

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

Programación en ensamblador: conceptos básicos

Luis Rincón Córcoles
Licesio J. Rodríguez-Aragón



Programa

Programación en ensamblador: conceptos básicos

1. Introducción.
2. Desarrollo de programas.
3. Sintaxis de un lenguaje ensamblador.
4. Pseudoinstrucciones en ensamblador.
5. Directivas en ensamblador.

- D.A. PATTERSON, J.L. HENNESSY. *Estructura y diseño de computadores*. Reverté, 2000.
- A. CLEMENTS. *Microprocessor Systems Design*. 3rd edition, ITP - PWS Publishing Company, 1997.
- J. SEPTIÉN, H. MECHA, R. MORENO, K. OLCOZ. *La familia del MC68000*. Síntesis, 1995.
- *M68000 8/16/32 Bit Microprocessors User's Manual*. 9th edition. Motorola, 1993.
- *Motorola M68000 Family Programmer's Reference Manual*. Motorola, 1992.
- P. DE MIGUEL. *Fundamentos de los computadores*. 7^a edición, Paraninfo, 1999.
- C. CERRADA, V. FELIU. *Estructura y Tecnología de Computadores I*. UNED, 1993
- W. STALLINGS. *Organización y Arquitectura de Computadores*. 5^a edición, Prentice-Hall, 2000.



1. Introducción

- Los computadores ejecutan programas.
- Programa: secuencia de operaciones conducentes a resolver un problema determinado.
- Características de los programas
 - Están compuestos por secuencias de instrucciones o sentencias.
 - Se escriben utilizando una notación formal conveniente.
 - Pueden ser escritos por personas (programadores), o bien pueden ser generados automáticamente mediante una herramienta adecuada.
 - Un programa en ejecución se encuentra cargado en memoria principal.
- Lenguaje de programación: una notación formal para describir algoritmos o funciones que serán ejecutadas por un computador.



Tipos de lenguajes de programación

- Lenguajes de bajo nivel: cercanos a la arquitectura de la máquina.
- Lenguajes de alto nivel: cercanos a la forma de pensar del programador.
- Lenguaje máquina: el único que la circuitería de la máquina es capaz de interpretar.
 - Sus instrucciones se encuentran codificadas en binario.
- Lenguajes simbólicos: no son directamente interpretables por la circuitería de la máquina.
 - Se codifican mediante símbolos alfanuméricos, de puntuación, paréntesis, separadores, etc.



Lenguajes de alto nivel

- Son métodos convenientes y sencillos de describir las estructuras de información y las secuencias de acciones precisas para ejecutar tareas concretas.
- Los lenguajes de alto nivel se acercan de alguna manera a la forma en que las personas resolvemos los problemas.
- Características:
 - Posibilidad de traducción automática a lenguaje máquina.
 - Independencia de la arquitectura del computador.
 - Transportabilidad entre diferentes computadores.
- Algunos tipos de lenguajes de alto nivel:
 - Lenguajes de propósito general.
 - Lenguajes de propósito específico (comerciales, científicos, educativos, etc).
 - Lenguajes procedimentales
 - Lenguajes declarativos
 - Lenguajes de diseño de sistemas de información.
- Los lenguajes de alto nivel son lenguajes simbólicos no comprensibles directamente por la circuitería del computador.



Lenguajes de bajo nivel

- Se encuentran totalmente vinculados a la estructura del computador.
- Están diseñados para sacar el máximo partido de las características físicas del computador.
- Características:
 - Dependencia absoluta de la arquitectura del computador.
 - Imposibilidad de transportar programas entre distintas máquinas, salvo que sean de la misma familia o compatibles.
 - Instrucciones poco potentes.
 - Programas muy largos.
 - Códigos de operación, datos y referencias en binario.
- Tipos:
 - Lenguaje máquina.
 - Códigos de operación, datos y referencias en binario.
 - Directamente interpretable y ejecutable por la circuitería del computador.
 - Lenguaje ensamblador.

