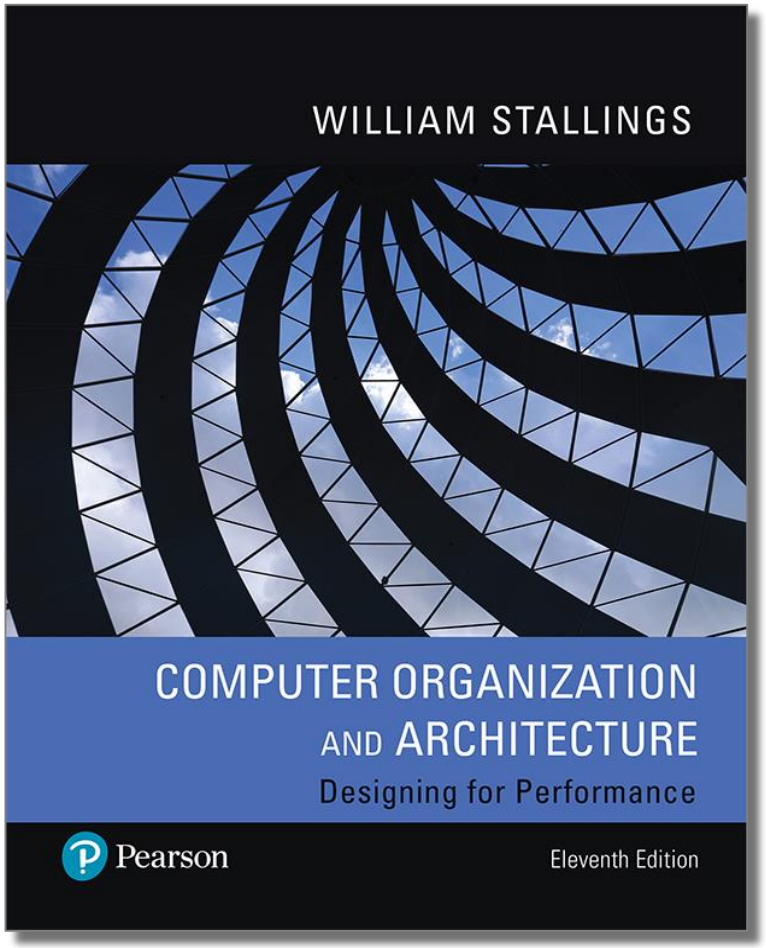
Computer Organization and Architecture

Designing for Performance

11th Edition



Chapter 3

A Top-Level View of Computer Function and 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*
  - The result of the process of connecting the various components in the desired configuration

