Principes de fonctionnements des machines binaires (PF1)
Examen du mercredi 13 janvier 2010
Seul document autorisé : une feuille manuscrite A4 recto/verso.
Ni calculette, ni téléphone.

Les exercices sont indépendants et peuvent être traités dans un ordre quelconque
Toutes les réponses devront être justifiées

Exercice 1.

Soit le nombre dont la représentation en base usuelle (la base dix) est `3981`.
1.1. Donner ses représentations en base deux, quatre, huit et seize.
1.2. Parmi les types entiers de Java, lesquels permettent de le stocker ?
1.3. Pour le plus petit de ces types, quelle est la représentation du nombre `-3981` ?
1.4. Soit la séquence Java suivante :

```
int m = 3981, n = -3981;
int and = m & n;
int or = m | n;
int xor = m ^ n;
```

a) Que contiennent les zones mémoire associées à `m`, `n`, `and`, `or` et `xor`. Vous pouvez exprimer le contenu des bits de ces zones sous forme binaire, octale ou hexadécimale, au choix.

b) Qu'afficheraient les instructions

```
Deug.println(and);
Deug.println(or);
Deug.println(xor);
```

1.5. On rappelle tout d'abord que dans la partie exposant (8 bits) d'un nombre flottant, la valeur qui est stockée est la valeur de l'exposant vrai augmentée de la valeur (en base dix) 127.

a) Soit la séquence suivante :

```
float a, b, c, d;
a = 3981 + 1.0f/2 + 1.0f/8 + 1.0f/16 + 1.0f/128;
b = -a;
c = 1.0f/(1024*4);
d = 1.0f/(1024*16);
```

Que contiennent les zones mémoire associées à `a`, `b`, `c` et `d` (vous pouvez exprimer le résultat en base deux, huit ou seize au choix) ?
b) Que contiennent les zones associées à `e` et `f` si, après la séquence précédente, on exécute

```
float e, f;
e = a + c;
f = a + d;
```

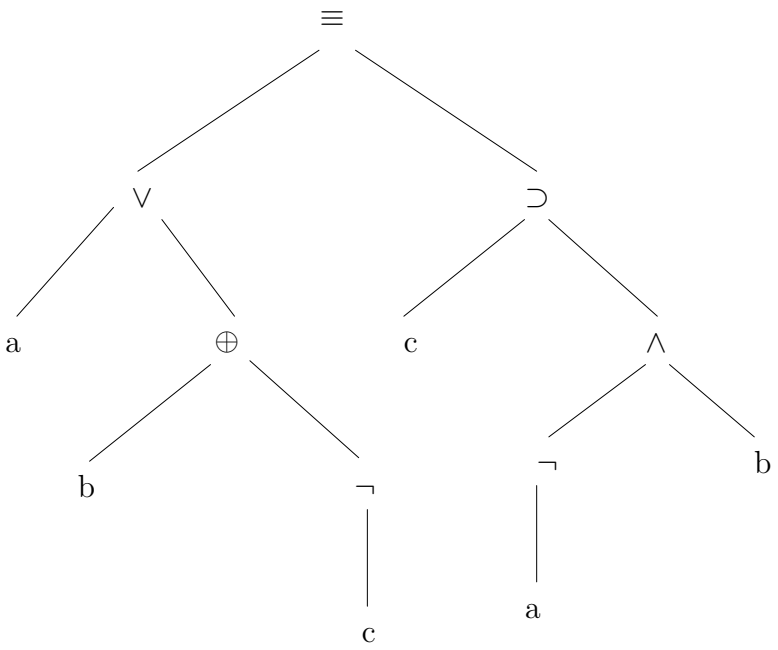
Exercice 2

2.1. Donner la table de la fonction de 4 variables a, b, c et d qui est vraie si le nombre qui s'écrit $abcd$ en base deux est différent de quinze et est un multiple de quatre ou de cinq ou de six ou de sept.
2.2. Donner la forme normale disjonctive de cette fonction.
2.3. En donner la forme simplifiée au moyen de la méthode de Karnaugh.

