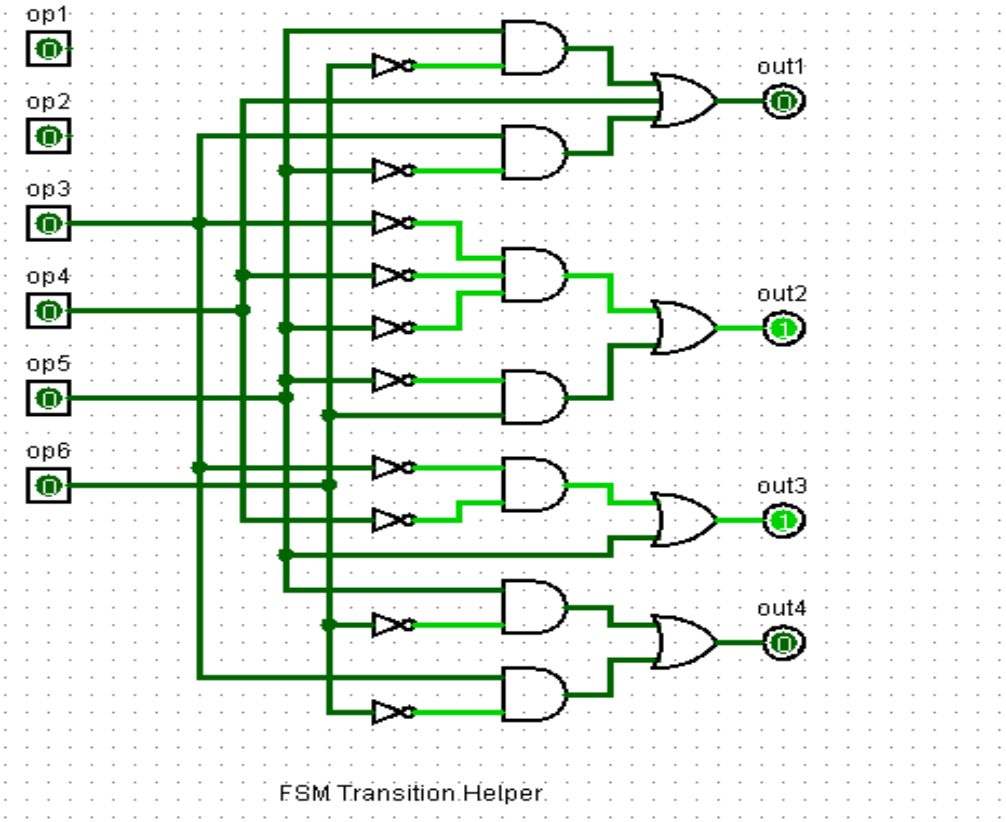
Computer Architecture CS322 Lab 9 Report

Name : Chandrawanshi Mangesh Shivaji
Roll Number : 1801CS16
Date : 13/11/2020

Task 1 Study the given multi-cycle implementation of the processor, and identity error in the design (if any)
Sol :-

There was error in the **control circuit** of the given file, the circuit was not correct according to the mentioned state transitions. So, I had to analyze the circuit once again correctly according to the state transitions. I created **an FSM transition helper** circuit and rectified the error.

