

# Algorithmes arithmétiques

## Examen

Florent de Dinechin

26 mai 2008

Tous documents autorisés.

Il est demandé de répondre sur la copie (le crayon à papier est accepté).

La taille des boîtes donne une idée de celle de la réponse attendue.

Utiliser si nécessaire des feuilles supplémentaires.

La plupart des questions sont indépendantes.

Le barème donné est indicatif, d'ailleurs il dépasse 20.

L'examen dure 3 heures.

# 1 La preuve par 9

Voici une technique que l'on m'a apprise quand j'étais petit pour vérifier les opérations que je calculais à la main.

Soit pour tout entier  $n$  l'entier  $V(n) \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$  obtenu récursivement ainsi : on additionne les chiffres décimaux de  $n$ . Si la somme est 9 on la remplace par 0. Si la somme est plus grande que 10, donc s'écrit sur plus d'un chiffre, on recommence : on additionne les chiffres de cette somme, etc.

La recette, dite de *preuve par 9*, est la suivante : pour  $\circ \in \{+, -, \times\}$ , une fois calculé  $r = a \circ b$ , calcule  $V(r)$ ,  $V(a)$  et  $V(b)$ . Si tu n'as pas  $V(r) = V(V(a) \circ V(b))$ , c'est que tu t'es planté.

**Question 1-1 (2 pt)** Maintenant que vous êtes grands, prouvez la preuve par 9.

Q1-1

**Question 1-2 (1 pt)** Généralisez à une base  $\beta$  quelconque. Cela marche avec un ensemble de chiffres redondants ?

Q1-2

**Question 1-3 (1 pt)** Je ne me souviens pas si on m'avait appris la preuve par 9 pour vérifier les divisions que je posais. Peut-on l'utiliser? Si oui, comment, si non, pourquoi.

Q1-3

**Question 1-4 (1 pt)** Suggérez des applications de techniques similaires dans le domaine de l'arithmétique des ordinateurs (question ouverte/bonus).

Q1-4

## 2 Arithmétique binaire complexe

Knuth a proposé en 1960 un système de numération de position en base  $\beta = 2i$ , où  $i$  vérifie  $i^2 = -1$ , et où les chiffres peuvent prendre les valeurs 0, 1, 2, ou 3.

