4. PROCESADORES DE LENGUAJES

Existen dos tipos de procesadores de lenguajes o traductores de lenguajes de programación:

- Compiladores.
- Intérpretes.

4.1. COMPILADORES

"Un compilador es un programa que lee un programa escrito en un lenguaje, el lenguaje fuente, y lo traduce a un programa equivalente en otro lenguaje, el lenguaje objeto" (Aho, 1990, p. 1) (ver Figura 4.1). Una función importante del compilador es informar al usuario de la presencia de errores en el programa fuente.

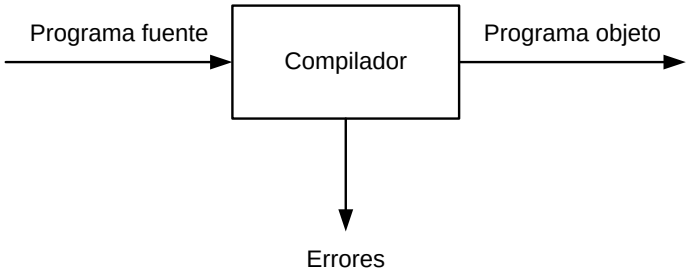

Figura 4.1. Compilador.

Un lenguaje objeto puede ser un lenguaje de programación o el lenguaje de máquina de cualquier computador. Por ejemplo, un desensamblador traduce lenguaje de máquina a lenguaje ensamblador (ver Figura 4.2).

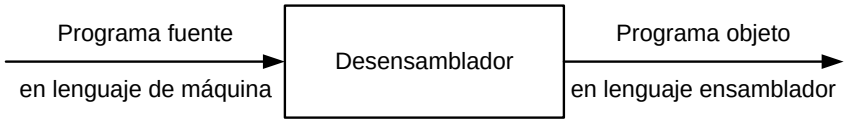

Figura 4.2. Desensamblador.

4.1.1. CONTEXTO DE UN COMPILADOR

Como se muestra en la figura 4.3, para crear un programa objeto ejecutable se pueden necesitar otros programas además de un compilador. Un programa fuente se puede dividir en módulos almacenados en distintos archivos. El **preprocesador** se encarga de la tarea de reunir el programa fuente en un solo archivo y también puede expandir abreviaturas, llamadas macros, a instrucciones del lenguaje fuente.

Después el **compilador** traduce el programa fuente a un programa objeto en lenguaje ensamblador. El lenguaje ensamblador es una versión mnemotécnica del código de máquina, donde se usan nombres en lugar de códigos binarios para las operaciones, y también se usan nombres para las direcciones de memoria.

A continuación, el **ensamblador** traduce el programa objeto en lenguaje ensamblador a código de máquina relocalizable.

El **enlazador** permite formar un solo programa a partir de varios archivos de código de máquina relocalizable. Estos archivos pueden haber sido el resultado de varias compilaciones distintas, y uno o varios de ellos pueden ser archivos de biblioteca (Aho, 1990, p. 19).

Código de máquina relocalizable significa que se puede cargar empezando en cualquier posición L de la memoria; es decir, si se suma L a todas las direcciones del código, entonces todas las referencias serán correctas. La función del **cargador** consiste en tomar el código de máquina relocalizable, modificar las direcciones relocalizables y ubicar las instrucciones modificadas y los datos en las posiciones apropiadas de la memoria para producir el código de máquina absoluto que se ejecuta en el computador.

