Chapter 1:
Data Storage

Introduction to Computer Science

by
Kai-Lung Hua

PEARSON
Addison Wesley

Chapter 1: Data Storage

- 1.1 Bits and Their Storage
- 1.2 Main Memory
- 1.3 Mass Storage
- 1.4 Representing Information as Bit Patterns
- 1.5 The Binary System

Chapter 1: Data Storage (continued)

- 1.6 Storing Integers
- 1.7 Storing Fractions
- 1.8 Data Compression
- 1.9 Communications Errors

Bits and Bit Patterns

- **Bit:** Binary Digit (0 or 1)
- Bit Patterns are used to represent information.
  - Numbers
  - Text characters
  - Images
  - Sound
  - And others

Boolean Operations

- **Boolean Operation:** An operation that manipulates one or more true/false values
- Specific operations
  - AND
  - OR
  - XOR (exclusive or)
  - NOT

Figure 1.1 **The Boolean operations AND, OR, and XOR (exclusive or)**

The AND operation

$$\begin{array}{lr} & 0 \\ \text{AND} & 0 \\ \hline & 0 \end{array} \qquad \begin{array}{lr} & 0 \\ \text{AND} & 1 \\ \hline & 0 \end{array} \qquad \begin{array}{lr} & 1 \\ \text{AND} & 0 \\ \hline & 0 \end{array} \qquad \begin{array}{lr} & 1 \\ \text{AND} & 1 \\ \hline & 1 \end{array}$$

The OR operation

$$\begin{array}{lr} & 0 \\ \text{OR} & 0 \\ \hline & 0 \end{array} \qquad \begin{array}{lr} & 0 \\ \text{OR} & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{lr} & 1 \\ \text{OR} & 0 \\ \hline & 1 \end{array} \qquad \begin{array}{lr} & 1 \\ \text{OR} & 1 \\ \hline & 1 \end{array}$$

The XOR operation

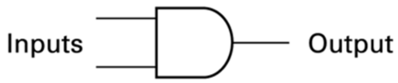
$$\begin{array}{lr} & 0 \\ \text{XOR} & 0 \\ \hline & 0 \end{array} \qquad \begin{array}{lr} & 0 \\ \text{XOR} & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{lr} & 1 \\ \text{XOR} & 0 \\ \hline & 1 \end{array} \qquad \begin{array}{lr} & 1 \\ \text{XOR} & 1 \\ \hline & 0 \end{array}$$

Gates

- **Gate:** A device that computes a Boolean operation
  - Often implemented as (small) electronic circuits
  - Provide the building blocks from which computers are constructed
  - VLSI (Very Large Scale Integration)

Figure 1.2 **A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values**

AND



Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

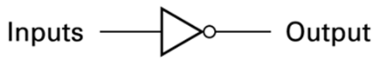
OR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

NOT



Inputs	Output
0	1
1	0

Flip-flops

- **Flip-flop:** A circuit built from gates that can store one bit.
  - One input line is used to set its stored value to 1
  - One input line is used to set its stored value to 0
  - While both input lines are 0, the most recently stored value is preserved

Figure 1.3 **A simple flip-flop circuit**

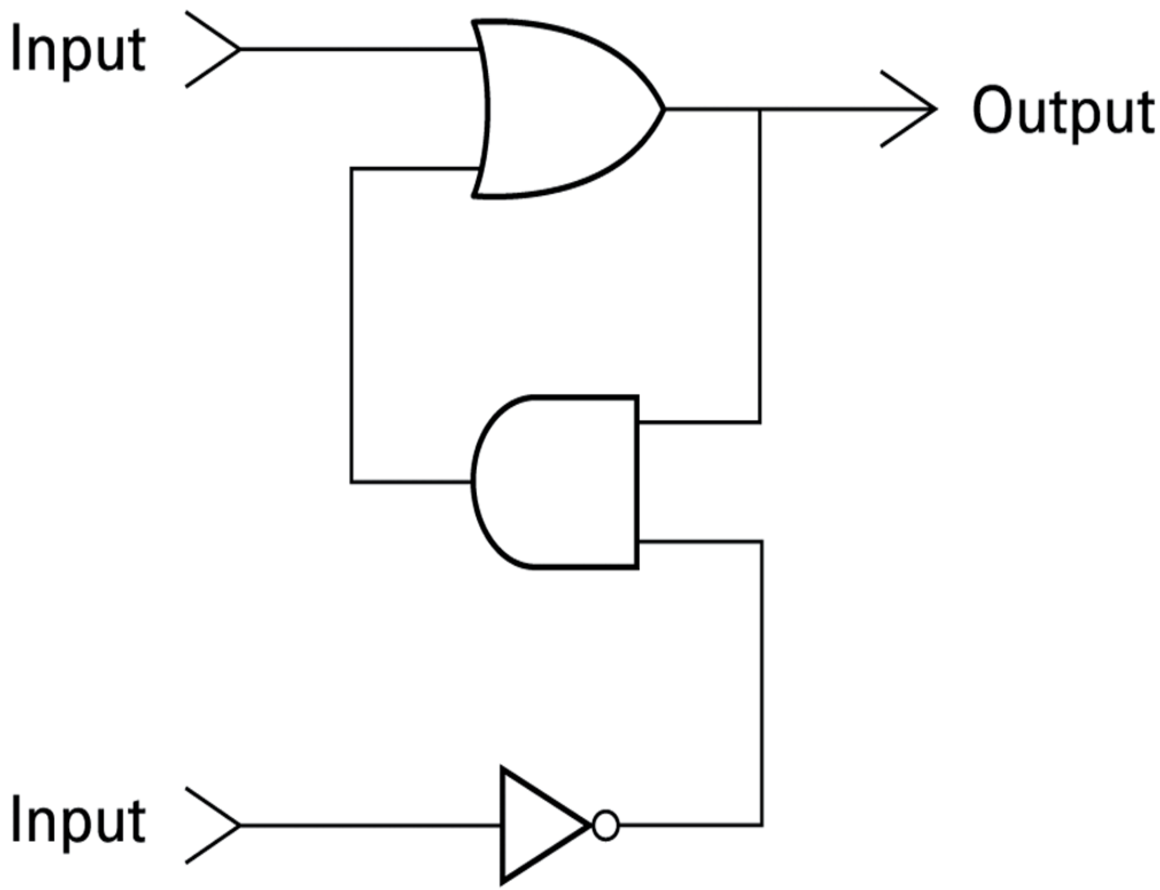


Figure 1.4 **Setting the output of a flip-flop to 1**

a. 1 is placed on the upper input.

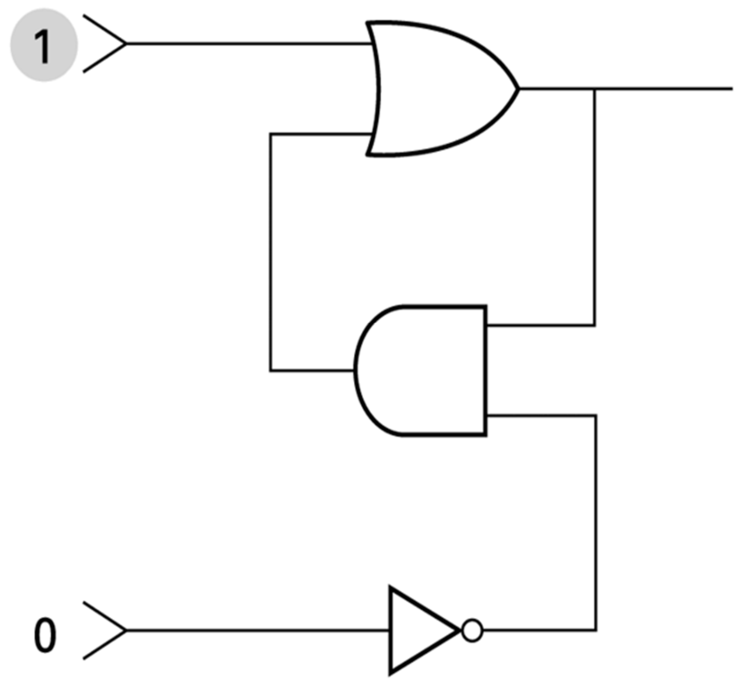


Figure 1.4 **Setting the output of a flip-flop to 1 (continued)**

b. This causes the output of the OR gate to be 1 and, in turn, the output of the AND gate to be 1.

