# Informatique: TD2

MP2I Lycée Pierre de Fermat

La calculatrice est **interdite** pour ce TD !

## Exercice 1.

*Hexadécimal*

**Q1.** Donnez l'écriture décimale des entiers suivants :

a) $\overline{100101}^2$ b) $\overline{253}^7$ c) $\overline{2AF}^{16}$

**Q2.** Écrivez $n$ en base $B$ pour :

a) $n = 100, B = 2$ b) $n = 91, B = 7$ c) $n = 3, B = 3$ d) $n = 59, B = 16$.

**Q3.** Soit $B$ une base quelconque. Écrivez $n$ en base $B$ pour :

a) $n = B^k$ avec $k \in \mathbb{N}$ b) $n = B^k - 1$ avec $k \in \mathbb{N}$

Rappel : une utilité de la base 16 (l'hexadécimal) est que l'on peut passer facilement de la base 16 à la base 2 et inversement : un chiffre en base 16 correspond à 4 chiffres en base 2.

**Q4.** Combien de chiffres hexadécimaux faut-il pour représenter un octet ? Et pour un entier 32 bits ?

**Q5.** Remplir le tableau suivant :

| | | | | | |
|---|---|---|---|---|---|
| **Dec** | 15 | 735 | 493 | | |
| **Bin** | 1111 | 0010 1101 1111 | | 1010 0110 0001 | |
| **Hex** | f | 2df | | | d4c |

## Exercice 2.

*Addition des entiers non-signés*

Dans cet exercice, on considère des entiers non-signés sur 8 bits.
Effectuer les additions suivantes **en binaire**, et dire lorsqu'elles causent un dépassement d'entier :

a) 95 + 132 b) 12 + 167 c) 7 + 64 d) 184 + 72

# Exercice 3. *Lecture en base B*

On considère l'algorithme suivant de lecture en base $B$ :

**Algorithme 1 :** Lecture depuis une base

**Entrée(s) :** $B$ une base, $a_{l-1} \dots a_0$ un mot sur l'alphabet $[\![0, B-1]\!]$
**Sortie(s) :** $n \in \mathbb{N}^*$ tel que $n = \overline{a_{l-1} \dots a_0}^B$

```
1 n ← 0;
2 i = 0;
  // Invariant: ...
3 tant que i < l faire
4 |   n ← n + a_i B^i;
5 |   i ← i + 1;
6 retourner n
```

**Q1.** Montrez la correction de cet algorithme à l'aide d'un bon invariant de boucle.
*Indication : Exprimez $n$ en base $B$ à chaque étape de l'algorithme.*

**Q2.** En fonction du nombre de chiffres $l$ lus, quel est le nombre total de multiplications effectuées par l'algorithme ? On considèrera que le calcul de puissance est fait naïvement, i.e. que calculer $x^k$ demande $k-1$ multiplications.

**Q3.** En introduisant une variable $p$ vérifiant l'invariant "$p = B^i$", réduire le nombre de multiplications que fait l'algorithme.

On s'intéresse maintenant à un deuxième algorithme, encore moins coûteux en nombre de multiplications, appelé la **méthode de Horner**. Cette méthode permet plus généralement d'évaluer la valeur d'un polynôme en un point, on l'applique ici au problème similaire de la lecture d'un nombre en base $B$ :

**Algorithme 2 :** Horner

**Entrée(s) :** $B$ une base, $a_{l-1} \dots a_0$ un mot sur l'alphabet $[\![0, B-1]\!]$
**Sortie(s) :** $n \in \mathbb{N}^*$ tel que $n = \overline{a_{l-1} \dots a_0}^B$

```
1 n ← 0;
2 i = l − 1;
  // Invariant: ...
3 tant que i ≥ 0 faire
4 |   n ← Bn + a_i;
5 |   i ← i − 1;
6 retourner n
```

**Q4.** Montrer la correction de cet algorithme. Quel est le nombre total de multiplications qu'il effectue en fonction de la longueur $l$ du mot en entrée ?

**Q5.** En s'inspirant de l'algorithme précédent, proposer un algorithme d'évaluation de polynôme économe en multiplications :

**Algorithme 3 :** Eval

**Entrée(s) :** $a_{l-1}, \dots, a_0$ coefficients d'un polynôme $P$, $x_0 \in \mathbb{R}$
**Sortie(s) :** $P(x_0) = \sum_{i=0}^{l-1} a_i x_0^i$

# Exercice 4. *Opérations bit à bit*

Lorsque l'on considère des entiers écrits en base 2, on doit avoir une vision duale : d'une part celle des entiers, d'autre part celle des mots sur l'alphabet $\{0, 1\}$.
Par exemple, l'octet 01001101 représente l'entier 77 ET est une suite de 8 valeurs booléennes.

