MIDDLE EAST TECHNICAL UNIVERSITY

NORTHERN CYPRUS CAMPUS

Computer Engineering Program

CNG 331

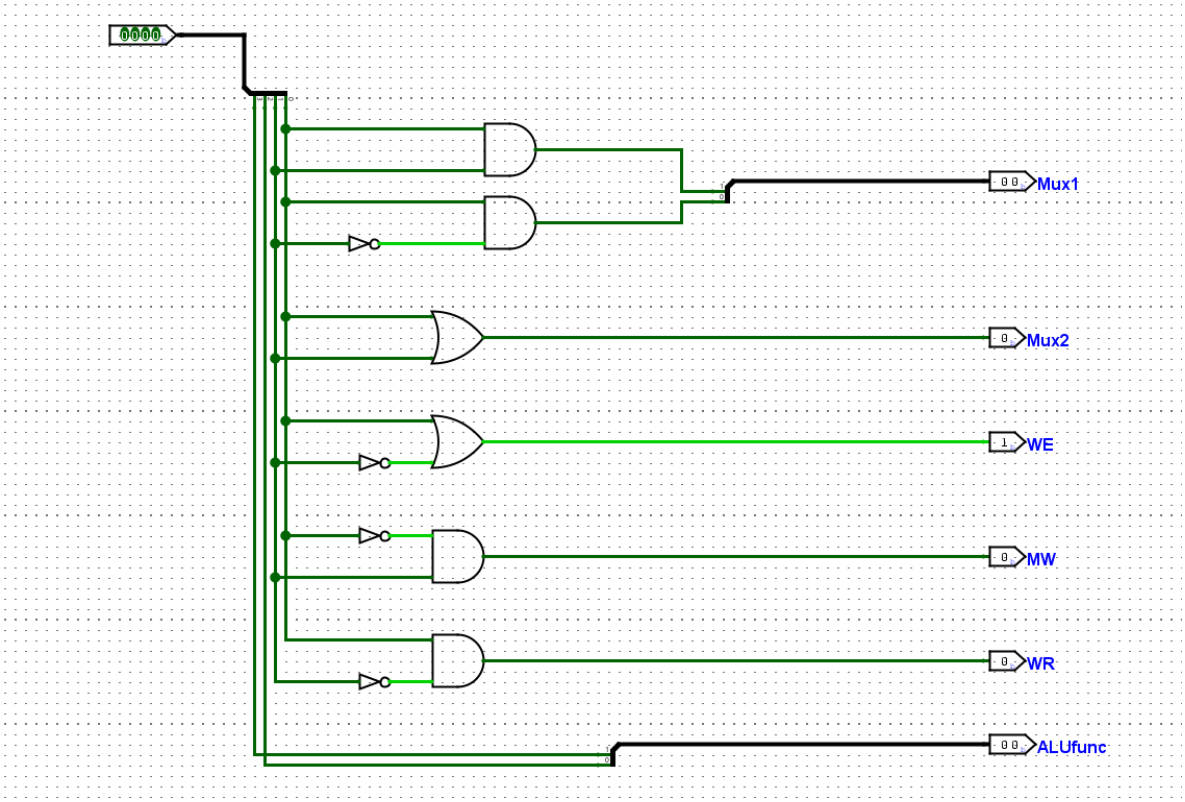
Term Project Part #2

Name: Oğuzhan Alpertürk

ID Number: 2315752

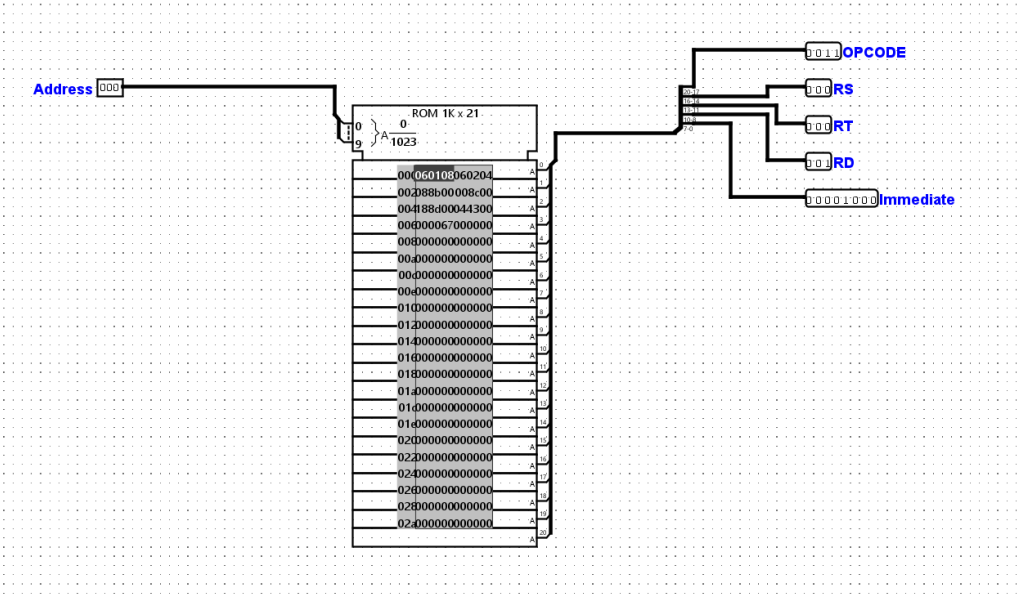
Control_unit:

Control Unit controls the circuit components by sending necessary signals according to the opcode.