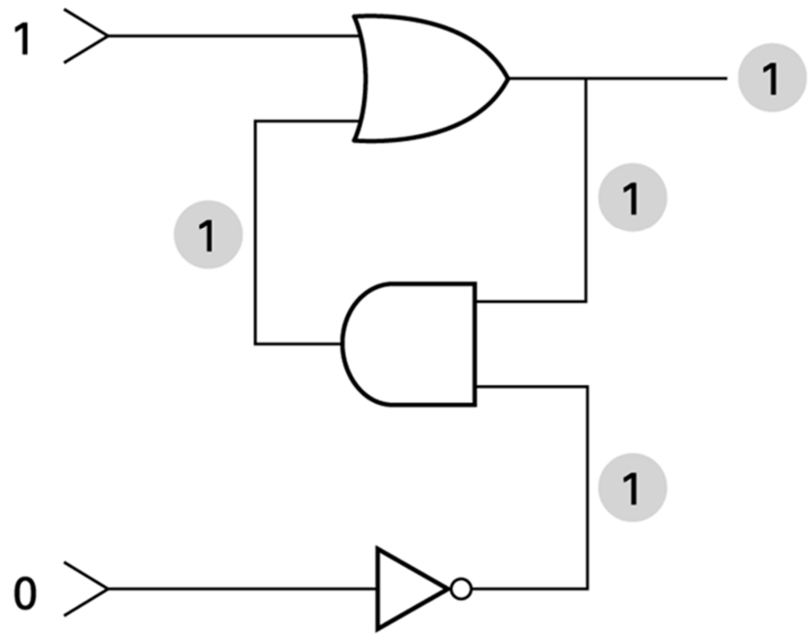


Figure 1.4 **Setting the output of a flip-flop to 1 (continued)**

c. The 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.

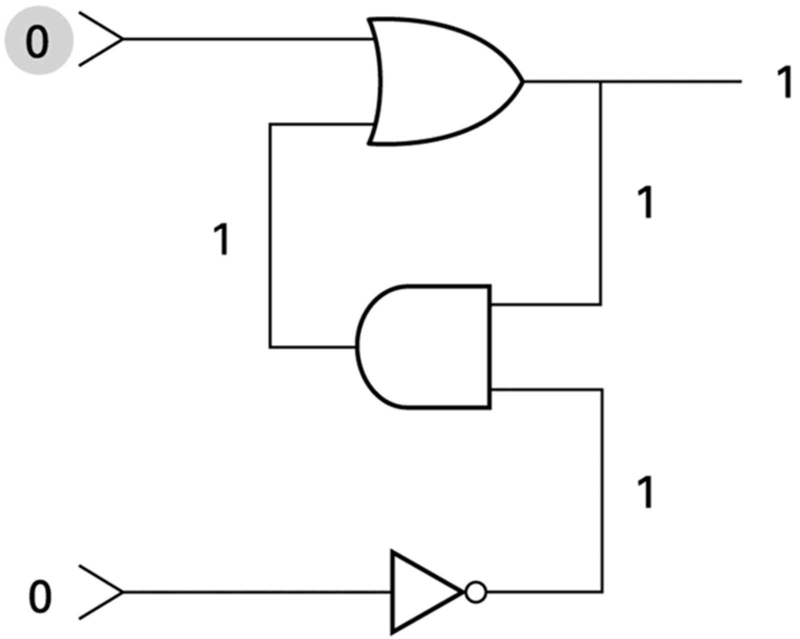
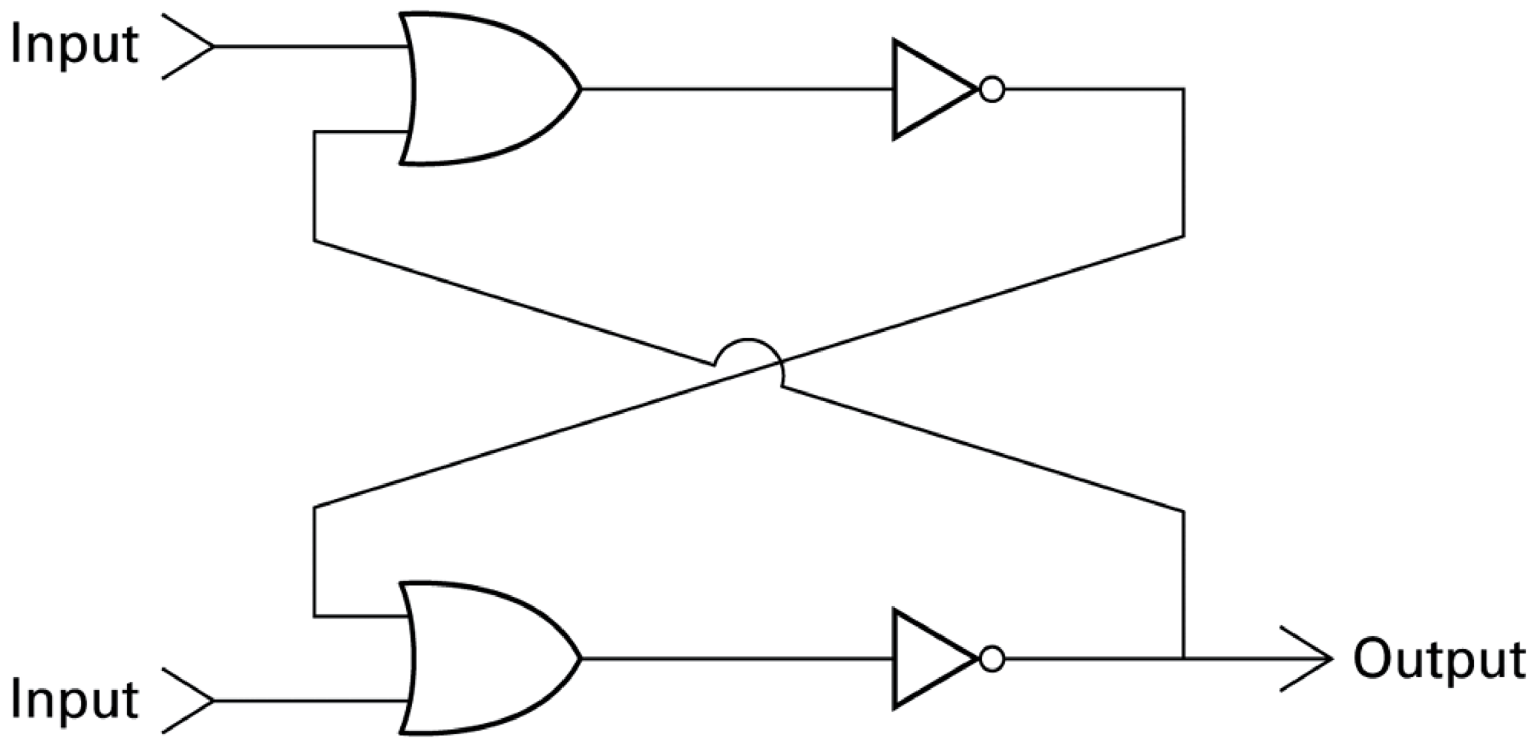


Figure 1.5 **Another way of constructing a flip-flop**



Hexadecimal Notation

- **Hexadecimal notation:** A shorthand notation for long bit patterns
  - Divides a pattern into groups of four bits each
  - Represents each group by a single symbol
- Example: 10100011 becomes A3

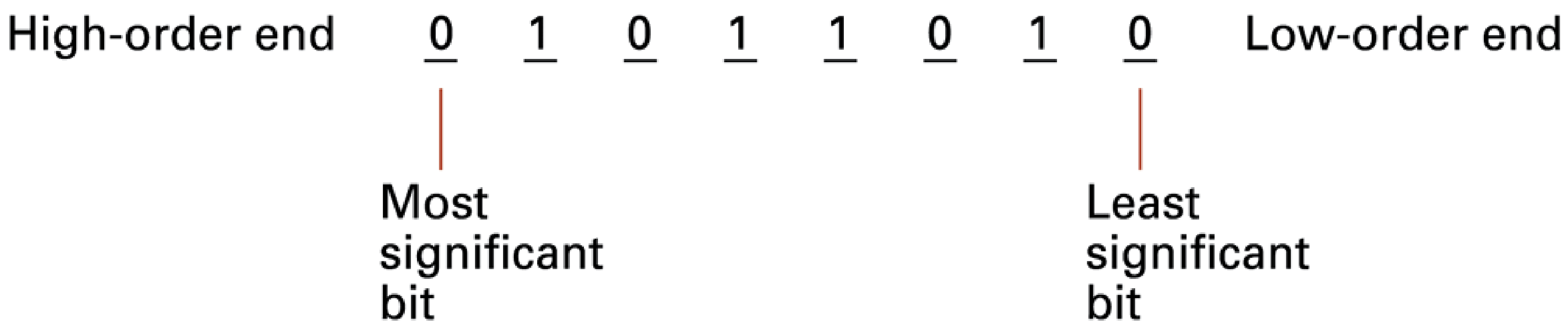
Figure 1.6 The hexadecimal coding system

