Experiment No 1

Introduction to Computer Architecture and Software Tools

1.1 Objectives:

After completing this experiment, student will be able to:

- Understand the fundamentals of computer architecture and Organization.
- Understand the essential software tools including LogiSim and Venus.
- Explore LogiSim as a logic simulator for designing and simulating digital circuits.
- Explore Venus as a RISC-V simulator aiding in the comprehension of assembly language programming.

1.2 Background Theory

Basic computer architecture refers to the fundamental structure and organization of a computer system, outlining how its various components interact and function. At its core, computer architecture encompasses the central processing unit (CPU), memory, input/output devices, and the interconnection pathways. The CPU is the brain of the system, responsible for executing instructions, while memory stores data and instructions for quick access. The instruction set architecture defines the set of instructions the CPU can execute. The flow of data between the CPU and memory, as well as external devices, is managed through buses. The von Neumann architecture, a prevalent model, features a single shared memory space for both data and instructions. This architecture has evolved over time, with innovations like pipelining, parallel processing, and multiple cores enhancing computational capabilities. Understanding computer architecture is crucial for both hardware and software development, as it lays the foundation for efficient and effective computing systems.

LogiSim is a graphical, user-friendly logic simulator designed for educational purposes in the field of digital logic and computer architecture. It enables users, particularly students, to create and simulate digital circuits through a drag-and-drop interface. With a wide array of pre-built logic gates and components, LogiSim allows the design and analysis of both combinational and sequential circuits. Its real-time simulation feature aids in understanding the functionality of digital circuits, making it a valuable tool for hands-on learning in the realm of logic design.

Venus, on the other hand, is a RISC-V simulator geared towards facilitating the study of assembly language programming and computer architecture concepts. Supporting various RISC-V

instruction set extensions, Venus provides an environment for users to write, debug, and execute RISC-V assembly programs. It enables users to visualize the execution of each instruction, observe changes in registers and memory, and gain insights into the workings of a RISC-V processor. Venus serves as a powerful educational tool for students and enthusiasts seeking to delve into the intricacies of computer architecture and assembly programming.

1.2.1 Basic Computer Architecture Components

1. **Central Processing Unit (CPU)**
    - The brain of the computer responsible for executing instructions.
    - Comprises the control unit, arithmetic logic unit (ALU), and registers.
    - Interprets and executes instructions stored in the computer's memory.
2. **Memory**
    - Stores data and instructions that the CPU accesses during operations.
    - Divided into primary memory (RAM) for temporary storage and secondary memory (e.g., hard drives) for long-term storage.
3. **Input/Output (I/O) Devices**
    - Facilitate communication between the computer and the external world.
    - Examples include keyboards, mice, monitors, printers, and storage devices.
4. **Motherboard**
    - Main circuit board connecting all components.
    - Contains CPU socket, memory slots, expansion slots, and interfaces for peripheral devices.
5. **Bus System**
    - Communication pathways connecting various components.
    - Data bus transfers information between the CPU, memory, and peripherals.
    - Address bus specifies locations in memory, and the control bus manages data flow.
6. **Registers**
    - Small, high-speed storage locations within the CPU.
    - Hold data, addresses, and control information for quick access by the CPU.
7. **Arithmetic Logic Unit (ALU)**
    - Performs mathematical and logical operations.
    - Executes instructions, such as addition, subtraction, AND, OR, and NOT.
8. **Control Unit**
    - Manages the flow of data and instructions within the CPU.
    - Decodes instructions, initiates operations, and controls data movement.
9. **Clock**
    - Synchronizes operations within the computer.
    - The clock signal regulates the timing of instruction execution and data transfer.
10. **Cache Memory**
    - High-speed, small-sized memory that stores frequently used instructions and data.
    - Improves CPU performance by reducing the time needed to fetch information from main memory.

11. Power Supply

- Provides electrical power to all components.
- Converts electrical power from an external source to voltages suitable for the computer's operation.

12. Expansion Cards

- Additional circuit boards that enhance the computer's capabilities.
- Examples include graphics cards, sound cards, and network interface cards.

