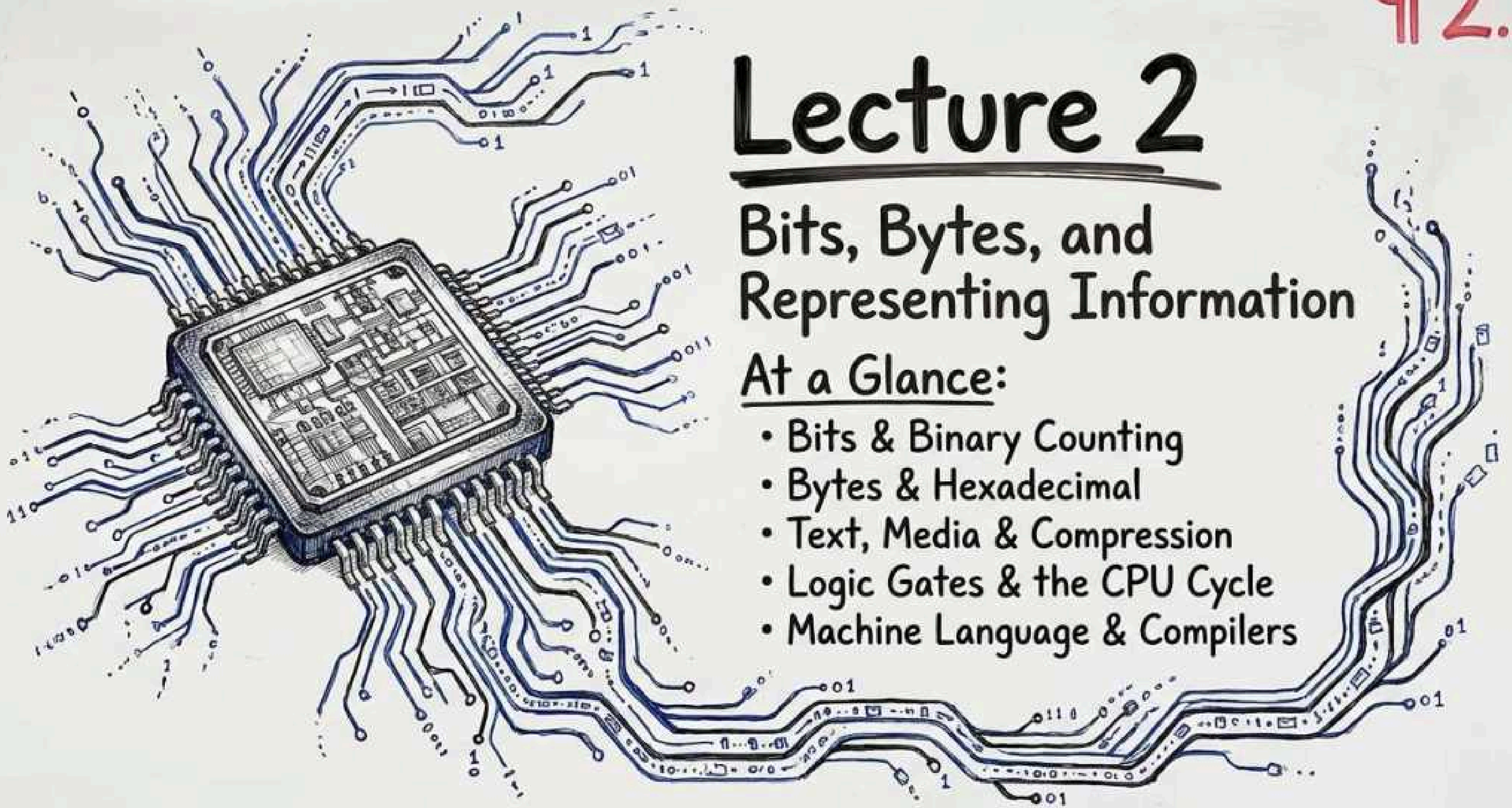
¶2.1



Lecture 2

Bits, Bytes, and Representing Information

At a Glance:

- Bits & Binary Counting
- Bytes & Hexadecimal
- Text, Media & Compression
- Logic Gates & the CPU Cycle
- Machine Language & Compilers

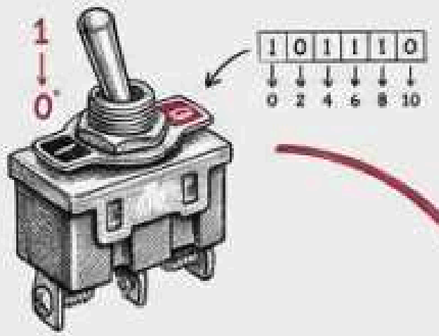
The Journey of Information

¶ 2.2

How a machine of switches holds everything.

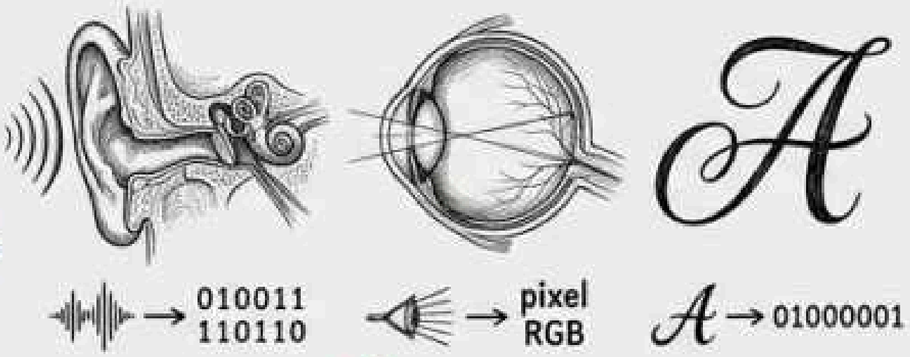
1. The Foundation

- Understanding the bit and binary counting.



