Mitchell Wade

Compilers - CS461

11/21/15

Project Assignment 4 - Intermediate Code Generation

This program involved how to translate code from a compiler in c into an intermediate form. The intermediate form would portable so that it could be transferred into any other language. The first step was jumping into the code that was given to me, and figuring out how all of the pieces worked together. Next, we started writing functions in the file sem.c that generated the intermediate code. The main files that helped while writing sem.c were the yacc file in cgram.y and also the header file that defined the id_entry and sem_rec data structures for storing information. The sem_rec data structure was the most important component because it allowed me to pass information between functions in sem.c. Another very important data structure was id_entry because it allowed me to pass an operand, an identifier, or any char * info, and it would return the proper symbol from the symbol table. This was very useful for linking arguments together that all needed to be printing out when calling a function like printf. I believe I did all that was required, and I also found a bug in Dr. Jantz code on top of completing the assignment. One of the last few statements of intermediate code that is printed out when invoking sem.c on input2 incorrectly printed out argi instead of args. This could be easily solved by just implementing an else loop in the call function that did not care what the s_mode of the given sem_rec data structure was. I decided to include this statement though: if(p->s_mode & T_STR) which would correctly print out the right output if Dr. Jantz code was correct. Please ask for more explanations if needed. I debugged and tested my solution by strictly using the given input files; however, specifically I would cat the current input file on the cmd line, run my code with the input file, and finally compare that with sem_base.exe. I compile my code using the makefile that was provided. There were multiple functions that were not necessary in order to get full credit for the code portion, and I had trouble with that. I fill like we should not use a high level language like c to generate intermediate code. It seems like a waste of resources and wouldn't speed up code generation because of how bulky all of the c code is with the parser, semantic record code, and other related files. You can untar my file using the cmd tar -xf pa4.tar.