Investiga sobre las fases del proceso de compilación.

- **Análisis léxico:**

  El análisis léxico es la primera fase en la que el compilador escanea el código fuente (este proceso se puede realizar de izquierda a derecha, carácter por carácter). El código fuente se divide en unidades léxicas más pequeñas llamadas tokens. Estos tokens pueden ser palabras clave, identificadores, operadores, números, etc.
  Aquí, el flujo de caracteres del programa fuente se agrupa en secuencias significativas identificando los tokens. Realiza el ingreso de los tickets correspondientes en la tabla de símbolos y pasa ese token a la siguiente fase.

  Las funciones principales de esta fase son:

  -Identificar las unidades léxicas en un código fuente.

  -Clasificar unidades léxicas en clases como constantes, palabras reservadas e introducirlas en diferentes tablas. Se eliminan los elementos que no son esenciales para la sintaxis del lenguaje, como los comentarios y los espacios en blanco.

  -Identificar token que no forma parte del idioma.

- **Análisis sintáctico:**

  El análisis de sintaxis consiste en descubrir la estructura del código. Determina si un texto sigue o no el formato esperado.

  El objetivo principal de esta fase es asegurarse de que el código fuente escrito por el programador sea correcto o no.
  El análisis de sintaxis se basa en las reglas basadas en el lenguaje de programación específico mediante la construcción del árbol de análisis con la ayuda de tokens. También determina la estructura del idioma de origen y la gramática o sintaxis del idioma.

  Las funciones principales de esta fase son:

  -Obtener tokens del analizador léxico

  -Comprobar si la expresión es sintácticamente correcta o no.

  -Informar todos los errores de sintaxis

  -Construir una estructura jerárquica que se conoce como árbol de análisis.

- **Análisis semántico:**

  El análisis semántico comprueba la coherencia semántica del código. Utiliza el árbol de sintaxis de la fase anterior junto con la tabla de símbolos para verificar que el código fuente dado sea semánticamente consistente. También comprueba si el código transmite un significado apropiado.
  Semantic Analyzer comprobará si hay discrepancias de tipos, operandos incompatibles, una función llamada con argumentos inadecuados, una variable no declarada, etc.

  Las funciones de la fase de análisis semánticos son:

  -Almacenar la información de tipo recopilada y guardarla en una tabla de símbolos o un árbol de sintaxis.

  -Realizar verificación de tipo.

  -En el caso de una discrepancia de tipos, donde no existen reglas de corrección de tipos exactas que satisfagan la operación deseada, se muestra un error semántico.

  -Recopilar información de tipos y comprobar la compatibilidad de tipos.

  -Comprobar si el idioma de origen permite los operandos o no.

- **Generación de código intermedio:**

  Una vez finalizada la fase de análisis semántico, el compilador genera código intermedio para la máquina de destino. Representa un programa para alguna máquina abstracta.
  El código intermedio se encuentra entre el lenguaje de alto nivel y el de máquina. Este código intermedio debe generarse de tal manera que facilite su traducción al código de máquina de destino.
  Funciones de generación de Código Intermedio:

  -Debe generarse a partir de la representación semántica del programa fuente.

  -Mantiene los valores calculados durante el proceso de traducción.

  -Traducir el código intermedio al idioma de destino.

  -Le permite mantener el orden de prioridad del idioma de origen.

  -Contiene el número correcto de operandos de la instrucción.

- **Optimización del código intermedio:**

  La siguiente fase es la optimización del código o código intermedio. Esta fase elimina líneas de código innecesarias y organiza la secuencia de declaraciones para acelerar la ejecución del programa sin desperdiciar recursos. El objetivo principal de esta fase es mejorar el código intermedio para generar un código que se ejecute más rápido y ocupe menos espacio.

Las funciones principales de esta fase son:

-Le ayuda a establecer un equilibrio entre la velocidad de ejecución y de compilación.

-Mejorar el tiempo de ejecución del programa de destino.

-Genera código simplificado aún en representación intermedia

-Eliminar código inalcanzable y deshacerse de variables no utilizadas

-Eliminar declaraciones que no se modifican del bucle

- **Generación del código final:**

  La generación de código es la última fase . Obtiene entradas de las fases de optimización del código y, como resultado, produce el código de página o el código objeto. El objetivo de esta fase es asignar almacenamiento y generar código de máquina reubicable.
  También asigna ubicaciones de memoria para la variable. Las instrucciones del código intermedio se convierten en instrucciones de máquina. Esta fase convierte el código optimizado o intermedio al idioma de destino.
  El idioma de destino es el código de máquina. Por lo tanto, todas las ubicaciones de memoria y registros también se seleccionan y asignan durante esta fase. El código generado por esta fase se ejecuta para tomar entradas y generar resultados esperados.