On rappelle les trois opérateurs booléens bits à bits : &, | et $\sim$. Ces trois opérateurs font respectivement un ET, un OU et un NON bit à bit. Par exemple sur 4 bits :

$$1001\&1100 = 1000, 1001|1100 = 1101 \text{ et } \sim 1001 = 0110$$

On fixe $l \in \mathbb{N}$ un nombre de bits sur lequel les nombres sont écrits. On note $\mathbb{N}_l$ l'ensemble des entiers représentables sur $l$ bits non-signés. Les opérateurs bit-à-bit agissent techniquement sur les mots binaires, i.e. sur les suites de 0 et de 1, mais on étend assez naturellement leur action au entiers de $\mathbb{N}_l$ : pour $n$ et $m$ deux entiers ayant pour écritures binaires respectives $a_{l-1} \dots a_0$ et $b_{l-1} \dots b_0$, on aura :

$$n\&m = \overline{(a_{l-1} \dots a_0)\&(b_{l-1} \dots b_0)}^2$$

et idem pour | et $\sim$. Autrement dit, pour calculer $n\&m$, on écrit $n$ et $m$ en binaire sur $l$ bits, on calcule le ET bit à bit, et l'on interprète le résultat comme un entier en base 2.

**Q1.** En se plaçant sur $l = 8$ bits, donner les résultats des calculs suivants :

1. $100\&52$
2. $31\&64$
3. $95|44$
4. $31|64$
5. $\sim 216$
6. $(\sim 100)|100$

**Q2.** Pour $x \in \mathbb{N}_l$, que vaut $x| \sim x$ ? Et $x\& \sim x$ ? (une preuve formelle est attendue).

**Q3.** Montrez que pour tout $x, y \in \mathbb{N}_l$, $x\&y \leq \min(x, y)$. Sur 8 bits, trouvez un cas où l'inégalité est stricte, et un où il y a égalité mais où $x \neq y$.

On rappelle également les opérateurs de ***décalage*** : >> et << :

$$\forall u = a_{l-1} \dots a_0 \in \{0, 1\}^*, \forall k \in \mathbb{N}, u << k = a_{l-1-k} \dots a_0 00 \dots 0$$

avec $k$ 0 ajoutés à droite de $u$ et $k$ bits perdus à gauche, et

$$\forall u = a_{l-1} \dots a_0 \in \{0, 1\}^*, \forall k \in \mathbb{N}, u >> k = 00 \dots 0 a_{l-1} \dots a_k$$

avec $k$ 0 ajoutés à gauche de $u$ et $k$ bits perdus à droite. On étend l'action de ces opérateurs aux entiers de $\mathbb{N}_l$.

**Q4.** Sur 8 bits, donner le résultat des calculs suivants :

1. $7 << 3$
2. $73 << 5$
3. $32 >> 3$
4. $55 >> 2$

**Q5.** Pour $a \in \mathbb{N}_l$ un entier et $k \in \mathbb{N}$, exprimer $a >> k$ et $a << k$ en fonction de $a$, $k$ et $l$.

**Q6.** Pour $n \in \mathbb{N}_l$, donner une formule permettant d'obtenir le $i$-ème bit de $n$ en n'utilisant que les opérateurs bit-à-bit et les opérateurs de décalage.

**Q7.** Pour $n \in \mathbb{N}_l$, donner une formule permettant de changer le $i$-ème bit de $n$ en un 1.

**Q8.** Pour $n \in \mathbb{N}_l$, donner une formule permettant d'inverser le $i$-ème bit de $n$, i.e. transformer un 1 en 0 et inversement. (*Vous pouvez sauter cette question et y revenir après avoir fait l'exercice 5 si vous n'y arrivez pas.)*

# Exercice 5.

*Ou exclusif*

Le but de cet exercice est d'écrire un algorithme d'addition en binaire n'utilisant que des opérations booléennes. On introduit un nouvel opérateur booléen : le ***OU exclusif***, aussi appelé ***xor***, noté $\oplus$ (Cf Fig. 1).

| $x$ | $y$ | $x \oplus y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

FIGURE 1 – Table de vérité du ***xor*** logique

**Q1.** Rappelez les tables de valeurs des opérateurs booléens && || et !.

**Q2.** Pour $a$ et $b$ deux booléens, exprimez $a \oplus b$ en n'utilisant que les opérateurs && || et !.

**Q3.** Montrer que le xor est associatif.

En C, il n'existe pas de symbole pour le xor logique. En revanche, on peut faire des xor bit à bit avec l'accent circonflexe `^`.

