

TD 5 (avec réponses)**1 Exercice 1**

On considère le programme suivant :

```
int i;
double a1 = m1;
double a2 = m2;
double a3 = m3;
double a4 = m4;
for (i=0; i<N; i++) {
    b1 = a1*m1 + a3*m2;
    b2 = a2*m1 + a4*m2;
    b3 = a1*m3 + a3*m4;
    b4 = a2*m3 + a4*m4;
    a1 = b1;
    a2 = b2;
    a3 = b3;
    a4 = b4;
}
```

On compile ce programme avec les options de compilation permettant de ne faire que des accès registres dans la boucle (on suppose qu'il y a suffisamment de registres disponibles), puis on l'exécute sur un processeur out-of-order de degré superscalaire 4 comportant un additionneur flottant double précision et un multiplieur flottant double précision. La fréquence d'horloge du processeur est de 2 Ghz. Tous les opérateurs sont complètement pipelinés. L'opération FMOV a une latence de 1 cycle et les opérations FADD et FMUL ont une même latence de X cycles, X étant une valeur qu'on cherchera à déterminer au cours de l'exercice.

1.1

Tracer le graphe de dépendance de données (dépendances RAW) correspondant à 2 itérations successives de la boucle. Les noeud du graphes sont les instructions FADD, FMUL, et FMOV, qu'on pourra représenter par les symboles $+$, \times et $=$.

1.2

Sachant que, lorsque $N = 10^9$, le temps d'exécution du programme vaut $T = 6.5$ secondes, calculez la valeur de X . On pourra supposer que le degré superscalaire (4) est suffisant et ne limite pas la performance, de même pour la fenêtre d'instructions du processeur.

1.3

Calculez le débit moyen de multiplications (en opérations par cycle).

1.4

De combien serait accélérée l'exécution si on supprimait complètement les conflits de ressources en ayant suffisamment d'opérateurs ?

2 Exercice 2

On considère la boucle ci-dessous :

```
boucle: F1 = LOAD R1
        F2 = F2 FADD F1
        R1 = R1 ADD 8
        R3 = R3 SUB 1
        BNZ R3, boucle
```

On suppose que R3 est initialisé avec une grande valeur N . On exécute la boucle sur un processeur out-of-order. On suppose que le degré superscalaire du processeur est suffisamment grand, et que la fenêtre d'instructions est de taille infinie. On suppose qu'il n'y a pas de conflits de ressource. L'addition sur les entiers a une latence de 1 cycle. L'addition en virgule flottante a une latence de 4 cycles. Calculez la performance en instructions par cycle.

3 Exercice 3

On considère la boucle ci-dessous, qui copie le contenu d'un tableau de flottants dans un autre tableau :

```
boucle: F1 = LOAD R1+R3
        STORE F1, R2+R3
        R3 = R3 ADD 8
        R4 = R4 SUB 1
        BNZ R4, boucle
```

On exécute cette boucle sur un processeur superscalaire out-of-order avec un cache de données de type write-back et des lignes de cache de 64 octets. Les opérations ADD et SUB ont une latence de 1 cycle. Les instructions LOAD et STORE ont une latence de 2 cycles en cas de hit dans le cache de données (1 cycle de calcul d'adresse + 1 cycle d'accès cache) et de 128 cycles en cas de miss.

On suppose que les tableaux sont beaucoup plus grands que le cache et que leurs adresses de début sont alignées sur des frontières de ligne de cache (cela signifie que l'adresse de chaque tableau est un multiple de 64). On suppose que l'associativité du cache est suffisante pour qu'une ligne chargée dans le cache ne soit pas évincée avant d'avoir été complètement utilisée. On suppose qu'il n'y a pas de mécanisme de préchargement.

3.1

En supposant qu'il n'y a aucune autre contrainte (fenêtre d'instructions illimitée, ressources d'exécution suffisantes, etc...), calculer la performance maximum en instructions par cycle.

3.2

Quel est le nombre maximum de miss de cache qui sont en cours simultanément ?

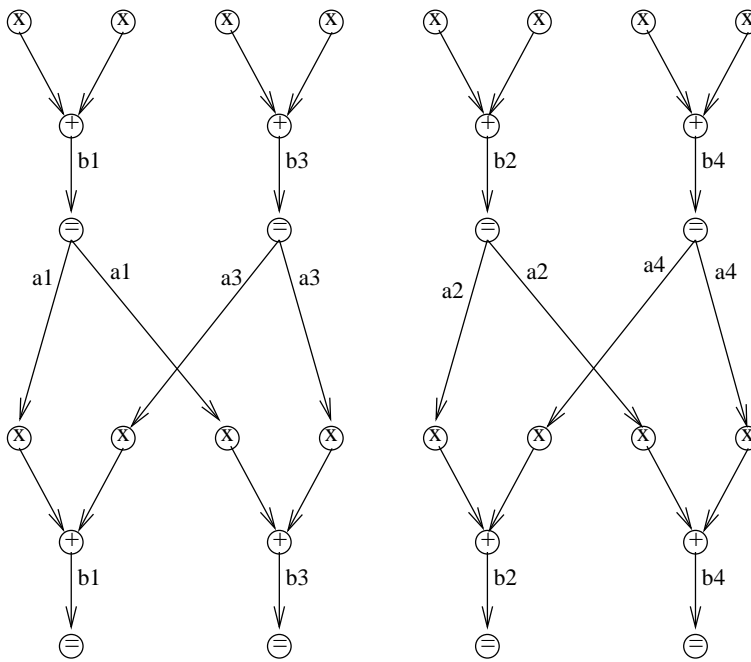
3.3

Quelle doit être la bande passante mémoire minimum (en octets par cycle) pour obtenir la performance maximum ?