Bit pattern	Hexadecimal representation
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Main Memory Cells

- **Cell:** A unit of main memory (typically 8 bits which is one **byte**)
  - **Most significant bit:** the bit at the left (high-order) end of the conceptual row of bits in a memory cell
  - **Least significant bit:** the bit at the right (low-order) end of the conceptual row of bits in a memory cell

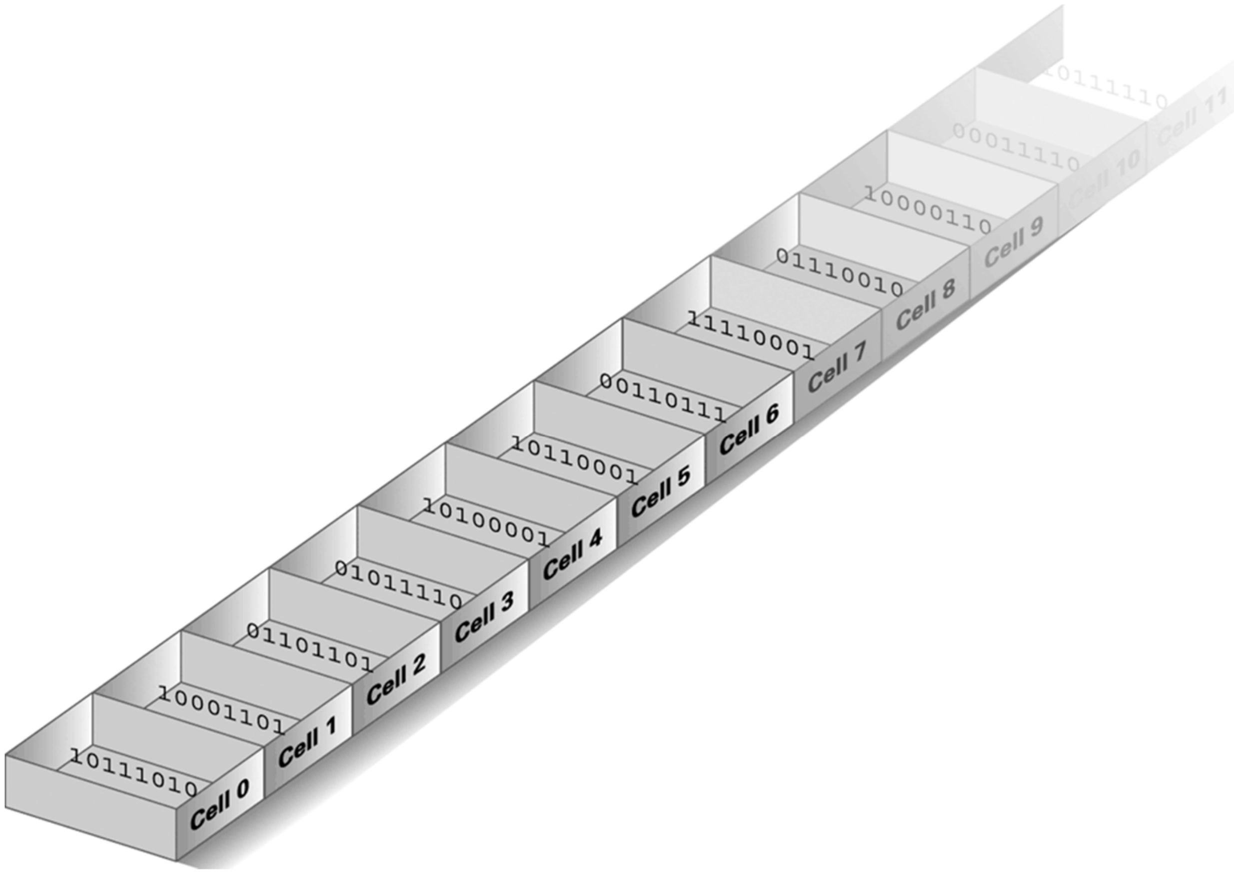
Figure 1.7 **The organization of a byte-size memory cell**



Main Memory Addresses

- **Address:** A “name” that uniquely identifies one cell in the computer’s main memory
  - The names are actually numbers.
  - These numbers are assigned consecutively starting at zero.
  - Numbering the cells in this manner associates an order with the memory cells.

Figure 1.8 **Memory cells arranged by address**



Memory Terminology

- **Random Access Memory (RAM):** Memory in which individual cells can be easily accessed in any order
- **Dynamic Memory (DRAM):** RAM composed of volatile memory

Measuring Memory Capacity

- **Kilobyte:** 2^{10} bytes = 1024 bytes
  - Example: 3 KB = 3 times1024 bytes
  - Sometimes “kibi” rather than “kilo”
- **Megabyte:** 2^{20} bytes = 1,048,576 bytes
  - Example: 3 MB = 3 times 1,048,576 bytes
  - Sometimes “megi” rather than “mega”
- **Gigabyte:** 2^{30} bytes = 1,073,741,824 bytes
  - Example: 3 GB = 3 times 1,073,741,824 bytes
  - Sometimes “gigi” rather than “giga”

Mass Storage

- On-line versus off-line
- Typically larger than main memory
- Typically less volatile than main memory
- Typically slower than main memory

Mass Storage Systems

- Magnetic Systems
  - Disk
  - Tape
- Optical Systems
  - CD
  - DVD
- Flash Drives

