Principes de fonctionnement des machines binaires

2019/2020

Pierluigi Crescenzi
Université de Paris, IRIF

- Tests et examens
  - CC : résultat des tests en TD / TP (semaine 4 et 10)
  - E0 : partiel (**samedi 26 octobre**)
  - E1 : examen mi décembre
  - E2 : examen fin juin
- Notes finales
  - Note session 1 : 25% CC + 25% E0 + 50% E1
  - Note session 2 : max(E2, 33% CC + 67% E2)
- Rappel
  - Pas de note $\Rightarrow$ pas de moyenne $\Rightarrow$ pas de semestre
- Site web
  - `moodlesupd.script.univ-paris-diderot.fr`

- Numération et arithmétique
- **Numération et arithmétique en machine**
- Numérisation et codage (texte, images)
- Compression, cryptographie, contrôle d'erreur
- Logique et calcul propositionnel
- Circuits numériques

- (Si) on manipule rarement les très très très grands nombres
- (Si) on manipule rarement les nombres avec une grande précision

- (Si) on manipule rarement les très très très grands nombres
- (Si) on manipule rarement les nombres avec une grande précision
  - Choix de fixer la taille des représentations

- (Si) on manipule rarement les très très très grands nombres
- (Si) on manipule rarement les nombres avec une grande précision
  - Choix de fixer la taille des représentations
    - Architectures communes 32 ou 64 **bits** (**bi**nary digi**t**) : arithmétique sur des nombres représentés sur 32 ou 64 bits

- (Si) on manipule rarement les très très très grands nombres
- (Si) on manipule rarement les nombres avec une grande précision
  - Choix de fixer la taille des représentations
    - Architectures communes 32 ou 64 **bits** (**bi**nary digi**t**) : arithmétique sur des nombres représentés sur 32 ou 64 bits
      - $2^{32} = 4294967296 \approx 4,3 \times 10^9$ et $2^{64} \approx 18,4 \times 10^{18}$

- (Si) on manipule rarement les très très très grands nombres
- (Si) on manipule rarement les nombres avec une grande précision
  - Choix de fixer la taille des représentations
    - Architectures communes 32 ou 64 **bits** (**bi**nary digi**t**) : arithmétique sur des nombres représentés sur 32 ou 64 bits
      - $2^{32} = 4294967296 \approx 4,3 \times 10^9$ et $2^{64} \approx 18,4 \times 10^{18}$
  - Un très grand nombre de choix pour représenter des entiers
    - $2^{32}!$ ou $2^{64}!$

- **Un choix parmi les 2^{32}! choix pour représenter les entiers de 0 à $2^{32}-1$**
  - Mots sur $\{0,1\}$ dans l'ordre lexicographique

- **Un choix parmi les 2^{32}! choix pour représenter les entiers de 0 à $2^{32}-1$**
  - Mots sur $\{0,1\}$ dans l'ordre lexicographique

	base 2	base 10
00000000 00000000 00000000 00000000	0	0

- **Un choix parmi les 2^{32}! choix pour représenter les entiers de 0 à $2^{32}-1$**
  - Mots sur $\{0,1\}$ dans l'ordre lexicographique

	base 2	base 10
00000000 00000000 00000000 00000000	0	0
00000000 00000000 00000000 00000001	1	1

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de 0 à $2^{32}-1$**
  - Mots sur $\{0,1\}$ dans l'ordre lexicographique

	base 2	base 10
00000000 00000000 00000000 00000000	0	0
00000000 00000000 00000000 00000001	1	1
00000000 00000000 00000000 00000010	10	2

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de 0 à $2^{32}-1$**
  - Mots sur $\{0,1\}$ dans l'ordre lexicographique

	base 2	base 10
00000000 00000000 00000000 00000000	0	0
00000000 00000000 00000000 00000001	1	1
00000000 00000000 00000000 00000010	10	2
00000000 00000000 00000000 00000011	11	3

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de 0 à $2^{32}-1$**
  - Mots sur $\{0,1\}$ dans l'ordre lexicographique

	base 2	base 10
00000000 00000000 00000000 00000000	0	0
00000000 00000000 00000000 00000001	1	1
00000000 00000000 00000000 00000010	10	2
00000000 00000000 00000000 00000011	11	3
...	...	...
11111111 11111111 11111111 11111101		4294967293
11111111 11111111 11111111 11111110		4294967294
11111111 11111111 11111111 11111111		4294967295

