Computer Science 1: IT and web

coefficient: 2
credit: 3
continuous control weight: 50%
exam weight: 50%

Mrs HamzaCherif Souaad

Course 3: Information representation and coding

Goals

- Introduce the notion of codification of information, by exposing the representation of different types of information, especially integers, real numbers, as well as characters.

1-Representation of natural numbers

1. Representation of natural numbers:

- A natural number is a positive or zero number. To represent such a number, we must determine the number of bits to use to encode it, which depends on the number we wish to encode.

- To encode natural integers between 0 and 255, we will only need 8 bits (= one byte) because $2^8 = 256$.

- Generally speaking, n-bit coding can be used to represent natural integers between 0 and 2n – 1.

1. Representation of natural numbers:

- **Examples:**
  - $9_{10} = (00001001)_2$
  - $128_{10} = (10000000)_2$

2-Representation of relative integers:

2. Representation of relative integers:

- A relative integer is an integer that can be negative or positive.
- The number must therefore be coded in such a way that we can know whether it is a positive number or a negative number.
- The trick is to use a coding called two's complement to encode a negative number.
- This representation allows us to perform the usual arithmetic operations naturally.

2.1 Representation of positive or zero relative integers

- A positive or zero relative integer will be represented in binary (base 2) as a natural integer, with the only difference being that the most significant bit (the leftmost bit) represents the sign.
- It is therefore necessary to ensure that for a positive or zero integer, the most significant bit is zero (0 corresponds to a positive sign, 1 corresponds to a negative sign).

2.1 Representation of positive or zero relative integers

- On 8 bits (1 byte), the coding interval is [-127, 127].
- On 16 bits (2 bytes), the coding interval is [-32767, 32767].
- On 32 bits (4 bytes), the coding interval is [-2147483647; 2147483647].
- Generally speaking, the largest positive relative integer coded on N bits will be 2n–1 – 1.

2.1 Representation of positive or zero relative integers

- **Example :**
- if we encode a positive relative integer on 4 bits, the largest number will be 0111 (i.e. 7 in decimal base).

2.2 Representation of negative relative integers (two's complement):

- A negative relative integer will be represented using two's complement coding.

2.2 Representation of negative relative integers (two's complement):

- **Two's complement principle:**
- Write the absolute value of the number in base 2. The most significant bit must be equal to 0.
- Reverse the bits: 0s become 1s and vice versa (one's complement).
- We add 1 to the result (overflows are ignored).
- This operation corresponds to the calculation of 2n – |x|, where n is the length of the representation and |x| the

2.2 Representation of negative relative integers (two's complement):

- **Example** :
- To encode the number -19 on 8 bits, simply:
  - Write 19 in binary: 00010011
  - Write its complement to 1: 11101100
  - Add 1 to the complement of 1: 11101101
- The binary representation of -19 on 8 bits is therefore: 11101101.

2.2 Representation of negative relative integers (two's complement):

- **Example** :
- Note that by adding a number and its two's complement we obtain 0.
    - 00010011+ 11101101=00000000 (with a carry of 1 which is eliminated).

2.3 Representation in sign and absolute value:

- The principle is to consider that the most significant bit is reserved to encode the sign with:
  - **0 → positive integer.**
  - **1 → negative integer.**
- The most significant bit is leftmost, the other bits encode the number in absolute value.
- It is necessary to know how many bits the number is encoded in to determine which bit encodes what.

2.3 Representation in sign and absolute value:

- **Example:** 4-bit coding:
- $(0111)_2 = 7$ because the most significant bit is 0.
- $(1111)_2 = -7$ because the most significant bit is 1.

3-Coding a decimal number as a fixed point

3. Coding a decimal number as a fixed point

- A representation of a fixed point number corresponding to a representation having a fixed number of digits after the decimal point.
- In this coding, the integer part of the number can be translated by positive powers of 2, and the fractional part will be translated by negative powers of 2.

3. Coding a decimal number as a fixed point

- **Example1 :**
  - $25=1*2^4+1*2^3+0*2^2+0*2^1+1*2^0$.
  - $0.375=0*2^{-1}+1*2^{-2}+1*2^{-3}$
- Therefore the number 25.375 is translated as 11001.011 in fixed point.

3. Coding a decimal number as a fixed point

- **Example2 :**
  - 8-digit word with (n,m) = (5,3) ⇒ n+m= 8
    - N $=(11001, 011)_2 = (25{,}375)_{10}$
    - N $=(11001011)_{2(5,3)} = (25{,}375)_{10}$
  - 3-bit word (m + n) = 3 `
    - $(010)_{2(3,0)} = (2)_{10}$
    - $(010)_{2(2,1)} = (1)_{10}$
    - $(010)_{2(1,2)} = (0.5)_{10}$
    - $(010)_{2(0,3)} = (0.25)_{10}$

4-Character encoding

4 Character encoding

- In a computer, data is always represented in binary form (a series of 0s and 1s).
- However, humans are generally not fluent in binary language. It must therefore “translate” everything so that the machine can execute the instructions relating to the installed programs.
- **How are the texts coded?**

4 Character encoding

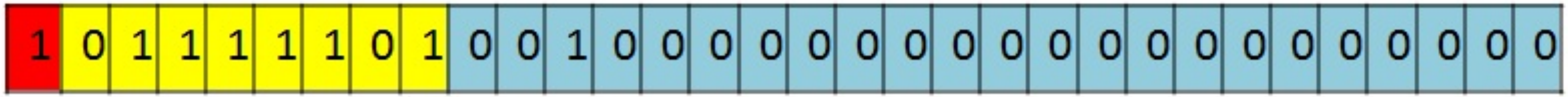
- A text is a series of characters, so we will instead ask ourselves the question, **how are the characters stored inside the machine?**
- The answer is simple, each character is associated with a binary code.

4.1 Character encoding : ASCII code

- Historically, one of the first methods of character encoding is called ASCII (American Standard Code for Information Interchange).
- In the ASCII coding system, each character is encoded on one byte. In reality of the 8 bits only 7 are used to encode the characters (the 8th bit, called parity bit, is used to detect errors). $2^7 = 128$ characters can be encoded in ASCII.

4.1 Character encoding : ASCII code

- For example the character 'A' is represented by the binary code 1000001, the character '4' is represented by the binary code 0110100.



4.1 Character encoding : ASCII code

- At the beginning of the history of computing this did not pose too many problems, but with the arrival of office automation tools (word processing, etc.), it became problematic: for example,
- for French, accented characters ('´e', '`a', '@', ...) are not coded in the ASCII system. To overcome this difficulty, the ASCII code has been extended to an 8 bit UTF code -8 (Universal character set Transformation Format).

4.1 Character encoding : UTF-8 code

- To ensure compatibility with ASCII, characters encoded in ASCII have exactly the same code in UTF-8.
- To exceed the ASCII character limit of 128 characters, in UTF-8, certain characters are encoded on more than one byte.
- The ASCII standard establishes a correspondence between a binary representation of the characters of the Latin alphabet and the symbols, the signs, which constitute this set.

4.1 Character encoding : UTF-8 code

- Example:
- The character 'a' is coded 1100001 (ASCII code = 97). the character 'A' is coded 1000001 (ASCII code = 65).
- The ASCII standard allows all kinds of machines to store, analyze and communicate textual information.