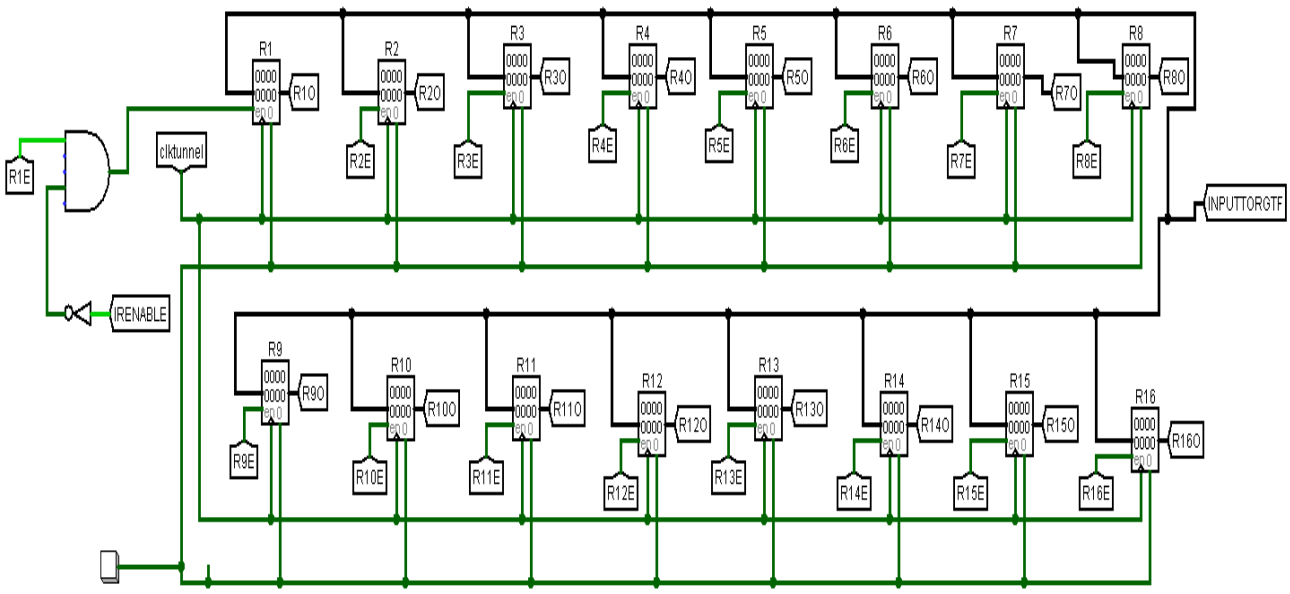
IMPLEMENTATION OF CIRCUIT

KOLLI JOGENDRA DURGA PRASAD

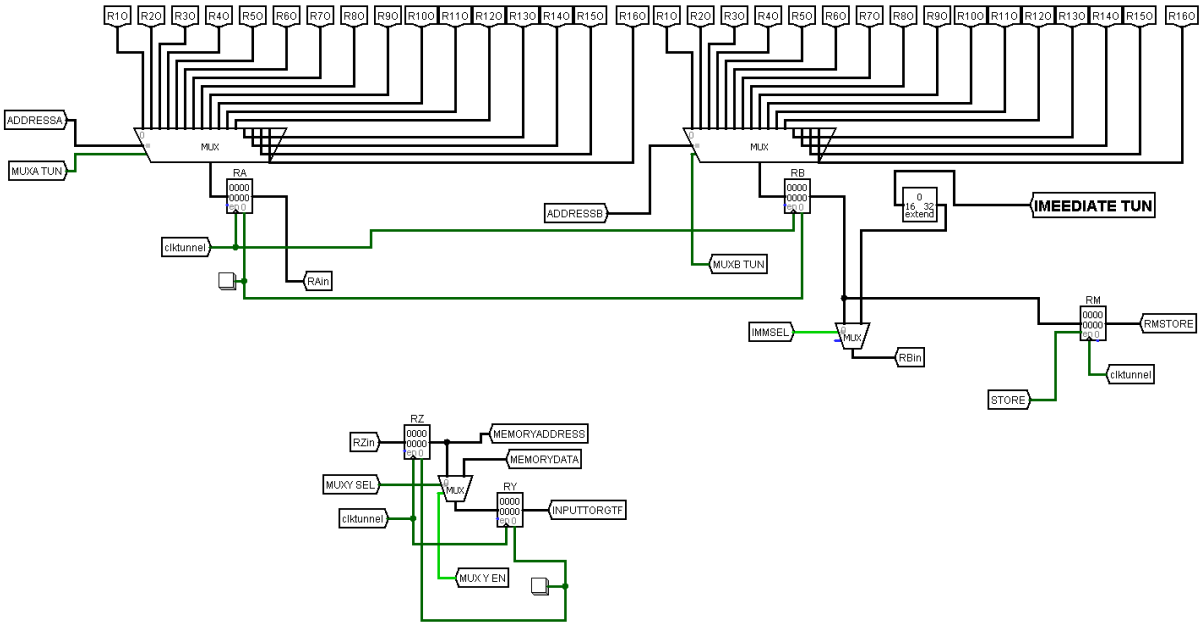
REGISTER FILE



I have implemented it using a 16-registers,we will give the input through the “INPUTTORGTF” tunnel by enabling the required register and we will take the output from each register from RxO tunnel(where x is 1,2,3 - - - 16)

