# Lab 01 Introduction to Venus Simulator for RISC-V Assembly Programming

---

## Lab Objectives:

- Familiarize with the Venus RISC-V simulator interface and features.
- Understand the basics of RISC-V assembly language syntax and instructions.
- Test and debug RISC-V assembly code in Venus.

---

## Tools and Setup

**Tools Required:**

1. A web browser with an internet connection.
2. Venus RISC-V Simulator (https://venus.cs61c.org/).

**Setup Instructions:**

1. Open your browser and navigate to Venus.
2. Familiarize yourself with the interface:
   - Editor Pane: For writing assembly code.
   - Console Pane: Displays program output and errors.
   - Registers and Memory: Monitors runtime values.
3. No installation is needed; Venus runs entirely online.

---

## RISC-V Assembly Basics

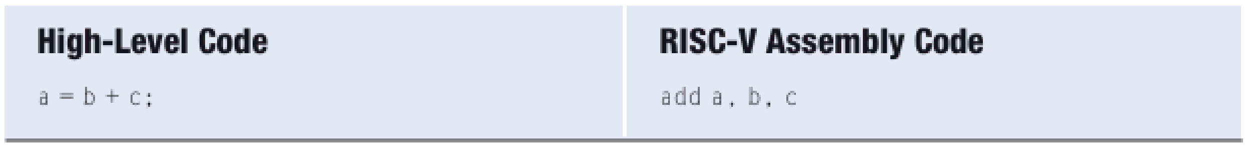
**Key Features of RISC-V:**

1. Simple and extensible architecture.
2. Fixed-length 32-bit instructions.
3. Registers (x0 to x31) are used for computation.

## Instruction Categories:

**1) Arithmetic Instructions (Add, Sub)**

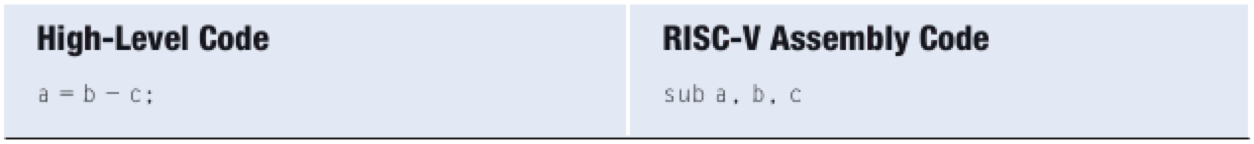
**Addition:**



| High-Level Code | RISC-V Assembly Code |
|---|---|
| `a = b + c;` | `add a, b, c` |

Example shows code for adding variables b and c and writing the result to a.

**Subtraction:**



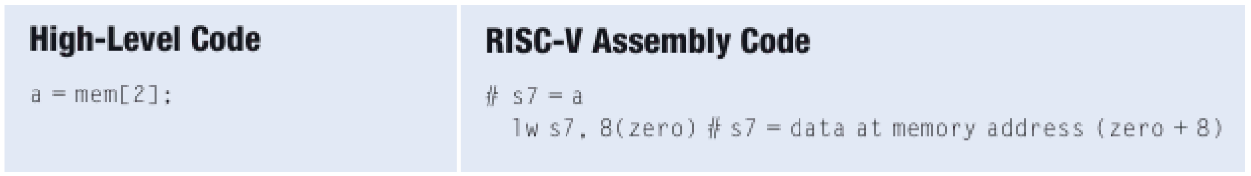
| High-Level Code | RISC-V Assembly Code |
|---|---|
| `a = b − c;` | `sub a, b, c` |

Example shows code for subtracting variables b and c and writing the result to a.

**2) Load and Store Instructions: (LW, SW)**

Lw (load word) loads a 32-bit value from memory into a register.
SW (store word) stores a 32-bit value from a register into memory.