**Question 2-1 (1 pt)** Remplissez le tableau suivant, et généralisez.

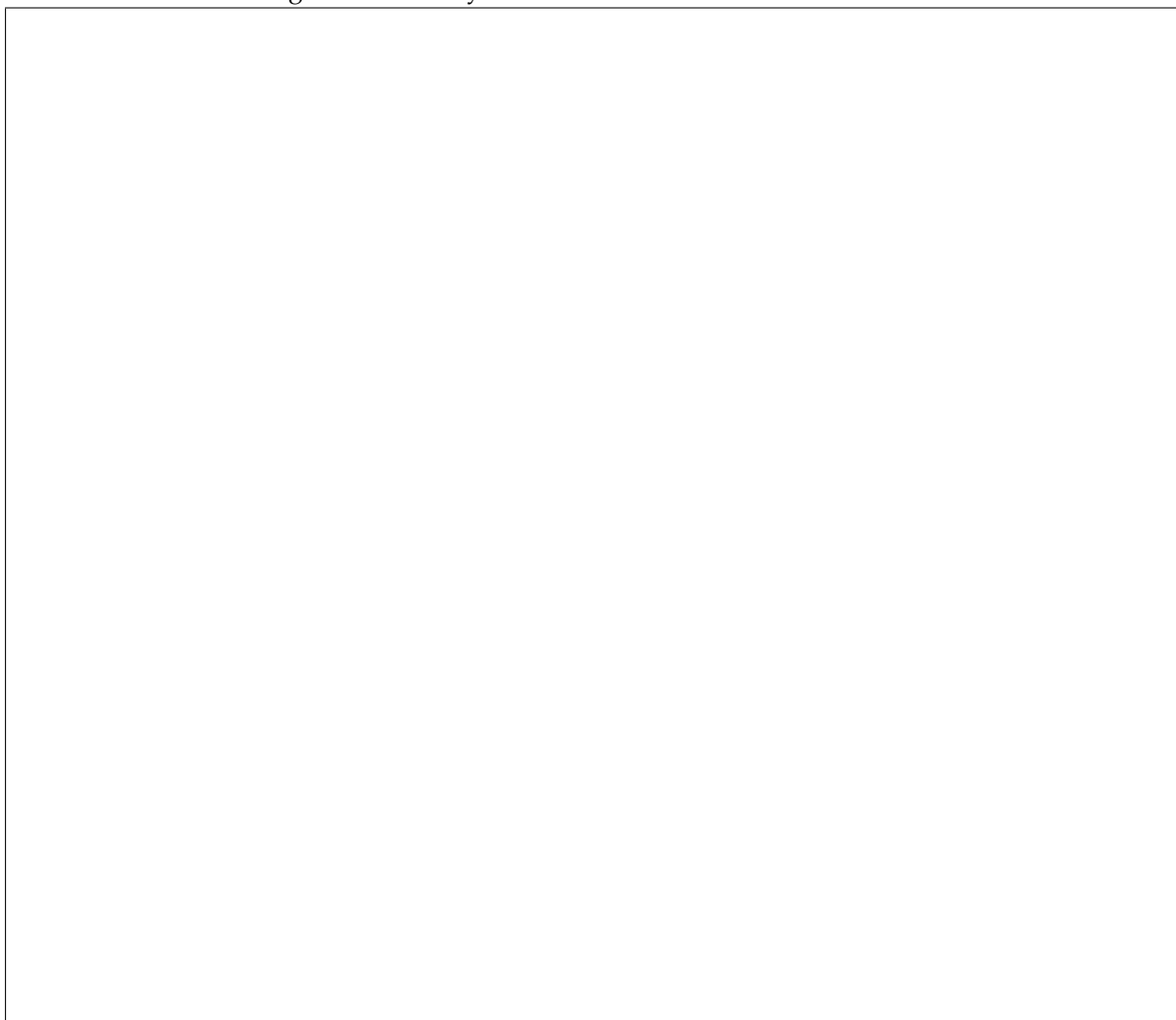
$i$	3	2	1	0	-1	-2	-3
$\beta^i$							

Q2-1

**Question 2-2 (1 pt)** Quelles sont les écritures de 0, 1,  $-1$ ,  $i$ , 17?

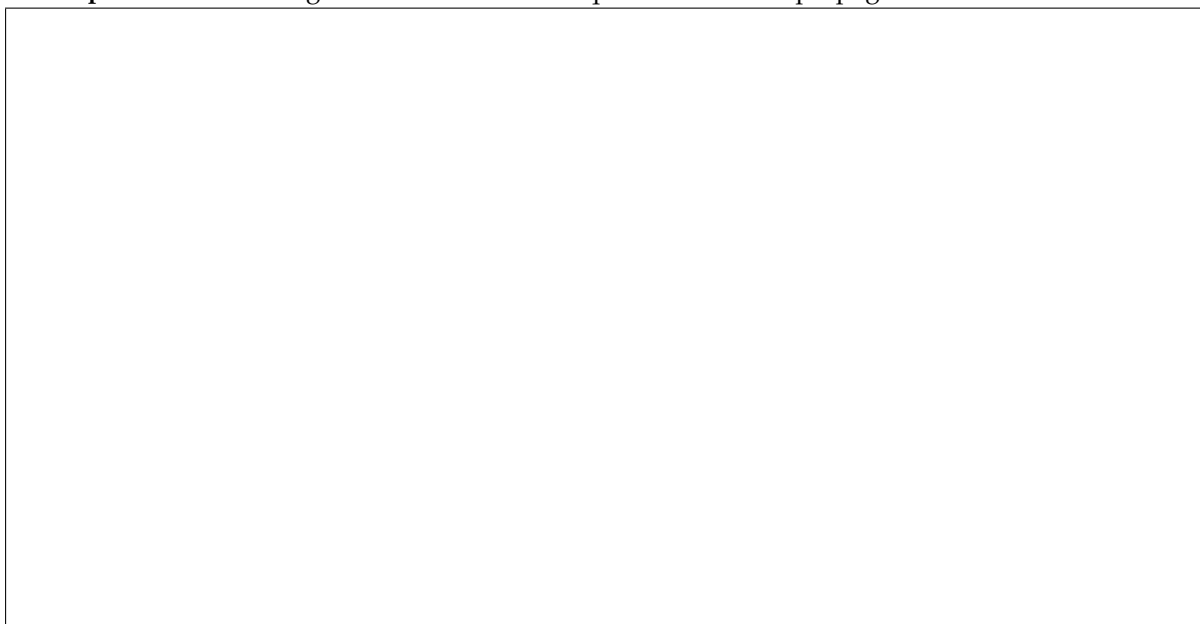
Q2-2

**Question 2-3 (2 pt)** Montrez que tout complexe de la forme  $a + ib$  avec  $a, b \in \mathbb{Z}$  peut s'écrire dans ce système sans avoir besoin d'un signe. Est-ce un système redondant ?



