KUZEY KIBRIS KAMPÜSÜ
ODTÜ METU
NORTHERN CYPRUS CAMPUS

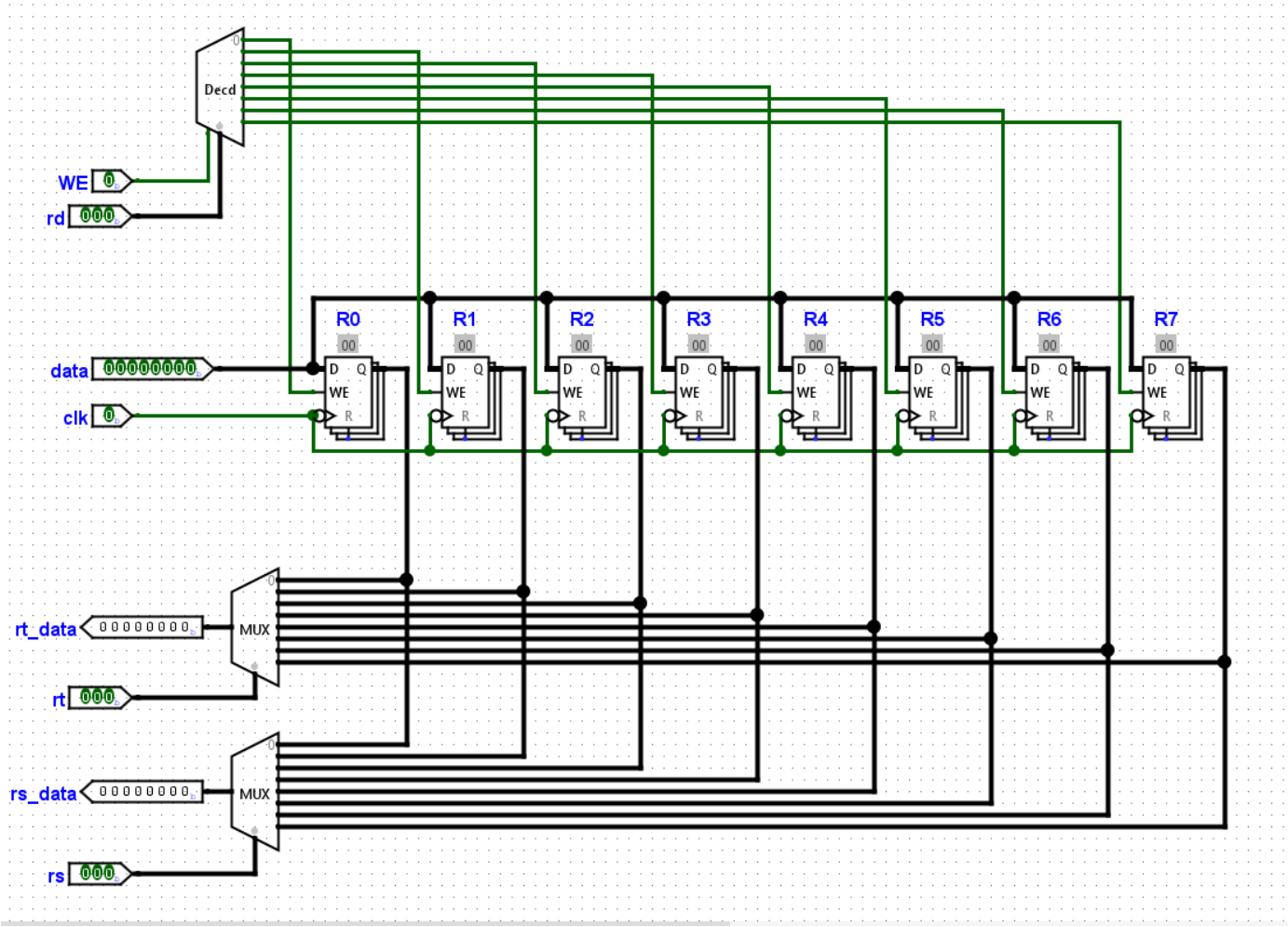
CNG 331 Project 2 Report

Name: Muhammad Somaan
ID: 2528404

Part 1)

Register File:

Schematic:



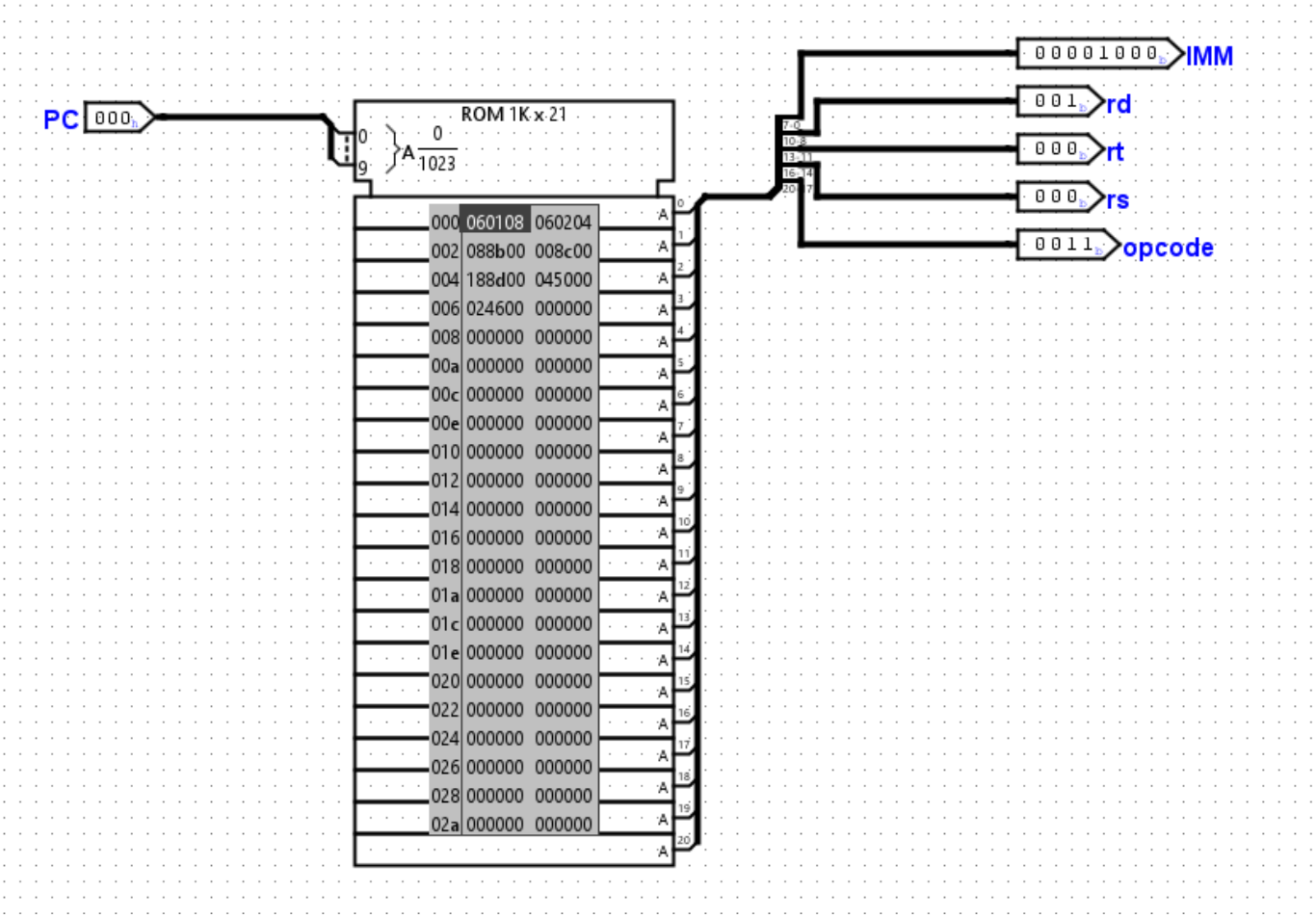
Comments: Register R0 to R7 are individual registers, all share the same clock and rt_data and rs_data is the output depending on the select rt and rs. To write data to registers, we provide WE input as 1 and rd select to specify the register to store the input data in.

Do we need enable for mux? No we do not since we want mux to work all the time.

