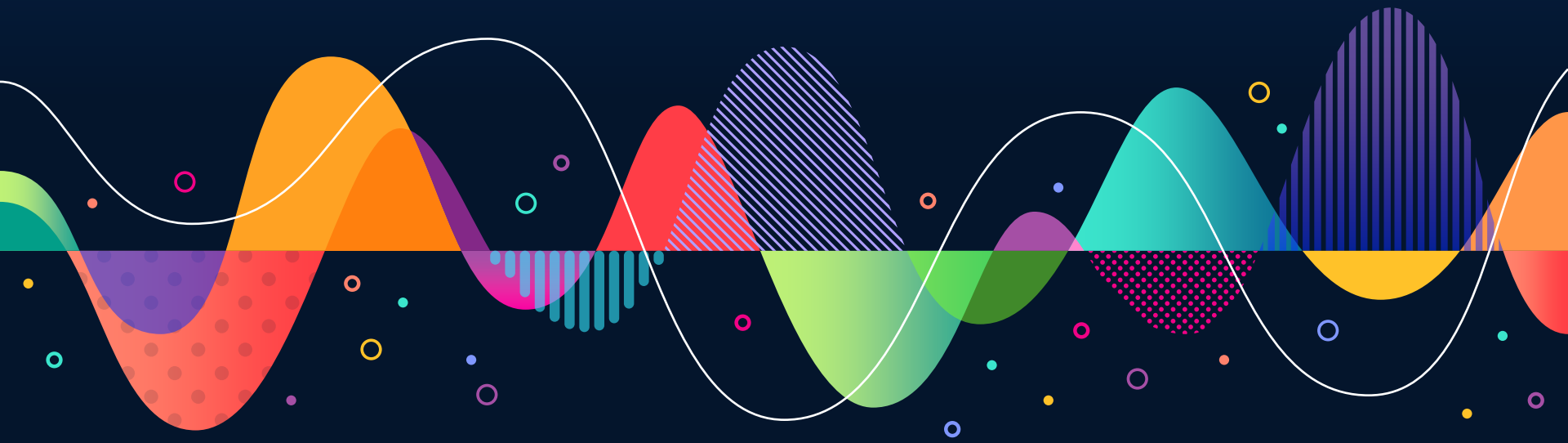
Arquitectura de los compiladores e intérpretes



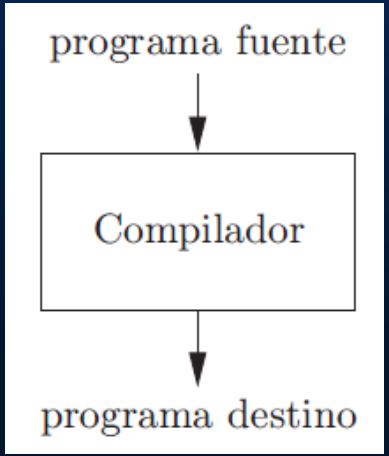
Definición de compiladores e intérpretes

Compilador

- Los lenguajes de programación son notaciones que describen los cálculos a las personas y las máquinas. Nuestra percepción del mundo en que vivimos depende de los lenguajes de programación, ya que todo el software que se ejecuta en todas las computadoras se escribió en algún lenguaje de programación.
- Pero antes de poder ejecutar un programa, primero debe traducirse a un formato en el que una computadora pueda ejecutarlo.
- Los sistemas de software que se encargan de esta traducción se llaman ***compiladores***.

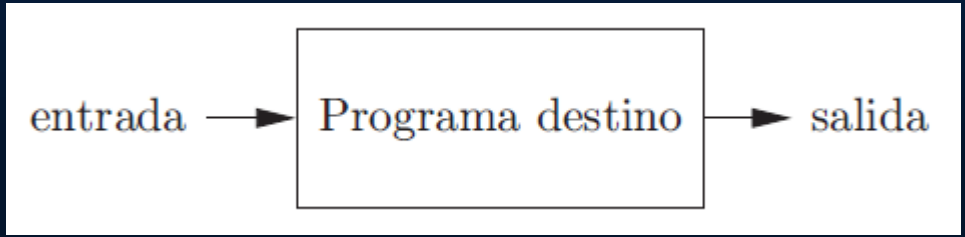
Compilador

- Un compilador es un programa que puede leer un programa en un lenguaje (el lenguaje fuente) y traducirlo en un programa equivalente en otro lenguaje (el lenguaje destino).
- Una función importante del compilador es reportar cualquier error en el programa fuente que detecte durante el proceso de traducción.
- El lenguaje fuente es generalmente un lenguaje de alto nivel, y el lenguaje destino (conocido como objeto) es un lenguaje de bajo nivel (código máquina).



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).



