Character Data and Character Encoding

Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

September 10, 2023

Outline

Acknowledgement

The content of most slides come from the authors of the textbook:

Null, Linda, & Lobur, Julia (2018). The essentials of computer organization and architecture (5th ed.). Jones & Bartlett Learning.

Table of Contents

Lesson Objectives

Students are expected to be able to

1. describe the fundamentals of numerical data representation and manipulation in digital computers;
2. convert between various radix systems;
3. convert and perform arithmetic in signed integer representations;
4. explain how errors can occur in computations because of overflow and truncation;
5. express floating numbers in floating-point representation;
6. *recognize the most popular character codes*; and
7. describe the concepts of error detecting and correcting codes.

Table of Contents

Character Data

Computers need to represent text data, i.e., human understandable characters, which requires us to represent human-understandable characters in bit patterns using some sort of character encoding scheme

- (Input) We also need to store the results of calculations, and provide a means for data input.
- (Output) Calculations aren't useful until their results can be displayed in a manner that is meaningful to people.

Calculations aren't useful until their results can be displayed in a manner that is meaningful to people.

Table of Contents

Evolution of Character Codes

Two observations:

- Computers become aster and have larger memory
- There is a need to support diverse languages on Earth

Character codes evolve from simple to complex and to support the world's languages

Early Character Codes

- 6-bit codes, e.g., the Binary-Coded Decimal (BCD) code, used by IBM mainframes in the 1950s and 1960s
- 8-bit codes, e.g., the Extended Binary-Coded Decimal Interchange Code (EBCDIC), one of the first widely-used computer codes that supported upper and lowercase alphabetic characters, in addition to special characters, such as punctuation and control characters.
- 7-bit codes, e.g., the American Standard Code for Information Interchange (ASCII).
- 8-bit codes, e.g., the Extended American Standard Code for Information Interchange (Extended ASCII), with added support for special characters.

EBCDIC and BCD are still in use by IBM mainframes today. ASCII and Extended ASCII the dominant character code outside the IBM mainframe world until recently.

Table of Contents

Unicode

Many of today's systems embrace Unicode, a system that can encode the characters of every language in the world.

The Java programming language, and some operating systems now use Unicode as their default character code.

The Unicode is evolving too.

- It begins with 16-bit character codes, and
- now extends to 21-bit.

Unicode Standard

- Codespace. The Unicode Standard defines a codespace covering interval $[0, 17 \times 2^{16})$
- Code point. Each code in the codespace is called a code point, often written as "U+Hexadecimal Value (at least 4 digits)" format.
  - In this foramt, the smallest codepoint is U+0000 while the largest U+10FFFF.
- The codespace is a systematic, architecture-independent.
- Actual text is processed as binary data via one of several Unicode encodings, e.g.,
  - UTF-8
  - UTF-16
  - UTF-32

Codeplanes

The Unicode codespace is divided into 17 planes, numbered 0 to 16. Plane 0 is called the basic plane while planes 1–17 the supplementary planes.

Plane #	Code Points	Plane Name	Acronym
0	U+0000–U+FFFF	Basic Multilingual Plane	BMP
1	U+10000–U+1FFFF	Supplementary Multilingual Plane	SMP
2	U+20000–U+2FFFF	Supplementary Ideographic Plane	SIP
3	U+30000–U+3FFFF	Tertiary Ideographic Plane	TIP
4–13	U+40000–U+DFFFF	unassigned	
14	U+E0000–U+EFFFF	Supplementary Special-purpose Plane	SSP
15–16	U+F0000–U+10FFFF	Supplementary Private Use Area planes	SPUA-A/B

Code Blocks in BMP

Each code plane consists of multiple code blocks, each is one of several contiguous ranges of numeric character codes with an assigned name.

- BMP contains characters for almost all modern languages, and numerous symbols.
- It is to support the unification of prior character sets as well as characters for writing.
- Most of the assigned code points in the BMP are used to encode Chinese, Japanese, and Korean (CJK) characters.

Example BMP Code Blocks

This table is to provide a snapshot, and is not accurate.

Scripts	Language	# of Characters	Code Points
Alphabets	Latin, Greek, Cyrilic, Hebrew, Arabic, ...	8192	U+0000–U+1FFFF
Symbols	Punctuation, Mathematics, Miscellaneous Symbols, ...	4096	U+2000–U+2FFFF
CJK	Chinese, Japanese, and Korean Symbols and Punctuation, ...	4096	U+3000–U+3FFFF
Han	Unified Chinese, Japanese, and Korean Ideographs and Extensions, ...	45056	U+4000–U+EFFF

Table of Contents

Unicode Encoding

Unicode code space is large, but a particular computer system only uses a subset most frequently. Unicode encoding can save space.

