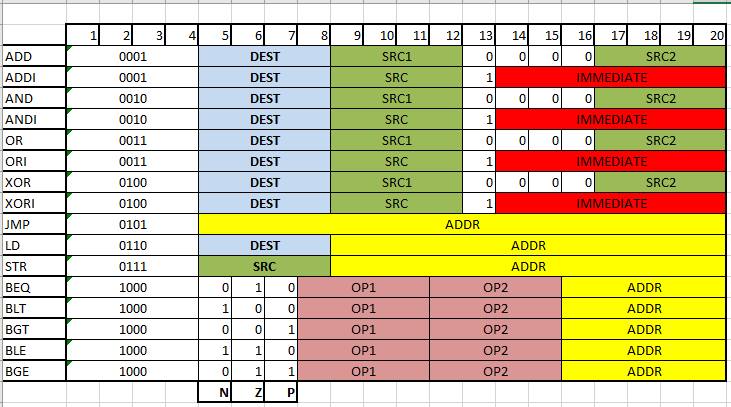
**CSE 315 DIGITAL LOGIC DESIGN TERM PROJECT REPORT**

**Main Goal Of The Project:**

Our main goal in this project is to design a processor for the following situations. (AND, OR, ADD, LD, ST, ANDI, ORI, ADDI, XOR, XORI, JUMP, BEQ, BGT, BLT, BGE, BLE). Processor will have 20 bit input. 10 bits are going to use by showing the adresses. There are 5 parts in the processor. **Register File** will hold register values and signal to write into any register. There will be 16 registers in processor. **Instruction Memory** will be a read-only memory and instructions will be stored in this component. If the current instruction is not one of the J,BR; the next instruction will be fetched and executed consecutively from this memory. **Data Memory** will be read-write memory which will store data. Program will be able to read data from data memory, and also store data to this memory. **Data Memory** will have 10 bits address width, and 20 bits data width. **Control Unit** will produce proper signals to all datapath components. And the last **Arithmetic Logic Unit (ALU)** will compute arithmetic operations ADD,OR,XOR,AND,ADDI,ORI,XORI,ANDI. In addition every instruction have unique operational code.

Here it is our instruction set. Just branches have same operational code. As you can see in the table.



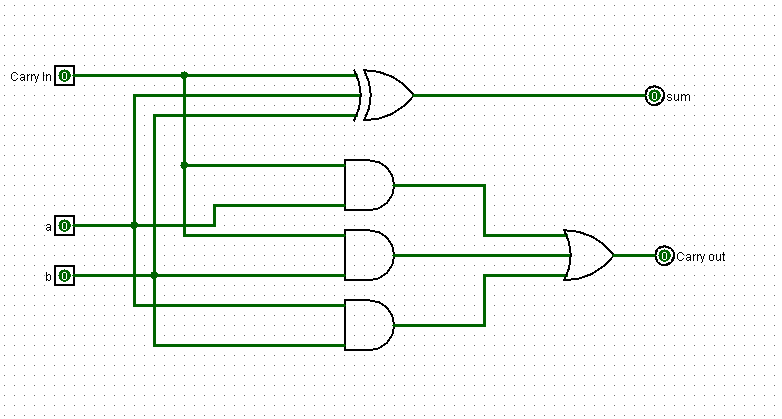
**First Iteration:**

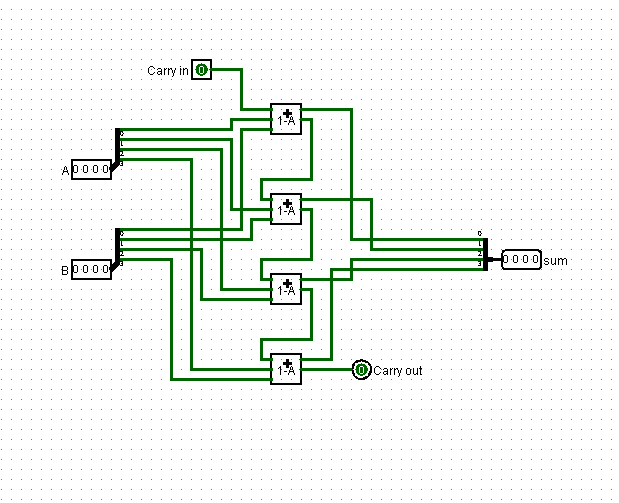
We write java code. Code gets an input from user. For example;

* ADD R7,R5,R6 this instruction makes R7 = R5 + R6 and then returns 20 bit width hexadecimal value which is 0x17506 in our code.
* Also you can look input.txt for user inputs and the Instruction Memory.txt for the outputs.

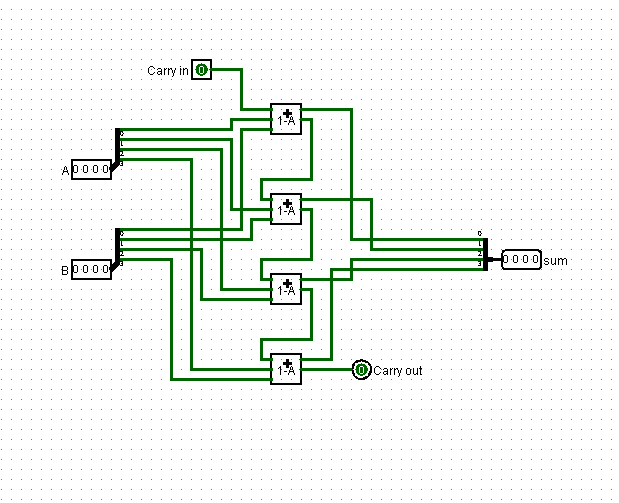
**Second Iteration:**

In this iteration we implement the logic circuits of processor on Logisim software.

