Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

**Tareas**

**Portafolio de tareas**

Nombre: **Martinez Reyes Fernando**

Profesor: **Dr. García Ruiz Alejandro Humberto**

Asignatura: **Programación de Microprocesadores**

**8**vo. Semestre – Grupo “**I**”

2022-1

**ÍNDINCE**

[1.- EL CARGADOR 2](#_Toc114245746)

[2.- CARACTERISTICA DE UN LENGUAJE ENSAMBLADOR 4](#_Toc114245747)

[3.- LENGUAJE DE BAJO NIVEL VS ALTO NIVEL 4](#_Toc114245748)

[4.- ¿POR QUÉ CONTINÚA UTILIZÁNDOSE EL LENGUAJE ENSAMBLADOR? 6](#_Toc114245749)

[5.- CINCO TRABAJOS Y SUS CARACTERÍSTICAS EN LAS QUE SE UTILICE EL LENGUAJE ENSAMBLADOR 8](#_Toc114245750)

[6.- ¿QUÉ ES NEMOTÉCNICO? 9](#_Toc114245751)

[7.- DIFERENCIA ENTRE ENSAMBLADOR Y MACROENSAMBLADOR 10](#_Toc114245752)

[8.- Tipos de mnemotécnicos 10](#_Toc114245753)

[9.- CODIGO ASCII 12](#_Toc114245754)

[10.- LITTLE ENDIAN AND BIGENDIAN 13](#_Toc114245755)

[11.- COMPLEMENTO A16 15](#_Toc114245756)

# 1.- EL CARGADOR

**Loader:**  es un programa especial que toma la entrada de archivos ejecutables del enlazador, los carga en la memoria principal y prepara este código para su ejecución por computadora. El cargador asigna espacio de memoria al programa. Incluso se establece referencia simbólica entre objetos. Se encarga de cargar programas y bibliotecas en el sistema operativo. Los sistemas informáticos integrados no tienen cargadores. En ellos, el código se ejecuta a través de ROM. Hay varios esquemas de carga siguientes:

1. Cargadores absolutos

2. Reubicación de cargadores

3. Cargadores de enlace directo

4. Cargadores de arranque

Las diferencias entre Linker y Loader son las siguientes:

| ENLACE | CARGADOR |
| --- | --- |
| La función principal de Linker es generar archivos ejecutables. | Mientras que el objetivo principal de Loader es cargar archivos ejecutables en la memoria principal. |
| El enlazador toma la entrada del código objeto generado por el compilador/ensamblador. | Y el cargador toma la entrada de archivos ejecutables generados por el enlazador. |
| La vinculación se puede definir como el proceso de combinar varias piezas de códigos y código fuente para obtener un código ejecutable. | La carga se puede definir como el proceso de cargar códigos ejecutables en la memoria principal para su posterior ejecución. |
| Los enlazadores son de 2 tipos: Editor de enlaces y Enlazador dinámico. | Los cargadores son de 4 tipos: Absoluto, Reubicación, Vinculación directa, Bootstrap. |
| Otro uso del enlazador es combinar todos los módulos de objetos. | Ayuda a asignar la dirección a códigos/archivos ejecutables. |
| Linker también es responsable de organizar los objetos en el espacio de direcciones del programa. | Loader también es responsable de ajustar las referencias que se utilizan dentro del programa. |

# 2.- CARACTERISTICA DE UN LENGUAJE ENSAMBLADOR

Las características del lenguaje ensamblador se mencionan a continuación:

1. Puede usar código de operación mnemotécnico que numérico, y también proporciona la información de cualquier error en el código.
2. Este lenguaje ayuda a especificar el operando simbólico, lo que significa que no necesita especificar la dirección de la máquina de ese operando. Se puede representar en forma de símbolo.
3. Los datos se pueden declarar usando notación decimal.

