Compléments et opérations sur les Bases 2 et 16

Nous avons déjà appris à :

- Passer d'une écriture en base 2 à une écriture en base 10, et réciproquement.
- Passer d'une écriture en base 16 à une écriture en base 10, et réciproquement.

Nous allons à apprendre à :

- Passer d'une écriture en base 2 à une écriture en base 16, et réciproquement.
- Additionner deux nombres entiers écrits en base 2 ou en base 16.
- Evaluer le nombre de bits nécéssaires à l'écriture en base 2 d'un entier, de la somme ou du produit de deux entiers

1. Base 2 <> Base 16

Une première méthode 😢

Pour passer de la base 2 à la base 16 ou de la base 16 à la base 2, on peut passer par la base 10.

Exercice 1 : On veut convertir `1a5c` en base 2.

1. Convertir `1a5c` en base 10.
2. Convertir le nombre obtenu en base 2.

Réponse :

1.
2.

Exercice 2: Convertir `10111001010` en base 16.

Réponse :

-
-

Synthèse :

- Cette méthode fonctionne dans tous les cas mais elle peut être très longue...

Une deuxième méthode 😀

- Elle s'appuie sur le fait que 16 et 2 sont des puissances de deux :
  - $16 = 2^4$
  - $2 = 2^1$
- On peut ainsi écrire chacun des chiffres de la base 16 avec 4 bits.

Exercice 3 :

Compléter le tableau ci-dessous en remplaçant les ?. Il est à connaître par coeur ou à retrouver rapidement.

Binaire	Héxa	Binaire	Héxa	Binaire	Héxa	Binaire	Héxa
0000	?	0100	?	1000	?	1100	?
0001	?	0101	?	1001	?	1101	?
0010	?	0110	?	1010	?	1110	?
0011	?	0111	?	1011	?	1111	?

De la base 2 vers la base 16

- On regroupe les chiffres binaires par 4 en partant de la droite en complétant éventuellement avec des `0` à gauche.
- On identifie chacun de ces regroupements à son chiffre héxadécimal à l'aide du tableau ci-dessus.

Exemple : `11010010100100010`

- On regroupe les chiffres binaires par 4 : `0001 1010 0101 0010 0010`
- On identifie chacun de ces regroupements à son chiffre héxadécimal : `1 A 5 2 2`
- L'écriture en base 16 est donc : `1A522`

Exercice 4 :

1. Exécuter la cellule ci-dessous et convertir les nombres affichés par python en base 16

In [1]:

```
%run -i "2-16.py"
```

```
a)  1000100101001
b)  1011111100010
c)  111101110110
```

Réponses :

- a)
- b)
- c)

1. En utilisant les fonctions `hex()` et `int()` , écrire des instructions pour vérifier les réponses.

In [1]:

```
#Réponses
```

De la base 16 vers la base 2

- On convertit chacun des chiffres en base 2 sur 4 bits.

Exemple : `A10B`

- On convertit chacun des chiffres sur 4 bits. `1010 0001 0000 1011`
- L'écriture en base 2 est `1010000100001011`

Exercice 5 :

1. Exécuter la cellule ci-dessous et convertir les nombres affichés par python en base 2

In [1]:

```
%run -i "16-2.py"
```

```
a)  13f7
b)  fd4
c)  10d4
```

Réponses :

- a)
- b)
- c)

1. En utilisant les fonctions `hex()` et `int()` , écrire des instructions pour vérifier les réponses.

In [2]:

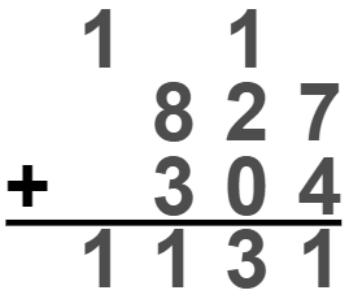
```
#Réponses
```

2. Additionner deux nombres

En base 10

Addition posée :

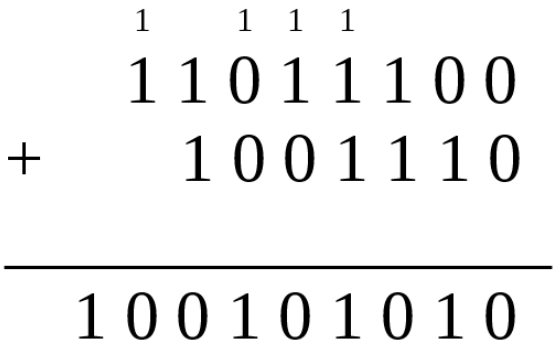
- On additionne les chiffres de même rang (unités , dizaines, centaines,...)
- On garde une retenue au rang suivant si la somme dépasse 10.



