Représentation de l'information en binaire

Les ordinateurs sont capables d'effectuer de nombreuses opérations sur de nombreux types de contenus (images, vidéos, textes, sons,...). Cependant, quel que soit leur type, ces contenus sont manipulés par la machine au travers d'une représentation n'utilisant que des 0 et des 1, qui est dite « binaire ». Nous présentons dans cette fiche les principes de cette représentation pour les nombres et les textes.

1 L'information et sa représentation

Une information est une connaissance qui fait sens pour les personnes concernées et qui peut être représentée par des symboles. Bien sûr, la notion de sens n'est pas pertinente pour un ordinateur, qui n'est qu'une machine, et c'est uniquement la représentation de l'information sur laquelle il va agir. Ainsi un traitement d'informations par un ordinateur consiste-t-il en des transformations de leurs représentations. Ces transformations peuvent être des calculs, des réagencements, des ajouts, des suppressions, des copies et toute autre opération consistant en des manipulations de symboles. Pour éviter l'expression « représentation d'informations », un peu longue, on parlera parfois de manière équivalente de « donnée ».

Dans un ordinateur, une donnée est toujours une succession (appelée « mot ») de caractères 0 et 1. Ces caractères 0 et 1 sont appelés des **bits** (contraction de « **bi**nary digi**ts** », c'est-à-dire « chiffres binaires » en anglais). Par exemple, le mot 1011001101011 est une donnée, le mot 0 est une donnée, et le mot 1 aussi. Puisque la donnée minimale ne peut avoir que deux valeurs, on dit que la représentation est binaire.

La raison pour laquelle les ordinateurs manipulent des données binaires est liée au fonctionnement de leurs composants physiques. Les transistors et les condensateurs, qui sont les éléments de base d'un ordinateur, possèdent deux états stables : activé/désactivé ou chargé/déchargé. Ainsi, un transistor dans l'état activé va-t-il stocker l'information 1 (ou 0 s'il est dans l'état désactivé).

Un bit ne pouvant contenir qu'une valeur parmi deux possibles (0 ou 1), il faudra combiner une multitude de bits pour représenter une information plus complexe comme une valeur numérique entière ou à virgule, positive ou négative. C'est l'objet des Sections 2 et 3. De même, pour représenter un texte, on va d'abord représenter chaque caractère du texte par un nombre entier, puis représenter chaque nombre entier par plusieurs bits. C'est l'objet de la Section 4.

2 Représentation des nombres naturels

Les nombres naturels sont les nombres entiers positifs ou nuls : zéro, un, deux trois, quatre, etc. Afin d'expliquer comment sont représentés ces nombres en binaire, nous revenons d'abord sur l'écriture usuelle des nombres, appelée représentation décimale.

2.1 Numérations en base dix et en base deux

Le système décimal, appelé aussi base dix, est notre système de calcul quotidien. Compter en base dix signifie d'abord utiliser dix symboles pour écrire les nombres. Ce sont les chiffres de 0 à 9. Cela signifie ensuite « compter par paquets de dix ». Avec la notation habituelle de gauche à droite, cela se traduit par le fait que chaque chiffre dans un nombre est associé à une puissance de dix, selon sa position.
Par exemple, 2014 est le nombre composé de quatre unités, une dizaine (une dizaine est un paquet de dix unités), zéro centaine (une centaine est composée de dix paquets de dix unités) et deux milliers (un millier correspond à dix paquets de dix paquets de dix unités).
Autrement dit :

$$2014 = 2 \times 1000 + 0 \times 100 + 1 \times 10 + 4 \times 1$$

$$= 2 \times 10 \times 10 \times 10 + 0 \times 10 \times 10 + 1 \times 10 + 4 \times 1$$

ce qui donne (écrit avec les puissances de dix correspondantes) :

$$2014 = 2 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 4 \times 10^0$$

Par convention, $1 = a^0$ pour n'importe quel nombre a.

