

---

## TD : Représentation des entiers/flottants

---

### 1 Exercices sur la numération

Dans toute cette section, on ne travaillera qu'avec des entiers positifs.

**Exercice 1.** *De la base 10 à une autre base.* Donner les représentations des nombres suivants dans la base indiquée.

- |                            |                    |
|----------------------------|--------------------|
| 1. 123 en bases 2, 3 et 5. | 3. 666 en base 2.  |
| 2. 2142 en base 16.        | 4. 1027 en base 7. |

**Corrigé.**

- $123 = \overline{1111011}^2 = \overline{11120}^3 = \overline{443}^5$ .
- $2142 = \overline{85E}^{16}$ .
- $666 = \overline{1010011010}^2$ .
- $1027 = \overline{2665}^7$ .

**Exercice 2.** *D'une base à la base 10.* Donner l'interprétation des nombres suivants en base 10.

- |                              |                           |
|------------------------------|---------------------------|
| 1. $\overline{2B4}^{16}$ .   | 3. $\overline{412}^5$ .   |
| 2. $\overline{11010010}^2$ . | 4. $\overline{20102}^3$ . |

**Corrigé.**  $\overline{2B4}^{16} = 692$ ,  $\overline{11010010}^2 = 210$ ,  $\overline{412}^5 = 107$ ,  $\overline{20102}^3 = 173$ .

**Exercice 3.** *Bases et puissances.* Convertir les nombres qui suivent dans la base indiquée, sans repasser pas par la base 10 (dix!).

- |                                      |   |                                     |
|--------------------------------------|---|-------------------------------------|
| 1. $\overline{4FC3}^{16}$ en base 2. | 3. $\overline{10110011011010011}^2$ en base 16. | 5. $\overline{184}^9$ en base 3.    |
| 2. $\overline{1231231}^4$ en base 2. | 4. $\overline{10110011011010011}^2$ en base 8.  | 6. $\overline{41023}^5$ en base 25. |

**Corrigé.**

1.  $\overline{4FC3}^{16} = \overline{100111111000011}^2$ .
2.  $\overline{1231231}^4 = \overline{1101101101101}^2$ .
3.  $\overline{10110011011010011}^2 = \overline{166D3}^{16}$ .
4.  $\overline{10110011011010011}^2 = \overline{263323}^8$ .
5.  $\overline{184}^9 = \overline{12211}^3$ .
6.  $\overline{41023}^5 = \overline{45D}^{25}$ , en convenant comme en hexadécimal que  $13 = \overline{D}^{25}$ .

**Exercice 4.** Combien faut-il de bits pour représenter tous les nombres à  $n$  chiffres (en base 10) ?

**Corrigé.** Il faut représenter les nombres jusqu'à  $10^n - 1$ . Un nombre non nul  $N$  possède  $\lfloor \log_2(N) \rfloor + 1$  bits, d'où  $1 + \lfloor \log_2(10^n - 1) \rfloor$  bits sont nécessaires.

**Exercice 5.** Quelle est la représentation binaire des nombres 1,7,31,127 ? Expliquer.

**Corrigé.**  $1 = \overline{1}^2$ ,  $7 = \overline{111}^2$ ,  $31 = \overline{11111}^2$  et  $127 = \overline{1111111}^2$ . Ces nombres sont égaux à une puissance de deux moins un, qui sont représentés comme une suite de uns en binaire.

**Exercice 6.** Soit  $p$  un entier supérieur ou égal à 2, et  $N$  un entier. Si l'on connaît l'écriture de  $N$  en base  $p$ , comment obtient-on (facilement) celles de  $p \times N$  et  $\left\lfloor \frac{N}{p} \right\rfloor$  ? (On rappelle que  $\lfloor . \rfloor$  dénote la partie entière). Le démontrer.