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de 0 à $2^{32}-1$**
  - Mots sur $\{0,1\}$ dans l'ordre lexicographique

représentation non signé

	base 2	base 10
00000000 00000000 00000000 00000000	0	0
00000000 00000000 00000000 00000001	1	1
00000000 00000000 00000000 00000010	10	2
00000000 00000000 00000000 00000011	11	3
...	...	...
11111111 11111111 11111111 11111101		4294967293
11111111 11111111 11111111 11111110		4294967294
11111111 11111111 11111111 11111111		4294967295

- Addition de deux nombres entiers non signés
  - Si la retenue à gauche est 1, un *débordement* arithmétique a eu lieu lors de l'addition

- Addition de deux nombres entiers non signés
  - Si la retenue à gauche est 1, un *débordement* arithmétique a eu lieu lors de l'addition
  - Exemple : 109+221 (8 bits, entiers de 0 à 251)

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

- Addition de deux nombres entiers non signés
  - Si la retenue à gauche est 1, un *débordement* arithmétique a eu lieu lors de l'addition
  - Exemple : 109+221 (8 bits, entiers de 0 à 251)

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1} \\ 0110\,1101 \\ +\ \ 1101\,1101 \\ \hline 1100\,1010 \end{array}$$

  - Débordement est facile à detecter

- **Un choix parmi les 2^{32}! choix pour représenter les entiers de $-2^{31}+1$ à $2^{31}-1$**
  - Mots sur $\{0,1\}$

- **Un choix parmi les 2^{32}! choix pour représenter les entiers de $-2^{31}+1$ à $2^{31}-1$**
  - Mots sur $\{0,1\}$

	base 10
0 0000000 00000000 00000000 00000000	0
0 0000000 00000000 00000000 00000001	1
...	
0 1111111 11111111 11111111 11111101	2147483645
0 1111111 11111111 11111111 11111110	2147483646
0 1111111 11111111 11111111 11111111	2147483647

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de $-2^{31}+1$ à $2^{31}-1$**
  - Mots sur $\{0,1\}$

	base 10
1 1111111 11111111 11111111 11111111	-2147483647
1 1111111 11111111 11111111 11111110	-2147483646
1 1111111 11111111 11111111 11111101	-2147483645
...	
1 0000000 00000000 00000000 00000001	-1
1 0000000 00000000 00000000 00000000	-0
0 0000000 00000000 00000000 00000000	0
0 0000000 00000000 00000000 00000001	1
...	
0 1111111 11111111 11111111 11111101	2147483645
0 1111111 11111111 11111111 11111110	2147483646
0 1111111 11111111 11111111 11111111	2147483647

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de $-2^{31}+1$ à $2^{31}-1$**
  - Mots sur $\{0,1\}$

bit de signe		base 10
1	1111111 11111111 11111111 11111111	-2147483647
1	1111111 11111111 11111111 11111110	-2147483646
1	1111111 11111111 11111111 11111101	-2147483645
	…	
1	0000000 00000000 00000000 00000001	-1
1	0000000 00000000 00000000 00000000	-0
0	0000000 00000000 00000000 00000000	0
0	0000000 00000000 00000000 00000001	1
	…	
0	1111111 11111111 11111111 11111101	2147483645
0	1111111 11111111 11111111 11111110	2147483646
0	1111111 11111111 11111111 11111111	2147483647

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de $-2^{31}+1$ à $2^{31}-1$**
  - Mots sur $\{0,1\}$

bit de signe		base 10
1	1111111 11111111 11111111 11111111	-2147483647
1	1111111 11111111 11111111 11111110	-2147483646
1	1111111 11111111 11111111 11111101	-2147483645
	...	
1	0000000 00000000 00000000 00000001	-1
1	0000000 00000000 00000000 00000000	-0
0	0000000 00000000 00000000 00000000	0
0	0[illegible]	[illegible]
0	1[illegible]	[illegible]645
0	1[illegible]	[illegible]646
0	1111111 11111111 11111111 11111111	2147483647

Zero possède deux représentations

- Représentation signe / valeur absolue
  - Difficultés au niveau des opérations arithmétiques

- Représentation signe / valeur absolue
  - Difficultés au niveau des opérations arithmétiques
    - Exemple : 16+24 = ? et 16 + (-24) = ?

