ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION PROJECT REPORT

PROJECT NO	:	2
PROJECT DATE	:	30.04.2020
GROUP NO	:	36

GROUP MEMBERS:

150170087	:	Sırrı Batuhan ÇOKSAK(Group Representative)
150170069	:	Furkan Yusuf GÜRAY
150170039	:	Fatih MURAT
150160142	:	Ekin Zuhat YAŞAR

SPRING 2020

Contents

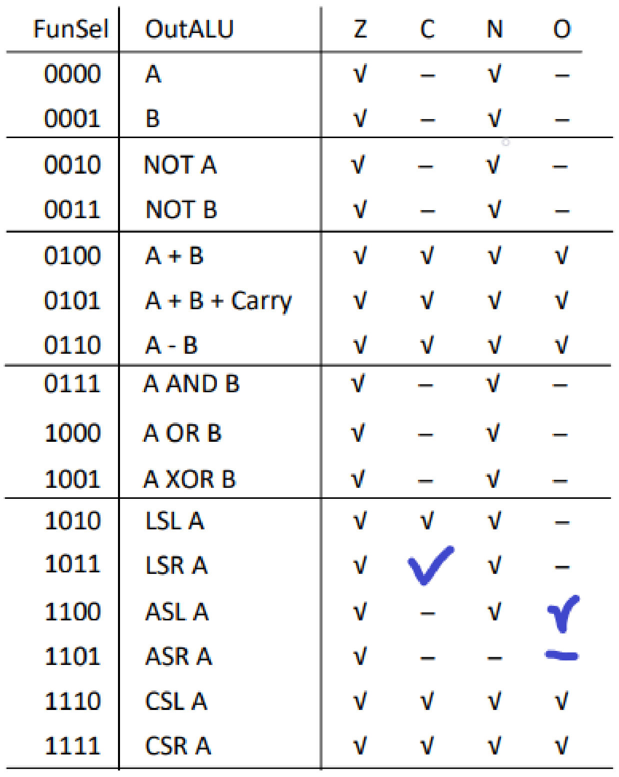
1 INTRODUCTION

In this project, we were expected to design and implement an ALU(part 1) and a system which includes several components from previous project and our ALU from part 1.

2 PART 1

In the first part of the project, we were expected to design an Arithmetic Logic Unit that has two 8-bit inputs, 4-bit FunSel input, single 8-bit output and a 4-bit OutFlag(Z/C/N/O) output.

To start with, we designed 16 different functions which corresponds to our 4-bit FunSel input. First 10 functions were self explanatory, as given in Figure 1 below. We implemented them straightforward using basic logic gates and several adders. For the last 6, we implemented shifting operations as libraries and loaded them into our ALU project.



FunSel	OutALU	Z	C	N	O
0000	A	√	–	√	–
0001	B	√	–	√	–
0010	NOT A	√	–	√	–
0011	NOT B	√	–	√	–
0100	A + B	√	√	√	√
0101	A + B + Carry	√	√	√	√
0110	A - B	√	√	√	√
0111	A AND B	√	–	√	–
1000	A OR B	√	–	√	–
1001	A XOR B	√	–	√	–
1010	LSL A	√	√	√	–
1011	LSR A	√	✓	√	–
1100	ASL A	√	–	√	✓
1101	ASR A	√	–	–	–
1110	CSL A	√	√	√	√
1111	CSR A	√	√	√	√

Figure 1: ALU function table

First, for FunSel 1010 and 1011, we implemented Logical Shift Left and Logical Shift Right circuits respectively. Using the Figure 2 below as a base for our logic. While implementing the logic, we also got necessary carry bit for Cout.

