# Preparador Informática

www.preparadorinformatica.com

# TEMA 33. INFORMÁTICA / S.A.I.

# PROGRAMACIÓN EN LENGUAJE ENSAMBLADOR. INSTRUCCIONES BÁSICAS. FORMATOS. DIRECCIONAMIENTOS.

# TEMA 33 INF / SAI: PROGRAMACIÓN EN LENGUAJE ENSAMBLADOR. INSTRUCCIONES BÁSICAS. FORMATOS. DIRECCIONAMIENTOS



Preparador Informática

## 1. INTRODUCCIÓN

Un lenguaje de programación es una herramienta que nos permite crear programas ejecutables para el ordenador. Podemos distinguir en:

- Lenguajes de programación de **bajo nivel**: Son lenguajes totalmente dependientes de la máquina, es decir que el programa que se realiza con este tipo de lenguajes no se pueden migrar o utilizar en otras máquinas. Al estar prácticamente diseñados a medida del hardware, aprovechan al máximo las características del mismo. Dentro de este tipo están el lenguaje máquina y el **lenguaje ensamblador**.
- Lenguajes de **alto nivel**: Son aquellos que se encuentran más cercanos al lenguaje natural que al lenguaje máquina. Se tratan de lenguajes independientes de la arquitectura del ordenador. Por lo que, en principio, un programa escrito en un lenguaje de alto nivel, lo puedes migrar de una máquina a otra sin ningún tipo de problema. Por ejemplo, C ó PASCAL.

En este tema trataremos en profundidad el denominado lenguaje ensamblador, cómo programar en este lenguaje, sus instrucciones básicas, formatos y direccionamientos.

## 2. PROGRAMACIÓN EN LENGUAJE ENSAMBLADOR

### 2.1. LENGUAJE MÁQUINA Y LENGUAJE ENSAMBLADOR.

El ordenador está constituido por una serie de componentes electrónicos, que son gestionados por medio de una serie de instrucciones que determinan qué actividad debe llevar a cabo en cada momento cada uno de los componentes del equipo. Una instrucción es un conjunto de unos y ceros que equivalen a acciones elementales de la máquina que se denomina lenguaje máquina. Este lenguaje fue el utilizado para programar los primeros ordenadores y sigue siendo utilizado para trasladar a los componentes del ordenador las operaciones que deben ejecutar. Tienes dos notables ventajas:

- Es directamente interpretable y ejecutable por el procesador.
- Los programas se adaptan completamente a los elementos y peculiaridades del ordenador, por lo que se pueden desarrollar programas muy eficientes en cuanto a tiempo de ejecución y a consumo de la capacidad de memoria.

No obstante, el lenguaje máquina tiene ciertos inconvenientes que hacen que sea muy incómodo de utilizar:

- Las instrucciones en lenguaje máquina han de ser en binario, o en octal, o en hexadecimal por lo que hace que la redacción del programa sea compleja y el programa resultante sea poco legible.
- El programador ha de realizar la asignación de registros y la asignación de memoria.

Para solucionar estos dos inconvenientes se utiliza el lenguaje ensamblador. Surge como una herramienta para simplificar la programación en instrucciones de máquina. Esta simplificación tiene dos aspectos fundamentales:

- El empleo de códigos nemónicos, para representar las instrucciones.
- El empleo de nombres simbólicos, para designar a los datos y a las referencias.

De esta forma, se libera al programador de tener que especificar posiciones de memoria y de recordar los códigos de cada instrucción.

No obstante, un programa en lenguaje ensamblador no es directamente ejecutable por el procesador, requiriendo previamente una traducción de los nemónicos y nombres simbólicos a los códigos máquina y direcciones correspondientes. En los lenguajes ensambladores cada nombre simbólico corresponde aproximadamente a una instrucción en código máquina, de manera que la traducción es muy sencilla. Esta traducción se realiza por medio de un programa traductor conocido como **ensamblador**.

Aunque los lenguajes ensambladores son distintos de un computador a otro, la mayor parte de ellos sigue una filosofía concreta, por lo que es de una gran utilidad seguir una terminología común; esto es lo que pretende el **estándar IEEE 694** de ensamblador para microprocesadores.