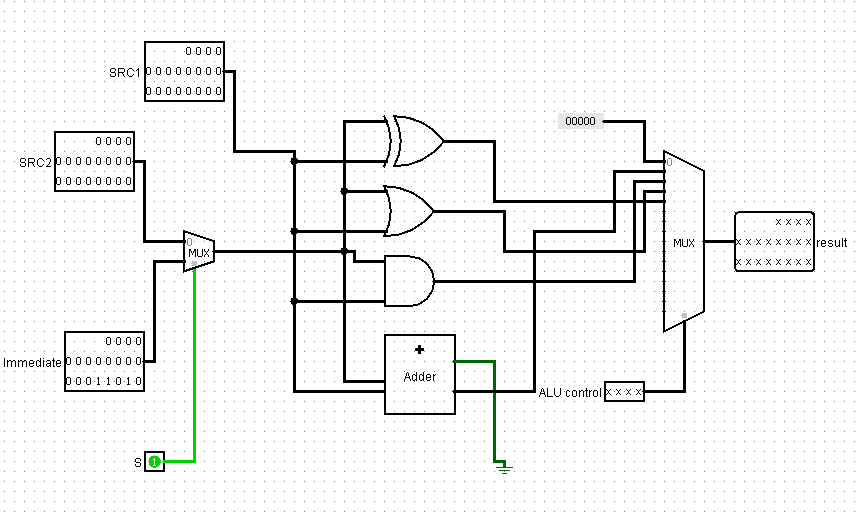
1-) 1 bit full adder

2-) 4 bit adder

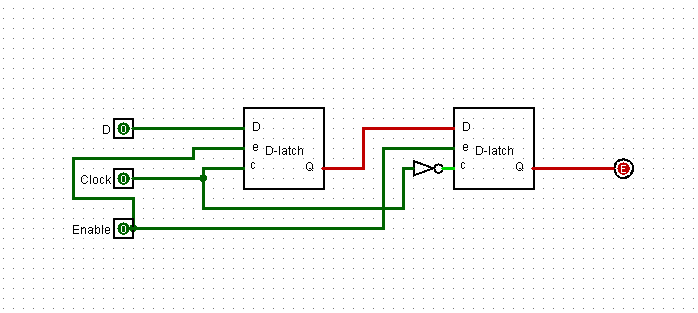
3-) 20 bit adder



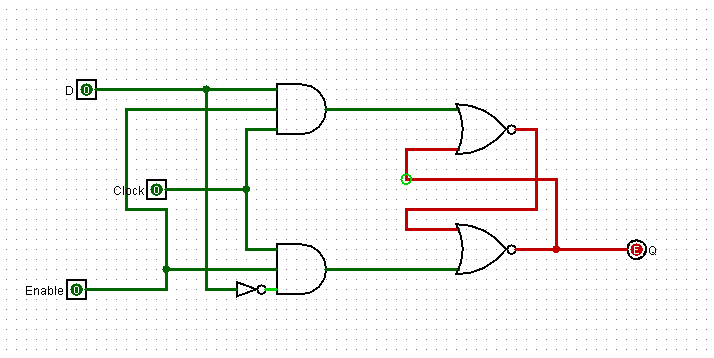
4-) ALU



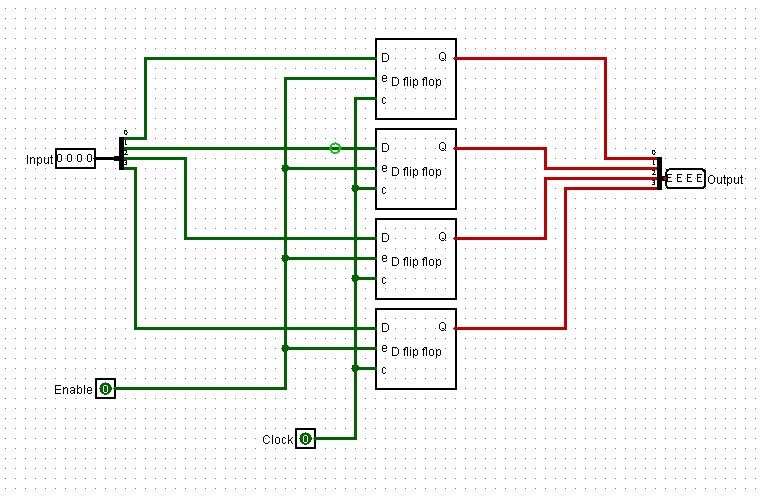
5-) D flip flop



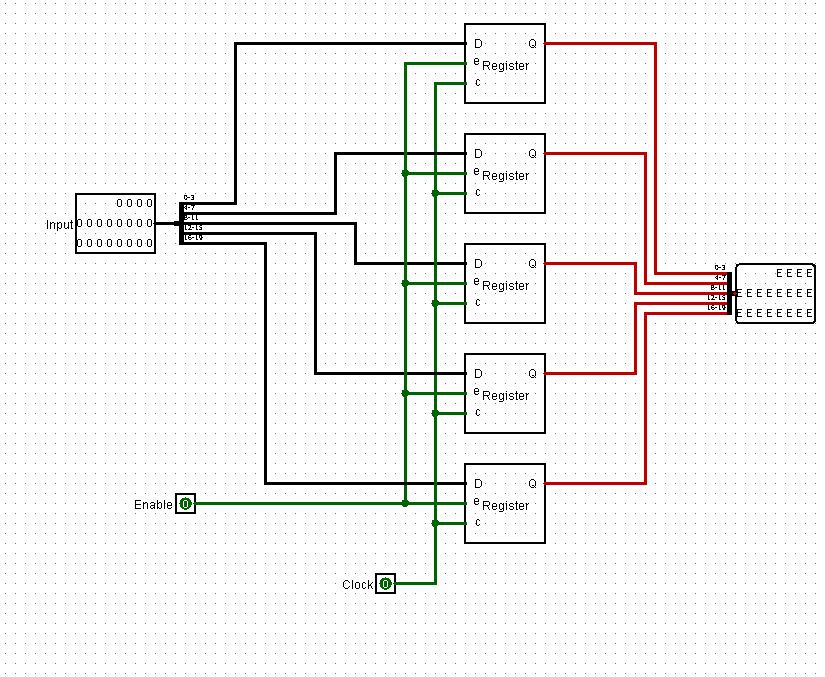
6-) D latch



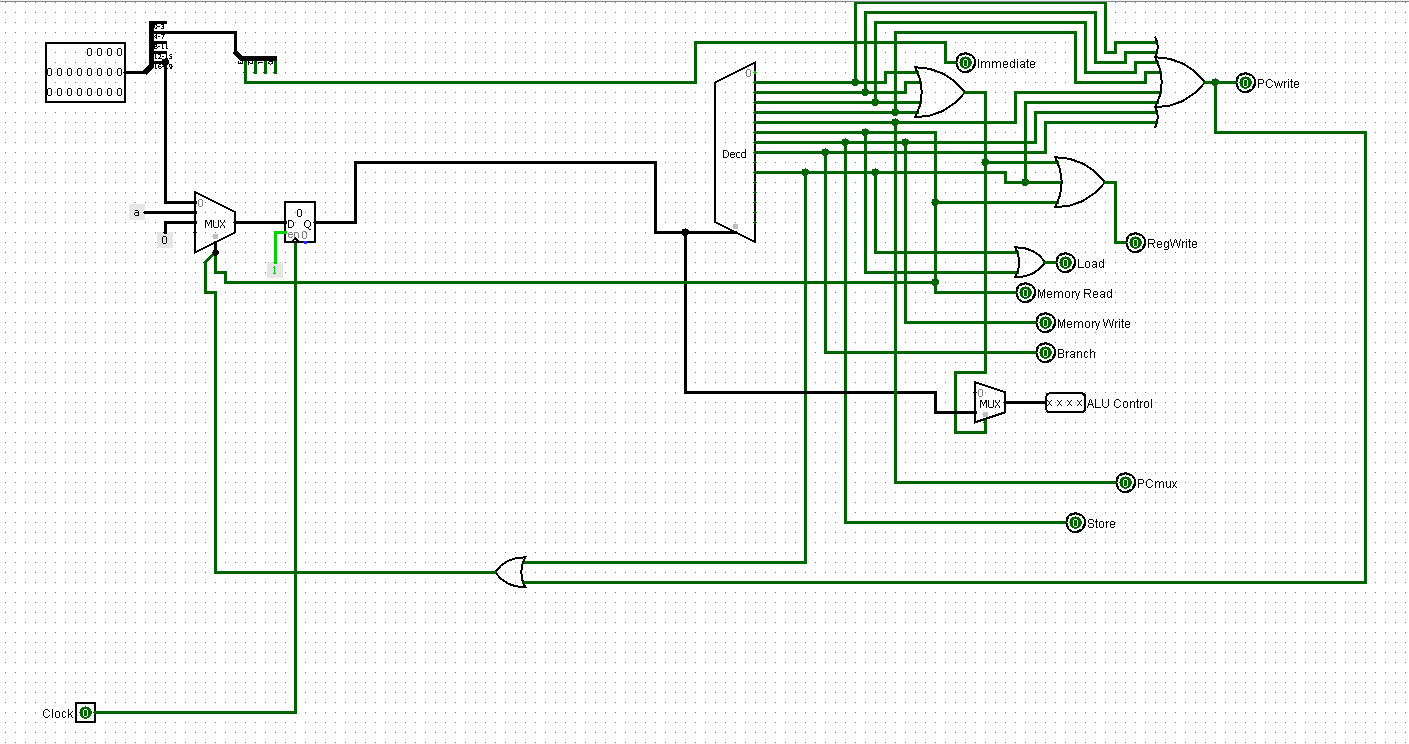
