

*Operating Systems:
Internals and Design Principles, 6/E*
William Stallings

Computer System Overview

Dr. Sanam Shahla Rizvi



Roadmap

➔ Basic Elements

- Processor Registers
- Instruction Execution
- Interrupts

Top-Level View

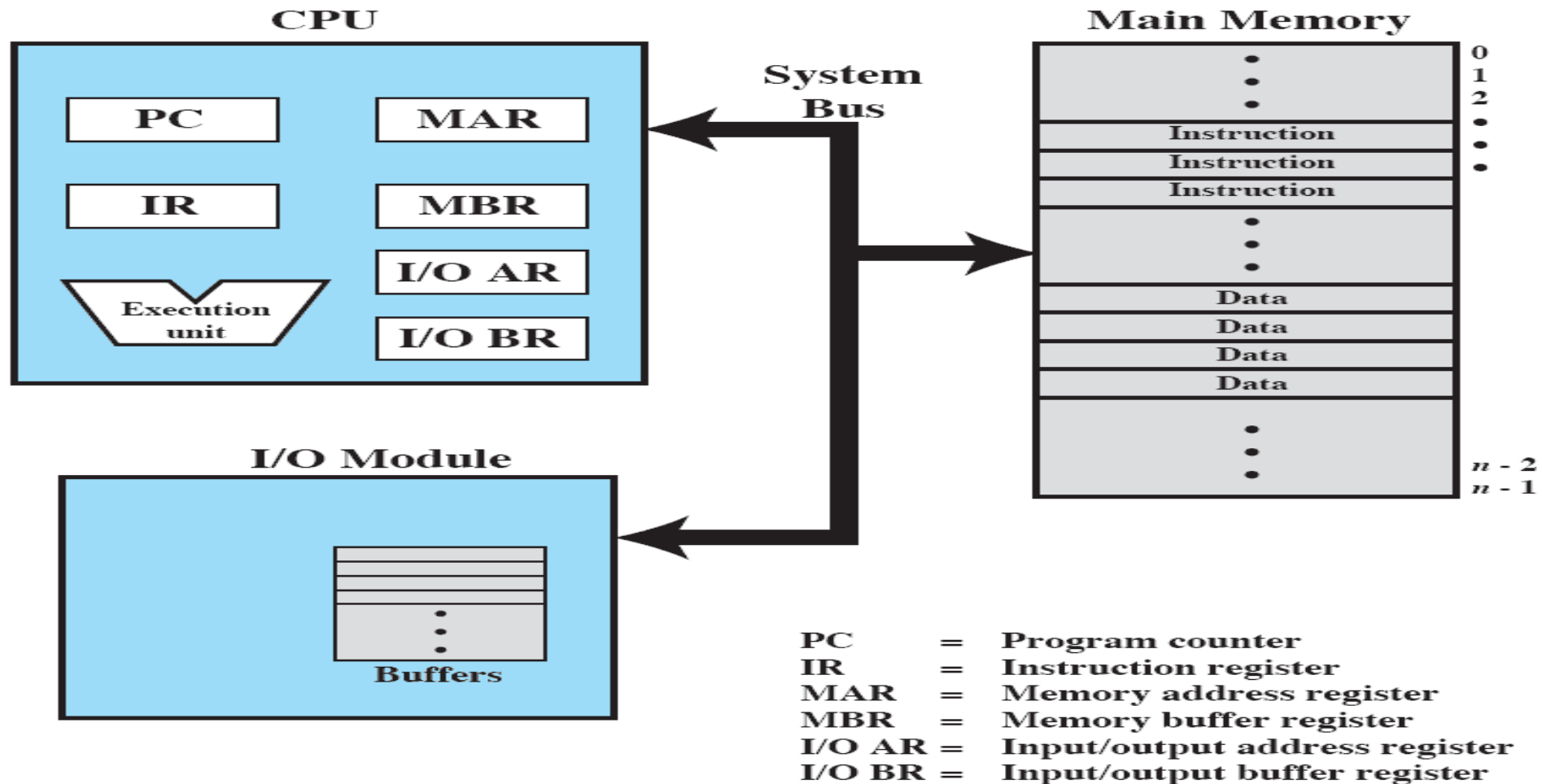


Figure 1.1 Computer Components: Top-Level View

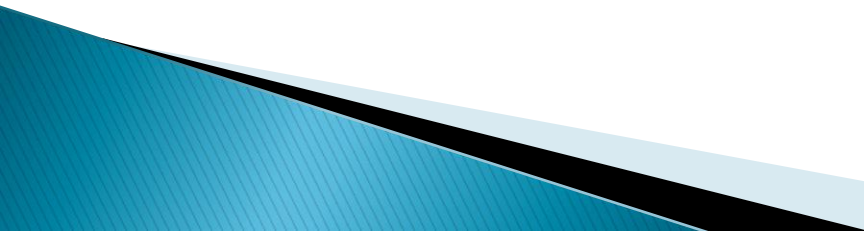
Roadmap

- Basic Elements

→ Processor Registers

- Instruction Execution
- Interrupts

Processor Registers

- ▶ Faster and smaller than main memory
 - ▶ User-visible registers
 - Enable programmer to minimize main memory references by optimizing register use
 - ▶ Control and status registers
 - Used by processor to control operating of the processor
 - Used by privileged OS routines to control the execution of programs
- 

User-Visible Registers

- ▶ May be referenced by machine language
 - Available to all programs – application programs and system programs
- ▶ Types of registers typically available are:
 - data,
 - address.

Control and Status Registers

- ▶ Program counter (PC)
 - Contains the address of an instruction to be fetched
- ▶ Instruction register (IR)
 - Contains the instruction most recently fetched
- ▶ Program status word (PSW)
 - Contains status information

Condition codes

- ▶ Usually part of the control register
 - Also called *flags*
- ▶ Bits set by processor hardware as a result of operations
 - Read only, intended for feedback regarding the results of instruction execution.

Roadmap

- Basic Elements
- Processor Registers
- ◦ Instruction Execution
- Interrupts

Instruction Execution

- ▶ A program consists of a set of instructions stored in memory
- ▶ Two steps
 - Processor reads (fetches) instructions from memory
 - Processor executes each instruction

