Report: Disassembling RISC-V Machine Code

B Rutwik Chandra (CS22BTECH11011)

October 12, 2023

1 Introduction

This code is designed to convert RISC-V machine code to its equivalent assembly syntax. It supports various instruction formats (R, I, S, B, U, J) as specified in the assignment.

2 Approach

This code uses C and follows a structured approach. It consists of several functions, each responsible for decoding a specific instruction format.

2.1 Hexadecimal to Binary Conversion

The `hexCharToBinary` and `hexToBinary` functions convert hexadecimal characters and strings to 32-bit binary strings.

2.2 Binary to Decimal Conversion

The functions `binaryToDecimal` and `TwosCompToDecimal` convert binary strings and 2's complement binary strings to decimal values respectively.

2.3 Instruction Decoding Functions

R_Type: Handles R-type instructions (add, sub, and, or, xor, sll, srl, sra).

I_Type: Handles I-type instructions (addi, andi, ori, xori, slli, srli, srai, ld, lw, lh, lb, lwu, lhu, lbu, jalr).

S_Type: Handles S-type instructions (sd, sw, sh, sb).

B_Type: Handles B-type instructions (beq, bne, blt, bge, bltu, bgeu).

U_Type: Handles U-type instructions (lui).

J_Type: Handles J-type instructions (jal).

2.4 Label Handling

The program supports adding labels to B and J instructions in Part 4. It keeps track of labels, associates them with the corresponding instruction, and generates appropriate output. This also handles the cases where the offset for the labels is negative.

2.5 File Reading and Output

The program reads the input from the `input1.txt` file and processes each instruction. The input file must contain a maximum of 50 addresses. This can also be changed in the code to any other value. It also handles output formatting, including labels.

3 Testing and Verification

The correctness of the code has been verified using the Ripes simulator. The process involved the following steps:

1. Entering instructions in assembly language in Ripes.
2. Obtaining the equivalent hexadecimal machine code.
3. Providing the machine code as input to the program.
4. Verifying that the program outputs the correct assembly syntax.

4 Assumptions

This code assumes that the correct hexadecimal values are given as input. i.e. the code gives the correct output only when the input hexadecimal addresses form a meaningful assembly instruction. This code doesn't display any error message if an incorrect hexadecimal address is entered.

5 Conclusion

The code successfully disassembles RISC-V machine code into its equivalent assembly syntax. It supports various instruction formats and handles labels in B and J instructions. The correctness of the code has been verified using the Ripes simulator.