7-) 4 bit register



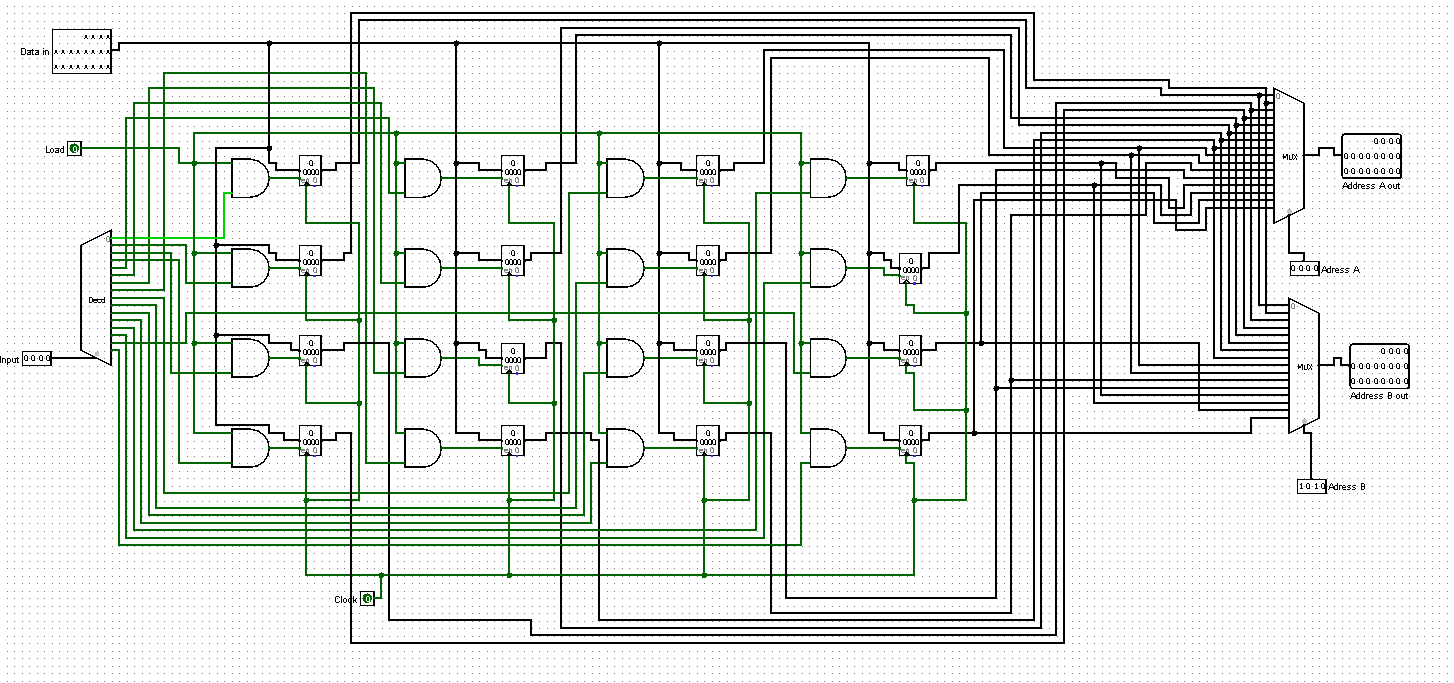
8-) 20 bit register



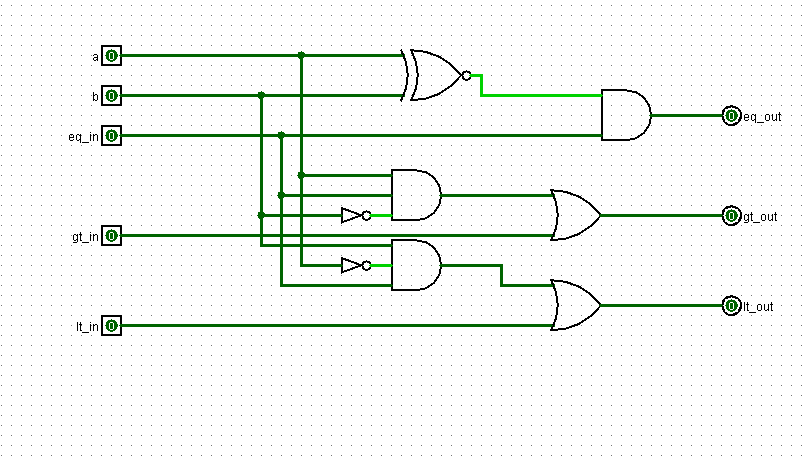
9-) Control Unit



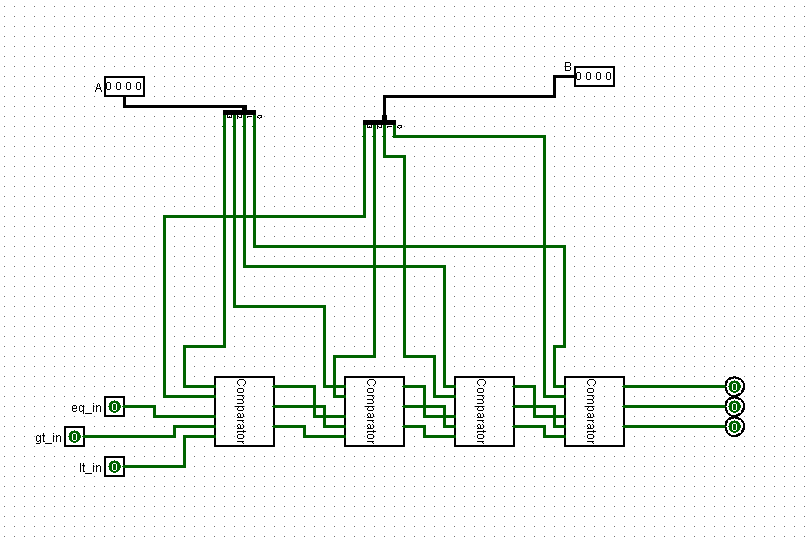
10-) Register File



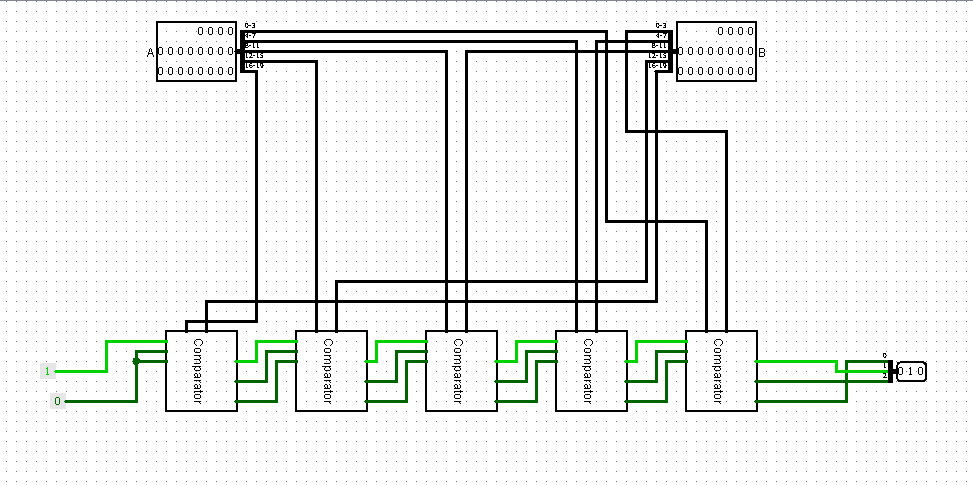
11-) 1 bit comprator



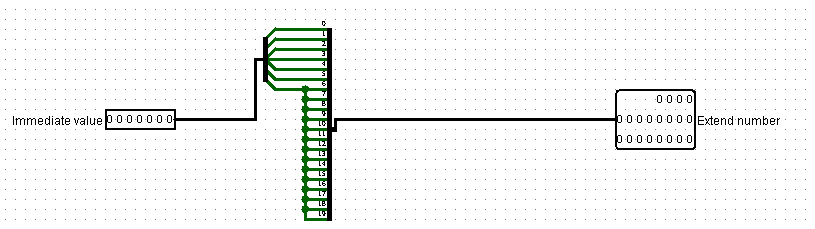
12-) 4 bit comprator



