Unidad 2

Introducción
Lenguajes y Compiladores

Principal material bibliográfico utilizado

- Compiladores Principios, técnicas y herramientas. Aho y Ullman. Addison Wesley.
- www.jorgesanchez.net
- www.iqcelaya.itc.mx/~vicente/Programacion/TradComp.pdf
- www.definicion.org/lenguaje-de-programacion.

Programa informático

Secuencia de pasos finita expresada en un lenguaje comprensible por una computadora, que resuelve un problema.

Conjunto de instrucciones o sentencias.

Lenguaje de programación

Un lenguaje de programación es aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes.

Cada lenguaje de programación utiliza un grupo de símbolos o reglas que tienen un significado.

Lenguaje de programación

Conjunto de símbolos, reglas sintácticas (forma de escribir) y semánticas (sentido de aquello que se escribe) junto con sus elementos y las expresiones.

Los lenguajes de programación surgen por la necesidad de automatizar tareas que realiza el usuario de forma repetitiva.

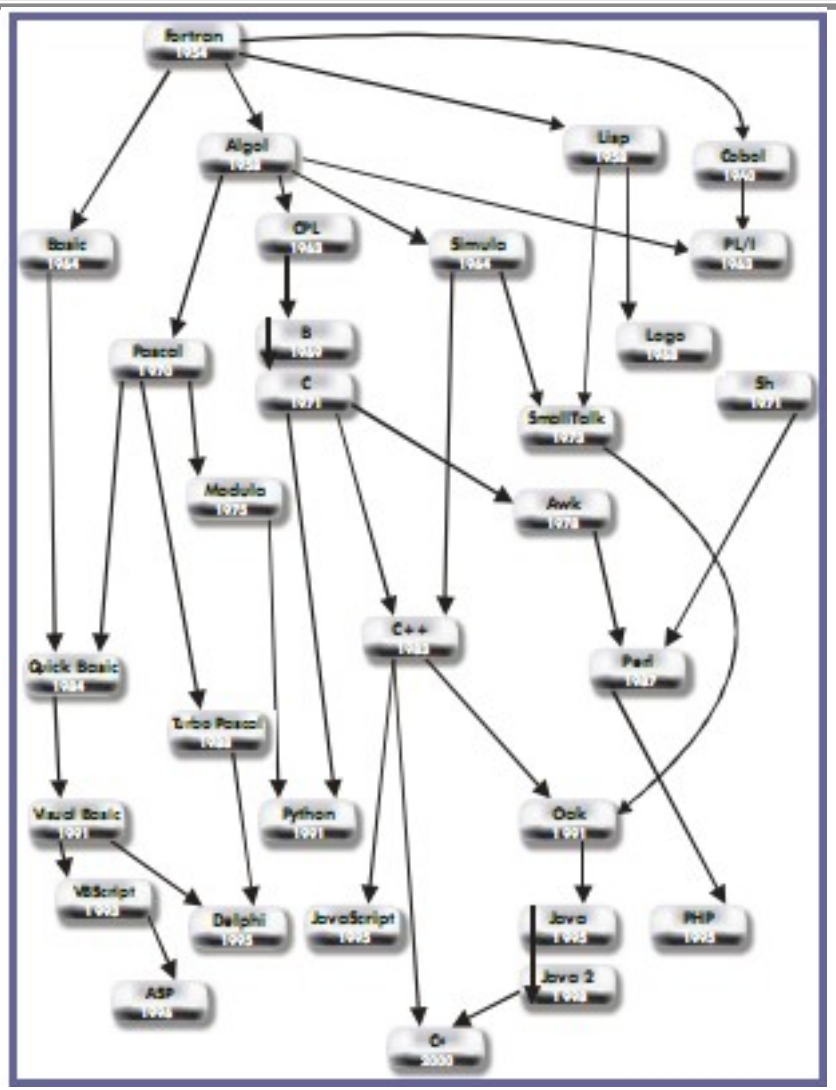
Generaciones de lenguajes

Una posible partición de generaciones podría ser.

- 1era. Generación (máquina 0 y 1)
- 2da. Generación (lenguaje ensamblador – código de máquina a una forma más textual).
- 3era. Generación (mayor comprensión humana, una instrucción se puede traducir en varias de bajo nivel (FORTRAN, ALGOL, COBOL fueron los iniciales).
- 4ta. Generación (se los podría relacionar mas con la clasificación de lenguajes orientados a objetos).
- 5ta. Generación (casi no se utiliza).

