Architecture des ordinateurs haute performance

Soraya Zertal

Laboratoire Li-PaRAD, Université de Versailles
E-mail: soraya.zertal@uvsq.fr

Cours 5 : ORDRES ET DEPENDANCES

Notion D'ordre - Ordre statique (OST)

Définition :
L'ordre statique (OST) d'une instruction dans une procédure est un entier qui lui est attribué et qui identifie de manière unique sa position dans l'ensemble des instructions qui composent la procédure en question.

L'ordre statique est généré par le compilateur.

Il n'est valable que pour une procédure $\Rightarrow$ ne peut être considéré comme un ordre global.

Notion D'ordre - Ordre statique (OST)

Exemple :

```
1       Lw R1,N(R0)
2       Add R2,R0,#0
3       Add R2,R1,R2
4       Sub R1,R1,#1
5       BNEZ R1,-8
6       Sw S(R0),R2
```

Notion D'ordre - Ordre d'exécution de référence

Définition :

Une machine séquentielle de référence est caractériée par :

1. A chaque cycle, une **seule** instruction est en cours d'exécution,
2. Toute instruction est **complètement** exécutée en un seul cycle.

Notion D'ordre - Ordre d'exécution de référence

Définition :
L'exécution d'un code sur une machine séquentielle de référence permet de définir un ordre total en associant à chaque instruction le numéro du cycle où elle a été exécutée.

Remarques :

1. La présence de branchements rend l'ordre d'exécution en général très différent de l'ordre statique,
2. L'ordre d'exécution correspond à une trace d'exécution.

Notion D'ordre - Ordre d'exécution de référence

L'exemple précédent à l'exécution :

```
1    Lw R1,N(R0)          9   Add R2,R1,R2
2    Add R2,R0,#0        10   Sub R1,R1,#1
3    Add R2,R1,R2        11   BNEZ R1,-8
4    Sub R1,R1,#1        12   Add R2,R1,R2
5    BNEZ R1,-8          13   Sub R1,R1,#1
6    Add R2,R1,R2        14   BNEZ R1,-8
7    Sub R1,R1,#1        15   Sw S(R0),R2
8    BNEZ R1,-8
```

Supposant que N est une variable entière qui vaut 4.

Notion de dépendances de données

Définition :

Deux instructions présentent une dépendance de données si et seulement si elles utilisent (en lecture ou en écriture) une même donnée qu'il s'agisse d'un registre ou d'un mot mémoire.

Dépendances de données en registre

Soient I1 et I2 deux instructions, supposons que :

- I1 est avant I2 selon **l'ordre d'exécution**,
- I1 et I2 utilisent une même donnée X.

4 cas possibles :

1. I1 lit X; I2 lit X. Dép. RAR (Read After Read)
2. I1 écrit X; I2 lit X. Dép. RAW (Read After Write)
3. I1 lit X; I2 écrit X. Dép. WAR (Write After Read)
4. I1 écrit X; I2 écrit X. Dép. WAW (Write After Write)

Exemple de dépendance RAR

RAR : Consommateur/consommateur

1. I1 : $Add\ R7, R4, R1 \quad (R7 \leftarrow [R4] + [R1])$
2. I2 : $Lw\ R8, 4(R9) \quad (R8 \leftarrow mem[[R9] + 4])$
3. I3 : $Mult\ R5, R2, R4 \quad (R5 \leftarrow [R2] * [R4])$

Il y a une dépendance RAR entre I1 et I3 car :

- I1 est exécutée avant I3,
- I1 et I3 utilisent toutes les deux en lecture le même registre R4.

Exemple de dépendance RAW

RAW : Producteur/Consommateur

1. I1 : $Add\ R7, R1, R4 \quad (R7 \leftarrow [R1] + [R4])$
2. I2 : $Lw \quad R8, 4(R9) \quad (R8 \leftarrow mem[[R9] + 4])$
3. I3 : $Sub\ R5, R2, R7 \quad (R5 \leftarrow [R2] - [R7])$

Il y a une dépendance RAW entre I1 et I3 car :

- I1 est exécutée avant I3,
- I1 et I3 utilisent toutes les deux le registre R7 : I1 en écriture et I3 en lecture.

Contrainte d'ordre : R7 doit être écrit par I1 avant d'être lu par I3

Exemple de dépendance WAR

WAR : Consommateur/Producteur

1. I1 : $Add\ R7, R1, R4 \quad (R7 \leftarrow [R1] + [R4])$
2. I2 : $Sub\ R10, R9, R8 \quad (R10 \leftarrow [R9] - [R8])$
3. I3 : $Lw \quad R4, 6(R2) \quad (R4 \leftarrow mem[[R2] + 6])$

Il y a une dépendance WAR entre I1 et I3 car :

- I1 est exécutée avant I3
- I1 et I3 utilisent toutes les deux le registre R4 : I1 en lecture et I3 en écriture

Contrainte d'ordre : R4 doit être lu par I1 avant d'être écrit par I3

Exemple de dépendance WAW (1)

WAW : Producteur/Producteur

1. I1 : $Mult\ R7, R1, R4 \quad (R7 \leftarrow [R1] * [R4])$
2. I2 : $Sub\ R10, R9, R8 \quad (R10 \leftarrow [R9] - [R8])$
3. I3 : $Add\ R7, R2, R3 \quad (R7 \leftarrow [R2] + [R3])$

Il y a une dépendance WAW entre I1 et I3 car :

- I1 est exécutée avant I3,
- I1 et I3 utilisent toutes les deux le registre R7 en écriture.

Contrainte d'ordre : R7 doit être écrit par I1 avant d'être écrit par I3

Exemple de dépendance WAW (2)

Le code précédent n'a pas de sens (code idiot) et ne peut être généré par un compilateur correct.
La raison est qu'aucune instruction n'utilise R7 entre les deux instructions I1 et I3 et par conséquent on n'utilise pas la première valeur produite par I1 dans R7.

