Escuela Politécnica Nacional

Nombre: Fernando Elicio Huilca Villagómez Fecha: 11/05/2025

Tablas de Símbolos. - Las tablas de símbolos son estructuras de datos que almacenan información asociada a los identificadores que aparecen en el programa fuente. Incluyen detalles como el tipo de dato, el nombre, la posición de memoria y el contexto o ámbito donde están definidos. Las características claves son:

Gestión por alcance (scope). - Cada bloque del programa tiene su propia tabla de símbolos. Esto permite implementar reglas de visibilidad y ocultamiento de variables.

Encadenamiento de tablas: Las tablas se enlazan jerárquicamente, permitiendo la búsqueda de identificadores en contextos externos sino se encuentran en el actual.

Operaciones básicas: Incluir (put) y buscar (get) indentificadores dentro de la tabla. Estas operaciones se activan durante la traducción del código fuente, como la producción del factor → id.

El uso correcto de tabla de símbolos permite que el compilador determine tipos, valide declaraciones y prepare adecuadamente la asignación de memoria y código de intermedio.

Generación de código intermedio.

El código intermedio es una representación abstracta del programa que sirve como puente entre el análisis del código fuente y la generación del código máquina.

Las representaciones principales son:

- Arboles sintácticos abstractos: Capturan la estructura del programa mediantes nodos jerárquicos. Se utilizan para validar la sintaxis

y propagan atributos.

Código de tres direcciones (TAC): Una forma más lineal, donde cada instrucción representa una operación básica del tipo x = y op z. Es ideal para realizar optimizaciones y se adapta bien a la generación de código de máquina.

El código intermedio puede dividirse en bloques básicos, que son secuencias de instrucciones que se ejecutan sin interrupciones internas. Estos bloques son únicos para optimizar localmente el código, mejorar la asignación de registros y facilitar el análisis de flujo de control.

Traducción y generación. - Durante esta etapa, el compilador realiza una comprobación estática, que verifica errores de tipos y el cumplimiento de las reglas del lenguaje antes de ejecutar el programa.

Se presentan estrategias para generar instrucciones parciales, que luego se completan con nombres temporales. Por ejemplo, si se tiene una operación j + k, el compilador puede generar un resultado parcial sin determinar de inmediato su destino lo que permite una mayor flexibilidad para posteriores optimizaciones.

Código de tres direcciones. - Se basa en una forma plana, más cercano al lenguaje ensamblador, facilita la transformación a otras formas. Ideal para aplicar técnicas de optimización como propagación de copias, reducción de subexpresiones comunes y movimiento de código invariantes fuera de bucles.