- Représentation signe / valeur absolue
  - Difficultés au niveau des opérations arithmétiques
    - Exemple : 16+24 = ? et 16 + (-24) = ?

16+24

```
  0 0010 000
    0001 0000
+   0001 1000
-------------
    0010 1000
```

40

- Représentation signe / valeur absolue
  - Difficultés au niveau des opérations arithmétiques
    - Exemple : 16+24 = ? et 16 + (-24) = ?

16+24

```
  0 0010 000
    0001 0000
+   0001 1000
-------------
    0010 1000
```

40

16+(-24)

```
  0 0010 000
    0001 0000
+   1001 1000
-------------
    1010 1000
```

-40 !

- Représentation signe / valeur absolue
  - Difficultés au niveau des opérations arithmétiques
    - Exemple : 16+24 = ? et 16 + (-24) = ?

16+(-24)

0 0010 000
0001 0000
1001 1000

+ 000[illegible]
0010 1000

40

Il nous faut deux algorithmes : l'un pour l'addition et le deuxième pour la soustraction

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de -2^{31} à $2^{31}-1$**
  - Mots sur $\{0,1\}$

	base 10
$1\ 0000000\ 00000000\ 00000000\ 00000000$	-2147483648
$1\ 0000000\ 00000000\ 00000000\ 00000001$	-2147483647
$\dots$	
$1\ 1111111\ 11111111\ 11111111\ 11111101$	-3
$1\ 1111111\ 11111111\ 11111111\ 11111110$	-2
$1\ 1111111\ 11111111\ 11111111\ 11111111$	-1
$0\ 0000000\ 00000000\ 00000000\ 00000000$	0
$0\ 0000000\ 00000000\ 00000000\ 00000001$	1
$\dots$	
$0\ 1111111\ 11111111\ 11111111\ 11111101$	2147483645
$0\ 1111111\ 11111111\ 11111111\ 11111110$	2147483646
$0\ 1111111\ 11111111\ 11111111\ 11111111$	2147483647

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de -2^{31} à $2^{31}-1$**
  - Mots sur $\{0,1\}$

bit de signe		base 10
1	0000000 00000000 00000000 00000000	-2147483648
1	0000000 00000000 00000000 00000001	-2147483647
	...	
1	1111111 11111111 11111111 11111101	-3
1	1111111 11111111 11111111 11111110	-2
1	1111111 11111111 11111111 11111111	-1
0	0000000 00000000 00000000 00000000	0
0	0000000 00000000 00000000 00000001	1
	...	
0	1111111 11111111 11111111 11111101	2147483645
0	1111111 11111111 11111111 11111110	2147483646
0	1111111 11111111 11111111 11111111	2147483647

- **Un choix parmi les $2^{32}!$ choix pour représenter les entiers de -2^{31} à $2^{31}-1$**
  - Mots sur $\{0,1\}$

bit de signe		base 10
1	0000000 00000000 00000000 00000000	-2147483648
1	0000000 00000000 00000000 00000001	-2147483647
	...	
1	1111111 11111111 11111111 11111101	-3
1	1111111 11111111 11111111 11111110	-2
1	1111111 11111111 11111111 11111111	-1
0	0000000 00000000 00000000 00000000	0
0	0000000 00000000 00000000 00000001	1
	...	
0	1111111 11111111 11111111 11111101	2147483645
0	1111111 11111111 11111111 11111110	2147483646
0	1111111 11111111 11111111 11111111	2147483647

Complément à 2

- Complément à 2 : comment coder

	base 10
1 0000000 00000000 00000000 00000000	-2147483648
1 0000000 00000000 00000000 00000001	-2147483647
1 0000000 00000000 00000000 00000010	-2147483646
...	
1 1111111 11111111 11111111 11111110	-2
1 1111111 11111111 11111111 11111111	-1
0 0000000 00000000 00000000 00000000	0
0 0000000 00000000 00000000 00000001	1
0 0000000 00000000 00000000 00000010	2
...	
0 1111111 11111111 11111111 11111110	2147483646
0 1111111 11111111 11111111 11111111	2147483647

