Lenguajes de Bajo Nivel

Lenguaje Máquina.

Es el sistema de códigos directamente interpretable por un circuito micro programable, como el microprocesador de una computadora o el microcontrolador de un autómata. Este lenguaje está compuesto por un conjunto de instrucciones que determinan acciones a ser tomadas por la máquina. Estas instrucciones son normalmente ejecutadas en secuencia, con eventuales cambios de flujo causados por el propio programa o eventos externos. El lenguaje máquina es específico de cada máquina o arquitectura de la máquina, aunque el conjunto de instrucciones disponibles pueda ser similares entre ellas.

Los circuitos micro programables son sistemas digitales, lo que significa que trabajan con dos únicos niveles de tensión. Dichos niveles, por abstracción, se simbolizan con el cero, 0, y el uno, 1, por eso el lenguaje de máquina solo utilice dichos signos. Esto permite el empleo de las teorías del algebra booleana y del sistema binario en el diseño de este tipo de circuitos y en su programación. La principal ventaja del lenguaje máquina es su alta velocidad, debida a la traducción inmediata de los códigos binarios.

Ejemplo de una instrucción en lenguaje máquina:

```
1    0010111000000001
2    0000000000001010
```

Lenguaje ensamblador:

Son la representación más entendible para el humano de los códigos del lenguaje máquina. Cada instrucción en lenguaje ensamblador aparece case a la par con el lenguaje máquina, esto debido a que los fabricantes de hardware diseñan sus chips pensando ya en las instrucciones de un lenguaje ensamblador.

Un ejemplo de una instrucción en lenguaje ensamblador:

```
1    mov ax, 10
```

A pesar de que el lenguaje ensamblador es más fácil de entender por las personas sigue teniendo las desventajas del lenguaje máquina. Realmente lo que hace es ayudar un poco a que el código sea más legible. El lenguaje ensamblador adicionalmente necesita un traductor (ensamblador) capaz de convertir dichos códigos en lenguaje máquina.

Ejemplos de ensambladores: TASM (Turbo Assembler de Borland), MASM (Microsoft Macro Assembler), NASM (Netwide Assembler- Libre).

Lenguaje de Alto Nivel

Son lenguajes más afines con el programador en los que una instrucción puede representar varias instrucciones en lenguaje máquina. Incluyen los lenguajes estructurados y no estructurados. Como ejemplos tenemos: Basic, Pascal, C, APL, FORTRAM, COBOL, LISP y PROLOG, etc.

Los lenguajes de alto nivel necesitan de un traductor que puede ser interpretador o compilador. Los interpretadores o intérpretes, necesitan de un programa auxiliar y que traduce en tiempo real las instrucciones al lenguaje máquina, por lo tanto, cada vez que un programa interpretado se ejecuta debe ejecutarse también su intérprete. Ejemplos de lenguajes interpretados: Basic, PROLOG, LISP, entre otros.

Los lenguajes compilados son aquellos que necesitan de un compilador para la traducción al lenguaje máquina. La traducción o compilación se hace solo una vez, y el resultado es un código objeto entendible por la máquina. Para ejecutar un programa compilado no se necesita de su compilador. Ejemplos de lenguajes compilados: Pascal, C, Fortran, Cobol, Modula-2, etc.

Algunas de las características de los lenguajes de alto nivel:

- Depuración más sencilla: debido a que el código es más legible, la depuración también se hace más fácil. Con la ayuda de editores (IDES - Entorno de Desarrollo Integrados) la compilación, depuración y ejecución se hacen más fácilmente.
- Productividad aceptable: son más productivos que los lenguajes de alto nivel.

- Algunos permiten la portabilidad: generalmente los interpretados.

Lenguajes de cuarta generación

La evolución de los lenguajes de programación es vista también desde el punto de vista de las generaciones. Los lenguajes de primera generación son los mismos lenguajes del nivel de la máquina. Los lenguajes de segunda generación son los ensambladores. Y los lenguajes de tercera generación son los nombrados en los lenguajes de alto nivel.

Una cuarta generación de lenguajes de programación es aún confusa de definir, algunos autores la relacionan con lenguajes visuales con facilidades de acceso a base de datos que aparecieron para desarrollar bajo la plataforma Windows. Entre ellos encontramos los lenguajes del Visual Studio de Microsoft (como Visual Basic, Visual FoxPro, Visual C); las herramientas de Borland (Delphi, JBilder, C Builder); entre otros.

También se puede incluir en esta generación, las herramientas que generan el código a partir de plantillas y configuraciones, lenguajes de consulta como SQL, los lenguajes script (JavaScript) y los lenguajes de marcado (HTML, XML).

Lenguajes Orientados a Objetos

La programación orientada a Objetos (POO) no es algo nuevo, pues existe desde los años 60 con lenguajes como Smaltalk, Simula, Ada. Se ha hecho más populares en los últimos 10 años con la aparición de Java y C#. La POO es una extensión de los lenguajes de Alto Nivel Estructurados que tratan de representar de una forma más sencilla el modelo del mundo real.

El concepto de programación Orientada a Objetos: agrupa un conjunto de técnicas que nos permiten desarrollar y mantener muchos más fácilmente programas de gran complejidad.

La POO intenta resolver principalmente problemas de la Ingeniería de Software como: portabilidad, reusabilidad, mantenibilidad, entre otros. Para ello se basa en las características

claves como el encapsulamiento, la herencia, polimorfismo, y el desarrollo orientado primero hacia el que, y luego hacia el cómo (interfaces).

Lenguajes Orientados a Objetos: Java, C#, Ruby, Php, V.B.Net, Python, Perl, JavaScript, C , Delphi.