ARCHITECTURE DES ORDINATEURS
Examen Décembre 2009
3H – Tous documents autorisés – Les questions sont indépendantes

PROGRAMMATION ASSEMBLEUR (NIOS II)

Q 1) Que fait la fonction F1 suivante?

```
F1:     ADD R1, R0, R2
        BGE R2,R0, Suite
        SUB R1,R0,R2
Suite : JMP R31
```

Soit la fonction F2 suivante

```
F2 :    SRAI R3,R2,31
        XOR R4,R3,R2
        SUB R1, R4, R3
        JMP R31
```

Q 2) En supposant que R2 contienne la variable x représentant un entier signé

1. Après exécution de l'instruction SRAI, que contient R3 lorsque x ≥ 0 ? Que contient R3 lorsque x <0 ?
2. Après exécution de l'instruction XOR, que contient R4 lorsque x ≥ 0 ? Que contient R4 lorsque x <0 ?
3. Après l'exécution de l'instruction SUB, que contient R1 lorsque x ≥ 0 ? Que contient R1 lorsque x <0 ?

Q 3) Est-ce que les fonctions F1 et F2 renvoient le même résultat ? Si oui, quel peut être l'intérêt d'utiliser la fonction F2 au lieu de la fonction F1 ?

PREDICTIONS DE BRANCHEMENT

Un programme a cinq branchements conditionnels notés Branch 1 à Branch 5. Le comportement des branchements est le suivant pour une exécution du programme (P pour branchement pris et N pour branchement non pris).

```
Branch 1: P-P-P-P-P
Branch 2: N-N-N
Branch 3: P-N-P-N-P-N-P-N
Branch 4: P-P-P-N-N-N
Branch 5: P-P-P-N-P-P-P-N-P
```

On suppose que le comportement de chaque branchement reste le même pour chaque exécution du programme. Les prédicteurs 1 bit et 2 bits sont toujours initialisés dans le même état avant chaque séquence d'exécution d'un des branchements.

Q 4) Donner le nombre de bonnes prédictions pour chaque branchement avec les prédicteurs suivants :

a) **Toujours pris**
b) **Toujours non pris**
c) **Prédicteur 1 bit, initialisé à prédit pris.**
d) **Prédicteur 2 bits, initialisé à prédit faiblement pris.**

OPTIMISATION DE BOUCLES

On ajoute au jeu d'instructions NIOS II les instructions flottantes simple précision (32 bits) de la figure 1 et 32 registres flottants F0 à F31 (F0 est un registre normal).
Les additions, soustractions et multiplications flottantes sont pipelinées. Une nouvelle instruction peut démarrer à chaque cycle. Les latences sont de 2 cycles pour LF et de 3 cycles pour les instructions flottantes.
Les branchements ne sont pas retardés. On utilise une prédiction de branchement statique : les branchements avant sont prédits « non pris » et les branchements arrière sont prédits « pris ». Un branchement mal prédit a une pénalité de 3 cycles.

LF	2	LF ft, déplac(rs)	rt ← MEM [rs +SIMM]
SF	1	SF ft, déplac.(rs)	ft → MEM [rs + SIMM]
FADD	3	FADD fd, fs,ft	fd ← fs + ft (addition flottante simple précision)
FSUB	3	FSUB fd, fs,ft	fd ← fs - ft (soustraction flottante simple précision)

Figure 1 : Instructions flottantes ajoutées (Ce ne sont pas les instructions NIOS)

Soit le programme assembleur P1, qui travaille sur des tableaux de flottants simple précision (float) X[N] et Y[N] rangés successivement en mémoire, avec N = 128. L'adresse de X[0] est initialement contenue dans le registre R1. L'adresse de S est 100_H.

Programme P1

```
        FSUB F0,F0,F0
        ADDI R2, R1, 512
Boucle :LF F1, 0(R1)
        LF F2, 512(R1)
        FADD F1, F1,F2
        FADD F0,F0,F1
        ADDI R1,R1,4
        BLT R1,R2, Boucle
        SF F0, 100H(R0)
```

Q 5) Donner le code C correspondant au programme P1.

Q 6) En montrant l'exécution cycle par cycle du programme assembleur P1 :

1. **donner le nombre de cycles par itération de la boucle du programme assembleur P1 (après réordonnancement éventuel des instructions) sans tenir compte des pénalités de branchement ;**
2. **donner le nombre de cycles total pour l'exécution du programme en tenant compte des pénalités de branchement.**

Q 7) Refaire la question précédente avec un déroulage de boucle d'ordre 2 (2 itérations de la boucle initiale dans la boucle déroulée).

1. **Donner le code assembleur de la boucle déroulée et son exécution cycle par cycle**
2. **Donner le nombre de cycles par itération de la boucle initiale**
3. **Donner le nombre de cycles total pour l'exécution du programme.**

CACHES

Un processeur a un cache L1 de 16 Ko, avec des blocs (lignes) de cache de 64 octets. Il utilise l'écriture simultanée et l'écriture non allouée (pas de défauts de cache en écriture). Les adresses mémoire sont sur 32 bits.
Il y a deux variantes :

- Correspondance directe
- Associativité 4 voies (ensembles de 4 blocs)

Soit un programme C qui utilise la fonction vsum()

```
#define N 2048
double A[N], B[N];
double vsum (double *v1, double *v2, int n)
{
double s=0.0;
int i;
for (i=0; i<n ; i++)
      s+= v1[i] + v2[i];
return (s);
}
```

Q 8) Donner le nombre de bits des champs :

- **déplacement dans le bloc, numéro de bloc et étiquette en correspondance directe;**
- **déplacement dans le bloc et numéro d'index et étiquette en associativité 4 voies.**

Q 9) Les tableaux A et B sont alloués consécutivement en mémoire. A est placé à partir de l'adresse 00000040_H. Dans quels blocs du cache vont A[0] et B[0] en correspondance directe ? Dans quels ensembles du cache vont A[0] et B[0] en associativité 4 voies ?

Q 10) La variable s est en registre. Quel est le taux d'échecs (nombre défauts / nombre d'accès)

1. **pour un appel vsum (A, B, N) si le cache utilise la correspondance directe ?**
2. **pour un appel vsum(A, A, N) avec la correspondance directe ?**
3. **pour un appel vsum (A, B, N) avec l'associativité 4 voies ?**

Q 11) Que devient le taux d'échecs avec l'associativité 4 voies pour la fonction vmsum (A,A,N) suivante :

```
double vmsum (double *v1, double *v2, int n){
double s=0.0;int i;
for (i=0; i<n/8 ; i++)
      s+= v1[i] + v2[i*8];
return (s);}
```