Basic Instruction Cycle

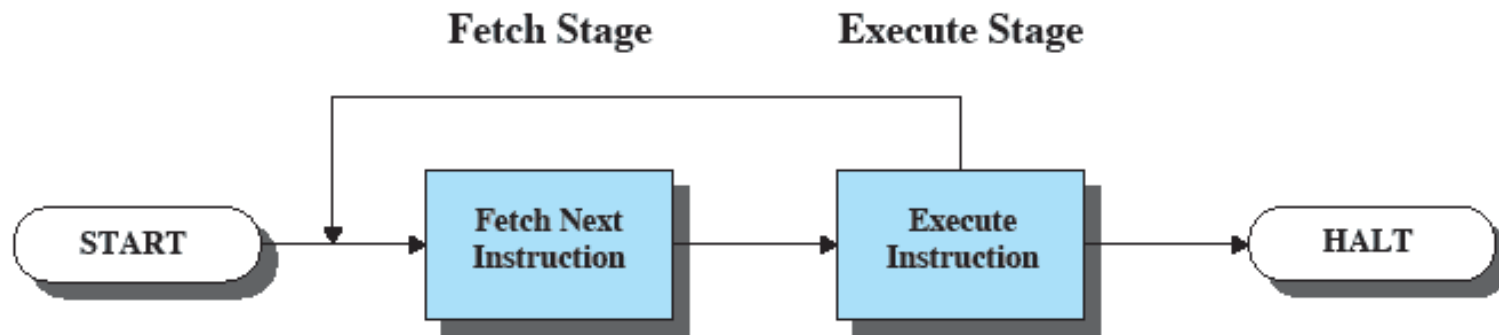
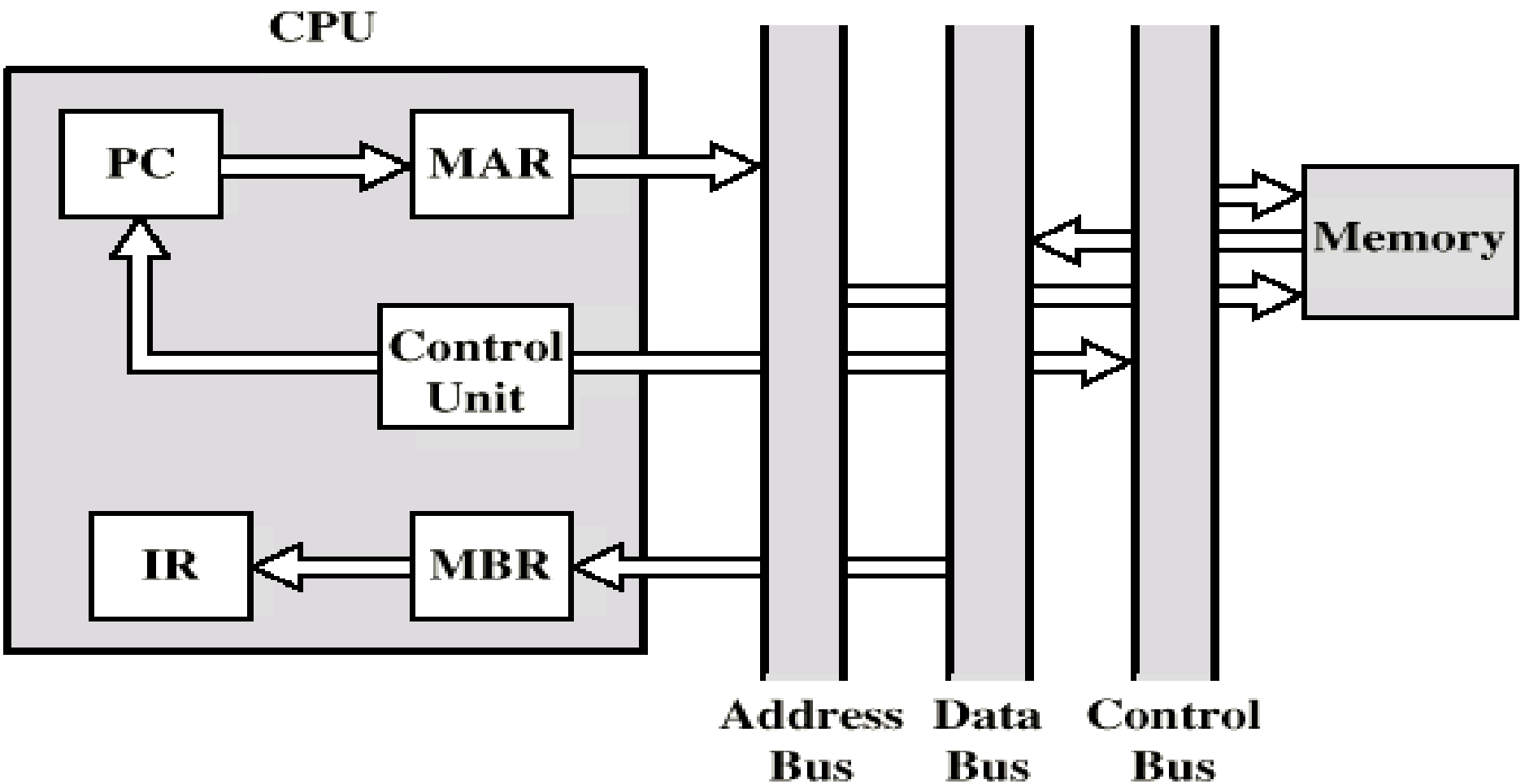


Figure 1.2 Basic Instruction Cycle

Instruction Fetch and Execute

- ▶ The processor fetches the instruction from memory
- ▶ Program counter (PC) holds address of the instruction to be fetched next
 - PC is incremented after each fetch



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

Figure 11.7 Data Flow, Fetch Cycle

Instruction Register

- ▶ Fetched instruction loaded into instruction register
- ▶ Processor's action 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.)

Characteristics of a Hypothetical Machine



(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 1.3 Characteristics of a Hypothetical Machine

Example of Program Execution

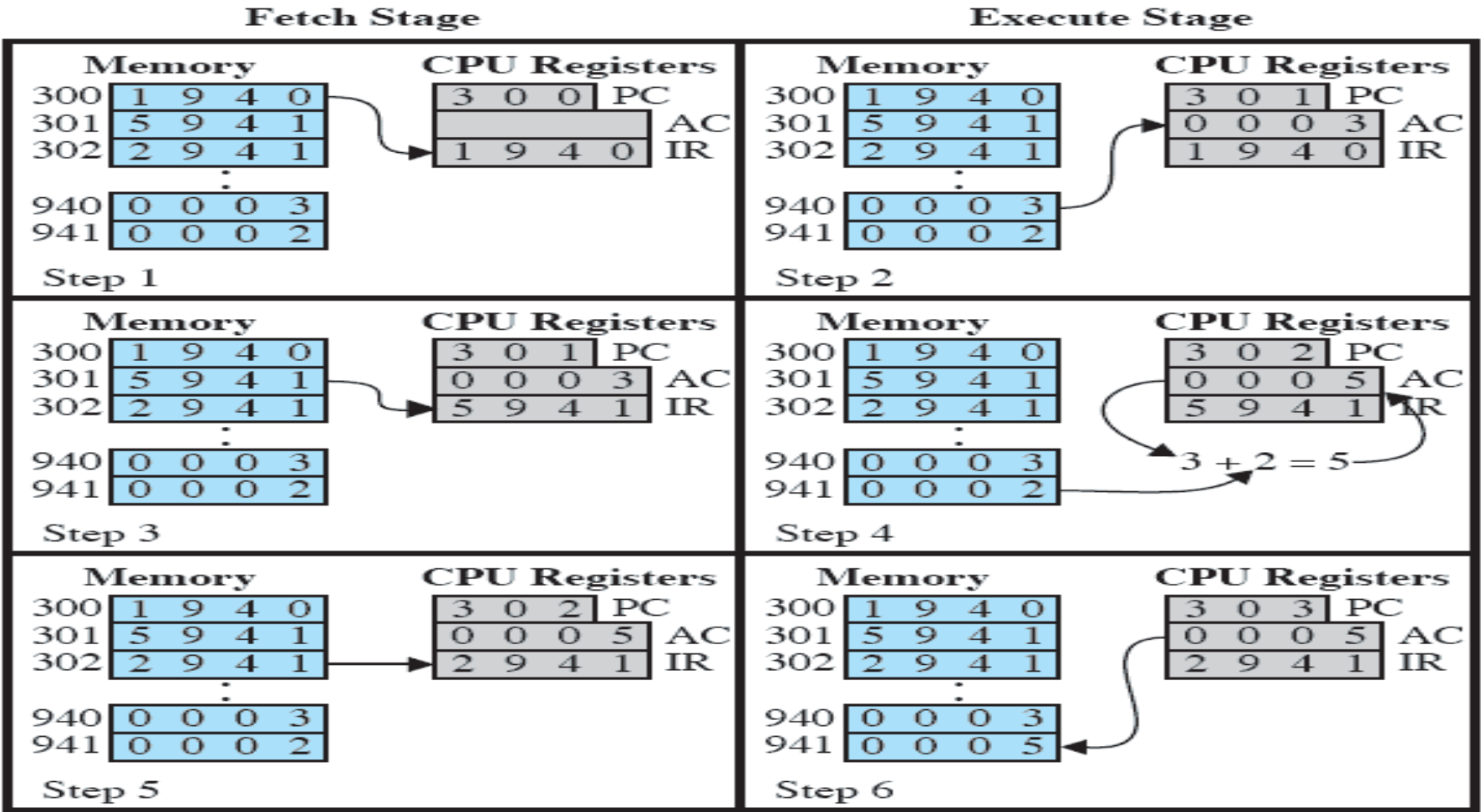


Figure 1.4 Example of Program Execution (contents of memory and registers in hexadecimal)

