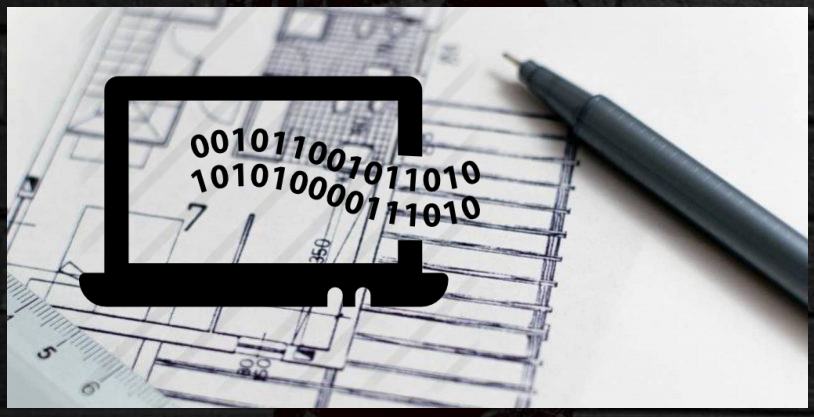
Arquitectura de los compiladores e intérpretes

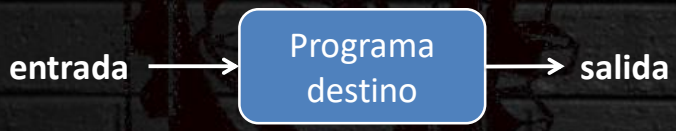


Compilador

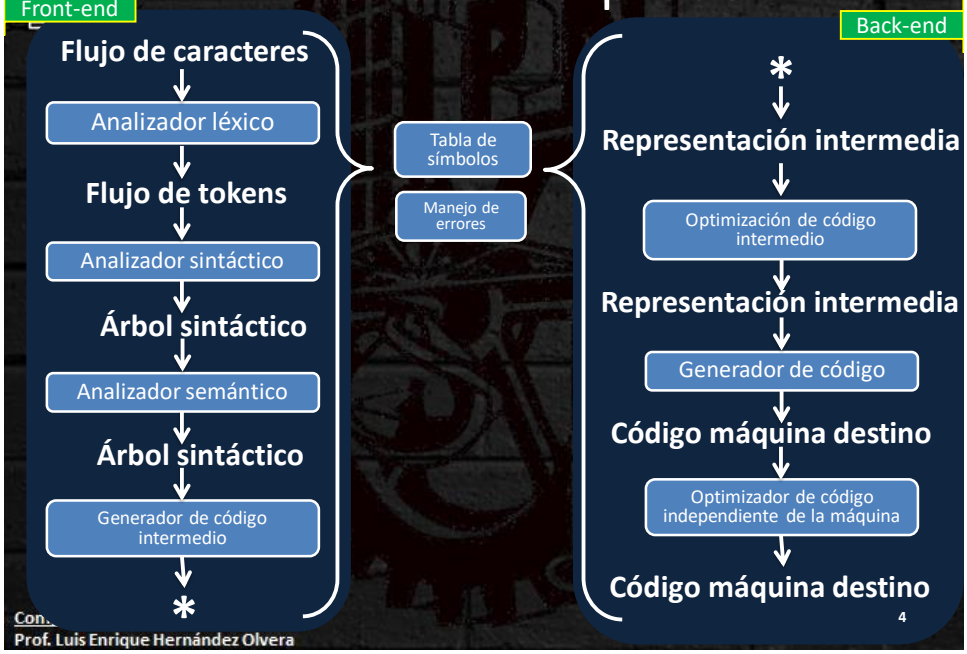
- Un compilador es un programa que puede leer un programa en un lenguaje formal (lenguaje fuente) y traducirlo en un programa equivalente en otro lenguaje formal (lenguaje destino).

Compilador

- Si el programa destino es un programa ejecutable en lenguaje máquina, entonces el usuario puede ejecutarlo para procesar las entradas y producir salidas (resultados).

