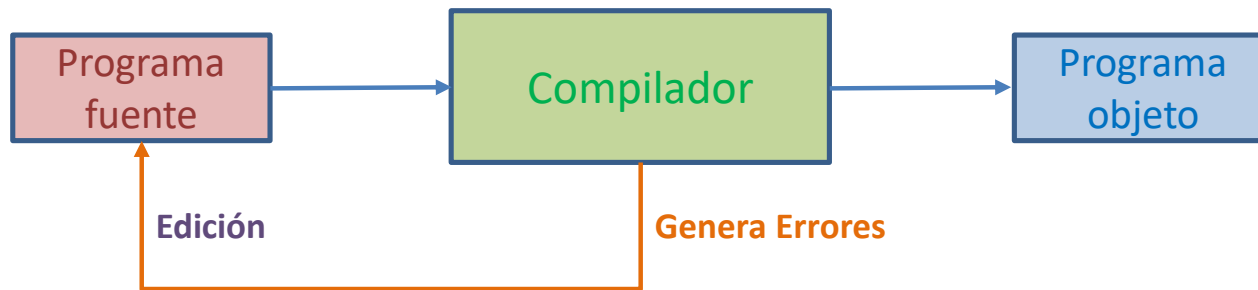


Introducción a los Compiladores

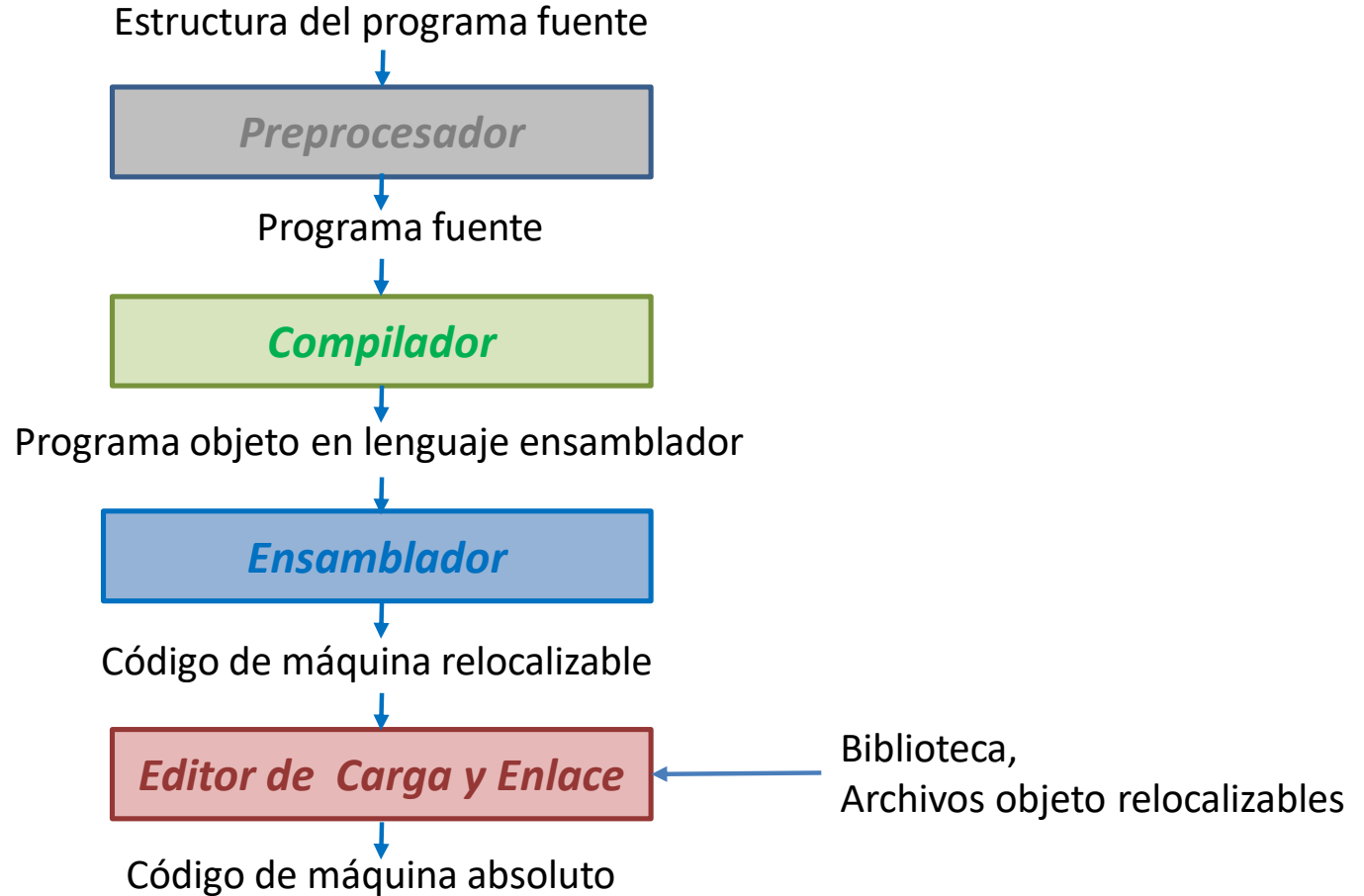
Compilador



Compilador: Programa que toma como entrada un **programa** escrito en un lenguaje **fuentes**, y lo traduce en un **programa objeto** equivalente.

Durante este proceso, el **compilador** informa al usuario de la presencia de **errores** en el **programa fuente** para su **edición**.

Contexto de un Compilador



Fases del Compilador

Etapas de Análisis:

Análisis Lineal o Léxico o Lexicográfico: La cadena de caracteres que constituye el **programa fuente** se lee de izquierda a derecha y de arriba hacia abajo.

Los caracteres leídos se agrupan en **componentes léxicos** (*tokens*) que son secuencia de caracteres con un significado colectivo.

La secuencia de caracteres que forman un *token* se denomina *lexema* del *token*.

Se eliminan los espacios en blanco, tabulaciones, y comentarios.

Ejemplo: Con el siguiente código fuente, desarrollar cada etapa:

$$x := 3 * a * b^2/c + 5 * b^3/d * c - 8/a * e$$

Fases del Compilador

Etaapa de Análisis:

Análisis Sintáctico o Jerárquico: Los **componentes léxicos** (*tokens*) se agrupan jerárquicamente en colecciones anidadas con un significado colectivo.

Se construye un **árbol sintáctico**. Si no se logra construir el árbol, se presenta un **error sintáctico**.

Se utilizan **reglas recursivas** para la construcción jerárquica.

Ejemplo: Se construye el árbol con el mismo código:

$$x := 3 * a * b^2/c + 5 * b^3/d * c - 8/a * e$$

Fases del Compilador

Etapas de Análisis:

Análisis Semántico: Se realizan revisiones para asegurar que los componentes de un programa se ajustan de un modo significativo.

