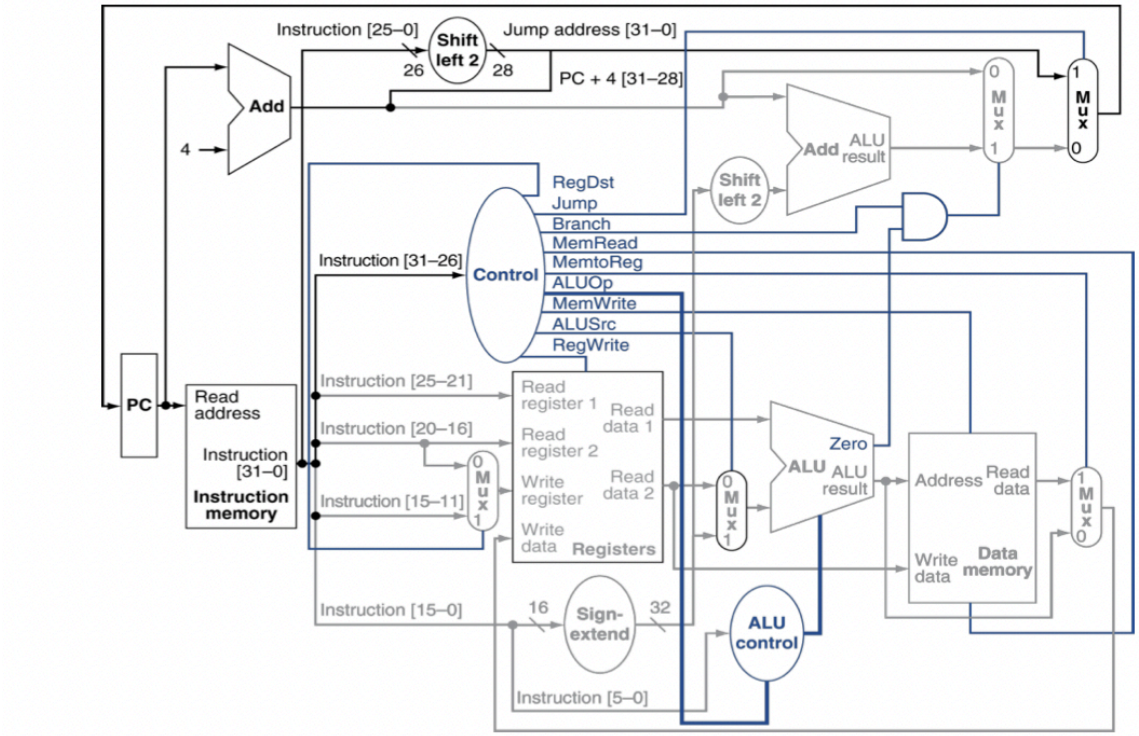
RISC-V Simulator

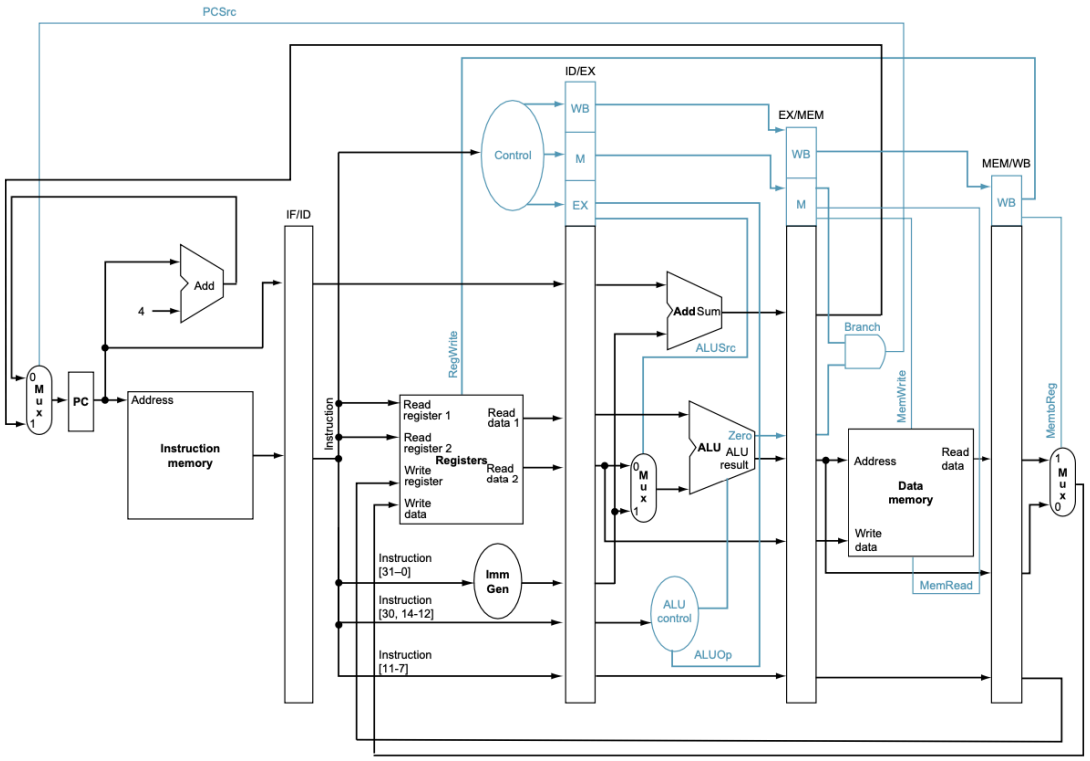
Shreyas Bhaktharam - sb9855

ECE-6913

1) Single Stage Processor



2) Five Stage Pipelined Processor



3) Metrics

```
SINGLE STAGE CORE PERFORMANCE METRICS
----------------------------------------
Total instructions: 6
Total cycles: 5
CPI: 0.833333
Instructions per cycle: 1.2
----------------------------------------

FIVE STAGE PIPELINE PERFORMANCE METRICS
----------------------------------------
Total instructions: 9
Total cycles: 8
CPI: 0.888889
Instructions per cycle: 1.125
----------------------------------------
```

From the metrics, it is evident that the five strage pipeline outperforms the single stage core across all metrics.

4) **Comparing results**

From the results from both cores, there are obvious advantages for choosing a five stage pipeline. Despite the slight increase in CPI, the overall instructions per cycle reduces and this reduction is apparent when the core executes 100's or 1000's of instructions. For smaller programs with more predictable and independent instructions, a single stage core processor would be the better option.

The five stage pipelined processor generally performs better than the single stage processor. It has a lower CPI and higher IPC, indicating better throughput. This is because the pipelined processor can execute multiple instructions simultaneously, with each stage working on a different instruction.

However, the pipelined processor's performance is somewhat reduced by stalls due to hazards. Without these stalls, its performance would be even better. The single stage processor, while simpler, executes only one instruction at a time, leading to lower performance.

5) **Optimizations**

Some optimizations include dynamic branch prediction, static branch prediction, instruction level parallelism. Adding a cache and implementing cache prefetching logic would also optimise some frequently accessed instructions.

Another worthwhile optimization would be to split the execution stage of the pipeline into multiple cycles for certain complex operations. This would result in higher clock frequency and better efficiency of the processor.

Implementing a branch predictor due to control hazards can reduce stalls and reduce penalties and improve latency for branch instructions.

Another major improvement would be to fetch commonly accessed instructions to reduce memory latency.