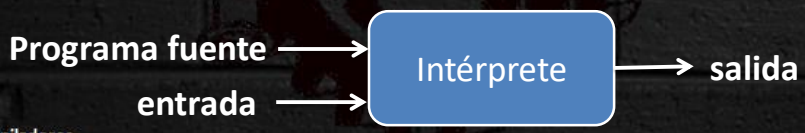


Fases del Compilador

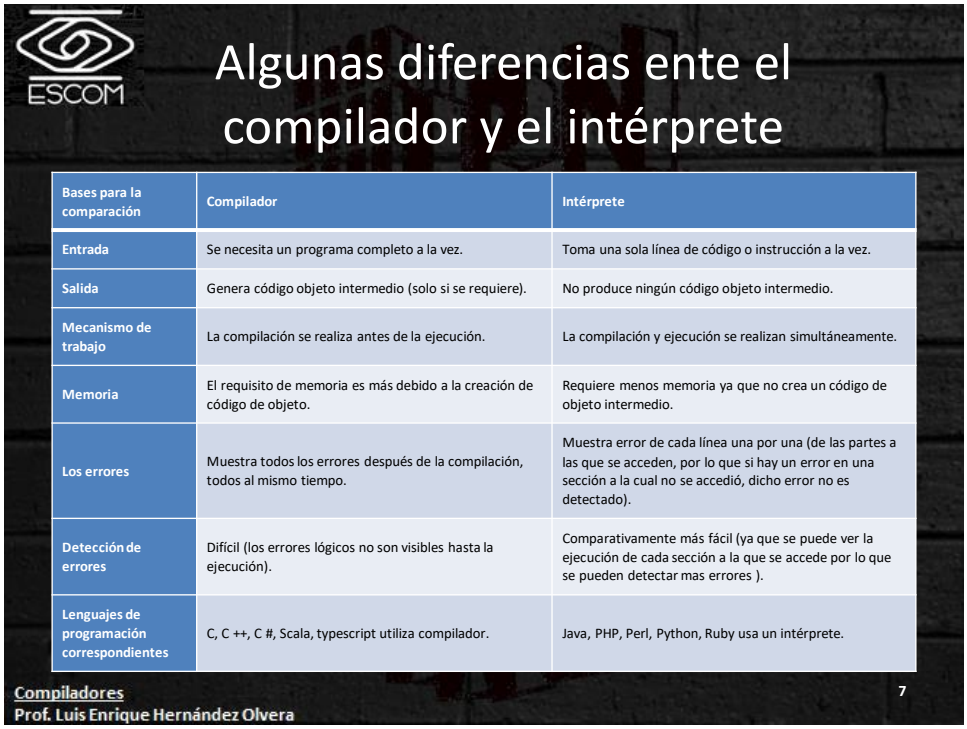


Intérprete

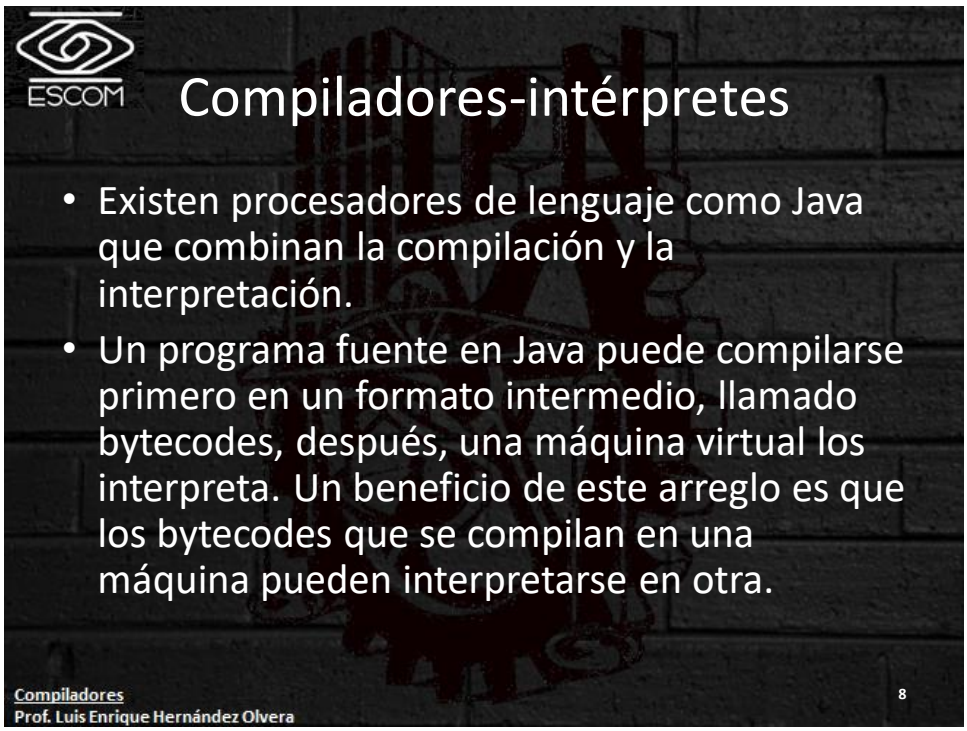
- Un intérprete es otro tipo común de procesador de lenguaje. En vez de producir un programa destino como una traducción, el intérprete ejecuta directamente las operaciones especificadas en el programa de origen (fuente) con las entradas proporcionadas por el usuario



Fases del Intérprete



Bases para la comparación	Compilador	Intérprete
Entrada	Se necesita un programa completo a la vez.	Toma una sola línea de código o instrucción a la vez.
Salida	Genera código objeto intermedio (solo si se requiere).	No produce ningún código objeto intermedio.
Mecanismo de trabajo	La compilación se realiza antes de la ejecución.	La compilación y ejecución se realizan simultáneamente.
Memoria	El requisito de memoria es más debido a la creación de código de objeto.	Requiere menos memoria ya que no crea un código de objeto intermedio.
Los errores	Muestra todos los errores después de la compilación, todos al mismo tiempo.	Muestra error de cada línea una por una (de las partes a las que se acceden, por lo que si hay un error en una sección a la cual no se accedió, dicho error no es detectado).
Detección de errores	Difícil (los errores lógicos no son visibles hasta la ejecución).	Comparativamente más fácil (ya que se puede ver la ejecución de cada sección a la que se accede por lo que se pueden detectar mas errores).
Lenguajes de programación correspondientes	C, C ++, C #, Scala, typescript utiliza compilador.	Java, PHP, Perl, Python, Ruby usa un intérprete.

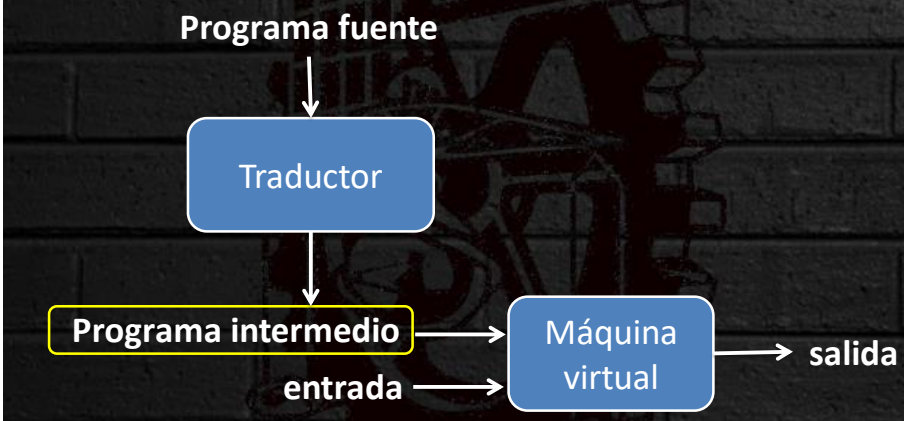


Compiladores-intérpretes

Su funcionamiento tiene lugar en dos fases diferentes:

- La fase de compilación o de traducción del programa: El programa de partida se compila y se traduce a un formato o lenguaje intermedio. Esta operación se realiza usualmente una sola vez.
- La fase de interpretación o de ejecución del programa: El programa compilado al formato o lenguaje intermedio se interpreta y se ejecuta. Esta operación se realiza tantas veces como se desee ejecutar el programa.

