

Chapter 4: Data Representation: How Computers Store Information

In this chapter, you will learn how computers represent text, numbers, images, audio, and video using digital data. You will explore binary, hexadecimal, and common methods of storing information while discovering how simple patterns of zeros and ones make modern computing possible. Understanding these concepts provides the foundation for many other areas of computer science.

Learning Objectives

- Explain why computers use the binary number system.
- Define the terms bit, byte, and binary digit.
- Convert simple binary numbers to decimal and decimal numbers to binary.
- Describe how text is represented using character encoding systems such as ASCII and Unicode.
- Explain how images, audio, and video are stored digitally.
- Compare units of digital storage, including bytes, kilobytes, megabytes, gigabytes, terabytes, and petabytes.
- Differentiate between data size and data transfer speed.
- Explain how data compression reduces file size.
- Describe how errors can occur during data transmission and how computers detect and correct them.
- Explain why understanding data representation is important in today's digital world.

Maria is preparing a presentation for her biology class. She types her report in a word processor, inserts photographs taken with her smartphone, records a short narration, and adds a short video demonstrating an experiment. Before submitting the assignment, she uploads everything to her cloud storage account.

Although Maria sees text, colorful images, clear audio, and smooth video, her computer stores every part of the project in the same basic form: billions of tiny **binary digits** called bits. The operating system, applications, and storage devices all work together to convert those bits into the words, pictures, sounds, and videos Maria sees on the screen, but the computer represents every type of digital information using only two symbols: 0 and 1.

Every time you send a text message, stream a movie, save a document, or upload a photo, your computer is processing data. Although digital information appears as text, images, music, and video, computers store and process all of it using binary numbers. Understanding data representation helps you make informed decisions about file storage, internet usage, image quality, backups, and data security. It also provides the foundation for understanding many other computing topics, including networking, cybersecurity, databases, programming, and artificial intelligence.

Whether you become a healthcare professional, business manager, teacher, engineer, artist, or entrepreneur, you'll work with digital information every day. Knowing how computers represent and manage data will help you use technology more effectively and understand the systems that increasingly shape our personal and professional lives.

Think About It

Imagine sending your favorite photo to a friend. Within seconds, the image travels across the internet, arrives on another device, and looks exactly the same. This happens even though computers only understand two electrical states: on and off. How can millions of colors, words, sounds, and videos all be represented using only zeros and ones? By the end of this chapter, you'll understand the surprisingly simple idea that makes all modern computing possible.

Binary

Every photograph you take, every song you stream, every document you write, and every website you visit has one thing in common: inside the computer, they are all stored as numbers.

Unlike people, computers cannot understand letters, pictures, sounds, or videos directly. Instead, they represent all information using only two possible values: **0** and **1**.

This system is called the **binary number system**, and it forms the foundation of all digital computing.

Binary Makes Digital Devices Possible

Computers are built from billions of tiny electronic switches called **transistors**. Each transistor has only two reliable states:

- On
- Off

These two electrical states are easy for electronic circuits to recognize, even when operating billions of times each second.

To represent these two states, computers use:

ELECTRICAL STATE	BINARY VALUE
Off	0
On	1

Because every transistor has only two possible states, binary is the natural language of computers.

Why Only Two Numbers?

To a person, it might seem more natural to use ten digits, each with a different numerical symbol we can recognize visually, for example, Arabic numerals: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

But computers do not perceive visual symbols the way people do. For this to work on a computer, each value would have to be represented by a different voltage level, for example, 10%, 20%, 30%, etc. But what if the computer reads a value with a

voltage level of 15%? Does it round down to 10% or up to 20%. This may seem a small difference, but in an equation, could end up being a huge error. It would also take longer to process, because the computer would need to investigate.

By using only two states (0%, off or 100%, on) the binary system makes computers reliable, fast, inexpensive, and accurate. This simple idea has allowed engineers to build increasingly powerful computers for more than seventy years.

Binary Is Everywhere

When you type the word *Hello*, your computer does **not** actually store the letters H-E-L-L-O. Instead, each letter is converted into a series of binary digits.

The same thing happens for photographs, movies, music, web pages, video games, text messages, and more.

