Table used to implement control:

instruction	**opcode**	**BR**	**JP**	**ALUinB**	**ALUop**	**DMwe**	**Rwe**	**Rdst**	**Rwd**	**output**	**input**
lw	0000	0	0	1	010	0	1	0	1	0	0
sw	0001	0	0	1	010	1	0	X	X	0	0
beq	0010	1	0	X	011	0	0	X	X	0	0
blt	0011	1	0	X	011	0	0	X	X	0	0
or	0100	0	0	0	000	0	1	1	0	0	0
and	0101	0	0	0	001	0	1	1	0	0	0
addi	0110	0	0	1	010	0	1	0	0	0	0
add	0111	0	0	0	010	0	1	1	0	0	0
sub	1000	0	0	0	011	0	1	1	0	0	0
subi	1001	0	0	1	011	0	1	0	0	0	0
shr	1010	0	0	0	100	0	1	1	0	0	0
shl	1011	0	0	0	101	0	1	1	0	0	0
j	1100	0	1	X	X	0	0	X	X	0	0
jal	1101	0	1	X	X	0	1	X	X	0	0
output	1110	0	0	X	X	0	0	0	0	1	0
input	1111	0	0	X	X	0	1	1	X	0	1

Parts of the processor that work: ALL parts (ALU, Jump, Branch, Memory, Control, I/O) work as all visible Gradescope test cases have passed.

Parts and their interfaces:

- **ALU: toggle A and B to change inputs and optype to change the operation to be performed**
- **Control: toggle opcode to see which control signals are activated for which instructions (0000 to 1111 in homework assignment order)**
- **RegisterFile: toggle data_in and register numbers to change what is written where**
- **EqualCheck and LessThanCheck: toggle register contents to determine whether first is = to second or first is < second or not.**
- **Everything else is very straightforward to test.**