4/1/2021

|  |
| --- |
|  |



|  |  |
| --- | --- |
| ALISHA SAJID  19B-074-EE | ***RISC V SINGLE SYCLE CORE ON LOGISIM*** |

***RISC V SINGLE CYCLE CORE ON LOGISIM***

***INTRODUCTION:***

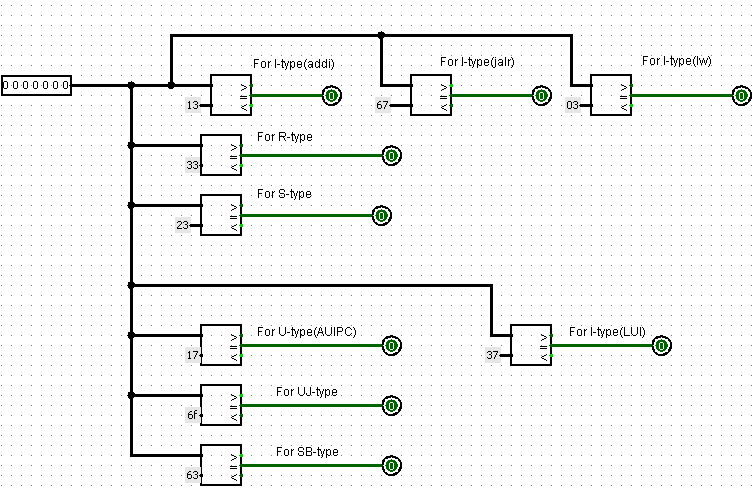
This is a RISC V single cycle core implemented on a logic simulator ***LOGISIM,*** developed on RV32I instruction set.It is based on the synchronous phenomena since on each rising edge, only singe instruction is being executed. The circuit consists of 32-bit ALU, 32-bit data bus, 16KB RAM/ROM, and 12-bit address bus for RAM. It has register file which consists of 32 registers each having data width of 32 bits.

***SUB CIRCUITS:***

1. TYPE DECODE:

This sub system is made to direct the microprocessor to work according to the type of instruction provided. It basically decodes the type of instruction that a user provide.

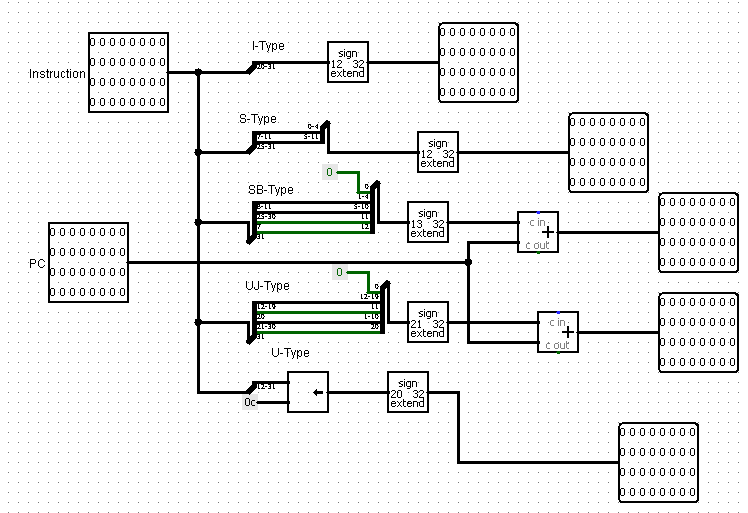
The circuit somewhat looks like:



1. IMMEDIATE GENERATION:

It is a sub system that is designed to generate the immediate in instruction codes if present. There can only be two types of immediates; 12 bit for example, in an I type instruction, the immediate is of 12 bits. 20 bit immediate; for example, in U and UJ type, the immediate is of 20 bits.

