L'informatique industrielle

L'informatique industrielle est une branche de l'informatique appliquée qui couvre l'ensemble des techniques de conception et de programmation, de systèmes informatisés à vocation industrielle, qui ne sont pas des ordinateurs.

Applications :

- Automates, robotique,
- Mesures de grandeurs physiques,
- Systèmes temps-réel,
- Systèmes embarqués.

I - Les systèmes de numération

Le système binaire

La base est **2**. Les seuls chiffres existants sont donc 0 et 1 ($0 \leq a_i < B$). C'est le seul système qu'on peut implémenter (implanter au niveau matériel) de façon simple : 2 états 0 ou 1, information ou pas d'information, relais ou interrupteur ouvert ou fermé, transistor passant ou non, courant ou pas de courant … ! On notera A_2.

$(10010101)_B = 1*2^0 + 0*2^1 + 1*2^2 + 0*2^3 + 1*2^4 + 0*2^5 + 0*2^6 + 1*2^7 = 1+4+16+128 = 149$

2^0	2^1	2^2	2^3	2^4	2^5 …	2^{10}	2^{11} …	2^{20}	2^{21} …
1	2	4	8	16	31	1024 (1k)	2048 (2k)	1048576 1024k=1M	2097152 2048k=2M

Tableau 1 : poids du système binaire

Notons en particulier que : 1k (kilo) $=2^{10}=1024$; 1M (méga) $=2^{20}=1024$k

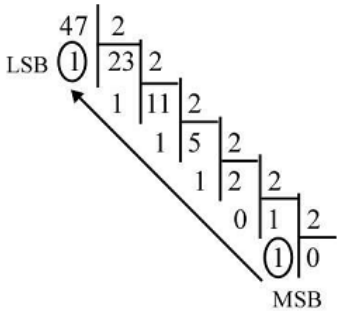
1G (giga) $= 2^{30} = 1024$M ; 1T (térra) $= 2^{40} =1024$G

Toute information est stockée, dans un ordinateur, sous forme de ***BIT*** (***BI****nary digi****T***). Ces informations sont regroupées par blocs de ***8 bits*** nommés ***OCTETs*** (***BYTE*** en anglais) ou par blocs de ***16 bits*** nommés ***MOTs*** (***WORD*** en anglais) ou même par blocs de ***32 bits*** nommés ***MOTs LONGs*** (***LONG WORD***).

Conversion d'un nombre positif en son équivalent binaire

Les restes successifs de la conversion entière par ***B*** jusqu'à ce que le quotient soit nul et lus de bas en haut représente A_B.

Application : $(47)_{10} = (?)_2$

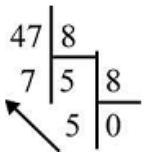


$(47)_{10} = (101111)_2$

Le système octal

La base est ***B***=8. Les seuls chiffres existants sont : 0, 1, 2, 3, 4, 5, 6, 7. On notera A_8.

$(47)_{10} = (57)_8$



Le système hexadécimal

La base est 16. Ce qui implique qu'il faut 16 chiffres pour représenter les nombres dans cette base. Les 10 chiffres du système décimal ne suffisaient donc pas pour coder les valeurs. Plutôt que d'inventer 6 nouveaux symboles, il a été décidé d'utiliser les 6 premières lettres de l'alphabet comme CHIFFRES.

