**CSE 3015 DIGITAL LOGIC DESIGN TERM PROJECT**



**YASİN ALPER BİNGÜL – 170517033**

**BEDİRHAN SARIHAN – 150119692**

**ONUR KAYA – 150119860**

**EMRE SAĞIROĞLU - 150119766**

**CSE 3015 – PROJECT REPORT**

**Project Detail:**

In this project,we designed a processor in logisim.This processor have 18 bits address and 18 bits data width.That will contain AND, OR, ADD, LD, ST, ANDI, ORI, ADDI, XOR, XORI, JUMP, BEQ, BGT, BLT, BGE, BLE.

AND instruction’s form is “AND DST,SRC1,SRC2”. DST, SRC1 and SRC2 are the registers. DST = SRC1 AND SRC2.

OR instruction’s form is “OR DST,SRC1,SRC2”. DST, SRC1 and SRC2 are the registers. DST = SRC1 OR SRC2.

ADD instruction’s form is “ADD DST,SRC1,SRC2”. DST, SRC1 and SRC2 are the registers. DST = SRC1+SRC2.

LD instruction’s form is “LD DST,ADDR”. DST is the register. ADDR is the 10 bits address of the data which will be taken from data memory. It gets the data in the suitable data and writes that data to DST register.

ST instruction’s form is “ST SRC,ADDR”. DST is the register. ADDR is the 10 bits address of the memory which will be store data from register. It gets data in the DST and writes that data to suitable memory address.

ANDI instruction’s form is “ANDI DST,SRC1,IMM”. DST and SRC1 are the registers. IMM value have 6 bits. DST = SRC1 AND IMM.

ORI instruction’s form is “ANDI DEST,SRC1,IMM”. DST and SRC1 are the registers. IMM value have 6 bits. DST = SRC1 OR IMM.

ADDI instruction’s form is “ADDI DST,SRC1,IMM”. DST and SRC1 are the registers. IMM value have 6 bits. DST = SRC1 + IMM.

XOR instruction’s form is “XOR DST,SRC1,SRC2”. DST, SRC1 and SRC2 are the registers. DST = SRC1 or SRC2.

XORI instruction’s form is “XORI DST,SRC1,IMM”. DST, SRC1 and SRC2 are the registers. IMM value have 6 bits. DST = SRC1 or IMM.

JUMP instruction’s form is “X ADDR”. X is our jump instruction.ADDR is 10 bits address.If our X value is jump,ADDR will added to PC value directly.

BEQ instruction’s form is “OP1,OP2,ADDR”. OP1 and OP2 are the registers. If OP1 and OP2 are equal,PC will be set to ADDR (PC-relative).

BGT instruction’s form is “OP1,OP2,ADDR”. OP1 and OP2 are the registers. If OP1 is greater than OP2,PC will be set to ADDR (PC-relative).

BLT instruction’s form is “OP1,OP2,ADDR”. OP1 and OP2 are the registers. If OP1 is less than OP2,PC will be set to ADDR (PC-relative).

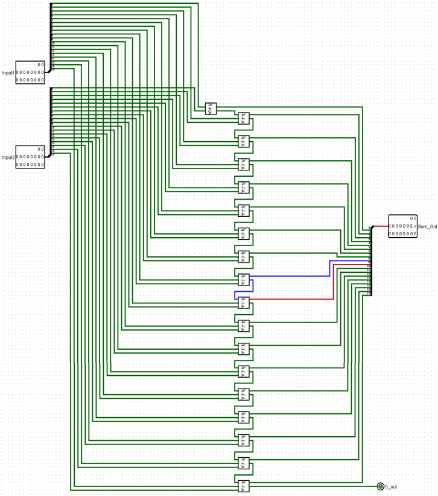
BGE instruction’s form is “OP1,OP2,ADDR”. OP1 and OP2 are the registers. If OP1 is greater than or equal to OP2,PC will be set to ADDR (PC-relative).

