

chapitre 1: Codage de l'information en machine.

I- Codage des nombres:

I-1- Système de numération

I-1-1- Définition.

Un système de numération décrit la façon avec laquelle les nombres sont représentés. Il est défini par

- un alphabet: qui constitue l'ensemble des symboles ou chiffres
- par les règles d'écriture et de lecture des nombres.

la base: est un nombre entier permettant de définir un système de numération.

I-1-2 - Conversion des entiers:

$10 \rightarrow b$ codage

$b \rightarrow 10$ décodage

$b_1 \neq 10 \rightarrow b_2 \neq 10$ transcodage.

a) Le codage

C'est le passage de la représentation d'un nombre défini en base 10 vers la représentation de ce même nombre en **base b**. Il s'effectue par une série de division entière du nombre par b jusqu'à obtention de 0 comme quotient et on prend les restes dans le sens contraire de la division.

Exemples :

$$(48)_{10} \longrightarrow (110000)_2$$

$$\begin{array}{r} 48 \overline{) 2} \\ 0 \quad 24 \overline{) 2} \\ \quad 0 \quad 12 \overline{) 2} \\ \qquad 0 \quad 6 \overline{) 2} \\ \qquad \quad 0 \quad 3 \overline{) 2} \\ \qquad \qquad 1 \quad 1 \overline{) 2} \\ \qquad \qquad \quad 1 \quad 0 \end{array}$$

b) Le décodage

C'est le passage de la représentation d'un nombre défini en base b vers la représentation de ce même nombre en base 10. Il s'effectue en utilisant tout simplement le polynôme suivant :

$$\sum_{i=0}^{n-1} a_i b^i$$

avec

b = la base

i = la position (à partir de la droite)

a_i = symbole à la position i

n = nombre de symboles.

Exemple :

$$(48)_{10} \longrightarrow (110000)_2$$

$$\begin{aligned} 1^5 1^4 0^3 0^2 0^1 0^0 &= 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + \dots \\ &= 48 \end{aligned}$$

Système de numération : Tableau d'équivalence

Hexadécimal		Octal		Binaire						Décimal
16^1	16^0	8^1	8^0	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	1
0	2	0	2	0	0	0	0	1	0	2
0	3	0	3	0	0	0	0	1	1	3
0	4	0	4	0	0	0	1	0	0	4
0	5	0	5	0	0	0	1	0	1	5
0	6	0	6	0	0	0	1	1	0	6
0	7	0	7	0	0	0	1	1	1	7
0	8	1	0	0	0	1	0	0	0	8
0	9	1	1	0	0	1	0	0	1	9
0	A	1	2	0	0	1	0	1	0	10
0	B	1	3	0	0	1	0	1	1	11
0	C	1	4	0	0	1	1	0	0	12
0	D	1	5	0	0	1	1	0	1	13
0	E	1	6	0	0	1	1	1	0	14
0	F	1	7	0	0	1	1	1	1	15
1	0	2	0	0	1	0	0	0	0	16

Ibrahima Kane
DUT1 Info

c) Le transcodage :

C'est le passage de la représentation d'un nombre défini en base b_1 vers la représentation de ce même nombre en base b_2 .

Le transcodage c'est un encodage suivi d'un codage.

→ Pour passer du binaire à l'octal on fait une série de regroupement de 3 bits de la droite vers la gauche et on donne l'équivalent octal de chaque groupe.

Exemple : $\underline{010} \underline{101} \rightarrow (25)_8$
 2 5

→ Pour passer de l'octal au binaire on donne l'équivalent binaire de chaque symbole sur 3 positions

→ Pour passer du binaire à l'hexadécimal on fait une série de regroupement de 4 bits de la droite vers la gauche et on donne l'équivalent hexadécimal de chaque groupe.

Ex : $(\underline{0011} \underline{0110} \underline{1001})_2 \rightarrow (369)_{16}$

→ Pour passer de l'hexadécimal à binaire on donne l'équivalent binaire sur quatre positions de chaque symbole.

I-1-3 - Conversions des décimaux :

Le codage d'un décimal se fait par une série de multiplication par la base jusqu'à obtention d'un nombre entier, d'un cycle ou épuisement de l'espace de stockage et on prend les parties entières dans le sens

de la multiplication.

