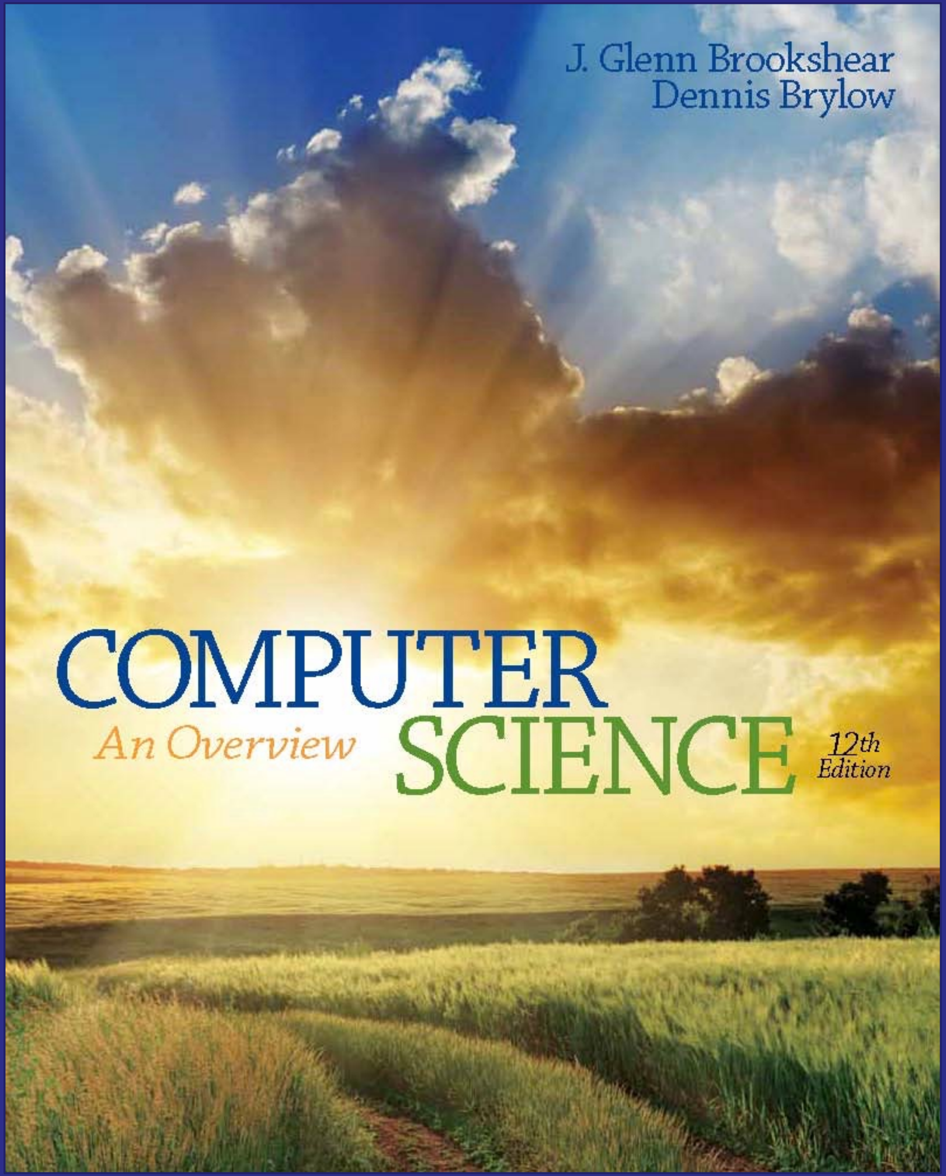
Chapter 1: Data Storage



PEARSON

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 possible input and output values of Boolean operations AND, OR, and XOR (exclusive or)**

The AND operation

	0		0		1		1
AND	0	AND	1	AND	0	AND	1
	0		0		0		1

The OR operation

	0		0		1		1
OR	0	OR	1	OR	0	OR	1
	0		1		1		1

The XOR operation

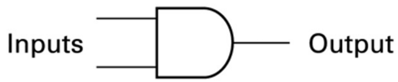
	0		0		1		1
XOR	0	XOR	1	XOR	0	XOR	1
	0		1		1		0

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

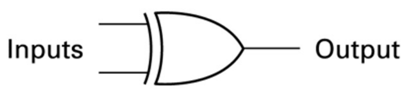


Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

OR

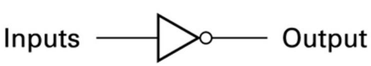
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR



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

NOT



Inputs	Output
0	1
1	0

Flip-flops (Memory)

