

ARCHITECTURE DES ORDINATEURS

Examen Juin 2007

3 H - Tous documents autorisés

NOMBRES FLOTTANTS

On représente sur 32 bits des nombres flottants avec les conventions suivantes :

Exposant e sur 8 bits en code excès 127 ; soit $E = e + 127$

Partie fractionnaire f sur 23 bits.

Le format est donné ci-dessous.

Un nombre flottant normalisé x est donné par

$x = 1.f \times 2^e$ si $0 < E < 255$ quand $s = 0$ (positif) ; NB : le 1 est implicite

$x = -1.f \times 2^e$ si $0 < E < 255$ quand $s = 1$ (négatif) ; NB : le 1 est implicite

$x = 0$ si $E=0$, $s = 0$ et $f = 0$

$x = \infty$ si $E=255$ et $f = 0$; $x = \text{NaN}$ si $E=255$ et $f \neq 0$.



Q 1) On considère les nombres normalisés positifs (zéro exclu).

- Quelle est la plus grande différence (écart) entre deux nombres normalisés positifs successifs ?
- Quelle est la plus petite différence (écart) entre deux nombres normalisés positifs successifs ?

Q 2) Donner la valeur décimale correspondant aux cas suivants (représentation hexadécimale de nombres flottants 32 bits)

- $4080\ 0000_H$
- $C1C8\ 0000_H$
- $7F80\ 0000_H$

Q 3) Donner les valeurs hexadécimales correspondant aux résultats des opérations suivantes

- $4080\ 0000_H + C1C80000_H =$
- $4080\ 0000_H \times C1C80000_H =$

EXECUTION D'INSTRUCTIONS

On utilise un sous ensemble du jeu d'instructions MIPS décrit en annexe.

Q 4) En partant à chaque fois du contenu de la table 1, donner le contenu des registres et des cases mémoire modifié après exécution des instructions MIPS ou séquences d'instructions. Dans chaque cas, on considérera que l'instruction est à l'adresse $1000\ 0000_H$

- ADD R8, R3, R4
- AND R9, R3, R4
- XOR R10, R3, R4
- BGEZ R0, +10 // 10 est un nombre d'instructions

- e) BGEZ R3, +10 // 10 est un nombre d’instructions
- f) LW R11, (R1+8)
- g) SRL R12,R3,8
- h) SRA R13,R3,8

Q 5) Donner l’instruction ou la suite des instructions MIPS pour effectuer les actions suivantes :

- a) Mettre 1 dans le registre R1
- b) Mettre à zéro les 4000 octets commençant à l’adresse mémoire F0000000_H (en supposant que le registre R2 contient F0000000_H)

Registre	Contenu (hexa)
R0	00000000
R1	00001000
R2	00001016
R3	8432A380
R4	ECDF1234
R6	00000020
R7	00000030

Adresse (hexa)	Contenu (hexa)
00001000	00000001
00001004	00000002
00001008	FEDC8765
0000100C	000000A4
00001010	00000005
00001014	00000006

Table 1 : contenu des registres du processeur (MIPS) et de cases mémoire

CACHES.

On suppose que le processeur utilisé a un cache données de 16 Ko, avec des blocs de 64 octets. Le cache utilise la réécriture avec écriture allouée (il y a des défauts de cache en écriture) Le processeur a des adresses sur 32 bits.

On considère le programme suivant, pour lequel les tableaux X et Y sont rangés successivement en mémoire à partir de l’adresse 1000 0000_H (adresse de X[0].)

```
double X[4096], Y[2048];
for (i=0 ; i<2048 ; i++)
    Y[i] = X[i+2048] - X[i] ;
```

Q6) 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 : a) correspondance directe, b) associativité quatre voies (quatre blocs par ensemble).

Q7) Quelles sont les adresses de X[2048] et Y[0] ?

Q8) Quel est le nombre total de défauts de caches lors de l’exécution du programme pour les deux cas suivants : a) correspondance directe, b) associativité quatre voies

OPTIMISATIONS DE PROGRAMME

On suppose une version pipelinée du processeur utilisant les instructions MIPS données en annexe. (Les branchements ne sont pas retardés : branchements normaux).

La latence des instructions est donnée dans la deuxième colonne des figures 3 et 4.

On rappelle qu’une latence de n signifie que si une instruction I démarre au cycle c , une instruction qui utilise le résultat de I ne peut démarrer qu’au cycle $c+n$. (une latence de 1 signifie qu’elle peut démarrer au cycle suivant).

La table 2 présente un programme C et le programme assembleur MIPS correspondant (On suppose que R3 contient au départ l’adresse de $X[0]$).

Q9) Quel est le temps d’exécution (en cycles par itération) de la boucle de la table 2. Optimiser la boucle et donner le nouveau temps d’exécution.

Table 2 : Programme C et programme assembleur

float X[100], S;	ADDI R5, R3, 400
int i ;	FSUB F0, F0, F0
for (i=0; i<100; i++)	Boucle :LF F1,(R3)
S+=X[i]*X[i];	FMUL F1,F1,F1
	FADD F0,F0,F1
	ADDI R3,R3,4
	BNEQ R3,R5, Boucle
	SF F0, (adresse de S)

PROGRAMME ASSEMBLEUR

Soit l’extrait de programme C suivant :

```
int x, y ;
main () {
while (x !=y) {
    if (x > y) x=x-y ;
    else y=y-x ; }
}
```

On suppose que la variable x est dans le registre R1 et la variable y dans le registre R2.

Q10) Ecrire le programme assembleur correspondant en utilisant les instructions MIPS fournies en annexe (branchements non retardés).