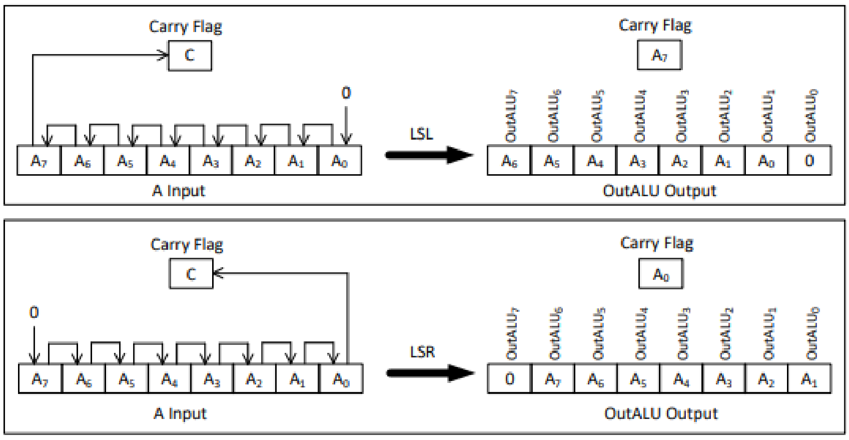


Figure 2: Logical shifts

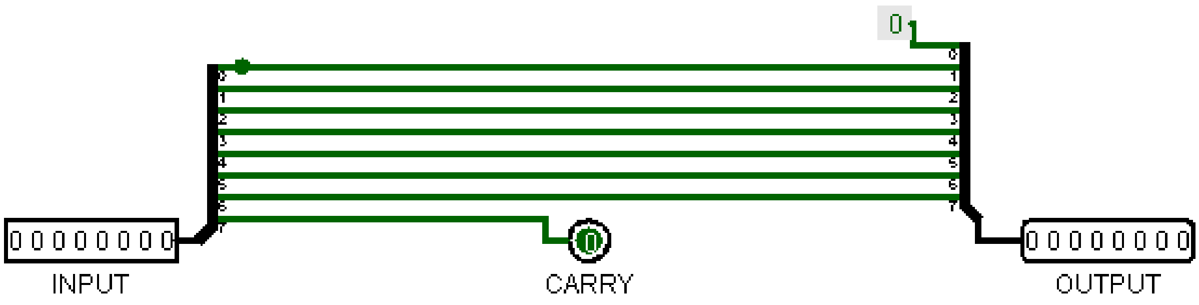


Figure 3: Our logical shift left

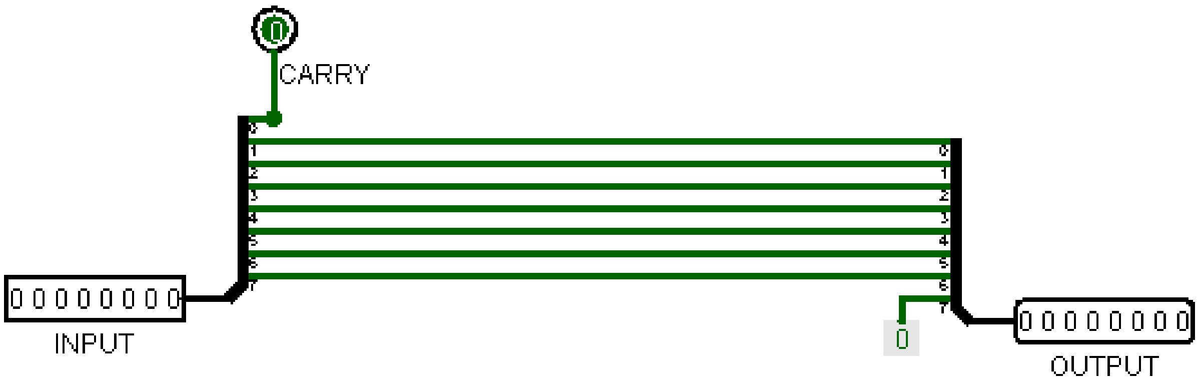


Figure 4: Our Logical shift right

For FunSel 1100 and 1101, we implemented Arithmetic Shift Left and Arithmetic Shift Right circuits respectively. Using the Figure 5 below as a base. We implemented necessary overflow and negative bits only in Arithmetic Shift Left(outside of this circuit, in main).

