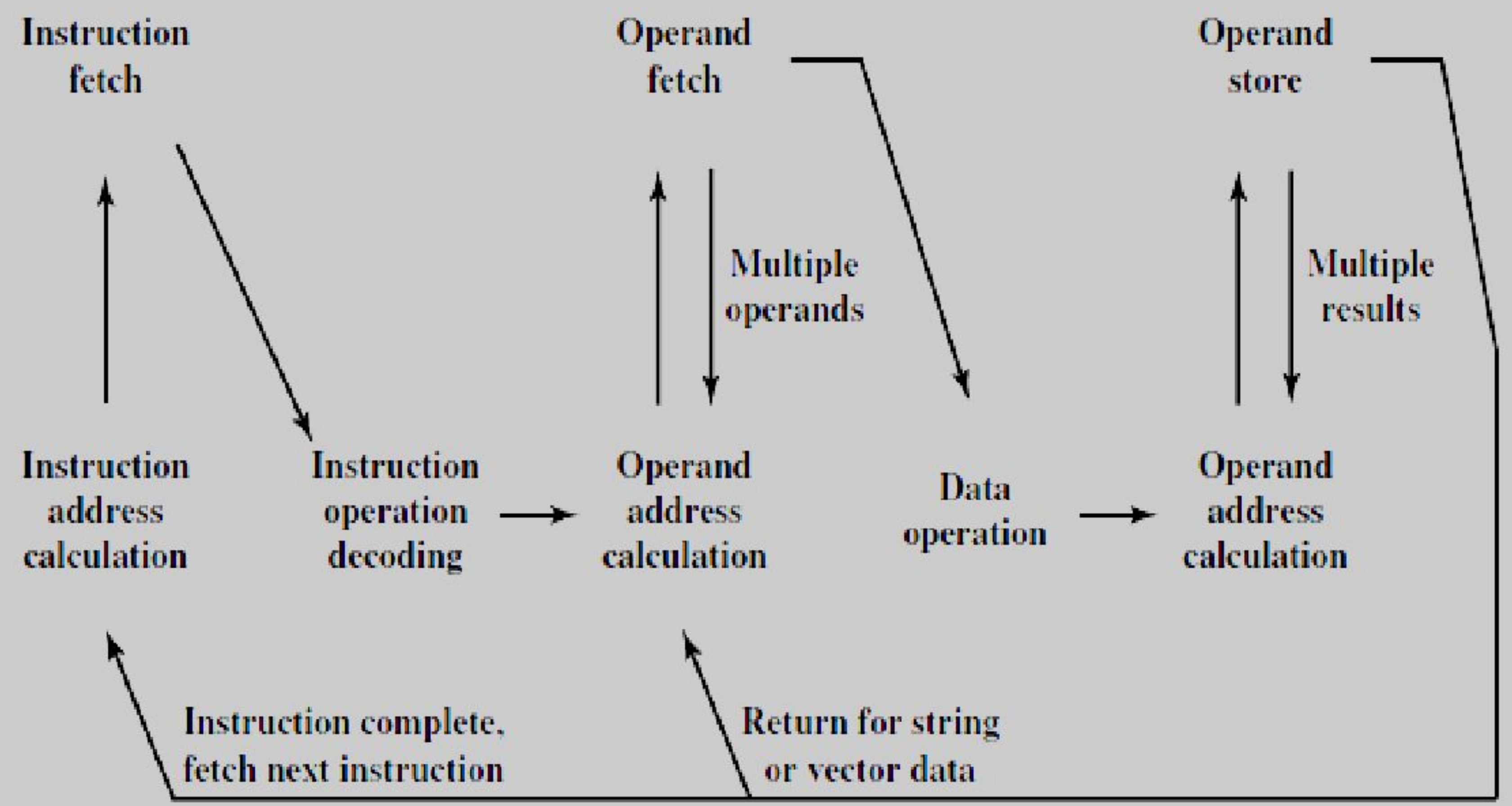

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data operation
Operand address calculation
Instruction complete, fetch next instruction
Return for string or vector data

