MIDDLE EAST TECHNICAL UNIVERSITY

NORTHERN CYPRUS CAMPUS

Computer Engineering Program

CNG 331

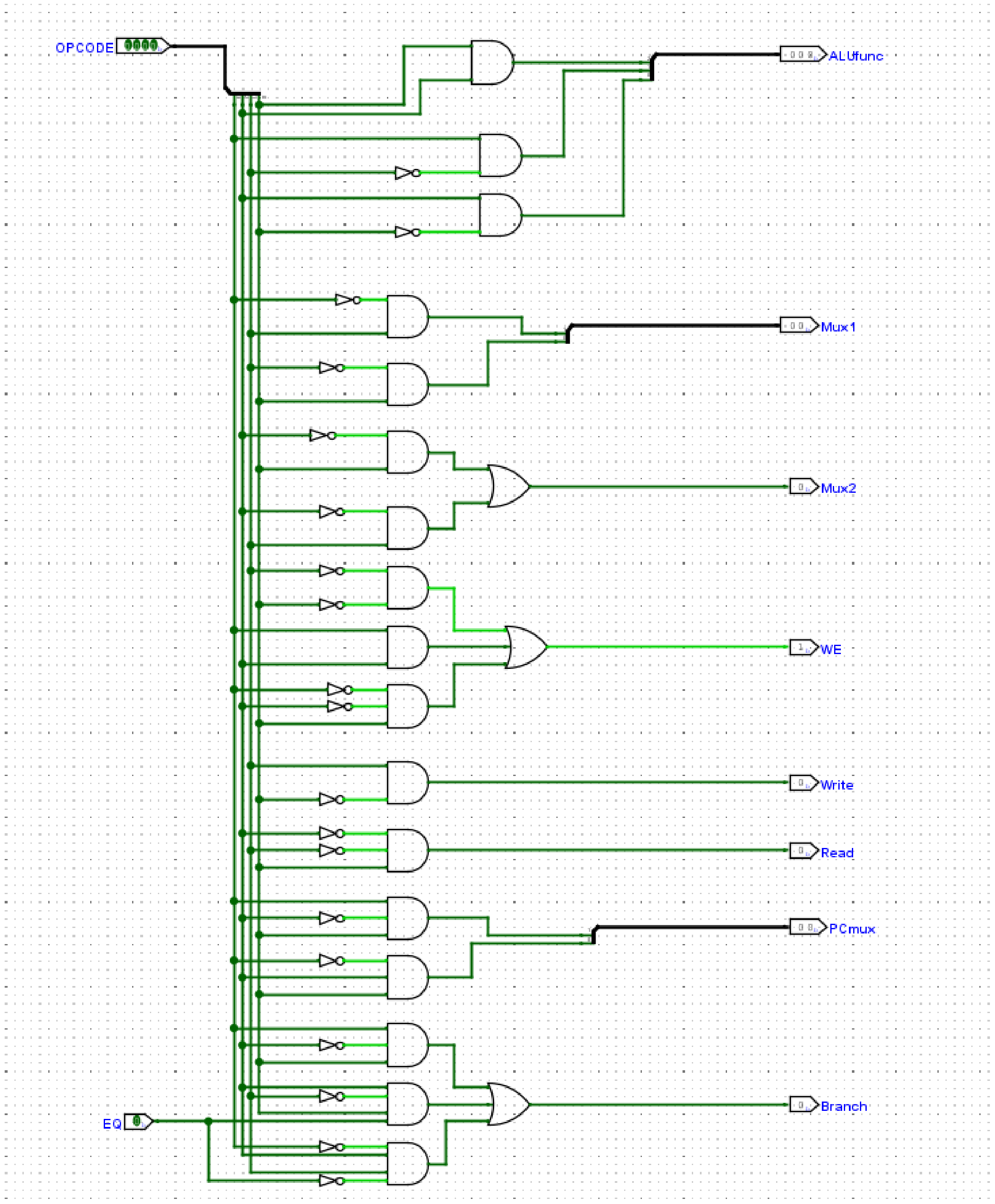
Term Project Part #3

Name: Oğuzhan Alpertürk

ID Number: 2315752

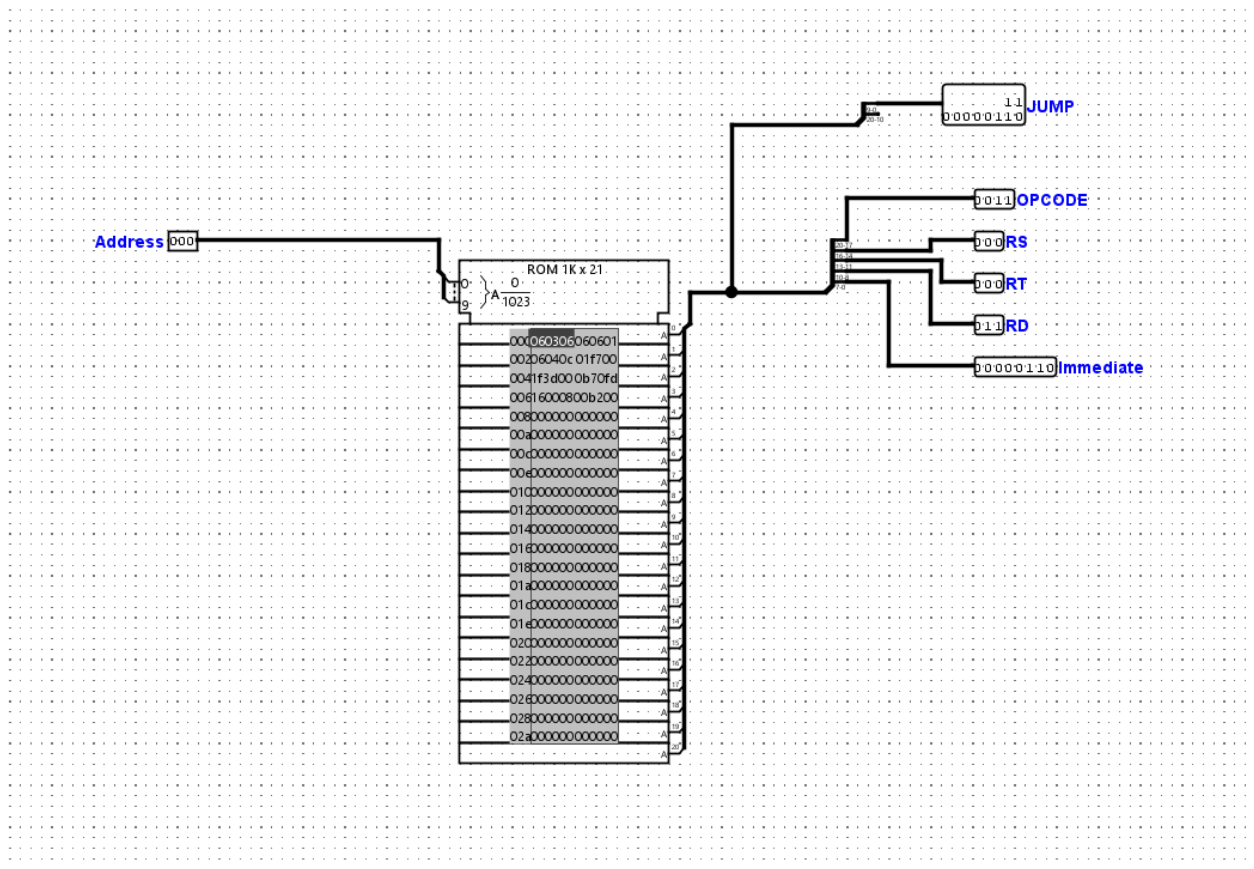
Control Unit:

The Control Unit controls the circuit components by sending necessary signals according to the opcode and EQ. I used K-Map to write a circuit for each output.

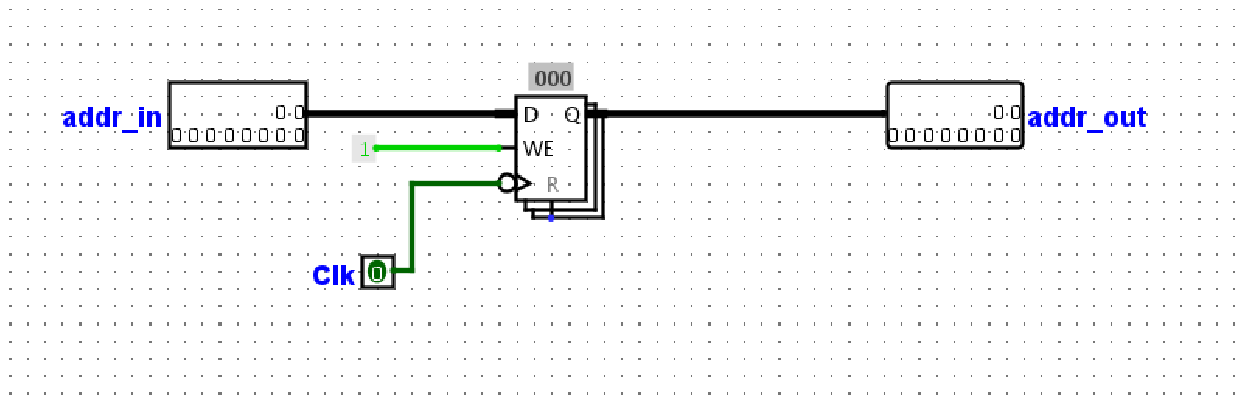


Instruction Memory (ROM):

Instruction memory stores the instructions using ROM. Takes the 10-bit address that comes from the PC (Program Counter) and outputs the 21-bit instruction in this address by dividing into opcode, rs, rt, rd, and immediate value.

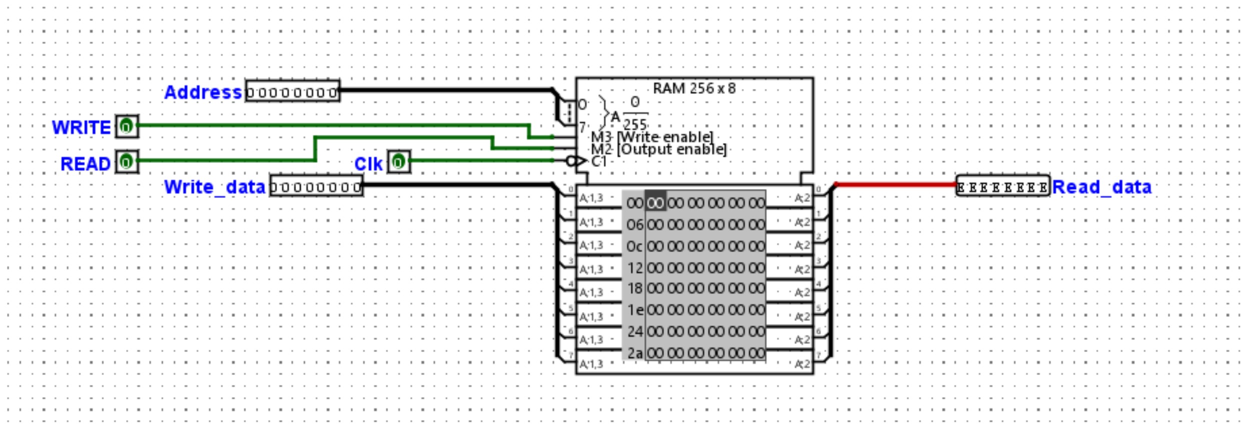


Program Counter:



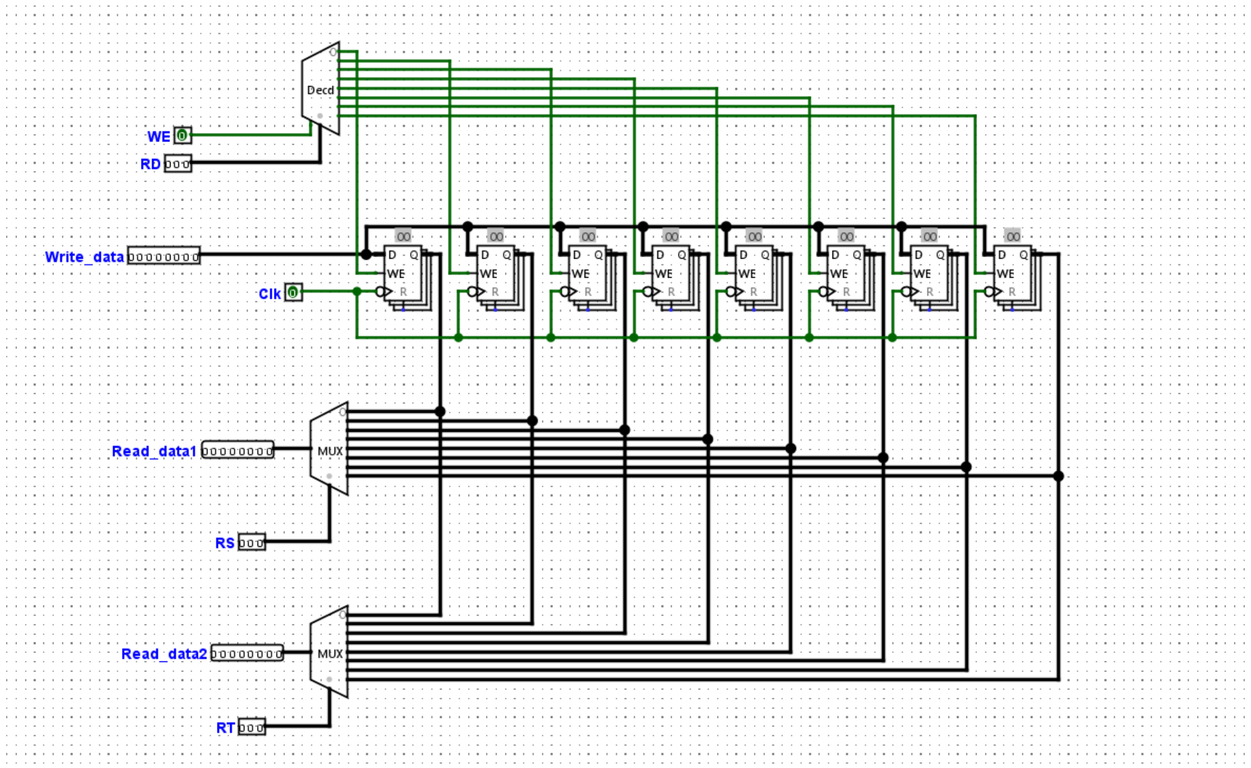
Data Memory (RAM):

Data Memory stores the data using RAM. According to the read and write signals which come from Control Unit, it writes the data in inputted address or reads the data in the given address and outputs it. I used a falling edge-triggered clock and 8-bit data width in my circuit



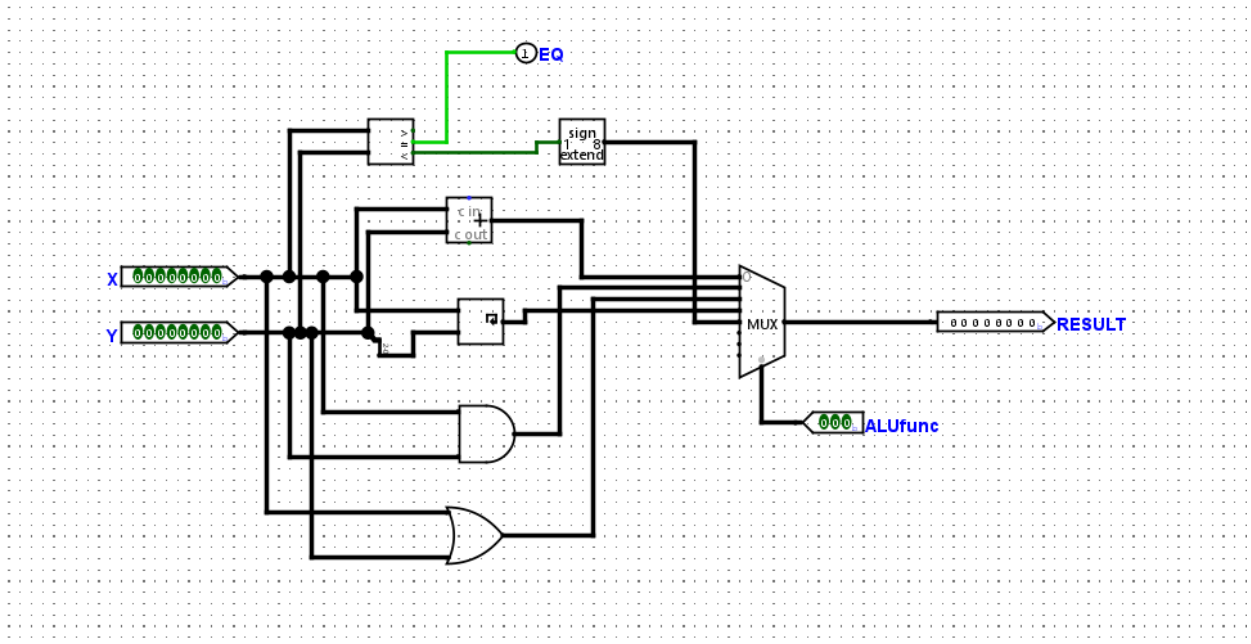
Register File:

In the register file, there are one decoder and two multiplexers which are used for writing and reading data from the register.

