Compiladores e intérpretes
Introducción

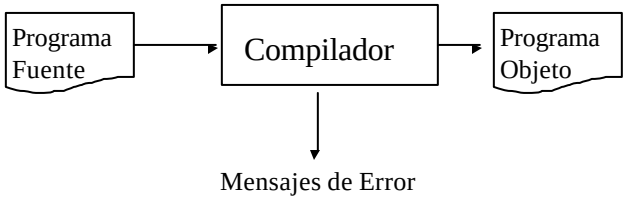
Profesor: Eridan Otto

Introducción

- Perspectiva histórica
- Motivación
- Definiciones
- Componentes y fases de un compilador

Compiladores e intérpretes
Introducción

- Definiciónes básicas
  - Traductor: desde un punto de vista general, es un proceso que convierte un programa escrito o texto en un lenguaje fuente a un texto o programa escrito en un lenguaje de destino.Incluyen tanto a los compiladores como a los intérpretes.
  - Compilador: Proceso de tradución que convierte un programa fuente escrito en un lenguaje de alto nivel, a un programa objeto en código de máquina, listo por tanto para su ejecución en el computador.



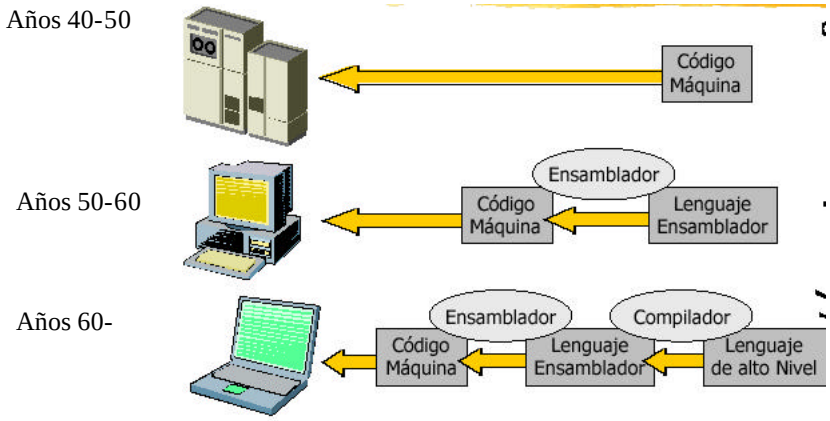
Compiladores e intérpretes
Introducción

- Definiciónes básicas
  - Intérprete: Ejecuta una a una las instrucciones de un programa de alto nivel. La entrada es un archivo en un lenguaje de alto nivel, la diferencia con un compilador es que la salida es una ejecución
  - Ej:Basic, LISP, PROLOG son interpretados
  - Ventaja: fácil depuración
  - Desventaja:lentitud y consumo de recursos (pues el intérprete ocupa tiempo y memoria)

Compiladores e intérpretes

Introducción

Perspectiva histórica:



Compiladores e intérpretes

Introducción. Perspectiva histórica:

- En los 50 los compiladores eran considerados programas muy difíciles de construir
- Ejemplo: Fortran evolucionó durante 18 años de trabajo en grupo.
- Hoy en día se han desarrollado técnicas sistemáticas,entornos de programación y herramientas de software que facilitan la tarea de desarrollo de compiladores, intérpretes y traductores

Compiladores e intérpretes

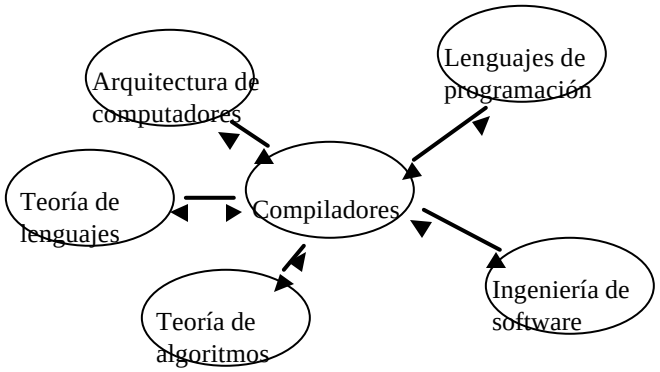
Introducción. Perspectiva histórica:

- Los primeros compiladores traducían fórmulas aritméticas a código de máquina
- No puede darse una fecha exacta del primer desarrollo
- Varios grupos en forma independiente desarrollaron técnicas de análisis y diseño de compiladores.
- Hoy en día hay gran variedad de compiladores para múltiples lenguajes de alto nivel , disponibles para muchas plataformas, es decir cubren gran variedad de códigos de máquina

Compiladores e intérpretes

Introducción.