Figure 3.6 Instruction Cycle State Diagram

Interrupts

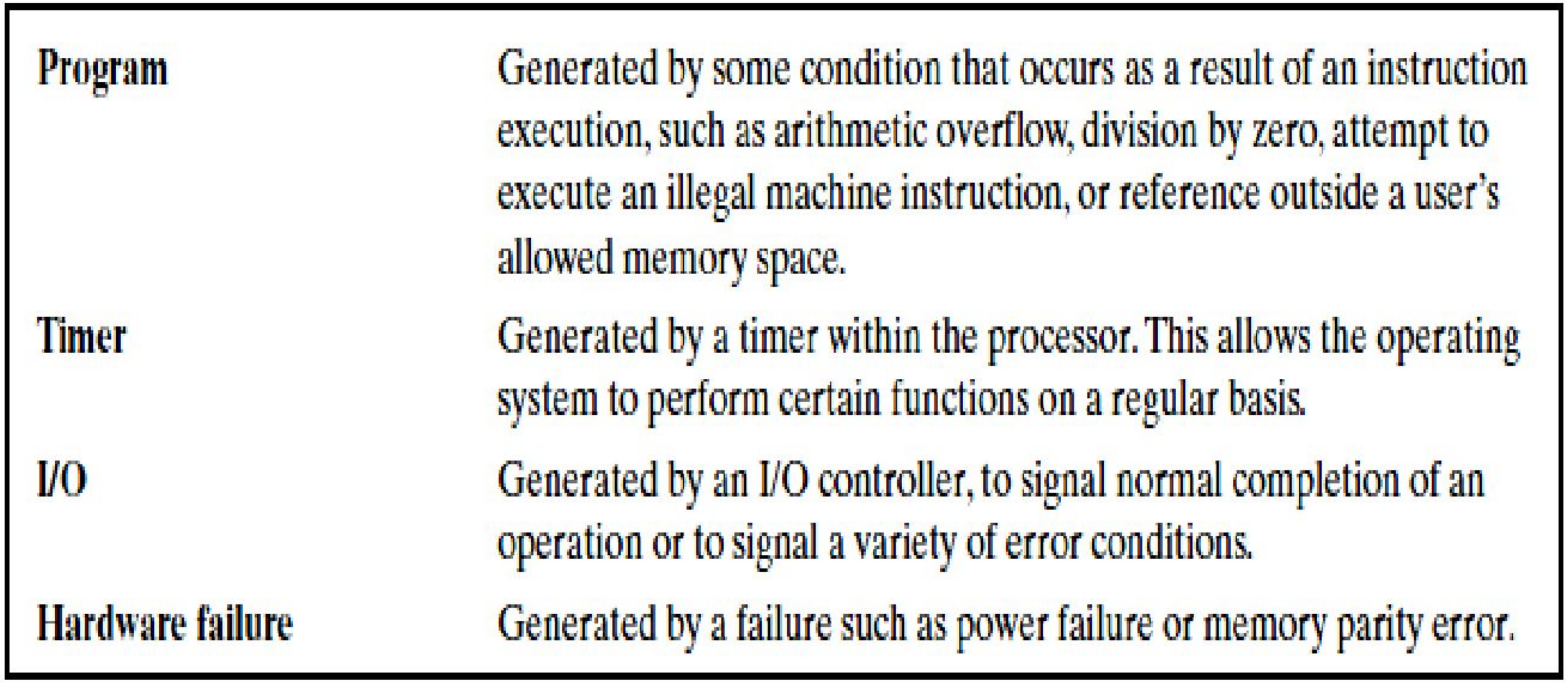


Table 3.1 Classes of Interrupts

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 or to signal a variety of error conditions.
Hardware failure	Generated by a failure such as power failure or memory parity error.

Interrupts

- communication between modules that results from interrupts
- For example, most external devices are much slower than the processor.
- Long and short IO
- Multiple interrupts: disabled and ISR

