Report Title

DUDEKULA DHEERAJ
ROLL NO: CS22BTECH11019

October 12, 2023

1 Introduction

In this report, we will discuss the implementation of a C program that takes RISC-V assembly instructions as input, converts them to binary, and decodes and prints the corresponding assembly language instructions. The program is organized into functions for different instruction types, and it uses global variables to store various components of the instructions.

2 Code Overview

The code is structured as follows:

- **Header Files and Global Variables**: The code includes standard C header files and declares global variables for instruction components and label arrays.
- **Decimal Function**: The ‘decimal‘ function is responsible for converting binary strings to decimal. It handles different binary string lengths and two’s complement representation for negative numbers.
- **Convert Function**: The ‘convert‘ function extracts various fields from a 32-bit binary instruction and converts binary representations to decimal.
- **Instruction Type Functions**: There are functions for R-type, I-type, S-type, B-type, J-type, and U-type instructions. These functions identify and print the corresponding assembly instructions.
- **Main Function**: The ‘main‘ function reads 20 hexadecimal instruction strings, converts them to binary, and processes each instruction. It calls the appropriate instruction type function based on the opcode.
- **Conversion from Hex to Binary**: Hexadecimal input strings are converted to binary using a series of ‘if‘ statements that map each hex digit to a 4-bit binary sequence.
- **Labeling Branches and Jumps**: The code keeps track of branch and jump addresses in arrays and labels these instructions correctly.
- **Output**: The program prints the corresponding assembly instructions to the standard output.
- **Code Organization and Readability**: The code is organized into functions with descriptive names, making it modular and readable.

3 Conclusion

In conclusion, this C program successfully converts hexadecimal RISC-V assembly instructions to binary and decodes them into assembly language instructions. The code structure is well-organized and allows for easy extensibility and maintenance.