Compiladores-intérpretes

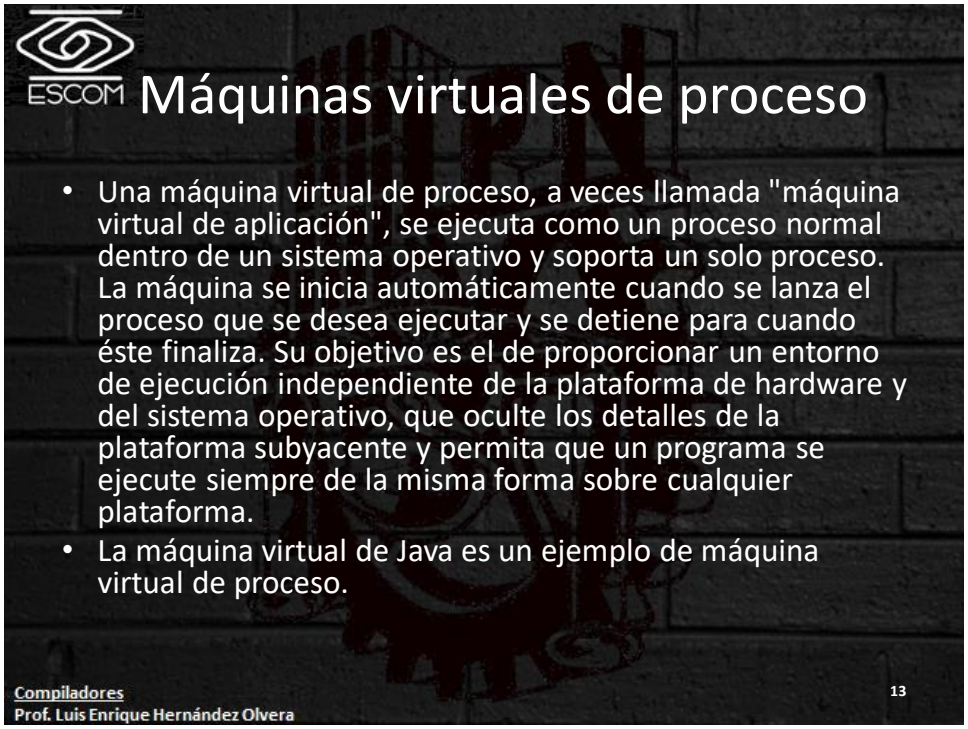


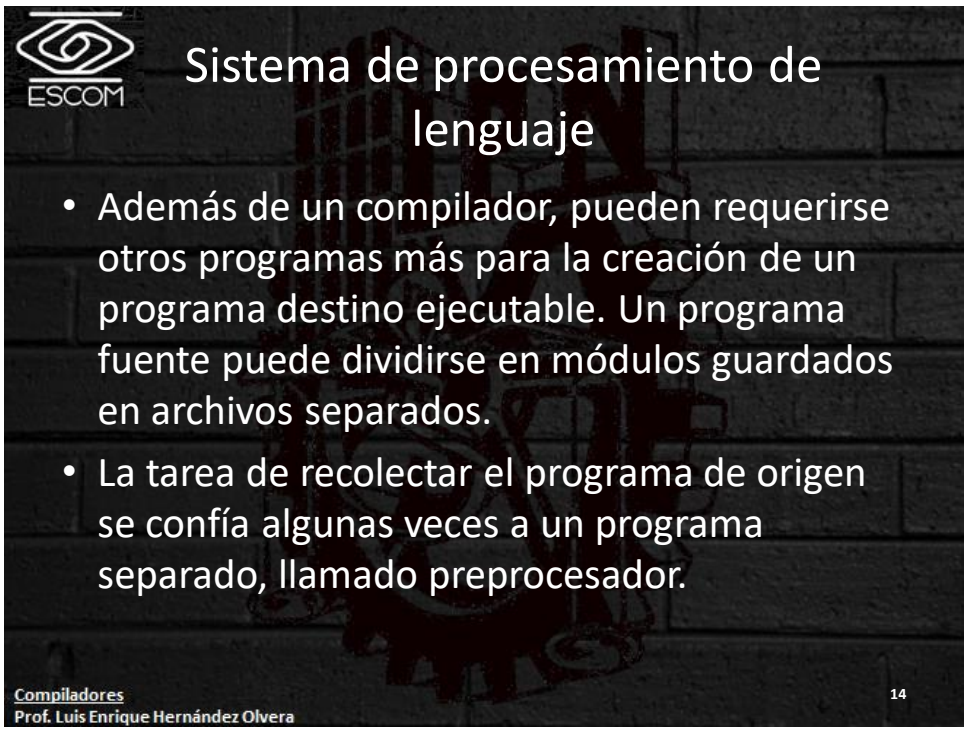
Máquina virtual

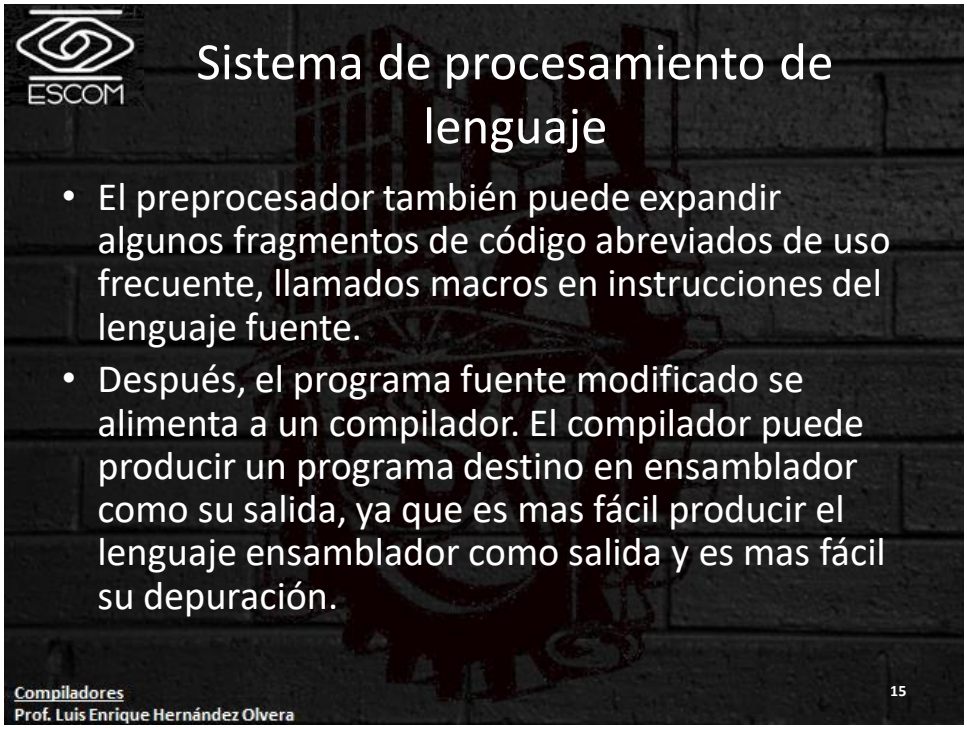
- Una máquina virtual es un contenedor de software perfectamente aislado que puede ejecutar sus propios sistemas operativos y aplicaciones como si fuera un ordenador físico.
- Los tipos de máquinas virtuales se pueden clasificar en dos grandes categorías según su funcionalidad y su grado de equivalencia a una verdadera máquina.
  - Máquinas virtuales de sistema (en inglés System Virtual Machine)
  - Máquinas virtuales de proceso (en inglés Process Virtual Machine)