En base 2

- On additionne les chiffres de même rang (unités , deuxaines, quatraines,...).
- On garde une retenue au rang suivant si la somme dépasse `10` (c'est à dire 2).
- Ainsi chaque rang, les résultats possibles sont :

Chiffre 1	Chiffre 2	Retenue entrante	Chiffre résultat	Retenue sortante
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



- On retrouve les résultats du circuit logique **XOR**

Exercice 6:

A quelle addition en base 10, l'exemple ci-dessus correspond-il ?

Réponse :

Exercice 7 :

Exécuter le code ci-dessous et calculer le résultat à l'aide de la méthode précédente:

In [ ]:

```
%run -i "add2.py"
```

```
    1011000001001
+      1110100001
    ---------------
             ?
```

En base 16

- On additionne les chiffres de même rang (unités , seizaines, deux cent cinquante sixaines,...).
- On garde une retenue au rang suivant si la somme dépasse `f` (c'est à dire 16).
- Dans l'exemple ci-contre, `b + f` correspond à $11 + 15 = 26 = 1 \times 16 + 10$, c'est à dire `1a` .
- C'est pour cela que l'on écrit `a` comme chiffre résultat et que l'on retient `1` .
- Il est intéréssant de mémoriser les valeurs en base 10 des chiffres héxadécimaux(en particulier de `a` à `f` :

$$\begin{array}{cccc} & 1 & 1 & \\ & a & 5 & f & 1 \\ + & & c & b & 9 \\ \hline & b & 2 & a & a \end{array}$$

Décimal	*Hexa*	Décimal	*Hexa*	Décimal	*Hexa*	Décimal	*Hexa*
0	*0*	4	*4*	8	*8*	12	*C*
1	*1*	5	*5*	9	*9*	13	*D*
2	*2*	6	*6*	10	*A*	14	*E*
3	*3*	7	*7*	11	*B*	15	*F*

Exercice 8 :

A quelle addition en base 10 l'exemple ci-dessus correspond-il ?

Réponse :

Exercice 9 :

Exécuter le code ci-dessous et calculer le résultat à l'aide de la méthode précédente:

In [ ]:

```
%run -i "add16.py"
```

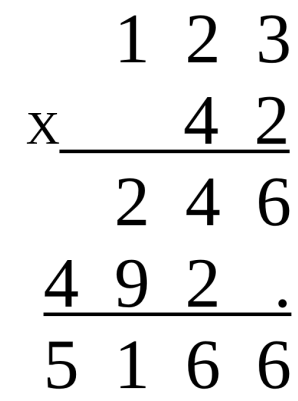
```
    1ba4
+   11d4
    ---------------
             ?
```

3. Multiplier deux nombres

En base 10

Multiplication posée :

- $123 \times 42 = 123 \times 40 + 123 \times 2$
- On écrit d'abord le résultat de 123×2
- Puis en dessous le résultat de 123×4 $dizaines$
- On additionne les deux résultats intermédiaires

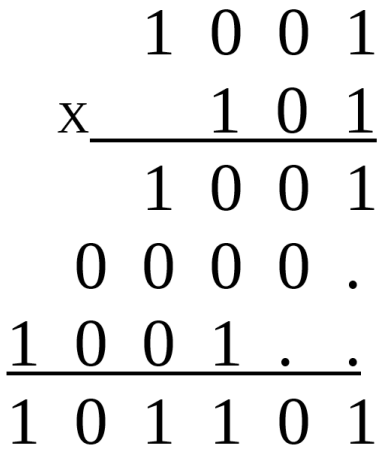


En base 2

- On garde exactement le même principe mais avec les chiffres `0` et `1`
- Ainsi à chaque calcul effectué, les résultats possibles sont :

Chiffre 1 x	Chiffre 2	= Chiffre résultat
0	0	0
0	1	0
1	0	0
1	1	1

- On retrouve les résultats d'un circuit logique **AND**.
- Le principe est le même en base 16 mais il y a plus de calculs à mémoriser, autant repasser par la base 2 !



Exercice 10 :

Vérifez le résultat en base 10 de l'opération donnée en exemple.

Réponse:

Exercice 11 :

Exécuter le code ci-dessous et calculer le résultat à l'aide de la méthode présentée.

