1. Lenguajes de programación.

Como hemos visto, en todo el proceso de resolución de un problema mediante la creación de software, después del análisis del problema y del diseño del algoritmo que pueda resolverlo, es necesario traducir éste a un lenguaje que exprese claramente cada uno de los pasos a seguir para su correcta ejecución. Este lenguaje recibe el nombre de lenguaje de programación.

Lenguaje de programación: conjunto de reglas sintácticas y semánticas, símbolos y palabras especiales establecidas para la construcción de programas. Es un lenguaje artificial, una construcción mental del ser humano para expresar programas.

Gramática del lenguaje: reglas aplicables al conjunto de símbolos y palabras especiales del lenguaje de programación para la construcción de sentencias correctas.

Léxico: es el conjunto finito de símbolos y palabras especiales, es el vocabulario del lenguaje.

Sintaxis: son las posibles combinaciones de los símbolos y palabras especiales. Está relacionada con la forma de los programas.

Semántica: es el significado de cada construcción del lenguaje, la acción que se llevará a cabo.

Hay que tener en cuenta que pueden existir sentencias sintácticamente correctas, pero semánticamente incorrectas. Por ejemplo, *"Un semáforo canta un zapato"* está bien construida sintácticamente, pero es evidente que semánticamente no.

Una característica relevante de los lenguajes de programación es, precisamente, que más de un programador pueda usar un **conjunto común** de instrucciones que sean comprendidas entre ellos. A través de este conjunto se puede lograr la **construcción** de un programa de forma **colaborativa**.

Los **lenguajes** de programación pueden ser **clasificados** en función de lo cerca que estén del **lenguaje humano o del lenguaje** de los **computadores**. El lenguaje de los computadores son códigos binarios, es decir, secuencias de unos y ceros. Detallaremos seguidamente las características principales de los lenguajes de programación.

1.1. Lenguaje máquina.

Este es el lenguaje utilizado directamente por el procesador, consta de un conjunto de instrucciones codificadas en binario. Es el sistema de códigos directamente interpretable por un circuito microprogramable.

Este fue el primer lenguaje utilizado para la programación de computadores. De hecho, cada máquina tenía su propio conjunto de instrucciones codificadas en ceros y unos. Cuando un algoritmo está escrito en este tipo de lenguaje, decimos que está en código máquina.

Programar en este tipo de lenguaje presentaba los siguientes inconvenientes:

- Cada programa era válido sólo para un tipo de procesador u ordenador.
- La lectura o interpretación de los programas era extremadamente difícil y, por tanto, insertar modificaciones resultaba muy costoso.
- Los programadores de la época debían memorizar largas combinaciones de ceros y unos, que equivalían a las instrucciones disponibles para los diferentes tipos de procesadores.
- Los programadores se encargaban de introducir los códigos binarios en el computador, lo que provocaba largos tiempos de preparación y posibles errores.

A continuación, se muestran algunos códigos binarios equivalentes a las operaciones de suma, resta y movimiento de datos en lenguaje máquina.

Algunas operaciones en lenguaje máquina.	
Operación	Lenguaje máquina
SUMAR	00101101
RESTAR	00010011
MOVER	00111010

Dada la complejidad y dificultades que ofrecía este lenguaje, fue sustituido por otros más sencillos y fáciles utilizar. No obstante, hay que tener en cuenta que todos **los programas** para poder ser **ejecutados**, han de traducirse siempre al **lenguaje máquina** que es el único **que entiende la computadora**.

1.2. Lenguaje Ensamblador.

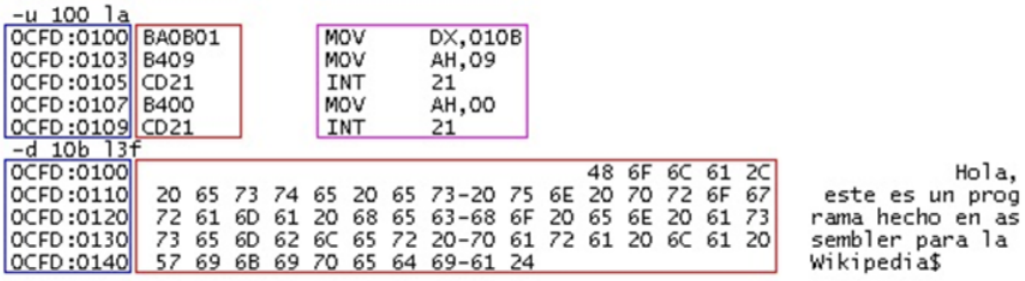
La **evolución** del lenguaje **máquina** fue el lenguaje **ensamblador**. Las instrucciones ya no son **secuencias binarias**, se sustituyen **por códigos de operación** que describen una operación elemental del procesador. Es un **lenguaje de bajo nivel**, al igual que el lenguaje máquina, ya que **dependen** directamente del **hardware donde son ejecutados**.

