Project Part 03(Report)

CSE332

SECTION: 9

Submitted to:

Syed Mahmud Husain (SMH2)

Department of Electrical & Computer Engineering

Submitted by:

Muhammad Zawad Mahmud

ID: 1931401642

Data Table:

N=abs(my serial-32)=abs(30-32)=2
as N<10. So, N=2+10=12

R -Format:

opcode	rd	rs	rt	func
4bit	2bit	2bit	2bit	2bit

I-Format:

opcode	rd	rt	immediate value
4bit	2bit	2bit	4bit

Instructions:

Binary	Hex
001010000000	280
000100100001	121
000100100010	122
0110 0000 1100	60c
0000 1000 1010	08a
1001 0000 1001	909
0110 0100 0011	643
0101 0001 0010	512

First instruction is 1c0. This is for ~~to~~ load. It gets value from 0 no memory adress and store the value at $t_0 register. The next instruction is 0f7 which will write 0 value at $t_1 register and 0f8 will write 0 at $t_2 register. upcoming instruction is 34A which will just compare $t_0 and $t_1 register and jump at ~~$t_{10}~~ 10th instruction. Next instruction is 68 which is sum = sum + i. I need to repeat this twice for sum = sum + 2i. Next instruction is 4d4. It will add value from $t_1 and 4 number memory. The next instruction is 316. This instruction compare $t_1 register value i and $t_2 register value. If it's not equal then return to 6 no instruction. If it is equal, the loop will break and it will move to next instruction. It is 2e8. It will store the register's value which is sum into the memory's 8 no memory.

Process of running:

Store value for X at 0 no memory and 1 at 4 no memory. Tick the clock unless the instructions are over. At last check the result at 8no memory.

Result of simulation:
For Load(lw) Execution:

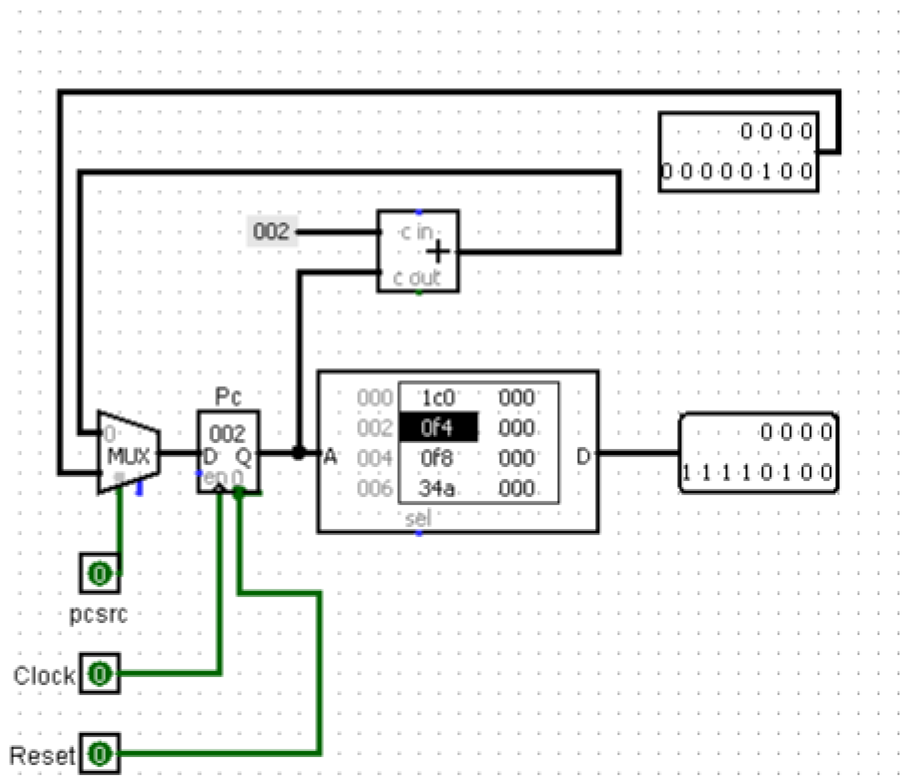

Figure-01: Instruction Fetch

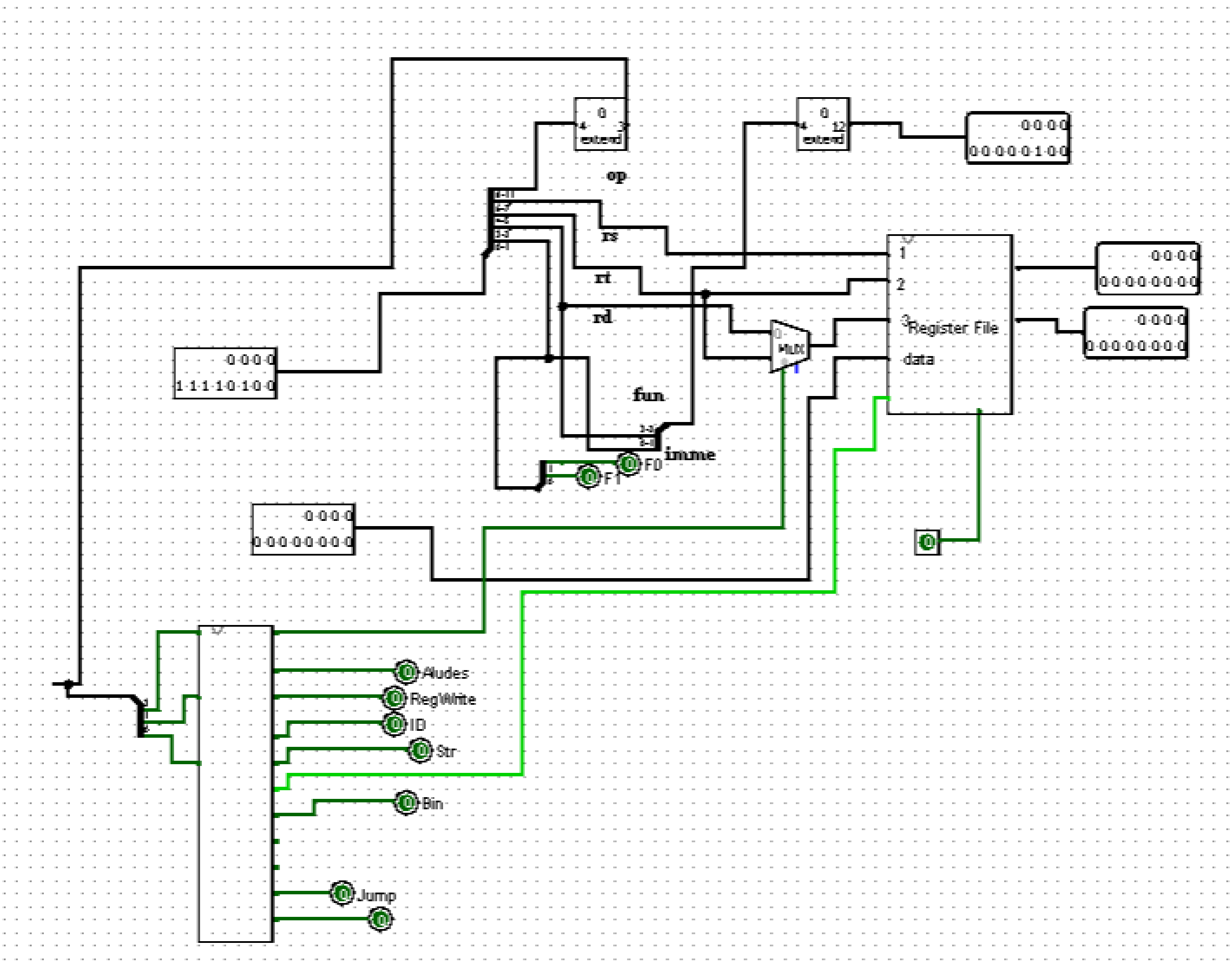

Figure-02: Instruction Decode

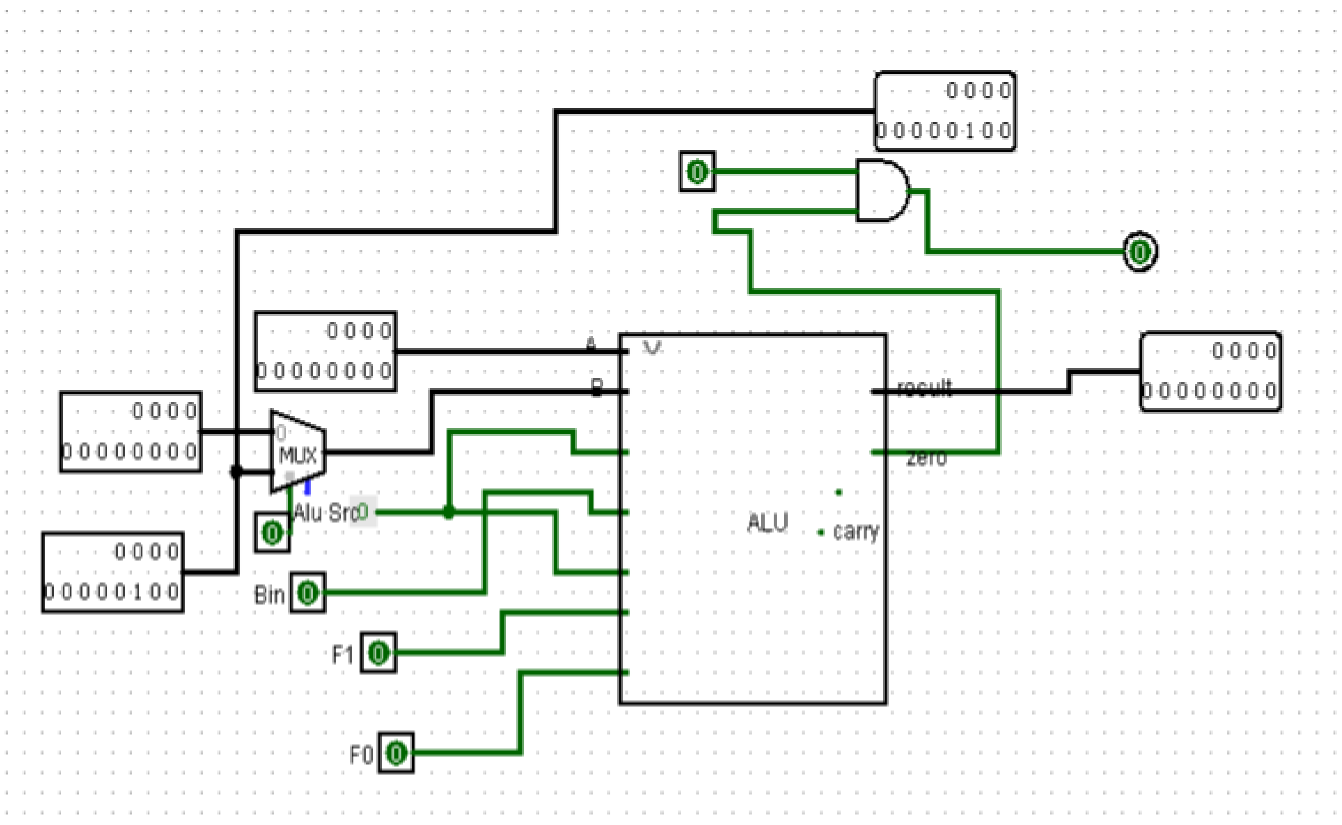

Figure-03: Execute

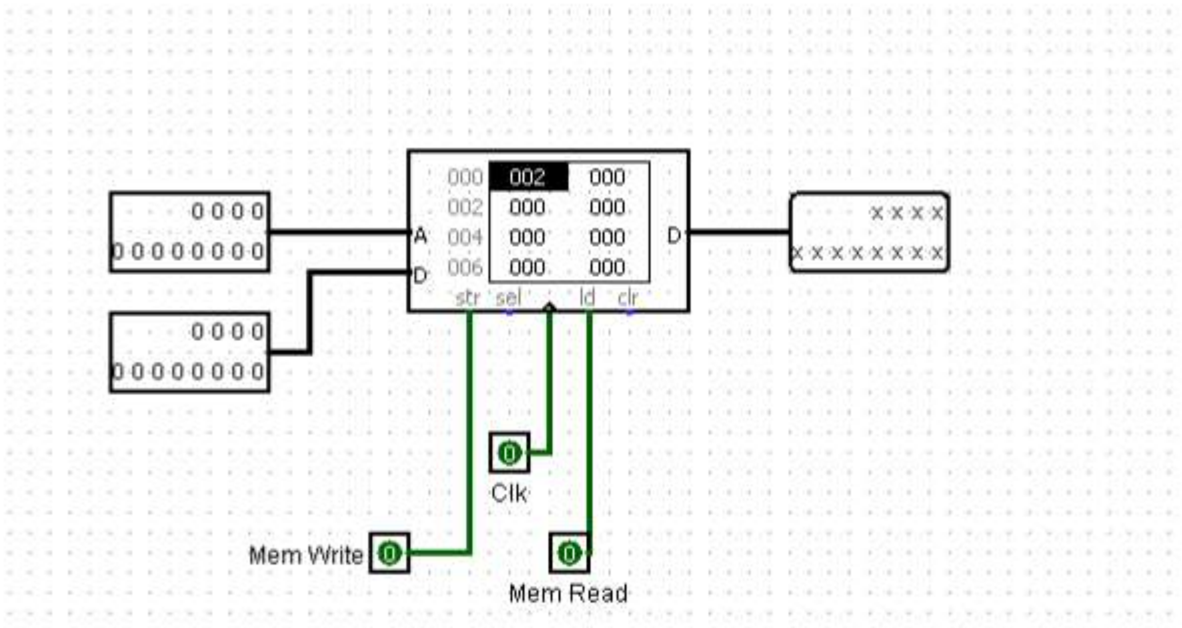

