LYCÉE de l'ÉLORN
le choix de votre réussite

Sciences de l'Ingénieur

Représentation des données
Opération sur les mots binaires

I - Opérations sur les nombres binaires

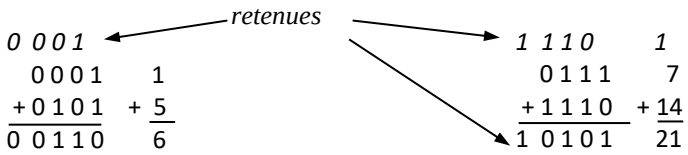
1 Opérations arithmétiques

1.1 Instructions d'additions

Les opérations arithmétiques se font de la même façon en binaire qu'en base décimale. Lorsque l'on dépasse la valeur maximale au niveau du rang *n* (résultat > 1) on génère une retenue au bit de rang *n+1*. De fait, une addition de 2 mots de n bits donne un résultat sur n+1 bits.

Ainsi : 0+0=00 ; 0+1 = 01 ; 1+0=01 ; 1+1=10

Voyons cela sur une addition de deux mots de 4 bits :



1.2 Instructions de soustractions

Même principe que précédemment en admettant les bases suivantes : 0–0=00 ; 1-0=01 ; 1-1=00 ; 0-1=11 ; 0-1-1=10

Ce qui donne sur 4 bits :

☞ *Faites l'opération 20-14 en binaire.*

0 1 1 1 *1*
1 1 1 0 14
- 0 1 1 1 - 7
0 0 1 1 1 07

2 Complément à 1 et à 2

2 .1 Le complément à 1 (ou complément restreint)

Le complément à un d'un nombre binaire se forme en soustrayant de 1 chaque bit de ce nombre. Cela revient à inverser chacun des bits.

Exemple : Si A = 00010101 alors $\overline{A^1}$ = 11101010

2.2 Le complément à 2 (ou complément vrai)

Soit A un nombre binaire, le complément à deux de A est $\overline{A^2} = \overline{A^1} + 1$

Exemple : Si A = 00001001 → $\overline{A^1}$ = 11110110 → $\overline{A^2}$ = 11110111

On présente ainsi une notion de **nombre négatif** en binaire puisque quel que soit A écrit sur n bits :

$$A + \overline{A}^2 = 0 \quad \text{d'où} \quad \mathbf{\overline{A}^2 = -A}$$

Intérêt : On peut remplacer les soustractions par les additions

2.3 Les nombres négatifs en base 2

Par convention, le bit de poids fort (MSB) d'un nombre défini son signe (+ ou -). Un nombre négatif s'exprime en complément à 2 en ajoutant un bit de poids fort pour intégrer le signe.

Soit un nombre de 8 bits : $\mathbf{A = a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0}$

Bit de signe
A > 0 $a_7 = 0$
A < 0 $a_7 = 1$

Signification
A > 0 : binaire pur
A < 0 : Complément à 2

Exemple A = - 3 → |A| = 3 ≡ **0**0000011 d'où $\overline{A}^1$ = 11111100 puis $\overline{A}^2$ = **1**1111101

0 : Signe **+**

Valeur absolue

1 : Signe -

Complément à 2 de +3 ≡ -3

Tableau des nombres positifs et négatifs sur un format de 4 bits :

0	0 000	*-1*	1 111
1	0 001	*-2*	1 110
2	0 010	*-3*	1 101
3	0 011	*-4*	1 100
4	0 100	*-5*	1 011
5	0 101	*-6*	1 010
6	0 110	*-7*	1 001
7	0 111		

A consulter : Connaissez-vous le Bug de l'an 2038 ? → https://fr.wikipedia.org/wiki/Bug_de_l%27an_2038

Application :

Donner sur 8 bits la réprésentation binaire des nombres suivants : 1, -1 , +125, -125, +110, -110

……

……

Soit A=125 et B=110. Réaliser en binaire sur 8 bits signés les opérations A+B; A-B; B-A; -A-B puis valider les résultats trouvés :

