**16-bit Single Cycle Mips Processor**

Hamad Sayed Mariam Mohamed Heba Sabri

1. *Design and* *Implementation* :

1.1 Data Path :

The design of the CPU is a basic MIPS design with a subset of the instruction set. As our designed ISA, Our CPU should be able to do 32 instructions, 16 of which are R-type, 13 are I-type, and 3 are J-type. The general data path starts with fetch instruction word from memory address in PC , then fetch the register file to read the source register(s) used for the instruction. Next, it depends on the instruction type, with R-type both values come from the registers and go through the ALU to compute the instruction operation and then stores it into . With I-type, one of the operands is an immediate value instead of register like the R-type’s instructions. In both R and I types, they write the result of the ALU into the designated destination register except for the “sw” which saves into data memory. With J-type instructions, after the result of the ALU (the new PC address) is computed, it sent back around to the PC at which point the PC control unit sends the selector for the mux to accept the branched or jump PC address.

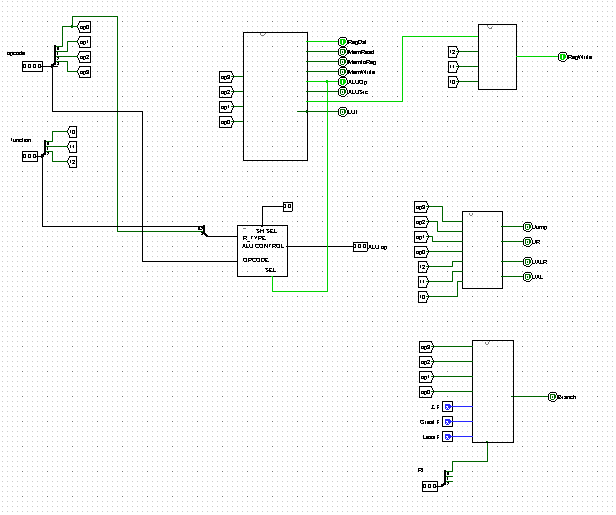
1.2 Control Unit :

There are four units for controlling the CPU: The Main Control unit, the ALU Control, and the PC Control. The Main Control unit handles the opcodes from the current instruction and enabling all of the relevant signals so that the rest of the processor will work as expected as described by the MIPS spec.

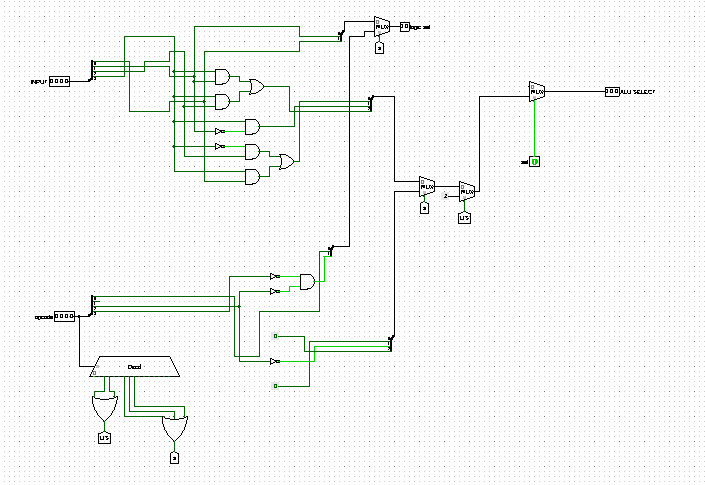
Some of the signals of the Main Control unit are also fed into the PC Control unit (J, JR, JAL, JALR, BEQ, BNE) and the ALU unit (ALUSrc, ALUOp) . The PC Control unit handles getting the next address that the PC register should save for the next cycle of the processor . The PC Control unit takes the signals of Jump, BEQ, and BNE, as well as Zero flag signal and computes from it a selection value to activate a channel in its multiplexer to select whether the PC will increment, jump, or branch. The ALU Control unit takes the signals ALUSrc and ALUOp from the Main control unit, and based on those signals, retrieves two data pieces A and B, where A is always from a register, and B could be either from a register or directly from the immediate value embedded in the instruction as selected by ALUSrc, and then based on the ALUOp value the ALU Operation is selected and then performed.

