Contributing to the RISC-V CPU Project

> **Contribution & Project Diary Notice**
>
> Following this workflow is essential. All commits are logged and will be used to create a detailed contribution map for the final project diary. This ensures a clear and accurate progress report of the entire project.
>
> - **Contribution Record:** All work must be submitted via pull requests. This creates a permanent record of who contributed what and when.
> - **Automated Project Diary:** The Git log serves as our official project diary. Your commit messages will directly inform the final progress report.

1. Setting Up Your Development Environment

This project uses Nix to provide a consistent development environment with all the necessary tools.

Install Nix:

First, install Nix on your system (Linux, macOS, or WSL on Windows) using the Determinate Systems installer, which is the recommended method:

```
curl -fsSL https://install.determinate.systems/nix | sh -s -- install --determinate
```

2. Getting the Code

This project uses the fork-and-pull-request workflow.

1. **Fork the Repository:**

   Click the "Fork" button at the top right of the main repository page: https://github.com/ethycS0/risc_v_cpu

2. **Clone Your Fork:**

   Clone your forked repository to your local machine. Replace `your_github_username` with your actual username.

   ```
   git clone git@github.com:your_github_username/risc_v_cpu.git
   cd risc_v_cpu # This is your Project Root
   ```

3. The Development Workflow

All development work should be done within the Nix development shell.

1. **Activate the Environment:**

From the project's root directory, run the following command. You must do this every time you open a new terminal to work on the project.

```
nix develop
```

This command will make all required tools (`GHDL`, `GTKWave`, etc.) available in your shell.

2. **Implement Your Code:**
   - Place new module implementations (e.g., `module.vhd`) in the `src/` directory.
   - Place corresponding testbenches (e.g., `tb_module.vhd`) in the `tb/` directory.
3. **Build and Test:**

   Use the provided `Makefile` to simulate your design.

   - **Run a simulation:**

     ```
     make run TB=tb_module
     ```

     (Replace `tb_module` with the name of your testbench file, without the `.vhd` extension).
   - **View waveforms:**

     After running a simulation, you can view the resulting waveforms with GTKWave:

     ```
     make view TB=tb_module
     ```

4. Submitting Your Contribution

Once your implementation is working correctly and fully tested, please follow these steps to submit a pull request.

1. **Clean the Project:**

   Always run the clean command before committing to remove generated simulation files.

   ```
   make clean
   ```

2. **Commit and Push Your Changes:**

   Stage, commit, and push your changes to your forked repository. It is a best practice to create a new branch for your feature.

   ```
   git add .
   git commit -m "feat: Implement the ALU and its testbench" # Change Message to Implementati
   on Details
   git push origin main
   ```

3. **Create a Pull Request:**

- Go to your forked repository on GitHub (`https://github.com/your_github_username/risc_v_cpu`).
- You should see a prompt to "Contribute" and "Open a pull request." Click it.
- Provide a clear title and a detailed description of your changes in the pull request.
- I will review the pull request and may request changes. If so, simply make the required changes, commit, and push them to your branch again. The pull request will update automatically.