

CISC Simulator Design Document

Rev 2.0

Group 2

Jonathan Pritchett

Ziwei Li

Haorong Xiao

Siyuan Zhang

Table of Contents

| | |
|--------------------------|----------|
| Table of Contents | 1 |
| Description | 2 |
| Data Representation | 2 |
| Registers | 2 |
| Memory | 2 |
| Instructions | 3 |
| Front Panel GUI | 3 |
| Figures | 4 |
| UI Mockup | 4 |
| Component Diagram | 5 |
| Class Diagram | 6 |

Description

The CISC Simulator is a program that creates a simple simulator of a rudimentary computer. In its current iteration, this simulated computer consists of a set of registers, a simple memory, a small set of instructions, and a graphical user interface to serve as the computer front panel.

Data Representation

Data in the computer shall be stored in 16-bit words. To represent this, a class “Word” will be created which stores a 16-bit word as a 16 char array. This class will have accessor methods allowing for easy translation of this data into ints and strings and will allow for accessing of individual and subsets of bits.

Registers

In this first iteration of the CISC simulator, the following 11 16-bit registers are used.

| | |
|---------|-----------------------------|
| R0...R3 | 4 General-purpose registers |
| PC | Program counter |
| IR | Instruction Register |
| MAR | Memory Address Register |
| MBR | Memory Buffer Register |
| X1...X3 | 3 Index Registers |

Each register consists of a single Word object. All of the registers are collectively stored within a single class, “Registers”, which may be instantiated once and shared among the components of the simulator.

Memory

The Memory system for this version consists of a single class that contains an array of 2048 “Word” objects. The memory is instantiated once and shared throughout the simulator.

Instructions

Five instructions are currently implemented within the simulator.

| Instruction | OpCode | Description |
|-------------|--------|---------------------------------|
| LDR | 01 | Load register from memory |
| STR | 02 | Store register to memory |
| LDA | 03 | Load register with address |
| LDX | 41 | Load index register from memory |
| STX | 42 | Store index register to memory |

Front Panel GUI

The front panel GUI provides the user with a way to view the contents of all of the registers and memory locations used by the simulator. It also allows the user to override any of these values manually and to run the simulator from its current state.

Figures

UI Mockup

Simple CPU Simulator

Registers

PC: 2752

IR: 1000100010110

R0: 45

R1: 782

R2: 7852

R3: 783

X1: 123

X2: 654

X3: 457

MAR: 3123

MBR: 7839

MFR: 4537

Memory

Address: 16

Value: 2146

☐ Override Locked

Override All Values

Load Test Program

☐ AutoRun

Step

Input Mode:

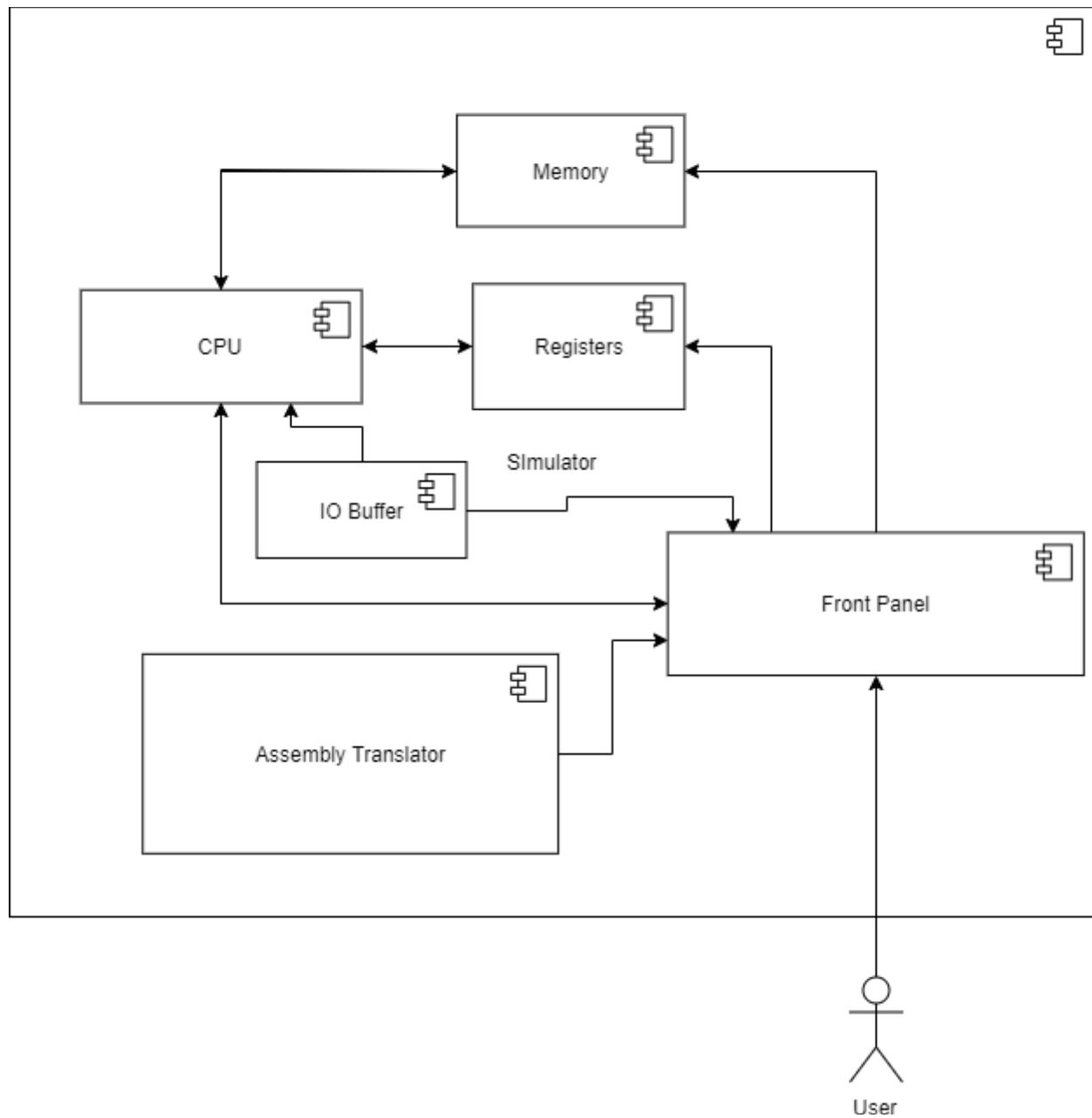
☐ Binary

☐ Octal

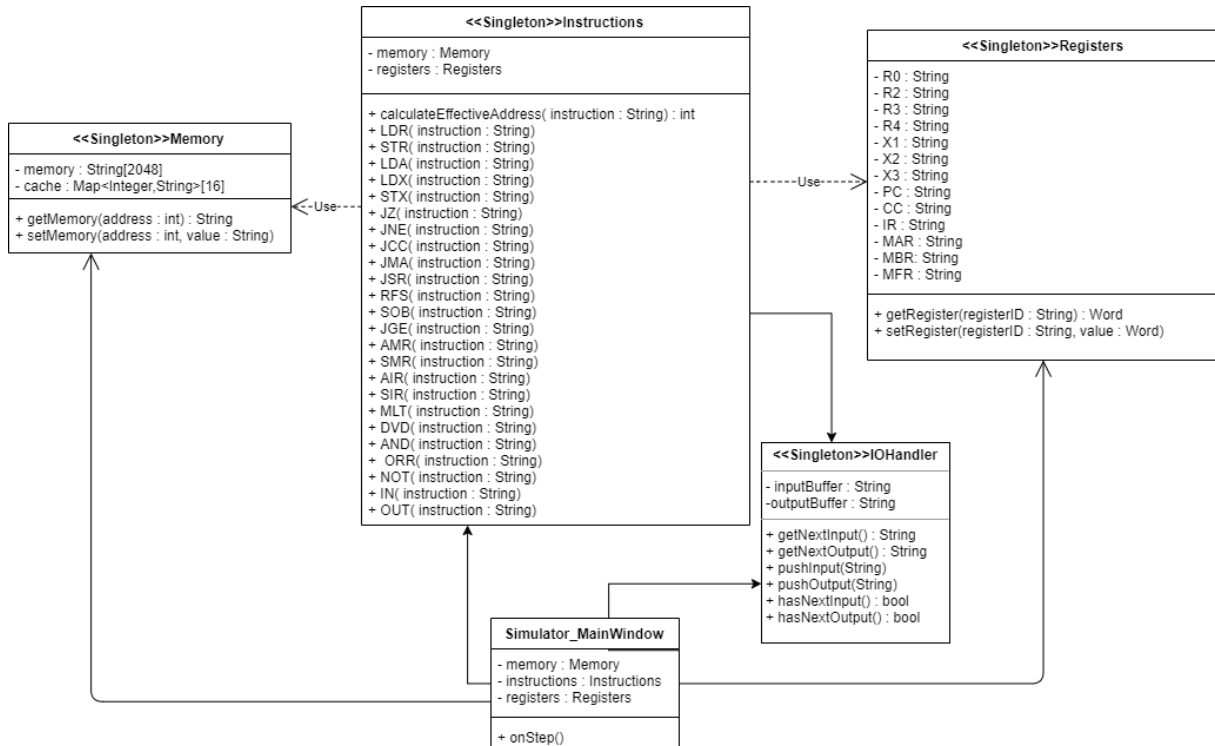
☒ Decimal

☐ Hex

Component Diagram



Class Diagram



Test Program 1

We initialize the test program by pressing the “load test program” button in the lower right corner.

We set up 4 memory addresses and 2 index registers:

- memory address 31 stores integer value 29
- memory address 30 stores integer value 31
- memory address 29 stores integer value 30
- index register X1 stores integer value 1 and index register X2 stores integer value 2.

Then, we have stored 5 instructions in memory:

- (1) Load register from memory: 000001 00 01 1 11110. (EA = memory[31] = 29)
This instruction loads effective address 29 by indexing and indirecting to register R0, which stores 30 representing by '11110'.

The screenshot shows a CISC Simulator window with a light green title bar. The main area is a light gray panel containing several input fields for registers and memory. The registers are labeled R0, R1, R2, R3, X1, X2, X3, PC, CC, IR, MAR, MBR, and MFR. The memory section is labeled 'Memory' and contains fields for 'Address' and 'Value'. There are also buttons for 'Override All Values', 'Load test program', 'AutoRun?', and 'Step'. The R0 register field is highlighted with a red rectangle.

| Register | Value |
|----------|------------------|
| R0 | 0000000000011110 |
| R1 | 0000000000000000 |
| R2 | 0000000000000000 |
