Nombres et manipulations

Julien Baste

IUT de Lille

Séance 03

2025/2026

Codage de nombre

En informatique on a besoin des nombres. Il faut pouvoir :

- les **représenter** . . . . stockage en binaire
- les **manipuler** . . . . arithmétique ($+$, $-$, $*$, $/$)
- le faire de manière automatique.

Contraintes en informatique : espace limité

- largeur mémoire fixe . . . . **taille de chacune des cases**

1 octet	:	256	valeurs possibles seulement
2 octets	:	65 536	valeurs possibles seulement
4 octets	:	$\sim$4 milliards	valeurs possibles (seulement)

- longueur mémoire fixe . . . . nombre de cases disponibles

Choix de codage

Type de nombres étudiés

- entier naturel ($\approx \mathbb{N}$) . . . . codage binaire **non signé**
- entier relatif ($\approx \mathbb{Z}$) . . . . codage binaire **signé**
- réel ($\approx \mathbb{R}$) . . . . codage binaire **flottant**

- Dans chaque cas on utilise :
  - une suite de symboles (*chiffres* 0 ou 1)
  - où la position précise d'un chiffre dans la suite est importante
- arithmétique adaptée à chaque cas
  - calcul doivent être facile à faire avec des opérations simples (portes logiques)
  - En particulier l'addition doit être naturelle

Encoder un nombre décimal positif en UNSIGNED-k

1. Convertir ce nombre décimal en base 2.
2. Si le nombre de bits obtenu dépasse k ; Rejeter le nombre :
   il n'est pas encodable en UNSIGNED-k car trop grand.
3. Si le nombre de bits est plus petit que k ; Ajouter des 0 à gauche jusqu'à avoir k bits.

Codage des entiers naturels : UNSIGNED-k

Encoder un nombre décimal positif en UNSIGNED-k

1. Convertir ce nombre décimal en base 2.
2. Si le nombre de bits obtenu dépasse k ; Rejeter le nombre :
   il n'est pas encodable en UNSIGNED-k car trop grand.
3. Si le nombre de bits est plus petit que k ; Ajouter des 0 à gauche jusqu'à avoir k bits.

Contraintes du codage :

- la taille des mots **largeur des mots** = k
  - mot de k bits
    - entiers de 0 à $2^k - 1$
    - 2^k nombres codables

En pratique

- $k = 8$, $k = 16$, $k = 32$ ou $k = 64$

Encoder un nombre décimal positif en UNSIGNED-k

1. Convertir ce nombre décimal en base 2.
2. Si le nombre de bits obtenu dépasse k ; Rejeter le nombre :
   il n'est pas encodable en UNSIGNED-k car trop grand.
3. Si le nombre de bits est plus petit que k ; Ajouter des 0 à gauche jusqu'à avoir k bits.

Example

Avec un **octet** ($k = 8$)

- on code les entiers de **0** à **255**
- c'est-à-dire **256** nombres
- `00000000` code 0
- `11111111` code 255

La valeur 4 :

- s'écrit en binaire `100` et
- se code `00000100`

- Arithmétique identique en base 2 qu'en base 10 — *positionnelle*
- Table d'addition
  - unités → **XOR**
  - retenue → AND
  - facilement automatisable

ADD	0	1
0		
1		

- Arithmétique identique en base 2 qu'en base 10 *positionnelle*
- Table d'addition
  - unités → **XOR**
  - retenue → AND
  - facilement automatisable

ADD	0	1
0	0 **0**	
1		

UNSIGNED-k : entier non signé (arithmétique)

Addition

- Arithmétique identique en base 2 qu'en base 10 *positionnelle*
- Table d'addition
  - unités → **XOR**
  - retenue → AND
  - facilement automatisable

ADD	0	1
0	**0 0**	**0 1**
1		

- Arithmétique identique en base 2 qu'en base 10 *positionnelle*
- Table d'addition
  - unités → **XOR**
  - retenue → AND
  - facilement automatisable

