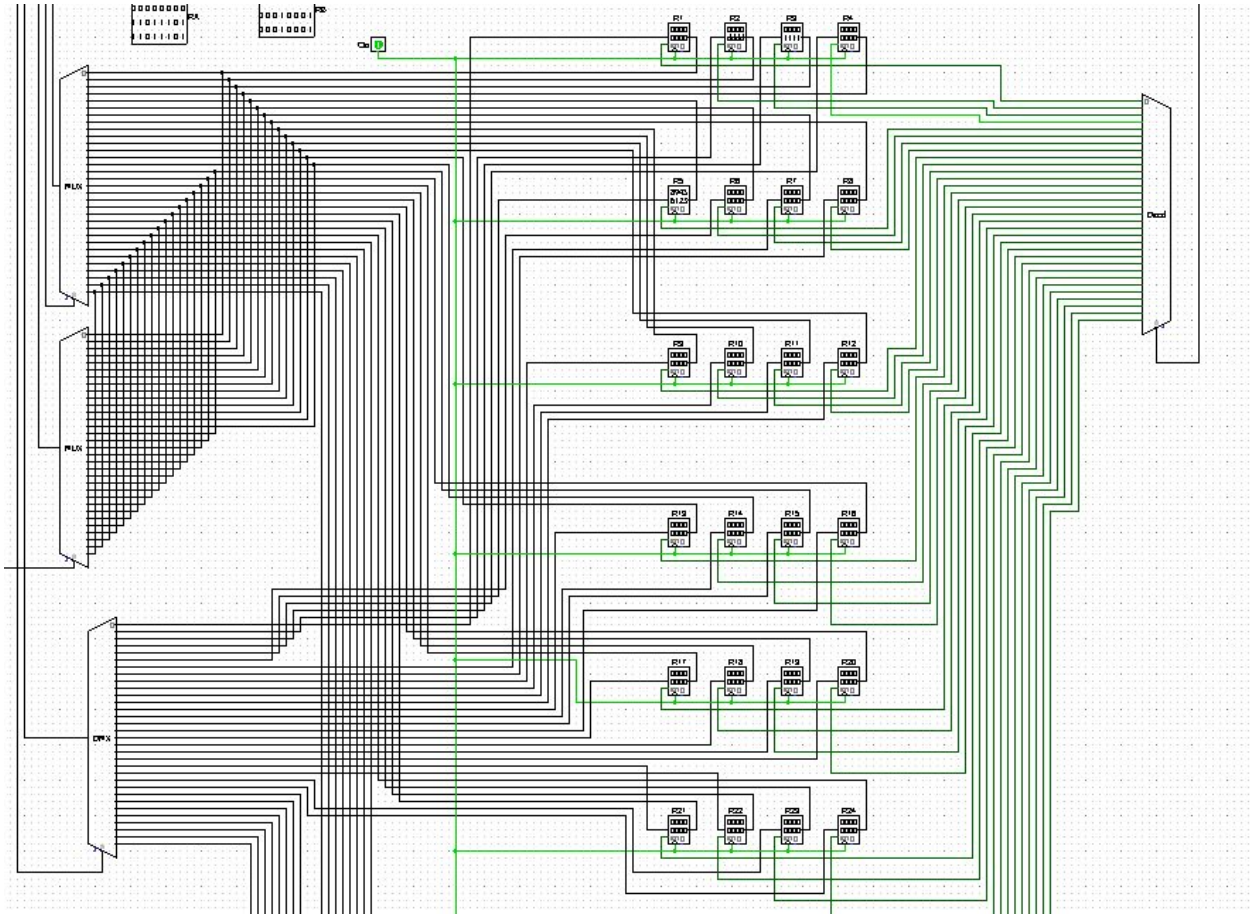
32 BIT CENTRAL PROCESSING UNIT

This mini project is a 32 bit CPU that follows 5 stage data path. It performs simple arithmetic operations like Addition(ADD), Subtraction (SUB), logical operations like OR, AND, XOR, register MOVE operation and memory operations LOAD and STORE. LOAD and STORE takes 3 clock cycles and other operations take 4 clock cycles to execute.

