Digital Logic & Design

Encoding Methods Compiled Notes

Instructor:
Wali Ullah Shinwari

Assistant Professor at department of Information Technology, faculty of computer Science

w.shinwari@kardan.edu.af

Digital Logic & Design

Encoding Scheme/methods/algorithms:

An **encoding method** refers to procedures or systems or algorithms for converting data, such as characters or symbols, into a specific format that can be understood by computers and transmitted efficiently between systems. It defines how information is represented in binary (0s and 1s) so that it can be processed, stored, and exchanged by digital systems like computers, communication devices, and data storage systems.

Purpose of Encoding Methods

The main purpose of an encoding method is to allow data (especially text) to be represented in a standardized format that can be:

1. **Stored and Processed**: Computers work with binary data, so encoding methods transform text and symbols (which humans use) into a binary format that machines can handle.
2. **Transmitted**: Encoding allows information to be transmitted between different systems, such as between computers over a network or between a server and a client. A standardized encoding ensures that both systems can interpret the data consistently.
3. **Efficient Storage**: Encoding methods help ensure that data is stored efficiently in memory or on disk. Different encoding schemes vary in how much memory they use to represent the same data.
4. **Interoperability**: By encoding data into a common format, different devices, software, and platforms can understand the same data, ensuring compatibility and interoperability across systems and applications.

Components of Encoding Methods

- **Code Size**: This refers to the number of bits (binary digits) used to represent each character or symbol. For example, ASCII uses 7 bits per character, while UTF-8 can use 1 to 4 bytes per character.
- **Character Set**: An encoding method defines a set of characters (alphabet, numerals, symbols) that it can represent. Some encoding schemes handle only a limited set of characters (e.g., ASCII), while others support many languages and symbols (e.g., Unicode encodings like UTF-8).
- **Mapping**: Encoding provides a unique binary code for each character in the set. For instance, the letter 'A' might be represented as `01000001` in ASCII.

Why Encoding Methods are Important

1. **Text Representation**: Textual information (letters, numbers, symbols) in different languages needs to be represented consistently in digital form. Encoding ensures that text written in English, Chinese, Arabic, or any other language can be converted into binary and displayed correctly on digital devices.

2. **Data Compression**: Encoding helps in compressing data to save storage space. Some methods focus on compact representation, using fewer bits for common characters and more for less common ones.
3. **Compatibility**: By using standardized encoding methods, data can be shared across different platforms, operating systems, and applications without the risk of misinterpretation. For example, web pages' use UTF-8 encoding to ensure text displays correctly on browsers worldwide.
4. **Security**: Encoding methods are often employed in cybersecurity to transform data into encoded forms (though not to be confused with encryption) that are machine-readable but human-incomprehensible.

Common Uses of Encoding Methods

- **Web Development**: UTF-8 is the standard encoding used in HTML and web applications to ensure that text is displayed correctly regardless of language.
- **File Storage**: ASCII and Unicode-based encodings are used in file formats like text files (.txt), programming source code files, and configuration files to store and interpret textual data.
- **Networking**: Data transmitted over the internet or other networks must be encoded into a standard format (like Base64 or UTF-8) so that the receiving system can decode it correctly.
- **Data Interchange**: Encoding methods are essential for exchanging data between systems in formats like XML, JSON, and CSV, ensuring that text data remains consistent across different environments.

Types of Encoding Methods:

There are different types of encoding scheme including:

- ✓ Legacy/out dated encoding methods;
  - BCD-4;
  - Bcd-6;
  - ANSII;
  - ASCII;
  - EBCDIC;
- ✓ Modern encoding methods:
  - UTF-8
  - UTF-16
  - UTF-32

Digital Logic & Design

1. BCD-4 (Binary Coded Decimal - 4 bits)

- **Purpose**: BCD-4 was created to represent decimal digits (0-9) in binary form, using 4 bits per digit.
- **Developer**: The origins of BCD stem from early computer design, dating back to the mid-20th century.
- **Code Size**: 4 bits (half a byte).
- **Total Characters**: It can represent 16 combinations (2^4), but typically only the digits 0-9 are used.
- **Applications**: Used in early calculators, digital clocks, and some early computer systems where decimal data needed to be represented efficiently.
- **Relevance Today**: Still used in digital systems where binary-encoded decimal digits are needed for simple numeric representations.

