A sample project to represent C/C++ skillset

Overview:

Write an application that will read each line of the input file and generates a Data structure to represent the operator, operands and other necessary attributes of a RISC v instruction-set.

You need to lexically analyze, if the instruction in each line is valid or not and if valid push them in your final list of Instructions and include other attributes of the instruction.

Grammar you need to implement, for the time being

- G0: instruction format in each line,
  `<op code> <space> <list of operands separated by comma> <list of operands ends with space, newline, or hash sign>`
- G1: You have a static list of instructions op code.
- G2: Each instruction has a fixed number of operands.
  `add, addi, sub, subi` have 3
  `la, li` can have 2
  `ecall` none
- G3: Bonus, operand can be of type register or a number.

For example,

- `add a2, a1, a0` is a valid instruction;
- whereas `add a0, a1` is not a valid since it has two operands instead of three, ref G2
- And `add a3 a0 a1 a2` is not valid because of G0
- and `dummy a0, a1, a2` is not valid, since the operator itself is not a valid instruction, ref: G1

Ignore the line that starts with `#` sign. (Bonus: ignore rest of the line when you find a hash sign in the middle of the line)

Please let me clarify this at the very beginning, solving the above problem is not the highest priority; we're mostly interested to learn about your exposure to C++ domain.

[A sample RISC v assembly blog for beginners]

Sample Input:

#all instructions are valid in following

```
add   a0, x0, x1
la    a1, hello
li    a3, 5
addi  a2, x0, 5
subi  a7, x0, 64
ecall
```

Features we are looking for in your application:

- C++ version needs to be 17+
- Makefile based project structure
- Hefty usage of OOP concepts, design patterns etc.
- Code optimization using features of latest C++ features
    - For example, Templating, ConstExpr, Casting, Structure.
    - Some tips are available here, https://abseil.io/tips/
- If you have questions, please understand these are kept deliberately open. Use your best assumption and we like to find how flexible your code is.