

SINGLE CYCLE PROCESSOR

By JAWWAD SARWAR

# SINGLE CYCLE PROCESSOR DESIGN REPORT

## INTRODUCTION

This report provides a comprehensive overview of a custom RISC-V processor designed to execute a subset of the RV32I instruction set architecture (ISA). The processor successfully runs C programs by translating them into RISC-V assembly, assembling them into machine code, and loading them into instruction memory for execution.

## INSTRUCTION SET COVERAGE

The processor supports the following instruction types from the RV32I ISA:  
- I-Type: Includes load instructions and immediate arithmetic/logic operations, excluding `ecall` and `ebreak`.  
- R-Type: Implements register-to-register arithmetic and logic operations.  
- S-Type: Handles store instructions.  
- B-Type: Enables conditional branch operations.  
- J-Type: Supports the **jal** instruction for jumps.  
- U-Type: Supports the **lui** instruction.

## MEMORY ARCHITECTURE

The processor features the following memory components:  
- Instruction Memory: A large memory space of 4MB, implemented to hold large program instructions also.  
- Data Memory: A 32KB memory module to store data during program execution.

## PROCESSOR MODULES

The processor is composed of several modules, each responsible for specific functions. Below is a brief description of each module:

### ALU (Arithmetic Logic Unit)

Performs arithmetic and logical operations based on control signals.

### CU (Control Unit)

Decodes instructions and generates control signals for other modules.

### inst\_mem (Instruction Memory)

Stores the program instructions loaded using the `readmemh` function.

### inst\_fetch (Instruction Fetch)

Handles program counter (PC) operations and retrieves instructions from memory.

### inst\_decode (Instruction Decode)

Decodes instructions and extracts control fields such as opcode, funct3, funct7, and operands.

### reg\_file (Register File)

Implements 32 general-purpose registers for storing intermediate computation values.

### data\_mem (Data Memory)

Stores and retrieves data used during program execution. Supports read and write operations.

### rv32i\_processor (Processor Core)

Integrates all modules and orchestrates instruction execution.

