

# Représentation de l'information

## Codage des nombres

Tristan LEY

Formation ISN - Jour 1

14 et 21 février 2012



- 1 Information binaire
- 2 Codage des nombres entiers
  - Conversions entre bases, 10, 2 et 16
  - Codage des nombres entiers naturels
  - Codage des nombres entiers relatifs
- 3 Codage des nombres réels
  - Notation scientifique
  - En détails
  - Norme IEEE 754
  - Codage des réels : dégâts !
- 4 Ressources bibliographiques

# Information binaire : conclusion

Vous la connaissez déjà, alors on peut débiter par là !

- **Une information imparfaite**

*...discrétisation du réel*

# Information binaire : conclusion

Vous la connaissez déjà, alors on peut débiter par là !

- **Une information imparfaite**

*...discrétisation du réel*

- **Une information équivoque**

*...on doit connaître le code pour la rendre intelligible !*

# Information binaire : conclusion

Vous la connaissez déjà, alors on peut débiter par là !

- **Une information imparfaite**

*...discrétisation du réel*

- **Une information équivoque**

*...on doit connaître le code pour la rendre intelligible !*

- **Une information robuste**

*...« 1 » est aisément distinguable de « 0 »*

# Information binaire : conclusion

Vous la connaissez déjà, alors on peut débiter par là !

- **Une information imparfaite**

*...discrétisation du réel*

- **Une information équivoque**

*...on doit connaître le code pour la rendre intelligible !*

- **Une information robuste**

*...« 1 » est aisément distinguable de « 0 »*

- **Une information adaptable**

*...nombre de supports peuvent l'accueillir*

# Information binaire : conclusion

Vous la connaissez déjà, alors on peut débiter par là !

- **Une information imparfaite**

*...discrétisation du réel*

- **Une information équivoque**

*...on doit connaître le code pour la rendre intelligible !*

- **Une information robuste**

*...« 1 » est aisément distinguable de « 0 »*

- **Une information adaptable**

*...nombre de supports peuvent l'accueillir*

- **Une information universelle**

*...à  $\varepsilon$  près !*

# Information binaire

- Binaire est souvent traduit par « sous forme de 1 et de 0 »



# Information binaire

- Binaire est souvent traduit par « sous forme de 1 et de 0 »
- Il serait plus correct de dire « information codée avec un système à 2 états »

# Information binaire

- Binaire est souvent traduit par « sous forme de 1 et de 0 »
- Il serait plus correct de dire « information codée avec un système à 2 états »
- L'utilisation de 1 et 0 est néanmoins bien pratique et universellement adoptée

# Information binaire

- Binaire est souvent traduit par « sous forme de 1 et de 0 »
- Il serait plus correct de dire « information codée avec un système à 2 états »
- L'utilisation de 1 et 0 est néanmoins bien pratique et universellement adoptée
- En particulier lorsqu'il s'agit de coder des nombres !

## 1 Information binaire

## 2 Codage des nombres entiers

- Conversions entre bases, 10, 2 et 16
- Codage des nombres entiers naturels
- Codage des nombres entiers relatifs

## 3 Codage des nombres réels

- Notation scientifique
- En détails
- Norme IEEE 754
- Codage des réels : dégâts !

## 4 Ressources bibliographiques

# Conversions entre bases 10, 2 et 16

# Conversions entre bases 10, 2 et 16

- La base 2 est particulièrement adaptée au codage binaire des nombres

# Conversions entre bases 10, 2 et 16

- La base 2 est particulièrement adaptée au codage binaire des nombres
- Écrire le nombre  $N$  en base 2, c'est trouver l'écriture  $B_n B_{n-1} \dots B_1 B_0$  où  $B_k \in \{0; 1\}$  telle que :

$$N = \sum_{k=0}^n B_k 2^k$$

# Conversions entre bases 10, 2 et 16

- La base 2 est particulièrement adaptée au codage binaire des nombres
- Écrire le nombre  $N$  en base 2, c'est trouver l'écriture  $B_n B_{n-1} \dots B_1 B_0$  où  $B_k \in \{0; 1\}$  telle que :

$$N = \sum_{k=0}^n B_k 2^k$$

- On écrira indifféremment  $N_d$  ou  $N_{10}$  pour indiquer une écriture en base 10,  $N_b$  ou  $N_2$  pour une écriture en base 2

$$N_b = B_n B_{n-1} \dots B_1 B_0$$



# Codage des nombres entiers naturels

- Dits également **entiers non signés**

# Codage des nombres entiers naturels

- Dits également **entiers non signés**
- Codés en base 2 sur  $n$  bits, on peut donc coder les nombres compris entre 0 et  $2^n - 1$  (0 à 65535 sur 16 bits)