7



Lenguaje ensamblador

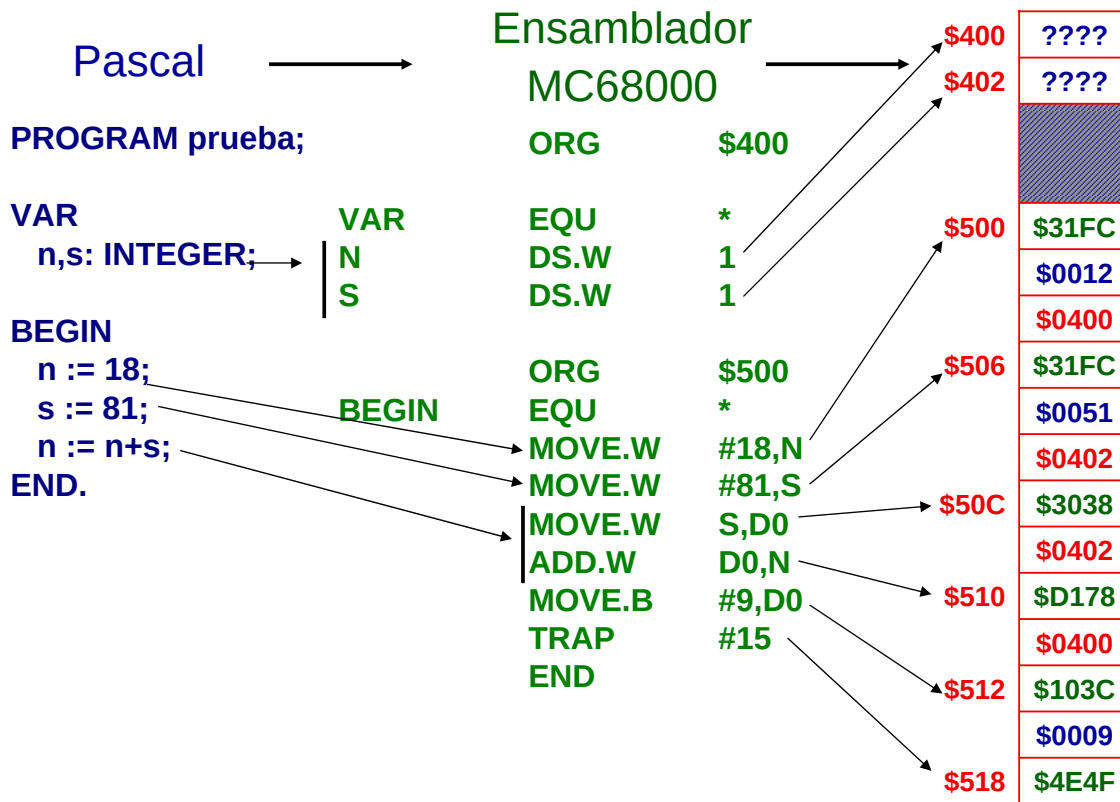
- El lenguaje ensamblador (o lenguaje de ensamble, *assembly language*) es la representación simbólica de la codificación binaria de un computador.
 - Códigos de operación representados mediante códigos nemotécnicos.
 - Datos y referencias codificadas mediante nombres simbólicos (símbolos o etiquetas).
- Existe una correspondencia biunívoca entre las instrucciones de máquina y las instrucciones de un lenguaje ensamblador.
 - Cada instrucción ensamblador es una codificación simbólica de una instrucción de máquina.
 - Excepción: ensambladores que proporcionan una máquina virtual con pseudoinstrucciones.
- Macroensambladores: permiten al programador definir secuencias de instrucciones con nombre.
- El lenguaje ensamblador debe ser traducido a lenguaje máquina para poder ser interpretado y ejecutado directamente por el computador.

8



Lenguajes de alto nivel y lenguajes de bajo nivel

Código máquina MC68000



9



Lenguajes de alto nivel y lenguajes de bajo nivel

➤ La codificación de programas en binario es conveniente y natural para la circuitería del computador, pero es difícil para un programador humano.

➤ El lenguaje ensamblador surgió para facilitar la escritura de programas de computador.

- Es un lenguaje simbólico que da nombres a las instrucciones de máquina, y permite dar nombres a posiciones de memoria que contienen instrucciones o datos.

➤ Los lenguajes de alto nivel facilitan la tarea de los programadores, ya que se encuentran más próximos a la forma de pensar de los humanos.

- Control estructurado de flujo.
- Comprobación de tipos.

➤ La programación en lenguajes de alto nivel es más productiva, ya que los programas son más cortos (en cuanto a líneas de código).

