**Capítulo 5**

La compilación es el proceso de traducción de un programa escrito en lenguaje de alto nivel a otro escrito en lenguaje ensamblador. El ensamblado es el proceso de traducción del programa escrito en lenguaje ensamblador a otro escrito en lenguaje de máquina. El enlace (linking) es el proceso de unificación de distintos módulos que fueron ensamblados en forma separada en un solo programa. La carga es el proceso de traslado del programa a memoria y su preparación para ser ejecutado.

La conversión de una sentencia de un lenguaje de alto nivel o una o más sentencias de lenguaje simbólico enfrenta al compilador con una cantidad de tareas relativamente complejas:

1. Análisis lexicográfico: reconocer dentro del texto del programa los símbolos básicos del lenguaje: letras, números y símbolos.
2. Análisis sintáctico: analizar los símbolos para reconocer la estructura de programación subyacente.
3. Análisis semántico: asociar las variables con los nombres que les fueron dados y posteriormente, con posiciones particulares de memoria para su almacenamiento. Además, determinar el tipo de todos los datos requeridos.
4. Asignación de acciones y generación de código: asociar las sentencias de programa con la secuencia apropiada del lenguaje ensamblador.
5. Además, el compilador asigna variables a registros, controla el uso de registros, y potencialmente, optimiza el programa.

Cuando se desarrolla el compilador, se debe incluir en su estructura la información acerca de la arquitectura de programación particular del procesador para el cual se lo desarrolla. Esta inclusión suele llamarse especificación de asignación del compilador. Por ejemplo, se debe decidir cómo ubicar los distintos tipos de variables y constantes dentro de los recursos de la máquina. Esto puede ser función tanto de la máquina como del lenguaje de alto nivel. En C, los enteros pueden ser de 16 bits, de 32 bits o de algún otro tamaño. En ARC los enteros serán convertidos en palabras de 32 bits.

Además, el autor del compilador debe tener en cuenta las características y limitaciones de la máquina cuando convierte construcciones desde el lenguaje de alto nivel. Por ejemplo, ARC requiere que todos los operandos aritméticos sean constantes inmediatas o variables asignadas a registros. Por lo tanto, el compilador debe generar el código necesario para acomodar todas las variables en registros antes de poder ejecutar cualquier instrucción aritmética.

Las variables globales tienen direcciones conocidas en el momento de compilación. Por el otro lado, las variables locales son las que son declaradas dentro de funciones o dentro de bloques y solo toman existencia cuando se ingresa en dicho bloque o función, desapareciendo cuando la función se invoca por última vez. Estas variables locales se almacenan en una pila. Las variables allí almacenadas cobran valor cuando se crea la pila y se invoca la función y desaparece cuando se sale por última vez de dicha función.

El proceso de ensamblado es líneal y relativamente simple, dado que hay una relación directa y univoca entre las sentencias del lenguaje simbólico y sus correspondientes expresiones en el lenguaje de máquina. Un ensamblador debe:

1. Permitir que el programador especifique la ubicación de las variables y programas al momento de la ejecución.
2. Ofrecer al programador la posibilidad de inicializar los valores de los datos en memoria antes de que se ejecute el programa.
3. Proveer expresiones para todas las instrucciones del lenguaje de máquina y modos de direccionamiento y traducir las sentencias válidas del lenguaje simbólico hacia sus valores binarios.
4. Permitir el uso de etiquetas para identificar o representar direcciones y constantes.
5. Ofrecer al programador una forma de especificar la dirección de comienzo de un programa.
6. Ofrecer cierto grado de cálculo al momento del ensamblado.
7. Permitir el uso de variables en un programa que hayan sido definidas y ensambladas en otro programa distinto.
8. Proveer la expansión de macro rutinas.

La mayoría de los ensambladores recorren dos veces el texto escrito en lenguaje simbólico (ensambladores de dos pasadas). **[En la primera, el ensamblador se dedica a determinar las direcciones de todos los datos e instrucciones del programa y a seleccionar qué instrucción del lenguaje de máquina debe generarse para cada instrucción del lenguaje simbólico.]** La directiva .org se usa para asignarle un valor al contador de posición, a partir del cual asigna direcciones a los datos. Además, durante esta pasada, se realizan las operaciones aritméticas y se insertan las definiciones de todas las etiquetas y constantes en la tabla de símbolos.

La razón principal para requerir una segunda pasada tiene que ver con la necesidad de permitir el uso de los símbolos dentro de un programa antes de que sean definidos. Después de la primera pasada, todos los símbolos estarán definidos e ingresados en la tabla, por lo que el ensamblador podrá generar el código de máquina con los valores de los símbolos conocidos. Genera dos archivos: el código objeto (ejecutable .exe) y el listado (información para el usuario, logs).

