ARCHITECTURE DES ORDINATEURS
Corrigé EXAMEN Novembre 2004

PARTIE 1 : REPRESENTATION DES NOMBRES

Q1) Représenter les nombres décimaux suivants sous forme de quatre chiffres hexadécimaux, en représentation complément à 2 sur 16 bits.

a) -1_{10}

FFFF

b) $+10\ 000_{10}$

2710

c) $-15\ 600_{10}$

C310

d) $+\ 40\ 000_{10}$

Impossible

Q2) Effectuer les opérations suivantes, en indiquant si le résultat est valide ou non

a) $49F5_H + 4932_H$

9327 débordement (positif + positif donne négatif)

b) $1470_H + ABCD_H$

C03D OK (addition de signes différents : jamais de débordement)

c) $1422_H - F012_H$

2410 OK (positif – négatif donne positif)$_H$

d) $8500_H - 2781_H$

5D7F Débordement (négatif – positif donne positif)

Q3) Donner la valeur décimale du nombre N dans le cas où les quatre chiffres hexadécimaux 8452_H représentent

a) un entier non signé

33874

b) un entier en complément à deux

-31662

c) un entier en représentation virgule fixe 1,15, où le premier bit est le bit de signe et les quinze bits suivants sont les bits après la virgule.

$-0{,}96624755859375 = -31662/2^{15}$

d) un nombre décimal en décimal codé binaire avec un chiffre décimal par groupe de 4 bits.

8452

PARTIE 2 : JEU D'INSTRUCTIONS ET PROGRAMMATION ASSEMBLEUR

Soit le processeur DLX

Tous les registres ont 32 bits.

Le processeur a 32 registres entiers, de R_0 à R_{31}. Le registre R_0 est câblé à 0 (R_0=0). On peut le lire, mais l'écriture dans R_0 ne produit aucun résultat. R_{31} contient l'adresse de retour des procédures. Il a 32 registres flottants, de F_0 à F_{31}. Il n'y a pas de registre flottant câblé à 0.
Les instructions sont de longueur fixe (32 bits). Le jeu d'instructions est donné en annexe.
La mémoire est adressable par octets.
NB : Les instructions de comparaison S__ positionnent le registre destination à 1 si la condition est vraie, et 0 si la condition est fausse.
Ex : SNE R_4, R_5, R_6 met R_4 à 1 si $R_5 \neq R_6$

La syntaxe assembleur est la suivante :
Instructions registres-registre
Code op, registre destination, registre source 1, registre source 2
Instructions registres-immédiat
Code op, registre destination, registre source 1, immédiat
Instructions mémoire
Code op, registre donnée, déplacement (registre adresse)
Instructions branchemet
Code op, registre, déplacement
Instructions de saut
Code op, déplacement ou Code op, registre

Q4) Quel est le nombre d'octets adressables par cette machine ?
Les registres ont 32 bits. Espace adressable : 2^{32} octets, soit 4 Go

Q5) Comment peut on exécuter avec le jeu d'instruction DLX les actions suivantes :
a) Mise à zéro du registre R_5
ADD R_5, R_0,R_0
Ou ADDI R5, R0, 0
Ou XOR R5, R5, R5
Ou SUB R5, R5, R5
Etc.
b) Incrémentation du registre R_4
ADDI R_4,R_4,1
c) Chargement de 4500_H dans le registre R_{10}
ADDI R_{10},R_0,4500_H
d) Chargement de la valeur FFFF 3000_H dans le registre R_2
LHI R_2,$FFFF_H$
ORI R_2,R_2,3000_H ou ADDI R_2,R_2,3000_H (attention à l'extension de signe. ORI est toujours OK, ADDI donne un résultat faux si la constante est négative)
e) Mise à zéro de la case mémoire d'adresse 0000 1000_H
SB R_0, 1000(R_0)
h) Mise à zéro des cases mémoire d'adresse 0000 8000_H à 0000 $80FF_H$
ORI R_1,R_0,80FC
SW R_0,0(R1)
ADDI R1,R1,-4

```
        BNEZ R1, -3
```

