Université de Montréal

Exercice noté 1- Programmation 1

par

Marie-Janne Blouin, 20159782

Cléo St-Martin, 20159519

Département de psychologie

Faculté des arts et des sciences

Travail présenté à Madame Alena Tsikhanovich

dans le cadre du cours IFT1016-B

Programmation 1

24 septembre 2019

1. **Conversions de bases :**

- **Déterminer la puissance de chaque chiffre pour un nombre de 5 chiffres en base 7**
  Nous avons choisi un nombre au hasard, par exemple le nombre 43210_7. Les puissances des chiffres seront donc les suivantes : $4*7^4 + 3*7^3 + 2*7^2 + 1*7^1 + 0*7^0 = 10738_{10}$
- **Convertir le nombre $2AA3_{16}$ en décimal**
  $2AA3_{16} = 2*16^3 + 10*16^2 + 10*16^1 + 3*16^0$
  $= 8192 + 2560 + 160 + 3$
  $= 10915_{10}$
- **Convertir le nombre $4B_{16}$**
  1. $4B_{16} \rightarrow$ Convertit en base 10 $= 4*16^1 + 11*16^0$
     $= 64 + 11$
     $= 75_{10}$

  2. $4B_{16} \rightarrow$ base 2 $= 4_{16} + B_{16} = 100_2 + 1011_2$
     $= 1001011_2$

  3. $4B_{16} \rightarrow$ Nous utiliserons le nombre en base binaire trouvé précédemment, soit 1001011_2 et nous allons regrouper les chiffres par 3 : 1_2 001_2 011_2
  $\rightarrow$ Nous allons convertir ces groupes de 3 chiffres en base binaire en chiffres en base octal :
  $1_2 = 1_8$
  $001_2 = 1_8$
  $011_2 = 3_8$

  Donc $4B_{16} = 113_8$

- **Comment peut-on encoder l'entier 1011_{10} avec la notation hexadécimale de JavaScript?**
  On convertit d'abord le nombre en base hexadécimale et on inscrit les symboles « 0x » devant le nombre en base 16.
  $1011_{10} \rightarrow$ Converti en base 16 $= 3*16^2 + 15*16^1 + 3*16^0 = 3F3_{16}$
  En JavaScript, $3F3_{16}$ sera donc noté 0x3F3. En entrant cette valeur dans *codeBoot*, on voit bel et bien que la valeur 1011_{10} est obtenue grâce à cette opération.

- $0xee = E*16^1 + 1E*16^0$
  $= 14*16^1 + 14*16$
  $= 238$

2. **Représenter 17_{10} selon la convention non signée sur 5 bits**
   - $17_{10} = 1*2^4 + 0*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 10001$

   Il est possible de confirmer que 17_{10} est bien 10001 selon la convention non signée grâce à la division successive. (On divise 17 par 2, ce qui donne 8. On soustrait donc 16 à 17 ce qui nous donne 1, qui correspond au reste de la division. Nous diviserons ensuite 8 par 2, et ainsi de suite, jusqu'à l'obtention d'un quotient de 0. Les restes de la division constitueront le nombre désiré, en écrivant d'abord le dernier reste obtenu et terminant par le premier reste obtenu)

3. **Quelles valeurs sont encodées par la convention complément à 2 sur 5 bits par les chaînes binaires suivantes :**
   - 01101 → Comme le nombre commence par 0, il est positif. Nous avons donc simplement besoin de faire la démarche suivante : $1*2^3 + 1*2^2 + 1*2^0 = 13_{10}$

   - 10011 → Comme le premier chiffre est 1, cela signifie que le nombre sera négatif, on doit donc trouver son complément :

   Trouver le complément :

```
 100000
- 10011
 -------
  01101
```

On obtient donc 01100_2 que nous allons transformer en base décimale. Nous avons rajouté le négatif devant la parenthèse, car nous avons déterminé précédemment que le nombre obtenu devait être négatif.

01101_2 → Base 10 → $-(1*2^0 + 1*2^2 + 1*2^3) = -13_{10}$

4. Quel est l'encodage en précision double IEEE 754 (64 bits) des nombres points flottants 3.15 et -4?

Anatomie d'un nombre à virgule flottante précision double IEEE 724 (64 bits) :

s (signe)	e(exposant)	f (fraction)
+ : 0 - : 1 Sur 1 bit	e1 + 1023_{10} Sur 11 bits	Ce que l'on retrouve après le point (ex : 1.f) Sur 52 bits

- 3.15_{10}
  - Conversion « 3 » → convertir en base binaire :
    $3_{10} = 1*2^1 + 1*2^0 = 11_2$
  - Conversion « 15 » → en base 2, on utilise la technique de multiplication successive :

0.15*2 = 0.30
0.30*2 = 0.60
0.60*2 = 1.20
0.20*2 = 0.40
0.40*2 = 0.80
0.80*2 = 1.60
…

Donc 0.15 en base 2 correspond à $0.001\overline{001}$
Alors 3.15_{10} correspond à 11.001001… donc si on normalise ce nombre, on obtient $1.1001001*2^1$

Dans ce cas, voici à quoi correspondent les champs :
  - s = 0 (car positif)
  - e = 1+1023 = 1024 = 2^{10} = 10000000000
  - f = $11001\overline{001}$ (totalisant 52 bits)
- -4_{10}
  - Conversion « 4 » → base binaire
    $4_{10} = 1*2^2 = 100_2$
  - Donc : $100.00 = 1.00*2^2$
  - s = 1 (car le nombre est négatif)
  - e = 2 + 1023 = 1025 = $2^{10} + 2^1$ = 10000000010
  - f = 0000…000 (totalisant 52 bits)

5.Trouvez la plus petite expression JavaScript (ayant le minimum de caractères, incluant les parenthèses et

symboles et pas de blanc) contenant les nombres 10, 2,3 et 4 (exactement une fois chaque) et les opérateurs +, - et * (autant de fois que vous le voulez), dont la valeur est de 9.

10 - 3 - 2 + 4 = 9

6.Lorsqu'on obtient un prêt hypothécaire à paiements fixes, on s'engage à rembourser le prêt sur un certain nombre de mois en payant le même montant à chaque mois. Si p est le montant du prêt, et que n est le nombre de mois, et que i est le taux d'intérêt annuel en pourcent, alors la formule mathématique suivante peut être utilisée pour calculer le m, le montant à payer chaque mois

Voir le deuxième document remis sur *Studium* intitulé Blouin_St-Martin_Exercice1.js