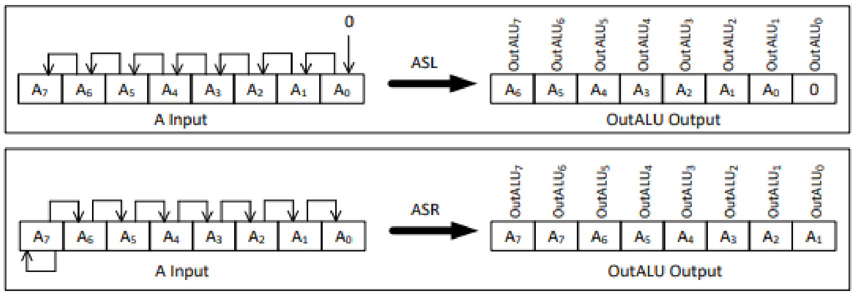


Figure 5: Arithmetic shifts

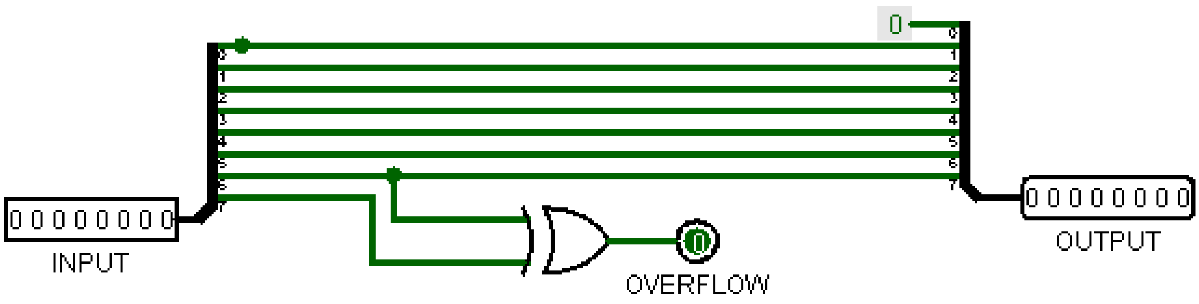


Figure 6: Our arithmetic shift left

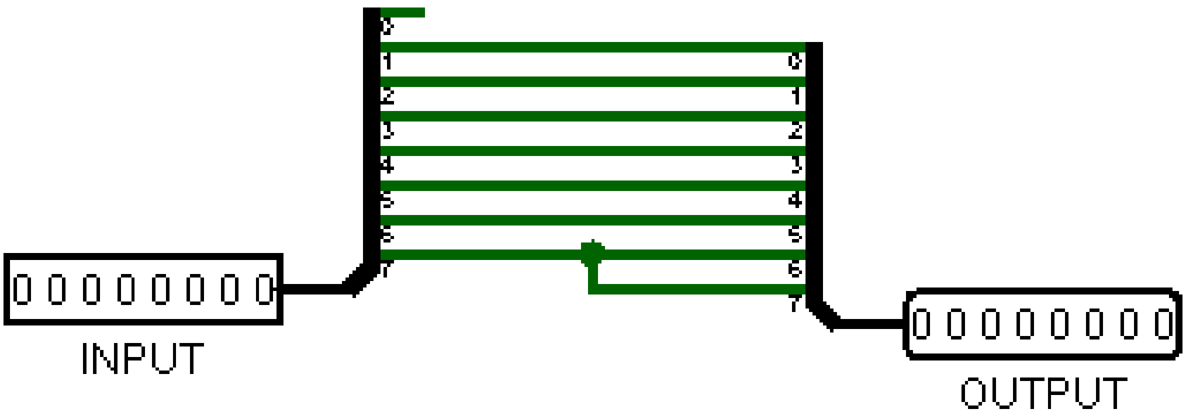


Figure 7: Our arithmetic shift right

For FunSel 1110 and 1111, we implemented Circular Shift Left and Circular Shift Right circuits respectively. Using the Figure 8 below as a base for our logic. We implemented necessary carry and overflow bits in both circuits.