Figure 1.9 **A magnetic disk storage system**

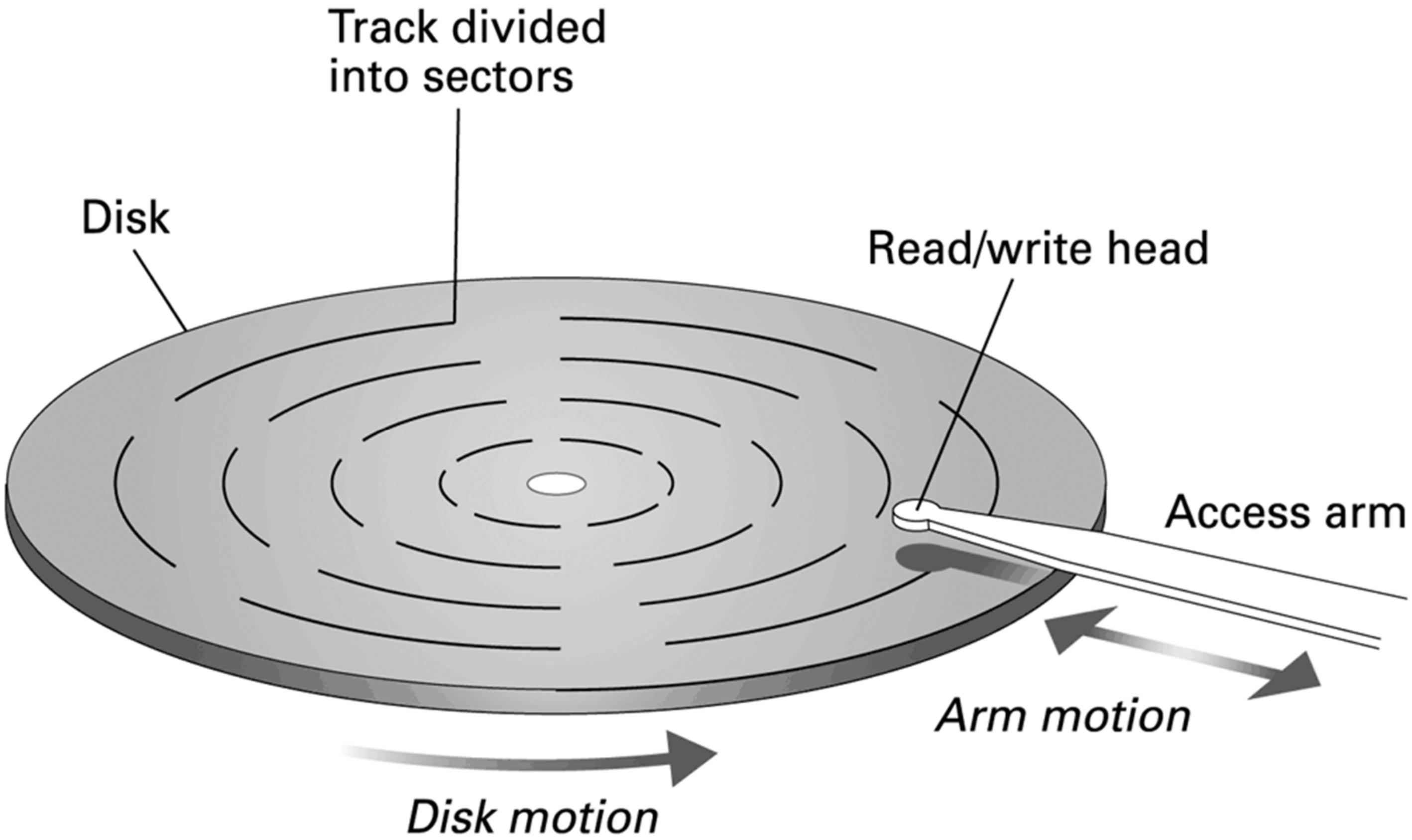


Figure 1.10 Magnetic tape storage

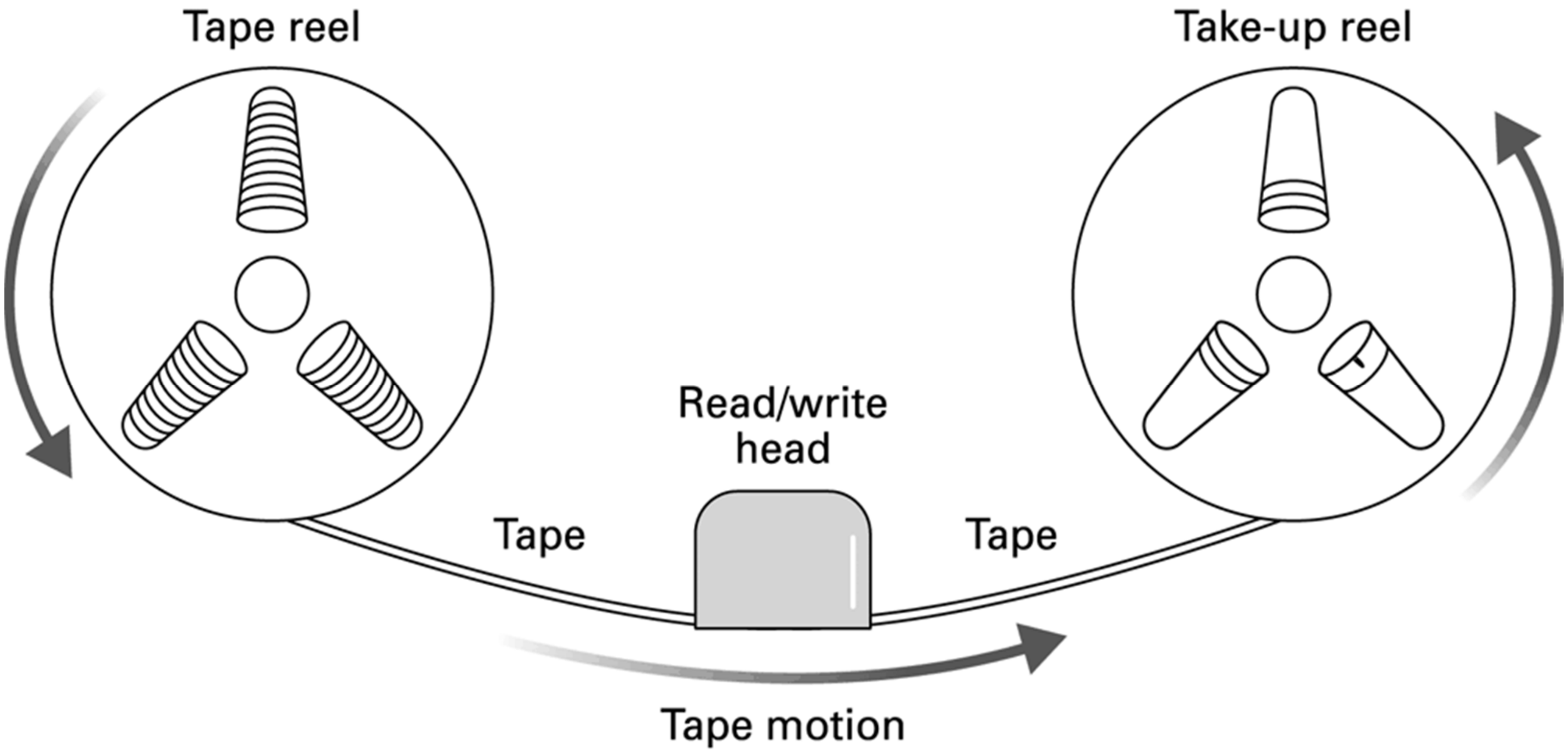
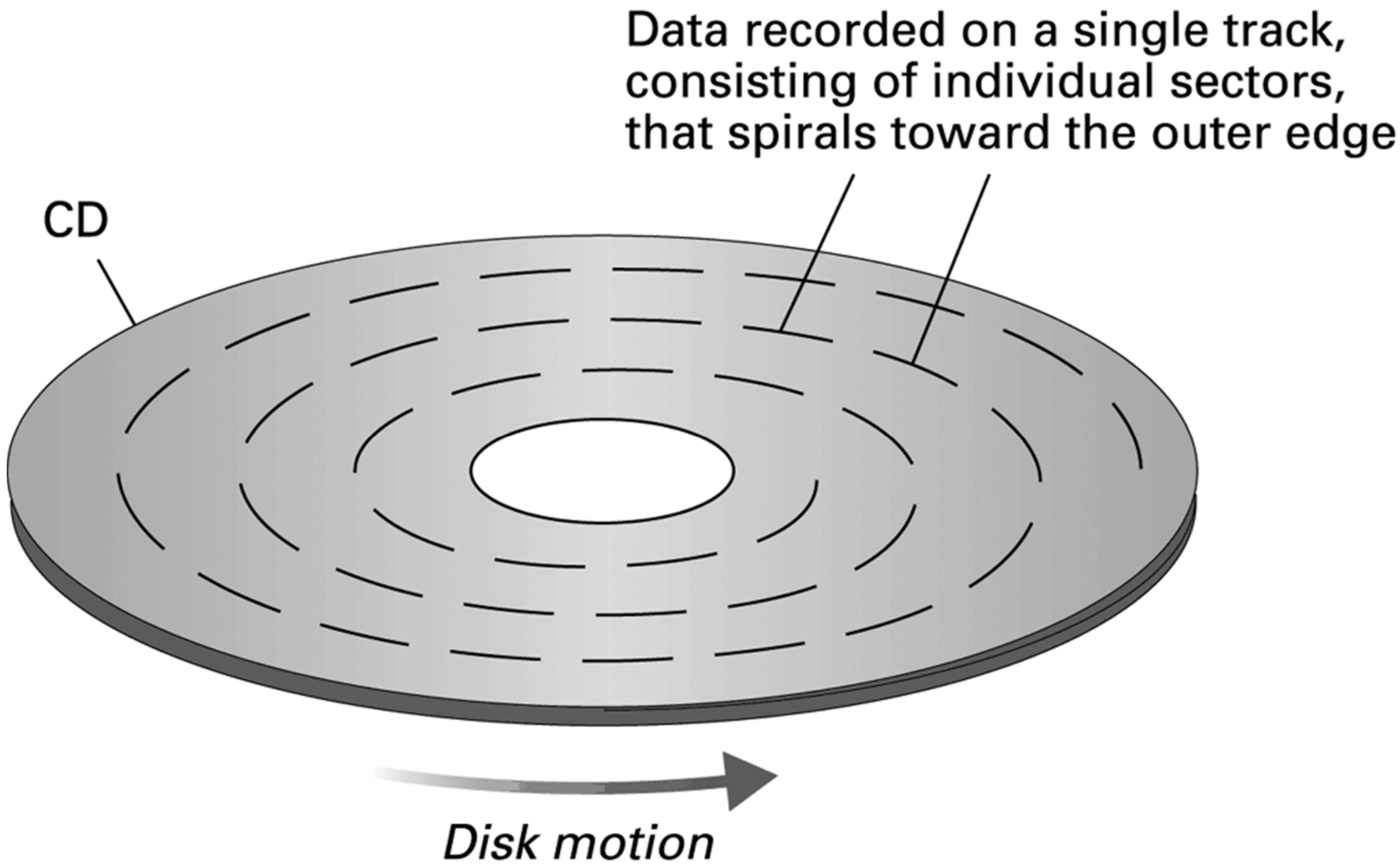