- Complément à 2 : comment coder

	base 10
1 0000000 00000000 00000000 00000000	-2147483648
1 0000000 00000000 00000000 00000001	-2147483647
1 0000000 00000000 00000000 00000010	-2147483646
...	
1 1111111 11111111 11111111 11111110	-2
1 1111111 11111111 11111111 11111111	-1
0 0000000 00000000 00000000 00000000	0
0 0000000 00000000 00000000 00000001	1
0 0000000 00000000 00000000 00000010	2
...	
0 1111111 11111111 11111111 11111110	2147483646
0 1111111 11111111 11111111 11111111	2147483647

divisions par 2

- Complément à 2 : comment coder
  - Nous utilisons $b + (1 - b) = 1$ et $\overbrace{11\cdots1}^{n} + 1 = \overbrace{100\cdots0}^{n}$

	base 10
$1\ 0000000\ 00000000\ 00000000\ 00000000$	-2147483648
$1\ 0000000\ 00000000\ 00000000\ 00000001$	-2147483647
$1\ 0000000\ 00000000\ 00000000\ 00000010$	-2147483646
$\cdots$	
$1\ 1111111\ 11111111\ 11111111\ 11111110$	-2
$1\ 1111111\ 11111111\ 11111111\ 11111111$	-1
$0\ 0000000\ 00000000\ 00000000\ 00000000$	0
$0\ 0000000\ 00000000\ 00000000\ 00000001$	1
$0\ 0000000\ 00000000\ 00000000\ 00000010$	2
$\cdots$	
$0\ 1111111\ 11111111\ 11111111\ 11111110$	2147483646
$0\ 1111111\ 11111111\ 11111111\ 11111111$	2147483647

divisions par 2

- Complément à 2 : comment coder
  - Nous utilisons $b + (1 - b) = 1$ et $\overbrace{11\cdots1}^{n} + 1 = \overbrace{100\cdots0}^{n}$

	base 10
1 0000000 00000000 00000000 00000000	-2147483648
1 0000000 00000000 00000000 00000001	-2147483647
1 0000000 00000000 00000000 00000010	-2147483646
...	
1 1111111 11111111 11111111 11111110	-2
1 1111111 11111111 11111111 11111111	-1
0 0000000 00000000 00000000 00000000	0
0 0000000 00000000 00000000 00000001	1
0 0000000 00000000 00000000 00000010	2
...	
0 1111111 11111111 11111111 11111110	2147483646
0 1111111 11111111 11111111 11111111	2147483647

1. coder valeur absolue
2. inverser tous les bits
3. ajouter 1

divisions par 2

- Complément à 2 : comment coder
  - Nous utilisons $b + (1 - b) = 1$ et $\overbrace{11\cdots1}^{n} + 1 = \overbrace{100\cdots0}^{n}$

	base 10
1 0000000 00000000 00000000 00000000	-2147483648
1 0000000 00000000 00000000 00000001	-2147483647
1 0000000 00000000 00000000 00000010	-2147483646
...	
1 1111111 11111111 11111111 11111110	-2
1 1111111 11111111 11111111 11111111	-1
0 0000000 00000000 00000000 00000000	0
0 0000000 00000000 00000000 00000001	1
0 0000000 00000000 00000000 00000010	2
...	
0 1111111 11111111 11111111 11111110	2147483646
0 1111111 11111111 11111111 11111111	2147483647

divisions par 2

1. coder valeur absolue
2. inverser tous les bits
3. ajouter 1

- Complément à 2 : comment coder
  - Nous utilisons $b + (1 - b) = 1$ et $\overbrace{11\cdots1}^{n} + 1 = \overbrace{100\cdots0}^{n}$

	base 10
1 0000000 00000000 00000000 00000000	-2147483648
1 0000000 00000000 00000000 00000001	-2147483647
1 0000000 00000000 00000000 00000010	-2147483646
...	
1 1111111 11111111 11111111 11111110	-2
1 1111111 11111111 11111111 11111111	-1
0 0000000 00000000 00000000 00000000	0
0 0000000 00000000 00000000 00000001	1
0 0000000 00000000 00000000 00000010	2
...	
0 1111111 11111111 11111111 11111110	2147483646
0 1111111 11111111 11111111 11111111	2147483647

divisions par 2

1. coder valeur absolue
2. inverser tous les bits
3. ajouter 1

- Complément à 2 : comment coder
  - Nous utilisons $b + (1 - b) = 1$ et $\overbrace{11\cdots1}^{n} + 1 = \overbrace{100\cdots0}^{n}$

