Architecture des microprocesseurs (GIF-3000)
Département de génie électrique et de génie informatique
Automne 2011

UNIVERSITÉ LAVAL

EXAMEN PARTIEL #1

Instructions : – Une feuille aide-mémoire recto-verso manuscrite est permise ;
– Utilisez le verso des pages pour les brouillons ;
– Durée de l'examen : 1 h 50.

Pondération : Cet examen compte pour 25% de la note finale.

Question	Points	Score
1	16	
2	20	
3	20	
4	16	
5	28	
Total:	100	

Signature : _____

Question 1 (16 points sur 100)

Soit un système informatique effectuant le contrôle d'une capsule spatiale, formé de deux processeurs distincts. La fiabilité de chacun de ces processeurs, dans des conditions d'exposition au rayonnement cosmique, est de 1000 heures (temps moyen avant une défaillance, MTTF). Supposons que le système de contrôle est opérationnel lorsqu'au moins une unité fonctionne correctement, et que le temps pour réinitialiser un processeur dans un état valide lors d'une défaillance est de 3 minutes (temps moyen de réparation, MTTR). Cependant, si une défaillance survient pour un deuxième processeur alors que le premier est en cours de ré-initialisation, le système de contrôle de la capsule atteint dans un état invalide.

(10) (a) Calculez le taux de panne de ce système de contrôle de la capsule, en supposant que les pannes des processeurs sont indépendantes.

Solution: Temps moyen avant d'avoir une première panne de processeur (sur deux unités) :

$$MTTF_{proc} = \frac{1000}{2} = 500$$

Probabilité que le deuxième processeur tombe en panne pendant la ré-initialisation du premier :

$$p_{panne} = \frac{3/60}{1000} = 5 \times 10^{-5}.$$

Temps moyen avant qu'une panne de ce type se produit :

$$MTTF_{panne} = \frac{MTTF_{proc}}{p_{panne}} = \frac{500}{5 \times 10^{-5}} = 10^7.$$

Donc, le taux de panne du système de contrôle est :

$$taux_{panne} = \frac{1}{10^7} = 10^{-7} = 100 \text{ FIT}.$$

(6) (b) Supposons maintenant que l'on veut utiliser la capsule spatiale pour une mission vers la planète Mars d'une durée de deux ans. On veut augmenter le nombre de processeurs du système de contrôle pour obtenir une fiabilité de 99,9995% pour la durée de la mission. On dit que le système de contrôle va fonctionner tant qu'au moins un processeur fonctionne en tout temps. Calculez le nombre de processeurs nécessaire pour atteindre cette fiabilité.

Solution: Taux de pannes horaire maximal du système pour la mission :

$$taux_{panne\ cible} = \frac{1 - 0{,}999995}{2 \times 365 \times 24} = 2{,}85^{-10}$$

Temps moyen avant une panne pour n processeurs :

$$MTTF_{panne} = \frac{MTTF_{proc}}{p_{panne}} = \frac{1000}{n} \frac{1}{(5 \times 10^{-5})^{n-1}},$$

$$taux_{panne} = \frac{1}{MTTF_{panne}} = \frac{n\,(5 \times 10^{-5})^{n-1}}{1000}.$$

Pour $n = 2$: $taux_{panne} = 10^{-7} > 2{,}85^{-10}$, c'est trop élevé.
Pour $n = 3$: $taux_{panne} = 7{,}5^{-10} > 2{,}85^{-10}$, encore trop élevé.
Pour $n = 4$: $taux_{panne} = 5^{-13} < 2{,}85^{-10}$, c'est suffisant.
Donc, avec quatre processeurs, le système de contrôle aura la fiabilité demandée.

Question 2 (20 points sur 100)

Supposons que l'on veut choisir un processeur pour une application particulière et que les deux choix suivants s'offrent à nous.

