Questions de revision
INF4170

12 avril 2006.

Numéro 1. (0 pts)
Cette question porte sur une mise en oeuvre du processeur MIPS utilisant un pipeline à huit (8) étages plutôt que cinq (5). Dans le cas de ce pipeline, on suppose que les étapes d'accès mémoire et d'execution (UAL) ont été décomposées en plusieurs cycles. La décomposition en étapes de l'exécution d'une instruction est donc la suivante, où chaque étape correspond à un étage du pipeline:

EI1: Initiation de l'extraction de l'instruction et calcul de CP+4.

EI2: Attente et terminaison de l'extraction de l'instruction.

DI: Décodage de l'instruction et extraction des registres.

EX1: Début de l'exécution de l'opération UAL (ou calcul de l'adresse destination).

EX2: Fin de l'exécution de l'opération UAL.

MEM1: Initiation de l'accès mémoire.

MEM2: Attente et terminaison de l'accès mémoire.

ER: Écriture du résultat dans le registre destination.

Une représentation schématisée de ce pipeline serait la suivante:

EI1	EI2	DI	EX1	EX2	MEM1	MEM2	ER

Un autre aspect important concernant la structure de ce pipeline est la façon dont la condition de branchement est traitée: on suppose que la condition de branchement est disponible et peut être utilisée au cours de l'étape `EX2` pour mettre à jour `CP` (si nécessaire). Évidemment, la nouvelle valeur de `CP` ne sera alors disponible qu'au début du cycle *suivant*.
Finalement, supposons que ce pipeline fonctionne avec une horloge de 3.6 GHz.

a) (0 pts) Pour le pipeline à 8 étages, identifiez les aléas de données *potentiels* (i.e., ceux qui existent, pour le pipeline indiqué, lorsqu'aucun mécanisme de résolution n'est utilisé) dans le segment de programme présenté plus bas.
Dans chacun des cas, indiquez les éléments suivants:

- Les instructions entre lesquelles l'aléa de données existe ainsi que le numéro du registre causant le conflit.
- Le type d'aléa, c'est-à-dire l'un des types suivants:

  A) L'aléa serait éliminé par l'ajout d'une unité d'envoi utilisant les valeurs disponibles dans les registres inter-étages.

  B) L'utilisation d'envois ne permettrait pas d'éliminer l'aléa de données.

```
(I1)   add  $2, $3, $4
(I2)   sub  $3, $4, $5
(I3)   add  $6, $2, $3
(I4)   add  $2, $6, $2
(I5)   lw   $8, 100($6)
(I6)   add  $13, $12, $8
(I7)   add  $4, $6, $8
(I8)   lw   $7, 100($8)
(I9)   sub  $6, $2, $8
(I10)  add  $3, $8, $2
(I11)  sub  $6, $8, $2
(I12)  add  $2, $4, $7
```

b) (0 pts) En supposant que les aléas de données de type A et B identifiés en a) sont effectivement résolus par des envois alors que les autres aléas (type C) sont résolus par l'ajout (par le compilateur) d'instructions `nop`, quel sera le temps requis pour l'exécution de ce segment de programme?
(N'oubliez pas le temps de remplissage du pipeline.)

Numéro 2. (0 pts) Cette question utilise le pipeline décrit en 1.
Supposons que l'on ait un programme dont 16 % des instructions sont des branchements conditionnels. Supposons de plus que 20% des branchements sont effectivement effectués (i.e., que la condition associée à un branchement est vraie, en moyenne, dans 20% des cas).

En utilisant le modèle du pipeline à 8 étages présenté plus haut, comparez les performances des machines décrites plus bas (M_1, M_2, M_3 et M_4). Plus précisément, donnez le nombre de cycles de pénalité pour chaque machine.

Remarque: Pour ce problème, examinez uniquement les aléas *de contrôle*, i.e., ignorez les délais possibles dûs aux aléas de données. (On peut supposer, par exemple, que la machine comporte une unité d'envoi et que les autres aléas sont traités correctement et efficacement par le compilateur.)

