DIRO
IFT 1215

INTRODUCTION AUX SYSTÈMES INFORMATIQUES

REPRÉSENTATION FLOTTANTE

Max Mignotte

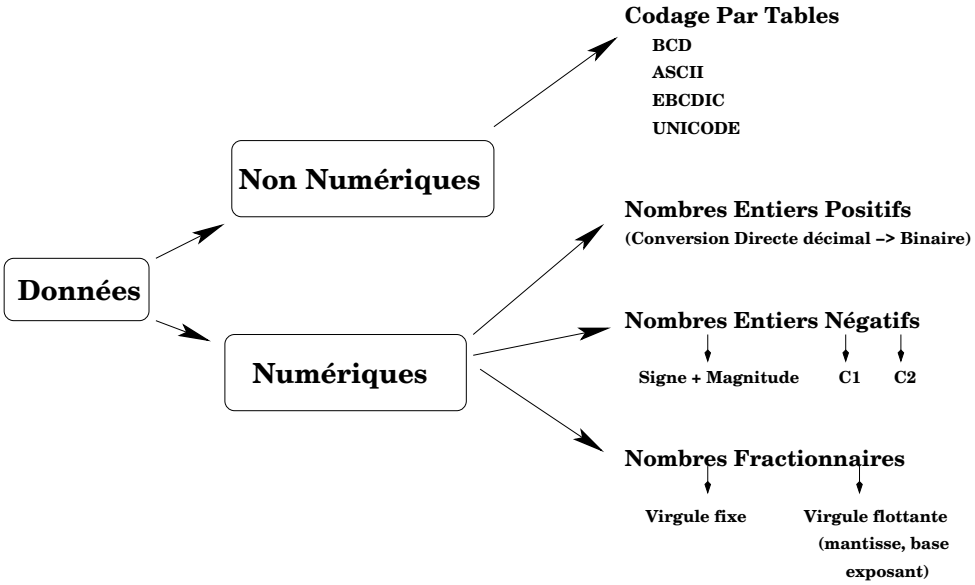
Département d'Informatique et de Recherche Opérationnelle
Http: //www.iro.umontreal.ca/~mignotte/
E-mail: mignotte@iro.umontreal.ca

REPRÉSENTATION FLOTTANTE

SOMMAIRE

REPRÉSENTATION FLOTTANTE
RÉCAPITULATIF

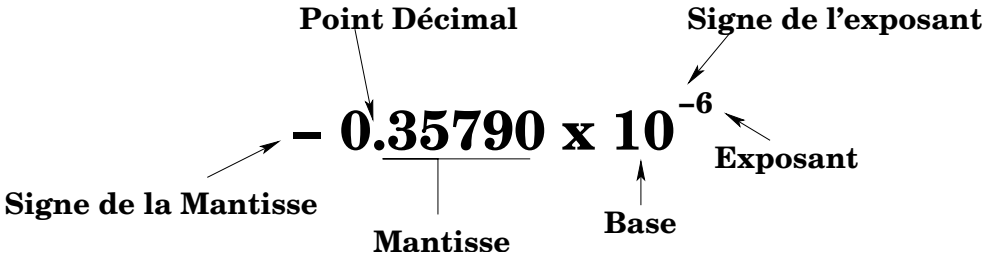
Récapitulatif



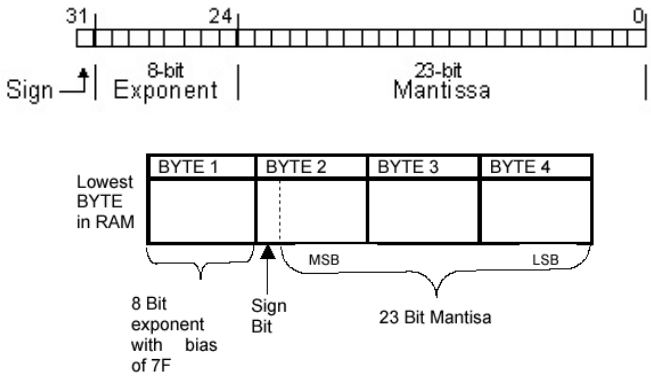
REPRÉSENTATION FLOTTANTE
INTRODUCTION

Introduction

• La représentation flottante permet la représentation de nombres très petits ou très grands en utilisant très peu de digits, au prix d'une représentation avec une certaine précision



