Compiladores

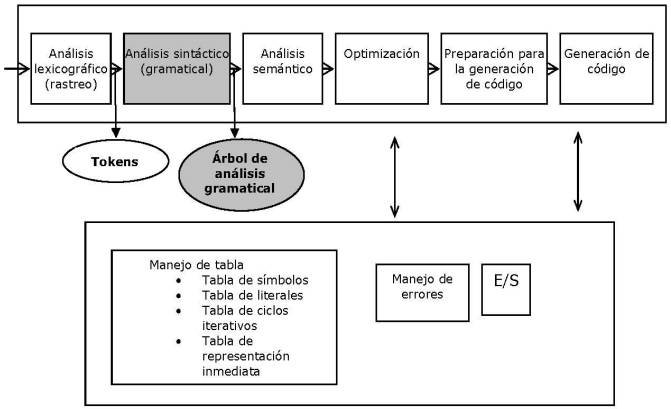
Lizbeth Escobedo
e: lizbeth.escobedo@cetys.mx
Escuela de Ingeniería, CINAP, Cetys

1. Introducción

Síntesis sesión anterior

Análisis Sintáctico Gramatical

Análisis sintáctico gramatical



Análisis sintáctico gramatical

- Se conoce también como análisis jerárquico debido a que **convierte una secuencia de tokens en un árbol de sintaxis** el cual es como sabemos una estructura jerárquica.
- **Agrupa los tokens del programa fuente en frases gramaticales** que el compilador usará en las siguientes etapas.

Análisis sintáctico gramatical

- El análisis sintáctico o fase gramatical de un compilador, agrupa los tokens en estructuras sintácticas en forma muy similar a como teníamos que estructurar las oraciones en la primaria.

- El niño pequeño corrió rápidamente
  - (estructurada)
- Rápidamente pequeño el corrió niño
- Pequeño rápidamente niño el corrió
  - (no estructurada)

Análisis sintáctico gramatical

- El niño pequeño corrió rápidamente

- Frase sustantiva y frase verbal

- Artículo, sustantivo, adjetivo
- Verbo y un adverbio

Análisis sintáctico gramatical: ejemplo

- bb*12
- Expresión
- X1:=a+bb*12
- sentencia de asignación

- determina si la secuencia de tokens es sintácticamente correcta, de acuerdo con la definición del lenguaje.

Análisis sintáctico gramatical

- La estructura reconocida por el análisis sintáctico **se describe en forma semejante a un árbol y es conocido como árbol sintáctico**, árbol gramatical o árbol de estructura.

Análisis sintáctico gramatical: ejemplo

- X1:=a+bb*12;
- X2:=a/2+bb*12;

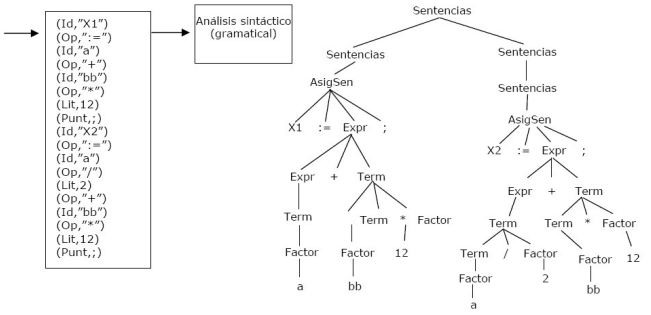
Análisis sintáctico gramatical: ejemplo

- X1:=a+bb*12;
- X2:=a/2+bb*12;

- Un ***TÉRMINO*** es un producto y cada parte de este producto se conoce como ***FACTOR***. Una suma de términos y factores se denomina ***EXPRESIÓN***.

Análisis sintáctico gramatical: ejemplo

- X1:=a+bb*12;
- X2:=a/2+bb*12;



Análisis Semántico

Análisis Semántico

- La fase de análisis semántico sigue a la fase de análisis gramatical y toma como entrada el árbol de análisis gramatical creado en la fase de análisis sintáctico.
- **Esta fase determina el significado (semántica) del programa** mediante la creación de tablas de símbolos, verificando cuales de las variables utilizadas ha sido definidas, y una infinidad de otras tareas anteriores a la generación de código

Análisis Semántico

- Dos tareas principales en la fase del análisis semántico son: La verificación estática del programa y la generación de una representación intermedia (RI).
- ***Verificación estática***: La generación estática completa del análisis iniciado por el analizador gramatical y efectúa actividades como la afirmación de que una variable con valor de caracter no haya sido asignada a una variable declarada como valor entero. Esto se conoce como verificación de tipos. Verificación de tipo y número de parámetros en la llamada a una rutina.
- ***Representación intermedia (RI):*** La representación intermedia, en ocasiones denominado Lenguaje Intermedio o Código Intermedio, es una forma alternativa para un árbol de análisis gramatical. A veces el analizador gramatical crea su representación intermedia en forma directa; y en ocasiones el árbol de éste se convierte a la representación.