ADD	0	1
0	0 **0**	0 **1**
1	0 **1**	

UNSIGNED-*k* : entier non signé (arithmétique)

- Arithmétique identique en base 2 qu'en base 10 *positionnelle*
- Table d'addition
  - unités → **XOR**
  - retenue → AND
  - facilement automatisable

ADD	0	1
0	0 **0**	0 **1**
1	0 **1**	1 **0**

- Algorithme
  1. codage binaire des 2 nombres
  2. addition chiffre à chiffre de la droite vers la gauche avec gestion de la retenue
  3. retenue initiale à 0

Example

$10 + 3 = 13$

		0	0	0	0	1	0	1	0	(← 10)
+		0	0	0	0	0	0	1	1	(← 3)
+	0	0	0	0	0	0	1	0	0	Retenue
=		0	0	0	0	1	1	0	1	(→ 13)

- Algorithme
  1. codage binaire des 2 nombres
  2. addition chiffre à chiffre de la droite vers la gauche avec gestion de la retenue
  3. retenue initiale à 0
- Taille de mot fixe (k) $\Rightarrow$ résultat de l'opération pas toujours représentable
  - si $x + y > 2^k - 1$ alors **débordement** (*overflow*)
  - repérable facilement : addition sur le bit de poids fort provoque une retenue à 1
  - les opérations avec débordement sont invalides

Example

$10 + 3 = 13$

		0	0	0	0	1	0	1	0	($\leftarrow$ 10)
+		0	0	0	0	0	0	1	1	($\leftarrow$ 3)
+	0	0	0	0	0	0	1	0	0	Retenue
=		0	0	0	0	1	1	0	1	($\rightarrow$ 13)

UNSIGNED-k : entier non signé (arithmétique)

- Algorithme
  1. codage binaire des 2 nombres
  2. addition chiffre à chiffre de la droite vers la gauche avec gestion de la retenue
  3. retenue initiale à 0
- Taille de mot fixe (k) $\Rightarrow$ résultat de l'opération pas toujours représentable
  - si $x + y > 2^k - 1$ alors **débordement** (*overflow*)
  - repérable facilement : addition sur le bit de poids fort provoque une retenue à 1
  - les opérations avec débordement sont invalides

Example

$128 + 128 = 256$

$$
\begin{array}{ccccccccccc}
 & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (\leftarrow 128) \\
+ & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (\leftarrow 128) \\
+ & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{Retenue} \\
\hline
= & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (\rightarrow 0)
\end{array}
$$

Somme de décalages

- multiplication = somme de décalage à gauche
  - décalage à gauche = multiplication par 2
  - uniquement pour les bits à 1 du multiplicateur
- division = somme de décalage à droite
  - décalage à droite = division par 2
- problème de débordement possible

Example

$13 \times 6 = 78$

$$
\begin{array}{ccccccccccccl}
 & & & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & & (\leftarrow 13) \\
 & & \times & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & & (\leftarrow 6) \\
\cline{3-11}
 & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & \\
+ & & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & & & \\
+ & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & & & & \\
\color{blue}{+} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{1} & \color{blue}{1} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & & \text{Retenue} \\
\hline
 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & & (\rightarrow 78)
\end{array}
$$

Entier signé

Objectifs : coder des entiers relatifs (dans $\mathbb{Z}$, donc positif ou négatif)

- codage en binaire
- représenter le signe et la valeur absolue
- préserver des opérations arithmétiques faciles à faire

Entier signé

Objectifs : coder des entiers relatifs (dans $\mathbb{Z}$, donc positif ou négatif)

- codage en binaire
- représenter le signe et la valeur absolue
- préserver des opérations arithmétiques faciles à faire

Codage naïf

- réserver un bit pour le signe
  - bit de poids fort
  - bit à 0 $\rightarrow$ signe positif
  - bit à 1 $\rightarrow$ signe négatif
- sur 1 octet on code de -127 à $+127$