**Corrigé.** En notant  $N = \overline{a_{n-1} \cdots a_0}^p$ , on a  $\lfloor N/p \rfloor = \overline{a_{n-1} \cdots a_1}^p$  et  $p \times N = \overline{a_{n-1} \cdots a_0 0}^p$ . En effet :

- $N = \sum_{k=0}^{n-1} a_k p^k$ , donc  $N/p = \sum_{k=0}^{n-1} a_k p^{k-1}$ , et  $\lfloor N/p \rfloor = \sum_{k=1}^{n-1} a_k p^{k-1} = \overline{a_{n-1} \cdots a_1}^p$ .
- $p \times N = \sum_{k=0}^{n-1} a_k p^{k+1} = \overline{a_{n-1} \cdots a_0 0}^p$ .

**Exercice 7.** *Schéma de Hörner pour le changement de base.* Soit  $N = \overline{a_{n-1} \cdots a_0}^b$  un entier encodé dans la base  $b$ . L'algorithme vu en cours pour obtenir  $N$  dans une base où on sait calculer (pour nous, la base 10), consiste à évaluer la somme  $\sum_{k=0}^{n-1} a_k b^k$ . L'algorithme de Hörner pour cette évaluation consiste à utiliser l'idée suivante :

$$N = a_0 + b \times (a_1 + b \times (a_2 + b \times (a_3 + \cdots + b \times (a_{n-2} + b \times a_{n-1}) \cdots)))$$

1. Écrire l'algorithme en pseudo-code.
2. L'implémenter en Python, en supposant les chiffres de  $N$  dans la base  $b$  stockés dans une liste.
3. Discuter de l'intérêt de cet algorithme par rapport à celui du cours, en nombre d'opérations.

**Corrigé.**

1. Voici :

---

**Entrées :** Les  $a_i$ , la base  $b$ .  
**Sortie :**  $\sum_{i=0}^{n-1} a_i b^i$   
 $s \leftarrow 0$ ;  
**pour chaque**  $i$  *allant de*  $n-1$  *à*  $0$  *par pas de*  $-1$  **faire**  
     $s \leftarrow b \times s + a_i$   
**retourner**  $s$

---

2. Donnons une fonction :

```
def eval_horner(L, b): #L contient [a0, a1,...]
    s=0
    for i in range(len(L)-1,-1,-1):
        s=s*b+L[i]
    return s
```

3. Cet algorithme fait  $n$  additions et  $n$  multiplications, avec  $n$  le nombre de chiffres. C'est un peu moins que l'algorithme du cours qui faisait  $2n$  multiplications (et autant d'additions).

## 2 Les entiers relatifs en base 2

Dans toute cette section, on ne travaillera qu'avec des nombres entiers. Pour les deux premiers exercices, on travaille avec des entiers naturels, pour les autres, on travaille sur des entiers relatifs représentés en complément à 2.

**Exercice 8.** *Additions d'entiers naturels sur 8 bits.* Pour chacun des couples de nombres  $a, b$  de 8 bits suivants, effectuer l'addition  $a + b$ . Quelles sont celles qui donnent lieu à un dépassement de capacité ?

1.  $(a, b) = (\overline{11111111}^2, \overline{00000001}^2)$
2.  $(a, b) = (\overline{10010111}^2, \overline{01011001}^2)$
3.  $(a, b) = (\overline{11010100}^2, \overline{00111001}^2)$
4.  $(a, b) = (\overline{01101010}^2, \overline{01101001}^2)$

**Corrigé.** Il se produit un dépassement de capacité lorsque la retenue sortante vaut 1, c'est le cas en 1) et 3).

$  \begin{array}{r}  \begin{array}{cccccccc}  1^1 & 1^1 & 1^1 & 1^1 & 1^1 & 1^1 & 1^1 & 1 \\  + & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\  \hline  \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0  \end{array} \\  \\  \begin{array}{r}  \begin{array}{cccccccc}  1^1 & 1^1 & 0^1 & 1 & 0 & 1 & 0 & 0 \\  + & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\  \hline  \boxed{1} & 0 & 0 & 0 & 0 & 1 & 1 & 0  \end{array}  \end{array}  $	$  \begin{array}{r}  \begin{array}{cccccccc}  1 & 0 & 0^1 & 1^1 & 0^1 & 1^1 & 1^1 & 1 \\  + & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\  \hline  \boxed{0} & 1 & 1 & 1 & 1 & 0 & 0 & 0  \end{array} \\  \\  \begin{array}{r}  \begin{array}{cccccccc}  0^1 & 1^1 & 1 & 0^1 & 1 & 0 & 1 & 0 \\  + & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\  \hline  \boxed{0} & 1 & 1 & 0 & 1 & 0 & 0 & 1  \end{array}  \end{array}  $
--	--

**Exercice 9.** *Multiplieur sur les entiers naturels.* Dans cet exercice, on va voir comment réaliser une multiplication à l'aide d'additions et de l'opérateur de décalage vers la gauche.