Roadmap

- Basic Elements
- Processor Registers
- Instruction Execution
- ➔ ◦ Interrupts

Interrupts

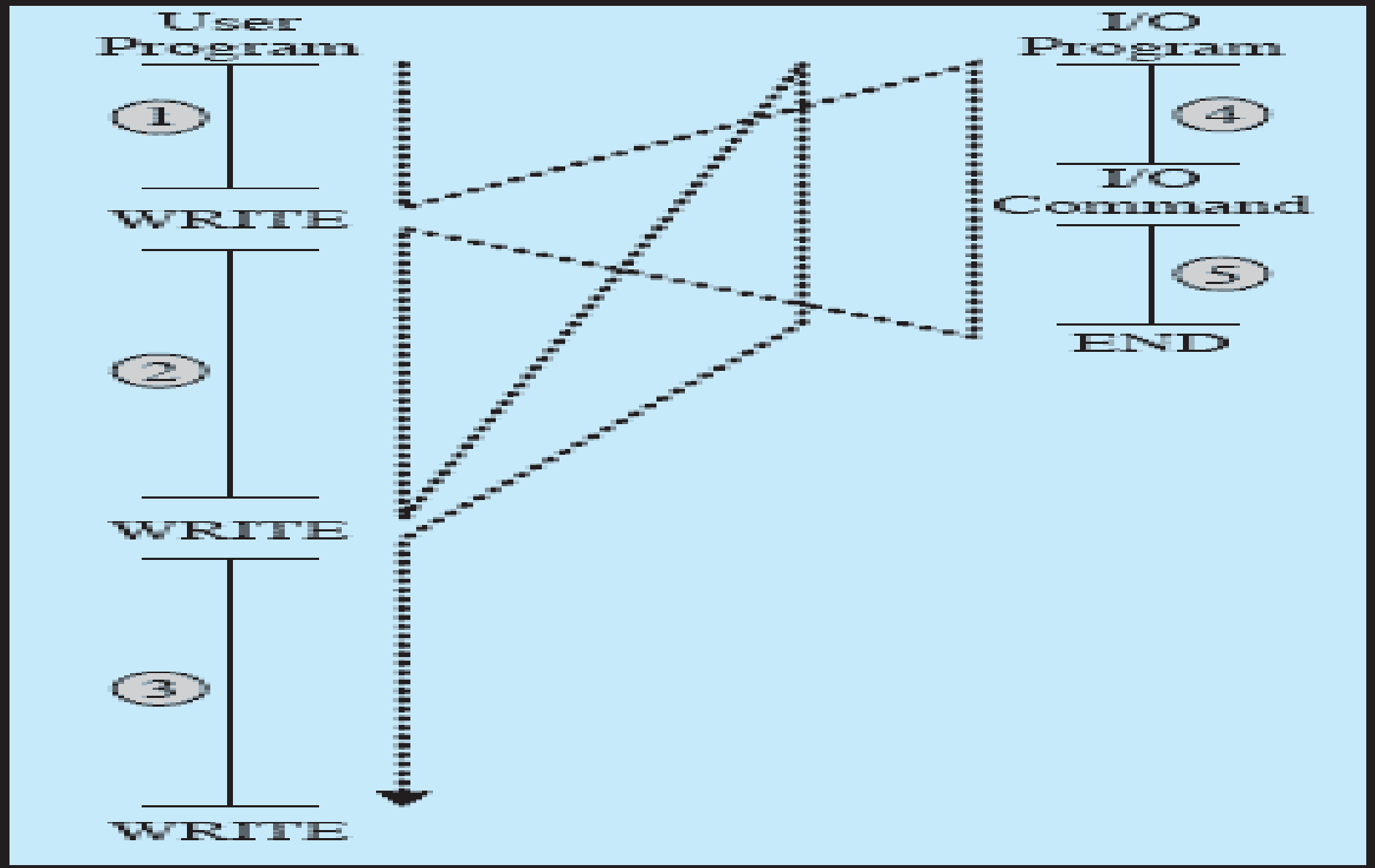
- ▶ Interrupt the normal sequencing of the processor
- ▶ Provided to improve processor utilization
 - Most I/O devices are slower than the processor
 - Processor must pause to wait for device

Common Classes of Interrupts

Table 1.1 Classes of Interrupts

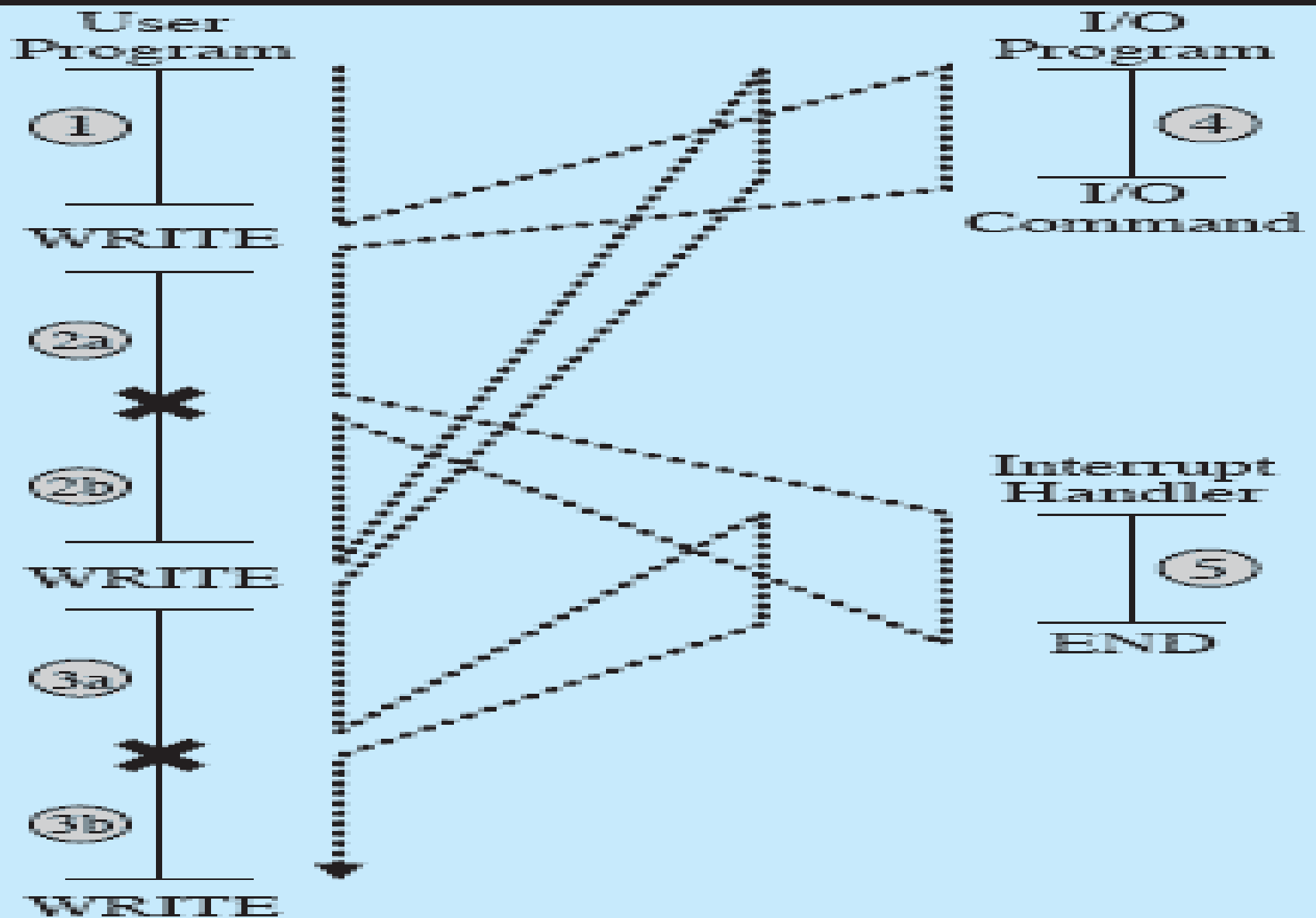
Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows 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.