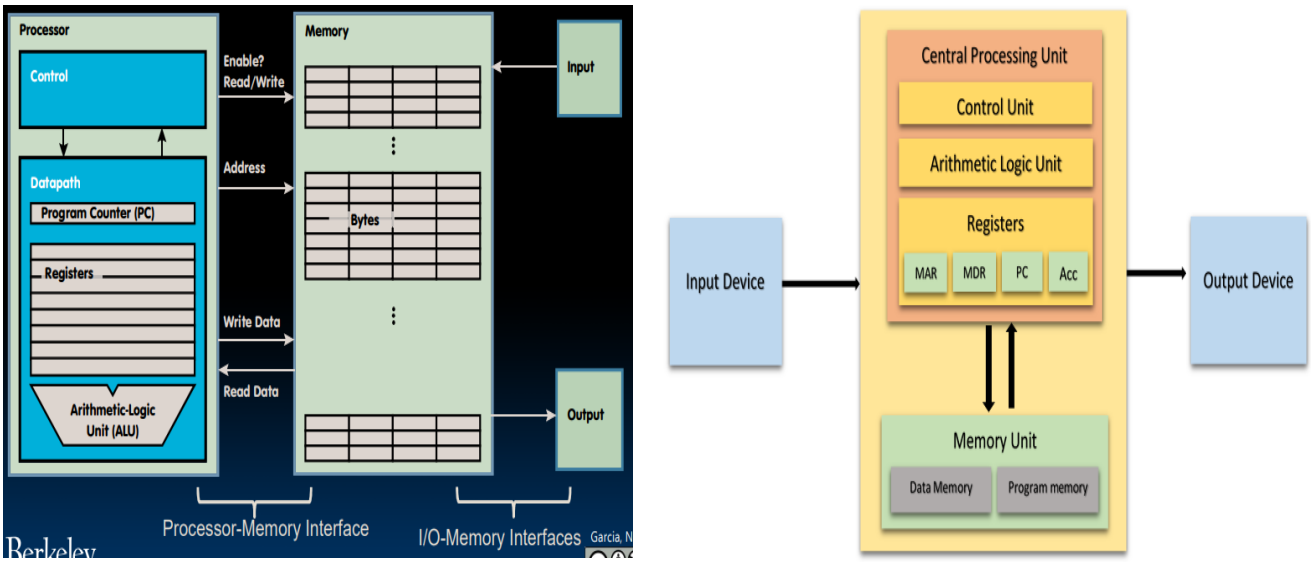


Figure 1.1: Basic Computer Architecture

1.2.2 LogiSim

Logisim is an educational tool for designing and simulating digital logic circuits. With its simple toolbar interface and simulation of circuits as they are built, it is simple enough to facilitate learning the most basic concepts related to logic circuits. With the capacity to build larger circuits from smaller subcircuits, and to draw bundles of wires with a single mouse drag, Logisim can be used (and is used) to design and simulate entire CPUs for educational purposes.

Students at colleges and universities around the world use Logisim for a variety of purposes, including:

1. A module in general-education computer science surveys
2. A unit in sophomore-level computer organization courses
3. Over a full semester in upper-division computer architecture courses

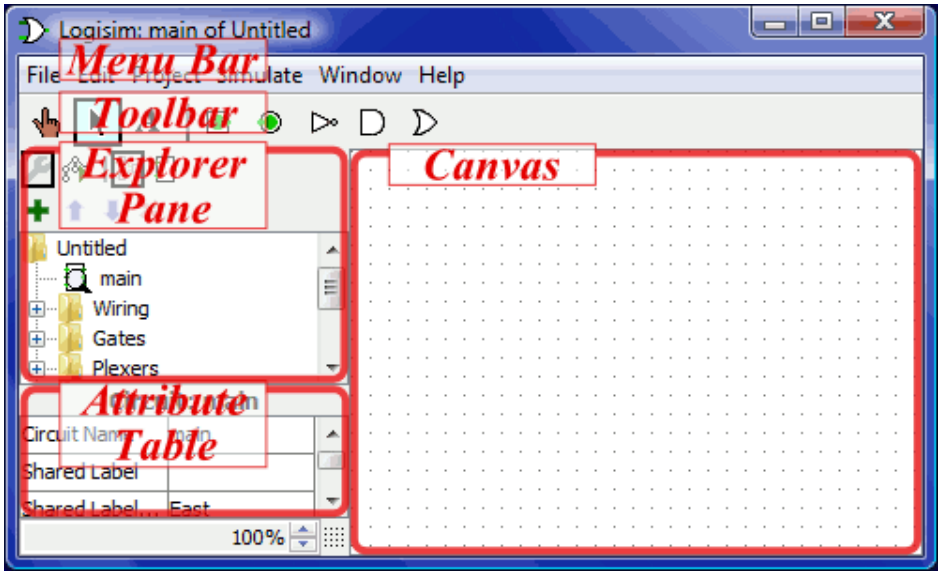


Figure 1.2: Logisim Interface

The explorer pane

Logisim organizes tools into libraries. They are displayed as folders in the explorer pane; to access a library's components, you have only to double-click the corresponding folder. Below, I have opened the Gates library and selected the NAND tool from it. You can see that Logisim now stands ready to add NAND gates into the circuit.

If you look through the choices in the Gates library, you'll notice that there was no need for us to develop a XOR circuit earlier: It's built into Logisim.

When you create a project, it automatically includes several libraries:

1. Wiring: Components that interact directly with wires.
2. Gates: Components that perform simple logic functions.
3. Plexers: More complex combinational components, like multiplexers and decoders.
4. Arithmetic: Components that perform arithmetic.
5. Memory: Components that remember data, like flip-flops, registers, and RAM.
6. I/O: Components that exist for the purpose of interacting with the user.

7. Base: Tools that are integral to using Logisim, though you probably won't need to dig into this library very often.

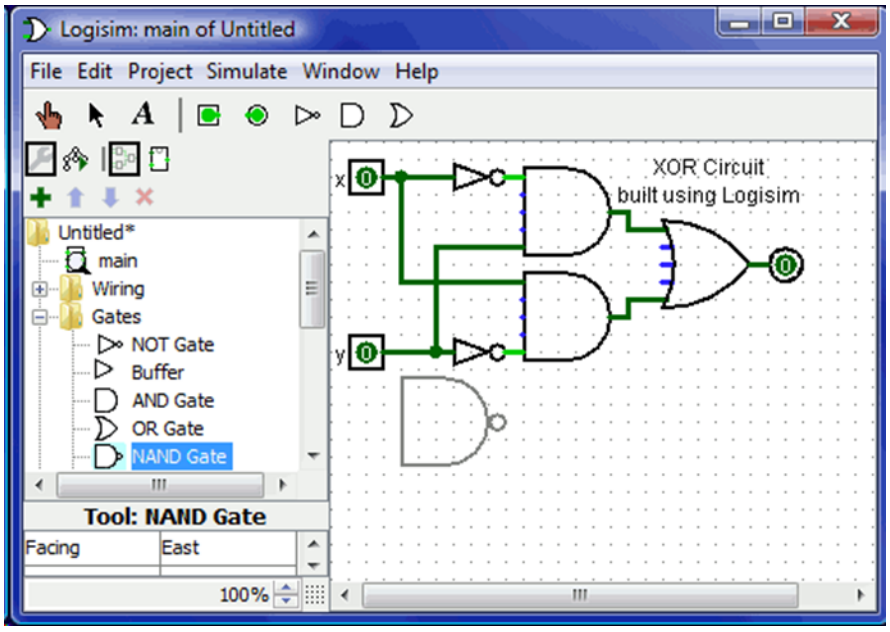


Figure 1.3: Explorer Pane

The attribute table

Many components have **attributes**, which are properties for configuring how the component behaves or appears. The **attribute table** is for viewing and displaying a component's attribute values.

