Représentation de données

Nombres entiers, relatifs et opérations arithmétiques

M. Tellene

Représentation de données

En informatique, les données sont représentées par des nombres

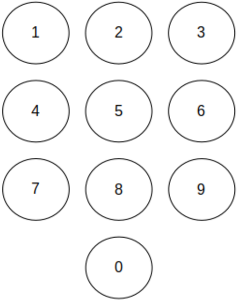
Mais lesquels ?

Représentation des données

La base « occidentale universelle » est la base 10 :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Mais il en existe d'autres !



Représentation de données

Dans l'Antiquité, les Grecs et Romains utilisaient une autre base, la base 12, ou système duodécimal

Représentation de données

En informatique, c'est la base 2, ou **binaire**, qui est utilisée.
Cette base est appelé base 2 car elle utilise 2 symboles :

- 0
- 1

Ces deux symboles sont appelés chiffres binaires ou bits

Représentation de données

Dans la mémoire d'un ordinateur, les chiffres binaires sont regroupés en **octets** [1] : c'est à dire par paquet de 8

Ces octets sont ensuite organisés en paquets de 2, 4 ou 8 appelés **mots machine** [2]

1. appelés bytes en anglais
2. appelés word en anglais

Représentation de données

Ce regroupement de bits en octets ou en mot machine mais aussi de manipuler d'autres données que des 0 ou des 1, comme :

- des nombres réels
- des (approximations de) nombres réels
- des caractères alpha-numériques
- des textes

Représentation de données

L'encodage le plus simple est celui des nombres entiers naturels. Il suffit d'interpréter un octet ou un mot machine comme un entier écrit en base 2

Avant de voir comment les entiers sont écrits en base 2, il faut voir l'écriture « traditionnelle », la base 10

Représentation de données

Un nombre entier en base 10 est une séquence de chiffres entre 0 et 9

Pour calculer la valeur d'une séquence c_{k-1}, c_{k-2}, ..., c_1, c_0 de k chiffres, on affecte à chaque chiffres c_i le poids 10^i, et on calcule la somme des termes $c_i \times 10^i$

Cela peut être résumé avec la formule suivante :

$$\sum_{i=n-1}^{0} c_i \times 10^i$$

Représentation de données

Exemple avec 61027 :

Séquence	6	1	0	2	7
Position					
Poids					

Représentation de données

Exemple avec 61027 :

Séquence	6	1	0	2	7
Position	4	3	2	1	0
Poids					

Représentation de données

Exemple avec 61027 :

Séquence	6	1	0	2	7
Position	4	3	2	1	0
Poids	10^4	10^3	10^2	10^1	10^0

Représentation de données

Exemple avec 61027 :

Séquence	6	1	0	2	7
Position	4	3	2	1	0
Poids	10^4	10^3	10^2	10^1	10^0

La valeur de la séquence est l'entier N calculé de la manière suivante :

$$N = 6 \times 10^4 + 1 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$$

Représentation de données

Exemple avec 61027 :

Séquence	6	1	0	2	7
Position	4	3	2	1	0
Poids	10^4	10^3	10^2	10^1	10^0

La valeur de la séquence est l'entier N calculé de la manière suivante :

$$N = 6 \times 10^4 + 1 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$$
$$N = 60000 + 1000 + 0 + 20 + 7$$
$$N = 61027$$

Représentation de données

Maintenant passons à la base 2

De manière similaire à l'encodage en base 10, une séquence de chiffres binaires peut s'interpréter comme un nombre écrit en base 2

Dans cette base les chiffres (0 ou 1) d'une séquence sont associés à un poids 2^i d'une puissance de 2 qui dépend toujours de la position i des chiffres dans la séquence

De manière simplifiée :

$\sum_{i=n-1}^{0} b_i \times 2^i$ où b_i correspond au bit à la position i

Représentation de données

Exemple avec `01001101` :

Séquence	0	1	0	0	1	1	0	1
Position								
Poids								

Représentation de données

Exemple avec `01001101` :

Séquence	0	1	0	0	1	1	0	1
Position	7	6	5	4	3	2	1	0
Poids								

Représentation de données

Exemple avec `01001101` :

Séquence	0	1	0	0	1	1	0	1
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Représentation de données

Exemple avec `01001101` :

