

# L1 SA & SN Semestre 2

## Traitement Numérique de l'Information (Architecture 1)

### Thème 1 : Numération et codage de l'information

# Plan

## □ Numération

- ❖ Introduction
- ❖ Système de numération – Cas général
- ❖ Addition en base B
- ❖ Changement de système de numération

## □ Représentation de l'information

- ❖ Introduction
- ❖ Représentation des Entiers naturels
- ❖ Représentation des Entiers relatifs
- ❖ Représentation des Réels (un aperçu rapide)

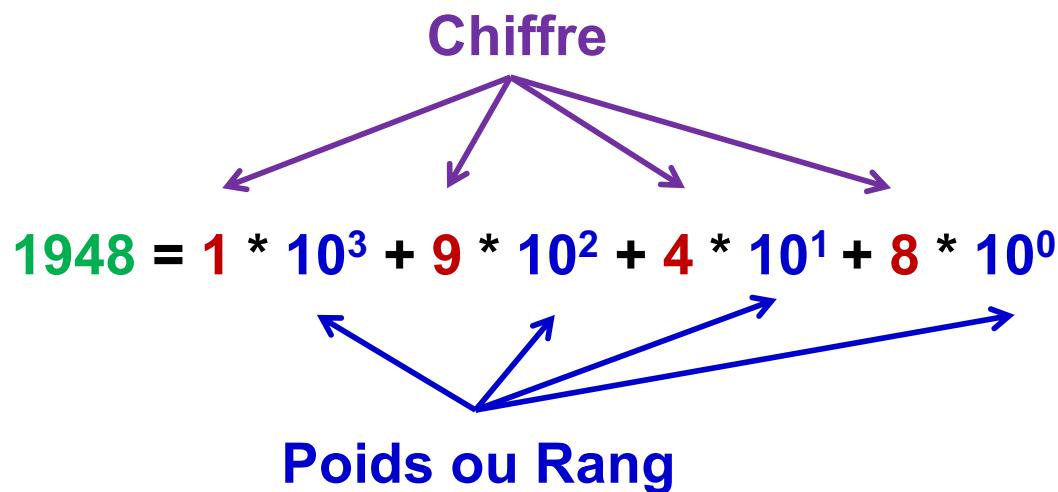
# Notion de « Base »

- Le **système numérique** dans lequel nous travaillons naturellement est le **système décimal**. Nous disons que la base de numération utilisée est **la base 10**.
- L'heure s'exprime en **base 12** ou **24**, les minutes et les secondes en **base 60**.
- Le **système numérique** est un "**système à positions**" :
  - ❖ Un nombre est représenté par une suite ordonnée de chiffres, dans laquelle la position de chaque chiffre est "associée à un poids".
  - ❖ Le poids est une puissance de la **base de numération**.

# Décomposition d'un nombre en puissance de 10

- La valeur du nombre est ainsi la "somme pondérée de ses chiffres".
- Exemple

Le nombre 1948 s'exprime sous la forme :



# Décomposition d'un nombre dans la base 10 : cas général

- Tout nombre entier positif exprimé dans le système décimal peut s'écrire sous la forme :

$$N = c_n \times 10^n + c_{n-1} \times 10^{n-1} + \cdots + c_0 \times 10^0 = \sum_{i=0}^n c_i \times 10^i$$

Avec  $c_i \in [0 - 9]$  et  $0 \leq i \leq n$

Remarque : si  $N = 0$  alors  $n = 0$  et  $c_n = 0$

- Le terme générique de ce développement  $c_i \times 10^i$  est formé du chiffre  $c_i$  et du poids (rang) de ce chiffre  $10^i$

# Système de Numération

## □ Définition

- ❖ La représentation des nombres est liée à un système de numération. Celui-ci est défini par :
  - Une base  $B$  de numération
  - Un alphabet  $A$  de  $B$  symboles :  $A = \{ a_0, a_1, \dots, a_{B-1} \}$

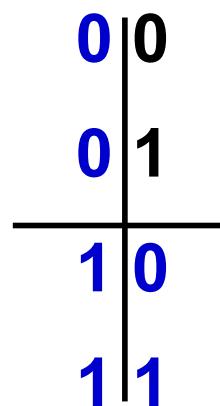
## □ Exemples :

- ❖ Système décimal
  - $B=10, A=\{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$
- ❖ Système Binaire
  - $B=2, A=\{ 0, 1 \}$
- ❖ Système Octal
  - $B=8, A=\{ 0, 1, 2, 3, 4, 5, 6, 7 \}$
- ❖ Système Hexadécimal
  - $B=16, A=\{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F \}$

# La base 2 : construction

0  
1

# La base 2 : construction



# La base 2 : construction

0	0 0
0	0 1
0	1 0
0	1 1
<hr/>	
1	0 0
1	0 1
1	1 0
1	1 1

# La base 2 : construction

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
<hr/>			
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

# Représentation dans un système de numération

- Pour la base de numération B choisie, les B symboles constituant l'alphabet permettent d'exprimer les nombres, ils constituent l'ensemble des « **chiffres** ».
- **ATTENTION ! très important**
  - ❖ Un nombre est une entité qui dispose **d'une représentation dans chaque système** de numération. Sa **valeur** est une **constante** indépendante de tout système de numération.
- Exemples :

$$N1 = (12)_{10}$$

$$= (1100)_2$$

$$= (14)_8$$

$$= (C)_{16}$$



$$N2 = (8)_{10}$$

$$= (1000)_2$$

$$= (10)_8$$

$$= (8)_{16}$$

# Représentation de la partie entière d'un nombre dans une base B

- La partie entière d'un nombre positif exprimé dans un système de numération (B, A) s'écrit sous la forme

$$N = c_n \times B^n + c_{n-1} \times B^{n-1} + \cdots + c_0 \times B^0 = \sum_{i=0}^n c_i \times B^i$$

