Fayoum University
Faculty of Engineering
Department of Electrical Engineering
2nd year , Computer Engineering

COMPUTER ARCHITECTURE PROJECT
PHASE 1
SINGLE CYCLE PROCESSOR

Under Super Vision of
Dr/Gihan Naguib
Eng/Mohamed Al Soadany

TEAM

Members :.

- **Demyana Email**
- **Sara Mohey Al Din**

Each Member Mission :.

- Demyana Email :

  Design the final datapath

  Design and implement the following subcircuits

  (I-memory ,D-memory , registers)

  Basic ALU design

- Sara Mohey Al Din :

  Implement final datapath

  Design and implement control unit

  ALU implementation

  Documentation

- Both :

  Simulation

DESIGN AND IMPLEMENTATION

Overall Datapath :.

Main Components

- **PC : The program counter register determine which instruction in the instruction memory will be implemented each clock.**

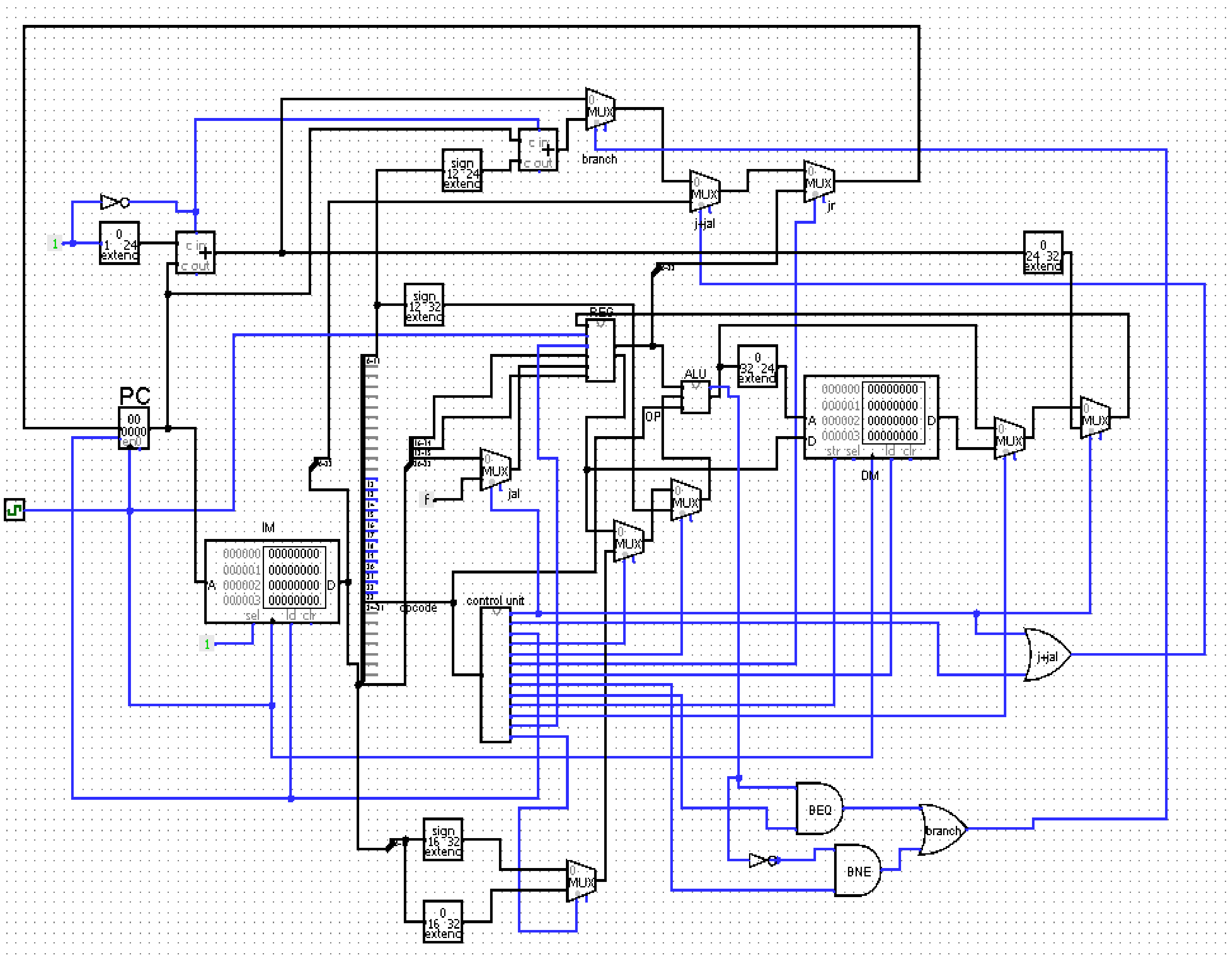
- **IM : The instruction memory "RAM" contains each instruction 32bit code .**

- **Register File**

- **Data Memory**

- **ALU : It performs all the arithmetic and logical operations needed in R-type ,I-type , load and store instructions**

- **Control Unit : It generates all the selection enable and control signals needed**


PC
IM
REG
ALU
OP
DM
control unit
opcode
branch
j+jal
jr
jal
BEQ
BNE
MUX
sign extend
extend

Each Component Implementation

PC :

It is a 24bit register with enable signal of 1-bit called “PC_signal” generated by the control unit .

The output is connected to address port in the instruction memory and to the next pc calculation part.