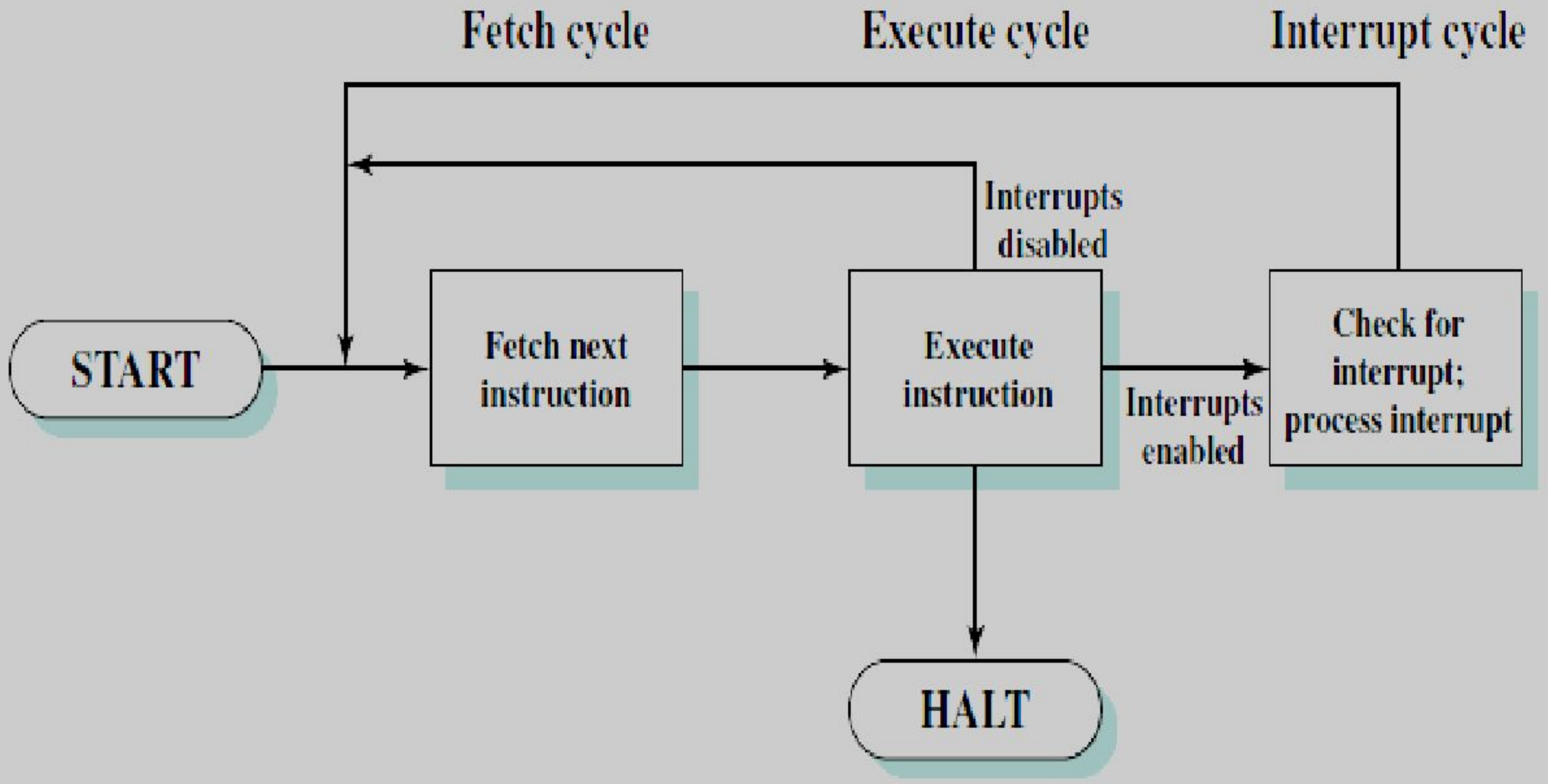


Figure 3.9 Instruction Cycle with Interrupts

