Final project
CSCE 434

For your final assignment, you are to write a compiler for the ZiNC language using LR techniques. Use of yacc/bison is required and lex/flex is encouraged. Your previous labs created the lexical analysis phase, and an assembler to translate intermediate code into machine code. At this point "all" you need to do is do the syntax analysis, semantic analysis, and intermediate code generation. Your intermediate code will be assembly language for an abstract stack machine, the one for which you wrote the two-pass assembler in Lab #2.

The operations on that stack machine are sufficient order to fully implement ZiNC.

For instance, the ZiNC statement N := readInt ; would generate the assembly code:

```
LVALUE         N
READINT
STO
```

As another example, the ZiNC statement **`FLAG  :=  X  <  y  ;`** would generate the code:

```
LVALUE    FLAG
RVALUE    X
RVALUE    Y
LT
STO
```

Note that the stack machine treats false as 0 and true as non-zero; you cannot assume that "true" will always evaluate to 1.

One thing that might create some difficulty is the necessity of generating LABELs and placing them properly in your intermediate code, especially if you have nested constructs. For example:

```
if FLAG then
   if FLAG2 then
     writeInt X;
   endif;
endif;
```

needs to generate code something like this:

```
RVALUE    FLAG
GOFALSE   LABEL0001
RVALUE    FLAG2
GOFALSE   LABEL0002
RVALUE    X
PRINT
LABEL          LABEL0002
LABEL     LABEL0001
```

If you were doing this “top-down,” the solution might look like this for the production:

<if_stmt> → if_sym <expr> then_sym <stmt_seq> endif_sym

```
void if_stmt()
{
     String label;
     match(if_sym);
     expr();
     match(then_sym);
     label = genLabel();
     emit(GOFALSE, label);
     stmt_seq();
     match(endif_sym);
     emit(LABEL, label);
}


String genLabel()
{
     static int numLabel = 0;
     numLabel++;
     return "Label" + numLabel;
}
```

Embedding actions into an LR parser generator should be similar.