Université Paris Sud D. Etiemble
Licence d'informatique 2004-2005

ARCHITECTURE DES ORDINATEURS
Examen Janvier 2005
1H 30 - Tous documents autorisés

PARTIE 1 : EXECUTION D'INSTRUCTIONS

On considère que les registres du processeur décrit en annexe contiennent les huit chiffres hexadécimaux suivants :

R0	0000 0000
R1	1234 5678
R2	8FFF 0000
R3	ABCD EF01
R4	FFFF FFFF
R5	8765 4321

Q1) Donner le contenu des registres R7 à R10 (sous forme de huit chiffres hexadécimaux) après exécution des instructions suivantes. Indiquer les cas de débordement

a) ADD R7, R2, R4
b) ADD R8, R2 ,R5
c) SUB R9, R1, R5
d) SUB R10, R3, R1

Q2) Donner le contenu des registres R12 à R16 (sous forme de huit chiffres hexadécimaux) après exécution des instructions suivantes :
e) OR R12, R1, R3
f) AND R13, R1, R3
g) XOR R14, R1, R3

Les registres F0 et F1 contiennent les huit chiffres hexadécimaux suivants :

F0	4000 0000
F1	BF80 0000

Q3) Donner le contenu de F2 après exécution de l'instruction ADDF F2, F1, F0

PARTIE 2 : PROGRAMMATION ASSEMBLEUR

Q4) Donner le code C correspondant au programme assembleur suivant où R5 contient l'adresse de départ d'un tableau T d'entiers. Que fait ce programme ?

```
ADDI R4, R5, 396        // Valeur décimale
ADD R1, R0,R0
SW R1, 0(R5)
ADDI R5, R5, 4
ADDI R2, R0, 1
```

```
        SW R2, 0(R5)
Boucle : ADDI R5,R5, 4
        ADD R3, R2,R1
        SW R3, 0(R5)
        ADD R1, R2, R0
        ADD R2, R3, R0
        BNEQ R4, R5, Boucle
```

PARTIE 3 : CACHES

On considère que le processeur décrit en annexe a un cache donnée de 8 Ko, avec des blocs (lignes) de 16 octets. Il utilise la réécriture (write back) avec écriture allouée (write allocate)

- Rappel : lors d'un échec cache en écriture, les caches à « écriture allouée » (write allocate) commencent par charger le bloc manquant depuis la mémoire principale. Il y a ensuite écriture dans le bloc de cache qui vient d'être chargé.
-

Le processeur exécute le programme C suivant :

```
float x[n], y[n], z[n] ;
int i ;
for (i=0 ; i<100 ; i++)
      z[i]=x[i]+y[i] ;
```

Les adresses de début des tableaux sont respectivement

@X[0] = F000 0000_H
@Y[0] = F000 2000_H
@Z[0] = F000 4000_H

Q5) Donner le nombre d'échecs cache avec pénalité d'échec par itération de la boucle dans les hypothèses suivantes :

a) le cache est à correspondance directe
b) le cache est associatif quatre voies (4 blocs par ensemble)
c) le cache est associatif deux voies (2 blocs par ensemble)

Dans les cas b) et c), la politique de remplacement est le LRU.

Q6) Reprendre la question Q5 en supposant que les adresses de début des tableaux sont maintenant

@X[0] = F000 0000_H
@Y[0] = F000 0200_H
@Z[0] = F000 0400_H

PARTIE 4 : TEMPS D'EXECUTION

Le processeur est pipeliné. Il peut commencer une nouvelle instruction à chaque cycle.
Les latences des instructions sont les suivantes (une instruction i qui s'exécute au cycle c a une latence de n cycles si une instruction qui dépend de i ne peut commencer avant le cycle c+n.):

- Instructions entières dont ADDI et BNE : 1 cycle
- LF : 2 cycles, SF : 1 cycle

- ADDF 3 cycles
-

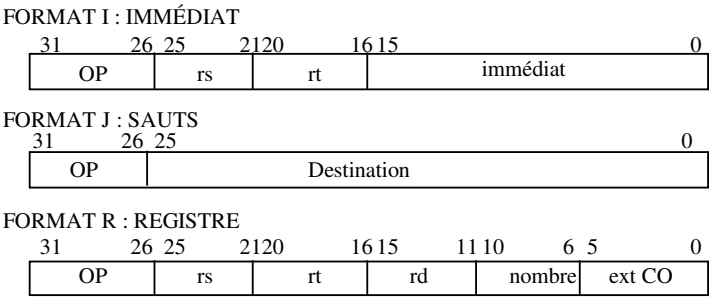
Soit le programme assembleur PA1 suivant :

```
        ADDI R4,R1,400        // 400 : valeur décimale
Boucle : LF F1, 0(R1)         // R1 contient initialement la valeur F000 0000H
        LF F2, 0(R2)          // R2 contient initialement la valeur F000 2000H
        ADDF F1, F1, F2
        SF F1, 0(R3)          // R3 contient initialement la valeur F000 4000H
        ADDI R1,R1, 4
        ADDI R2, R2, 4
        ADDI R3, R3, 4
        BNEQ R1,R4, Boucle
```

