

ARCHITECTURE DES ORDINATEURS

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) -1₁₀
- b) +10 000₁₀
- c) -15 600₁₀
- d) + 40 000₁₀

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

- a) 49F5_H + 4932_H
- b) 1470_H + ABCD_H
- c) 1422_H - F012_H
- d) 8500_H - 2781_H

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

- a) un entier non signé
- b) un entier en complément à deux
- 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.
- d) un nombre décimal en décimal codé binaire avec un chiffre décimal par groupe de 4 bits.

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 R₀ à R₃₁. Le registre R₀ est câblé à 0 (R₀=0). On peut le lire, mais l'écriture dans R₀ ne produit aucun résultat. R₃₁ contient l'adresse de retour des procédures. Il a 32 registres flottants, de F₀ à F₃₁. 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 R₄, R₅, R₆ met R₄ à 1 si R₅ ≠ R₆

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 branchement

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 ?

Q5) Comment peut on exécuter avec le jeu d'instruction DLX les actions suivantes :

- a) Mise à zéro du registre R5
- b) Incrémentation du registre R4
- c) Chargement de 4500H dans le registre R10
- d) Chargement de la valeur FFFF 3000H dans le registre R2
- e) Mise à zéro de la case mémoire d'adresse 0000 1000H
- f) Mise à zéro des cases mémoire d'adresse 0000 8000H à 0000 80FFH

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, F11, 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)

```

- donner le code C correspondant au programme P1
- que représente le programme P2 par rapport au 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

- Quel est le nombre de défauts de caches pour exécuter le programme P1 ? (on donnera le nombre total de défauts de caches)
- Quel est le nombre de défauts de caches pour exécuter le programme P2 ? (on donnera le nombre total de défauts de caches).

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).

- quel est le temps d'exécution total en cycles d'horloge du programme P1 ?
- optimiser la boucle P1 pour réduire son temps d'exécution. Quelle est le nouveau temps d'exécution total du programme P1 ?

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

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.

- quel est la latence d'une instruction UAL entière lorsqu'elle est suivie d'une autre instruction UAL ?
- quelle est la latence d'une instruction de chargement de nombre flottant (LF) lorsqu'elle est suivie par une opération flottante (addition ou multiplication) ?
- quelle est la latence d'une instruction flottante (addition ou multiplication) lorsqu'elle est suivie par une autre instruction flottante ?
- quel est le délai de branchement pour les instructions du type BNEZ ?
- quel est le délai de branchement des instructions de type JR ?

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