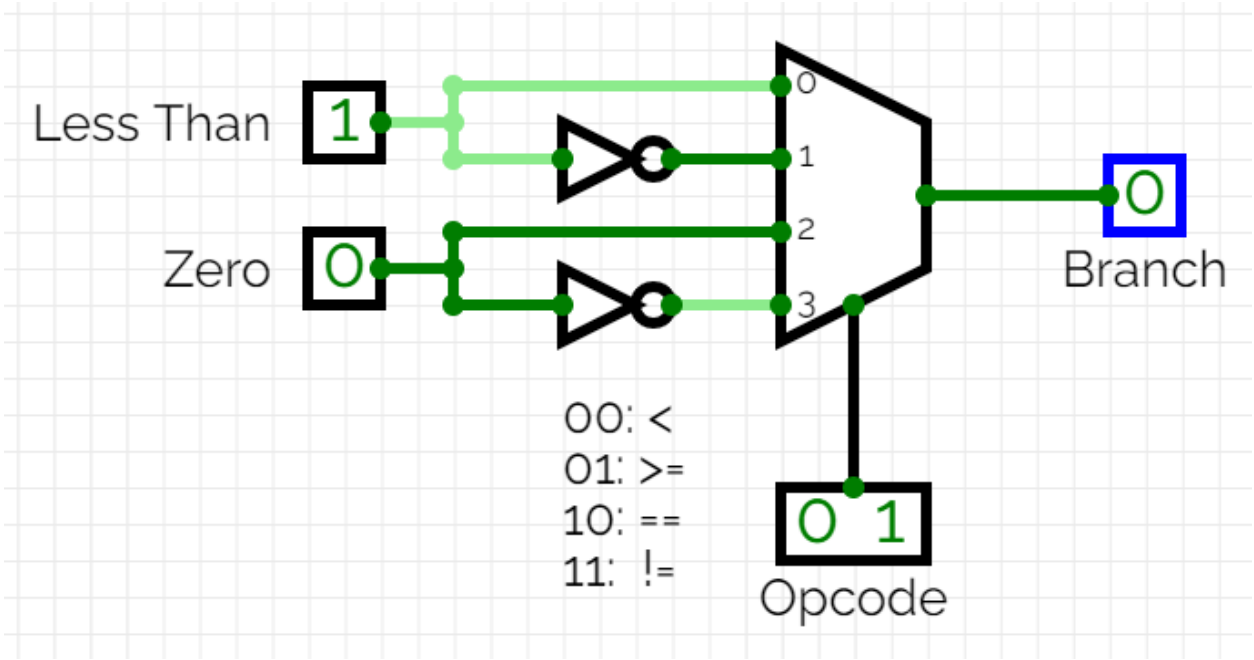
Noel Ball
HW3

Question 1:

A.)

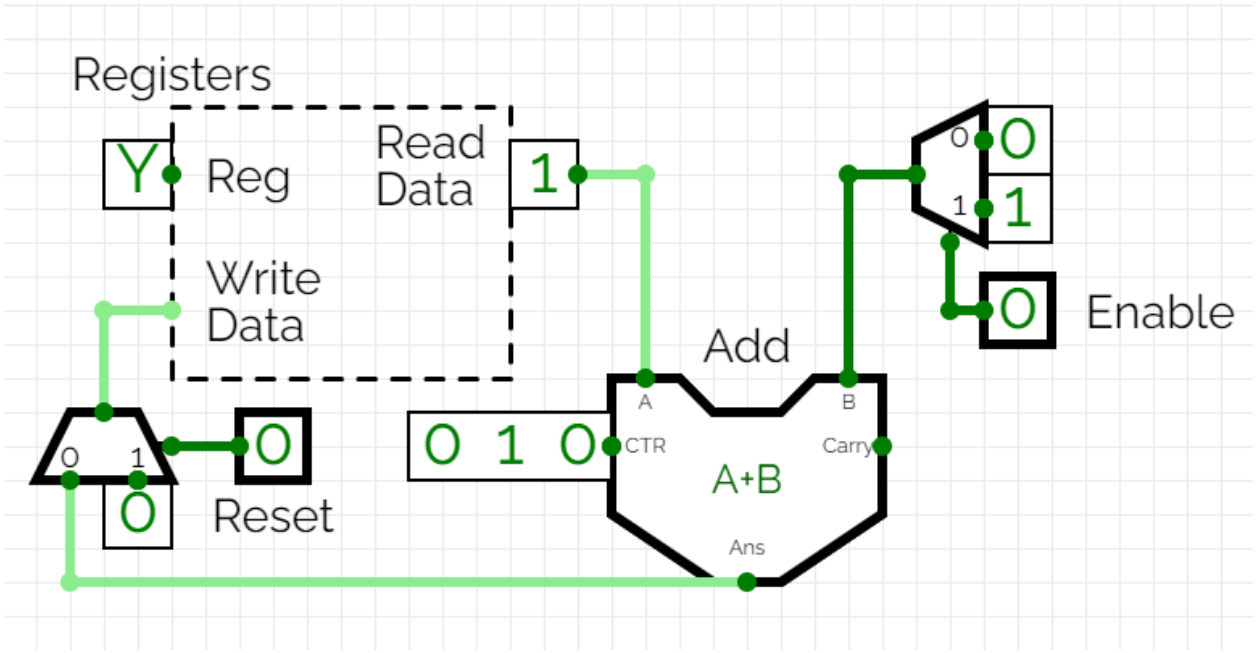
Opcode	00 <	…	…	01 ≥	…	…	10 ==	…	…	11 !=	…	…
Zero	0	1	0	0	1	0	0	1	0	0	1	0
Lt	0	0	1	0	0	1	0	0	1	0	0	1
Branch	0	0	1	1	1	0	0	1	0	1	0	1



Branch is true when:
Opcode 00 and Lt is true,
Opcode 01 and Not Lt is true,
Opcode 10 and Zero is true,
Opcode 11 and Not Zero is true.