Part 2)

Instruction Memory:

Schematic:



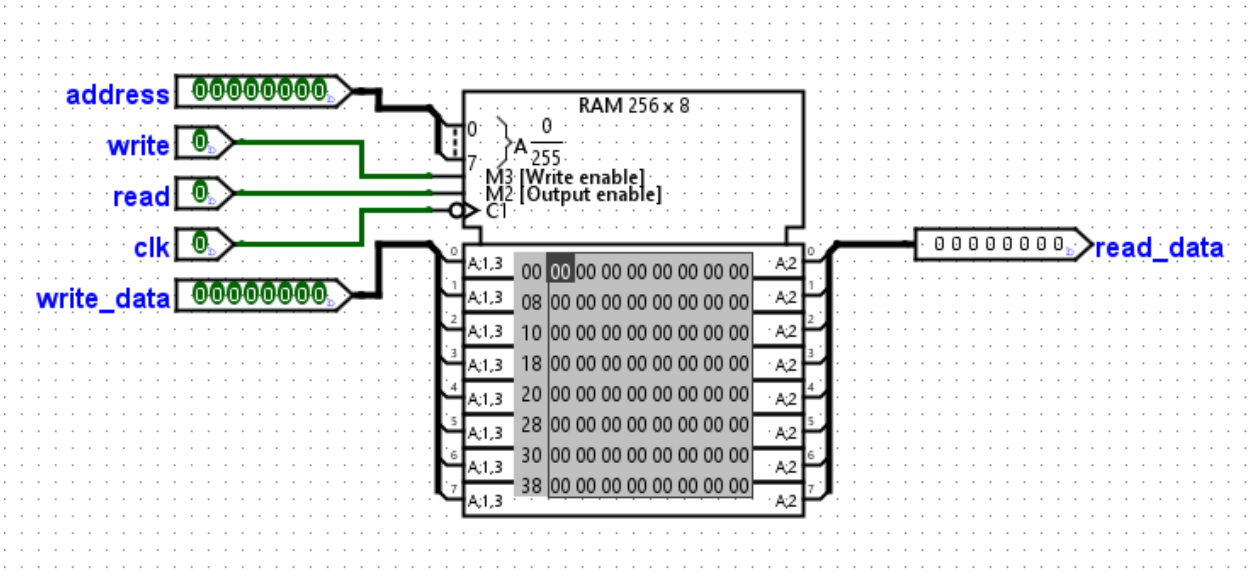
ROM Properties:

ROM "Instruction_Memory"	
FPGA supported:	Supported
Address Bit Width	10
Data Bit Width	21
Line size	Single
Allow misaligned?	No
Contents	(click to edit)
Label	Instruction_Memory
Label Font	SansSerif Bold 16
Label Visible	No
Appearance	Logisim-Evolution

Comments: ROM stores a 21 bit data, since our instruction length is 21 bit. The address bit width is 10 bits, allowing for 1024 instructions to be stored. ROM takes PC input and fetches that instruction and outputs the 21 bit data, into 5 parts. Opcode, Rs, Rt, Rd and Immediate value IMM.

Data Memory:

Schematic:



RAM Properties:

RAM "My_Ram"	
FPGA supported:	Not supported
Address Bit Width	8
Data Bit Width	8
Enables:	Use byte enables
Ram type	volatile
Use clear pin	No
Trigger	Falling Edge
Asynchronous read:	Yes
Data bus implementation	Separate data bus for read an...
Label	My_Ram
Label Font	SansSerif Bold 16
Label Visible	No
Appearance	Logisim-Evolution

Comments: RAM stores 8 bit data with 8 bit address. My data memory takes address and data as input and then takes write or read flags as inputs to determine if data is to be read from the RAM or it should be stored. I provide clock clk as input to keep all clocks in CPU the same. Read_data outputs the data in the address provided if it is a read operation.
Does the RAM store your values as big endian or little endian? It stores it in hexadecimal big-endian.

RAM vs ROM:

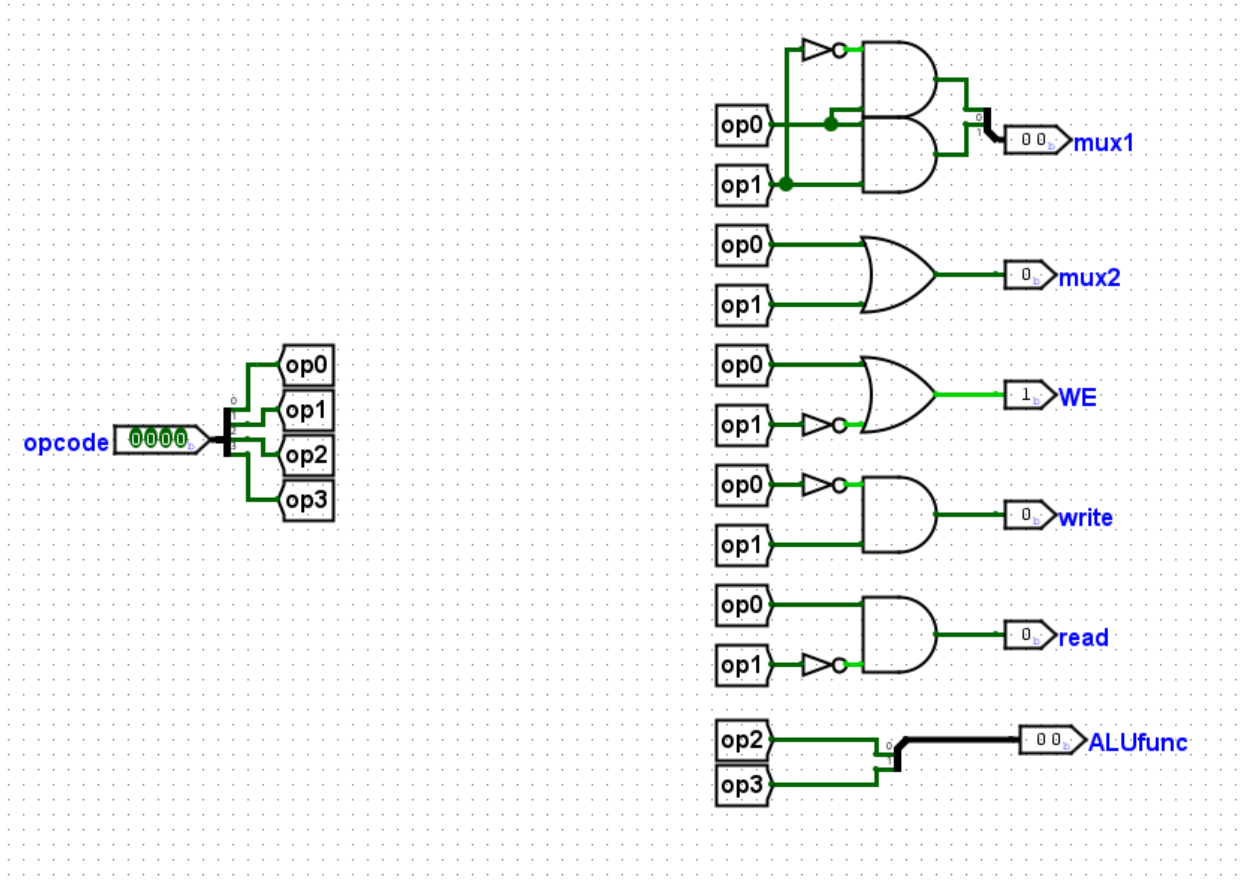
RAM: RAM is random access memory, which is a volatile memory meaning that all the contents stored in the memory are lost once the power is lost. This means that whatever we store in RAM is lost if the CPU is turned off. RAM is useful for storing data temporarily, for example, it used to store the data for calculations in a program.

ROM: ROM is read only memory, which is a non-volatile memory, meaning that no data is lost even if power is lost. This is especially useful for storing boot up sequence for computers, and in our case it is used to store the instruction set, since we don't want to lose our instructions. Since it is a read-only memory, data cannot be written to it.

CPU)

Control Unit:

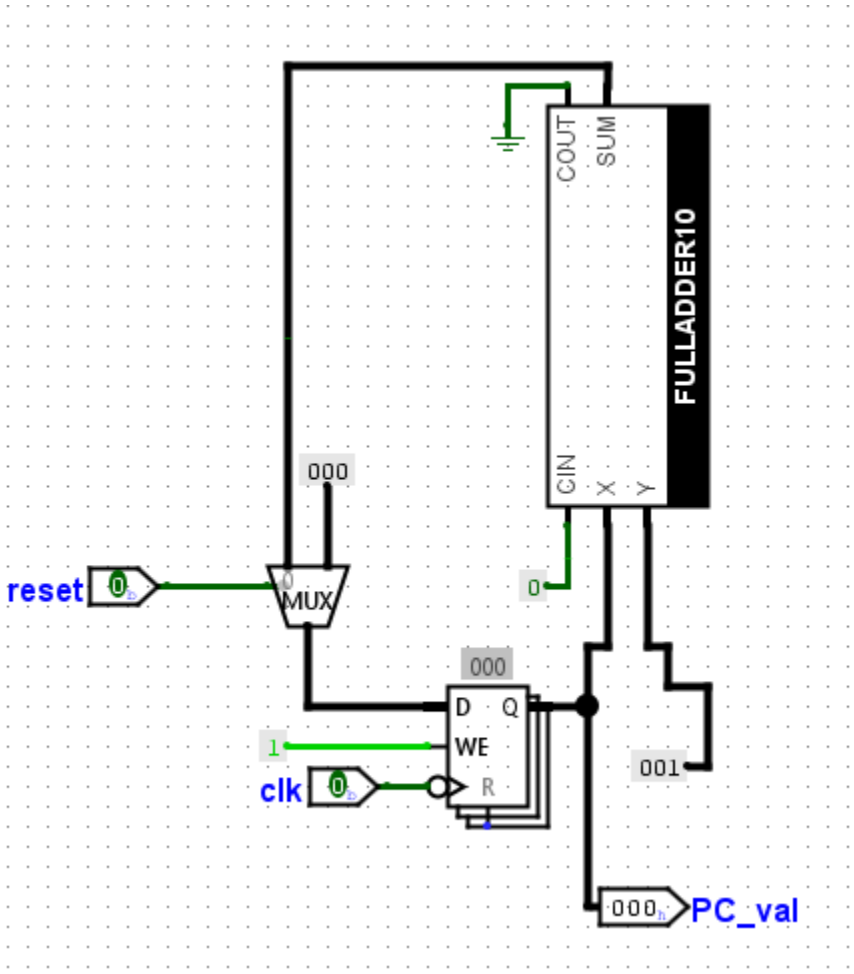
Schematic:



Comments: opcode is taken as input to decide upon different outputs. Mux1 and Mux2 output is the select for mux1 and mux2 in cpu, WE is write