

*Instructions: 3 hours closed book, notes. Place answers in answer boxes at the bottom of each page. Show all work in blank space above the answer box. No electronic devices permitted.*

1. Let  $X=12_{(8)}$ ,  $Y=71.1_{(8)}$

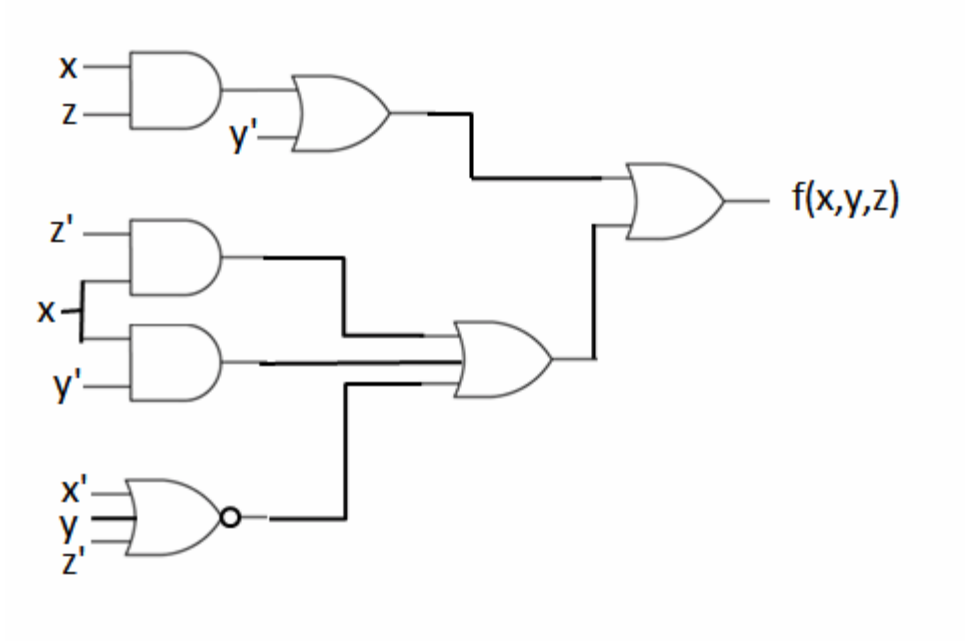
(a) Compute  $Z_{(8)}=11_{(8)} * X + Y$

(b) Express  $Z$  as a binary number

(c) What is the 8's complement of  $Y$ ?

1 (a)	1 (b)	1 (c)
-------	-------	-------

2. The circuit diagram shown is a non-minimal realization of  $f(x,y,z)$ . Sketch the circuit diagram of a minimal realization of  $f$ . HINT: simplify  $f$ .



2.

3.

$$f(w, x, y, z) = \sum m(0, 1, 2, 7, 8, 9, 10, 11)$$

(a) Sketch the Karnaugh Map for  $f$ .

(b) Write a minimal DNF realization for  $f$ .

(c) Suppose you could assign any one cell of the KM as don't-care, making  $f$  an incomplete Boolean function. Which cell would you assign in order to make the resulting minimal DNF realization as small as possible? Justify your choice.

3 (a)

