Lab 8 Report

Nelakuditi Rahul Naga - AI20BTECH11029

Coding Approach

I have implemented the RISC-V Disassembler using Python. To do this, I have first written some utility functions. Their descriptions are as follows :

1. **binary_to_decimal :** This function takes a string input in binary format and outputs an decimal number. This function also outputs a negative integer if the input is an immediate value and it's signed bit is 1.
2. **binary_to_hexadecimal :** This function takes a string input in binary format and outputs an equivalent hexadecimal string.
3. **hexadecimal_to_binary :** This function takes a string input in hexadecimal format and outputs an equivalent binary string.

The RISC-V Disassembler is implemented using the functions **instruction_array_decoder** and **instruction_array_decoder_with_text_support**. The functions take input in the format of a list and output the equivalent assembly instructions. Both the functions support R,I,S,B,J and U-format instructions. But as the function names clearly convey, the 1^{st} function just mentions the offset in the third operand rather than the textual label for B/J type instructions. But the 2^{nd} function also supports textual labels for the B/J type instructions. Both the functions make use of the utility functions mentioned above. The utility function **binary_to_hexadecimal** is particularly useful in the **lui** instruction (U-format) while mentioning the immediate value in hexadecimal format.

Testing Correctness

To check the correctness of my code, I have first utilized the example given in Lab 8 problem statement and verified that my functions give the correct answer. I have also written some simple assembly programs, converted them to machine code using Ripes simulator and then fed them to my functions to check the correctness of my code.