1 Réponses de l'exercice 1

1.1

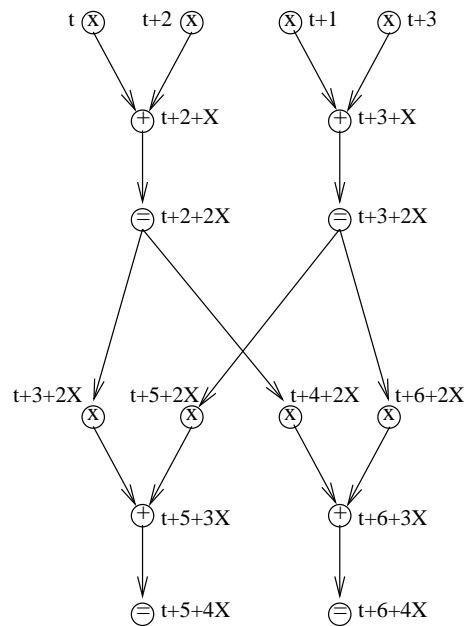
Le graphe de dépendances de données peut être représenté comme ci-dessous :



1.2

D'abord, on peut raisonnablement supposer que le branchement de la boucle est bien prédit (sauf à la dernière itération de la boucle, qu'on peut négliger car N est grand).

Nous annotons les opérations dans le graphe dépendance avec le premier cycle de l'opérations. Comme les sous-graphes sont identiques, nous pouvons simplifier la résolution de l'exercice en n'en considérant qu'un seul et en vérifiant que le résultat final est cohérent du point de vue de l'utilisation des ressources. Nous obtenons le sous-graphe ci-dessous :



Il faut remarquer que le fait d'avoir un seul multiplieur force les multiplications à s'effectuer à des cycles différents. Nous obtenons un scheduling périodique ou chaque période correspond à une itération de la boucle et vaut $2X + 3$ cycles. La durée totale de l'exécution vaut $(2X + 3)N = (2X + 3) \times 10^9$ cycles et est égale à $F \times T = 2 \times 10^9 \times 6.5$. La solution de l'équation est $X = 5$ cycles. Cette valeur est compatible avec le deuxième sous-graphe, puisque décaler le deuxième sous-graphe de 4 cycles par rapport au premier ne génère pas de conflit sur les opérateurs.

1.3

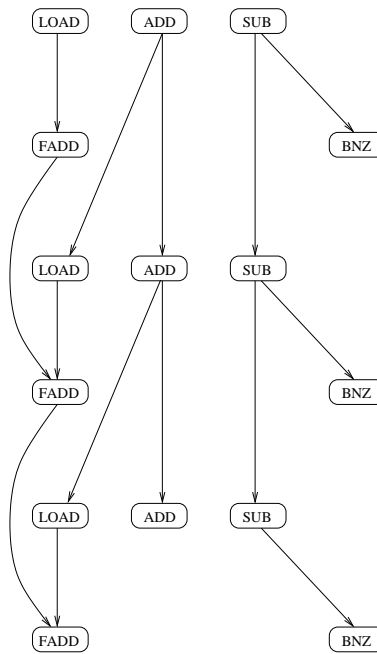
On fait 8 multiplications par itérations. On a un débit moyen de $8/(2X + 3) \approx 0.6$ multiplications par cycle.

1.4

S'il y avait suffisamment d'opérateurs, le temps d'exécution serait égal à la longueur (en cycles) du chemin critique du graphe de dépendance, qui vaut $(2X + 1)N$. L'accélération vaut $(2X + 3)/(2X + 1) = 13/11 \approx 1.18$.

2 Réponse de l'exercice 2

Voici le graphe de dépendance de données correspondant à 3 itérations :



Le ADD et le LOAD de l'itération k sont lancés au cycle k . Le FADD de l'itération k est lancé après le cycle $4(k - 1)$. La latence mémoire a une valeur finie, donc même si on fait des miss de cache, il existe un k tel que pour toutes les itérations supérieures à k , seul le FADD est sur le chemin critique. La longueur du chemin critique vaut asymptotiquement $4N$ cycles. La performance en instructions par cycle vaut

$$\frac{5N}{4N} = 1.25$$

3 Réponse de l'exercice 3

3.1

Le chemin critique du graphe de dépendance de données est constitué par la chaîne de ADD. Sa longueur en cycles est égale au nombre d'itérations. Comme il y a 5 instructions par itération, la performance maximum est de 5 instructions par cycle

3.2

On lance un LOAD à chaque cycle. Comme un load sur 8 est un miss, on a un miss de LOAD tous les 8 cycles. Pendant la durée d'un miss (128 cycles), 15 autres miss de LOAD sont lancés. On a donc à chaque instant 16 miss de LOAD en cours simultanément.

Le cache est write-back et on doit aussi tenir compte des miss faits par les STORE. Comme les tableaux sont alignés sur des frontières de ligne, lorsqu'on fait un miss sur un LOAD, le STORE

dépendant de ce LOAD fait également un miss. Ce miss est lancé 64 cycles après le miss du LOAD. On a donc à chaque instant 32 miss en cours simultanément.

3.3

On a un miss de LOAD et un miss de STORE tous les 8 cycles. Chaque miss fait une lecture mémoire de 64 octets (1 ligne). Le débit de lecture mémoire est donc de $2 \times 64/8 = 16$ octets par cycle. Mais attention, les tableaux sont beaucoup plus grands que le cache. Les lignes chargées dans le cache et que l'on a modifiées finissent par être éjectées du cache. Donc il faut tenir compte du débit des écritures mémoire. On fait un STORE par cycle. Cela correspond à 8 octets, et donc à un débit d'écriture de 8 octets par cycle. La bande passante totale doit être de 24 octets par cycle.