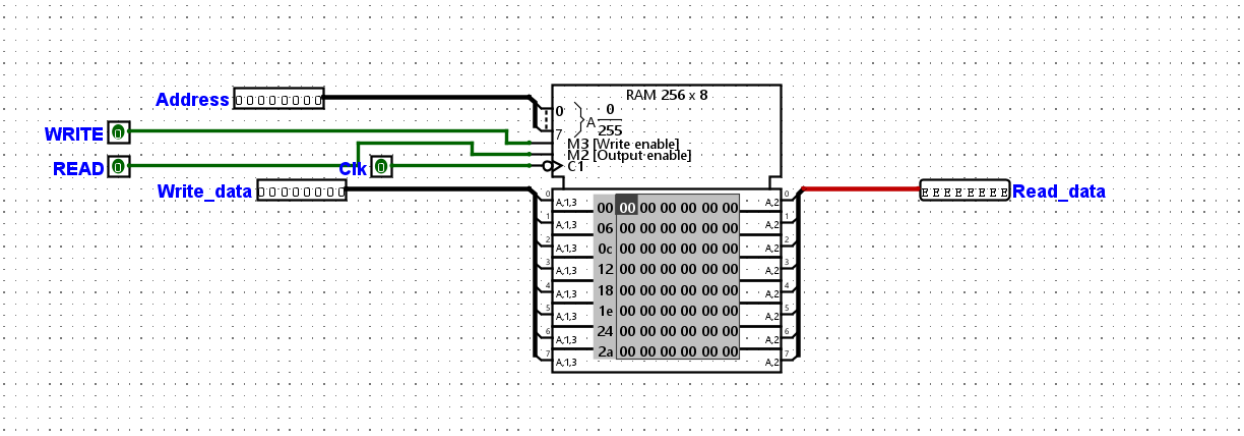
Instruction Memory (ROM):

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.



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.

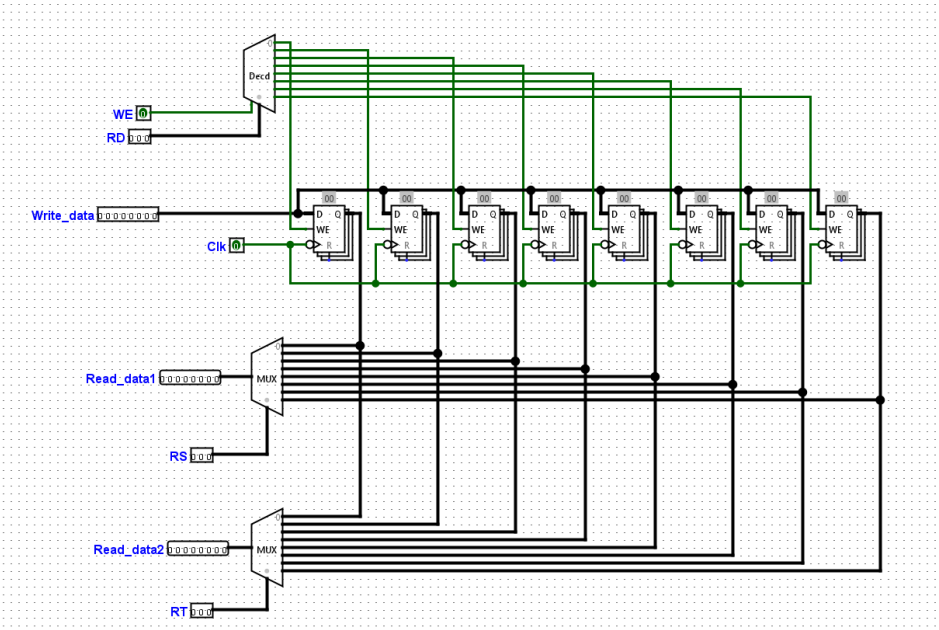


Difference between RAM and ROM:

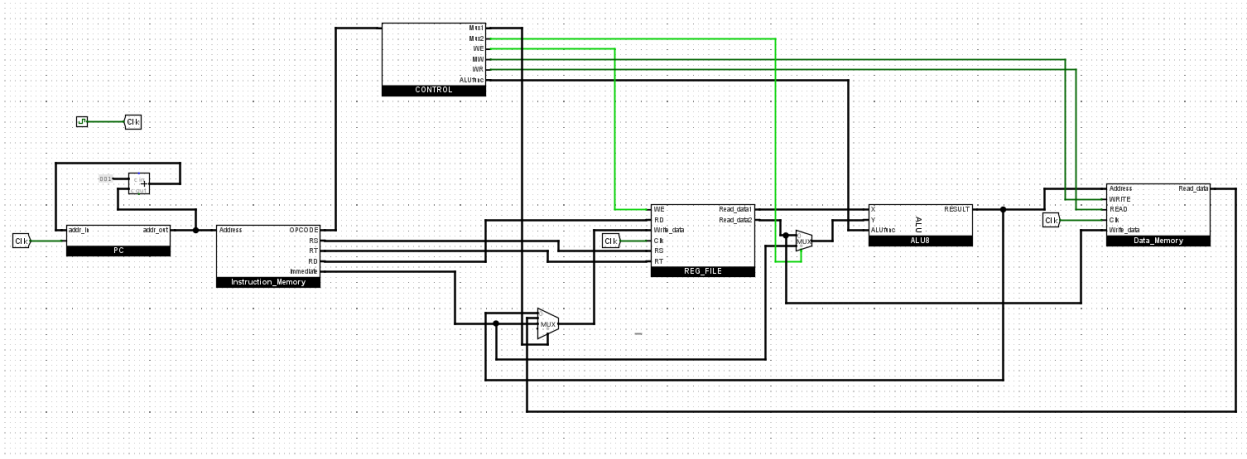
RAM is a temporary memory unit. ROM is a non-volatile memory unit. That is, when you turn off your device, data in RAM is deleted, but data in ROM is not. While RAM has the function of reading and writing data, ROM offers only the function of reading data. RAM is very fast, while ROM is comparatively slower. ROM has a lower capacity, while RAM has a higher capacity. Data stored in RAM is easy to access, while data in ROM is more difficult.

