ARCHITECTURE DES ORDINATEURS
Examen Décembre 2008 (CORRIGÉ)
3H – Tous documents autorisés

PROGRAMMATION ASSEMBLEUR

Q 1) Que fait la suite des quatre instructions suivantes ?

```
LDW R9,0(R5)
LDW R10, 0(R4)
STW R10, 0(R5)
STW R9, 0(R4)
```

Permute le contenu de la case mémoire dont l'adresse est dans R5 avec le contenu de la case mémoire dont l'adresse est dans R4.

Q 2)Que fait le programme ci-dessous, qui travaille sur un tableau T[100] avec T[0] à l'adresse A000 0000H ? Donner le programme C correspondant.

```
        ORHI R2, A000H
        ADDI R3,R2, 40010 // Adresse T(100)
        MOV R4,R2       // adresse T(i=0)
Bouclei MOV R5,R4       //adresse T(j)
        ADDI R6,R4,4 //adresse T(k=i+1)
Bouclek:        LDW R7,0(R6) // T(k)
        LDW R8,0(R5) //T(j)
        BGE R7, R8, Suite //Si R7 <R8 alors adresse T[j] = adresse de T[k] soit j = k
        MOV R5,R6
Suite :  ADDI R6,R6,4 // k=k+1
        BLT R6, R3, Bouclek // si k+1 <N+1, aller à bouclek
        MOV R9,R8       // Cette instruction et les 3 suivantes permutent T[j] et T[i]
        LDW R10, 0(R4)
        STW R10, 0(R5)
        STW R9, 0(R4)
        ADDI R4,R4,4
        BLT R4,R3,Bouclei // tant que j+1 <N+1
```

```
for (i=0;i<100;i++)
{
j=i;
for (k=i+1;k<100;k++)
        if (a[k]<a[j])
                j=k;
tmp=a[j];
```

```
a[j]=a[i];
a[i]=tmp;
}
```
Tri insertion.
.

PIPELINE

Soit un processeur qui a les pipelines suivants

Entiers
LI DI LR EX AC LD ER

Flottants
LI DI LRF EX1 EX2 EX3 EX4 RT ER

avec la signification suivante :
LI : lecture des instructions dans le cache instructions
DI : décodage des instructions
LR : lecture registres entiers
LRF : lecture des registres flottants
EX : exécution UAL pour les entiers, et calcul des adresses (mémoire et branchements)
EXi : phase d'une exécution flottante, résolution des branchements (condition connue)
AC : accès au cache
LD : fin de l'accès cache (succès ou échec). Si succès, on dispose à la fin de LD de la donnée lue.
RT : Résolution des exceptions
ER : Écriture registres.
Tous les "bypass" nécessaires existent.

Q 3) Donner les latences des instructions source (colonne de gauche) lorsqu'elles sont suivies par les instructions destination (colonne de droite) pour un processeur scalaire (une instruction par cycle)

NB : une latence de n cycles signifie que deux instructions peuvent se suivre aux cycles i et i+n.

	Instruction source	Instruction consommateur	Latence
a	ADD R1,R2,R3	UALR4,R1,R2	1
b	SUB R1,R2,R3	ST R1,0 (R5)	1
c	AND R1,R2,R3	LD R6, 4(R1)	1
d	LDW R1, 0(R7)	XOR R2, R1, R3	3
e	OR R1, R2,R3	BGT R1,R9, déplacement	1
f	FADD F1,F2,F3	FSUB F4,F1,F5	4
g	LDF F1,0(R2)	FMUL F5,F1,F2	3
h	FSUB F1,F2,F0	STF F1,0 (R5)	3

a)