To select which component's attributes you wish to view, click the component using the Edit tool ( ). (You can also right-click (or control-click) the component and choose Show Attributes from the popup menu. Also, manipulating a component via the Poke tool ( ) or the Text tool ( ) will display that component's attributes.)

The below screen shot demonstrates what things look like after selecting the upper input of our XOR circuit and scrolling down to view the Label Font attribute.

To modify an attribute value, click on the value. The interface for modifying the attribute will depend on which attribute you are changing; in the case of the Label Font attribute, a dialog box will appear for selecting the new font; but some attributes (like Label) will allow you to edit the value as a text field, while others (like Label Location) will display a drop-down menu from which to select the value.

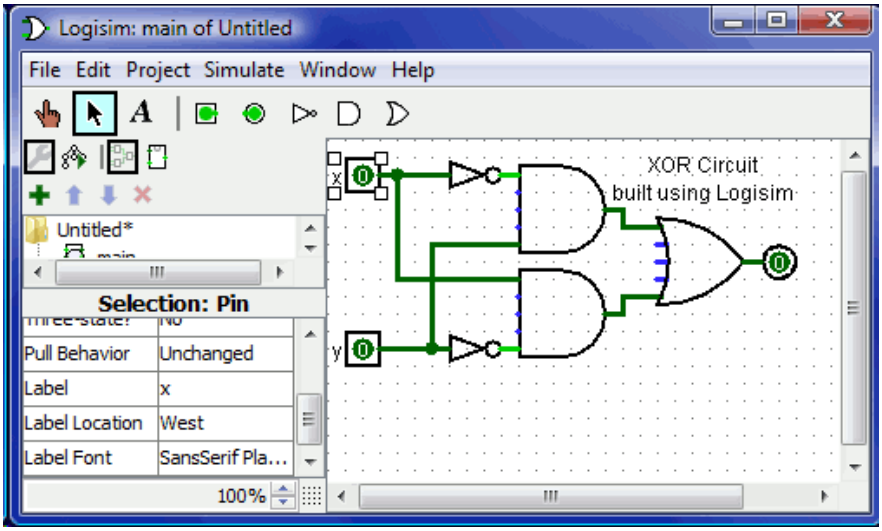


Figure 1.4: The Attribute Table

Tool attributes

Every tool for adding components to a circuit also has a set of attributes, which are imparted to the components created by the tool, although the components' attributes may be changed later without affecting the tool's attributes. When you select a tool, Logisim will change the attribute table to display that tool's attributes.

For example, suppose we want to create smaller AND gates. Right now, each time we select the AND tool, it creates a large AND gate. But if we edit the Gate Size attribute just after selecting the tool (before placing its AND gate into the circuit), we'll be changing the attributes for the tool, so that future AND gates added using the tool would be narrow instead.

With some tools, the tool's icon reflects some of the attributes' values. One example of this is the Pin tool, whose icon faces the same way as its Facing attribute says.

The tools in the toolbar each have a separate attribute set from the corresponding tools in the explorer pane. Thus, even though we changed the toolbar's AND tool to create narrow AND gates,

the AND tool in the Gates library will still create wide AND gates unless you change its attributes too.

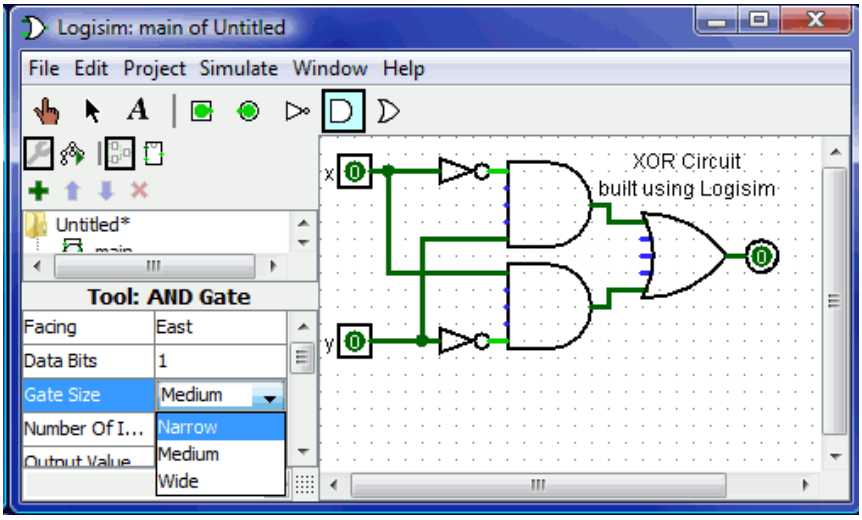


Figure 1.5: The Attribute Table

How to Get Started:

1. **Download and Install:**
    - Visit the Logisim website and download the latest version suitable for your operating system. Install it following the provided instructions.
2. **Create a New Circuit:**
    - Open Logisim and create a new project. Start designing your circuit by adding components from the toolbar.
3. **Wire Components:**
    - Connect components by dragging wires between them. Establish logical connections according to your design.
4. **Simulation:**
    - Use the simulation tool to test your circuit. Provide input values and observe the output behavior.
5. **Save and Export:**
    - Save your project for future editing. You can also export your circuit as a file or an image.

