CSE 315 DIGITAL LOGIC DESIGN
TERM PROJECT

REPORT

Ahmet Akıl 150118038

Fatih Emin Öge 150118034

Zeynep Alıcı 150119517

Instruction Set Architecture and Assembler

AND	0	0	0	0	D	S	T	S	R	C1	S	R	C2			
ADD	0	0	0	1	D	S	T	S	R	C1	S	R	C2			
LD	0	0	1	0	D	S	T	A	D	D	R	E	S	S		
ST	0	0	1	1	S	R	C	A	D	D	R	E	S	S		
ANDI	0	1	0	0	D	S	T	S	R	C	I	M	M			
ADDI	0	1	0	1	D	S	T	S	R	C	I	M	M			
CMP	0	1	1	0	O	P	1	O	P	2						
JUMP	0	1	1	1	S	A	D	D	R	E	S	S				
JE	1	0	0	0	S	A	D	D	R	E	S	S				
JA	1	0	0	1	S	A	D	D	R	E	S	S				
JB	1	0	1	0	S	A	D	D	R	E	S	S				
JBE	1	0	1	1	S	A	D	D	R	E	S	S				
JAE	1	1	0	0	S	A	D	D	R	E	S	S				

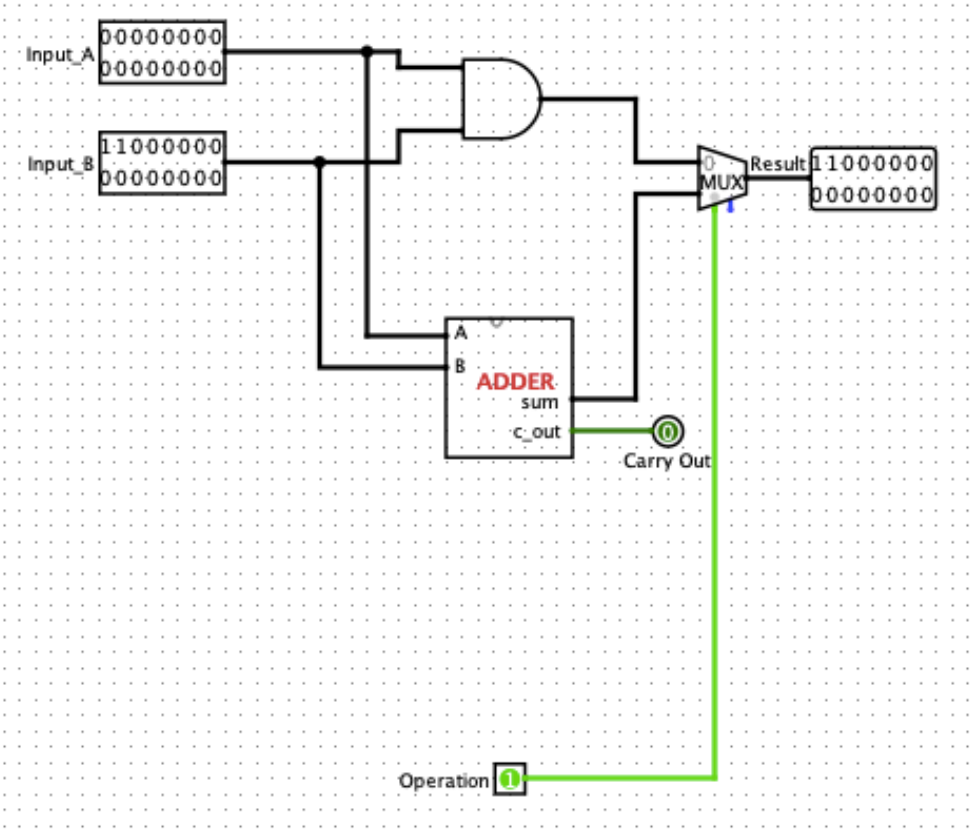
We have 13 instructions. So, 4 bits are enough to indicate all instructions in our instruction set architecture. We indicate 8 general-purpose registers with 3 bits. These general-purpose registers also exist in Register-File component of our CPU.

JE, JA, JB, JBE, JAE: S + ADRESS is an offset value. Offset values consist of 11 bits because of negative values. S is a sign bit.

After, we designed our instruction set, we implemented assembler with Python. Assembler converts our instructions to binary codes.

Arithmetic Logic Unit (ALU)

Arithmetic Logic Unit (ALU) handles aritmetic and logical operations. According to our instruction set, ALU perfroms AND, ADD, ANDI, ADDI operations.