FSM Transition Helper



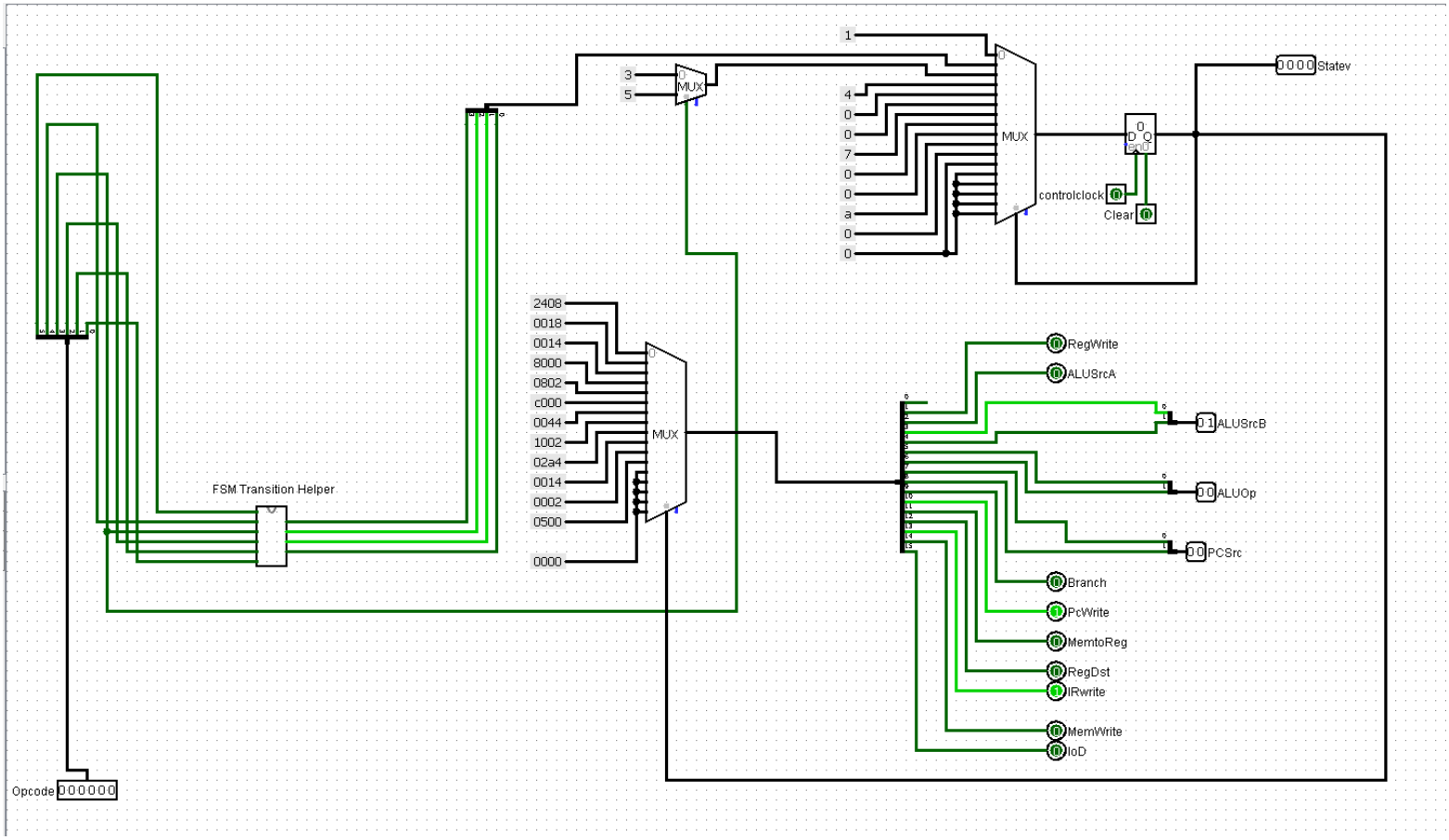
out1 = op5 ~op6 + op4 + op3 ~op5

out2 = ~op3 ~op4 ~op5 + ~op5 op6

out3 = ~op3 ~op4 + op5

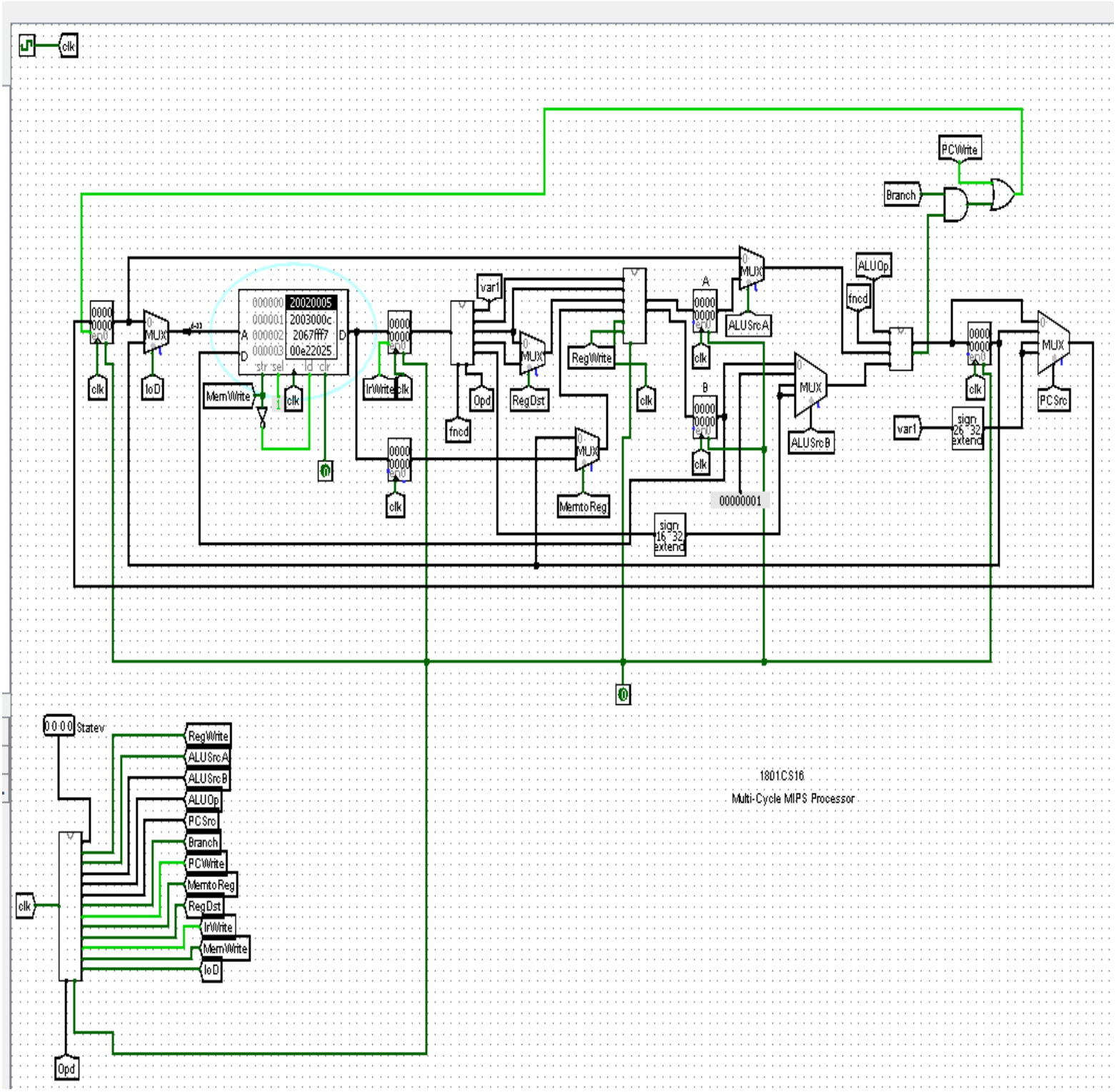
out4 = op5 ~op6 + op3 ~op6

Rectified Control Circuit

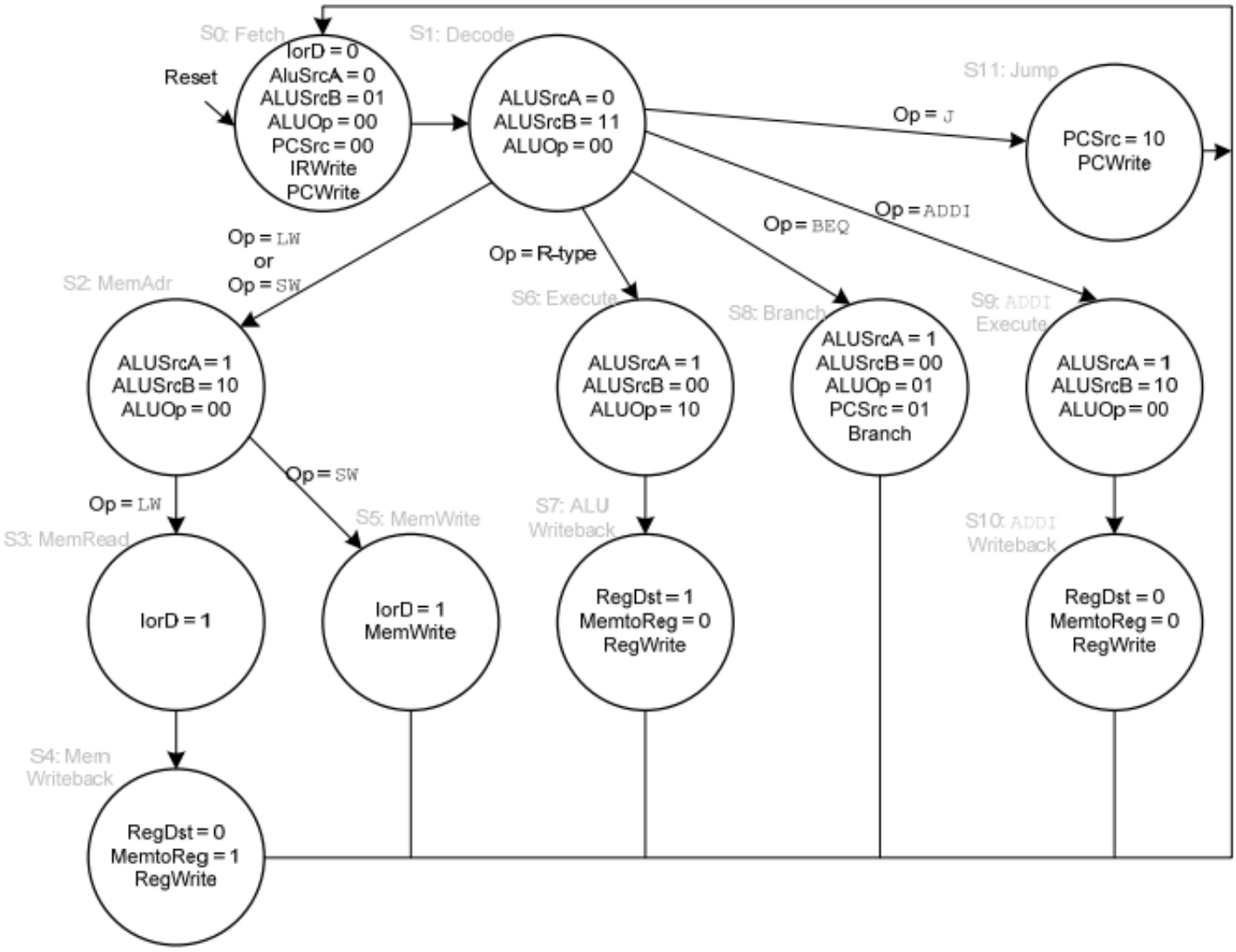


Operation	Opcode	Fsm state transition	Mux value(for 2)
lw	100011	0010	0
sw	101011	0010	1
R-type	000000	0110	x
j	000010	1011	x
beq	000100	1000	x
addi	001000	1001	x

Final Multi-Cycle Processor



Finite State Machine



This is the FSM for which above MIPS multi-cycle processor is designed

Submissions :

Filename : 1801CS16_Lab9_task1.circ

Sample Loadfile : loadfile_task1.dat

Task 2: Study the given multi-cycle implementation of the processor and test using your own test program (for eg. Sum of 5 numbers using basic instruction; create a new mem.dat)