- **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

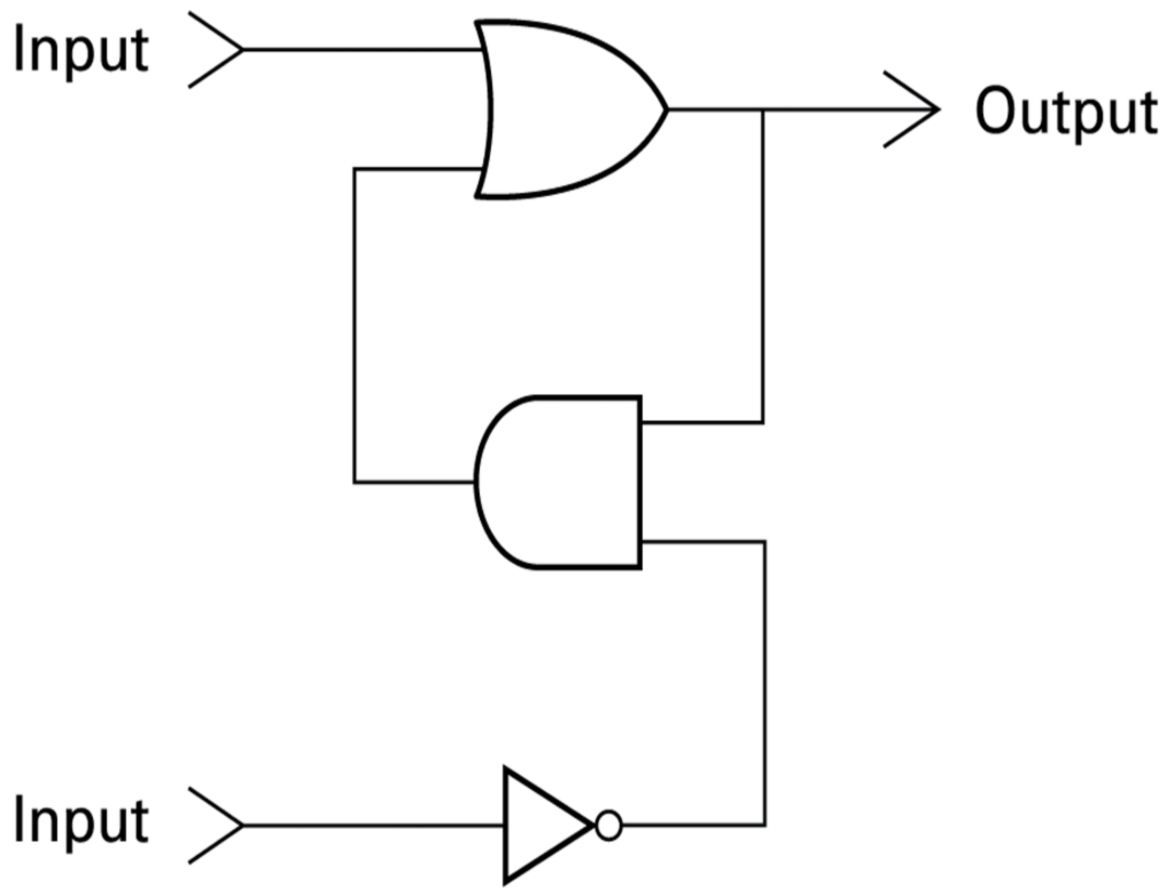


Figure 1.4 **Setting the output of a flip-flop to 1**

a. First, a 1 is placed on the upper input.