Hardware and Software Approaches

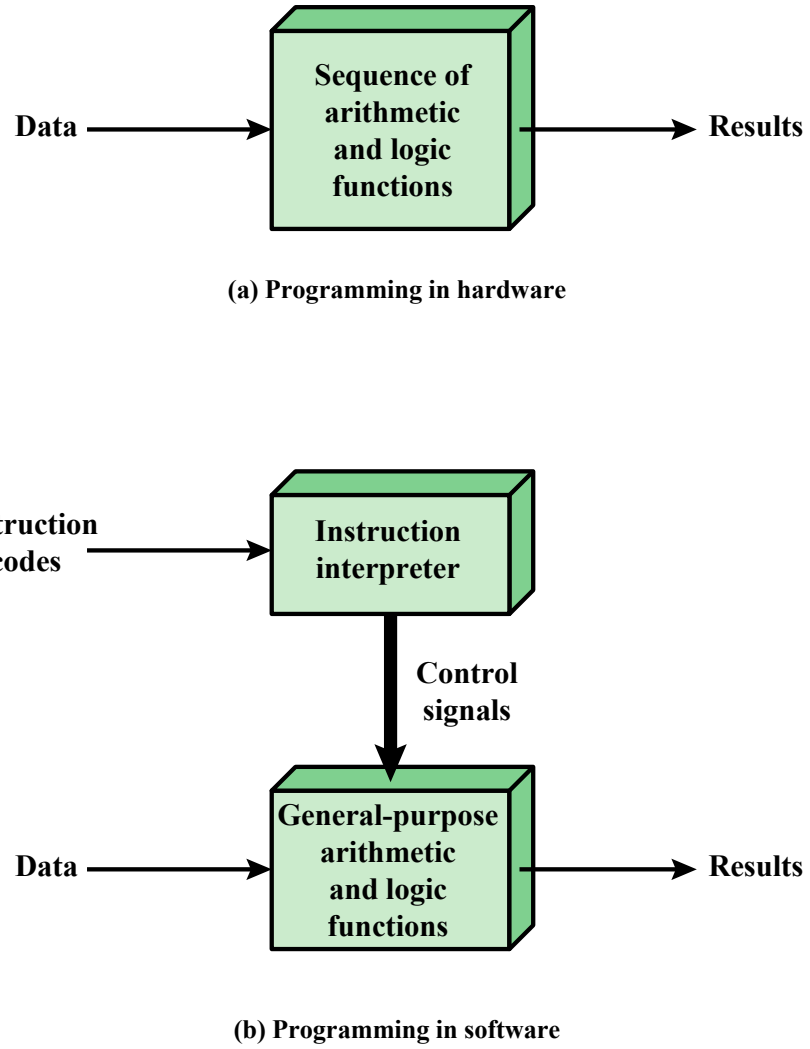


Figure 3.1 Hardware and Software Approaches

Software and I/O Components

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

Memory, MAR, and MBR

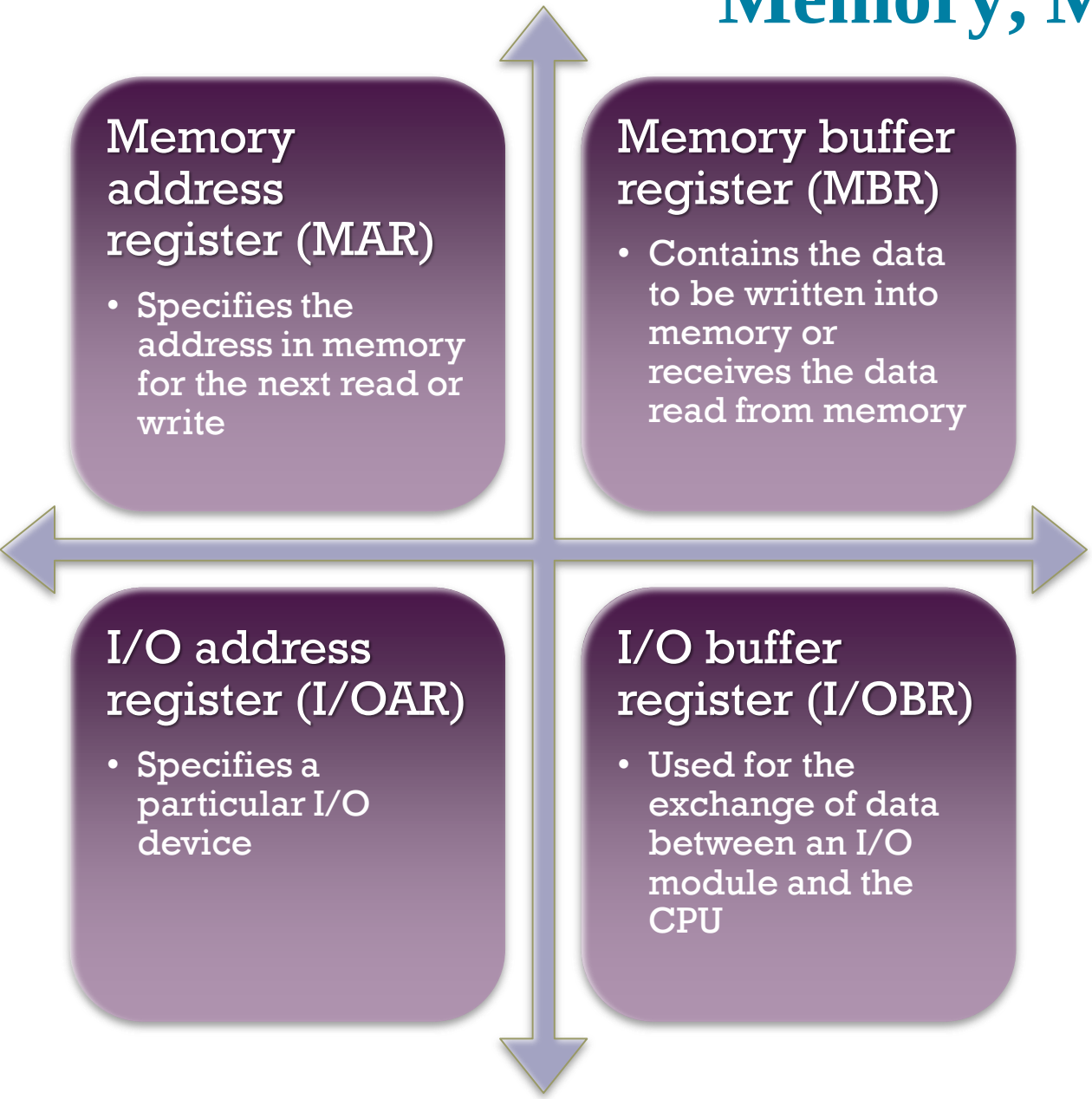


Figure 3.2

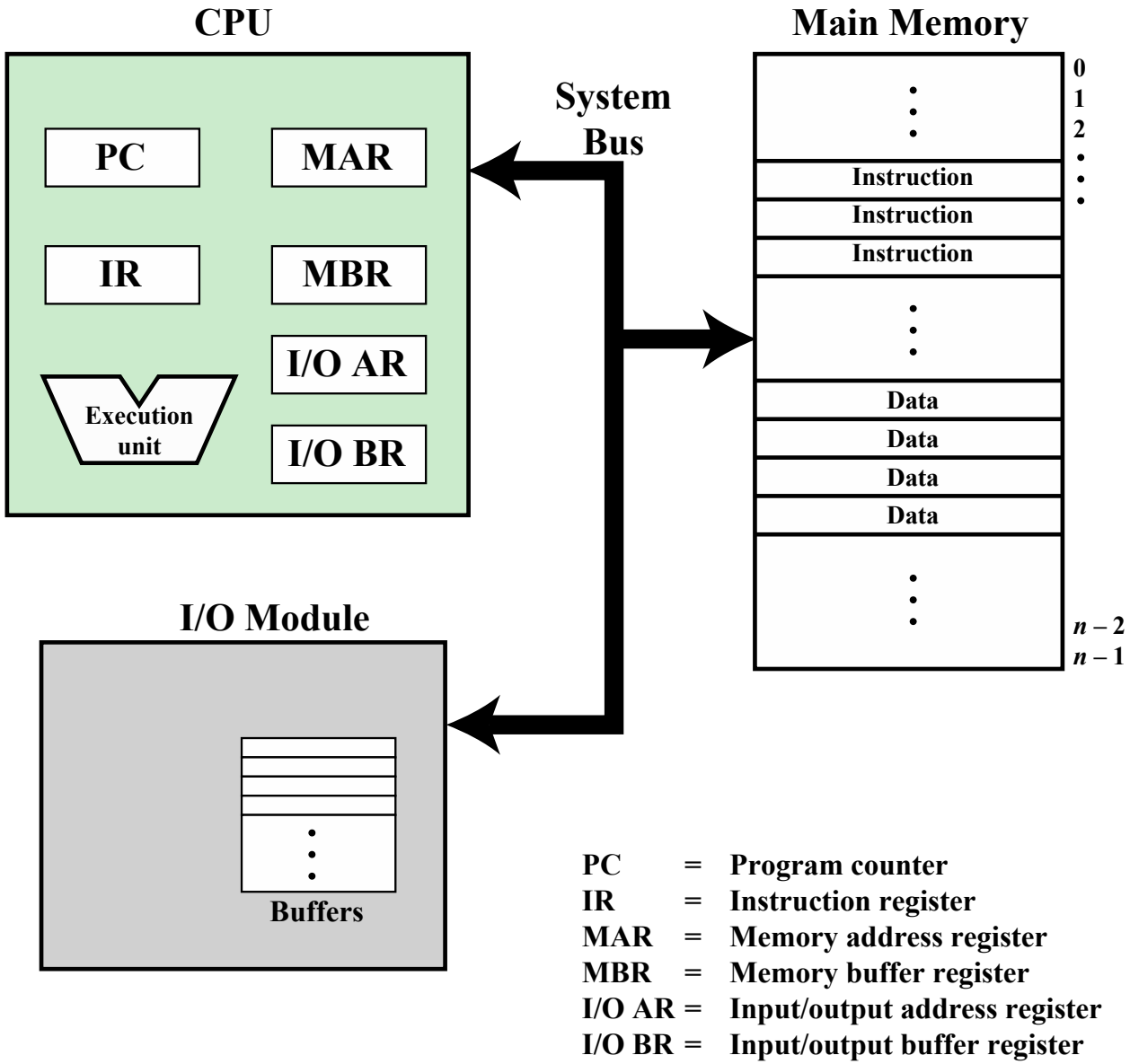


