Sintaxis y Semántica de los Lenguajes

Alumnos: Mathias Castets, Yazmin Ferrufino **Docente: Marcelo Lipkin** **K2101**

Informe de TP Final: Compilador Micro Extendido

Alumno: Mathias Castets

El compilador desarrollado procesa codigo fuente “Micro” y procesa un condigo intermedio. Su arquitectura se basa en un **analizador sintáctico descendente recursivo** (ASDR) que valida gramática, y al mismo tiempo, invoca a rutinas semánticas que se encargar de comprobar los tipos de datos y generar el codigo.

Este proceso inicia con el **scanner**: se trata de un autómata finito que esta implementado con tabla. Este fue modificado para reconocer nuevos tokens, como por ejemplo REAL y CARÁCTER. A medida que se descomponen las declaraciones (ListaDeclaraciones), se carga en la Tabla de Símbolos (TS) los nombres de las variables que el usuario declara en el codigo, sino tambien con su tipo (TIPOENTERO, TIPOREAL o TIPOCARACTER).

La parte central del análisis semántico son las funciones GenInfijo y Asignar. La primer funcion fue modificada para implementar un sistema de “**promoción” de tipos**: cuando se detecta una operación aritmética en la que interviene al menos un dato de tipo real, el resultado se promueve a tipo real y se genera el codigo de operación “SumarReal”/”RestarReal”. Si nos encontramos ante numeros de tipo entero, se genera “Sumar”/”Restar”. Esto será vital a la hora de comunicar con la Maquina Virtual, ya que no es lo mismo sumar 2 enteros, que 2 numeros reales, los flags empleados varían, asi como la forma en la que se interpreta el resultado.

Al mismo tiempo, Asignar impone una comprobación de tipos, solo podemos asignar datos que correspondan al tipo de variable declarada (entero := entero, carácter := carácter, real := real), pero a la vez contemplando la “excepción” de real := entero, para poder hacer, por ejemplo, real :=10. Además, en el caso de operaciones lógicas, los resultados booleanos al ser 0 y 1, se establece como tipo entero.

Para el caso de las sentencias de control (si, mientras, repetir), implementamos un sistema de generación de etiquetas y saltos. Desarrollamos la funcion Condicion que evalúa una comparación cuyo resultado booleano (0 o 1) es guardado en una variable temporal (temp). Luego, ChequearCondicion usa ese temporal para generar SaltoSiFalso o SaltoSiVerdadero. De esta forma, la lógica de cada estructura se construye combinando este tipo de saltos. La estructura **mientras** comprueba la condición y salta al final cuando es falsa; los **si** saltan al bloque sino si la condición es falsa y el bloque **entonces** usa un salto incondicional para omitir el sino; y un **repetir** salta de vuelta al inicio si la condición es falsa.

Finalmente, el programa llega a su fin con la instrucción **detiene**.