Debugging of Simulators

We will download the corresponding version of the simulator and unzip the files in the directory that we have mapped in Ubuntu.

Once we have the simulator code in the directory, we copy the ".vscode" folder in the same directory:

.vscode	10/02/2021 13:56	Carpeta de archivos	
Buses.c	15/02/2019 14:24	Archivo C	2 KB
Buses.h	15/02/2019 13:52	Archivo H	1 KB
Buses.o	22/02/2021 15:39	Archivo O	7 KB
ComputerSystem.c	21/01/2020 13:40	Archivo C	6 KB
ComputerSystem.h	15/02/2019 14:07	Archivo H	2 KB
ComputerSystem.o	22/02/2021 15:39	Archivo O	16 KB
Instructions.def	21/01/2020 13:02	Archivo DEF	1 KB
MainMemory.c	19/02/2019 12:03	Archivo C	2 KB
MainMemory.h	15/02/2019 14:26	Archivo H	1 KB
MainMemory.o	22/02/2021 15:39	Archivo O	6 KB
Makefile	16/02/2021 17:57	Archivo	2 KB
Messages.c	12/05/2017 16:26	Archivo C	2 KB
Messages.h	30/01/2020 14:54	Archivo H	1 KB
Messages.o	22/02/2021 15:39	Archivo O	9 KB
messages.txt	22/02/2021 16:06	Documento de tex...	1 KB
OperatingSystem.c	21/01/2020 13:31	Archivo C	3 KB
OperatingSystem.h	12/05/2017 16:26	Archivo H	1 KB
OperatingSystem.o	22/02/2021 15:39	Archivo O	9 KB
Processor.c	31/01/2020 12:22	Archivo C	8 KB
Processor.h	21/01/2020 13:11	Archivo H	1 KB
Processor.o	22/02/2021 15:39	Archivo O	18 KB
programToBeExecuted	21/01/2020 14:29	Archivo	1 KB
Simulator	22/02/2021 15:39	Archivo	40 KB
Simulator.c	17/01/2020 14:51	Archivo C	1 KB
Simulator.h	12/05/2017 16:26	Archivo H	1 KB
Simulator.o	22/02/2021 15:39	Archivo O	7 KB

Figure 1 Folder with .vscode configuration and the simulator files

We enter the Ubuntu bash (where we will have a symbolic link to the desktop):

Figure 2 Symbolic link created in the first session

We will enter the directory where we have the simulator code, that is:

cd Desktop/sesionX/

After moving to the directory where we have the simulator, we will execute the command code . on the directory itself, thus creating a VS Code WSL server on which we will work.

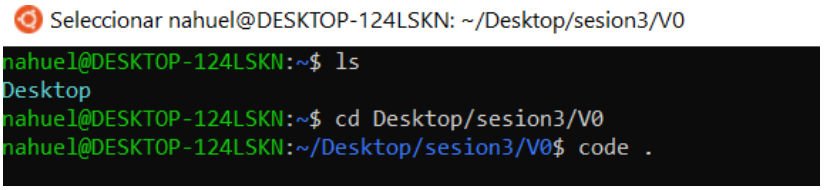

Figure 3 Creating the WSL server of VS Code

A window with the VS Code editor will open, in which we will be able to work and debug the code we are executing:

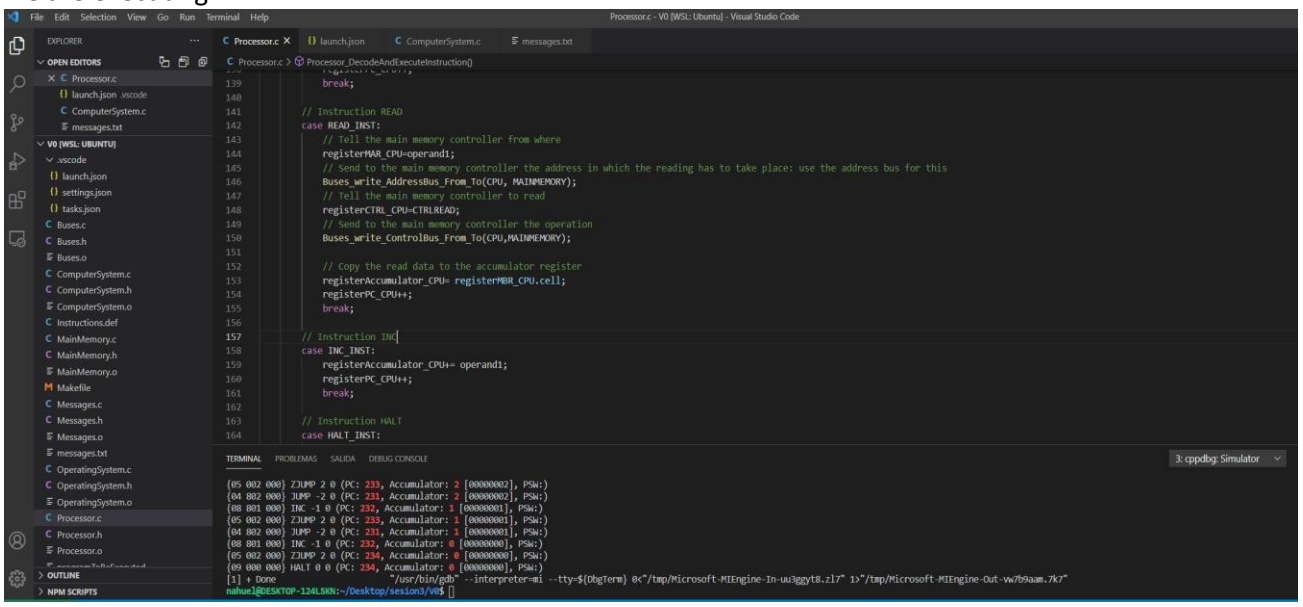

Figure 4 VS Code editor

Before doing any other steps, you must go to the extensions tab and install the c debugger and compiler and the WSL extension.

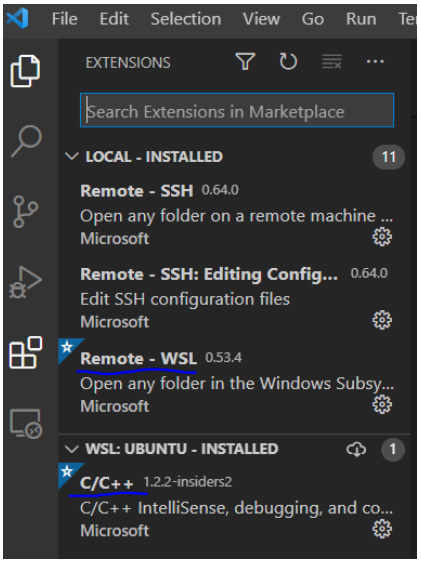

Figure 5 Installing the necessary addons

Once this is installed we can debug the program and/or run it.

Debugging en Visual Studio Code

Visual Studio Code allows you to specify a series of steps to be performed before-after running the debugging of the code. These can be found in the files contained in the folder .vscode: launch.json y tasks.json. The first file (launch.json) specifies the program, compiler and arguments.

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Launch Simulator",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/Simulator",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}",
            "environment": [],
            "MIMode": "gdb",
```

Figure 6 Some of the launch.json code

The tasks.json file contains a series of steps that, in this case are simply an execution of the makefile (make clean and make). This step is executed prior to the start of the program debugging.

Debugging is similar to any other IDE, you can put breakpoints in the code and run it step by step to detect where the potential bug may be.

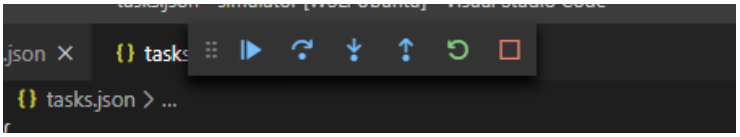

Figure 7 Debugging with VS Code