Entier signé

Objectifs : coder des entiers relatifs (dans $\mathbb{Z}$, donc positif ou négatif)

- codage en binaire
- représenter le signe et la valeur absolue
- préserver des opérations arithmétiques faciles à faire

Codage naïf

- réserver un bit pour le signe
  - bit de poids fort
  - bit à 0 $\rightarrow$ signe positif
  - bit à 1 $\rightarrow$ signe négatif
- sur 1 octet on code de -127 à $+127$

Problèmes

- 0 est codé 2 fois : `10000000` et `00000000`
- opérations arithmétiques difficiles à exécuter

Pour un nombre n codé sur k bits :

- **Complément à 1** (CA1) .... `NOT` bit à bit
  - écart entre le **plus grand** nombre exprimable ($2^k - 1$) et n .... $n + \texttt{CA1}(n) = 2^k - 1$

Example

$$
\begin{array}{ccccccccccl}
 & & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & \leftarrow 42 \;\; = n \\
+ & & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \leftarrow 213 = \texttt{CA1}(42) \\
\hline
= & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \rightarrow 255 = 2^8 - 1
\end{array}
$$

Pour un nombre n codé sur k bits :

- **Complément à 1** (CA1) .... NOT bit à bit
  - écart entre le **plus grand** nombre exprimable ($2^k - 1$) et n .... $n + \texttt{CA1}(n) = 2^k - 1$
- **Complément à 2** (CA2) .... NOT bit à bit puis ajout de 1
  - écart entre le **nombre** de nombres exprimables (2^k) et n .... $n + \texttt{CA2}(n) = 2^k$

Example

$$
\begin{array}{cccccccccl}
 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & \leftarrow 42 \;= n \\
+ & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & \leftarrow 214 = \texttt{CA2}(42) \\
\hline
= 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rightarrow 256 = 2^8
\end{array}
$$

L’encodage complément à deux : CA2

Le complément à deux (CA2) permet de coder une séquence de *k* bits en une autre séquence de *k* bits.

Encoder une suite de bits en CA2

1. Inverser la valeur de chaque bit (0 devient 1 et 1 devient 0)
2. On ajoute 1

Décoder un code CA2

1. Inverser la valeur de chaque bit (0 devient 1 et 1 devient 0)
2. On ajoute 1

Exemple :
Codage des bits 0110 en CA2 :

Valeur initiale	→	0110
Inversion bit à bit	→	1001
Ajout de 1	→	1010
Résultat :	→	1010

Exemple :
Décodage du code 1010 encodé en CA2 :

Valeur initiale	→	1010
Inversion bit à bit	→	0101
Ajout de 1	→	0110
Résultat	→	0110

Codage des entiers signés : SIGNED-k

Encoder un nombre décimal en SIGNED-k

Si ce nombre est positif :

1. Retourner UNSIGNED-k de ce nombre

Si ce nombre est négatif :

1. Prendre la valeur absolue de ce nombre
2. Convertir le résultat en binaire sur k bits
3. Retourner CA2 de la sequence binaire précédemment trouvée

Décoder un code en SIGNED-k

On regarde le premier bit de ce code

Si ce bit est 0 :

1. Convertir le code binaire en décimal

Si ce bit est 1 :

1. Décoder en suivant le décodage de CA2
2. Convertir le code obtenu en décimal
3. Multiplier par -1

SIGNED-k : Entier signé (remarques)

Idée : *calcul modulo*

Représentation limitée à k bits implique arithmétique **modulo** 2^k

Example

Si $k = 4$ on calcule modulo 2^4, c'est-à-dire modulo 16

- sur 4 bits ajouter 15 revient à enlever 1
  - $(1 + 15)\%16 = 0$
  - $(2 + 15)\%16 = 1$
  - $(3 + 15)\%16 = 2$
  - etc.