Figure-04: Memory excess

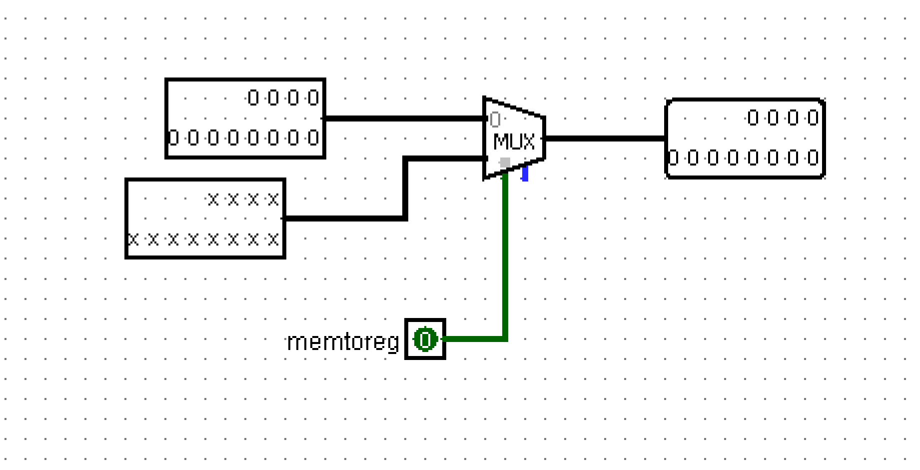

Figure-05: Write Back

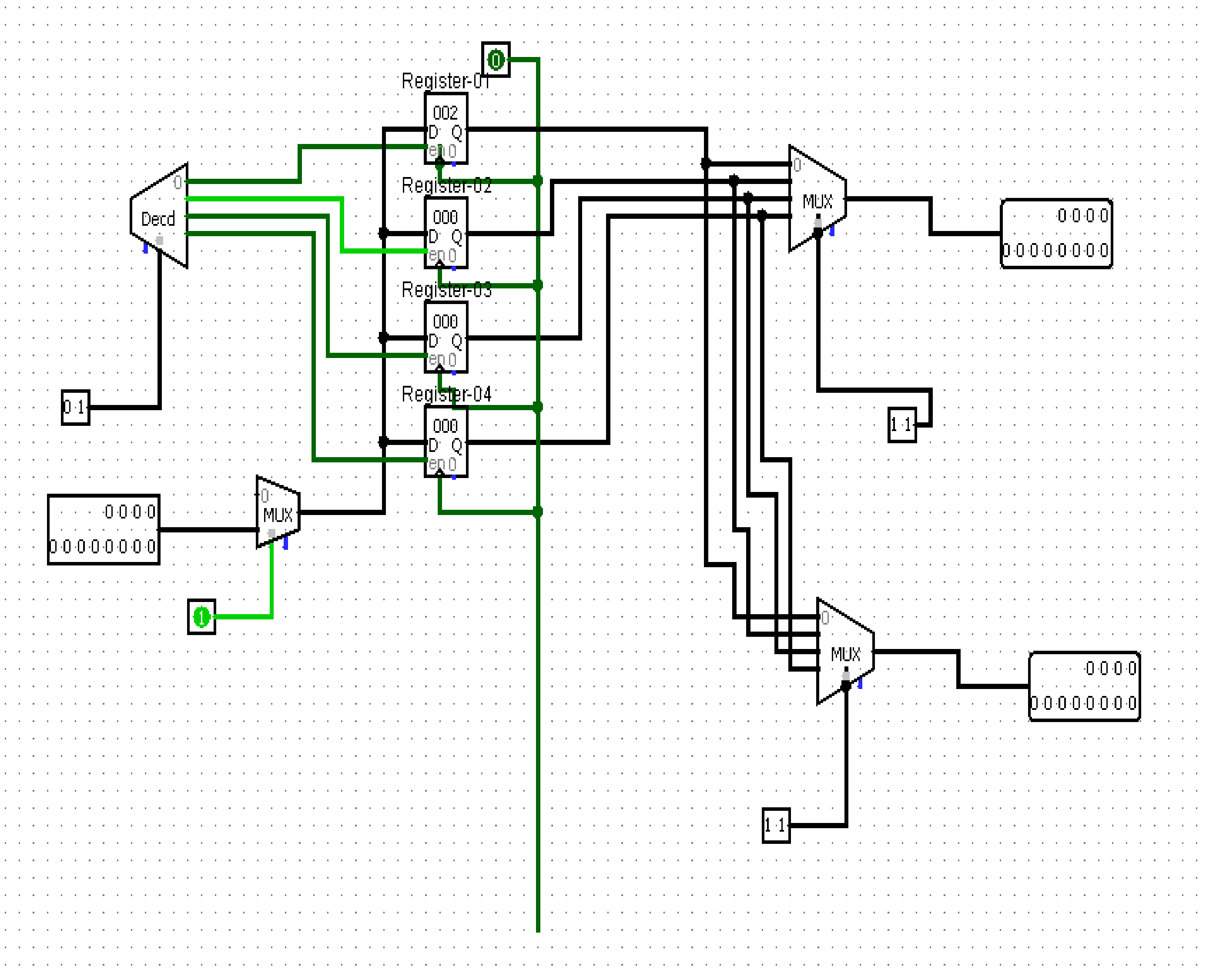

Figure-06: Register File

Result of simulation:

For Store(sw) Execution:

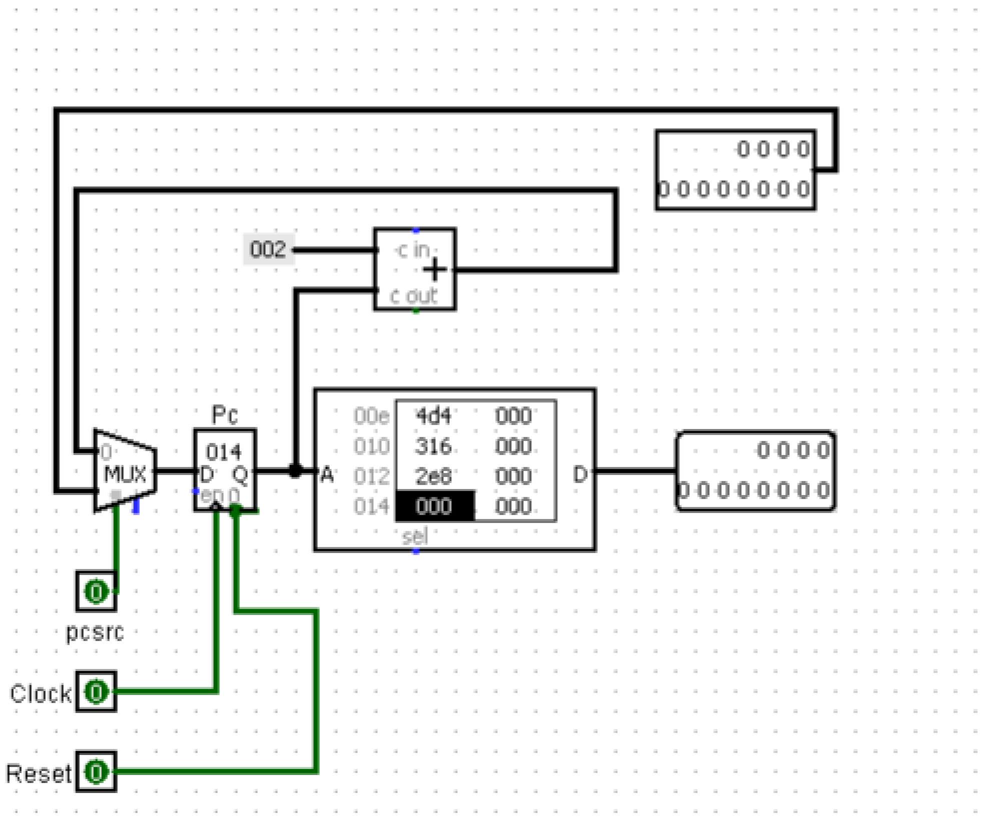

Figure-01: Instruction Fetch

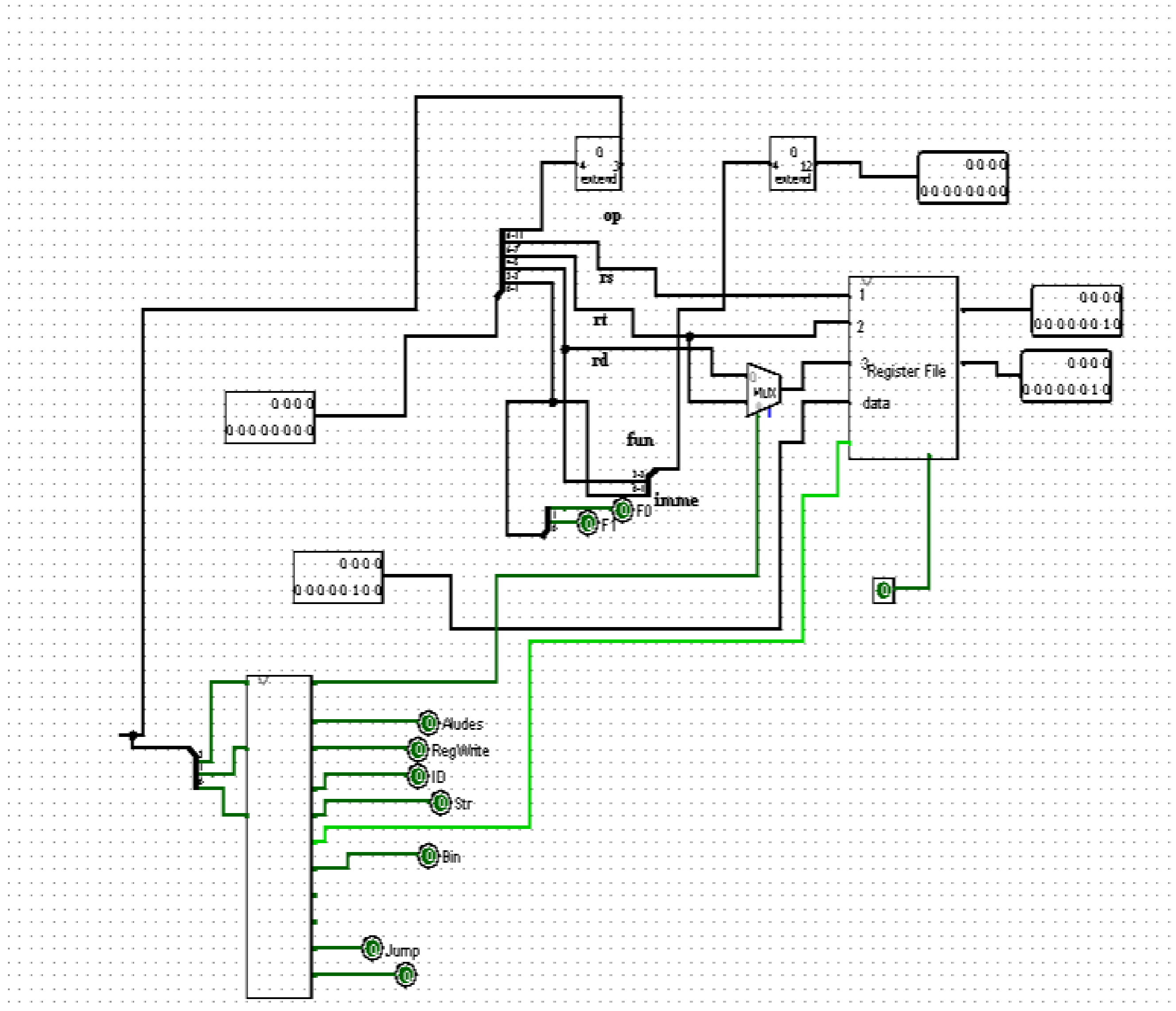

Figure-02: Instruction Decode

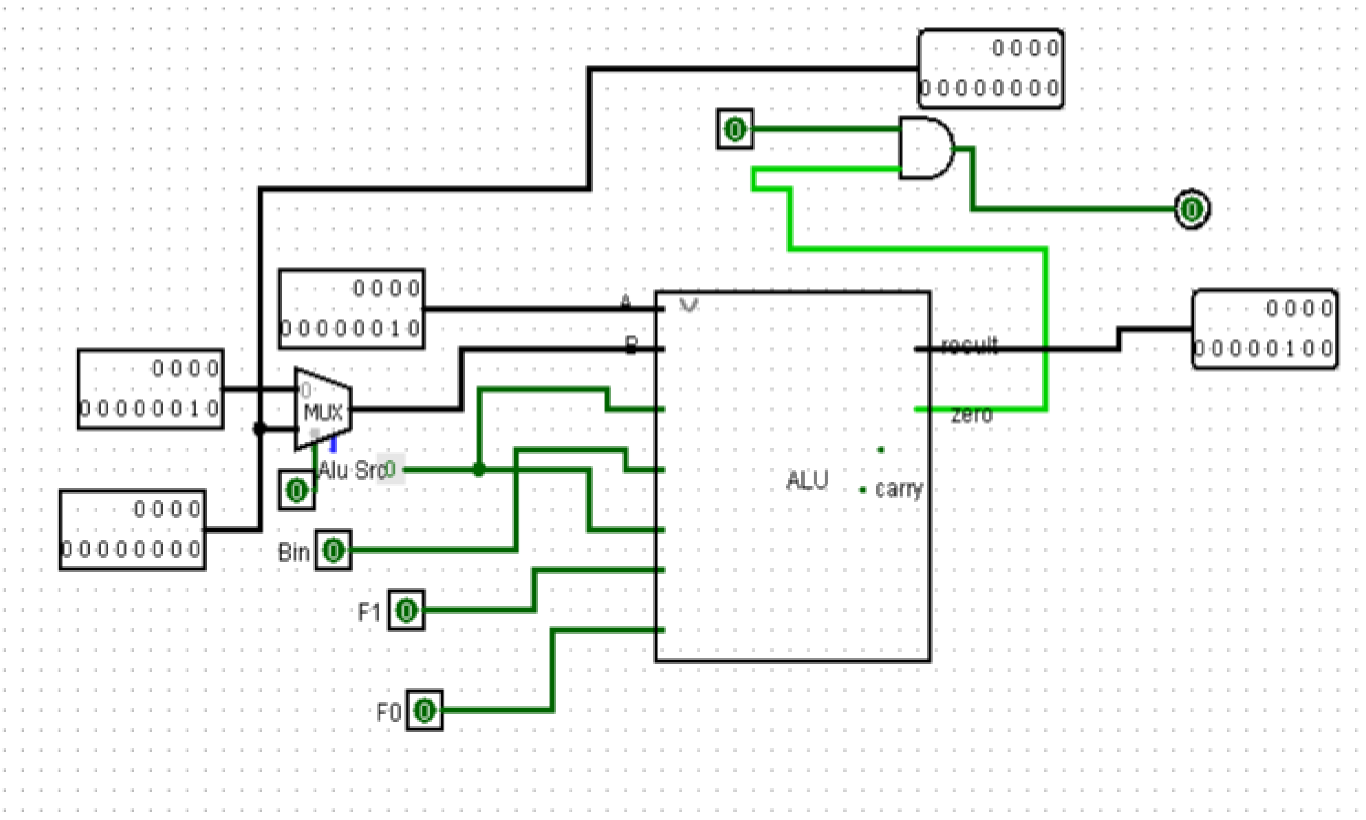

Figure-03: Execute

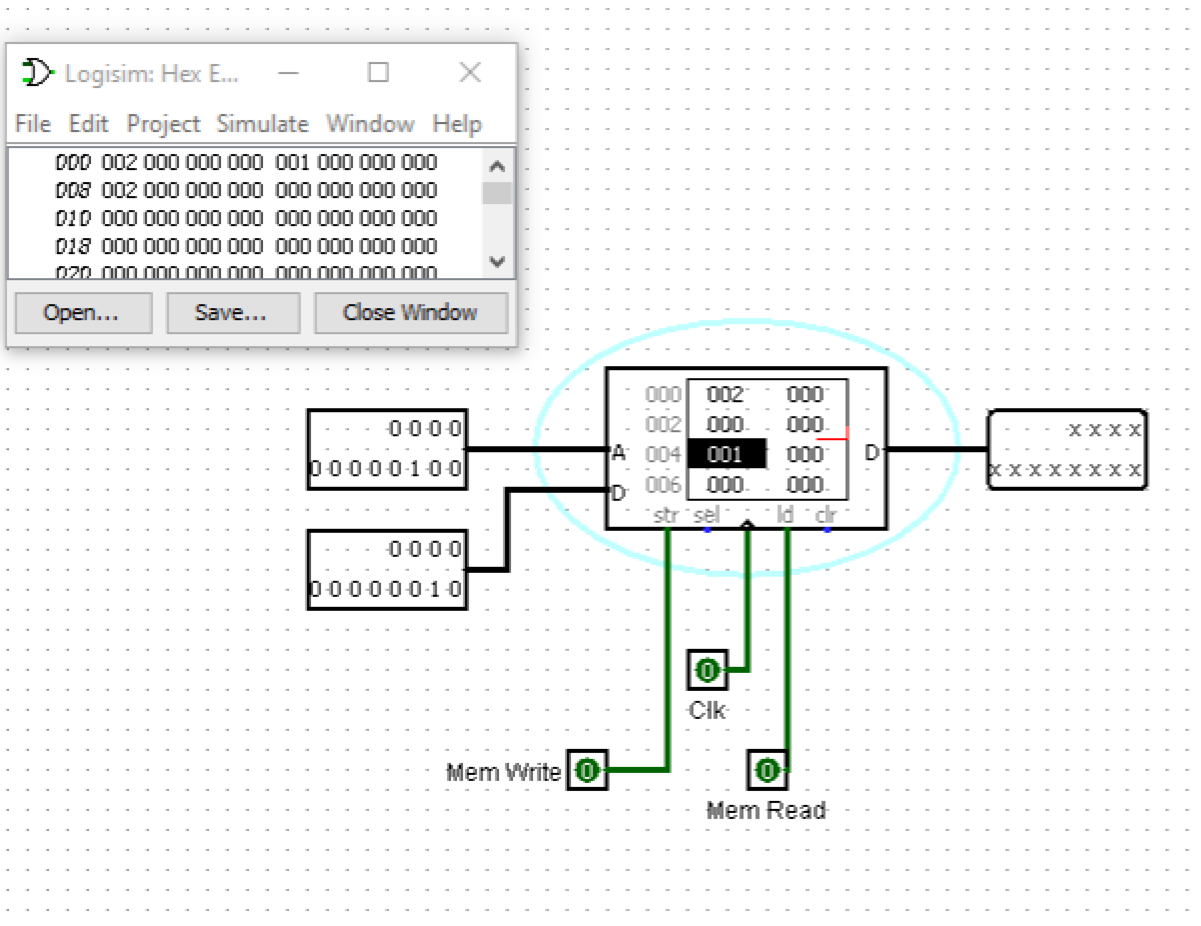

