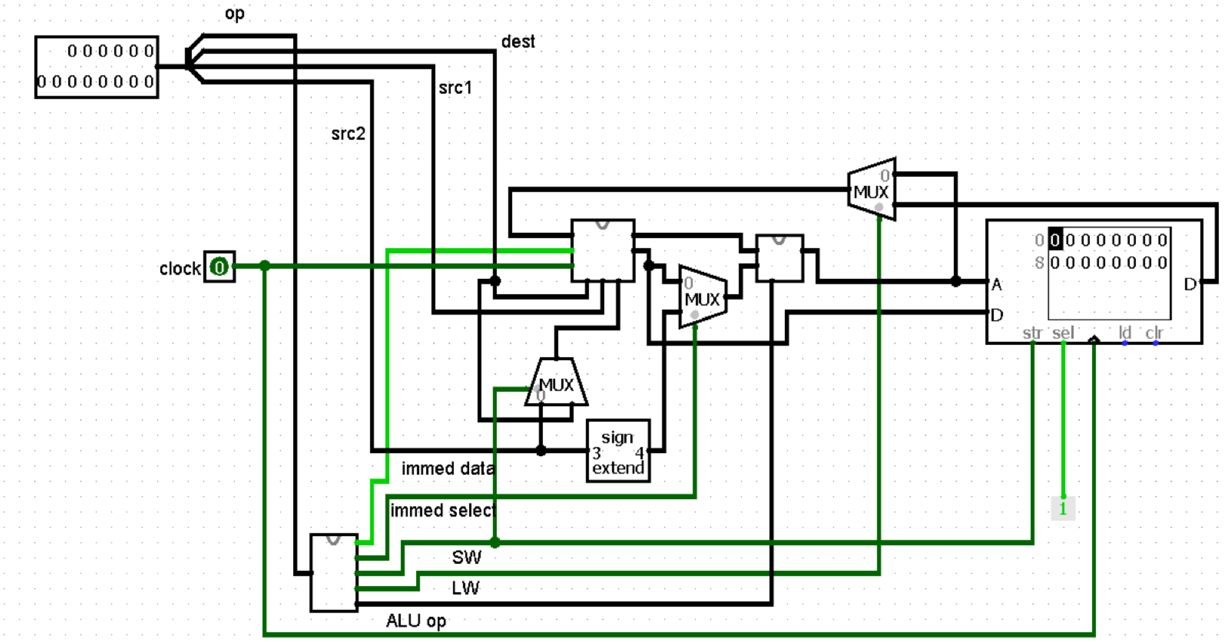
Lab 4

Testing the circuit (ALU with mem):