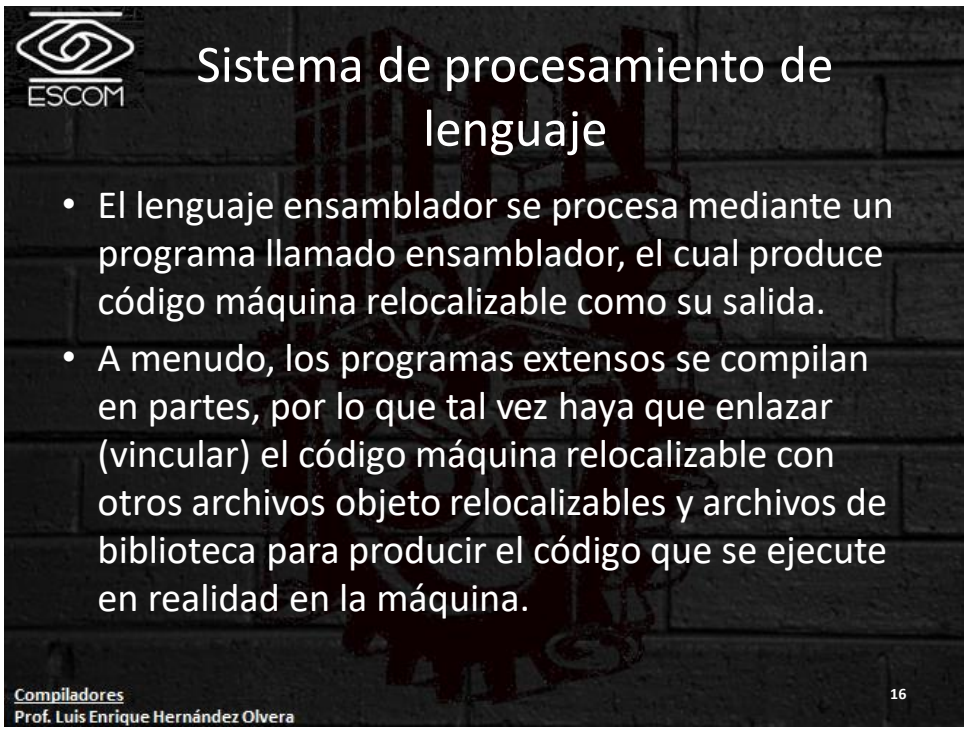
Máquinas virtuales de sistema

- Las máquinas virtuales de sistema, también llamadas máquinas virtuales de hardware, permiten a la máquina física subyacente multiplicarse entre varias máquinas virtuales, cada una ejecutando su propio sistema operativo. A la capa de software que permite la virtualización se la llama monitor de máquina virtual o hypervisor. Un monitor de máquina virtual puede ejecutarse o bien directamente sobre el hardware o bien sobre un sistema operativo ("host operating system").
- Varios sistemas operativos distintos pueden coexistir sobre la misma computadora, en sólido aislamiento el uno del otro, por ejemplo para probar un sistema operativo nuevo sin necesidad de instalarlo directamente. Algunos ejemplos son:
  - Oracle VM VirtualBox
  - VMware Workstation


ESCOM
Máquinas virtuales de proceso
• Una máquina virtual de proceso, a veces llamada "máquina virtual de aplicación", se ejecuta como un proceso normal dentro de un sistema operativo y soporta un solo proceso. La máquina se inicia automáticamente cuando se lanza el proceso que se desea ejecutar y se detiene para cuando éste finaliza. Su objetivo es el de proporcionar un entorno de ejecución independiente de la plataforma de hardware y del sistema operativo, que oculte los detalles de la plataforma subyacente y permita que un programa se ejecute siempre de la misma forma sobre cualquier plataforma.
• La máquina virtual de Java es un ejemplo de máquina virtual de proceso.
Compiladores
Prof. Luis Enrique Hernández Olvera
13


ESCOM
Sistema de procesamiento de lenguaje
• Además de un compilador, pueden requerirse otros programas más para la creación de un programa destino ejecutable. Un programa fuente puede dividirse en módulos guardados en archivos separados.
• La tarea de recolectar el programa de origen se confía algunas veces a un programa separado, llamado preprocesador.
Compiladores
Prof. Luis Enrique Hernández Olvera
14


ESCOM
Sistema de procesamiento de lenguaje
• El preprocesador también puede expandir algunos fragmentos de código abreviados de uso frecuente, llamados macros en instrucciones del lenguaje fuente.
• Después, el programa fuente modificado se alimenta a un compilador. El compilador puede producir un programa destino en ensamblador como su salida, ya que es mas fácil producir el lenguaje ensamblador como salida y es mas fácil su depuración.
Compiladores
Prof. Luis Enrique Hernández Olvera
15


ESCOM
Sistema de procesamiento de lenguaje
• El lenguaje ensamblador se procesa mediante un programa llamado ensamblador, el cual produce código máquina relocalizable como su salida.
• A menudo, los programas extensos se compilan en partes, por lo que tal vez haya que enlazar (vincular) el código máquina relocalizable con otros archivos objeto relocalizables y archivos de biblioteca para producir el código que se ejecute en realidad en la máquina.
Compiladores
Prof. Luis Enrique Hernández Olvera
16

Sistema de procesamiento de lenguaje

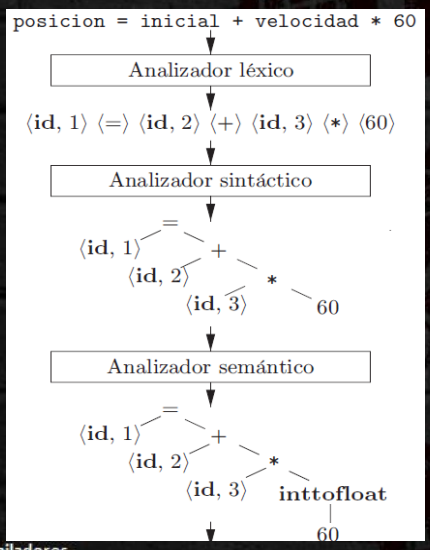
- El enlazador resuelve las direcciones de memoria externas, en donde el código en un archivo puede hacer referencia a una ubicación en otro archivo. Entonces, el cargador reúne todos los archivos objeto ejecutables en la memoria para su ejecución.

