Chap. 5 : Génération de code

Introduction

- Nous allons compléter l'analyseur sémantique construit dans le chapitre 3 pour générer le **P-Code** correspondant au programme **mini-pascal** analysé.
- Ainsi nous allons procéder comme suit :
  - Effectuer l'allocation mémoire.
  - Chaque reconnaissance d'une règle de la grammaire va déclencher la génération de P-Code à l'aide des procédures :
    - **generer1** (Génération d'une instruction P-Code sans opérande)
    - **generer2** (Génération dune instruction P-Code avec un opérande)

Allocation mémoire

- Au niveau du langage mini-pascal, on ne se préoccupe pas de la gestion de la mémoire. On la manipule par l'intermédiaire des symboles.
- C'est le rôle du compilateur d'allouer ces symboles en mémoire. A chaque symbole, le compilateur doit associer un emplacement en mémoire dont la taille dépend du type du symbole.
- Pour cela on choisit les adresses au fur et à mesure de l'analyse des déclarations en incrémentant un **offset** qui indique la place occupée par les déclarations précédentes.
- A la fin des déclarations, il est possible de déterminer l'emplacement mémoire à réserver dans la pile (Instruction P-Code INT).

Redéfinition du type SYMBOLES

Pour chaque symbole, son adresse mémoire est stockée dans la table des symboles :

```
STRUCTURE SYMBOLES {
        TOKEN : TOKENS   (*le token du symbole *)
        NOM     : CHAINE[8] (*la forme textuelle du Ssymbole*)
        CLASSE : CLASSE_IDF (*Contient la classe de l'idf*)
        ADRESSE : ENTIER      (*Adresse mémoire du symbole*)
}
OFFSET : ENTIER
```

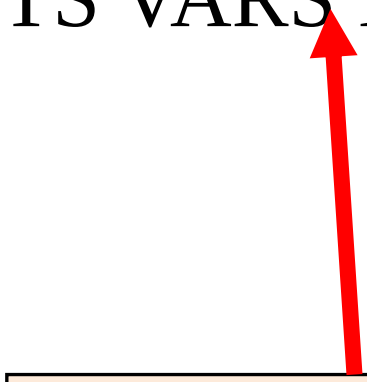
Redéfinition de entrer_symb

On modifie la procédure ENTRER_SYM pour tenir compte de cette allocation mémoire :

```
procedure ENTRER_SYMB (c : CLASSE_IDF)
debut
 dernier_symb ← dernier_symb +1
 TABLE_SYMB[dernier_symb].nom ← SYMB
 TABLE_SYMB[dernier_symb].classe ← c
 TABLE_SYMB[dernier_symb].token ← TOKEN
 si c=variable ou c=constante alors
    OFFSET ← OFFSET + 1
    TABLE_SYMB[dernier_symb].adresse ← OFFSET
 fin si
fin
```

Génération de début programme

BLOCK ::= CONSTS VARS INSTS



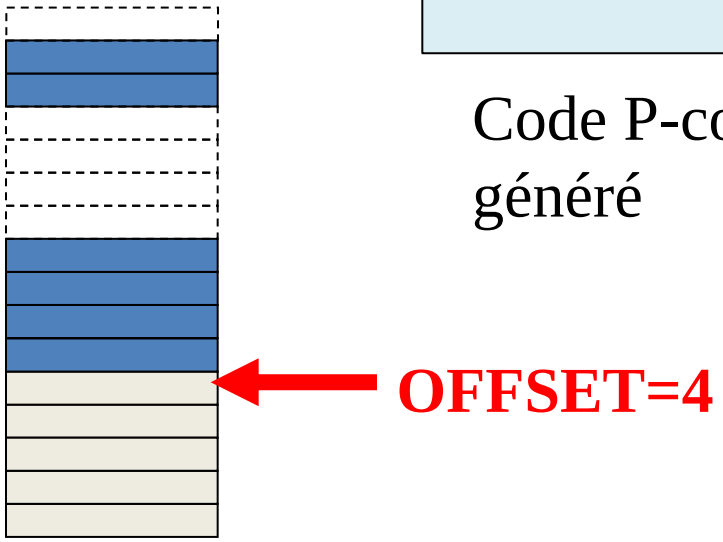
RESERVATION DE LA ZONE MEMOIRE

Génération de début programme

```
procedure BLOCK ;
debut
        OFFSET ← -1 ;
        CONSTS ;
        VARS ;
        PCODE[0].MNE=INT;
        PCODE[0].suite=OFFSET;
        INSTS
fin ;
```

INT OFFSET

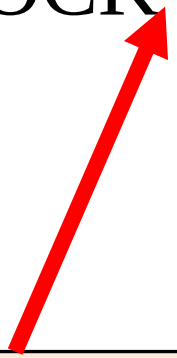
Code P-code généré



Si 2 constantes et 3 variables, à la fin, OFFSET=4 et la taille de la mémoire Réservée est 5 places.

Génération de fin de programme

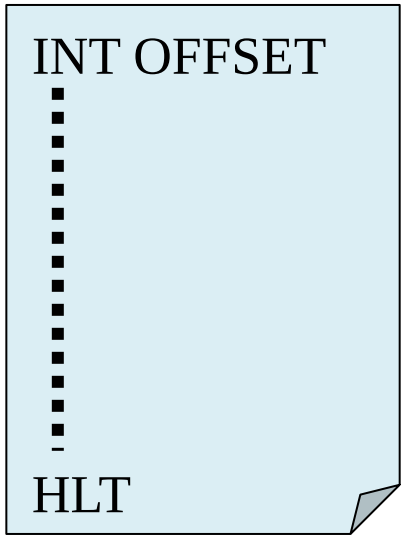
PROGRAM ::= program ID; BLOCK .



GENERATION DE L'ARRET DU PROGRAMME

Génération de fin de programme

```
procedure PROGRAM ;
debut
        TESTE(PROGRAM_TOKEN, PROGRAM_ERR) ;
        TESTE_ET_ENTRE(ID_TOKEN, ID_ERR) ;
        TESTE(PVIR_TOKEN, PVIR_ERR) ;
        BLOCK ;
        GENERER1 (HLT) ;
        TESTE(PNT_TOKEN, PNT_ERR) ;
fin ;
```



Code P-code généré

Travail à faire :

- Les déclarations nécessaires
- Définir les actions sémantiques dans les règles :
  - PROGRAM
  - CONSTS
  - VARS
  - BLOCK
  - AFFEC
  - FACT
  - TERM
  - EXPR
  - ECRIRE
  - LIRE