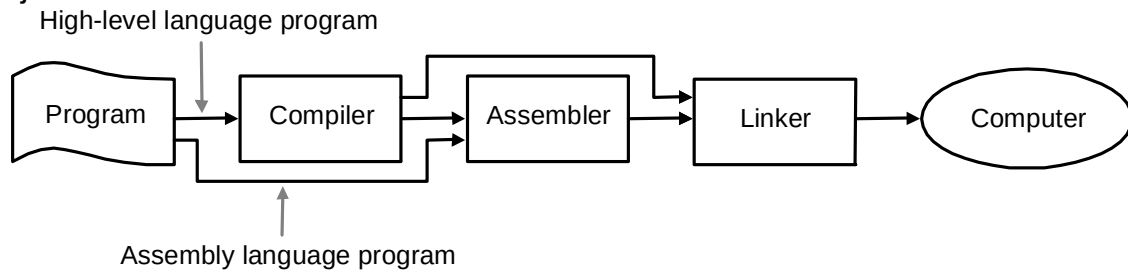
➤ Hoy en día la práctica totalidad de los programadores trabaja utilizando lenguajes de alto nivel.

10



Lenguajes de alto nivel y lenguajes de bajo nivel

- La circuitería del computador tan solo comprende los programas escritos en lenguaje máquina.
- Los programas escritos en lenguajes simbólicos deben ser traducidos a lenguaje máquina antes de ser ejecutados.
 - Compilador (*compiler*): traductor de lenguaje de alto nivel a lenguaje ensamblador.
 - Hoy día los compiladores pueden traducir los programas directamente a lenguaje máquina.
 - Ensamblador (*assembler*): traductor de lenguaje ensamblador a lenguaje máquina.
 - Montador (*linker*): crea el código máquina ejecutable final.
 - Cargador (*loader*): carga el código ejecutable en memoria y lo prepara para su ejecución.



11



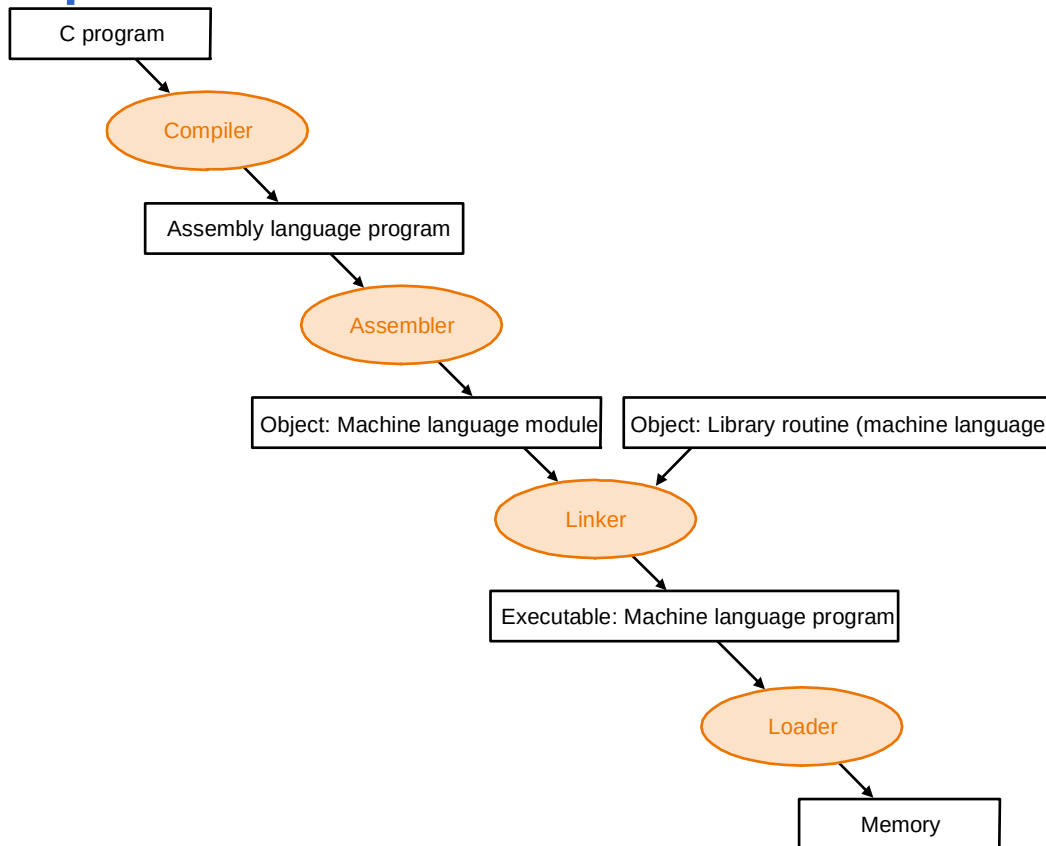
Terminología

- Código (programa) fuente: código escrito por el programador.
 - Puede estar escrito en cualquier lenguaje: alto nivel, ensamblador, código máquina (inusual).
 - Puede contener errores sintácticos porque el programador haya escrito mal el programa.
- Código (programa) objeto: código obtenido al traducir el código a lenguaje máquina.
 - No contiene errores sintácticos.
 - A veces no es directamente ejecutable.
- Código (programa) ejecutable: listo para ser ejecutado en el computador.
 - Puede contener errores lógicos debidos a que el programa no esté bien diseñado y no realice correctamente la función para la cual se creó.

