Article Tangente Numération
Aurélie Lagoutte

Titre : Représentation des données en binaire

La mémoire d'un ordinateur permet de stocker uniquement des 0 et des 1. Et pourtant, on y enregistre des nombres, des photos, et même des vidéos !

Du système décimal au système binaire

Après les travaux théoriques de Leibnitz en 1703 et Boole en 1854, de solides bases mathématiques faisaient apparaître le système binaire comme puissant système de représentation de données.
De même que le système décimal désigne l'écriture des nombres à partir de 10 chiffres différents (0, 1, …. , 9), le système dit *binaire* tient son nom du fait qu'une unité élémentaire d'écriture ne peut avoir que deux valeurs différentes: 0 ou 1. Un chiffre binaire est appelé un *bit,* contraction de l'anglais *binary digit*.

C'est Shannon qui, en 1937, fait passer le système binaire de la théorie vers la pratique : grâce à des relais, interrupteurs et fils électriques, il crée des circuits électroniques contenant toute la puissance du système binaire découvert par ses aïeux : en se fixant une tension "seuil", il est possible de distinguer les fils où passe le courant (tension au-dessus du seuil) de ceux où il ne passe pas. Les premiers transmettent le bit 1 alors que les seconds transmettent le bit 0.

Mais alors, comment trouver un sens à tous ces 0 et ces 1 ? Tout simplement grâce à des conventions adoptées par toute la communauté informatique. Tout comme les espaces et la ponctuation permettent de délimiter les mots et les phrases, les ensembles de bits sont divisés en "blocs" de taille fixe sur chaque machine. Un bloc de 8 bits est appelé un octet. Lire un octet en mémoire est similaire à lire un mot : il faut savoir dans quelle "langue", selon quelle convention, il a été écrit.

Représentation des nombres

La convention utilisée pour représenter les entiers positifs est très largement inspirée du système décimal : lorsque 2456 représente la valeur 2x 1000 + 4 x 100 + 5 x 10 + 6 + 1, la suite de bits 1101 représente la valeur 1 x 8 + 1 x 4 + 0 x 2 + 1 x 1 = 13. A chaque position correspond un poids qui est une puissance de la base utilisée (10 pour le décimal, 2 pour le binaire) ; plus la position est vers la gauche, plus l'exposant est élevé ; le chiffre présent à chaque position indique un facteur par lequel il faut multiplier le poids avant de le sommer avec le reste. Il est ainsi possible de représenter n'importe quel entier positif en binaire,

même si le nombre de bits à utiliser sera souvent bien supérieur au nombre de chiffres nécessaires en décimal : sur n bits, il existe 2^n combinaisons différentes, qui encodent donc les entiers de 0 à 2^{n-1}. Par exemple, un octet peut représenter n'importe quel nombre entre 0 et 255 ; alors qu'en décimal, la limite de 8 chiffres nous permet tout de même d'écrire tous les entiers de 0 à 99 999 999 !

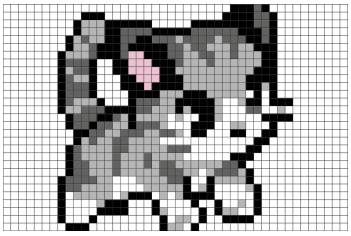
Bien évidemment, n'importe quel ordinateur peut également travailler avec des entiers relatifs. Mais comment sont-ils encodés ? Contrairement au cas des entiers positifs, aucune convention n'a réussi à cumuler tous les avantages souhaités, par conséquent plusieurs conventions co-existent. La plus simple consiste à rajouter un bit supplémentaire à gauche du nombre pour indiquer si le signe est négatif (avec un 1) ou positif (avec un 0). Ainsi, pour écrire -70, il suffit de commencer par écrire 1 comme bit de signe, suivi de l'écriture de 70 en binaire : 1100 0110. Cette convention est très simple à comprendre, mais malheureusement elle rend les additions très compliquées. En pratique, c'est donc souvent une autre convention plus adaptée aux calculs, appelée notation en Complément à 2, qui est utilisée.