1.3 Component circuits and Overall DataPath :

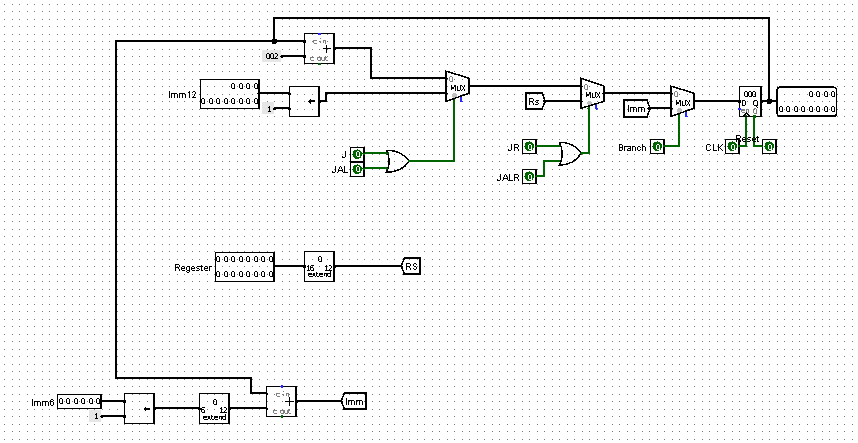
Main control



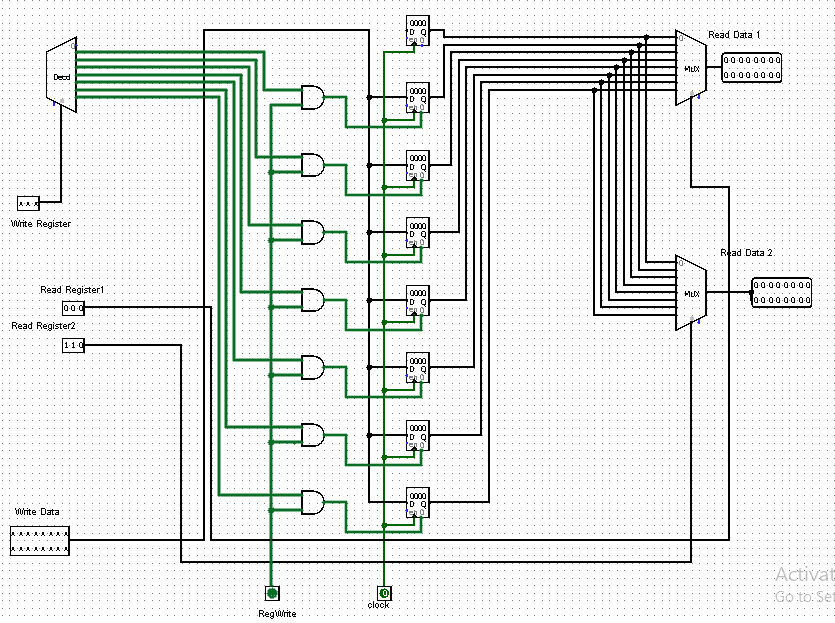
ALU Control



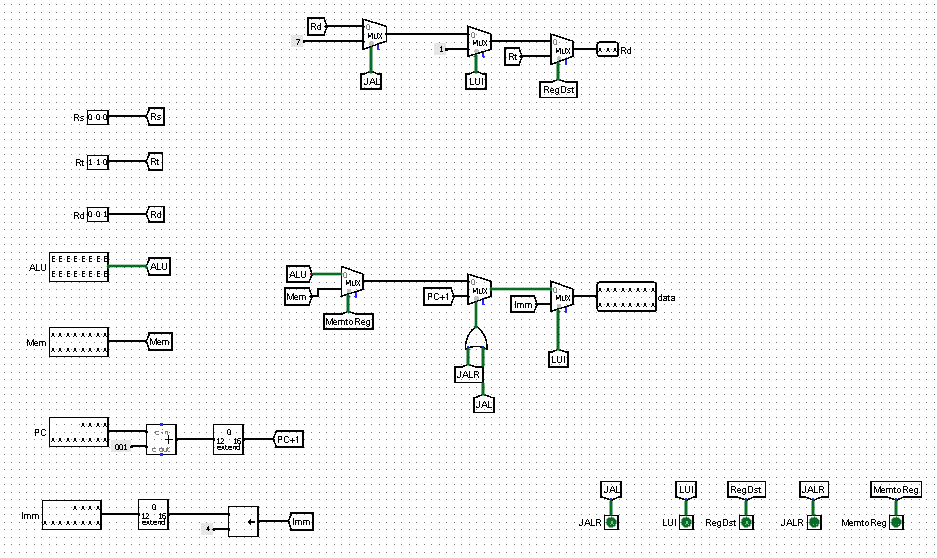
PC Counter



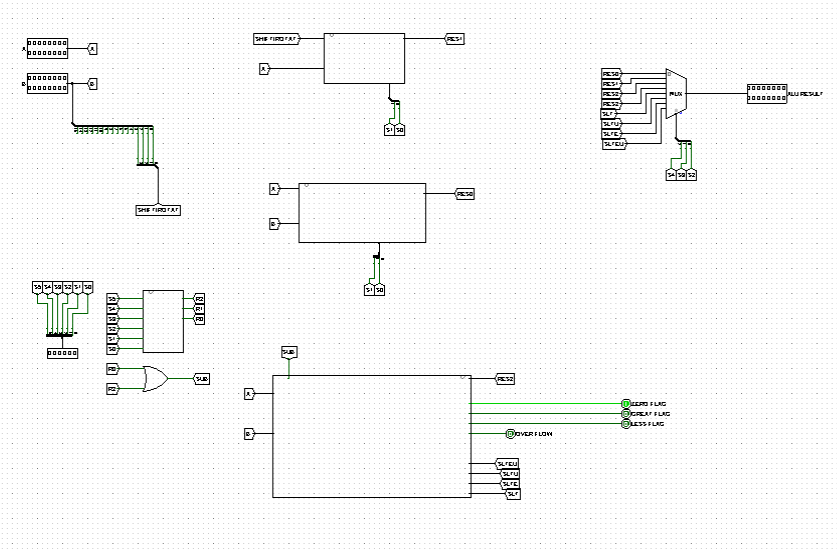
