Here is a summary of my way of designing an assembler:

At the beginning I read the inputs from a text file. In the following I split them line by line and also delete the ",".

In the next I define Some functions. One of them that helps us to find which instruction (ADD, AND, SUB and OR) is called. And pass us to it's exact function for the remaining calculation. And the others that help us to get the MOD and Direction Bit.

Then I define the Instructions. And in each instruction I check for the registers size (8, 16 or 32 bits) of the given operands. The calculation of each instructions is based on their exact opcodes and other **buts and ifs** :) At the end it will return the machine code of the given inputs.

Also, I defined the registers lists based on their size. And defining the registers values.

In calculation part, after considering the opcodes and assigning the registers values code, the string of integers will be converted to binary in the first level, and then then whole answer will be converted to the hexadecimal format.

At the end I consider a printing part that will print the machine code of each inputs in separate lines if the machine code is valid.

It is necessary to say that I specified different warnings for the different invalid inputs.

"It always seems impossible until it's done."

"Good Luck"