CS 3432 – Computer Organization Authors: Dr. Shirley Moore, Robert Alvarez
Lab 3 - Expression Compiler Backend for RISC-V

Instructions:
The assignment would be an INDIVIDUAL assignment. It is permitted to share ideas with other classmates BUT you must write your code. Meaning, code implementation must be unique.

Introduction:
A compiler typically has frontend and backend components. The front end 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 front end 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.

Task:
1. [12 pts]. You are given files main.c, build_tree.c, build_tree.h, backend.c. The file main.c contain the main() function. Construct a Makefile that:
   a. Creates an executable when "make" is run.
   b. Run the executable when target "run"

   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
   ```

2. [17 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.
   a. Create a new file to test such a function.
      - The function can be implemented in one line using the trinary operator and the bitwise AND operator.
      - No need to implement log2() function, you can import it from math.h standard library.
   b. Modify the Makefile so when "make testPowTwo" the tester file is compiled and executed.
   c. The tester should test your function with at least 5 powers of two numbers and 5 that are not.
      - Instead of hardcoding values, you can use the rand() function to generate random integers.

3. [17 pts] Compile and run the code in build_tree.c. The postorder() function traverses the AST and outputs a postfix expression. Currently, the code allows only single-letter variables using the first 10 lower-case letters. Modify the code so it can read variables of up to 5 characters long where each character may be a lower-case, upper-case, or an underscore. After examination and code modification, explain the algorithm on how the functions are implemented.

4. [22 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. You should implement strength reduction by transforming multiplication by a power of two to use a shift instruction instead of a multiplication instruction (the function you wrote for part 2 above will be useful here).
    - 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.

5. [22 pts] Add code to the generate_code() function to handle expressions that contain constants. For binary operations, the code you are given handles the case where both the left and right children of the operator node are REG-type nodes. You will need to handle the following additional cases:
    - The left child is REG type, and the right child is CONST type.
    - The left child is CONST type, and the right child is REG type.
    - The left child is CONST type, and the right child is CONST type.

    For unary operations, the code you wrote for part a should handle the case where the operand of a unary operator is a REG-type node. You will need to add code that handles the case where the operand is a CONST type node.

6. [20 pts] Write a new input file (.txt file) that tests all the possible cases implemented in the previous 3 points. In a Word document justify why each line is testing a case requested above. Justification consists of:
    a) Parsing the expression and creating the AST.
    b) Traverse the AST and, as you do it, assign/release registers as needed.
    c) Mention at what point of the AST traversal an assembly instruction is generated and what branch in the backend.c is being executed.
    d) Also, provide the output file (.txt file) of such input file.

7. [10 points] Write code to handle immediate values that are larger than 12 bits.

Grading:

- The total maximum number of points is 120. Lab would be grade out of 100.

Deadline (Blackboard): April 9th, 2024, by 11:59 pm.

- Source code (Only .c and .h files, and 1 Makefile)