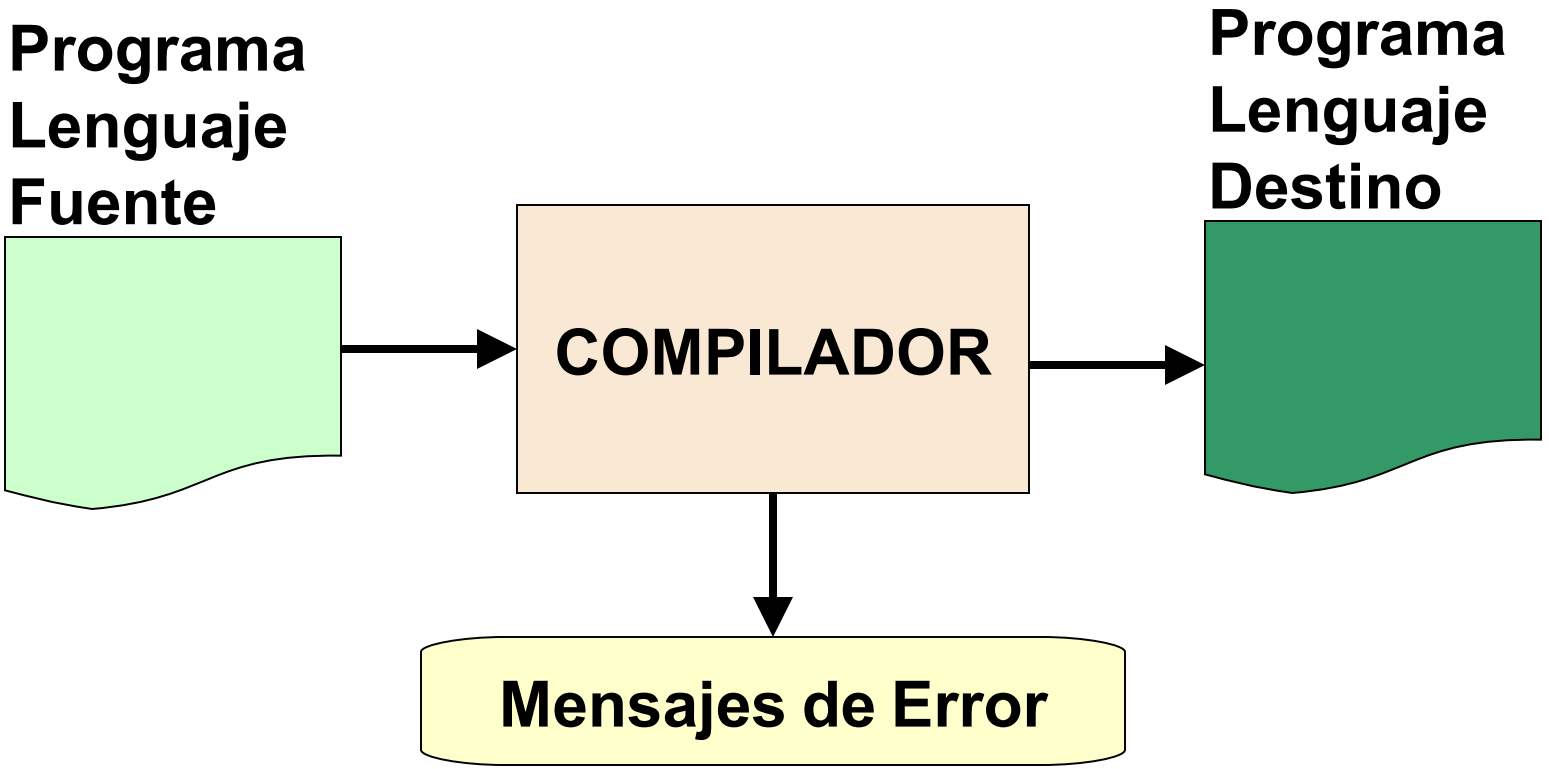
Compiladores: Introducción

Pontificia Universidad Javeriana Cali
Ingeniería de Sistemas y Computación
Prof. Gloria Inés Alvarez V.
(galvarez@puj.edu.co)
Basado en [Aho, 2007, chp. 1]

Qué es un COMPILADOR?