## 2.2. ELEMENTOS BÁSICOS DEL LENGUAJE ENSAMBLADOR.

### A) COMENTARIOS.

El uso de comentarios a lo largo de un programa puede mejorar su claridad, en especial en lenguaje ensamblador, donde el propósito de un conjunto de

instrucciones con frecuencia no es claro. Un comentario empieza con punto y coma (;) y, en donde quiera que lo codifique, el ensamblador supone que todos los caracteres a la derecha de esa línea son comentarios. Ejemplo:

```
1. ; Toda esta línea es un comentario.
2. ADD AX, BX ; Comentario en la misma línea que la instrucción.
```

## B) PALABRAS RESERVADAS.

Ciertas palabras en lenguaje ensamblador están reservadas para sus propósitos propios, y son usadas solo bajo condiciones especiales. Podemos encontrar palabras reservadas entre otras cosas para las instrucciones (MOV, ADD...), para directivas (END, SEGMENT...), para operadores (FAR, SIZE...), etc. El uso de una palabra reservada para un propósito equivocado provoca que el ensamblador genere un mensaje de error.

## C) IDENTIFICADORES.

Un identificador es un nombre que se aplica a elementos en el programa. Los dos tipos de identificadores son: **nombre**, que se refiere a la dirección de un elemento de dato y **etiqueta**, que se refiere a la dirección de una instrucción. Las mismas reglas se aplican tanto para los nombres como para las etiquetas. Un identificador puede usar letras del alfabeto, dígitos y ciertos caracteres especiales. El primer carácter de un identificador debe ser una letra o un carácter especial, excepto el punto. Ya que el ensamblador utiliza algunos símbolos especiales en palabras que inician con el símbolo @, debe evitar usarlo en sus definiciones. El ensamblador trata las letras mayúsculas y minúsculas como iguales. La longitud máxima de un identificador es de 31 caracteres.

## D) INSTRUCCIONES.

Un programa en lenguaje ensamblador consiste en un conjunto de enunciados. Los dos tipos de enunciados son:

- **Instrucciones**, (MOV, ADD...) que el ensamblador traduce a código máquina.
- **Directivas**, que indican al ensamblador que realiza una acción específica, como definir un elemento de dato.

A continuación, está el formato general de un enunciado, en donde los corchetes indican una entrada opcional:

```
[identificador (nombre/etiqueta)] operación [operando(s)] [;comentario]
```

Por ejemplo:

```
Directiva:    COUNT DB 1        ;Nom, Op, Operando
Instrucción: MOV AX, 0          ;Operación, 2 Operando
```

En el siguiente apartado veremos las instrucciones básicas del lenguaje ensamblador más detalladamente.

## 3. INSTRUCCIONES BÁSICAS. FORMATOS.

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. y cada una de ellas da lugar a una única instrucción en lenguaje máquina.

### 3.1. TIPOS DE INSTRUCCIONES

El juego de instrucciones puede ser realmente sencillo. Por ejemplo, la máquina de Turing utiliza solamente las cuatro instrucciones de escribir, mover a la izquierda una posición y leer, mover a la derecha una posición y leer, y parar. Del mismo modo se pueden construir computadores con juegos de instrucciones muy reducidos, aunque desde el punto de vista de los computadores comerciales, lo que tiene interés es que sean rápidos y económicos, por lo que habrá que dotarles de un conjunto bien seleccionado de instrucciones. La selección del juego de instrucciones de un computador, por tanto, es uno de los puntos más críticos de su diseño. A continuación, presentaremos las instrucciones consideradas por el estándar IEEE 694 agrupadas en:

**A) Instrucciones de transferencia de datos:** Las instrucciones de transferencia de datos permiten repetir, en el operando destino, la información almacenada en el operando origen, quedando este último sin modificar. Destino y origen pueden ser registros o posiciones de memoria. Por ejemplo, MOV, PUSH y POP.

**B) Instrucciones que modifican la secuencia del programa:** Las instrucciones de modificación de secuencia de programa permiten alterar la secuencia normal de ejecución del mismo. De forma genérica se dice que son instrucciones de salto o bifurcación, puesto que, en vez de pasar a la instrucción

que ocupa la posición siguiente, "saltan" a ejecutar las instrucciones que se encuentran en otra posición de memoria. Dentro de estas tenemos las de:

- JMP etiqueta (Salto incondicional): Siempre se realiza el salto.
- JXX etiqueta (Salto condicional): Transfieren el control del programa a otro punto sólo si se cumple una condición. Por ejemplo, JZ.
- LOOP etiqueta: Instrucción de tipo iterativo que permite la ejecución de bucles dentro del programa.
- CALL procedimiento: Realiza una llamada a una subrutina lo que provoca que se abandone la ejecución del programa principal, almacenando la instrucción de vuelta en la pila. Para devolver el control, el procedimiento llama en su finalización a la instrucción RET.