Intérprete

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

Intérprete

- El programa destino en lenguaje máquina que produce un compilador es, por lo general, más rápido que un intérprete al momento de asignar las entradas a las salidas.

- No obstante, por lo regular, el intérprete puede ofrecer mejores diagnósticos de error que un compilador, ya que ejecuta el programa fuente instrucción por instrucción.

Ventajas del compilador VS intérprete

- Algunas de las ventajas de compilar frente a interpretar son:
  - Se compila una vez; se ejecuta muchas veces.
  - La ejecución del programa objeto es mucho más rápida que si se interpreta el programa fuente.
  - El compilador tiene una visión global del programa, por lo que la información de mensajes de error es más detallada.

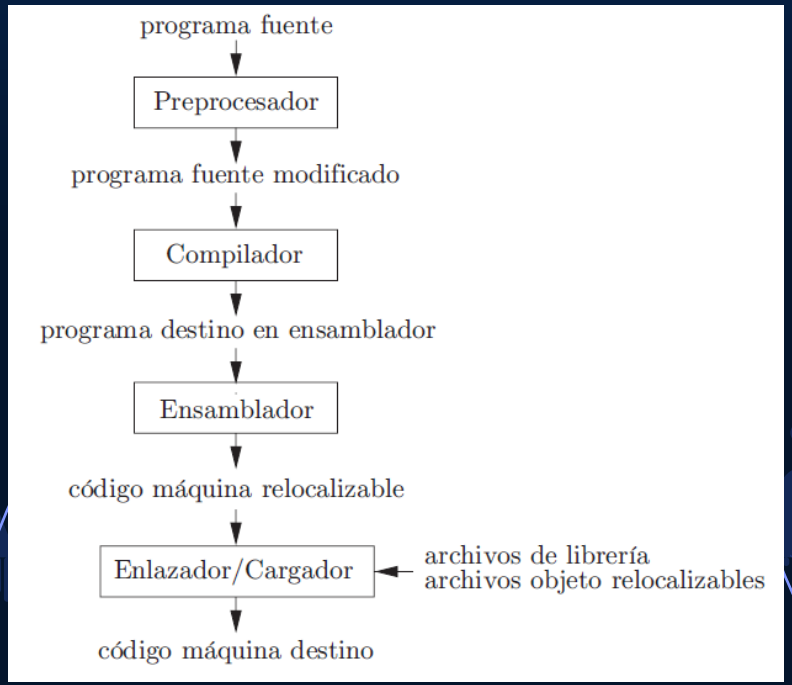
Ventajas del intérprete VS compilador

- Por otro lado, algunas de las ventajas de interpretar frente a compilar son:
- Un interprete necesita menos memoria que un compilador. En las primeras etapas de la informática eran más abundantes dado que los ordenadores tenían poca memoria.
- Permiten una mayor interactividad con el código en tiempo de desarrollo.
- En algunos lenguajes (Smalltalk, Prolog, LISP) está permitido y es frecuente añadir código según se ejecuta otro código, y esta característica solamente es posible implementarla en un intérprete.

Arquitectura de los compiladores e intérpretes

Sistema de procesamiento de lenguaje

- Además de un compilador, pueden requerirse otros programas más para la creación de un programa destino ejecutable.



Sistema de procesamiento de lenguaje

- 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.*

- *El preprocesador* también puede expandir algunos fragmentos de código abreviados de uso frecuente, llamados macros, en instrucciones del lenguaje fuente.

Sistema de procesamiento de lenguaje

- 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 más fácil producir el lenguaje ensamblador como salida y es más fácil su depuración.