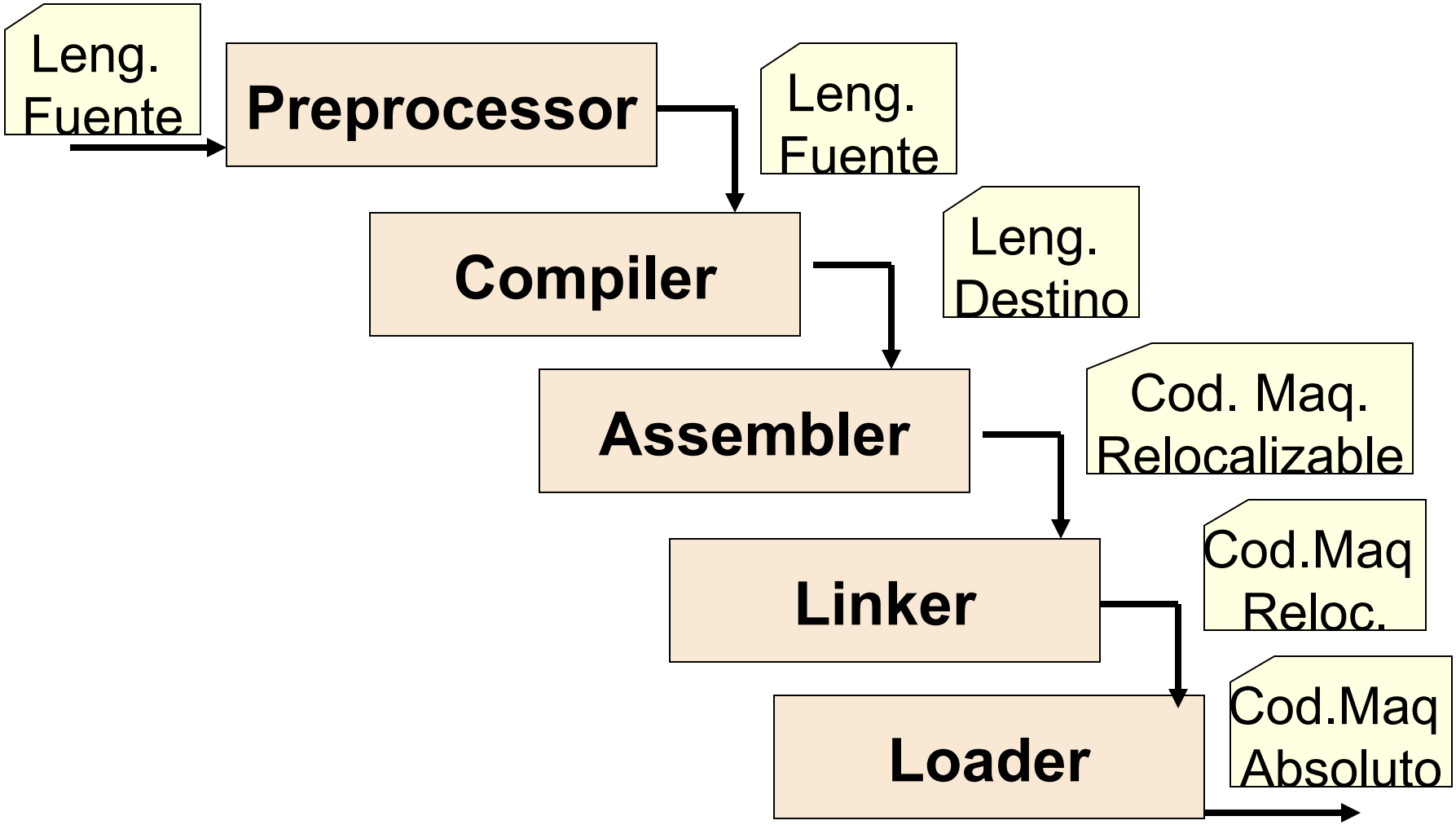


Qué vamos a aprender en este curso?

- Técnicas sistemáticas para manejar las principales tareas que ocurren durante el proceso de compilación.

El desarrollo del primer compilador de Fortran (1950´s) tomó 18 años hombre.