Idée : *calcul modulo*

Représentation limitée à k bits implique arithmétique **modulo** 2^k

Example

Si $k = 4$ on calcule modulo 2^4, c'est-à-dire modulo 16

- sur 4 bits ajouter 15 revient à enlever 1
  - $(1 + 15)\%16 = 0$
  - $(2 + 15)\%16 = 1$
  - $(3 + 15)\%16 = 2$
  - etc.
- sur 4 bits ajouter 14 revient à enlever 2
  - $(1 + 14)\%16 = 15$
  - $(2 + 14)\%16 = 0$
  - $(3 + 14)\%16 = 1$
  - etc.

SIGNED-k : Entier signé (remarques)

Idée : *calcul modulo*

Représentation limitée à k bits implique arithmétique **modulo** 2^k

Example

Si $k = 4$ on calcule modulo 2^4, c'est-à-dire modulo 16

- sur 4 bits ajouter 15 revient à enlever 1
  - $(1 + 15)\%16 = 0$
  - $(2 + 15)\%16 = 1$
  - $(3 + 15)\%16 = 2$
  - etc.
- sur 4 bits ajouter 14 revient à enlever 2
  - $(1 + 14)\%16 = 15$
  - $(2 + 14)\%16 = 0$
  - $(3 + 14)\%16 = 1$
  - etc.
- le codage fait donc en sorte que
  - $15 \to -1$
  - $14 \to -2$,
  - etc.

SIGNED-k : Entier signé (remarques)

avec unsigned-*4*	$\leftarrow$	code	$\rightarrow$	**avec signed-*4***
0	$\leftarrow$	0000	$\rightarrow$	0
1	$\leftarrow$	0001	$\rightarrow$	1
2	$\leftarrow$	0010	$\rightarrow$	2
3	$\leftarrow$	0011	$\rightarrow$	3
4	$\leftarrow$	0100	$\rightarrow$	4
5	$\leftarrow$	0101	$\rightarrow$	5
6	$\leftarrow$	0110	$\rightarrow$	6
7	$\leftarrow$	0111	$\rightarrow$	7
8	$\leftarrow$	1000	$\rightarrow$	-8
9	$\leftarrow$	1001	$\rightarrow$	-7
10	$\leftarrow$	1010	$\rightarrow$	-6
11	$\leftarrow$	1011	$\rightarrow$	-5
12	$\leftarrow$	1100	$\rightarrow$	-4
13	$\leftarrow$	1101	$\rightarrow$	-3
14	$\leftarrow$	1110	$\rightarrow$	-2
15	$\leftarrow$	1111	$\rightarrow$	-1

SIGNED-k : Entier signé (remarques)

Avantages

- le bit de poids fort représente le signe
- on peut toujours coder 2^k nombres différents
  - tous les entiers entre -2^{k-1} et $2^{k-1} - 1$
  - avec un octet on code de -128 à $+127$
- l'arithmétique est préservée

Entier signé (arithmétique)

Arithmétique simple

Definition

Pour ajouter deux nombres, il suffit de faire la somme bit à bit comme on le fait dans le cas non signé.

La seule différence notable concerne la détection des débordements :

1. uniquement pour les additions de deux nombres aux signes identiques
2. détections
   - *humaine*

     signe du résultat $\neq$ signe (commun) des deux opérandes
   - automatique

     retenue finale $\neq$ retenue propagée sur le bit de poids fort

Entier signé (arithmétique)

		0	0	0	1	0	0	0	0	$\leftarrow$ 16
+		0	0	0	0	1	0	1	1	$\leftarrow$ 11
+	$\boxed{0}$	$\boxed{0}$	0	0	0	0	0	0	0	Retenue
=		0	0	0	1	1	0	1	1	$\rightarrow$ 27

