COMP 506, Spring 2019

Project 2 Questionnaire

Name: \_\_\_\_Kexi Dang\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NetId: \_\_\_\_kd38\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Briefly discuss your experience building the ILOC code generator for Demo. What things were easy to do? What things were hard?

At first, I read the ILOC and simulator documentation to familiar with all the instructions and syntax. Because we need to generate ILOC in a new file, I defined all the operation and emit format in one file. Then I tried to implement a hash table as the symbol table to represent storage layout. These steps are preparation work for semantic parsing.

When working on bison, I process type casting first, and then insert to symbol table when encountering declaration. Then I parse scalar first with basic write and basic calculation to see whether it can generate the correct ILOC, which is a big breakthrough to me. After that, I handle the IF-ELSE, WHILE, FOR with labels and define more structures to hold all the types. At last I start to do with array which is more difficult, especially in computing the actual offset when using column-major-order to spread a multiple dimension array.

For testing, at first I could just compare the ILOC code generated by my parser and ref parser on clear when parsing a very small demo program. However, when the program getting larger, I need truly understand the ILOC that I generate and using –t flag to debug.

The easy part is implementing symbol table, parsing basic operation (+/-/\*//) and IF-THEN-ELSE. Symbol table is just the structure to hold all the variables, and we have learned how to handle the basic operations and ITE structure in class.

The hard part is processing a reference if it is an array. An array could have arbitrary dimension, so we need to figure out a formula to compute the real offset in memory. There are two difficult points, first is the time to compute the offset (I did it in reference generated grammar), the second is using ILOC sentence to compute the offset (We only have two dimension’s formula in class). Another thing needed to pay attention is the array of INT should be word aligned, which means its base address must be divided by 4.

Furthermore, when I insert mid-productions, it will generate some new conflicts. I need to refactor the grammar, for example, merging some common parts which are both used in Stmt and Withelse to solve the conflicts, which takes me a lot of time.

1. Are there topics that we should cover in class to better prepare you for this project?

Yes, I think we should know more about the type knowledge in Bison, such as the syntax and meaning of the declaration in %union and %type. And it will save students some time if we could talk some ILOC grammar and some ILOC example programs in class. Also, just for me, I need review some syntax in C language especially in pointer and struct type.

1. What, if anything, would you do differently if you could start over on this project?

One thing is that I could have a better implementation on symbol table. I will think more about efficiency and handling conflicts. The other thing is I can also add some comment in ILOC just like what the lab2\_ref does.

If I have more time, I could think about how to optimize the ILOC I generated by reducing the number of instructions and cycles. Maybe use some methods like LVN or SVN to do optimization. I think we could do this in Lab3.