LAB4 -- Symbol Table with YACC

Re-submit Assignment

Due Feb 24 by 11:59pm **Points** 40 **Submitting** a file upload **File Types** pdf

This mini-project combines several concepts of compilation. In particular you will learn how to use more complex data types for the YACC stack. You will also incorporate the symbol table we worked with in the last assignment into this project.

Project:: Enhance Lab2.2 by including a symbol table into the system, changing the register names to allow any valid name, require that variable names be defined before use, and assign memory locations to each named variable.

In analyzing your lab2.2 project, you will see that the system allows for 26 registers named 'a' through 'z'. Think of these as variable names. In lab2.2, the system uses ASCII math to make 'a' - 'z' be 0 - 25, index into memory "regs" We wish to have variable names with longer names, and we wish to require our users to declare the variable names before use.

In order to do this, we need to extend the Context Free Grammar for our calculator, change your LEX program to match variable names, change our YACC directives to either store names in our symbol table, or find their address value when in use.

Your calculator will now require the following:

int V1;
int V2;...
int Vn;

expr \n | Vi=expr \n

You may want to consult **http://www.tldp.org/HOWTO/Lex-YACC-HOWTO-6.html (http://www.tldp.org/HOWTO/Lex-YACC-HOWTO-6.html)**

To accomplish this, you most likely need to do the following:

1) Extend the context free grammar of the calculator

P -> DECLS list

DECLS -> DECLS DECL | empty

DECL -> int VARIABLE ';'

2) update the lex program to match variable names instead of just single lower case characters

3) create a %UNION type in YACC to allow LEX to return integer or string

3a) define each token in the CFG to be either one of the types in the union

4) in lex, use strdup() to copy yytext to yylval.string so that YACC can have a copy of the variable name whenever a VARIABLE matches

5) insert semantic directives (C-code) to deal with when a VARIABLE is matched

5a) When you are in the declaration mode, you need to ensure that the name is not a copy, if not a copy, insert into symbol table.

5b) when a Variable is in an expression, fetch its value by getting the ADDR component out of the symbol table

5c) when a VARIABLE is on the left hand side, set the value in the memory (regs) based on the assigned slot from the symbol table

6) You will most likely need to modify symbol.Insert(char *). I created a FetchAddr(char *) to get the assigned slot for a label to help me solve this problem

Deliverables

Your name, project name

1) A copy of your LEX code.

2) A copy of your YACC code ***WITH*** the symbol table code -- suggest that you #include your symbol table code like you do for your lex code

3) demonstration that your code works

a) Show that if you define more than the MAX number of variables, then you error

b) Show that you cannot declare a variable more than once

c) Show that you cannot use a variable that is not defined

d) Show that a declared variable can be set and used (left and right hand side).