# Codage des nombres entiers naturels

- Dits également **entiers non signés**
- Codés en base 2 sur  $n$  bits, on peut donc coder les nombres compris entre 0 et  $2^n - 1$  (0 à 65535 sur 16 bits)
- On représente **tous** les bits, même les 0 à gauche (cela explicite le format)

# Codage des nombres entiers naturels

- Dits également **entiers non signés**
- Codés en base 2 sur  $n$  bits, on peut donc coder les nombres compris entre 0 et  $2^n - 1$  (0 à 65535 sur 16 bits)
- On représente **tous** les bits, même les 0 à gauche (cela explicite le format)
- On regroupe souvent les bits par 4 pour simplifier la lecture et la conversion en hexadécimal

# Codage des nombres entiers naturels

- Dits également **entiers non signés**
- Codés en base 2 sur  $n$  bits, on peut donc coder les nombres compris entre 0 et  $2^n - 1$  (0 à 65535 sur 16 bits)
- On représente **tous** les bits, même les 0 à gauche (cela explicite le format)
- On regroupe souvent les bits par 4 pour simplifier la lecture et la conversion en hexadécimal

**Exemple :**  $10_d = 1010_b$  sur 4 bits

# Codage des nombres entiers naturels

- Dits également **entiers non signés**
- Codés en base 2 sur  $n$  bits, on peut donc coder les nombres compris entre 0 et  $2^n - 1$  (0 à 65535 sur 16 bits)
- On représente **tous** les bits, même les 0 à gauche (cela explicite le format)
- On regroupe souvent les bits par 4 pour simplifier la lecture et la conversion en hexadécimal

**Exemple :**  $10_d = 1010_b$  sur 4 bits

$535_d = 0000\ 0010\ 0001\ 0111_b$  sur 16 bits

# Méthode par pesées

On cherche la plus grande puissance de 2 dans le nombre, puis la plus grande dans la différence, etc.

- $2^5 = 32 < 43$  et  $2^6 = 64 > 43$ , donc  
 $43 = 2^5 + (43 - 32) = 2^5 + 11.$

# Méthode par pesées

On cherche la plus grande puissance de 2 dans le nombre, puis la plus grande dans la différence, etc.

- $2^5 = 32 < 43$  et  $2^6 = 64 > 43$ , donc  
 $43 = 2^5 + (43 - 32) = 2^5 + 11.$
- $2^4 = 16 > 11$  et  $2^3 = 8 < 11$ , donc  
 $43 = 2^5 + 2^3 + (43 - 32 - 8) = 2^5 + 2^3 + 3.$



# Méthode par pesées

On cherche la plus grande puissance de 2 dans le nombre, puis la plus grande dans la différence, etc.

- $2^5 = 32 < 43$  et  $2^6 = 64 > 43$ , donc  
 $43 = 2^5 + (43 - 32) = 2^5 + 11.$
- $2^4 = 16 > 11$  et  $2^3 = 8 < 11$ , donc  
 $43 = 2^5 + 2^3 + (43 - 32 - 8) = 2^5 + 2^3 + 3.$
- $2^2 = 4 > 3$  et  $2^1 = 2 < 3$  donc  
 $43 = 2^5 + 2^3 + 2^1 + (43 - 32 - 8 - 2) = 2^5 + 2^3 + 2^1 + 1.$

# Méthode par pesées

On cherche la plus grande puissance de 2 dans le nombre, puis la plus grande dans la différence, etc.