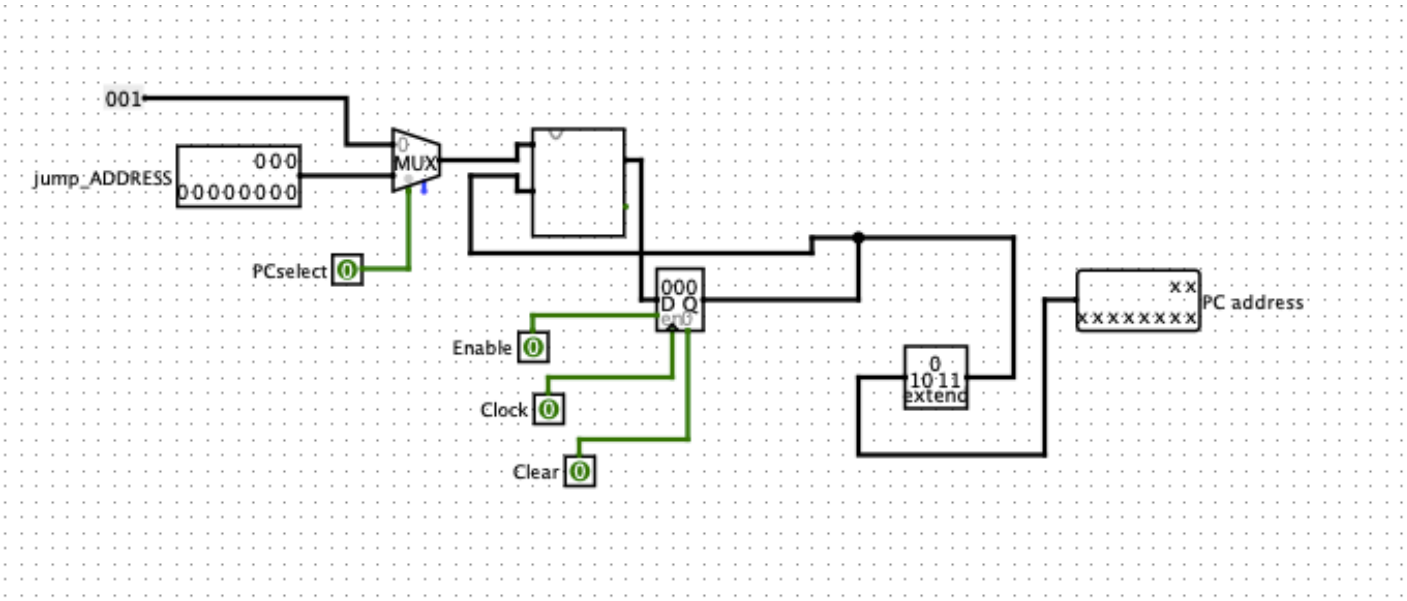


Multiplexor selects operation by select bit (Operation).
If Operation is 0, AND operation works otherwise ADD operation works.

According to our architecture, inputs consist of 16 bits. Also, adder composed of fifteen full adders and a half adder.

Program Counter Register

PC register is designed to track instructions and store the value of program counter (PC).

