NOM, Prénom, numéro d'étudiant :

CC-TD-1A - LIF6 - lundi 7/3/16 - noté sur 10 - durée : 30 minutes

Aucun document autorisé; calculatrices, téléphones et ordinateurs interdits. Le barème pourra être modifié.

Exercice 1. Un processeur est doté d'une mémoire de 256 o : à chaque adresse correspond une case de 1 octet. Il dispose de 4 registres généraux, indicés de 0 à 3 et notés `R0...R3`. Les instructions sont codées sur 1 ou 2 octets :

		premier octet								second octet
assembleur	action	7	6	5	4	3	2	1	0	7...0
`NOT RD,RS`	`RD <- ~RS` en complément à 2 sur 8 bits	1	0	0	1	RD		RS		—
`ADD RD,RS`	`RD <- RD + RS` en complément à 2 sur 8 bits	1	0	1	0	RD		RS		—
`INC RD,RS`	`RD <- RS + 1` en complément à 2 sur 8 bits	1	0	1	1	RD		RS		—
`ST RS,adr`	`mem[adr] <- RS`	0	1	0	0	RS		0	0	adr
`LD RD,adr`	`RD <- mem[adr]`	0	1	0	1	RD		0	0	adr
`HALT`	Met fin à l'exécution du programme	0	0	0	0	0	0	0	0	—

Dans ce tableau :

- `RD` désigne un registre de destination et `RS` un registre source (`R0...R3`) ;
- `adr` désigne une adresse codée sur 1 octet, et `mem[adr]` est la case mémoire d'adresse `adr` ;
- `~RS` est la négation bit-à-bit de `RS`. Par exemple, $\texttt{~}(01001101)_2 = (\bar{0}\bar{1}\bar{0}\bar{0}\bar{1}\bar{1}\bar{0}\bar{1})_2 = (10110010)_2$.

Le programme suivant calcule `mem[c] <- mem[a] - mem[b]` (il n'y a pas de dépassement de capacité ici) :

```
        LD R0,a      // R0 <- mem[a]
        LD R1,b      // R1 <- mem[b]
        NOT R2,R1    // R2 <- ~R1
        INC R2,R2    // R2 <- R2 + 1
        ADD R2,R0    // R2 <- R0 + R2 = mem[a] - mem[b] (en l'absence de dépassement)
        ST R2,c      // mem[c] <- R0
        HALT         // fin du programme
a:       -64         // constante décimale stockée en complément à 2 sur 8 bits
b:        57         // constante décimale stockée en complément à 2 sur 8 bits
c:        0          // emplacement où sera stocké le résultat
```

Question 1 (1 pt)**.** Que vaut `R2` après l'exécution des instructions `NOT R2,R1` et `INC R2,R2` (on attend une formule, pas un nombre) ? .....

Question 2 (1 pt)**.** Comment sont codés $(-64)_{10}$ et $(57)_{10}$ en complément à 2 sur 8 bits ? Posez vos calculs, et notez proprement vos résultats.

.....
.....
.....
.....
.....
.....
$(-64)_{10} =$.....

.....
.....
.....
.....
.....
.....
$(57)_{10} =$ .....

Question 3 (1,5 pt)**.** Posez, en complément à 2 sur 8 bits, l'opération $(-64)_{10} + (-57)_{10}$; encadrez votre résultat. Convertissez votre résultat en décimal en posant vos calculs.

.....
.....
.....
.....
.....
.....
.....

Question 4 (0.5 pt)**.** Après l'exécution de l'instruction `ADD R2,R0`, donnez le contenu du registre `R2` en binaire.

.....

Question 5 (3 pts)**.** Le tableau du début de l'exercice est reproduit ici :

		premier octet								second octet
assembleur	action	7	6	5	4	3	2	1	0	7...0
`NOT RD,RS`	`RD <- ~RS` en complément à 2 sur 8 bits	1	0	0	1	RD		RS		—
`ADD RD,RS`	`RD <- RD + RS` en complément à 2 sur 8 bits	1	0	1	0	RD		RS		—
`INC RD,RS`	`RD <- RS + 1` en complément à 2 sur 8 bits	1	0	1	1	RD		RS		—
`ST RS,adr`	`mem[adr] <- RS`	0	1	0	0	RS		0	0	adr
`LD RD,adr`	`RD <- mem[adr]`	0	1	0	1	RD		0	0	adr
`HALT`	Met fin à l'exécution du programme	0	0	0	0	0	0	0	0	—

Donnez le codage du programme : pour chaque ligne, donnez l'adresse du 1er octet, le codage en binaire, et le codage en hexadécimal de l'instruction ou de la donnée. Rayez les octets qui ne sont pas utilisés.

instruction ou donnée	adresse hexadécimale	premier octet 7	6	5	4	3	2	1	0	second octet 7	6	5	4	3	2	1	0	codage hexadécimal
`LD R0,a`	$1A$																	
`LD R1,b`																		
`NOT R2,R1`																		
`INC R2,R2`																		
`ADD R2,R0`																		
`ST R2,c`																		
`HALT`																		
`a: -64`																		
`b: 57`																		
`c: 0`																		

Exercice 2. Cet exercice est disjoint du précédent !

Question 6 (1 pt)**.** Quelle est l'écriture de $(1201)_3$ en décimal ? Posez votre calcul.

.....
.....
.....
.....
.....

Question 7 (2 pts)**.** Quelle est l'écriture de $(5,35)_{10}$ en binaire ? Posez votre calcul.

.....
.....
.....
.....
.....
.....
.....