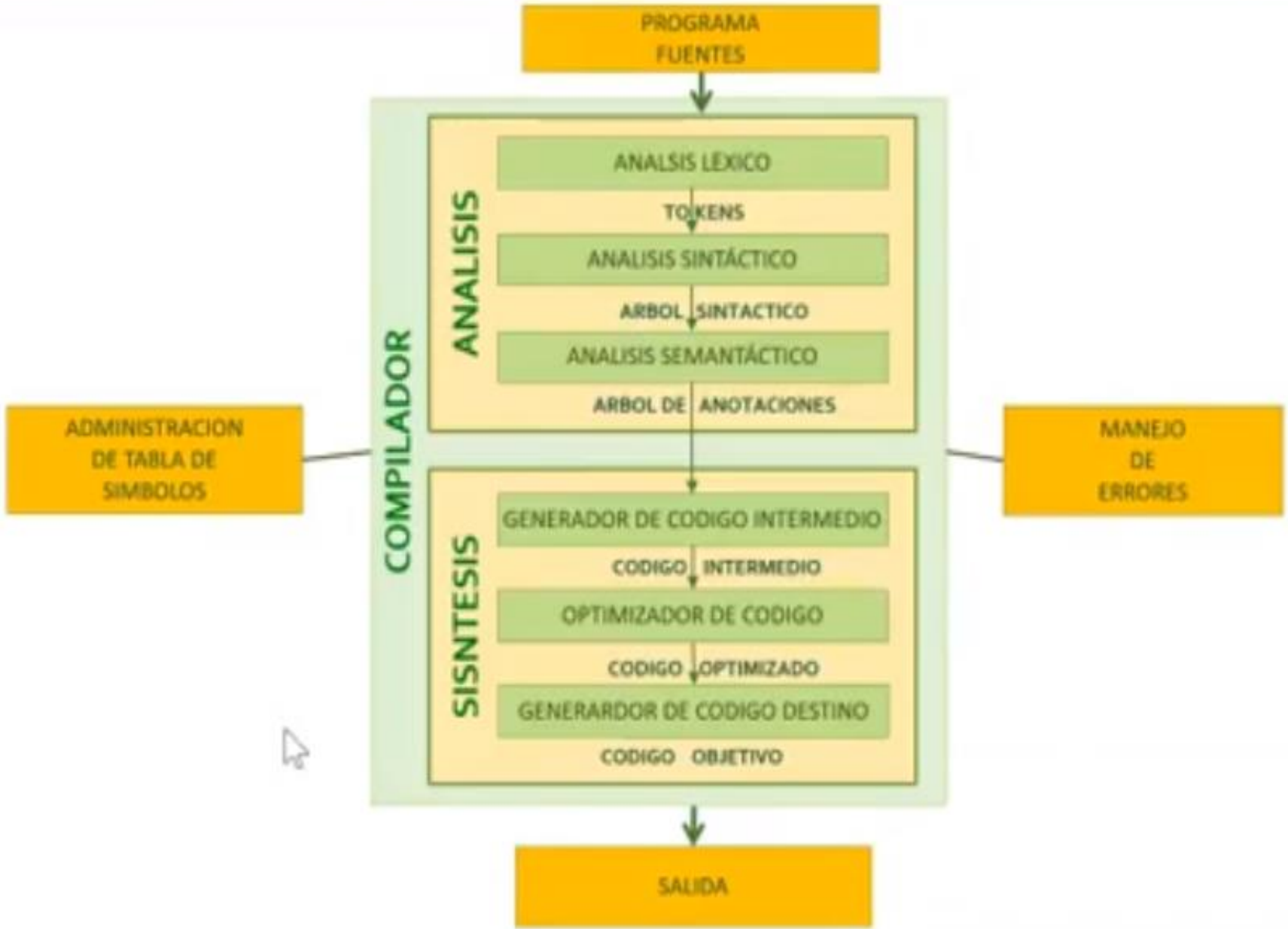
Estructura general del compilador

La parte de la **compilación** está dividida en dos tareas principales:

Análisis (Front End): En la tarea del análisis se divide al programa fuente en sus elementos, componentes y crea una representación intermedia del programa fuente.

Síntesis (Back End): La tarea de la síntesis construye el programa objeto deseado a partir de la representación intermedia. La *síntesis* es la que requiere las técnicas más especializadas.

Fases del compilador



Front End (Análisis)

Fases que dependen del lenguaje fuente

- Análisis Léxico.
- Análisis Sintáctico.
- Análisis Semántico (Estático).
- Creación de la Tabla de Símbolos.
- Generación de Código Intermedio.
- Optimización.
- Manejo de errores correspondiente a las fases del Front End.

Back End (Síntesis)

Fases que dependen de la máquina destino

- Generación de la salida.
- Optimización.
- Manejo de errores correspondiente a las fases del Back End.
- Operaciones sobre la Tabla de Símbolos.

Fases de la compilación

Análisis léxico (*explorador o scanner*)

Esta fase del compilador efectúa la lectura real del programa fuente, el cual generalmente está en la forma de un flujo de caracteres. El rastreador realiza lo que se conoce como análisis léxico: recolecta secuencias de caracteres en unidades significativas denominadas **tokens**: las cuales son como las palabras de un lenguaje natural, como el español.

Análisis Sintáctico (*explorador o scanner*)

El analizador sintáctico recibe el código fuente en la forma de **tokens** proveniente del analizador léxico y realiza el **análisis sintáctico**, que determina la estructura del programa. Esto es semejante a realizar el análisis gramatical sobre una frase en un lenguaje natural. El análisis sintáctico determina los elementos estructurales del programa y su relaciones. Los resultados del análisis sintáctico por lo regular se representan como un **árbol de análisis gramatical** o **un árbol sintáctico**.

Análisis Semántico

La semántica de un programa es su "significado", en oposición a su sintaxis, o estructura. La semántica de un programa determina su comportamiento durante el tiempo de ejecución, pero la mayoría de los lenguajes de programación tienen características que se pueden determinar antes de la ejecución e incluso no se pueden expresar de manera adecuada como sintaxis y analizarse mediante el analizador semántico.

Generador de código intermedio

El *código intermedio* es un código abstracto independiente de la máquina para la que se generará el código objeto. El código intermedio ha de cumplir dos requisitos importantes: ser fácil de producir a partir del análisis sintáctico, y ser fácil de traducir al lenguaje objeto. Esta fase puede no existir si se genera directamente código máquina, pero suele ser conveniente emplearla.

Optimizador de código

Esta etapa es utilizada para el mejoramiento del código o para su optimización. El punto más anticipado en el que la mayoría de las etapas de operación se pueden realizar es precisamente después del análisis semántico, y puede haber posibilidad para el mejoramiento del código que dependerán sólo del código fuente.

Generación de código destino

El generador de código destino toma el código optimizado y genera el código para la máquina objetivo (ensamblador). Las posiciones de memoria se seleccionan para cauda una de las variables usadas por el programa. Después, cada una de las instrucciones intermedias se traduce a una secuencia de instrucciones de máquina que ejecuta la misma tarea. Un aspecto decisivo es la asignación de variables a registro.