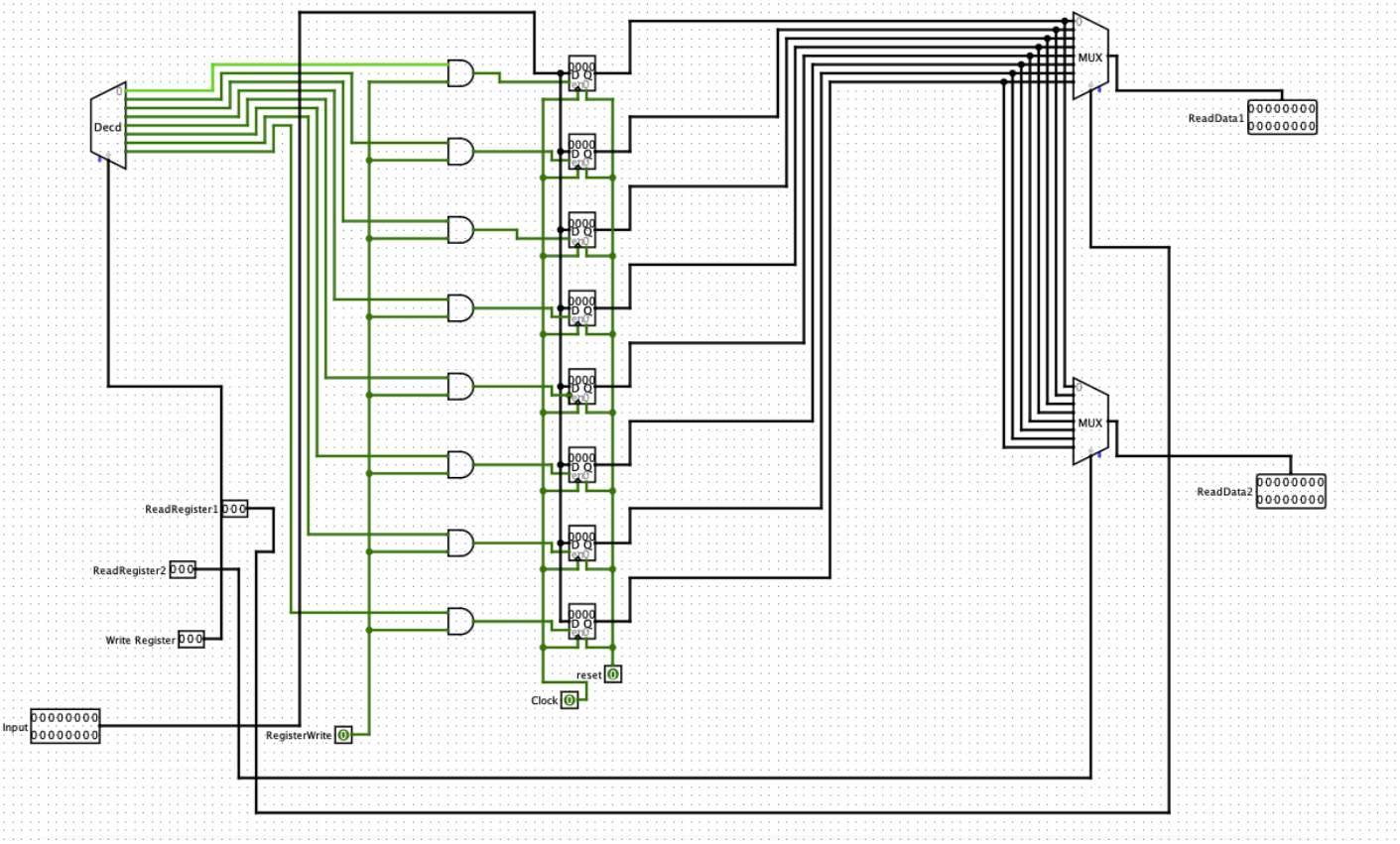


Jump and conditional jump instructions increases or decreases PC by address offset. All other instructions increase PC value by 1. For that purpose, we have PCselect signal. This signal generated by Control Unit.

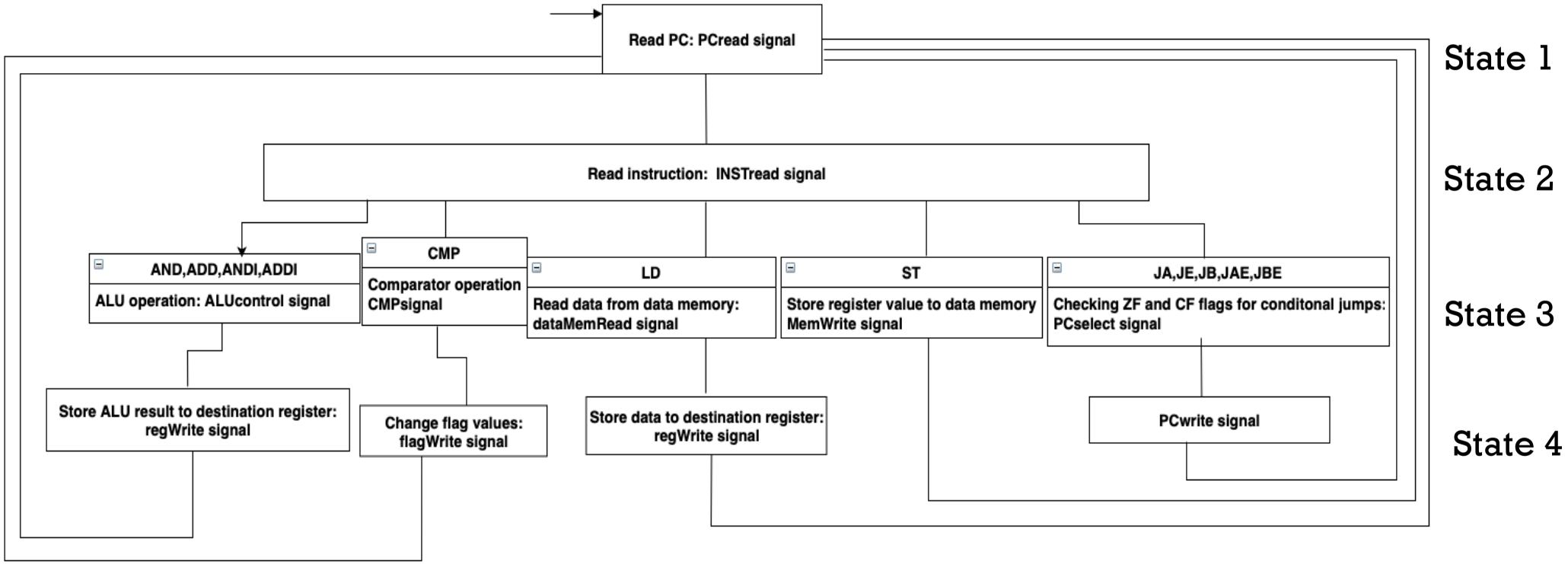
Our memory addresses have 10 bits but address offset value of consist of 11 bits because of two's complement. Therefore, there is a bit extender.