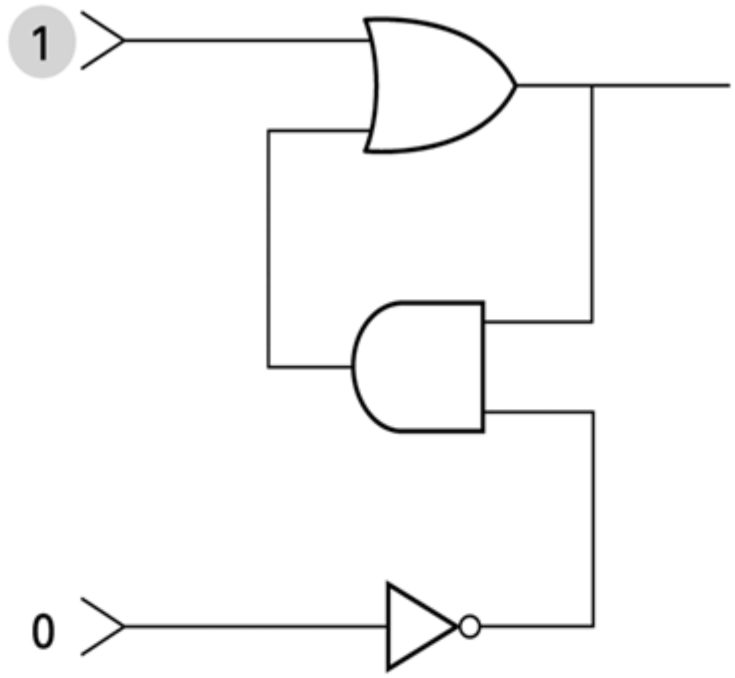


Figure 1.4 **Setting the output of a flip-flop to 1 (continued)**

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

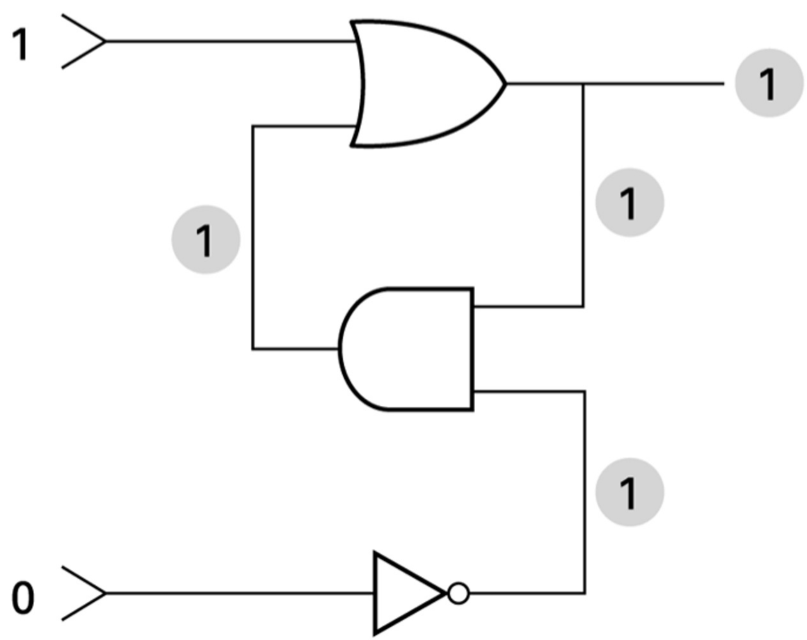
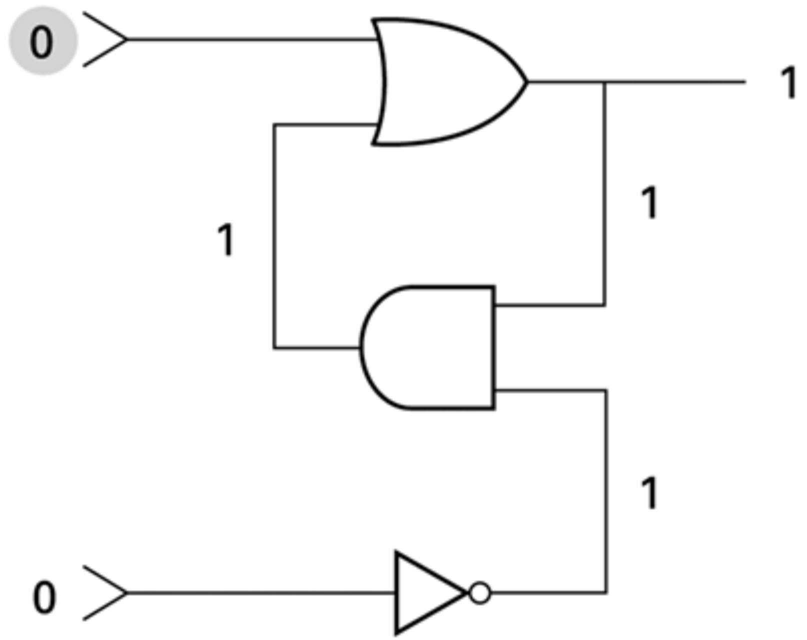


Figure 1.4 Setting the output of a flip-flop to 1 (continued)

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



The Binary Number System

The traditional decimal system is based on powers of ten.

The Binary system is based on powers of two.