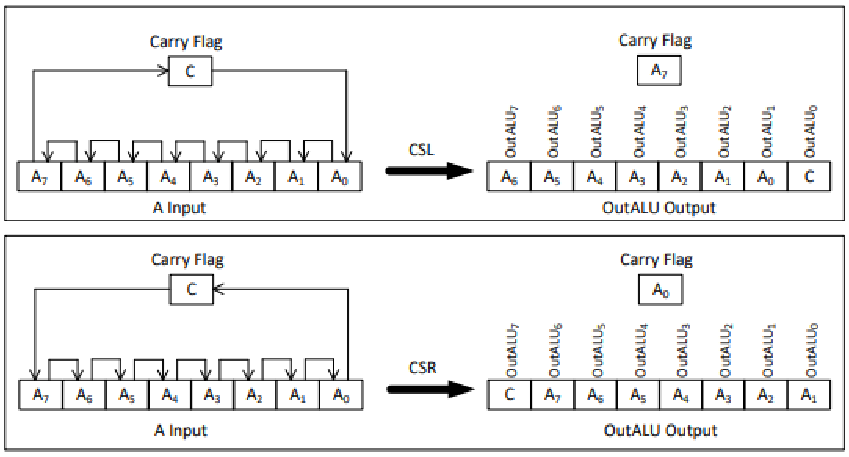


Figure 8: Circular shifts

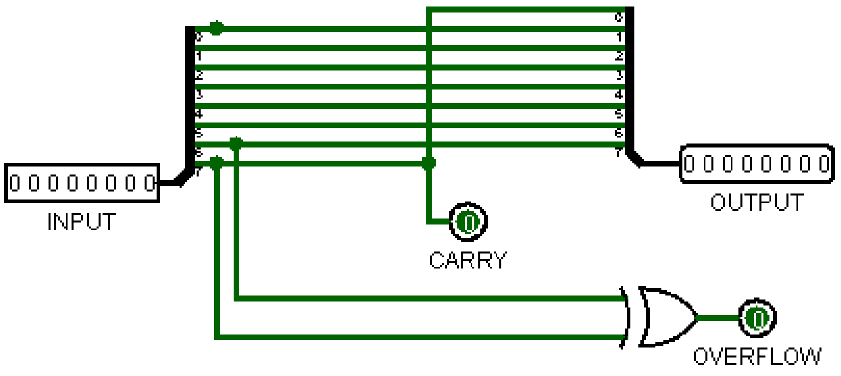


Figure 9: Our Circular shift left

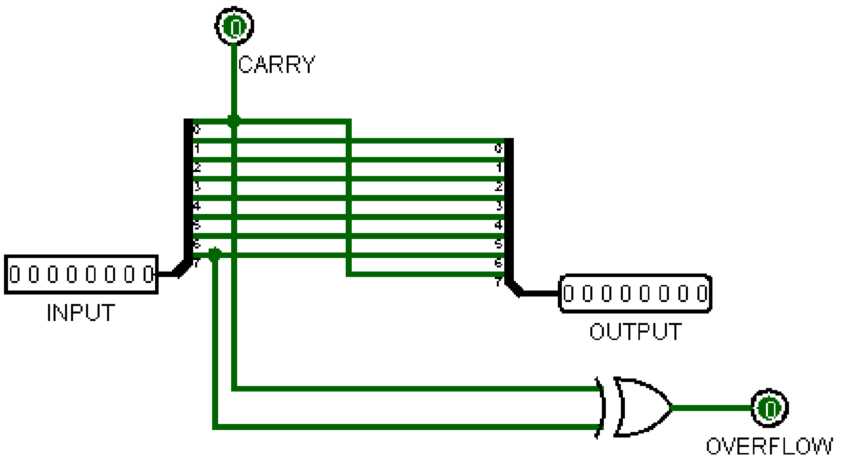


Figure 10: Our Circular shift right

Figure 11 shows the final structure of our ALU design. Our bottom 3 multiplexers select C/N/O bits necessary and for our Z bit, we compare the first multiplexers output with 0x00 to check if it is zero or not. Then we combine these bits ZCNO to our 4-bit OutFlag output.