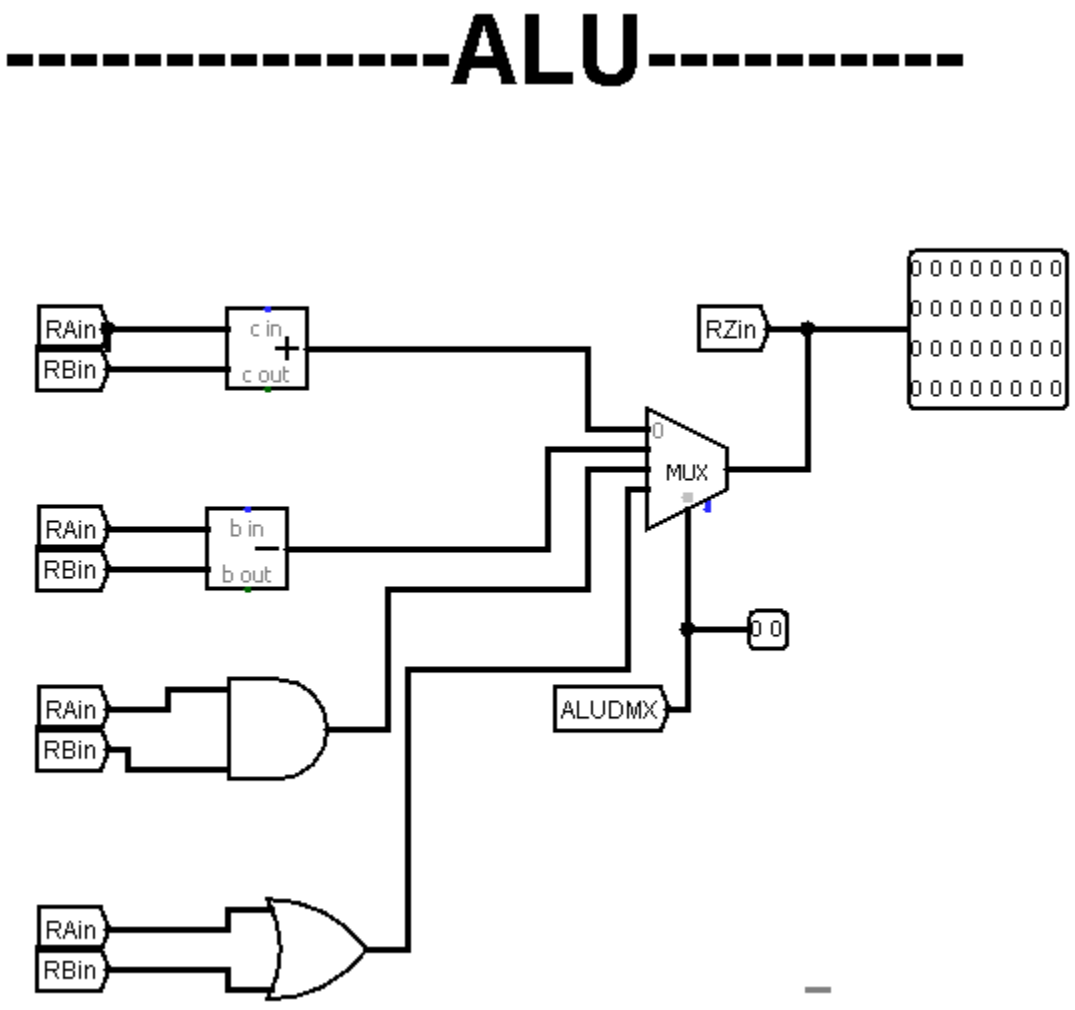
FIVE STAGE PIPELINE

This is the five stage pipeline (except the registerfile) we will get the input from the register file from **RxO** tunnel(where x is 1,2,3 - - - 16) we will send it to RA and RB (if required) or we will take a value from immediate tunnel , then we will send it to the ALU unit through **RAin** and **RBin** and we will get the output from **RZin** tunnel we will select the mux-y according to the instruction, then it goes to RY(not required for store and halt) and then to the register file through "**INPUTTORGTF**" tunnel (not required for store and halt)



ALU

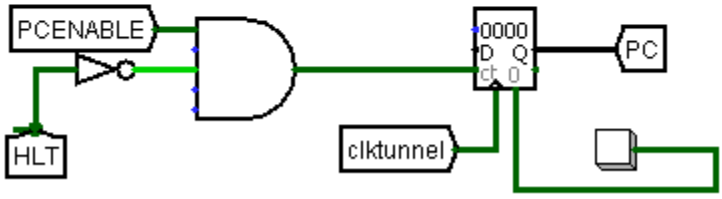
We will take the values from **RAin** and **RBin** tunnels and execute the basic Addition,Subtraction,OR,AND operations through Adder,Subtractor,ORgate and AND gate and we will give the output to the **RZin** tunnel



Program Counter

I have implemented it using a counter which keeps counting when **PC ENABLE** tunnel is selected and stops counting when hlt instruction is given

