Examen de compilation

M1 Informatique Fondamentale, ENS Lyon
Mercredi 5 janvier 2011

Durée: 3 heures. Tous les documents sont autorisés.
Ce partiel est constitué de 5 exercices. Les exercices 1, 2, 3 et 4 sont assez simples et pourront être traités en priorité. L'exercice 5 demande davantage de réflexion et pourra être traité sur le temps restant.

Exercice 1. *Procédures imbriquées*

On considère un programme de type Pascal avec des procédures imbriquées de la façon suivante:

```
proc P()
{
  proc Q()
  {
    proc T()
  }
  proc R()
  {
    proc S()
  }
}
```

On suppose que la séquence suivante est exécutée: P appelle R, R appelle S, S appelle P, P appelle R, R appelle Q, Q appelle T.

Q1. Dessiner la pile de contrôle en faisant figurer les liens statiques.

Exercice 2. *Allocation de registres*

On considère le fragment de code suivant:

```
L0:
  x <- y + x
  w <- 2 * x
  if s = u goto L1
  x <- w + u
  u <- u - 1
  goto L2
L1:
  s <- w + 1
L2:
  y <- s + x
  if y > 1000 goto L0
L3:
```

Q2. Dessiner le graphe de flot de contrôle

Q3. Rappeler les équations de flot de données pour le calcul des variables vivantes avant et après chaque affectation. Détailler les étapes du calcul du point fixe à l'aide d'un tableau. Annoter le graphe de flot de contrôle avec le résultat.

Q4. Combien vaut MAX_LIVE ? Justifier.

Q5. Dessiner le graphe d'interférence. Donner une allocation de registres pour $K = 4$ registres.

Q6. Donner une allocation de registres pour $K = 3$ registres et le code correspondant.

Exercice 3. *SSA*

On considère le graphe de flot de contrôle de l'exercice précédent.

Q7. Donner les ensembles de dominance. Tracer l'arbre de dominance.

Q8. Donner les frontières de dominance.

Q9. Passer le programme en SSA. On détaillera les étapes.

Q10. Donner sans justifier le graphe d'interférence du programme obtenu. Quelle propriété possède-t-il ? En quoi est-ce utile pour l'allocation de registres ?

Exercice 4. *Transformations unimodulaires*

On considère le programme suivant:

```
for(t=0; t <= 3; t++)
  for(i=0; i <= 3; i++)
    a(i) = a(i-1) + a(i) + a(i+1);
```

Q11. Donner les vecteurs de dépendences.

On considère maintenant la matrice:

$$U = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$$

et la transformation associée $(u, v) = U(i, t)$.

Q12. Montrer que U est une transformation unimodulaire valide.

Q13. Dessiner le domaine d'itération original et le domaine d'itération transformé en indiquant l'ordre d'exécution des opérations.

Q14. Donner en détaillant le code transformé. Que peut-on dire de la boucle interne ?

Q15. Montrer que U défini un tuilage valide. Donner le code transformé avec des tuiles de taille 2 (dans les deux directions). Dessiner le domaine d'itération tuilé.

Exercice 5. *Transformations non-unimodulaires*

On se propose de construire une méthode de génération de code adaptée aux transformations spécifiées par une matrice T à coefficients rationnels de déterminant non nul, et plus seulement une matrice unimodulaire.

On note $T = HU$ la forme de Hermite de T. On rappelle que H est une matrice triangulaire inférieure à coefficients positifs dont le plus grand élément d'une ligne est sur la diagonale et que U est une matrice unimodulaire. Si $\mathcal{I}$ est le domaine d'itération du programme original, on dit que $T\mathcal{I}$ est le domaine *transformé* et que $U\mathcal{I}$ est le domaine *auxiliaire*.

Q16. Soient $x, x' \in \mathcal{I}$ avec $Ux \ll Ux'$. Montrer que $Tx \ll Tx'$. Qu'est ce que çà signifie?
Q17. En déduire un algorithme simple de génération de code.

On note $\alpha_1, \ldots, \alpha_n$ les éléments diagonaux de H, δ_k et δ'_k deux vecteurs dont les k premières composantes valent 0, la composante k vaut 1, et les composantes suivantes sont quelconques.

Q18. Montrer que si x et $x + \delta_k$ sont deux points de $U\mathcal{I}$, alors $H(x + \delta_k) - Hx = \alpha_k \delta'_k$. Qu'est ce que çà signifie?

Q19. (difficile) Comment transformer le nid de boucle généré en Q17 pour itérer directement dans $T\mathcal{I}$?