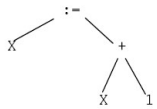
Análisis Semántico

- La **salida** "teórica" de la fase de análisis semántico sería **un árbol semántico**. Consiste en un árbol sintáctico en el que cada una de sus ramas ha adquirido el significado que debe tener.

La sentencia de asignación:

X:=X+1

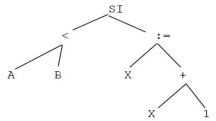
Tiene un árbol abstracto de sintaxis:



Un árbol de abstracto sintáctico para:

SI (A<B) ENTONCES X:=X+1

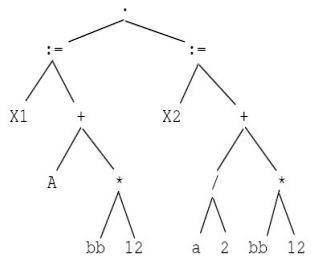
Es:



Análisis Semántico

Ejemplo 6:

El ejemplo de dos líneas convertido a un árbol abstracto sintáctico:

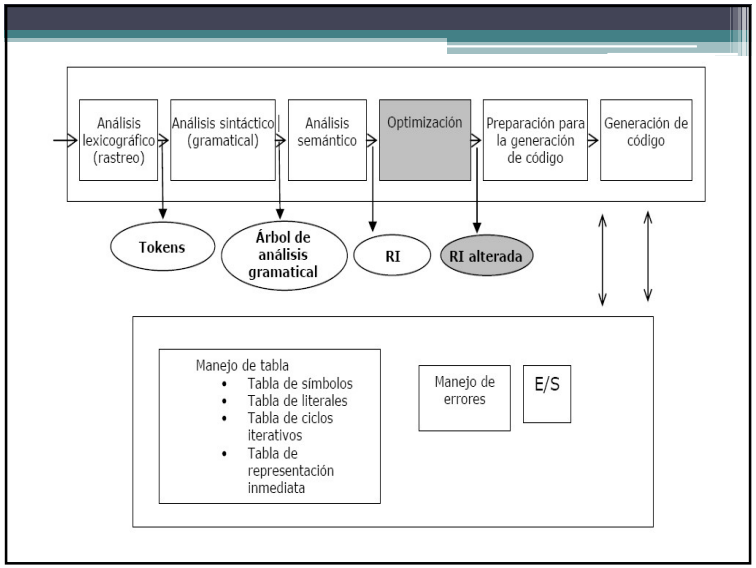


Análisis Semántico

- En el caso de los **operadores polimórficos** (un único símbolo con varios significados), el **análisis semántico determina cuál es el aplicable**.
- A := B + C

- En Pascal, el signo "+" sirve para sumar enteros y reales, concatenar cadenas de caracteres y unir conjuntos. El análisis semántico debe comprobar que B y C sean de un tipo común o compatible y que se les pueda aplicar dicho operador. Si B y C son enteros o reales los sumará, si son cadenas las concatenará y si son conjuntos calculará su unión.