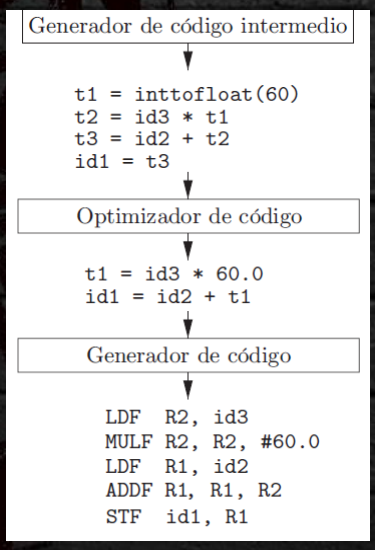
Sistema de procesamiento de lenguaje

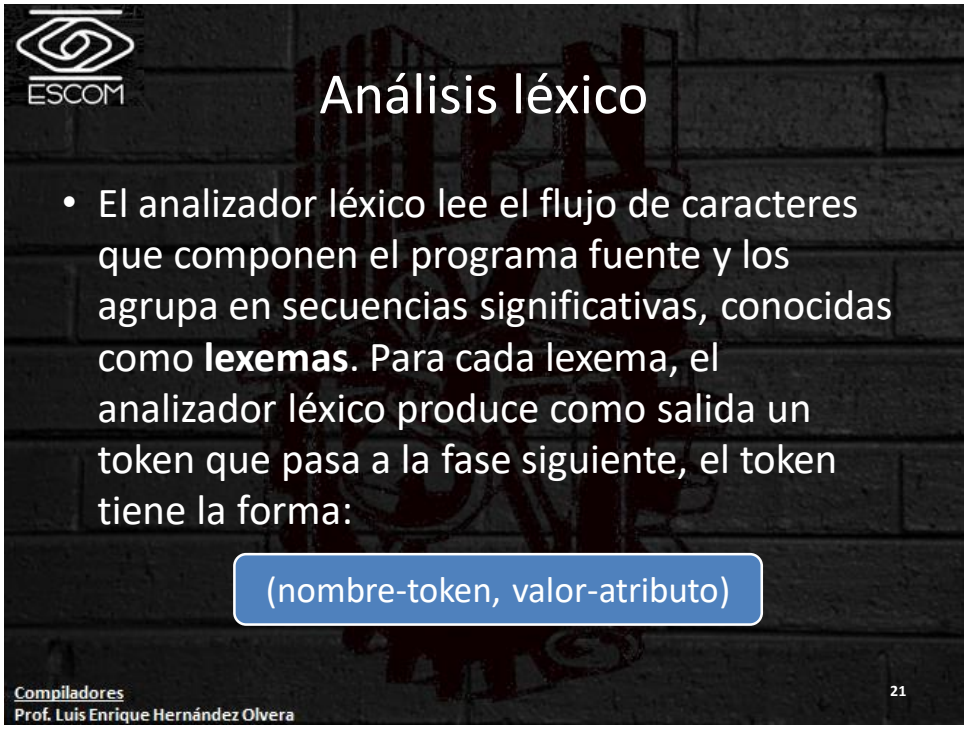
Lenguaje máquina relocalizable

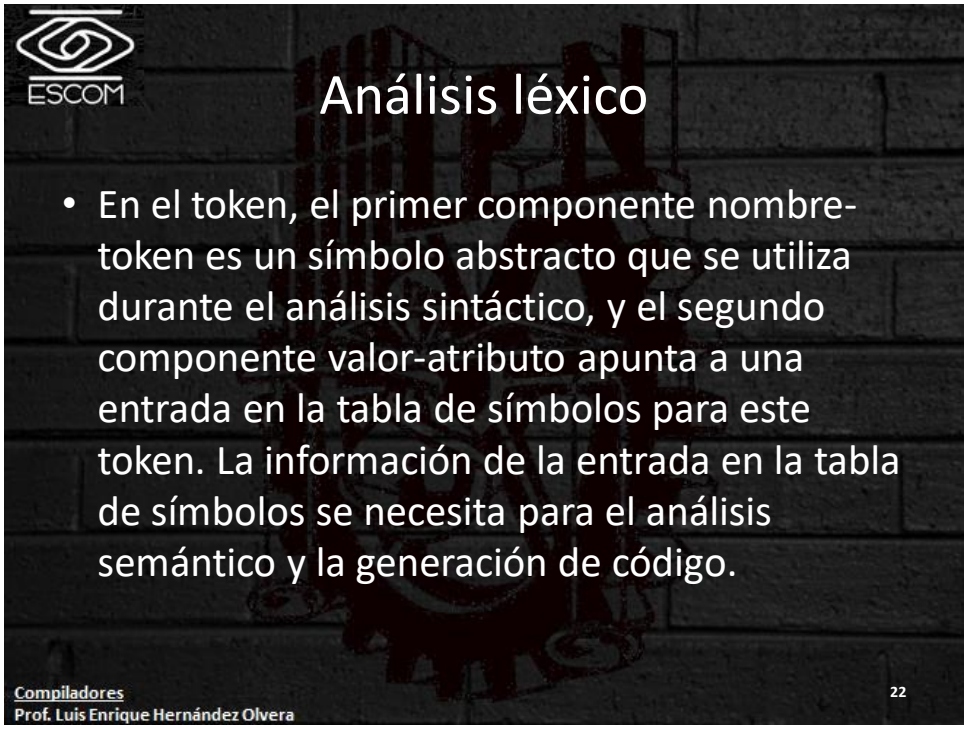
- Está constituido por módulos objeto. El código se genera con desplazamientos de direcciones, lo cual permite enlazar diferentes módulos objeto (ej., bibliotecas).
  - Muy flexible, permite compilar rutinas por separado y llamar a rutinas ya compiladas en otros módulos.
  - Método más utilizado en compiladores comerciales
  - Es *necesario un enlazador para crear el ejecutable*.