1.2.3 Venus Risc-V Simulator

The Visual Studio Code extension embeds the popular Venus RISC-V simulator. It provides a standalone learning environment as no other tools are needed. It runs RISC-V assembly code with the standard debugging capabilities of VS Code.

To use it as educational tool, further views are added as described below.

1. **Installation**

   You can easily install the extension via the marketplace by searching for "riscv":

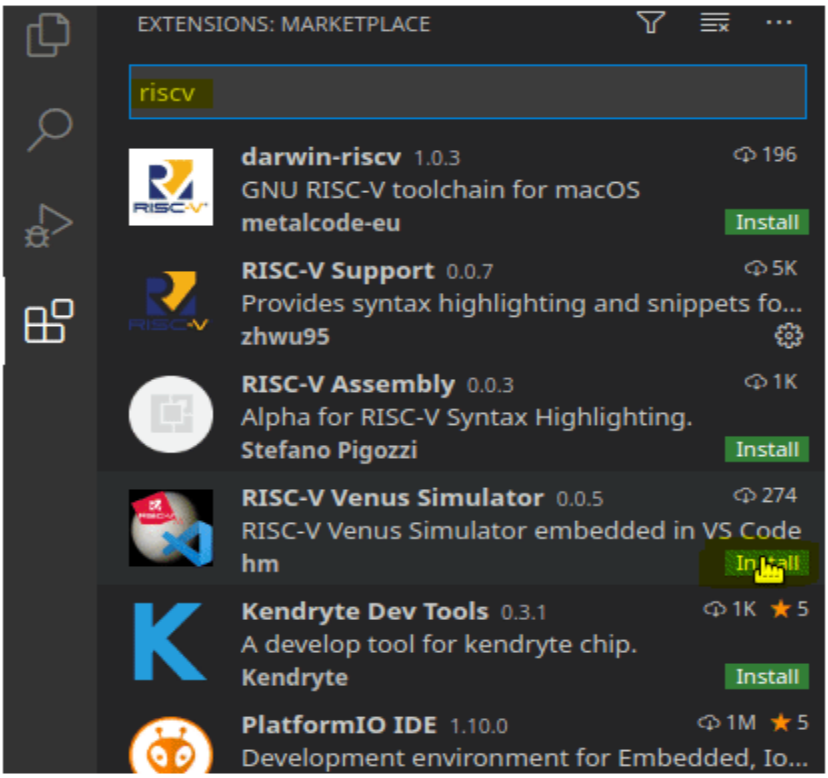


Figure 1.6: Venus Installation

Alternatively, you can install it via VS Code Quick Open (CTRL+P):

```
ext install hm.riscv-venus
```

2. Getting Started

You can debug any assembler file you are editing. We recommend using the RISC-V Support extension for syntax highlighting.

Just start debugging with "Run and Debug" to debug the current file:

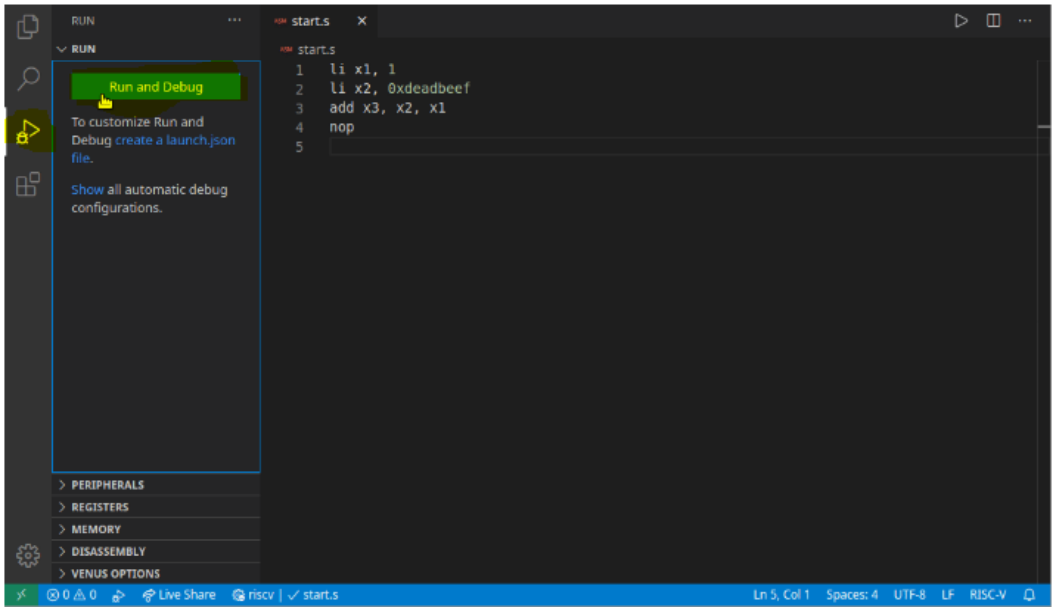


Figure 1.7: Run and Debug

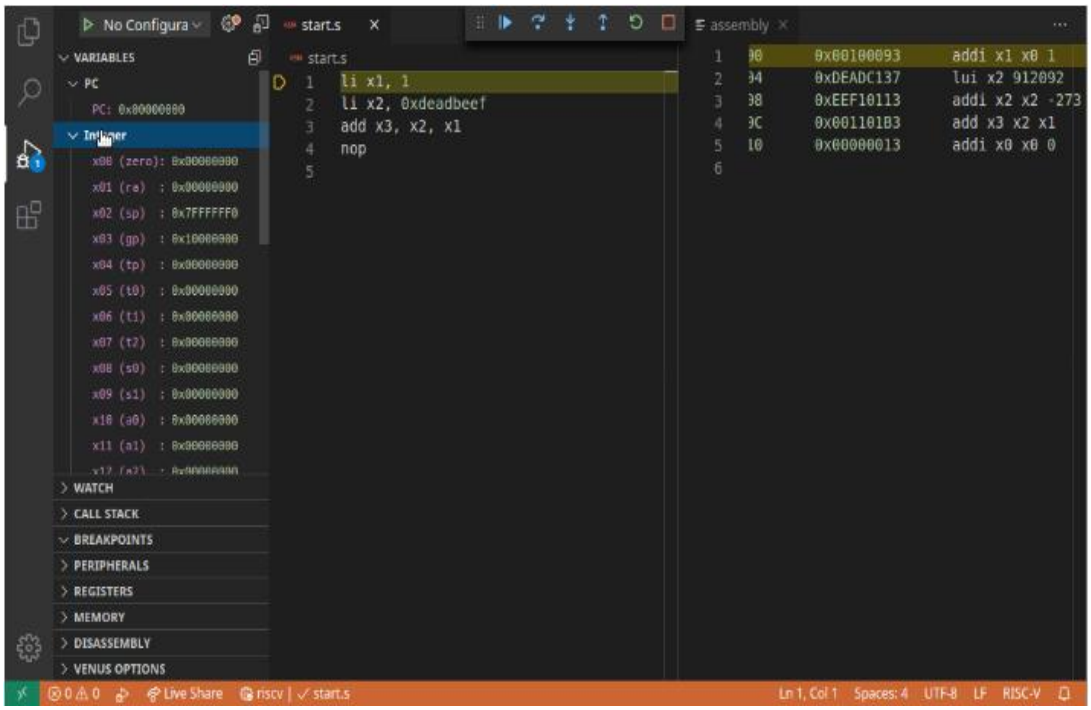


Figure 1.8: Instruction Execution

You can step through the code and pseudo-operations are expanded automatically. The actually executed instructions are displayed in an assembly view (auto-opens if set in extension-settings). Breakpoints are of course also supported.

The basic Venus environmental calls are also supported. Further environment calls are also supported for educational toy examples are described in the following.

3. UI and launch paramters

From, the Debugger you can open the "Venus Options". From there you can open settings, views, documentation and more. Also there are commands for all these actions which you can execute with CTRL+P and typing "Venus: ..."

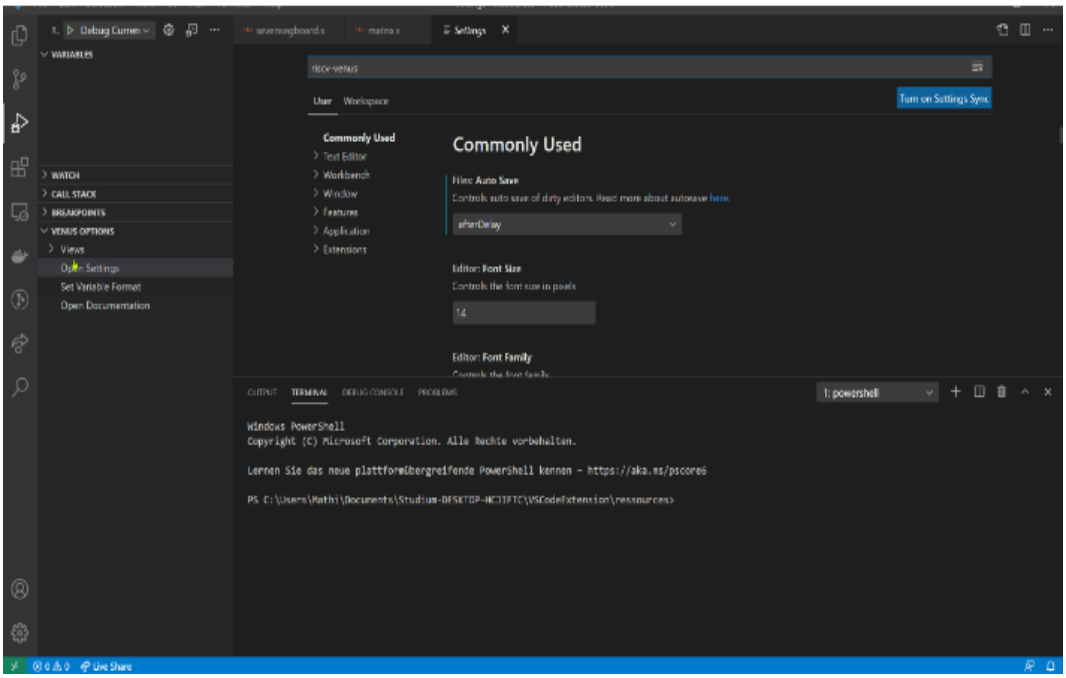


Figure 1.9: UI and Launch Parameters

4. View memory

You can view the memory contents by opening the memory view with CTRL+P and "Venus: Open Memory".

