PREMIERE GENERALE

NUMERIQUE ET SCIENCES DU NUMERIQUE

CODAGE D'UN NOMBRE ENTIER POSITIF

- ▷ **Histoire de l'informatique**
- ▷ **Représentation des données : types et valeurs de base**
- ▷ Représentation des données : types construits
- ▷ Traitement de données en tables
- ▷ Interactions entre l'homme et la machine sur le Web
- ▷ Architectures matérielles et systèmes d'exploitation
- ▷ Langages et programmation
- ▷ Algorithmique

Jadis, les peuples possédaient leurs propres systèmes de numération. Les égyptiens utilisaient un **système décimal** sans le chiffre `0`, tandis que les sumériens utilisaient un système en base `60`. Les grecs et les romains utilisaient un système de comptage additif.

Les babyloniens, introduisent dès 1800 av J.C. l'idée de **position**, c'est-à-dire qu'un chiffre n'a pas la même valeur selon sa poistion dans le nombre. C'est en Inde, au IVe siècle, que les mathématiciens associent la numération de position,le système décimal et le symbole zéro qui signifie rien. Ce système rend les calculs possibles, ce qui préfigure l'élaboration de la machine à calculer.

Le philosophe et mathématicien allemand **Leibnitz** conçoit la première machine à calculer, capable de multiplier des nombres de 8 chiffres. Pour représenter les données, il propose le **système de numération binaire** composé de **zéros et de uns** qu'il juge favorable au calcul mécanique.

1 – INTRODUCTION

Pour représenter un nombre entier, il est possible d'utiliser différentes bases (ou systèmes de numération), parmi lesquelles on peut citer : la **base 10** (**décimale**) ; la **base 2** (**binaire**) qui est la base utilisée en informatique et dans l'**algèbre de Boole** ; la **base 16** (base **hexadécimale**) qui est une base utilisée en informatique.

Naturellement, nous utilisons la **base 10**, appelée **base décimale**. Pratique pour compter sur les 10 doigts de nos mains, la base décimale utilise les caractères `{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}`. À partir de 10 unités, nous utilisons des dizaines, des centaines, puis des milliers… La **position** appelée **rang** de chaque chiffre est essentielle.

Pour décomposer un entier dans la base 10, il faut identifier le rang de chaque chiffre et le multiplier par la puissance de 10 appropriée, puis faire la somme de tous les termes de la décomposition :

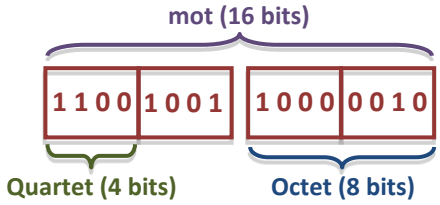
Rang	3	2	1	0
Chiffre	7	8	4	6

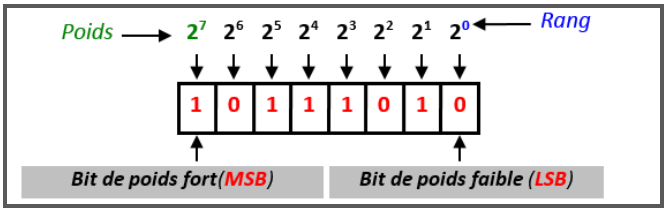
7846_{10} = ...

2 – SYSTEME BINAIRE

2.1 – Présentation

Un système numérique doit, avant de pouvoir manipuler les nombres, les textes, les images ou les sons, les représenter comme des **suites de 0 et de 1**. La valeur 0 ou 1 est appelée **booléen**, **chiffres binaires** ou encore **bit** (**binary digit**). Une suite de chiffres binaires est appelée **mot binaire**. Suivant le processeur, la taille du mot sera différente. Les mots les plus courants ont une taille de **8** (**octet** ou **byte** en anglais), **16**, **32** ou **64 bits**.





Dans un système binaire ou **base 2**, il n'y a que **deux chiffres possibles** : **0** et **1**. Le système de numération binaire est indiqué par l'**indice 2** ou par les **symboles 0b**.

$$(1000\ 1011)_2 = 0b1000\ 1011$$

2.2 – Conversion binaire vers décimal

Pour trouver l'équivalent décimal d'un nombre binaire, il suffit de faire la **somme des produits de chaque bit par le poids de son rang**. Pour un mot binaire de n bits :

$$(b_{n-1}\ b_{n-2}\ \dots\ b_2\ b_1\ b_0)_2 = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

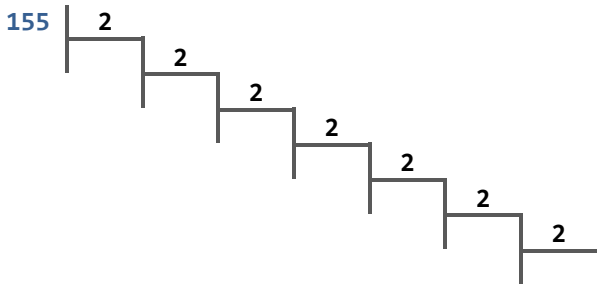
Exemple pour un octet (n = 8)

0b10111010 = ... **=** .....