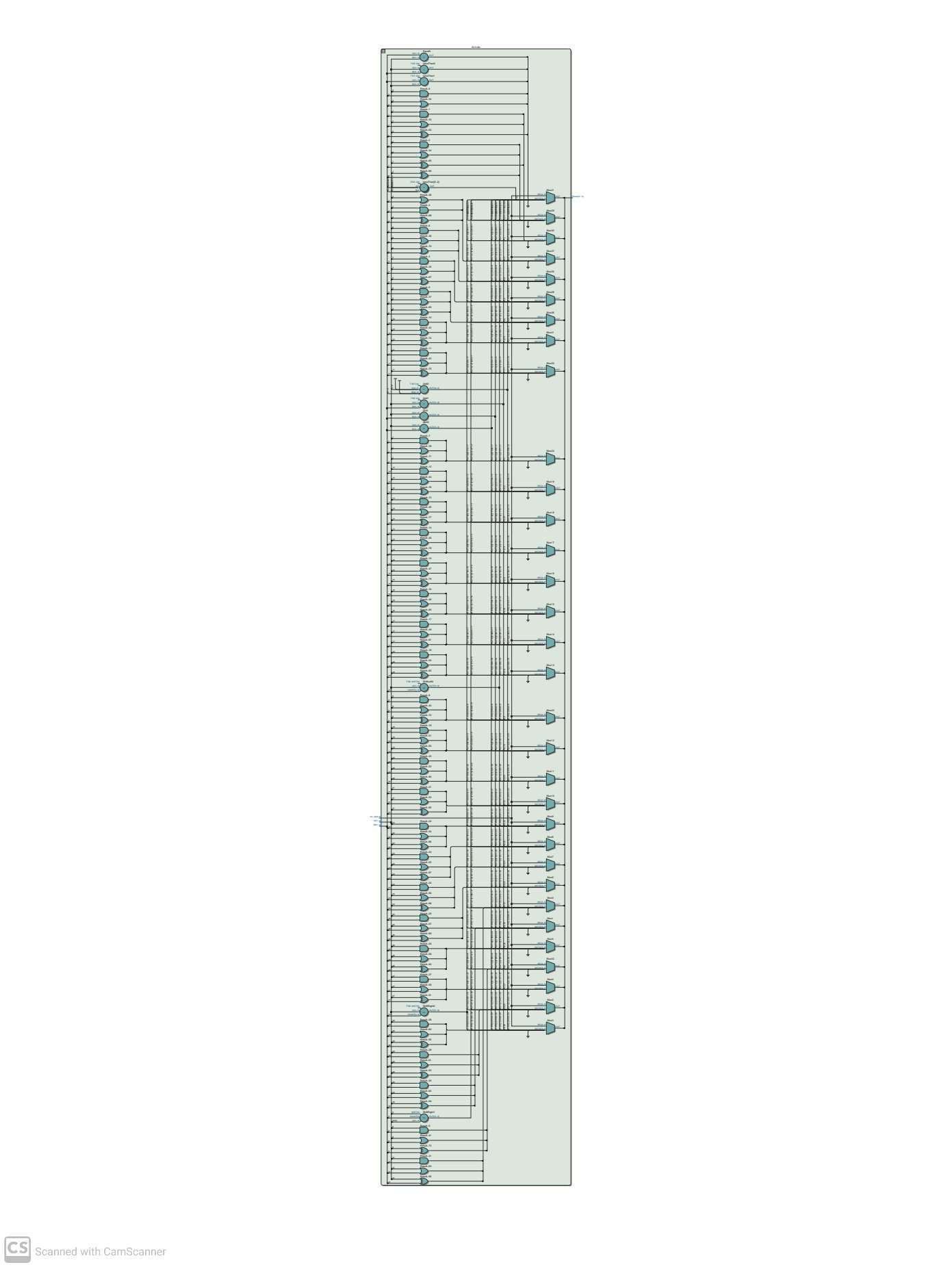
## WAVEFORMS

## 

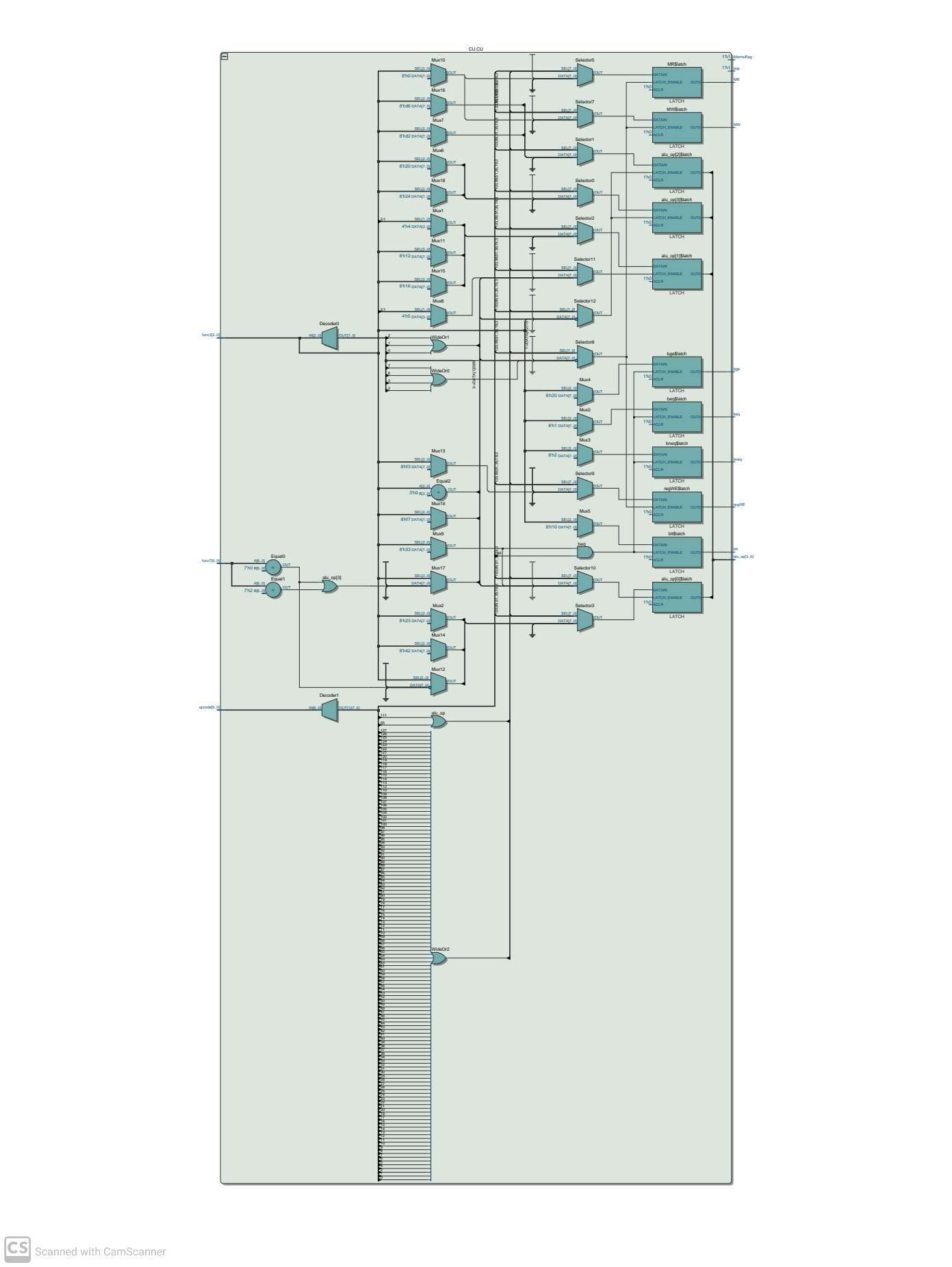
