APELLIDO: Dastugue NOMBRE: Matias DNI: 39.770.783 Lenguajes y Compiladores – UNLAM TERCERA EVALUACIÓN DE APRENDIZAJE	**01/12/2020** **POSICIÓN**

Evaluación de Aprendizaje Nº 3

***Ejercicio* II**

Se sabe que la fórmula de acceso que el compilador agrega al código ejecutable para acceder a un array de dos dimensiones ordenado por filas es la siguiente:

*z(i,j):=dir[v(Fi,Ci)] + [((i - Fi) * (Cn - Ci + 1)) + (j - Ci)]*tamaño del componente (tipo)*

Suponga que el compilador solo soporta límites fijos para sus vectores. Es decir, el límite inferior y superior de las filas (Fi y Fn) y de las columnas (Ci y Cn) son conocidos en tiempo de compilación.

Por ejemplo: para acceder al componente z(i,j) del vector int z(10..18,20..30)

*z (i,j) = dir [z (10,20)] + [((i–10) * (30-20+1)) + (j-20)] * 2 bytes*

También se sabe que es posible hacer una optimización por reducción simple en esta fórmula al generar polaca inversa. Indique dónde se podría hacer. Explique y ejemplifique cómo lo haría

Reducción simple (constant folding).

Se realizará la optimización partiendo de la representación intermedia del ejemplo de arriba:

z (i,j) = dir [z (10,20)] + [((i–10) * (30-20+1)) + (j-20)] * 2 bytes

La Polaca inversa sin reducción simple obtenida es la siguiente:

i	**10**	**-**	**30**	**20**	**-**	**1**	**+**	*****	**j**	**20**	**-**	**+**	**size**	*****	**addr**	**+**	**z**	**=**
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

NOTA:

- Dir [z (10,20)] se reemplaza por **addr** para simplificación.
- z(i,j) se reemplaza por **z** para simplificación.
- *tamaño del componente (tipo) se reemplaza por **size** para simplificación.*

APELLIDO: Dastugue NOMBRE: Matias DNI: 39.770.783 Lenguajes y Compiladores – UNLAM TERCERA EVALUACIÓN DE APRENDIZAJE	01/12/2020 POSICIÓN

Para hacer la reducción simple, se utilizará el siguiente algoritmo:

- Recorremos la PI (Polaca Inversa). Si es un operando, apila.
- Si es operador, desapila los anteriores y si son constantes, resuelve y apila. Entonces se sustituyen por el resultado.

Dicho esto, para nuestro ejemplo se podrá hacer reducción simple de la siguiente manera:

20 1
30 **10** **11**

Como lo mencionamos anteriormente, vamos apilando operandos hasta encontrar un operador . Desapilamos los dos operadores y al ser constantes, aplicamos la operación y volvemos a apilar. Para nuestro caso, se puede operar al apilar los operandos **20** y **30**. Cuando leemos el operando **-**, desapilamos. Al ser dos constantes, realizamos la operación y volvemos a apilar (**10**). Volvemos a repetir el proceso y apilamos el **1**. Al llegar al operando, desapilamos. Al ser dos constantes nuevamente, realizamos la operación y apilamos (**11**). A partir de de acá, ya no se pueden realizar más optimizaciones, dando como resultado, la siguiente polaca con Reducción Simple:

i	**10**	**-**	**11**	**X**	**X**	**X**	**X**	*****	**j**	**20**	**-**	**+**	**size**	*****	**addr**	**+**	**z**	**=**
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Se produce una **"baja lógica"** de las celdas marcadas. No podrán perderse celdas, pues se perderían las referencias en los saltos condicionales.

***Ejercicio* III**

Suponga un lenguaje que tiene las siguientes reglas de promoción numérica para sus tipos de datos

int → long int → double
float → double

Suponga también que en una versión del compilador se generó el siguiente conjunto de tercetos para la sentencia:

w= b*c*d+ (2.0*a) con int d; double w; long a,c; float b

APELLIDO: Dastugue NOMBRE: Matias DNI: 39.770.783 Lenguajes y Compiladores – UNLAM TERCERA EVALUACIÓN DE APRENDIZAJE	**01/12/2020** **POSICIÓN**

25 (*, b, c)
26 (*, [25], d)
27 (*, 2.0 , a)
28 (+, [26], [27])
29 (=, w, [28])

En otra versión del compilador se pide que el mismo genere los tercetos con conversiones. Escribir como quedaría el conjunto de tercetos de la sentencia anterior con conversiones de tipo según la promoción numérica establecida.

NOTA: El valor 2.0 se tomó como constante del tipo Float

25. (CFD,b,__)	Double
26. (CLD,c,__)	Double
27. (*, [25], [26])	Double
28. (CIL,d,__)	Long
29. (CLD, [28], __)	Double
30. (*, [27], [29])	Double
31. (CFD, 2.0, __)	Double
32. (CLD, a, __)	Double
33. (*, [31], [32])	Double
34. (+, [30], [33])	Double
35. (=, w, [34])	Double

CFD = Convert Float to Double
CLD = Convert Long to Double
CIL = Convert Int to Long