Computer Architecture - CS2323. Autumn 2023

Lab-2 (Detecting RISC-V Instruction Type)

We discussed that each RISC-V instruction is a 32-bit word with various bits helping to encode various aspects. Among these, there are few bits which define the opcode for the corresponding instruction. As a part of this lab assignment, you are expected to process the opcode field of the instructions and classify them into R/I/B/S/J/U type of instruction. Use the first page of the RISC-V reference card that was shared on moodle to understand which instruction falls under which category. Any instruction given in that first page could be used to evaluate your code.

Input: The input instruction will be provided to you through the last 32-bits of the register x4. You need to do appropriate processing and store the type of instruction in the register x10. x10 should contain a decimal value as per the following encoding:

R-type: 1
I-type: 2
B-type: 3
S-type: 4
J-type: 5
U-type: 6

As an example, if we give 0x00000000FCD18613 in register x4, then the value in register x10 should be 2 (indicating I-type instruction) after executing your code.

The following code template can be helpful.

```
.data
#if needed, else ignore the data section

.text
   #your code starts here


  #    WRITE YOUR CODE HERE


   #The final result should be in register x10
```

Instructions:

1. Use Ripes simulator from: https://github.com/mortbopet/Ripes/releases/download/v2.2.4/Ripes-v2.2.4-linux-x86_64.AppImage - the 2.2.4 version is more stable and reliable than later ones.

2. Configure simulator for 64-bit processor (click on the processor button below File in the top-left and select 64-bit single cycle processor).
3. While doing this exercise, try to use breakpoints, single stepping, etc. features of the simulator for a better understanding. We will need these features when debugging the programs in subsequent assignments. Also, see the corresponding disassembled (translated) code in the right pane.
4. Validate your code with various types of instructions, using different register operands, etc.

Submission instructions:

1. Submit the assembly code as a file named YOUR_ROLLNUM.s (e.g., CSYYBTECHXXXXX.s)
2. The assignment should be done individually
3. Copying from others or any other source is strictly prohibited and subject to strict penalty
4. Assignments will be tested for similarity among each other and any violation will be reported appropriately
5. Submission deadline: 26 September 2023, 11:59 PM