12



Jerarquía de traducción



13



Lenguajes de alto nivel y lenguajes de bajo nivel

⇒ Factores que miden la calidad de los programas ejecutables:

- Tamaño en número de palabras de memoria.
- Velocidad.

⇒ Tradicionalmente los compiladores generaban código máquina de inferior calidad que el que podían escribir programadores humanos.

- Las memorias son mucho mayores hoy en día: el tamaño ha dejado de ser crítico.

⇒ Los compiladores proporcionan hoy en día un código máquina de alta calidad pequeño y rápido, haciendo poco atractiva la programación en ensamblador.

- Los programadores de ensamblador siguen teniendo ventaja en cuanto a que disponen de un mayor conocimiento global del programa que les permite realizar determinadas optimizaciones del código que resultan muy difíciles para los compiladores.

⇒ Puede ser recomendable programar en ensamblador cuando la velocidad del programa y su tamaño sean críticos.

- Caso especial: computadores empotrados (*embedded computers*).

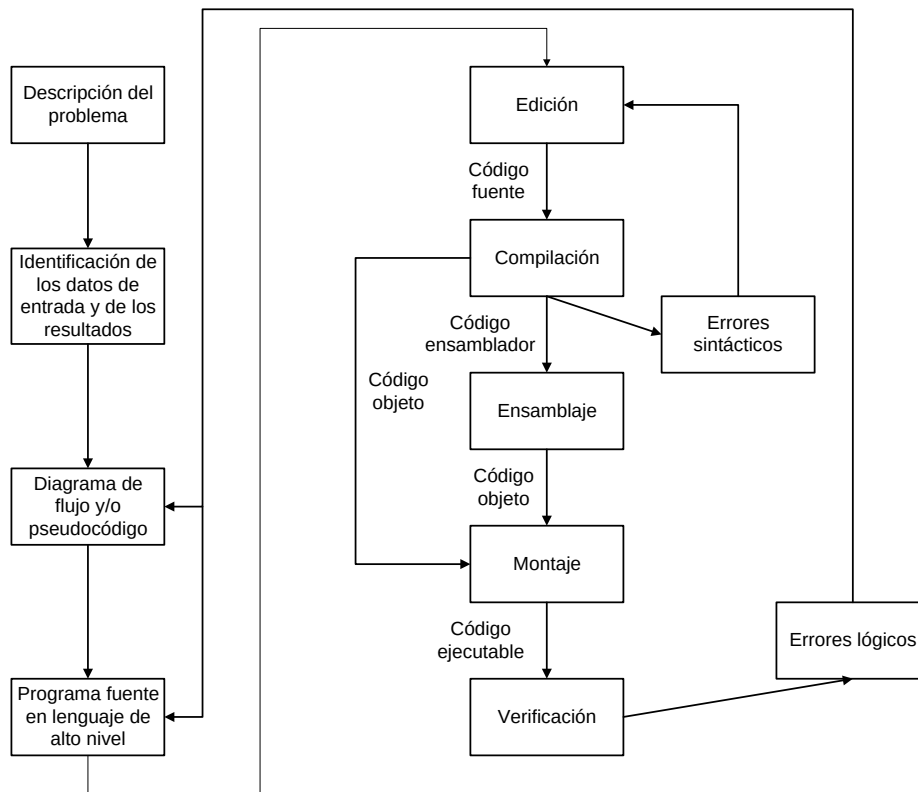
⇒ Solución mixta:

- Programar en alto nivel la mayor parte del código.
- Programar en ensamblador las partes críticas en cuanto a velocidad.
- Programar en ensamblador los sistemas con un tamaño de memoria muy reducido.

14

2. Desarrollo de programas

⇒ Fases de desarrollo de un programa escrito en un lenguaje de alto nivel.



