General Feedback

- Good job on the project!
- Well-defined project structure and maintaining every milestone.

Documentation

README.md

Feedback Item	Priority	Remarks
Begin with a short summary	**High**	Needs Improvement
Organize sections with headings	**High**	Needs Improvement
Add installation instructions (C++ compiler requirements and Qt setup, including version compatibility, makefile, etc.)	**High**	Needs Improvement
Consider adding a "Getting Started" section (Optional)	**Low**	Advice
Explain how to execute UVSim from the command line, with sample commands	**Low**	Advice
Use backticks (```) for code snippets and commands to distinguish them from the rest of the text	**Low**	Advice

Diagrams

Feedback Item	Priority	Remarks
Thank you for providing UI skeleton	-	Positive Feedback
Add Use Case Diagram	**High**	Needs Improvement
Add Sequence Diagram for the main functionality	**Medium**	Needs Improvement
Add State Diagram	**Low**	Needs Improvement

Coding

Naming Conventions

Feedback Item	Priority	Remarks
Variable names are meaningful and follow the camelCase convention	-	Excellent

Feedback Item	Priority	Remarks
Consistent naming scheme for variables and functions	-	Excellent

Documentation

Feedback Item	Priority	Remarks
Documentation is minimal; there is room for improvement	**Medium**	Needs Improvement
The code includes comments explaining complex logic	-	Good
Classes lack brief descriptions of their purposes	**Medium**	Needs Improvement
Some functions lack detailed descriptions of their purposes and parameters	**Medium**	Needs Improvement
More inline comments would be helpful to explain the code's logic	**Medium**	Needs Improvement

Best Practices

Feedback Item	Priority	Remarks
Certain functions, particularly accessor methods, could benefit from the use of `const` to prevent unintended modifications	**Low**	Advice

Error Handling

Feedback Item	Priority	Remarks
Code lacks comprehensive error handling. Implementing try-catch blocks and validating inputs would improve robustness	**Critical**	Needs Immediate Attention

Code Organization

Object-Oriented Design

Feedback Item	Priority	Remarks
Overall good job on the object-oriented design	-	Meets Expectations
Limited use of inheritance and polymorphism. Implementing an interface or abstract class for instruction types could improve flexibility and enable better extension	**High**	Needs Improvement

Feedback Item	Priority	Remarks
`MainWindow` handles many tasks (memory manipulation and UI management). Separate these responsibilities into one or more controller classes	**High**	Needs Improvement
`fourDigitInput` and `sixDigitInput` in `Instructions` don't need to be public	**Low**	Advice

Design Principles

Feedback Item	Priority	Remarks
Improve adherence to the Single Responsibility Principle	**High**	Needs Improvement
`MainWindow` has too many responsibilities	**High**	Needs Improvement
Redundant logic for handling four- and six-digit instructions; implement a more generic approach to handle both cases	**Medium**	Needs Improvement
Replace magic numbers (like 5, 7 for instruction length checks) with named constants	**Low**	Advice

Design Patterns

Feedback Item	Priority	Remarks
~~`Accumulator` class could benefit from the Singleton pattern~~	~~**Medium**~~	Advice
~~Implement Singleton pattern for `MainMemory` class to prevent multiple instances~~	**Medium**	~~Advice~~
Use Command pattern in `doInstruction` function to manage instructions; each instruction can have a dedicated class with a `run` method	~~**Medium**~~	~~Advice~~
Adopt Factory pattern to instantiate different instruction types (e.g., Read, Write)	**Medium**	Advice

Testing

Feedback Item	Priority	Remarks
Good attempt to cover different functionalities, including edge cases	-	Meets Expectations
Tests should check expected outputs more rigorously rather than using only printed messages (e.g., dividing by zero in `divideTests`)	**Medium**	Needs Improvement
Add tests for operations like branching on accumulator values to improve coverage	**Medium**	Needs Improvement

Feedback Item	Priority	Remarks
Use assertions instead of print statements for success/failure to clearly identify test results	**Medium**	Needs Improvement

Please ensure to address the high-priority and critical items first, as they have a significant impact on the overall quality and functionality of the project. The medium and low-priority items should be considered as you continue to refine and improve your application.

Keep up the good work, and continue to build upon the strong foundation you've established!