For the control unit pictured in figure 1, I followed the figure given to us in the project handout to design the top-level design. The IR Register has 0x9000 loaded into it and is then dissected. IR(15) goes out as the I bit. Then bits IR(12-14) go through a 3x8 decoder and are outputted to the control unit as D0-D7. Finally, IR(0-11) gets loaded into the control unit as B0-B11. Down below the IR register, there is a 3-bit synchronous sequence counter which goes through another 3x8 decoder and the output from that goes into the control unit as T0-T7.

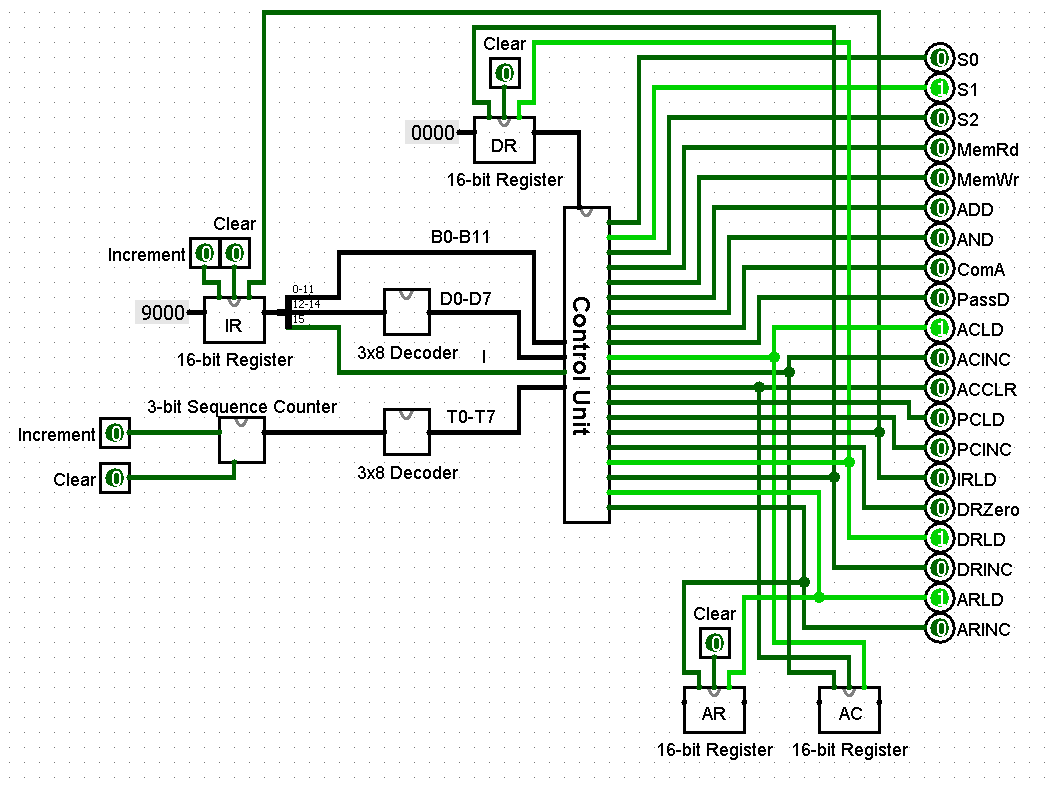


Figure 1: Control unit, top-level design with all registers implemeneted

Going into the Control unit, We will find a bunch of smaller, lower level blocks which derive the control signals. The control unit takes it four inputs which are: B0-B11, D0-D7, I and T0-T7. With these inputs it distributes them to the 8 smaller blocks which are

* Selector bit block [Output: S0-S2 ]
* Memory Block [Output: MemRd & MemWr ]
* ALU Block [Output: ADD, AND, ComA & PassD ]
* AC Block [Output: ACLD, ACINC & ACCLR ]
* PC Block [Output: PCLD & PCINC ]
* IR Block [Output: IRLD ]
* DR Block [Output: DRZero, DRLD & DRINC ]
* AR Block [Output: ARLD & ARINC ]

