Proyecto 4

Generador de Código

Lenguaje C-

Descripción

Hacer un programa en Python, llamado **`cgen.py`** que contenga una función:

- **`codeGen(tree, file)`**
  - La cual recibe **`tree`** que es el Árbol Sintáctico Abstracto (AST, por sus siglas en inglés) creado por el **`Parser`** (proyecto 2) y una variable **`file`** que es un string que contiene el nombre del archivo donde se dejará el código (incluyendo la extensión).
  - Desde luego que puede usar la tabla de símbolos que se generó.
  - Debe generar código MIPS para correrlos en SPIM.

Posteriormente, continuará con la definición de todas las funciones auxiliares que requiera.

Prueba del programa

Todos los archivos necesarios para que corra (todos los que se entregan en Bb, incluyendo **lexer.py**, **Parser.py**, **semantica.py** y el que contiene el archivo de prueba se colocarán en la misma carpeta para evitar el uso de paths adicionales.

El generador de código será probado con un script que iniciará importando tanto el archivo con los tipos globales como el parser, analizador semántico y generador de codigo. Posteriormente seguirá invocando el **`parse()`** y asignando lo que regresa a una variable **AST**, luego invocará la función **`tabla()`** para generar la tabla de símbolos y **`semantica()`** para revisar la semántica, pasándoles el AST y la variable de impresión.

La función **`parser(imprime)`**, como en el proyecto 2, deberá manejar las siguientes variables globales:

`posicion`: contiene la posición del siguiente carácter del string que se debe analizar. Deberá poder modificarla en su funcionamiento.

`progLong`: contiene la logitud del programa. Sólo la leerá cuando la requiera.

`programa`: contiene el string del programa completo. Sólo la leerá cuando lo requiera.

Por la forma en la que funciona Python al definir las variables y al hacer los **`import`**, la única forma que tenemos de pasarles estas variables globales será por medio de una función que las reciba y le pase el valor recibido a las variables globales que utiliza su programa.

La función para el paso de valores globales la tienen que agregar a su programa y tendrá la siguiente forma:

```
def globales(prog, pos, long):
    global programa
    global posicion
    global progLong
    programa = prog
    posicion = pos
    progLong = long
```

El script con el que se prueba será el siguiente:

```
from globalTypes import *
from parser import *
from semantica import *
from cgen import *

f = open('sample.c-', 'r')
programa = f.read()          # lee todo el archivo a compilar
progLong = len(programa)     # longitud original del programa
programa = programa + '$'    # agregar un caracter $ que represente EOF
posicion = 0                 # posición del caracter actual del string

# función para pasar los valores iniciales de las variables globales
globales(programa, posicion, progLong)

AST = parser(true)
semantica(AST, true)
codeGen(AST, “file.s”)
```

Para la entrega

- Un documento con:
    - Portada con nombre y matrícula.
    - Sección de introducción donde se explique qué tipo de código se genera MIPS, TM u otro, explicando la razón por la que fue seleccionado.
    - Manual del usuario, con un ejemplo paso por paso, con impresiones de pantalla, sobre cómo tengo que compilar y correr un programa escrito en C-.
    - Colocar como apéndices los documentos generados en las tres entregas anteriores y la definición del lenguaje.
- Todos los archivo Python (comentados) y TXT necesarios para que corra.
    - Si usa un emulador diferente a MIPS o TM, subirlo también.

NOTA: Si este proyecto se fuera a entregar a un tercero, es indispensable entregar la definición del lenguaje.