La diferenciación de las clasificaciones a partir de la 3er generación no fue muy precisa, se consideran a partir de dicha generación lenguajes de alto nivel y a los anteriores de bajo nivel

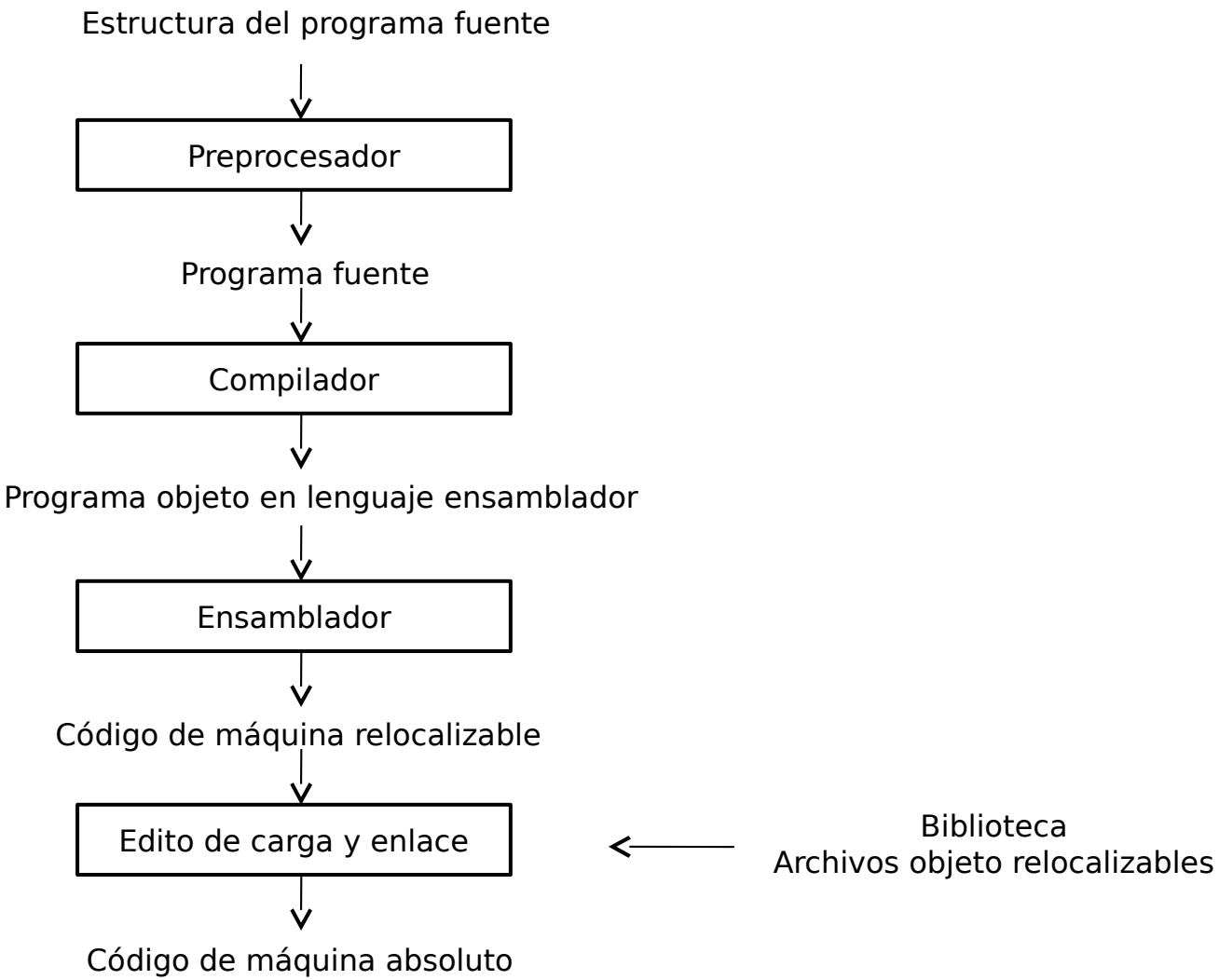
Historia


Fortran
Algol
Lisp
Cobol
Basic
CPL
Simula
PL/I
Logo
Pascal
B
C
Sh
SmallTalk
Modula
Awk
C++
Quick Basic
Perl
Turbo Pascal
Visual Basic
Python
Oak
VBScript
Delphi
JavaScript
Java
PHP
ASP
Java 2
C#