Processeur	Performance entiers	Performance virgules flottantes
#1	4063	2712
#2	6245	2086

Les valeurs de performances entiers et virgules flottantes peuvent être directement comparées et les valeurs élevées sont meilleures que les valeurs faibles. On suppose que pour notre application particulière, un processeur passe 65% du temps dans des calculs sur des nombres entiers et le temps restant (35%) à faire des calculs sur des nombre à virgules flottantes.

(10) (a) Calculez le gain en performances pour notre application en utilisant le processeur #2 comparativement au processeur #1.

Solution: Performance pondérée du processeur #1 :

$$\text{Perf}_1 = 0{,}65 \times 4063 + 0{,}35 \times 2712 = 3590{,}2$$

Performance pondérée du processeur #2 :

$$\text{Perf}_2 = 0{,}65 \times 6245 + 0{,}35 \times 2086 = 4789{,}4$$

Ratio performance processeur #2 relativement au processeur #1 :

$$\text{gain}_2 = \frac{\text{Perf}_2}{\text{Perf}_1} = \frac{4789{,}4}{3590{,}2} = 1{,}334$$

(10) (b) Supposons maintenant qu'il existe une version modifiée du processeur #1 permettant une parallélisation complète des opérations à valeurs entières sur deux unités ALU. Calculez le gain en performance offert par ce système à deux unités ALU relativement à la version de base du processeur #1 ayant une seule unité ALU. Indiquez également si cette version à deux ALU du processeur #1 est plus intéressante en terme de performance que le processeur #2 de base (avec une seule unité ALU).

Solution: Loi d'Amdahl :

$$\text{gain} = \frac{1}{\text{frac}_{\text{seq}} + \frac{\text{frac}_{\text{par}}}{\text{gain}_{\text{par}}}}$$

$$\text{gain}_{\text{2ALU}} = \frac{1}{0{,}35 + \frac{0{,}65}{2}} = \frac{1}{0{,}35 + 0{,}325} = 1{,}4815$$

Le processeur #1 avec deux unités ALU est plus intéressant que le processeur #2 de base, comme il offre de meilleurs performances, soit un gain de $1{,}4815$ comparativement à $1{,}334$.

Question 3 (20 points sur 100)

Soit le code suivant.

```
1              DADDIU  R2,R0,#8
2              DMULU   R3,R2,R1
3              LD      R6,0(R3)
4   Boucle:    DMULU   R3,R2,R1
5              LD      R7,0(R3)
6              DADDU   R8,R6,R7
7              SD      R8,0(R3)
8              DADDIU  R1,R1,#1
9              DSUBU   R4,R5,R1
10             BNEZ    R4,Boucle
11  Fin:       ...
```

(10) (a) Supposez que les valeurs suivantes (entiers non-signés sur 64 bits) sont stockés en mémoire et registres avant l'exécution du code.

Mem[0]:	2	R1:	0
Mem[8]:	3	R2:	45
Mem[16]:	5	R3:	3
Mem[24]:	7	R4:	18
Mem[32]:	11	R5:	7
Mem[40]:	13	R6:	8
Mem[48]:	17	R7:	31
Mem[56]:	19	R8:	1
Mem[64]:	23	R9:	8

Effectuez une exécution de ce code en utilisant ces valeurs et donnez le résultat de l'exécution de ce programme dans le tableau ici-bas.

Solution:

Mem[0]:	4	R1:	7
Mem[8]:	5	R2:	8
Mem[16]:	7	R3:	48
Mem[24]:	9	R4:	0
Mem[32]:	13	R5:	7
Mem[40]:	15	R6:	2
Mem[48]:	19	R7:	17
Mem[56]:	19	R8:	19
Mem[64]:	23	R9:	8

(10) (b) Donnez le détail de l'exécution, dans le tableau de la page suivante, d'une itération complète de ce programme dans un pipeline MIPS, jusqu'à l'instruction $i + 9$. Le pipeline ne fait pas d'envoi de données, mais une valeur inscrite en registre à l'étage WB au temps j sera disponible à l'étage EX au temps $j + 1$.

