Examen II – Lenguajes y Autómatas.

29 – Noviembre – 2018

Nombre: Julio César García Hernández

Matricula: 15480089

1. Es la fase cuyo objetivo consiste en modificar el código objeto generado.
2. por el generador de código, para mejorar su rendimiento.
3. Cuáles son los tipos de optimización.
4. Ejemplos de optimización dependientes de la máquina.
5. Ejemplos de optimización independientes de la máquina.
6. Explica con tus palabras como funciona el reordenamiento de código.
7. Explica un ejemplo de optimización en tiempo de compilación utilizando cuádruplas.
8. Explica un ejemplo de eliminación de redundancias.
9. Explica un ejemplo de reordenamiento de operaciones.
10. Quien es el Crush famoso de su profesor.




1.- Optimización de código

2.- Optimización de código … (esta pregunta es continuación del enunciado de la pregunta 1)

3.- Dependientes e Independientes de la máquina

4.-

- Minimización del uso de registros en máquinas en las que no se disponga de un conjunto de registros muy grande
- Uso de instrucciones especiales de la máquina
- Reordenación de código.

5.-

- Ejecución parcial del código por parte del compilador
- Eliminación de código que resulta redundante
- Cambio de orden de algunas instrucciones
- Es frecuente que los bucles sean poco eficientes, porque se ejecuten en su cuerpo instrucciones que podrían estar fuera de él, o porque la reiteración inherente al bucle multiplique la ineficiencia causada por el uso de operaciones costosas, cuando podrían utilizarse otras menos costosas y equivalentes.

6.- Este busca conseguir la generación de menos instrucciones de código, modificando el orden de las instrucciones, también minimiza el uso de registros y de variables temporales.

7.- Dado el siguiente programa en C:

```
{       int i;
        float f;
        i=2+3;
        i=4;
        f=i+2.5;        }
```

La siguiente secuencia de cuádruplas es equivalente al bloque anterior. En ellas se utiliza el operador CIF, que significa convertir entero (integer) en real (float).

(+, 2, 3, t1)

(=, t1, , i)

(=, 4, , i)

(CIF, i, , t2)

(+, t2, 2.5, t3)

(=, t3, , f)

En la primera cuádrupla (+, 2, 3, t1), se resuelve la suma y su resultado es 5, el cual se guarda en la variable temporal t1, por último, se añade a T el par (t1, 5) en q. T {(t1,5)}

En la segunda cuádrupla (=, t1, , i) se sustituye t1 por 5, quedaría (=, 5, , i), como T no contiene ningún par para i y 5 es un valor constante, se añade (i, 5) a T. {(t1, 5), (i, 5)}

En la tercera cuádrupla (=, 4, , i) se elimina de T el par (i, 5) y 4 como es constante se añade a T (i, 4). {(t1, 5), (i, 4)}

En la cuarta cuádrupla (CIF, i, , t2) se sustituye en i por 4, quedando (CIF, 4, , t2), se resuelve el CIF 4, el resultado es 4.0 (como se mencionó anteriormente el CIF convierte un int a float), posteriormente se elimina la cuádrupla y se añade a T el par (t2, 4.0) (t2 = variable temporal 2). {(t1, 5), (i, 4), (t2, 4.0)}

En la quinta cuádrupla (+, t2, 2.5, t3) se sustituye t3 por 4.0, quedando (+, 4.0, 2.5, t3), después se resuelve sin errores 4.0 + 2.5 = 6.5 y se guarda en la variable temporal t3, se elimina la cuádrupla y se añade a T el par (t3, 6.5). {(t1, 5), (i, 4), (t2, 4.0), (t3, 6.5)}

En la sexta y última cuádrupla (=, t3, , f) se sustituye t3 por 6.5 quedando (=, 6.5, , f), como no hay ningún para para f y 6.5 es constante se añade a (f, 6.5) a T. {(t3, 6.5). {(t1, 5), (i, 4), (t2, 4.0), (t3, 6.5), (f, 6.5)}

Las cuádruplas optimizadas quedarían:

(+, 2, 3, t1) ---------- Eliminada

(=, t1, , i) ---------- (=, 5, , i)

(=, 4, , i) ---------- (=, 4, , i)

(CIF, i, , t2) ---------- Eliminada

(+, t2, 2.5, t3) ---------- Eliminada

(=, t3, , f) ---------- (=, 6.5, , f)

8.- Considérese el siguiente fragmento de código escrito en el lenguaje de programación C:

```
int a,b,c,d;

a = a+b*c;

d = a+b*c;

b = a+b*c;
```

Con los algoritmos de generación de cuádruplas se obtiene la secuencia de cuádruplas

```
int a,b,c,d;

a = a+b*c;      (*, b, c, t1)
                (+, a, t1, t2)
                (=, t2, , a)
d = a+b*c;      (*, b, c, t3)
                (+, a,t3, t4)
                (=, t4, , d)
b = a+b*c;      (*, b, c, t5)
                (+, a,t5, t6)
                (=, t6, , b)}
```

La primera instrucción almacena el valor de b*c en una nueva variable temporal.

La segunda almacena en una nueva variable temporal el valor de la suma con la variable a de la variable temporal creada en la primera cuádrupla.

La tercera instrucción asigna el resultado, contenido en la variable temporal creada en la segunda cuádrupla, a la variable correspondiente, según el código fuente.

Con la observación se puede identificar las redundancias, en la primera instrucción son necesarias las 3 cuádruplas. En la segunda, ya que b y c no han cambiado de valor, pero a si, en lugar de calcular de nuevo el valor de su producto, se puede tomar directamente de la variable temporal t1. Para la tercera cuádrupla, no ha cambiado en valor de ninguna de las 3 variables, por lo que el valor de la expresión completa puede tomarse directamente de la variable temporal t4.

Las cuádruplas optimizadas serian:

(*, b, c, t1)

(+, a,t1, t2)

(=, t2, , a)

(+, a,t1, t4)

(=, t4, , d)

(=, t4, , b)

9.- Se puede seguir el siguiente orden al escribir las expresiones aritméticas:

1. Términos que no sean variables ni constantes.
2. Variables indexadas (arrays) en orden alfabético.
3. Variables sin indexar en orden alfabético.
4. Constantes.

Esto puede facilitar la localización de subexpresiones comunes y la aplicación de otras optimizaciones.

Las siguientes expresiones aritméticas:

a=1+c+d+3; es equivalente a a=c+d+1+3;

b=d+c+2; es equivalente a b=c+d+2;

Al considerar la segunda versión, se puede reducir el número de operaciones al observar que hay una subexpresión común: c+d.

10.- Emma Watson <3