Représentation des entiers en machine

1 Notions utiles

On a compris que, dans un ordinateur, toutes les informations sont représentées en chiffres binaires (0 ou 1). On les appelle des bits (pour *binary digits* en anglais).

Ils sont regroupés par paquets de 8 qu'on appelle des octets (*bytes* en anglais), puis sont organisés en mots (*words* en anglais) de 2, 4 ou 8 octets.

Par exemple : une machine en 32 bits manipule des mots de 4 octets (8 x 4 = 32 bits) pour effectuer des opérations.

Ces regroupements permettent de représenter autre chose que des 0 ou des 1 comme des nombres entiers, des (approximations de) nombres réels, des caractères alpha-numériques.

Il est nécessaire de définir des encodages pour décrire ces représentations.

2 Codage des entiers non-signés

Les entiers naturels sont usuellement représentés par des mots de 16 bits (2 octets) ou 32 bits (4 octets).

Les entiers naturels sont codés en binaire naturel, en complétant éventuellement avec des zéros à gauche.

Par exemple, sur 8 bits, le nombre 50 est codé par l'octet : $0011\ 0010$.

Le nombre 0 est codé par l'octet : $0000\ 0000$.

Le plus grand entier représentable sur huit bits est 255, codé par l'octet $1111\ 1111$.

Un octet permet donc de réprésenter tous les nombres entiers de 0 à 255.

Sur 16 bits, on peut coder tous les entiers de 0 à $65\,535$.

Sur 32 bits, on peut coder tous les entiers de 0 à $4\,294\,967\,295$.

Et donc plus généralement, sur n bits, on peut coder tous les entiers de 0 à $2^n - 1$.

Exercice 1

Déterminer l'écriture sur 32 bits de 10^6.

Réponse : $0000\ 0000\ 0000\ 1111\ 0100\ 0010\ 0100\ 0000$.

3 Le système hexadécimal

Afin de simplifier les écritures des octets, on utilise la base hexadécimale dont les 16 chiffres sont, dans l'ordre :

$$0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ A,\ B,\ C,\ D,\ E,\ F$$

La correspondance entre systèmes de numération décimale, binaire et hexadécimale est la suivante :

base 10	base 2	base 16	base 10	base 2	base 16
0	0	0	8	1000	8
1	1	1	9	1001	9
2	10	2	10	1010	A
3	11	3	11	1011	B
4	100	4	12	1100	C
5	101	5	13	1101	D
6	110	6	14	1110	E
7	111	7	15	1111	F

Donc en groupant par 4 les bits d'un nombre binaire, on peut trouver sa valeur hexadécimale :

$$\overbrace{1010}^{A}\ \overbrace{0101}^{5}\ \overbrace{1111}^{F}\ \overbrace{0011}^{3}$$

Exercice 2 Compléter le tableau ci-dessous.

Écriture décimale	33	100	106	200	205	255
Écriture binaire	10 0001	110 0100	110 1010	1100 1000	1100 1101	1111 1111
Écriture hexadécimale	21	64	6A	C8	CD	FF

Exercice 3 Déterminer l'écriture hexadécimale sur 32 bits de 10^6. Réponse : 00 0F 42 40

4 L'addition binaire dans un ordinateur

Les micro-processeurs sont tous équipés de circuits permettant d'additionner des octets : ils appliquent l'algorithme d'addition colonne par colonne. Par exemple, sur 8 bits, l'opération $41 + 77 = 118$ est faite de la manière suivante :

$$\begin{array}{rcccccccc} & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ + & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

Problème : le résultat renvoyé par le micro-processeur est faux si la somme est supérieure au plus grand entier représentable. Par exemple, l'addition $255 + 1$ sur 8 bits donne 0.

$$\begin{array}{rcccccccc} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ + & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Cette retenue ne peut pas être prise en compte sur 8 bits

Lorsque la somme dépasse le plus grand entier représentable, le micro-processeur ignore donc la retenue mais il peut néanmoins signaler le problème dans un registre (*overflow flag*, ou *carry flag*).

Exercice 4 Quel est le résultat de l'addition $236 + 50$ renvoyé par un micro-processeur programmé sur 8 bits ?

236 est codé par $1110\,1100$ et 50 par $0011\,0010$.

Effectuons l'addition binaire :

$$\begin{array}{cccccccccc} & & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ + & & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ \hline & (1) & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}$$

En ignorant la retenue qui dépasse, le micro-processeur renvoie le nombre 30.

On peut aussi voir que $236 + 50 = 286$ et que $286 - 30 = 256$.

5 Codage des entiers signés

Codage naïf

Pour représenter les entiers relatifs sur un octet, il est tentant de vouloir attribuer au bit le plus à gauche (appelé bit de poids fort) le rôle du signe : on décide que s'il vaut 0, le nombre codé est positif (et négatif sinon).

On aurait ainsi :

$$20 = 0001\,0100 \qquad \text{et} \qquad -20 = 1001\,0100.$$

Malgré sa simplicité, ce procédé présente deux défauts rédhibitoires.