**Q4.** On se place sur 8 bits. Calculer :

a) $9 \hat{} 20$ b) $91 \hat{} 107$ c) $255 \hat{} 183$

**Q5.** La question précédente change t-elle si l'on se place sur 16 ou 32 bits ?

**Q6.** Sur $l$ bits, pour $n$ un entier non-signé représentable, que vaut :

a) $n \hat{} 0$ ? b) $n \hat{} (2^l - 1)$ ? c) $n \hat{} n$ ?

**Q7.** Pour $n, m$ représentables sur $l$ bits, à quelle condition a-t-on que $n \hat{} m = n + m$ ?

**Q8.** Que fait le code suivant (on suppose que $x$ et $y$ ont été définies plus haut) :

```
1 x = x^y;
2 y = x^y;
3 x = x^y;
```

# Exercice 6.

*Addition binaire*

Cet exercice est à faire après l'exercice 5, et vous pouvez utiliser librement le OU exclusif.

Soient $a, b, c$ des booléens. On note $s$ et $c'$ les booléens qui représentent l'addition $a + b + c$, avec $s$ le résultat et $c'$ la retenue. Par exemple pour $a = b = c = 1$, on a $s = 1$ et $c' = 1$.

**Q1.** Exprimez $s$ et $c'$ en fonction de $a, b, c$ en n'utilisant que des opérateurs booléens.

L'idée de l'algorithme d'addition est d'utiliser les formules précédentes pour additionner deux nombres binaires $a_{k-1} \dots a_0$ et $b_{k-1} \dots b_0$ en faisant des additions bit à bit et en propageant la retenue.

**Q2.** De combien de bits a t-on besoin pour stocker le résultat d'une addition de deux nombres à $k$ bits ?

**Q3.** Écrivez un algorithme d'addition binaire :

**Algorithme 4 :** Addition binaire

**Entrée(s) :** $a = a_{k-1} \dots a_0$ et $b = b_{k-1} \dots b_0$ deux mots binaires
**Sortie(s) :** $s = s_k \dots s_0$ le résultat de l'addition de $a$ et $b$

# Exercice 7.

*Encodage de jeux*

On veut encoder une grille de morpion par un entier non signé 32 bits. On veut que notre encodage nous permette de récupérer l'état de la grille facilement en manipulant les bits.

On suppose qu'on a deux joueurs : le joueur X et le joueur O. Une case peut contenir un X, un O ou rien. On assigne à chaque case un nombre entre 0 et 3 selon son contenu :

$$\begin{array}{lcl} \text{X} & \mapsto & 2 \\ \text{O} & \mapsto & 3 \\ \text{vide} & \mapsto & 0 \end{array}$$

On remarque que le nombre 1 ne correspond à rien. Ensuite, on numérote les 9 cases d'une grille comme suit :

| | | |
|---|---|---|
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

Enfin, en notant $a_i$ le nombre correspondant au contenu de la case $i$, on encode une grille $G$ par :

$$\sum_{i=0}^{8} a_i 4^i$$

**Q1.** Donnez les écritures en bases 2, 4, et 16 de l'encodage des grilles suivantes :

a)

| | | |
|---|---|---|
| | | |
| | | |
| | | |

b)

| | | |
|---|---|---|
| | | $O$ |
| $X$ | $X$ | $O$ |
| | | $X$ |

c)

| | | |
|---|---|---|
| $O$ | $X$ | $O$ |
| $X$ | $X$ | $O$ |
| | $O$ | $X$ |

**Q2.** Si $n$ est l'encodage d'une grille et $0 \leq i \leq 8$, exprimez à l'aide des opérateurs >> et & une formule calculant un booléen indiquant si la case $i$ de la grille est remplie ou non.

**Q3.** Vérifier que l'encodage proposé tient sur 32 bits. Quel est le nombre minimal de bits nécessaire ?

**Q4.** Soit $n$ l'encodage d'une grille, et $0 \leq i \leq 8$. On suppose que la case $i$ de la grille encodée par $n$ est remplie. Exprimez à l'aide des opérateurs >> et & une formule calculant un booléen indiquant si la case $i$ de la grille contient un O ou un X.

**Q5.** Soit $n$ l'encodage d'une grille, et $0 \leq i \leq 8$. On suppose que la case $i$ de la grille encodée par $n$ est vide. Exprimez à l'aide des opérateurs << et | une formule calculant l'encodage de la grille obtenue en rajoutant le symbole X dans la case $i$

**Q6.** Idem pour le symbole O.

**Q7.** Expliquez, selon le même principe, comment on pourrait encoder une grille de Sudoku. De combien de bits aurait-on besoin ?