Lycée Jean Moulin **Draguignan**
NSI 1re Année .....

Arithmétique en binaire

Chouette, des maths...

Comme nous l'avons vu, il est possible de convertir n'importe quel nombre entier positif exprimé en base 10 en un nombre binaire, c'est à dire en base 2.
Une fois que nous avons les nombres exprimés en binaire, il est normal de vouloir faire des opérations dessus, sans repasser par la base 10.
Nous allons principalement travailler sur l'addition. La table d'addition ci-dessous donne le résultat de l'addition de nombres à un seul chiffre :

+	0	1
0	0	1
1	1	10

Il y a une retenue dans le cas $1+1$, et uniquement dans ce cas.
Pour additionner deux nombres à plusieurs chiffres, il faut commencer par rajouter des 0 à gauche du plus court des deux et ensuite, on fait l'addition chiffre par chiffre, en partant de la droite et en faisant attention aux retenues. Voici quelques exemples :

$$\begin{array}{r} {\scriptstyle 1} \\ 0\ 1 \\ +\ 0\ 1 \\ \hline 1\ 0 \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ 1\ 0\ 0} \\ 0\ 0\ 1\ 1 \\ +\ 1\ 0\ 0\ 1 \\ \hline 1\ 1\ 0\ 0 \end{array} \qquad \begin{array}{r} {\scriptstyle } \\ 0\ 1\ 0\ 1 \\ +\ 1\ 0\ 1\ 0 \\ \hline 1\ 1\ 1\ 1 \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ \ 1} \\ 0\ 1\ 0\ 1 \\ +\ 0\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 0 \end{array} \qquad \begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1\ 0} \\ 0\ 1\ 1\ 1\ 1 \\ +\ 0\ 0\ 0\ 0\ 1 \\ \hline 1\ 0\ 0\ 0\ 0 \end{array}$$

Exercice 1 : Effectuer les additions suivantes :

1) $110+1=$ 2) $101+11=$ 3) $1100+0011=$
4) $1011+1000=$ 5) $1000+1000=$ 6) $1111+1111=$

Pour la suite, nous appellerons **bits**, les chiffres des nombres écrits en binaire.
On peut remarquer quelques propriétés :

Base 10	Base 2
00	0000
01	0001
02	0010
03	0011
04	0100
05	0101
06	0110
07	0111
08	1000
09	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Propriété :
- Un nombre binaire de la forme $10\cdots0$ correspond à une puissance de 2. S'il a k bits, alors le nombre est égal à 2^{k-1}.
- Un nombre binaire de la forme $1\cdots1$, avec k fois le bit 1 est égal à 2^k-1.
- Ajouter un 0 à la fin d'un nombre revient à le multiplier par 2.
- Pour déterminer le nombre de bits qu'il faut pour représenter le nombre n, il faut trouver le plus grand entier k tel que $2^{k-1} \leq n < 2^k$. Il faut alors k bits pour représenter n.

D'après les résultats précédents, nous en déduisons qu'avec k bits, il est possible de représenter 2^k nombres. Si on représente les nombres positifs, cela correspond aux nombres de 0 à 2^k-1. Ci-contre se trouve la table des 16 entiers positifs pouvant être écrits avec 4 bits.
Mais comment faire pour les nombres négatifs ?

Qui peut le plus, peut le moins

Une première méthode consiste simplement à utiliser le bit le plus à gauche pour indiquer le signe. Ainsi, sur 4 bits, 3 est représenté par 0011, donc −3 sera représenté par 1011.

Exercice 2 : Compléter le tableau ci-contre en calculant la représentation des entiers négatifs.

Cette représentation très simple pose néanmoins plusieurs problèmes. Tout d'abord, si on fait la somme −3+3, on aimerait trouver 0.

$$\begin{array}{r} {\scriptstyle 1\ 1\ \ \ } \\ 1\,0\,1\,1 \\ +\,0\,0\,1\,1 \\ \hline 1\,1\,1\,0 \end{array}$$

On remarque qu'on ne trouve pas 0, mais −6.
De plus, avec 4 bits, il est possible de représenter les entiers de −7 à 7. Cela ne fait que 15 valeurs différentes, en comptant 0. C'est parce qu'il est possible d'écrire −0 avec 1000. Ce n'est donc pas une solution satisfaisante.

Base 10	Binaire
−7	
−6	
−5	
−4	
−3	
−2	
−1	1001
−0	1000
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

On va essayer de faire un peu mieux

Un deuxième méthode consiste à inverser les bits du nombre. Par exemple, si on prend des nombres de 4 bits, 5 s'écrit 0101. On peut représenter −5 par 1010. C'est ce qu'on appelle le **complément à un**, puisque pour chaque bit, on choisit celui qui permet que la somme des deux soit 1. On remarque que les premiers nombres positifs ont leur bit à gauche qui vaut 0 alors que leurs opposés ont un 1. On retrouve le bit de signe de la première méthode.