- $2^5 = 32 < 43$  et  $2^6 = 64 > 43$ , donc  
 $43 = 2^5 + (43 - 32) = 2^5 + 11.$
- $2^4 = 16 > 11$  et  $2^3 = 8 < 11$ , donc  
 $43 = 2^5 + 2^3 + (43 - 32 - 8) = 2^5 + 2^3 + 3.$
- $2^2 = 4 > 3$  et  $2^1 = 2 < 3$  donc  
 $43 = 2^5 + 2^3 + 2^1 + (43 - 32 - 8 - 2) = 2^5 + 2^3 + 2^1 + 1.$
- $2^0 = 1$ , on a donc le nombre en entier :

$$43_d = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 101011_b$$

# Méthode de Horner

- Enchaînements de divisions euclidiennes par 2 du nombre décimal (puis des quotients successifs) jusqu'à ce le quotient vaille 0.

# Méthode de Horner

- Enchaînements de divisions euclidiennes par 2 du nombre décimal (puis des quotients successifs) jusqu'à ce le quotient vaille 0.
- On obtient le codage binaire du nombre en inscrivant les différents restes à la suite, en partant du dernier et en remontant vers le premier.

# Méthode de Horner

$$43 = 2 \cdot 21 + \mathbf{1}$$

# Méthode de Horner

$$\begin{aligned} 43 &= 2 \cdot 21 + \mathbf{1} \\ &= 2 \cdot (2 \cdot 10 + \mathbf{1}) + \mathbf{1} \end{aligned}$$

# Méthode de Horner

$$\begin{aligned}43 &= 2 \cdot 21 + \mathbf{1} \\&= 2 \cdot (2 \cdot 10 + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot 5 + \mathbf{0}) + \mathbf{1}) + \mathbf{1}\end{aligned}$$

# Méthode de Horner

$$\begin{aligned}43 &= 2 \cdot 21 + \mathbf{1} \\&= 2 \cdot (2 \cdot 10 + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot 5 + \mathbf{0}) + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot 2 + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{1}\end{aligned}$$



# Méthode de Horner

$$\begin{aligned}43 &= 2 \cdot 21 + \mathbf{1} \\&= 2 \cdot (2 \cdot 10 + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot 5 + \mathbf{0}) + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot 2 + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot (2 \cdot 1 + \mathbf{0}) + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{1}\end{aligned}$$

# Méthode de Horner

$$\begin{aligned}43 &= 2 \cdot 21 + \mathbf{1} \\&= 2 \cdot (2 \cdot 10 + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot 5 + \mathbf{0}) + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot 2 + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot (2 \cdot 1 + \mathbf{0}) + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot (2 \cdot (2 \cdot \mathbf{0} + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{1}\end{aligned}$$

# Méthode de Horner

$$\begin{aligned}43 &= 2 \cdot 21 + \mathbf{1} \\&= 2 \cdot (2 \cdot 10 + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot 5 + \mathbf{0}) + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot 2 + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot (2 \cdot 1 + \mathbf{0}) + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{1} \\&= 2 \cdot (2 \cdot (2 \cdot (2 \cdot (2 \cdot (2 \cdot \underline{0} + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{0}) + \mathbf{1}) + \mathbf{1}\end{aligned}$$

Donc :

$$43_d = 101011_b$$

# Représentation hexadécimale

- Le binaire, c'est long (écriture, mémorisation) !

# Représentation hexadécimale

- Le binaire, c'est long (écriture, mémorisation) !
- $16 = 2^4$  : on « découpe » un nombre binaire en tranches de 4 bits (depuis la droite)

# Représentation hexadécimale

- Le binaire, c'est long (écriture, mémorisation) !
- $16 = 2^4$  : on « découpe » un nombre binaire en tranches de 4 bits (depuis la droite)
- On convertit ces tranches en base 16 (facile !)