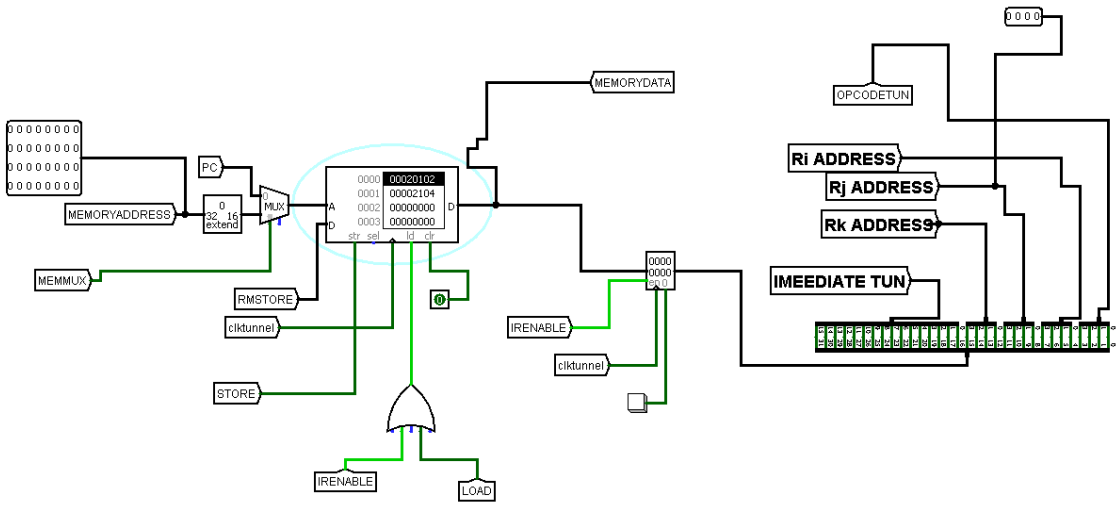
----------PC---------



MEMORY

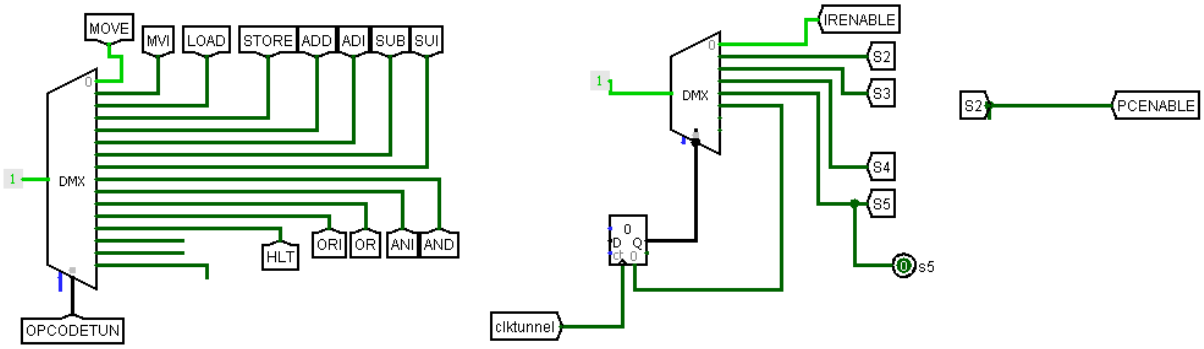
1) It consists of a RAM having a 2^{16} addresses each having 32-bit memory
2) Generally in stage 1 **Program Counter** gives the address of the instruction to be loaded,then mux will select **PC** tunnel , then the instruction will be loaded into the register then it will go to the splitter and goes to the respective tunnels
3) Also in the stage 3 of load mux will select **MEMORYADDRESS** tunnel and gives the output to the **MEMORYDATA** tunnel