Conceptos Relacionados



Compiladores e intérpretes

Introducción.

Motivación: el conocimiento de estos tópicos permite

- Saber más sobre corrección y eficiencia del código
- Profundizar más sobre lenguajes:
  - Tipos:clases, estáticos, dinámicos, polimorfismo, sobrecarga de operadores, conversiones
  - Estructura de bloques, ámbitos
  - Paso de parámetros
  - Gestión de memoria, punteros

Compiladores e intérpretes

Introducción.

Motivación: el conocimiento de estos tópicos permite

- Aplicación de la teoría a la práctica
  - Antes de el uso de teoría de autómatas y lenguajes formales, técnicas de programación, los compiladores eran muy malos.
- Aplicar teoría y herramientas a otros campos:
  - Intérpretes de comandos y consultas en interfases usuarias
  - Formateadores de textos (latex)
  - Lenguajes de simulación (GPSS)
  - Editores de texto

Compiladores e intérpretes

Introducción.

Compilador, definiciones I:

- Ensamblador
  - Compilador de bajo nivel, el lenguaje fuente tiene una estructura simple que permite una traducción , una a una de una sentencia fuente a una en código de máquina.
- Compilador cruzado:
  - Compilador que toma un lenguaje fuente y genera un código objeto, este objeto es para una plataforma o computador distinto en el que se compila.Ejemplo uso: fase de desarrollo de nuevos computadores.

Compiladores e intérpretes

Introducción.

Compilador, definiciones II:

- Compilar-linkear-ejecutar versus compilar-ejecutar:
  - La primera opción permite la modularización, compilando por separado las partes y luego enlazándolas.
  - La segunda opción es la más simple. El compilador deja en memoria directamente un módulo cargable que se ejecuta a continuación.
- Compilador de una o varias pasadas:
  - "pasada":es el recorrido total de todo el fuente, con algún objetivo específico. Por ejemplo, recursión indirecta:

    a()-✍b() y b()--✍a(), se pueden hacer dos pasadas.

Compiladores e intérpretes

Introducción.

Compilador, definiciones III:

- Compilador incremental (interactivo o conversacional):
  - Si se descubren errores, luego de modificado el fuente, se compilan sólo las modificaciones.
- Autocompilador:
  - Compilador escrito en el propio lenguaje que compila
  - Facilita la portabilidad, ejemplo C.
- Metacompilador:
  - Es un programa que tiene como entrada una gramática y genera el "compilador" del lenguaje definido por la misma, en realidad genera el código del autómata, se debe añadir código para lograr un compilador. Ej: Lex,Yacc

Compiladores e intérpretes

Introducción.

Compilador, definiciones IV:

- Decompilador

Es un programa que tiene como entrada código de máquina y lo traduce a un lenguaje de alto nivel.

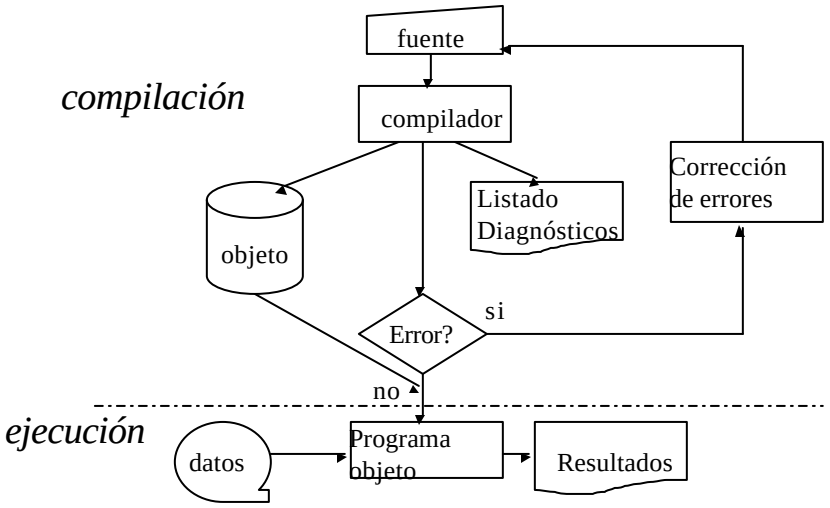
En la práctica es muy difícil "volver atrás", en la práctica existen "desensambladores", sirven para opciones de depuración.