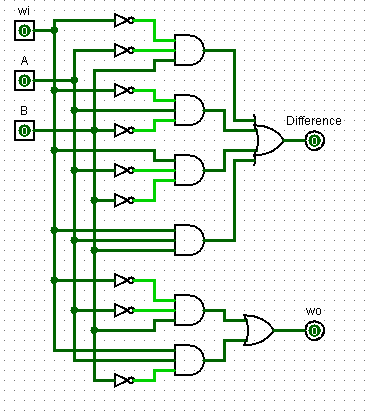
13-) 20 bit comprator



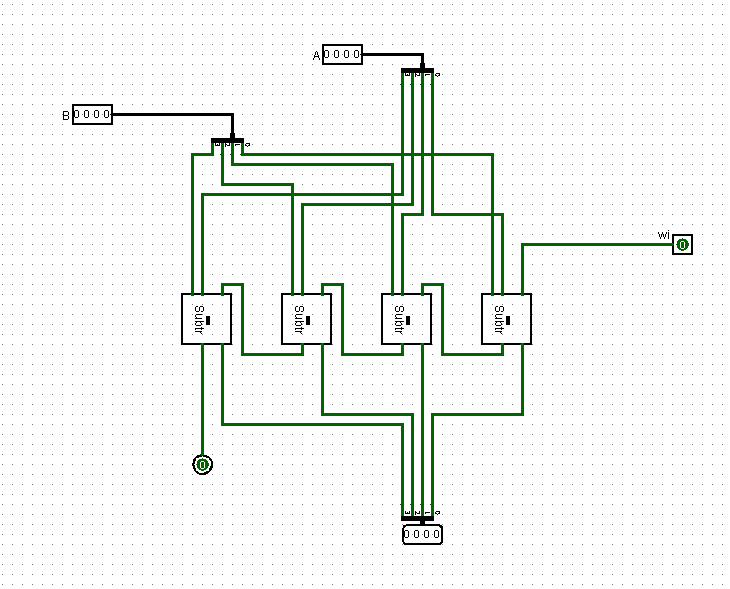
14-) Sign extend 7 – 20



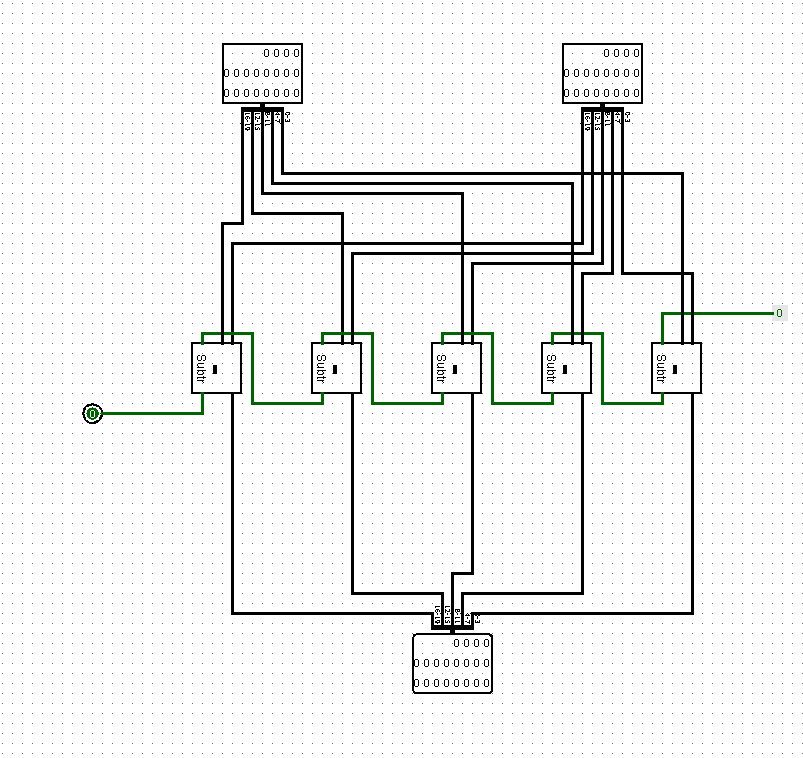
15-) 1 bit substractor



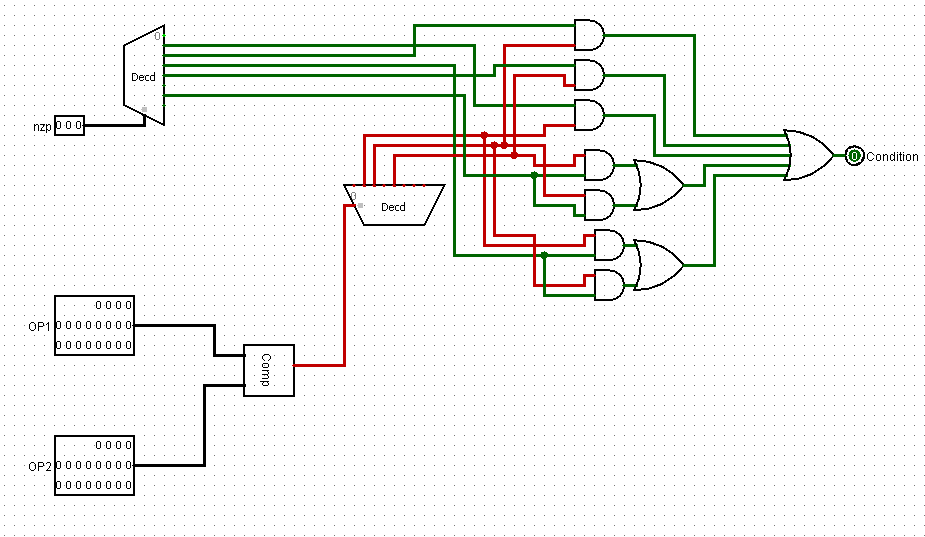
16-) 4 bit substractor



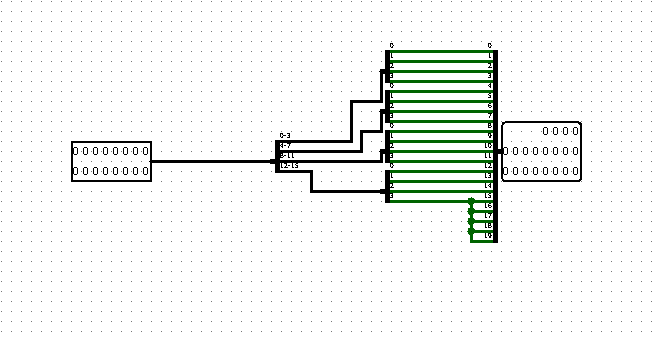
17-) 20 bit substractor