# Représentation hexadécimale

- Le binaire, c'est long (écriture, mémorisation) !
- $16 = 2^4$  : on « découpe » un nombre binaire en tranches de 4 bits (depuis la droite)
- On convertit ces tranches en base 16 (facile !)
- On obtient une représentation hexadécimale du nombre

# Représentation hexadécimale

- Le binaire, c'est long (écriture, mémorisation) !
- $16 = 2^4$  : on « découpe » un nombre binaire en tranches de 4 bits (depuis la droite)
- On convertit ces tranches en base 16 (facile !)
- On obtient une représentation hexadécimale du nombre
- **Exemple** :  $947_d = 11\ 1011\ 0011_b$



# Représentation hexadécimale

- Le binaire, c'est long (écriture, mémorisation) !
- $16 = 2^4$  : on « découpe » un nombre binaire en tranches de 4 bits (depuis la droite)
- On convertit ces tranches en base 16 (facile !)
- On obtient une représentation hexadécimale du nombre
- **Exemple** :  $947_d = 11\ 1011\ 0011_b$   
 $11_b = 0011_b = 3_d = 3_h$  et  $1011_b = 11_d = B_h$

# Représentation hexadécimale

- Le binaire, c'est long (écriture, mémorisation) !
- $16 = 2^4$  : on « découpe » un nombre binaire en tranches de 4 bits (depuis la droite)
- On convertit ces tranches en base 16 (facile !)
- On obtient une représentation hexadécimale du nombre
- **Exemple** :  $947_d = 11\ 1011\ 0011_b$   
 $11_b = 0011_b = 3_d = 3_h$  et  $1011_b = 11_d = B_h$

$$947_d = 11\ 1011\ 0011_b = 3B3_h$$

# Représentation hexadécimale

- Le binaire, c'est long (écriture, mémorisation) !
- $16 = 2^4$  : on « découpe » un nombre binaire en tranches de 4 bits (depuis la droite)
- On convertit ces tranches en base 16 (facile !)
- On obtient une représentation hexadécimale du nombre

- **Exemple** :  $947_d = 11\ 1011\ 0011_b$   
 $11_b = 0011_b = 3_d = 3_h$  et  $1011_b = 11_d = B_h$

$$947_d = 11\ 1011\ 0011_b = 3B3_h$$

- La conversion réciproque suit le même principe que pour le binaire (avec des puissances de 16)

# Codage des nombres entiers relatifs

- Il faut gérer le signe : c'est plus délicat !

# Codage des nombres entiers relatifs

- Il faut gérer le signe : c'est plus délicat !
- Difficulté : l'addition doit être facile et efficace

# Codage des nombres entiers relatifs

- Il faut gérer le signe : c'est plus délicat !
- Difficulté : l'addition doit être facile et efficace (c'est l'opération la plus fréquente !)

# Codage des nombres entiers relatifs

- Il faut gérer le signe : c'est plus délicat !
- Difficulté : l'addition doit être facile et efficace (c'est l'opération la plus fréquente !)
- Nécessité d'une représentation des entiers naturels « transparente pour l'addition »

*Sinon il faudra systématiquement distinguer des cas selon les signes des opérandes*

*Ce sera coûteux en temps machine !*

# Représentation signe / valeur absolue

- Le signe est une donnée binaire : soit « + », soit « - »



# Représentation signe / valeur absolue

- Le signe est une donnée binaire : soit « + », soit « - »
- Idée simple : le coder *via* 1 bit

# Représentation signe / valeur absolue

- Le signe est une donnée binaire : soit « + », soit « - »
- Idée simple : le coder *via* 1 bit
- À y être : le bit le plus à gauche (bit de poids fort, ou MSB — ***Most Significant Bit*** en anglais)

# Représentation signe / valeur absolue

- Le signe est une donnée binaire : soit « + », soit « - »
- Idée simple : le coder *via* 1 bit
- À y être : le bit le plus à gauche (bit de poids fort, ou MSB — ***Most Significant Bit*** en anglais)
- **Exemple** : sur 8 bits

$$110000001_b = -65_d$$

$$010000001_b = 65_d$$

# Représentation signe / valeur absolue : inconvenients

- Deux codages pour 0 :  $+0$  et  $-0$

# Représentation signe / valeur absolue : inconvenients

- Deux codages pour 0 :  $+0$  et  $-0$
- Le bit de signe n'a pas le même statut que les autres bits !

