Département de génie électrique et de génie informatique André Zaccarin
Université Laval

GIF-17455 – Architecture des micro-processeurs

Examen Partiel

Jeudi le 7 novembre 2002 Durée: 13h30-15h20

Question 1

Valeurs intiales: R1 = 80; F4 = 0;

```
Loop:  LD F1, 1000 (R1)      ; F1 = MEM(R1+1000)
       LD F2, 2000 (R1)      ; F2 = MEM(R1+2000)
       SUB.D. F3, F1,F2      ; F3 = F1 – F2
       ADD.D, F4, F4, F3     ; F4 = F4 + F3
       SUB.I, R1,R1,#8       ; R1 = R1 – 8;
       BNEZ R1, Loop         ; Si R1 différent de 0, goto Loop
```

Les 2 premières itérations de la boucle exécutées avec un processeur dynamique (et l'algorithme de Tomasulo) donne le tableau suivant:

Instruction	Émission	Exécution	Accès Mémoire	Écriture (WB)
LD F1, 1000(R1)	1	2	12	13
LD F2, 2000 (R1)	2	3	4	5
SUB.D. F3, F1,F2	3	14-16		17
ADD.D. F4, F4, F3	4	20-22		23
SUB.I R1,R1,#8	5	6		7
BNEZ R1,Loop	6	8		
LD F1,1000(R1)	9	10	11	12
LD F2, 2000 (R1)	10	11	13	14 (port d'accès mémoire occupé)
SUB.D. F3, F1,F2	11	17-19		20
ADD.D, F4, F4, F3	12	24-26		27
SUB.I,R1,R1,#8	13	14		15
BNEZ R1,Loop	14	16		

Le processeur dispose d'une unité fonctionnelle pour effectuer les opérations en points flottants, et d'une unité fonctionnelle pour les opérations ALU (incluant le calcul des adresses pour les accès en mémoire). Il n'y a qu'un port d'accès à la mémoire.

Le processeur dispose de suffisamment de stations de réservation et d'entrées dans le tampon de lecture (load buffer) pour éviter les arrêts du pipeline. Il n'y a pas de prédiction de branchement. Si deux instructions peuvent démarrer leur exécution en même temps, l'instruction émise en premier a priorité.

Les accès en mémoire des LDs se font respectivement 10, 1, 1 et 1 cycles après le calcul de l'adresse (cycle d'exécution) si les ressources le permettent.

a) Selon le tableau de la page précédente, le **`LD F1,…`** de la 2ème itération est complété avant celui de la 1ère itération. Est-ce possible ? Si oui, expliquez le processus qui permet au **`SUB.D F3,F1,F2`** de la 1ère itération d'utiliser les bonnes valeurs pour l'exécution de l'opération. Si non, corrigez les valeurs du tableau.

b) Le processeur est modifié pour supporter la spéculation. Faites l'hypothèse que les branchements sont prédits comme étant pris. Donnez le nouveau tableau d'exécution du code. Justifiez bien les délais dans l'exécution. Il y a suffisamment d'entrées dans le tampon de réordonnancement (Reorder Buffer, ROB) pour éviter les arrêts (stalls) du pipeline.

c) Référez-vous au tableau de la page précédente. Selon vous, quelle ressource (unité fonctionnelle, port d'accès mémoire, etc.) doit-on ajouter pour réduire le plus la durée d'exécution des 2 premières itérations ? Quel serait le gain ? Vous n'avez pas à refaire le tableau. Suggestion: justifier les délais du tableau.

d) Quelle est la dépendance qui limite le plus l'accélération de l'exécution de cette boucle ?

e) (question bonus) Est-ce que la boucle est parallélisable ? Si oui, expliquer comment. Vous pouvez faire des changements au code.