Avec  $c_i \in A$  et  $0 \leq i \leq n$

Remarque : si  $N = 0$  alors  $n = 0$  et  $c_n = 0$

- Représentation condensée

- ❖ Dans la notation condensée toutes les positions des chiffres doivent être représentées
- ❖  $N = (c_n c_{n-1} \dots c_1 c_0)_B$
- ❖ Quand aucune ambiguïté n'est à craindre :  $N = c_n c_{n-1} \dots c_1 c_0$

# Représentation de la partie fractionnaire d'un nombre dans une base B

- La partie fractionnaire d'un nombre positif exprimé dans un système de numération (B, A) s'écrit sous la forme

$$N = c_{-1} \times B^{-1} + c_{-2} \times B^{-2} + \cdots + c_{-m} \times B^{-m} = \sum_{i=-m}^{-1} c_i \times B^i$$

Avec  $c_i \in A$  et  $-m \leq i \leq -1$

- Représentation condensée

❖  $N = (0, c_{-1} \dots c_{-m})_B$

# Représentation d'un nombre quelconque dans une base B

## □ Représentation dans le système de numération (B, A)

$$N = c_n \times B^n + c_{n-1} \times B^{n-1} + \cdots + c_0 \times B^0 + c_{-1} \times B^{-1} + c_{-2} \times B^{-2} + \cdots + c_{-m} \times B^{-m}$$

$$N = \sum_{i=-m}^n c_i \times B^i$$

Avec  $c_i \in A$  et  $-m \leq i \leq n$

## □ Représentation condensée

$$\diamond N = (c_n c_{n-1} \dots c_1 c_0, c_{-1} \dots c_{-m})_B$$

# Addition en base 10

## □ Exemple 1 :

❖ Addition de deux nombres en base 10 : 945 + 87

Retenue	1	1	1	
		9	4	5
+			8	7
	1	0	3	2

$$\text{d'où } (945)_{10} + (87)_{10} = (1032)_{10}$$

# Addition en base 2

## □ Exemple 2 :

❖ Addition de deux nombres en base 2 :  $1011 + 11$

Retenue	0	1	1	
	1	0	1	1
+			1	1
	1	1	1	0

$$\text{d'où } (1011)_2 + (11)_2 = (1110)_2$$

# Addition en base B

## □ EXERCICES:

❖ Effectuer les additions suivantes:

**21805 + 9376 en base 10**

**11011101 + 101110 en base 2**

**2572 + 310 en base 8**

# Changement de système de numération (Changement de base)

- Nous connaissons la représentation du nombre **N** dans le système de numération **(B<sub>0</sub>,A<sub>0</sub>)**. Nous voulons obtenir la représentation de **N** dans le système de numération **(B<sub>1</sub>, A<sub>1</sub>)**
- **La valeur du nombre ne change pas**, seule sa représentation change en fonction du système de numération utilisé.
  - ❖ Le nombre N=12 exprimé dans la base 10 (décimal), s'exprime par:
    - 1100 dans la base 2 (binaire)
    - 14 dans la base 8 (octal)
    - C dans la base 16 (hexadécimal)

# Pourquoi et comment changer de base ?

## □ Système naturel

- ❖ Les nombres sont exprimés dans la **base décimale**

## □ Dans les ordinateurs

- ❖ Les nombres sont exprimés dans la **base binaire**
- ❖ Les bases **octale** et **hexadécimale** servent à simplifier rapidement la représentation interne (binaire) de l'ordinateur

## □ Maîtriser le passage d'une représentation à une autre

## □ Convertir des parties entières et fractionnaires séparément

- ❖ Les algorithmes utilisés sont différents

# Conversion d'un nombre positif base $B_0 \Rightarrow$ base 10

- Évaluer les chiffres et les puissances de la base  $B_0$  dans base 10
- Effectuer les produits et l'addition finale pour obtenir l'expression du nombre dans le système ( $B_1=10$ ,  $A_1=\{0,1,\dots,9\}$ )

$$N = (c_n c_{n-1} \dots c_1 c_0, c_{-1} \dots c_{-m})_{B_0}$$

$$N = c_n \times B_0^n + c_{n-1} \times B_0^{n-1} + \dots + c_0 \times B_0^0 + c_{-1} \times B_0^{-1} + c_{-2} \times B_0^{-2} + \dots + c_{-m} \times B_0^{-m}$$

## □ Exemples

$$\diamond N = (11010,01)_2 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^{-2}$$

$$= 16 + 8 + 2 + 0,25 = (26,25)_{10}$$

$$\diamond N = (1C)_{16} = 1 * 16 + C = 16 + 12 = (28)_{10}$$

# Conversion d'un nombre positif base $B_0 \Rightarrow$ base 10

## □ EXERCICES

❖ Convertir en décimal les nombres suivants :

$$(1011001)_2$$

$$(\text{FACE})_{16}$$

❖ Effectuer l'addition et la conversion suivantes :

$$(1011)_8 + (11)_8 = (?)_8 = (?)_{10}$$

# Conversion d'un nombre positif base 10 $\Rightarrow$ base $B_1$

- Soit un nombre N exprimé dans le système ( $B_0=10$ ,  $A_0=\{0,1,\dots,9\}$ )
- Exprimons-le dans le système ( $B_1, A_1$ ) :

$$N = (c_n c_{n-1} \dots c_1 c_0, c_{-1} \dots c_{-m})_{B_1}$$

$$N = c_n \times B_1^n + c_{n-1} \times B_1^{n-1} + \dots + c_0 \times B_1^0 + c_{-1} \times B_1^{-1} + c_{-2} \times B_1^{-2} + \dots + c_{-m} \times B_1^{-m}$$

