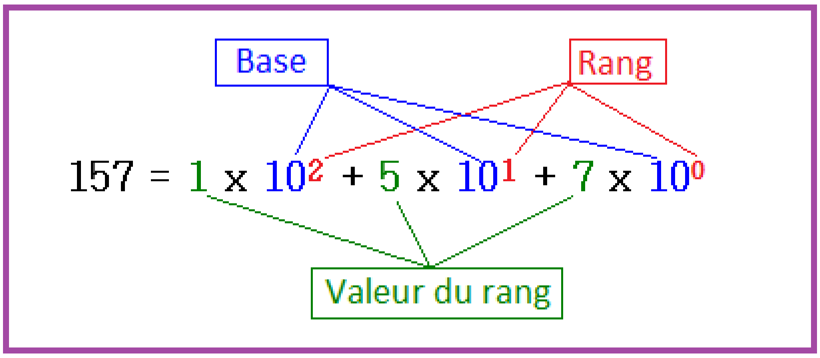
N U M E R A T I O N

1. QUELQUES BASES DE NUMERATION

1.1 La numération décimale (base 10)

La base 10 est notre système usuel de numération. Elle comporte 10 symboles (chiffres) de 0 à 9. Il s'agit d'une numération de position car la place des chiffres dans le nombre est importante. (Par exemple, le chiffre 5 n'a pas la même signification (le même « poids ») dans les nombres 157 - 548 - 665)



Quelle est la valeur du rang 2 dans le nombre 548 ?

La valeur du rang 2 dans le nombre 548 est 5.

1.2 La numération binaire (base 2)

Cette numération est due aux travaux des mathématiciens Neper (Napier), Leibniz et Boole. Cette numération a été définie bien avant l'invention de l'ordinateur mais s'est révélée remarquablement adaptée à cet outil. Le but était de représenter les nombres avec le moins de symboles possibles, soit 2 valeurs. Le « 0 » et le « 1 », appelés BIT (BInary digiT), c'est le plus petit élément d'information stockable par un ordinateur. Cela correspond à deux états possibles d'un circuit électrique (ouvert ou fermé).

$10011101_{(2)} = 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$10011101_{(2)} = 128 + 0 + 0 + 16 + 8 + 4 + 0 + 1 = 157_{(10)}$

En binaire, 157 est représenté par un nombre de 8 bits, soit un octet. Le bit de rang le plus petit est le bit de « poids faible », celui du rang le plus grand est le bit de « poids fort ».

1.3 La numération hexadécimale (base 16)

L'ordinateur ne manipule que des « 0 » et des « 1 », mais c'est lourd à traduire. Par exemple, $2012_{(10)} = 11111011100_{(2)}$ soit 11 bits ! On travaille donc un nouveau codage qui utilise 16 symboles. Les 10 chiffres usuels et les 6 premières lettres de l'alphabet en majuscule, soit : 0-1-2-3-4-5-6-7-8-9-A(10)-B(11)-C(12)-D(13)-E(14)-F(15)

$9D_{(16)} = 9 \times 16^1 + 13 \times 16^0 = 144 + 13 = 157_{(10)}$

La numération hexadécimale est une écriture plus condensée qui consiste à regrouper les quartets.

Exemple :

$9_{(16)} = 1001_{(2)}$

$D_{(16)} = 1101_{(2)}$

$9D_{(16)} = 10011101_{(2)}$

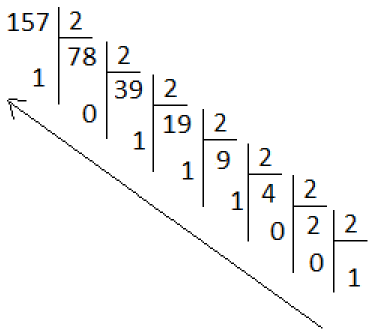
2. PASSAGE D'UNE BASE A UNE AUTRE

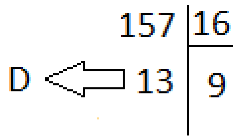
2.1 Tableau Décimal-Binaire-Hexadécimal

Compléter le tableau ci-dessous.

