**Conversion Décimal vers Binaire**

Soit N un nombre Entier.

La division Euclidienne nous dit

Dividende = (diviseur \* quotient) + reste

Le Dividende va commencer par N.  
 Le diviseur sera 2.  
 On note q0 q1 q2 … qN les quotients successifs.  
 On note r0 r1 r2 … rN les restes successifs.  
 Les restes successifs seront le nombre encodé en binaire.  
 On continue jusqu’à ce que qN = 0.

14 / 2 = (q0) 7 (r0) 0  
 (q1) 7 / 2 = (q1) 3 (r1) 1  
 (q2) 3 / 2 = (q2) 1 (r2) 1  
 (q3) 1 / 2 = (q3) 0 (r3) 1  
 (q3 vaut 0, on arrête)

Le nombre 14 en binaire : 1110  
 Vérification : 2^3 + 2^2 + 2^1   
 = 8 + 4 + 2   
 = 14.

N | 2  
 |---  
 r0 | q0 | 2  
 |---  
 r1 | q1 | 2  
 |---  
 r2 | q2 | 2  
 |---  
 rN | qN

Le nombre final sera :

rN ... r2 r1 r0

**Conversion Décimal vers Hexadécimal**

Le principe de la conversion Décimal vers Hexadécimal est le même que le principe Décimal vers Binaire.

En plus il faut utiliser la notation hexadécimale pour chaque chiffre.

Au lieu de diviser par 2, on va diviser par 16.

Convertir le nombre 244 en binaire.

244 / 16 = (q0) 15 + 4 (r0)  
 (q0) 15 / 16 = (q1) 0 + 15 (r1)  
 (q1) vaut 0, on arrête)

Les chiffres du nombre 244 en hexadécimal sont : (15)(4)

Le 15 s’écrit F en hexadécimal.

Le nombre 244 s’écrit en hexadécimal : F4

Vérification :

F4 : 15\*16^1 + 4\*16^0 = 240 + 4 = 244.

Résultat correct.

**Conversion Hexadécimal <-> Binaire**

Chaque chiffre en hexadécimal s’écrit sur 4 bits en binaire.

Pour la conversion binaire -> hexadécimal il faut regrouper les chiffres par groupe de 4.

Pour la conversion hexadécimal -> binaire, chaque chiffre devient un groupe de 4 bits.

F 4

244 (10) = F4 (16) = 1111 0100 (2)

Vérification

1 1 1 1 0 1 0 0

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128 + 64 + 32 + 16 + 4 = 244.

**Limitations et Implémentation**

Prenons un système qui a 8 bits de mémoire.  
On lui envoie la valeur 1 1 1 1 1 1 1 1, soit 255.  
et on lui demande d’ajouter 18 à cette valeur.

1 1 1 1 1 1 1 1 255

+ 1 0 0 1 0 + 18

------------------ ------

**1**  0 0 0 1 0 0 0 1 **#**  17

\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ovverflow \_\_\_\_\_\_/  
 Bit de dépassement de capacité

Le Bit de dépassement est géré par le microproceseur.

Selon les langages de programmation, ce dépassement de capacité est plus ou moins bien géré.

En langage C ce n’est pas évident : 255 + 18 = 17 sur 8 bits.

Dans d’autres de programmation ont prévu un mécanisme plus ou moins performant pour détecter et gérer ce dépassement de capacité.

En langage C, le nombre de bits réservés à une variable de type entier dépend du compilateur et du système auquel le programme est destiné. Les entiers peuvent être signés ou non, selon les besoins.

En Java, il y a plusieurs variables de type entier, les entiers sont toujours signés, et le même nombre de bits sont alloués, quelque soit le système concerné.

En Python, un certain nombre de bits sont réservés pour les nombre entiers. Si la valeur absolue du nombre augmente de trop, Python bascule automaitquement vers un autre mécanisme d’encodage pour les entiers dont la limite théorique dépend de la quantité de mémoire que le processus Python a le droit d’utiliser.