Register File

Register file includes all of eight general-purpose registers. (R0, R1, R2, R3, R4, R5, R6, R7). They can be used by all of instructions need to write to a register or read from a register. For example, AND, AND, ADDI, ANDI, CMP, ST, LD instructions need register file to write to or read from. The values of two registers can be read and can be write input data to one of them.



Finite State Machine

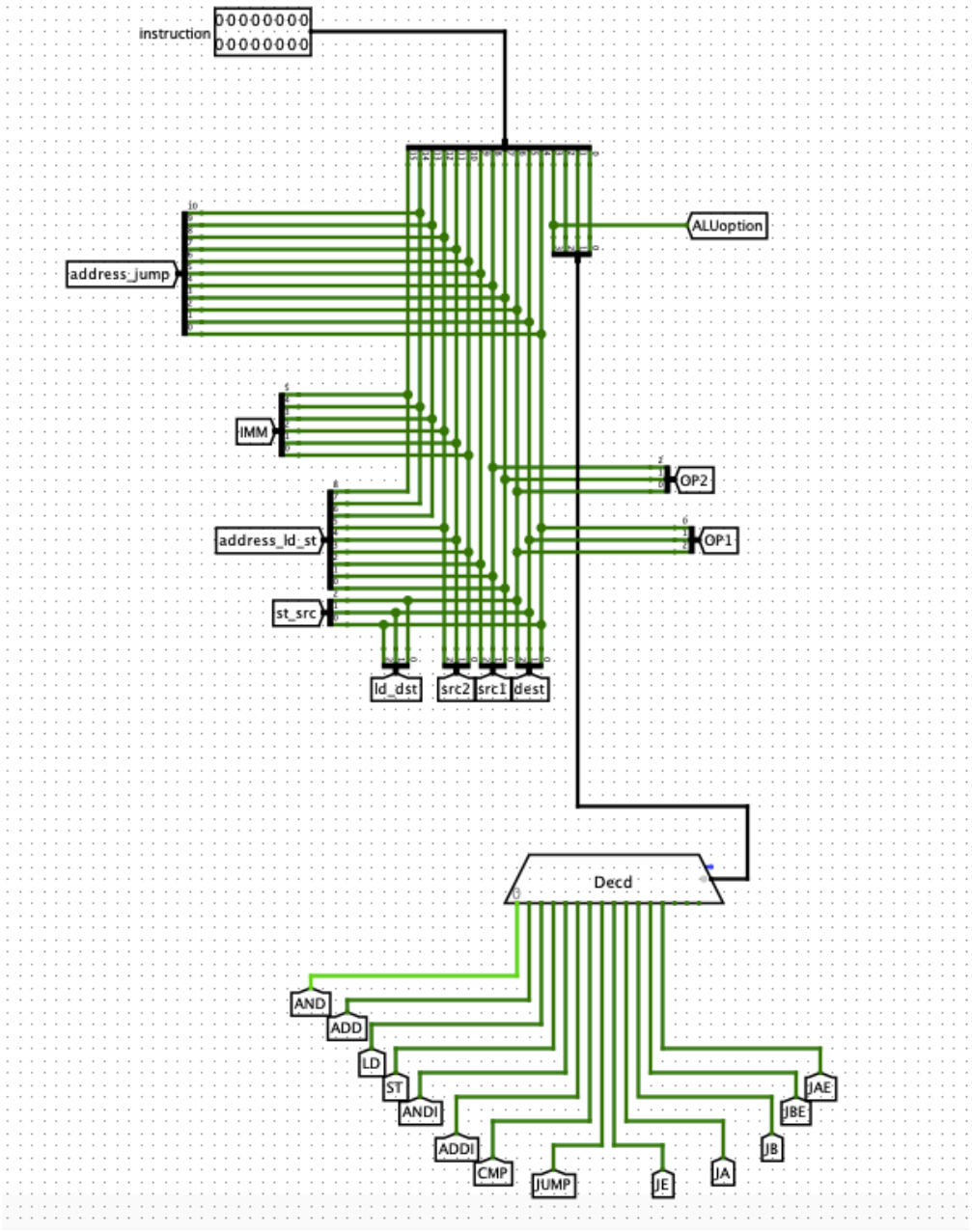


Control Unit and Signals

After creating the basic components of our project, we created the Control Unit by using Finite State