……

……

……

……

……

……

……

……

3 Codage des grandeurs réelles : 32 bits à virgule flottante.

La donnée est un nombre signé (+/-) sur 32 bits à virgule flottante respectant la norme IEEE 754. Elle est formée de trois éléments ; la mantisse (nombre fractionnaire) sur 23 bits, l'exposant (décalé de la valeur 127) sur 8 bits et le bit de signe (S = 1 pour un nombre négatif) et sont répartis comme suit :

Rang	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	S	Exposant								Mantisse																						
Poids		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

Le nombre a pour valeur : $((-1)^S$ x 1+mantisse) x $2^{(\text{exposant-127})}$

Exemple
Soit le nombre % 00111110 00100000 00000000 00000000
En isolant les bits on trouve le format suivant : 0 01111100 01000000000000000000000,

Le signe est nul, l'exposant est 124 - 127 = -3, et la mantisse est 0,25 ($0 \times 2^{-1} + 1 \times 2^{-2}$). Le nombre représenté est donc $+1{,}25 \times 2^{-3}$, ce qui donne + 0,15625.

☞ Exprimer en base 10 le nombre réel de codage IEEE 754 qui est codé en hexadécimal avec la valeur suivante : *$C2D5A000*

.....
.....
.....
.....

4 Instructions logiques

Les instructions logiques permettent de faire des opérations bit-à-bit sur des nombres binaires. C'est-à-dire en considérant chacun des bits de même poids indépendamment des autres, sans se soucier de la retenue.

- L'instruction **AND** : Cette instruction met le bit du résultat à 1 si les deux bits de même poids des opérandes sont à 1, sinon il le met à zéro.
- L'instruction **OR** (OU) met le bit du résultat à 0 si les deux bits de même poids des opérandes sont à 0, sinon il le met à un.
- La fonction **XOR** (OU exclusif) met le bit du résultat à 1 si les deux bits de même poids sont différents.
- La fonction **NOT** (NON inversion) inverse chacun des bits d'un nombre binaire.

Exemple :

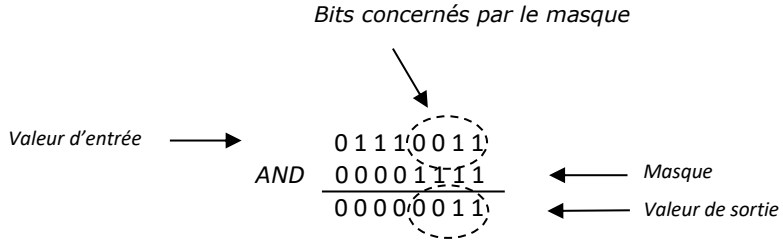
AND 0 1 1 1 / 1 1 1 0 = 0 1 1 0

OR 0 1 1 1 / 0 1 1 0 = 0 1 1 1

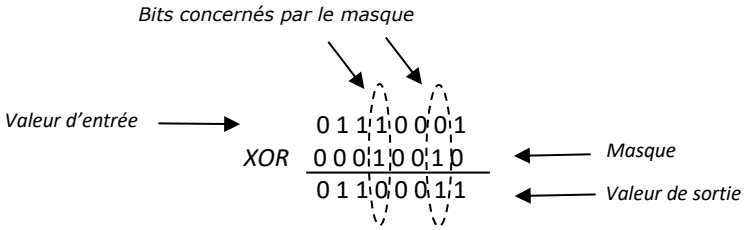
NOT (1 1 0 0) = 0 0 1 1

Application du Masquage :

Il est possible de masquer, c'est-à-dire mettre à zéro tous les bits qui ne nous intéressent pas dans un nombre. Dans l'exemple ci-dessous, seuls les 4 derniers bits de l'octet 01110011 seront récupérés dans le résultat de l'opération 01110011 AND 00001111



Ici l'on procède à l'inversion des bits 2 et 5 de l'octet 01110011 avec l'opération 01110011 XOR 00010010



Exemple d'application sur l'adressage IP :

On utilise le principe du masquage avec l'adressage IP. A partir d'une adresse IP et d'un masque de sous-réseau on peut générer l'adresse commune dite « adresse réseau » des systèmes intégrés à un même réseau.

☞ Compléter ce tableau :

IP décimale	192	168	155	24
IP binaire				
Masque décimal	255	255	240	0
Masque binaire				
Adresse réseau (netid) **= IP and Masque**				