CS2323 - Computer Architecture, Autumn 2023

LAB-3 Report
RISC-V Disassembler

Name: Mayank Parasramka

Roll.no: AI22BTECH11018

Introduction

Lab-3 aims to develop a RISC-V disassembler in C, capable of converting RISC-V machine code into its corresponding assembly syntax. The disassembler is designed to support various RISC-V instruction formats, including R, I, S, B, J, and U types. The disassembler's output includes labelled branch and jump instructions as an extension for Part-4.

Implementation Approach

The entire code is written using several functions, breaking down the code to support various instruction formats.

Input:

The input is assumed to be directly available in an array within the code. To add our own inputs, we can add more “Hex-digits” (Disassembled code- 32 bits) and also change the line count to the total number of elements added to the string array.

Output:

The output is in the form:

- Each line has one risk-v code in assembly syntax for the corresponding disassembled hex digits.
- The labels for B/J-type instructions are put in their corresponding lines as per the PC (Program Counter).

Functions:

1. hexToBinary: Converts a hexadecimal string (machine code) into its binary equivalent.
2. binaryToDecimal: Converts a binary string representing an unsigned integer to its decimal equivalent.
3. binaryToDecimalsigned: Converts a binary string representing a signed integer in two's complement form to its decimal equivalent
4. findOpcode: Extracts the opcode part (last [0:6] bits) of a binary RISC-V instruction.
5. findfunct3: Extracts the funct3 field of a binary RISC-V instruction.
6. findfunct7: Extracts the funct7 field of a binary RISC-V instruction.
7. findrs1, findrs2, findrd: Extracts the source (rs1, rs2) and destination (rd) register fields of a binary RISC-V instruction.
8. R_format: Disassembles R-format instructions (e.g., add, sub, and, or, etc.).
9. I_format_1: Disassembles I-format instructions with immediate values (e.g., addi, andi, ori, etc.).
10. I_format_2: Disassembles I-format instructions with loads (e.g., lb, lw, lh, etc.).
11. S_format: Disassembles S-format instructions (e.g., sb, sh, sw, etc.).
12. B_format: Disassembles B-format instructions (e.g., beq, bne, blt, etc.) and handles labels for branch instructions.
13. J_format: Disassembles J-format instructions (e.g., jal) and handles labels for jump instructions.
14. U_format: Disassembles U-format instructions (e.g., lui, auipc).
15. Main Function: It is used to call the function corresponding to each instruction type by verifying the Opcode of the binary. The labels of the B/J-type instruction type are also added in the output through the main function.

Testing:

The code is tested for many different input cases. Please find it in the zip file.

Disclaimer: Do change the line count for the corresponding number of strings.