Pour représenter un nombre en binaire, nous n'avons que deux symboles, 0 et 1, mais toujours le même principe : « compter par paquets de deux ». Ainsi, toujours avec la notation usuelle de gauche à droite, cela se traduit par le fait que chaque chiffre dans un nombre binaire est associé à une puissance de deux.
Par exemple 11111011110 est le nombre composé de zéro unité, une « deuzaine » (un paquet de deux unités), une quatraine (deux paquets de deux), une huitaine (deux paquets de deux paquets de deux), une seizaine (deux paquets de deux paquets de deux paquets de deux), zéro paquet de trente deux, un paquet de soixante quatre, un paquet de cent vingt huit, un paquet de deux cent cinquante six, un paquet de cinq cent douze et un paquet de mille vingt quatre.
Autrement dit, en revenant pour le calcul par commodité à l'écriture habituelle en base dix :

$$11111011110 = 1\times2^{10}+1\times2^9+1\times2^8+1\times2^7+1\times2^6+0\times2^5+1\times2^4+1\times2^3+1\times2^1+1\times2^1+0\times2^0$$

ce qui donne (en remplaçant les puissances de deux par leur valeur) :

$$11111011110 = 1\times1024+1\times512+1\times256+1\times128+1\times64+0\times32+1\times16+1\times8+1\times4+1\times2+0\times1 = 2014$$

Le même nombre est donc représenté en base dix par 2014 et en base deux par 11111011110. C'est pourquoi il est important lorsqu'on s'intéresse aux représentations des nombres, de préciser de laquelle on parle : ainsi 1000 en base dix, c'est mille, mais 1000 en base deux, c'est huit. Et en base cinq (on utilise les cinq symboles 0,1,2,3,4 et on fait des paquets de cinq), ce même 1000 représenterait cent vingt cinq (un paquet de cinq paquets de cinq paquets de cinq unités).

Pour préciser qu'on utilise la notation binaire, on peut convenir d'utiliser une fonte différente pour l'écriture du nombre, comme nous l'avons fait jusqu'ici dans cette fiche. Par ailleurs, pour éviter de confondre à l'oral un nombre et sa représentation dans une base autre que la base dix, il est préférable (mais parfois fastidieux), de l'énoncer chiffre par chiffre : on dirait ainsi « un zéro zéro zéro en base deux » pour 1000, et on réserverait « mille » pour 1000 en base dix, puisque le nombre mille en binaire s'écrit 1111101000.

Nous venons de voir comment trouver la représentation en décimal d'un nombre (représenté en) binaire. Pour effectuer l'opération inverse, c'est-à-dire trouver la représentation binaire d'un nombre écrit en base dix, il suffit de chercher la plus grande puissance de 2 qui est inférieure ou égale à ce nombre, de lui enlever cette puissance de 2 et de recommencer jusqu'à obtenir 0. Toutes les puissances de 2 retranchées au nombre à convertir seront associées à un coefficient de 1. Les autres puissances de 2 dans le nombre converti seront associées à un coefficient de 0.
Par exemple, nous souhaitons écrire mille (1000) en binaire. La plus grande puissance de 2 qui est inférieure à 1000 est $512 = 2^9$, puisque $2^{10} = 2 \times 512 = 1024$ est trop grand. On sait donc que le coefficient associé à 2^9 dans la représentation binaire de mille sera 1. On retranche ensuite 512 à 1000, ce qui donne $1000 - 512 = 488$, et on recommence le processus avec 488 jusqu'à ce qu'on obtienne 0 :
$1000 - 512 = 488$ avec $2^9 = 512$
$488 - 256 = 232$ avec $2^8 = 256$
$232 - 128 = 104$ avec $2^7 = 128$
$104 - 64 = 40$ avec $2^6 = 64$
$40 - 32 = 8$ avec $2^5 = 32$
$8 - 8 = 0$ avec $2^3 = 8$
La représentation de mille en base deux est donc 1111101000.

2.2 Mémoire d'un ordinateur : du bit à l'octet

Puisqu'un ordinateur ne manipule en fin de compte que des mots composés de 0 et de 1, on dit que le bit est l'unité de base de sa mémoire. En très simplifié, on peut se représenter la mémoire d'un ordinateur comme un tableau ayant une seule ligne et autant de cases qu'il peut mémoriser de bits. Par exemple, on peut imaginer la mémoire d'un ordinateur pouvant stocker huit bits ainsi :

Portion de mémoire stockant la donnée 147 :

1	0	0	1	0	0	1	1

Cependant, il est clair que disposer de dix bits de mémoire est largement insuffisant pour pouvoir stocker des données intéressantes. Dans la section précédente, nous avions besoin de onze bits pour représenter le nombre 2014 par 11111011110. Comme 2014 reste un nombre assez petit, on peut imaginer qu'il faudra beaucoup plus de place dans la mémoire d'un ordinateur.