ANNEXE

Les figures donnent la liste des instructions disponibles. La signification des abréviations est la suivante :

IMM correspond aux 16 bits de poids faible d’une instruction.

ZIMM est une constante sur 32 bits, avec 16 zéros suivis de IMM (extension de zéros)

SIMM est une constante sur 32 bits, avec 16 fois le signe de IMM, suivi de IMM (extension de signe)

ADBRANCH est l’adresse de branchement, qui est égale à $NCP + SIMM$ (NCP est l’adresse de l’instruction qui suit le branchement).

ADD	1	ADD rd, rs, rt	$rd \leftarrow rs + rt$ (signé)
ADDI	1	ADDI rt, rs, IMM	$rt \leftarrow rs + \text{SIMM}$ (signé)
ADDIU	1	ADDIU rt, rs, IMM	$rt \leftarrow rs + \text{SIMM}$ (le contenu des registres est non signé)
ADDU	1	ADDU rd, rs, rt	$rd \leftarrow rs + rt$ (le contenu des registres est non signé)
AND	1	AND rd, rs, rt	$rd \leftarrow rs \text{ and } rt$
ANDI	1	ANDI rt, rs, IMM	$rt \leftarrow rs \text{ and } \text{ZIMM}$
BEQ	1	BEQ rs,rt, IMM.	si $rs = rt$, branche à ADBRANCH
BGEZ	1	BGEZ rs,IMM.	si $rs \geq 0$, branche à ADBRANCH
BGEZAL	1	BGEZAL rs, IMM.	adresse de l'instruction suivante dans R31 si $rs \geq 0$, branche à ADBRANCH
BGTZ	1	BGTZ rs,IMM.	si $rs > 0$, branche à ADBRANCH
BLEZ	1	BLEZ rs,IMM.	si $rs \leq 0$, branche à ADBRANCH
BLTZ	1	BLTZ rs,IMM.	si $rs < 0$, branche à ADBRANCH
BLTZAL	1	BLTZAL rs, IMM.	NCP dans R31. si $rs < 0$, branche à ADBRANCH
BNEQ	1	BNEQ rs,rt, IMM.	si $rs \neq rt$, branche à ADBRANCH
J	1	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	1	JAL destination	Même action que J. Range adresse instruction suivante dans R31
JR	1	JR rs	Saute à l'adresse dans rs
LUI	1	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
LB	2	LB rt, IMM(rs)	$Rt_{7-0} \leftarrow \text{MEM8}[rs + \text{SIMM}]$; $Rt_{31-8} \leftarrow$ extension de signe
LBU	2	LBU rt, IMM.(rs)	$Rt_{7-0} \leftarrow \text{MEM8}[rs + \text{SIMM}]$; $Rt_{31-8} \leftarrow$ extension de zéros.
LW	2	LW rt, IMM.(rs)	$rt \leftarrow \text{MEM}[rs + \text{SIMM}]$
OR	1	AND rd, rs, rt	$rd \leftarrow rs \text{ or } rt$
ORI	1	ANDI rt, rs, IMM	$rt \leftarrow rs \text{ or } \text{ZIMM}$
SLL	1	SLL rd, rt, nb	Décale rt à gauche de nb bits et range dans rd
SLT	1	SLT rd, rs, rt	$rd \leftarrow 1$ si $rs < rt$ avec rs signé et 0 autrement
SLTI	1	SLTI rt, rs, IMM	$rt \leftarrow 1$ si $rs < \text{SIMM}$ avec rs signé et 0 autrement
SLTIU	1	SLTIU rt, rs, IMM	$rt \leftarrow 1$ si $rs < \text{ZIMM}$ avec rs non signé et 0 autrement
SLTU	1	SLTU rt, rs, rt	$rd \leftarrow 1$ si $rs < rt$ avec rs et rt non signés et 0 autrement
SRA	1	SRA rd, rt, nb	Décaler (arithmétique) rt à droite de nb bits et ranger dans rd
SRL	1	SRL rd, rt, nb	Décaler (logique) rt à droite de nb bits et ranger dans rd.
SUB	1	SUB rd, rs, rt	$rd \leftarrow rs - rt$ (signé)
SUBU	1	SUBU rd rs, rt	$rd \leftarrow rs - rt$ (non signé)
SW	1	SW rt, IMM.(rs)	$rt \Rightarrow \text{MEM}[rs + \text{IMM}]$
XOR	1	XOR rd, rs, rt	$rd \leftarrow rs \text{ xor } rt$
XORI	1	XORI rt, rs, IMM	$rt \leftarrow rs \text{ xor } \text{ZIMM}$

Figure 1 : Instructions entières MIPS utilisées (NB : les branchements ne sont pas retardés)

LF	2	LF ft, IMM(rs)	$rt \leftarrow \text{MEM}[rs + \text{SIMM}]$
SF	1	SF ft, IMM.(rs)	$ft \rightarrow \text{MEM}[rs + \text{SIMM}]$
FADD	3	FADD fd, fs,ft	$fd \leftarrow fs + ft$ (addition flottante simple précision)
FMUL	3	FMUL fd, fs,ft	$fd \leftarrow fs * ft$ (multiplication flottante simple précision)
FSUB	3	FSUB fd, fs,ft	$fd \leftarrow fs - ft$ (soustraction flottante simple précision)
FDIV	12	FDIV fd,fs,ft	$fd \leftarrow fs / ft$ (division flottante simple précision)

Figure 2 : Instructions flottantes ajoutées (Ce ne sont pas les instructions MIPS)