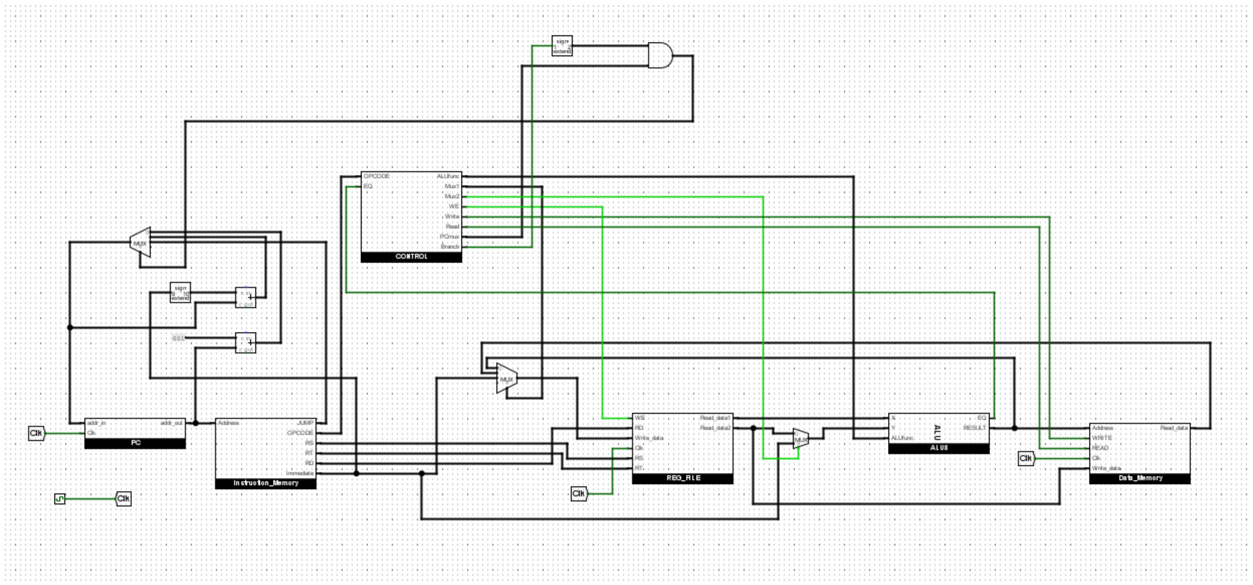


ALU:

I added a comparator and EQ flag to check the equality of the inputs. I also used less than (<) output of the comparator for SLT operation.



Whole Schematic:



Test Instructions:

The assumption for register addresses and names :

000 -> $R0

001 -> $R1

010 -> $R2

011 -> $R3

100 -> $R4

101 -> $R5

110 -> $R6

111 -> $R7

1-)

LI R3,6

Binary: 0000 0110 0000 0011 0000 0110

Hex: 060306

opcode: 0011

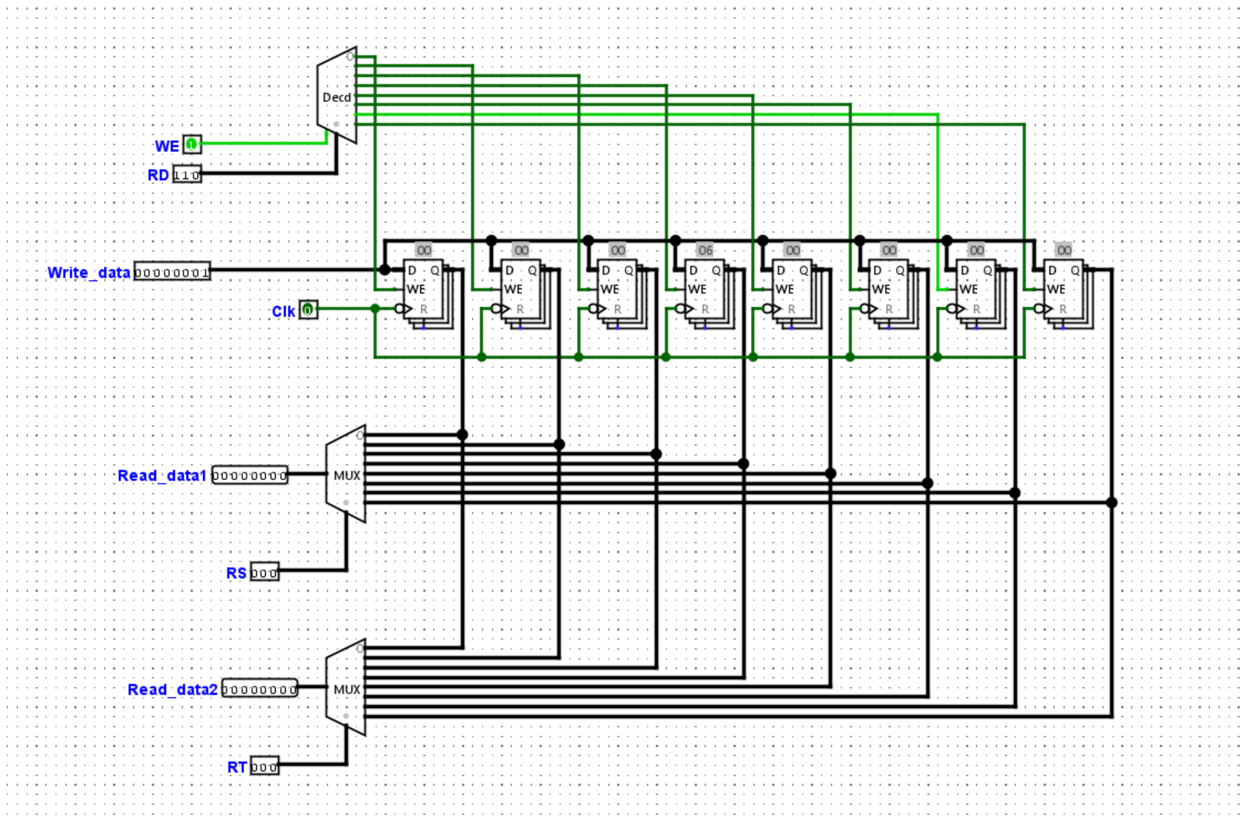
rs: 000 ($R0)

rt: 000 ($R0)

rd: 011 ($R3)

immediate: 0000 0110

The instruction loads 6 to register 3



2-)