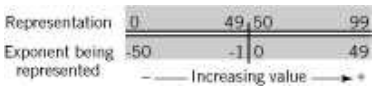
Format Flottant sur 32 bits



• Bit de Signe (0 positive ou 1 négative)

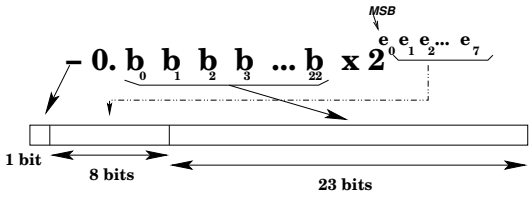
• Exposant de -127 à +128 avec biais de 127 (exemple: vraie exposant de -6 ▷ 127-6 = 121)

Biais de 50 ▷

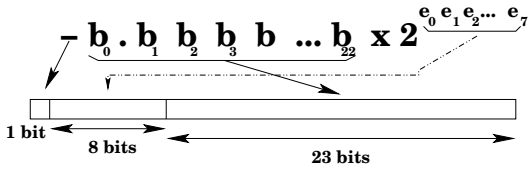


REPRÉSENTATION FLOTTANTE
CONVERSION EN BASE 10

1



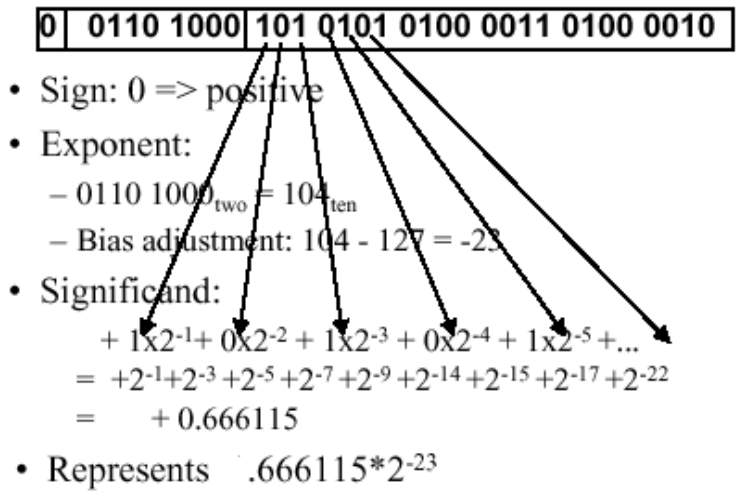
2



• **Normalisation** : On peut toujours s'arranger pour que $b_0 \neq 0$ (dans certain système il n'est pas stocké ▷ [bit caché])

Exemple

IEEE Format 1 sans bit caché conversion en base 10



REPRÉSENTATION FLOTTANTE
CONVERSION EN FLOTTANT

Exemples : Base 10 Format 1

Ex1: Convertir $+9.9520_{10}$ en notation flottante avec un exposant biaisé de 50 (2 digits) et 5 digits pour la mantisse

- Notation flottante normalisée
  $\triangleright\ 0.99520 \times 10^{1}$
- Chiffre Positif, exposant de $50 + 1 = 51$

Ex2: Convertir $+0.067850_{10}$ pour le même format

- Notation flottante normalisée
  $\triangleright\ 0.67850 \times 10^{-1}$
- Chiffre Positif, exposant de $50 - 1 = 49$
  $\triangleright\ 0\,49\,67850$

Ex3: Que représente $\triangleright\ 0\,52\,10020$ dans ce format ?

- Chiffre positif, exposant de $52 - 50 = 2$
  $\triangleright +\ 0.10020 \times 10^{2} = 10.020$

REPRÉSENTATION FLOTTANTE
ARITHMÉTIQUE EN FLOTTANT

Addition & Soustraction

Ex: Faire $9.9520_{10} + 0.067850_{10}$
en notation flottante avec un exposant biaisé de 50 (2 digits) et 5 digits pour la mantisse

$-1-$ *Conversion* $9.9520_{10} \triangleright$ 0 51 99520,

$0.067850_{10} \triangleright$ 0 49 67850

$-2-$ *Aligner le plus faible exposant sur le plus fort*

0 49 67850 $\triangleright$ 0 51 0067850

$-3-$ Additionner les deux mantisses

$$\begin{array}{ll} 0\,51 & 99520 \\ 0\,51 & 0067850 \\ \hline 0\,51 & (1)0019850 \\ 0\,52 & 10019850 \end{array}$$

$-4-$ On arrondi car la mantisse à seulement 5 chiffres

0 52 10020

- Perte de Précision !

