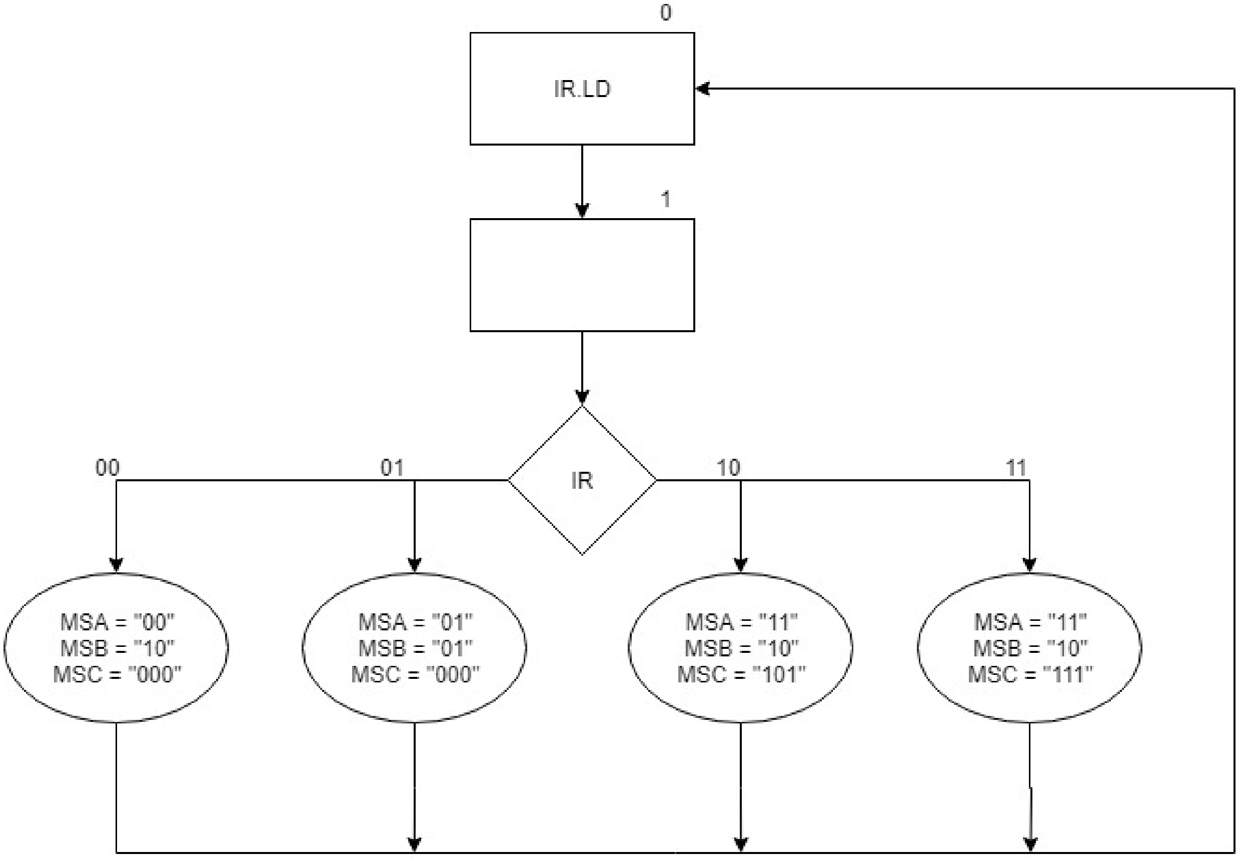
Part 1 Introduction: 1st ALU Controller

Part 1 ASM chart.



Part 1 next state truth table.