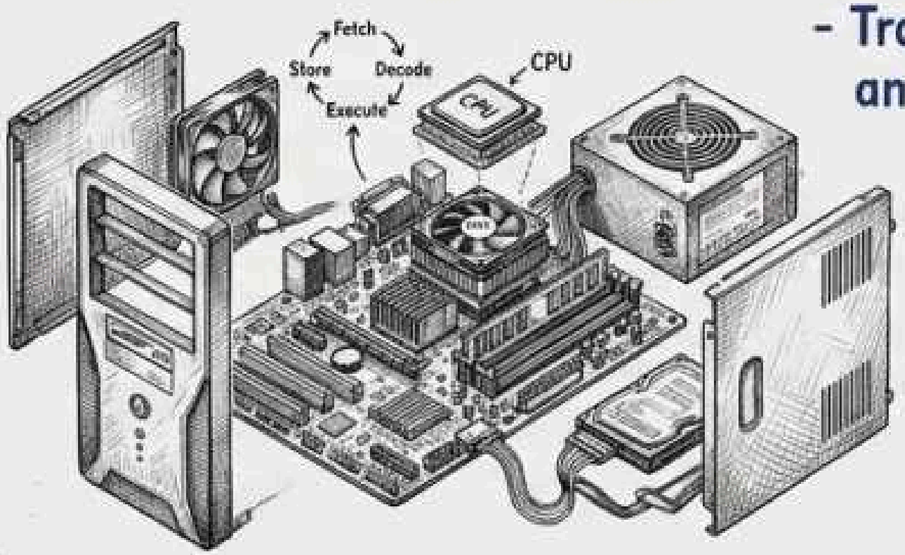
2. The Translation

- Transforming text, pictures, and sound into numbers.



3. The Execution

- Exploring computer architecture and instruction cycles.

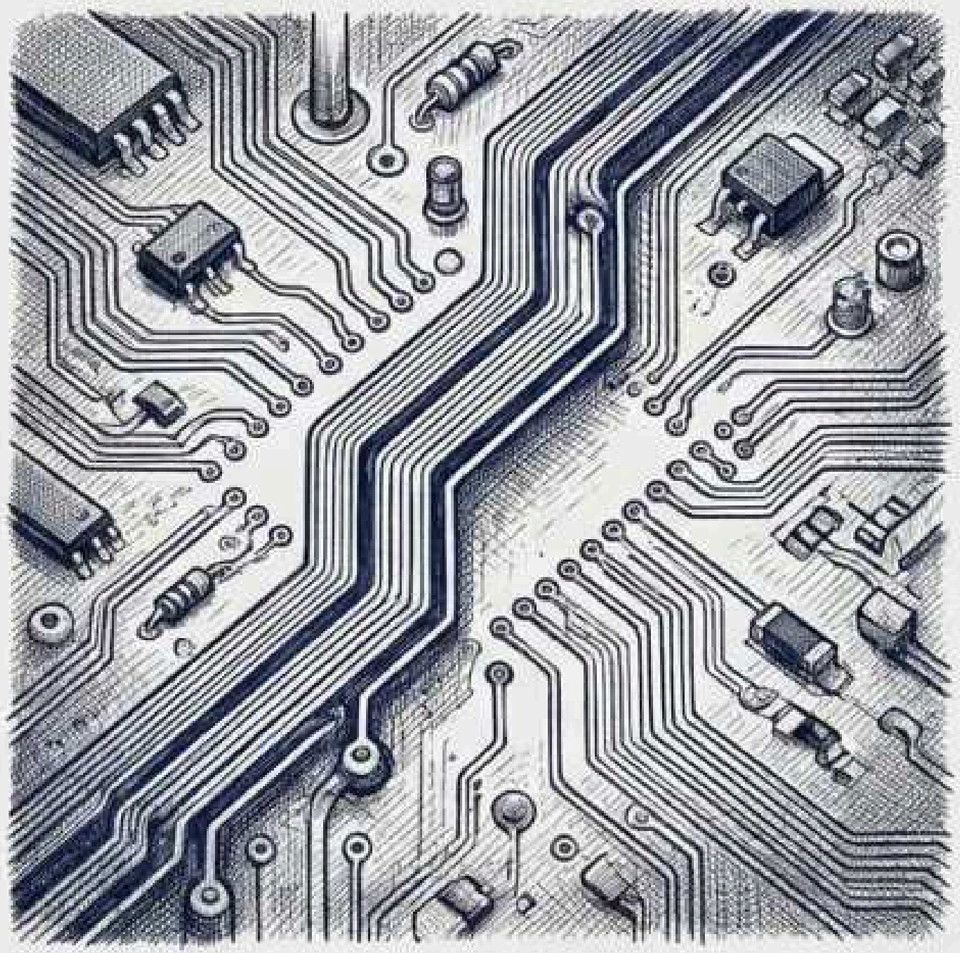


Goal: Demystify the computer from magic into a very fast, very careful machine.

The Bit: The Universal Particle of Data

¶ 2.3

Short for Binary Digit



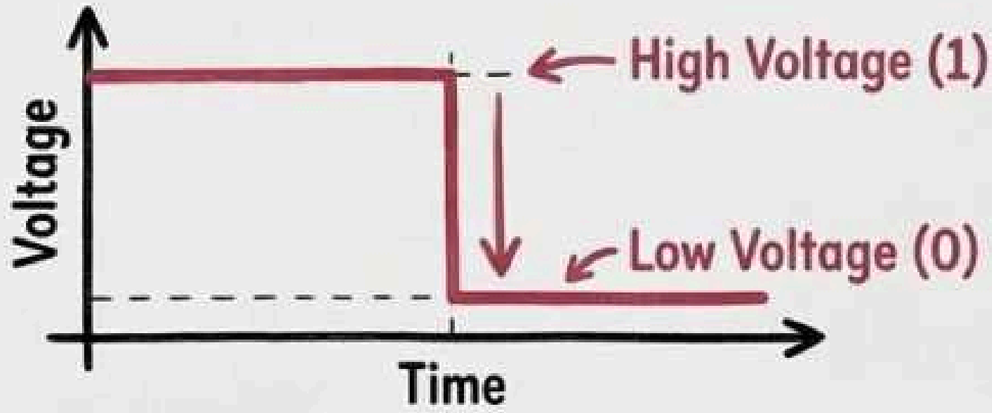
The Physical State:

Electronic circuits operate in just two states:

- ON → High Voltage → 1
- OFF → Low Voltage → 0

The Concept:

Everything a computer holds—letters, photographs, songs, programs—is ultimately a long, continuous pattern of these two states.



Why Only Two States?