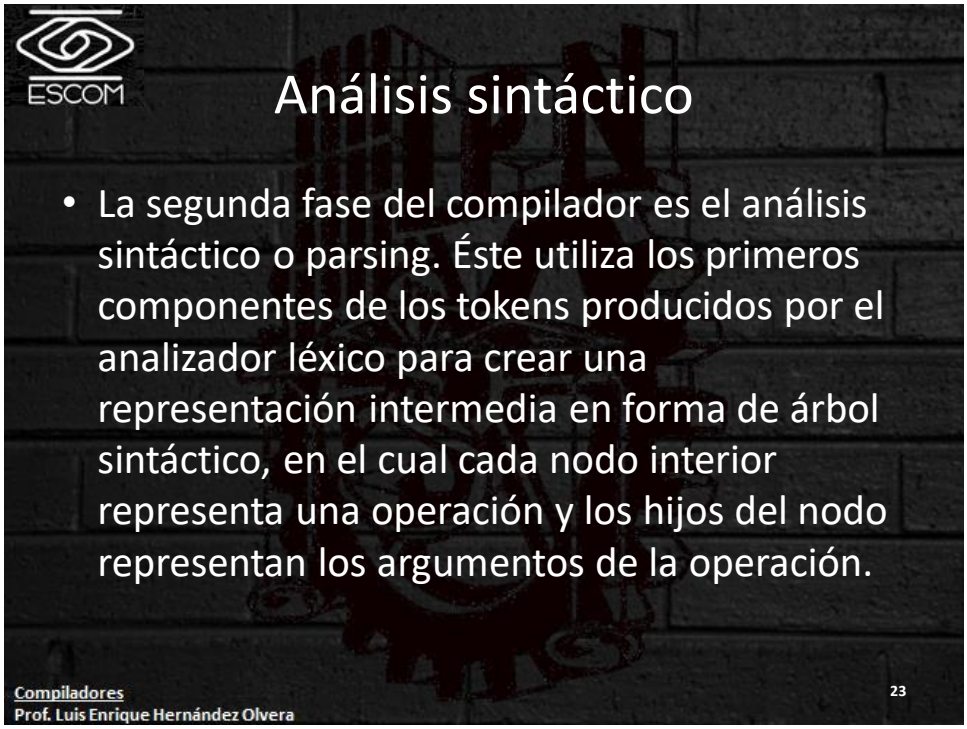
Estructura de un compilador

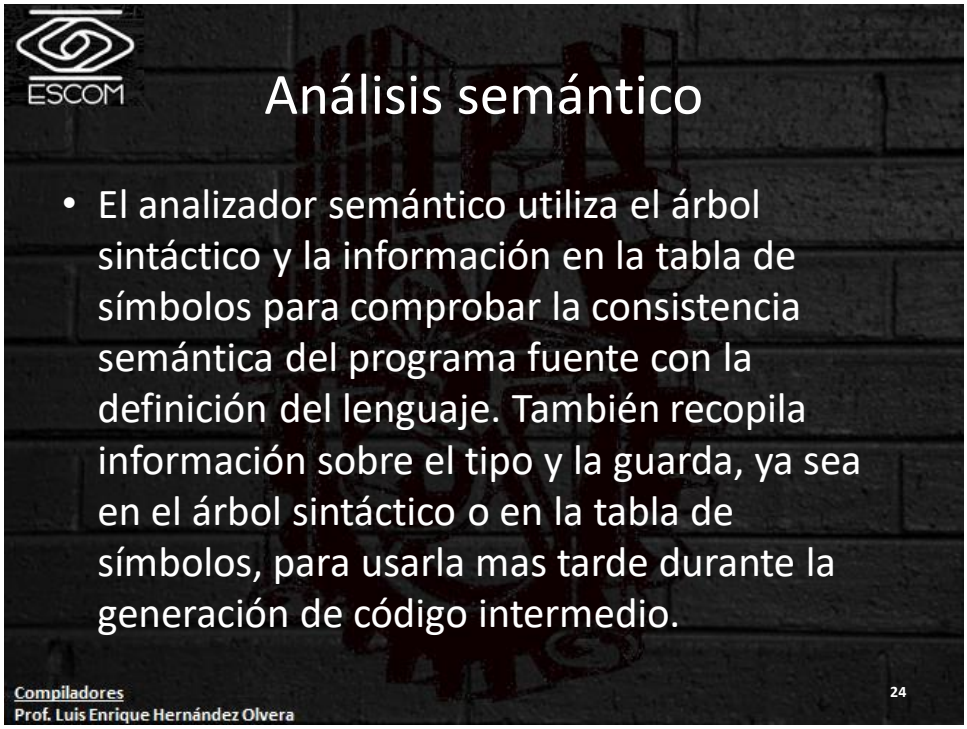


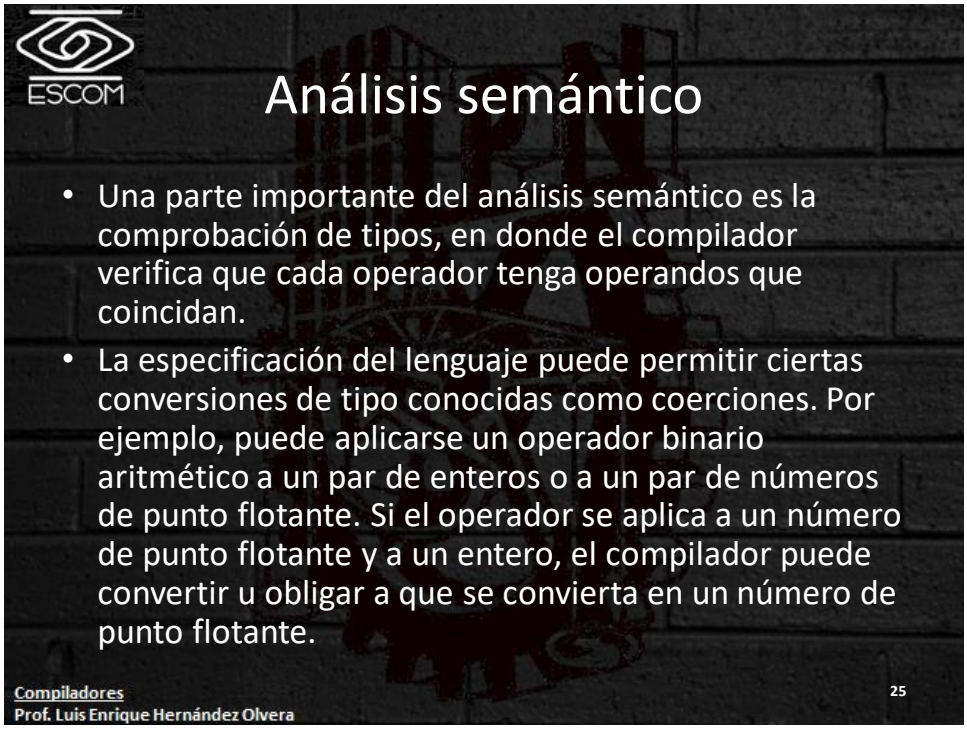


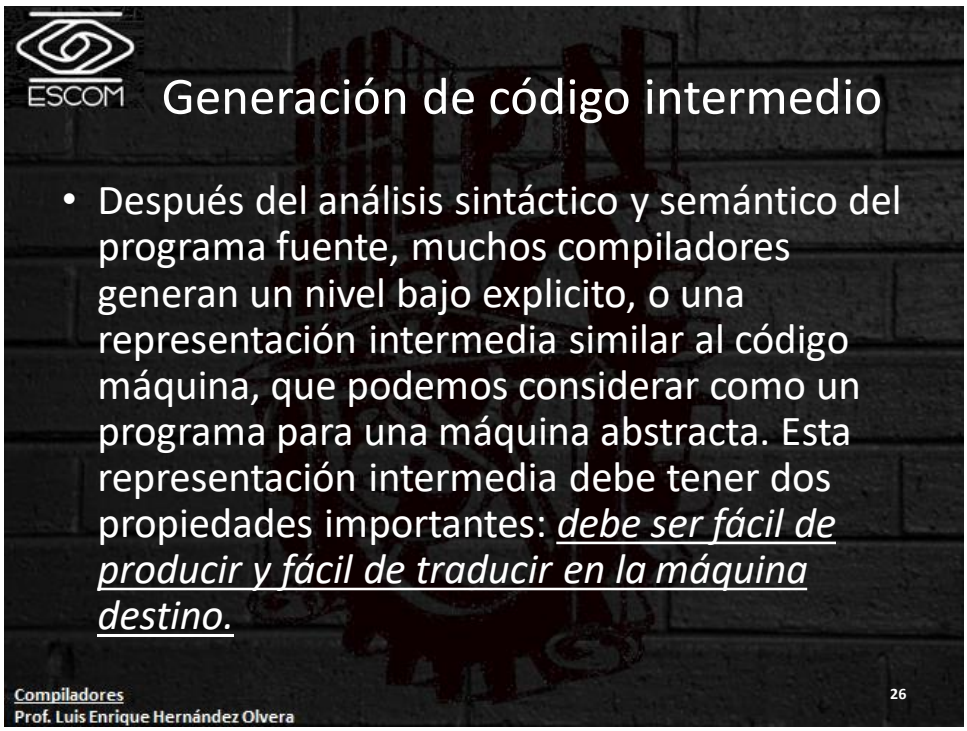

ESCOM
Análisis léxico
• El analizador léxico lee el flujo de caracteres que componen el programa fuente y los agrupa en secuencias significativas, conocidas como **lexemas**. Para cada lexema, el analizador léxico produce como salida un token que pasa a la fase siguiente, el token tiene la forma:
(nombre-token, valor-atributo)