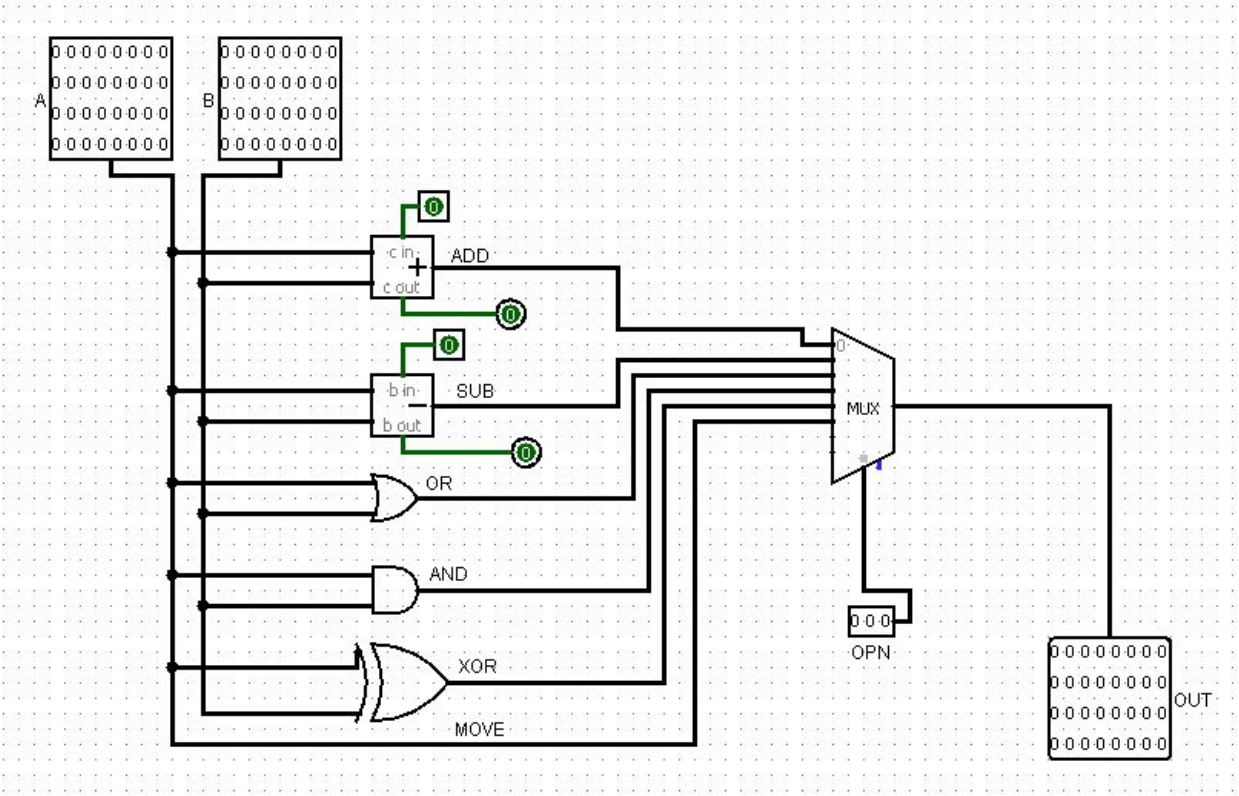
The whole circuit is divided into different modules:

1) Register File:
   It contains 32 registers which can store 32 bit data. Each register is identified by a specific 5 bit long code. Three registers can be selected at once, two for reading and one for writing. Only that register will be enabled which is required to be written. This register is enabled by a decoder.



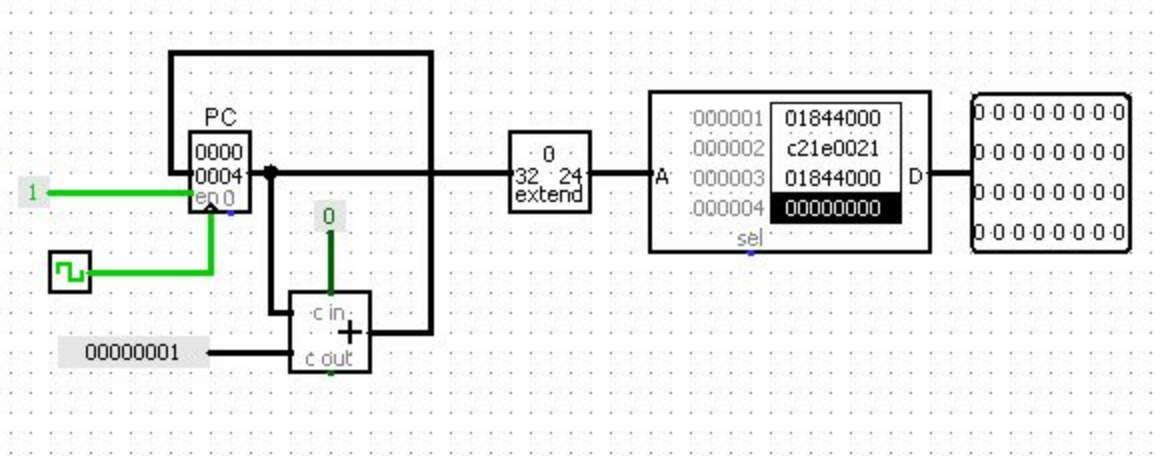
2) ALU (Arithmetic and Logical Unit):

