Computer Science 426 --- Fall 2013
ASLSOYASL Code Generation Project 6

1. Administrative Details: This project is due at 5pm on Saturday, November 30th. PLEASE NOTE WELL THAT THIS IS AT THE END OF THANKSGIVING BREAK. MY EMAIL WILL BE LIMITED ON THANKSGIVING DAY (THURSDAY THE 28TH) AND THE NEXT DAY (FRIDAY THE 29TH).

2. The Project: This project requires you to generate PAL code for ASLSOYASL (a straight line subset of yet another simple language). ASLSOYASL consists of the following constructs:

Global variable declarations.
Assignment statements.
cin with and without a parameter (i.e. cin; and cin >> x;)
swap statements
cout of string constants, expressions, and endl.

Other constructs like, loops, if statements, functions, and parameters are not part of ASLSOYASL. The following is a sample ASLSOYASL program:

```
program demo;
        int  input1, input2, sum, avg;
begin
  cout << 'Enter an integer: ';
  cin >> input1;
  cout << 'Enter a second integer: ';
  cin >> input2;
  sum = input1 + input2;
  avg  = (input1 + input2) div 2;
  cout << 'Press return to see the answer';
  cin;
  cout << 'The average is: ' <<  avg
end.
```

You are to add code generation to your expression parser and to those parts of your recursive descent parser that are involved with the ASLSOYASL subset. To do this you will associate an "action" with each reduction rule for the operator precedence parser. Each time a reduction is carried out, the action will be invoked. The action will cause one or more lines of PAL code to be generated and written to a file named out.pal which should be available to your parser class.

Variable names will be looked up in the symbol table when they are pushed onto the parse stack. Add an additional field to the parse stack which will store a pointer to the symbol table entry for that variable. You will also need to implement a routine to create a new entry in the symbol table for temporary variables. You might call the new entries: $1, $2, $3,... Since these are not legal YASL identifier names, there is no chance of a conflict between a user's identifier name and a temporary name. You can use the command sprintf to generate these names as discussed in class.

The generated code should assume **all** variables (temps and otherwise) are located on the stack. It should access these variables relative to the R0 register which should be given the value of the SP at the start of the program. Later (project 7) you will be modifying the code so that variable references can also be relative to the AP and the FP (for parameters and local variables.) **Because of this it will be VERY**

helpful later if you write a method to print an assembly language instruction. The first parameter to the method is a literal string that contains the instruction to print. The next two parameters operate as a pair to specify the first argument to the instruction. Either ptr1 will point to the argument in the symbol table and str1 will be empty (""), OR ptr1 will be NULL and str1 will contain a string literal that specifies the argument. Here are some examples:

```
printInstruction("addw", ptr1, "", NULL, "R2")
printInstruction("movw", NULL, "FP", NULL, "SP")
printInstruction("addw", NULL, "#4", ptr2, "")
```

3. Errors to Check for and Other Notes:

(a) Be sure that types are correct within an expression (for example you can't add an integer to a boolean -- for that matter you can't add a boolean to a boolean!)

(b) Be sure that the type of the right-hand side and left-hand side of an assignment statement are compatible (you must assign an integer expression to an integer variable, and a boolean expression to a boolean variable.) This means you expression parser will have to be modified to return the type of the expression that it parses (integer or boolean.)

(c) Be sure the two variables involves in a swap are of the same type.

(d) When a variable of type boolean is printed, print 0 for false and 1 for true. Do not allow boolean's to be read with a cin statement.

(e) By tradition false has the value 0 and true has the value 1. Therefore it is legal to do something such as: bool1 = false < true;
since false is indeed less then true, the variable "bool1" will be assigned true (1).

Anytime you find an error in this project you should generate an error message, print the current line, and terminate the program.

4. Getting Started:

Start with base-code that includes a complete project five along with the pre-project six that was done in class the day you heard the horn and earned the peppermint patty.

Confirm that your pre-project is working.

Look at roughly the first 10 -12 pages of notes from 11/12 up to the slide that says "Suggestions for Starting Project Six" and get this part of the project to work.

Then implement the code needed to handle:

- expression evaluation limited to int variables and addition only
- assignment statements
- cout so you can test the work above

Once this is working you can add support for all other integer operators (multiplication, division, etc.). Then add support for the swap operator and for boolean variables and everything else required by the project. Make sure all error checking (see above) is being done.