| R3 | 0000000000000000 |
| X1 | 0000000000000001 |
| X2 | 0000000000000010 |
| X3 | 0000000000000000 |
| PC | 1011 |
| CC | 0000000000000000 |
| IR | 0000010001111110 |
| MAR | 11101 |
| MBR | 0000000000011110 |
| MFR | 0000000000000000 |

Memory

| Field | Value |
|---------|------------------|
| Address | 0000000000000000 |
| Value | 0000000000000000 |

Buttons: Override All Values, Load test program, AutoRun? (checkbox), Step

(2) Store register to memory: 000010 00 00 0 11100. (EA = 28)

This instruction stores the value of R0 to the effective address 28. Then, the content of memory address 28 become 30.

The screenshot displays the CISC Simulator interface. On the left, a list of registers and their values is shown: R0 (0000000000011110), R1 (0000000000000000), R2 (0000000000000000), R3 (0000000000000000), X1 (0000000000000001), X2 (0000000000000010), X3 (0000000000000000), PC (1011), CC (0000000000000000), IR (0000010001111110), MAR (11101), MBR (0000000000011110), and MFR (0000000000000000). On the right, the 'Input Mode' is set to 'binary' and the 'Instruction' is 'Item 1'. At the bottom left, a 'Memory' section is highlighted with a red box, showing 'Address' as '11100' and 'Value' as '0000000000011110'. To the right of the memory section are buttons for 'Override All Values', 'Load test program', and 'Step', along with a 'Lock Override' checkbox and an 'AutoRun?' checkbox.