# Représentation signe / valeur absolue : inconvénients

- Deux codages pour 0 :  $+0$  et  $-0$
- Le bit de signe n'a pas le même statut que les autres bits !
- On n'est plus dans une *numération* positionnelle, on est face à un *codage* binaire

# Représentation signe / valeur absolue : inconvénients

- Deux codages pour 0 :  $+0$  et  $-0$
- Le bit de signe n'a pas le même statut que les autres bits !
- On n'est plus dans une *numération* positionnelle, on est face à un *codage* binaire
- Conséquence : l'addition binaire usuelle ne fonctionne plus !

$$00000011_b + 10000100_b = 10000111_b$$

soit  $3 + (-4) = -7$  au lieu de  $-1$

# Représentation signe / valeur absolue : inconvénients

- Deux codages pour 0 :  $+0$  et  $-0$
- Le bit de signe n'a pas le même statut que les autres bits !
- On n'est plus dans une *numération* positionnelle, on est face à un *codage* binaire
- Conséquence : l'addition binaire usuelle ne fonctionne plus !

$$00000011_b + 10000100_b = 10000111_b$$

soit  $3 + (-4) = -7$  au lieu de  $-1$

- Il doit y avoir moyen de faire mieux...



# Représentation par complément à 2

- C'est encore un codage (pas un système de numération)

# Représentation par complément à 2

- C'est encore un codage (pas un système de numération)
- On représente des entiers *relatifs* à l'aide d'entiers *naturels*

L'intervalle entier  $\llbracket -2^{n-1}; 2^{n-1} - 1 \rrbracket$  devient  $\llbracket 0; 2^n - 1 \rrbracket$

# Représentation par complément à 2

- C'est encore un codage (pas un système de numération)
- On représente des entiers *relatifs* à l'aide d'entiers *naturels*  
L'intervalle entier  $\llbracket -2^{n-1}; 2^{n-1} - 1 \rrbracket$  devient  $\llbracket 0; 2^n - 1 \rrbracket$
- Les entiers de 0 à  $2^{n-1} - 1$  sont représentés par eux-mêmes  
Les entiers de  $-2^{n-1}$  à  $-1$  sont représentées par les entiers de  $2^{n-1}$  à  $2^n - 1$

# Représentation par complément à 2

- C'est encore un codage (pas un système de numération)
- On représente des entiers *relatifs* à l'aide d'entiers *naturels*  
L'intervalle entier  $\llbracket -2^{n-1}; 2^{n-1} - 1 \rrbracket$  devient  $\llbracket 0; 2^n - 1 \rrbracket$
- Les entiers de 0 à  $2^{n-1} - 1$  sont représentés par eux-mêmes  
Les entiers de  $-2^{n-1}$  à  $-1$  sont représentées par les entiers de  $2^{n-1}$  à  $2^n - 1$
- **Par définition**  $A \in \llbracket -2^{n-1}; -1 \rrbracket$  est codé par  $2^n + A$

# Représentation par complément à 2

- C'est encore un codage (pas un système de numération)
- On représente des entiers *relatifs* à l'aide d'entiers *naturels*  
L'intervalle entier  $\llbracket -2^{n-1}; 2^{n-1} - 1 \rrbracket$  devient  $\llbracket 0; 2^n - 1 \rrbracket$
- Les entiers de 0 à  $2^{n-1} - 1$  sont représentés par eux-mêmes  
Les entiers de  $-2^{n-1}$  à  $-1$  sont représentées par les entiers de  $2^{n-1}$  à  $2^n - 1$
- **Par définition**  $A \in \llbracket -2^{n-1}; -1 \rrbracket$  **est codé par**  $2^n + A$
- Le signe de l'entier est porté par le bit de poids fort ( $0 \leftrightarrow +$ ,  $1 \leftrightarrow -$ )

*Déterminer le signe d'un entier naturel est donc également très facile*

# Complément à 2 : avantages

- 0 n'est codé que d'une seule façon

# Complément à 2 : avantages

- 0 n'est codé que d'une seule façon
- L'addition et la soustraction binaires fonctionnent

*Cela revient à faire des additions de distances à  $-2^n$  modulo  $2^n$*

## Complément à 2 : avantages

- 0 n'est codé que d'une seule façon
- L'addition et la soustraction binaires fonctionnent

