

Final Teoria 4/7/2019

1-) El Proceso de Compilacion tiene 2 etapas, la primera etapa de análisis o también llamada front end que es dependiente del lenguaje. Y otra etapa llamada de síntesis o backend que es independiente del lenguaje y dependiente de la máquina. Esta etapa solo se ejecuta si la etapa de Análisis no ocurre ningún error.

Etapa de Análisis: en esta etapa entra en juego los analizadores léxico, sintáctico y semántico. También se utilizan las tablas de tipos y tabla de símbolos. Resumidamente el analizador léxico separa los caracteres del código fuente y los agrupa formando los tokens retorna a medida que el AS lo solicita los tokens. El AL opera bajo petición del AS. Se dice que es una subrutina del AS. El AS recibe los token generados por el AL y se encarga de Agruparlos para construir las sentencias de acuerdo a las reglas gramaticales. Es decir la tarea del AS es determinar si un programa es correcto o no desde

la sintaxis. Produce como salida un árbol sintáctico. Luego el Analizador Semántico debe corroborar que exista coherencia en el programa. Para esto consulta la información en la tabla de tipos y tabla de símbolos.

En este paso, recibe los árboles sintáticos del AS y evalúa que tengan un significado de acuerdo a las reglas semánticas del lenguaje y produce como salida árboles semánticos.

En caso de haber errores en alguno de estos analizadores se almacenan en la tabla de gestión errores y se muestran razón al finalizar la etapa de análisis.

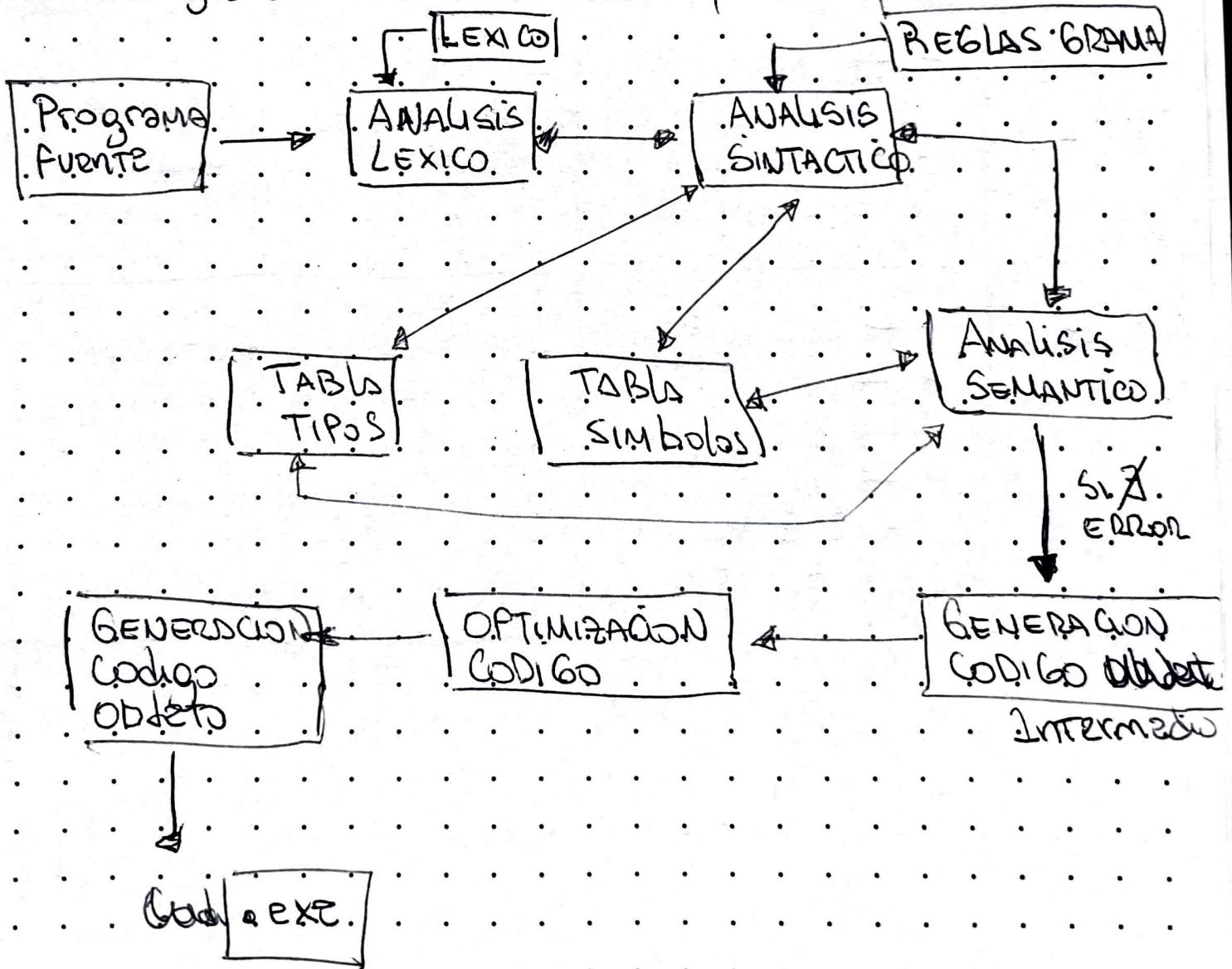
Etapa de síntesis: En esta etapa entra la generación del Código objeto.