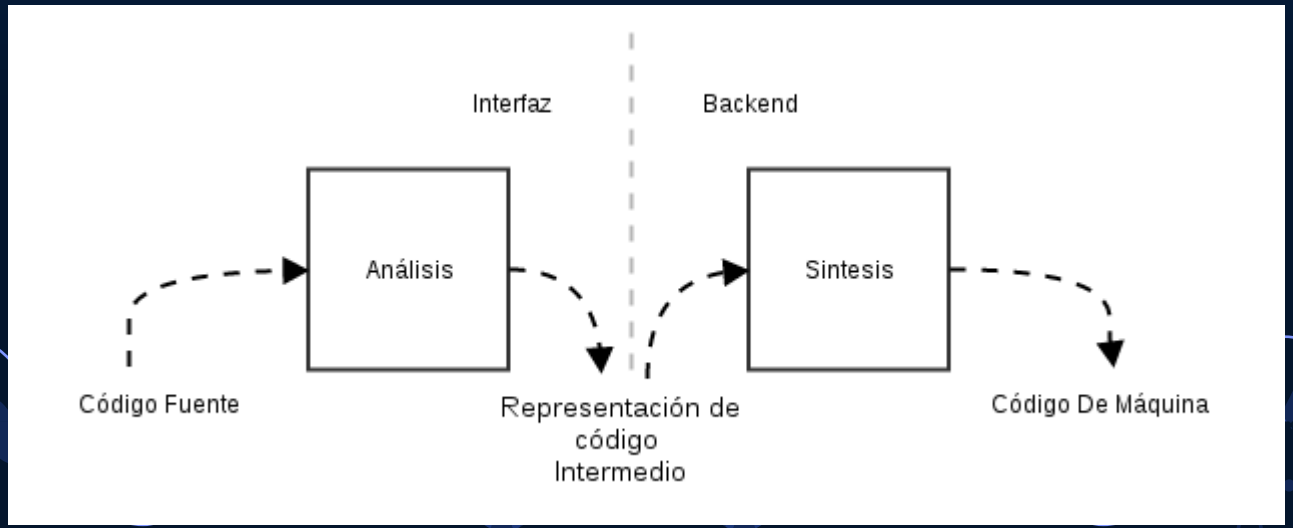
- A continuación, el lenguaje ensamblador se procesa mediante un programa llamado *ensamblador, el cual produce código* máquina relocalizable como su salida.

Sistema de procesamiento de lenguaje

- 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.
- 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.

La estructura de un compilador

- Hasta este punto, hemos tratado al compilador como una caja simple que mapea un programa fuente a un programa destino con equivalencia semántica. Si abrimos esta caja un poco, podremos ver que hay dos procesos en esta asignación: análisis y síntesis.



La estructura de un compilador

- La parte del análisis divide el programa fuente en componentes e impone una estructura gramatical sobre ellas.
- Después utiliza esta estructura para crear una representación intermedia del programa fuente. Si la parte del análisis detecta que el programa fuente está mal formado en cuanto a la sintaxis, o que no tiene una semántica consistente, entonces debe proporcionar mensajes informativos para que el usuario pueda corregirlo. La parte del análisis también recolecta información sobre el programa fuente y la almacena en una estructura de datos llamada tabla de símbolos, la cual se pasa junto con la representación intermedia a la parte de la **síntesis**.

La estructura de un compilador

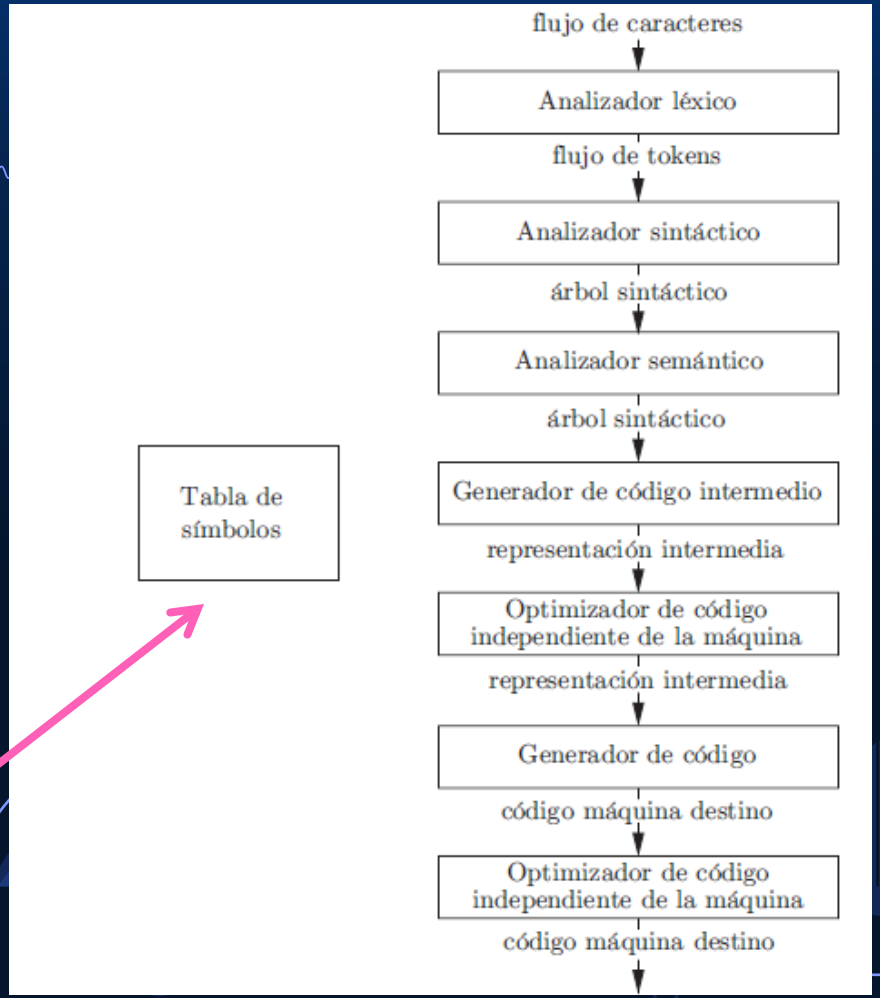
- La parte de la síntesis construye el programa destino deseado a partir de la representación intermedia y de la información en la tabla de símbolos.