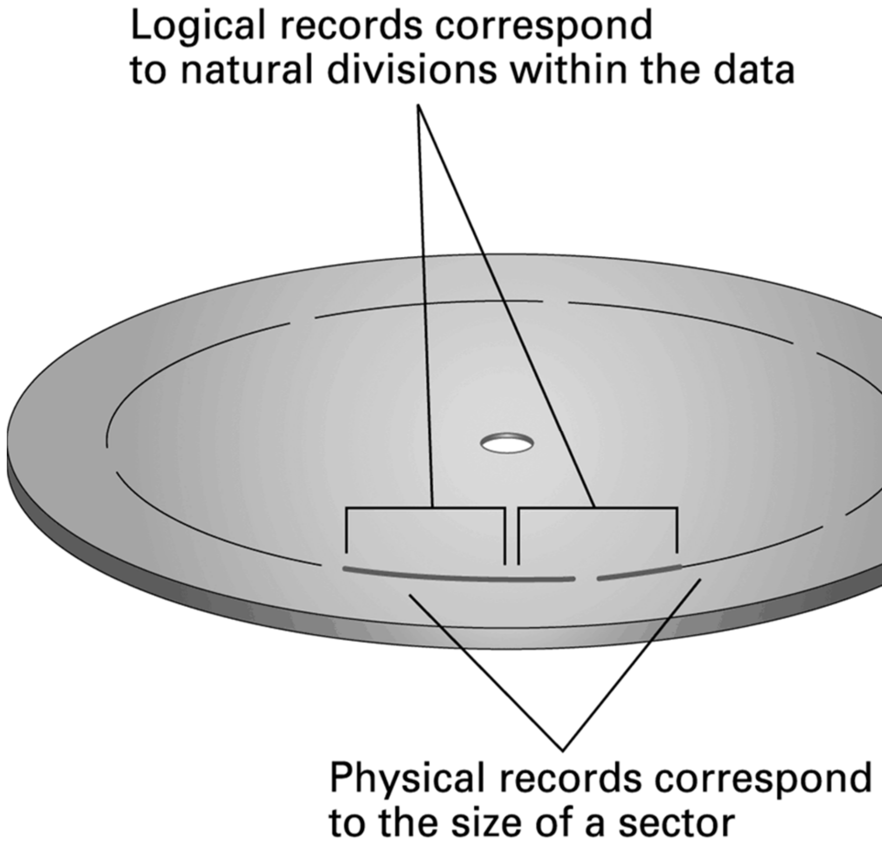
Figure 1.11 **CD storage**



Files

- **File:** A unit of data stored in mass storage system
  - **Fields** and **keyfields**
- Physical record versus Logical record
- **Buffer:** A memory area used for the temporary storage of data (usually as a step in transferring the data)

Figure 1.12 **Logical records versus physical records on a disk**



Representing Text

- **Each character (letter, punctuation, etc.) is assigned a unique bit pattern.**
  - ASCII: Uses patterns of 7-bits to represent most symbols used in written English text
  - Unicode: Uses patterns of 16-bits to represent the major symbols used in languages world side
  - ISO standard: Uses patterns of 32-bits to represent most symbols used in languages world wide

Figure 1.13 **The message "Hello." in ASCII**

01001000	01100101	01101100	01101100	01101111	00101110
H	e	l	l	o	.

Representing Numeric Values

- Binary notation: Uses bits to represent a number in base two
- Limitations of computer representations of numeric values
  - Overflow – occurs when a value is too big to be represented
  - Truncation – occurs when a value cannot be represented accurately

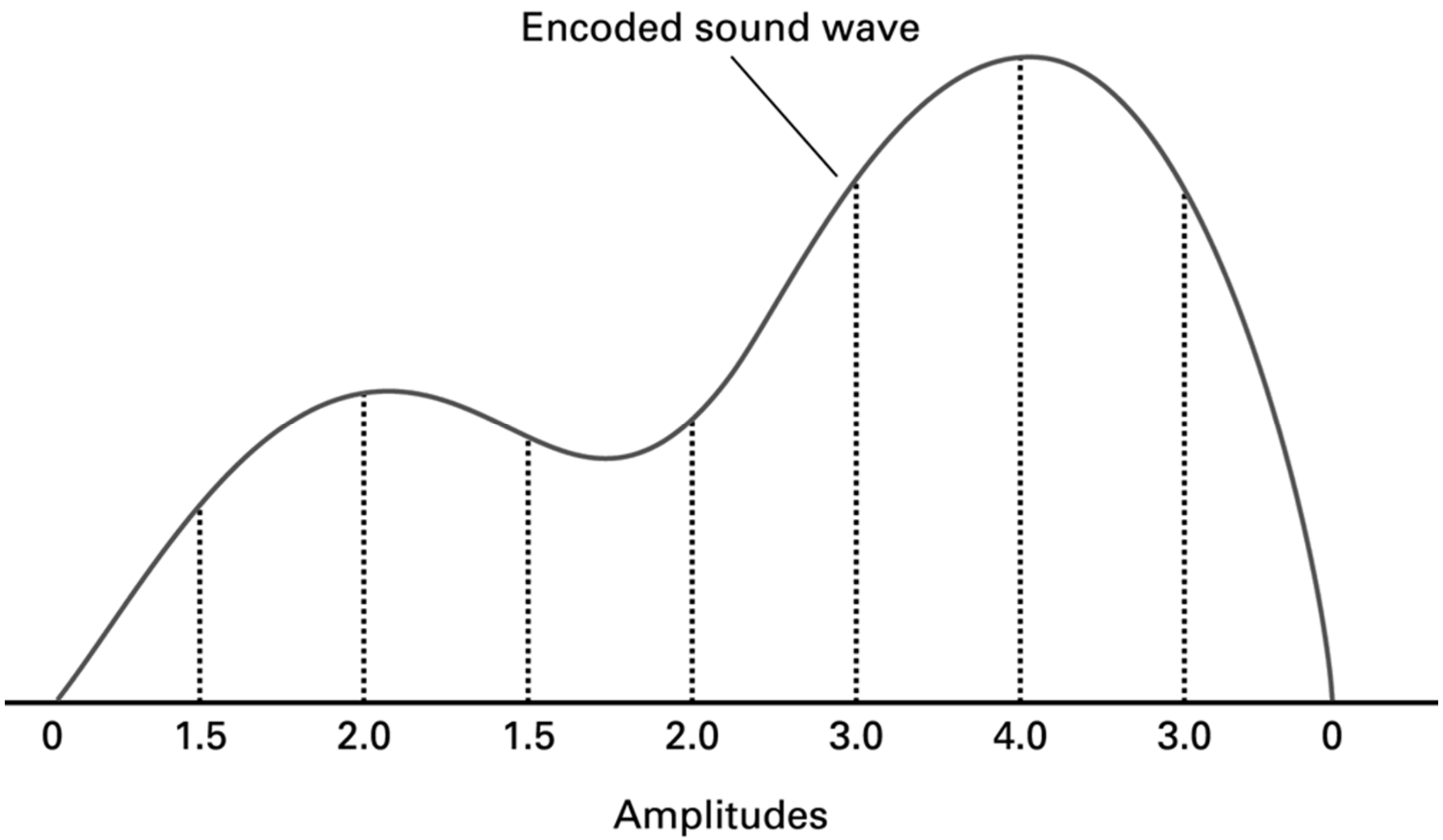
Representing Images

- Bit map techniques
  - Pixel: short for “picture element”
  - RGB
  - Luminance and chrominance
- Vector techniques
  - Scalable
  - TrueType and PostScript

Representing Sound

- Sampling techniques
  - Used for high quality recordings
  - Records actual audio
- MIDI
  - Used in music synthesizers
  - Records “musical score”

Figure 1.14 **The sound wave represented by the sequence 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0**



The Binary System

The traditional decimal system is based on powers of ten.

The Binary system is based on powers of two.

