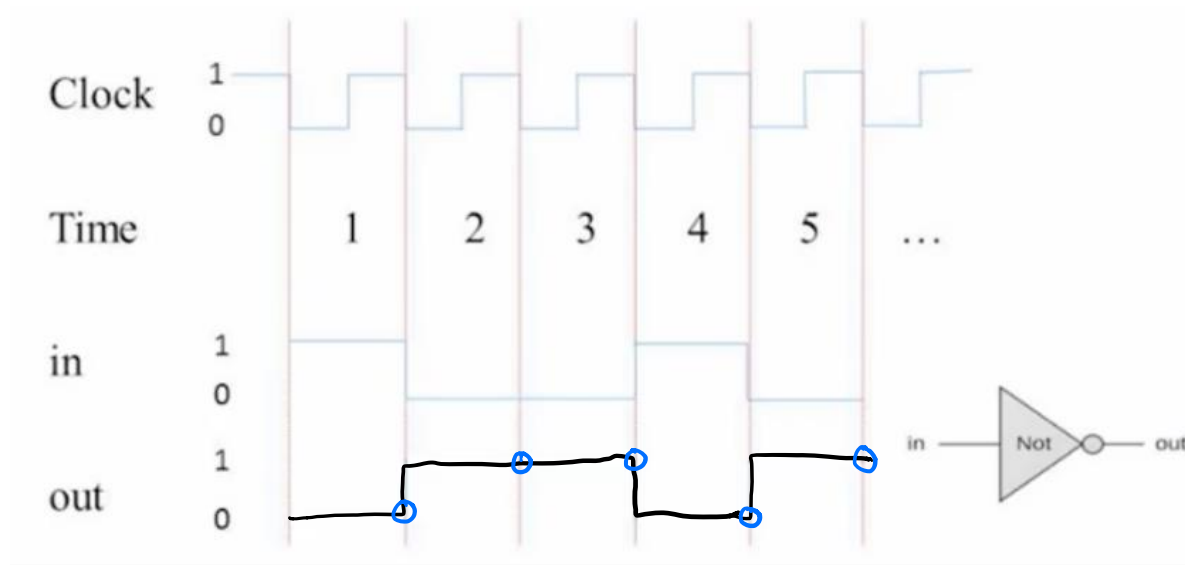# Lab #4 (Sequential Logic)

Name: _Raul Aguilar_
Section/Time: _CS 220 2148_
Date: _September 25, 2020_

1. For the following combinational logic (not gate):

    a. Label all the ticks and tocks on the clock line.

    b. Complete (draw) the output line, given the input during that time.
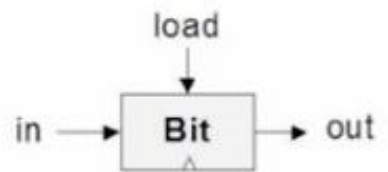
    c. Circle all the latch points.

2. For the following sequential logic (clocked data flip-flop (DFF)):

    a. Complete (draw) the output line, given the input during that time.

    b. Circle all the latch points.
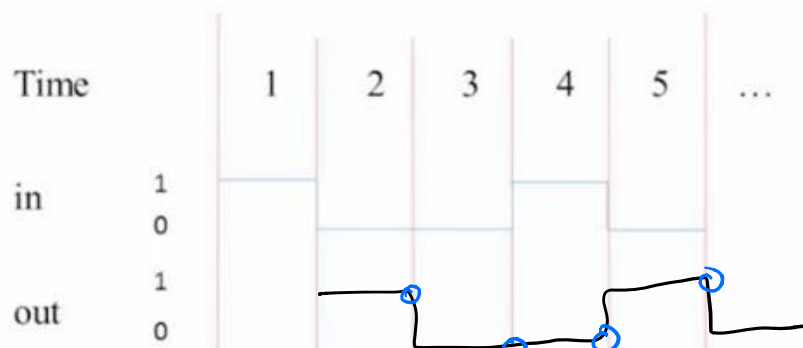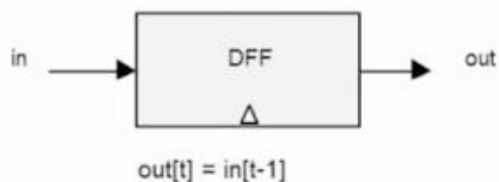
3. For the following sequential logic (1-Bit Register):

    a.   Complete (draw) the output line, given the input during that time.

    b.   Circle all the latch points.