Figure 1.13 **The base ten and binary systems**

Figure 1.14 **Decoding the binary representation 100101**

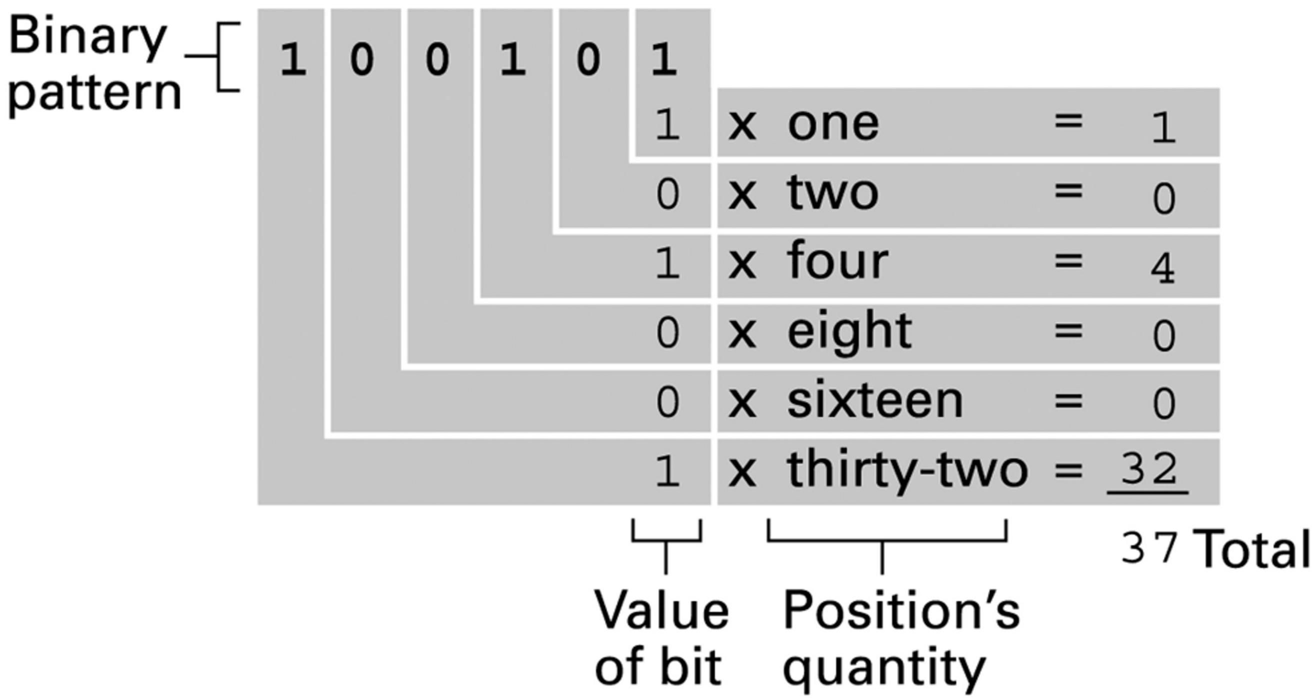


Figure 1.15 **An algorithm for finding the binary representation of a positive integer**