¶ 2.4

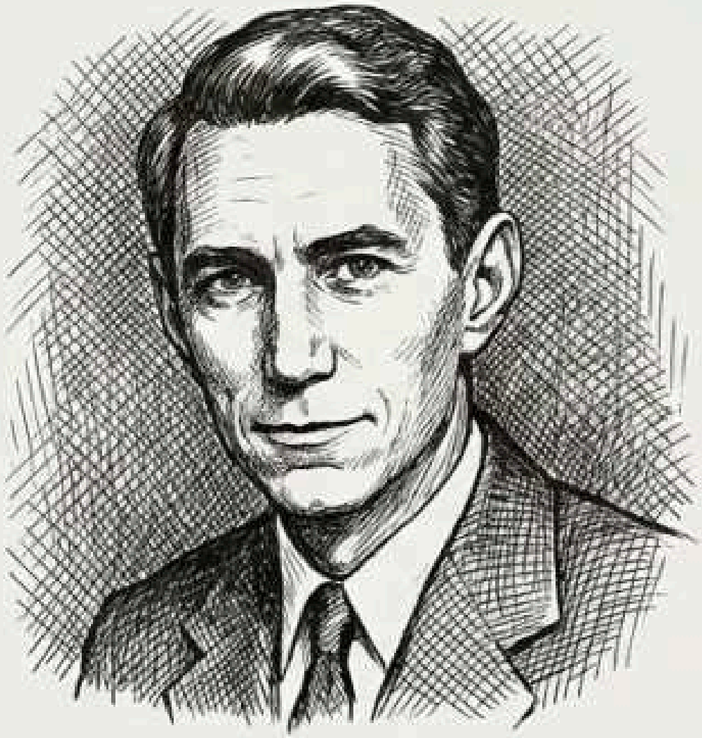
The Problem of Electrical Noise



- Two states are easy to distinguish reliably. Even when electrical noise distorts a signal, a high voltage is clearly distinct from a low voltage.

The Hardware: Transistors

- Modern processors contain billions of transistors—tiny switches controlled by electricity.

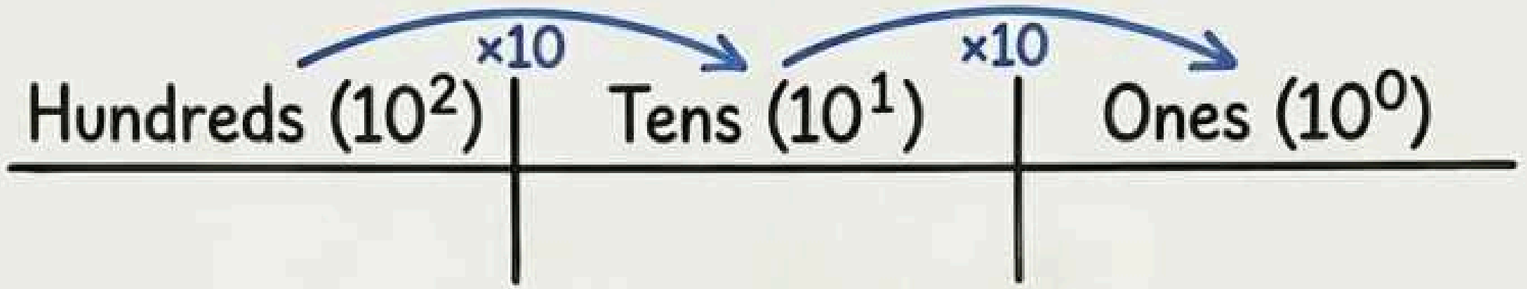


The Pioneer: Claude Shannon

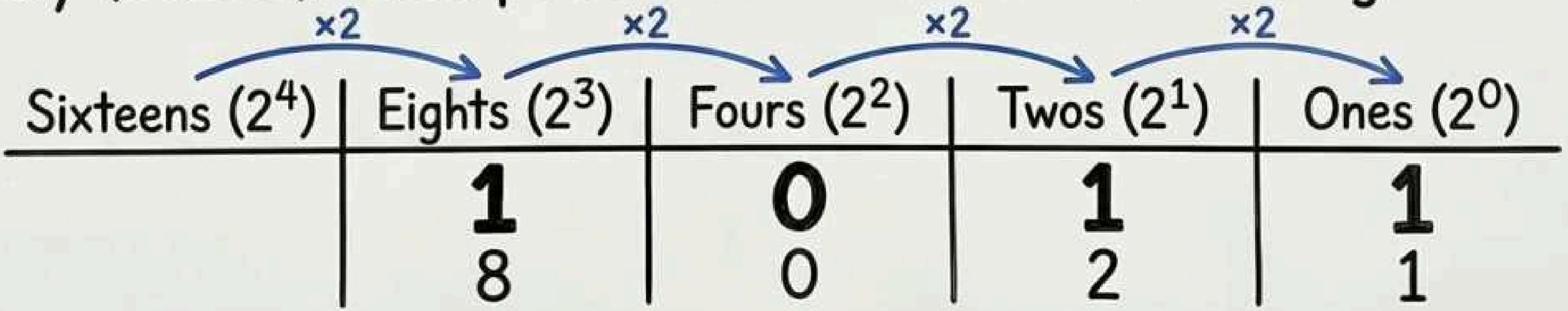
- (1937): Proved switching circuits can carry out logic.
- (1948): Established Information Theory, cementing the bit as the basic unit of information.

Counting in Binary

Decimal (Base 10): Each place is worth 10x the one to its right.



Binary (Base 2): Each place is worth 2x the one to its right.



The Equation: 8 + 0 + 2 + 1 = **11**

Decimal to Binary Conversion

¶ 2.6

Rule: Find the largest place value that fits, subtract, and repeat.

Target Number: 13

1. Largest value that fits is **8**. (13 − 8 = 5) → Place a 1 **1**
2. Largest value that fits is **4**. (5 − 4 = 1) → Place a 1 **1**
3. Does **2** fit in 1? No. → Place a 0 **0**
4. Largest value that fits is 1. (1 − 1 = 0) → Place a 1 **1**

Result: 1101

Capacity Insight:
8 bits = 2^8 = 256 different patterns
(Counts from 0 to 255)

Bytes and Storage Multiples

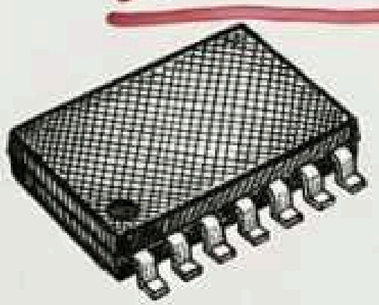
¶ 2.7

1 Byte = 8 Bits (The standard unit for memory & storage)



The Scale of Data:

- Terabyte (TB): ~ 1,000,000,000,000 Bytes
- Gigabyte (GB): ~ 1,000,000,000 Bytes
- Megabyte (MB): ~ 1,000,000 Bytes
- Kilobyte (KB): ~ 1,000 Bytes



Real World Application: A smartphone with 128 GB of storage holds roughly 128,000,000,000 bytes of information.