- Preprocesador

Es un proceso anterior a la compilación que permite modificar el programa fuente usando macroinstrucciones y directivas. Por ejemplo en C: #define constante 100. #include "modulo.c"

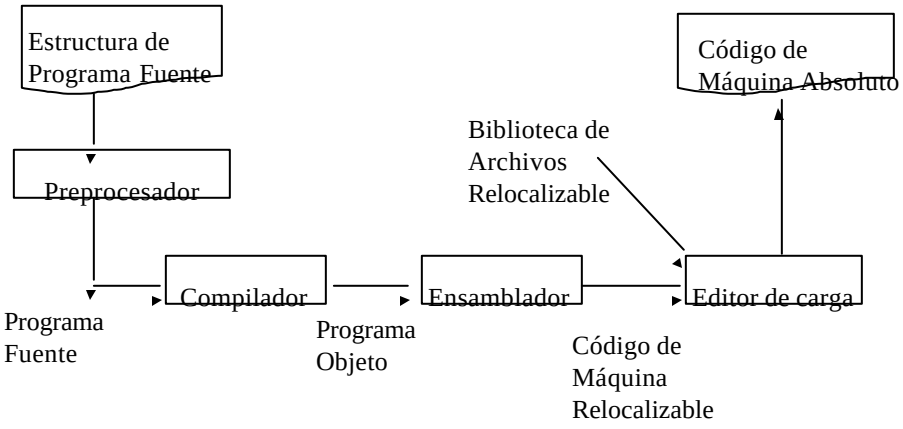
Compiladores e intérpretes

Introducción. Esquema de compilación



Compiladores e intérpretes

Introducción. Esquema de compilador

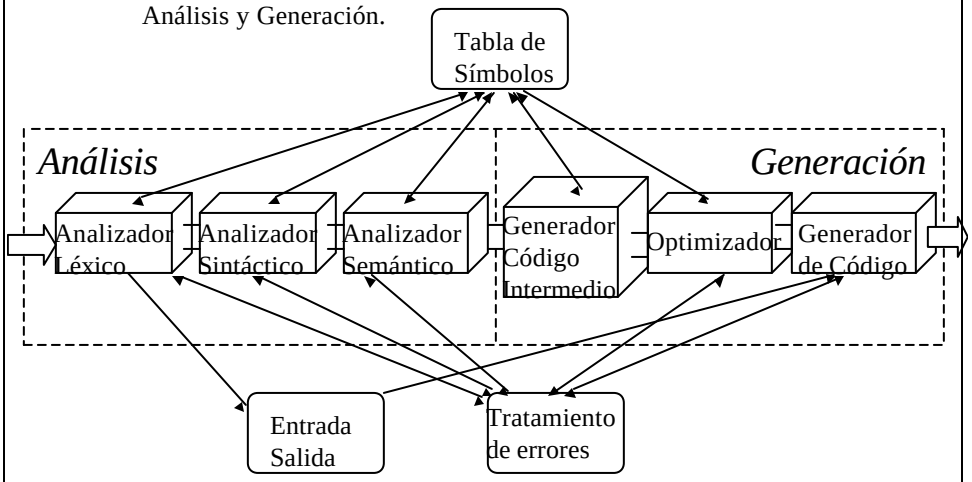


Compiladores e intérpretes

Introducción. Fases de un compilador

– Un compilador se divide en dos fases principales

Análisis y Generación.

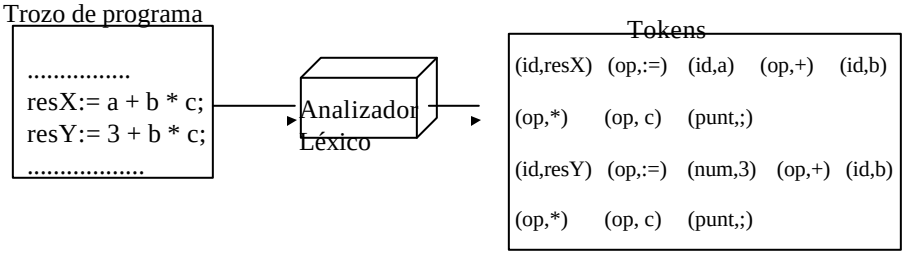


Compiladores e intérpretes

Introducción. Fases de un compilador

– Analizador Léxico (scanner)

– Realiza un análisis lineal del archivo. La cadena de entrada se lee de izquierda a derecha y se va agrupando en componentes léxicos (***TOKENS***), que son secuencias de caracteres con un significado colectivo. Por ejemplo:números, identificadores (variables, constantes, tipos, nombres de funciones...), palabras reservadas, signos de final de instrucción. Cada componente es asociada a la categoría que pertenece.