Séquence	0	1	0	0	1	1	0	1
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Vu comme un entier de 8 bits, cet octet correspond au nombre N calculé de la manière suivante :

$$N = 0\times2^7+1\times2^6+0\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$$

Représentation de données

Exemple avec `01001101` :

Séquence	0	1	0	0	1	1	0	1
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Vu comme un entier de 8 bits, cet octet correspond au nombre N calculé de la manière suivante :

$$N = 0\times2^7+1\times2^6+0\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$$
$$N = 0 + 64 + 0 + 0 + 8 + 4 + 0 + 1$$
$$N = 77$$

Représentation de données

Dans une séquence b_{n-1}, b_{n-2}, ..., b_1, b_0 de n chiffres :

- b_{n-1} est le **bit de poids fort**
- b_0 est le **bit de poids faible**

Représentation de données

Mais comment passer du décimal au binaire ?

Deux méthodes possibles :

- En utilisant des soustractions
- En utilisant des divisions euclidiennes

Représentation de données

Méthode utilisant la soustraction :

Combien vaut 143 en binaire ?

Représentation de données

Méthode utilisant la soustraction :

Combien vaut 143 en binaire ?

On doit se poser une question avant de commencer, ce nombre peut-il être contenu sur un octet ?

Quel est le plus grand entier faisable sur un octet ?

Représentation de données

Méthode utilisant la soustraction :

Combien vaut 143 en binaire ?

On doit se poser une question avant de commencer, ce nombre peut-il être contenu sur un octet ?

Quel est le plus grand entier faisable sur un octet ?

11111111 est le plus grand entier encodable sur un octet

Représentation de données

Combien vaut `11111111` ?

Représentation de données

Combien vaut `11111111` ?

Séquence	1	1	1	1	1	1	1	1
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Représentation de données

Combien vaut 11111111 ?

Séquence	1	1	1	1	1	1	1	1
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$N = 1\times2^7+1\times2^6+1\times2^5+1\times2^4+1\times2^3+1\times2^2+1\times2^1+1\times2^0$
$N = 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1$
$N = 255$

Représentation de données

D'une manière générale, le plus grand entier naturel réalisable sur n bits est égal à :

$$2^n - 1$$

- sur 8 bits $\rightarrow 2^8 - 1 \rightarrow 255$
- sur 16 bits $\rightarrow 2^{16} - 1 \rightarrow 65535$
- ...

Représentation de données

Revenons à notre problème : combien vaut 143 en binaire ?

143 est-il représentable sur 1 octet ?

Représentation de données

Revenons à notre problème : combien vaut 143 en binaire ?

143 est-il représentable sur 1 octet ?

Oui car $143 < 255$

Représentation de données

Étant donné que 143 est représentable avec 1 octet, nous pouvons poser ceci :

Séquence								
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Mais comment l'utiliser ?

Représentation de données

Étant donné que 143 est représentable avec 1 octet, nous pouvons poser ceci :

Séquence								
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Mais comment l'utiliser ?

On va commencer par 2^7 et on se demande si 2^7 rentre dans 143

Représentation de données

Étant donné que 143 est représentable avec 1 octet, nous pouvons poser ceci :

Séquence								
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Mais comment l'utiliser ?

On va commencer par 2^7 et on se demande si 2^7 rentre dans 143

La réponse est oui, on fait donc 2 choses :

- On met un « 1 » dans le tableau à la position 7
- On enlève 2^7 à 143

Représentation de données

Voilà ce que nous avons :

Séquence	1							
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Reste à calculer :
$143 - 2^7 \rightarrow 143 - 128 = 15$

Représentation de données

Voilà ce que nous avons :

Séquence	1							
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Reste à calculer :
$143 - 2^7 \rightarrow 143 - 128 = 15$

Que fait-on après ? On continue avec 2^6 et 15...

Représentation de données

Correction : comment représenter 143 en binaire ?

Représentation de données

Correction : comment représenter 143 en binaire ?