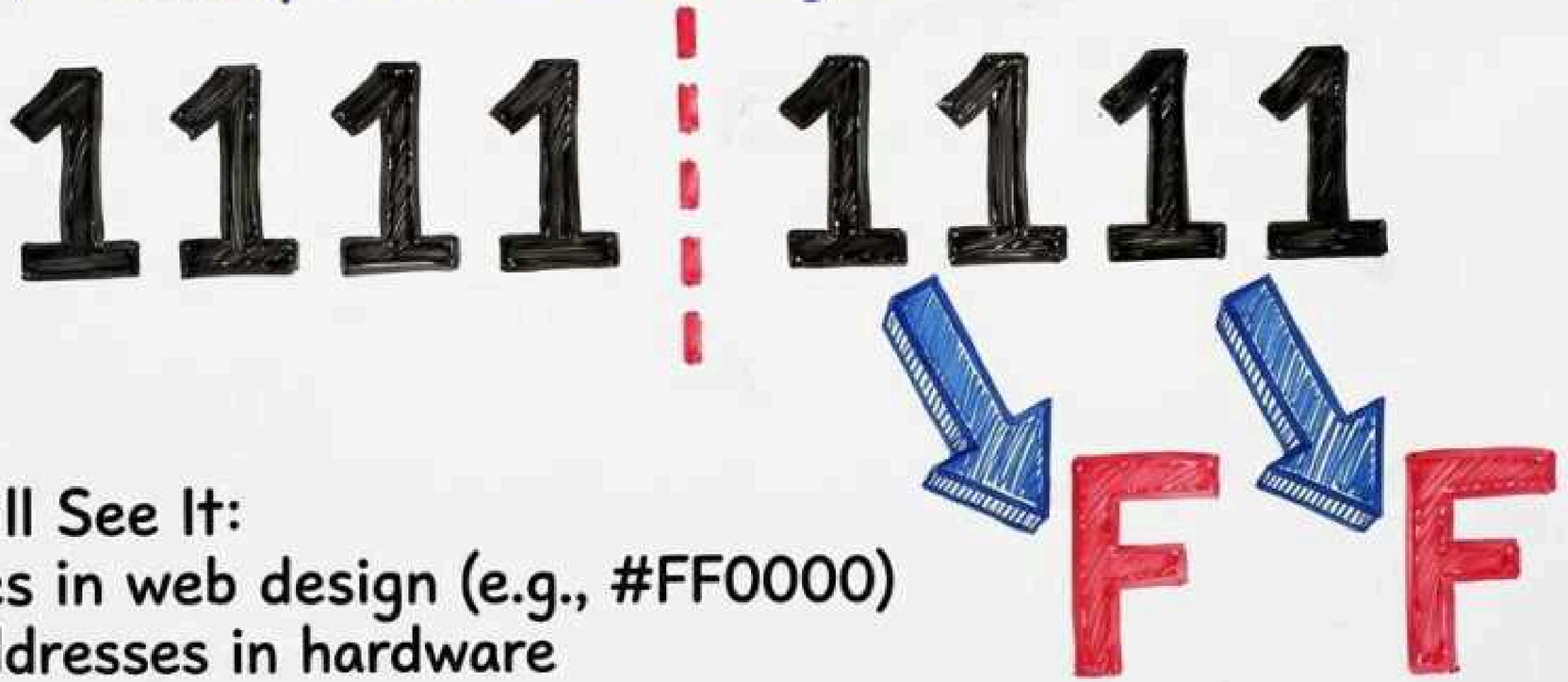
Hexadecimal: A Shorthand for Bits

¶ 2.8

Programmers use Base 16 to write binary compactly.

The Conversion Mechanic:

- 4 bits = Exactly 1 Hexadecimal digit
- 8 bits (1 byte) = Exactly 2 Hexadecimal digits



Where You Will See It:

- Color codes in web design (e.g., #FF0000)
- Memory addresses in hardware

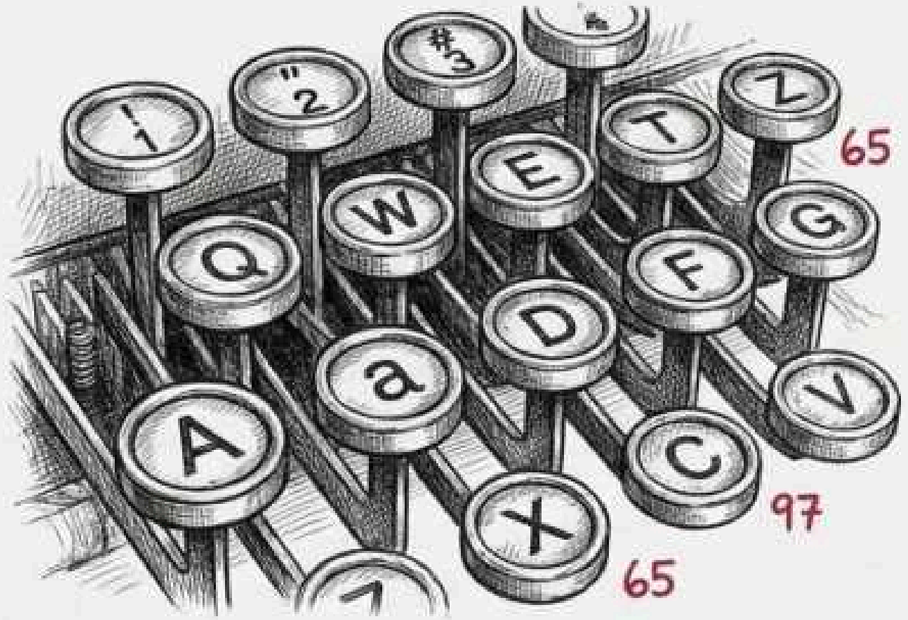
(You do not need to master Base 16 now—just recognize it as a compact way of writing long binary strings).