Dans cette partie, on considèrera des caches parfaits (pas d'échecs cache).
Q7) Quel est le temps d'exécution en cycles d'horloge par itération du programme PA1 ?
Q8) Donner une version PA2 du programme PA1 optimisée par simple réordonnancement des instructions pour réduire le temps d'exécution. Quel est alors le temps d'exécution en cycles d'horloge par itération de la version PA2 ?
Q9) Quel serait le temps d'exécution par itération de la boucle initiale avec un déroulage d'ordre 4 (4 itérations par boucle déroulée) pour la version PA2 ? On est dans le cas où le nombre d'itérations est un multiple de 4

ANNEXE

Le processeur considéré est une version simplifiée du MIPS R2000 pour la partie entière et complétée pour la partie flottante. Il a 32 registres entiers de 32 bits, notés R0 à R31. Le registre R0 est câblé à 0. Il a 32 registres flottants de 32 bits notés F0 à F31. Le registre F0 est normal.
Il y a trois formats d'instructions (figure 1)



OP : code opération
ext CO : extension du code opération

Figure 1 : Formats d'instructions.

Les instructions Load et Store utilisent le mode Immédiat. La mémoire est adressée par octet. Les comparaisons rangent le résultat (booléen vrai ou faux) dans un registre général. Le registre contient 1 (vrai) ou 0 (faux).

Les instructions utilisées sont données dans la figure 2.
IMM est l'immédiat sur 16 bits dans l'instruction. SIMM est l'immédiat de 16 bits étendu sur 32 bits avec extension de signe. ZIMM est l'immédiat de 16 bits étendu sur 32 bits avec 16 zéros à gauche. ADBRANCH est l'adresse de l'instruction suivante + SIMM

ADD	R	ADD rd, rs, rt	rd ← rs + rt (signé)
ADDF	R	ADDF fd, fs, ft	fd ← fs + ft (addition flottante simple précision)
ADDI	I	ADDI rt, rs, IMM	rt ← rs + SIMM (signé)
AND	R	AND rd, rs, rt	rd ← rs and rt
ANDI	I	ANDI rt, rs, IMM	rt ← rs and ZIMM
BEQ	I	BEQ rs,rt, déplac.	si rs = rt, branche à ADBRANCH
BGEZ	I	BGEZ rs,déplac.	si rs ≥ 0, branche à ADBRANH
BGEZAL	I	BGEZAL rs, déplac.	adresse de l'instruction suivante dans R31 si rs ≥ 0, branche à ADBRANH
BGTZ	I	BGTZ rs,déplac.	si rs > 0, branche à ADBRANH
BLEZ	I	BLEZ rs,déplac.	si rs ≤ 0, branche à ADBRANH
BLTZ	I	BLTZ rs,déplac.	si rs < 0, branche à ADBRANH
BLTZAL	I	BLTZAL rs, déplac.	adresse de l'instruction suivante dans R31 si rs < 0, branche à ADBRANH
BNEQ	I	BNEQ rs,rt, déplac.	si rs ≠ rt, branche à ADBRANCH
J	J	J destination	Décale l'adresse destination de 2 bits à gauche, concatène aux 4 bits de poids fort de CP et saute à l'adresse obtenue
JAL	J	JAL destination	Même action que J . Range adresse instruction suivante dans R31
JALR	R	JALR rs, rd	Saute à l'adresse dans rs. Range adresse instruction suivante dans rd
JR	R	JR rs	Saute à l'adresse dans rs
LF	I	LF ft, depl(rs)	ft ← MEM [rs + SIMM] : Chargement d'un flottant simple précision
LUI	I	LUI rt, IMM	Place IMM dans les 16 bits de poids fort de rt. Met 0 dans les 16 bits de poids faible de rt
LW	I	LW rt, déplac.(rs)	rt ← MEM [rs + SIMM]
OR	R	AND rd, rs, rt	rd ← rs or rt
ORI	I	ANDI rt, rs, IMM	rt ← rs or ZIMM
SF	I	SF ft, déplac.(rs)	ft → MEM [rs + SIMM] : rangement d'un flottant simple précision
SLL	R	SLL rd, rt, nb	Décale rt à gauche de nb bits et range dans rd
SLT	R	SLT rd, rs, rt	rd ← 1 si rs < rt avec rs signé et 0 autrement
SLTI	I	SLTI rt, rs, IMM	rt ← 1 si rs < SIMM avec rs signé et 0 autrement
SRA	R	SRA rd, rt, nb	Décaler (arithmétique) rt à droite de nb bits et ranger dans rd
SRL	R	SRL rd, rt, nb	Décaler (logique) rt à droite de nb bits et ranger dans rd.
SUB	R	SUB rd, rs, rt	rd ← rs - rt (signé)
SUBU	R	SUBU rd rs, rt	rd ← rs - rt(non signé)
SW	I	SW rt, déplac.(rs)	rt → MEM [rs + IMM]
XOR	R	XOR rd, rs, rt	rd ← rs xor rt
XORI	I	XORI rt, rs, IMM	rt ← rs xor ZIMM

Figure 2 : Jeu d'instructions