Computer Architecture - CS2323. Autumn 2023

Lab-3 (RISC-V Disassembler)

As we just finished discussing instruction encoding and decoding in the current lectures, this assignment will help you assimilate those concepts through practical implementation. You need to write C/C++ code to convert a given RISC-V machine code to its equivalent assembly syntax. The input machine code could be provided as a separate file or assumed to be directly available in an array within the code.

Example: If the input is as follows (consider each term below is a hex-digit):

```
007201b3
00720863
00c0006f
00533623
100004b7
00c50493
```

Output should be:

```
add x3, x4, x7
beq x4, x7, L1
jal x0, L1
sd x5, 12(x6)
lui x9, 0x10000
L1: addi x9, x10, 12
```

We provide staged problem statements and you may complete all or only a few parts (based on your available time and interest).

Part-1 (50% weightage): Support only R-format instructions to their equivalent assembly code. The following instructions must be supported:
add, sub, and, or, xor, sll, srl, sra

Part-2 (20% weightage): Part-1 + I-format + S-format instructions to be supported. The following instructions must be supported:
add, sub, and, or, xor, sll, srl, sra, addi, andi, ori, xori, slli, srli, srai, ld, lw, lh, lb, lwu, lhu, lbu, sd, sw, sh, sb, jalr

Part-3 (20% weightage): Part-2 + B-format + J-format + U-format: The following instructions must be supported:
add, sub, and, or, xor, sll, srl, sra, addi, andi, ori, xori, slli, srli, srai, ld, lw, lh, lb, lwu, lhu, lbu, sd, sw, sh, sb, beq, bne, blt, bge, bltu, bgeu, jal, lui

For B/J instructions, when you generate the assembly code, you could just mention the offset in the third operand rather than the label.
e.g., it is sufficient to write the disassembled instruction as beq x4, x7, 16 rather than actually putting a label in the assembly code.

Part-4 (10% weightage): Part-3 + support adding a textual label for the B/J instructions.
E.g., Rather than writing beq x4, x7, 16; your output should be beq x4, x7, L1 and L1 should be placed at an appropriate place in the disassembled code.

Submission instructions:

The assignment is to be done individually. Submit your code and instructions to compile/execute your code should be put in a separate README file. Prepare a short report on your coding approach and what all you did for testing your code to be correct. Submit a zip file containing the following:

1. Source files of code
2. README: A text file containing instructions to compile/execute your code
3. Testcases, if you tried out specific input files
4. report.pdf - a short report on your coding approach and what all you did for testing your code to be correct. If a proper report is not submitted, you will receive 10% lesser marks.

Submit the zip file with name CSYYBTECHZZZZZ.zip

Deadline: 12-Oct-2023 (Thursday), 11.59 pm.
Demo: We may later conduct a demo where you will show the functioning of your code to the TAs and accordingly be assigned marks.

Note: Submissions are subject to similarity check and hence submit only your own work. Any similarity will be strictly dealt with.
Your submissions will be evaluated on Ubuntu 20.04 machines and hence you must check them to be working on such a setup.

Some implementation specific aspects:

1. If it helps your implementation, you may consider that the offset in part-4 is positive.

2. You may also consider that the input test program given during your demo would have a maximum of 20 lines within it.