Sol :-

Test Program : Sum of 5 Numbers

```
# Test the MIPS multi cycle Processor Task 2
# Write a program to add 5 numbers and store the result in data memory location X
# First data (5(05H),22(16H),16(10H),34(22H),48(30H)) [all in decimal rep.] is stored in memory location 50H to 54H sequentially
# sum will be stored in register $6 and then at memory location 96 i.e. 60H

#       Assembly        Description             Address     Machine


        lw $1,80($0)    # $1 = 5                0           8c010050
        lw $2,81($0)    # $2 = 22               4           8c020051
        lw $3,82($0)    # $3 = 16               8           8c030052
        lw $4,83($0)    # $4 = 34               12          8c040053
        lw $5,84($0)    # $5 = 48               16          8c050054
        addi $6,$0,$0   # $6 = 0                20          20060000
        add $6,$1,$2    # $6 <= 5 + 22 = 27     24          00223020
        add $6,$6,$3    # $6 <= 27 + 16 = 43    28          00c33020
        add $6,$6,$4    # $6 <= 43 + 34 = 77    32          00c43020
        add $6,$6,$5    # $6 <= 77 + 48 = 125   36          00c53020
        sw $6,96($0)    # write addr 96 = 125   40          ac060060

# mem addr 60H(96) => 7DH(125)
```