Solution:

			t	$t+1$	$t+2$	$t+3$	$t+4$	$t+5$	$t+6$
i	DADDIU	R2,R0,#8	IF	ID	EX	MEM	WB		
$i+1$	DMULU	R3,R2,R1		IF	ID	bulle	bulle	EX	MEM
$i+2$	LD	R6,0(R3)			IF	bulle	bulle	ID	bulle
$i+3$	DMULU	R3,R2,R1						IF	bulle
$i+4$	LD	R7,0(R3)							
$i+5$	DADDU	R8,R6,R7							
$i+6$	SD	R8,0(R3)							
$i+7$	DADDIU	R1,R1,#1							
$i+8$	DSUBU	R4,R5,R1							
$i+9$	BNEZ	R4,Boucle							

			$t+7$	$t+8$	$t+9$	$t+10$	$t+11$	$t+12$	$t+13$
i	DADDIU	R2,R0,#8							
$i+1$	DMULU	R3,R2,R1	WB						
$i+2$	LD	R6,0(R3)	bulle	EX	MEM	WB			
$i+3$	DMULU	R3,R2,R1	bulle	ID	EX	MEM	WB		
$i+4$	LD	R7,0(R3)		IF	ID	bulle	bulle	EX	MEM
$i+5$	DADDU	R8,R6,R7			IF	bulle	bulle	ID	bulle
$i+6$	SD	R8,0(R3)						IF	bulle
$i+7$	DADDIU	R1,R1,#1							
$i+8$	DSUBU	R4,R5,R1							
$i+9$	BNEZ	R4,Boucle							

			$t+14$	$t+15$	$t+16$	$t+17$	$t+18$	$t+19$	$t+20$
i	DADDIU	R2,R0,#8							
$i+1$	DMULU	R3,R2,R1							
$i+2$	LD	R6,0(R3)							
$i+3$	DMULU	R3,R2,R1							
$i+4$	LD	R7,0(R3)	WB						
$i+5$	DADDU	R8,R6,R7	bulle	EX	MEM	WB			
$i+6$	SD	R8,0(R3)	bulle	ID	bulle	bulle	EX	MEM	WB
$i+7$	DADDIU	R1,R1,#1		IF	bulle	bulle	ID	EX	MEM
$i+8$	DSUBU	R4,R5,R1					IF	ID	bulle
$i+9$	BNEZ	R4,Boucle						IF	bulle