REPRÉSENTATION FLOTTANTE
ARITHMÉTIQUE EN FLOTTANT

Multiplication & Division

- On multiplie les mantisses et on ajoute les exposants

Ex: Faire $9.9520_{10} \times 0.067850_{10}$
en notation flottante avec un exposant biaisé de 50 (2 digits) et 5 digits pour la mantisse

$-1-$ *Conversion* $9.9520_{10} \triangleright$ 0 51 99520,

$0.067850_{10} \triangleright$ 0 49 67850

$-2-$ On ajoute les exposants

$$\begin{array}{r} 51 = 50 \ + 1 \\ 49 = 50 \ - 1 \\ - - - - - - - - - \\ 50 + 0 = 50 \end{array}$$

$-3-$ On multiplie les deux mantisses
$0.99520 \times 0.67850 = 0.6752432$

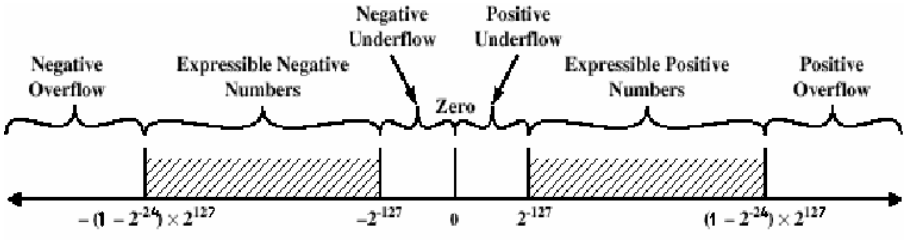
$-4-$ On normalise et on tronque $\boxed{0\ 50\ 67524}$

REPRÉSENTATION FLOTTANTE
IEEE 754 STANDARD

IEEE 754 Standard

Simple Précision

- Bit de signe 1 bit (Bit caché)
- Exposant sur 8 bits (biais de 127)
- Mantisse de 24 bits



Double Précision

- Bit de signe 1 bit
- Exposant sur 11 bits (biais de 1023)
- Mantisse de 52 bits

- Format 2
  i.e., avec normalisation du style $1.MMMMM...$
- Bit caché
- Le nombre 0 est défini par une mantisse remplie de 0 et un exposant remplie de 0
- Le nombre ∞ est défini par une mantisse remplie de 0 et un exposant remplie de 1
- NaN est représente par
  0 111111111 011 0111 0000 0000 0000 0000

REPRÉSENTATION FLOTTANTE
PACKED DECIMAL FORMAT

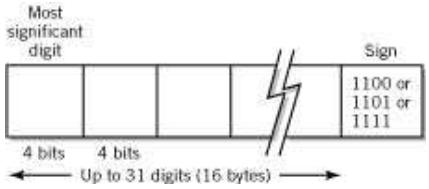
Format décimal empaqueté

• Dans certaine application, il est important de maintenir une précision parfaite pour la représentation des nombres réels (ex: dollars et cents)

• Quelque langage de haut niveau (ex: Cobol) fournit un format qui permet de spécifier un nombre de décimal désiré

• Chaque digit du nombre en base 10 est codé en BCD

• Utilisé par IBM System 370/390 et Compq Alpha



Exemple: $-324.6 = 0000\ 0011\ 0010\ 0100\ 0110\ 1101$

Programmation

Calcul avec des Entiers

- Plus facile pour l'ordinateur
- Très haute précision ou même sans perte de précision
- Plus rapide (moins d'accès mémoire, circuit logique plus simple, etc.)
- Langage C : *short int* (16 bits), *int* (32 bits), *long* (64 bits)

Calcul avec des flottants

- Variable ou constante avec une partie fractionnaire
- Perte de précision
- Moins rapide
- Langage C : simple précision *float* (32 bits), double précision *double* (64 bits), quadruple précision *long double* (128 bits)