It takes the content of two registers as input and performs operations like ADD, SUB, OR, AND etc on them. The desired output is selected using a multiplexer.



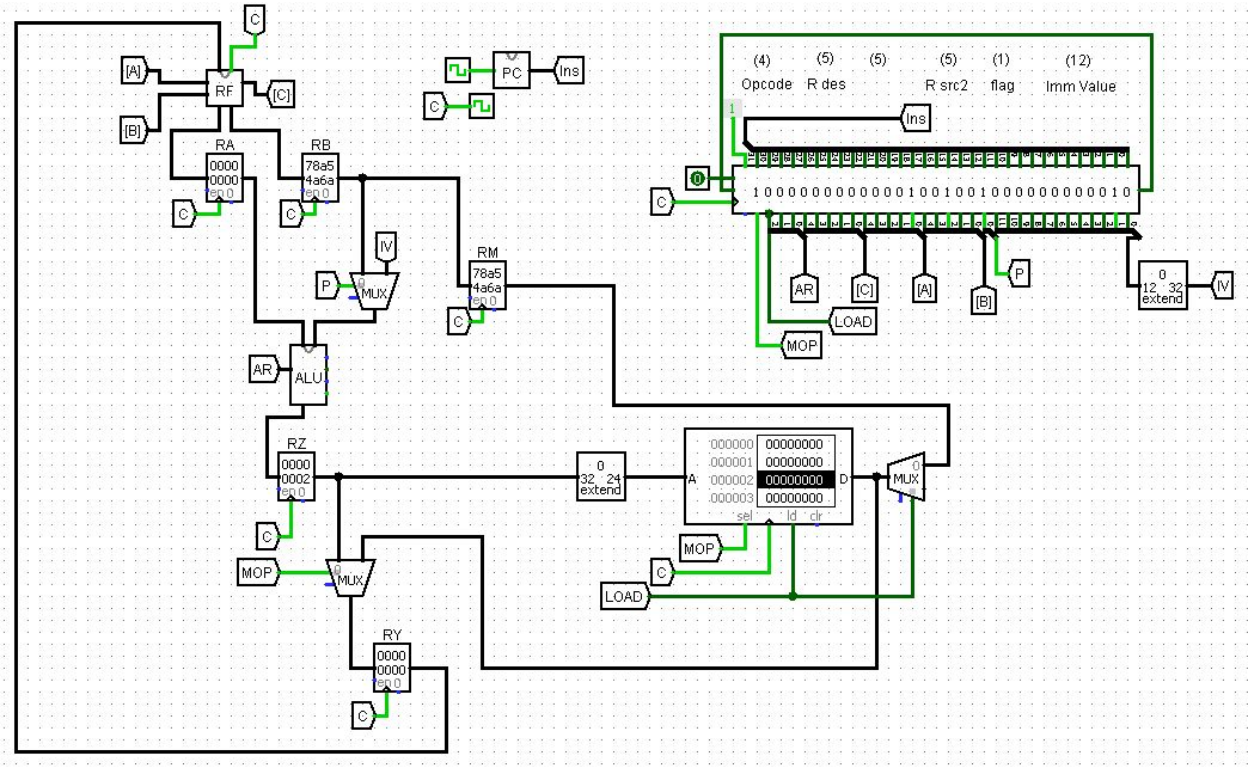
4) Program Counter:

It consists of a ROM which consists of 32 bit instructions and a program counter register. This register consists of the address of the ROM from which next instruction has to be fetched. After every clock cycle, content of the register increments by 1 and moves to the address of the next instruction. The instruction is fetched into the instruction register which is in the CPU module.



