Performance Modeling - RISC-V processor

This will be an INDIVIDUAL project

Phase 2: (Due December 15th 11:59PM)

1) Draw the schematic for a five stage processor and fill in your code in the provided file to run the simulator.

2) Measure and report average CPI, Total execution cycles, and Instructions per cycle by adding performance monitors to your code. Make sure you output these values to a file.

3) What optimizations or features can be added to improve performance? (Extra credit)

4) Compare the results from both the single stage and the five stage pipelined processor implementations and explain why one is better than the other.

Notes:

1. Please watch the video uploaded to brightspace. It may answer

2. Use the same folder format as mentioned in Phase 1.

3. StateResult file for FiveStage will NOT be used for grading.

4. The register file should output the register values for how many ever cycles your code runs. (Note that this will be higher than your single stage results)

5. Go through the branch instruction's explanations in the PDF. The outputs for the small testcases provided for branch instructions aren't accurate and you should be getting a couple more cycles in your output.

6. Your report will have the same format as you did in Phase 1. But this time, you have to answer the additional question of comparison between both pipelines.

Phase 1: (Due November 7th 11:59PM)

1) Draw the schematic for a single stage processor and fill in your code in the provided file to run the simulator.

2) Measure and report average CPI, Total execution cycles, and Instructions per cycle by adding performance monitors to your code. Make sure you output these values to a file.

3) What optimizations or features can be added to improve performance? (Extra credit)

Your code will be tested against 10 test cases. 3 of which will be released 5 days prior to your submission. (2nd November)

Notes:

1. **Please modify the existing code to use the correct way to handle folder paths. (Use `os.path.join()` instead of hardcoding OS dependent forward/back slashes).**

2. You'll only submit a zipped folder named `netID.zip`

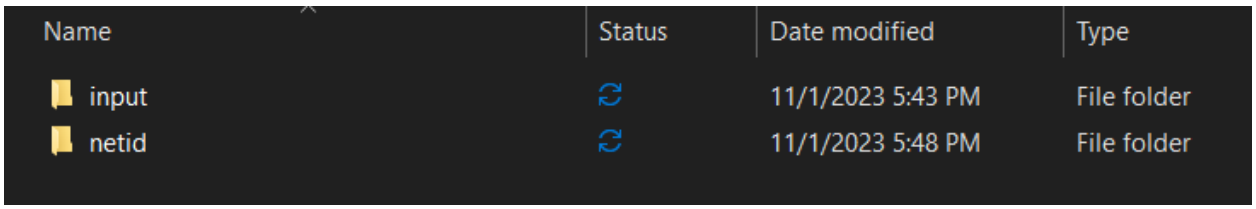
   Make sure you follow the folder structure shown below.
   Your code will be in one folder named `netID` with an entry point file (main.py or main.cpp).
   There'll be a second folder called `input/` with each subfolder named `testcase0/`, `testcase1`, `etc.`, and each of these test cases will contain 2 files, `imem.txt` and `dmem.txt`.
   After running your code, a third folder should be created as an output folder with the name `output_netID/` with subfolders named `testcase0/`, `testcase1`. Each subfolder must contain 4 files: `PerformanceMetrics_Result.txt`, `SS_DMEMResult.txt`, `SS_RFResult.txt`, `StateResult_SS.txt`.

3. A sample test case is already on brightspace under the project section.

```
Project root
|
|_ input/
|      |_testcase0
|      |      |_imem.txt
|      |      |_dmem.txt
|      |_testcase1
|      |      |_imem.txt
|      |      |_dmem.txt
|      |_testcasen
|             |_imem.txt
|             |_dmem.txt
|
|_ netID/
|      |_main.py (or) main.cpp
|      |_additional_folders_and_files
|      |_README (in case there's anything you want to mention about your code/dependencies)
|
|_ output_netID/
       |_testcase0
       |      |_PerformanceMetrics_Result.txt
       |      |_SS_DMEMResult.txt
       |      |_SS_RFResult.txt
       |      |_StateResult_SS.txt
       |
```

```
|_testcasen
      |_PerformanceMetrics_Result.txt
      |_SS_DMEMResult.txt
      |_SS_RFResult.txt
      |_StateResult_SS.txt
```

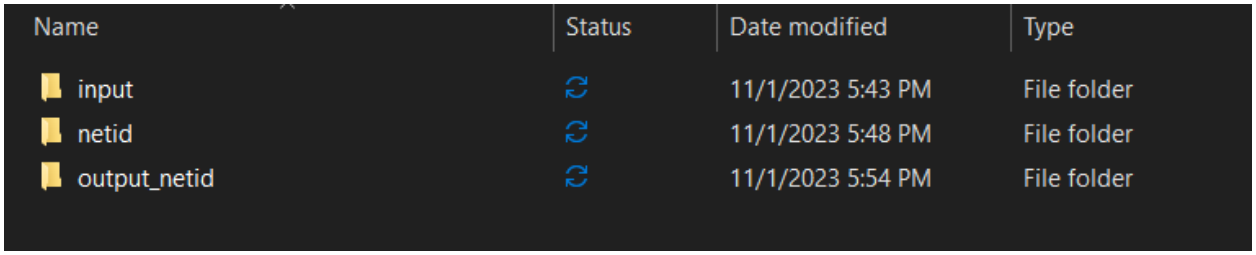
Solve with your folder structure as:



After running

```
$ python3 netid/main.py
```

The expected output folder structure should be as follows:



You will be submitting the zipped netid folder, and the PDF report along with it on brightspace. Do not upload the input/output folders.

Follow the above instructions and folder structure as is.
Your submission on brightspace will require 2 items.

1. **The zipped folder named netid.zip (eg. pr2359.zip)**
2. **The Project report PDF**

The PDF MUST be present in the brightspace submission just like any other HW you've submitted in the past.
Upon extracting the zipped folder, only ONE folder should be visible with the name *netid*
There should be a file with the name ***main.py* or ***main.cpp*** in this folder.**
Adding additional files is okay but ONE of the above two MUST be present.

If any additional modules/imports are required, specify them in a readme, and provide a requirements.txt.

netid is unique to you. My net id is pr2359, so my folders will be named pr2359/, output_pr2359/ etc.
Many submissions in Phase 1 submitted a folder with the name `netID`. There won't be a penalty in Phase 1 submissions, but there will be points deducted in Phase 2.

Your code will be run with one of the following commands:

```
$ python pr2359/main.py
```

```
$ g++ pr2359/main.cpp -o pr2359/main; ./pr2359/main
```

This will be the ONLY command that will be run.

After this, the output folder must be created and must have the outputs of ALL test cases. In case you've written the code to output only one test case, please add a for-loop and run it for all testcases. The number of testcases is easy to count based on the number of folders present inside the input/ folder.

Before submitting, please test it locally.
Make sure you don't have any output files or folders initially. Run one of the above commands (replace the netid with your own), and ensure the correct folder with the name with output_netid is present. (Only one parent output folder)
There should be the same number of output folders within output_netid as there are in input/.

Please keep in mind to use the os module to obtain a folder or file path. (Some parts of the existing code must also be changed)
Hardcoded forward and backslashes will cause errors.

Runtime errors that are caused by "No file/folder found" will be penalized.
It is better to have a wrong output than no output.