LI R6,1

Binary: 0000 0110 0000 0110 0000 0001

Hex: 060601

opcode: 0011

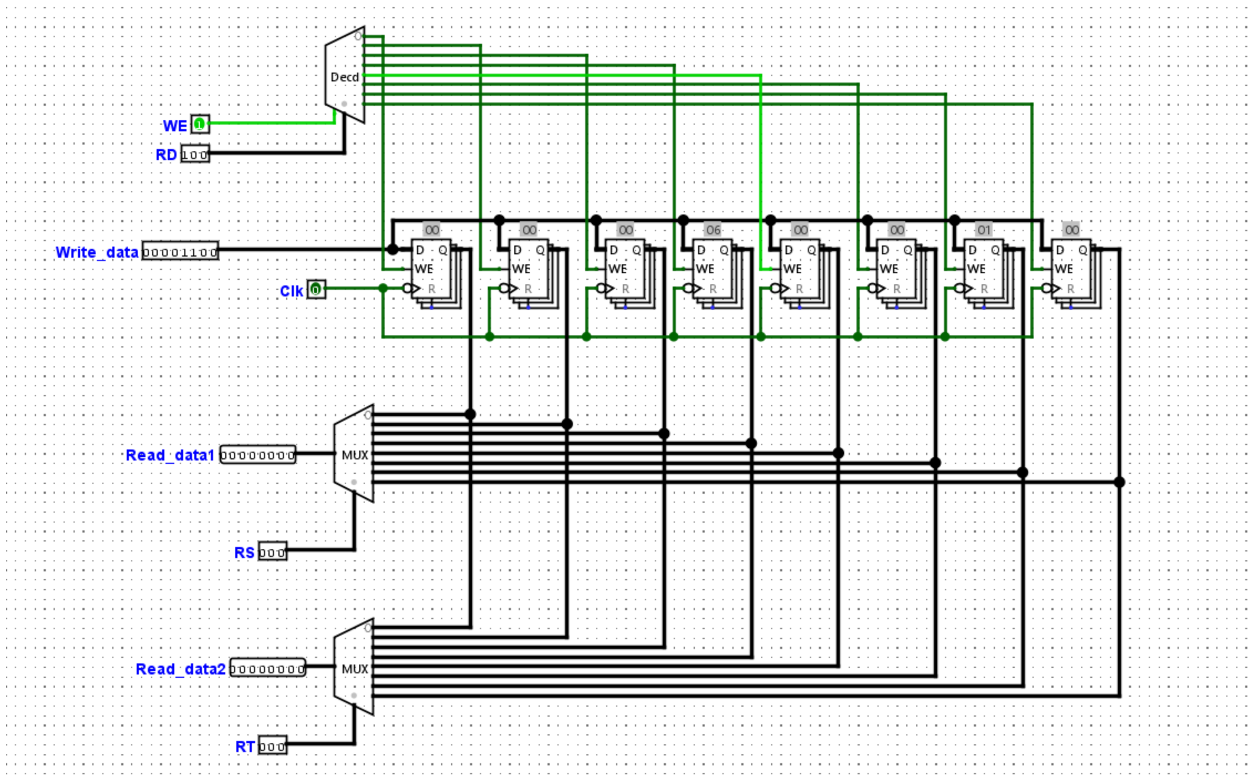
rs: 000 ($R0)

rt: 000 ($R0)

rd: 110 ($R6)

immediate: 0000 0001

The instruction loads 1 to register 6



3-)

LI R4,12

Binary: 0000 0110 0000 0100 0000 1100

Hex: 06040C

opcode: 0011

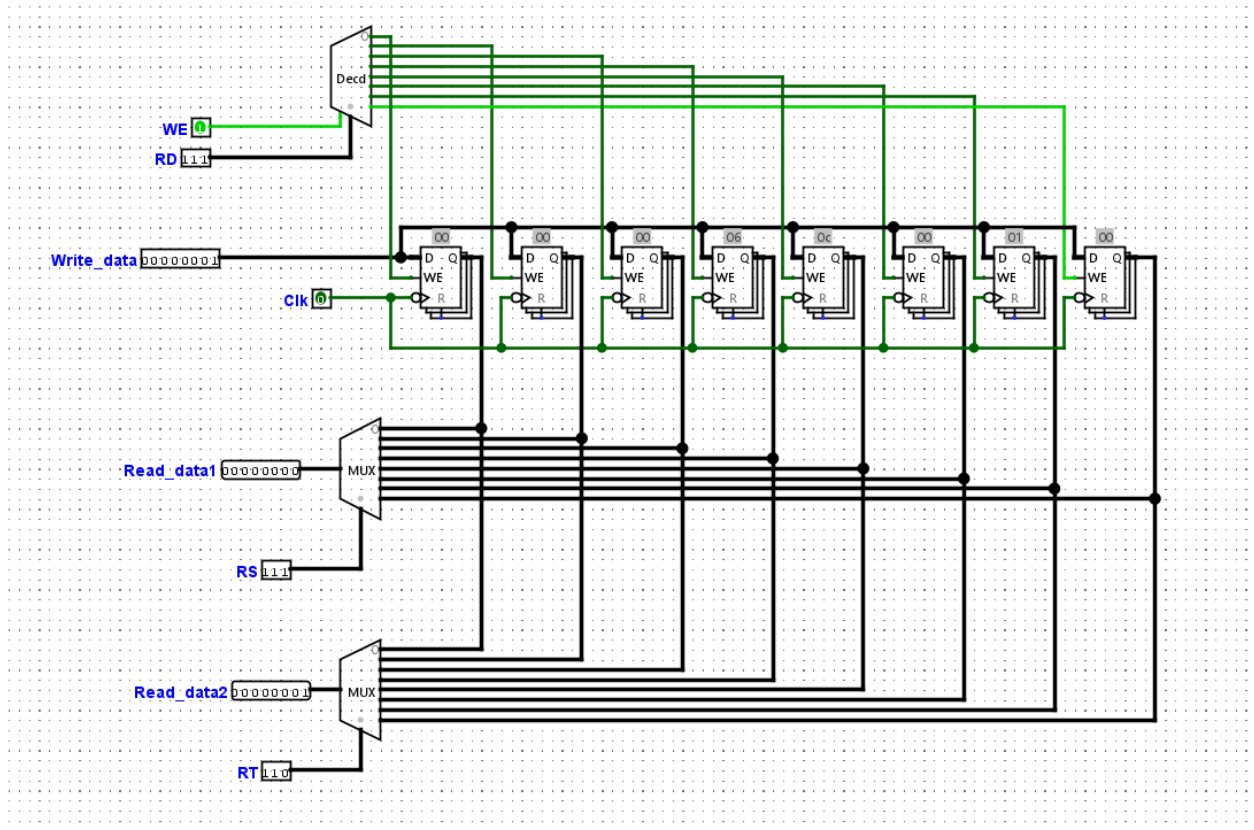
rs: 000 ($R0)

rt: 000 ($R0)

rd: 100 ($R4)

immediate: 0000 1100

The instruction loads 12 to register 4



4-)