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

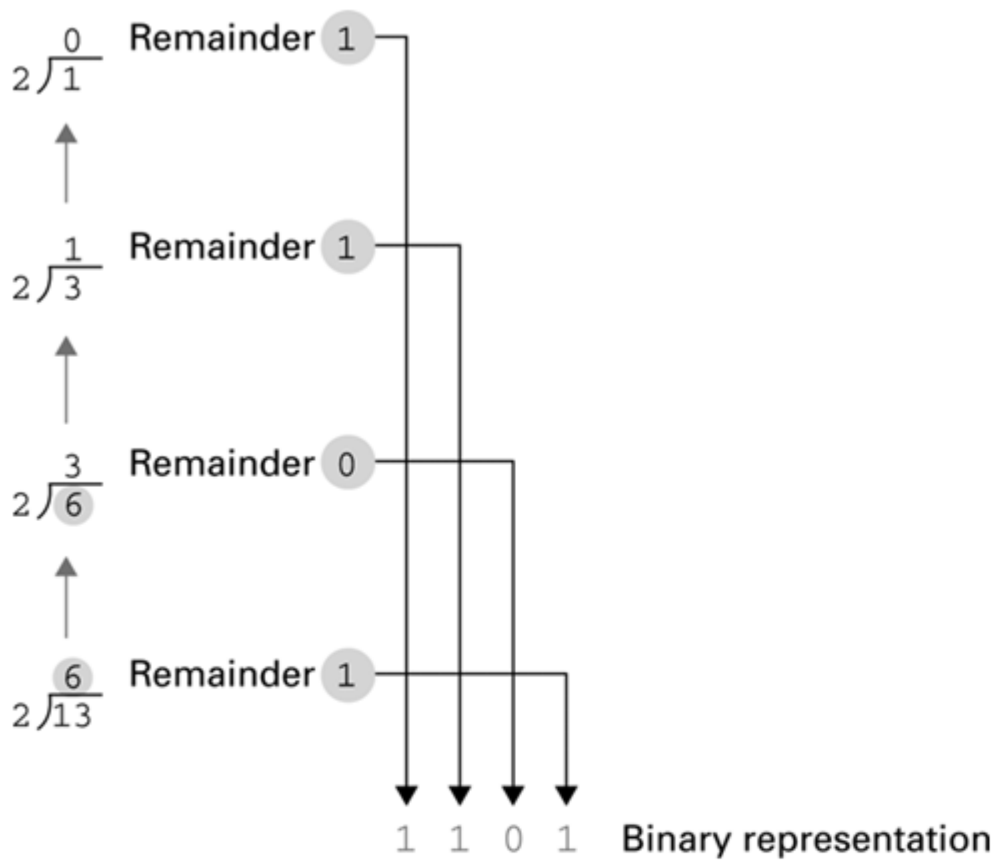


Figure 1.17 The binary addition facts

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

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