Contexto de un Compilador

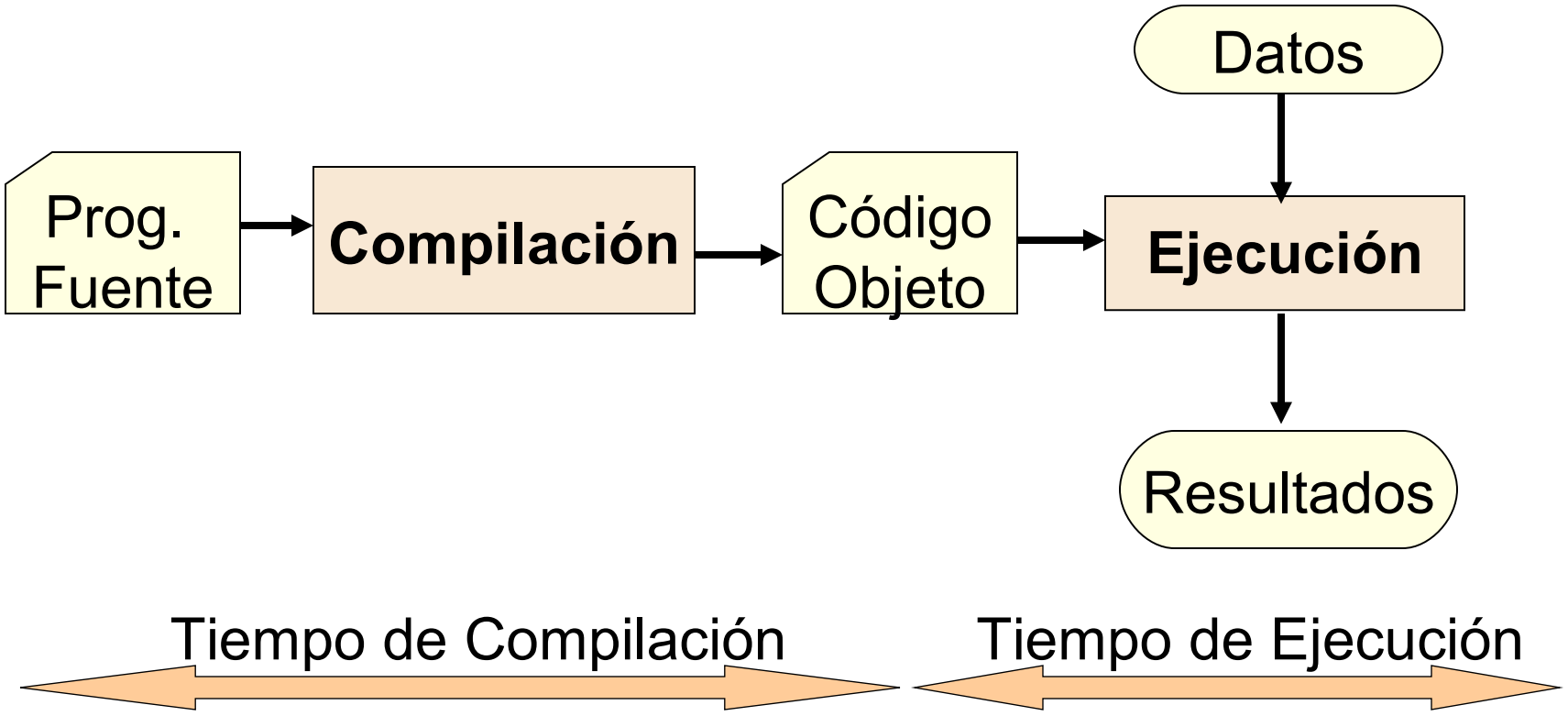


Diferencia entre Compilador e Interprete

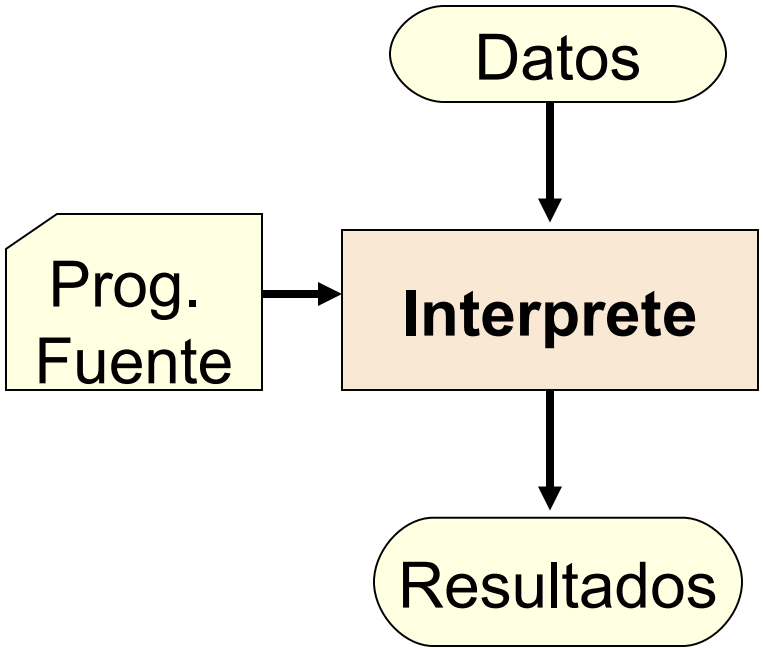
- Interprete: no realiza la traducción, desarrolla las operaciones que se especifican en el programa fuente. Procesa el programa y la entrada simultáneamente.

- Compilador.
  - Tiempo de Compilación
  - Tiempo de Ejecución

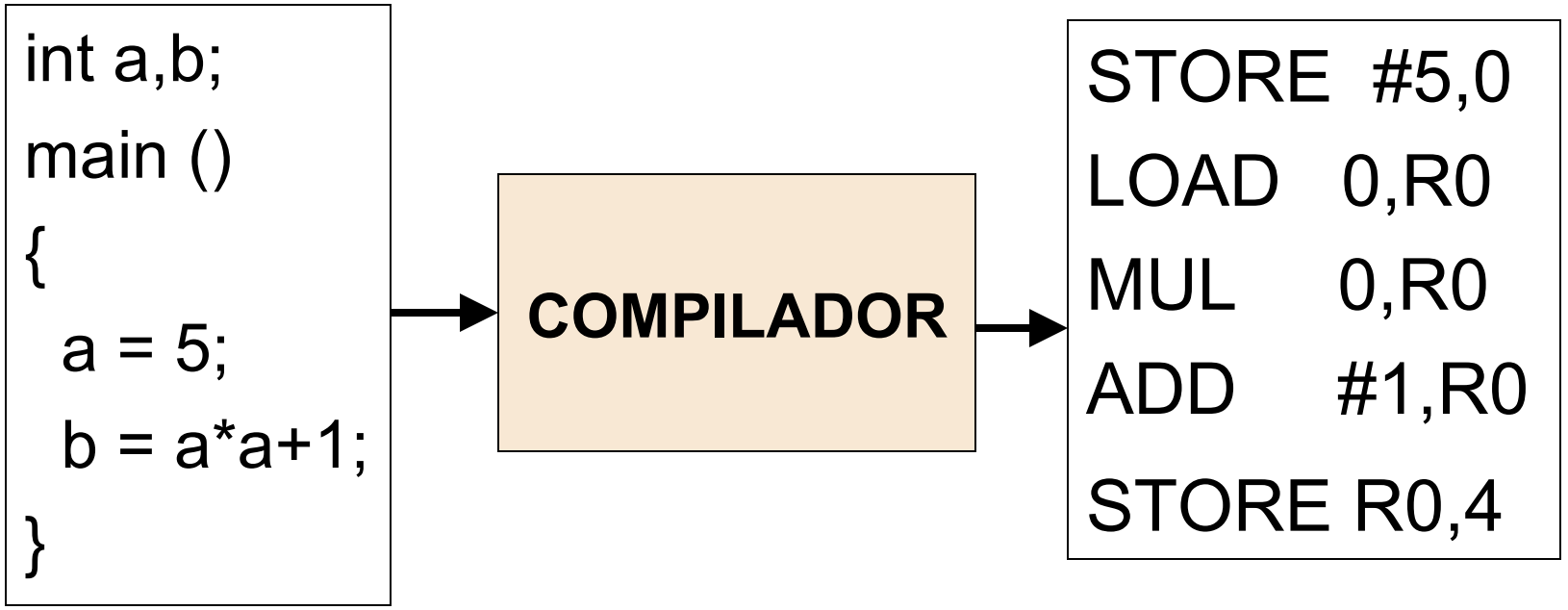
Compilador


Datos
Prog. Fuente
Compilación
Código Objeto
Ejecución
Resultados
Tiempo de Compilación
Tiempo de Ejecución

Interprete



Tarea del Compilador



Como lo hace?