LI	DI	LR	**EX**	AC	LD	ER					
	LI	DI	LR	**EX**	AC	LD	ER				
b)											
LI	DI	LR	**EX**	AC	LD	ER					
	LI	DI	LR	EX	**AC**	LD	ER				
c)											
LI	DI	LR	**EX**	AC	LD	ER					
	LI	DI	LR	**EX**	AC	LD	ER				
d)											
LI	DI	LR	EX	AC	**LD**	ER					
	LI	DI	LR			**EX**	AC	LD	ER		
e)											
LI	DI	LR	**EX**	AC	LD	ER					
	LI	DI	LR	**EX**	AC	LD	ER				
f)											
LI	DI	LRF	EX1	EX2	EX3	**EX4**	RT	ER			
	LI	DI	LRF				**EX1**	EX2	EX3	EX4	RT
g)											
LI	DI	LR	EX	AC	**LD**	ER					
	LI	DI	LRF			EX1	EX2	EX3	EX4	RT	ER
h)											
LI	DI	LRF	EX1	EX2	EX3	**EX4**	RT	ER			
	LI	DI	LR	EX			**AC**	LD	ER		

CACHES.

On suppose que le processeur utilisé a un cache données de 64 Ko, avec des blocs de 64 octets. Il utilise l'écriture simultanée (write through) non allouée. On rappelle qu'avec l'écriture non allouée, il n'y a pas de défauts de cache en écriture

Soit le programme suivant P1

```
#define N 256
int X[N], Y[N], S ;
for (i=0 ; i<N ; i++)
S+= |X[i]-Y[i]|;
```

On suppose que les tableaux X[N], Y[N] sont rangés à partir de l'adresse hexadécimale $1000\ 0000_H$.

Q 4)

a) Quelles sont les adresses de X[0] et Y[0] ?

Un int.a 4 octets. Chaque tableau utilise 4 x 256 = 1024 octets
X[0] = 1000 0000 H
Y[0] = 1000 0400 H

b) Quel est pour ce cache le nombre de bits pour l'adresse dans le bloc, le nombre de bits d'index et le nombre de bits d'étiquette dans les deux cas suivants : 1) correspondance directe, 2) associativité quatre voies (quatre blocs par ensemble).

Les blocs ont 64 octets (6 bits). Il y a 64 Ko/64 = 1024 blocs. Un bloc contient 16 entiers 32 bits

En correspondance directe, on a 6 bits (adresse dans le bloc), 10 bits d'index et 16 bits d'étiquettes. En associativité quatre voies, on a 6 bits (adresse dans le bloc), 8 bits d'index et 18 bits d'étiquettes.

c) Quel est le nombre total de défaut de caches lors de l'exécution du programme P1 pour les deux cas suivants : a) correspondance directe, b) associativité deux voies (deux blocs par ensemble) ?

En correspondance directe, X[0] va dans le bloc 0 et Y[0] va dans le bloc 16. Il n'y a pas de conflit. On a donc deux défauts de cache toutes les 16 itérations (1/8). Le nombre total de défauts est de 256/8=32 défauts.
La situation est la même avec l'associativité quatre voies.

Q 5 : Pour quelle valeur minimale de N étant une puissance de 2 aura-t-on deux défauts de cache par itération avec la correspondance directe ?

Il y a conflit si les deux adresses de départ correspondent au même bloc du cache. Avec 6 bits d'adresse dans le bloc et 10 bits d'index, il faut que les deux adresses diffèrent à partir du $17^{ème}$ bit, soit par exemple
X[0] = 1000 0000H
Y[0] = 1001 0000H
Le tableau X a alors pour taille 2^{16} octets soit 2^{14} int. Ceci correspond à N =2^{14} = 16388

OPTIMISATION DE BOUCLES

On ajoute au jeu d'instructions NIOS II des instructions flottantes simple précision (32 bits) (Figure 2) et 32 registres flottants F0 à F31 (F0 est un registre normal).
Les additions, soustractions et multiplications flottantes sont pipelinées. Une nouvelle instruction peut démarrer à chaque cycle. Les latences sont de 2 cycles pour LF et de 4 cycles pour les instructions flottantes.
Les branchements ne sont pas retardés. On suppose qu'il n'y a pas de pénalité de branchement.