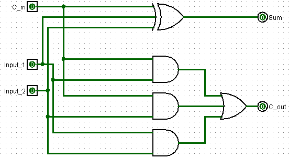
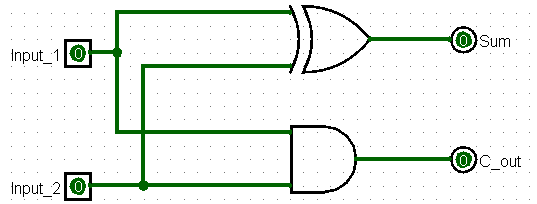
BLE instruction’s form is “OP1,OP2,ADDR”. OP1 and OP2 are the registers. If OP1 is less than or equal to OP2,PC will be set to ADDR (PC-relative).

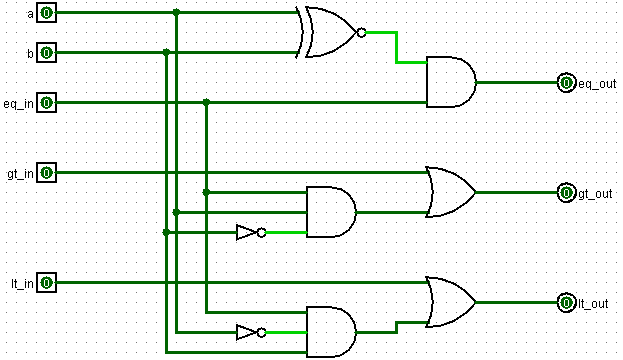
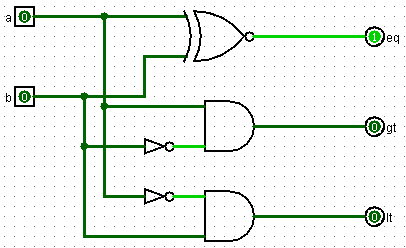
**STEP1 : ASSEMBLY LANGUAGE**

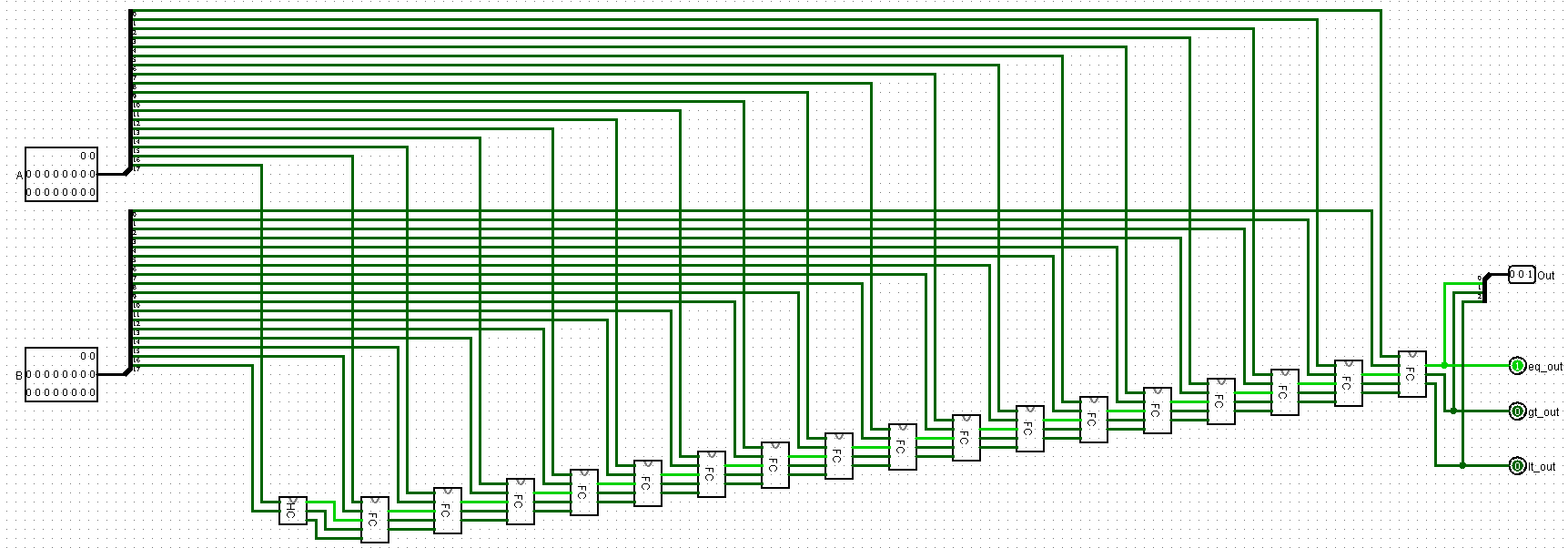
In this part of our project, we have created an ISA, which is the general plan schedule of our project.In order to perform our transactions in this ISA, we used a 4-bit opcode[17:14].However, we have created a 6-bit field for the immediate value.Finally, we have created a 10-bit field for the address part. After creating our ISA section,we created an assembler to translate this assembly language into 18-bit hexadecimal values.We designed this assembler with the python program, which is the language of our choice.At the end of this process, it transmits a txt file that we will send as an input value back to us as an output txt file translated into the corresponding hexadecimal values.