# 3.- LENGUAJE DE BAJO NIVEL VS ALTO NIVEL

Diferencia entre lenguajes de alto y bajo nivel

|  |  |  |
| --- | --- | --- |
| **Parámetro** | **Lenguaje de alto nivel** | **Lenguaje de bajo nivel** |
| Básico | Estos son lenguajes amigables para los programadores que son manejables, fáciles de entender, depurar y ampliamente utilizados en la actualidad. | Estos son lenguajes amigables con las máquinas que son muy difíciles de entender por los seres humanos pero fáciles de interpretar por las máquinas. |
| Facilidad de ejecución | Estos son muy fáciles de ejecutar. | Estos son muy difíciles de ejecutar. |
| Proceso de Traducción | Los lenguajes de alto nivel requieren el uso de un compilador o un intérprete para su traducción al código máquina. | El lenguaje de bajo nivel requiere un ensamblador para traducir directamente las instrucciones del lenguaje máquina. |
| Eficiencia de la memoria | Estos lenguajes tienen una eficiencia de memoria muy baja. Significa que consumen más memoria que cualquier lenguaje de bajo nivel. | Estos lenguajes tienen una eficiencia de memoria muy alta. Significa que consumen menos energía en comparación con cualquier lenguaje de alto nivel. |
| Portabilidad | Estos son portátiles de cualquier dispositivo a otro. | Un usuario no puede transferirlos de un dispositivo a otro. |
| comprensibilidad | Los lenguajes de alto nivel son amigables para los humanos. Son, por tanto, muy fáciles de entender y aprender por cualquier programador. | Los lenguajes de bajo nivel son aptos para máquinas. Son, por tanto, muy difíciles de entender y aprender por cualquier ser humano. |
| Dependencia de las máquinas | Los lenguajes de alto nivel no dependen de las máquinas. | Los lenguajes de bajo nivel dependen de la máquina y, por lo tanto, son muy difíciles de entender para un usuario normal. |
| depuración | Es muy fácil depurar estos lenguajes. | Un programador no puede depurar fácilmente estos lenguajes. |
| Mantenimiento | Los lenguajes de alto nivel tienen una técnica de mantenimiento simple y completa. | Es bastante complejo mantener cualquier lenguaje de bajo nivel. |
| Uso | Los lenguajes de alto nivel son muy comunes y ampliamente utilizados para la programación en los tiempos actuales. | Los lenguajes de bajo nivel no son muy comunes hoy en día para la programación. |
| Velocidad de ejecución | Los lenguajes de alto nivel tardan más en ejecutarse en comparación con los lenguajes de bajo nivel porque requieren un programa de traducción. | La velocidad de traducción de los lenguajes de bajo nivel es muy alta. |
| Abstracción | Los lenguajes de alto nivel permiten una mayor abstracción. | Los lenguajes de bajo nivel permiten muy poca o ninguna abstracción. |
| Necesidad de hardware | Uno no requiere un conocimiento de hardware para escribir programas. | Tener conocimientos de hardware es un requisito previo para escribir programas. |
| Instalaciones proporcionadas | Los lenguajes de alto nivel no proporcionan varias facilidades a nivel de hardware. | Los lenguajes de bajo nivel están muy cerca del hardware. Ayudan a escribir varios programas a nivel de hardware. |
| Facilidad de modificación | El proceso de modificación de programas es muy difícil con programas de alto nivel. Es porque cada declaración en él puede ejecutar un montón de instrucciones. | El proceso de modificación de programas es muy fácil en programas de bajo nivel. Aquí, puede asignar directamente las declaraciones a las instrucciones del procesador. |
| Ejemplos | Algunos ejemplos de lenguajes de alto nivel incluyen Perl, BASIC, COBOL, Pascal, Ruby, etc. | Algunos ejemplos de lenguajes de bajo nivel incluyen el lenguaje de máquina y el lenguaje ensamblador. |

