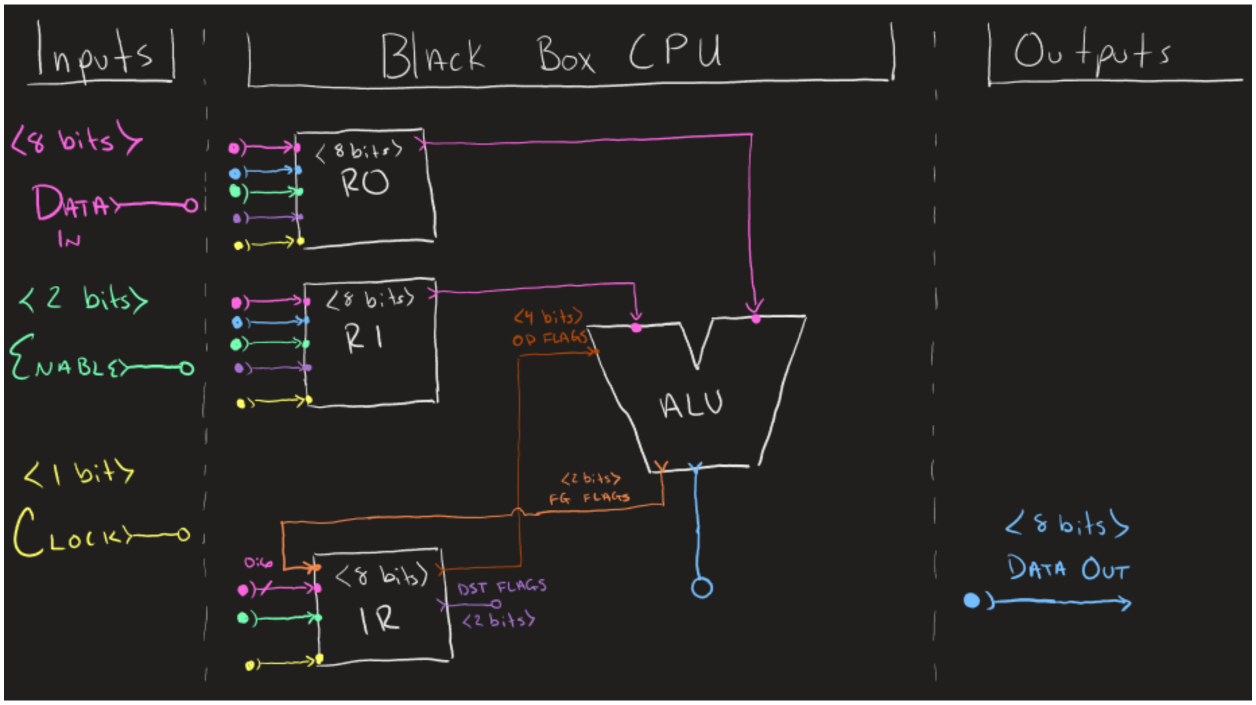
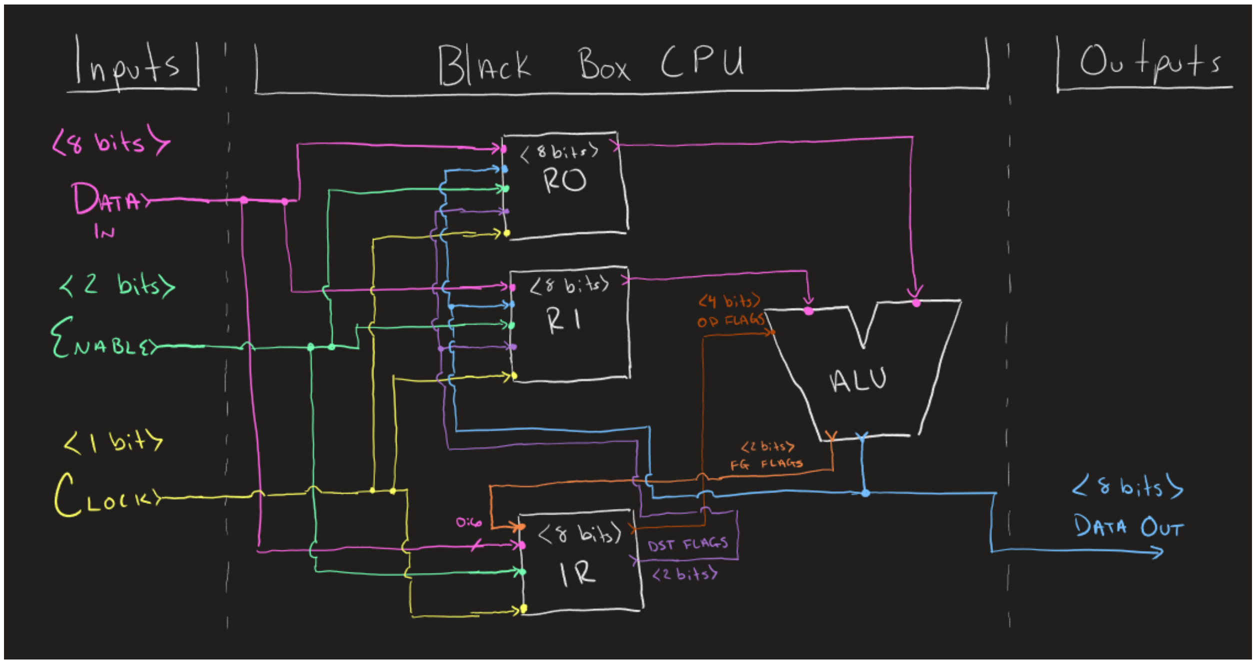
CSC 565 – Computer System Architecture
Deliverable 2
Cason Konzer

○ BLOCK DIAGRAM WITH INPUT OUTPUT PINS





○ CLOCK SIGNAL

External Clock used for flexibility, and so the programmer knows when to read values on the output bus. (Realistic implementation supports both internal and external clock).