Register File



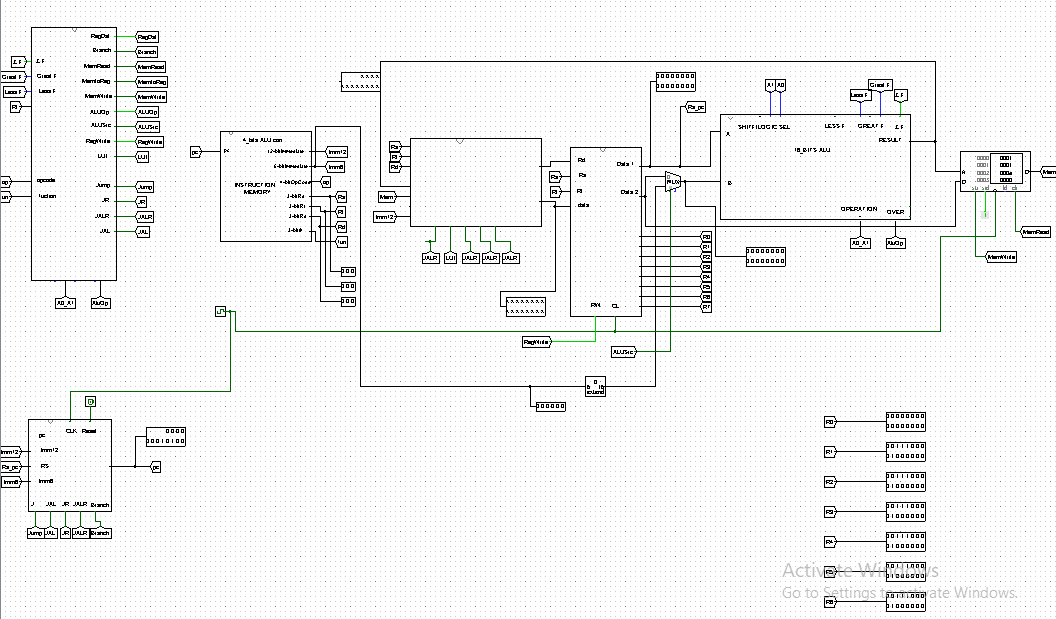
Register File Decoder



ALU



CPU circuit



1.4 Tables :

Table 1: Main Control Table

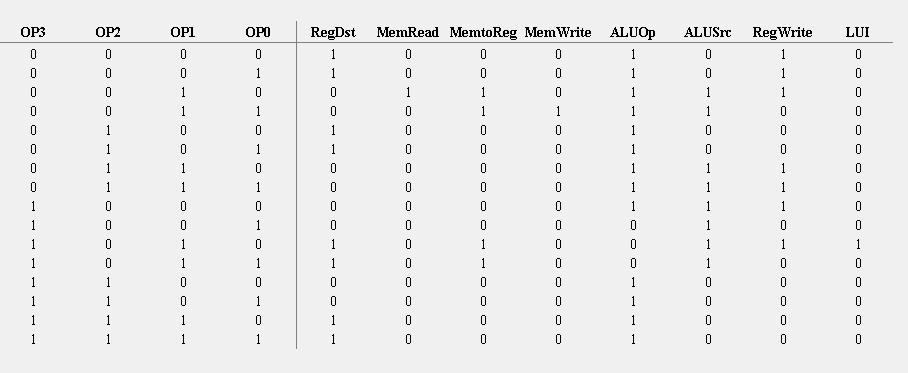


Table 2: Jump