4) In stage 3 of store instruction mux will select **MEMORYADDRESS** tunnel and store the value coming from **RMSTORE** tunnel into the memory

----------MEMORY----------



CONTROL SIGNAL

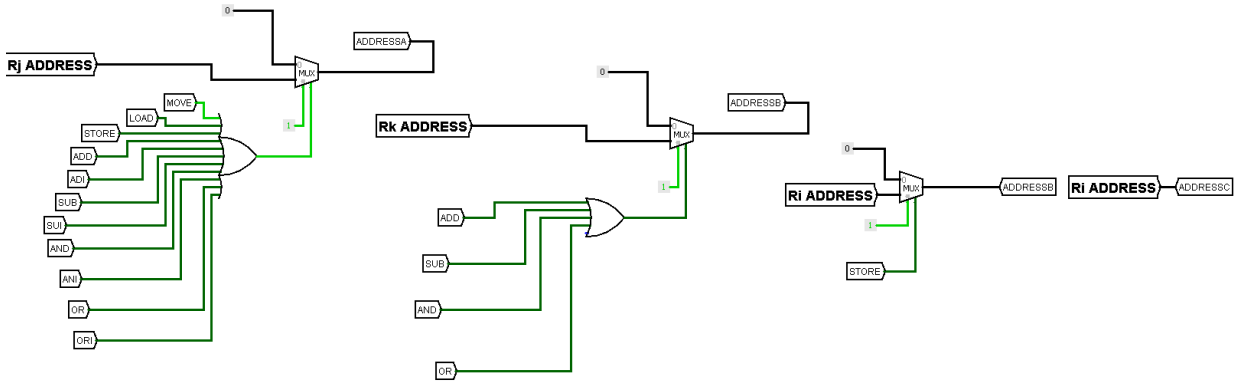
1)



In above we have 2 demultiplexers

1) In the first demultiplexer we will select the required using **OPCODETUN** tunnel which comes from the splitter we used in **MEMORY** section
2) Second one is the counter which is connected to the demultiplexer The counter counts for 5 times then on the 6 th time it's values gets cleared and goes to zero like this, it counts for 5 times in each cycle and selects the respective values on the demux