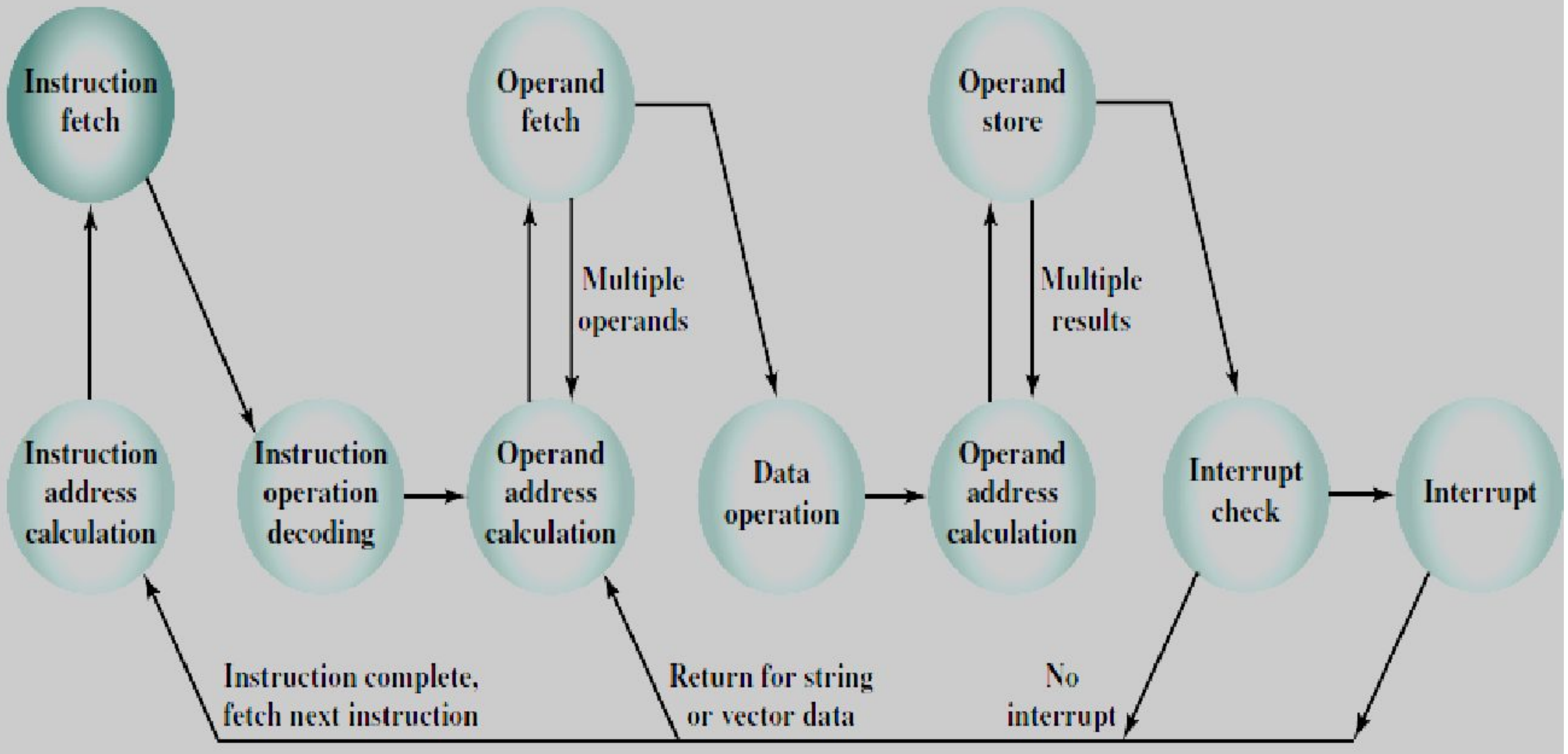
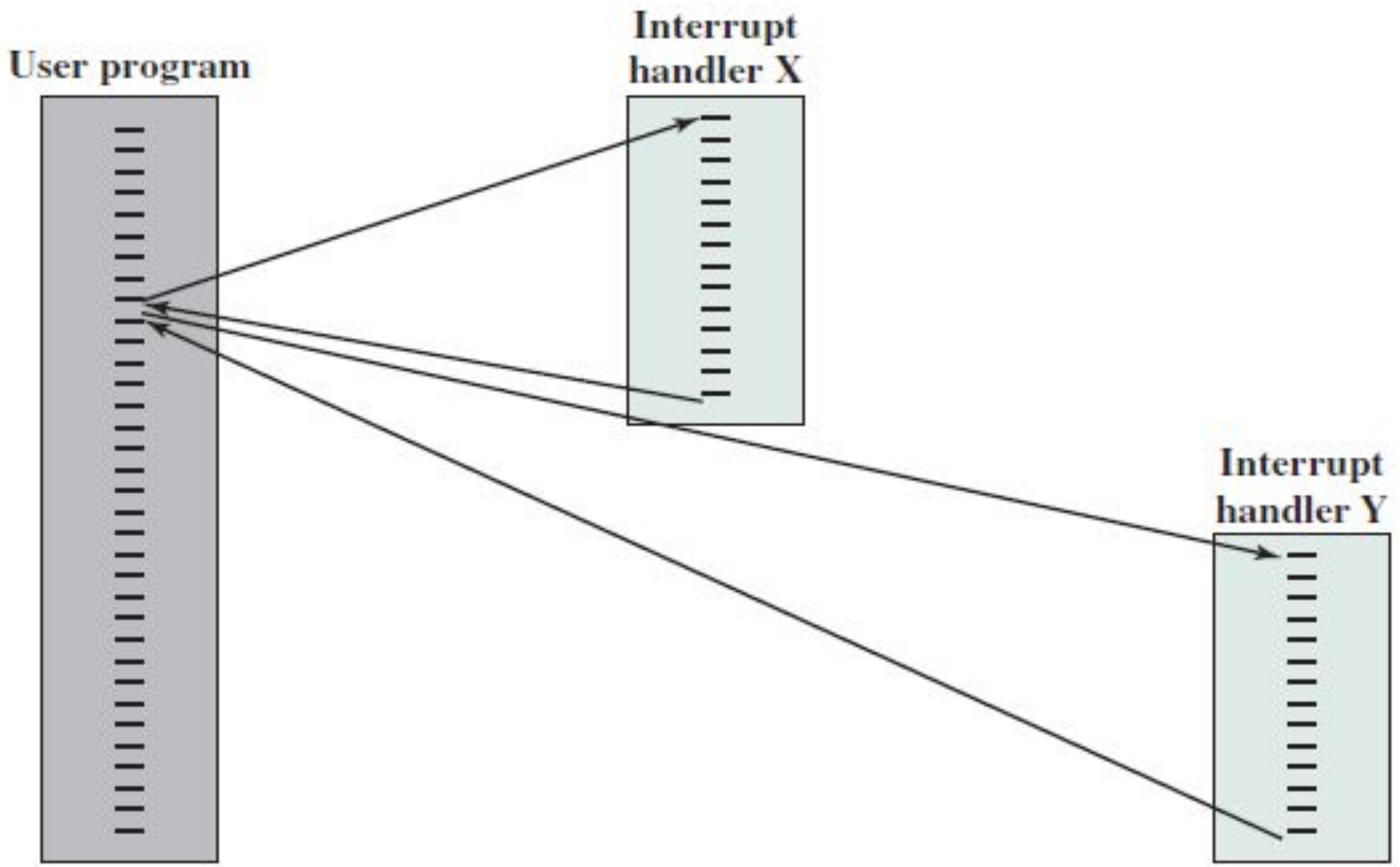
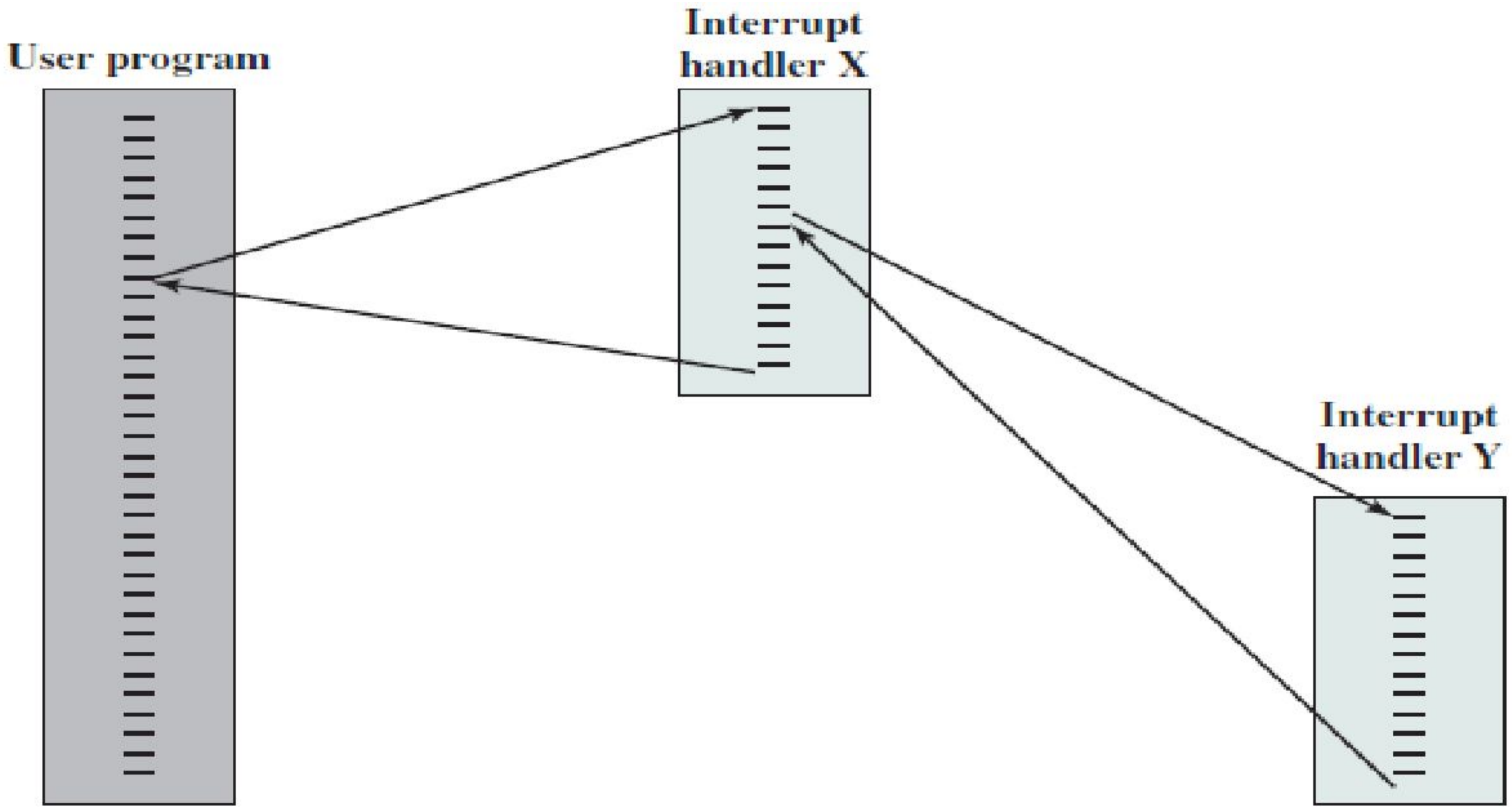

