1. Computer Architectures

The Management of the Information

Summary

- The management of the Information
- The management of the Information: Binary Code
- ASCII Code
- Understanding Digital Data Sizes
- Binary Code: Properties
- Converting Binary to Decimal
- Converting Decimal to Binary
- Boolean Algebra
- The Von Neumann Model

The Management of the Information

In the context of computer architectures, **information** refers to data that is **processed**, **stored**, and **transmitted** by **electronic devices**. It encompasses all forms of **digital data** which computers manipulate and manage to perform tasks and operations.

Examples of "information":

- Text document (e.g., DOCX file);
- Drawing (e.g., DXF, DWG files);
- Spreadsheet (e.g., XLSX file);
- Image (e.g., BMP, TIFF files);
- Video (e.g., AVI, MP4 files);
- Audio (e.g., MP3, WAV, files);

... and so on.

The Management of the Information: Binary Code/1

Any information **stored** or **retrieved** by a device (such as desktop computers, laptops, or smartphones) must be represented in a language that the device can understand. This language is known as **Binary Code**.

Binary Code is defined as a **coding system** that uses the binary digits **0** and **1** to represent letters, digits, or other characters in a computer or other electronic devices, such as smartphones and tablets.

More specifically, a **BIT (Binary digit)** is the **basic unit of information** used by devices to manage all types of data. A single BIT can exist in one of two states: 0 or 1.

NOTE: With a single BIT, only two elements can be represented.

For example:

1 = "A"

0 = "B"

What about more complex information? BITS COMBINATION.

The Management of the Information: Binary Code/2

Bits and Combinations:

- One bit can have 2 states: 0 or 1.
- With **N bits** we can create $\mathbf{2^N}$ **distinct combinations**.

Examples:

- **1 bit:** $2^1 = 2$ combinations (0, 1)
- **2 bits:** $2^2 = 4$ combinations (00, 01, 10, 11)
- **3 bits:** $2^3 = 8$ combinations (000, 001, 010, 011, 100, 101, 110, 111)

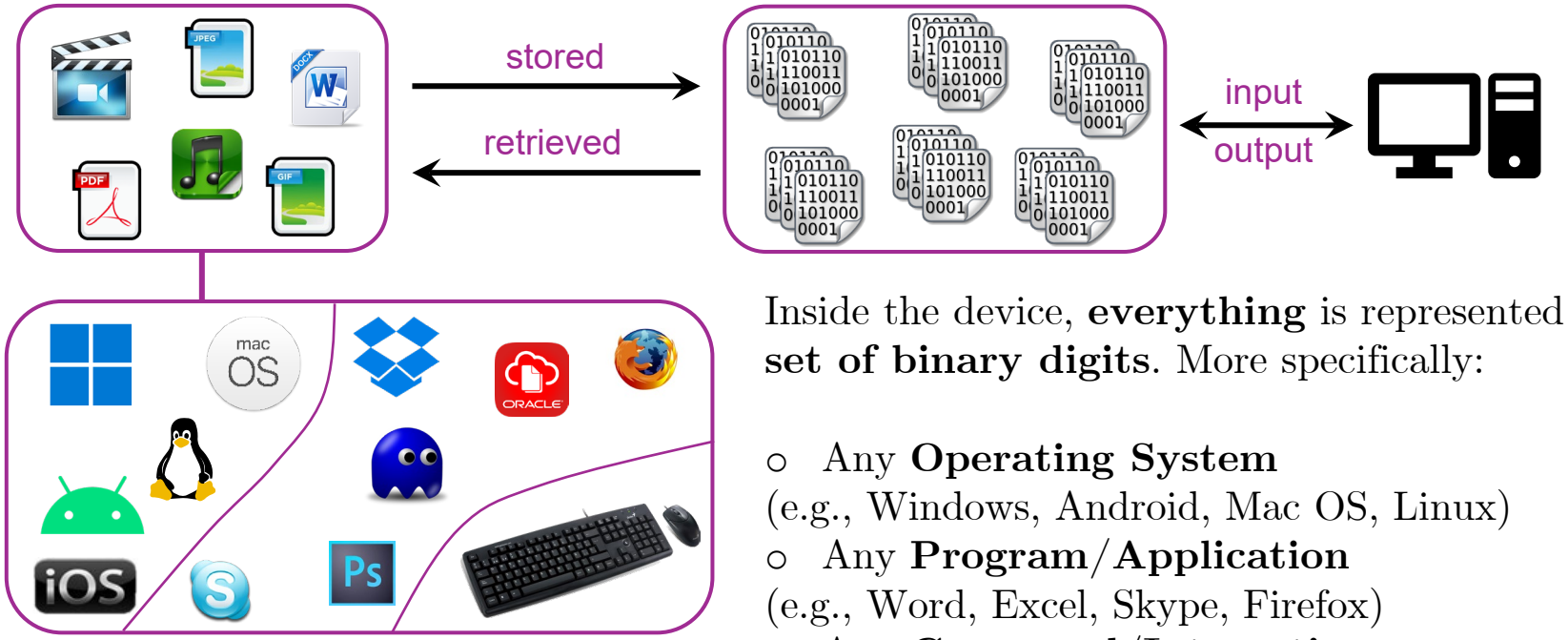
Range of Values:

With **N bits**, the representable values range from **0** to $\mathbf{2^N - 1}$.

Examples:

- **1 bit:** $2^1 - 1 = 1$, representable values in range [0,1]
- **2 bits:** $2^2 - 1 = 3$, representable values in range [0,3]
- **3 bits:** $2^3 - 1 = 7$, representable values in range [0,7]

The Management of the Information: Binary Code/3



Inside the device, **everything** is represented by a **set of binary digits**. More specifically:

- Any **Operating System**
(e.g., Windows, Android, Mac OS, Linux)
- Any **Program/Application**
(e.g., Word, Excel, Skype, Firefox)
- Any **Command/Interaction**
(e.g., Keyboard, Mouse, Touchpad)

ASCII Code

ASCII: The American Standard Code for Information Interchange typically uses **7 bits** to represent each character, which allows for **128 unique character** codes.

Extended ASCII: ASCII is sometimes extended to **8 bits**, which enables an **additional 128 characters**, bringing the total to 256. This extension isn't standardized in the same way as the original 7-bit ASCII, and there are various extended ASCII sets, like ISO 8859-1 or Windows-1252, which include characters for **specific languages** or **graphical symbols**.

Types of Characters:

- **Alphanumeric Characters:** This includes all uppercase and lowercase English letters and numbers (0-9).
- **Symbols:** ASCII includes a set of common punctuation symbols and other miscellaneous symbols like @, #, $, etc.
- **Control Characters:** These are non-printable characters that control the flow of text or its processing in some way. Examples include TAB (horizontal tab), LF (line feed), CR (carriage return), and BEL (bell/alert).

ASCII Code - Examples

Standard ASCII Table (7-bit)

Decimal	Hex	Binary	Character	Description
32	20	00100000	(space)	Space
48	30	00110000	0	Digit Zero
65	41	01000001	A	Uppercase A
97	61	01100001	a	Lowercase a
10	0A	00001010	LF	Line Feed
13	0D	00001101	CR	Carriage Return

Extended ASCII Table (8-bit)

Decimal	Hex	Binary	Character	Description
128	80	10000000	Ç	Latin Capital Letter C with Cedilla
165	A5	10100101	¥	Yen Sign
178	B2	10110010	²	Superscript Two
225	E1	11100001	á	Lowercase a with acute
245	F5	11110101	õ	Lowercase o with tilde

Understanding Digital Data Sizes/1

This table provides a comparison of data sizes using different prefixes, their decimal sizes, binary approximations, and practical examples.