2. BCD-6 (Binary Coded Decimal - 6 bits)

- **Purpose**: BCD-6 expanded the capability of BCD-4, allowing representation of both numeric and alphanumeric characters.
- **Developer**: Like BCD-4, it evolved in the early days of computing.
- **Code Size**: 6 bits.
- **Total Characters**: 64 characters (2^6), enough to include digits (0-9), upper-case alphabetic characters (A-Z), and some control characters.
- **Applications**: Used in early punched card systems and telegraphy.
- **Relevance Today**: Rarely used today, as more efficient encoding methods have taken over.

3. ASCII (American Standard Code for Information Interchange)

- **Purpose**: ASCII was developed to standardize character encoding, allowing computers to share data such as text consistently.
- **Developer**: Developed by the American National Standards Institute (ANSI) in 1963.

- **Code Size**: 7 bits, though often stored in 8-bit (1 byte) for convenience.
- **Total Characters**: 128 characters (2^7), including printable characters (letters, digits, punctuation) and control characters.
- **Applications**: Widely used in early computing for text data representation, programming languages, and file formats.
- **Relevance Today**: Still fundamental to many encoding systems, with UTF-8 retaining compatibility with ASCII.

4. ANSI (American National Standards Institute)

- **Purpose**: ANSI extended ASCII to allow more characters, particularly for international languages and symbols.
- **Developer**: American National Standards Institute.
- **Code Size**: 8 bits (1 byte).
- **Total Characters**: 256 characters (2^8), including additional symbols, graphics, and characters from non-English languages.
- **Applications**: Used in Windows and other systems for extended character sets.
- **Relevance Today**: Largely replaced by Unicode systems (e.g., UTF-8), but still found in legacy systems and text encoding on Windows.

5. EBCDIC (Extended Binary Coded Decimal Interchange Code)

- **Purpose**: Developed by IBM, EBCDIC was used primarily in IBM mainframes and early computers to represent text and control data.
- **Developer**: IBM, around the 1960s.
- **Code Size**: 8 bits (1 byte).
- **Total Characters**: 256 characters (2^8).
- **Applications**: Used in IBM systems, particularly mainframes and early computing environments.

- **Relevance Today**: Still used in some legacy IBM systems but has been mostly replaced by modern encodings.

6. UTF-8 (Unicode Transformation Format - 8 bits)

- **Purpose**: UTF-8 is a variable-length character encoding system designed to encode all Unicode characters, ensuring backward compatibility with ASCII.
- **Developer**: UTF-8 was created by Ken Thompson and Rob Pike at Bell Labs in 1992.
- **Code Size**: Variable (1-4 bytes).
- **Total Characters**: Capable of encoding 1,112,064 characters.
- **Applications**: Most widely used encoding in web pages, email, databases, and file systems.
- **Relevance Today**: Dominates modern text encoding due to its efficiency and compatibility with ASCII. It's the default encoding for many systems, including web browsers and operating systems.

7. UTF-16 (Unicode Transformation Format - 16 bits)

- **Purpose**: UTF-16 is a variable-length encoding method designed to encode Unicode characters using 16 bits, which can be extended to 32 bits for certain characters.
- **Developer**: UTF-16 is part of the Unicode Standard, developed by the Unicode Consortium.
- **Code Size**: Variable (2 or 4 bytes).
- **Total Characters**: 1,112,064 characters.
- **Applications**: Used in Windows operating systems, Java programming language, and certain file formats (e.g., Microsoft Word documents).
- **Relevance Today**: Commonly used in certain applications that need a balance between memory efficiency and full Unicode support.

8. UTF-32 (Unicode Transformation Format - 32 bits)

- **Purpose**: UTF-32 is a fixed-length encoding method that uses 32 bits to encode each Unicode character.

Digital Logic & Design

- **Developer**: Unicode Consortium.
- **Code Size**: 4 bytes (32 bits).
- **Total Characters**: 1,112,064 characters.
- **Applications**: Used in applications where memory usage is not a concern, but ease of indexing and fixed character width is important (e.g., internal data structures).
- **Relevance Today**: Rarely used for data transmission or storage due to its inefficiency in terms of memory but useful for internal processing where fixed-size encoding is important.

Comparison of Encoding Methods

Encoding Method	Code Size	Total Characters	Application Areas
BCD-4	4 bits	10 (digits 0-9)	Early calculators, digital clocks
BCD-6	6 bits	64	Early telegraphy, punched cards
ASCII	7 bits	128	Early computing, programming languages
ANSI	8 bits	256	Legacy Windows systems, extended symbols
EBCDIC	8 bits	256	IBM mainframes
UTF-8	1-4 bytes	1,112,064	Web pages, databases, emails
UTF-16	2 or 4 bytes	1,112,064	Windows OS, Java
UTF-32	4 bytes	1,112,064	Internal processing, data structures

Each of these encoding methods serves different purposes based on their design and efficiency, with modern systems favoring UTF-8 for its flexibility and backward compatibility with ASCII.