Compiladores
Prof. Luis Enrique Hernández Olvera
21

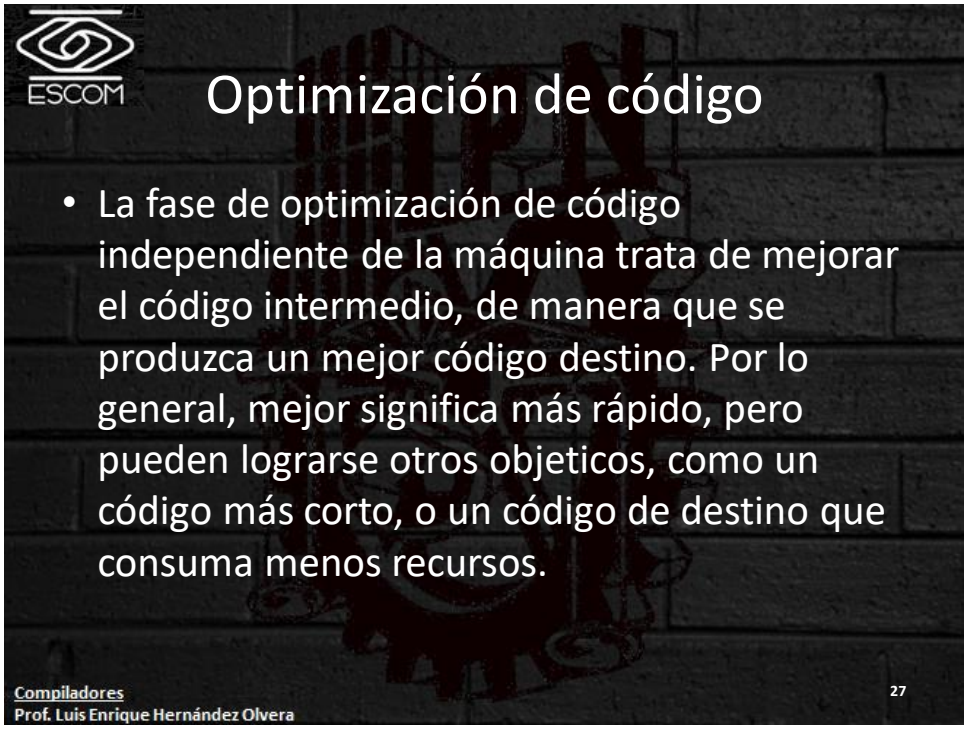

ESCOM
Análisis léxico
• En el token, el primer componente nombre-token es un símbolo abstracto que se utiliza durante el análisis sintáctico, y el segundo componente valor-atributo apunta a una entrada en la tabla de símbolos para este token. La información de la entrada en la tabla de símbolos se necesita para el análisis semántico y la generación de código.
Compiladores
Prof. Luis Enrique Hernández Olvera
22

