Unité : Systèmes logiques 1 (SysLog1)

Représentation des nombres et d'informations non numérique

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

E. Messerli & R. Mosqueron

Septembre 2021

Polycopié : Electronique numérique

- Introduction pages 1 à 4
- Systèmes de numération et codes pages 5 à 20
- Arithmétique binaire pages 21 à 34
  - Addition binaire
  - Nombres signés C1 et C2
  - Addition et soustraction en C2
  - Addition en BCD

Représentation des nombres naturels

Le système de numération romaine a été heureusement remplacé par un système de numération de position dans une base choisie

Couramment nous utilisons la base 10

Représentation des nombres naturels

Exemples :

IV + *XII* = *XVI* ➔ pas de règles répétitives

Autres nombres : *MMD* = *2500* & *MCMLIII* = 1953

Symboles : …. M = 1000, D = 500, C = 100, L = 50, X = 10, V = 5, I = 1

En base 10 :

$$1953 = 1 \times 10^3 + 9 \times 10^2 + 5 \times 10^1 + 3 \times 10^0$$
$$1953 = 1 \times 1000 + 9 \times 100 + 5 \times 10 + 3 \times 1$$
$$1953 = 1000 + 900 + 50 + 3$$

Systèmes de numération "Historique" …

- Le premier système décimal est né en Chine vers l'an -2000
- La numération de position avec un zéro (*çunya* qui était un point à l'origine) est apparut dans un traité de cosmologie qui s'intitule "*les parties de l'univers*" écrit en sanscrit en 458
  - *Traduit en arabe, çunya devient sifr qui, traduit en italien, donna zéfiro. Et de zéfiro à zéro* ☺

… systèmes de numération "Historique"

- En 825:
  Les algorithmes de calcul et l'équation ont étés dévoilés à Bagdad par "Al-Khawarizmi" dans son livre "*livre de l'addition et de la soustraction d'après le calcul (système de numération) des Indiens*".
- Le mot "Algorithme" vient du nom de ce mathématicien
- L'écriture des nombres de droites (poids faible) à gauche (poids fort) nous vient des arabes
  - On parle fréquemment de chiffres arabes

Système de numération de position …

- Système de numération :
  - liste ordonnée de symboles appelés chiffres
  - règles définissant leur utilisation pour créer des nombres
  - règles définissant un jeu d'opérations telles que l'addition, la multiplication, etc., appelé arithmétique.

- Le nombre de symboles différents est appelé base.

… système numération de position

- Le nombre de symboles différents étant très inférieur aux valeurs quantitatives que l'on désire représenter

  => on utilise la juxtaposition de chiffres pour créer des nombres

- Chaque chiffre dans un nombre se voit attribuer un facteur de pondération, ou "poids", qui dépend de sa position dans le nombre.
- Nombre N_b dans une base b:
  - $N_b = a_n a_{n-1} ... a_1 a_0 , a_{-1} ... a_{-m}$ (numération de position)
- Exemples :
  - $N_{10} = 273.15$ $N_2 = 10110.01$

Base 10 et base 2 !

- Si, dans la vie courante, nous utilisons la base 10, les ordinateurs utilisent la base 2, car
  - L'inconvénient de la base 10 est la difficulté de stocker et de transmettre les 10 niveaux avec un signal
  - L'avantage de la base 2 est la facilité de stocker et de transmettre les 2 niveaux avec un signal.
    Elle permet l'utilisation d'éléments électroniques bistables et une transmission fiable, même dans des environnements bruités et imprécis

Historique des ordinateurs : http://histoire.info.online.fr/

Système de numération "Décimal"

- Basé sur dix symboles, soit les chiffres de "0 .. 9"
- Ces chiffres sont classés de droite à gauche "unités, dizaines, centaine, milliers, …."
- Chaque nombre peut être représenté par un vecteur:
  N(k-1), .. , N(2), N(1), N(0)
- Et:

  $Nbr = N(k-1).10^{k-1} + \ldots + N(2).10^{2} + N(1).10^{1} + N(0).10^{0}$

  $Nbr = \sum_{i=0}^{k-1} N_i \,.\, 10^i$ avec $N_i \in \{0, 1, 2, \ldots 8, 9\}$

  $Nbr \in [0, 10^k - 1]$

- Exemple:
  espace de dimension k = 3 (3 chiffres), nombres de 000 à 999
  $483 = 4 \,.\, 10^2 + 8 \,.\, 10^1 + 3 \,.\, 10^0$

Système de numération "Binaire"

- Basé sur deux symboles, soit les chiffres 0 et 1 !
- Ces chiffres sont classés de droite à gauche "unités, multiple de 2, multiple de 4, multiple de 8, …."
- Chaque nombre peut être représenté par un vecteur:
  N(k-1), .. N(2), N(1), N(0)
- Et:

  $Nbr = N(k-1).2^{k-1} + \ldots + N(2).2^2 + N(1).2^1 + N(0).2^0$

  $Nbr = \sum_{i=0}^{k-1} N_i \,.\, 2^i$ avec $N_i \in \{0, 1\}$

  $Nbr \in [0, 2^k - 1]$

- Exemple:
  espace de dimension k = 3 (3 chiffres), nombres de 000 à 111
  $101 = 1 \,.\, 2^2 + 0 \,.\, 2^1 + 1 \,.\, 2^0$

Conversion dans une base B

- N_b converti en base B:
  - $N_B = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_0 b^0 + a_{-1} b^{-1} + a_{-m} b^{-m}$
- p. ex.:
  - $N_2 = 1101$; b=2 dans base B=10

  d'où

  $$N_{10} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13$$

  Les opérations doivent **être faites** dans la base B

Conversion Binaire à Décimal : $N_2 \rightarrow N_{10}$

- Utilisation de la numération de position :
  - $N_2 = a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \ldots + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + \ldots + a_{-m} \cdot 2^{-m}$
- p. ex.:

  $N_2 = 1001{,}1$ en utilisant la représentation

  $N_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1}$

en calculant en base 10 nous obtenons 9,5, d'où $N_{10} = 9{,}5$

Calcul possible pour nous car il est fait en base 10

Conversion Décimal à Binaire : $N_{10} \rightarrow N_2$ ...

