Computer Organization and Architecture

(21ECSC201)

Activity Report

TEAM B-03

GUIDED BY

NEELAM SOMANNAVAR MAM

Course details

Course Name	Computer Organization and Architecture
Course Code	21ECSC201
Semester	III
Division	B
Year	2023-24

Team Details

Group No:

Si. No.	Roll No.	S. R. N.	Student Name
1.	03	02FE22BCS013	AKHILESH JOSHI
2.	13	02FE22BCS035	DILIPSINGH RAJPUROHIT
3.	37	02FE22BCS105	SANIKA UTTARKAR
4.	63	02FE22BCS147	SOUJANYA MIRAJKAR

1. Problem Statement

Design and simulate a 2-stage pipelined 8-bit processor which executes 1 address format code snippet using Direct addressing mode.

2. Brief description of the functional specifications of the Processor designed

1. Processor Architecture:

- The processor consists of two pipeline stages: Instruction Fetch (IF) and Execution (EX).
- It operates on 8-bit data.

2. Instruction Set and Addressing Mode:

- The processor supports a 1-address format, meaning each instruction specifies one operand.
- Direct addressing mode is employed, indicating that the operand's address is directly provided in the instruction.

3. Registers:

- The processor includes a minimal set of registers, such as an Accumulator (ACC) to store intermediate results during execution.

4. Instruction Format:

- The instruction format is designed to accommodate the 1-address format and direct addressing mode.
- Example format: `OPCODE ADDRESS`, where OPCODE represents the operation code, and ADDRESS represents the direct address of the operand.

5. Pipeline Stages:

- *Instruction Fetch (IF):* Fetches the instruction from memory.
- *Execution (EX):* Executes the instruction, performing the specified operation using the data retrieved from the direct address.

6. Pipeline Hazard Handling:

- Address hazards need to be addressed, ensuring that the correct operand is available for execution.
- Possible solutions include stalling the pipeline or implementing forwarding mechanisms.

7. Memory Access:

- The processor interacts with memory to fetch and store data. In direct addressing mode, the specified address is used to access data directly from memory.

8. Control Unit:

- The control unit manages the flow of instructions through the pipeline, controls the execution of operations, and handles any required data transfers.

9. Simulation and Testing:

- The designed processor should be simulated using appropriate tools to validate its functionality.
- Test cases should be created to cover a range of instructions, addressing scenarios, and potential edge cases.

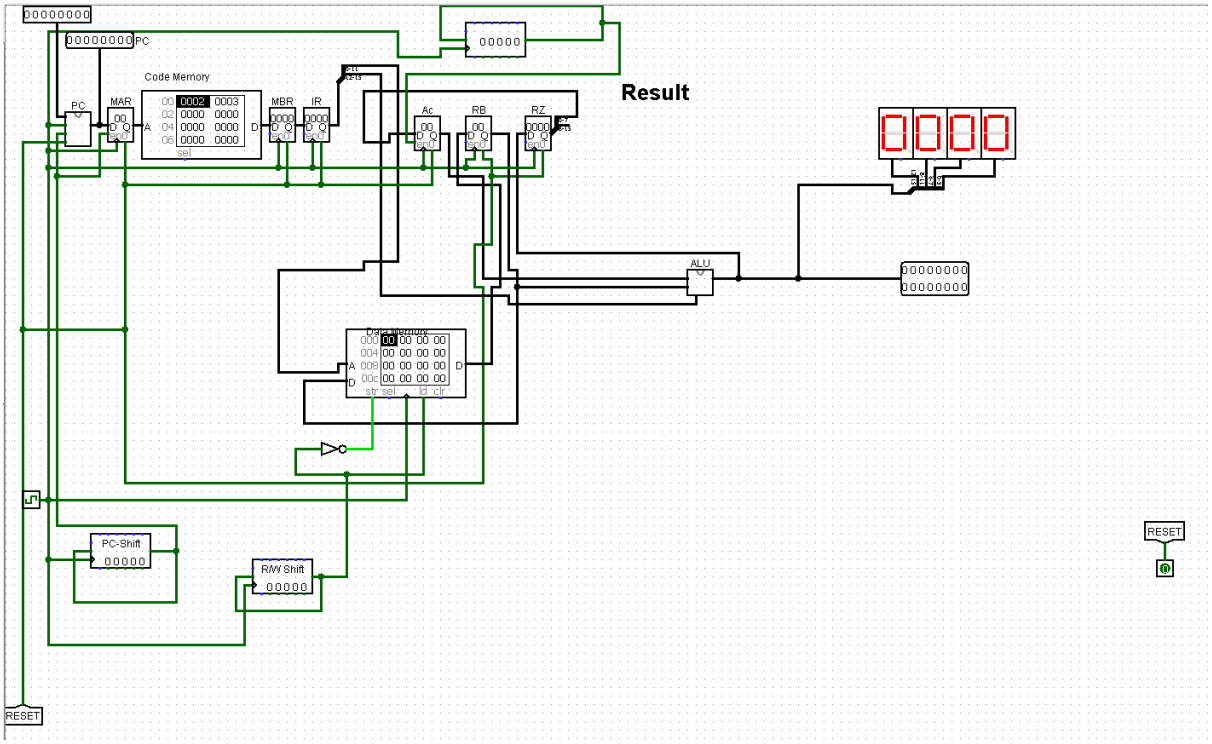
10. Performance Considerations:

- Evaluate the performance of the pipelined processor, considering factors such as throughput, cycle time, and any potential bottlenecks.
- Optimize the design iteratively based on simulation results.

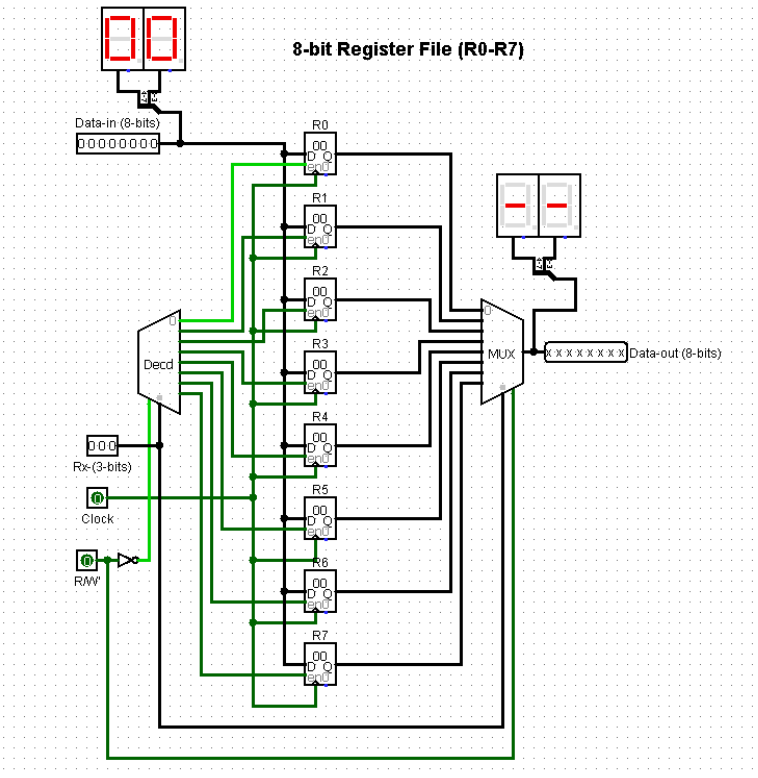
3. Code snippet executed on the machine

```
; Code Snippet 1: Addition
Assume memory locations for operands and result:
; Memory location 0x10: Operand1
; Memory location 0x20: Operand2
; Memory location 0x30: Result

; Load operand1 into ACC (Accumulator)
LOAD 0x10
; Add operand2 to ACC
ADD 0x20
; Store the result back to memory
STORE 0x30

; Code Snippet 4: Product of Two Numbers
Assume memory locations for operands and result:
; Memory location 0x80: Operand1
; Memory location 0x81: Operand2
; Memory location 0x90: Result

; Load the first operand into ACC
LOAD 0x80
; Multiply ACC by the second operand
MUL 0x81
; Store the result back to memory
STORE 0x90
```

