

TEMA

33



CEDE

Programación en lenguaje ensamblador. Instrucciones básicas. Formatos. Direccionamientos.

elaborado por
EL EQUIPO DE PROFESORES
DEL CENTRO DOCUMENTACIÓN

1. INTRODUCCIÓN

Una computadora digital o, mejor dicho, su parte física, sólo distingue datos de tipo binario, es decir, constituidos por dos únicos valores a los que se denomina valor 0 y valor 1 y que, físicamente, se materializan con tensiones comprendidas entre 0 y 0,4 voltios y entre 4 y 5 voltios, respectivamente. Para representar datos que contengan una información se utilizan una serie de unos y ceros y cuyo conjunto indica dicha información.

La información que hace que el hardware de la computadora realice una determinada actividad se llama "instrucción". Por consiguiente una instrucción es un conjunto de unos y ceros. Las instrucciones así formadas equivalen a acciones elementales de la máquina, por lo que al conjunto de dichas instrucciones que son interpretadas directamente por la máquina se denomina lenguaje máquina.

El lenguaje máquina fue el primero que empleó el hombre para la programación de las primeras computadoras. Una instrucción en lenguaje máquina puede representarse de la siguiente forma:

```
011011001010010011110110
```

Esta secuencia es fácilmente ejecutada por la computadora, pero es de difícil interpretación, siendo aún más difícil la interpretación de un programa (conjunto de instrucciones) escrito de esta forma.

Con la práctica en el manejo de la máquina se cayó en la cuenta de que se podría utilizar la propia máquina para ayudar en la traducción de estos programas. Es decir, que si a una máquina elemental se le dotaba de un programa, también elemental, que tradujera un número determinado de caracteres alfabéticos en una secuencia de unos y ceros, se podrá escribir un programa constituido por una secuencia de grupos de caracteres alfabéticos, en la que cada uno de los grupos indicaría una acción a realizar por el ordenador y, una vez escrito el programa, sería la propia máquina la que pasaría los grupos de caracteres a bits.

De esta forma aparecieron los lenguajes ensambladores (Assembler, en inglés). Poco a poco, con el avance de la programación (software), estas primeras y sencillas ayudas se fueron haciendo más complejas.

Se introdujo la posibilidad de indicar a la computadora la dirección de un salto en la secuencia de ejecución de un programa mediante la utilización de etiquetas.

A los programas que permiten pasar del programa escrito de esta manera (programa fuente, en ensamblador) al lenguaje máquina también se les llama normalmente "ensambladores".

Aún con todas estas sofisticaciones y ayudas, el programador del lenguaje ensamblador debe conocer perfectamente el sistema físico (hardware) de la máquina con que trabaja, pues aunque emplee mnemotécnicos, etiquetas, etc., éstas sirven para indicar una posición de memoria determinada, un registro o cualquier otra parte de la máquina.

Por eso se dice que el lenguaje ensamblador es un lenguaje de bajo nivel, es decir, ligado con el «hard» concreto de una determinada máquina.

2. INSTRUCCIONES BÁSICAS

En Informática, una instrucción es un conjunto de símbolos que la computadora es capaz de interpretar con objeto de realizar las acciones que dichos símbolos le indican.

Las instrucciones en Lenguaje Ensamblador son muy elementales; indican acciones muy concretas, nombrando incluso a los elementos que tienen los datos o que guardan los resultados, como son los registros, posiciones de memoria, periféricos, etc.

2.1. TIPOS DE INSTRUCCIONES

Podemos clasificar las instrucciones, según la acción que ordenan, en los siguientes tipos:

Instrucciones de transferencia de datos: Dicen a la UCP que copie el dato de un registro o posición de memoria en otro registro u otra posición de memoria.

Instrucciones de entrada y salida: Indican a la UCP que debe ordenar la transferencia de datos entre algún elemento interno de la computadora (registros, posiciones de memoria, etc.) y alguno externo (periféricos). Suele haber una instrucción para pasar dato desde un periférico al interior de la computadora (entrada) y otra para pasar el dato del interior al periférico (salida).

Instrucciones de cálculo: Indican a la UCP la operación que debe realizar. Pueden ser aritméticas (sumar, restar, etc.), lógicas (O, Y, NO, etc.), de comparación y de rotación y

desplazamiento de los bits de un registro o posición de memoria hacia la izquierda o hacia la derecha, etc.