Décimal		Binaire				Hexadécimal
	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
1	0	**1**	**0**	**1**	**0**	A
1	1	**1**	**0**	**1**	**1**	B
1	2	**1**	**1**	**0**	**0**	C
1	3	**1**	**1**	**0**	**1**	D
1	4	**1**	**1**	**1**	**0**	E
1	5	**1**	**1**	**1**	**1**	F

2.2 Conversion d'un décimal en binaire et hexadécimal





On lit le nombre en "remontant", soit: 9 D

On lit le nombre en "remontant", soit: 1 0 0 1 1 1 0 1

2.3 Conversion d'un hexadécimal en binaire et décimal

On lit dans le tableau la correspondance des valeurs.
$9_{(16)}$ correspond à $1001_{(2)}$
$D_{(16)}$ correspond à $1101_{(2)}$
Donc $9D_{(16)} = 10011101_{(2)}$

$9D_{(16)} = 9 \times 16^1 + 13 \times 16^0 = 144 + 13 = 157_{(10)}$

2.4 Conversion d'un binaire en hexadécimal et décimal

On regroupe les bits par quartets et on lit la correspondance des quartets dans le tableau.
$1\ 0\ 0\ 1\ 1\ 1\ 0\ 1_{(2)}$ donne les quartets 1 0 0 1 1 1 0 1 soit $9D_{(16)}$

$10011101_{(2)} = 1 \times 2^7 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 = 157$

2.5 Exercices :

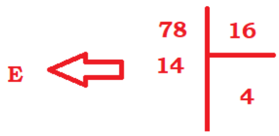
- Convertir 11011 en décimal puis en hexadécimal.

$11011_{(2)} = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 =$
$11011_{(2)} = 16 + 8 + 0 + 2 + 1 = 27$

On regroupe les bits par quartets : 0001 1011 = 1B

On lit le nombre en "remontant" soit: $27_{(10)} = 1B_{(16)}$

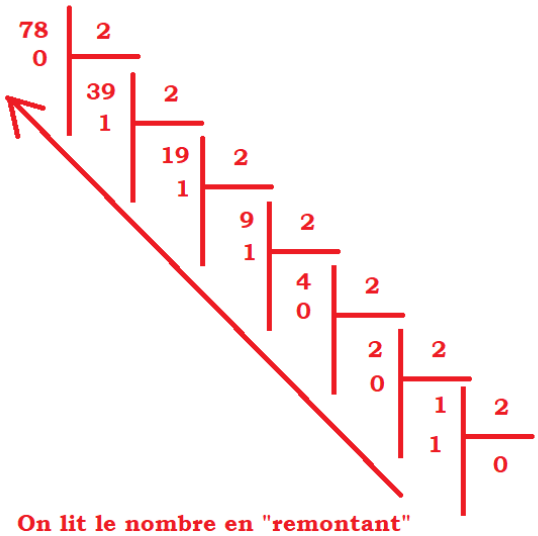
- Convertir 78 en hexadécimal puis en binaire.



On lit le nombre en "remontant" soit: $78_{(10)} = 4E_{(16)}$

On utilise le tableau, $4_{(16)}$ correspond à $0100_{(2)}$ et $E_{(16)}$ correspond à $1110_{(2)}$
Soit : $4E_{(16)} = 0100\ 1110_{(2)}$

Ou alors on effectue des divisions successives par 2 :



On lit le nombre en "remontant"
Donc, $78_{(10)} = 0100\ 1110_{(2)}$

- Convertir 23A en binaire puis en décimal.

On utilise le tableau : $2_{(16)}$ correspond à $0010_{(2)}$, $3_{(16)}$ correspond à $0011_{(2)}$ et $A_{(16)}$ correspond à $1010_{(2)}$.
Donc, $23A_{(16)} = 0010\ 0011\ 1010_{(2)}$

En décimal :
$23A_{(16)} = 2\times16^2 + 3\times16^1 + 10\times16^0 = 512 + 48 + 10 = 570_{(10)}$
Ou :
$23A_{(16)} = 0010\ 0011\ 1010_{(2)} =$
$23A_{(16)} = 0\times2^{11} + 0\times2^{10} + 1\times2^9 + 0\times2^8 + 0\times2^7 + 0\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 =$
$23A_{(16)} = 0 + 0 + 512 + 0 + 0 + 0 + 32 + 16 + 8 + 0 + 2 + 0 = 570_{(10)}$

- Quel est le plus grand nombre (décimal) que l'on peut coder avec un octet ?