1-Bit Register

load

in → | Bit | → out

if load(t-1) then out(t)=in(t-1)
else out(t)=out(t-1)

load   1
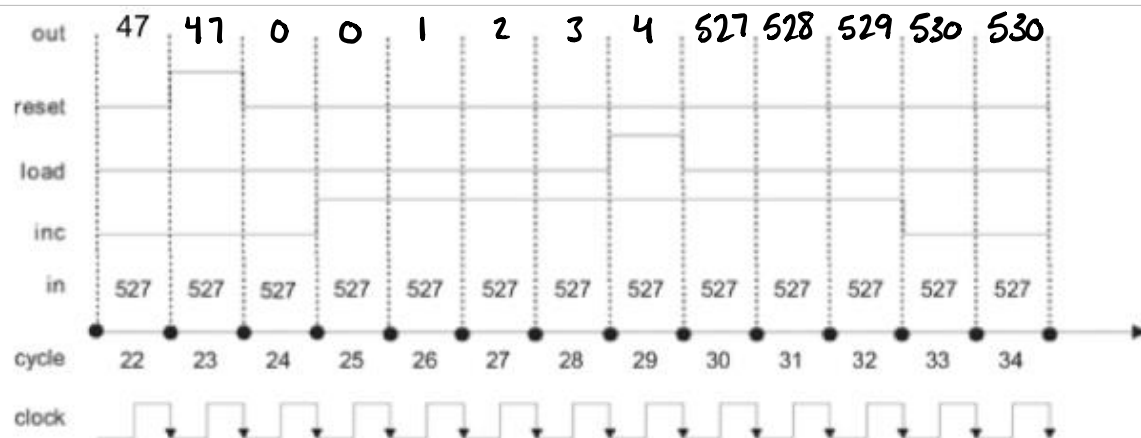   0

in   1
        0

out   1
        0
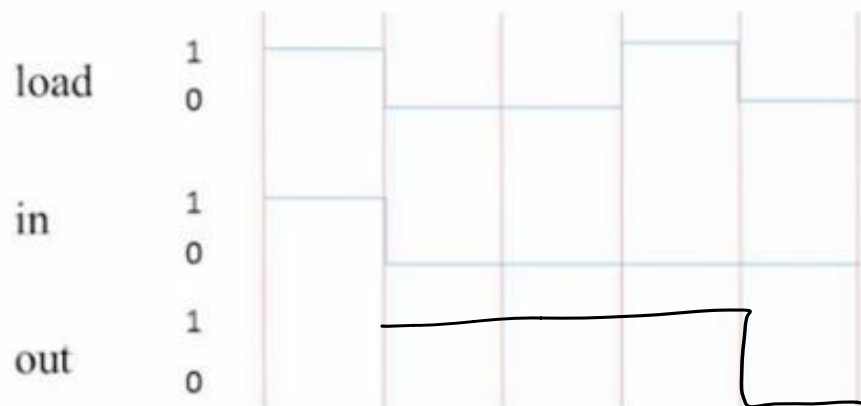
4. Why do we use discrete time steps instead of continuous time?
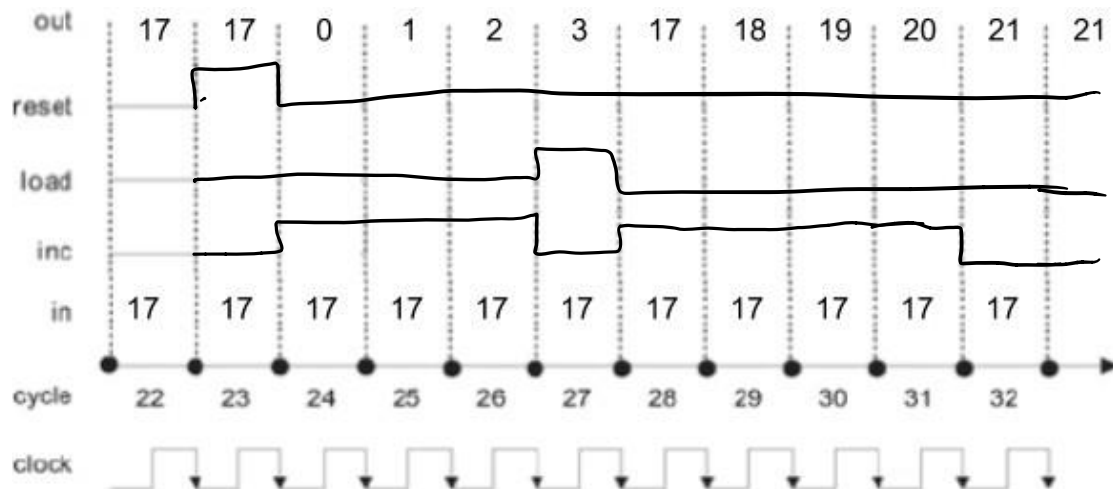
To allow chip or current to stabalize

5. **What are the *out* values for each clock cycle below?** Note: input starts at 527, output starts at 47 (from previous clock cycle).

We assume that we start tracking the counter in time unit 22, and the counter's control bits (reset, load, inc) start at 0. All values below are arbitrary, to test your understanding of clock cycles, latching, etc.

> HINT: Control bits dictate NEXT output
> because outputs latch on tock (end of cycle).
> Order matters! (top to bottom)

out	47	47	0	0	1	2	3	4	527	528	529	530	530

reset

load

inc

in	527	527	527	527	527	527	527	527	527	527	527	527	527
cycle	22	23	24	25	26	27	28	29	30	31	32	33	34

clock

6. **What are the control bits (reset, load, inc) values given the input and output values below?** Note: All control bits start low (0). You can draw the lines and/or put the 0/1 values for each clock cycle and control bit

out	17	17	0	1	2	3	17	18	19	20	21	21

reset

load

inc

in	17	17	17	17	17	17	17	17	17	17	17
cycle	22	23	24	25	26	27	28	29	30	31	32

clock

Lab #4 (Sequential Logic)

Name: _____

Section/Time: _____

Date: _____

Part 1: Definitions

- Define what a "word" is in computer architecture:

    Value stored in a register
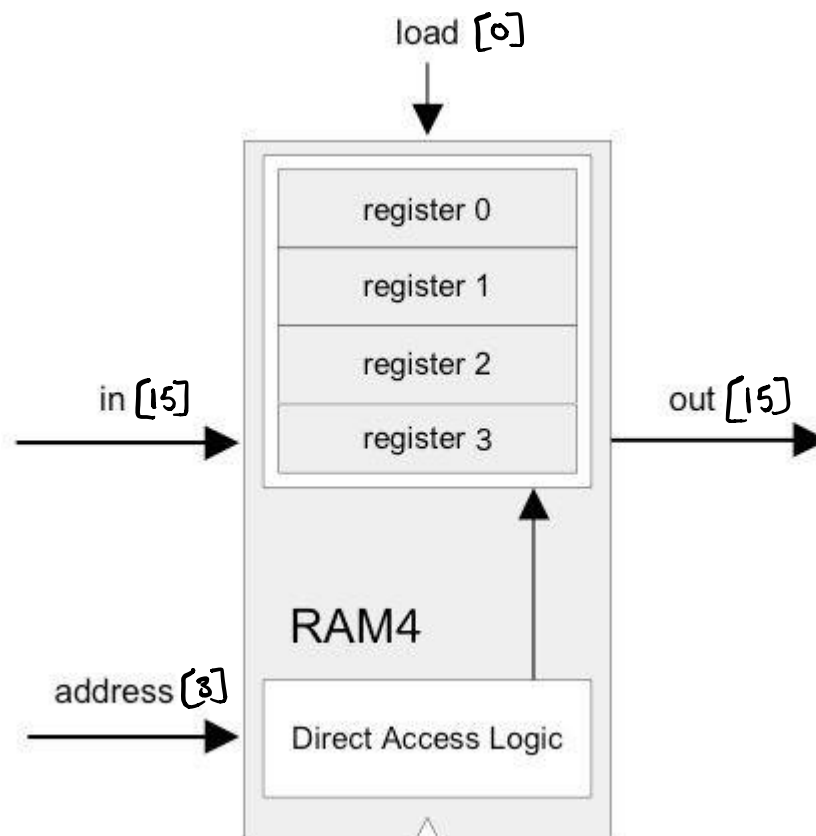
- What is the term for the size of a "word"?

    width

- What is the size of the word in the HACK architecture?

    16-bits

Part 2: Bus Sizes

- Label the size of each of the inputs/outputs for the Ram4 chip below (in, address, load, out). You can assume the register sizes are the same as our HACK architecture.

load [0]

register 0

register 1

register 2

in [15]   register 3   out [15]

RAM4

address [3]

Direct Access Logic

Part 3: Reasoning

- Why did you choose the address size above?

    00
    01
    11
    10

Part 4: Bus Sizes
- **Write out the address for each register:**

Register #	In Binary	
	[1]	**[0]**
0		
1		
2		
3		

- **Each register has it's own load and in values. Fill out the table below for each situation:**

Situation:	Register 0		Register 1		Register 2		Register 3		RAM4 OUTPUT (out, after end of cycle)
	in	**load**	**in**	**load**	**in**	**load**	**in**	**load**	
Store -15 in Register 0	1	-15	0		0		0		−15
Store -15 in Register 1			1	-15					
Store -15 in Register 2					1	-15			
Store -15 in Register 3							1	-15	
Store -15 in Register 1 and 2			1	-15	1	-15			

- **When does the load value matter? When doesn't it?**

when load = 1

- **What chip (that we've built in the course so far) would help for handling the load?**

DMux4Way

- **When does the in value matter? When doesn't it?**

when load = 1

- **What chip (that we've built in the course so far) would help for handling the in?**

if (load)
$$out[t+1] = in[t]$$ = Mux
else
$$out[t+1] = out[t]$$

Write the following numbers in binary (NO CALCULATOR! Use back of paper):
  ○ *Tip: put a space between every 4 digits (grouping from right to left)*

Decimal Number:	Binary number:
7	0000 0000 0000 0111
63	0000 0000 0011 1111
511	0000 0001 1111 1111
4095	0000 1111 1111 1111
16383	0011 1111 1111 1111

- **The above values can be thought of as the last register in a specific RAM chip**
  ○ **Ex: RAM8 = 8 registers = 0, 1, 2, 3, 4, 5, 6, 7 are possible registers**
- **What would be the address sizes for the following n Registers:**

RAMn (n = num. of registers)	Number of bits for address (think highest value…)
RAM8	$2^3 = 8$ 3 bits
RAM64	$2^6 = 64$ 6 bits
RAM512	$2^9 = 512$ 9 bits
RAM4K (n=4096)	$2^{12} = 4096$ 12 bits
RAM16K (n=16384)	$2^{14} = 16384$ 14 bits

- **How does one calculate the size of the address (number of bits) from the word size for the registers?**