Compiladores e intérpretes

Introducción. Fases de un compilador

– Analizador sintáctico (parser)

  – Realiza un análisis jerárquico agrupando los componentes léxicos en frases gramaticales que el compilador utiliza. Genera un árbol de derivación. El ejemplo anterior se deriva de la siguiente gramática:

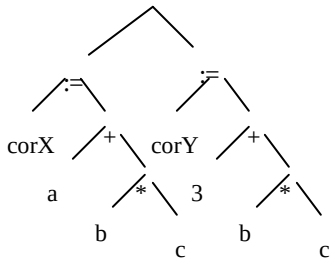
sentencias ::= sentencia **;** sentencias | sentencia
sentencia ::= asignacion | condicional | iterativa
asignacion ::= id **:=** exp
condicional ::= **if** condicion **then** sentencias **else** sentencias;
iteretiva ::= **while** condicion **do** sentencias;
exp ::= id | num | id op exp | num op exp
op := **+** | **-** | ***** | **/**
id ::= [**A-Za-z**] [**A-Za-z0-9**]*
num ::= [**0-9**]*

Compiladores e intérpretes

Introducción. Fases de un compilador

– Analizador semántico

  – Busca errores semánticos, reúne información sobre los tipos; identifica operadores y operandos en base al árbol sintáctico producido en el análisis anterior. Ejemplos de error: operación entre tipos de datos incompatibles, rangos permitidos, existencia de variables. En cualquiera de estos tres análisis pueden producirce errores.



Compiladores e intérpretes

Introducción. Fases de un compilador

- Generador de código intermedio
  - Algunos compiladores generan una representación explícita del programa fuente. Este código es independiente de la máquina y a veces se usa en conjunto con intérpretes, en lenguajes independientes de la plataforma como JAVA. Esta representación debe ser fácil de producir, ayudar a la optimización y fácil de traducir al programa objeto.
  - Ejemplo: código de tres direcciones, donde cada instrucción tiene como máximo tres operandos, supone una CPU en la que cada posición de memoria puede actuar como un registro de la CPU. Reglas:
    - Cada instrucción de tres direcciones tiene a lo sumo un operador (además de la asignación). Puede tener menos.
    - El compilador debe generar un nombre temporal para guardar los valores calculados por cada instrucción.
    - Ejemplo para la segunda instrucción:

      t1 = b*c

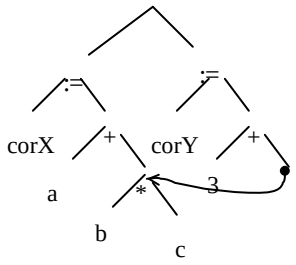
      t2 = int-to-real(3)

      corX = t1 + t2

Compiladores e intérpretes

Introducción. Fases de un compilador

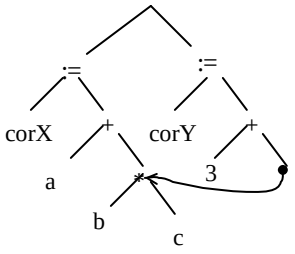
- Optimización
  - Esta fase trata de mejorar el código intermedio, o las estructuras que generarán el código definitivo, de modo de que resulte un código de máquina más rápido de ejecutar.
  - Ejemplo: en las dos asignaciones se puede apreciar que b*c es ocupado an ambas sentencias por lo que:



Compiladores e intérpretes

Introducción. Fases de un compilador

- Generador de código
  - Esta fase final de un compilador. Genera el código objeto, que por lo general consiste en código de máquina relocalizable o código ensamblador.Las posiciones de memoria relativas se seleccionan para cada variable. El uso de los registros de la CPU es relevante.




```
Push a              a ->pila
Push b              b->pila
Load ( c), R1       c->R1
Mult   S,  R1       b*c->R1
Store R1, R2        R1->R2
Add    S, R1        a+b*c->R1
Store R1,(corX) R1 ->CorxX
Add  #3, R2         3+b*c->R2
Store R2,(corY) R1 ->corY
```

Compiladores e intérpretes

Introducción. Fases de un compilador

- Agrupación lógica de un compilador
  - Fase de análisis, dependen del lenguaje fuente y son independientes de la máquina. Controla la corrección del programa fuente, manejando errores en cada etapa. Produce las estructuras necesarias para la generación de código.
  - Fase de generación, depende de la máquina y del lenguaje intermedio.Hace uso intensivo de la tabla de símbolos.