Lecture 20

Simple Processor

- Continued from previous lecture
- 4 instructions
  - Load: `Rx <= Data`
  - Move: `Rx <= Ry` (multiple registers can read from the bus at the same time)
  - Add: `Rx <= Rx + Ry`
  - Subtract: `Rx <= Rx - Ry`
- Encoding for each intstruction (2 bit encoding in this case)
- Encoding for each register (2 bits in this case)

Simplified Example

```
mov R2, R1 \\R2 <- R1
Add R1, R2 \\R1 <- R1 + R2
```

where the instruction encoding for the second operation would be `100110`, where `10` stands for "Add", `01` for R1 and `10` for R2, and a FSM is used to complete the process

- Each instruction has 8 bits
- In the example above, there are 2 trailing bits that are ignored (because the processor knows they are not needed for "Add")
- The longest operation needs 3 steps: Add/Sub
- A 2 bit counter is needed to count which step we are on

Q_1W_0	$T_0T_1T_2T_3$
00	1000
01	0100
10	0010
11	0001

- Function register
  - takes in f_1f_0 and returns 4 bits representing the operations
  - takes Rx and returns 4 bits representing the value of Rx similar to the above table
  - same for Ry
  - Control Signals for each step

	T_1	T_2	T_3
I_0: Load	Extern = 1 Rin[3:0] X Done = 1		

	T_1	T_2	T_3
	Rin[3:0] = X[3:0]		
I_1: move	Rout[3:0] = Y[3:0]		
	Done = 1		
	Rout[3:0] = X[3:0]	Rout[3:0] = Y[3:0]	Gout = 1
I_2: Add	Ain = 1	Add/Sub = 0	Rin[3:0] = X[3:0]
		Gin = 1	Done = 1
	Rout[3:0] = X[3:0]	Rout[3:0] = Y[3:0]	Gout = 1
I_3: Sub	Ain = 1	Add/Sub = 1	Rin[3:0] = X[3:0]
		Gin = 1	Done = 1

Assume enables not shown are 0

Ain = $(I_2 + I_3)T_1$
Gin = $(I_2 + I_3)T_2$
Extern = $I_0 I_1$
Gout = $(I_2 + I_3)T_3$
Done = $(I_0 + I_1)$T_1$ + $(I_2 + I_3)T_3$
FRin = wT_0
clear = Done + $\overline{w}T_0$