

TD 1 : Représentation de données

Introduction : on peut avoir différents types de données à traiter, exemples :

- Alphabétique
- Numérique
- Analogique

On remarque que certaines données ne sont pas des nombres, or les données traitées par un ordinateur sont toujours numériques. Il faudra donc convenir d'un certain codage par exemple pour les lettres et caractères spéciaux.

D'autre part, certaines données sont discrètes et finies : lettre, nombre entier dans un certain intervalle. D'autres données sont continues : réel dans un certain intervalle, signal analogique (tension électrique proportionnelle à l'amplitude du phénomène physique correspondant : voix, température...), on aura alors une représentation approximative de ces données puisqu'elles peuvent prendre une infinité de valeurs alors que l'on ne dispose que d'un espace mémoire fini.

N.B. : pour représenter un signal analogique, il faut le convertir en numérique, ceci est réalisé par des composants électroniques appelés ADC (Analogic Digital Converter), ils procèdent en plusieurs étapes :

- Échantillonnage
- Quantification
- Codage

En informatique, on utilise la numération binaire (0 ou 1).

A) Numération binaire, hexadécimale

Rappel : Numération de base a

Soit a un entier naturel, $a \geq 2$ et $A = \{e_0, e_1, \dots, e_{a-1}\}$ un alphabet. Chaque élément de A est appelé chiffre. Pour tout nombre entier p, il existe un unique n-uplet de chiffres $(\alpha_n, \dots, \alpha_0)$ tels que

$$p = \sum_{i=0}^n \alpha_i a^i .$$

$\alpha_n \dots \alpha_0$ est l'écriture de p en base a, on note aussi $p = \overline{\alpha_n \dots \alpha_0}^a$

Exemples :

En base 10, $A = \{0, 1, \dots, 9\}$, $9738 = \overline{9738}^{10} = 9 \times 10^3 + 7 \times 10^2 + 3 \times 10^1 + 8 \times 10^0$

En base 2, $A = \{0, 1\}$, $\overline{10011}^2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 0 + 0 + 2 + 1 = 19$

Exercice I

a) Convertir en nombre décimaux les nombres binaires suivants : 1110, 1111011, 10011111.

b) Convertir en binaire les nombres de la base 10 suivants (on écrira sur un octet=8 bits, exemple $10 = 8 + 2 = \overline{00001010}^2$) : 15, 29, 166.

c) Quel est le plus grand octet que l'on peut coder sur un octet? Deux octets?

N.B. : Il existe un algorithme pour convertir un décimal en binaire : on effectue les divisions successives par 2 et on s'arrête au premier quotient nul, par exemple pour 13 :

$13 : 2 = 6$ reste 1

$6 : 2 = 3$ reste 0

$3 : 2 = 1$ reste 1

$1 : 2 = 0$ reste 1

On lit le nombre en base 2 en remontant les restes : 1101.

Rappel : Numération hexadécimale, $a=16$, on prend $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$.

Donc A vaut 10, B vaut 11, C vaut 12, D vaut 13, E vaut 14, F vaut 15.

Exemple : $\overline{35C}^{16}$ (avec barre au dessus) $= 3 \times 16^2 + 5 \times 16^1 + 12 \times 16^0 = 768 + 80 + 12 = \overline{860}^{10}$

Exercice II :

- a) Convertir en décimaux les hexadécimaux suivants : 75, A8, DE, FF.
b) Convertir en hexadécimaux les décimaux suivants : 18, 103, 216.

B) Opérations élémentaires en binaire

Les opérations élémentaires s'effectuent en binaire de façon analogue à ce que l'on pratique en base 10, on remarquera que : $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, $1 + 1 = 10$, $0 \times 0 = 0$, $0 \times 1 = 1 \times 0 = 0$, $1 \times 1 = 1$.

Exercice III

Effectuer les opérations suivantes en binaire (pour la division, il s'agit ici de la division entière i.e. Avec un quotient et un reste entiers) puis vérifier le résultat en transformant les nombres en décimaux.

- a) $110+101$, $1001+11111$, $111010+1010$.
b) $101-10$, $11011-111$, $11001-1111$.
c) 101×11 , 1101×101 .
d) $1101:10$, $1110:11$, $10111:11$.

C) Représentations d'un entier signé

Il y a plusieurs façon de représenter un entier relatif:

- *signe et valeur absolue : un bit sert à coder le signe, 0 pour +, 1 pour -, si le nombre est représenté sur un octet, -12 sera codé 10001100.*
- *complément logique (appelé aussi complément restreint ou complément à 1) : pour les nombres négatifs, on remplace 0 par 1 et 1 par 0, si le nombre est représenté sur un octet, -12 sera codé 11110011.*
- *complément à 2 (appelé aussi complément arithmétique ou complément vrai) : pour les nombres négatifs, on rajoute 1 au complément à 1, si le nombre est représenté sur un octet, -12 sera codé 11110100. On remarquera que cela correspond à $2^8+(-12)=256-12=244=128+64+32+16+4=11110100^2$.*

Les deux premières représentations ont l'inconvénient d'avoir deux codages pour 0 (par exemple sur un octet 00000000 et respectivement 10000000, 11111111), sur n bits on représente ainsi les nombres N tels que $-(2^{n-1}-1) \leq N \leq 2^{n-1}-1$ soit 2^n-1 valeurs. La première représentation est difficile à utiliser pour les opérations car il faut traiter à part le bit de signe, en général on utilise les représentations par compléments.