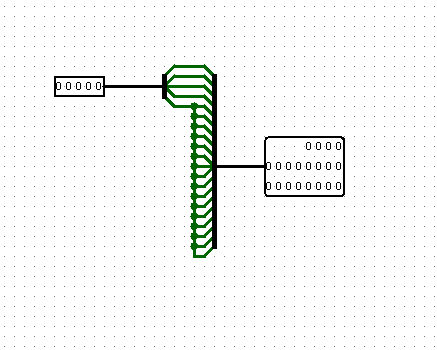
18-) Branch



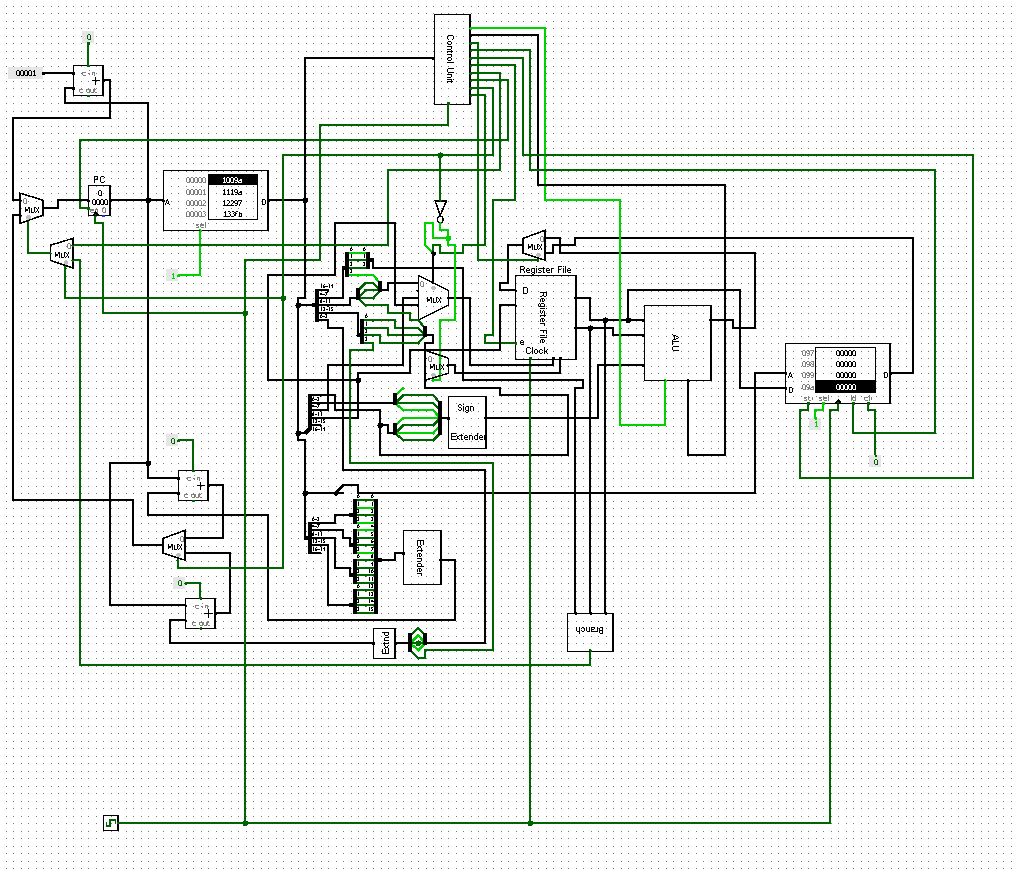
19-) Sign extender 16-20



20-) Extender 5 – 20



21-) Main



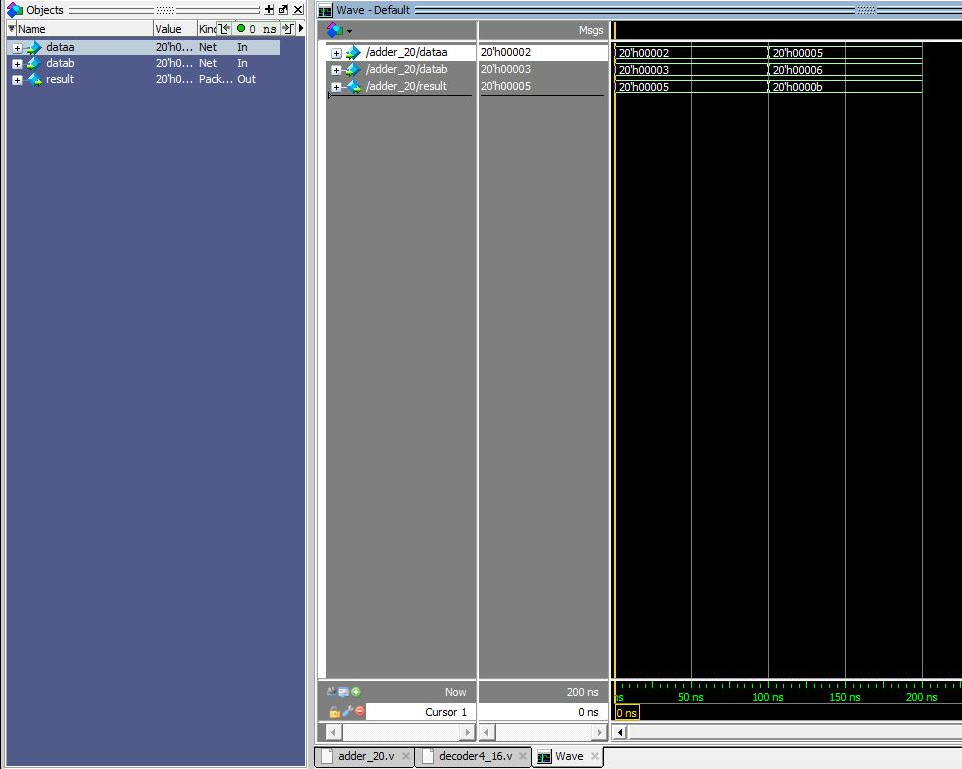
**Third Iteration:**

In this iteration we used Verilog HDL software to design processor. Each component of our Logisim design are defined in Verilog.