Machine. Control units detects the instructions and generates all signals required for the processor.

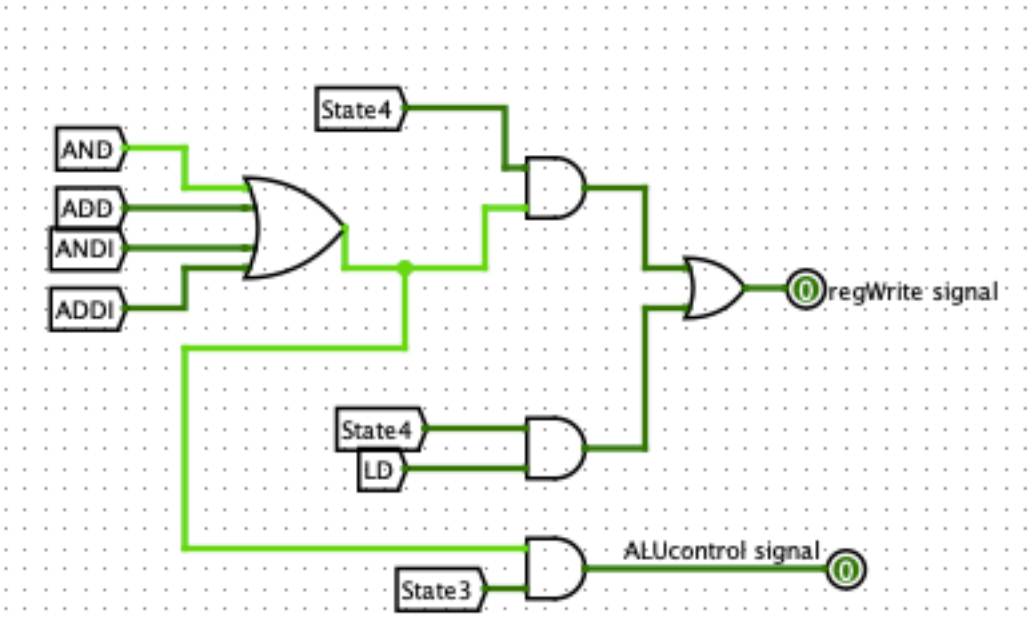
Parts of Control Unit

State 1 and State 2 are common for all instructions.
PCread signal generated at State 1 and then reads PC from CPU. And then INSTread signal generated at State 2. Then instructions reads from memory from CPU.

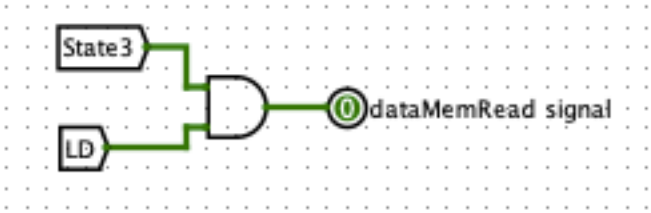


After instructions reads, Control unit detects instruction by decoder and then creates all information we need such as source register, destination registers, jump addresses.