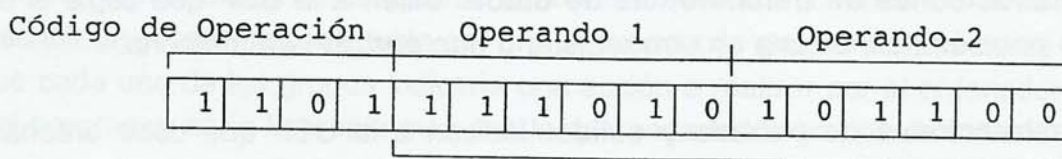
Instrucciones de ruptura de secuencia: Indican a la UCP que la siguiente instrucción que debe ejecutarse no es la que corresponde según la secuencia física del programa, sino que debe realizar un salto a otro punto del mismo programa. Estos saltos pueden ser incondicionales o condicionales, según se realicen siempre o sólo cuando se cumpla alguna condición, que, normalmente, dependerá de algún bit del registro de estado de la UCP.

Instrucciones de control de la UCP: Modifican el contenido del registro de estado o pueden hacer que la UCP se detenga en la interpretación de instrucciones hasta nueva orden, que se dará mediante algún dispositivo externo a la UCP.

2.2. PARTES CONSTITUYENTES DE UNA INSTRUCCIÓN

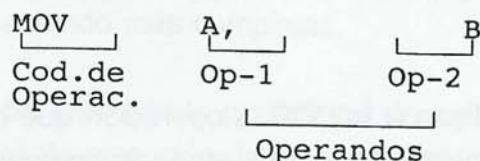
En toda instrucción podemos diferenciar dos partes o campos. Una de estas partes indica a la UCP la operación que debe ejecutar, y se llama campo de código de operación; la otra le indica de dónde debe tomar los datos y dónde poner los resultados, y se denomina campo de operandos.

En la Figura se muestra una instrucción indicando el campo de código de operación y los campos de operandos. Como se puede apreciar, la parte (a) de la figura representa una instrucción en el lenguaje máquina. es decir, un conjunto de unos y ceros que la máquina es capaz de interpretar de forma sencilla, pero que al programador le resulta más complicado hacerlo. La parte (b) de la figura muestra cómo sería la misma instrucción, pero ahora en lenguaje ensamblador. En esta forma de representación el programador la entiende fácilmente, pero para que la máquina sea capaz de ejecutarla deberá traducirla primero a lenguaje máquina.



Campo de operandos

(a) Instrucción en lenguaje máquina



(b) Instrucción en lenguaje ensamblador

Campo código de operación: Este campo es obligatorio en todas las instrucciones, ya que, como se ha dicho, indica a la UCP precisamente la operación que debe realizar. Cada instrucción u operación estará codificada, en lenguaje máquina, en una serie de ceros y unos de forma que no haya dos instrucciones diferentes que tengan el mismo código.

El número de bits que se emplean en el código puede ser idéntico en todas las instrucciones de una determinada máquina, en cuyo caso se dice que esa máquina tiene un código de operación fijo; o puede variar, en cuyo caso se dice que la máquina tiene un código de operación variable o expandido.

El código de operación suele estar ocupando los bits más significativos de la palabra, los situados más a la izquierda, como se ve en la Figura (a). Si una instrucción ocupa más de una palabra, el código de operación estará en la primera palabra que lea la UCP, para que ésta pueda ir interpretando la instrucción y saber si después hay más palabras pertenecientes a la misma instrucción o no.

También en lenguaje ensamblador se indica el campo de código de operación en primer término.

A los caracteres que forman el código de operación en lenguaje ensamblador se les denomina "mnemotécnicos".

Campo de operandos. Además del código de operación, en una instrucción podemos tener otro campo, como ya se ha indicado anteriormente, llamado campos de operandos. Este campo no está presente en todas las instrucciones, ya que hay algunas que no emplean datos y otras en la que la localización de los mismos está implícita en el propio código de operación. La longitud de este campo es normalmente variable, dependiendo del número de operandos que utilice la instrucción y de la forma que se indique a la UCP el acceso a los mismos. De existir campos de operandos, se codifica la dirección de los mismos a continuación del campo de código de operación, como se puede ver en la Figura.