Everything eventually becomes long sequences of 0s and 1s.



Figure 4.1 – The Binary World

This illustration shows that all types of digital information, including text, images, audio, and video, are ultimately represented inside a computer as binary digits (0s and 1s). A computer converts different forms of information into binary for processing, storage, and transmission.

What Is a Bit?

A **bit** (short for **binary digit**) is the smallest unit of digital information. A bit can contain only one of two values: 0 or 1.

Think of it like a light switch; it can be either ON or OFF.

One bit by itself cannot represent very much information; it can only represent two values: 0 or 1. But two bits together can represent four different combinations: 00, 01, 10, and 11. That's twice as many combinations. Adding a third bit gives you eight possible values: 000, 001, 010, 011, 100, 101, 110, 111, and adding a fourth bit gives you sixteen: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111.

Every additional bit doubles the number of possible combinations:

Possible Values per Bit

Number of Bits	Possible Values
1 bit	2 values
2 bits	4 values
3 bits	8 values
4 bits	16 values
5 bits	32 values
6 bits	64 values
7 bits	128 values
8 bits	256 values

Table 4.1 – Possible Values per Bit

What Is a Byte?

Because one bit can represent only two possibilities, computers group bits together.

A group of **eight bits** is called a **byte**.

A byte can represent 256 different combinations of bits, from 00000000 to 11111111.

A single byte can represent:

- one letter
- one number
- one punctuation symbol
- many other kinds of information
- 256 different combinations

Almost every measure of computer storage begins with the byte.

Units of Digital Information

Storage Unit	Approximate Size
Byte (B)	8 bits
Kilobyte (KB)	About 1 thousand bytes (1,024 bytes)
Megabyte (MB)	About 1 million bytes (1,024 kilobytes)
Gigabyte (GB)	About 1 billion bytes (1,024 megabytes)
Terabyte (TB)	About 1 trillion bytes (1,024 gigabytes)
Petabyte (PB)	About 1 quadrillion bytes (1,024 terabytes)

Table 4.2 – Units of Digital Information

We'll learn much more about storage units later in this chapter.

Did You Know?

A modern smartphone contains **tens of billions of transistors**. Each transistor switches between two electrical states billions of times every second, allowing the device to perform everything from sending text messages to running sophisticated AI applications.

AI Activity

Ask an AI assistant such as ChatGPT, Copilot, or Gemini:

"Explain binary numbers using an everyday analogy without using any mathematical formulas." Then consider the following:

1. What analogy did the AI use?
2. Did the explanation make binary easier to understand?
3. Was the explanation accurate?
4. How could you improve your prompt to make the explanation even clearer?

Base

Number Systems

The Decimal Number System

The decimal number system is a place system that uses a base of **10**.

The rightmost place is the ones place. Moving from right to left, the next place is the tens place, then the hundreds place, the thousands place, and so on. Each place is **10** times the previous place.

In the decimal (base 10) number system, there are 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. If you are counting up to ten in the decimal system, 0 through 9 are represented in the ones place, but if you add one more, ten is represented with a 1 in the tens place and 0 in the ones place.

A Number Represented in Decimal

Place:	1000 (Thousands)	100 (Hundreds)	10 (Tens)	1 (Ones)
Digit:	1	0	4	3
Value:	1x1000	0x100	4x10	3x1

Table 4.3 – A Number Represented in Decimal

The number 1043 means:

- 1 group of 1000
- 0 groups of 100
- 4 groups of 10
- 3 groups of 1

You cannot leave out the 0 in the 100s place. 1043 is not the same as 143.

The Binary Number System

The binary system uses a base of 2. There are 2 digits: 0 and 1. Binary is used in computers because each bit (**binary digit**) can have just one of two values. We usually represent those two values as 1 and 0, but it can also be thought of as true or false; on or off.

The binary number system is a place system. The rightmost place is the ones place. As we move from right to left, each place is 2 times the previous place. If you are counting up in binary, 0 through 1 are represented in the ones place, but if you add one more, two is represented as 1 in the twos place and 0 in the ones place.