# 4.- ¿POR QUÉ CONTINÚA UTILIZÁNDOSE EL LENGUAJE ENSAMBLADOR?

A pesar de la prevalencia de [los lenguajes de alto nivel](https://www.techopedia.com/definition/3925/high-level-language-hll) que se utilizan principalmente para el desarrollo de aplicaciones y programas de software, no se puede subestimar la importancia del lenguaje ensamblador en el mundo actual. Un [programador](https://www.techopedia.com/definition/4813/programmer) aún puede ganar mucho si puede aprender a codificar en lenguaje ensamblador e implementarlo. En estos días, el lenguaje ensamblador permite manipular el hardware directamente, abordar problemas críticos relacionados con el rendimiento y también brindar acceso a instrucciones especiales para los procesadores. Los usos del lenguaje ensamblador incluyen controladores de dispositivos de codificación, sistemas en tiempo real, sistemas integrados de bajo nivel, códigos de arranque, ingeniería inversa y más.

**Control total sobre los recursos de un sistema.**

Como programador, lo más cerca que puede estar del [procesador](https://www.techopedia.com/definition/28254/processor) de una máquina es usando lenguaje ensamblador. Aquí, puede escribir código para acceder a los registros e incluso manejar [direcciones de memoria](https://www.techopedia.com/definition/323/memory-address) directamente para recuperar valores y punteros. Entonces, si está escribiendo un programa que tiene un gran [algoritmo](https://www.techopedia.com/definition/3739/algorithm) , se beneficiará enormemente. Esto se debe principalmente a que el lenguaje ensamblador es la puerta de entrada a la optimización en velocidad, ofreciendo así una gran eficiencia y rendimiento.

**Comprender la función del procesador y la memoria**

Si está escribiendo un programa destinado a ser un compilador o un controlador de dispositivo, entonces una comprensión completa de la función del procesador es una gran ventaja. Entonces, en este caso, la mejor opción es escribir algo de código en lenguaje ensamblador y ver cómo funciona el procesador y la memoria. Sin embargo, una cosa a tener en cuenta es que el lenguaje ensamblador es simbólico, por lo que puede parecer críptico. Además, el código fuente en lenguaje ensamblador siempre es más grande que el de un lenguaje de alto nivel. Sin embargo, dedicar tiempo y esfuerzo para dominarlo puede beneficiar enormemente a uno en términos de comprensión.

**Acceso directo al hardware**

El lenguaje ensamblador es el único lenguaje que habla directamente a la computadora/máquina. Es el idioma que reconoce una determinada [CPU](https://www.techopedia.com/definition/2851/central-processing-unit-cpu) y diferentes CPU reconocen diferentes tipos de ellos. Sin embargo, dado que cada sección del binario tiene un cierto significado, puede ser algo fácil de comprender.

**El lenguaje ensamblador es transparente.**

En comparación con los lenguajes de alto nivel, que se encuentran principalmente en forma de tipos de datos abstractos, el lenguaje ensamblador es simple y transparente. Esto se debe en gran parte a que tiene un pequeño número de operaciones. Por lo tanto, esto es muy útil para el análisis de algoritmos, que consiste en semántica y flujo de control. También facilita la [depuración](https://www.techopedia.com/definition/16373/debugging) , ya que es menos complejo. En general, hay menos gastos generales en comparación con los lenguajes de alto nivel.

# 5.- CINCO TRABAJOS Y SUS CARACTERÍSTICAS EN LAS QUE SE UTILICE EL LENGUAJE ENSAMBLADOR

El lenguaje ensamblador se usa a menudo cuando:

* [**Controladores de dispositivos**](https://en.wikipedia.org/wiki/Device_driver)**de construcción:**Cuando necesite acceder directamente al hardware y realizar una comunicación de bajo nivel.
* [**depuración**](https://msdn.microsoft.com/en-us/library/windows/hardware/ff540661(v=vs.85).aspx)**código nativo compilado:**cuando su código se compila como una versión de lanzamiento, se optimiza y se ejecuta en una plataforma de destino; conocer el ensamblaje a menudo ayuda a depurar bloqueos y errores cuando las técnicas normales de depuración no están disponibles.  
  También es relevante dónde falla su programa, pero dentro de una biblioteca estándar o una función del sistema operativo a la que ha llamado. no siempre tiene el código fuente del sistema operativo o la biblioteca estándar, por lo que es realmente útil comprender el código de ensamblaje subyacente.
* [**Ingeniería inversa**](http://resources.infosecinstitute.com/x86-assembly-language-applicable-to-reverse-engineering-the-basics-part-1/)**-**Si está tratando de comprender el código compilado de otra persona, o está tratando de modificar el comportamiento del código existente, necesitará mucha comprensión del lenguaje ensamblador.
* **Investigación de seguridad:**a menudo relacionada con la ingeniería inversa. Los investigadores de seguridad utilizan su comprensión del ensamblado para encontrar vulnerabilidades en proyectos existentes o para analizar e investigar malware.
* [**Optimización del rendimiento**](http://stackoverflow.com/questions/577554/when-is-assembler-faster-than-c)**-**una comprensión profunda del hardware subyacente y de cómo un procesador ejecuta el código en lenguaje ensamblador puede permitir a los programadores ejecutar código crítico. Si echa un vistazo al código fuente de las bibliotecas estándar, a menudo puede crear código ensamblador en línea.  
  *memcpy*es un ejemplo de una función de biblioteca estándar que casi siempre se implementa mediante ensamblaje en línea.  
  Sin embargo, tenga en cuenta que el compilador puede funcionar muy bien y puede resultar muy difícil codificar algo más rápido en el ensamblaje si no tiene mucha experiencia con el proceso y el hardware que está utilizando.
* **Construyendo**[**compiladores**](https://en.wikipedia.org/wiki/Compiler)**-**Los compiladores se utilizan para convertir el lenguaje de programación de alto nivel en código de máquina. Esto es cierto para el código C/C++ que se compila una vez y se distribuye en código de máquina binario, y es cierto para el código Javascript/Python que a menudo se puede ejecutar con un [compilador Just in Time.](https://en.wikipedia.org/wiki/Just-in-time_compilation)(PyPy, V8).  
  [Cuando se crean](https://en.wikipedia.org/wiki/Go_(programming_language)) nuevos lenguajes de programación, cuando se actualizan los sistemas operativos o cuando se lanzan nuevas CPU: grandes equipos de desarrolladores ayudan a crear compiladores para estos objetivos.
* **Escritura**[**de sistemas operativos**](https://en.wikipedia.org/wiki/Operating_system)**-**Los sistemas operativos modernos tienen mucho código C/C++, pero también requieren un conocimiento profundo del hardware subyacente. Tareas comunes como cambio de contexto, administración de memoria, funciones de seguridad, manejo de interrupciones, memoria virtual, E/S y acceso al disco, todas requieren código ensamblador de bajo nivel.

# 6.- ¿QUÉ ES NEMOTÉCNICO?

Un mnemotécnico es un término, símbolo o nombre utilizado para definir o especificar una función informática. Los mnemotécnicos se utilizan en informática para proporcionar a los usuarios un medio para acceder rápidamente a una función, servicio o proceso, evitando el método real más largo utilizado para realizarlo o lograrlo. El lenguaje ensamblador también usa un mnemotécnico para representar la operación de la máquina o el código de operación.

En la programación de computadoras, a las funciones de programación de rutina se les asigna un mnemotécnico que es más corto pero proporciona la misma funcionalidad que la función original.  
  
En lenguaje ensamblador, los mnemotécnicos se utilizan para especificar un código de operación que representa una instrucción de lenguaje de máquina completa y operativa. Esto luego es traducido por el ensamblador para generar el código objeto. Por ejemplo, el mnemotécnico MOV se usa en lenguaje ensamblador para copiar y mover datos entre registros y ubicaciones de memoria.

# 7.- DIFERENCIA ENTRE ENSAMBLADOR Y MACROENSAMBLADOR

**Ensamblador:** es un programa que convierte un lenguaje de bajo nivel en código de máquina, y existe una correspondencia uno a uno entre las declaraciones del lenguaje de origen y las instrucciones de la máquina.