- A la parte del análisis se le llama comúnmente el front-end del compilador; la parte de la síntesis (propiamente la traducción) es el back-end.

Fases de un compilador

- El proceso de compilación opera como una secuencia de fases, cada una de las cuales transforma una representación del programa fuente en otro.

La tabla de símbolos, que almacena información sobre todo el programa fuente, se utiliza en todas las fases del compilador.



Máquinas virtuales

Máquinas virtuales

- Una máquina virtual es un software que simula un sistema de computación y puede ejecutar programas como si fuese una computadora real. Este software en un principio fue definido como "un duplicado eficiente y aislado de una máquina física".

Tipos de máquinas virtuales

- **Máquinas virtuales de sistema.**
  Es aquella que emula a un ordenador completo. Es un software que puede hacerse pasar por otro dispositivo de tal modo que puedes ejecutar otro sistema operativo en su interior.

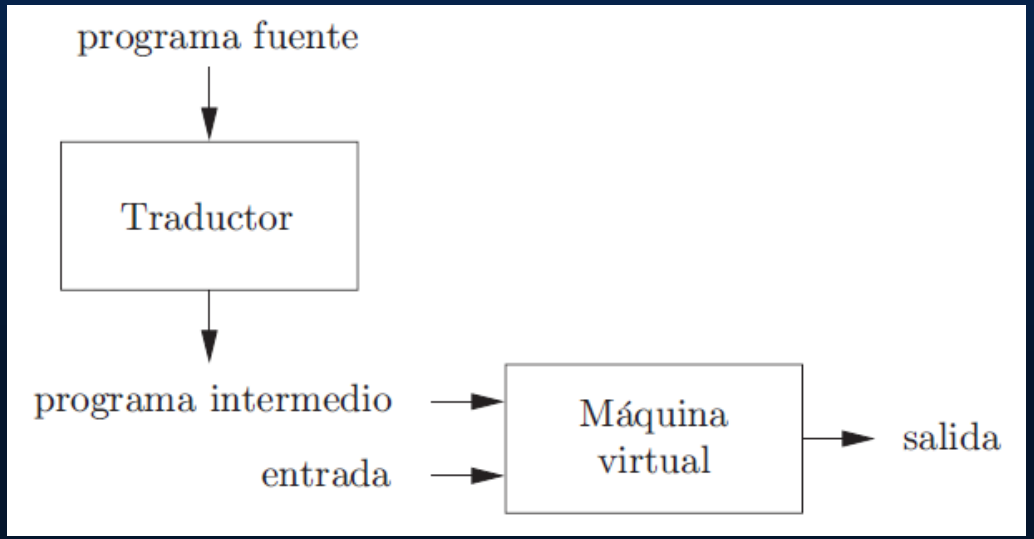
- **Máquinas virtuales de proceso.**
  Ejecutan programas informáticos en un entorno independiente de la plataforma. Enmascaran la información del hardware o el sistema operativo subyacentes. Esto permite que el programa se ejecute de la misma manera en cualquier plataforma.

Máquinas virtuales

- Los procesadores del lenguaje Java combinan la compilación y la interpretación.
- Un programa fuente en Java puede compilarse primero en un formato intermedio, llamado bytecodes. Después, una máquina virtual los interpreta.
- Un beneficio de este arreglo es que los bytecodes que se compilan en una máquina pueden interpretarse en otra, tal vez a través de una red.
- Para poder lograr un procesamiento más rápido de las entradas a las salidas, algunos compiladores de Java, conocidos como compiladores just-in-time (justo a tiempo), traducen los bytecodes en lenguaje máquina justo antes de ejecutar el programa intermedio para procesar la entrada.

Máquinas virtuales

Compilador híbrido



Máquinas virtuales

- El lenguaje Java establece que el compilador no genera código para una máquina determinada sino para una virtual, la Java Virtual Machine (JVM), que posteriormente será ejecutado por un intérprete, normalmente incluido en un navegador de Internet.
- El gran objetivo de esta exigencia es conseguir la máxima portabilidad de los programas escritos y compilados en Java, pues es únicamente la segunda fase del proceso la que depende de la máquina concreta en la que se ejecuta el intérprete.