- ### NUMBER OF REGISTERS

Register	**Description**
IR	Instruction Register (Also Holds Flags)
R0	Register 0
R1	Register 1

3 Registers (Data Out is coded as a register)

- ### FLAGS

Supporting Signed and Overflow, Overflow only relevant for SUB instructions, Signed supported for both SUB and OR

- ### OTHER CONTROL REGISTERS THAT WILL BE NEEDED TO PERFORM OPERATIONS

Use extra bits in IR to hold flags from ALU output, as well as pass ‘instruction’ flags specifying operands, operation, and output destinations.

- ### A DRAFT OF THE INSTRUCTION SET

Instruction/Flag Bit Encoding

OVF	**SGN**	**OR**	**SUB**	**OPO 1**	**OPO 0**	**DST 1**	**DST 0**
Overflown Output	Signed Output	ALU flag to use OR	ALU flag to use SUB	Operand 1	Operand 0	Destination 1	Destination 0
BIT 7	*BIT 6*	*BIT 5*	*BIT 4*	*BIT 3*	*BIT 2*	*BIT 1*	*BIT 0*

Destination Encoding

DST 1	**DST 0**	**ALU Write Back Register(s)**
0	0	No Write Back (only bus)
0	1	Write Back to R0
1	0	Write Back to R1
1	1	Write Back to R0 & R1

ALU Encoding

OR	**SUB**	**ALU Operation**
0	0	unused
0	1	Integer Subtraction
1	0	Logical Or
1	1	unused

Operation Encoding

OPO 1	OPO 0	Output (* is the operation)
0	0	R0 * R0
0	1	R1 * R0
1	0	R0 * R1
1	1	R1 * R1

Output Flag Encoding

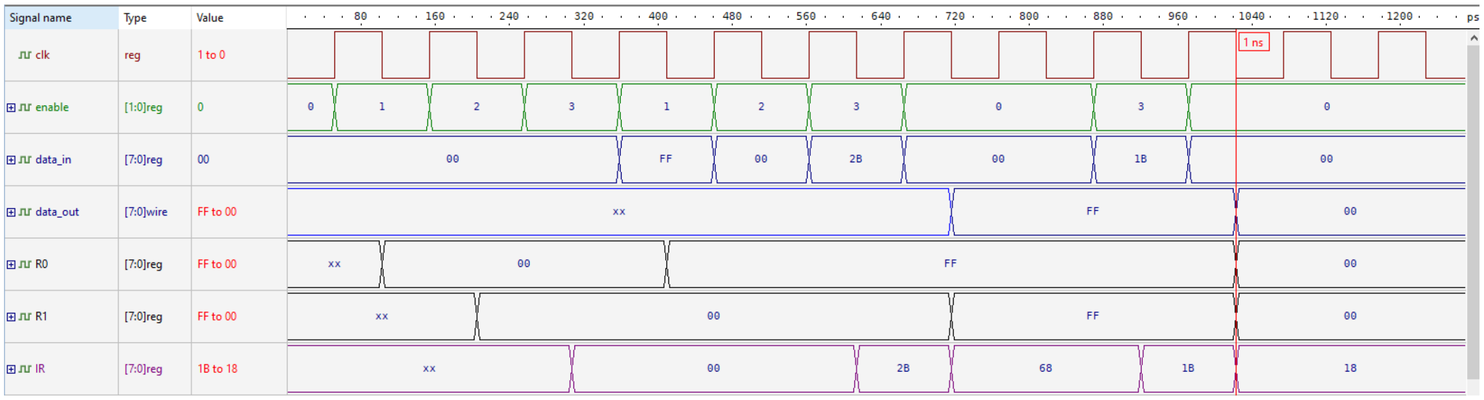
OVF	SGN	Characteristics
0	0	Operation has not overflow and output has positive sign
0	1	Operation has not overflow and output has negative sign
1	0	Operation has overflow and output has positive sign
1	1	Operation has overflow and output has negative sign

Enable Encoding

ENA 1 *(BIT 1)*	ENA 0 *(BIT 0)*	Enabled Registers
0	0	None (required for ALU output loads)
0	1	R0
1	0	R1
1	1	IR

- ### TIMING DIAGRAM FOR THE FOLLOWING INSTRUCTIONS

```
// RESET ALL REGISTERS
// LOAD FF INTO R0
// LOAD 00 INTO R1
// DATA_OUT, R0, R1 = R0 | R1 = FF
// NO OP
// DATA_OUT, R0, R1 = R0 - R1 = 00
// NO OP
// NO OP
```


Signal name
Type
Value
ps
1 ns
clk
reg
1 to 0
enable
[1:0]reg
data_in
[7:0]reg
00 FF 00 2B 00 1B 00
data_out
[7:0]wire
FF to 00
xx FF 00
R0
[7:0]reg
FF to 00
xx 00 FF 00
R1
[7:0]reg
FF to 00
xx 00 FF 00
IR
[7:0]reg
1B to 18
xx 00 2B 68 1B 18