- Utilisation de la numération de position :

$$N_{10} = a_n \; a_{n-1} \ldots \; a_0 \; a_{-1} \; \ldots$$
$$N_{10} = a_n \cdot 10^n + \ldots + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + \ldots$$

pour obtenir la valeur en base 2 :

$$N_2 = a_n \cdot 1010^n + \ldots + a_0 \cdot 1010^0 + a_{-1} \cdot 1010^{-1} + \ldots$$

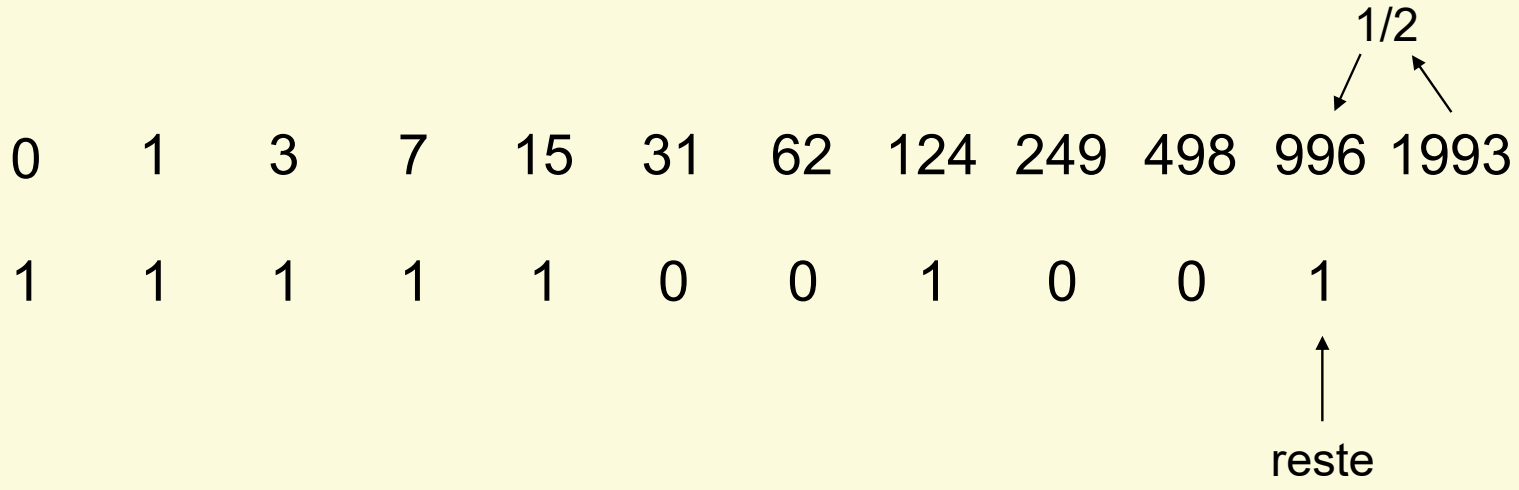
Calcul impossible pour nous, car il doit être fait dans la base 2 !

… Conversion Décimal à Binaire : $N_{10} \rightarrow N_2$ …

- Partie entière E (à m+1 chiffres) d'un nombre, sous forme polynomiale, en base b :
  - $E_b = a_m \cdot b^m + a_{m-1} \cdot b^{m-1} + \ldots + a_1 \cdot b^1 + a_0 \cdot b^0$
  - $E_b = (a_m \cdot b^{m-1} + a_{m-1} \cdot b^{m-2} + \ldots + a_1) \cdot b^1 + a_0$
- Le reste de la division E_b / b est a_0
- Les restes des divisions successives par la base b souhaitée donnent les chiffres du nombre en base b, à partir de la virgule

… Conversion Décimal à Binaire : $N_{10} \rightarrow N_2$ …

- Les calculs seront faits en décimal
- Exemple : $1993.8_{10} = ?_2$
  - Partie entière $1993 = 111\ 1100\ 1001_2$

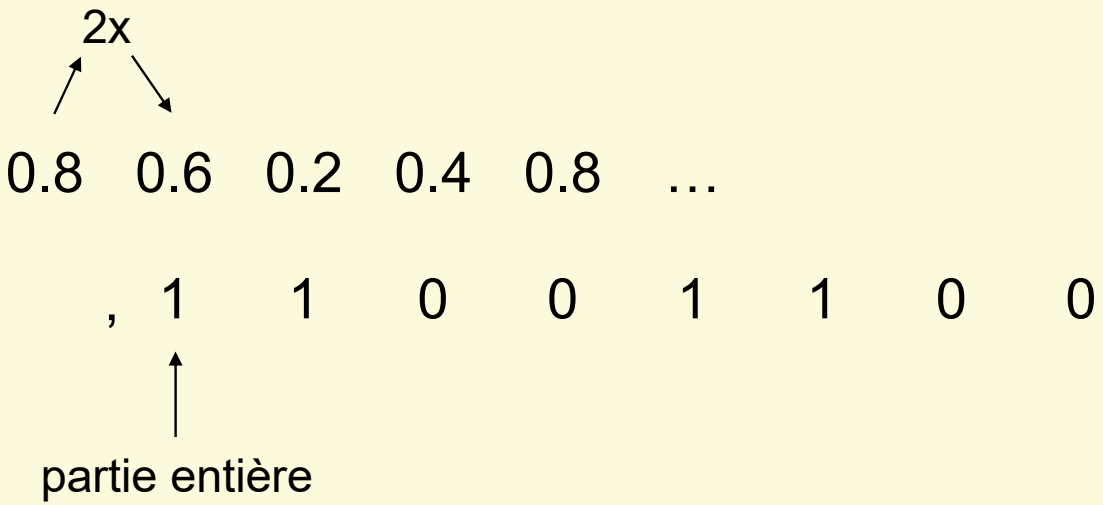


… Conversion Décimal à Binaire : $N_{10} \rightarrow N_2$ …

- Partie fractionnaire F (à k chiffres) d'un nombre, sous forme polynomiale, en base b :
  - $F_b = a_{-1} \cdot b^{-1} + a_{-2} \cdot b^{-2} + \ldots + a_{-k} \cdot b^{-k}$
  - $F_b = (a_{-1} \cdot b^{0} + a_{-2} \cdot b^{-1} + \ldots + a_{-k} \cdot b^{-k+1}) \cdot b^{-1}$
- La partie entière de la multiplication $F_b \cdot b$ est a_{-1}
- Les parties entières des multiplications successives par la base b souhaitée donnent les chiffres du nombre en base b, à partir de la virgule

… Conversion Décimal à Binaire : $N_{10} \rightarrow N_2$ …

- Partie fractionnaire 0.8 $= 0{,}1100\ \overline{1100}_2$



Système de numération "Hexadécimal"

- Basé sur seize symboles (10 chiffres + 6 lettres), soit :

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- Il s'agit d'une base multiple du binaire : **$16 = 2^4$**

- Très utilisée pour représenter les nombres binaires de manière compacte :

  1 chiffre hexadécimal <=> 4 chiffres binaires

Passage du binaire en hexadécimal

- Il suffit de grouper les chiffres binaires 4 par 4
- Exemple :
  - $N_2 = \underbrace{1001}\ \underbrace{0110}\ \underbrace{1100}\ \underbrace{0100}$