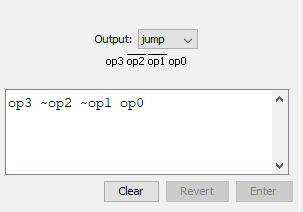


Table 3: JR

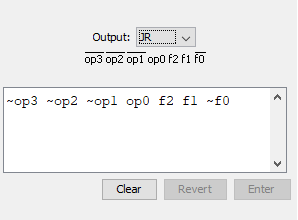


Table 4: JALR

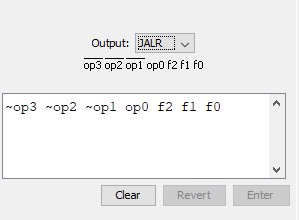


Table 4: JAL

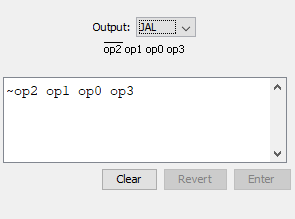
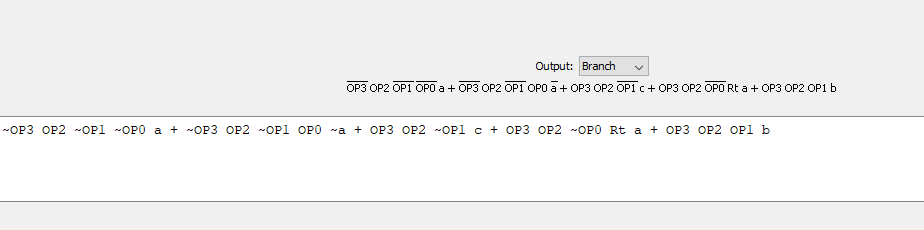


