Logisim 4

Points: 16

In this problem you will be implementing a 4 bit CPU. This means that the size of the data path will be 4 bits big.

Instruction Format

We will be using fixed length instruction formats for this problem. This means that all instructions will be the same size. We will be using 2 different instruction formats for this problem, R-type and I-type. In R-type instructions both operands will come from the registers. In I-type instructions the first operand will come from the register file and the second will be contained in the instruction.

R-Type

Name	Bits	Description
OpCode	12 - 15	Determines what operation should be performed
C	8 - 11	The destination register. The C in the formula C = A OP B
A	4 - 7	The first source register. The A in the formula C = A OP B
B	0-3	The second source register. The B in the formula C = A OP B

I-Type

Name	Bits	Description
OpCode	12 - 15	Determines what operation should be performed
C	8 - 11	The destination register. The C in the formula C = A OP Imm
A	4 - 7	The first source register. The A in the formula C = A OP Imm
Immediate	0-3	The immediate value. The Imm in the formula C = A OP Imm

Instructions

Operation	Encoding(The value in the OpCode field)	Description
HALT	0000	The CPU ceases execution
NOP	0001	Do nothing
LOAD	0010	Reg_C = Immediate
MOVE	0011	$Reg_C = Reg_A$
ANDR	0100	$Reg_C = Reg_A$ AND Reg_B
ANDI	0101	$Reg_C = Reg_A$ AND Immediate
ORR	0110	$Reg_C = Reg_A$ OR Reg_B
ORI	0111	$Reg_C = Reg_A$ OR Immediate
XORR	1000	$Reg_C = Reg_A$ XOR Reg_B
XORI	1001	$Reg_C = Reg_A$ XOR Immediate
NOT	1010	Reg_C = NOT Reg_A
NEGATE	1011	$Reg_C = -Reg_A$
ADDR	1100	$Reg_C = Reg_A + Reg_B$
ADDI	1101	$Reg_C = Reg_A +$ Immediate
SUBR	1110	$Reg_C = Reg_A - Reg_B$
SUBI	1111	$Reg_C = Reg_A$ - Immediate

CPU Components

1. Memory.
    1. We will be using a ROM module to store our instruction. The address bit width will be 5 bits wide and the data bit width will be 16 bits wide
    2. This is already given to you and comes preloaded with the testing program
2. Register File
    1. A collection of 16 registers numbered 0 through 15.
    2. This has already been given to you so you don't have to implement it
3. ALU
    1. This is where you actually execute the instruction
4. Decoder
    1. A bunch of combinational logic that will enable or disable the appropriate control signals in your CPU

Testing

You have also been give a file called CPU_Inputs.py. This contains the program that you can use to create test programs. If you look inside it you will find the solutions for the testing program that is being used. We will check your answers by looking inside your register file and making sure they have the right values.