Register File:

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



Whole Schematic:



Test Instructions:

Assumption for register addresses and names :

000 -> $t0

001 -> $t1

010 -> $t2

011 -> $t3

100 -> $t4

101 -> $t5

110 -> $t6

111 -> $t7

1-)

060108 - > 0000 0110 0000 0001 0000 1000

opcode: 0011

rs: 000

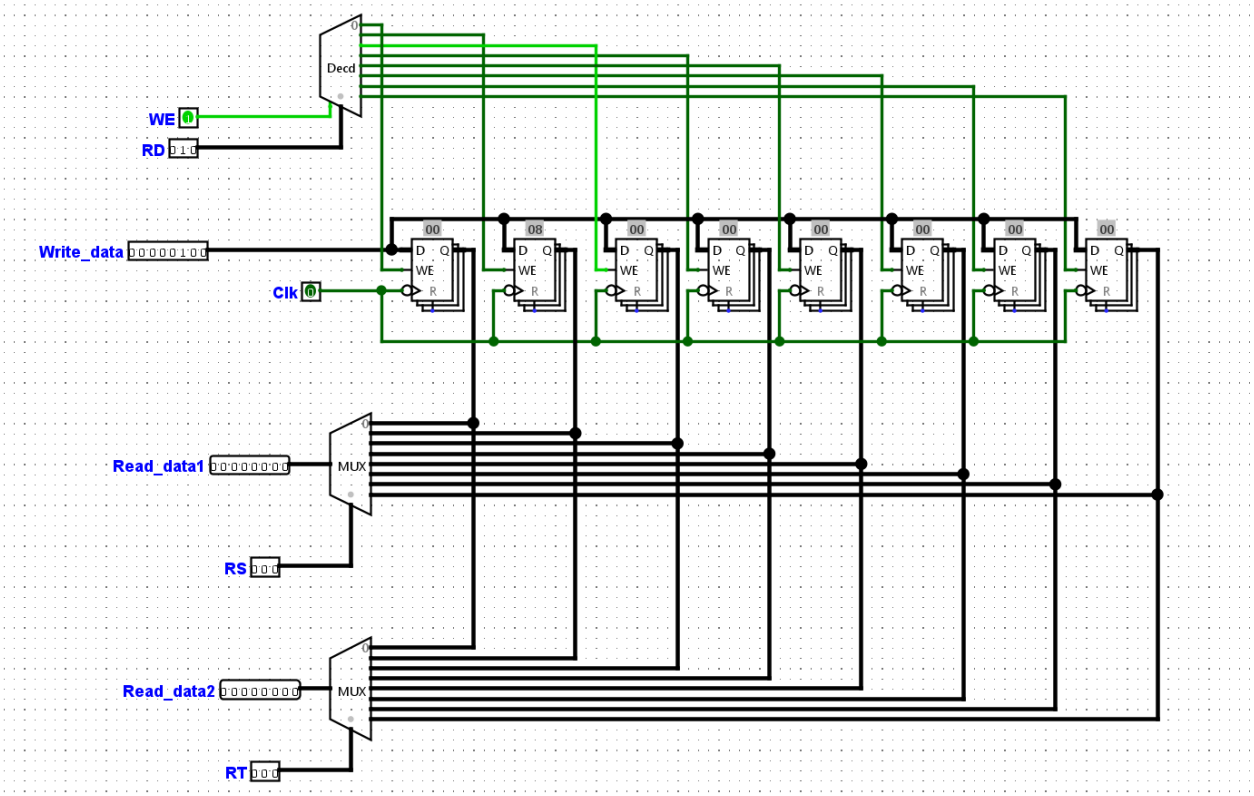
rt: 000

rd: 001

immediate: 0000 1000

LI $t1, 0x08

The instruction loads 0x08 to register 1.



2-)

