Your Name: _____ SID: _____

	EECS151/251A **Spring 2022 Midterm**	
Name of the person on left (or aisle)	Proctor's Name	Name of the person on right (or aisle)

Question	1	2	3	4	**Total**
Max. points	7	13	24	26	**70**

Exam Notes:
You have 80 minutes to work, starting at 12:40PM Pacific Time and ending at 2:00PM Pacific Time.

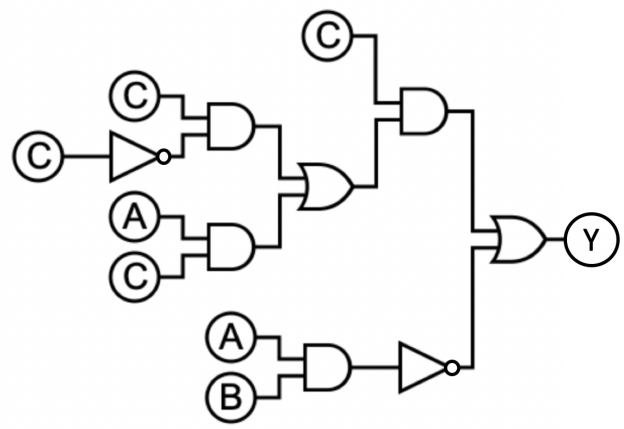
Before 12:40pm Pacific Time, you may write down your name, SID, names of the persons next to you, etc., on the first page but you may NOT begin working.

Please write your name and SID on EVERY page.

Problem 1: Boolean Simplification [7 Points]

Part a)

(1) Please write down the Boolean expression directly for the following circuit (match the logic gates below without simplifying). [2pts]



(2) Simplify the Boolean expression in part (1). [2pts]

Part b)

Please translate the following K-map into Boolean expression and simplify. [3pts]

$$Y(A, B, C, D) =$$

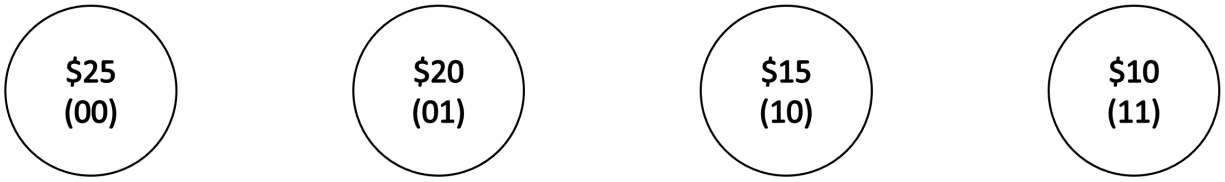
CD / AB	00	01	11	10
00	0	0	1	1
01	0	1	1	1
11	0	1	1	1
10	0	0	1	1

Problem 2: Finite State Machines: Fastrak [13 Points]

Part a)

You are designing an FSM for a Fastrak (in-car toll tracking) device for the Bay Area. When you first purchase and place it in your car, you load it with $25. Every use of the toll bridge costs $5 and automatically charges from the device. Once the Fastrak device has reached $5, it reloads itself from your credit card with $20.

Draw a Mealy FSM with an input that is 1 for 1 cycle every time a toll bridge is used and an output that is 1 for 1 cycle every time a reload is needed. You may assume the reload always occurs by one cycle after the output is high. You will not incur tolls in consecutive cycles. [4pts]



Part b)

Redraw the FSM for a Moore machine. [4pts]

Part c)

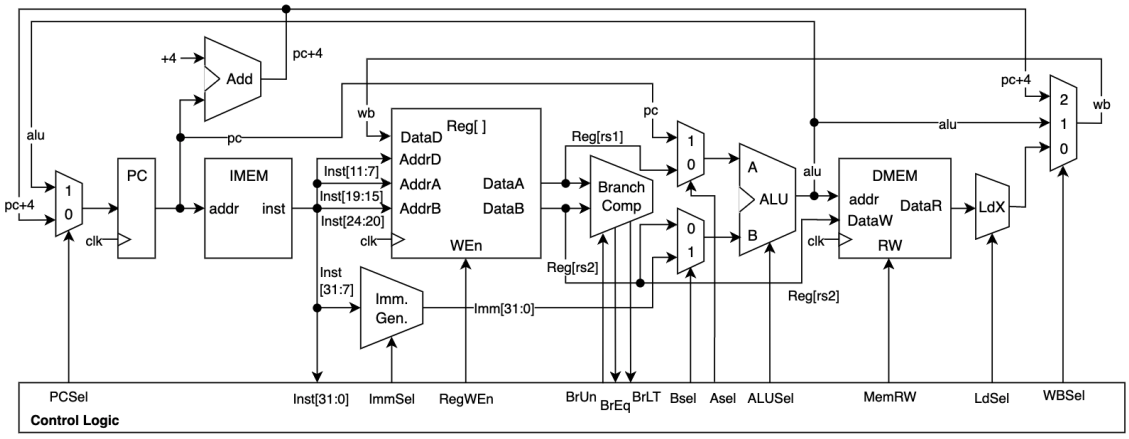
Find the minimum logic to determine the new state *n1,n0* where *n0* is the LSB, from the current state *c1,c0* where *c0* is the LSB, and the input *in* based on the Mealy machine. [3pts]