Q2-3

**Question 2-4 (2 pt)** Donnez un algorithme d'addition. A quoi ressemble la propagation de retenue ?



Q2-4

Aoki, Amada et Higuchi préfèrent utiliser les chiffres  $\{-3, -2, -1, 0, 1, 2, 3\}$ , avec la même base 2i.

**Question 2-5 (1 pt)** Comparez avec un système qui consisterait à coder séparément la partie réelle et la partie imaginaire en base 4 avec le même ensemble de chiffres.

Q2-5

### 3 Arithmétique décimale

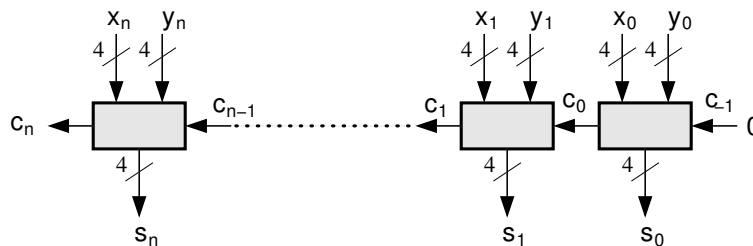
On représente un nombre en *décimal codé en binaire* (ou BCD) : chaque chiffre décimal est codé sur 4 bits. En général, dans toute la suite, un "chiffre" désignera un chiffre décimal.

**Question 3-1 (1 pt)** Quelle est la proportion de codes utiles dans cette représentation ? Quel codage du décimal permet de l'améliorer ?

