

## Architecture et système – Examen

**Durée : 3 heures, tous documents autorisés, calculatrices, ordinateurs et téléphones portables interdits.**

1<sup>er</sup> juin 1999

**Avertissement.** Bien que le sujet ne comporte que deux problèmes, de nombreuses questions sont indépendantes.

### 1 Système

La commande dos `set` affiche, définit ou supprime des variables d'environnement de MS-DOS:

**Question 1.** *Quels effets produisent les commandes suivantes<sup>1</sup> ?*

```
set
set a=
set a=c:\windows
```

**Question 2.** *Quelle commande dos faut-il taper pour connaître l'affectation de la variable dos `a` ?*

Pour la suite, on souhaite refaire en ADA les commandes système issues de la manipulation de variables d'environnement (ci-dessus); pour cela, on dispose des fonctions et procédures suivantes:

- dans le package `Ada.Command_Line.Environment`, des fonctions :
  - fonction `Environment_Value` (`Number` : in `Positive`)  
`return String`; qui, pour un entier `i`, fournit la valeur de l'affectation de la  $i^{\text{e}}$  variable d'environnement.  
*Exemple.* `Environment_Value (i)` retourne `vari=valeuri`
  - fonction `Environment_Count` `return Natural`; qui fournit le nombre de variables d'environnement affectées

---

1. tapées juste après le prompt (ou invite)

- dans le package GNAT.OS\_Lib, de
  - la fonction `function Getenv (Name : String) return String` qui, pour une chaîne `ch`, fournit la valeur de l'affectation de la variable d'environnement `ch` ;
  - la procédure `procedure Setenv (Name,value : String)` qui, pour deux chaînes `nm` et `val`, admet comme effet de bord l'affectation de la variable d'environnement `nm` avec la valeur `val` ;
  - la fonction `function Getwd return String` qui fournit le nom du répertoire courant ;
  - la procédure `procedure Setwd (Name : String)` qui, pour une chaîne `nm` admet comme effet de bord le changement de répertoire `nm`.
- dans le package Ada.Command\_Line, des fonctions :
  - fonction `Argument (Number : in Positive) return String`; qui, pour un entier `i`, fournit la valeur du  $i^{\text{e}}$  paramètre de la ligne de commande MS-DOS.
  - fonction `Argument_Count return Natural`; qui fournit le nombre de paramètres de la ligne de commande MS-DOS.

Les fonctions de lecture et d'affichage de chaînes sont bien sûr définies dans le package Ada.Integer\_Text\_IO.

**Question 3.** *Écrivez une commande système `list-set` qui remplace celle du dos permettant d'afficher la liste et le contenu des variables d'environnement. On demande d'écrire le programme complet.*

**Question 4.** *Écrivez une commande système `voir` qui remplace celle du dos permettant d'afficher le contenu de la variable d'environnement dont le nom est passé en paramètre sur la ligne de commande. On demande d'écrire le programme complet.*

**Question 5.** *Modifiez le programme de la question précédente pour qu'il accepte plusieurs variables en arguments.*

**Question 6.** *Écrivez deux commandes système : `retiens` qui permet d'affecter le contenu d'une variable d'environnement avec le nom du répertoire courant. Et une autre, `retour`, qui permet de revenir à tout moment dans le répertoire préservé par la commande `retiens`. On demande d'écrire les programmes complets.*

## 2 Problème : code DCB

### 2.1 Codage

Les ordinateurs ne savent manipuler que des données binaires. Par contre, les hommes expriment en général les nombres en décimal. Lors d'un calcul interactif avec un ordinateur, le processus est alors le suivant :

1. écriture des opérandes en base 10 ;
2. conversion vers la base 2 ;
3. calcul en base 2 ;
4. conversion du résultat vers la base 10 ;
5. affichage du résultat.

On remarque qu'à chaque étape de calcul, il y a trois conversions de base nécessitant, pour passer de la base 10 vers la base 2, une série de divisions euclidiennes, et pour passer de la base 2 vers la base 10, une série de multiplications. Ces conversions sont très coûteuses en temps de calcul.

Pour répondre à ces problèmes, un autre système de codage des nombres a été introduit : le DCB ou décimal codé en binaire. Nous allons nous y intéresser dans ce problème, et en particulier voir comment additionner deux nombres codés en DCB.

L'idée du codage DCB est de coder chaque chiffre décimal en binaire et de considérer le nombre écrit comme la suite de ces codes. Supposons par exemple qu'on code chaque chiffre binaire sur un octet, le nombre  $538_{(10)}$  s'écrirait alors 00000101 00000011 00001000.

On veut coder les nombres en DCB avec le code le plus compact possible.

**Question 7.** *Quel est le nombre de bits minimal nécessaire pour coder en DCB un chiffre décimal quelconque ? Justifiez votre réponse.*

Dans toute la suite, quand on parlera de codage DCB, on codera chaque chiffre décimal sur ce nombre minimal de bits.