A Number Represented in Binary

Place:	8	4	2	1
Digit:	1	0	1	1
Value:	1x8	0x4	1x2	1x1

Table 4.4 – A Number Represented in Binary

The value of the binary number 1011 = $1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 = 8 + 2 + 1 = 11$ decimal.

For practical purposes we just add together the value of each place that has a one in it.

We read the binary number 1011 as "one zero one one." Do not say "one thousand eleven," those terms refer to the decimal system.

Binary and Decimal Conversions

Converting from Binary to Decimal

Let's, for example, convert the binary number **11011** to decimal:

1. Write out the place values above each digit. Start on the right with the ones place, then go from right to left and multiply by 2 for each additional place: 1, 2, 4, 8, 16, 32, 64, 128, 256, etc.

$$\begin{array}{rcccccc} 16 & 8 & 4 & 2 & 1 & \\ 1 & 1 & 0 & 1 & 1 & \end{array}$$

2. Add together all the place values that have a 1 in them.

$$16 + 8 + 2 + 1 = 27$$

The binary number 11011, converted to a decimal, is 27.

In speaking we would say 11011 in base 2 is 27 in base 10.

We can write this as $11011_2 = 27_{10}$.

We could also say 11011 in binary equals 27. (If we don't specify a base, it is assumed that we mean decimal.)

Converting from Decimal to Binary

There are two ways to convert decimal to binary.

Method 1: Subtraction

Let's, for example, convert the decimal number **27** back to binary.

Write out the place values for binary numbers from **right to left**. Start with 1 and multiply by 2 for each additional place until the next place value is greater than the number we are converting:

$$16 \quad 8 \quad 4 \quad 2 \quad 1$$

*The next place value in the sequence would be 32, which is greater than 27, so we can stop at 16.

For each place value from **left to right**: if the value to convert is greater than or equal to ($\geq$) the place value, write a 1 in the place. Subtract the place value from the value to convert and use that number as the remaining value to convert for the next step.

27 is greater than 16. Write 1 in the 16s place.

16	8	4	2	1
1				

With the 16s represented, subtract 16 from 27 to get the remaining value to convert. $27 - 16 = 11$.

11 is greater than 8. Write 1 in the 8s place.

16	8	4	2	1
1	1			

With the 8s represented, subtract 8 from 11 to get the remaining value to convert. $11 - 8 = 3$.

3 is not greater than or equal to 4. Write 0 in the 4s place.

16	8	4	2	1
1	1	0		

Because no 4s are represented, the remaining value stays the same. **3**.

3 is greater than 2. Write 1 in the 2s place.

16	8	4	2	1
1	1	0	1	

With the 2s represented, subtract 2 from 3 to get the remaining value to convert. $3 - 2 = 1$.

1 is equal to 1. Write 1 in the 1s place.

16	8	4	2	1
1	1	0	1	1

With all values represented and the remaining value at zero ($1 - 1 = 0$), we are done with the conversion.

To check your work, you can use the previous method to add up the values represented with 1s. ($16 + 8 + 2 + 1 = 27$)

Method 2: Divide by 2

Let's, for example, convert the decimal number **27** back to binary.

Write out the place values for binary numbers from **right to left**. Start with 1 and multiply by 2 for each additional place until the next place value is greater than the number we are converting:

16 8 4 2 1

*The next place value in the sequence would be 32, which is greater than 27, so we can stop at 16.

For each place value from **right to left**, divide the value to convert by 2 (using only whole numbers) and write the remainder in the place.

$27 / 2 = 13$ with a remainder of 1. Write 1 in the 1's place.

16 8 4 2 1
 1

The value to work on is now **13**.

$13 / 2 = 6$ with a remainder of 1. Write 1 in the 2's place.

16 8 4 2 1
 1 1

The value to work on is now **6**.

$6 / 2 = 3$ with a remainder of 0. Write 0 in the 4's place.

16 8 4 2 1
 0 1 1

The value to work on is now **3**.

$3 / 2 = 1$ with a remainder of 1. Write 1 in the 8's place.

16 8 4 2 1
 1 0 1 1

The value to work on is now **1**.

$1 / 2 = 0$ with a remainder of 1. Write 1 in the 16's place.