060204 - > 0000 0110 0000 0010 0000 0100

opcode: 0011

rs: 000

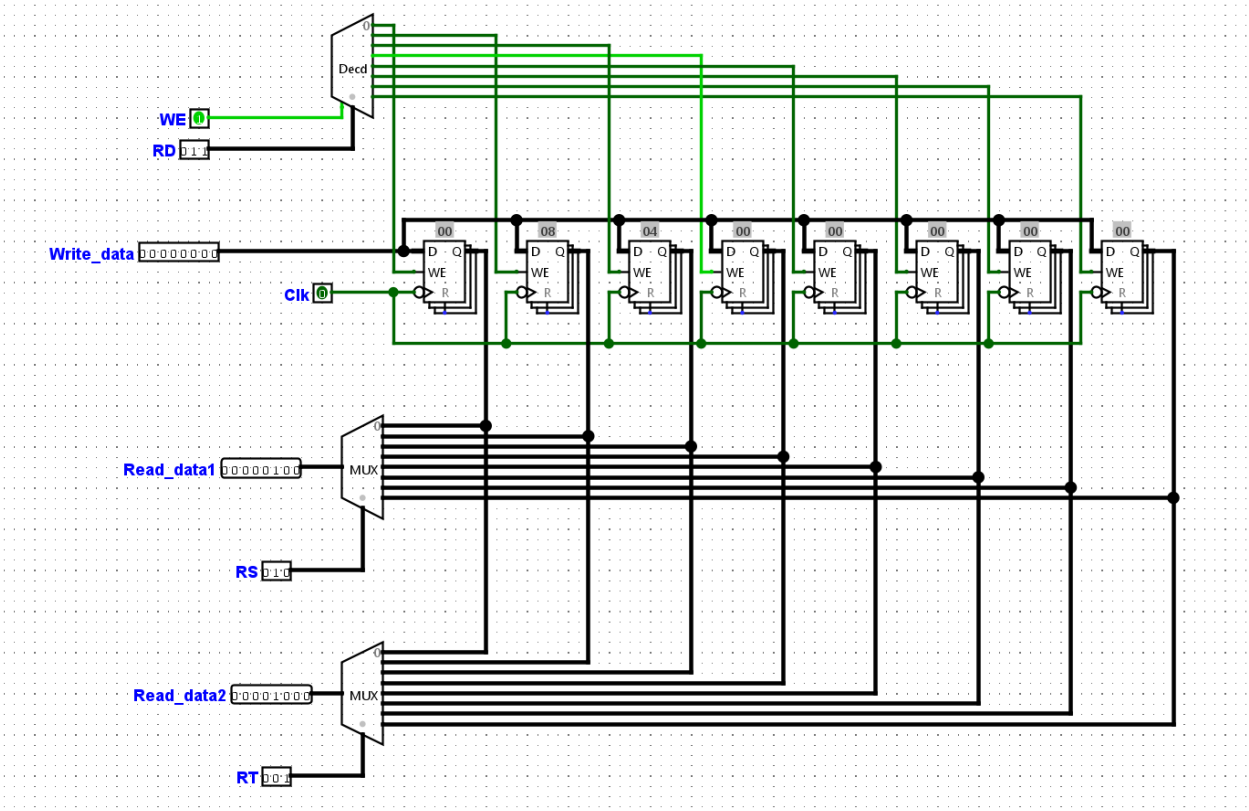
rt: 000

rd: 010

immediate: 0000 0100

LI $t2, 0x04

The instruction loads 0x04 to register 2.



3-)

088b00 - > 0000 1000 1000 1011 0000 0000

opcode: 0100

rs: 010

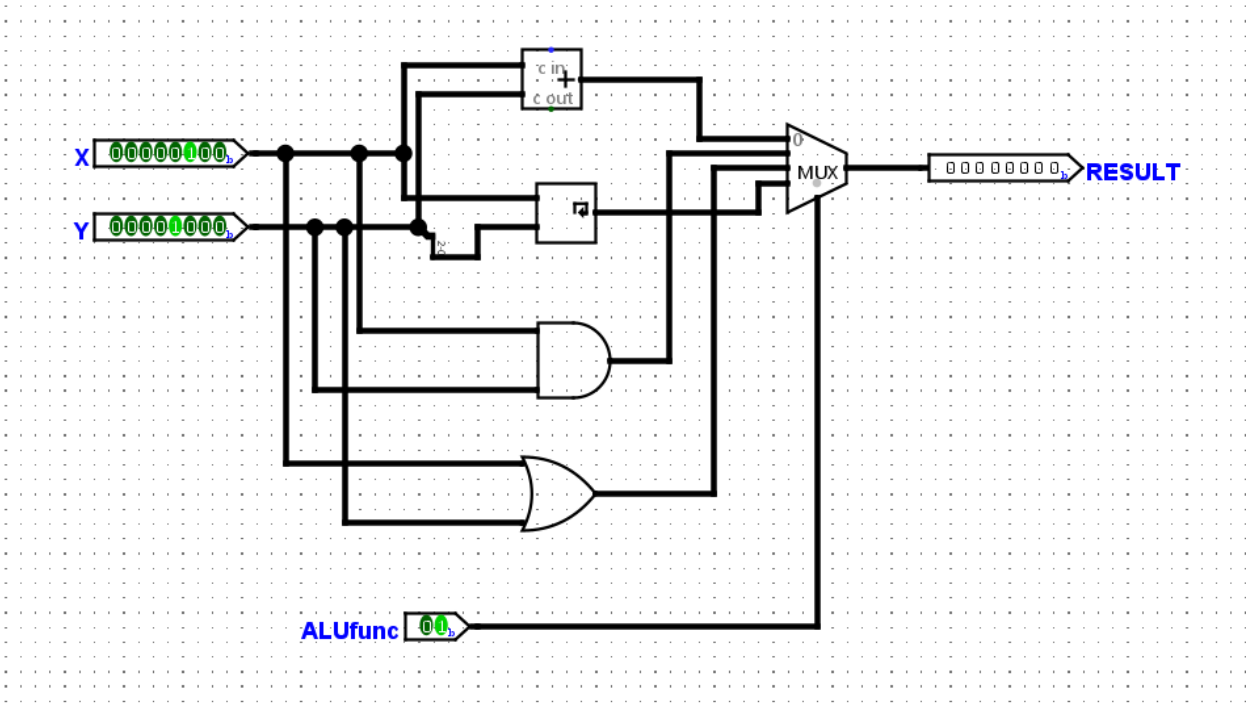
rt: 001

rd: 011

immediate: 0000 0000

AND $t3, $t2, $t1

The instruction makes AND operation with r1 register value and r2 register value and writes the result into the r3 register.



