EL PROCESO DE TRADUCCIÓN

TRADUCCIÓN A LENGUAJE MÁQUINA

- Tanto los programas escritos en ensamblador como los escritos en lenguajes de alto nivel, necesitan ser traducidos al lenguaje máquina para poder ser ejecutados por esta.

TRADUCCIÓN A LENGUAJE MÁQUINA

- La traducción de un programa en ensamblador á código máquina es muy sencilla y se realiza mediante un traductor también llamado ***ensamblador***

TRADUCCIÓN A LENGUAJE MÁQUINA

- La traducción de un programa escrito en un lenguaje de alto nivel es un proceso más complejo y ha de realizarse mediante un compilador o un intérprete

COMPILADORES

- Los compiladores traducen el programa entero y luego se monta generando un programa ejecutable por si sólo.
- Ejemplos de lenguajes que utilizan normalmente compilador: C, C++, Ada, COBOL, FORTRAN, Pascal,...

Ventajas
Más rápida la ejecución
Sólo se traduce una vez
Los ejecutables obtenidos se ejecutan sin el compilador

Desventajas
Más difícil la depuración. Menos control sobre dónde están los errores

INTÉRPRETES

- Los intérpretes se encargan de realizar la traducción instrucción a instrucción en el momento en el que se ejecuta el programa. No se genera fichero ejecutable.
- Ejemplos de lenguajes que utilizan normalmente intérprete: BASIC, JavaScript, PHP, Python, Lisp, Perl, Prolog, Smalltalk

Ventajas
Más fácil la depuración. Más control sobre dónde están los errores
Sólo se traducen las instrucciones que se ejecutan

Desventajas
Más lenta la ejecución
Se necesita el intérprete en el momento de la ejecución

MÁS DIFERENCIAS: COMPILADORES O INTÉRPRETES

- Los programas compilados se compilan una vez y se utilizan cuantas veces se desee sin necesidad de volver a compilar. Los lenguajes interpretados son traducidos cada vez que se ejecutan y necesitan siempre del intérprete.
- Los compiladores analizan todo el programa y no generan resultados si no es correcto todo el código. Los intérpretes analizan las instrucciones según las necesitan y pueden iniciar la ejecución de un programa con errores e incluso terminar correctamente una ejecución de un programa con errores siempre que no haya sido necesario el uso de las instrucciones que contienen dichos errores.
- Un compilador traduce cada instrucción una sola vez. Un intérprete debe traducir una instrucción cada vez que la encuentra.
- Los programas son compilados para una arquitectura determinada y no pueden ser utilizados en otras arquitecturas no compatibles (pueden existir distintos compiladores para generar ejecutables para diferentes arquitecturas). Un lenguaje interpretado puede ser utilizado en cualquier arquitectura que disponga de un intérprete sin necesidad de cambios.
- Los lenguajes compilados son más eficientes que los interpretados y además permiten distribuir el programa de forma confidencial (sólo ejecutables).
- Es más sencillo empaquetar lenguajes interpretados dentro de otros lenguajes, como JavaScript o PHP dentro de HTML.

PROCESO DE COMPILACIÓN

INTÉRPRETES

- Inicialmente, los lenguajes interpretados eran compilados instrucción por instrucción; es decir, cada instrucción era compilada a medida que estaba a punto de ser ejecutada, y si un bucle o una subrutina hacía que ciertas instrucciones se ejecutaran múltiples veces, estas debían ser recompiladas repetidamente.
- Esta forma de generar código máquina cada vez es menos común. Actualmente, la mayoría de los lenguajes interpretados usan una representación intermedia, que combina tanto la compilación como la interpretación.

LENGUAJES INTERMEDIOS

- En este caso, un compilador puede producir una cierta forma de representación intermedia del programa, como el bytecode o el código enhebrado, que entonces es ejecutado por un intérprete de este código intermedio.
- Ejemplos de esta forma de ejecución pueden ser Python, Java o Ruby. La representación intermedia puede ser tratada después de varias formas: interpretando cada vez que se vaya a ejecutar, o cada vez que un cambio en el código fuente es detectado antes de la ejecución (como en Python).

LENGUAJES INTERMEDIOS -MÁQUINAS VIRTUALES

- Para compartir ventajas de ambos procesos algunos sistemas utilizan una traducción en dos fases. Primero el programa original (código fuente) es compilado a un lenguaje intermedio, portable e interpretable *(C, bytecode,...)*. En una segunda fase, el código intermedio es interpretado en cada arquitectura. Ésta aproximación es la que se realiza, por ejemplo, en Java.

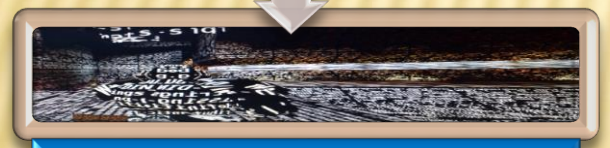
MÁQUINAS VIRTUALES JAVA

CÓDIGO JAVA
- En un editor se generan los ficheros fuente en Java

COMPILADOR
- Los ficheros fuente son compilados a un lenguaje intermedio bytecode

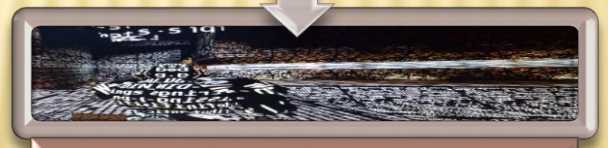
BYTECODE .CLASS
- Ficheros objeto Bytecode portables

MÁQUINA VIRTUALJAVA PARA WINDOWS
- ANALIZADOR DE BYECODE
- CARGADOR DE CLASES
- INTÉRPETRE



ARQUITECTURA WINDOW
- SISTEMA OPERATIVO
- FIRMWARE
- HARDWARE

MÁQUINA VIRTUALJAVA PARA UNIX
- ANALIZADOR DE BYECODE
- CARGADOR DE CLASES
- INTÉRPETRE



ARQUITECTURA UNIX
- SISTEMA OPERATIVO
- FIRMWARE
- HARDWARE

MÁQUINA VIRTUALJAVA PARA SOLARIS
- ANALIZADOR DE BYECODE
- CARGADOR DE CLASES
- INTÉRPETRE

ARQUITECTURA SOLARIS
- SISTEMA OPERATIVO
- FIRMWARE
- HARDWARE

LENGUAJES INTERMEDIOS

- Por otro lado, algunas aplicaciones permiten ser programadas con lenguajes declarativos. Estos lenguajes no tienen por objeto solicitar acciones a la computadora sino solicitar acciones a la aplicación sobre la que se ejecutan. Es el caso, por ejemplo, de SQL, un lenguaje declarativo de cuarta generación diseñado para trabajar con bases de datos. Este lenguaje SQL es interpretado por el motor de la Base de Datos, no ejecutado por una máquina determinada.