15

Desarrollo de programas en ensamblador: fases

1. Especificación del problema.
2. Elaboración del algoritmo de la solución.
3. Diseño del diagrama de flujo.
4. Codificación en ensamblador.
5. Edición del programa fuente.
6. Traducción del programa fuente a código máquina.
7. Montaje del programa ejecutable.
8. Carga y ejecución del programa.
9. Verificación del funcionamiento del programa.

16



Desarrollo de programas en ensamblador: fases

1. Especificación del problema.

- Estudio detallado de los requisitos que debe cumplir la solución.
- Análisis de las entradas al programa y las salidas que debe producir.

2. Elaboración del algoritmo de la solución.

- Establecimiento de los módulos y las subrutinas que compondrán el programa.
- Definición y organización de los datos simples y las estructuras de datos necesarias.
- Si hay varios módulos, se codifica cada uno de ellos por separado.

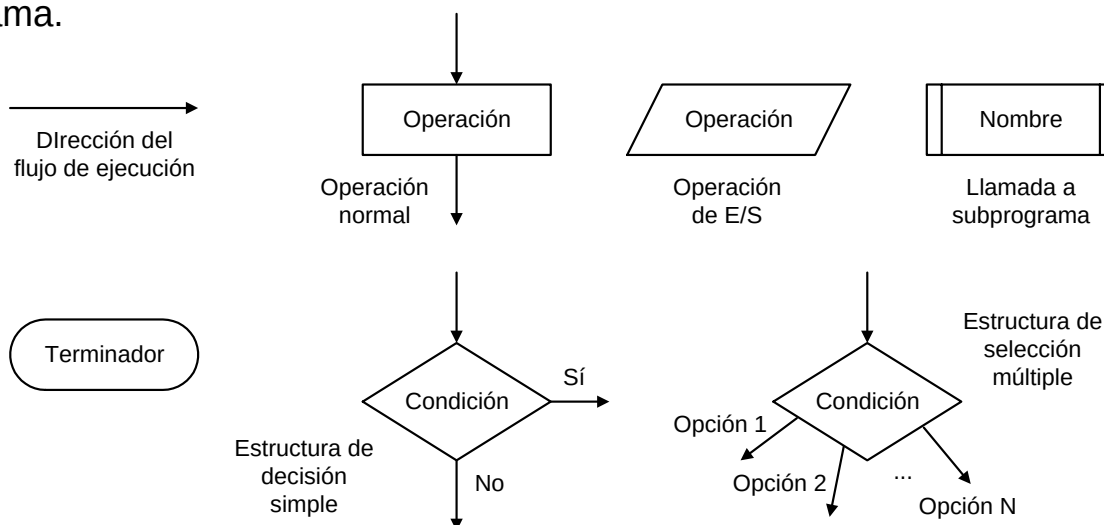
17



Desarrollo de programas en ensamblador: fases

3. Diseño del diagrama de flujo.

- Diagrama de flujo: notación gráfica que permite representar la estructura de un programa.



➤ Tipos:

- Orientado al problema: capta la estructura del algoritmo de la solución.
- Orientado al lenguaje: capta peculiaridades del lenguaje de programación utilizado.

18



Desarrollo de programas en ensamblador: fases

4. Codificación en ensamblador.

- El programa fuente se escribe en papel..
- Si el diagrama de flujo es suficientemente detallado, esta fase es inmediata.
- Si hay varios módulos, se codifica cada uno de ellos por separado.

5. Edición del programa fuente.

- Transcripción del programa escrito en lenguaje ensamblador a un fichero de texto.
- Para ello se utilizará como herramienta un editor de textos ASCII.
- Si hay varios módulos, se edita cada uno de ellos por separado.

19



Desarrollo de programas en ensamblador: fases

6. Traducción del programa fuente a código máquina.

- Ensamblador
 - Traductor de lenguaje ensamblador a lenguaje máquina.
 - Genera un fichero con el código objeto equivalente al código fuente completo, junto con información necesaria para el montaje.
- Compilador
 - Traductor de lenguaje de alto nivel a ensamblador.
 - Hoy todos los compiladores traducen directamente a lenguaje máquina.
 - En tal caso generan un fichero con el código objeto equivalente al código fuente completo, junto con información necesaria para el montaje.
- Si el código fuente contiene errores sintácticos, no se genera código objeto.
- Intérprete: traductor de lenguaje de alto nivel a lenguaje máquina.
 - Un intérprete traduce y ejecuta las instrucciones del programa fuente una por una, sin generar fichero alguno con código objeto.
 - Los intérpretes son propios de los llamados **lenguajes interpretados** (BASIC, LISP, etc).