Storing Text: Characters as Numbers

¶ 2.9

ASCII (The Early Standard)

- Assigned numbers 0–127 to characters.
  - A = 65
  - a = 97
- Limitation: Covered English only.



Unicode & UTF-8 (The Modern Standard)

- Assigns a unique number to every character in every writing system globally.
- Inclusions: Global alphabets, specialized symbols, and emoji.
- UTF-8: The most common encoding method to store Unicode.



Storing Numbers: The Price of Speed

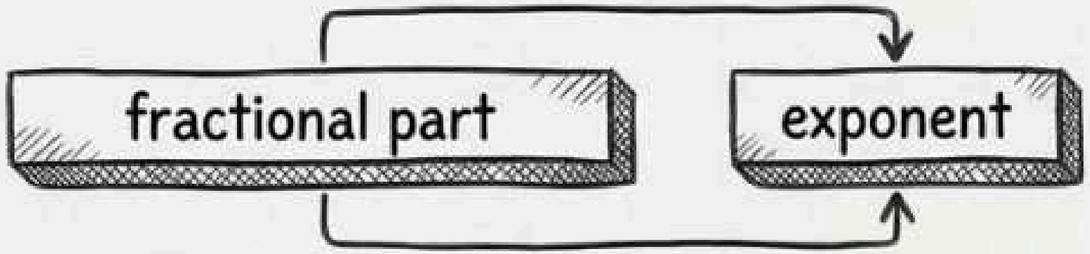
Whole Numbers (Integers):

Stored directly in binary (with a specific trick used for negative values). Exact and reliable.

Fractional Numbers (Floating Point):

Stored in a format resembling scientific notation.

- Advantage: Extremely fast processing.
- Disadvantage: Not perfectly exact.



The Rounding Error Reality:

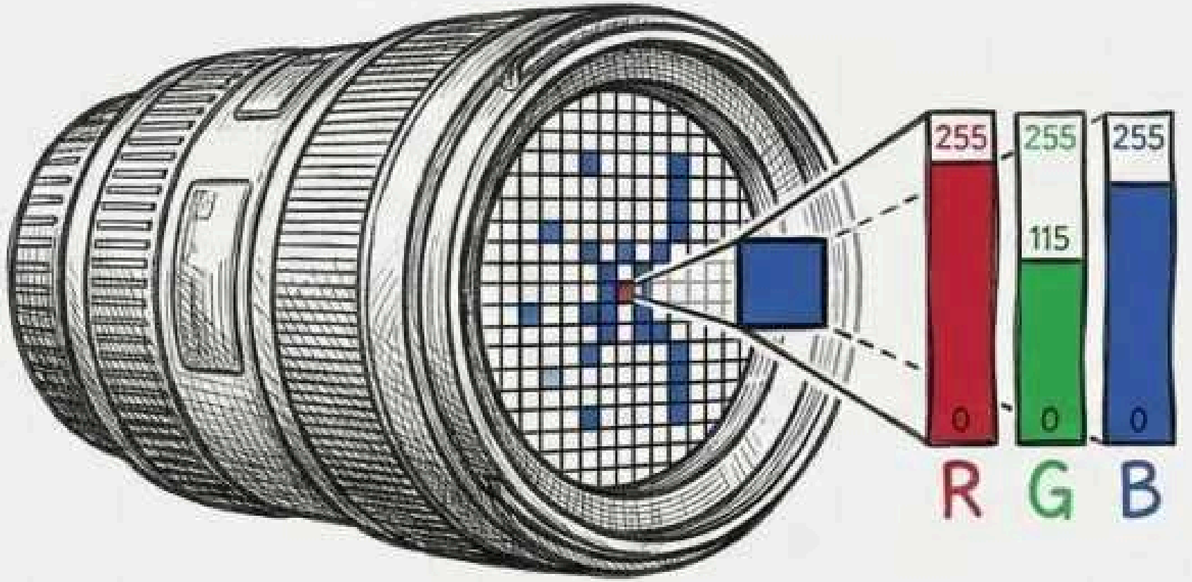
In most programming languages, tiny rounding errors are mathematically normal:

$$0.1 + 0.2 \neq 0.3$$

Digitizing Media: Pictures & Sound

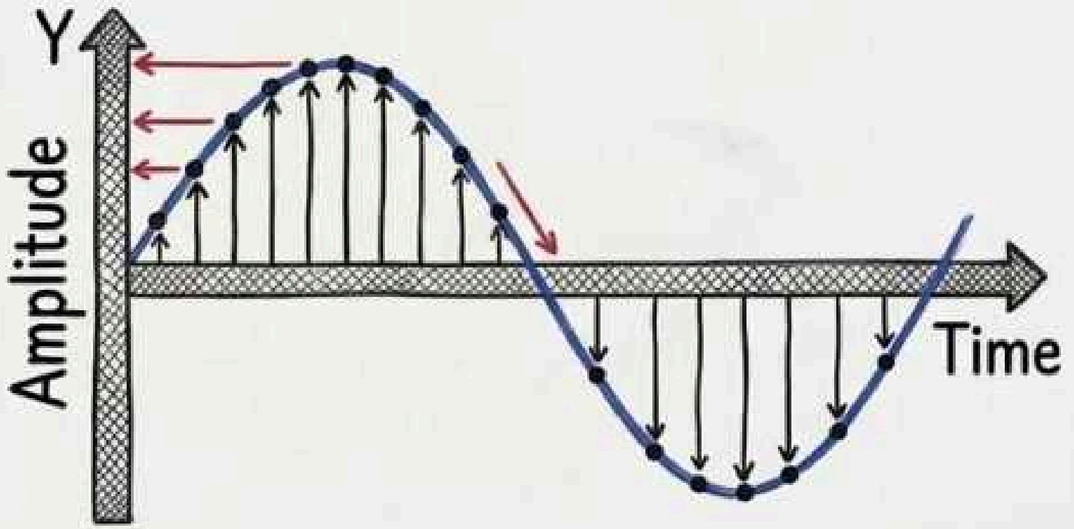
Pictures (Pixels):

- A grid of tiny squares. Each pixel uses 3 numbers (0–255) representing Red, Green, and Blue.
  - 1 pixel = 3 bytes.
  - Example: A 12 million pixel photo requires 36 million bytes (raw).