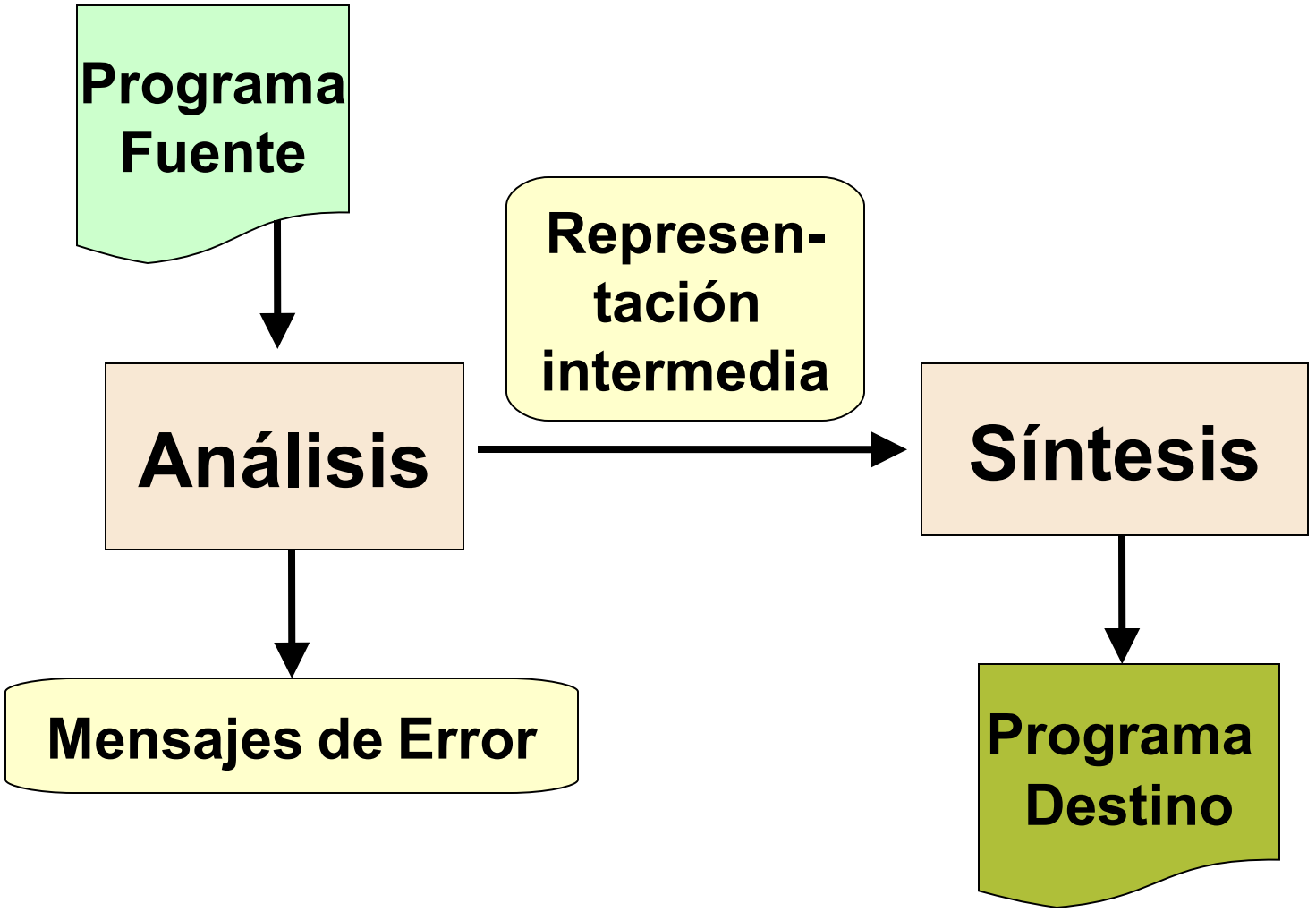
Compilar se parece a traducir

- Reconocer los símbolos (letras, dígitos, signos de puntuación).
- Agrupar los símbolos en palabras.
- Asociar a cada palabra significado y atributos.
- Verificar que la estructura de la oración sea correcta.
- Juntar los significados parciales y asegurarse que la idea resultante tiene sentido.

Compilar se parece a traducir

- Una vez se ha entendido el significado de una oración bien construida, se procede a realizar la acción indicada en ella. Para esto se generan instrucciones en el lenguaje que entiende el computador.
- El nuevo conjunto de instrucciones se revisa por si puede ser mejorado.
- Finalmente se ejecutan las instrucciones.