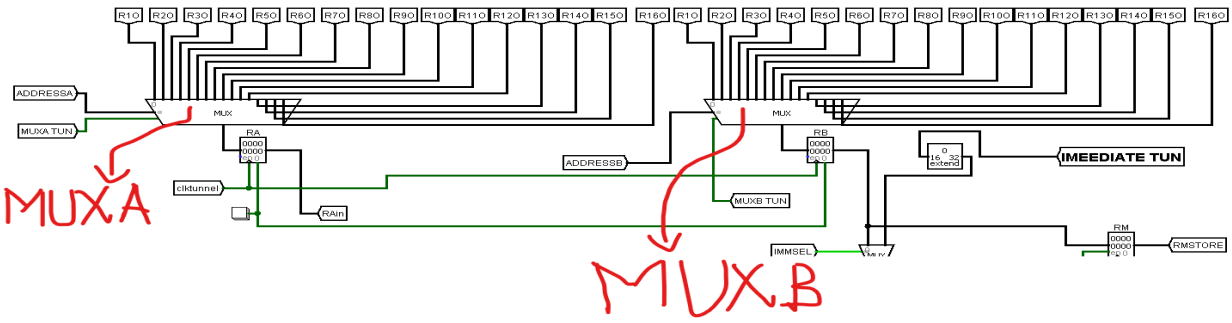
2)



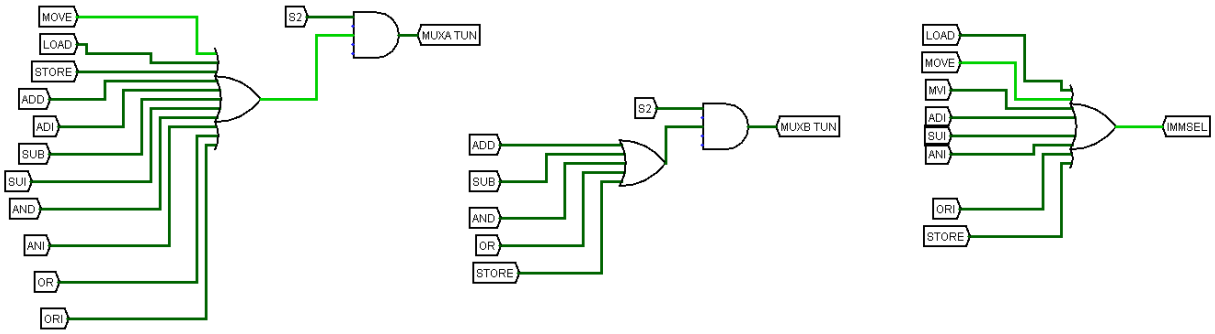
In the above we have

Form of general instruction INS,Ri,Rj,Rk/IMMEDIATE

1) Except for HLT,MVI , for every instruction Rj Address will be given to the ADDRESSA tunnel of MUXA (which is present in the second stage of pipeline)
2) Only for ADD,SUB,OR,AND Rk Address will be given to the ADDRESSB tunnel of MUXB (which is present in the second stage of pipeline)
3) For every instruction Ri ADDRESS will go to ADDRESSC except for halt and store
4) For store Ri ADDRESS will go to ADDRESSB



3)

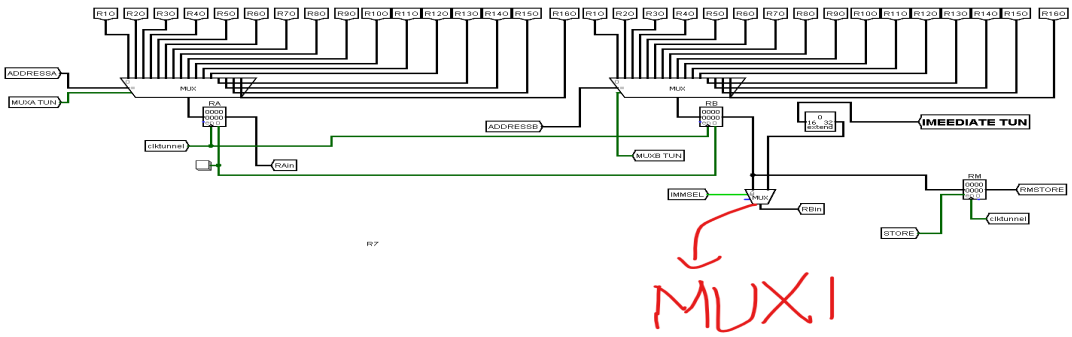


The above is for enabling the MUXA and MUXB using **MUXA TUN** tunnel and **MUXB TUN** tunnel