	base 10
1 0000000 00000000 00000000 00000000	-2147483648
1 0000000 00000000 00000000 00000001	-2147483647
1 0000000 00000000 00000000 00000010	-2147483646
...	
1 1111111 11111111 11111111 11111110	-2
1 1111111 11111111 11111111 11111111	-1
0 0000000 00000000 00000000 00000000	0
0 0000000 00000000 00000000 00000001	1
0 0000000 00000000 00000000 00000010	2
...	
0 1111111 11111111 11111111 11111110	2147483646
0 1111111 11111111 11111111 11111111	2147483647

divisions par 2

1. coder valeur absolue
2. inverser tous les bits
3. ajouter 1

- Complément à 2 : comment coder
  - Nous utilisons $b + (1 - b) = 1$ et $\overbrace{11\cdots1}^{n} + 1 = \overbrace{100\cdots0}^{n}$

	base 10
1 0000000 00000000 00000000 00000000	-2147483648
1 0000000 00000000 00000000 00000001	-2147483647
1 0000000 00000000 00000000 00000010	-2147483646
...	
1 1111111 11111111 11111111 11111110	-2
1 1111111 11111111 11111111 11111111	-1
0 0000000 00000000 00000000 00000000	0
0 0000000 00000000 00000000 00000001	1
0 0000000 00000000 00000000 00000010	2
...	
0 1111111 11111111 11111111 11111110	2147483646
0 1111111 11111111 11111111 11111111	2147483647

divisions par 2

1. coder valeur absolue
2. inverser tous les bits
3. ajouter 1

- Complément à 2 : comment coder
  - Nous utilisons $b + (1 - b) = 1$ et $\overbrace{11\cdots1}^{n} + 1 = \overbrace{100\cdots0}^{n}$

	base 10
$1\ 0000000\ 00000000\ 00000000\ 00000000$	-2147483648
$1\ 0000000\ 00000000\ 00000000\ 00000001$	-2147483647
$1\ 0000000\ 00000000\ 00000000\ 00000010$	-2147483646
1	-2
1	-1
0	0
0	1
0	2
$\cdots$	
$0\ 1111111\ 11111111\ 11111111\ 11111110$	2147483646
$0\ 1111111\ 11111111\ 11111111\ 11111111$	2147483647

- Comment decoder
  - Si bit de signe est 0
    - Méthode d'Horner
  - Si bit de signe est 1
    1. Inverser tous les bits
    2. Méthode de Horner
    3. Ajouter 1

absolue
s les bits

divisions par 2

- Complément à 2 : opérations arithmétiques
  - Exemple : 16+24 = ? et 16 + (-24) = ?

- Complément à 2 : opérations arithmétiques
  - Exemple : 16+24 = ? et 16 + (-24) = ?
    - 16 = 00010000
    - 24 = 00011000
    - -24 = 11101000

- Complément à 2 : opérations arithmétiques
  - Exemple : 16+24 = ? et 16 + (-24) = ?
    - 16 = 00010000
    - 24 = 00011000
    - -24 = 11101000

16+24

$$\begin{array}{r} {\scriptstyle 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0} \\ 0001\,0000 \\ +\ \ 0001\,1000 \\ \hline 0010\,1000 \end{array}$$

40

- Complément à 2 : opérations arithmétiques
  - Exemple : 16+24 = ? et 16 + (-24) = ?
    - 16 = 00010000
    - 24 = 00011000
    - -24 = 11101000

16+24

```
  0 0010 000
    0001 0000
+   0001 1000
-------------
    0010 1000
```

40

16+(-24)

```
  0 0000 000
    0001 0000
+   1110 1000
-------------
    1111 1000
```

-8 !

- Complément à 2 : opérations arithmétiques
  - Exemple : 16+24 = ? et 16 + (-24) = ?
    - 16 = 00010000
    - 00011000
    - 1000

Un seul algorithme pour l'addition et pour la soustraction

16+(-24)

0 0000 000
0001 0000
1000

0 0010 00
0001 0000
+ 0001 1000
0010 1000

40

- Complément à 2 et débordement
  - Retenue à gauche 1 n'implique pas nécessairement débordement

- Complément à 2 et débordement
  - Retenue à gauche 1 n'implique pas nécessairement débordement
  - Exemples (4 bits) : 9-4, -9-4, -9+9
    - $9 \rightarrow 01001$ et $-9 \rightarrow 10111$
    - $4 \rightarrow 00100$ et $-4 \rightarrow 11100$