Mnemotécnico: son palabras especiales, que sustituyen largas secuencias de ceros y unos, utilizadas para referirse a diferentes operaciones disponibles en el juego de instrucciones que soporta cada máquina en particular.

En ensamblador, cada instrucción (mnemotécnico) se corresponde a una instrucción del procesador. En la siguiente tabla se muestran algunos ejemplos.

Algunas operaciones y su mnemotécnico en lenguaje Ensamblador.	
Operación	Lenguaje Ensamblador
MULTIPLICAR	MUL
DIVIDIR	DIV
MOVER	MOV

En el siguiente gráfico puedes ver parte de un programa escrito en lenguaje ensamblador. En color rojo se ha resaltado el código máquina en hexadecimal, en magenta el código escrito en ensamblador y en azul, las direcciones de memoria donde se encuentra el código.



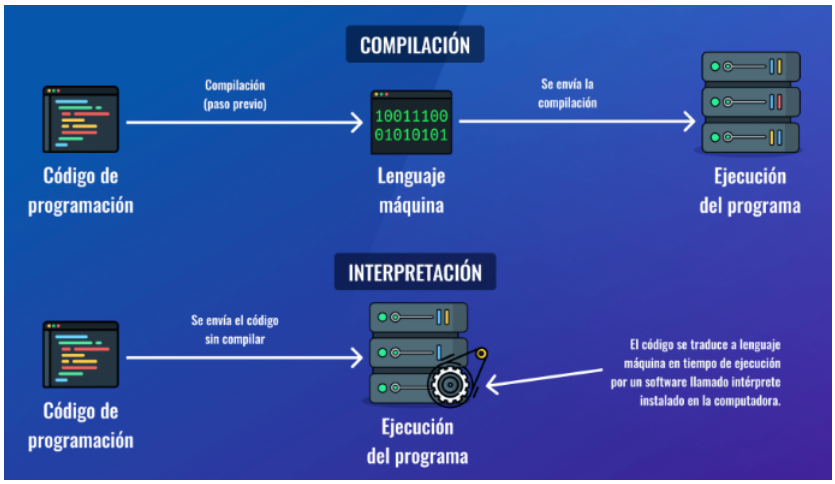
Pero aunque ensamblador fue un intento por aproximar el lenguaje de los procesadores al lenguaje humano, presentaba múltiples **dificultades**:

- Los programas seguían dependiendo directamente del hardware que los soportaba.
- Los programadores tenían que conocer detalladamente la máquina sobre la que programaban, ya que debían hacer un uso adecuado de los recursos de dichos sistemas.
- La lectura, interpretación o modificación de los programas seguía presentando dificultades.

Todo programa escrito en lenguaje ensamblador necesita de un intermediario, que realice la traducción de cada una de las instrucciones que componen su código al lenguaje máquina correspondiente. Este intermediario es el programa ensamblador. El programa original escrito en lenguaje **ensamblador** constituye el **código fuente** y el programa traducido al **lenguaje máquina** se conoce como **programa objeto** que será directamente ejecutado por la computadora.

1.3. Lenguajes compilados.

Para paliar los problemas derivados del uso del lenguaje ensamblador y con el objetivo de acercar la programación hacia el uso de un lenguaje más cercano al humano que al del computador, nacieron los **lenguajes compilados**. Algunos ejemplos de este tipo de lenguajes son: Pascal, Fortran, Algol, C, C++, etc.



Al ser lenguajes más cercanos al humano, también se les denomina **lenguajes de alto nivel**. Son más fáciles de utilizar y comprender, las instrucciones que forman parte de estos lenguajes utilizan palabras y signos reconocibles por el programador.