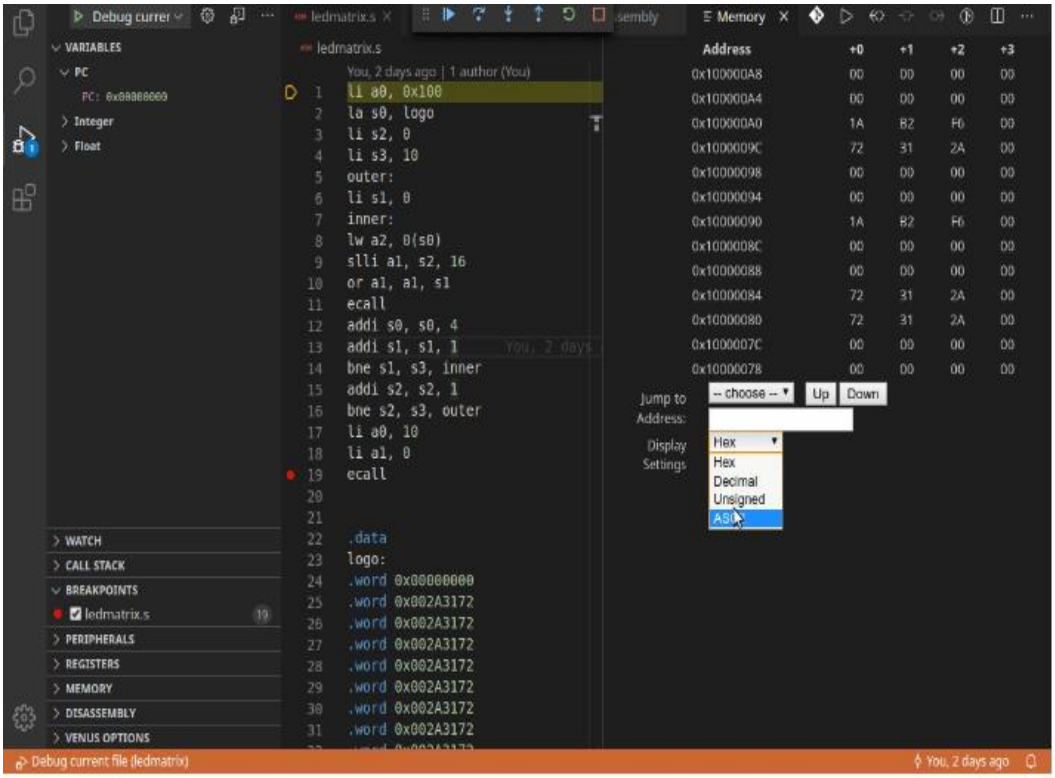


Figure 1.10: View Memory

You can select which segment to view and scroll through the memory words (as in the online Venus version).

5. Accessing registers

You can change the registers by clicking on their value, update the value and press enter.

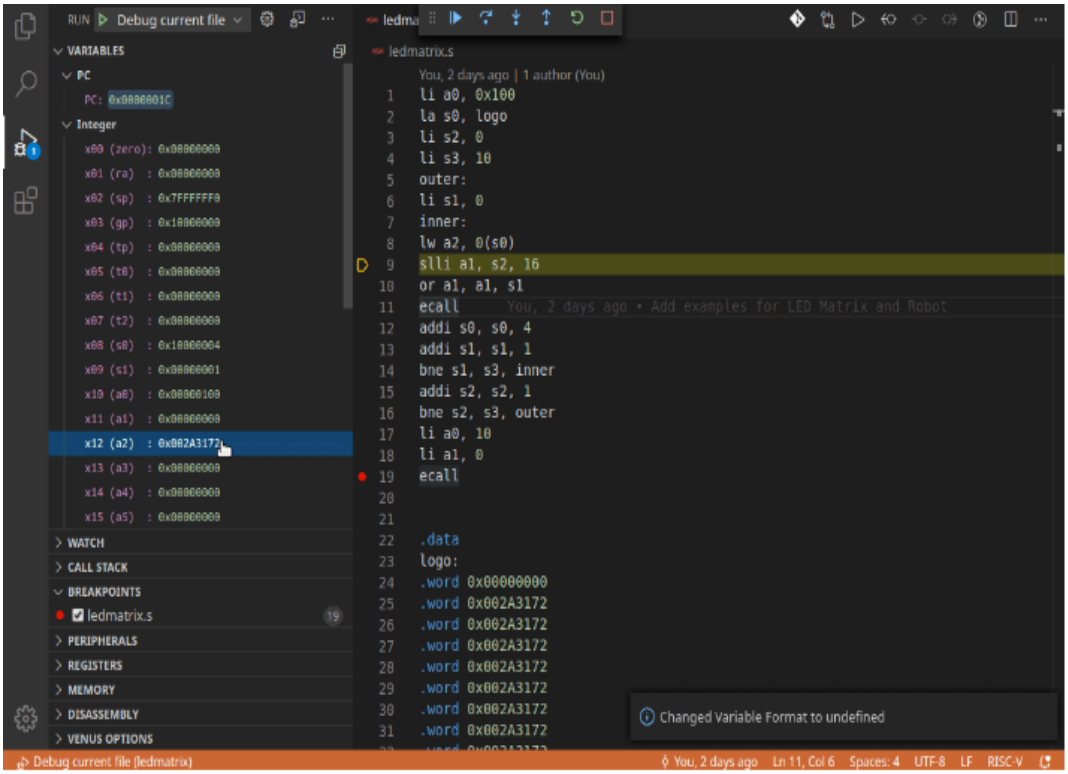


Figure 1.11: Accessing Registers

To change to format of the registers, click on "Venus Options" or press CTRL+P and enter "Venus: Set Variable Format".

