Maîtrise d'informatique le 26 novembre 2003

Compilation

— examen de première session —

Exercice 1 Traduction, génération et optimisation de code

1. Traduire en code SPARC virtuel le programme suivant, sachant qu'un entier occupe 4 octets.

```
program Tris;

var T: array[0..NMAX-1] of integer;
    n: integer;

procedure echange(var x,y: integer);
    {échange les valeurs des deux variables}

function TriBulles(n : integer): integer;
var i,j;
begin
    for i := 1 to n-1 do
        for j := n downto i+1 do
            if T[j-1] > T[j] then echange(T[j-1],T[j])
    TriBulles := T[n/2]
        {cette instruction équivaut à "return T[n/2]"}
end

begin
    {code correspondant à la lecture de n et de T}
    println(TriBulles(n))
end
```

2. Donner le code à trois adresses correspondant aux boucles de la fonction TriBulles.
3. Construire ensuite le graphe de flot de contrôle pour ce code.
4. Optimiser le code obtenu, en précisant soigneusement les techniques d'optimisation utilisées.

Exercice 2 Graphe de flot de contrôle et grammaires attribuées

Considérons la grammaire suivante, représentant un langage de programmation très simple.

```
instr : affectation
      | instr instr
      | WHILE exprBool DO instr ENDWHILE
```

L'objectif est d'attribuer cette grammaire afin d'obtenir le graphe du flot de contrôle d'un programme écrit dans ce langage. Faute de temps, on ne générera pas le code à trois adresses, mais seulement les blocs et les arcs les reliant. On suppose disposer de fonctions de manipulation de graphes comme `CreerBloc():bloc`, `AjouterArc(b1, b1: bloc)`, `FusionnerBlocs(b1, b2: bloc)` ou d'autres que vous aurez spécifiées.

1. Construire le graphe de flot de contrôle pour un programme de la forme

   ```
   WHILE exprBool
   DO
       affectation
       affectation
   ENDWHILE
   affectation
   ```

2. Construire la grammaire attribuée obtenant le graphe de flot de contrôle. Pour ce faire, remarquer que le gfc correspondant à toute `instr` possède un bloc d'entrée, par lequel on rentre dans le code de l'instruction, et un bloc de sortie, par lequel on quitte le code de l'instruction.

3. Même question que précédemment, si l'on rajoute à la grammaire la règle de production

   ```
   instr : IF exprBool THEN instr ELSE instr ENDIF
   ```

   Observer que maintenant le gfc d'une instruction peut comporter plusieurs blocs de sortie!