Q	IR_1	IR_0	IR.LD	MSA		MSB		MSC			Q+
0	*	*	1	0	1	1	0	0	0	0	1
1	0	0	0	0	0	1	0	0	0	0	0
1	0	1	0	0	1	0	1	0	0	0	0
1	1	0	0	1	1	1	0	1	0	1	0
1	1	1	0	1	1	1	0	1	1	1	0

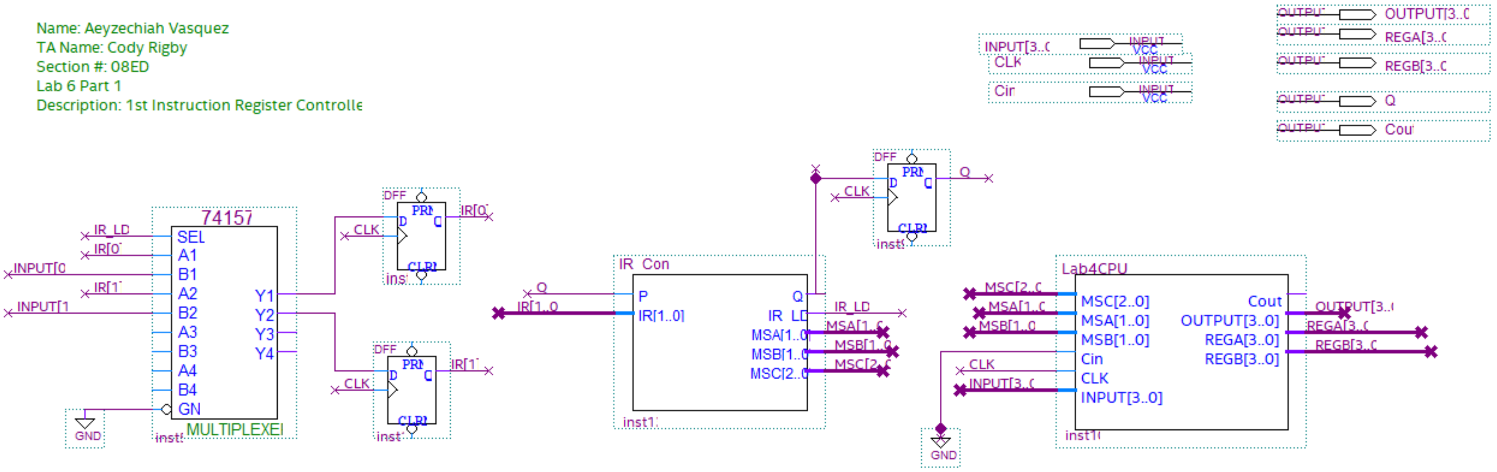
Part 1 VHDL.

```
--Name:    Aeyzechiah Vasquez
--Lab 6 Part 1
--Section #: 08ED
--TA Name: Cody Rigby
--Description: VHDL design of IR Controller

library ieee; use ieee.std_logic_1164.all;
```

```
entity IR_Cont is
     port(
           P: in std_logic;
           IR: in std_logic_vector(1 downto 0);
           Q,IR_LD: out std_logic;
           MSA, MSB: out std_logic_vector(1 downto 0);
           MSC: out std_logic_vector(2 downto 0));
end IR_Cont;

architecture behavior of IR_Cont is
     begin

     MSA <= "00" when
           (P='1' and IR="00")
           else "11" when
           (P='1' and IR="10") or
           (P='1' and IR="11")
           else "01";

     MSB <= "01" when
           (P='1' and IR="01")
           else "10";

     MSC <= "101" when
           (P='1' and IR="10")
           else "111" when
           (P='1' and IR="11")
           else "000";

     Q <= not P;
     IR_LD <= not P;

end behavior;
```