Figure 3.2 Computer Components: Top-Level View

Figure 3.3

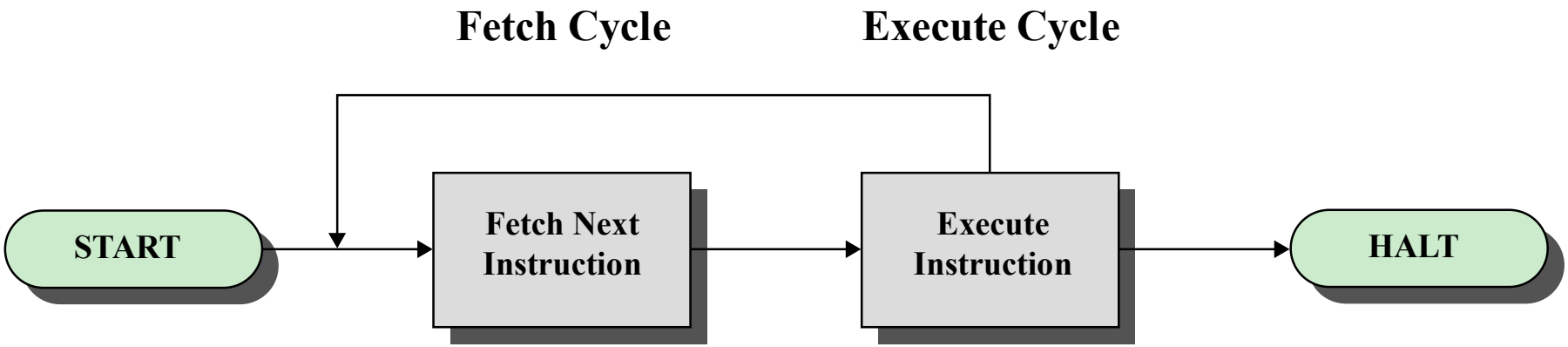


Figure 3.3 Basic Instruction Cycle

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

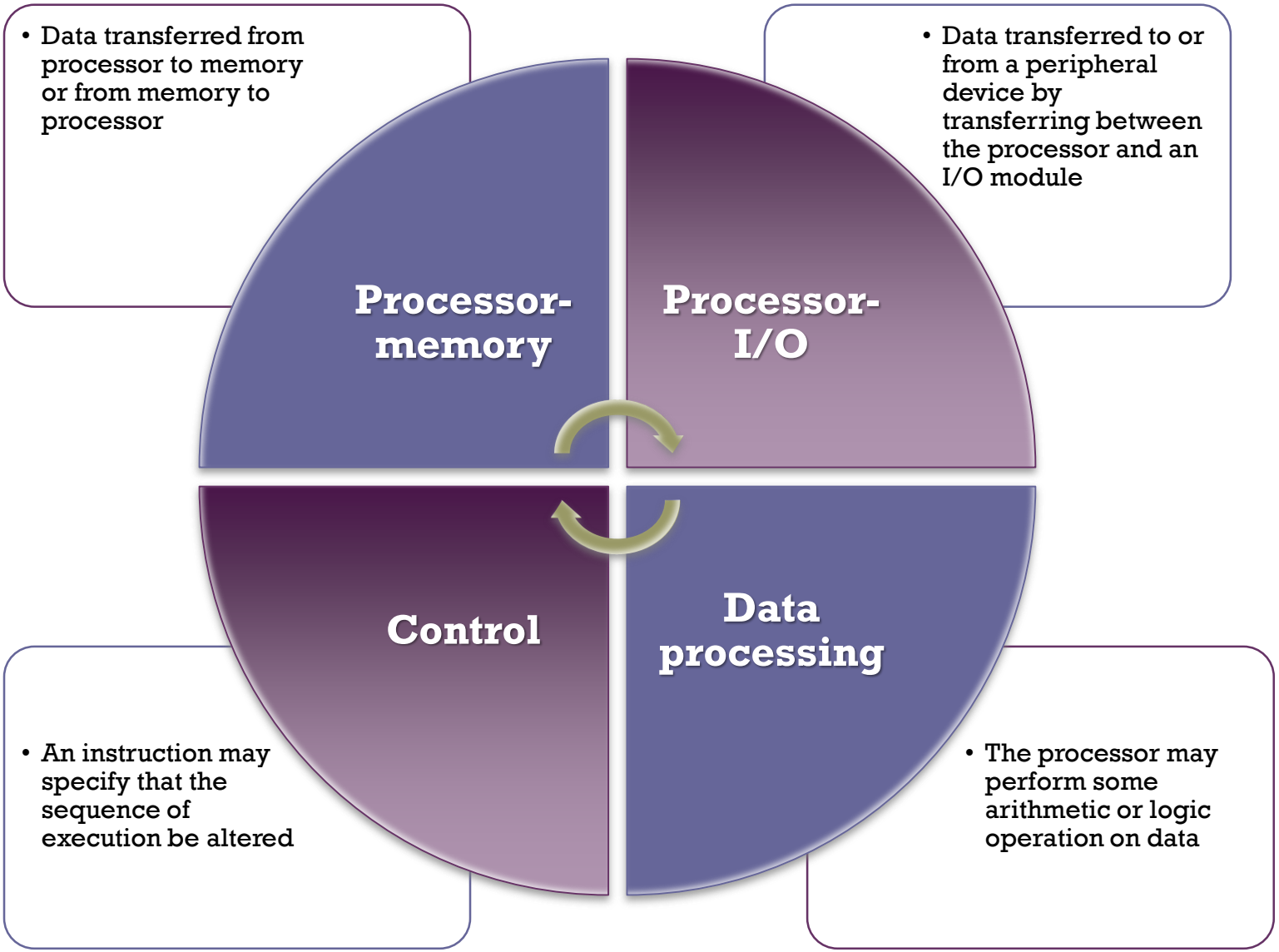
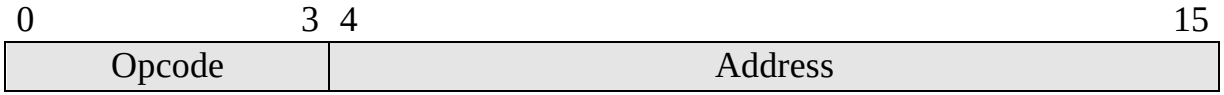
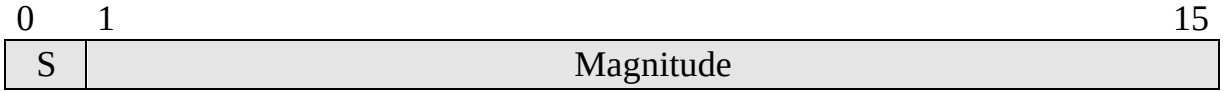


