Université Paris 7- Denis Diderot
L1 d'Informatique — Examen partiel du 24 novembre 2012

Principes de fonctionnement des machines binaires

Durée : 3h. Les documents et les appareils électroniques de toutes sortes ne sont pas autorisés.

*Toutes vos réponses doivent être justifiées. Les exercices sont indépendants, et peuvent être traités dans un ordre quelconque.*
*Sauf indication contraire, tous les nombres sont donnés en base dix.*
*On trouvera en fin d'énoncé des rappels qui pourraient être utiles.*

## Exercice 1 :

Effectuez les changements de base suivants :

| Base 2 | Base 8 | Base 10 | Base 16 |
|---|---|---|---|
| $10011101_2$ | $235_8$ | $157_{10}$ | $9D_{16}$ |
| $1111101100_2$ | $1754_8$ | $1004_{10}$ | $3EC_{16}$ |
| $101101100001_2$ | $5541_8$ | $2913_{10}$ | $B61_{16}$ |
| $101001111100_2$ | $5174_8$ | $2684_{10}$ | $A7C_{16}$ |

On attend le calcul, présenté de manière claire, des valeurs permettant de remplir ce tableau.

$\overline{10}\ \overline{011}\ \overline{101}_2 = 235_8$

$10011101_2 = 1 + 4 + 8 + 16 + 128 = 157_{10}$

$\overline{1001}\ \overline{1101}_2 = 9D_{16}$

$1754_8 = \overline{1}\ \overline{111}\ \overline{101}\ \overline{100}_2$

$1111101100_2 = 4 + 8 + 32 + 64 + 128 + 256 + 512 = 1004_{10}$

$\overline{11}\ \overline{1110}\ \overline{1100}_2 = 3EC_{16}$

$2913_{10} = 2048 + 512 + 256 + 64 + 32 + 1 = 101101100001_2$

$\overline{101}\ \overline{101}\ \overline{100}\ \overline{001}_2 = 5541_8$

$\overline{1011}\ \overline{0110}\ \overline{0001}_2 = B61_{16}$

$A7C_{16} = \overline{1010}\ \overline{0111}\ \overline{1100}_2$

$\overline{101}\ \overline{001}\ \overline{111}\ \overline{100}_2 = 5174_8$

$101001111100_2 = 4 + 5 + 16 + 32 + 64 + 512 + 2048 = 2684_{10}$

## Exercice 2 :

**Question 1 :** Ecrire une table d'addition et une table de multiplication en base 7. En s'aidant de ces tables, effectuer la multiplication suivante : $6543 \times 4015$.

| $+$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 10 |
| 2 | 2 | 3 | 4 | 5 | 6 | 10 | 11 |
| 3 | 3 | 4 | 5 | 6 | 10 | 11 | 12 |
| 4 | 4 | 5 | 6 | 10 | 11 | 12 | 13 |
| 5 | 5 | 6 | 10 | 11 | 12 | 13 | 14 |
| 6 | 6 | 10 | 11 | 12 | 13 | 14 | 15 |

| $\times$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 0 | 2 | 4 | 6 | 11 | 13 | 15 |
| 3 | 0 | 3 | 6 | 12 | 15 | 21 | 24 |
| 4 | 0 | 4 | 11 | 15 | 22 | 26 | 33 |
| 5 | 0 | 5 | 13 | 21 | 26 | 34 | 42 |
| 6 | 0 | 6 | 15 | 24 | 33 | 42 | 51 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | 6 | 5 | 4 | 3 |
| $\times$ | | | | 4 | 0 | 1 | 5 |
| | | | 4 | 6 | 0 | 1 | 1 |
| | | | 6 | 5 | 4 | 3 | |
| | 3 | 6 | 1 | 3 | 5 | | |
| 3 | 6 | 3 | 1 | 2 | 4 | 4 | 1 |

**Question 2 :** En base 16, effectuer l'addition suivante : $B9624AF + 6698CB03$.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | B | 9 | 6 | 2 | 4 | A | F |
| $+$ | 6 | 6 | 9 | 8 | C | B | 0 | 3 |
| | 7 | 2 | 2 | E | E | F | B | 2 |

## Exercice 3 :

Soit $r$ le nombre rationnel dont la représentation en base 10 est $(49;5(6)^\omega)_{10}$.

**Question 1 :** Donner sa représentation en base 2.

$(49;5(6)^\omega)_{10} = 49;5 + \frac{2}{30} = 32 + 16 + 1 + \frac{1}{2} + \frac{1}{15}$ et $\frac{1}{15} = \frac{16}{15\ 16} = \frac{15+1}{15\ 16} = \frac{1}{16} + \frac{1}{15\ 16}$ donc $\frac{1}{15} = (0;(0001)^\omega)_2$ et $(49;5(6)^\omega)_{10} = (110001;1(0010)^\omega)_2$.

**Question 2 :** Donner sa représentation en machine en format `float`.

$(110001;1(0010)^\omega)_2$ a :
- 0 pour bit de signe ;
- 5 pour exposant, que l'on écrit donc sous la forme $5 + 127 = (10000100)_2$ ;
- 1000 1100 1000 1000 1000 1000 pour 24 premiers bits de mantisse, que l'on écrit donc 1000 1100 1000 1000 1000 100.

Sa représentation en tant que `float` est donc 01000010010001100100010001000100.

## Exercice 4 :

**Question 1 :** Donner la représentation du nombre $-173$ en `float`.

$173 = 128 + 32 + 8 + 4 + 1 = (10101101)_2$, donc $-173$ a
- 1 pour bit de signe ;
- 7 pour exposant, que l'on écrit donc $7 + 127 = (10000110)_2$ ;
- 0101 1010 0000 0000 0000 0000 pour 24 premiers bits de mantisse, que l'on écrit donc 0101 1010 0000 0000 0000 000.

