User Guide: Simple Memory Implementation

CSCI 6461: Computer System Architecture Team 10

March 1, 2025

1 Overview

This document serves as the user guide for the CSCI 6461 Machine Simulator. This simulator includes: a simple memory module, a controller, and a GUI. The memory module simulates a single-port memory containing 2048 words, each 16-bit wide. The control acts as the body of the simulator and it controls the memory access, loadfile reads, and instruction execution. The GUI is the face of the simulator, which visualizes the instruction/memory operations. This guide explains how to compile, run, and test the simulator, and describes its key operations.

2 System Requirements

- Java Development Kit (JDK): Version 1.8 or later.
- Development Environment: Any Java IDE (e.g., Eclipse, NetBeans) or a text editor with command line tools.
- Operating System: Windows, macOS, or Linux.

3 Installation and Compilation

3.1 Compiling the Code

1. Download the four files: Memory.java, Control.java, CPU.java, and GUI.java.
2. Open a terminal or command prompt and navigate to the directory containing each file
3. Compile the code for each file using the command:

   ```
   javac file.java  (with file being each of the four files listed above)
   ```

4. Upon successful compilation, a .class file will be created.
5. Repeat for each of the four .java files.

3.2 Creating an Executable JAR File (Optional)

1. Package the compiled classes into a JAR file:

   ```
   jar cvf Simulator.jar -C path/to/classes .
   ```

   **with path/to/classes being the folder in which you have all of the simulator files

2. Run the JAR file with:

   ```
   java -jar Simulator.jar
   ```

Note: This can also be done in Eclipse through Export > Jar > Select all files (and zip/jar location)

4 Running the Memory Module

4.1 Test Execution

The Memory class includes a main method that performs a basic test:

- Reads and displays the initial value at memory address 0 (expected to be 0).
- Writes a sample value (e.g., 12345) to memory address 0.
- Reads and displays the value at memory address 0 after the write operation.

The GUI includes a call to a given loadfile that performs a test of the simulator.

4.2 Expected Output

Running the program should produce output similar to:

```
Initial value at address 0: 0
Value at address 0 after write: 12345
```

5 Class Operations

The memory module supports the following operations:

- readWord(int address): Returns the 16-bit word stored at the specified address.
- writeWord(int address, short data): Writes a 16-bit word to the specified address.
- reset(): Clears the memory by setting all memory locations to zero.

All operations include bounds checking to ensure that the address is within the valid range (0 to 2047).

The control class supports the following operations:

- computeEA (byte ix, byte addr, byte indirect): Computes effective address based on provided inputs.
- stepSimulator(): Steps through simulator 1 step. Reads an instruction and performs the given operation.
- haltSimulator(): Halts simulator.
- isInstruction(int value, int address): Checks if instruction is valid
- storeData(): Stores data in memory.
- printRegisters(): As the name implies, it prints the contents of the registers.
- loadLF (String file) : Runs when user selects 'Init', loads loadfile into memory.
- ldr,str,lda,ldx,stx : Functions to perform the load/store operands.
- runSimulator(): Starts at memory location in the PC, reads instructions and performs operations until a HLT is read.

The GUI class supports the following operations:

- openFileChooser(): Opens file explorer to allow the user to select a loadfile.

- runSimulator(): Runs the simulator (calls control).
- storeData(): Stores current data.
- stepExecution():Steps through 1 simulator step (calls control).
- haltExecution(): Halts the execution of the simulator (calls control).
- updateMemoryDisplay(): Update the GUI with the contents of memory.
- updateGUI(): Updates the GUI with the contents of the CPU.
- updateRegisterDisplay(): Helper function for updateMemoryDisplay().

The CPU class supports the following operations:

- fetch(): Fetches instruction from memory.
- decode(): Decodes the instruction.
- execute(): Executes the instruction.
- isHalted(): Returns halted state.
- setHalted(boolean state): Sets halted state.

6 Troubleshooting

6.1 Compilation Issues

- Verify that JDK 1.8 or later is installed.
- Ensure that the file is named Memory.java and is in the correct directory.

6.2 Runtime Issues

- An IllegalArgumentException indicates an attempt to access an address outside the valid range (0 to 2047). Check the address values being used.
- Invalid instructions will lead to errors.
- Ensure your system meets the required specifications for running the simulator.

7 Conclusion

This user guide outlines how to compile, run, and test the load/store version of the CS6461 project simulator. For further details, consult the design notes and additional project documentation.