In [ ]:

```
%run -i  "multi2.py"
```

```
    10000
x     101
   ________
          ?
```

4. Evaluer le nombre de bits nécéssaires à l'écriture en base 2 :

Les opérations effectués par un ordinateur se font sur un nombre de bits fixés(8,16,32,64,...). Il peut être important de mesurer le nombre de bits nécéssaires en fonction des valeurs utilisées et des opérations arithmétiques à effectuer.

Entiers naturels

Exercice 12 :

Partie A : Combien de bits sont nécéssaires pour représenter :

1. L'entier 0 en base 2?
2. L'entier 255 en base 2 ?
3. L'entier 256 en base 2 ?
4. L'entier 2^n avec $n \in \mathbb{N}$ en base 2 ?
5. L'entier $2^n - 1$ avec $n \in \mathbb{N}$ en base 2 ?

Partie B:

1. Encadrer le nombre 29356 entre 2 puissances de 2.
2. En déduire le nombre de bits nécéssaires pour le représenter.
3. Déterminer le nombre de bits nécéssaires pour représenter 6431.

Réponse:

Partie A :

1.
2.
3.
4.
5.

Partie B :

1.
2.
3.

A retenir :

- Avec n bits, on représente 2^n informations différentes en particulier tous les entiers naturels de 0 à $2^n - 1$
- Pour évaluer le nombre de bits nécéssaires à la représentation d'un entier naturel N :
  - Si N est une puissance de 2 : il existe un entier k, tel que $2^k = N$, il faut $k + 1$ bits.
  - Sinon, N est strictement compris entre deux puissances de 2 : il existe un entier k tel que $2^k < N < 2^{k+1}$, il faut $k + 1$ bits.
- **Autrement dit le nombre de bits nécéssaires est égal à $k + 1$ ou k est le plus petit entier tel que $2^k \leq N$.**

Pour aller plus loin :

- Le nombre de bits nécéssaires au codage d'un entier naturel N peut se calculer à l'aide d'un formule faisant intervenir le logarithme en base 2 (du nom d'une fonction que vous étudierez en terminale).
- Ainsi le nombre de bits nécéssaires est égal à la partie entière de $log_2(N) + 1$
- Exemple (voir ci-dessous) : Pour coder 12341, on calcule $log_2(12341) \approx 13.59$. Il faut 14 bits.

In [2]:

```
from math import *
print(log(12341,2))
```

```
13.591171681377787
3.0
```

Exercice 13 :

1. Combien de bits sont nécéssaires pour représenter les 26 lettres de l'alphabet en minuscule ?
2. Combien de bits sont nécéssaires pour coder 16777216 couleurs ?
3. Combien de bits sont nécéssaires pour représenter chacune des 34968 communes françaises par un code binaire unique ?
4. On veut attribuer un code unique à chacun des 66524000 Français. Combien de bits sont nécéssaires pour représenter ce code ?

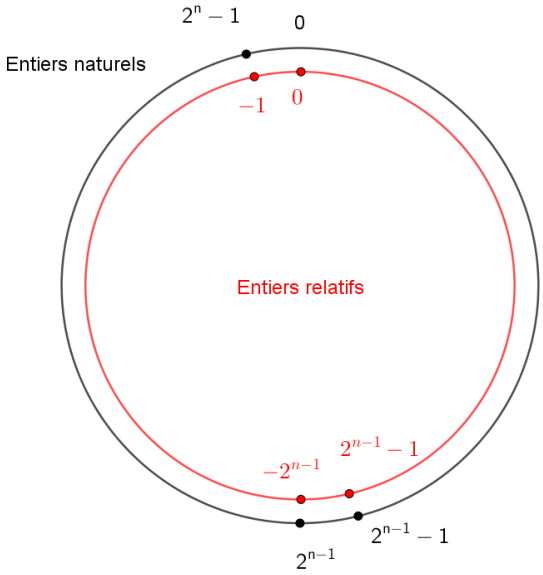
Réponse :

1.
2.
3.
4.

Entier relatifs codés en C2

Rappel :

Pour le complément à deux (two's complement) sur n bits, on dispose de 2^n états possibles, avec lesquels on représente :



- Les entiers positifs de 0 à $2^{n-1} - 1$
- Les entiers négatifs de -2^{n-1} à -1
- Les entiers relatifs positifs sont représentés comme les entiers naturels.
- Un entier relatif x strictement négatif est représenté comme l'entier naturel $x + 2^n$ en base 2.

