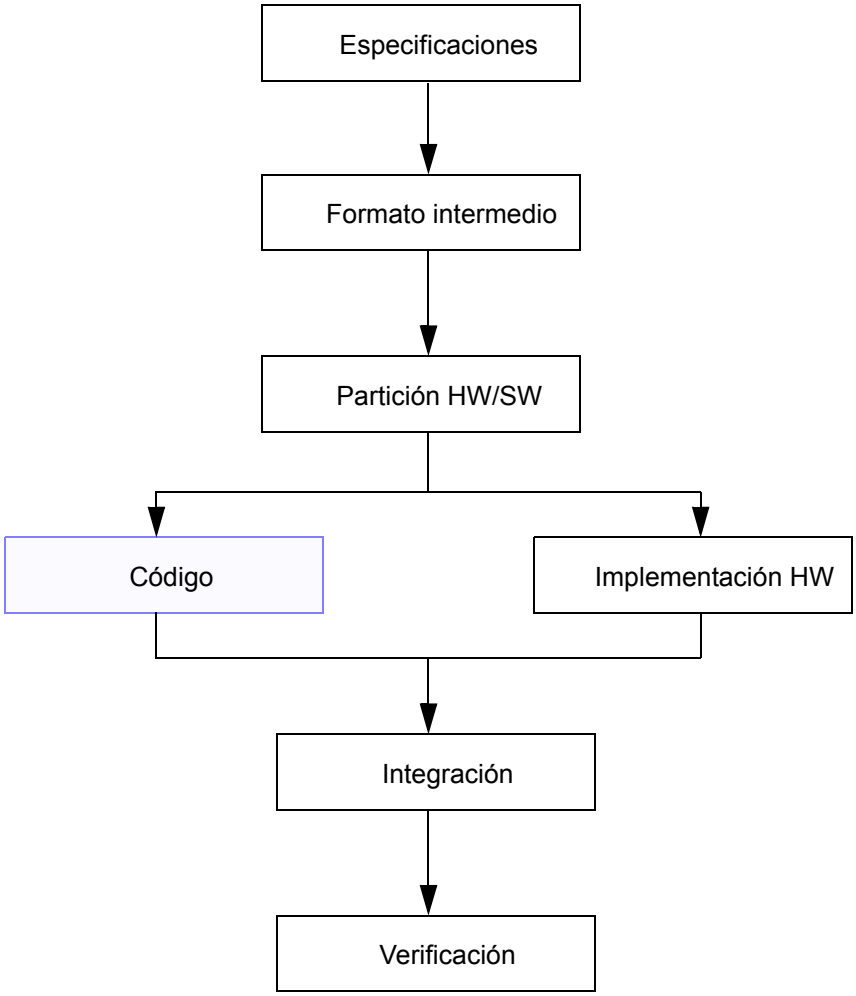
Tema V

Generación de Código

Una vez que se ha realizado la partición HW/SW y conocemos las operaciones que se van a implementar por hardware y software, debemos abordar el proceso de estas implementaciones. Seguiremos con la implementación software. Este tema ya ha sido ampliamente tratado a lo largo de diferentes asignaturas, así que no se hará mucho hincapíe.



5.1. Introducción

En la siguiente etapa del diseño debemos generar el código de las operaciones que van a ser implementadas vía software, para lo cual se debe mapear las especificaciones de alto nivel a los recursos disponibles en el procesador (o arquitectura software del sistema).

A la hora de abordar esta tarea es necesario tener en cuenta dos aspectos fundamentales:

- Arquitecturas del sistema software
  - Ya tenga un conjunto de instrucciones complejo o reducido.
  - El sistema esté dividido en etapas de pipeline o no.
  - El esquema de memoria, tengan un tratamiento idéntico las instrucciones y datos o no.
  - El tipo de procesamiento en paralelo o no
- Criterios de optimización
  - Rendimiento
  - Tamaño
  - Consumo
  - Utilización de recursos

Estos aspectos nos servirán para poder minimizar el código de las operaciones, y así adecuarlo a las características del sistema global. En concreto, la arquitectura software nos va a servir para utilizar los aspectos que nos lleven a obtener unos criterios de optimización deseados.

Como se trata de un programa software, el flujo de diseño será el mismo que el de cualquier programa, mostrado en la figura 5.1. En este flujo el código inicial debe pasar por una serie de procesos antes de obtener el código ejecutable almacenado en memoria y listo para su ejecución. Estos procesos son la compilación, el ensamblaje, el linkado y el cargado. Todos ellos pueden ser realizados de forma automática sin ninguna intervención del programador (a excepción de introducir el código fuente); no obstante esta solución implica un tiempo de diseño muy bajo a costa de una optimización muy pobre. Lo que se suele hacer es utilizar unos compiladores apropiados para los requerimientos de sistemas empotrados y realizar el resto de procesos de forma automática, por ello hablaremos de la compilación en último lugar.

