Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

Código Intermedio

Trabajo integrador final

Practica y construccion de compiladores

Autores:

Julián González

2021

Resumen

Este documento explicara los codigos intermedios usados por los compiladores GCC y CLANG/LLVM, como asi comparar sus principales caracteristicas viendo ventajas y desventajas entre ellos.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introduccion

El codigo intermedio es un codigo interno usado por el compilador para representar el codigo fuente. El codigo intermedio esta diseñado para llevar a cabo el procesamiento del codigo fuente, como es la optimizacion y la traduccion a codigo maquina.

Una de las caracteristicas mas esenciales del codigo intermedio es ser independiente del *hardware*. Por lo tanto, permite la portabilidad entre distintos sistemas.

Otra propiedad importante de todo codigo intermedio es su facil generacion a partir del codigo fuente, como asi tambien su facil traduccion al codigo maquina para la arquitectura deseada.

No existe un unico codigo intermedio, sino que hay distintos tipos y categorias, variando de compilador en compilador. Aunque un mismo compilador puede usar varios tipos de codigo intermedio en el proceso.

A continuacion, se presentan los codigos intermedios utilizados por los compiladores GCC y CLANG/LLVM, especificando las caracteristicas de cada uno y comparando sus prestaciones posteriormente.

Capítulo 2

Codigo intermedio de GCC

A continuacion se exponen las distintos codigos intermedios que GCC utiliza en la compilacion. Los distintos codigos intermedios estan relacionados en la forma que la salida de cada uno es la entrada del siguiente, avanzando desde una representacion general de alto nivel hacia una especifica de bajo nivel.

I *Generic*

Generic es un codigo intermedio independiente del lenguaje con estructura de arbol que es generado por el *front end*. *Generic* es capaz de representar todos los lenguajes admitidos por GCC. *Generic* se produce eliminando construcciones especificas del lenguaje del arbol de parseo.

II *Gimple*

Gimple es un codigo intermedio de tres direcciones resultante de desglosar *Generic* en tuplas de no mas de tres operandos, a traves de la herramienta interna de GCC llamada *Gimplifier*. *Gimple* introduce variables temporales para poder computar expresiones complejas y permite supervisar el flujo de control a nivel inferior con sentencia secuenciales y saltos incondicionales. *Gimple* es el codigo intermedio principal de GCC (los lenguajes C y C++ se convierten a *Gimple* sin pasar por *Generic*), ademas de ser conveniente para optimizar.

Existen tres tipos de *Gimple*:

- *Gimple* de alto nivel que es lo que se obtiene despues de desglosar el *Generic*.
- *Gimple* de bajo nivel que se obtiene al linealizar todas las estructuras de flujo de control de del *Gimple* de alto nivel, incluidas las funciones anidadas, el manejo de excepciones y los bucles.
- *Gimple* SSA es el *Gimple* de bajo nivel reescrito en la forma SSA.

III *RTL*

RTL es un codigo intermedio de bajo nivel semejante al lenguaje ensamblador.

Bibliografía