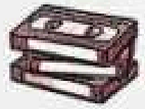
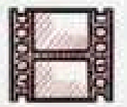
Sound (Waves):

- Stored by measuring the height of an audio wave many times per second.
- Example: A standard Compact Disc captures 44,100 samples every second.



Compression: Shrinking Raw Data

Because raw data is overwhelmingly large.

Type	How it Works	The Trade-off	Common Uses
Lossless	Shrinks data algorithmically and restores it exactly bit-for-bit.	Smaller space saving, perfect fidelity.	.zip files,  text documents,  data backups. 
Lossy	Throws away minor details that human eyes/ears are unlikely to notice.	Massive space saving, permanent small loss of quality.	Photos (JPEG),  Music (MP3),  Video (MP4). 

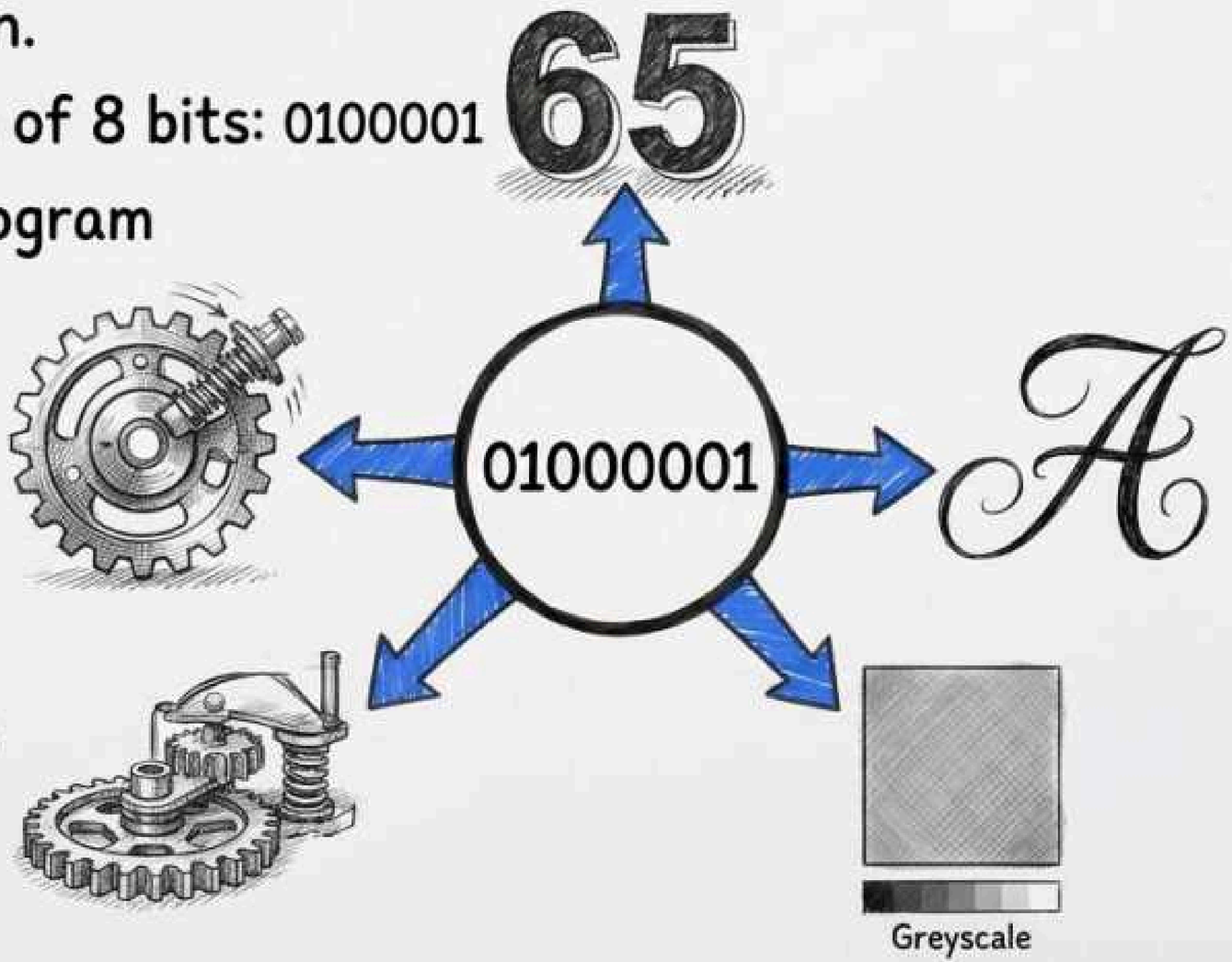
The Key Lesson: Meaning Requires Context

Bits have no meaning on their own.

Consider this exact same pattern of 8 bits: 0100001

Depending entirely on how the program interprets it, this could mean:

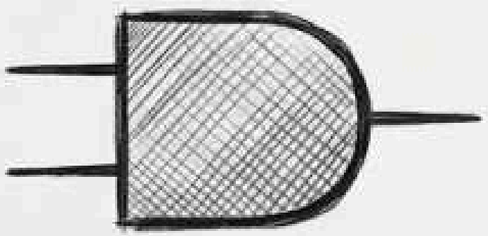
1. The number 65
2. The capital letter A
3. A specific shade of grey in a photograph
4. Part of a machine instruction for the CPU



This requirement for context is why programming languages have types.

Computation: Logic Gates

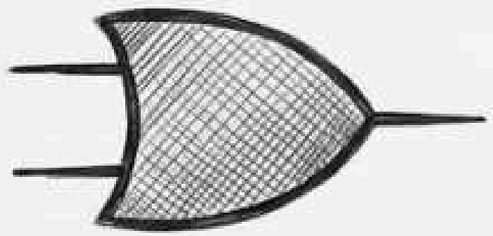
Gates combine bits. Billions of gates wired together create a processor.



In 1	In 2	Out
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate:

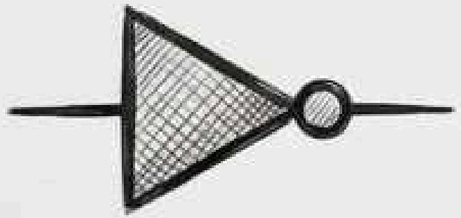
Outputs 1 ONLY when both inputs are 1.



In 1	In 2	Out
0	0	0
0	1	1
1	0	1
1	1	1

OR Gate:

Outputs 1 when at least one input is 1.

