

Fiche de cours n°5 OPERATION SUR LES NOMBRES BINAIRES

Professeur : G. Maléjacq

Les instructions arithmétiques et logiques sont effectuées par l'Unité Arithmétique et Logique (**UAL**) d'un microprocesseur. Il s'agit d'opérations directement effectuées sur les bits de la donnée que l'on traite.

Sont comprises dans cette appellation :

- les instructions d'addition
- les instructions de soustraction
- les instructions de décalage
- les instructions de rotation
- les instructions logiques (ET, OU, ...)

Instructions d'addition

Les opérations arithmétiques se font de la même façon en binaire qu'en base décimale. C'est-à-dire que lorsque l'on dépasse la valeur maxi au niveau du bit n (lorsque l'on dépasse la valeur 1) on a une retenue ou un report au bit $n+1$.

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

Voyons cela sur des mots de 4 bits :

$0\ 001$	\leftarrow retenues	$1\ 110$	1
0001		0111	7
$+0101$		$+1110$	$+14$
00110		$1\ 0101$	21
$+5$			
6			

Instructions de soustraction

Même principe que précédemment en admettant les bases suivantes :

$0-0=00$; $0-1=11$; $1-0=01$; $1-1=00$ Ce qui donne sur 4 bits :

$0\ 111$	1
1110	14
-0111	-7
$0\ 0111$	07

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 $A = 10101$ alors $\bar{A}^1 = 01010$

Le complément à 2 (ou complément vrai)

Soit A un nombre binaire : Le complément à deux de A est $\bar{A}^2 = \bar{A}^1 + 1$

Exemple : Si $A = 1001 \rightarrow \bar{A}^1 = 0110 \rightarrow \bar{A}^2 = 0111$

On présente ainsi une notion de **nombre négatif** en base 2 car quel

que soit A écrit sur n bits :

$$A + \bar{A}^2 = 0 \text{ sur ses } n \text{ bits} \quad \text{d'où} \quad \bar{A}^2 = -A$$

Intérêt : On peut remplacer les soustractions par les additions

Les nombres négatifs en base 2

Par convention, le dernier bit d'un mot (bit de poids fort) désigne le signe du mot (+ ou -). Un nombre négatif s'exprime en complément à 2 en ajoutant 1 bit de poids fort pour intégrer le signe.

Soit un nombre de 4 bits : $A = a_3 \ a_2 \ a_1 \ a_0$

Bit de signe
 $A > 0 \ a_3 = 0$
 $A < 0 \ a_3 = 1$

Signification
 $A > 0$: binaire pur
 $A < 0$: Compl à 2

Exemple $A = -3 \rightarrow |A| = 3 \equiv 0011$ d'où $\bar{A}^1 = 1100$ puis $\bar{A}^2 = 1101$

0 : Signe + Valeur absolue 1 : Signe -
 Complément à 2 de +3 $\equiv -3$

Tableau des nombres positifs et négatifs sur 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		

Application :

Donner sur 8 bits la représentation binaire des nombres suivants : 1, -1 , +33, -33, +50, +100, -100

.....

.....

.....

Soit A=+14 et B=+12 réaliser en binaire sur 5 bits les opérations suivantes :

A+B A-B B-A -A -B

.....

.....

.....

.....

.....

.....

.....

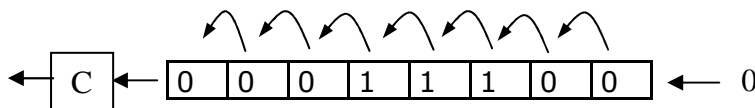
Décalage et rotation

Ces instructions permettent de décaler d'un côté ou de l'autre les bits des registres accumulateurs. Cette opération qui semble inutile a en fait plusieurs applications très intéressantes, dont:

- permettre de lire un à un les bits du registre (car les bits sortant à gauche positionnent l'indicateur de retenue C)
- permettre une **multiplication par 2^n** (en effet le fait de décaler un nombre binaire d'un chiffre à gauche le multiplie par 2, ainsi en effectuant cette opération n fois on obtient une multiplication par 2^n)
exemple:
00010 (2 en décimale)
00100 (on décale à gauche on obtient 4)
01000 (on décale à gauche on obtient 8)
- permettre une **division par 2^n** (comme précédemment mais en effectuant une rotation sur la droite)

Une **opération de décalage** déplace chacun des bits d'un nombre binaire sur la gauche (ou la droite), mais ceux-ci sortent vers le bit de retenue **C** lorsqu'ils arrivent au bit de poids fort (ou de poids faible) puis sont définitivement perdus.

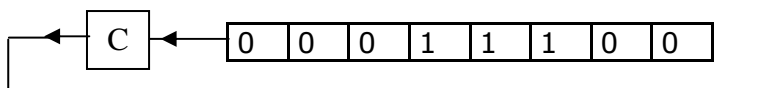
exemple:



00011100
00111000 (on décale d'un bit à gauche)
01110000 (on décale d'un bit à gauche)
11100000 (on décale d'un bit à gauche)
11000000 (on décale d'un bit à gauche)

Une **opération de rotation** agit comme une opération de décalage à la différence près que les bits qui sortent d'un côté rentrent de l'autre...

exemple:



00011100
00111000 (on effectue une rotation d'un bit à gauche, C est initialement à 0)
01110000 (on effectue une rotation d'un bit à gauche, C est à 0)
11100000 (on effectue une rotation d'un bit à gauche, C est à 0)
11000000 (on effectue une rotation d'un bit à gauche, C est à 1)
10000001 (on effectue une rotation d'un bit à gauche, C est à 1)

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 indépendamment des autres, sans se soucier de la retenue).

- L'instruction **AND** (ET) multiplie les bits de même poids deux à deux et stocke le résultat dans le registre de destination. Cette instruction met donc le bit du résultat à 1 si les deux bits de même poids de la source et de la destination sont tous deux à 1, sinon il le met à zéro.
- L'instruction **OR** (OU) met donc le bit du résultat à 0 si les deux bits de même poids de la source et de la destination sont tous deux à 0, sinon il le met à un.
- La fonction **NOT** (NON inversion) inverse chacun des bits d'un nombre binaire.

Exemple :

$$\begin{array}{r}
 \text{AND} \quad 0111 \\
 \quad \quad 1110 \\
 \hline
 \quad \quad 0110
 \end{array}
 \qquad
 \begin{array}{r}
 \text{OR} \quad 0111 \\
 \quad \quad 0110 \\
 \hline
 \quad \quad 0111
 \end{array}
 \qquad
 \text{NOT}(1100) = 0011$$

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. Pour cela il faut créer une valeur masque: un nombre dont les bits de poids qui nous intéressent sont à 1, les autres à 0. Il suffit alors de faire un **AND** entre le nombre à masquer et le masque pour ne conserver que les bits auxquels on s'intéresse.

Dans l'exemple ci-dessous, seuls les 4 derniers bits seront récupérés dans le résultat de l'opération

Bits concernés par le masque

$$\begin{array}{r}
 \text{Valeur d'entrée} \rightarrow 0111:0011 \\
 \text{AND} \quad \quad \quad 00001111 \\
 \hline
 \quad \quad \quad 00000011 \\
 \text{Valeur de sortie}
 \end{array}$$

Masque

Ici l'on procède à l'inversion des bits 2 et 5

Bits concernés par le masque

$$\begin{array}{r}
 \text{Valeur d'entrée} \rightarrow 0111:0011 \\
 \text{XOR} \quad \quad \quad 00010010 \\
 \hline
 \quad \quad \quad 01100001 \\
 \text{Valeur de sortie}
 \end{array}$$

Masque