$143 > 2^7$? Oui donc on fait $143 - 2^7 = 15$

$15 > 2^6$? Non donc on garde 15

$15 > 2^5$? Non donc on garde 15

$15 > 2^4$? Non donc on garde 15

$15 > 2^3$? Oui donc on fait $15 - 2^3 = 7$

$7 > 2^2$? Oui donc on fait $7 - 2^2 = 3$

$3 > 2^1$? Oui donc on fait $3 - 2^1 = 1$

$1 > 2^0$? Oui donc on fait $1 - 2^0 = 0$

Représentation de données

On remplit enfin le tableau :

Séquence	1							
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Représentation de données

On remplit enfin le tableau :

Séquence	1							
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Séquence	1	0	0	0	1	1	1	1
Position	7	6	5	4	3	2	1	0
Poids	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

143 donne donc 10001111 en binaire

Représentation de données

Méthode utilisant la division euclidienne :

Combien vaut 19 en binaire ?

On procède par division euclidienne par 2

3. Il est possible d'écrire 00010011 pour mettre 19 sous forme d'octet

Représentation de données

Méthode utilisant la division euclidienne :

Combien vaut 19 en binaire ?

On procède par division euclidienne par 2

$$\begin{array}{r|l} 19 & 2 \\ \cline{2-2} \boxed{\mathit{1}} & 9 \\ & \end{array} \qquad \begin{array}{r|l} 9 & 2 \\ \cline{2-2} \boxed{\mathit{1}} & 4 \\ & \end{array} \qquad \begin{array}{r|l} 4 & 2 \\ \cline{2-2} \boxed{\mathit{0}} & 2 \\ & \end{array} \qquad \begin{array}{r|l} 2 & 2 \\ \cline{2-2} \boxed{\mathit{0}} & 2 \\ & \end{array} \qquad \begin{array}{r|l} 1 & 2 \\ \cline{2-2} \boxed{\mathit{1}} & 0 \\ & \end{array}$$

3. Il est possible d'écrire 00010011 pour mettre 19 sous forme d'octet

Représentation de données

Méthode utilisant la division euclidienne :

Combien vaut 19 en binaire ?

On procède par division euclidienne par 2

$$\begin{array}{c|c} 19 & 2 \\ \cline{2-2} \boxed{\mathit{1}} & 9 \\ & \end{array} \qquad \begin{array}{c|c} 9 & 2 \\ \cline{2-2} \boxed{\mathit{1}} & 4 \\ & \end{array} \qquad \begin{array}{c|c} 4 & 2 \\ \cline{2-2} \boxed{\mathit{0}} & 2 \\ & \end{array} \qquad \begin{array}{c|c} 2 & 2 \\ \cline{2-2} \boxed{\mathit{0}} & 2 \\ & \end{array} \qquad \begin{array}{c|c} 1 & 2 \\ \cline{2-2} \boxed{\mathit{1}} & 0 \\ & \end{array}$$

On retrouve l'écriture binaire de 19 en lisant **les restes dans le sens inverse**

3. Il est possible d'écrire 00010011 pour mettre 19 sous forme d'octet

Représentation de données

Méthode utilisant la division euclidienne :

Combien vaut 19 en binaire ?

On procède par division euclidienne par 2

$$
\begin{array}{c|c} 19 & 2 \\ \hline \boxed{\mathit{1}} & 9 \end{array}
\qquad
\begin{array}{c|c} 9 & 2 \\ \hline \boxed{\mathit{1}} & 4 \end{array}
\qquad
\begin{array}{c|c} 4 & 2 \\ \hline \boxed{\mathit{0}} & 2 \end{array}
\qquad
\begin{array}{c|c} 2 & 2 \\ \hline \boxed{\mathit{0}} & 2 \end{array}
\qquad
\begin{array}{c|c} 1 & 2 \\ \hline \boxed{\mathit{1}} & 0 \end{array}
$$

On retrouve l'écriture binaire de 19 en lisant **les restes dans le sens inverse**

19 donne donc 10011[3] en binaire

3. Il est possible d'écrire 00010011 pour mettre 19 sous forme d'octet

Représentation de données

Les deux méthodes sont équivalentes mais la division euclidienne peut être très fastidieuse sur les grands nombres

Représentation de données

Petite question : 1001 est écrit en base 10 ou en base 2 ?

Représentation de données

Petite question : 1001 est écrit en base 10 ou en base 2 ?

Pour ne pas se mélanger avec les différentes bases, on ajoute un indice dans la notation

Représentation de données

Petite question : 1001 est écrit en base 10 ou en base 2 ?

