Semaine 03 – Représentation des nombres en machine

Exercice 1

1. Combien de mots de 4 bits différents existe-t-il ? Et de 32 bits ? Et de n bits ?
2. Combien y a-t-il de fichiers différents de 24355 octets ?
3. On veut coder l'alphabet (`a`, ..., `z`) en binaire. Combien faut-il de bits pour réaliser un tel codage ? Et si on veut également coder les capitales (`A`, ..., `Z`) ? Et les chiffres ?

Exercice 2

1. Quel est le plus grand entier *non signé* représentable sur 8 bits ? 16 bits ? 32 bits ? 64 bits ?
2. On suppose que `x` est une variable codée sur un octet (non signé) et que sa valeur est `11111001`. Donner la valeur de `x` en base 10. On incrémente `x` de 1. Que vaut `x` ? On ajoute 5 (ou `00000101`) à `x`. Que vaut `x` ? On incrémente `x` de 1. Que vaut `x` ?

Exercice 3

1. Donner la représentation des entiers suivants en complément à 2 sur 8 bits : 0, 13, -3, 42, -17, 127, -128, 131.
2. Quel est le plus grand entier *signé* représentable sur 8 bits ? 16 bits ? 32 bits ? 64 bits ? Et le plus petit ?
3. On suppose que `x` est une variable codée sur un octet en complément à 2 et que sa valeur est `01111001`. Donner la valeur de `x` en base 10. On incrémente `x` de 1. Que vaut `x` ? On ajoute 5 (ou `00000101`) à `x`. Que vaut `x` ? On incrémente `x` de 1. Que vaut `x` ?

Exercice 4

1. Quelles sont les valeurs représentées par `00101101` et `10101101` ? Et `01100011` et `11100011` ?
2. Quelle relation numérique y a-t-il entre les valeurs représentées (en complément à 2 sur n bits) par des mots w et w' qui ne diffèrent que par le bit de signe ?

Exercice 5

1. Soit x un nombre en complément à 2 sur 8 bits. Comment s'écrit-il sur 16 bits ?
2. À quelle condition un nombre écrit en complément à 2 sur 16 bits a-t-il une écriture sur 8 bits ? Plus généralement, même question pour un passage de m bits à n bits, avec $m > n$.
3. Préciser si les nombres suivants peuvent être encodés en complément à 2 sur un octet seulement :

$$(240)_{10}, \quad (100)_{10}, \quad (-130)_{10}, \quad (255)_{10}, \quad (\text{FF82})_{16}, \quad (0426)_{16}, \quad (\text{FF6A})_{16}, \quad (\text{F0C4})_{16}.$$

Exercice 6 Calculer les additions suivantes sur 8 bits en complément à deux, en prenant soin de noter les retenues :

$$14 + 59, \quad 59 + 80, \quad 59 + (-80), \quad -59 + (-14), \quad -59 + (-80), \quad 59 + (-59).$$

Donner la valeur décimale de chaque résultat. Faire la "preuve" par 3 pour chaque calcul.
Certains des résultats obtenus à la question précédente ne sont pas conformes à l'attente ou à l'arithmétique ordinaire (dépassement de capacité, *overflow* en anglais). Que s'est-il passé ? Quel test simple sur les retenues permet de détecter ces dépassements de capacité ?

Exercice 7 On travaille ici sur des entiers signés sur 16 bits en complément à 2. Faire les opérations suivantes, faire la "preuve" par 3, et donner à chaque fois les valeurs en base 10 des opérandes et des résultats :

1. $(0000110110110011) + (0000110111001001)$
2. $(0000110110110011) - (0000110111001001)$
3. $(0000110110110011) + (1111110111001001)$
4. $(1111110110110011) + (0000110111001001)$
5. $(0000110110110011) * (1111110111001001)$
6. $(1111110110110011) * (1111110111001001)$

Préciser lesquelles de ces opérations sont valides dans l'arithmétique ordinaire.

Exercice 8

1. Donner le réel dont la représentation IEEE 754 (codée en hexadécimale) est `C14C0000`.
2. Donner la représentation IEEE 754 32 bits des nombres $1,25$, $-13,3$, et $-2^{-5} - 2^{-8} - 2^{-10}$.
3. Calculer le plus petit nombre strictement positif représentable dans cette norme ? Et le plus grand ?

Exercice 9 On s'intéresse au test suivant : `( 0.9f - 0.8f ) == ( 0.8f - 0.7f )`.

1. Quelle est la valeur binaire de `0.9f` ?
2. Écrire en binaire `0.8f` et `0.7f`.
3. Calculer alors `0.9f - 0.8f` et `0.8f - 0.7f` en utilisant leurs représentations binaires.

Exercice 10 On veut travailler sur le codage des entiers **non signés**.

1. Écrire une fonction `decimaleVersBinaire` qui prend en argument une variable `n` de type `int`. Cette fonction doit d'abord vérifier que `n` est une valeur possible pour un entier (non signé) sur un octet, puis elle calcule et affiche les valeurs des huit bits `c0`, `c1`, `c2`, `c3`, `c4`, `c5`, `c6`, et `c7`.
2. Écrire une fonction `binaireVersDecimale` qui prend en argument un codage binaire (sous la forme de huit variables de type `int` par exemple). Cette fonction doit utiliser la méthode de Horner et renvoyer la valeur représentée par ce codage pour des entiers non signés (on suppose que chaque paramètre vaut `0` ou `1`).

Exercice 11 On veut travailler cette fois sur le codage des **entiers signés** en complément à deux. Reprendre l'exercice **10** et écrire les fonctions `decimaleVersBinaireSigne` et `binaireSigneVersDecimale` correspondantes.

Exercice 12 On veut travailler sur le codage des entiers **non signés** mais dans une base quelconque.

1. Écrire une fonction `decimaleVersBaseB` qui prend en arguments un entier `n` et une base `b`.
2. Écrire une fonction `baseBVersDecimale` qui prend en argument un codage sur plusieurs (huit par exemple) chiffres dans une base `b` donnée aussi en argument.

Exercice 13 Déterminer le nombre et le type des arguments utiles, puis écrire les fonctions de conversion `decimaleVersAvižienis` et `avižienisVersDecimale`.