En lenguaje ensamblador se emplean símbolos alfanuméricos para representar los operandos. Pueden ser nombres de variables que representen posiciones de memoria, registros, etc., utilizando una representación decimal, octal, hexadecimal, etc.

2.3. FORMATOS

Como ya se ha visto en los apartados anteriores, una instrucción puede tener, además del código de operación, un número variable de operandos; pueden ser varios, sólo uno o

ninguno. Esto hace que una misma máquina pueda tener instrucciones de diferentes formas y longitudes.

Las variantes posibles en cuanto a las longitudes de código de operación, campos de operando e instrucción:

(a) Longitud de instrucción fija, código de operación extendido.

Cod.Op.	Operandos
---------	-----------

Cod.Op.	Operandos
---------	-----------

Cod.Op.

(b) Longitud de instrucción variable, código de operación fijo.

Cod.Op.	Operandos
---------	-----------

Cod.Op.	Operandos
---------	-----------

Cod.Op.

(c) Longitud de instrucción variable, código de operación extendido.

Cod.Op.	Operandos
---------	-----------

Cod.Op.	Operandos
---------	-----------

Cod.Op.

A continuación se estudiarán los diferentes formatos que podemos encontrar en diferentes máquinas, dependiendo del número de operandos o direcciones que utilicen en la instrucción.

Formato de cuatro direcciones. En la actualidad no es frecuente encontrar máquinas con este formato de instrucción. Dos direcciones indican los operandos con los que operará la UCP, otra dirección le indicará dónde debe guardar el resultado, y la otra le dirá dónde se encuentra la siguiente instrucción a ejecutar.

Estas máquinas no utilizan registro contador de programa, ni son necesarias las instrucciones de salto específicas. El problema de este formato es la gran cantidad de bits que necesitan las instrucciones para poder codificar tantas direcciones.

En la figura se muestra una instrucción de este formato, suponiendo que la máquina puede realizar accesos a cualquier posición de una memoria de 64K palabras de capacidad. Los bits necesarios para cada dirección son 16, por lo que se necesitarán $16 \times 4 = 64$ bits, más los correspondientes al código de operación, para cada instrucción.

bn....b64	b63.....b48	b47....b32	b31.....b16	b15.....b0
Cod.Op.	Operando1	Operando2	Resultado	Siguiente Inst.

Formato de tres direcciones. Ante la necesidad de simplificar las máquinas para abaratar costes y aumentar el rendimiento, se fue reduciendo la longitud de las instrucciones. Una manera de conseguirlo es eliminando información en las mismas.

Un avance muy importante en este sentido es la aparición del contador de programa, registro que guarda la dirección de la siguiente instrucción a ejecutar, por lo que ya no es necesario incluirla en la instrucción. De esta manera se pierde algo de flexibilidad en el secuenciamiento de las instrucciones, ya que la dirección se va calculando a medida que se ejecuta el programa de forma independiente a las instrucciones. Se hace necesario el uso de instrucciones de ruptura de secuencia, pero se ahorran bits en cada instrucción.

La figura muestra este formato suponiendo la misma posibilidad de direccionamiento que en el apartado anterior.

bn....b48	b47.....b32	b31.....b16	b15.....b0
Cod.Op.	Operando1	Operando2	Resultado

Formato de dos direcciones. El formato anterior aún ocupa muchas posiciones. En el ejemplo de la Figura anterior sólo los operandos necesitan 48 bits. En consecuencia, se optó por eliminar la dirección que indica dónde se guarda el resultado, empleándose par este fin uno de los dos operandos especificados en instrucción. El operando que recoge el resultado se llama destino; el otro se llama operando fuente. El operando destino puede utilizarse también para aportar el dato que se necesita para realizar la operación indicada en el código de operación junto con el operando fuente. Indudablemente, después de ejecutada la instrucción el contenido del operando destino se habrá modificado, perdiéndose el contenido anterior.

Este formato tiene variantes. Una consistiría en emplear un registro para especificar uno de los operandos; de esta forma se reduce la longitud total de la instrucción, ya que para indicar un registro de la UCP son necesarios muchos menos bits que para una dirección de memoria.