- Le nombre zéro est représenté de deux manières différentes par les octets $0000\,0000$ et $1000\,0000$.
- L'algorithme d'addition colonne par colonne de fournit pas le bon résultat. Par exemple, les opérations

$$\begin{array}{cccccccccc} & & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ + & & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline & & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{array} \qquad \begin{array}{cccccccccc} & & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ + & & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline & & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{array}$$

  mènent aux égalités (fausses) $20 + (-20) = -40$ et $30 + (-20) = -50$.

Il est donc indispensable de trouver une autre façon de coder les entiers relatifs, et la solution se trouve précisément dans le fait que, sur k bits, la machine ne fait pas la différence entre 2^k et 0.

Méthode n° 1 : le complément à 2

On convient qu'un nombre positif ou mot binaire m est représenté par son écriture binaire sur k bits, et on cherche à coder $-m$ de sorte que le résultat de l'opération $m + (-m)$ soit égal à 0.
Comme, sur k bits, la machine ne fait pas la différence entre 2^k et 0, il suffit que :
$-m$ soit codé par l'écriture binaire de $2^k - m$.

car $m - m = 0 = 2^k$ sur k bits $\implies -m = 2^k - m$

Illustrations

- Sur un octet, on code le nombre -1 par l'écriture binaire de $256 - 1 = 255$, soit $1111\,1111$.

  On a vu que l'opération $1111\,1111$+$0000\,0001$ donne $0000\,0000$, et on a bien $-1 + 1 = 0$.

- Sur un octet, le nombre -20 est codé par l'écriture binaire de $256 - 20 = 236$, soit $1110\,1100$.

  Vérifions les calculs $20 + (-20)$ et $30 + (-20)$:

$$\begin{array}{cccccccccc} & & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ + & & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline & (1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \qquad \begin{array}{cccccccccc} & & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ + & & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline & (1) & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}$$

Conclusions sur le complément à 2

Si le bit de poids fort vaut 1, alors l'entier représenté est négatif.
Sur $k = 8$ bits, le plus grand entier positif a donc pour code $0111\,1111$, soit 127 .
On peut ainsi coder sur 8 bits tous les entiers relatifs de -128 à 127 .
De manière générale, on peut coder sur k bits tous les entiers relatifs de -2^{k-1} à $2^{k-1} - 1$.

Sur 8 bits, le tableau ci-dessous donne les valeurs des entiers signés représentés par les octets.

Octet	Valeur	Octet	Valeur
$0000\,0000$	0	$1000\,0000$	-128
$0000\,0001$	1	$1000\,0001$	-127
$0000\,0010$	2	$1000\,0010$	-126
$\dots$	$\dots$	$\dots$	$\dots$
$0111\,1110$	126	$1111\,1110$	-2
$0111\,1111$	127	$1111\,1111$	-1

Exercice 5

Décoder le nombre entier relatif représenté par l'octet $1010\,1111$.

L'octet $1010\,1111$ représente un nombre négatif car le bit de poids fort vaut 1.
On a $1010\,1111_2 = 175$.

C'est l'écriture binaire de $2^8 + (-m)$, donc le nombre cherché est : $175 - 256 = -81$.

Exercice 6 – *(Une autre méthode)*

1. Coder sur un octet les nombres -17 et -84.

   -17 est codé par l'octet $1110\,1111$ -84 est codé par l'octet $1010\,1100$

2. Déterminer l'écriture binaire de 17, 84, ainsi que de $255 - 17 = 238$ et $255 - 84 = 171$.

$$17 = 0001\,0001 \qquad 84 = 0101\,0100$$
$$238 = 1110\,1110 \qquad 171 = 1010\,1011$$

3. En déduire ce qui suit :

Méthode n° 2 : l'inversion de bits

Le code sur k bits d'un entier positif $m \leq 2^{k-1}$ est son écriture binaire.
Pour coder sur k bits un entier négatif $-m \geq 2^{k-1} - 1$, il suffit :

- d'écrire m en binaire ;
- d'inverser tous les bits ;
- d'ajouter 1.

Exercice 7

Coder sur $k = 16$ bits le nombre -2017.

$2017 = 0000\,0111\,1110\,0001_2$

On inverse les bits : $1111\,1000\,0001\,1110$

On ajoute 1 : $1111\,1000\,0001\,1111$

Exercice 8

Coder sur $k = 16$ bits le nombre -32.

$32 = 0000\,0000\,0010\,0000_2$

On inverse les bits : $1111\,1111\,1101\,1111$

On ajoute 1 : $1111\,1111\,1110\,0000$

Exercice 9

Coder sur un octet les nombres -100 et -64 et effectuer l'opération machine $-100 + (-64)$. Que se passe-t-il ?

-100 est codé par l'octet $1001\,1100$ -64 est codé par l'octet $1100\,0000$

$$\begin{array}{rcccccccc} & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ + & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline (1) & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{array}$$

La machine donne $+92$ comme résultat de l'opération $-100 + (-64)$: le problème vient du fait que -164 ne peut être codé sur 8 bits. La machine génère un *overflow flag*.