1. Soit  $N$  et  $M$  deux entiers naturels de  $n$  bits. Combien faut-il de bits (dans le pire de cas) pour représenter le produit  $N \times M$ ? (On discutera suivant si  $n = 1$  ou  $n > 1$ ).
2. On suppose dorénavant que  $n > 1$ , et que l'on dispose de suffisamment de bits pour écrire le résultat de la multiplication. Soit  $a = \overline{a_{n-1} \cdots a_0}^2$ . On dénote par  $a \ll 1$  l'entier dont la représentation binaire est  $\overline{a_{n-1} \cdots a_0 0}^2$ , et plus généralement par  $a \ll p$  l'entier représenté par  $\overline{a_{n-1} \cdots a_0 \underbrace{0 \cdots 0}_p}^2$ . Traduite sur les entiers, à quelle opération correspond un décalage à gauche?
3. Si  $M = 2^p$ , quel opération simple permet d'obtenir le produit  $N \times M$ ?
4. En déduire un algorithme permettant de multiplier  $N$  et  $M$ , n'effectuant que des tests sur les bits de  $M$ , des décalages à gauche, et des additions.

**Corrigé.**

1. Si  $n = 1$ , un seul bit suffit. Sinon avec  $n \geq 2$ ,  $N$  et  $M$  sont inférieurs à  $2^n - 1$ , donc le produit est inférieur à  $2^{2n} - 2^{n+1} + 1 \geq 2^{2n-1}$ , donc  $2n$  bits sont suffisants, et sont nécessaires par exemple lorsque  $N = M = 2^n - 1$ .
2. Un décalage à gauche correspond à une multiplication par 2.
3. Dans ce cas  $N \times M = N \ll p$ .
4. Il suffit de parcourir les bits de  $M = \overline{a_{n-1} \cdots a_0}^2$ , à l'aide d'une variable  $s$  initialement nulle. Si  $a_i$  vaut 1, on rajoute  $N \ll i$  à  $s$ . Une fois  $M$  parcouru,  $s$  contient  $N \times M$ .

**Exercice 10. Entiers relatifs sur 8 bits.** On rappelle qu'en représentation en complément à 2 sur  $n$  bits (complément à  $2^n$  pour être exact!), on peut représenter tous les nombres de l'intervalle  $[-2^{n-1}, 2^{n-1} - 1]$ . Donner les représentations des entiers suivants, sur 8 bits.

- |             |              |               |
|-------------|--------------|---------------|
| 1. $N = 0$  | 3. $N = 42$  | 5. $N = 127$  |
| 2. $N = -1$ | 4. $N = -42$ | 6. $N = -128$ |

**Corrigé.**  $0 = \overline{00000000}^2$ ,  $-1 = \overline{11111111}^2$ ,  $42 = \overline{00101010}^2$ ,  $-42 = \overline{11010110}^2$ ,  $127 = \overline{01111111}^2$  et  $-128 = \overline{10000000}^2$ , en complément à deux.

**Exercice 11. Entiers relatifs sur 8 bits, l'inverse.** À l'inverse de l'exercice précédent, donnez maintenant la valeur des entiers relatifs suivants, codés sur 8 bits en complément à 2.

- |                                |                                |                                |
|--------------------------------|--------------------------------|--------------------------------|
| 1. $N = \overline{01101010}^2$ | 2. $N = \overline{11000101}^2$ | 3. $N = \overline{10001110}^2$ |
|--------------------------------|--------------------------------|--------------------------------|

**Corrigé.**  $\overline{01101010}^2 = 106$ ,  $\overline{11000101}^2 = -56$ ,  $\overline{10001110}^2 = -114$ , en complément à deux.

