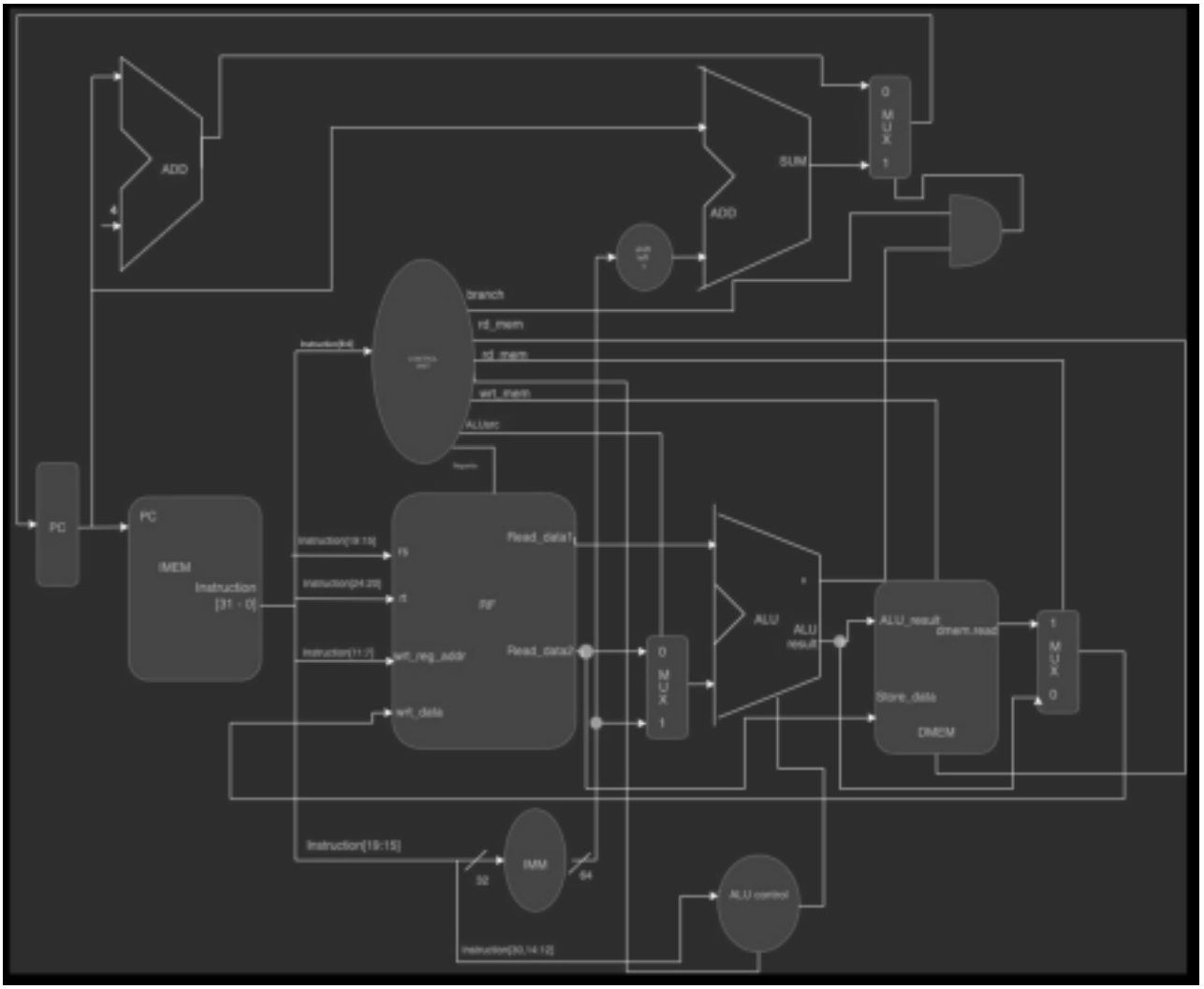
ECE 6913

COMPUTING SYSTEMS ARCHITECTURE

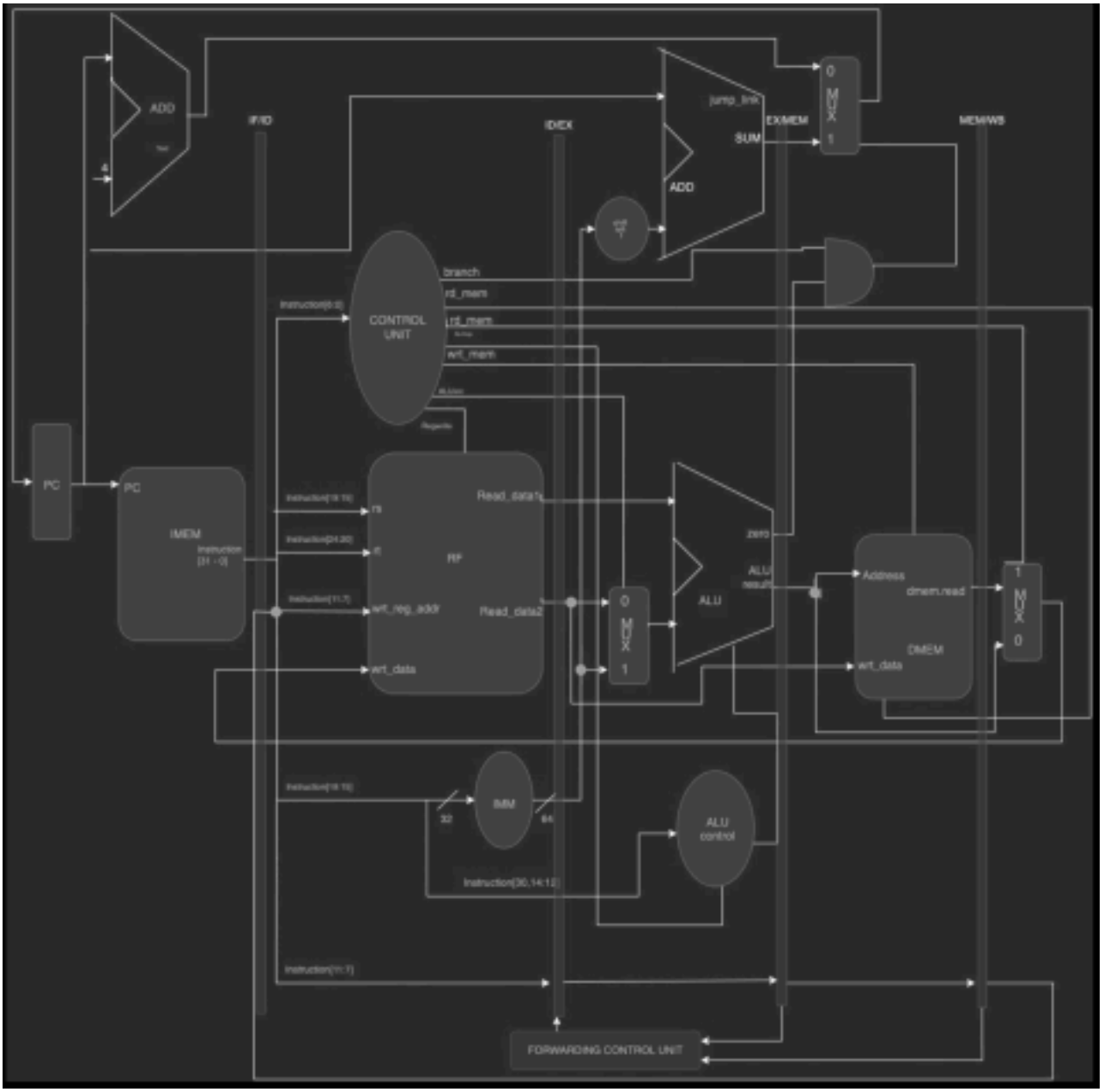
PROJECT : A

Srushti Jagatp sj4182

Task 1: Single Stage Schema2c



Task 2: Five Stage Pipeline Schema2c



Task 3: Results

Testcase 1:

PerformanceMetrics.txt

```
Performance of Single Stage:
#Cycles -> 6
#Instructions -> 5.0
CPI -> 1.2
IPC -> 0.8333333333333334
```

Metrics_FS_Result.txt .../testcase3 | PerformanceMetrics_FS_Result.txt .../testca

submission > sample_testcases_S24 > input > testcase1 > PerformanceMetrics

```
Five Stage Pipelined implementation:
        Number of cycles: 8
        Number of instructions executed: 5.0
        Average CPI: 1.6
        Instructions per cycle: 0.625
```

Testcase 2:

PerformanceMetrics.txt

```
Performance of Single Stage:
#Cycles -> 40
#Instructions -> 39.0
CPI -> 1.0256410256410255
IPC -> 0.975
```

Metrics_FS_Result.txt .../testcase3 | PerformanceMetrics_FS_Result.txt .../testcase2

submission > sample_testcases_S24 > input > testcase2 > PerformanceMetrics_FS_Result.txt

```
Five Stage Pipelined implementation:
        Number of cycles: 44
        Number of instructions executed: 39.0
        Average CPI: 1.1282051282051282
        Instructions per cycle: 0.8863636363636364
```

Testcase 3:

PerformanceMetrics.txt

```
Performance of Single Stage:
#Cycles -> 7
#Instructions -> 4.0
CPI -> 1.75
IPC -> 0.5714285714285714
```

main.py | PerformanceMetrics_FS_Result.txt

submission > sample_testcases_S24 > input > testcase3 > PerformanceMetric

```
Five Stage Pipelined implementation:
        Number of cycles: 9
        Number of instructions executed: 4.0
        Average CPI: 2.25