Sistema para procesamiento de un lenguaje

Un programa fuente se puede dividir en módulos almacenados en archivos distintos.
El Preprocesador, se encarga de unirlos, también puede expandir abreviaturas, llamadas a macros, etc.
El programa objeto (en este caso en lenguaje ensamblador), puede requerir procesa_ miento adicional . El Ensamblador traduce el lenguaje ensamblador (lenguaje objeto) a código maquina. El editor de carga y enlace efectúa el enlazado con rutinas de biblioteca y se produce el código que realmente se ejecute en la máquina.

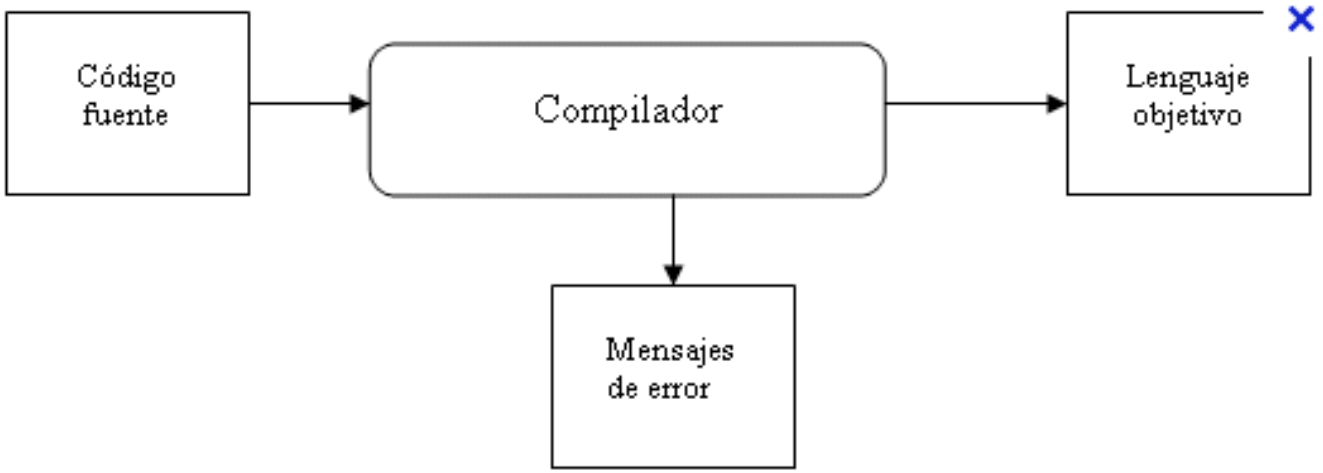
Sistema para procesamiento de un lenguaje



Compiladores

Programa que lee en un lenguaje fuente y lo traduce a otro lenguaje (el lenguaje objeto).

Un lenguaje objeto puede ser otro lenguaje o el código máquina mismo.



Compilación

Dos partes principales:

- ANALISIS: divide el programa fuente en sus partes componentes y crea una representación intermedia del programa fuente. Se genera una estructura de árbol (llamado árbol sintáctico), donde se determinan las operaciones del programa fuente.

- SINTESIS: construye el programa objeto. Se requieren las técnicas mas especializadas.

Compilación

Análisis léxico ó lineal `o exploración: la cadena de caracteres que constituye el programa fuente se lee de izquierda a derecha y se agrupa en componentes léxicos (secuencias de caracteres con un significado colectivo)

Análisis jerárquico ó sintáctico: los componentes léxicos se agrupan jerárquicamente en colecciones anidadas con un significado colectivo.

Análisis semántico: se realizan ciertas revisiones para asegurar que los componentes de un programa se ajustan de un modo significativo

Compilación

Análisis léxico ó lineal ò exploración

Ejemplo: tomamos una instrucción de un programa fuente.

