UNIVERSIDAD SAN PABLO DE GUATEMALA

Facultad de Ciencias Empresariales

Ingeniería en sistemas y ciencias de la computación

Generación de código intermedio y optimización en compiladores modernos

Arquitectura de Negocios

M.A. Walter Ordoñez

Harleth Daniel Garcia Alfaro

Carné: 1900348

“Generación de Código Intermedio y Optimización en Compiladores modernos”

Introducción:

En un compilador moderno, la generación de código intermedio es la etapa situada después del análisis semántico y antes de la generación de código máquina. Su propósito es transformar el código fuente a una representación intermedia mas sencilla, independiente del hardware y más fácil de optimizar.

El código intermedio permite que el compilador reduzca errores, mejore el rendimiento y aplique optimizaciones como eliminación de redundancias, propagación de constantes y uso eficiente de memoria. Sin esta capa intermedia, cada lenguaje debería generar código especifico para cada arquitectura, lo cual sería lento y costoso.

Marco Teórico

El código intermedio

(IR – Internmediate Representation) es una estructura que representa el programa de manera simplificada. Lo más comunes son:

- Código de 3 direcciones (Three Address Code - TAC): una instrucción contiene máximo 2 operandos y un resultado, por ejemplo:

  t1 = a + b

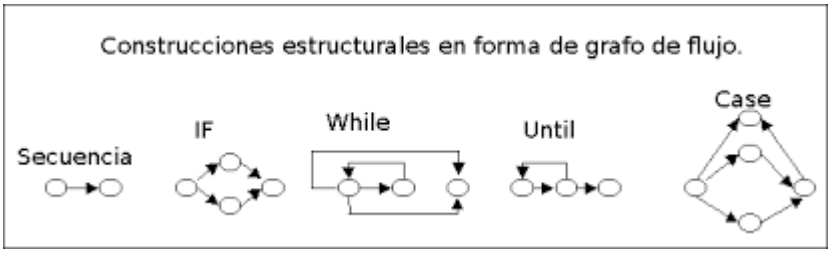
  t2 = t1 + c

  x = t2

- Árbol de Sintaxis Abstracta (AST): Representa la estructura jerárquica de una expresión:

```
          (*)
         /   \
       (+)    c
      /   \
     a     b
```

- Grafos de Control de Flujo (CFG): Representan bloques básico y saltos condiciónales



Optimización de Código

Las optimizaciones buscan mejorar velocidad, memoria y tamaño del programa sin cambiar su comportamiento:

Ejemplos de optimización comunes:

Tipo	Ejemplo
Propagación de constantes	X = 2 + 3 ~> X = 5
Eliminación de código muerto	Variables que nunca se usan desaparecen
Eliminación de redundancias	Si t1 = a + b se repite, usar el valor previo
Optimización de bucles	Mover cálculos que no cambian fuera del ciclo
Reducción de fuerza	X = y * 2 ~> x = y << 1 (mas rápido)

Ejemplos prácticos de generación y optimización de código intermedio

Ejemplo No. 1: Expresión aritmética:

Código fuente:

X = (a + b) * (a + b)

Generación de código intermedio (TAC)

T1 = a + b

T2 = a + b

T3 = t1 * t2

X = t3

Optimización:

Solo se necesita calcular a + b una vez:

T1 = a + b

T3 = t1 + t1

X = t3

Aquí se aplicó eliminación de redundancia común.

Ejemplo No.2: Eliminación de código muerto

Código fuente:

```
a = 5

b = a + 2

a = 10

print(a)
```

TAC sin optimizar:

```
T1 = 5

a = t1

t2 = a + 2

b = t2

t3 = 10

a = t3

print(a)
```

Optimizado:

```
a = 10
print(a)
```

La asignación a = 5 y b = a + 2 nunca influyen en la salida ~> código muerto eliminado

Implementación practica

El programa genera código de 3 direcciones a partir de una expresión aritmética muy simple.

****El programa lo estaré compartiendo en el espacio de la tarea****

Conclusiones

- El uso de representación intermedias permite que los compiladores sean portables entre distintas arquitecturas.
- El código intermedio facilita la aplicación de optimizaciones, permitiendo mejorar el rendimiento del programa antes de generar el código máquina.
- La optimización reduce el uso de memoria, acelera la ejecución y disminuye operaciones redundantes, logrando softwares más eficientes.

Referencias

Aho, A., Lam, M., Sethi, R., & Ullman, J. (2007). Compilers: Principles, Techniques, and Tools. Pearson Education.

Cooper, K., & Torczon, L. (2011). Engineering a Compiler. Elsevier.