Por ejemplo, tomando la misma capacidad de memoria que en los apartados anteriores, vimos que para indicar una dirección de memoria eran necesarios 16 bits. Si suponemos que la UCP posee 8 registros de trabajo, serían suficientes tres bits para hacer referencia a cada uno de ellos. Esta variante del formato de dos direcciones también se denomina formato de una dirección y media.

Otra variante consistiría en especificar como operandos dos registros, con lo que se reduciría aún más la longitud de la instrucción.

En la Figura se muestran las variantes de este formato.

bn...b32	b31.....b16	b15.....b0
Cod.Op.	Operando1	Operando2

(a) dos direcciones de memoria

bn...b19	b18.....b3	b2 b1 b0
Cod.Op.	Operando1	Operando2

(b) una dirección de memoria y un registro

bn...b6	b5 b4 b3	b2 b1 b0
Cod.Op.	Operando1	Operando2

(c) dos registros

Formato de una dirección. En este formato se indica el registro de forma implícita en el código de operación, por lo que en el campo de operandos queda sólo la dirección de un operando. El otro operando, que recibe el resultado, se supone que es un determinado registro de trabajo de la UCP, denominado generalmente "acumulador".

Formato de cero direcciones. Existen máquinas que utilizan para guardar datos una estructura denominada pila. La pila no es más que un conjunto de posiciones consecutivas de memoria, y para acceder a las mismas se utiliza un registro llamado "apuntador de pila", que guarda la dirección de la cabecera de la pila, o elemento al que vamos a acceder. Cuando almacenemos un nuevo dato, este registro se incrementará y, cuando saquemos un dato, se decrementará para indicar el siguiente elemento.

Con esta estructura sólo hace falta especificar la operación, ya que la UCP supone que los operandos están guardados en la pila, adonde los va a buscar con la ayuda del apuntador de pila, y una vez realizada la operación supone que el resultado debe guardarlo en la misma pila.

3. MODOS DE DIRECCIONAMIENTO

Las diferentes formas de indicar en el campo de operandos dónde se encuentran los datos que la UCP tendrá que procesar se denominan "modos de direccionamiento".

Aunque los diseñadores de procesadores mezclan estos modos para aprovechar las ventajas de unos y otros o conseguir determinadas características en las máquinas, básicamente podemos distinguir cuatro modos de direccionamiento:

- Inmediato.
- Absoluto.
- Indirecto.
- Indexado.

Modo de direccionamiento inmediato. En este método, el valor del operando es el que consta en la propia instrucción. Se emplea este modo de direccionamiento cuando se desea utilizar constantes en el programa.

Es el de más rápido acceso al dato por obtenerse en la fase de búsqueda de la instrucción, sin necesidad de volver a leer ninguna otra vez la memoria principal. Sin embargo, no permite modificar el dato sin modificar la instrucción.

Modo de direccionamiento absoluto. En este modo, el campo de operandos indica la dirección en la que se encuentra el dato. Con este método debe hacerse un acceso a la memoria para buscar el valor del operando. Es, pues, un modo de direccionamiento más lento que el inmediato. Sin embargo, puede modificarse el valor del dato sin modificar para nada la instrucción; basta modificar el contenido de la posición de memoria que indica el campo de operandos.

Este método proporciona una manera sencilla y rápida de acceder a la posición de una variable; sin embargo, tiene dos problemas:

1. Que la variable siempre tiene que estar en una posición fija de memoria.

2. Que el campo de operando debe tener la longitud suficiente para poder indicar cualquier posición de memoria.

Una variante de este direccionamiento es el denominado "direccionamiento a registro", en el cual la dirección indica que el campo operando corresponde a la de uno de los registros de la UCP, no a una posición de memoria. De esta forma se gana en rapidez, ya que el acceso a un registro es mucho más rápido que el acceso a una posición de memoria, y en longitud de la instrucción.

El inconveniente de este modo de direccionamiento es que el valor a utilizar debe ponerse dentro del registro antes de usarlo y, como el número de registros es escaso, una vez utilizado debe guardarse de nuevo en la memoria.

Modo de direccionamiento indirecto. En este modo y en el siguiente se empleará el concepto de "dirección efectiva" para indicar la dirección de memoria donde se encuentra el dato.