*Cela revient à faire des additions de distances à  $-2^n$  modulo  $2^n$*

- Il y a toujours des cas particuliers à gérer (liés aux dépassements de capacité)

*Ce phénomène survient **toujours** lorsqu'on calcule avec un nombre **fini** de bits*



# Complément à 2 : avantages

- 0 n'est codé que d'une seule façon
- L'addition et la soustraction binaires fonctionnent

*Cela revient à faire des additions de distances à  $-2^n$  modulo  $2^n$*

- Il y a toujours des cas particuliers à gérer (liés aux dépassements de capacité)

*Ce phénomène survient **toujours** lorsqu'on calcule avec un nombre **fini** de bits*

- Un bit « qui sort » à gauche (*carry bit*) ne cause pas toujours une erreur. On peut avoir un résultat faux sans bit qui déborde

*Les erreurs viennent d'un report de l'avant-dernier bit dans le dernier bit sans report du dernier bit en dehors du mot, ou bien à un report en dehors du mot sans report de l'avant dernier sur le dernier bit*

- 1 Information binaire
- 2 Codage des nombres entiers
  - Conversions entre bases, 10, 2 et 16
  - Codage des nombres entiers naturels
  - Codage des nombres entiers relatifs
- 3 **Codage des nombres réels**
  - Notation scientifique
  - En détails
  - Norme IEEE 754
  - Codage des réels : dégâts !
- 4 Ressources bibliographiques

# Codage des nombres réels : notation scientifique

- Difficulté : coder efficacement la position de la virgule

# Codage des nombres réels : notation scientifique

- Difficulté : coder efficacement la position de la virgule
- Solution : la notation scientifique !

$$x = \pm a \cdot 10^n \text{ où } a \in [1; 10[ \text{ et } n \in \mathbb{Z}$$

$\pm$  est le signe,  $a$  la mantisse,  $n$  l'exposant

*Il n'y a donc qu'un seul chiffre non nul à gauche de la virgule*

# Codage des nombres réels : notation scientifique

- Difficulté : coder efficacement la position de la virgule
- Solution : la notation scientifique !

$$x = \pm \cdot a \cdot 10^n \text{ où } a \in [1; 10[ \text{ et } n \in \mathbb{Z}$$

$\pm$  est le signe,  $a$  la mantisse,  $n$  l'exposant

*Il n'y a donc qu'un seul chiffre non nul à gauche de la virgule*

- En binaire, cela donne :

$$(-1)^{\text{signe}} \cdot \text{mantisse} \cdot 2^{\text{exposant}}$$

# Codage des nombres réels : détails

- le signe est porté par le bit de poids fort ( $0 \leftrightarrow +$ ,  $1 \leftrightarrow -$ , un classique...)

# Codage des nombres réels : détails

- le signe est porté par le bit de poids fort ( $0 \leftrightarrow +$ ,  $1 \leftrightarrow -$ , un classique...)
- la base (2) n'est pas codée (elle est implicite)

# Codage des nombres réels : détails

- le signe est porté par le bit de poids fort ( $0 \leftrightarrow +$ ,  $1 \leftrightarrow -$ , un classique...)
- la base (2) n'est pas codée (elle est implicite)
- l'exposant sera codé biaisé

*Cela veut dire qu'on lui ajoute une constante pour le rendre positif*

*On n'a ainsi pas à coder son signe et à gaspiller 1 bit*

*Cela facilite la comparaison des nombres par l'ordinateur*



# Codage des nombres réels : détails

- le signe est porté par le bit de poids fort ( $0 \leftrightarrow +$ ,  $1 \leftrightarrow -$ , un classique...)

- la base (2) n'est pas codée (elle est implicite)

- l'exposant sera codé biaisé

*Cela veut dire qu'on lui ajoute une constante pour le rendre positif*

*On n'a ainsi pas à coder son signe et à gaspiller 1 bit*

*Cela facilite la comparaison des nombres par l'ordinateur*

- l'exposant est choisit pour que la mantisse soit de la forme 1,... (en binaire)

*On évite ainsi de coder le 1 devant la virgule dans l'écriture de la mantisse*

*On gagne ainsi un autre bit !*

# Codage des nombres réels : norme IEEE 754

## Problèmes

- Combien de bits pour coder un nombre réel ?

