Introduction to Compiler Construction 2016

Summary

Team: c-star-fox
Member(s): Stefan Fuchs (github.com/idefux)

As you can see, my team is a one-man show. I did the work alone for two reasons. First, I only joined the class at the end of March because I was on a business trip in Switzerland the previous four weeks. Second, it would have been very hard for me to meet with team members at regular times because I'm working full time and I have a family. Therefore, in general, I would have been only available during night hours. Due to this, my project surely lacks elegance and great design, which you can get from working together in a team and the input of others. On the other hand, as every single line of code is my doing, I gained full insight into the selfie project and I'm fully aware of every new functionality and the majority of the base functionality. Furthermore, I greatly improved my skills in gdb debugging and analyzing the assembly in order to find some bugs that hid deep in my blind spots.

In its current state my selfie can do all of the functionality asked for in the assignments. In detail:

Bitwise Shift Operators: Works.

Constant Folding: Works. Folds anything from term to compExpression. Can fold statements like *x + 4 + 5*. Maybe there are some corner cases, which my tests do not cover.

Arrays: Works. Many tests in the Makefile. Does the symbol counting. Printing the symbol occurrence table is commented out.

Struct Declarations: Works. I have chosen a rather sophisticated way by introducing a new table "user-defined-types" with references to field entries. With this, it is easily extensible for other kinds of user types. Furthermore, for me it didn't feel right to store struct fields into the SymbolTable as they are only part of the structure definition and cannot live without the struct. Placing them in the symbol table would have needed a reference to the parent struct. SymbolTable entries are references to symbols that are independent in their scope.

Struct access: Works. SymbolTable entries make use of the type *struct SymbolTable*. Cannot do nested struct accesses e.g. *s1->fieldA->fieldZ*. Can access arrays inside structs e.g. *s1->fieldA[0]* but my c* does not allow for nested struct access e.g. *s1->fieldA->fieldZ*.

Boolean operators (individual): Works. Implementation of "not" operator is handled only at compile time and does not emit any instructions in control flow. This is done by inverting comparison operators or true / false jumps. But the not operator can also be used in data flow (*x = !y*). In this case, the operator is handled at runtime by emitting MIPS instructions.

Boolean operators (algebra): Works. I implemented many tests in the Makefile and I'm pretty confident about the functionality.

Memory management: Works. Can free and reuse memory. Functionality is as discussed in the class. Uses a singly linked list that is built into the freed memory cells. I figured out that I need to use *storeVirtualMemory()* myself. Was not that hard because the code is already there in the emulators OP_SW function.

What I did beyond what you requested:
- Extensive testing in Makefiles (see the test subdir of my repo).
- I was the only green check mark in the constant folding assignment where you introduced the travis continuous integration framework.
- I worked out all the assignments alone.