Examen (Sujet D)
Circuits et architecture des ordinateurs

— *Master d'informatique* —

Janvier 2014, durée 2h30.

L'examen se compose de quatre exercices indépendants. Les seuls documents autorisés sont une feuille de mémento (double A4). Tous les appareils électroniques sont interdits à l'exception des montres.

▶ **Exercice 1** On considère la table de vérité ci-dessous qui donne la valeur de la sortie S en fonction des trois entrées binaires E_0, E_1 et E_2.

E_2	E_1	E_0	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

a) Donner une formule booléenne exprimant S en fonction de E_0, E_1 et E_2.
b) Donner un circuit avec des portes OR, AND et NOT pour calculer S à partir de E_0, E_1 et E_2. Est-il possible d'éviter les portes NOT ?
c) Donner un circuit avec uniquement des portes NOR.

▶ **Exercice 2** On considère le programme suivant écrit en assembleur LC-3.

```
          .ORIG x3000
mystere:  LD R0,value
          JSR myst
          TRAP x25                ; HALT
value:    .FILL 21

myst:     ADD R6,R6,-1
          STR R1,R6,0
          AND R1,R1,0
          AND R0,R0,R0
          BRz fini
          BRp bit0
bit1:     ADD R1,R1,1
bit0:     ADD R0,R0,R0
          BRp bit0
          BRn bit1
fini:     ADD R0,R1,0
          LDR R1,R6,0
          ADD R6,R6,1
          RET
          .END
```

a) Donner le contenu en binaire du registre R0 lorsque le programme atteint l'instruction TRAP.

b) Donner le contenu du registre R0 lorsque le programme atteint l'instruction `TRAP` si la valeur chargée dans le registre R0 à la première instruction était respectivement 0, 1 et 3.
c) Que calcule la routine `myst` quand la valeur contenue au départ dans R0 est positive ?

▶ **Exercice 3** Le but de cet exercice est d'écrire une routine `asc2int` qui convertit une chaîne de caractères en entier. Cette routine prend en paramètre une chaîne uniquement constituée de chiffres entre 0 et 9 comme `"1234"` par exemple. Elle retourne l'entier dont la chaîne est l'écriture en base 10, c'est à dire 1234 pour la chaîne précédente. On peut supposer que la chaîne est valide et contient effectivement que des chiffres entre 0 et 9. Il n'est pas demandé de le vérifier. Un scénario possible d'utilisation de la routine `asc2int` est le suivant. La chaîne déclarée par `STRINGZ` occupe 5 mots mémoire. Les 4 premiers contiennent les codes ASCII des caractères `'1'`, `'2'`, `'3'` et `'4'`, c'est-à-dire les valeurs 49, 50, 51 et 52. Le dernier mot mémoire contient la valeur 0 pour terminer la chaîne.

```
          .ORIG x3000
          LEA R0,chaine          ; Chargement de l'adresse d'une chaîne
          JSR asc2int            ; Appel de la routine
          ...                    ; Au retour, R0 contient 1234

chaine:   .STRINGZ "1234"

asc2int:   ...                   ; Routine à écrire
          .END
```

a) Écrire une fonction C qui prend en paramètre une chaîne de caractères de type `char*` et terminée par le caractère `'\0'` et retourne la valeur entière de celle-ci.
b) Écrire une routine en langage d'assembleur LC-3 qui réalise la même opération. La routine devra prendre l'adresse de la chaîne dans le registre R0 et retourne le résultat dans le registre R0. Le programme peut librement utiliser les deux routines `mult10` et `digit` données ci-dessous.

```
; Multiplie le contenu de R1 par 10 en utilisant R5
mult10:   ADD R1,R1,R1
          ADD R5,R1,R1
          ADD R5,R5,R5
          ADD R1,R1,R5
          RET

; Retourne la valeur d'un chiffre contenu dans R2 en utilisant R5
digit:    LD R5,asc0
          ADD R2,R2,R5
          RET
asc0:     .FILL -48              ; -'0'
```

▶ **Exercice 4** On considère le cache d'un processeur utilisant des adresses sur 16 bits. On suppose que le cache est constitué de 16 lignes de 16 octets. On suppose qu'au départ le cache est vide, c'est-à-dire que toutes les lignes cachent les octets à l'adresse 0000. Le processeur accède successivement aux adresses suivantes : 4000, 3040, 4001, 3404, 4002, 30C4, 4003 et 34C4. Dans les deux cas suivants, donner les éléments correspondant à chaque accès : défaut ou succès, ligne du cache concernée, chargement ou non. On suppose pour simplifier que tous les accès sont des lectures. On note $b_{15}b_{14}\ldots b_2b_1b_0$ les 16 bits d'une adresse.

a) On suppose que le cache est direct. La ligne est déterminée par les 4 bits $b_7b_6b_5b_4$ de chaque adresse.
b) On suppose que le cache est 2-associatif. Chaque ligne i du cache est associée avec la ligne $i+8$ pour $0 \leq i \leq 7$. La paire de lignes est déterminée par les 3 bits $b_6b_5b_4$ des adresses.