Optimización



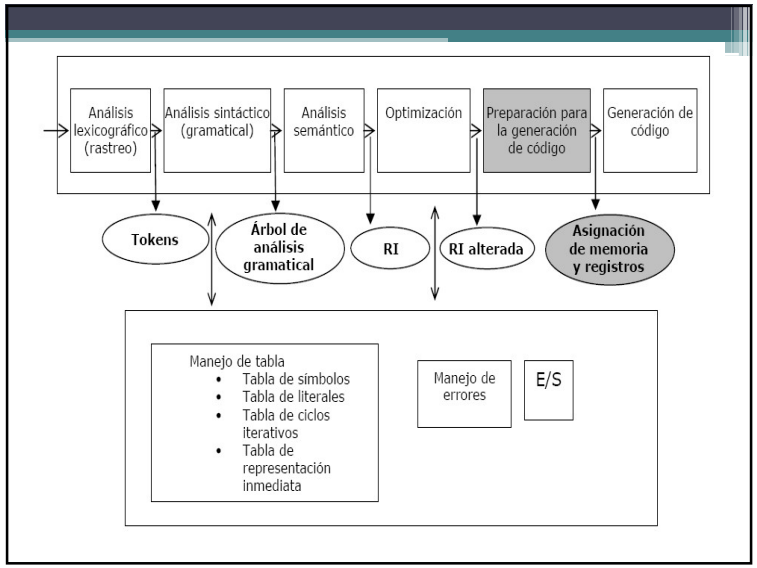
Optimización

- La fase de optimización **cambia la representación** intermedia de modo que la fase final de generación de código producirá **código que se ejecutará más rápido u ocupará menos espacio** (o ambas cosas).
- Debe al menos eliminar código que no hace nada así como variables no referenciadas. Un buen optimizador de código debe modificar la manera como se pretende implementar un algoritmo para que este sea ejecutado más eficientemente, los optimizadores de código pueden configurarse para optar por ahorro de memoria o por ahorro de tiempo de procesador.

Optimización

- Pueden aplicarse optimizaciones en diferentes etapas de la compilación:
  - durante la creación de la representación intermedia,
  - durante la transformación de una representación intermedia en otra,
  - durante la traducción del código intermedio a la salida,
  - luego de generar la salida

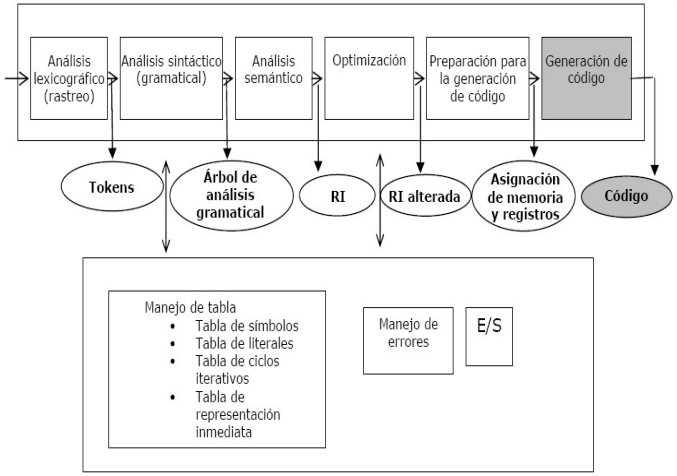
Preparación para la generación de código



Preparación para la generación de código

- Dos de las cuestiones en la preparación de la generación de código son, entonces, la ***ASIGNACIÓN DE MEMORIA Y LA ASIGNACIÓN DE REGISTROS***.
- **Asignación de memoria**: La asignación y el mantenimiento de espacio de memoria para conservar los valores de las variables y expresiones se conoce como asignación de memoria. Uno de los dilemas en la asignación de memoria es si la memoria debe ser asignada en forma estática, esto es, fijada en tiempo de compilación, o si el almacenamiento puede estar en una pila cuyo tamaño cambia a medida que se ejecuta el programa.
- **Asignación de registros**: Los registros se utilizan para conservar los valores de las variables y expresiones.

Generación de código



Generación de código

- La generación de código significa la traducción de la representación intermedia a un lenguaje como ensamblador.

Tabla de símbolos

Tabla de símbolos

- Es una estructura de datos que contiene **un registro para cada identificador utilizado en el código fuente**, con campos que contienen información relevante para cada símbolo (atributos).
- Tiene información acerca de los identificadores que aparecen en el código fuente, como el tipo de identificador (nombre de función, nombre de una variable, etc.), la **memoria** asignada, el **ámbito** o **alcance** del mismo, etc.

Tabla de símbolos

- El **analizador léxico introduce registros en la tabla de símbolos** pero deja en blanco campos cuya contenido no se puede determinar durante el análisis lineal, estos serán llenados y/o utilizados por las fases restantes.
- Durante la Generación de Código **se ingresa información para los atributos de los símbolos**, y se usa esa información de diversas maneras.
- Durante la Generación de Código puede ser necesario incorporar **nuevas entradas a la Tabla** de Símbolos.

Manejador de errores

Manejo de errores

- Es necesario que el compilador sea capaz de **recuperarse de los errores** encontrados y de **proporcionar mayor información** que conduzca a la fácil reparación de los mismos.

Manejo de errores

- La fase de análisis léxico detecta errores donde los **caracteres no forman ningún token valido** o en donde aparezcan caracteres inválidos.
- La fase de análisis sintáctico detecta errores donde se **violan las reglas de sintaxis**.
- La fase de análisis semántico detecta errores donde las construcciones son sintácticamente correctas, pero **no tienen ningún sentido**

Manejo de errores

- Por ejemplo, la construcción
- id1 + id2
- no tiene coherencia si id1 es el nombre de una función y id2 es el nombre de un arreglo

Ejemplo

- https://www.youtube.com/watch?v=SRIWJQ-mLcg

Síntesis de la sesión