4-)

008c00 - > 0000 0000 1000 1100 0000 0000

opcode: 0000

rs: 010

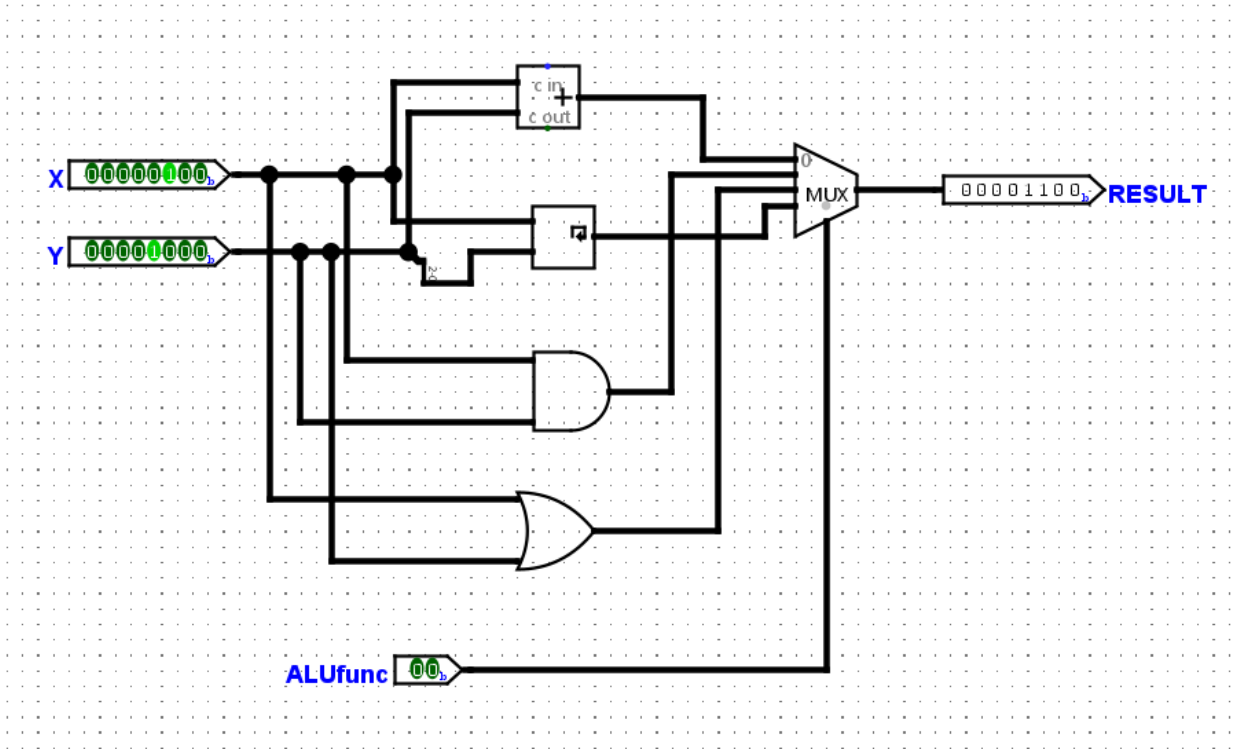
rt: 001

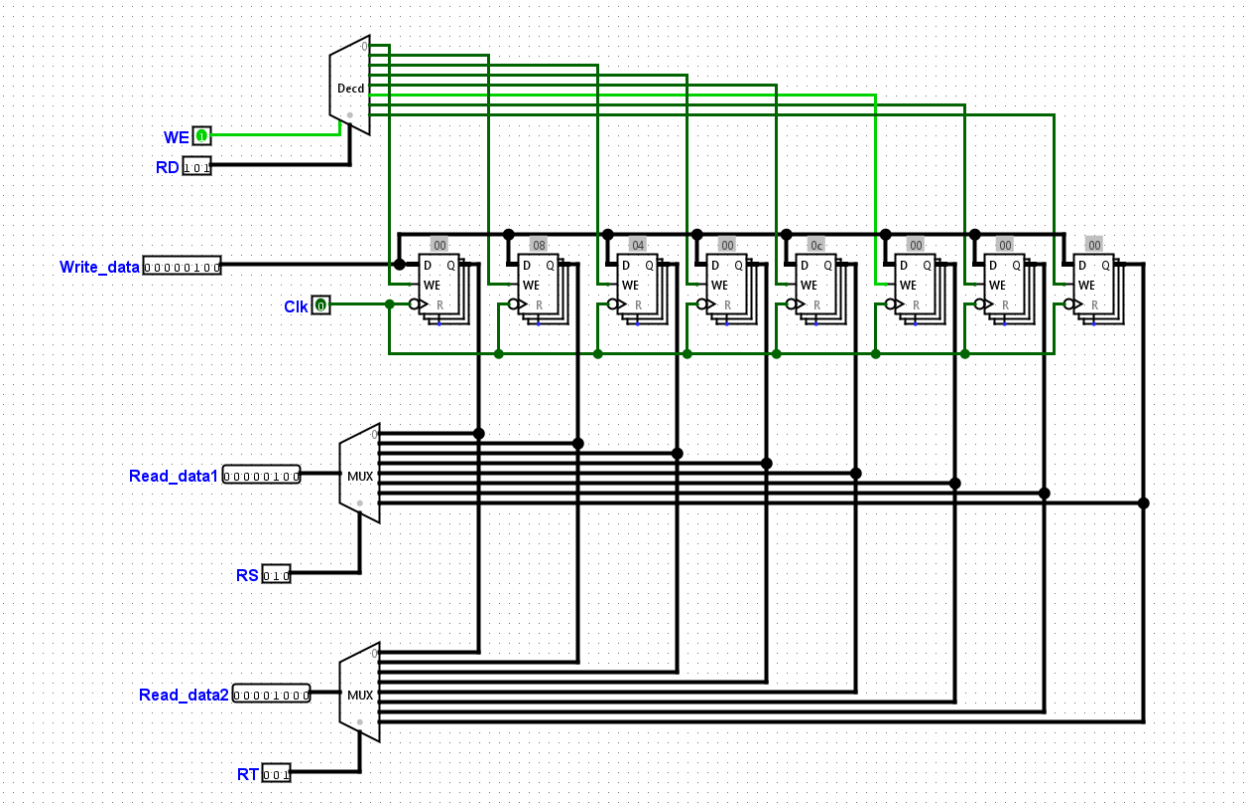
rd: 100

immediate: 0000 0000

ADD $t4, $t2, $t1

The instruction adds the values in register 2 and register 1 and writes the result into register 4.





5-)

