Lenguajes y Compiladores

Análisis Léxico

Expresiones Regulares

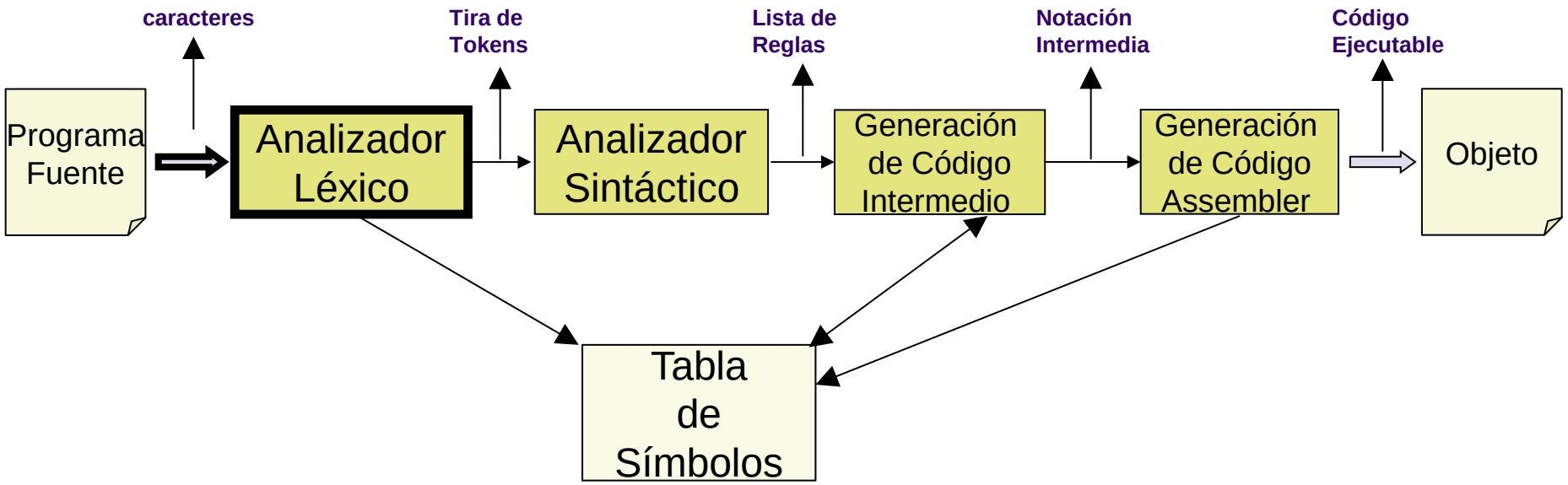
Analizador Lexicográfico

La compilación es un proceso destructivo en el cual el programa fuente se transforma por completo (o casi) en un lenguaje que la máquina comprende y ejecuta (objeto).

Este proceso contiene básicamente 4 etapas diferentes:

- Análisis léxico
- Análisis sintáctico
- Generador de código intermedio
- Generador de código assembler

Analizador Lexicográfico



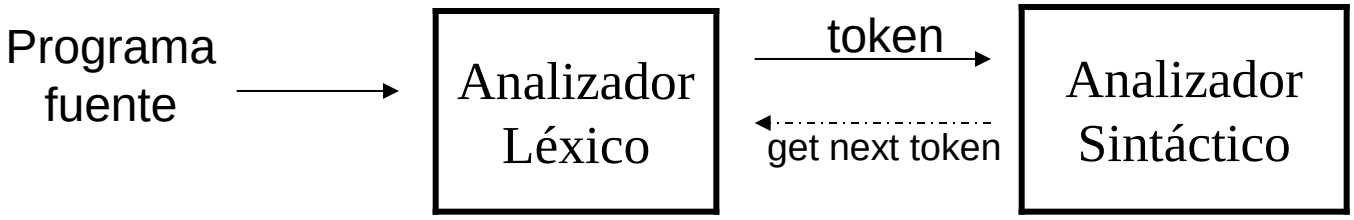
Análisis Léxico

- Lee el programa fuente.
- Agrupa los caracteres en unidades llamadas ***tokens***.

token: Secuencia de caracteres que forman una unidad significativa

Análisis Léxico

- Normalmente, un Analizador Léxico no retorna una lista de tokens, sino que retorna un token cuando el Analizador Sintáctico se lo pide



Tokens

- Identificadores (cadena que comienza con una letra y continúa con letras y/o dígitos)
- Constantes (secuencia de dígitos)
- Palabras reservadas
  - Ejemplos: IF, THEN, ELSE
- Operadores
  - Ejemplos: ‘+’, ‘-’, ‘*’, ‘/’
- Comparadores
  - Ejemplos: ‘>’, ‘<‘, ‘>=‘, ‘<=‘, ==
- etc.