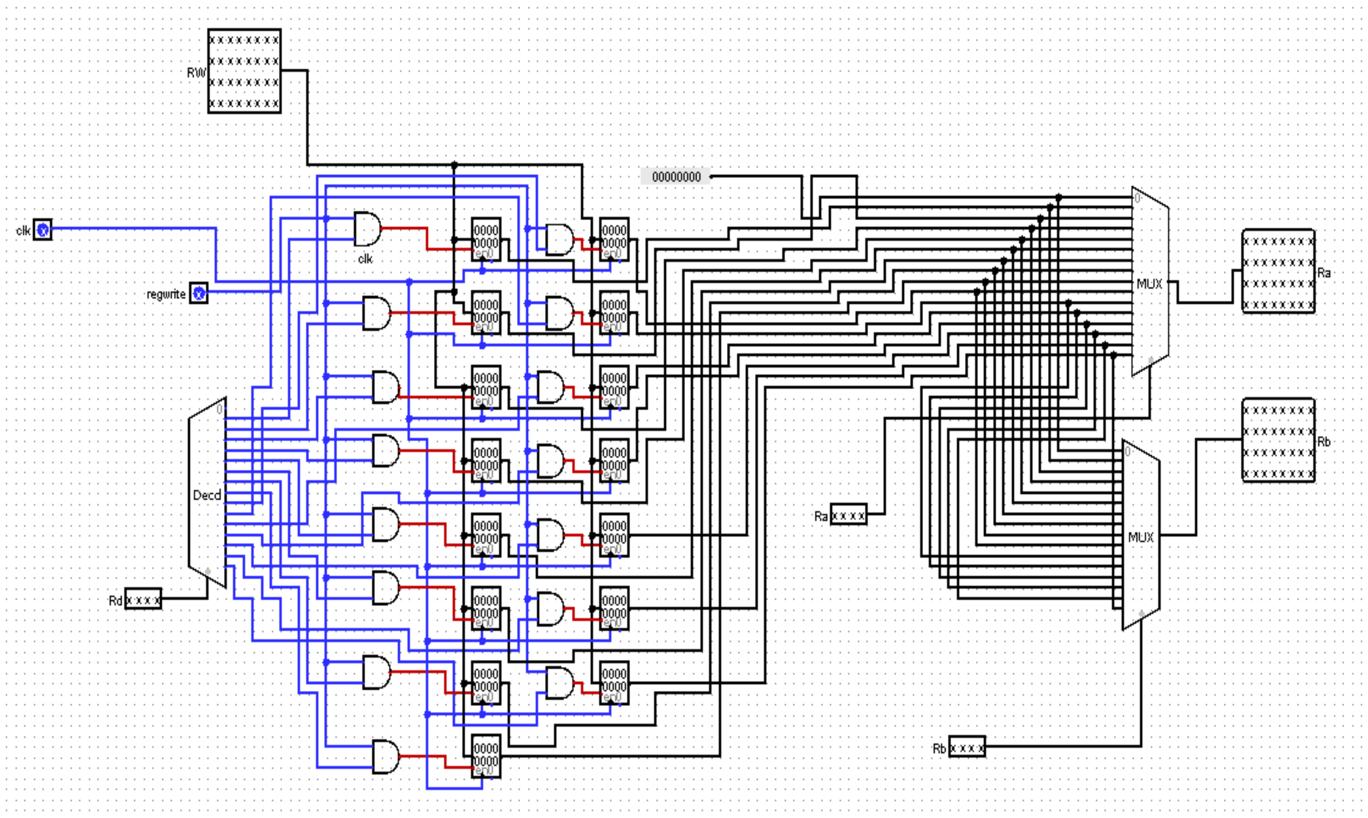
The next pc is connected to the register input part where at the clock edge it gets it’s new value to be used in the next instruction.
It takes one of those four values:

- the previous current value incremented by 1 if all branch and jump signals have a value of zero
- the value of register Ra if jr instruction is detected by IM
- the value of a 24bit immediate if a j or jal is detected
- the value of the current pc value added to a 12bit immediate in case of branch

IM :

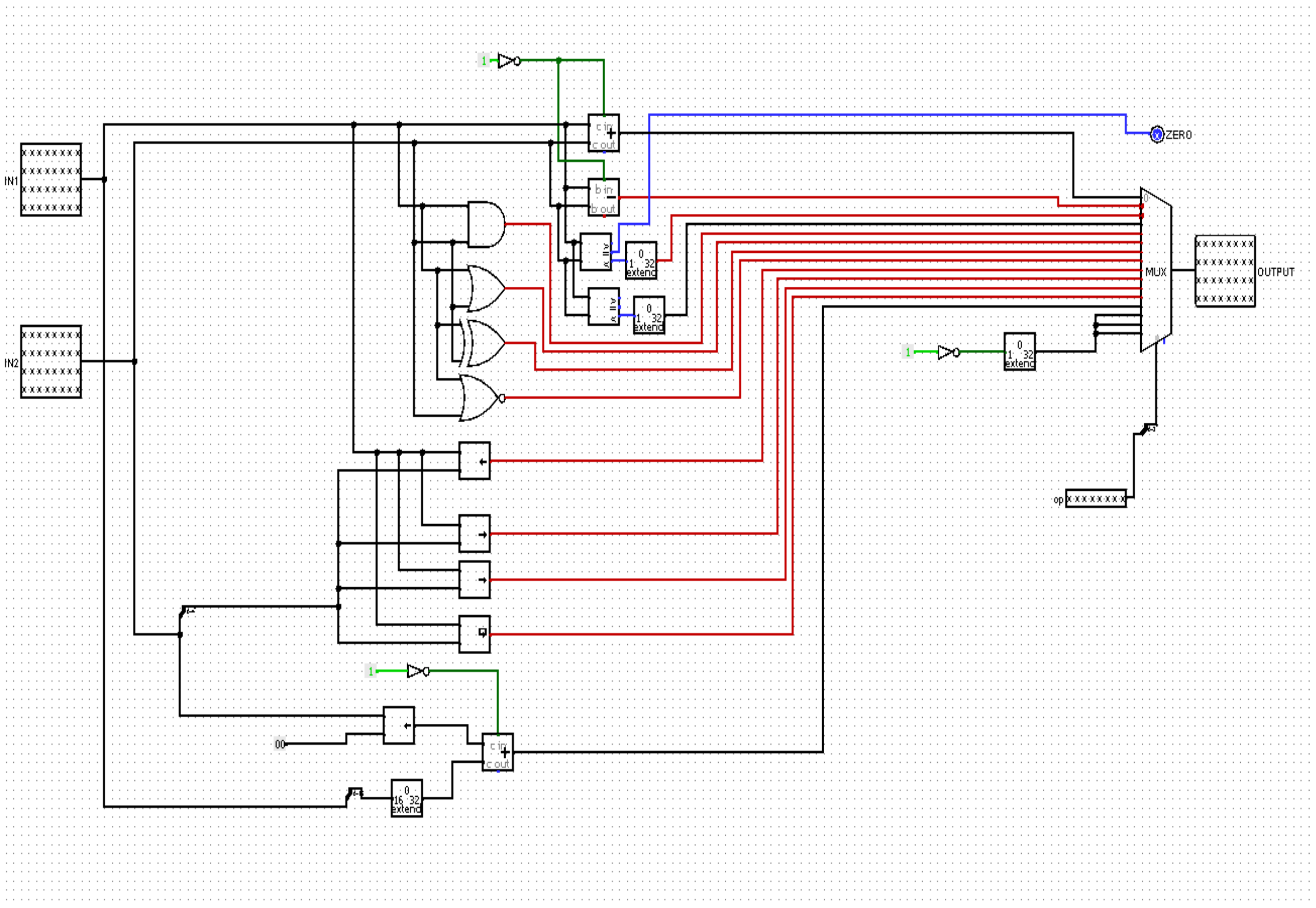
- It’s enabled using constant 1 and have an address of the pc value .
- It’s load port is connected to “PC_signal”.
- The IM generates the 32bit instruction where the upper eight represents the op code.

Register File :



- It consists of [16:32] bit registers with a constant one with value of zero.
- The two read registers Ra and Rb are selected using two 32 bit data multiplexers with a 4bit selectors .
- The selector for Ra take the value of [16:19] bits in the instruction code ,while that used to get Rb takes the value [12:15].
- The selected registers' value appear at the Ra and Rb ports.
- The write register Rd is selected using a 4 to 16 decoder , where the four bits have the value [20:23]
- Each bit of the sixteen is anded with the signal "regwrite" and connected to one of the 16 registers enable port.
- The enabled register gets the value of the input Rd.

