2.2 Representaciones de código Intermedio

En el proceso de traducir un programa fuente a código destino, un compilador puede construir una o más representaciones intermedias, las cuales pueden tener una variedad de formas. Los árboles sintácticos son una forma de representación intermedia; por lo general, se utilizan durante el análisis sintáctico y semántico.

Después del análisis sintáctico y semántico del programa fuente, muchos compiladores generan un nivel bajo explícito, o una representación intermedia similar al código máquina, que podemos considerar como un programa para una máquina abstracta. Esta representación intermedia debe tener dos propiedades importantes: debe ser fácil de producir y fácil de traducir en la máquina destino.

Existe una forma intermedia llamada código de tres direcciones, que consiste en una secuencia de instrucciones similares a ensamblador, con tres operandos por instrucción. Cada operando puede actuar como un registro. La salida del generador de código intermedio consiste en la secuencia de código de tres direcciones.

t1 = inttofloat(60)

t2 = id3 * t1

t3 = id2 + t2

id1 = t3

Hay varios puntos que vale la pena mencionar sobre las instrucciones de tres direcciones. En primer lugar, cada instrucción de asignación de tres direcciones tiene, por lo menos, un operador del lado derecho. Por ende, estas instrucciones corrigen el orden en el que se van a realizar las operaciones; la multiplicación va antes que la suma en el programa fuente. En segundo lugar, el compilador debe generar un nombre temporal para guardar el valor calculado por una instrucción de tres direcciones. En tercer lugar, algunas "instrucciones de tres direcciones" como la primera y la última en la secuencia anterior, tienen menos de tres operandos.

2.2.1 Notación Polaca

La notación polaca, también conocida como notación de prefijo o notación prefija, es una forma de notación para la lógica, la aritmética, el álgebra y la computación. Su característica distintiva es que coloca los operadores a la izquierda de sus operandos. Si los operadores es fija, el resultado es una sintaxis que carece de paréntesis u otros signos de agrupación, y todavía puede ser analizada sin ambigüedad.

La notación polaca es la originada por un Autómata con pila, en la que los operadores siempre preceden a los operandos sobre los que actúan, y que tiene la ventaja de no necesitar paréntesis:

Estándar

Ejemplo 1: 2 * (3 + 5)

Ejemplo 2: 2 * 3 + 5

Polaca

Ejemplo 1: * 2 + 3 5

Ejemplo 2: + * 2 3 5

2.2.2 Código P

*El código P comenzó como un código ensamblador objetivo estándar producido por varios compiladores Pascal en la década de 1970 y principios de la de 1980. Fue diseñado para código real para una máquina de pila hipotética la idea era hacer que los compiladores de Pascal se transportaran fácilmente requiriendo solo que se volviera a escribir el intérprete de la maquina P para una plataforma, el código P también ha probado ser útil como código intermedio y sean utilizado varias extensiones y modificaciones del mismo en diverso compiladores de código nativo, La mayor parte para lenguaje tipo Pascal.

*Como el código P fue diseñado para ser directamente ejecutable, contiene una descripción implícita de un ambiente de ejecución particular que incluye tamaños de datos, además de mucha información específica para la maquina P, que debe conocer si se desea que un programa de código P se comprensible. La máquina P está compuesta por una memoria de código, una memoria de datos no específica para variables nombre das y una pila para datos temporales, junto como cualquiera registro que sea necesario para mantener la pila y apoyar la ejecución.

COMPARACIÓN

*El código P en muchos aspectos está más código de maquina real que al código de tres direcciones. Las instrucciones en código P también requiere menos de tres direcciones: tosas las instrucciones que hemos visto son instrucciones de "una dirección" o "cero direcciones". Por otra parte, el código P es menos compacto que el código de tres direcciones en términos de números de instrucciones, y el código P no esta "auto contenido" en el sentido que las instrucciones funciones implícitas en una pila (y las localidades de pila implícitas son de hecho las direcciones "perdidas"). La ventaja respecto a la pila es que contiene los valores temporales necesarios en cada punto del código, y el compilador no necesita asignar nombre a ninguno de ellos, como el código de tres direcciones.

Ejemplo

· — Se puede considerar esta una representación intermedia como un programa para una maquina abstracta.

· — Traducción de una proposición

2.2.3 Triplos

En la historia de los compiladores han sido utilizadas una amplia variedad de representaciones intermedias como lo es la siguiente clase de representación de código intermedio de un árbol de 3 direcciones, 2 para los operandos y una para la ubicación del resultado. esta clase incluye un amplio número de representaciones diferentes entre las cuales encontramos cuadruplos y triples. la principal diferencia entre estas notaciones y la notación postfija es que ellos incluyen referencias explicitas para los resultados de los cálculos intermedios, mientras que la notación posfija los resultados son implícitos al representarlos en una pila.

§ La diferencia entre triples y cuadruplos es que con los triples es referenciado el valor intermedio hacia el número del triple que lo creo, pero en los cuádruplos requiere que ellos tengan nombres implícitos.

§ Los triples tienen una ventaja obvia de ser más consistente, pero ellos dependen de su posición, y hacen que la optimización presente cambios de código mucho más compleja.

Para evitar tener que introducir nombres temporales en la tabla de símbolos, se hace referencia a un valor temporal según la posición de la proposición que lo calcula. Las propias instrucciones representan el valor del nombre temporal. La implementación se hace mediante registros de solo tres campos (op, arg1, arg2).

En la notación de tripletes se necesita menor espacio y el compilador no necesita generar los nombres temporales. Sin embargo, en esta notación, trasladar una proposición que defina un valor temporal exige que se modifiquen todas las referencias a esa proposición. Lo cual supone un inconveniente a la hora de optimizar el código, pues a menudo es necesario cambiar proposiciones de lugar.

Una forma de solucionar esto consiste en listar las posiciones a las tripletas en lugar de listar las tripletas mismas. De esta manera, un optimizador podría mover una instrucción reordenando la lista, sin tener que mover las tripletas en si.

2.2.4 Cuádruplos

Es una estructura tipo registro con cuatros campos que se llaman:

Operador	Operando1	Operando2	Resultado

Donde operando1, operando2 y resultado pueden ser constantes, identificadores y variables temporales definidos por el compilador mientras que operador representa una operación arbitraria.

Operador	Operando1	Operando2	Resultado
*	C	D	T1
+	B	T1	T2
=	T2		A

EJEMPLO:
A := B + C * D

- <Operación>, <operando1>, <operando2>, <resultado>

Ejemplo:

(A+B)*(C+D)-E

+, A, B, T1

+, C, D, T2

*, T1, T2, T3

-, T3, E, T4

Las cuádruplas facilitan la aplicación de muchas optimizaciones, pero hay

que tener un algoritmo para la reutilización de las variables temporales

(reutilización de registros del procesador)

BIBLIOGRAFÍA

En Compiladores e Interpretes. Teoria y Practica, de M Moreno, M de la cruz, A Ortega y E Pulido, 78-85, 191-194, 199-204, 221-223. España: Pearson- Prentice Hall, 2006.

Gabriel Velasco. (2019). 2.2 Representaciones de código Intermedio. 16/04/2021, de BlogsPots Sitio web: https://gabrielvelascoalfaroautomatas2.blogspot.com/2019/09/22-representaciones-de-codigo-intermedio.html