Méthode :

On veut savoir combien de bits ssont nécéssaires pour coder un entier relatif N.

- Lorsque N est positif: On applique la même méthode que précédemment et ajoute le bit de poids fort à gauche qui indique le signe (dans ce cas `0`)
- Lorsque N est strictement négatif :
  - Si N est est de la forme -2^k, $k \in \mathbb{N}$, alors il faut k bits $+1$ bit pour le signe (dans ce cas `1`), c'est à dire $k + 1$ bits
  - Sinon , il existe k tel que $-2^{k+1} < N < -2^k$, il faut $k + 1$ bits $+1$ bit pour le signe (dans ce cas `1`), c'est à dire $k + 2$ bits.

Exemple :

1. Combien de bits sont nécéssaires pour coder 129 en C2 ?
   - $2^7 < 129 < 2^8$, il faut $7 + 1 + 1 = 9$, 9 bits.
2. Combien de bits sont nécéssaires pour coder -129 en C2 ?
   - $-2^8 < -129 < -2^7$, il faut 9 bits.
3. Combien de bits sont nécéssaires pour coder -128 en C2 ?
   - $-128 = -2^7$.Il faut 8 bits.

Exercice 14 :

1. Combien de bits sont nécéssaires pour coder 1341 en C2 ?
2. Combien de bits sont nécéssaires pour coder -1024 en C2 ?
3. Combien de bits sont nécéssaires pour coder -2047 en C2 ?

Réponse:

1.
2.
3.

Somme de deux entiers naturels

Exercice 15:

1. Effectuer les additions suivantes :
   - `1001` + `100`
   - `1101` + `111`
   - `1010` + `11`
   - `1111` + `1111`

1. Compléter le tableau ci-dessous :

Nombre de chiffres du premier opérande	Nombre de chiffres du deuxième opérande	Nombre de chiffres du résultat
-	-	-
-	-	-
-	-	-
-	-	-

1. Que constate-t-on ?

1. On additionne deux nombres à 8 bits.

   - Combien de bits au minimum comporte le résultat ?
   - Combien de bits au maximum comporte le résultat ?

1. On additionne un nombre à n bits et un nombre m bits. On suppose que $m \leq n$.

   - Combien de bits au minimum comporte le résultat ?
   - Combien de bits au maximum comporte le résultat ?

Réponses :

1.
   -
   -
   -
   -

1. Voir tableau

1.
2.
   -
   -
3.

A retenir :

- Lorsque l'on addtionne un nombre à m chiffres binaires et un nombre à n chiffres binaires (on suppose que $m \leq n$), le résultat est un nombre à n ou $n+1$ bits.

Exercice 16:

On considère un capteur dans un système embarqué d'un véhicule volant. Ce capteur mesure l'accélération horizontale qui est ensuite converti sur 8 bits :

1. A l'instant t_1, l'accélération mesurée est 60. Convertir ce nombre en base 2.
2. A l'instant t_2, l'accélaration augmente de 240. Quel sera le résultat converti en base 2 ?
3. Que va comprendre le système embarqué ?

Réponse :

1.
2.
3.

A retenir :

- Le dépassement de capacité(overflow) peut se produire lorsque l'espace mémoire prévu est sous dimensionné par rapport aux valeurs que l'on souhaite représenter. Cela peut avoir de lourdes conséquences...
- Un bug à 370 millions d'€ : https://www.youtube.com/watch?v=gp_D8r-2hwk (https://www.youtube.com/watch?v=gp_D8r-2hwk)
- Dans le cas de l'ajout de deux entiers codé en C2 :
  - Si les deux entiers sont de signes contraires, alors il n'y a aucun risque de dépassement de capacité.
  - S'ils sont de même signe, il peut y avoir dépassement de capacité, comme pour les entiers naturels.

Produit de deux entiers naturels

Exercice 17 :

Exécutez le code ci-dessous autant de fois que nécéssaire pour compléter la partie "A retenir" ci-après.

In [7]:

```
%run -i  "multi2x.py"
```

```
1) 111010 x 101110 = 101001101100
2) 110011 x 100110 = 11110010010
3) 10010 x 1010 = 10110100
4) 110001 x 11111 = 10111101111
5) 101101 x 11110 = 10101000110
```

A retenir

Lorsque l'on multiplie un nombre à m bits par un nombre à n bits(on suppose que $m \leq n$), le résultat comporte:

- Au minimum ? bits
- Au maximum ? bits

FIN