ALU :



- It has three input ports (the lower 4bits of the op code , IN1 and IN2).
- It can perform 13 arithmetic and logical operations between the two inputs IN1 and IN2.
- It's outputs are output and zero , where output is the result of the chosen operation and have 32 bits ,while the zero is a 1bit signal generated from the comparison of IN1 and IN2 (with a value of 1 when they are equal).
- The required output of a specific operation is selected using the 15x1 MUX .
- The selector is connected to the lower 4 bits of the op code

- The following table shows the selector value for each instruction

Upper 4 bits of the opcode	Corresponding Instructions	operation
0	add , lw , sw , addi	add
1	sub	subtract
2	slt , slti	Set less than
3	sltu , sltiu	Unsigned set less than
4	and , andi	and
5	Or , ori	or
6	Xor , xori	Exclusive or
7	Nor , nori	nor
8	sll	Shift left logical
9	srl	Shift right logical
10	sra	Shift right arithmetic
11	ror	Rotate right
12	lui	Load upper immediate

- The value of IN2 can take one of the following values
  register Rb in case of branch and R-type instructions
  extended immediate [0:11] in case of load and store
  extended immediate [0:15] in case of I-type

- This choice is decided using two multiplexers with selectors I0 and I1 ,which are generated from the control unit

DM :

- It’s controlled using two signals memread (must be equale to 1 in case of load) and memwrite (must be equale to 1 in case of store) .

- In case of load it generates the 32bit data at it’s output which are driven to a multiplexer to be written back in the register file .

- In case of store it stores the value of register Rb .

Control Unit

Output signal	use
j	Jump instruction
jal	Jump and link instruction
jr	Jump register instruction
beq	Branch if equal
bne	Branch if not equal
I0	Select between extended imm^{16} and Rb
I1	Select between the previous choice and extended imm^{12} to be IN2 in ALU
memread	Is 1 for load
memwrite	Is 1 for store
regwrite	Enable writing data in register file
PC_signal	Is 1 for all 16 instructions otherwise it disables the pc register
memtoreg	Select the written back to register data (from ALU or DM)
zero_ext	It determine whether the 16bit immediate in Itype would be sign(=0) or zero extended (=1)

Truth table

4b Ub	jump	jr	branch	I0	I1	mem read	mem write	WB	memtoreg	ins
0	0	0	0	0	0	0	0	1	0	R
1	0	0	0	x	1	1	0	1	1	lw
2	0	0	0	x	1	0	1	0	x	sw
3	0	0	1	0	0	0	0	0	x	branch
4	0	1	0	x	x	0	0	0	x	jr
5	0	0	0	1	0	0	0	1	0	I
6	1	0	0	x	x	0	0	0	x	J/jal

Using k maps the following equations were derived where the upper 4bits of the opcode are ABCD and the least sig bit is H

$$\text{jump} = BC$$
$$\text{jr} = B\overline{C}\,\overline{D}$$
$$I0 = B$$
$$\text{branch} = CD$$
$$I1 = C\overline{D} + \overline{B}\,\overline{C}D = C\overline{D} + \text{memread}$$
$$\text{memwrite} = \overline{B}C\overline{D}$$
$$WB = \overline{C}(D + \overline{B})$$
$$\text{memtoreg} = \overline{B}D$$
$$j = \text{jump and } \overline{H}$$

$$\boxed{\begin{aligned} \text{Jal} &= \text{jump and H} \\ \text{beq} &= \text{branch and } \overline{\text{H}} \\ \text{bne} &= \text{branch and H} \end{aligned}}$$

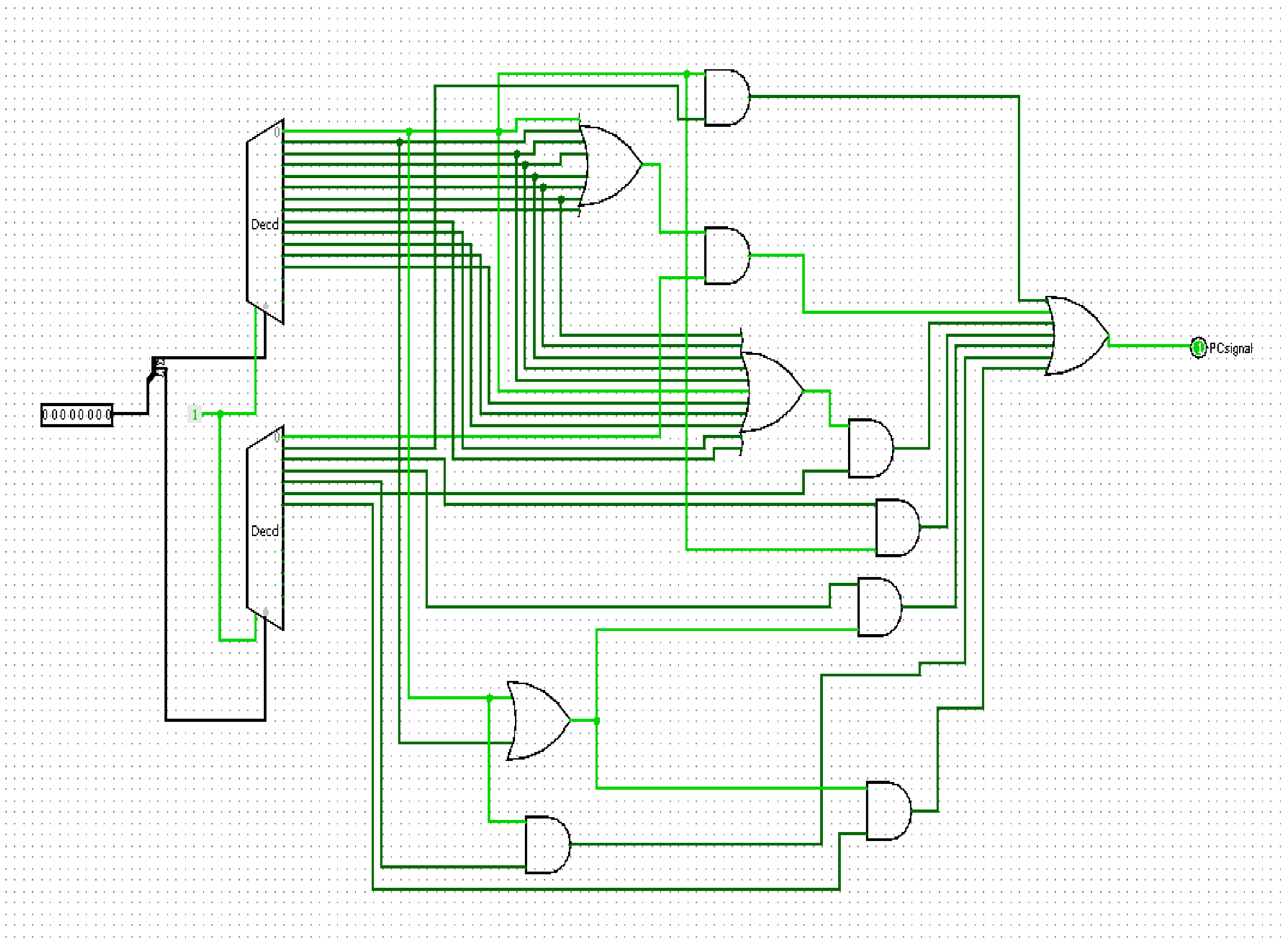
zero_ext is a function of the lower 4 bits of the opcode "EFGH"

