CS 536 Compilers

Program 6 – Generating MIPS code

Drew Kyser, Mike Tuchler

Our compiler assembles some important information during the name analysis tree traversal that helps us during the code generation phase. We have a global variable for the offset that serves two purposes: it tells us if a variable is local or global, and it gives each variable's Sym object a value for its offset. At the beginning and end of each function, the offset is globally reset to 0. Then, once the formals are calculated, the offset is decremented by 8 for the space for the return address and control link. Then, during the function body where local variables are found, each local will decrement the offset by an additional 4 and thus each local variable has a non-zero offset. Then, with a given ID, you can tell if it is global by checking if its offset is 0. We use the offset information to give FnSyms the space required for local variables and parameters.

We also use a global String variable to keep track of the label of the function exit block, so each return statement in a function can jump to the same proper return. This also helps us differentiate the main method return and other function returns more easily.

For IdNode, three different bits of information are required at different times. When we know an expression evaluates to an IdNode, we can cast it as such and call the proper codeGen method on it. Our three methods are:

- codeGen(), which pushes the value of the Id onto the stack
- codeGenAddr(), which pushes the address of the Id onto the stack (global or local)
- genJumpandLink(), which is used for function calls

Having a strong understanding of the stack and how it operates during compilation helped us complete this assignment.