188d00 - > 0001 1000 1000 1101 0000 0000

opcode: 1100

rs: 010

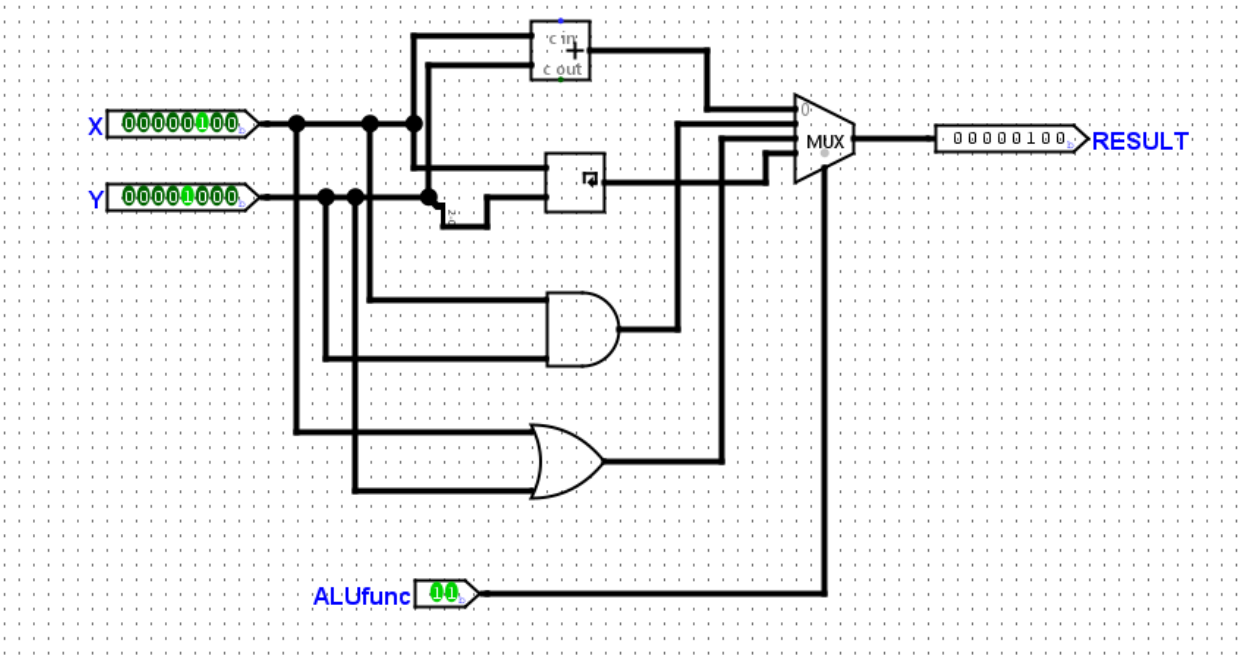
rt: 001

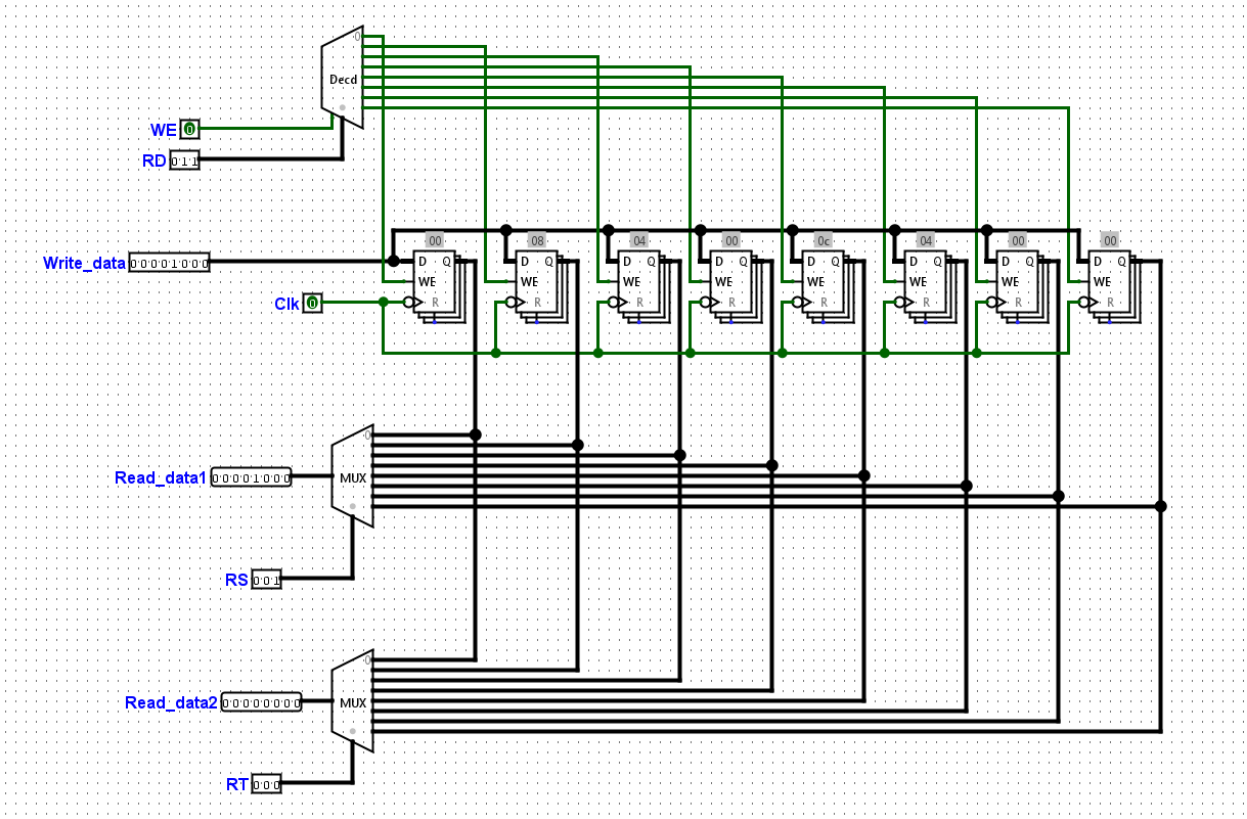
rd: 101

immediate: 0000 0000

ROTL $t5, $t2, $t1

The instruction rotates the number in register 2 for the value in register 1 times and writes the result into register 5.


X
00000100
Y
00001000
c in
c out
MUX
00000100
RESULT
ALUfunc
00


Decd
WE
0
RD
Write_data
00001000
Clk
0
00
08
04
00
0c
04
00
00
Read_data1
00001000
MUX
RS
001
Read_data2
00000000
MUX
RT
000

6-)