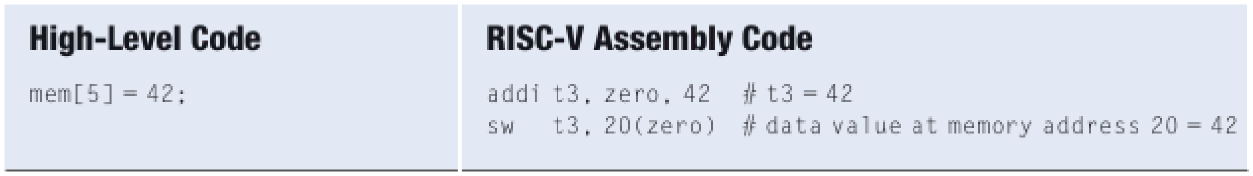
**Reading Memory (Load Instructions):**



| High-Level Code | RISC-V Assembly Code |
|---|---|
| `a = mem[2];` | `# s7 = a`<br>`  lw s7, 8(zero) # s7 = data at memory address (zero + 8)` |

The load word instruction, lw, reads a data word from memory into a register. The lw instruction specifies the memory address using an offset added to a base register. Here each data word is 4 bytes, so the word address is four times the word number. Word number 0 is at address 0, word 1 is at address 4, word 2 at address 8, and so on.

**Writing Memory (Store Instructions):**



| High-Level Code | RISC-V Assembly Code |
|---|---|
| `mem[5] = 42;` | `addi t3, zero, 42   # t3 = 42`<br>`sw   t3, 20(zero)  # data value at memory address 20 = 42` |

The store word instruction, sw, writes a data word from a register into memory. Example writes the value 42 from register t3 into memory word 5, located at address 20.

## Debugging and testing in venus

**Running a Program:**

- Write your code in the editor pane.
- Click the Assemble and Simulate button.

**Setting Breakpoints:**

- Click on the line numbers to pause execution at specific points.

**Viewing Registers and Memory:**

- Monitor values in the Register Panel during execution.

**Example Debugging Scenario:**

If your program does not produce the expected output, check for:

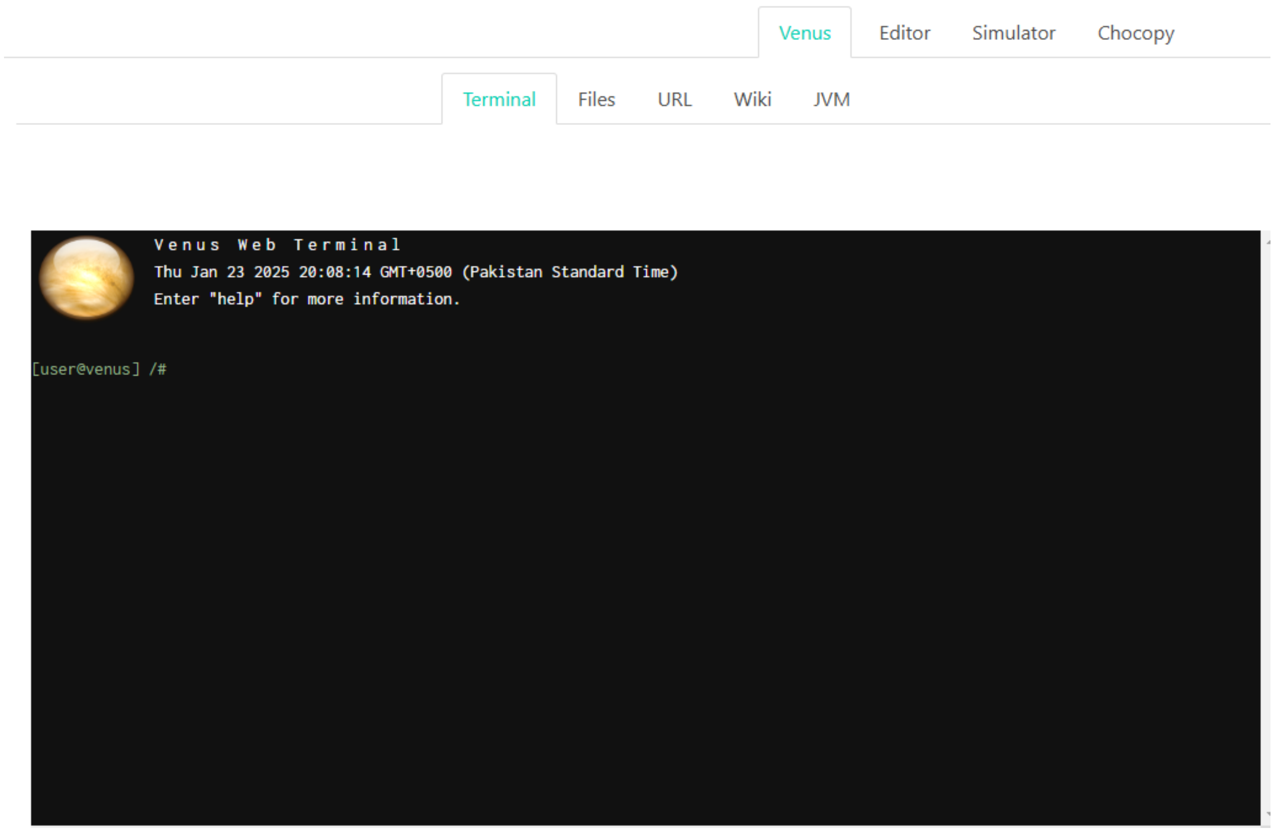
- Incorrect register usage.
- Missing or incorrect offset values in load/store instructions.