20



Desarrollo de programas en ensamblador: fases

7. Montaje del programa ejecutable.

➤ El código objeto procedente del traductor normalmente no puede ejecutarse directamente, ya que le falta cierta información que lo impide.

- Código objeto procedente de bibliotecas predefinidas.
- Código objeto procedente de la traducción de otros módulos.
- Referencias cruzadas entre objetos de diferentes módulos.
- Información dependiente del sistema operativo.

➤ El montador se encarga de:

- Resolver las referencias entre elementos de varios módulos.
- Asignar direcciones a los diferentes módulos del programa.
- Unir todos los módulos generando un fichero conteniendo el módulo ejecutable.

8. Carga y ejecución del programa.

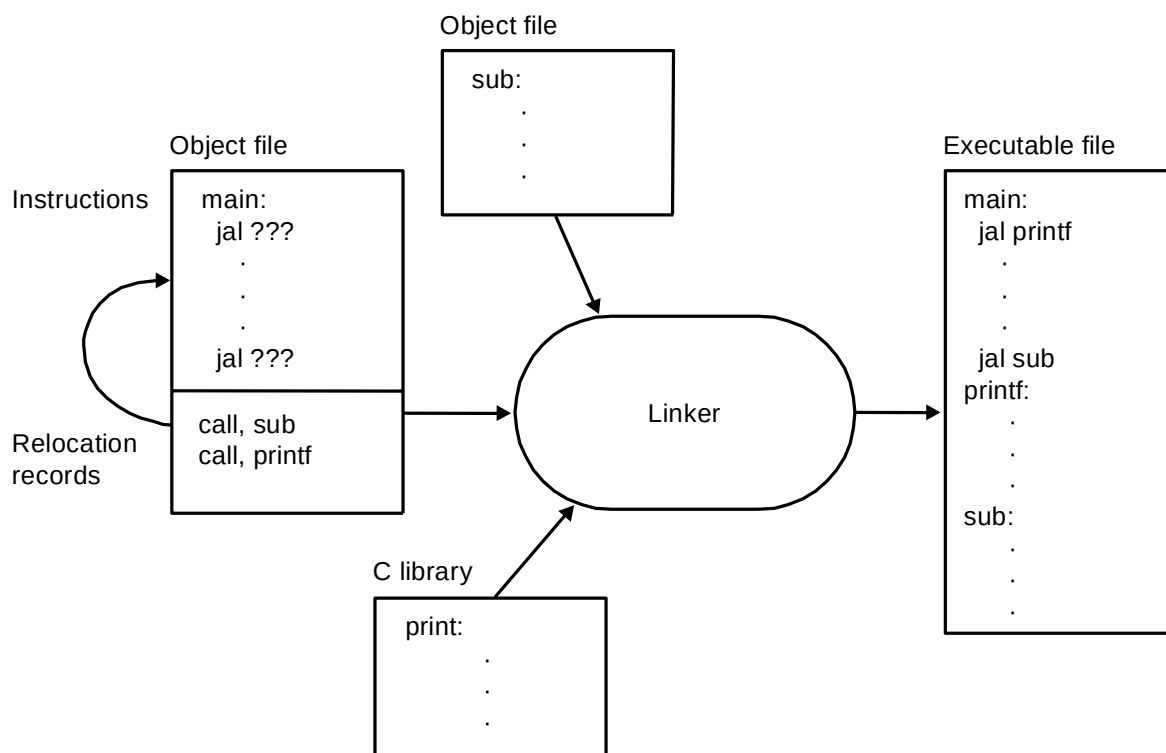
➤ Consiste en la transferencia del programa ejecutable a la memoria del computador desde el fichero en disco, y en el posterior lanzamiento de su ejecución.

➤ Herramienta utilizada: cargador.

Pertenece al sistema operativo.



Montaje del programa ejecutable





Desarrollo de programas en ensamblador: fases

9. Verificación del funcionamiento del programa.

- Comprobación de los efectos causados por la ejecución de las instrucciones en las variables y el estado de la máquina.
 - Ejecutándolo directamente.
 - Utilizando un depurador.
- Depurador: herramienta que permite verificar el funcionamiento de los programas paso a paso a nivel de código fuente.
 - Permite cargar programas en la memoria del computador.
 - Permite ejecutar programas paso a paso o de forma continua.
 - Permite establecer puntos de ruptura para detener la ejecución del programa en lugares determinados.
 - Permite ver los contenidos de los registros y la memoria, y modificar sus contenidos en cualquier instante durante la ejecución de los programas.
- Caso particular: programa monitor.
 - Se encuentra grabado en la memoria ROM del computador.
 - Permite depurar programas en lenguaje máquina.

