Performance Modeling - RISC-V processor - (PROJECT A)

GROUP:
MEMBER 1: Moin Mohammad Haroon Khan (mk8793)
MEMBER 2: Priyanshi Basant Singh (ps4609)

Tasks:

1) Draw the schematic for a single-stage processor and fill in your code to run the simulator. (20 points)

Answer:

The 5 stages of Instruction Execution are as follows:
Stage 1: Instruction fetch
Stage 2: Instruction decode
Stage 3: ALU (Arithmetic and Logic Unit)
Stage 4: Memory Access
Stage 5: Register Write

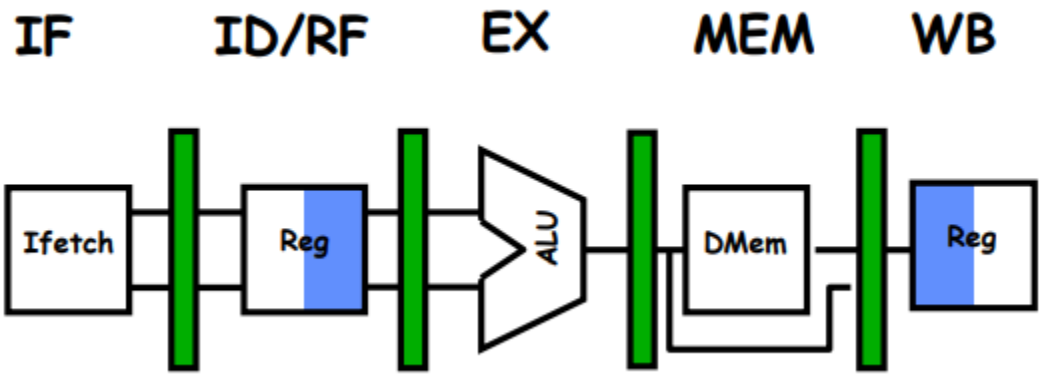
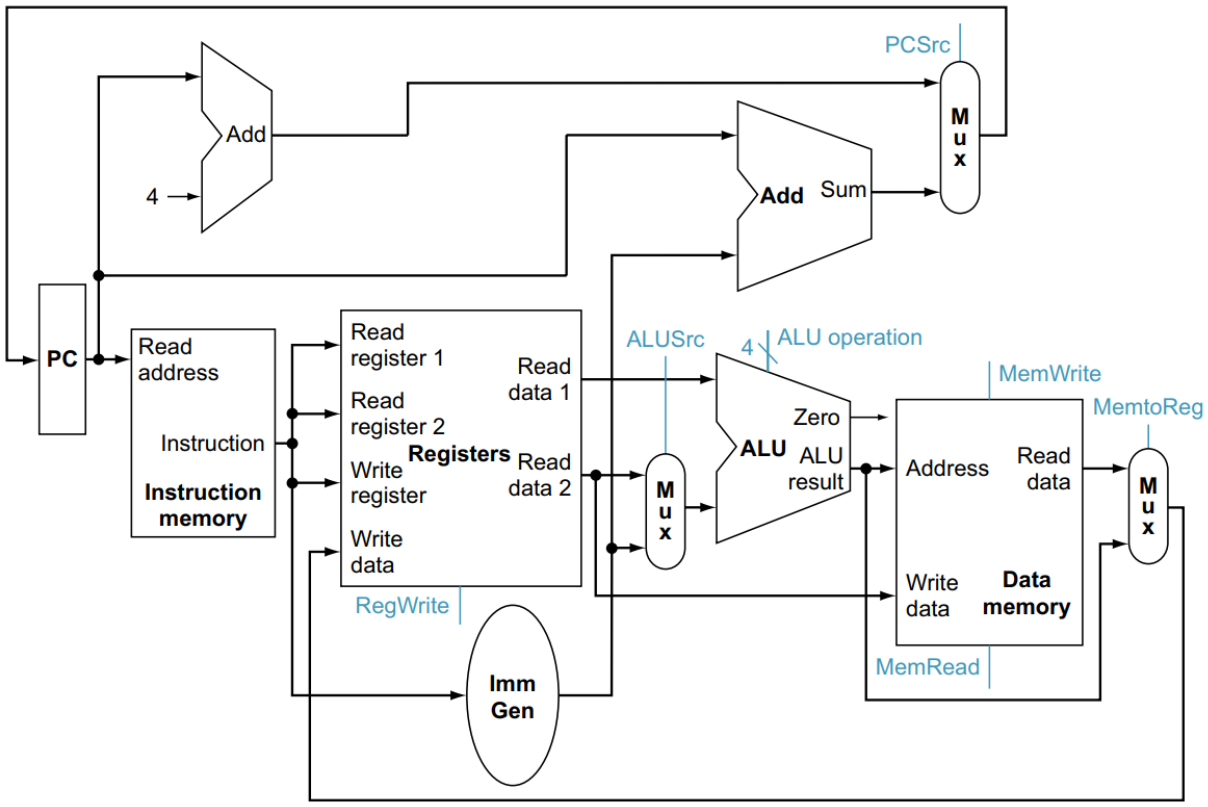