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data operation
Operand address calculation
Interrupt check
Interrupt
Instruction complete, fetch next instruction
Return for string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

Figure 3.12 Instruction Cycle State Diagram, with Interrupts



(a) Sequential interrupt processing



(b) Nested interrupt processing

Figure 3.13 Transfer of Control with Multiple Interrupts

Interconnection structures

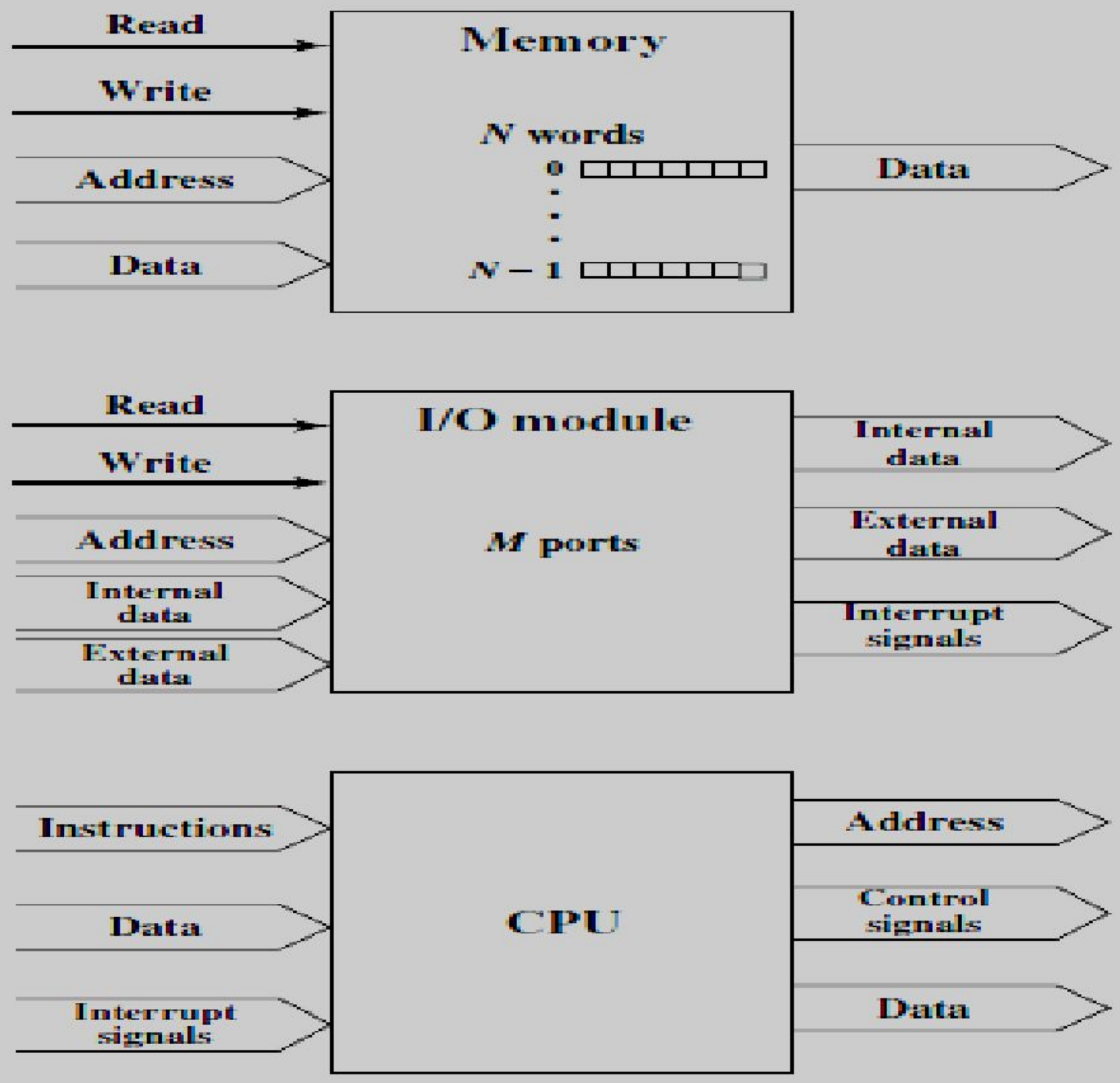


Figure 3.15 Computer Modules

- Bus interconnection: shared transmission medium
  - System bus
  - Data bus
  - Address bus
  - Control bus: both command and timing information
- Point to point interconnection

- **Memory write:** causes data on the bus to be written into the addressed location.
- **Memory read:** causes data from the addressed location to be placed on the bus.
- **I/O write:** causes data on the bus to be output to the addressed I/O port.
- **I/O read:** causes data from the addressed I/O port to be placed on the bus.
- **Transfer ACK:** indicates that data have been accepted from or placed on the bus.
- **Bus request:** indicates that a module needs to gain control of the bus.
- **Bus grant:** indicates that a requesting module has been granted control of the bus.
- **Interrupt request:** indicates that an interrupt is pending.
- **Interrupt ACK:** acknowledges that the pending interrupt has been recognized.
- **Clock:** is used to synchronize operations.
- **Reset:** initializes all modules.

Bus interconnection

- An instruction cycle consists of an instruction fetch, followed by zero or more operand fetches, followed by zero or more operand stores, followed by an interrupt check (if interrupts are enabled).
- The major computer system components (processor, main memory, I/O modules) need to be interconnected in order to exchange data and control signals: shared system bus consisting of multiple lines.
- Key design elements for buses include arbitration (whether permission to send signals on bus lines is controlled centrally or in a distributed fashion): timing and width design decisions.