Input / Output / Machine Codes

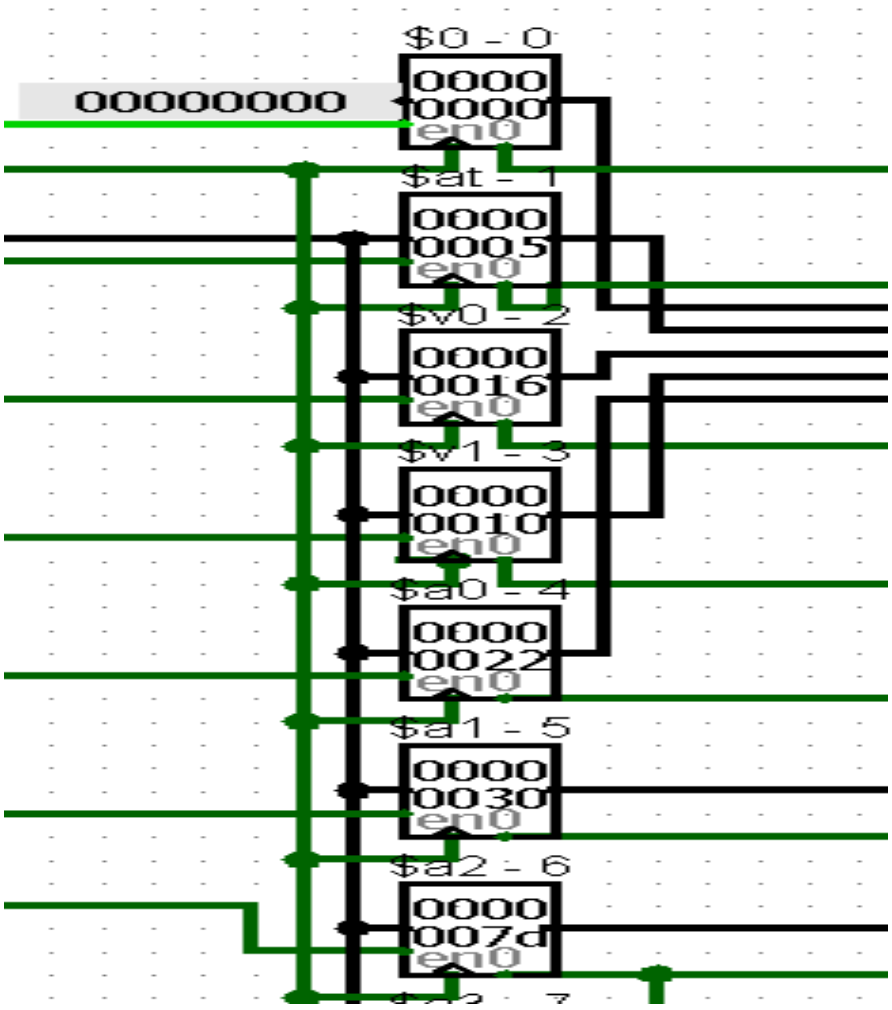
```
000000 8c010050 8c020051 8c030052 8c040053 8c050054 20060000 00223020 00c33020 00c43020 00c53020 ac060060
000010 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000030 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000050 00000005 00000016 00000010 00000022 00000030 00000000 00000000 00000000 00000000 00000000 00000000
000060 0000007d 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Output Sum

Input must be done manually by the checker

Same MIPS multi-cycle processor as Task1 is used and is sufficient to run the program

Final State of Used Registers



Submissions :

Filename : 1801CS16_Lab9_task2.circ

Loadfile : loadfile_task2.dat

Test Program : 1801CS16_Lab9_task2_code.txt