**Exercice 12. Additions d'entiers relatifs.** Effectuer les additions des couples suivants représentant des entiers relatifs. Pour lesquelles y a-t-il dépassement de capacité? (c'est à dire que la somme attendue n'est pas représentable sur 8 bits?)

- |  |  |
|--|--|
| 1. $(a, b) = (\overline{11111111}^2, \overline{00000001}^2)$ | 4. $(a, b) = (\overline{11010100}^2, \overline{10011001}^2)$ |
| 2. $(a, b) = (\overline{10010111}^2, \overline{01011001}^2)$ | 5. $(a, b) = (\overline{01010100}^2, \overline{00011001}^2)$ |
| 3. $(a, b) = (\overline{11010110}^2, \overline{10001010}^2)$ | 6. $(a, b) = (\overline{01010100}^2, \overline{01011001}^2)$ |

**Corrigé.** Il y a dépassement de capacité sur entiers relatifs si et seulement si les deux dernières retenues (sur le bit de poids fort, et la retenue sortante) sont différentes. Il y a donc dépassement dans les cas 3), 4) et 6).

$  \begin{array}{r}  \begin{array}{cccccccc} 1^1 & 1^1 & 1^1 & 1^1 & 1^1 & 1^1 & 1^1 & 1 \end{array} & (L) \\  + & \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} & (M) \\  \hline  \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0  \end{array}  $	$  \begin{array}{r}  \begin{array}{cccccccc} 1 & 0 & 0^1 & 1^1 & 0^1 & 1^1 & 1^1 & 1 \end{array} & (L) \\  + & \begin{array}{cccccccc} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array} & (M) \\  \hline  \boxed{0} & 1 & 1 & 1 & 1 & 0 & 0 & 0  \end{array}  $
$  \begin{array}{r}  \begin{array}{cccccccc} 1 & 1 & 0^1 & 1^1 & 0^1 & 1^1 & 1 & 0 \end{array} & (L) \\  + & \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array} & (M) \\  \hline  \boxed{1} & 0 & 1 & 1 & 0 & 0 & 0 & 0  \end{array}  $	$  \begin{array}{r}  \begin{array}{cccccccc} 1 & 1 & 0^1 & 1 & 0 & 1 & 0 & 0 \end{array} & (L) \\  + & \begin{array}{cccccccc} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{array} & (M) \\  \hline  \boxed{1} & 0 & 1 & 1 & 0 & 1 & 1 & 0  \end{array}  $
$  \begin{array}{r}  \begin{array}{cccccccc} 0 & 1 & 0^1 & 1 & 0 & 1 & 0 & 0 \end{array} & (L) \\  + & \begin{array}{cccccccc} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{array} & (M) \\  \hline  \boxed{0} & 0 & 1 & 1 & 0 & 1 & 1 & 0  \end{array}  $	$  \begin{array}{r}  \begin{array}{cccccccc} 0^1 & 1 & 0^1 & 1 & 0 & 1 & 0 & 0 \end{array} & (L) \\  + & \begin{array}{cccccccc} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array} & (M) \\  \hline  \boxed{0} & 1 & 0 & 1 & 0 & 1 & 1 & 0  \end{array}  $

**Exercice 13.** *Opposé d'un entier.* Soit  $N$  un entier relatif codé en complément à 2 sur  $n$  bits. On note  $\overline{N}$  l'entier obtenu en transformant les bits de  $N$  égaux à 1 en zéro et réciproquement.

1. Que vaut  $N + \overline{N}$  ?
2. Comment obtenir la représentation de l'opposé d'un nombre en complément à 2, avec une telle transformation et une addition ?
3. Que se passe-t-il avec  $N = -2^{n-1}$  sur  $n$  bits ? Expliquer.