Figure 3.4



(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

Figure 3.5

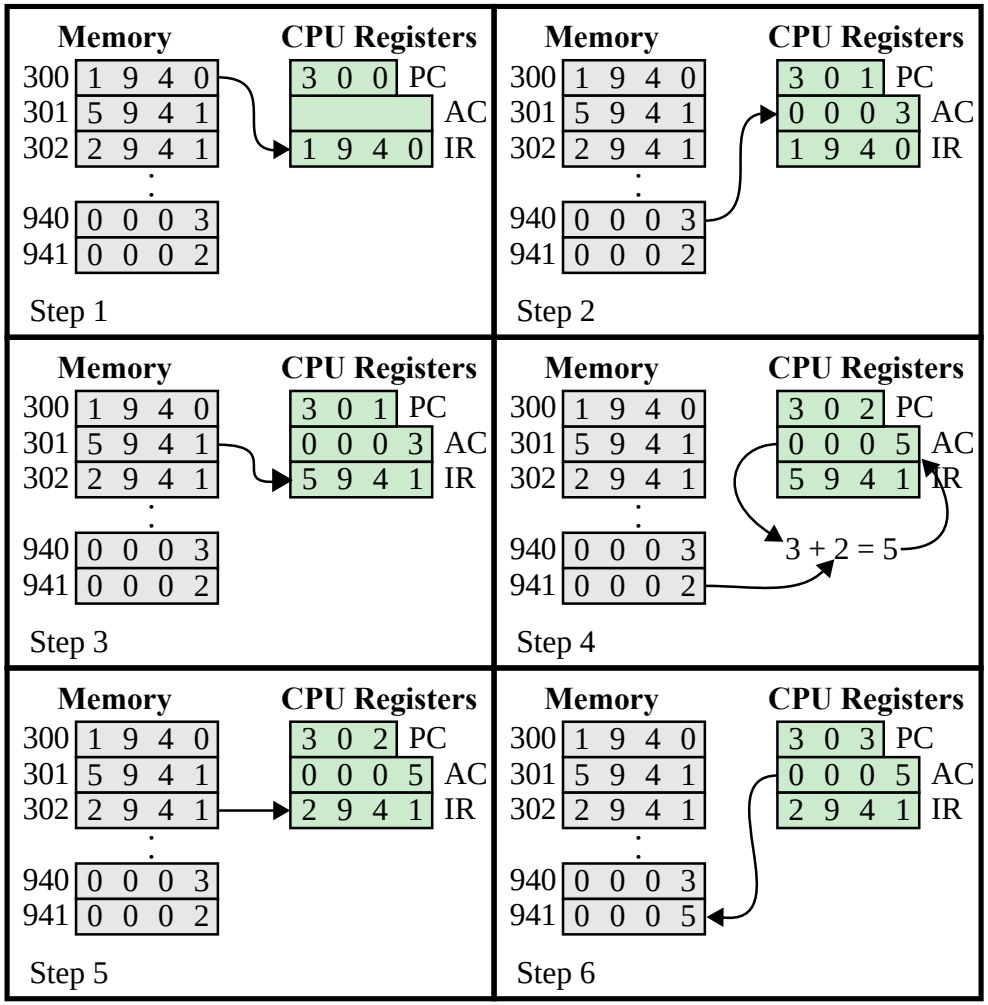


Figure 3.5 Example of Program Execution (contents of memory and registers in hexadecimal)

Figure 3.6

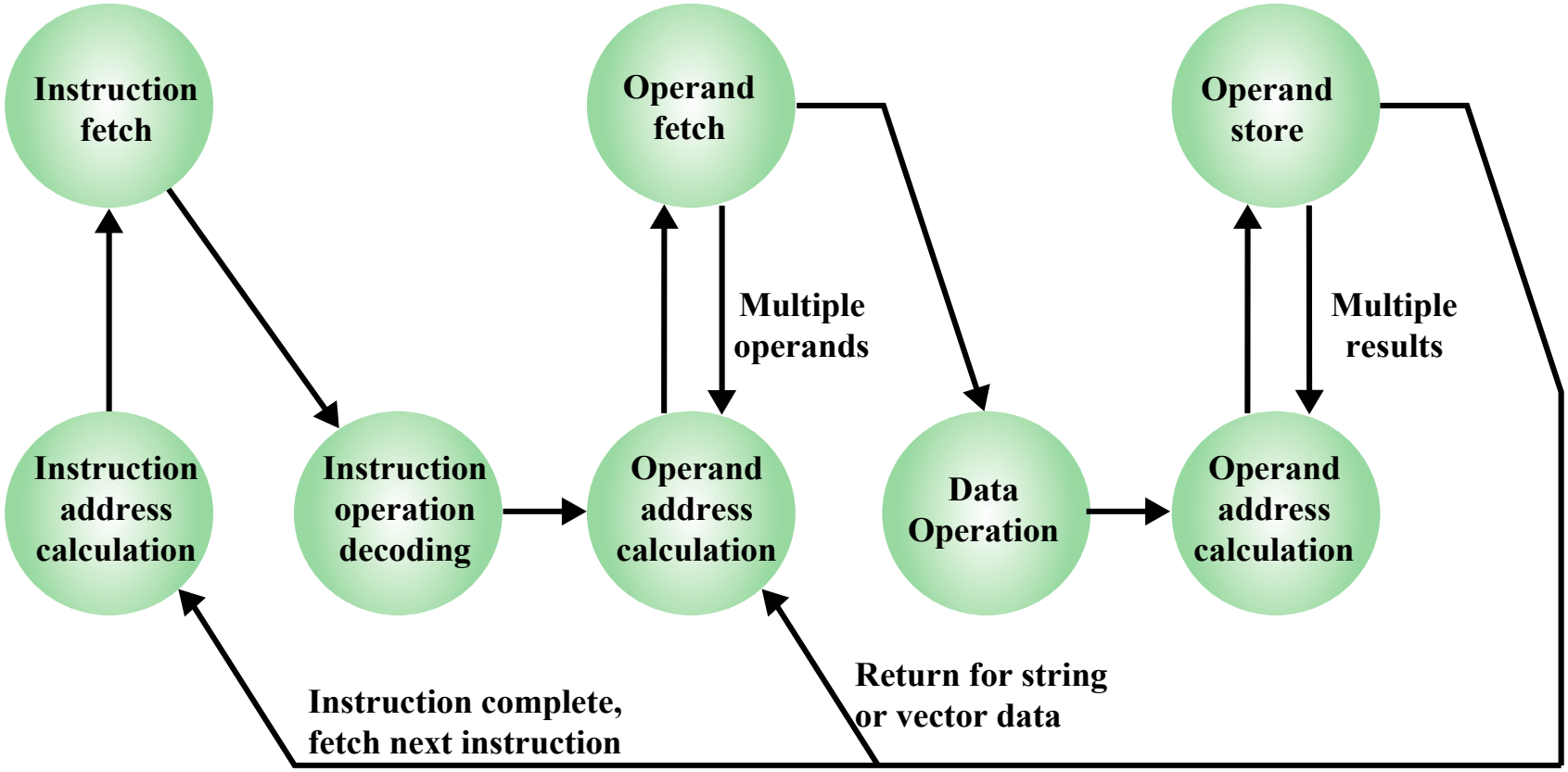


Figure 3.6 Instruction Cycle State Diagram

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.