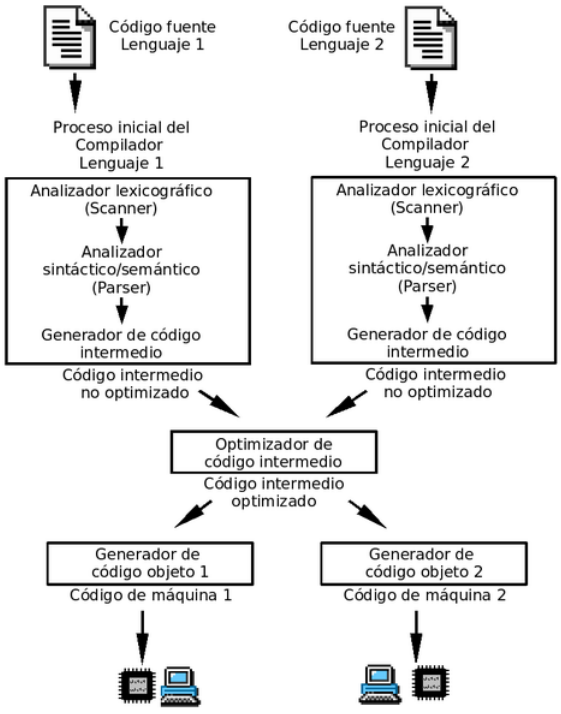
¿Cuáles son sus **ventajas**?

- Son mucho más fáciles de aprender y de utilizar que sus predecesores.
- Se reduce el tiempo para desarrollar programas, así como los costes.
- Son independientes del hardware, los programas pueden ejecutarse en diferentes tipos de máquina.
- La lectura, interpretación y modificación de los programas es mucho más sencilla.

Pero un programa que está escrito en un lenguaje de alto nivel también tiene que traducirse a un código que pueda utilizar la máquina. Los programas traductores que pueden realizar esta operación se llaman compiladores.

Compilador: Es un programa cuya función consiste en traducir el código fuente de un programa escrito en un lenguaje de alto nivel a lenguaje máquina. Al proceso de traducción se le conoce con el nombre de compilación.

Para ilustrar el proceso de compilación de programas te proponemos el siguiente enlace:



El compilador realizará la traducción y además informará de los posibles errores. Una vez subsanados, se generará el programa traducido a código máquina, conocido como **código objeto**. Este programa aún no podrá ser ejecutado hasta que no se le añadan los módulos de enlace o bibliotecas, durante el proceso de *enlazado*. Una vez finalizado el enlazado, se obtiene el **código ejecutable**.

1.4. Lenguajes interpretados.

Se caracterizan por estar diseñados para que su ejecución se realice a través de un **intérprete**. Cada instrucción escrita en un lenguaje interpretado se analiza, traduce y ejecuta tras haber sido verificada. Una vez realizado el proceso por el intérprete, la instrucción se ejecuta, pero no se guarda en memoria.

Intérprete: es un programa traductor de un lenguaje de alto nivel en el que el proceso de traducción y de ejecución se llevan a cabo simultáneamente, es decir, la instrucción se pasa a lenguaje máquina y se ejecuta directamente. No se genera programa objeto, ni programa ejecutable.

Los **lenguajes interpretados** generan **programas de menor tamaño** que los generados por un compilador, al no guardar el programa traducido a código máquina. Pero presentan el inconveniente de ser algo **más lentos**, ya que han de ser **traducidos durante su ejecución**. Por otra parte, **necesitan** disponer en la máquina del programa **intérprete ejecutándose**, algo que no es necesario en el caso de un programa compilado, para los que sólo es necesario tener el programa ejecutable para poder utilizarlo.

Ejemplos de lenguajes interpretados son: **Perl, PHP****, Python, JavaScript,** etc.

A medio camino entre los lenguajes compilados y los interpretados, existen los lenguajes que podemos denominar **pseudo-compilados o pseudo-interpretados**, es el caso del **Lenguaje Java**. Java puede verse como compilado e interpretado a la vez, ya que su código fuente se compila para obtener el código binario en forma de **bytecodes**, que son estructuras parecidas a las instrucciones máquina, con la importante propiedad de **no ser dependientes de ningún tipo de máquina** (se detallarán más adelante). La **Máquina Virtual Java** se encargará de **interpretar** este código y, para su ejecución, lo traducirá a código máquina del **procesador** en particular **sobre el que se esté trabajando**.

Debes conocer

Puedes entender por qué Java es un lenguaje compilado e interpretado a través del siguiente enlace.

Comparacion del lenguaje Java con otros lenguajes