Calculator Compiler Optimisation

Final Project - Compiler Design

Naveen Bharadwaj

Introduction

Each module is called from the main calc.y file

General flow of the program starts in the calc.l and calc.y files

Frontend converts input calculator language into TAC lines

TAC lines are sent to the data dependency module

Dependences are printed out to the console

Implementation

Implemented Front-end TAC generation with 3 nested if/else statements

Data dependency works concomitantly with TAC generation

Tracks all 3 dependencies

Using optimisation to reduce the number of variables in registers

Backend is used to make C from unoptimised and optimised TAC

Data Dependency Analysis

Read after write, Write after read and Write after write

Printed for every TAC line, with only one hop of data dependence

Backward from current line to find dependence

Checks for Write and Anti, if written to in current

Checks for Flow dependency, if written from in current

CSE Optimisation

The recognised form is x = a op b, ignores other forms

Recognises commutative operations

Only creates temp variable if subexpression is used more than once

Checks for future invalidation

Invalidates if subexpression was assigned to a value on any path

Heuristic Optimisation

Reduces the # of redundant operations and variables

Records the # of changes made

Fix point is reached when CSE returns zero changes

CSE is only used for inserting a temp variable

Results

0.4%

Faster for 5m loops

3.5%

Faster for bigger sets