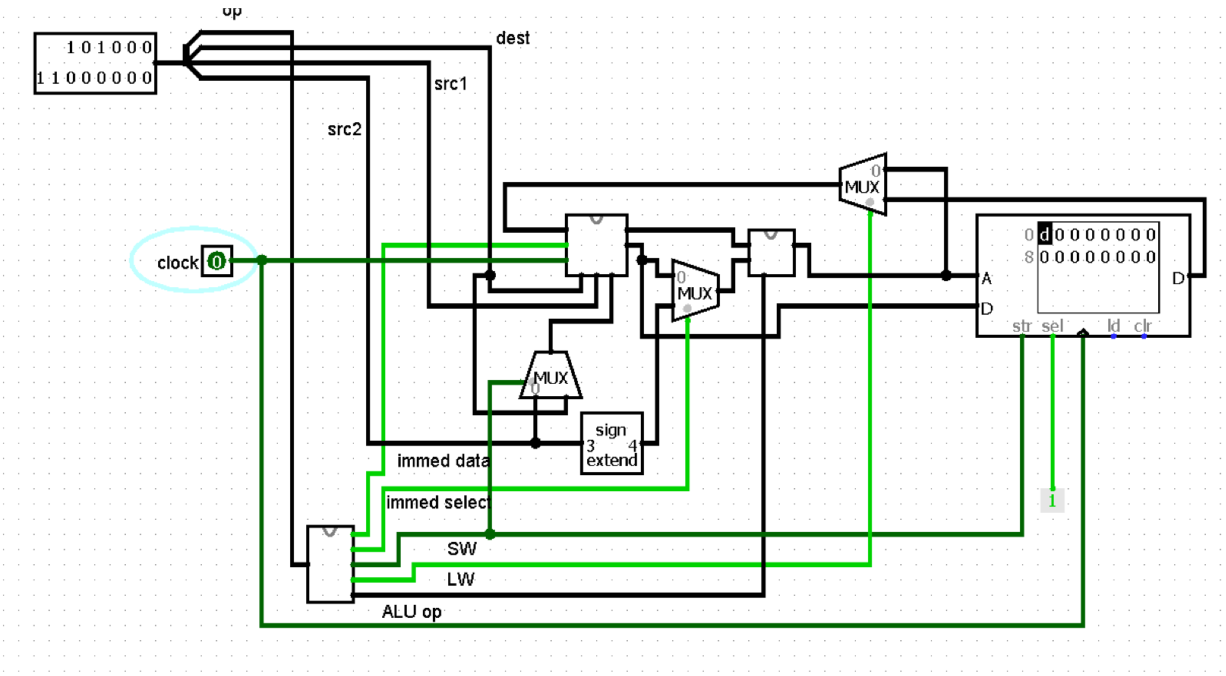


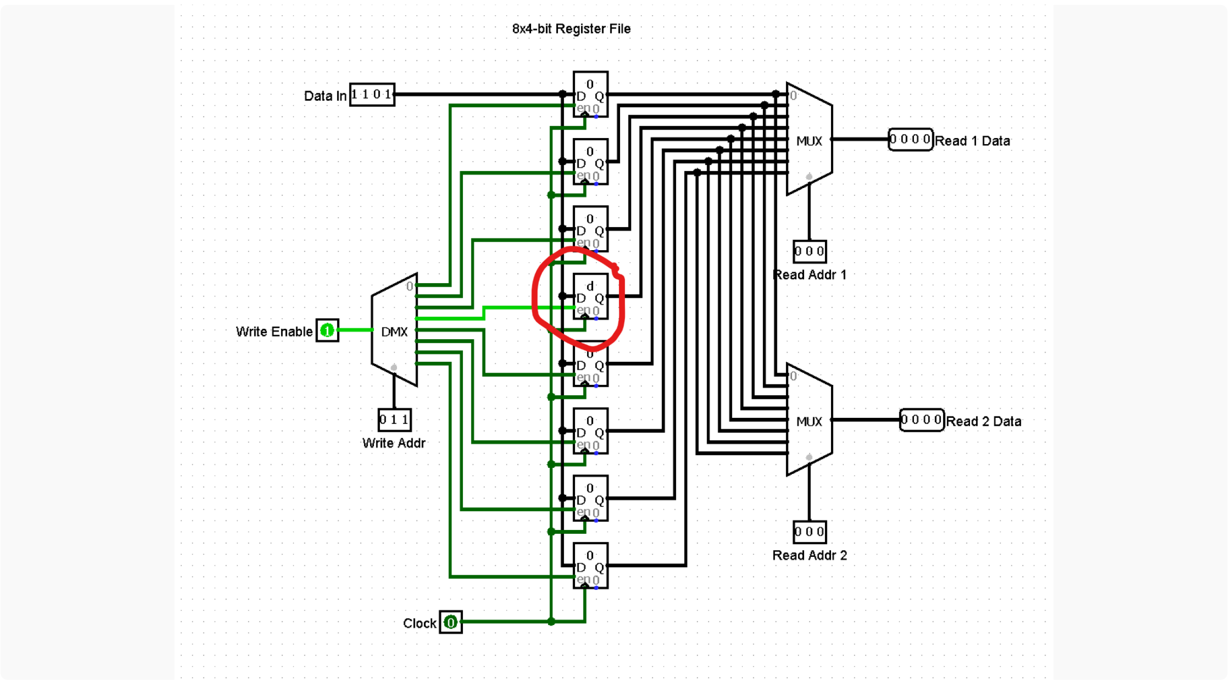
First instruction:

Poke a value like `d` into data memory location 0, then poke your instruction word so that it performs the equivalent of `lw $3 0($0)`. This should put the value `d` into register 3, presuming register 0 is 0 to begin with.

```
lw $3 0($0) = 1 0 1 00 011 000 000 (I/SW/LW/OP/WA/RA1/offset)
```

After rising clockedge:

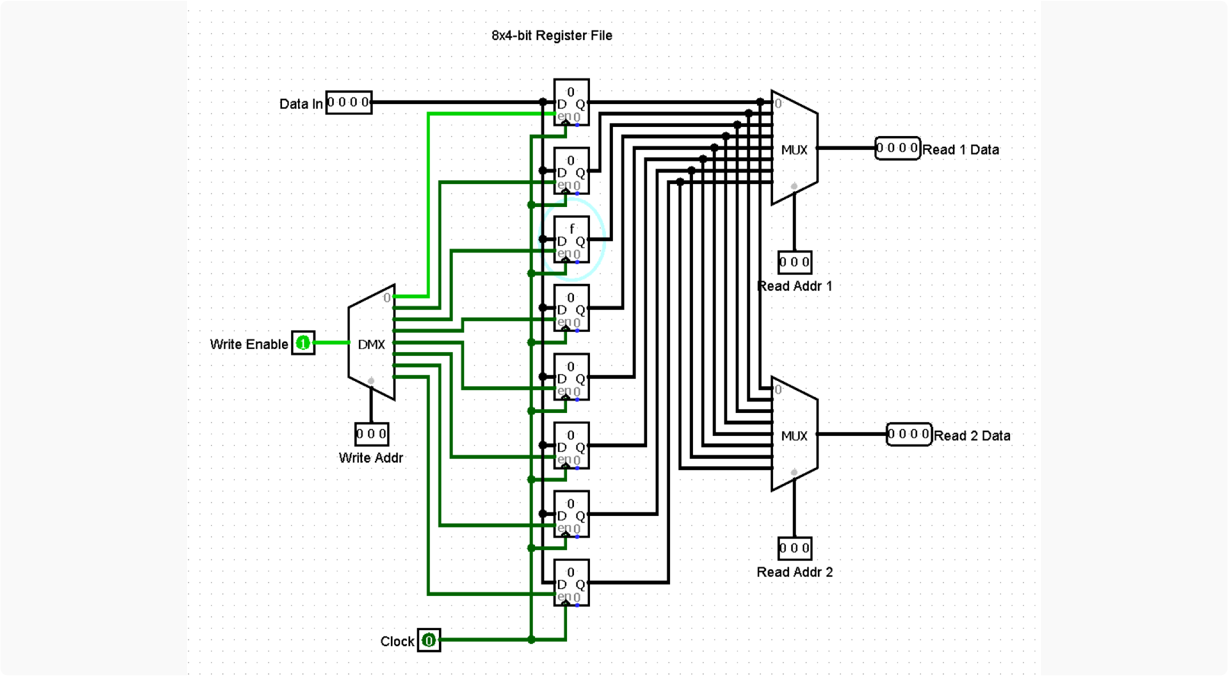


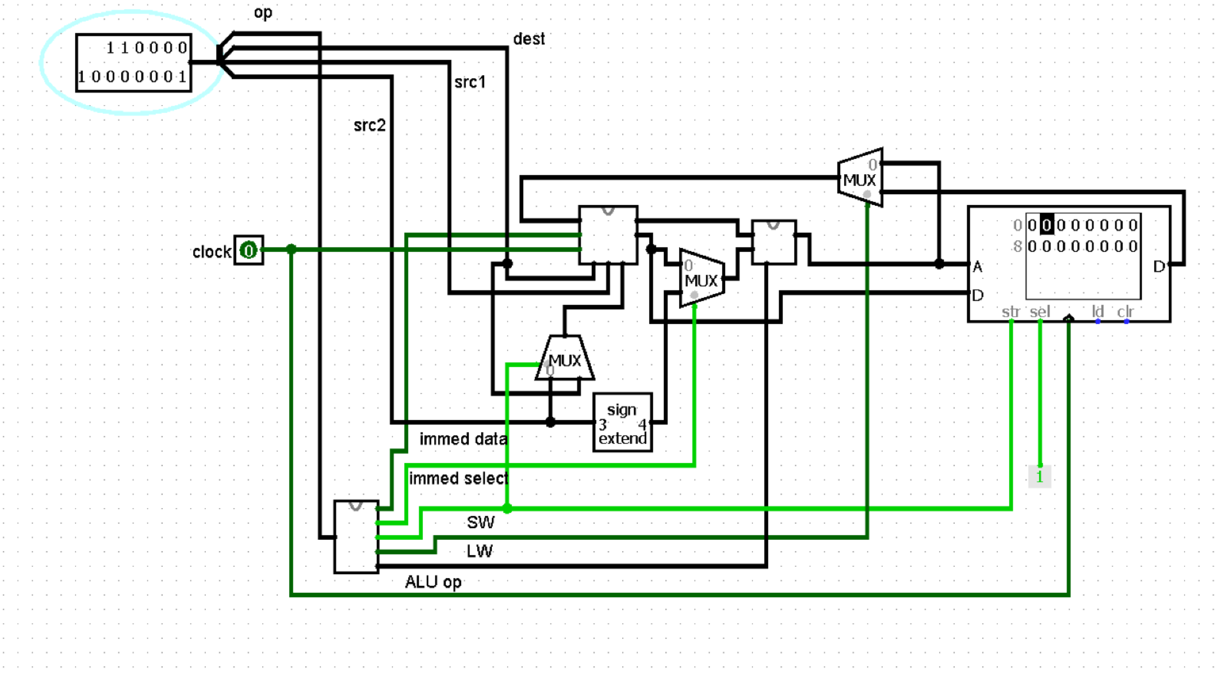


Second Address: Put a value like 'f' into register 2, then poke your instruction word bits so that it performs the equivalent of `sw $2 1($0)`. This should write the value `f` into memory address

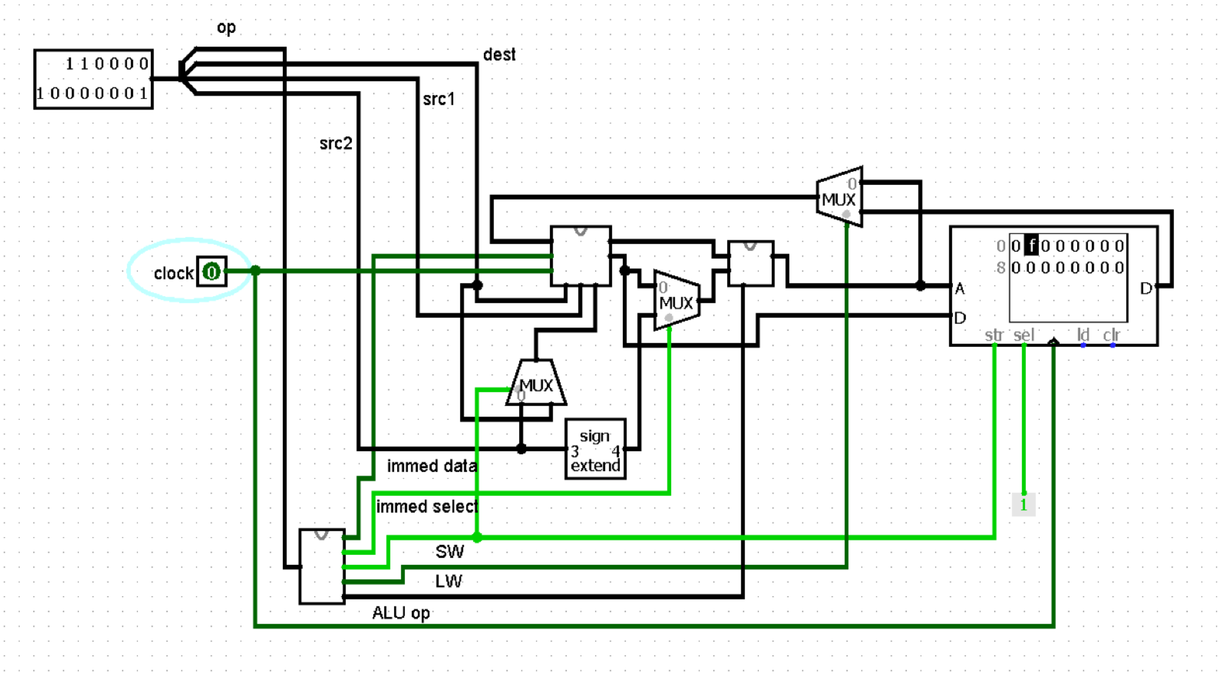
```
sw $2 1($0) = 1 1 0 00 010 000 001 (I/SW/LW/OP/RA2/RA1/offset)
```