- UTF-8, mostly widely used
- UTF-16
- UTF-32

UTF-8

A is a variable-length character encoding standard that use 1 to 4 bytes to represent a Unicode character. The following table defines the conversion between Unicode code point and variable UTF-8 character bytes:

Code Point	Bits in Bytes			
	Byte 1	Byte 2	Byte 3	Byte 4
U+0000–U+007F	$0b_6b_5b_4b_3b_2b_1b_0$			
U+0080–U+07FF	$110b_{10}b_9b_8b_7b_6$	$10b_5b_4b_3b_2b_1b_0$		
U+0800–U+FFFF	$1110b_{15}b_{14}b_{13}b_{12}$	$10b_{11}b_{10}b_9b_8b_7b_6$	$10b_5b_4b_3b_2b_1b_0$	
U+10000–U+10FFFF	$11110b_{20}b_{19}b_{18}$	$10b_{17}b_{16}b_{15}b_{14}b_{13}b_{12}$	$10b_{11}b_{10}b_9b_8b_7b_6$	$10b_5b_4b_3b_2b_1b_0$

UTF-8: Examples

	Character and Code Point		UTF-8 Code	
	U+Hex	Binary	Binary	Hex
$	U+0024	010 0100	0010 0100	24
£	U+00A3	000 1010 0011	1100 0010 1010 0011	C2A3
乐	U+4E50	0100 1110 0101 0000	1110 0100 1011 1001 1001 0000	E4B990
𩸽	U+2825F	0 0010 1000 0010 0101 1111	1111 0000 1010 1000 1000 1001 1001 1111	F0A88A9F

Surrogate Codes

BMP defines two surrogate codes.

- Reserved for encoding non-BMP characters using a pair of 16-bit codes: one High Surrogate and one Low Surrogate.
- A single surrogate code point will never be assigned a character.
- The High Surrogate: U+D800–U+DBFF
- Low Surrogate: U+DC00–U+DFFF

UTF-16

In general, for code points from U+0000 to U+FFFF excluding the surrogate codes:

- the character is encoded directly using the 16 bit code point

For code points from U+010000 to U+10FFFF, the character encoded as two 16-bit code units called a surrogate pair, consisting of a high surrogate and a low surrogate

UTF-16: Surrogate Pair

1. 0x10000 is subtracted from the code point (U), leaving a 20-bit, i.e.,
$$U' = U - 0x10000 \tag{1}$$
number (U') in the hex number range 0x00000-0xFFFFF.
2. The high ten bits (in the range 0x000-0x3FF) are added to 0xD800 to give the first 16-bit code unit or high surrogate (W_1), which will be in the range 0xD800-0xDBFF.
$$W_1 = U' >> 10 \tag{2}$$
$$W_1 = W_1 + 0xD800 \tag{3}$$
3. The low ten bits (also in the range 0x000-0x3FF) are added to 0xDC00 to give the second 16-bit code unit or low surrogate (W_2), which will be in the range 0xDC00-0xDFFF.
$$W_2 = U'0x03FF \tag{4}$$
$$W_2 = W_2 + 0xDC00 \tag{5}$$

UTF-8: Examples

Character	Character and Code Point U+Hex	Character and Code Point Binary	UTF-16 Code Hex
$	U+0024	0000 0000 0010 0100	0024
£	U+00A3	0000 0000 1010 0011	00A3
乐	U+4E50	0100 1110 0101 0000	4E50

Character	Character and Code Point U+Hex	Character and Code Point Binary	UTF-16 Code W_1	UTF-16 Code W_2
𠘨	U+2825F	0010 1000 0010 0101 1111	D860	DE5F

UTF-32

A fixed-length encoding used to encode Unicode code points that uses exactly 32 bits (four bytes) per code point

To encode a Unicode character, simply pad the code point to 32 bits.

Table of Contents

Summary and Q&A

You are expected to be able to

1. *recognize the most popular character codes*;

Any questions on:

- Character Data
- Early Character Codes
- Unicode and Unicode Character Encoding
  - UTF-8
  - UTF-16
  - UTF-32