CS 3432 Spring 2023

Shirley Moore, Instructor

Lab 1a 50 points

**Expression Compiler Backend for RISC-V: Part a**

A compiler typically has frontend and backend components. The frontend parses a source code program and produces an abstract syntax tree (AST) representation of the program. The backend traverses the AST and outputs assembly or machine code instructions for a specific ISA. Other actions may also be carried out, such as transformations of the AST to implement optimizations such as strength reduction and common subexpression elimination.

For this assignment, you are given a build\_tree() function that takes a C r-value expression as an argument and returns a pointer to an AST. That is, you are given a frontend for a mini-compiler that handles a subset of C expressions. An r-value is an expression that can appear on the right-hand side of an assignment statement. Your job is to write a backend that traverses the AST and outputs RISC-V assembly code.

Please carry out the following tasks:

1. (10 pts). You are given files main1a.c, build\_tree1a.c, build\_tree.h, backend1a.c. To compile these files into an executable named compile using the gcc compiler, type

gcc -o compile main1a.c build\_tree1a.c backend1a.c -I.

Construct a Makefile that will manage building the code.

To run the code, type

./compile

and then enter an expression and press return. To stop entering expressions and quit the program,

type Ctrl-D for end-of-file. You can also redirect input from a file that contains expression strings one

per line, e.g.,

./compile <input1a.txt

1. (10 pts) Write a function that takes an unsigned integer as an argument and determines if that integer is a power of 2. If the integer is a power of 2, the function should return the logarithm to the base 2 of the integer, otherwise the function should return 0. Write a driver program to test your function and test your function thoroughly to make sure it is correct.
2. (10 pts) Examine the code in build\_tree1a.c and explain how it works. Note how the postorder() function traverses the AST and outputs a postfix expression. Note that currently the code allows only single-letter variables using the first 10 lower-case letters. You may remove this restriction if you wish, but you are not required to do so.
3. (10 pts) Write a backend that traverses the AST and outputs a RISC-V translation of the expression. You may use register x0 as needed and you may use registers x5 through x31 to hold variable values and intermediate results. For this part a, you may assume that expressions use only variables and do not use constants. Note that you will need to convert the unary negation and NOT operators to binary operators since RISC-V does not have any instructions for unary operations. You are given a starter code in the file backend1a.c. You are also given a sample input file input1a.txt and the expected output in file output1a.txt. The starter code will handle all the binary operations and the unary minus operation. Your task for part a is to add code to handle the unary not operation. You should also add more test cases to thoroughly test your code.
4. (10 pts) Describe any difficulties or obstacles you ran into in doing this lab and if/how you overcame them. Summarize what you learned from doing this lab.