ADD R7,R7,R6

Binary: 0000 0001 1111 0111 0000 0000

Hex: 01F700

opcode: 0000

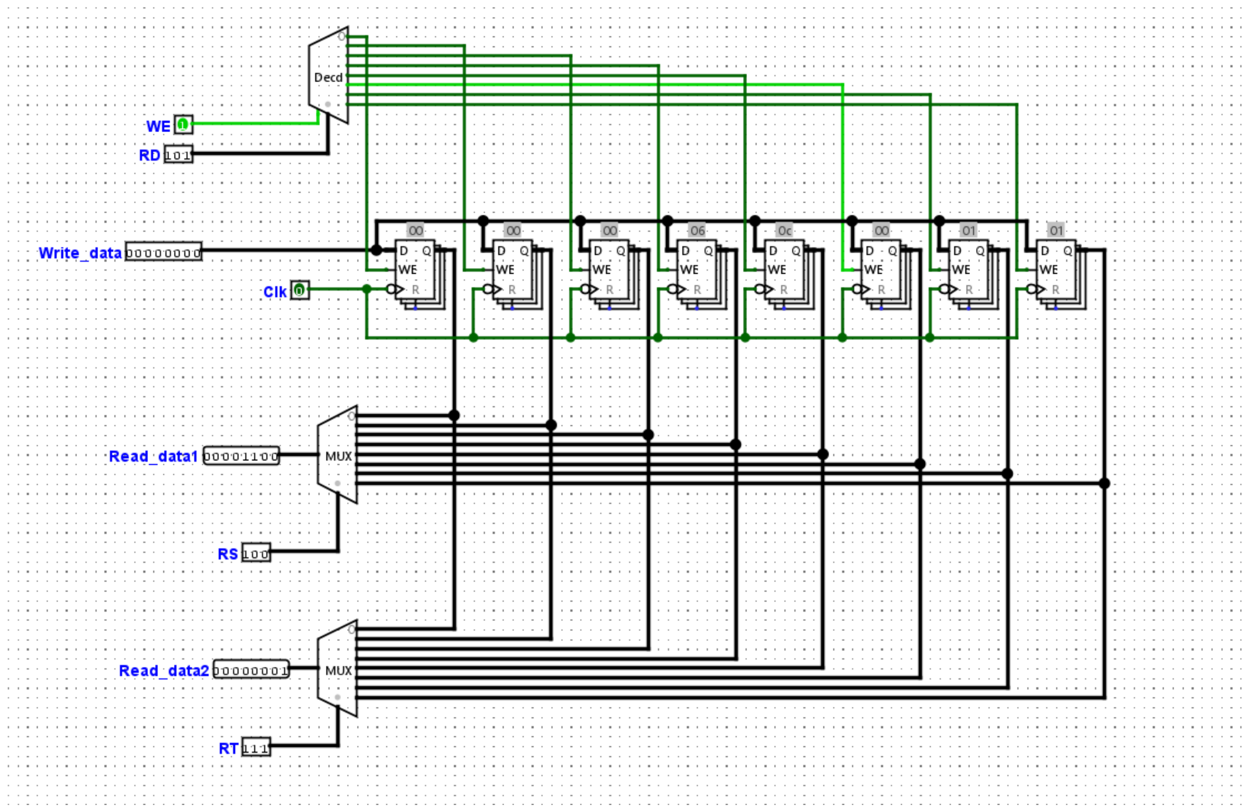
rs: 111 ($R7)

rt: 110 ($R6)

rd: 111 ($R7)

immediate: 0000 0000

The instruction adds the values in register 7 and register 6 and loads the result into register 7.



5-)

SLT R5,R4,R7

Binary: 0001 1111 0011 1101 0000 0000

Hex: 1F3D00

opcode: 1111

rs: 100 ($R4)

rt: 111 ($R7)

rd: 101 ($R5)