The Gate logic for each of these blocks will be displayed below in figures 3-10

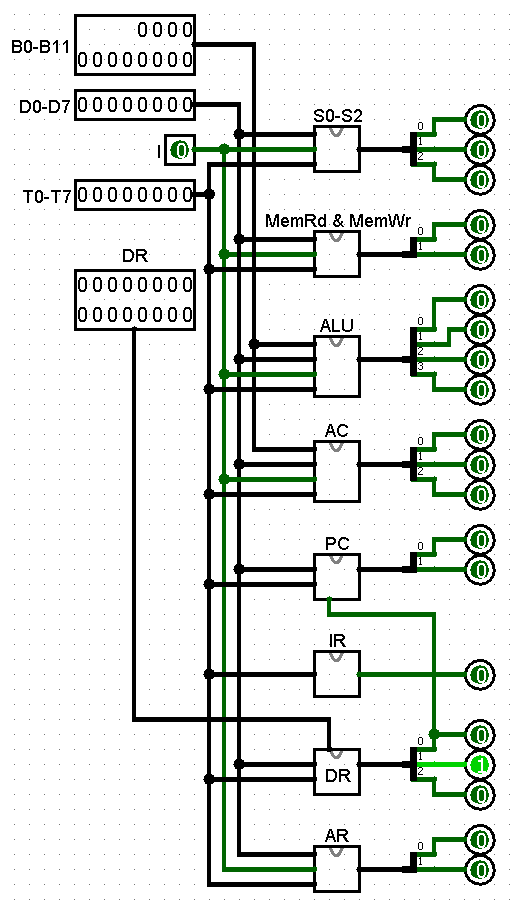


Figure 2: Control Unit

In order to determine the equations to find the 3 selector bits, I followed the information in the I/O Table displayed in table 1. From this I derived an equation for each input. After deriving each input, I inserted the 8 bit output into an 8x3 encoder in order to get a 3 bit binary number in the form of the selector bits and this was outputted back to the control unit which then was outputted out of the control unit to the rest of the circuit.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Inputs | | | | | | | Outputs | | | Register selected for bus |
| X1 | X2 | X3 | X4 | X5 | X6 | X7 | S2 | S1 | S0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | None |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | AR |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | PC |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | DR |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | AC |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | IR |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | None |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | Memory |

Table 1: I/O Table for the Selector bit Block

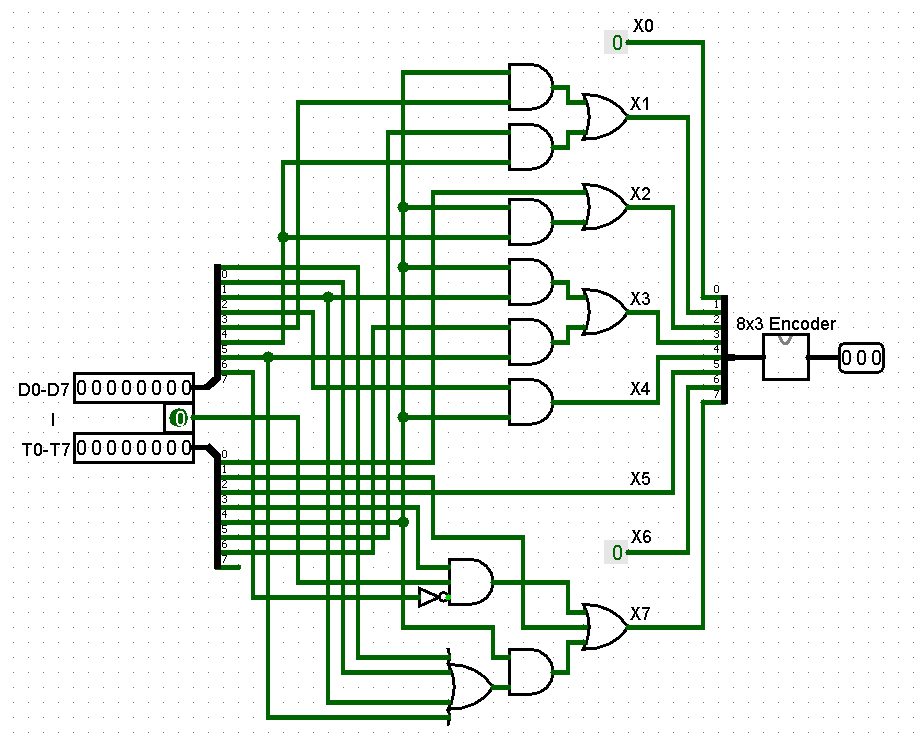


Figure 3: Selector bit Block for S0, S1 and S2

For the memory block, I looked at the instructions to determine when the Memory is read and when the memory is written to. From there, I derived logic for all the cases and inputted into the CAD Tool.