5) CPU:

It is similar to the 5 stage data path usually followed. It takes the opcodes for 2 source registers and any immediate value, performs operations on them and store the resultant value in the destination register. Each decision made during the process is based on the Instruction opcode values. This module also contains the instruction register.
It also consists of a RAM for storing data and updating it. It takes a 32 bit address as input and either loads data from the address into the register or stores data into the main memory from a source register. The above operations are specified as LOAD and STORE.The memory address size is limited to 24 bits in logisim. Hence, in the 32 bit address only the 24 least significant bits are considered.



The various operations supported are:

1) ADD
2) SUB
3) MOV
4) OR
5) AND
6) XOR

The 5 stages followed for each instruction set are:

1) Instruction fetch
2) Instruction decode

3) ALU operation
4) Memory operations
5) Write back

Instruction Format:

OpCode (4 bit)	Destination Register (5 bit)	Source Register 1 (5 bit)	Source Register 2 (5 bit)	Flag for Immediate value (1 bit)	Immediate Value (12 bit)
It specifies which operation to perform	It specifies the register where the final result should be stored	It specifies the first input register	It specifies the second input register	It tells us whether to consider immediate value or not	It is the immediate value

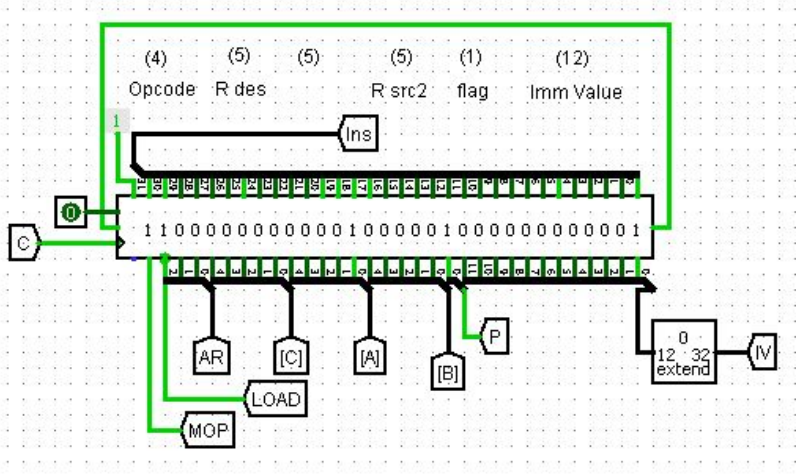
OpCode for various operations:

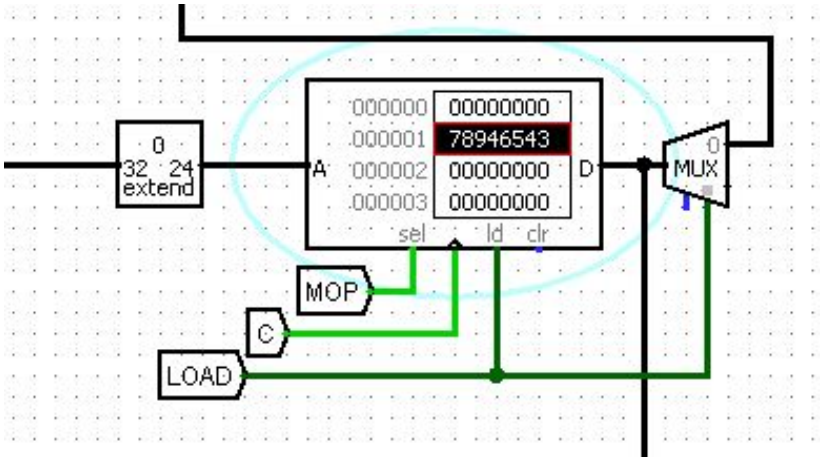
ADD	0000
SUB	0001
OR	0010
AND	0011
XOR	0100
MOVE	0101
LOAD	1100
STORE	1000