Figure 3.7

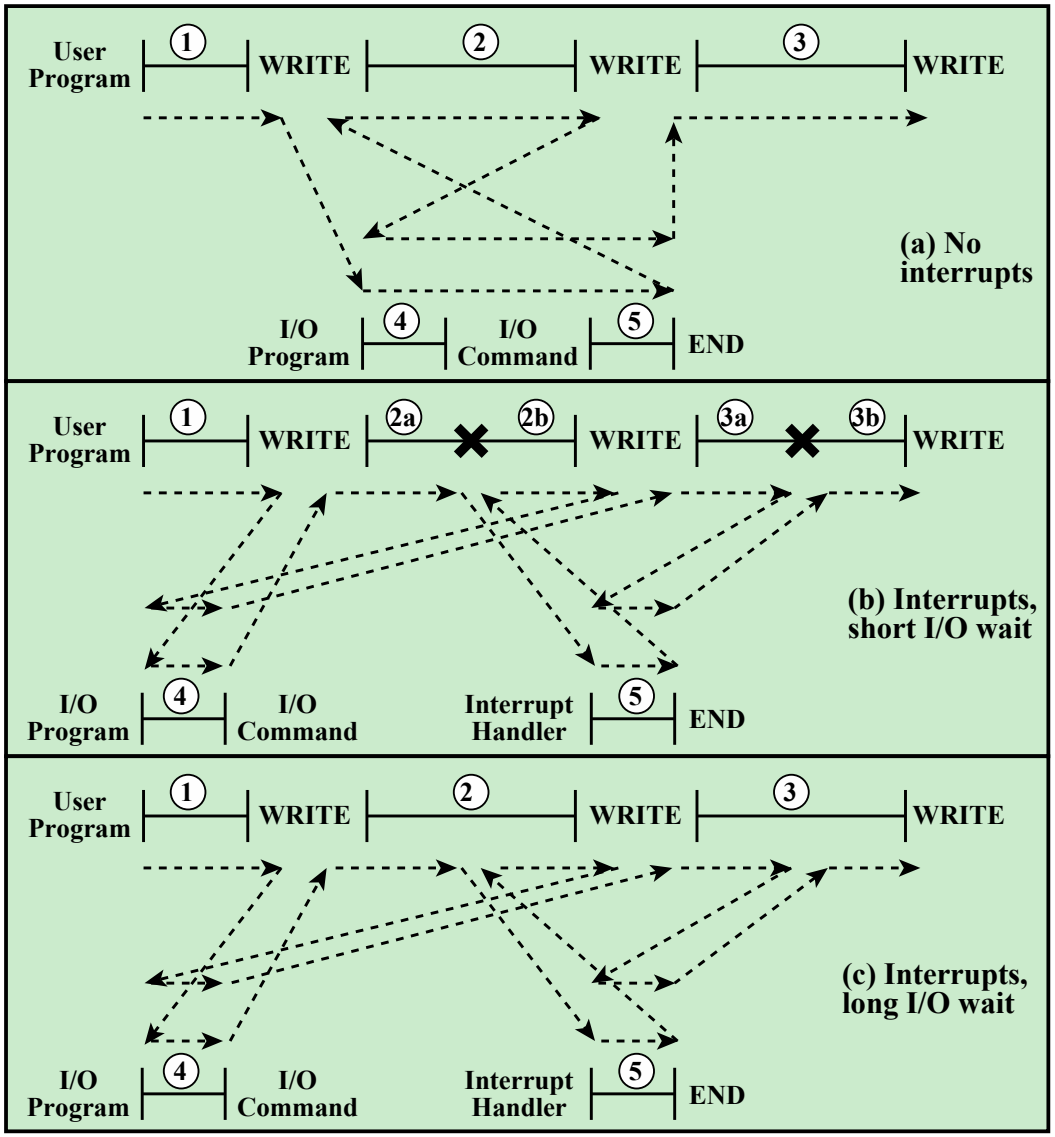


Figure 3.7 Program Flow of Control Without and With Interrupts

Figure 3.8

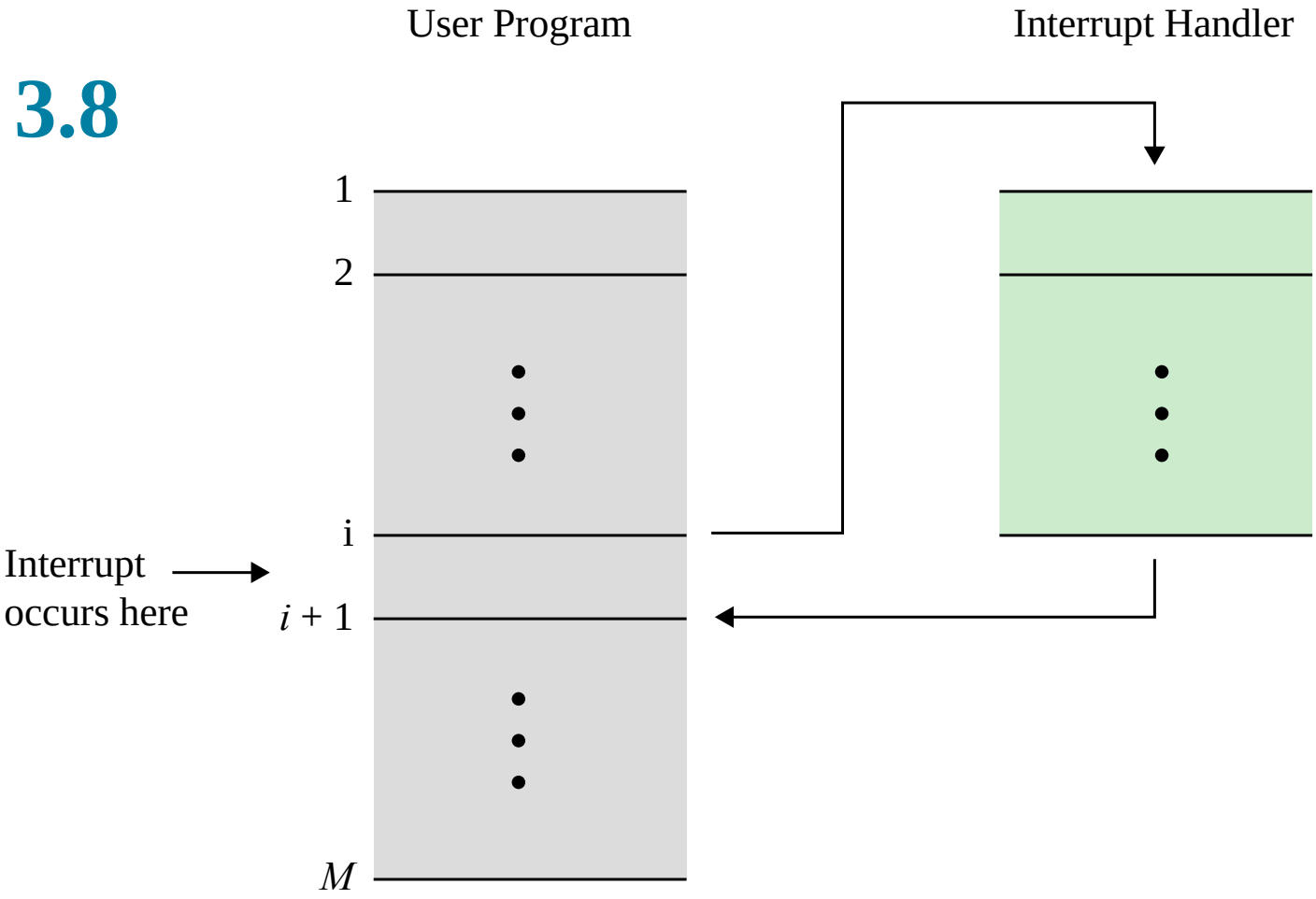


Figure 3.8 Transfer of Control via Interrupts

Figure 3.9

