CORRIGE Partiel Architecture des Ordinateurs - 20 Octobre 2015

Tous documents autorisés. Calculatrices interdites.
Durée 2h

Une justification **concise** sera donnée pour chaque question. Lorsqu'un résultat est demandé en notation hexadécimale, la notation binaire ne sera pas acceptée.

1. Représentation des entiers [2 pts]

Q1. On note # l'opération effectuée par un additionneur sur 8 bits. Pour les opérations suivantes, donner le résultat en notation **hexadécimale**, donner la retenue, et indiquer si le résultat est égal à celui de l'opération arithmétique d'addition lorsque les opérandes sont interprétés en naturels et en relatifs (complément à deux), suivant le format de la table.

	Opération	Résultat	Retenue (0/1)	Correct en naturels (Oui/Non)	Correct en relatifs (Oui/Non)
A	0x45 # 0x74	0xB9	0	Oui	Non
B	0xF0 # 0x75	0x65	1	Non	Oui
C	0xF1 # 0xA4	0x95	1	Non	Oui
D	0x34 # 0x45	0x79	0	Oui	Oui

Justification

- entiers naturels : le résultat est correct ssi il n'y a pas de retenue
- entiers relatifs : le résultat est correct ssi de signe contraire (cas B), ou de même signe et bit de signe = retenue (les autres cas)

2. Jeu d'instructions MIPS [6 pts]

Pour Q2, Q3, Q5, les résultats seront donnés en notation **hexadécimale**.

Q2. Donner le codage des instructions :

a) ADDIU R10, R1, 9 *b)* SUB R3, R1, R2 *c)* LW R2, 4(R1)

ADDIU R10, R1, 9 : format RI, codé par 0x242A0009

SUB R3, R1, R2 : format RR, codé par 0x00221822

LW R2, 4(R1) : format RI, codé par 0x8C220004

Justification : écrire le codage binaire.

Q3. L'état initial des registres est : R2 = 0xA7654321 et R3 = 0x0000000F. Donner le contenu du registre R1 après l'exécution des instructions :

a) ADD R1, R2, R3 *b)* ADDIU R1, R2, 0xFFFF *c)* ADDI R1, R2, 0xFFFF *d)* SRA R1, R2, 8
Remarque : ces instructions ne modifient pas R2 et R3.

a) ADD R1, R2, R3 R1 = 0xA7654330

b) ADDIU R1, R2, 0xFFFF R1 = 0xA7654320, extension de signe

c) XORI R1, R2, 0xF0F0 R1 = 0x A765B3D1

d) SRA R1, R2, 8 R1 = 0x FFA76543

e) SLT R1, R2, R3 R1 = 0x00000001 prédicat vrai car en relatifs, R2 < 0 < R3

f) SLTU R1, R2, R3 R1 = 0x00000000 prédicat faux car en naturels R3 < R2

Q4. Initialement R2 = 0x81234567. Donner un exemple de contenu de R3 tel que l'exécution des instructions ADD R1, R2, R3 et ADDU R1, R2, R3 fournisse des résultats différents.

ADD effectue un trap on overflow, alors que ADDU est silencieux.

R3=0x8000000 fournit un exemple d'overflow.

Q5. La mémoire est organisée en Big Endian. L'état de la mémoire est donné par la table 1. Le registre R1 contient 0x10000008. Donner le contenu du registre R2 après l'exécution des instructions :

a) LW R2, -4(R1) *b)* LW R2 , -8 (R1) *c)* LB R2, -7(R1) *d)* LHU R2, -2(R1)

Adresse	Valeur
0x10000000	0x85
0x10000001	0xB0
0x10000002	0x12
0x10000003	0x03
0x10000004	0xAA
0x10000005	0xBB
0x10000006	0xCC
0x10000007	0xDD

Table 1 – Plan mémoire

a) LW R2, -4(R1) R2 = 0xAABBCCDD

b) LW R2 , -8 (R1) R2 = 0x85B01203

c) LB R2, -7(R1) R2 = 0xFFFFFFB0

d) LHU R2, -2(R1) R2 = 0x0000CCDD

3. Programmation [10 pts]

On utilise le jeu d'instruction MIPS. On rappelle que le registre R0 est câblé à 0.

Q6. Soit le programme Prog1. Si R1 contient initialement 3, quelle est la valeur (notation **décimale**) de R2 après l'exécution de prog1 ? Quelle est l'opération arithmétique réalisée par prog1 ?

Prog1
SLL R2, R1, 4 ADDU R2, R2, R1 SLL R2, R2, 2 ADDU R2, R2, R1

SLL R2, R1, 4 // r2 = 16*r1
ADDU R2, R2, R1 // r2 = 16*r1 + r1 = 17 *r1
SLL R2, R2, 2 // r2 = r2*4 = 68*r1
ADDU R2, R2, R1 //r2 = r2+r1 = 69*r1

R2 contient 207. L'opération est la multiplication par 69.