BCD-4:

Digital Logic & Design

BCD-4 encodes each decimal digit with its binary equivalent using four bits. So decimal digits are simply represented in four bits by their direct binary values. A disadvantage of this is that only 10 of the possible 16 (2^4) codes that four bits can produce are used. Hence it is an inefficient code. Nevertheless, the advantages usually outweigh this disadvantage and so it is regularly used.

- ✓ It stands for "Binary Coded Decimal 4 bit".
- ✓ Code size: For each /letter it assigns 4 bits.
- ✓ Encoding Capacity: $2^n = 2^4$ = 16 symbols
- ✓ It encodes ten digits, numbers and some symbols.

BCD-6:

The BCD-6 code is the adaptation of the punched card code to a six-bit binary code by encoding the digit rows (nine rows, plus unpunched) into the low four bits, and the zone rows (three rows, plus unpunched) into the high two bits.

- ✓ BCD-6 Stands for "Binary Coded Decimal 6 bit".
- ✓ Code size: For each symbol/letter it assigns 6 bits.
- ✓ Encoding Capacity: $2^n = 2^6$ = 64 symbols
- ✓ It encodes mostly English letters, and some other character.

ASCII Encoding method:

ASCII is the most common character encoding format for text data in computers and on the internet. In standard ASCII-encoded data, there are unique values for 128 alphabetic, numeric or special additional characters and control codes. *ASCII characters are limited to the lowest 128 Unicode characters*, from U+0000 to U+007F.

- ✓ ASCII stands for "American Standard code for information interchange".
- ✓ Code size: For each symbol/letter it assigns 7 bits.
- ✓ Encoding Capacity: $2^n = 2^7$ = 128 symbols and letters.
- ✓ Early machines were based on this method.

EBCDIC:

Digital Logic & Design

EBCDIC is an eight-bit encoding scheme that standardizes how alphanumeric characters, punctuation and other symbols are interpreted by a computer's operating system (OS) and applications.

- ✓ EBCDIC stands for “Extended Binary Coded Decimal Interchange Code”.
- ✓ Code size: For each symbol/letter it assigns 8 bits.
- ✓ Encoding Capacity: $2^n = 2^8$ = 256 symbols and letters.
- ✓ Early machines were using this method as well.

UTF-8:

UTF-8 is an encoding system for Unicode. It can translate any Unicode character to a matching unique binary string, and can also translate the binary string back to a Unicode character. This is the meaning of “UTF”, or “Unicode Transformation Format.” Farvardin 15, 1403 AP.**UTF-8** is a Character standard used for electronic communication. Defined by the Unicode Standard, the name is derived from *Unicode Transformation Format – 8-bit*.

UTF-8 is capable of encoding all 1,112,064 (17 planes times 2^{16} code points per plane, minus 2^{11} technically-invalid surrogates) valid Unicode code points using a variable-width encoding of one to four one-byte (8-bit) code units.

- ✓ UTF-8 stands for “Unicode Transformation format 8 bits”.
- ✓ First it assigns 8 bits codes, then 16 bits, then 24 bits, and finally 32 bits.
- ✓ It is a variable size-length encoding method. It does not assign same size code for every symbol.
- ✓ Encoding Capacity: $2^8 + 2^{16} + 2^{24} + 2^{32}$.
- ✓ It can encode all symbols/characters of the world.

UTF-16:

- ✓ UTF-16 stands for “Unicode Transformation format 16 bits”.
- ✓ It is a variable size-length encoding method. It does not assign same size code for every symbol.
- ✓ Initially letters are encoded in 16 bits, then 32 bits.
- ✓ Encoding Capacity: $2^{16} + 2^{32}$.
- ✓ It can encode Known characters of the world.

UTF-32:

Digital Logic & Design

- ✓ UTF-32 stands for “Unicode Transformation format 32 bits”.
- ✓ It is a fixed-length encoding method. It assigns same size (32 bits) code for every symbol.
- ✓ Encoding Power: 2^{32} .
- ✓ It can encode all languages, punctuations marks, special characters, non-printable characters, and all significant symbols ($\pi, \propto, \emptyset$), etc.