Ainsi, pour faciliter la lecture par les humains de ces nombres dont la représentation en binaire est très longue, on a décidé de grouper les bits par paquets de huit. On appelle « octet » un groupe de

huit bits. Attention, un octet se dit « byte » en anglais : un byte est un groupe de huit bits, et non pas un seul bit.

Lorsque l'ordinateur mémorise un nombre, il lui réserve toujours un nombre entier d'octets, en comblant les cases inutilisées à gauche de la représentation du nombre par des 0. Ainsi pour stocker 111 11011110, qui est la représentation binaire de 2014, un octet ne peut pas suffire, puisqu'il y a onze bits. On utilise donc deux octets c'est-à-dire seize bits, et on écrit en fait 00000111 11011110. Sur deux octets, c'est-à-dire seize bits, on peut représenter $2^{16} = 65536$ nombres différents : le plus petit d'entre eux est représenté par 00000000 00000000, c'est le nombre 0, et le plus grand est représenté par 11111111 11111111, c'est le nombre 65535. En résumé, si l'on dispose de deux octets (16 bits) en mémoire, on peut stocker toute valeur comprise entre 0 et $2^{16} - 1 = 65535$.

En conclusion, il faut retenir que les nombres manipulés par un ordinateur sont limités par le nombre d'octets utilisés pour stocker leur représentation binaire.

2.3 Les unités

L'octet est la principale unité de mesure de la taille de la mémoire d'un ordinateur. Plus précisément, la taille de la mémoire est donnée par un nombre d'octets qui est généralement une puissance de deux, ou un multiple d'une puissance de deux. Ainsi une mémoire de 4 Giga octets correspond-elle à $4 \times 2^{30} = 4 \times 1\ 073\ 741\ 824 = 4\ 294\ 967\ 296$ octets.

Toutefois, le préfixe Giga mentionné ci-dessus est en réalité un abus de langage. En effet, Giga et les autres préfixes courants (Kilo, Méga, ...) sont normalement des raccourcis pour exprimer des puissances de dix et non des puissances de deux. On les utilise cependant pour désigner des puissances de deux en jouant sur les approximations donnée par le tableau suivant :

Puissance de 2	Approximation par une puissance de 10	Préfixe utilisé
$2^{10} = 1024$	$10^3 = 1000$	Kilo
$2^{20} = 1048576$	$10^6 = 1000000$	Méga
$2^{30} = 1073741824$	$10^9 = 1000000000$	Giga
$2^{40} = 1099511627776$	$10^{12} = 1000000000000$	Tera
$2^{50} = 1125899906842624$	$10^{15} = 1000000000000000$	Peta

Pour la petite histoire, certains fabricants de disques durs jouent (à leur avantage) sur cette approximation en vendant par exemple un disque dur d'1 To (Tera octet) contenant précisément 10^{12} octets, et non 2^{40}. Ainsi le client perd 99 511 627 776 octets, soit plus de 99 Go (Giga octets).

3 Représentation des entiers relatifs et des nombres à virgule

Les nombres entiers relatifs sont les nombres entiers positifs ou négatifs : $0, 1, 2, 3, 4$, etc., mais aussi $-1, -2, -3, -4$, etc.

3.1 Prise en compte du signe des entiers

Le signe d'un nombre peut prendre exactement deux valeurs (positif ou négatif), par conséquent, il suffit d'un bit supplémentaire pour le représenter. On convient que 0 correspond à un nombre positif et 1 à un nombre négatif.
Ainsi, la façon la plus simple de représenter un entier relatif en binaire consiste à faire précéder la représentation en binaire de sa valeur absolue (c'est-à-dire sans le signe) d'un 0 s'il est positif et d'un 1 s'il est négatif. Avec cette méthode, si on utilise quatre bits par nombre, 0101 représente 5 et 1101 représente -5.
Lorsque les nombres occupent (par exemple) deux octets, on utilise alors un bit pour le signe et les quinze autres bits pour la valeur absolue. Dans ce cas, le plus petit nombre (négatif) qu'on puisse représenter est 11111111 11111111, c'est-à-dire $-(2^{15}-1) = -32767$, et le plus grand (positif) est 01111111 11111111, c'est-à-dire $+(2^{15}-1) = +32767$.
Cependant, il y a dans ce cas deux façons d'écrire zéro, l'une comme un nombre positif (00000000 00000000) et l'autre comme un nombre négatif (10000000 00000000). Ce n'est évidemment pas souhaitable, et comme il y a aussi d'autres inconvénients sur lesquels nous ne nous étendrons pas, cette méthode simple n'est pas celle qui est utilisée dans la plupart des ordinateurs.