Task 3 Add one new instructions to the given architecture and test using new test program.

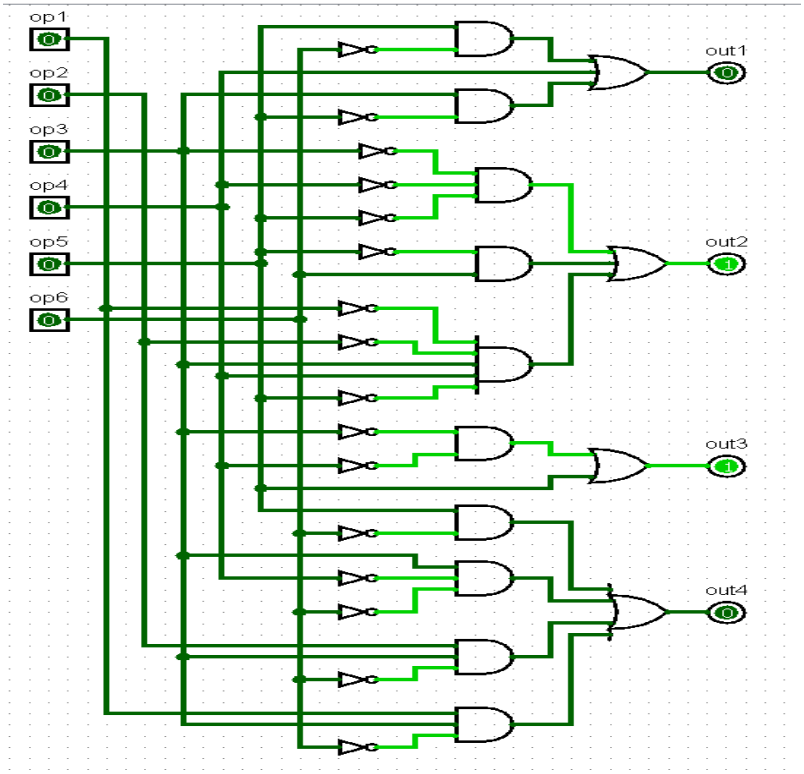
Sol : I have added **andi** instruction as a new instruction opcode = 001100

This is an I type instruction. Example andi $s1, $s4, 0x0045 ($s1= content of $s4 and 0x0045)

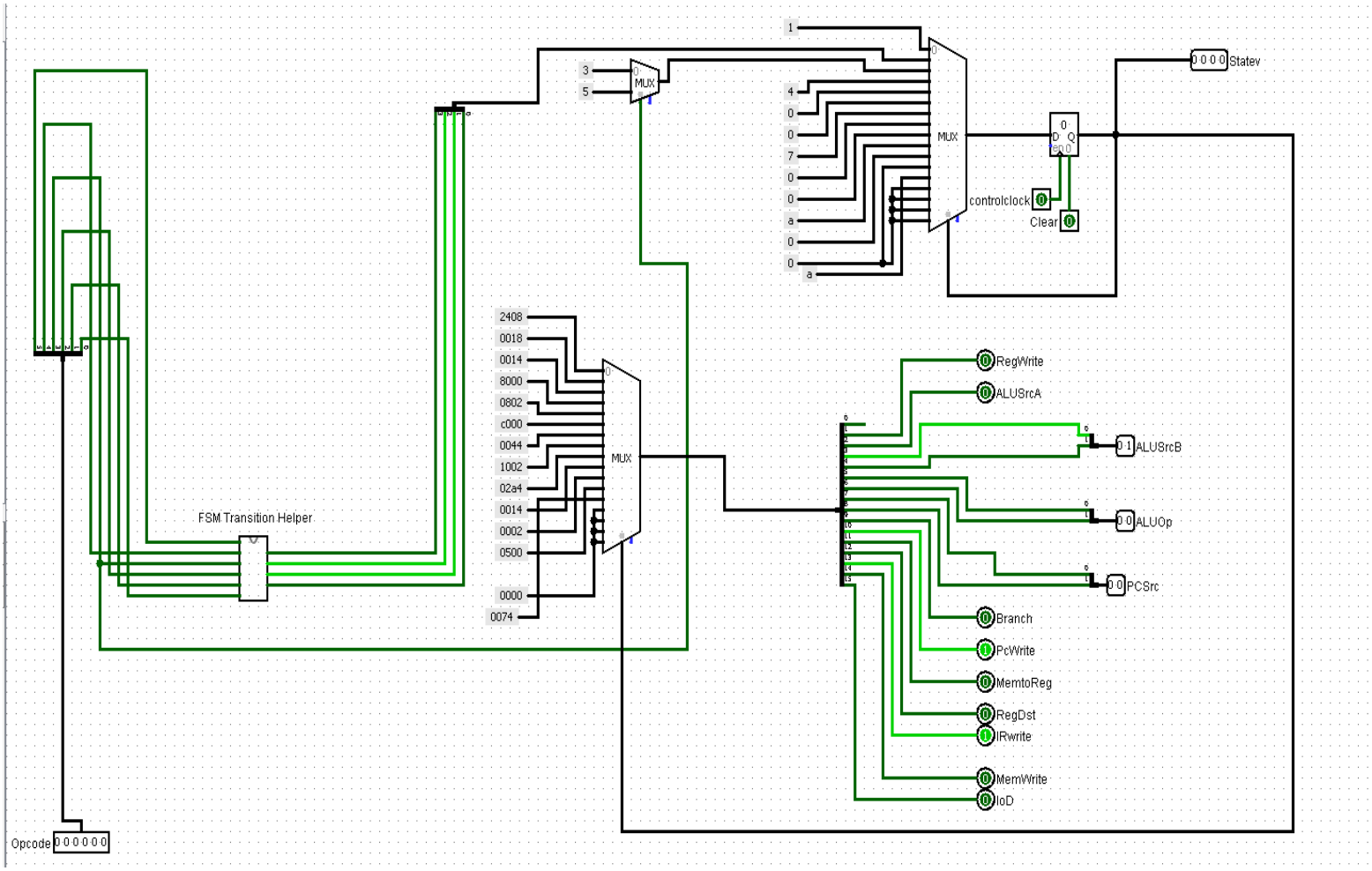
Modified Table after adding addi instruction

Operation	Opcode	Fsm state transition	Mux value(for 2)
lw	100011	0010	0
sw	101011	0010	1
R-type	000000	0110	x
j	000010	1011	x
beq	000100	1000	x
addi	001000	1001	x
andi	001100	1100	x

FSM Controller

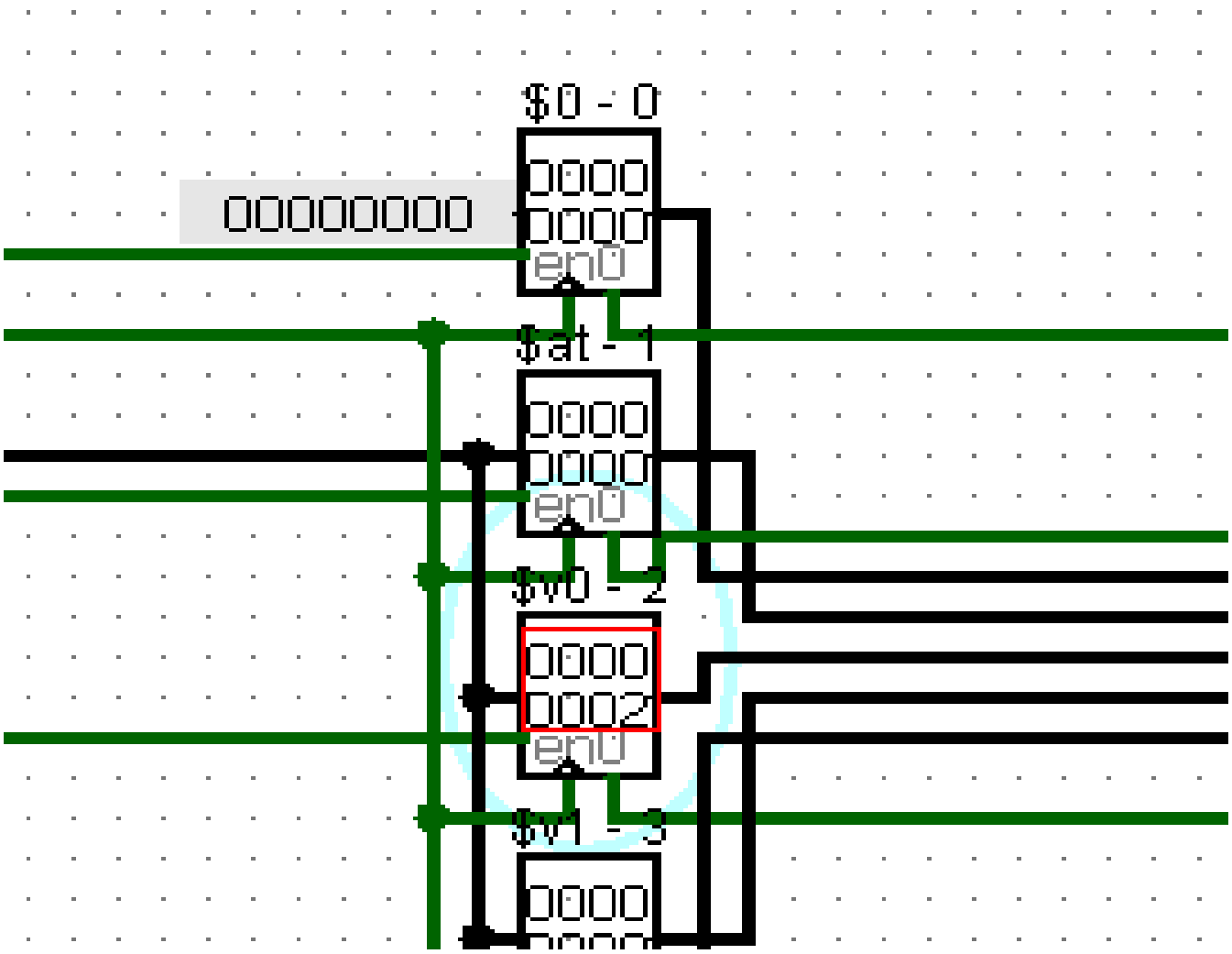


Now we can run our controller and check that state is transitioning to state 12 and then to state 10 and then back to state 0. Now for the fsm state we write value a (for next state) and for the control signals we need ALUSrca=1 and ALUSrcb=10 and ALUop=11. Required value is 0x0074 for all correct control signals. Final Contol unit is as below




```
1
2  # Test the MIPS multi cycle Processor Task 3
3  # andi instruction
4
5  #       Assembly                Machine
6
7          addi $2, $0, 0x0002    2002000a
8          andi $2, $2, 0x000a    3403000a
9
10
```

In this sample program we first write $2 = 2 (0010 in binary)
In the second operation we do $2 = (contents of $2) and (a)
We get 0010 & 1010 = 0010 ie 0x0002
Encoding is stated above and Final Output is shown below



Submissions :

Filename : 1801CS16_Lab9_task3.circ

Loadfile : loadfile_task3.dat

Test Program : 1801CS16_Lab9_task3_code.txt