Exercice 3 : Compléter le tableau ci-contre en calculant la représentation des entiers négatifs.

Une nouvelle fois, on a deux façon d'écrire 0, avec 0000 et 1111. C'est un des problèmes de cette méthode. Une des opérations de base pour un processeur, c'est de tester si un nombre est nul. Avec cette représentation, il faut donc tester si le nombre n'est composé que de 0 ou que de 1. Ce n'est pas une solution optimale. Par contre, la somme de deux nombres opposés donne toujours 1111, ce qui correspond bien à 0.

Base 10	Compl. à 1
−7	
−6	
−5	
−4	
−3	
−2	1101
−1	1110
−0	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Complément à deux

La solution retenue pour représenter les entiers relatifs dans la plupart des systèmes s'appelle **complément à deux**. Il existe plusieurs façons de la définir.

L'idée principale est de garantir que la somme d'un nombre et de son opposé donne 0. Prenons la représentation de 3 sur 4 bits : 0011. On veut trouver le nombre qui vérifie l'égalité ci-contre.

$$\begin{array}{r} 0\,0\,1\,1 \\ +\,?\,?\,?\,? \\ \hline 0\,0\,0\,0 \end{array}$$

Pour cela on commence par faire le complément à un, ce qui donne 1111 en faisant la somme.

$$\begin{array}{r} 0\,0\,1\,1 \\ +\,1\,1\,0\,0 \\ \hline 1\,1\,1\,1 \end{array}$$

Ce n'est pas le résultat voulu, mais si on ajoute 1 au résultat, on obtient 10000, qui donne 0000 si on oublie la retenue.
Il faut donc prendre $1100 + 0001 = 1101$, ce qui donne le résultat ci-contre.

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ \ } \\ 0\,0\,1\,1 \\ +\,1\,1\,0\,1 \\ \hline 0\,0\,0\,0 \end{array}$$

Méthode :
Pour obtenir le complément à deux, on commence d'abord par prendre l'inverse bit à bit, puis on ajoute 1 au nombre trouvé, en oubliant la dernière retenue si elle fait augmenter le nombre de bits nécessaires.
$3_{10} = 0011_2 \to 1100_2 + 1 = 1101_2 = -3_{10}$

Base 10	Compl. à 2
−8	1000
−7	1001
−6	1010
−5	1011
−4	1100
−3	1101
−2	1110
−1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

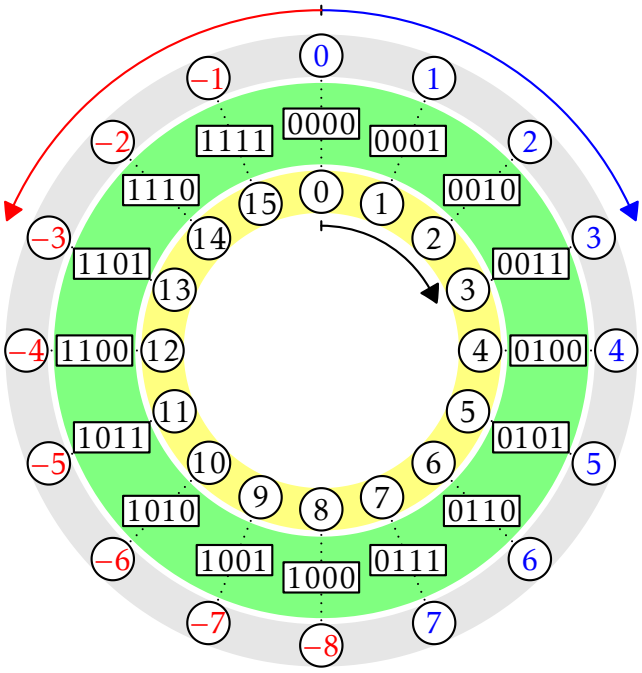
On remarque que, par construction, la somme de deux nombres opposés donne bien 0. On peut aussi remarquer que 0 n'a qu'une seule écriture possible.
En effet, si on veut représenter −0, on prend l'inverse bit à bit, qui est 1111. En ajoutant 1, on retrouve 0000.
Enfin, cette fois, on peut bien représenter 16 valeurs différentes, allant de −8 à 7, ce qui est mieux qu'avec les deux autres méthodes.

