Codage de l'information dans les systèmes informatiques

Les ordinateurs sont des outils extrêmement polyvalents, pouvant traiter des informations de toute nature : textes, images, vidéos, sons, etc. Or les circuits internes d'un ordinateur ne savent que manipuler que des données binaires, c'est-à-dire des nombres écrits en base 2. Comment arriver à coder tout type d'information en binaire ?

I./ Représentation des nombres :

1./ Entiers naturels :

Les entiers naturels sont des nombres entiers positifs ou nuls. Ce sont les plus simples à représenter : il suffit tout simplement de les traduire en base 2. Par exemple, le nombre entier 85 se code 1010101 en binaire (voir fiche méthode).

Pour des raisons pratiques, les nombres entiers sont codés en utilisant un nombre fixe de bits, ce qui fixe la fourchette de nombres que l'on souhaite représenter. Par exemple, avec 8 bits, on peut coder les nombres de 0 (00000000) à 255 (11111111).

Exercices : (Pour tous les exercices, l'utilisation de la calculatrice de Windows est autorisé)

❶ Combien de nombres différents peut-on représenter en utilisant un codage sur 4 bits ? Sur 10 bits ? Sur 16 bits ?

4 bits : on peut coder 2^4 = 16 nombres (de 0 à 15).
10 bits : on peut coder 2^{10} = 1024 nombres (de 0 à 1023)
16 bits : on peut coder 2^{16} = 65536 nombres (de 0 à 65535)

❷ Traduire en binaire les nombres entiers naturels suivants : 24, 136, 725, 1999.

$(24)_{10} = (11000)_2$
$(136)_{10} = (10100000)_2$
$(725)_{10} = (1101011101)_2$
$(1999)_{10} = (11111001111)_2$

2./ Entiers relatifs :

Les entiers relatifs sont des nombres entiers qui peuvent être positifs, négatifs ou nuls. Une possibilité pour pouvoir noter ce type de nombres est d'utiliser un des bits pour coder le signe. On peut prendre pour convention que le bit de signe vaudra 0 pour un nombre positif et 1 pour un nombre négatif. Prenons deux exemples en codant les nombres sur 8 bits :

1	0	0	1	1	1	0	0

$(0011100)_2 = (28)_{10}$. Le premier le plus à gauche (en grisé) est le bit de signe. Le nombre représenté est donc égal à -28.

0	1	1	1	0	0	0	1

$(1110001)_2 = (113)_{10}$. Le premier le plus à gauche (en grisé) est le bit de signe. Le nombre représenté est donc égal à +113.

Avec 8 bits, on peut représenter les nombres de -127 (**1**1111111) à +127 (**0**1111111). Cette technique de représentation des entiers relatifs présente quelques défauts. L'un d'entre eux est que le nombre 0 peut être codé de deux façons différentes : (**0**0000000) ou (**1**0000000). Dans les ordinateurs actuels, c'est donc une représentation différente pour les nombres négatifs qui est généralement utilisée : la *notation en complément à deux*. Nous ne détaillerons pas ici cette méthode.

Exercices :

❶ En utilisant cette technique de codage, quelle est la fourchette de nombres que l'on peut coder en utilisant 10 bits ? Et 16 bits ? (Le nombre de bits indiqués tient compte du bit de signe).

10 bits : 1 bit de signe + 9 bits pour le nombre. $2^9 = 512 \Rightarrow$ de –511 à +511
16 bits : 1 bit (signe) + 15 bits (nombre). $2^{15} = 32768 \Rightarrow$ de –32767 à +32767

❷ Traduire en binaire les nombres entiers relatifs suivants : -48, +111, -30500. On utilisera un codage sur 8 bits pour les deux premiers nombres, et 16 bits pour le troisième nombre.

–48 ⇒ 10110000
+111 ⇒ 01101111
–30500 ⇒ 1111011100100100
(Le bit souligné correspond au bit de signe)

3./ Nombres réels :

Un ordinateur sait aussi manipuler les nombres réels, c'est-à-dire les nombres à virgule. Une astuce pour représenter ce type de nombre est de constater que tout nombre réel x peut s'écrire sous la forme : $x = a \times 10^b$, où le nombre a est appelé *mantisse* et le nombre b est appelé *exposant*. Par exemple : 3,14159 = 314159 $\times$ 10^{-5}. Dans ce cas, a = 314159, et b = -5. Les nombres a et b sont des entiers relatifs que nous savons coder en binaire. La représentation binaire du nombre à virgule 3,14159 sera donc la représentation binaire des deux nombres entiers 314159 et -5. On appelle cette méthode de codage la *représentation des nombres en virgule flottante*. Il existe une autre méthode, appelée *représentation en virgule fixe*, moins performante mais plus rapide.