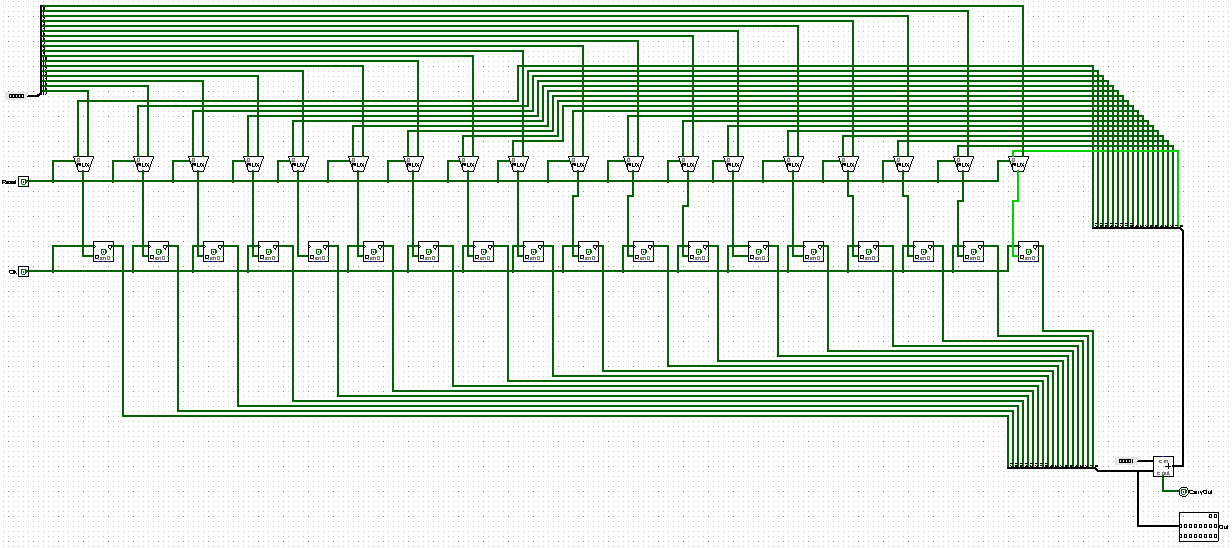
**STEP2:LOGISIM COMPONENTS**

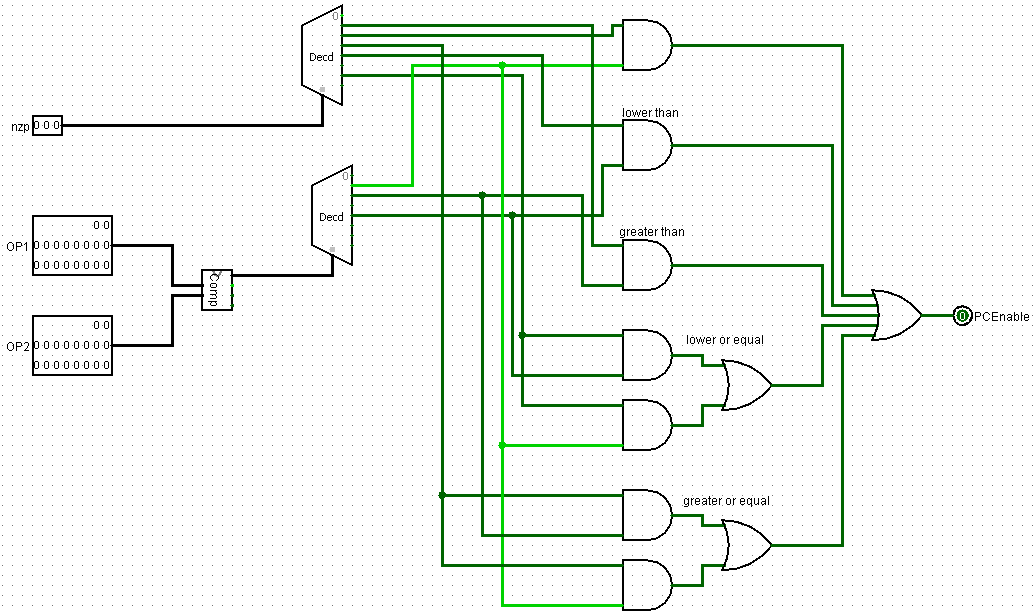
 **1-Bit Full Adder** **1-Bit Half Adder 18 Bit Adder**