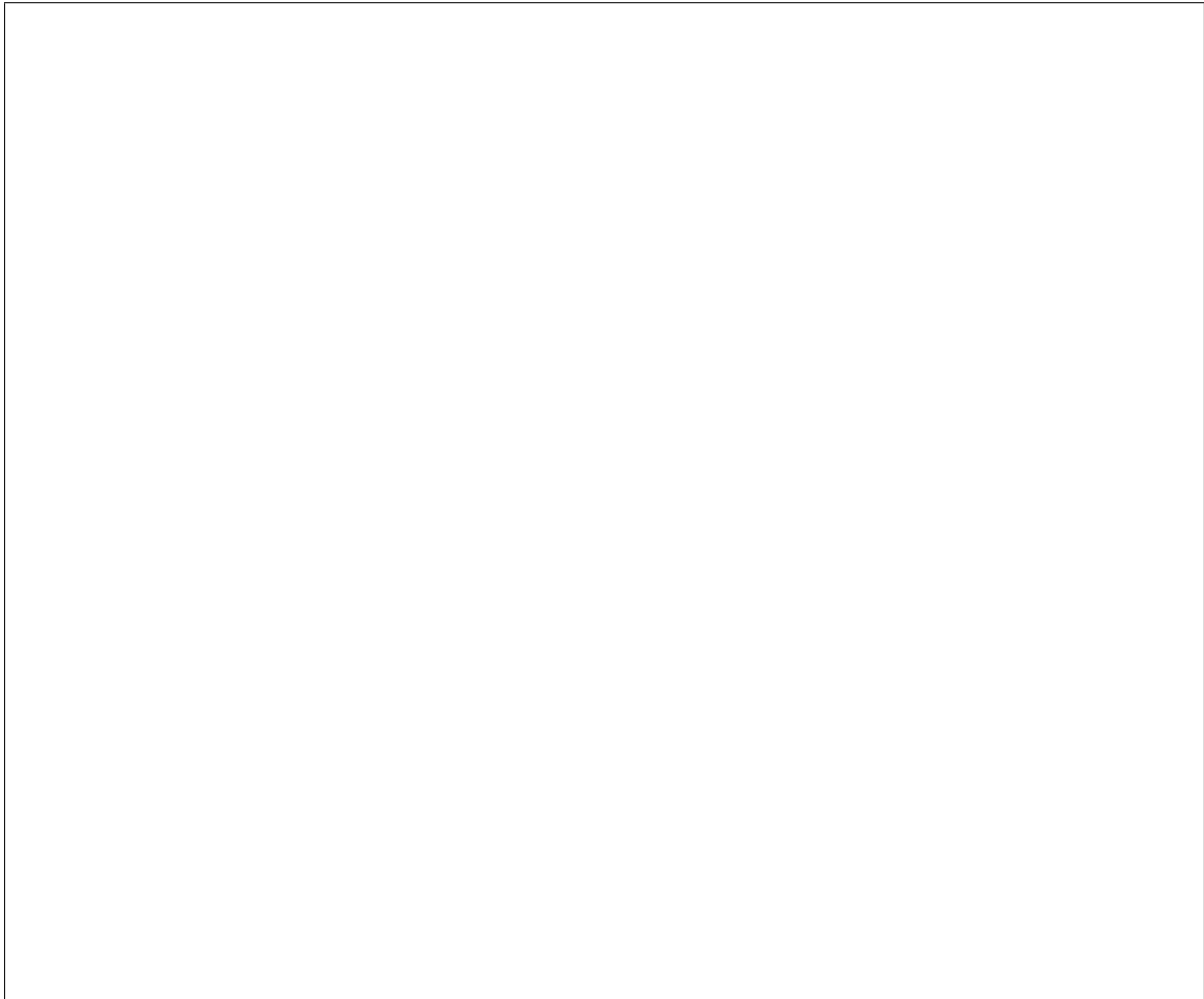
Q3-1

#### 3.1 Addition et soustraction

On va construire l'additionneur à propagation de retenue représenté sur la figure suivante.



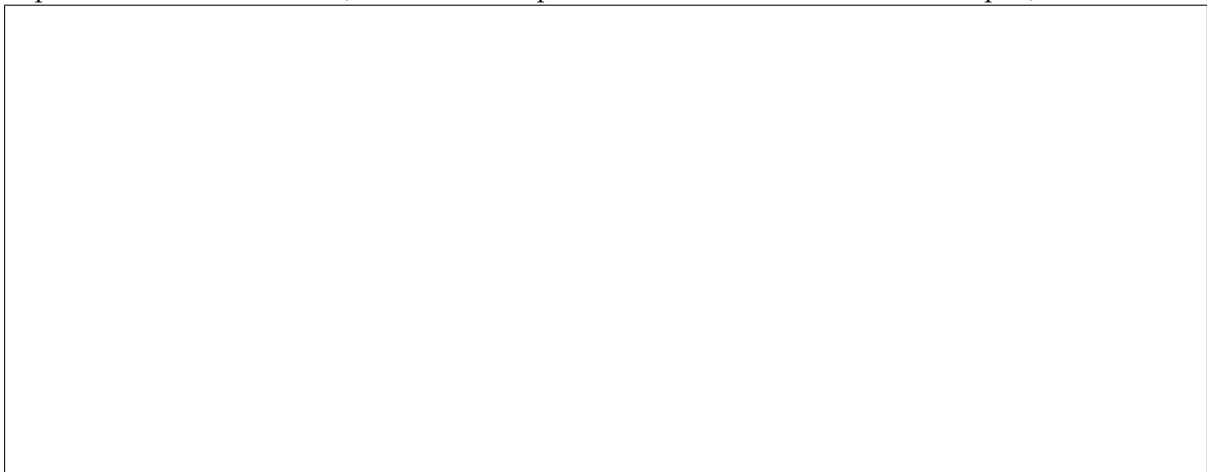
**Question 3-2 (2 pt)** Construisez une des cellules d'addition décimale représentée sur cette figure, au moyen de deux petits additionneurs binaires (si nécessaire avec leurs retenues entrantes et sortantes), d'un multiplexeur<sup>1</sup>, et puis c'est tout. Justifiez le bon fonctionnement, en particulier montrez que le chiffre résultat est bien toujours compris entre 0 et 9. Vous pourrez étiqueter les signaux intermédiaires avec l'intervalle d'entiers qu'ils peuvent porter.



Q3-2

<sup>1</sup> Un multiplexeur est une boîte à trois entrées,  $x_0$ ,  $x_1$ , et  $s$  et une sortie  $y$ .  $s$  est de type booléen,  $x_0$ ,  $x_1$ , et  $y$  sont de types quelconques mais identiques entre eux. Le multiplexeur réalise la fonction : si  $s = 0$  alors  $y = x_0$  sinon  $y = x_1$ .