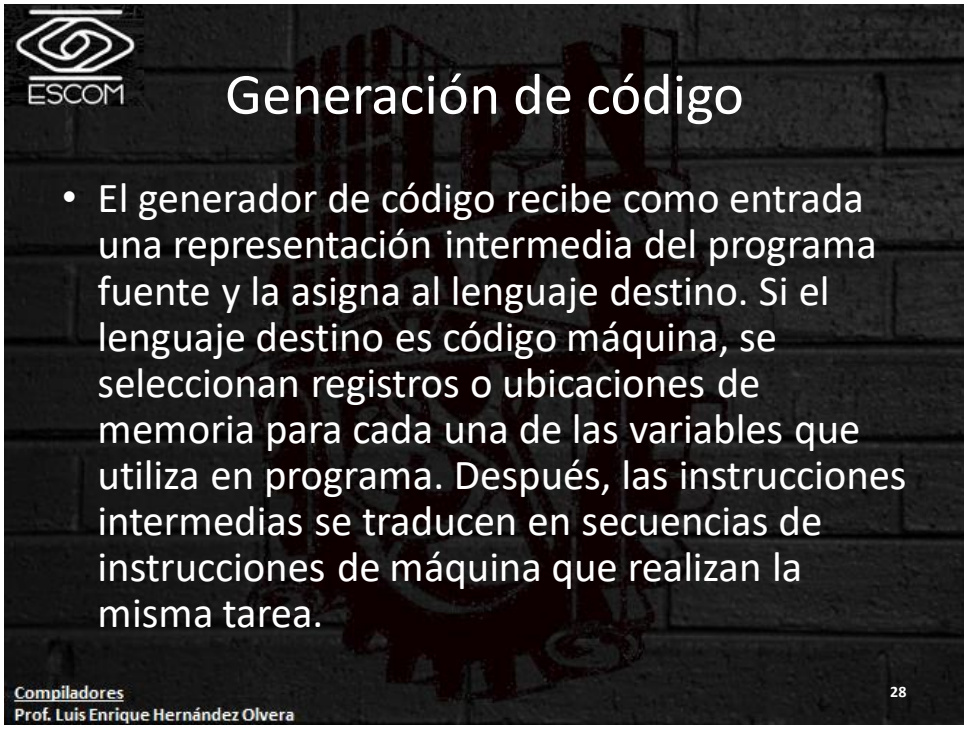

ESCOM
Análisis sintáctico
• La segunda fase del compilador es el análisis sintáctico o parsing. Éste utiliza los primeros componentes de los tokens producidos por el analizador léxico para crear una representación intermedia en forma de árbol sintáctico, en el cual cada nodo interior representa una operación y los hijos del nodo representan los argumentos de la operación.
Compiladores
Prof. Luis Enrique Hernández Olvera
23

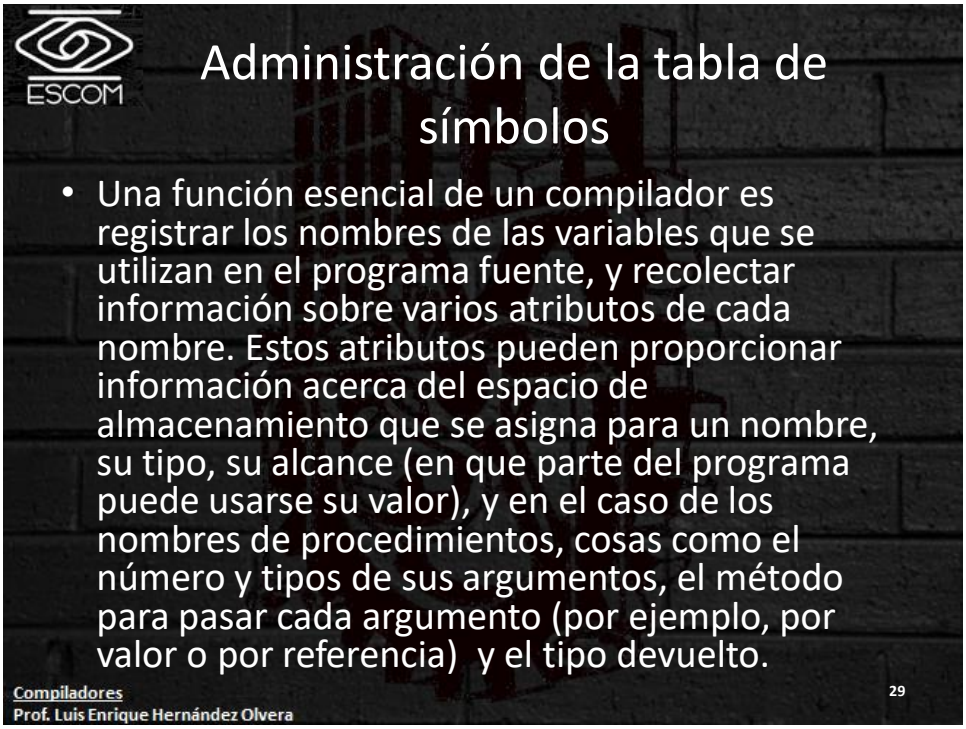

ESCOM
Análisis semántico
• El analizador semántico utiliza el árbol sintáctico y la información en la tabla de símbolos para comprobar la consistencia semántica del programa fuente con la definición del lenguaje. También recopila información sobre el tipo y la guarda, ya sea en el árbol sintáctico o en la tabla de símbolos, para usarla mas tarde durante la generación de código intermedio.
Compiladores
Prof. Luis Enrique Hernández Olvera
24


ESCOM
Análisis semántico
• Una parte importante del análisis semántico es la comprobación de tipos, en donde el compilador verifica que cada operador tenga operandos que coincidan.
• La especificación del lenguaje puede permitir ciertas conversiones de tipo conocidas como coerciones. Por ejemplo, puede aplicarse un operador binario aritmético a un par de enteros o a un par de números de punto flotante. Si el operador se aplica a un número de punto flotante y a un entero, el compilador puede convertir u obligar a que se convierta en un número de punto flotante.
Compiladores
Prof. Luis Enrique Hernández Olvera
25


ESCOM
Generación de código intermedio
• Después del análisis sintáctico y semántico del programa fuente, muchos compiladores generan un nivel bajo explicito, o una representación intermedia similar al código máquina, que podemos considerar como un programa para una máquina abstracta. Esta representación intermedia debe tener dos propiedades importantes: *debe ser fácil de producir y fácil de traducir en la máquina destino.*
Compiladores
Prof. Luis Enrique Hernández Olvera
26


ESCOM
Optimización de código
• La fase de optimización de código independiente de la máquina trata de mejorar el código intermedio, de manera que se produzca un mejor código destino. Por lo general, mejor significa más rápido, pero pueden lograrse otros objeticos, como un código más corto, o un código de destino que consuma menos recursos.
Compiladores
Prof. Luis Enrique Hernández Olvera
27


ESCOM
Generación de código
• El generador de código recibe como entrada una representación intermedia del programa fuente y la asigna al lenguaje destino. Si el lenguaje destino es código máquina, se seleccionan registros o ubicaciones de memoria para cada una de las variables que utiliza en programa. Después, las instrucciones intermedias se traducen en secuencias de instrucciones de máquina que realizan la misma tarea.
Compiladores
Prof. Luis Enrique Hernández Olvera
28


ESCOM
Administración de la tabla de símbolos
• Una función esencial de un compilador es registrar los nombres de las variables que se utilizan en el programa fuente, y recolectar información sobre varios atributos de cada nombre. Estos atributos pueden proporcionar información acerca del espacio de almacenamiento que se asigna para un nombre, su tipo, su alcance (en que parte del programa puede usarse su valor), y en el caso de los nombres de procedimientos, cosas como el número y tipos de sus argumentos, el método para pasar cada argumento (por ejemplo, por valor o por referencia) y el tipo devuelto.
Compiladores
Prof. Luis Enrique Hernández Olvera
29