

Langages et Traducteurs

TDs 10 et 11 : Génération de Code

Exercice 1.

En utilisant les fonctions de génération de code vues en cours, écrivez la séquence de code produite pour les fragments de programme suivants. On suppose que les déplacements associés à x_1 , x_2 et x_3 sont 4, 8 et 12.

<pre>x1 := 3; x2 := 3 + x1 + x2; si non (x2 = x1) alors x1 := x2 sinon x2 := 0</pre>	<pre>x2 := 0; tantque ((x3 < 4) and (x2 = x1)) x3 := x3 + 1;</pre>
--	---

Exercice 2.

On étend le langage `while` en lui ajoutant :

- l'opérateur booléen "ou exclusif" ($e_1 \text{ xor } e_2$ est vrai ssi un seul des deux opérandes est vrai) ;
- l'instruction `repete c jusqu'a e`.

Complétez les fonctions de génération de code vues en cours pour prendre en compte ces extensions.

Exercice 3.

```
procedure P(void);
  var x ;

  procedure P1 (a)
    var x1 ;

    procedure P2(b, c);
      var x2 ;
      begin { P2 }
        x2 := c ;
        x := x1 + x2 + b ;    /* instruction (2) */
      end; { P2 }

    begin { P1 }
      x1 := a ;
      P2 (x+1, a) ;    /* instruction (1) */
      x := 2 ;
    end; { P1 }

begin { P }
  x := 0 ;
  P1 (5);
end { P }
```

1. Dessinez le contenu de la pile lors de l'exécution de la procédure P2.
2. Donnez le code obtenu pour les instructions (1) et (2).

Exercice 4.

```
procedure P(void);
  var z : integer;
  procedure P1(function q(c:integer):integer);
  begin
    z:=q(4);
  end; { P1 }
  function R (x : integer):integer;
  begin
    R:=x+2; // Le résultat de R est la valeur x+2
  end; { R }
  procedure P2(void);
  begin
    P1(R);
  end; { P2 }
begin
  P2();
end;
```

1. Dessinez l'arbre des appels correspondant à l'exécution de cette procédure
2. Dessinez le contenu de la pile lors de l'exécution de R
3. Ecrivez le code associé aux procédures P, R, P1 et P2.