Posicion := inicial + velocidad * 60

Se detectarían los siguientes componentes léxicos:

- Identificador: posicion, inicial, velocidad
- Operador: asignación :=, suma +, multiplicación *
- El numero 60

- Los espacios en blanco por lo general se eliminan en este análisis.

Compilación

Análisis léxico ó lineal ò exploración

Por lo general los componentes son:

Identificadores: empiezan por una letra y son una sucesión de caracteres no pertenecientes a las palabras reservadas.

Palabras reservadas: BEGIN, END, VAR, etc.

Operadores relacionales: >, <, ==, =<, >=, <>.

Operadores aritméticos: +, -, *, etc.

Compilación

Análisis sintáctico ò jerárquico:

Agrupar los componentes léxicos en frases gramaticales, que por lo general se representan mediante un árbol de análisis sintáctico.

La estructura jerárquica de un programa por lo general se expresa utilizando reglas recursivas

Por ejemplo un lenguaje podría

tener las siguientes reglas:

1) Cualquier identificador es una expresión
2) Cualquier numero es una expresión
3) Si exp1 y exp2 son expresiones,

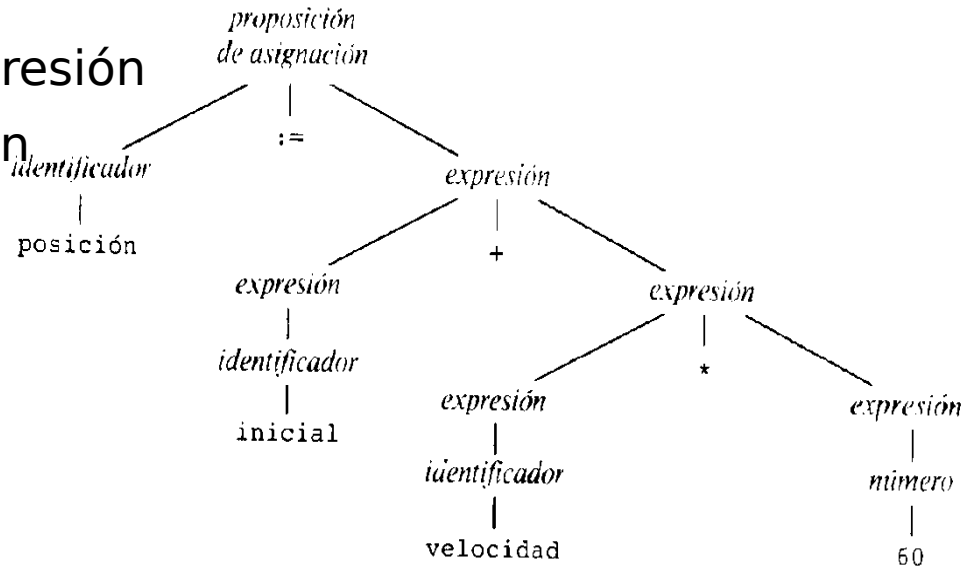
   entonces también lo son:

   exp1 + exp2

   exp1 * exp2

   (exp1)

Ejemplo:



Compilación

Análisis sintáctico ò jerárquico:

Otras reglas podrían ser

1) Si id1 es un identificador y exp2 es una expresión, entonces id1:=exp2 es una proposición.
2) Si exp1 es una expresión y prop2 es una proposición, entonces:

   mientras (exp1) hacer prop2

   Si (exp1) entonces prop2

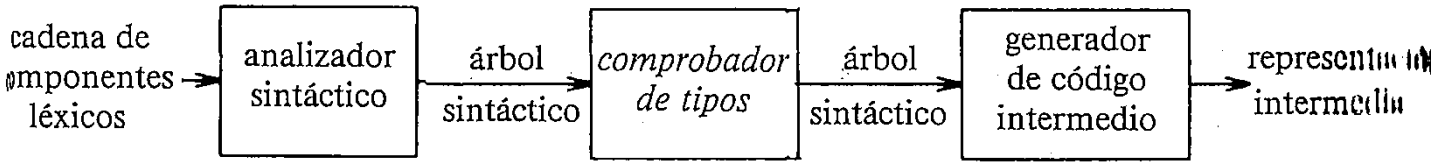
Son proposiciones.

Compilación

Análisis semántico:

Revisa el programa fuente para encontrar errores semánticos y reúne información sobre los tipos para la fase posterior de generación de código.

