William Junda Huang
Jonathan Lim
Raymond Wu

Design Doc

Our code generation design uses the Visitor design pattern to analyze the parse tree and create an Intermediate representation. This core of this code is in the Backend class which extends the visitor class. This class takes the generated parse tree, and converts the nodes into an IR Representation, which is subsequently converted into assembly code. To generate an IR Representation, we use the class IRBuilder which operates on BasicBlocks. The IRBuilder keeps a representation of Basic blocks as well as contains functions to convert these blocks into assembly instructions. These basic blocks are composed of a list of Instructions represented by the Instruction class, which basically provides formats for corresponding x86 instructions. Our instructions class uses various Opcodes that are enumerated by our Opcode class. Our Allocator class takes care of allocating registers appropriately. We do a series of passes to convert set instructions and restoring registers. In addition, we do an elementary version of register lifetime analysis to recycle Registers. We either allocate variables to the registers or to the stack with default preference to the register. We also consider the case where there can exist two values in an instruction that are both in memory, which is not allowed. In this case, we swap out one of the values inside the memory into the register.