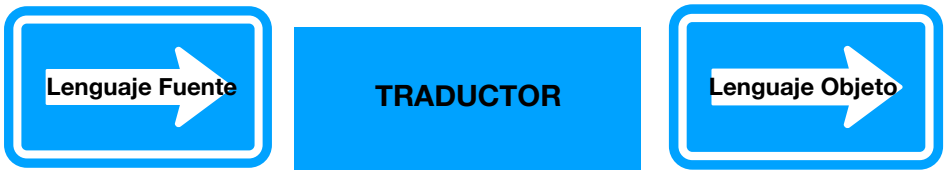
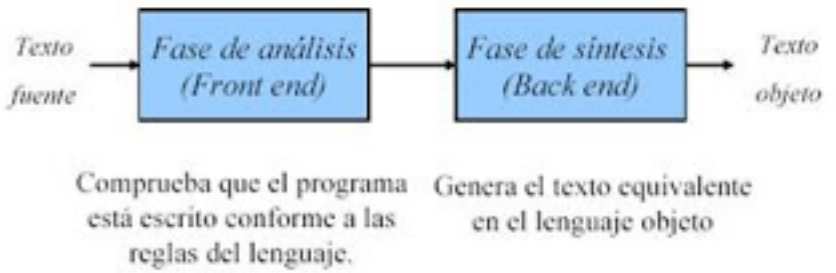
1.4 Estructura de un Traductor

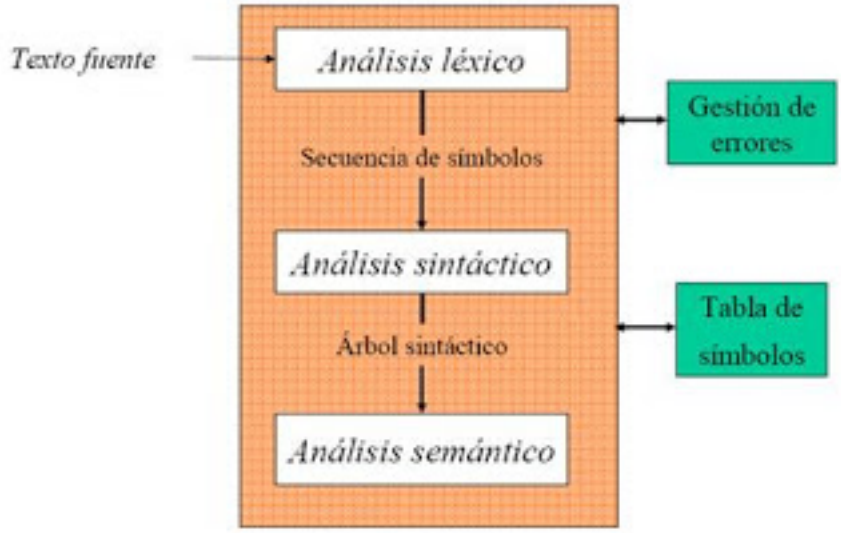
Un traductor es un programa que tiene como entrada un texto escrito en un lenguaje (lenguaje fuente) y como salida produce un texto escrito en un lenguaje (lenguaje objeto) que preserva el significado de origen. Ejemplos de traductores son los ensambladores y los compiladores.



