1 Optimización de código intermedio

Para el proceso de optimización se utilizará el método conocido como mirilla (Peephole) y optimización por bloque.

1.1 Optimización por mirilla

El método de mirilla consiste en utilizar una ventana que se mueve a través del código de 3 direcciones, la cual se le conoce como mirilla, en donde se toman las instrucciones dentro de la mirilla y se sustituyen en una secuencia equivalente que sea de menor longitud y lo más rápido posible que el bloque original. El proceso de mirilla permite que por cada optimización realizada con este método se puedan obtener mejores beneficios.

El método de mirilla consiste en utilizar una ventana que se mueve a través del código de 3 direcciones, la cual se le conoce como mirilla, en donde se toman las instrucciones dentro de la mirilla y se sustituyen en una secuencia equivalente que sea de menor longitud y lo más rápido posible que el bloque original. El proceso de mirilla permite que por cada optimización realizada con este método se puedan obtener mejores beneficios.

Los tipos de transformación para realizar la optimización por mirilla serán los siguientes:

- Eliminación de instrucciones redundantes de carga y almacenamiento.
- Eliminación de código inalcanzable.
- Optimizaciones de Flujo de control.
- Simplificación algebraica y reducción por fuerza.

Cada uno de estos tipos de transformación utilizados para la optimización por mirilla, tendrán asociadas reglas las cuales en total son 18, estas se detallan a continuación:

9.1.1 Eliminación de instrucciones redundantes de carga y almacenamiento

9.1.1.1 Regla 1

Si existe una asignación de valor de la forma a = b y posteriormente existe una asignación de forma b = a, se eliminará la segunda asignación siempre que a no haya cambiado su valor. Se deberá tener la seguridad de que no exista el cambio de valor y no existan etiquetas entre las 2 asignaciones:

Ejemplo	Optimización
t2 = b; b = t2;	t2 = b

9.1.2 Eliminación de código inalcanzable

Consistirá en eliminar las instrucciones que nunca serán utilizadas. Por ejemplo, instrucciones que estén luego de un salto condicional, el cual direcciona el flujo de ejecución a otra parte y nunca llegue a ejecutar las instrucciones posteriores al salto condicional. Las reglas aplicables son las siguientes:

9.1.2.1 Regla 2

Si existe un salto condicional de la forma Lx y exista una etiqueta Lx:, todo código contenido entre el goto Lx y la etiqueta Lx, podrá ser eliminado siempre y cuando no exista una etiqueta en dicho código.

Ejemplo	Optimización
goto L1; <instrucciones> L1:	L1:

9.1.2.2 Regla 3

Si existe un alto condicional de la forma if <cond> goto Lv; goto Lf; inmediatamente después de sus etiquetas Lv: <instrucciones> Lf: se podrá reducir el número de saltos negando la condición, cambiando el salto condicional hacia la etiqueta falsa Lf: y eliminando el salto condicional innecesario a goto Lf y quitando la etiqueta Lv:.

Ejemplo	Optimización
if a == 10 goto L1; goto L2; L1: <instrucciones> L2:	if a != 10 goto L2; <instrucciones> L2:

9.1.2.3 Regla 4

Si se utilizan valores constantes dentro de las condiciones de la forma if <cond> goto Lv; goto Lf; y el resultado de la condición es una constante verdadera, se podrá transformar en un salto incondicional y eliminarse el salto hacia la etiqueta falsa Lf.

Ejemplo	Optimización
if 1 == 1 goto L1; goto L2;	goto L1;

9.1.2.4 Regla 5

Si se utilizan valores constantes dentro de las condiciones de la forma if <cond> goto Lv; goto Lf; y el resultado de la condición es una constante falsa, se podrá transformar en un salto incondicional y eliminarse el salto hacia la etiqueta verdadera Lv.

Ejemplo	Optimización
`if 1 == 0 goto L1;` `goto L2;`	`goto L2;`

9.1.3 Optimizaciones de flujo de control

Con frecuencia los algoritmos de generación de código intermedio producen saltos hacia saltos, saltos condicionales hacia saltos condicionales o saltos condicionales hacia saltos. Los saltos innecesarios podrán eliminarse, con las siguientes reglas:

9.1.3.1 Regla 6

Si existe un salto incondicional de la forma goto Lx donde existe la etiqueta Lx: y la primera instrucción, luego de la etiqueta, es otro salto, de la forma goto Ly; se podrá realizar la modificación al primer salto para que sea dirigido hacia la etiqueta Ly: , para omitir el salto condicional hacia Lx.

Ejemplo	Optimización

`goto L1;`
`<instrucciones>`
`L1:`
`goto L2;`
	`goto L2;`
	`<instrucciones>`
	`L1:`
	`goto L2;`

9.1.3.2 Regla 7

Si existe un salto incondicional de la forma if <cond> goto Lx; y existe la etiqueta Lx: y la primera instrucciones luego de la etiqueta es otro salto, de la forma goto Ly; se podrá realizar la modificación al primer salto para que sea dirigido hacia la etiqueta Ly: , para omitir el salto condicional hacia Lx.

Ejemplo	Optimización

`if t9 >= t10 goto L1;`
`<instrucciones>`
`L1:`
`goto L2;`
	`if t9 >= t10 goto L2;`
	`<instrucciones>`
	`L1:`
	`goto L2;`

9.1.4 Simplificación algebraica y por fuerza

Con frecuencia los algoritmos de generación de código intermedio producen saltos hacia saltos, saltos condicionales hacia saltos condicionales o saltos condicionales hacia saltos. Los saltos innecesarios podrán eliminarse, con las siguientes reglas:

9.1.4.1 Regla 8

Eliminación de las instrucciones que tenga la siguiente forma:

Ejemplo	Optimización
`x = x + 0;`	`#Se elimina la instrucción`

9.1.4.2 Regla 9

Eliminación de las instrucciones que tenga la siguiente forma:

Ejemplo	Optimización
`x = x - 0;`	`#Se elimina la instrucción`

9.1.4.3 Regla 10

Eliminación de las instrucciones que tenga la siguiente forma:

Ejemplo	Optimización
`x = x * 1;`	`#Se elimina la instrucción`

9.1.4.4 Regla 11

Eliminación de las instrucciones que tenga la siguiente forma:

Ejemplo	Optimización
`x = x / 1;`	`#Se elimina la instrucción`

Para las reglas 12, 13, 14, 15 es aplicable la eliminación de instrucciones con operaciones de variable distinta a la variable de asignación y una constante, sin modificación de la variable involucrada en la operación, por lo que la operación se descartará y se convertirá en una asignación.

9.1.4.5 Regla 12

Ejemplo	Optimización
`x = y + 0;`	`x = y;`

9.1.4.6 Regla 13

Ejemplo	Optimización
`x = y - 0;`	`x = y;`

9.1.4.7 Regla 14

Ejemplo	Optimización
`x = y * 1;`	`x = y;`

9.1.4.8 Regla 15

Ejemplo	Optimización
`x = y / 1;`	`x = y;`

Se deberá realizar la eliminación de reducción por fuerza para sustituir por operaciones de alto costo por expresiones equivalentes de menor costo.

9.1.4.9 Regla 16

Ejemplo	Optimización
`x = y * 2;`	`x = y + y;`

9.1.4.10 Regla 17

Ejemplo	Optimización
`x = y * 0;`	`x = 0;`

9.1.4.11 Regla 18

Ejemplo	Optimización
`x = 0 / y;`	`x = 0;`