Part d)

Find the minimum logic to determine the output *out* from the current state *c1,c0*, where *c0* is the LSB, and the input *in* based on the Mealy Machine. [2pts]

Problem 3: RISC-V Instructions/Datapath [24 Points]

You may refer to the RISC-V Green Card on the last pages of this exam.



The single-cycle datapath above implements a subset of the RV32I instruction set.

Part a)

In the fabrication of any digital circuit, there may be manufacturing defects. One type involves a signal being shorted to GND or VDD (stuck-at-zero or stuck-at-one). Among the provided RISC-V instructions, please mark **N** in the table entries to the instructions that will **no longer work for the following stuck signals**, and leave others as blank. [12pts]

-	ASel is GND	BSel is VDD	RegWEn is GND
beq			
and			
jal			
auipc			
lw			
sw			
addi			
jalr			

Part b)

We want to implement a function called `maxpool_bias`, that pools out maximum value of a given array `arr`, with length of `LEN` and then add with integer `bias` with the final value. The following code describes its operation.

```
def maxpool_bias(arr, bias):
    c = arr[0]
    for i in range(1, LEN):
        c = max(c, arr[i])
    c += bias
```

Assume that intermediate values of `c`, and `arr` elements are stored in register file, so that you do not need to access memory (no need for `sw` or `lw`). The length of the array `LEN` is 3 (each array element pre-loaded in register `rs1, rs2, rs3`), `bias` is pre-loaded in register `rs4`, and result c is written to register `rd`.

For simplicity, we will be using the following unrolled version:

```
c = arr[0]
c = max(c, arr[1])  # Iteration 1
c = max(c, arr[2])  # Iteration 2
c += bias           # Final stage
```

(1) Write the RISCV instruction set flows to complete `maxpool_bias` operation described as the above code block without changing any datapath or control signals. Use only the register `rs1, rs2, rs3, rs4, rd`. You might not need to use all the lines provided. [4pts]

```
                add rd, rs1, x0
Iteration1:

_____

_____

_____

Iteration2:

_____

_____

_____

FinalStage:

_____

_____

_____
```

(2) Calculate the number of cycles it would take for the single cycle datapath to finish the operation. [2pts]

Min: _____

Max: _____

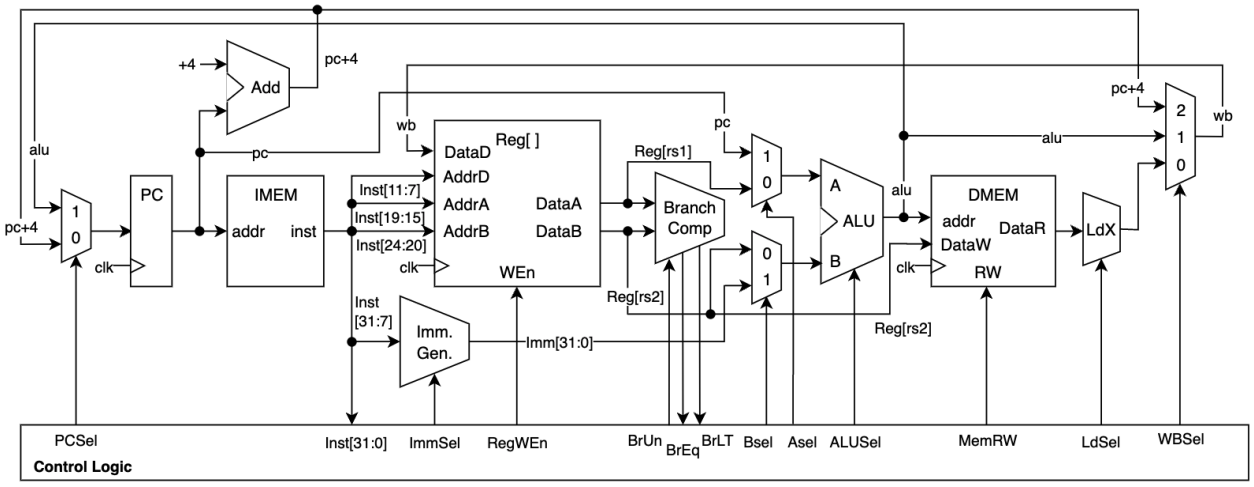
Part c)

This time, we implement a new instruction, `max` that pools out maximum value between two registers and returns it to the destination register by only modifying existing hardware components and control signals.