Figure 1.15 **The base ten and binary systems**

Figure 1.16 **Decoding the binary representation 100101**

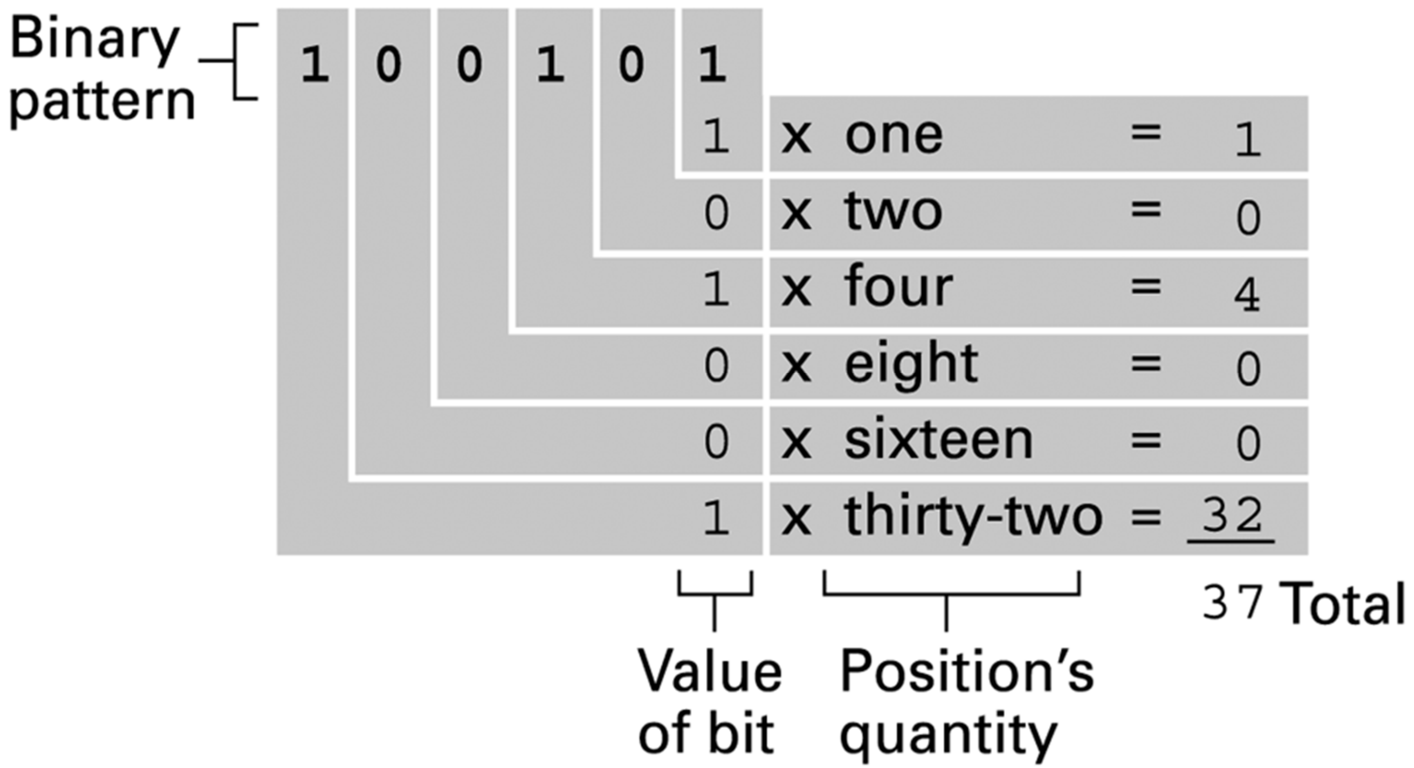


Figure 1.17 **An algorithm for finding the binary representation of a positive integer**

Step 1. Divide the value by two and record the remainder.

Step 2. As long as the quotient obtained is not zero, continue to divide the newest quotient by two and record the remainder.

Step 3. Now that a quotient of zero has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.

Figure 1.18 **Applying the algorithm in Figure 1.15 to obtain the binary representation of thirteen**

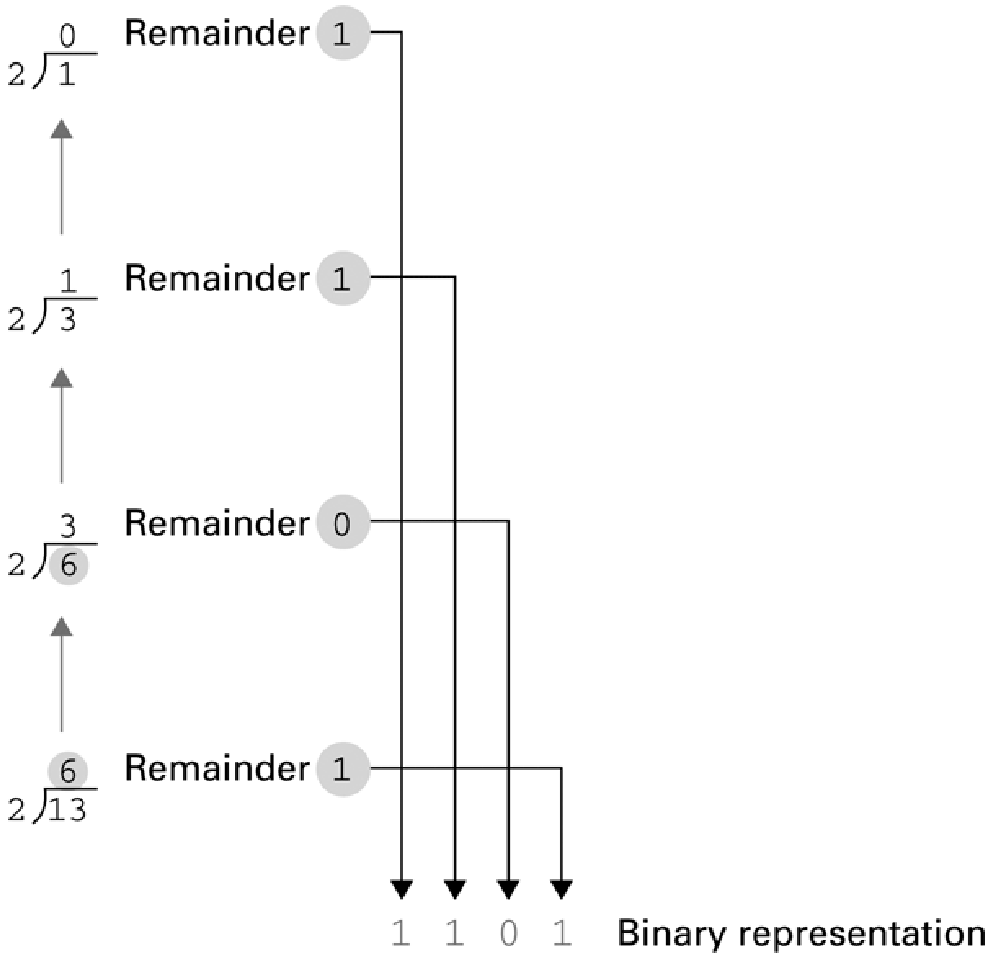
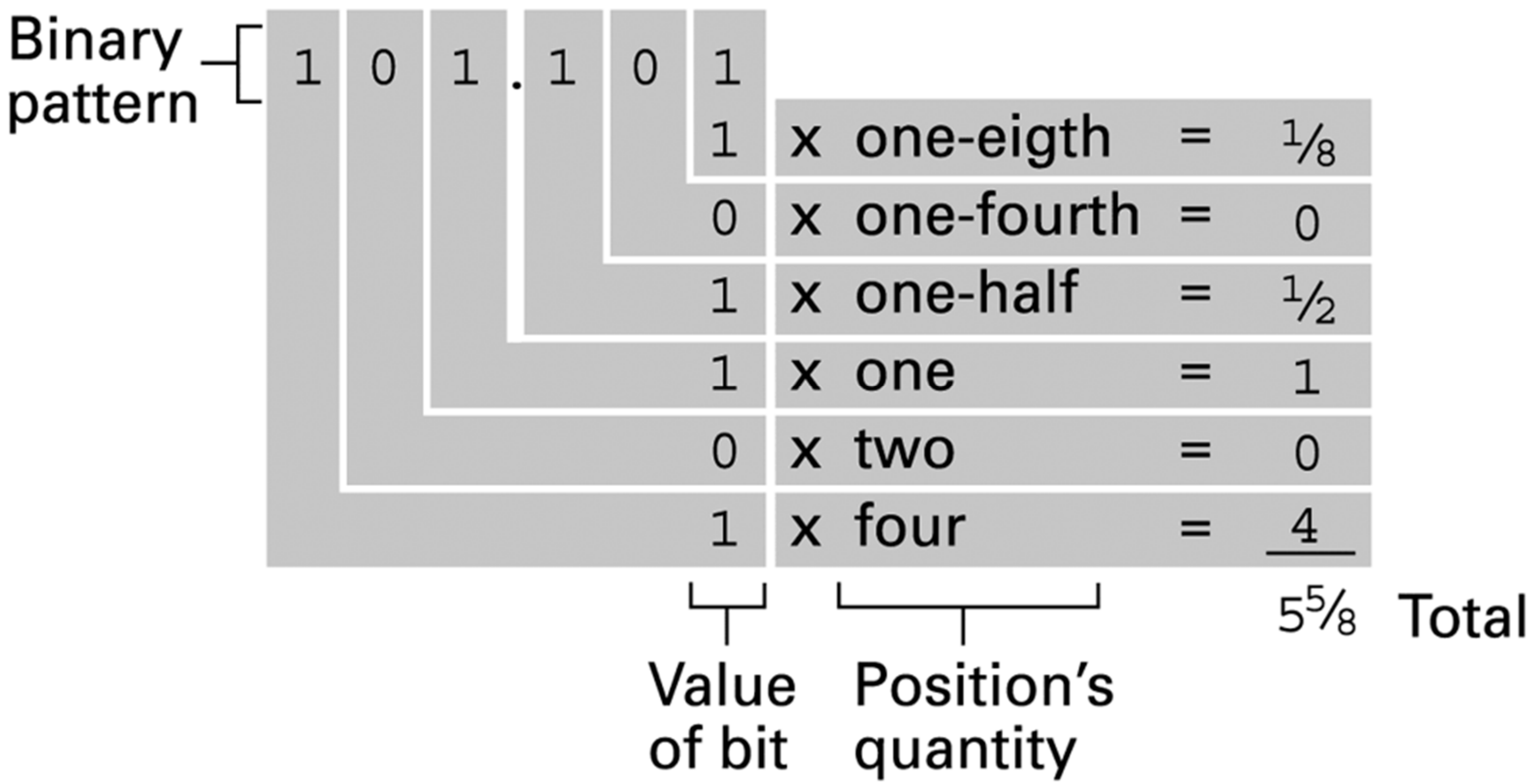


Figure 1.19 **The binary addition facts**

$$
\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array}
\qquad
\begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array}
\qquad
\begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array}
\qquad
\begin{array}{r} 1 \\ +\,1 \\ \hline 10 \end{array}
$$

Figure 1.20 **Decoding the binary representation 101.101**



Storing Integers

- **Two's complement notation:** The most popular means of representing integer values
- **Excess notation:** Another means of representing integer values
- Both can suffer from overflow errors.

