Le code à générer pour une instruction d'écriture telle write (e1, e2, e3) est le suivant

<code1>	empile la valeur de l'expression e1
PRN	imprime cette valeur
<code2>	empile la valeur de l'expression e2
PRN	imprime cette valeur
<code3>	empile la valeur de l'expression e3
PRN	imprime cette valeur

Les P-Codes <code1>, <code2> et <code3> sont générés lors de l'analyse des expressions e1, e2 et e3 par des appels à EXPR.
On modifiera la procédure ECRIRE en conséquence.

Le code à générer pour une instruction de lecture telle read (v1, v2, v3) est le suivant :

LDA <adresse de v1>	empile l'adresse de la variable v1
INN	lit un entier, le stocke dans v1
LDA <adresse de v2>	empile l'adresse de la variable v2
INN	lit un entier, le stocke dans v2
LDA <adresse de v3>	empile l'adresse de la variable v3
INN	lit un entier, le stocke dans v3

Modifier la procédure LIRE en conséquence.

Nous étudions ici la génération de code pour les instructions de rupture de séquence (if COND then INST et while COND do INST). Cette génération est faite sur les modèles suivants :

Pour l'instruction if

	code généré pour COND
	if not COND then goto LABEL
	code généré pour INST
LABEL	suite...

Pour l'instruction while.

DEBUT	code généré pour COND
	if not COND then goto LABEL
	code généré pour INST
	goto DEBUT
LABEL	suite...

Le problème est que lors de la génération du saut conditionnel à LABEL, on ne connaît pas encore la valeur de ce label puisqu'on ne connaît pas a priori la longueur du code généré pour INST.

On va donc générer des instructions de saut incomplètes (sans numéro d'instruction) et mémoriser les numéros des instructions incomplètes pour pouvoir les compléter lorsque l'information sera disponible

Cette facilité de compléter une instruction préalablement générée est offerte par l'intermédiaire des deux procédures NEXT_INST et REMPLIR_INST :

procedure NEXT_INST (var COMPT:integer) ;
la procédure NEXT_INST retourne dans le compteur indiquant le numéro de la prochaine instruction qui sera générée ;

procedure REMPLIR_INST (NUM_INST, DEP:integer) ;
la procédure REMPLIR_INST complète l'instruction de saut NUM_INST avec le numéro d'instruction DEP.

On modifie donc la procédure SI en conséquence :

```
procedure SI ;
var SAUT, SUITE : integer ;
begin
   TESTE (IF_TOKEN) ;
   COND ;
   TESTE (THEN_TOKEN) ;
   NEXT_INST (SAUT) ;
   GENERER2 (BZE, 0) ; (* 0 car incomplet *)
   INST ;
   NEXT_INST (SUITE) ;
   REMPLIR_INST (SAUT, SUITE)
end ;
```

On fait de même pour la procédure TANTQUE :

```
procedure TANTQUE ;
var DEBUT, SAUT, SUITE ; integer ;
begin
   TESTE (WHILE_TOKEN) ;
   NEXT_INST (DEBUT) ;
   COND ;
   TESTE (DO_TOKEN) ;
   NEXT_INST (SAUT) ;
   GENERER2 (BZE, 0) ; (* 0 car incomplet *)
   INST ;
   GENERER2 (BRN, DEBUT) ;
   NEXT_INST (SUITE) ;
   REMPLIR_INST (SAUT, SUITE)
end ;
```

On reprend les déclarations

```
type  MNEMONIQUES – (ADD,SUB,MUL,DIV,EQL,NEQ,GTR,LSS,GEQ,LEQ,
        PRN,INN,INT,LDI,LDA,LDV,STO,BRN,BZE,HLT) ;
      INSTRUCTION = record
        MNE : MNEMONIQUES ;
        SUITE : integer
      end
var   PCODE : array [0 .. TAILLECODE] of INSTRUCTION ;
      PC : integer ;
```

Les fonctions de génération de code GENERER1 et GENERER2 s'écrivent simplement :

```
procedure GENERER1 (M:MNEMONIQUES) ;
begin
   if PC = TAILLECODE then ERREUR ;
   PC := PC + 1 ;
   with PCODE [PC] do
      MNE := M
end ;
```

Les fonctions de génération de code GENERER2 :

```
procedure GENERER2 (M:MNEMONIQUES ; A:integer) ;
begin
   if PC = TAILLECODE then ERREUR ;
   PC := PC + 1 ;
   with PCODE [PC] do begin
      MNE := M ;
      SUITE := A
   end
end ;
```

Les procédures NEXT_INST et REMPLIR_INST s'écrivent :

```
procedure NEXT_INST (var COMPT:integer ) ;
begin
   COMPT := PC + 1
end ;
procedure REMPLIR_INST (NUM_INST, DEP : integer) ;
begin
   PCODE [NUM_INST]. SUITE := DEP
end ;
```

Références

Aho, A. V. and Ullman, J. D. (1977) The Principles of Compiler Design, Addison Wesley, Reading, Mass.

Bornat, R. (1979), Understanding and Writing Compilers, Macmillan.

Gries, D. (1971), Compiler Construction for Digital Computers, Wiley, N.Y.