ECE-GY 6913
CSA Project Phase-1
Abdul Samad Zaheer Khan
ak9943@nyu.edu

Performance Modelling RISC-V Processor

The phase-1 of the project simulates Single-step RISC-V processor and stores the necessary performance metrics.

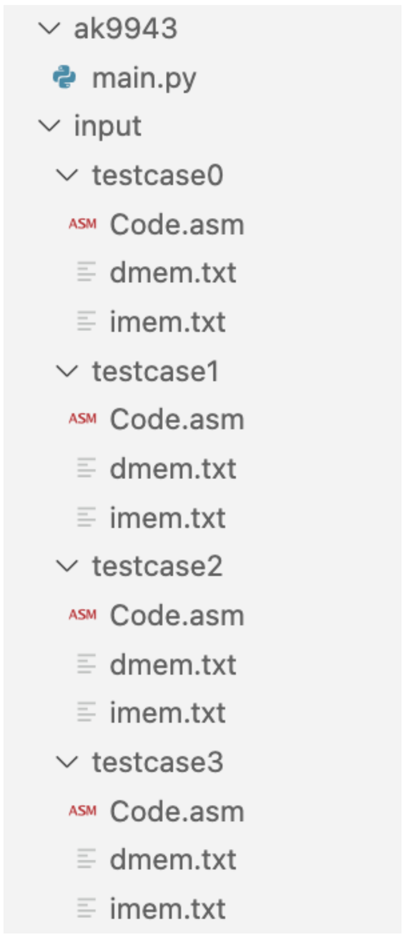
The following folder structure is used:-

```
ak9943_csa_project_phase_1
|
|__ak9943/
|       |
|       |__main.py
|
|__input/
|       |
|       |__testcase0/
|       |       |
|       |       |__dmem.txt
|       |       |__imem.txt
|       |
|       |__testcase1/
|       |       |
|       |       |__dmem.txt
|       |       |__imem.txt
|       |
|       |__testcase2/
|       |       |
|       |       |__dmem.txt
|       |       |__imem.txt
|       |
|       |__testcase3/
|               |
|               |__dmem.txt
|               |__imem.txt
|
|
|__output_ak9943/
        |
        |__PerformanceMetrics_Result.txt
        |__SS_DMEMResult.txt
        |__SS_RFResult.txt
        |__StateResult_SS.txt
```

The folder **output_ak9943** is created after running the command **python ak9943/main.py** in the **ak9943_csa_project_phase_1** folder.

The files imem.txt and dmem.txt are provided before runtime for every testcase.

The folder structure before running the command is:

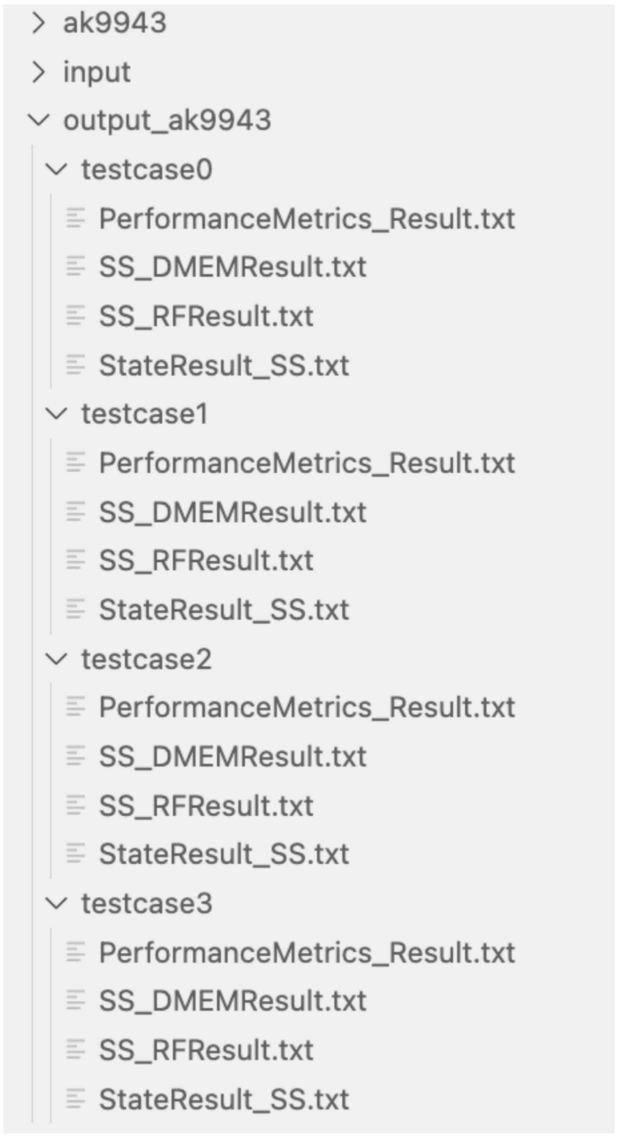


```
ak9943
  main.py
  input
    testcase0
      Code.asm
      dmem.txt
      imem.txt
    testcase1
      Code.asm
      dmem.txt
      imem.txt
    testcase2
      Code.asm
      dmem.txt
      imem.txt
    testcase3
      Code.asm
      dmem.txt
      imem.txt
```