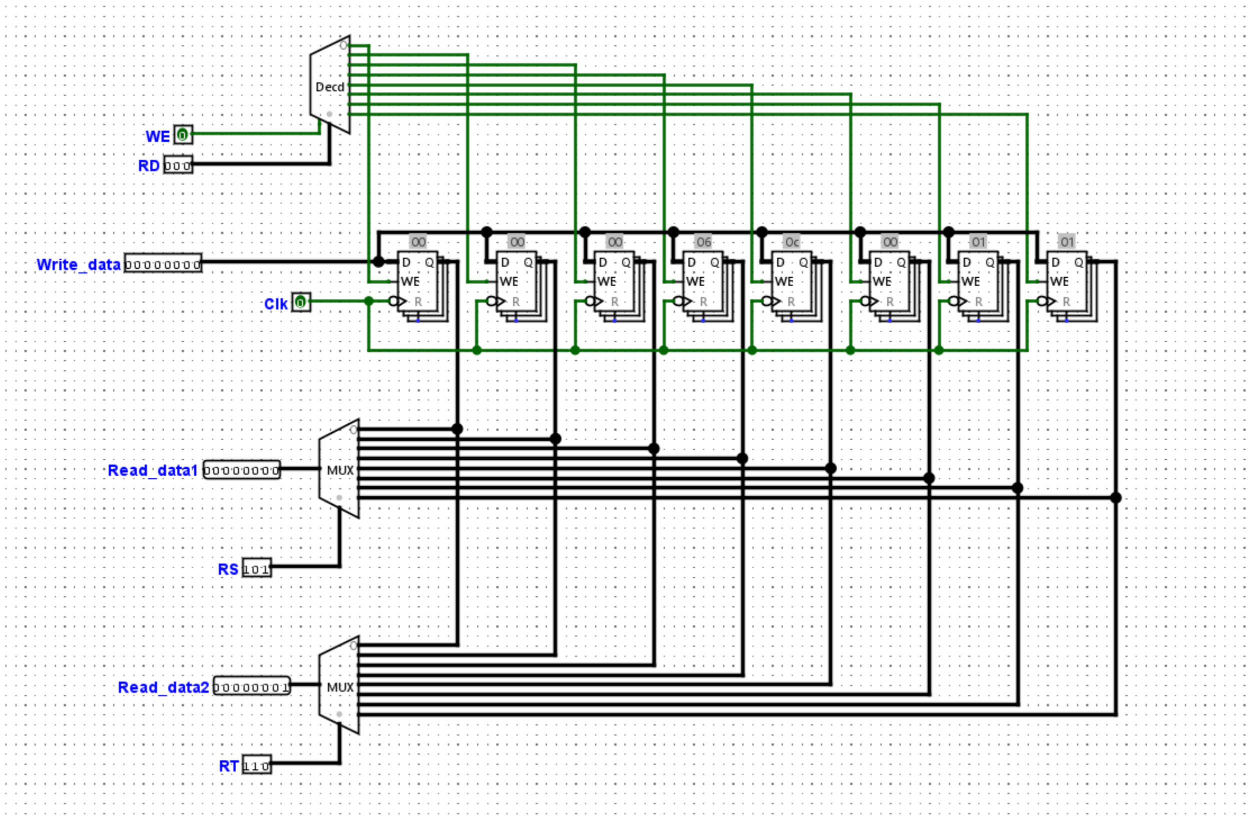
immediate: 0000 0000

if the value in $R4 is less than the value in $R7 then the instruction puts 1 in $R5. Else it puts 0 in $R5.

$R4 = 0x0c

$R5 = 0x00

So, $R5 will be 0.



6-)

BEQ R6,R5, L1

Binary: 0000 1011 0111 0000 1111 1101

Hex: 0B70FD

opcode: 0101

rs: 101 ($R5)

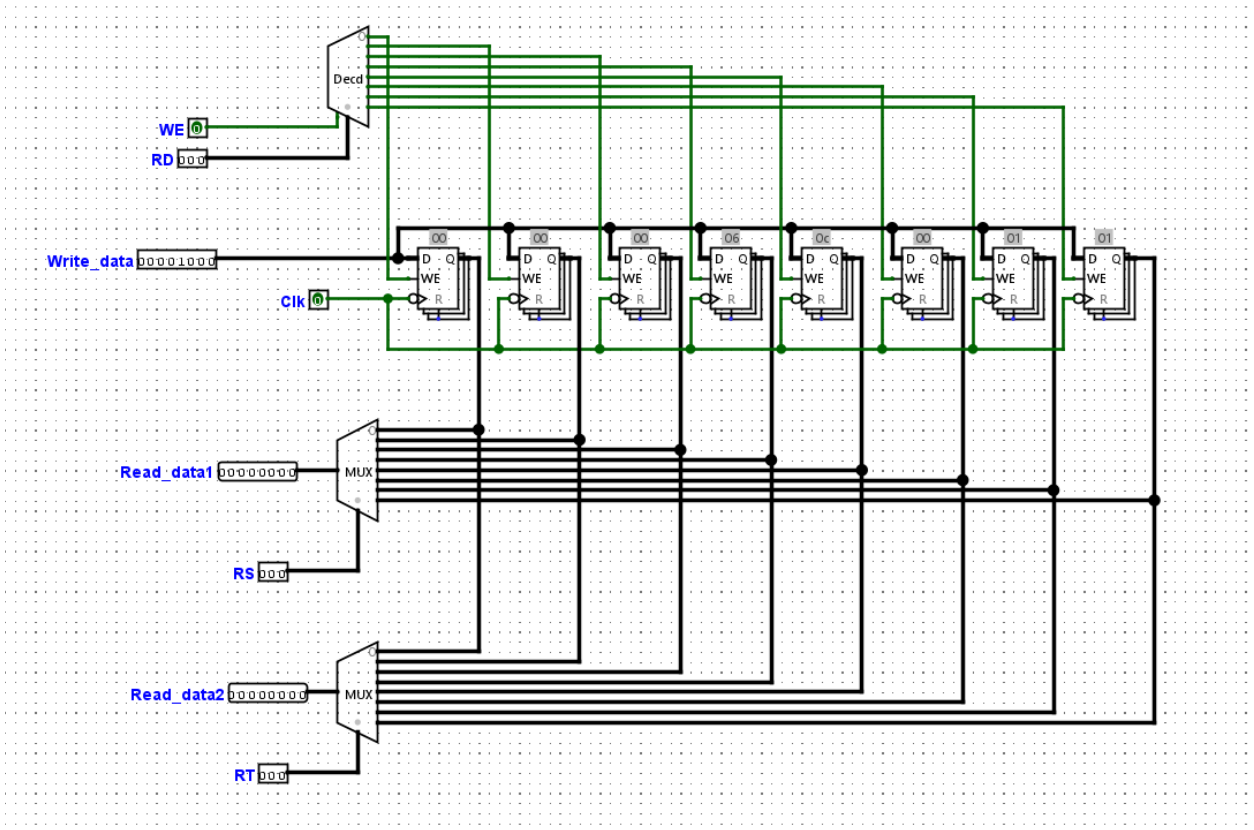
rt: 110 ($R6)

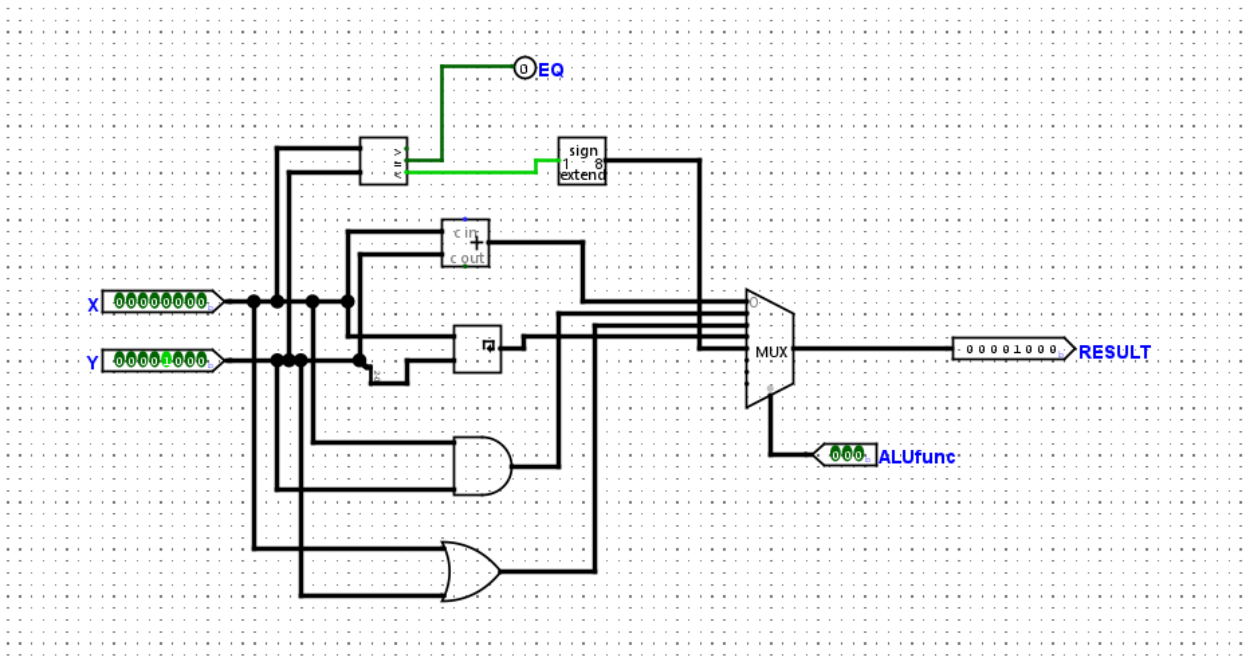
rd: 000 ($R0)