Figure 1.21 Two's complement notation systems

a. Using patterns of length three

Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	−1
110	−2
101	−3
100	−4

b. Using patterns of length four

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

Figure 1.22 **Coding the value -6 in two's complement notation using four bits**

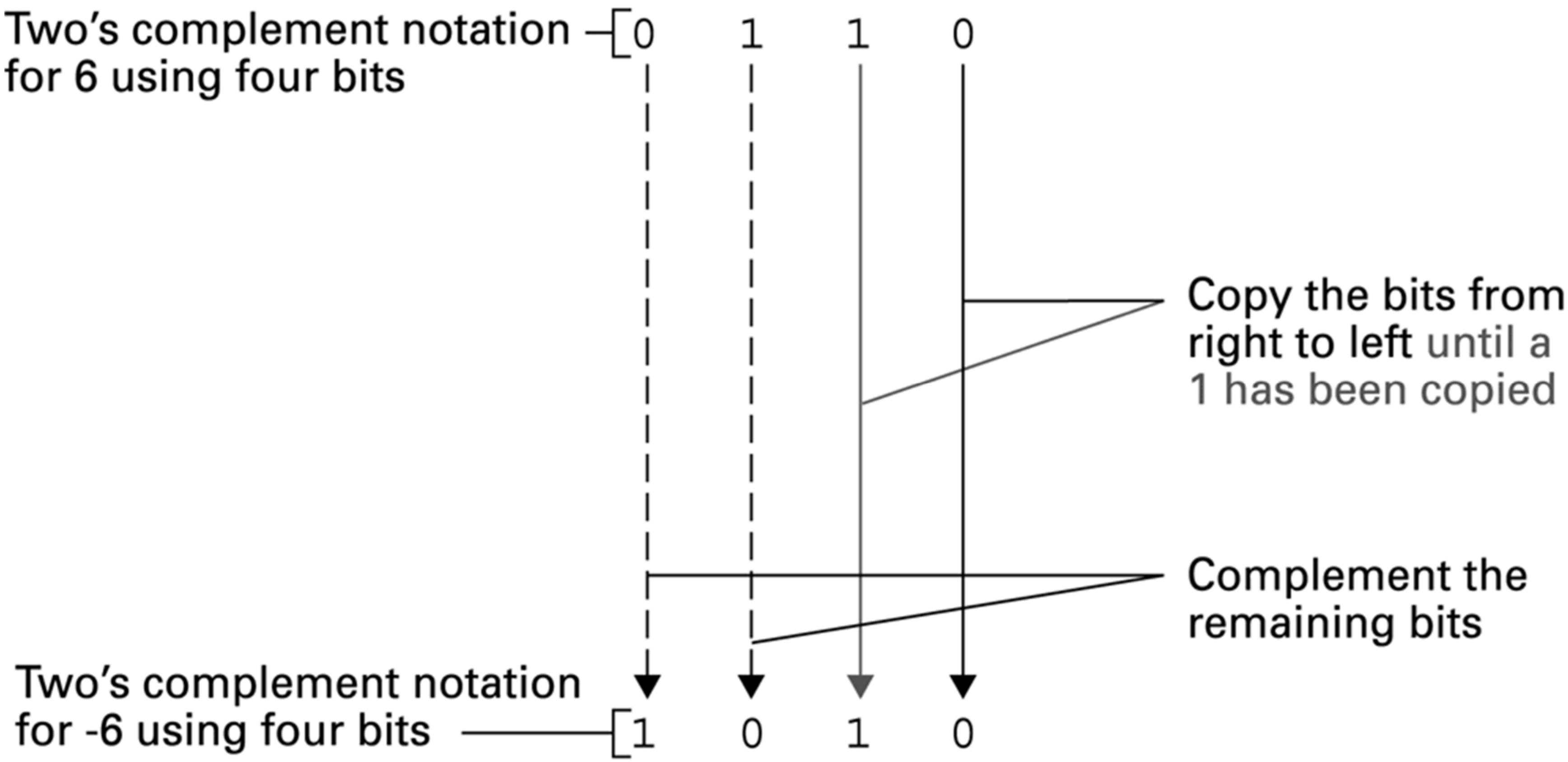


Figure 1.23 **Addition problems converted to two's complement notation**

Problem in base ten		Problem in two's complement		Answer in base ten
3 + 2	→	0011 + 0010 = 0101	→	5
−3 + −2	→	1101 + 1110 = 1011	→	−5
7 + −5	→	0111 + 1011 = 0010	→	2

Figure 1.24 An excess eight conversion table

Bit pattern	Value represented
1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	−1
0110	−2
0101	−3
0100	−4
0011	−5
0010	−6
0001	−7
0000	−8

Figure 1.25 **An excess notation system using bit patterns of length three**

Bit pattern	Value represented
111	3
110	2
101	1
100	0
011	−1
010	−2
001	−3
000	−4

Storing Fractions

- **Floating-point Notation:** Consists of a sign bit, a mantissa field, and an exponent field.
- Related topics include
  - Normalized form
  - Truncation errors