Modelo de Compilación

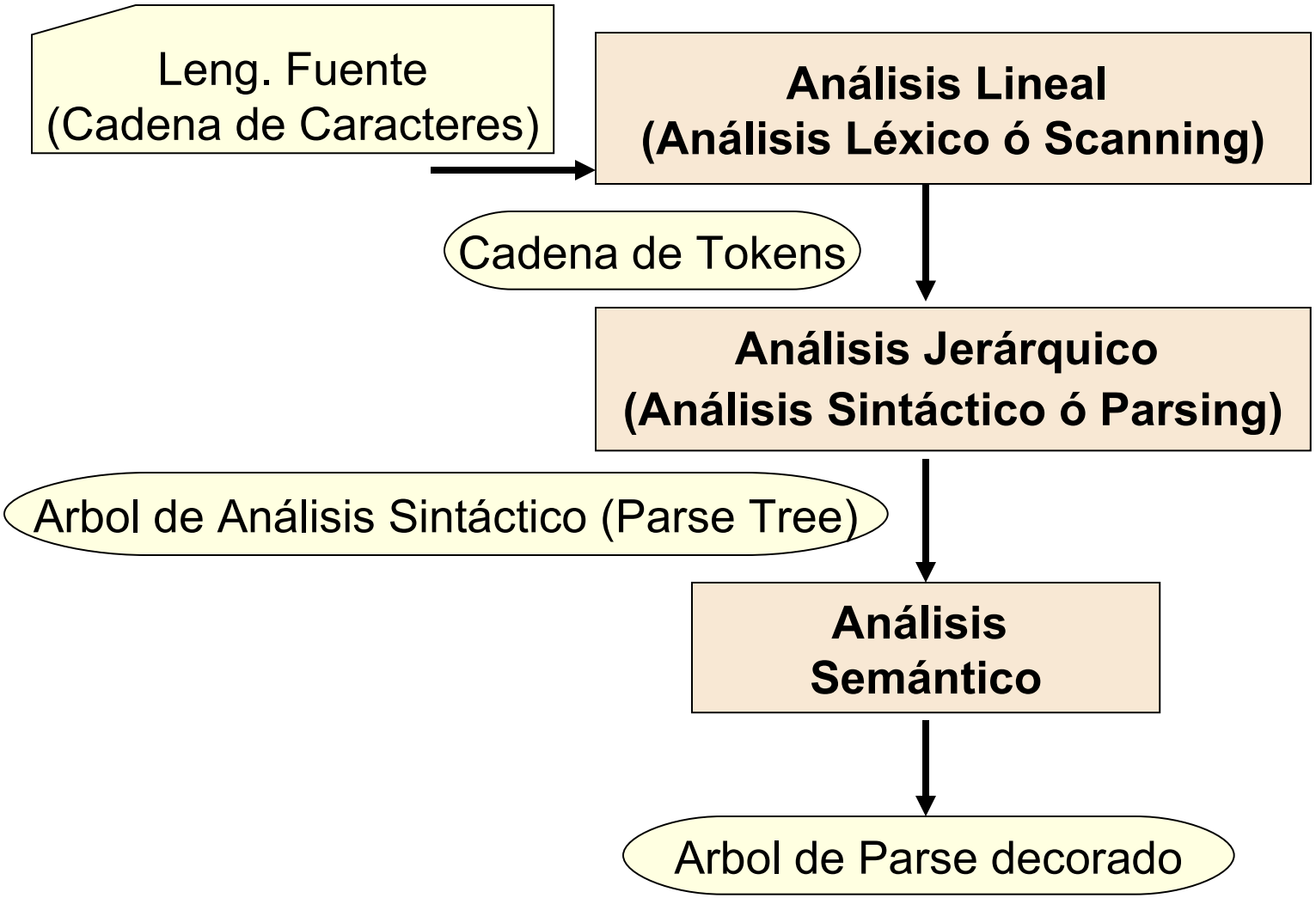


Modelo de Compilación

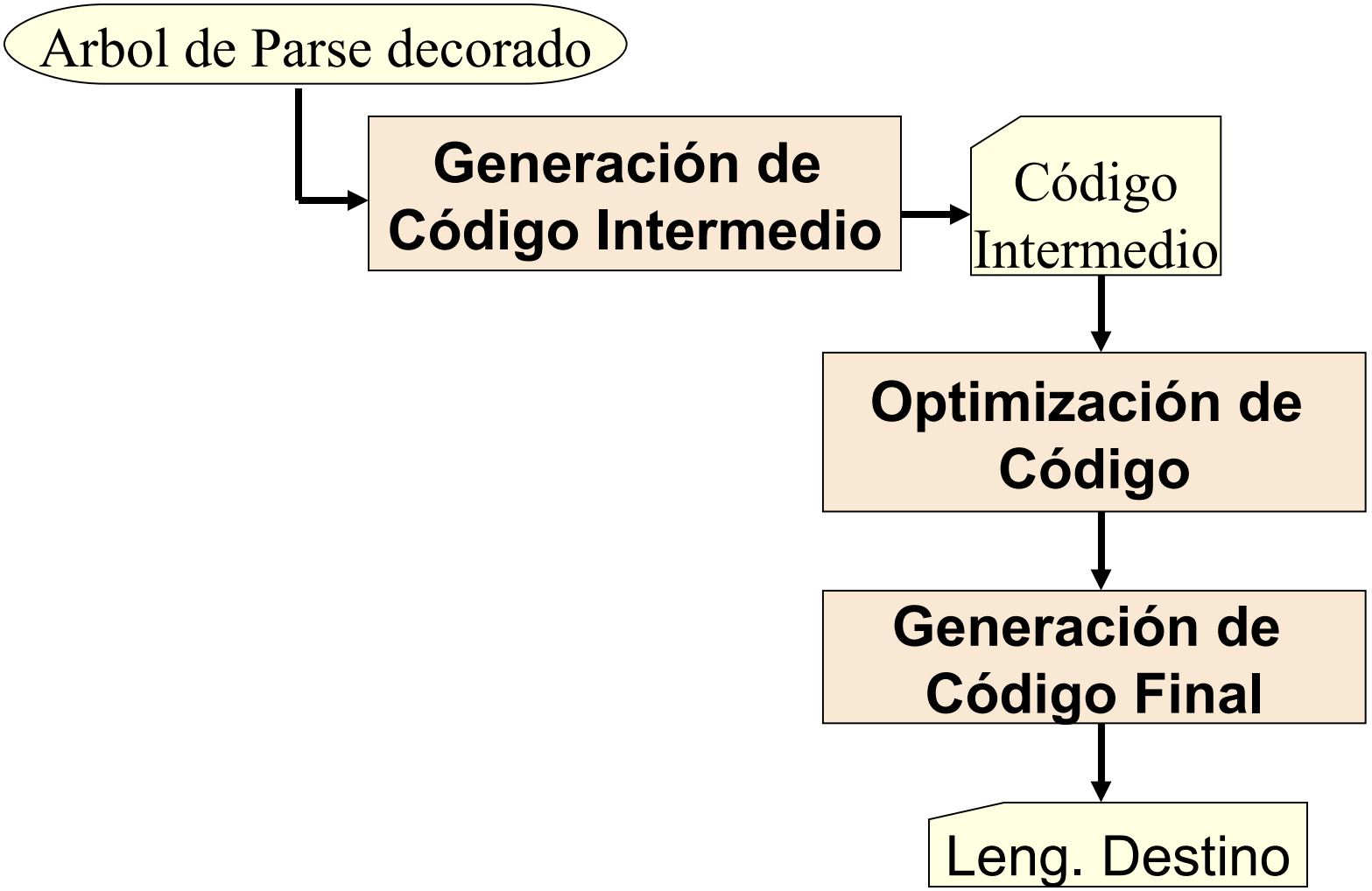
- Análisis: divide el programa fuente en las partes que lo forman y crea una representación intermedia