*Ainsi, toujours avec la même méthode, il est possible de réaliser facilement des soustractions, car  $N - M = N + (-M)$ . C'est ainsi qu'elles sont réalisées dans un processeur.*

**Corrigé.**

1.  $N + \overline{N} = \overline{11 \cdots 1}^2 = -1$  (c'est légitime car l'opération ne produit pas de retenue, il n'y a pas dépassement de capacité sur entiers relatifs).
2. On en déduit que  $-N = \overline{N} + 1$ , il suffit donc d'ajouter 1 à  $\overline{N}$  pour obtenir  $-N$ .
3. Cet entier est le seul pour lequel l'addition produit un dépassement de capacité. On retombe (en ignorant la retenue sortante) sur  $N$ . En effet,  $-N = 2^{n-1}$  n'est pas représentable sur  $n$  bits.

**Exercice 14.** *Caractérisation des dépassements de capacité sur entiers relatifs.* Soit  $a = \overline{a_{n-1}a_{n-2} \cdots a_0}^2$  et  $b = \overline{b_{n-1}b_{n-2} \cdots b_0}^2$  deux entiers relatifs de  $n$  bits en complément à 2. On note  $r_0 = 0$ , et pour tout  $i \in \llbracket 1, n \rrbracket$ , on note  $r_i$  la retenue correspondant à l'addition des bits  $a_{i-1}, b_{i-1}$  et  $r_{i-1}$ . On cherche à montrer que l'addition se fait sans dépassement de capacité si et seulement si,  $r_n = r_{n-1}$ .

1. Vérifier que le résultat est vrai sur les exemples de l'exercice 12.
2. Supposons que  $a$  et  $b$  représentent deux entiers naturels. Que valent  $a_{n-1}, b_{n-1}$  et  $r_n$  ? Justifier que l'addition se fait sans dépassement si et seulement si  $r_{n-1} = 0$ , et conclure pour les entiers naturels.
3. Supposons que  $a$  et  $b$  soient deux entiers strictement négatifs. Que valent  $a_{n-1}, b_{n-1}$  et  $r_n$  ? Justifier que l'addition se fait sans dépassement si et seulement si  $r_{n-1} = 1$ , et conclure pour les entiers négatifs.
4. Enfin, supposons que  $a$  est un entier naturel, et  $b$  un entier strictement négatif. Justifier qu'il n'y a pas dépassement de capacité. Que vaut  $a_{n-1} + b_{n-1}$  ? Conclure en discutant suivant les valeurs de  $r_{n-1}$ .

*En pratique, c'est exactement comme ça que l'on teste le dépassement de capacité sur entiers relatifs dans un processeur !*

**Corrigé.**

1. Ça fonctionne !
2.  $a_{n-1} = b_{n-1} = 0$ . Quelle que soit la valeur de  $r_{n-1}$ ,  $r_n$  vaut 0. Tout se passe comme une addition sur  $n - 1$  bits sur entiers naturels, il n'y a donc pas dépassement si et seulement si  $r_{n-1}$  vaut 0, d'où le résultat.
3.  $a_{n-1} = b_{n-1} = 1$ . Quelle que soit la valeur de  $r_{n-1}$ ,  $r_n$  vaut 1. Si  $r_{n-1}$  vaut 0, alors il y a dépassement (sinon on obtiendrait un nombre positif en sommant deux négatifs). Sinon, le résultat est correct.
4. Il n'y a jamais dépassement de capacité sur entiers relatifs lorsqu'on additionne deux nombres de signes opposés (on reste dans l'intervalle de représentation). On a  $a_{n-1} + b_{n-1} = 1$ . Si  $r_{n-1} = 1$ , alors  $r_n = 1$ , et si  $r_{n-1} = 0$ , alors  $r_n = 0$ , donc les deux dernières retenues sont égales.

### 3 Exercices sur les flottants

Dans toute cette section, on considérera des nombres flottants, donnés par leur représentation par signe, mantisse, exposant. On rappelle ici le nombre de bits utilisés dans les formats classiques (simple précision 32 bits et double précision 64 bits), et on donne également une représentation de flottants sur 9 bits « maison » que l'on utilisera pour certains exercices (ça fait moins de chiffres !) Dans la représentation ci-dessous, l'exposant décalé est un entier naturel, et on utilise un décalage correspondant à  $2^{\text{taille de l'exposant décalé} - 1} - 1$ .