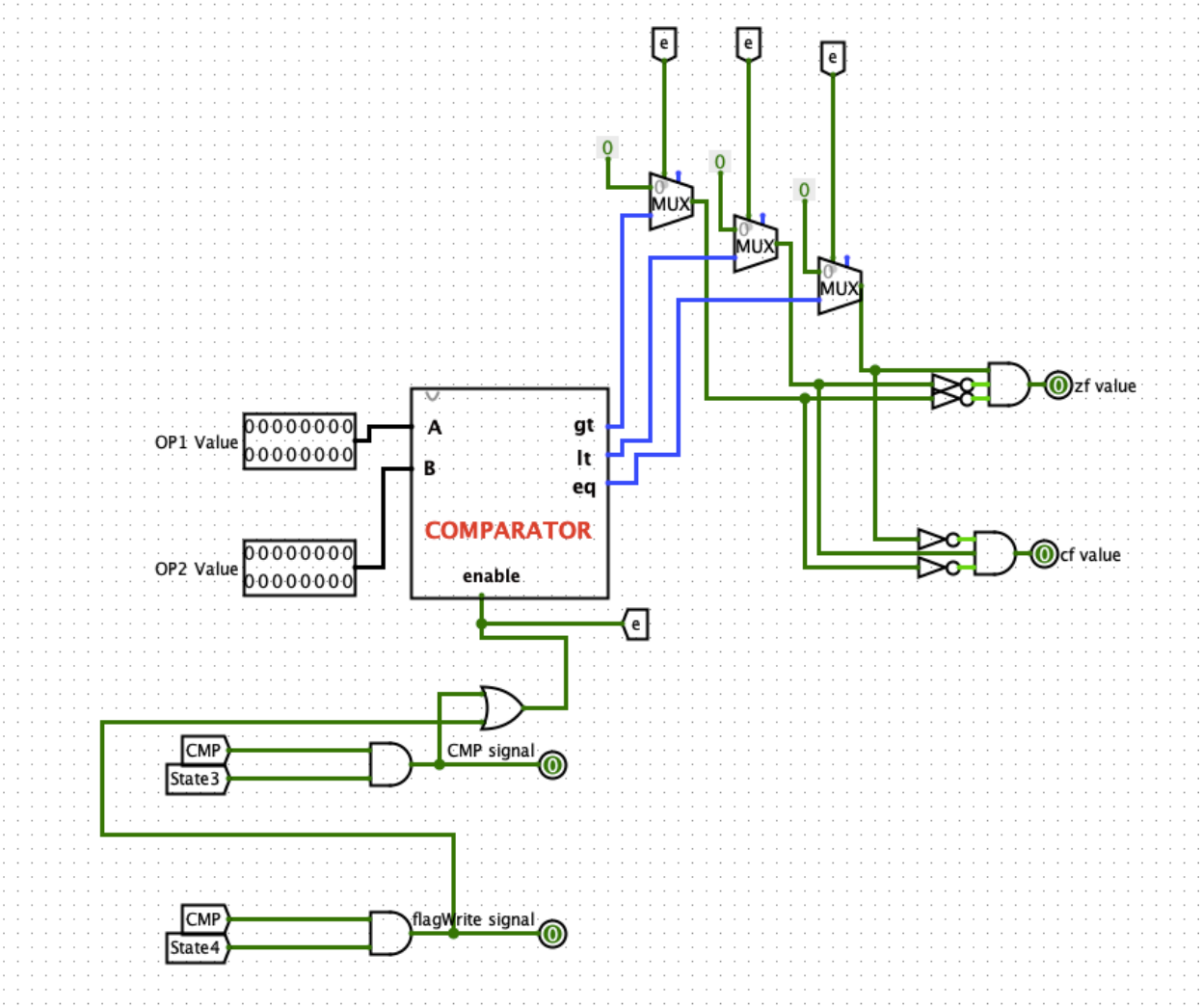
As from the third state, the states differ for instructions:



Contol Unit generates ALUcontrol signal at State 3 and then generate regWrite signal at State 4 for AND, ADD, ANDI, ADDI operations but also generates again regWrite signal at state 4 for LD instruction.

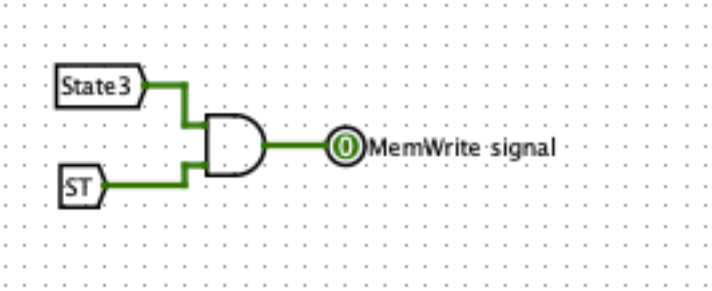


Before State 4, Control Unit generates dataMemRead signal for LD. Becaues LD instruction needs read the value from memory.