- Síntesis: construye el programa destino a partir de la representación intermedia

Fases de Análisis



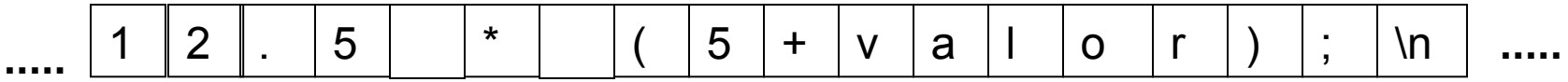
Fases de Síntesis



Además

- Durante todas las fases se usa una Tabla de Símbolos
- Durante las fases de análisis se hace Detección, Reporte y Recuperación de Errores.

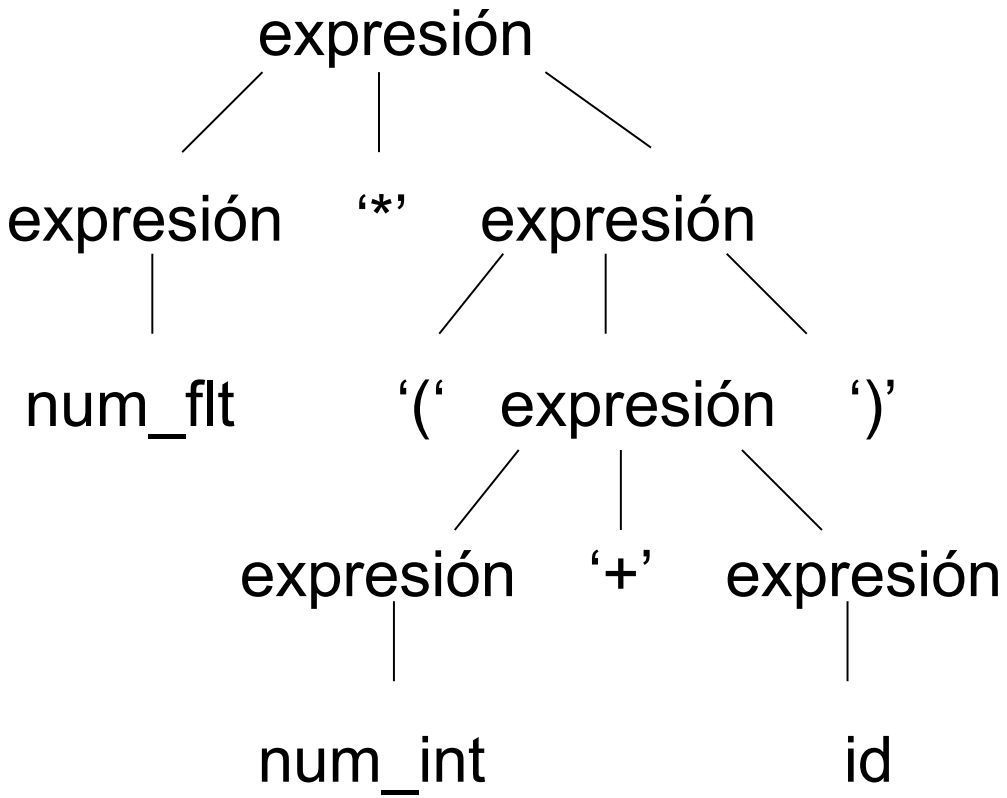
Análisis Léxico (Scanner)



Número flotante (12.5)
Espacio
Operador de multiplicación
Espacio
Paréntesis abierto
Número entero (5)
Operador de suma
Identificador (valor)
Punto y coma
Salto de línea