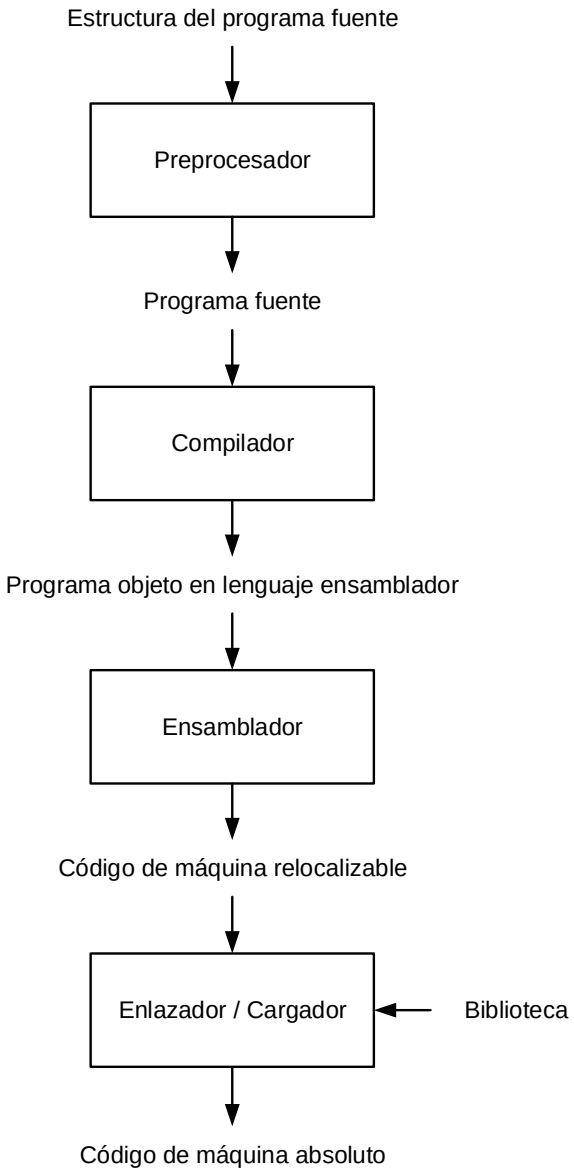

Figura 4.3. Contexto de un compilador.

Por ejemplo, para calcular b = a + 2 (Aho, 1990, p. 19):

Programa fuente

b = a + 2

Programa objeto en lenguaje ensamblador

```
MOV a, R1
ADD #2, R1
MOV R1, b
```

Se reserva una palabra de cuatro bytes para cada identificador y se asignan las direcciones empezando a partir del byte 0 (ver Tabla 4.1)

Tabla 4.1. Tabla de identificadores.

Identificador	Dirección
a	0
b	4

Código de máquina relocalizable

```
0001 01 00 00000000 *
0011 01 10 00000010
0010 01 00 00000100 *
```

- Los cuatro primeros bits son el código de la instrucción, donde 0001, 0010 y 0011 representan las instrucciones LOAD, STORE y ADD, respectivamente. LOAD y STORE significan trasladar de memoria a un registro y viceversa.
- Los dos bits siguientes designan un registro.
- Los dos bits siguientes son el modo de direccionamiento, donde 00 y 10 representan los modos directo e inmediato, respectivamente.
- Los últimos ocho bits hacen referencia al operando.
- El * representa el bit de relocalización que se asocia a cada operando.

Código de máquina absoluto

Si L = 15 = 00001111 entonces

```
0001 01 00 00001111
0011 01 10 00000010
0010 01 00 00010011
```

4.1.2. ESTRUCTURA DE UN COMPILADOR

En la compilación hay dos partes (ver Figura 4.4):

- Análisis (*front-end*).
- Síntesis (*back-end*).

4.1.2.1. Análisis

"La parte del análisis divide al programa fuente en sus elementos componentes y crea una representación intermedia del programa fuente" (Aho, 1990, p. 2).

El análisis consta de tres fases:

- Analizador léxico (*lexer*).
- Analizador sintáctico (*parser*).
- Analizador semántico.

Analizador léxico

El analizador léxico lee los caracteres que componen el programa fuente, los agrupa en secuencias, llamadas lexemas, y produce como salida un componente léxico para cada lexema.

Analizador sintáctico

El analizador sintáctico comprueba si la secuencia de componentes léxicos puede ser generada por la gramática del lenguaje fuente.

Analizador semántico

El analizador semántico verifica si cada operador tiene los operandos permitidos por la especificación del lenguaje fuente.

4.1.2.2. Síntesis

La parte de la síntesis construye el programa objeto a partir de la representación intermedia.