Before rising clock edge



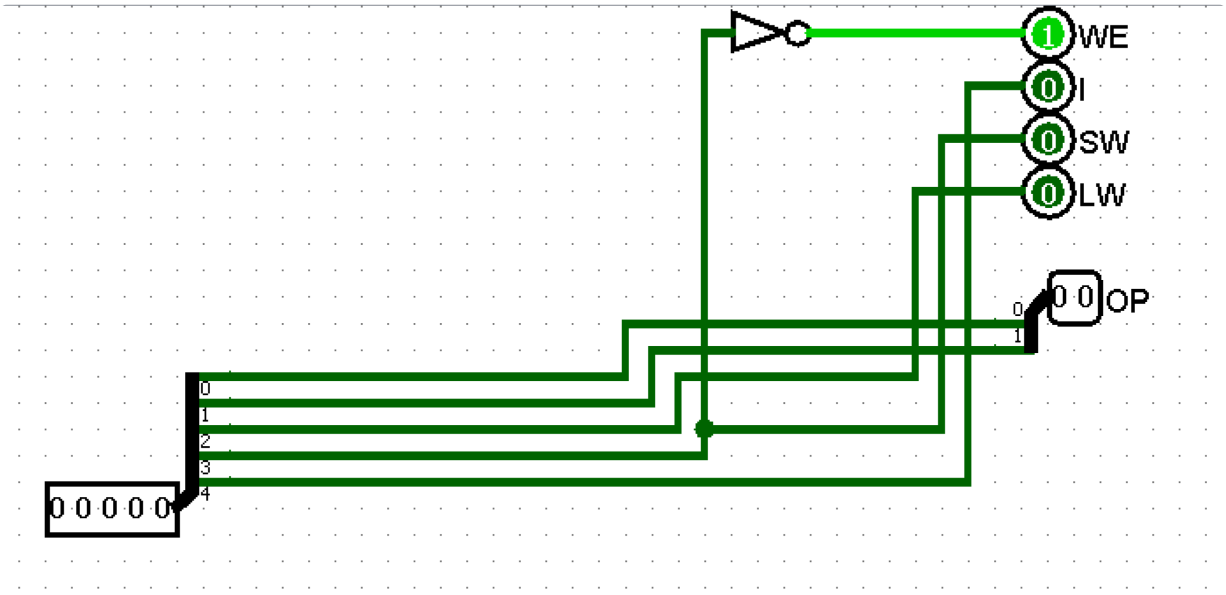


After rising clockedge:

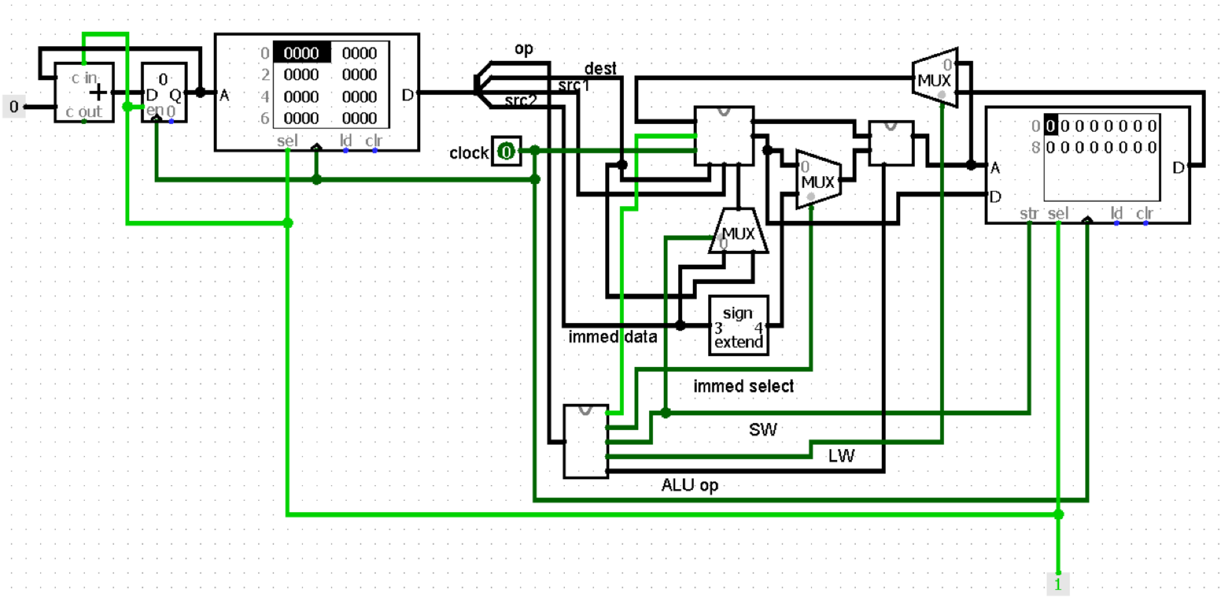


Control Logic

The module:



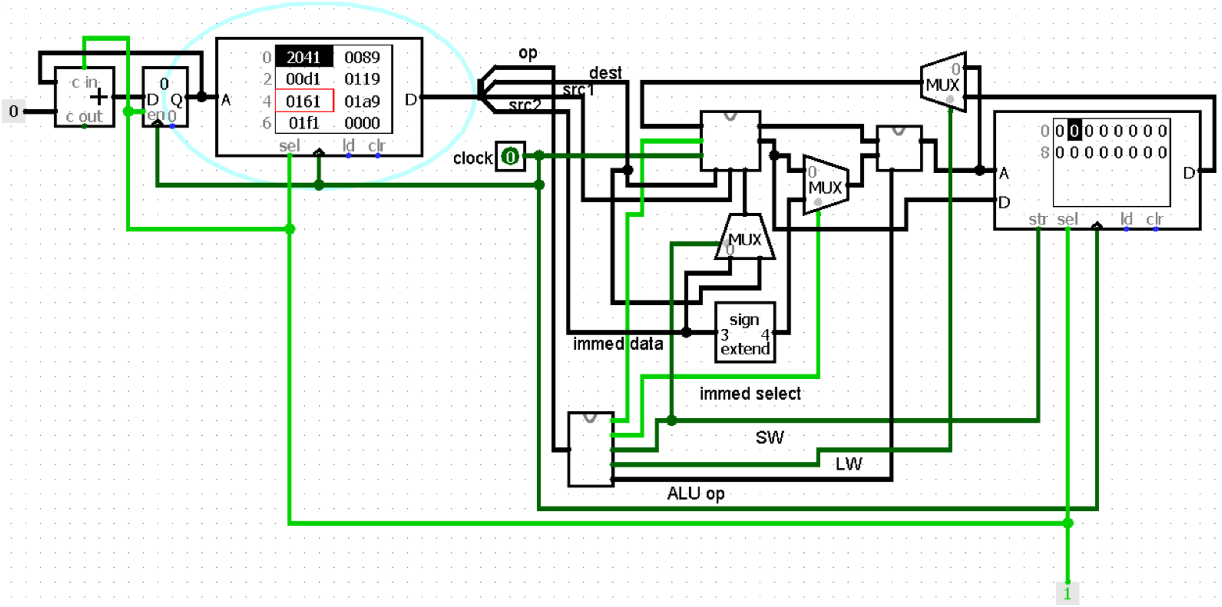
Full CPU:

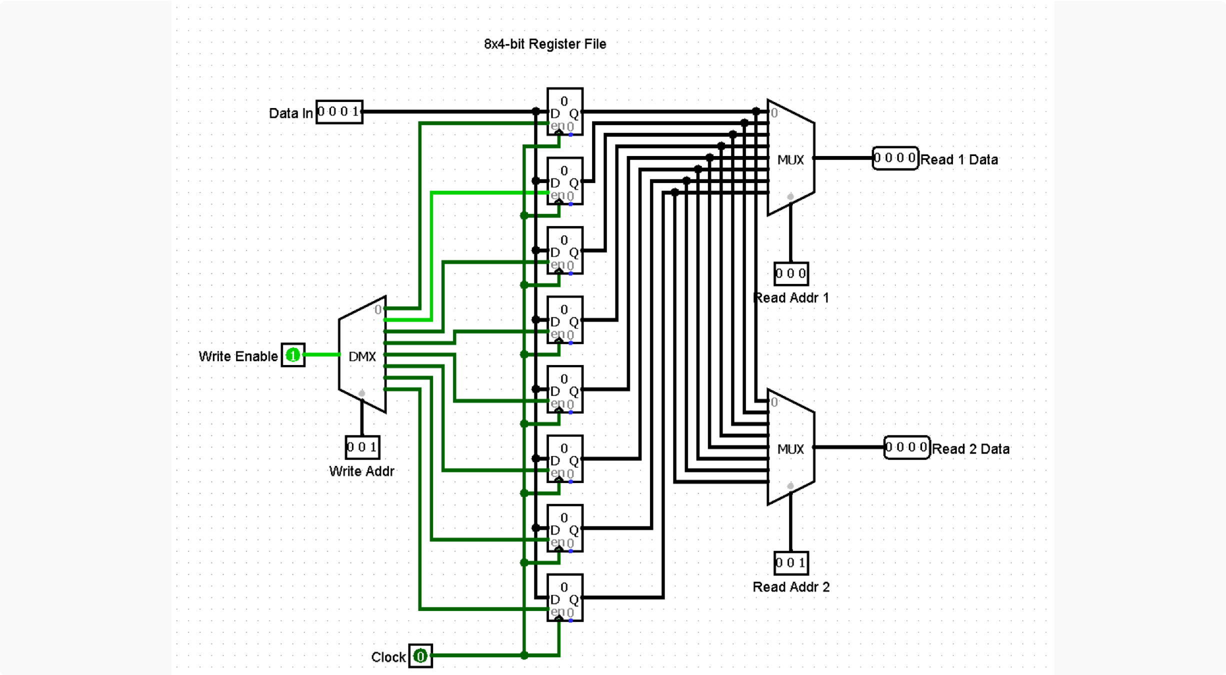


Testing;

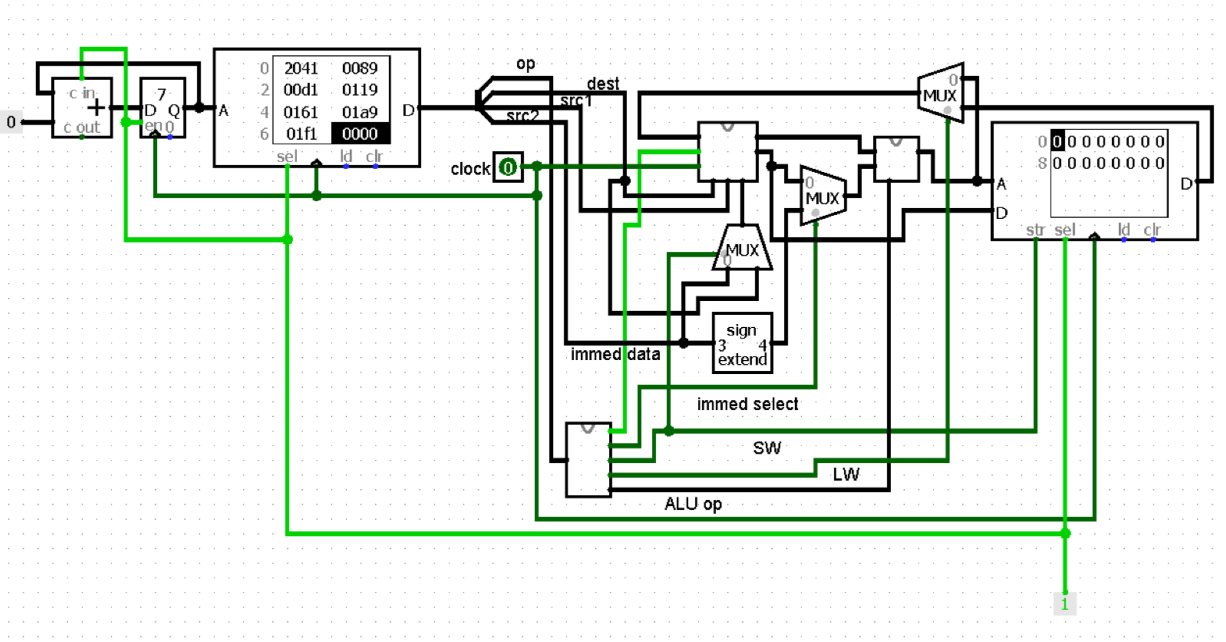
1) Loading the 14-bit hex commands in the instruction memory