Sa représentation en tant que `float` est donc 11000011001011010000000000000000.

**Question 2 :** Expliquer pourquoi seul l'exposant est modifié par la division de ce `float` par 128 pour la représentation du résultat de cette division, et en déduire cette représentation du résultat de la division de ce `float` par 128.

Le réel $r = (-1)^s 2^{e\ \ 127}(1; m)_2$ est représenté par le `float` $s\ e\ m$, donc le réel $\frac{r}{128}$ est représenté par le `float` $s$ $(e-7)$ $m$ : seul son exposant est modifié. En particulier, $\frac{173}{128}$ est représenté par le float 10111111101011010000000000000000.

**Question 3 :** On définit le codage suivant pour les fractions de la forme $\frac{p}{q}$. Soit `pp` la représentation en `short` de $p$ et `qq` celle de $q$. On code alors la fraction $\frac{p}{q}$ par la concaténation des représentations en short de $p$ puis $q$, c'est-à-dire `ppqq`. Donner la représentation de la fraction $\frac{173}{128}$ avec ce codage.

En tant que `short`, $-173$ est représenté par 1111111101010011 et 128 est représenté par 0000000010000000. La fraction $\frac{173}{128}$ est donc représentée, dans notre codage, par 11111111010100110000000010000000.

**Question 4 :** Donner un exemple de fraction qui soit représentable dans ce codage mais qui *ne* soit *pas* représentable sans perte de précision dans le type `float`.

$\frac{1}{3}$ est représenté par 00000000000000010000000000000011 dans notre codage. Toutefois, seuls *certains* nombres de la forme $2^a b$, où $a$ et $b$ sont des entiers relatifs, peuvent être représentés sans perte de précision dans le type `float` : $\frac{1}{3}$ n'est pas un tel nombre, donc n'est pas représentable sans perte de précision dans le type `float`.

**Question 5 :** Donner un exemple de fraction irréductible qui *ne* soit *pas* représentable dans ce codage mais qui soit représentable sans perte de précision dans le type `float`.

$\frac{2^{17}}{1}$ est représenté par le `float` 01001000000000000000000000000000 sans perte de précision. Toutefois, $2^{17}$ ne peut pas être représenté sous forme de `short` (seuls les entiers compris entre $-2^{15}$ et $2^{15} - 1$ le peuvent), donc $\frac{2^{17}}{1}$ n'est pas représentable dans notre codage.

## Exercice 5 :

**Question 1 :** Soit un code binaire de longueur fixe codant les lettres minuscules et majuscules de notre alphabet (donc codant 52 caractères en tout). Quelle longueur au minimum le code doit-il faire ?

Un code de longueur $L$ permet de coder au plus $2^L$ caractères. Puisque $2^5 < 52 < 2^6$, nul code de longueur $L \leq 5$ ne peut coder les lettres minuscules et majuscules, mais certains codes de longueur $L = 6$ le peuvent. Notre code doit donc avoir une longueur minimum $L = 6$.

**Question 2 :** Ecrire en code ASCII le texte suivant : *Bond007*

On réécrit les lettres une par une pour obtenir le code suivant :

$$\overbrace{01000010}^{B}\ \overbrace{01101111}^{o}\ \overbrace{01101110}^{n}\ \overbrace{01100100}^{d}\ \overbrace{00110000}^{0}\ \overbrace{00110000}^{0}\ \overbrace{00110111}^{7}$$

## Exercice 6 :

On scanne une image ayant 24 cm de large et 13,5 cm de hauteur à 300 dpi en noir et blanc.

**Question 1 :** Poser l'opération qui permet de calculer le poids de cette image. Faire le calcul, sachant qu'un pouce $\simeq 2{,}54$ cm (on pourra se contenter d'une valeur approchée).

L'image contient $300 \times \frac{24}{2{,}54} \simeq \frac{300 \ \ 24}{2{,}5} = 2880$ colonnes de pixels et $300 \times \frac{13{,}5}{2{,}54} \simeq \frac{300 \ \ 13{,}5}{2{,}5} = 1620$ lignes de pixels. Chaque pixel est codé sur 1 bit. Le poids de l'image est donc de $\frac{300 \ \ 24 \ \ 300 \ \ 13{,}5 \ \ 1}{2{,}54 \ \ 2{,}54} \simeq 2880 \times 1620 \simeq 3000 \times 1500 = 4500000$ bits, soit environ $\frac{4500000}{8 \ \ 1024} \simeq 550$ kO.

**Question 2 :** On désire l'afficher sur un écran 27 pouces de format 16/9 ayant une définition de $2560 \times 1440$ pixels. La résolution choisie pour scanner l'image est-elle suffisante ?

Notre image contient plus de colonnes que l'écran ne peut en afficher, et plus de lignes que l'écran ne peut en afficher. La résolution du scanner est donc suffisante.

**Rappels :**

On rappelle que les types `byte`, `short` et `int` codent les entiers sur 8, 16 et 32 bits respectivement, en utilisant la convention du complément à 2 pour la représentation des nombres négatifs.

On rappelle la représentation en machine du type `float` : 1 bit pour le signe, 8 bits pour l'exposant (valeur de l'exposant vrai augmentée de 127) et 23 bits pour la mantisse.

Le standard ASCII code les caractères alphanumériques de 0 à 127. Les caractères sont alignés sur 8 bits. On y trouve notamment :

- de 48 à 57 : les chiffres de 0 à 9
- de 65 à 90 : l'alphabet des majuscules (A à Z)
- de 97 à 122 : l'alphabet des minuscules (a à z).