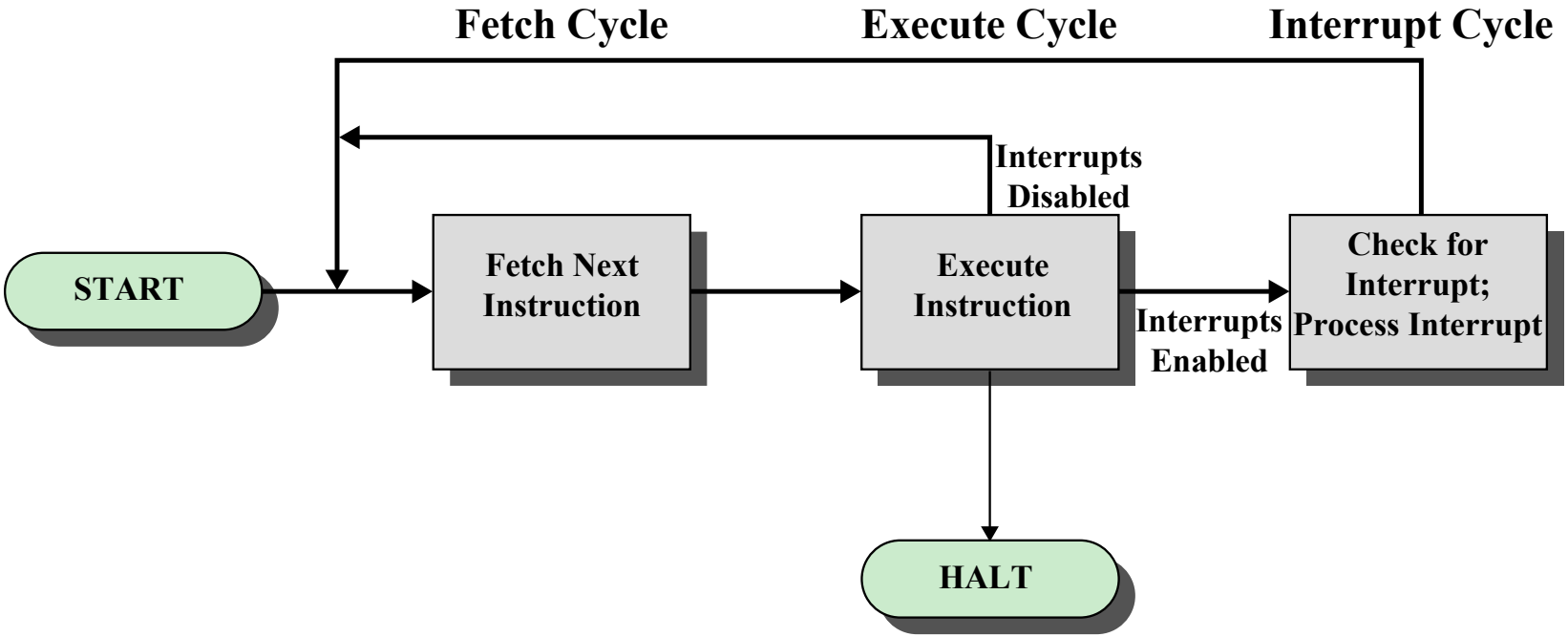


Figure 3.9 Instruction Cycle with Interrupts

Figure 3.10

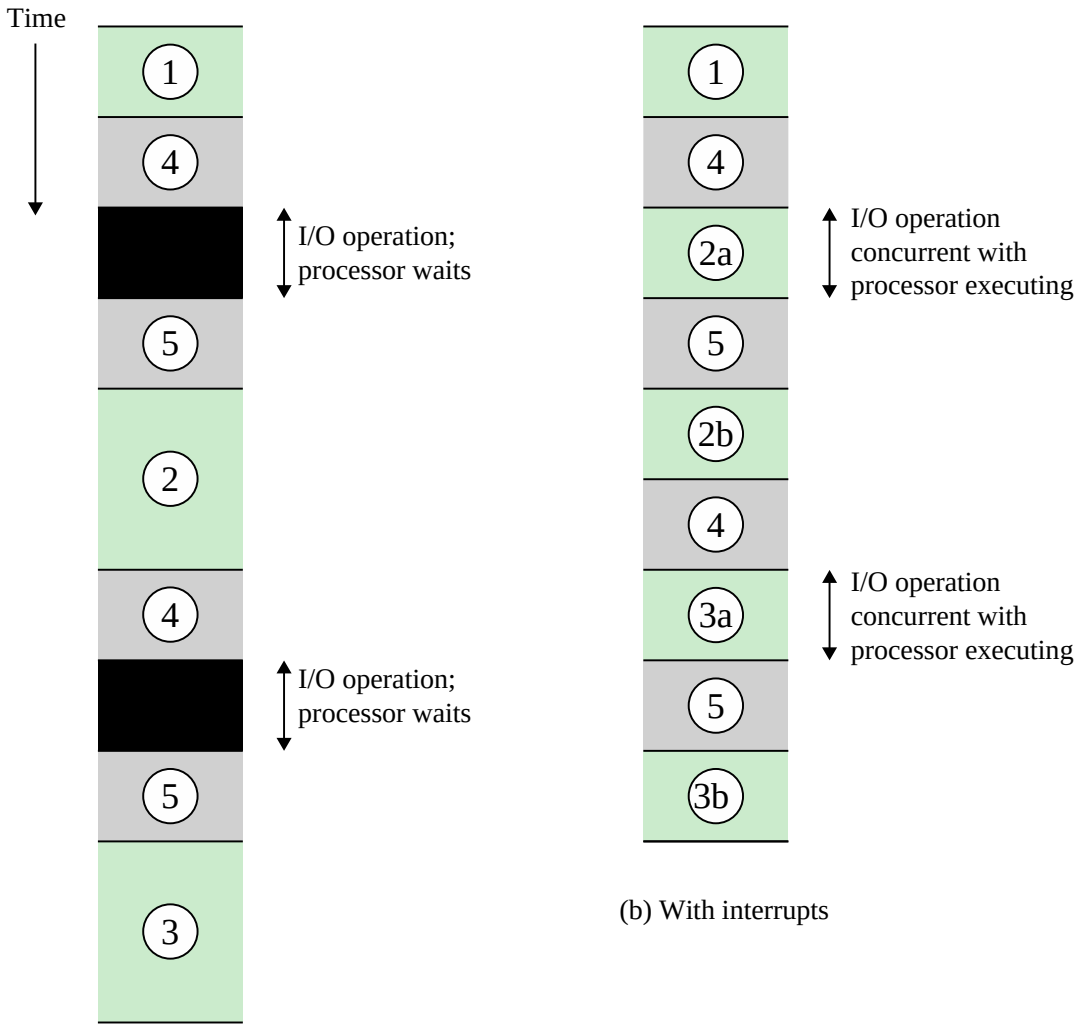


Figure 3.10 Program Timing: Short I/O Wait

Figure 3.11

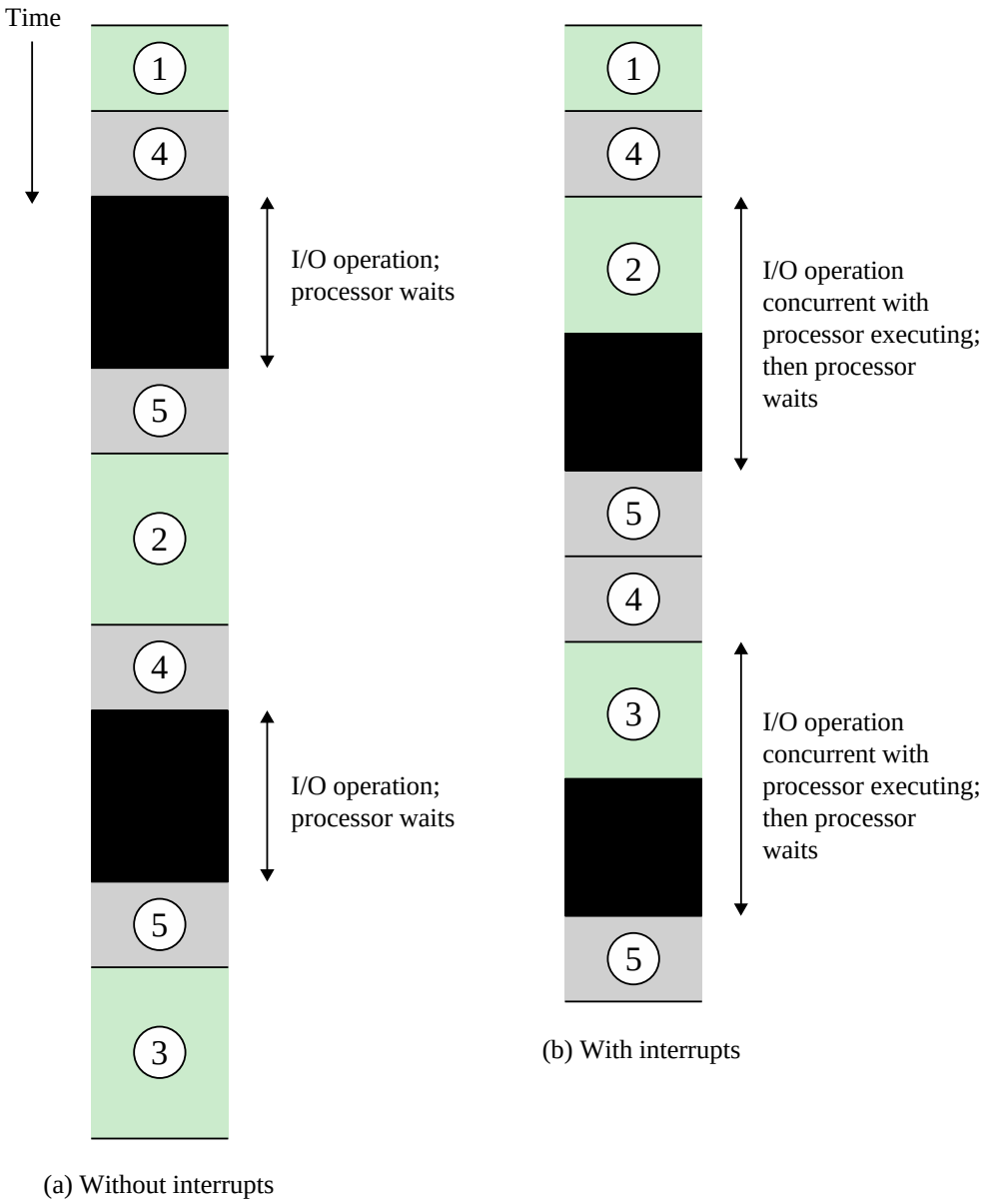