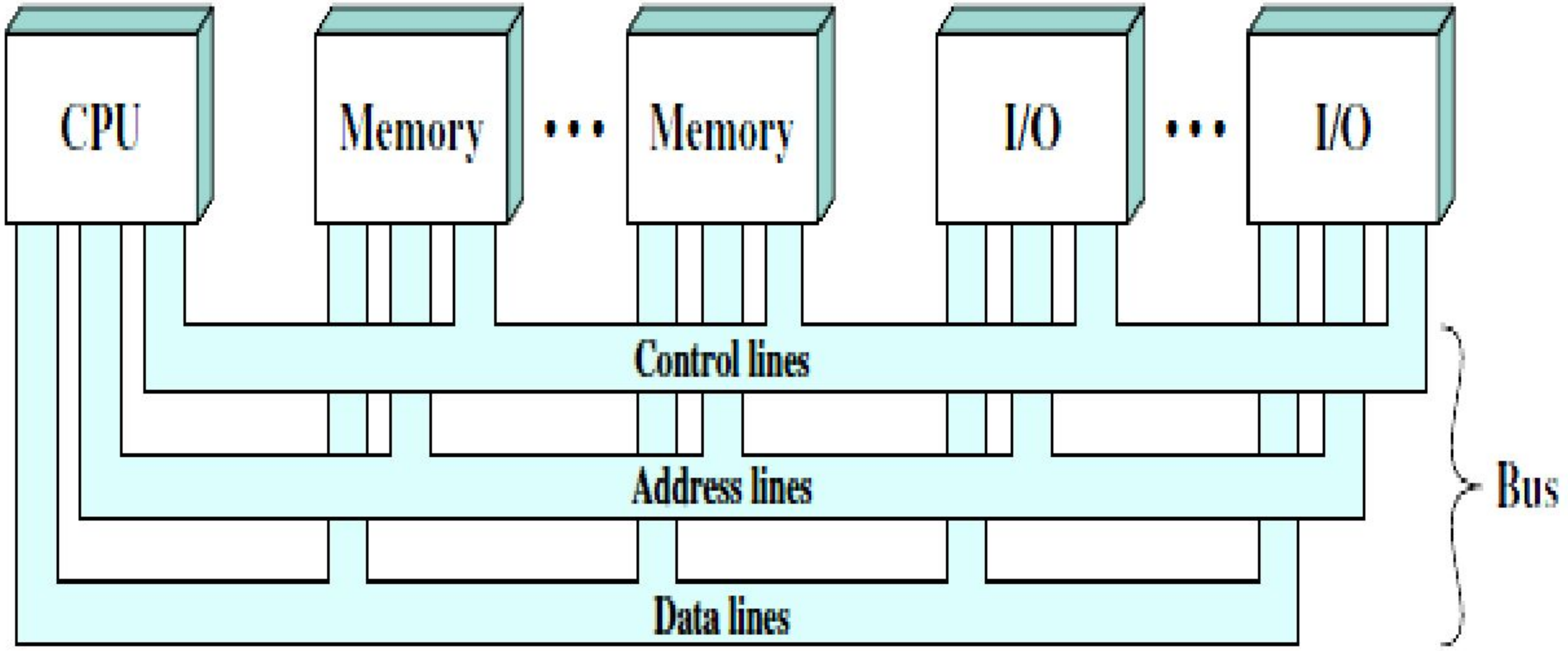


Figure 3.16 Bus Interconnection Scheme

Quick Path