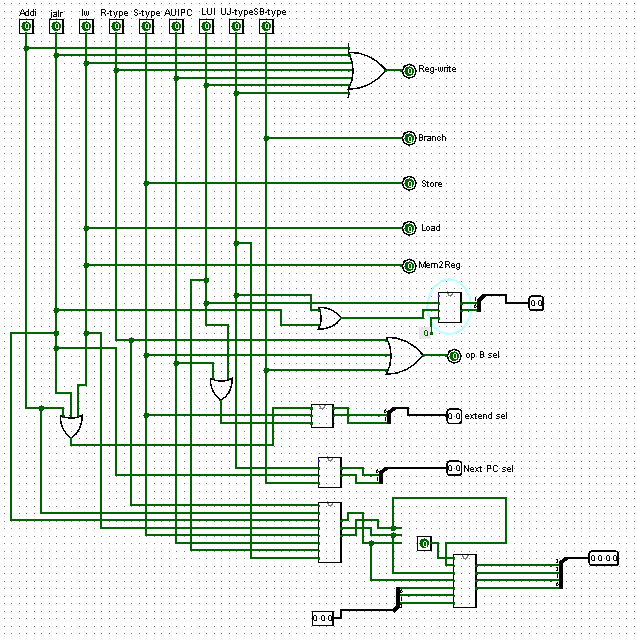
The circuit looks like:



1. CONTROL DECODE:

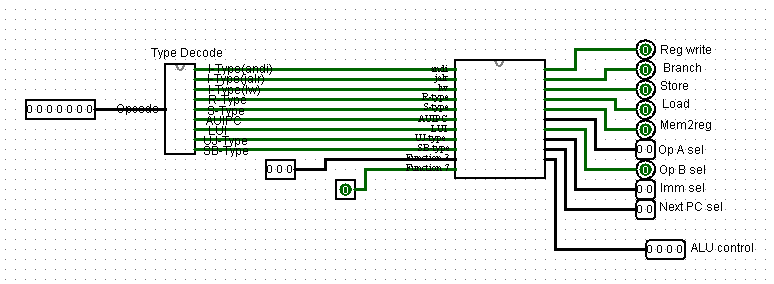
This unit is used to decode how the instructions would work if one instruction is executed at a time or maybe two or more than two instructions are executed simultaneously.

The circuit looks like:



1. CONTROL UNIT:

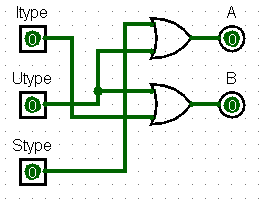
The basic system which consists of both type decode and control where all the decoding is happening. It looks like:



1. IMMEDIATE SELECT:

An extremely simple circuit made using the combinational analysis option in window tab. It is used in order to select the last bit of immediate for either of the three types namely, I type, U type and S type.

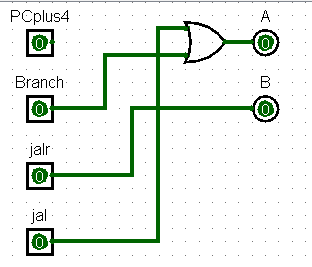
It looks like:



1. NEXT PC SELECT:

The main circuit which is the top consists of a program counter commonly called as PC which holds the address of the instruction. Similarly, next PC select generates the PC for the next instruction to be executed. Also made using the same combinational analysis option of logisim.

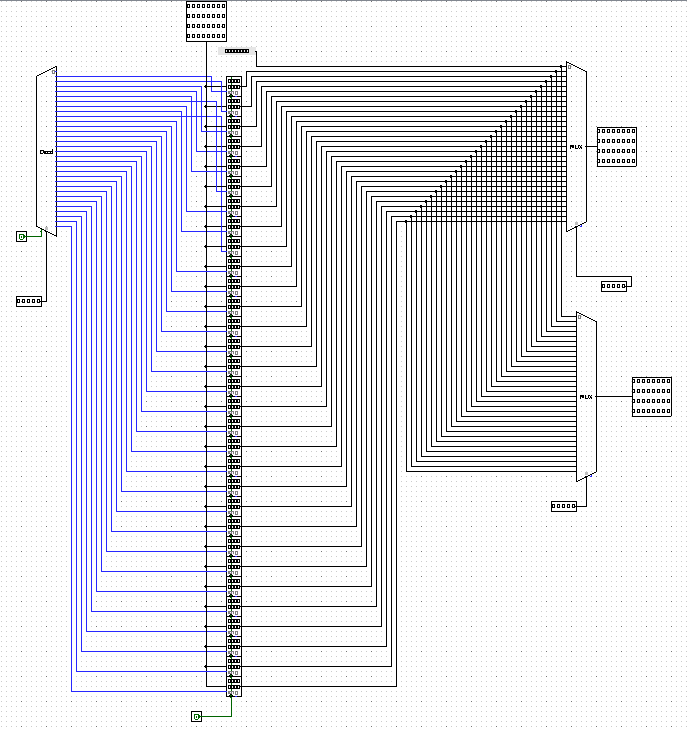
The circuit is:



1. REGISTER FILE:

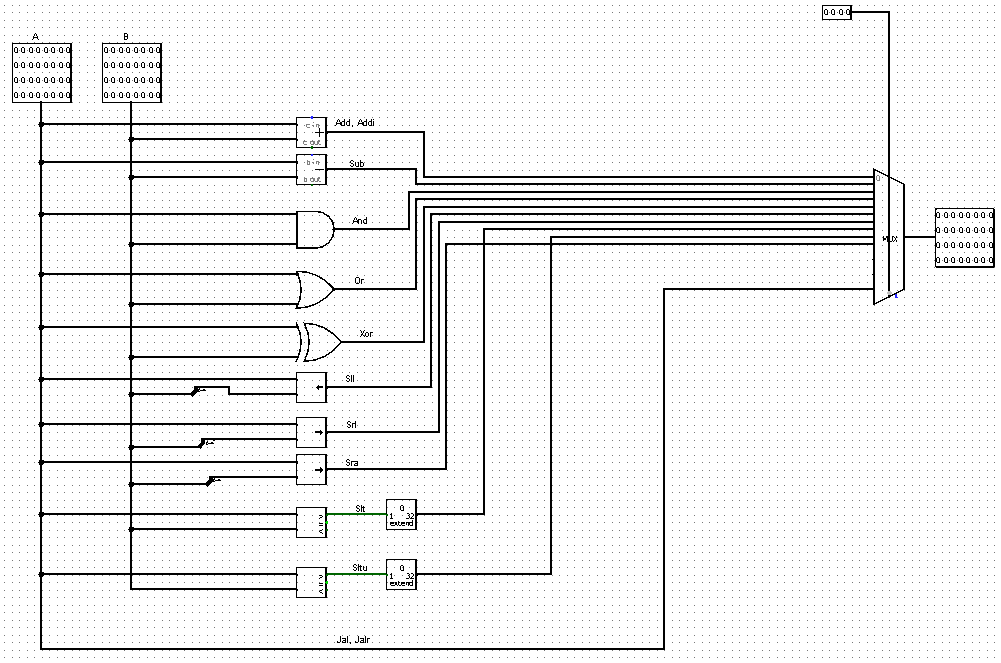
A 32 bit register file which holds both the source registers (may be a single source register) and destination register for the instruction.

The circuit looks like:

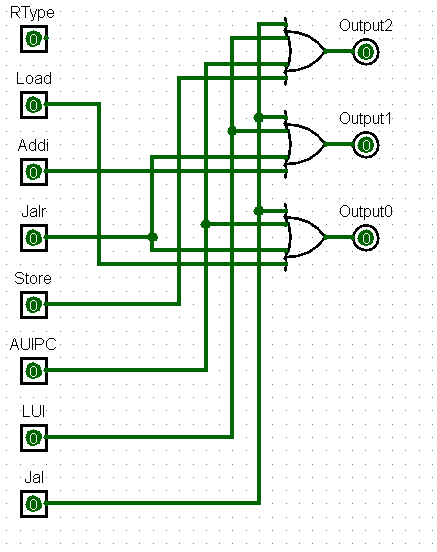


1. ARITHEMATIC AND LOGIC UNIT(ALU):

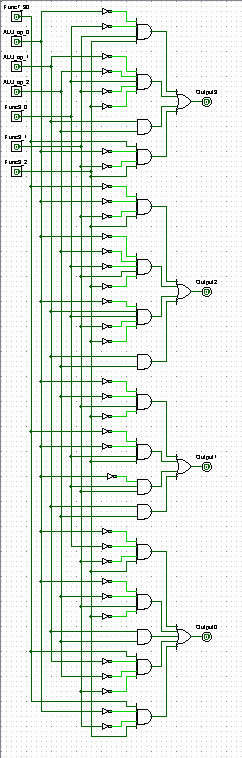
Next comes the Arithmetic and Logic Unit where all the arithmetic and logic operations are being conducted. It has several sub systems in it such as adders, subtracters, shifters, logical AND, OR, Not and XOR. Also, jal and jalr instructions which are to be by passed without performing any operation to them. The circuit looks like:



Also, there are ALU operation control and ALU control which are both made again using the combinational analysis option. ALU op has 8 input pins marked as all 8 types of instructions and 3 outputs.

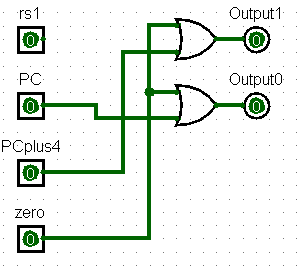


ALU control has 7 input pins which are 30th bit of function 7, all three outputs of ALU op and all 3 bits of function 3. It has 4 outputs as shown in the circuit below.



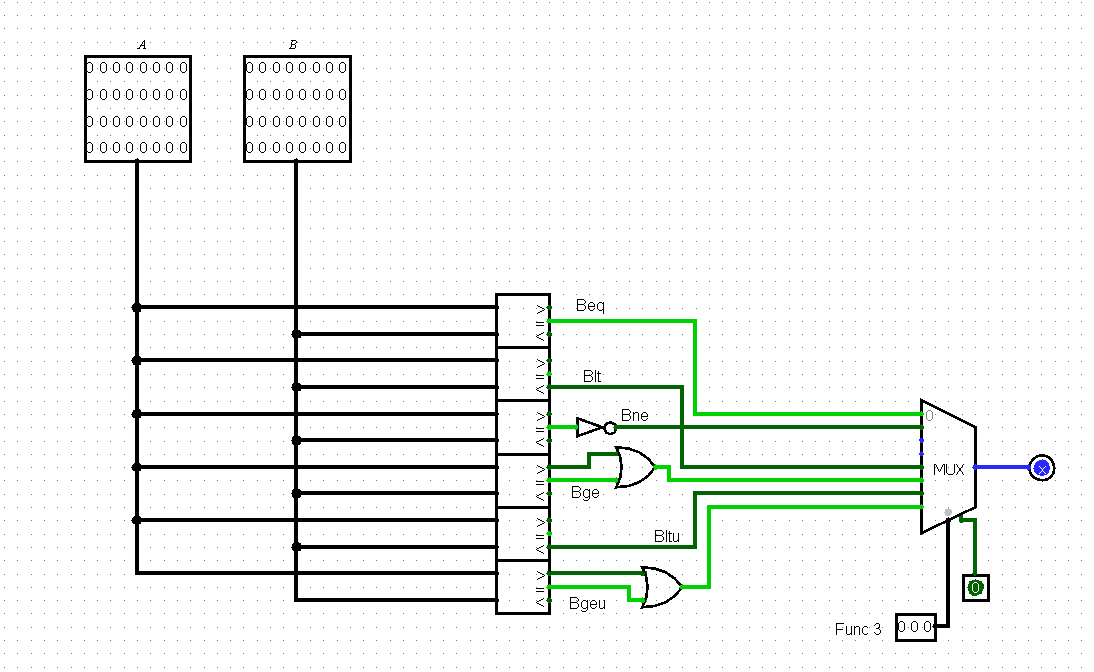
1. OPERAND A SELECT:

There can be four different choices which are rs1, source register 1, current PC, PC plus 4 which is the PC for the exactly next instruction and zero. It is designed to select only one of these in order to provide only single output in operand A.