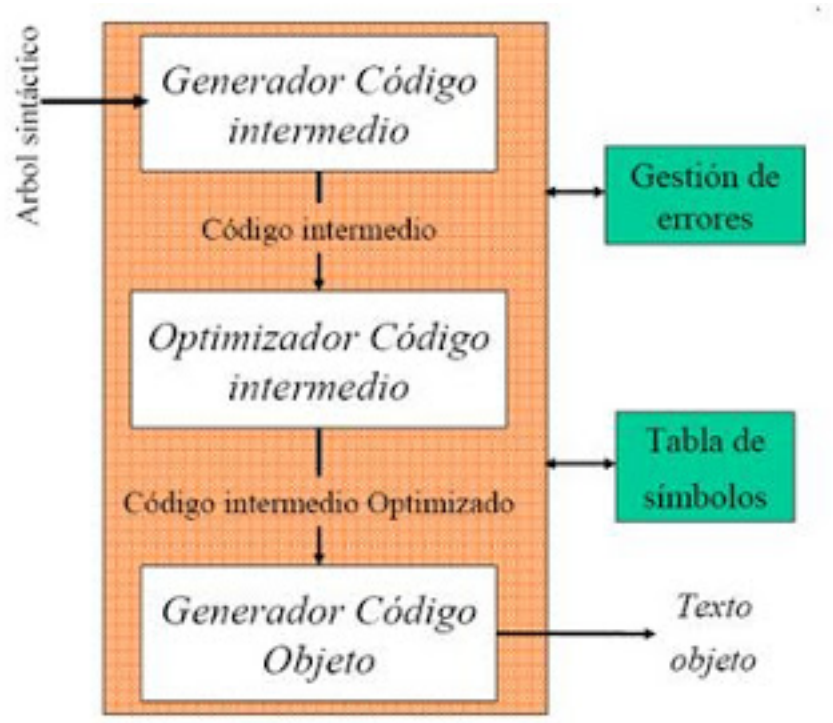
En el proceso de traducción se identifican dos fases principales:



Fase de análisis



Fase de Síntesis



1.4.1 ensambladores.

El programa ensamblador es el programa que realiza la traducción de un programa escrito en ensamblador a lenguaje máquina. Esta traducción es directa e inmediata, ya que las instrucciones en ensamblador no son más que nemotécnicos -nemónicos- de las instrucciones máquina que ejecuta directamente el CPU.

Tipos de ensambladores

Podemos distinguir entre tres tipos de ensambladores:

- **Ensambladores básicos**. Son de muy bajo nivel, y su tarea consiste básicamente en ofrecer nombres simbólicos a las distintas instrucciones.

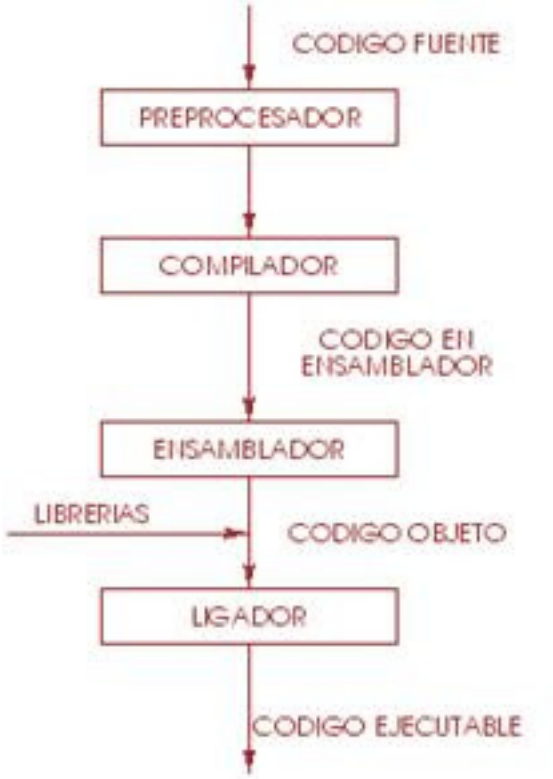
- **Ensambladores modulares**, o macro ensambladores. Descendientes de los ensambladores básicos. Hacen todo lo que puede hacer un ensamblador, y además proporcionan una serie de directivas para definir e invocar macroinstrucciones.

- **Ensambladores modulares 32-bits o de alto nivel**. Son ensambladores que aparecieron como respuesta a una nueva arquitectura de procesadores de 32 bits, realizan la misma tarea que los anteriores, permitiendo también el uso de macros, permiten utilizar estructuras de programación más complejas propias de los lenguajes de alto nivel.

1.4.2 compiladores.

Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, es decir programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior (lenguaje máquina). Generando un programa equivalente a capas del interprete.

Estructura de un Compilador.



