ARCHITECTURE DES ORDINATEURS
PARTIEL Octobre 2007 – 2H (CORRIGÉ)

Pour toutes les questions, on utilise le jeu d'instructions NIOS II.

PARTIE 1 : Exécution d'instructions

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

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

Q 1) Donner le contenu des registres R6 à R11 (sous forme de huit chiffres hexadécimaux) après exécution des instructions suivantes.

a) ADD R6, R2, R1 ; R6 = 12335678
b) SUB R7, R1, R5 ; R7 = 8ACF1357
c) SLLI R8,R5, 4 ; R8= 765 43210
d) SRAI R9, R5,8 ; R9 = FF876543
e) SRLI R10, R2, 1 ; R10 = 7FFF8000
f) MUL R11, R1,R4 ; R11 = EDCB A988 (R4 = -1 donc R11 = -R1)

Q 2) Donner le contenu des registres R12 à R16 (sous forme de huit chiffres hexadécimaux) après exécution des instructions suivantes :

a) OR R12, R1, R3 ; R12 = BBFD FF79
b) AND R13, R1, R3 ; R13 = 0204 4600
c) XOR R14, R1, R3 ; R14 = B9F9 B979

Q 3) Donner le contenu (sous forme de huit chiffres hexadécimaux) du registre PC après exécution des instructions suivantes en supposant à chaque fois que l'adresse de l'instruction est:1000 0000_H.

a) BGT R4,R5, +8 ; R4>R5 donc pris PC = NPC+12 = 1000 $000C_H$
b) BGEU R4,R2,+12 ; R4>R2 donc pris PC =NPC+16 = 1000 0010_H
c) BNE R0,R0, +4 ; condition fausse PC = NPC=PC+4 = 1000 0004_H.

PARTIE 2 : Programmation assembleur

Soit le contenu des cases mémoires suivantes

Adresse mémoire	Contenu
$F0000000_H$	Partie basse de A
$F0000004_H$	Partie haute de A

$F0000008_H$	Partie basse de B
$F000000C_H$	Partie haute de B
$F0000010_H$	Partie basse de S
$F0000014_H$	Partie haute de S
$F0000018_H$	Débordement

Soient deux nombres A et B **non signés** de 64 bits dont les parties basses et hautes sont rangées en mémoire. On veut calculer la somme S sur 64 bits et ranger les résultats en mémoire. On range également en mémoire le résultat du débordement : 0 si résultat correct, 1 si débordement.

Q 4) Ecrire le programme assembleur NIOS II qui effectue la somme S = A+B

```
        ORHI R1,R0,F000H  // R1 contient l'adresse F0000000H
        LDW  R2, 0(R1)
        LDW  R3, 8(R1)
        ADD R4,R2,R3
        STW  R4, 16(R1)
        LDW R5,4(R1)
        LDW R6, 1210(R1)
        ADD R7,R6,R5
        BGT R4,R2,Suite        // CMPLT R8,R4,R2
        ADDI R7,R7,1           // ADD R7,R7,R8
Suite : STW R7, 2010(R1)
        MOV R8, R0
        BGT R7,R5, Fin         //CMPLT R9,R7,R5
        ADDI R8,R8,1           //ADD R8,R8,R9
Fin :   STW R8, 2410(R1)
```

NB : la variante droite évite les branchements conditionnels ;

PARTIE 3 : Désassemblage

Q 5) Donner les programmes C correspondants au programmes assembleur P1qui travaille sur un tableau X1 : Que fait ce programme ?

Programme P1

```
        MOV R2, N
        SLLI R2,R2,2
        ADDI R2, R1,R2
        LDW R2, 0(R1)
        MOV R3, R2
        MOV R4, R2
Boucle : ADDI R1,R1,4
        BGT R1,R2, FIN
        LDW R5,0(R1)
        BLE R2,R5, Suite
```

```
        MOV R2,R5
        BEQ R0,R0, Boucle
Suite : BGE R3,R5, Boucle
        MOV R3,R5
        BEQ R0,R0, Boucle
FIN :
```

```
P1 :
Int X1[N], Min, Max, i ;
Min = X1[0];
Max = X1[0];
For (i=1; i<N; i++) {
If (X[i] < Min) Min = X[i];
If (X[i] > Max) Max=X[i];
}
```

P1 calcule le MIN et le MAX d'un tableau de N entiers signés.