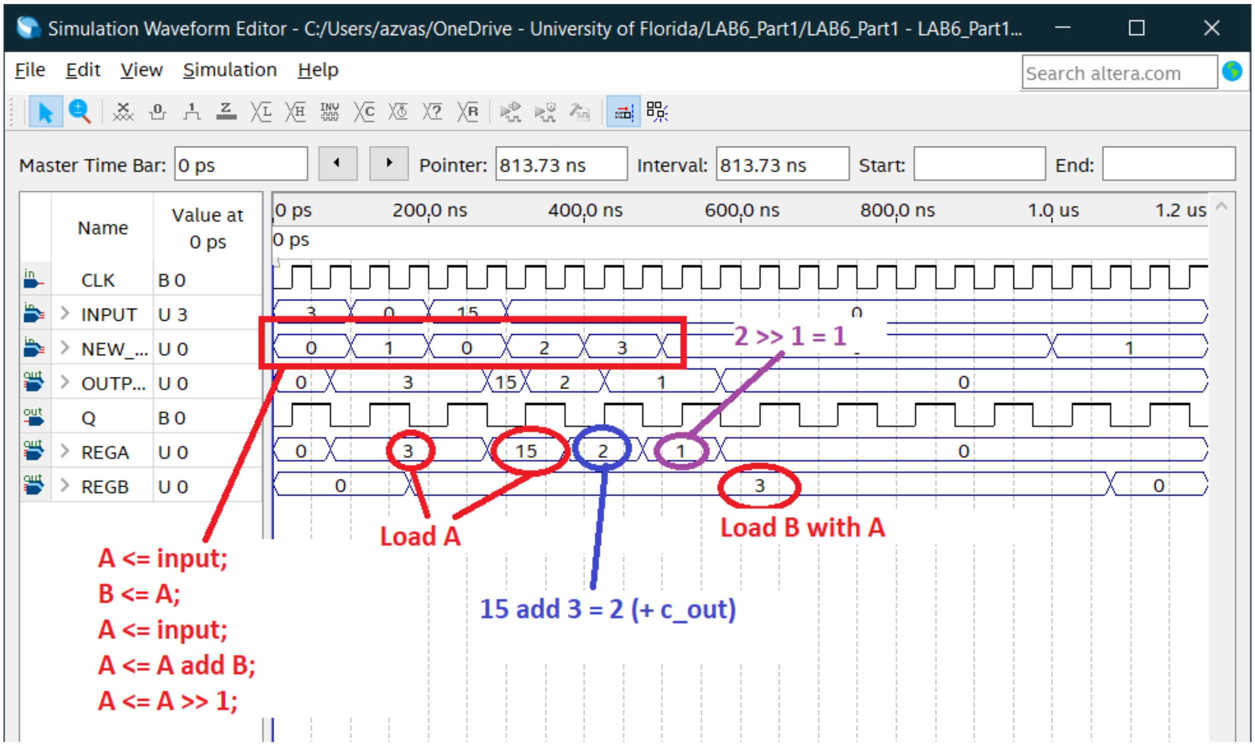
Vasquez, Aeyzechiah
Lab 6
EEL3701

Part 1 schematic.



Name: Aeyzechiah Vasquez
TA Name: Cody Rigby
Section #: 08ED
Lab 6 Part 1
Description: 1st Instruction Register Controlle

Part 1 simulation and annotation.



Simulation Waveform Editor - C:/Users/azvas/OneDrive - University of Florida/LAB6_Part1/LAB6_Part1 - LAB6_Part1...

Master Time Bar: 0 ps Pointer: 813.73 ns Interval: 813.73 ns Start: End:

Name	Value at 0 ps
CLK	B 0
INPUT	U 3
NEW_...	U 0
OUTP...	U 0
Q	B 0
REGA	U 0
REGB	U 0

A <= input;
B <= A;
A <= input;
A <= A add B;
A <= A >> 1;

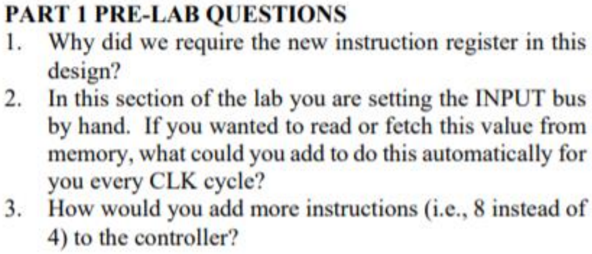
Load A

15 add 3 = 2 (+ c_out)

2 >> 1 = 1

Load B with A

Part 1 pre-lab questions.