- M_1: Machine où le pipeline est gélé (i.e., les instructions qui suivent le branchement sont suspendues) jusqu'à ce que la condition de branchement soit calculée et utilisable (au cours de l'étape `EX2`).
- M_2: Machine prédisant que le branchement ne sera pas effectué (prédiction statique), donc qui amorce toujours l'exécution des instructions qui suivent en séquence l'instruction de branchement; lorsque le branchement doit effectivement s'effectuer, le pipeline doit alors être nettoyé.
- M_3: Machine utilisant une méthode dynamique de prédiction des branchements avec un taux de prédiction correcte de 95%.
- M_4: Machine utilisant des branchements différés et où on suppose que le compilateur réussit à remplir efficacement (i.e., sans instruction `nop`, donc avec une instruction utile au calcul) seulement 50% (en moyenne) des cases de délai de branchement.

Numéro 3. (0 pts)
Pour cette exercice supposons une architecture à plusieurs cycles par instruction. Cette architecture est composée des quatres (4) modules suivants: Unité mémoire, ual, banc de registre et registre PC. Cette architecture contient les registre interne suivant: PC, DR1, DR2, DR3, IR. Voici la liste des micro-codes que peut recevoir chacun de ces modules:

- Unité mémoire:

  - **Lire-pc**: Lit une instruction à l'adresse indiquée par PC, le résultat est placé dans le registre IR.
  - **Lire-ual*i***: Lit une donnée à l'adresse indiquée par le registre de l'ual, le résultat est placé dans le registre DRi.
  - **Écrire-ual*i***: Écrit la valeur du registre DRi dans la case mémoire indiquée par le registre de l'ual.
  - **Nop**: L'unité mémoire ne fait rien.

- Ual:
  - **Add(*a*, *b*)**: Additionne les valeurs a et b et place le résultat dans le registre de l'ual.
  - **AddTemp(*a*, *b*)**: Additionne les valeurs a et b et place le résultat dans un registre temporaire de l'ual.
  - **Sub(*a*, *b*)**: Soustrait les valeurs a et b et place le résultat dans le registre de l'ual.
  - **Nop**: L'ual ne fait rien.

  Les valeurs possible pour a et b sont les suivantes:

  DR*i* .

  PC : pour a seulement.

  4 : pour b seulement.

  0 : pour b seulement.

- Banc de registre (et décodage):
  - **Écrire-ual**: Écrit la valeur du registre de l'ual dans le registre destination de l'instruction contenue dans le registre IR.
  - **Écrire-reg*i***: Écrit la valeur du registre DRi dans le registre destination de l'instruction contenue dans le registre IR.
  - **Lire*i*-vers*j***: Lit le i^{iem} registre de l'instruction et écrit le résultat dans le registre DRj.
  - **Lire-imm*i*-vers*j***: Lit la i^{iem} valeur immédiate de l'instruction et place cette valeur dans le registre DRj.
  - **Nop**: Le banc de registre ne fait rien.

- Registre PC:
  - **Écrire-ual**: Écrit le résultat du registre de l'ual dans le registre PC.
  - **Saut*i***: Écrit la valeur du registre DRi dans le registre PC.
  - **Branchement**: Écrit la valeur du registre temporaire de l'ual dans le registre PC si la valeur dans le registre de l'ual est égal à zéro.
  - **Nop**: aucun chagement au registre PC.

Il est possible d'exécuter une micro-code par module à chaque cycle d'horloge. Voici le code pour lire une instruction:

cycle	Mémoire	Ual	Registres	PC
1	Lire-pc	Add(PC, 4)	Nop	Nop
2	Nop	Nop	Nop	Écrire-ual

Voici le code pour exécuter l'instruction:

`add $a, $b, $c` # $a := $b + $c

cycle	Mémoire	Ual	Registres	PC
3	Nop	Nop	Lire2-vers1	Nop
4	Nop	Nop	Lire3-vers2	Nop
5	Nop	Add(DR1, DR2)	Nop	Nop
6	Nop	Nop	Écrire-ual	Nop

Écrivez les micro-programmes nécessaire pour les instructions suivantes:

a) (0 pts) sub ($a), imm1($b), @imm2($c)
mem[$a] := mem[imm1 + $b] - mem[mem[imm2 + $c]]

b) (0 pts) beq imm1($a), $b, imm2
si mem[imm1 + $a] == $b alors PC := PC + imm2

Numéro 4. (0 pts)
Soit une architecture avec le pipeline à 7 étapes suivant:

EI1	EI2	DI	EX	MEM1	MEM2	ER

a) (0 pts) Donnez une suite d'instructions assembleur MIPS qui contient un aléas de données qui peut être résolu par une unité d'envoie et un autre aléas qui ne peut être résolu par l'unitée d'envoie.
Nous voulons comparer les performances de notre pipeline avec une machine super-scalaire. Le programme que nous utilisons à 120 000 000 instructions distribuées de la façon suivante :

Classe	Distribution
arithmétique	35 %
chargement	18 %
rangement	17 %
branchement	22 %
saut	8 %

b) (0 pts) Sachant que le pipeline a une vitesse d'horloge de 2.0 GHz, calculez le temps d'exécution de ce programme **sans** tenir compte des problèmes d'aléas de données et de contrôles.

c) (0 pts) Pour sa part le super-scalaire réussit à terminer l'exécution de 3 instructions par cycle (en moyenne). Il a une fréquence d'horloge de 0.8 GHz. Calculez le temps d'exécution du programme sur cette machine.

d) (0 pts) Comparez les performances des deux (2) machines.

e) (0 pts) Une stratégie de prédiction statique prédisant que les branchements n'ont jamais lieu est utilisée pour une première mise en oeuvre. Le programme est composé à 22 % d'instruction BEQ. À l'aide de mesures faites sur le programme P, nous savons que 15% des branchements ont lieu.
Une deuxième architecture contient un mécanisme de prédiction dynamique à deux bits. Sur cette architecture le taux de prédiction correcte est de 65 %.
Laquel des deux architectures va causer le moins de pénalitées et de combien (en %) sera t-elle plus performante?

Numéro 5. (0 pts)

Cette question porte sur une mémoire cache à correspondance direct où chaque bloc contient 4 mots. Une technique d'écriture à l'éjection est choisie.
Supposez les valeurs suivantes en mémoire RAM:

adresse	valeur
0	12
1	32
2	45
3	54
4	13
5	34
6	25
7	74
8	53
9	74

adresse	valeur
10	25
11	64
12	53
13	24
14	53
15	75
16	54
17	52
18	78
19	43

adresse	valeur
20	26
21	47
22	85
23	14
24	75
25	15
26	36
27	74
28	53
29	54

adresse	valeur
30	25
31	64
32	25
33	74
34	52
35	85
36	36
37	85
38	22
39	64

Voici l'état de la cache après plusieurs accès :

adr.	étiq.	V	S	0	1	2	3
0	1	1	0	54	52	78	43
1		0	0				
2	1	1	1	75	10	36	74
3	0	1	0	53	24	53	75

Pour la suite de lecture/écriture suivante indiquez s'ils causent des échec ou des succès. Donnez aussi l'état de la cache final (après les six accès).

L:n = Lecture de la case mémoire **n**.

É(x):n = écriture de la valeur **x** à la case mémoire **n**.

	L/É	S/É
a)	L:10	
b)	L:13	
c)	É(111):15	

	L/É	S/É
d)	L:26	
e)	E(222):37	
f)	L:24	

adr.	étiq.	V	S	0	1	2	3
0		0	0				
1		0	0				
2		0	0				
3		0	0				

Numéro 6. (0 pts) Voici un code MIPS :

```
lw $2, 100($3)
sw $2, 100($3)
add $2, $4, $5
sub $2, $2, $6
sw $2, 0($7)
```

Identifiez les anti-dépendances incluses dans ce code et réécrivez le code en enlevant les anti-dépendances. Considerez que le programmeur a accès aux registres $0 à $31 et que le banc de registre virtuel contient les registres $0 à $127.

Numéro 7. (0 pts)

a) (0 pts) L'architecture IA-64 utilise un jeu d'instructions RISC. En quoi un jeu d'instructions CISC serait-il un mauvais choix pour une telle architecture?

b) (0 pts) Quelles sont les avantages du parallèlisme d'instruction explicite par rapport au parallèlisme implicite ?