**Ensamblador de macros:** realiza la misma tarea que el ensamblador, pero a veces hay una correspondencia de uno a muchos entre las instrucciones del lenguaje de origen y las instrucciones de la máquina.

# 8.- Tipos de mnemotécnicos

**Tipos de técnicas mnemotécnicas**

Aquí hay ocho tipos de técnicas mnemotécnicas que puede usar:

1. Nemotécnicas de ortografía
2. Funciones mnemotécnicas
3. Mnemotécnicas que riman
4. Nota mnemotécnica de organización
5. Nemónicos de aliteración
6. Nemónicos de canciones
7. Nemónicos de organización
8. Nemónicos visuales

**1. Nemotécnicas de ortografía**

Los mnemotécnicos de ortografía lo ayudan a recordar cómo deletrear una palabra difícil con patrones, frases o reglas. Por ejemplo, para memorizar la ortografía de *"separar"* , puede usar la frase *"hay una rata en separado".*Otro mnemotécnico de ortografía común para recordar que *"I"* viene antes de *"E"* en muchas palabras es *"Nunca creas una mentira".*

**2. Funciones mnemotécnicas**

Un tipo de entrenamiento de la memoria mnemotécnica consiste en identificar visualmente una característica destacada de una persona con la que te encuentras por primera vez. Esta técnica te ayuda a asociar mejor su nombre y rostro. Por ejemplo, podría conocer a una nueva colega llamada Daniela, que tiene grandes ojos azules. Puede recordarla como *"Daniela de ojos azules"* para conectar una característica sobre su apariencia con su nombre para que pueda recordarla rápidamente cuando se reencuentre.

**3. Mnemotécnicas de rimas**

Otra técnica mnemotécnica común es usar rimas para memorizar información. Un ejemplo bien conocido de una regla mnemotécnica que rima es la frase *"En mil cuatrocientos noventa y dos, Colón navegó por el océano azul".*La información que desea memorizar es la fecha en que Colón comenzó sus viajes y saber que rima con *"océano azul"* le ayuda a recordar *"1492".*Puede aplicar esta técnica a cualquier información que necesite recordar en el trabajo.

**4. Tenga en cuenta los mnemotécnicos de organización**

Algunas personas aprenden mejor cuando organizan sus pensamientos en notas. Esta es una excelente técnica mnemotécnica para usar en su trabajo, ya sea que se esté preparando para una presentación o necesite memorizar nuevos datos. Por ejemplo, puede convertir las ideas principales en preguntas y escribirlas en un lado de una tarjeta mientras agrega las respuestas en el lado opuesto. Entrenas a tu cerebro para ver preguntas y recordar las respuestas cuando haces esto repetidamente.

**5. Nemónicos de aliteración**

Cuando desea recordar el nombre de una persona, la aliteración es una herramienta conveniente. Si conoces a una colega llamada Sabina, puedes pensar en otras palabras que comiencen con la letra *"S"* que la describan. Usted puede encontrar que Sabina es sofisticada, sincera y con estilo. Cuando necesite recordar su nombre, recordará las tres palabras que asocia con ella y podrá limitar su nombre a una que comience con la letra *"S".*

**6. Nemónicos de canciones**

Algunas personas aprenden mejor cuando pueden cantar, por lo que puede insertar la información que desea memorizar en una canción. Un ejemplo popular es la canción *“ABC”* que usan los escolares para aprender el alfabeto. Cuando canten esto repetidamente, comenzarán a recordar el orden del alfabeto. Puedes hacer la misma técnica mnemotécnica en el trabajo poniendo nueva información en un formato de canción.