2.2 – Conversion décimal vers binaire

Cette conversion peut être réalisée par la méthode des **divisions successives par 2** (division euclidienne) ou la méthode des **substitutions.**

Divisions successives par 2



155 = ...

Substitutions

Poids									
Octet									
Reste									

155 = ...

3 – LE SYSTEME HEXADECIMAL

3.1 – Présentation

Le langage binaire est difficilement manipulable par l'homme pour de grandes séries binaires. On utilise alors le **système hexadécimal** (**base 16**). Celui-ci comporte **16 symboles** (**base 16**) : **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E,** et F. Pour indiquer la base **16**, on peut utiliser les indices **16** mais en programmation on place les symboles **0x** devant le nombre : **(AA)16 = 0xAA**.

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

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. Chauqe chiffre hexadécimal correspond à **quatre 4 bits** ce qui permet une écriture plus compacte. L'hexadécimal a été utilisé la première fois en **1956** par les ingénieurs de l'ordinateur **Bendix G-15**.

- Bendix G-15 -

3.2 – Conversion binaire vers hexadécimal

Exemple

0b **0110** **1111** 0011 **Regroupement en quartet**

↓ ↓ ↓

Conversion binaire hexadécimal 0b01101111 = ...

↓ ↓ ↓

0x **Conversion binaire hexadécimal**

3.3 – Conversion hexadécimal vers binaire

Exemple

0x **D** **7** **E**

↓ ↓ ↓

Conversion hexadécimal décimal 0xD7E = ...

↓ ↓ ↓

0b **Conversion décimal binaire**

4 – REPRESENTATION D'UN NOMBRE ENTIER SOUS PYTHON

Pour Python, un nombre est un nombre, peu importe la base. Il peut s'écrire et s'afficher dans une base quelconque.

Tester et **expliquer** les instructions suivantes :

```
>>> bin(42)
>>> hex(42)
>>> 0b101011
>>> 0x2B
>>> int('101010', 2)
>>> int('C2A', 16)
```

5 – EXERCICES

5.1 – Conversions

1. **Convertir** les nombres binaires suivants dans leur équivalent décimal :

`N1 = 0b10101010` ; `N2 = 0b11001100` ; `N3 = 0b111000111000` ; `N4 = 0b100000000`

2. **Convertir** les nombres décimaux suivants dans leur équivalent binaire :

`N1 = 255` ; `N2 = 256` ; `N3 = 1025` ; `N4 = 2012` ; `N7 = 2048`

3. **Convertir** les nombres binaires suivants dans leur équivalent hexadécimal :

`N1 = 0b10101010101` ; `N2 = 0b11111111` ; `N3 = 0b1110001110`

4. **Convertir** les nombres hexadécimaux suivants dans leur équivalent binaire :

`N1 = 0xB3` ; `N2 = 0x1F2` ; `N3 = 0xFF` ; `N4 = 0x2014`

5. **Convertir** les nombres hexadécimaux suivants dans leur équivalent décimal :

`N1 = 0xB3` ; `N2 = 0x1F` ; `N3 = 0xFF`; `N4 = 0x1A5`

6. **Convertir** les nombres décimaux suivants dans leur équivalent hexadécimal :

`N1 = 255` ; `N2 = 256` ; `N3 = 2012` ; `N4 = 2048`

5.2 – Conversion binaire - décimal

On modélise la représentation binaire d'un entier non signé par une chaîne de caractères dont les éléments sont `0` ou 1. Par exemple, la chaîne `'10100110'` représente l'écriture binaire de l'entier dont l'écriture décimale est : `2**7 + 2**5 + 2**2 + 2**1 = 166`.

Compléter la fonction convertir permettant qui renvoie la valeur décimale dont la dont la représentation binaire est donnée la chaine de caractères `chaine`. **Tester** la fonction.

```
def convertir(chaine):
    n = ..................................      # taille de la chaine
    val_dec = ..................        # Initialisation à 0 de la variable val_dec
    for i in range(0, n):
        rang = n - 1 -i       # calcul du rang de 7 à 0
        val_dec = val_dec + ..................................................... # Calcul valeur décimale
    return val_dec
```

```
>>> convertir('1010011')
83
```

```
>>> convertir('10000010')
130
```