**C) Instrucciones aritméticas:** Las instrucciones de cálculo aritmético más comunes son: ADD, SUB, MUL...

**D) Instrucciones lógicas:** Las instrucciones lógicas, más usuales, son AND, OR, NOT, y XOR. Conviene recordar que las operaciones lógicas se realizan en cada uno de los bits de los operandos de forma independiente y que modifican los bits de estado.

**E) Instrucciones de desplazamiento:** Las operaciones de desplazamiento modifican los bits de estado, siendo las más utilizadas SHIFT y ROTATE.

**F) Instrucciones de bit:** El grupo de las instrucciones de bit incluye las tres instrucciones siguientes: BIT TEST, BIT SET, y BIT CLEAR.

**G) Instrucciones de entrada/salida:** Las instrucciones de entrada/salida son en realidad instrucciones de transferencia, con la peculiaridad de que el destino o el origen es un registro de un periférico. Muchos computadores no tienen este tipo de instrucciones, realizándose estas funciones con las instrucciones de LOAD, STORE y/o MOVE. Las instrucciones más usuales son IN y OUT.

**H) Otras instrucciones:** Existen una serie de instrucciones que no se pueden clasificar en ninguna de las categorías anteriores. Entre ellas se pueden destacar WAIT, CMP, HALT, CONVERT, ENTER, EXIT...

## 3.2. FORMATOS

En el formato general del lenguaje ensamblador cada instrucción está compuesta de 4 campos cómo hemos visto anteriormente:

```
[etiqueta] operación [operando(s)] [;comentarios]
```

- El campo **etiqueta** debe seguir las reglas explicadas para los identificadores.
- El campo **operación** contiene el nemotécnico de la instrucción, que es de 2 a 6 caracteres.
- El campo **operando** contiene la posición o posiciones donde están los datos que van a ser manipulados por la instrucción.
- El campo **comentario** (opcional) se utiliza para propósitos de documentación.

Veamos un ejemplo de instrucción en lenguaje ensamblador:

| Código Operación | Operando 1 | Operando 2 |
|---|---|---|
| MOV | A, | B |

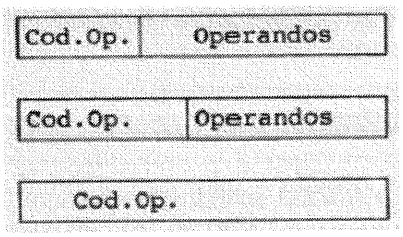
y su traducción a lenguaje máquina:

| Código Operación | | | | Operando 1 | | | | | | Operando 2 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

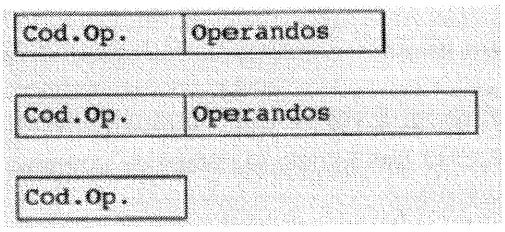
Una instrucción puede tener distintos formatos dependiendo de las longitudes:

Preparador Informática

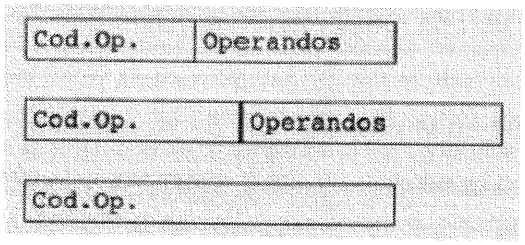
- Longitud de instrucción fija, código de operación extendido.




- Longitud de instrucción variable, código de operación fijo.



- Longitud de instrucción variable, código de operación extendido.




También puede tener distintos formatos dependiendo del número de operandos o direcciones que utilicen en la instrucción:

- Formato de cuatro direcciones: Dos indican los operandos, otra dirección donde guardar el resultado y la otra indica dónde se encuentra la siguiente instrucción a ejecutar. Actualmente no es habitual encontrar máquinas con este formato.

| Código Operación | Operando 1 | Operando 2 | Resultado | Siguiente instrucción |
|---|---|---|---|---|

- Formato de tres direcciones: Al aparecer el contador del programa, que es el registro donde se guarda la dirección de la siguiente instrucción a ejecutar, ya no sería necesario incluirla en la instrucción.

| Código Operación | Operando 1 | Operando 2 | Resultado |
|---|---|---|---|

Preparador Informática

- Formato de dos direcciones: Posteriormente se optó por eliminar la dirección que indica donde se guarda el resultado, empleándose para este fin uno de los operandos especificados en instrucción.

| Código Operación | Operando 1 | Operando 2 |
|---|---|---|

- 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 supones que es un determinado registro de trabajo de la CPU, denominado generalmente acumulador.