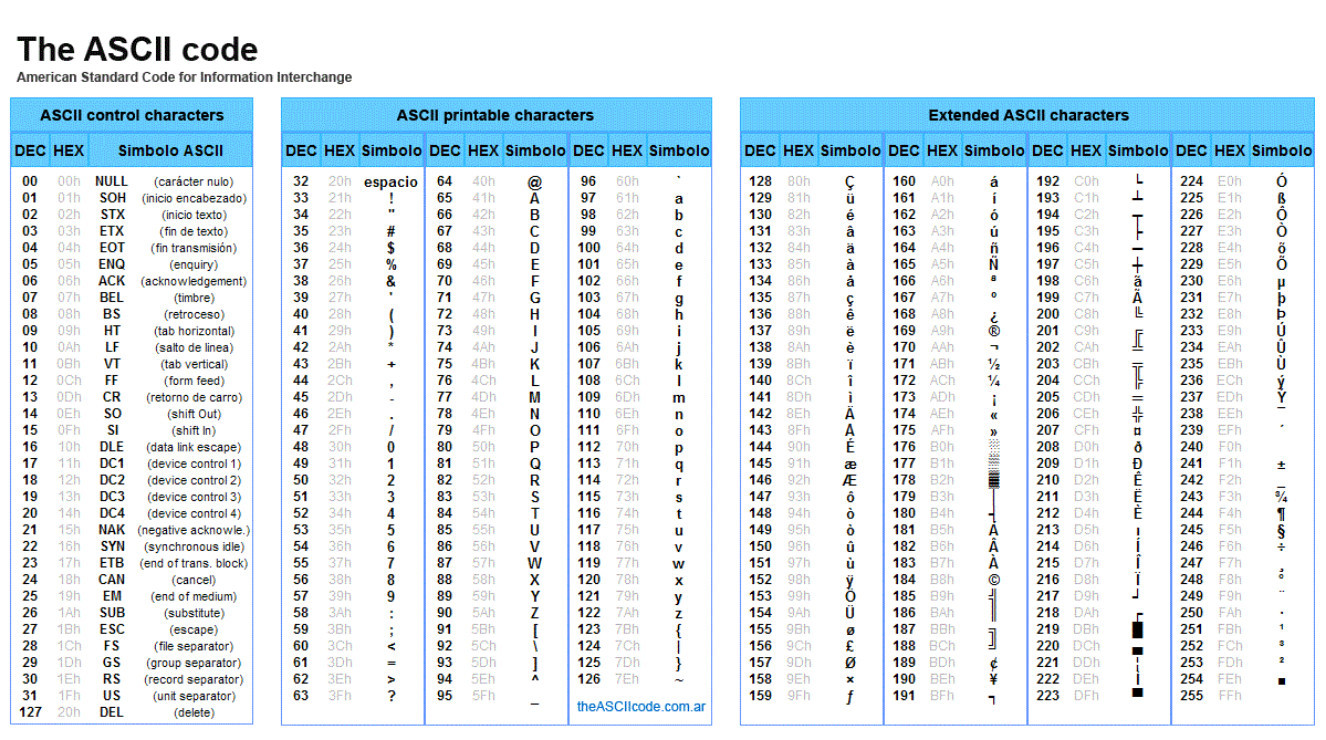
**7. Nemotecnia de la organización**

Agrupar la información le ayuda a recordarla más fácilmente. Si tiene un gran grupo de palabras o números que necesita memorizar, puede dividirlos para recordarlos rápidamente. Por ejemplo, si necesita recordar los números *"456159753481"* , puede dividirlos en grupos más pequeños. Si los agrupa en *"4561 5975 3481",* es posible que tenga más posibilidades de recordarlos.

**8. Nemotécnicas visuales**

Vincular imágenes implica crear una historia visual para conectar la información que necesita memorizar. Cada ítem te lleva a recordar el siguiente artículo. Por ejemplo, es posible que deba recordar traer su computadora portátil, anteojos para leer, bloc de notas y bolígrafo a su próxima reunión. Puede crear una historia corta para vincular estos elementos para que no olvide ninguno de ellos.