16 8 4 2 1
1 1 0 1 1

The value to work on is now **0**. Stop

To check your work, you can use the previous method to add up the values represented with 1s. ($16 + 8 + 2 + 1 = 27$)

Converting Decimal to Binary, Two Methods

Method 1: Subtraction (Left to Right)

Value to Convert	Action	Binary Places					Remaining Value
27	$27 \geq 16$	16	8	4	2	1	$27 - 16 = 11$
	Write 1 in the 16s place.	1					
11	$11 \geq 8$	16	8	4	2	1	$11 - 8 = 3$
	Write 1 in the 8s place.	1	1				
3	$3 < 4$	16	8	4	2	1	Remaining value stays the same. 3
	Write 0 in the 4s place.	1	1	0			
3	$3 \geq 2$	16	8	4	2	1	$3 - 2 = 1$
	Write 1 in the 2 place	1	1	0	1		
1	$1 \geq 1$	16	8	4	2	1	$1 - 1 = 0$
	Write 1 in the 1 place.	1	1	0	1	1	STOP

Result: $27_{10} = 11011_2$

Method 2: Divide by 2 (Right to Left)

Value to Convert	Action	Binary Places					Remaining Value
27	27 / 2 = 13 , R 1 Write 1 in the 1s place. Remaining value = 13.	16	8	4	2	1	13
						1	
13	13 / 2 = 6 , R 1 Write 1 in the 2s place. Remaining value = 6.	16	8	4	2	1	6
					1	1	
6	6 / 2 = 3 , R 0 Write 0 in the 4s place. Remaining value = 3.	16	8	4	2	1	3
				0	1	1	
3	3 / 2 = 1 , R 1 Write 1 in the 8s place. Remaining value = 1.	16	8	4	2	1	1
			1	0	1	1	
1	1 / 2 = 0 , R 1 Write 1 in the 16s place. Remaining value = 0. STOP.	16	8	4	2	1	0
		1	1	0	1	1	

Result: $27_{10} = 11011_2$

Table 4.5 – Converting Decimal to Binary, Two Methods

Hexadecimal

If we write out a large number as a binary number, it can be quite long. Hexadecimal (hex) is a base-16 number system that is used because every four binary digits can be represented by one hexadecimal digit.

In binary, base 2, there are 2 digits:

0 and 1

In decimal, base 10, there are 10 digits:

0, 1, 2, 3, 4, 5, 6, 7, 8, and 9

In hexadecimal, base 16, there are 16 digits:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F

The table below shows the corresponding values of decimal, binary, and hexadecimal:

Decimal, Binary, and Hexadecimal Values

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Table 4.6 – Decimal, Binary, and Hexadecimal Values

Technicians use **hexadecimal (base-16)** because it provides a shorter, easier-to-read way to represent binary numbers. Since each hexadecimal digit represents exactly four binary digits (bits), long strings of 0s and 1s can be written much more compactly. For example, the binary number **11111111** can be written as **FF** in hexadecimal. This makes it easier to read memory addresses, identify error codes, debug software, and work with computer hardware. Hexadecimal is also commonly used to represent colors in web design, where values such as **#FF0000** represent the color red.

How Binary/Hexadecimal is used to store Data

RGB Colors

Colors on a computer screen are represented by combining three colors **Red**, **Green**, and **Blue**.

Each dot on a screen is called a **pixel** (picture element). Each pixel is a combination of 3 lights: red, green and blue. The combination of these three colors is called the RGB color.

The value of each of the three colors can run from **0** (off) to **255** (full blast). When all of the colors are off (0), there is no light: the color is black. When all 3 lights are on full blast (255), the color is white.

All colors of the spectrum are made by combining red, green, and blue.

Red and Green make Yellow.

Green and Blue make Cyan.

Red and Blue make Magenta.



Figure 4.2 – RGB Colors

The decimal values of the colors are usually shown in hexadecimal.

In hexadecimal, 255 is FF.