Pour ne pas se mélanger avec les différentes bases, on ajoute un indice dans la notation

- $(1001)_2$ est en base 2
- $(1001)_{10}$ est en base 10

Représentation de données

Nous avons vu qu'il existait le binaire (base 2) et le décimal (base 10), mais il existe une autre base très utilisée en informatique : **l'hexadécimal**

Mais pourquoi créer un autre système de représentation de données ?

Représentation de données

Petit rappel :

Les circuits mémoires d'un ordinateurs sont groupés par octets

Une architecture 32 bits est constituée de quatre octets pouvant représenter jusqu'à **4 294 967 296 valeurs distinctes**

A vous de faire le calcul pour 64 bits...

Représentation de données

Utiliser la base 2 devient vite très **fastidieux** en raison du nombre de positions nécessaires pour l'écriture d'un (grand) entier dans cette base

Il nous faut donc une autre base qui permette de simplifier cette écriture tout en ayant une correspondance rapide avec la base 2 :

$\rightarrow$ le système hexadécimal, de base 16

Représentation de données

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Représentation de données

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9						

Représentation de données

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Représentation de données

Étant donné que l'écriture hexadécimale comporte 16 chiffres, combien de bits sont nécessaires pour représenter ces 16 chiffres différents ?

Représentation de données

Étant donné que l'écriture hexadécimale comporte 16 chiffres, combien de bits sont nécessaires pour représenter ces 16 chiffres différents ?

4, car $2^4 = 16$

Par conséquent :

1 octet = 8 bits = 2 symboles hexadécimaux

Représentation de données

Étant donné que l'écriture hexadécimale comporte 16 chiffres, combien de bits sont nécessaires pour représenter ces 16 chiffres différents ?

4, car $2^4 = 16$

Par conséquent :

1 octet = 8 bits = 2 symboles hexadécimaux

Enfin pour pouvoir différencier l'écriture hexadécimale des autres écritures, on ajoute un 16 en indice

Exemple : $(A52B)_{16}$

Représentation de données

De manière similaire aux bases 2 et 10, on peut représenter les séquences de chiffres hexadécimaux en colonnes en indiquant position et poids des chiffres

Exemple : $(2A0D)_{16}$

Représentation de données

De manière similaire aux bases 2 et 10, on peut représenter les séquences de chiffres hexadécimaux en colonnes en indiquant position et poids des chiffres

Exemple : $(2A0D)_{16}$

Séquence	2	A	0	D
Position	3	2	1	0
Poids	16^3	16^2	16^1	16^0

Quelle est la décomposition de ce nombre hexadécimal ?

Représentation de données

De manière similaire aux bases 2 et 10, on peut représenter les séquences de chiffres hexadécimaux en colonnes en indiquant position et poids des chiffres

Exemple : $(2A0D)_{16}$

Séquence	2	A	0	D
Position	3	2	1	0
Poids	16^3	16^2	16^1	16^0

Quelle est la décomposition de ce nombre hexadécimal ?

$$(2A0D)_{16} = 2 \times 16^3 + A \times 16^2 + 0 \times 16^1 + D \times 16^0$$
$$(2A0D)_{16} = 8192 + 2560 + 0 + 13$$
$$(2A0D)_{16} = 10765$$

Représentation de données

Cette manière de passer de l'hexadécimal au décimal n'est pas aisée : il est compliqué de faire 16^x de tête.

Il faut donc trouver un moyen plus simple pour effectuer cette conversion

Représentation de données

Cette manière de passer de l'hexadécimal au décimal n'est pas aisée : il est compliqué de faire 16^x de tête.

Il faut donc trouver un moyen plus simple pour effectuer cette conversion

$$\rightarrow \text{hexadécimal} \rightarrow \text{binaire} \rightarrow \text{décimal}$$

Représentation de données

La base 16 est souvent utilisée pour simplifier l'écriture de nombres binaires

En effet, on peut facilement passer d'un nombre en base 2 à un nombre en base 16 en regroupant les chiffres binaires par 4 [4]

4. cf 1 octet = 8 bits = 2 chiffres hexadécimaux

Représentation de données

La base 16 est souvent utilisée pour simplifier l'écriture de nombres binaires

En effet, on peut facilement passer d'un nombre en base 2 à un nombre en base 16 en regroupant les chiffres binaires par 4 [4]

Par exemple, la séquence de bits $(1010010111110011)_2$ correspond au nombre hexadécimal $(A5F3)_{16}$:

4. cf 1 octet $=$ 8 bits $=$ 2 chiffres hexadécimaux

Représentation de données

La base 16 est souvent utilisée pour simplifier l'écriture de nombres binaires

En effet, on peut facilement passer d'un nombre en base 2 à un nombre en base 16 en regroupant les chiffres binaires par 4 [4]

Par exemple, la séquence de bits $(1010010111110011)_2$ correspond au nombre hexadécimal $(A5F3)_{16}$:

$$\underbrace{1010}_{A}\underbrace{0101}_{5}\underbrace{1111}_{F}\underbrace{0011}_{3}$$

4. cf 1 octet $=$ 8 bits $=$ 2 chiffres hexadécimaux

Représentation de données

On note que la transformation inverse est aussi très simple puisqu'il suffit de traduire chaque chiffre hexadécimal avec 4 bits selon le tableau de correspondance suivant

Chiffre hexadécimal	Bits
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Chiffre hexadécimal	Bits
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Représentation de données

Il est possible de faire des opérations arithmétiques sur les nombres en représentation binaire

Nous allons voir l'addition et la multiplication

Représentation de données

Il est possible de faire des opérations arithmétiques sur les nombres en représentation binaire

Nous allons voir l'addition et la multiplication

L'addition en base 2 fonctionne comme l'addition que vous connaissez, sauf que $(1)_2 + (1)_2 = (10)_2$, en fait 0 avec une retenue de 1

Représentation de données

Il est possible de faire des opérations arithmétiques sur les nombres en représentation binaire

Nous allons voir l'addition et la multiplication

L'addition en base 2 fonctionne comme l'addition que vous connaissez, sauf que $(1)_2 + (1)_2 = (10)_2$, en fait 0 avec une retenue de 1

$$\begin{array}{r} 1 \\ + \quad 1 \\ \hline \end{array} \qquad \begin{array}{rr} & ^{1}\ 1 \\ + & 1 \\ \hline & 0 \end{array} \qquad \begin{array}{rr} & ^{1}\ 1 \\ + & 1 \\ \hline 1 & 0 \end{array}$$

Représentation de données

Exemple avec des octets : $(21)_{10} + (25)_{10}$

Représentation de données

Exemple avec des octets : $(21)_{10} + (25)_{10}$

$$\begin{array}{ccccccccc} & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ \hline \end{array}$$

Représentation de données

Exemple avec des octets : $(21)_{10} + (25)_{10}$

$$\begin{array}{ccccccccc} & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ \hline \end{array}$$

$$\begin{array}{ccccccccc} & & & {}^{1} & & & & {}^{1} & \\ & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ \hline & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{array}$$

Il est important de noter que si on ne dispose que de 8 bits, on ne pourra stocker le résultat d'une addition supérieure à 255

Représentation de données

Passons à la multiplication

La multiplication binaire s'effectue selon le principe de la multiplication décimale, on multiplie donc le multiplicande par chacun des bits du multiplicateur

On décale les résultats intermédiaires obtenus et on effectue ensuite l'addition de ses résultats partiels

Représentation de données

Passons à la multiplication

La multiplication binaire s'effectue selon le principe de la multiplication décimale, on multiplie donc le multiplicande par chacun des bits du multiplicateur

On décale les résultats intermédiaires obtenus et on effectue ensuite l'addition de ses résultats partiels

$$\begin{array}{ccccc} & 1 & 0 & 1 & 1 \\ \times & & & 1 & 1 \\ \hline \end{array}$$

Représentation de données

Passons à la multiplication

La multiplication binaire s'effectue selon le principe de la multiplication décimale, on multiplie donc le multiplicande par chacun des bits du multiplicateur

On décale les résultats intermédiaires obtenus et on effectue ensuite l'addition de ses résultats partiels

$$
\begin{array}{cccccc}
 & & 1 & 0 & 1 & 1 \\
\times & & & & 1 & 1 \\
\hline
\end{array}
\qquad\qquad
\begin{array}{cccccc}
 & & 1 & 0 & 1 & 1 \\
\times & & & & 1 & 1 \\
\hline
 & & {\scriptstyle 1} & {\scriptstyle 1} & & \\
 & {\scriptstyle 1} & 1 & 0 & 1 & 1 \\
 & 1 & 0 & 1 & 1 & \text{x} \\
\hline
1 & 0 & 0 & 0 & 0 & 1
\end{array}
$$