Tokens

- Los tokens se diferencian de la cadena de caracteres que representan.
- La cadena de caracteres es el ***Lexema*** o valor léxico.
  - Existen tokens que se corresponden con un único lexema
    - Ejemplo: Palabra Reservada **IF**
  - Existen tokens que pueden representar lexemas diferentes
    - Ejemplo: Identificador **Plazo**, **main**, **Tasa, vec1**

Análisis Léxico

Ejemplo

Cualquier Programador que lea el siguiente código:

```
begin
        total .= parcial * 21- desc
end
```

Entenderá que en esta sentencia de asignación hay:

- Una palabra reservada: **begin**
- Un identificador: **total**
- Un operador de asignación: **.=**
- Un identificador: **parcial**
- Un operador de multiplicación: *****
- Una constante: **21**
- Un operador de sustracción: **-**
- Un identificador: **desc**
- Una palabra reservada: **end**

Tokens

- Puesto que un token puede representar más de un lexema, el A.L. debe enviar información adicional al A.S., en forma de atributo/s. Esa información será usada en las próximas etapas del compilador.
- En la práctica, la información adicional para cada token que genere más de un lexema se almacena en una **Tabla de Símbolos**, y los atributos de cada token difieren de acuerdo a sus características

Tabla de Símbolos

En la práctica la Tabla de Símbolos se puede implementar con una estructura de datos que contiene un registro para cada token que pueda representar más de un lexema y los atributos del mismo

NOMBRE	TIPO	VALOR	ALIAS	LIMITE	LONGITUD
a1	Real	__		10	
b1	Real	__			
_variable1	CteString	variable1			9
_30.5	CteReal	30.5			
_USD	CteString	USD	DOL		3
p1	Pointer Real	__			

Análisis Léxico: Funciones

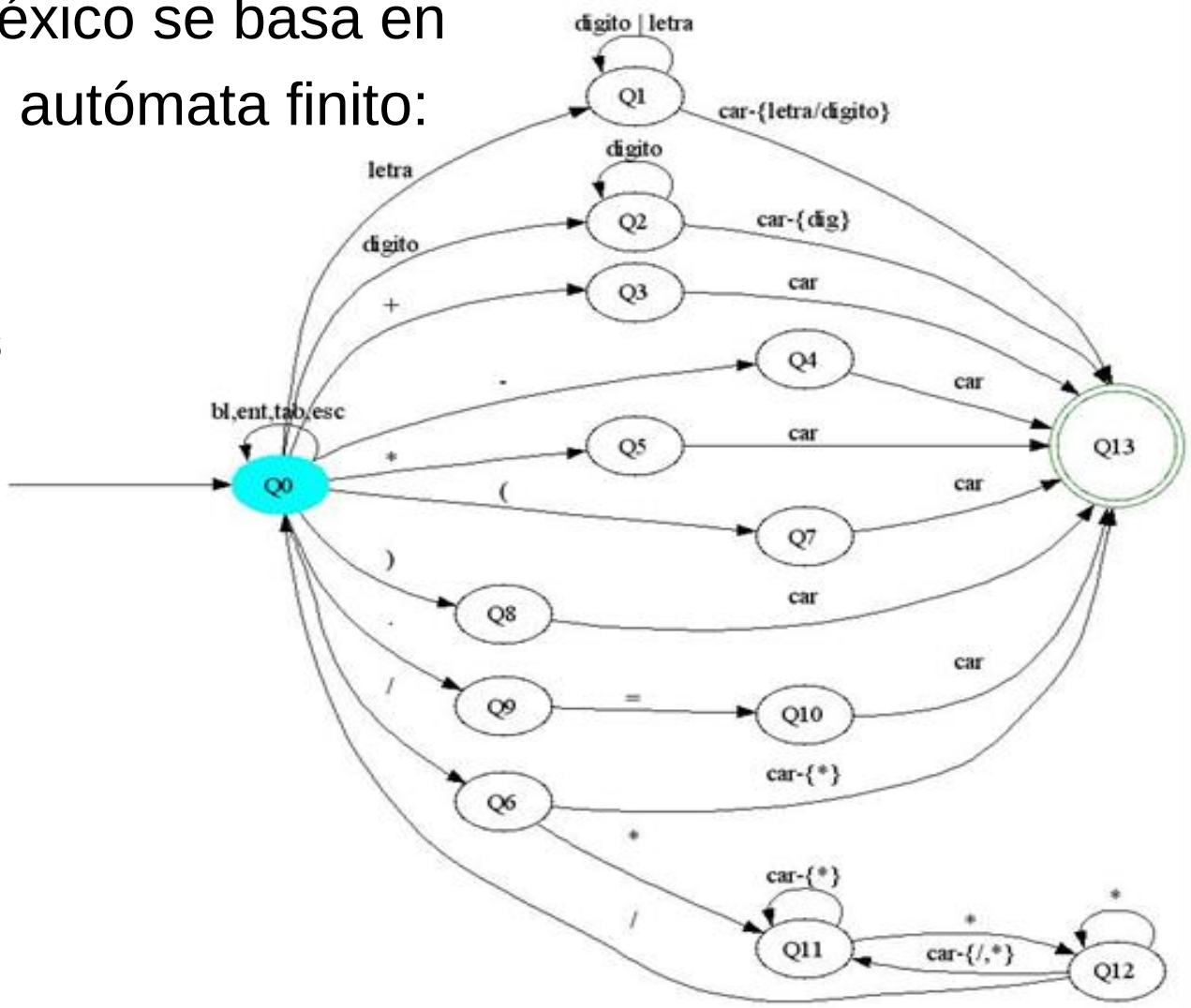
- Reconocer tokens
- Eliminar blancos, tabulaciones, newlines
- Llevar la cuenta de los saltos de línea, para correlacionar los mensajes de error con el programa fuente
- Expandir macros
- Eliminar comentarios
- Informar errores léxicos