Exercice 3

Dans cet exercice, les symboles $\neg$, $\vee$, $\wedge$, $\oplus$, $\supset$ et $\equiv$ représentent respectivement la négation, la disjonction (OU), la conjonction (ET), le OU exclusif (XOR), l'implication et l'équivalence.

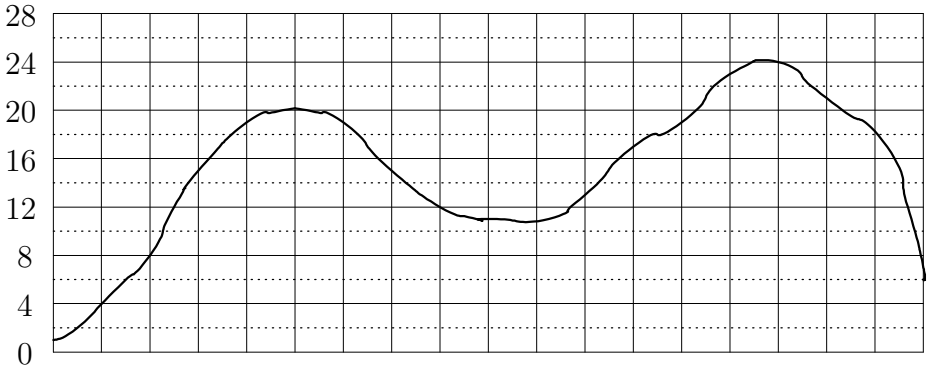
On considère l'arbre suivant :



3.1. Donner les formes polonaises préfixée et postfixée (suffixée) de l'expression qui lui correspond.
3.2. Donner la forme parenthésée en notation infixe usuelle de cette expression (la seule hypothèse qu'on fera est que la négation est prioritaire par rapport aux autres opérateurs).
3.3. Construire la table de vérité de cette expression.

Exercice 4

Quelle est la suite de bits obtenue en numérisant la courbe donnée avec l'échantillonnage et la quantification sur 8 valeurs définis par le quadrillage. La valeur de chaque échantillon sera arrondie à la valeur de quantification la plus proche (on s'aidera pour cela des traits pointillés) et le nombre de bits utilisé sera minimal.



Exercice 5

Dans les calculs de cet exercice, on considérera qu'un pouce correspond à $2,5$ cm.

On considère une photo de $12,5$ cm $\times 7,5$ cm.
On scanne cette photo en 16 millions de couleurs.

Pour chacune des deux résolutions 100 et 300 dpi, donner :
a) les dimensions de l'image numérique obtenue exprimée en pixels ;
b) le poids de l'image numérique obtenue ;
c) les dimensions minimales d'un écran de résolution 75 dpi nécessaires pour son affichage.

Exercice 6

6.1. Qu'appelle-t-on adressages immédiat, direct, basé/indexé, indirect ?

6.2. Quel est l'intérêt et pourquoi est-il nécessaire de disposer d'autres types d'adressage que les adressages immédiat et direct ?

6.3. Soit une machine dont les mots ont 4 octets et dont le contenu (partiel de la mémoire) est :

adresse (sous forme décimale)	contenu (sous forme décimale)
400	2000
404	412
408	20
412	3000
416	305

On considère le programme suivant (les commentaires indiquent l'effet des instructions) :

```
LOAD R0 400 // chargement du registre R0 en adressage direct
LOAD R1 #1002 // chargement du registre R1 en adressage immédiat
LOAD R2 #100 // chargement du registre R2 en adressage immédiat
ADD R0 R1 // ajouter au contenu de R0 celui de R1
OPP R1 // changer le signe du contenu de R1
ADD R1 @404 // ajouter à R1 une valeur obtenue par adressage indirect
STORE R1, 900 (R2) // stocker le contenu de R1 à une adresse indexée par le registre R2
```

Après exécution de cette séquence, que contiendra le mot mémoire d'adresse 1000 ?