Représentation de données

Exemple avec des nombres plus grands : $(21)_{10} \times (25)_{10}$

Représentation de données

Exemple avec des nombres plus grands : $(21)_{10} \times (25)_{10}$

$$
\begin{array}{cccccccccc}
 & & & & & 1 & 0 & 1 & 0 & 1 \\
 & & & & \times & 1 & 1 & 0 & 0 & 1 \\
\hline
 & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & 1 & 0 & 1 & 0 & 1 \\
 & & & & 0 & 0 & 0 & 0 & 0 & \mathrm{x} \\
 & & & 0 & 0 & 0 & 0 & 0 & \mathrm{x} & \mathrm{x} \\
 & & 1 & 0 & 1 & 0 & 1 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\
 & 1 & 0 & 1 & 0 & 1 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1
\end{array}
$$

Représentation de données

L'encodage des entier relatifs est plus délicat.

Représentation de données

L'encodage des entier relatifs est plus délicat.

L'idée principale est d'utiliser le bit de poids fort d'un mot mémoire pour représenter le signe d'un entier :

- 0 indique un entier positif
- 1 indique un entier négatif

Exemples :

- $(0011)_2$
- $(1101)_2$

Représentation de données

L'encodage des entier relatifs est plus délicat

L'idée principale est d'utiliser le bit de poids fort d'un mot mémoire pour représenter le signe d'un entier :

- 0 indique un entier positif
- 1 indique un entier négatif

Exemples :

- $(0011)_2 \rightarrow$ positif (3 en l'occurrence)
- $(1101)_2 \rightarrow$ négatif (-5 en l'occurrence)

Représentation de données

Avec cet encodage, un mot binaire de n bits permet de représenter les entiers relatifs dans l'intervalle
$-(2^{n-1} - 1)$ à $2^{n-1} - 1$

- sur 4 bits, on peut représenter tous les entiers entre -7 et 7
- sur 8 bits, on peut représenter tous les entiers entre -127 et 127

Représentation de données

Malheureusement, cet encodage simpliste souffre de deux problèmes :

- le nombre 0 possède 2 représentations
- il complique les opérations arithmétiques

Représentation de données

Premier problème, « 0 » possède 2 représentations

Représentation de données

Premier problème, « 0 » possède 2 représentations

Sur 4 bits, $(0000)_2$ et $(1000)_2$ représentent tous les deux 0

$\rightarrow$ un 0 « positif »

$\rightarrow$ un 1 « négatif »

Représentation de données

Second problème, la complexification des opérations arithmétiques

Par exemple, pour additionner deux entiers relatifs, il faut faire une addition ou une soustraction selon que les entiers sont du même signe ou non.

Ainsi, l'addition de $(5)_{10}$ et de $(-5)_2$ donnera :

Représentation de données

Second problème, la complexification des opérations arithmétiques

Par exemple, pour additionner deux entiers relatifs, il faut faire une addition ou une soustraction selon que les entiers sont du même signe ou non.

Ainsi, l'addition de $(5)_{10}$ et de $(-5)_2$ donnera :

$$\begin{array}{ccccc} & \overset{1}{0} & 1 & \overset{1}{0} & 1 \\ + & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 \end{array}$$

Avec cet encodage $(5)_{10} + (-5)_{10} = (-2)_{10}$

Représentation de données

La solution la plus commune pour résoudre ces problèmes est d'utiliser l'encodage dit par **complément à 2**

Dans cet encodage :

- le bit de poids fort est utilisé pour représenter le signe des entiers
- la représentation des nombres positifs est inchangée
- mais celle des négatifs utilise le complément à 2

Représentation de données

La théorie : le complément à 2 d'un mot binaire *m* sur *n* bits s'obtient en inversant la valeur des *n* bits de *m* puis en ajoutant 1 au mot binaire obtenu (sans tenir compte de la retenue finale)

Représentation de données

La théorie : le complément à 2 d'un mot binaire m sur n bits s'obtient en inversant la valeur des n bits de m puis en ajoutant 1 au mot binaire obtenu (sans tenir compte de la retenue finale)

La pratique : quel est le complément à 2 de $(011)_2$

Inversion du mot binaire : $(011)_2 \rightarrow (100)_2$

Ajout de 1 au mot obtenu :

$$\begin{array}{cccc} & 1 & 0 & 0 \\ + & & & 1 \\ \hline & 1 & 0 & 1 \end{array}$$

Représentation de données

Avec la méthode du complément à 2, un mot binaire de n bits permet de représenter les entiers relatifs dans l'intervalle :

$$[-2^{n-1} ; 2^{n-1} - 1]$$

Représentation de données

Comment être sûr que l'on a calculé le bon complément à 2 ?