- La partie entière et la partie fractionnaire du nombre sont traitées séparément
  - ❖ Les algorithmes utilisés pour la conversion sont différents

# Conversion d'un nombre entier positif base 10 $\Rightarrow$ base $B_1$ (1/2)

$$N = (c_n c_{n-1} \dots c_1 c_0)_{B_1}$$

$$N = c_n \times B_1^n + c_{n-1} \times B_1^{n-1} + \dots + c_i \times B_1^i + \dots + c_2 \times B_1^2 + c_1 \times B_1^1 + c_0 \times B_1^0$$

Avec  $c_i \in A_1$  et  $0 \leq i \leq n$

- Il peut se mettre sous la forme :

$$N = (c_n \times B_1^{n-1} + c_{n-1} \times B_1^{n-2} + \dots + c_i \times B_1^{i-1} + \dots + c_2 \times B_1^1 + c_1 \times B_1^0) \times B_1 + c_0$$

- Division euclidienne de N par  $B_1$ , dans la base 10

## ❖ Quotient

- $Q_1 = c_n \times B_1^{n-1} + c_{n-1} \times B_1^{n-2} + \dots + c_i \times B_1^{i-1} + \dots + c_2 \times B_1^1 + c_1 \times B_1^0$

## ❖ Reste

- $R_1 = c_0$  = chiffre de **poids faible (rang 0)** de la représentation de N dans la base  $B_1$
- $c_0 = N \bmod B_1$

# Conversion d'un nombre entier positif base 10 $\Rightarrow$ base $B_1$ (2/2)

- On recommence avec  $Q_1$

$$Q_1 = c_n \times B_1^{n-1} + c_{n-1} \times B_1^{n-2} + \cdots + c_i \times B_1^{i-1} + \cdots + c_2 \times B_1^1 + c_1 \times B_1^0$$

- $Q_1$  peut se mettre sous la forme :

$$Q_1 = (c_n \times B_1^{n-2} + c_{n-1} \times B_1^{n-3} + \cdots + c_i \times B_1^{i-2} + \cdots + c_2 \times B_1^0) \times B_1 + c_1$$

- Division euclidienne de  $Q_1$  par  $B_1$ , dans la base 10

- ❖ Quotient

- $Q_2 = c_n \times B_1^{n-2} + c_{n-1} \times B_1^{n-3} + \cdots + c_i \times B_1^{i-2} + \cdots + c_2 \times B_1^0$

- ❖ Reste

- $R_2 = c_1$  = chiffre de **rang 1** de la représentation de N dans la base  $B_1$
    - $c_1 = Q_1 \bmod B_1$

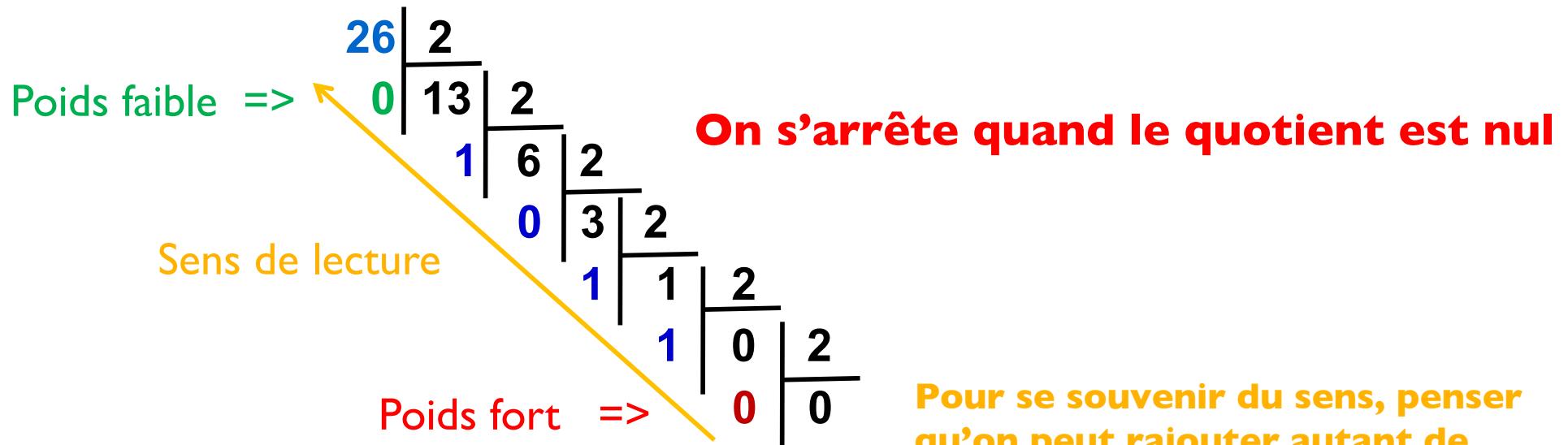
- On recommence avec  $Q_2$

- On s'arrête lorsque  $Q_i = 0$ ,

- ❖  $R_i = c_i$  = chiffre de **poids fort** de la représentation de N dans la base  $B_1$

# Exemple de conversion d'un nombre entier positif base 10 $\Rightarrow$ base B<sub>1</sub>

- Considérons le nombre N = (26)<sub>10</sub> exprimé en base 10
- Nous voulons obtenir la représentation de ce même nombre dans la base 2
  - ❖ Effectuer des divisions par 2



- (26)<sub>10</sub> = (011010)<sub>2</sub>
- Autre exemple : (427)<sub>10</sub> = (1AB)<sub>16</sub>

# Conversion d'un nombre entier positif base 10 $\Rightarrow$ base $B_1$

## □ EXERCICES

- ❖ Convertir  $(6254)_{10}$  en base 16
  
- ❖ Convertir  $(256)_{10}$  et  $(255)_{10}$  en base 2. Que remarque-t-on ?