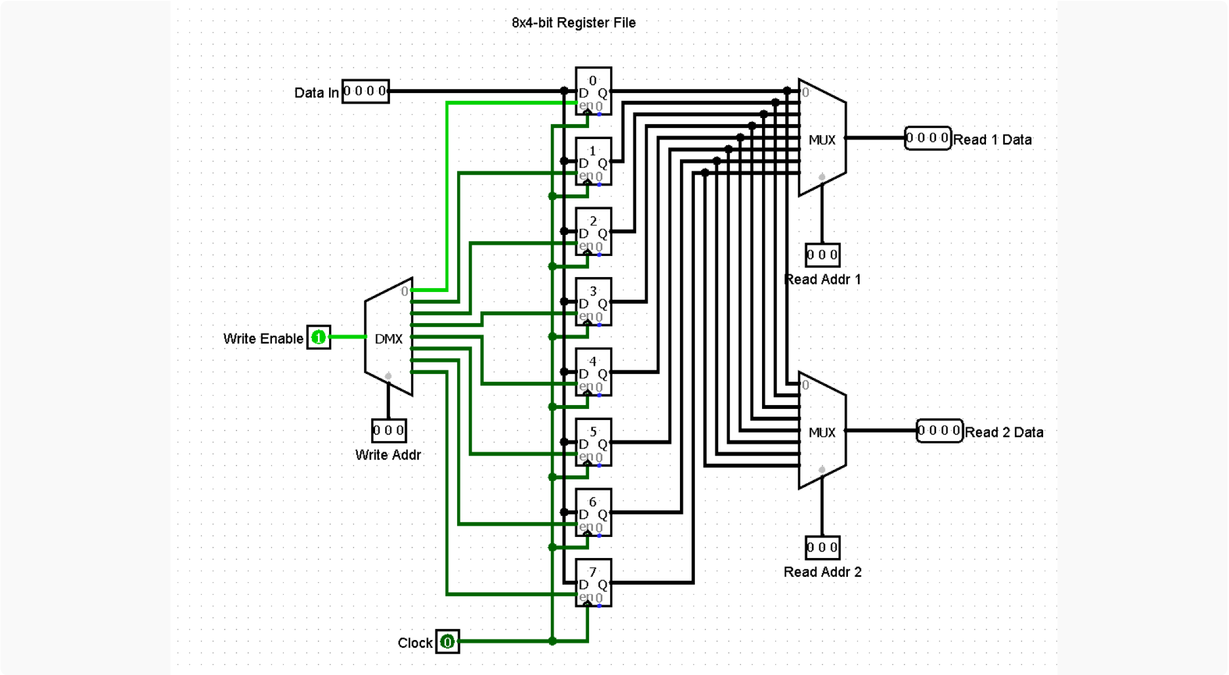
Before rising clock edge:





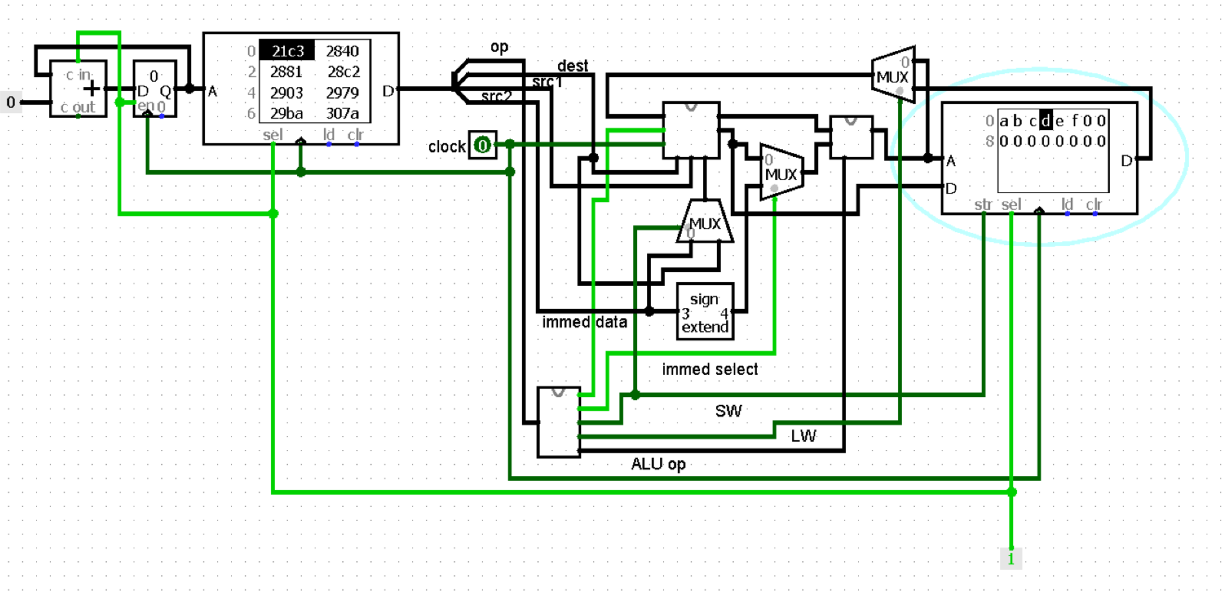
After multiple rising clockedges

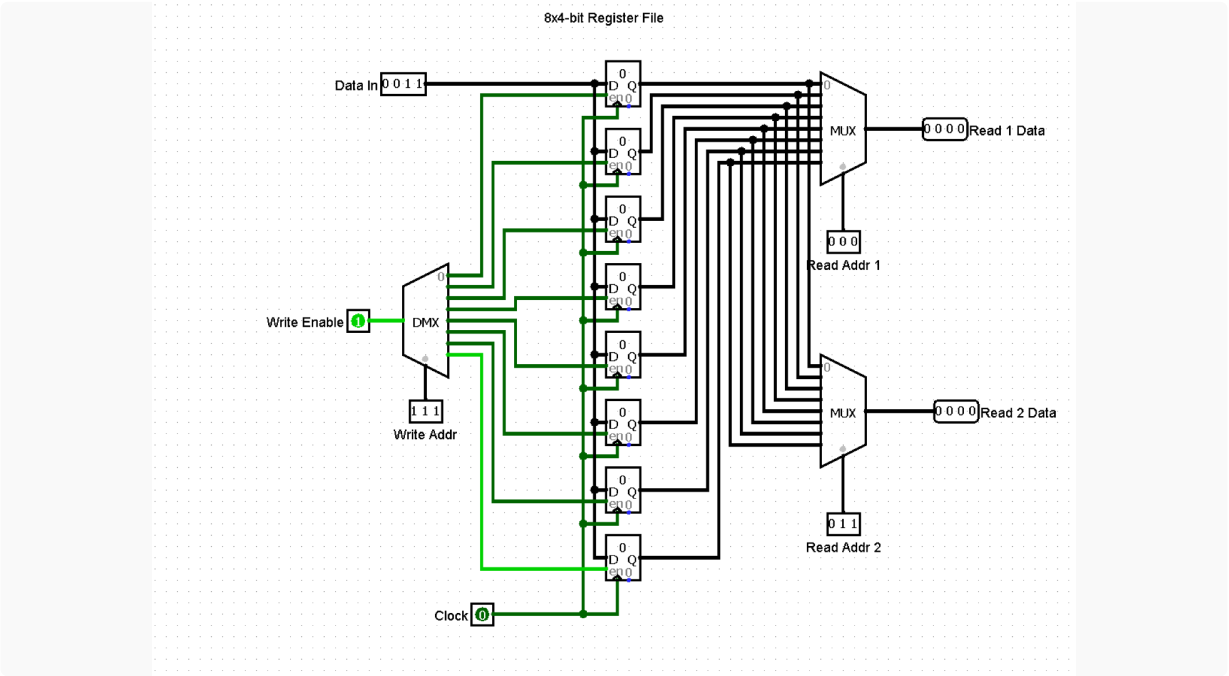




2) Reverse: Loading the 14-bit hex commands to in the instruction memory

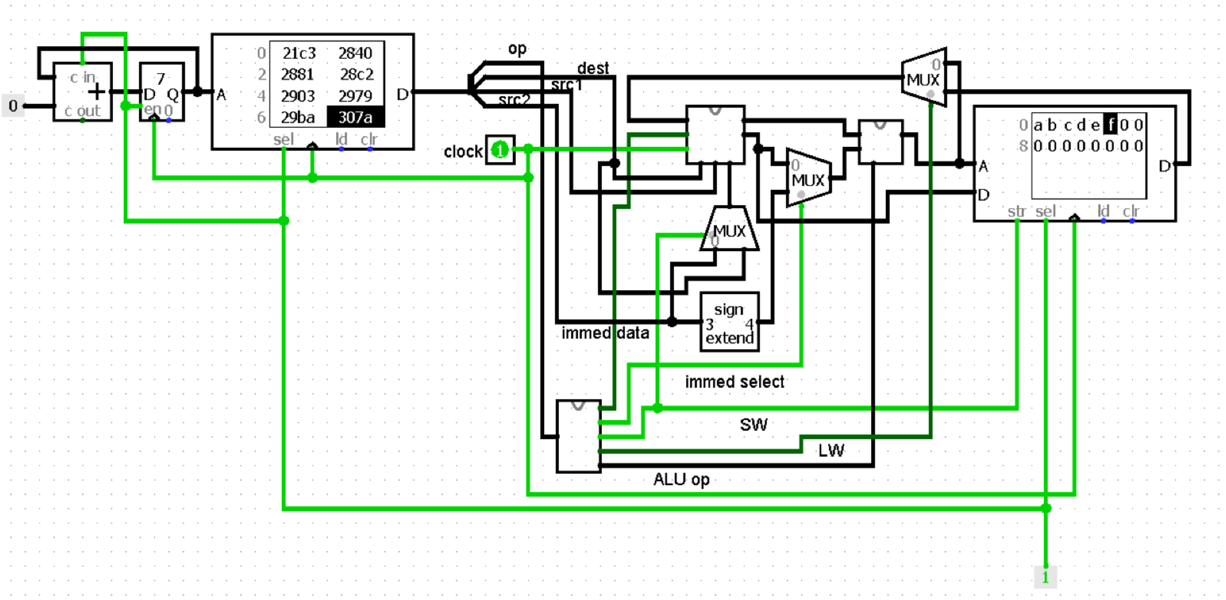
Loaded the commands before the rising cloclkedge