Interconnect (QPI)

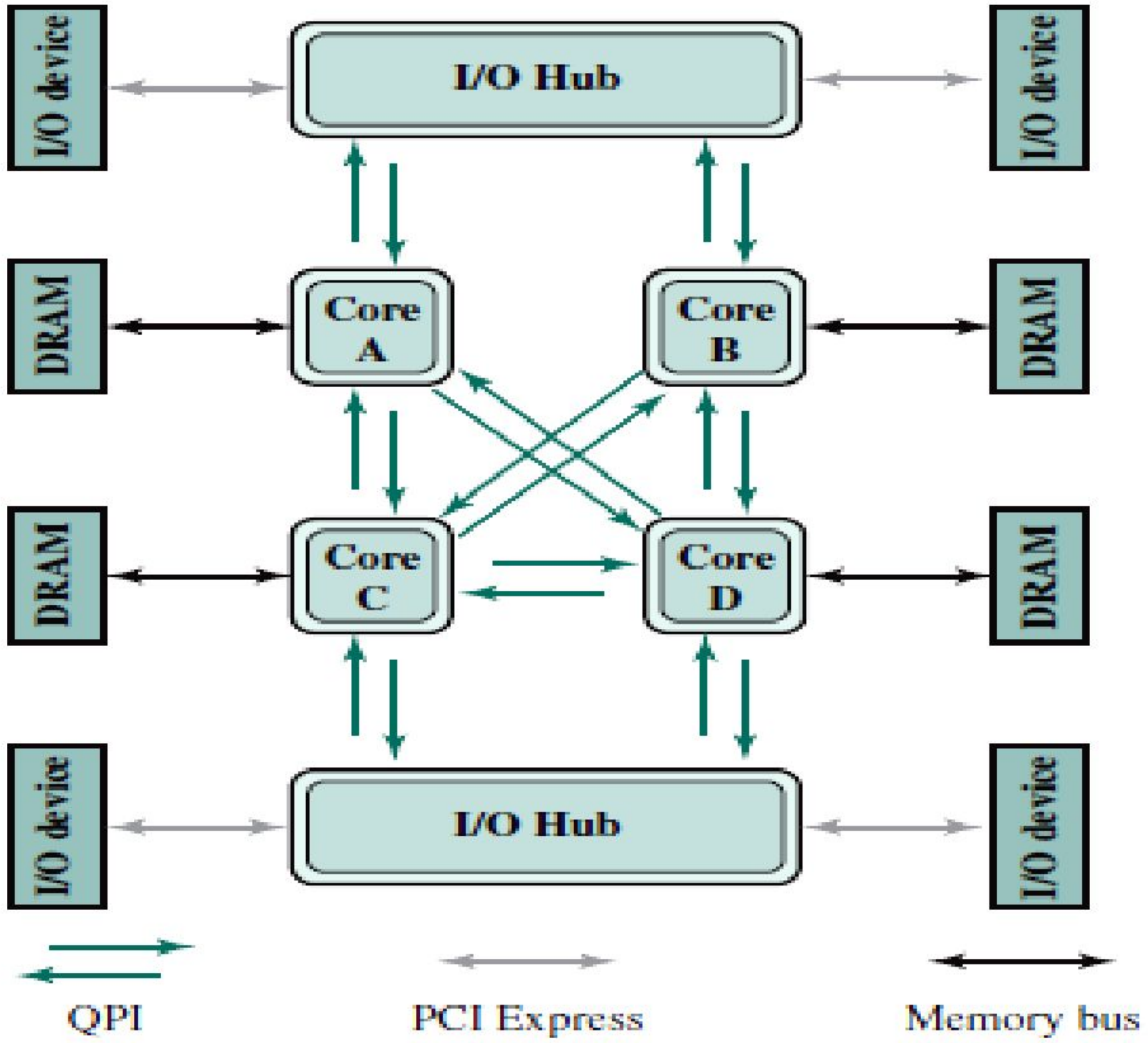
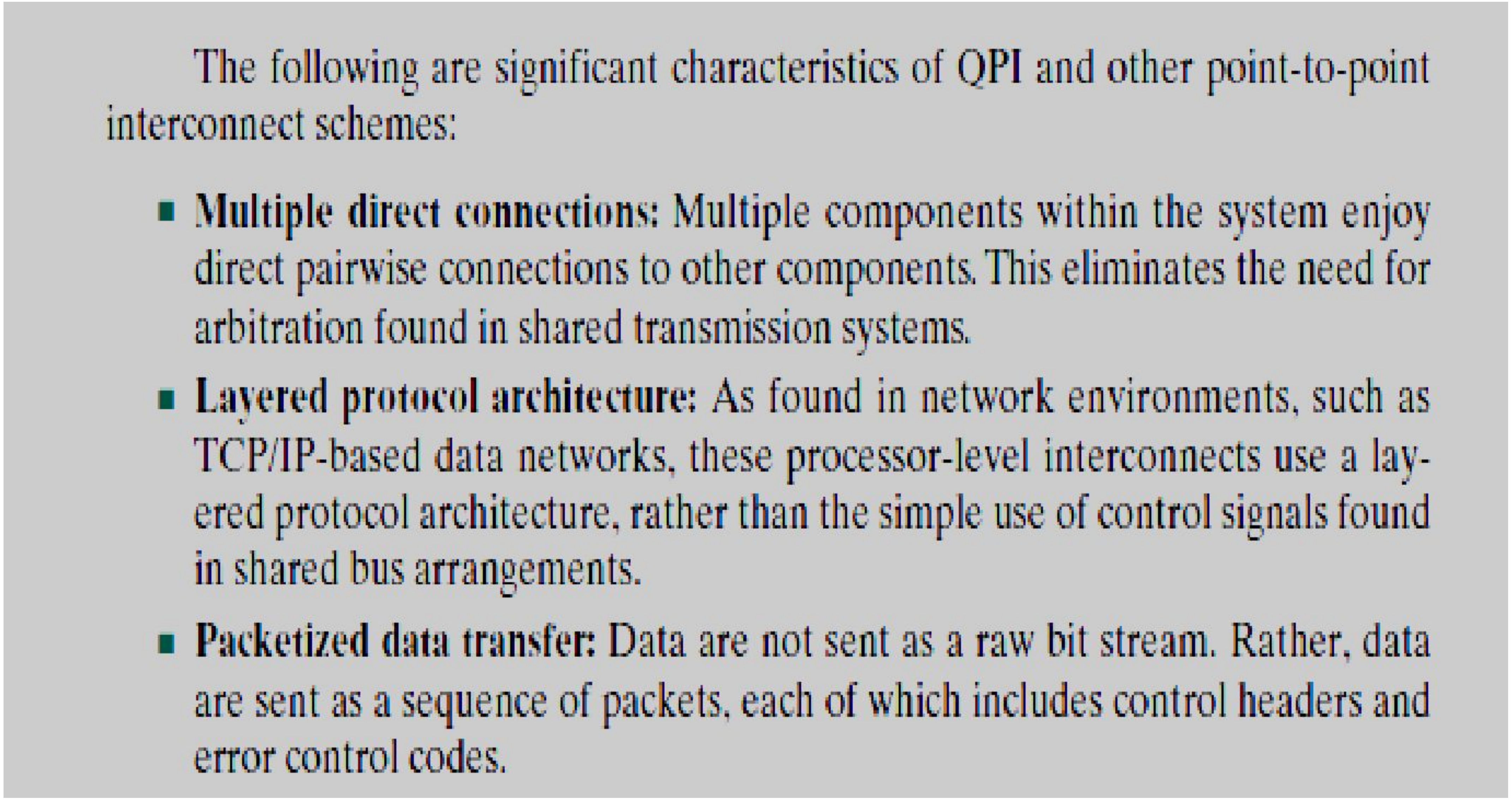


