ARCHITECTURE DES ORDINATEURS
Corrigé Examen Juin 2012
3H – Tous documents autorisés
Les questions sont indépendantes

On utilise le jeu d'instructions NIOS II et le jeu d'instructions ARM.

EXECUTION D'INSTRUCTIONS NIOS

Q 1) Comment peut on exécuter avec le jeu d'instructions NIOS les actions suivantes :

a) Mise à zéro du registre R_5

ADD R_5, R_0,R_0

b) Incrémentation du registre R_4

ADDI R_4,R_4,1

c) Chargement de 4500_H dans le registre R_{10}

ORI R_{10},R_0,4500_H

d) Chargement de la valeur FFFF 3000_H dans le registre R_2

ORHI R_2, R0, $FFFF_H$

ORI R_2,R_2,3000_H

e) Mise à zéro de l'octet d'adresse 0000 1000_H

STB R_0, 1000(R_0)

h) Mise à zéro des cases mémoire d'adresse 0000 8000_H à 0000 $80FF_H$

Boucle :ORI R_1,R_0,8000

ADDI R1,R1,100H
STW R_0,0(R1)
ADDI R1,R1, 4
BNE R1,R2,Boucle

PROGRAMMATION ASSEMBLEUR

Q 2) Donner le code C correspondant au code assembleur NIOS

Le programme travaille sur un tableau d'entiers dont l'adresse du premier élément T[0) est à l'adresse 0x4000 0000

```
orhi r3, 0x4000
addi r4,r3,80
stw r0,0(r3)
add r1,r0,r0
addi r2,r1,1
```

```
     stw r2,4(r3)
     addi r3,r3,8
LOOP :
     add r5,r1,r2
     stw r5,0(r3)
     addi r3,r3,4
     add r1,r2,r0
     add r2,r5,r0
     ble r3,r4, LOOP
```

```
int T[20], I;
T[0]=0;
T[1]=1;
For (i=2; i<20;i++)
     T[i] = T[i-1] + T[i-2];
```

Q 3) Que fait le programme ?

Etablit le tableau des 20 premiers nombres de Fibonnaci »

i	T[i]
0	0
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55
11	89
12	144
13	233
14	377
15	610
16	987
17	1597
18	2584
19	4181

Soit le programme C suivant

```
int a, b ;
while (a != b) {
if (a>b) a=a-b;
if (a<b) b=b-a;}
```

On suppose que a et b ont été initialement chargés dans les registres r1 et r2 par des instructions ldr.

Q 4) Ecrire le code ARM de la boucle while.

```
Boucle :CMP  r1,r2
:        SUBGT r1,r1,r2
         SUBLT r2,r2,r1
         BNE Boucle
```

CACHES

On suppose qu'un processeur a un cache données de 32 Ko, avec des blocs de 64 octets. Le cache utilise l'écriture simultanée avec écriture non allouée (il n'y a pas de défauts de cache en écriture)
Le processeur a des adresses sur 32 bits.

Q 5) 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 pour un cache à correspondance directe

6 bits pour l'adresse dans le bloc, 9 bits pour l'index et 17 bits pour l'étiquette

Q 6) Dans quels blocs du cache vont les flottants X[0] et X[2048] qui sont aux adresses $1000\ 0000_H$ et $1000\ 2000_H$. ?

X[0] est dans le bloc 0 ; X[2048] est dans le bloc 128

Q 7) Quel est le nombre total de défauts de caches lors de l'exécution des boucles pour le cache à correspondance directe pour les trois programmes suivants?

On considère les programmes suivants, pour lequel le tableau X est rangé en mémoire à partir de l'adresse $1000\ 0000_H$ (adresse de X[0].)

```
float X[4096] ;
1)  for (i=0 ; i<2048 ; i++)
        S+ =X[i] ;
2) for (i=0 ; i<2032 ; i++)
        S+ =X[i] + X[i+16];
3) for (i=0 ; i<2048 ; i++)
      S+ =X[i] + X[i+2048];
```

Il y a 16 flottants par ligne de cache.

1) Il y a un défaut de cache toutes les 16 itérations, soit un total de 128 défauts
2) La première itération provoque 2 défauts (i=0 à 15, puis i=16 à 31) et toutes les itérations suivantes un seul défaut. Au total, il y a 128 défauts

3) Chaque itération provoque 2 défauts de cache toutes les 16 itérations, soit un total de 256 défauts. (Il n'y pas de conflit entre X[i] et X[i+2048].

OPTIMISATIONS DE PROGRAMME

On suppose une version pipelinée du processeur utilisant les instructions ARM.
La latence de toutes les instructions arithmétique et logique est de 1 cycle, sauf pour la multiplication entière MUL (32 bits x 32 bits et résultat sur 32 bits) qui a une latence de 4. Les instructions de chargement (LDR) ont une latence de 3 cycles ou 4 cycles (voir Table). 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 présente un programme C et le programme assembleur ARM correspondant (On suppose que R3 contient au départ l'adresse de X[0] et R4 contient l'adresse de Y[0])

Q9) Quel est le temps d'exécution (en cycles) de la boucle ci-dessus. Indiquer une optimisation possible et donner le nouveau temps d'exécution ?

12 cycles.
En remontant l'instruction SUBS, on obtient 11 cycles

Q10) Quel serait le temps d'exécution (en cycles par itération de la boucle initiale) avec un déroulage de boucle d'ordre 4 ?

Il n'y a plus de suspensions : 18 cycles pour 4 itérations, soit 4,5 cycles/itération

Programme C	Programme assembleur

int X[100], Y[100], S, i ;
for (i=0; i<100; i++)
S+=X[i]*Yi];
	MOV R5, 100
	MOV R0,#0
	Boucle :LDR R1, [R3], #4
	LDR R2, [R4], #4
	MUL R1,R1,R2
	ADD R0,R0,R1
	SUBS R5,R5,#1
	BGT Boucle

1 Boucle :LDR R1, [R3], #4	1 Boucle :LDR R1, [R3], #4	1 Boucle :LDR R1, [R3], #4
2 LDR R2, [R4], #4	2 LDR R2, [R4], #4	2 LDR R2, [R4], #4
3	3 SUBS R5,R5,#1	3 LDR R7, [R3], #4
4	4	4 LDR R8, [R4], #4
5	5	5 LDR R9, [R3], #4
6 MUL R1,R1,R2	6 MUL R1,R1,R2	6 LDR R10, [R4], #4
7	7	7 LDR R11, [R3], #4
8	8	8 LDR R12, [R4], #4
9	9	9 MUL R1,R1,R2
10 ADD R0,R0,R1	10 ADD R0,R0,R1	10 MUL R7,R7,R8

11 SUBS R5,R5,#1 12 BGT Boucle	11 BGT Boucle	11 MUL R9,R9,R10 12 MUL R11,R11,R12 13 ADD R0,R0,R1 14 ADD R0,R0, R7 15 ADD R0,R0, R9 16 ADD R0,R0,R11 17 SUBS R5,R5,#1 18 BGT Boucle

ANNEXE : Modes d'adressage et latences des instructions LDR

Mode d'adressage	Assembleur	Action	Latence
Déplacement 12 bits, Pré-indexé	[Rn, #déplacement]	Adresse = Rn + déplacement	3
Déplacement 12 bits, Pré-indexé avec mise à jour	[Rn, #déplacement] !	Adresse = Rn + déplacement Rn ← Adresse	4
Déplacement 12 bits, Post-indexé	[Rn], #déplacement	Adresse = Rn Rn ← Rn + déplacement	4