Examen final de compilation

Énsiie, semestre 3

10 janvier 2011

Durée : 1h45.
Tout document personnel autorisé (pas de prêt entre voisins).
Ce sujet comporte quatre exercices indépendants, qui peuvent être traités dans l'ordre voulu. Il contient 3 pages.
Le barème est donné à titre indicatif, il est susceptible d'être modifié. Le total est sur 20 points.

Exercice 1 : Syntaxe et sémantique (5 points)

1. Donner l'arbre de syntaxe abstraite des instructions Pseudo Pascal suivantes :
   a) `i := 1 + t[2 * x]`
   b) `t[3 * i] := 2 + x`
   c) `if c <= 2 then t := new array of integer [2] else f(4)`
2. Donner la sémantique de l'instruction `t[3*i] := 2 + x` lorsque `x` est une variable globale valant 4, `t` est un tableau global de 2 cases rempli de 3, et `i` est une variable locale valant 0. On donnera le détail de toutes les règles d'inférence utilisées.
3. On considère le programme Pseudo Pascal suivant :

```
1  var x : integer
2
3  function f(y : integer)
4    var z : integer
5    if y <= 0
6    then f := 2
7    else begin
8      x := f(y - 1);
9      z := x * y;
10     f := z - 3
11   end
12
```

```
13      begin
14        x := 1;
15        x := f(x)
16      end
```

Donner la sémantique de ce programme. On pourra se contenter de ne donner que les grandes étapes (c'est-à-dire celles avec un état différent).

Exercice 2 : Réécriture (3 points)

Les système de réécriture suivants terminent-ils, sont-ils confluents ? Justifiez vos réponses. (`a b c d e h` sont des constantes, `f g` des symboles de fonction et x une variable.)

1. $\texttt{a} \to \texttt{b} \quad \texttt{a} \to \texttt{c}$
2. $\texttt{a} \to \texttt{b} \quad \texttt{b} \to \texttt{a}$
3. $\texttt{a} \to \texttt{b} \quad \texttt{b} \to \texttt{a} \quad \texttt{a} \to \texttt{c} \quad \texttt{b} \to \texttt{d}$
4. $\texttt{f}(\texttt{g}(x)) \to \texttt{d} \quad \texttt{g}(\texttt{h}) \to \texttt{e} \quad \texttt{d} \to \texttt{f}(\texttt{e})$

Exercice 3 : Convention d'appel (6 points)

On considère le programme Pseudo Pascal suivant :

```
1  var x : integer
2
3  function f(y : integer)
4    var z : integer
5    if y <= 0
6    then f := 2
7    else begin
8      x := f(y - 1);
9      z := x * y;
10     f := z - 3
11   end
12
13 begin
14   x := 1;
15   x := f(x)
16 end
```

Convention par pile

Dans cette partie, on suppose qu'on utilise la convention d'appel par pile : les arguments, l'adresse de retour et la valeur de retour sont passés sur la pile, et les variables locales y sont sauvegardées.

1. Représenter la trame de la fonction `f`.
2. Quelle est l'évolution de la pile lors de l'exécution du programme ?

Convention d'appel MIPS

Dans cette partie, on suppose qu'on utilise la convention d'appel MIPS comme vue en cours. On suppose d'autre part que la variable locale `z` dans `f` est stockée dans le registre *caller-saved* `$t0` et que la variable globale `x` est stockée dans le registre `$s0`.

3. Quels registres `f` doit-elle sauvegarder ? En déduire la trame de `f`. (Précision : en tant que variable globale, `$s0` n'a pas besoin d'être sauvegardé.)
4. Décrire l'évolution de la pile et du contenu des registres `$a0 $v0 $ra $t0 $s0` au cours du programme.

Exercice 4 : Allocation de registres (6 points)

On considère le programme Pseudo Pascal suivant :

```
var a, b, c, d, e, f : integer
begin
  a := b;
  d := f + c;
  while d > 0 do
  begin
    e := d;
    if f <= 0 then c := e else c := f;
    b := f + d;
    f := d - 2
  end
end
```

1. Dessiner le graphe de flot de contrôle correspondant (en gardant une syntaxe Pseudo Pascal pour les instructions de base).
2. Donner les variables vivantes en chacun des points du programme.
3. Quelles instructions pourraient-elles être éliminées ?
4. Dessiner le graphe d'interférence du programme de départ (avec les arêtes de préférence).
5. Essayer de 2-colorier ce graphe en appliquant l'algorithme de George et Appel. On détaillera le déroulement de l'algorithme et on justifiera le critère utilisé lors d'une fusion.
6. En utilisant ce 2-coloriage, donner le programme RTL correspondant au programme initial, dans lequel on aura alloué les variables aux deux registres `$s0` et `$s1` et pour lequel on utilise `$t0` et `$t1` pour sauvegarder ou restaurer la valeur des variables éventuellement spillées.