Représentation de données

Comment être sûr que l'on a calculé le bon complément à 2 ?

Prenons l'exemple de $(-4)_{10}$:

Représentation de données

Comment être sûr que l'on a calculé le bon complément à 2 ?

Prenons l'exemple de $(-4)_{10}$:

$(-4)_{10} \rightarrow (4)_{10} = (0100)_2$

Inversion du mot binaire : $(0100)_2 \rightarrow (1011)_2$

Ajout de 1 au mot obtenu :

$$\begin{array}{ccccc} & 1 & 0 & 1 & 1 \\ + & & & & 1 \\ \hline & 1 & 1 & 0 & 0 \end{array}$$

Représentation de données

On a donc obtenu $(-4)_{10} = (1100)_2$

Représentation de données

On a donc obtenu $(-4)_{10} = (1100)_2$

Le bit de poids fort, en plus d'être utilisé pour représenter le signe des entiers, est interprété comme ayant la valeur -2^{n-1} pour un entier écrit sur n bits

La séquence de bits $b_{n-1}b_{n-2}b_1b_0$ est interprété comme :

$$N = -b_{n-1} \times 2^{n-1} + \sum_{0 \leqslant i < n-1} b_i \times 2^i$$

Représentation de données

On a donc obtenu $(-4)_{10} = (1100)_2$

Le bit de poids fort, en plus d'être utilisé pour représenter le signe des entiers, est interprété comme ayant la valeur -2^{n-1} pour un entier écrit sur n bits

La séquence de bits $b_{n-1} b_{n-2} b_1 b_0$ est interprété comme :

$$N = -b_{n-1} \times 2^{n-1} + \sum_{0 \leqslant i < n-1} b_i \times 2^i$$

$(-4)_{10} = -1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$

$(-4)_{10} = -8 + 4 + 0 + 0$

Représentation de données

Avec le complément à 2, l'addition de deux mots binaire m_1 et m_2 représentant des entiers positifs ou négatifs s'effectue comme l'addition binaire $m_1 + m_2$, sans se soucier du signe des entiers codés par m_1 et m_2

Reprenons l'addition $(5)_{10} + (-5)_{10}$

Représentation de données

Avec le complément à 2, l'addition de deux mots binaire m_1 et m_2 représentant des entiers positifs ou négatifs s'effectue comme l'addition binaire $m_1 + m_2$, sans se soucier du signe des entiers codés par m_1 et m_2

Reprenons l'addition $(5)_{10} + (-5)_{10}$

- $(5)_{10} \rightarrow (0101)_2$
- $(-5)_{10} \rightarrow (1011)_2$

Représentation de données

Avec le complément à 2, l'addition de deux mots binaire m_1 et m_2 représentant des entiers positifs ou négatifs s'effectue comme l'addition binaire $m_1 + m_2$, sans se soucier du signe des entiers codés par m_1 et m_2

Reprenons l'addition $(5)_{10} + (-5)_{10}$

- $(5)_{10} \rightarrow (0101)_2$
- $(-5)_{10} \rightarrow (1011)_2$

$$\begin{array}{ccccc} & & {\scriptstyle 1} & {\scriptstyle 1} & \\ & 1 & 1 & 0 & 1 \\ + & 1 & 0 & 1 & 1 \\ \hline & 0 & 0 & 0 & 0 \end{array}$$

Représentation de données

Le complément à 2 règle également le problème de la double représentation du zéro.

En effet, pour le montrer il suffit de faire la démonstration suivante :

$(0)_{10} = (00000000)_2$

Inversion du mot binaire : $(00000000)_2 \rightarrow (11111111)_2$

Ajout de 1 au mot obtenu :

$$\begin{array}{ccccccccc} & {}^{1} & {}^{1} & {}^{1} & {}^{1} & {}^{1} & {}^{1} & {}^{1} & \\ & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ + & & & & & & & & 1 \\ \hline & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$