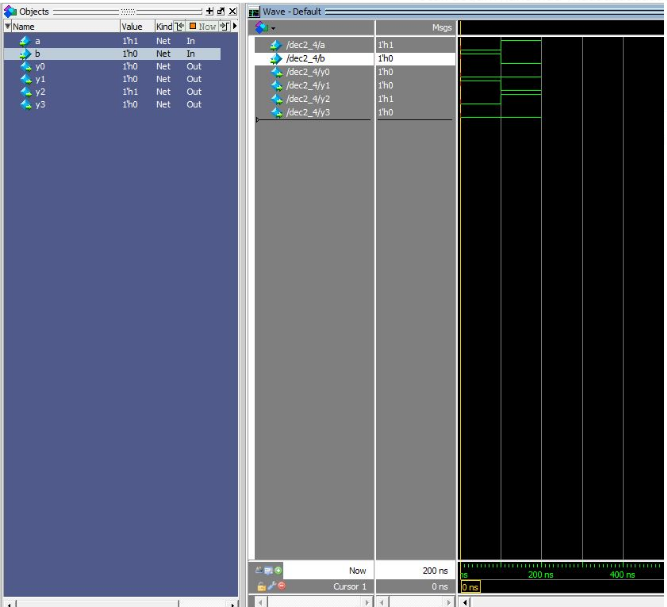
**Output Screenshots in Verilog:**

**1-) Adder**

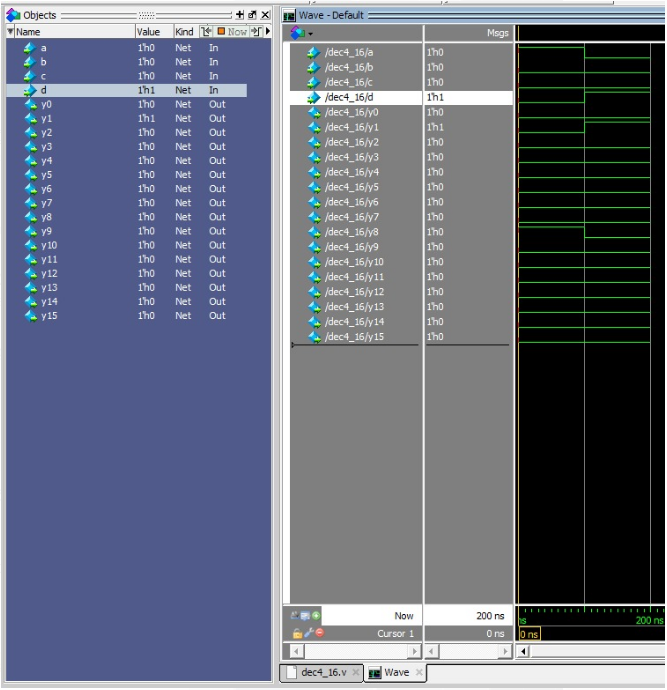
**Adding two inputs.**



**2-) Decoder (2x4)**



**3-) Decoder (4x16)**



**4-) ALU**

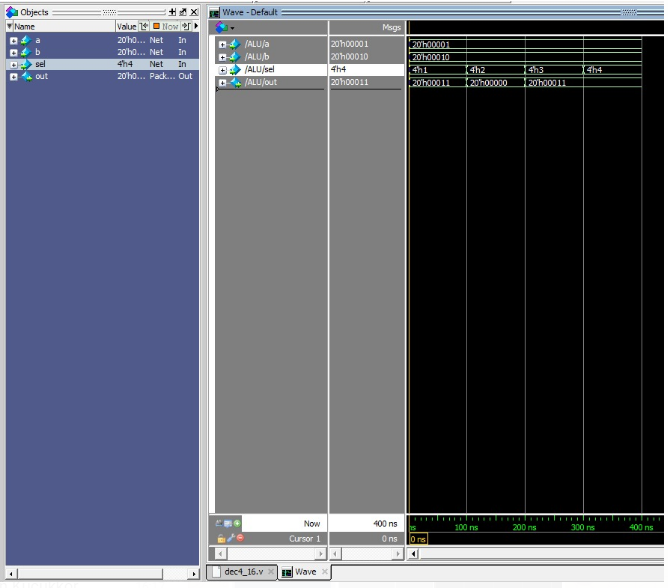
**A=1 B=2**

**When the selection bits is: 0001 => add , output = 3**

**When the selection bits is: 0010 => and , output = 0**

**When the selection bits is: 0011 => or , output = 3**

**When the selection bits is: 1000 => xor , output = 3**



**5-) Branch**

**A=1 B=1**

**When the selection bits is: 010 => equal condition , output = 1**

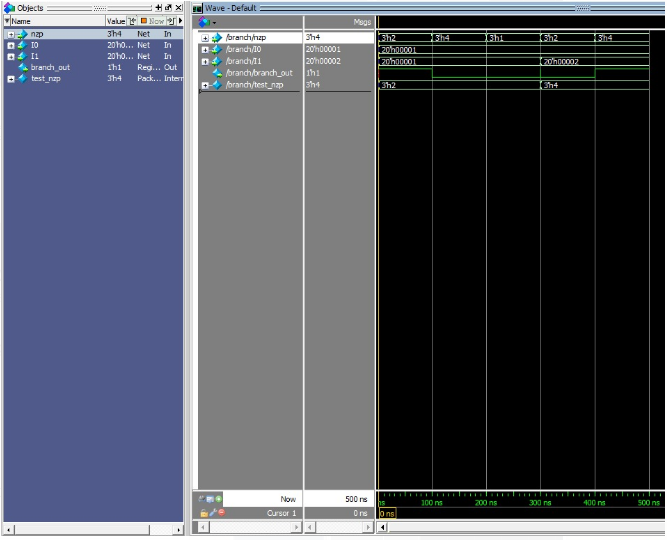
**When the selection bits is: 100 => less condition , output = 0**

