Partiel Architecture des Ordinateurs – 25 Octobre 2019

Tous documents autorisés. Calculatrices interdites. Durée 2h

Une justification concise sera donnée lorsque c'est nécessaire. Lorsqu'un résultat est demandé en notation hexadécimale, la notation binaire ne sera pas acceptée.

1. Représentation des entiers sur 8 bits [4 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 interprétées en naturels ou en relatifs (compléments à deux), suivant le format de la table.

Opération	Résultat	Retenue (0/1)	Correct en naturel(Oui/Non)	Correct en relatifs (Oui/Non)
0x19 # 0x33	0x4C	0	Oui	Oui
0x57 # 0xC3	0x11A	1	Non	Oui
0xFF # 0xFF	0xFE	1	Non	Oui
0x11 # 0xAA	0xBB	0	Oui	Oui

Q2. Donner la représentation en **complément à deux** sur 8 bits (**représentation hexadécimale**) des nombres suivants : -8, -128, 20, 70.

-8 → 0xF8

-128 → 0x80

20 → 0x14

70 → 0x46

2. Représentation des réels, norme IEEE754 [4 pts]

On considère le format IEEE754 simple précision.

Q3. Coder 40,25 (représentation **hexadécimale**).

40,25 → 0x42210000

Q4. En déduire le code de 40,25 (**représentation hexadécimale**) pour les flottants double précision.

40,25 → 0x404420000 00000000

Q5. Donner la valeur (**représentation décimale**) des nombres réels suivants : 0x41000000, 0xBFC00000.

0x41000000 → 8

0xBFC00000 → -1,5

Q6. Sans effectuer de conversion au format IEEE754, donner le résultat de l'addition suivante dans les flottants simple précision : $2^{20} + 1{,}5.2^{-10}$

Résultat = 2^{20}

3. Jeu d'instructions MIPS [6 pts]

Q7. Donner le codage (**représentation hexadécimale**) des instructions :

a) ANDI R2, R4, 0xF0F0

0x3082F0F0

b) OR R2, R2, R3

0x00431025

c) JR R31

03E00008

Q8. L'état initial des registres est : R1 = 0x73459876 et R2=0xFF0000FF.

Donner le contenu du registre R3 (**représentation hexadécimale**) après l'exécution des instructions

a) ADDIU R3,R2,0x700F

R3 = 0xFF00710E

b) OR R3, R1, R2

R3 = 0xFF4598FF

c) SRA R3, R2, 31

R3 = 0xFFFFFFFF

La mémoire est organisée en Big Endian. L'état de la mémoire est donnée par la Table 1. On considère que les variables de type « int » x,y,z sont rangées en mémoire à la suite à partir de l'adresse 0x00001000 :

Adresse	Contenu
0x00001000 - 0x00001003	x
0x00001004 - 0x00001007	y
0x00001008 - 0x0000100B	z

Q9. Donner le code assembleur du Code 1

Code 1
y = 2*x ; z = x+y+z ;

LI r1, 0x00001000
LW r2, 0(r1)
ADDU r3, r2, r2
SW r3, 4(r1)

```
LW r4, 8(r1)
ADDU  r4, r4, r2
ADDU r4, r4, r3
SW r4, 8(r1)
```

Q10. Donner le contenu du registre R3 (**représentation hexadécimale**) après la séquence suivante

Séquence 1

LI R1, 0x00001000
LI R2, 0XFF00FF00
SW R2, 4(R1)
LB R3,6(R1)
SW R3, 8(R1)

R3 prend 0xFFFFFFFF après LB.

Q11. Données l'état des variables x,y et z (**représentation hexadécimale**) à la suite de cette séquence.

A la fin de la séquence on a
x est inchangé
y = 0xFF00FF00
z = 0xFFFFFFFF

4. Conditionnelles et boucles [4 pts]

Q12. On supposera que les tableaux x, y et z sont implémentés en mémoire aux adresses 0x00001000, 0x00002000 et 0x00003000. La variable s est implémentée à l'adresse 0x1000 0000. Écrire en assembleur MIPS32 le code de la boucle suivante :

```
int x[100];
int y[100];
int z[100] ;
int s=0 ;
for(int i=0 ; i<100 ; i++) {
   z[i] = x[i] +2*y[i]
   s += x[i]
}
```

```
LI r1, 0x1000
XOR r5, r5, r5
ADDIU r7, r1, 40
bcl :
  LW r2, 0(r1)
  LW r3, 0x1000(r1)
  ADDU r3, r3, r3
  ADDU r3, r2, r3
  SW r3, 0x2000(r1)
  ADDU r5, r5, r2
  ADDIU r1, r1, 4
  BNE r1, r7, bcl
```

LI r1, 0x10000000
SW r5, 0(r1)

5. Procédure [3 pts]

On considère le fragment de code suivant :

```
Lb1:
        SLT R3,R1,R0
        BNE R3, R0, p2
        NOT R2,R1
        ADDIU R2, R2, 1
        JR R31
p2:
        ADDIU R2, R1, -1
        NOT R2, R2
        JR R31
main :
        LI R1, 0x7000FFFE
        JAL Lb1
        JR R31
```

Q13. Donner la valeur du registre R2 (**représentation hexadécimale**) après l'exécution de ce programme.

R2 = 0x8FFF0002

Q13. Si on modifie la première ligne en LI R1, 0xFFFFFFFF, quelle est la valeur de R2 (**représentation hexadécimale**) après exécution ?

R2 = 0x000000001

Q9. Que fait la « fonction » Lb1 ?

Elle calcule l'opposé d'un nombre en c à 2.