RGB Values

RGB Value (in hex)	Color	Pixels		
		Red	Green	Blue
000000	Black	0%	0%	0%
FF0000	Red	100%	0%	0%
00FF00	Green	0%	100%	0%
0000FF	Blue	0%	0%	100%
FFFF00	Yellow	100%	100%	0%
00FFFF	Cyan	0%	100%	100%
FF00FF	Magenta	100%	0%	100%
FFFFFF	White	100%	100%	100%

Table 4.7 – RGB Values

You can experiment with RGB colors at <https://zebra0.com/computer-concepts/data-representation/rgb.php>

Tech Tip

The primary colors used in light, such as on computer screen, work differently than primary colors used in paints or pigments.

In paints or pigments, the primary colors are red, yellow, and blue. All colors are made using combinations of these colors, plus black and white. The secondary colors are orange, green, and purple.

With light, the primary colors are red, green, and blue. White light is made using all three at their brightest, and black is the absence of light with all three colors turned off. The secondary colors are yellow, cyan, and magenta.

How Text Is Stored

Although we see letters, numbers, punctuation, and symbols on a screen, computers store all text as binary numbers. Each character is assigned a unique numeric value using a **character encoding** system. When you type a letter on a keyboard, the computer converts it into its corresponding numeric code, stores that code as binary, and later converts it back into a character when the text is displayed. This process happens so quickly that it appears instantaneous to the user.

One of the earliest character encoding systems was **ASCII (American Standard Code for Information Interchange)**. ASCII assigned a unique code for 128 common characters, including uppercase and lowercase letters, numbers, punctuation marks, and control characters. As computers became more widely used around the world, ASCII was no longer sufficient because it could not represent the thousands of characters used in different languages.

Today, most computers use **Unicode**, a universal character encoding standard that supports virtually every written language, along with mathematical symbols, currency symbols, and emojis. Because of Unicode, people around the world can create, share, and display text consistently across different computers, smartphones, and websites.

ASCII Example (7-bit)		Unicode Example (16-bit)	
Character	Binary	Character	Binary (hex)
A	1000001	A	0041
B	1000010	Ω	03A9
a	1100001	你	4F60
1	0110001	😊	1F642

Figure 4.3 – ASCII and Unicode

How Images Are Stored

Digital images are made up of tiny dots called **pixels** (short for *picture elements*).

Each pixel stores color information using numbers rather than paint or ink. Most color images use the **RGB (Red, Green, Blue)** color model, where different amounts of red, green, and blue light are combined to create millions of possible colors.

The more pixels an image contains, the greater its **resolution**, which usually results in a sharper and more detailed picture. However, higher-resolution images also require more storage space because they contain more pixel data.

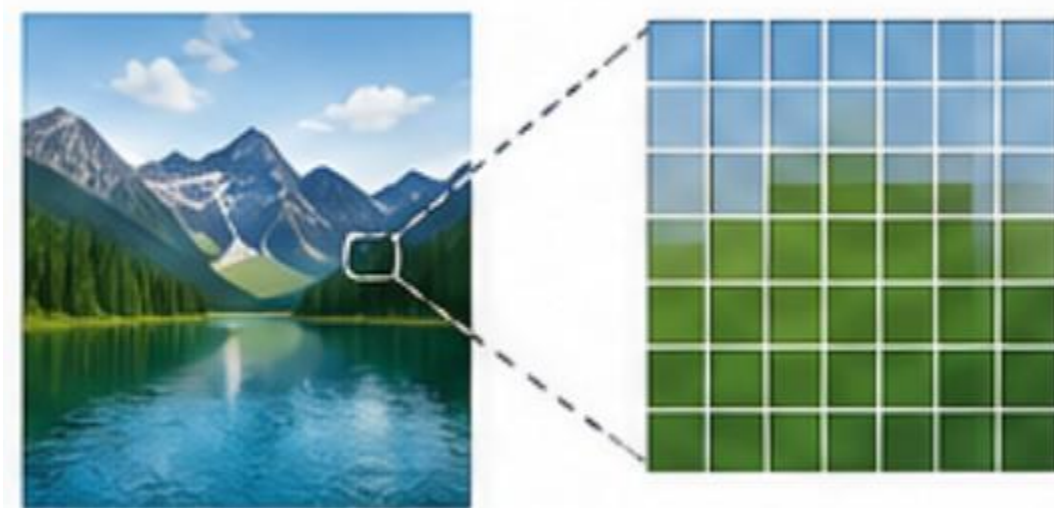


Figure 4.4 – Pixel Resolution

Images are saved in different **file formats**, each designed for specific purposes.