Un code plus réaliste :

1. I1 : $Sub\ R7, R1, R4 \quad (R7 \leftarrow [R1] - [R4])$
2. I2 : $BNEZ\ R6, -8 \quad (si [R6] ! = 0, aller\ avant\ sub)$
3. I3 : $Add\ R7, R2, R3 \quad (R7 \leftarrow [R2] + [R3])$

Dépendances de données en mémoire

L'exemple précédent reste simple mais supposons que l'on change I3 :

1. I1 : $SW \quad 4(R1), R7 \quad (mem[[R1] + 4] \leftarrow [R7])$
2. I2 : $ADD \ R5, R2, R4 \quad (R5 \leftarrow [R2] + [R4])$
3. I3 : $LW \quad R6, 16(R8) \quad (R6 \leftarrow mem[[R8] + 16])$

La dépendance existe si et seulement si ([R1]+4) = ([R8]+16).

Cela nécessite la connaissance des contenus de R8 et R1.

Ordre induit par les dépendances de données

Les dépendances RAW, WAR et WAW induisent une contrainte sur l'ordre d'exécution des instructions concernées : cet ordre doit être obligatoirement respecté pour préserver la sémantique du programme (résultats corrects)

Précision : La contrainte porte plutôt sur l'ordre des opérations d'accès (lecture et écriture).

Ordre induit par les dépendances de données

RAR n'induit aucune contrainte : aucun ordre à respecter dans l'exécution des instructions concernées.

Les dépendances RAR indiquent simplement une réutilisation d'une variable.

Sémantique des dépendances de données

Les dépendances WAR indiquent qu'un même nom de variable est réutilisé (en lecture puis en écriture) comme dans l'exemple ci dessous :

1. $Sub\ R1, R4, R7 \quad (R1 \leftarrow R4 - R7)$
2. $Mult\ R8, R9, R10 \quad (R8 \leftarrow R9 * R10)$
3. $Lw \quad R4, 8(R2) \quad (R4 \leftarrow mem[[R2] + 8])$

On peut éviter cette dépendance en utilisant le **renommage de registres**.

Sémantique des dépendances de données

Dans l'exemple précédent, si dans I3, R4 est remplacé par R5 (supposé non utlisé) donnant $LW \quad R2, R5, 8$:

1. La dépendance WAR disparait
2. La sémantique est respectée à condition de propager dans la suite des instructions le remplacement de R4 par R5.

Sémantique des dépendances de données

Dans une dépendance WAR, il n y a pas de communication d'information utile entre les deux instructions impliquées dans la dépendance.

Si la donnée est un registre, la dépendance WAR est dûe à une mauvaise gestion des registres par le compilateur.

Si la donnée est une référence mémoire, la dépendance WAR incombe à l'utilisateur (programmeur).

Sémantique des dépendances de données

Les dépendances WAW indiquent qu'un même nom de variable est réutilisé deux fois en écriture. De même que dans le cas WAR, une substitution de registre fait disparaitre le problème.

Idem que pour une dépendance WAR :
il n y a pas communication d'information utile entre les deux instructions impliquées dans la dépendance.

Si la donnée est un registre, la dépendance WAW est dûe à une mauvaise gestion des registres par le compilateur.
Si la donnée est une référence mémoire, la dépendance WAW incombe à l'utilisateur.

Sémantique des d'épendances de données

Les dépendances RAW, au contraire, correspondent à une transmission d'information (valeur et non plus nom) entre deux instructions.

Remarques :

1. Bien distinguer entre valeur et nom,
2. Seules les dépendances **RAW sont incontournables**.

Calcul des dépendances de données (matériel)

Le calcul des dépendances de données suppose la connaissance de l'ordre d'exécution et donc en général ne peut être calculé qu'à l'exécution par le matériel.

Deux problèmes sont à résoudre :

1. Détermination de l'ordre d'exécution (faisable mais attention aux branchements)
2. Détermination d'une variable commune :
    1. Registre : facile , il suffit de comparer les numéros
    2. Mémoire : plus compliqué : il faut calculer les adresses

Si une dépendance ne peut être calculée, l'approche conservatrice (il y a dépendance) doit être prise.

Calcul des dépendances de données (logiciel)

Il est intéressant de précalculer les dépendances de données à la compilation. Deux niveaux peuvent se faire :

1. Niveau source : Les adresses mémoires peuvent être évaluées par contre les registres ne sont pas visibles
2. Niveau assembleur : Les registres sont accéssibles par contre les adresses mémoires sont complexes à évaluer

Dans les deux cas, l'ordre d'exécution est très difficile à calculer (pb lié aux branchements)
Idem pour le matériel : Si une dépendance ne peut être calculée, l'approche conservatrice (il y a dépendance) doit être prise.

Exemple

Soit le code suivant, établir la liste des dépendances de données pour UNE itération

```
1 Add R1, R0,#0
2 Add R3, R0,#1
3 LW R4,A(R1)
4 Add R5,R0,#1
5 Mult R5,R4, R5
6 Sub R4,R4,R3
7 BNEZ R4, -8
8 SW  B(R1),R5
9 Add  R1,R1,#4
10 J -28
```

Exemple

UNE itération signifie : cdt non satifaite pour BNEZ et Jump non exécutée.

Registre	RAR	RAW	WAR	WAW
R0	(2,1) (4,2)			
R1	(8,3) (9,8)	(3,1)		
R4	(6,5)	(5,3) (7,6)		
R5		(5,4) (8,5)		

Les transparents des cours suivants ne seront pas fournis.

Une prise de note lors du cours en Amphi est donc fortement conseillée.