Q6) Soit les programmes assembleur P1 et P2.
R1 contient initialement l'adresse d'un tableau X de flottants simple précision;
R2 contient initialement l'adresse d'un tableau Y de flottants simple précision;
La variable S est à l'adresse 0000 1000
P1 :

```
                SUBF F0, F0, F0
                ADDI R3,0,128
Boucle :        LF F1, 0(R1)
                LF F2,0(R2)
                MULTF F1,F1,F2
                ADDF F0,F0,F1
                ADDI R1,R1,4
                ADDI R2,R2,4
                ADDI R3,R3,-1
                BNEZ R3, Boucle
                SF F0, 1000(R0)

P2              SUBF F9,F9,F9
                SUBF F10,F10,F10
                SUBF F11,FF1,F11
                SUBF F12,F12,F12
                ADDI R3,0,128
Boucle :        LF F1, 0(R1)
                LF F3, 4(R1)
                LF F5, 8(R1)
                LF F7,12(R1)
                LF F2,0(R2)
                LF F4, 4(R2)
                LF F6, 8(R2)
                LF F8, 12(R2)
                MULTF F1,F1,F2
                MULTF F3,F3,F4
                MULTF F5,F5,F6
                MULTF F7,F7,F8
                ADDF F9,F9,F1
                ADDF F10,F10,F3
                ADDF F11,F11,F5
                ADDF F12,F12,F7
                ADDI R1,R1,16
                ADDI R2,R2,16
                ADDI R3,R3,-4
                BNEZ R3, Boucle
                ADDF F9, F9, F10
                ADDF F11,F11,F12
                ADDF F9,F9,F11
                SF F9, 1000(R0)
```

a) donner le code C correspondant au programme P1
b) que représente le programme P2 par rapport au programme P1 ?

Programme P1 :

```
float X[128], Y[128] ; S ;
S=0.0 ;
for (i=0 ; i<128 ; i++)
      S+=X[i]*Y[i]; // produit scalaire
```

Le programme P2 est la version déroulé (déroulage d'ordre 4) du programme P1.

Q7) Comportement du cache données

Le processeur DLX a un cache de données de 8 Ko, à correspondance directe, avec des blocs de 16 octets.
On suppose que les deux tableaux X[N] et Y[N] de flottants simple précision de la question Q6 sont aux adresses :
Adresse de X[0] : $A000\ 0000_H$
Adresse de Y[0] : $B000\ 0000_H$

a) Quel est le nombre de défauts de caches pour exécuter le programme P1 ? (on donnera le nombre total de défauts de caches)

En correspondance directe, les accès à X[i] et Y[i] vont dans le même bloc de cache. Il y a donc 2 défauts de cache par itération, soit un total de 256 défauts de cache.

b) Quel est le nombre de défauts de caches pour exécuter le programme P2 ? (on donnera le nombre total de défauts de caches).

Le déroulage par 4 charge d'abord 4 éléments de X (soit un bloc de cache de 4 floats = 16 octets) puis 4 éléments de Y. Il y a donc seulement 2 défauts de cache pour 4 itérations, soit un total de 64 défauts de cache.

Si le cache est à « écriture alloué », il y a un défaut de cache supplémentaire dans les deux cas pour l'écriture de S.

Q8) Temps d'exécution
Le processeur DLX 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 i+n.):

- Instructions entières dont ADDI et BNE : 1 cycle
- LF : 2 cycles, SF : 1 cycle
- ADDF et MULTF : 4 cycles

Dans cette question, on considèrera des caches parfaits (pas de défauts de cache).
a) quel est le temps d'exécution total en cycles d'horloge du programme P1 ?

P1	SUBF F0, F0, F0
P2	ADDI R3,0,128
1 Boucle :	LF F1, 0(R1)
2	LF **F2**,0(R2)
3	
4	MULTF F1,F1,**F2**
5	
6	
7	
8	ADDF F0,F0,**F1**
9	ADDI R1,R1,4
10	ADDI R2,R2,4
11	ADDI R3,R3,-1
12	BNEZ R3, Boucle
1 Fin	SF F0, 1000(R0)

Temps boucle : 12 cycles
Temps total : 12x128 + 2 + 1 = 1539 cycles

b) optimiser la boucle P1 pour réduire son temps d'exécution. Quelle est le nouveau temps d'exécution total du programme P1 ?

P1	SUBF F0, F0, F0
P2	ADDI R3,0,128
1 Boucle :	LF F1, 0(R1)
2	LF **F2**,0(R2)
3	
4	MULTF F1,F1,**F2**
5	ADDI R1,R1,4
6	ADDI R2,R2,4
7	ADDI R3,R3,-1
8	ADDF **F0,**F0,**F1**
9	BNEZ R3, Boucle
S1	
S2	
S3 Fin	SF **F0**, 1000(R0)

Temps boucle : 9 cycles. Entrée boucle : 2 cycles. Sortie boucle : 3 cycles
Temps total : 9 x 128 + 2+3 = 1155 cycles

c) quel est le temps d'exécution total du programme P2 ?