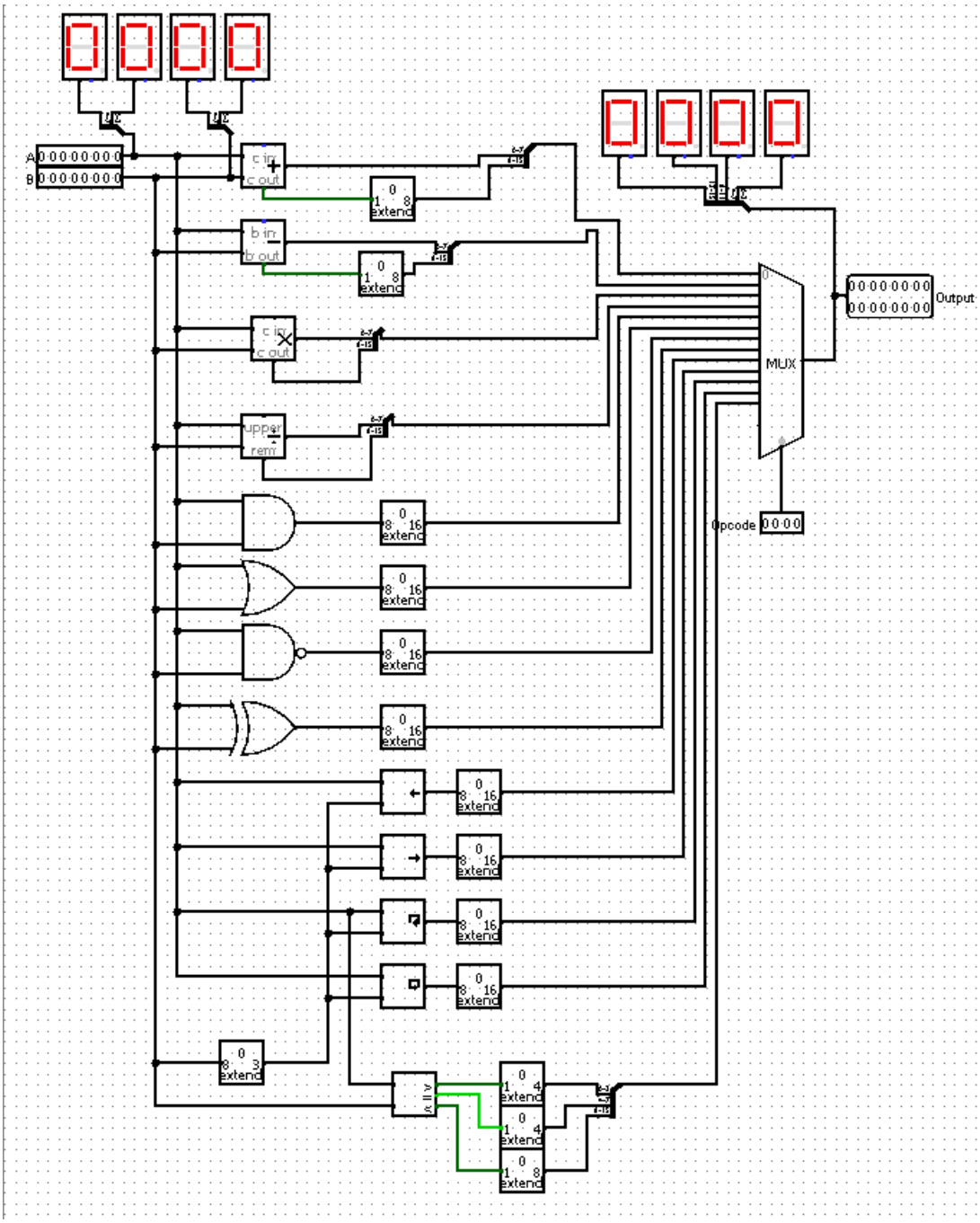
4. Processor Design



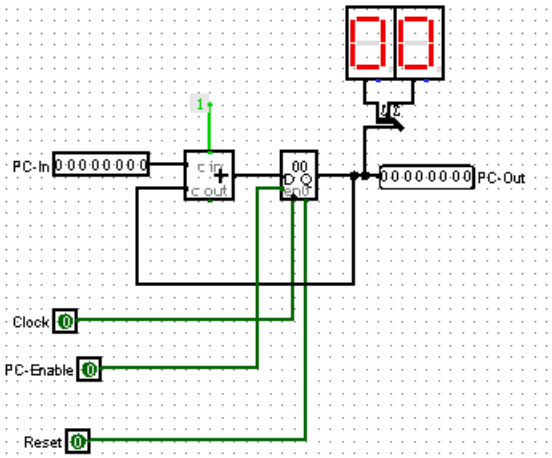
8BIT REGISTER FILE



ALU DESIGN



PC DESGIN :



5. Challenges Faced

- Calculating the bits
- Correcting the incompatible width
- Debugging the connections
- Unable to execute all the ALU operations
- Balancing the size of the register
- Unable to get correct output for given input
- Balancing simplicity of the instruction set with the need for functionality

6. How you overcame those challenges

Calculating the Bits:

Perform a thorough analysis of the data requirements for each component . Use simulation tools to verify that the chosen bit widths meet the design specifications.

