ACA Project Submission – Readme File

The project was done in Python, and the final file is named as *"simulator.py"*. No additional external libraries were utilized for this project (the inbuilt *sys* library was included to be able to parse the command-line arguments).

The following lists 2 possible methods to execute the simulation program.

Method 1 – Utilizing the Makefile:

The program can also be executed by using the Makefile. The Makefile defines 5 options, as follows:

- *make run_default*
    - This option will execute the simulation program using the default instruction set files which were included as part of the project document. These input files are located in the *default* folder.
- *make run_test1*
    - This option will execute the simulation program using the Test Case 1 files. These input files are located in the *test_case_1* folder.
- *make run_test2*
    - This option will execute the simulation program using the Test Case 2 files. These input files are located in the *test_case_2* folder.
- *make run_test3*
    - This option will execute the simulation program using the Test Case 3 files. These input files are located in the *test_case_3* folder.
- *make clean*
    - This option will delete the *result.txt* file, cleaning the directory and leaving only the necessary files for executing the program.

Executing the simulation program through the Makefile will create a *result.txt* file with the output in the same directory as the *simulation.py* program.

Method 2 – Directly from the Command-line with custom configuration and data files:

The program can be run directly from the command-line, by passing in the 4 necessary arguments as follows in the command-line:

python3 ./simulator.py <inst_file.txt> <data_file.txt> <config_file.txt> <result_file.txt>

The above command will use the first 3 arguments (*<inst_file.txt>, <data_file.txt> & <config_file.txt>*) to set up and run the scoreboard simulation, and then produce an output in the *<result_file.txt>* file defined in the 4th argument.

An example of the above is shown below:

```
[nishanr1@linux1 aca] python3 ./simulator.py inst.txt data.txt config.txt result.txt
```

Additionally, you can adjust the verbosity of the program in the terminal by providing an **optional** 5th argument. (By default, this value is set to 2).

- If the verbosity value is set to 0, the simulation program will display minimal output.
- If the verbosity value is set to 1, the simulation program will display the status of the scoreboard and its active instructions at each clock cycle.
- If the verbosity value is set to 2, the simulation program will display the status of the scoreboard, and also display the active instructions in the pipeline at each clock cycle.

Examples are shown below.

```
[nishanr1@linux1 aca] python3 ./simulator.py inst.txt data.txt config.txt result.txt 0
```

OR

```
[nishanr1@linux1 aca] python3 ./simulator.py inst.txt data.txt config.txt result.txt 1
```

OR

```
[nishanr1@linux1 aca] python3 ./simulator.py inst.txt data.txt config.txt result.txt 2
```