1	SUBF F9,F9,F9	
2	SUBF F10,F10,F10	
3	SUBF F11,FF1,F11	
4	SUBF F12,F12,F12	
5	ADDI R3,0,128	

1 Boucle :	LF F1, 0(R1)	
2	LF F3, 4(R1)	
3	LF F5, 8(R1)	
4	LF F7,12(R1)	
5	LF F2,0(R2)	
6	LF F4, 4(R2)	
7	LF F6, 8(R2)	
8	LF F8, 12(R2)	
9	MULTF F1,F1,F2	
10	MULTF F3,F3,F4	
11	MULTF F5,F5,F6	
12	MULTF F7,F7,F8	
13	ADDF F9,F9,F1	
14	ADDF F10,F10,F3	
15	ADDF F11,F11,F5	
16	ADDF F12,F12,F7	
17	ADDI R1,R1,16	
18	ADDI R2,R2,16	
19	ADDI R3,R3,-4	
20	BNEZ R3, Boucle	
1	ADDF F9, F9, F10	
2	ADDF **F11**,F11,F12	
3		
4		
5		
6	ADDF **F9**,F9,**F11**	
7		
8		
9		
10	SF **F9**, 1000(R0)	

Boucle : 20 cycles pour 4 itérations soit 5 cycles par itération.
Prologue 5 cycles
Epilogue : 10 cycles
Temps total : 5 x 128 + 5 + 10 = 655 cycles

Q9) Pipeline

On suppose qu'une variante du processeur DLX a les pipelines suivants :
Instructions entières : 5 étages
LI DI/LR EX MEM ER
Instructions de chargement flottant (LF) : 5 étages
LI DI/LR EX MEM EF
Instructions flottantes (addition ou multiplication) : 7 étages
LI DI LF EX1 EX2 EX3 EF

avec la signification suivante :
LI : lecture des instructions dans le cache instructions
DI : décodage des instructions
LR : lecture registres entiers
LF : lecture des registres flottants
EX : exécution UAL pour les entiers, et calcul des adresses (mémoire et branchements)
EXi : phase d'une exécution flottante
MEM : accès au cache données
ER : Écriture registres entiers.
EF : Ecriture registres flottants
Tous les "bypass" nécessaires existent.

a) quel est la latence d'une instruction UAL entière lorsqu'elle est suivie d'une autre instruction UAL ?

LI LR **EX** MEM ER
LI LR **EX** MEM ER
Latence = 1

b) quelle est la latence d'une instruction de chargement de nombre flottant (LF) lorsqu'elle est suivie par une opération flottante (addition ou multiplication) ?

LI LR EX **MEM** EF
LI DI LF **EX1** EX2 EX3 EF
Latence = 1

c) quelle est la latence d'une instruction flottante (addition ou multiplication) lorsqu'elle est suivie par une autre instruction flottante ?

LI DI LF EX1 EX2 **EX3** EF
LI DI LF **EX1** EX2 EX3 EF
Latence = 3

d) quel est le délai de branchement pour les instructions du type BNEZ ?

LI LR **EX** MEM ER
LI LR EX MEM ER
Latence = 3
Délai de branchement = 2

e) quel est le délai de branchement des instructions de type JR ?
LI **LR** EX MEM ER
LI LR EX MEM ER
Latence = 2
Délai de branchement = 1

Jeu d'instructions DLX

Type d'instruction et code-op	Signification de l'instruction
Transferts de données	**Transfère les données entre des registres et la mémoire ; le seul mode d'adressage mémoire est (registre + déplacement signé de 16 bits)**
LB, LBU, SB	Chargement octet, octet non signé, rangement octet
LH, LHU, SH	Chargement demi-mot, demi-mot non signé, rangement demi-mot
LW, SW	Chargement mot, rangement mot (de/vers des registres entiers)
LF, SF	Chargement flottant simple précision, rangement flottant simple précision
Arithmétique et logique	**Opérations sur les données entières ou logiques dans des registres entiers;**
ADD, ADDI, ADDU, ADDUI	Addition, addition immédiats (tous les immédiats ont 16 bits). Les opérandes sont signés pour ADD et ADDI (avec extension de signe de l'immédiat). Les opérandes sont non signés pour ADDU et ADDUI (avec extension de 0 pour l'immédiat)
SUB, SUBU	Soustraction signée, non signée
AND, ANDI	Et, et immédiat (extension de 0 pour l'immédiat)
OR, ORI, XOR, XORI	Ou, ou immédiat, ou exclusif, ou exclusif immédiat (extension de 0 pour l'immédiat)
LHI	Chargement haut immédiat (charge la partie haute d'un registre avec un immédiat) et met à zéro la partie basse
SLL, SRL, SRA, SLLI, SRLI, SRAI	Décalages : sous forme immédiate (S__I) ou variable (S__); les décalages sont logique à gauche, logique à droite, et arithmétique
S__, S__I	Positionner la condition : "__" peut être EQ, NE, LT, GT, LE, GE
Contrôle	**Branchements conditionnels et sauts; relatifs CP ou par registre**
BEQZ, BNEZ	Branchement si registre entier égal/non égal à zéro; déplacement relatif de 16 bits ajouté à CP
J, JR	Sauts : déplacement de 26 bits ajouté à CP (J) ou destination dans le registre (JR)
Flottant	**Opérations flottantes sur le format simple précision (SP)**
ADDF	Addition de nombres SP
SUBF	Soustraction de nombres SP
MULTF	Multiplication SP
DIVF	Division SP