Terminal output after successfully running the python command to simulate the RISC-V ISA is:

```
● abdulsamadkhan@Abduls-Air ak9943_csa_project_phase_1 % python ak9943/main.py
  IO Directory: /Users/abdulsamadkhan/Downloads/ak9943_csa_project_phase_1/input/testcase3
  IO Directory: /Users/abdulsamadkhan/Downloads/ak9943_csa_project_phase_1/input/testcase2
  IO Directory: /Users/abdulsamadkhan/Downloads/ak9943_csa_project_phase_1/input/testcase0
  IO Directory: /Users/abdulsamadkhan/Downloads/ak9943_csa_project_phase_1/input/testcase1
```

Folder structure after running the python command **python ak9943/main.py** in the **ak9943_csa_project_phase_1** folder to generate performance metrics:

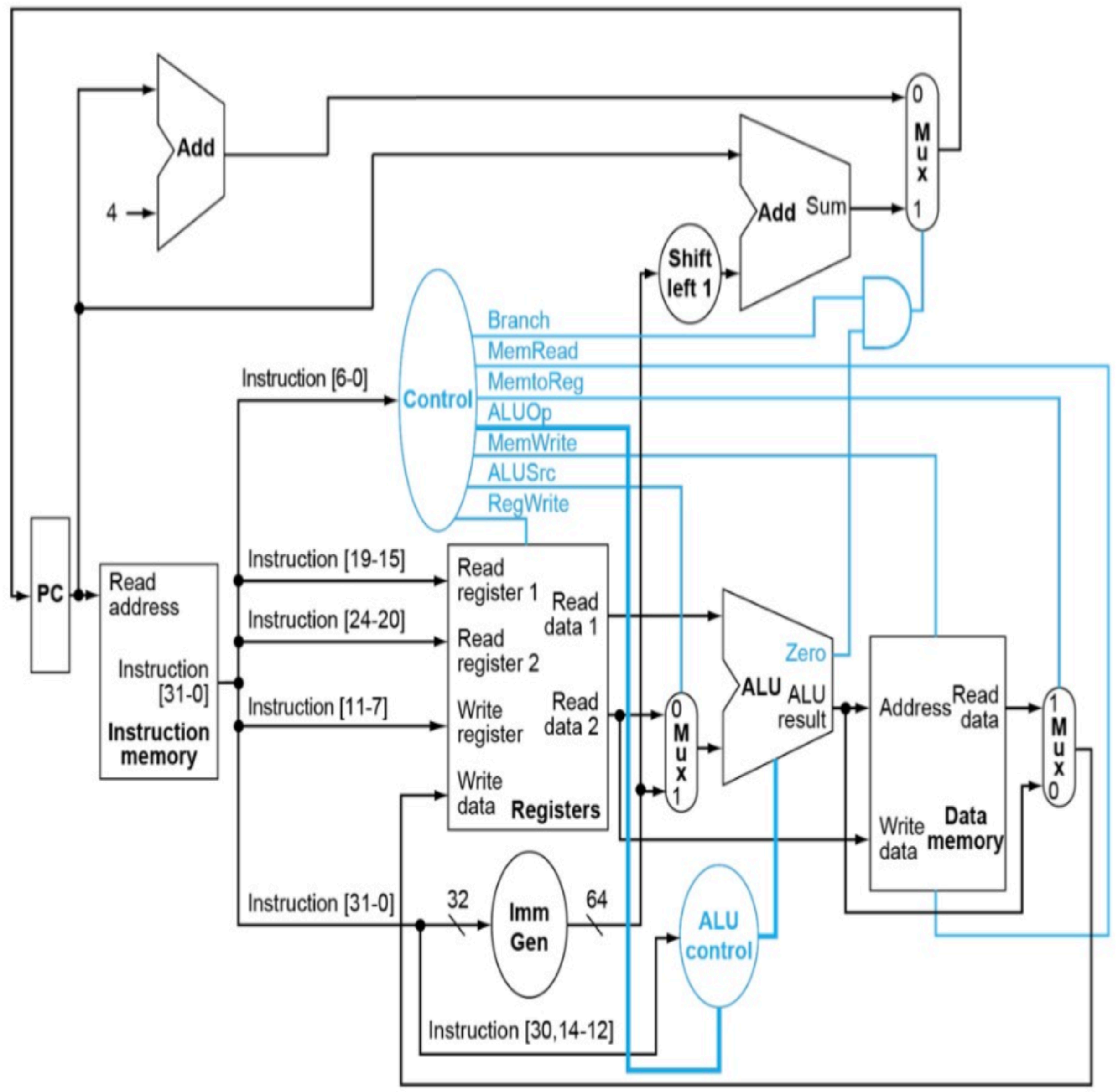


```
> ak9943
> input
v output_ak9943
