El Proceso de Compilación

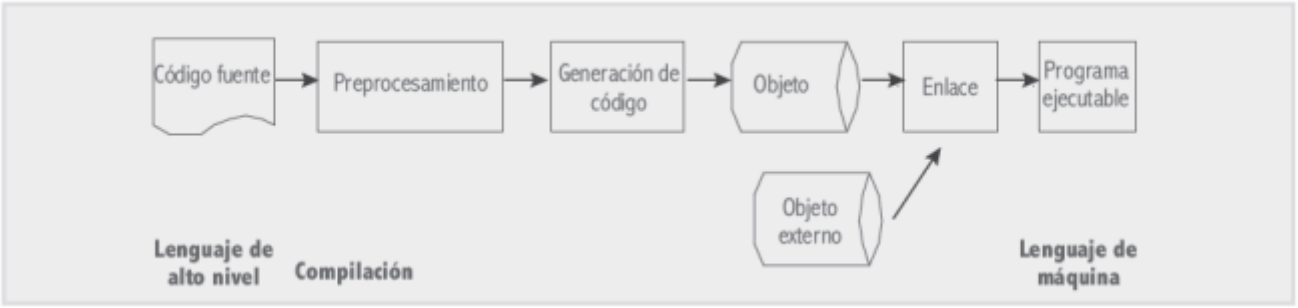
Introducción

Vamos analizar por separado las etapas más importantes que comprenden el proceso de compilación: preprocesamiento, generación de código y enlace :)

Proceso de Compilación

En el ámbito de las computadoras, los algoritmos se expresan mediante lenguajes de programación, como C, Python, C#, Java, Ensamblador, etc. Sin embargo, esta representación no es suficiente, ya que el procesador necesita una expresión mucho más detallada del algoritmo, que especifique en forma explícita todas las señales eléctricas que involucra cada operación.

La tarea de traducción de un programa desde un lenguaje de programación (ya sea alto o bajo nivel) hasta el lenguaje de máquina se denomina compilación, y la herramienta encargada de ello es el compilador. En la figura siguiente se pueden distinguir las etapas más importantes:



Preprocesamiento

La primera etapa del proceso de compilación se conoce como preprocesamiento. En ocasiones a esta etapa ni siquiera se la considera parte de la compilación, ya que es una traducción previa y básica que tiene como finalidad “acomodar” el código fuente antes de que éste sea procesado por el compilador en sí.

El preprocesador modifica el código fuente según las directivas que haya escrito el programador. Por ejemplo en lenguaje C, estas directivas se reconocen en el código fuente porque comienzan con el carácter #, (por ejemplo, #define... o #include...):

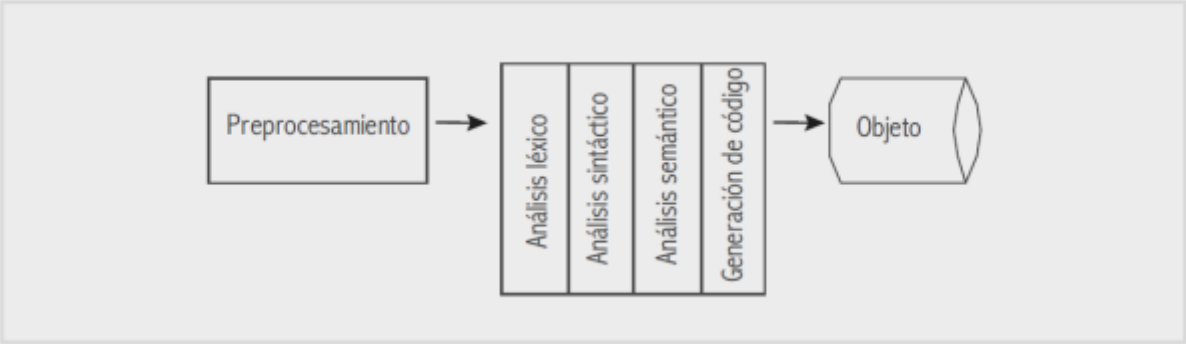
```
   #define, #undef, #error, #include
```

