I- Codage binaire

1) Passer de la notation binaire à la notation décimale.

1011_2 est l'addition de droite à gauche de une unité, une *deuzaine*, zéro *quatraine*, une *huitaine*.

Soit mathématiquement:

$$1011_2 = 1\times2^3+0\times2^2+1\times2^1+1\times2^0$$

$$1011_2 = 8+0+2+1$$

$$1011_2 = 11_{10}$$

2) Passer de la notation décimale à la notation binaire

On regroupe les objets par paquets de 2 en réalisant des divisions successives jusqu'à obtenir un quotient égal à 0.

Trouver en base deux la représentation du nombre 13_{10}

```
    13| 2
       |---
     1 | 6 | 2
           |---
         0 | 3 | 2
               |---
             1 | 1 | 2
                   |---
                 1 | 0
```

L'écriture du nombre se fait alors de droite à gauche :

$$13_{10} = 1101_2$$

On peut vérifier le résultat:

$$1101_2 = 1\times2^3+1\times2^2+0\times2^1+1\times2^0$$

$$1101_2 = 1\times2^3+1\times2^2+0\times2^1+1\times2^0$$

$$1101_2 = 8+4+0+1$$

$$1101_2 = 13_{10}$$

II-L’octet

Les mémoires actuelles sont toutes composées de cellules mémoires capables de retenir un bit. En mettant pleins de ces cellules dans un seul composant, et en mettant quelques circuits électroniques pour gérer le tout, on obtient une mémoire. L'état d'un circuit mémoire, se décrit par une suite finie de 0 et de 1, que l'on appelle un mot. Par exemple, le mot 100 décrit l'état d'un circuit composé de trois circuits mémoires un bit, respectivement dans l'état 1, 0 et 0.
Dans la mémoire des ordinateurs les circuits mémoire un bit sont souvent groupés par huit : les octets. On utilise souvent des nombres exprimés en notation binaire, c'est-à-dire en base deux, sur un, deux, quatre ou huit octets, soit 8, 16, 32 ou 64 bits. Un octet permet de représenter 2^8 , c'est-à-dire 256 , valeurs différentes. Un ou plusieurs octets permettent ainsi de coder des valeurs numériques ou des caractères.

III-Codage hexadécimal

La notation binaire bien qu'adaptée aux composants électroniques, ne l'est pas du tout pour l'homme. La base 16, le système hexadécimal rend l'utilisation du binaire plus humaine. Un chiffre hexadécimal est un mot de 4 bits puisque $2^4=16$ Le système hexadécimal est utilisé notamment en électronique numérique et en informatique car il est particulièrement commode et permet un compromis entre le code binaire des machines et une base de numération pratique à utiliser pour les ingénieurs. En effet, chaque chiffre hexadécimal correspond exactement à quatre chiffres binaires (ou bits), rendant les conversions très simples et fournissant une écriture plus compacte.

Les chiffres utilisés sont: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Voici les correspondances des chiffres entre les bases hexadécimale, décimale et binaire.

Chiffre hexadécimal	Représentation décimale	Représentation binaire
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

1) Passer de la notation binaire à la notation hexadécimale

Soit le mot binaire de 16 bits: `1010101111100001`

On est d'accord, en binaire, c'est *inhumain*, par contre en hexadécimal, cela devient beaucoup plus lisible et manipulable: $\mathbf{1010\,1011\,1110\,0001_2 = \mathit{ABE}\,1_{16}}$

Ou si l'on tient vraiment à notre bonne vielle base 10:

$$\mathbf{\mathit{ABE}\,1_{16} = 10 \times 16^3 + 11 \times 16^2 + 14 \times 16^1 + 1 \times 16^0 = 44001_{10}}$$

Pour passer de l'écriture binaire à l'écriture hexadécimale, il suffit de regrouper les chiffres 4 par 4.

Représentation du mot binaire 101101

Base		**Regroupement, de droite à gauche, par bloc de 4 bits**	
2	$\mathbf{101011_2}$	**0010**	**1011**
16		**2**	***D***

Soit: $\mathbf{101011_2 = 2\,\mathit{D}_{16}}$

2) Passer de la notation décimale à la notation hexadécimale

On regroupe les objets par paquets de 16 en réalisant des divisions successives jusqu'à obtenir un quotient égal à 0.

Trouver en base seize la représentation du nombre $\mathbf{286_{10}}$

```
     286  | 16
          |---
    14(e)| 17 | 16
               |---
             1 | 1 | 16
                   |---
                 1 | 0
```

L'écriture du nombre se fait alors de droite à gauche en remplaçant les nombres décimaux par leurs écritures hexadécimales: $\mathbf{286_{10} = 11\,\mathit{E}_{16}}$

On peut vérifier le résultat :

$$\mathbf{11\,\mathit{E}_{16} = 1 \times 16^2 + 1 \times 16^1 + 14 \times 16^0}$$
$$\mathbf{11\,\mathit{E}_{16} = 1 \times 256 + 1 \times 16 + 14 \times 1}$$
$$\mathbf{11\,\mathit{E}_{16} = 256 + 16 + 14}$$
$$\mathbf{11\,\mathit{E}_{16} = 286_{10}}$$

IV-Opération arithmétique

Les règles utilisées en base dix s'appliquent de la même façon dans les autres bases.

1) Addition binaire

Réaliser l'addition binaire $\mathbf{1011_2 + 1010_2}$, puis vérifier le résultat en base 10.

2) Addition hexadécimale

Réaliser l'addition hexadécimale $\boldsymbol{A803_{16} + 2D35_{16}}$, puis vérifier le résultat en base 10.

3) Soustraction binaire

Réaliser la soustraction binaire $\mathbf{1011_2 - 1010_2}$, puis vérifier le résultat en base 10.

4) Soustraction hexadécimale.

Réaliser la soustraction hexadécimale $\boldsymbol{A803_{16} + 2D35_{16}}$, puis vérifier le résultat en base 10.

5) Conversions et opérations en python

Bien entendu comme ce genre d'opérations est courante en informatique, python possède des fonctions pour manipuler les nombres entiers en base décimale **int** , en base deux **bin** , en base seize **hex**