Flow of Control without Interrupts



(a) No interrupts

Interrupts and the Instruction Cycle



(b) Interrupts; short I/O wait

Transfer of Control via Interrupts

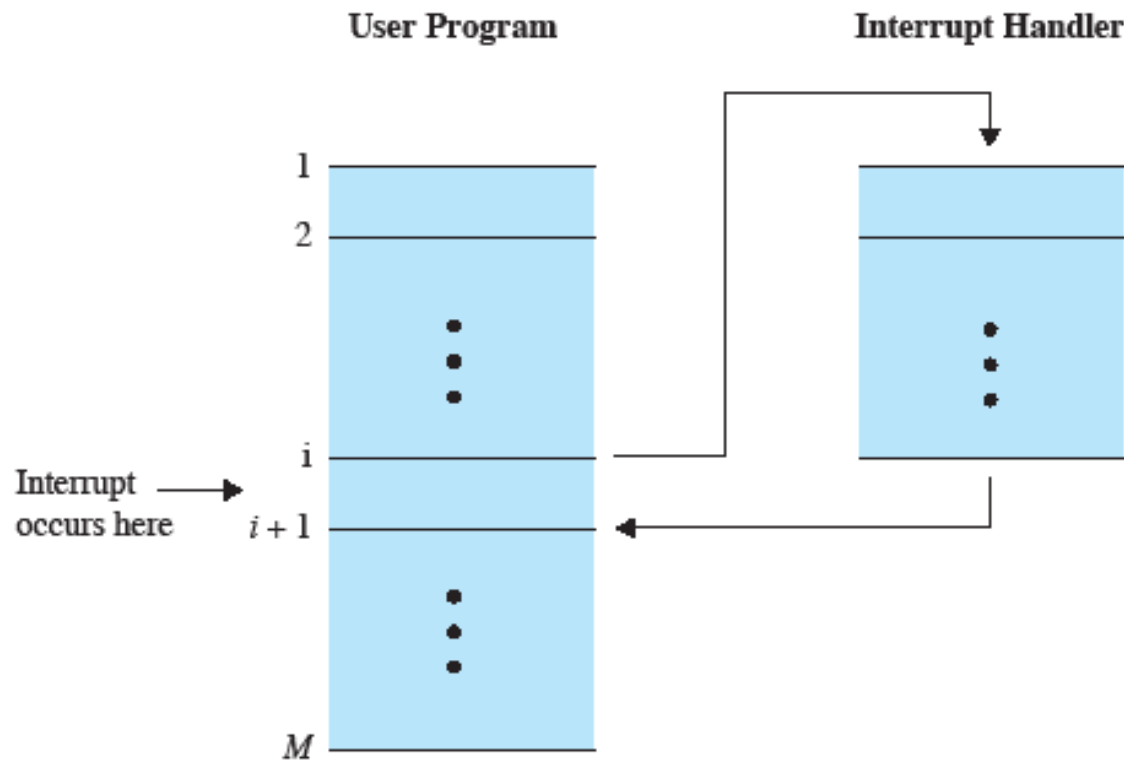


Figure 1.6 Transfer of Control via Interrupts

Instruction Cycle with Interrupts

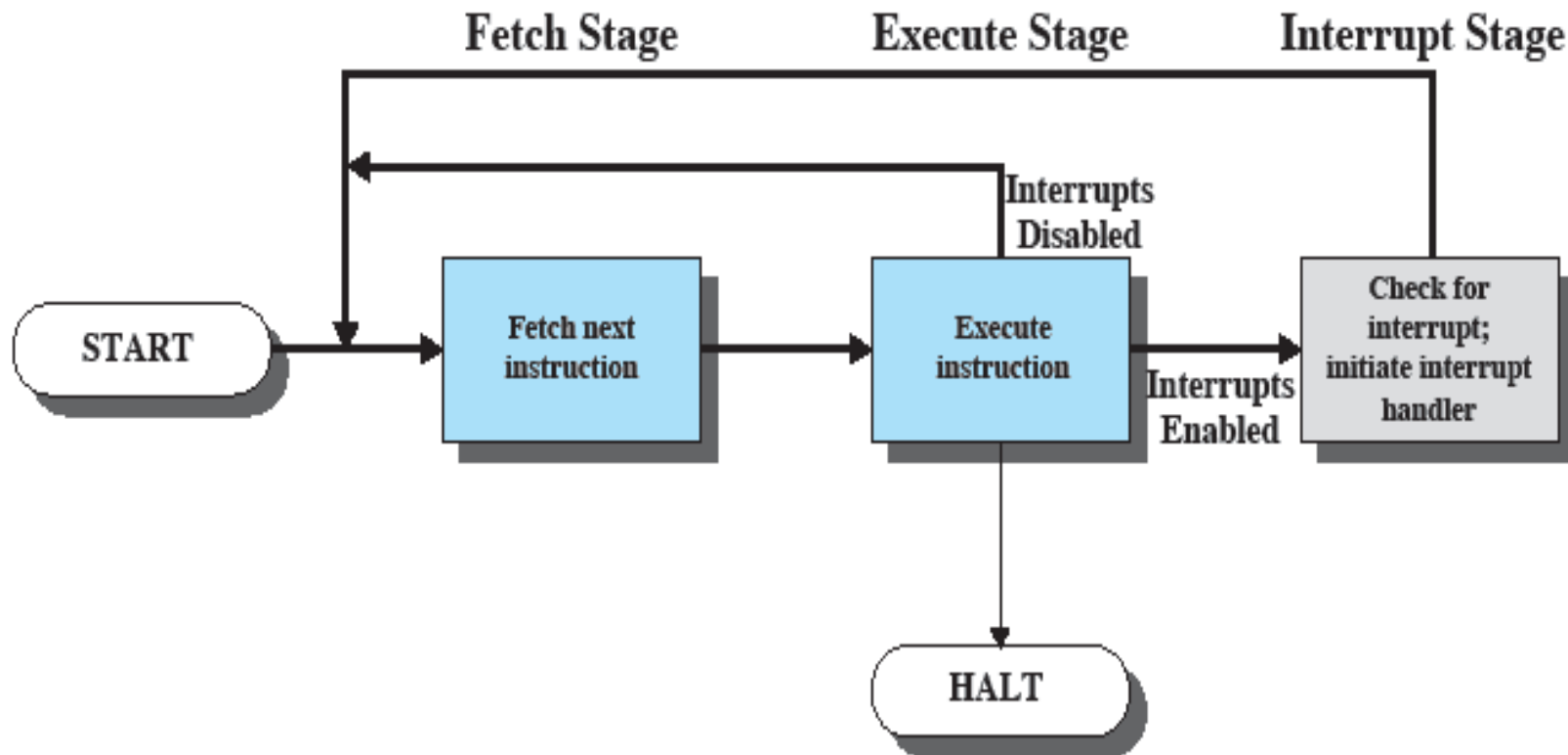
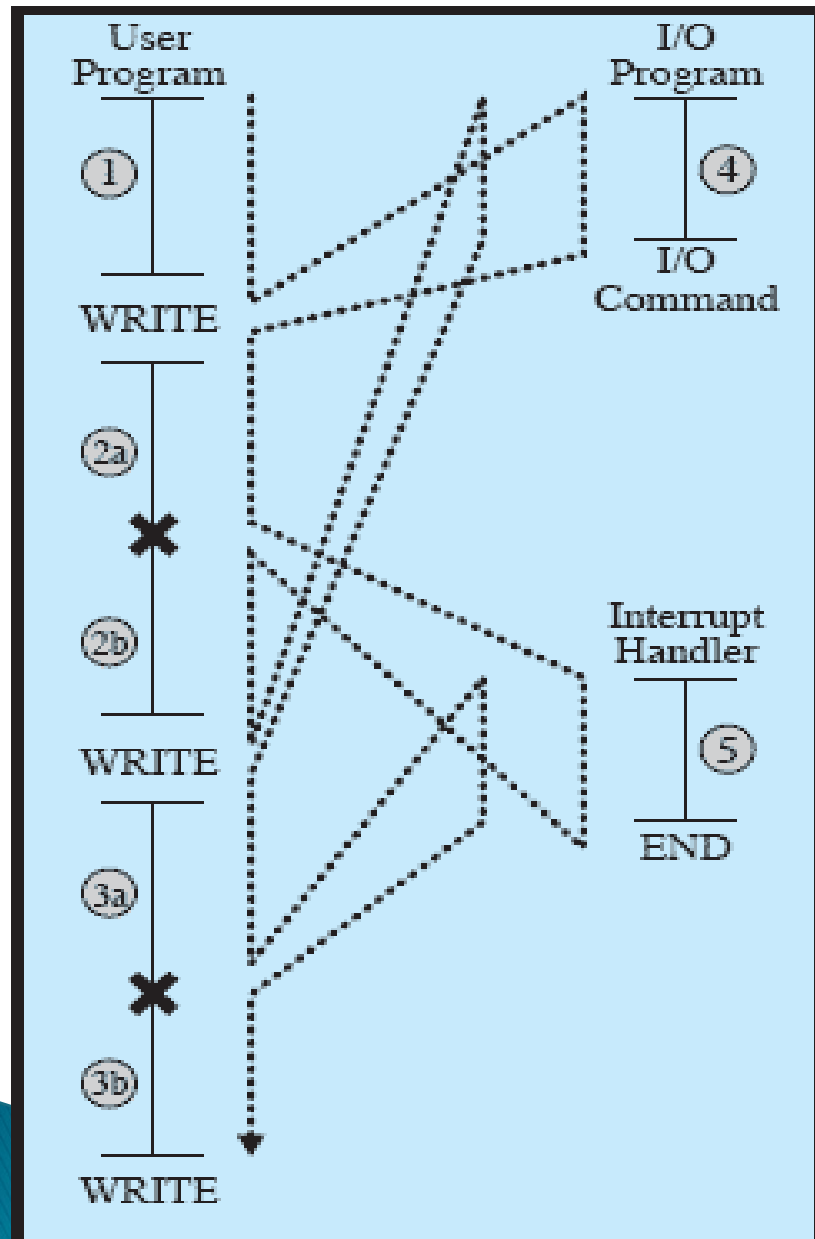