- A group of **8 bits** is named **Byte**;

Abbr.	Prefix-Byte	Decimal Size	Size in Thousands	Binary Approximation	An Example
K	Kilo-	10^3	$1,000^1$	$1,024 = 2^{10}$	A text file (e.g., 1 KB)
M	Mega-	10^6	$1,000^2$	$1,024^2 = 2^{20}$	A song (e.g., MP3 file, 4 MB)
G	Giga-	10^9	$1,000^3$	$1,024^3 = 2^{30}$	A HD film (e.g., MP4 file, 1.5 GB)
T	Tera-	10^{12}	$1,000^4$	$1,024^4 = 2^{40}$	A large backup drive (e.g., 1 TB of data)
P	Peta-	10^{15}	$1,000^5$	$1,024^5 = 2^{50}$	Storage of large-scale research data (e.g., climate data, 1 PB)
E	Exa-	10^{18}	$1,000^6$	$1,024^6 = 2^{60}$	Entire data usage of a major tech company (e.g., Google, 1 EB)

Understanding Digital Data Sizes/2

Inside each device, **information** is represented by a fixed set of bytes:

- 16 bits (**2 bytes**)
- 32 bits (**4 bytes**)
- 64 bits (**8 bytes**)

The number of bytes indicates the "power" of a device. The greater the number of bytes:

- The greater the device's ability **to perform complex computations**
- The greater the device's capacity **to manage large amounts of information**
- The greater the device's capability **to process complex instructions**

Binary Code: Properties

Like every numerical system, binary code has its own **intrinsic properties**. One key property, shared with other **positional systems**, is the ability to convert numbers from one numerical base to another.

Furthermore, the **binary system** supports the **four basic operations**:

- Addition;
- Subtraction;
- Multiplication;
- Division.

NOTE:

A number represented using **two digits** is called a **binary number** (**base two**).

A number represented using **ten digits** is called a **decimal number** (**base ten**).

Converting Decimal to Binary

To convert a decimal number (base 10) to a binary number (base 2), follow these steps:

1. **Divide** the number by 2.
2. **Record** the remainder (0 or 1). This will be the least significant bit (LSB).
3. **Continue dividing** the quotient obtained in the previous step by 2, **recording** the remainders.
4. **Repeat** until the quotient is 0.
5. **Write the remainders in reverse order**: from the last obtained to the first.

   This represents the number in binary.

Converting 13_{10} to Binary:

$13 \div 2 = 6$, remainder $= 1$
$6 \div 2 = 3$, remainder $= 0$
$3 \div 2 = 1$, remainder $= 1$
$1 \div 2 = 0$, remainder $= 1$

Result in binary: 1101_2

Converting Binary to Decimal

To convert a binary number (base 2) to a decimal number (base 10), follow these steps:

1. **Write down** the binary number.
2. **List the powers of 2** from right to left, starting with 2^0 under the rightmost bit.
3. **Multiply each bit** by the corresponding power of 2.
4. **Sum** the results to get the decimal number.

Converting 1101_2 to Decimal:

$$\begin{aligned} 1101 &= 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 8 + 4 + 0 + 1 = 13 \end{aligned}$$

Result in decimal: 13_{10}

Binary Addition/1

1. **Take** the two binary numbers you want to add.

   NOTE: Make sure both numbers are of the same length.

   If not, add leading zeros to the shorter number until they are of the same length.

2. **Start adding** the bits from the rightmost side (least significant bit) of the two binary numbers, following these rules:

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

3. **Proceed by adding** the successive bits and considering any carries from the previous columns. If after adding the last bit there remains a carry, write it in the leftmost column of the result.

