Code Optimization Coursework – Group 28 Report

* Algorithm for Simple folding

simpleFolding() is the method we used to fold constants with arithmetic and logic operands. The way we did that is straight forward. In a for-loop, the program scans the list of byte code instructions and checks if there are constant values (one or two) followed with a recognized operand. Once such instance is found, it will perform operation corresponding to the operand in the byte code and replace both the previous values and operand with a new value. This for-loop will repeat until there is no new places to be optimized.

In our implementation, we also handled if-instructions which follow a constant. As the result can be decided in that case, we delete the branch which will never be visited.

* Algorithm for Constant variables

We use constantVariables() to handle constant variables. In a list scan which is executed repeatedly, if a constant-load-instruction (LDC, ICONST) followed by a store-instruction is found and no other store-instructions targeting at the same memory location, the program will replace all the load-instructions which point to the specific memory with a LDC constant instruction. After that, simpleFolding() is called for a further attempt to fold constant. The list scan procedure will finish when no new updates can be made.

* Algorithm for Dynamic variables

When the variables are assigned dynamically during the code running, the optimization becomes more complex especially in different layer of the scopes. We use a recursiveFoldVariable() method to fold variables in each layer of scopes recursively.

Similar to the way for handling constant variables, a list scan chooses one variable to fold each time. If there are load-instructions after the store-instruction, replace them with a constant. However, if there is evidences that the following instructions are in a loop or if-else block, the program will call a new recursiveFoldVariable() to check in the limited scope of instructions. The constant stored for replacement will be set to unknown at the start of the new scope. A recursiveFoldVariable() will return a location value showing where is the final constant, which can be used to replace load-instructions after the loop scope.

The dynamicVariables() method will try all the variables to fold and call the recursive function with the largest scope. Once the recursive functions are returned, a constantVariables() function will be called to fold new constant variables generated from the dynamic folding. The program will quit dynamicVariables() when there are no more changes made in the recursive function.

*Team Member:*

*Sandipan Ganguly*

*Mengyang Wu*

*Desislava Koleva*