Un octet comporte 8 bits, le plus grand nombre correspond donc à $11111111_{(2)}$.
Donc, $1111\ 1111_{(2)} = FF_{(16)} = 15\times16^1 + 15\times16^0 = 240 + 15 = 255_{(10)}$
Ou,
$1111\ 1111_{(2)} = 1\times2^7 + 1\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$
$1111\ 1111_{(2)} = 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255_{(10)}$

- Combien faut-il de symboles hexadécimaux pour coder un nombre binaire de 8 bits ?

Un nombre binaire de 8 bits est composé de 2 quartets, donc il faut 2 symboles hexadécimaux pour le coder.

3. CONVERSIONS AVEC UN TABLEUR

3.1 Conversion binaire-décimal

	A	B	C	D	E	F	G	H	I
1				Nombre binaire					Nombre décimal
2	1	1	0	0	1	1	0	1	205
3									

Quelle formule faut-il écrire dans la case I2 pour que son contenu soir égal à la valeur décimale du nombre binaire écrit dans les cases A2 à H2 ?

SOMME =A2*2^7 + B2*2^6 + C2*2^5 + D2*2^4 + E2*2^3 + F2*2^2 + G2*2^1 + H2* 2^0

	A	B	C	D	E	F	G	H	I
1				**Nombre binaire**					**Nombre décimal**
2	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**=A2*2^7 + B2*2^6 + C2*2^5 + D2*2^4 + E2*2^3 + F2*2^2 + G2*2^1 + H2* 2^0**
3									
4									

Ecrire cette formule et vérifier sa validité avec quelques exemples.

	A	B	C	D	E	F	G	H	I
1				**Nombre binaire**					**Nombre décimal**
2	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**204**
3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
4	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**255**
5	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**127**

3.2 Conversion décimal -binaire

Convertir le nombre décimal 205 en nombre binaire par la méthode des divisions successives par 2.

	A	B	C	D	E	F	G	H	I
1	Nombre Dec	bit1	bit2	bit3	bit4	bit5	bit6	bit7	bit8
2	205	102	51	25	12	6	3	1	0
3		1	1	0	0	1	1	0	1
4									

Utiliser la fonction ENT(nombre), qui arrondi le nombre à l'entier immédiatement inférieur. (Ligne2)

Et la fonction MOD (nombre1 ; nombre2) qui donne le reste de la division du nombre1 par le nombre2. (Ligne3)

	A	B	C	D	E	F	G	H	I
1	**Nombre Décimal**	**bit1**	**bit2**	**bit3**	**bit4**	**bit5**	**bit6**	**bit7**	**bit8**
2		**ENT(A3/2)**	**ENT(B3/2)**	**ENT(C3/2)**	**ENT(D3/2)**	**ENT(E3/2)**	**ENT(F3/2)**	**ENT(G3/2)**	**ENT(H3/2)**
3	**255**	127	63	31	15	7	3	1	0
4		MOD(H3;2)	MOD(G3;2)	MOD(F3;2)	MOD(E3;2)	MOD(D3;2)	MOD(C3;2)	MOD(B3;2)	MOD(A3;2)
5		1	1	1	1	1	1	1	1
6		**ENT(A7/2)**	**ENT(B7/2)**	**ENT(C7/2)**	**ENT(D7/2)**	**ENT(E7/2)**	**ENT(F7/2)**	**ENT(G7/2)**	**ENT(H7/2)**
7	**127**	63	31	15	7	3	1	0	0
8		MOD(H7;2)	MOD(G7;2)	MOD(F7;2)	MOD(E7;2)	MOD(D7;2)	MOD(C7;2)	MOD(B7;2)	MOD(A7;2)
9		0	1	1	1	1	1	1	1

Le bit1 du tableau est-il le bit de « poids fort » ou le bit de « poids faible » ?

Le bit1 est le bit de « poids fort » car il a le plus grand rang.

3.3 Conversion en utilisant les fonctions prédéfinies du tableur

Le tableur comporte des fonctions qui permettent de faire directement les conversions.

BINDEC convertit un nombre binaire en un nombre décimal
BINHEX convertit un nombre binaire en un nombre
héxadécimal DECBIN convertit un nombre décimal en un
nombre binaire DECHEX convertit un nombre décimal en un
nombre héxadécimal HEXBIN convertit un nombre héxadécimal
en un nombre binaire HEXDEC convertit un nombre
héxadécimal en un nombre décimal