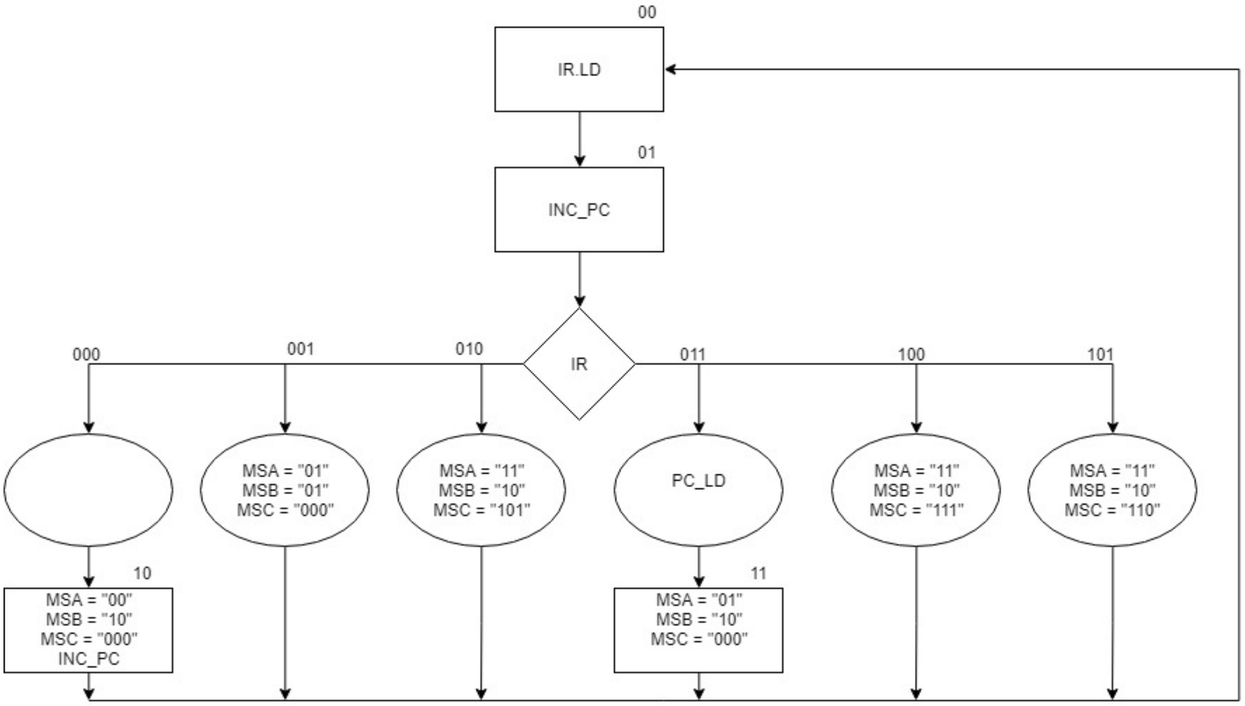
PART 1 PRE-LAB QUESTIONS

1. Why did we require the new instruction register in this design?
2. In this section of the lab you are setting the INPUT bus by hand. If you wanted to read or fetch this value from memory, what could you add to do this automatically for you every CLK cycle?
3. How would you add more instructions (i.e., 8 instead of 4) to the controller?

1. We required the new instruction register to automate the mux select lines.
2. I could connect the input bus to the output of a parallel-out register or to the data output of a 4-bit memory device such as a ROM or RAM.
3. Adding more instructions to the IR requires more bits. 8 instructions would require 3 bits. I change my 4-instruction IR to an 8-instruction IR by adding one more bit to the IR.

Part 2 Introduction: 2nd ALU Controller

Part 2 ASM chart.