---

## Parts of Venus Simulator

1. **Venus:**
   Venus is the **core framework** that powers the simulator. It runs entirely in a web browser, eliminating the need for local installation. This makes it highly accessible and portable.

- **Key Features**:
  - Provides an assembler to translate RISC-V assembly code into machine code.
  - Simulates the execution of RISC-V programs.
  - Includes a user-friendly interface for debugging.



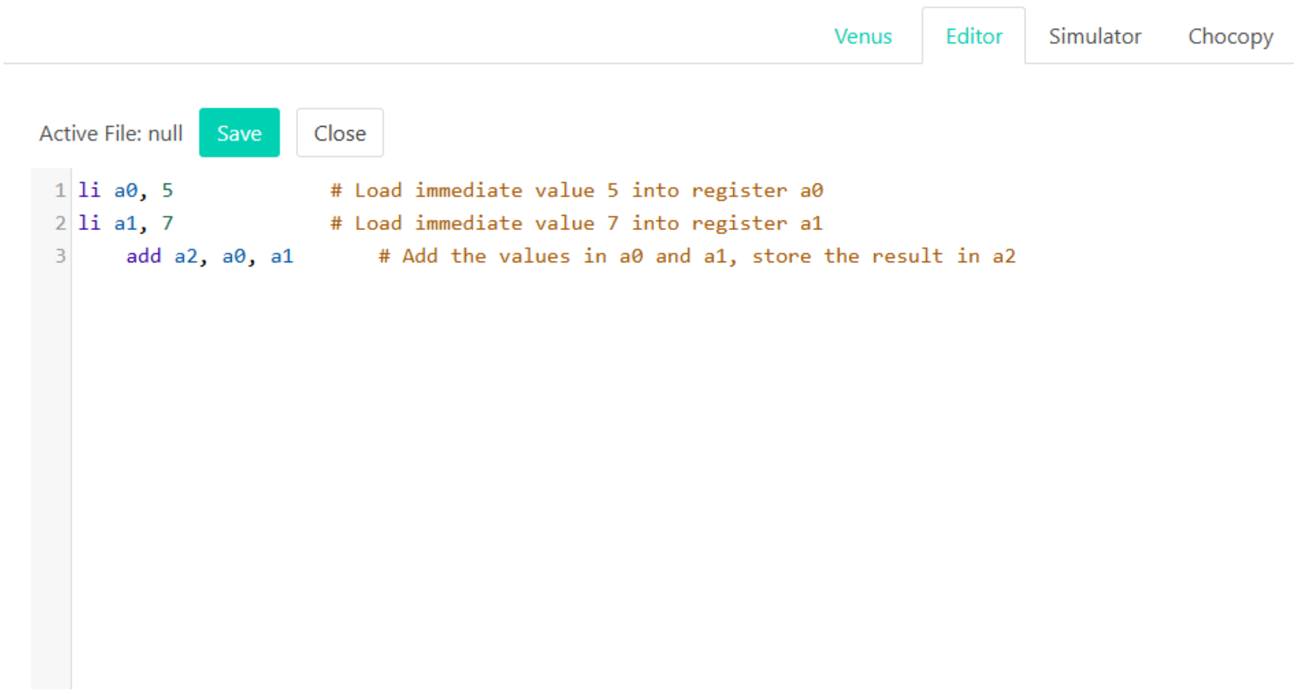
```
Venus Web Terminal
Thu Jan 23 2025 20:08:14 GMT+0500 (Pakistan Standard Time)
Enter "help" for more information.

[user@venus] /#
```