$$\boxed{\text{zero\_ext} = \text{F (AnorD + AnorC)}}$$

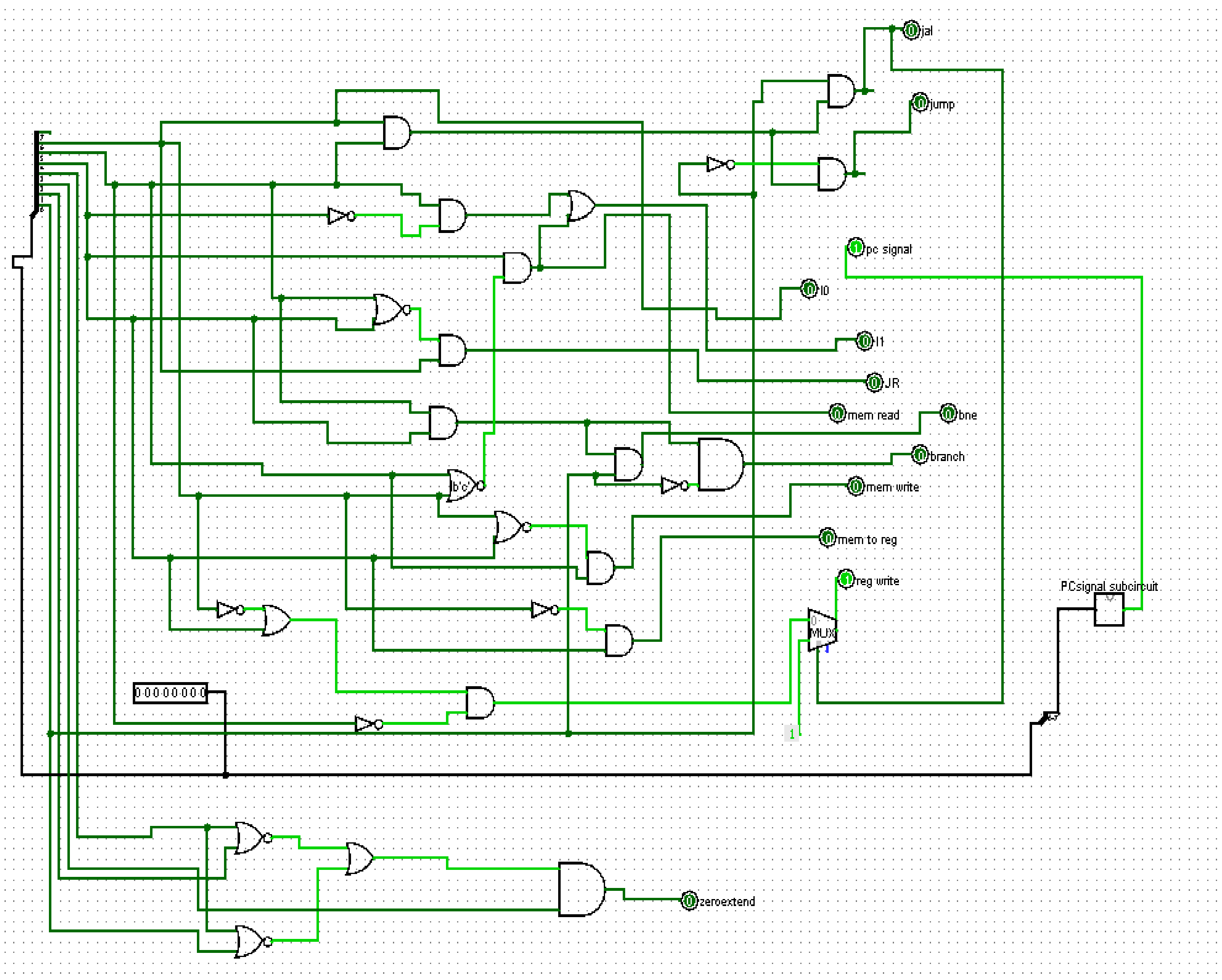
PC_signal equals 1 for all values of opcode corresponding to the 16 instructions

Upper 4 bits in decimal	Lower 4 bits in decimal	Instruction
0	0:7	Rtype
1	0	load
2	0	store
3	0:1	Beq and bne
4	0	jr
5	0	Itype
	2:6	
	8:12	
6	0:1	J and jal

This table could be implemented using two 4X16 decoders as shown in the following figure