    $N_{16} = 9\quad 6\quad C\quad 4 = 96C4$

fréquemment noté : 0x96C4

Passage de l'hexadécimal au binaire

- Il suffit de scinder chaque chiffre hexadécimal en 4 bits (chiffre binaire)
- Exemple :

N_{16} = 5 6 E 4 = 0x56E4

N_2 = 0101 0110 1110 0100

souvent les '0' non significatif ne sont pas écrit :

N_2 = 101 0110 1110 0100

Systèmes de numération : Notations

- Convention : par défaut base 10, sinon base explicitée

- Exemples de notation :
  - En base 2 : 1001_2, ou 0b1001, ou B"1001"
  - En base 16 : 0xA2E, *ou* $A2E_{16}$*, ou A2Ehex, ou h'A2E*
- Vocabulaire : un chiffre binaire est appelé *bit* :
  - contraction de "*binary digit*", signifie aussi "*petit morceau*"

Exercice série I

1. Déterminer la relation qui permet de connaître la valeur maximale Val_{max} , en décimal, d'une représentation en binaire de nombre entier non signé sur n bits

Solution: Val_{max} =

Exemple: n = 4 bits ➔ Val_{max} =

Exercice série I

2. Passage de binaire à décimal:

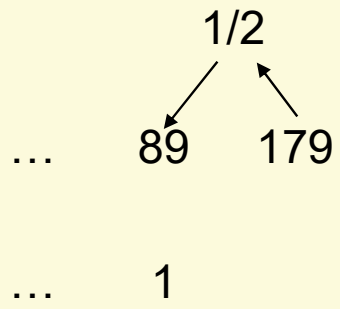
```
01101011 =
         =
         =
```

3. Passage d'hexadécimal à décimal:

```
    A8CE =
         =
         =
```

Exercices série I

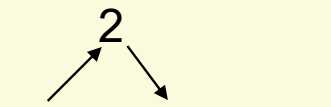
4. N_{10} = 179 = en binaire ?



N_2 =

Exercices série I

5. N_{10} = 0,45 = en binaire ?



0,45 0,9 …

0 , 0 …

N_2 =

Exercices série I

6. Passage de binaire à hexadécimal:

N_2 = 0110 1011 0101 =

N_{16} =

7. Passage d'hexadécimal à binaire:

0xA8CE = A 8 C E

Exercices série I

8. Passage de binaire à hexadécimal:

N_2 = 10010110,100101 =

N_{16} =

9. Passage d'hexadécimal à binaire:

B75.17 = B 7 5 , 1 7

Code BCD, Binary Coded Decimal

- Représentation du décimal directement en binaire en utilisant uniquement les codes de 0 à 9 des nombres binaires codés sur 4 bits.
- Exemple :
  - N_{10} = 679 en BCD : N_{BCD} =0110 0111 1001
- Le BCD n'utilise qu'une partie des codes possibles :
  - 12 bits en BCD => max 999
  - 12 bits en binaire => max 2^{12}-1 = 4095 soit 4 fois plus !!

Conversion d'un nombre BCD

- ATTENTION :

  Le nombre BCD est toujours en décimal !

- Il faut donc utiliser la conversion décimal à binaire pour obtenir sa valeur en binaire.

Exercices série I

10. Passage de BCD à décimal

N_{BCD} = 0110 1001 0101

N_{10} =

11. Passage de BCD à binaire

a) N_{BCD} = 0111

N_2 =

b) N_{BCD} = 0100 0101 1001

N_2 =

N_{BCD} en décimal : N_{10} = 459
convertit en binaire N_2 = 1 1100 1011

Représentation des nombres entiers

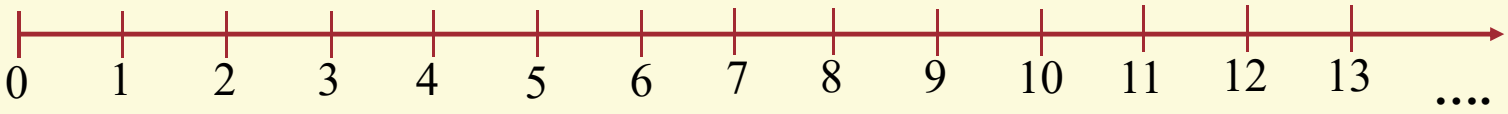
- Les instructions des ordinateurs traitent des nombres de taille fixe, soit : 4, 8, 16, 32 ou 64 bits
- Avec un nombre composé de n bits, on ne peut représenter que 2^n valeurs entières différentes
- On souhaite disposer de valeurs positives et de valeurs négatives
- On souhaite pouvoir réaliser les 4 opérations arithmétiques (add, sub, mul, div) de la façon la plus simple possible

Représentation des nombres entiers

- Nombre entier non signé
  - Représentation binaire standard, voir pages précédentes
- Nombre entier signé
  - Choisir une représentation du signe :
    - A la main on utilise le signe "-" qui précède le nombre positif
    - dans le tableur Excel, la couleur Rouge est parfois utilisé
    - …..
  - En électronique numérique on a dédié un bit: le "bit de signe", mais il existe plusieurs représentations :
    - signe & valeur absolue, complément à 1, complément à 2, biaisée, …
    - la représentation la plus utilisée est le complément à 2

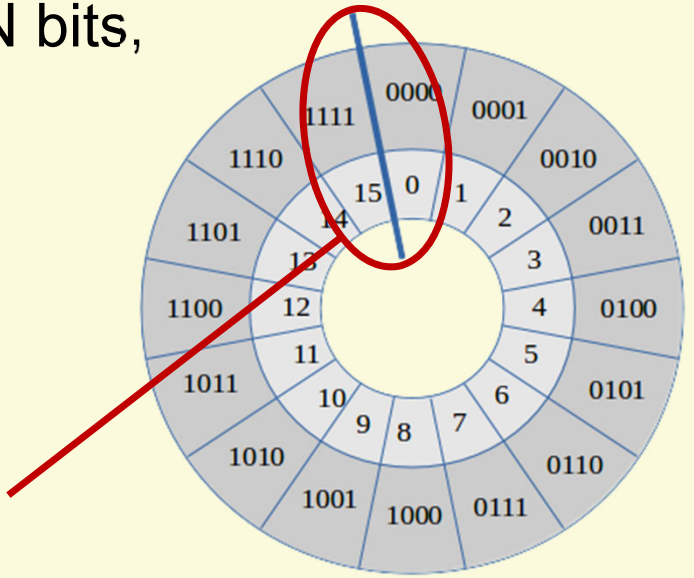
Nombres entiers non-signés

- Les nombres peuvent être représentés sur une droite:



- En informatique: nombres représentés sur N bits, d'où une plage **limitée**!