        Instructions per cycle: 0.4444444444444444
```

Test Cases	No. of Cycles	CPI	IPC
	SS	SS	SS
TC1	6	1.2	0.833
TC2	40	1.026	0.975
TC3	7	1.75	0.571

Test Cases	No. of Cycles	CPI	IPC
	FS	**FS**	**FS**
TC1	8	1.6	0.625
TC2	44	1.128	0.88
TC3	9	2.25	0.44

Average CPI:

Single Stage: 1.325
Five Stage: 1.659

Task 4: Comparison of Results

Unlike a single-stage configuration, the pipelined method introduces additional cycles for execution. However, it compensates with a notably higher clock frequency compared to the single-stage setup. This advantage stems from its utilization of combinational logic between pipeline registers, allowing for independent processing regardless of time constraints and facilitating faster operation. While a single-stage implementation requires fewer cycles and achieves a lower CPI with higher IPC, it ultimately consumes more time compared to its pipelined counterpart.

Task 5: Optimization

Performance improvements can be attained through the following optimizations:

- Enhancing the Arithmetic Logic Unit (ALU) by utilizing exclusive OR instead of a multiplexer for selecting operation results, thereby boosting circuit speed. Additionally, employing one-hot encoding within binary encoding to generate control signals for selecting ALU calculation results enhances the operating frequency of the ALU.

- Optimizing data alignment and sign extension for data memory output to ensure efficient operation.

- Upgrading the instruction fetch unit, which involves implementing a pipelined branch mechanism. Introducing a two-stage pipelining approach for the branch predictor can enhance the frequency of the instruction fetch unit.