# 9.- CODIGO ASCII



# 10.- LITTLE ENDIAN AND BIGENDIAN

Endianness es un término que describe el orden en que se almacena una secuencia de  [bytes](https://www.techtarget.com/searchstorage/definition/byte)  en la memoria de la computadora. Endianness puede ser grande o pequeño, y los adjetivos se refieren a qué valor se almacena primero.

Big-endian es un orden en el que el "extremo grande" (el valor más significativo de la secuencia) se almacena primero, en la dirección de almacenamiento más baja. Little-endian es un orden en el que el "pequeño extremo" (valor menos significativo en la secuencia) se almacena primero.

**El caso de big-endian**

En una computadora big-endian, los dos bytes necesarios para el  número [hexadecimal](https://www.techtarget.com/whatis/definition/hexadecimal)  4F52 se almacenarían como 4F52 en almacenamiento. Por ejemplo, si 4F se almacena en la dirección de almacenamiento 1000, 52 estará en la dirección 1001. En un sistema little-endian, se almacenaría como 524F, con 52 en la dirección 1000 y 4F en 1001.

Para las personas que usan lenguajes que se leen de izquierda a derecha, big-endian parece la forma natural de pensar en almacenar una cadena de caracteres o números, en el mismo orden en que espera que se le presenten. De esta manera, mucha gente considera big-endian como el almacenamiento de algo  *hacia adelante*  , tal como lo leen.

**El caso de little-endian**

Un argumento a favor del orden little-endian es que a medida que aumenta un valor numérico, es posible que deba agregar dígitos a la izquierda. Por ejemplo, un número no exponencial más alto tiene más dígitos. Por lo tanto, una suma de dos números a menudo requiere mover todos los dígitos de un número ordenado big-endian almacenado. Esta adición mueve todo a la derecha.

En un número almacenado de forma little-endian, los bytes menos significativos pueden permanecer donde están. Los nuevos dígitos se agregan a la derecha en una dirección más alta. Esto significa que algunas operaciones informáticas pueden ser más sencillas y rápidas de realizar.

**Orden de bits**

Tenga en cuenta que dentro de los órdenes de bytes big-endian y little-endian, los bits dentro de cada byte son big-endian. Esto significa que el flujo de bits representado por un número determinado de bytes almacenados no intenta ser big-endian o little-endian. Por ejemplo, si el hexadecimal 4F se almacena primero o último con otros bytes en un rango de direcciones de almacenamiento determinado, el orden de bits dentro del byte será 01001111.

Es  *posible*  ser big-endian o little-endian sobre el orden de los bits. Pero las unidades centrales de procesamiento ( [CPU](https://www.techtarget.com/whatis/definition/processor) ) y los programas casi siempre están diseñados para un orden de bits big-endian. En la transmisión de datos, sin embargo, es posible tener cualquier orden de bits.

Los expertos observan que las direcciones de nombres de dominio de Internet y las direcciones de correo electrónico son little-endian. Por ejemplo, una versión big-endian de nuestra dirección de nombre de dominio sería:

com.techtarget.www

**Usos de big-endian y little-endian**

Tanto big-endian como little-endian se utilizan ampliamente en la electrónica digital. La CPU normalmente determina el endianness en uso.

Los mainframes 370 de IBM, las computadoras basadas en la mayoría de las computadoras con conjunto de instrucciones reducido ( [RISC](https://www.techtarget.com/whatis/definition/RISC) ) y los microprocesadores de Motorola utilizan el enfoque big-endian. El Protocolo de control de transmisión/Protocolo de Internet ( [TCP/IP](https://www.techtarget.com/searchnetworking/definition/TCP-IP) ) también utiliza el enfoque big-endian. Por esta razón, big-endian a veces se denomina  *orden de red* .

Por otro lado, los procesadores Intel, los DEC Alpha y al menos algunos programas que se ejecutan en ellos son little-endian.