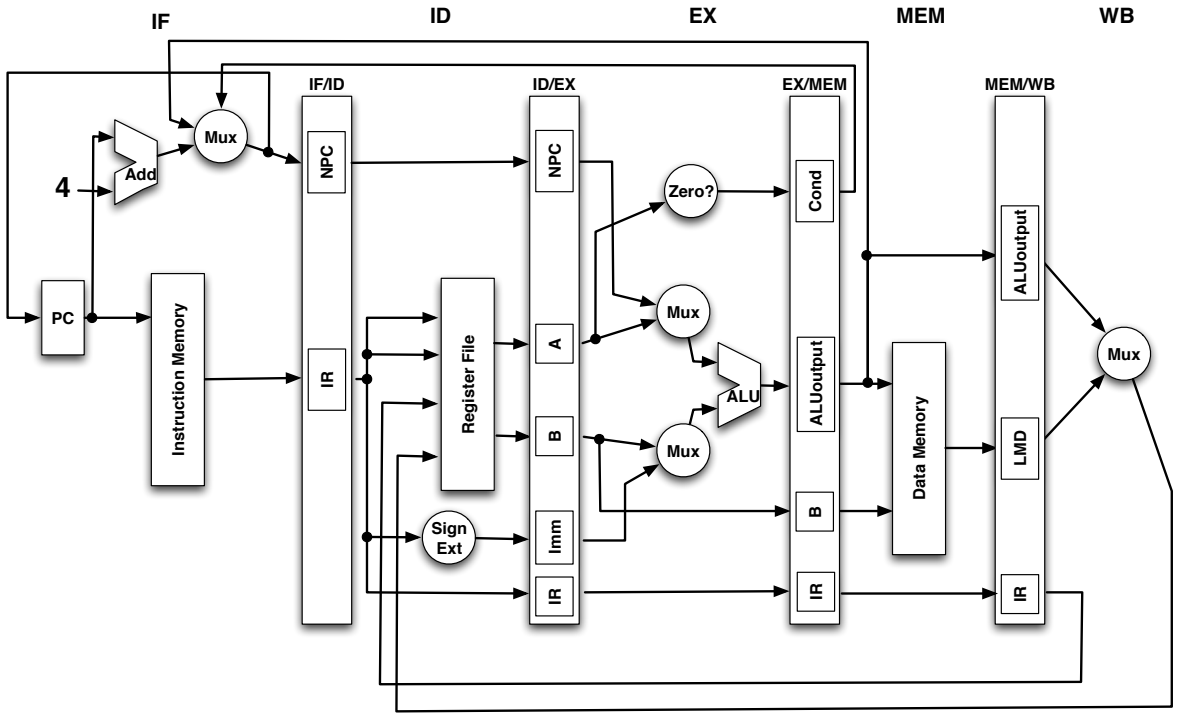
			$t+21$	$t+22$	$t+23$	$t+24$	$t+25$	$t+26$	$t+27$
i	DADDIU	R2,R0,#8							
$i+1$	DMULU	R3,R2,R1							
$i+2$	LD	R6,0(R3)							
$i+3$	DMULU	R3,R2,R1							
$i+4$	LD	R7,0(R3)							
$i+5$	DADDU	R8,R6,R7							
$i+6$	SD	R8,0(R3)							
$i+7$	DADDIU	R1,R1,#1	WB						
$i+8$	DSUBU	R4,R5,R1	bulle	EX	MEM	WB			
$i+9$	BNEZ	R4,Boucle	bulle	ID	bulle	bulle	EX	MEM	WB

L'instruction $i+9$ atteint donc l'étage IF au cycle d'horloge $t+19$.

Question 4 (16 points sur 100)

Soit la figure suivante, qui présente la micro-architecture du MIPS (32 bits) implantée en

pipeline, avec envoi de données, tel que vu en classe.



Supposons que ce processeur en pipeline exécute les instructions suivantes.

			t	$t+1$	$t+2$	$t+3$	$t+4$	$t+5$	$t+6$	$t+7$	$t+8$
i	`DADDIU`	`R3,R3,#32`	IF	ID	EX	MEM	WB				
$i+1$	`LD`	`R1,64(R3)`		IF	ID	EX	MEM	WB			
$i+2$	`DMULU`	`R2,R1,R1`			IF	ID	bulle	EX	MEM	WB	
$i+3$	`SD`	`R2,512(R3)`				IF	bulle	ID	EX	MEM	WB

(4) (a) Quelle est la valeur du registre `ID/EX.Imm` au cycle $t+2$?

Solution: La valeur du registre `ID/EX.Imm` est 32, soit la valeur de la constante utilisée dans l'addition de l'instruction i (`DADDIU`).

(4) (b) Quelle est la valeur du registre `EX/MEM.ALUoutput` au cycle $t+4$?

Solution: La valeur du registre `EX/MEM.ALUoutput` correspond à R3+64, soit l'adresse mémoire où sera lu la valeur stockée dans R1 par l'instruction $i+1$ (`LD`).

(4) (c) Quelle est la valeur du registre `EX/MEM.B` au cycle $t+7$?

Solution: La valeur du registre `EX/MEM.B` correspond à la valeur de R2, soit la valeur qui sera écrite en mémoire par l'instruction $i+3$ (`SD`).

(4) (d) Est-ce qu'il y aurait un moyen de réduire le temps d'exécution du programme, seulement en inversant l'ordre d'exécution d'instructions, sans changer la rectitude du programme ?

Si oui, indiquez quelles instructions il faudrait inverser et le nombre de cycles nécessaires à ce que la quatrième instruction (le `SD` à $i + 3$) atteigne l'étage WB.