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

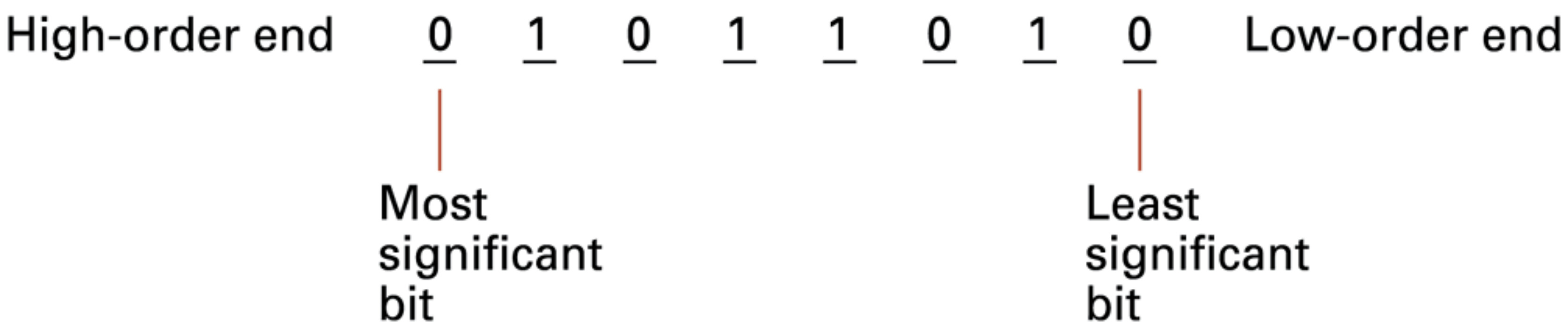
Figure 1.6 **The hexadecimal coding system**

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

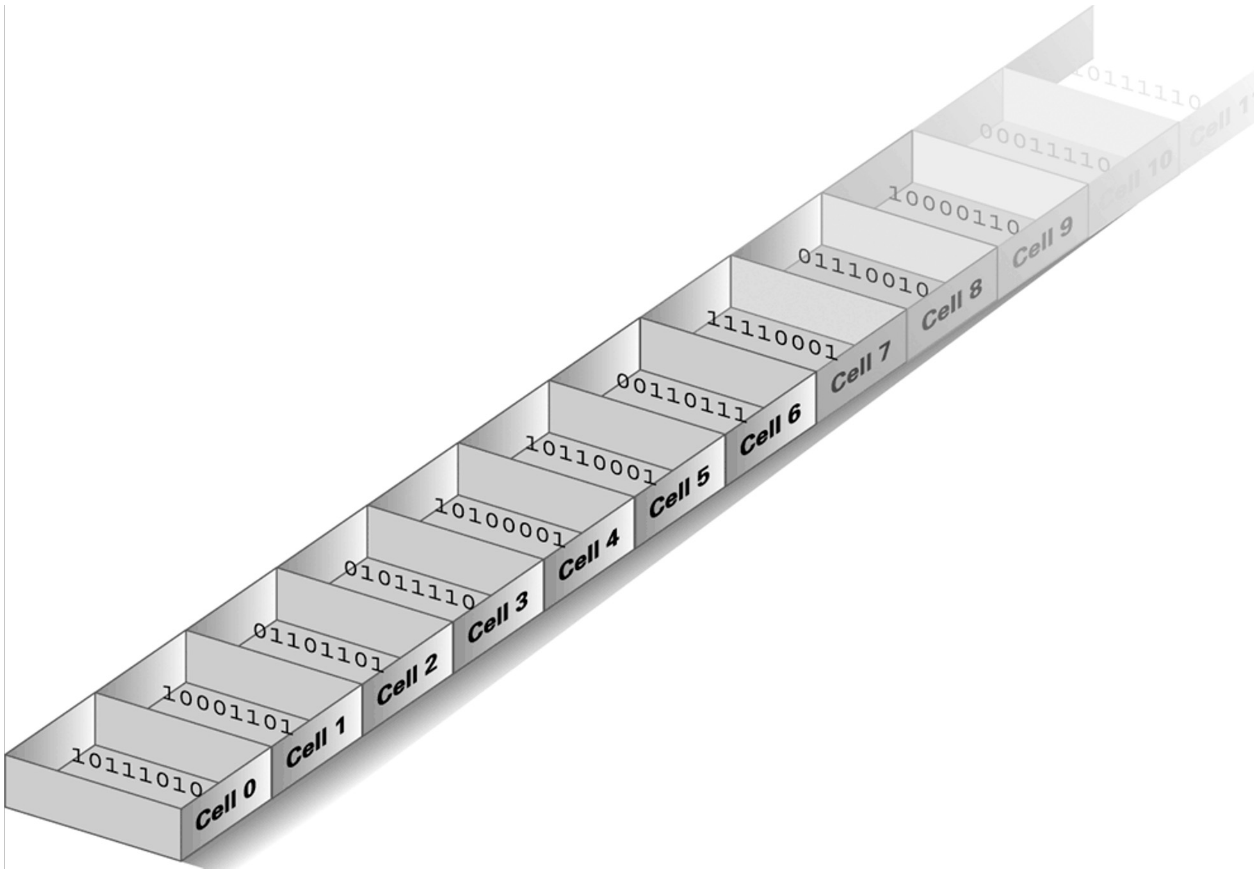
Figure 1.7 The organization of a byte-size memory cell



