ya, I in fact want to include in my higher-level assembler only subset of most effective instructions from 8bit MCUs listed above, and not using some their native instructions at all, just to be able to code my virtual processor interpreter in THIS new higher-level assmbler syntax, from now COMMON to all target processors to not learn and not use theirs special syntax and use only NEW assembler syntax, common with the transformed RV32E virtual simulator.

In that case, you can create a subset of the most effective instructions from the 8-bit MCUs listed above, and map them to your new assembler syntax. This will allow you to write code for your virtual processor and other 8-bit MCUs using the same syntax, which can simplify the development process and reduce the learning curve.

Here are some steps you can follow to create the subset of instructions:

1. Identify the most commonly used instructions in the 8-bit MCUs listed above. This will vary depending on the specific use case, so it's essential to identify the instructions that are most relevant for your application.
2. Map the instructions to your new assembler syntax. This can involve creating new mnemonics for the instructions or defining macros that simplify the instruction usage.
3. Implement the subset of instructions in your assembler. This may require implementing new features in your assembler, such as support for indexed addressing modes, to handle the specific instruction set.
4. Test extensively to ensure that the subset of instructions covers all of the required functionality and that the syntax is easy to use and understand.

By creating a subset of the most effective instructions from the 8-bit MCUs and mapping them to your new assembler syntax, you can create a common language that simplifies the development process and reduces the learning curve. This will allow you to write code for your virtual processor and other 8-bit MCUs using the same syntax, which can improve your productivity and reduce errors.