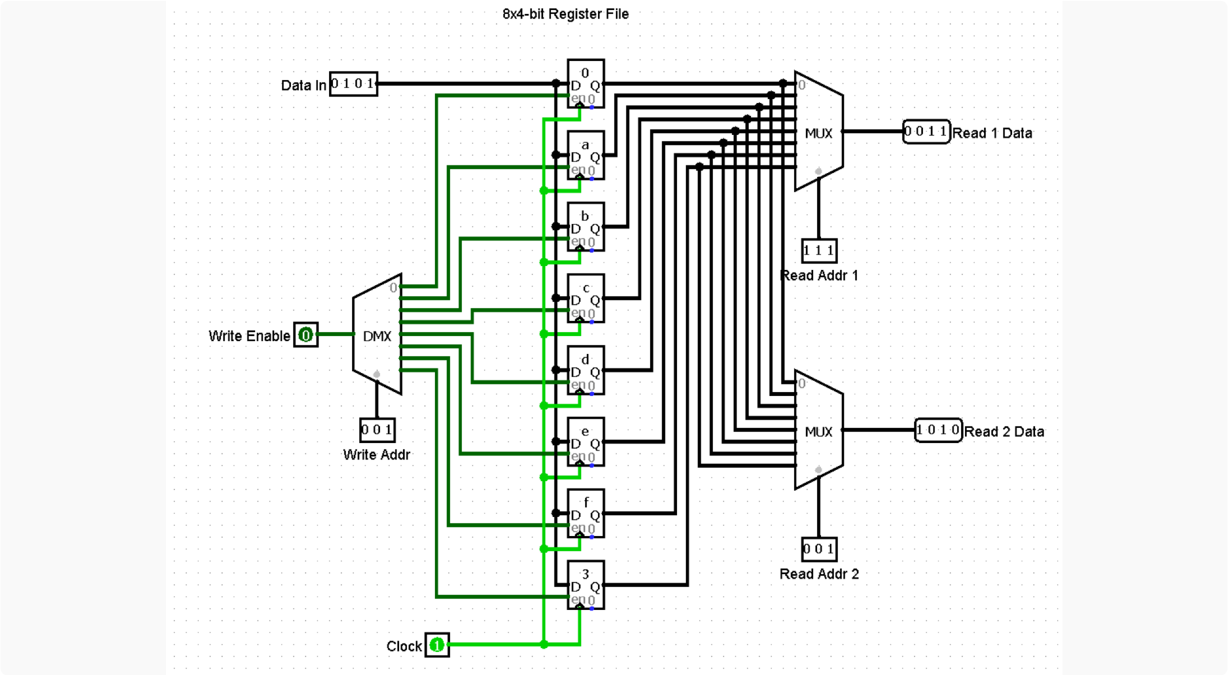




Running some cycles to finish all the load word instructions:

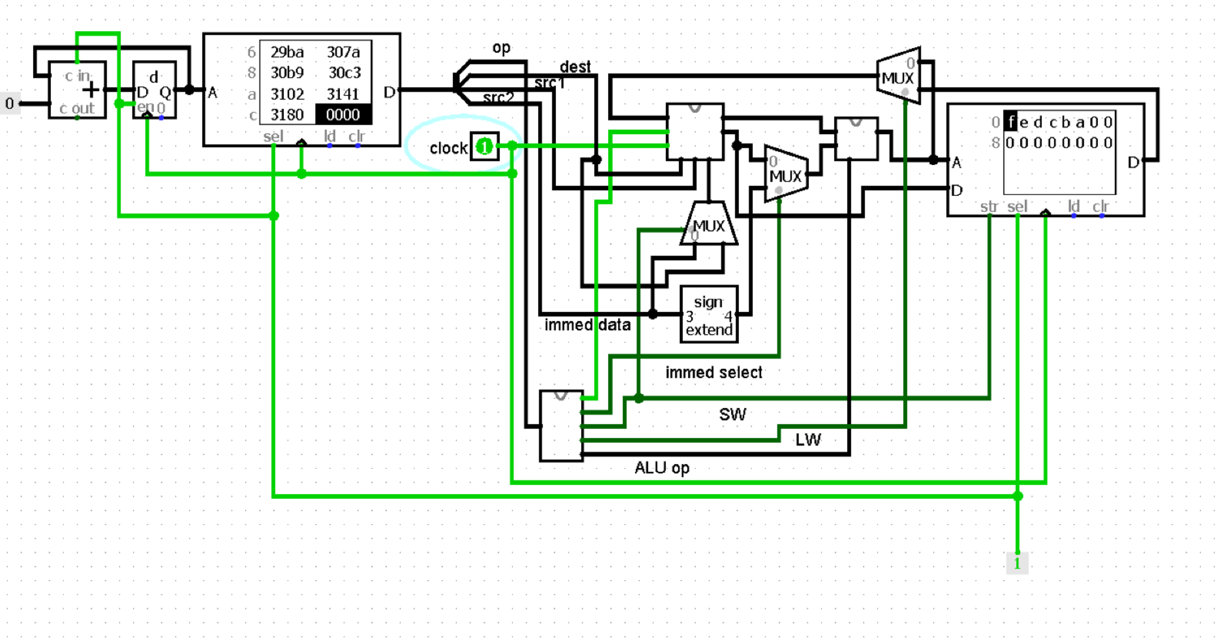
Now all alphabets a through f will be loaded in the registers:





We see that all the alphabets are now stored in the registers. After running the store word instructions, they will be reversed in the data memory.

After some more rising clockedges



After running the store word instructions, we see that all the alphabets are reversed in the data memory.