# Conversion d'un nombre fractionnaire base 10 $\Rightarrow$ base $B_1$

$$N = (0, c_{-1} \dots c_{-m})_{B_1}$$

$$N = c_{-1} \times B_1^{-1} + c_{-2} \times B_1^{-2} + \dots + c_{-i} \times B_1^{-i} + \dots + c_{-m} \times B_1^{-m}$$

Avec  $c_i \in A_1$  et  $-m \leq i \leq -1$

- Le problème important : conservation de la précision du nombre

- ❖ N s'exprime dans le système décimal sous la forme :

$$N = (0, d_{-1} \dots d_{-m})_{10}$$

- ❖ Pour ne pas perdre de précision, il faut exprimer la partie fractionnaire dans le système de numération ( $B_1, A_1$ ) avec un nombre k de chiffres tel que :

$$B_1^{-k} \leq 10^{-m} < B_1^{-(k-1)}$$

# Calcul de la précision par encadrement

□ Combien de chiffres en base 2 pour garder la précision pour  $N = (0,789)_{10}$  ?

❖ En base 10

- Précision :  $10^{-3} = \frac{1}{1000}$

❖ En base 2

- Précision :  $2^{-k}$ , k étant le nombre de chiffres après la virgule

$$\frac{1}{2^k} \leq \frac{1}{1000} < \frac{1}{2^{k-1}} \Rightarrow \frac{1}{2^{10}} \leq \frac{1}{1000} < \frac{1}{2^9} \Rightarrow \frac{1}{1024} \leq \frac{1}{1000} < \frac{1}{512}$$

- Donc :  $k = 10$

□ Cas Général

$$\frac{1}{B_1^k} \leq \frac{1}{B_0^m} < \frac{1}{B_1^{k-1}}$$

# Calcul de la précision par formule

$$\log(2^{-k}) \leq \log(10^{-3})$$

$$-k \times \log(2) \leq -3 \times \log(10)$$

$$k \geq 3 \times \frac{1}{\log(2)} = 9,965784$$

- ❖ k est l'entier immédiatement supérieur à cette valeur
  - Donc : k = 10

## □ Cas Général

$$\log(B_1^{-k}) \leq \log(B_0^{-m})$$

$$-k \times \log(B_1) \leq -m \times \log(B_0)$$

$$k \geq m \times \frac{\log(B_0)}{\log(B_1)}$$

$$\log(2) = 0,301029995 ; \log(8) = 0,903089987 ; \log(16) = 1,204119983$$

# Conversion d'un nombre fractionnaire base 10 $\Rightarrow$ base $B_1$ (1/2)

$$N = (0, c_{-1} \dots c_{-m})_{B_1}$$

$$N = c_{-1} \times B_1^{-1} + c_{-2} \times B_1^{-2} + \dots + c_{-i} \times B_1^{-i} + \dots + c_{-m} \times B_1^{-m}$$

Avec  $c_i \in A_1$  et  $-m \leq i \leq -1$

## □ Multiplication de N par $B_1$ , dans la base 10

$$N \times B_1 = c_{-1} \times B_1^0 + c_{-2} \times B_1^{-1} + \dots + c_{-i} \times B_1^{-(i-1)} + \dots + c_{-m} \times B_1^{-(m-1)}$$

$$N \times B_1 = c_{-1} + (c_{-2} \times B_1^{-1} + \dots + c_{-i} \times B_1^{-(i-1)} + \dots + c_{-m} \times B_1^{-(m-1)})$$

$$N \times B_1 = c_{-1}, c_{-2} \dots c_{-m}$$

### ❖ Partie entière

- $c_{-1}$  = chiffre de **rang -1** de la représentation de N dans la base  $B_1$

### ❖ Partie fractionnaire

- $N_1 = c_{-2} \times B_1^{-1} + \dots + c_{-i} \times B_1^{-(i-1)} + \dots + c_{-m} \times B_1^{-(m-1)}$

# Conversion d'un nombre fractionnaire base 10 $\Rightarrow$ base $B_1$ (2/2)

- On recommence avec la nouvelle partie fractionnaire  $N_1$

$$N_1 \times B_1 = c_{-2} \times B_1^0 + \cdots + c_{-i} \times B_1^{-(i-2)} + \cdots + c_{-m} \times B_1^{-(m-2)}$$

$$N_1 \times B_1 = c_{-2} + (c_{-3} \times B_1^{-1} + \cdots + c_{-i} \times B_1^{-(i-2)} + \cdots + c_{-m} \times B_1^{-(m-2)})$$

$$N \times B_1 = c_{-2}, c_{-3} \dots c_{-m}$$

## ❖ Partie entière

- $c_{-2}$  = chiffre de **rang -2** de la représentation de N dans la base  $B_1$

## ❖ Partie fractionnaire

- $N_2 = c_{-3} \times B_1^{-1} + \cdots + c_{-i} \times B_1^{-(i-2)} + \cdots + c_{-m} \times B_1^{-(m-2)}$

- On recommence avec la nouvelle partie fractionnaire  $N_2$

- On s'arrête lorsque la précision est atteinte (**k chiffres**)

- ❖  $c_{-k}$  = chiffre de **rang -k** de la représentation de N dans la base  $B_1$