## Editor:

The Editor is the area within Venus where you write your RISC-V assembly code. It's a simple text editor that supports syntax highlighting for RISC-V, which improves readability and reduces errors.

- **Key Features**:
  - Line numbers to help navigate the code.
  - Syntax highlighting for RISC-V assembly instructions.
  - Error messages displayed when the code fails to assemble.

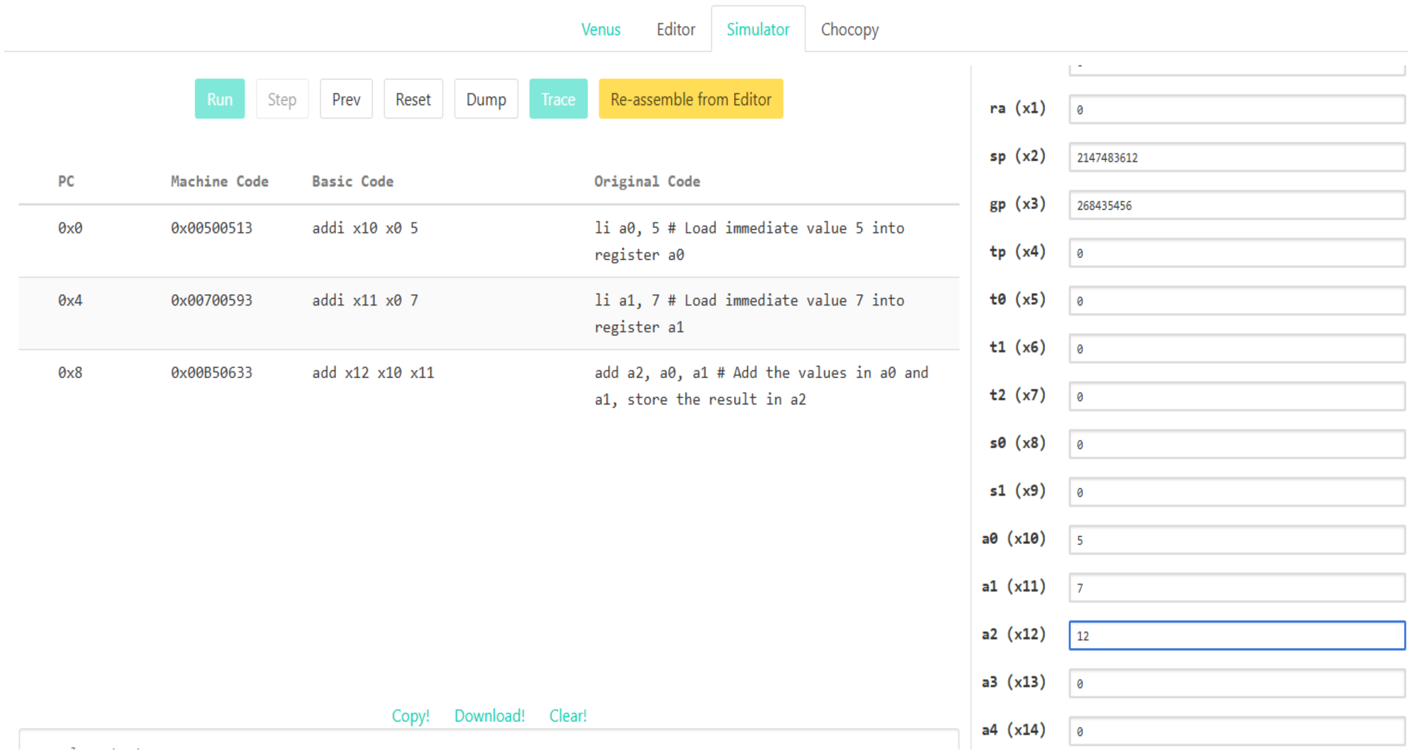


```
li a0, 5              # Load immediate value 5 into register a0
li a1, 7              # Load immediate value 7 into register a1
    add a2, a0, a1        # Add the values in a0 and a1, store the result in a2
```

