Name: DOKKU HEMANADH
Roll no.: CS22BTECH11018

RISC-V Instruction Disassembler Report

1 Introduction

The provided RISC-V disassembler is implemented in C++ and translates machine code instructions into corresponding assembly instructions. The disassembler employs a modular approach, extracting essential fields from instructions through bitwise operations and accurately computing immediate values. The disassembled instructions are stored in vectors for further processing and analysis.

2 Functionality Overview

The disassembler includes functions for:

- Converting hexadecimal strings to integers.
- Extracting opcode, registers, and function codes using bitwise manipulation.
- Calculating immediate values for different instruction formats (I, J, S, B, U).
- Storing disassembled instructions in vectors for further analysis.

The main function reads machine code from a file, decodes it, handles branches and jumps, and resolves labels dynamically for precise disassembly.

3 Instruction Classification and Explanation

3.1 Control Transfer Instructions

3.1.1 Conditional Branches (`beq, bne, blt, bge, bltu, bgeu`)

These instructions alter the control flow based on conditions. The disassembler computes offsets and adjusts program flow accordingly.

3.1.2 Unconditional Jumps (`jal, jalr`)

`jal` performs an unconditional jump and saves the return address. `jalr` jumps to an address computed from a base register and an immediate offset, also saving the return address.

3.2 Immediate Instructions

3.2.1 `lui`

`lui` loads an immediate value into the upper bits of a register.

3.2.2 `auipc`

`auipc` adds an immediate value to the current PC and stores the result in a register, forming the base address for relative addressing.

3.3 Arithmetic and Logic Instructions

3.3.1 ALU Operations (`add, sub, xor, or, and, sll, srl, sra, slt, sltu`)

These instructions perform various arithmetic and logic operations.

3.3.2 Immediate ALU Operations (`addi, xori, ori, andi, slti, sltiu, slli, srli, srai`)

Similar to previous operations but with immediate values.

3.4 Load and Store Instructions

3.4.1 Load (`lb, lh, lw, ld, lbu, lhu, lwu`) and Store (`sb, sh, sw, sd`) Instructions

These instructions load and store values from/to memory.

4 Labeling Logic

The disassembler handles branch and jump instructions involving labels by dynamically resolving targets. It modifies instructions to reference existing labels or creates new ones when necessary.

5 Testing and Verification

The disassembler has undergone rigorous testing with diverse RISC-V machine code inputs and edge cases. Cross-verification with the Ripes simulator ensures the accuracy and correctness of the disassembled instructions.