**Conversions entre bases**

Nous avons l’habitude d’effectuer toutes nos opérations arithmétiques à l’aide de nombres codés en **décimal ou** **base 10**, c’est-à-dire à l’aide des **chiffres de 0 à 9.**

Exemple :

192 est équivalent à (192)10 qui est la notation pour dire que le nombre 192 est codé en base 10.

Quelle est la valeur de ce nombre ?

(192)10 peut être vu comme :

2\*100 = 2

+ 9\*101 = 90

+ 1\*102 = 100

Total 2 + 90 + 100 = 192 (CQFD…)

Tout cela pour dire que 192 vaut bien 192 ….

Une autre façon de représenter cela est d’utiliser un abac :

|  |  |  |
| --- | --- | --- |
| 102 = 100 = les centaines | 101 = 10 = les dizaines | 100 = 1 = les unités |
| 1 | 9 | 2 |

Mais, en informatique, comme vous le savez peut-être, on ne travaille pas en base 10 mais bien en **base 2 (appelé binaire).** Ceci parce que la machine ne connaît que **2 chiffres (0 et 1)** qui correspondent au fait que le courant passe ou ne passe pas dans des circuits intégrés. Toutes les informations, y compris bien sur les nombres, sont codées en binaire.

Une donnée de type entier sera représentée par un nombre fini et fixe de bits N.

Si N = 2, il est possible de représenter :

00

01

10

11

Soit 22 = 4 possibilités.

Si N = 3, il est alors possible de représenter 8 nombres (23 = 8), et ainsi de suite.

*Exemples à faire au cours (3 – 4 bits)*

**Conversion binaire (base 2) vers décimal (base 10).**

Exemple :

(10011)2 = ( ? )10

Procédons comme ci-dessus, mais en utilisant des exposants de 2 (au lieu de 10).

(10011)2 =

1\*20 = 1

+ 1\*21 = 2

+ 0\*22 = 0

+ 0\*23 = 0

+ 1\*24 = 16

Total 1 + 2 + 16 = 19

(10011)2 = (19)10

Avec un abac, on obtient :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 25 = 32 | 24 = 16 | 23 = 8 | 22 = 4 | 21 = 2 | 20 = 1 |
|  | 1 | 0 | 0 | 1 | 1 |

*Exercices à faire au cours*

**Conversion décimal (base 10) vers binaire (base 2).**

Exemple :

(126)10 = ( ? )2

Ici nous n’allons plus multiplier mais diviser. Attention : nous allons utiliser la **division entière (notée div)** et le **reste de la division entière ou modulo (noté mod).**

La division entière s’obtient en divisant le nombre en restant dans le domaine des nombres entiers .

Exemples :

6 div 2 = 3

Ici, pas de problème. La division « tombe juste ».

7 div 2 = 3

??? Ce n’est pas juste !!! La division de 7 par 2 vaut 3.5 ! Il s’agit ici de la division dite réelle.

La division entière n’accepte pas de résultat qui ne soit pas un nombre entier. Donc 7 div 2 = 3 et il reste 1 que nous ne savons pas diviser. Ce reste s’appelle le modulo que nous calculons grâce à l’opérateur mod.

Reprenons :

6 div 2 = 3 6 mod 2 = 0 (car il ne reste rien après l’opération div)

7 div 2 = 3 7 mod 2 = 0 (car il reste 1 après l’opération div)

Revenons à notre exemple :

(126)10 = ( ? )2

126 div 2 = 63 126 mod 2 = 0

63 div 2 = 31 63 mod 2 = 1

31 div 2 = 15 31 mod 2 = 1

15 div 2 = 7 15 mod 2 = 1

7 div 2 = 3 7 mod 2 = 1

3 div 2 = 1 3 mod 2 = 1

1 div 2 = 0 1 mod 2 = 1

Prenons tous les restes des divisions successives du dernier jusqu’au premier et nous obtenons le nombre en base 2.

Soit (1111110)2.

*Faites la conversion dans l’autre sens pour vérifier.*

*Exercices à faire au cours.*

**Conversion hexadécimal (base 16) vers décimal (base 10)**

Un autre système de notation des nombres souvent utilisé dans le domaine informatique est la notation **hexdécimale ou en base 16**, c’est-à-dire qui utilise 16 chiffres pour représenter les nombres. Pourquoi utiliser ce système particulièrement abscond ? Parce que, si vous voulez tout représenter en binaire, vous arrivez très rapidement à des nombres tellement longs qu’ils en deviennent très vite illisibles.

**Qu’est-ce que l’hexadécimal ?**

C’est un système qui utilise 16 chiffres pour représenter les nombres. 16 chiffres ??? Mais nous n’en avons que 10 !!! C’est tout simplement parce que nous sommes habitués à notre système décimal ! Il n’a pas été trop difficile de passer du décimal au binaire vu que nous sommes habitués aux chiffres 0 et 1. Passons maintenant à l’hexa.

Les 10 premiers chiffres sont les mêmes qu’en décimal et les 6 suivants seront représentés par des lettres majuscules.

Nous avons donc les chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Qui représentent les valeurs 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Les conversions se font de la même manière que précédemment mais en utilisant des exposants de 16. (C’est un peu plus difficile que les exposants de 2, il faut bien l’admettre…)

Exemple :

(AF1)16 = ( ? )10

1\*160 = 1

+ F\*161 = 240

+ A\*162 = 2560

Total 1 + 240 + 2560 = 2801

(AF1)16 = (2801)10

Avec un abac, cela donne :

|  |  |  |
| --- | --- | --- |
| 162 = 256 | 161 = 16 | 160 = 1 |
| A | F | 1 |

*Exercices à faire au cours*

**Conversion décimal (base 10) vers hexadécimal (base 16)**

Les conversions se font de la même manière que précédemment mais en divisant par 16 (moins facile que diviser par 2, je vous le concède…).

Exemple 7423 div 16 = 463 7423 mod 16 = F

463 div 16 = 28 463 mod 16 = F

28 div 16 = 1 28 mod 16 = C

1 div 16 = 0 1 mod 16 = 1

Prenons tous les restes des divisions successives du dernier jusqu’au premier et nous obtenons le nombre en base 16.

Soit (1CFF)16

*Faites la conversion dans l’autre sens pour vérifier.*

*Exercices à faire au cours.*

*Leur faire faire l’exercice avec base 8 (importante pour le codage des permissions en Unix) et autre base (6 ou 7 pe).*

**Remarques importantes**

Dans toutes les bases, la base s’écrit toujours (10)x

Les bases en puissance de 2 sont meilleures au point de vue codage. En effet, toutes les combinaisons possibles sur N bits sont utilisées. Il n’y a pas de combinaisons « perdues ».

Exemples :

Base 2 1 bit 2 chiffres (0-1)

Base 4 2 bits 4 chiffres (0-3)

Base 8 3 bits 8 chiffres (0-7)

Base 16 4 bits 16 chiffres (0-F)

……

Il est donc important de connaître les puissances de 2 en informatique (bit, byte, Kb, Mb, ….).

Si il reste du temps, faire quelques additions dans les différentes bases (2, 16, 8).

Si il reste *encore* du temps, codage des négatifs (complément à 1, complément à 2).