  v testcase0
      PerformanceMetrics_Result.txt
      SS_DMEMResult.txt
      SS_RFResult.txt
      StateResult_SS.txt
  v testcase1
      PerformanceMetrics_Result.txt
      SS_DMEMResult.txt
      SS_RFResult.txt
      StateResult_SS.txt
  v testcase2
      PerformanceMetrics_Result.txt
      SS_DMEMResult.txt
      SS_RFResult.txt
      StateResult_SS.txt
  v testcase3
      PerformanceMetrics_Result.txt
      SS_DMEMResult.txt
      SS_RFResult.txt
      StateResult_SS.txt
```

Questions:

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

Solution:

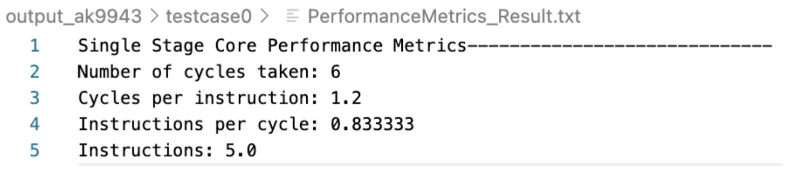


2. Measure and report average CPI, Total execution cycles, and Instructions per cycle by adding performance monitors to your code. Make sure you output these values to a file.