Exercice IV

- a) Additionner les représentations en complément à deux sur un octet de 12 et -12.
b) Représenter sur un octet suivant les trois méthodes les nombres décimaux suivants : 45, -45, -107, -124.

D) Nombres fractionnaires

En binaire, les chiffres après la virgule correspondent aux puissances négatives de 2, par exemple : $0,0111^2=0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} = 0 + 0 + 0,25 + 0,125 + 0,0625 = 0,4375$.

Pour convertir un décimal inférieur à 1 en binaire, on effectue les multiplications successives par 2 comme ci-dessous et on s'arrête lorsque la partie fractionnaire est nulle ou lorsque le nombre de bits correspond à la taille voulue, par exemple pour convertir 0,8125 :

$$0,8125 \times 2 = 1,625 = 1 + 0,625$$

$$0,625 \times 2 = 1,25 = 1 + 0,25$$

$$0,25 \times 2 = 0,5 = 0 + 0,5$$

$$0,5 \times 2 = 1 = 1 + 0,0$$

On lit les parties entières de haut en bas : $0,8125 = 0,1101^2$.

Si on veut un codage de la partie fractionnaire sur 4 bits pour convertir 0,6928 :

$0,6928 \times 2 = 1,3856 = 1 + 0,3856$
 $0,3856 \times 2 = 0,7712 = 0 + 0,7712$
 $0,7712 \times 2 = 1,5424 = 1 + 0,5424$
 $0,5424 \times 2 = 1,0848 = 1 + 0,0848$
 d'où $0,6928 \approx 0,1011^2$ ($0,1011^2 = 0,6875$).

Exercice V

- Convertir $0,01011^2$ en décimal
- Convertir 0,71875 et 0,40871 en binaire avec la partie fractionnaire sur 5 bits.

Au niveau de la machine la virgule n'existe pas, elle est gérée par le programmeur de façon virtuelle. La virgule peut être fixe (on convient par exemple de coder la partie fractionnaire sur un octet) ou flottante, on représente alors le nombre N sous la forme $N = M \times 2^E$ où M est la mantisse normalisée (i.e. $1 \leq |M| < 2$), E est l'exposant (un entier relatif). L'exposant est souvent représenté sans signe, par exemple sur 4 bits on peut représenter les entiers de 0 à 15, on décide que 0 représente en fait -8, 1 représente -7, ..., 15 représente 7, on dit que l'exposant est biaisé ou décalé (ici de 8).

Exercice VI

Dans tout l'exercice, on suppose que la représentation d'un nombre en virgule flottante est sur 16 bits : 1 bits de signe de la mantisse, 4 bits pour l'exposant biaisé à 7, 11 bits pour la mantisse normalisés, par exemple $-6,5^{10} = -110,1^2 = -1,101^2 \times 2^2$ sera codé 1 1001 110100000000 (on a inséré deux blancs pour faciliter la lecture).

- Donner la valeur décimale de 0 0110 11011000000.
- Représenter 0,125, -98, 51,625.
- Peut-on représenter -874?

Remarque : Lorsque les nombres sont représentés en virgule flottante pour les multiplier il suffit d'additionner les exposants, de multiplier les mantisses, puis de normaliser le résultat éventuellement ($(0,4 \times 10^{-1}) \times (0,2 \times 10^4) = 0,08 \times 10^3 = 0,8 \times 10^2$). Par contre pour l'addition si les exposants sont différents il faut dénormaliser la plus petite valeur, additionner les mantisses, puis normaliser le résultat éventuellement ($(0,3 \times 10^3) + (0,972 \times 10^4) = (0,03 \times 10^4) + (0,972 \times 10^4) = 1,002 \times 10^4 = 0,1002 \times 10^5$).

Exercice VII

Dans la norme IEEE 754 la simple précision est codée sur 32 bits : 1 bit de signe de la mantisse, 8 bits pour l'exposant biaisé à 127, 23 bits pour la mantisse normalisée, comme la mantisse normalisée est un nombre qui commence toujours par 1, ... (par exemple 1,101), on ne représente pas le 1 à gauche de la virgule, on dit qu'on a un bit caché.

- Donner la valeur décimale du code suivant écrit en hexadécimale C50A0D00
- Représenter -4062, 308,375 en donnant le code sous forme hexadécimale.

N.B.: Dans certains ordinateurs destinés à des applications commerciales, les opérations s'effectuent en base 10, il existe plusieurs façons de coder des décimaux en binaire (on code les chiffres de 0 à 9, puis on juxtapose les codes lorsque le nombre à plusieurs chiffres, en BCD, Binary Coded Decimal, par exemple les chiffres sont codés sur 4 bits, 25 sera codé 0010 0101). Les entrées-sorties sont beaucoup plus rapides, par contre les opérations sont compliqués à programmer.