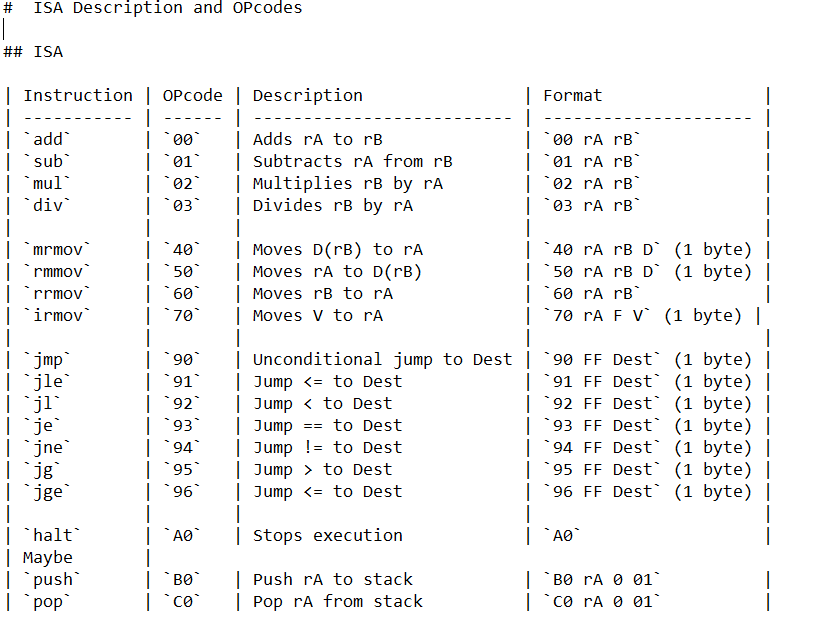
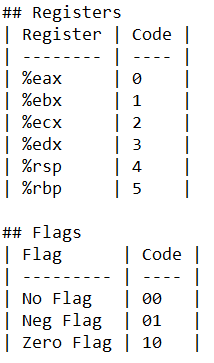
Group 19

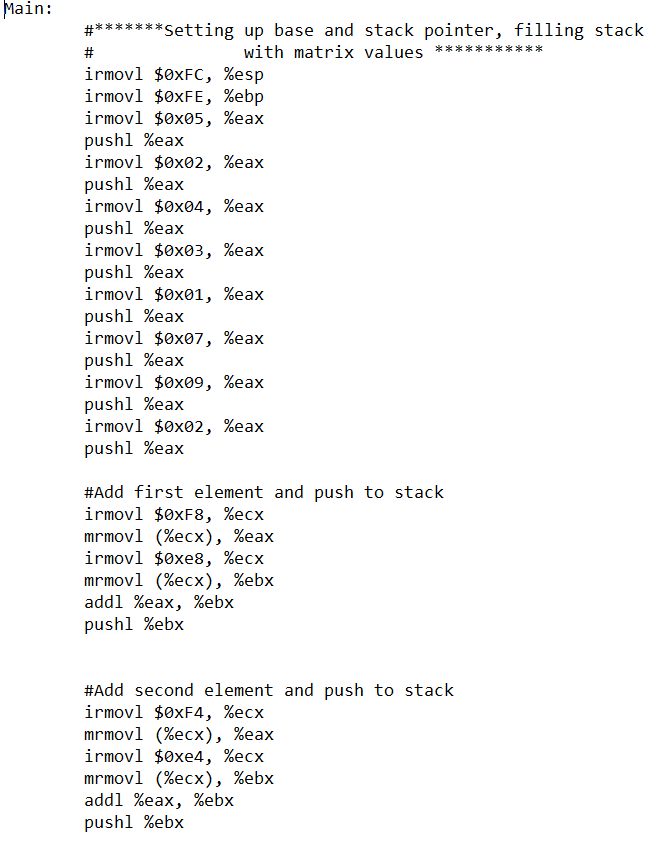
Final Project Report

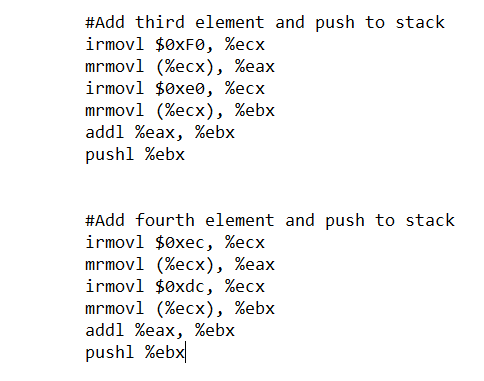
Problem 1)