- Formato de cero direcciones: En las máquinas que utilizan para guardar datos una estructura de pila y usan el registro llamado "apuntador de pila" sólo es necesario especificar la operación ya que la CPU supone que los operandos están guardados en la pila, adonde los va a buscar con ayuda del apuntador de pila y cuando termina la operación supone que el resultado debe guardarlo en la misma pila.

## 4. DIRECCIONAMIENTOS

Llamamos modos de direccionamiento a las diferentes formas de indicar en el campo de operandos dónde se encuentran los datos que la CPU tendrá que procesar. Los distintos modos de direccionamiento que podemos encontrar son:

### A) Modo de direccionamiento Inmediato:

El direccionamiento se llama inmediato cuando el objeto, en este caso un operando, se encuentra contenido en la propia instrucción. 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 de la memoria principal. No permite modificar el dato sin modificar la instrucción.

### B) Modo de direccionamiento directo:

Un direccionamiento se llama directo cuando expresa la dirección real del objeto. Es un modo de direccionamiento más lento que el inmediato, pero puede modificarse el valor del dato sin modificar la instrucción. Este tipo de direccionamiento presenta tres alternativas:

- La información contenida en la instrucción puede ser el identificador de un registro. En algunos textos se considera este caso como otro tipo de direccionamiento que se denomina direccionamiento de registro.
- La información contenida en la instrucción es una dirección completa de memoria principal.
- La información contenida en la instrucción es una dirección limitada, que permite referirse solamente a una parte del mapa de memoria. Este tipo de direccionamiento suele llamarse direccionamiento de página base y es muy frecuente en los microprocesadores.

### C) Modo de direccionamiento indirecto:

El direccionamiento indirecto comienza con un direccionamiento directo, pero se diferencia de aquél en que la dirección obtenida no apunta al objeto deseado sino su dirección. Por tanto, para obtener el objeto deseado se requiere un acceso adicional a memoria principal. Con esto 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.

### D) Modo de direccionamiento indexado:

En este modo la instrucción no contiene la dirección del objeto, sino un desplazamiento D sobre una dirección marcada por un puntero. La dirección se calcula sumando el desplazamiento D al puntero de referencia, que suele estar almacenado en un registro de la CPU.

El registro que sirve de puntero puede ser del tamaño deseado. Por ello, este direccionamiento es más compacto, pero requiere realizar la operación de suma del puntero más el desplazamiento. La mayoría de los computadores permiten desplazamientos positivos y negativos.

## 5. CONCLUSIÓN

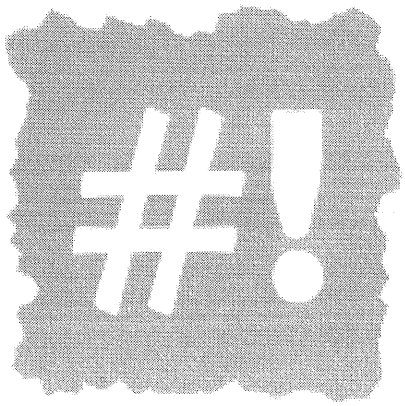
En este tema he podido mostrar como las dificultades que ofrecía el lenguaje máquina dio lugar a la aparición del lenguaje ensamblador mejorando notablemente la programación. Hemos visto cómo sus instrucciones mejoraban y facilitaban mucho la labor del programador, pero aun así el lenguaje ensamblador mantenía la mayoría de los inconvenientes del lenguaje máquina (repertorio muy reducido de instrucciones, formato rígido, baja portabilidad, fuerte dependencia del hardware...). Por todos estos problemas la evolución de los lenguajes de programación no se detuvo y fueron apareciendo nuevos lenguajes cada vez más completos, sofisticados y, sobre todo, más fácilmente comprensibles pero que a pesar de todo para poder ejecutarse necesitan ser traducidos previamente a un lenguaje máquina interpretable por el ordenador.

El uso actual del lenguaje ensamblador se limita a tareas muy específicas en ámbitos en los que se requiere un control muy exhaustivo del hardware o se deben ejecutar tareas de forma muy eficiente, como los programas de arranque

del ordenador, ejecutables independientes para dispositivos muy específicos, drivers, etc.

## 6. BIBLIOGRAFÍA


- Prieto, A. y otros. **Introducción a la informática**. Editorial McGraw-Hill.
- Abel, P. **Lenguaje ensamblador y programación para PC IBM y compatibles.** Editorial Prentice Hall
- https://standards.ieee.org/standard/694-1985.html (IEEE, Institute of Electrical and Electronics Engineers)

Preparador Informática