JPEG (.jpg) files use compression to create smaller file sizes, making them ideal for photographs and sharing images online, although some image quality may be lost.

PNG (.png) files use lossless compression, preserving image quality and supporting transparent backgrounds, making them popular for logos, graphics, and screenshots.

GIF (.gif) files are commonly used for simple graphics and short animations.

SVG (.svg) files store images as mathematical shapes instead of pixels. Because SVG images are vector graphics, they can be enlarged or reduced without losing quality, making them especially useful for illustrations, icons, and company logos.

How Sound Is Stored

Sound travels through the air as continuous waves, but computers store sound as digital data. To do this, a microphone converts sound waves into electrical signals, and an **analog-to-digital converter (ADC)** measures, or **samples**, the sound at regular intervals.

The number of samples taken each second is called the **sample rate**. The amount of information recorded for each sample is called the **bit depth**. Higher sample rates and bit depths produce more accurate, higher-quality recordings, but they also create larger audio files. For example, music on an audio CD is typically recorded at a sample rate of **44.1 kHz** with a **16bit** depth, providing clear, high-quality sound.

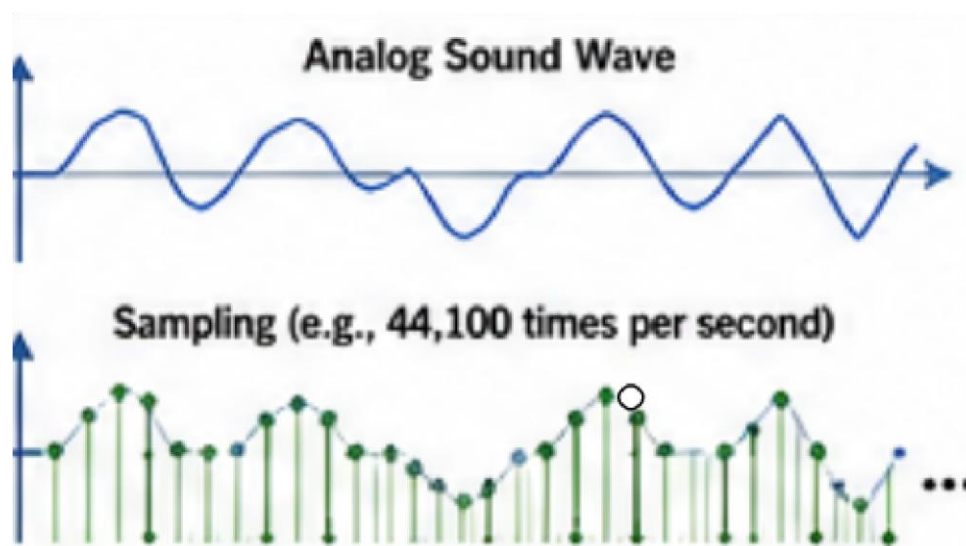


Figure 4.5 – Digital Sampling of an Analog Sound Wave

How Digital Audio Is Sampled

Sound in the real world is continuous and smooth, like a wave that rises and falls over time. All sounds, whether from a plucked guitar string or a person's vocal cords, vibrate at a certain frequency and volume. These sounds can be visualized as a sound wave. The height of the wave (amplitude) represents the volume or loudness, and the distance between waves (wavelength) represents the frequency or pitch.

Analog sound waves are measured (sampled) many times per second and the amplitude at each point is stored as a series of numbers. The **sample rate** is how often the sound is sampled. The **bit depth** is how accurate each sample is.

CD-Quality Audio

The standard for CD-quality audio is 16-bit/44.1 kHz.

44.1 kHz means the audio is sampled 44,100 times per second.

16-bit refers to the precision (or resolution) of each sample.

16 bits (two bytes) allows for a range of 56,536 values.

Key Idea: Each sample is a number that represents the amplitude (loudness) of the sound at a specific moment in time.

Digital Audio Formats

Digital audio can be stored in several different file formats, each designed for a particular purpose.

WAV (.wav) files store uncompressed audio and preserve the highest quality, making them popular for professional recording and editing.