1. BRANCH MODULE:

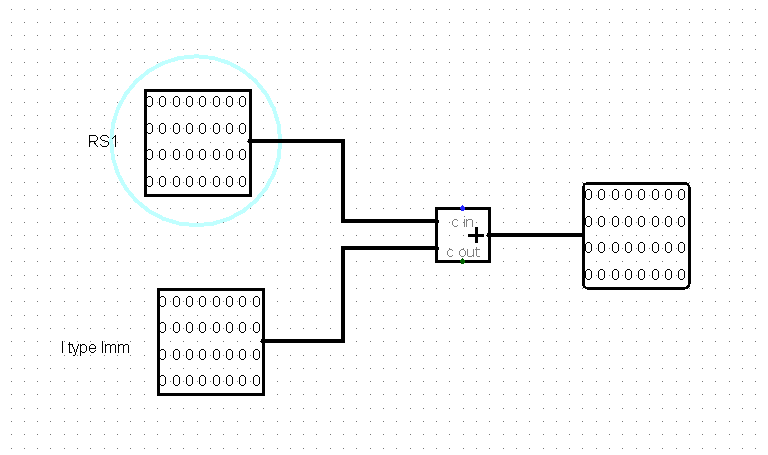
It is designed to perform branch instructions such as beq, bne, bge, bgeu, blt and bltu. It looks like:



Lastly,

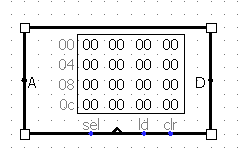
1. Jalr MODULE:

A module designed to perform Jalr instructions properly. Since jalr instruction just adds the value in source register 1 and immediate of I type therefore the circuits depicts the same thing in circuitry form.



1. RAM:

A data memory which executes store and load instructions only. Also called as data memory. It looks like:



1. ROM:

Each program instruction in the form of hex codes are loaded in ROM and are then executed through the entire circuitry. Also known as instruction memory. It looks like:



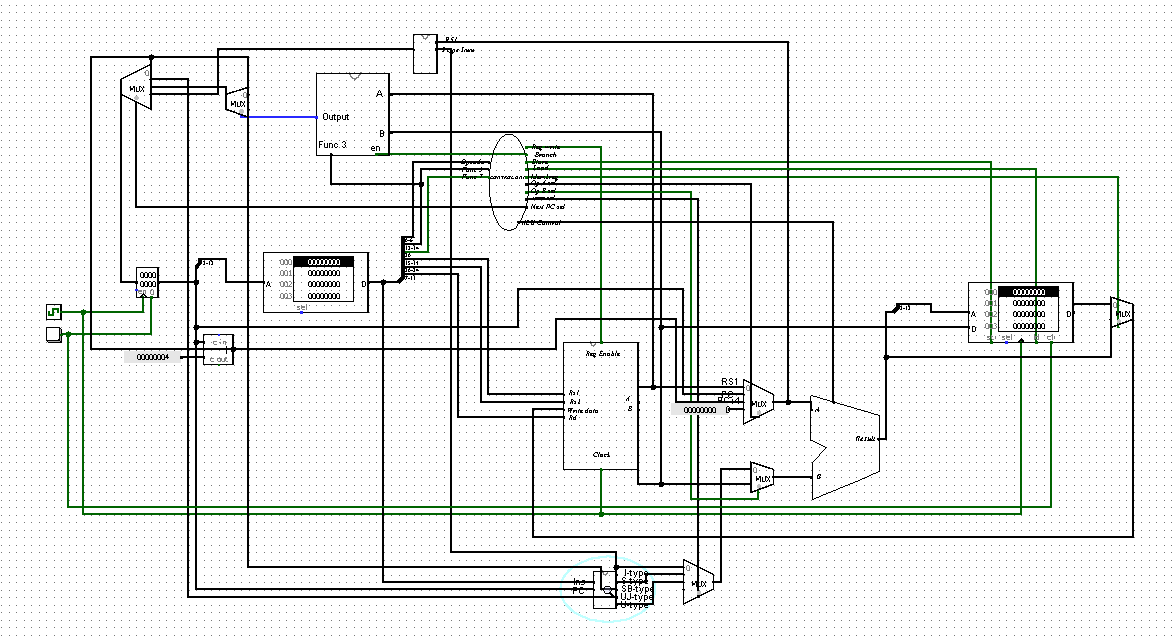
***METHODOLOGY:***

Each instruction is being decoded and then executed in the processor using all the sub systems. The circuit is implemented on a Logic Simulating software called as ***LOGISIM***. All the different circuits/components are firstly developed and are then connected to design the top of the microprocessor. The circuit is required to be tested in order to see if it works properly or not, few codes are generated, may be simpler ones having two to three lines of instructions or may be a long code consisting of more than 9 to 10 lines of codes on online assembly language simulator called ***VENUS***. The hex codes of these instructions are copied using the dump option on Venus. The same hex code is copied on a text file with header as v2.0 raw and saved in the format of abc.mem. The same file is loaded into the ***ROM*** by right clicking the ROM and using the load image option in there. It lets us load the hex codes of the instructions and allows us to see how the instruction is performing in the circuit.

***IMPLEMENTATION:***

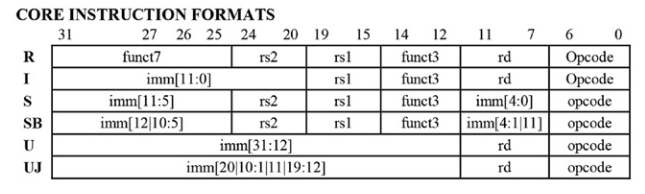
The circuit is implemented on logic simulator named as ***LOGISIM*** where each instruction is executed in the circuit step by step.

The final circuit looks like:



***RESULT:***

The results that can be concluded using the entire circuitry is that how each instruction behaves as different and how they’re executed. The type of instructions are as follows:



As the table suggests, each instruction set has different immediate bits, different destination and source registers, different function 3 and function 7 bits.

* ***VERIFICATION:***

The circuit is verified as the loaded instructions are properly executing through the circuit. The hardware implementation is yet to be done rest all is working properly.

* ***TEST* *PROGRAMS*:**

**#1**

***Assembly Code:***

addi x4,x4,4

jump:

addi sp, sp, -4

sw x4,0x0(sp)

addi x5,x5,1

lw x6,0x0(sp)

addi sp,sp,4

jal jump

***Hex Code:***

00420213

ffc10113

00412023

00128293

00012303

00410113

fedff0ef

**#2**

***Assembly Code:***

addi t1, x0, 10 #initialize

beq t0 , t1, end #if

addi t2, t2, 1 #counter(increament by 1)

addi tp, tp, 11 #multiplying by 11(adding 11 everytime)

jalr ra, x0, 4 #jump to PC 4

end:

***Hex Code:***

00a00313

00628863

00138393

00b20213

004000e7

**#3**

***Assembly Code:***

add x2, x3, x4

xor x3, x4, x3

sll x4, x3, x4

slt x5, x7, x8

or x6 x7, x8

***Hex Code:***

00418133

003241b3

00419233

0083a2b3

0083e333

***CONCLUSION*:**

This is a RISC V single cycle core implemented on a logic simulator ***LOGISIM,*** developed on RV32I instruction set.It is based on the synchronous phenomena since on each rising edge, only singe instruction is being executed. The circuit consists of 32-bit ALU, 32-bit data bus, 16KB RAM/ROM, and 12-bit address bus for RAM. It has register file which consists of 32 registers each having data width of 32 bits.

***REFERENCES:***

* Computer Organization and design By David A. Peterson, John H. Hennesy(RISC-V edition).
* RISC-V instruction set manual(Volume I).