Figure 1.7 Instruction Cycle with Interrupts

Short I/O Wait



(b) Interrupts; short I/O wait

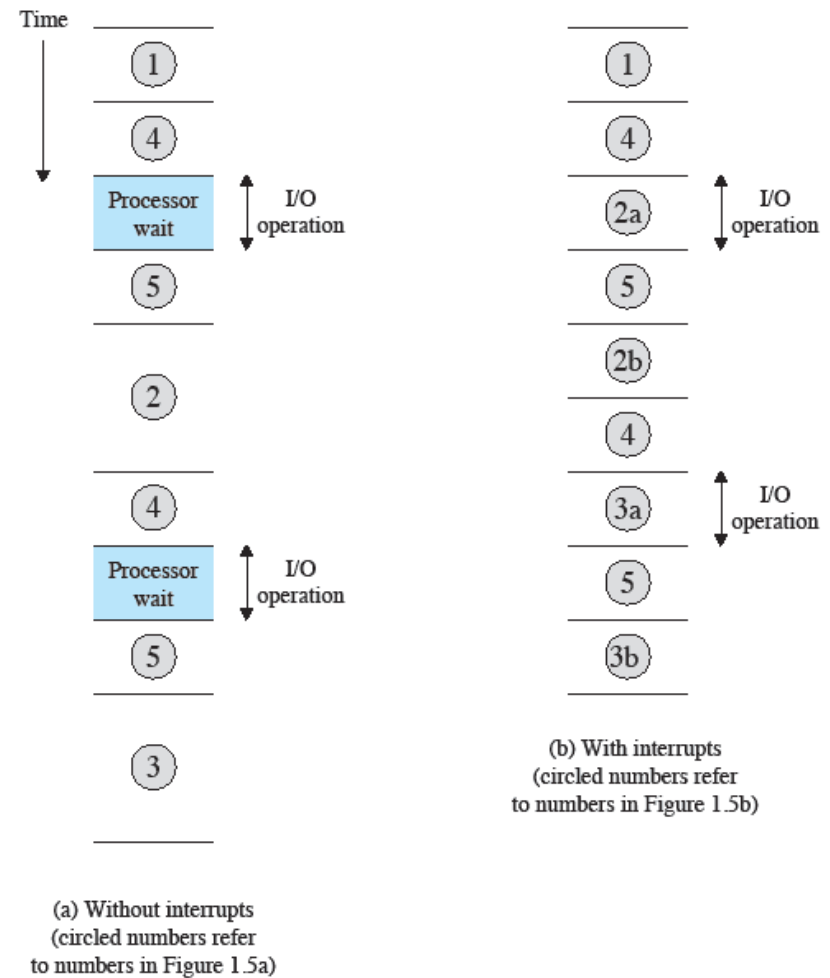
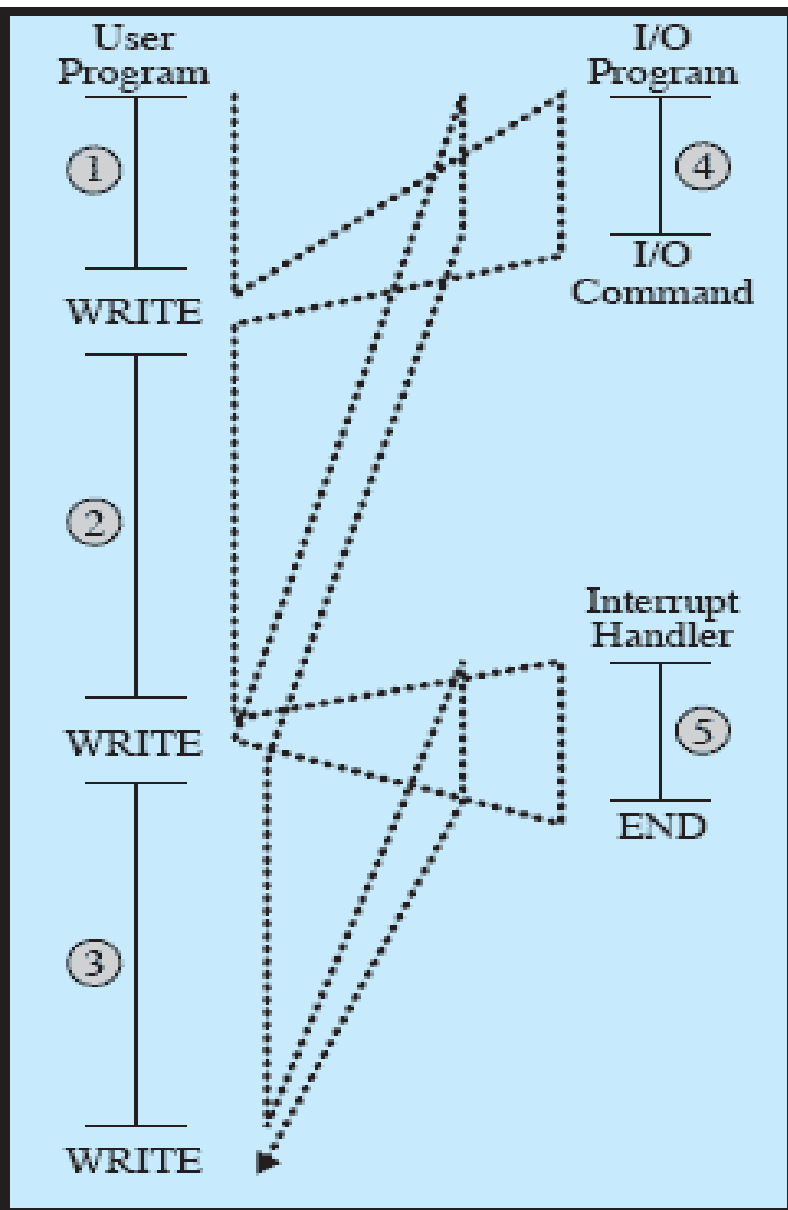
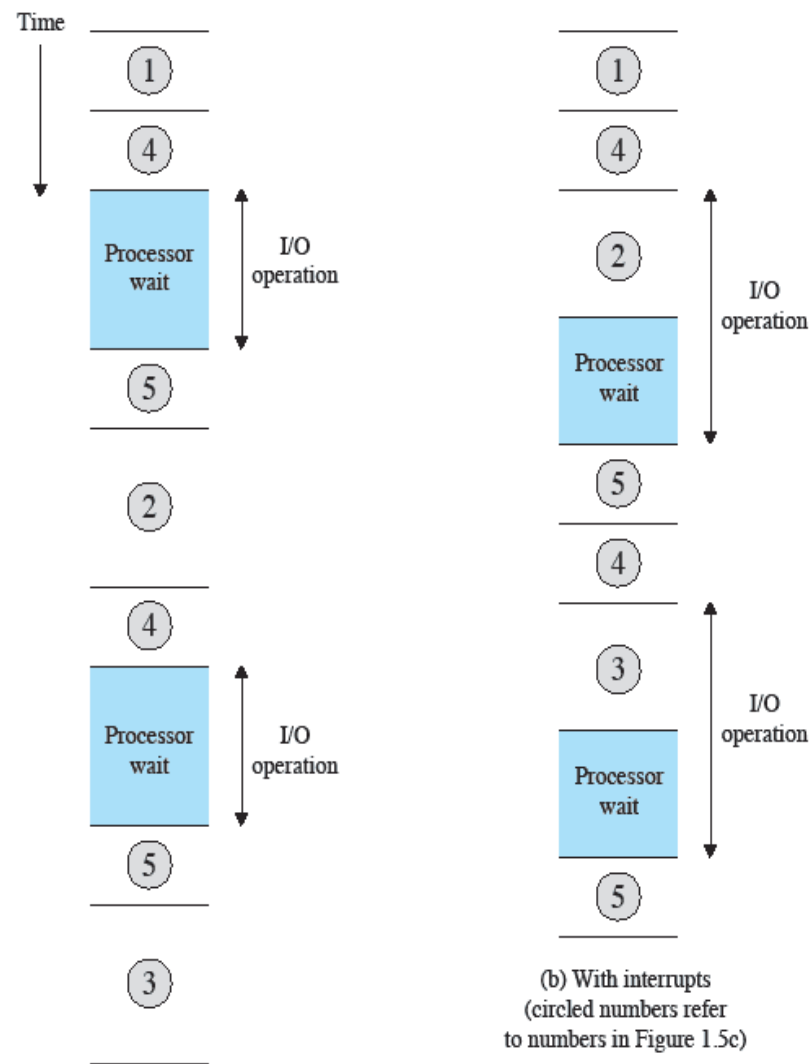


Figure 1.8 Program Timing: Short I/O Wait

Long I/O wait



(c) Interrupts; long I/O wait



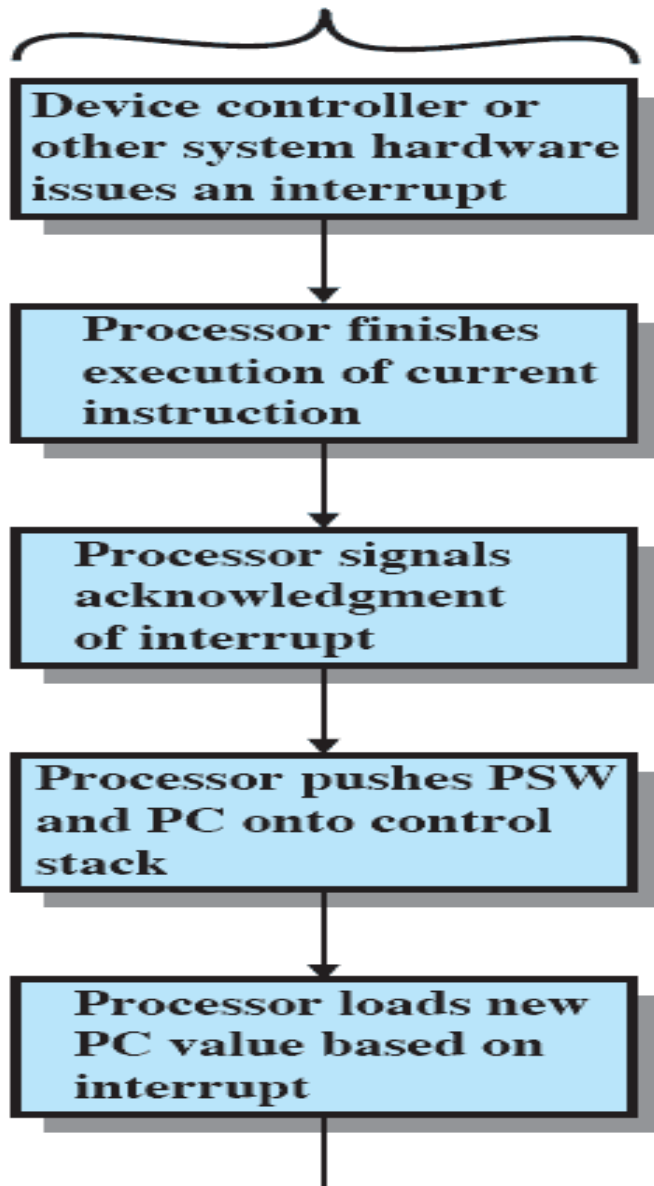
(a) Without interrupts
(circled numbers refer to numbers in Figure 1.5a)

(b) With interrupts
(circled numbers refer to numbers in Figure 1.5c)