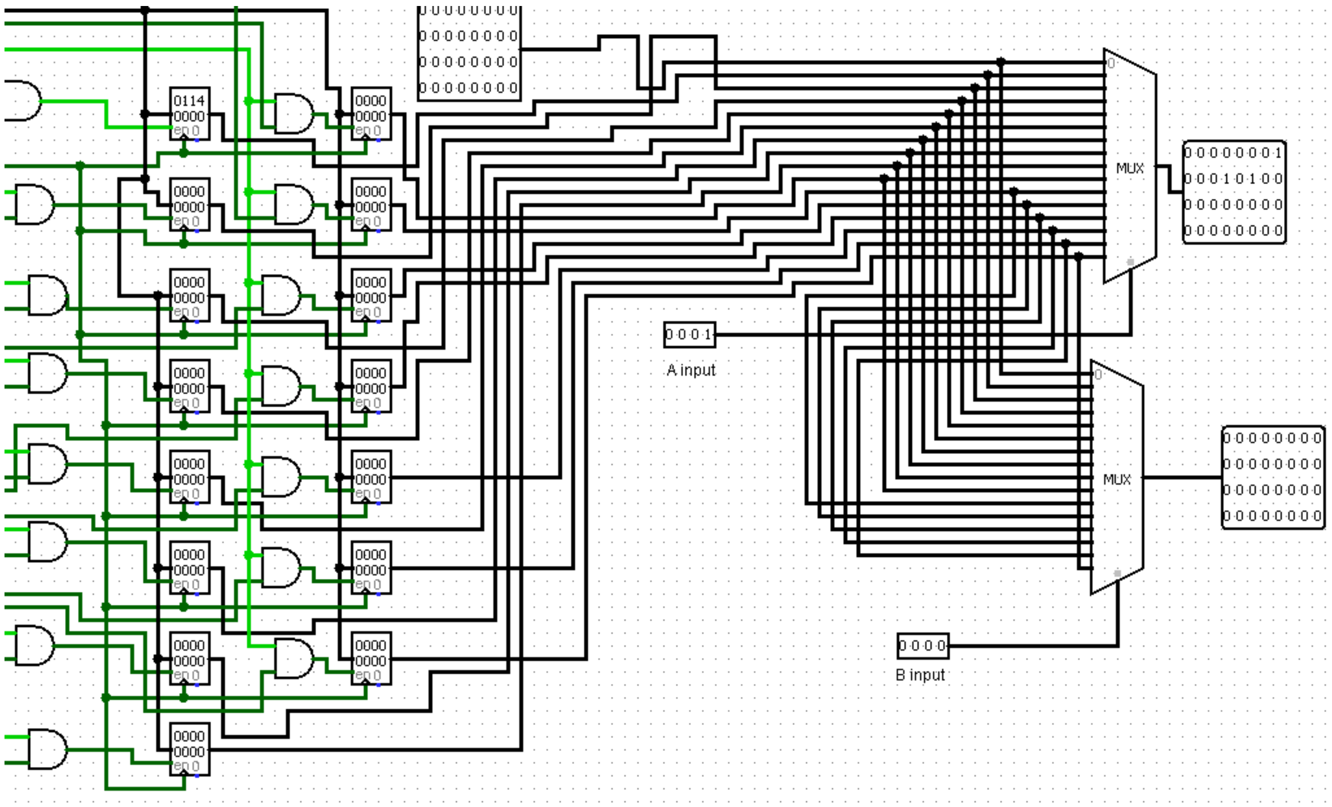


Using logic gates this control unit was constructed

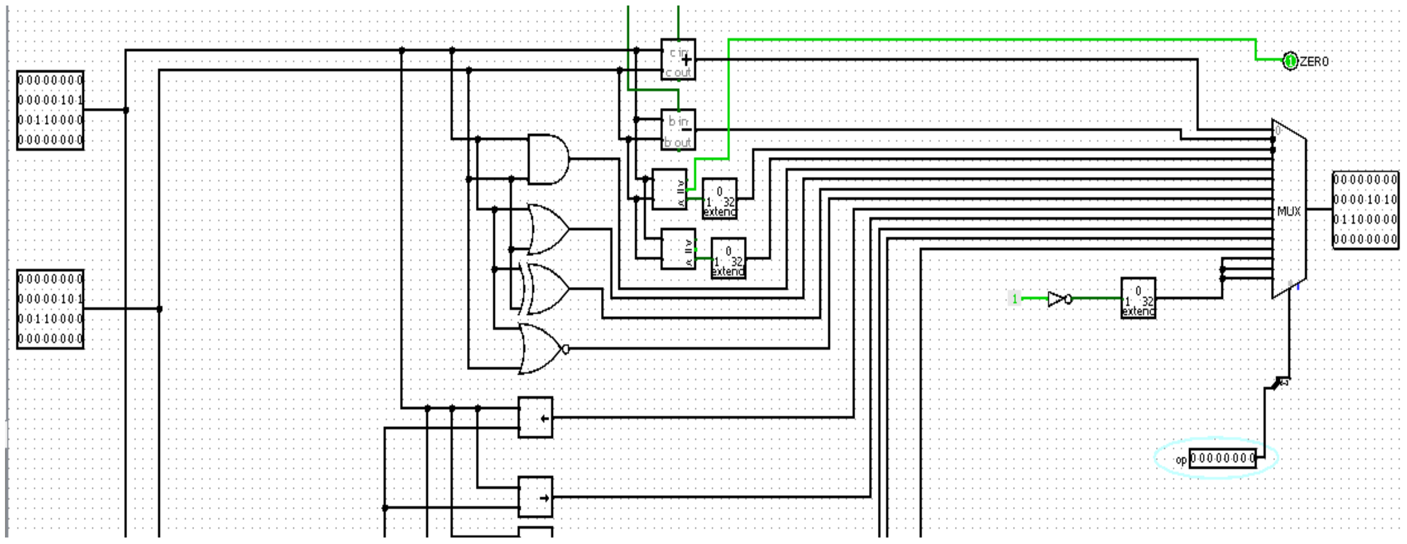


SIMULATION

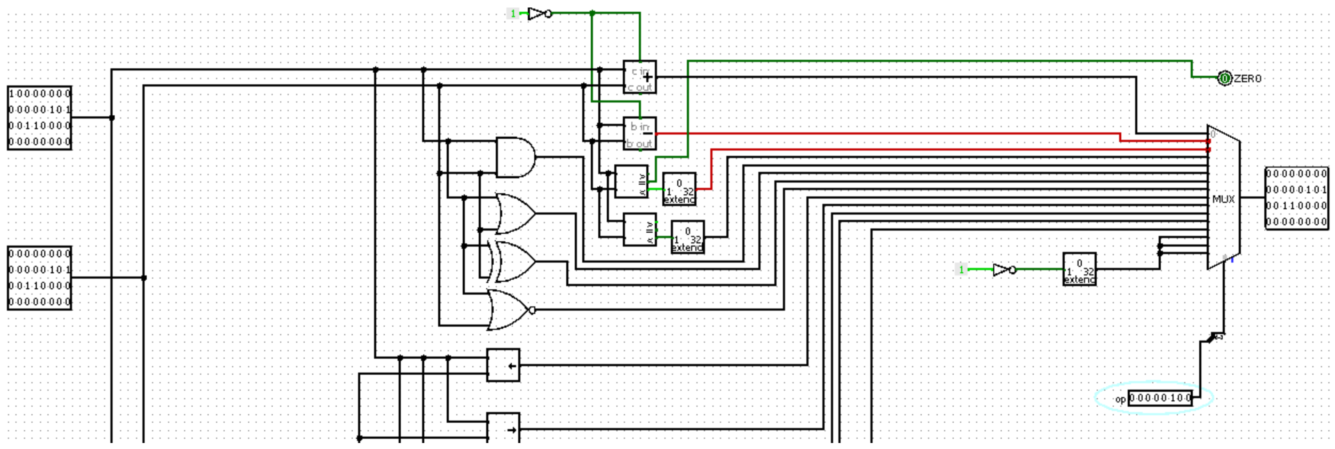
Register File



ALU



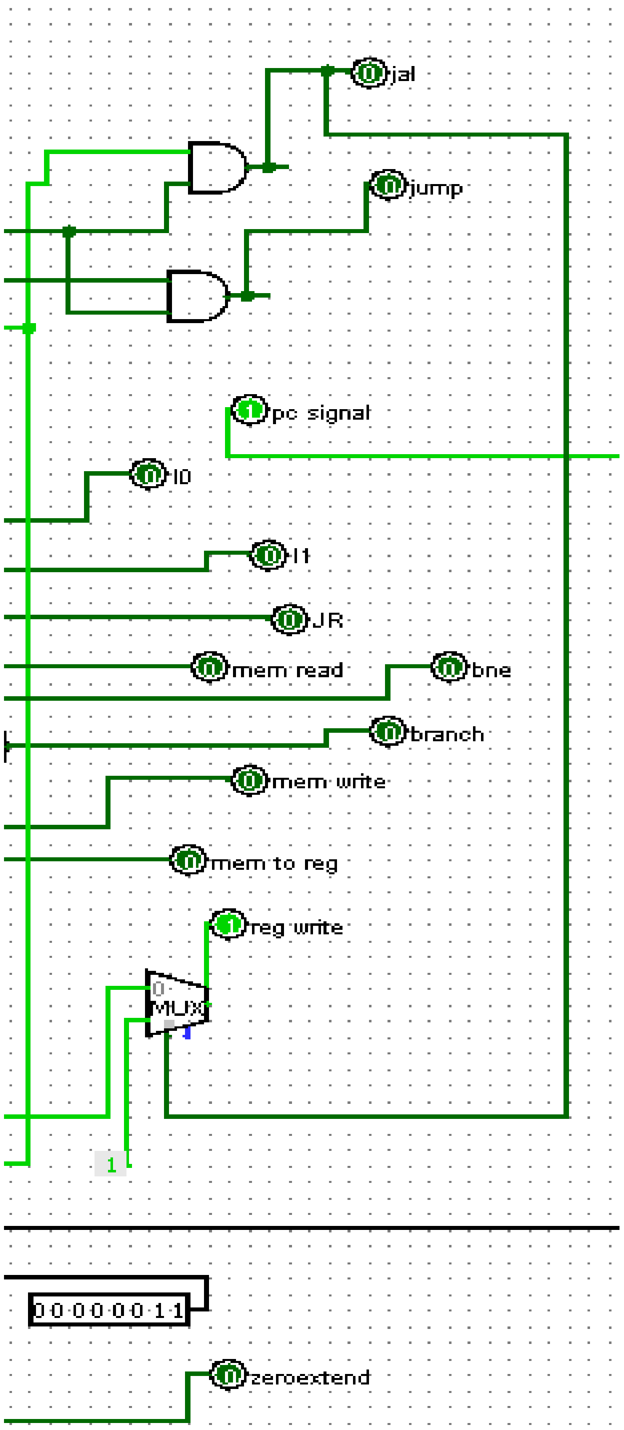
ZERO
MUX
op 00000000
00000000
00000101
00110000
00000000
00001010
01100000



