Examen final de compilation

ÉNSIIE, semestre 3

mardi 5 janvier 2016

Durée : 1h45.
Tout document personnel autorisé (pas de prêt entre voisins).
Ce sujet comporte 5 exercices indépendants, qui peuvent être traités dans l'ordre voulu.
Il contient 4 pages.
Le barème est donné à titre indicatif, il est susceptible d'être modifié. Le total est sur 20 points.
Il va de soi que toute réponse devra être justifiée.

Exercice 1 : Syntaxe (2 points)

Donner l'arbre de syntaxe abstraite des instructions Pseudo Pascal suivantes, ou expliquer pourquoi elles ne sont pas syntaxiquement correctes :

1. `a := 2 + 3 * b`
2. `if x < y then x := y else y := x`
3. `while x < y and y < z do y := z - x`
4. `while x := y do y := x`

Exercice 2 : Analyse syntaxique (3 points)

On considère la grammaire suivante :

$$\begin{aligned} I &\to gLd \\ &\mid i \\ L &\to LvL \\ &\mid I \end{aligned}$$

1. Construire l'automate déterministe LR(0) pour cette grammaire.
2. Cette grammaire est-elle LR(0) ?
3. Proposer une grammaire LR(0) reconnaissant le même langage.

Exercice 3 : Réécriture (5 points)

On considère trois opérateurs `vide`, `miroir`, `noeud` attendant respectivement 0, 1 et 2 arguments, ainsi que le système de réécriture suivant :

$$\texttt{miroir}(\texttt{miroir}(X)) \to X$$
$$\texttt{noeud}(\texttt{miroir}(X), \texttt{miroir}(Y)) \to \texttt{miroir}(\texttt{noeud}(Y, X))$$

1. Donner la ou les forme(s) normale(s) de $\texttt{noeud}(\texttt{miroir}(\texttt{vide}), \texttt{miroir}(\texttt{miroir}(\texttt{vide})))$.
2. Le système de réécriture est-il terminant ?
3. Le système de réécriture est-il confluant ?
4. Calculer les paires critiques du système. Lesquelles sont-elles joignables ?
5. Orienter les paires critiques non joignables de façon à ce que le système soit terminant. Les nouvelles paires critiques sont-elles joignables ?
6. Compléter le système jusqu'à obtenir un système confluent.

Exercice 4 : Graphe de flot de contrôle et allocation de registres (7 points)

Soit le programme Pseudo-Pascal suivant :

```
if x <= z and y <= z
  then begin
    hyp := z;
    c1 := x;
    c2 := y
  end
else if x <= y and z <= y
  then begin
    hyp := y;
    c1 := x;
    c2 := z
  end
else begin
    hyp := x;
    c1 := y;
    c2 := z
  end;
pyt := hyp * hyp - c1 * c1 - c2 * c2;
if pyt = 0 then
  r := 1
else
  r := 0
```

1. Donner le graphe de flot de contrôle correspondant en syntaxe Pseudo Pascal. Ne pas oublier le comportement coupe-circuit du `and`.
2. Modifier le graphe pour que les nœuds contiennent des instructions MIPS sur des pseudo-registres. Le cas échéant, il faudra décomposer le calcul des expressions en suite d'instructions élémentaires. On pourra utiliser l'instruction `ble` (*branch if less or equal*) pour traduire les conditions. On supposera que les variables `x y z hyp c1 c2 pyt r` sont stockées respectivement dans les pseudo-registres `%0 %1 %2 %3 %4 %5 %6 %7`.
3. En déduire sa traduction en langage RTL.
4. Indiquer dans un tableau quels pseudo-registres sont vivants en chacun des points du programme. On n'oubliera pas les pseudo-registres introduits lors de la traduction en RTL.
5. Donner le graphe d'interférence du programme (avec les arêtes de préférence). On indiquera sur les arêtes le numéro d'une instruction qui a causé l'interférence ou la préférence.
6. Essayer de 2-colorier ce graphe en appliquant l'algorithme de George et Appel. On détaillera bien chaque étape. On utilisera le critère de George pour justifier les fusions éventuelles.

Exercice 5 : Convention d'appel (3 points)

On considère le programme Pseudo Pascal suivant :

```
1  program
2  var x : integer;
3
4  function carre(u : integer) : integer;
5    begin
6     carre := u * u
7    end;
8
9  function norme(y : integer; z : integer) : integer;
10   begin
11     norme := carre(z) + carre(y)
12   end;
13
14 begin
15   x := -2;
16   x := norme(3, x)
17 end.
```

On suppose qu'on utilise la convention d'appel MIPS comme vue en cours. On suppose d'autre part que la variable globale `x` est stockée dans le registre `$s0`.

1. Quels registres `carre` doit-elle sauvegarder ? En déduire la trame de `s`. (Précision : en tant que variable globale, `$s0` n'a pas besoin d'être sauvegardé.)
2. Même question pour `norme`.
3. Pour expliciter la convention d'appel, quelles instructions faut-il ajouter :
   a) Au début de `carre` ?
   b) À la fin de `carre` ?
   c) Au début de `norme` ?
   d) Avant le premier appel à `carre` dans `norme` ?
   e) Après le premier appel à `carre` dans `norme` ?
   f) Avant le second appel à `carre` dans `norme` ?
   g) Après le second appel à `carre` dans `norme` ?
   h) À la fin de `norme` ?