TD 2 : Architecture des ordinateurs

Exercice 1

Pour chacune des caractéristiques présentées ci-dessous, dite si elle caractérise uniquement l'architecture RISC ou CISC, ou bien les deux architectures :

1. Une grande variété de modes d'adressage est disponible.
2. Architecture de type rangement/chargement.
3. Possède des instructions utilisant une valeur constante comme opérande, valeur directement conservée dans l'instruction.
4. Les instructions sont relativement simples et peuvent s'exécutées assez rapidement.
5. N'importe quelle instruction peut utiliser un opérande conservé en mémoire.
6. Possède un mode d'adressage permettant d'indiquer une adresse effective en mémoire via un registre de base plus un déplacement (appelé : offset).
7. Toutes les instructions possèdent la même taille (même nombre de bits**)**.

Exercice 2

Un concepteur de compilateur essaie de choisir entre deux séquences de code pour une machine donnée. Les concepteurs du matériel ont fourni les informations suivantes :

Classe d'instruction	CPI pour cette classe
A	1
B	2
C	3

Pour une expression particulière d'un langage de haut niveau, le concepteur du compilateur envisage deux séquences de code qui nécessitent les nombres d'instructions suivantes :

	Nombre d'instructions pour la classe d'instruction		
Séquence de code	**A**	**B**	**C**
1	2	1	2
2	4	1	1

1. Quelle séquence de code exécute le plus d'instructions ? Laquelle sera la plus rapide ?
2. Quel est le CPI de chaque séquence ?

Exercice 3

1. Expliquez le rôle des directives suivantes utilisées dans le langage d'assemblage MIPS : **.data ; .text ; .byte ; .asciiz ; .space ; .word ; .align.**
2. Donnez un exemple d'utilisation de chaque directive.

Exercice 4 fragment de code qui affiche un message.

1. Écrire le fragment de code qui permet la lecture d'un nombre du clavier.
2. Écrire le fragment de code qui affiche un nombre.
3. Écrire le fragment de code qui met fin à l'exécution du programme.

Exercice 5

1. Commenter le programme suivant et indiquer ce que fait la fonction F(X,Y,Z). On suppose que la variable X se trouve dans le registre $3, Y dans $2 et Z dans le registre $5 et le résultat dans $7.

```
add $7, $3 , $3        $7←$3 + $3
add $7, $7 , $7
add $7, $7 , $5
add $7, $7 , $5
add $7, $7 , $2
```

2. Déterminer la durée d'exécution de ce programme sur processeur à un cycle par instruction ayant une fréquence d'horloge 1Ghz.

Exercice 6

Supposez que vous voulez multiplier deux variables **a** et **b**, stockées dans les positions de mémoire M[adr] et M[adr +4], respectivement, pour affecter cette valeur à la variable Val, stockée à la position de mémoire M[adr+8] C'est-à-dire, vous voulez effectuer l'opération:

Val = a*b ou: M[adr+8] = M[adr]*M[adr+4]. Avec **adr = 0x10010000.**

Le processeur possède 8 registres (R0...R7). Les instructions du langage machine du processeur sont:

```
        LOAD   R1, M[adr]          #  R1 ← a
        LOAD   R2, M[adr+4]        #  R2 ← b
        ADD    R3, R0, R0          #  R3 ← 0
        BZ     R1, write           # IF a == 0, ranger le résultat dans la mémoire
etiq:   BZ     R2, write           # IF b == 0, ranger le résultat dans la mémoire
        ADD    R3, R3, R1          #  R3 ← R3 + a
        DEC    R2                  #  b ← b – 1
        JMP    etiq
write: STORE R3, M[adr+8]          #  M[adr] ← R3
```

Avant d'écrire le programme, nous devons trouver un algorithme réalisant la tâche voulue.

1. Représentez l'algorithme de ce programme.
2. Donnez le code correspondant à cet algorithme en langage d'assemblage MIPS.

Exercice 7

Soit la suite d'instructions suivantes :

```
        add $9, $0 , $0
Loop:
        lw $10, 1000($9)
        add $10, $10 , $2
        sw $10, 1000($9)
        add $9, $9 , $3
        beq $9, $4, loop
        j end
```

Supposons que le corps de la boucle s'effectue K fois, i.e. K itérations sont effectuées (K positif non nul).

1. Déterminer le nombre total d'instructions statiques.
2. Déterminer le nombre total d'instructions dynamiques.
3. Dans le cas de la mise en œuvre (1 cycle par instruction), combien de cycles seront requis pour l'exécution de la suite d'instructions ci-dessus ?

Exercice 8

Supposons que le registre R0 de taille 32bits d'un microprocesseur supportant le système de stockage des données *big* et *little-endian*, est initialisé par la valeur 0x23456789. Donner le contenu du R1 après exécution du programme ci-dessous en *big-endian (gros-boutiste)*. Même question dans le cas où le programme soit exécuté en *little-endain (petit-boutise)*.

STORE Word R0, MEM[$0] ➔ charger dans la mémoire le mot 0x23456789 à l'adresse ($0)

LOAD BYTE R1, MEM[1 + $0] ➔ charger dans R1 le byte qui se trouve dans la mémoire à l'adresse (1+$0)

Exercice 9

1. Quel est l'avantage de l'alignement des mots mémoires ?
2. Étant donné les adresses mémoires suivantes :

(a) 12345678 (c) 9128ADCC

(b) ABCD755A (d) 38B0F050

- Déduire si les 32-bits stockés en mémoire sont alignés ou non.
- Même question dans le cas des 64-bits.

Exercice 10

Pour chacune des transferts ci-dessous, écrire l'instruction d'assemblage correspondant dans le cas où les opérandes sont signés, puis dans le cas où ils sont non signés.

$5 ← $12 + $4
$7 ← $4 - $3
$8 ← $8 + 1
$17 ← $9 – 13
$13 ← $20 and $15
$6 ← decal age à gauche de $7
$4 ← decalage à droite de $9

Exercice 11

1. Quelle est la différence entre les instructions de saut suivantes ?
   1. **j etiquette**
   2. **jr $31**
   3. **jal proc1**

2. Supposons que A est un ensemble de 100 mots et que le compilateur a associé les variables g et h aux registres $s1 et $s2. Supposons aussi que l'adresse du premier élément du tableau A est en $s3.

g = h + A[8];

- Écrire le code d'assemblage MIPS correspondant.

Exercice 12

- Écrire l'instruction MIPS correspondante à chaque. Dans tous les cas l'étiquette "suite" figurera dans l'instruction de branchement.

1. if $7 < 0 then instructions end_if suite:	
2. if $15 ≥ 0 then instructions end_if suite:	
3. if $11 ≤ 0 then instructions end_if suite:	
4. if $2 == $10 then instructions end_if suite:	

- Ecrire les deux instructions MIPS de chaque condition de test ci-dessous. On utilisera le registre $1 comme registre intermédiaire :

1. if $21 < $17 then instructions end_if suite:		
2. if $18 ≥ $13 then instructions end_if suite:		
3. if $21 > $17 then instructions end_if suite:		
4. if $18 ≤ $13 then		

instructions end_if suite:		

Exercice 13
Écrire en langage d'assemblage MIPS le fragment C suivant :

if (t1 < t2)
{
t3 = t1 ;
else
t3 = t2 ;
}	

Même question pour le fragment suivant :

t2 = 0;
while (t1 > 0)
{
t2 = t2 + t1;
t1 = t1-1 ;
}	

t2 = 0 ;
for (t1=1 ; t1< N ; t1++)
{
t2 = t2 +2×t1;
}	