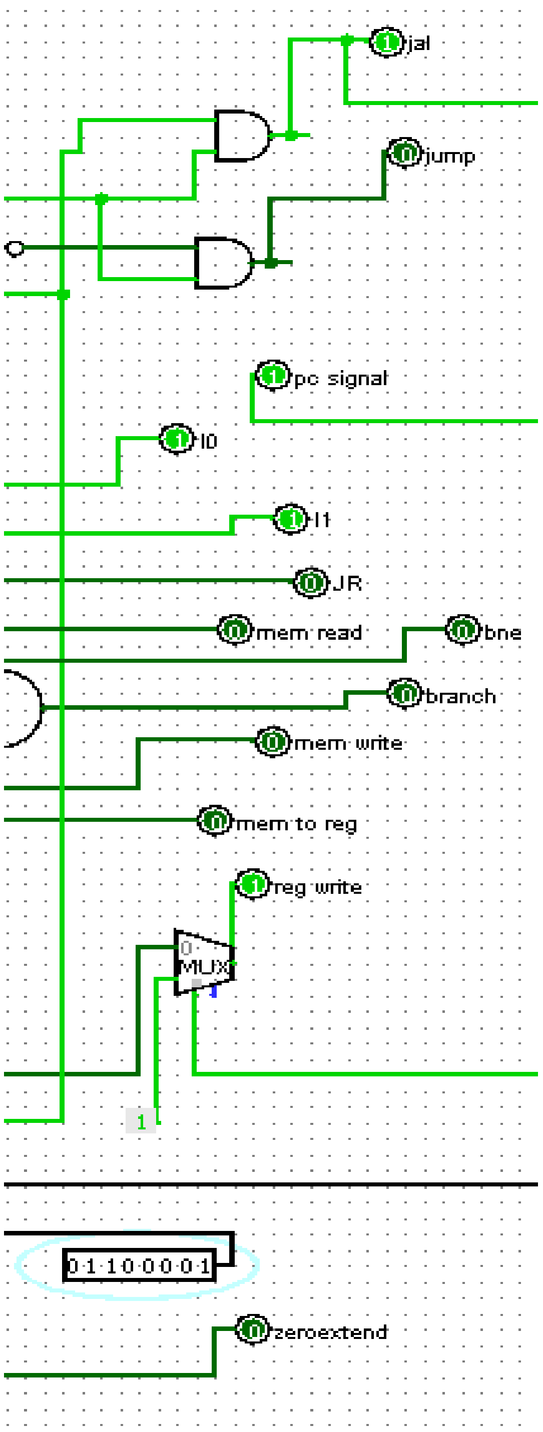
ZERO
MUX
op 00000100
10000000
00000101
00110000
00000000

Control Unit

sltu



jal



Full Datapath Testing

- initialize PC to zero
- Initialize Data Memory as shown

```
0 00000000
1 00000000
2 00000000
3 00000007
```

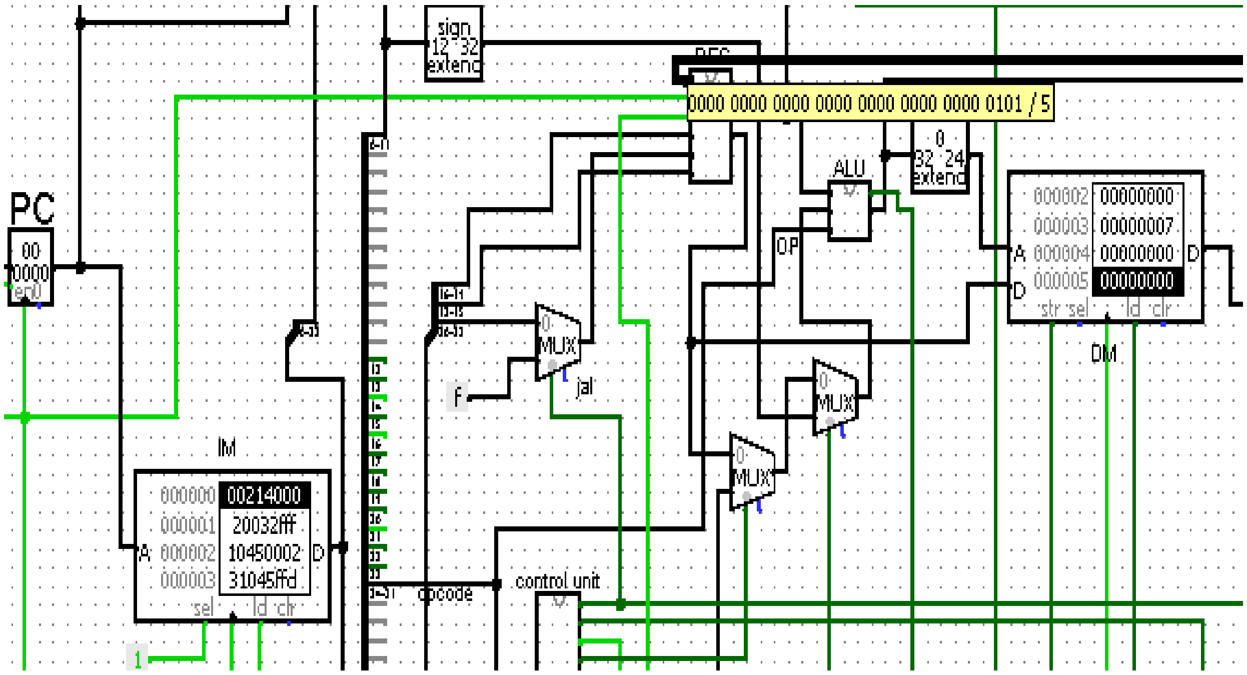
- initialize the following registers as shown in the table