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
- **Megabyte:** 2^{20} bytes = 1,048,576 bytes
  - Example: 3 MB = 3 times 1,048,576 bytes
- **Gigabyte:** 2^{30} bytes = 1,073,741,824 bytes
  - Example: 3 GB = 3 times 1,073,741,824 bytes

Mass Storage

- Additional devices:
  - Magnetic disks
  - CDs
  - DVDs
  - Magnetic tape
  - Flash drives
  - Solid-state disks
- Advantages over main memory
  - Less volatility
  - Larger storage capacities
  - Low cost
  - In many cases can be removed

Figure 1.9 **A magnetic disk storage system**

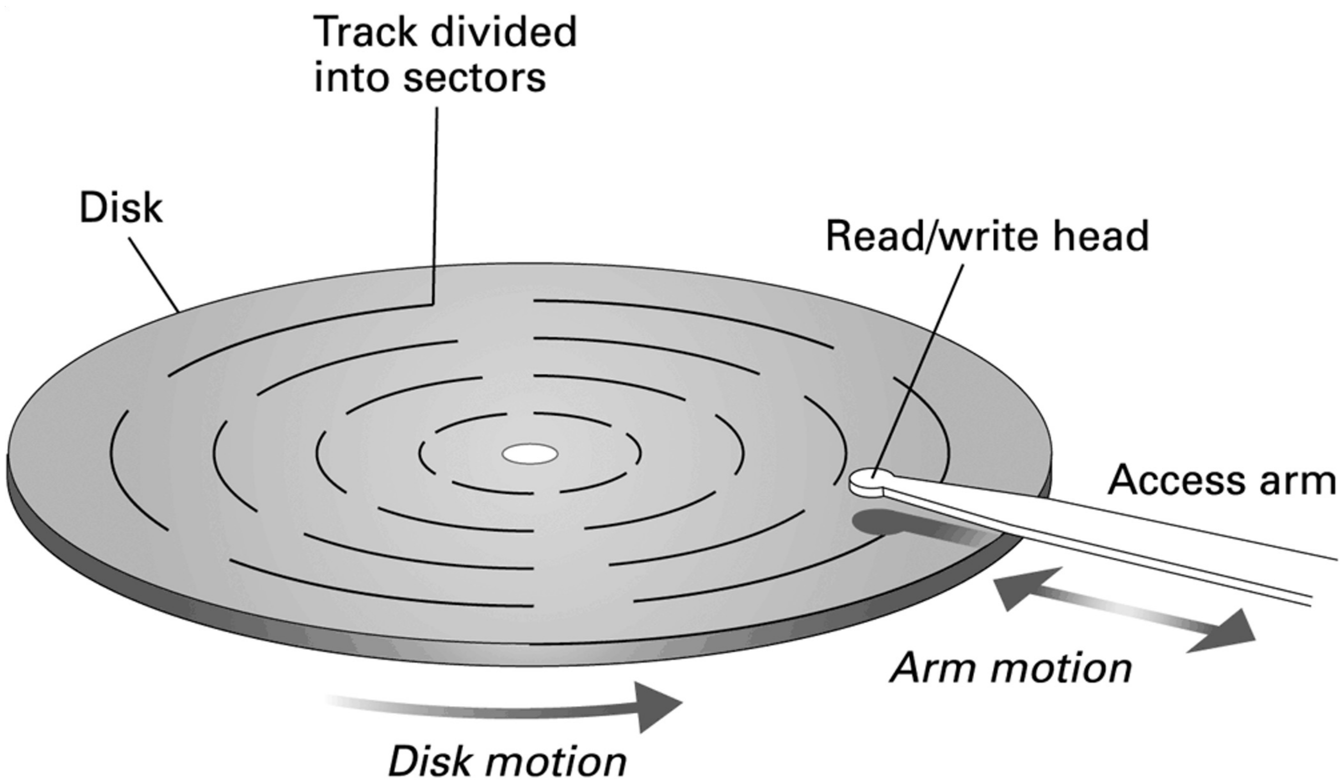
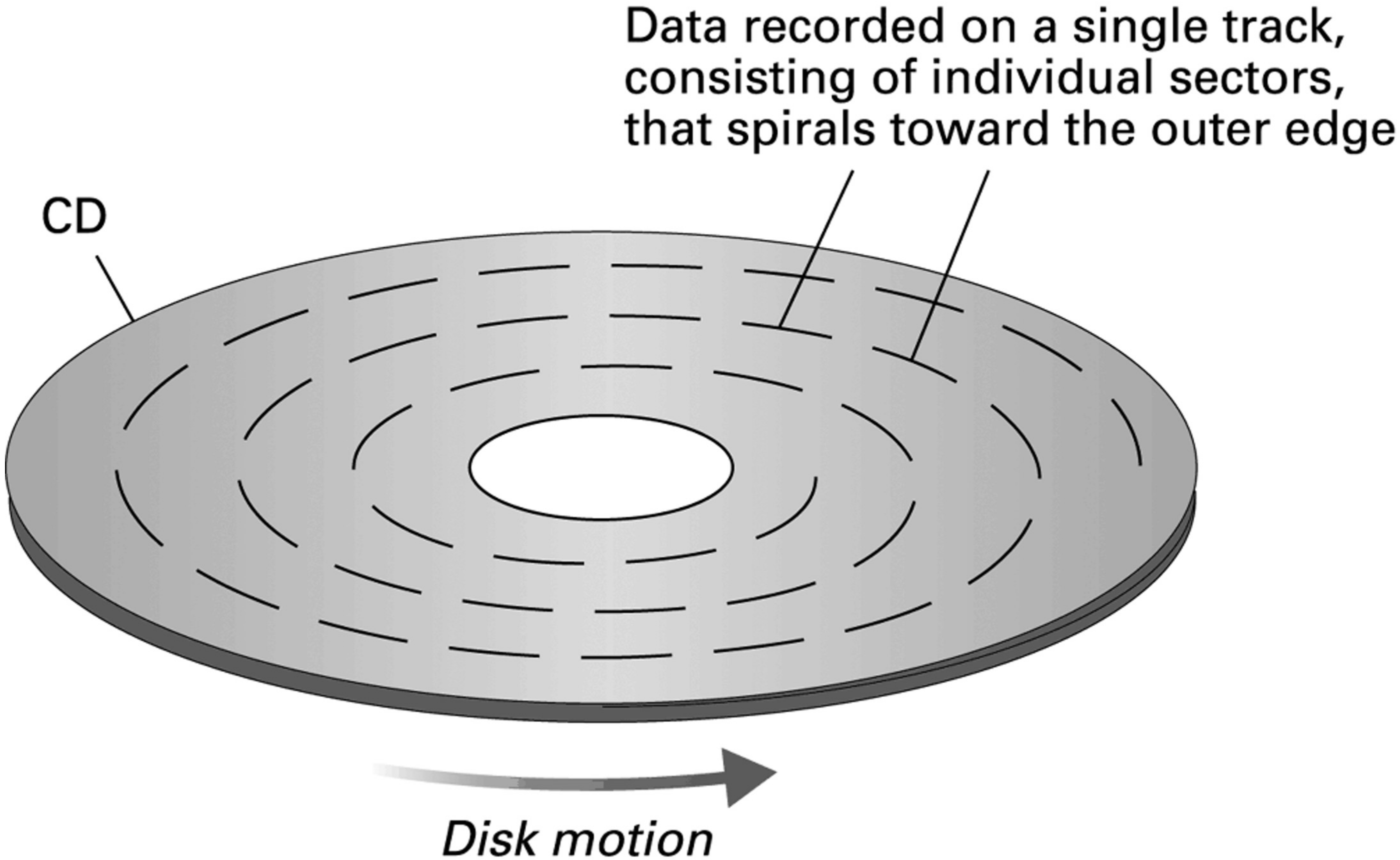