LF	2	LF ft, déplac(rs)	rt ← MEM [rs +SIMM]
SF	1	SF ft, déplac.(rs)	ft → MEM [rs + SIMM]
FADD	4	FADD fd, fs,ft	fd ← fs + ft (addition flottante simple précision)
FMUL	4	FMUL fd, fs,ft	fd ← fs * ft (multiplication flottante simple précision)
FSUB	4	FSUB fd, fs,ft	fd ← fs - ft (soustraction flottante simple précision)
F2I	1	F2I rd, ft	rd ← ft (copie d'un registre flottant dans un registre entier)

Figure 1 : Instructions flottantes ajoutées (Ce ne sont pas les instructions NIOS)

Soit le programme assembleur P1, qui travaille sur des tableaux de flottants simple précision (float) X[N] et Y[N] rangés successivement en mémoire, avec N = 256. L'adresse de X[0] est initialement contenue dans le registre R1. L'adresse de S est 100H.

```
        FSUB F0,F0,F0
        FSUB F3,F3,F3
        ADDI R2, R0,400_10
Boucle :LF F1, 0(R1)
        LF F2, 1024(R1)
        FSUB F1, F1,F2
        F2I R3,F1
        BGE R3,R0, suite
        FSUB F1,F0,F1
Suite :  FADD F3,F3,F1
        ADDI R1,R1,4
        BLT R1,R2, Boucle
        SF F3, 100_H(R0)
```

Q 6) Expliquer pourquoi la comparaison de nombres flottants peut se faire de la même manière que la comparaison de nombres entiers en complément à deux, ce qui justifie l'instruction F2I dans le code précédent

Dans la représentation flottante simple précision, le bit de poids fort est le bit de signe (0 pour positif ou nul et 1 pour négatif), exactement comme pour les nombres entiers en complément à 2. L'instruction F2I permet donc de récupérer le bit de signe du nombre flottant dans un registre entier et utiliser les instructions entières pour connaître le signe du nombre flottant.
Attention : l'instruction F2I n'est pas une instruction de conversion flottant vers entier. Dans le registre entier, on obtient la représentation binaire du nombre flottant, pas l'équivalent entier de la valeur flottante.

Q 7) Donner le code C correspondant au programme P1

```
#define N 100
float X[N], Y[N],S;
for (i=0 ; i<N ; i++)
      S+=fabs(X[i]-Y[i]);// fabs est la fonction valeur absolue en
                            flottant
```

Q 8) En montrant l'exécution cycle par cycle du programme assembleur P1, donner le nombre de cycles par itération de la boucle du programme assembleur P1 dans les deux cas (BGE pris et BGE non pris).

NB : cette question est faisable même sans avoir répondu à la question Q9

12 cycles si BGE est pris, et 16 cycles s'il est non pris

Q 9) Quel est le nombre de cycles par itération de la boucle après optimisation ?

11 cycles si BGE est pris et 15 cycles si BGE est non pris.

Question 8

	FSUB F0,F0,F0
	FSUB F3,F3,F3
	ADDI R2,R1,400
1	Boucle : LF F1, 0(R1)
2	LF F2, 1024(R1)
3	
4	FSUB F1,F1,F2
5	
6	
7	
8	F2I R3,F1
9	BGE R3,R0, suite
10	Suite : FADD F3,F3,F1
11	ADDI R1,R1,4
12	BLT R1,R2, Boucle
13	
14	SF F3, 100_H(R0)
15	

BGE pris

	FSUB F0,F0,F0
	FSUB F3,F3,F3
	ADDI R2,R1,400
1	Boucle : LF F1, 0(R1)
2	LF F2, 1024(R1)
3	
4	FSUB F1,F1,F2
5	
6	
7	
8	F2I R3,F1
9	BGE R3,R0, suite
10	FSUB F1,F0,F1
11	
12	
13	
14	Suite : FADD F3,F3,F1
15	ADDI R1,R1,4
16	BLT R1,R2, Boucle
17	
18	SF F3, 100_H(R0)