B.)

Reset	1	0	0
Enable	X	0	1
Y	0	Y	Y + 1



Read Data accesses the value in Reg and gives it to an ALU set to add from '010' control line.
If Enable is set to true, 1 is added to Y, else Y is unchanged.
Then the ALU output is written to Reg, unless Reset is set to true then 0 is written instead.

Question 2:

Instruction	A	B	C	D	E
Add x_1, x_1, x_2	0x111	0x222	0x333	0x333	0x404
Ld x_2, 5(x_3)	0x333	0x5	0x338	0xeee	0x404
L: beq x_2, x_2, L	0x222	0x222	X	X	0x400

Question 3:

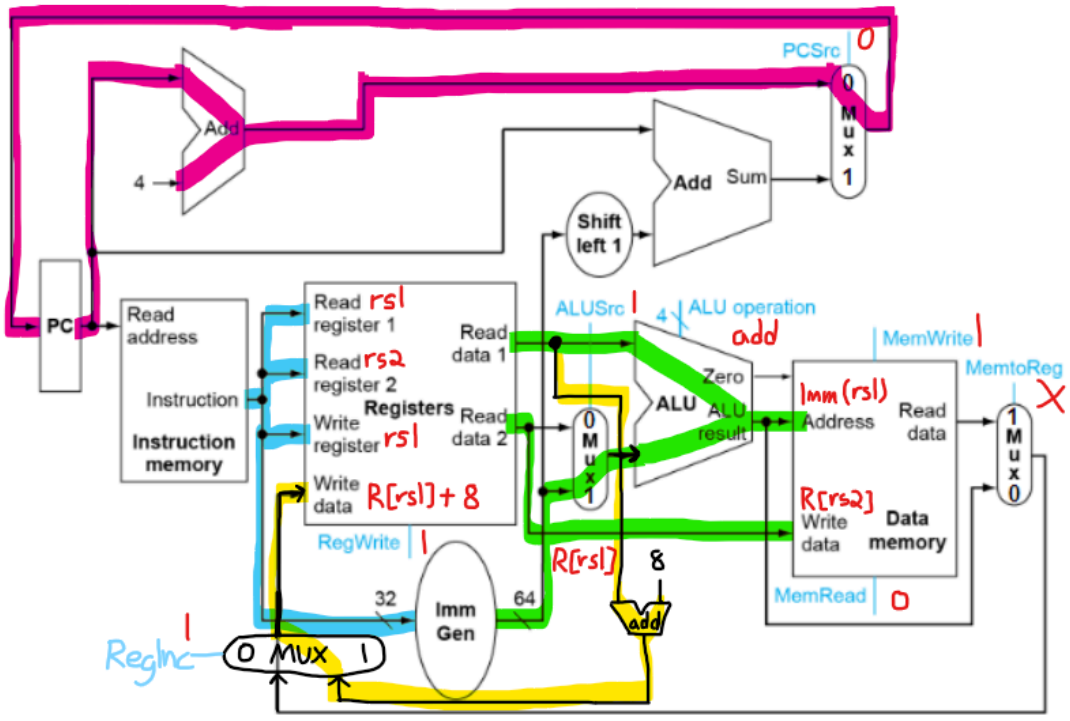
Instruction	ALUOp	ALUSrc	PCSrc	MemtoReg	MemRead	MemWrite	RegWrite
Ori x_5, x_7, 9	OR |	1	0	0	0	0	1
Slt x_6, x_7, x_8	LT <	0	0	0	0	0	1
Ld x_7,16(x_2)	ADD +	1	0	1	1	0	1
Sd x_{10}, 0(x_5)	ADD +	1	0	X	0	1	0
J foo	X	X	1	X	0	0	0

Question 4:

Ai.) No changes are required to implement ldrr. ALUSrc needs to be set to 0 so the ALU can get the second register read instead of an immediate. Setting the ALU operation to add sums the first and second register reads, giving the address to load memory from. MemRead needs to be set to 1 and MemWrite to 0 since we are loading not storing, and MemtoReg and RegWrite need to be set to 1 for the value from memory can be written to the destination register. PCSrc should be set to 0 for a typical increment of the program control to move to the next instruction.

Aii.) The R-type instruction format can be used because it offers both rs1 and rs2 source register fields while still including the rd destination register field. The func3 and func7 fields can be used to specify the add operation and differentiate it from similar instruction calls using the same format and opcode.

Bi.) Implementing sdinc will need changes. The MemtoReg mutex can't help increment rs1 so a new mutex 'RegInc' will be needed. It can be controlled to 1 when we need to increment a register, or controlled to 0 to be ignored for typical instructions. The MemtoReg output wire will be the 0 input, and it will output to Write Data in registers either for a typical instruction or to write the increment. A new wire can branch from Read Data 1 giving R[rs1] to an add ALU along with a constant 8 for the increment. This goes to the 1 input of RegInc to go to Write Data, and rs1 needs to be passed in to the Write Register input in registers. The rest of the control lines should have ALUSrc as 1, ALU operation as add, MemWrite as 1, MemRead as 0, MemtoReg as don't care, RegWrite as 1, and PCSrc as 0.



Bii.) The S-type instruction format can be used because it offers both rs1 and rs2 source register fields and a specific immediate field for the memory offset to store to. The func3 and second immediate fields can be used to specify the add operation and differentiate it from similar instruction calls using the same format and opcode.