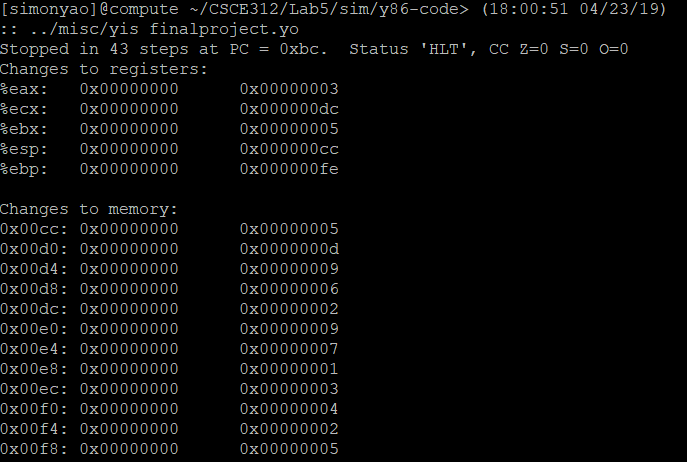




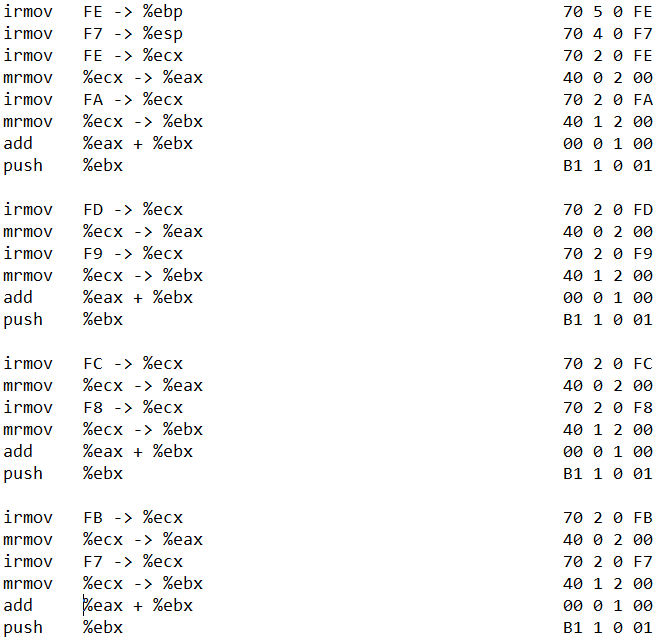
Problem 2)

The assumptions for Matrix A, B, and C are that they are all 2x2 matrices. The corresponding values from the first value in matrix A and B should be added to produce the first value in matrix C. Continue this for the other 3 values. This produces matrix C. Eax holds the value of matrix A and is added to ebx which holds the matrix B value. The resulting ebx value is pushed for the corresponding matrix C value. 