Exercice 4 : Calculer les représentations binaire en 8 bits des nombres suivants et de leur opposé :

1) 1 2) 5 3) 10
4) 56 5) 75 6) 116

Si on place sur un cercle les nombres entiers, leur représentation binaire et les nombres correspondant en complément à deux, on obtient le diagramme ci-contre.
On remarque que si on continuait le tour, le nombre 16 se trouverait sur 0. C'est parce que si on ne garde que 4 bits, 10000 devient 0000. De plus, si on prend un nombre positif avec la même écriture binaire qu'un nombre négatif et qu'on soustrait les deux, on obtient 16 :

$$15 - (-1) = 14 - (-2) = \cdots = 8 - (-8) = 16$$



Méthode :
On obtient une autre méthode pour trouver le complément à 2 de $-n$ sur k bits, avec n positif.
Il suffit de prendre la représentation binaire du nombre $2^k - n$. Sur 4 bits, l'opposé de 7 correspond donc à :
$2^4 - 7 = 16 - 7 = 9 = 1001_2$

Méthode :
Exemple : Représentation de −140 sur 1 octet:
Sur 8 bit le plus grand entier représentable est $2^8 - 1 = 255$, donc 256 et 0 ont la même représentation.
On doit avoir $140 + (-140) = 0(256)$, donc la représentation en binaire de −140 sur 1 octet est celle de $256 - 140 = 116 \rightarrow 01110100_2$

Exercice 5 : Calculer les représentations binaire en 8 bits des nombres suivants :

1) −156 2) −5 3) −100
4) −255 5) −34 6) −116

Méthode :
Dans l'autre sens : On considère l'entier négatif dont la représentation en binaire sur 1 octet est 01110100_2
Comment déterminer son écriture en base 10?
Comme on travaille sur 1 octet 256 et 0 ont la même représentation.
$01110100_2 \rightarrow 116_{10}$ et $256 - 116 = 140$, le nombre est donc −140

Exercice 6 : Déterminer le nombre en base 10 correspondant aux nombres suivants qui utilisent le complément à deux sur 8 bits.

1) 11111010 2) 00010110 3) 00011111
4) 10111111 5) 10101001 6) 01111110

Il était une fois...

Il existe plusieurs façons de faire les multiplications en binaire. On peut simplement "dépiler" un des deux facteurs :

$$A \times B = \underbrace{A + A + A + \cdots + A}_{B \text{ fois}}$$

Nous allons plutôt poser les multiplications, comme "à l'école". Commençons avec des nombres positifs, dans l'exemple ci-contre.

```
    1 1 0 1
  × 1 0 1 1
  ---------
    1 1 0 1
  1 1 0 1
  0 0 0 0
1 1 0 1
---------
1 0 0 0 1 1 1 1
```

Le résultat obtenu a 8 bits. C'est parce que lors d'une multiplication de deux nombres de k bits, on peut obtenir un résultat à $2k$ bits.
On peut noter que la multiplication est relativement simple à poser puisqu'à chaque bit du nombre du bas, si c'est un 1, on recopie le nombre de haut, sinon on met des 0. Il n'y a donc aucune multiplication à faire, juste des additions.

Exercice 7 : Vérifier le résultat de la multiplication précédente en convertissant les nombres en base 10.

Exercice 8 : Poser en binaire les multiplications suivantes :

1) 0011×0011 2) 1011×1001 3) 1111×1111

Compléments...Multiplier des entiers relatifs

Pour les nombres relatifs, c'est un peu plus compliqué. Tout d'abord on étudie le signe des deux facteurs A et B :

- Si $A > 0$ et $B > 0$, on pose $A \times B$, selon la méthode ci-dessus.
- Si $A < 0$ et $B > 0$, on pose $A \times B$, selon la méthode présentée ci-dessous.
- Si $A > 0$ et $B < 0$, on pose $B \times A$, selon la méthode présentée ci-dessous.
- Si $A < 0$ et $B < 0$, on pose $(-A) \times (-B)$, selon la méthode ci-dessus.
- Si l'un des deux est nul, le résultat est 0.

Si le premier nombre est négatif, on complète avec k bits devant chacun des nombres en recopiant le bit de gauche à chaque fois. C'est à dire qu'on met des 1 pour un nombre négatif et des 0 pour un nombre positif. On pose ensuite la multiplication en ne gardant que les $2k$ derniers bits.

Par exemple pour faire 1100×0101, on pose la multiplication ci-contre.

Cette multiplication correspond à -3×5. Le nombre obtenu est bien négatif. Il est exprimé sur 8 bits. On inverse les bits et on trouve 00001110. On ajoute 1 et on obtient 0001111, qui correspond à 15. Le résultat de la multiplication était bien -15.
En fait, il est possible de poser la multiplication de n'importe quels entiers relatifs en complétant à gauche les bits de signe, mais la méthode présentée permet de simplifier les multiplications obtenues.

```
  1 1 1 1 1 1 0 1
× 0 0 0 0 0 1 0 1
-----------------
  1 1 1 1 1 1 0 1
  0 0 0 0 0 0 0
  1 1 1 1 0 1
  0 0 0 0 0
-----------------
  1 1 1 1 0 0 0 1
```

Exercice 9 : Déterminer le résultat des multiplications suivantes, entre nombre relatifs exprimés sur 4 bits.

1) 0011×1001 2) 1011×0110 3) 1111×1111

Ce n'est pas ainsi que les multiplications sont effectuées par les processeurs. Il existe des techniques bien plus rapides, mais plus complexes à comprendre.