Figure 1.26 **Floating-point notation components**

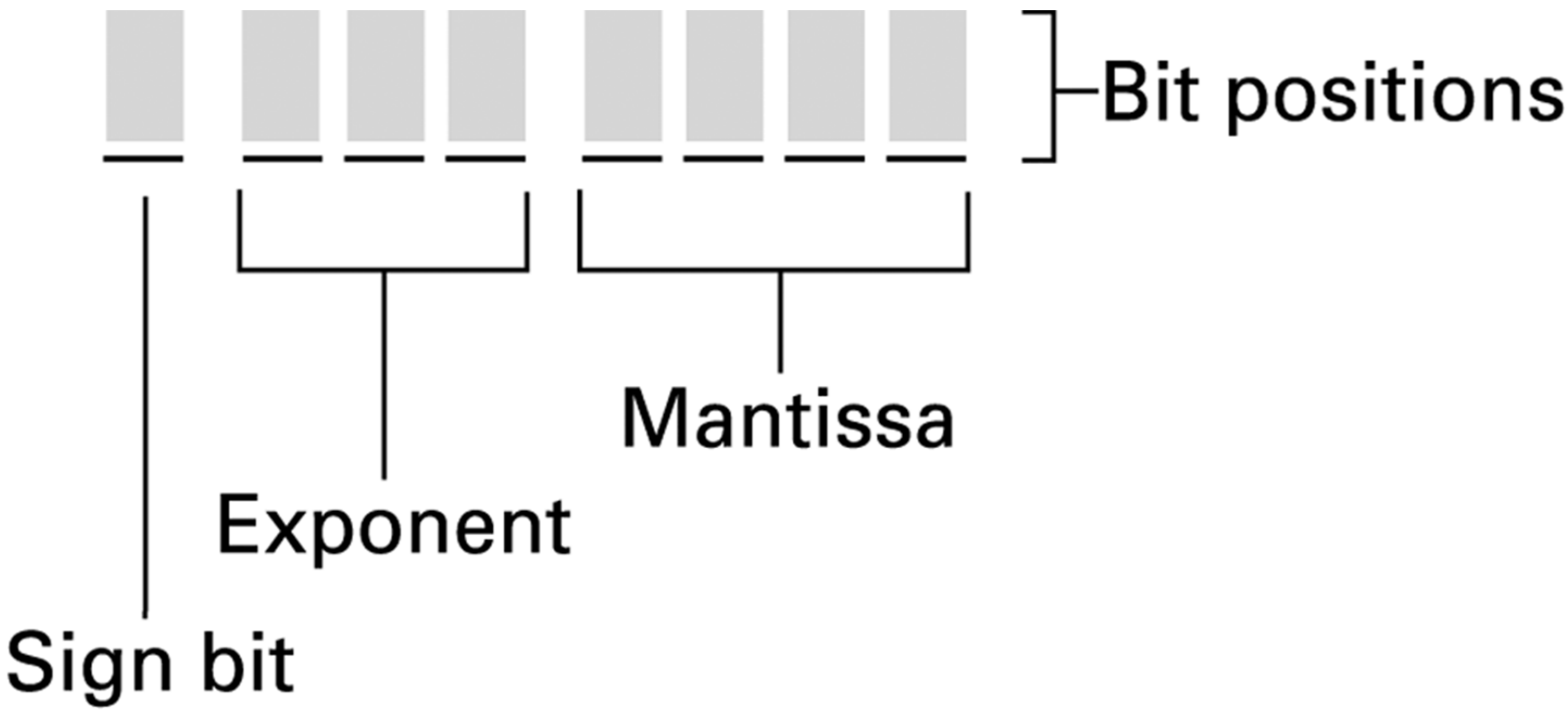
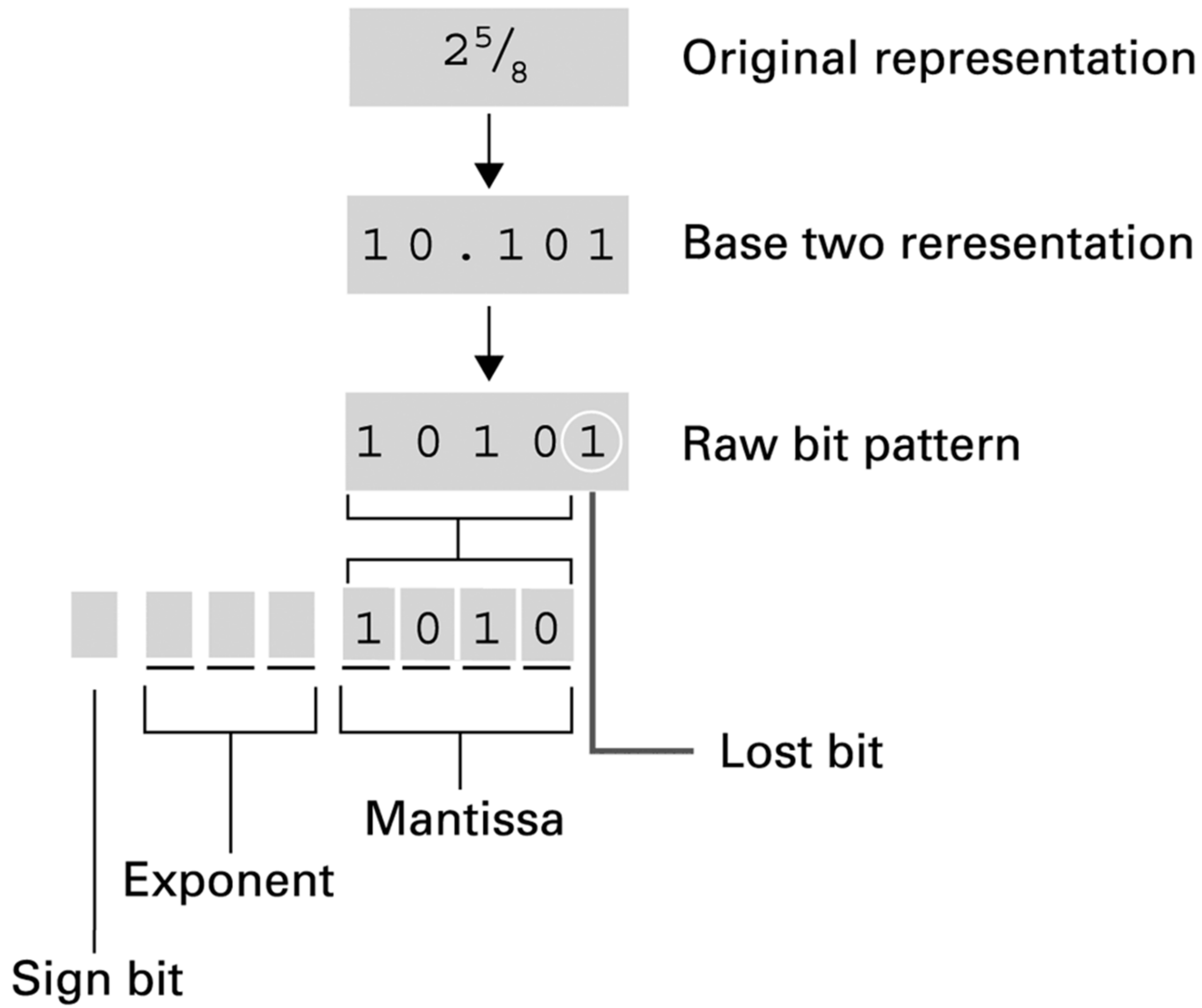


Figure 1.27 **Encoding the value 2 5⁄8**



Data Compression

- Lossy versus lossless
- Run-length encoding
- Frequency-dependent encoding
  (Huffman codes)
- Relative encoding
- Dictionary encoding (Includes adaptive dictionary encoding such as LZW encoding.)

Compressing Images

- GIF: Good for cartoons
- JPEG: Good for photographs
- TIFF: Good for image archiving

Compressing Audio and Video

- MPEG
  - High definition television broadcast
  - Video conferencing
- MP3
  - Temporal masking
  - Frequency masking

Communication Errors

- Parity bits (even versus odd)
- Checkbytes
- Error correcting codes

Figure 1.28 The ASCII codes for the letters A and F adjusted for odd parity

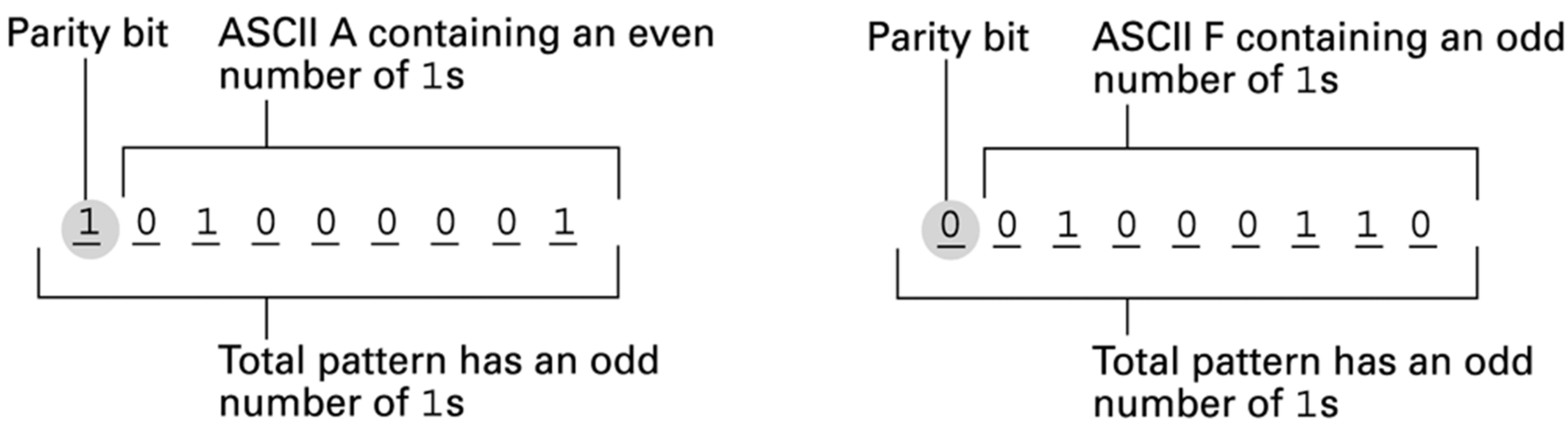


Figure 1.29 **An error-correcting code**

Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

Figure 1.30 **Decoding the pattern 010100 using the code in Figure 1.30**

Character	Code	Pattern received	Distance between received pattern and code
A	0 0 0 0 0 0	0 **1** 0 **1** 0 0	2
B	0 0 1 1 1 1	0 **1** **0** 1 **0** **0**	4
C	0 1 0 0 1 1	0 1 0 **1** **0** **0**	3
D	0 1 1 1 0 0	0 1 **0** 1 0 0	**1**
E	1 0 0 1 1 0	**0** **1** 0 1 **0** 0	3
F	1 0 1 0 0 1	**0** **1** **0** **1** 0 **0**	5
G	1 1 0 1 0 1	**0** 1 0 1 0 **0**	2
H	1 1 1 0 1 0	**0** 1 **0** **1** **0** 0	4

Smallest distance