Table 5: Branch



1. *Simulation and* *Testing* :

2.1 Describe the test programs that you used to test your design with enough

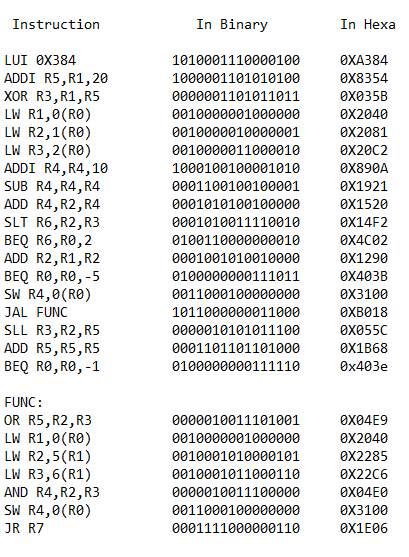
comments describing the program, its inputs, and its expected output:

For testing we use a program check the following 14 instructions:

LUI,Addi,Xor,lw,sub,Add,slt,Beq,sw,JAL,sll,or,And,Jr. We included hex

text files with preloaded memory files that are compatible with logisim.

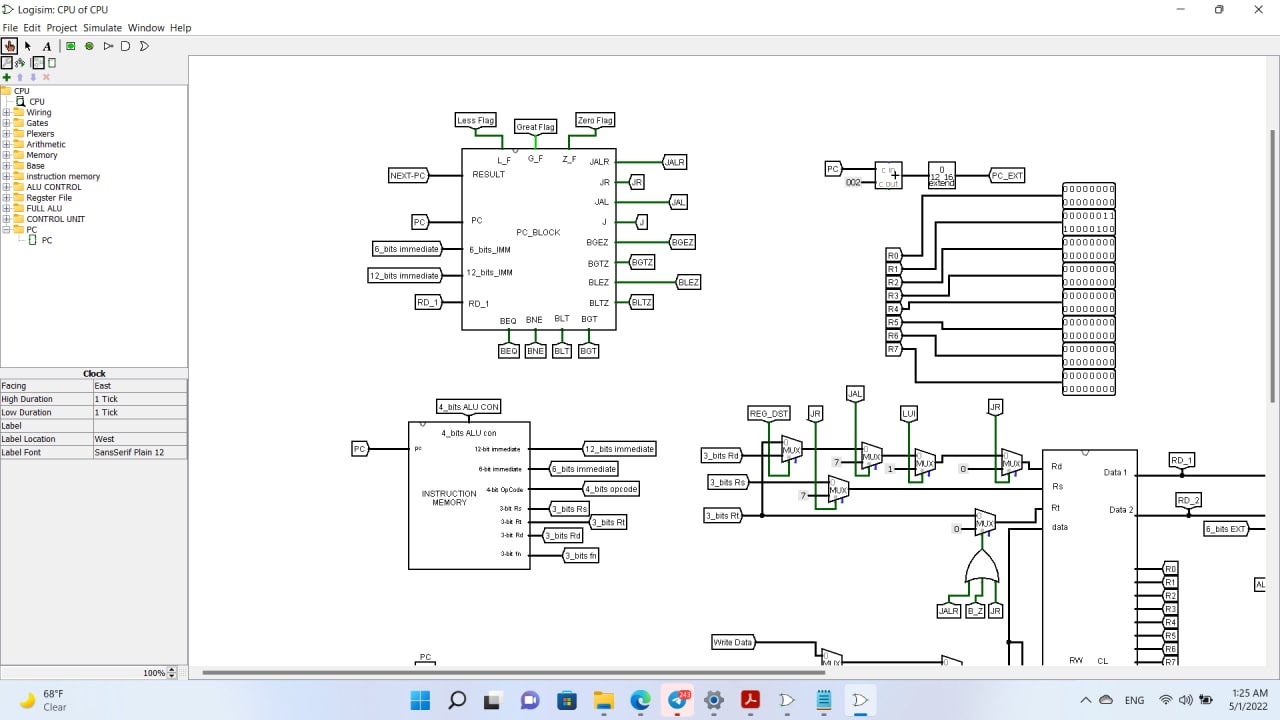
The Test Code :

