Compiladores
Laboratorio 19

Objetivo
Desarrollar e integrar la generación de código intermedio y ensamblador para un lenguaje imperativo con soporte para funciones, tipos de retorno int y void, y estructuras de control como if, while y break.

Programa

Se tiene implementado el scanner , parser para la gramática:

- Program ::= VarDecList FunDecList
- VarDecList ::= (VarDec)*
- FunDecList ::= (FunDec)+
- FunDec ::= **fun** Typ id **(**[ParamDecList] **)** Body **endfun**
- Typ::= **int** | **void**
- Body ::= VarDecList StmtList
- ParamDecList ::= Type id (**,** Type id)*
- VarDec ::= **var** Type VarList **;**
- Type ::= id
- VarList :: id (,id)*
- StmtList ::= Stmt **(;** Stmt **)***
- Stmt ::= **id =** CExp |
  - **print (** CExp **)**
  - **if** CExp **then** Body [**else** Body] **endif**
  - **while** CExp **do** Body **endwhile**
  - **break**
  - **id (** [ArgList] **)**
  - **return (** [CExp] **)**
- CExp ::= Exp [(**<**) Exp]
- Exp ::= Term ((**+** | **-**) Term)*
- Term ::= Factor ((*****|**/**) Factor)*
- Factor ::= id | Num **|** Bool | **(** Exp **) | id (** [ArgList] **)**
- ArgList ::= CExp (, CExp)*
- Bool **::= true | false**

Problema

1. **Implementar la clase FCallStm**
   Desarrollar la representación y generación de código para llamadas a funciones usadas como sentencia. Esta clase debe evaluar los argumentos, realizar la llamada y descartar el valor de retorno (si lo hubiera).

2. **Soporte para funciones void**
   Habilitar funciones sin valor de retorno (void). Se debe permitir el uso de return(); dentro de ellas, pero no return(exp).

3. **Implementar la sentencia break**
   Añadir soporte para la instrucción break dentro de bucles while. Debe generar un salto incondicional al final del bucle. Es necesario validar que break solo se use dentro de contextos de bucle válidos.