Question 2

L'équation de performance d'un CPU est la suivante:

Temps d'éxécution = # instructions x CPI x durée d'un cycle

Pour chacune des décisions de design décrites ci-dessous, indiquer l'effet sur chacun des trois termes de l'équation du CPU (augmente, diminue, pas d'effet) avec brève justification

- Augmenter le nombre d'étages du pipeline d'un processeur statique de 5 à 8
- Ajouter le passage des données (data forwarding)
- Ajouter une instruction de délai de branchement
- Utiliser un compilateur qui fait du déroulement de boucle et réorganisation de code(scheduling)

Question 3

Nous désirons modifier un processseur MIPS statique implantée en pipeline avec 5 étages (IF,ID,EX,MEM,WB) pour pouvoir supporter des instructions ALU avec **prédication**.

Le processeur utilise la prédiction statique "*branchement non-pris*"; la résolution des branchements conditionels n'est faite qu'à la fin du cycle MEM, et la résolution des sauts (jump, branchements inconditionnels) est faite à la fin du cycle ID. Le processeur est réalisé avec le passage des données (data forwarding).

Nous ajoutons des instructions pour initialiser les registres de prédications, et les instructions avec prédication. L'instruction **`CMP.EZ pT,pF=R1,0`** place dans le registre **`pT`** la valeur 1 si R1=0, et 0 autrement (le registre **`pF`** contient l'inverse du registre **`pT`**). Les valeurs des registres **`pT`** et **`pF`** sont calculées au cycle MEM.

Les instructions avec prédication sont de la forme **`(px)Opération ALU.`** Lors de l'exécution des instructions avec prédication, le processeur a besoin de la valeur du registre **`px`** lors du cycle EX. Si le registre **`px`** est égal à 1, l'instruction se comporte de la même façon que l'opération sans prédication. Si le registre **`px`** est égal à 0, l'instruction se termine avec la fin du cycle EX.

Le pseudo-code que nous désirons exécuter est le suivant:

```
if (R1 == 0)
      OP1 xx, xx, xx
else {
      OP2 xx, xx, xx
      OP3 xx, xx, xx }
```

Voici le code "MIPS" sans et avec prédication:

	Code MIPS sans prédication	Code modifié avec prédication
	`BNEZ R1, A`	`CMP.EZ pT,pF= R1,0`
	`OP1 xx,xx,xx`	`(pT) OP1.S xx,xx,xx`
	`Jump B`	`(pF) OP2.S xx,xx,xx`
`A:`	`OP2 xx,xx,xx`	`(pF) OP3.S xx,xx,xx`
	`OP3 xx,xx,xx`	
`B:`	`(suite du code)`	

Le CPI du MIPS sans prédication et sans compter les arrêts (stalls) causés par l'exécution des *if ( ) else ( )* est de 1.2 . Dix pour cent (10%) des instructions sont des instructions *if ( ).* La réalisation du MIPS avec prédication augmente la durée du cycle de 8% par rapport à la réalisation sans prédication.

Si R1 = 0 pour *x* % des groupes *if ( ) else ( )* , pour quelles valeurs de *x* l'addition de la prédication augmente-t-elle la performance du processeur ?

Note: OP1,OP2 et OP3 dénotent des opérations ALU sans dépendance entre elles ou avec R1

Question 4

Un programme contient 4 branchements à l'intérieur d'une boucle. Lors de l'exécution de la boucle (10 itérations), le comportement suivant est observé:

	1	2	3	4	5	6	7	8	9	10
B1	N	N	N	T	T	N	T	T	N	T
B2	N	T	T	N	T	T	N	N	T	N
B3	N	T	N	T	N	T	N	T	N	T
B4	T	T	T	T	T	T	T	T	T	N

Quel est le nombre de bonnes prédictions pour le branchement **B3** si on utilise

a) un prédicteur à un bit initialisé à N

b) un prédicteur à deux bits initialisé au mode "fortement pris" (T fort)

Pourquoi ce prédicteur est-il meilleur que celui en a) ?

c) un prédicteur (2,2) avec l'initialisation suivante:

Prédictions précédentes	Valeur initiale des prédicteurs
(N,N)	N fort
(N,T)	N fort
(T,N)	N fort
(T,T)	N fort

Donnez l'état final des 4 prédicteurs.

Si le comportement observé des branchements demeure le même, est-ce que le taux de bonnes prédictions lors du prochain passage dans la boucle sera meilleur ?