MP3 (.mp3) and **AAC (.aac)** files use **lossy compression** to greatly reduce file size by removing sounds that are less noticeable to the human ear, making them ideal for music streaming and portable devices.

FLAC (.flac) files use **lossless compression**, reducing file size while preserving every detail of the original recording.

Choosing the right audio format involves balancing sound quality, storage space, and how the audio will be used.

How Video Is Stored

Digital video is a sequence of still images, called **frames**, displayed rapidly to create the illusion of motion. Most videos are shown at **24fps**, **30fps**, or **60fps** (frames per second), meaning that dozens of individual images are displayed every second.

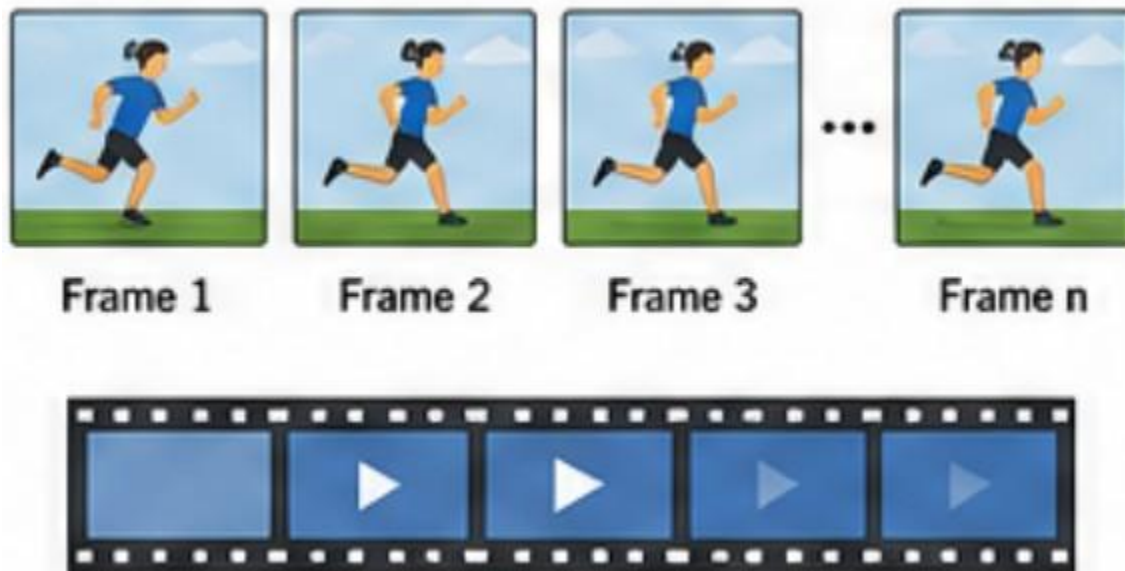


Figure 4.6 – Individual Frames in a Digital Video

Each frame is made up of pixels, just like a digital photograph. Higher resolutions, such as **1080p (Full HD)** or **4K UHD (Ultra HD)**, contain more pixels and produce sharper images, but they also require significantly more storage space.

Because uncompressed video files are extremely large, most digital videos use **compression** to reduce file size while maintaining good visual quality and good sound quality if the video includes synchronized digital audio.

Popular formats such as **MP4 (.mp4)**, **MOV (.mov)**, and **AVI (.avi)** store video using compression techniques that remove redundant information within each frame and between frames. This allows movies, online videos, and video calls to be stored and transmitted much more efficiently.

Streaming services such as YouTube and Netflix also compress video so it can be delivered over the internet without requiring users to download the entire file before watching.

Data Compression

Digital files such as documents, photos, music, and videos can require a large amount of storage space. Larger files can also take longer to download.

Data Transfer

Data transfer speed is how fast information can travel on a network. A business network may have speeds of up to 10 Gbps for internal communications, but home internet is usually much slower, typically 100 Mbps – 2 Gbps, depending on the service plan.

Data transfer is measured in **bits** per second, not **bytes**. Remember, there are 8 bits in every byte, so a 100 Mbps connection can only transfer about 12.5 MB per second.