Figure 3.11 Program Timing: Long I/O Wait

Figure 3.12

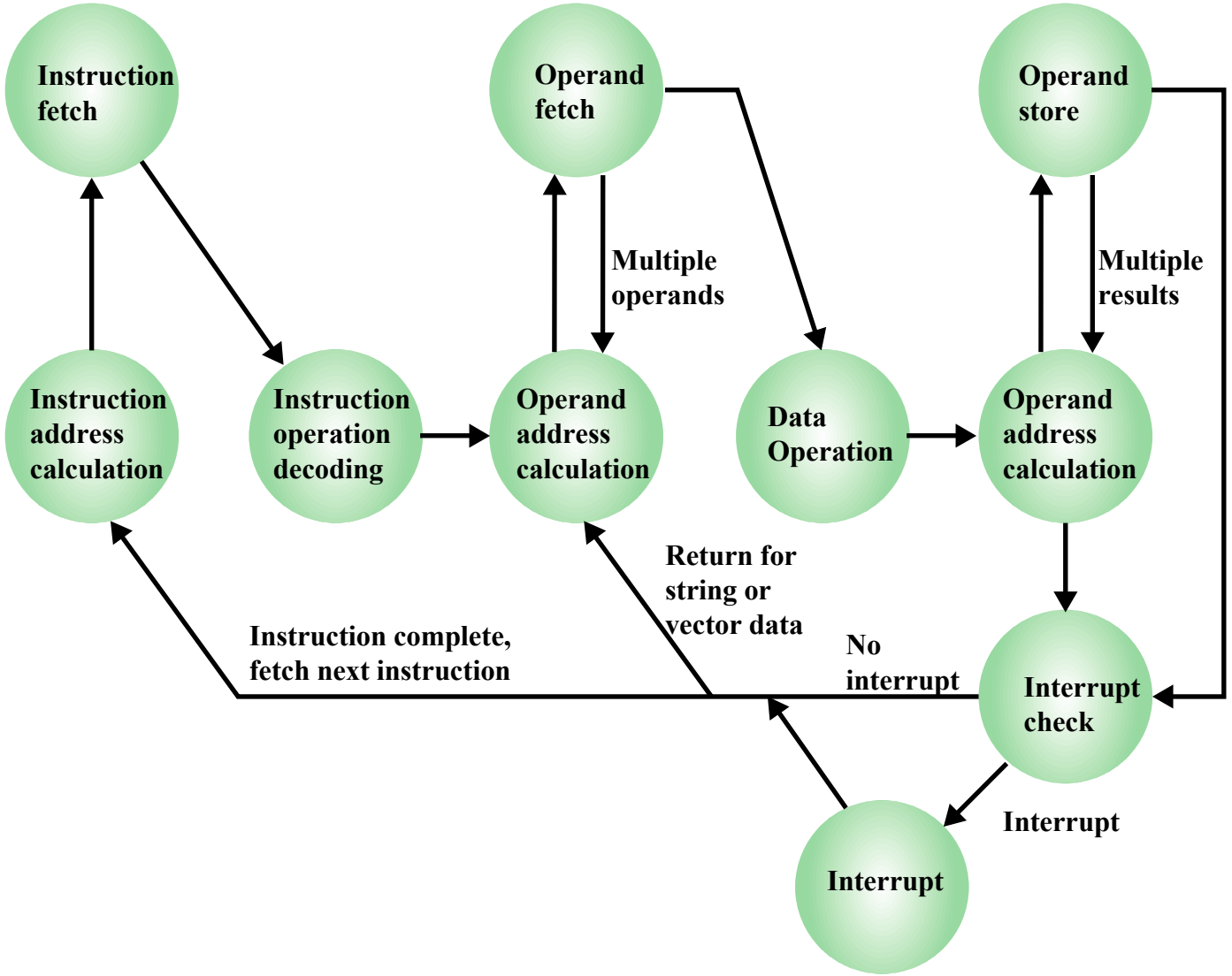


Figure 3.12 Instruction Cycle State Diagram, With Interrupts

Figure 3.13

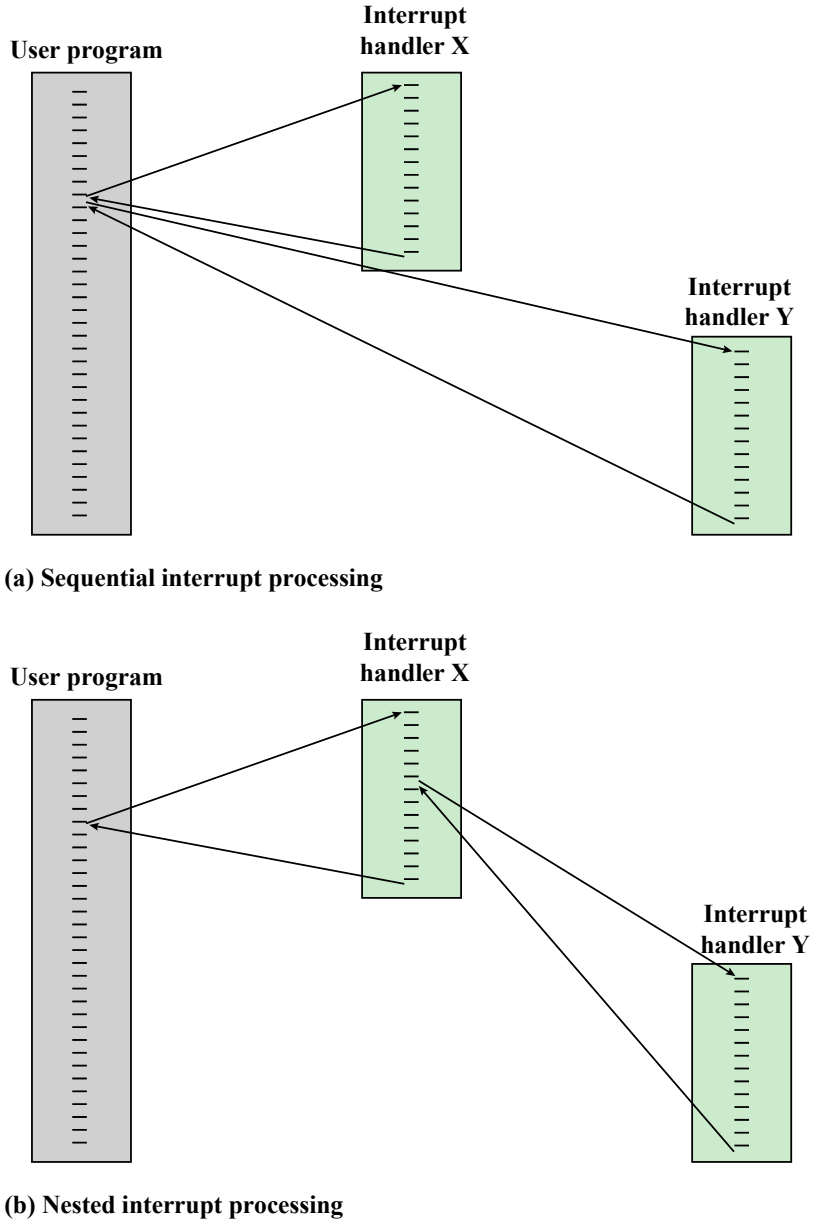