# Conversion $N=(0,67)_{10}$ base 10 $\Rightarrow$ base 2 (1/2)

## □ Calculer la précision dans la base 2

- ❖ Précision dans base 10 :  $10^{-2}$
- ❖ Combien faut-il de chiffres dans la base 2 pour garder la même précision ?
  - Méthode par encadrement

$$\frac{1}{2^k} \leq \frac{1}{100} < \frac{1}{2^{k-1}} \Rightarrow \frac{1}{2^7} \leq \frac{1}{100} < \frac{1}{2^6} \Rightarrow \frac{1}{128} \leq \frac{1}{100} < \frac{1}{64}$$

- Donc :  $k = 7$
- Méthode avec la formule

$$\log(2^{-k}) \leq \log(10^{-2})$$

$$k \geq 2 \times \frac{1}{\log(2)} = 6,6438$$

- Donc :  $k = 7$

# Conversion $N=(0,67)_{10}$ base 10 $\Rightarrow$ base 2 (2/2)

$$0,67 \times 2 = 1,34 \Rightarrow \text{chiffre } 1 \quad (1)$$

$$0,34 \times 2 = 0,68 \Rightarrow \text{chiffre } 0 \quad (2)$$

$$0,68 \times 2 = 1,36 \Rightarrow \text{chiffre } 1 \quad (3)$$

$$0,36 \times 2 = 0,72 \Rightarrow \text{chiffre } 0 \quad (4)$$

$$0,72 \times 2 = 1,44 \Rightarrow \text{chiffre } 1 \quad (5)$$

$$0,44 \times 2 = 0,88 \Rightarrow \text{chiffre } 0 \quad (6)$$

$$0,88 \times 2 = 1,76 \Rightarrow \text{chiffre } 1 \quad (7)$$

d'où  $N = (0,67)_{10} \sim (0,1010101)_2$

# Conversion d'un nombre positif base 10 $\Rightarrow$ base $B_1$ Règle de calcul

## Conversion de la partie entière

- ❖ Dans le système de numération  $(B_0, A_0)$ , on procède par divisions de N puis des quotients successifs obtenus par  $B_1$
- ❖ Les quotients et restes sont exprimés dans  $(B_0, A_0)$
- ❖ Les restes successifs représentent les chiffres, exprimés dans  $A_0$ , de la représentation du nombre dans le système de numération  $(B_1, A_1)$ .
- ❖ Il suffit alors d'exprimer ces chiffres dans  $A_1$ : les chiffres sont obtenus des faibles poids vers les forts poids

## Conversion de la partie fractionnaire

- ❖ Dans le système  $(B_0, A_0)$ , on procède par multiplications de N puis des parties fractionnaires successives par la base  $B_1$
- ❖ Les résultats sont exprimés dans le système de numération  $(B_0, A_0)$
- ❖ Les parties entières successives obtenues sont les chiffres de la représentation du nombre dans le système de numération  $(B_1, A_1)$ , et sont exprimées dans  $A_0$
- ❖ Il suffit alors de les exprimer dans  $A_1$ : les chiffres sont obtenus dans le bon ordre, des poids forts vers les poids faibles

# Conversion d'un nombre positif base 10 $\Rightarrow$ base B<sub>1</sub>

## □ EXERCICES

❖ Convertir en décimal les nombres suivants :

$$(1001110,011)_2$$

$$(BAC,C)_{16}$$

❖ En conservant la précision, convertir en décimal : (CD3,DB)<sub>16</sub>

❖ Convertir les décimaux suivants :

$$(0,2)_{10} = (?)_2$$

$$(29,1289)_{10} = (?)_{16}$$

❖ En conservant la même précision, réaliser les conversions

$$(0,10)_{10} = (?)_2 = (?)_{10} \quad \text{Que se passe-t-il?}$$

# Conversions rapides

- Ces conversions que nous appelons « rapides » concernent le passage d'un système de numération ( $B, A$ ) à un système de numération ( $B^p, A_p$ ), ou l'inverse
- Par exemple, si un nombre  $N$  est exprimé en base 2, nous voulons exprimer ce même nombre dans la base 8 ( $=2^3$ ) ou dans la base 16 ( $=2^4$ )

# Conversion rapide du système de numération (B, A) vers le système de numération ( $B^p$ , $A_p$ )

## □ Conversion d'une partie entière

- ❖ Décomposer le nombre exprimé dans le système de numération (B, A)
  - Des faibles poids vers les forts poids (de la droite vers la gauche) en « groupes » de p chiffres
  - Ajouter éventuellement des zéros en tête pour compléter le groupe de plus fort poids (le plus à gauche) à p chiffres
- ❖ Remplacer chaque groupe par sa valeur dans  $A_p$

$$\square N = (10111011)_2$$

- ❖  $(\textcolor{red}{0}10 \ 111 \ 011)_2 \Rightarrow (2 \ 7 \ 3)_8$
- ❖  $(1011 \ 1011)_2 \Rightarrow (\text{B } \text{B})_{16}$

## □ Conversion d'une partie fractionnaire

- ❖ Décomposer le nombre exprimé dans le système de numération (B, A)
  - Des forts poids vers les faibles poids (de la gauche vers la droite) en « groupes » de p chiffres
  - Ajouter éventuellement des zéros pour compléter le groupe de plus faible poids (le plus à droite) à p chiffres
- ❖ Remplacer chaque groupe par sa valeur dans  $A_p$
- ❖  $N = (0,0011)_2$   
 $= (0,001 \ 100)_2 \Rightarrow (0,1 \ 4)_8$   
 $= (0,0011)_2 \Rightarrow (0,3)_{16}$