Les chiffres existants dans cette base sont donc : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. On notera A_{16} ou A_h. On utilise cette représentation quand il s'agit de noter simplement de longues séries de bits. Un chiffre hexadécimal son équivalent en binaire s'écrit sur 4 bits (Tableau 2)

Nombres décimaux	Représentation binaire				En Hexa
	2^3=8	2^2=4	2^1=2	2^0=1	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	A
11	1	0	1	1	B
12	1	1	0	0	C
13	1	1	0	1	D
14	1	1	1	0	E
15	1	1	1	1	F

Conversion binaire – octal – hexadécimal

Elle est simple du fait que : $8 = 2^3$ → groupe de 3 bits

$16 = 2^4$ → groupe de 4 bits

$1 \quad 5 \quad E \quad \rightarrow (15E)_h$

Exp. : $\left(\overline{1}\,\overline{0101}\,\overline{1110}\right)_2$

$\overline{5} \quad \overline{3} \quad \overline{6} \quad \rightarrow (538)_8$

On groupe les bits par 3 (octal) ou par 4 (hexadécimal) en partant de la droite pour la partie entière et de la gauche pour la partie fractionnaire. Ceci produit une condensation de l'écriture, la représentation physique restant la même.

$1k = 1024 \rightarrow (1000000000)_2 \rightarrow (2000)_8 \rightarrow (400)_h$

3 Code DCB (Décimal codé binaire, anglais BCD)

Chaque chiffre décimal (0, …, 9) est représenté par un groupe de quatre bits en binaire. Un nombre BCD est donc formé de groupes de 4 bits indépendants, qui regroupés, ne forment pas de valeur binaire.

Exp. :

$(45)_{10} = (0100\ 0101)_{BCD}$

$(5609)_{10} = (0101\ 0110\ 0000\ 1001)_{BCD}$

4 Représentation des caractères alphanumériques (*Donner le tableau du code ASCII*)

Le code ASCII (American Standard Code for Information Interchange) est appelé aussi code ISO (International Organization for Standerdization). Le code ASCII de base utilise 7 bits et permet de représenter 128 (2^7) caractère différents : les lettres majuscules et minuscules, les chiffres (de 0 à 9), les caractères spéciaux et symboles numérique (&, *, $ …), les caractères de control …

Le huitième bit est un bit de parité qui rétablit un nombre pair de 1 dans le caractère pour la détection d'erreur. Si ce huitième bit est utilisé directement pour le codage, le code peut représenter 256 (2^8) caractères différents.

5 Arithmétique binaire

Principe :

		A + B		A - B	
A	B	Résultat	Retenue	Résultat	Retenue
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	0	1	0
1	1	0	1	0	0

Les circuits travaillent sur des nombres qui ont toujours la même longueur (format). Par exemple, si le format est de 8 bits, le nombre $(\mathbf{10111})_2$ doit être complété par des zéros : $(\mathbf{00010111})_2$. En effectuant une addition, une ***retenue*** (***carry***) peut apparaître. En effectuant une soustraction, une ***retenue*** (***borrow***) peut apparaître. En manipulant des nombres trop grands pour le format, on obtient un ***dépassement de capacité*** (***overflow***).

Exp. :

Addition

$$\begin{array}{r} 59 \\ +42 \\ \hline 101_{10} \end{array} \qquad \begin{array}{r} 0^{1}0^{1}1^{1}1^{1}011 \\ +\,0\,0\,1\,0\,1\,0\,1\,0 \\ \hline 0\,1\,1\,0\,0\,1\,0\,1 \end{array}$$

Soustraction

$$\begin{array}{r} {}^{7}\not{8}^{1}5 \\ -\ 5\ 7 \\ \hline 2\ 8 \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ \ 10} \\ {\scriptstyle 0\ \not{1}\not{0}\ \not{0}\ 10} \\ 0\not{1}\,0\not{+}0\,1\,0\,1 \\ -\,0\,0\ 1\ 1\ 1\,0\,0\,1 \\ \hline 0\,0\ 0\ 1\ 1\,1\,0\,0 \end{array}$$

Multiplication

$$\begin{array}{r} 1101 \\ \times\ \ 101 \\ \hline {}^{1}1101 \\ {}^{11}0000 \\ 1101 \\ \hline 1000001 \end{array} \qquad \begin{array}{r} 13 \\ \times 5 \\ \hline 65 \end{array}$$

Division

$$\begin{array}{r|l} 55 & 5 \\ \hline 5 & 11 \end{array} \qquad \begin{array}{r|l} 110\ 111 & 101 \\ \hline 111 & 1011 \\ 101 & \\ \hline \end{array}$$

5.1 Codage des nombres négatifs

Le bit de poids fort (MSB : Most Signicant Bit) est bit de signe. Il indique le signe du nombre : 0 → positif ; 1 → négatif. Il existe trois modes de représentation.