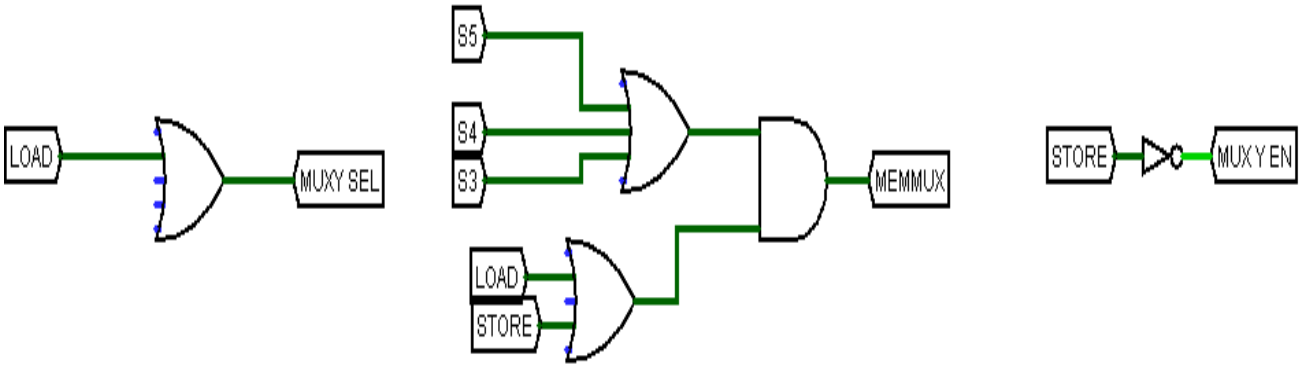
1) We enable MUXA for the following MOVE, LOAD,STORE, ADD,ADI , SUB,SUI, AND,ANI,OR,ORI
2) We enable MUXB for the following ADD,SUB, AND,OR,STORE

We use **IMMSEL** tunnel for selecting the input for MUXI to choose the immediate value instead of value coming from register

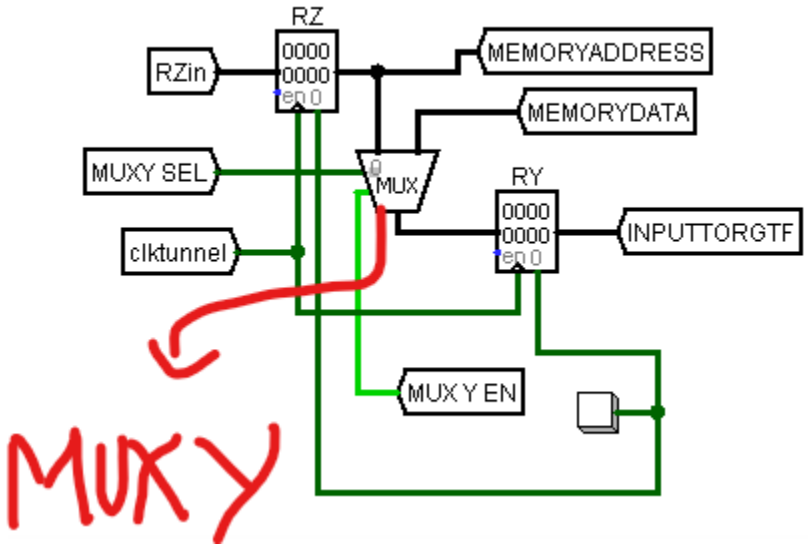
1) We take immediate values for the following MOVE, LOAD,ADI ,SUI, ANI,ORI,LOAD,MVI
2) we take immediate for MOVE as 0 just to bypass the ALU by adding 0 to RA value