Ex:

$$0, d_1 d_2 \dots d_p \times b = E_1, d_1^1 \dots d_p^1$$

$$0, d_1^1 \dots d_p^1 \times b = E_2, d_1^2 \dots d_p^2$$

\vdots

$$0, d_1^{k-1} \dots d_p^{k-1} \times b = E_k, \dots$$

$$\left(\begin{array}{c} E_1 \\ E_2 \\ \vdots \\ E_k \end{array} \right) \downarrow$$

$$(0, d_1 d_2 \dots d_p)_{10} \rightarrow (0, E_1 E_2 \dots E_k)_b.$$

$$* (0, 25)_{10} \rightarrow (0, 01)_2$$

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

$$0,5 \times 2 = 1,0 \quad 1 \Rightarrow 0,01$$

$$* (0, 05)_{10} \rightarrow (0, 000011\dots)_2$$

$$0, 05 \times 2 = 0,1$$

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

$$0,2 \times 2 = 0,4$$

$$0,4 \times 2 = 0,8$$

$$0,8 \times 2 = 1,6$$

$$0,6 \times 2 = 1,2$$

$$0,2 \times 2 = 0,4$$

$$\left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ \parallel \end{array} \right)$$

$$\Rightarrow (0, 000011\dots)_2$$

Pour décoder un décimal on utilise le polynôme suivant $\sum_{i=1}^P d_i b^{-i}$ avec :

- d_i le décimal à la position i
- i la position
- b la base et
- P le nombre de décimaux

Ex:

$$(0,01^2)_2 \rightarrow (0,25)_{10}$$

$$0 \times 2^{-1} + 1 \times 2^{-2} = 1 \times \frac{1}{4} = 0,25$$

Le transcodage d'un décimal se fait à travers un décodage suivi d'un ~~deco~~ codage.

II- Représentation de l'information en machine

II-1- Représentation des entiers:

Pour représenter un entier en machine on doit coder le nb et le représenter sur les n bits, n qui est le nb de bits sur lesquels on va représenter l'information permet de définir l'intervalle des nombres pouvant être manipulés. C'est ainsi que nous avons l'intervalle

$$[0, 2^n - 1]$$

Ex:

$$n=1 \quad [0, 1]$$

$$n=2 \quad [0, 3]$$

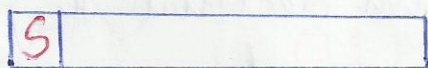
$$n=3 \quad [0, 7] = [0, 2^3 - 1]$$

II-2. Représentation des entiers signés:

a) Valeur absolue + signe

Pour représenter un nombre signé en codage valeur absolue plus signe on met le signe **S** sur le bit au poids le plus fort et la valeur absolue au niveau des $n-1$ bits restants. Le signe **S** est égal à 0 si le nbre est positif et $S=1$ si le nbre est négatif.

Ce qui nous définit un intervalle de travail qui est $[-2^{n-1}, -0] \cup [+0, +2^{n-1} - 1]$



b) Codage complément à 1

Coder un nbre positif en **Cà 1** revient à faire son codage valeur absolue + signe. Pour le nbre négatif il faudra complémenter la représentation v.a + signe de l'opposé du nbre.

Complémenter un nombre binaire revient à changer les 0 en 1 et les 1 en 0.

Ex:
 $n=5$

$$+4 \quad \text{Cà 1}(+4) = \text{v.a. S}(+4) \Rightarrow 00100$$

$$\begin{aligned} -4 \quad \text{Cà 1}(-4) &= \text{compl v.a. S}(\text{opp}(-4)) \\ &= \text{compl}(\text{v.a. S}(+4)) \\ &= \text{compl}(00100) = 11011 \end{aligned}$$

c) Complément à 2:

Pour représenter un nbre positif en C à 2 il suffit de faire la représentation v.a.s du nombre. S'il est négatif son complément à 2 c'est égal à son complément à 1 plus 1.

$$C \text{ à } 2(N) = v.a.s(N) \quad N > 0$$

$$C \text{ à } 2(N) = C \text{ à } 1(N) + 1 \quad N < 0$$

d) Mode excédent m.