register	R1	R2	R3	R4	R5
value	5	3	2	0	1

- Store these instructions in the IM

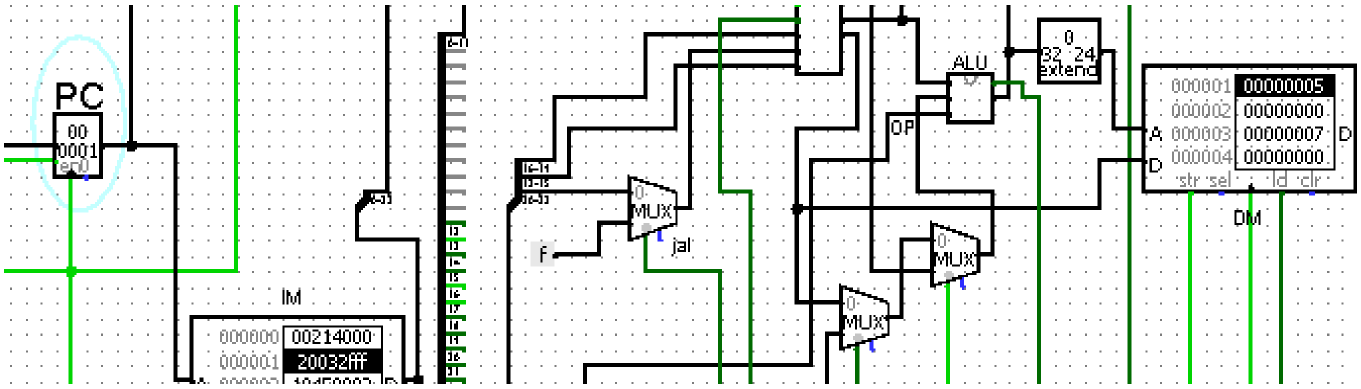
Address in IM	Instruction in assembly	Machine language instruction “value in IM” in hexadecimal
0	Add R2 ,R1 ,R4	00214000
1	Sw R2 , -1 (R3)	20032FFF
2	Lw R4 , 2 (R5)	10450002
3	Bne R4 , R5 ,-3	31045FFD

Instruction 0



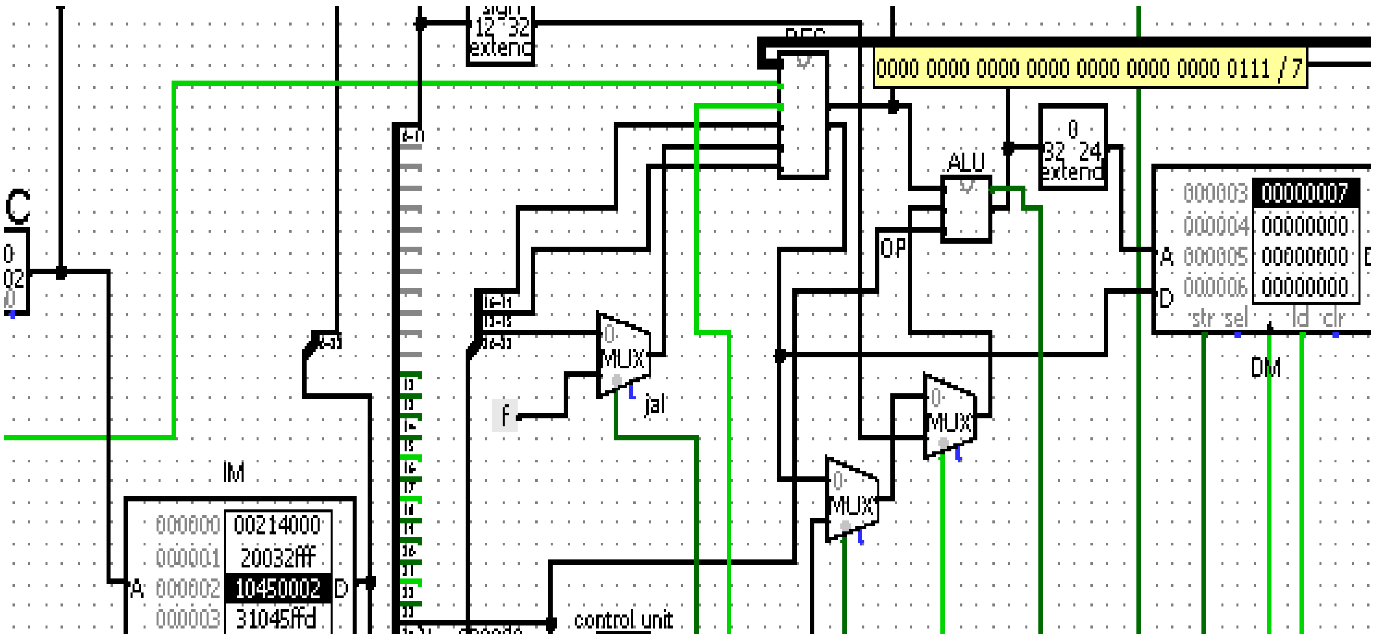
- PC = 0
- Rd(R2) = 5 + 0 , this value is written back into register file

Instruction 1



- PC = 1
- value of R2 = 5 is stored in the memory
- The address of the stored value is R3 - 1 = 1