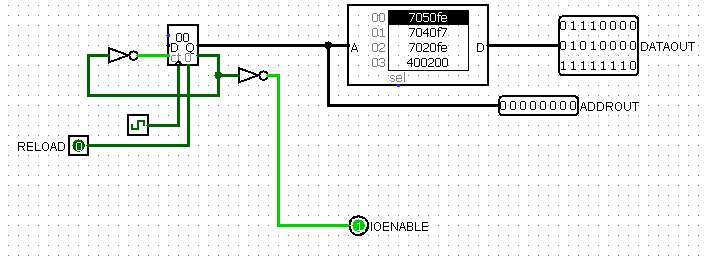


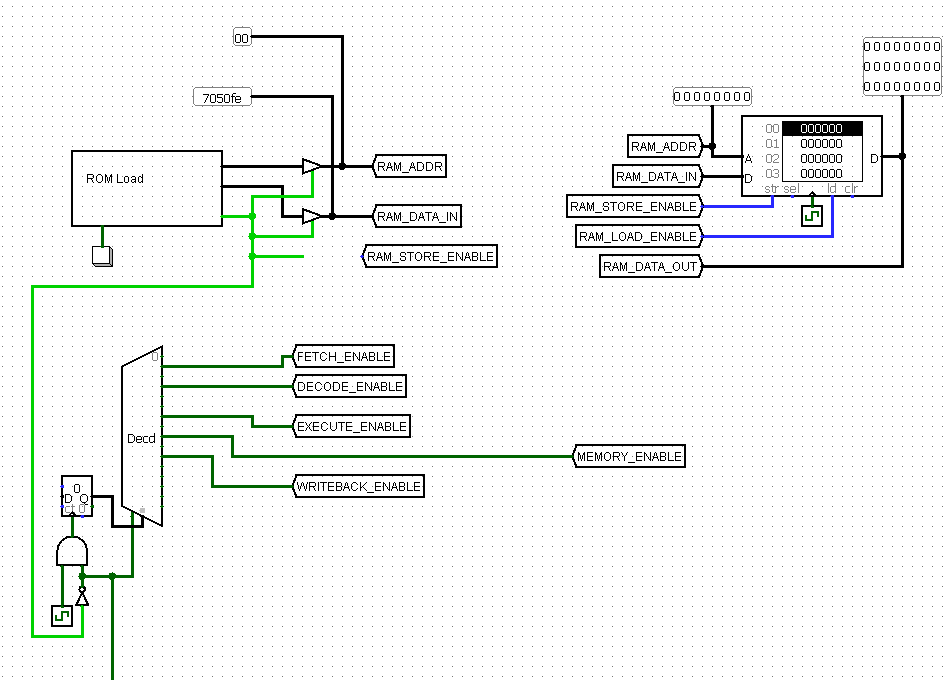


Our ISA program:



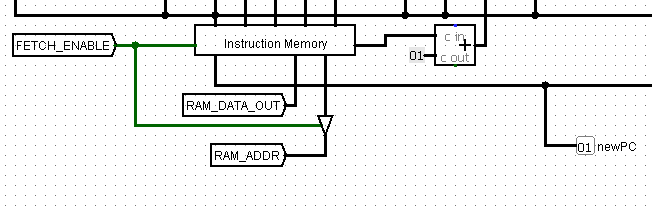
The architecture is divided based upon the Y86 sequential architecture implementation shown in class with the 5 sections fetch, decode, execute, memory, and write back. The PC is then updated. The first step is to load instructions and data from the ROM to the RAM. This is performed in a sub circuit known as the ROM loader. The ROM loader is shown:



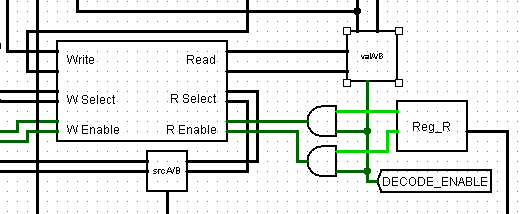


The processor then begins evaluating instructions sequentially based on the 5 steps with implemented no-OP’s to compensate for register delay.

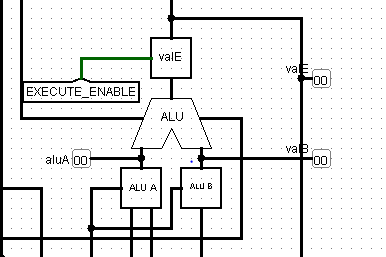
Fetch: For the fetch step, the processor starts at an address of 00. It reads the instructions located at that memory location into the instruction memory register and is used to feed busses that hold the iCode, iFun, rA, rB, and valC.



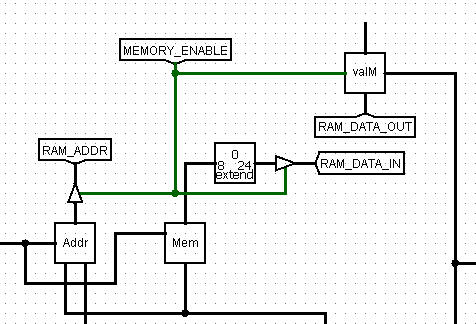
Decode: The decode step is then initiated by using the OP code to decide which sources to read from for the valA and valB busses.



Execute: Next, the execute stage begins where the values at ALU A and B are passed into the ALU. The instructed function is performed based on the iFun value. This is then stored into valE.



Memory: Next, the memory stage is initiated depending on the iCode. It either stores or loads data to/from the RAM. Fetched values are stored in valM.



Write back: Finally, the program writes back data values from valE and/or valM into the register file at the location specified by the instruction. The PC is then incremented based on whether or not a jump was encountered. The processor then loops into the same set of steps.