En mode excédent m au lieu de représenter X (l'entier signé) on va représenter sur les m bits l'entier naturel $X + 2^{m-1}$.

$$X + 2^{m-1} \leq 2^m - 1$$

$$X + 1 \leq 2^m - 2^{m-1}$$

$$X + 1 \leq 2^m (2 - 1)$$

$$|X + 1| \leq 2^{m-1}$$

$$m \geq \frac{\ln |X + 1|}{\ln 2} + 1$$

II-3. Représentation des réels : flottante

Pour faire une représentation flottante on met d'abord le nombre sous forme **mantisse exposant** : $\pm 0, m \times b^e$ et on choisit de choisir que ce qui varie au niveau de cette représentation à savoir le signe, la mantisse et l'exposant. Nous avons deux types de représentation flottante :

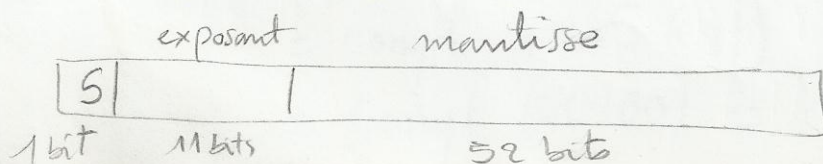
- la représentation flottante simple précision (32 bits) et
- la représentation flottante double précision (64 bits)

Les 32 bits de la représentation flottante simple précision sont définies comme suit :

- le bit au poids le plus fort constitue le bit de signe qui est à 0 si le nombre est positif et 1 si le nombre est négatif
- les 8 bits suivants sont réservés à l'exposant qui est codé en mode excédent 8 et
- les 23 bits restant accueillent la mantisse

Les 64 bits de la représentation flottante double précision sont représentés comme suit :

- le bit au poids le plus fort correspond au signe S.
- les 11 bits suivants à l'exposant codés en mode excédent 11
- et les 52 bits restant à la mantisse



$$* (-0,192)_{10}$$

$$SP \quad \boxed{1 \mid 10000000 \mid 0011000100100110110100}$$

$$0 + 2^{8-1} = 128 \Rightarrow 10000000$$

$$* (+0,5)_{10} \rightarrow (0,1)_2$$

$$0,1 = 0,1 \times 2^0 \rightarrow 0 + 2^{8-1} = 128$$

$$\boxed{0 \mid 10000000 \mid 10000000000000000000 \dots 0}$$

$$* (-15)_{10} \rightarrow (1111)_2$$

$$4 \times 2^{8-1}$$

$$1111 \times 2^0 = 1111,0 \times 2^0 = 1111,0 \times 2^4 \times 2^{-4} = 0,1111 \times 2^4$$

{ diviser un nbre par la base revient à décaler vers la gauche ; ici on a divisé 4 fois par 2 }

$$\boxed{1 \mid 10000100 \mid 1111000 \dots 0}$$

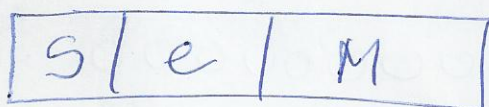
$$\text{exposant} = 4 + 2^{8-1} \Rightarrow \begin{array}{r} 100000000 \\ 1000 \\ \hline 10000100 \end{array}$$

Représente flottante normalisée

Pour faire une représentation flottante normalisée il faut avoir la représentation mantisse exposant, définie de telle manière que la mantisse commence par 1 on omet de représenter ce 1.

$$\pm 0, m \times 2^e \Rightarrow \pm 0, 1m' \times 2^{e'}$$

Valeur de flottant: $(-1)^s \times 0, M \times 2^{e-2^{m-1}}$



Exemple:

Valeur de $\boxed{1/10000100 | 111100 \dots 0}$

$$(-1)^1 \times 0, 111100 \dots \times 2^{2^7+2^2-2^7}$$

$$(-1)^1 \times 0, 1111 \times 2^4$$

$$(-1)^1 \times (1111)_2 \Rightarrow (-15)_{10}$$

Valeur d'un flottant normalisé:

$$(-1)^s \times 0, 1M \times 2^{e-2^{m-1}}$$

$$\Rightarrow (-1)^s \times 1, M \times 2^{e-2^{m-1}-1}$$

ex = $(-1)^{-1} \times 1, 111 \times 2^{e-2^{m-1}-1}$

$$= (-1)^1 \times 1, 111 \times 2^3$$

$$= (-1111)_2 = (-15)_{10}$$

II-4) Représentation des caractères:

BCD (6bit) $\rightarrow 2^6 = 64$ codes \leftrightarrow 64 symboles

ASCII (7bits) $\rightarrow 2^7 = 128$ codes \leftrightarrow 128 symboles

EBCDIC (8bits) $\rightarrow 2^8 = 256$ codes \Rightarrow 256 symboles

ASCII (8bits) $\rightarrow 2^8 = 256$ -
(Generalise').