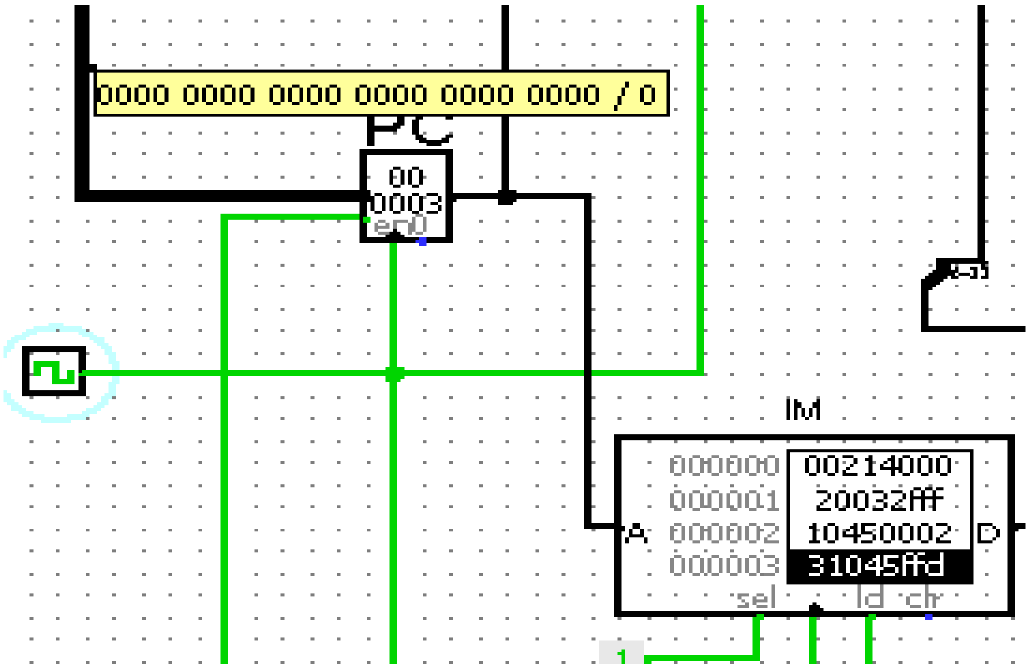
Instruction 2



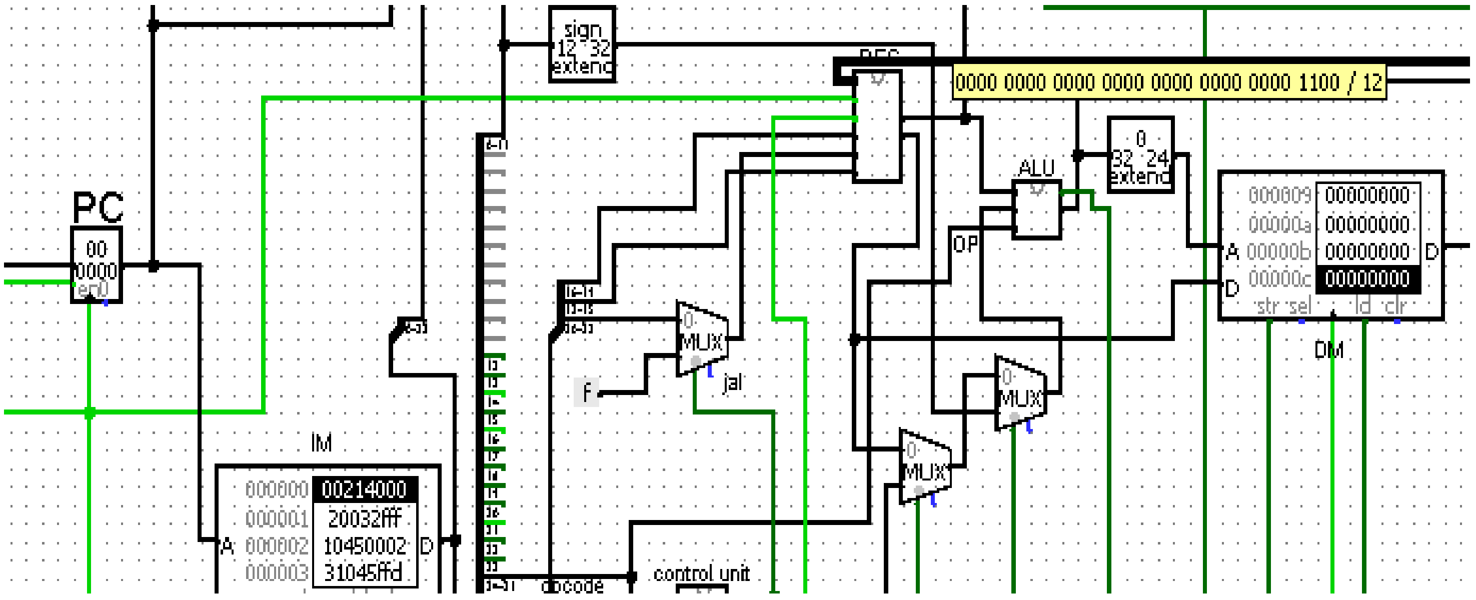
- PC = 2
- address of loaded data = R5 + 2 = 3
- data is loaded in R4 , so R4 = 7

Instruction 3

- PC = 3
- R4 != R5 (i.e, 0 != 1)
- next PC = PC – 3 = 0

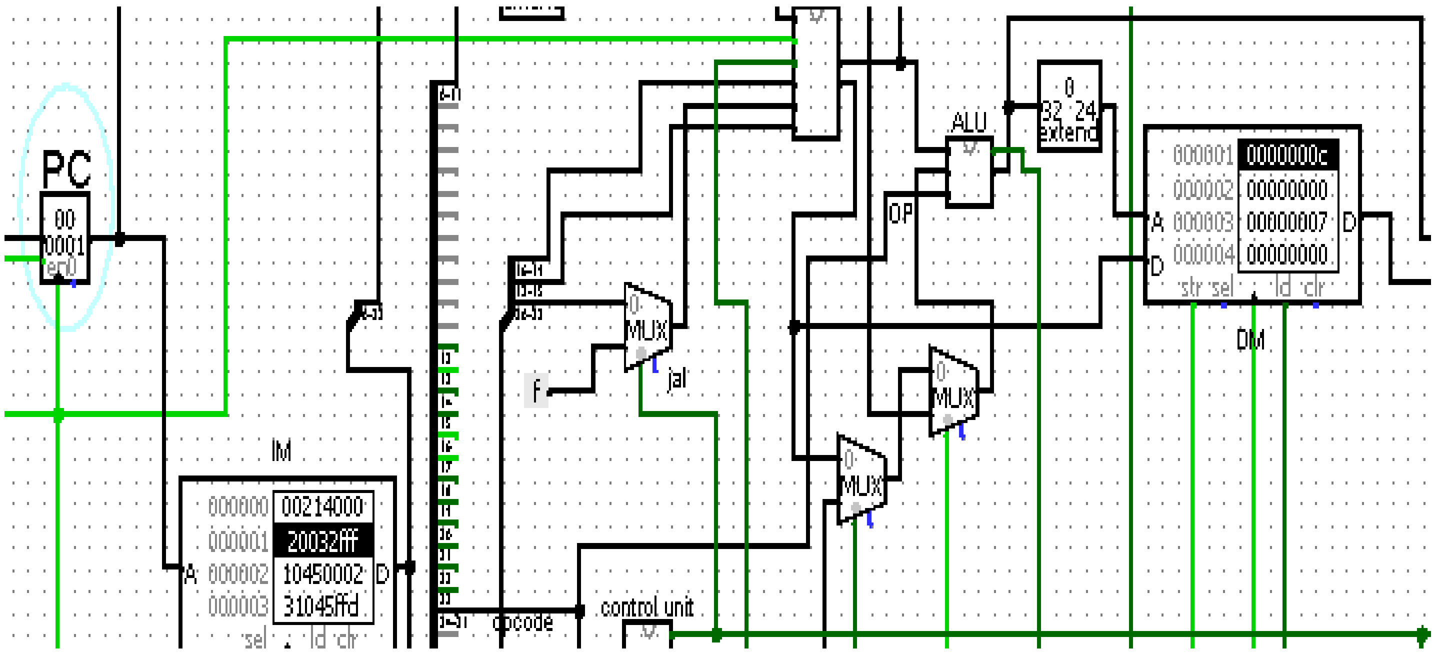


Instruction 0 “ second loop ”



- PC = 0
- written back data into R2 = R1 + R4 = 5 + 7 = 12

<u>Instruction 1 “ second loop ”</u>



- PC = 1
- R2 = 12 is stored into memory at address = R3 -1 = 1