Course Design on Compiler Principles

Project2 Report

Haizhong Zheng | 5110309588 | 2013 年 12 月 22 日

Part 1 Introduction

1.1 Project purpose

To Build a Small C Compiler, which can translate a Small C program into a LLVM IR program(.ll).

1.2 Major work

Build a Lexical analyzer

Build a syntax analyzer

Build a type-check program

Build a code generation tools

1.3 Tools

K-ubuntu 13.04

Bision

Flex

GCC 4.7.3

Sublime 2

Part 2 Lexical Analyzer

Use flex to handle the small C file. Using flex to generate the lex.yy.c to get token in small C, and then put the token to yacc file.

In project 2, just add "read" "write" token to lex file.

Part 3 Syntax Analyzer

3.1 Introduction

The syntax analyzer is generated by yacc(bision). Use the grammar supported. We can easily to build the syntax analyzer.

The yacc will help us to build a syntax analyzer. In the syntax-directed translation the yacc file will build a grammar tree.

3.2 Changes

In project 2, I use the stack built by yacc instead of the one built by me. Add some grammar to handle “read” and “write”. And change the grammar in define struct member. New syntax rules don’t allow struct members have initial exp..

```
    | BREAK ';'
    | READ '(' EXP ')' ';'
    | WRITE '(' EXP ')' ';'
    ;
```

```
STSPEC:
    STRUCT OPTTAG '{' STRCDEFS '}'
    | STRUCT ID
    ;
```

```
STRCDEFS:
    STRCDEF STRCDEFS
    |
    ;

STRCDEF:
    SPEC STRCDECS ';'
    ;

STRCDECS:
    VAR ',' STRCDECS
    |VAR
    ;
```

Part 4 Build block, symbol table

4.1 Introduction

Traverse the grammar tree and collect the information of the grammar tree.

Put the information of the grammar tree to the symbol table.

4.2 Data structure

A block includes struct symbol table, var symbol table.

The external block includes fun symbol table.

A struct symbol table includes struct name, the number of the member, and the information of the member.

A var symbol table includes the type of the var and the name of the var.

A fun symbol table includes the ret type of the function and the parameter's information.

```
// function symbol table
typedef struct _paraitem
{
    char *type;
    char *name;
} paraItem,*pParaItem;

typedef struct _funitem
{
    char *name;
    char *retType;
    int paraNum,cur;
    pParaItem *paraList;
} funSymTableItem, *pFunSymTableItem;

typedef struct _funtable
{
    int  funNum,cur;
    pFunSymTableItem *funList;
} funSymTable;
```

```
//struct symbol table
typedef struct _strcMem
{
    char *type;
    char *name;
    int arrNum,cur;
    int *arrList;
} strcMem, *pStrcMem;

typedef struct _strcitem
{
    char *name;
    int memNum,cur,redefine;
    pStrcMem *memList;
} strcSymTableitem, *pStrcSymTableitem;

typedef struct _strctable
{
    int strcNum,cur;
    pStrcSymTableitem *strcList;
} strcSymTable;
```

```
//variable symbol table
typedef struct _varitem
{
    char *type;
    char *name;
    int arrNum;
    int redefine;
    int isPara;
    treeNode *init;
    stmtBlock *defineBlock;
} varSymTableItem,*pVarSymTableItem;

typedef struct _variabletable
{
    int  varNum,cur;
    pVarSymTableItem *varList;
} varSymTable;
```

Part 5 Type check

5.1 type system

Top > int,struct,int arr > e

When check a type of the expression, if the type of this expression is the type needed or the sub-type of needed. This type will be ok

5.2 Other type check

Predefine

Main

Check the member of struct.

Check the para of fun

Check the type of the expression.

Check undefined variable.

Part 6 Code generation

6.1 Step 1

Generate the definition of the struct and variable.

6.2 Step 2

Generate the initial of the variable.

6.3 Step 3

Generate the code of stmt. Handle the control flow.

Part 7 testcase

It is nearly deadline.

I just put one sample: 8 queens