Data Representation

Introduction

- Binary digit (bit): A single 0 or 1
- A specific character can be represented using a group of 0s and 1s.
- Byte: Number of bits used to store 1 character
    - 8 bits long
        - So it can represent 2^8 = 256 different characters
- Character set: The complete set of characters a computer system can use
    - Each character has its own binary code
    - e.g. ASCII, Unicode

American Standard Code for Information Interchange (ASCII)

- Uses 7 bits per character
    - The 8th bit is either unused or used for parity checking (detecting errors during data transmission).
- Has 128 unique codes
    - 32 control codes (non-printing characters e.g. Enter, Tab, Backspace)
    - 96 printable characters
- Only suitable for English

Unicode

- Uses 1-4 bytes per characters
- Can support over 1 million possible characters
    - Essential for representing all the world's various languages and scripts
- First 128 characters are identical to ASCII

    - Retains backward compatibility with ASCII — any text written in ASCII/extended ASCII will be correctly displayed in Unicode.
- Usage:
    - Documents and software that handle multiple languages
    - Supports software localisation — standard applications can be adapted for use in different cultures by simply changing the language and interface
    - Global communication
- Unicode with UTF-8 encoding is the most common format on the Internet.
- Expected to eventually replace ASCII as the universal character set standard

Comparing character sets

Feature	ASCII	Unicode (UTF-8)
Bits per character	7 bits (stored in 8)	1-4 bytes
Characters supported	128	Over 1 million
Content covered	English letters, digits, symbols, control codes	All world languages, scripts, emojis and symbols
Backward compatibility	Original standard	First 128 codes are the same as ASCII
Standardisation	Fixed	International standard is UTF-8
Usage	Early computers	Modern software, internet, multilingual documents

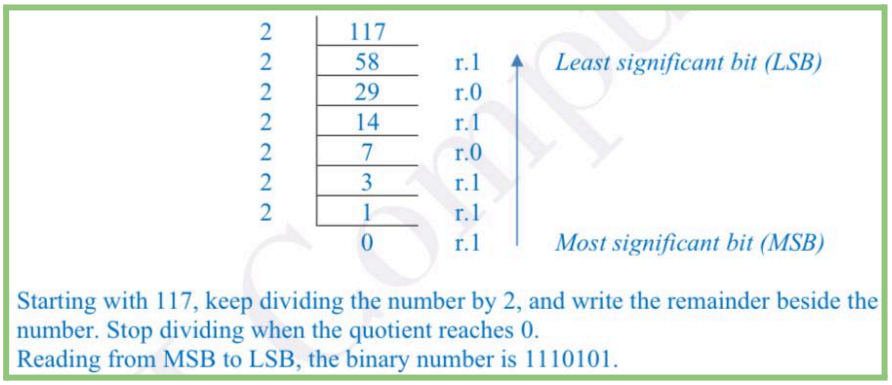
Limitation	English only	Variable-length encoding, so larger storage for some characters

Number systems

- Denary
    - Standard number system
    - Base-10
- Binary
    - Only has 2 digits: 0 and 1
    - Other representations need to be converted to binary for the computer to process them.
    - Binary addition:
        - 0 + 0 = 0
        - 0 + 1 = 1
        - 1 + 0 = 1
        - 1 + 1 = 0 and carry 1
    - If each number is represented by a byte, the smallest binary number 00000000 will represent 0 in denary, and the largest binary number 11111111 will represent 255 in denary.
- Hexadeimal (hex)
    - Base-16
    - Uses the digits 0-9 and letters A-F
    - **Advantage:** Shorter and more readable representation of binary numbers.
        - Long binary patterns can be written as short hex numbers without losing the relationship between the value and the individual bits in the pattern, as each group of 4 bits is represented by one digit.

- Usage:
  - Memory
    - Represents memory location addresses in a shorter and easier to read format than binary
  - Hex colours
    - Represents colours defined using the RGB system, where each channel is represented by 2 hex digits
    - Makes colour values short, readable and easy to use in digital design
  - Bitwise operations (setting, clearing, checking etc. bits in memory or registers)
    - Used to manage individual bits at low-level programming
    - Hex numbers provide a readable shortcut to binary values, making it easier to see which bits are being affected.

Conversions

- Denary to binary:
  - Divide by 2 until the quotient reaches 0, taking note of the remainder (1 or 0) at each step.
  - Read the remainders in reverse order.
  - 

- Binary to denary:

- Multiply each digit by the corresponding position value and sum them up.
- Using the same example **10110110**, starting from the right most digit,

  $\mathbf{0}x2^0 + \mathbf{1}x2^1 + \mathbf{1}x2^2 + \mathbf{0}x2^3 + \mathbf{1}x2^4 + \mathbf{1}x2^5 + \mathbf{0}x2^6 + \mathbf{1}x2^7$

  $= 0 + 2 + 4 + 0 + 16 + 32 + 0 + 128$

  $= 182$

- Binary to hexadecimal: Split the number into 4-bit groups and find the respective hexademical codes of each.
- Hexadecimal to binary: Find the 4-bit groups represented by each hex digit.
- Denary to hexademical:
  - Divide the number by 16 until the quotient reaches 0, taking note of the remainder at each step.
  - Convert each remainder to its hexadecimal equivalent and read in reverse order.
  - Example: **Convert denary 2603 to hexadecimal.**

16	2603		
16	162	r.11 (B)	*Least significant bit (LSB)*
16	10	r.2	
	0	r.10 (A)	*Most significant bit (MSB)*

    Starting with the denary number 2603, keep dividing by 16, and write the remainder beside the number. Stop dividing when the quotient reaches 0.
    Reading from MSB to LSB, the hexadecimal number is A2B.
- Hexadecimal to denary:
  - Convert each hex digit to its denary value.
  - Multiply each denary value by its positional value and add up the results.
  - Example: **Convert hexadecimal A2B to denary.**
    - Convert each hex digit to its denary value: $A_{16} = 10$, $2_{16} = 2$, $B_{16} = 11$
    - Multiply each denary value by its positional value and sum the results:
      - $A2B = (10 \times 16^2) + (2 \times 16^1) + (11 \times 16^0) = 2603.$

@star.notes