Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Архитектура вычислительных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

«Компилятор»

БГУИР КП 1-40 04 01 009 ПЗ

Студент гр. 753504
Валетко А. Н.
Руководитель: ассистент кафедры информатики
Леченко А. В.

Минск 2019

Содержание

1. Общая информация

Компилятор – это программа, которая считывает текст программы, написанной на одном языке – исходном, и транслирует (переводит) его в эквивалентный текст на другом языке – целевом. Одна из важных ролей компилятора состоит в сообщении об ошибках в исходной программе, обнаруженных в процессе трансляции.

Если целевая программа представляет собой программу на машинном языке, она затем может быть вызвана пользователем для обработки некоторых входных данных и получения некоторых выходных данных.

В рамках данного курсового проекта был реализован компилятор для собственного языка программирования. Новый строго типизированный язык программирования, компилируемый в промежуточное представление с последующей компиляцией в машинный код.

2. Структура компилятора

Лексический анализатор

Лексический анализ – первая фаза компиляции. Лексический анализатор читает поток символов составляющих исходную программу, и группирует эти символы в значащие последовательности, называющиеся лексемами. Для каждой лексемы анализатор строит выходные токены. Они передаются последующей фазе, синтаксическому анализатору.

Синтаксический анализатор

Вторая фаза компилятора – синтаксический анализ или разбор. Анализатор использует первые компоненты токенов, полученных при лексическом анализе, для создания древовидного промежуточного представления, которое описывает грамматическю структуру потока токенов. Типичным представлением называется синтаксическое дерево.

Семантический анализатор

Семантический анализатор использует синтаксическое дерево и информацию из таблиц символов для проверки исходной программы на семантическую согласованность с определением языка. Он также собирает информацию о типах и сохраняет её в синтаксическом дереве или в таблице в таблице символов для последующей генерации промежуточного представления.

Генератор промежуточного кода

В процессе трансляции исходной программы в целевой код компилятор может создавать одно или несколько промежуточных представлений различного вида. Синтаксические деревья являются видом промежуточного представления. После синтаксического и семантического анализа исходной программы многие компиляторы генерируют явное низко уровневое или машинное промежуточное представление, которое можно рассматривать как программу для абстрактной вычислительной машины.

Машинно-независимая оптимизация кода

Фаза машинно-независимой оптимизации кода пытается улучшить промежуточный код, чтобы затем получить более качественный целевой код. Как правило это означает более быстрый или занимающий меньше памяти.

Генератор кода

Генератор кода получает в качестве входных данных промежуточное представление исходной программы и отображает его в целевой язык.

3. Синтаксис языка

Поддерживаемые типы данных

int 4 – байта

float 8 - байт

bool 1 - байт

char 2 -байта

также поддерживаются массивы данных типов

поддерживаются неявные преобразования типов к большему типу

Поддерживаемые операторы языка

Операторы ветвления if, else if, else

Логические операции || и &&

Операторы сравнения ==, != и т.д.

Константы true & false

Циклы while & do while

Операторы continue & break

Работают так же как в C/C++

4. Примеры кода и скомпилированного промежуточного представления

Пример кода программы:

```
{
        int i; int j; float[10][10] a;
        i = 0;
        while(true) {
            j = 0;
            while(true) {
                a[i][j] = 0.0;
                if( j >= 10 ) break;
            }
            if( i >= 10 ) break;
        }
        i = 0;
        while(true) {
            a[i][i] = 1.0;
            if( i >= 10 ) break;
        }
    }
```

Промежуточное представление для данного кода:

```
L1:       i = 0
L3:L5:    j = 0
L6:L8:    t1 = i * 80
          t2 = j * 8
          t3 = t1 + t2
          a [ t3 ] = 0.0
L9:       iffalse j >= 10 goto L6
L10:      goto L7
          goto L6
L7:       iffalse i >= 10 goto L3
L11:      goto L4
          goto L3
L4:       i = 0
L12:L13:t4 = i * 80
          t5 = i * 8
          t6 = t4 + t5
          a [ t6 ] = 1.0
L14:      iffalse i >= 10 goto L12
L15:      goto L2
          goto L12
L2:
```

Список использованной литературы

1. Альфред В. Ахо, Моника С. Лам, Рави Сети, Джеффри Д. Ульман Компиляторы: принципы, технологии и инструментарий (Книга Дракона-2), 2 издание. 2008 - [1178 с.]
2. Мозгов М.В. Классика программирования: Алгоритмы, языки, автоматы, компиляторы. Практический подход. 2006 – [320 с.]