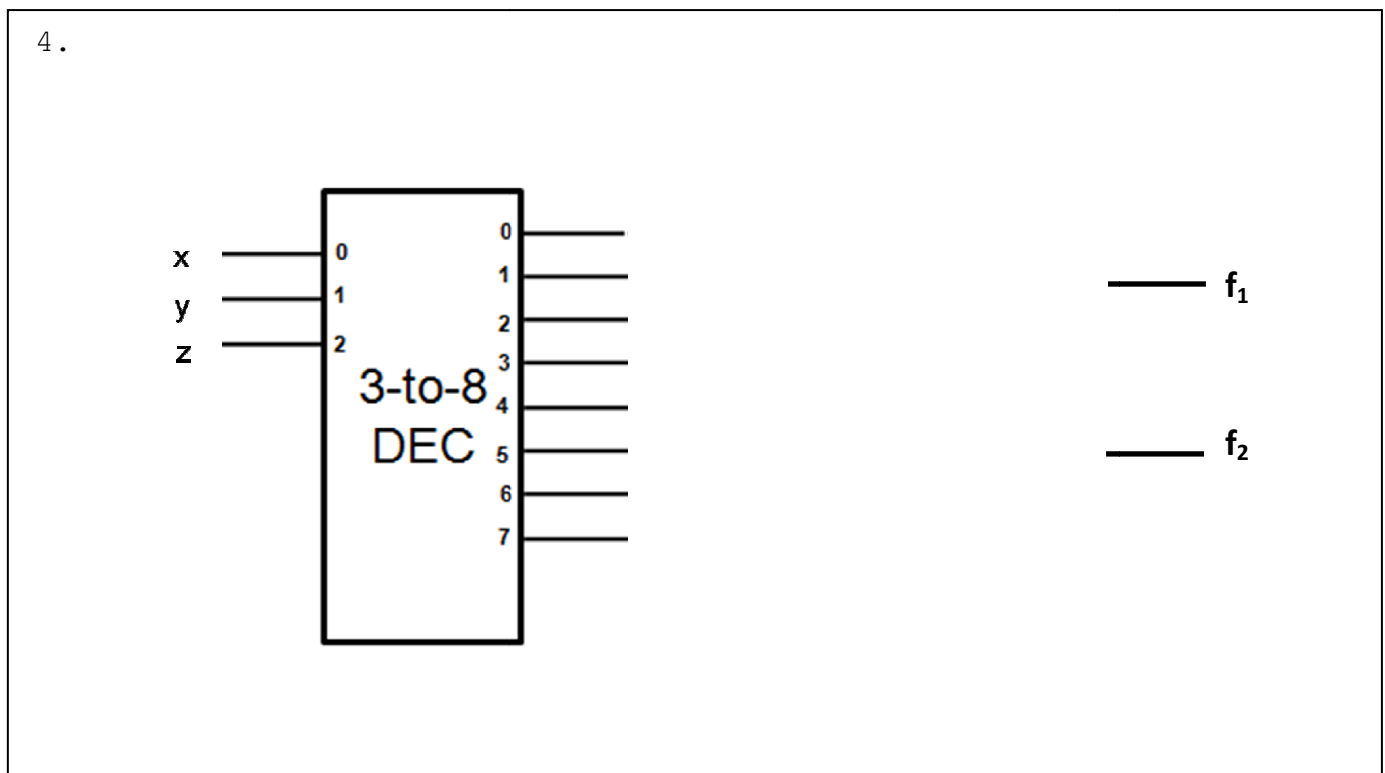
yz


WX

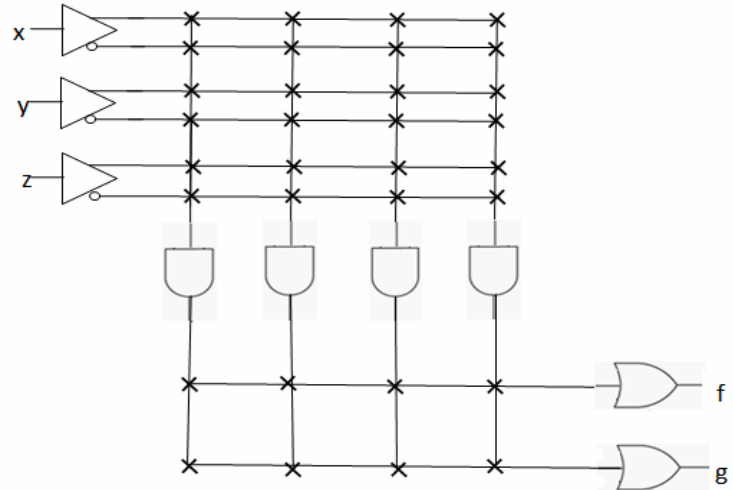
3 (b)