La tarea del ensamblador consiste en pasar del código de ensamblador (generalmente a base de instrucciones de bajo nivel expresadas como nemotécnicos) a un código objecto que pueda ser interpretable por el sistema software en el que se vaya a ejecutar. Por lo tanto, el ensamblador deberá realizar los siguientes procesos:

- Expansión de abreviaturas y macros
- Resolución de etiquetas locales
- Tabla de reubicación
- Obtención de la salida en el formato máquina del sistema software destino

La tarea del linkador consiste en pasar del formato máquina a un código ejecutable en el sistema software destino. Por lo tanto, deberá realizar las siguientes operaciones:

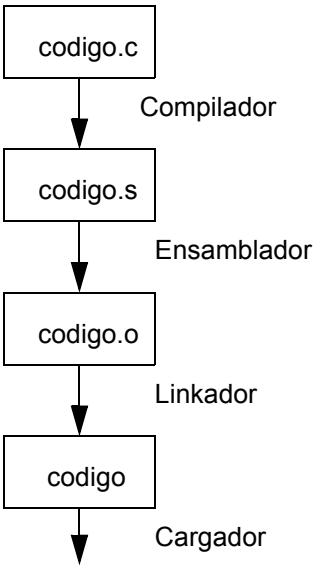


Figura 5.1.- Flujo de diseño de un programa software.

- Leer todos los objecto y librerías utilizadas.
- Combina las diferentes secciones (de una forma inteligente).
- Resuelve los símbolos globales.
- Como salida proporciona un código ejecutable con una nueva tabla de reubicación.

La tarea del cargador consiste en llevar el código ejecutable a la memoria principal del sistema software para su ejecución. Para ello deberá realizar los siguientes procesos:

- Copia el código ejecutable suministrado por el linkador a memoria.
- Reserva espacio para la pila.
- Reubica las direcciones en función de la tabla.
- Pasa el control al código de startup.

El compilador, a pesar de ser el primero en realizar sus tareas, lo hemos dejado para el final ya que es el elemento en el que se pueden realizar las optimizaciones. La tarea del compilador consiste en pasar una secuencia de instrucciones (generalmente de un lenguaje de alto nivel como puede ser el C) a una descripción en lenguaje ensamblador. Para ello debe realizar los siguientes procesos:

- Analizador léxico,
- Parser,
- Analizador semántico,
- Traductor,
- Optimizador,

Estos elementos estarán presentes cuando la compilación se realiza a bajo nivel. No obstante, al ir implementándose compiladores dedicados a estos sistemas, también se puede abordar la

compilación desde un nivel mixto, es decir, todavía no desde un nivel de abstracción alto, pero no tan bajo como el anterior. En este útlimo caso, las dos últimas tareas son sustituidas por las siguientes:

- Generador de código intermedio,
- Optimizador,
- Generador de código,

como se puede ver en la figura 5.2.

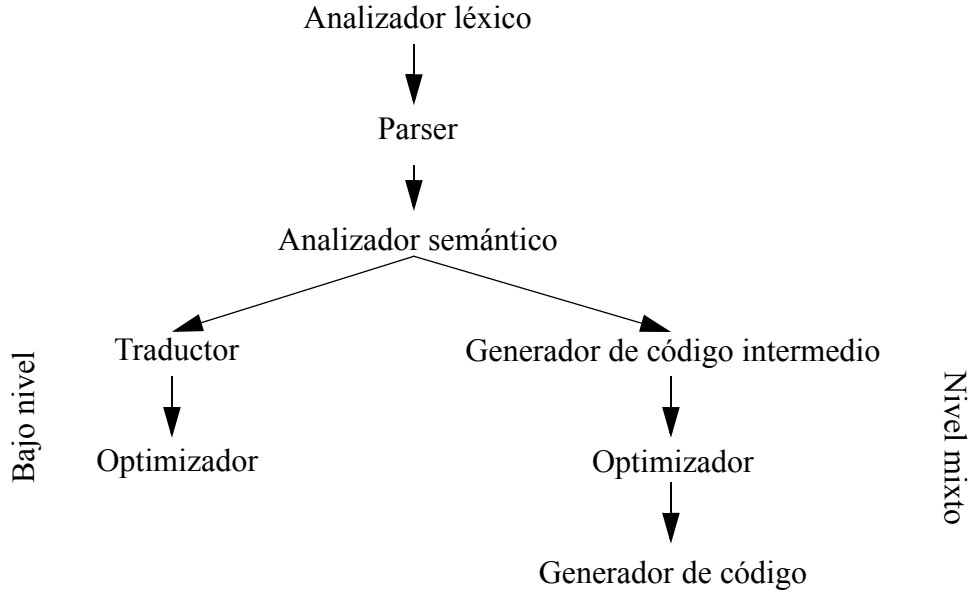


Figura 5.2.- Pasos del compilador

5.2. Análisis

Debido a que a lo largo de la carrera ya habéis cursado suficientes asignaturas relativas a la programación, aquí vamos a indicar una serie de directrices para aumentar las prestaciones del código utilizado, a través de las siguientes actuaciones:

- reducción del consumo de potencia
- aumento de velocidad
- reducción del área de memoria

5.2.1. Análisis de prestaciones

La generación del código provocará diferentes consumos de potencia según los procesos y accesos a memoria que estén programados realizarse. El consumo se puede dividir en diferentes partes del sistema software:

- Datapaths de la ALU y las unidades funcionales

- Cache y sistemas de memoria
- Buses del sistema
- Circuitos de control y distribución más lógica de control.

En cambio, el tiempo de ejecución se puede dividir en dos contribuciones bien diferenciadas:

- Instrucciones existentes en el código
- Número de accesos a memoria

Por último, el área de memoria dependerá del número y tipo de variables que se hayan declarado en el código.

Suponiendo que el algoritmo implementado es el más óptimo, vamos a dar una serie de directrices que sirven para reducir el consumo de potencia y aumentar las prestaciones en el sistema software, mostrados en la figura 5.3:

- Permutaciones de bucles (figura 5.3a), para mejorar la localidad al acceder a los elementos del array de forma secuencial. De esta forma todos los datos del array (sino la mayoría) estarán localizados en la memoria cache.
- Reducción del tamaño de la ventana de los bucles (figura 5.3b), de esta forma se reduce el número de iteraciones de cada bucle.
- Fusión de bucles (figura 5.3c), reduciendo el espacio de memoria que se recorre eliminando bucles innecesarios.
- Desdoblado de bucles (figura 5.3d), de esta forma reducimos el número de iteraciones que deben realizar los bucles.
- Reindexar la dimensión (figura 5.3e), de esta forma se mejora la localidad igual que en el primer caso.
- Minimizar los accesos a memoria (figura 5.3f), y así reducimos las instrucciones que se deban ejecutar en cada bucle a las necesarias.
- Reducción de área en memoria (figura 5.3g), y spuestamente también se reducen los accesos a ella.

MAYOR CONSUMO		MENOR CONSUMO

for (i=0; i<N; i++)
  for (j=0; j<N; j++)
  P[j][i] = Q[j][i]
	(a)	for (j=0; j<N; j++)
		  for (i=0; i<N; i++)
		  P[j][i] = Q[j][i]

for (i=0; i<N; i++)
  for (j=0; j<N; j++)
  for (k=0; k<N; k++)
  X[i][j]=Y[i][j]+Z[k][j]
	(b)	for (i1=0; i1<N; i1+=t)
		  for (j1=0; j1<N; j1+=t)
		  for (k1=0; k1<N; k1+=t)
		  for (i=i1; i<min(i1+t,N); i++)
		  for (j=j1; j<min(j1+t,N); j++)
		  for (k=k1; k<min(k1+t,N); k++)
		  X[i][j]=Y[i][j]+Z[k][j]

for (i=0; i<N; i++)
  P[i] = Q[i]
for (i=0; i<N; i++)
  P[i] = P[i]*R[i]
	(c)	for (i=0; i<N; i++) {
		  P[i] = Q[i]
		  P[i] = P[i]*R[i]
		}

for (i=0; i<N; i++)
  for (j=0; j<N; j++)
  for (k=0; k<N; k++)
  X[i][j]=Y[i][j]+Z[k][j]
	(d)	for (i=0; i<N; i+=b)
		  for (j=0; j<N; j++)
		  for (k=0; k<N; k++) {
		  X[i][j]=Y[i][j]+Z[k][j]
		  X[i+1][j]=Y[i][j]+Z[k][j]
		  ...
		  X[i+b-1][j]=Y[i][j]+Z[k][j]
		}

for (j=0; j<N; j++)
  for (i=0; i<N; i++)
  P[j][i] = Q[i][j]
	(e)	for (j=0; j<N; j++)
		  for (i=0; i<N; i++)
		  P[j][i] = Q1[j][i]

for (i=0; i<N; i++)
  for (j=0; j<N; j++)
  P[i] = ...
	(f)	for (i=0; i<N; i++) {
		  P[i] = ...
		  for (j=0; j<N; j++)
		...

for (i=0; i<N; i++) {
  ... = P[i]
  for (j=0; j<N; j++)
  P[i] = R[i]
}
	(g)	for (i=0; i<N; i++) {
		  ... = P[i]
		  for (j=0; j<N; j++)
		  Q[i] = R[i]
		}

Figura 5.3.- Técnicas para reducir el consumo de potencia y aumentar las prestaciones.