- Complément à 2 et débordement
  - Retenue à gauche 1 n'implique pas nécessairement débordement
  - Exemples (4 bits) : 9-4, -9-4, -9+9
    - $9 \rightarrow 01001$ et $-9 \rightarrow 10111$
    - $4 \rightarrow 00100$ et $-4 \rightarrow 11100$

9-4

$$\begin{array}{r} 11\,000 \\ 01001 \\ +\ 11100 \\ \hline 00101 \end{array}$$

- retenue
- résultat positif (5)
- pas de débordement

- Complément à 2 et débordement
  - Retenue à gauche 1 n'implique pas nécessairement débordement
  - Exemples (4 bits) : 9-4, -9-4, -9+9
    - $9 \rightarrow 01001$ et $-9 \rightarrow 10111$
    - $4 \rightarrow 00100$ et $-4 \rightarrow 11100$

9-4

$$\begin{array}{r} 11\,000 \\ 01001 \\ +\ 11100 \\ \hline 00101 \end{array}$$

- retenue
- résultat positif (5)
- pas de débordement

-9-4

$$\begin{array}{r} 11\,100 \\ 10111 \\ +\ 11100 \\ \hline 10011 \end{array}$$

- retenue
- résultat négatif (-13)
- pas de débordement

- Complément à 2 et débordement
  - Retenue à gauche 1 n'implique pas nécessairement débordement
  - Exemples (4 bits) : 9-4, -9-4, -9+9
    - $9 \rightarrow 01001$ et $-9 \rightarrow 10111$
    - $4 \rightarrow 00100$ et $-4 \rightarrow 11100$

9-4

$$\begin{array}{r} 11\,000 \\ 01001 \\ +\ 11100 \\ \hline 00101 \end{array}$$

- retenue
- résultat positif (5)
- pas de débordement

-9-4

$$\begin{array}{r} 11\,100 \\ 10111 \\ +\ 11100 \\ \hline 10011 \end{array}$$

- retenue
- résultat négatif (-13)
- pas de débordement

-9+9

$$\begin{array}{r} 11\,111 \\ 10111 \\ +\ 01001 \\ \hline 00000 \end{array}$$

- retenue
- résultat positif (0)
- pas de débordement

- Complément à 2 et débordement
  - Débordement
    - Si la somme de deux nombres positifs donne un nombre négatif
    - Si la somme de deux nombres négatifs donne un nombre positif

- Complément à 2 et débordement
  - Débordement
    - Si la somme de deux nombres positifs donne un nombre négatif
    - Si la somme de deux nombres négatifs donne un nombre positif
  - Exemples (4 bits) : 9+8, -9-8
    - $9 \rightarrow 01001$ et $-9 \rightarrow 10111$

- Complément à 2 et débordement
  - Débordement
    - Si la somme de deux nombres positifs donne un nombre négatif
    - Si la somme de deux nombres négatifs donne un nombre positif
  - Exemples (4 bits) : 9+8, -9-8
    - $9 \rightarrow 01001$ et $-9 \rightarrow 10111$
    - $8 \rightarrow 01000$ et $-8 \rightarrow 11000$

9+8

$$\begin{array}{r} {\scriptstyle 01\,000} \\ 0\,1001 \\ +\ 0\,1000 \\ \hline 1\,0001 \end{array}$$

- pas de retenue
- résultat négatif (-15)
- débordement

- Complément à 2 et débordement
  - Débordement
    - Si la somme de deux nombres positifs donne un nombre négatif
    - Si la somme de deux nombres négatifs donne un nombre positif
  - Exemples (4 bits) : 9+8, -9-8
    - $9 \rightarrow 01001$ et $-9 \rightarrow 10111$
    - $8 \rightarrow 01000$ et $-8 \rightarrow 11000$

9+8

$$\begin{array}{r} 01\,000 \\ 0\,1001 \\ +\ 0\,1000 \\ \hline 1\,0001 \end{array}$$

- pas de retenue
- résultat négatif (-15)
- débordement

-9-8

$$\begin{array}{r} 10\,000 \\ 1\,0111 \\ +\ 1\,1000 \\ \hline 0\,1111 \end{array}$$

