CS207 - FCS Lab
Course Project Details
Date: 9th Oct 2025. Deadline: 12th Nov 2025 11:59PM. Weightage: 25%

This is a group task. Our goal is to build a RISC-V assembler (like Venus assembler part). Form groups of three (no more/no less). Pick one of your group members as Single Point of Contact (SPOC in short). SPOC will create a git-repo for the team. All the code update and sync-up need to be done using your repo. We will be monitoring this activity from time to time.

Conversion of Assembly code to Machine code

Here, you are doing the job of a 64bit RISC-V assembler. You will write a C/C++/Java/Python program that reads a RISC-V assembly code as input in a `input.asm` file and generates the machine code as output in a `output.mc` file.

i) `.asm` file will have a format like below (one assembly instruction in each line)

```
add x1, x2, x3
andi x5, x6, 10
...
```

ii) `.mc` file is expected to have a format like below (<address of instruction><delimiter - space><machine code of the instruction><delimiter - space - comma><assembly instruction><delimi space - #><opcode-func3-func7-rd-rs1-rs2-immediate>one in each line):
Similar to Venus let us assume that code segment starts at 0x0000 0000 and data segment starts at 0x1000 0000,heap at 0x1000 8000, and stack segment at 0x7FFF FFDC.

So, your code segment of `.mc` file would look like:

```
0x0 0x003100B3 , add x1,x2,x3 # 0110011-000-0000000-00001-00010-00011-NULL
0x4 0x00A37293 , andi x5,x6,10 # 0010011-111-NULL-00101-00110-000000001010

0x8 <your termination code to signify end of text segment and in turn, end of
the assembly program>
Your data segment in the same .mc would look like:
0x10000000 0x10
....
```

Your assembler code needs to parse the `.asm` file one or more times to precisely calculate and replace various labels appropriately.
Regarding the instructions that you need to support:
1. Support for Pseudo instructions is not compulsory.
2. Limit to RISC-V 64bit ISA, specifically, below 37 instructions:

- R format - `add, addw, and, or, sll, slt, sra, srl, sub, subw, xor, mul, mulw, div, divw, rem, remw`
- I format - `addi, addiw, andi, ori, lb, ld, lh, lw, jalr`

- S format - `sb, sw, sh, sd`
- SB format - `beq, bne, bge, blt`
- U format - `auipc, lui`
- UJ format - `jal`
- Also, you can skip support for floating point operations.

In addition, you need to provide support for the assembler directives: `.text, .data, .byte, .half, .word, .dword, .asciz`.

Late submissions will attract penalty of 30% per day. Revert if you have any questions.