045000 - > 0000 0100 0101 0000 0000

opcode: 0010

rs: 001

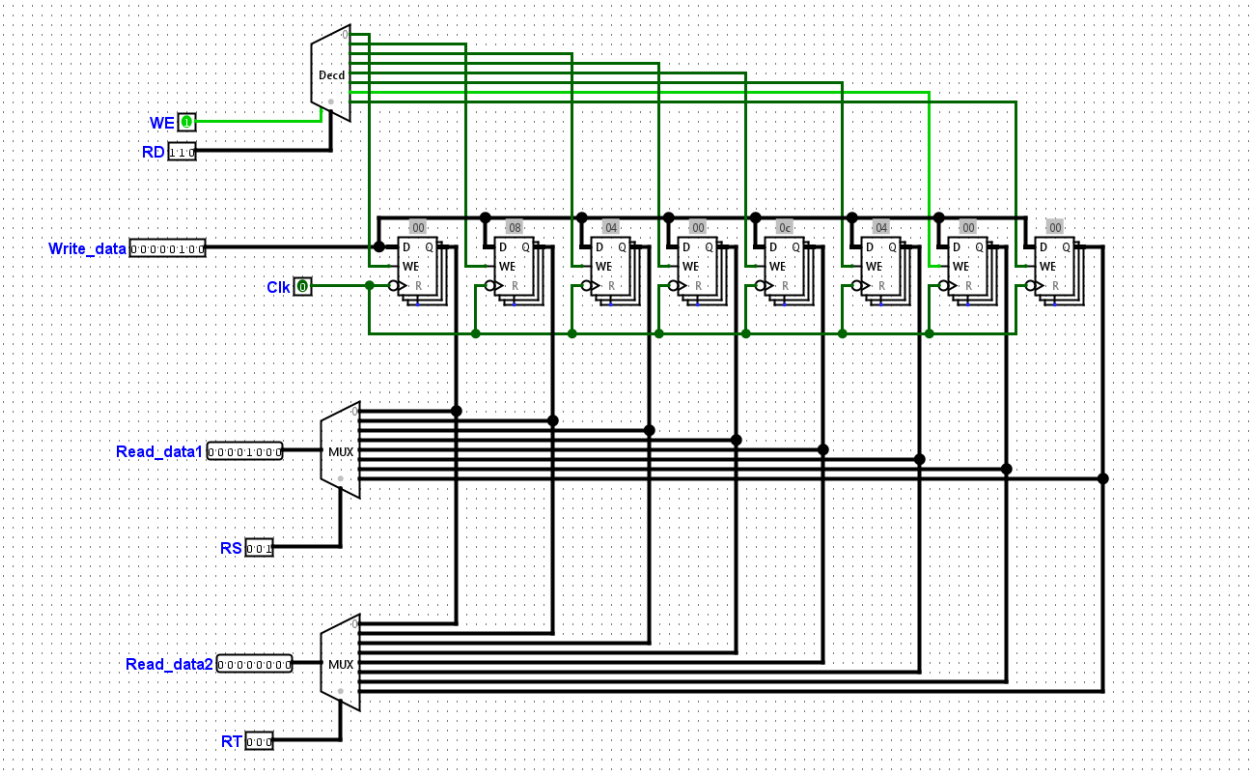
rt: 010

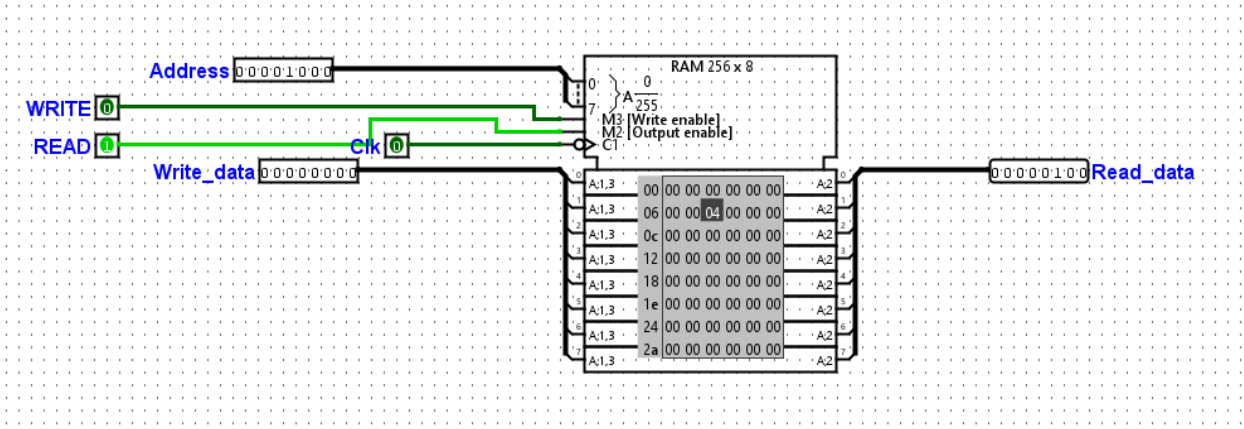
rd: 000

immediate: 0000 0000

ST $t2, 0($t1)

The instruction stores the value in register 2 (0x04) into the address [0x08 + 0x00]





7-)

024600 - > 0000 0010 0100 0110 0000 0000