En ce qui concerne les nombres réels, c'est une norme appelée IEEE 754 qui s'est imposée. Elle s'inspire de la notation scientifique : l'écriture $-2{,}347 \times 10^2$ pour désigner -234,7 est imitée par l'écriture $-1{,}01101 \times 2^3$ pour désigner -1011,01. Tout comme le chiffre des dixièmes en décimal représente le poids 10^{-1}, le premier bit après la virgule représente quant à lui le poids 2^{-1}, le suivant 2^{-2}, etc…. Sur l'exemple précédent, 1011,01 vaut donc $1 \times 2^4 + 0 \times 2^3 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 11{,}25$. L'écriture $-1{,}01101 \times 2^3$ permet de décomposer le nombre en 3 parties : le signe, codé sur 1 bit comme précédemment ; l'exposant, ici égal à 3, souvent codé sur 8 bits ; et la mantisse, ici 1,01101 , souvent codée sur 23 bits. Cela permet d'obtenir une granularité de précision proportionnelle à l'ordre de grandeur du nombre représenté. Cette écriture est appelée "en virgule flottante" car la virgule est déplacée par rapport à sa position initiale et c'est l'exposant qui compense ce déplacement ; et par abus de langage, le mot *flottant* est souvent utilisé pour parler d'un nombre réel en informatique.

Les autres données : texte, image, …

Parmi les données les plus essentielles à stocker, outre les nombres, se trouve le texte, qui n'est finalement rien d'autre qu'une suite de caractères (lettre, ponctuation, etc …). L'écriture d'un texte en binaire relève d'une méthode des plus élémentaires : à chaque caractère est associé un numéro, via une table commune de correspondance entre caractère et numéro, puis lors de l'encodage d'un texte, chaque caractère est remplacé par l'écriture binaire de son numéro. La table de correspondance la plus connue est l'ASCII. Par l'exemple, le numéro du caractère 'A' dans la table ASCII est 65, son écriture sur 7 bits est donc 100 0001. Les codes ASCII n'utilisant que 7 bits, seul un jeu restreint de 128 (soit 2^7) caractères fait partie de la table : lettres majuscules et minuscules, signes de ponctuation, quelques symboles mathématiques et quelques caractères spéciaux (par exemple, le retour à la ligne). Sous forte influence anglophone, cela était suffisant dans les premières décades de l'informatique. Cependant l'existence de caractères propres à certaines langues (ex : *à, ç …*) ont poussé à étendre la table ASCII sous diverses déclinaisons nationales, provoquant par la suite de nombreux problèmes de décodage : qui n'a jamais vu un affichage erroné des caractères accentués, comme par exemple *goÃ»tÃ©* à la place de *goûté* ? Ces dernières années, c'est le standard international Unicode qui est en train de s'imposer comme table

universelle, visant à pouvoir représenter n'importe quel caractère de n'importe quel système d'écriture.



Quelle que soit l'information à représenter grâce à une séquence finie de bits, il suffit donc de mettre en place une convention de représentation qui est connue par tous les interlocuteurs : ainsi, une image est découpée en pixels, petits carrés de couleur unie. Il suffit donc de s'accorder au départ sur la numérotation des couleurs et sur l'ordre dans lequel on écrira le code couleur de chaque pixel. Ainsi, il est possible d'envoyer une image sous la forme d'une suite de bits. Cette méthode se généralise naturellement aux vidéos, qui sont des suites finies d'images.

Ainsi, le système binaire s'est imposé en informatique : il permet de coder tous les formats de données actuels malgré les contraintes technologiques des systèmes de mémoire.

Le saviez-vous ?

Un QR code est tout simplement une représentation visuelle d'une suite de bits : chaque pixel encode une information binaire (blanc ou noir). Flasher un QR Code, c'est demander à la caméra de votre smartphone de lire en une fraction de seconde plusieurs dizaines de caractères (et même jusqu'à plusieurs milliers dans la version la plus récente !), qui forment généralement une URL. C'est plus rapide que de la recopier à la main !

Kilo-octet ? Kibi-octet ?

Dans les premières heures de l'informatique, des préfixes de mesure du système décimal ont été utilisées pour désigner des tailles de mémoire, bien que celles-ci soient en réalité dimensionnées par des puissances de 2.
Ainsi, un ensemble de 1024 ($=2^{10}$) octets fut appelé un Kilo-octet, par analogie d'ordre de grandeur avec les kilomètres, kilogrammes, etc… De même, un ensemble de 1024 Kilo-octets fut appelé Méga-octet, et ainsi de suite pour les Giga-octets et Tera-octets.
Pour mettre fin à cette ambiguïté, la Commission électronique internationale introduit en 1998 de nouveaux préfixes pour les puissances de 2 : un Kibi-octet pour désigner 2^{10} = 1024 octets, un Mébi-octet pour 2^{20} octets, et ainsi de suite pour Gibi-octet et Tébi-octet. Les préfixes décimaux, désormais voués à disparaître, retrouvent une signification cohérente avec les autres unités (un Kilo-octet = 1000 octets, etc…).
Malheureusement, les préfixes décimaux sont en pratique encore régulièrement utilisés, que ce soit dans leur ancienne ou nouvelle signification, ce qui entretient la confusion auprès du grand public.