Note : les ordinateurs utilisant un nombre donné de bits pour représenter les nombres en virgule flottante, certains nombres seront remplacés par une approximation. Le nombre π par exemple a une infinité de nombres après la virgule. Pour le représenter, il faudrait donc utiliser une infinité de bits, ce qui est impossible. Les calculs effectués par un ordinateur (ou une calculatrice) souffrent donc tous d'une certaine imprécision. Plus le nombre de bits utilisés pour coder un nombre est grand, meilleure est la précision du calcul.

II./ Représentation des textes :

1./ Code ASCII standard :

Un texte est une suite de caractères qui peuvent être des lettres (majuscules ou minuscules), des signes de ponctuation, des chiffres, des caractères spéciaux, etc. La méthode permettant de coder sous forme de nombres binaires un ensemble de caractères est donc d'attribuer à chaque caractère un numéro particulier. Par exemple, A = 1, B = 2, etc. Le texte sera ainsi représenté dans la mémoire de l'ordinateur par une suite de nombres binaires. Au début de l'informatique, chaque constructeur utilisait sa propre table de correspondance, ce qui pouvait créer des incompatibilités. C'est pourquoi un code standard fut inventé, le *code ASCII*. Ce codage utilise des nombres binaires à 7 bits. Il peut donc représenter 128 caractères, ce qui suffit amplement pour la langue anglaise.

La correspondance entre nombres et caractères pour le code ASCII 7 bits est donnée dans le tableau suivant. Les caractères de code compris entre 0 et 31 sont des caractères de contrôle. Leur rôle initial était de commander les périphériques branchés à l'ordinateur (écran, terminal, imprimante, etc.). Les caractères affichables commencent à partir du numéro 32.

0 NUL	32 espace	64 @	96 `
1 SOH	33 !	65 A	97 a
2 STX	34 "	66 B	98 b
3 ETX	35 #	67 C	99 c
4 EOT	36 $	68 D	100 d
5 ENQ	37 %	69 E	101 e
6 ACK	38 &	70 F	102 f
7 BEL	39 '	71 G	103 g
8 BS	40 (	72 H	104 h
9 HT	41)	73 I	105 i
10 LF	42 *	74 J	106 j
11 VT	43 +	75 K	107 k
12 FF	44 ,	76 L	108 l
13 CR	45 -	77 M	109 m
14 SO	46 .	78 N	110 n
15 SI	47 /	79 O	111 o
16 SLE	48 0	80 P	112 p
17 CS1	49 1	81 Q	113 q
18 DC2	50 2	82 R	114 r
19 DC3	51 3	83 S	115 s
20 DC4	52 4	84 T	116 t
21 NAK	53 5	85 U	117 u
22 SYN	54 6	86 V	118 v
23 ETB	55 7	87 W	119 w
24 CAN	56 8	88 X	120 x
25 EM	57 9	89 Y	121 y
26 SIB	58 :	90 Z	122 z
27 ESC	59 ;	91 [	123 {
28 FS	60 <	92 \	124 |
29 GS	61 =	93]	125 }
30 RS	62 >	94 ^	126 ~
31 US	63 ?	95 _	127 ■

Question : Que veut dire ASCII ? En quelle année et par qui a été inventée cette technique de codage des caractères ?

ASCII : American Standard Code for Information Interchange
(Code américain normalisé pour l'échange d'information)
Le code ASCII a été inventé en 1961 par Robert Bemer.

2./ Code ASCII étendu :

Le code ASCII standard (à 7 bits) ne convenait pas pour écrire la plupart des autres langues que l'anglais, à cause notamment de l'absence de lettres accentuées. C'est pourquoi une version étendue à 8 bits a été développée.

Exercices :

❶ Avec 8 bits, combien de caractères différents peut-on représenter ?

$2^8 = 256$. On peut donc représenter 256 caractères avec 8 bits.

❷ Traduire le message secret suivant, écrit en utilisant le code ASCII à 8 bits :

01010110 01001001 01010110 01000101 00100000 01001100
00100111 01001001 00101110 01000011 00101110 01001110
00101110

VIVE L'I.C.N.