5.1.1 Signe et valeur absolue

Le MSB est le bit e signe, les autres bits indiquent la valeur absolue des nombres.

Exp. :

$(17)_{10} = 0\ 10001$
$(-17)_{10} = 1\ 10001$

et non $(32+17)_{10} = (49)_{10}$ exprimé en binaire pure

Cette représentation produit une bonne lisibilité, mais nécessite un transcodage pour les calculs arithmétiques.

5.1.2 Complément à 1

Si ***n*** est le nombre de bits constituant le nombre ***N***, il est appelé aussi complément à $\mathbf{2^n-1}$. C'est le complément logique ($-N = \overline{N}$) de tout les bits, bit à bit : $-N + N = 2^n + 1$.

Exp. :

$(17)_{10} = 0\ 10001$
$(-17)_{10} = 1\ 01110$

et non $(31+8+4+2)_{10}=(46)_{10}$ exprimé en base 2

5.1.3 Complément à 2

Si n est le nombre de bits constituant le nombre N, il est appelé aussi complément à 2n. C'est le complément à 1 auquel on ajout 1 ($-N = \overline{N} + 1$) : $-N + N = 2^n$.

Exp. :

$(17)_{10} = 0\ 10001$
$(-17)_{10} = 1\ 01110$ (complément à 1)
$+\quad 1$
$(-17)_{10} = 1\ 01111$ (complément à 2) et non $(47)_{10}$ en base 2

Autres exemples :

17 + 8 =25

Signe	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	1	0	0	0	1
0	0	1	0	0	0
0	1	1	0	0	1

17 + 17=34

Signe	2^4 16	2^3 8	2^2 4	2^1 2	2^0 1
0	1	0	0	0	1
0	1	0	0	0	1
1	0	0	0	1	0

Dans ce dernier exemple, le résultat est faux, car le signe des deux opérandes est positif (bit de signe à 0) et celui du résultat est négatif : il y a dépassement de capacité.

5.1.4 La soustraction

Elle est ramenée à une addition effectuée sur les nombres représentés en complément à deux. Avant d'effectuer les opérations il faut toujours définir le format (la taille) de représentation des nombres (dans les exemple ça sera sur 8 bits; le bit MSB est toujours le bit de signe).

17 + (-8) = +9

retenue	Signe							
	0	0	0	1	0	0	0	1
	1	1	1	1	1	0	0	0
1	0	0	0	0	1	0	0	1

On ne tient pas compte de la retenue.

Le complément à 2 de (-8)

	0	0	0	0	1	0	0	0	(+8)
	1	1	1	1	0	1	1	1	c/1
+								1	
	1	1	1	1	1	0	0	0	c/2 (-8)

(-17) + (-8) = ***X***= -25

retenue	Signe							
	1	1	1	0	1	1	1	1
	1	1	1	1	1	0	0	0
1	1	1	1	0	0	1	1	1

On ne tient pas compte de la retenue.

	1	1	1	0	0	1	1	1	Résultat (***X***)
	0	0	0	1	1	0	0	0	c/1
+								1	
	0	0	0	1	1	0	0	1	c/2 (+25)

	0	0	0	0	1	0	0	0	(+8)
	1	1	1	1	0	1	1	1	c/1
+								1	
	1	1	1	1	1	0	0	0	c/2 (-8)

	0	0	0	1	0	0	0	1	(+17)
	1	1	1	0	1	1	1	0	c/1
+								1	
	1	1	1	0	1	1	1	1	c/2 (-8)