Q7. Le tableau d'entiers X est implanté à partir de l'adresse 0x10000000, et contient {1,2,3,4,5,6,7,8}. Le tableau d'entiers Y est implanté à partir de l'adresse 0x20000000. L'état initial est :

R10 = 0x10000000, R11 = 0x20000000

Donner le contenu (notation **décimale**) du tableau Y après l'exécution de Prog2

```
        ADDI R12, R10, 32       // R12 = première adresse invalide
        ADDI R2, R0, 0          // R2 = 0
bcl :   LW R1, 0(R10)           // R1 = X[i]
        ADD R2, R2, R1          // R2 = R2 + X[i]
        SW R2, 0(R11)           //Y[i] = R2
        ADDI R10, R10, 4        // i++ pour X
        ADDI R11, R11, 4        //i++ pour Y
        BNE R10, R12,bcl        // test sortie de boucle
```

Le programme range les sommes successives (*préfixes*) du tableau X dans le tableau Y, soit
Y = {1, 3, 6, 10, 15, 21, 28, 36}

Q8. Le tableau d'entiers X est implanté à partir de l'adresse 0x10000000, et contient {9,9,7,4,8,8,8,2}. L'état initial est : R10 = 0x10000000, R3 = R4 = 0x00000001

Prog2	*Prog3*
ADDI R12, R10, 32	ADDI R12, R10, 28
ADDI R2, R0, 0	LW R1, 0(R10)
bcl : LW R1, 0(R10)	Deb : LW R2, 4(R10)
ADD R2, R2, R1	ADDI R4, R4, 1
SW R2, 0(R11)	BEQ R1, R2, Choix
ADDI R10, R10, 4	ADDI R4, R0, 1
ADDI R11, R11, 4	Choix: SLT R5, R4, R3
BNE R10, R12,bcl	MOVZ R3, R4, R5
	ADDI R1, R2, 0
	ADDI R10, R10, 4
	BNE R10, R12, bcl

a) Donner la valeur (notation **décimale**) des registres R1, R2, R3 et R4 après la première itération de *Prog3*. Même question après la deuxième itération.

		Itération 1	Itération 2
	ADDI R12, R10, 28		
	LW R1, 0(R10)	R1=X[0]=9	
Deb :	LW R2, 4(R10)	R2=X[1]=9	R2=X[2]=7
	ADDI R4, R4, 1	R4 =2	R4 = 3
	BEQ R1, R2, Choix	R1= R2, bcht	R1<> R2
	ADDI R4, R0, 1	---	R4 = 1
Choix:	SLT R5, R4, R3	R5 = (2<1)	R5 = (1 <2)
	MOVZ R3, R4, R5	R3 =2	---
	ADDI R1, R2, 0	R1 = 9	R1 = 7
	ADDI R10, R10, 4		
	BNE R10, R12, bcl		

c) Donner la valeur (notation **décimale**) du registre R3 après l'exécution de *Prog3*.

R3 = 3

```
                ADDI R12, R10, 28   // R12 = adresse de X[7]
                LW R1, 0(R10)       // R1 = 3
        Deb :   LW R2, 4(R10)       // R2 = X[i+1]
                ADDI R4, R4, 1      // R4++
                BEQ R1, R2, Choix
                ADDI R4, R0, 1      // si R1<>R2 R4 = 1 (sinon inchangé)
        Choix:  SLT R5, R4, R3      // R5 = (R4 <= R3)
                MOVZ R3, R4, R5     // si R4>R3 alors R4 = R3 ie R3 = max(R3, R4)
                ADDI R1, R2, 0      // préparation itération suivante
                ADDI R10, R10, 4    // i++ pour X
                BNE R10, R12, bcl   // test sortie de boucle
```

d) Que fait ce programme ?

Calcul de la plus longue séquence constante dans le tableau.

Algorithme (toutes les variables sont entières) :

```
xprev = X[0] ;
for (i= 0 ; i < N-1 ; i++) {
        if (xprev == X[i+1])
                runningMax++ ;
        else
                runningMax=1 ;
        globalMax=max(globalmax, runningMax) ;
x       prev = X[i+1] ;
}
```

4. Mémoire [2 pts]

Q9. On considère la déclaration C suivante :

```
char c[3];
double y[2] ;
short x ;
int* z ;
```

Le placement est aligné et en big endian. Les variables sont allouées dans l'ordre à partir de l'adresse 0x00001000. Donner l'adresse de la variable c[2], de la variable x et de la variable z.

Les adresses demandées sont en rouge.

Adresses	*Contenu*
0x00001000	c[0]
0x00001001	c[1]
0x00001002	c[2]
0x00001003-0x00001007	Inutilisé, alignement
0x00001008-0x0000100F	y[0]
0x00001010-0x00001017	y[1]
0x00001018-0x00001019	x
0x0000101A-0x0000101B	Inutilisé, alignement
0x0000101C-0x0000101F	z