Correction du contrôle continu 2 en Architecture des ordinateurs pour le HPC

S. Zertal

7 Avril 2017, Durée = 1h.15

Exercice 1 :

Soit le code suivant, écrit avec le jeu d'inscription vu en cours.

```
(1) ADDI R1,R0,#adr
(2) ADDI R2,R0,#0
(3) ADDI R9,R0,#9
(4) ADDI R6,R0,#100

(A) BNEZ R6, 8
(B) J 36
(C) LW R4,0(R1)
(D) SUB R4,R4,R9
(E) BNEZ R4,8
(F) ADDI R2,R2,#1
(G) SUBI R6,R6,#1
(V) ADDI R1,R1,#4
(Z) Lw R1,0(R1)
(W) J -36
(X) SW RES(R0), R2
```

1. Ce programme compte le nombre d'occurences du chiffre 9 sur les 100 premiers elements d'une liste chainée.

   ```
   struct elem{
   ```

```
int num;
struct elem *suiv;
}
int res;
struct elem *pt;
pt=adr; res = 0;

for(i=1;i<100;i++)
{ if (pt->num == 9)
      res ++;
  pt=pt->suiv;
  }
```

2. Les dépendances qui figurent dans ce code pour une itération

	RAR	RAW	WAR	WAW
R0	(2,1)(3,2) (4,3)(x,4)			
R1	(v,c)	(c,1)(Z,V)		
R2		(F,2)(x,f)		
R4		(D,C)(E,D)		
R6	(G,A1)	(A1,4)(A2,G)		
R9		(D,3)		

3. Les pénalités justifiées (on ne dispose d'aucune technique, ni même la subdivision du cycle) Les dependances RAR ont des pénalités nulles car il n y a pas de modifications (autre version) de la données.

Dep	Pen	Justif
R1(c,1)	0	Dist = $5cy > 4cy$
R1(Z,V)	3cy	Dist=1cy
R2(F,2)	0	$Dist > 4cy$
R2(X,F)	0	$Dist > 4cy$
R4(D,C)	3cy	Dist=1cy
R4(E,D)	3cy	Dist=1cy
R6(A1,4)	3cy	Dist = 1cy
R6(A2,G)	0cy	Dist = 4cy
R9(D,3)	0cy	Dist = 4cy

Exercice 2 :

On dispose d'une mémoire organisée en 2 bancs, numérotés 0 et 1 (chacun de cycle 4T) et fonctionnant en mode pipeline (i.e. capable de fournir en régime maximal un mot à chaque T). On considère le code suivant :

```
for (i=0 ; i < 1000 ; i++)
    f(B[i],C[i+1],A[2*i]);
```

Les tableaux B, C et A sont indicés de 0 à 1999 mais notre code ne manipule que 1000 éléments de chacun d'eux.
Le premier élément de chaque tableau est rangé sur un banc qu'on appele le banc de base. Les éléments suivants du tableau seront rangés cycliquement sur les bancs. Chaque requête émise vers la mémoire reste en attente tant que le banc concerné n'est pas disponible. De plus, il ne peut se produire plus qu'une requête mémoire par cycle.

Partie I :

On considère une exécution en mode scalaire, La séquence d'accès est :
B[0],C[1],A[0],B[1],C[2],A[2], ..., B[999],C[1000],A[1998].

1. En prenant en considération toutes les différentes combinaisons de base pour les tableaux A, B et C ; donnez le temps d'exécution de ce code.
   Il y a 4 bases differentes possibles (B,C,A) :
   (0,0,0) = 8000T
   (0,1,0) = 9000T
   (0,0,1) = 8500T
   (0,1,1) = 9003T

2. Le placement de base le plus optimal pour ces tableaux : qui permet l'exécution la plus rapide est celui selon la base (B=0,C=0,A=0)

3. L'accélération obtenue en choisissant le placement de la question précédente par rapport au placement de base (0,0,0) est 1 (c'est la réf même)

Partie II :

On considère une exécution en mode vectoriel avec un vecteur de taille 4. La séquence d'accès est :
B[0],B[1],B[2],B[3],C[1],C[2],C[3],C[4],A[0],A[2],A[4],A[6], ... , B[996],B[997],B[998],B[999],
C[997],C[998],C[999],C[1000],A[1992],A[1994],A[1996],A[1998].

1. Calcul du le temps d'exécution du code pour les différentes combinaisons du placement de base pour les 3 tableaux (B,C,A) :
   (0,0,0) = 8500T
   (0,1,0) = 8000T
   (0,0,1) = 8001T
   (0,1,1) = 8250T

2. La combinaison la plus optimale est (B=0,C=1,A=0)

3. Laccélération obtenue par la combinaison de la question précédente par rapport à celle la plus optimale du mode scalaire est
   Acc = 1 car le temps le plus optimal (pour des bases differentes) dans chaque mode d'accès est 8000T

4. En sachant que la machine vectorielle est 3 fois plus coûteuse que celle scalaire, On peut calculer l'efficacité du passage au vectoriel : 1/3 ce qui est largement inferieur à 1 et donc ce passage n'est pas avantageux du tout car je ne peux jamais faire mieux que le scalaire (meme pour la meilleure base.