Write a MIPS simulator that consists of assembler and virtual machine.

Assembler:

The assembler is able to take an assembly program written in a subset of MIPS-32 instruction set and translate it into machine language. The assembler should be able to do the following:

1- Provide a GUI interface and allow the user to load a program:
   - By typing in the instructions directly.
   - Or by copying and pasting.
   - (Optional) By loading from a file.
2- Support the following instructions:
   - The basic functions without pseudo-instructions and without function calls.
   - Loads and stores: **`lw`**, **`sw`**
   - ALU operations: **`add`**, **`addi`**, **`sub`**, **`and`**, **`or`**, **`andi`**, **`ori`**, **`sll`**, **`slt`**, **`slti`**, **`lui`**
   - Branches/jumps: **`jr`**, **`j`**, **`beq`**, **`bne`** and **labels**. Assembler must decide jump destination.
3- The assembler will take as input the text of the assembly program and will output a list of machine language instructions. (See the assumptions in MIPS Virtual Machine section below)
4- Note: Your assembler does not work like QTSPIM. So no need to support data declarations, data types, data and text sections, etc. You will get input programs in pure MIPS assembly.

MIPS-32 Virtual Machine:

The virtual machine simulates the structure of MIPS-32 processor **and can run the machine language program produced by the assembler**. Your machine should be able to do the following:

1- Provide a GUI to allow the user to see the status of the **Program Counter,** MIPS **registers** (or at least the important ones), the relevant part of the **memory** (were used variables are stored) and the **currently executing instructions** and its type (R, I, J), parts like opcode, registers, etc.
2- The machine will have buttons to execute the next instruction or the entire program.
3- Make the following assumptions:
   - Memory has only two segments: Text (Program) segment and Data segment.
   - The program is always loaded starting from memory location **0**. (Start of text segment)
   - Data is stored starting from memory location **0x00001000**. (Start of Data segment)

4- A close app is here https://www.facebook.com/groups/1749816315311531/permalink/1789809447978884/

Design and Development:

1- Develop the program in JAVA. Take advantage of JAVA Swing GUI components.
2- **Write file headers with date, author, version, etc. and which Java version / tools you used.**
3- Use object-oriented design properly.
    - First think of the classes your system should have in this system.
    - Next, think of the responsibility of each class and its attributes and methods to do its job.
    - Next, think of the external **API** (**public methods**) that other classes can use in this class.
    - Finally, think of how these classes cooperate to achieve the task of the program.
    - (Advanced - Optional) Consider using the **Command Design Pattern**.
4- How will you parse and assembly to (1) find if an instruction is valid or not, (2) what each instruction is and (3) the labels that accompany some instructions?
    - There is two ways to do so.
    - The first is ad-hoc parsing بالفهلوةusing Java regular expressions. Here are some links to learn more. But you kind find better resources:
        - http://zetcode.com/java/regex/
        - https://stackoverflow.com/questions/5688704/regular-expression-for-assembly-instructions
        - https://www.journaldev.com/634/regular-expression-in-java-regex-example

The second is using a proper parser or generating one using a parser generator. This **is the ideal but harder way.** A parser generator is a tool that generates a parser for a language given a "**grammar**" representing this language.