3 (c)

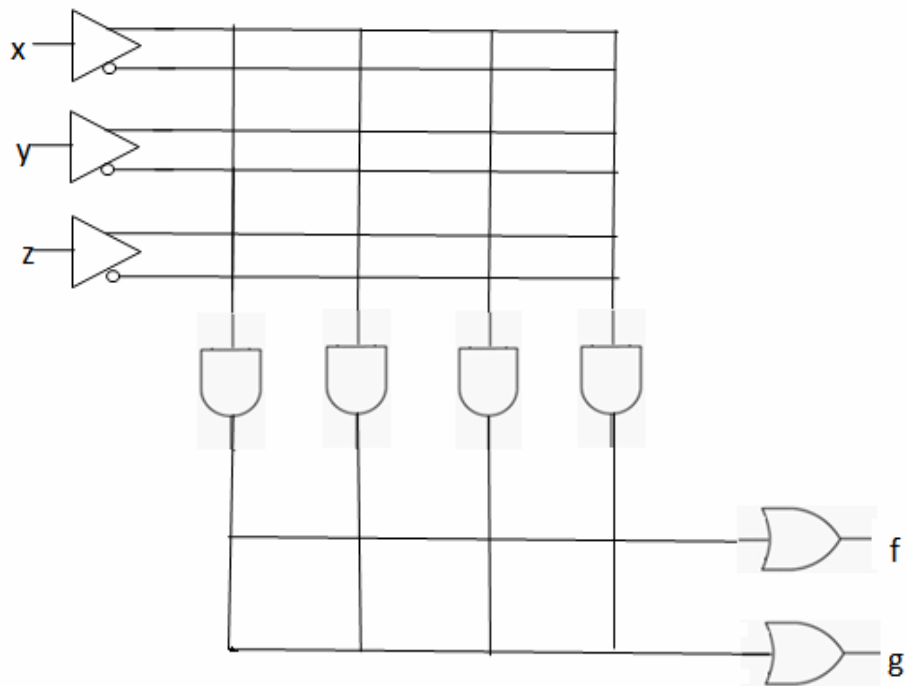
4. Design a 3-input 2-output circuit so that its output  $f_1=1$  when exactly 2 of its 3 inputs are 1, and its other output  $f_2=1$  when all of its 3 inputs are 1 ( $f_2$  will be used to turn on a yellow warning lamp and  $f_1$  a red danger lamp). Use the given 3-8 decoder plus some additional gates to design this circuit.



5. Design a circuit with 3 inputs  $x, y, z$  and 2 outputs  $f, g$  where  $f=x+y$ ,  $g=xy'+x'y+z$ . Use the  $3 \times 4 \times 2$  PLA chip shown immediately below. Recall that both the AND-array and OR-array are fuse programmable in PLAs. Show your answer in the answer box by putting x's in the locations where the fuses are not blown. This PLA chip does not include XOR gates at the outputs.



5.



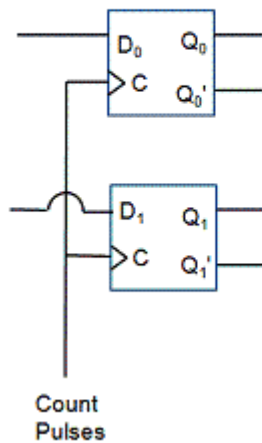


7. Design a synchronous mod-4 down-counter with counting sequence shown in the table to the left. Use 2 positive edge-triggered D flip-flops. The function table for this type of flip-flop is shown in the other table. Complete the circuit diagram begun in the answer box.

$Q_0$	$Q_1$
1	1
0	1
1	0
0	0

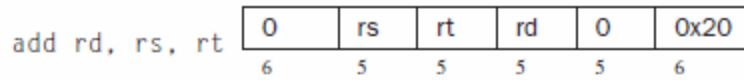
Inputs		Outputs	
D	C	$Q^+$	$Q'^+$
0	↑	0	1
1	↑	1	0
X	0	Q	Q'

7.