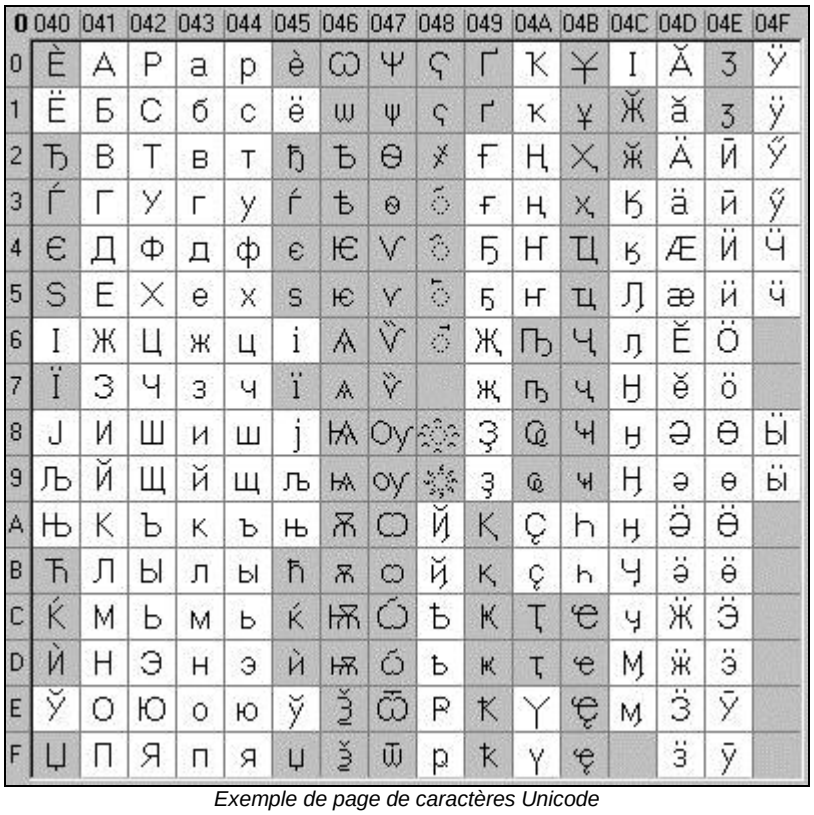
❸ Dans la question précédente, on a séparé les groupes de 8 bits pour rendre la lecture plus aisée et pour vous faciliter la tâche. En réalité, dans la mémoire de l'ordinateur, le message se présenterait sous la forme :

01010110010010010101011001000101001000000100110000100111010010010011011001000011001011100100111000101110

très peu lisible pour un être humain. D'où l'utilisation de la *notation hexadécimale* (base 16). Traduire le message précédent en nombres hexadécimaux.

3./ Code Unicode :

Le code ASCII étendu ne suffit pas à coder tous les caractères de toutes les langues écrites existantes. C'est pourquoi un nouveau standard a été créé, *l'Unicode*. Unicode permet de coder 65536 caractères.



Exemple de page de caractères Unicode

<u>*Questions :*</u> En quelle année Unicode a-t-il été inventé ? Combien de bits sont nécessaires pour coder un caractère en Unicode ?

Première publication d'Unicode : 1991
2^{16} = 65536. Il faut donc 16 bits (2 octets) pour coder un caractère en Unicode.

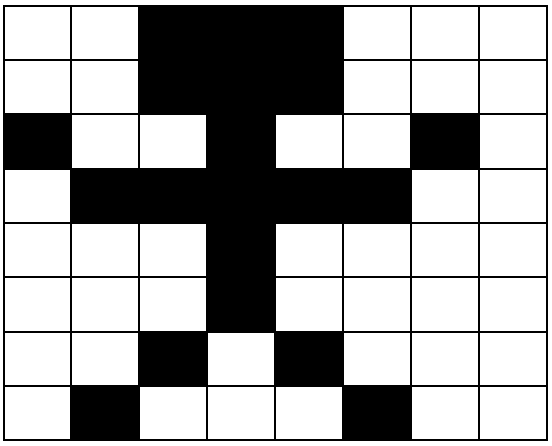
III./ Représentation des images :

Nous nous limiterons à l'étude des *images bitmap*. C'est la façon la plus simple de coder une image : l'image est décrite pixel par pixel (un *pixel* = un point de l'image). Chaque pixel est décrit par un nombre indiquant sa couleur.

1./ Image bitmap monochrome :

Dans une image monochrome, les pixels sont soit noirs, soit blancs. L'état de chaque pixel peut donc être codé par un seul bit. On peut utiliser la convention suivante : 0 = blanc, 1 = noir.

Soit l'image monochrome suivante, dont les dimensions sont : 8 pixels de large et 8 pixels de haut. Elle est donc composée de $8 \times 8 = 64$ pixels. Le nombre de bits nécessaires pour la coder est donc de 64, soit 8 octets.



0	0	1	1	1	0	0	0	38_{16}
0	0	1	1	1	0	0	0	38_{16}
1	0	0	1	0	0	1	0	92_{16}
0	1	1	1	1	1	0	0	$7C_{16}$
0	0	0	1	0	0	0	0	10_{16}
0	0	0	1	0	0	0	0	10_{16}
0	0	1	0	1	0	0	0	28_{16}
0	1	0	0	0	1	0	0	44_{16}

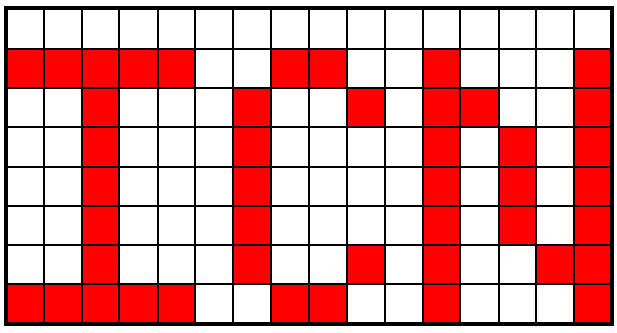
Si on convertit les nombres binaires correspondants en hexadécimal, l'image peut être représentée par le nombre suivant : $(3838927C10102844)_{16}$.