Un componente importante del análisis semántico es la **verificación de tipos.** Se verifica si cada operador, tiene operandos permitidos por la especificación del lenguaje.



Compilación

Análisis semántico:

Un compilador debe informar de un error si se aplica un operador a un operando incompatible.

Un verificador de tipos se asegura de que el tipo de una construcción coincida con el previsto en su contexto.

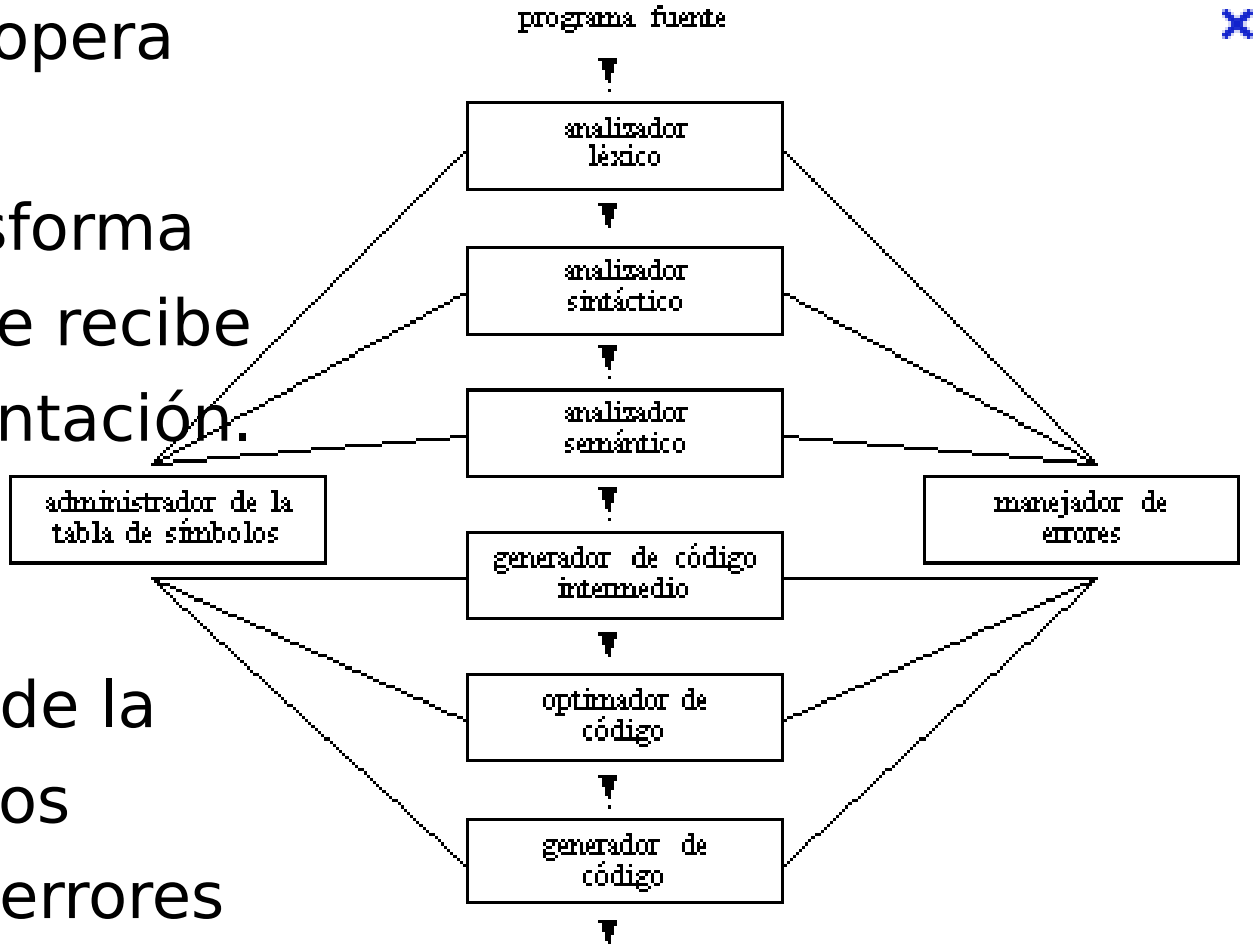
Por ejemplo el operador aritmético predefinido “mod” en Pascal, exige operandos de tipo entero, el comprobador de tipos debe asegurarse de que los operandos sean de tipo entero.