- retenue
- résultat positif (15)
- débordement

- Complément à 2 et débordement
  - Débordement
    - Si la somme de deux nombres positifs donne un nombre négatif
    - Si la somme de deux nombres négatifs donne un nombre positif
  - Exemples (4 bits) : 9+8, -9-8
    - $9 \rightarrow 01001$ et $-9 \rightarrow 10111$
    - $8 \rightarrow$

Jamais un débordement si les deux nombres sont de signes différents

$$\begin{array}{r} 01001 \\ +\ 01000 \\ \hline 10001 \end{array}$$

- pas de retenue
- résultat négatif (-15)
- débordement

$$\begin{array}{r} 10111 \\ +\ 11000 \\ \hline 01111 \end{array}$$

- retenue
- résultat positif (15)
- débordement

- (Si) on manipule rarement les très très très grands nombres
- (Si) on manipule rarement les nombres avec une grande précision
  - Choix de fixer la taille des représentations
    - Architectures communes 32 ou 64 bits :
  - Une infinité de choix pour représenter des réels
  - Problème : comment indiquer à la machine la position de la virgule ?
    - Deux méthodes
      - Virgule fixe : la position de la virgule est fixe
      - Virgule flottante : la position de la virgule change

- **Représentation en virgule fixe**
  - Partie entière est représentée sur n bits
  - Partie fractionnelle sur p bits
  - Un bit est utilisé pour le signe

- **Représentation en virgule fixe**
  - Partie entière est représentée sur n bits
  - Partie fractionnelle sur p bits
  - Un bit est utilisé pour le signe
  - Exemple : $n = 3$ et $p = 2$

- **Représentation en virgule fixe**
  - Partie entière est représentée sur n bits
  - Partie fractionnelle sur p bits
  - Un bit est utilisé pour le signe
  - Exemple : $n = 3$ et $p = 2$

	base 10
$0\ 000\ 00$	0

- **Représentation en virgule fixe**
  - Partie entière est représentée sur n bits
  - Partie fractionnelle sur p bits
  - Un bit est utilisé pour le signe
  - Exemple : $n = 3$ et $p = 2$

	base 10
$0\ 000\ 00$	0
$0\ 000\ 01$	$0,25$

- **Représentation en virgule fixe**
  - Partie entière est représentée sur n bits
  - Partie fractionnelle sur p bits
  - Un bit est utilisé pour le signe
  - Exemple : $n = 3$ et $p = 2$

	base 10
$0\ 000\ 00$	0
$0\ 000\ 01$	$0,25$
$0\ 000\ 10$	$0,5$

- **Représentation en virgule fixe**
  - Partie entière est représentée sur n bits
  - Partie fractionnelle sur p bits
  - Un bit est utilisé pour le signe
  - Exemple : $n = 3$ et $p = 2$

	base 10
$0\ 000\ 00$	0
$0\ 000\ 01$	$0,25$
$0\ 000\ 10$	$0,5$
$0\ 000\ 11$	$0,75$

- **Représentation en virgule fixe**
  - Partie entière est représentée sur n bits
  - Partie fractionnelle sur p bits
  - Un bit est utilisé pour le signe
  - Exemple : $n = 3$ et $p = 2$

	base 10
$0\ 000\ 00$	0
$0\ 000\ 01$	$0,25$
$0\ 000\ 10$	$0,5$
$0\ 000\ 11$	$0,75$
$0\ 001\ 00$	1

- **Représentation en virgule fixe**
  - Partie entière est représentée sur n bits
  - Partie fractionnelle sur p bits
  - Un bit est utilisé pour le signe
  - Exemple : $n = 3$ et $p = 2$

	base 10
0 000 00	0
0 000 01	$0,25$
0 000 10	$0,5$
0 000 11	$0,75$
0 001 00	1

Dans cette représentation les valeurs sont limitées et nous n'avons pas une grande précision

- **Représentation en virgule flottante**
  - Chaque nombre réel peut s'écrire de la façon suivante :
    $$n = \pm m \times b^e$$
  - m : mantisse
  - b : base
  - e : exposant
- Exemple
  - $(13,11)_{10} = +1,311 \times 10^1 = +0,1311 \times 10^2 = +131,1 \times 10^{-1}$
  - $(-110,101)_2 = -1,10101 \times 2^2 = -0,110101 \times 2^3 = 11010,1 \times 2^{-2}$