  La représentation sur un cercle implique
  un risque de débordement !
  en anglais, nommé : **carry**



Débordement !
(carry)

Représentation des nombres signés

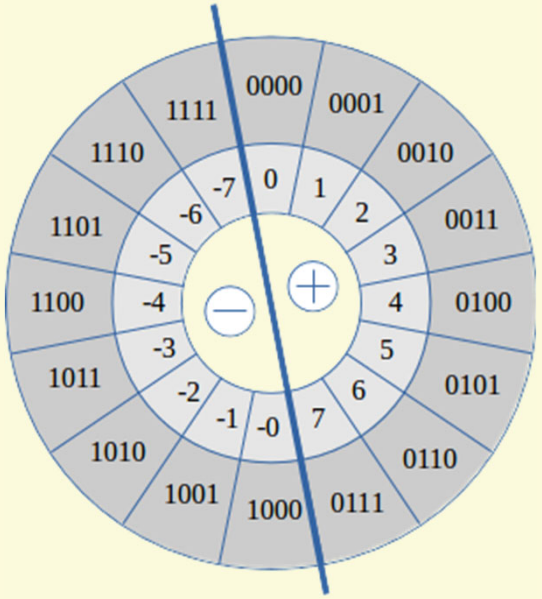
- Plusieurs représentations des nombres entiers signés en binaire :
  - Signe-amplitude
  - Complément à 1
  - Complément à 2^n
  - Excédent de $2^{n-1} - 1$

Nombres négatifs: signe-amplitude

Signe-amplitude :

- le bit de gauche est utilisé pour le signe (0 pour positif, 1 pour négatif), les autres pour la valeur absolue
  - utilisée pour la mantisse des nombres en virgule flottante (notation scientifique)

- Inconvénients :
  - Double représentation du zéro
  - Algorithmique complexe, même pour l'addition



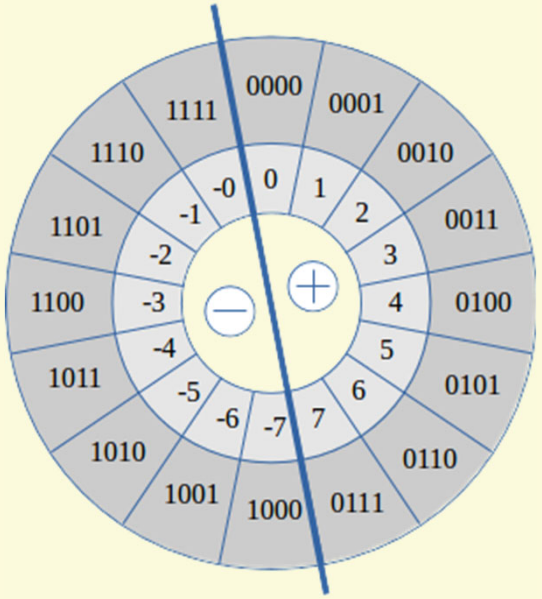
Nombres négatifs: complément à 1

Complément à 1 : en fait, c'est le complément à $2^n - 1$

- Avantage : facile à calculer (inverser tous les bits)
- Formule pour calculer le complément à 1 :
  $C1(A) = 2^n - 1 - A = not(A)$ (inversion bit à bit)

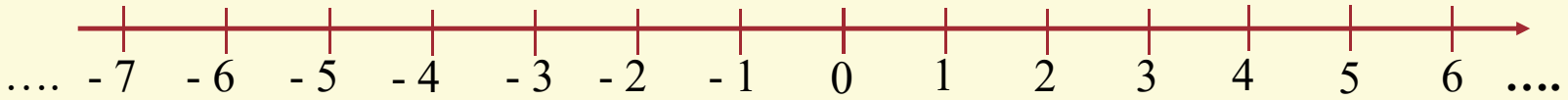
- Inconvénients :
  - Double représentation du zéro

- Représentation pas utilisée



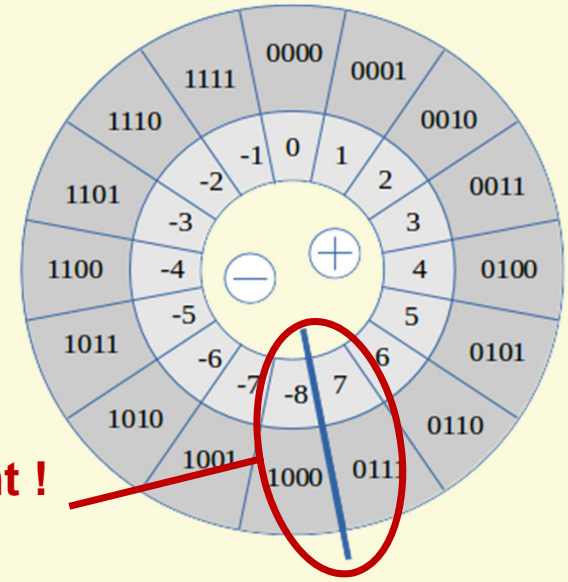
Nombres négatifs : complément à 2

- Les nombres réels représentés sur une droite:



- En informatique: nombres sur N bits!
  plage limitée => cercle => débordement !
- Complément à 2 :
  - Complément à 2^n , soit: $C2(A) = 2^n - A$
  - Représentation naturel des nombres négatifs
  - Avantage : même circuit d'addition pour les nombres non-signés et signés

- Représentation la plus utilisée

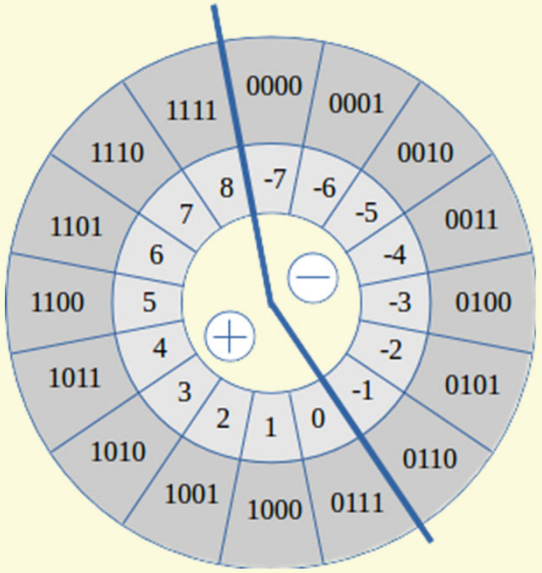


Débordement !
(overflow)

Nombres négatifs : excédent

Excédent : c'est le complément à 2^{n-1} -1

- les cercles signés en C2 est tourné de demi-tour afin d'avoir les nombres négatif depuis "0...0"
- Avantage : nombre positif > nombre négatif
- Formule pour calculer nombre en excédent: Excédent(A) = A + (2^{n-1} -1)
- utilisé pour l'exposant des nombres en virgules flottante



- Inconvénient :
  - impossible de faire des calculs directement

Nombres négatifs : complément à 2^n...