Binary Addition/2

Example 1 (1011 + 1101):

```
  1011
+ 1101
----------
 11000 (Result: carry applied)
```

Step-by-step addition with carry:
1 + 1 = 0 (carry 1)
1 + 0 + 1 (carry) = 0 (carry 1)
0 + 1 + 1 (carry) = 0 (carry 1)
1 + 1 + 1 (carry) = 1 (carry 1 to a new column)

Example 2 (1011 + 101):

```
  1011
+ 0101
----------
 10000 (Result: carry applied)
```

Step-by-step addition with carry:
1 + 1 = 0 (carry 1)
1 + 0 + 1 (carry) = 0 (carry 1)
0 + 1 + 1 (carry) = 0 (carry 1)
1 + 0 + 1 (carry) = 0 (carry 1 to a new column)

Binary Subtraction/1

1. **Take** the two binary numbers you want to subtract.

   NOTE: Make sure both numbers are of the same length.

   If not, add leading zeros to the shorter number until they are of the same length.

2. **Start subtracting** the bits from the rightmost side (least significant bit) of the two binary numbers, following these rules:

Case	A - B	Difference	Borrow
1	0 - 0	0	0
2	1 - 0	1	0
3	1 - 1	0	0
4	0 - 1	1	1

3. **Proceed by subtracting** the successive bits and considering any borrows from the previous columns. If after subtracting the last bit there remains a borrow, write it in the leftmost column of the result.

Binary Subtraction/2

Example 1 (1110 – 1001):

```
  1110
- 1001
 --------
  0101 (Result: borrow applied)
```

Step-by-step subtraction with borrow:
0 - 1 = 1 (borrow 1)
1 - 0 - 1 (borrow) = 0
1 - 0 = 1
1 - 1 = 0

Example 2 (1011 - 101):

```
  1011
-  0101
-----------
  0110 (Result: borrow applied)
```

Step-by-step subtraction with borrow:
1 – 1 = 0
1 – 0 = 1
0 - 1 = 1 (borrow 1)
1 - 0 - 1 (borrow) = 0

Boolean Algebra/1

Boolean Algebra is a branch of algebra that deals with **true** or **false** values, typically denoted as **1** and **0**, respectively. It is fundamental in the field of **computer science** and **digital electronics** because it is used to design and analyze the **behavior of digital circuits** and **logic gates**.

Key Concepts:

- **Binary Variables:** Represented as 1 (true) and 0 (false).
- **Logical Operations:**

  AND $(\cdot)$: Yields true if both operands are true $(1 \cdot 1 = 1)$.

  OR $(+)$: Yields true if at least one operand is true $(1 + 0 = 1)$.

  NOT $(\neg)$: Yields the inverse of the operand $(\neg 1 = 0)$.

Boolean Algebra/2

TRUTH TABLES

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	NOT A
0	1
1	0

Boolean Algebra/3

- **Commutative:**

A OR B = B OR A

A AND B = B AND A

- **Distributive:**

A OR (B AND C) = (A OR B) AND (A OR C)

A AND (B OR C) = (A AND B) OR (A AND C)

- **De Morgan's Law:**

NOT (A AND B) = (NOT A) OR (NOT B)

NOT (A OR B) = (NOT A) AND (NOT B)

- **Precedence Rules:**

- NOT has the highest precedence;

- followed by AND;

- lastly, OR.

To alter the precedence, use parentheses.

Boolean Algebra/4

Example:

A AND NOT (B OR C)

A	B	C	A AND NOT (B OR C)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

The Von Neumann Model/1

The **Von Neumann Architecture** is a **foundational concept for modern computing systems**, characterized by its structure where data and program code are stored in the same memory space.

The typical **Von Neumann Model** consists of 4 main components:

- **Memory:** Stores both data and instructions.
- **Control Processing Unit:**
    - **Arithmetic Logic Unit (ALU):** Executes all arithmetic and logical operations.
    - **Control Unit (CU):** Decodes program instructions and controls the other components based on these instructions.
- **Input and Output (I/O):** Handle data exchange between the computer system and the external environment.
- **Bus:** Provides a communication system that transfers data between components.

The Von Neumann Model/2