Solution:

Single Stage Performance Metrics

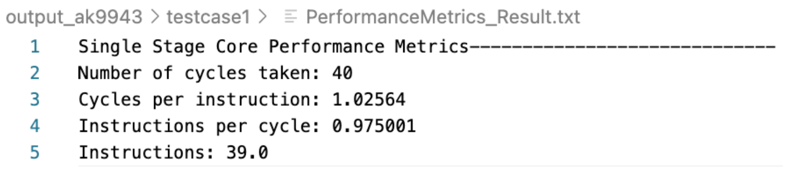
Testcase-0:
Number of cycles taken: 6
Cycles per instruction: 1.2
Instructions per cycle: 0.833333
Instructions: 5.0



output_ak9943 > testcase0 > PerformanceMetrics_Result.txt

```
Single Stage Core Performance Metrics-----------------------------
Number of cycles taken: 6
Cycles per instruction: 1.2
Instructions per cycle: 0.833333
Instructions: 5.0
```

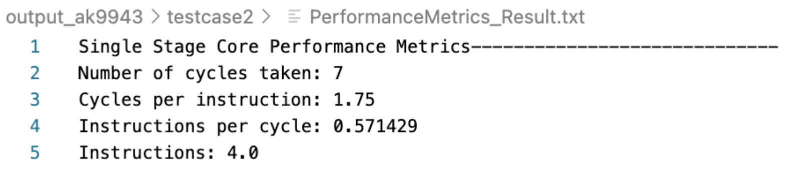
Testcase-1:
Number of cycles taken: 40
Cycles per instruction: 1.02564
Instructions per cycle: 0.975001
Instructions: 39.0



output_ak9943 > testcase1 > PerformanceMetrics_Result.txt

```
Single Stage Core Performance Metrics-----------------------------
Number of cycles taken: 40
Cycles per instruction: 1.02564
Instructions per cycle: 0.975001
Instructions: 39.0
```

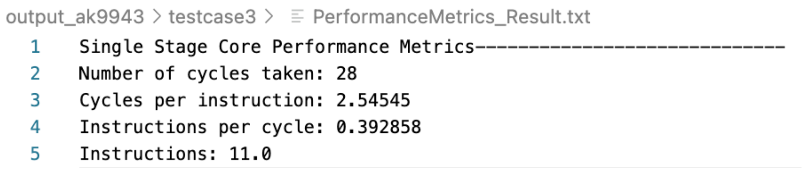
Testcase-2:
Number of cycles taken: 7
Cycles per instruction: 1.75
Instructions per cycle: 0.571429
Instructions: 4.0



output_ak9943 > testcase2 > PerformanceMetrics_Result.txt

```
Single Stage Core Performance Metrics-----------------------------
Number of cycles taken: 7
Cycles per instruction: 1.75
Instructions per cycle: 0.571429
Instructions: 4.0
```

Testcase-3:
Number of cycles taken: 28
Cycles per instruction: 2.54545
Instructions per cycle: 0.392858
Instructions: 11.0



output_ak9943 > testcase3 > PerformanceMetrics_Result.txt

```
Single Stage Core Performance Metrics-----------------------------
Number of cycles taken: 28
Cycles per instruction: 2.54545
Instructions per cycle: 0.392858
Instructions: 11.0
```

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

Solution:

The following optimizations or features can be added to improve performance:

1. We can lower the overhead needed in between the steps to load and store required data.
2. Branch mis-prediction latency is a major issue affecting processor performance. To resolve this control flow hazard we can substitute arithmetic and lookup operations for the branch instructions.
3. At hardware level we can use small and fast caches to reduce memory access latency. This includes instruction caches and data caches to speed-up execution.
4. Another hardware level optimization is using simple static branch prediction techniques improve the flow of instruction execution, reducing the flow of control flow changes.
5. Software level optimization may include enhancing the compiler to generate more efficient code specifically designed for the RISC-V ISA.
6. Another software level optimization is to apply loop unrolling in software to reduce the overhead of loop control and increase the instruction throughput.
7. Optimizing the ALU for faster computation by using carry lookahead adders or other types of fast arithmetic circuits.
8. Using profiling to identify bottlenecks and optimizing critical sections of the code.
9. Optimize interrupt handling mechanisms to minimize latency and the overhead of context switching.
10. Improving bus architecture to allow faster data transfer rates between the processor and the peripherals.