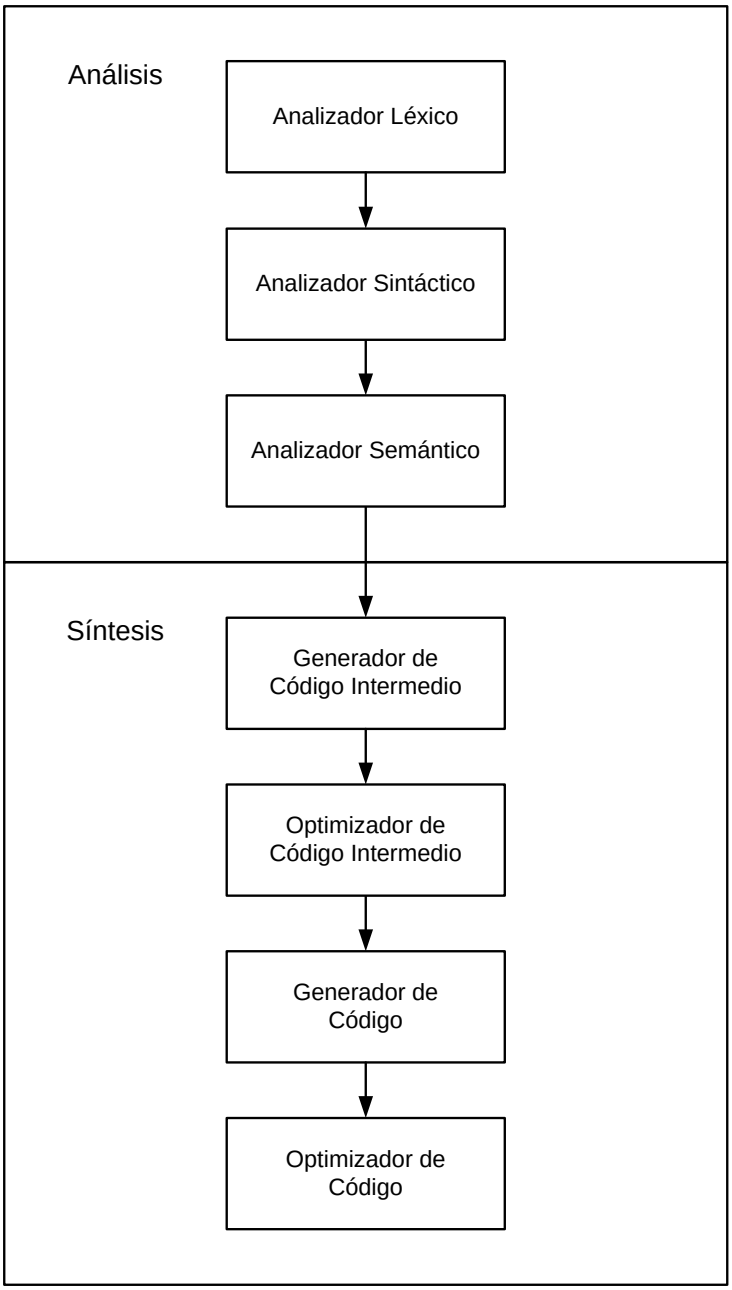

Figura 4.4. Estructura de un compilador.

La síntesis puede constar de hasta cuatro fases:

- Generador de código intermedio.
- Optimizador de código intermedio (opcional).
- Generador de código.
- Optimizador de código (opcional).

Generador de código intermedio

Formas de código intermedio:

- Notación postfija.
- Árboles sintácticos.
- Código de tres direcciones.

Optimizador[4] de código intermedio

El optimizador de código intermedio trata de mejorar[5] el código intermedio con optimizaciones independientes de la máquina.

Generador de código

El generador de código tiene tres tareas principales:

- Selección de instrucciones.
- Asignación de registros.
- Ordenamiento de instrucciones.

Optimizador[6] de código

El optimizador de código trata de mejorar[7] el código con optimizaciones dependientes de la máquina.

[4] [6] El término optimizador no es adecuado porque no hay forma de garantizar que el código resultante sea el mejor posible.

[5] [7] En términos de:
- Velocidad de ejecución.
- Tamaño del código.
- Consumo de energía.

4.1.2.3. Tabla de símbolos

Un identificador es una secuencia de caracteres que hace referencia a una entidad, como un objeto de datos, un procedimiento, una clase o un tipo.

"Una tabla de símbolos es una estructura de datos que contiene un registro por cada identificador, con los campos para los atributos del identificador" (Aho, 1990, p. 11). Estos atributos pueden proporcionar información sobre la memoria asignada a un identificador, su tipo, su ámbito y, en el caso de procedimientos, el número y los tipos de sus argumentos, el método para pasar cada argumento y el tipo que retorna.

La tabla de símbolos se utiliza en todas las fases del compilador.

4.2. INTÉRPRETES

"En lugar de producir un programa objeto como resultado de una traducción, un intérprete realiza las operaciones que implica el programa fuente" (Aho, 1990, p. 3).

4.2.1. VENTAJA

Mejor diagnóstico de errores. Un intérprete permite suspender la ejecución del programa en cierto punto, modificar el programa fuente y continuar la ejecución desde el mismo punto.

4.2.2. DESVENTAJAS

Menor velocidad de ejecución. Cada vez que se ejecuta un programa con un intérprete es necesario realizar la parte del análisis.

Mayor tamaño del programa. El programa ejecutado por un intérprete incluye al intérprete.