**When the selection bits is: 001 => greater condition , output = 0**

**A=1 B=2**

**When the selection bits is: 010 => equal condition , output = 0**

**When the selection bits is: 100 => less condition , output = 1**

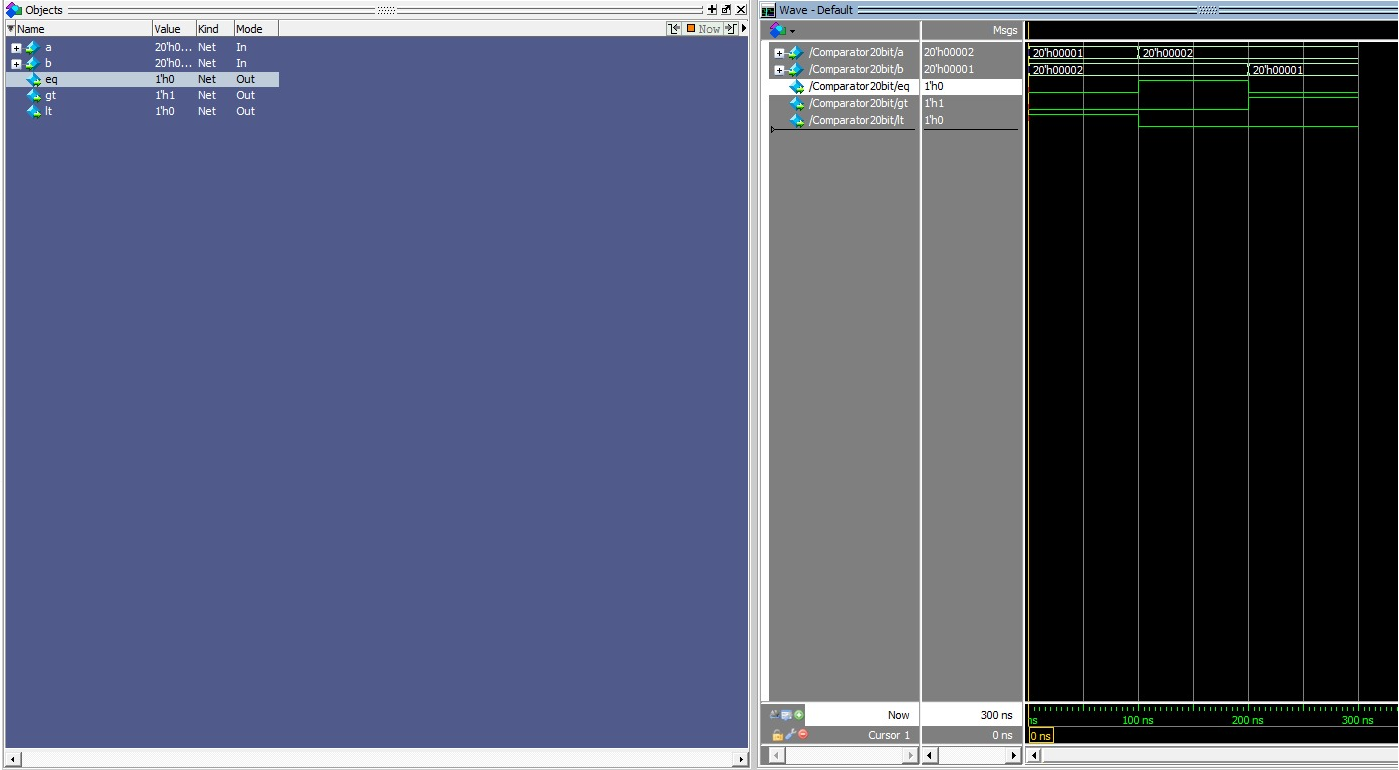


**6-) Comprator**

**A=1 , B=2 => output = less**

**A=2 , B=2 => output = equal**

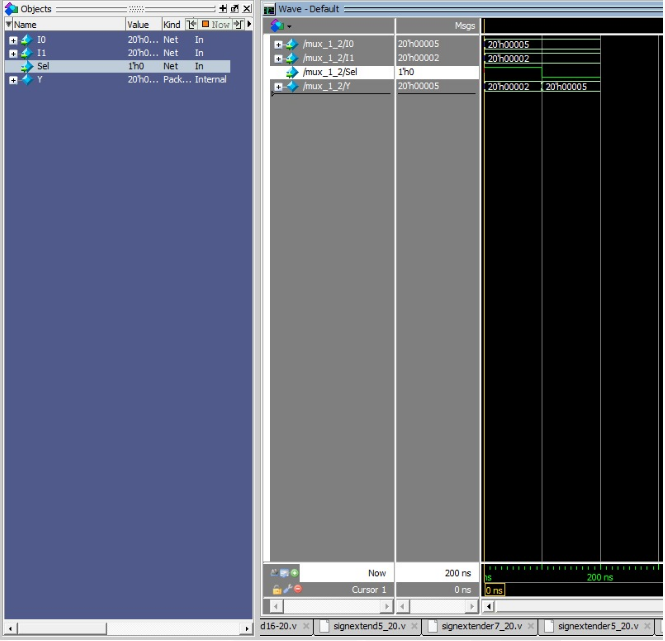
**A=2 , B=1 => output= greater**



**7-) Mux(1x2)**

**İf selection bit = 0 ,output =first input (I0)**

**İf selection bit = 1 ,output = second input (I1)**



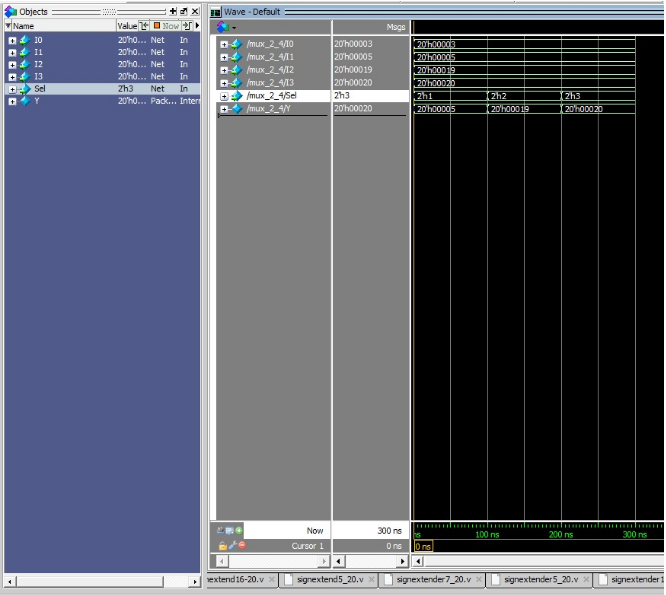
**8-) Mux (2x4)**

**İf selection bit = 00 ,output =first input (I0)**

**İf selection bit = 01 ,output = second input (I1)**

**İf selection bit = 10 ,output =first input (I2)**

**İf selection bit = 11 ,output =first input (I3)**



**9-) RAM**

**Inputs: Clock=0, Write=1, Load=0, Addres=1, Data=20’h00015**

**Data is not written in the RAM. Since clock=0.**

**Inputs: Clock=1, Write=1, Load=0, Addres=1, Data=20’h00015**

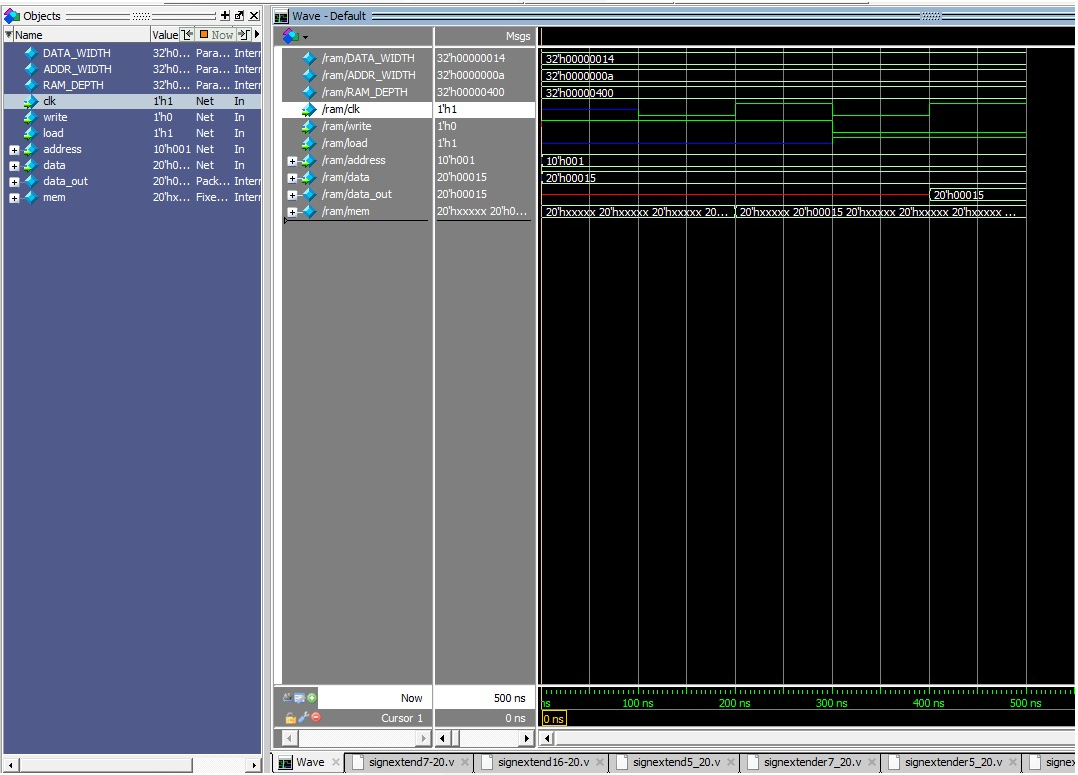
**Data is written in the RAM. Since clock=1.**