La méthode utilisée dans les ordinateurs ne diffère de la précédente que pour les nombres négatifs. Commençons par un petit exemple, où on n'utilise que quatre bits par nombre. Les huit entiers positifs ou nuls de zéro à sept, sont représentés simplement en binaire par 0000, 0001, ..., 0110, 0111, comme on vient de le voir : le premier bit 0 représente le signe et les trois bits suivants représentent la valeur absolue du nombre.
Les huit entiers strictement négatifs de -8 à -1 sont représentés respectivement par 1000, 1001, ..., 1110 et 1111. Notons que le premier bit est bien un 1. Plus précisément, la représentation du nombre négatif x est celle de l'entier naturel $2^4 + x$: pour représenter -8, on calcule $2^4 - 8 = 16 - 8 = 8$, et on a vu à la Section 2 que 8 en base dix est représenté par 1000 en base deux, donc maintenant la représentation de -8 est 1000. De même, la représentation de -1 est 1111 car on a $2^4 - 1 = 16 - 1 = 15$, qui s'écrit 1111 en base deux.
Avec cette méthode, appelée la notation en complément à deux, on a une légère asymétrie, puisque les nombres représentés vont de -8 à -1 pour les négatifs, puis 0 , et de 1 à 7 pour les positifs. On n'a plus qu'une seule façon d'écrire zéro, c'est 0000. Les représentations des nombres négatifs sont reconnaissables au fait qu'elles commencent par un 1, tandis que celles des nombres positifs ou nuls commencent par un 0.

Maintenant, si on se place dans le cas plus général où chaque nombre occupe k octets ($8 \times k$ bits), la notation en complément à deux permet de représenter les nombres positifs ou nuls de 0 à $2^{8k-1} - 1$ et les nombres strictement négatifs de -2^{8k-1} à -1 :

- Il n'y a qu'une seule façon d'écrire zéro, c'est 00000000 ... 00000000.
- Un entier relatif strictement positif x compris entre 1 et $2^{8k-1} - 1$ est représenté par l'entier naturel x. Sa représentation commence par un 0.
  Par exemple, sur 2 octets, 2014 est représenté par 00000111 11011110.

- Un entier relatif strictement négatif x compris entre -2^{8k-1} et -1 est représenté par l'entier naturel $2^{8k}+x$, qui, pour sa part, est compris entre $2^{8k}-2^{8k-1}=2^{8k-1}$ et $2^{8k}-1$. Cette représentation commence par un 1.
  Par exemple, sur deux octets, -2014 est représenté par 11111000 00100010, qui est la notation en base deux de l'entier naturel $2^{16}-2014=65536-2014=63522$.

3.2 Les nombres à virgule

Comme dans le cas des entiers relatifs, nous allons voir deux façons de représenter en binaire les nombres à virgule.
En notation en base dix, les chiffres à gauche de la virgule représentent des unités, des dizaines, des centaines etc., tandis que les chiffres à droite de la virgule représentent des dixièmes, des centièmes, des millièmes, etc. Par exemple, le nombre $20,14$ correspond à deux dizaines, zéros unités, un dixième et quatre centièmes. Autrement dit, on a : $20,14=2\times 10+0\times 1+1\times 0,1+4\times 0,01=2\times 10^1+0\times 10^0+1\times\frac{1}{10^1}+4\times\frac{1}{10^2}$.
On peut faire exactement la même chose en base deux : le nombre $101,011$ représente une quatraine, zéro deuzaine, une unité, zéro demi, un quart et un huitième, soit $101,011$ en base deux représente $4+1+\frac{1}{4}+\frac{1}{8}=5,375$ en base dix.
Avec cette notation, on utilise un signe particulier pour la virgule, et ce n'est pas possible en informatique puisque les seuls signes dont nous disposons pour écrire sont 0 et 1. Pour remédier à ce problème, on peut décider de fixer la position de la virgule à l'avance, par exemple trois bits à gauche et trois bits à droite, si on utilise six bits par nombre. On dit qu'il s'agit d'une représentation en « virgule fixe ». Avec cette convention, on peut écrire le nombre $101,011$, mais pas le nombre $1010,11$, puisqu'il nécessite quatre bits à gauche de la virgule. De plus, si on veut écrire des nombres très grands ou très proches de zéro (par exemple pour le calcul scientifique), on a des notations très longues, et on sait que dans un ordinateur, la place dont on dispose pour représenter un nombre est limitée à un nombre fixe d'octets. C'est pourquoi on utilise plus souvent en informatique un autre format de représentation des nombres à virgule.