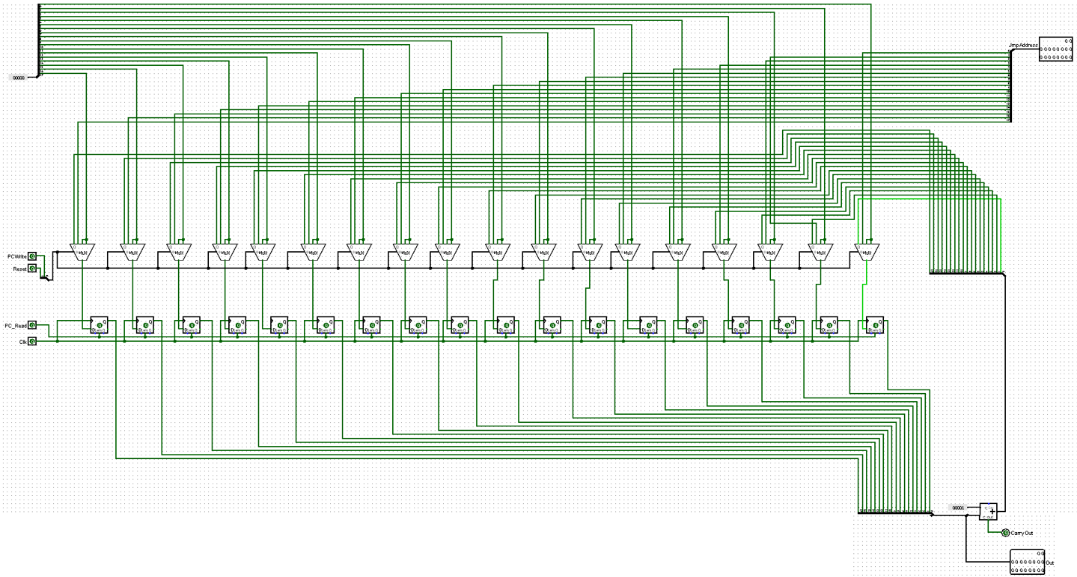
****

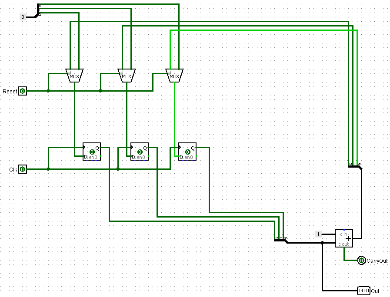
 **Full Comparator Half Comparator**

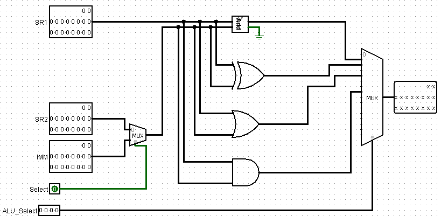
**18 Bits Comparator**

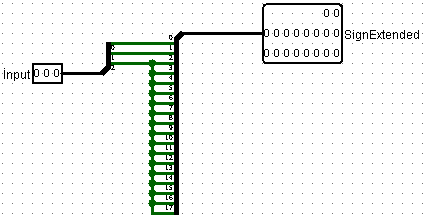
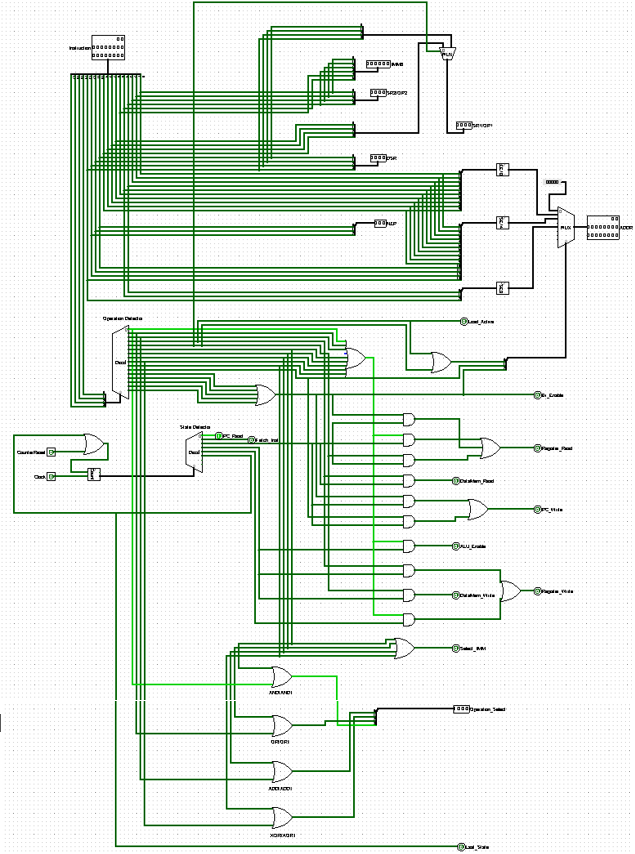
 **Counter**