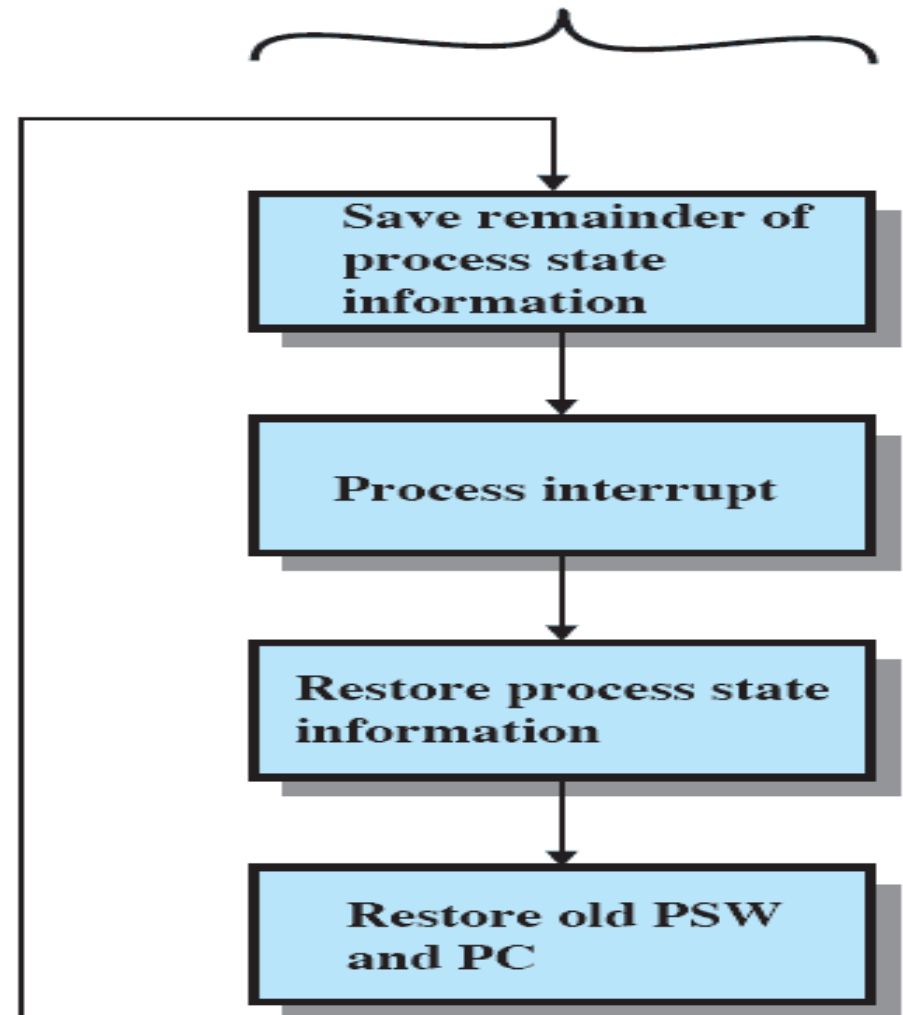
Figure 1.9 Program Timing: Long I/O Wait

