### **Lab 12:** **WebAssembly in Rust**

**Exercise: WebAssembly Interactivity**

In this exercise, we will create a Rust program that performs a basic mathematical operation (e.g., addition), compile it to WebAssembly, and create a web page to interact with the WebAssembly module.

**Install the required tools:**

Make sure We have Rust and wasm-pack installed on Wer system. If We don't have wasm-pack installed, We can install it by following the instructions at https://rustwasm.github.io/wasm-pack/installer/.

1. Create a new Rust project using cargo:
2. Open Wer terminal/command prompt and run the following command:

cargo new wasm\_interactivity

cd wasm\_interactivity

Open the Cargo.toml file and add the following dependencies:

[lib]

crate-type = ["cdylib"]

[dependencies]

wasm-bindgen = "0.2.75"

Open the src/lib.rs file and implement a function to perform the mathematical operation. For example, we'll create a function to add two numbers:

use wasm\_bindgen::prelude::\*;

#[wasm\_bindgen]

pub fn add(a: i32, b: i32) -> i32 {

a + b

}

Build the WebAssembly module:

Run the following command to build the WebAssembly module:

wasm-pack build --target web

After building the WebAssembly module, We'll find a pkg directory in Wer project. Inside the pkg directory, there will be a wasm\_interactivity.js file and a wasm\_interactivity\_bg.wasm file. These are the JavaScript glue code and the compiled WebAssembly binary, respectively.

Create a new HTML file named index.html in the root directory of Wer project:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>WebAssembly Interactivity</title>

</head>

<body>

<h1>WebAssembly Interactivity</h1>

<p>Enter two numbers to add:</p>

<input type="number" id="num1">

<input type="number" id="num2">

<button onclick="addNumbers()">Add</button>

<p id="result">Result: </p>

<script>

async function init() {

// Import the WebAssembly module

const { add } = await import('./pkg/wasm\_interactivity');

// Function to add two numbers and display the result

window.addNumbers = function() {

const num1 = parseInt(document.getElementById('num1').value);

const num2 = parseInt(document.getElementById('num2').value);

const result = add(num1, num2);

document.getElementById('result').textContent = `Result: ${result}`;

};

}

init();

</script>

</body>

</html>

1. Save the file and return to the terminal/command prompt.
2. Start a local web server:

We can use any local web server of our choice. For example, We can use http-server by installing it globally:

npm install -g http-server

Then, navigate to the root directory of Wer project and run the following command:

http-server

Open web browser and visit the URL provided by the local web server (e.g., http://localhost:8080). We should see a web page with two input fields and a button.

Enter two numbers in the input fields and click the "Add" button. The result of the addition operation will be displayed on the page.

Congratulations! We have successfully completed the lab exercise on WebAssembly in Rust programming. We learned how to create a simple Rust program, compile it to WebAssembly, and interact with it using JavaScript in a web page. WebAssembly opens up exciting possibilities for running performant Rust code directly in web browsers.

**Happy coding!**