**Project Setup: with VS Code and MinGW (**[**source**](https://code.visualstudio.com/docs/cpp/config-mingw)**)**

**1. Get VS Code Ready:**

* **1.1 Install VS Code:**
  + Download and install Visual Studio Code from the official website ([code.visualstudio.com](https://westvirginiauniversity-my.sharepoint.com/personal/ieelwarfalli_mail_wvu_edu/Documents/code.visualstudio.com) ). This is your code editor.
* **1.2 Add C/C++ Tools:**
  + Open VS Code.
  + Go to the Extensions view (the square icon on the left sidebar).
  + Search for "C/C++" and install the Microsoft C/C++ extension. This adds essential C/C++ support to VS Code.

**2. Set Up Your Compiler (MinGW):**

* **2.1 Install MinGW:**
  + Download and install MSYS2 from [here](https://github.com/msys2/msys2-installer/releases/download/2024-12-08/msys2-x86_64-20241208.exe). Follow the installation instructions carefully.
  + Open the "MSYS2 UCRT 64-bit" terminal.
  + Run this command: pacman -S --needed base-devel mingw-w64-ucrt-x86\_64-toolchain
  + This installs the necessary tools to compile your C/C++ code.
* **2.2 Verify Installation:**
  + Open a regular Command Prompt (type "cmd" in the Windows search bar).
  + Type **gcc --version** and press Enter.
  + If you see version information, MinGW is installed correctly. If you get an error, double check the installation steps from 2.1.
* **2.3 Configure the PATH:**
  + If **gcc --version** returns an error, you need to add the MinGW bin folder to your system's PATH.
  + Find the MinGW installation directory (usually C:\msys64\ucrt64\bin).
  + Search for "environment variables" in the Windows search bar and open "Edit the system environment variables".
  + Click "Environment Variables...".
  + In the "System variables" section, find and select the "Path" variable, then click "Edit...".
  + Click "New" and add the path to your MinGW bin folder
  + Click "OK" on all open windows.
  + Close and reopen the command prompt and verify **gcc --version** once again.

**3. Test Your Setup:**

* **3.1 Create a Test File:**
  + Open VS Code. Create a new file named ***hello\_world.c***.
  + Copy and paste the following code:

#include <stdio.h>

int main() {

printf("Hello, world!\n");

return 0;

}

* **3.2 Compile and Run:**
  + Open a new terminal in VS Code (Terminal > New Terminal).
  + Type **gcc hello\_world.c -o hello\_world.exe** and press Enter. This compiles your code.
  + Type **.\hello\_world**.exe and press enter. If you see "Hello, world!", your setup is working

**4. Get Project Code:**

* **4.1 Download from eCampus:**
  + **Download** the project code files from your eCampus course page.
  + **Unzip** the files to a folder on your computer.

**5. Start Coding**

* Open the project folder in VS Code and begin working on your project.

**6.**  **Steps to Compile**

1. **Ensure MinGW is Installed and Configured:**
   * As previously outlined, ensure you have MSYS2 with the MinGW-w64 toolchain installed and that the **bin** directory is added to your system's **PATH** environment variable. Verify by running **gcc --version** in your command prompt.
2. **Open VS Code and Navigate to the Project Folder:**
   * Open VS Code.
   * Go to "File" > "Open Folder..." and select the folder containing your .c files.
3. **Open a Terminal in VS Code:**
   * Go to "Terminal" > "New Terminal".
4. **Compile the Project:**
   * In the terminal, use the following **gcc** command:

***gcc*** *ADD.c ADDI.c AND.c ANDI.c BEQ.c BNE.c DIV.c LUI.c LW.c MFHI.c MFLO.c MIPS\_Instruction.c MIPS\_Interpreter.c MULT.c OR.c ORI.c SLT.c SLTI.c SUB.c SW.c -o MIPS\_Interpreter.exe*

1. **Run the Ex*ecutable*:**

* After successful compilation, you can run the program by typing:  
  .\MIPS\_Interpreter.exe

**Notes:**

* **Errors:**
  + If you encounter compilation errors, carefully examine the error messages. They will usually indicate the file and line number where the problem occurred.
  + Make sure there are no typos in the file names in the compile command.
* **Makefiles:**
  + For larger projects, using a Makefile is highly recommended. A Makefile automates the compilation process, making it easier to manage dependencies and rebuild only the necessary files. This is a more advanced topic.
* **VS Code Tasks:**
  + VS Code also supports "tasks," which allow you to define custom build commands. You can create a task to run the **gcc** command, which can simplify the compilation process.