- Complément à 2^n => représentation naturelle

  3 - 4 = -1, soit 0011 - 0100 = 1111 !

- Un compteur-décompteur n bits en binaire pur
  - compte en boucle : 0, 1, … 2^n – 1, 0, 1, …
  - sur 4 bits:
    - si compte + 1 depuis 0 ("0000"):
      "0000" → "0001" → "0010" → "0011" → …
      0 +1 +2 +3 …
    - si décompte - 1 depuis 0 ("0000"):
      "0000" → "1111" → "1110" → "1101" → …
      0 -1 -2 -3 …

…nombres négatifs : complément à 2^n…

- Sur un compteur n bits, la valeur –1 est naturellement représentée par 2n – 1, qui est la valeur obtenue en décomptant 1 fois depuis 0
- Avec cette représentation, en additionnant –1 et +1 on obtient 2^n :
  - -1 est le complément à 2^n de +1
  - -2 est le complément à 2^n de +2, etc
- D'où : «représentation en complément à 2^n»
- Autre terminologie souvent utilisée :

  ce nombre est *(écrit)* «en *(notation)* complément à 2»

...nombres négatifs : complément à 2^n...

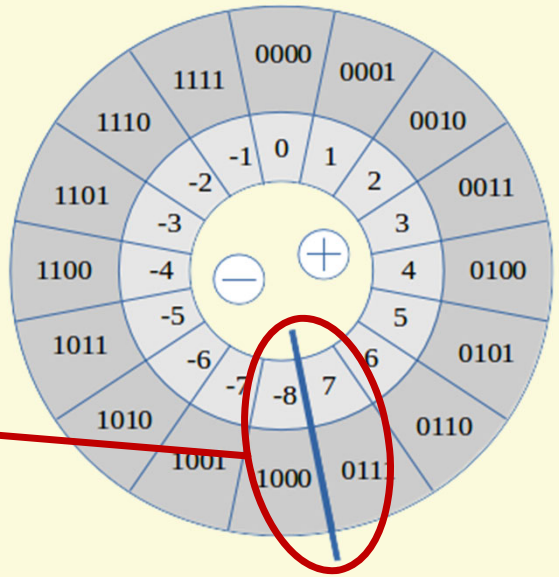
- La représentation en complément à 2^n d'un nombre négatif –A, s'obtient en calculant le complément à 2^n de +A, soit :

$$C_2(A) = 2^n - A$$

- Calcul du complément à 2^n d'un nombre à la main:
  - Il s'obtient en inversant chacun des bits n du nombre, puis en ajoutant 1
  - $C_2(A)$ = not A + 1 (démonstration à la suite)

…nombres négatifs : complément à 2^n

- Représentation sur un cercle des nombres signés en complément à 2
  - Débordement sur 4 bits :
    il a lieu entre +7 et –8 !
  - Celui-ci est nommé: Overflow



Débordement :
Overflow

Soustraction avec complément à 2

Exemple : 25 – 18 = +7, en utilisant le C2, on calcul : 25 + $C_2(18)$

exposant de 2:	5	4	3	2	1	0	
+ 18 :	0	1	0	0	1	0	
$C_1(18)$	1	0	1	1	0	1	☺ simple inversion des bits
+ 1 :						1	
$C_2(18)$:	1	0	1	1	1	0	☺ 1ère étape $C_2(18)$

retenues :	1	1	1				
+ 25 :	0	1	1	0	0	1	☺
+ $C_2(18)$:	1	0	1	1	1	0	☺ 2ème étape 25 + $C_2(18)$
+ 7 :	0	0	0	1	1	1	☺ Résultat signé !!!

Relation entre C_1 et C_2

- Complément à 1 :
  - $C_1(A) = 2^n - 1 - A = \text{not}(A)$

- Complément à 2 :
  - $C_2(A) = 2^n - A = 2^n - 1 + 1 - A = C_1(A) + 1$

    d'où:

    $C_2(A) = \text{not}(A) + 1$ (utilisé très fréquemment)

Résumé nombres négatifs complément à 2^n

- Avantages notation «complément à 2»
  - Représentation naturelle des nombres négatifs
  - Cohérence avec le fonctionnement d'un compteur
  - Soustraction via complément à 2 et une addition
  - Même circuit pour les nombres non signés et signés en complément à 2 (C2)
    - Attention: débordements différents (carry, overflow)
  - une seule représentation de la valeur zéro
- Inconvénient de la notation «complément à 2»
  - Comparaison de nombre (voir ci-après)

Comparaison de nombres

- On utilise les valeurs de 1 à $2^{n-1}-1$ pour les nombres positifs,
  0 pour le nombre nul,
  et 2^n-1 à 2^{n-1} pour les nombres négatifs
- Soit : A=11110000 et B=01110000.
  Est-ce A est plus grand que B ?
  - Si A et B sont des nombres sans signe, alors OUI (A>B)
  - Si A et B sont des nombres signés dans la représentation «en complément à 2^n», alors NON (A<B)
    (car A est négatif et B est positif)

Exercices série II

1. En binaire, sur 8 bits, écrivez les nombres suivants
   - sans signe : $+128_{10}$ - 1000 0000
   - en notation «complément à 2» : -128_{10} - 1000 0000
   - en notation «complément à 2» : $+127_{10}$ - 0111 1111
   - en notation «complément à 2» : $+128_{10}$ - impossible
   - le complément à 2 de : $+128_{10}$ - -128 = 1000 0000
   - le complément à 2 de : -127_{10} - 0111 1111
   - en notation «signe-amplitude» : -127_{10} - 1111 1111
   - en notation «signe-amplitude» : $+128_{10}$ - impossible

Exercices série II

2. Comment se justifie la recette de cuisine pour calculer le complément à 2 d'un nombre «inverser tous les bits puis ajouter 1» ?
   En examinant les chiffres 1 à 1 depuis la droite, trouvez une autre recette.
3. Extension de nombres signés : comment étendre sur 2n bits un nombre de n bits, signé, en notation «complément à 2»?

Etendre un nombre en complément à 2 …

- Pour étendre, par exemple de 8 à 16 bits, un nombre sans signe, il suffit de le compléter avec des 0 sur sa gauche
  - Exemple : le nombre 8bits sans signe 1001 1100
    étendu sur 16 bits devient 0000 0000 1001 1100
- Qu'en est-il pour un nombre signé, en notation «complément à 2»?
  - Si le nombre est positif, on le complète avec des 0, comme un nombre sans signe
  - Si le nombre est négatif ?