Debe asegurarse de que la desreferenciacion se aplique a un apuntador, de que la indización se haga solo sobre una matriz, de que una función

Fases de un compilador

Un compilador opera en fases.

Cada fase transforma el programa que recibe en otra representación.



Vemos que la administración de la tabla de símbolos y el manejo de errores Interactúan con todas las fases.

Compilación – tabla de símbolos

Es una estructura de datos que contiene un registro por cada identificador, con los campos para los atributos del identificador (estos atributos pueden proporcionar información sobre la memoria asignada, su tipo, su ámbito, en el caso de nombres de procedimientos, cosas como el numero y tipos de sus argumentos, la forma de pasarlos, etc.). La estructura permite encontrar rápidamente el registro de cada identificador y almacenar o consultar datos de ese registro.

Cuando el analizador léxico encuentra un identificador lo inserta en la tabla de símbolos. Las fases restantes introducen información y después la utilizan de diferentes formas.

Compilación – detección de errores

Cada fase puede encontrar errores, la idea es que el compilador no se detenga cada vez que encuentra un error sino que le de un tratamiento.

Se manejan errores en todas las fases.

Compilación - Generación de código intermedio

Después del análisis, algunos compiladores generan una representación intermedia del programa fuente.

Esta representación intermedia debe ser fácil de producir y fácil de traducir al programa objeto.

Compilación - Optimador de código

Esta fase trata de mejorar el código intermedio, para que resulte un código de maquina mas rápido de ejecutar.

Compilación - Generación de código

Es la fase final del compilador, la generación del código objeto en lenguaje de maquina relocalizable o código ensamblador.

Las posiciones de memoria se seleccionan para cada una de las variables usadas por el programa.

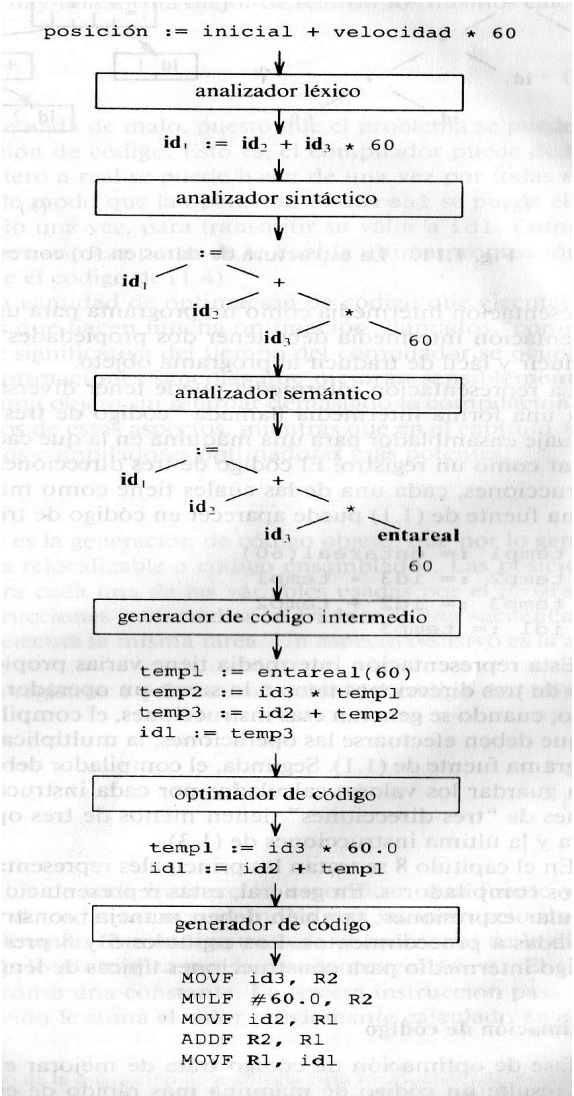
Después cada una de las instrucciones intermedias se traduce a una secuencia de instrucciones de maquina.

Un aspecto importante es la asignación de variables a registros.