Generación de Código intermedio: Luego del AS y ASE algunos compiladores generan un código intermedio de fácil traducción al lenguaje de máquina de un determinado procesador.

Generación de Código objeto: Genera el código ejecutable.

El código objeto se puede generar en

- Lenguaje de Máquina absoluto: el programa se aloja en una posición fija de memoria. Se ejecuta con el .exe.
- Lenguaje de Máquina reinterpretable: las posiciones de memoria son dinámicas. Se genera el .exe con un link editor.
- Lenguaje simbólico: útil en compus. con pocos recursos.



3-)

GRAMATICA	PRODUCCIONES	LENGUAJE	MAQUINA
GRAMATICA TIPOS 0	$\alpha \rightarrow \gamma$ $\alpha \in (V_T \cup V_N)^*$ $\gamma \in (V_T \cup V_N)^*$	Lenguaje irrestricto	MÁQUINA TURING
GRAMATICA TIPOS 1	$\alpha A \beta \rightarrow \alpha \gamma \beta$ $A \in V_N$ $\alpha \beta \in (V_T \cup V_N)^*$ $\gamma \in (V_T \cup V_N)^*$	Lenguaje Dependiente CONTEXTO	ALFA
GRAMATICA TIPOS 2	$A \rightarrow \gamma$ $A \in V_N$ $\gamma \in (V_T \cup V_N)^*$	Lenguaje Independiente CONTEXTO	PUSH - DOWN
GRAMATICA TIPOS 3	$A \rightarrow \alpha B \beta$ (Der) $A \rightarrow \alpha$ $A \rightarrow \beta \alpha$ (Iza) $A \rightarrow \alpha$ $A, B \in V_N$ $\alpha \in V_T$	Lenguaje regulares	AUTOMATA FINITO

5-)

a-) $V_T = \{a, c\}$; $V_N = \{S, B, C\}$

b-) $S \rightarrow aB$ $S \rightarrow CC$
 aC CC
 $S \rightarrow aB$
 aa

L(b) = { ja, ab, ac }.

c-) Es una gramática de t.p. 3. o regular.

d-) La máquina que lo reconoce es el AUTOMATA FINITO.

4-) PASOS PARA PASAR DE UNA E.R. A UN AFDO.

1-) Determinar la expresión regular que representa al lenguaje.

2-) Mediante el método de thompson construir la AFN que representa la expresión regular.

3-) Mediante el método de subconjunto armar un AFD.

2. partir del AFN que reconoce el mismo lenguaje que el AFD.

4-) OPTIMIZAR el AFD para llegar al AFDO.

2.-) AUTOMATA FINITO: es una maquina que reconoce lenguajes regulares. Se puede representar graficamente con una cinta y un control de estados. La cinta contiene los simbolos del alfabeto de entrada. Puede ser infinita.

Los AF. se componen de 3 tipos de estados, un estado inicial, un estado final y estados intermedios para llegar desde el estado inicial al estado final.

Para que un AF. reconozca un lenguaje debe ser capaz de reconocer todas las cadenas que pertenecen al lenguaje y no puede reconocer una cadena que no pertenece al lenguaje.

Diferencias entre AFD y AFN.

- En los AFD no existen las transiciones ϵ , en cambio en un AFN si son posibles.
- En los AFD para un estado S y un solo simbolo " a " hay a lo sumo una transicion a otro estado. En cambio en los AFN puede haber 0, uno o mas transiciones de forma simultanea.
- Los AFD se obtienen con el metodo de subconjunto previamente habiendo obtenido el AFN con el metodo de thompson.

- (b) • Los AFD debido a que son deterministas son menos complejos que los AFN. Ya que tienen varias opciones de transiciones a otros estados.
- Algunos lenguajes no pueden ser reconocidos por un AFD pero si por un AFN.
-

6-) $(a \mid b^*) \cdot b$