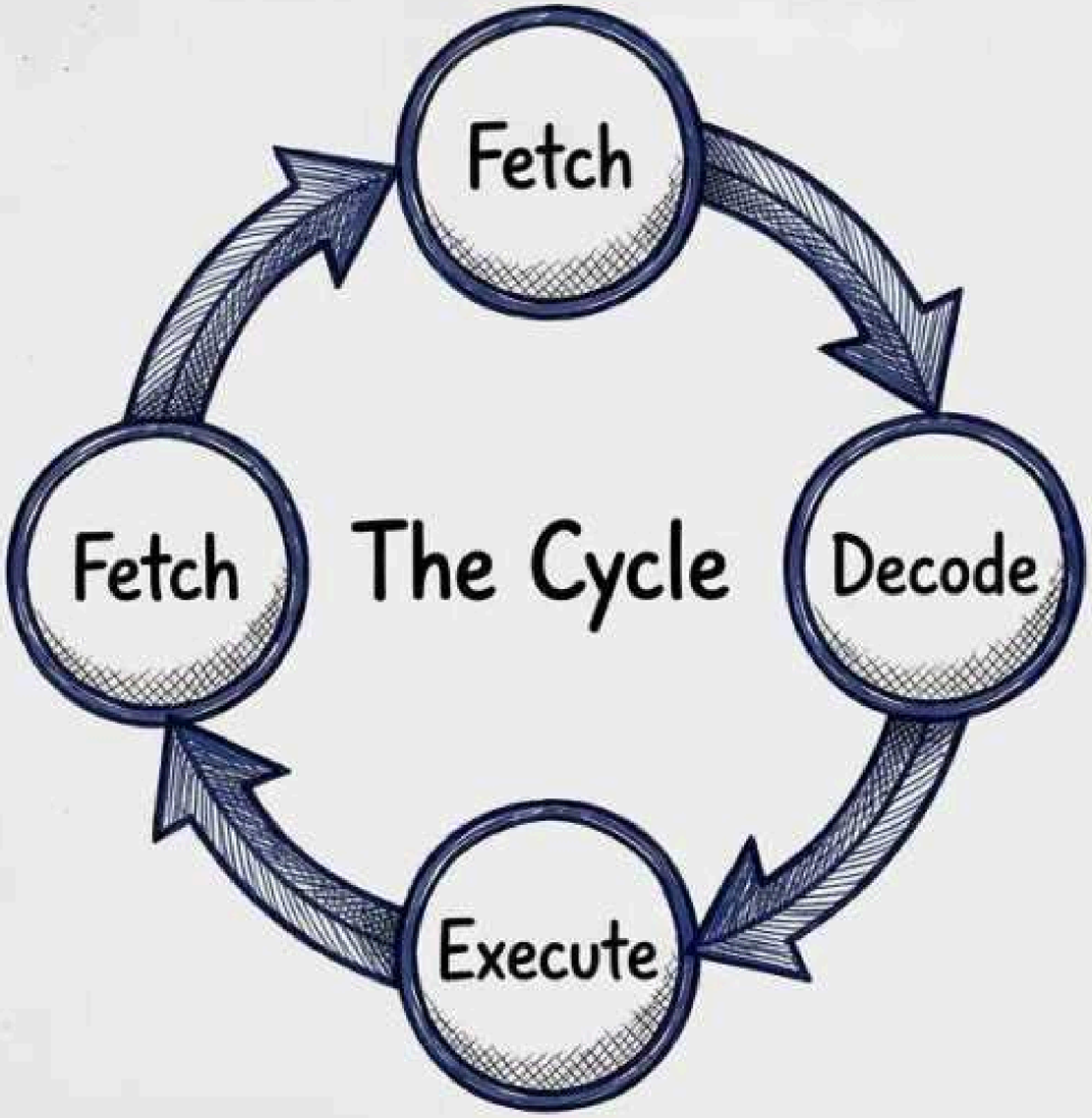


NOT Gate:

Flips its input (1 → 0, or 0 → 1).

In	Out
0	1
1	0

The Result: By wiring these simple rules together, circuits can mathematically add numbers, compare values, and remember states.

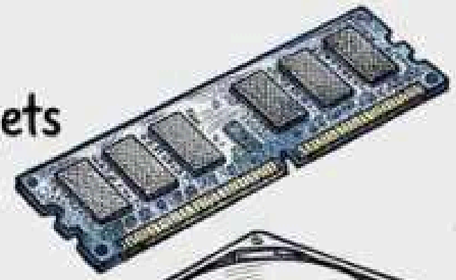


Inside the Machine

The Processor Cycle (Billions of times per second):

1. Fetch: Retrieve an instruction from memory.
2. Decode: Determine what the instruction means.
3. Execute: Carry out the physical logic.

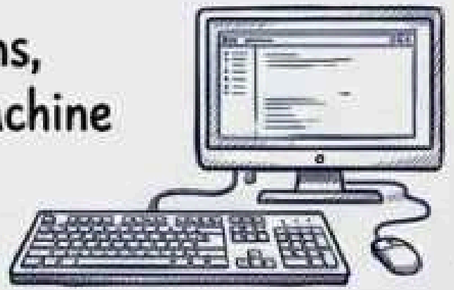
Hardware Architecture:



- RAM (Memory): Fast, but forgets everything when power is lost.



- Storage: Slower, but retains contents permanently.



- I/O Devices: Keyboards, screens, and sensors connecting the machine to the physical world.