En el modo de direccionamiento indirecto la dirección que se indica en la instrucción no es la dirección de dato, sino la dirección de la "dirección efectiva". Es decir, que para acceder al dato la UCP deberá buscar en la dirección que se indica en la instrucción y ahí hallará otra dirección, que es la que le indicará donde se encuentra el dato. Al final de la instrucción se habrán realizado dos accesos a memoria o registros, hasta llegar al dato.

¿Qué se persigue con semejante complicación de direccionamiento? Se consigue poder variar la posición del dato sin modificar la instrucción; es suficiente con variar el contenido de la dirección indicada en la instrucción, que, como hemos visto antes, puede ser un registro o una posición de memoria.

Modo de direccionamiento indexado. En este modo de direccionamiento, la dirección efectiva se calcula sumando un valor, llamado índice, a la dirección indicada en la instrucción. El valor índice se encuentra, generalmente, en algún registro de la UCP. En algunas computadoras existen registros especializados en guardar estos valores. En estos casos, los registros suelen indicarse en la instrucción de forma implícita y suele haber instrucciones especiales para incrementar o decrementar estos registros denominados "registros de índice".

La dirección efectiva se calcula sin modificar el contenido del registro de índice ni de la dirección que se indica en la instrucción. Esta última dirección también se denomina "dirección base", y el valor de índice, "desplazamiento".

El empleo de este método de direccionamiento es parecido al anterior. sin embargo, en el direccionamiento indirecto el registro que se empleaba para guardar la dirección efectiva debía tener la suficiente longitud como para indicar cualquier posición de memoria. Por ejemplo, si la memoria es de 64K palabras, el registro será como mínimo de 16 bits.

En el direccionamiento indexado el registro índice no tienen por qué ser tan grande. Por ejemplo, con un registro de 8 bits se puede acceder a 256 posiciones, desde la dirección base hasta la dirección base + 255, margen que en la mayoría de las aplicaciones es suficiente.

VENTAJAS Y DESVENTAJAS DEL LENGUAJE ENSAMBLADOR

Las ventajas del lenguaje ensamblador son:

- **Velocidad.** Como trabaja directamente con el microprocesador al ejecutar un programa, este lenguaje es el más cercano a la máquina y procesa más rápido.
- **Eficiencia de tamaño.** Un programa en ensamblador no ocupa mucho espacio en memoria porque no tiene que cargar librerías y demás como ocurre en los lenguajes de alto nivel.
- **Flexibilidad.** Es flexible porque todo lo que puede hacerse con una máquina, puede hacerse en el lenguaje ensamblador de esta máquina; los lenguajes de alto nivel tienen en una u otra forma limitaciones para explotar al máximo los recursos de la máquina.

Las desventajas del lenguaje ensamblador son:

- **Tiempo de programación.** Como es un lenguaje de bajo nivel requiere más instrucciones para realizar el mismo proceso, en comparación con un lenguaje de alto nivel. Por otro lado, requiere de más cuidado por parte del programador, pues es propenso a que los errores de lógica se reflejen más fuertemente en la ejecución.
- **Programas fuente grandes.** Por las mismas razones que aumenta el tiempo, crecen los programas fuentes; simplemente se requieren más instrucciones primitivas para describir procesos equivalentes. Esto es una desventaja porque dificulta el mantenimiento de los programas, y nuevamente reduce la productividad de los programadores.
- **Peligro de afectar recursos inesperadamente.** Que todo error que podamos cometer, o todo riesgo que podamos tener, podemos afectar los recursos de la máquina, programar en este lenguaje implica que la máquina se bloquee o se reinicie si algunas instrucciones hacen lo que no deben. Con este lenguaje es perfectamente posible (y sencillo) realizar secuencias de instrucciones inválidas, que normalmente no aparecen al usar un lenguaje de alto nivel.

- **Falta de portabilidad.** Porque para cada máquina existe un lenguaje ensamblador; por ello, evidentemente no es una selección apropiada de lenguaje cuando deseamos codificar en una máquina y luego llevar los programas a otros sistemas operativos o modelos de computadoras.