BGE non pris

Question 9

	FSUB F0,F0,F0
	FSUB F3,F3,F3
	ADDI R2,R1,1024
1	Boucle : LF F1, 0(R1)
2	LF F2, 1024(R1)
3	ADDI R1,R1,4
4	FSUB F1,F1,F2
5	
6	
à7	
8	F2I R3,F1
9	BGE R3,R0, suite
10	Suite : FADD F3,F3,F1
11	BLT R1,R2, Boucle
12	
13	SF F3, 100_H(R0)
14	

BGE pris

	FSUB F0,F0,F0
	FSUB F3,F3,F3
	ADDI R2,R1,1024
1	Boucle : LF F1, 0(R1)
2	LF F2, 1024(R1)
3	ADDI R1,R1,4
4	FSUB F1,F1,F2
5	
6	
7	
8	F2I R3,F1
9	BGE R3,R0, suite
10	FSUB F1,F0,F1
11	
12	
13	
14	Suite : FADD F3,F3,F1
15	BLT R1,R2, Boucle
16	
17	SF F3, 100_H(R0)

BGE non pris

Mis en forme : Normal

On suppose que l'on dispose d'une prédiction de branchement statique : les branchements avant sont prédits « non pris » et les branchements arrière sont prédits « pris ». Un branchement mal prédit a une pénalité de 3 cycles

Q 10) En tenant compte des pénalités de branchement et des comportements possibles de BGE et BLT, donnez

a) la valeur minimale du temps d'exécution du programme
b) la valeur maximale de temps d'exécution du programme

BGE est prédit « non pris » et BLT est prédit « pris »
1 erreur de prédiction pour BLT (fin de programme) et de 0 (BGE toujours pris) à 100 (BGE toujours non pris)
Temps « minimal : 100 BGE pris avec 100 mauvaises prédictions BGE et 1 mauvaise prédiction BLT + 2 cycles en sortie
(11 cycles +3 cycles)*100 + 3 +2 = 1405 cycles soit 14,05 cycles/itération
Temps « maximal » : 100 BGE non pris avec une mauvaise prédiction sur BLT et 2 cycles en sortie
15 *100 + 3 cycles + 2 cycles =1505 cycles soit 15,05 cycles/itérations

Q 11) Peut on définir une instruction FABS (valeur absolue) sur un registre flottant ? Que doit faire cette instruction ?

Oui. Mettre à 0 le bit de signe (bit 31) du nombre flottant pour le rendre positif.

On suppose maintenant qu'il existe l'instruction FABS avec une latence de 1

Q 12) Quelle est maintenant le nombre de cycles par exécution de la boucle optimisée ? Quel est le facteur de déroulage maximal que l'on peut utiliser et que devient le nombre de cycles par itération ?

	FSUB F0,F0,F0
	ADDI R2,R1,1024
1	Boucle :LF F1, 0(R1)
2	LF **F2**, 1024(R1)
3	ADDI R1,R1,4
4	FSUB F1,F1,**F2**
5	
6	
7	
8	FABS (F1,F1)
9	FADD F0,F0,F1
10	BLT R1,R2, Boucle
P1	
P2	
P3	SF F0, 100_H(R0)

Nombre de cycles par itération : 10
Il y a 2 registres utilisés par itération + le registre F0 utilisé pour l'accumulation. Le facteur de déroulage max est donc 15. (2*15 <= 32)
Une itération utilise 2 LF, 1 FSUB, 1 FABS et 1 FADD et il y a 2 instructions de gestion de boucle (ADDI et BLT).
La boucle avec un déroulage d'ordre 15 aura donc 15*5+2 =77 instructions.
Soit 77/15 = 5.1 cycles/itération.