Simple Interrupt Processing

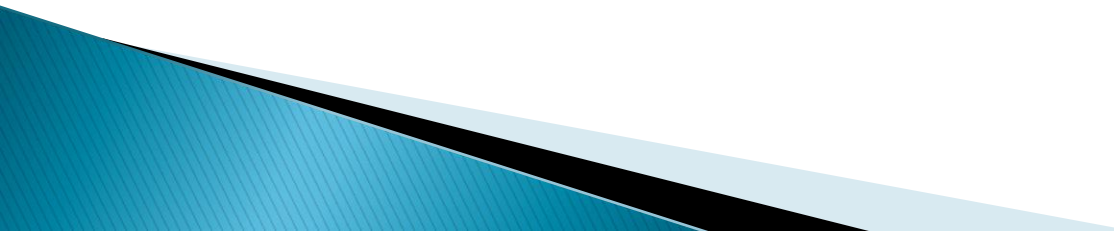
Hardware



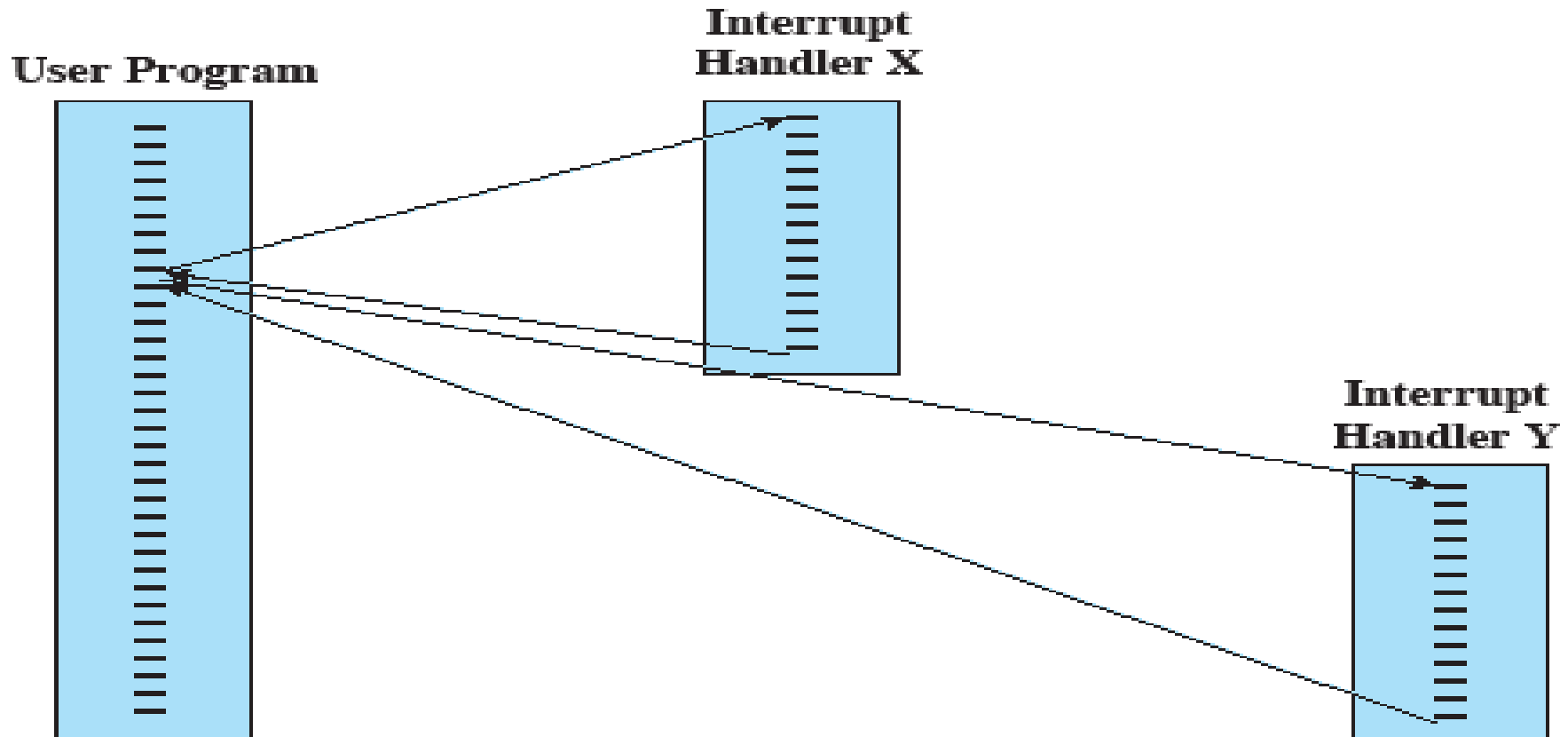
Software



Multiple Interrupts

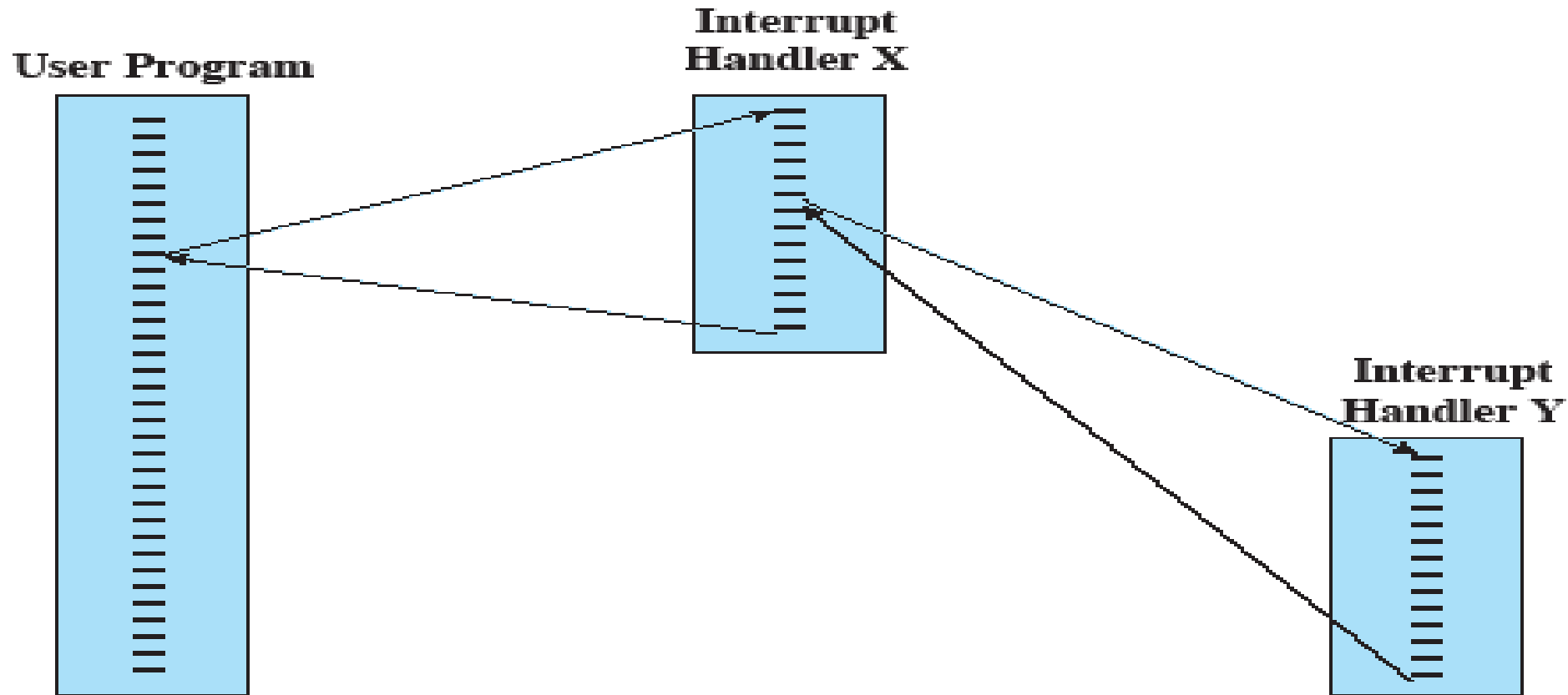
- ▶ Suppose an interrupt occurs while another interrupt is being processed.
 - ▶ Two approaches:
 - Disable interrupts during interrupt processing