immediate: 1111 1101

if the value in $R6 is equal the value in $R5 then we should go address of L1 which is 3. If $R6 == $R5, then the current address is 5, PC = 5. In branch operations, we add next address and immediate value to find the branch address. For example here, if $R6 == $R5 then (5+1) – 3 = 3 and this is the address of L1. So, immediate should be -3. If they are not equal, then PC continues to increment 1.

In our code, $R5 != $R6. So, the EQ flag is zero, and nothing will change in the register file.





7-)

J END

Binary: 0001 0110 0000 0000 0000 1000

Hex: 160008

opcode: 1011

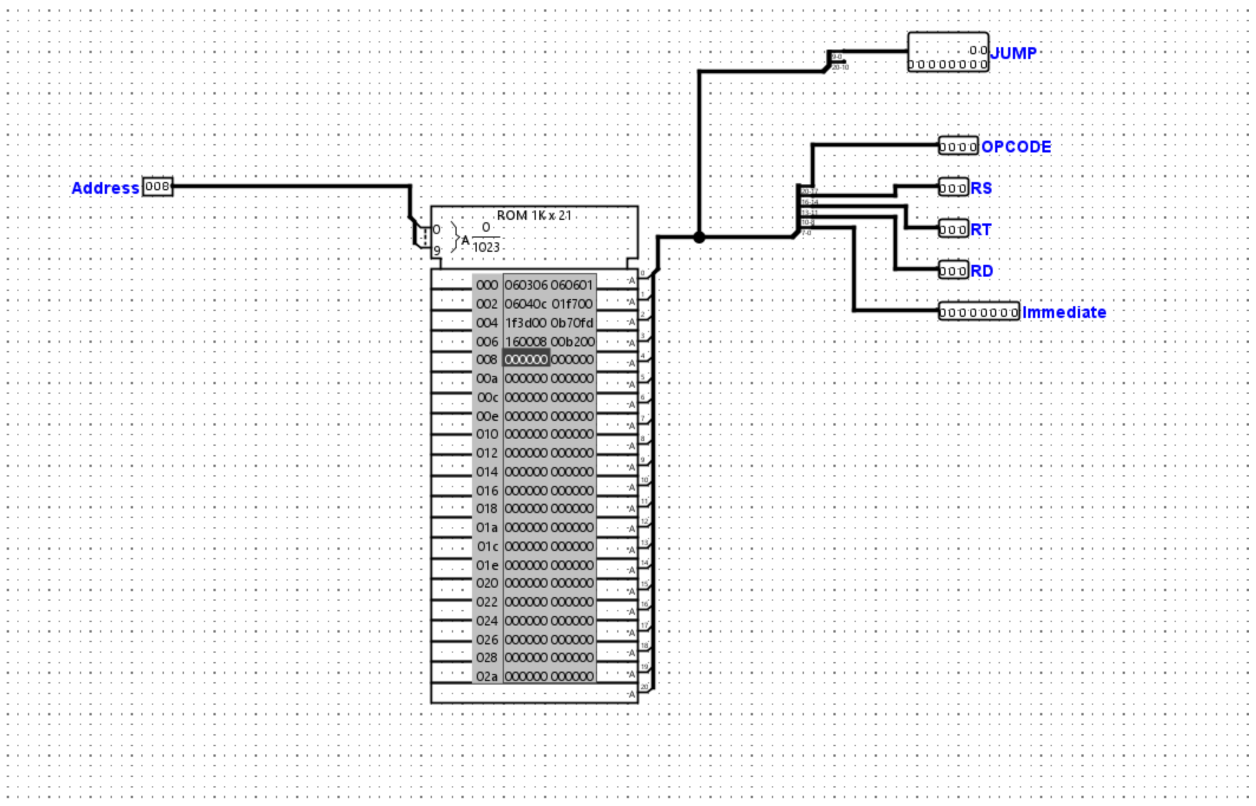
rs: 000 ($R0)

rt: 000 ($R0)

rd: 0(00) ($R0)

JUMP: (00) 0000 1000

In Jump operation, we directly assign PC value to JUMP value which should be the address of END = 8.



```
000 060306 060601
002 06040c 01f700
004 1f3d00 0b70fd
006 160008 00b200
008 000000 000000
00a 000000 000000
00c 000000 000000
00e 000000 000000
010 000000 000000
012 000000 000000
014 000000 000000
016 000000 000000
018 000000 000000
01a 000000 000000
01c 000000 000000
01e 000000 000000
020 000000 000000
022 000000 000000
024 000000 000000
026 000000 000000
028 000000 000000
02a 000000 000000
```

8-)

ADD R2,R2,R6

Binary: 0000 0000 1011 0010 0000 0000

Hex: 00B200

opcode: 0000

rs: 010 ($R2)

rt: 110 ($R6)

rd: 010 ($R2)

immediate: 0000 0000

The instruction adds the values in register 2 and register 6 and loads the result into register 2.

This instruction will not be executed because the jump instruction skips this instruction.