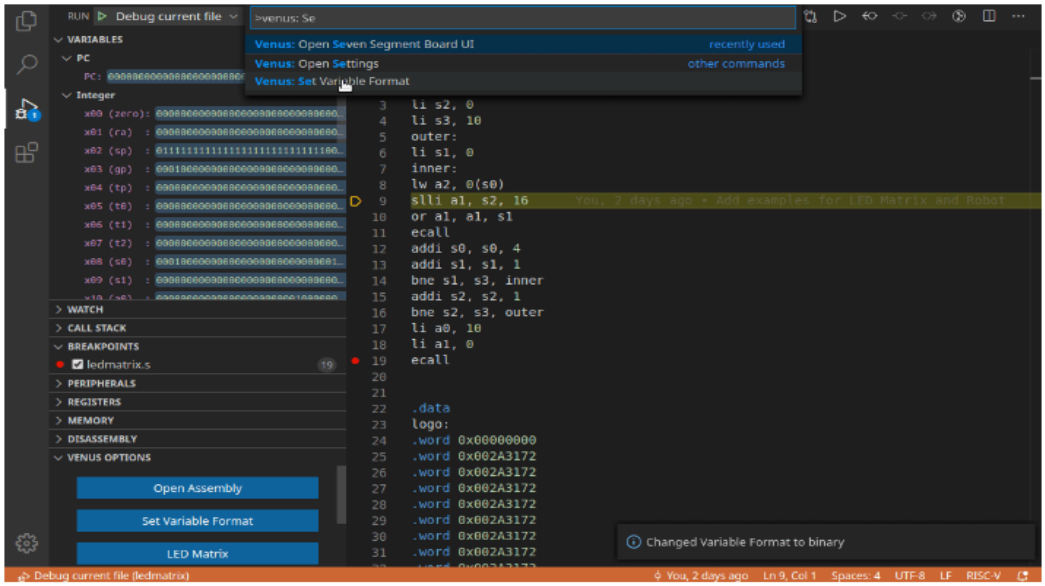
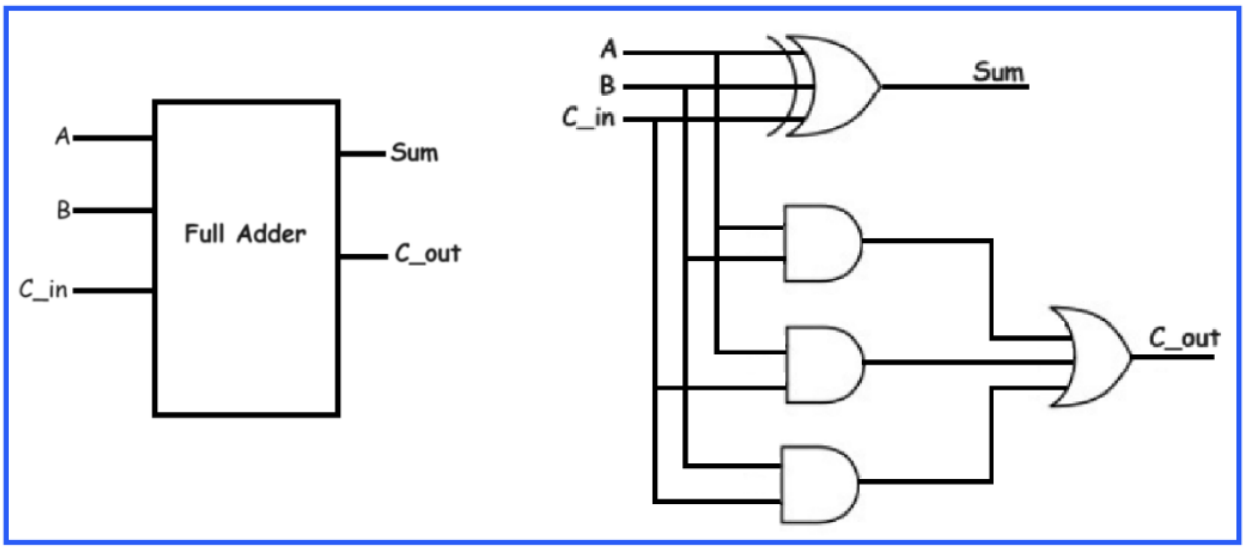