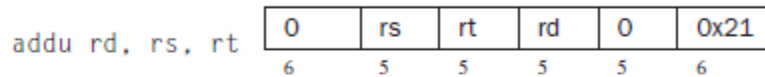


## Reference Sheet

### Addition (with overflow)

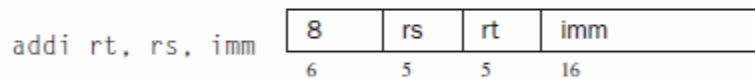


### Addition (without overflow)



Put the sum of registers *rs* and *rt* into register *rd*.

### Addition immediate (with overflow)

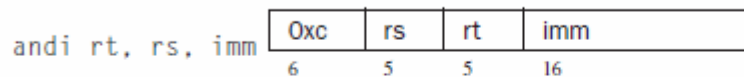


### Addition immediate (without overflow)



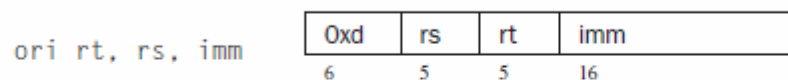
Put the sum of register *rs* and the sign-extended immediate into register *rt*.

### AND immediate



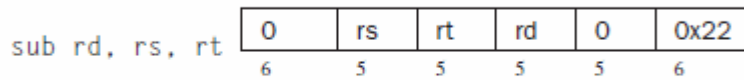
Put the logical AND of register *rs* and the zero-extended immediate into register *rt*.

### OR immediate

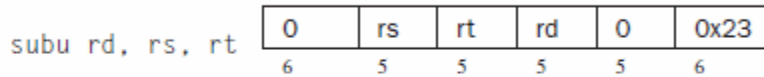


Put the logical OR of register *rs* and the zero-extended immediate into register *rt*.

### Subtract (with overflow)

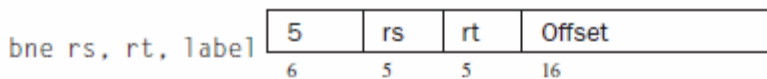


### Subtract (without overflow)



Put the difference of registers *rs* and *rt* into register *rd*.

### Branch on not equal



Conditionally branch the number of instructions specified by the offset if register *rs* is not equal to *rt*.

### Branch on equal zero

beqz rsrc, label *pseudoinstruction*

Conditionally branch to the instruction at the label if *rsrc* equals 0.

### Branch on greater than equal

bge rsrc1, rsrc2, label *pseudoinstruction*

### Branch on greater than equal unsigned

bgeu rsrc1, rsrc2, label *pseudoinstruction*

Conditionally branch to the instruction at the label if register *rsrc1* is greater than or equal to *rsrc2*.

### Jump



Unconditionally jump to the instruction at target.

### Load immediate

`li rdest, imm` *pseudoinstruction*

Move the immediate `imm` into register `rdest`.

### Load word

`lw rt, address`

0x23	rs	rt	Offset
6	5	5	16

Load the 32-bit quantity (word) at `address` into register `rt`.

### Load address

`la rdest, address` *pseudoinstruction*

Load computed `address`—not the contents of the location—into register `rdest`.

### Load halfword

`lh rt, address`

0x21	rs	rt	Offset
6	5	5	16

### Load unsigned halfword

`lhu rt, address`

0x25	rs	rt	Offset
6	5	5	16

Load the 16-bit quantity (halfword) at `address` into register `rt`. The halfword is sign-extended by `lh`, but not by `lhu`.

### Store halfword

`sh rt, address`

0x29	rs	rt	Offset
6	5	5	16

Store the low halfword from register `rt` at `address`.

### Store word

`sw rt, address`

0x2b	rs	rt	Offset
6	5	5	16

Store the word from register `rt` at `address`.