(1) What type of instruction is the new instruction? [1pt]

Answer: _____

(2) Describe the hardware and control signal change to support this new instruction. You can mark your changes in the following datapath diagram. [4pts]



(3) Calculate the number of cycles it would take for the `maxpool_bias` operation in part b with the new instruction and updated datapath. [1pt]

Answer: _____

Problem 4: Pipelining/Hazards [26 Points]

Part a)

Sasha wants to improve the performance of their single-cycle RISC-V datapath. After analyzing their design, Sasha breaks down the delay of each stage of their datapath. Additionally, Sasha calculates that adding a pipeline register would add 0.2ns of additional delay to the previous pipeline stage. They then consider potential three-stage and five-stage pipeline implementations, depicted in Figure 1. In these designs the register file is synchronously written to at the end of a cycle, and read asynchronously (*i.e.* a write from the writeback stage is only available on the following cycle.)

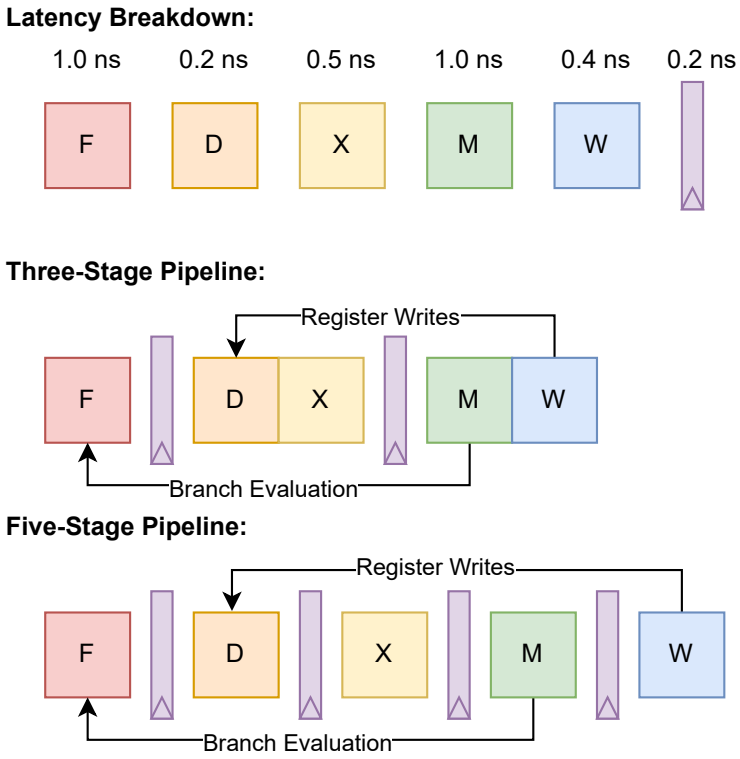


Figure 1: Potential CPU Pipelines

(1) Sasha wants to select an optimal CPU design. Assuming no hazards, what is the instruction latency (*i.e.* from fetch to writeback) for a single instruction for each design? What about the minimum clock period? [6pts]

CPU Implementation	**Instruction Latency (ns)**	**Minimum Clock Period (ns)**
Single Cycle		
Three-Stage		
Five-Stage		

(2) If Sasha wants to optimize their design to maximize instruction throughput (*i.e.* instructions executed per unit time) assuming no hazards, which design should they choose and why? [2pts]

Part b)

(1) Consider the following assembly program. Fill in the pipeline tables below for both the three-stage and five-stage implementation, factoring in hazards and assuming no forwarding/branch prediction. [12pts]

Some columns of the table may be left blank. You may abbreviate the combined decode/execute stage as "D" and the combined memory/writeback stage as "M".

```
add  x1, x2, x3
sub  x4, x5, x1
xor  x2, x3, x1
bne  x1, x2, not_t // branch is not taken
ori  x1, x2, 5
```

Three-Stage Pipeline:

Instruction\Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
`add x1, x2, x3`	F	D	M																
`sub x4, x5, x1`		F																	
`xor x2, x3, x1`																			
`bne x1, x2, not_t`																			
`ori x1, x2, 5`																			

Five-Stage Pipeline:

Instruction\Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
`add x1, x2, x3`	F	D	X	M	W														
`sub x4, x5, x1`		F																	
`xor x2, x3, x1`																			
`bne x1, x2, not_t`																			
`ori x1, x2, 5`																			

(2) For each implementation, how many cycles occur between the completion of the first instruction to the last? How much time elapses? [2pts]

CPU Implementation	**Cycles**	**Time (ns)**
Three-Stage		
Five-Stage		

(3) If Sasha wants to optimize their design to maximize instruction throughput for the given program, which design should they choose and why? [2pts]

(4) Would this answer change given full forwarding/bypassing for data dependencies and a branch-not-taken predictor (assuming there is no implementation cost for either feature)? [2pts]