Figure 1.10 **CD storage**



Flash Drives

- **Flash Memory** – circuits that traps electrons in tiny silicon dioxide chambers
- Repeated erasing slowly damages the media
- Mass storage of choice for:
  - Digital cameras
  - Smartphones
- **SD Cards** provide GBs of storage

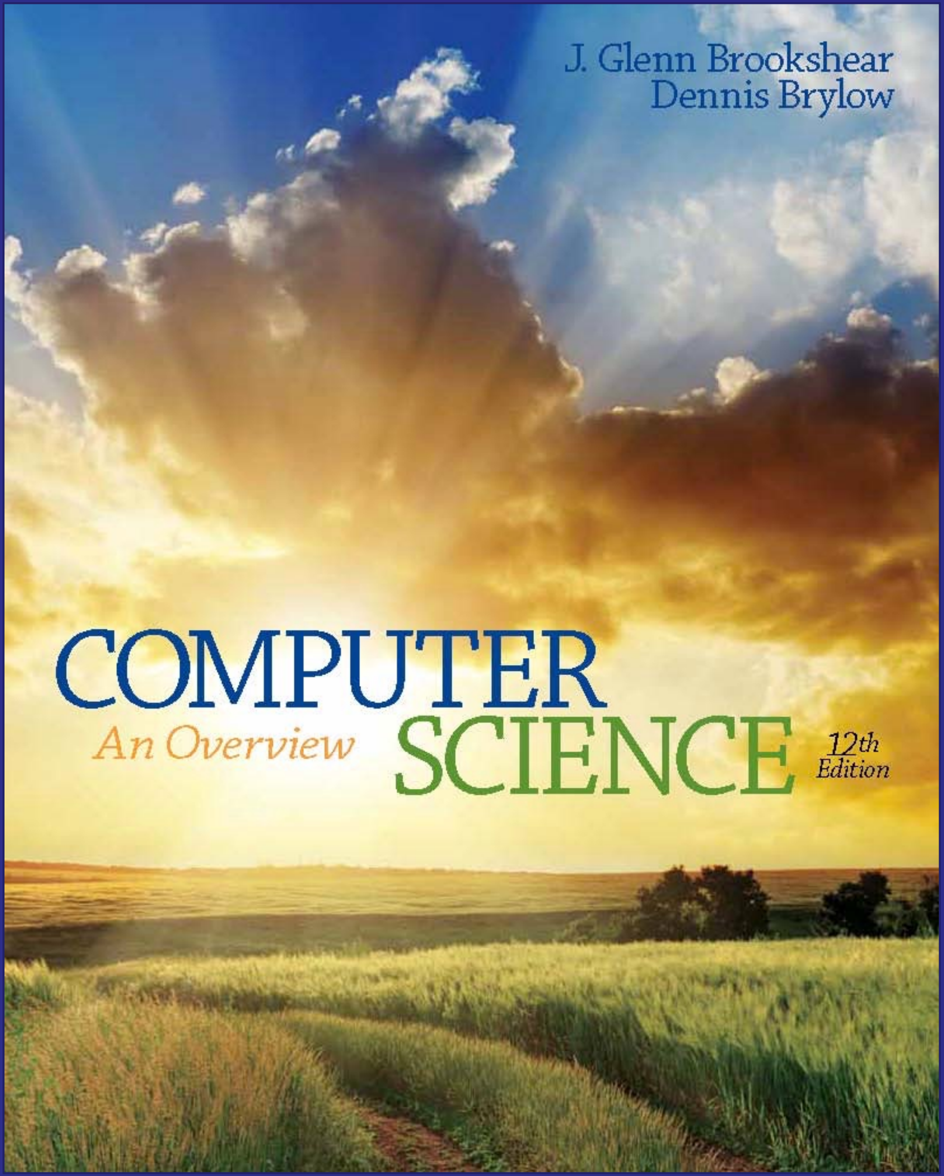
Representing Text

- **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
  - ISO developed a number of 8 bit extensions to ASCII, each designed to accommodate a major language group
  - **Unicode**: Uses patterns up to 21-bits to represent the symbols used in languages world wide, 16-bits for world's commonly used languages

Figure 1.11 The message "Hello." in ASCII or UTF-8 encoding

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

End of Chapter



PEARSON