# 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 Acheived - MISC

1. More graceful error termination.

Compiler > TestFiles > err2.tig

```
1  /* Hello World! */
2  let
3
4      var N := "\tHell
5      o\n\t\tWorld!\n"
6  in
7      print (N)
8  end
9
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL

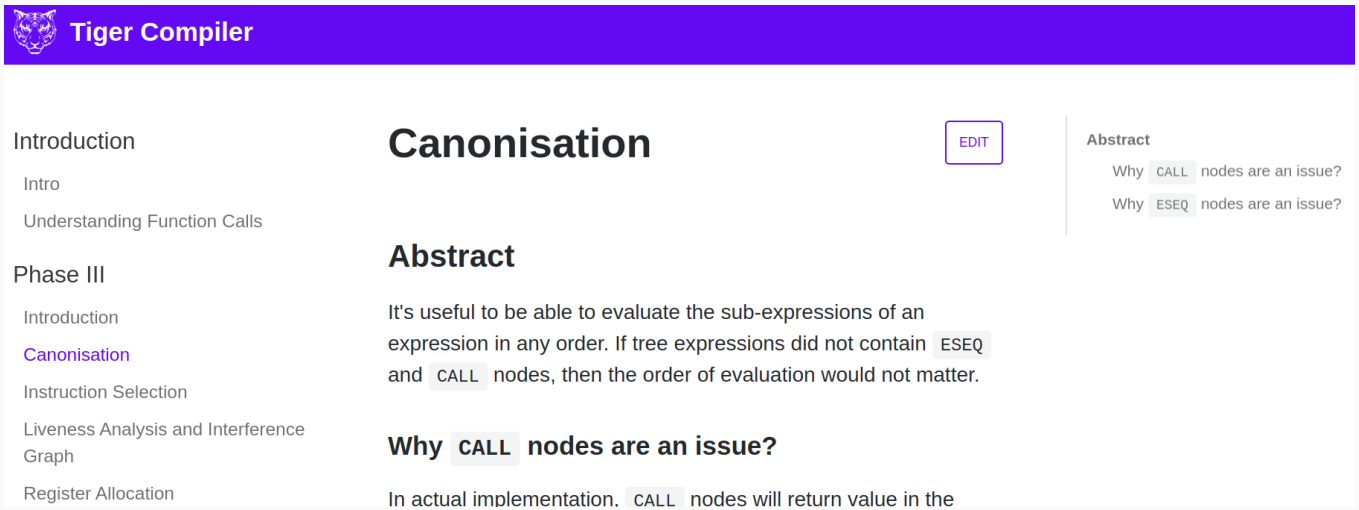
```
LEXING ERROR: Error is at line no: 4 and column no is: 21. Message: Newline without terminating string

uncaught exception Error
  raised at: tiger.lex.sml:137.9-137.23
             tiger.lex.sml:2577.45
             parse.sml:34.53
```

PROBLEMS 28 OUTPUT DEBUG CONSOLE TERMINAL 1: Code

