CSCI 6461: Computer Simulator
User Guide

Group 10

March 15, 2025

1 Introduction

This user guide provides instructions for setting up and using the CSCI 6461 Computer Simulator. The simulator is designed to execute programs written in the **C6461 instruction set architecture (ISA)**, supporting basic memory operations, arithmetic instructions, branching, and input/output.

2 System Requirements

To run the simulator, ensure the following requirements are met:

- **Operating System:** Windows, macOS, or Linux
- **Java Version:** JDK 8 or higher
- **Memory Requirement:** At least 512MB RAM
- **IDE or Command Line:** The project can be executed using an IDE such as IntelliJ, Eclipse, or via the terminal.

3 Project Structure

The simulator consists of several key components:

- **CPU.java** - Manages instruction execution.
- **Memory.java** - Implements memory and cache management.
- **Control.java** - Handles instruction decoding and execution control.
- **GUI.java** - Provides a graphical user interface for user interaction.
- **Simulator.jar** - A compiled executable version of the project.
- **load.txt** - Stores machine code instructions to be loaded into memory.

4 Installation and Setup

4.1 Option 1: Running in an IDE

1. Open an IDE (e.g., IntelliJ IDEA, Eclipse, or VS Code).
2. Import the Java project files.
3. Ensure that JDK 8 or later is installed.
4. Compile and run the **GUI.java** file.

4.2 Option 2: Running from the Command Line

1. Open a terminal or command prompt.
2. Navigate to the directory containing the **Simulator.jar** file.
3. Run the following command:

Listing 1: Running the Simulator

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

5 Using the Simulator

5.1 Loading a Program

1. Click the **"Load"** button in the GUI.
2. Select a **load.txt** file containing machine code instructions.
3. The memory table will update to show the loaded program.

5.2 Executing Instructions

- **Run:** Click the "Run" button to execute the loaded program continuously.
- **Step:** Click the "Step" button to execute one instruction at a time.
- **Halt:** Click the "Halt" button to stop execution immediately.

5.3 Register and Memory View

- The simulator displays the current values of registers (**GPRs, IXRs, PC, IR, MAR, MBR**).
- The memory section shows stored values.
- The cache area displays memory blocks currently cached.

6 Writing and Running Test Programs

6.1 Format of load.txt

Each instruction in **load.txt** is written in **octal format**, with the structure:

```
[memory address] [instruction]
```

Example:

```
000012 102106
000013 002027
```

6.2 Running the Test Programs

1. Write the **assembly program** in the C6461 instruction set.
2. Convert the instructions into **machine code (octal)**.
3. Save the machine code in **load.txt**.
4. Load the file into the simulator.
5. Execute the program using the **Run** or **Step** buttons.

6.3 Example Test Programs

Two test programs are included in this project:

6.3.1 Test 1: Closest Number Finder

- Reads **20 signed integers** from user input.
- Accepts a **target number**.
- Finds and prints the **closest number** from the list.

6.3.2 Test 2: Word Search in a Paragraph

- Reads **six sentences** from a file.
- Prints the sentences on the console.
- Asks the user for a **word to search**.
- Searches for the word and prints its **sentence number and position**.

7 Troubleshooting and Debugging

7.1 Common Errors and Fixes

- **Simulator does not start:** Ensure that **Java is installed** and the correct JDK version is used.
- **Machine code is not executing correctly:** Verify that the instructions in **load.txt** follow the correct format.
- **Incorrect instruction execution:** Use the **Step** function to debug step-by-step and check the instruction decoding output.
- **Jump instructions not working:** Ensure that the **effective address (EA) computation** is correctly implemented.

7.2 Debugging Mode

The simulator includes debug messages for instruction execution. Debugging output can be enabled in **Control.java** by checking:

Listing 2: Debugging Example

```
System.out.println("[DEBUG] Executing Instruction at PC: " + mem.PC);
```

8 Future Improvements

- Implement **floating-point arithmetic** for enhanced operations.
- Optimize **cache performance** with additional replacement policies.
- Add support for **vector processing** and advanced memory models.

9 Conclusion

This user guide provides a comprehensive overview of how to set up, run, and troubleshoot the **CSCI 6461 Computer Simulator**. The project successfully implements a working instruction execution cycle, supporting test programs in **C6461 assembly**. Future improvements can extend its functionality for more complex operations.