Entier signé (arithmétique)

$$
\begin{array}{cccccccccccc}
 & & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & \leftarrow 16 \\
+ & & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & & \leftarrow 11 \\
+ & \boxed{0} & \boxed{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & \text{Retenue} \\
\hline
= & & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & & \rightarrow 27
\end{array}
$$

$$\underbrace{16 + 11 = 27}_{\textbf{OK}}$$

Entier signé (arithmétique)

		0	0	0	1	0	0	0	0	$\leftarrow 16$
+		0	0	0	0	1	0	1	1	$\leftarrow 11$
+	$\boxed{0}$	$\boxed{0}$	0	0	0	0	0	0	0	Retenue
=		0	0	0	1	1	0	1	1	$\rightarrow 27$

$$\underbrace{16 + 11 = 27}_{\textbf{OK}}$$

		0	1	1	1	1	1	1	1	$\leftarrow 127$
+		0	0	0	0	0	0	0	1	$\leftarrow 1$
+	$\boxed{0}$	$\boxed{1}$	1	1	1	1	1	1	0	Retenue
=		1	0	0	0	0	0	0	0	$\rightarrow -128$

Entier signé (arithmétique)

		0	0	0	1	0	0	0	0	$\leftarrow 16$
+		0	0	0	0	1	0	1	1	$\leftarrow 11$
+	$\boxed{0}$	$\boxed{0}$	0	0	0	0	0	0	0	Retenue
=		0	0	0	1	1	0	1	1	$\rightarrow 27$

$$\underbrace{16 + 11 = 27}_{\textbf{OK}}$$

		0	1	1	1	1	1	1	1	$\leftarrow 127$
+		0	0	0	0	0	0	0	1	$\leftarrow 1$
+	$\boxed{0}$	$\boxed{1}$	1	1	1	1	1	1	0	Retenue
=		1	0	0	0	0	0	0	0	$\rightarrow -128$

$$\underbrace{127 + 1 \neq -128}_{\textbf{Débordement}}$$

Entier signé (arithmétique)

		0	0	0	1	0	0	0	0	$\leftarrow 16$
+		1	1	1	1	1	0	1	1	$\leftarrow -5$
+	1	1	1	1	0	0	0	0	0	Retenue
=		0	0	0	0	1	0	1	1	$\rightarrow 11$

Entier signé (arithmétique)

		0	0	0	1	0	0	0	0	$\leftarrow 16$
+		1	1	1	1	1	0	1	1	$\leftarrow -5$
+	1	1	1	1	0	0	0	0	0	Retenue
=		0	0	0	0	1	0	1	1	$\rightarrow 11$

$$16 - 5 = 11$$

Entier signé (arithmétique)

$$
\begin{array}{cccccccccc}
 & & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \leftarrow 16 \\
+ & & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & \leftarrow -5 \\
\color{blue}{+} & \color{blue}{1} & \color{blue}{1} & \color{blue}{1} & \color{blue}{1} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \text{Retenue} \\
\hline
= & & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & \rightarrow 11
\end{array}
$$

$$16 - 5 = 11$$

$$
\begin{array}{cccccccccc}
 & & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & \leftarrow 3 \\
+ & & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & \leftarrow -5 \\
\color{blue}{+} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{0} & \color{blue}{1} & \color{blue}{1} & \color{blue}{0} & \text{Retenue} \\
\hline
= & & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & \rightarrow -2
\end{array}
$$

Entier signé (arithmétique)

		0	0	0	1	0	0	0	0	$\leftarrow 16$
+		1	1	1	1	1	0	1	1	$\leftarrow -5$
+	1	1	1	1	0	0	0	0	0	Retenue
=		0	0	0	0	1	0	1	1	$\rightarrow 11$

$$16 - 5 = 11$$

		0	0	0	0	0	0	1	1	$\leftarrow 3$
+		1	1	1	1	1	0	1	1	$\leftarrow -5$
+	0	0	0	0	0	0	1	1	0	Retenue
=		1	1	1	1	1	1	1	0	$\rightarrow -2$

$$3 - 5 = -2$$

C'est tout pour aujourd'hui