ARCHITECTURE DES ORDINATEURS
Examen Décembre 2007
3H – Tous documents autorisés

EXECUTION D'INSTRUCTIONS

On utilise le jeu d'instructions NIOS II, y compris les pseudo-instructions.

Q 1) Donner l'instruction ou la suite des instructions NIOS II pour effectuer les actions suivantes :

a) Mettre à zéro le registre R1
b) Mettre 00008000_H dans le registre R2
c) Mettre $F000A000_H$ dans le registre R3
d) Diviser par 2 le contenu du registre R4, interprété en signé
e) Multiplier par 17 le contenu du registre R5
f) Mettre à zéro les cases mémoire entre les adresses F0000000 et $F000FFFF_H$

PROGRAMMATION ASSEMBLEUR

Q 2) Ecrire une procédure NIOS II qui vérifie si un caractère ascii, contenu dans l'octet de poids faible de R2, est un chiffre, et renvoie 1 dans R2 si vrai et 0 sinon. (On suppose que les bits 8 à 31 de R2 sont à 0).

On rappelle que les chiffres sont codés entre 30_H (0) et 39_H (9)

PIPELINES

Un processeur a les pipelines suivants pour les trois types d'instructions travaillant sur les entiers

- Instructions arithmétiques simples
  LI DI LR EX ER
- Instructions mémoire
  LI DI LR EX AM ER
- Instructions de multiplication
  LI DI LR M1 M2 M3 ER

Les différents étages ont la signification suivante :

- LI à l'accès au cache instructions (lecture de l'instruction)
- DI : décodage
- LR : Lecture des opérandes dans le banc de registres
- EX : Calculs UAL pour les opérations arithmétiques et logiques et pour les calculs d'adresse mémoire et de branchement.
- AM : accès au cache données
- M1, M2, M3 : multiplication entière pipelinée
- ER : Ecriture du résultat dans le banc de registres.

Tous les circuits d'anticipation (bypass) existent.

Q 3) Donner les latences entre instruction producteur et instruction consommateur en nombre de cycles, pour les cas suivants.

NB : la valeur n signifie que deux instructions dépendantes peuvent se suivre aux cycles i et i+n. UAL correspond aux instructions arithmétiques et logiques simples. MUL correspond à la multiplication. LDW et STW sont les instructions de chargement et de rangement

	Instruction producteur	Instruction consommateur	Latence
a	UAL Ri, -, -	UAL -, Ri, -	
b	UAL Ri, -, -	STW Ri, ()	
c	UAL Ri, -, -	LDW Rj, (Ri)	
d	LDW Ri, ()	UAL -, Ri, -	
e	UAL Ri	MUL -,Ri,-	
f	LDW Ri, ()	MUL -,Ri,-	
g	MUL Ri,	STW Ri, ()	
h	LDW Ri, ()	STW Ri, ()	

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
float X[N], Y[N], Z[N] ;
for (i=0 ; i<N ; i++)
if (X[i] > Y[i]) Z[i] = X[i]-Y[i] ;
      else Z[i] = 0.0 ;
```

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

Q 4)

a) **Quelles sont les adresses (hexadécimal) de X[0], Y[0] et Z[0] ?**

b) **Quel est pour ce cache le nombre de bits pour le déplacement 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 deux voies (deux blocs par ensemble).**

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) ?**

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 ?

OPTIMISATION DE BOUCLES

On ajoute au jeu d'instructions NIOS II des instructions flottantes simple précision (32 bits) (Figure 1) 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 3 cycles pour les instructions flottantes.
L'instruction FCMLT Fd, Fs, Ft est une instruction de transfert conditionnel. Si Ft <0.0 alors le contenu de Fs est copié dans Fd. Sinon, l'instruction ne fait rien (NOP). Dans les deux cas, sa latence est de 3.
Les branchements ne sont pas retardés. On suppose qu'il n'y a pas de pénalité de branchement.

LF	2	LF Ft, imm16(rs)	Ft ← MEM [rs +simm16]
SF	1	SF Ft, imm16(rs)	Ft → MEM [rs + simm16]
FADD	3	FADD Fd, Fs,Ft	Fd ← Fs + Ft (addition flottante simple précision)
FMUL	3	FMUL Fd, Fs,Ft	Fd ← Fs * Ft (multiplication flottante simple précision)
FSUB	3	FSUB Fd, Fs,Ft	Fd ← Fs - Ft (soustraction flottante simple précision)
FDIV	12	FDIV Fd,FS,Ft	Fd ← Fs / Ft (division flottante simple précision)
FCMLT	3	FCMOVLT Fd, Fs, Ft	Si Ft <0.0 alors Fd ← Fs sinon NOP

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

Soit le programme assembleur P2, qui travaille sur des tableaux de flottants simple précision (float) X[N], Y[N] et Z[N] rangés successivement en mémoire, avec N = 256. L'adresse de X[0] est initialement contenue dans le registre R1.

```
        FSUB F0,F0,F0
        ADD R2, R1, 1024
Boucle :LF F1, 0(R1)
        LF F2, 1024(R1)
        FSUB F1,F1,F2
        FCMLT F1,F0,F1
        SF F1,2048(R1)
        ADDI R1,R1,4
        BLT R1,R2, Boucle
```

Q 6) Donner le code C correspondant au programme P2

Q 7) En montrant l'exécution cycle par cycle du programme assembleur P1 (après optimisation), donner le nombre de cycles par itération de la boucle du programme assembleur P1.

NB : cette question est faisable même sans avoir répondu à la question Q6

Q 8) Donner le programme assembleur optimisé et l'exécution cycle par cycle avec un déroulage de boucle d'ordre 2. Quel est maintenant le nombre de cycles par itération de la boucle initiale ?

Q 9) Donner directement (sans détailler l'exécution cycle par cycle) le nombre de cycles par itération de la boucle initiale pour un déroulage de boucle d'ordre 4

PROGRAMMATION SIMD IA32

Soient les define pour le jeu d'instructions IA32

#define ld16(a)	_mm_load_si128(&a)	//chargement aligné (entiers)
#define st16(a, b)	_mm_store_si128(&a, b)	//rangement aligné (entiers)
#define orh(a,b)	_mm_or_si128(a,b)	//ou logique
#define xorh(a,b)	_mm_xor_si128(a,b)	//ou exclusif
#define maxhs(a,b)	_mm_max_epi16(a,b)	// max sur 8 x16 bits signés
#define minhs(a,b)	_mm_min_epi16(a,b)	// min sur 8 x 16 bits signés
#define addhs(a,b)	_mm_add_epi16(a,b)	// addition 8 x 16 bits signée
#define subhs(a,b)	_mm_sub_epi16(a,b)	//soustraction 8 x 16 bits signée

Soit les variables _m128i a, b, c ;

Q 10) Que font les instructions suivantes

b = xorh (a, a) ;
c= subhs (a, a) ;

Q 11) Que fait le programme suivant ?

```
#define N 1024
_mi128 X[N/8], A, tmp1, tmp2 ;

      A = xorh (A,A) ;
for (i=0 ; i< N/8 ; i++) {
      tmp1 = ld16 (&X[i]) ;
      tmp2 = subhs (A,tmp1) ;
      tmp1 = maxhs (tmp1,tmp2)
      st16 (&X[i], tmp1) ;}
```

Q 12) Ecrire la version scalaire (sans types ni intrinsics IA-32) du programme C équivalent.