**Inputs: Clock=0, Write=0, Load=1, Addres=1, Data=20’h00015**

**Data is not loaded into Data\_out since clock=0**

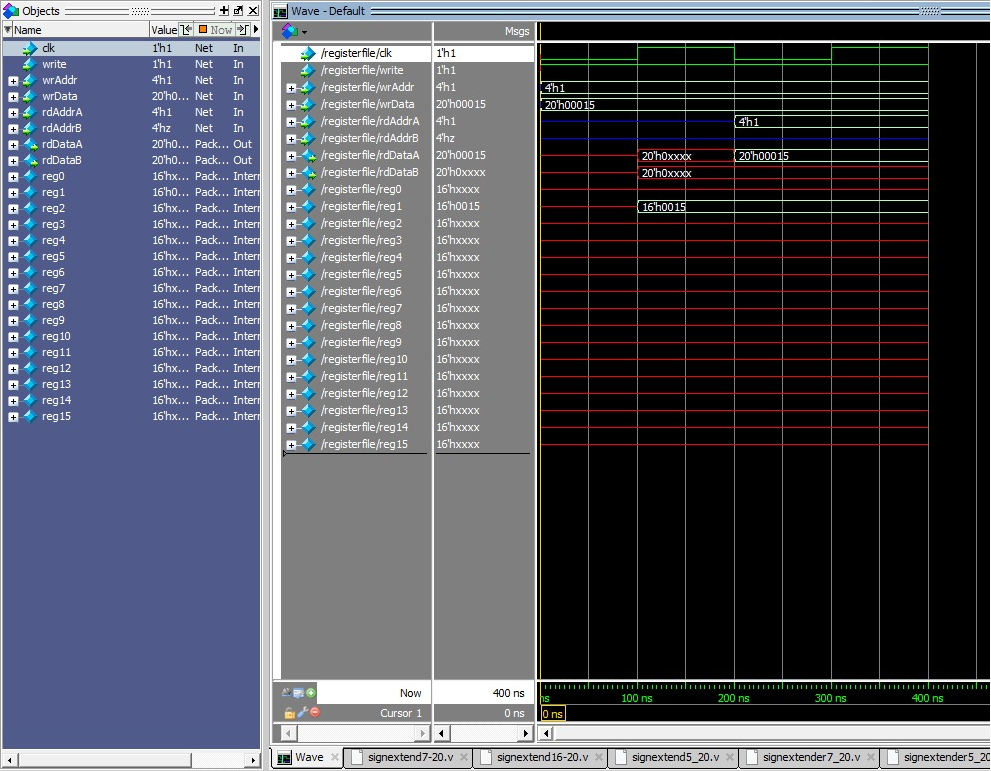
**Inputs: Clock=1, Write=1, Load=0, Addres=1, Data=20’h00015**

**Data is written in the RAM. Since clock=1. And Data\_out=15.**



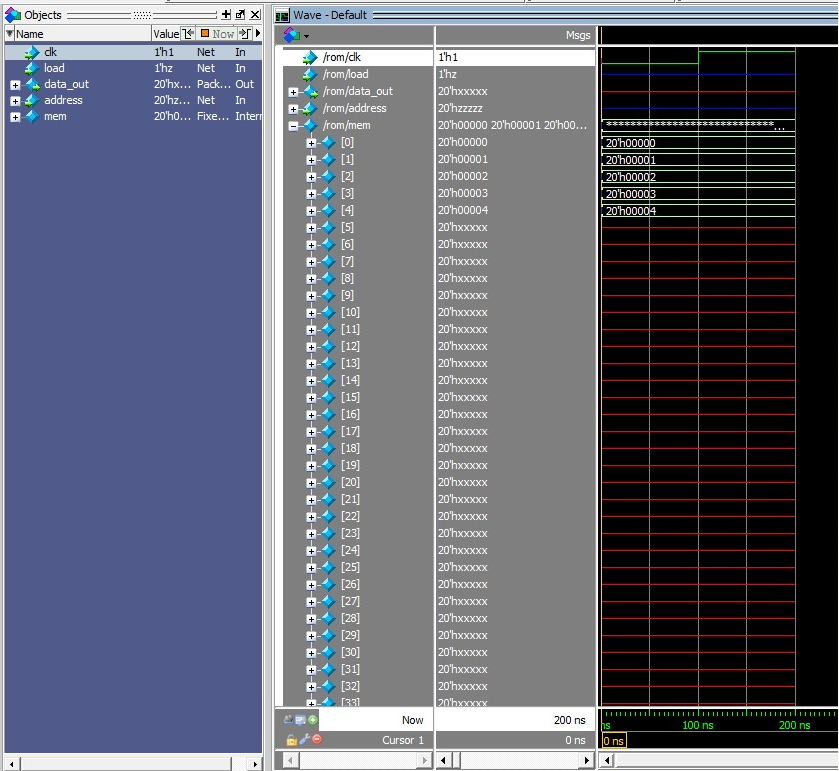
**10-)Register File**

* **When clock=0 , write=1, wrAddr = 1 , wrData = 15 , Data is not written in the register since clock=0**
* **When clock=1 , write=1, wrAddr = 1 , wrData = 15 , Data is written in the register since clock=1**
* **When clock=0 and both clock=1 write=0, rdAddrA = 1 , wrAddr = 1 , wrData = 15 , The data which stays in register1 is loaded into rdDataA**



**11-) ROM**

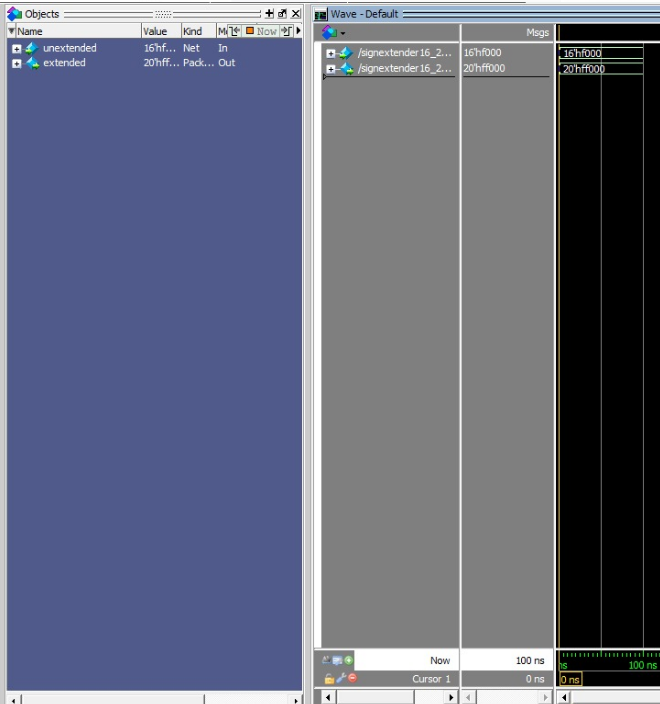
**Instructions taken from .txt file written to the ROM.**



**12-) Sign Extender (16 to 20)**

**Unextended: 16’hf000**

**Extended: 20’hff000**



**Also we used 5 – 20 and 7-20 since they work in the same way we did not included their sreenshots.**