CS 61C Spring 2015
Guerrilla Section 1: Hardware & CPU

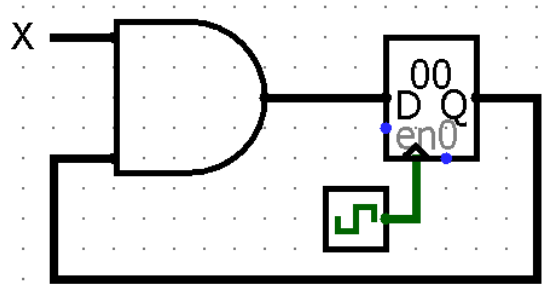
Problem 1:

a) Convert the following truth table to a Boolean expression and simplify it. An X means we don't care about the value of that output (it can be either 0 or 1).

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	X
1	1	1	X

b) Draw the transition state diagram from a FSM that reads a binary string bit-by-bit and outputs whether the total number of 1s seen since the beginning is divisible by 3.

c) For the circuit below, assume that the setup time is 15ns, hold time is 30ns, and the AND gate delay is 10ns. If the clock rate is 10 MHz and x updates 25ns after the rising edge of the clock, what are the minimum and maximum values for the clk-to-Q delay to ensure proper functionality?



Min: _____ns

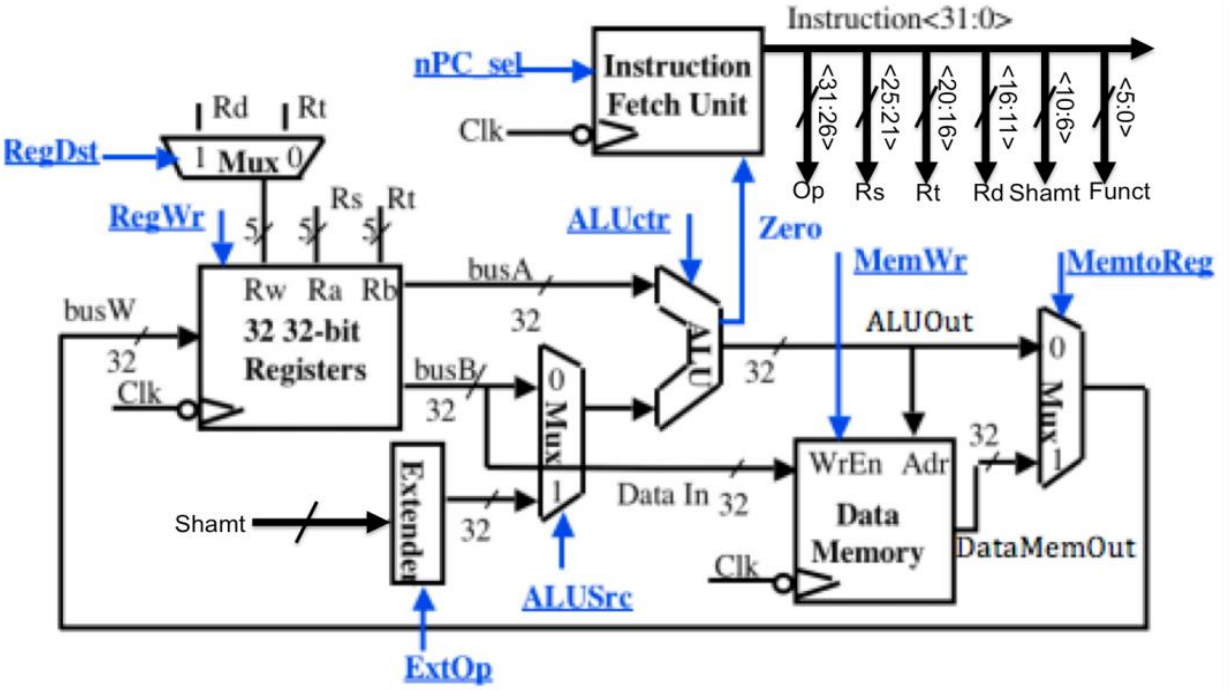
Max: _____ns

Problem 2 (adapted from Sp04 Final):

We want to implement a new I-type instruction `swai` (store word then auto-increment). The operation performs the regular `sw` operation, then increments the value in the `rs` register by 1.

The RTL for the `swai` instruction is:

```
Mem[R[rs] + SignExtImm] = R[rt]; R[rs] = R[rs] + 1; PC = PC + 4
```



a) Modify the single-cycle MIPS datapath (shown above), and describe your changes in the space below. Your modification may use simple adders, mux chips, wires, and new control signals. You may replace original labels where necessary.

b) Fill out the values of the control signals in the table below, including any new control signals that you have added in part A.

RegDst	RegWr	nPC_sel	ExtOp	ALUSrc	ALUctr	MemWr	MemtoReg		

Problem 3 (adapted from Su13 Final):

Assume that we run the following snippet of code on a 5-stage pipelined MIPS CPU with **no optimizations**. Branch checking is done in the execute stage. Assume that `$a1 = 1` at the beginning of the code.

```
            lw $t0, 0($a0)
loop:       beq $a1, $0, exit
            sll $t0, $t0, 2
            addiu $a1, $a1, -1
            sw $t0, 0($a0)
            j loop
exit:
# when we reach the exit label, we're done
```

a) After which instructions are stalls needed? What is the total number of clock cycles for these instructions to finish execution (when the pipeline becomes empty)? You may use the table below as scratch space.

Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
lw																
beq																
sll																
addiu																
sw																
j																
beq																

b) Consider the following optimizations *separately*. How many FEWER cycles are taken for the addition of each optimization?

   a. Forwarding

   b. Branch prediction of never take branch