- (3) Load register with address : 000011 01 01 1 11101. (EA = memory[30] = 31)
This instruction loads the content in effective address to register R1 by indexing and indirecting. The effective address here is 31. Therefore, 31 is stored in R1.

The screenshot shows the CISC Simulator interface with the following components:

- Registers:** R0 (0000000000011110), R1 (11111), R2 (0000000000000000), R3 (0000000000000000), X1 (0000000000000001), X2 (0000000000000010), X3 (0000000000000000).
- Control and Status:** PC (1101), CC (0000000000000000), IR (0000110101111101), MAR (11100), MBR (0000000000011110), MFR (0000000000000000).
- Memory:** Address (11100), Value (0000000000011110).
- Input and Instruction:** Input Mode (binary), Instruction (Item 1).
- Buttons and Controls:** Override All Values, Load test program, Lock Override (checkbox), AutoRun? (checkbox), Step.

(4) Load index register from memory: 101001 00 10 1 11100. (EA = memory[30] = 31)

This instruction loads the content in effective address to index register X2 by indexing and indirecting. Consequently, the content of Index Register 2 becomes 29.

The screenshot displays the CISC Simulator interface with various registers and memory fields. The registers shown are R0, R1, R2, R3, X1, X2, X3, PC, CC, IR, MAR, MBR, and MFR. The values for these registers are as follows:

| Register | Value |
|----------|------------------|
| R0 | 0000000000011110 |
| R1 | 11111 |
| R2 | 0000000000000000 |
| R3 | 0000000000000000 |
| X1 | 0000000000000001 |
| X2 | 0000000000011101 |
| X3 | 0000000000000000 |
| PC | 1110 |
| CC | 0000000000000000 |
| IR | 1010010010111100 |
| MAR | 11111 |
| MBR | 0000000000011101 |
| MFR | 0000000000000000 |

The X2 register value, 0000000000011101, is highlighted with a red rectangle. Below the registers, the Memory section shows the Address as 11100 and the Value as 0000000000011110. On the right side, there are controls for Input Mode (binary), Instruction (Item 1), and buttons for Override All Values, Load test program, AutoRun?, and Step.

(5) Store index register to memory: 101010 00 10 0 00000. (EA = 29)

This instruction stores the value of index register 2 to memory[29].

CISC Simulator Design Document
Rev 1.0

The screenshot displays the CISC Simulator interface, which includes a title bar with standard window controls. The main area contains several input fields for registers and memory, along with control buttons. The registers and their values are as follows:

| Register | Value |
|----------|------------------|
| R0 | 0000000000011110 |
| R1 | 11111 |
| R2 | 0000000000000000 |
| R3 | 0000000000000000 |
| X1 | 0000000000000001 |
| X2 | 0000000000011101 |
| X3 | 0000000000000000 |
| PC | 1111 |
| CC | 0000000000000000 |
| IR | 1010100010000000 |
| MAR | 11101 |
| MBR | 0000000000011110 |
| MFR | 0000000000000000 |

Below the registers, there is a "Memory" section with two input fields: "Address" (containing "11101") and "Value" (containing "0000000000011101"). These two fields are highlighted with a red rectangular border. To the right of the memory fields are two buttons: "Override All Values" and "Load test program". Below these buttons are two checkboxes: "Lock Override" (unchecked) and "AutoRun?" (unchecked). A "Step" button is located to the right of the "AutoRun?" checkbox.

Test Program 2

Here is the general idea of our code. We store the paragraph and the word to find in the memory. Every character is stored in an address. We keep recording the character until we meet a space, a period or a question mark. Then, we record the word number and the sentence number. After that, we compare the word we record for now and the word supposed to find. If it is the same, the program will output the word number and the sentence number. Otherwise, it will keep recording the next word and compare until it finds the answer.