Figure 3.17 Multicore Configuration Using QPI

Point to point interconnection



The following are significant characteristics of QPI and other point-to-point interconnect schemes:

- **Multiple direct connections:** Multiple components within the system enjoy direct pairwise connections to other components. This eliminates the need for arbitration found in shared transmission systems.
- **Layered protocol architecture:** As found in network environments, such as TCP/IP-based data networks, these processor-level interconnects use a lay-ered protocol architecture, rather than the simple use of control signals found in shared bus arrangements.
- **Packetized data transfer:** Data are not sent as a raw bit stream. Rather, data are sent as a sequence of packets, each of which includes control headers and error control codes.

PCI Express

- The **peripheral component interconnect (PCI)** is a popular high-bandwidth, processor-independent bus.
- PCI delivers better system performance for high-speed I/O subsystems .
- e.g., graphic display adapters, network interface controllers, and disk controllers).

- A key requirement for PCIe 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.
- Applications such as video-on-demand and audio redistribution are putting real-time constraints on servers.

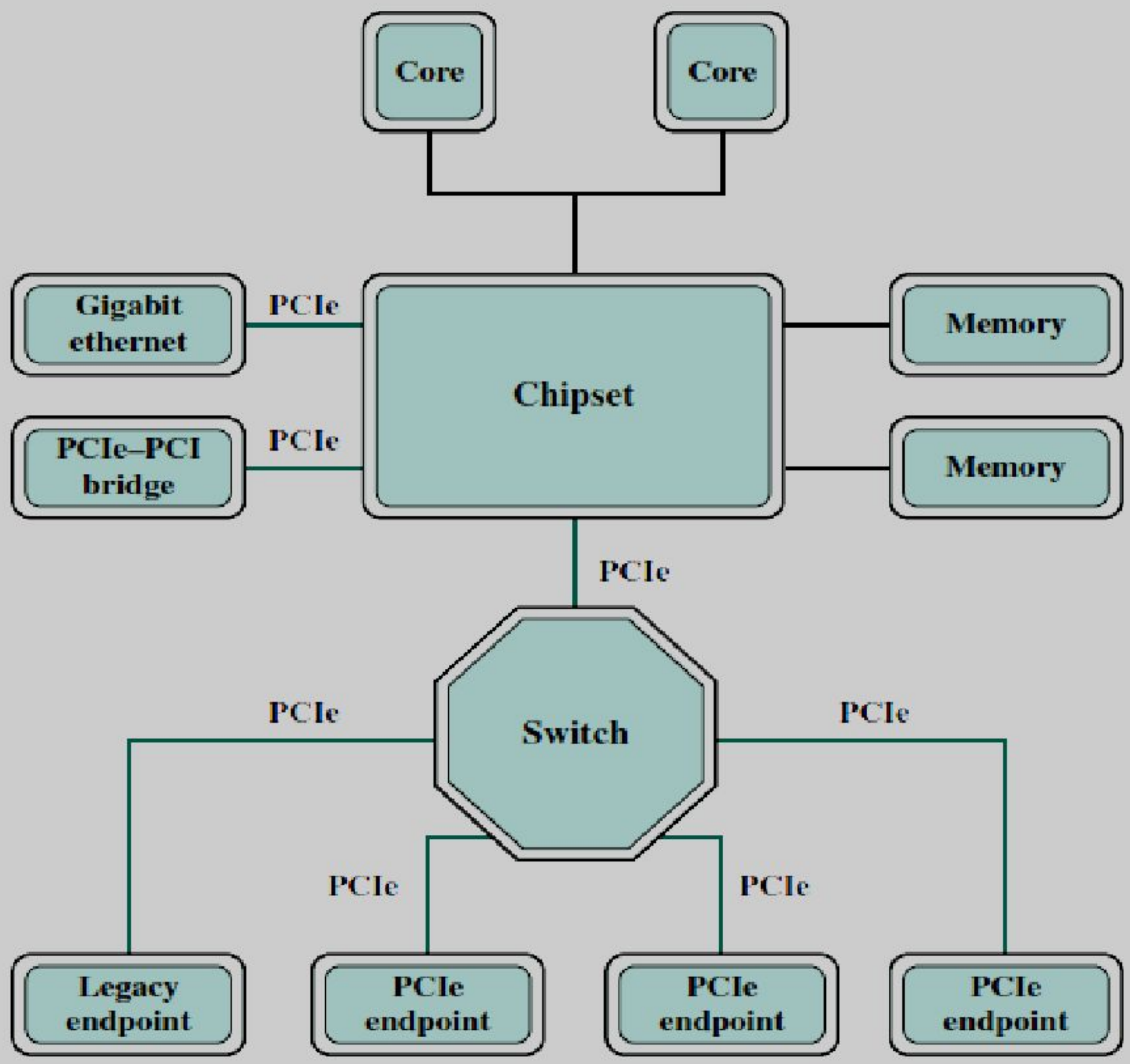


Figure 3.21 Typical Configuration Using PCIe

Thank you

```
1000  Start
1001  OP1 02H
1003  OP2 02H
1004  ADD OP1 OP2
1005  Store result OP3
1006   END
```

```
      1000  Start
      1001  OP1 02H
1003      OP2 02H
      1004   INR 1
      1005  ADD OP1 OP5
      1006  Store result OP3
      1007   END
```

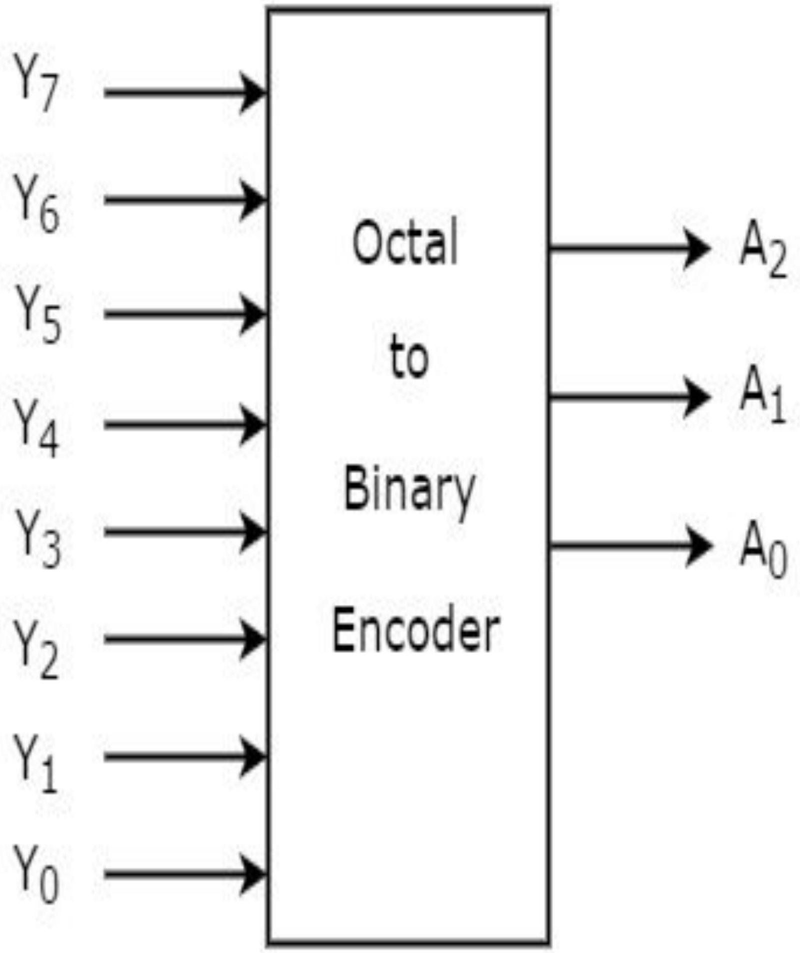
```
2000 Start
2001 OP4 01H
2003  ADD OP4 OP2
2004  Store result OP5
2005  END (Return)
```

```
1000  Start
1001  OP1 02H
1003  OP2 02H
1004  INR 1
1005  ADD OP1 OP7
1006  Store result OP3
1007  END
```

```
2000  Start
2001  OP4 01H
2003  ADD OP4 OP2
2004  Store result OP5
2005  INR 2
2006  END (Return)
```

```
3000 Start
3001 OP6 01H
3003  ADD OP6 OP5
3004  Store result OP7
3005  END (Return)
```

Octal to Binary Encoder



Truth Table

Input	A2	A1	A0
D0			
D1			
D2			
D3			
D4			
D5			
D6			
D7			

(7 6 1) = ( )