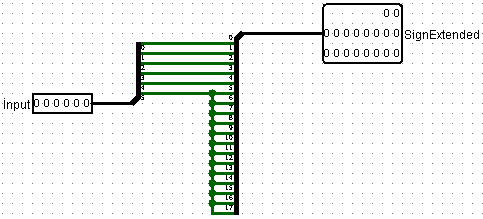
 **Branch**

 **Program Counter**

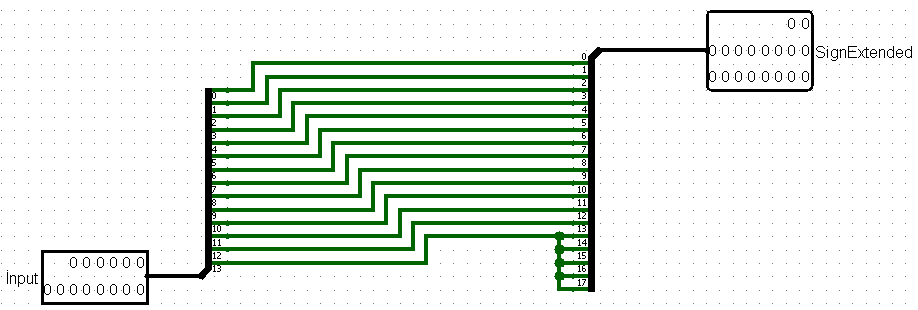
**Counter 3**

**ALU**

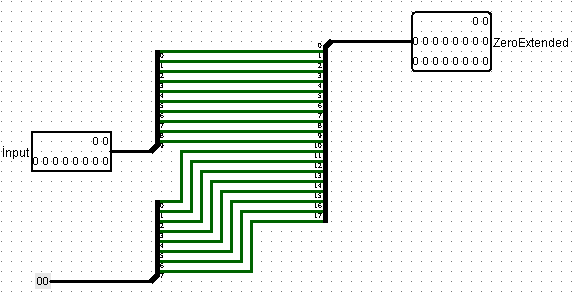
**Control Unit Sign Extender 3**

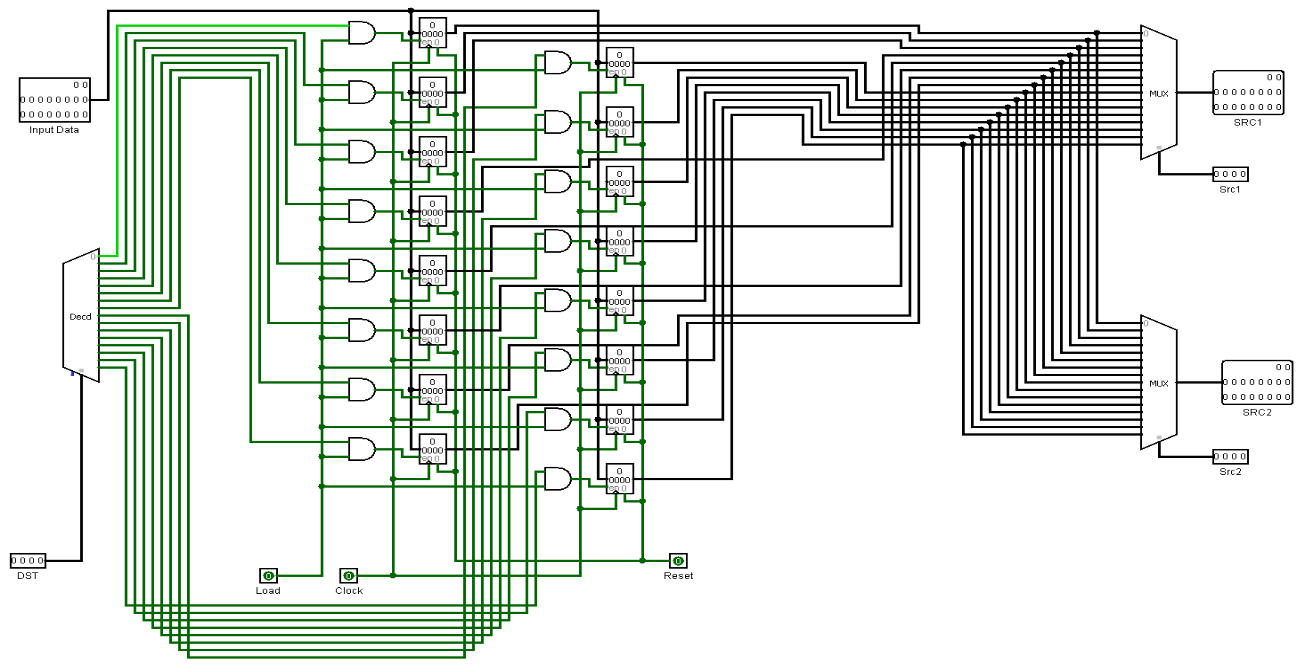
****

**Sign Extender 6**

****

**Sign Extender 14**

 **Zero Extender**

 **Register File**

**STEP3 : DESIGN WITH CONTROL UNIT**

In this part, as the first stage, we first perform the reading process from the PC.Then, after our reading process is completed, we get all our instructions. At the next stage, if our chosen instruction is ld, this instruction will read the value from the data memory at the next stage and write it to the register at the next stage. If our chosen instruction is st, this instruction will perform a read operation on the register at the next stage and write this value to the data memory at a later stage.If our chosen instruction is jump, at the next stage, the program counter write holds the address of the command that will be used after that. If AND,OR,ADD,ANDI,ORI,ADDI,XOR,XORI is one of the instructions we have selected, the reading process is performed first via the register, at a later stage, the values read from the corresponding register are passed through the logic process, at a later stage, this result is printed to the desired register. If BEQ,BGT,BLT,BGE,BLE is one of the instructions we choose, firstly, the reading from the register is done, in the next step the values ​​read from the registers are compared in the branch, in the next step, if the nzp bits are compatible with the comparison result, pc enable is activated,then the address of the pc will be set as the given adress. After the last stage is completed, it returns to pc read. This is the last stage of all operations.