Fig 1: Schematic for a single-stage processor

In this schematic, the processor executes one instruction at a time in a single stage. Each stage has a specific task:

- Fetch: fetches the next instruction from memory based on the current program counter (PC).
- Decode: decodes the instruction to determine the operation to be performed and the operands to be used.
- Execute: performs the operation specified by the instruction and produces a result.
- Memory: accesses memory to read or write data as required by the instruction.
- Write: writes the result of the operation back to the appropriate register.

STAGES of EXECUTION and DATAPATH:



A single-stage RISC-V processor typically consists of the following components:

Program Counter (PC): This component holds the memory address of the next instruction to be executed.

Instruction Memory: This component stores the instructions that the processor will execute. It typically consists of a ROM or a cache.

Instruction Decoder: This component decodes the instruction fetched from the instruction memory and generates the necessary control signals to execute the instruction.

Register File: This component stores the processor's general-purpose registers. The number of registers in the register file varies based on the RISC-V implementation.

ALU: This component performs arithmetic and logical operations on the data stored in the registers.

Data Memory: This component stores the data that the processor uses during its computations. It typically consists of a RAM or a cache.

Control Unit: This component coordinates the activities of the other components to ensure that instructions are executed correctly.

---

3) Measure and report average CPI, Total execution cycles, and Instructions per cycle for both these cores by adding performance monitors to your code. (Submit code and print results to console or a file.) (5 points)

Answer:

Single Stage Core Performance Metrics
Number of cycles taken: 12
Cycles per instruction(CPI): 1.0
Instructions per cycle: 1.0

---

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.(5 points)

Answer:

When comparing a single-stage processor and a five-stage pipelined processor, the pipelined processor generally has a higher performance but also a higher complexity.

In a single-stage processor, each instruction is executed in a single cycle without any pipeline stages. This design is simple and easy to implement, but it does not take advantage of pipelining, which can result in lower performance. As a result, a single-stage processor may have a lower clock frequency and/or a longer execution time per instruction compared to a pipelined processor.

On the other hand, a pipelined processor has multiple pipeline stages that allow multiple instructions to be executed simultaneously. This design can achieve a higher performance by overlapping the execution of different instructions, reducing the overall execution time per instruction. However, pipelining also introduces new challenges, such as data dependencies, branch prediction, and pipeline hazards that must be addressed through techniques such as forwarding, stalling, and speculative execution.