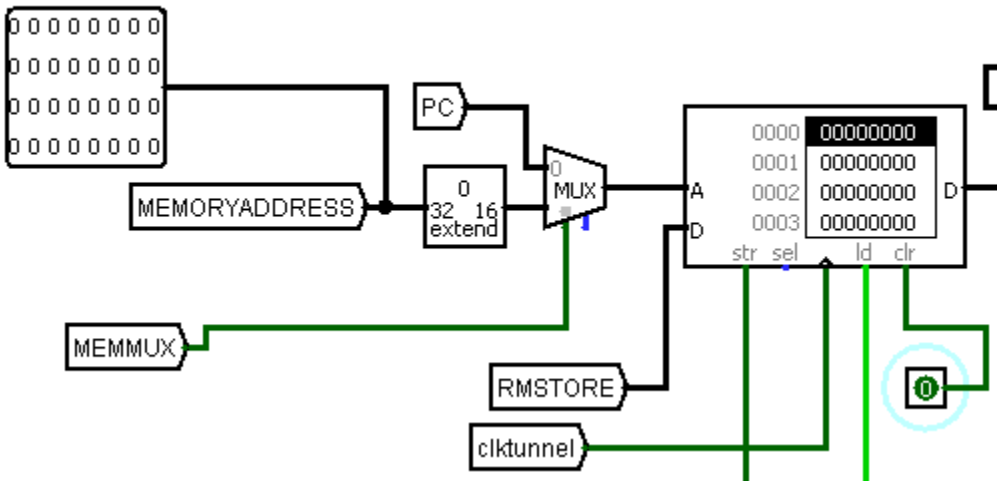
4)



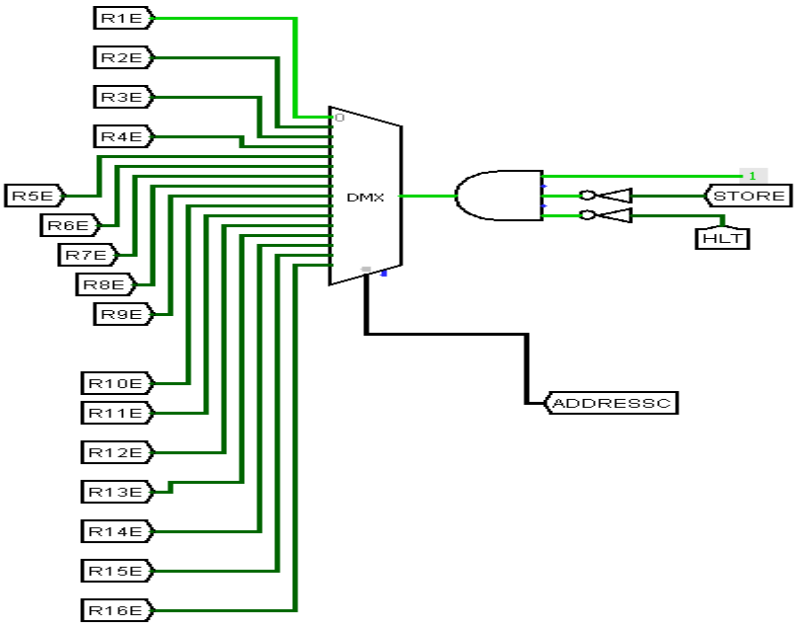
1) **MUXY SEL** tunnel is used for selection of MEMORYDATA instead of RZ register value .we take MEMORYDATA only for LOAD
2) We disable MUXY for Store (means **MUXY EN** =0)



3)**MEMUX** tunnel which is used in the memory section selects **MEMORY ADDRESS** instead of **PC.** we use that only for LOAD,STORE



5)



ADDRESSC is used to enable the register in which we will store the final input (In STORE and HLT we won't write back to the register)

HEXCONVERTER

This is used to convert our binary code to the HEX CODE we have to enter the values into particular slots using our **encoding scheme(which is in the encoding pdf)**