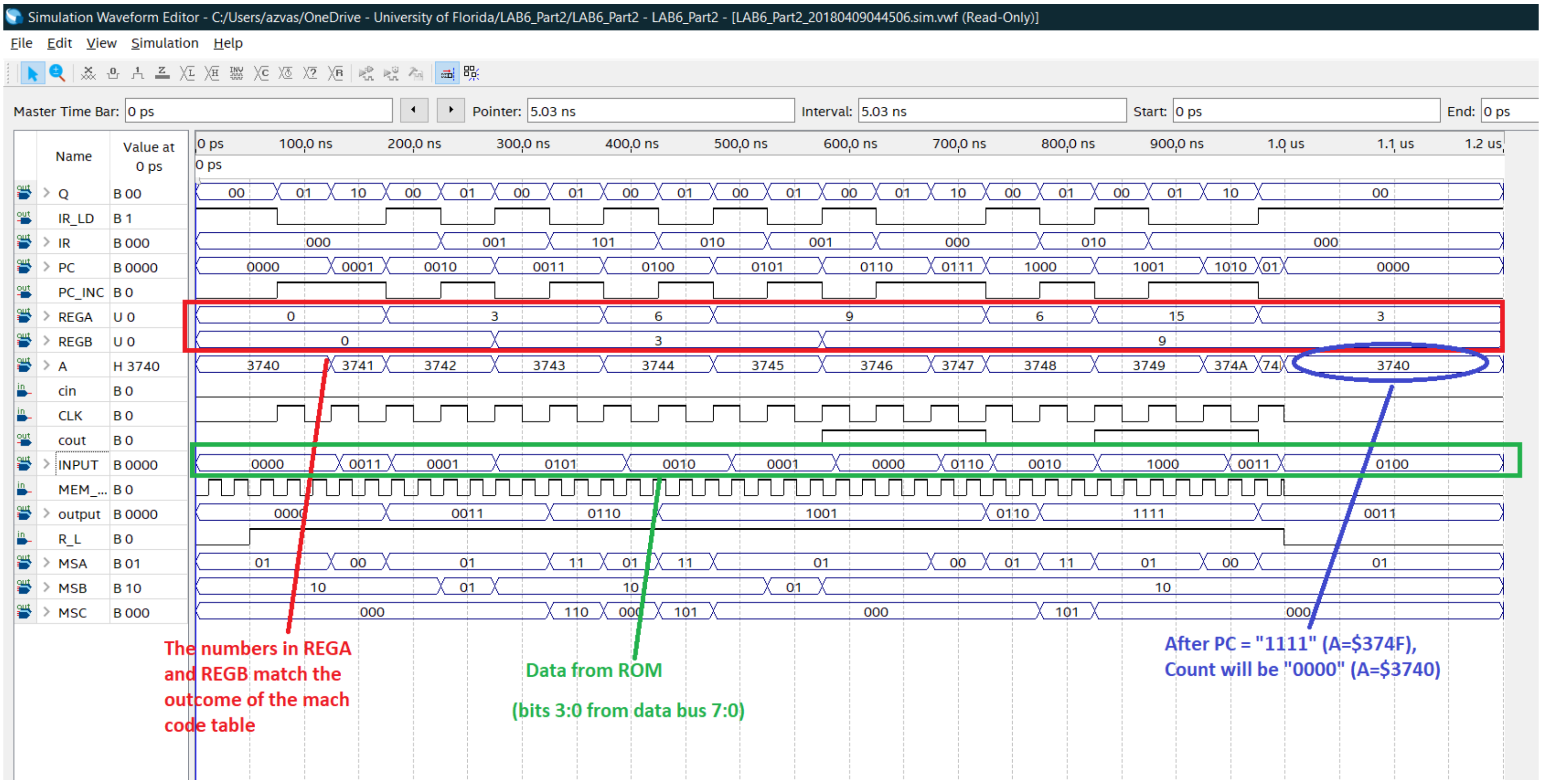
Part 2 next state truth table.

Q		IR			MSA		MSB		MSC			IR.LD	INC_PC	PC.LD	Q+	
0	0	*	*	*	0	1	1	0	0	0	0	1	0	0	0	1
0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1	0
0	1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	0
0	1	0	1	0	1	1	1	0	1	0	1	0	1	0	0	0
0	1	0	1	1	0	1	1	0	0	0	0	0	1	0	1	1
0	1	1	0	0	1	1	1	0	1	1	1	0	1	0	0	0
0	1	1	0	1	1	1	1	0	1	1	0	0	1	0	0	0
0	1	1	1	0	0	1	1	0	0	0	0	0	1	0	0	0
0	1	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0
1	0	*	*	*	0	0	1	0	0	0	0	0	1	0	0	0
1	1	*	*	*	0	1	1	0	0	0	0	0	0	1	0	0

Part 2 machine code assembling.