**Question 3-3 (1 pt)** Le code Excess-3 a été proposé pour remplacer le codage BCD. Les 4 bits codent alors en binaire 3 plus le chiffre : 0 est codé par 0011, 1 est codé par 0100, etc. Trouvez une utilité pour implémenter la soustraction (on ne demande pas de construire un soustracteur complet).



Q3-3

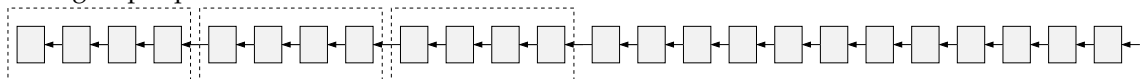
### 3.2 Conversions

**Question 3-4 (1 pt)** Donnez un algorithme itératif qui convertit en binaire un nombre codé en BCD. On ne fera intervenir que la multiplication par 10 et des additions binaires. Comment construit-t-on un multiplieur par 10?

Q3-4

La conversion d'un nombre en binaire sur  $4k$  chiffres vers un nombre en base 16 sur  $k$  chiffres est triviale, il suffit de grouper les bits par 4. Voici un algo itératif qui fait la même chose bit à bit. Oui, il est trivial aussi.

- On place côte-à-côte deux registres à décalage de  $4k$  bits, celui de gauche ayant ses bits moralement groupés par 4.



- Dans celui de droite on place le mot binaire à convertir
- On le décale bit à bit vers la gauche. Entre chaque décalage, on ne fait rien.
- Après  $4k$  cycles, le registre de gauche contient le nombre codé en base 16.

**Question 3-5 (3 pt)** Remplacez le "on ne fait rien" ci-dessus par autre chose pour obtenir un algorithme de conversion d'un entier binaire en BCD. Justifiez bien. Quelle est la taille du registre de gauche? Dessinez l'état des registres au cours des itérations de la conversion de 63 en BCD, en commentant les opérations effectuées à chaque décalage.

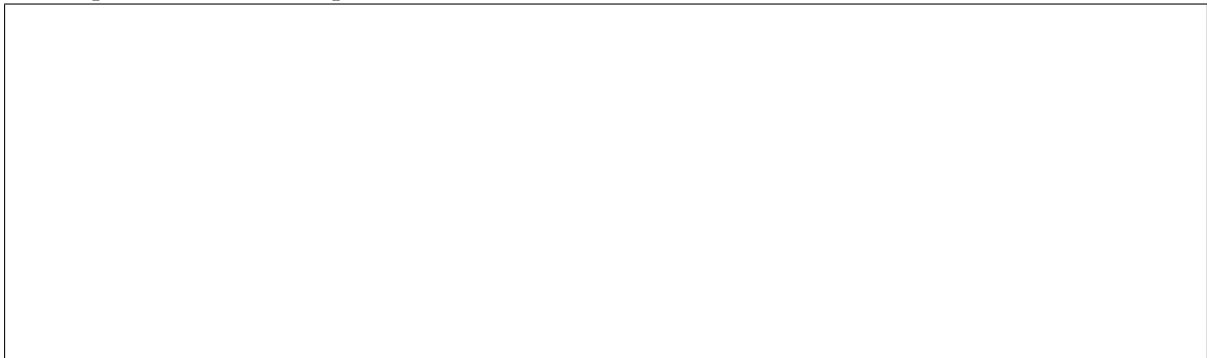
Indice : cet algo est connu dans la littérature sous le nom de *shift and add-3*.

Si vous le généralisez à d'autres bases que 10 (lesquelles) vous aurez des points de bonus.



Q3-5

**Question 3-6 (2 pt)** Saurez-vous modifier l'algo précédent pour qu'il calcule la division euclidienne d'un entier binaire par 10 (donnant un quotient et un reste tous deux en binaire)? Question sans doute difficile.



Q3-6

## 4 Arithmétique flottante

Pour rappel, en double précision, l'exposant E est un entier codé sur 11 bits, et la partie fractionnaire F de la mantisse est codée sur 53 bits.

On considère un morceau de code qui prend en entrée deux nombres entiers entre -1000 et 1000, les *caste* dans des variables flottantes double-précision a et b en s'assurant que  $a > b$ , puis calcule, tout en flottant :

$$0.125 \times \left( 1 - \frac{a}{\sqrt{a \times a + b \times b}} \right)$$

**Question 4-1 (2 pt)** Il y a plein d'opérations exactes là-dedans, lesquelles et pourquoi ?

Q4-1