… étendre un nombre en complément à 2

- Si le nombre est négatif, en passant de 8 à 16 bits on passe de «complément à 2^8» à «complément à 2^{16}» !
- Il faudra lui ajouter $2^{16} - 2^8$ = 1111111100000000
- Pour étendre de k bits un nombre signé, en notation «complément à 2», il faut le compléter sur sa gauche k copies du bit de signe

- Exemple : le nombre 8bits signé 1001 1100
  étendu sur 16 bits devient 1111 1111 1001 1100

Exercice série II

4. Ecrivez en binaire sur 8 bits en C2 (nombre signé) les nombres suivants:
   - -37 +37 = 0010 0101, -37 = C2(+37)= 1101 1011
   - +53 +53 = 0011 0101
5. Ecrivez en binaire sur 12 bits en C2 les mêmes nombres que ci-dessus, soit -37 et +53

   -37 = 1111 1101 1011

   +53 = 0000 0011 0101
6. Pour les deux représentations effectuez le calcul :
   - 53 – 37 Que constatez-vous ? Même résultat

53 – 37 = 53 + C2(37) =00 00 0011 0101 + 1111 1101 1011 = 0000 0001 0000

Exercices série III

- Quelles multiplications sont les plus faciles à faire en décimal?
- Et en binaire?
- Peut-on multiplier par dix en binaire à l'aide d'une simple addition?
  - Proposer une solution avec une seule addition.

Représentation d'information non numérique

- Définir une convention
- Convention pour chaque type d'information
  - texte
  - image
  - son
  - ….
- Il existe souvent plusieurs conventions pour un même type d'information. Exemple : une image
  - format tif, gif, jpeg, ……

Représentation des textes

- Le fichier texte est constitué d'une séquence de bits, organisés en bytes: un caractère de texte est alors représenté par un ou plusieurs bytes
- Par exemple, le programme `hello.c`, écrit en C:

```
#include <stdio.h>
int main()
{
   printf("hello, world\n");
}
```

est sauvé par l'éditeur dans le fichier **`hello.c`**, dont le contenu, byte par byte, est le suivant:

Code des caractères en décimal :

# 35	h 104	<sp> 32	, 44
i 105	> 62	<sp> 32	<sp> 32
n 110	\n 10	<sp> 32	w 119
c 99	i 105	p 112	o 111
l 108	n 110	r 114	r 114
u 117	t 116	i 105	l 108
d 100	<sp> 32	n 110	d 100
e 101	m 109	t 116	\ 92
<sp> 32	a 97	f 102	n 110
< 60	i 105	(40	" 34
s 115	n 110	" 34	) 41
t 116	(40	h 104	; 59
d 100	) 41	e 101	\n 10
i 105	\n 10	l 108	} 125
o 111	{ 123	l 108	
. 46	\n 10	o 111	

Code des caractères en hexadécimal :

#	23	h	68	<sp>	20	,	2C
i	69	>	3E	<sp>	20	<sp>	20
n	6E	\n	0A	<sp>	20	w	77
c	63	i	69	p	70	o	6F
l	6C	n	6E	r	72	r	72
u	75	t	74	i	69	l	6C
d	64	<sp>	20	n	6E	d	64
e	65	m	6D	t	74	\	5C
<sp>	20	a	61	f	66	n	6E
<	3C	i	69	(	28	"	22
s	73	n	6E	"	22	)	29
t	74	(	28	h	68	;	3B
d	64	)	29	e	65	\n	OA
I	69	\n	0A	l	6C	}	7D
o	6F	{	7B	l	6C		
.	2E	\n	0A	o	6F		

Code ASCII …

- Chaque symbole d'un texte est représenté par une certaine chaîne de bits, un code
- Le code le plus utilisé est celui de l'American National Standard Institute (ANSI), appelé ASCII (American Standard Code for Information Exchange)
- Seul le code sur 7 bits, 128 caractères, est standard !
  - caractères anglais : lettres (majuscules/minuscules), chiffres 0 à 9
  - caractères de ponctuation, arithmétique, …
  - certains caractères de contrôle (fin de ligne, tabulateur, etc)
- Un huitième bit a été ajouté par la suite, afin d'inclure des caractères d'autres langues. Mais, malheureusement, il n'y a pas une implémentation unique de cette solution

… code ASCII

- Exemple:

01000010	01101111	01101110	01101010	01101111	01110101	01110010
0x42	0x6F	0x6E	0x6A	0x6F	0x75	0x72
B	o	n	j	o	u	r

- Un huitième bit a été ajouté par la suite, afin d'inclure des caractères d'autres langues. Mais, malheureusement, il n'y a pas une implémentation unique de cette solution

Table ASCII sur 7 bits (code 0 à 127)

(standard)

Ctrl	Déc	Hex	Car	Code
^@	0	00		NUL
^A	1	01		SOH
^B	2	02		STX
^C	3	03		ETX
^D	4	04		EOT
^E	5	05		ENQ
^F	6	06		ACK
^G	7	07		BEL
^H	8	08		BS
^I	9	09		HT
^J	10	0A		LF
^K	11	0B		VT
^L	12	0C		FF
^M	13	0D		CR
^N	14	0E		SO
^O	15	0F		SI
^P	16	10		DLE
^Q	17	11		DC1
^R	18	12		DC2
^S	19	13		DC3
^T	20	14		DC4
^U	21	15		NAK
^V	22	16		SYN
^W	23	17		ETB
^X	24	18		CAN
^Y	25	19		EM
^Z	26	1A		SUB
^[	27	1B		ESC
^\	28	1C		FS
^]	29	1D		GS
^^	30	1E	▲	RS
^-	31	1F	▼	US

Déc	Hex	Car
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

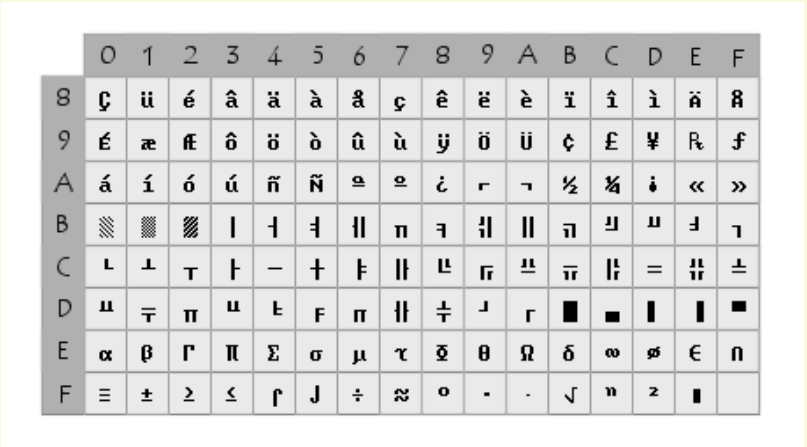
Déc	Hex	Car
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Déc	Hex	Car
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂*