Figure-04: Memory excess

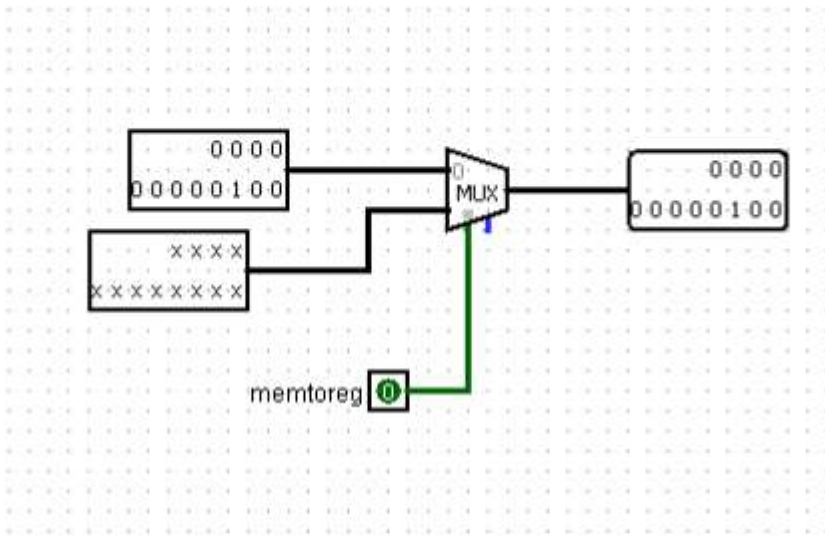

Figure-05: Write Back

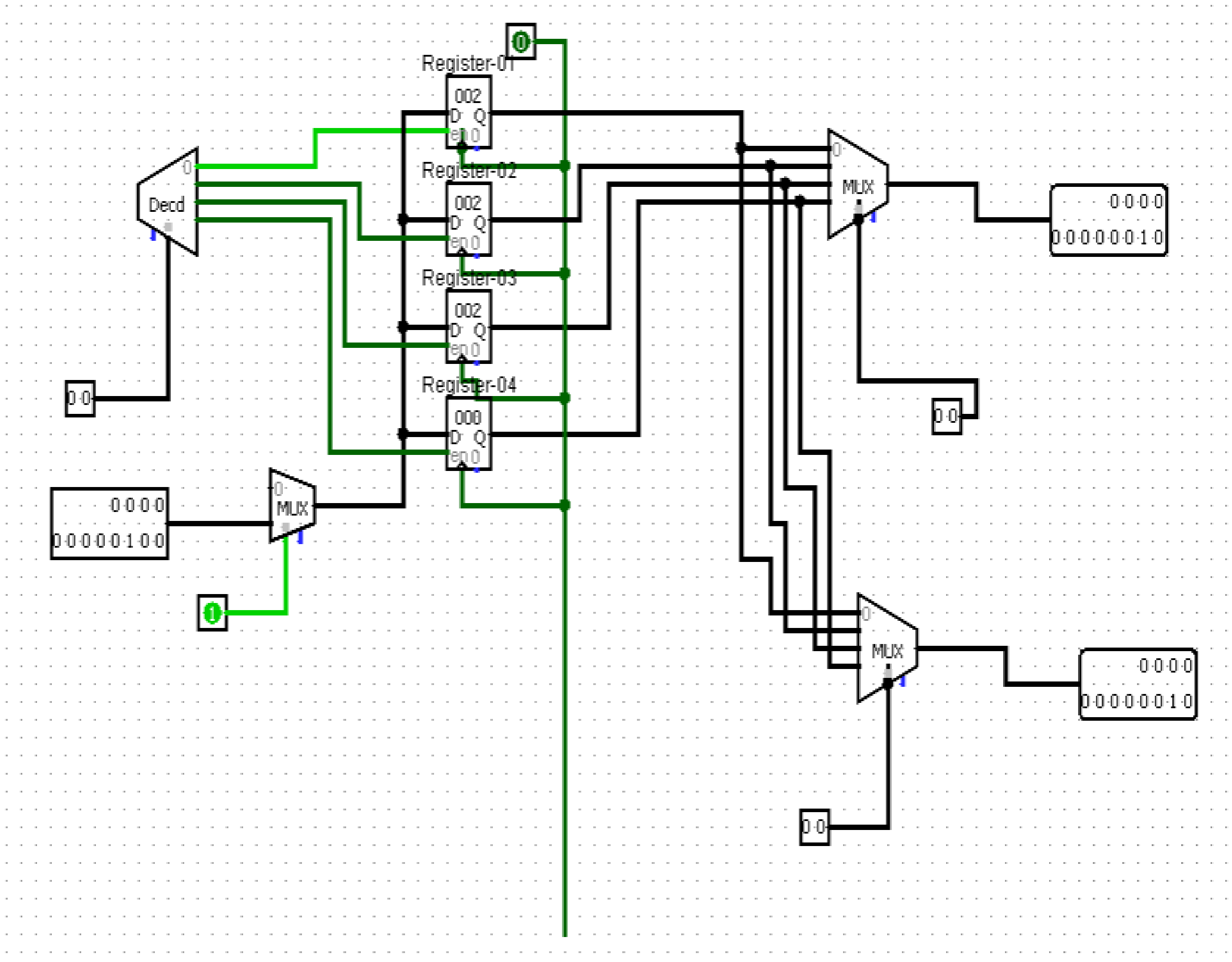

Figure-06: Register File

Result of simulation:

For Jump(beq) Execution:

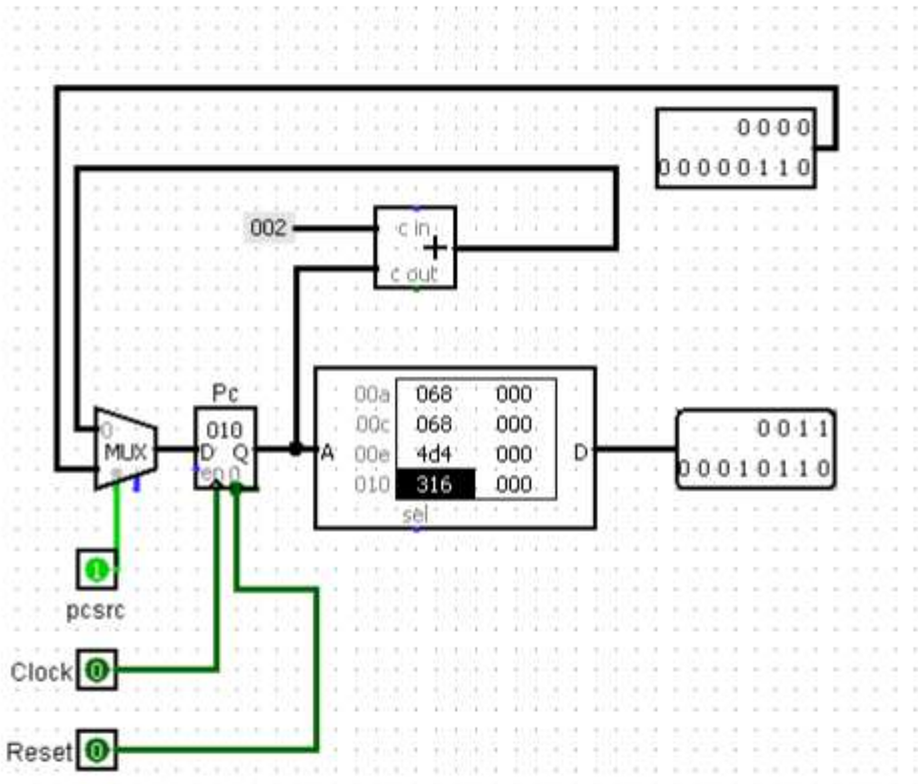

Figure-01: Instruction Fetch

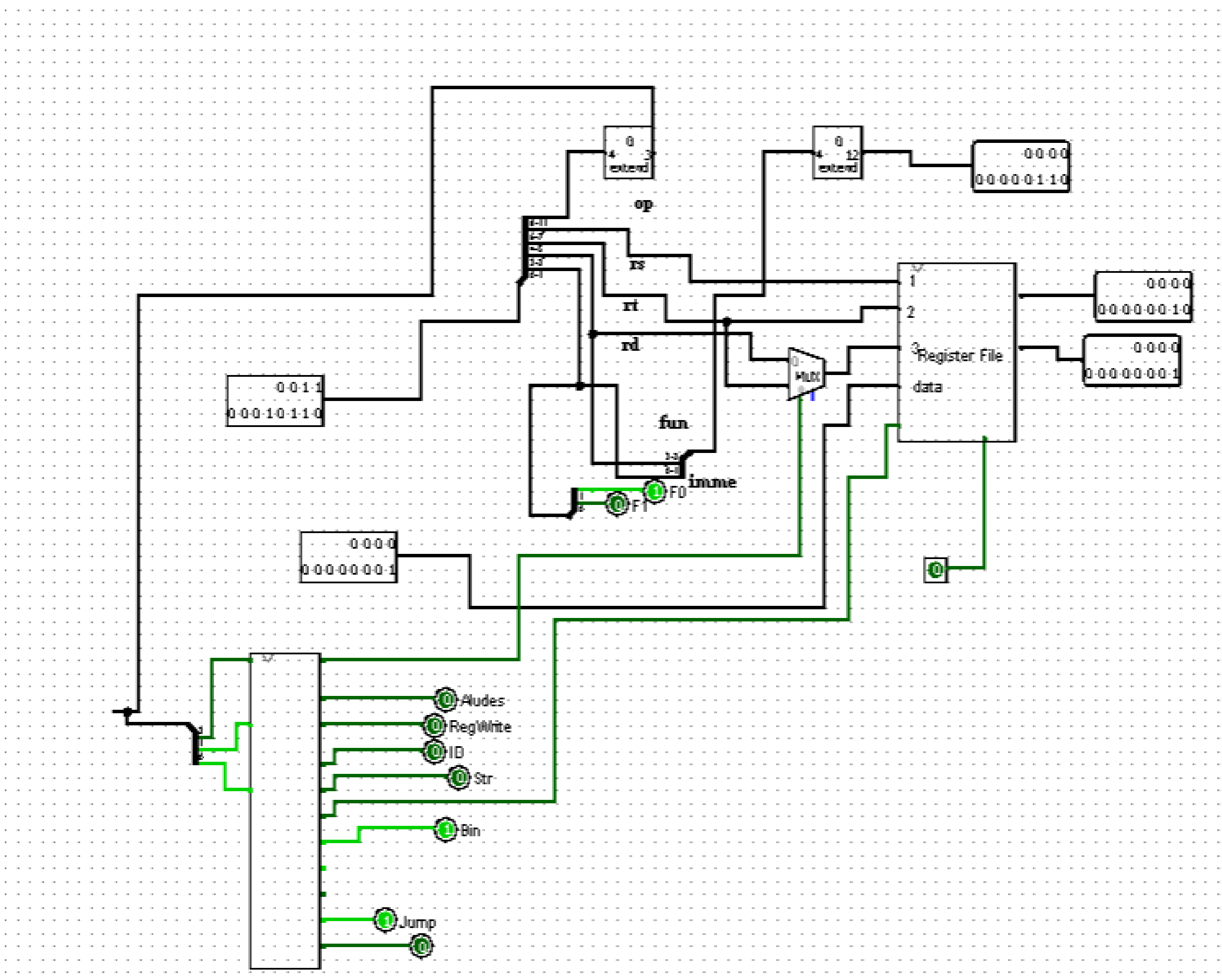

Figure-02: Instruction Decode

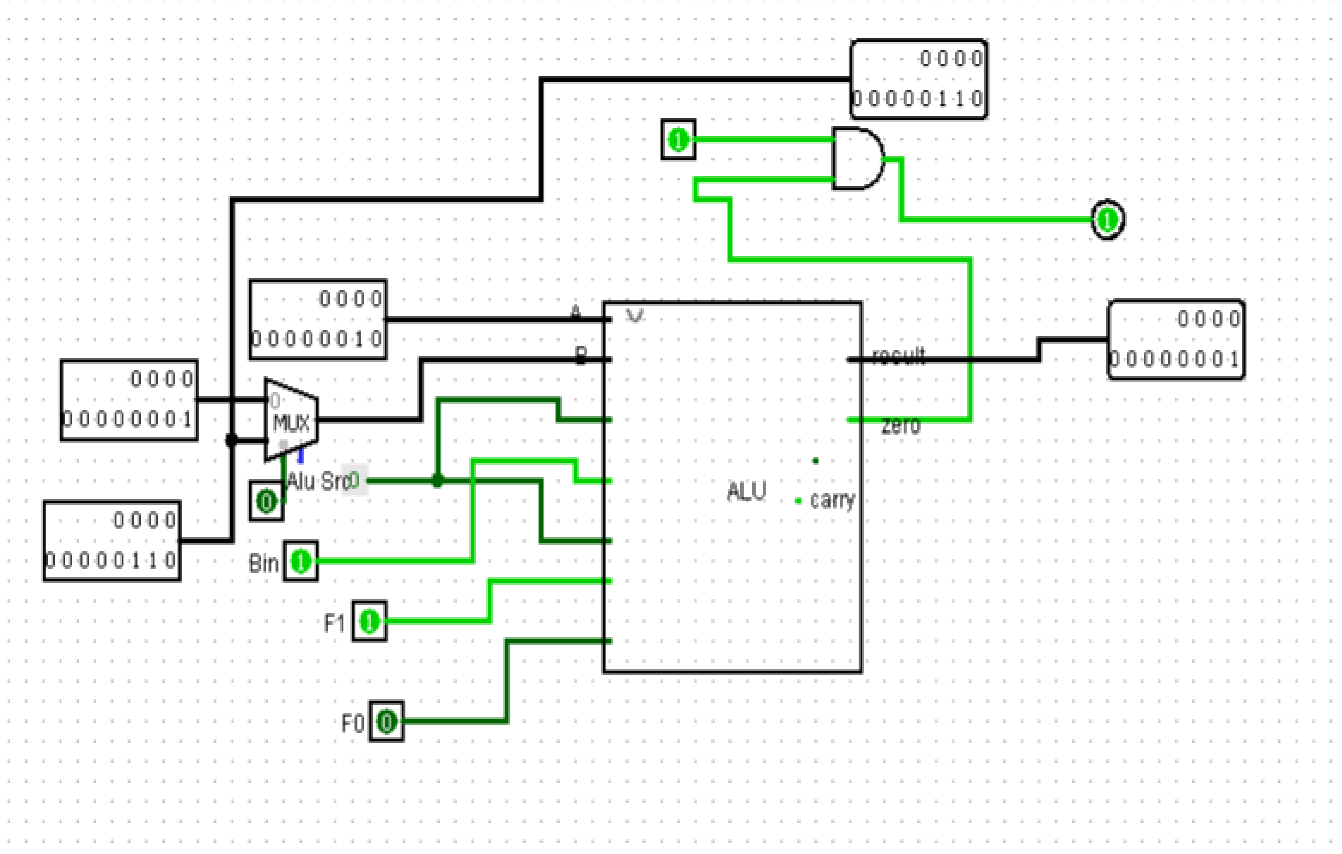

Figure-03: Execute

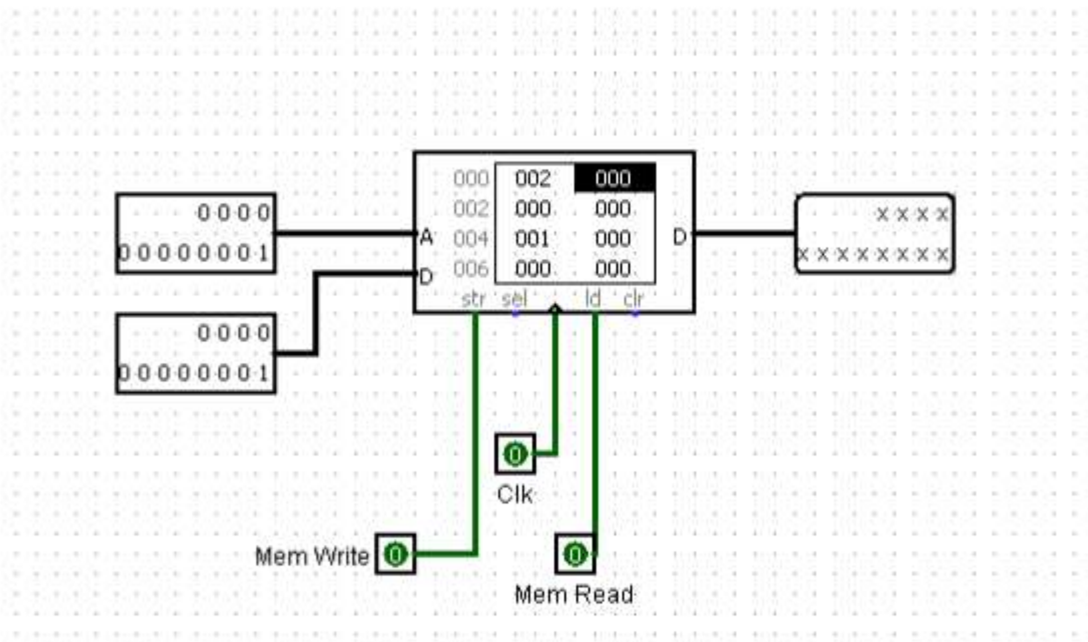

Figure-04: Memory

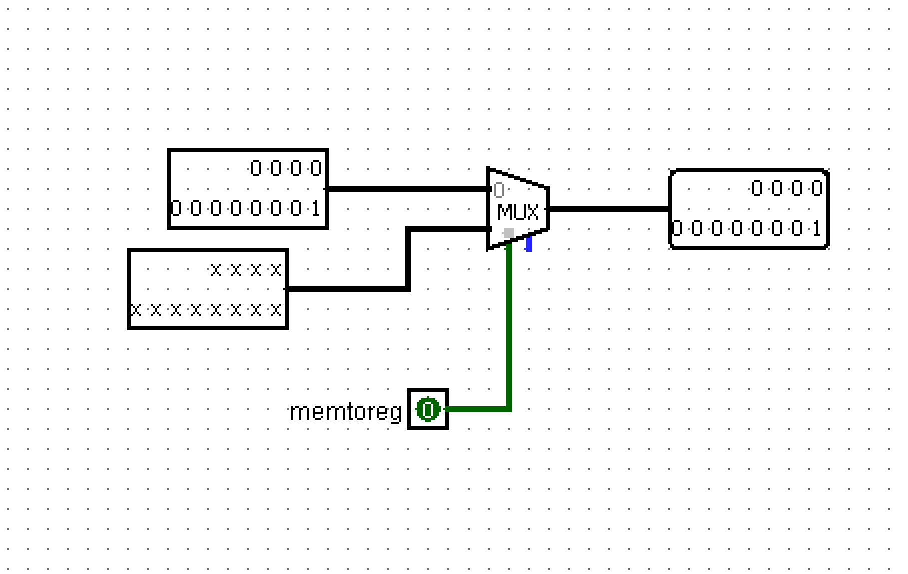

Figure-05: Write Back

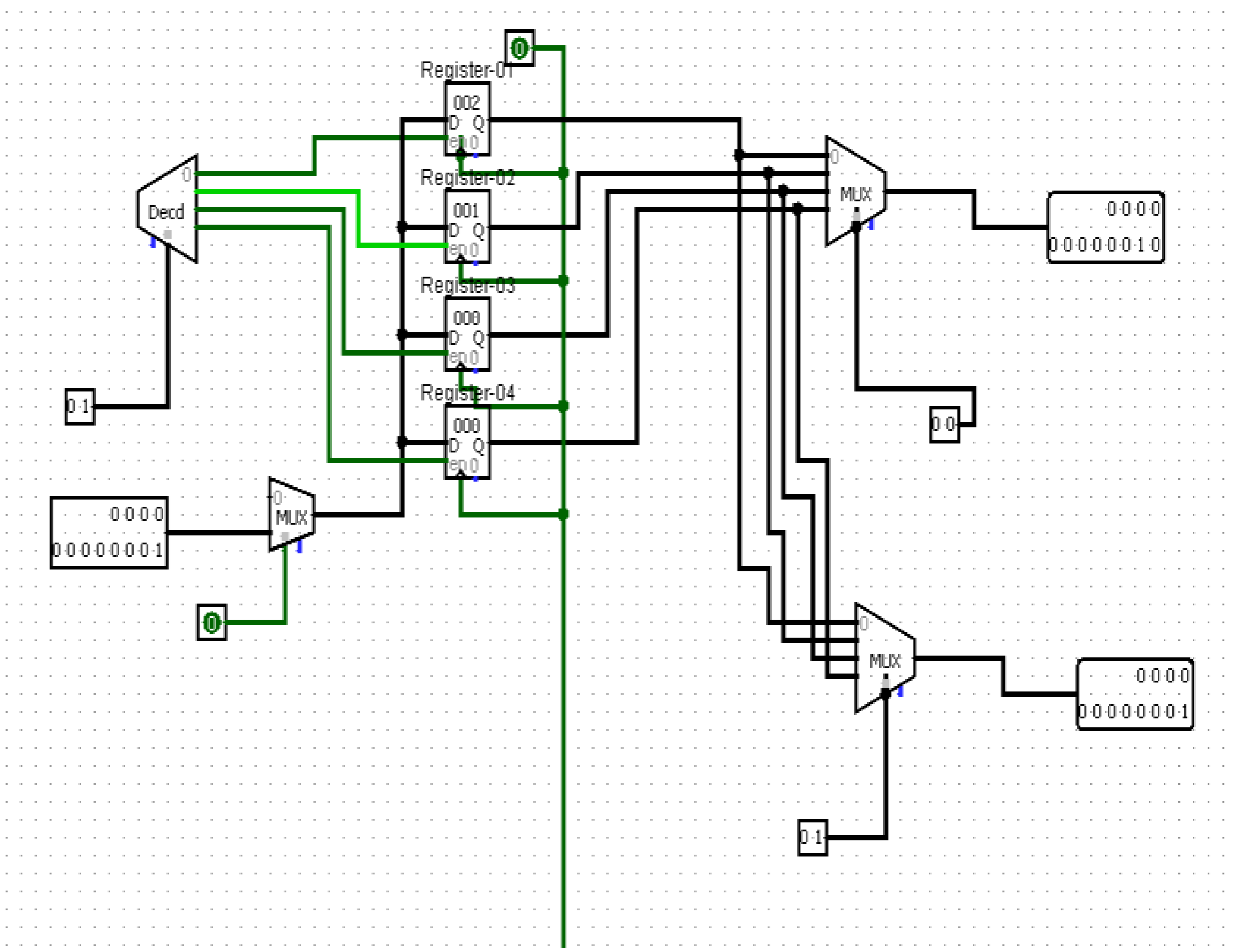

Figure-06: Register File

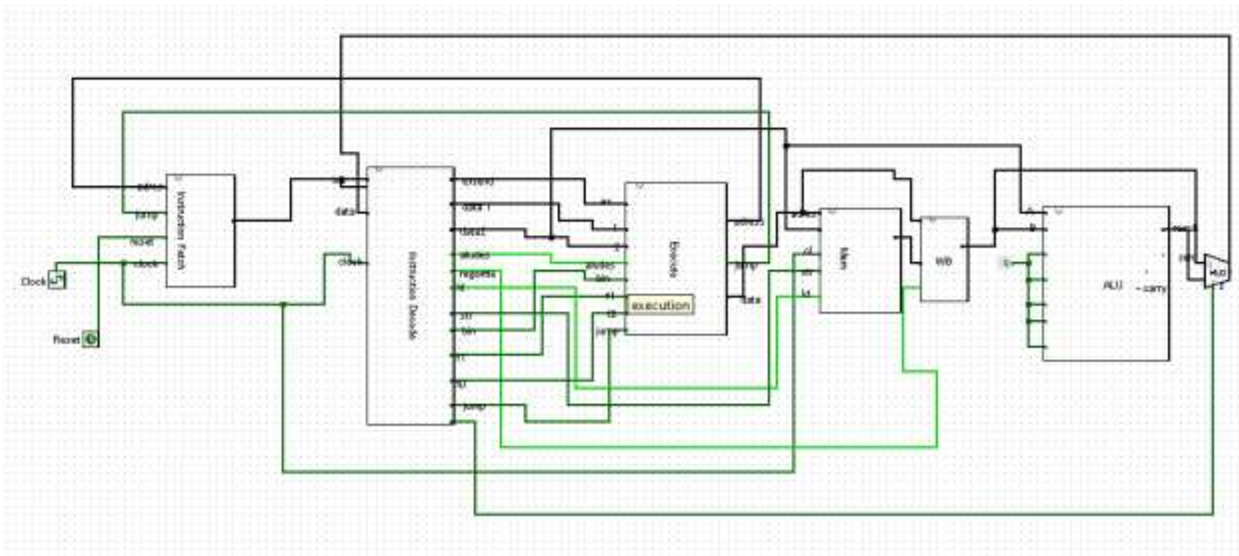

Figure-07: Full Data-path

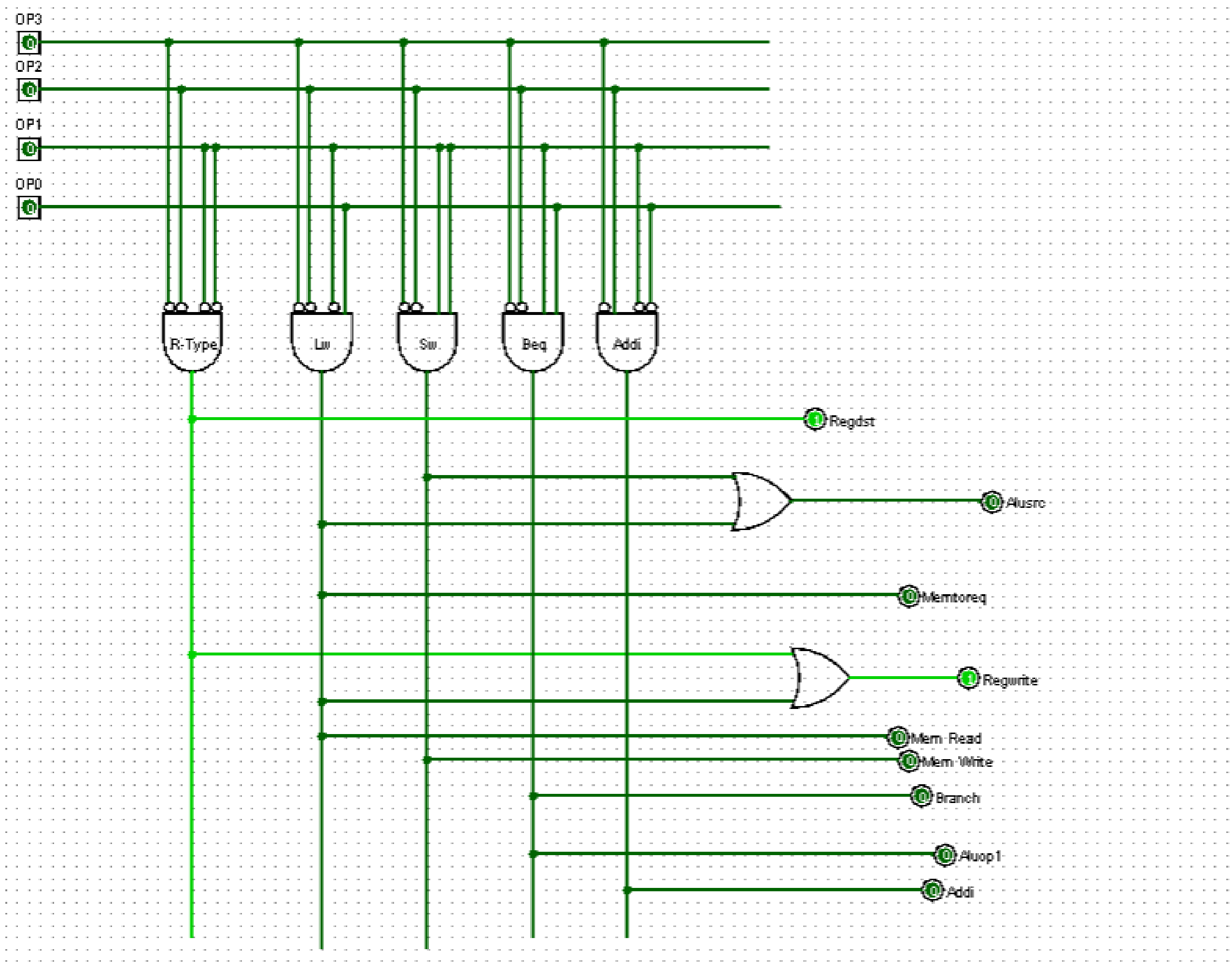

Figure-08: Main Control Unit

Discussion:

For question-03 of my midterm assignment, the objective was to design a full data-path of single cycle implementation of MIPS.

For hardware specification first I have to try to divide my bit in ISA format which is,
$N=2^n$
$12=2^n$
$n=\log12/\log2=3.58 \approx 4$
I need to give rs=rd=rt=4 which I can't give then I will be left with 0 bits but I'll need 2 bit each for op and function. So, I need to go for customized way.
For functions, I have to give 2 bits as I need to perform 4 operations (load, store, add, sub). After giving 2 bit at func, I left with 10 bits. I can give 2 bits each for rs, rd & rt so I have given 2 bits for those and 4 bit to opcode.

