ARCHITECTURE DES ORDINATEURS
Corrigé Examen Juin 2010
3H – Tous documents autorisés
Les questions sont indépendantes

On utilise le jeu d'instructions NIOS II.

PROGRAMMATION ASSEMBLEUR

Q 1) Ecrire la suite des instructions qui exécute le programme C ci-dessous. On supposera que les variables g, h, k sont aux adresses hexadécimales 20, 24 et 28.

```
int g, h, k
if (g>h) k=g;
      else k=h;
```

```
        LDW R1,20 (R0)
        LDW R2,24 (R0)
        BGT R1,R2, Suite
        ADD R1,R2,R0  // MOV R1,R2
Suite : STW R1,28 (R0)
```

Q 2) Ecrire la suite d'instructions qui calcule $y = 2^x$ (2 puissance x). La valeur x, inférieure ou égale à 30, est initialement dans le registre R2. La résultat sera dans le registre R1. On souhaite obtenir la version la plus rapide.

```
        ADDI R1, R0, 1
        SLL R1,R1,R2
```

Q 3) Ecrire en assembleur NIOS un programme calculant le nombre d'entiers 32 bits négatifs compris dans les cases d'adresses F000 0000 à F000 FFFC. Le résultat sera rangé dans R3.

```
        ORHI R1,R0, F000H
        OR R2,R1,FFFCH
        XOR R3,R3,R3 // Toutes les variantes pour R3=0 ;
Boucle :LDW R4,0(R1)
        BGE R4,R0, Suite
        ADDI R3,R3,1
Suite   ADDI R1,R1,4
        BLT R1,R2,Boucle
```

Q 4) Proposer une version de la question Q3 pour laquelle on n'utilise qu'un seul branchement par itération : le branchement de boucle.

```
        ORHI R1,R0, F000H
        OR R2,R1,FFFCH
        XOR R3,R3,R3 // Toutes les variantes pour R3=0 ;
```

```
Boucle :LDW R4,0(R1)
        SRLI R4,R4,31 // bit de signe décalé de 31 positions vers la droite ou CMPLT R4,R4,R0
        ADD R3,R3,R4
        ADDI R1,R1,4
        BLT R1,R2,Boucle
```

CACHES.

On suppose que le processeur utilisé a un cache données de 1 Ko à correspondance directe, avec des blocs de 32 octets.

Soit le code C

```
int SIZE, STRIDE;
int A[SIZE];
// Initialisation de SIZE et STRIDE
for(j=0;j<N;j++)
      for(i=0;i<SIZE;i=i+STRIDE)
            X+=A[i];
```

On suppose que N est très grand. On suppose également que toutes les valeurs exceptées celles du tableau A sont dans des registres.

Q 5) Quel est le taux d'échec (nombre d'échecs/nombre d'accès) lorsque STRIDE = 2 et SIZE = 256 ? Donner successivement

Le cache peut contenir 256 entiers, soit 32 blocs de 8 entiers.
Avec un stride de 2, un bloc contient 4 entiers
- le nombre d'accès pour la boucle interne
  `for(i=0;i<256;i=i+2)donne 128 accès`
- le nombre d'échecs cache pour le premier accès à la boucle interne
  `il y a 32 octets par ligne, soit 8 fois 32 bits, mais seulement 4 accès à A[i]. Il y a donc 128/4 = 32 échecs de démarrage`
- le nombre d'échecs cache pour les accès suivant à la boucle interne
  Aucun échec
- le taux d'échecs en fonction de N
  Taux d'échec = 32 /128 N = 1/4N
- le taux d'échecs lorsque N tend vers l'infini.
  Taux d'échec = 0%.

Q 6) Remplir le tableau ci-dessous (pour N très grand)

	STRIDE = 2	STRIDE = 4	STRIDE = 8
SIZE = 256	0 %	0%	0%
SIZE = 512	25%	50%	100%

EXECUTION DE BOUCLES

On ajoute au jeu d'instructions NIOS II des instructions flottantes simple précision (32 bits) (Figure 2) 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 4 cycles pour les instructions flottantes.
Les branchements ne sont pas retardés.

LF	2	LF ft, déplac(rs)	ft ← 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)
FMUL	3	FMUL fd, fs,ft	fd ← fs * ft (multiplication 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 P3, qui travaille sur des tableaux de flottants simple précision (float) X[N] et Y[N] rangés successivement en mémoire, avec N = 256. L'adresse de X[0] est initialement contenue dans le registre R1. F0 contient une constante flottante A..

```
        ADDI R2, R1,512₁₀
Boucle :LF F1, 0(R1)
        LF F2, 1024(R1)
        FMUL F1,F1,F0
        FADD F2,F2,F1
        ADDI R1,R1,4
        SF F2, 1020 (R1)
        BLT R1,R2, Boucle
```

Q 7) Donner le code C correspondant au programme P3

```
#define N 256
float X[N], Y[N], A;
for (i=0; i<N; i++)
Y[i] += A * X[i];
```

Q 8) En montrant l'exécution cycle par cycle du programme assembleur P3, donner le nombre de cycles par itération de la boucle du programme assembleur en supposant qu'il n'y a aucune pénalité de branchement. Quel est le temps d'exécution total en cycles d'horloge ?

```
        ADDI R2, R1,512₁₀
1Boucle :LF F1, 0(R1)
2       LF F2, 1024(R1)
3       FMUL F1,F1,F0
4       ADDI R1,R1,4
5
6       FADD F2,F2,F1
7
8
9       SF F2, 1020 (R1)
10      BLT R1,R2, Boucle
```

10 cycles

Temps total : 1 + 256 * 10 = 2561

Q 9) Quel est le nombre de cycles par itération de la boucle avec déroulage d'ordre 2, en supposant qu'il n'y a pas de pénalité de branchement ?

```
        ADDI R2, R1,512_10
1Boucle :LF F1, 0(R1)
2       LF F3, 4(R1)
3       LF F2, 1024(R1)
4       LF F4, 1028 (R1)
5       FMUL F1,F1,F0
6       FMUL F3,F3, F1
7       ADDI R1,R1,8
8       FADD F2,F2,F1
9       FADD F4,F4,F3
10
11      SF F2, 1016 (R1)
12      SF F4, 1020 (R1)
13      BLT R1,R2, Boucle
```

Soit 13/2 =6,5 cycles /itération
Temps d'exécution total : 1 +6.5*256 = 1665 cycles

Q 10) En déroulant au maximum la boucle, quel nombre de cycles par itération obtient-on ?

F0 est commun à toutes les itérations. Il faut deux registres flottants par itération. La boucle est donc déroulable 15 fois.

Il y a alors
30 LD et 15 SF, 15 FMUL et 15 FADD
1 ADDI et 1 BLT
Soit un total de 77 instructions pour 15 itérations = 5,13 cycles/itération.