La représentation binaire des nombres à virgule utilisée en informatique est du même type que la notation dite « scientifique » utilisée par les calculatrices : le nombre $201,4$ s'écrit ainsi $2,014E2$ puisque $201,4=2,014\times 10^2$. L'exposant peut-être négatif si le nombre est compris entre -1 et 1 : par exemple $0,00214$ s'écrit $2,014E-3$, car il est convenu de noter 10^{-3} pour $\frac{1}{10^3}$. La mantisse peut-être précédée du signe $-$ si le nombre est négatif : par exemple $-201,4$ s'écrit $-2,014E2$.

En binaire, on peut adopter une notation similaire, en écrivant la mantisse et l'exposant en base deux. Commençons de nouveau par un petit exemple : on utilise un bit pour le signe du nombre (0 pour un nombre positif et 1 pour un nombre négatif), quatre bits pour l'exposant (un entier relatif représenté en complément à deux) et cinq bits pour la mantisse (un nombre à virgule avec un seul bit à gauche de la virgule, compris entre un inclus et deux exclu), soit dix bits en tout. Le nombre représenté par 1 0101 11100 est négatif (le premier bit vaut 1), il a pour exposant 5 (les quatre bits suivants sont 0101) et pour mantisse $1,1100$ en base deux, soit $1+\frac{1}{2}+\frac{1}{4}=1,75$ en base dix, c'est

donc le nombre $-1,75 \times 2^5 = -56$. De même, le nombre représenté par 0 1101 11100 est positif, son exposant est négatif et vaut $2^4 - 13 = -3$, et sa mantisse est de nouveau $1,1100$ en base deux, c'est-à-dire $1,75$ en base dix, c' est cette fois le nombre $1,75 \times 2^{-3} = 1,75 \times \frac{1}{8} = 0,21875$

La représentation des nombres à virgule communément utilisée en informatique (norme IEE 754) est une variante de la précédente, qu'on appelle la représentation en « virgule flottante ». On se place maintenant dans le cas où chaque nombre occupe huit octets, soit 64 bits :

- On réserve toujours le premier bit pour le signe (0 pour un nombre positif et 1 pour un nombre négatif).
- Les onze bits suivants sont utilisés pour l'exposant, qui est un nombre relatif compris entre -1022 et 1023. Pour faciliter la comparaison entre deux nombres à virgule, on représente en fait l'exposant x comme l'entier naturel $x + 1023$, qui est compris entre 1 et $2046 = 2^{11} - 2$ (donc représentable sur onze bits en base 2). On réserve les valeurs 0 et 2047 pour des cas particuliers (voir ci-dessous).
- Il reste 52 bits pour la mantisse. Comme le premier bit est forcément un 1, on ne l'écrit pas, et on utilise l'ensemble des 52 bits après la virgule. La mantisse est donc un nombre binaire à virgule compris entre 1 inclus et 2 exclu.
- On garde à part quatre valeurs particulières :
  - 0 00000000000 0000000000000000000000000000000000000000000000000000 et 1 00000000000 0000000000000000000000000000000000000000000000000000 représentent tous deux zéro ;
  - 0 11111111111 0000000000000000000000000000000000000000000000000000, qu'on note $+\infty$, représente une valeur trop grande pour être représentée ;
  - 1 11111111111 0000000000000000000000000000000000000000000000000000, qu'on note $-\infty$, représente une valeur négative trop petite pour être représentée ;
  - 1 11111111111 1111111111111111111111111111111111111111111111111111, qu'on note NaN (not a number), représente une valeur indéfinie.
- Notons que certains mots de 64 bits ne sont pas utilisés du tout : ce sont ceux qui ont un exposant 00000000000 ou 11111111111 et qui ne sont pas parmi les quatre valeurs particulières.