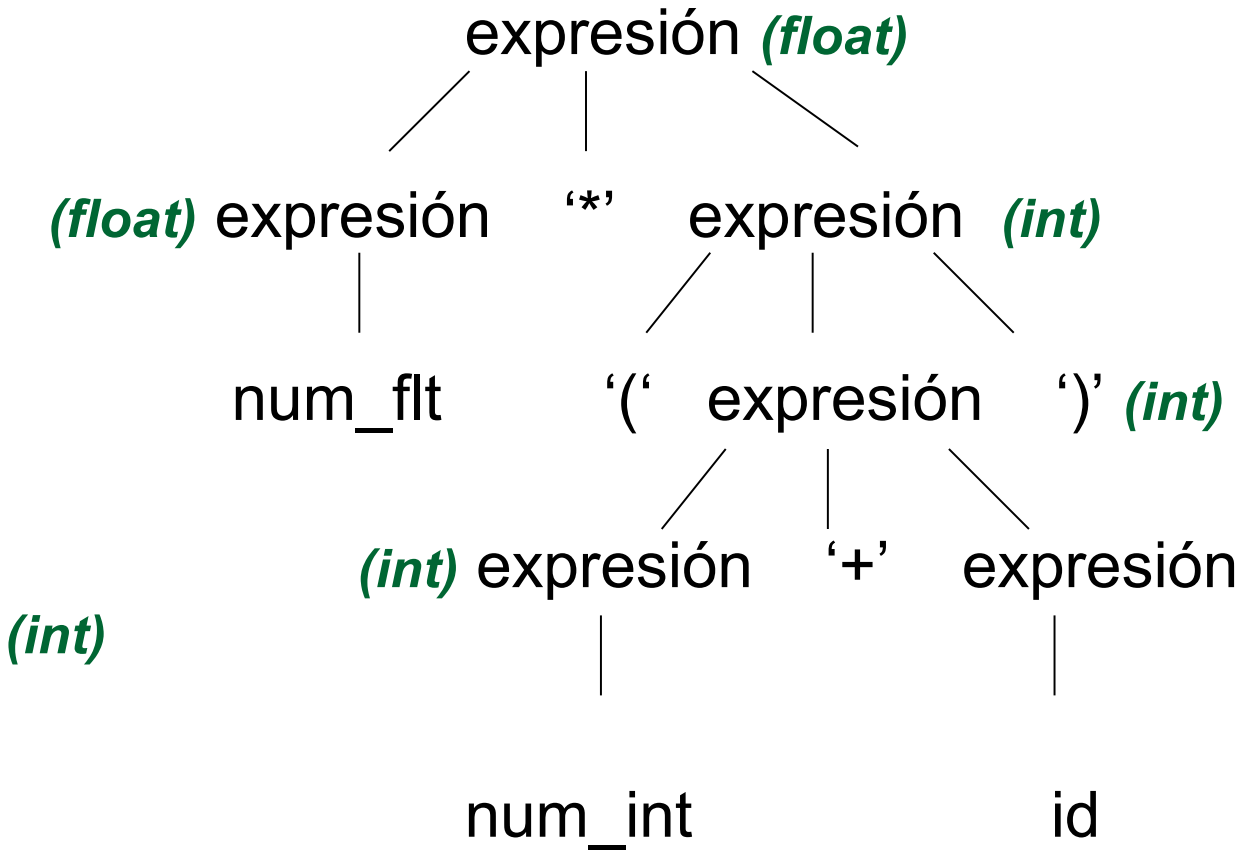
Análisis Sintáctico (Parser)

Num_flt ‘*’ ‘(‘ num_int ‘+’ id ‘)’ ‘;’



Análisis Semántico

Num_flt ‘*’ ‘(‘ num_int ‘+’ id ‘)’



Generación de código intermendio

- Código de una máquina virtual.
- Código de tres direcciones.
  - Cada instrucción consta de sólo tres operandos
  - Sólo hay un operador además de la asignación
- Debe ser un código fácil de generar y fácil de convertir en instrucciones para el procesador.

Optimización de código

- Busca mejorar el código intermedio para:
  - Hacerlo más rápido
  - Lograr que consuma la menor cantidad de memoria posible.
- La optimización puede ser desde trivial hasta muy sofisticada.
- Puede tomar una porción significativa del tiempo total de compilación.

Generación de código

- Se genera código de máquina relocalizable.
- El código generado realiza la misma tarea que el código de entrada.
- Un aspecto clave es la asignación de variables a registros.