Compilación

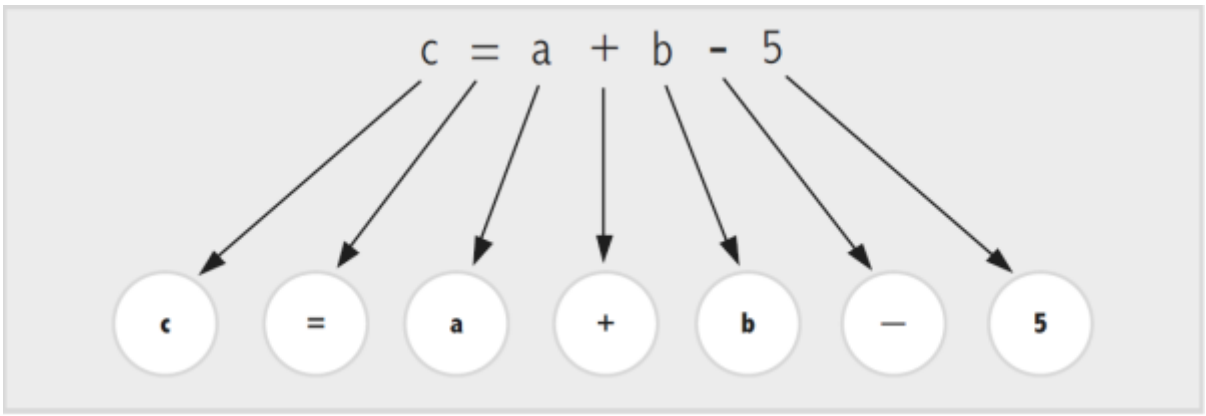
La compilación es el proceso que, en sí, traduce el lenguaje de programación (ya sea alto o bajo nivel)

en lenguaje de máquina. Dentro de esta etapa pueden reconocerse, al menos, cuatro fases:

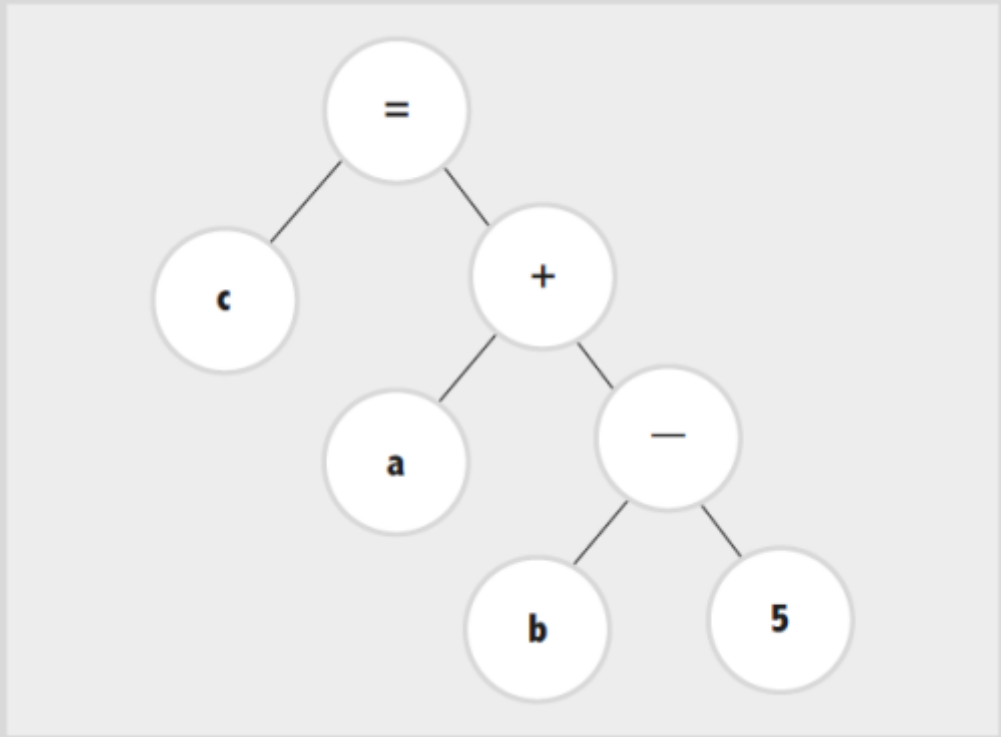
- Análisis léxico.
- Análisis sintáctico.
- Análisis semántico.
- Generación de código.