4 Représentation des textes

Pour terminer, nous nous intéressons à la représentation des textes. Un texte est vu comme une suite de caractères qui ont chacun un code numérique, c'est-à-dire un nombre correspondant. Ensuite, il suffit de représenter ce code numérique en binaire pour obtenir la représentation du caractère.

Les caractères sont non seulement les lettres de l'alphabet, majuscules et minuscules, mais aussi les signes de ponctuation, l'espace, les chiffres, les parenthèses, etc. En français, il faut encore ajouter les lettres accentuées, le ç, le œ, et d'autres caractères spéciaux pour d'autres langues (ñ ou ŏ). Il y a aussi des langues qui n'utilisent pas l'alphabet latin, comme le grec, le russe, le chinois, le

japonais, le coréen, l'arabe...
Le format Unicode est un format universel permettant de représenter des textes écrits dans n'importe quelle langue. Il utilise jusqu'à 32 bits par caractère et permet d'en représenter plus de 110000 différents. Il existe plusieurs versions d'Unicode, parmi lesquelles la norme appelée UTF-8 est destinée à être utilisée sur l'ensemble des ordinateurs de la planète. Ce n'est malheureusement pas encore tout-à-fait le cas, et on rencontre de temps en temps des problèmes d'encodage des caractères lorsqu'on tente de lire un texte en utilisant un format différent de celui avec lequel il a été représenté. Par exemple, dans un texte écrit à l'aide du format ISO-8859-1 (que nous n'évoquerons pas plus avant) et lu à l'aide du format UTF-8, tous les « é » seront remplacés par des « Ã© », et ainsi de suite.

Nous présentons ci-dessous un format, qui est de moins en moins utilisé, mais qui a l'avantage d'être simple à comprendre : le code ASCII (American Standard Code for Information Interchange). Dans la version la plus simple, chaque caractère est codé sur 7 bits (en réalité sur un octet, mais avec le premier bit toujours à 0). Il y a donc $2^7 = 128$ caractères possibles, correspondant aux nombres de 0 à 127. La liste de ces 128 caractères est donnée dans le tableau de la Figure 1.
On observe que les lettres majuscules A, B, C, ..., Z sont codées par les nombres consécutifs 65, 66, 67, ..., 90. Les lettres minuscules a, b, c, ..., z correspondent aux nombres 97 à 122. Il est intéressant de noter que les chiffres de 0 à 9 sont codés par les nombres de 48 à 57. Les codes de 0 à 31 ne correspondent pas à des caractères du clavier, mais à des symboles de mise en page comme le retour à la ligne (« CARRIAGE RETURN »).
Par exemple, l'expression « Le code ASCII » se code numériquement par 76 101 32 99 111 100 101 32 65 83 67 73 73, et en binaire elle devient 01001100 01100101 01000000 01100011 01101111 01100100 01100101 01000000 01000001 01010011 01000011 01001001 01001001.
On note que ce format très limité ne permet pas de représenter tous les caractères utilisés en français, puisqu'il est limité aux 26 lettres majuscules et minuscules.

5 Conclusion

Nous avons vu dans cette fiche quelques principes de base de la représentation de l'information dans un ordinateur, quand cette information est de type numérique ou textuelle.
Le domaine de l'informatique appelé « arithmétique des ordinateur » est celui qui étudie les différentes façons de représenter les nombres en informatique, et les meilleures façons de calculer selon la représentation choisie.
Pour les lecteurs souhaitant aller plus loin, il existe de nombreux ouvrages destinés aux étudiants des premières années d'université traitant de ce sujet, et on trouve aussi facilement des cours en accès libre sur internet. Comme ouvrage de référence en français sur ce thème, on peut par exemple citer [1] (voir ci-dessous).

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

FIGURE 1 – Les codes ASCII en décimal (base 10), hexadécimal (base 16), octal (base 8) et binaire (base 2). La dernière colonne donne le caractère correspondant.

Références

[1] J.-C. Bajard et J.-M. Muller (coord.). *Calcul et arithmetique des ordinateurs*. Hermès (Traité IC2), 2004.