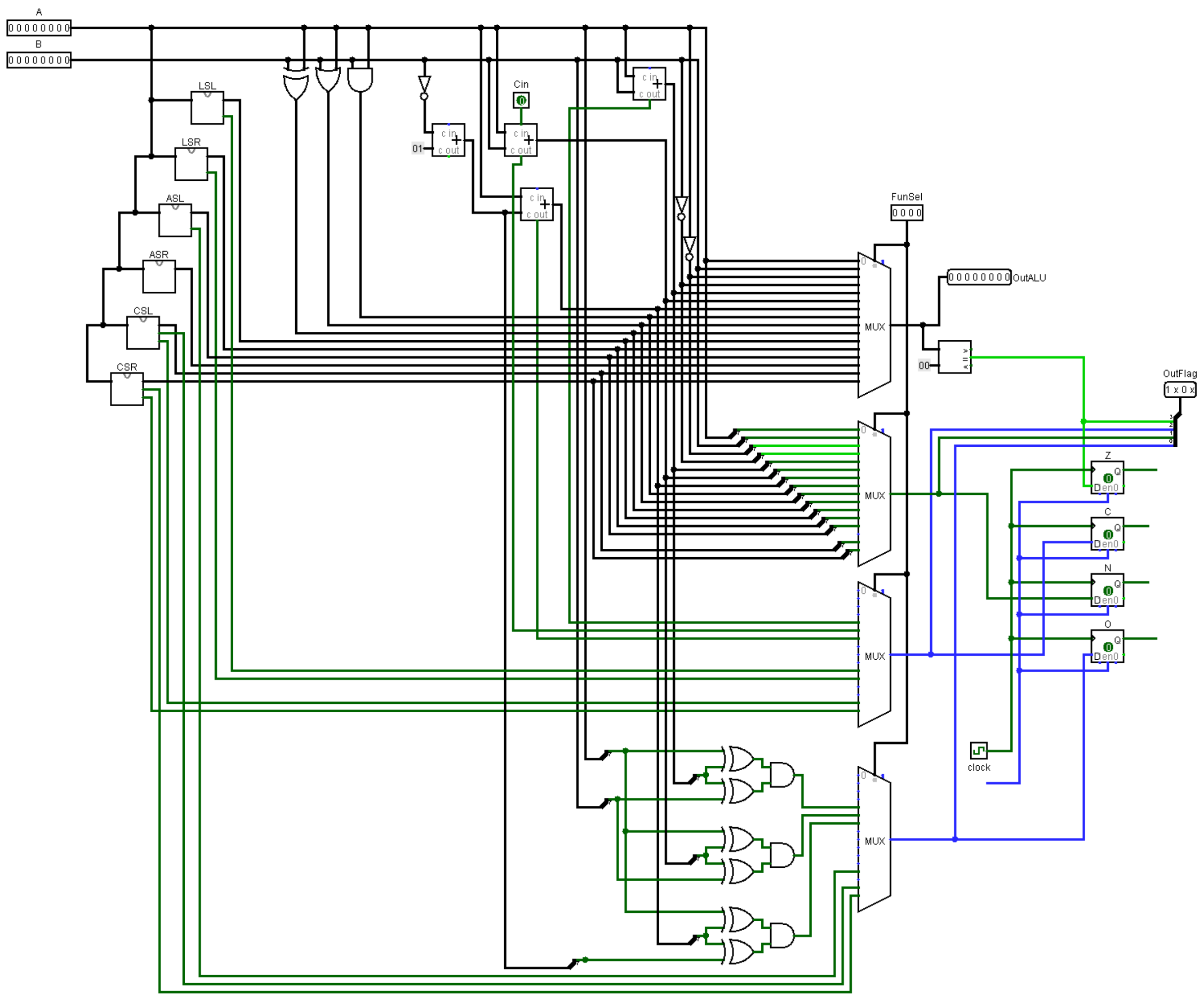