# Codage des nombres réels : norme IEEE 754

## Problèmes

- Combien de bits pour coder un nombre réel ?
- Parmi eux, combien pour coder la mantisse ?  
Combien pour celui de l'exposant ?

# Codage des nombres réels : norme IEEE 754

## Problèmes

- Combien de bits pour coder un nombre réel ?
- Parmi eux, combien pour coder la mantisse ?  
Combien pour celui de l'exposant ?

***Un même programme fonctionnant sur différents ordinateurs doit donner les mêmes résultats !***

# Codage des nombres réels : norme IEEE 754

## Problèmes

- Combien de bits pour coder un nombre réel ?
- Parmi eux, combien pour coder la mantisse ?  
Combien pour celui de l'exposant ?

***Un même programme fonctionnant sur différents ordinateurs doit donner les mêmes résultats !***

Réponse (1985)

**La norme IEEE 754**

# La norme IEEE 754 en détails

- Elle fixe plusieurs formats (32 bits : simple précision ; 64 bits : double précision, etc.)

# La norme IEEE 754 en détails

- Elle fixe plusieurs formats (32 bits : simple précision ; 64 bits : double précision, etc.)
- Pour chaque format, elle définit le nombre de bits qui codent mantisse et exposant

# La norme IEEE 754 en détails

- Elle fixe plusieurs formats (32 bits : simple précision ; 64 bits : double précision, etc.)
- Pour chaque format, elle définit le nombre de bits qui codent mantisse et exposant
- Elle définit les opérations arithmétiques courantes



# La norme IEEE 754 en détails

- Elle fixe plusieurs formats (32 bits : simple précision ; 64 bits : double précision, etc.)
- Pour chaque format, elle définit le nombre de bits qui codent mantisse et exposant
- Elle définit les opérations arithmétiques courantes
- Elle prévoit plusieurs méthode d'arrondis

# La norme IEEE 754 en détails

- Elle fixe plusieurs formats (32 bits : simple précision ; 64 bits : double précision, etc.)
- Pour chaque format, elle définit le nombre de bits qui codent mantisse et exposant
- Elle définit les opérations arithmétiques courantes
- Elle prévoit plusieurs méthode d'arrondis
- Elle normalise la valeur du biais pour coder l'exposant

Si on lui alloue  $e$  bits, le biais est  $B = 2^{e-1} - 1$

# Cas particuliers de la norme IEEE 754

- 0 peut être codé de 2 façons différentes ( $+0$  et  $-0$ ) !

# Cas particuliers de la norme IEEE 754

- 0 peut être codé de 2 façons différentes ( $+0$  et  $-0$ ) !
- Le bit de poids fort de la mantisse (*celui qui est caché*) est déterminé par la valeur de l'exposant décalé.

# Cas particuliers de la norme IEEE 754

- 0 peut être codé de 2 façons différentes (+0 et -0) !
- Le bit de poids fort de la mantisse (*celui qui est caché*) est déterminé par la valeur de l'exposant décalé.
- Si l'exposant décalé est différent de 0 et de  $2^e - 1$ , le bit (*caché*) de poids fort de la mantisse est 1 : **le nombre est dit normalisé**

# Cas particuliers de la norme IEEE 754

- 0 peut être codé de 2 façons différentes (+0 et -0) !
- Le bit de poids fort de la mantisse (*celui qui est caché*) est déterminé par la valeur de l'exposant décalé.
- Si l'exposant décalé est différent de 0 et de  $2^e - 1$ , le bit (*caché*) de poids fort de la mantisse est 1 : **le nombre est dit normalisé**
- Si l'exposant décalé est nul, le bit de poids fort de la mantisse est nul : **le nombre est dénormalisé**

