Université d'Orléans — DEA d'Informatique

Module "compilation avancée"

Sujet d'examen proposé par C. Eisenbeis et F. Thomasset

Mars 2001

<u>**Exercice 1**</u>

Exemple 1

```
    DO I = 1, 100
S:     X(I) = Y(I+2) + 1
T:     Y(I) = X(I+1) + 1 + X(I)
    END DO
```

1. Calculez les dépendances[1] existant entre les instructions de cette boucle, en précisant leur type : flot, anti, output.
   Formez le graphe de dépendances.
2. L'algorithme de distribution de boucle d'Allen-Kennedy permet-il de trouver du parallélisme ? Justifiez votre réponse.
3. Montrez qu'en introduisant un tableau auxilliaire, il est possible de transformer le code de cette boucle de sorte que la distribution de boucle produise des boucles parallèles.

<u>**Exercice 2**</u>

Exemple 2

```
DO I = 1, 50
   DO J = I, 200 - I
      X(I+1,J) = X(I,J+2)
               + X(I,J)
               + X(I+1,J)
               + X(I+2,J-1)
               + X(I+1,J+1)
   END DO
END DO
```

1. Calculez les vecteurs de dépendances[1] ; vérifiez que les dépendances sont uniformes. Précisez le type de chaque dépendance : flot, anti, output.
2. On cherche à construire une transformation unimodulaire de l'espace d'itération permettant de trouver du parallélisme. Appliquez la méthode de Lamport pour construire progressivement une direction de front.
3. En déduire directement une transformation unimodulaire de l'espace d'itérations. Exprimer les nouveaux compteurs de boucles en fonctions des anciens `I,J`.

1. *"memory-based"*

4. Construction d'un nouveau nid de boucles contenant du parallélisme :

```
DO K1 = ? , ?
   DO K2 = ? , ?
      ???
   END DO
END DO
```

Quelles sont les bornes des nouvelles boucles ? Montrez que l'on peut appliquer la méthode de Fourier-Motzkin pour trouver l'expression de ces bornes.

Exercice 3 Considérons une architecture VLIW capable d'exécuter en parallèle deux accès à la mémoire et deux additions. Les opérations sont pipelinées, ainsi que les accès à la mémoire, avec des latences de 3 pour les accès mémoire et 2 pour les additions.

Exemple 3

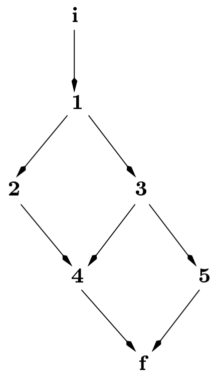
```
    DO I = 1, 100
S:     X(I) = Y(I) + 1
T:     Y(I+2) = X(I+1) + 1
    END DO
```

1. Vérifier que le graphe de dépendances est sans cycle.
2. En combien de tops d'horloge (ou "clock cycles") peut-on exécuter une itération de la boucle originale de l'exemple 3, en utilisant un ordonnancement local ?
3. Ecrire le code de la boucle déroulée *trois fois*. Donner la performance moyenne de la boucle ainsi déroulée.
4. Donner au moins une technique pour améliorer encore cette performance (sans écrire de code).

Exercice 4 On *définit* la relation de post-dominance entre les nœuds d'un graphe de flot de contrôle (avec entrée **i**, sortie **f**) comme suit :

Un nœud **a** post-domine un nœud **b** si tout chemin de **b** à **f** passe par **a**.

1. Quel rapport pouvez-vous établir avec une autre relation vue en cours ?
2. Étant donné le cfg G ci-dessous, indiquez quels sont les nœuds en relation de post-dominance.



3. On définit la relation de dépendance de type *contrôle* comme suit :
   un nœud **a** est contrôle-dépendant d'un nœud **b** si :
   - **a** post-domine un successeur de **b** ;
   - mais **a** ne post-domine pas **b**.

   On notera en abrégé **b** δ^c **a**.
   Dans le graphe ci-dessus, quels nœuds sont contrôle-dépendants du nœud **c** ?
4. Quel rapport pouvez-vous établir entre δ^c et la frontière de dominance ?