Cualquier compilador debe realizar dos tareas principales: análisis del programa a compilar y síntesis de un programa en lenguaje máquina. Para el estudio de un compilador, es necesario dividir su trabajo en fases. Cada fase representa una transformación al código fuente para obtener el código objeto. En cada una de las fases se utiliza un administrador de la tabla de símbolos y un manejador de errores.

Componentes en que se divide un compilador:

Análisis Léxico. En esta fase se lee los caracteres del programa fuente y se agrupan en cadenas que representan los componentes léxicos. A la secuencia de caracteres que representa un componente léxico se le llama lexema (o con su nombre en inglés token).

Análisis Sintáctico. Los componentes léxicos se agrupan en frases gramaticales que el compilador utiliza para sintetizar la salida.

Análisis Semántico. Intenta detectar instrucciones que tengan la estructura sintáctica correcta, pero que no tengan significado para la operación implicada.

Generación de código Intermedio. Se puede considerar esta operación intermedia como un subprograma para una máquina abstracta, esta representación debe tener dos propiedades importantes: debe ser fácil de producir y fácil de traducir al programa objeto.

Optimización de Código. Se trata de mejorar el código intermedio, de modo que resulte un código de máquina más rápido de ejecutar.

Generación de Código. Esta constituye la fase final de un compilador.
Administrador de la tabla de símbolos. Se encarga de manejar los accesos a la tabla de símbolos, en cada una de las etapas de compilación de un programa.

Manejador de errores. Es posible encontrar errores. De esta forma podrán controlarse más eficientemente los errores encontrados en cada una de las fases de la compilación de un programa.

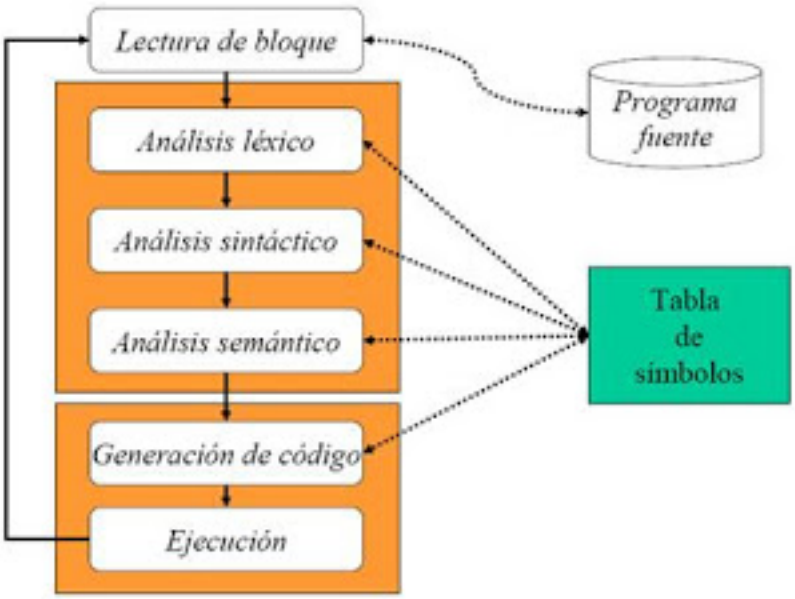
1.4.3 interpretes.

Los intérpretes realizan normalmente dos operaciones:

- Traducen el código fuente a un formato interno.
- Ejecutan o interpretan el programa traducido al formato interno.

La primera parte del intérprete se llama a veces "el compilador", aunque el código interno que genera no es el lenguaje de la máquina, ni siquiera lenguaje simbólico, ni tampoco un lenguaje de alto nivel.

Estructura



Particularidades de la interpretación:

- Ahorra memoria.
- Produce un resultado que no se puede almacenar, lo cual hace la ejecución lenta.
- No es demasiado eficiente, cada vez que se entra en un bucle se analizaran sus sentencias.
- Facilita el proceso de depuración.
- No produce resultados transportables.

La interpretación es útil en:

- Sistemas interactivos.
- Programas de pequeña envergadura.
- Programas de prototipo y de enseñanza.