También hay formas mixtas de endianness. Por ejemplo, el punto flotante VAX usa mixed-endian, también conocido como *middle-endian* . El orden de los bytes en una palabra de 16 bits difiere del orden de las palabras de 16 bits dentro de una palabra de 32 bits. Los procesadores bi-endian pueden operar en modo little-endian o big-endian y cambiar entre los dos.

Los compiladores de lenguajes, como  [Java](https://www.theserverside.com/definition/Java)  o FORTRAN, deben saber de qué manera se almacenará el código objeto que desarrollan. Los convertidores se pueden usar para cambiar un tipo de endian a otro cuando sea necesario.

**Historia de la endianidad**

En 1980, el científico informático Danny Cohen aplicó los términos *big-endian* y *little-endian* a la electrónica digital en su artículo "Sobre las guerras santas y una súplica por la paz".

Big-endian y little-endian se derivan de Los  *viajes de Gulliver de Jonathan Swift,* en el que los Big Endians eran una facción política que rompía sus huevos en el extremo grande, o "de la manera primitiva". Se rebelaron contra el rey liliputiense, que exigió a sus súbditos, los Little Endians, que les rompieran los huevos por el extremo más pequeño.

Si bien muchas computadoras centrales son big-endian, la mayoría de las computadoras modernas son little-endian.

Endianness es inicialmente una decisión arbitraria del proveedor de semiconductores que puede tener un efecto a largo plazo en una línea de productos. Cuando los proveedores actualizan su tecnología, mantienen el endianness existente para ayudar a mantener la compatibilidad con versiones anteriores. Por ejemplo, los diseñadores del Motorola 68000 y el Intel 8086 (predecesor de la familia x86) eligieron su endianness en la década de 1970 y continúan usando sus respectivos endianness en la actualidad.

# 11.- COMPLEMENTO A16

Porque la computadora no sabe cómo restar, pero la computadora puede sumar 2 números. Cuando queremos sumar un número negativo (-ve), se producirá un problema y la computadora solo puede calcular significa que solo puede realizar operaciones de suma, por eso usamos el complemento de 1 y 2 para cambiar el valor negativo a positivo, luego la computadora puede realizar operaciones de suma.

es decir, 7–5 = 7+(-5)

7+(complemento a 2 de 5) siempre usamos representación de 4 bits porque la computadora almacena un número en forma hexadecimal

7= 0111. ,

5 = 0101

Complemento a 1 1010+1 = 1011

Entonces la computadora puede realizar sumas como

0111+1011 = 1 0010 (el primer 1 descartado)

Tenemos 10 = 2.

También en la multiplicación usamos el algoritmo de multiplicación de Booth, pero la computadora no realiza la multiplicación, sino que la realiza como una suma múltiple.

es decir, 2\*3 = 2+2+2.

El complemento a 16 es un método para representar números negativos en hexadecimal (base 16).

Para formar el complemento de 16 de un número hexadecimal como 0x1234, resta cada dígito de 0xF y luego agrega 1 a todo. Restar de 0xF (15) forma el complemento de 15 y luego sumarle uno forma el complemento de 16. (¿Consíguelo?)

Entonces, 0xFFFF - 0x1234 es 0xEDCB y al agregar 1 se obtiene 0xEDCC, que es la representación en complemento de 16 de 0x1234 negativo.

Existe una representación en complemento de b para cada base b. La belleza del complemento de b es que para calcular AC en base b, usted forma el complemento de b de C y luego SUMA en lugar de restar.

En cuanto a por qué se usa, bueno, no lo es. Las computadoras usan el complemento a 2 porque son binarios, no (normalmente) hexadecimales. Sucede que si representas números hexadecimales en binario, los patrones de bits para el complemento de 16 son exactamente los mismos que los patrones de bits para el complemento de 2, pero nadie dice "mi computadora usa el complemento de 16".

El complemento a dos hace que sea particularmente fácil construir sumadores y restadores. Para sumar, ingresa A y C, SUMA y obtiene A+C. Para restar, ingresa A y ~B y establece el bit CARRY IN, SUMA y obtiene AB. ~B significa B con todos los bits invertidos.