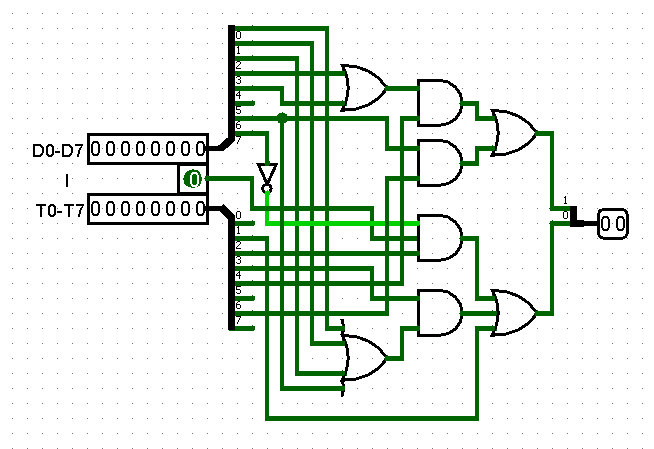


Figure 4: Memory Block for MemWr and MemRd

For the ALU block, I looked into when each operation was used during the instructions and derived the four equations for ADD, AND, Complement AC and Pass DR.

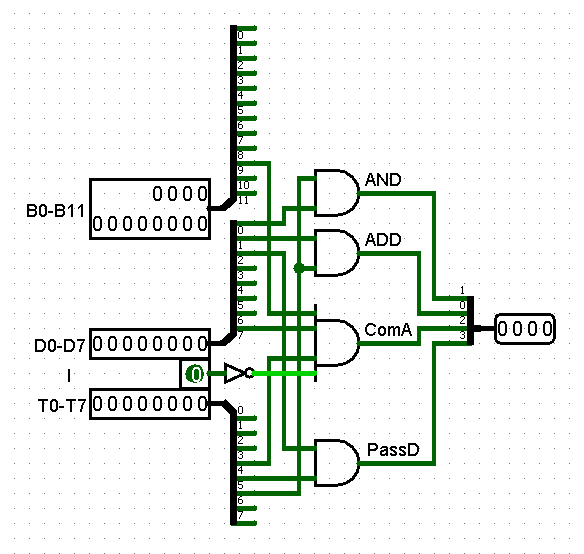
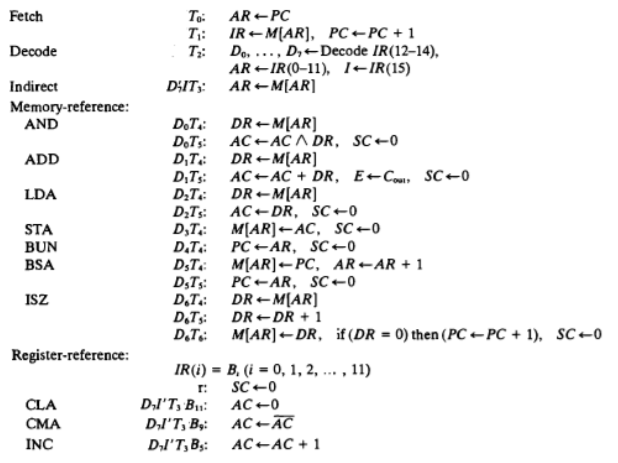


Figure 5: ALU Block for AND, ADD, ComA and PassD

I followed this same level of derivations for the remaining blocks in figures 6-10. I will show the instructions that I used to derive all of the equations below.