Now in instruction fetch, for this implementation I needed program counter (PC). I had to give a mux at the left side of pc to identify jump. I have taken 12-bit rising edge register as PC. Then I needed an instruction memory. I have used rom as my instruction memory. I also needed a 12-bit adder. Firstly, I took a 12-bit register and an adder. I put register's output and 001 as adder's input as I'll need to add 1 while clock ticks. I connected a clock with my register and the input is adder's output. The clock will help to go to the next stage. Whenever the clock will tick, register will go to next stage which is the current stage+1. I also have a clear pin in my register which will take it to 0 stage asynchronously. I gave the register's output as my rom's address bit. The address bit and data bit of my rom both are 12 bits as I need to design a data-path for 12-bit CPU. I used a constant 1 as rom's selection pin as when the selection pin is 0, the rom is disabled. To execute the loop all required instructions, I converted those to hexa from binary and stored it into rom.
In instruction decode, for this implementation I needed register file which will store data. I have designed $2^3=8$ bit register file for it. In my register file I have used a 2X1 Mux which's selection pin is regwrite to determine if the data will be written or not. My register file will read two 12-bit data and there is a clock connected with each register. After the clock tickels, the write port input will be saved into the addressed register. Here I have a 2X1 mux at the left side of instruction decode. The selection pin of the mux is regdes and it will identify the

instruction(R-type/I-Type).
In execute, I have to use my alu as I had to do add and sub both. For loop's instructions, I had to do add and for beq I had to do sub as if the sub will be 0, the zero flag will be 1 and the brach/jump from the main control circuit and the zero flag I had to use an and gate & I had to put the and gate's output as the selection pin of the mux at the left side of instruction fetch to identify jump.
In memory excess, I have used a ram which has 12-bit address bit and 12-bit data-bit. For load, memory read pin will be 1 and for store memory write pin will be 1. Here in the ram's memory address, I have to load the data.
In write back, I have only used a mux to identify if the data will be written to the register file or not. The mux's selection pin is memtoreg which I did in main control circuit.
I have to design a control circuit to identify which pin's of muxs will be on/off for a particular instruction. I have designed it using PLA.
Finally, I combined all my 5 circuits into one circuit to implement the single cycle operation.