enable for register file and then read and write output are flags for data memory, and ALUfunc is the output for deciding the alu function.

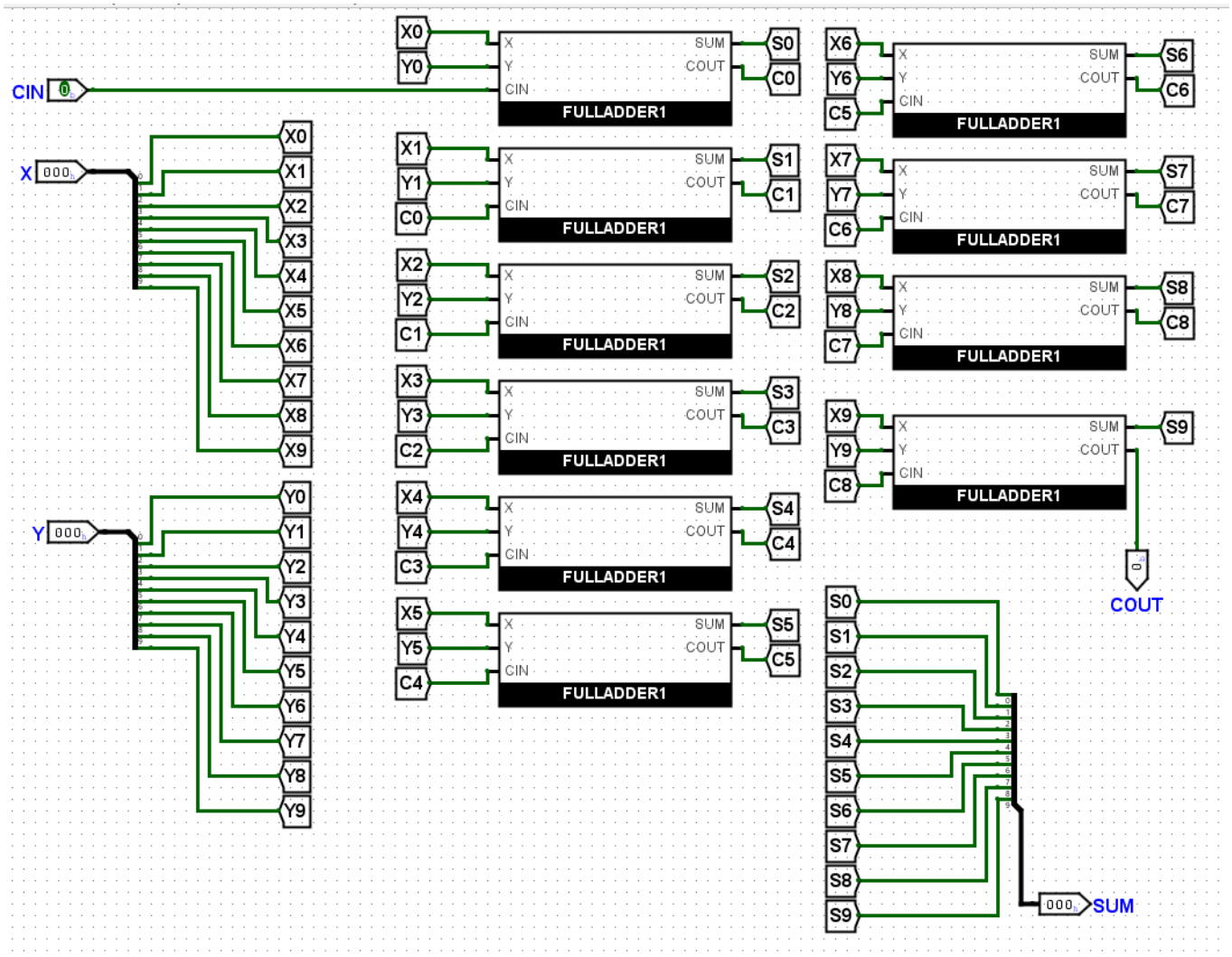
Program Counter:

Schematic:



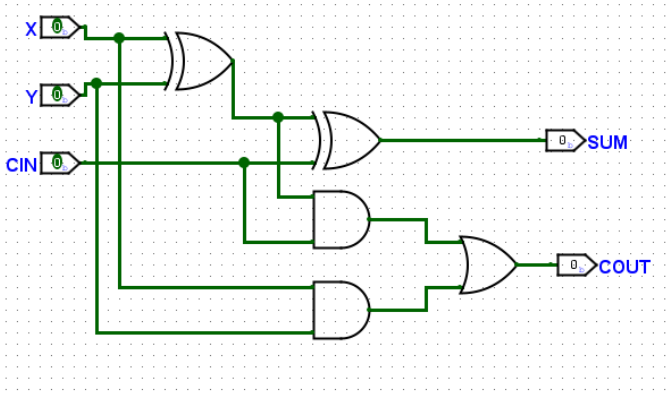
FULLADDER10:

Schematic:



FULLADDER1:

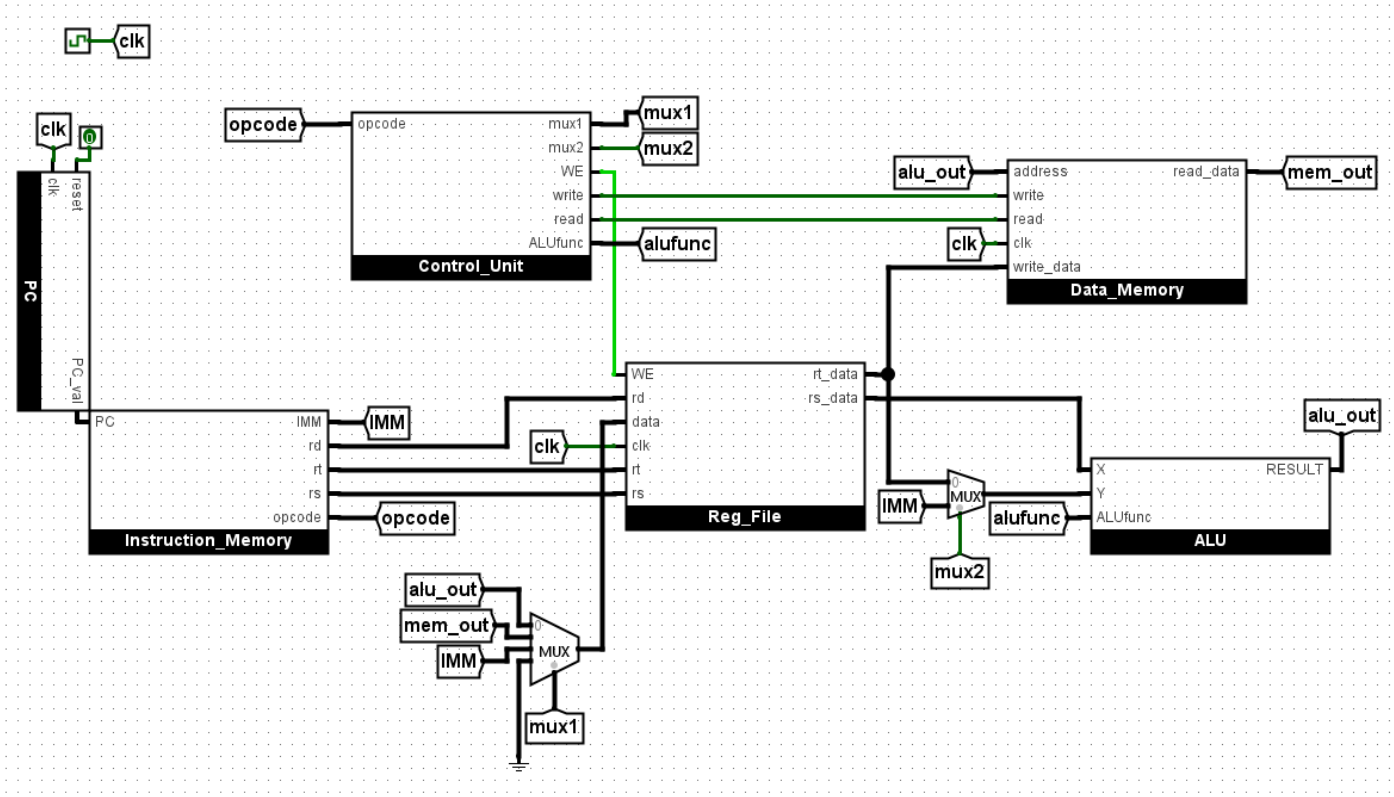
Schematic:



Comments: My program counter takes clk clock as input for that all clocks in CPU are same. Reset input is used to reset the pc. Program Counter is a register which starts from 0 and then for every clock cycle it is incremented by 1. For addition a 10 bit full adder is used. After incrementing it outputs the value to PC_val.

CPU:

Schematic:



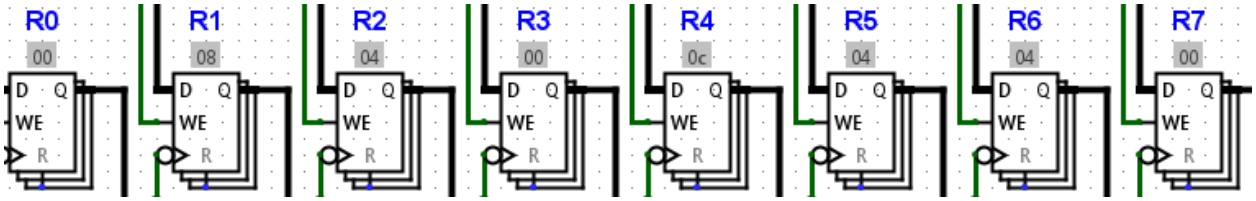
Comments: CPU is designed according to the schematic provided.

TEST)

Assembly Code:

LI R1, 8
LI R2, 4
AND R3, R2, R1
ADD R4, R2, R1
ROTL R5, R2, R1
ST R2, 0(R1)
LD R6, 0(R1)
ADD R0, R0, R0 ….

Register Contents:



Comments: First 8 is loaded in to register 1 then 4 is loaded into register 2. Then and operation is performed on 8 and 4 and result is stored in register 3 which is zero. Then 8 and 4 are added and stored in register 4 with value 12 which is 0c in hexadecimal. Afterwards 4 is rotated left by 8 bits, which is same as not rotating it since the registers hold 8 bit data, the result is 4 again and is stored in register 5. Then 4 is stored in data memory at the address provided by register 1 which is 8 or 00001000 in binary. Then data is loaded into register 6 from data memory at location 8 (00001000) which is 4. After wards it continue to add register 0 to itself and stores in register 0. Since register 0 value is 0, it remains 0.