23

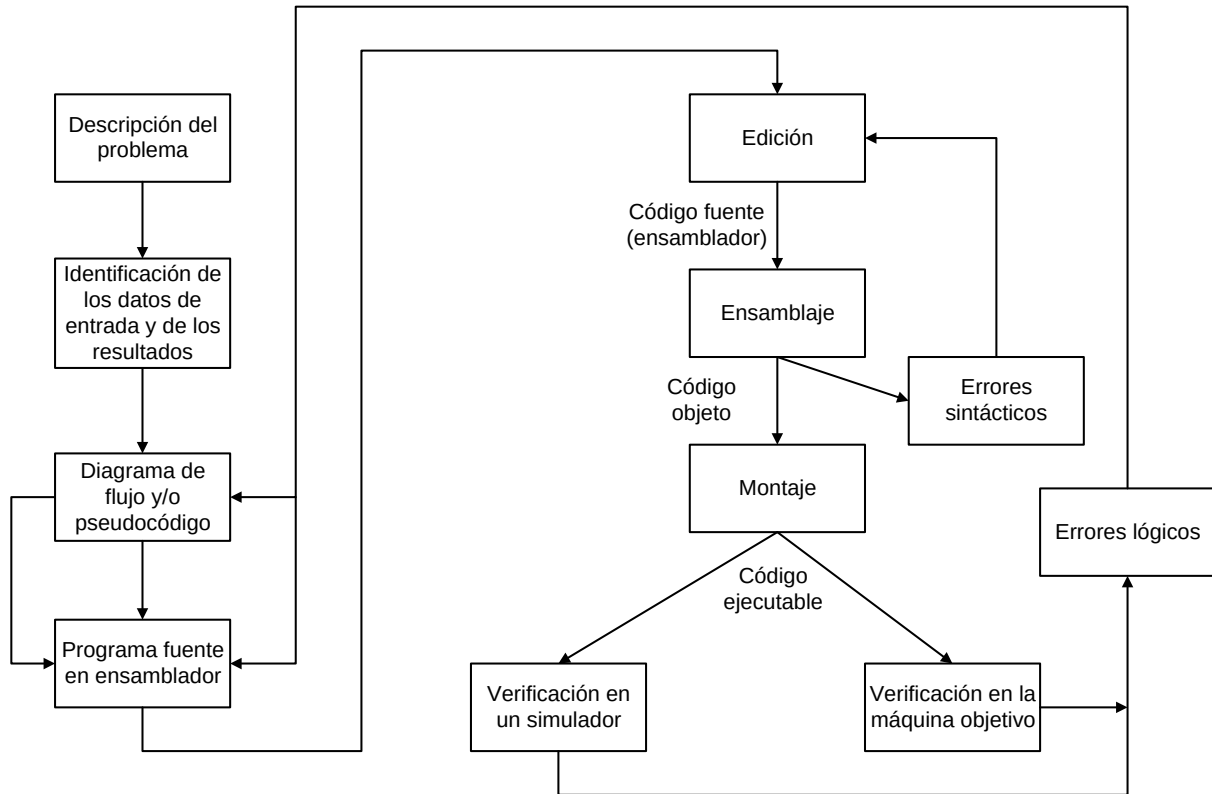


Desarrollo cruzado de programas

- Se realiza cuando el programa será ejecutado en una máquina (**máquina objetivo**) con una arquitectura distinta de la del computador en que se ha desarrollado (**sistema de desarrollo**).
 - Sistemas empotrados.
 - Sistemas aún no implementados.
- Traducción: ensamblador cruzado.
 - El código fuente y el código máquina generado corresponden con la arquitectura de la máquina objetivo, mientras que el traductor corre en el sistema de desarrollo.
- Montaje: montador de enlaces cruzado.
 - Genera código para la máquina objetivo, pero corre en el sistema de desarrollo.
- Verificación: mediante un simulador o mediante la máquina objetivo.
 - Simulador: herramienta que corre en la máquina de desarrollo y que emula la máquina objetivo.
 - Los simuladores ofrecen posibilidades de depuración del programa ejecutable.