 - Sequential interrupt processing
 - Use a priority scheme
 - Nested interrupt processing
- 

Sequential Interrupt Processing



(a) Sequential interrupt processing

Nested Interrupt Processing



(b) Nested interrupt processing

Example of Nested Interrupts

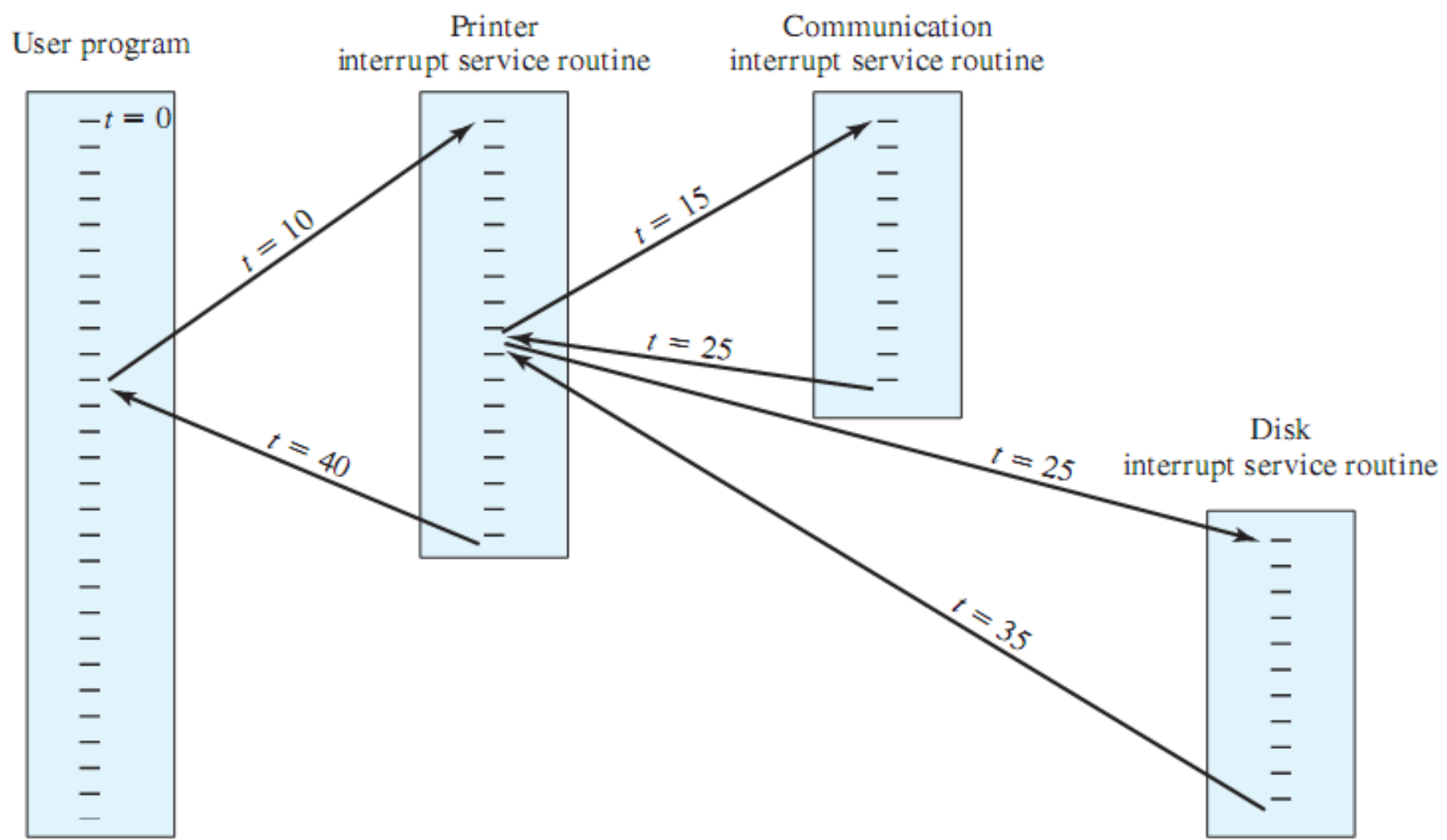


Figure 1.13 Example Time Sequence of Multiple Interrupts