# Cas particuliers de la norme IEEE 754

Soit une mantisse codée par le nombre binaire  $b_1b_2\dots b_k$

- Si l'exposant décalé  $E$  **n'est pas** nul, alors le nombre décimal correspondant vaut

$$(-1)^s \left( 1 + \frac{\sum_{i=0}^{k-1} b_i 2^i}{2^k} \right) \cdot 2^{E-B}$$

- Si l'exposant décalé est nul, alors le nombre décimal correspondant vaut

$$(-1)^s \left( 0 + \frac{\sum_{i=0}^{k-1} b_i 2^i}{2^k} \right) \cdot 2^{1-B}$$

**1 - B** car le bit caché vaut 0, donc on a « perdu » une puissance de 2 !

# Cas particuliers de la norme IEEE 754

## Résumé des exceptions

Type	Exposant décalé	Mantisse
Zéros	0	0
Nombres dénormalisés	0	différente de 0
Nombres normalisés	1 à $2^e - 2$	quelconque
Infinis	$2^e - 1$	0
NaNs	$2^e - 1$	différente de 0



# Codage des réels : dégâts !

```
Python 3.2.2 (default, Sep  5 2011, 21:17:14)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license()>>> 1-0.9
0.099999999999999998
```

**Normal ! Décomposition en puissances *inverses* de 2 :**

$$\begin{aligned}0,2 &= 0 \cdot 2^{-1} + 0,2 \\&= 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0,2 \\&= 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0,075 \\&= \dots\end{aligned}$$

# Codage des réels : dégâts !

Plus lisible :

$$0,2 \cdot 2 = 0,4 = \mathbf{0} + 0,4$$

$$0,4 \cdot 2 = 0,8 = \mathbf{0} + 0,8$$

$$0,8 \cdot 2 = 1,6 = \mathbf{1} + 0,6$$

$$0,6 \cdot 2 = 1,2 = \mathbf{1} + 0,2$$

$$0,2 \cdot 2 = 0,4 = \mathbf{0} + 0,$$

...

Donc :

$$0,2_d = 0,00110011..._b = 1,1001100... \cdot 2^{-3}_b$$

- 1 Information binaire
- 2 Codage des nombres entiers
  - Conversions entre bases, 10, 2 et 16
  - Codage des nombres entiers naturels
  - Codage des nombres entiers relatifs
- 3 Codage des nombres réels
  - Notation scientifique
  - En détails
  - Norme IEEE 754
  - Codage des réels : dégâts !
- 4 Ressources bibliographiques

# Ressources bibliographiques

- [http://www.arcanapercipio.com/lessons/l\\_information\\_binaire/l\\_information\\_binaire.html](http://www.arcanapercipio.com/lessons/l_information_binaire/l_information_binaire.html)
- [http://www.arcanapercipio.com/lessons/codage\\_binaire\\_des\\_nombres/codage\\_binaire\\_des\\_nombres.html](http://www.arcanapercipio.com/lessons/codage_binaire_des_nombres/codage_binaire_des_nombres.html)
- <http://www.liafa.jussieu.fr/~carton/Enseignement/Architecture/Cours/Coding/index.html>
- [http://fr.wikipedia.org/wiki/IEEE\\_754](http://fr.wikipedia.org/wiki/IEEE_754)
- [http://www.binaryconvert.com/convert\\_float.html](http://www.binaryconvert.com/convert_float.html)

# Erratum

- Erreur dans Arcana Percipio dans la table des valeurs particulières de la norme IEEE 754, sur les nombres dénormalisés (Mantisse : tous les bits à 0)

# Erratum

- Erreur dans Arcana Percipio dans la table des valeurs particulières de la norme IEEE 754, sur les nombres dénormalisés (Mantisse : tous les bits à 0)
- Erreur dans Arcana Percipio, toujours, dans l'application d'exploration du codage IEEE 754, qui divise par 2 une fois de trop pour les nombres dénormalisés !

# Erratum

- Erreur dans Arcana Percipio dans la table des valeurs particulières de la norme IEEE 754, sur les nombres dénormalisés (Mantisse : tous les bits à 0)
- Erreur dans Arcana Percipio, toujours, dans l'application d'exploration du codage IEEE 754, qui divise par 2 une fois de trop pour les nombres dénormalisés !
- Erreur dans Carton : la plus petite valeur positive représentable est  $2^{-23} \cdot 2^{-126} = 2^{-249}$  (nombre dénormalisé)