Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ciencias de la Computación e Informática

Definición y Diseño del Proyecto
"Mini C"

CI-1322
Autómatas y Compiladores
Profesor Luis Quesada
Grupo 01

951516 Marcos Jiménez Mata

A01651 José Frutos Porras

15 de octubre, 2014

Descripción del Tema

Para la realización de este proyecto se tratará de aplicar un enfoque práctico de lo aprendido en el transcurso de las clases utilizándolo en la creación de un compilador. Se tratará de simular los diferentes pasos que realizan los compiladores que normalmente se utilizan para el desarrollo de software. Para ello se seguirán los procesos que utiliza el analizador léxico (Comprobación y generación de Tokens) y los que utiliza el analizador sintáctico (Comprobación de gramáticas y Generación de un Árbol Sintáctico).

Idealmente se querría hacer un compilador de C Ansi 89 completo, para poder aprovechar las etapas de compilación de gcc (preprocesamiento, compilación, ensamblado y enlace) y reemplazar así la etapa de compilación.

Pero debido a falta de tiempo (y conocimiento), este compilador aceptará un mini-lenguaje con sintaxis similar a la de C. Esto significa que se presentaría un error si se intenta incluir una biblioteca estándar. En cambio vamos a permitir una función print y comentarios tipo C++.

Se utilizará como lenguaje de programación Python 3 y además se usará el analizador léxico y sintáctico Ply[2].

Lista Provisional de "*Tokens*" a Utilizar

- Palabras Clave (Reservadas)

Nombre	Atributo	Expresión Regular
IF		if
ELSE		else
WHILE		while
PRINT		print

- Operadores

Nombre	Atributo	Expresión Regular
(		\(
)		\)
*		*
/		/
%		%
+		\+
-		\-
<		<
>		>
==		==
=		=

- Tipos de Datos

Nombre	Atributo	Expresión Regular
NUM	Entero sin signo	$[0\text{-}9]^+$
ID	Nombre de variable o puntero a la tabla de símbolos	$[a\text{-}zA\text{-}Z0\text{-}9\_]^+$

- Otros

Nombre	Atributo	Expresión Regular

{		\{
}		\}
;		;

Reconocido por el lexer pero descartado:

- Comentarios: empiezan por “//” y terminan al terminar la línea. “//.*$”
- Espacios en blanco: “[\t\n]+”

Otros caracteres deberían producir un error léxico.

Gramática Provisional [4]

lista_instr → instr; instr

instr → if (expr) instr

| if (expr) instr else instr

| while (cond) instr

| print(expr)

| { lista_instr }

cond → expr > expr | expr < expr | expr == expr | expr

expr → expr + term | expr - term | term

term → term * fact | term / fact | term % fact | fact

fact → (expr) | id | num

id → id = id | id = num

Producto Esperado

Debido a que aún no se conoce el tipo de salida que se puede obtener con el *parser*, se presentarán dos posibilidades de qué hacer con el producto de éste.

1) La primera opción sería usar el *frontend* creado como un intérprete del mini C lo que tiene una utilidad inmediata ya que este es el lenguaje más conocido para el grupo y muchas veces se quiere hacer pequeños programas para pruebas, cálculos y otros.
2) La otra opción sería que al final del análisis se tenga como producto un cierto tipo de secuencia de instrucciones de tres direcciones de manera que estas puedan ser transformadas luego en lenguaje ensamblador y luego ser ensambladas. No se conoce aún la dificultad que esto pueda traer, ni tampoco que tan complicado podría ser pasar a ensamblador código arbitrario que podría tener manejo de la pila con las variables locales y creación de etiquetas para los saltos.

Sea cual sea el la opción elegida, se espera crear una herramienta que permita reafirmar los conocimientos que se han ido adquiriendo en el curso. Utilizando especialmente el análisis léxico y el análisis sintáctico con una herramienta que no se había utilizado anteriormente como lo es Python y la implementación de Lex y Yacc para este lenguaje, Ply.

Referencias

[1]The Linux Documentation Project. *Creating our own front end*. Disponible en Internet: http://www.tldp.org/HOWTO/GCC-Frontend-HOWTO-7.html [Consulta: Octubre, 2014]

[2] Beazey, David. *PLY (Python Lex-Yacc)* Disponible en Internet: http://www.dabeaz.com/ply/ DABEAZ LLC [Consulta: Octubre, 2014]

[3] Conti, Juan José. (2007) *MiniLisp (un ejemplo de ply)* Disponible en Internet: http://www.juanjoconti.com.ar/2007/11/02/minilisp-un-ejemplo-de-ply/ [Consulta: Octubre, 2014]

[4] Aho, Alfred et al. (2008) *Compiladores Principios, tecnicas y herramientas*. 2a Edición. Pearson Educación.