# Conversion du système de numération ( $B^p$ , $A_p$ ) vers le système de numération ( $B$ , $A$ )

- Exprimer chaque chiffre du nombre exprimé dans le système de numération ( $B^p$ ,  $A_p$ ) sous la forme d'un nombre de  $p$  chiffres dans  $A$
- Supprimer les éventuels zéros inutiles
- Exemples :

$$N = (13,6)_8 = (001\ 011, 110)_2 = (1011,11)_2$$

$$N = (B,C)_{16} = (1011,1100)_2 = (1011,11)_2$$

# Bilan

- Pour les conversions, on utilisera les schémas suivants :
  - ❖ de 16 vers 2 (ou de 2 vers 16) : conversion rapide
  - ❖ de 16 vers 10 : Calcul polynomial
  - ❖ de 10 vers 16 : Divisions/multiplications
  - ❖ de 2 vers 10 :  $2 \Rightarrow 16 \Rightarrow 10$
  - ❖ de 10 vers 2 :  $10 \Rightarrow 16 \Rightarrow 2$
  - ❖ de 8 vers 16 :  $8 \Rightarrow 2 \Rightarrow 16$
  - ❖ de 16 vers 8 :  $16 \Rightarrow 2 \Rightarrow 8$

# Exercices

- Convertir  $(1100011001100)_2$  successivement en bases 8 et 16
  
- Convertir  $(BAC07)_{16}$  en base 2
  
- $(AB,C)_{16} = (???)_8$
  
- Multiplications :
  - $(110111)_2 \times 4 = (?)_2$
  - $(110111,1)_2 \times 8 = (?)_2$
  - $(257)_8 \times 64 = (?)_8$
  - $(341,123)_8 \times 64 = (?)_8$
  - $(AF3)_{16} \times 256 = (?)_{16}$
  - $(10F,234BA)_{16} \times 65536 = (?)_{16}$
  
- Divisions:
  - $(123,65)_{10} / 1000 = (?)_{10}$
  - $(11,1101)_2 / 16 = (?)_2$

# Exercices d'auto-entraînement

□ Compléter le tableau suivant :

Binaire	Décimal	Hexadécimal
101010		
	53	
		F5
11010111		
	111	
		FF
110101		
	155	
		CAFE

# Exercices d'auto-entraînement

- Ecrire en hexadécimal : 1K (1K=1 Kilo=1024); 2K; 4K; 7K; 8K; 20K
- Ecrire en base 2 les nombres décimaux suivants : 0; 2; 4; 7; 257; 102
- Ecrire en base 16 les nombres décimaux suivants : 0; 8; 15; 16; 80; 90. 256; 770
- Ecrire les nombres hexadécimaux suivants en decimal puis en binaire :  
7; 128; FFF; FAB; D; 100; FAC
- Classer dans l'ordre croissant :  
 $(11111001)_2$ ;  $(1101)_{10}$ ;  $(1101)_{16}$ ;  $(1000)_{16}$ ;  $(1000)_2$ ;  $(10000)_{10}$
- Convertir en base 10 :  $(111,101)_2$ ;  $(36B,38)_{16}$ ;
- Convertir en hexadécimal:  $(33,242)_{10}$ ;  $(100001,00111101)_2$

# Exercices d'auto-entraînement

- On considère un dispositif de mesure qui convertit linéairement une grandeur analogique  $V_a$  comprise entre 0 et 150 (Volts par exemple) en une valeur entière codée sur  $n$  bits. On souhaite que le pas maximal de quantification soit égal à 0,1
  - ❖ Quel nombre de bits faut-il pour représenter toutes les valeurs de  $V_a$  ?
  - ❖ Quel code représente la valeur 0? Quel code représente la valeur 150?
  - ❖ Donner le code hexadécimal associé à la valeur 143,2

# Plan

## □ Numération

- ❖ Introduction
- ❖ Système de numération – Cas général
- ❖ Addition en base B
- ❖ Changement de système de numération

## □ Représentation de l'information

- ❖ Introduction
- ❖ Représentation des Entiers naturels
- ❖ Représentation des Entiers relatifs
- ❖ Représentation des Réels (un aperçu rapide)

# Codage de l'information

## □ Introduction

- ❖ Un **codage** consiste à établir une loi de correspondance appelée « **code** » entre un **ensemble d'informations à représenter** et un **ensemble de configurations possibles**
  - Pré-requis : Partie sur la Numération
- ❖ Dans la machine, les nombres sont représentés avec un système binaire
  - $B = 2$ ,  $A = \{0, 1\}$
- ❖ Pour encoder  $N$  informations en binaire, il faut utiliser  $k$  positions binaires avec :  $2^{k-1} < N \leq 2^k$

# Le bit

- Unité élémentaire de codage de l'information
- « **bit** »
  - ❖ Contraction de *binary digit* (chiffre binaire)
    - prend ses valeurs dans {0, 1}
- Codage de l'information
  - ❖ Séquences de *bits* = mots mémoire
  - ❖ Tous les nombres d'un même *type* représentés avec le même nombre de *bits*
  - ❖ Pour représenter les nombres :
    - mots de taille fixe (32 bits, 64 bits)

# L'octet

- Groupe de 8 *bits* codant  $256 = 2^8$  valeurs différentes comprises dans l'intervalle [0, 255]
- Exemples en binaire :

$(0)_{10} = (0000\ 0000)_2$	$(128)_{10} = (1000\ 0000)_2$
$(15)_{10} = (0000\ 1111)_2$	$(253)_{10} = (1111\ 1101)_2$
$(127)_{10} = (0111\ 1111)_2$	$(255)_{10} = (1111\ 1111)_2$

- Exemples en hexadécimal (2 quartets, groupes de 4 *bits*) :