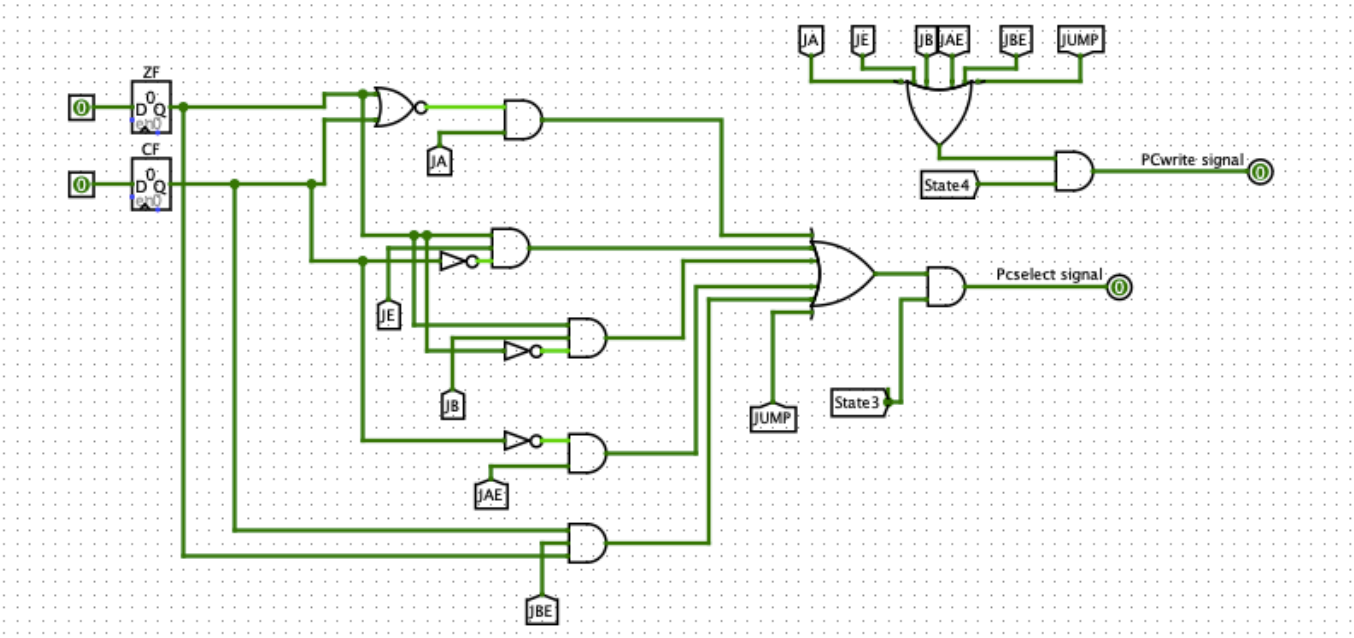


For CMP, CMPsignal is generated at State 3. And then read values of OP1 and OP2 registers from RegisterFile at the CPU. Then Comparator executes CMP operation and gives outputs such as gt, lt, eq.

At state 4 flagWrite signal is generated. Then according to outputs (gt, lt, eq) obtained at State 3, the values of CF an ZF flags are changed at CPU.