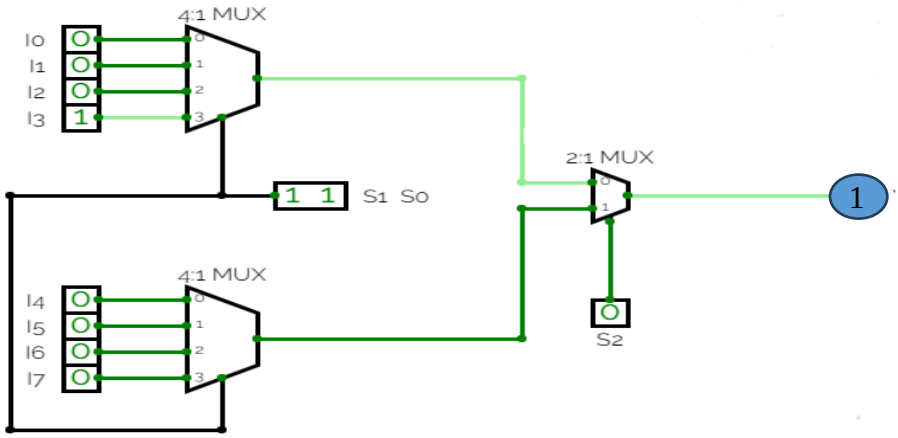
Figure 1.12: Formatting the Registers

1.3 Lab Activities

1.3.1 Lab Task-1: Implement and Verify the Following Full Adder in Logisim



1.3.2 Lab Task-2: Implement and Verify 8:1 Mux using 4:1 Mux using LogiSim



Sr. No	S0	S1	S2	I0	I1	I2	I3	I4	I5	I6	I7	Y
1	0	0	0	0	X	X	X	X	X	X	X	0
2	0	0	0	1	X	X	X	X	X	X	X	1
3	0	0	1	X	0	X	X	X	X	X	X	0
4	0	0	1	X	1	X	X	X	X	X	X	1
5	0	1	0	X	X	0	X	X	X	X	X	0
6	0	1	0	X	X	1	X	X	X	X	X	1
7	0	1	1	X	X	X	0	X	0	0	0	0
8	0	1	1	X	X	X	1	X	1	1	1	1
9	1	0	0	X	X	X	X	0	X	X	X	0
10	1	0	0	X	X	X	X	1	X	X	X	1
11	1	0	1	X	X	X	X	X	0	X	X	0
12	1	0	1	X	X	X	X	X	1	X	X	1
13	1	1	0	X	X	X	X	X	X	0	X	0
14	1	1	0	X	X	X	X	X	X	1	X	1
15	1	1	1	X	X	X	X	X	X	X	0	0
16	1	1	1	X	X	X	X	X	X	X	1	1

Cognitive Activity

Q1. You need an 8:1 MUX but only have 4:1 MUX and 2:1 MUX components available. Explain, step by step, how you would connect two 4:1 MUXes and one 2:1 MUX to build it, and state how many select lines are needed in total and what each one does. **(5)**

Ans: _____

_____

_____

_____

_____

Q2. A Full Adder is built from two Half Adders and one OR gate. Trace the circuit for inputs A=1, B=1, Cin=1: show the Sum and Carry produced by the first Half Adder, the Sum and Carry produced by the second Half Adder, and the final Sum and Cout of the Full Adder. **(5)**

Ans: _____

_____

_____

_____

_____

Obtained Marks	**/10**

Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation.

Student's Signature: _____ **Date: _____**

Psychomotor Activity 1 — Experimental Setup

Criteria	Performance Level	Description	Marks
Tool Setup (4)	Level 4	Opens and configures LogiSim and/or Venus correctly and independently.	4
	Level 3	Configures the required tool with minor errors or assistance.	3
	Level 2	Completes the setup with considerable assistance.	2
	Level 1	Setup is incomplete and requires substantial assistance.	1
File/Design Setup (4)	Level 4	Correctly creates/selects the required LogiSim circuit/design and/or Venus assembly file and required components.	4
	Level 3	File/design setup is mostly correct with minor errors or omissions.	3
	Level 2	Setup is partially completed and requires assistance.	2
	Level 1	Major errors or omissions are present in the setup.	1
Parameter Setup (2)	Level 2	Correctly defines required inputs, data, registers, components and initial conditions.	2
	Level 1	Defines most required inputs/components with minor errors.	1
Total Obtained			/10

If a student is absent:

Criteria	Description	Marks
Absent	Student is absent from the laboratory activity.	**0**

Instructor Name: _____ Sign: _____

Psychomotor Activity 2 — Implementation + Results

Criteria	Performance	Description	Marks
Implementation (4)	Level 4	Implements the required Full Adder and 8:1 MUX using 4:1 MUX correctly and independently in LogiSim.	4
	Level 3	Implements the required design with minor errors or assistance.	3
	Level 2	Partially implements the design and requires considerable assistance.	2
	Level 1	Implementation is largely incomplete and requires substantial assistance.	1
Execution/Simulation (2)	Level 2	Runs/simulates the LogiSim designs correctly and appropriately handles basic execution errors.	2
	Level 1	Runs/simulates the work with minor errors or assistance.	1
Output/Results	Level 4	Obtains correct results and clearly demonstrates the required circuit behavior/output for the tested inputs.	4
	Level 3	Obtains mostly correct results with minor errors in output or presentation.	3
	Level 2	Obtains partially correct results with noticeable errors or omissions.	2
	Level 1	Results are largely incomplete or require substantial correction.	1
Total Obtained			/10

If a student is absent:

Criteria	Description	Marks
Absent	Student is absent from the laboratory activity.	0

Affective Activity — Lab Report + Viva Voce

Criteria	Performance	Description	Marks
Lab Report Completeness & Organization	Level 4	Report is complete, well organized, clearly presents procedure, observations, measurements, and discussion.	4
	Level 3	Report is complete and organized with minor omissions.	3
	Level 2	Report is partially complete and requires improvement in organization/detail.	2
	Level 1	Report is incomplete or poorly organized.	1
Results & Conclusion	Level 2	Correctly presents observations/results and provides an appropriate conclusion.	2
	Level 1	Results or conclusion are incomplete or contain minor errors.	1
Viva Voce	Level 4	Clearly explains oscilloscope operation, function generator settings, waveform measurements, and experimental observations.	4
	Level 3	Explains most concepts correctly with minor prompting.	3
	Level 2	Demonstrates basic understanding but requires prompting.	2
	Level 1	Demonstrates limited understanding of the experiment.	1

The student has not submitted the report and has not appeared in the Viva Voce.

Criteria	Description	Marks
Lab Report and Results Not Submitted	Student has not submitted the report and results file.	0
Absent in Viva Voce	Student is absent from the viva voce.	0

Instructor's Signature: _____ **Date:** _____