$(0)_{10} = (00)_{16}$	$(128)_{10} = (80)_{16}$
$(15)_{10} = (0F)_{16}$	$(253)_{10} = (FD)_{16}$
$(127)_{10} = (7F)_{16}$	$(255)_{10} = (FF)_{16}$

# **TD : exercice n° 1.1**

# Entiers naturels non signés

- Binaire “pur”
  - ❖ k bits permettent de coder  $2^k$  valeurs différentes comprises dans l'intervalle  $[0, 2^k-1]$
- Exemple avec 2 octets
  - ❖ Mais aussi 4 quartets ou 16 bits, permettent de représenter les nombres de l'intervalle  $[0, 2^{16}-1] = [0, 65535]$
  - ❖ En binaire pur :
    - de  $(0000\ 0000\ 0000\ 0000)_2$  à  $(1111\ 1111\ 1111\ 1111)_2$
  - ❖ En hexadécimal :
    - de  $(0000)_{16}$  à  $(FFFF)_{16}$

# TD : exercice n° 2

# Entiers relatifs en VA+S

## □ Signe et Valeur Absolue (notation en VA+S (k bits)VAS)

- ❖ Par convention : le bit de plus fort poids (le plus à gauche) représente le signe

- 0 = signe +
- 1 = signe -

signe	valeur absolue sur k -1 bits
-------	------------------------------

- ❖ Les autres bits représentent la valeur absolue du nombre

## □ k bits : ( $2^k$ ) nombres de l'intervalle $[-(2^{k-1}-1), +(2^{k-1}-1)]$

- ! ❖ Il y a deux représentations possibles pour le zéro

- Par exemple, sur 8 bits :  $(0000\ 0000)_{VAS}$  et  $(1000\ 0000)_{VAS}$

## □ Intervalles de représentation :

- ❖ Sur 4 bits :  $2^4$  nombres de l'intervalle  $[-7, +7]$
- ❖ Sur 8 bits :  $2^8$  nombres de l'intervalle  $[-127, + 127]$
- ❖ Sur 16 bits :  $2^{16}$  nombres de l'intervalle

$$[ -(2^{15} -1), +(2^{15} -1) ] = [ - 32\ 767, + 32\ 767 ]$$

# Entiers relatifs en VA+S

## □ Exemples sur 8 bits

- ❖  $(0111\ 1111)_{VA+S} = (+127)_{10}$
- ❖  $(1111\ 1111)_{VA+S} = (-127)_{10}$
- ❖  $(0000\ 0001)_{VA+S} = (+1)_{10}$
- ❖  $(1000\ 0001)_{VA+S} = (-1)_{10}$
- ❖  $(0000\ 0000)_{VA+S} = (+0)_{10} = (0)_{10}$
- ❖  $(1000\ 0000)_{VA+S} = (-0)_{10} = (0)_{10}$

## □ Inconvénient

- ❖ Impossibilité d'utiliser l'addition bit à bit

$(1011)_{VA+S}$  (code de  $(-3)_{10}$ ),

+  $(0010)_{VA+S}$  (code de  $(+2)_{10}$ ),

=  $(1101)_{VA+S}$  (code de  $(-5)_{10}$ ) !

## □ Solution : Utilisation du Complément à 2

# TD : exercice n° 3

# Entiers relatifs en Complément à 2

- Sur  $k$  bits :  $2^k$  nombres de l'intervalle  $[-2^{k-1}, +(2^{k-1}-1)]$
- Intervalles de représentation :
  - ❖ Sur 4 bits :  $2^4$  nombres de l'intervalle  $[-2^3, +(2^3-1)] = [-8, +7]$
  - ❖ Sur 8 bits :  $2^8$  nombres de l'intervalle  $[-128, + 127]$
  - ❖ Sur 16 bits :  $2^{16}$  nombres de l'intervalle  
 $[-(2^{15}), +(2^{15}-1)] = [-32\ 768, +32\ 767]$
- Exemples sur 8 bits :
  - ❖  $(0000\ 0000)_{C2}$  code de  $(0)_{10}$  une seule représentation pour 0
  - ❖  $(1000\ 0000)_{C2}$  code de  $(-128)_{10}$  plus petite valeur représentable
  - ❖  $(0111\ 1111)_{C2}$  code de  $(+127)_{10}$  plus grande valeur représentable
  - ❖  $(0000\ 0001)_{C2}$  code de  $(+1)_{10}$

# Entiers relatifs en C2

## (complément à 2)

- Complément à 2 sur k bits (notation en C2 :  $(k \text{ bits})_{C2}$ )
- Par convention
  - ❖ Si  $0 \leq N < 2^{k-1}$  (bit de fort poids = 0)
    - N est représenté comme un entier sans signe sur  $(k - 1)$  bits
    - et le bit de fort poids est égal à 0
    - $C2(N) = (N)_2$
  - ❖ Si  $N < 0$  et  $|N| \leq 2^{k-1}$  (bit de fort poids = 1)
    - N est représenté par  $C2(|N|)$  (cad par  $(2^k - |N|)$ )
    - le nombre correspondant à ce code appartient à  $[-(2^{k-1}), -1]$
    - et le bit de fort poids est égal à 1
- Le nombre 0 est codé par k bits à 0

# Entiers relatifs en C2

## Cas des nombres négatifs sur k bits

□  $C2(N) = 2^k - |N| = (2^k - 1) + 1 - |N| = (2^k - 1) - |N| + 1$

- ❖ Prendre d'abord le **complément à 1** de  $|N|$  :
  - si le bit  $x = 1$  alors son complément à 1 est  $(1 - x) = 0$
  - si le bit  $x = 0$  alors son complément à 1 est  $(1 - x) = 1$
- ❖ Puis **ajouter 1** et garder les  $k$  chiffres de plus faible poids