Languages & Translation

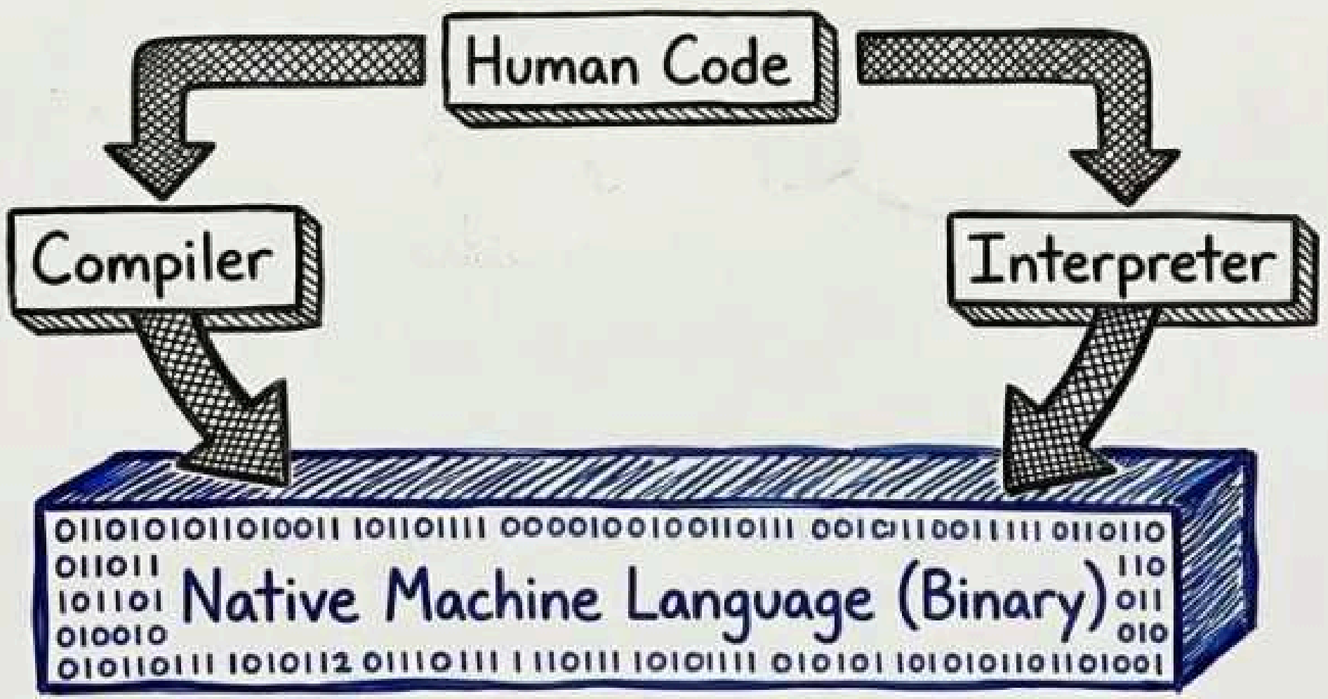
¶ 2.16

A processor only understands its own native Machine Language (numeric commands like add, jump, compare).

Writing raw machine language is slow and error-prone, so we write in human-readable programming languages.

How we translate to the machine:

- Compiler: Translates the entire program into machine language all at once before it runs.
- Interpreter: Translates and runs the code one line at a time, on the fly. (Python normally uses this method).



Lecture Question / Exercise

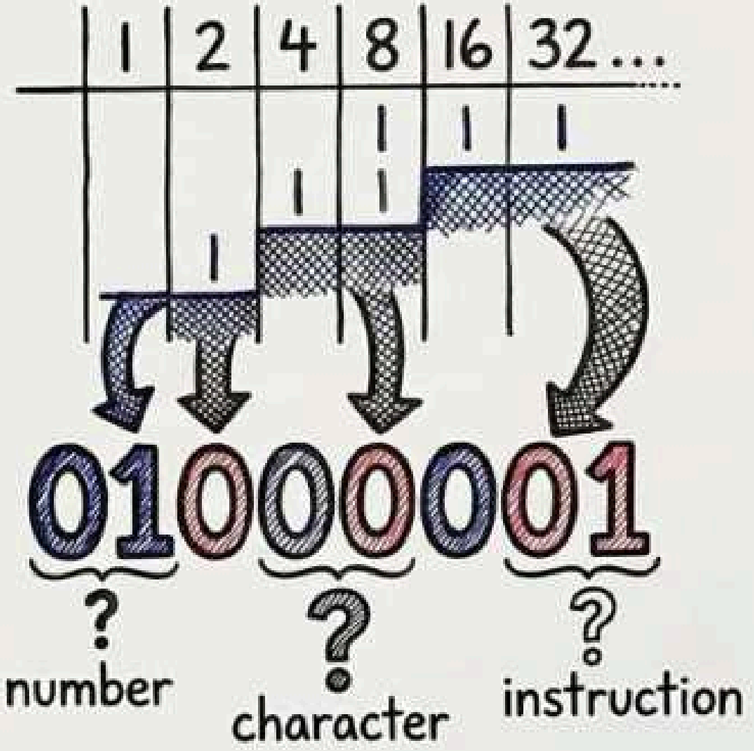
Part 1: Apply Place Values

Choose a small number (such as your age or the day of the month) and convert it to binary using the place value subtraction method.

Part 2: Decode Context

Consider the binary pattern: 01000001

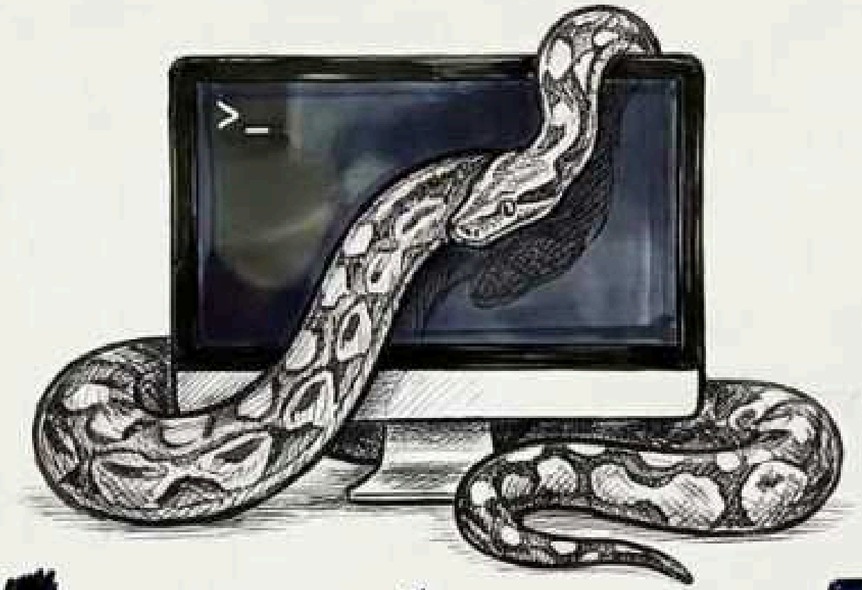
- What could this pattern mean?
- What specific information would you need to know in order to decide its true meaning?



Looking Ahead

Next: Lecture 3

We will take the foundations of bits, data types, and logical execution to begin writing actual programs using the Python language.



Lecture 3