Análisis Léxico: Diseño

Un analizador léxico se basa en el diseño de un autómata finito:

Autómata Finito

grafo con estados finitos y transiciones entre ellos



Análisis Léxico: Diseño

- En general el analizador léxico es una función que lee carácter a carácter la entrada (programa) y va recorriendo el autómata hasta que reconoce un token , esto último sucede cuando llega a un estado final.

- Una vez que reconoció un componente léxico o token devuelve el tipo de token reconocido ó un error y en general es invocada cada vez que el analizador sintáctico requiere que se chequee la sintaxis de un token.

Análisis Léxico: Diseño

El autómata se modela manualmente o con **expresiones regulares (Regex)**.

- Existen herramientas como LEX, FLEX que se ocupan de armar el autómata, la función que lo recorre y analiza los tokens y, a las que sólo es necesario escribirles las expresiones regulares correspondientes.
- El resultado que entregan es una función externa que puede ser invocada directamente de las implementaciones de las próximas etapas.
- El autómata y su implementación pueden desarrollarse sin el uso de estas herramientas.

Análisis Léxico: Expresiones Regulares

- Las expresiones regulares son patrones que se utilizan para definir/testear cadenas de caracteres de un lenguaje regular.
- Los lexemas son cadenas de un lenguaje regular.
- Las expresiones regulares tienen un lenguaje propio
- Muchos lenguajes de programación las incorporaron, ya sea directamente o a través de una biblioteca de clases.

Análisis Léxico: Expresiones Regulares

- El lenguaje de expresiones regulares contiene los símbolos : ( ) * + ? , etc
- Esto depende de cada implementación

- En general:
  - x^* denota cero o más ocurrencias de x
  - x^+ denota una o más ocurrencias de x
  - x? denota ninguna o una ocurrencia de x
  - r/s denota una cadena que es representada por la ER r o por la expresión regular s
  - r.s denota una cadena que es representada por la ER r seguida de la expresión regular s

Análisis Léxico: Expresiones Regulares

DIGITO	[0-9]
LETRA	[a-zA-Z]

CONST_REAL	{DIGITO}+"."{DIGITO}+
CONST_INT	{DIGITO}+
ID	{LETRA}({LETRA}|{DIGITO}|_)*

Análisis Léxico: Errores

No son muchos los errores que puede detectar un analizador léxico. Entre los más característicos se encuentran:

- Caracter inválido durante el mapeo de los símbolos de entrada
- Constante fuera de rango
- Identificador excede el tamaño determinado
- Comentario sin cerrar

Análisis Léxico

Página LyCUnlam – Descargas

Instalador Flex-Bison
Readme para instalación
Ejemplos Varios
Apunte de uso de Flex
Apunte de uso de Bison

¿Preguntas?