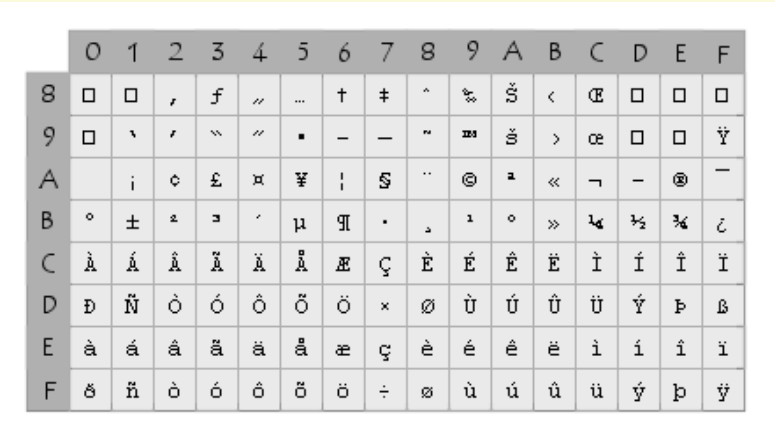
Code ASCII étendu, 8 bits (code 128 à 255)

Version premier

IBM PC



Version plus récente



Code Unicode

L’extension avec le huitième bit dans le code ASCII n’est pas standard. Il n'y a pas une implémentation unique de cette solution!

- Pour résoudre ces problèmes, un nouveau code, Unicode, a été adopté par des fabricants de matériel et de logiciel, utilisant 16 bits pour représenter chaque symbole. C'est le code utilisé par Java
- Un autre code, utilisant 32 bits par symbole, a été développé par l'ISO (International Organization for Standardization)

Code ASCII UTF-8 …

- Nouvelle norme du code ASCII
- Corrige les inconvénients des anciennes tables
- Compatible avec le standard Unicode
- Chaque caractère est codé sur une suite d'un à quatre octets

… code ASCII UTF-8

Exemples de codage UTF-8

Caractère	No caractère	Codage binaire UTF-8
A	65	**0**1000001
é	233	**110**00011 **10**101001
€	8364	**1110**0010 **10**000010 **10**101100

Dans toutes les chaînes de caractères UTF-8, on remarque que :

- tout octet de bit de poids fort nul => code un caractère US-ASCII sur un octet
- tout octet de bits de poids fort valant 11 est le premier octet d'un caractère codé sur plusieurs octets ;
- tout octet de bits de poids fort valant 10 est à l'intérieur d'un caractère codé sur plusieurs octets.

Conversion nombre ASCII en binaire

- ASCII à décimal :
  - utiliser la table ASCII
- puis conversion de décimal à binaire

Exercice :

N_{ASCII} = 0x333531

N_2 = ?

Nombre décimal représenté en ASCII : N_{10} = ..

N_{BCD} = 0x351 => N_{10} = 351 N_2 = ..

Représentation des images …

- Deux grandes catégories:
  - techniques de *bitmap*
  - techniques vectorielles
- Dans un bitmap, une image est une collection de points, appelés pixels (*picture element*). En noir et blanc, la valeur de chaque pixel est 0 ou 1. En couleur, chaque pixel est représenté par une combinaison de bits indiquant la couleur du pixel
- Assez souvent, la couleur d'un pixel est divisée en trois composantes, une pour chacune des trois couleurs primaires (rouge, vert, bleu, ou RGB)

… représentation des images

- Le principal problème des bitmaps est le changement d'échelle d'une image. Pour pallier à ce problème, on utilise des techniques vectorielles, où une image est représentée par un ensemble de lignes et courbes
- TrueType est un système vectoriel, développé par Microsoft et Apple, pour décrire le graphisme des caractères . PostScript, développé par Adobe, permet de décrire aussi bien des caractères que des données graphiques plus générales

Exemple d'image JPEG

Fichier image "Logo_REDS.jpg"

Reconfigurable & Embedded Digital Systems

Contenu fichier "Logo_REDS.jpg" (32'788 bytes)

logo_REDS.jpg

```
00000000h: FF D8 FF E0 00 10 4A 46 49 46 00 01 02 01 00 48 ; ÿØÿà..JFIF.....H
00000010h: 00 48 00 00 FF ED 13 74 50 68 6F 74 6F 73 68 6F ; .H..ÿí.tPhotosho
00000020h: 70 20 33 2E 30 00 38 42 49 4D 03 ED 0A 52 65 73 ; p 3.0.8BIM.í.Res
00000030h: 6F 6C 75 74 69 6F 6E 00 00 00 00 10 00 48 00 00 ; olution......H..
00000040h: 00 01 00 02 00 48 00 00 00 01 00 02 38 42 49 4D ; .....H......8BIM
00000050h: 04 0D 18 46 58 20 47 6C 6F 62 61 6C 20 4C 69 67 ; ...FX Global Lig
00000060h: 68 74 69 6E 67 20 41 6E 67 6C 65 00 00 00 00 04 ; hting Angle.....
00000070h: 00 00 00 1E 38 42 49 4D 04 19 12 46 58 20 47 6C ; ....8BIM...FX Gl
00000080h: 6F 62 61 6C 20 41 6C 74 69 74 75 64 65 00 00 00 ; obal Altitude...
00000090h: 00 04 00 00 00 1E 38 42 49 4D 03 F3 0B 50 72 69 ; ......8BIM.ó.Pri
000000a0h: 6E 74 20 46 6C 61 67 73 00 00 00 09 00 00 00 00 ; nt Flags........
000000b0h: 00 00 00 00 01 00 38 42 49 4D 04 0A 0E 43 6F 70 ; ......8BIM...Cop
000000c0h: 79 72 69 67 68 74 20 46 6C 61 67 00 00 00 00 01 ; yright Flag.....
000000d0h: 00 00 38 42 49 4D 27 10 14 4A 61 70 61 6E 65 73 ; ..8BIM'..Japanes
000000e0h: 65 20 50 72 69 6E 74 20 46 6C 61 67 73 00 00 00 ; e Print Flags...
000000f0h: 00 0A 00 01 00 00 00 00 00 00 00 02 38 42 49 4D ; ............8BIM
00000100h: 03 F5 17 43 6F 6C 6F 72 20 48 61 6C 66 74 6F 6E ; .õ.Color Halfton
00000110h: 65 20 53 65 74 74 69 6E 67 73 00 00 00 48 00 2F ; e Settings...H./
00000120h: 66 66 00 01 00 6C 66 66 00 06 00 00 00 00 00 01 ; ff...lff.......
```