**Question 8.** *Pour chacun des nombres décimaux suivants donnez leur écriture décimale, leur écriture DCB et l'écriture hexadécimale de ce codage DCB :*

$$\begin{aligned}764_{(10)} &= \dots_{(DCB)} = \dots_{(16-DCB)} \\925_{(10)} &= \dots_{(DCB)} = \dots_{(16-DCB)} \\ \dots_{(10)} &= 1001001101011000_{(DCB)} = \dots_{(16-DCB)} \\ \dots_{(10)} &= 10011010010000_{(DCB)} = \dots_{(16-DCB)} \\ \dots_{(10)} &= \dots_{(DCB)} = 1434_{(16-DCB)} \\ \dots_{(10)} &= \dots_{(DCB)} = 991_{(16-DCB)}\end{aligned}$$

**Question 9.** *Quelle remarque peut-on faire sur l'écriture hexadécimale d'un nombre codé en DCB ?*

## 2.2 Addition 1 chiffre

On a à notre disposition un additionneur 4 bits classique avec retenues, entrante et sortante, et les portes logiques de base. Le but est de construire un additionneur 1 chiffre DCB à partir de ces composants.

**Question 10.** *Construisez une table contenant pour chaque chiffre décimal les informations suivantes :*

- le résultat de l'ajout de 1 à la représentation DCB de ce chiffre calculé par l'additionneur binaire classique ;
- le résultat de la même opération en DCB ;
- la différence (DCB - binaire) entre ces deux résultats (interprétés tous deux en binaire).

chiffre	+1 binaire	+1 DCB	différence
$0_{(10)}$	$1_{(2)}$	$1_{(DCB)}$	$0_{(10)}$
...	...	...	...
$9_{(10)}$	...	...	...

**Question 11.** *Quelles sont les valeurs de la différence entre la somme DCB et la somme binaire de deux chiffres DCB ? Plus précisément quand le résultat des deux sommes est-il différent et que vaut la différence dans ce cas ?*

**Question 12.** *Dessinez un circuit qui va détecter quand la somme calculée par un additionneur 4 bits est fautive (c'est-à-dire lorsque la différence calculée à la question précédente est non nulle). Ce circuit a comme entrées les 4 chiffres de la somme et la retenue sortante de l'additionneur. Sa sortie vaut 1 s'il y a erreur et 0 sinon.*

*Indication.* Considérez séparément la retenue sortante et les chiffres du résultat. Vous pouvez également passer par la table de vérité de ce circuit.

**Question 13.** *En utilisant le circuit de la question précédente et deux additionneurs 4 bits dessinez un additionneur DCB qui prend deux chiffres DCB et une retenue en entrée et qui produit un chiffre DCB et une retenue en sortie.*

## 2.3 Additionneur

**Question 14.** *Comment construire un additionneur DCB 4 chiffres à partir d'additionneurs DCB 1 chiffre ? Dessinez le circuit obtenu (sans détailler l'additionneur 1 chiffre).*

**Question 15.** *Calculez l'addition suivante en DCB et en faisant apparaître les retenues et les détections et corrections d'erreurs :  $3782 + 2635$ .*

*Note.* Le calcul de base est l'addition binaire classique.

## 2.4 Assembleur Z80

Le microprocesseur Z80 possède des dispositifs permettant le calcul en DCB. Il s'agit de l'instruction DAA (Decimal Adjust Accumulator) et du bit H (Half-carry) du registre indicateur. Ce bit indique s'il y a eu une retenue entre le 4<sup>e</sup> et le 5<sup>e</sup> bit lors d'une addition. Le rôle de l'instruction DAA est de remettre sous forme DCB le résultat d'une addition de nombres en format DCB faite avec l'instruction ADD.

**Question 16.** *Pourquoi a-t-on besoin de l'indicateur H pour faire fonctionner l'instruction DAA ?*

**Question 17.** *Dessinez un circuit qui réalise l'instruction DAA. Vous pouvez utiliser un additionneur 8 bits.*

**Question 18.** *Reconnaissez et expliquez le fonctionnement de l'algorithme suivant. Comparez son temps d'exécution et sa taille en octets avec le même calcul utilisant DAA.*

```
ADD A,66h
LD C,A
ADD A,B
LD D,F
LD E,A
LD A,C
XOR B
XOR E
NEG
AND 11h
RLCA
LD C,A
RLCA
ADD A,C
SUB E
CPL
LD F,D
```

## 2.5 Soustraction

**Question 19.** *Proposez un codage des nombres négatifs permettant d'effectuer la soustraction en utilisant l'additionneur DCB. Justifiez votre réponse. Quel serait le codage de -735 (sur 32 bits) dans votre système ?*