Exercices :

❶ On considère une image bitmap monochrome de 800 pixels sur 600. Combien cette image contient-elle de pixels ? Combien de bits sont nécessaires pour la coder ? Donner la taille de l'image en octets et en kibioctets, sachant que : un octet = 8 bits et que : 1 kibioctet = 1024 octets.

800 × 600 = 480 000 pixels ⇒ 480 000 bits
480 000 ÷ 8 = 60 000 octets
60 000 ÷ 1024 ≈ 58,6 kibioctets

❷ Dessiner l'image en noir et blanc de 8 pixels de haut pour 16 pixels de large dont le codage hexadécimal est : 00 00 F9 91 22 59 22 15 22 15 22 15 22 53 F9 91. Que représente cette image ?



2./ Image bitmap couleur :

Dans le cas d'une image couleur, chaque pixel peut prendre plus de deux états possibles. Par exemple, si l'image est en 256 couleurs, il faut pouvoir coder 256 états différents pour chaque pixel : il nous faut donc 8 bits (un octet) par pixel, soit 8 fois plus que pour une image monochrome. Mais 256 couleurs sont insuffisantes pour avoir un rendu réaliste. On utilise donc le format *bitmap couleurs réelles* (ou *True Colors*), qui utilise trois octets (soit 24 bits) pour coder un pixel. Trois octets permettent de coder 16.777.216 couleurs, soit environ 16,7 millions, ce qui est amplement suffisant, l'œil humain pouvant distinguer environ 2 millions de couleurs.

Exercices :

❶ On reprend l'exemple précédent de l'image bitmap de 800 pixels par 600. Si cette image est maintenant au format bitmap couleurs réelles, combien faut-il de bits pour la coder ? Quelle est sa taille en octets, en kibioctets, et en mébioctets, sachant que 1 mébioctet = 1024 kibioctets ?

800 × 600 × 3 = 1 440 000 octets
1 440 000 ÷ 1024 ≈ 1 406 kibioctets
1 406 ÷ 1024 ≈ 1,37 mébioctets

❷ Même question pour une image bitmap en couleurs réelles de taille 1024 pixels par 800. Que remarque-t-on ?

1024 × 800 × 3 = 2 457 600 octets
2 457 600 ÷ 1024 = 2 400 kibioctets
2 400 ÷ 1024 ≈ 2,34 mébioctets
On remarque que la place occupée par les images en mémoire ou sur le disque dur augmente très rapidement avec leur taille en pixels.

IV./ Conclusion :

Nous avons donc vu comment les nombres usuels, les textes ou les images pouvaient être représentés par des nombres binaires, et par là même pouvaient être manipulés par un ordinateur. Mais nous n'avons fait que survoler les techniques de codage de l'information dans les systèmes informatiques. Nous n'avons notamment pas parlé des techniques permettant de représenter les sons ou les séquences vidéo. Un autre point que nous n'avons pas évoqué est celui de la *compression* : certaines données prennent énormément de place en mémoire (voir l'exemple des images bitmap), et les techniques de compression permettent de réduire la taille de ses données. Il y a deux types de méthodes de compression : la *compression non destructive*, où il n'y a aucune perte de données (par exemple les fichiers ZIP), et la *compression destructive*, où il y a une perte plus ou moins importante d'information (par exemple les images JPEG ou les sons en MP3).

Pour les plus rapides :

En 1974, les scientifiques américains ont envoyé un message radio à partir du radiotélescope d'Arecibo, à Porto Rico. Ce message était en binaire et comportait 1679 bits.

❶ Retrouver le contenu du message, ainsi que sa traduction.
❷ Pourquoi la longueur du message a été fixée à 1679 bits ?
❸ Pourquoi utiliser le langage binaire pour cette tentative de communication interstellaire ?

❶ **cf. « Message d'Arecibo » sur Wikipédia**
❷ **1679 est un nombre premier égal au produit de 23 par 73. Il n'y a donc que deux possibilités pour placer les bits du message dans une grille (23 lignes de 73 colonnes ou 73 lignes de 23 colonnes).**
❸ **Le binaire permet de limiter les problèmes d'interférences pendant le trajet du signal, car il n'y a que deux niveaux (0 et 1) facilement identifiables).**