Addr		**Mach code**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
$3740	LDAA #3	%000	3	X						
$3742	TAB	%001	3	3						
$3743	SAL	%101	6	3						
$3744	ABA	%010	8	3	0	9	3	0	7	3
$3745	TAB	%001	8	9	0	0	3	3	7	7
$3746	LDAA #6	%000	6	9	6	0	6	3	6	7
$3748	ABA	%010	15	9	6	0	9	3	13	7
$3749	SAR	%100	7	9	3	0	4	3	6	7
$374A	JMP 4	%011	7	9	3	0	4	3	6	7
$374C	TAB	%001								

Part 2 simulation and annotation.



Simulation Waveform Editor - C:/Users/azvas/OneDrive - University of Florida/LAB6_Part2/LAB6_Part2 - LAB6_Part2 - [LAB6_Part2_20180409044506.sim.vwf (Read-Only)]

Master Time Bar: 0 ps
Pointer: 5.03 ns
Interval: 5.03 ns
Start: 0 ps
End: 0 ps

Name	Value at 0 ps
Q	B 00
IR_LD	B 1
IR	B 000
PC	B 0000
PC_INC	B 0
REGA	U 0
REGB	U 0
A	H 3740
cin	B 0
CLK	B 0
cout	B 0
INPUT	B 0000
MEM_...	B 0
output	B 0000
R_L	B 0
MSA	B 01
MSB	B 10
MSC	B 000

File Edit View Simulation Help
0 ps 100.0 ns 200.0 ns 300.0 ns 400.0 ns 500.0 ns 600.0 ns 700.0 ns 800.0 ns 900.0 ns 1.0 us 1.1 us 1.2 us
Q: 00 01 10 00 01 00 01 00 01 00 01 00 01 10 00 01 00 01 10 00
IR: 000 001 101 010 001 000 010 000
PC: 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 01 0000
REGA: 0 3 6 9 6 15 3
REGB: 0 3 9
A: 3740 3741 3742 3743 3744 3745 3746 3747 3748 3749 374A 74 3740
INPUT: 0000 0011 0001 0101 0010 0001 0000 0110 0010 1000 0011 0100
output: 0000 0011 0110 1001 0110 1111 0011
MSA: 01 00 01 11 01 11 01 00 01 11 01 00 01
MSB: 10 01 10 01 10
MSC: 000 110 000 101 000 101 000

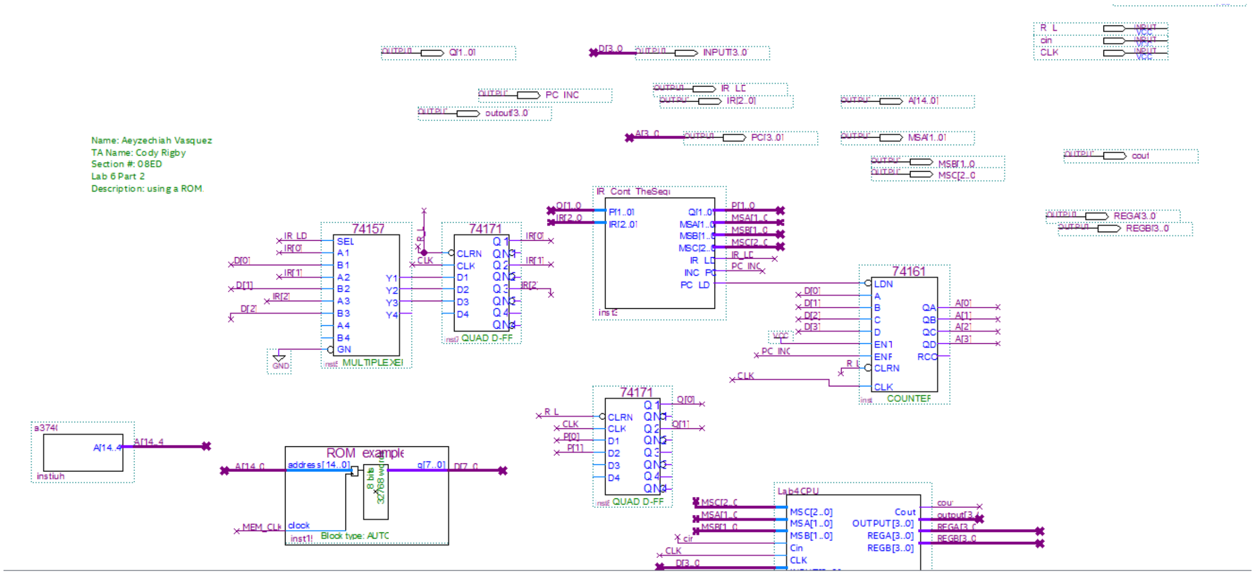
The numbers in REGA and REGB match the outcome of the mach code table