24

Desarrollo cruzado de programas



25

3. Sintaxis de un lenguaje ensamblador

➤ Tipos de líneas en un programa en ensamblador:

- Instrucciones: ejecutables por el computador.
- Pseudoinstrucciones: secuencias de instrucciones con nombre predefinido.
- Directivas: indicaciones para el traductor.
- Líneas de comentario: suelen comenzar por un carácter especial, y son ignoradas por el traductor en cuanto a la generación de código objeto.

➤ Estructura de una instrucción en memoria:

Dirección	Código de operación	Direcciones de operandos
-----------	---------------------	--------------------------

➤ Estructura de una línea de un programa escrito en ensamblador:

Etiqueta	Nemotécnico	Operandos	Comentarios
----------	-------------	-----------	-------------

➤ Los campos de una línea de programa se separan mediante delimitadores de campo.

26

- ➔ La etiqueta identifica la línea que encabeza.
- ➔ Recomendaciones:
 - Utilizar sólo letras o números (comenzar por una letra).
 - Colocar la primera letra de la etiqueta en el primer carácter de la línea.
- ➔ Las etiquetas permiten al programador definir símbolos.
 - Símbolo: tira de caracteres que identifica un valor.
- ➔ Cuando el traductor se encuentra con una etiqueta, la inserta en la tabla de símbolos.

Nombre del símbolo	Tipo del símbolo	Valor del símbolo
SIMB1	TIPO1	VALOR1
SIMB2	TIPO2	VALOR2
...	...	...

- ➔ Identifica:
 - Un código de operación de una instrucción del lenguaje.
 - Una pseudoinstrucción.
 - Una directiva.
- ➔ El nemotécnico nunca comienza en el primer carácter de la línea.



Campos de operandos

- Identifican los operandos que intervienen en la instrucción o directiva.
- Tipos de operandos:
 - Constantes
 - Numéricas: suelen llevar un prefijo o un sufijo que identifica la base en que están expresadas: decimal (D), binaria (%B), octal (O,@,Q,C) o hexadecimal (H,\$, 0x).
 - Alfabéticas: entre comillas.
 - Símbolos
 - Predefinidos: registros, contador de dirección de ensamblaje, etc.
 - Implícitos.
 - Explícitos.
 - Expresiones
 - Pueden incluir constantes, símbolos implícitos o explícitos, el símbolo del contador de dirección de ensamblaje, paréntesis u operadores sencillos (+, -, *, /, %, &, |, ^, <, etc).
 - **¡LAS EXPRESIONES NO PUEDEN INCLUIR NOMBRES DE REGISTROS NI USAR EL CONTENIDO DE LAS VARIABLES!**

29



Comentarios

- Son indicaciones para una mejor comprensión del programa ensamblador.
- Pueden situarse a continuación de los operandos, con al menos un espacio en blanco entre medias.
 - En ocasiones puede exigirse incluir además algún otro carácter delimitador ("*", ":", "#").
- En ocasiones se incluyen líneas completas de comentario.
 - En ese caso, la línea de comentario comenzará con un carácter delimitador en la primera posición de la línea ("*", ":", "#").

30



4. Pseudoinstrucciones en ensamblador

- Una pseudoinstrucción es una instrucción ofrecida por el ensamblador, pero no realizada por la circuitería.
- A una pseudoinstrucción le corresponde una secuencia definida de instrucciones realizadas por la circuitería.
- Las pseudoinstrucciones proporcionan a la máquina un lenguaje más rico que el que realiza la circuitería.
- Las pseudoinstrucciones facilitan la programación en ensamblador sin complicar la circuitería (aunque sí complican el diseño del traductor).
- No todos los traductores cuentan con pseudoinstrucciones.

31



5. Directivas en ensamblador

- Informan al traductor acerca de cómo debe traducir el código fuente a código máquina, pero no producen instrucciones en lenguaje máquina.
- Las directivas pueden indicar información relativa a diversas cuestiones:
 - El aspecto del listado de ensamblaje.
 - La definición de direcciones absolutas del programa.
 - Los tamaños y nombres simbólicos adjudicados a constantes y variables.
 - Las direcciones de ensamblado de las instrucciones y los datos.
 - La definición de símbolos importables y exportables.
 - La inclusión y delimitación de secciones en el código.
 - Etc.
- Las directivas son procesadas por el traductor, que a partir de ellas adopta las acciones pertinentes para realizar el proceso de ensamblaje y generación de listados y código objeto.
- El código objeto no contiene ninguna directiva.
 - El programa ejecutable desconoce la existencia de directivas.
 - Las directivas nunca son tenidas en cuenta por el computador en tiempo de ejecución.

32