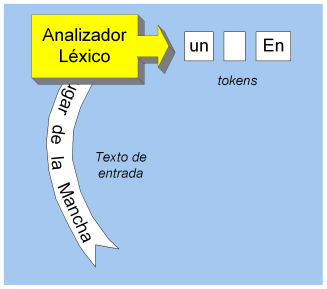
INVESTIGA SOBRE LAS FASES DEL PROCESO DE COMPILACIÓN.

Investiga sobre las fases del proceso de compilación.

Análisis léxico:

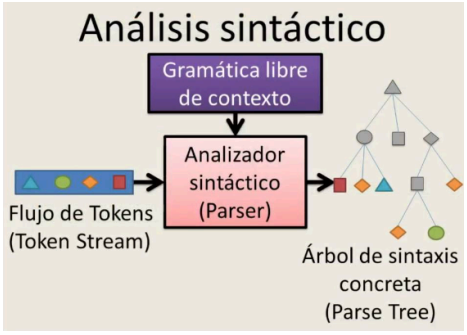
El análisis léxico es la primera fase del proceso de compilación, en la que el código fuente se convierte en una secuencia de tokens.
¿Qué es un token?
Un token es una unidad básica del lenguaje, como palabras clave, identificadores, operadores o delimitadores. En esta etapa, se eliminan los espacios en blanco y los comentarios, y se detectan posibles errores como el uso de caracteres no permitidos.
¿Cuál es el objetivo principal?
El objetivo principal es identificar correctamente las piezas que componen el código, para que puedan ser procesadas en fases posteriores como el análisis sintáctico y semántico.



Análisis sintáctico
El análisis sintáctico es la segunda fase del proceso de compilación, en la que se organiza la secuencia de tokens obtenida en el análisis léxico para verificar si siguen una estructura válida según las reglas gramaticales del lenguaje de programación.

¿Qué es una estructura válida?
Es un conjunto de reglas que definen cómo deben combinarse los tokens para formar sentencias o expresiones correctas en el lenguaje. Por ejemplo, una estructura válida sería una declaración de variable bien formada o una expresión matemática correctamente escrita.

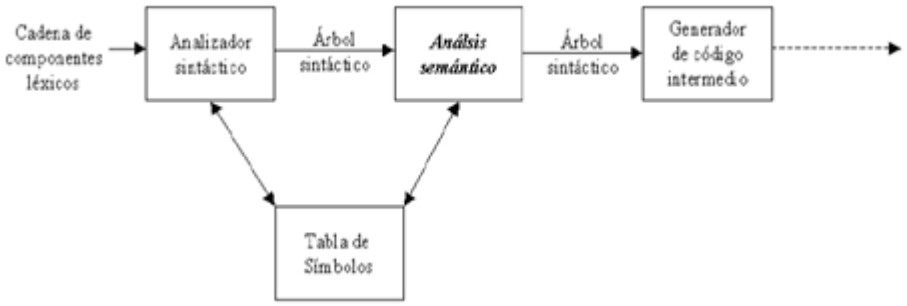
¿Cuál es el objetivo principal?
El objetivo principal es construir un árbol sintáctico que representa la estructura jerárquica del código y detectar errores de sintaxis, como paréntesis mal colocados o declaraciones mal formadas, para garantizar que el código tenga sentido gramaticalmente.



Análisis semántico:
El análisis semántico es la tercera fase del proceso de compilación, donde se verifica que las estructuras creadas en el análisis sintáctico tengan un significado coherente y correcto dentro del contexto del lenguaje.

¿Qué significa que algo sea semánticamente correcto?
Ser semánticamente correcto significa que las operaciones y expresiones en el código tienen sentido. Por ejemplo, asegurarse de que no se intente sumar un número con una cadena de texto o que una variable esté correctamente declarada antes de su uso.

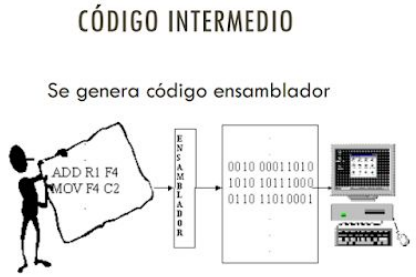
¿Cuál es el objetivo principal?
El objetivo principal es detectar errores lógicos o de tipo en el código, como incompatibilidades en los tipos de datos, uso indebido de variables o llamadas incorrectas a funciones. Todo esto garantiza que el programa no solo sea sintácticamente válido, sino que también sea lógicamente correcto.



Generación de código intermedio:
La generación de código intermedio es la cuarta fase del proceso de compilación, donde se traduce el código fuente en una representación más simple y abstracta, que es independiente de la máquina.

¿Qué es el código intermedio?
El código intermedio es una representación a medio camino entre el código fuente y el código máquina. Suele estar en un formato más fácil de optimizar y trasladar a diferentes arquitecturas. Un ejemplo común es el uso de tres direcciones (instrucciones que usan a lo sumo tres operandos).

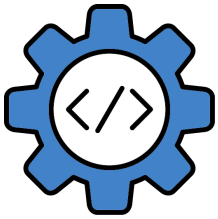
¿Cuál es el objetivo principal?
El objetivo principal es crear un puente entre el código de alto nivel y el código específico de la máquina. Esto facilita la optimización y mejora la portabilidad del compilador, permitiendo que el código se traduzca a diferentes arquitecturas de hardware de manera más eficiente.



Optimización del código intermedio:
La optimización del código intermedio es la fase del proceso de compilación donde se mejoran las instrucciones generadas en la etapa anterior, para hacer el código más eficiente sin alterar su comportamiento.

¿Qué implica optimizar el código?
Optimizar el código implica reducir el número de instrucciones, mejorar el uso de recursos (como registros y memoria), eliminar cálculos redundantes y simplificar operaciones complejas.

¿Cuál es el objetivo principal?
El objetivo principal es mejorar el rendimiento del código, tanto en términos de velocidad de ejecución como de uso de memoria, sin cambiar el resultado final del programa. Esto permite que el código sea más rápido y eficiente cuando se ejecute en la máquina final.



Generación del código final:
La generación del código final es la fase en la que el código intermedio optimizado se traduce en instrucciones específicas para la arquitectura del procesador en la que el programa será ejecutado.

¿Qué es el código final?
El código final es el conjunto de instrucciones en lenguaje máquina o ensamblador que el procesador puede ejecutar directamente. Este código es totalmente dependiente de la plataforma para la que se compila.

¿Cuál es el objetivo principal?
El objetivo principal es producir un código ejecutable que sea entendible y ejecutable por el hardware, asegurando que el programa funcione de manera óptima en el sistema objetivo.