In this project, I use the Processor class to load file with *loadFile* method, conduct basic syntax check in *checkFormat*, and execute the program without dependence on the specific instructions. All supported instructions are grouped by their parameter lists into 3 Enum classes, which are *ZeroArgOperation*, *OneArgOperation* and *TwoArgOperation*. Each Enum class has an abstract *execute* method, which is implemented differently for each Enum instances in order to facilitate information hiding.

As for the extensibility of the project, adding a new assembly instruction NEW is relatively easy. For a future programmer, one should first add NEW into the corresponding Enum class and implement its *execute* method. Then in the Processor class NEW should be added to the correct HashSet to enable keyword identification and syntax check, and it is done.

However, for a different computer processor model, it would be harder for my interpreter to adjust into. Since the number and order of registers is hard-coded in my program, changing the register would require bunch of modifications and is error-prone. A potential improvement could be using more constant within the structure to replace the “magic numbers” currently in the program.

Finally, the interpreter supports multiple assembly instructions written in one line. Besides, character representation for the ASCII symbol is supported as well, such as LOADI R0 ‘a’ or LOADI R5 ‘ ’.