Se lleva a cabo la **verificación de tipos** en el uso de datos y variables.

Ejemplo: Se realiza el análisis semántico con el mismo código:

$$x := 3 * a * b^2/c + 5 * b^3/d * c - 8/a * e$$

Fases del Compilador

Etapas de Síntesis:

Generación de código intermedio: Representación intermedia del programa fuente.

Se realizará mediante el *código de tres direcciones*, que es como el lenguaje ensamblador en que cada posición de memoria actúa como un registro.

Código de tres direcciones:

1. Cada instrucción tiene **máximo tres operandos**.
2. Tiene **máximo un operador**, además de la asignación.
3. Se generan **nombres temporales** para guardar cálculos.
4. Algunas instrucciones tienen **menos de tres operandos**.

Ejemplo: Se realiza la generación de código intermedio con el mismo código:

$$x := 3 * a * b^2/c + 5 * b^3/d * c - 8/a * e$$

Fases del Compilador

Etapas de Síntesis:

Optimización de código: Mejorar del código intermedio, de tal forma que el código se ejecute más rápidamente.

Una forma de mejorarlo es utilizando una instrucción para cada operador en la representación del árbol después del análisis semántico.

Es decir, todas las instrucciones quedan en formato de *código de tres direcciones*.

Ejemplo: Se realiza la optimización de código con el mismo código:

$$x := 3 * a * b^2/c + 5 * b^3/d * c - 8/a * e$$

Fases del Compilador

Etapas de Síntesis:

Generación de código objeto: Generar código de máquina relocizable o código ensamblador.

Ejemplo: Se realiza la generación de código objeto con el mismo código:

$$x := 3 * a * b^2/c + 5 * b^3/d * c - 8/a * e$$