An uncompressed 1080p video will take up about 1.49 Terabytes of space for every hour of video. That is about 24.8 Gigabytes per minute and 413.9 Megabytes per second of video (3311 Megabits). On a 100 Mbps internet connection, it would take over 30 seconds to download each second of video, making it impossible to stream. An uncompressed 4K video would take even longer.

Data Compression

Data compression is the process of reducing a file's size so it takes up less storage and can be transferred more quickly over the internet. Compression works by finding more efficient ways to store the data or by removing information that is less important. Smaller files save disk space, download faster, and are easier to share through email or cloud storage.

There are two main types of compression:

Lossless compression reduces file size without losing any original information, allowing the file to be restored exactly as it was before compression. ZIP files and PNG images are common examples of lossless compression.

Lossy compression creates even smaller files by permanently removing some data that is considered less noticeable to people. JPEG images, MP3 audio, and MP4 videos all use lossy compression to balance file size with acceptable quality.

Choosing the right type of compression depends on whether preserving every detail or minimizing file size is more important.

Errors in Data Transmission

Whenever data is sent over a network or the internet, there is a small chance that some of the bits can be changed or lost because of electrical interference, weak wireless signals, damaged cables, or other transmission problems. Even changing a single bit from a **0** to a **1** (or vice versa) can cause incorrect information to be received.

To help ensure accuracy, computers and networking equipment use **error detection** techniques, such as parity bits, checksums, and cyclic redundancy checks (CRC), to determine whether the received data matches what was originally sent.

If an error is detected, the receiving device may request that the data be **resent**, or it may use **error correction** techniques to recover the original information without requiring another transmission.

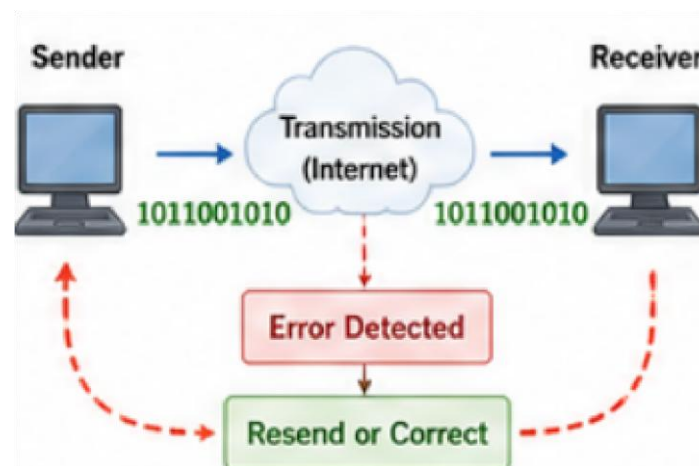


Figure 4.7 – Error Detection

These methods help ensure that files, emails, web pages, videos, and other digital data arrive correctly, even when they travel across long distances and many different networks. Error detection and correction are essential for reliable communication and help keep digital information accurate and complete.

Review

Chapter Summary

All digital information is stored as ones and zeroes using the binary number system. Computers are composed of billions of tiny switches called transistors which have only two reliable states: off and on. Binary is a base-2 number system that recognizes each transistor as a bit with 2 possible values. A byte is a group of eight bits that can store 256 values. Digital storage is measured in Bytes, Kilobytes, Megabytes, Gigabytes, and so on. Hexadecimal is a base-16 number system that represents every four binary digits with one hex digit. Every pixel in a computer image is a combination of three colors: red, green, and blue, stored in hex as RGB colors. Text is stored in binary using Unicode. Sound is digitized by sampling audio at a specified sample rate and bit depth. Digital video is a series of still images played in sequence. Digital media files can be saved in a variety of formats to save space and make them faster to download. Data compression reduces file size through either lossy or lossless compression. Error detection is used to ensure accuracy when transferring files over a network or the internet.

Quick Check

1. Why do computers use binary instead of decimal?
2. What is a bit? What is a byte? How many bits are in one byte?
3. Why are transistors well suited for binary computing?
4. How can you convert a binary number to decimal?
5. How can you convert a decimal number to binary?
6. Why is hexadecimal useful?
7. Explain how RGB colors work.
8. How do computers store text?
9. How do computers store audio?
10. What are the two types of data compression?