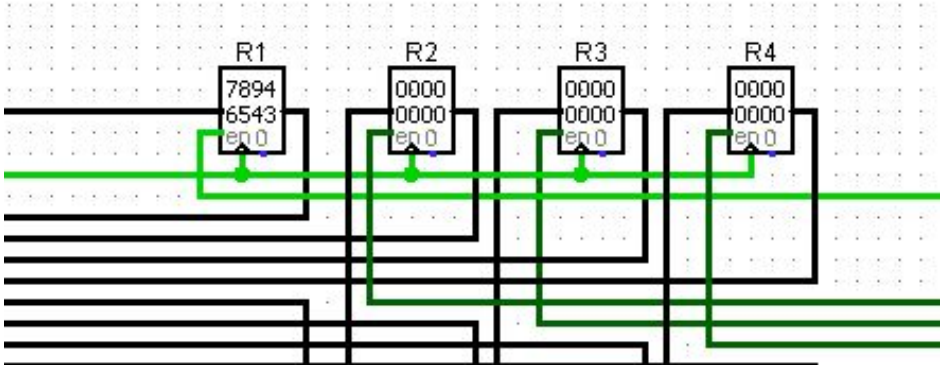
Understanding the instruction for different operations:

LOAD R1, R2, F, IV :

Load the contents of [R2] + IV in R1. Consider IV only if F = 1.
Binary instruction: 1100 00000 00001 00000 1 000000000001
Hexadecimal code: c0041001
Implementation pictures:

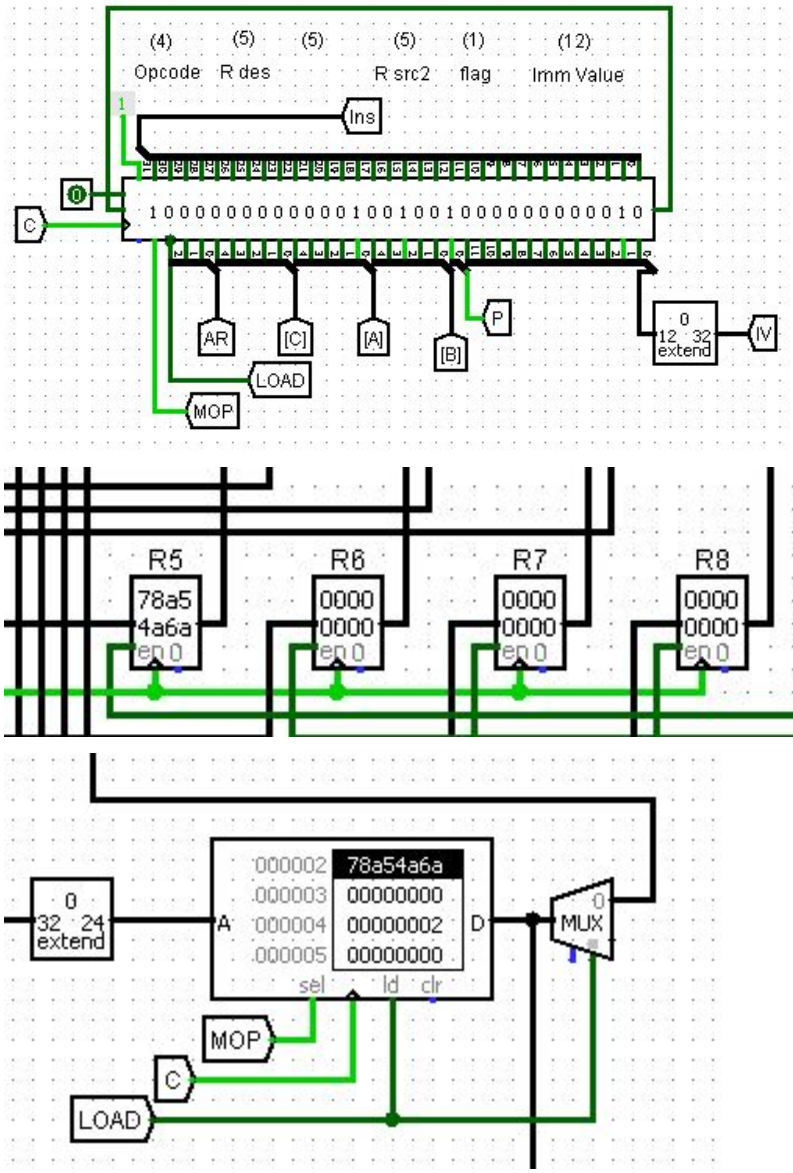






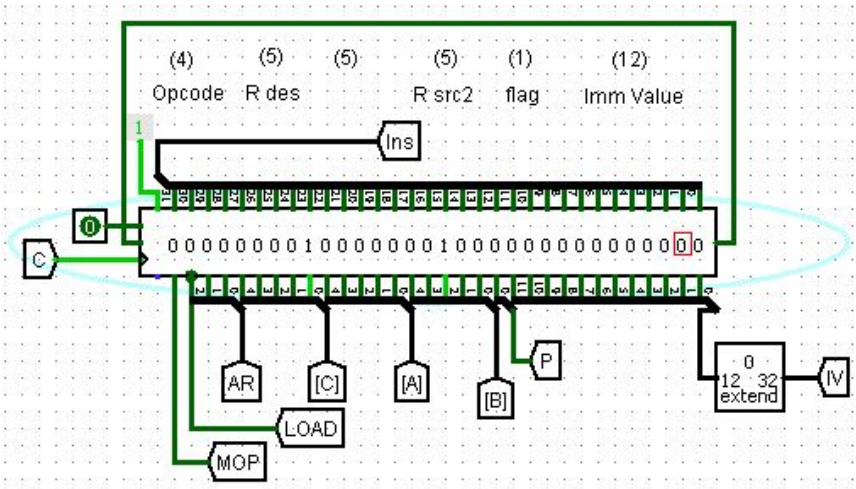
STORE X, R1, R2, F, IV :

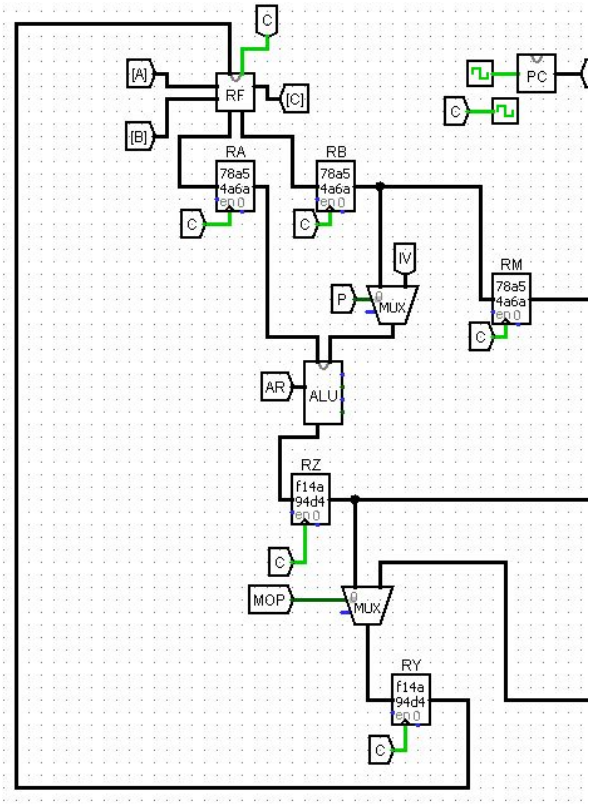
Store the contents of R2 in the memory location specified by R1+IV. Consider IV only if F = 1.
Binary instruction: 1000 00000 00001 00100 1 000000000010
Hexadecimal code: 80049002

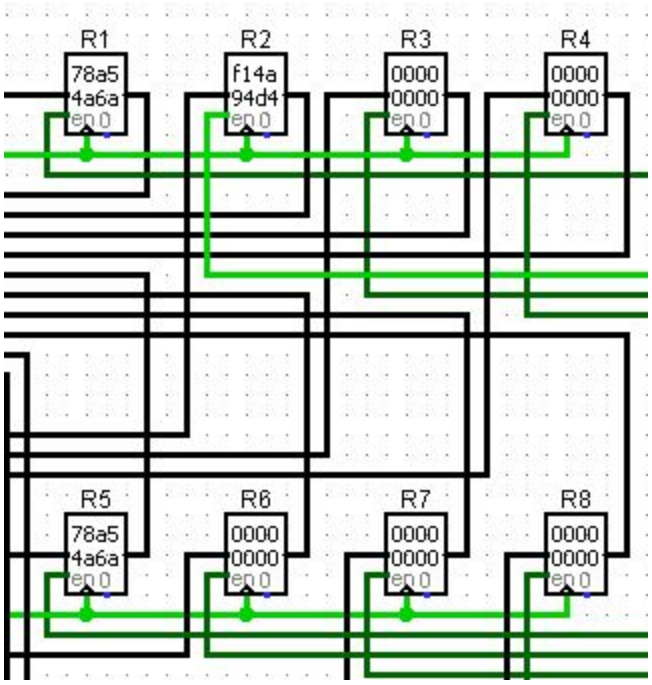


OPP R4 R1 R2 :