Luego de haber traducido, el ensamblador le agrega al módulo traducido la siguiente información:

* El nombre y tamaño del modulo.
* La dirección del símbolo de comienzo, si es que se define alguno en el módulo.
* Información acerca de símbolos globales y externos.
* Información acerca de las rutinas de biblioteca a las que el módulo hace referencia.
* Los valores de cualquier constante que deba cargarse en memoria.
* Información de reubicación: el ensamblador especifica cuales direcciones pueden reubicarse y cuáles no.

El editor de enlace o linker es un programa que combina programas ensamblados por separado, conocidos como módulos objeto, en un único programa o módulo de carga. El linker resuelve todas las referencias globales y externas y reubica las direcciones de los diferentes módulos. El módulo de carga puede ser cargado en memoria por medio de un cargador que también puede necesitar modificar direcciones si el programa se carga en una dirección distinta a la dirección de origen de carga usada por el linker.

DLL o bibliotecas de enlace dinámico es una técnica que pospone el enlace de algunos componentes hasta que sean requeridos efectivamente durante el momento de la ejecución.

Al combinar los módulos ensamblados por separado, el linker debe:

* Resolver referencias de direcciones externas a los módulos a medida que los vincula.
* Reubicar cada módulo por medio de la combinación más apropiada.
* Especificar el símbolo de comienzo del módulo de carga.
* Especificar los contenidos e identidades de los diversos segmentos de memoria (si los hay).

La directiva .global indica al ensamblador que debe señalizar al símbolo como disponible para otros módulos objeto durante la fase de enlace. Se la utiliza en el módulo en que se define un símbolo.

La directiva .extern identifica a un rótulo usado en un módulo pero que se encuentra definido en otro. Se la utiliza en cada uno de los módulos que haga referencia a aquel en el que fue definido.

El ensamblador define como reubicables a aquellos símbolos que pueden admitir que sus direcciones sean modificadas durante el proceso de enlace. La reubicación se realiza a través del linker. El ensamblador tiene la responsabilidad de determinar que rótulos son reubicables cuando construye la tabla de símbolos.

**El cargador (loader) es un programa que ubica el módulo de carga en la memoria principal. Carga los diversos segmentos de memoria con los valores apropiados e inicializa ciertos registros como el puntero de pila y el contador de programa.** En un sistema operativo existen varios programas que residen en memoria, por lo que no hay forma de que el ensamblador o el linker sepan en qué dirección residirán dichos programas. **El loader reubica esos módulos en el momento de la carga. Esto se conoce como cargador reubicador. El loader simplemente modifica las direcciones reubicables que encuentra dentro de un módulo de carga para que coexistan varios programas al mismo tiempo.**

Un programa cargador y de enlace (linking loader) ejecuta tanto el proceso de enlace como el de carga.

Las rutinas usadas habitualmente, como los paquetes de software para manejo de memoria o gráficos, solo deben estar presentes en un lugar: la biblioteca DLL. El resultado de esto es un programa de menor tamaño, dado que no hace falta que cada programa posea su propia copia del código DLL**. La actualización de las bibliotecas DLL para incorporar mejoras en sus prestaciones o eliminar errores requiere que las correspondientes modificaciones sean incorporadas en un solo punto, con lo que lo programas que las usan no necesitan ser recompilados en un paso separado.** Ahora bien, esas mismas prestaciones se pueden convertir en desventajas porque la conducta del programa puede modificarse en forma no deseada (por ejemplo cuando un programa se queda sin memoria por el uso de una biblioteca DLL de mayor tamaño). Las bibliotecas DLL deben estar presentes en todo momento y tienen que contener la versión esperada por cada programa.

Una macro es una instrucción que contiene a dos o más instrucciones. Se usa para simplificar el código. El proceso de traducir una macro a su equivalente en lenguaje simbólico se conoce como expansión de la macro. En el cuerpo de la macro pueden aparecer otros comandos, como otras macros o incluso llamados a la misma macro, lo que permite que se expanda en forma recursiva durante el momento del ensamblado.

En el caso de un ensamblador que soporte macros, debe existir una etapa de expansión de las mismas, que debe llevarse a cabo antes de las dos vueltas del proceso de ensamblado. Se realiza por medio de un preprocesador de macroinstrucciones antes del ensamblado del programa. Este proceso puede ser invisible al programador, dado que puede ser invocado por el ensamblador.

La expansión de una macro requiere dos pasos:

1. Registrar las definiciones de las mismas.
2. Generar las sentencias en el lenguaje simbólico (en el caso de aceptar definiciones recursivas, este paso puede complicarse mucho).