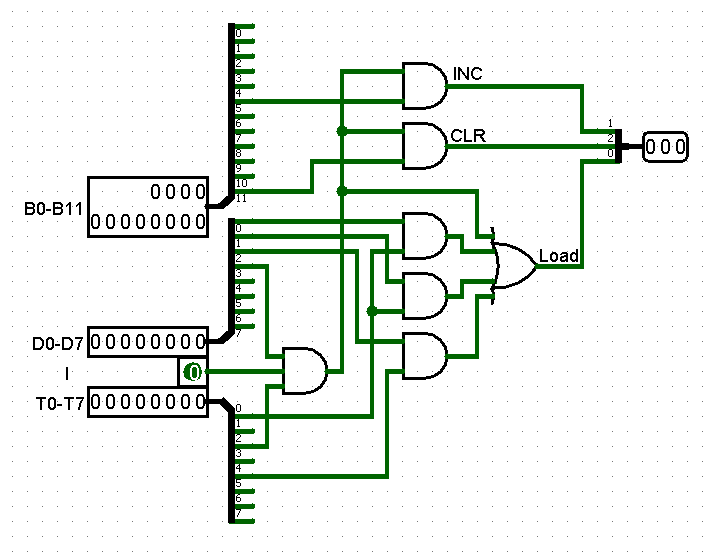


Figure 6: AC Block for ACINC, ACCLR and ACLD

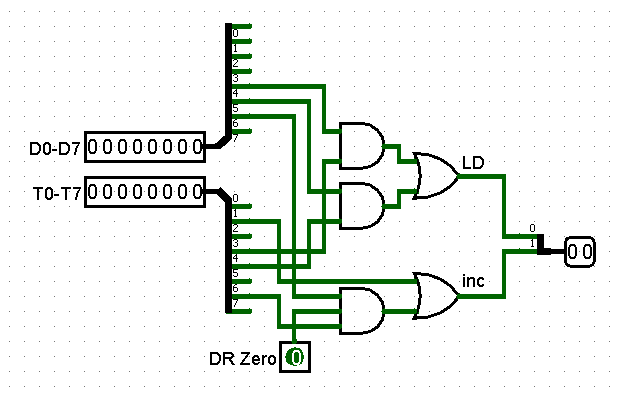


Figure 7: PC Block for PCLD and PCINC

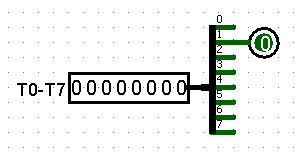


Figure 8: IR Block for IRLD

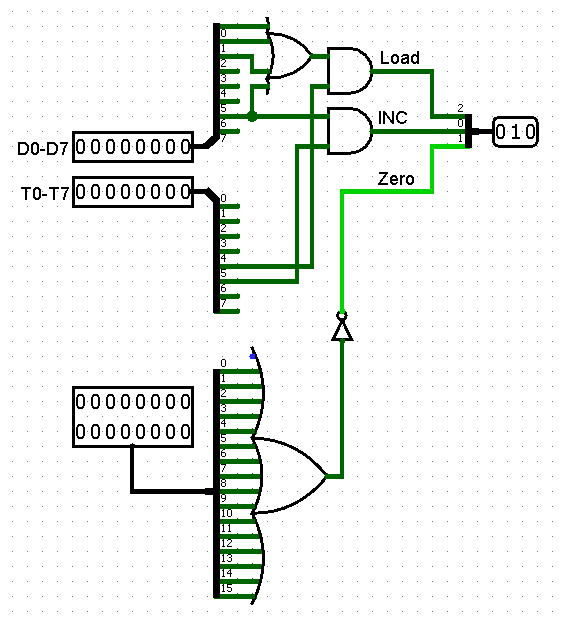


Figure 9: DR Block for DRLD. DRINC and DRZero.

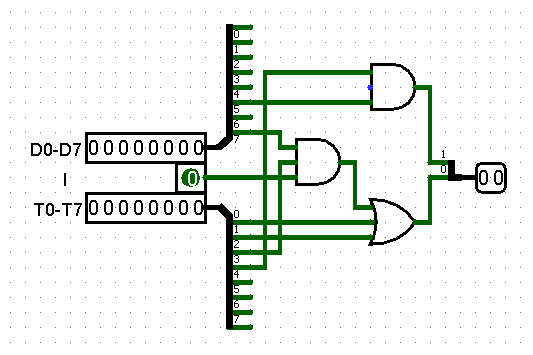


Figure 10: AR Block for ARLD and ARINC