Operate on the contents of R2, R1 and store in R4.OPP can be ADD, SUB, OR, XOR, AND.
Binary instruction: 0000 00001 00000 00100 0 000000000000 (ADD)
Hexadecimal code: 00808000

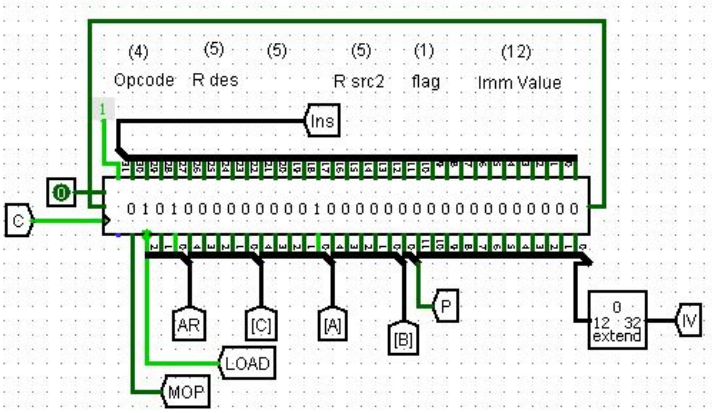


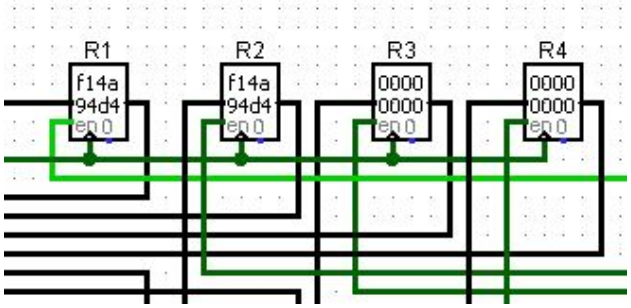


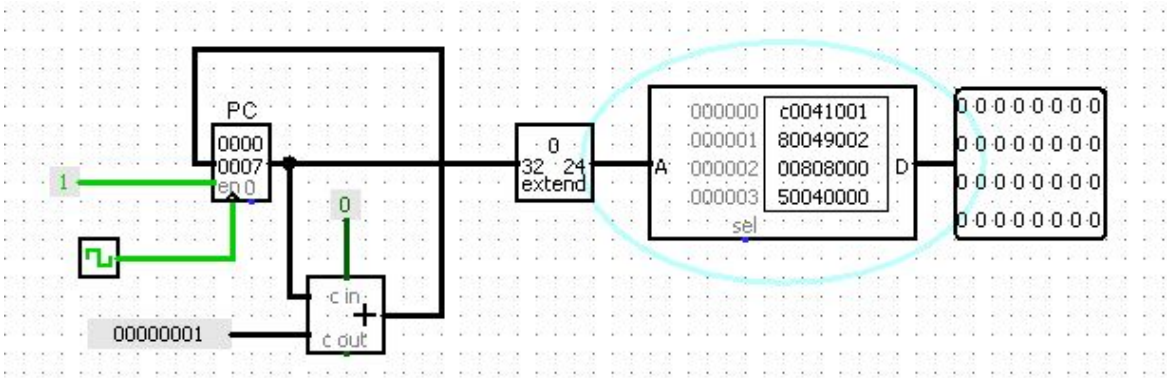


MOVE R1,R2 :

It copies the contents of R2 into R1.
Binary instruction: 0101 00000 00001 00000 0 000000000000
Hexadecimal code: 50040000







While writing all these instructions, the other bis which are sometimes not considered (according to the operation) can be kept 0. For example, when F = 0, i.e immediate is not considered, it can be made 0. Also, in store X can be anything. It can be noticed that the OpCodes for arithmetic, logical operations and move start with 0 and memory operations start with 1. In memory operations the second bit specifies whether to load or store.