CREACIÓN Y DEPURACIÓN DE PROGRAMAS EN LENGUAJE ENSAMBLADOR

Edición

Los archivos fuente de código ensamblador deben estar en formato ASCII standard. Para esto puede usarse cualquier editor que permita crear archivos sin formato, e.g. Edlin, Edit, Write, El editor del Turbo Pascal, Works, Word, WordStar, etcétera. Las declaraciones pueden ser introducidas en mayúsculas y/o minúsculas. Una buena práctica de programación es poner todas las *palabras reservadas* (directivas e instrucciones) en mayúsculas y todo lo del usuario en minúsculas para fines de facilidad de lectura del código.

Las sentencias pueden comenzar en cualquier columna, no pueden tener más de 128 caracteres, no se permiten líneas múltiples ni códigos de control, y cada línea debe ser terminada con una combinación de *line-feed* y *carriage-return*. Los comentarios se declaran con ; y terminan al final de la línea.

Ensamblado

El ensamblado se lleva a cabo invocando al **MASM**. Este puede ser invocado, usando una línea de comando, de la siguiente manera:

```
MASM archivo [, [objeto][, [listado][, [cross]]]] [opciones] [;]
```

donde:

Archivo. Corresponde al programa fuente. Por defecto se toma la extensión **.ASM**.

Objeto. Es el nombre para el archivo objeto.

Listado. Nombre del archivo de listado de ensamblado.

Cross. Es un archivo de referencias cruzadas.

Opciones.

La otra forma de invocar al ensamblador es sólo tecleando **MASM** y respondiendo a la información que se solicita.

LINK

De la misma forma que el ensamblado, la fase de linkado se lleva a cabo con el **LINK**. Este puede ser invocado de la misma forma que el **MASM**. Los parámetros que este requiere son:

LINK objeto [, [ejecutable] [, [mapa] [, [librería]]]] [opciones] [;]

donde:

Objeto. Es el nombre para el archivo **.OBJ**.

Ejecutable. Nombre del archivo **.EXE**.

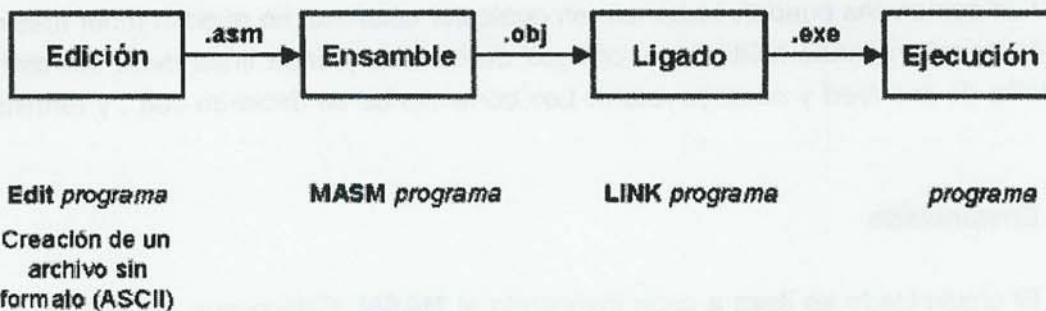
Mapa. Nombre del archivo mapa.

Librería. Nombre del archivo biblioteca de rutinas.

Opciones.

Ejecución

Para la ejecución del programa simplemente basta teclear su nombre en el prompt del Sistema Operativo. Con esto el programa será cargado en memoria y el sistema procederá a ejecutarlo.



RESUMEN

Las instrucciones en Lenguaje Ensamblador son muy elementales; indican acciones muy concretas, nombrando incluso a los elementos que tienen los datos o que guardan los resultados, como don los registros, posiciones de memoria, periféricos, etc.

Podemos clasificar las instrucciones, según la acción que ordenan, en los siguientes tipos:

- Instrucciones de transferencia de datos.
- Instrucciones de entrada y salida.
- Instrucciones de cálculo.
- Instrucciones de ruptura de secuencia.
- Instrucciones de control de la UCP.

En las instrucciones de lenguaje ensamblador podemos diferenciar dos partes o campos. Una de estas partes indica a la UCP la operación que debe ejecutar, y se llama campo de código de operación; la otra le indica de dónde debe tomar los datos y dónde poner los resultados, y se denomina campo de operandos.

Las variantes posibles en cuanto a las longitudes de código de operación, campos de operando e instrucción:

- (a) Longitud de instrucción fija, código de operación extendido.
- (b) Longitud de instrucción variable, código de operación fijo.
- (c) Longitud de instrucción variable, código de operación extendido.