Université d'Alger 1

Faculté des Sciences
Département Informatique

Examen L2-ADO

25-Janvier-2024

Durée : 90 minutes

Nom : Prénom : Groupe : Matricule :		/ 20

Exercice 1 : (6 points)

Supposons que nous exécutions les instructions MIPS suivantes

```
  li   $t0, 2
  li   $t1, 5
  slt  $t2, $t1, $t0
  beq  $t2, $zero, skip
  addi $t1, $t2, 3
skip:
  li   $v0, 42
```

Dans le tableau suivant, inscrivez chacun des registres modifiés et leurs valeurs après l'exécution du code. La première colonne est remplie pour vous.

Registre	**$t0**	**$t1**	**$t2**	**$v0**			
Valeur	**2**	**5**	**0**	**42**			

Exercice 2 : (4 points) (-1 point pour chaque réponse fausse)

Les processeurs d'aujourd'hui possèdent deux modes d'exécution : le mode superviseur, où toutes les opérations sont autorisées, et le mode utilisateur, où certaines opérations sont désactivées. Les programmes ordinaires s'exécutent en mode utilisateur. Ils ne peuvent pas apporter de modifications aux parties du système qui leur permettraient de contourner les protections, de lire ou d'écrire des données non autorisées ou d'interférer avec le fonctionnement du système. Pour chacun des éléments suivants, cochez la case dans la colonne « mode utilisateur » s'il est possible d'autoriser ces opérations en mode utilisateur. Sinon, cochez « mode superviseur uniquement ».

Opération	Mode utilisateur	Mode superviseur uniquement
Stocker une valeur dans un registre général comme **$t0**	X	
Désactivez les interruptions afin que rien n'interrompe le processus en cours pendant qu'il fait quelque chose d'important		X
Lancer une opération d'E/S sur un périphérique à l'aide d'E/S mappées en mémoire		X
Changer le compteur ordinal	X	

Exercice 3 : (4 points) (-1 point pour chaque réponse fausse)

Entourez la réponse correcte.

- **Vrai / Faux.** MIPS est une architecture « load-store »
- **Vrai / Faux.** Un processeur avec une fréquence d'horloge plus rapide a toujours des performances plus élevées qu'un processeur avec une horloge plus lente.
- **Vrai / Faux.** La localité temporelle fait référence à l'idée que les données en mémoire dont l'adresse est proche des données qui ont été consultées récemment sont susceptibles d'être consultées prochainement.
- **Vrai / Faux.** Le champ d'opcode de toutes les instructions MIPS au format J est 0.

Exercice 4 : (4 points)

Le code MIPS suivant peut être utilisé pour incrémenter la valeur de deux variables entières (similaire à « i++ ; j++ » en C).

```
(a)  li    $t0,1            # affecte 1 à $t0
(b)  lw    $t1,100($t5)     # lecture de la première variable
(c)  add   $t1,$t0,$t1      # incrémentation
(d)  sw    $t1,100($t5)     # sauvegarde
(e)  lw    $t2,104($t5)     # lecture de la seconde variable
(f)  add   $t2,$t0,$t2      # incrémentation
(g)  sw    $t2,104($t5)     # sauvegarde
```

1/ Supposons que ces instructions soient exécutées sur un processeur MIPS en pipeline à 5 étages avec la mise en œuvre de la détection des aléas et le transfert de données. Le processeur doit-il insérer des cycles de suspension matérielle (bulles de pipeline) pour éviter des résultats incorrects lors de l'exécution ? Si oui, combien et où sont-ils nécessaires ? (indiquez cela avec des phrases comme : *n* cycles insérés entre les instructions (x) et (y).)

Le processeur doit être suspendu pendant un cycle entre (b) et (c) et à nouveau entre (e) et (f). Dans ces deux cas, le résultat d'une lecture-mémoire est requis comme valeur en entrée pour l'instruction suivante, et même avec le transfert, la valeur n'est pas disponible à partir de l'unité de mémoire à temps pour la renvoyer au cycle d'exécution de l'instruction suivante.

(2 points)

2/ Sans changer ce que fait le code, est-ce que les instructions ci-dessus peuvent être réorganisées pour réduire le nombre de suspension nécessaires ? Si c'est le cas, décrivez le réarrangement et expliquez combien de cycles de suspension (s'il y en a) sont nécessaires dans votre version révisée du code.

Tout réarrangement qui ramène les instructions de lecture plus tôt afin que leurs valeurs ne soient pas nécessaires à l'instruction suivante éliminera le besoin de suspension. Un moyen simple de le faire est d'ordonner les instructions comme suit : (b), (e), (a), (c), (d), (f), (g).

(2 points)

Exercice 5 : (4 points)

Voici un code partiel en langage assembleur MIPS :

Adresse	Etiquette	Instructions
0x00400000		bgtz $a1, loop
. . .		. . .
0x00403000	loop:	addu $a0, $a1, $v0
. . .		. . .
0x00410000		bne $a0, $zero, loop

1/ Calculez la valeur immédiate sur 16 bits (en hexadécimal) associée à **loop** dans l'instruction **bgtz.**

(1 point)

bgtz: (0x00403000-0x00400004)/4 = 0x2FFC/4 = 0x0BFF (sur 16 bits)

2/ Calculez la valeur immédiate sur 16 bits (en hexadécimal) associée à **loop** dans l'instruction **bne.**

(1 point)

bne: (0x00403000-0x00410004)/4 = 0xF2FFC/4 = 0xCBFF (sur 16 bits)

3/ Supposons que nous interprétions le mot suivant comme une séquence de caractères ASCII. Qu'est-ce qu'il signifie ?

(2 points)

0x42796521	**0x42 = 'B', 0x79 = 'y', 0x65 = 'e', 0x21 = '!' ⇨ 0x42796521 = 'Bye!'**