opcode: 0001

rs: 001

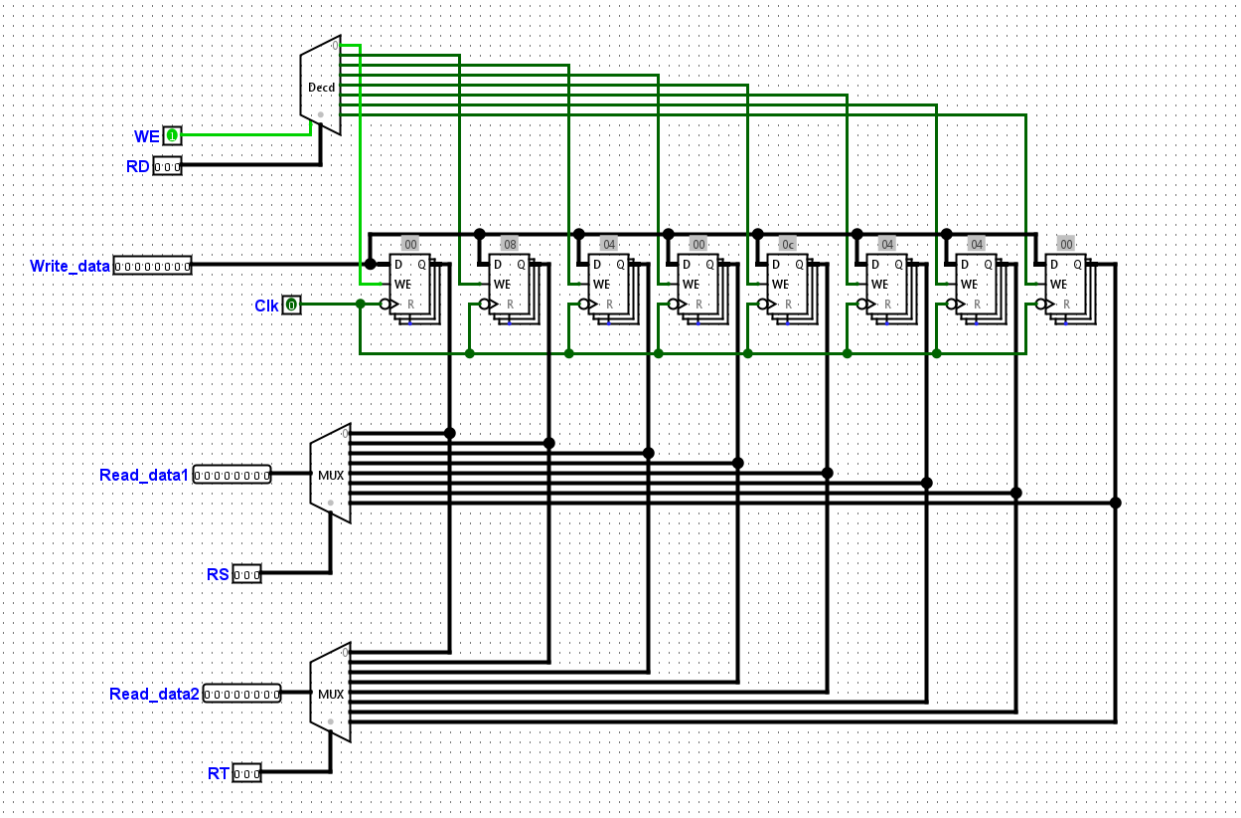
rt: 000

rd: 110

immediate: 0000 0000

LD $t6, 0($t1)

It loads the value in the address 0x08 (0 + register 1 value 0x08 = 0x08) into register 6.


Decd
WE
RD
Write_data
Clk
00
08
04
00
0c
04
04
00
D
Q
WE
R
Read_data1
MUX
RS
Read_data2
MUX
RT