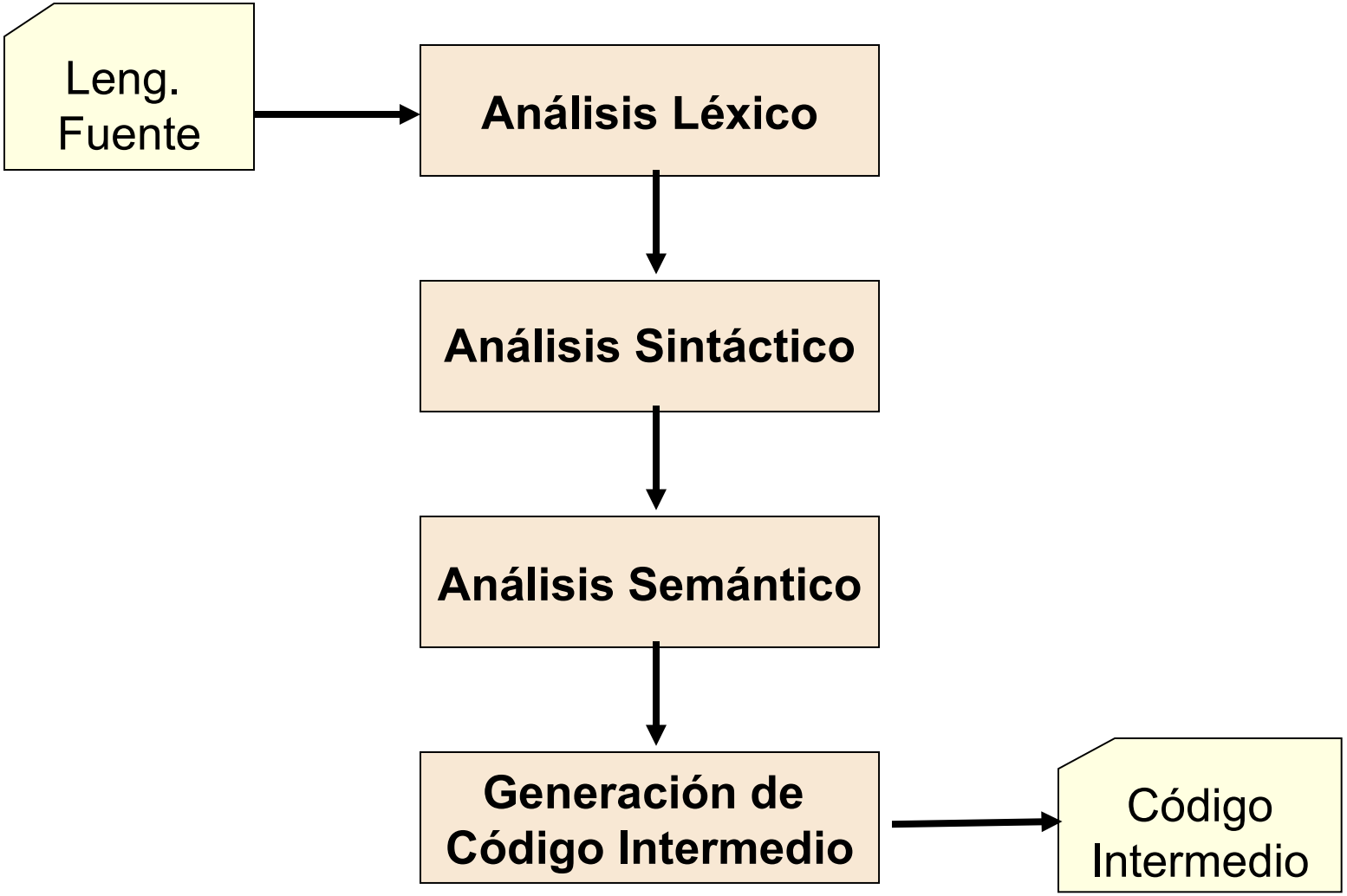
Contexto de la compilación

- Preprocesamiento
  - Resuelve macros
  - Incluye archivos
  - Puede extender el lenguaje con nuevas estructuras de control o de datos
- Ensamble
  - Recibe código en lenguaje ensamblador y genera código listo para ejecutarse.
  - Se puede ensamblar en dos pasadas: primero se construye una tabla de símbolos, luego se genera el código de máquina relocalizable.

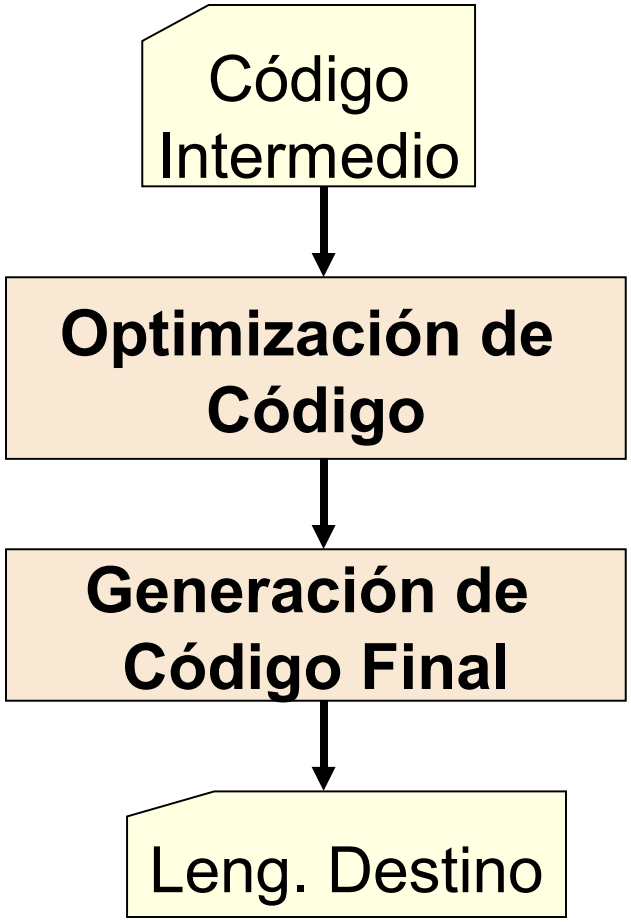
Contexto de la compilación

- Normalmente el mismo programa realiza carga y encadenamiento.
- Carga:
  - Transforma código relocalizable en código absoluto.
- Encadenamiento:
  - Construye un sólo programa a partir de código que está en varios archivos.
  - Resuelve referencias externas.

Front-End


Leng.
Fuente
Análisis Léxico
Análisis Sintáctico
Análisis Semántico
Generación de
Código Intermedio
Código
Intermedio

Back-End


Código Intermedio
Optimización de Código
Generación de Código Final
Leng. Destino

Pasadas

1 Pasada = leer una vez un archivo

Se realizan varias fases con una pasada

Si se tienen pocas pasadas:

- Ventaja: Mayor velocidad de respuesta
- Desventaja: Mayor uso de memoria

Ejemplo: Un Compilador de 4 Pasadas

- 1a. Pasada: Preprocesador
- 2a. Pasada: Front-End
- 3a. Pasada: Back-End (genera código assembler)
- 4a. Pasada: Ensamblador

Herramientas útiles

- Generador de analizadores sintácticos.
- Generador de analizadores léxicos.
- Motor de traducción dirigida por sintaxis.
- Generador automático de código.
- Motor de flujo de datos.