□ Exemple :

- ❖ Coder  $(-70)_{10}$  en C2

$$|-70| = (0100\ 0110)_2 \quad C1(0100\ 0110)_2 = (1011\ 1001)_{C1}$$

$$(1011\ 1001)_{C1} + 1 = (1011\ 1010)_{C2} \text{ on sait que c'est négatif car poids fort passe à 1}$$
$$= C2(0100\ 0110)_2 = \text{codage de } (-70)_{10}$$

- Le codage en complément à 2 de  $(-70)_{10}$  est  $(1011\ 1010)_{C2}$

- ❖ Coder  $(+70)_{10}$  en C2

- Nombre positif donc codé comme en VA + signe :  $(0100\ 0110)_2$

# Entiers relatifs en C2

## Cas des nombres négatifs sur k bits

□ Exemple inverse : quelle est la valeur de  $(1111\ 0110)_{C2}$  ?

❖  $(1111\ 0110)_{C2}$  est le codage d'une valeur négative (bit fort poids=1)

- Pour retrouver sa valeur absolue, on prend son C1

$$C1(1111\ 0110)_2 = (0000\ 1001)_{C1}$$

- On ajoute 1

$$\begin{aligned} C2(1111\ 0110)_2 &= C1(1111\ 0110)_2 + 1 = (0000\ 1001)_{C1} + 1 \\ &= (0000\ 1010)_{C2} \quad \text{positif car bit fort poids=0} \\ &= \text{codage de } (+10)_{10} \end{aligned}$$

d'où  $(1111\ 0110)_{C2}$  est le codage de  $(-10)_{10}$

□ Quelle est la valeur de  $(0111\ 0110)_{C2}$  ?

❖  $(0111\ 0110)_{C2}$  est le codage d'une valeur positive (bit fort poids=0)

- On connaît directement sa valeur absolue  $(0111\ 0110)_2$
- C'est donc le codage de  $7 \times 16 + 6 = (+118)_{10}$

# Entiers relatifs en C2

- Quelle est la valeur de  $N= (00000101)_{C2}$  ?

# Entiers relatifs en C2

- Quelle est la valeur de  $N = (00000101)_{C2}$  ?

$(00000101)_{C2}$  est positif car fort poids=0

D'où :

$$(00000101)_{C2} = (00000101)_2 = 2^2 + 2^0 = \text{codage de } (+5)_{10}$$

# Entiers relatifs en C2

- Quelle est la valeur de  $N = (11111101)_{C2}$  ?

# Entiers relatifs en C2

- Quelle est la valeur de  $N = (11111101)_{C2}$  ?

$(11111101)_{C2}$  est négatif car fort poids=1

D'où :

passer en complément à 1:  $C_1(11111101)_2 = (00000010)_{C1}$

et rajouter 1 pour obtenir la valeur de  $|N|$

$(00000010)_{C1} + 1 = (0000\ 0011)_{C2} = \text{codage de } (+3)_{10}$

positif car fort poids = 0

on conclut donc que  $(11111101)_{C2} = \text{codage de } (-3)_{10}$

# Entiers relatifs en C16

## □ Idem qu'en binaire, en “complémentant” à 16

### ❖ Notation sur k bits :

- C16(N) opération de codage en complément à 16
- $(xxx\dots xxx)_{C16}$  codage de N en complément à 16
- CF(xxx...xxx) opération de codage en complément à F
- $(xxx\dots xxx)_{CF}$  codage en complément à F

### ❖ Exemple sur 8 bits (un octet) en hexadécimal :

- $(+70)_{10} = (46)_{16} = (0100\ 0110)_2$  positif car fort poids = 0
- $CF(46)_{16} = (B9)_{CF}$   $(B = F - 4, 9 = F - 6)$
- $(B9)_{CF} + 1 = (BA)_{C16} = (1011\ 1010)_{C2}$   
 $= C16(0100\ 0110)_2 = (1011\ 1010)_{C16}$
- $C16(46)_{16} = (BA)_{C16} = (1011\ 1010)_{C2}$
- $(BA)_{C16}$  est le codage en complément à 16 de  $(-70)_{10}$

# Opérations en C2

## □ Addition et soustraction :

### ❖ N1 + N2 se fait par addition bit à bit

- Exemple sur 4 bits :  $(-3)_{10} = (1101)_C2$  et  $(+2)_{10} = (0010)_C2$

$$\begin{aligned} (-3)_{10} + (+2)_{10} &= \\ (1101)_C2 &+ (0010)_C2 \\ &= (1111)_C2 \quad \text{codage en C2 de } (-1)_{10} \end{aligned}$$

$$(-3) + 2 = (-1) \dots$$

### ❖ N1 - N2 se fait par N1 + C<sub>2</sub>(N2)

- Exemple sur 4 bits:  $(-3)_{10} - (+2)_{10} = (1101)_C2 + C_2(0010)_2 =$

$$\begin{aligned} (1101)_C2 &+ (1110)_C2 \\ &= (1011)_C2 \quad = \text{codage en C2 de } (-5)_{10} \end{aligned}$$

$$(-3) + (-2) = (-5) \dots$$

# Débordement de capacité en C2

- **Le bit de plus fort poids permet de savoir que :**
  - ❖ N est négatif si ce bit est égal à 1
  - ❖ N est positif si ce bit est égal à 0
- **Si l'addition de deux nombres de même signe  
(même bit de plus fort poids) donne un résultat de  
signe contraire alors :**
  - ❖ Débordement de capacité
  - ❖ Le nombre résultant est au-delà des capacités de représentation



# **TD : exercices n° 4 et 5**