El `análisis léxico` extrae del archivo fuente todas las cadenas de caracteres que reconoce como parte del vocabulario y genera un conjunto de tokens como salida. En caso de que parte del archivo de entrada no pueda reconocerse como lenguaje válido, se generarán los mensajes de error correspondientes.



A continuación, durante el `análisis sintáctico`, se procesa la secuencia de tokens generada con anterioridad, y se construye una representación intermedia, que aún no es lenguaje de máquina, pero que le permitirá al compilador realizar su labor con más facilidad en las fases sucesivas. Esta representación suele ser un árbol.



Durante el `análisis semántico` se utiliza el árbol generado en la fase previa para detectar posibles violaciones a la semántica del lenguaje de programación, como podría ser la declaración y el uso consistente de identificadores (por ejemplo, que el tipo de dato en el lado derecho de una asignación sea acorde con el tipo de dato de la variable destino, en el lado izquierdo de la asignación).

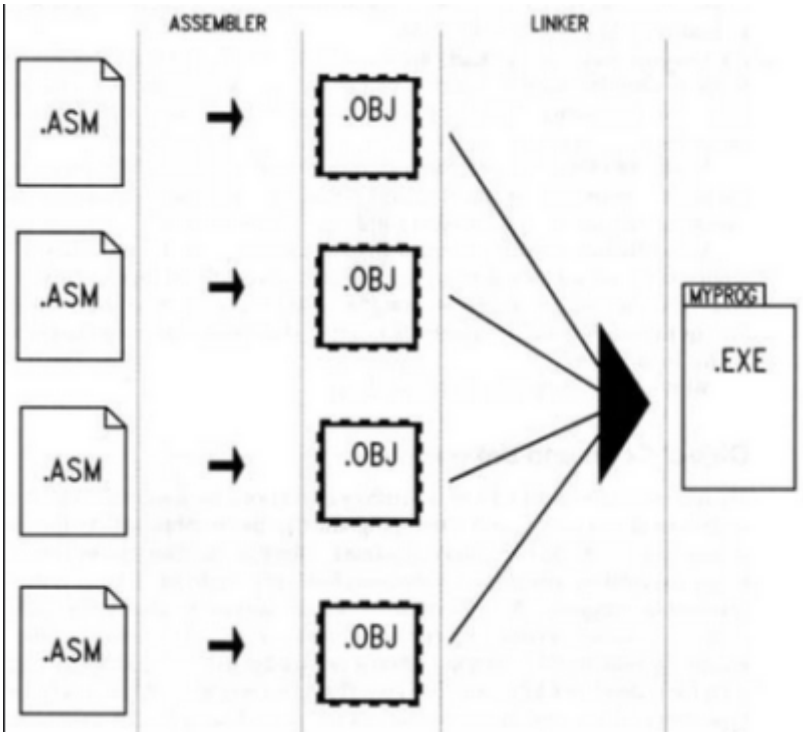
Por último, en la generación de código se transforma la representación intermedia en lenguaje de máquina (código objeto). En los casos típicos esta fase involucra mucho trabajo relacionado con la optimización del código, antes de generarse el lenguaje de máquina.

En el caso de un lenguaje ensamblador, el compilador se llama ensamblador y el codigo obtenido se llama código máquina y puede ser procesado directamente por el procesador.

Enlace

No siempre las aplicaciones se construyen de manera monolítica, a partir de un solo archivo fuente. En la práctica sólo se escribe una parte, y lo demás se toma de bibliotecas externas que, en la última etapa de la compilación, se enlazarán unas con otras para generar la aplicación ejecutable final. Ésta es, básicamente, la tarea del `enlazador`, también conocido como `linker`.

Los archivos objeto que se enlazan a nuestro programa se denominan bibliotecas externas que, por su parte, pueden haber sido construidas por nosotros mismos o pueden provenir de terceras partes (por ejemplo, las bibliotecas estándares del compilador). Una biblioteca, en este contexto, es una colección de funciones y en el proceso de enlace se añade al código objeto el código de la función a la que se hizo referencia. Si tenemos otros códigos objetos el proceso de enlace los combina para obtener un único ejecutable, como se ilustra en la siguiente figura:



— *Mariano Vargas*

Volver

(230)