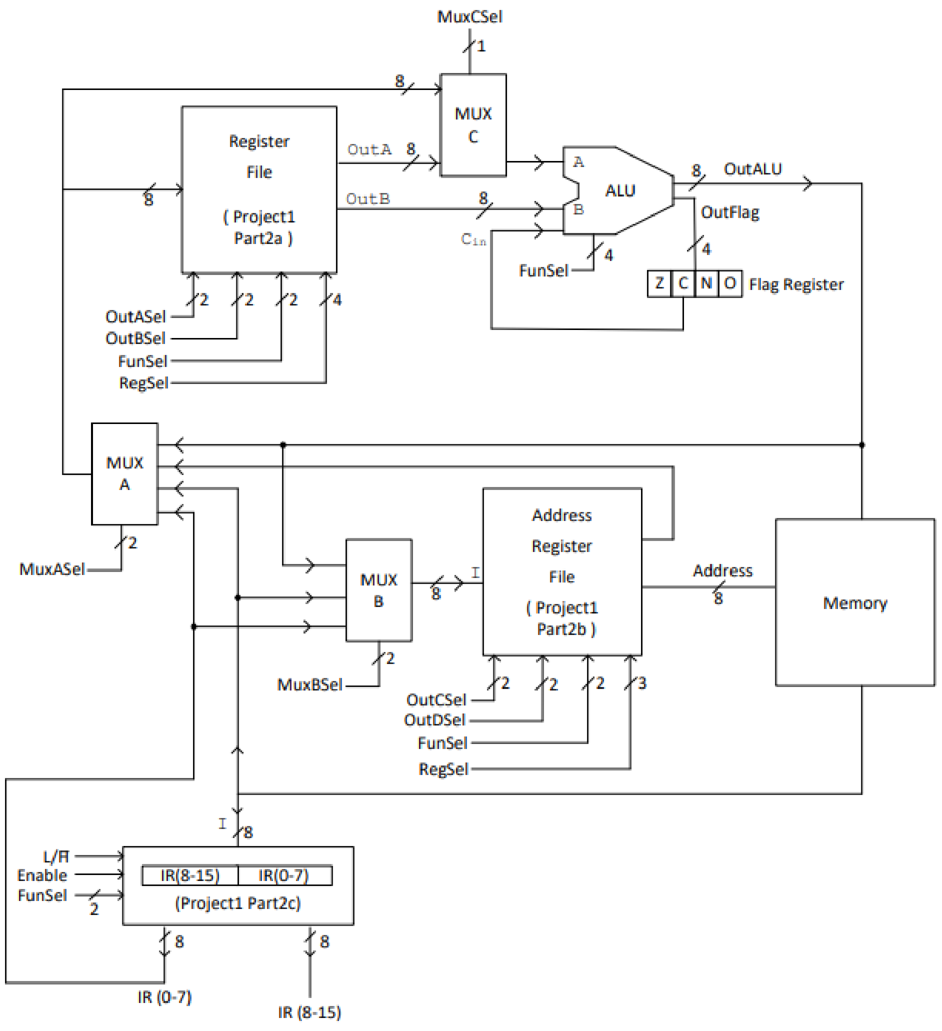