## Simulator:

The Simulator executes the assembled RISC-V program and allows you to view the state of the machine during execution. After assembling your code, use the simulator to execute and debug it. Inspect the register values and memory to verify the behavior of your code.

- **Key Features**:
  - **Registers View**: Shows the values of general-purpose and special-purpose registers (e.g., a0, sp, ra).
  - **Memory View**: Allows you to inspect memory addresses and the stored data.
  - **Execution Control**: Offers buttons for step-by-step execution, running to completion, or resetting the simulation.
  - **Debugging Tools**: Includes breakpoints and the ability to monitor instruction execution.



| PC | Machine Code | Basic Code | Original Code |
|---|---|---|---|
| 0x0 | 0x00500513 | addi x10 x0 5 | li a0, 5 # Load immediate value 5 into register a0 |
| 0x4 | 0x00700593 | addi x11 x0 7 | li a1, 7 # Load immediate value 7 into register a1 |
| 0x8 | 0x00B50633 | add x12 x10 x11 | add a2, a0, a1 # Add the values in a0 and a1, store the result in a2 |

| | |
|---|---|
| ra (x1) | 0 |
| sp (x2) | 2147483612 |
| gp (x3) | 268435456 |
| tp (x4) | 0 |
| t0 (x5) | 0 |
| t1 (x6) | 0 |
| t2 (x7) | 0 |
| s0 (x8) | 0 |
| s1 (x9) | 0 |
| a0 (x10) | 5 |
| a1 (x11) | 7 |
| a2 (x12) | 12 |
| a3 (x13) | 0 |
| a4 (x14) | 0 |

## Lab Procedure:

1. **Open Venus**:
   Navigate to the Venus Simulator using your browser. No installation is required.
2. **Write Code in the Editor**:
   Write a simple RISC-V program (e.g., adding two numbers). Save your code.
3. **Assemble and Simulate**:
   Click on **"Assemble"** to convert your code into machine instructions. Use the **Simulator** to execute and debug the program.

## Lab Tasks:

**Task 1:**

Write and simulate RISC-V code to calculate the sum of three integers.

**Task 2:**

Translate the following high-level code into RISC-V assembly language. Assume variables $a - c\ are$ held in registers $\mathrm{s0 - s2}$, and $\mathrm{f - j}$ are in $\mathrm{s3 - s7}$.

1) $a = b - c;$ 2) $f = (g + h) - (i + j);$

**Task 3:**

Write an assembly program to multiply 3 with-17 and save the result in t3.

**Task 4:**

Write a RISC-V assembly program to divide-50 by 3. Store the quotient in t2 and remainder in t3.