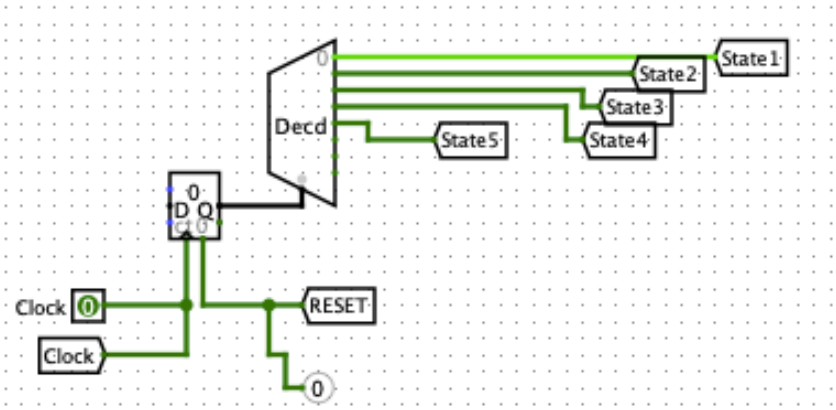


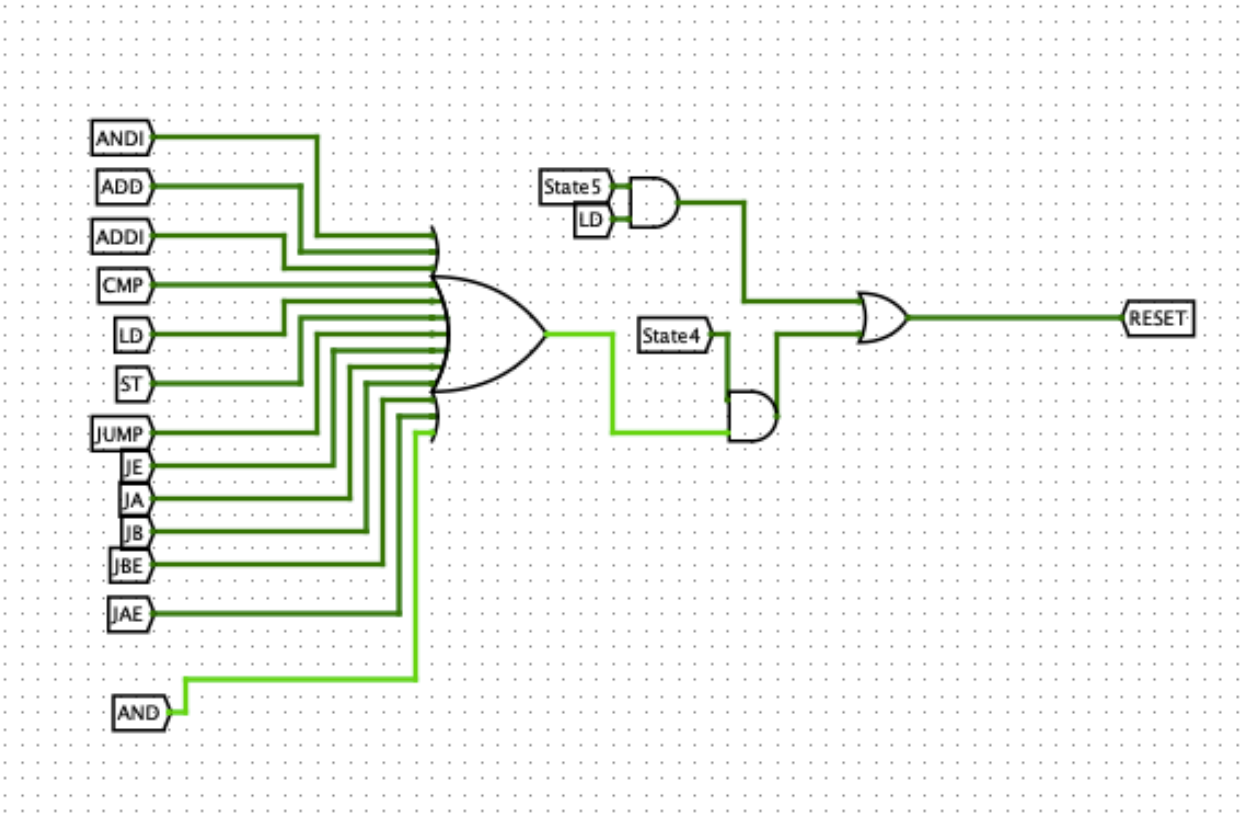
For ST, MemWrite signal is generated at State 3. Then writing is executed by using RegisterFile at CPU.



For conditional jumps we need values of ZF and CF at CPU. If conditions are true for conditional jumps, Pcselect signal is generated at State 3. This signal is a signal for PC. If this signal generated PC changed by address offset. At state 4 Pcwrite signal generated. And PC value is changed by address offset or increased by one.

How do we check which stage we are at?





After one of them instructions is ended, we reset the system.

COMPARATOR AND COMBINATIONAL LOGIC FOR FLAG VALUES

eq	gt	lt	ZF	CF
1	0	0	1	0
0	1	0	0	0
0	0	1	0	1

CF = NOT(gt)*NOT(eq)*lt

ZF = NOT(gt)*NOT(lt)*eq