Compilación

TABLA DE SÍMBOLOS

1	posición	...
2	inicial	...
3	velocidad	...
4		



Ensambladores, Cargadores y Editores de enlace

Cuando el compilador genera código ensamblador (versión mnemotécnica del código de maquina donde se utilizan nombres en lugar de códigos binarios), se necesitan de programas que conviertan el código ensamblador en lenguaje de máquina (0 y 1), dichas conversiones comienzan con el Ensamblador quien transforma en 0 y 1, y continúan por el Editor de carga y enlace, que entre otras operaciones, permiten formar un solo programa a partir de varios archivos de código máquina y cargar el programa en la memoria.

Ensambladores, Cargadores y Editores de enlace

Ejemplo de código ensamblador:

MOV a, R1

ADD #2, R1

MOV R1, b

Ejemplo de código de maquina absoluto:

0001 01 00 00001111

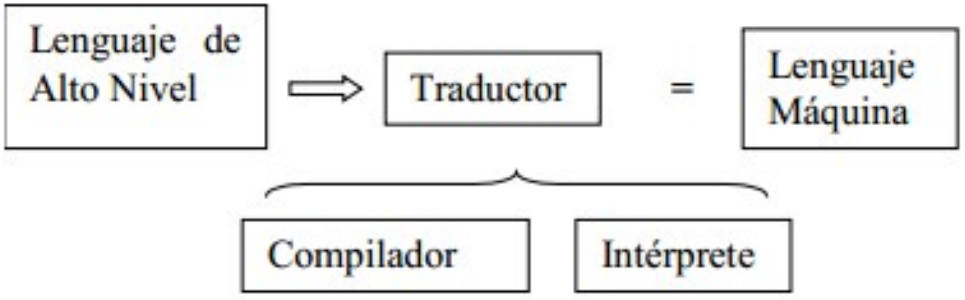
0011 01 10 00000010

0010 01 00 00010011

Traductores de un lenguaje de programación

Los traductores son programas que traducen los programas en código fuente, escritos en lenguajes de alto nivel, a programas escritos en lenguaje máquina.

Los traductores pueden ser de dos tipos: compiladores e intérpretes



Traductores de un lenguaje de programación

Compilador

Un compilador es un programa que lee el código escrito en un lenguaje (lenguaje origen), y lo traduce en un programa equivalente escrito en otro lenguaje (lenguaje objetivo). Como una parte fundamental de este proceso de traducción, el compilador le hace notar al usuario la presencia de errores en el código fuente del programa.

Intérprete

Los intérpretes no producen un lenguaje objetivo como en los compiladores. Un intérprete lee el código como está escrito e inmediatamente lo convierte en acciones; es decir, lo ejecuta en ese

Diferencias entre lenguajes interpretados y lenguajes compilados

Un Intérprete va paso a paso mientras que el compilador lo traduce todo del tirón. Ambos pasan de un lenguaje fuente a un código binario.

Un programa que ha sido compilado puede correr por sí sólo, pues en el proceso de compilación se lo transformo en otro lenguaje (lenguaje máquina).

Un intérprete traduce el programa cuando lo lee, convirtiendo el código del programa directamente en acciones. La ventaja del intérprete es que dado cualquier programa se puede interpretar en cualquier plataforma (sistema operativo). En cambio, el archivo generado por el compilador solo funciona en la plataforma en donde se le ha creado. Sin embargo, hablando de la velocidad de ejecución, un archivo compilado es de 10 a 20 veces más rápido que un archivo interpretado.

Entornos integrados de desarrollo

A fin de que todos los procesos relacionados a la programación sean más cómodos y sencillos, todas las herramientas que permiten al programador crear programas, generalmente se integran en un único software conocido como entorno de programación o IDE (Integrate Development Environment, Entorno integrado de desarrollo).

Por lo general, estas herramientas agrupan:

- Editor de código: programa utilizado para escribir el código, si bien alcanza con un procesador de texto, es conveniente que este contenga: Coloreado inteligente de instrucciones, Correctores de errores, Ayuda.
- Compilador: desde el programa fuente al código maquina.
- Lanzador/Cargador: Prueba la ejecución de la aplicación sin abandonar el entorno.
- Depurador: Para realizar pruebas sobre el programa.

Desde el punto de vista del usuario de estos entornos, solo hay dos grandes pasos: Compilar (conseguir el código ejecutable) y ejecutar (cargar el código maquina para que se ejecute).

FIN