Représentation des sons

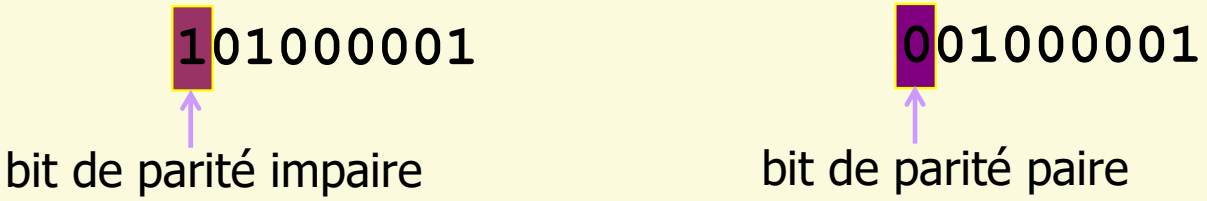
- La méthode la plus courante est l'échantillonnage de l'onde sonore à des intervalles réguliers. La valeur obtenue à chaque échantillon est codée en binaire
- Pour un CD, le son est échantillonné 44'100 fois par seconde, et les valeurs sont stockées en utilisant 16 bits par canal
- Un autre code très utilisé est MIDI (Musical Instrument Digital Interface). Dans ce cas, on ne code pas les sons, mais la façon de les produire dans un synthétiseur: quel instrument, quelle note et quelle durée. Ainsi, par exemple, un piano qui joue une note do pendant deux secondes est codé avec seulement trois bytes, au lieu des millions de bits nécessaires pour un CD

Compression des données

- *Run-length encoding*: lorsqu'une donnée est une suite de séquences avec la même valeur, on peut remplacer ces séquences par un code qui indique la valeur répétée et le nombre de fois qu'elle apparaît
- *Relative encoding*: cette méthode est utilisée pour enregistrer les différences entre blocs consécutifs de données, au lieu et en place des blocs entiers. Chaque bloc de données est codé donc en fonction de ses rapports avec le bloc précédent

Détection des erreurs de transmission

- Une méthode simple est celle de supposer que tout paquet de bits transmis contient un nombre impair de 1. Si un paquet est reçu avec un nombre pair de 1, une erreur a eu lieu
- Pour cela, il faut ajouter un bit additionnel à chaque paquet de données: le bit de parité. Un code ASCII deviendra par exemple un code à 9 bits. La valeur du bit de parité est choisie de telle façon que la parité soit respectée
- Il est possible d'avoir une parité impaire ou paire
- Exemple: le code ASCII de A devient:



Solution des exercices

Exercice série I

1. Déterminer la relation qui permet de connaître la valeur maximale Val_{max} , en décimal, d'une représentation en binaire de nombre entier non signé sur n bits

Solution: $Val_{max} = 2^n - 1$

Exemple: n = 4 bits ➔ $Val_{max} = 2^4 - 1 = 15_{10}$ $(= 1111_2)$

Exercice série I

2. Passage de binaire à décimal:

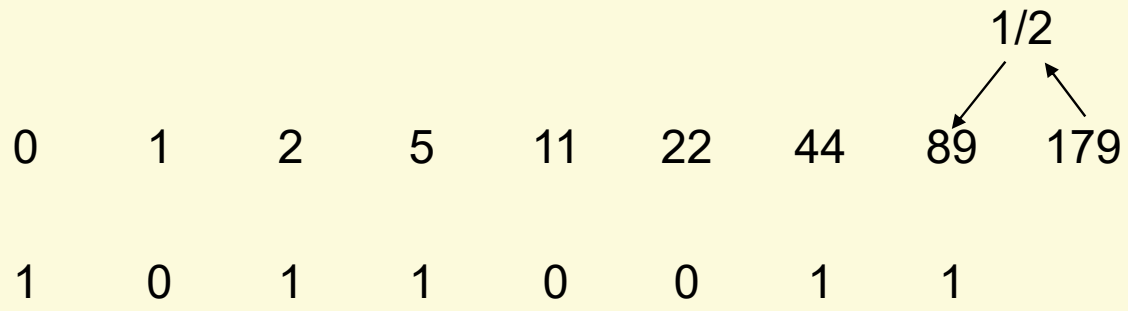
$$01101011 = 0x2^7+1x2^6+1x2^5+0x2^4+1x2^3+0x2^2+1x2^1+1x2^0$$
$$= 64+32+8+2+1$$
$$= 107$$

3. Passage d'hexadécimal à décimal:

$$A8CE = 10x16^3+8x16^2+12x16^1+14x16^0$$
$$= 40960+2048+192+14$$
$$= 43214$$

Exercices série I

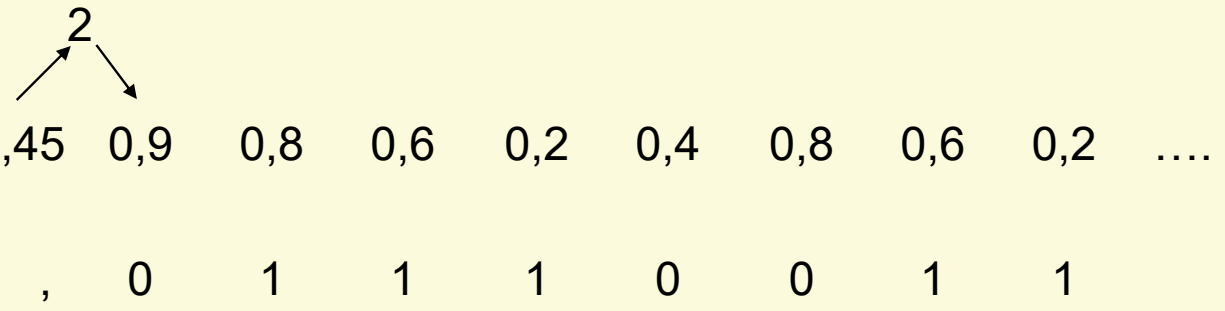
4. N_{10} = 179 = en binaire ?



N_2 = 10110011 = 0xB3

Exercices série I

5. N_{10} = 0,45 = en binaire ?



N_2 = $0,01110\overline{011}...$

Exercices série I

6. Passage de binaire à hexadécimal:

$N_2 = \underbrace{0110}_{6}\ \underbrace{1011}_{B}\ \underbrace{0101}_{5} =$

$N_{16} = 6\ B\ 5$

7. Passage d'hexadécimal à binaire:

0xA8CE = A 8 C E

1010 1000 1100 1110

Exercices série I

8. Passage de binaire à hexadécimal:

$N_2 = 10010110,100101\mathit{00} =$

$N_{16} = 9\ 6\ ,\ 9\ 4$

9. Passage d'hexadécimal à binaire:

B75.17 = B 7 5 , 1 7

1011 0111 0101,0001 0111

Exercices série I

10. Passage de BCD à décimal

N_{BCD} = 0110 1001 0101

N_{10} =

11. Passage de BCD à binaire

a) N_{BCD} = 0111

N_2 =

b) N_{BCD} = 0100 0101 1001

N_2 =

N_{BCD} en décimal : N_{10} = 459
convertit en binaire N_2 = 1 1100 1011