Figure 11: Insides of our ALU

3 PART 2

In this part, we designed a system which includes the ALU that we designed in part 1. The difference for our ALU was that our OutFlag output was put in a register than fed back into Cin of said ALU design.



MuxASel	MuxAOut
00	IROut (0-7)
01	Memory Output
10	Address Register OutC
11	OutALU

MuxBSel	MuxBOut
00	ϕ
01	IROut (0-7)
10	Memory Output
11	OutALU

MuxCSel	MuxCOut
0	MuxAOut
1	OutA

Figure 12: The system in Part 2

Figure 13 further explains our implementation of this project's Part 2. We rewired the inputs and outputs of the registers from first project to better fit our circuit. We connected inputs and outputs of all 3 multiplexers according to the table in figure 12. We added hex digit displays to all necessary wires that needed to be observed. In multiplexer B, we fed back the output of multiplexer into the 00 slot to implement "don't care" gate. Also all our clock inputs are wired to the single clock that we put in top right.

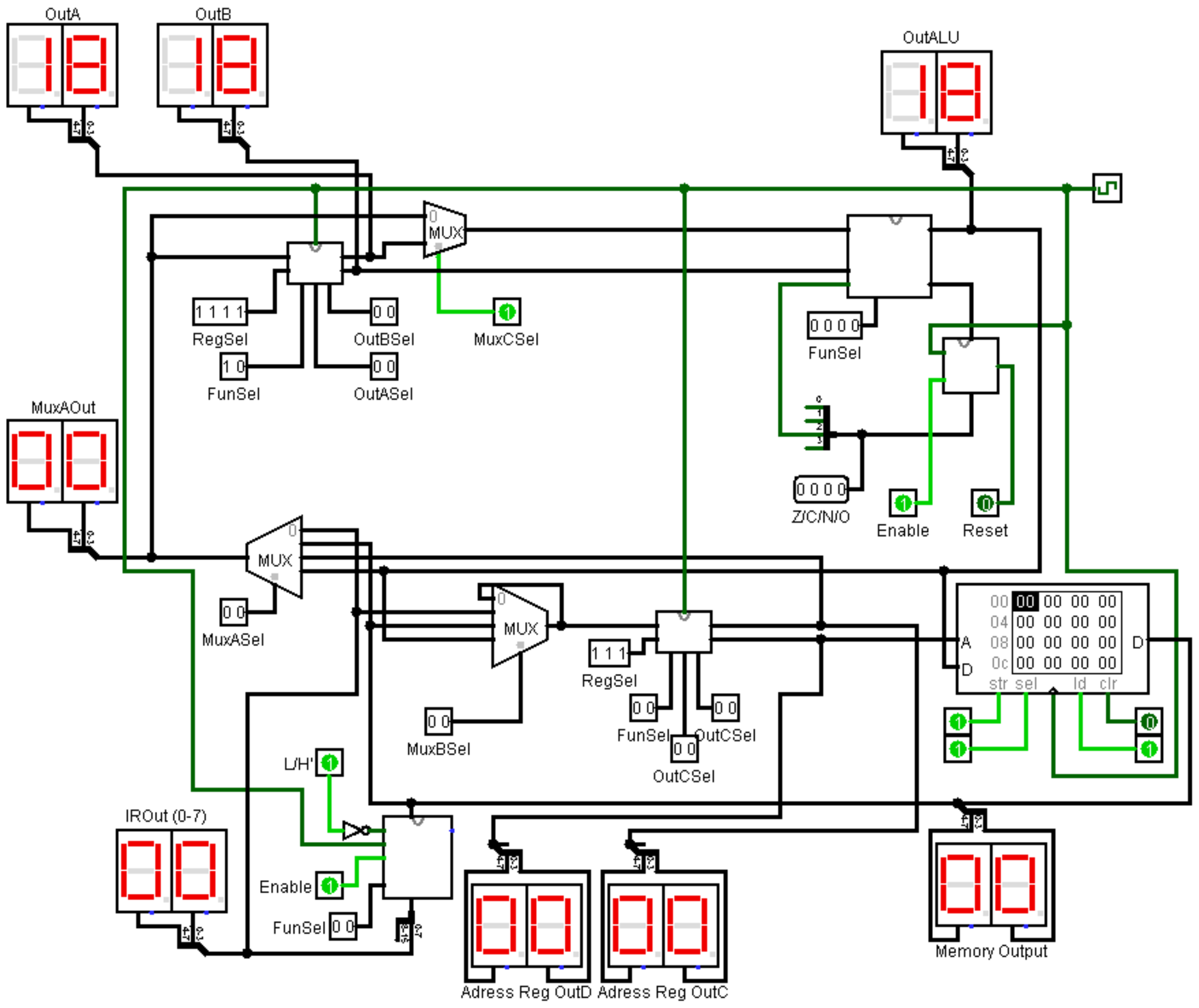


Figure 13: RegSel control inputs.

4 CONCLUSION

In conclusion, we designed and implemented several libraries to accomplish designing the system. Due to recent events of our time, we were unable to meet up. We had to use a voice chat program, which allowed us to share our screens so that all four of us could see the project in progress. Furthermore, using Overleaf share function, we were able to utilize multiple computers to build the project as we present it now.

References

[1] Overleaf documentation https://tr.overleaf.com/learn.

[2] Software tool (logisim) http://www.cburch.com/logisim/