PARTIE 4 : Pipeline

Les instructions UAL et les instructions mémoire d'un processeur implantant le jeu d'instructions ARM ont les pipelines suivants :

UAL

LI1	LI2	LI3	DI	LR	EX1	EX2	EX3	ER

Mémoire

LI1	LI2	LI3	DI	LR	CA	LM1	LM2	ER

Avec les significations suivantes :
LI1 : 1ère étage de lecture de l'instruction
LI2 : 2ème étage de lecture de l'instruction
LI3 : 3ème étage de lecture de l'instruction
DI : Décodage
LR : Lecture des registres
EX1 : 1er étage d'exécution (décalage du deuxième opérande)
EX2 : 2ème étage d'exécution (opération UAL)
EX3 : 3ème étage d'exécution (fin opération UAL pour l'arithmétique saturée)
ER : Ecriture du résultat
CA : Calcul adresse mémoire
LM1 : 1er étage lecture donnée
LM2 : 2ème étage lecture donnée.

Q 6) En supposant que tous les circuits d'anticipation (« bypass ») soient disponibles, quel est le nombre de cycles de suspension (avec 0 quand il n'y a pas de suspension) dans les cas suivants :

a) LDW R1, 0(R2) suivi de ADD R3,R4, R1 LSL #3
b) ADD R1,R2,R3 suivi de SUB R5,R1,R6

a) 2 cycles

LI1	LI2	LI3	DI	LR	CA	LM1	**LM2**	ER						
	LI1	LI2	LI3	DI	LR			**EX1**	EX2	EX3	ER			

b) 2 cycles

LI1	LI2	LI3	DI	LR	EX1	EX2	**EX3**	ER						
	LI1	LI2	LI3	DI	LR			**EX1**	EX2	EX3	ER			

Q 7) Pour les instructions de branchement, l'adresse de branchement est disponible à la fin du cycle EX2. Quel est le délai de branchement ?

NB : on rappelle que le délai est de n lorsque n instructions suivant le branchement ont commencé à s'exécuter avant que l'instruction cible du branchement ait commencé à s'exécuter

Délai de branchement : 6 cycles

LI1	LI2	LI3	DI	LR	EX1	**EX2**	EX3	ER							
							LI1	LI2	LI3	DI	LR	EX1	EX2	EX3	ER

PARTIE 5 : Prédiction de branchements

1) Soit le code suivant C1 qui intervient dans une boucle.

```
If (x is odd)                  // branchement b1.
      then increment a ;       // branchement b1 non pris
                               //(NB : odd signifie "impair")
```

Le branchement b1 est non pris si la condition est vraie et pris si la condition est fausse.
Les prédicteurs sont initialisés à

- NP (non pris) pour le prédicteur 1 bit
- FNP (fortement non pris) pour le prédicteur 2 bits

On considère neuf itérations de la boucle, pour lesquelles x prend les valeurs 2, 3, 4, 5, 6, 7, 8, 9,10.

Q 8) Remplir le tableau suivant. Quel est le nombre de bonnes prédictions pour chaque prédicteur ?

Prédicteur 1 bit	X=	2	3	4	5	6	7	8	9	10
Prédiction		NP	P	NP	P	NP	P	NP	P	NP
Comportement de b1		P	NP	P	NP	P	NP	P	NP	P
Prédicteur 2 bits										
Prédiction		FNP	**fNP**	FNP	**fNP**	FNP	**fNP**	FNP	**fNP**	FNP
Comportement de b1		P	**NP**	P	**NP**	P	**NP**	P	**NP**	P

1 bit : 0 bonnes prédictions
2 bits : 4 bonnes prédictions

2) Soit le code suivant C2 qui intervient dans une boucle.

```
if (x is prime)                // branchement b2.
      then increment b ;       // branchement b2 non pris
                               // prime  signifie "nombre premier")
```

Le branchement b2 est non pris si la condition est vraie et pris si la condition est fausse.
Les prédicteurs sont initialisés à

- NP (non pris) pour le prédicteur 1 bit
- FNP (fortement non pris) pour le prédicteur 2 bits

Q 9) Remplir le tableau suivant. Quel est le nombre de bonnes prédictions pour chaque prédicteur

Prédicteur 1 bit	X=	2	3	4	5	6	7	8	9	10
Prédiction		**NP**	**NP**	NP	P	NP	P	NP	**P**	**P**
Comportement de b2		**NP**	**NP**	P	NP	P	NP	P	**P**	**P**
Prédicteur 2 bits										
Prédiction		**FNP**	**FNP**	FNP	**fNP**	FNP	**fNP**	FNP	fNP	**fP**
Comportement de b2		**NP**	**NP**	P	**NP**	P	**NP**	P	P	**P**

1 bit : 4 bonnes prédictions
2 bits : 5 bonnes prédictions