Figure 3.13 Transfer of Control with Multiple Interrupts

Figure 3.14

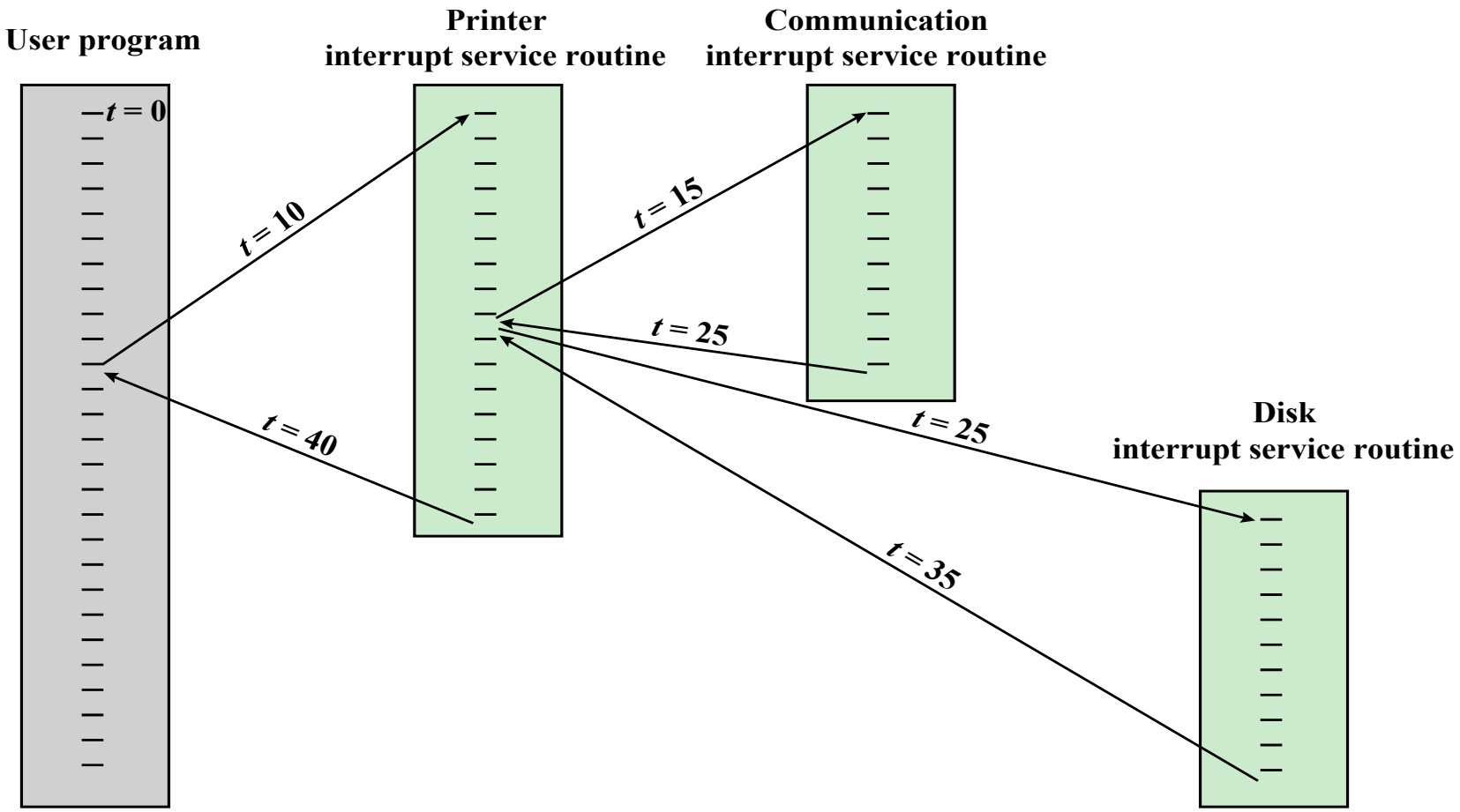


Figure 3.14 Example Time Sequence of Multiple Interrupts

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)

Copyright