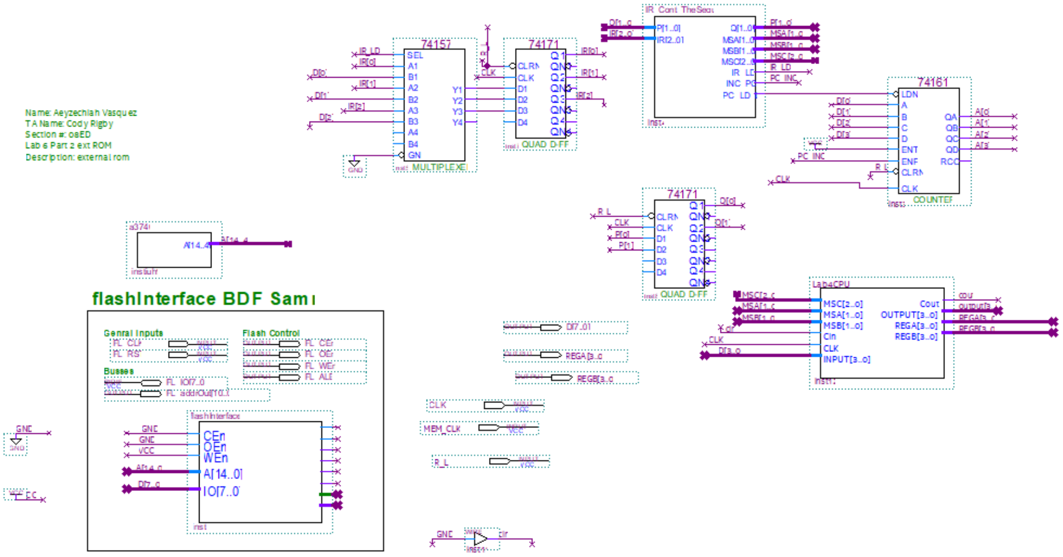
Data from ROM

(bits 3:0 from data bus 7:0)

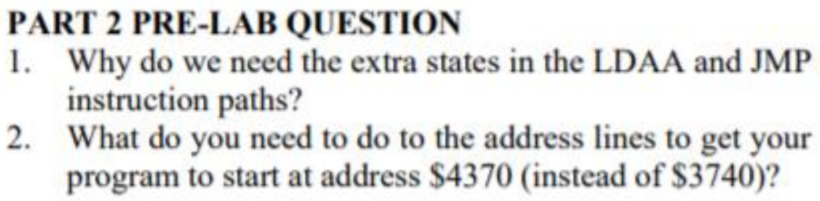
After PC = "1111" (A=$374F), Count will be "0000" (A=$3740)

Part 2 schematic.





Part 2 Pre-lab questions.



PART 2 PRE-LAB QUESTION

1. Why do we need the extra states in the LDAA and JMP instruction paths?
2. What do you need to do to the address lines to get your program to start at address $4370 (instead of $3740)?

1. Extra states are necessary because the operators require additional operands from the user and thus will need another address to be hold more data.
2. I would set the first 11 (most significant) address lines to “100 0011 0111.”

Applications.

The concept of memory and performing instructions is incredibly powerful and essential to modern software. The applications of what I learned in this lab has immense application.

Problems encountered.

I encountered issues with programming the FLASH on the CPLD. I have worked on it for a long time myself and eventually sought help from a TA. However, in the end, I could not solve the issue. I could have avoided this issue by starting the lab early and giving myself enough time to debug and troubleshoot these issues.