```
LEXING ERROR: Error is at line no: 4 and column no is: 21. Message: Newline without terminating string
```

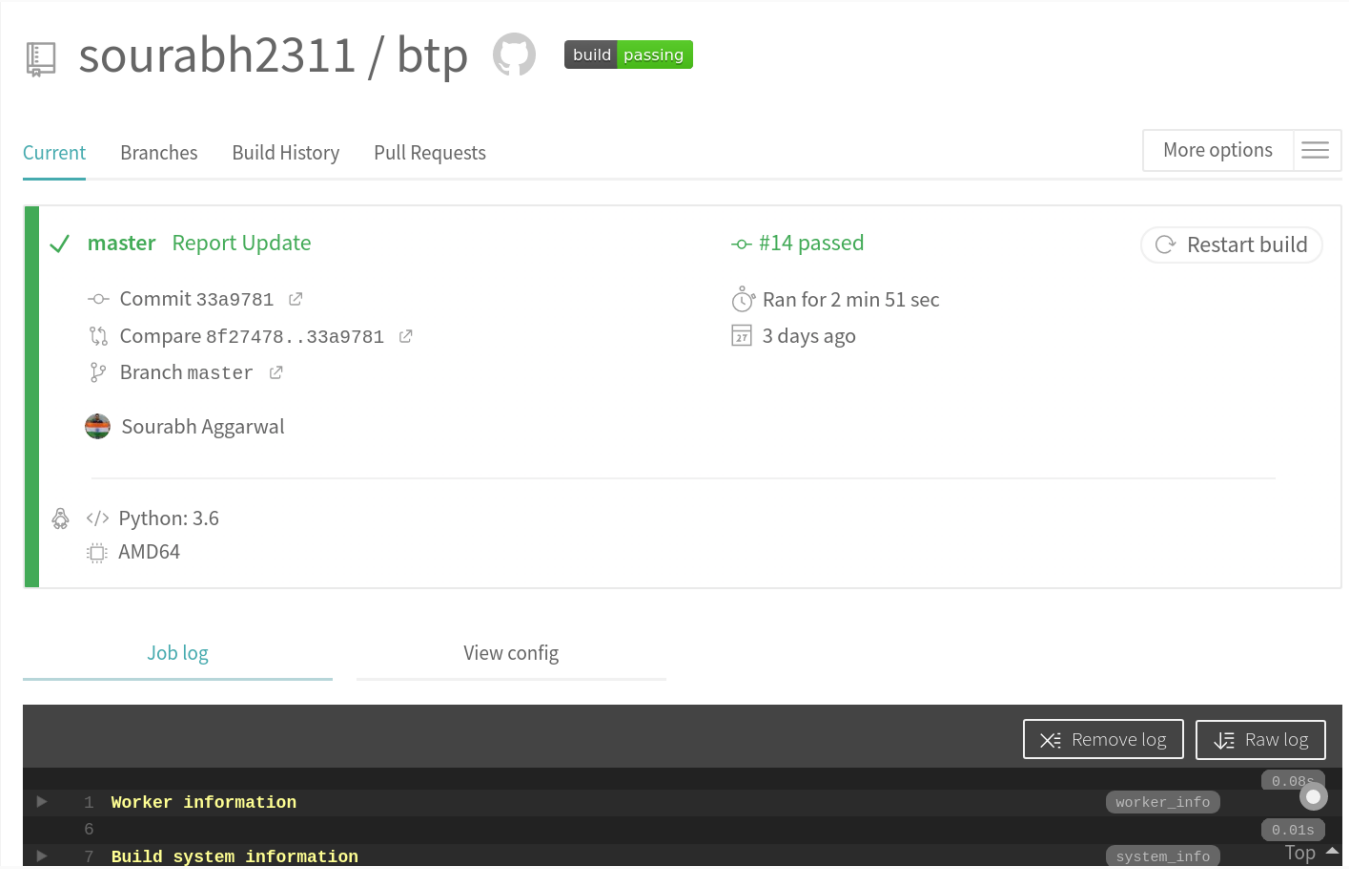
# What Has Been Acheived

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 Acheived

3. 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 Acheived

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

Compiler > TestFiles > tc2.tig

```
1  let
2      function last (a : int, b : int, c : int, d : int, e : int, f : int, g :
       int, h : int, i : int, j : int, k : int, l : int) : int = l
3  in printI(last(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)); print("\n"); exit(0)
4  end
```

PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL 1: Code

```
12
Exited with code: 0

Program terminated by calling exit
```

**Added support for more than one function arguments**

Browse files

master

**sourabh2311** committed on 4 Oct  1 parent 74cc398  commit 5270be5c637331a29cfb40092c3b5011735652fa

Showing **5 changed files** with **674 additions** and **3 deletions**.

Unified Split

39 Compiler/accessConv.sml

```
@@ -0,0 +1,39 @@
+ signature ACCESSCONV =
+ sig
+   val frameToTree: RiscFrame.access list -> Tree.access list
+   val treeToFrame: Tree.access list -> RiscFrame.access list
+ end
```

# What Has Been Acheived

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

Compiler > TestFiles > tc9.tig

```
1  let
2      var N := 8
3      var S := N << 2
4      var D := S >> 2
5  in
6      (printI(S); print(" and "); printI(D); print("\n"); exit(0))
7  end
8
```

PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL 1: Code

```
32 and 8
Exited with code: 0

Program terminated by calling exit
```

# What Has Been Acheived

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

Compiler > TestFiles > tc10.tig

```
1  let
2      var N := 8
3      var M := 4 * N
4      var O := N * 4
5  in
6      (printI(M); print("\n"); printI(O); print("\n"); exit(0))
7  end
8
```

PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL 1: Code

```
32
32
Exited with code: 0
```

Compiler > TestFiles > ASM tc10.tig.s

```
28      mv s10, s11
29      li s7, 8
30      slli s1, s7, 2
31      mv s11, s1
32      slli s1, s7, 2
```

# What Has Been Acheived

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.

# What Has Been Acheived

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.

# Road Ahead

- 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!