## 

## RTL DIAGRAMS:

## ALU



## CONTROL UNIT:



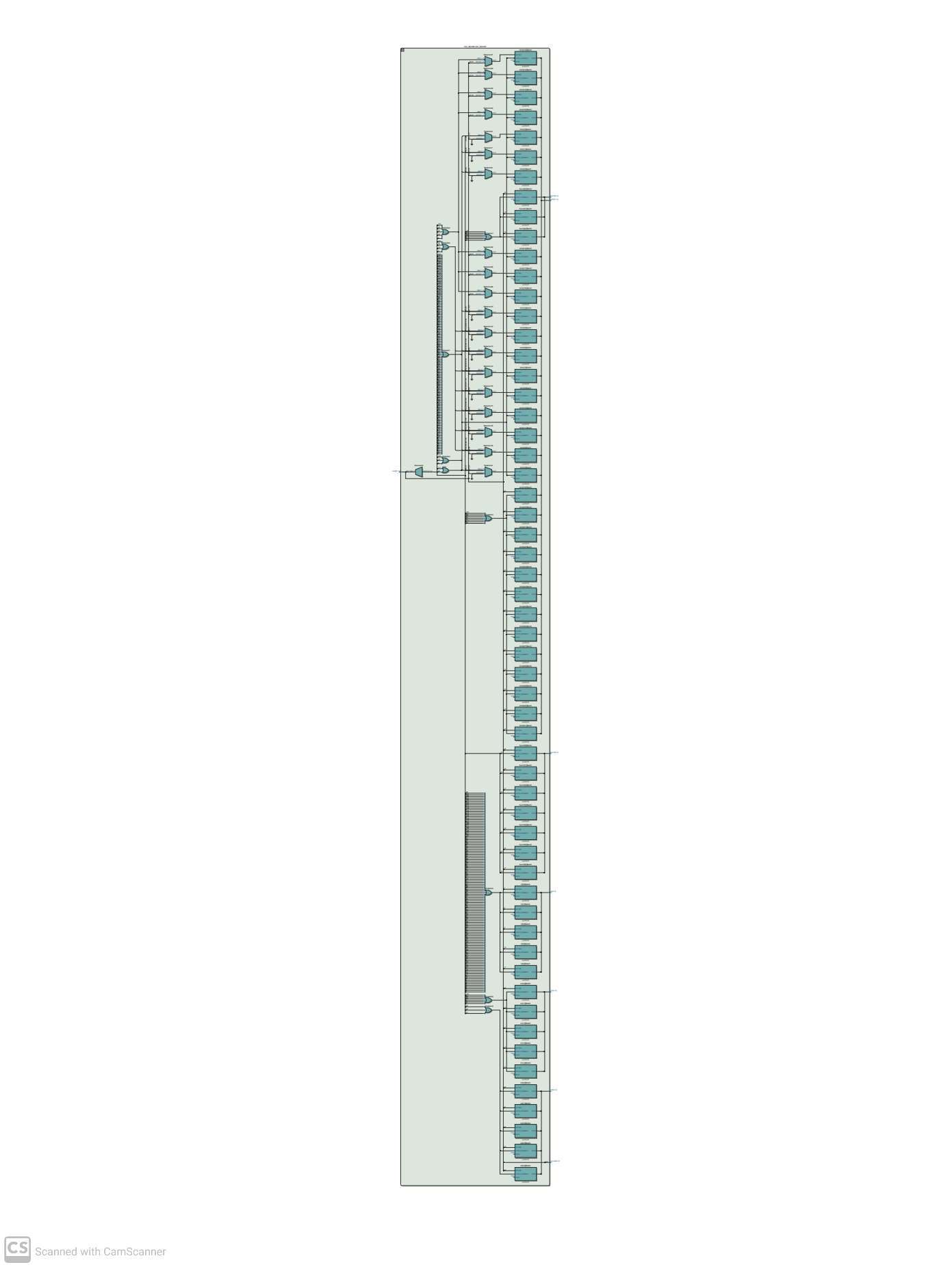
## INSTRUCTION MEMORY:

## 

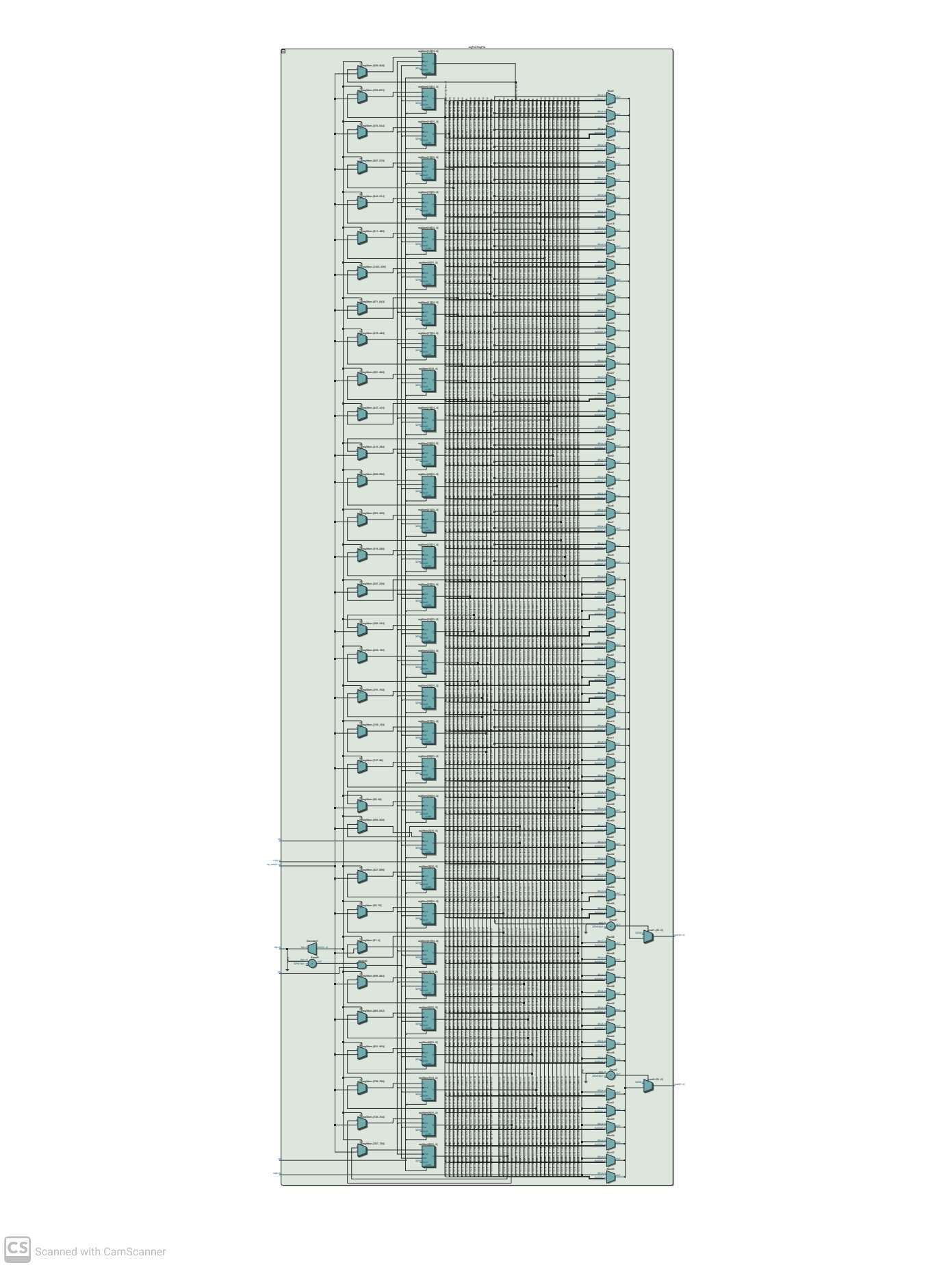
## INSTRUCTION FETCH:

## 

## INSTRUCTION DECODE:



## REG FILE:



## DATA MEMORY:

## 

## COMPLETE PROCESSOR (TOP FILE):

## 

## TECHINCALLY RUNNING C PROGRAM ON BARE-METAL

## Workflow Description

The design workflow involves the following steps:

C code

RISC-V assembly

HEX to Processor Memory

Assembly to HEX

## FUTURE WORK

* **Pipelining**: Implement a pipelined architecture to enhance instruction throughput by executing multiple stages simultaneously, including adding hazard detection and forwarding for efficient execution.
* **OS Support**: Introduce privilege modes and system call support to enable the processor to run operating systems, laying the groundwork for multitasking and resource management.

## CONCLUSION

This custom RISC-V Single Cycle processor demonstrates a successful implementation of the RV32I ISA with key modules and functionalities. The design supports a range of instruction types, adequate memory structures, and modular components, making it a versatile foundation for further development.