Consider the following instruction,

```
0:      LW R1, R0, #0       // Load Mem[R0+0] into R1 - Value 8
4:  C1:  ADDI R2, R2, #4     // Increment R2 by 4
8:      BNE R1, R2, C1      // If R1 != R2 branch to C1
12:     HALT                // Halt
```

For above instruction,

Single Stage Core Performance Metrics----------
Number of cycles taken: 7
Cycles per instruction: 1
Instructions per cycle: 1

Five Stage Core Performance Metrics----------
Number of cycles taken: 10
Cycles per instruction: 2.5
Instructions per cycle: 0.4

In the single-stage core, each instruction is executed in a single cycle, so the total number of cycles taken is equal to the number of instructions, which is 4. Therefore, the single-stage core takes 7 cycles to complete the execution of all instructions, including the BNE branch instruction that causes a control hazard and results in a stall.

In the five-stage pipelined core, the instructions can be overlapped in different stages of the pipeline, which can reduce the overall execution time per instruction. However, pipelining also introduces additional overhead due to pipeline hazards, such as data dependencies and control hazards. In this case, the BNE branch instruction also causes a control hazard that results in a stall.

As a result, the five-stage pipelined core takes 10 cycles to complete the execution of all instructions, which includes 2 stalls due to the control hazard caused by the BNE branch instruction.

In terms of performance metrics, the single-stage core has a CPI of 1, indicating that each instruction takes one cycle to execute. The five-stage pipelined core has a CPI of 2.5, indicating that on average, each instruction takes 2.5 cycles to execute. However, the five-stage pipelined core also has a higher instructions-per-cycle (IPC) metric, which indicates that it can execute more instructions in a given cycle, on average. The IPC of the five-stage pipelined core is 0.4, meaning that it can execute 0.4 instructions per cycle, on average.

In summary, the five-stage pipelined core performs better in terms of the total number of cycles taken to execute the instructions, but it also has a higher CPI due to the overhead introduced by pipelining. The choice between a single-stage core and a pipelined core depends on the specific requirements of the application, such as the required performance, power consumption, and design complexity.

---

5) What optimizations or features can be added to improve performance? (Extra credit 1 point)

Answer:

A single-stage RISC-V processor is a very basic design and has limited performance compared to more complex multi-stage designs. However, there are still some optimizations and features that can be added to improve its performance, such as:

1. **Pipelining:** Pipelining is the process of breaking down the processor's instruction execution into multiple stages so that multiple instructions can be executed simultaneously. By doing this, the processor can improve its performance and increase the throughput of the system.
2. **Branch prediction:** A branch prediction unit can be added to the processor to predict the outcome of a branch instruction before it is actually executed. This can reduce the number of pipeline stalls that occur when the processor has to wait for the branch instruction to be executed before continuing with the next instruction.
3. **Caches:** The addition of data and instruction caches can reduce the amount of time the processor spends accessing main memory, which is slower. This can improve the overall performance of the processor by reducing the number of cycles spent waiting for data to be fetched from memory.
4. **Speculative execution:** Speculative execution is a technique where the processor predicts the outcome of a conditional instruction and begins executing the instructions that follow it before the outcome of the conditional instruction is known. This can improve performance by reducing pipeline stalls.
5. **Out-of-order execution:** Out-of-order execution is a technique where the processor executes instructions in a different order than they are presented in the program, as long as the data dependencies are preserved. This can improve performance by allowing the processor to execute instructions that are not dependent on the results of other instructions, reducing pipeline stalls.
6. **Multi-issue:** Multi-issue is a technique where the processor can execute multiple instructions in a single clock cycle, as long as they are independent of each other. This can improve performance by increasing the number of instructions executed per clock cycle.
7. **SIMD instructions:** SIMD (Single Instruction Multiple Data) instructions can be added to the processor to perform the same operation on multiple data elements in parallel. This can improve performance in applications that involve a lot of data processing, such as an image or audio processing.

These optimizations and features can be added to a single-stage RISC-V processor to improve its performance, but they will also increase the complexity of the processor's design. As a result, there is often a trade-off between performance and design complexity.