Solution: Il n'est pas possible de réduire le temps d'exécution du programme seulement en inversant l'ordre d'exécution des instructions, car chaque instruction dépend du résultat de l'instruction précédente. Inverser l'ordre d'exécution des instructions briserait la rectitude du programme.

Question 5 (28 points sur 100)

Répondez aussi brièvement et clairement que possible aux questions suivantes.

(4) (a) La consommation en puissance des microprocesseurs dépend de quatre éléments : le nombre de transistors, la charge capacitive, le voltage d'opération et la fréquence du microprocesseur. Indiquez les éléments parmi ceux-ci qui ont contribué à l'augmentation continue de la puissance consommées par les microprocesseurs dans les 30 dernières années.

Solution: L'augmentation du nombre de transistors et de leur fréquence sont les principaux facteurs ayant contribué à l'augmentation de la puissance consommées dans les processeurs depuis les années 1980.

(4) (b) On dit que l'amélioration de la bande passante dans les systèmes informatiques est beaucoup plus rapide que l'amélioration de la latence. Expliquez l'impact de cette situation sur la performance actuelle des disques durs magnétiques.

Solution: Les disques durs magnétiques modernes ont une latence relativement grande comparativement à la banque passante, ce qui fait que l'obtention du premier élément d'un bloc prend un certain délais (latence), alors que l'on reçoit par la suite les éléments suivants du bloc à haut débit (bande passante). Ainsi, la lecture d'un bloc de données contiguës sera en général beaucoup plus rapide que la lecture de la même quantité de données distribuées sur le disque.

(4) (c) Indiquez pourquoi il est important de rendre disponible le code source des programmes utilisés dans les suites de benchmarks d'usage général.

Solution: Nous devons avoir accès à ce code source afin de pouvoir le compiler pour des architectures nouvelles, où la compatibilité avec les binaires d'architectures existante n'est pas possible. De plus, cet accès au code source permet de compiler le code à l'aide d'un compilateur exploitant bien les caractéristiques de l'architecture du microprocesseur.

(4) (d) Indiquez le principal avantage et le principal désavantage d'un encodage de taille fixe dans les jeux d'instructions.

Solution: Principal avantage : la logique matérielle pour décode les instructions est en général plus simple qu'avec un encodage de taille variable.
Principal désavantage : la taille des programmes informatique résultant risque d'être plus importante.

(4) (e) Indiquez l'intérêt d'utiliser des jeux d'instruction avec de nombreux registres d'usage général (comme avec MIPS), par opposition à une architecture avec moins de registres et des contraintes sur leur utilisation (comme avec x86).

Solution: L'usage de nombreux registres d'usage général donne beaucoup de liberté au compilateur dans l'assignation de valeurs dans des registres. Ainsi, on peut réduire le nombre de transfert de données entre le processeur et la mémoire vive, et également réduire le nombre d'instructions nécessaire uniquement pour passer des valeurs entre les registres uniquement pour satisfaire les contraintes imposée par l'architecture.

(4) (f) Expliquez pourquoi on veut généralement concevoir des architectures de microprocesseurs pipelinés où la charge de travail entre chaque étage du pipeline est aussi équilibrée que possible.

Solution: Dans un pipeline, on peut aller aussi rapidement que l'étage le plus lent. Donc, si la charge de travail est déséquilibré, on risque d'être fortement ralenti par les étages où la logique matérielle nécessaire pour effectuer tous les traitements est plus complexe.

(4) (g) Expliquez en quoi consiste un aléa de type « écriture après écriture » (*write after write*, WAW). Indiquez également dans quel type d'architecture de pipeline cet aléa est possible.

Solution: Un aléa de type « écriture après écriture » survient lorsque deux écritures d'une valeur dans un registre ou espace mémoire particulier ne surviennent pas dans le bon ordre relativement à la séquence d'un programme. Cet aléa est possible seulement lorsque le nombre d'étages du pipeline est variable selon le type d'instruction exécuté, par exemple avec les opérations à points flottants sur MIPS.