Debugging the Connections:

Employ simulation tools to trace and monitor data and control signals.

Execution of all ALU operation:

- Review the ALU design and control logic to ensure it supports all necessary operations.
- Simulate specific ALU operations to identify and address any limitations.
- Verify that control signals are correctly generated for each ALU operation.

Balancing the Size of the Register:

- Evaluate the data storage requirements for various operations in the processor.
- Adjust register sizes based on the needs of the instructions and data types.

Unable to Get Correct Output for Given Input:

- Conduct extensive testing with diverse input scenarios to identify discrepancies.
- Use debugging tools to trace the flow of data and instructions, isolating the source of errors.
- Verify the correctness of memory access, data paths, and control signals.

7. Brief Comparative study of contemporary processors

CPU (Central Processing Unit):

- **Definition:** The CPU is the primary component of a computer that performs arithmetic, logic, control, and input/output operations. It interprets and executes instructions from the computer's memory.

- **Contemporary Aspect:** Modern CPUs often have multiple cores, allowing them to execute multiple tasks simultaneously through parallel processing.

Register:

- **Definition:** Registers are small, fast storage locations within the CPU used to store temporary data, operands, and addresses. They play a crucial role in the execution of instructions.
- **Contemporary Aspect:** Modern processors have various types of registers, including general-purpose registers, floating-point registers, and vector registers, each serving specific purposes to enhance overall performance.

ALU (Arithmetic Logic Unit):

- **Definition:** The ALU is a digital circuit within the CPU responsible for performing arithmetic and logic operations, such as addition, subtraction, AND, OR, and NOT.
- **Contemporary Aspect:** Contemporary ALUs are highly optimized and may include features like SIMD (Single Instruction, Multiple Data) for parallel processing and support for advanced instructions to accelerate specific tasks.

Core Memory (Cache):

- **Definition:** Core memory refers to the fast and small-sized memory units located within the CPU,

often known as cache memory. It stores frequently used data to reduce access times.

Data Memory (RAM):

- **Definition:** Data memory, typically in the form of RAM (Random Access Memory), is used for storing data that the CPU can access quickly. It is volatile and temporary.

PC (Program Counter):

- **Definition:** The Program Counter is a register that keeps track of the memory address of the next instruction to be executed.
- **Contemporary Aspect:** In modern processors, the PC is crucial for maintaining the instruction execution sequence, and it is often complemented by branch prediction and speculative execution mechanisms to improve performance.

MBR (Memory Buffer Register) and MAR (Memory Address Register):

- **Definition:** MBR holds data to be written to memory or receives data read from memory. MAR holds the address in memory of the data to be read or written.
- **Contemporary Aspect:** These registers are fundamental for memory operations.
  Modern processors optimize memory access

through techniques like out-of-order execution and prefetching.

8. Conclusion

1. Understanding Processor Architecture: **Designing a pipelined processor requires a solid understanding of processor architecture, including concepts such as instruction fetching, decoding, execution, and write-back stages.**

2. Pipeline Hazards and Solutions: **Recognizing and addressing pipeline hazards is crucial. These may include data hazards (dependencies between instructions), control hazards (conditional branches), and structural hazards (resource conflicts). Implementing techniques like forwarding, stalling, and branch prediction can mitigate these issues.**

3. Instruction Set Architecture (ISA): **Knowing the instruction set and addressing modes of your processor is fundamental. In your case, dealing with a 1-address format code snippet with direct addressing mode implies a specific structure for instructions and data retrieval.**

Activity Takeaways:

1. Simulation and Testing: **Simulating the designed processor allows you to observe its behaviour and identify any issues. It's essential to create thorough test cases to validate the correctness and efficiency of your processor design.**

2. Optimizing Pipelined Execution: **Evaluate the performance of your pipelined processor and look for ways to optimize its execution. This might involve adjusting pipeline stages, improving instruction throughput, or minimizing pipeline stalls.**

3. Documentation and Reporting: **Proper documentation is crucial for any project. Clearly document the design choices, challenges faced, solutions implemented, and the reasoning behind your decisions. This documentation will be valuable for future reference or if the project is extended or handed over to others.**

4. Iterative Design Process: Building a processor is often an iterative process. Be prepared to refine your design based on simulation results, identify and resolve bottlenecks, and continually enhance the performance and reliability of your processor.