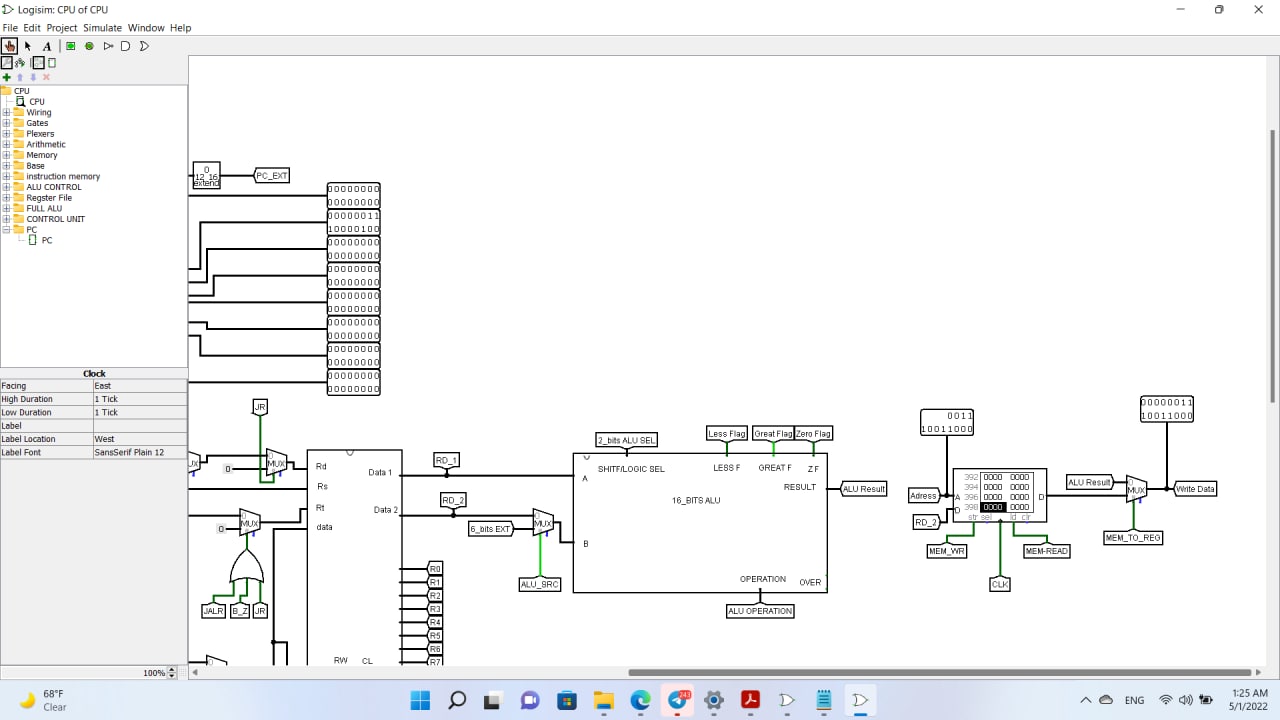


2.3 List all the instructions that were tested and work correctly:

all of the instructions have been implemented and operate correctly.

2.4 Provide snapshots of the Simulator window with your test program loaded and showing the simulation output results:





This instruction is testing LUI.

1. *Team* *Work :*

The control signal module and register circuit developed by Eng Mariam . The Alu circuit and Alu control developed by Eng Hamad . And the memories , converting the test code to hex and write the documentation by Eng Heba . The team make Zoom meetings to implementing the processor Datapath and Testing.