Software

This directory contains source files of sample codes and user codes which will be executed on the phoeniX processor. In this directory, there are three subdirectories included:

- `Sample_Assembly_Codes`
- `Sample_C_Codes`
- `User_Codes`

The code execution and simulation on the phoeniX RISC-V processor follow two distinct branches: one for Linux systems and another for Windows systems.

Linux

Running Sample Codes

The directory `/Software` contains sample codes for some conventional programs and algorithms in both Assembly and C which can be found in `/Sample_Assembly_Codes` and `/Sample_C_Codes` sub-directories respectively.

phoeniX convention for naming projects is as follows; The main source file of the project is named as `{project.c}` or `{project.s}`. This file along other required source files are kept in one directory which has the same name as the project itself, i.e. `/project`.

Sample projects provided at this time are `bubble_sort`, `fibonacci`, `find_max_array`, `sum1ton`. To run any of these sample projects simply run `make sample` followed by the name of the project passed as a variable named project to the Makefile.

```
make sample project={project}
```

For example:

```
make sample project=fibonacci
```

Provided that the RISC-V toolchain is set up correctly, the Makefile will compile the source codes separately, then using the linker script `riscv.ld` provided in `/Firmware` it links all the object files necessary together and creates `firmware.elf`. It then creates `start.elf` which is built from `start.s` and `start.ld` and concatenate these together and finally forms the `{project}_firmware.hex`. This final file can be directly fed to our verilog testbench. Makefile automatically runs the testbench and calls upon `gtkwave` to display the selected signals in the waveform viewer.

Running Your Own Code

In order to run your own code on phoeniX, create a directory named to your project such as `/my_project` in `/Software/User_Codes/`. Put all your `.c` and `.s` files in `/my_project` and run the following `make` command from the main directory:

```
make code project=my_project
```

Provided that you name your project sub-directory correctly and the RISC-V Toolchain is configured without any troubles on your machine, the Makefile will compile all your source files separately, then using the linker script `riscv.ld` provided in `/Firmware` it links all the object files necessary together and creates `firmware.elf`. It then creates `start.elf` which is built from `start.s` and `start.ld` and concatenate these together and finally forms the `my_project_firmware.hex`. After that, `iverilog` and `gtkwave` are used to compile the design and view the selected waveforms.

> Further Configurations : The default testbench provided as `phoeniX_Testbench.v` is currently set to support up to 4MBytes of memory and the stack pointer register `sp` is configured accordingly. If you wish to change this, you need configure both the testbench and the initial value the `sp` is set to in `/Firmware/start.s`. If you wish to use other specific libraries and header files not provided in `/Firmware` please beware you may need to change linker scripts `riscv.ld` and `start.ld`.

Windows

Running Sample Codes

We have meticulously developed a lightweight and user-friendly software solution with the help of Python. Our execution assistant software, `AssembleX`, has been crafted to cater to the specific needs of Windows systems, enabling seamless execution of assembly code on the phoeniX processor.

This tool enhances the efficiency of the code execution process, offering a streamlined experience for users seeking to enter the realm of assembly programming on pheoniX processor in a very simple and user-friendly way.

Before running the script, note that the assembly output of the Venus Simulator for the code must be also saved in the project directory. To run any of these sample projects simply run python `AssembleX_V1.0.py sample` followed by the name of the project passed as a variable named project to the Python script. The input command format for the terminal follows the structure illustrated below:

```
python AssembleX_V1.0.py sample {project_name}
```

For example:

```
python AssembleX_V1.0.py sample fibonacci
```

After execution of this script, firmware file will be generated and this final file can be directly fed to our Verilog testbench. AssembleX automatically runs the testbench and calls upon gtkwave to display the selected signals in the waveform viewer application, gtkwave.

Running Your Own Code

In order to run your own code on phoeniX, create a directory named to your project such as `/my_project in /Software/User_Codes/`. Put all your ``user_code.s` files in my_project and run the following command from the main directory:

```
python AssembleX_V1.0.py code my_project
```

Provided that you name your project sub-directory correctly the AssembleX software will create `my_project_firmware.hex` and fed it directly to the testbench of phoeniX processor. After that, iverilog and GTKWave are used to compile the design and view the selected waveforms.