Tiger to RISC V Compiler

Sourabh Aggarwal

Mentor: Dr. Piyush P. Kurur

November 2019

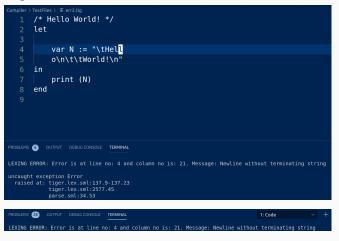
Department of Computer Science And Engineering

IIT Palakkad



What Has Been Achieved - MISC

1. More graceful error termination.



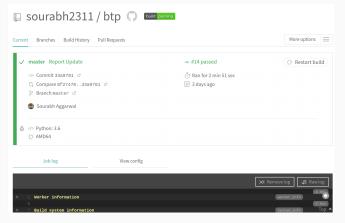
2. Wrote complete documentation of my compiler at tigercompiler.ml. This is done to help me and anyone interested in this project to quickly revise the fundamentals and understand the working of this compiler.





What Has Been Achieved

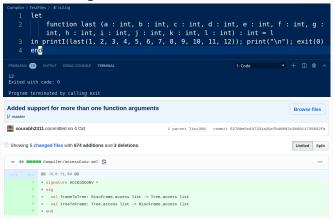
Wrote automated testing using Travis. Now I'll be able to see whether my changes don't break the existing functionalities and also it is useful in case someone sends a pull request.





What Has Been Achieved

 Fixed a major bug; Initially my compiler supported only fixed number of arguments. Now this has been extended to support any number of arguments.



5. Added 2 more arithmetic operations, viz. left shift and right shift.

```
let
         var N := 8
        var S := N << 2
         var D := S >> 2
     in
          (printI(S); print(" and "); printI(D); print("\n"); exit(0))
     end
PROBLEMS (19) OUTPUT DEBUG CONSOLE TERMINAL
                                                      1: Code
32 and 8
Exited with code: 0
Program terminated by calling exit
```



What Has Been Achieved

 Implemented multiplication by power of 2 optimization inside basic block thus laid foundation for other basic block optimizations like constant propagation, constant folding.



- 7. Started work on giving a guess of literal in case of small typo.
 - Printing suggestions which are atmost 2 distance apart. This will bring the time complexity of standard DP approach of $O(n^2)$ to just O(n).
 - Currently the issue is that to implement this, I would have to do lots of modification of the current code. Although I have abstracted out Not Found error messages out with the environment, what is just left is to compare the literal with those of nearby length in the environment.
 - The main issue is that in the current design, environment just have integers mapped to environment entry. We got this integer by mapping string to counter, not storing the reverse map. This can be worked around by using Atom.

8. Started work on improving my Register Allocator. Current version is a bit simplified version of the algorithm mentioned in the text and is without coalescing.



- Register Allocator as mentioned.
- Basic blocks has to optimizations as mentioned.
- Error messages improvement in semantic phase as mentioned.
- String comparison has to be made as simple as "str1 > str2", etc., instead of calling the string comparison functions to determine it
- To implement ability to include pre-written code (header) files.
- To implement garbage collection.
- To implement dataflow analyses such as reaching definitions and available expressions and use them to implement some of the optimizations.
- To implement first-class function values in Tiger, so that functions can be passed as arguments and returned as results.
- Add support for compile time (initial) arguments.

Thanks!